xCore

Control System User Manual

A Partner You can Rely on in Production

xCore

Control System User Manual

Control System Version: V3.1.1
[Remarks]
Document Version: [Status]

ROKAE (Shandong) Robotics Co., Ltd.

Shandong, China

Contents

1Manual Overview

1.1About the Manual

Thank you for choosing our ROKAE robot system.
The Manual describes the following instructions for the xCore control system:

- Composition and basic operation of the control system
- Programming and advanced parameter setting of the control system
- Option function introduction of the control system
- RL command set
- Error code list of the control system

Please read the Manual and other related manuals carefully before installing and using the robot system.
After reading, keep it properly for future reference.

1.2Target group

The Manual is intended for:

- Operators
- Product technicians
- Service technicians
- Robot programmers

Please ensure that the above personnel have acquired the knowledge of control system operation and have received our training.

1.3How to read the Manual

The Manual includes a separate safety section that must be read through before proceeding with any installation or maintenance procedures.

1.4Illustrations in the Manual

Due to product upgrades or other reasons, some figures in the Manual may differ from the actual products. However, the operating procedures are correct.
Also, figures from other models may be used to describe some general information.

1.5Contact

For information about the maintenance and repair of the robot, please contact our after-sales department or the local dealer.
Get the following information ready before contacting us:

- Controller model/serial number
- Robot model/serial number
- Software name/version
- Problems with the system

1.6Manual reading guide

The Manual is divided into the following chapters.

Chapter	Title	Content Summary
1	Manual Overview	General situation of the Manual.
2	Safety	Safety-related matters.
3	Glossary	Glossary involved in the Manual.
4	Basic Knowledge of Robot	Some necessary basic knowledge of robot.
5	Robot System Structure and Connection	Robot system structure and physical connection of different models.
6	HMI Introduction	Introduction to the overall layout and functions of HMI.
7	Basic Operation of the Control System	Basic operations commonly used in the control system; and demonstration of the most basic operations of industrial and collaborative robots by examples.
8	Programming	Detailed introduction to the use of programming module.
9	Setting	Detailed introduction to various settings of the control system.
10	Communication	Detailed introduction to communication functionality and usage.
11	Safety	Introduction to safety-related functions.
12	Process Package	Overview of process package.
13	Log	Introduction to log functionality and usage.
14	Options	Introduction to option module functionality and usage.
15	RL Commands	Detailed introduction to all RL commands.
16	Appendix	Details of user permission, and functionality and usage of end-effector handles of collaborative models.

17	Troubleshooting	Fault codes, handling list.

1.7Revision history of the Manual

Version No.	Date	Main Revision Content
V2.1	November 2023	New manual creation;
V2.2	March 2024	2.2 Added new function descriptions
V3.0	December 2024	3.0 Added new function descriptions
V3.1	July 2025	3.1 Added new function descriptions

1.8Related manuals

The xCore Control System not only offers impeccable core functionalities, but also encompasses an extensive array of advanced features. Regarding the extended functions, we provide the following documents. You can contact us if you need them.

Name	Introduction
xCore Control System User Manual	Describe the basic functions of the xCore Control System;
xVision User Manual	Describe the basic functions of xVision;
xCore_SDK_Android User Manual	Describe the use of xCore-SDK;
xCore_SDK_Python User Manual	Describe the use of xCore-SDK;
xCore_SDK_C++ User Manual	Describe the use of xCore-SDK;
xCore_RCI_User Manual	Describe the use of RCI;
User Manual for Conveyor Tracking Function	Describe the use of conveyor tracking function;
Tray Process Package User Manual	Describe the use of tray process package;
Stacking Process Package User Manual	Describe the use of stacking process package;
RokaeStudio User Manual	Describe the use of off-line programming software;
PV Typesetting Process Package User Manual	Describe the use of PV typesetting process package;
PV Inserting Process Package User Manual	Describe the use of PV inserting process package;
Profinet User Manual	Describe the use of Profinet bus;
EthernetIP User Manual	Describe the use of EthernetIP bus;
External Axis Track User Manual	Describe the use of external axis rack;

2Safety

2.1Introduction

This chapter describes the safety principles and procedures that need to be noted when using the robot.

The contents related to the design and installation of the external safety protection devices of the robot are not covered in this chapter. You can contact your system integrator to obtain such information.

2.2Safety responsibilities

ROKAE is dedicated to but not liable for providing reliable safety information. Even if all operations are carried out according to the safe operation instructions, we can not guarantee that our industrial robots will not cause personal and property losses.

2.3Safety symbols

There may be different degrees of danger when operating the robot in accordance with the Manual, so there will be a special safety symbol in the vicinity of dangerous operation instructions to remind the user to be careful. The contents include:

An icon that indicates safety level and the corresponding name, such as warning, danger, note, etc.;

A brief description given to illustrate the possible consequences if the operator fails to eliminate the danger;

The operating instructions on how to eliminate dangers.

2.3.1Safety level

Icon	Name	Explanation
	DANGER	Failure to follow the contents with this sign will cause serious or even fatal harm to the personnel, and also will/may cause serious damage to the robot.
	Warning	Failure to follow the contents with this sign may cause serious and even fatal personal injury, and also will cause great damage to the robot.
	Electric shock hazard	It indicates that the current operation may cause an electric shock hazard, which will result in a serious or even fatal injury.
	Caution	Failure to follow the contents with this sign may cause personal injury, and also may cause damage to the robot.
	ESD	It indicates that the components involved in the current operation are sensitive to static electricity. Failure to follow the contents with this sign may cause damage to the components.
	Note	It is used to prompt some important information or prerequisites.

2.3.2Hazard description

Icon	Name	Explanation
	Squeezing	Operators and maintenance personnel who enter the motion range of the robot during debugging, repair, overhaul, and tool clamping may be injured.
	Hand pinching	The maintenance personnel have a risk of hand pinching when approaching belt drive parts during the maintenance.
	Strike	Operators and maintenance personnel who enter the motion range of the robot during debugging, repair, overhaul, and tool clamping may be seriously injured.
	Friction	Operators and maintenance personnel who enter the motion range of the robot during debugging, repair, overhaul, and tool clamping may be injured.
	Parts flying out	Operators and maintenance personnel who enter the motion range of the robot during debugging, repair, overhaul, and tool clamping may be seriously injured if tools or work objects are ejected due to loose clamping.

	Fire	Electrical short-circuit and burning wires/devices may cause fire, which may result in serious injuries.
	Hot surface	During the maintenance and repair of the equipment, if maintenance personnel touch the robot's hot surface, they may be burned.

2.4Safe stop

There are three ways to stop the robot: STOP 0, STOP 1, and STOP 2.
Safe stop refers to a stop triggered by the safety controller, which only supports STOP 0 and STOP 1, while STOP 2 can only be triggered by the control system.

STOP 0	As the stop method of the highest safety level, STOP 0 cuts off the power supply of motors and closes the band-type brakes of all joints immediately after it is triggered. During the stopping process, the robot is uncontrolled, so it may deviate from the programmed path after it is stopped. The safe stop of manual mode belongs to STOP 0. STOP 0 supports deceleration to a complete stop at maximum capability.
STOP 1	Once STOP 1 is triggered, the control system immediately executes the deceleration process along the programmed path. Thereafter, whether the robot comes to a complete stop or not, the safety controller will cut off the power supply of motors and close the band-type brakes of all joints. Since the stop is controlled, in most cases, the robot will finally stop on the programmed path. Therefore, This emergency stop method provides the best protection for nearby equipment. The safe stops arising from the opening of the safety gate/safety grating in automatic mode and pressing of the emergency stop button in automatic mode are STOP 1. STOP 1 supports two modes of stopping: deceleration to a complete stop at maximum capability, and normal planned stopping.
STOP 2	Once STOP 2 is triggered, the control system immediately executes the deceleration along the programmed path until the robot stops completely. The power supply of the motors remains on and the band-type brakes are still open, while the robot stays in the current position. Stopping the robot through the stop button on the HMI and external signals trigger the stopping mode STOP 2.

Note

1. Emergency stop is only used to stop the robot immediately in dangerous circumstances. Emergency stop shall not be used for normal stops. Otherwise, extra and unnecessary wear may be caused to the brake and transmission system, which will eventually reduce the robot's service life.

2.5Safety devices

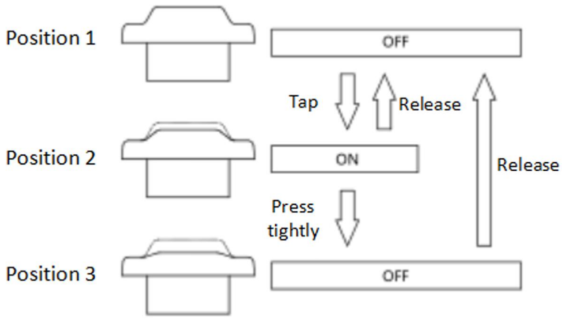
2.5.1Emergency stop (E-stop)

Emergency stop buttons are for manually triggering an emergency stop, and most of them are red in the shape of a mushroom head. In general, a yellow substrate, protective casing, or warning sign is also attached to the emergency stop button.

The emergency stop button is mechanically locked through the safety lock mechanism when it is pressed and must be reset through manual release. Most emergency stop buttons are released by rotation along the rotation direction indicated on the button surface. Additionally, some buttons also support releasing by upward pulling.

2.5.2Enabling device

The enabling device is a special switch with 2 segments of pressing and 3 positions, which is also called three-position enabling switch (hereinafter referred to as "enabling switch"), and is used to control the on and off of the power supply of the robot in the manual mode, thus realizing the motion enabling of the robot. The motor power is powered on only when the enabling switch is pressed and kept in the middle so that the robot is in a state ready for motion. Releasing or pressing the switch all the way down will cut off the motor power.



> **Note**
>
> The yellow button on the Handheld Enabling Device is an enabling switch. When the enabling switch is pressed and held in the middle position, the power supply of the motor is turned on and automatically enabled, and the system is in a power-on state, and you can jog the robot or execute a program. Releasing or pressing the switch all the way down will cut off the motor power and make the system return to the power-off state.
> In order to use the Teach Pendant safely, the following requirements must be observed:
> 1. The enabling switch shall function properly in any circumstances;
> 2. The enabling switch shall be released immediately when no robot motion is required during programming or debugging; and
> 3. Any person who enters the robot's working space must carry a handheld enabling device to prevent others from starting the robot without the knowledge of the involved personnel.

> **Warning**
>
> It is strictly prohibited to use external devices to keep the enabling switch locked or stopped in the middle position!

2.6Safety precautions in various situations

2.6.1Safety precautions in manual mode

In manual mode, the motion of the robot is under manual control. You can only Jog the robot or execute a program when the enabling
switch is held in the middle position. The manual mode is used during the programming and debugging of the robot, as well as the commissioning of the workstation.

2.6.1.1Speed limit in manual mode

The motion velocity of the robot end-effector is limited to less than 250 mm/s in manual mode, that is, no matter when you Jog the robot or execute a program, regardless of the set velocity in the program, the maximum motion velocity of the robot end-effector will not exceed 250 mm/s.

2.6.1.2Bypassing external safety signals

In manual mode, signals of external safety devices such as the safety door and safety grating will be bypassed, i.e. in manual mode, the
system can still perform motor-enabling operations even if the safety door is opened, and the system will not prompt the safety door opening information for the convenience of debugging.

2.6.2Safety precautions in auto mode

The auto mode is used for robot program running during the formal production process. In auto mode, the enabling switch will be bypassed so that the robot can run automatically without manual intervention.

2.6.2.1Activating external safety signals

In auto mode, external safety devices such as the safety door and safety grating will be activated. When the safety door is opened, the motor power supply will be cut off and
the band-type brake will be closed.

2.6.3Safety requirements for installation and operation

- Handling and installation of the robot equipment must be carried out according to the methods described in the Manual. Otherwise, the robot may fall due to misoperation, thus leading to personal injury and death or equipment damage.
- When the robot equipment is put into use for the first time after installation, it is necessary to run it at low velocity first and then gradually increase the velocity rather than running it at high velocity from the start.
- By default, program and system variable information is stored in the controller storage device. In order to prevent data loss caused by accidents, it is recommended that the user makes data backups regularly.

2.6.4Safety requirements for debugging

Debugging shall be carried out outside the safeguarded space as much as possible. When debugging must be carried out inside the safeguarded space, special attention shall be paid to the following matters:

- Carefully check the situation inside the safeguarded space and enter into it only after confirming there is no danger.
- Confirm the positions of all debugging personnel inside the safeguarded space.
- Confirm the status of the entire system before proceeding with the work.
- Make sure that the emergency stop button can be pressed whenever necessary.
- Run the robot at low velocity.
- When the above debugging is finished, the debugging personnel must stay outside the safeguarded space.

2.6.5Safety requirements for maintenance

- It is necessary to carefully check the situation in the safeguarded space and confirm that there is no danger before entering the safeguarded space. The positions of all maintenance personnel in the safeguarded space shall be confirmed.
- When the power supply is switched on, some maintenance operations may pose a risk of electric shock. Therefore, the power supply of the robot equipment and system needs to be cut off before the maintenance is carried out.
- During the maintenance, other personnel shall be prevented from switching on the power supply accidentally.
- To avoid unnecessary personal injury or adverse impact on the equipment, you shall not place any part of your body on any part of the robot equipment during the operation.
- Appropriate lighting shall be provided during the maintenance.
- In case of part replacement, make sure to use parts specified by ROKAE. Otherwise, the robot equipment may be damaged.
- Parts removed during the replacement (such as screws) shall be correctly installed back to their original positions. If you find the parts not enough or redundant, you need to confirm again and make sure to install them correctly.

2.6.6Safe handling on the production line

In most cases, the robot is just a part of the production line. Therefore, robot faults will not only affect the robot itself, but may also affect the entire production line. Likewise, problems with other parts of the production line may also affect the robot. For this reason, a fault remedial plan shall be designed by personnel who are very familiar with the entire production line to improve the safety of the whole system.

- Pay attention to other devices that interact with the robot

For example, when a robot needs maintenance, you must first remove it from the production line, as well as remove other devices interacting with the robot, such as the robot responsible for feeding materials to the above robot.

- Pay attention to other running devices around the robot

For example, robots on the production line need to grab work objects from the conveyor belt. Therefore, when a robot fails, in order to guarantee uninterrupted
production, the conveyor belt may keep running while the robot is being repaired. The robot maintenance personnel must pay extra attention to safety, give advance consideration to the risks that might arise from the running conveyor belt, and develop detailed safety measures for working in such an environment.

2.6.7Safe handling of fire

It is required to keep calm when a fire hazard is imminent or has not yet begun to spread. You can use on-site fire-extinguishing devices to put out the flame. It is strictly prohibited to use water to put out a fire caused by a short-circuit fault.

Warning

The fire-extinguishing devices in the working space of the robot shall be self-prepared by the user, and the user shall choose appropriate fire-extinguishing devices according to the actual situation.

If the fire has spread and is beyond control, the workers on the site shall notify other workers immediately to give up their personal belongings and evacuate immediately through emergency exits rather than try to put out the fire. DO NOT use the elevators, and be sure to inform the fire brigade during evacuation. If one person's clothing catches fire, ask him/her not to run but to lie flat on the ground immediately, and put out the fire using clothes or other suitable items and methods.

2.6.8Safe handling of electric shock

When someone gets an electric shock, do not panic and cut off the power supply immediately.
Appropriate methods and measures shall be adopted without hesitation according to specific site conditions. Generally, there are several methods and measures as follows:

1. If the power switch or button is very near to the point of the electric shock, it shall be switched off at once to cut off the power supply.
2. If the power switch or button is far away from the point of the electric shock, it is recommended to use insulated pliers or an axe, knife, and shovel with a dry wooden handle to cut off the live wire on the mains side (power supply side), and the cut wire must not contact with the human body.
3. If the wire is over or under the body of the victim, it is suggested to use a dry stick, board, bamboo pole, or other tools with an insulated handle (by gripping the insulated handle) to remove the wire. No metal bar or wet object shall be used to prevent the rescuer from also getting an electric shock.

Handling of the victim after separation from the power source

1. If the victim is conscious, make him/her lie on the back, keep a close watch over him/her, and let him/her not stand or walk for the time being.
2. If the victim is confused, make him/her lie on the back to keep the airways open, and call the victim or pat him/her on the shoulder at an interval of 5 seconds to judge if he/she loses consciousness completely. Do not call the victim by shaking his/her head. Meanwhile, contact the hospital as soon as possible.
3. If the victim loses consciousness, confirm his/her respiratory conditions and heartbeat within 10 seconds. If neither breath nor arterial pulse is sensed, the victim may have a cardiac arrest and shall be given immediate first aid treatment by cardiopulmonary resuscitation.

3Glossary

This chapter briefly introduces some terms used in the Manual.

Glossary	Definition
RobotAssist	A host computer software of ROKAE xCore Control System, with functions such as robot motion control, programming, parameter configuration, and status monitoring, can run on such devices as xPad2 Teach Pendant and PC.
HMI	Human Machine Interface.
HMID	Human Machine Interface Device.
RC	Robot Controller.
RCI	Rokae Control Interface, external control interface for ROKAE robots, with real-time underlying control supported.
SDK	Software Development Kit, which will gradually replace RCI to realize underlying control of robots through C++ and other languages.
Project	A management collection of programs, tasks, and other objects that control the operation of the robot; data objects of a project can be exported and reused in other projects or robots.
Task	In xCore, it is as it suggests.
Module	In xCore, we refer it to as program module.
Elbow	It is the angle between the arm plane and the reference plane. The arm plane refers to the plane formed by the robot's lower arm and upper arm, and the reference plane refers to the arm plane formed when the three axes are set to zero and the end-effector reaches the predefined pose.
RL	Rokae Robot Language. It provides various commands to assist the robot in building projects.
xPad2	Teach Pendant.
RSC	Robot Safety Controller.
JOG	Inching.
Null-space motion	For robots with redundant degrees of freedom, null-space motion can be utilized to move the robot's joints while keeping the end-effector stationary.
PERS variable	Persistent variable: During the execution of a program, if the value of this type of variable changes, the variable will be automatically amended from the initial value to the current value, thus achieving the effect of "Persistent" storage.

4Basic Knowledge of Robot

4.1Introduction to this chapter

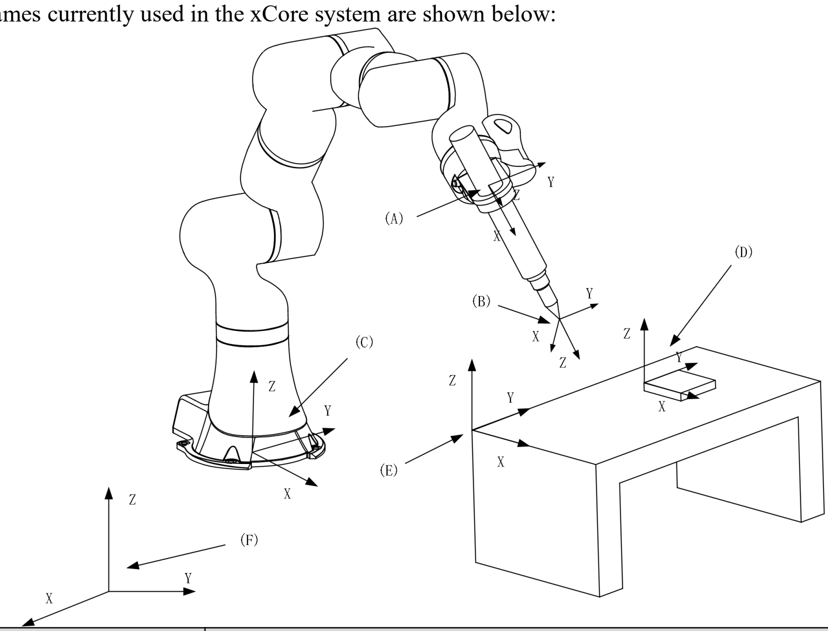
This chapter introduces the basic knowledge of the robot. Familiarity with the contents of this chapter will help to better understand and master the use of the control system and robot.

4.2Frame

Any object (tool, work object, etc.) in space has six degrees of freedom (DOF): three translational degrees of freedom and three rotational degrees of freedom. The three translational degrees of freedom constitute the position; the three rotational degrees of freedom constitute the orientation; and these six degrees of freedom are collectively referred to as pose. The pose of an object can be described by the frame attached to it, generally using the Cartesian frame (hereinafter referred to as the "frame").

The robot is a mechanism with multiple degrees of freedom. Its typical operation mode is to use a tool attached to the flange to execute the movements relative to external work objects. This mode of operation can be described through the frame and its relative motion.

The frames currently used in the xCore system are shown below:



No.	Frame	Meaning
A	Flange frame	Defined in the center of the robot flange; The flange frame is defined relative to the base frame;
B	Tool frame	Defined at the end of the tool; When the tool is a handheld tool (ordinary tool), the tool frame is defined relative to the flange frame; and when the tool is an external tool, the tool frame is defined relative to the user frame;
C	Base frame	Defined in the center of the robot base; The base frame is defined relative to the world frame. Note: If the robot is not installed in the default way, such as upside-down or slanted installation, the base frame needs to be calibrated first;
D	Work object frame	Defined in the work object; When the work object is an external work object (ordinary work object), the work object frame is defined relative to the user frame; and when the work object is a handheld work object, the work object frame is defined relative to the flange frame;
E	User frame	The user frame is used as a reference when defining the work object frame, and it cannot be used separately; The user frame is defined relative to the world frame;
F	World frame	The world frame is generally used as the reference frame and has no specific position. When a single robot is installed normally, it coincides with the robot base frame by default; and when multiple robots or external devices are involved in collaboration, the unification of motion reference can be achieved by unifying their world frames into a same frame;

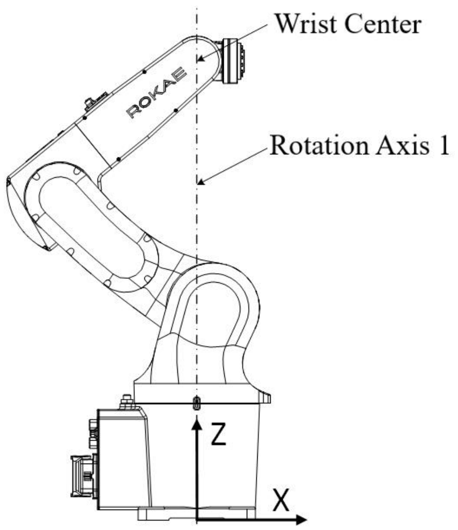
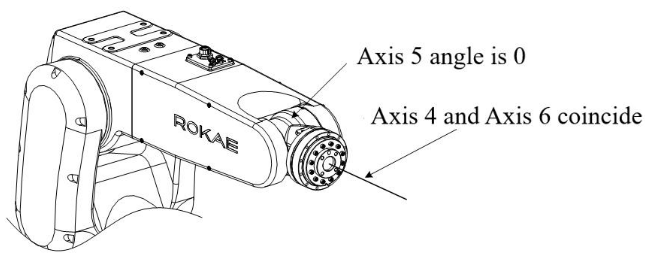
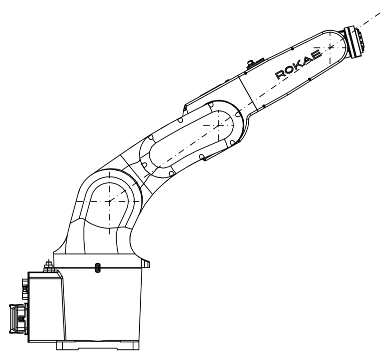
4.3Singularity

There are a few special poses in the robot's working space that the robot can arrive at using a myriad of different joint configurations. Such poses are called singularities. Singularities may cause problems to the control system when calculating joint angles based on Cartesian space pose.
There are no singularity problems when the robot performs joint motion.
When the robot executes a Cartesian space trajectory near a singular point, the speed of some joints may be very fast, potentially leading to an error report and subsequent cessation of the robot's operation.

4.3.1Typical singular positions of robots

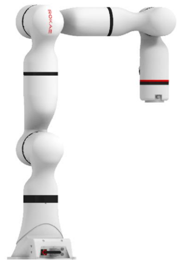
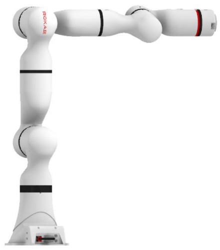
Robots with different configurations have different singular positions. Typical singular positions of some robot configurations are described below.

4.3.1.1Singular position of the six-axis industrial robot

Singularity	Configuration	Explanation
Shoulder singularity		When the robot's wrist center is located on the Axis 1.
Wrist singularity		When the Axis 4 and Axis 6 coincide (Axis 5 angle is 0).
Elbow singularity		When the wrist center, Rotation Axis 2, and Rotation Axis 3 are in a straight line.

4.3.1.2Singular position of ER PRO collaborative robot

The singularity of ER RPO collaborative robot can be divided into the following cases:

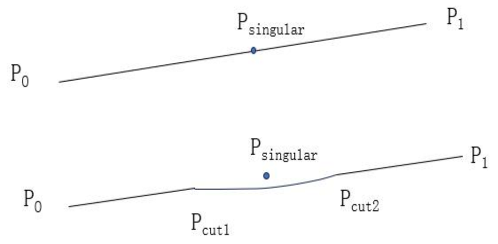
Singularity	Configuration	Explanation
Axis 2 singularity		When the angle of Axis 2 is equal to 0°, the robot is unable to distinguish between the angles of Axis 1 and Axis 3 when solving the inverse kinematics.
Axis 4 singularity		When the angle of Axis 4 is 0°, the robot is restricted to move in the direction parallel to Axis 3 or 5. This singularity causes the robot to lose one degree of freedom at the root of the wrist (the root of the wrist is unable to move along the axis of the arm). In this case, the Axis 3 and Axis 5 positions cannot be obtained through inverse kinematics.
Axis 6 singularity		When the robot's angle of Axis 6 is equal to 0°, the robot is unable to distinguish between the angles of Axis 5 and Axis 7 when solving the inverse kinematics.
Wrist center singularity		When the wrist center is directly above Axis 1, the robot cannot accurately determine the Axis 1 angle when solving the inverse solution.

4.3.2Singularity avoidance

The singularity problems stem from the robot configuration and cannot be completely avoided. In practical task programming, if the robot must pass through the vicinity of the singularities, it can be considered to reduce some constraints (such as orientation or path accuracy) to make the robot pass through the singularities smoothly.

xCore Control System also provides a variety of singularity avoidance methods:

Method I	Axis 4 Locking	Before enabling the singularity avoidance method, it is necessary to move the robot's Axis 4 to 0° or ±180°. After this singularity avoidance method is activated, the robot will keep Axis 4 immobile, and perform orientation interpolation in a specialized manner on the premise of ensuring the accurate TCP position. You can refer to the sections of RL command SingAreaLockAxis4 on/off and jog mode.
Method II	Cartesian Sacrifice Orientation	After this singularity avoidance method is enabled, the wrist-type singularities can be passed through by changing the morphology of the robot. Note: With this function, the wrist morphology of the robot during motion may differ from that taught during teaching (not only at the teaching

<table>
<tr><td></td><td>Interpolation</td><td>points where singularities are traversed, but also potentially at subsequent teaching points).</td></tr>
<tr><td>Method III</td><td>Joint Space Interpolation</td><td>After this singularity avoidance method is enabled, the control system will perform singularity detection on the subsequent Cartesian trajectories until this function is turned off.
For Cartesian trajectories that do not contain singularities, the robot moves along the ordinary trajectory. When a Cartesian trajectory contains a singularity, the control system detects it and splits the original trajectory P_0P_1 into three segments: P0Pcut1, Pcut1Pcut2, and Pcut2P1. The segments P0Pcut1 and Pcut2P1 continue to follow the original trajectory and undergo Cartesian interpolation, while the segment Pcut1Pcut2 adopts joint space interpolation to navigate around the singularity. The three trajectory segments are smoothly transitioned using a turning zone, as shown in the following figure.


Singularity avoidance: Schematic diagram of joint space trajectory interpolation
For specific use, you can refer to the RL command SingAreaJointWay.</td></tr>
</table>

Note:

- Near singularities, the movement amplitude of a robot's joints tends to be significant. Therefore, you need to confirm whether it is necessary to use a singularity avoidance command. It is preferred to avoid singularities by altering the trajectory points.
- When using a singularity avoidance command, it is recommended to first confirm that the robot's trajectory with the singularity avoidance command enabled satisfies the operational requirements before performing the official operation.
- Near singularities, the movement amplitude of a robot's joints tends to be significant, so you need to confirm the surrounding environment before using it.
- In view of the above reasons, if the robot operating point or program run logic is affected by external signals, it is recommended to carefully confirm the program logic and trajectory before use.

The specific characteristics of the above three singularity avoidance methods are as follows:

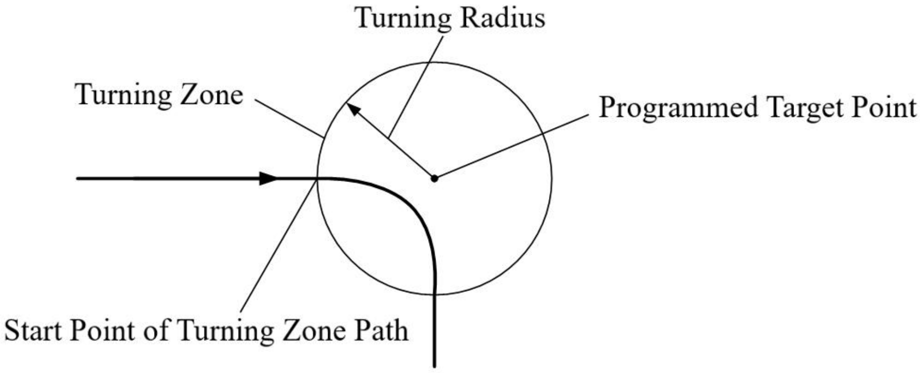
Mode and Feature	Axis 4 Locking	Cartesian Sacrifice Orientation Interpolation	Joint Space Interpolation
Motion feature	1. Before enabling this command, it is necessary to first move the robot's Axis 4 to 0° or ±180°. 2. After this command is enabled, the robot keeps Axis 4 immobile for subsequent motion commands.	1. After enabling this command, the robot will change the wrist morphology, resulting in a change in the tool orientation to allow passage through trajectories with wrist singularities. 2. The wrist morphology during motion may sometimes differ from the taught morphology, not only at the teaching points where singularities are traversed, but also potentially at subsequent teaching points.	1. The sections of the trajectory before and after the singularity adopt joint interpolation (MoveAbsJ) for movement, while the remaining sections retain the original Cartesian trajectory's movement method. A turning zone is used for a smooth transition between the above two movement methods.
Trajectory form change	1. Position trajectory remains unchanged. 2. Special orientation interpolation methods are employed.	1. Position trajectory remains unchanged. 2. Orientation interpolation method is altered.	1. Position trajectory is altered. 2. Orientation trajectory is altered.
Reachability of target points	The robot's workspace is partially reachable. Specifically, partial target points are reachable when the Axis 4 is set at 0° or ±180°.	The robot's workspace is partially reachable.	The robot's entire workspace is reachable.
Turning zone feature	1. Generating turning zones between similar trajectories is	1. Generating turning zones between similar trajectories is supported,	Generating turning zones is supported between singularity avoidance trajectories, as well as between singularity avoidance

	supported, specifically, between the motion trajectories when SingAreaLockAxis4 is on and off. 2. Generating turning zones between different types of trajectories is not supported, that is, generating turning zones is not supported between motion trajectories that precede or follow the activation of the SingAreaLockAxis4 on/off command.	specifically, between the motion trajectories when SingAreaWrist is on and SingAreaWrist4 is off. 2. For different types of trajectories, generating turning zones is supported between joint space trajectories and singularity avoidance trajectories, while generating turning zones is not supported between Cartesian space trajectories and singularity avoidance trajectories.	trajectories and ordinary trajectories.
Lookahead feature	The lookahead mechanism is interrupted. Specifically, the SingAreaLockAxis4 on/off serves as a blocking command, and the control system will only continue to parse singularity avoidance commands after the robot has completed the trajectory preceding the activation of the SingAreaLockAxis4 on command. Similarly, the control system will resume parsing subsequent commands only after the robot has executed the motion command preceding the activation of the SingAreaLockAxis4 off command.	The lookahead mechanism is not interrupted.	The lookahead mechanism is not interrupted.
Whether the singularity avoidance is mandatory	Mandatory. After the axis locking, all Cartesian motion commands are interpolated by the special interpolation form corresponding to the locked axis until the axis locking function is turned off.	Mandatory. After the sacrifice orientation singularity avoidance is enabled, all Cartesian motion commands are interpolated by the special interpolation form corresponding to the locked axis until the sacrifice orientation singularity avoidance function is turned off.	Not mandatory. After this singularity avoidance is enabled, the control system automatically detects whether there are singularities in each Cartesian motion trajectory. The trajectory that only contains singularities will adopt special forms of interpolation, while the trajectory that does not contain singularities still employs the motion form of ordinary trajectories for interpolation. Note: The calculation amount of the control system will increase after this function is enabled, so this function is not enabled generally unless necessary.
Applicable scenarios	1. The trajectory where the flange remains parallel to the base or moves along the z-axis of the base. 2. During the robot's movement, it is permissible to keep Axis 4 immobile, such as stacking.	Do not mind the orientation accuracy of trajectories, nor how the robot reaches the target points, only pursuing the ability to reach the Cartesian position of the target points.	Do not mind the position accuracy and orientation accuracy of trajectories, nor how the robot reaches the target points, only pursuing the ability to reach the target points.
Supported model	Industrial standard six-axis series (XB, NB model), collaborative	Industrial standard six-axis series (XB, NB model), collaborative xMate	Industrial standard six-axis series (XB, NB model)

	xMate CR/SR	CR/SR	
Whether Jog is supported	Supported	Not supported	Not supported

4.4Turning zone

The motion of a robot typically involves sequentially executing multiple trajectories programmed and set by the user. Usually, these trajectories are not smoothly connected, and there are various "spikes" between them. The presence of these "spikes" forces the robot to first stop at the end of a trajectory before starting the next trajectory. To enable continuous motion between trajectories, it is necessary to eliminate such "spikes", and different trajectories can be smoothly connected by generating turning zones. See the following figure:



The turning zone type includes Cartesian space turning zone and joint space turning zone. For the detailed definition and specific parameters of the turning zone, please refer to the section below, "RL Command"-"zone".

4.5Lookahead mechanism

Lookahead means that the control system handles the subsequent program commands in advance when the robot is executing the current command during robot movement.
The introduction of the lookahead mechanism can be advantageous in the following aspects:

- Obtain the speed of the front trajectory, the acceleration requirements, and the constraints of the robot itself, so as to plan the control strategy for optimal performance;
- Plan the turning trajectory of the turning zone according to the settings of the programmed turning zone;
- Acquire an abnormal state near the soft limit/boundary and singular points, etc., so that it can be handled in advance;

For a more detailed introduction to the lookahead mechanism, refer to the section below, "Programming"-"About RL program"-" RL program debugging".

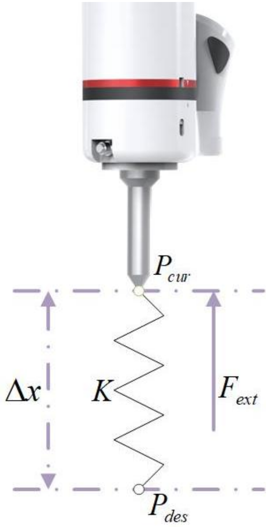
4.6Force control

4.6.1Introduction to force control

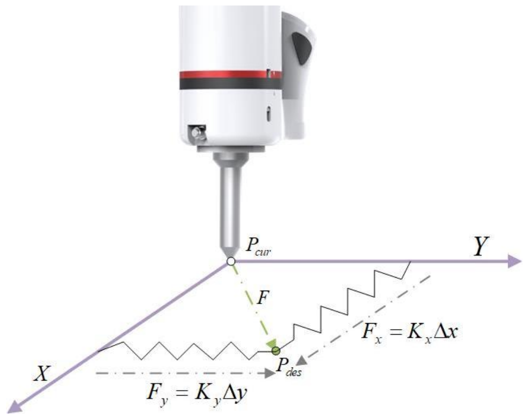
The robot force control is a process of interaction between the robot end-effector and forces in the external environment. During non-contact robot motion control, only the position control process (velocity and accuracy) is considered. When there is contact with the environment, pure position control requires very high accuracy of the robot and the environment to avoid damage to the robot and the environment caused by contact forces resulting from positional deviations.
Unlike pure position control, robot force control introduces a force/torque feedback loop when interacting with the environment. The loop is used to change the motion characteristics of the robot, which enables dynamic interaction with the external environment. When there is deviation or uncertainty between the robot and the external environment, the force control will intelligently adjust the preset position trajectory to eliminate the internal force caused by the position deviation and ensure a smooth and safe interaction process

4.6.2Impedance control

Compared with traditional industrial robots, xMate collaborative robot is equipped with torque sensors in its joints, which enable it to sense joint torque precisely. The joint torque information allows the xMate collaborative robot to achieve force control through impedance, making the robot have compliant interactive behaviors. This means the interaction between the robot and the environment is like a virtual spring stiffness and damping system. At this point, the robot is sensitive to external forces, which can cause the robot to deviate from a predetermined trajectory. When the external forces disappear, the robot can rebound to some extent.

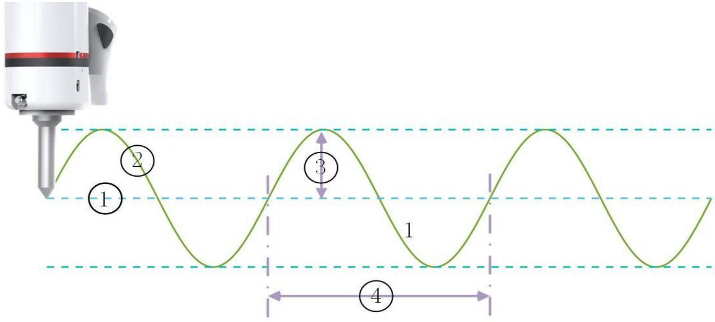


In the process of impedance motion, the actual position of the robot will deviate from the desired position when affected by the external forces in the environment. The deviation depends on the impedance stiffness and the external forces, and it can be calculated through the ratio between the external force and the impedance stiffness. As shown above, in the impedance control mode, with impedance stiffness set to K and under the action of external force Fext, the robot's current position Pcur will deviate from the desired position Pdes, and the position deviation is Δx. The impedance force caused by this deviation and the external force will eventually reach an equilibrium.
The impedance stiffness in each direction can be set individually, and the impedance force in each direction is the product of the impedance stiffness and the position deviation in this direction. The impedance forces in all directions are ultimately combined to form the total impedance force. In the figure below, the robot's current position Pcur deviates from the desired position Pdes due to the action of external forces in the impedance mode. In the X and Y directions, the deviations are Δxand Δy, the impedance stiffnesses are Kx and Ky, and the impedance forces are Fx and Fy, respectively. The total impedance force F = Fx + Fy.



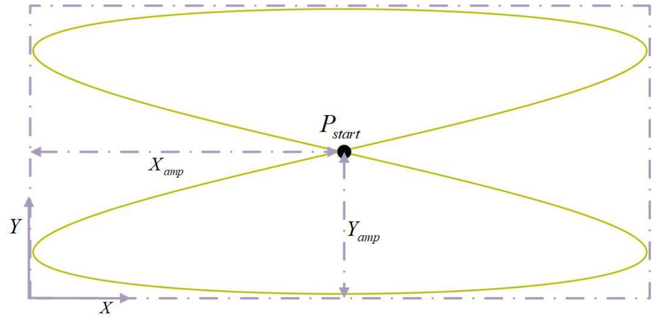
4.6.3Force control search

When assembling work objects, humans can feel the change in force by hand. If an obstruction (a work object is stuck) is detected, humans will try shaking to ensure a smooth installation. Force control allows the robot to do the same thing, i.e. overlay. The robot supports sine overlay rotating around an axis and Lissajous overlay within a plane. Overlay is an additional movement superimposed on the robot's predetermined motion. Overlay allows the robot to exhibit a certain degree of shake, enabling it to better overcome obstacles during assembly.
Below is a sine overlay:



1	Desired trajectory	2	Actual trajectory (desired trajectory + overlay)
3	Overlay amplitude	4	Overlay period

Lissajous overlay refers to the application of sine search motions in two perpendicular directions within a plane, and the frequencies of the two overlays are often proportional. For example, below shows the Lissajous overlay in the XY plane, where the frequency ratio of x- and y-direction overlay are 2:1. The center point Pstart is the desired pose, Xamp is the amplitude of the x-direction overlay, and Yamp is the amplitude of the y-direction overlay.

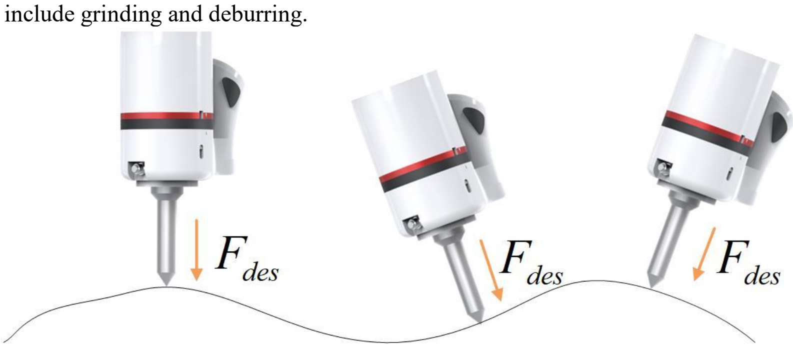


4.6.4Force control application

The application scenarios of force control for industrial robots can roughly be divided into two categories: constant force tracking and force-controlled assembly.

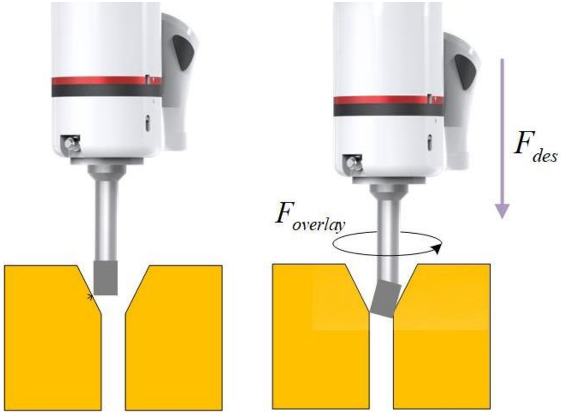
4.6.4.1Constant force tracking

Below is a constant force tracking scenario. The robot ensures a constant contact force Fdes with the surface, while the robot can conform to the surface curve. Main applications of constant force tracking include grinding and deburring.



4.6.4.2Force-controlled assembly

If pure position control is used during the assembly, the robot may easily collide with the work object due to position and modeling errors, which can cause damage to the work object or the robot. But with force control, the robot will try to overlay (shake) to overcome the obstruction when it senses an external force over the limit (work object jamming), thus allowing smooth work object installation. As shown below, the position control on the left results in a collision during assembly, while the force control on the right pushes the robot into the assembly hole through the desired force Fdes, and the jamming is prevented through overlay Foverlay.


F_{des}
$F_{overlay}$

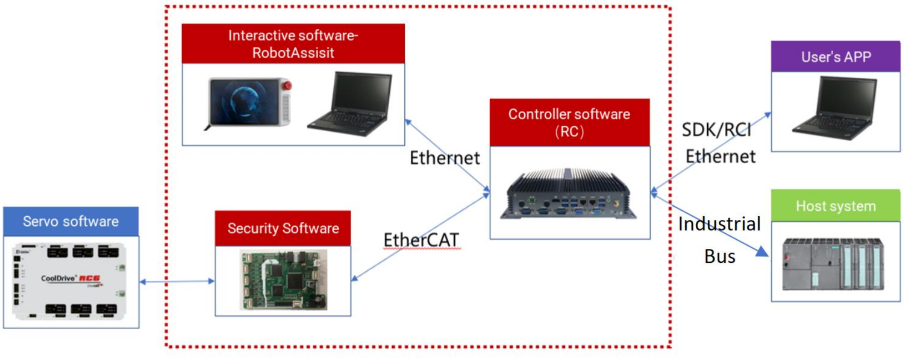
5Robot System Structure and Connection

5.1Introduction to this chapter

ROKAE has several series of robots. This chapter mainly introduces the system structure and connection mode of different series of robots to deepen users' understanding of robot systems.
Users can optionally read the contents of this chapter based on the model they use.

5.2Control system structure

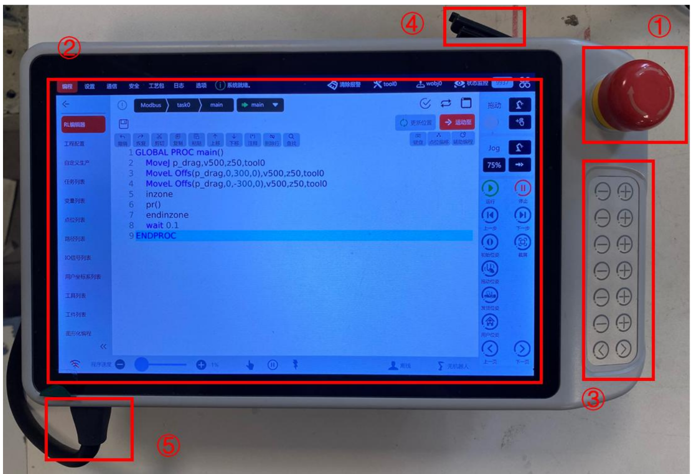
xCore control system is based on CS architecture, including HMI software (RobotAssist) and controller software RC.

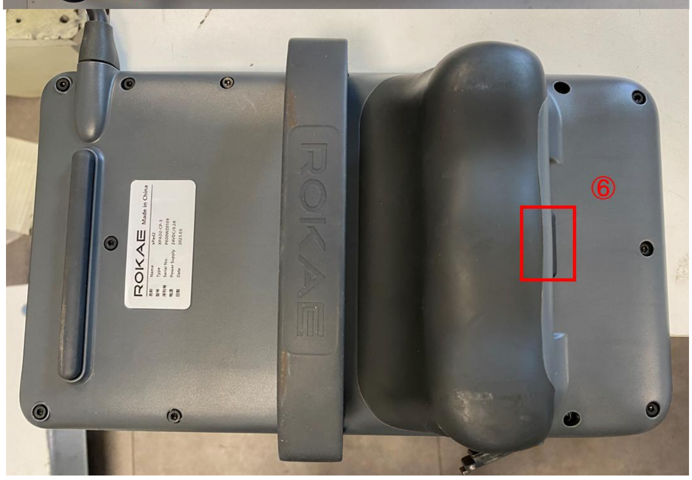


Control System Components

5.2.1xPad2 Teach Pendant introduction

The buttons and their functions of xPad2 Teach Pendant are described below.



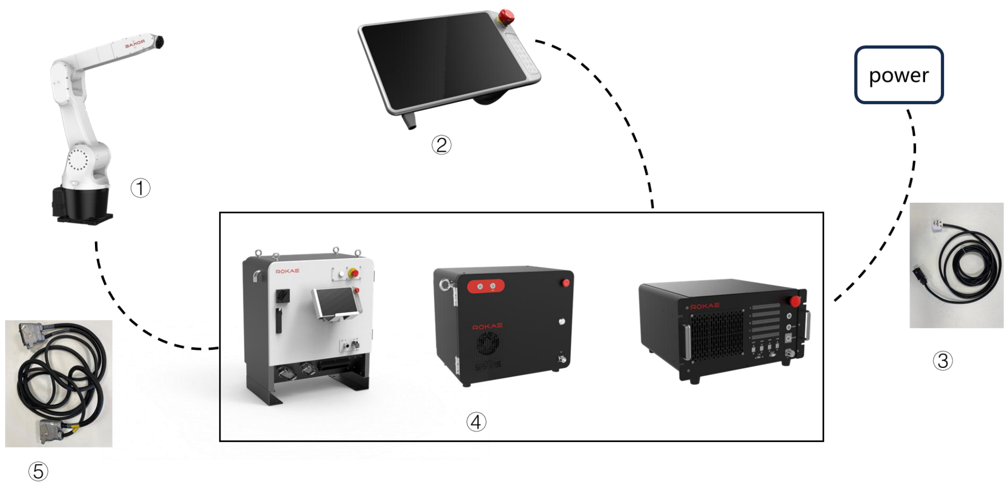


①	Emergency stop button;
②	Touch screen;
③	Physical buttons;
④	USB drive interface;
⑤	Connecting cable, for connecting with control cabinet or robot;
⑥	Three-position enabling switch;

5.3Industrial robot system composition

This chapter mainly introduces the structure, wiring, and power-on start-up methods of industrial robots. There may be certain differences in the robot body and control cabinet depending on the robot's specific model. For more information, please refer to the *XBC5 Series Controller (xCore System) Product Manual*.

The main structure and wiring relations of an industrial robot system are shown in the figure below, mainly including: robot body, Teach Pendant, control cabinet, relay cable, and power cord.

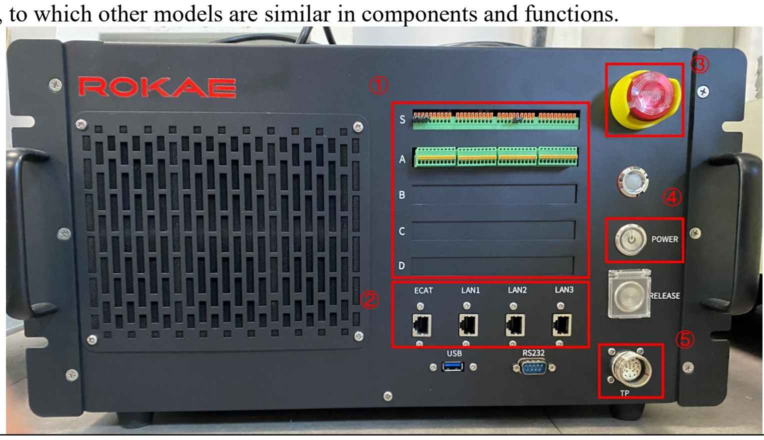


①	Robot body;
②	Teaching pendant;
③	Power cable;
④	Control cabinet;
⑤	Connecting cable, for connecting with control cabinet or robot;

5.3.1XBC5 series controller introduction

XBC5 series control cabinets include three models: XBC5, XBC5-E, XBC5-M.

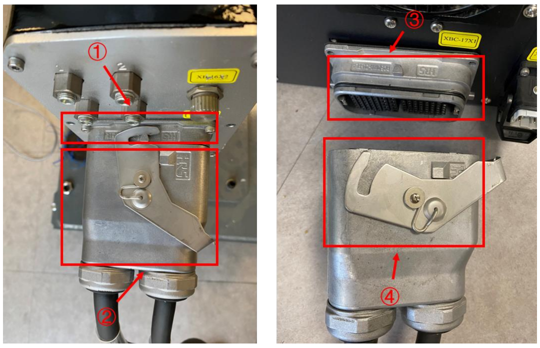
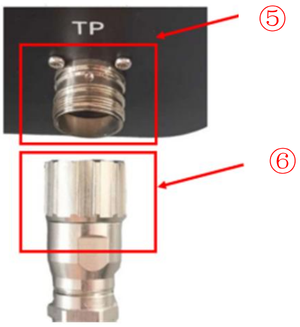
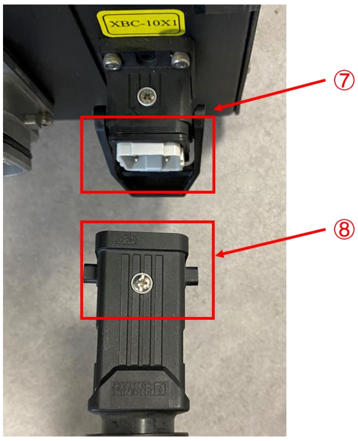
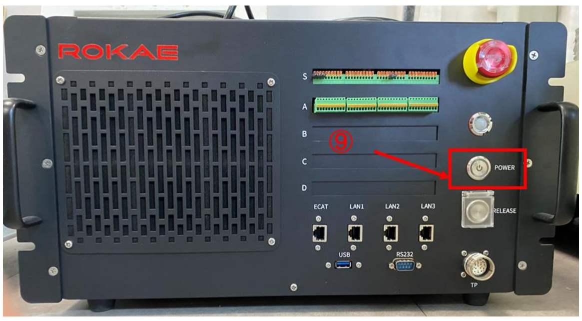
Taking XBC5-M as an example, the main components and functions of the cabinet are briefly introduced, to which other models are similar in components and functions.



①	Security/Universal IO wiring terminals;
②	Network interface: including debugging interface, EtherCAT expansion network

	interface, and visual interface;
③	Emergency stop switch: Used to control the motor's band-type brake in case of emergency;
④	Power switch: Used to control the startup & shutdown of the robot;
⑤	Teach Pendant wiring port: Used to connect xPad2;

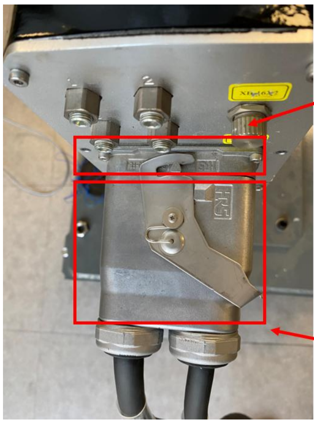
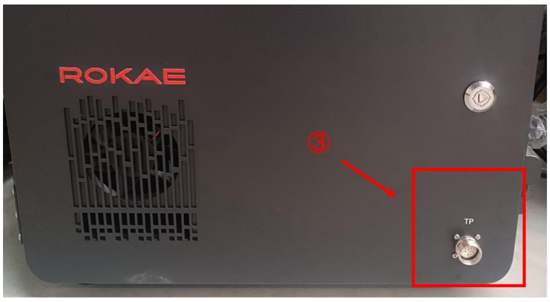
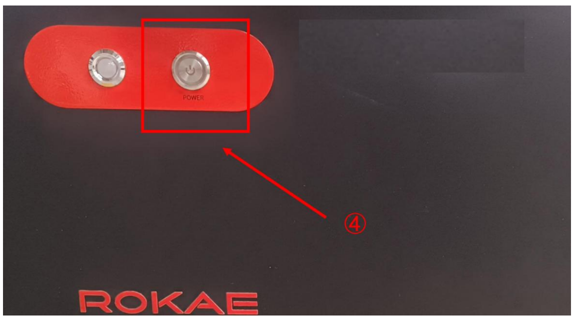
5.3.2XBC5-M controller wiring, power-on, and start-up

<table>
<tr><th>Step</th><th>Graphical Representation</th><th>Explanation</th></tr>
<tr><td>1. Connect the robot body with the controller through the relay cable;</td><td>
</td><td>As there is a difference between the two ends of the relay cable's heavy-duty connector, please confirm before connecting.</td></tr>
<tr><td>2. Connect the Teach Pendant with the controller according to the figure;</td><td>
</td><td></td></tr>
<tr><td>3. Connect the power cord with the controller;</td><td>
</td><td>The interface of the power cord on the side of the controller is designed with a buckle.</td></tr>
<tr><td>4. Start up the control cabinet after powering it on.</td><td>
</td><td>After the power cord is powered on and the POWER button is pressed, the xPad2 Teach Pendant will automatically start up.</td></tr>
</table>

①	Robot body side loading plug;
②	Trunk side load plug;

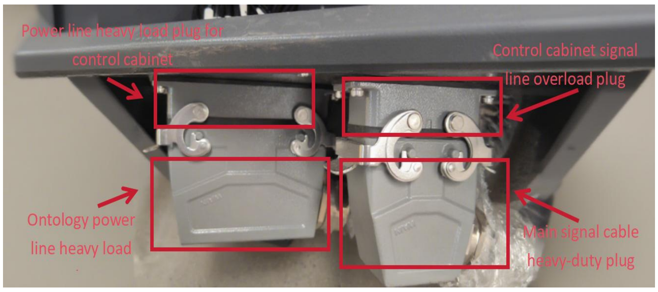
③	Control cabinet side load plug;
④	Trunk side load plug;
⑤	Control cabinet side teaching pendant connection port;
⑥	Teach pendant side connection port;
⑦	Control cabinet side power cord connection port;
⑧	Power cord side connection port;
⑨	Power button;

5.3.3XBC5 controller wiring, power-on, and start-up

Step	Graphical Representation	Explanation
1. Connect the robot body with the controller through the relay cable;		The power cord and relay cable on the controller side are designed as integrated units respectively, eliminating the need for additional installation.
2. Connect the Teach Pendant with the controller according to the figure;		
3. Start up the control cabinet after powering it on.		After the power cord is powered on and the POWER button is pressed, the xPad2 Teach Pendant will automatically start up.

①	Robot body side loading plug;
②	Trunk side load plug;
③	Teach pendant side connection port;
④	Power button;

5.3.4XBC5-E controller wiring, power-on, and start-up

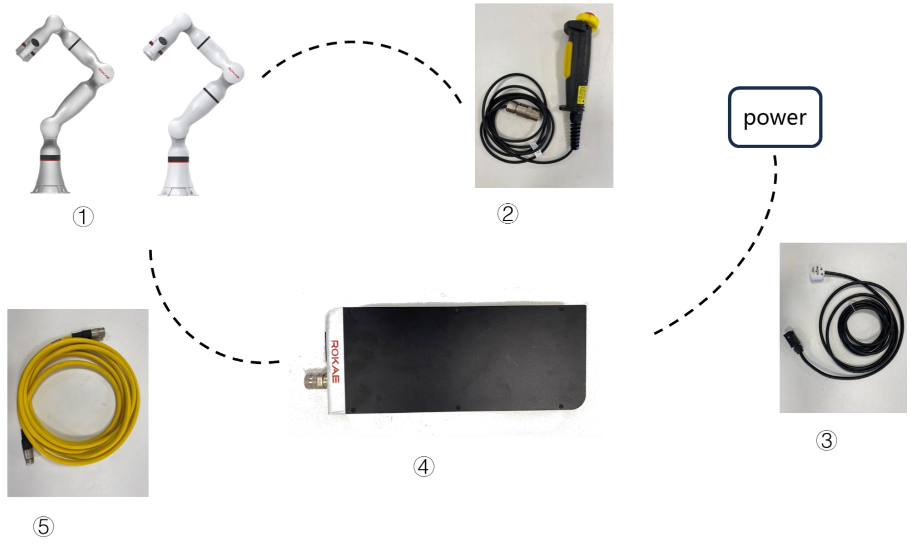
Step	Graphical Representation	Explanation
1. Connect the robot body with the controller through the relay cable;		The power cord and relay cable on the controller side are designed as integrated units respectively, eliminating the need for additional installation.
2. Connect the Teach Pendant with the controller according to the figure;		
3. Start up the control cabinet after powering it on.		After the power cord is powered on and the POWER button is pressed, the xPad2 Teach Pendant will automatically start up.

5.4Collaborative robot system composition

5.4.1ER and ER PRO

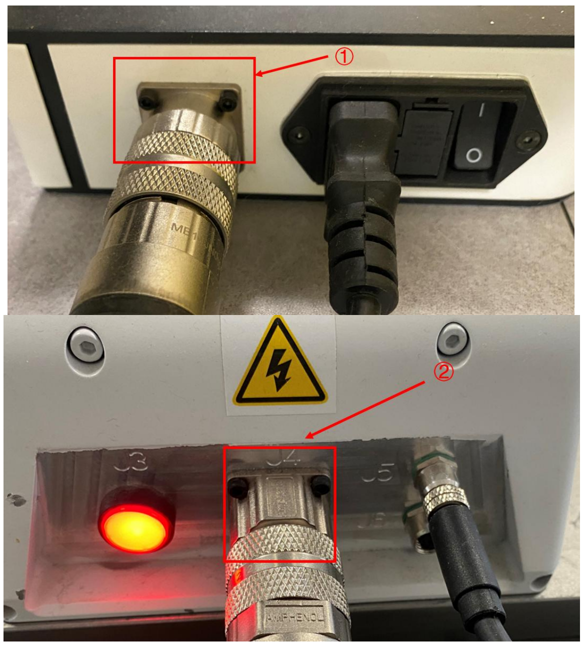
ER and ER PRO series are designed without a controller, and their system composition is shown in the figure below.

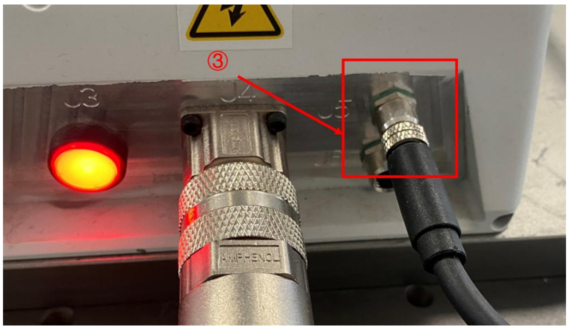
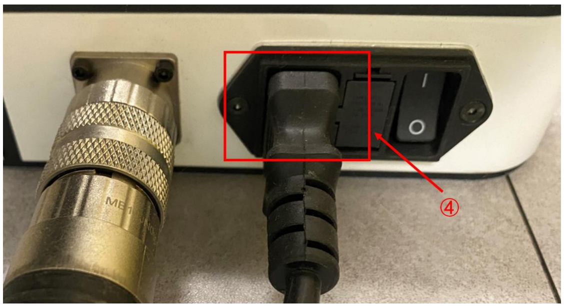
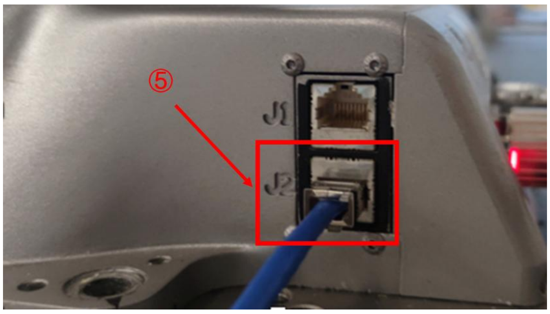
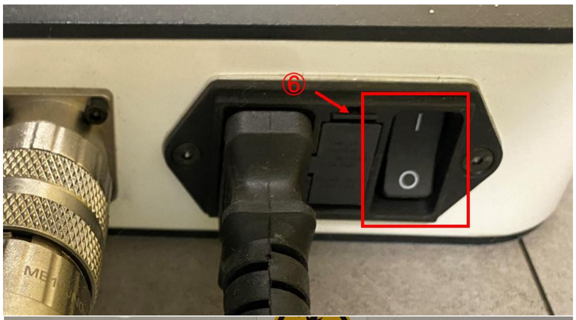
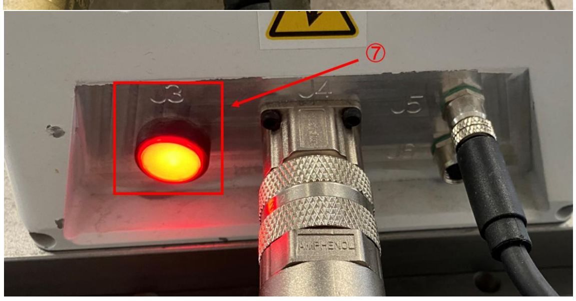
Attention: ER series robots do not support xPad2 Teach Pendant.



①	Robot body;
②	Handheld enable;
③	Power cable;
④	Transformer;
⑤	Connecting cable, for connecting with control cabinet or robot;

For ER series robots, you can refer to the following steps for connection and power-on.

Step	Graphical Representation	Explanation
1. Connect the robot body with the power adapter through the relay cable;		

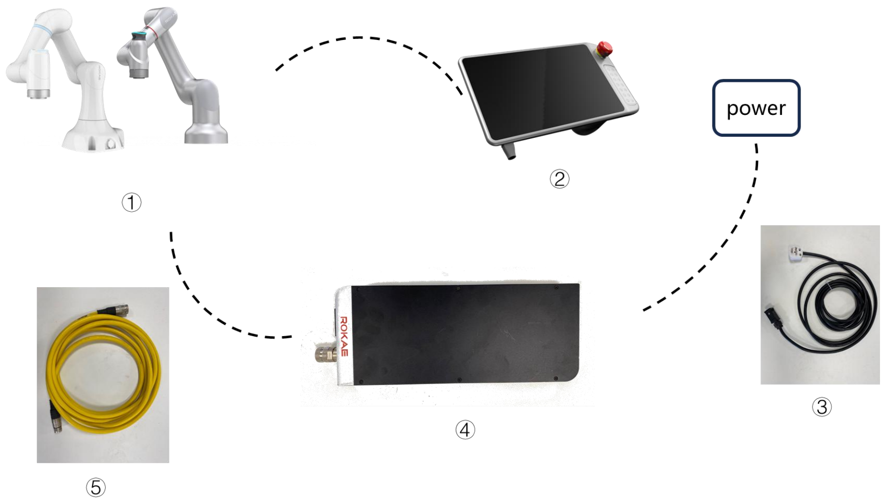
2. Connect the handheld enabling device.		
3. Connect the power cord.		
4. Connect HMI.		As ER series robots do not support connecting Teach Pendant, please connect via PC. See below for details.
5. Connect the power supply, and press the power adapter [switch] and the robot body power supply [switch] in sequence.		

①	Power adapter side trunk port;
②	Robot body side trunk line port;

③	Handheld enable port on the side of the robot body;
④	Power cord port on power adapter side;
⑤	Network cable port;
⑥	Power adapter switch;
⑦	Robot body switch;

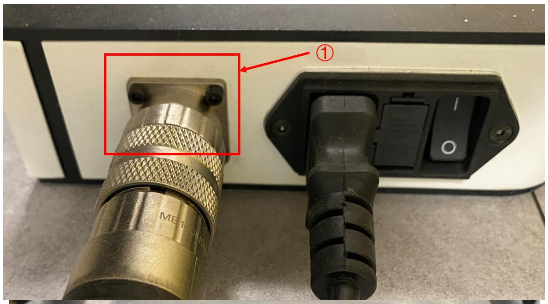
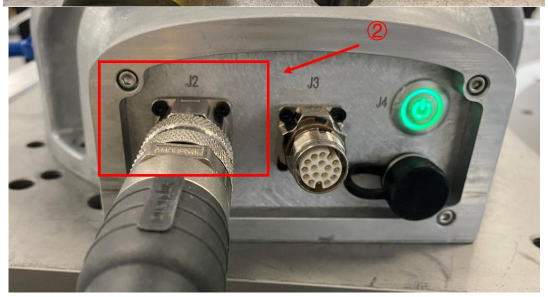
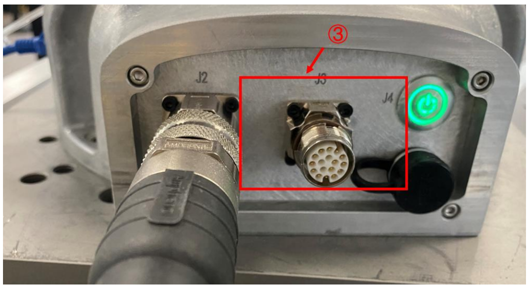
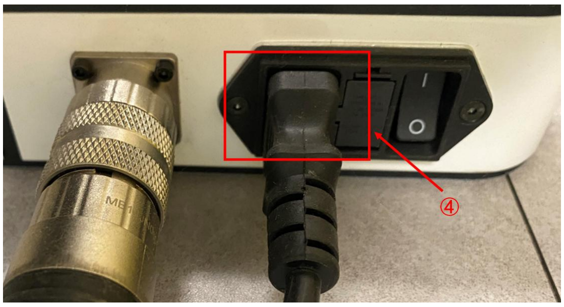
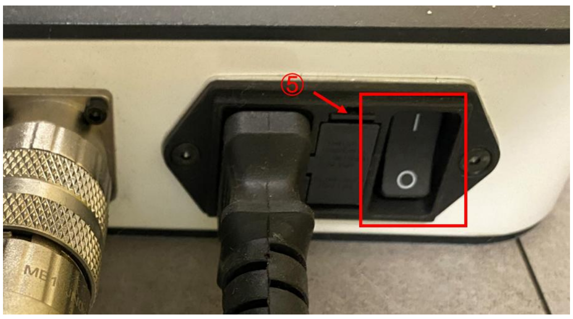
5.4.2CR and SR

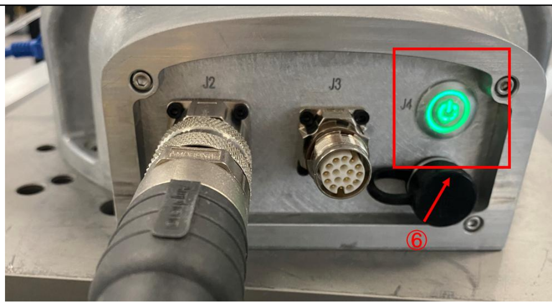
CR and SR series collaborative robots are designed without a controller, and their system composition is shown in the figure below.



①	Robot body;
②	Teaching pendant;
③	Power cable;
④	Transformer;
⑤	Connecting cable, for connecting with control cabinet or robot;

For CR series robots, you can refer to the following steps for connection and power-on. SR series is similar to CR series in connection and power-on.

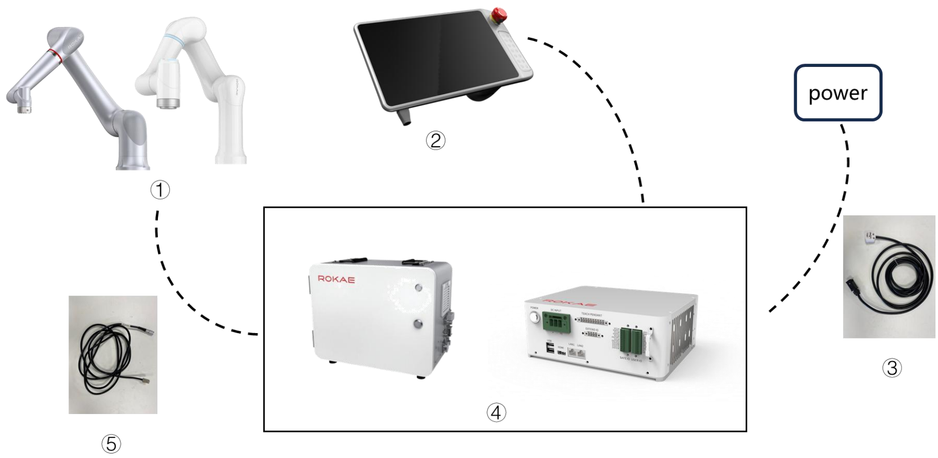
Step	Graphical Representation	Explanation
1. Connect the robot body with the power adapter through the relay cable;		
2. Connect the Teach Pendant;		
3. Connect the power cord;		
4. Connect the power supply, and press the power adapter [switch] and the robot body power supply [switch] in sequence.		After the robot is started up, the Teach Pendant will be automatically started up.



①	Power adapter side trunk port;
②	Robot body side trunk line port;
③	Handheld enable port on the side of the robot body;
④	Power cord port on power adapter side;
⑤	Power adapter switch;
⑥	Robot body switch;

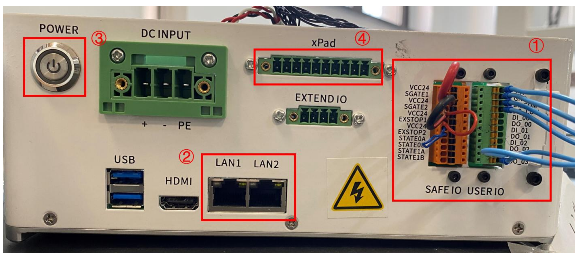
5.4.3CR-C and SR-C

CR and SR series collaborative robots are designed with a controller, and their system composition is shown in the figure below.

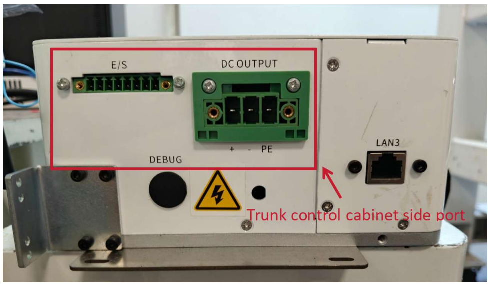
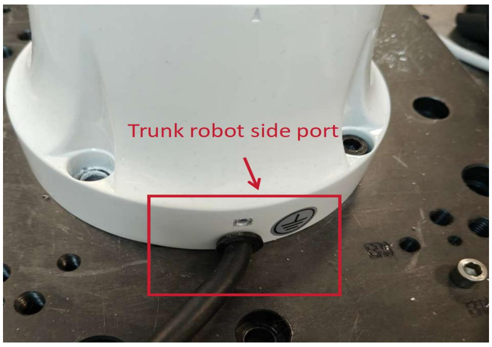
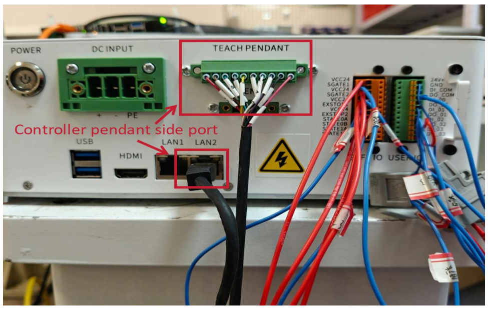
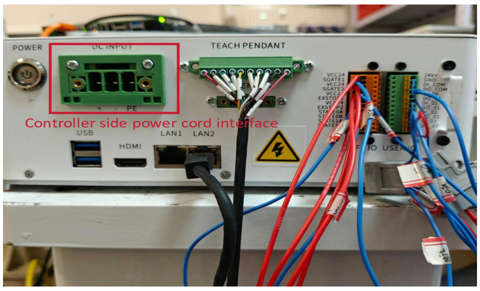


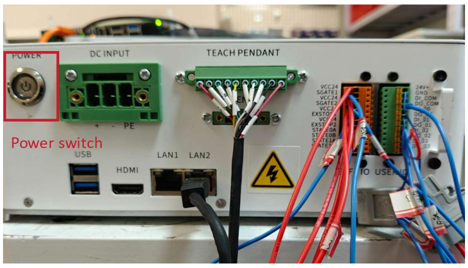
①	Robot body;
②	Teaching pendant;
③	Power cable;
④	Control cabinet;
⑤	Connecting cable, for connecting with control cabinet or robot;

5.4.3.1SR-C controller and its wiring, power-on and start-up



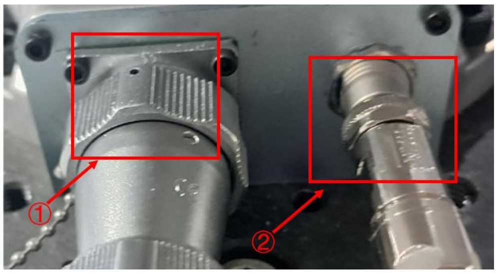
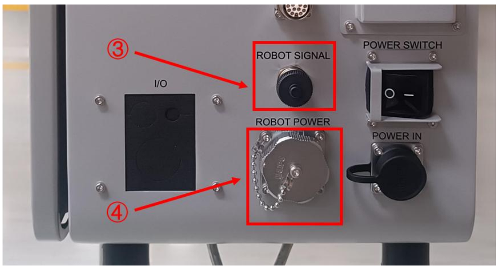
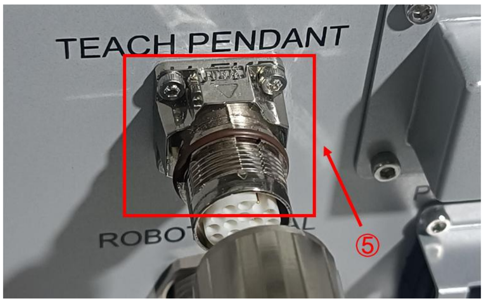
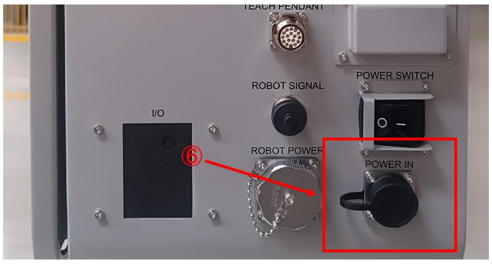
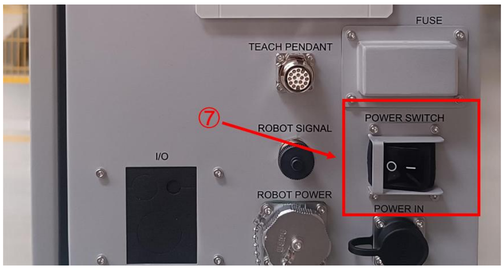
① Security/Universal IO wiring terminals;
② Network interface: including debugging interface and visual interface;
③ POWER switch: Used to control the power on/off state of the robot;
④ Teach Pendant wiring port: used to connect xPad2.

Step	Graphical Representation	Explanation
1. Connect the robot body with the controller through the relay cable;	 	As there is a difference between the two ends of the relay cable's heavy-duty connector, please confirm before connecting.
2. Connect the Teach Pendant with the controller according to the figure;		
3. Connect the power cord with the controller;		The interface of the power cord on the side of the controller is designed with a buckle.

4. Start up the control cabinet after powering it on.		After the power cord is powered on and the POWER button is pressed, the xPad2 Teach Pendant will automatically start up.

5.4.3.2CR-C controller and its wiring, power-on, and start-up

①	Security/Universal IO wiring terminals;
②	Realy cable interface: Used to connect the robot and controller;
③	Teach Pendant wiring port: Used to connect xPad2

Step	Graphical Representation	Explanation
1. Connect the robot body with the controller through the relay cable;	 	Note that the CR-C series controller has two relay cables: relay power cord and relay signal cable.
2. Connect the Teach Pendant with the controller according to the figure;		
3. Connect the power cord with the controller;		The interface of the power cord on the side of the controller is designed with a buckle.
4. Start up the control cabinet after powering it on.		After the power cord is powered on and the POWER button is pressed, the xPad2 Teach Pendant will automatically start up.

①	Robot side relay power cord port;
②	Robot side relay signal line interface;
③	Control cabinet side relay signal line port;
④	Control cabinet side relay power cord port;
⑤	Control cabinet side teaching pendant port;
⑥	Control cabinet side power cord port;

⑦	Power switch;

5.5HMI and robot connection

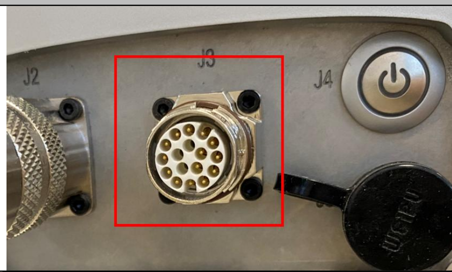
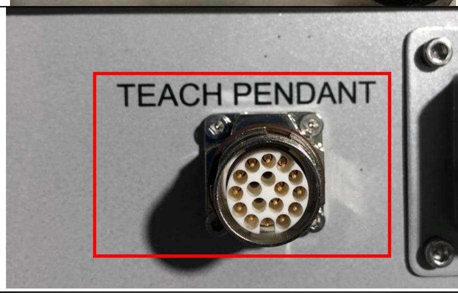
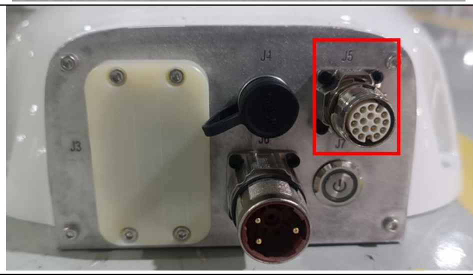
Robot Assist, as the host computer software of the robot, can run on PC, xPad2, and other devices. You can connect the device where the Robot Assist software is located and the robot to the same LAN (local area network) and establish a connection with the connected robot by robot detection, manually entering the controller service address, etc.

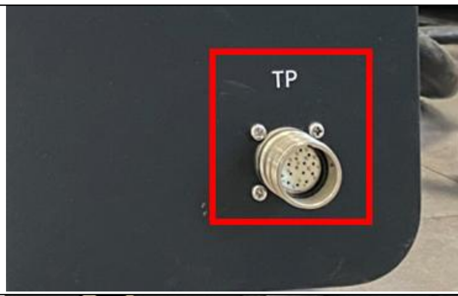
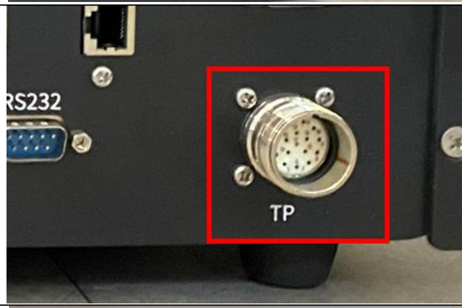
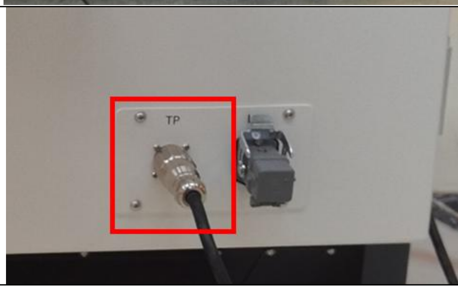
5.5.1xPad2 and robot connection

For the use of the Teach Pendant xPad2, the default network segment of the Teach Pendant is 192.168.1.X. You need to first modify the IP address of the Teach Pendant to be in the same network segment as the robot body, and then connect the xPad2 to the corresponding port of the robot;

5.5.1.1Hardware connection

5.5.1.2Connection configuration

Model	Introduction	Picture
CR	The xMate CR series robot xPad2 wiring port is located at the base.	
CR-C	The xMate CR-C series robot xPad2 wiring port is located on the upper part of the control cabinet.	
SR	The xMate SR series robot xPad2 wiring port is located at the base.	
SR-C	SR special controller (need to add an adapter)	

XBC5	The XBC5 series controller xPad2 wiring port is located at the bottom of the controller.	
XBC5-M	The XBC5-M series controller xPad2 wiring port is located at the bottom of the controller.	
XBC5-E	The XBC5-E series controller xPad2 wiring port is located at the bottom of the controller.	

After the hardware connection is completed and the robot is started up, xPad2 will be automatically started up and start its built-in Robot Assist software.

5.5.2PC and robot connection

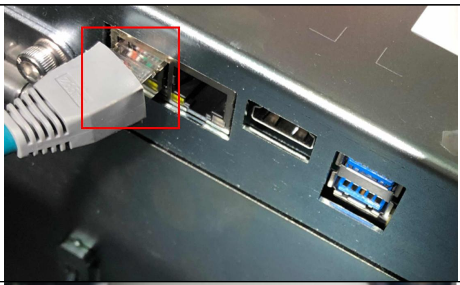
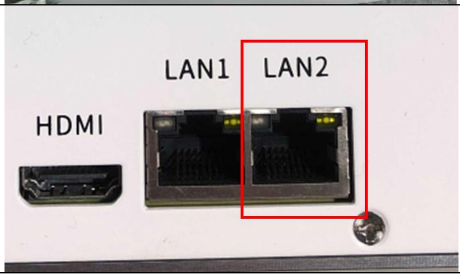
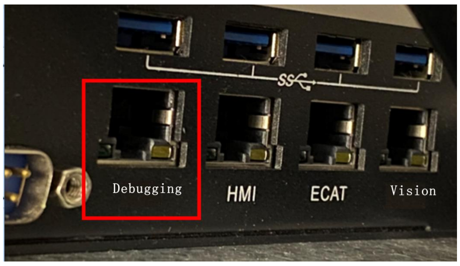
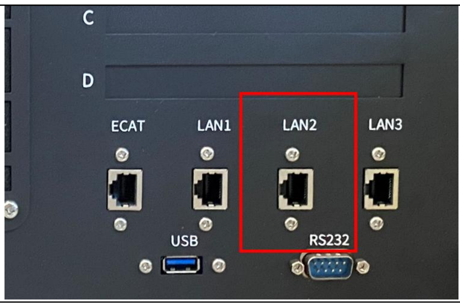
RobotAssist software can run on the PC, and then the PC can be connected with the robot or controller.

5.5.2.1Hardware connection

5.5.2.2One-to-one HMI and robot connection

When using a PC on which Robot Assist is running to debug a robot, the PC can be directly connected to the robot via network cable (table+illustration concretization);

Model	Introduction	Picture
ER/ER PRO	The xMate ER series cobot features two Ethernet interfaces on the base. The J2 port defaults to the fixed IP address of 192.168.0.160.	
CR	The xMate CR series cobot features only one Ethernet interface J1 (standard configuration) on the base, which defaults to the fixed IP address of 192.168.2.160.	

CR-C	xMate CR has a controller	
SR	xMate SR (J2 network interface)	
SR-C	The xMate SR-C debugging network interface is LAN2, whose default IP address is 192.168. 0.160;	
XBC5/XBC 5E	There are four Ethernet interfaces from left to right on the controller, which are: ● Debugging network interface, whose default configuration is the fixed IP address of 192.168.0.160; ● EtherCAT device expansion network interface, used for slave station extension; ● Visual network interface, for connecting industrial cameras, whose default configuration is the fixed IP address of 192.168.2.160; ● Bus extension network interface (optional).	
XBC5 M	LAN2 is the debugging network interface of the XBC5 M controller, and its IP address defaults to 192.168.0.160;	

5.5.2.3One-to-multiple HMI and robot connection

When switching between multiple robots, these robots can be connected to the same LAN, and the PC on which Robot Assist is running will detect the robots available for connection on the same network segment;

5.5.2.4Wireless connection

For scenarios where a wired connection is not convenient (such as on AGVs), the robot can be connected to a wireless router via the reserved network interface (the network interface on the xMate cobot base; and the visual/debugging network interface of industrial robot controller) on the robot controller and then to the HMID wirelessly.

5.5.2.5Connection configuration

5.5.2.6Direct cable connection

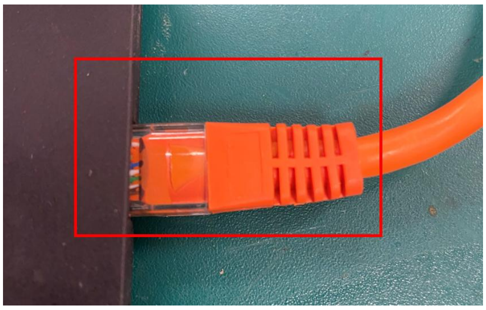
Both the robot base and the controller feature one network interface that defaults as the debugging network interface with the fixed IP address of 192.168.0.160. This IP address is the same for all robots and is not recommended to be modified arbitrarily. The PC on which Robot Assist is running can be connected to the network interface directly via a network cable to control the robot.

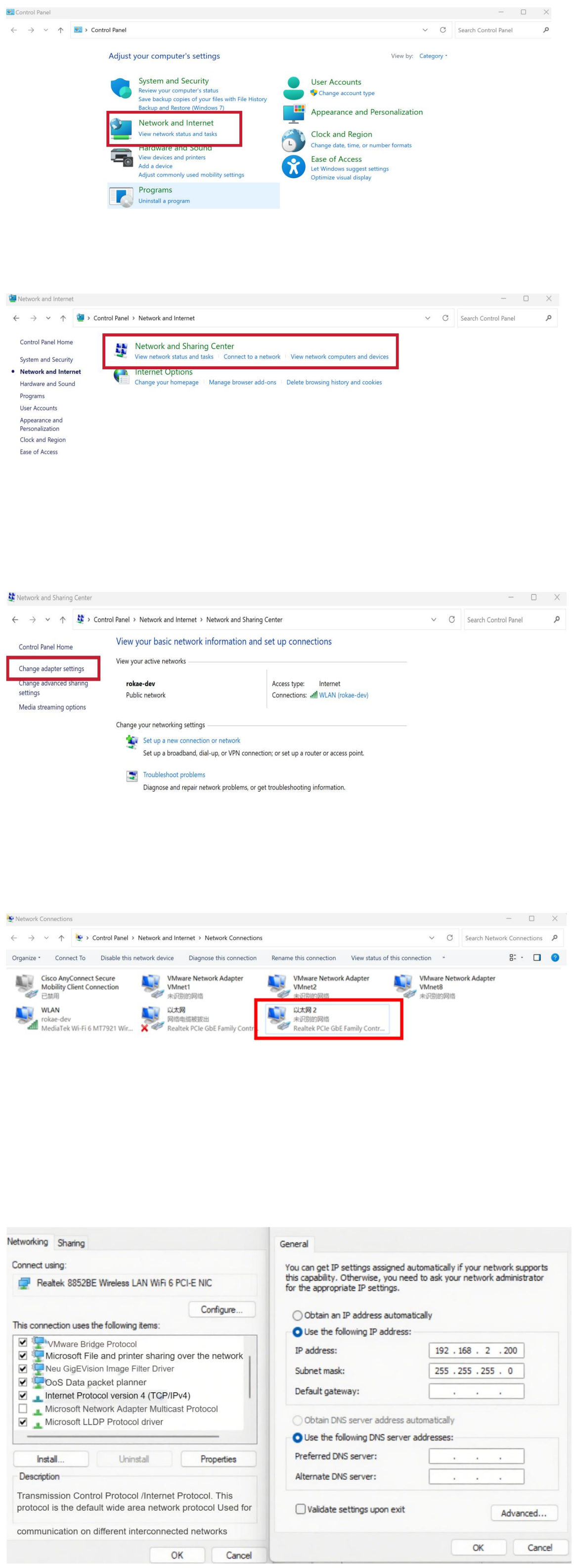
5.5.2.7External network interface connection

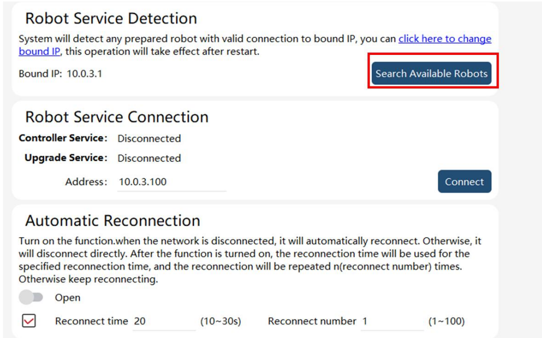
External network interface connection supports two types of settings: obtain an IP address automatically or assign a static IP address.
Obtain an IP address automatically — After the network interface J1 of cobots or the visual network interface of industrial robots is set to DHCP mode, and the robot is connected via the network interface to a router with DHCP, which automatically assigns an IP address to the robot, the robot can then be detected and connected via robot detection.
Assign a static IP address — After the network interface J1 of cobots or the vision network interface of industrial robots is set to the IP address in the required network segment, and the robot is connected via the network interface to a router, the robot can be visited and controlled via the robot's IP address.

5.5.2.7.1 Direct cable connection of devices such as PC

Both the robot base and the controller feature one network interface that defaults as the debugging network interface with the fixed IP address of 192.168.0.160. This IP address is the same for all robots and is not recommended to be modified arbitrarily. The PC on which Robot Assist is running can be connected to the network interface directly via a network cable to control the robot.
When using a mobile device such as a PC to connect to a robot, it is necessary to ensure that the LAN port address of the mobile device is in the same network segment as the robot. Regarding the modification method of PC (win11) static IP and robot (CR series) connection, you can refer to the following process steps:

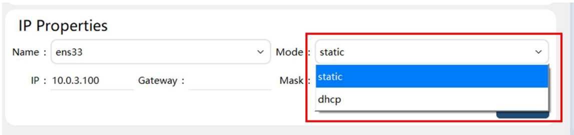
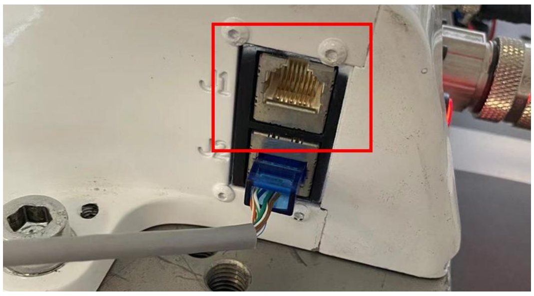
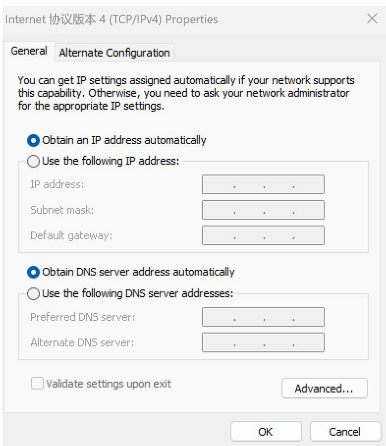
Step	Graphical Representation	Explanation
1. Network cable and robot connection. One end of the network cable is connected to the PC network interface, and the other end is connected to the robot network interface.	 	The default network segment of the network port at the side end of the CR series robot base is "192.168.2.XX".

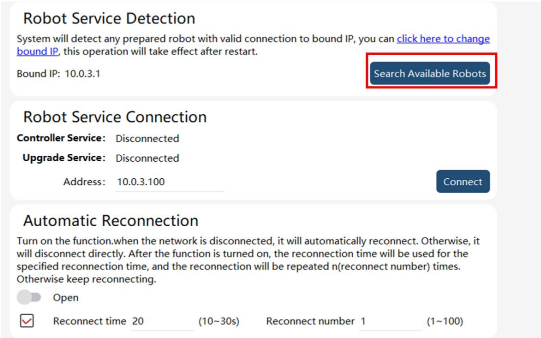
2. Local static IP modification. Enter the PC [Control Panel] -> [Network and Internet] -> [Network and Sharing Center] -> [Change adapter settings] -> Right-click to open the corresponding network interface [Properties] -> Double-click on [Internet Protocol Version 4 (TCP/IPv4)] -> Modify the IP address, subnet mask and default gateway of the terminal device (PC) and click [OK].		The IP address of the terminal device (PC) can be modified to any IP address that is not occupied in the same network segment as the robot, and its subnet mask and default gateway are consistent with those of the robot.

3. HMI and robot connection		

5.5.2.7.2 Wireless connection of devices such as PC

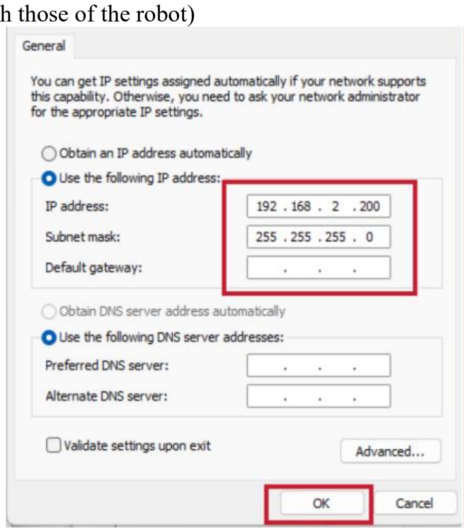
After the network interface J1 of cobots or the visual network interface of industrial robots is set to DHCP mode, and the robot is connected via the network interface to a router with DHCP, which automatically assigns an IP address to the robot, the robot can then be detected and connected via robot detection.

Step	Graphical Representation	Explanation
1. Modify the IP property of the robot system to dhcp.		See Chapter 6 for details;
2. Connect the robot to a router.		Set the network interface J1 of cobots or the visual network interface of industrial robots to DHCP mode, and connect the robot via the network interface to a router with DHCP, which automatically assigns an IP address to the robot.
3. Connect the PC to the router network in the same network segment, and set the IP acquisition mode to DHPC.		Enter the [Internet Protocol Version 4 (TCP/IPv4)] page, and refer to the part of Step 2 of the above manual IP modification.

4. Connect HMI to the robot		

5.5.2.7.3 IP address modification

Using the Windows 10 operating system as an example, connect one end of the Ethernet cable to the robot's J2 interface and the other end to the terminal device (PC); click on the "Start > Control Panel" menu on the terminal device (PC), and select "Network and Sharing Center" (the "Network and Sharing Center" window will pop up); click on "Local Area Connection" in the "Network and Sharing Center" window (the "Local Area Connection Status" interface will appear); click on "Properties" in the "Local Area Connection Status" interface, (the "Local Area Connection Properties" interface will appear); double-click on "Internet Protocol Version 4 (TCP/IPv4)" in the "Local Area Connection Properties" interface, (the "Internet Protocol Version 4 (TCP/IPv4) Properties" interface will appear); and select "Use the following IP address" in the "Internet Protocol Version 4 (TCP/IPv4) Properties" interface, modify the IP address, subnet mask, and default gateway of the terminal device (PC), and confirm the changes. (The IP address of the terminal device (PC) can be modified to any IP address that is not occupied in the same network segment as the robot, and its subnet mask and default gateway are consistent with those of the robot)

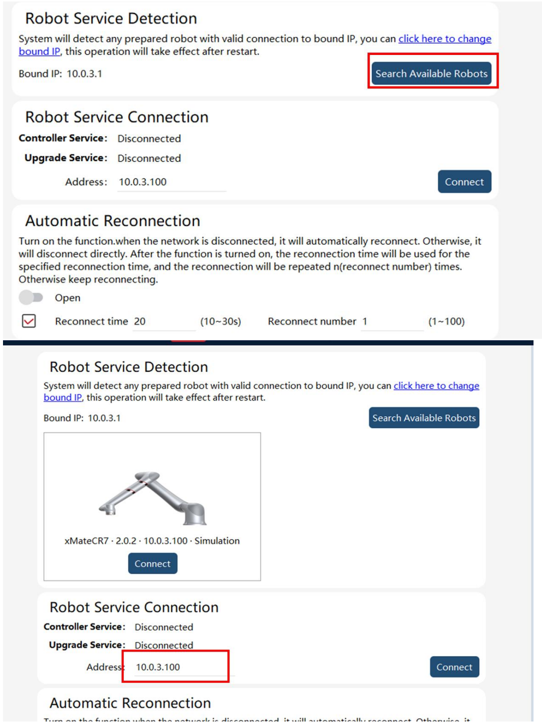
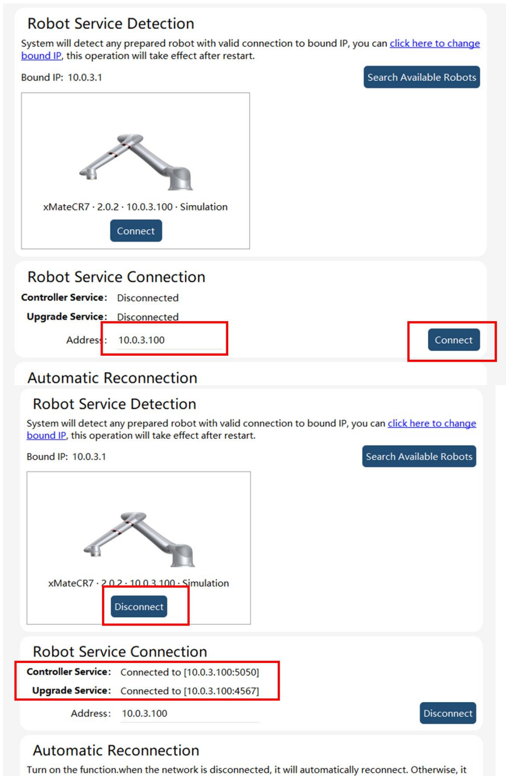


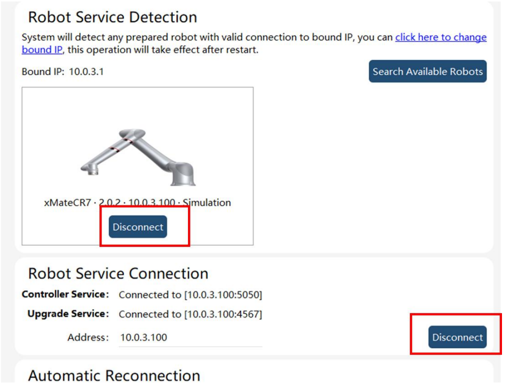
5.5.3Robot detection and connection

> **Warning**
>
> When manually modifying the IP address of the robot's network interfaces, do not set different network interfaces as static IP addresses of the same network segment; do not arbitrarily modify the network mode and IP address (192.168.0.160) of the debugging network interface; do not arbitrarily modify the network mode and IP address (192.168.1.160) of the Teach Pendant xPad's network adapter card.

HMI can detect and display all robots available on the same network segment for connection. You can detect and connect robots by following these steps.

Step	Graphical Representation	Explanation

1. Search for available robots. Click on the network icon on the bottom status bar to rapidly enter the robot search interface, and click on [Search Available Robot].		When searching for robots, please make sure the device on which Robot Assist is running and the robots are on the same network and the network is connected.
2. Connect the robots. Enter the IP address of the robots and click on [Connect].		When the robots are connected successfully, [Controller Service] and [Upgrade Service] will display "Connected to XXXX". The bottom status bar icon changes to . Simultaneous connection of multiple Robot Assist is not supported. Another Robot Assist can only be connected after the current Robot Assist is confirmed to be disconnected or the robot is restarted.

3. Disconnect the robots. Click on the Disconnect button in the Connection interface to disconnect Robot Assist from the controller.		The Robot Assist connection can be restored in the same way it is connected for the first time.

6HMI Introduction

6.1Introduction to this chapter

This chapter outlines the basic layout of xCore's HMI software and the distribution and role of its main functions.

Users need to read this chapter before actually using the robot.

6.2RobotAssist introduction

RobotAssist is the host computer software of the xCore control system, with functions including robot motion control, task editing, parameter setting, and status monitoring. The software can be installed on PC, Surface, and xPad2 Teach Pendant. The devices can control a robot after being connected to it as long as they are in the same network segment as the robot.

When you use the xPad 2, if the teach pendant encounters a crash or blue screen issue, the RobotAssist software will automatically restart to resume operation. (Please note that this auto-restart feature is not applicable to the PC side.)

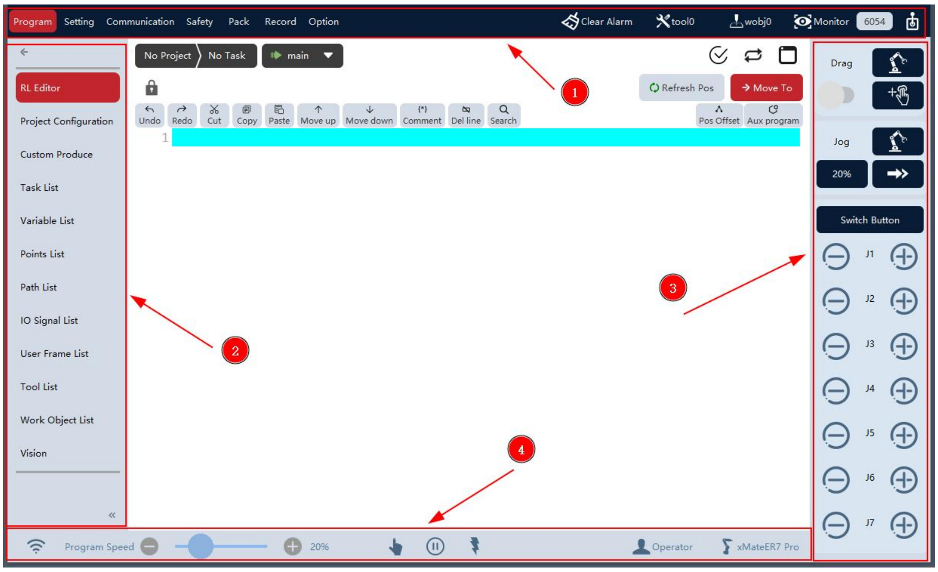
Besides xPad2, we suggest using a tablet or a laptop as the operating terminal. The recommended configurations are shown in the table below.

Terminal type	Tablet	Terminal type	Laptop
ROM	32 GB	ROM	32 GB
RAM	4 GB	RAM	4 GB
Screen size	8.0 inches and above	GPU	Intel HD Graphics 4000 or higher
Network communication	Wi-Fi	Network communication	Wi-Fi or wired LAN
Operating system	Windows 7 64 bit, Windows 10 64 bit, Ubuntu20.04		
CPU	Intel Core I3 and above		

> **Note**
>
> Robot Assist interacts with the controller in real time. When it is used on a PC, frequent changes in window size may cause the interface to stop refreshing. In this case, you can restore it by pressing Alt+Tab to switch between windows.

6.3General layout of HMI

The main operation interface is usually composed of 4 main areas, including: top status bar, bottom status bar, left sidebar, and right operation interface.



①	Top status bar
②	Left sidebar
③	Right operation interface
④	Bottom status bar

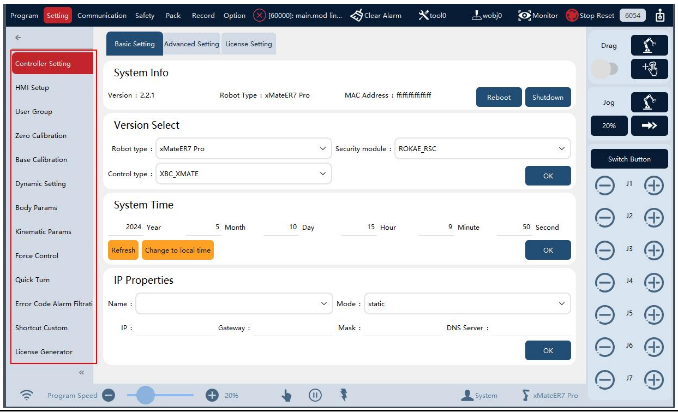
6.3.1Top status bar

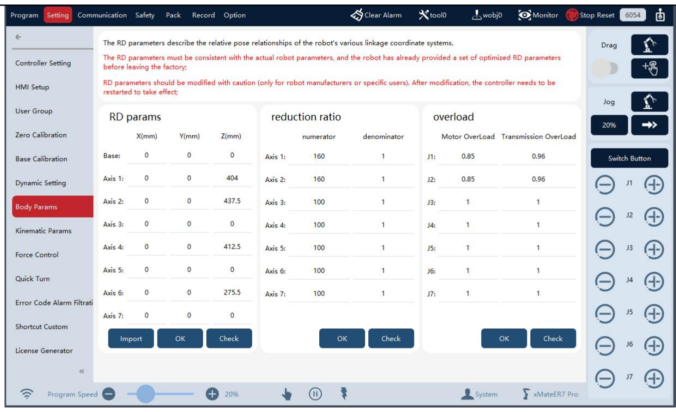
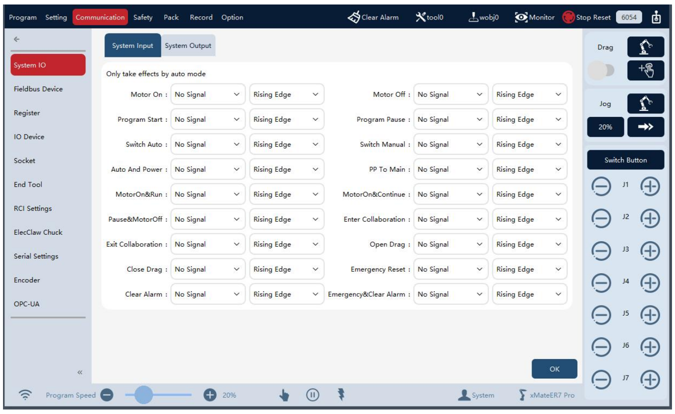
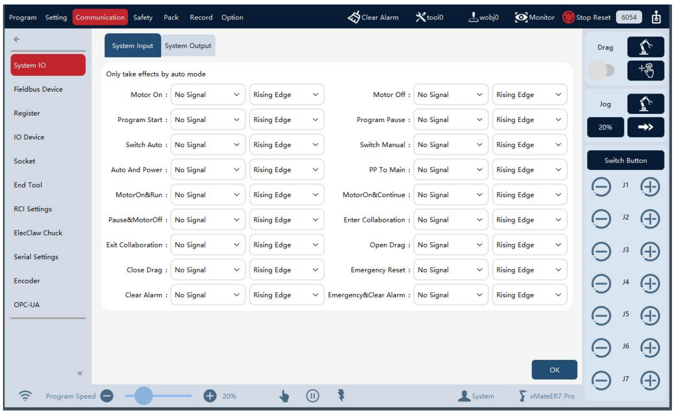
Top status bar, including: several first-level menu buttons (programming, settings, communication, safety, process package, log, options), instant log, clear alarm button, tool information button, work object information button, status monitoring button, RSC reset button, security check button, and operation interface button.

①	First-level menu buttons, including programming, settings, communication, safety, process package, log, and options. Click one of the above buttons to go to the corresponding sub-interface;
②	Instant log, showing the latest controller log information of the system. Click it to go to the controller log sub-interface;
③	Clear alarm button. Click it to clear the alarm status of the instant log and the controller;
④	Tool selection button, displaying the information of the tool used currently. Click it to select the tool to be used;
⑤	Work object button, displaying the information of the work object used currently. Click it to select the work object to be used;
⑥	Status monitoring interface button. Click it to open/close the status monitoring interface;
⑦	Emergency stop reset button (available only on robots that are equipped with a safety controller). When the robot is in an emergency stop, safety gate open, or safety stop state, after eliminating the factors that cause the above states, clicking this button can restore the robot to its normal operating state;
⑧	Security checksum, used to check the five pages involved in security: soft limit, virtual wall, collision detection, security monitor, and collaboration mode. When the five types of settings change, this page will pop up for reconfirmation;
⑨	Operation interface button. Click it to open/close the operation interface;

6.3.2Left sidebar

When switching between different functions through the top status bar, such as programming, settings, and communication, the left sidebar will display the corresponding submenu items for each function.

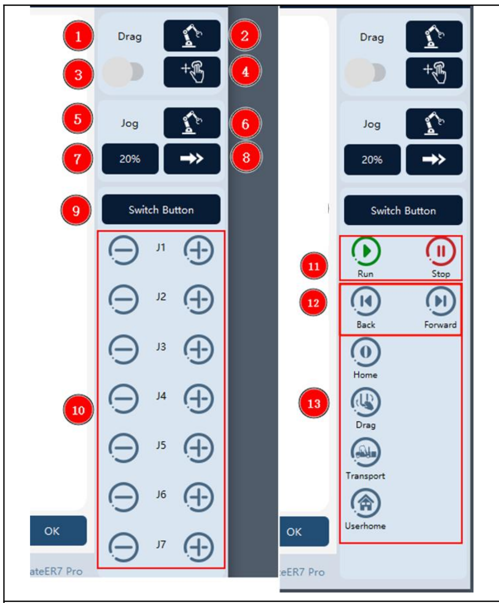
	After clicking "Settings" on the top status bar, the left sidebar displays all the "Settings" submenu items.
	After clicking "Body Parameters", the "Body Parameters" setting page is accessed.

	
	After clicking "Communication" on the top status bar, the left sidebar displays all the "Communication" submenu items.
	After clicking "Settings" on the top status bar again, the left sidebar displays all "Settings" submenu items, and the middle interface displays the "Body Parameters" page accessed last time by default.

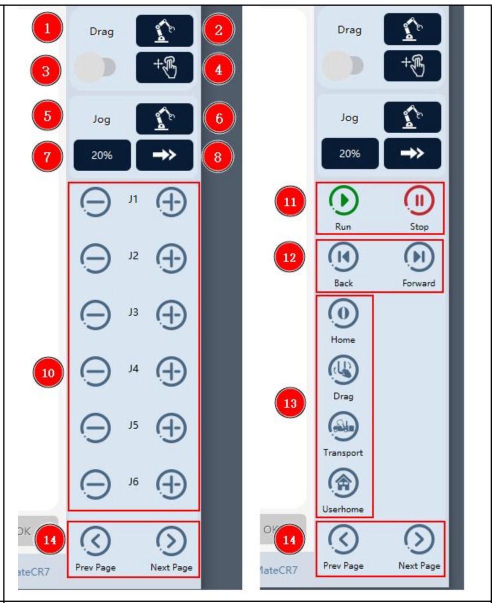
6.3.3Right operation interface

You can click or on the top status bar to open the operation interface, which can be used to change the robot control mode, control robot motion, and perform pose teaching.

The robot supports two types of pose teaching: Jog mode and Drag mode (for cobots only). Jog mode, in which the robot is controlled to move in the corresponding direction through the Jog button. Drag mode, in which the robot motion is directly and manually guided by using the end-effector drag Pilot/xPanel handle.



Operation interface of 7-axis robots



Operation interface of 6-axis or below robots

①	Drag settings zone, indicating that ②, ③, and ④ are drag-related options.
②	Drag space setting: joint space drag and Cartesian space drag.
③	Drag enabling switch: turn on/off Drag mode.
④	Drag mode setting: For joint space drag, only free drag is available. For Cartesian space drag, the three options of translational drag only, rotational drag only, and free drag are available.
⑤	Jog settings zone, indicating that ⑥, ⑦, and ⑧ are Jog-related options.
⑥	Jog reference system setting. It is used to select the single axis mode and Cartesian mode during Jog, as well as the reference frame in Cartesian mode, including: world frame, base frame, flange frame, tool frame, and work object frame.
⑦	Jog speed setting. Set the robot Jog speed between 1% and 100% (expressed in a percentage relative to the top Jog speed limit of 250 mm/s).
⑧	Jog step mode setting. Set Jog mode to Continuous Jog or Stepping Jog, and the stepping increment can be adjusted.
⑨, ⑭	Switch function area. Switch between the Jog button and buttons ⑪, ⑫, and ⑬.
⑩	Jog button. For a 7-axis robot, J1 to J7 are displayed in the case of joint space Jog and X/Y/Z/A/B/C and Elbow in the case of Cartesian space Jog. For a 6-axis robot, J1 to J6 are displayed in the case of joint space Jog and X/Y/Z/A/B/C in the case of Cartesian space Jog.
⑪	Program start/stop button.
⑫	Previous (not supported yet)/next program running buttons.
⑬	Shortcut button area. It allows you to configure the shortcut functions of the buttons in the custom button interface. The default configuration includes four buttons: Initial Pose, Drag Pose, Shipping Pose, and User Pose. For a more detailed introduction, you can refer to the section, "Settings"-"Quick adjustment".

Note

It is important to make sure that the robot is currently in manual mode and powered off before performing Jog and turning on the Drag enabling switch.
It is not allowed to start the program when the register with the pause function or system IO has not been reset.

6.3.4Bottom status bar

The bottom status bar displays the connection status between Robot Assist and the robot, program running speed, robot operating mode, robot status, motor status, current user login information, and robot model.

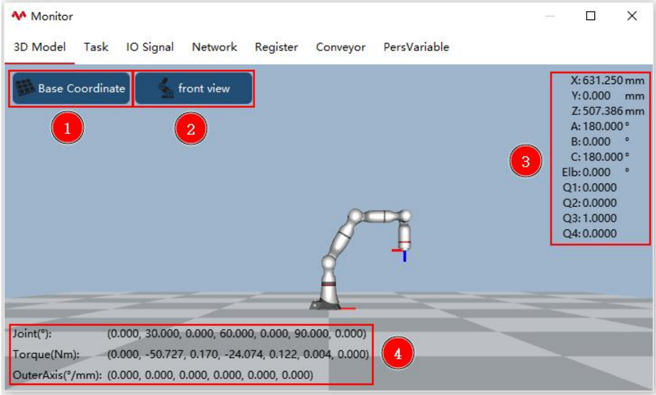
①	Connection status between the RobotAssist software and the robot. Click this button to open the connection setting page of the robot. The icon means disconnected, and the icon means connected. The animation icon means that an attempt is being made to connect to the robot. The icon is the state where the upgrade service is connected and the controller service is not connected, and in this state, the control system upgrade operation can be carried out, while the robot cannot be operated and the robot parameters cannot be set.
②	Program running speed adjustment control, used to adjust the RL program running speed, with an adjustable range of 1%−100%. This parameter independently affects the program running speed in both manual and automatic modes. The program speed (-/+1%) can be fine-tuned by using the slider or clicking buttons "-/+". Attention: The upper limit of program speed may be affected by "Upper limit of program speed in manual mode" and "Upper limit of initial speed of program in automatic mode" in "Advanced Settings" of "Controller Settings". You can refer to the relevant chapters for more information.
③	Robot operating mode is divided into manual and automatic modes, which are described in more detail in the chapter "Basic Operation of the Control System". Manual mode, in which users usually write and debug programs. Automatic mode, in which users usually carry out continuous automatic production.
④	Robot state, including the following specific states. , Idle state. The program is stopped, and the robot is not in motion. , Program running state. The program is running, and the button turns red when the robot is in motion. , Drag mode. The robot can be dragged when the controller is in Drag mode. The button turns red when the robot is in motion. , Demo mode. The controller plays the Demo when it is in Demo mode. The button turns red when the robot is in motion. , Identification mode. The controller is in Identification mode, and the button turns red when the robot is in motion. , Jog mode. The controller is in Jog mode and changes when the Jog button is pressed or released. RCI , RCI mode. The controller is in RCI mode, and the button turns red when the robot is in motion. , Collaboration mode. The controller is in Collaboration mode, which is displayed in combination with other statuses in the upper right corner of the icon. , Error state. The robot system is in Error state. , Debug mode. The controller is in Debug mode, and the button turns red when the robot is in motion.
⑤	Project semi-static state. The button is displayed when the robot is in the semi-static task running state; otherwise, it is not displayed. Click this button to stop the semi-static task.
⑥	, Power-on state. The robot motor is in the power-on state. Click this button to power it off. , Power-off state. The robot motor is in the power-off state. Click this button to power it on. E , Emergency stop state. The robot is in the emergency stop state, and the robot motor cannot be powered on. G , Safety gate state. The safety gate is open, and the robot motor cannot be powered on. S , Safe stop state (for only models equipped with a safety controller). The robot is in a safe stop state, which means that the safety controller detects that the work or communication is abnormal, or a parameter exceeds the safety threshold set by the safety controller, and the robot cannot be powered on.
⑦	Current login user information: Operator, Teacher, Programmer, Admin, and System. Click the button to go to the user login interface.
⑧	Robot model information.

6.4Status monitoring

Click the "Status Monitoring" button on the top status bar to open the floating status monitoring interface. Through the status monitoring interface, the following items can be monitored: robot 3D model, task running status, IO signal, network connection status, register variables, conveyor belt status, and PERS variable information, which is convenient for users to quickly understand the robot status.

6.4.13D model monitoring

The interface visually displays the current state of the robot in the form of a 3D model. The 3D model's viewpoint can be switched by clicking and dragging, and the model can return to the default viewpoint by clicking the "Front view" button.



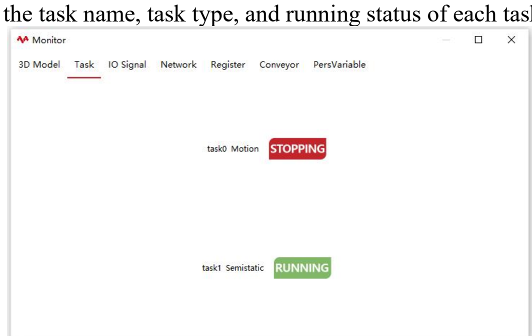
①	Frame switching: selectable base frame/world frame/work object frame. The base frame may not coincide with the world frame when the robot is not installed upright or when a group of robots is used.
②	Front view: Click the "Front view" button to return the model to the default viewpoint.
③	End-effector pose: The position and orientation (RPY or quaternion) of the robot end-effector relative to a certain frame (work object frame, base frame, and world frame).
④	Joint angle, joint torque, and external axis information.

Note

At present, only RobotAssist on PC supports 3D model display, and only status data can be displayed on xPad2.

6.4.2Task monitoring

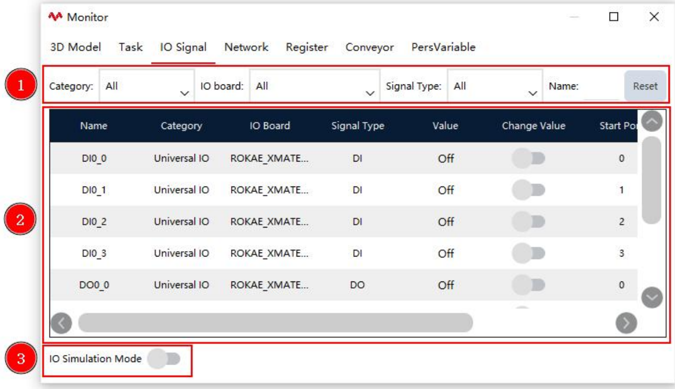
This interface displays the task name, task type, and running status of each task in the current project.



6.4.3IO signal monitoring

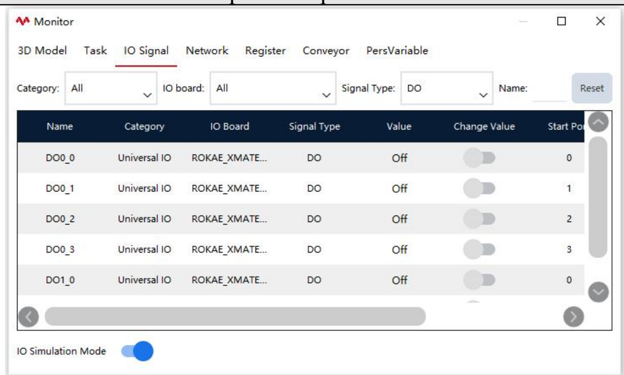
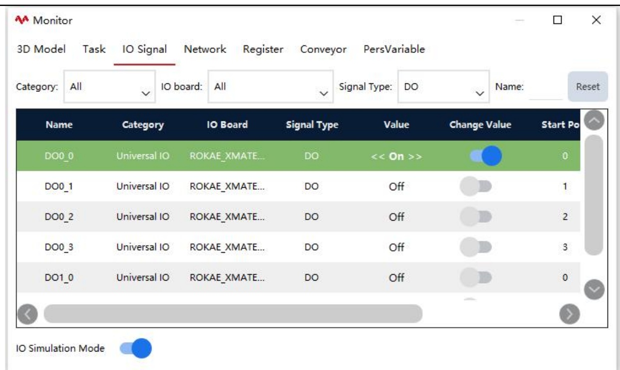
For xMate cobots, the IO signal monitoring interface displays the 4-channel DI and DO signals on the robot base and the 2-channel DI and DO signals at the end-effector by default.

For industrial robots, the IO signal monitoring interface displays configured IO signals in the controller by default.



①	Filter, used to filter the displayed IO. The selectable filter conditions include category, IO board, signal type, and name. Click the "Reset" button to reset the filter conditions
②	IO signal list
③	Open IO Simulation mode to simulate DI signal value

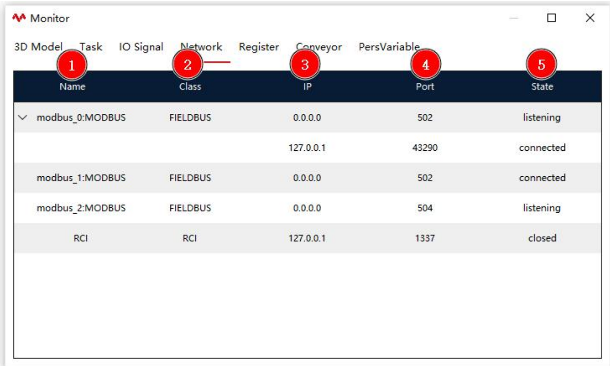
Operating steps of IO Simulation mode:

Step	Graphical Representation	Explanation
1. Go to the IO Signal page and click the [IO Simulation Mode] enabling switch to activate Simulation mode.		The Simulation mode has access restrictions and requires Operator-level or higher permissions for use.
2. Click the DI/DO value buttons to start the simulation.		Note that even not in the Simulation mode, DO can also be forced to output.

6.4.4Network connection monitoring

This interface displays the network information that is currently connected with the controller, including: name, type, IP, port, and status.

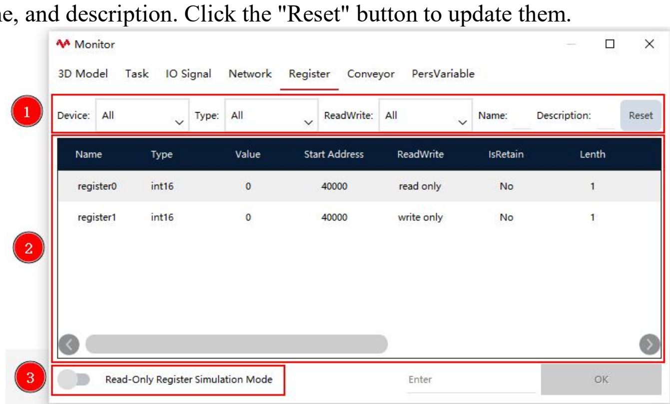
ROKAE



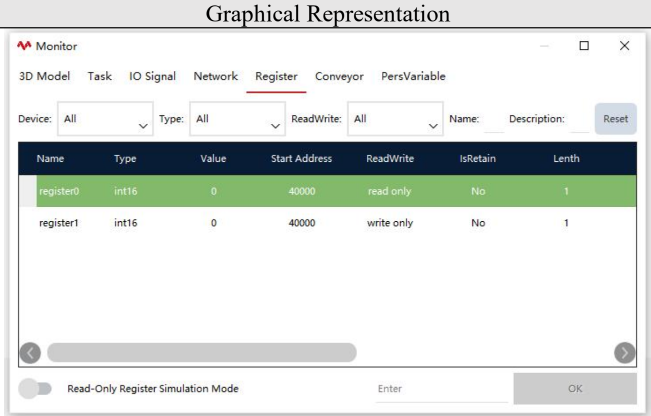
①	Name: MODBUS, RCI, and SYS_SOCKET are system default unique names. User-defined names are displayed for new connections.
②	Type: The connection status of MODBUS, RCI, and SOCKET can be displayed. Corresponding connections can be added and configured in the relevant interface. SYS_SOCKET refers specifically to the connection of external communication.
③	IP: For a client-side connection, the IP address of the target server is displayed. For a server-side connection, its own IP address is displayed.
④	Port: For a client-side connection, the port number of the target server is displayed. For a server-side connection, its own port number is displayed.
⑤	Status: Generally, there are three types of connection status: Connected, Disconnected, and Connecting. For a server-side connection, it displays Monitoring when it is disconnected.

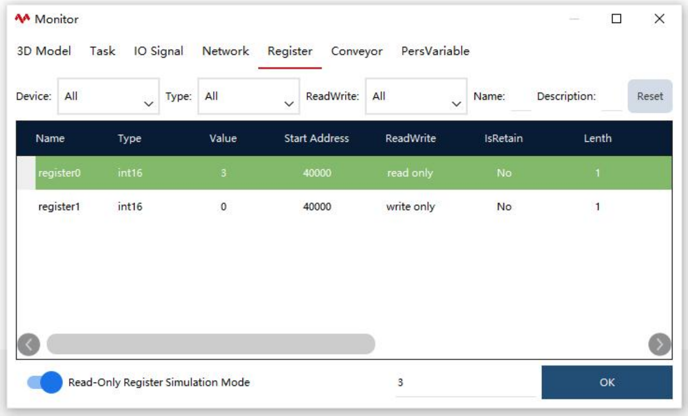
6.4.5Register monitoring

This interface displays the information of each register. If you think that there are too many registers displayed by default, you can filter them. The selectable filter conditions include: device, type, read-write, name, and description. Click the "Reset" button to update them.



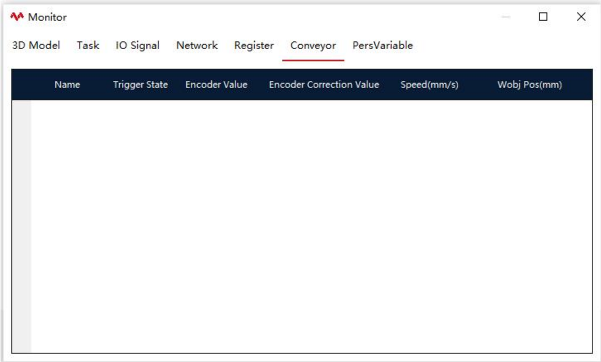
①	Filter: Used to filter the displayed registers.
②	Register signal list.

Step	Graphical Representation	Explanation
Write-only register assignment: Select the write-only register, enter the expected value in the input box to the left of the "Write" button, and click the "Write" button to change the current value of the register.		1. Users with the permission level of Operator are not allowed to perform writing; 2. Only write-only registers and registers that are not bound with function codes can be assigned.

Read-only register assignment: Enable the "write-only register stimulation mode", enter the expected value in the input box to the left of the "Write" button, and click the "Write" button to change the current value of the register.		

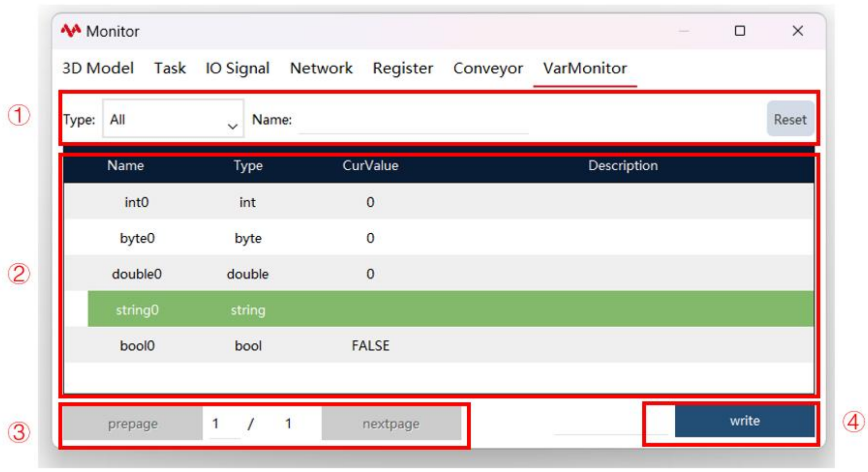
6.4.6Conveyor belt monitoring

Used to cooperate with the conveyor belt process package and monitor the status of conveyor belt. See the conveyor belt tracking process package for details.

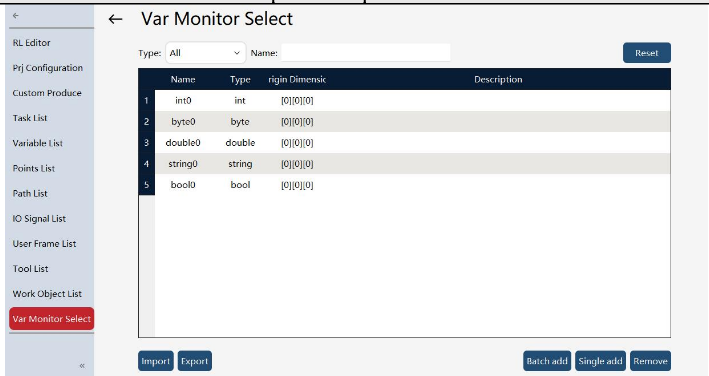
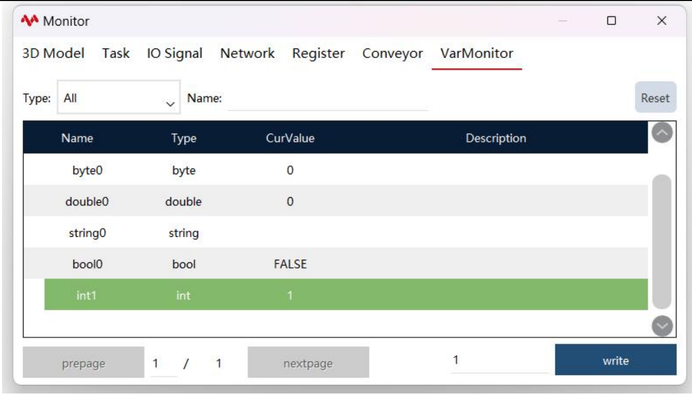


6.4.7Variable monitoring

This interface displays the real-time information of variables listed in the variable list.



①	Filter: Filter according to class insertion type and name filter. Note: Currently, variable monitoring supports only the monitoring of individual variables or arrays of the following types: bool, int, double, byte, and string.
②	Monitor the variable list.
③	Previous/Next: Only up to 10 variables can be monitored per page.
④	Write button: Assign the variables.

Step	Graphical Representation	Explanation
1. In the variable monitoring selection interface, add variables to be monitored using the "Batch Add" button or the "Single Add" button.		Variable monitoring only monitors variables that exist in the selection interface
2. Write: In the "Modified Value" column, write the value of the variable to be modified, and click "Write" button to write the value to the variable in real time.		Users with the permission level of Operator are not allowed to perform writing. When modifying a non-PERS variable, clicking "PPTOMAIN" will revert the changed value of the non-PERS variable to its original state.

Note: After adding a new variable to be monitored, you need to first execute the "PPTOMAIN" operation before you can read or modify the variable.

6.5Programming module overview

RL editor	Mainly used for writing and debugging RL programs.
Project configuration	Used for operations such as project creating, loading, import and export, and push.
Custom production	Users can customize several controls to conveniently monitor and edit registers, DI/DO signals, and project variables.
Task list	Used for viewing, creating, editing, and importing and exporting tasks.
List of variables	Used for viewing, creating, editing, and importing and exporting variables.
Point list	Used for viewing, creating, editing, and importing and exporting points.
Path list	Used for viewing, creating, editing, and importing and exporting paths.
IO signal list	Used for viewing, creating, editing, and importing and exporting IO signals.
User frame list	Used for viewing, creating, editing, and importing and exporting user frames.
Tool list	Used for viewing, creating, editing, and importing and exporting tool frames.
Work object list	Used for viewing, creating, editing, and importing and exporting work object frames.
Variable monitoring selection interface	In the variable monitoring selection interface of status, you can perform the following operations on monitored variables: import, export, batch add, single add, and deletion.

6.6Setting module overview

Controller settings	Controller system-related setting interface, including robot type, system time, and system IP.
HMI settings	HMI-related settings, including language switching and Teach Pendant IP settings.
User group	User management, including login and password management.
Calibration	Perform robot zero calibration, force sensor calibration, soft calibration, and base frame calibration
Frame calibration	Set the global tool, global work object, and global user frames.
Dynamic settings	Robot dynamics-related settings.
Body parameters	Robot kinematics-related settings, such as RD parameters, reduction ratio,

	and overload coefficient.
Motion parameters	Acceleration and deceleration performance, safety control, and other settings for the robot.
Force control parameters	Force control-related settings for the robot.
Quick adjustment	Quick orientation adjustment settings.
Electronic nameplate	Electronic nameplate-related settings. (This function is only supported by some models)
Error code alarm filtering	Error code alarm filtering-related settings.
Custom buttons	Custom button binding functionality.

6.7Communication module overview

System IO	System input and system output signal settings.
External communication	External communication interface settings based on the Socket.
Register	Register-related settings.
IO device	IO device settings.
Bus devices	Configure various bus expansion modules.
RCI settings	RCI communication settings.
xPanel settings	CR robot xPanel settings.
Electric gripper and suction cup	Settings and tests of all kinds of electric grippers and suction cups.
Serial port settings	Serial port-related settings.
Encoder	Encoder settings required for conveyor belt tracking function.
OPC-UA	OPC-UA-related settings.

6.8Safety module overview

Joint limit	Robot joint limit settings.
Robot limits	Robot speed, reduced mode speed, power, and momentum settings.
Virtual wall	Virtual wall-related settings.
Collision detection	Collision detection-related settings.
Safe region	Safe region-related settings.
Safety position	Safety position-related settings.
Safety DO settings	Safety DO-related settings.

6.9Process package module overview

Support stacking, tray, conveyor belt tracking, PV typesetting, PV inserting, and other process packages. Please refer to the corresponding chapters.

6.10Log module overview

HMI logs	Display the current operation interface log information.
Controller logs	Display the running log of the controller connected to the robot.
Operation logs	Display logs related to robot operation.
Log timeline	Display the log history visually through a timeline.
Internal logs	Used to display the underlying log information of the Teach Pendant or RobotAssist.
Diagnostic setting	Used to assist developers in problem diagnosis.
Hardware status	Add display of health monitoring information for the IPC and teach pendant.
Diagnostic data monitoring	Supported only on the PC HMI; this feature allows the use of condition verification function. It is unavailable on the teach pendant.

6.11Option module overview

Connect	RobotAssist software and controller connection, related operations and settings.
About ROKAE	Version information and company profile.
Software upgrade	Control system software upgrade and backup related operations.
Export	Control system configuration export.
Import	Control system configuration import.
File manager	Several folders involved in the RobotAssist software.

Demos	Demonstration of some features.

7Basic Operation of the Control System

7.1Introduction to this chapter

This chapter provides an overview of the most commonly used basic and requisite operations for robots.
Users need to read this chapter before actually using the robot.

7.2Operating mode

Robot operating mode includes manual mode and automatic mode.

7.2.1Switch manual

The manual mode is mainly used for robot programming and debugging. In manual mode, all robot motions are controlled manually by the user, and the robot will power on the motor and respond to the motion commands only when its motion is enabled (the three-position switch is in the middle position).

The manual mode is typically used to execute the following tasks:

- Jog the robot back close to the path after an emergency stop to continue running the program;
- Create and write RL programs;
- Debug the RL program, including but not limited to start, stop, single-step run, and program position update;
- Set control system parameters and calibrate frames;
- View and modify variables;

7.2.2Switch auto

The automatic mode is used for continuous automated production, in which the three-position enabling switch will be bypassed and the robot can work normally without manual intervention. When the robot is in automatic mode, the system IO signals, etc. can be used to control the robot and obtain the robot's operating status.

The automatic mode is typically used to execute the following tasks:

- Load, start, and stop the RL program;
- Return to the original programming path after an emergency stop;
- Back up the system;
- Clean the tools (as per the process requirements);
- Machine and process the work objects;

The usage restrictions in automatic mode include but are not limited to:

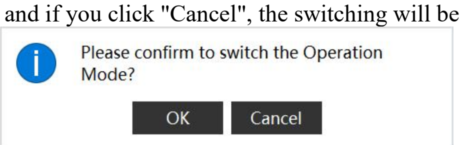
- Not allowed to Jog.
- Not allowed to modify configuration files, configure the number of IO boards, or set the robot installation method.
- Not allowed to restore the backup.
- Not allowed to grant function authorization.
- Not allowed to set soft limits.
- Not allowed to create, modify, and delete IO.
- Not allowed to perform parameter identification.
- Not allowed to turn on/off collision detection.
- Not allowed to turn on/off collaboration mode.
- Not allowed to turn on/off drag teaching in automatic mode.
- Not allowed to perform calibration.
- Not allowed to create new variables.
- Not allowed to update or restore to factory settings.

7.2.3Mode confirmation and switching

You can learn about the current mode of the control system by checking the mode icon on the bottom status bar of the HMI software.

	The controller is in manual mode.
	The controller is in automatic mode.

Users can switch between different operating modes by clicking the mode icon in the HMI interface. Before switching, a dialog box will pop up for confirmation, and if you click "OK", the system will switch the operating mode; and if you click "Cancel", the switching will be canceled.



Attention: For safety, when switching modes, the system will cut off the power supply. If the system is executing the RL program and the robot is in motion, the system will trigger STOP 1 to stop.

7.2.3.1Switching from manual mode to automatic mode

When operators need to verify the programs at all states and speeds, or when the programs are ready for full production, the system can be switched to automatic mode.
You can switch from manual mode to automatic mode in the following ways.

Way I	HMI button: Click the operating mode icon on the HMI interface to switch from manual mode to automatic mode. Before switching, a dialog box will pop up for confirmation, and if you click "OK", the system will switch the operating mode; and if you click "Cancel", the switching will be canceled. 
Way II	System IO: "automatic mode".
Way III	External communication: "switch_mode:auto"+"\r".
Way IV	Register function code: ctrl_switch_operation_auto_manual.
Way V	SDK.

DANGER

In automatic mode, the robot may be triggered to move by an external signal without any warning. Before switching to automatic mode, it is necessary to make sure that collision detection is enabled to prevent personal injury from accidental collisions between the robot and personnel!

7.2.3.2Switching from automatic mode to manual mode

You can switch from automatic mode to manual mode in the following ways.

Way I	HMI button: Click the operating mode icon on the HMI interface to switch from automatic mode to manual mode. Before switching, a dialog box will pop up for confirmation, and if you click "OK", the system will switch the operating mode; and if you click "Cancel", the switching will be canceled. 
Way II	System IO: "manual mode".
Way III	External communication: "switch_mode:manual"+"\r".
Way IV	Register function code: ctrl_switch_operation_auto_manual.
Way V	SDK.

7.3Power on and off

Please first read the introduction to enabling devices in Safety devices, Chapter 2.

7.3.1Motor on

In manual mode, the user can power on the motor by pressing the yellow three-position enabling switch on the handheld enabling device and holding it in the middle position. You can judge whether the power-on is successful by listening to the power-on sound of the robot or observing that the power-on button on the bottom status bar on the HMI software interface turns red.

Note

If the power-on fails, observe the real-time log to determine the robot's status at this time and switch the robot to a state that supports power-on before trying again.

In automatic mode, click the power-on button on the bottom status bar on the HMI software to power on the motor. The method to determine whether the motor is properly powered on in this mode is the same as that in manual mode.

7.3.2Motor off

In manual mode, the user can power off the motor by releasing or pressing the yellow three-position enabling switch all the way down to keep it in Position 1 or Position 3.
In automatic mode, click the power-off button on the bottom status bar of the Robot Assist interface to power off the motor.

Warning

In case of emergency, press the emergency stop button on the manual enabling device for emergency robot power-off. When the robot needs to be powered on again, please reset the emergency stop switch manually.

7.4Motion control

7.4.1Jog

Jog supports multiple frames/modes, as shown in the table below.

<table>
<tr><th></th><th>Explanation</th><th>Remarks</th></tr>
<tr><td>World frame</td><td rowspan="5">Cartesian space Jog, moving in the direction of a given frame. For example, if you select "world frame", Jog X, the robot will move in the X direction of the world frame; and if you select "tool frame", Jog B, the robot will rotate in the Y direction of the tool frame.</td><td></td></tr>
<tr><td>Flange frame</td><td></td></tr>
<tr><td>Base frame</td><td></td></tr>
<tr><td>Tool frame</td><td></td></tr>
<tr><td>Work object frame</td><td></td></tr>
<tr><td>Singularity avoidance</td><td>It is mainly used to avoid wrist singularities during Cartesian Jog, and all its XYZ motions are relative to the base frame.
It is necessary to Jog Axis 4 to 0° or ±180° before performing Cartesian XYZ Jog. Using J4 in this mode can quickly Jog Axis 4 to the above angle. Based on this, Axis 4 angle will be locked and no longer change for Jog XYZ, and the robot's orientation changes along with the arm plane rotation. During Jog Ry, the flange of the robot rotates around the normal direction of the plane formed by the upper and lower arms. (Jog Ry can only be performed when the Axis 4 angle is 0° or ±180°). The Jog J6 is the same as the Jog Axis 6 in joint space, and only Axis 6 is adjusted.
The base frame jog of CR series 5-axis robots corresponds to the singularity avoidance mode of 6-axis collaborative robots, that is, during Jog X/Y/Z, the orientation of the robot will change along with the arm plane rotation. During Jog Ry, the robot flange will rotate around the normal of the arm plane. Jog J5 is the same as Jog Axis 5 in joint space.
Note: When full DH compensation is enabled, singularity avoidance cannot be achieved. It is recommended to use this feature with full DH compensation disabled.</td><td rowspan="2">You need to first turn on the "Stacking Debugging Mode" on the "Advanced Settings" page under "Motion Parameters" of "Settings". The current version is applicable to robot models: industrial standard six-axis series (XB, NB models) and xMate CR/SR collaborative series.</td></tr>
<tr><td>Parallel base</td><td>It is mainly used to avoid wrist singularities during Cartesian Jog, and all its XYZ motions are relative to the base frame.
It is necessary to first Jog the flange in a state parallel to the base before performing Cartesian XYZ Jog. Using J4 and Ry in this mode can quickly make the flange reach a state parallel to the base. During Jog XYZ, the robot's orientation does not change, but the singularities of the robot's wrist can be automatically avoided. The usage of J4, Ry, and J6 in this mode is consistent with that in the singularity avoidance mode.
CR Series 5-axis robots use Ry in this mode to quickly make the flange parallel to the base.
Note: When full DH compensation is enabled, singularity avoidance cannot be achieved during jogZ.</td></tr>
<tr><td>Joint space</td><td>Each axis movement is controlled individually.</td><td></td></tr>
</table>

Note

Due to the configuration limitation of the xMate series 5-axis robots, when Jog xyz is performed in the ordinary frame, the robot's orientation will change along with the arm plane rotation. During Jog A, the robot flange will rotate around the normal of the arm plane. Jog B is the same as Jog

Axis 5 in joint space. The C button is invalid.

Jog supports two modes: continuous and stepping:

- In continuous Jog mode, when the robot is powered on and the jog button is held down, the robot will move continuously at the set Jog velocity until either the enabling switch or the Jog button is released.
- In stepping Jog mode, after the robot is powered on, every time the Jog button is pressed, the robot moves for a given step length. Users can choose the appropriate step length according to their needs, which is mainly for accurately adjusting the pose of the robot.

> Note
>
> In stepping Jog mode, it is necessary to hold down the Jog button and wait until the robot moves for the specified step length before releasing the Jog button. A short press can make the robot stop moving in advance.

Jog speed setting:
Jog speed can be set to control the robot motion speed during Jog, with the speed range from 0.1% to 100% (100% corresponds to the robot's top TCP speed of 250 mm/s). Subject to safety regulations, the TCP linear speed in both Cartesian space Jog and joint space Jog shall not exceed 250mm/s.

7.4.1.1Jog low-speed mode

To facilitate users' fine adjustment of motion points, in the Continuous Jog mode, if the Jog speed is configured to 1% or below, the robot will enter Jog low-speed mode.
In this mode, the robot's motion speed will be locked at 20%, while the step length for single Jog clicks will follow the table below.

Jog Speed	Joint Space JOG	Cartesian JOG
1%	1.0°	1.0 mm
0.5%	0.5°	0.5 mm
0.1%	0.1°	0.1 mm

7.4.2Quick adjustment

The right operation interface of the HMI provides a quick pose adjustment function. The supported quick pose adjustments include: Initial Pose, Drag Pose, Shipping Pose, and Home Pose.
The quick pose adjustment is available in manual mode in a way similar to Jog operation. In manual mode, the robot is powered on via the enabling switch. When the button for the corresponding target pose is pressed, the robot will move to the target pose in the joint space. The motion speed can be adjusted via the Jog speed.
Drag Pose, Shipping Pose, and Home Pose support user customization, which can be set on the "Quick Adjustment" page under the "Settings". If their parameters are not set, their default configuration is used.

7.4.3Drag

During point teaching, the programming time can be greatly shortened by dragging and positioning; Robot dragging, combined with trajectory recording and trajectory reproduction, can simplify the difficulty of teaching in some continuous trajectory scenes.
Please refer to the Appendix for the introduction and usage examples of the end-effector handles of each collaborative robot model.

Attention: The Drag Mode and its extended functions (end-effector handle, point teaching, continuous trajectory teaching, and trajectory reproduction) are only available for xMate series cobots.

Drag mode	Type	Explanation
Joint space		Each axis moves independently and can be directly adjusted to reach the desired pose;
Cartesian space	Translational drag only	The robot can be guided to translate along all directions of Cartesian space;
	Rotational drag only	The robot can be directly guided manually to rotate around TCP;
	Free drag	Translation and rotation are supported in Cartesian space;

When the robot is in manual mode and powered off, turn on the drag enabling switch on the operation interface, the robot is powered on automatically and enables Drag Mode. Press the enabling button on the end-effector drag handle simultaneously to drag the robot for point teaching and trajectory recording.

Note

The appropriate drag way and drag space are set before the Drag Mode is enabled. After Drag Mode is enabled, the robot will be powered on automatically. In this case, the drag mode and drag space cannot be set, and you need to turn off Drag Mode before setting them.

Warning

Before enabling Drag Mode, please make sure the robot's dynamic parameters and load parameters are set accurately. Otherwise, enabling Drag Mode may fail, or the robot may float during dragging.
Set the parameters using the dynamic parameter identification function and the load identification function provided by the system.

DANGER

The following parameters must be set correctly before the drag teaching is used, otherwise, when the angle of each axis is in the wrong state, the controller cannot calculate the correct output torque, and the robot drag function cannot be used normally.

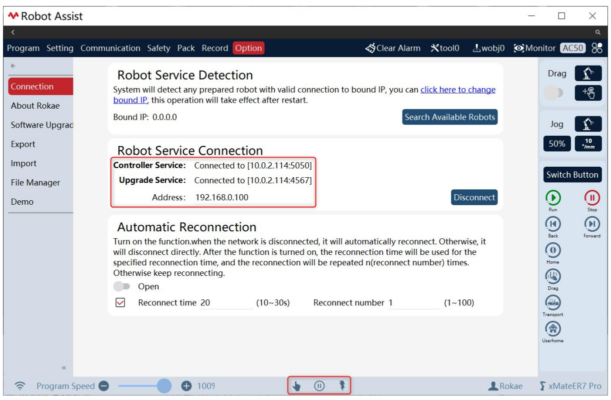
- Robot model.
- Robot installation method: floor mounting or ceiling mounting.
- Dynamic parameters of the robot and its load.
- Mechanical zero calibration.

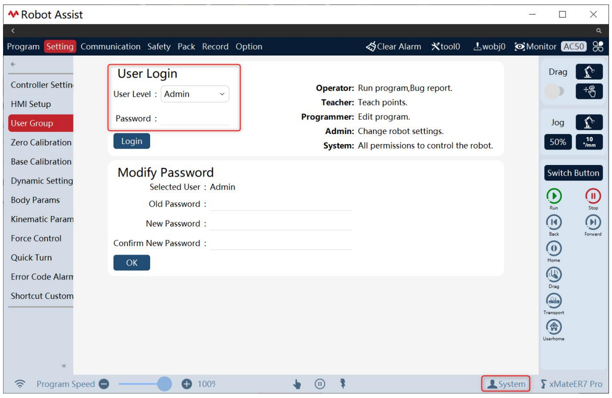
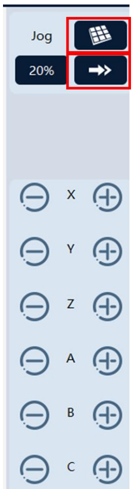
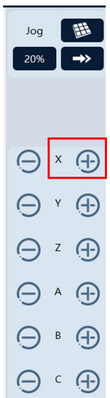
7.5Continuous trajectory playback

After a successful continuous trajectory teaching, playback the recorded trajectory on the recording interface and confirm, and then save it manually after confirmation.
Check Loop in the playback mode for looped playback.
The playback rate can be set between 1% and 300%. It is recommended that users set the playback rate in the range of 1% to 100%. When the playback rate is greater than 100%, a following error of the drive may occur.
For operation examples, refer to the section below, "Programming"-"Path list".

7.6Operation example I: Industrial robots realize Jog motion

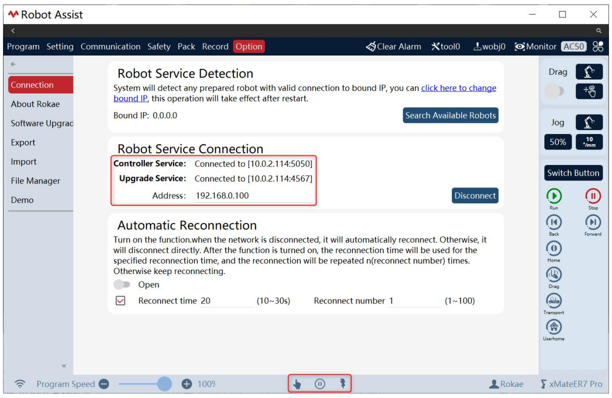
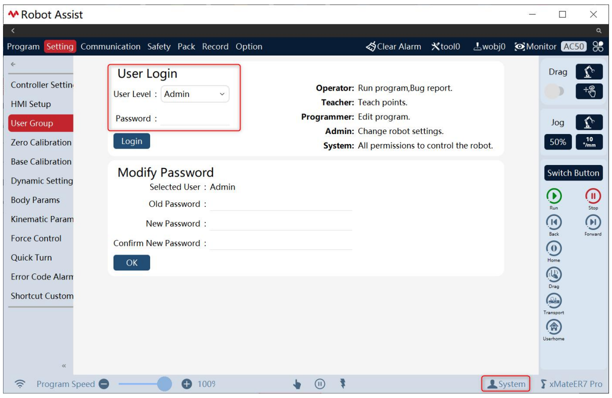
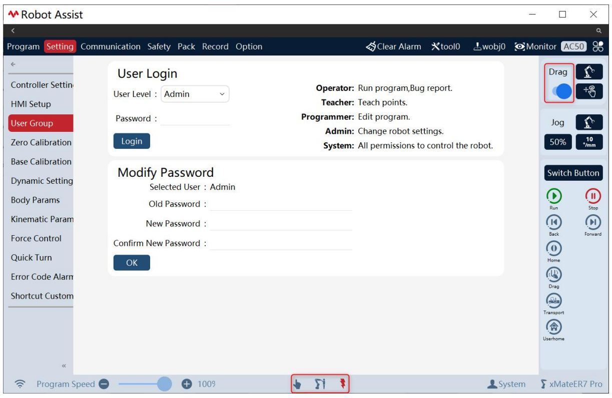
Before the real robot operation, it is necessary to ensure that the HMI is connected to the robot and both are powered on and operating normally. (see Chapter 5 for details).

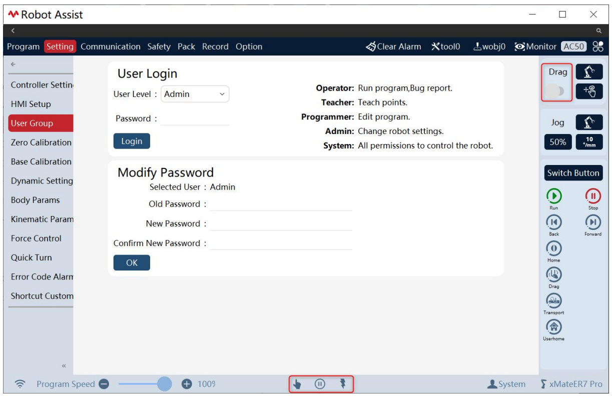
Step	Graphical Representation	Explanation
1. Check robot status. Ensure that the robot is normally connected and is in manual mode;		

2. Switch login user.		
3. Select [Base Frame] as the Motion Frame. The specific frames are detailed in Chapter 4. Also, choose [Continuous Motion].		The movement direction of the manipulator end-effector is the same as the direction of the base frame.
4. Hold the enabling switch of the Teach Pendant.		When the manual power-on is successful, there will be an obvious power-on sound, and the robot status bar will show that the robot is in the manual power-on state. During jog, it is necessary to keep the enable switch pressed.
5. Jog the robot. Click on the teach pendant to select the desired direction for movement. This example demonstrates moving in the positive X-direction of the base frame.		In continuous jog mode, it is necessary to ensure that the motion button remains pressed. Once the motion button is released, the robot will stop moving.
6. Turn off Jog. Release the robot motion button, and release the enabling button.		After the manual power-off, the robot status bar will show that the robot is in the manual power-off state.

7.7Operation example II: CR collaborative robots realize drag

Before the real robot operation, it is necessary to ensure that the HMI is connected to the robot and both are powered on and operating normally. (see Chapter 5 for details).
This section demonstrates the drag function of collaborative robots mainly based on the xMate CR7 collaborative robot.

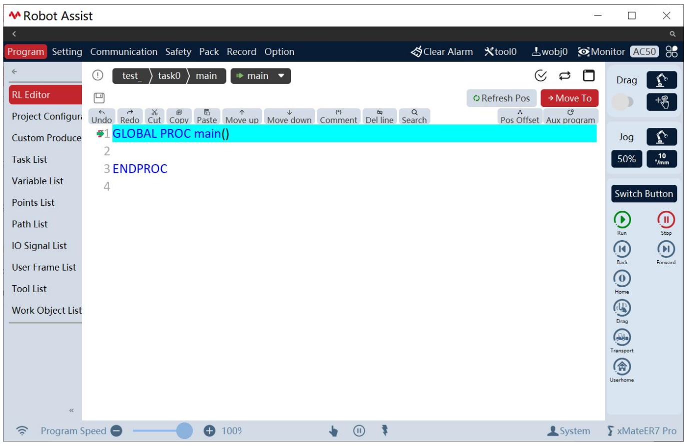
Step	Graphical Representation	Explanation
1. Check robot status. Ensure that the robot is normally connected and is in manual mode;		Make the robot in Drag Pose as much as possible. Make sure that the drag button is set to the non-selectable state when the robot is in automatic mode.
2. Switch login user.		
3. Click [Drag] button. The drag button will show that the drag button is activated, the robot status bar will show that the robot is in Drag mode, and the robot will make an obvious power-on sound;		

4. Press and hold the drag enabling switch of the robot to drag the robot to any position in the robot's workspace;		During drag, hold down the drag button to perform drag operation.
5. Release the robot drag enabling switch and click the [Drag] button. The drag button will show that the drag button is turned off, the robot status bar will show that the robot is in manual power-off mode, and the robot will make an obvious power-off sound.		Other operations are allowed only after the drag state is exited.

8Programming

8.1Introduction to this chapter

Industrial/collaborative robots are highly flexible production tools that can be programmed by users to meet different needs.
This chapter will introduce all aspects of programming the xCore control system.
Starting with this chapter, users will gradually gain an in-depth understanding of advanced use methods such as xCore programming, setting, and communication.



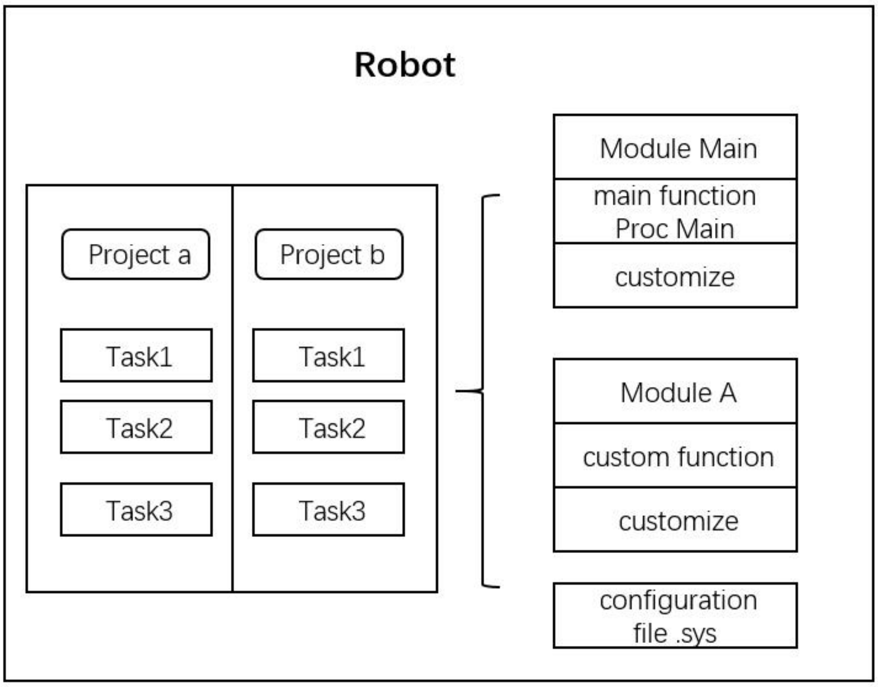
8.2Introduction to project

xCore manages users' programming on a project basis. A typical project includes RL program, custom user interface, tasks, variables, points, paths, IO, user frame, tool frame, and work object frame.

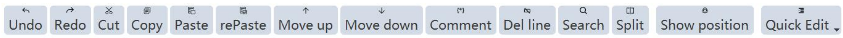
It is divided into four levels according to the range size:

Project	Project	The highest level, including various information. A project can contain multiple tasks, each of which is independent and interacts with each other only by the interfaces provided; The robot can only select and execute one project at a time;
Task	Task	A task can contain multiple program modules, but there is only 1 main.mod; In the main.mod, there is a GLOBAL PROC main, which serves as the entry function for the entire project. Loading and executing a project is essentially executing the main function;
Module	Module	It is divided into program module (.mod) and system module (.sys), and a module is a program file; Each program module contains certain data variables and functions, which are used to realize specific robot functions; Routine operations such as copying and deleting can be performed for each program file; The module is defined as: PROC main() … ENDPROC PROC test1() … ENDPROC PROC test2() … ENDPROC
Function	Routine	Users can call robot functions or other modules according to their own needs within the function; The function can be defined by users. Different custom functions can be saved in the same program file, or be saved in different program files;

The interrelationship between projects, tasks, program modules, and functions is shown in the following:



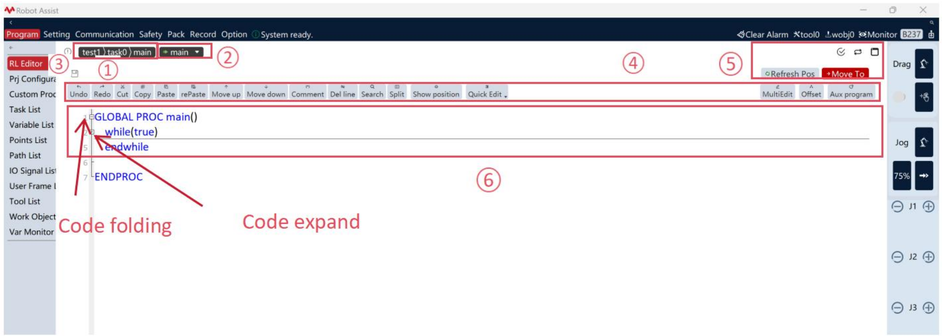
> Note
>
> In each module, the code area located in front of the file and before the first function is called the declaration area, which is used to store variable declarations for the GLOBAL and LOCAL scopes. Variables defined in this area will be reinitialized each time the pptomain is executed.
> For example: int0 is defined in the variable list with an initial value of 0; while int1 is defined in the declaration area with an initial value of 0.
> When the loop runs, int0 is incremented by 1 after each run, and its print value is "1, 2, 3..."; while int1 will be reinitialized to 0 each time it runs, and its print value is "1, 1, 1...".




```
//Statement
int int1 = 0;
//
GLOBAL PROC main()
   int0++;   //int0 is defined in the viriable list
   int1++;   //int1 is defined in the statement area
   Print("int0=",int0);
   Print("int1=",int1);
   Pause;
ENDPROC
```

8.3RL editor

8.3.1Overview



①	Selection, switching, and editing of project, task, and program files.
②	Program debugging quick positioning button, supporting (1) Quick positioning to the main function; (2) Quick positioning to the cursor; (3) Quick positioning to a certain function of the current program module.
③	Save the current task.
④	Program edit toolbar: Undo, Repeat, Cut, Paste, Copy, Reverse paste, Move up one row, Move down one row, Batch comment, Delete current row, Find, View location, Quick edit, Batch modify, Offset, and Auxiliary programming.
⑤	Updating the position information of a point, moving to a certain point, program syntax check, loop mode, and output terminal.
⑥	Program editing area, where RL commands are edited through auxiliary programming and other methods.

8.3.2Tool introduction

①	Project synchronization identification. Red identification means that the loading program in the controller is not synchronized with the current program on the HMI. Gray identification means that the loading program in the controller has been synchronized with the current program on the HMI.
②	Current project: Click to switch to the project configuration page.
③	Current task: Click to switch between tasks.
④	Current program module: Click to switch between program modules.
⑤	Clicking and dragging the dropdown clamp of the pointer allows you to select the target for the pointer to move to. You can move the program pointer to the Main function (equivalent to program reset), move the program pointer to the line where the cursor is located, or move the program pointer to a specific function.
⑥	Display the save status. Clicking this button will immediately save the program locally and push it to the controller. Green: The program has been edited but is not synchronized to the controller. Gray: The program has been synchronized to the controller.
⑦	Undo the previous operation.
	Redo the previous operation undone.
	Cut the selected line of code. Multiple lines can be cut at the same time.
	Copy the selected line of code. Multiple lines can be copied at the same time.
	Insert the copied or cut content into the line of the cursor.
	Paste copied lines in reverse order: For example, if lines 1, 2, and 3 are copied, pasting

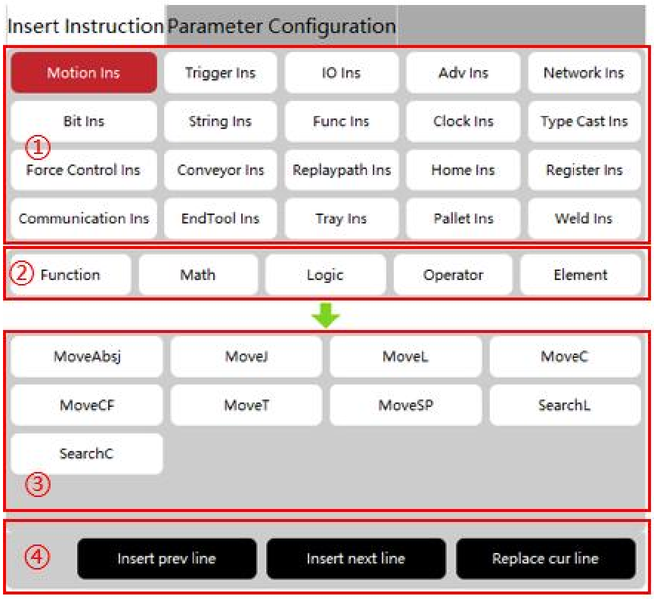
	will display them as lines 3, 2, 1.
	rePaste copied lines in reverse order: For example, if lines 1, 2, and 3 are copied, pasting will display them as lines 3, 2, 1.
	Move the selected code line up one line. Multiple lines can be moved up at the same time.
	Move the selected code line down one line. Multiple lines can be moved down at the same time.
	Comment or uncomment the selected code.
	Delete the selected line or the line where the cursor is located.
	Find keywords.
	Show position: See the "Show position function" section below for details.
	Split screen: See "Split screen function" below for details
	Quick edit - One-click format: Format all programs in the current project (e.g., add indentation, align operators).
	Quick edit - One-click expand: Expand all loaded programs in the current RL editor.
	Quick edit - One-Click collapse: Collapse all loaded programs in the current RL editor.
⑧	Check the current program for specific obvious errors, such as duplicate function names and missing key identifiers. It cannot check out all syntax errors.
⑨	Loop mode switching: The program executes in a loop. The program only runs once.
⑩	Display Print information and syntax information.
⑪	Update the pose of the point on the line where the cursor is located to the current pose. Note: Simply position the cursor on the line containing the point.
⑫	"Move to" function, see the "Move to function" section below for details.
⑬	Batch modify: Perform batch modifications to speed, turning zone, tool, and work object parameters.
⑭	Point offset tool. See "Point offset tool" below for details.
⑮	Auxiliary programming tools.

8.3.3Auxiliary programming

The auxiliary programming interface can assist programmers in quickly building program frameworks, inserting program commands, and changing command attribute configurations. The auxiliary programming interface includes two parts: Insert Command and Attribute Settings.

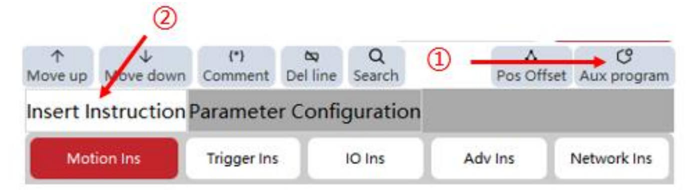
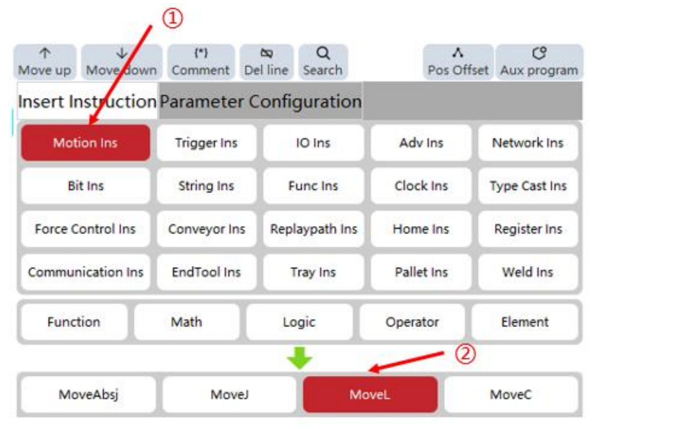
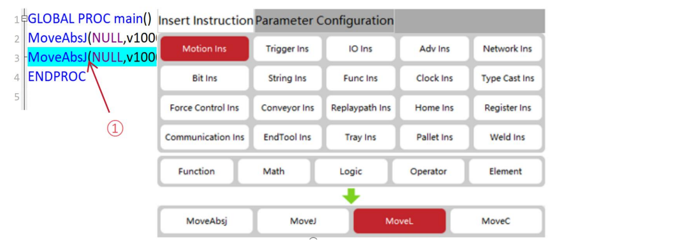
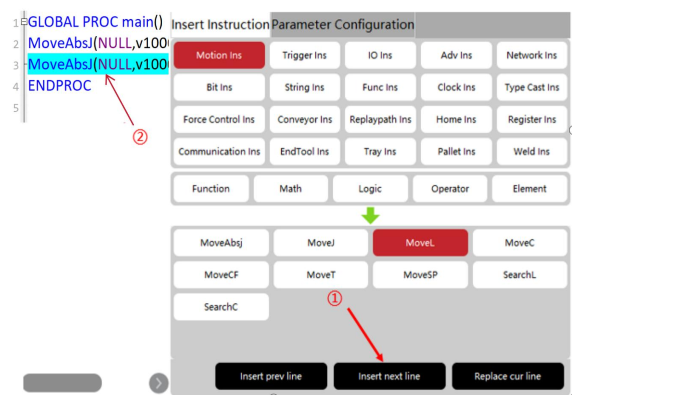
8.3.3.1 Insert command

Insert Command is responsible for inserting desired commands into program text.

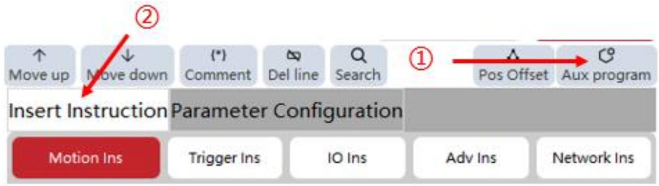
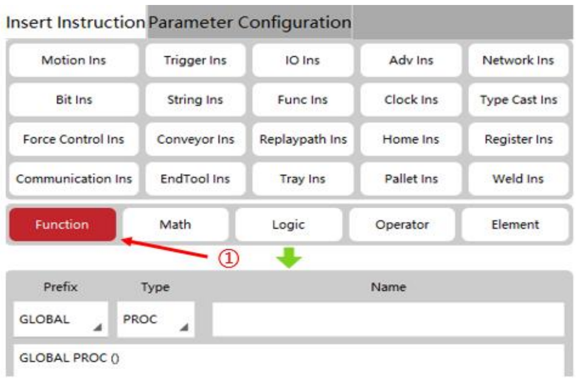
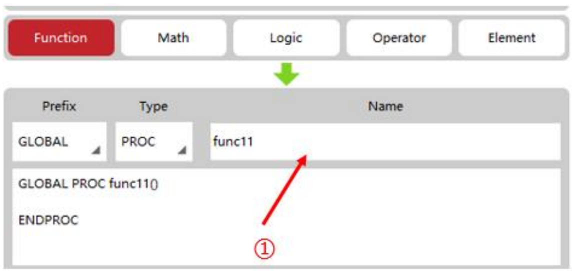
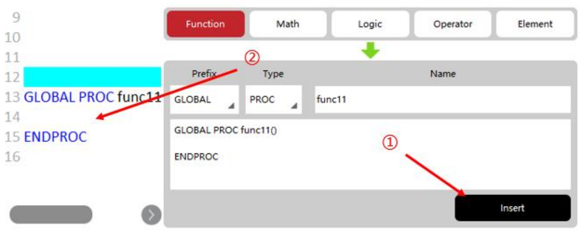


S/N	Explanation
①	Program command group selection area;
②	Program common element selection area;
③	Program command/element selection area, for selecting a subdivided command;
④	Command/selection insertion confirmation area, used to insert the default parameters or elements of the selected command into the project text;

8.3.3.1.1 Example I: Insert MoveL command

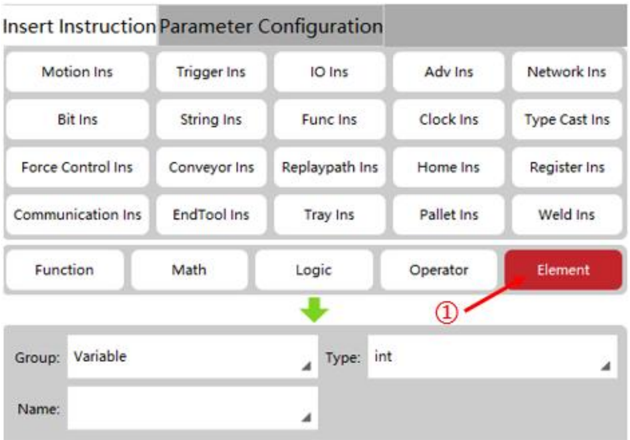
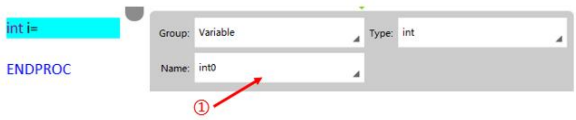
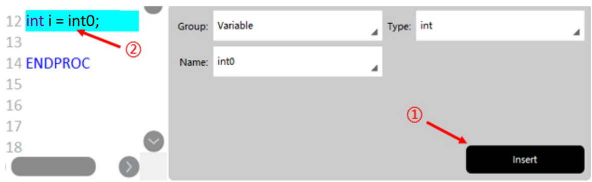
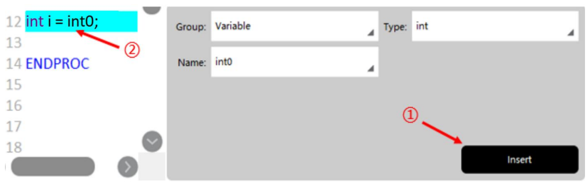
Step	Graphical Representation	Explanation
1. Click the "Auxiliary Programming" -> "Insert Instruction" button to open the Insert Instruction interface;		To open the Insert Instruction interface.
2. Select "MoveL" from "Motion Commands";		After selecting "Motion Commands", other commands will be hidden. Then, select the "MoveL" command.
3. Select the previous line of the desired insert line using the cursor;		To select the desired insert line.
4. Click the "Insert Next Row" button;		To insert the command into the line below the selected line in "3".

8.3.3.1.2Example II: Insert a function

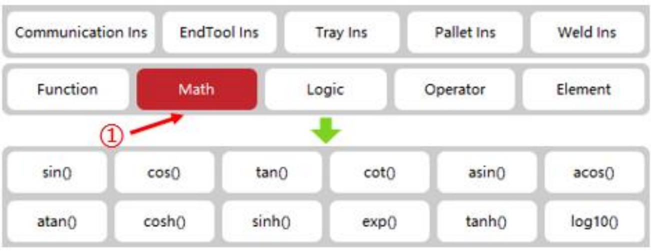
Step	Graphical Representation	Explanation
1. Click the "Auxiliary Programming" -> "Insert Instruction" button to open the Insert Instruction interface.		To open the Insert Instruction interface.
2. Select the "Function".		
3. In the input box below "Name," fill in the desired function name.		
4. Click the "Insert" button.		After clicking the "Insert" button, the text for the corresponding function is created at the end of the text.

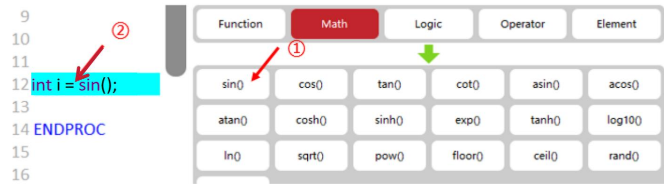
8.3.3.1.3Example III: Insert an element

Step	Graphical Representation	Explanation
1. Click the "Auxiliary Programming" -> "Insert Instruction" button to open the Insert Instruction interface;		To open the Insert Instruction interface

2. Select the "Element";		
3. Select the variable and other information you want to insert in the "Element" interface;		
4. Move the cursor to the desired text position;		
5. Click the "Insert" button;		After clicking the "Insert" button, text will be generated at the corresponding position of the cursor

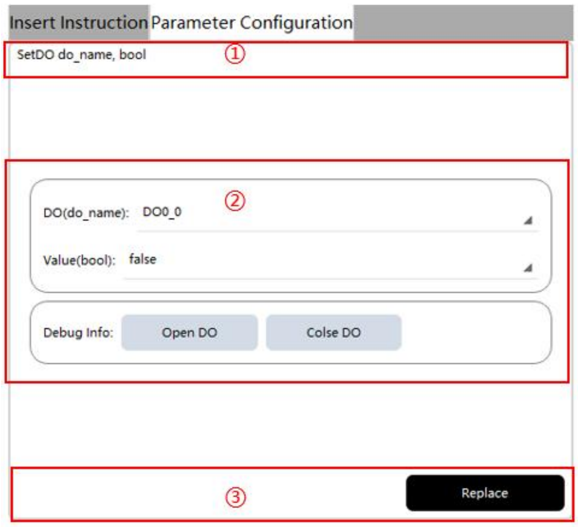
8.3.3.1.4Example IV: Insert math/logic/operator

Step	Graphical Representation	Explanation
1. Click the "Auxiliary Programming" -> "Insert Instruction" button to open the Insert Instruction interface;		To open the Insert Instruction interface
2. Select the "Math";		
3. Move the cursor to the desired text position;		

4. Click the "sin()" button;		After clicking the corresponding button, the text sin () will be generated at the corresponding position of the cursor;

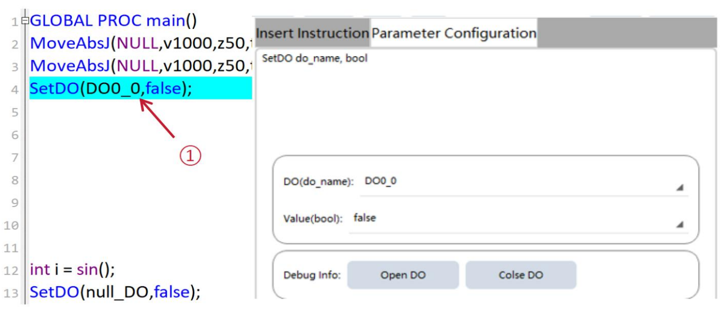
8.3.3.2 Attribute settings

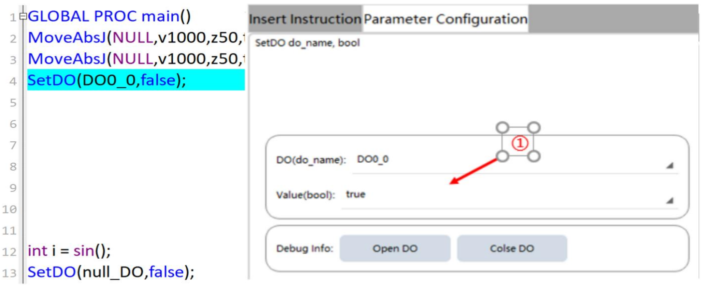
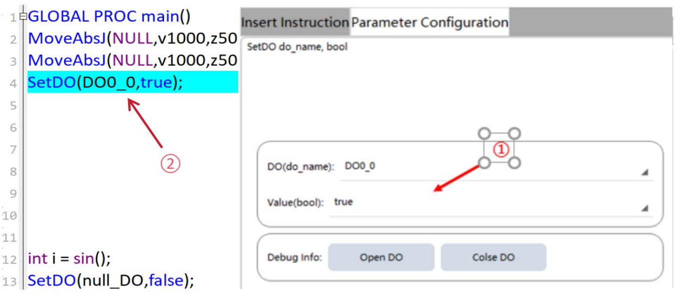
Attribute Settings is responsible for updating the command parameter information for the selected line.



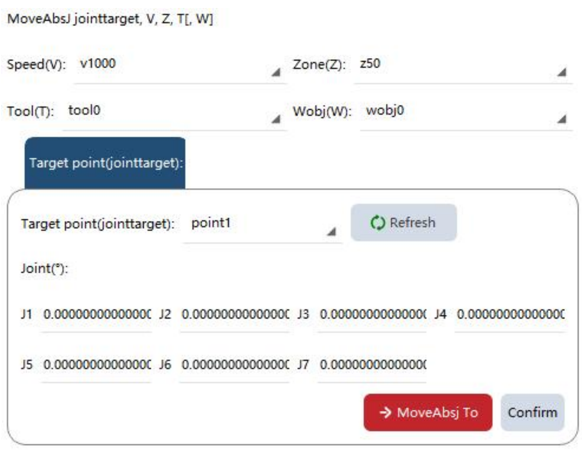
S/N	Explanation
①	Attribute definition area, which describes the parameter name of the current command
②	Attribute selection area, for modifying the parameters of the selected command
③	Attribute confirmation area. Click the "Replace" button to replace the command parameters in the current line in the program text

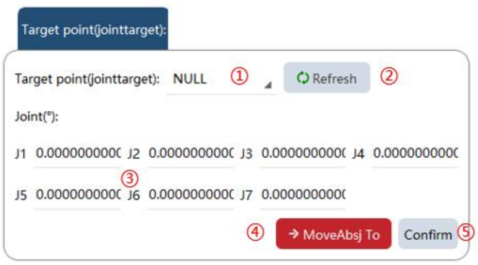
8.3.3.2.1 Example I: Configure the SetDO command attributes

Step	Graphical Representation	Explanation
1. Select the desired line containing the command that needs to be modified in the text;		After opening the Attribute Settings interface, the details of the command will be expanded in the interface

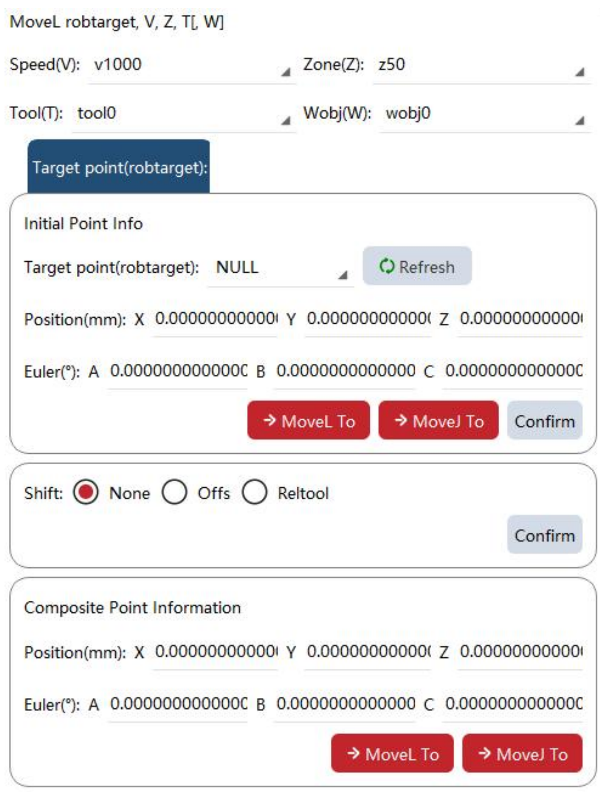
2. Select the desired parameters in the "Attribute Settings" interface;		
3. Click the "Replace" button		After clicking the "Replace" button, the parameters of the command in the selected line are updated to the set parameters

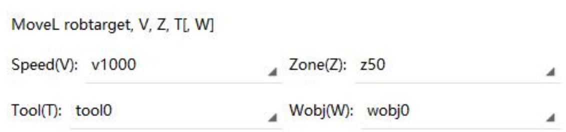
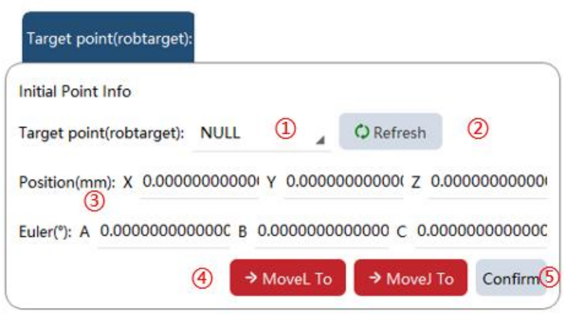
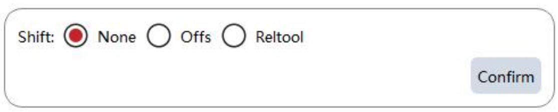
8.3.3.2.2 Example II: Configure the MoveAbsJ command attributes

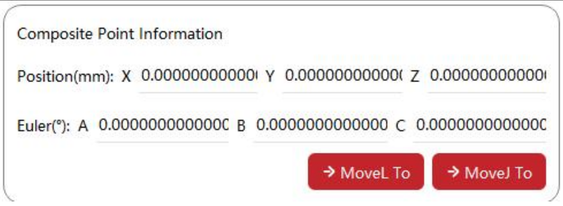


S/N	Graphical Representation	Explanation
1. Motion parameter settings;		Set the speed, turn zone, tool, and work object parameters of the MoveAbsJ command.
2. Point setting;		1. Select the point; 2. Update the pose information of the selected point to the robot's current pose; 3. Point information can be manually modified; 4. Manually Jog the robot to the selected point; 5. Click the "OK" button to update the data modified in 3 to the point data;

8.3.3.2.3 Example III: Configure the MoveL command attributes



S/N	Graphical Representation	Explanation
1. Motion parameter settings;		Set the speed, turn zone, tool, and work object parameters of the MoveL command;
2. Point setting;		1. Select the point; 2. Update the pose information of the selected point to the robot's current pose; 3. Point information can be manually modified; 4. Manually Jog the robot to the selected point; 5. Click the "OK" button to update the data modified in 3 to the point data;
3. Offset setting		
4. Offset point debugging interface.		This interface is non-editable and is intended for use after applying an offset through Jog settings.

8.3.4Point offset tool

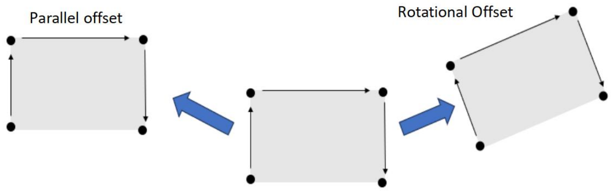
8.3.4.1Overview

This tool allows for global offset modification of taught points within a user-selected range of motion commands in an RL program. The current controller version supports three offset methods: Program offset, Angle offset, and Mirror offset.

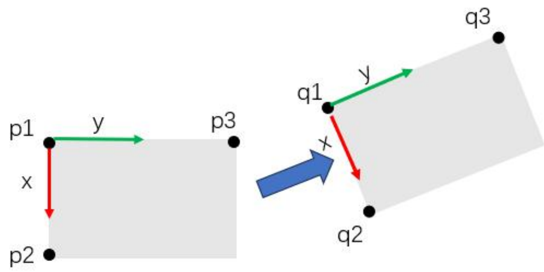
8.3.4.2Program offset parameters introduction

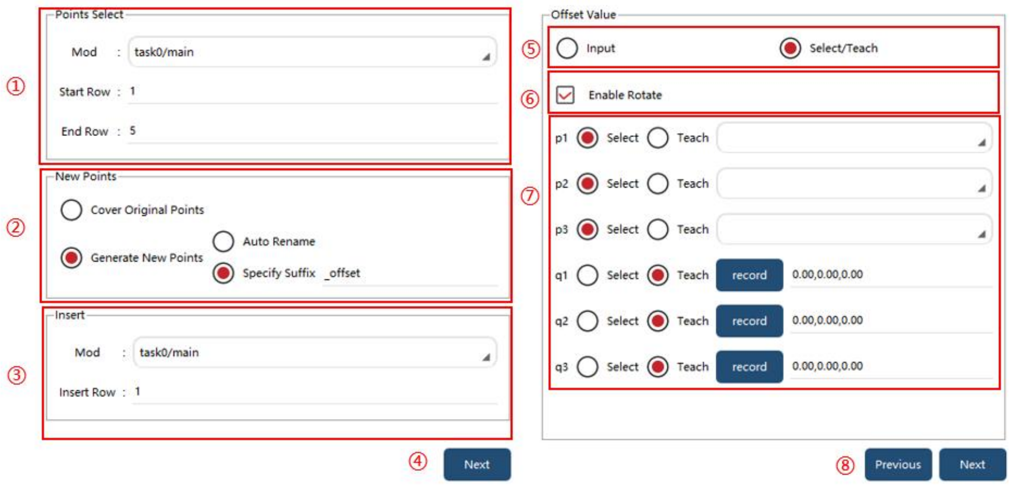
This tool allows for the global offset of taught points within a user-defined range of motion commands in the RL program.

When the work object is required to be translated only, the parallel offset is used, and only the xyz values are adjusted. When you want to adjust both the position and attitude of the work object, after the "Orientation Variable" is turned on, the orientation values ABC will also be adjusted.



Principle: Calculate the relationship between the original frame and the offset frame through 6 points including p1, p2, p3, q1, q2, and q3. When "Pose Variable" is not turned on, only p1 and q1 need to be set.





①	Point Selection: Select the position of the point that needs to be offset, and specify task and mod, start line, and end line. Then, this tool will search for the points involved in all

	commands from the start line to the end line. Attention: Commands commented out with "//" are not searched.
②	New point generation method: You can choose between "Overwrite Original Point" or "Generate New Point". If "Generate New Point" is selected, you can choose the naming method for the new point: "Automatic Renaming" or "Original Point Name+Specified Suffix".
③	Insertion position: Specify the position where the newly generated command will be inserted. By selecting either task or mod, the generated command text will be created starting from the designated insert line.
④	Click the "Next" button to go to the Offset setting page.
⑤	Offset matrix generation method: "Input" or "Select/Teach" can be selected, and the "Pose Variable" is supported only in the latter.
⑥	Choose whether the pose is variable or not. If not, only translation is considered.
⑦	Under the "Select/Teach" condition, 2 points or 6 points used to calculate the pose need to be specified. For each point, you can select the "Select", and then select the existing points in the corresponding pull-down list. You can also select "Teach" and click the "Record" button to record the current position of the robot.
⑧	Click the "Back" button to enter the Point Selection page. Click the "Next" button to start calculating the offset points.

Use restrictions:

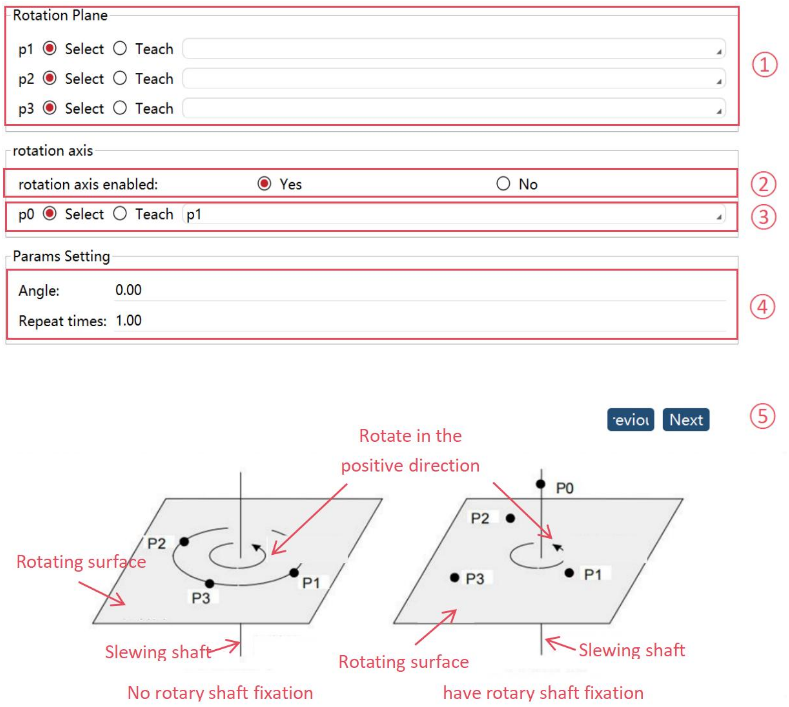
1. It is recommended to keep the number of offset points per time within 1000, otherwise, interface freezing or loss of points may occur.
2. When the system calculates an offset point, it will automatically check whether the point is reasonable, and if not, an error message will be displayed, and the generation of commands will fail. But the points before the unreasonable points will still be generated.

Cause of point unreasonableness: ① Parameter error ② Singular position ③ Point unreachable ④ Other errors.

3. If there is no point between the start line and the end line of the specified mod, an error message, "No point to be offset found", will be reported.
4. The robot needs to be connected when this tool is used, otherwise, an error message of "robot communication error" will be reported.
5. The angle between the vectors formed by any two points of p1, p2, and p3 shall be greater than 1°, otherwise, it will be judged that the three points are collinear and an error message of "reference point error" will be reported; The same applies to q1, q2, and q3.
6. The point name in the command needs to be consistent with that in the point list (case-insensitive), otherwise, an error message of "point xxx not found" will be reported.
7. When the tool page is open, and when you perform operations such as adding or removing tasks, adding or removing points, editing RL, and switching between projects on the "Task List", "Point List", and RL editing pages, this tool will not automatically refresh. It is necessary to close and reopen the tool page manually for refreshing this tool.

8.3.4.3Angle offset parameters introduction

This function provides angular offset capability for Cartesian points. It allows for rotational offset of taught points within a user-defined range of motion commands in the RL program by specified angles.



The point selection for Angle offset is identical to Program offset. After the effective range is selected, perform the following operations to complete the angular offset.

①	Define the rotation plane: The user needs to specify three points (P1, P2, P3) to determine the reference rotation plane for the Angle offset function. These three points can be selected from the point list or confirmed through teaching.
②	Disable rotation axis: Set [Use rotation axis] to "No". The system will calculate a circle based on the 3 specified reference points P1, P2, and P3 (all lying on the circumference), with the rotation axis passing through the circle center perpendicular to the circular plane and the rotation direction following the vector from P1 to P2.
③	Enable rotation axis: Set [Use rotation axis] to "Yes". The system will calculate a rotation plane based on the 3 specified reference points P1, P2, and P3. Then, specify an additional reference point P0, where the rotation axis is defined as the line passing through P0 and perpendicular to the rotation plane, with the rotation direction following the vector from P1 to P2.
④	Set offset angle and repeat count: This function will rotate the points multiple times, generating a new set of motion commands for each repetition.

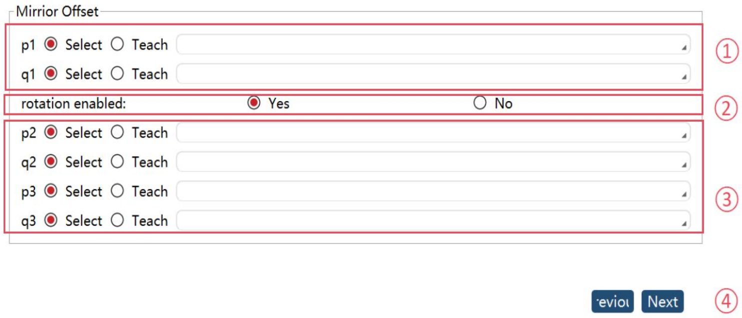
Note:

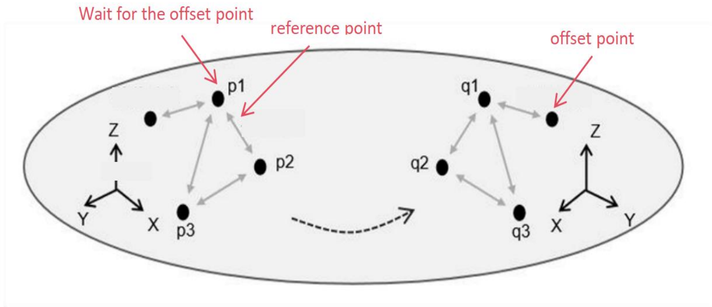
1、This function does not support angular offsets for joint-space points

2、After performing angular offsets, the orientation of robot points in the new RL program may differ from the original program. Always verify through manual slow-speed single-step execution and individual point confirmation before initiating continuous operation

8.3.4.4Mirror offset parameters introduction

This function provides mirror offset functionality for Cartesian points. Users can select the points to be offset and apply mirror offsets according to specified rules.

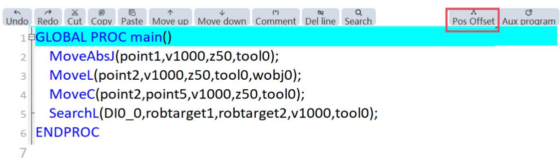
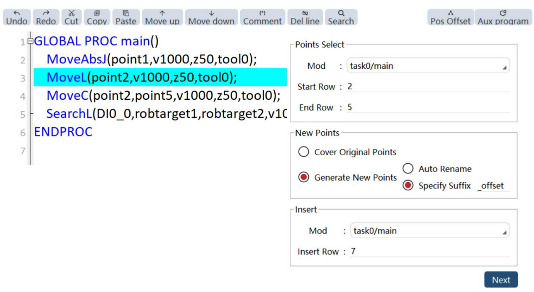


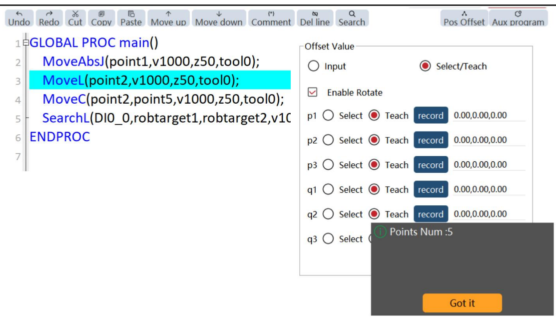
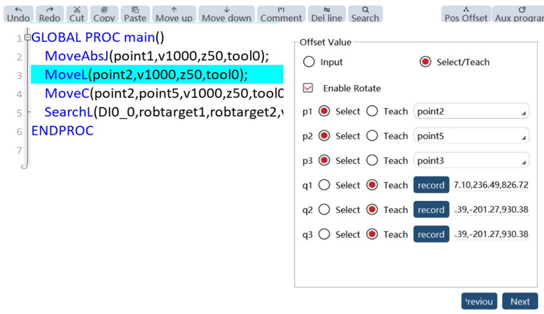
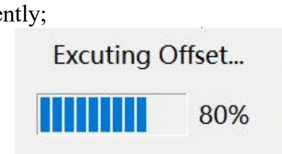
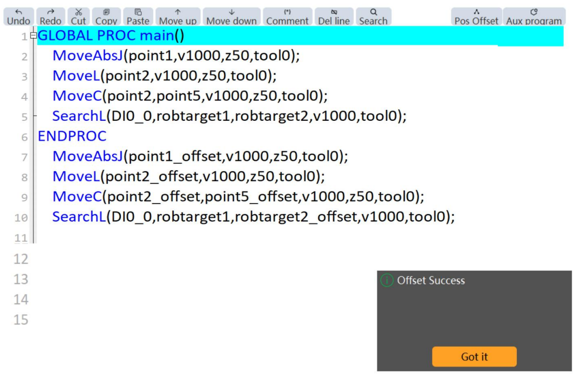
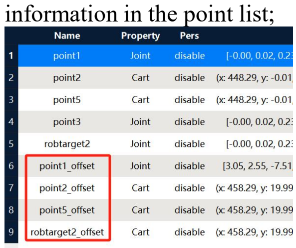


The point selection for Mirror offset is identical to Program offset. After the effective range is selected, perform the following operations to complete the mirror offset.

①	Select mirror points: Choose reference points p1 and q1 as the mirror basis. These points can be selected from the point list or taught manually.
②	Rotation disabled: Set [Use rotation] to "No". The system will calculate the mirror plane based on the positions of p1 and q1. The position of the point to be offset will be mirrored across this plane, while its orientation remains unchanged before and after the offset.
③	Rotation enabled: Set [Use rotation] to "Yes". Additional reference points p2, q2, p3, and q3 must be specified. The triangle formed by (p1, p2, p3) must be symmetrical to the triangle formed by (q1, q2, q3) across the intended mirror plane. In this mode, both the position and orientation of the points to be offset will be mirrored across this plane, resulting in symmetrical orientations before and after the offset.

8.3.4.5Operation examples

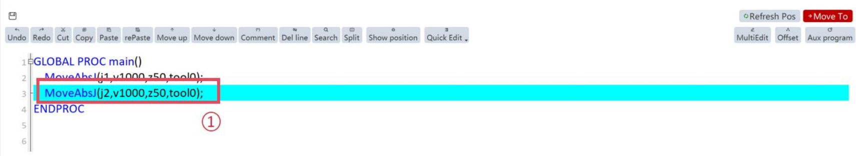
Step	Graphical Representation	Explanation
1. Click the "Point Offset" button to open the tool;		It can only be used in manual mode with the permission level of "Programmer" and above;
2. Select the point to be offset, set the parameters, and click the "Next" button;		Here, select the points of lines 2 to 5 in the "main" of the "task0" task; The point generation method is "generate a new point", with the name suffix of the new point needing to be specified; The insertion position selects the 7th line of the "mod" file;

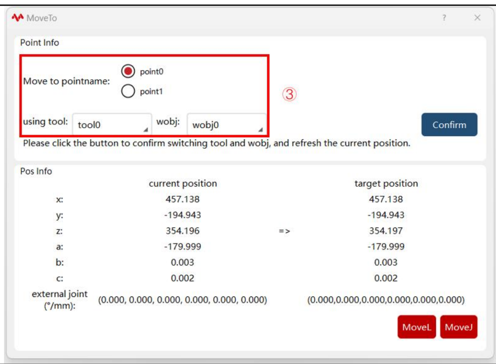
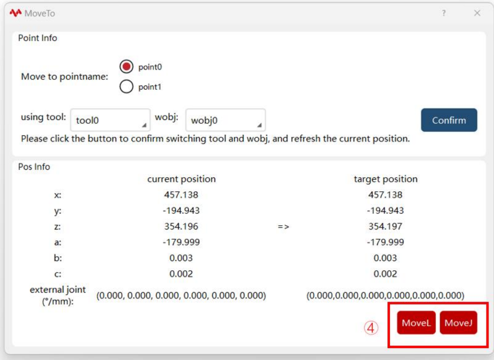
3. Prompt "number of points to be offset";		The points to be offset here are point1 in line 2, point2 in line 3, point2 in line 4, and robtarget2 in line 5. Note: 1. The "point2" is calculated twice, but it actually will be offset only once, so only one offset point will be generated.
4. Set the offset calculation method and reference point, and click the "Next" button;		As the progress bar pops up during the execution of the offset, please wait patiently; 
5. Prompt "offset successful". The contents of the 7th to 10th are newly generated commands;		You can see the generated point information in the point list; 

8.3.5Move to function

"Move to" Function: This function enables the robot to move to the selected point by executing a series of operations;

8.3.5.1Operation examples

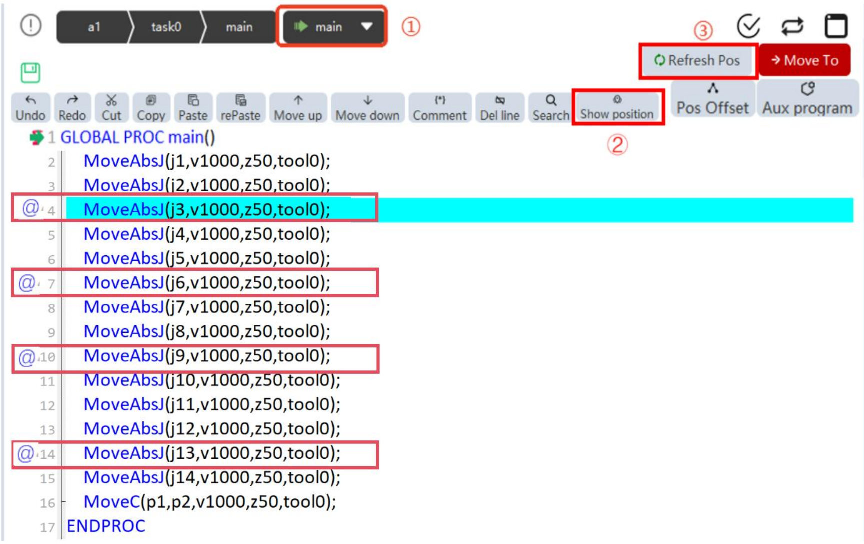
Step	Graphical Representation	Explanation
1. In the RL editor interface, select the corresponding motion command;		It can only be used in manual mode with the permission level of "Programmer" and above;
2. Click the "Move to" button in the top-right corner;		

3. In the "Move to" pop-up window, select the corresponding tool, work object, and point;			Single-point motion command: No need to select a point; Multi-point motion command: Confirm the selected points; Choose the tool and work object for the move to;
4. In the "Move to" pop-up window, select the target point and execute the "Move to" function.			Select "MoveJ," "MoveL," or "MoveAbsJ" (displayed only for joint angle points) in the bottom-right corner, and long press to trigger the "Move to" function.

8.3.6Show position function

8.3.6.1Overview

"Show position" function: This function is primarily designed for users to identify the current stop position of the robot and determine which line of the RL program it corresponds to.



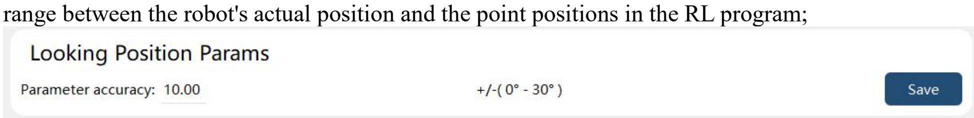
Triggering the show position function can be done in the following ways:

1. Clicking the "PPToMain" button;
2. Clicking the "Show position" button;
3. Clicking the "Refresh Pos" button;

When the "Show position" function is triggered, a  symbol will be displayed on one or more lines in the RL program interface, indicating that the robot's current position matches the point position used on those lines.

8.3.6.2Show position parameter settings

Refer to the "Settings" -> "HMI Settings" -> "Looking Position Params" to configure the tolerance range between the robot's actual position and the point positions in the RL program;

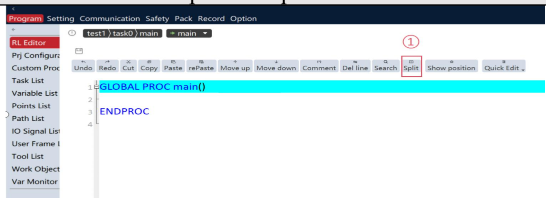
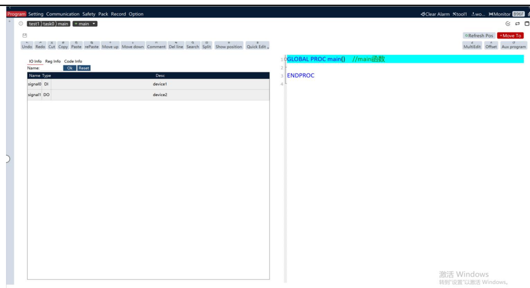
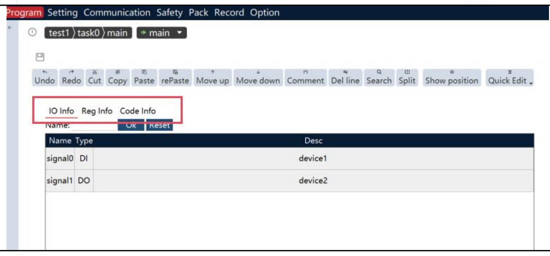


8.3.7Split screen function

8.3.7.1Overview

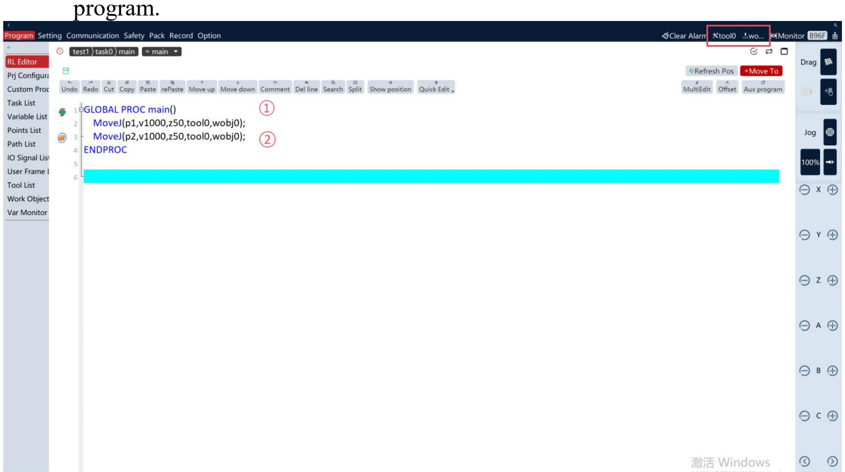
The function is primarily designed to facilitate quick access to I/O information, register information, and other function interfaces while users edit RL programs;

8.3.7.2Steps

Step	Graphical Representation	Explanation
1. Click the "Split screen" button in the top toolbar;		It can only be used in manual mode with the permission level of "Programmer" and above;
2. Display the left split-screen interface;		During split-screen mode, the right operation panel and left menu panel automatically close; When exiting split-screen, the two panels automatically expand;
3. Switch split-screen information display		Click the tab bar on the split-screen panel to switch between I/O information, Register information, and RL information views; The I/O list and Register list support name search functionality;

8.3.8Tool/work object pointer following function

When the robot is in Manual mode, the tool/work object group in the upper-right corner of the teach pendant will automatically switch to match the tool/work object assigned to the currently selected RL program line. Note: This function is not supported for custom tools/work objects defined in the RL program.

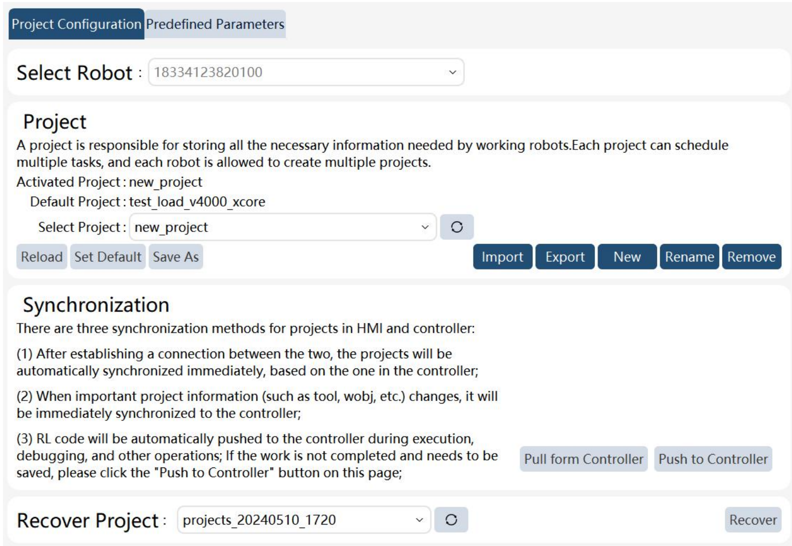


① When Line 2 is selected, the tool/work object in the upper-right corner automatically updates to tool0, wobj0

② When Line 3 is selected, the tool/work object in the upper-right corner automatically updates to g_tool_0, g_wobj_0

8.4Project configuration

The Project Configuration interface is used for the relevant configuration of the current project.



8.4.1Robot selection

When the RobotAssist software is running on the PC side and the robot is not connected, if you want to edit the RL project offline, you can select a previously connected robot here, and the system will switch and display the project data of the robot.
When the RobotAssist software is connected to a robot, the currently connected robot is shown here.

8.4.2Project

Reload: Reload the selected project.
Compared to previous versions, Version 3.1 has implemented adjustments to the RL programming language syntax. The main changes are as follows:

1. All functions and register commands must be enclosed in parentheses () when called;
2. Every expression must be terminated with a semicolon ;

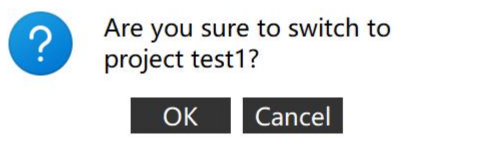
For example, in older versions, the program code may be written like this:

```
MoveL p1,v3000,z50,tool0
int a = 200
ConfL off
```

Whereas in Version 3.1, it shall be modified to the following format:

```
MoveL(p1,v3000,z50,tool0);
int a = 200;
ConfL(off);
```

Upon upgrading to Version 3.1, when you click "Reload" in the "Project Configuration" interface or click the "PPToMain" button in the "Program Editing" interface, the system will automatically detect and display a prompt to guide you through the project upgrade process.



Please note that once the upgrade is complete, if you need to revert to a previous version, you can export the current project in the 3.1 format. The exported ZIP file will include "ProjectName_old1.zip," which is a backup of your project before the upgrade. Be sure to keep this backup file properly stored for future needs.

Set as default: Set the selected project as the default project, which can be automatically loaded when the robot is started up.
Save as: Save as a selected project and push it to the controller.
Import: Open the Import Project page and select parameters such as the project system and project path to be imported.
Export: Open the Export Project page where you can select the project you wish to export and specify the export path. The system supports exporting projects in the following formats: xCorev2.2, xCore3.0, and xCore3.1 formats.
New: You can click it to create a new project. The project name can only be a collection of letters, numbers, and an underscore.
Rename: Rename the currently selected project.

Delete: Delete the currently selected project.

> ⚠ Warning
>
> Files cannot be recovered once deleted!

8.4.3Synchronization

There are three ways to synchronize the project files in HMI with those in the controller:

1. After the connection between the two is established, their project files will be automatically synchronized immediately, with the project files in the controller taking precedence;
2. When the important information of a project (such as tool and work object) changes, this change will be synchronized to the controller immediately;
3. The RL code will be automatically pushed to the controller when performing operations such as operation debugging. If the work is not completed and you need to save it, you can click the "Push to Controller" button on this page.

"Load from Controller" button: The teach pendant will re-synchronize and load the projects from the controller, and this operation will directly overwrite all current projects in the teach pendant.
"Push to Controller" button: The teach pendant will synchronize its current projects to the controller.

8.4.4Restore project

When the project is modified, the controller will back it up periodically. In the drop-down list, select a backup project named by time and click "Restore" to restore the backup project. Click to refresh the backup project list.

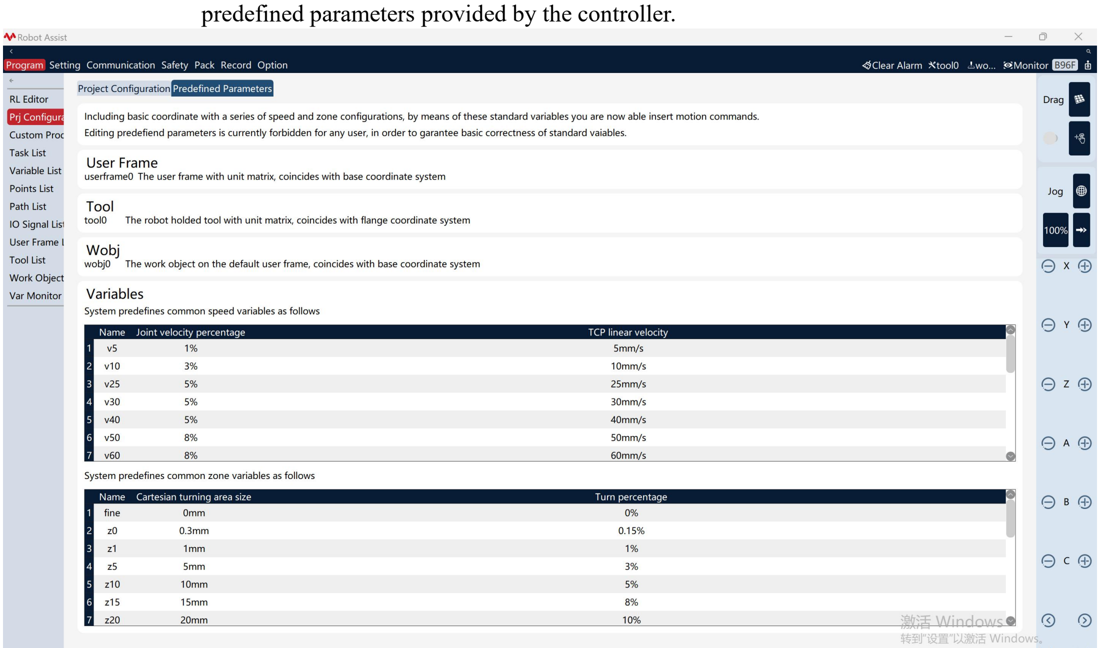
8.4.5Project upgrade

Normally, you do not need to manually use the "Project upgrade" function in the "Project Configuration" interface. However, in certain special scenarios where the upgrade cannot be performed by clicking "Reload" or the "PPToMain" button, you may manually trigger the upgrade process by clicking this function button.

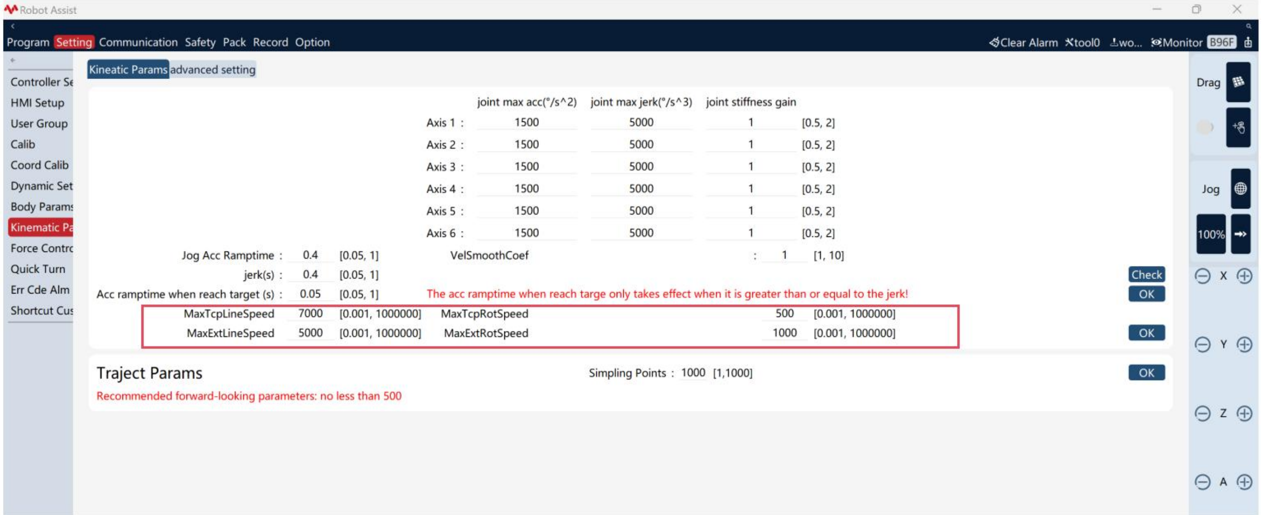
Please note that once the upgrade is complete, if you need to revert to a previous version, you can export the current project in the 3.1 format. The exported ZIP file will include "ProjectName_old1.zip," which is a backup of your project before the upgrade. Be sure to keep this backup file properly stored for future needs.

8.4.6Predefined parameters

The predefined parameters include the basic frame parameters, as well as a train of standard speed and turn zone parameters, and these variables are used as parameters for the RL command. You can view the physical meanings of the variables on this page, but they are not allowed to be edited currently.
In the Programming tab -> Project Configuration -> Predefined parameters page, you can view the predefined parameters provided by the controller.



In the Settings tab -> Motion Parameters -> Motion Parameters page, you will find the configuration options for the predefined parameter vmax, including the maximum TCP velocities and maximum external axis velocities. vmax is the only predefined parameter that can be configured by the user.

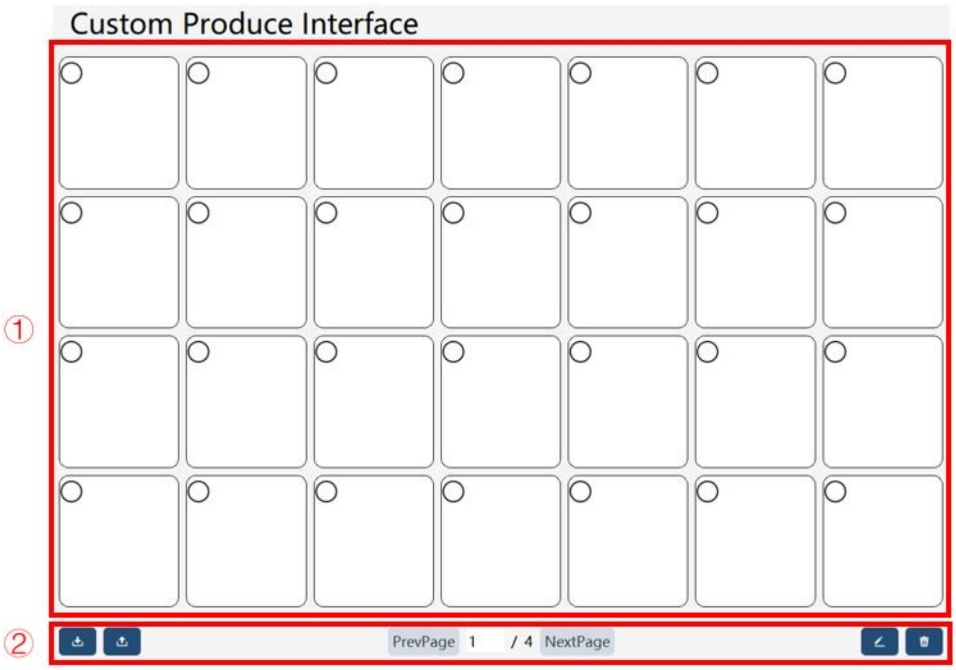


8.5Custom production

The custom production interface provides a simple and intuitive interactive mode for users. Users can create a concise monitoring and interaction interface simply by selecting registers, DI/DO signals, points, and project variables of interest.

The functions supported by this interface include: displaying and editing registers, displaying the status of DI signals, enabling DO signals, updating point positions, and displaying and editing project variables.

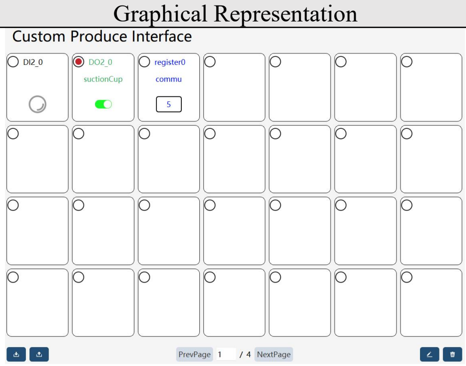
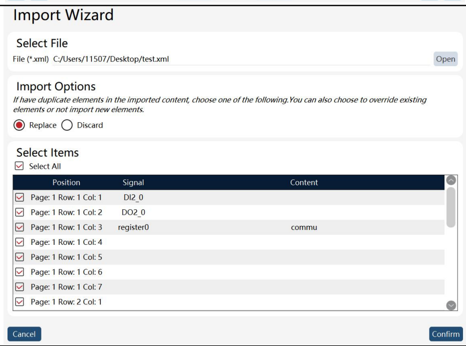
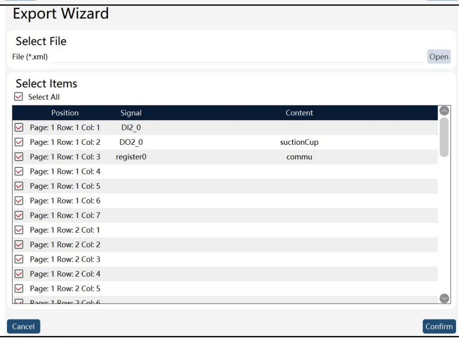
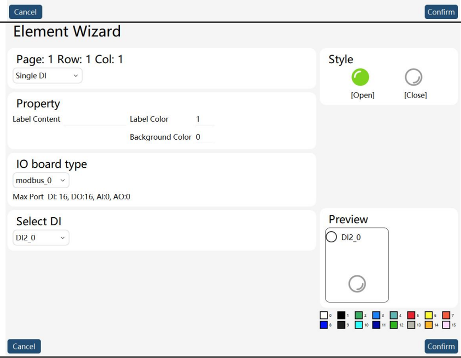
8.5.1Overview

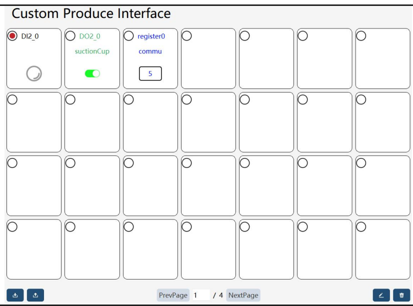


①	The Operation/Display Panel mainly includes the selection, display, and operation of different controls. The currently selectable controls include single DI, single DO, point position update, registers, and project variables.
②	Interface editing column. It involves data export from and import to the custom production interface, page switching, and individual control editing and deletion.

The Operation/Display Panel contains multiple production task controls, which can be imported, exported, edited, and deleted through the buttons below.
The control contains three parts, among which the top part shows the control signal name, the middle part refers to the control's label text (customized by users), and the bottom part is available for users' operations/displays on the control. Various controls provide different interactions.

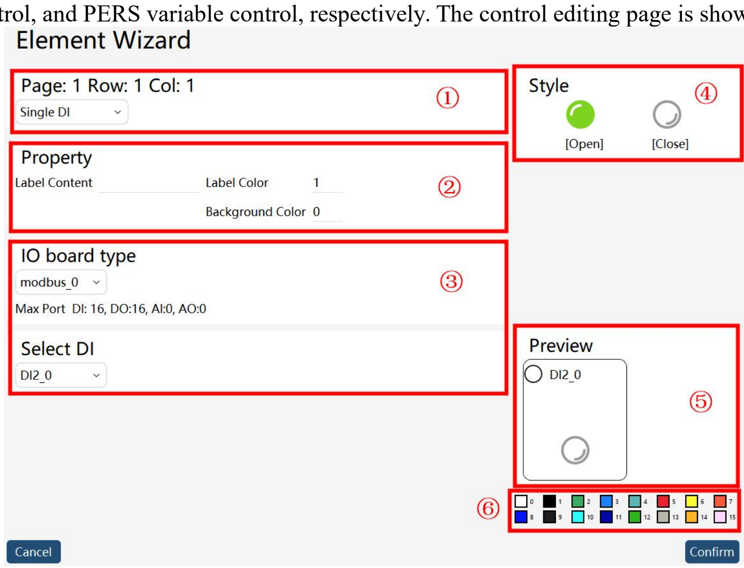
8.5.2Basic operations

Operation	Graphical Representation	Explanation
Control selection		Clicking the circle in the top left corner of a control allows you to select it on the interface, and the circular frame in the top left corner of the selected control will turn red. For instance, the currently selected control on the interface is the "suctionCup" control in the first row and second column.
Import		Click the  button to enter the import interface, and select the file import path, the import entries, and whether to replace.
Export		Click the  button to enter the export interface and select the controls that need to be exported and the export path.
Edit		Click the  button to enter the control editing page. Detailed usage instructions are provided below.

Delete		Click the button to delete the selected control.

8.5.3Control introduction

With four types of controls offered by the custom production interface, users can develop the interface style suited for the production conditions. They are the single DI control, single DO control, register control, and PERS variable control, respectively. The control editing page is shown below.

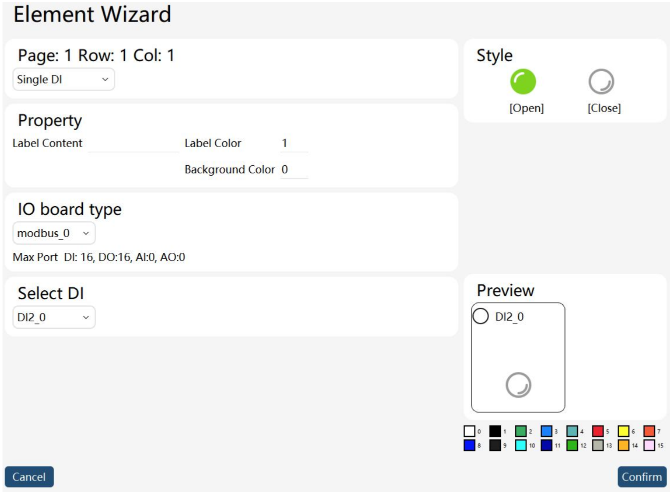


①	Select the control type, including: single DI, single DO, register, PERS variable, and none.
②	General attributes, including: label text, label color, and background color. See ⑥ for the representative label colors.
③	Special attributes, which are different for each control. See below for details.
④	Pattern bar, providing the data expression pattern shown by various controls. The pattern is not available for some controls, such as register controls.
⑤	Preview bar, where you can preview the final style effect of the current edited control.
⑥	16 colors are available, and you can enter the desired color number in ②.

8.5.3.1Single DI control

Single DI control can be used for DI display. Types of optional IO boards include: (1) IO devices configured by users on the IO Device interface; and (2) User DI signals configured by users in the project's IO signal list.

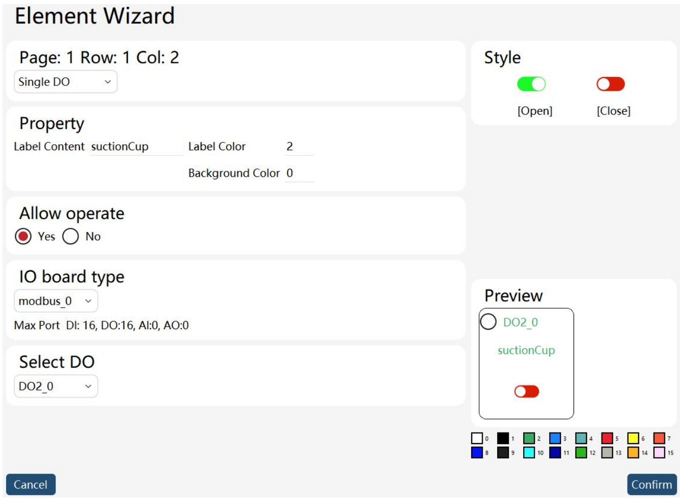
The editing interface is shown below:



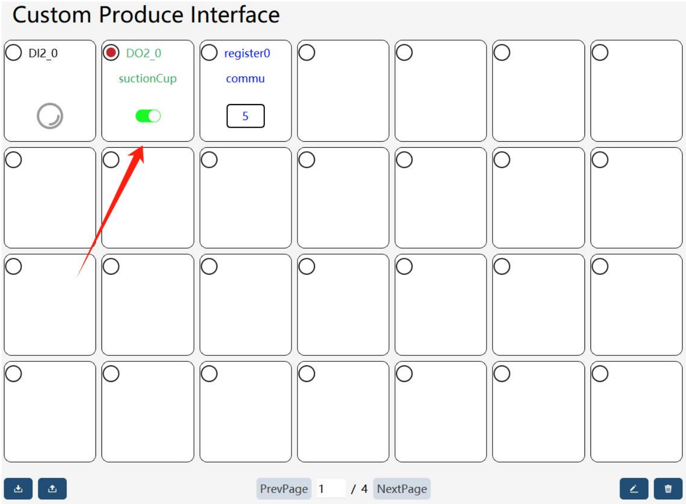
When the selected DI is True, the "On" icon under the control pattern is green. When the IO is False, the "Off" icon under the control pattern is gray.
The single DI control is not available for control interaction and is only used for DI status display.

8.5.3.2Single DO control

Single DO control can be used for DO signal display and settings. Types of optional IO boards include: (1) IO devices configured by users on the IO Device interface; and (2) User DO signals configured by users in the project's IO signal list.
The editing interface is shown below:

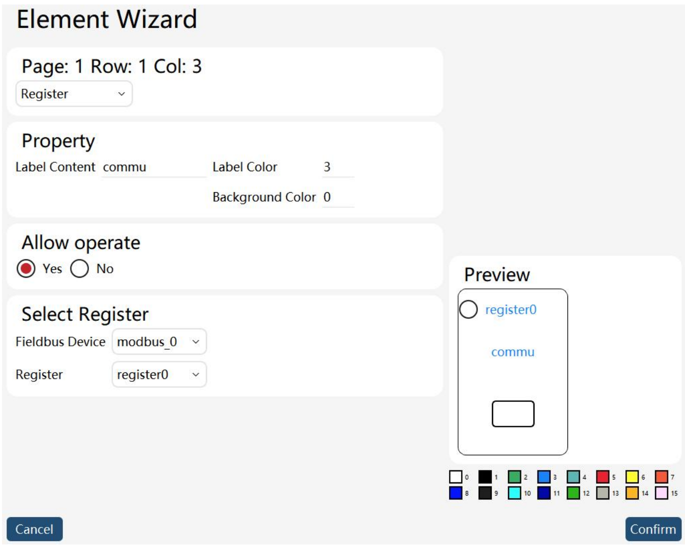


When the selected IO is True, the "On" icon under the control pattern is green. When the IO is False, the "Off" icon under the control pattern is red.
The single DO control provides button on/off interaction, and the "Allow Operation" attribute can be used to set whether a control can be operated on the "Operation/Display Panel".
Click the button within the red frame in the "Operation/Display Panel" interface to operate the DO signal.



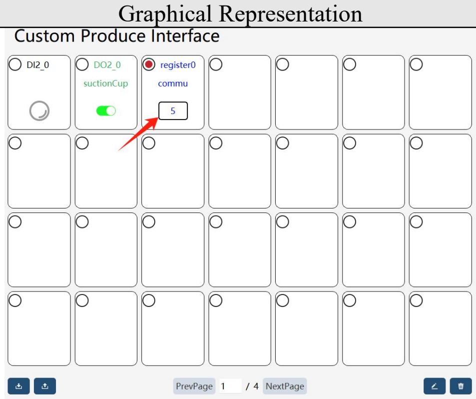
8.5.3.3Register control

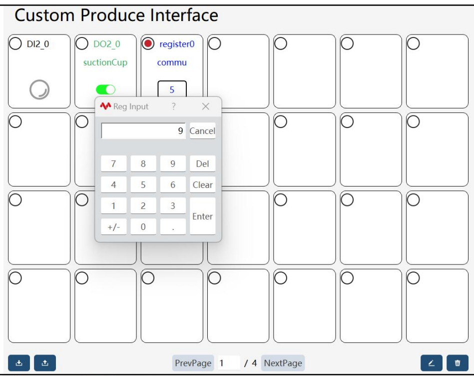
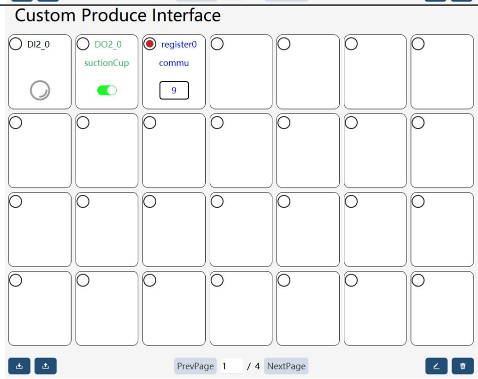
Register controls can monitor and modify the register values configured under "Communication" - "Registers". Registers can be filtered by "Bus device".
Attention: The register control does not support the register array type.
The editing interface is shown below:



The register control does not provide a status-style display, but it can display the register value.
The register control is edited by entering values using a numeric keyboard. On the "Operation/Display Panel" interface, you can click on the value "Display/Edit Box" for the value to enter the register value (attention: writing is only allowed when the register is set to "Write-only" mode).

Value writing steps:

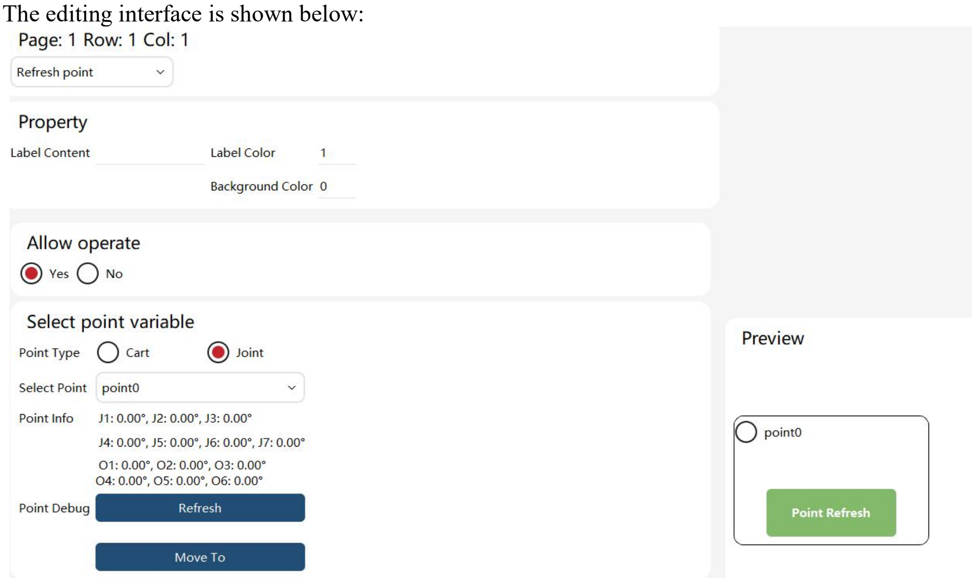
Step	Graphical Representation	Explanation
1. Selection		Click on the value display/editing box in the register control.

2. Input		Input the register value to be set.
3. Successful value writing.		

8.5.3.4Point position update control

The point position update control can modify point information in the point list, with selectable point types including Cartesian and joint.

The editing interface is shown below:

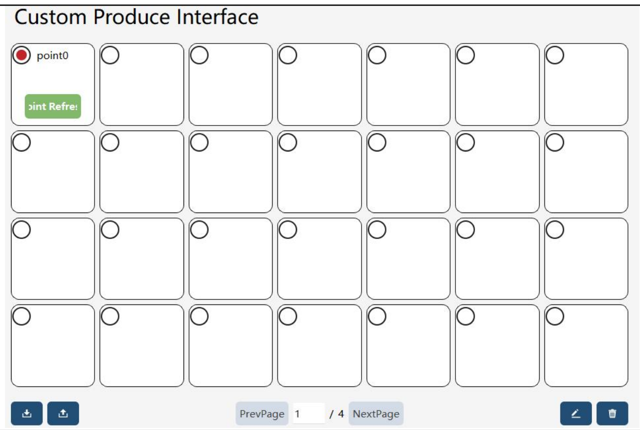
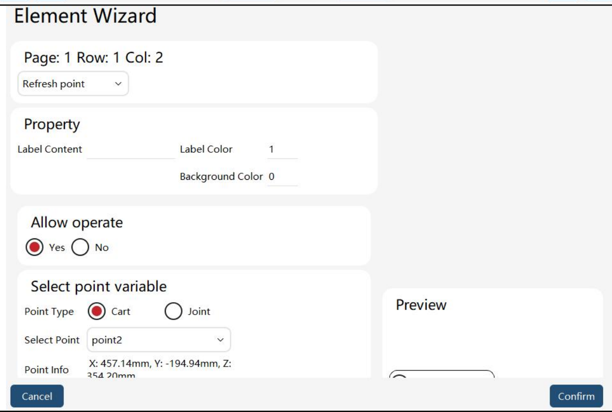
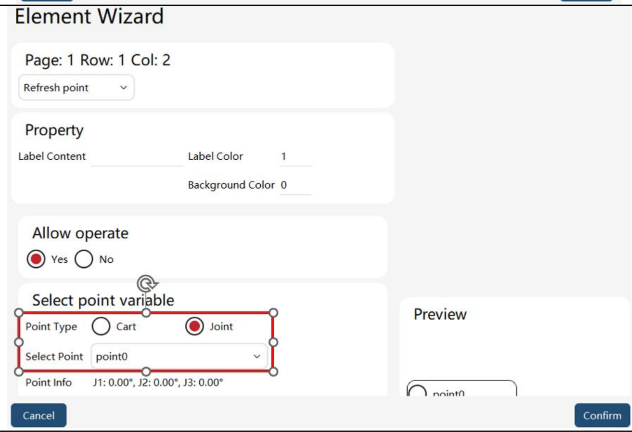
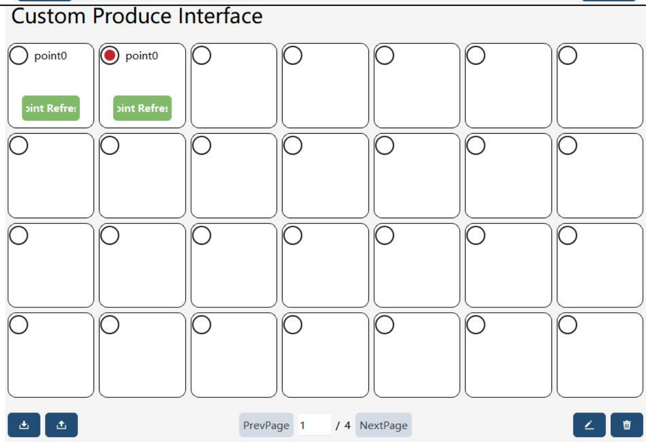


The point position update control does not provide status style display but offers an "Update Position" button.

Clicking the "Update Position" button on the "Operation/Display Panel" interface allows you to update the values of the selected point.

Steps for updating point position:

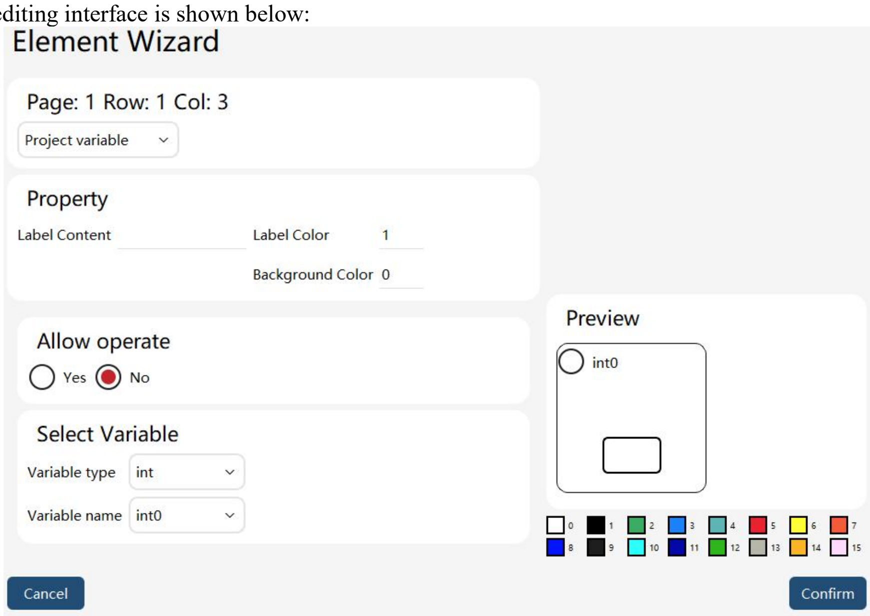
Step	Graphical Representation	Explanation

1. Selection		Select an empty control
2. Select the point position update function		
3. Select the point whose position needs to be updated		Support both Cartesian and joint point types
4. Complete		Click "Complete" to update the point position in the custom production interface

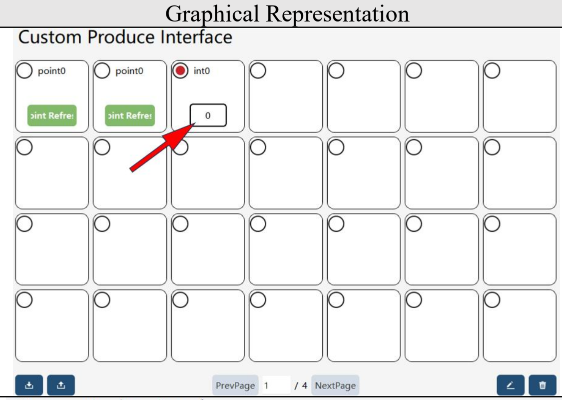
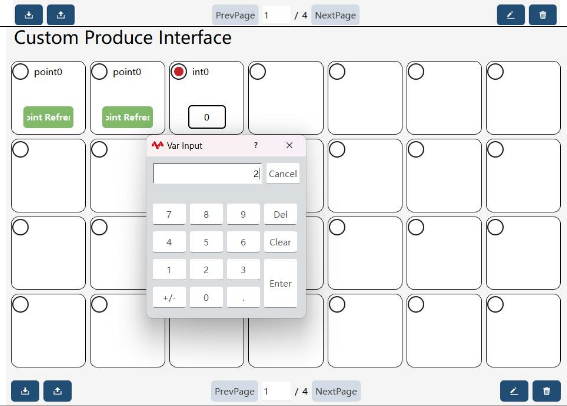
8.5.3.5Project variable control

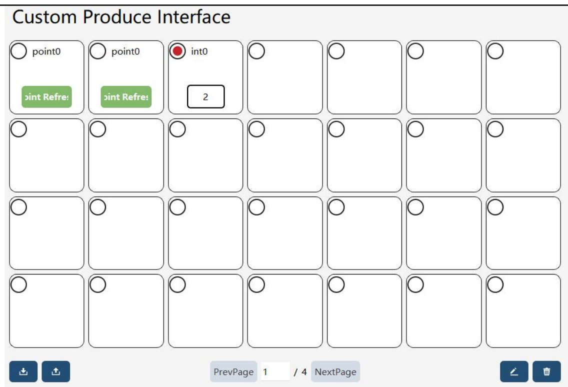
The project variable control can monitor and modify the variables configured under "Project" - "Variable List". The selectable variable types include int, byte, bool, and double. Variable array types

are not supported.
The editing interface is shown below:



The project variable control does not provide a status-style display, but it can display the project variable values.
The project variable control is edited by entering values using a numeric keyboard. The "Allow Operation" attribute can be used to set whether the control can be operated on the "Operation/Display Panel".
Click the value display/editing box on the "Operation/Display Panel" interface to write values to project variables.
Value writing steps:

Step	Graphical Representation	Explanation
1. Selection		Click the value display/editing box in the project variable control
2. Input		Input the project variable value to be set

3. Successful value writing.		

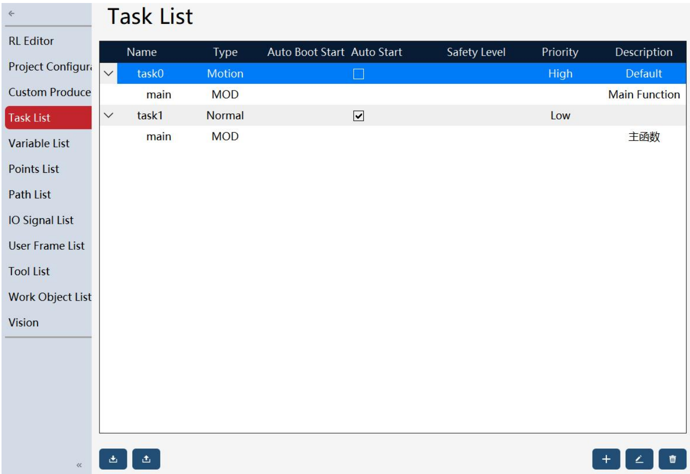
8.6Task list

The xCore control system supports multitasking.

Through multitasking, the "parallelism" of multiple robot programs can be realized. Typical application scenarios are shown below:

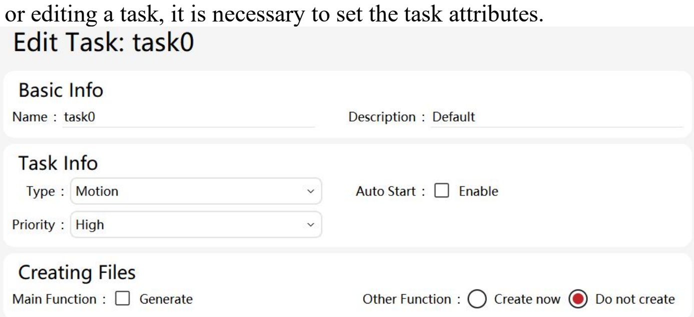
- Monitor continuously one certain signal even if the Main program stops operating (it is similar to the background PLC function, but its response speed is much lower);
- While the robot executes the main program of motion, it performs data reception, transmission, and other data processing with external devices, without being restricted by the execution logic of the main program;
- Receive some inputs through the teach pendant while working;

The "Task List" in the xCore system provides a management interface for parallel processing tasks. Users can view the attributes of existing tasks, create new tasks, edit tasks, and delete tasks on this interface.



8.6.1Task attributes

When creating or editing a task, it is necessary to set the task attributes.



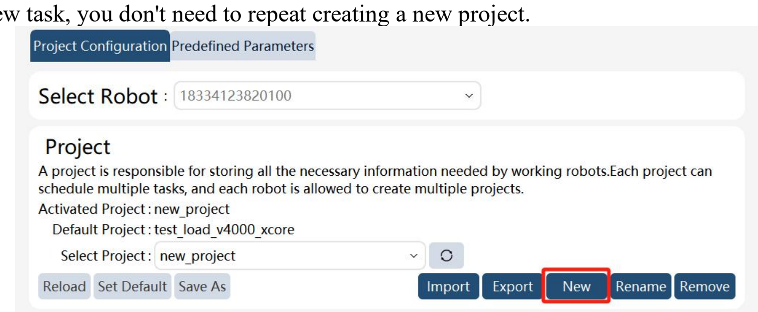
Task attributes	Description
Task name	The task name must be unique among all tasks, it should only be composed of alphanumeric characters and underscores. Its initial character shall not be a number and the maximum length of the task name is 20 characters.

Description	Describe the role of the task to assist users in understanding.
Task type	It includes routine tasks, motion tasks, semi-static tasks, visual tasks (used along with xVision). Among them, a motion task refers to the control of the robot's motion using RL commands. Only one motion task can run in a single project.
Autostart	It is used along with the Production mode. When selected, the program starts to re-execute when the system is restarted. It will not be stopped by the teach pendant or emergency stop under normal conditions.
Priority	Set the task priorities.
Create file	When the main function generation is checked, the main function will be generated automatically after task creation. The same applies to other functions.

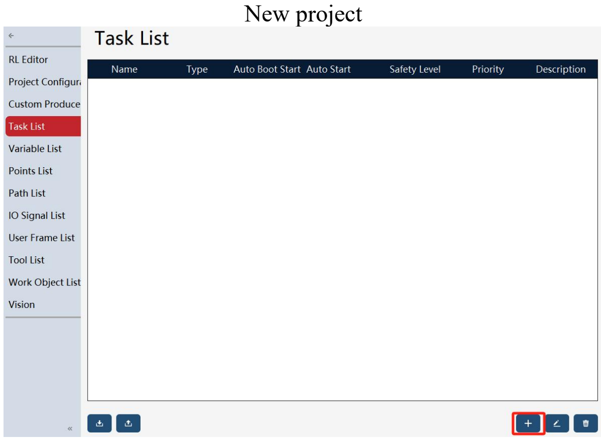
8.6.2Regular tasks and motion tasks

8.6.2.1New task

You need to create a new project before creating the first task. If you already have a project and want to add a new task, you don't need to repeat creating a new project.

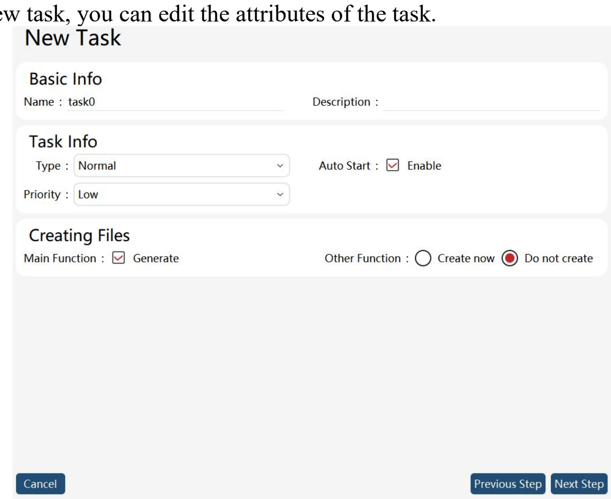


New project



New task

After creating a new task, you can edit the attributes of the task.



Task attribute editing

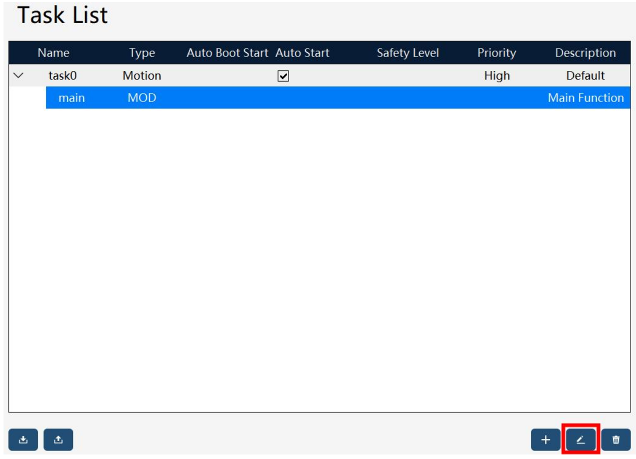
Note:

- 10 tasks are supported.
- A maximum of one motion task can run at a time.

- Changes in the task type, task entry function, and whether it is a motion task attribute take effect immediately.

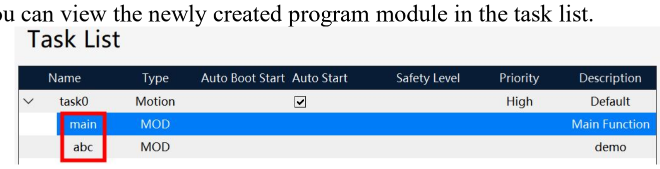
8.6.2.2New program module

Each task can include several program modules (mod files). As shown below, you can click the Edit button,

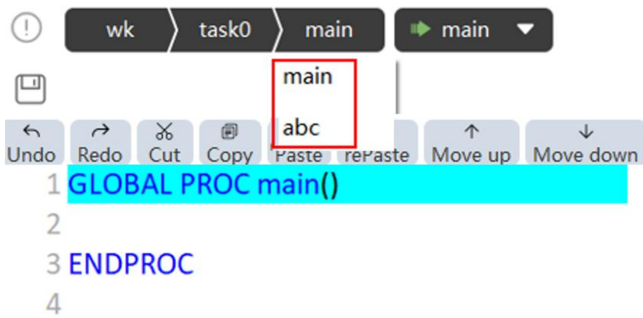


and click "+" on the new page to create a new program module. After entering the basic information such as the name and description of the module, you can click "OK" to complete the creation of the program module.

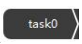
At this point, you can view the newly created program module in the task list.



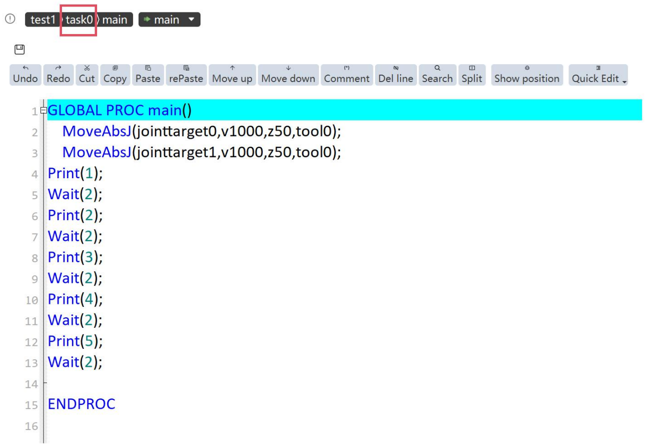
In the upper part of the RL editor, you can also select a new program module and program it.



8.6.2.3Starting and running

Click 

in the upper part of the RL editor to select a task. Use the Start/Stop button or external signals to control the start/stop of the selected task in the condition of manual enabling or automatic power-on.



Use restrictions:

- Generally, a background task will run cyclically. If a task does not contain any wait commands, the background task may consume too much computing resources, causing the controller to be unable to handle other tasks;
- The scopes of variable VARS and the constant CONST are limited to their respective tasks, but the GLOBAL-level PERS variable is a global variable;
- When PPToMain is executing, all non-running tasks execute PPToMain;
- When there are tasks running, it is forbidden to modify the contents in the Task List interface;

8.6.2.4Inter-task communication

The inter-task communication supports two methods: PERS variable and interruption.

Inter-task communication by PERS variable

- Global-level PERS variables with the same name shall be defined in all task projects that require communication, and the data type and dimension of variables shall be identical;
- PERS variables shall be used to control task execution and data transmission where necessary;
- All variables and tasks in the variable list and point list are available at will;

Use restrictions:

- You just need to specify an initial value for the PERS variable in one of the tasks. If you have specified an initial value for the same PERS variable in multiple tasks, the initial value defined in the first running task will be used.
- When a task waits for another task by means of the PERS variable and the WaitUntil or WHILE command, it is necessary to pay attention to coordinating with the wait command (greater than 0.1s) to avoid the program quickly executing the empty judgment command, and thus occupying too much system resources.

8.6.3Semi-static task

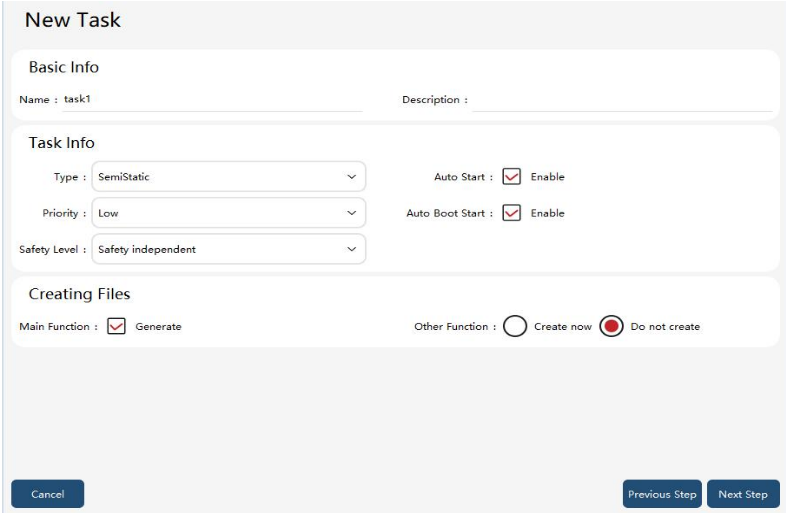
Since v2.0.1, the xCore control system supports performing semi-static tasks.
The semi-static task belongs to the multi-task function and runs a program written in RL language. Compared with regular and motion tasks, semi-static tasks have two features as follows: (1) After being properly configured, they can self-start after being switched on without any commands such as power-on and start-up commands; (2) The pause button is not effective on the semi-static task. With these two features, the operation cycle of a semi-static task covers almost the entire time from power-on to power-off of the control system.

Typical application of semi-static takes:

- Judge the robot position periodically, and notify the host computer via registers, IOs, etc.;
- Output custom heartbeat signal;
- Transmit data among multiple devices;

8.6.3.1Semi-static task creation

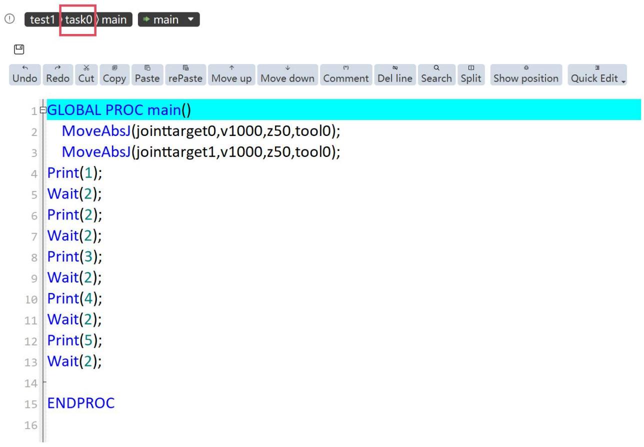
In the task list, click "+" to create a new task. Select Semi-static Task in the Type, complete other attribute settings, and click the "Next" button to create the semi-static task.



8.6.3.2Starting and stopping of semi-static tasks

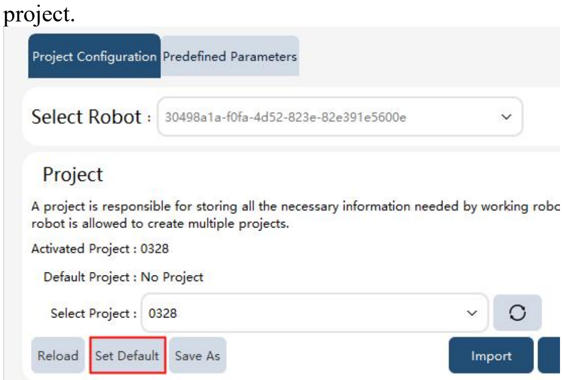
Just like the regular task startup, after creating and configuring a semi-static program, the semi-static task can be started by powering on in manual or automatic mode and clicking the “Start” button. If there is a semi-static task running in the project, there will be a prompt  in the middle of the bottom status bar. At this time, clicking pause or power off will not stop the semi-static task. Clicking the "semi-static status" button will pop up a confirmation dialog box 

, and clicking "OK" will stop the semi-static task. When the semi-static task does not trigger a program exception, only this button can stop the semi-static task (the semi-static task can be paused only after the regular task is paused).



8.6.3.3Configuring semi-static task for self-start

In the project configuration interface, set the project associated with the semi-static task as the default project.



In the task list, check the two options, running on startup or not and running or not.

Task List

Name	Type	Auto Boot Start	Auto Start	Safety Level
task0	Motion		☑	
task1	Semistatic	☑	☑	Safety independent

Finally, click pptomain to synchronize the project configuration from the HMI to the controller, allowing the semi-static task to self-start.

8.6.3.4Safety level of semi-static tasks

In addition to configurations such as priority and "running or not" of regular tasks, the semi-static task has an additional feature: safety level.
It is used to define the controller's exception handling policy for the semi-static task as the semi-static task operates abnormally (e.g. data reception failed).

- Safety irrelevant: It applies to the host computer or operators who do not care about the operating condition of semi-static tasks. A semi-static task configured with this safety level will be stopped individually if its operation is faulty, with the operation of other tasks not affected.
- System stop: Applicable when the data of semi-static tasks affect the program's safety logic (e.g. deciding the motion point via a semi-static task, and informing the host computer how to control the robot via a semi-static task). The semi-static tasks of the System stop level will make all tasks paused if there is any fault in their operation.

8.6.3.5Recommendations for semi-static task debugging

- The controller supports single-step debugging of semi-static tasks in manual power-on mode. However, it is still recommended to complete debugging with regular tasks first, and then change the task type to semi-static task to avoid unexpected startups of unfinished semi-static tasks during reboots in the debugging process;
- For semi-static tasks, it is recommended to add error handling for commands that may fail. For example, the ReadXX command may time out or fail due to network fluctuations, external device issues, or other unexpected factors. To ensure that semi-static tasks continue execution with minimal interruption, it is advisable to use a try_catch statement to protect the code that may fail, and perform appropriate error handling in the catch statement (e.g., use goto to jump back to before the Read command and retry the data reading);
- Due to the specialty of long-term execution inherent to semi-static tasks, the single-run and loop-run settings in the upper-right corner of the debugging interface do not apply to semi-static tasks. All semi-static tasks run in a continuous loop by default.

8.6.4Task monitoring

The current running status of each task can be monitored in the status monitoring.

8.7List of variables

8.7.1Variable naming rules

Variable names in the RL language can consist of letters, underscores, and numbers, but must start with a letter or underscore "_". However, variable names cannot be the same as system keywords. You can see Keywords pre-definition for RL system keywords.
In addition, there are precautions as follows:
In the same module, GLOBAL and LOCAL level variables with duplicated names are not allowed;
In different modules, GLOBAL variables with duplicated names are not allowed;
In different modules, LOCAL variables with duplicated names are allowed;
In the same module, no variables (GLOBAL, LOCAL, excluding ROUTINE) are allowed to have naming conflicts with functions in this module;
In different modules, no naming conflicts of GLOBAL level functions and variables are allowed;

> Note
>
> When a variable name contains two characters only, it is important to note that the second character shall not be "h" or "b", otherwise, the variable may be converted to hexadecimal or binary. For more information, please refer to the Number system conversion.

8.7.2Variable scope

The RL language system defines three scopes:

GLOBAL	It is visible to any program module of the current task, and can be declared in the module declaration area; If variables need to be accessed across tasks, they shall be declared using the GLOBAL PERS keyword. Attention: The variables in the variable list and point list are all global, and are readable and writable by all tasks.
LOCAL	It is only visible to the current program module, and can be declared in the module declaration area;
ROUTINE	It is only visible within the current function and can only be declared within the function body, and the scope type (GLOBAL or LOCAL) is not allowed to be specified when the scope variable is declared; Attention: Scope only applies to variables, not custom functions.

8.7.3Storage type

Each variable can be divided into three kinds: VAR (Variable), PERS (Persistent Variable), and CONST (Const Variable), depending on whether it can be modified during program execution.

VAR（Variable）	Variable, can be re-assigned in the process of program execution;
CONST（Const Variable）	Constant variable, cannot be re-assigned during program operation, and this type must be determined at the beginning;
PERS（Persisten Variable）	Continuous variable. During program execution, if the value of the variable type changes, the variable will be automatically modified from the initial value to the current value, thus achieving the effect of "Persistent" storage; Note: ● Even if the value of a PERS type variable is changed while the program is running, the initial value displayed in the program editor declaration area is not immediately refreshed, and the initial value displayed in the program editor declaration area is updated to the latest value only when the program reloads or stops; ● Regardless of whether the program is running or not, only the initial value of the PERS variable can be viewed in the "Variable Management" interface, and its current value can be viewed through status monitoring or the print command.

8.7.4Keywords pre-definition

The following are reserved keywords (case insensitive) that are predefined for the RL language:
Module、EndModule、Proc、EndProc、Func、EndFunc、SetDO、DO_ALL、SetGO、SetAO、WaitDI、Wait、WaitUntil、WaitWObj、WBID、Q、P、J、V、W、T、S、L、CA、DURA、IGNORELEFT、EJ、1J、FCBV、FCCV、FCOL、FCXYZ、FCCART、PE、PER、TCP、ORI、EXJ、CFG、PDIS、JDIS、MoveAbsJ、MoveJ、MoveL、MoveC、MoveT、LOCAL、TASK、GLOBAL、VAR、CONST、PERS、INV、DOT、CROSS、sin、cos、tan、asin、cot、acos、atan、atan2、sinh、cosh、tanh、ln、log10、pow、exp、sqrt、ceil、floor、abs、rand、GetCurPos、Print、PrintToFile、ClkRead、TestAndSet、IF 、Else、Endif、WHILE、ENDWHILE、for、from、to、endfor、Break、Continue、Del、Int、Double、Bool、String、BYTE、Robtarget、Speed、Zone、Tool、Wobj、Jointtarget、TriggData、Load、FCBoxVol、FCSphereVol、FCCylinderVol、FCXyzNum、FCCartNum、Pose、CLOCK、INTNUM、SYNCIDENT、TASKS、Call、Return、EXIT、Pause、StopMove、StartMove、StorePath、RestoPath、True、False、Interrupt、When、Offs、CalcJointT、CalcRobT、CRobT、RelTool、SocketCreate、SocketClose、SocketSendByte、SocketSendInt 、 SocketSendString 、 SocketReadString 、 SocketReadBit 、 SocketReadInt 、SocketReadDouble、AccSet、MotionSup、TriggIO、TriggJ、TriggL、TriggC、On、Off、clock、intnum、userframe、pinf、ninf、FCFRAME_WORLD、FCFRAME_TOOL、FCFRAME_WOBJ、FCFRAME_PATH、FCPLANE_XY、FCPLANE_XZ、FCPLANE_YZ、FC_LINE_X、FC_LINE_Y、FC_LINE_Z、FC_ROT_X、FC_ROT_Y、FC_ROT_Z、Offs、CalcJoinT、CalcRobT、CRobT、RelTool、Start、Time、ClkReset、ClkStart、ClkStop、CONNECT、WITH、IDisable、IEnable、ISignalDI、Single、SingleSafe、WaitWobj、DropWobj、WobjIdentifier、WobjAngle、ActUnit、DeactUnit、INTNO、Exp、DoubleToStr、WaitSyncTask、FCAct、FCDeact、FCLoadID、FCCalib、FCSupvForce 、 FCSupvTorque 、 FCSupvPosBox 、 FCSupvPosSphere 、 FCSupvPosCylinder 、FCSupvOrient 、 FCSupvOrient 、 FCSupvReoriSpeed 、 FCSupvTCPSpeed 、 FCCondForce 、FCCondTorque、FCCondOrient、FCCondReoriSpeed、FCCondPosBox、FCCondPosCylinder、FCCondPosSphere、FCCondTCPSpeed、FCCondWaitWhile、FCRefLine、FCRefRot、FCRefSpiral、

FCRefCircle、FCRefForce、FCRefTorque、FCRefStart、FCRefStop、FCSetSDPara

8.7.5Number system conversion

The RL language supports direct entry of hexadecimal, binary, or values of scientific notation by adding a number system identifier after a number or letter.
Example 1
After the "h" suffix is added after 0−9, a−f, or A−F, the RL compiler treats the corresponding number or letter as hexadecimal and converts it to decimal in the compiler. For example:
8h stands for 8 in hexadecimal and 8 in decimal;
bh stands for b in hexadecimal and 11 in decimal;
25h stands for 25 in hexadecimal and 37 in decimal;

Example 2
After the "b" suffix is added after 0−9, a−f, and A−F, the RL compiler treats the corresponding number or letter as binary. For example:
1b stands for 1 in binary and 1 in decimal;
10b stands for 10 in binary and 2 in decimal;
1010b stands for 1010 in binary and 10 in decimal;

Example 3
Adding the "e±x" after a number indicates that the number is multiplied by 10 to the x power. For example:
5e+20 represents 5×10^20;
26e−15 represents 26×10^(−15);
112e−10 represents 112×10^(−10);

8.7.6Variable declaration

A declaration must be made before using a variable. The format of the variable declaration command is as follows:
SCOPE STORAGE TYPE varname [= value]
Where:
1. SCOPE refers to variable scope. Please refer to Variable Scope;
2. STORAGE refers to variable storage type. Please refer to Storage Types;
3. TYPE refers to variable type, and can be a basic type or a special type. Please refer to Variable Type;
4. varname is the variable name. Please refer to Variable Naming Rules;

The content in square bracket [ ] is optional and can be either initialized or not when variables are declared. For variables that are not explicitly initialized when they are declared, the system automatically assigns different initial values as per the type of the variables. The default initial value may cause program execution problems in some cases, so it is recommended to initialize each manually added variable.

Example
There are a few examples of variable declarations as follows:
Example 1
VAR int counter = 8; //Declare the integer variable count and assign an initial value of 8
VAR double time = 2.5; //Declare floating-point variable time and assign an initial value of 2.5
VAR bool ifOpen = true; //Declare the variable bool type ifOpen and assign the initial value of true

Example 2
In general, no duplicate names are allowed for variables:
VAR int counter = 8;
VAR double counter = 2.5;
The compiler will report an error message at this time by prompting "Failed to add variable".

Example 3
However, a global variable and a local variable can have the same variable name:
VAR int counter = 1;
GLOBAL int counter = 555;
Although variables with different scopes allow duplicate names, it is not recommended to use variables with duplicate names in order to avoid confusion and misuse, unless the variables with duplicate names have special technological advantages.

Note
Variables cannot be declared inside loop statement blocks such as while, otherwise, duplicate

declarations will be caused when this part of the code is repeatedly executed, resulting in a "Fail to add variable" error. Please declare the variables outside the loop body.

Use restrictions
The ROUTINE variable that declares the PERS storage type is not supported;
When there is a duplicate name for variables or functions of different levels, the compiler will decide which variable to be used based on the priority of the scope. Variables with the highest priority order will be selected first, and those with lower priority order will be obscured and hidden. The priority of scopes is as follows:

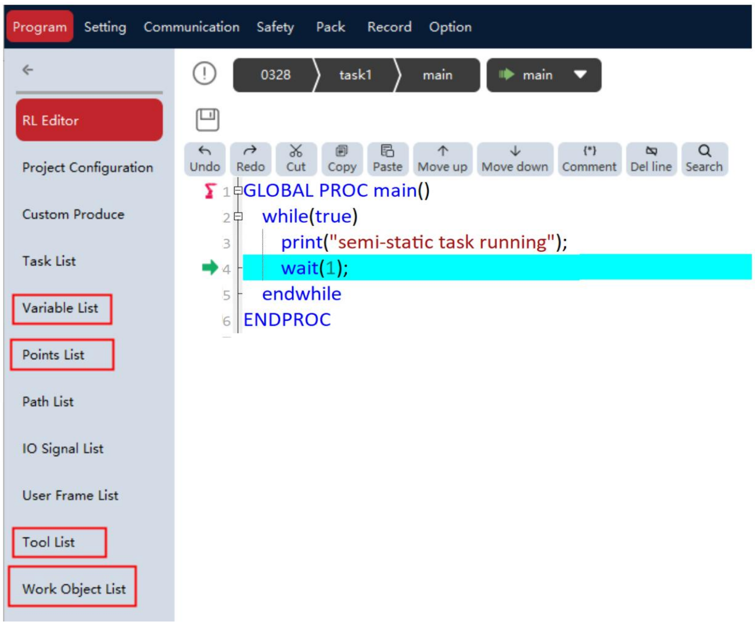
- When the variable names are duplicated, the priority of scopes is as follows: ROUTINE> LOCAL> GLOBAL;
- When the function names are duplicated, the priority of scopes is as follows: LOCAL > GLOBAL;

8.7.7User variable hold

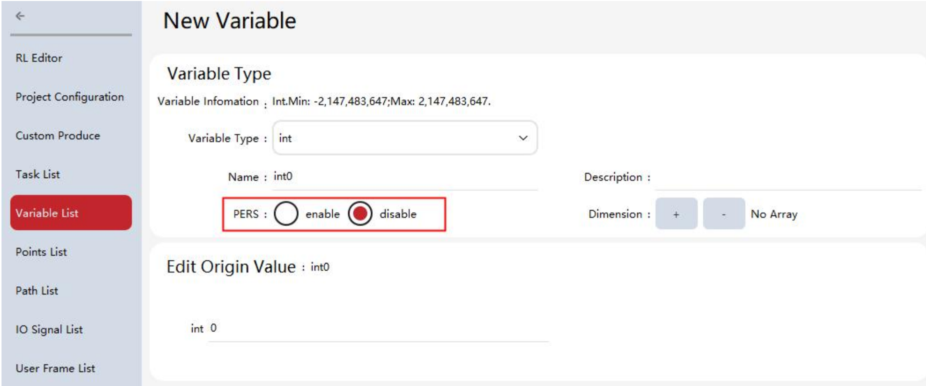
The user variable "a" with hold is created in an RL project. This user variable is marked as a PERS variable, then the value of this variable is held on the non-volatile storage media when RL stops, the robot restarts, shuts down, or is powered off. When the robot is powered on again or RL is running again, the value of variable a is restored to the value held. The initial value is assigned only when the variable is created for the first time or re-edited. (Attention: Only PERS variables added to the variable list possess the hold attribute. However, PERS variables defined in the variable declaration area have no hold attribute.)

The persistence is supported for the following user variable types: Int, byte, double, bool, string, pose, speed, zone, fcboxvol, Fcspherevol, fccylindervol, fcxyznum, fccartnum, torqueinfo, socketserver, socketconn, and serials.

User variable hold configuration
On the RL project interface, the entries where persistent variables can be created as shown in the red boxes below:



Click the variable list, point list, tool list, or work object list to create user variables of corresponding type. All variables for which the persistent attribute can be created have a "persistent" attribute item, where "yes" indicates that the variable is a persistent variable, marked as a PERS variable. For example, create a PERS variable of int type, whose configuration is as follows: (Other types can be configured by analogy)



Modification of PERS variable

The PERS variables of the xCore control system are stored in the form of initial value+hold value. The initial value refers to the data input by users to the variable list, and the hold value is the data, after it is modified by the program, stored on the non-volatile storage medium. The hold values of base types (int\bool\double) can be observed by status monitoring, and structure data (e.g., points, tools, work objects) can be printed by the print command.

During the operation of RL program, the PERS variable can be modified by the operator "=", and the modified data will be stored as a hold value within the controller. For the next time of program operation, the hold value of corresponding variable is preferred to be read, and if there is no hold value, the initial value of the variable is read.

If the PERS variable is modified by the point update button or the editing function of the variable list, the initial value and hold value are modified at the same time.

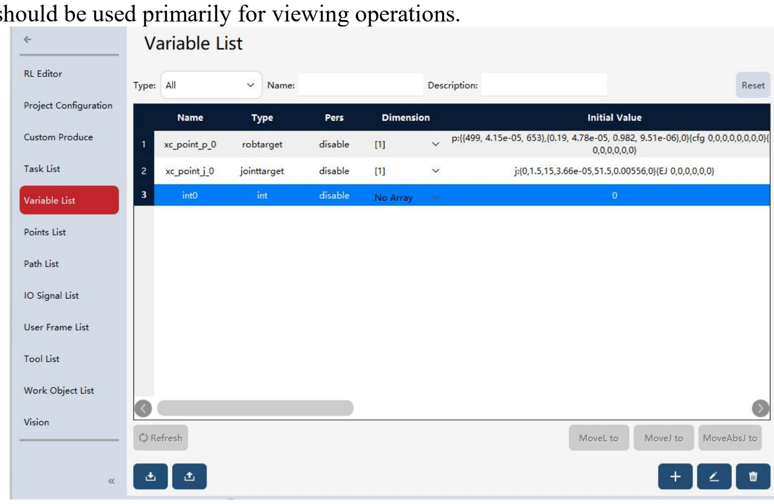
8.7.8Variable list operation

The variable list interface allows the creating, viewing, modification, and deletion of almost all variables in the robot system. The currently supported variable types include:

S/N	Variable Type	Description
1	System predefined variables	Variables that cannot be modified by users and are used to store certain system parameters, such as tool0/wobj0.
2	User predefined variables	Variables that can be modified by users and used in multiple programs, such as user-calibrated tools and work objects.
3	Program variables	Variables defined by the user in the program, which are generally used only in the current program and its subprograms. Program variables include most of the variable types supported by the system.

8.7.8.1Variable viewing

For some types of variables that have specifically defined steps, such as: speed/zone (defined and modified on the auxiliary programming interface). Although such variables can be viewed and modified in the Variable View interface, it is still recommended to use the dedicated interface for modification for the sake of convenient operation and fewer errors. The variable management interface should be used primarily for viewing operations.



> Note
>
> The variables that can be viewed and modified in the variable list interface are limited to those used in the currently loaded robot program. Therefore, the displayed variables will change when other programs are loaded.

8.7.8.2Variable editing

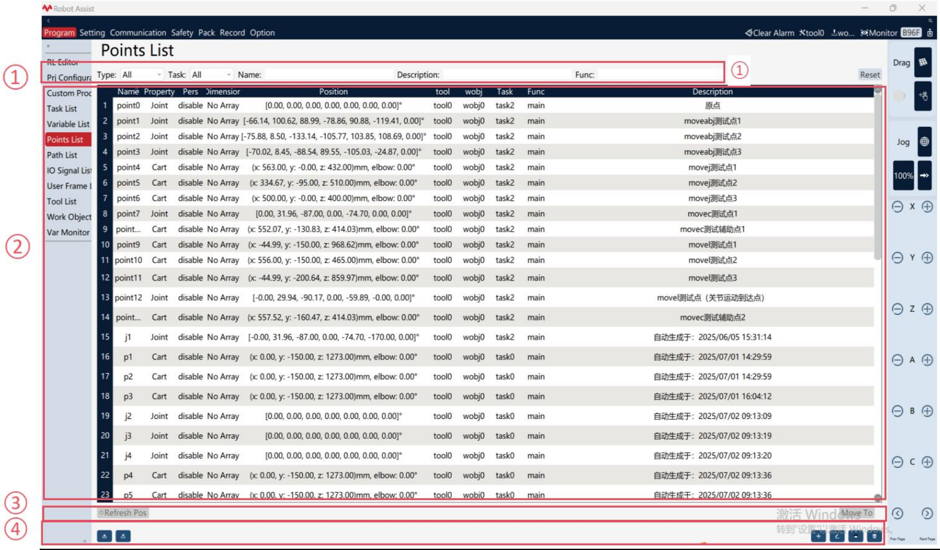
If you need to add variables or modify certain existing variables, you can click the "New" or "Modify" button to enter the variable editing page for relevant operation.

Type	Used to select variable types when creating new variables. All supported types are listed in the sidebar on the left.
Name	Enter the variable name.
Dimension	To create or modify arrays, supporting up to 3D arrays.
Persistent	Define as a persistent (pers) variable or non-persistent variable
Value	Display the variable value
Description	Provide a description of the variable

8.8Point list

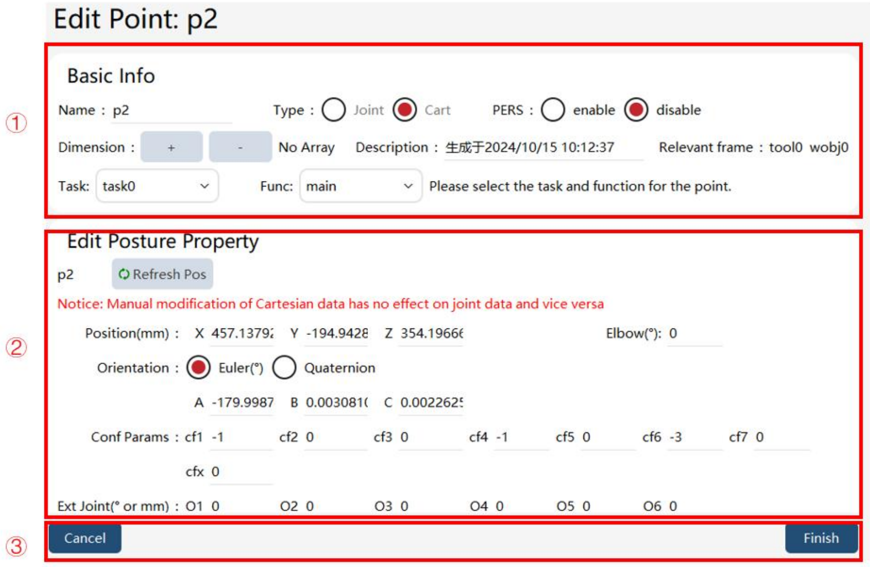
8.8.1Overview

The point list is used to manage the robot points involved in projects in a unified manner. The points used in the RL program need to be configured in the point list before they can be used in the program.



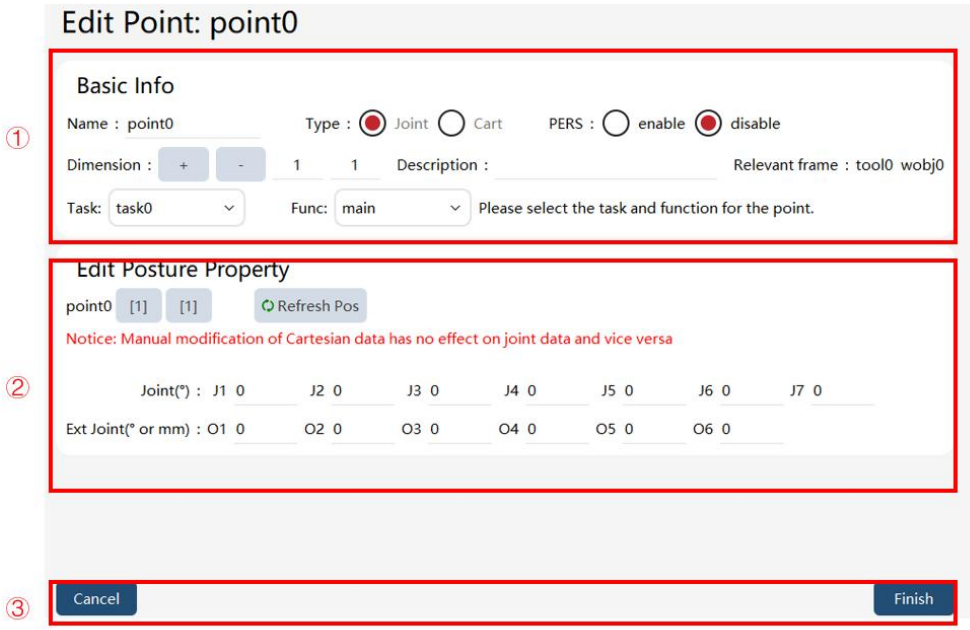
①	Point filter: Allow you to filter which points are displayed in the list below. You can filter points by type, task, name, description, and function.
②	Point display: Show the attributes of each point, including name, attribute, persistence, position, tool frame, work object frame, task, function, etc.
③	Point editing, use the "Update Position" button to update the point coordinates to the current point position of the robot; The button on the right is used to move the robot to this point position: Clicking the "Move to" button will open the motion window. For Cartesian-type points, you can click the MoveL or MoveJ button: The robot will move to this point position in a straight line or through joint space motion; and for joint-type points, you can click the MoveAbsJ button: The robot will move to this point position through joint motion.
④	Function button area: Import Point, Export Point, Create Point, Edit Point, Delete Point, and Multi-select Points

Point editing page:
Cartesian space point:



①	Basic information, including point name, description, persistence, dimensions, task, and function.
②	Pose information. The point type is not modifiable. Clicking the "Update Position" button will update the point to the current pose, or you can change it manually. Manual modifications to Cartesian data will not affect joint data, and vice versa. Conf parameters: These parameters can be modified for the corresponding point position.
③	Clicking "Cancel" returns to the point list interface without saving any changes; clicking the "Complete" button saves the current changes and returns to the point list.

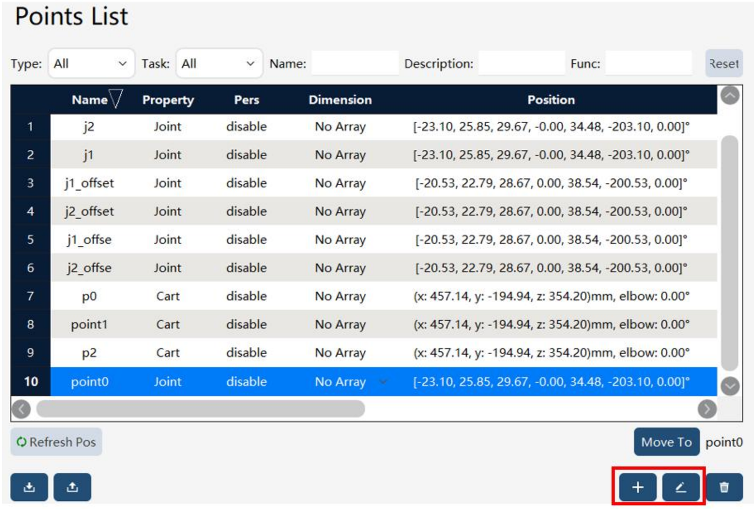
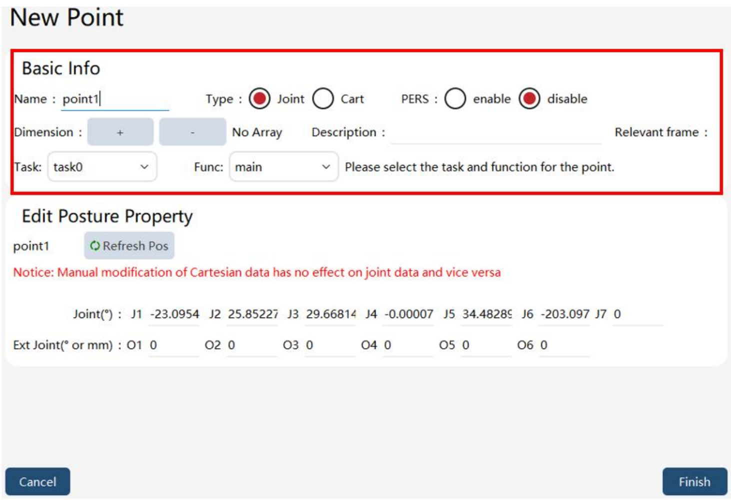
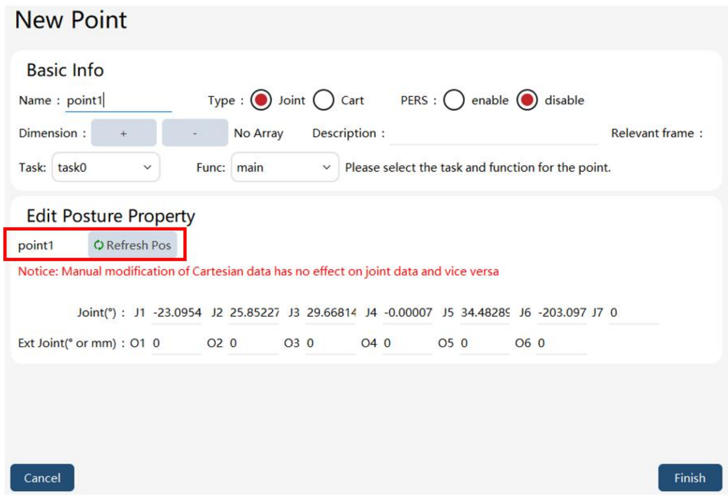
Joint space point:

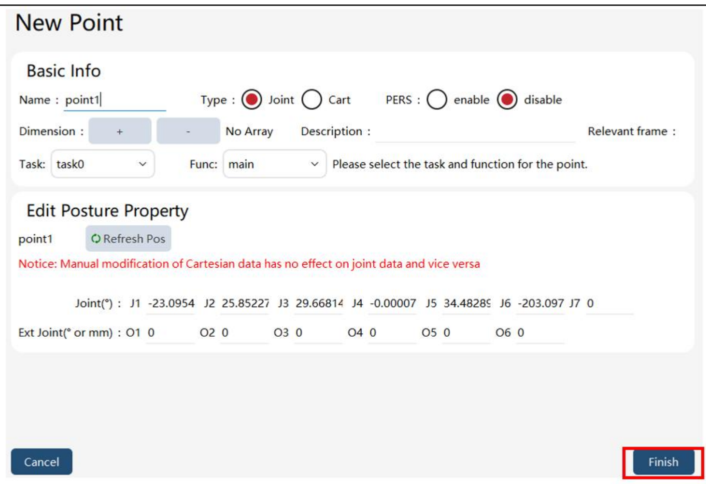


①	Basic information, including point name, description, persistence, dimensions, associated frames, task, and function; clicking the dimension add or delete buttons can modify the point's dimensions.
②	In the pose information, when the array is multi-dimensional, different dimensions can be selected to modify their contents. Clicking the "Update Position" button will update the point to the pose of the currently selected dimension, or you can change it manually. For points of Cartesian type, manually modifying the data does not change the joint data, and vice versa.
③	Clicking "Cancel" returns to the point list interface without saving any changes; clicking the "Complete" button saves the current changes.

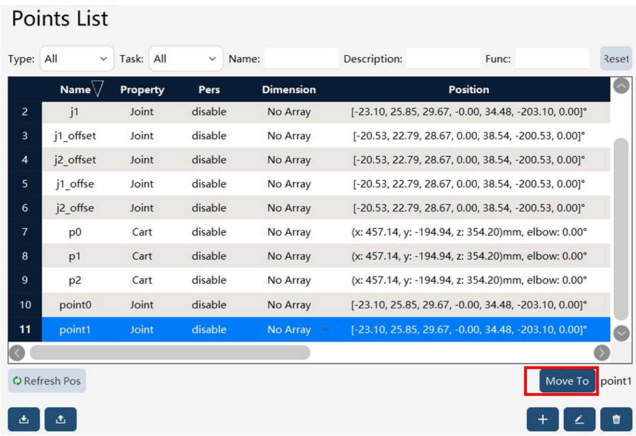
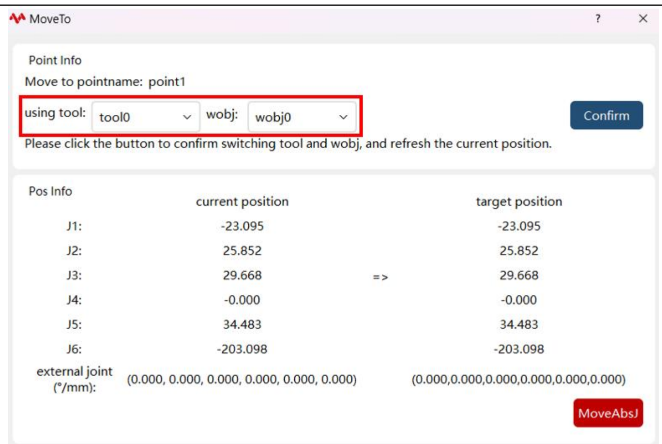
8.8.2Operation examples

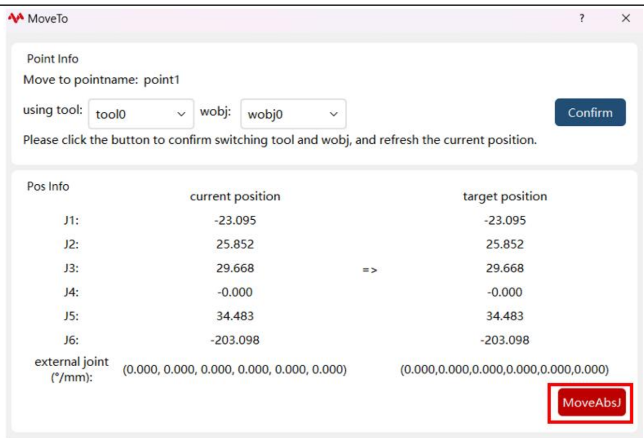
8.8.2.1Point creation/editing

Operation	Graphical Representation	Explanation
1. Creating/editing		Click the or button to enter the point editing page
2. Basic information editing		Point type can only be selected when creating a new point; When editing a point, you can only select the name, point description (optional), persistence, dimensions (optional), task, function, etc.; Users can set the point's dimensions using the dimension add and delete buttons.
3. Pose information editing		Select the type of point and obtain the current pose of the robot through "Update Position". If the point is a new point, the current pose will be obtained by default.
4. Complete		Click the "Complete" button to save the current information.

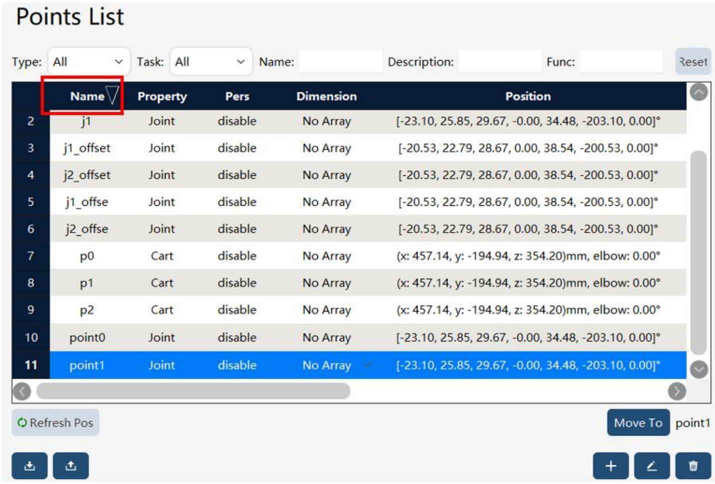
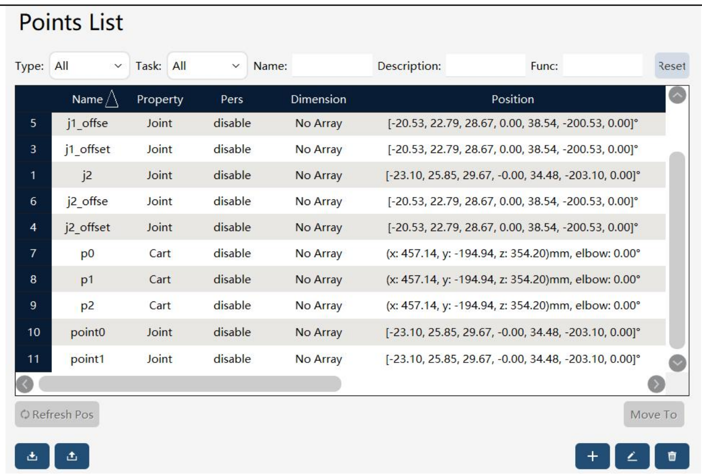
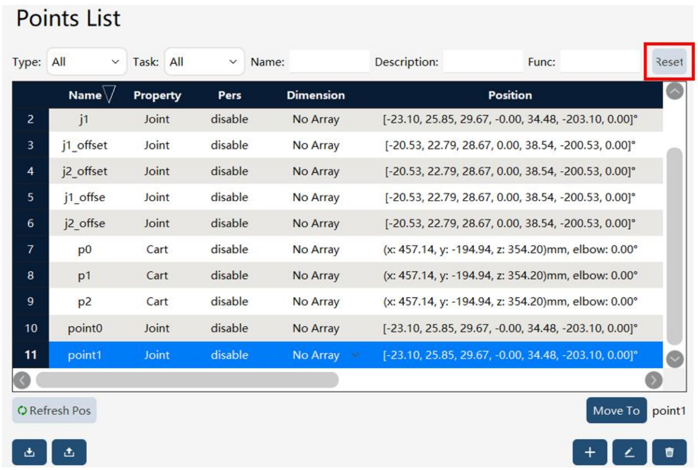
<table>
<tr><td></td><td>
</td><td></td></tr>
</table>

8.8.2.2Point "Move to"

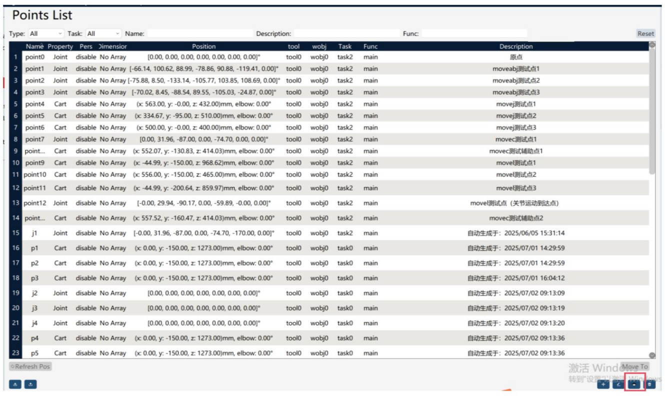
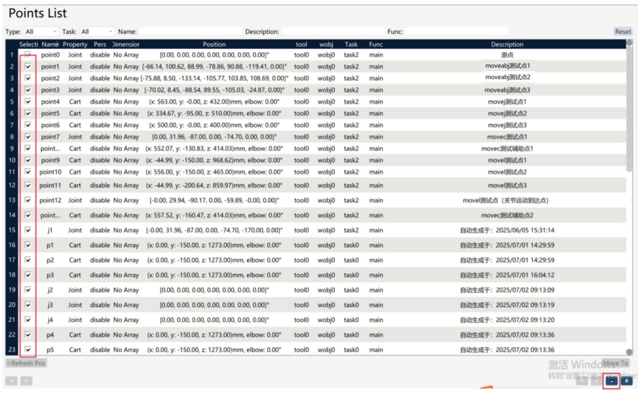
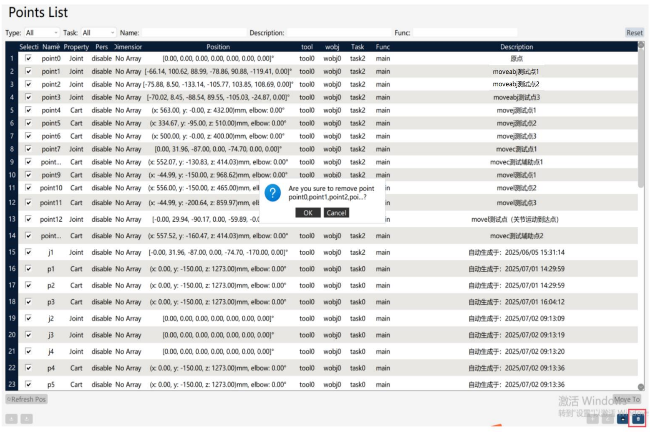
<table>
<tr><th>Operation</th><th>Graphical Representation</th><th>Explanation</th></tr>
<tr><td>1. Select the corresponding point and click the "Move to" button</td><td>
</td><td></td></tr>
<tr><td>2. Select the tool and work object for point motion</td><td>
</td><td>The "Move to" button can only be clicked without clicking the "Confirm" button if the selected tool and work object match those in the point information. If they do not match, the "Move to" button is grayed out and cannot be used directly; motion requires clicking the "Confirm" button first.</td></tr>
<tr><td>3. Perform the "Move to" operation</td><td></td><td>For joint-type points, only MoveAbsJ motion is allowed. For Cartesian-type points, only MoveJ and MoveL motions are allowed.</td></tr>
</table>

<table>
<tr>
<td></td>
<td>
</td>
<td></td>
</tr>
</table>

8.8.2.3Point sorting

<table>
<tr><th>Operation</th><th>Graphical Representation</th><th>Explanation</th></tr>
<tr><td>1. Upon the first entry to the point list interface, the sorting indicator is displayed by default on the right side of the name</td><td>
</td><td>A downward arrow indicates descending order, and an upward arrow indicates ascending order</td></tr>
<tr><td>2. Click to sort</td><td>
</td><td>The point list will be sorted according to the current sorting symbol. Note: The initial descending symbol when first entering the interface does not indicate that the point list is sorted in descending order until the sorting function has been used.</td></tr>
<tr><td>3. Reset the sorting</td><td>
</td><td>Clicking "Reset" will restore the point list to its state before any sorting was applied.</td></tr>
</table>

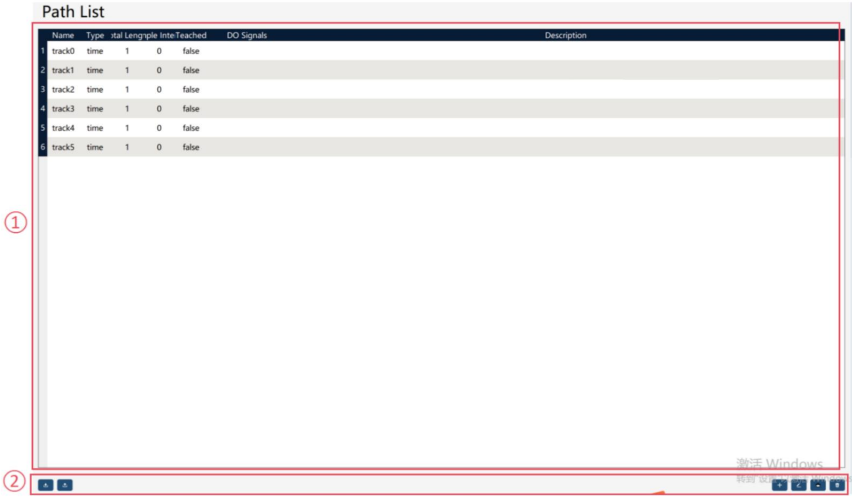
8.8.2.4Batch deletion of points

Operation	Graphical Representation	Explanation
1. Click the Multi-select button		Click the multi-select button to enter multi-selection mode.
2. Select points to be deleted		Check the box before each point to select, or use "Select All" to choose all points.
3. Delete the selected points		Click the Delete button and confirm by clicking "OK" in the confirmation dialog box to delete the selected points.

8.9Path list

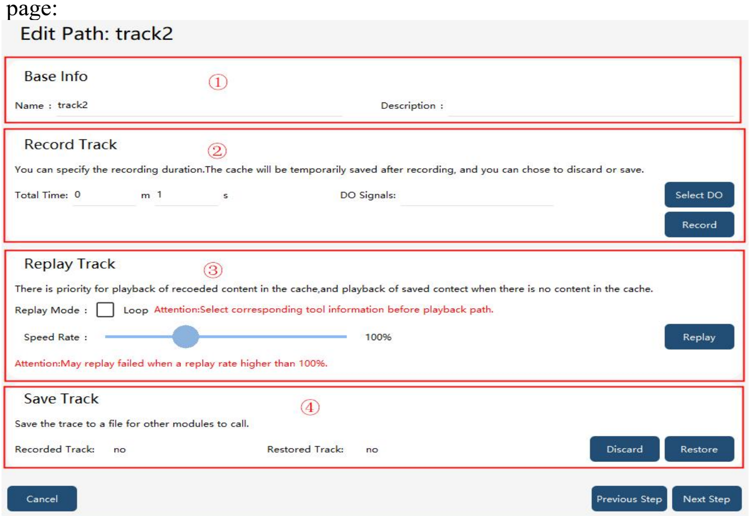
8.9.1Overview

The path list is used to record the trajectories of the drag teaching and perform operations such as trajectory playback.



①	Path list, which displays relevant attributes, including: name, type, total length, sampling interval, whether to include trajectory, DO signal, and description.
②	Function button area: Import Path, Export Path, Create Path, Edit Path, Multi-select Paths, and Delete Path.

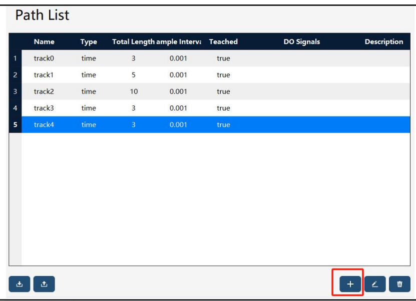
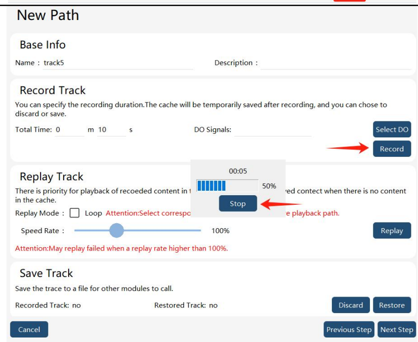
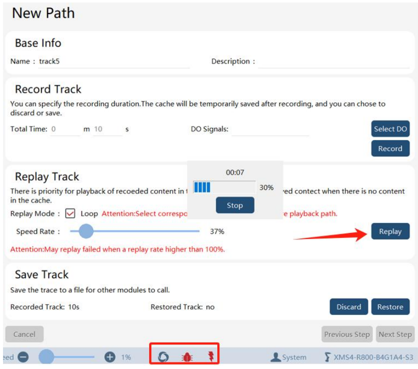
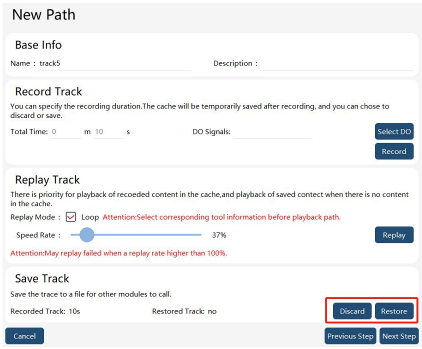
Path editing page:



①	Basic information, for setting path name and description.
②	Path recording. If you need to record DO signals, you click the "Select DO" button and choose the DO signal in the pop-up dialog box. Each DO signal can also be mapped to a DI signal. In this case, changes in the DI signal will be recorded during path recording, and during subsequent playback, the output of the DO signal will be based on the recorded DI signal. If no DI signal is associated, changes in the DO signal will be directly recorded. At this time, the DO signal output can be manually set on the "Status Monitoring" - "IO Signal" interface. Total duration: Total recording time
③	Playback, for play backing the recorded paths. You can set whether it runs in a loop, the running rate, etc. You can click the "Playback" button to play back the recorded path and confirm whether the recorded path is consistent with the expected one.
④	The trajectories can be saved in the cache region for subsequent use. Discard button: Discard this recording

8.9.2Operation examples

Operation	Graphical Representation	Explanation
1. Turn on the drag mode. Click the "+" button in the lower right corner to create a new path.		This operation can also be performed by pressing the end-effector buttons of the robot.

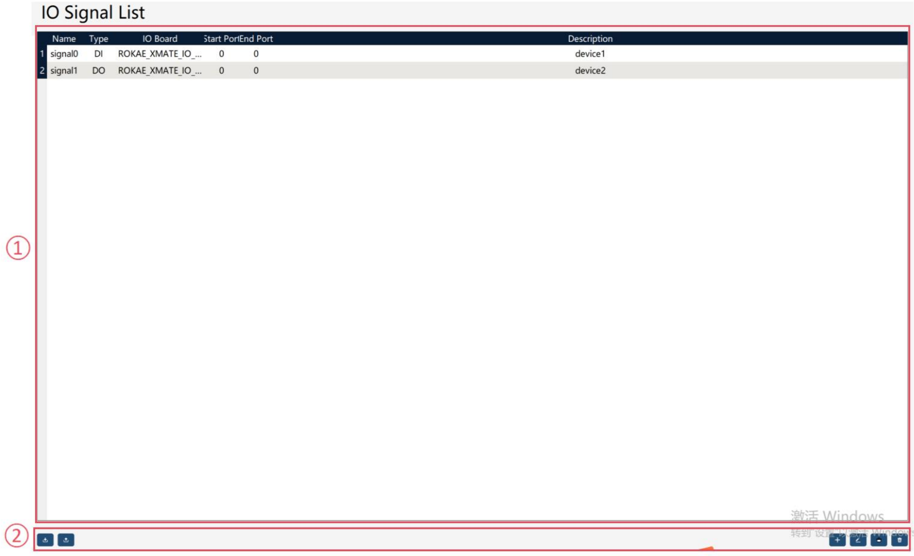
		
2. Set the recording duration, and after pressing the "Record" button, drag the end-effector to record the trajectory until the timing ends or the "Stop" button is clicked.		Attention: Please stop dragging before the end of the recording time, otherwise, the path recording will fail.
3. Turn off the drag enable button and switch to auto mode. Click the power-on button to power on the robot. Set the replay speed rate and replay mode, and click the "Replay" button to view the trajectory playback.		
4. Confirm the trajectory after the trajectory playback is finished. Click the "Restore" button to save the recorded trajectory as a file successfully.		

8.10IO signal list

8.10.1Overview

In addition to the default Universal IO signals created by the "IO device" in the enabled state, if you want to use the IO device alone to create a user IO signal, you need to create it on the "IO Signal List" page.

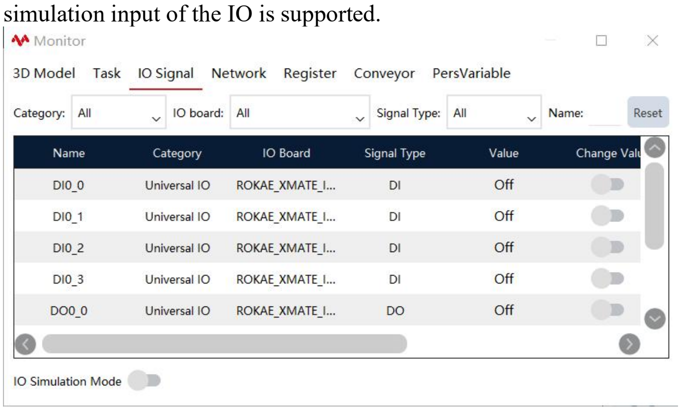
The signalxx type variables are used to store and access IO signals in the RL program. For details, refer to relevant sections about RL commands.



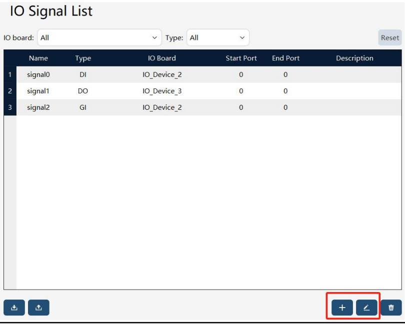
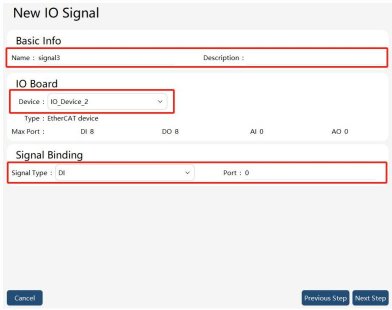
①	IO signal list, where you can view IO signal attributes, including: ● Signal name; ● Type: including DI/DO/GI/GO; ● IO board: It can be a standard IO module provided by ROKAE, and can also be a Profinet bus or Ethernet/IP bus device; ● Start port and end port: the corresponding physical address that a IO signal mapped.
②	Function button area, where you can import, export, create, edit, multi-select, and delete IO signals

> Warning
>
> 1. If there is an error in the IO configuration, for example, when the mapped IO port exceeds the physical limits or the IO is repeatedly assigned, the controller system will enter the SYS_ERR state at the system starting and give an alarm message on the HMI. In this case, the user is only allowed to enter the system configuration interface to correct the wrong configuration, with no other operation allowed.
> 2. User IO signals cannot be mapped to system outputs.

The configured IO can be viewed on the status monitoring interface, on which the forced output or simulation input of the IO is supported.



8.10.2Operation examples

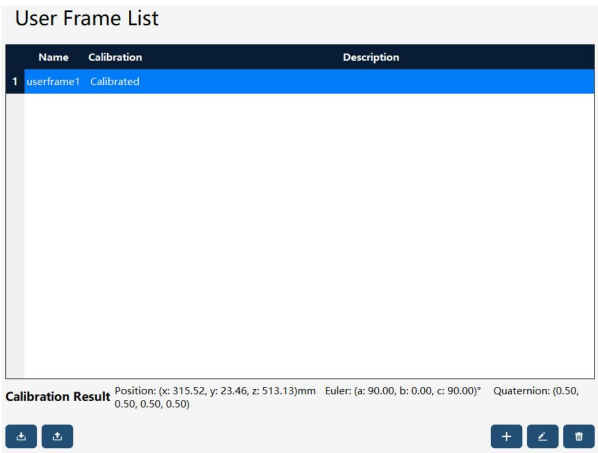
Operation	Graphical Representation	Explanation
1. Click the "+" button in the lower right corner to create a new IO.		Users with the permission level of "Programmer" or above can perform creation and edit operations.
2. Attribute settings.		Set IO signal name, description, IO board, signal type, and port.

3. Using IO Signal in the RL program.		For the input signal (DI/GI), the state of the input node can be read directly in the RL program using the variable name of the input signal.
		For example: Use the state of the digital input as a criterion for judgment
		IF (di1 == true)
		do something…
		ENDIF
		For the output signal (DO/GO), specialized commands SetDO and SetGO can be used in the RL program. See the Explanation of each command below for details.

8.11User frame list

8.11.1Overview

The user frame is used as a reference frame when defining the work object frame, and it cannot be used separately.

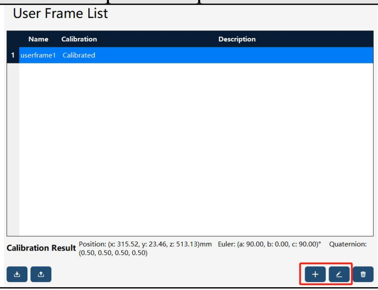
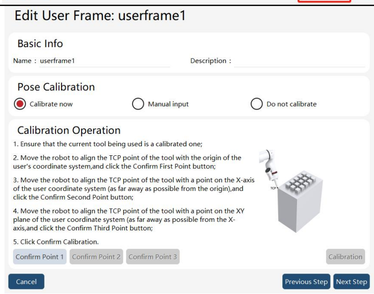


When establishing a user frame, you can choose "Calibration now", "Manual input" or "Do not calibrate".

When "Calibration now" is selected, the 3-point method is used to calibrate. Before calibrating the

user frame, the user needs to first calibrate a tool and then use the TCP of the tool to calibrate the user frame. For this, it is recommended to use a tool with tips.
"Manual input" is allowed if the user frame is known in advance. Another option is "Do not calibrate", in which case the user frame is considered as the world frame by default.

8.11.2Operation examples

Operation	Graphical Representation	Explanation
1. Click the "+" button in the lower right corner to create a new user frame.		Users with the permission level of "Programmer" or above can perform creation and edit operations.
2. Follow the steps shown in the figure to perform the calibration.		Jog the robot to guide the calibrated tool TCP to sequentially point to the desired frame's origin, a point on the x-axis, and a point on the xy-plane or on the y-axis. Click the "Confirm Point 1", "Confirm Point 2", and "Confirm Point 3" buttons accordingly. After successfully confirming all three points, click the "Calibration" button.

8.12Tool list

8.12.1Overview

A tool is a device that is installed on the flange of a robot to complete a specific processing procedure. Common tools include pneumatic/electric grippers, welding guns, and sprinklers. No tool is attached to the robot when it is delivered from the factory, and you need to purchase or design appropriate tools according to the actual situation and complete the installation and settings in order to make the robot work.
Any tool shall be calibrated to get the TCP data before it is used.
In the xCore controller system, the data type corresponding to tools is "tool". For detailed explanations of the "tool", please refer to the section "RL Commands-Variables".

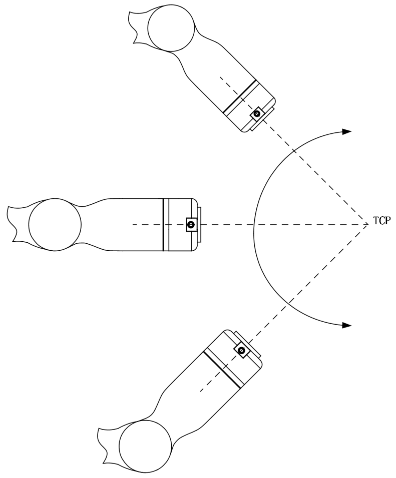
8.12.2Basic concept

Tool attributes include: center point and orientation, which represent the geometric parameters of the tool; and weight, center of mass and rotational inertia, which represent the dynamic parameters of the tool.

> **Note**
> tool0 is a tool variable pre-defined by the system. Its tool frame coincides with the flange frame and both share the same dynamic parameter of 0. The tool0 variable is not allowed to be modified.

8.12.2.1Tool center point

Tool Center Point (TCP) is a specific point on the tool that is normally used by the robot to carry out processing work, such as the wire tip of a welding gun and a tip of a pneumatic gripper. The robot can rotate around the TCP and change its orientation while keeping the position of the TCP unchanged.



Different tools may have different TCP, and determining an appropriate TCP according to actual conditions can significantly increase programming efficiency.

> Note
>
> Unless otherwise specified, all references to "robot position, velocity, acceleration" in the Manual refer to the position, velocity, and acceleration of TCP relative to the work object frame.

8.12.2.2Tool frame

The calibration of tool frame refers to the process of determining the pose of the tool frame relative to the flange frame.

If the pose information of the tool relative to the flange is known, you can select "manual input" on the teach pendant and input it directly without performing the calibration process.

If the pose information of the tool relative to the flange is unknown, xCore provides three methods for tool frame calibration:

- Four-point method, used to calibrate the center point of the tool frame.
- Three-point method, used to calibrate the orientation of the tool frame.
- Six-point method, used to calibrate the center point and orientation of the tool frame at the same time, which is equivalent to the integration of the four-point method and the three-point method.

Additionally, xCore provides the "TCP correction function" to further improve the position accuracy after tool frame calibration.

8.12.2.3Load parameters

The xCore system utilizes load-type variables to store the load parameters of tools. Attention: When external tools are used, the load parameters in the tool variables store the load of the handheld work object.

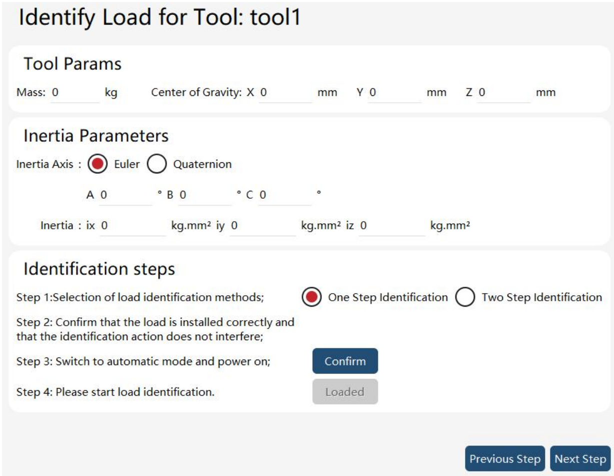
There are two ways to define a tool's load parameters:

If the user knows the tool load data, the user can select the manual input method on the tool frame calibration interface and input the corresponding data directly;

If the tool load data are unknown, they can be obtained using the load identification function of the xCore system.

8.12.2.4Load identification

The load identification function can conveniently calculate the dynamic parameters of the tool.



Two methods are supported for load identification: one-step identification and two-step identification. Industrial robots only support two-step identification, and the precision of two-step identification is usually superior to that of one-step identification.

"Two-step identification" operation procedure:
1. Switch the robot to the automatic mode and power on;
2. Run the empty load identification program in the no-load state and wait for the program to complete;
3. Mount the tool load, run the load identification program, and wait for the program to complete;
4. When the identification is completed, the identification result window pops up, and you click "Save" to save.

Collaborative robots support "one-step identification", whose operation procedure is as follows:
1. Select the one-step identification method for load identification;
2. Install the load and ensure that it is properly mounted and that there is no interference during the identification process;
3. Switch to automatic mode and power on;
4. Click the "Loaded button", run the identification program, and wait for the program to complete;
5. Confirm the identification results and save them.

Note

Please make sure to accurately define the dynamic parameters of the new tool. Otherwise, failure to do so will affect the motion of the robot and even damage the robot due to excessive load on some serious occasions.
Before identification, switch on and preheat the robot in advance for more than half an hour to improve the identification precision.
Load inertia calculation is based on the flange frame.
Load identification is only supported when the robot is installed upright.

Note

The following circumstances during identification will cause the identification to stop and cause any identification data received to be lost. In this case, the user has to re-start the identification:

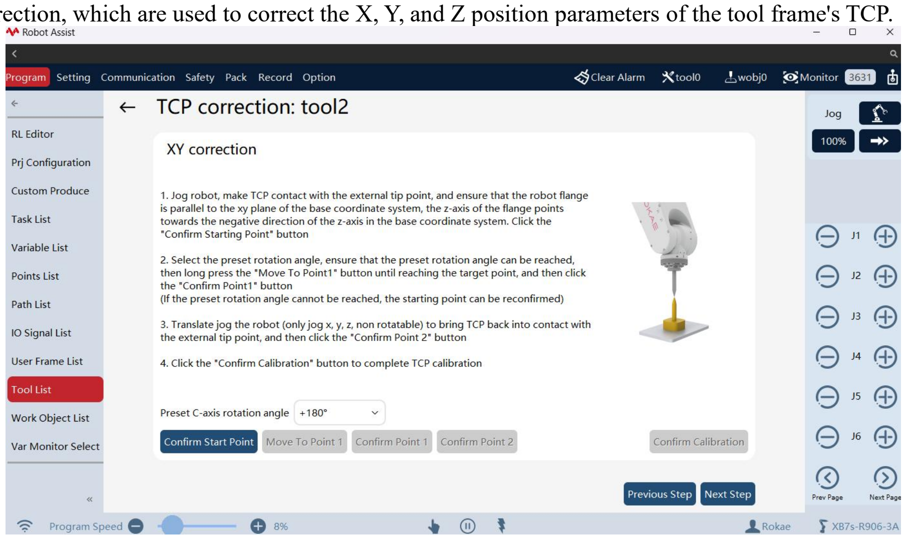
- The user selects other tools or switches to other interfaces halfway through identification;
- The user triggers the emergency stop or safety stop for external parts when the identification program is running;
- The user switches from automatic mode to manual mode when the identification program is running.

Warning

The identification program needs to be executed in the automatic mode, so all prevention measures shall be effective. As the external control signal may start the robot at any time, you need to switch to automatic mode for identification program execution only after the installation is complete and all personnel have retreated to a safe area.

8.12.2.5TCP correction

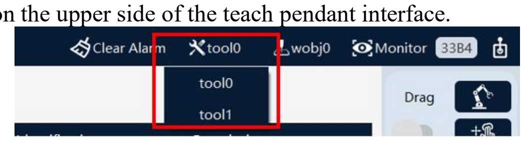
xCore provides the TCP correction function to correct the Tool Center Point (TCP) of the tool frame. For handheld tools that have undergone pose calibration, use the TCP correction function to improve the accuracy of the TCP position. In cases where deformation of the tool's end effector or errors in the tool installation position cause significant deviations between the theoretical TCP and the actual TCP, the TCP correction function can be used to correct and quickly recalibrate the TCP of the tool frame. The TCP correction function currently supports two types of correction: XY correction and Z correction, which are used to correct the X, Y, and Z position parameters of the tool frame's TCP.



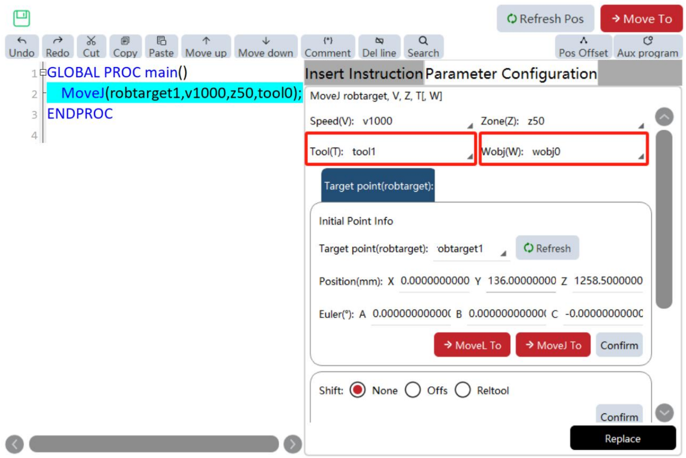
Note
1. The TCP correction function currently supports only handheld tools.
2. The TCP correction function is available for robots in upright, side-mounted, and inverted installations.
3. Only tools that have undergone pose calibration can use the TCP correction function.
4. If the "Gravity Compensation" function is enabled, to ensure the accuracy of TCP correction, please set the tool load accurately.
5. The TCP correction function is only available for use with 6-axis robots.

8.12.2.6Use of tool frame

Used during Jog:
If it is necessary to use a special tool for Jog operation, select the desired tool in the drop-down list of the 'Tool' in the menu on the upper side of the teach pendant interface.



Used in the RL program:
It is very simple to use a special tool in the program, and you just need to use the desired tool in the "Tool" parameter of the motion statement. When the "Aux Program" interface of the teach pendant is used to write motion commands, the default "Tool" and "Wobj" are tool0 and wobj0.



8.12.2.7External tools and handheld tools

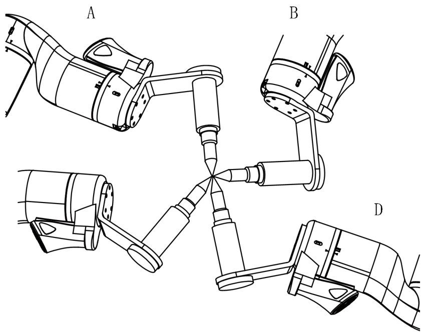
In most cases, a tool is installed on the robot, and the tool moves with the robot to complete specified work. Such a tool is called handheld tool. Typical handheld tools include: grippers, suction cups, and welding guns.

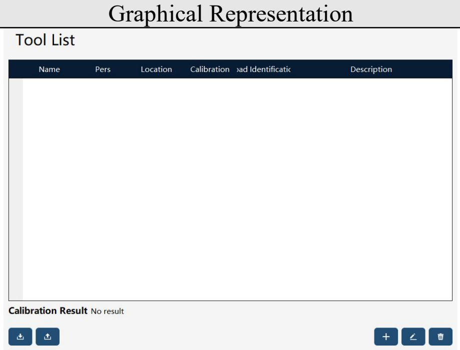
In certain specific situations, installing a tool onto the robot may affect its normal usage, such as during grinding or gluing. In these cases, it would be more appropriate to mount the work object on the robot and fix the tool at a specific external location. We call these tools that are installed outside the robot and fixed at a certain location external tools (some brands call them Stationary Tool or Remote TCP), and the corresponding work objects are called handheld work objects.

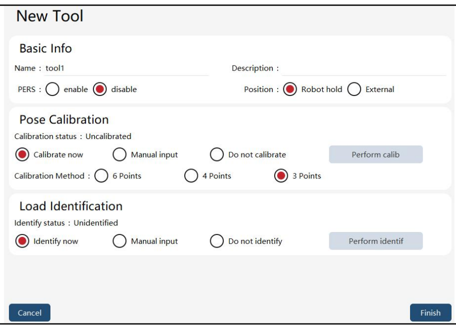
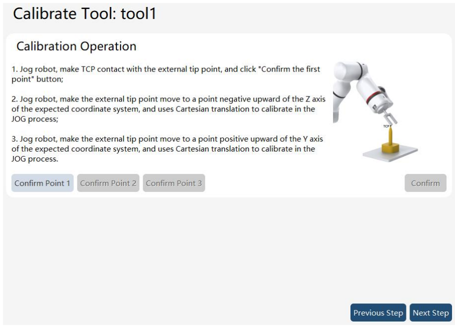
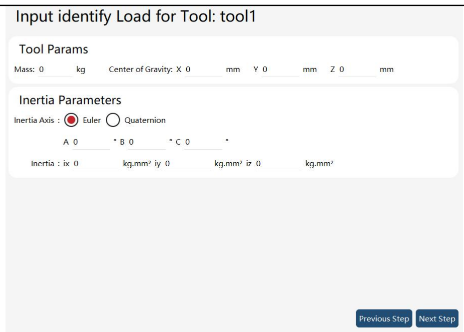
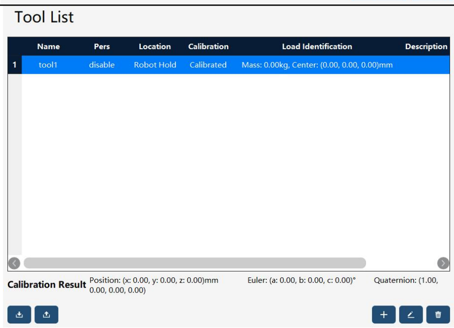
8.12.3Operation examples

8.12.3.1New handheld tool

Before the calibration of tool frame, the user needs to prepare a fixed external point, which shall be located within the robot's working range and can be contacted by the calibrated tool in a very flexible orientation.



Operation	Graphical Representation	Explanation
1. Click "+" in the bottom right corner of the tool list to enter the New Tool wizard interface.		Users with the permission level of "Programmer" or above can perform creation and edit operations.

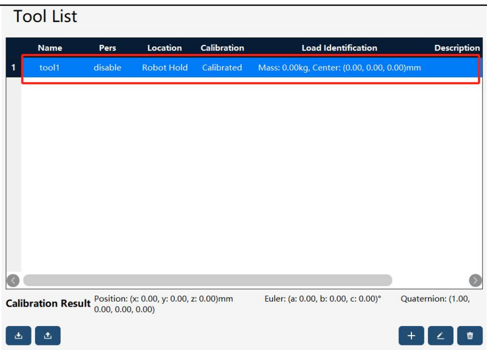
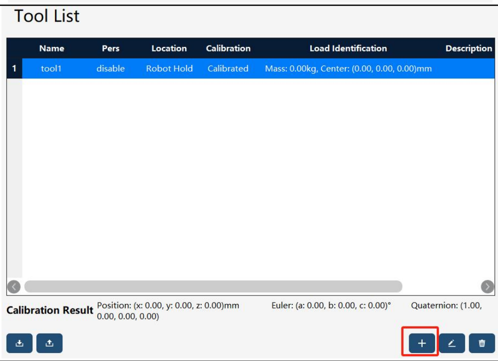
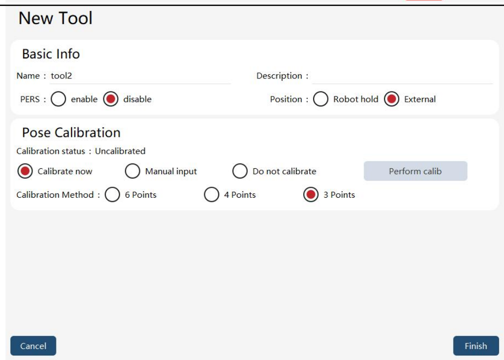
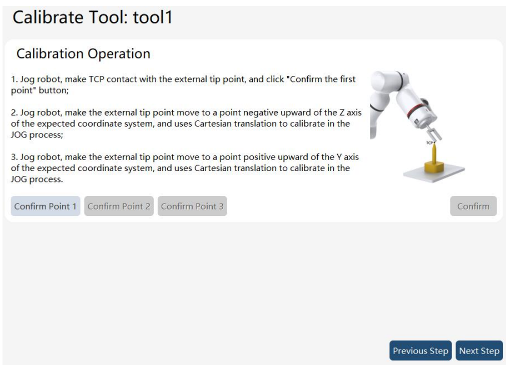
2. Set the tool attribute: Robot hold.		
3. Pose calibration, taking the three-point method as an example: Select "Calibrate now" and set the calibration method to "3 Points". Then, click the "Perform calib" button to enter the "New User Frame" page to perform the calibration.		Jog the robot to move the calibrated tool TCP to the points of the desired frame in sequence: the origin, a point on the x-axis, and a point on the xy-plane or on the y-axis, and click the "Confirm Point 1", "Confirm Point 2", and "Confirm Point 3" buttons accordingly. After confirming all three points, click the "Confirm" button to return to the "New Tool" page, where the "Calibration Status" will now be displayed as "Calibrated." When the four-point method is used to calibrate the tool origin, the orientation differences between the four points shall be as large as possible. In other words, the robot shall try to contact the external point in different orientations.
4. Load identification (taking manual input as an example): Select "Manual Input," click the "Perform manual input" button, and enter the "Tool Load Identification" page.		Input the physical information of the tool, click the "Next Step" button, and return to the "New Tool" page. At this point, the "Identify status" will be displayed as "Identified".
5. Click the "Next" button to complete the creation of the new external work object.		The newly created external work object will be displayed in the work object list.

8.12.3.2New external tool

The calibration methods for external tools are consistent with those for handheld tools, supporting three methods: six-point method, four-point method, and three-point method.

Attention: To calibrate the external tool frame, it is necessary to use the already calibrated handheld tool.

Operation	Graphical Representation	Explanation

1. If a calibrated handheld tool already exists in the list, it can be used to assist in the calibration of an external tool. In this example, we will use "tool1" as the reference.		The external tool to be calibrated shall have a tip.
2. Click "+" in the bottom right corner of the tool list to enter the New Tool wizard interface.		Users with the permission level of "Programmer" or above can perform creation and edit operations.
3. Set the tool attribute: External.		
4. Switch between tools in the upper right corner and select the already calibrated handheld tool, "tool1".		
5. Pose calibration, taking the four-point method as an example: Select "Calibrate now" and set the calibration method to "4 Points". Then, click the "Perform calib" button to enter the "Tool Calibration" page to perform the calibration.		Jog the robot so that the TCP of the calibrated tool can point at the origin of the desired work object frame in different orientations, and then confirm the first, second, third, and fourth points respectively. The orientation difference between the 4 points shall be as large as possible, which means the robot shall try to contact the external point in different orientations. After the calibration is completed, the system will pop up the calibration error. Select whether to re-calibrate according to the error situation (refer to the confirmed calibration accuracy).

Note:

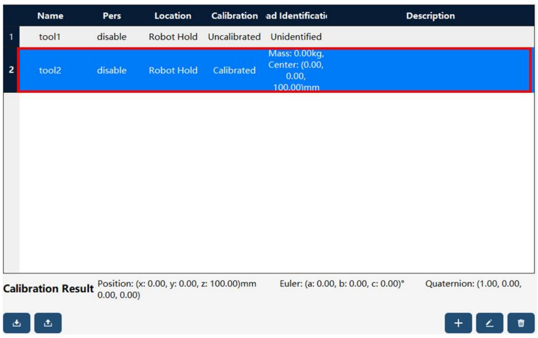
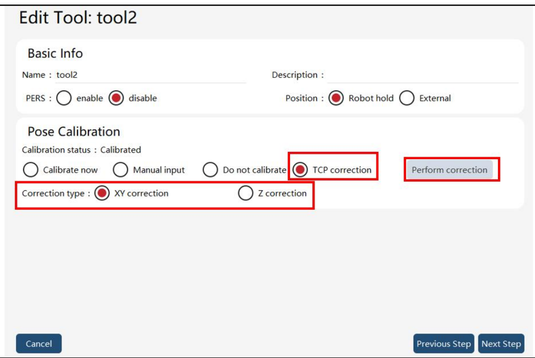
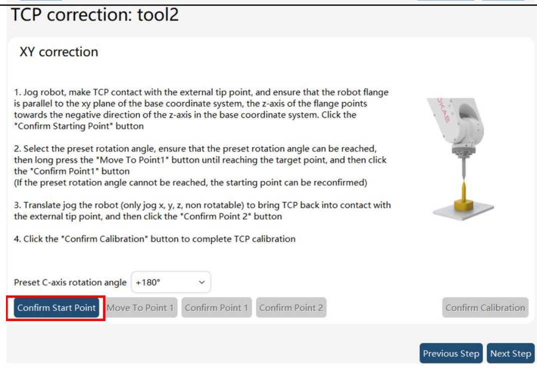
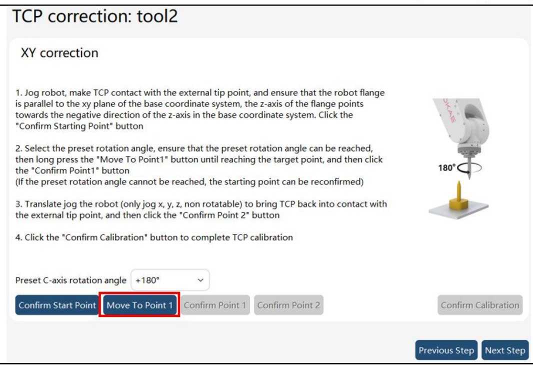
The external tool must be used together with the corresponding work object, meaning among the Position parameters which are selected at the same time in the tool and work object respectively, one must be External while the other be Robot hold. Otherwise, the system will prompt an error and forbid jogging the robot.

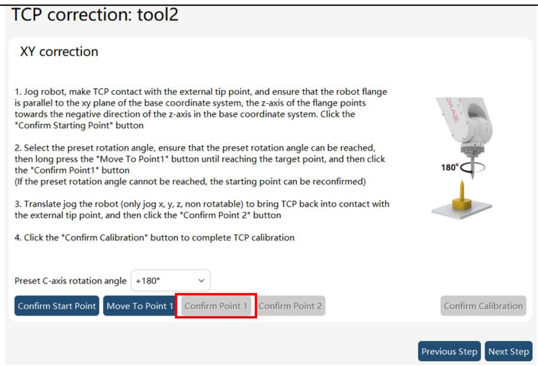
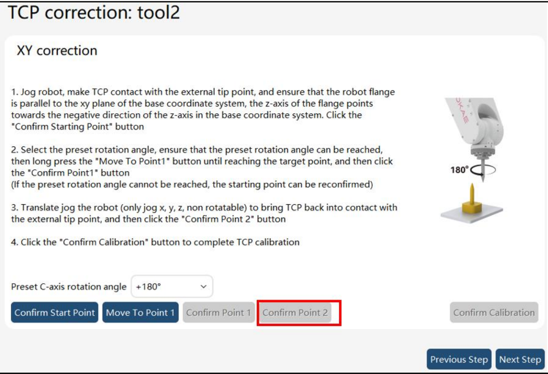
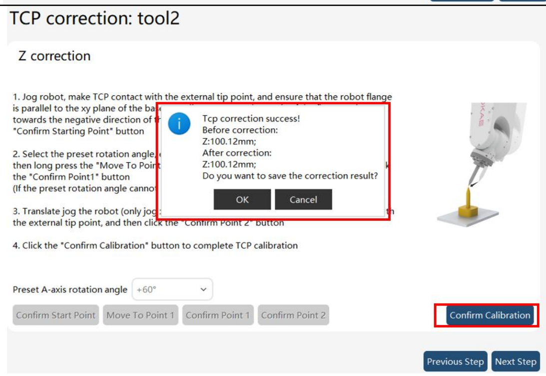
The reference frames for defining tool frames and work frames of external tools differ from those for defining tool frames and work frames of normal tools. You can refer to the following table.

Frame name	Definition of a normal tool relative to …	Definition of an external tool relative to …
Work object frame	User frame	User frame
User frame	World frame	Flange frame
Tool frame	Flange frame	World frame

8.12.3.3Use of TCP correction

Before using the TCP correction function, prepare a fixed external tip point that must be within the robot's working range and accessible by the tool to be calibrated in a relatively flexible orientation.

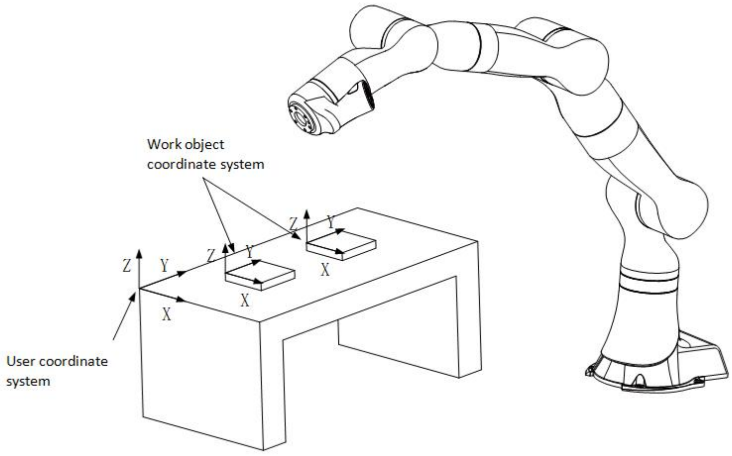
Operation	Graphical Representation	Explanation
1. In the tool list interface, select the tool that needs correction and click the "Edit" button in the lower right corner to enter the Edit Tool wizard interface.		Users with the permission level of "Programmer" or above can perform creation and edit operations. The external tool to be calibrated shall have a tip. The tool needs to be calibrated and have its load set, and it must be a handheld tool.
2. Using "XY correction" as an example Enter the pose calibration interface. Select the "TCP Correction" button, choose the correction type, and then click the "Execute Correction" button to enter the specified type of TCP correction interface.		TCP correction supports two types of corrections: "XY Correction" and "Z Correction". "XY Correction": Correct the X and Y parameters of the tool's TCP position. "Z Correction": Correct the Z parameter of the tool's TCP position.
3. Adjust the tool's pose and confirm the TCP correction starting point		Jog the robot so that the tool tip contacts the external tip, ensuring that the robot flange is parallel to the world frame's XY plane and the flange's Z axis points in the negative direction of the world frame's Z axis. Then, click the "Confirm Starting Point" button.
4. Set the preset rotation angle and move to the first point		Select a preset rotation angle, ensuring that it is reachable. Then, long-press the "Move to First Point" button until the target point is reached. (If none of the preset rotation angles are reachable, reconfirm the starting point)

5. Confirm the first point of TCP correction		After the robot moves to the target point, click the "Confirm First Point" button.
6. Confirm the second point of TCP correction		Jog the robot in translation (only jog X, Y, Z; no rotation) so that the tool tip contacts the external tip point again, and then click the "Confirm Second Point" button.
7. Confirm correction and set correction result		Click the "Confirm Correction" button to complete the TCP correction. The interface displays a pop-up window showing the TCP correction result. Click the "Confirm" button to set the tool's TCP to the corrected result, or click "Cancel" to keep the tool's TCP as it was before correction.

8.13Work object list

8.13.1Overview

Work object refers to the object that is processed or handled by a robot with a tool.
The xCore system uses wobj (Work Object) type variables to describe an actual work object. Defining a work object means creating a wobj variable.
The motion trajectories of robots are defined within the work object frame for two good reasons:
(1) When the work object moves or multiple identical work objects are being processed, the user only needs to recalibrate the work object frame, and all the paths in the program can be updated accordingly without the need to rewrite all paths in the program;
(2) It allows the processing of the work objects that are moved by an external axis (such as track and positioner);
Each work object is jointly defined by two frames: one is user frame, which can be understood as the workbench/table where the work object is placed and is particularly useful when multiple identical work objects are handled; and the other is work object frame, which can be interpreted as the work object itself on the workbench. The path points of the robot are described based on the tool position relative to the work object position.

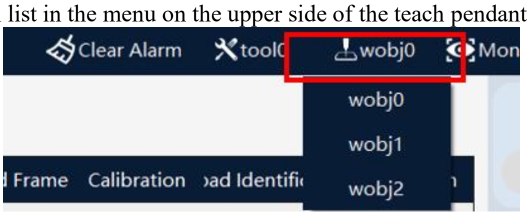


For using the external tool function, the corresponding work object shall be installed on the robot. In this case, the work object is called handheld work object. The handheld work object also needs the calibration of the work object frame and must use the calibrated external tool for calibration. For more details, please refer to the external tool function.

> Note
>
> Wobj0 is a work object variable pre-defined by the system, and both its user frame and work object frame coincide with the world frame.
> Same as tool0, wobj0 cannot be modified as well.
> For PCB 3- or 4-axis robots, the work object frame only supports manual input. The components of orientation A and C are set to 0, and manual user modification is prohibited.

8.13.2Use of work object frame

Used during Jog:
If it is necessary to perform Jog operation in a special work object frame, select the desired work object in the drop-down list in the menu on the upper side of the teach pendant interface.



Used in the RL program:
It is very simple to use a special work object in the program, and you just need to use the desired work object in the "Wobj" parameter of the motion statement. When programming the motion commands on the "Aux Program" interface of the teach pendant, the "Tool" and "Wobj" in default are consistent with those used during Jog operation, which are the currently selected "Tool" and "Wobj" in the menu on the upper side of the interface are currently selected. For the detailed operation steps, please refer to Insert Command.

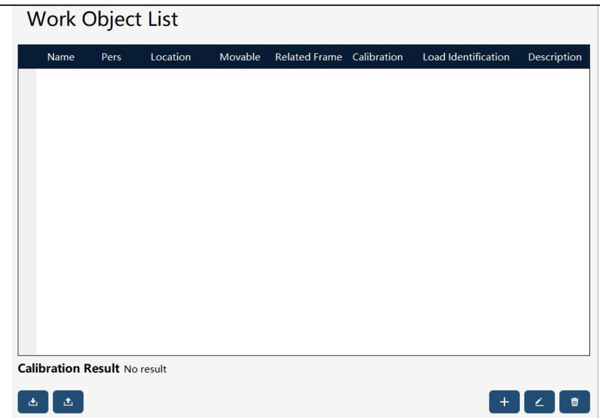
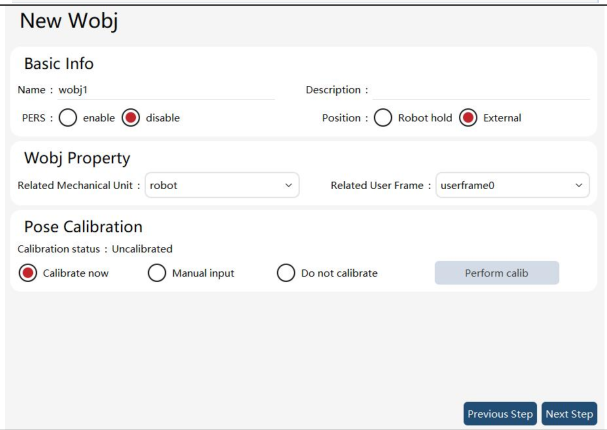
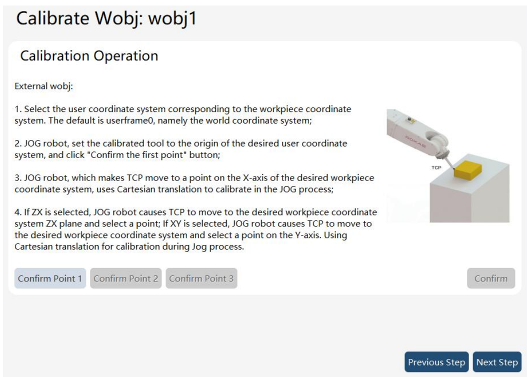
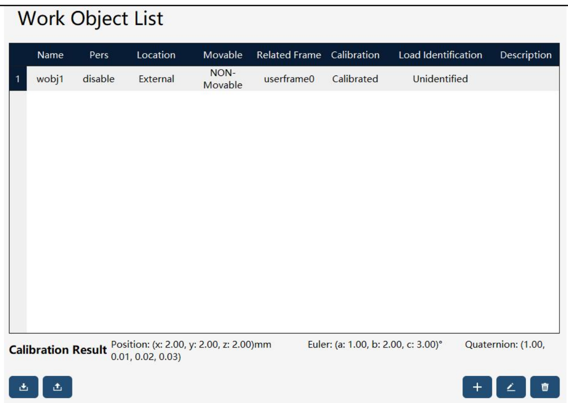
> Note
>
> Generally, the work object parameter for the motion command is optional. Therefore, unless otherwise specified, the system will use wobj0 by default, which coincides with the world frame.
> To use the external tool function, all work object parameters corresponding to the tools must be designated.

8.13.3Operation examples

8.13.3.1New external work object

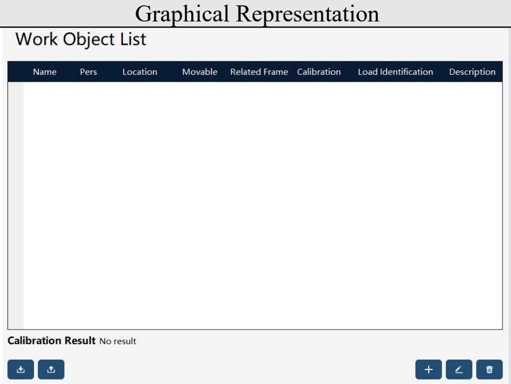
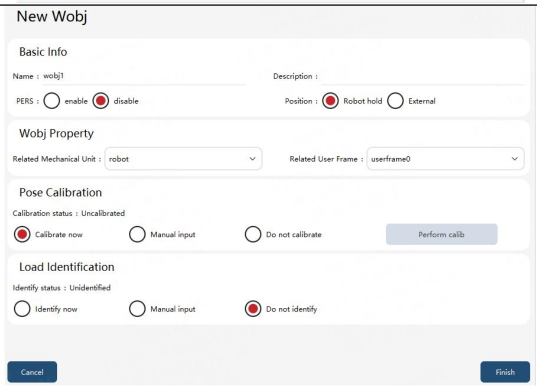
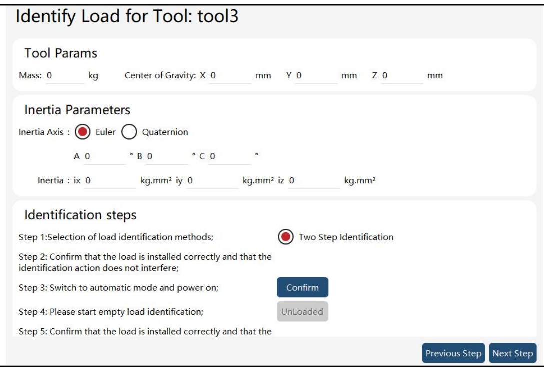
To calibrate the external work object frame, it is necessary to use the already calibrated handheld tool for assistance.

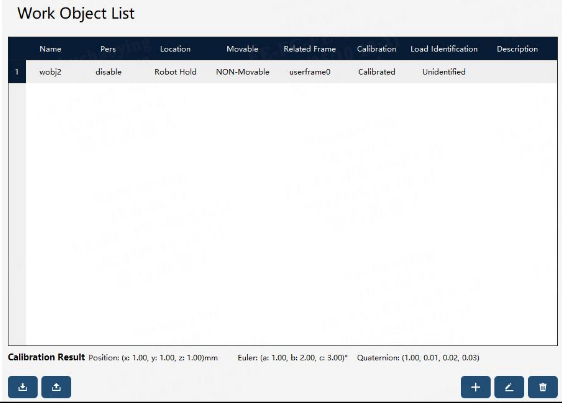
Operation	Graphical Representation	Explanation

<table>
<tr><td>1. Click "+" in the bottom right corner of the work object list to enter the New Work Object wizard interface.</td><td>
</td><td>Users with the permission level of "Programmer" or above can perform creation and edit operations</td></tr>
<tr><td>2. Set the work object attribute to external.</td><td>
</td><td></td></tr>
<tr><td>3. Switch the tool in the upper right corner to the already calibrated handheld tool, "tool1"</td><td>
</td><td></td></tr>
<tr><td>4. Click the "Perform Calibration" button to enter the work object calibration interface.</td><td>
</td><td>Jog the robot to point the calibrated tool TCP to the desired origin of the work object frame, and click the "Confirm Point 1" button. Then, Jog the robot to point the TCP to a point on the desired X-axis of the work object frame, and click the "Confirm Point 2" button. After that, Jog the robot to point the TCP to a point on the desired Y-axis of the work object frame, and click the "Confirm Point 3" button. Finally, click "Confirm Calibration" to bring up a dialog box for confirming the calibration results, and click "Confirm." Then, Click the "Next" button to return to the "New Work Object" page.</td></tr>
<tr><td>5. Click the "Next" button to complete the creation of the new work object.</td><td>
</td><td>The newly created work object will be displayed in the work object list.</td></tr>
</table>

8.13.3.2New handheld work object

To calibrate the handheld work object frame, it is necessary to use the already calibrated external tool for assistance.

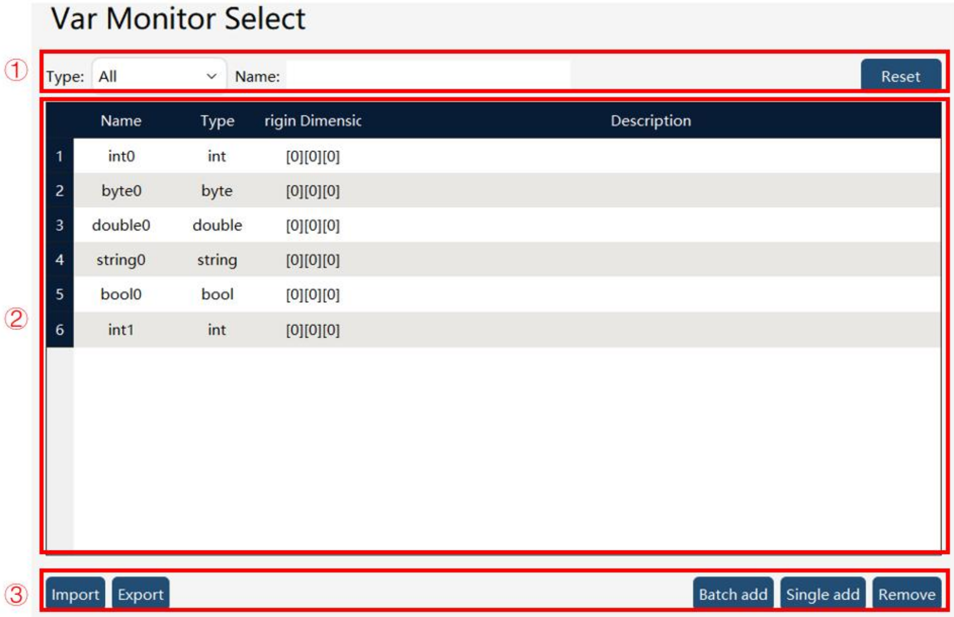
Operation	Graphical Representation	Explanation
1. Click "+" in the bottom right corner of the work object list to enter the New Work Object wizard interface.		Users with the permission level of "Programmer" or above can perform creation and edit operations.
2. Set the work object attribute: handheld		"External" refers to the case where the work object is not fixed relative to the end-effector of the robot, while "handheld" means the case where the work object is fixed relative to the end-effector of the robot.
3. Switch the tool in the upper right corner to the already calibrated external tool, tool2		
4. Load identification is equivalent to tool load identification		
5. Click the "Execute Calibration" button to enter the work object calibration interface.		Jog the robot to point the calibrated tool TCP to the desired origin of the work object frame, and click the "Confirm Point 1" button. Then, Jog the robot to point the TCP to a point on the desired X-axis of the work object frame, and click the "Confirm Point 2" button. After that, Jog the robot to point the TCP to a point on the desired Y-axis/XOY-plane of the work object frame, and click the "Confirm Point 3" button. Finally, click "Confirm Calibration" to bring up a dialog box for

		confirming the calibration results, and click "Confirm." Then, Click the "Next" button to return to the "New Work Object" page
6. Click the "Next" button to complete the creation of the new work object		The newly created work object will be displayed in the work object list.

8.14Variable monitoring selection interface

8.14.1Overview

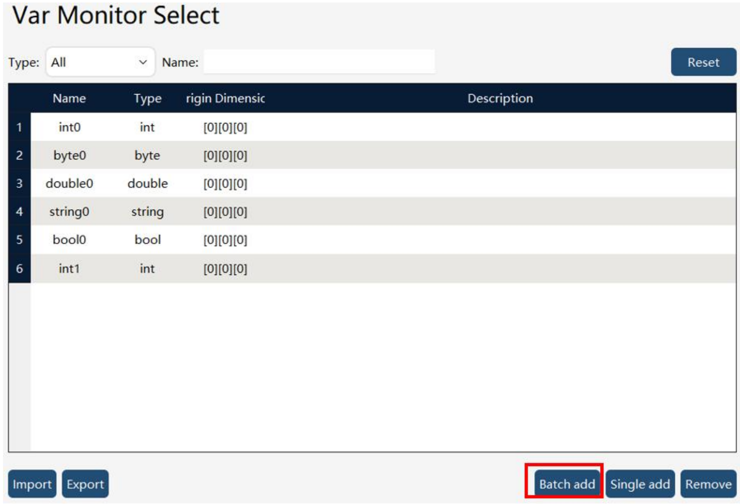
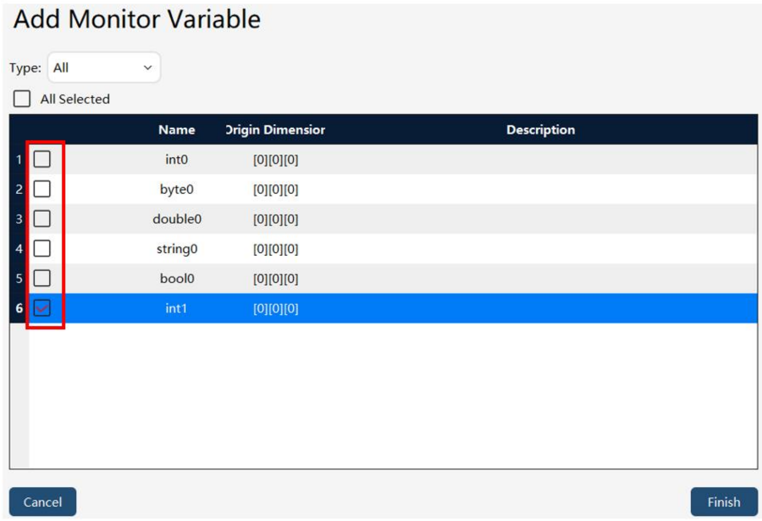
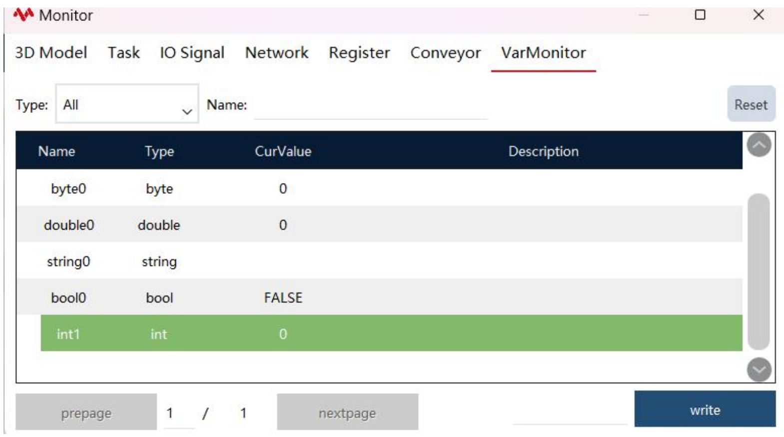
The variable monitoring selection interface is used to add variables that need to be monitored.



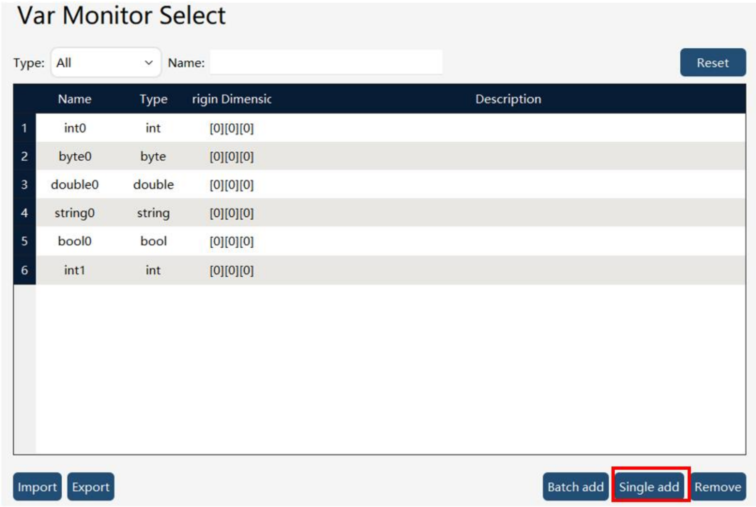
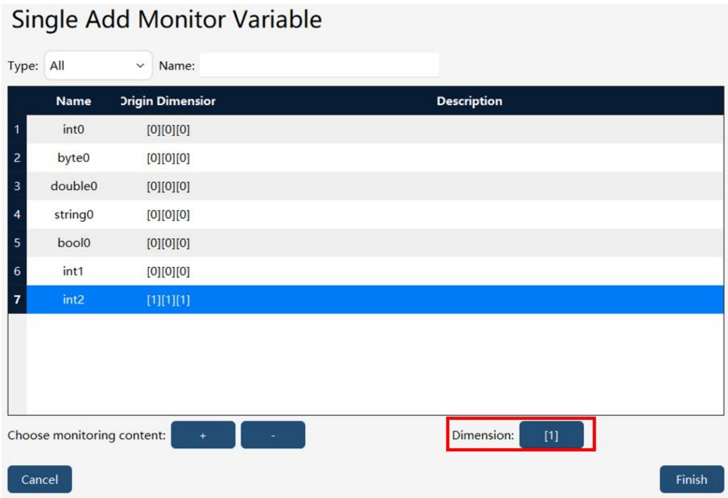
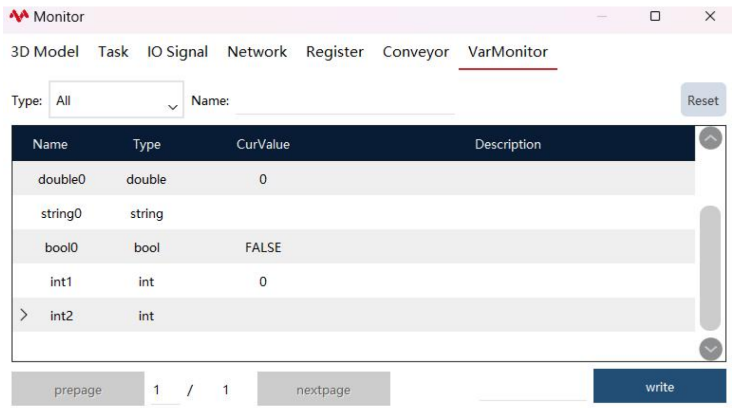
①	Monitored variable filter: Filter to display only the monitored variables needed to be displayed in the list below.
②	Variable monitoring display: Show attributes of each variable, including name, type, original dimension, and description.
③	Function button area: Import import monitored variables, Export export monitored variables, Batch add batch add, Single add single add, Remove delete monitored variables.

8.14.2Operation examples

8.14.2.1Batch add monitored variables

<table>
<tr><th>Operation</th><th>Graphical Representation</th><th>Explanation</th></tr>
<tr><td>1. Click the "Batch Add" button</td><td>
</td><td></td></tr>
<tr><td>2. Select the variables you need to monitor, check the boxes within the red frame, and then click the "Complete" button.</td><td>
</td><td>Note: When the monitored variable is an array, batch addition will monitor all elements within the array.</td></tr>
<tr><td>3. Open the Variable Monitoring in Status Monitoring to view the monitoring status.</td><td>
</td><td></td></tr>
</table>

8.14.2.2Single add monitored variable

Operation	Graphical Representation	Explanation
1. Click the "Single Add" button		
2. Select the variables you need to monitor, click "Complete," and if the selected variable is an array, you can choose dimensions to monitor specific elements within the array.		Note that dimension selection is only available when the variable is an array.
3. Open the Variable Monitoring in Status Monitoring to view the monitoring status.		

8.15About RL program

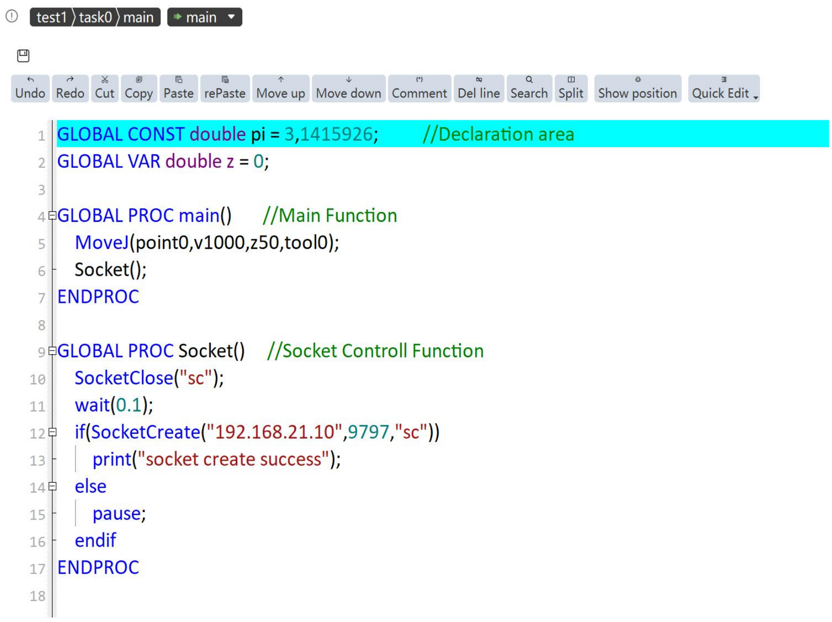
8.15.1RL program format and syntax

8.15.1.1Overview

The name suffix of RL language program file is .mod, and the mod is the abbreviation of module. For example: For MoveObj.mod or PickSomething.mod, each program file forms a program module.
RL language commands are not case-sensitive. For example, MoveAbsJ, moveabsj, and MOVEABSJ all can be recognized. However, in order to maintain a uniform language style, it is recommended to capitalize the initial letters.

8.15.1.2Program structure

Here is a simple RL program:



The entire program is divided into two major sections, the declaration area, and the implementation area. The area before the first function in each Mod file is the declaration area. For example, in main.mod, the part before GLOBAL PROC main is the declaration area. In the declaration area, variables or constants can be defined. The variables defined in this area will be reset to their initial values each time the program is executed.
VAR or CONST keyword represents storage type, with VAR indicating a variable and CONST declaring a constant. If a variable's storage type is not explicitly declared, it defaults to being a VAR.

There are several differences between variables declared in the declaration area and those listed in the variable list:
When a certain task finishes running and is reset, the variables defined in the declaration area within the task will be reset;
The variables in the variable list, owned by the entire project, are the common variables. Non-PERS variables in the variable list are only reset upon execution of pptomain, and PERS variables can only be modified through the editing function in the variable list or by the RL program.

8.15.2RL program debugging

8.15.2.1Program pointer

The program pointer points to the line that has been parsed and run by the program.
On the HMI interface, the program pointer is indicated by a small green arrow (also called the green pointer).

```
GLOBAL PROC main()      //Main Function
  MoveJ(robtarget3,v1000,z50,tool0);
  Socket();
  wait(2);
  MoveJ(robtarget3,v1000,z50,tool0);
ENDPROC
int ww=2;
GLOBAL PROC Socket()    //Socket Controll Function
  SocketClose("sc");
  wait(0.1);
  if(SocketCreate("192.168.21.10",9797,"sc"))
    print("socket create success");
  else
    pause;
```

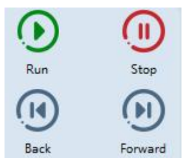
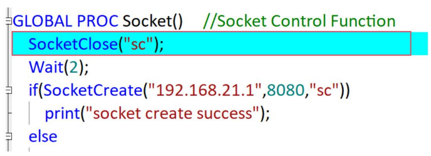
8.15.2.2Motion pointer

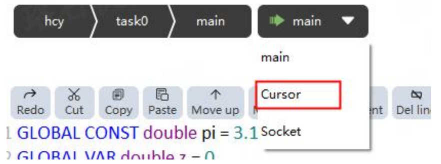
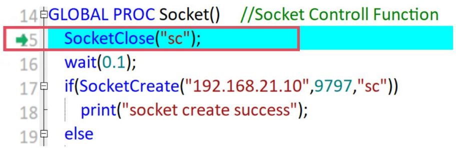
The motion pointer points to the current command the robot is executing;
On the HMI interface, the motion pointer is indicated by a red arrow.

```
GLOBAL PROC main()  //Main Function
MoveJ(robtarget3,v1000,z50,tool0,wobj0);
Socket();
ENDPROC
GLOBAL PROC Socket()
  SocketClose("sc");
  Wait(0.1);
  If(SocketCreate("192.168.0.11",9797,"sc",1,"\r"))
    Print("Socket create success");
  Else
    Pause;
  Endif
ENDPROC
```

8.15.2.3Move program pointer

If you need to start the program after a line from the middle of the program, you can use this function to move the program pointer to
the line where the cursor is, and then the program can be executed from a new position.

Operation	Graphical Representation	Explanation
1. Click the "Stop" button on the right operation panel to pause the RL program.		
2. Click the line to which you expect the pointer to move to make it selected.		The background color is light blue when selected.
3. Click the drop-down arrow next to the Program Debug Quick Positioning button, and select "Cursor" from the list.		After selection, this button displays the word "Cursor".

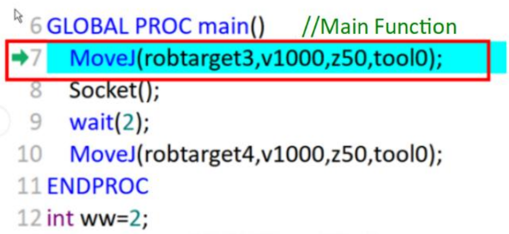
		
4. Click the Program Debug Quick Positioning button to move the pointer to the selected line.		

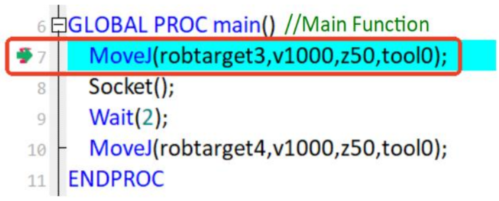
Use restrictions:
1. When using this function, the following commands will be ignored, and the compiler's compile position will be directly moved to the target line.
In addition, all other commands will not be executed:
- All motion commands;
- SetDO, SetGO, Return, Wait, Print, and all Socket commands;
- Function call line;

2. The condition of the flow control command is ignored when moving the program pointer.
3. Do not move the program pointer across functions. It is necessary to first use the "program pointer to function" to move the program pointer to
the beginning of a function, and then use the pointer function of a program.
4. The pointer of a program can only be moved to the motion command line.

8.15.2.4Single-step debugging

The single-step operation status is also known as Single-step Mode, as against the Continuous Mode. The robot can switch between the two modes in most cases.
The single-step operation is mainly used for the program debug. The robot will try to execute one line of commands as much as possible each time it runs in a single step, and pause the program after the commands are completed, making it easy to confirm whether the teaching points of each line meet the requirements. When a multi-task project is being debugged, single-step debugging will only execute the tasks displayed on the HMI debugging interface, and the rest tasks will not be called.
If the single-step debugging executes read data commands (ReadDouble, ReadString, etc.), time-related commands (Wait, WaitUntil, etc.), and logic commands (IF, GOTO, etc.), it will take two to three clicks to complete the command due to the command characteristics.

Operation	Graphical Representation	Explanation
1. Move the pointer to a certain line.		
2. Click the "Next" button on the right operation panel.		

<table>
<tr>
<td>3. Click the "Next" button again, and you can see that the robot is executing the line. After completing the execution, click the "Next" button again to start the next step.</td>
<td>
</td>
<td>If a function is encountered, it will jump to the inside of the function, and the executing line can be located through the program pointer (green arrow).</td>
</tr>
</table>

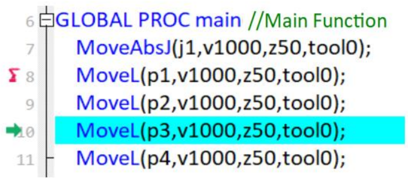
Use restrictions:
1. In Continuous Mode where programs are executed automatically, and the turning zone should be processed, motion lookahead is available.
2. In Single-step Mode where commands are executed directly without processing the turning zone, motion lookahead is not available.
3. In Continuous Mode, motion only starts when there are enough lookahead points, and the system only continues to parse the command when the robot is in place.
4. In Single-step mode, all next-step signals are triggered by the interface, without turning zone processing and lookahead.
5. In Single-step Mode, no response is made when "Next" is clicked during motion.
6. In Continuous Mode, callbacks during motion are responded to according to the lookahead logic.
7. The next step can go to any line and execute the instruction literally. RL programs only process "program commands", without distinguishing between motion commands and logic commands.
8. When the robot pauses on the turning zone in Continuous Mode, the next step will go back to the target point corresponding to the current turning zone.

8.15.2.5Step back debugging

Step back debugging, also known as previous step debugging, allows you to revert directly to the last correct position when a path error is detected during debugging, eliminating the need for multiple JOG operations to exit the erroneous trajectory and thereby improving debugging efficiency.
Non-motion commands typically will be skipped and do not take effect during the step back process
Force control commands, motion setup commands, and logic commands will stop the step back process.

8.15.2.6Step back use

xCore supports switching directly to Step Back or Step Back from the PPTO (Point-to-Point Operation) cursor after pausing
When performing the first Step Back motion after positioning the PPTO cursor, since there is no trajectory information available, the Cartesian point will be forcibly converted to a MoveJ motion.

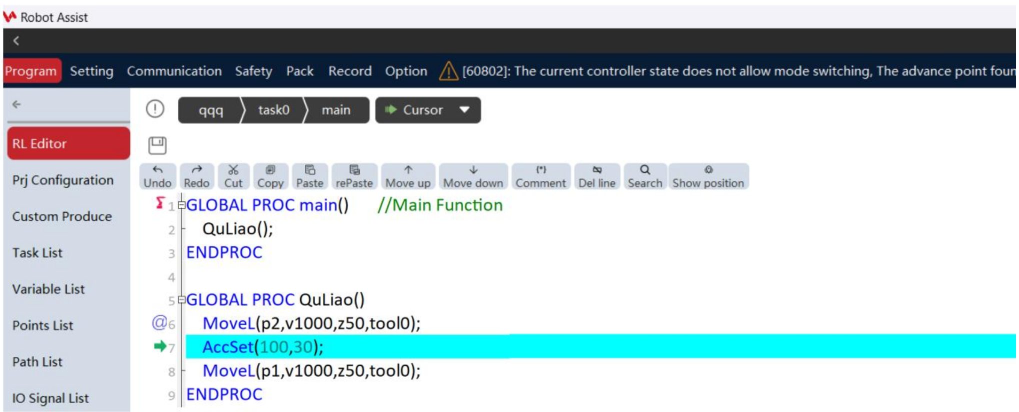
<table>
<tr><th>Operation</th><th>Graphical Representation</th><th>Explanation</th></tr>
<tr>
<td>1. Pause during execution.</td>
<td>
</td>
<td>The lookahead has reached line ten, but the robot is still on the trajectory from J1 to P1</td>
</tr>
<tr>
<td>2. Click the Previous Step button on the right-side operation panel.</td>
<td>
</td>
<td>The lookahead pointer will jump to the motion pointer, indicating that the system has entered step back mode</td>
</tr>
</table>

	GLOBAL PROC QuLiao() MoveAbsJ(j1,v1000,z50,tool0); MoveL(p1,v1000,z50,tool0); MoveL(p2,v1000,z50,tool0); MoveL(p3,v1000,z50,tool0); MoveL(p4,v1000,z50,tool0); ENDPROC	
3. Upon clicking the Previous Step button again, the robot begins the step back motion.	GLOBAL PROC QuLiao() MoveAbsJ(j1,v1000,z50,tool0); MoveL(p1,v1000,z50,tool0); MoveL(p2,v1000,z50,tool0); MoveL(p3,v1000,z50,tool0); MoveL(p4,v1000,z50,tool0); ENDPROC	The robot will revert along the same path it was following at the time of pause (in this case, a MoveL motion) back to point J1 as indicated by the lookahead pointer

8.15.2.7Step back use restrictions

The step back function is primarily used for debugging points, so most non-motion commands are either ineffective or restricted during step back mode. Additionally, a few motion commands cannot be stepped back due to their inherent nature and are also restricted. If an attempt is made to step back through restricted commands, the controller will report an error.

For example:



Supported commands for step back debugging:

Type	Specific Commands
Supported motion commands	MoveAbsJ, MoveJ, MoveL, MoveC, and Home
Special motion commands	SearchL, SearchC, TrigL, TrigJ, and TrigC (Note 1)
Unsupported motion commands	MoveSP, MoveT, and MoveCF
Non-motion commands that terminate step back	Conveyor commands, force control commands, logic commands, function calls, advanced commands (Note 2)
Skipped non-motion commands	All other non-motion commands

Note 1: Function such as search and Trig do not take effect during step back; they only produce motion effects.

Note 2: Commands such as GetRobotState, BreakLookAhead, and GetRobotMaxLoad will be skipped. All other commands in this category will terminate the step back process.

8.15.2.8Regain path

In some specific situations, the robot's position will deviate from its programmed path, for example:

During the period when the program is stopped (except for program stop caused by program reset), the robot is moved to another position by Jog;

The emergency stop is triggered when the program runs, and the robot executes STOP 0;

When the program starts again from the stop position, if the system detects that the robot has deviated from the programmed path,

the robot will then first perform a Regain Path motion to return to the original programmed path.

To ensure safety, the movement speed of the robot is slower when returning to the programmed path, and the movement of the robot can be stopped at any time
by pressing the "Stop" button on the Teach Pendant.

Use restrictions
1. The robot performs a joint trajectory when returning to the path, so the path of the end-effector is unpredictable. Please monitor for potential collisions with the surrounding environment.
2. The control system will only perform path deviation detection when the robot resumes program execution from the point where it was previously interrupted. If a deviation is detected, the system will execute the Regain Path operation.
3. If the program is reset, the system will not check for path deviation and will start execution directly from the first program line. Take precautions to prevent potential collisions.

8.15.2.9Loop mode

Click the "Loop mode" button to switch to loop mode or single mode.

Loop mode:
All tasks are reset after 0.5s when they reach the endproc of the specified function (default is the Main function), and restart execution from the first line of the last specified function.
PPToMain operation takes the main function as the specified function;
PPToFunc takes the jump objective function as the specified function;
PPToLine does not affect the execution target of the function in the next loop.

Single mode:
All tasks (excluding semi-static tasks) are stopped permanently upon execution to the specified function (default is the Main function), until the project is reloaded next start.

8.15.2.10Lookahead mechanism

8.15.2.10.1Basic concept

The lookahead mechanism cannot be turned off. The system automatically looks ahead when running the program. You can use the Program
Pointer to view the lookahead position. From a lookahead perspective, RL commands can be divided into four categories: motion commands, non-stop lookahead commands, turning zone execution commands, and stop lookahead commands.

8.15.2.10.2Motion commands

The commands that control the robot to produce actual motion effect are shown in RL programming commands for details, in which all "Motion commands", "Trigger commands", "Drag and replay commands" and "Home" commands belong to the classification of motion commands.

8.15.2.10.3Non-stop lookahead commands

The command is executed immediately after the lookahead pointer is parsed, and then the lookahead

pointer continues to run downwards to parse the next command, without affecting the turning zone between the two motion commands.
Non-stop lookahead commands: Print command, logical judgment command, variable assignment operation, user-defined function, collision detection dynamic threshold command, and motion parameters dynamic modification command;
Example:

```
MoveL (p1);
IF (condition_1)
   Print (“meet condition 1”);
   MoveL (p2);
ENDIF
MoveL (p3);
```

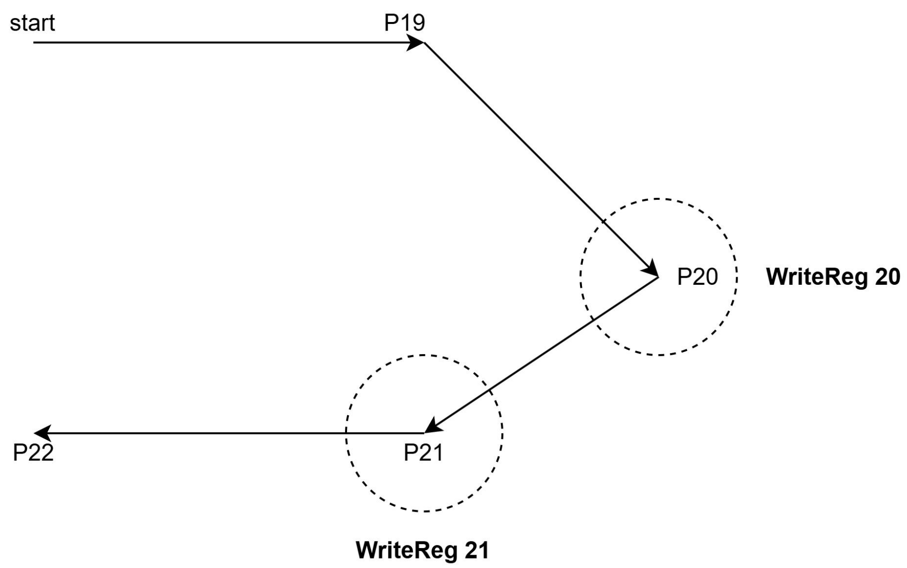
By running the above program, if condition 1 is met, the robot will plan a continuous trajectory motion of p1 > p2 > p3, and print the string "meet condition 1" when looking ahead to the print command.
In the case where condition 1 is not met, the robot will plan a continuous trajectory motion with p1 > p3.

8.15.2.10.4Turning zone execution commands

The turning zone command is executed when the preceding motion command either enters the turning zone or completes its execution.
It is used to send a signal to the external device during the movement, indicating which motion commands the robot has moved to.
Turning zone execution commands: WriteRegByName, SetDO, SetGO, SetAO, PulseDO, PulseReg, InZone，SetVarValue, and SpeedRefrsh.
Example:

```
MoveL P19
MoveL P20
WriteRegByName (reg_position, 20);
MoveL P21
WriteRegByName (reg_position, 21);
MoveL P22
```

When the above motion commands are executed, the robot will plan a continuous motion trajectory of P19 > P20 > P21 > P22. As the robot is about to complete the P20 motion and enters the turning zone from P20 to P21, it will immediately write the number 20 into the register reg_position. Similarly, when the robot is about to complete the P21 motion and enters the turning zone from P21 to P22, it will immediately write the number 21 into the register reg_position. External devices can track the robot's motion progress by simply reading this register.



> **Note**
> The triggering time of the turning zone commands is affected by the performance of the robot

itself, the size of the turning zone, whether the turning zone is generated, and the actual running speed. It is suitable for the scene where the robot is ready to move. If it is required to trigger the signal accurately at a certain position of the trajectory, it is necessary to use trigger series commands.
If there is no turning zone or if a turning zone cannot be generated, the associated command will be triggered after the corresponding motion command is completed.

8.15.2.10.5Stop lookahead commands

Except for the above three commands, all the other commands are stop lookahead commands, and the controller will execute the commands after the robot completes all the movements before the commands.
Example:
MoveL (P1);
MoveL (P2);
Wait (10);
MoveL (P3);
The Wait commands belong to the stop lookahead command. The robot will move to P2 and start waiting for ten seconds after the deceleration stop is completed, and it will start to go to P3 after the waiting is completed.

When a motion command uses a Fine turning zone, the lookahead pointer in the RL program will stop at that line and wait for the movement to complete before continuing with lookahead
Example:
MoveAbsJ (j1,v1000,z50,tool0);
MoveL (p1,v1000,z50,tool0);
MoveL (p2,v1000,Fine,tool0);
// All subsequent commands will only run after the MoveL P2 has finished
MoveL (p3,v1000,z50,tool0);

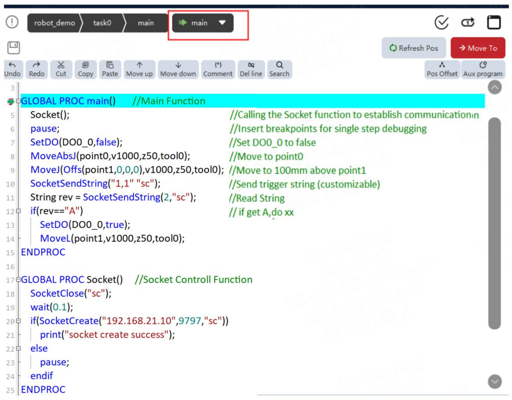
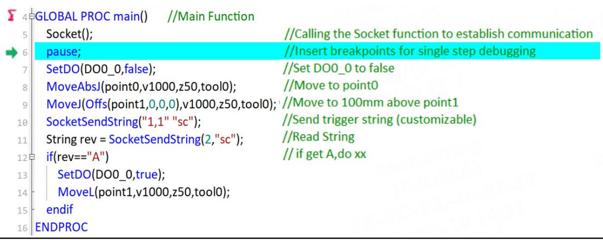
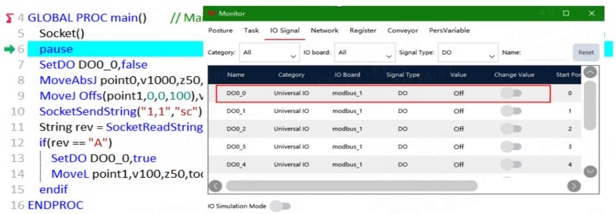
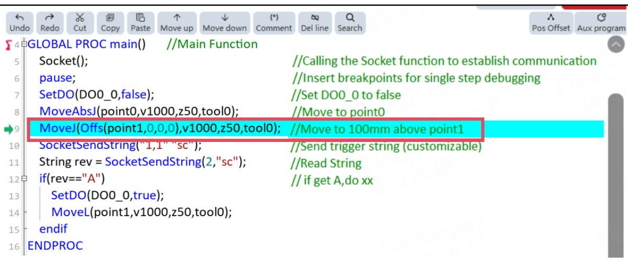
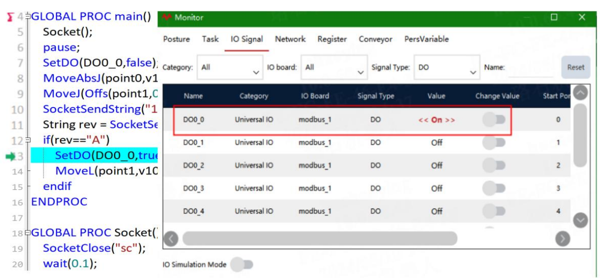
8.15.2.11Interrupt function

```
GLOBAL PROC main()
  IRegister(intnum0,DI3_1,\Posflank,"aaa");
While(true)
  PathRecStart();
  MoveL(p1,v1000,z50,tool0,wobj0);
  MoveL(p2,v1000,z50,tool0,wobj0);
  MoveL(p3,v1000,z50,tool0,wobj0);
  MoveL(p4,v1000,z50,tool0,wobj0);
  MoveL(p5,v1000,z50,tool0,wobj0);
  PathRecStop();
Endwhile
ENDPROC

TRAP aaa
  bool0=GetRecStartStatus();
  If(bool0==true)
    PathRecBwd();
  Else
    Print("the program will pause as the interrupt is not triggered when recording is turned on");
    pause;
  Endif
  //Insert move command(moveL,moveJ,moveC,moveabsj) and other command
  PathRecFwd();
ENDTRAP
```

The interrupt function can be triggered by a DI signal or a register, interrupting the current program execution to run an interrupt function. Key features and considerations of the interrupt function are as follows:
1. The interrupt function can interrupt motion commands, persistent commands (network/communication read commands, connection acceptance commands,
SocketCreate, opendev, SocketClose, CloseDev, SendString, SendByte), and ordinary commands (type conversion, string operations, and other instantly completed commands).
2. If an interrupt occurs during the execution of a motion command, the current motion will stop, and the interrupt function will execute instead. After the commands inside the interrupt function are

completed, the robot will return to the original path and resume execution. The regain path operation after the interrupt function is not part of the interrupt. The regain path function performs motion in joint space. It is recommended to use path recording commands in the interrupt function to ensure the robot correctly returns to a safe position.
3. Except for waituntil, other persistent commands are not truly interrupted; instead, they are moved to background execution. Error handling and task wake-up for these commands will be delayed until the interrupt ends. If a wait command is executed during an interrupt and the wait time expires after the interrupt ends, execution will proceed.
4. During an interrupt, register-read commands and external signal-control commands behave the same as in continuous motion, while register-write and external signal-monitoring commands remain responsive. Pause, emergency stop, and collision can halt interrupt execution. Single-step operation is allowed during an interrupt.
5. Interrupt-related commands can only be used in motion tasks, and only motion tasks can execute interrupts.
6. Interrupts cannot be used while entering force control mode. Once an interrupt is registered, force control mode cannot be initiated with FcInit.
7. For an interrupt configured with single-trigger mode, if it receives an interrupt signal after being disabled by IDisable, it will still be treated as completion of the single trigger
8. After an interrupt is registered with IRegister, it remains triggerable. However, when the task ends or resets, the registered interrupt is canceled until IRegister is executed again.
9. If an interrupt is already being executed, newly triggered interrupts will not be acknowledged.
10. When a task is executing an interrupt, or when normal operation resumes (due to either manual start command or interrupt exit triggering scheduler reset), the system will neither acknowledge new interrupts nor halt the current task.
11. Within interrupt functions, no interrupt-related commands (except GetTrapData) or tray stacking configuration commands can be called. Interrupts are prohibited when: laser welding is active, conveyor startup commands are executing, or end-effector motion commands are in progress — special attention must be paid when using these features.
12. If an interrupt occurs during drag playback, playback will not resume after the interrupt.

8.15.3Debugging example

Operation	Graphical Representation	Explanation
1. After editing the RL program, use pptomain to move the pointer to the main function, and check the program for syntax errors.		To facilitate single-step debugging, pause command can be inserted into the program.
2. Switch the mode to "manual mode", power on the robot, and adjust the program speed to a small value to make the robot run at a low speed.		
3. Click the "Start" button on the right operation panel.		
4. When the program runs to pause, it will pause until the "Next" button or the "Start" button is clicked.		The "Next" button will single-step the program
5. Click the "Next" button to start single-step debugging.		Run the program to the SetDO line, turn on status monitoring, and check if the DO is in the "off" state
6. Continue to carry out single step to see if the robot has reached the desired point position.		
7. Or move the cursor to a certain line, and click the "Next" button to check if the command runs successfully.		Run the program to the SetDO line, turn on status monitoring, and check if the DO is in the "on" state

8. After confirming that there are no issues with the program, execute pptomain again and keep the "program speed" low. After running it completely once, the program speed can be increased, and the program can be switched to "automatic mode" to execute.		

9Setting

9.1Introduction to this chapter

This chapter provides a detailed introduction to various settings of the xCore control system.

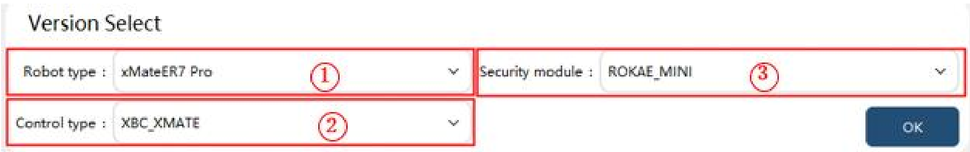
9.2Controller settings

9.2.1Basic settings

9.2.1.1System information

①	Version: control system version information;
②	Robot type: robot model information;
③	MAC address: local MAC address;
④	Restart: Note that all configuration information should be saved before restarting; Shut down: The controller software can only be restarted after the control cabinet is powered off and then powered on again.

9.2.1.2System configuration

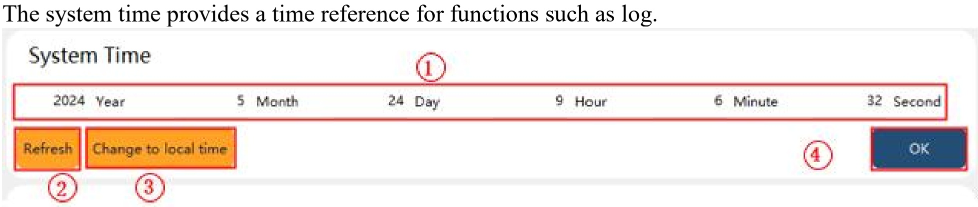


①	Robot type: Select the robot model;
②	Control cab type: Select the control cab type;
③	Safeboard: Select the type of safeboard;

Note: Please do not modify the system configuration. In special circumstances, adjustments should be made under the guidance of the manufacturer.

9.2.1.3System time

The system time provides a time reference for functions such as log.

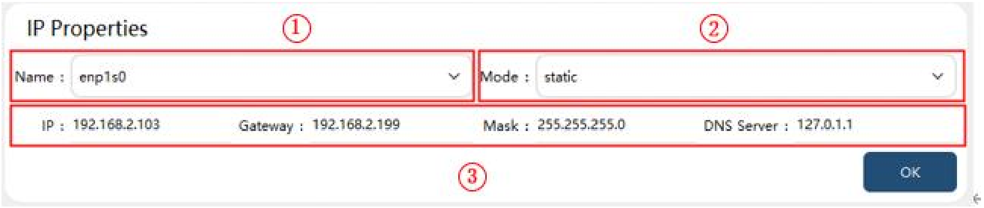


①	Time value: controller time; //not refresh in real time
②	Obtain controller time: After clicking, refresh to display the controller time, and the user can confirm whether the value is reasonable;
③	Set to current time: Update the controller system time to the current system time of the device where the RobotAssist software is located, without the need to click "OK" again.
④	The user can manually modify the time in ①, and set the controller time after clicking "OK";

> Warning
>
> 1. The system time is the absolute time standard for log information. Do not modify it arbitrarily. Wrong system time will make it impossible for the user to trace the moment of a relevant event through the log.
> 2. Do not frequently perform the two operations: obtain controller time or set to current time. The interval between two operations (either one or both) should be greater than 5 seconds.

9.2.1.4System IP properties

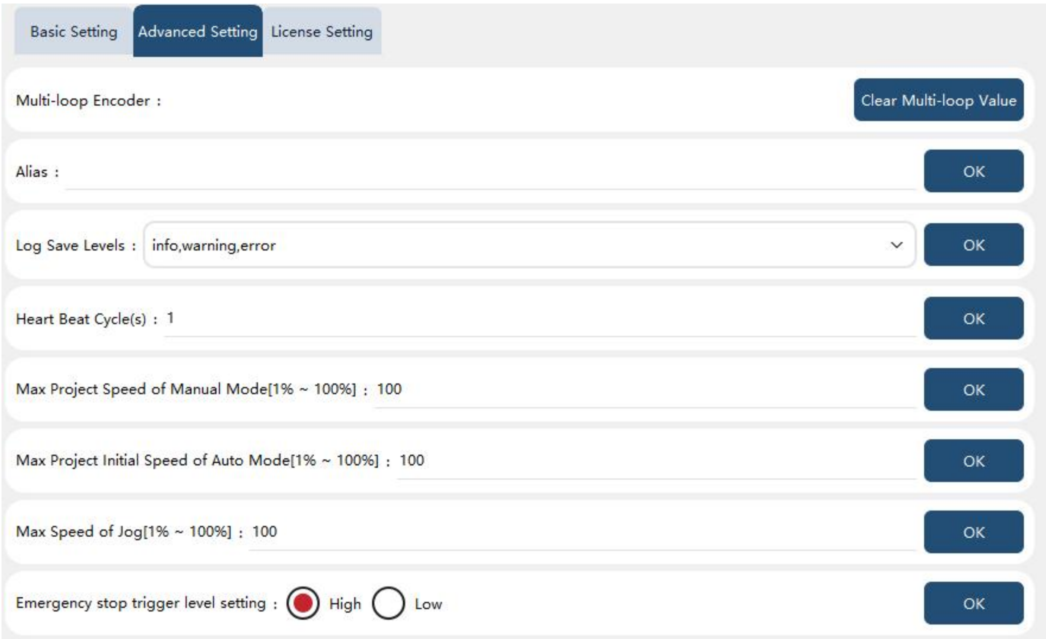
Configure the connection mode, IP, and subnet mask of the robot's external network port on this page.



①	Name: Display the automatically obtained network card name;
②	Mode: manual (IP can be changed) or automatic (IP is automatically assigned);
③	IP, gateway, subnet mask, and DNS server;

Note: The IP address of the debugging network port can only be modified to 192.168.0.160 or 169.254.160.160. When the IP of the debugging network port is 192.168.0.160, the IP of other network ports cannot be modified to the 192.168.0 network segment; when the IP of the debugging network port is 169.254.160.160, the IP of other network ports can be modified to the 192.168.0 network segment but not to 192.168.0.160.

9.2.2Advanced settings

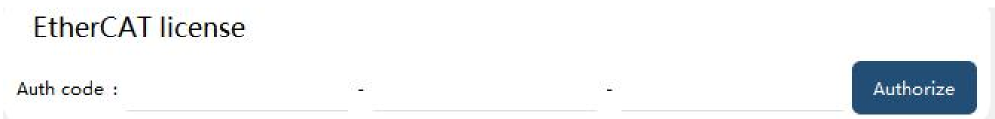


Multi loop encoding	Clear the multi loop value of encoder. Be careful!
Alias	Set an alias for each controller, so that the robots in the same LAN can identify the controllers conveniently. The alias will be displayed on the "Options" -> "Connections" -> "Robot Detection" interface.
Log save level	There are three levels of log: "info", "warning", and "error", ranking from low to high. Set the level from which the log is kept. The log of lower levels will only be displayed online, and will not be kept in the log.
Heartbeat cycle	"One high level + one low level" is a heartbeat cycle, and the duration of high level is the same as that of low level. The heartbeat signal can be bound to the DO signal through the system IO (see the system IO part for details), or bound to the register through the "sta_heartbeat" (see the register part for details).
Deviation path sphere radius	Unit: mm. Range: [0, 9999] Definition: Given a value x, when the robot pauses at point A, a sphere is formed with point A as the center and x as the radius. If the Euclidean distance between the robot's current position and point A is less than x, the deviation path sphere radius condition is satisfied. This parameter is used in conjunction with the function code "sta_near_path". For details, refer to the register section.
Deviation path	Unit: degrees (°).

orientation angle	Range: [0°, 180°] Definition: Given a value a°, when the robot pauses at point B, using B's orientation as the reference, the current robot position's coordinate system is compared with B's coordinate system. If the angular deviation between these two coordinate systems is less than a°, then the path deviation sphere angle condition is considered satisfied. This parameter is used in conjunction with the function code "sta_near_path". For details, refer to the register section.
Upper limit of program speed in manual mode	This parameter limits the upper limit that the program speed can be set when adjusting the program speed through the program speed slider and the "-/+" fine-tuning button in manual mode. Example: After setting this value to 20%, if it is currently in automatic mode and the program speed is 40%, the working mode will be switched from automatic mode to manual mode, and the program speed will be automatically set to 20%. When it is in automatic mode, the program speed is not affected by this parameter. Note: 1. When setting the program speed through the register function code (ctrl _ set _ program _ speed) or SDK, if the robot is in manual mode, the actual program speed will be limited by this parameter; if the robot is in automatic mode, the actual program speed is the set value of the register function code or SDK. 2. When the automatic mode is switched to the manual mode (the switching method includes: the mode switching button on the HMI, the register function code, the system IO, the external communication, and the SDK), the program speed will be limited by this parameter.
Automatic mode program initial speed upper limit	Through this parameter, the program can run in automatic mode at a relatively low speed (the value set by this parameter), enhancing the safety of the operation. After running, the user can adjust the program speed according to the actual situation. Example: The parameter is set to 5%. If the current program speed is 60%, when the program is run in automatic mode, the program speed will be automatically adjusted to 5%; if the current program speed is 3%, the program speed will remain at 3% when running the program in automatic mode.
Default program running speed after startup	When this option is enabled, the controller will set the program running speed to the configured default value upon reboot, ensuring a deterministic program running speed after each controller restart. This setting cannot exceed the manual mode speed limit or automatic mode speed limit. If the configured value exceeds these above limits, it will automatically be reduced to the corresponding limit value.
Jog speed limit	Robot Jog speed cannot be adjusted to a value greater than this upper limit.
Emergency stop trigger level type setting	Through this setting, it is determined whether the register, system output, and external communication output\state bits related to the emergency stop state are "high" or "low" after the emergency stop is triggered.

9.2.3Authorization settings

9.2.3.1EtherCAT authorization



An authorization code is used to authorize the EtherCAT communication.

> **Note**
> The robot cannot be powered on if the authorization code expires or the authorization fails.

9.2.3.2Function authorization

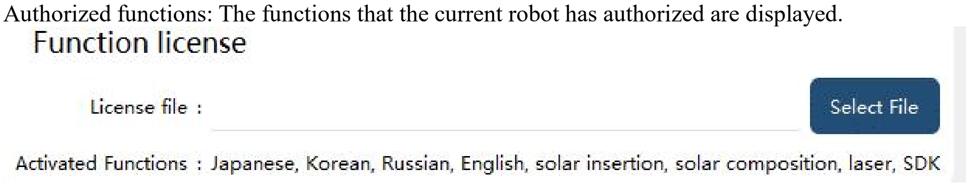
The xCore system supports some optional software functions, which are divided into two types:

1. Languages: Japanese, Korean, Russian, and English;
2. Process packages: PV inserting process package, PV typesetting process package, laser welding process package, and SDK secondary development interface;

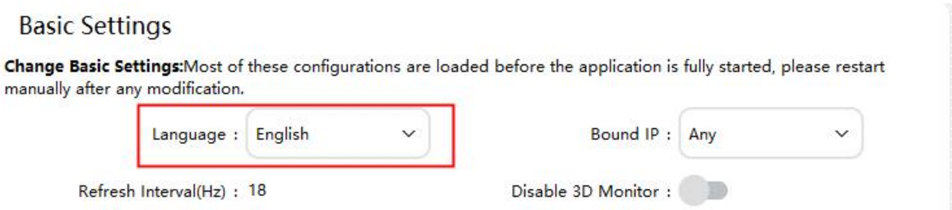
To enable optional functions, you can purchase an authorization code file based on the list of function to be enabled.

Note: The authorization information is bound to the controller of the robot, one robot, one file.

Activation method: In "Settings"−"Controller settings"−"Authorization settings"−"Function authorization", click "Select file", and select authorization file for authorization. If the authorization is successful, restart RobotAssist according to the prompts to enable the optional functions.
Authorization validity: At present, the function authorization is permanently valid, and the controller version upgrade, configuration deletion, factory reset, etc. do not affect the authorization state.
Authorized functions: The functions that the current robot has authorized are displayed.

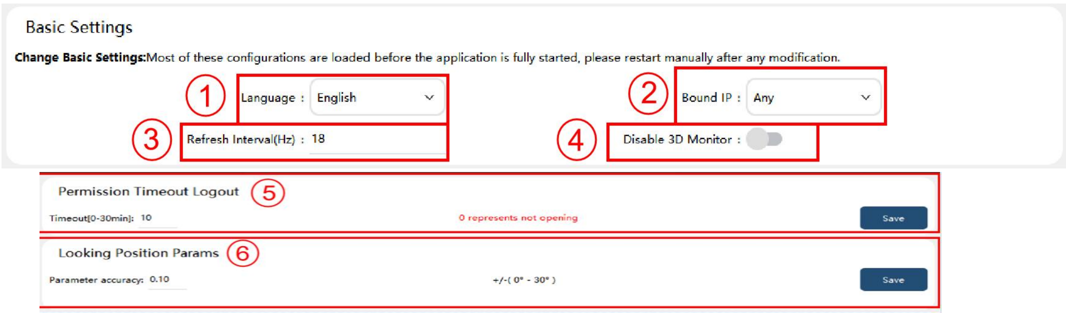


Associated content: In "HMI Settings"−"Basic Settings"−"Language", only authorized languages are displayed.

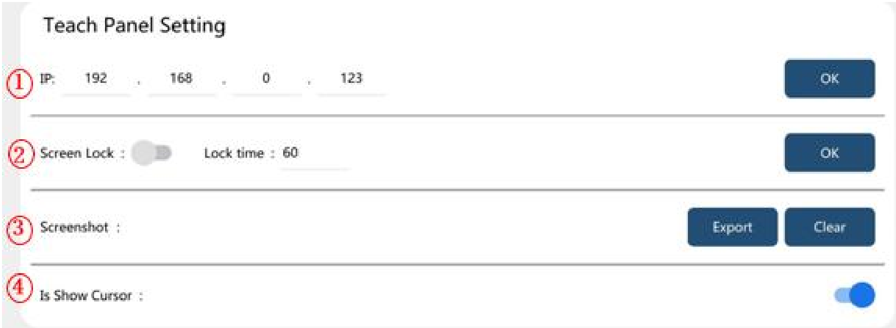


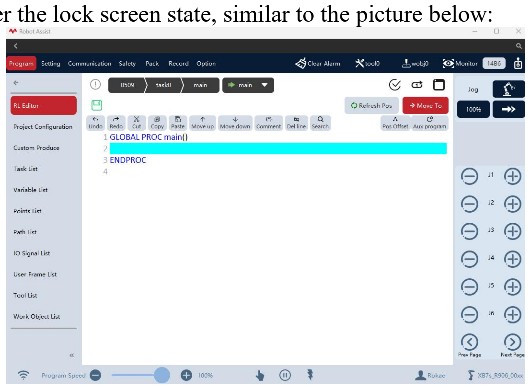
9.3HMI settings

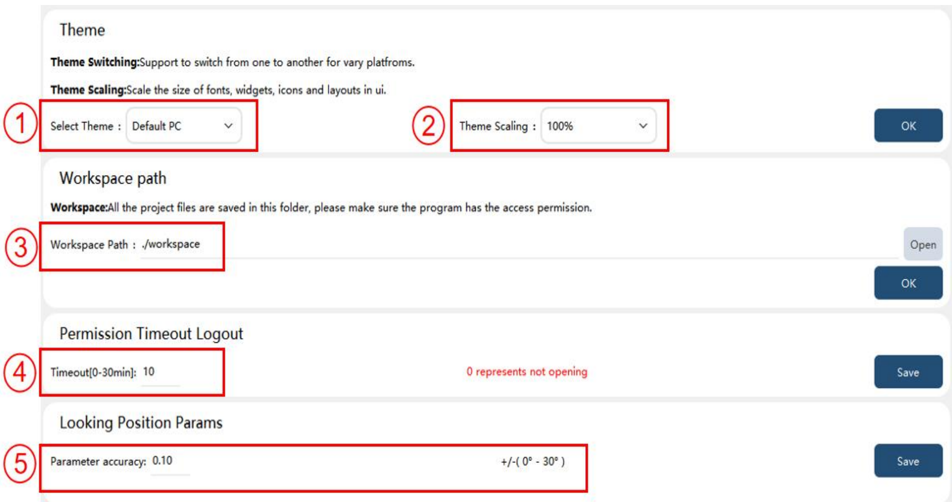
9.3.1Basic settings



①	Language: Chinese, English, Japanese, Korean, and Russian are supported. Note: 1. Chinese is the default language, and other languages need to be officially authorized by ROKAE for use; 2. For xCore V1.7 onwards, the multilingual log is supported (only for "controller logs"). When the HMI switches the language type, the controller will also switch with the language content of the HMI. When the HMI and controller language settings are different, the controller will switch to the language mode corresponding to the HMI when the connection is established. The controller does not need to restart after switching the language. This mode is a hot switching, but some detailed log information only takes effect after switching, and the logs generated before switching the language may not be able to switch to the corresponding language.
②	Bind IP address: The robot detection function of "Options − Connections" can set whether to perform detection through a fixed PC-side IP here.
③	Refresh interval: the refresh rate of the 3D display page of status monitoring, with the upper limit is 100 Hz, and the default is 18 Hz.
④	Turn off the 3D display: Turn off the 3D display for status monitoring, and restart takes effect.
⑤	Timeout logout for permissions: After a certain period of inactivity, the logged-in administrator will automatically log out. The default set timeout is 10 minutes; setting this value to 0 disables this feature.
⑥	View position parameters: In the RL programming interface, this function allows viewing the point position accuracy parameter. It supports fuzzy queries within an angular range, with a default tolerance of 1°.



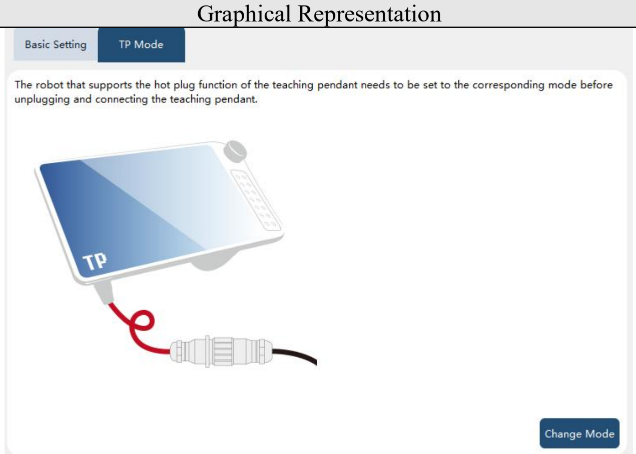
①	IP (only for xPad2): Set the static IP address for the Teach Pendant connected to the robot.
②	Lock screen (only for xPad2): Reduce the accidental touch of Teach Pendant by unrelated personnel. After turning on this function, if the Teach Pendant is not operated for a period of time (idle time), or after clicking the lock screen shortcut key, the Teach Pendant will enter the lock screen state, similar to the picture below:  Unlock the screen by long pressing the circular button on the lock screen for 3 seconds, or short pressing the lock screen shortcut key again.
③	Screenshot (only for xPad2): Take a screenshot of the Teach Pendant screen and save it in the Teach Pendant directory. The picture format is JPG.
④	Cursor displayed or not: Whether the Teach Pendant cursor is set.



①	Choose a theme: Choose a different display style;
②	Theme size: Adjust the size of controls and fonts;
③	Workspace directory: Set the folder to save project files.
④	Timeout logout for permissions: When there is no activity on the interface for longer than the set timeout period, the user permission will be switched to Operator
⑤	Point position accuracy: Set the point position accuracy

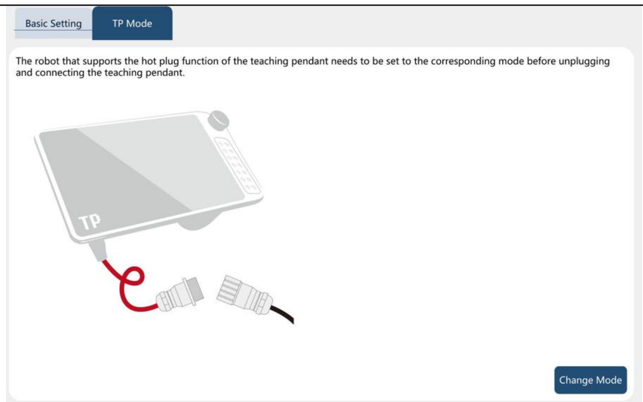
9.3.2Teach Pendant mode

When the robot is turned on and used, it may be necessary to disconnect the physical connection between the xPad2 Teach Pendant and the robot. If the physical connection is directly disconnected, the robot will enter an emergency stop state. If you want not to affect the normal operation of the robot, you can follow the following steps.

Step	Graphical Representation	Explanation
1. View the current mode of the Teach Pendant on this page, the default is "Teach Pendant mode". Click the "Switch mode" button at this time, and try to switch the mode to "No Teach Pendant mode". After the switching is successful, the interface displays the mode as "No Teach Pendant mode".		● In the Teach Pendant mode, the Teach Pendant image is colored, and the connector is in a connected state; ● In the No Teach Pendant mode, the Teach Pendant image is gray, and the connector is in a disconnected state;
2. After switching to "No Teach Pendant mode", the physical connection of the demonstrator xPad2 to the robot can be disconnected. (Refer to the section "Robotic system composition and connections".)		At this time, the robot will not enter an emergency stop state.

To reconnect the Teach Pendant xPad2 to the robot, follow the steps below:

Step	Graphical Representation	Explanation
1. Establish a physical connection between the Teach Pendant xPad2 and the robot. (Refer to the section "Robotic system composition and connections".)		

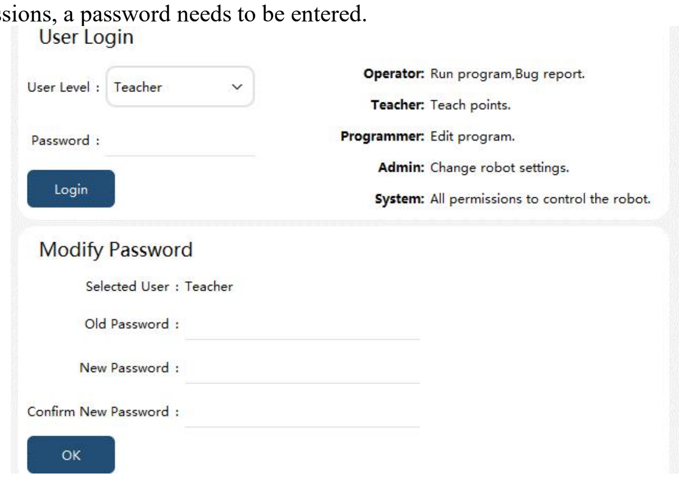
2. View the current mode of the Teach Pendant on this page, and the state should be "No Teach Pendant mode". Click the "Switch mode" button, and try to switch the mode to "Teach Pendant mode". After the switching is successful, the interface displays the mode as "Teach Pendant mode".		● In the Teach Pendant mode, the Teach Pendant image is colored, and the connector is in a connected state; ● In the No Teach Pendant mode, the Teach Pendant image is gray, and the connector is in a disconnected state;

Note

1. The plug & play Teach Pendant function is only available for some models. For models that do not support this function, the system will prompt "Failed to switch Teach Pendant Mode". For detailed model configurations, please consult our technical support.
2. The Teach Pendant mode is only recommended to be turned on when there is a need to remove the Teach Pendant. It is not recommended to turn on it at will during normal debugging and use.

9.4User group

The xCore system is equipped with five levels of built-in users, which are Operator, Teacher, Programmer, Admin, and System based on their operating permissions.
After connecting to the controller, it defaults to logging in with operator permission. When switching to other permissions, a password needs to be entered.



User	Default Password	Permission Description
Operator	None	Run programs and report bugs
Teacher	123	Teach point positions
Programmer	1234	Edit program
Admin	12345	Change robot settings
System	123456	All permissions to control the robot

Note:

- A user of a higher permission level can modify the password of a same- or lower-level user.
- Operator-level user passwords cannot be modified.
- Switching from a high-level user to a low-level user does not require entering a password.

Please refer to the appendix for details of the permissions of each user group.

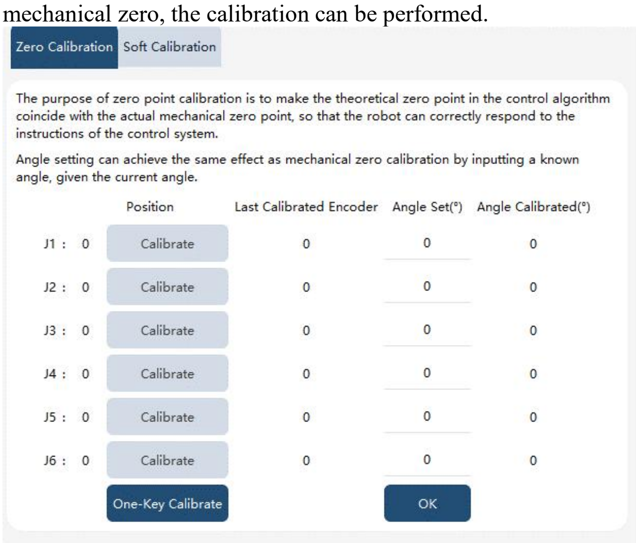
9.5Calibration

The xCore system provides robot calibration functions, including mechanical zero calibration, soft calibration (industrial robot), force sensor zero calibration (collaborative robot), and base frame calibration. The calibration function can perform one-key calibration or single-axis calibration.

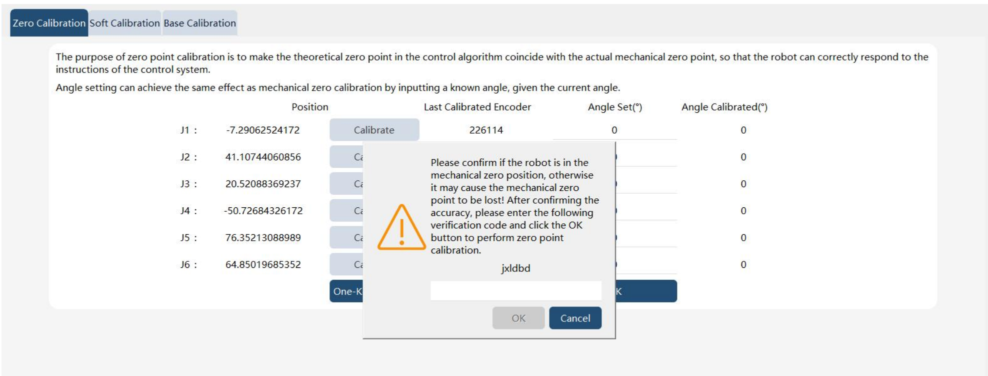
9.5.1Zero calibration

The zero calibration here refers to the mechanical zero calibration, which aims to make the theoretical zero of the robot coincide with the actual mechanical zero.
The zero scale is preset on the robot body, and the joints are aligned, that is, after returning to the

mechanical zero, the calibration can be performed.



To prevent users from losing the zero due to accidental operation during zero calibration, after clicking the "Calibrate" button for each axis or the "One-Click Calibration" button, a verification code must be entered and "Confirm" clicked to make the calibration operation take effect.



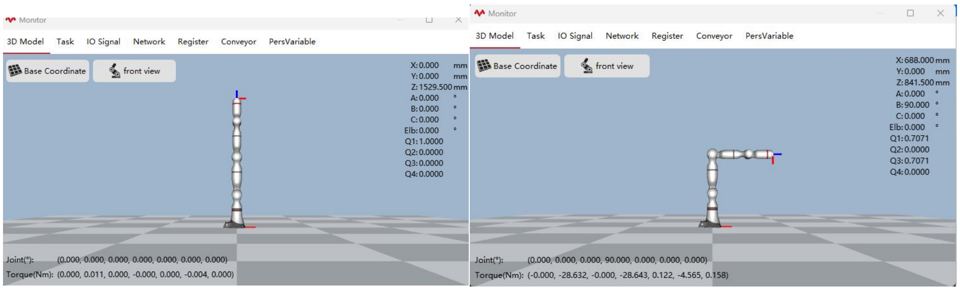
Warning

Please do not calibrate the mechanical zero arbitrarily, and ensure that all robot joints are at the zero point using the mechanical zero calibration block before calibration.
Do not perform the mechanical zero calibration on the robot after it is calibrated by a laser tracker. Otherwise, the zero calibrated by the laser tracker will be lost, therefore affecting the robot accuracy. In case the zero of the robot is lost, please contact ROKAE to restore the zero.

In some space-constrained scenarios, the robot may not be able to return to the mechanical zero, so the "Angle Calibration" function can be used at this time. The prerequisite for using this function is to know the joint angle of the robot at the calibration time, input it into the "Angle Calibration", and then calibrate it, which can achieve the same effect as calibrating at the zero position.

Example:

Taking the xMate7 Pro robot as an example, assuming there is an obstacle above the 4-axis space, the robot cannot reach the vertical state of the mechanical zero and needs to perform zero point calibration. The robot can be adjusted to the right angle state shown in the following figure through jogging. At this point, the 4-axis is 90 degrees. Then, in the "Angle Calibration", enter the current corresponding angle information (the 4-axis: 90 degrees, and the rest: 0 degrees) to proceed.



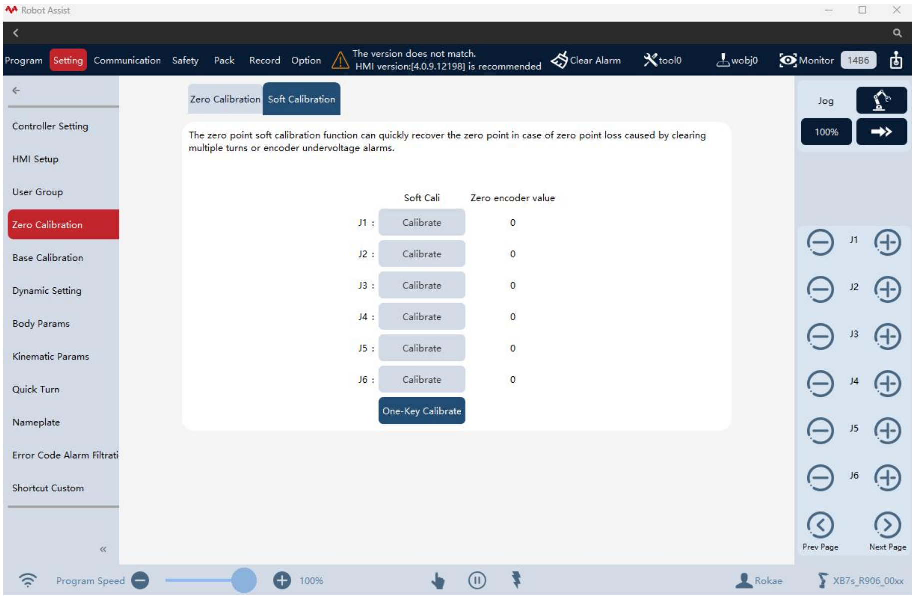
Please note that in the above example, although it is calibrated in a different orientation, the zero of the robot remains in a vertical state. Therefore, if you directly use the Quick Turn to Zero function after a successful angle calibration by inputting the current angle of the 4-axis at 90 degrees, the robot will still move to the vertical state of the mechanical zero and thus collide with the obstacles! So bear in mind that the Angle Calibration function calibrates the zero. It does not mean that the zero is at the current angle.

9.5.2Soft calibration

The Soft Calibration function refers to the function of the robot to quickly recover the zero after the zero is lost due to abnormal operations such as encoder battery undervoltage, disassembly of the battery, or accidental touch and removal of multiple loops.

Before using this function, it is necessary to manually jog the robot to the zero (the wide and narrow calibration slots are aligned, and the narrow slots are completely located in the wide slots), and this function cannot restore the zero at any angle.

The Soft Calibration function is for industrial robots and collaborative robots.



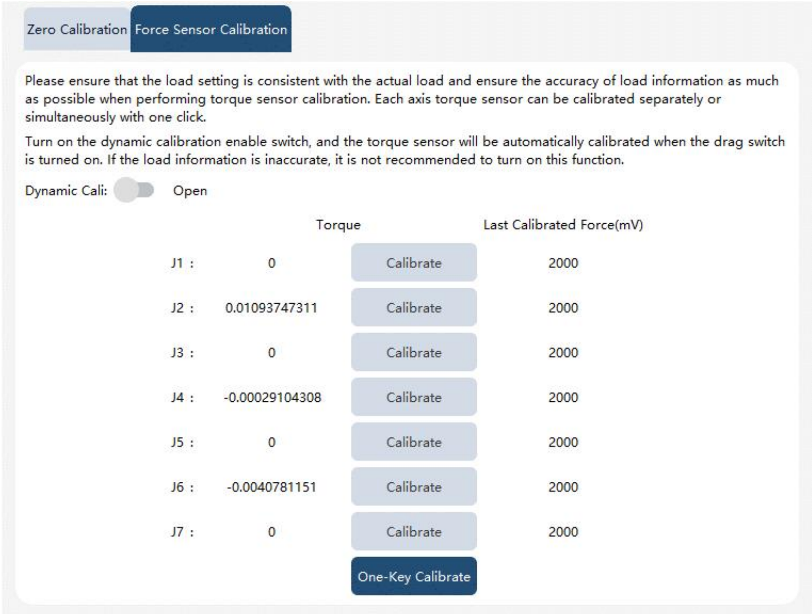
As shown in the figure above, the soft calibration operation steps are as follows:

1. Manually jog the robot to zero (the wide and narrow calibration slots are aligned, and the narrow slots are completely located in the wide slots);
2. Enter the "Main Menu – Settings – Zero Calibration– Soft Calibration" interface;
3. Click the "one-key calibration" button and confirm the pop-up prompt to retrieve the zero, and the zero encoder value displays the encoder value of the zero of each axis;
4. Click the "Calibration" button corresponding to each axis, and confirm the pop-up prompt to retrieve the single-axis zero.

Attention: If the zero point is calibrated through angle calibration, it is necessary to first jog to the vicinity of the angle calibration value, and then perform soft calibration to retrieve the zero point.

9.5.3Force sensor calibration

The calibration is aimed at the xMate series of collaborative robots. During the long-term use of the robot, the torque sensor may inevitably produce zero drift, which is manifested as the robot dragging and drifting.



After a similar problem occurs, the robot can be adjusted to the zero position to perform single-axis calibration or one-key calibration.

If you want to calibrate the force sensor at any position, you can turn on the "Dynamic calibration" switch, and then perform single-axis calibration or one-key calibration. The calibration accuracy of this method may not be as good as that of performing force sensor calibration at the zero position of the robot. In addition, after the dynamic calibration function is turned on, when the force control is turned on in the drag and RL program, the system will automatically zero to ensure the normal use of force control related functions.

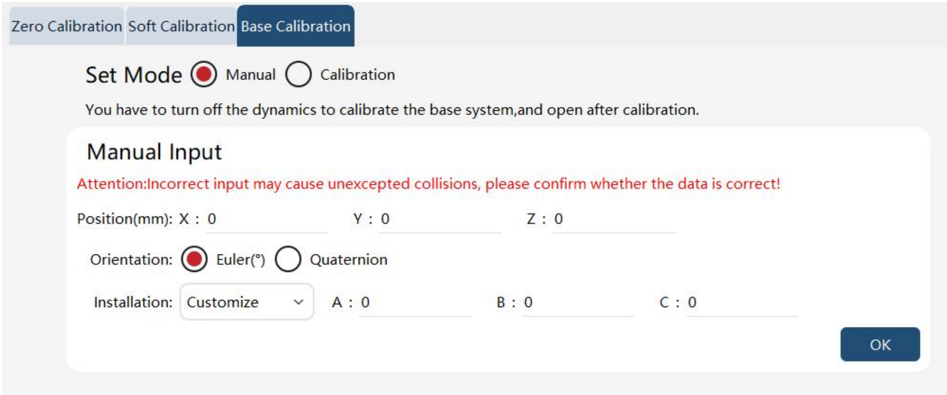
> Warning
>
> Dynamic calibration involves two risks:
> 1. If the robot is in contact with the environment during dragging, i.e., the robot is in a non-free state, the calibrated zero may have a big error, which may result in the wrong torque calculated and failure to enable force control;
> 2. When the drag is turned on or the RL force control is enabled, you may encounter a dynamic calibration error, indicating that the current sensor torque deviates greatly from the theoretical model. You can check the following points: 1) whether the load is set correctly; 2) whether the base frame is set correctly; 3) whether the mechanical zero has a large offset; and 4) whether there is direct force contact with the external environment when the function is turned on.
>
> For these reasons, dynamic calibration should not be turned on unless the torque sensor zero sees serious drifting. This function switch defaults to the off state.

9.6Calibration of the base frame

The base frame is located at the center of the robot base, described relative to the world frame, and defines the robot's relative pose to the world frame. In scenarios such as robot inversion, wall installation, and multi-robot collaboration, it is usually necessary to first calibrate the base frame; otherwise, abnormalities such as robot overload, shaking, and force control failure may occur.

There are two methods for calibrating the base frame: manual input and six-point calibration.

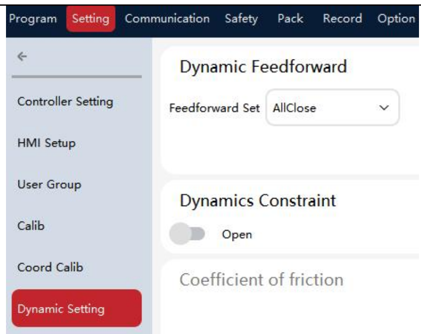
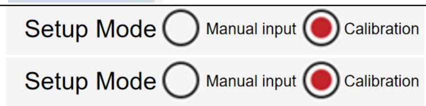
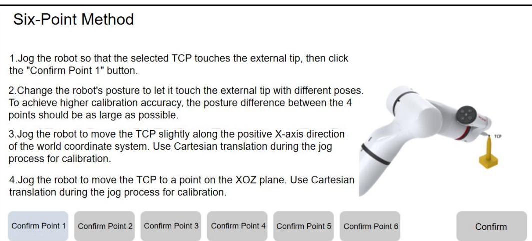
9.6.1Manual input



Manually input the position and orientation of the base coordinate relative to the world frame, and the orientation can be selected from Euler angles or Quaternions.

If you choose the two preset installation methods (front and back), the orientation value can be automatically set; if you choose the "Custom" method, it can be manually entered.

9.6.2Six-point calibration

Step	Graphical Representation	Explanation
1. Calibrate a tool;		
2. If it is not front installation, turn off the dynamics feedforward first (Settings -> Dynamic settings – Dynamic feedforward);		
3. Select "Calibration" for the calibration mode of the base frame;		
4. Define the position of auxiliary positions, which is optional;		When the robot's tool cannot reach the world coordinate system due to excessive distance, the base frame can be calibrated using auxiliary points. The auxiliary position is defined according to the world frame.
5. Follow the HMI steps to complete the settings for the six points, and click "OK".		
6. Choose whether to save the base frame data according to the calibration result.		

9.7Frame calibration

9.7.1Global tool list

9.7.1.1Overview

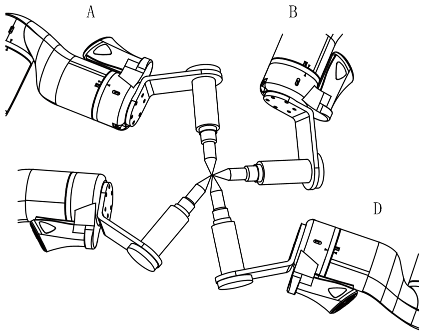
Global tool definition is consistent with tool definition. In the xCore control system, the data type corresponding to global tools is "g_tool_num", with the num ranging from 0 to 15, totaling 16 global tools. Global tools also belong to the tool type. For detailed descriptions of tools, please refer to the "RL commands - Variables" section. Note that global tools differ from tools within individual projects; global tools can be used across all projects.

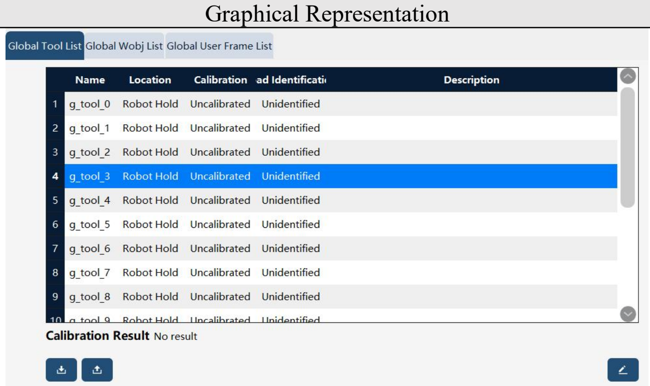
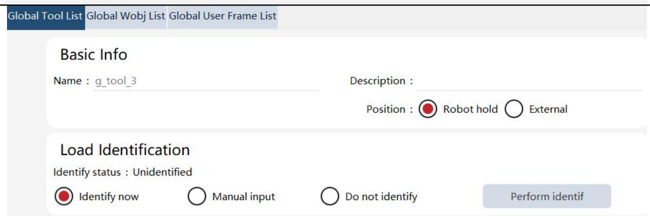
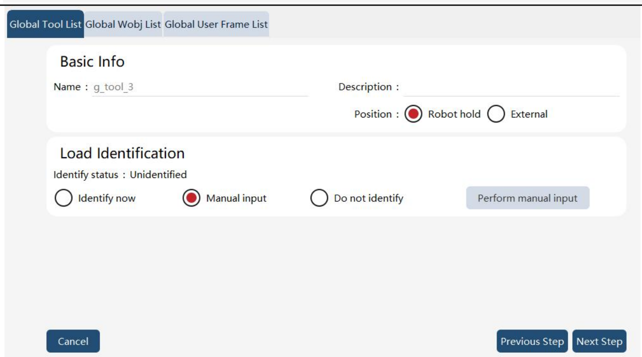
9.7.1.2Basic concept

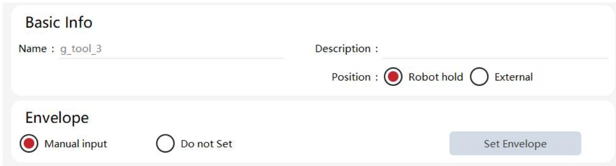
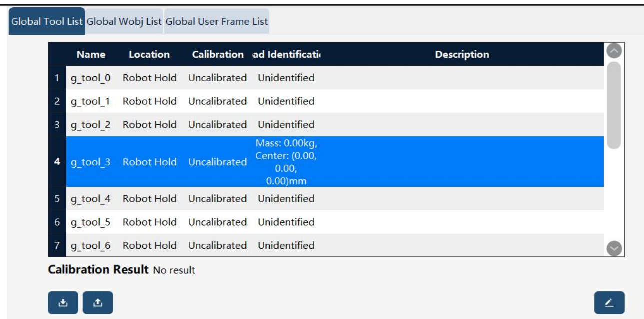
Refer to the basic concepts in 8.12.2 for details.

9.7.1.3Operation example - creating a global handheld tool

Before the calibration of tool frame, the user needs to prepare a fixed external point, which shall be located within the robot's working range and can be contacted by the calibrated tool in a very flexible orientation.



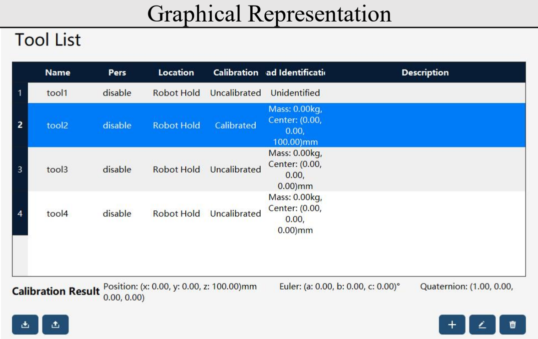
Operation	Graphical Representation	Explanation
1. In the Coordinate System Calibration - Global Tool List interface, select the global tool you need to edit.		When logged in with "Programmer" or higher permissions, editing operations can be performed on the selected global tool. However, only editing is allowed; new tools cannot be added, nor can existing tools be deleted.
2. Set the tool attribute: Robot hold.		
3. Load identification (taking manual input as an example): Select "Manual Input," click the "Perform manual input" button, and enter the "Tool Load Identification" page.		Input the physical information of the tool, click the "Next Step" button, and return to the "New Tool" page. At this point, the "Identify status" will be displayed as "Identified".

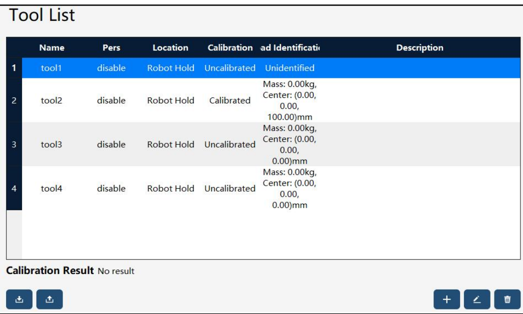
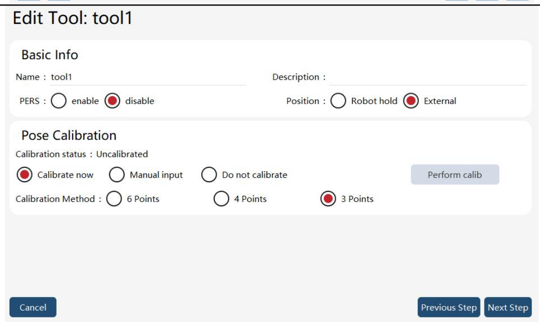
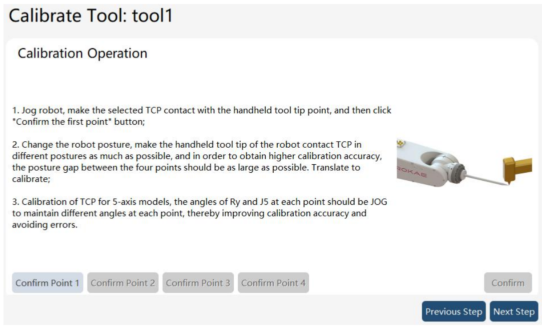
4. Pose calibration, taking the three-point method as an example: Select "Calibrate now" and set the calibration method to "3 Points". Then, click the "Perform calib" button to enter the "New User Frame" page to perform the calibration.		Jog the robot to move the calibrated tool TCP to the points of the desired frame in sequence: the origin, a point on the x-axis, and a point on the xy-plane or on the y-axis, and click the "Confirm Point 1", "Confirm Point 2", and "Confirm Point 3" buttons accordingly. After confirming all three points, click the "Confirm" button to return to the "New Tool" page, where the "Calibration Status" will now be displayed as "Calibrated." When the four-point method is used to calibrate the tool origin, the orientation differences between the four points shall be as large as possible. In other words, the robot shall try to contact the external point in different orientations.
5. Envelope: Select "Manual Input", click "Execute Envelope Editing", and enter the "Envelope Settings" interface to choose the envelope.		
6. Click the "Complete" button to finish editing the global tool.		The newly created tool will be displayed in the global tool list.

9.7.1.4Operation example - creating a global external tool

The calibration methods for global external tools are consistent with those for global handheld tools, supporting three methods: six-point method, four-point method, and three-point method.

Attention: To calibrate the external tool frame, it is necessary to use the already calibrated handheld tool.

Operation	Graphical Representation	Explanation
1. If a calibrated handheld tool already exists in the list, it can be used to assist in the calibration of an external tool. In this example, we will use "tool2" as the reference.		The external tool to be calibrated shall have a tip.

<table>
<tr>
<td>2. Click the "Edit" button in the lower-right corner of the Global Tool List interface to enter the Global Tool wizard interface.</td>
<td>
</td>
<td>Users with the permission level of "Programmer" or above can perform edit operations.</td>
</tr>
<tr>
<td>3. Set the tool attribute: External.</td>
<td>
</td>
<td></td>
</tr>
<tr>
<td>4. Switch between tools in the upper right corner and select the already calibrated handheld tool, "tool2".</td>
<td>
</td>
<td></td>
</tr>
<tr>
<td>5. Pose calibration, taking the four-point method as an example: Select "Calibrate now" and set the calibration method to "4 Points". Then, click the "Perform calib" button to enter the "Tool Calibration" page to perform the calibration.</td>
<td>
</td>
<td>Jog the robot so that the TCP of the calibrated tool can point at the origin of the desired work object frame in different orientations, and then confirm the first, second, third, and fourth points respectively. The orientation difference between the 4 points shall be as large as possible, which means the robot shall try to contact the external point in different orientations.
After the calibration is completed, the system will pop up the calibration error. Select whether to re-calibrate according to the error situation (refer to the confirmed calibration accuracy).</td>
</tr>
</table>

Note:
The external tool must be used together with the corresponding work object, meaning among the Position parameters which are selected at the same time in the tool and work object respectively, one must be External while the other be Robot hold. Otherwise, the system will prompt an error and forbid jogging the robot.
The reference frames for defining tool frames and work frames of external tools differ from those for defining tool frames and work frames of normal tools. You can refer to the following table.

Frame name	Definition of a normal tool relative to …	Definition of an external tool relative to …
Work object frame	User frame	User frame
User frame	World frame	Flange frame
Tool frame	Flange frame	World frame

9.7.2Global work object list

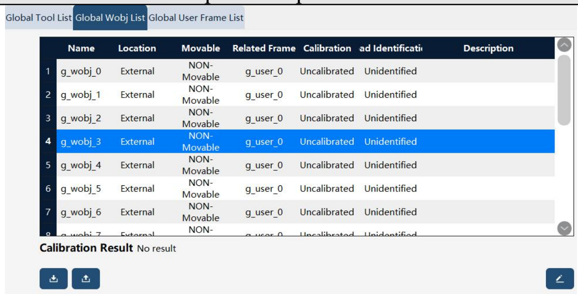
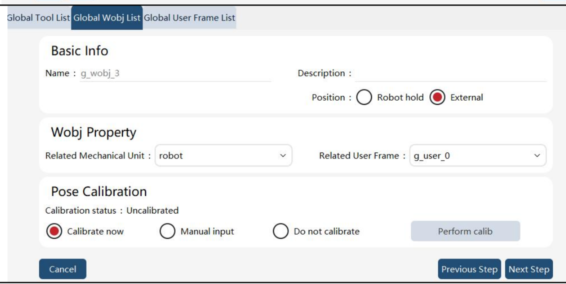
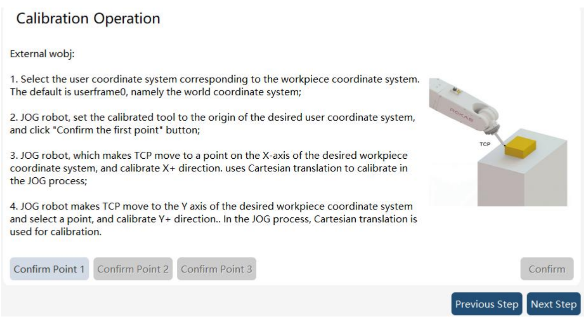
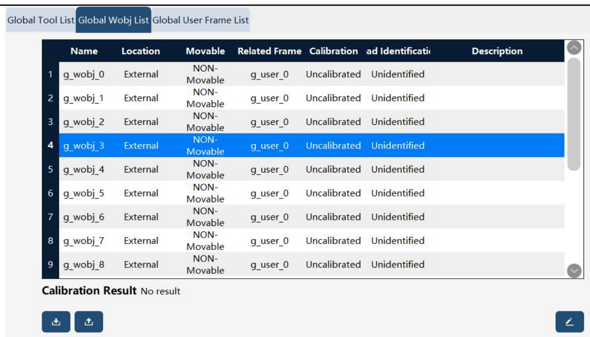
9.7.2.1Overview

Global work object definition is consistent with work object definition; see 8.13.1 for details. In the xCore control system, the data type corresponding to global work objects is "g_wobj_num", with the num ranging from 0 to 15, totaling 16 global work objects. Global work objects also belong to the work object type. For detailed descriptions of work object, please refer to the "RL commands - Variables" section. Note that global work objects differ from work objects within individual projects;

global work objects can be used across all projects.

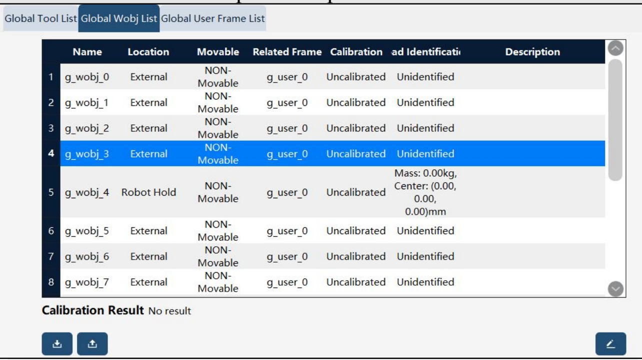
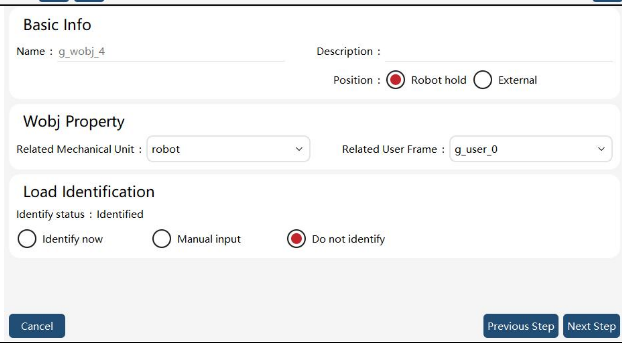
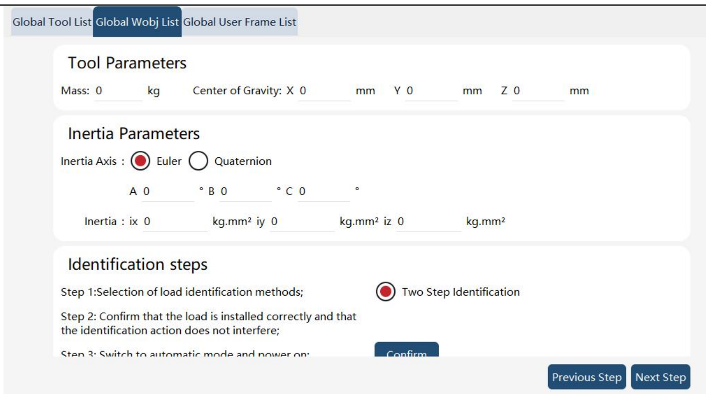
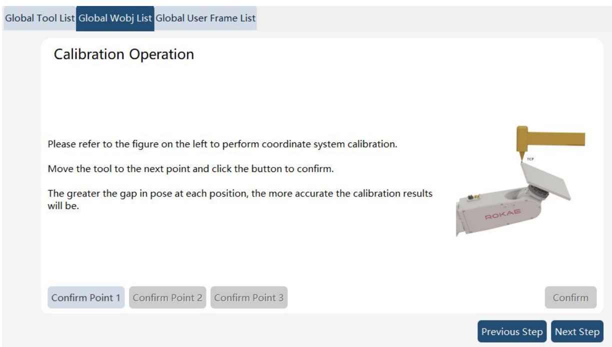
9.7.2.2Operation example - creating a global external work object

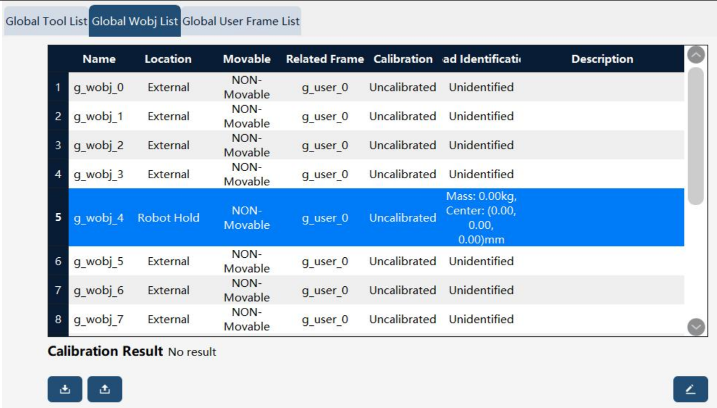
To calibrate the global external work object frame, it is necessary to use an already calibrated handheld tool for assistance.

<table>
<tr><th>Operation</th><th>Graphical Representation</th><th>Explanation</th></tr>
<tr><td>1. Click the "Edit" button in the lower-right corner of the Work Object List interface to enter the Global Work Object Edit wizard interface.</td><td>
</td><td>Users with the permission level of "Programmer" or above can perform edit operations</td></tr>
<tr><td>2. Set the global work object attribute: external.</td><td>
</td><td></td></tr>
<tr><td>3. Switch the tool in the upper right corner to the already calibrated global handheld tool, "tool2"</td><td>
</td><td></td></tr>
<tr><td>4. Click the "Execute Calibration" button to enter the global work object calibration interface.</td><td>
</td><td>Jog the robot to point the calibrated tool TCP to the desired origin of the work object frame, and click the "Confirm Point 1" button. Then, Jog the robot to point the TCP to a point on the desired X-axis of the work object frame, and click the "Confirm Point 2" button. After that, Jog the robot to point the TCP to a point on the desired Y-axis of the work object frame, and click the "Confirm Point 3" button. Finally, click "Confirm Calibration" to bring up a dialog box for confirming the calibration results, and click "Confirm." Then, Click the "Complete" button to return to the "Global Work Object List Editing" page.</td></tr>
<tr><td>5. Click the "Complete" button to finish editing the global work object.</td><td>
</td><td>The successfully edited global work object will be displayed in the Global Work Object List.</td></tr>
</table>

9.7.2.3Operation example - creating a global handheld work object

To calibrate the global handheld work object frame, it is necessary to use an already calibrated global external tool for assistance.

Operation	Graphical Representation	Explanation
1. Click "Edit" in the bottom right corner of the Work Object List interface to enter the New Work Object wizard interface.		Users with the permission level of "Programmer" or above can perform edit operations.
2. Set the work object attribute: handheld		"External" refers to the case where the work object is not fixed relative to the end-effector of the robot, while "handheld" means the case where the work object is fixed relative to the end-effector of the robot.
3. Switch the tool in the upper right corner to the already calibrated external tool, "tool2"		
Click "Execute Load Identification" to enter the Load Identification interface.		
5. Click the "Execute Calibration" button to enter the work object calibration interface.		Jog the robot to point the calibrated tool TCP to the desired origin of the work object frame, and click the "Confirm Point 1" button. Then, Jog the robot to point the TCP to a point on the desired X-axis of the work object frame, and click the "Confirm Point 2" button. After that, Jog the robot to point the TCP to a point on the desired Y-axis/XOY-plane of the work object frame, and click the "Confirm Point 3" button. Finally, click "Confirm Calibration" to bring up a dialog box for confirming the calibration

<table>
<tr><td></td><td></td><td>results, and click "Confirm." Then, Click the "Next" button to return to the "New Work Object" page</td></tr>
<tr><td>5. Click the "Complete" button to finish editing the global work object</td><td>

<table>
<tr><th></th><th>Name</th><th>Location</th><th>Movable</th><th>Related Frame</th><th>Calibration</th><th>ad Identificati</th><th>Description</th></tr>
<tr><td>1</td><td>g_wobj_0</td><td>External</td><td>NON-Movable</td><td>g_user_0</td><td>Uncalibrated</td><td>Unidentified</td><td></td></tr>
<tr><td>2</td><td>g_wobj_1</td><td>External</td><td>NON-Movable</td><td>g_user_0</td><td>Uncalibrated</td><td>Unidentified</td><td></td></tr>
<tr><td>3</td><td>g_wobj_2</td><td>External</td><td>NON-Movable</td><td>g_user_0</td><td>Uncalibrated</td><td>Unidentified</td><td></td></tr>
<tr><td>4</td><td>g_wobj_3</td><td>External</td><td>NON-Movable</td><td>g_user_0</td><td>Uncalibrated</td><td>Unidentified</td><td></td></tr>
<tr><td>5</td><td>g_wobj_4</td><td>Robot Hold</td><td>NON-Movable</td><td>g_user_0</td><td>Uncalibrated</td><td>Mass: 0.00kg, Center: (0.00, 0.00, 0.00)mm</td><td></td></tr>
<tr><td>6</td><td>g_wobj_5</td><td>External</td><td>NON-Movable</td><td>g_user_0</td><td>Uncalibrated</td><td>Unidentified</td><td></td></tr>
<tr><td>7</td><td>g_wobj_6</td><td>External</td><td>NON-Movable</td><td>g_user_0</td><td>Uncalibrated</td><td>Unidentified</td><td></td></tr>
<tr><td>8</td><td>g_wobj_7</td><td>External</td><td>NON-Movable</td><td>g_user_0</td><td>Uncalibrated</td><td>Unidentified</td><td></td></tr>
</table></td><td></td></tr>
</table>

9.7.3Global user frame list

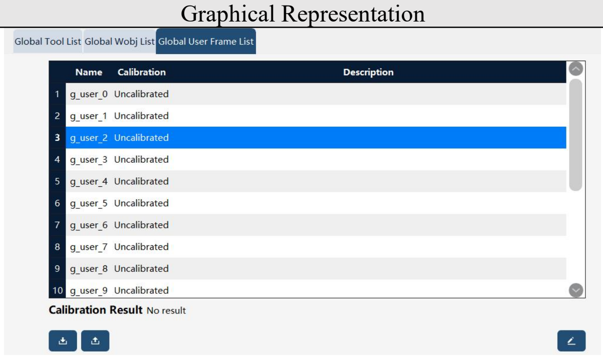
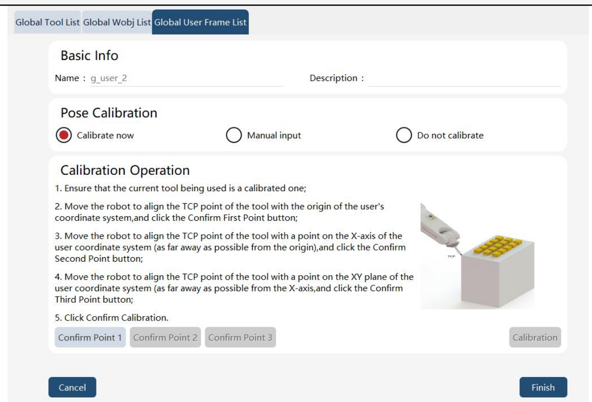
9.7.3.1Overview

The global user frame serves as a reference frame when defining the work object frame or global work object frame and is not used independently.

	Name	Calibration	Description
1	g_user_0	Uncalibrated	
2	g_user_1	Uncalibrated	
3	g_user_2	Uncalibrated	
4	g_user_3	Uncalibrated	
5	g_user_4	Uncalibrated	
6	g_user_5	Uncalibrated	
7	g_user_6	Uncalibrated	
8	g_user_7	Uncalibrated	
9	g_user_8	Uncalibrated	
10	g_user_9	Uncalibrated	

When establishing a global user frame, you can choose "Calibration now", "Manual input" or "Do not calibrate".
When "Calibration now" is selected, the 3-point method is used to calibrate. Before calibrating the global user frame, the user needs to first calibrate a tool and then use the TCP of the tool to calibrate the user frame. For this, it is recommended to use a tool with tips.
"Manual input" is allowed if the global user frame is known in advance. Another option is "Do not calibrate", in which case the global user frame is considered as the world frame by default.

9.7.3.2Operation examples

<table>
<tr><th>Operation</th><th>Graphical Representation</th><th>Explanation</th></tr>
<tr><td>1. Click the "Edit" button in the lower-right corner to edit a global user frame.</td><td>
</td><td>Users with the permission level of "Programmer" or above can perform edit operations.</td></tr>
<tr><td>2. Follow the steps shown in the figure to perform the calibration.</td><td>
</td><td>Jog the robot to guide the calibrated tool TCP to sequentially point to the desired frame's origin, a point on the x-axis, and a point on the xy-plane or on the y-axis. Click the "Confirm Point 1", "Confirm Point 2", and "Confirm Point 3" buttons accordingly.
After successfully confirming all three points, click the "Calibration" button.</td></tr>
</table>

9.8Dynamic settings

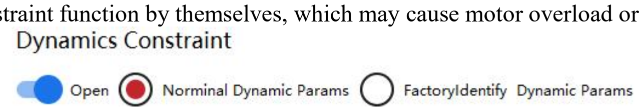
The dynamic settings page is used to set the dynamic parameters of the robot. Dynamic settings are related to functions such as robot force control, drag teaching, virtual walls, and collision detection. Please ensure the robot's dynamic settings are reasonable. Otherwise, the above functions may not work properly or may cause the robot to shake abnormally.

9.8.1Dynamic feedforward

The dynamic feedforward switch determines whether the controller turns on or off the dynamic feedforward function and is turned on by default. Users are not recommended to turn off the dynamic feedforward function by themselves, which may cause jitter when power on and worse trajectory accuracy. The dynamic feedforward should be turned off in certain situations, including base frame calibration when the robot adopts non-front installation and friction identification.

9.8.2Dynamic constraint

The dynamic constraint switch determines whether the controller turns on or off the dynamic constraint function and is turned on by default. Users are not recommended to turn off the dynamic constraint function by themselves, which may cause motor overload or abnormal shaking.



When the dynamic constraint switch is turned on, two options including "Nominal Dynamic Params" and "Factory Identify Dynamic Params" are available. "Nominal Dynamic Params" means nominal parameters will be used in the dynamic control. The same models using the "Nominal Dynamic Params" will deliver the exact same motion velocity and takt time when executing the same motion program, yet the motion performance may be weaker, or there may be motor overload. "Factory Identify Dynamic Params" will allow the robot to be in the best dynamic control status for the shortest takt time allowed, and the motor will be protected from overload. But robots running the same motion program may be slightly different in velocity and takt time.

9.8.3Vibration suppression

The vibration suppression switch determines whether the controller enables the vibration suppression function, which is disabled by default.

When the vibration suppression switch is turned on, the controller compensates for the flexibility in the robot joints, suppressing vibrations caused by joint flexibility to improve position and path accuracy. Changes in the robot's load parameters can affect the amount of joint flexibility compensation, so it is necessary to set accurate load parameters when using the vibration suppression function.

> Note
>
> 1. The vibration suppression setting takes effect immediately without requiring controller reboot.
> 2. The VibSuppression command can be used in RL Project to modify vibration suppression status (see Section 15.4.16.39). .
> 3. Currently, the vibration suppression function is only supported on specific robot models. Refer to the following table for compatible models.

Table of compatible models for vibration suppression:

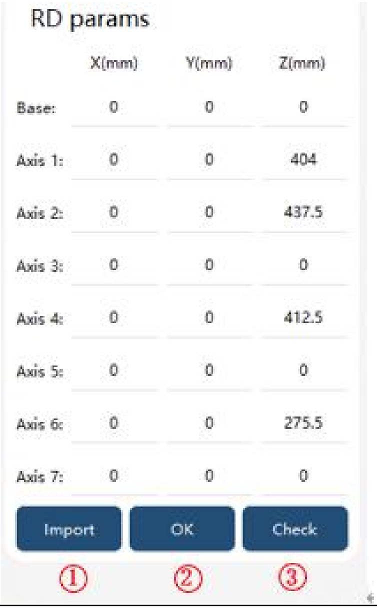
PRODUCT LINE	MAIN MODEL	SUB-MODELS
COLLABORATIVE ROBOT (6-AXIS)	SR3	All sub-models
	SR4	All sub-models
	SR5	All sub-models
	CR7	All sub-models
	CR12	All sub-models
	CR18	All sub-models
	CR20	All sub-models
	CR35	All sub-models
COLLABORATIVE ROBOT (5-AXIS)	CR17	All sub-models
	CR25	All sub-models
INDUSTRIAL ROBOT (6-AXIS)	NB4	NB4-R475-3B
		NB4h-R580-3B
	NB12s	NB12s-1214-5A
		NB12s-1016-5A
		NB12s-1611-5A
	NB12h	NB12h-1214-6A
	NB25s	NB25s-1221-67
		NB25s-2020-67
		NB25s-2518-67
		NB25s-3016-67
		NB25s-3518-67
	NB25h	NB25h-2518-77
		NB25h-1222-77
	NB80s	NB80s-8022-21
		NB80s-5026-21
	XB4h	XB4h-R596-3B
	XB7s	XB7s-R707-3A
		XB7s-R906-0A
		XB7s-R906-3A
	XB10s	XB10s-R1206-3B

9.9Body parameters

The body parameters include RD parameters, DH parameters, reduction ratio, overload coefficient, coupling coefficient, and other robot body related data. The parameters on this page directly affect the accuracy of the motion. Please do not modify them without the permission and assistance of the robot manufacturer.
Note: Starting from Version 3.0, all models except for the PCB 3-axis/4-axis models use DH parameters.

9.9.1RD parameters

RD parameters describes the relative pose relationship between the robot's link frames. They are the foundation for robot kinematics.

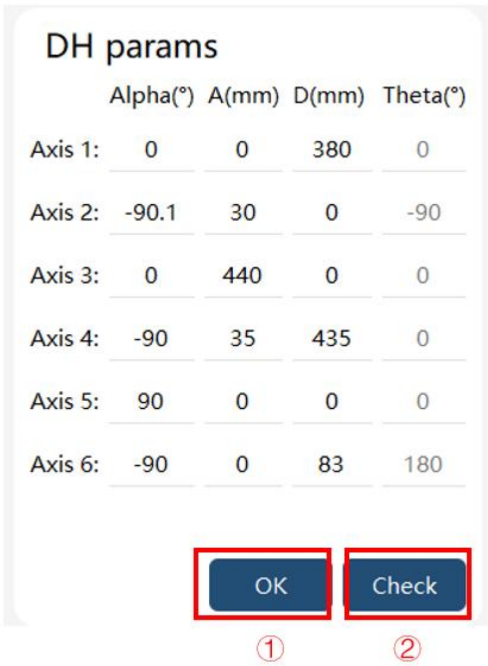


①	Import: Click to select the RD parameter file, which is generally not necessary;
②	OK: After manually modifying the RD parameters, click "OK" to take effect;
③	Check: After clicking, the current RD parameters and system default RD parameters will be displayed;

Note:
There will be a set of default parameters before the robot leaves the factory. Users need to confirm the rationality of RD parameters before modifying or importing parameters. After the RD parameter is modified, the controller needs to be restarted to take effect.

9.9.2DH parameters

DH parameters describe the relative pose relationship between the robot's link frames. They are the foundation for robot kinematics.



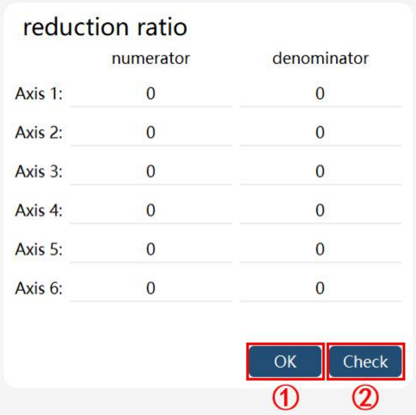
①	Confirm: After manually modifying the DH parameters, click "Confirm" to apply the changes;
②	Check: After clicking, the current DH parameters and system default DH parameters will be displayed;

Note:
There will be a set of default parameters before the robot leaves the factory. Users need to confirm the

rationality of DH parameters before modifying or importing parameters. After the DH parameters are modified, the controller needs to be restarted for the changes to take effect.
The DH parameters are of the improved type, and users can only modify the parameters Alpha, A, and D.

9.9.3Reduction ratio

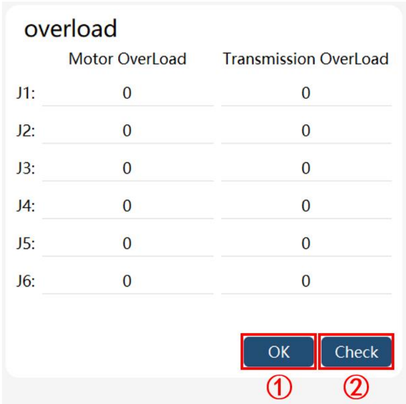
The reduction ratio is the parameter of the reducer in each axis of the robot. Do not modify without permission and assistance from the robot manufacturer.



①	OK: After manually modifying the reduction ratio parameters, click "OK" to take effect;
②	Check: After clicking, the current reduction ratio parameters and system default reduction ratio parameters will be displayed;

9.9.4Overload coefficient

Motor overload coefficient setting.



①	OK: After manually modifying the overload coefficient parameters, click "OK" to take effect; Please modify it carefully with the support of the manufacturer!
②	Check: After clicking, the current overload coefficient parameters and system default overload coefficient parameters will be displayed;

9.9.5Coupling coefficient

In some robots, each axis has a coupling relationship, while the coupling coefficient describes the coupling relationship between these axes. Do not modify without permission and assistance from the robot manufacturer.

9.10Motion parameters

9.10.1Basic motion parameters

Motion parameters include the maximum speed, the maximum acceleration, and the maximum jerk of each axis of the robot, which affect the motion rhythm and ride of the robot. The robot has a set of default parameters before leaving the factory. Modifying the above parameters may cause the robot to shake abnormally, report errors, and affect the service life of the robot. Please modify it carefully.

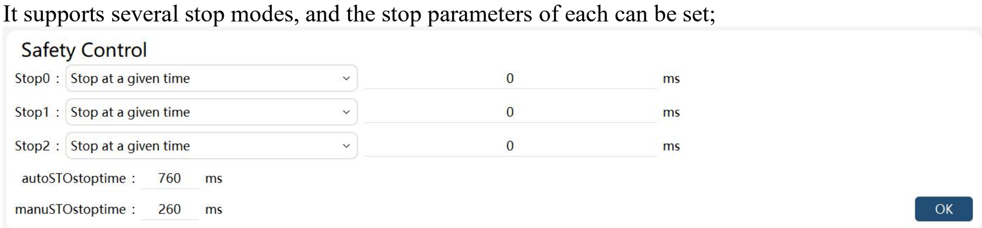
1	Joint max acc: The upper limit of acceleration for each axis of the robot, which is limited by the torque capacity of the motor. When the dynamic constraint is turned off, this parameter takes effect; when the dynamic constraint is turned on, this parameter fails. This value is generally not less than 3−5 times the maximum axis speed.
2	Joint max jerk: The upper limit of the jerk of each axis of the robot. Under normal circumstances, with the increase of jerk, the rhythm of the robot will increase to make it easily shake when moving. If there are many small turning zones in the robot's motion, it is possible to increase the jerk appropriately to increase the rhythm, but attention should be paid to the robot's shake. This value shall not less than 3−5 times the maximum axis acceleration.
3	Joint flexibility coefficient: A coefficient for adjusting the stiffness model of robot joints, with a configurable range of [0.5−1.2]. Modifying this coefficient will affect both gravity compensation and vibration suppression performance. Note: This value generally does not need to be adjusted, and the default value is enough. In special circumstances, please modify it under the guidance of the manufacturer.
4	Jog acc ramptime: The time it takes for the robot's acceleration to increase from minimum to maximum during Jog, which only takes effect in 10 mm step Jog and continuous mode Jog. The smaller the value, the faster the robot Jog starts and stops, but the more prone to shake. If the robot reacts slowly when Jog is felt, the value can be adjusted appropriately.
5	Acceleration rise time: The time it takes for the robot to increase its acceleration from a minimum to a maximum during motion program execution. The smaller the value, the faster the robot accelerates, and vice versa.
6	Deceleration rise time during final stop phase: For target points without a turning zone, this is the time it takes for the robot's deceleration to increase from its minimum value to 0 during the final stop phase. The longer the deceleration rise time, the more gradual and smooth the robot's stopping will be. If the acceleration rise time is greater than the deceleration rise time during the final stop phase, the robot will use the longer acceleration rise time for stopping.
7	VelSmoothCoef: It is used to adjust the speed smoothing process when the robot passes through the turning zone. The larger the value, the smaller the speed deceleration when the robot passes through the turning zone, and the easier it is to shake. When the value is set to 1.0, the velocity is not smoothened when the robot passes through the turning zone. This parameter is mainly used to improve the extreme performance of the robot. When debugging, first confirm the shake of the robot when the value is 1.0: if the robot shakes violently, it indicates that the robot has reached its extreme performance, and this parameter does not need to be increased; if the robot runs smoothly but experiences severe deceleration when passing through turning zones, gradually increasing this

	parameter can improve motion smoothness. When debugging this parameter, it is recommended to gradually increase it in steps of 0.1−0.5.
8	Check: After clicking, the current motion coefficient parameters and system default motion coefficient parameters will be displayed;
9	Maximum TCP linear velocity: Set the vmax parameter for TCP linear speed
10	Maximum TCP rotation velocity: Set the vmax parameter for TCP rotation speed
11	Maximum external axis linear velocity: Set the vmax parameter for external axis linear velocity
12	Maximum external axis rotation velocity: Set the vmax parameter for external axis rotation velocity
13	Position command smoothing parameter: Configure the smoothing level applied by the controller to the robot's joint position commands, with an adjustable parameter range of [1−1,024]. When the value is set to 1, the controller applies no smoothing to the joint position commands. As the value increases, the smoothing level increases, resulting in smoother and slower robot motion. Modifying the position smoothing parameter affects both the robot's cycle time and position accuracy. Note: This value generally does not need to be adjusted, and the default value is enough. In special circumstances, please modify it under the guidance of the manufacturer. Robots using the ARM platform do not support position command smoothing parameter setting.
14	Sampling points: The larger the value, the smoother and faster the robot motion, but it will affect the timeliness of logic and signal processing. Note that this value generally does not need to be adjusted, the default value is enough. In special circumstances, please modify it under the guidance of the manufacturer.

9.10.2Advanced settings

9.10.2.1Safety control

It supports several stop modes, and the stop parameters of each can be set;



Currently, the deceleration to a complete stop at maximum capability is being used. After receiving the stop signal, ensure that the robot path is not offset and at least one motor is planned to stop according to the maximum deceleration capacity.

Considering that the actual performance of different robots can be easily influenced by various factors under different working conditions, this stopping method permits the configuration of a scaling factor, which can be set within the range of [0.1, 1]. When it is set to the maximum value of 1, the robot quickly reaches the motor's maximum capability for deceleration and stopping. When it is set to the minimum value of 0.1, the robot gradually approaches the motor's maximum capability, resulting in a smoother deceleration and stop. The larger the scaling factor, the more quickly the robot reaches the motor's maximum torque value, resulting in a more rapid stop. Conversely, the smaller the scaling factor, the more gradually the robot approaches the motor's maximum torque value, leading to a smoother and more gradual stop.

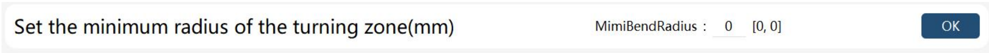
Additionally, the stop coefficient for manual mode (Stop 0) and the stop coefficient for automatic mode (Stop 1) can be set independently.

9.10.2.2Search command max stop distance

When a Search command is used and the stop mode is selected for a quick stop, the distance traveled by the robot TCP from the receipt of the stop signal to the full stop of the robot shall not exceed this value.

9.10.2.3Minimum turning zone radius

The minimum turning zone radius setting defines the shortest permissible turning zone size that can be generated. This parameter can be used to avoid generating a turning zone too short and to make motion smoother. When the control system detects that the length of a trajectory is below the set value of this parameter and the trajectory needs to generate a turning zone, the control system will automatically combine the trajectory and the adjacent trajectories into one trajectory and generate a turning zone with an appropriate length. The larger the value, the longer the minimum turning zone and the smoother the robot passes through the turning zone. When this parameter is set to 0, the control system strictly follows the parameters to generate the turning zone.



9.10.2.4Stacking debug mode

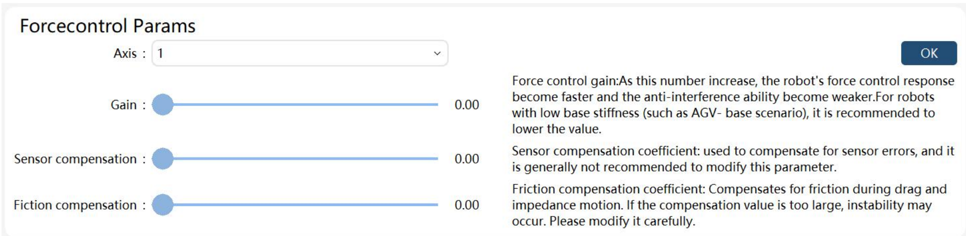
The stacking debug mode option will only be displayed and available when the model type is CR, SR, and Industrial Six-Axis Robot (NB, XB) series models. When this mode is turned on, two Jog frames corresponding to four-axis locking are added: singularity avoidance and parallel base.
Note: CR series 5-axis models turn on this mode by default. At this time, Jog under the base frame corresponds to the singularity avoidance frame of the 6-axis model.

9.10.2.5Default Conf

The "RL" option in Defualt Conf is used to set whether motion commands strictly adhere to the movement constraints defined by the Conf information for each point after reloading an RL project.
Note: After modifying this option, you need to run PPTOMAIN to apply the changes and make the option effective.
The "Point Move to" option in Default Conf is used to set whether motion commands strictly adhere to the movement constraints defined by the Conf information for each point when performing "Point Move to" operations.
Note: By default, all the 4 buttons in the Default Conf are enabled until modifications are made in this interface.

9.11Force control parameters

9.11.1Force control parameters

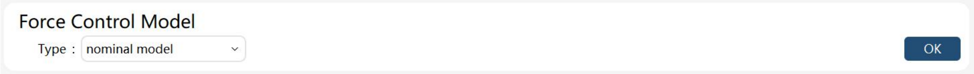


Force control gain: It is used to adjust the response speed of robot force control. Note that the larger the value, the faster the robot force control response, but the weaker the anti-interference ability. The default value is 1.0. If the rigidity of the robot base is low, such as when placing the robot on a mobile chassis, the force control gain value should be appropriately reduced. When the end-effector load is close to the upper limit in the load pattern, the use of dragging and impedance may cause shaking, at this time the force control gain should be appropriately reduced, and the base stiffness should be adjusted to low if the control system version is below V2.0.
Sensor compensation coefficient: It is used to compensate for sensor error, generally which does not need to be modified. If there is a force moving in the positive direction of the joint during the robot's dragging process, this parameter value can be appropriately reduced. On the contrary, if there is a force moving in the negative direction of the joint, this parameter value can be appropriately increased. The default value is 0.5.
Friction compensation coefficient: It is used to compensate for friction during dragging and impedance motion. Too large compensation coefficient may cause instability, so please modify it with caution. The default value is 0.5.

Note:
Force control parameters are for advanced developers, please modify them carefully.

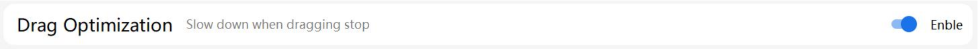
9.11.2Force control model

There are three options for the force control model, and this function is a developer's option, please modify it carefully. The default nominal model is used after the initialization and when no modification is made.



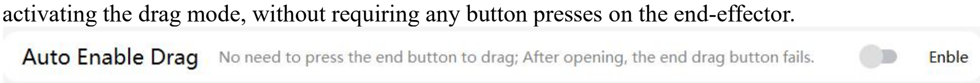
9.11.3Drag optimization

Improve the drag experience by handling the stop at the end of the drag. This function is enabled by default after the initialization and when no modification is made.



9.11.4Drag without end-effector button operations

The collaborative robot supports dragging operations without the need to press the drag button on the end-effector. By enabling the automatic drag mode switch, the robot can be directly dragged after activating the drag mode, without requiring any button presses on the end-effector.



Note:
1. Once the automatic drag mode switch is enabled, ensure safety precautions are followed when activating the drag mode. If the robot exhibits any abnormal behavior, immediately press the emergency stop (E-Stop) button;
2. The end-effector drag button becomes inactive once the automatic drag mode switch is enabled.

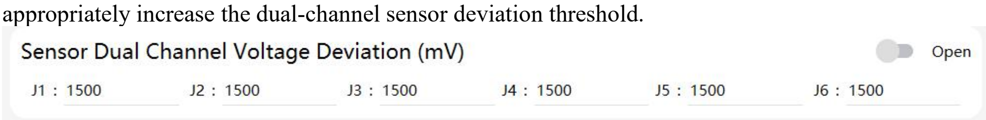
9.11.5Force control model deviation threshold setting

When the drag mode is enabled, the collaborative robot will self-check the deviation between the sensor torque and the theoretical dynamics model torque. If the deviation is significant, enabling the drag mode will fail. If the drag mode still fails to initiate after verifying that the load settings are correct, you can appropriately increase the force control model deviation threshold.

Note:
1、When the force control model deviation threshold setting is not enabled, the controller uses default values

9.11.6Dual-channel sensor deviation threshold setting

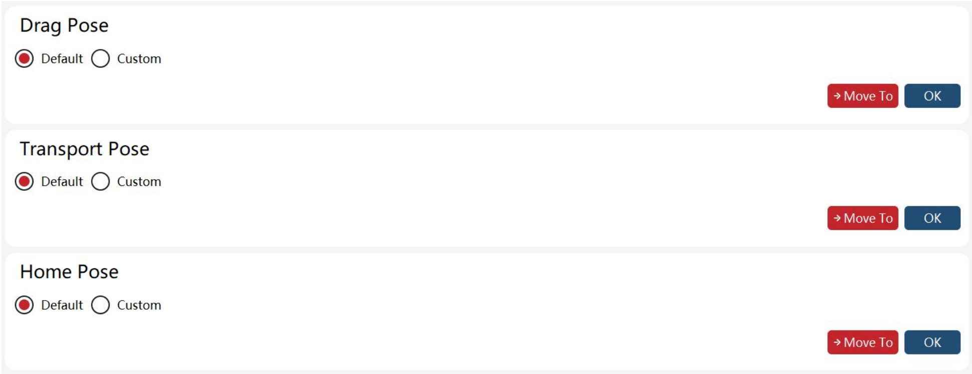
When the drag mode is enabled, the collaborative robot will self-check the deviation between the voltage values of the two sensor channels. If the deviation is significant, the drag mode initiation will fail, and a warning indicating a large voltage deviation will be displayed. In this case, you can appropriately increase the dual-channel sensor deviation threshold.



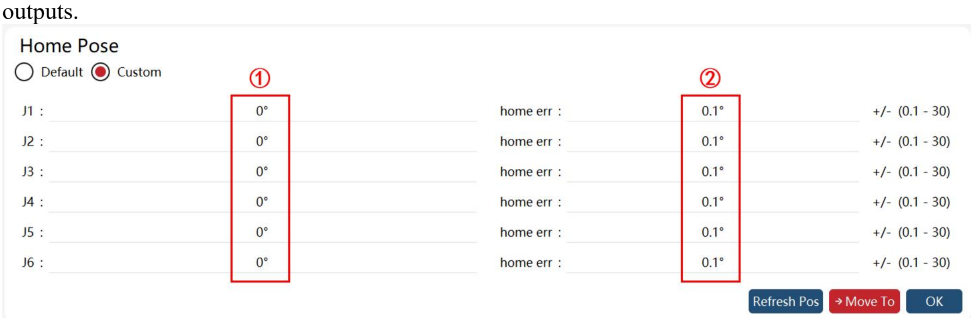
Note:
1. When the dual-channel sensor deviation threshold setting is not enabled, the controller uses default values

9.12Quick adjustment

Users can define multiple commonly used positions. By using buttons or commands, the robot can quickly adjust its position and orientation (pose). It supports custom poses including the drag pose, transport pose, and Home pose. The above poses have default values for specific models, and also support user customization.

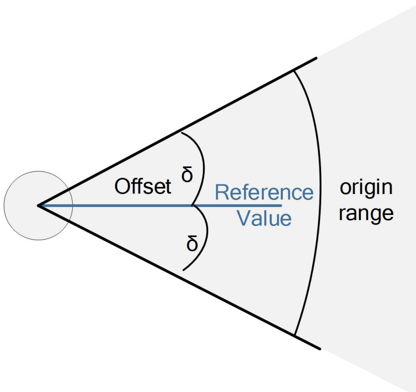


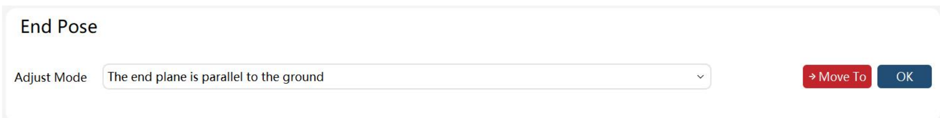
The Home pose is quite unique, in addition to defining the reference position, it also requires setting a set of offsets. When the difference between the robot’s current position and the reference position is less than the specified offset, the robot is considered to be in the Home pose. In this case, the system I/O signal "Home State" and the function code register "sta_home" will generate corresponding outputs.



S/N	Name	Meaning
①	Reference Value	The reference value of origin for each joint.
②	Offset	The float value of the origin range symmetrically around the reference value. Offset value range: [0.1,30]. For example, if the reference value is 1° and the offset value is 3°, the origin falls in the range of [-2°,4°].

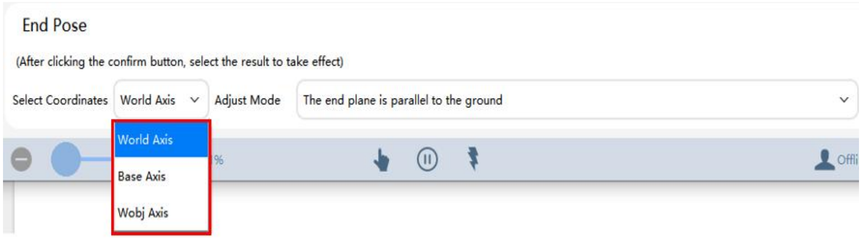
The relationship between the origin range, the reference value, and the offset value is shown as follows:



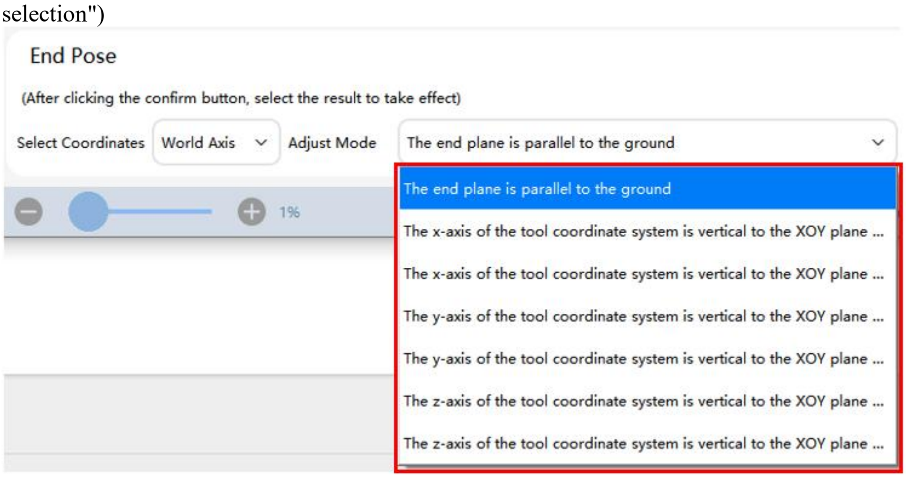


This quick adjustment function also allows rapid adjustment of the end-effector to specified orientations while maintaining the robot's TCP position and arm angle (the concept of arm angle applies only to 7-axis robots). The tool quick adjustment offers two configurable options: "Frame selection" and "Adjustment method".

The Frame selection dropdown includes:



The Adjustment method dropdown includes: (When a frame is selected in the "Frame selection", the contents of the "Adjustment method" dropdown will change accordingly. The example below shows the available options of the "Adjustment method" when the world frame is selected in the "Frame selection")



Procedure:

The usage of the quick adjustment function is similar to JOG operation. The specific steps are as follows:

1. Ensure that the correct tool and work object are selected;
2. Determine the target pose of the robot's end-effector;
   a) Select the frame: (For example, select the world frame as shown in the figure below)
   b) Select the adjustment method: (For example, Align the flange plane parallel to the ground, as shown in the figure below)
   c) Click "Confirm" to apply;

2. Switch to manual mode and power on;

3. Press and hold the "Move to" button. The robot will move to the target pose via joint-space trajectory;

Note:

1. When using the quick adjustment function to adjust the pose, please turn on the soft limit to avoid unexpected collisions.
2. The motion speed can be adjusted via the Jog speed. If the power is turned off or the "Move to" button is released during motion, the robot will stop moving.
3. Use matched tool/work object types (hand-held tool/external work object or hand-held work object/external tool). Otherwise, motion will be prohibited, and corresponding error prompts will be displayed for incorrect selection;
4. In the case of an external tool, adjustments perpendicular to the XOY plane of either the world or base frame are not allowed, and incorrect selections will trigger corresponding error

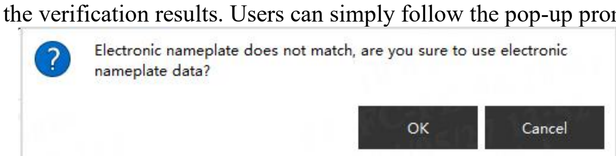
prompts.

9.13Electronic nameplate

The electronic nameplate designed for some industrial robots is installed in the robot body. It is mainly used to save the data of the robot body and avoid the loss of basic data after the replacement of the industrial computer or the controller cabinet.

The software function of electronic nameplates is mainly divided into two parts: Controller and RobotAssist software. The controller carries out the data reading, verification, and coverage of the electronic nameplate, while the RobotAssist software issues operation commands related to the electronic nameplate and displays data.

After the controller is turned on, it will first check if there is an electronic nameplate. If there is an electronic nameplate, it will read the data normally, perform data verification, and store the verification result. If there is no electronic nameplate and the user does not choose to use the electronic nameplate, it will directly operate with the controller data. If there is no electronic nameplate and the user chooses to use the electronic nameplate, a prompt "there is no electronic nameplate" will appear. After Robot Assist is connected to the controller, it will first check the verification results of the electronics nameplate data in the controller, and give different pop-up prompts based on the verification results. Users can simply follow the pop-up prompts.



If the data in the electronic nameplate is successfully used, it will overwrite the data in the controller by default.

There are three situations where pop-up prompts appear:

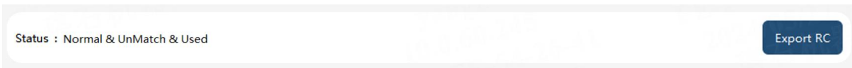
(1) If an electronic nameplate is detected and its data is different from that in the controller, a pop-up window will prompt "Do you want to use the data in the electronic nameplate?". Select "Yes" to use the data in the electronic nameplate and "No" to use the data in the controller;

(2) After choosing to use the data in the electronic nameplate once, the electronic nameplate data will be used by default after restart. If the data in the controller is again different from that in the electronic nameplate, a pop-up window will prompt, "Do you want to use the data in the electronic nameplate?";

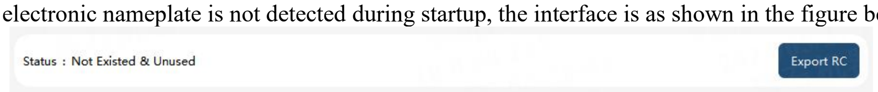
(3) If the electronic nameplate is not detected during startup, the controller data will be used by default. If the electronic nameplate data is used once and cannot be detected after restart, a pop-up window will prompt "Do you want to use the data in the controller?". Select "Yes", the controller data will be used normally. Select "No", the controller will be in a malfunction state and cannot be operated. In this case, restart the controller to solve the problem.

> Note
>
> When the model data in the electronic nameplate does not match that in the controller, the data in the electronic nameplate cannot be used. To use the electronic nameplate data successfully, please ensure the model data in the controller is as same as that in the electronic nameplate.

Click "Settings"->"Electronic nameplate" in turn on the HMI software interface, and the information in the electronic nameplate will appear. If the controller detects an electronic nameplate, whether the electronic nameplate is used or not, this interface will display the information of the relevant parameter segments in the electronic nameplate. The status of the electronic nameplate can be determined through the status field on this interface, with three parameters representing: electronic nameplate status, matching of electronic nameplate data with controller data, and whether the "use electronic nameplate" button is clicked when starting up (as long as the "use electronic nameplate data" button is clicked, regardless of whether the data is successfully used, this position will be displayed as used), as shown in the following figure:

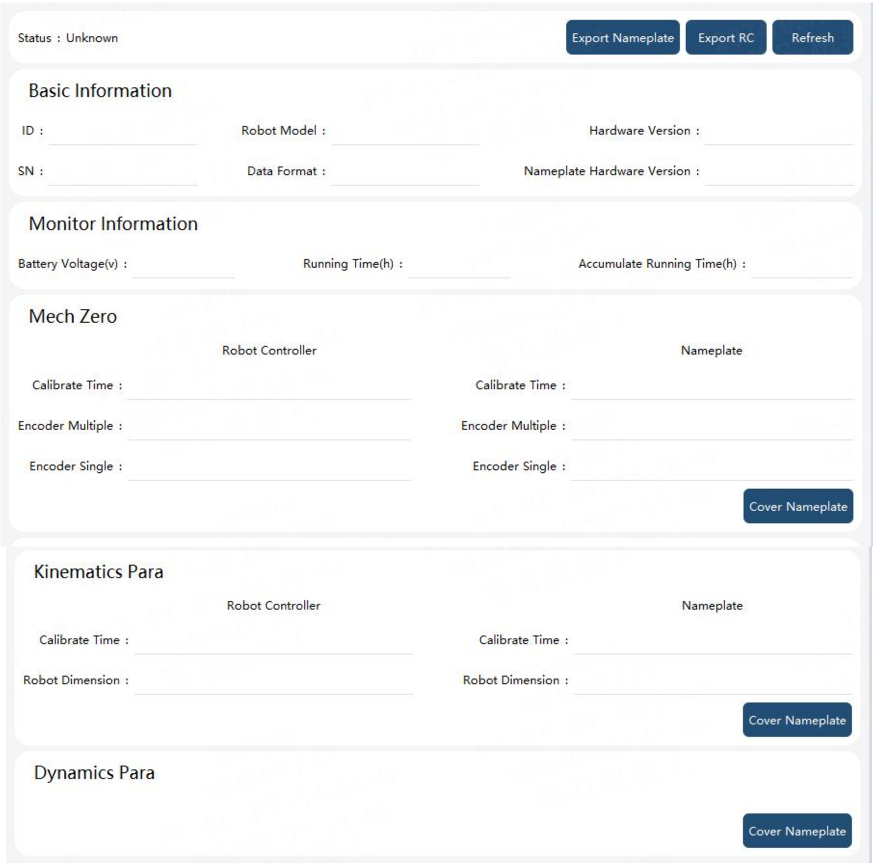


If an electronic nameplate is not detected during startup, the interface is as shown in the figure below:



Function	Explanation
Export RC	Export the data of relevant parameter segments in the controller to a file
Export nameplate	Export the data in the electronic nameplate to a file

Refresh	Synchronize the information of the electronic nameplate
Basic information	The parameter segments about the basic information of the electronic nameplate. It is unable to be modified manually
Battery voltage	The actual battery voltage of the encoder. It is measured during startup and every 24 hours after a startup. It is unable to be modified manually
Running time	When the motor runs, the running time increases accordingly. The value is refreshed every hour on the interface. This parameter cannot be modified manually;
Mechanical zero parameters and kinematic parameters	The current values of the controller and the electronic nameplate will be displayed on the interface, respectively. This parameter cannot be modified manually;
Dynamic parameters	The parameter segment is not displayed on the interface;
Overwrite electronic nameplate data	Overwrite the data in the electronic nameplate with the data in the controller.

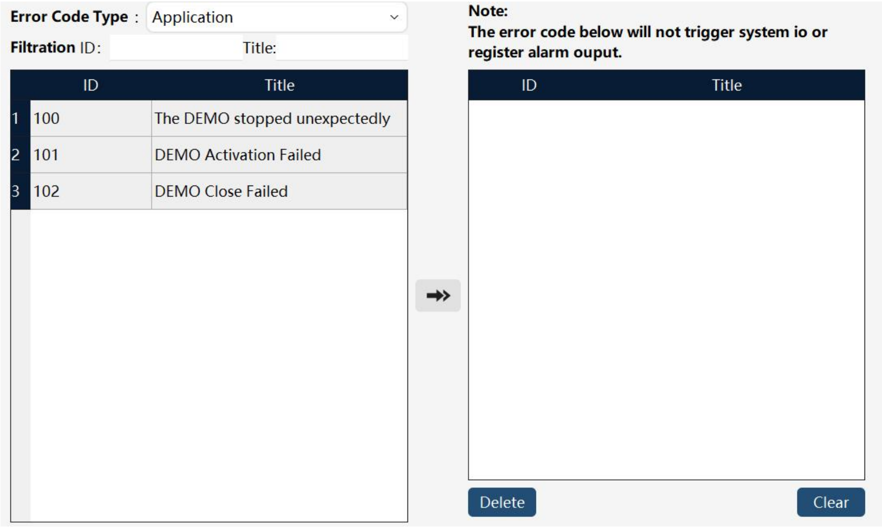


Note

1. All exported data are encrypted.
2. When the electronic nameplate is used, the controller automatically synchronizes the modified data to the electronic nameplate after the robot performs zero calibration, robot parameter modification, or dynamic parameter identification.

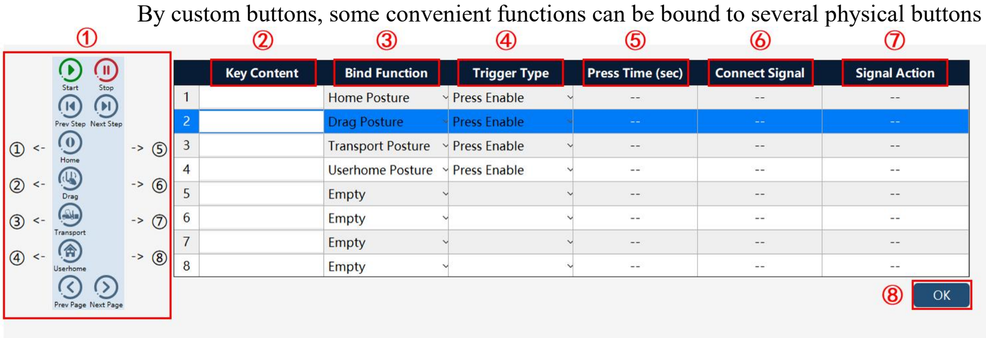
9.14Error code alarm filtering

When an error level alarm occurs in xCore, it will trigger the system IO or register output high level bound to the alarm state. If the customer does not want certain alarms to trigger alarm state outputs (system IO or registers), this function can be used for setting.



9.15Custom buttons

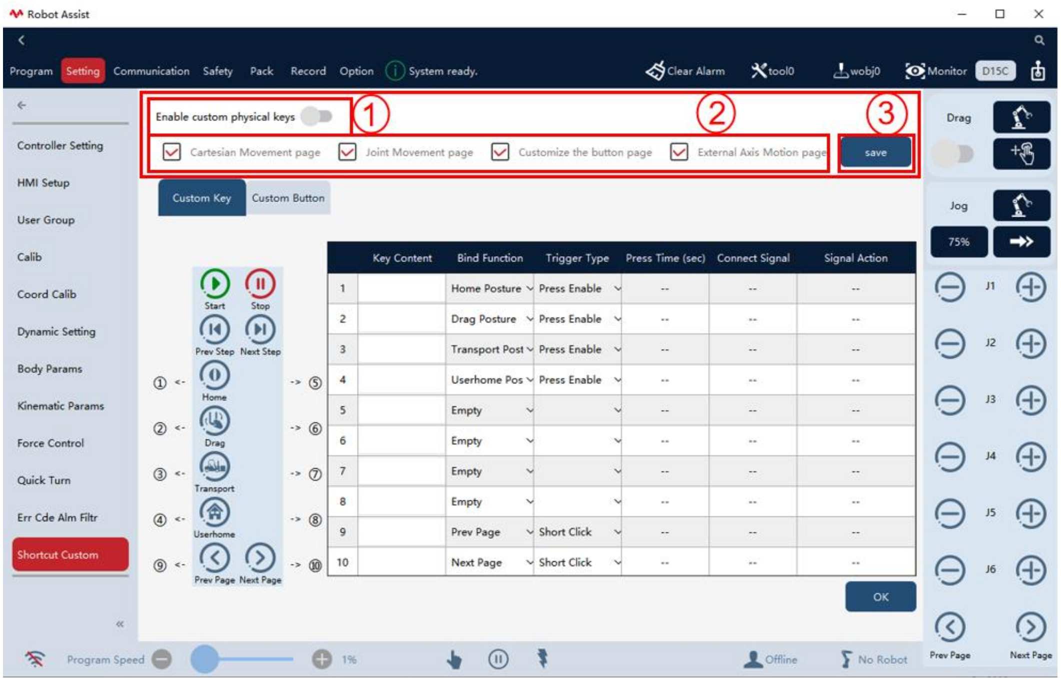
By custom buttons, some convenient functions can be bound to several physical buttons on xPad2.



①	The positions of these 8 buttons can be customized;
②	Key content: Users may set the button name, which is displayed below the corresponding button icon. Names should not be too long to avoid incomplete display.
③	Bind function: There are several functions for selecting, including "Empty, Forced DO, Forced DI, Screenshot, Soft Keyboard, Initial Pose, Drag and Drop Pose, Delivery Pose, User Pose, and Screen Lock (only supporting for xPad2 Teach Pendant)".
④	Trigger type: It includes press for enabling, short press for triggering, and long press for triggering. Press for enabling means that pressing the button enables the corresponding function. Short press for triggering means that pressing the button immediately triggers the bound function. Long press for triggering means that pressing and holding the button for a long time triggers the bound function, with the long press time settable. Different bound functions have various triggering modes.
⑤	Press time: After selecting "long press for triggering," the long press time is set in seconds, with a range of [0.5, 5].
⑥	Connect signal: When the bound function is "forced DO" and "forced DI", the signal can be selected here.
⑦	Signal action: When the bound function is "Forced DO" and "Forced DI", the signal change is detected by the signal behavior, supporting "0", "1", and "Alternate." "0" refers to after triggering, the corresponding signal is set to 0, while "1" means after triggering, it is set to 1. "Alternate" represents that after triggering, if the current signal is 0, it is set to 1, and vice versa.
⑧	OK: The modification of the custom button will take effect after clicking "OK".

9.15.1Custom button disable function

This function is primarily used to prevent accidental triggering of physical buttons on the custom page. Users can customize the enable/disable status of physical buttons on the custom page.



This function includes the following parts:

① Custom physical button enable button: Control the enable/disable relationship of physical button switches across pages.

When the enable button is turned off, all Physical button switches below the enable button are deactivated, and the physical buttons on all pages remain active.

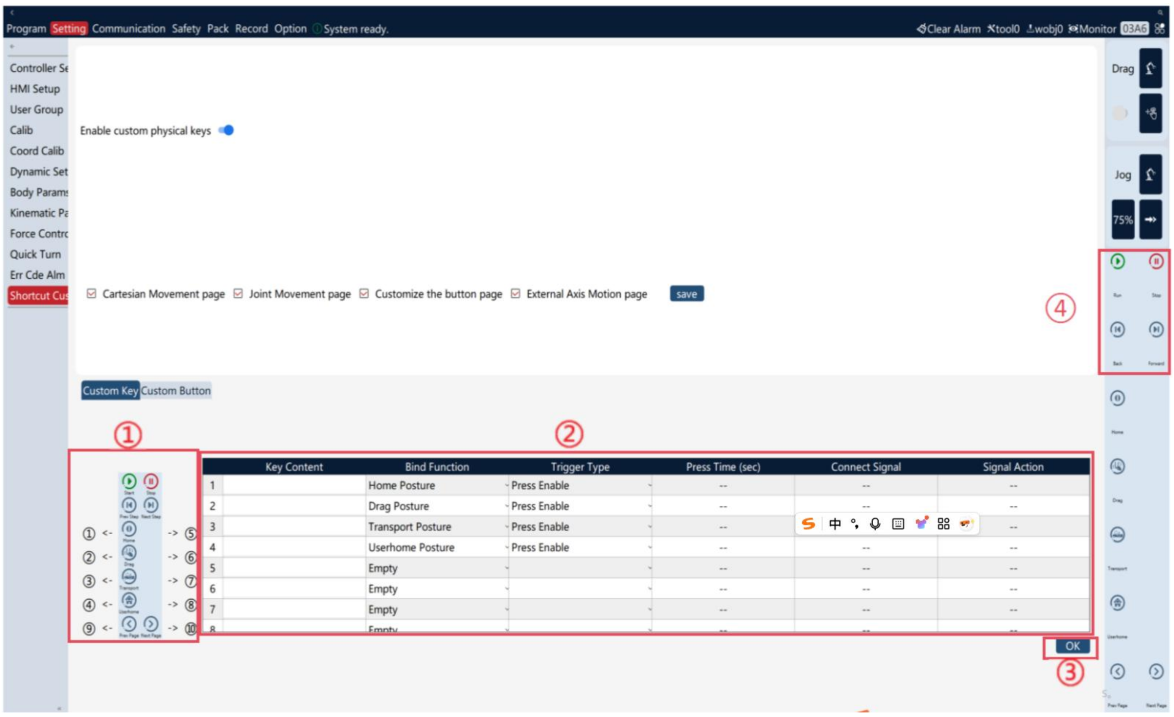
When the enable button is turned on, only physical buttons on the selected pages of Physical button switch module become active.

② Physical button switch module: These switches become active only when the enable button is turned on; checking the box before a specific page indicates that the physical button functions on that page are effective.

③ Save button: Modified states are applied only after clicking the Save button.

9.15.2Custom button - Insert Next Row

This function is designed for industrial robots and provides four configurable custom button functions. The custom button configuration feature is not displayed on the interface of collaborative robots.



This function includes the following parts:
① Custom button style display module: Used to display the names and icons of configured function buttons, with numeric identifiers that correlate to corresponding functions in the right-side table.
② Custom function design table: Allow switching the style and function of custom buttons by modifying the "Bound Function" column.
③ Save button: Modified table configurations will only be saved and applied after clicking the Save button.
④ Custom button function module: The part of the software where the configured custom button functions can take effect.

10Communication

10.1Introduction to this chapter

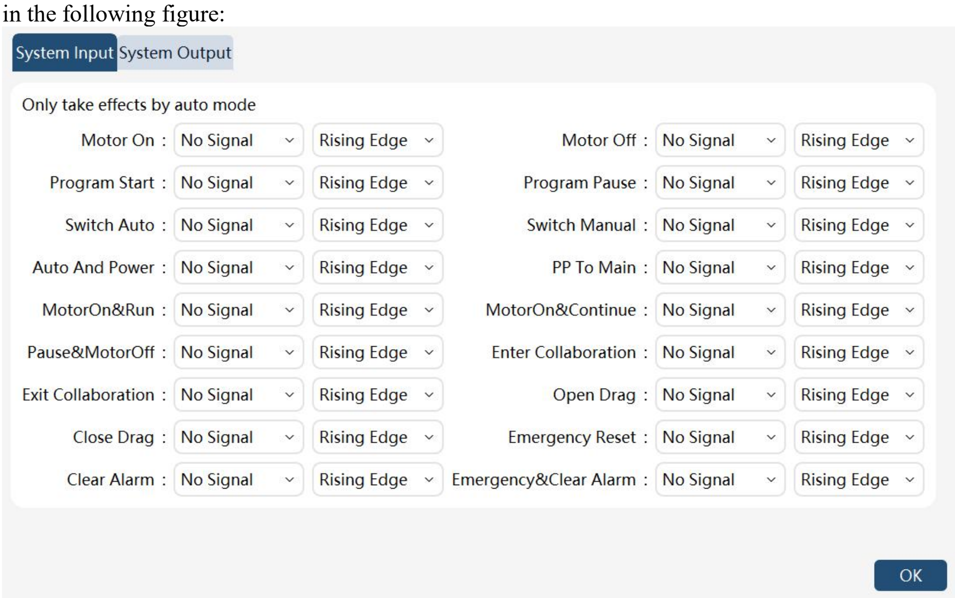
This chapter mainly introduces various communication settings of the xCore control system, including system IO, registers, bus devices, and end-effector.

10.2System IO

System IO is divided into two types: system input and system output. The external controller can send various commands to the xCore control system through system input, such as power-on of the motor, start-up of the program, and emergency stop reset. The xCore system can also use system output to send robot status to the outside world, such as power on/off status and operating status.

10.2.1System input

On the HMI main interface, click "Communication" -> "System IO" to enter the system IO settings interface, and click the "System Input" tab to enter the system input configuration interface, as shown in the following figure:



The system inputs supported by the xCore control system are as follows:

System Input	Trigger Method	Remarks
Motor on	Posedge/Negedge	
Motor off	Posedge/Negedge	
Start program	Posedge/Negedge	After triggering the function code, if the teach pendant displays the alarm "Program not synchronized to controller, startup failed", synchronize the program and retrigger the function code.
Program pause	Posedge/Negedge	
Program pause 1	Posedge/Negedge	This function serves the same purpose as the "Pause Program" function
Switch auto	Posedge/Negedge	It is effective in manual mode
Switch manual	Posedge/Negedge	
Auto and power	Posedge	It is effective in manual mode and switches to automatic mode and power on. The robot does not respond to the command when it is in power-on mode.
PP to main	Posedge	
Motor on & run	Posedge	Power on, pptomain, and running in order. After triggering the function code, if the teach pendant displays the alarm "Program not synchronized to controller, startup failed", synchronize the program and retrigger the function code.
Motor on & continue	Posedge	Power on and running in order. After triggering the function code, if the teach pendant displays the alarm "Program not synchronized to controller, startup failed",

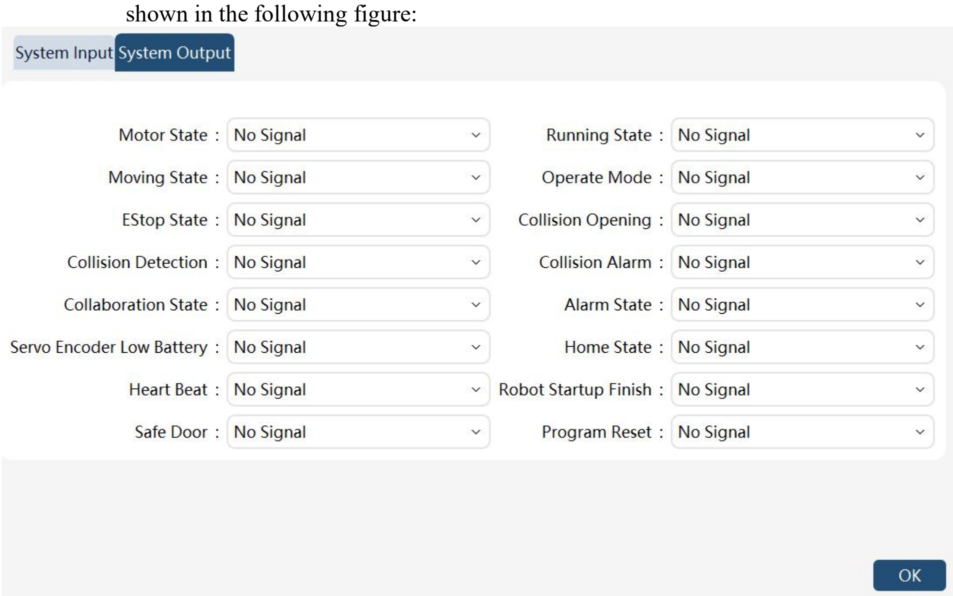
		synchronize the program and retrigger the function code.
Pause & motor off	Posedge	Pause, wait for the robot to stop, and power off
Enter the reduced mode	Posedge/Negedge	Collaborative robot only
Exit the reduced mode	Posedge/Negedge	Collaborative robot only
Open drag	Posedge/Negedge	Collaborative robot only. It is required to enable Drag mode on the interface
Close drag	Posedge/Negedge	Collaborative robot only. It is required to enable Drag mode on the interface
Emergency reset	Posedge	
Clear alarm	Posedge	
Emergency & clear alarm	Posedge	

Note:

- All system inputs are pulse-triggered. To ensure that the xCore system receives external commands correctly, please ensure that the pulse width of the external input is not less than 300 milliseconds.
- There is a corresponding relationship between the functions that support the triggering of the posedge and negedge (power on, power off, manual mode, automatic mode, start program, pause program, enter collaboration mode, exit collaboration mode, startDrag, and stopDrag) to ensure safety, for example, the functions of "power on" and "power off" are corresponding. If DI1 is selected for "power on", DI1 cannot be used for functions other than "power off".
- Most system input function is only valid in Automatic mode, and the signal from the system input in manual mode will be ignored.
- It is not allowed to start the program through any ways when the register equipped with the pause function or system IO has not been reset.
- The “Pause Program” and “Pause Program 1” functions serve the same purpose. Either function will pause the program upon triggering from any signal path.

10.2.2System output

On the HMI main interface, click "Communication" -> "System IO" to enter the system IO settings interface, and click the "System Output" tab to enter the system output configuration interface, as shown in the following figure:



The system outputs supported by the xCore system are as follows:

System Output	Valid Output	Invalid Output	Remarks
Motor state	Motor power-on	Motor power-off	
Running state	Program running	Program not running	
Moving state	Moving	Stationary	Only detect the robot's motion status when detecting motion commands and Jog in the RL program. (Note: In identification, dragging, force control, and drag playback, even if the robot is in motion, the output is still stationary.)

Operate mode	Switch auto	Manual mode/Wait mode	
EStop state	EStop state	Non emergency stop	The output of this state is affected by the "Emergency Stop Trigger Level Type" setting. When it is set to high level, the output is valid when triggering a soft emergency stop, and invalid when not triggered; when it is set to low level, the output is invalid when triggering a soft emergency stop, and valid when not triggered.
Collision opening	Open	Close	Cobots only
Collision detection	Triggered	Not triggered	Cobots only
Collision alarm	Collision detection alarm triggered	Alarm reset	Cobots only
Reduced mode state	Reduced mode triggered	Reduced mode not triggered	Cobots only
Alarm state	Alarm state	No alarm	
Servo encoder low battery	Low voltage alarm	Normal voltage	
Home state	Each joint of the robot is at Home	Each joint of the robot is not at Home	
Heart beat	Heart beat		Click "Settings –> Controller Settings" and set the heartbeat cycle
Robot startup finish	The robot controller finishes the power-on	The robot controller does not finish the power-on	
Safety door	Safety gate opened	Safety gate closed	
Program reset	Program reset	Program not reset	
External estop state	External estop state	Non-external estop state	The safeboard adopts a mini board, and the firmware version is not less than 1.0.8.7
Handheld estop state	Handheld estop state	Non-handheld estop state	The safeboard adopts a mini board, and the firmware version is not less than 1.0.8.7
PPTOMAIN execution state	The current program is executing pptomain	The current program is not executing pptomain	
Soft E-stop trigger state	Soft E-stop trigger state	Soft E-stop not triggered	
On planned path	Robot's current position is on the planned path	Robot's current position is not on the planned path	Refer to register function code "sta_on_path" for detailed usage
Near planned path	Robot's current position is near the planned path	Robot's current position is not near the planned path	Refer to register function code "sta_near_path" for detailed usage

Note:

- The system output status is valid in both manual and automatic modes. However, for safety and availability considerations, these signals are only to be used when the xCore is in Automatic Mode.
- After an IO point is bound to the system IO, it cannot be forced to output or simulate input operations.
- All other system output signals are active at a high level except the "Operating Mode" signal.
- For the signal "Operating Mode", the output is at a high level in Automatic mode and low in Manual mode.

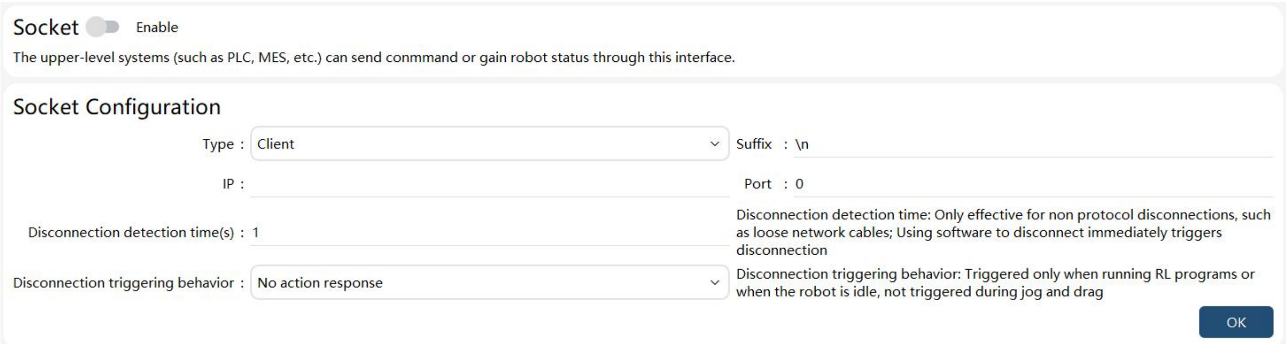
10.3External communication

10.3.1Overview

The xCore system provides a Tcp Socket-based external communication interface supporting both server and client through which host systems (PLC, MES, etc.) can send control commands to the robot or obtain the robot status.

10.3.2Configurations

Before using the interactive commands, configure the parameters related to the Socket communication and enable the function. The configuration interface is located in HMI -> "Communication" -> "External Communication", as shown in the following figure:



Note:
The Socket communication interface supports the robot to serve as a client or server, but only one state at a time.

When the robot is used as a client, the following parameters need to be configured:

Parameters	Explanation
IP	Server IP, such as the IP address of the connected PLM and MES systems.
Port	Server-side listening port
Suffix	When the server sends control or monitoring commands to the robot, an additional suffix character is required at the end of the command. They are

	typically simple terminators such as \r, \n, or \t. Please note that combined suffixes can be used here without limitation on length, such as \r\n, \r\t, or \r\n\t. Visible characters such as letters can also be used.

The robot used as a server supports multiple connections. In this case, please pay attention to the control sequence on the client side to avoid any conflict. The following parameters need to be configured:

Parameters	Explanation
Port	Server-side listening port
Suffix	When the server sends control or monitoring commands to the robot, an additional suffix character is required at the end of the command. They are typically simple terminators such as \r, \n, or \t. Please note that combined suffixes can be used here without limitation on length, such as \r\n, \r\t, or \r\n\t. Visible characters such as letters can also be used.

Note

To ensure the stability of the robot's motion control, the control system allocates only a portion of its computational resources to network communication functions. When the robot acts as a socket server listening for connections, if it receives extremely frequent network connection requests or data streams resembling a "DDoS attack", this may cause the robot's network connections to external devices (such as the teach pendant and other equipment) to disconnect or result in operational lag.

Network connection or data interaction frequency must remain below 1 per millisecond (1/ms).

10.3.3Interactive commands

Interactive commands include control commands and monitoring commands.

The following table gives the specific command content and format. (Assuming the user uses "\ r" as the specified command terminator, "\ r" is an escape character representing carriage return, and the decimal value is 13).

Control commands:

Command name	String sent	Return value	Remarks
Close the socket interface	“xCore::SocketInterface::Disable” +“\r”	No return value	
Start the socket interface	“xCore::SocketInterface::Enable” +“\r”	No return value	
Start program	“start”+“\r”	"true" if success; "false" if failed	It is not allowed to start the program through system IO when the register equipped with the pause function or system IO has not been reset; After triggering the function code, if the teach pendant displays the alarm "Program not synchronized to controller, startup failed", synchronize the program and retrigger the function code;
Stop program	“stop”+“\r”	"true" if success; "false" if failed	
Clear servo alarms	“clear_alarm”+”\r”	"true" if success; "false" if failed	
Program pointer	“pp_to_main”+“\r”	"true" if success;	

to main		"false" if failed	
Motor power-on	"motor_on"+"\r"	"true" if success; "false" if failed	
Motor power-off	"motor_off" + "\r"	"true" if success; "false" if failed	
Switch to Manual mode	"switch_mode:manual"+"\r"	"true" if success; "false" if failed	
Switch to Automatic mode	"switch_mode:auto"+"\r"	"true" if success; "false" if failed	
Enable Drag mode	"open_drag"+"\r"	"true" if success; "false" if failed	Collaborative robot only
Disable Drag mode	"close_drag"+"\r"	"true" if success; "false" if failed	Collaborative robot only
Obtain project list	list_prog + "\r"	Return to project list; "null" if no project is returned	
Obtain current project	current_prog + "\r"	Return to current load project; "null" if no project is returned	
Switch to project	load_prog + (project name) + "\r"	"true" if success; "false" if failed	
Set DO value	"setdo:" + "IO name, IO value" + "\r"	"true\r" if successful "false\r" if not found	"setdo: DO0-0,true\r" (pay attention to English punctuation)
Update the time on the controller and teach pendant	set_robot_time:time + "\r" (time format:Y YYY-MM-DD hh:mm:ss)	"true" if success; "false" if failed	
Emergency reset	estop_reset + "\r"	"true" if success; "false" if failed	(RSC only, not applicable to mini board)
Emergency & clear alarm	estopreset_and_clearalarm +"\r"	"true" if success; "false" if failed	
Power on, program pointer to main, and start program in order	motoron_pptomain_start +"\r"	"true" if success; "false" if failed	After triggering the function code, if the teach pendant displays the alarm "Program not synchronized to controller, startup failed", synchronize the program and retrigger the function code;
Power on and start program in order	motoron_start +"\r"	"true" if success; "false" if failed	After triggering the function code, if the teach pendant displays the alarm "Program not synchronized to controller, startup failed", synchronize the program and retrigger the function code;
Pause program and power down in order	pause_motoroff +"\r"	"true" if success; "false" if failed	
Set program	set_program_speed: +program	"true" if success;	

running rate	speed +"\r"	"false" if failed	
Trigger and release robot soft E-stop	set_soft_estop:true/false +"\r"	"true" if success; "false" if failed	
Execute switch to automatic mode and then power on	switch_auto_motoron +"\r"	"true" if success; "false" if failed	
Open the corresponding safe region	open_safe_region: +safe region index+"\r"	"true" if success; "false" if failed	Index range [1−10]
Close the corresponding safe region	close_safe_region: +safe region index+"\r"	"true" if success; "false" if failed	Index range [1−10]
Enable reduced mode	open_reduced_mode +"\r"	"true" if success; "false" if failed	
Disable reduced mode	close_reduced_mode +"\r"	"true" if success; "false" if failed	

Monitoring commands:

Command name	String sent	Return value	Remarks
Motor power state	"motor_on_state" + "\r"	"true" if success, motor power-on; "false" if failed, motor power-off;	
Program status	"robot_running_state" + "\r"	"true" if success, running; "false" if failed, not running;	
EStop state	"estop_state" + "\r"	When the emergency stop trigger level type is set to high: true, emergency stop; false, non-emergency stop. When the emergency stop trigger level type is set to low: true, non-emergency stop; false, emergency stop.	The return value of this state is affected by the "Emergency Stop Trigger Level Type" setting. When it is set to high level, the output is valid when triggering a soft emergency stop, and invalid when not triggered; when it is set to low level, the output is invalid when triggering a soft emergency stop, and valid when not triggered.
Fault	"fault_state" + "\r"	"true" if success, fault; "false" if failed, non-fault;	
Operating mode	"operating_mode" + "\r"	"true" if success, automatic mode; "false" if failed, manual mode/wait mode;	
Get Cartesian position	"cart_pos" + "\r"	Cartesian position string + "\r";	
Get Cartesian position	"cart_pos_name" + "\r"	"cart_pos: " + Cartesian position string + "\r";	
Get axis position	"jnt_pos" + "\r"	Axis position string + "\r";	
Get axis position	"jnt_pos_name" + "\r"	"jnt_pos: " + axis position string +"\r";	
Get axis velocity	"jnt_vel" + "\r"	Axis speed string + "\r";	Unit: rad/s
Get axis velocity	"jnt_vel_name" + "\r"	"jnt_vel: "+ axis speed string + "\r";	Unit: rad/s
Get axis torque	"jnt_trq" + "\r"	Axis torque string + "\r";	Unit: N.m
Get axis torque	"jnt_trq_name" + "\r"	"jnt_trq:" + axis torque string	Unit: N.m

		+ "\r";	
Home state output	“home_state” + “\r”	"true" with output; "false" without output	
Collision detection state	“collision_state” + “\r”	"true" if trigger collision; "false" if no collision	Collaborative robot only
Obtain robot task state	“task_state” + “\r”	The task currently performed by the robot. These include: ready; jog; load_identify; dynamic_identify; drag; program; demo; rci; debug;	Please refer to the "HMI Introduction" for the icon and description of the current status of the robot
Obtain alarm state	“alarm_state”+”\r”	"true" if there is an alarm, "false" if no alarm is present	
Obtain collision detection alarm state	“collision_alarm_state” + ”\r”	"true" if there is an alarm, "false" if no alarm is present	
Obtain collision detection enable state	“collision_open_state” + ”\r”	"true" if collision detection is enabled "false" if it is not enabled	
Check if the controller is powered on	“controller_is_running” + ”\r”	"true" if powered on "false" if not powered on	
Low-voltage alarm state of encoder	“encoder_low_battery_state”+ ”\r”	"true" if there is a low voltage alarm "false" if there is no low voltage alarm	
Obtain robot error code	“robot_error_code” + ”\r”	"robot error code" if an error is present "0" if there is no error	
Obtain pause state of RL	“program_full” + ”\r”	Pause state of RL, including: 0: initialization state; 1: RL running; 2: HMI paused; 3: system IO paused; 4: register function code paused; 5: external communication paused; 6: DK paused; 7: paused by Pause command; 10: emergency stop; 11. safety gate; 12. paused due to other factors.	
Obtain program reset state	“program_reset_state”+ ”\r”	Output "true" if the program pointer is at the first line of the main function and the program has not started	

		running; otherwise, output "false"	
Obtain the actual speed of the current program execution	“program_speed” + “\r”	Actual speed of the current program execution Range: 1 to 100	
Obtain program busy state	“robot_is_busy” + “\r”	"true" if currently in pptomain Otherwise, "false"	Obtain whether the robot is executing time-consuming operations such as pptomain
Obtain whether the robot is in motion	“robot_is_moving” + ”\r”	"true" if the robot is moving Otherwise, "false"	
Obtain safety gate state	“safe_door_state” + ”\r”	"true" if safety gate is open Otherwise, "false"	
Obtain soft E-stop trigger status	“soft_estop_state” + ”\r”	"true" if the soft E-stop is triggered; otherwise, "false"	
Obtain Cartesian velocity	“cart_vel” + “\r”	Cartesian velocity+"\r"	
Obtain the pose of the robot's TCP	“tcp_pose” + “\r”	Pose of the robot's TCP+"\r"	
Obtain the velocity of the robot's TCP	“tcp_vel” + “\r”	Velocity of the robot's TCP+"\r"	
Obtain the composite linear velocity of the robot's TCP	“tcp_vel_mag” + “\r”	Composite linear velocity of the robot's TCP+"\r"	
Obtain external E-stop state	“ext_estop_state” + “\r”	"true" if the external E-stop is triggered "false" if it is not triggered	The safeboard is a mini board, and the firmware version is not less than 1.0.8.7
Obtain handheld E-stop state	“hand_estop_state” + “\r”	"true" if the handheld E-stop is triggered "false" if non-handheld E-stop is triggered	The safeboard is a mini board, and the firmware version is not less than 1.0.8.7
Obtain collaboration mode state	“collaboration_state” + “\r”	"true" if collaboration mode is triggered "false" if non-collaboration mode is triggered	
Obtain reduced mode state	reduced_mode_state + “\r”	"true" if it is enabled; "false" if it is not enabled	
Obtain IO state	io_state: + IO names (multiple IOs separated by commas) + "\r"	Return IO values as "true" or "false"; "null" if the corresponding IO is not found	
On path verification	“sta_on_path” + “\r”	"true" if on path "false" if off path	Refer to register function code "sta_on_path" for detailed usage
Near path point verification	“sta_near_path” + “\r”	"true" if near path point "false" if not near path point	Refer to register function code "sta_near_path" for detailed usage

Note:

	String format	Unit
Cartesian position	x, y, z, a, b, c, q1, q2, q3, q4	x, y, and z in mm; a, b, and c in degree; Q1~q4 are orientation quaternions;

Axis position	j1, j2, j3, j4, j5, j6, j7	Robot axis angle in rad; Track position in m;
Axis velocity	vj1, vj2, vj3, vj4, vj5, vj6, vj7	Robot velocity angle in rad/s; Track velocity in m/s;
Axis torque	tj1, tj2, tj3, tj4, tj5, tj6, tj7	The unit of the robot axis and track torque is the thousandth of the rated torque of the motor;

10.4Bus devices

10.4.1Overview of bus devices

CC-Link, Modbus, EtherCAT, and PROFINET are supported.
CC-Link includes CC-Link devices (connected via EtherCAT) and CC-Link IE Field Basic.
EtherCAT can be used to expand bus modules such as IO modules, PROFINET, and EtherNet/IP.

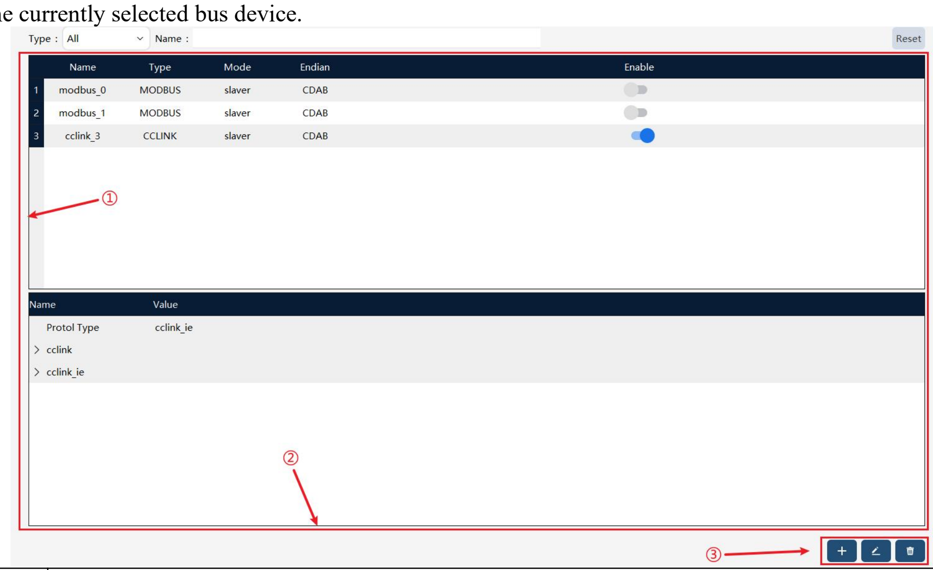
Supported Bus	Protocol	Supported method	Remarks
Modbus	TCP	Master and slave	
	UDP	Not supported	
	RTU	Master and slave	Industrial robots only
CC-Link	485	Remove device station (slave)	Industrial robots only
	IE Field Basic	Remove device station (slave)	

The following function codes are supported in Modbus:

Function code	Meaning	Supported
0x01	Read coil	Supported
0x05	Write a single coil	Supported
0x0F	Write multiple coils	Supported
0x02	Read discrete input	Supported
0x04	Read input register	Not supported
0x03	Read holding register	Supported
0x06	Write a single holding register	Supported
0x10	Write multiple holding registers	Supported

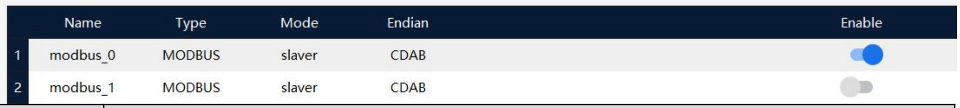
10.4.2Bus devices parameter configuration

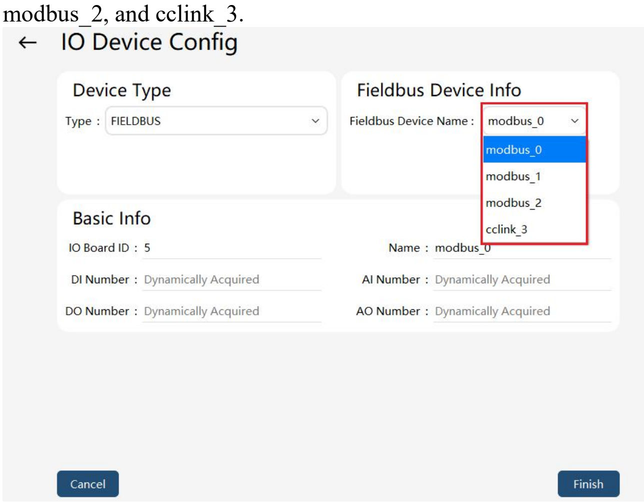
The page is at: "Communications" -> "Fieldbus Devices". The page is divided into two parts. The upper part manages all bus connections and allows for individual opening and closing operations for each bus connection. When the bus connection is closed, the IO configured for this connection will not be displayed in "Status Monitoring" -> "IO Signal". The lower part is the attribute parameters of the currently selected bus device.

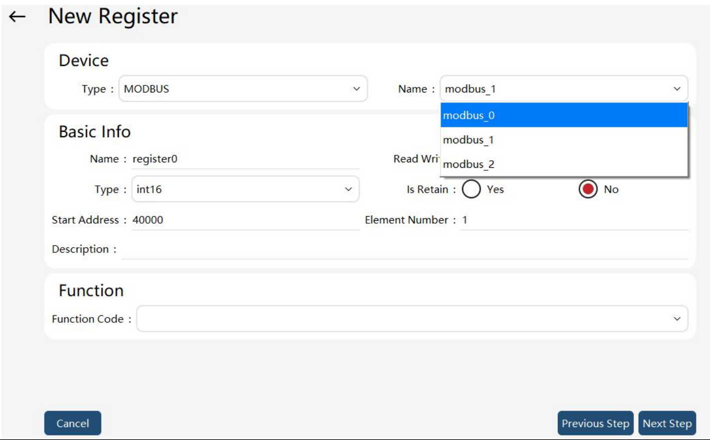


①	List of bus devices.
②	The parameters of the currently selected bus device.

③	The bus device operation buttons, from left to right, are Create, Edit, and Delete.



<table>
<tr><th>Parameter Name</th><th>Parameter Explanation</th></tr>
<tr><td>Name</td><td>The name is used when configuring "IO Device" and "Register."
For example, the names in the following figure are modbus_0, modbus_1, modbus_2, and cclink_3.


As shown in the figure below:
"IO device config" name field is used in IO device configuration to indicate which bus the IO device is related to;
"Register" name field is used in Register to indicate which bus the register is related to;


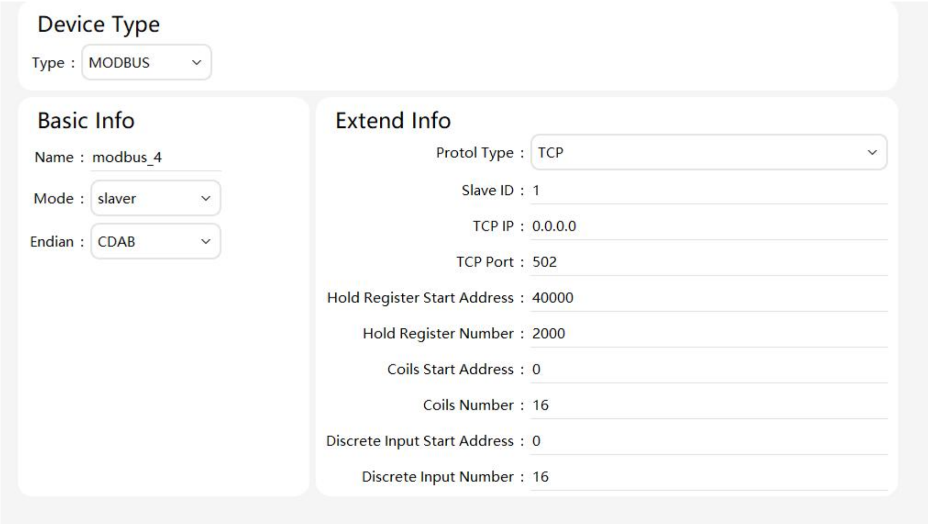
</td></tr>
<tr><td>Type</td><td>It can be selected when adding/editing a bus device. Only CC-Link, Modbus, EtherCAT, and PROFINET are supported. Note:
CC-Link includes CC-Link devices (connected via EtherCAT) and CC-Link IE Field Basic.
EtherCAT can be used to expand bus modules such as IO modules, Profinet, and EtherNet/IP.</td></tr>
<tr><td>Mode</td><td>It indicates whether the current robot is acting as a master or slave on the bus.</td></tr>
<tr><td>Endian</td><td>It is mainly for registers. Since each register occupies 2 bytes, there are many hexadecimal sequences of the two bytes. This attribute needs to correspond to the master and the slave, otherwise, the data will not meet the expectations. Four types of endianness are supported: ABCD, CDAB (default), BADC, and DCBA.</td></tr>
<tr><td>Enabling button</td><td>The bus function can be enabled or disabled through this button. Each bus device to be enabled or disabled individually is supported.</td></tr>
</table>

	Note: After a bus device is disabled, the IOs configured on the bus device will not be displayed in "Status Monitoring" -> "IO Signal".

10.4.2.1Modbus communication

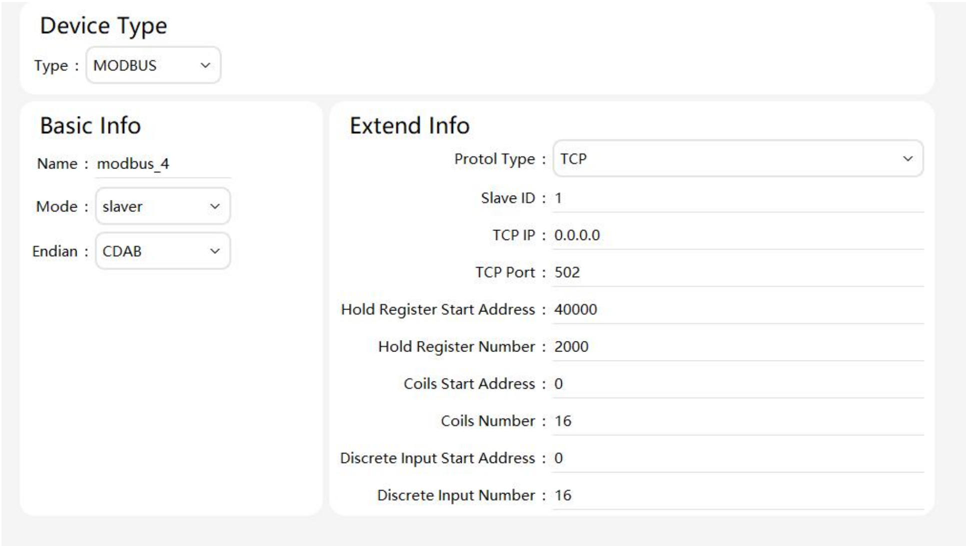
On the bus device page, click on the bottom right corner  to enter the new communication bus device page, and select the device type as "MODBUS". It supports the TCP and RTU protocol, and the device can be configured as a master or slave.

10.4.2.2Modbus TCP configuration



Parameter	Introduction
Type	"master", the robot serves as a master; "slave", the robot acts as a slave;
Slave ID	When the robot serves as a slave, ensure that the overall configuration of the bus does not conflict with other slaves. When the robot serves as a master, it indicates the target slave ID that the robot expects to communicate with. Note: When the robot serves as a master, it only supports single-slave communication with external devices;
TCP/IP	When the robot serves as a slave, fill in 0.0.0.0, which means all network cards are monitored. When the robot serves as a master, fill in the IP address of the target slave ID that the robot communicates with;
TCP port	The port number when the slave uses the TCP protocol.
Holding register start address	The start address of the register affected by the function codes 0x03, 0x06, and 0x10. Each register occupies 2 bytes. For write-only registers, when the robot acts as a slave, the holding register function code is 0x03, and when the robot acts as a master, the holding register is 0x06 or 0x10. For read-only registers, when the robot acts as a slave, the holding register function code is 0x06 or 0x10, and when the robot acts as a master, the holding register function code is 0x03.
Holding Registers Number	The number of holding registers from the holding register start address.
Coil start address	The start address of the register affected by the function codes 0x01, 0x05, and 0x0F.
Coils Number	The number of coil registers from the coil start address.
Discrete Input Start Address	The start address of the register affected by the function code 0x02.
Discrete Input Number	The number of discrete input registers from the discrete input start address.

10.4.2.3Modbus RTU configuration



The Modbus RTU conception is partly the same as the Modbus TCP conception, which will not be repeated here. Only the differences are described as follows:

RTU serial port name: Indicates the serial port medium used for bus communication. Configure it in "Communication" -> "Serial Port Configuration", including the parameters for communication.

10.4.2.4CC-Link communication

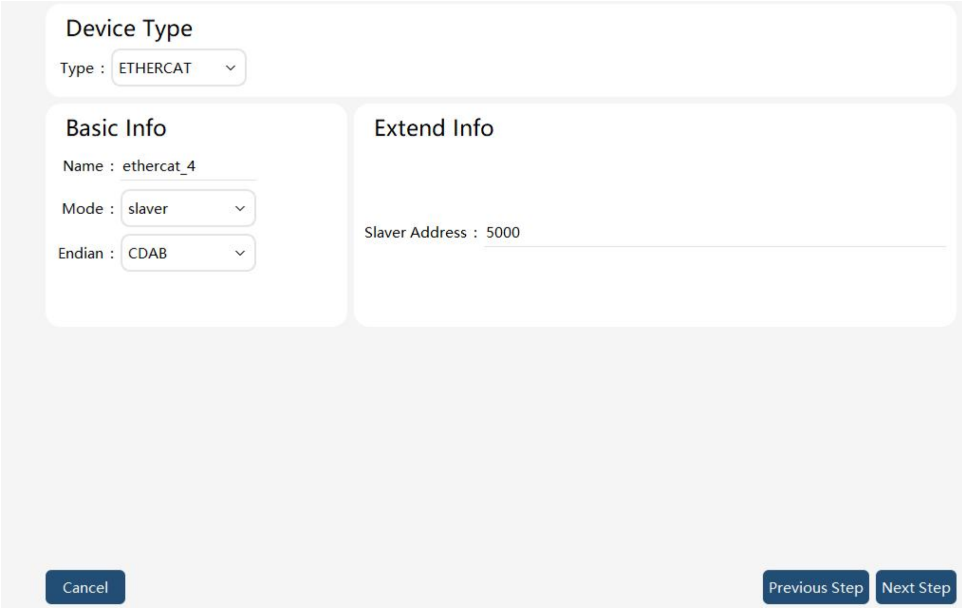
On the bus device page, click on the bottom right corner  to enter the new communication bus device page, and select the device type as "CCLINK". It supports the CC-Link and CC-Link IE Field Basic protocol, and the device can be configured as slave only.

10.4.2.5CC-Link configuration

Parameter	Introduction
Protol type	cclink_ie means the CC-Link IE Field Basic communication protol that directly uses the robot's Ethernet port.
cclink_ie NetCard	Configure which Ethernet card is used for communication.
cclink_ie Occupied Station Number	1 to 16 occupied stations can be configured. The default number is 4. Default values are recommended.
cclink_ie Protol Version	Ver1 or Ver2 is optional. Please ensure that it is consistent with that of the master.

10.4.2.6EtherCAT communication

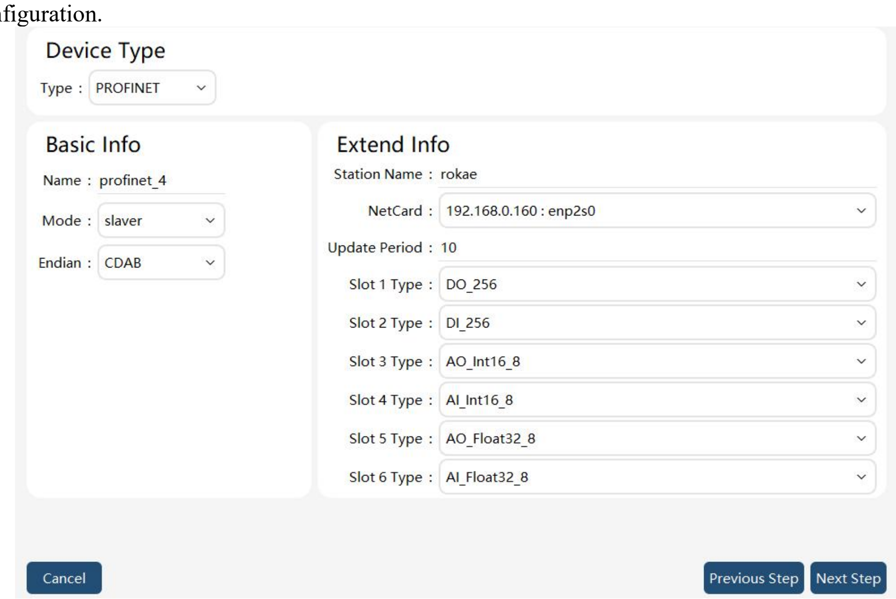
On the bus device page, click on the bottom right corner  to enter the new communication bus device page, and select the device type as "ETHERCAT". EtherCAT can be used to access PROFINET and EtherNet/IP gateway modules.



Slaver Address: The slave address number in the EtherCAT bus topology.
Note: Since the EtherCAT slave address number 1000-4000 is occupied by the robot internal devices, to avoid device address conflict, the EtherCAT slave address number of extended devices should not be less than 5000.

10.4.2.7PROFINET communication

On the bus device page, click on the bottom right corner  to enter the new communication bus device page, and select the device type as "PROFINET". The device can be configured as a slave only. One PROFINET slave can be configured for one robot, and multiple robots can join the same PROFINET network by modifying the PROFINET slave name to enable multiple slaves. The model selected for Slots 1-6 should be consistent with the correspondent-side configuration.



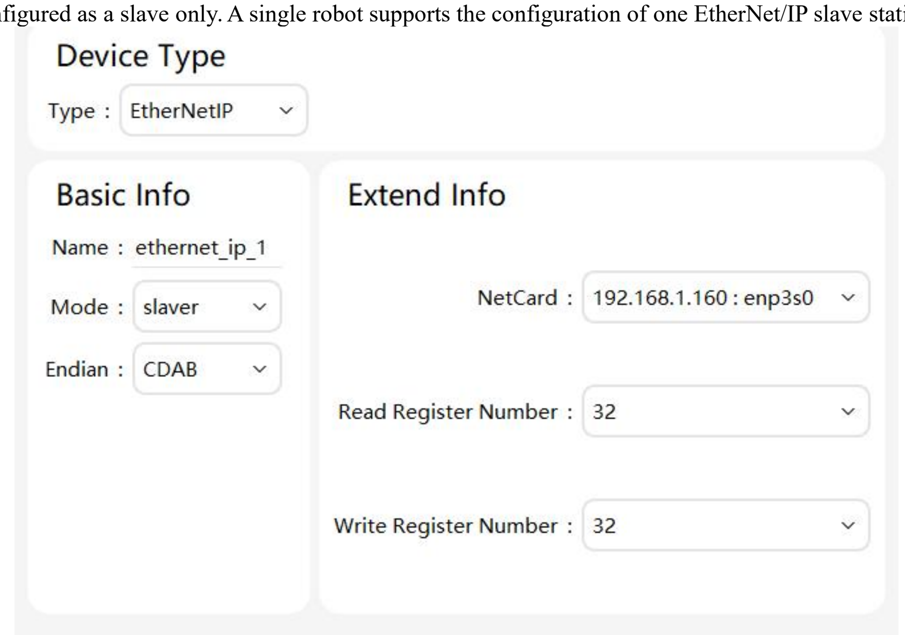
Parameter explanation:

Parameter	Explanation
Device type	Select PROFINET.
Name	Equipment code.
Type	Only slaver is supported.
Endian	Select DCBA generally, depending on the agreement between the communicating parties.
Station name	PROFINET slave name. It should be consistent with the correspondent-side configuration. Chinese characters, uppercase letters, and underlines are not allowed.

NetCard	Select the network port to connect to the correspondent; includes the network card IP and name.
Update period	Default to 10ms, minimum 2ms.
Slot 1 type	Only DO_256 model can be selected, indicating that 256 digital quantities are output from the robot to the correspondent via slot 1.
Slot 2 type	Only DI_256 model can be selected, indicating that there are 256 digital inputs from the correspondent to the robot via slot 2.
Slot 3 type	The option models include AO_Int16_8/ AO_Int16_16/ AO_Int16_32/ AO_Int16_64/ AO_Int16_128/ AO_Int16_256. AO_Int16_8 means that there are 8 int16 analog outputs from the robot to the correspondent via slot 3, and so forth.
Slot 4 type	The option models include AI_Int16_8/ AI_Int16_16/ AI_Int16_32/ AI_Int16_64/ AI_Int16_128/ AI_Int16_256. AI_Int16_8 means that there are 8 int16 analog inputs from the correspondent to the robot via slot 4, and so forth.
Slot 5 type	The option models include AO_Float32_8/ AO_Float32_16/ AO_Float32_32/ AO_Float32_64/ AO_Float32_128/ AO_Float32_256. AO_Float32_8 means that there are 8 float32 analog outputs from the robot to the correspondent via slot 5, and so forth.
Slot 6 type	The option models include AI_Float32_8/ AI_Float32_16/ AI_Float32_32/ AI_Float32_64/ AI_Float32_128/ AI_Float32_256. AI_Float32_8 means that there are 8 flaot32 analog inputs from the correspondent to the robot via slot 6, and so forth.

10.4.2.8Ethernet/IP communication

On the bus device page, click  on the bottom right corner to enter the new communication bus device page, and select the device type as "EtherNetIP". The device can be configured as a slave only. A single robot supports the configuration of one EtherNet/IP slave station.



Parameter explanation:

Parameter	Explanation
Device type	That is, EtherNetIP.
Name	Bus device name, cannot be the same as the names of other bus devices.
Mode	Only slaver is supported.
Endian	Select CDAB generally, depending on the agreement between the communicating parties.
NetCard	Select the name of the network card connecting to the EtherNet/IP master station. The dropdown box will display the IP address and name of the network card.
Read-only Registers Number	Number of read-only registers for the EtherNet/IP slave station: Each register represents 2 bytes. Options are 32, 64, 128, and 248 registers.
Write-only Registers Number	Number of write-only registers for the EtherNet/IP slave station: Each register represents 2 bytes. Options are 32, 64, 128, and 248 registers.

Notes:

- Only one EtherNet/IP bus device is supported. Attempting to create more will result in an error message.
- Register addresses start from 0. If 32 is selected, it indicates that the device has registers with addresses from 0 to 31. When configuring register mappings in the "Registers" interface, pay attention to the address range.
- Read-only and write-only are defined from the perspective of the xCore control system: Read-only registers correspond to EtherNet/IP master Output data; write-only registers correspond to EtherNet/IP master Input data.
- The number of read-only and write-only registers also represents the amount of communication data. The more data, the greater the communication load. Therefore, it is recommended to select the smallest number of registers that meets the requirements.
- The number of bytes for Input and Output data configured in the master station should match the number of bytes contained in the write-only and read-only registers of the slave station, respectively. Otherwise, communication may fail.
- The read-only and write-only registers of the EtherNet/IP bus device are two separate data areas.

  The addresses for configured read-only and write-only registers can overlap.
- The EtherNet/IP slave station of the xCore control system does not provide an EDS file by default. Its Input Assembly Instance ID is 1, and its Output Assembly Instance ID is 2. The master station must match these settings during configuration; otherwise, communication may fail.

10.5Register

10.5.1Overview of registers

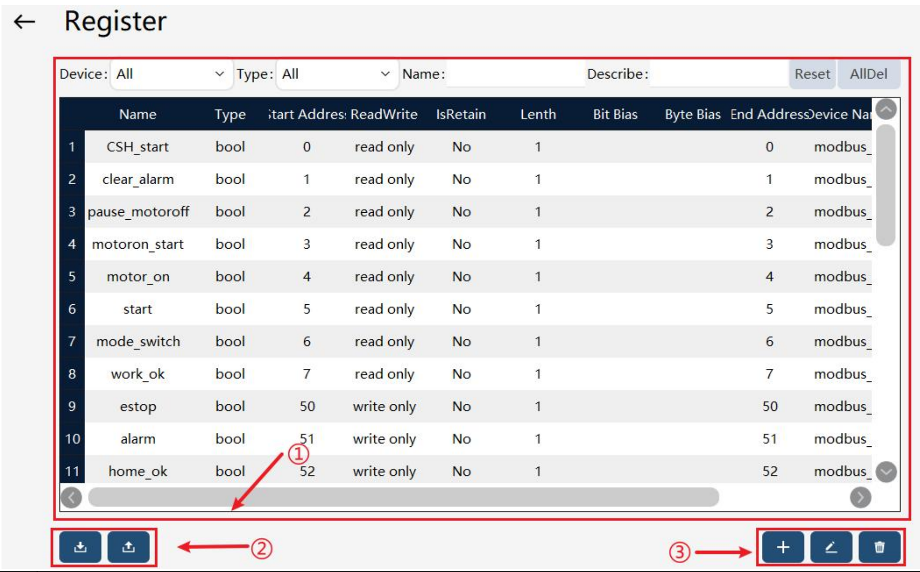
The register represents the available variables within a robot, which are generally used for data communication with external devices, so as to control the robot and obtain its status. The register can also be used as a variable in the current RL project. The register variables can be operated by commands or assignments.

Note:

- The register is a concept of robots themselves, rather than belonging to bus devices. A register can be created or edited by specifying which bus device it is bound to for communication.
- Each register occupies 2 bytes. For different types of variables, the number of registers occupied is different.

10.5.2Register parameter configuration

On the "Communication" -> "Register" page, you can view existing registers and perform Add, Edit, and Delete.



①	Register list.
②	Register import and export button, click to switch to the register import and export page, and perform import and export.
③	The register operation buttons, from left to right, are Create, Edit, and Delete.

	Name	Type	Start Address	ReadWrite	IsRetain	Lenth	Bit Bias	Byte Bias	End Address	Device Name	Function
1	CSH_start	bool	0	read only	No	1			0	modbus_0	ctrl_moto...
2	clear_alarm	bool	1	read only	No	1			1	modbus_0	ctrl_clear_...

Parameter	Explanation
Name	In RL, register variables can be accessed through this name. Note: The list cannot have duplicate names, nor can it have duplicate names with any variables in the RL list. Otherwise, RL will have variable conflicts, which may result in unpredictable consequences.
Type	bit, byte, bool, int16, float, and int32 are supported.
Start Address	The register addresses of the same read and write attributes in the same bus device cannot be cross-occupied, and the register addresses of different read and write attributes in the Modbus bus device cannot be cross-occupied. For example, if one register occupies 41000-41003, another register cannot start from 41002.
ReadWrite	ReadWrite attribute, indicating whether the register is read or written from the robot's perspective (not from the master or slave's perspective). Write-only registers are used for robot external output status; read-only registers are used by robots to obtain commands sent from external devices.
IsRetain	When the register is set to hold, the value of this IsRetain on a non-volatile storage medium during robot restart, shutdown, power outage, or RL stop. When the robot powers on again or RL is running again, the value of register is restored to the value held before the robot shuts down or RL is stopped.
Length	The length represents the number of variables. For variables greater than 1, variable references can be made using arrays, with subscripts starting from 1. Note: It is different from the number of registers. For example: Registers 40140−40153, the variable type is float, and each float occupies 2 registers with a size of 7. Therefore, the number of registers occupied is 2 * 7 = 14.
Bit Bias	Bit Bias represents the position of the bit type register mapped to the register. A register occupies two bytes, which is 16 bits, and the bit offset refers to the position of the corresponding register, with an offset value of 1−16. When creating a bit type register, the bias value can be set if the element number is 1, on the contrary, it cannot be set.
Byte Bias	Byte Bias represents the position of the Byte type register mapped to the register. A register occupies two bytes, which is 16 bits, while a byte variable only requires 8 bits. Therefore, when creating a byte register variable, it is necessary to choose whether to map to 8 bits of LSB (1−8) or 8 bits of MSB (9−16) of the register.
End Address	The end address represents the last register address occupied by the register variable. When the register variables are arranged continuously, the user can quickly understand the space occupied by the register through this value. For example, the start address of the next register can be determined by adding 1 to the value of this item.
Device name	The device name is defined when the "bus device" is created, indicating which bus device is bound to the register. The register can be bound to the CC-Link, CC-Link IE Field Basic, Modbus, and EtherCAT devices.
Function	The content in this column is some fixed function codes, indicating the robot function corresponding to this register. Function codes are divided into read-only and write-only function codes, as detailed in the Register Function Code section below.

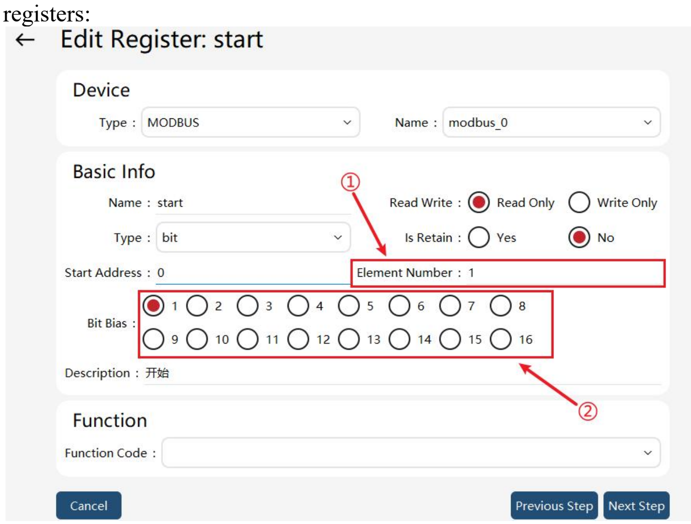
The parameters of each column in the register list are explained in the following table:

10.5.3Register type

Type	Explanation
bit	Only one bit of a register is occupied, and the bit array needs to appear in integer multiples of 16 bits. For example, for a bit type register starting from 41000-bit, a variable with a size of 64 occupies 4 registers from 41000 to 41003.
byte	Only a certain 8 bits of a register are occupied, and LSB (the first 8 bits of the register) or MSB (the last 8 bits of the register). When creating a byte register array, the default is LSB, and MSB and LSB cannot be changed.
bool	Occupy 1 register.
int16	Occupy 1 register.

float	Occupy 2 register.
int32	Occupy 2 register. Note: When the device type is PROFINET, creation is not supported.

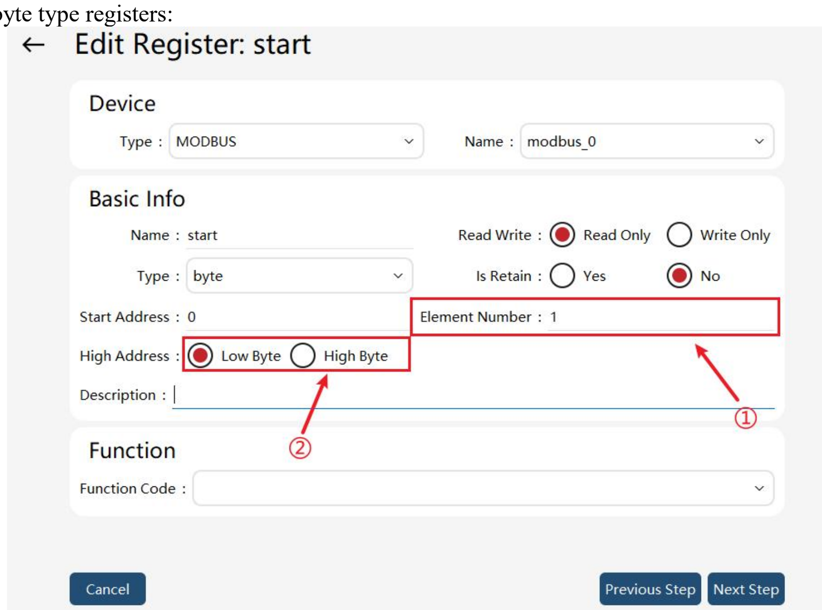
About bit type registers:



①	Element Number.
②	Bit Bias. As shown in the figure above, if the element number is 1, it indicates that a certain bit of a register is occupied. The number of bit biases can be set, with optional values ranging from 1 to 16.

When the element number of the bit type register is greater than 1, i.e. the bit variable array, it is not allowed to set the bit bias and perform function binding.
When the input of the element number in a bit type register is greater than 1, the bias option is automatically hidden, and the offset is set to 1.

About byte type registers:



①	Element Number.
②	Byte address.

As shown in the figure above, if the element number is 1, it indicates that certain 8 bits of a register are occupied, and the byte address can be set,
with optional values range of LSB (1-8) and MSB (9-16).
When the element number of the byte type register is greater than 1, i.e., the byte variable array, the byte address is not allowed to be set, with a default of LSB.
Note: It is not allowed to enable the program through the register when it is equipped with the pause function or system IO has not been reset.

10.5.4Register function code

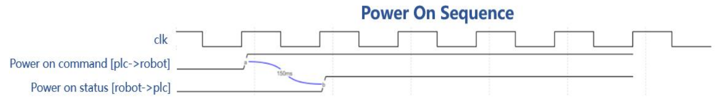
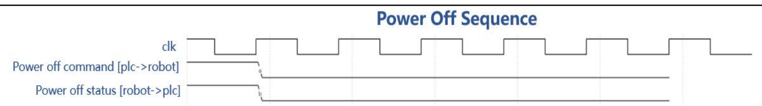
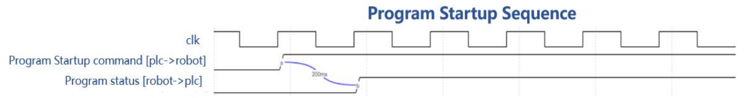
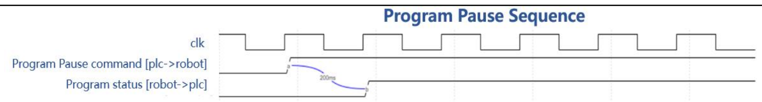
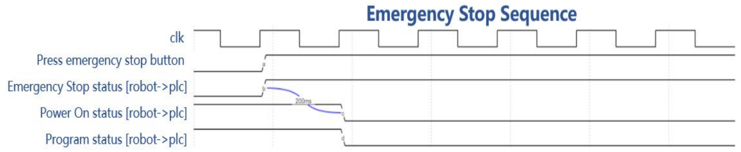
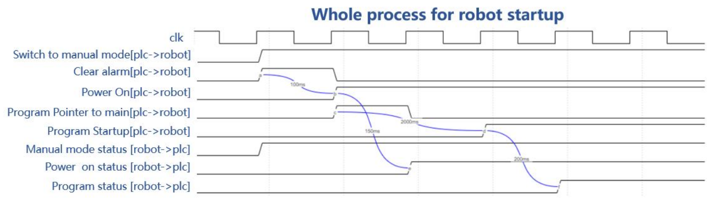
10.5.4.1Read-only function code

The read-only function codes are mostly used for control signals, which are usually sent by external devices to the robot to indicate its actions. For robots, these registers are read-only. Currently supported control signals:

Function Code Name	Supported Binding Types	Function
Blank	N/A	No function, custom input.
ctrl_clear_alarm	bit/bool/byte/int16	Clear servo alarms. Posedge (0→1): Clear alarm; set to 0: Reset.
ctrl_estop_reset	bit/bool/byte/int16	Emergency stop reset. Posedge (0→1): Estop reset; set to 0: Reset.
ctrl_jjwc_A	bit/bool/byte/int16	The trigger type is pulse trigger, which is active at a high level. When the signal 0->1, the robot stops. After triggering by this signal, the robot cannot continue to run, and can only run again after pptomain, pptofunc, or pptocurs. In addition, the sta_jjwc_B signal is set to 1 (high level). When the signal 1-> 0, the sta_jjwc_B signal is set to 0 (low level).
ctrl_motor_off	bit/bool/byte/int16	Execution of power off. Posedge (0→1): Power off; set to 0: Reset
ctrl_motor_on	bit/bool/byte/int16	Execution of power on. Posedge (0→1): Power on; set to 0: Reset
ctrl_motor_on_off	bit/bool/byte/int16	Motor power on or off: 1, power on; 0, power off.
ctrl_motoron_pptomain_start	bit/bool/byte/int16	Power on, Pointer to main, and start program in order. Posedge (0→1): Power off; set to 0: Reset. After triggering the function code, if the teach pendant displays the alarm "Program not synchronized to controller, startup failed", synchronize the program and retrigger the function code.
ctrl_motoron_start	bit/bool/byte/int16	Power on and start program in order. Posedge (0→1): Power off; set to 0: Reset. After triggering the function code, if the teach pendant displays the alarm "Program not synchronized to controller, startup failed", synchronize the program and retrigger the function code.
ctrl_pause_motoroff	bit/bool/byte/int16	Pause program and execution of power off. Posedge (0→1): Power off; set to 0: Reset.
ctrl_pptomain	bit/bool/byte/int16	Program pointer to main. Posedge (0→1): Power off; set to 0: Reset.
ctrl_program_start	bool/byte/int16	Start the RL program. Posedge (0→1): Power off; set to 0: Reset. After triggering the function code, if the teach pendant displays the alarm "Program not synchronized to controller, startup failed", synchronize the program and retrigger the function code.
ctrl_program_start_stop	bit/bool/byte/int16	Program running/stop. Set to 1: Program run; set to 0: Program stop
ctrl_program_stop	bit/bool/byte/int16	Stop the RL program. Posedge (0→1): Power off; set to 0: Reset.
ctrl_set_program_speed	bit/bool/byte/int16	Set program running rate. Input value represents the running rate. Example: Input "10" sets the rate to 10
ctrl_soft_estop	bit/bool/int16	Control the robot's soft emergency stop, 1: not trigger the soft emergency stop; 0: trigger soft emergency stop.
ctrl_switch_auto_motoron	bit/int16/byte/bool	Switch to Automatic mode first, then power on. Posedge (0→1): Power off; set to 0: Reset.
ctrl_switch_operation_auto	bool/byte/int16	Switch to Automatic mode. Posedge (0→1): Switch to Automatic mode; set to 0: Reset.
ctrl_switch_operation_auto_manu	bit/bool/byte/int16	Switch between Automatic mode and Manual mode, set to 1: Automatic mode; set to 0: Manual mode.
ctrl_switch_operation_manu	bit/bool/byte/int16	Switch to the Manual mode. Posedge (0→1): Switch to Manual mode; set to 0: Reset.

enable_safe_region01~enable_safe_region10	bit/bool/byte/int16	Corresponding safe region enabled. Posedge (0→1): Enable safe region; set to 0: Reset
ext_cmd_set	bit/bool/int16	Remote control function: issue commands. See "Remote Control".
ext_request_data	int16 array	Remote control function: command function code. Array, register with a fixed size of 8.
ext_reset	bit/bool/int16	Remote control function: overall function reset. See "Remote Control".
ext_resp_get	bit/bool/int16	Remote control function: Acknowledge and clear the previous command response.
ctrl_estop_reset_and_clear_alarm	bit/bool/byte/int16	Reset E-stop state and clear alarm. Posedge (0→1): Reset E-stop state and clear alarm; set to 0: Reset.
ctrl_reduced_mode	bit/bool/byte/int16	Trigger the robot's reduced mode. Posedge (0→1): Trigger the robot's reduced mode; set to 0: Reset.

Description: All system inputs of the above system registers are pulse-triggered. To ensure that the xCore system receives external commands correctly, please ensure that the pulse width of the external input is not less than 60 milliseconds.

	Preconditions/Precautions	
Motor on	The robot is in automatic mode; The robot has no alarm;	
Motor off		
Program startup	The robot is in automatic mode; There is robot program pointer; The robot is powered on; The robot has no alarm;	
Program pause	The external device sends a program pause through a register (without power-off);	
Emergency stop handling	Trigger the emergency stop signal or external emergency stop sequence logic. (Note: After the emergency stop is pressed, the program will stop, the motor will be powered off, and the pointer will be lost, so it is required to clear the alarm, power on, and move the program pointer to main.)	
Whole process for robot startup	Operation sequence: (1) Switch the robot to manual mode; (2) Clear alarm (clear servo alarm or controller error alarm). (3) Power on the motor; (4) Move the program	

	pointer to main. (The larger the project, the longer time the command pptomain takes. It is recommended to reserve 2s for execution, and send the program startup signal after the command is completed.); (5) Start the program;	

10.5.4.2Write-only function code

The write-only function codes are mostly used for state signals, which refer to the signals sent by the robot to the outside world for feeding back the robot's state, including the power-on state, program state, etc. For robots, a register being write-only indicates that it can be bound to the state signals. The following state signals are currently supported.

Function Code Name	Supported Binding Types	Function
Blank		No function, custom output.
ext_error_code	int16	Remote control function: error code.
ext_resp_set	bit/bool/ int16	Remote control function: response after command execution.
ext_response_data	int16 array	Remote control function: data to be fed back. Array, register with a fixed size of 8.
sta_alarm	bit/bool/byte/int16	Servo alarm status, 1: servo alarm; 0: no alarm.
sta_board_DI0~sta_board_DI3	bit/bool/byte/int16	Real-time output of signal state of self-developed IO board and Solidot IO board.
sta_board_DO0~sta_board_DO3	bit/bool/byte/int16	Real-time output of signal state of self-developed IO board and Solidot IO board.
sta_collision	bit/bool/byte/int16	Collision detection status, 1: collision detected; 0: no collision.
sta_collision_alarm	bool/byte/int16	Collision detection alarm, 1: collision detected; 0: no collision; alarm cleared.
sta_collision_open	bool/byte/int16	Open state of collision detection. 1: Collision detection enabled; 0: Collision detection disabled.
sta_controller_is_running	bool/byte/int16	Running signal of controller: 1: running controller; 0: controller not running.
sta_encoder_low_battery	bool/byte/int16	Low-voltage alarm state of encoder.
sta_error_code	int16	The robot reports an error code, which differs from the error code value in the robot log by 30000. For example, the error code 50002 for the robot log "out of range of motion" was obtained through sta_errorCode as 20002.
sta_estop	bit/bool/byte/int16	EStop state This value is affected by the emergency stop trigger level type setting. When it is set to high level, 1: the current emergency stop is triggered; 0: normal. When it is set to low level, 0: the current emergency stop is triggered; 1: normal.
sta_heartbeat	bit/bool/byte/int16	Heartbeat signal, write-only. Click "Settings –> Controller Settings" and set the heartbeat cycle.
sta_home	bit/bool/byte/int16	Whether each joint of the robot is at the Home point, 1: at the Home point; 0: not at Home point.
sta_jjwc_B	bit/bool/byte/int16	Real-time state output. Trigger action: passive trigger. When the ctrl_jjwc_A signal is 0->1, the sta_jjwc_B signal is set to 1. When the ctrl_jjwc_A signal is 1->0, the sta_jjwc_B signal is set to 0.
sta_motor	bit/bool/byte/int16	Motor power on status, 1: powered on; 0: not powered on.

sta_operation_mode	bit/bool/byte/int16	Current operating mode, 1: Automatic mode; 0: Manual mode.
sta_program	bit/bool/byte/int16	Whether it is currently in the program running state, 1: program running; 0: free.
sta_program_full	byte/int16	RL pause state, 0: initialization state; 1: RL running; 2: HMI pause; 3: System IO pause; 4: Register function code pause; 5: External communication pause; 6: SDK pause; 7: Pause command pause; 10: Emergency stop; 11: Safety door; 12: Pause for other factors.
sta_program_not_run	bool/byte/int16	Non-execution of RL program, 1: non-execution of RL program; 0: execution of RL program.
sta_program_reset	bool/byte/int16	Program reset success signal; the output is 1 when the program pointer is on the first line of the main function, otherwise, the output is 0.
sta_program_speed	int16	Query the current program running speed (in percentage terms).
sta_robot_is_busy	bit/bool/byte/int16	Whether the current robot is performing time-consuming operations such as pptomain, 1: performing; 0: free.
sta_robot_moving	bit/bool/byte/int16	Whether the robot is in motion, 1: the robot is in motion; 0: the robot is stationary. Only detect the robot's motion status when detecting motion commands and Jog in the RL program. (Note: In identification, dragging, force control, and drag playback, even if the robot is in motion, the output is still stationary.)
sta_safe_door	bit/bool/byte/int16	The register-bound state signal output is valid when the safety gate is opened, but invalid when the safety gate is closed (active at high level, but inactive at low level).
sta_safe_jnt_pos1~sta_safe_jnt_pos8	bool/byte/int16	Safety position triggering state, 1: safety position reached; 0: safety position unreached.
sta_safe_region01~sta_safe_region10	Bool/byte/int16	Safe region triggering state. 1: Safe region triggered.
sta_soft_estop	bit/bool/int16	Output of soft emergency stop status. This status value is affected by the emergency stop trigger level type setting. When it is set to high level, the status value is 1 when triggering a soft emergency stop, and 0 when not triggered; when it is set to low level, the status value is 0 when triggering a soft emergency stop, and 1 when not triggered.
sta_cart_pose	float array	Query the current Cartesian pose of the robot. Requirements for bound registers: float array, size - 8.
sta_cart_vel	float array	Cartesian speed of robot.
sta_jnt_pose	float array	Query the current joint angle of the robot. Requirements for bound registers: float array, size - 8.
sta_jnt_trq	float array	Query the current joint torque of the robot. Requirements for bound registers: float array, size - 8, unit: N.m.
sta_jnt_vel	float array	Query the current joint velocity of the robot. Requirements for bound registers: float array, size - 8, unit: rad/s.
sta_tcp_pose	float array	Pose of the robot TCP. Requirements for bound registers: float array, size - 7.
sta_tcp_vel	float array	Velocity of the robot TCP. Requirements for bound registers: float array, size - 7.
sta_tcp_vel_mag	float	Robot TCP resultant linear velocity.
sta_ext_estop	Bool/byte/int16	External estop state 1: external emergency stop state; 0: non-external emergency stop state The safeboard is a mini board, and the firmware version is not less than 1.0.8.7
sta_hand_estop	bool/byte/int16	Handheld estop state 1: external emergency stop state; 0: non-external emergency stop state The safeboard is a mini board, and the firmware version is not less than 1.0.8.7
sta_sys_stop_di	bit/bool/byte/int16	Output the value of the signal bound to the system pause. "sta_sys_stop_di" outputs 1 when either the "Pause Program" or "Pause Program 1" signal is triggered, and outputs 0 when neither of the two signals is triggered.
sta_reduced_mode	bit/bool/byte/int16	Whether the robot is currently operating in reduced mode. 1: In

		reduced mode.
sta_on_path	bit/bool/byte/int16	Whether the robot's current position is on the preset trajectory 1: Yes, e.g., during program execution or when paused, the robot remains on the preset path. 0: No, e.g., path deviation during JOG, clicking pptomain, reloading the project, or after pptocurs repositioning. Notes: 1. When the emergency stop button is pressed, due to servo oscillation and power-off jitter, the controller determines that the robot has deviated from the preset trajectory, and therefore the sta_on_path function code value is 0. 2. When the safety gate is opened and a safety stop is triggered, the sta_on_path function code value becomes 0.
sta_near_path	bit/bool/byte/int16	Whether the robot's current position is near the preset trajectory 1: Yes, if both the path deviation sphere radius and path deviation sphere angle conditions are satisfied. 0: No, for all other cases. The path deviation sphere radius and path deviation sphere angle parameters are configured in Settings → Controller settings → Advanced settings. When both the path deviation sphere radius and path deviation sphere angle are set to 0, this function code is equivalent to sta_on_path

10.5.5RL read/write register example

The control system reads and modifies the registers in two ways: command or assignment.
Command provides WriteRegByName and ReadRegByName. Assignment is more intuitive and simple, using the operator "=".

10.5.5.1Command

WriteRegByName(modbus_reg[index], rl_symbol)
Modbus-reg is the register name configured in "Communication" -> "Register", which can be offset at the first address of the corresponding register using [index]. The index range is [1, maximum register size], and the default index = 1.

The data in the control system can be output to its bound devices through registers.
For example, "int rl_value" is defined in the control system. If you want to output it to an external device, you can specify a register, such as the first register of "mtcp_wo_i", and add a WriteRegByName command in the RL language. The value will be sent to the external device associated with "mtcp _ wo _ i".

```
GLOBAL PROC main()
  int rl_value =0;
  while(true)
//Production cycle
//Command for controlling the robot's actions
//...
//...


//Count plus one
    rl_value+=1;
    WriteRegByName(mtcp_wo_i[1],rl_value);
  endwhile
```

①

ReadRegByName(modbus_reg[index], rl_symbol)
This command is similar to WriteRegByName, which updates the value of a register to the RL program variable. For example, it is used to control the execution process and motion parameters of RL programs.

10.5.5.2Assignment

Directly use the operator "=". For example, "mtcp_wo_i[1] = 1" is to update the value of the first element of the register mtcp_wo_i to 1. Similarly, "a = mtcp_wo_i[1]" is to update the value of the first element of the register mtcp_wo_i to the variable a of the RL program.

10.5.6Register remote control

Remote control is a combination function performed with registers of 7 different functions. It is used to achieve complex business logic interactions in a specific sequence. External devices can fulfill functions such as robot Jog, updating point position, obtaining robot position and status, etc. via the remote control function.

Register function
External devices use four types of registers to control the robot. These registers are read-only for the robot.

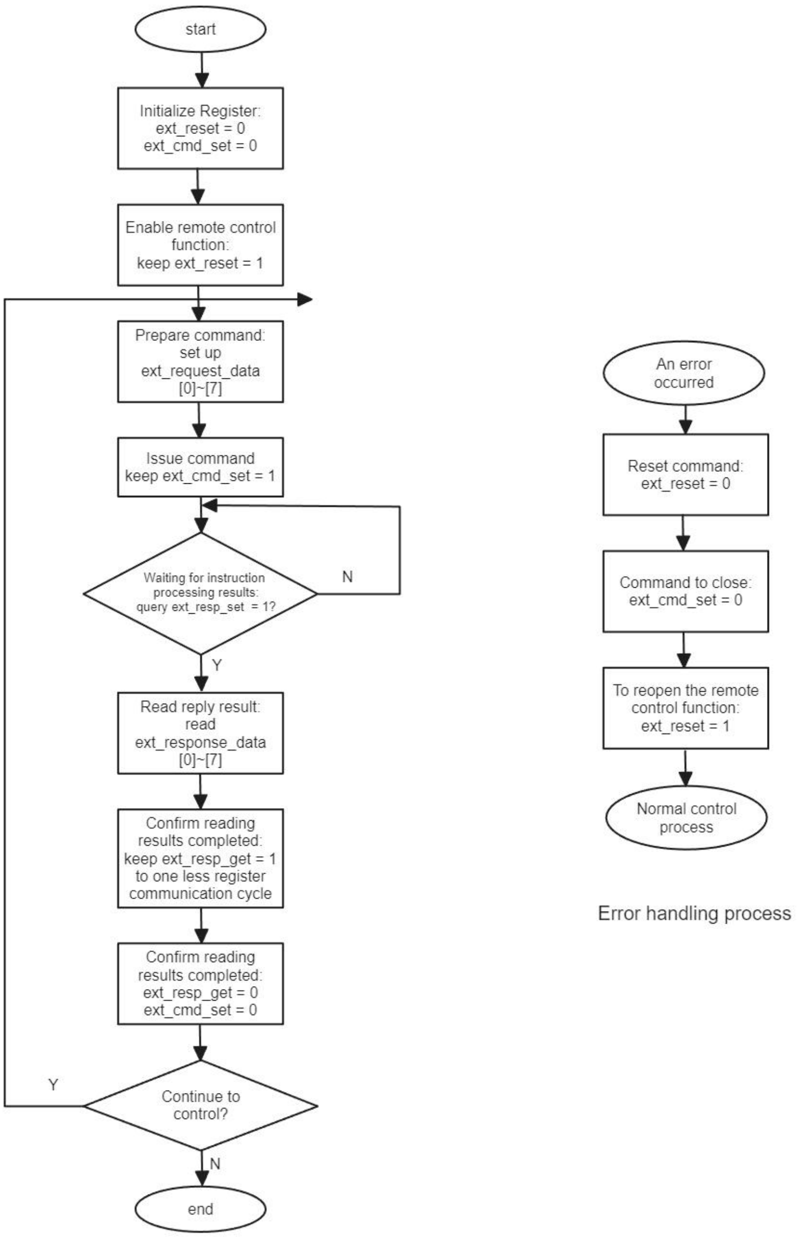
Function Code Name	Attribute	Type	Length	Function
ext_cmd_set	Read-only	int16/bool/bit	1	Issuing commands: 1. Set ext_cmd_set to 1 to send a request for command execution. The request is responded only when ext_cmd_set is set to 1. 2. To avoid misoperation, be sure to set the command data to the data area before execution. (The command data is temporarily stored in the cache and is responded only when ext_cmd_set is 1). 3. After the command is executed, clear ext_cmd_set (set it to 0).
ext_reset	Read-only	int16/bool/bit	1	Function reset: 1. The signal is used to enable the remote control function. Always keep the register state at 1 when using the function. 2. The function stops when the register state is 0. 3. The signal is also used for commands to reset or interrupt the action when the interface function is abnormal.
ext_resp_get	Read-only	int16/bool/bit	1	Acknowledge and clear the previous command response, and reset ext_resp_set to 0.
ext_request_data	Read-only	int16	8	Command function code. Array, register with a fixed size of 8. For details, refer to the introduction in the function code section.

External devices use three types of registers to obtain the robot status. These registers are write-only for the robot.

Function Code Name	Attribute	Type	Length	Function
ext_error_code	Write-only	int16	1	Remote control function: error code.
ext_resp_set	Write-only	int16/bool/bit	1	After responding to the control command, the robot sets the register to 1, indicating that the command is executed.
ext_response_data	Write-only	int16	8	Remote control function: data to be fed back. Array, register with a fixed size of 8.

10.5.6.1Procedure

The combined use of 7 types of registers and control flow are shown in the figure below.



Error handling process

Normal control process

10.5.6.2Command format

Commands and responses are implemented with 8 registers individually.

The command signal ext_request_data (eight registers occupied: reg0 - reg7) is used to specify the data area of the commands and relevant parameters. A command consists of multiple characters:
1) Character: a 16-bit register.
2) Command format: a command consists of up to 8 characters and varies with the command. The shortest command consists of 1 character.

Command No.	Command No. 1	Command No. 2	……	Command No. 7

The response signal ext_response_data (eight registers occupied: reg0 - reg7) is used to obtain the data area of the responses. A response consists of multiple characters:
1) Character: a 16-bit register.

Command No.	Response No. 1	Response No. 2	……	Response No. 7

2) Response format: a response consists of up to 8 characters. and varies with the received command. The shortest response consists of 1 character. However, an abnormal response always occupies 3 characters.

The available command numbers are shown in the table below:

Command Type	Explanation	Command Code	Command Length	
			Command	Response
JOG	Set Jog space	1	2	3
	Obtain Jog space	2	1	4
	Set Jog speed	3	2	3
	Obtain Jog speed	4	1	4
	Set Jog step length	5	2	3
	Obtain Jog step length	6	1	4
	Start Jog	7	4	2
	Stop Jog (without parameters)	8	1	2
	Update point position	9	2	2
	Move to point position	10	2	2
Set information	Set tools	11	2	3
	Obtain current tool id	12	1	4
	Set work object	13	2	3
	Obtain current work object id	14	1	4

10.5.6.3Command description

(1) Set Jog space:

Command/Reply	Command Code	Parameter 1	Parameter 2
Set Jog space	1	Frame: 1: Joint space 2: World frame 3: Flange frame 4: Base frame 5: Tool frame 6: Work object frame	N/A
Reply	1	Result: 0 - Succeed; 1 - Fail.	Error code

(2) Obtain Jog space:

Command	Command Code	Parameter 1	Parameter 2	Parameter 3
Obtain Jog space	2	N/A	N/A	N/A
Reply	2	Result: 0 - Succeed; 1 - Fail	Error code	Frame: 1: Joint space 2: World frame 3: Flange frame 4: Base frame 5: Tool frame 6: Work object frame

(3) Set Jog speed:

Command/Reply	Command Code	Parameter 1	Parameter 2
Set Jog speed	3	Jog speed (1−100)	N/A
Reply	3	Result: 0 - Succeed; 1 - Fail	Error code

(4) Obtain Jog speed:

Command	Command Code	Parameter 1	Parameter 2	Parameter 3
Obtain Jog speed	4	N/A	N/A	N/A
Reply	4	Result: 0 - Succeed; 1 - Fail	Error code	Jog speed (1−100)

(5) Set Jog step length:

Command/Reply	Command Code	Parameter 1	Parameter 2
Set Jog step length	5	1: Continuous 2: 10 mm step length 3: 1 mm step length 4: 0.1 mm step length 5: 0.01 mm step length	N/A
Reply	5	Result: 0 - Succeed; 1 - Fail	Error code

(6) Obtain Jog step length:

Command	Command Code	Parameter 1	Parameter 2	Parameter 3
Obtain Jog step length	6	N/A	N/A	N/A
Reply	6	Result: 0 - Succeed; 1 - Fail	Error code	1: Continuous 2: 10 mm step length 3: 1 mm step length 4: 0.1 mm step length 5: 0.01 mm step length

(7) Start Jog:
The command is dependent on command code 1: set Jog space. In joint space, the value of parameter 1 represents the joint number (J1–J7: 1 for J1, ..., 7 for J7); in Cartesian space, it represents the (x, y, z, a, b, c, and elb) number (1 for x, ..., 7 for elb).

Command/Reply	Command Code	Parameter 1	Parameter 2
Start Jog	7	Operation mode: Joint space – representing joint number; Cartesian space – representing (x, y, z, a, b, c, and elb)	Jog direction: 1: negative 2: positive
Reply	7	Result: 0 - Succeed; 1 - Fail	Error code

(8) Stop Jog:

Command/Reply	Command Code	Parameter 1
Stop Jog	8	N/A
Reply	8	Result: 0 - Succeed; 1 - Fail.

(9) Update point position:

Command/Reply	Command Code	Parameter 1	Parameter 2
Update point position	9	Number in the RL project point list	N/A
Reply	9	Result: 0 - Succeed; 1 - Fail	Error code

(10) Move to point position:

Command/Reply	Command Code	Parameter 1	Parameter 2
Move to point position	10	Motion mode: 1: MoveAbsj; 2: MoveJ; 3: MoveL	Number in the RL project point list
Reply	10	Result: 0 - Succeed; 1 - Fail	Error code

(11) Set current tool:

Command/Reply	Command Code	Parameter 1	Parameter 2
Set current tools	11	Number in the RL project tool list	N/A
Reply	11	Result: 0 - Succeed; 1 - Fail	Error code

(12) Obtain current tool id:

Command/Reply	Command Code	Parameter 1	Parameter 2	Parameter 3
Obtain current tool id	12	N/A	N/A	N/A
Reply	12	Result: 0 - Succeed; 1 - Fail	Error code	Current tool id

(13) Set current work object:

Command/Reply	Command Code	Parameter 1	Parameter 2
Set current work object	13	Number in the RL project work object list	N/A
Reply	13	Result: 0 - Succeed; 1 - Fail	Error code

(14) Obtain current work object id:

Command/Reply	Command Code	Parameter 1	Parameter 2	Parameter 3
Obtain current work object id	14	N/A	N/A	N/A
Reply	14	Result: 0 - Succeed; 1 - Fail	Error code	Current work object id

10.5.6.4Error code

During command configuration, parameter errors, robot status mismatch, or other conditions may lead to configuration failure. Error codes can be used to check the robot's problems in this case.

The control system has three types of error codes:
ext_response_data: error code of command execution results.
ext_error_code: The command cannot be executed, for example, the robot is busy, or the remote control flag bit is incorrect, etc.
sta_error_code: the robot error code. Read the register when an error occurs during Jog.

Normally, the error code should be used according to the following steps:
After sending the execution command (ext_cmd_set=1), first read ext_error_code. If there is no error code, read the return value of ext_response_data. If the return value is not zero, read the error code of ext_response_data.
For motion operations (Jog and move to point position), if the above return values are both 0, read sta_error_code to see if there is a stop in the motion caused by an error (such as singularity and overrun).

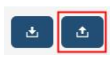
ext_error_code description:

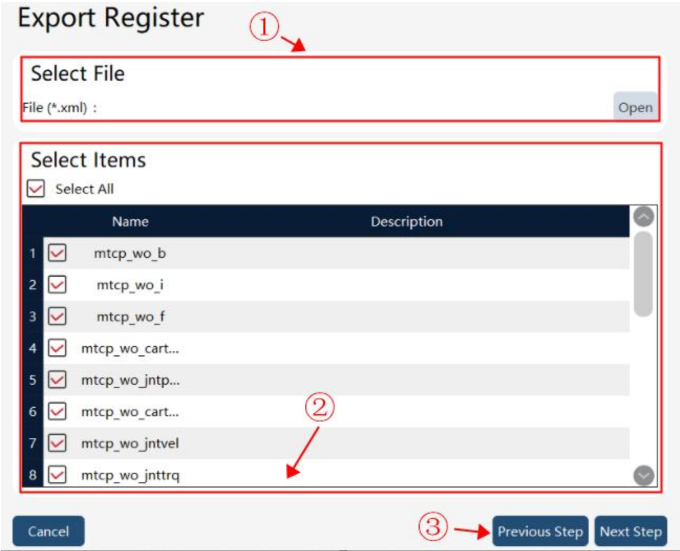
Error code	Meaning	Remarks
01	Unsupported command	
02	Invalid parameter	
03	Incorrect control flag bit	Check whether ext_resp_set is 0 or 1.
04	Robot busy	The robot is executing a command and is forbidden to respond to others.
05	No corresponding number found	Tool, point position, and work object id
06	Unmatched point type and motion type	The point type does not match the motion type for the "Move to point position" command. For example, only the MoveAbsJ command can be used for joint space points, and only the MoveJ or MoveL command can be used for Cartesian space points.
07	Unmatched number of axes entered and model	
11	Incorrect Manual/Auto Mode	
12	Incorrect robot status. Please check if the robot is in Jog Mode.	The robot can only be jogged in Jog Mode and can not be jogged in non-Jog Mode such as Drag Mode.
13	Incorrect power-on status	The robot can only be jogged when powered on.
14	The robot is in non-position mode and can not be jogged	Similar to error code 12.
15	Report algorithm error when unable to start Jog	The error is reported when the robot cannot be jogged for various reasons.
20	Encounter singularity	
21	Moved to target point	If the robot moves to a point it has reached earlier, an error occurs.

10.5.7Register import and export

The register import and export function can quickly copy register configurations from one robot to another robot without reconfiguring registers.

10.5.7.1Register export

On the register page, click the button in the bottom left corner  to enter the register export interface

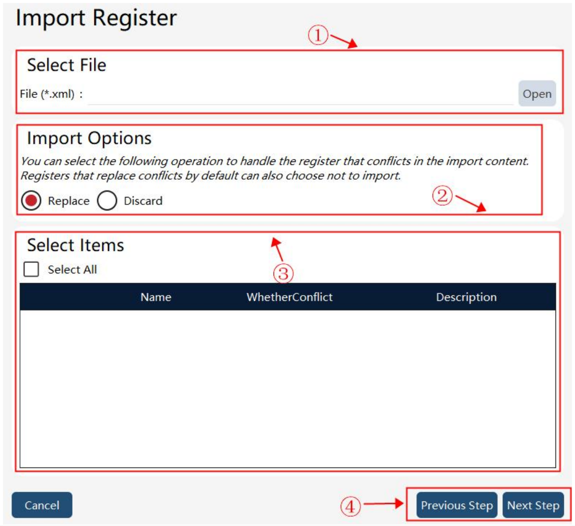


①	Register export target file (including path).
②	Register export selection list.
③	Register export operation button.

Export steps: First, enter or select the register export target file path in ①, then check the register to be exported in the register list in ②, and finally click the "Next" button in ③ to execute the export. The exported register file can be generated under the corresponding path in ①.

10.5.7.2Register import

On the register page, click the button in the bottom left corner  to enter the register import interface



①	Register import file.
②	Register import options, which are strategies for handling conflicting items.
③	Register import selection entry.
④	Register import operation button.

Import steps: First, select the register file to be imported in ①, then set the conflicting register strategy in ②, then select the register to be imported in ③, and finally click the "Next" in ④ to perform the import to import the selected register to the local machine.

10.5.7.3Conflict checking during register import

The same device and register properties (read and write) cannot have the same register address. If the same, if the import option is set to not import, the original register will prevail, and conflicting registers will not be imported; if the import option is set to auto replace, the newly imported register will prevail, and the conflicting register will be overwritten. A pop-up window will prompt the user to choose whether to replace the current register.

When creating the 7 registers starting with ext: ext_cmd_set, ext_resp_set, ext_resp_get, ext_reset, ext_response_data, ext_request_data, and ext_error_code, if the register has been bound by the register address, these addresses cannot be bound by another register. When importing the above 7 registers, if the function codes have already been bound in the HMI, and the newly imported register list also involves such function codes, the newly imported ones will prevail, and the original conflicting register will be overwritten. A pop-up window will prompt the user to choose whether to replace the current register.

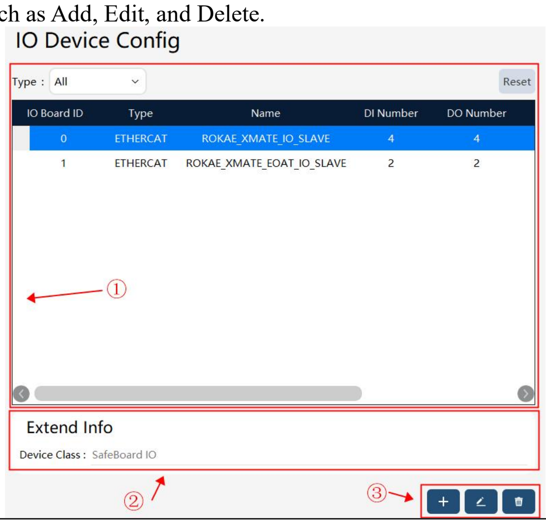
10.6IO device

10.6.1Overview

IO devices support four signal types: DI, DO, AI, and AO. Signal sources include: controller cabinet built-in, EtherCAT expansion, and field bus expansion. For industrial robots, the controller cabinet has several built-in DIs and DOs. For cobots, the base and the end-effector have several built-in DIs and DOs. For industrial robots, the EtherCAT expansion interfaces are reserved on the controller cabinet to connect EtherCAT expansion modules to generate new DI, DO, AI, and AO. The Modbus bus expansion can also be configured with IOs.

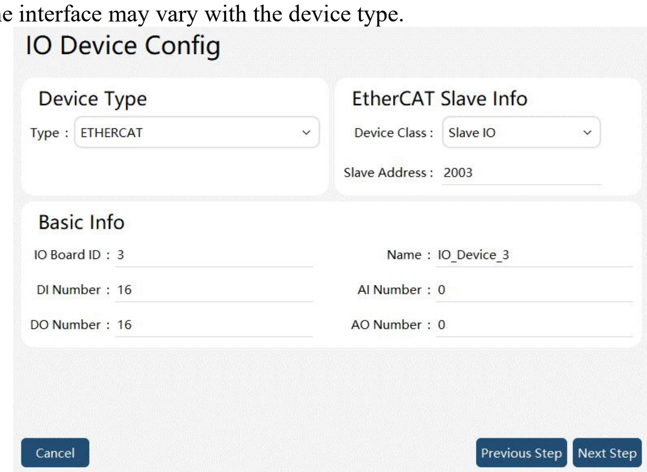
10.6.2Parameter configuration

You can view all current IO devices on the "Communication" -> "IO Device" page, and perform operations on them such as Add, Edit, and Delete.

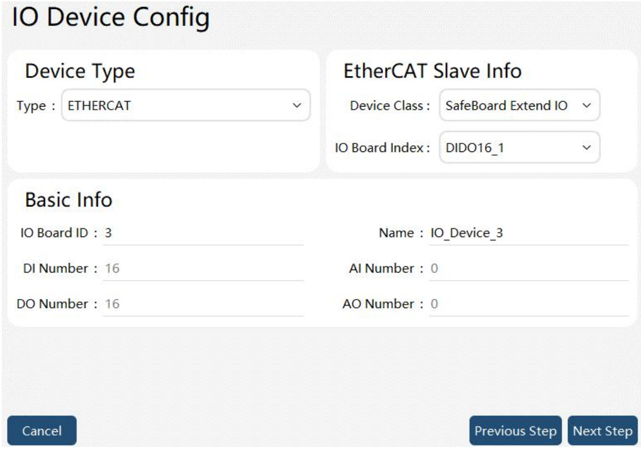


①	List of IO device.
②	IO device extension attributes.
③	The IO device operation buttons, from left to right, are Create, Edit, and Delete.

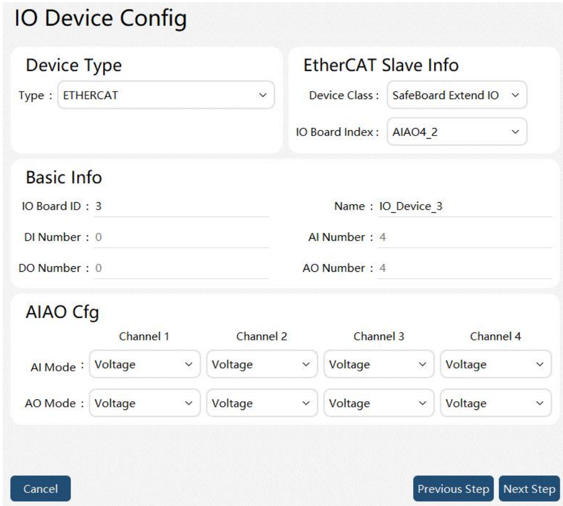
Click the Create button in ③ on the IO device configuration page in the above figure to enter the IO device (including ETHERCAT, FIELDBUS, and ROKAE _IO devices) configuration interface. The parameters on the interface may vary with the device type.



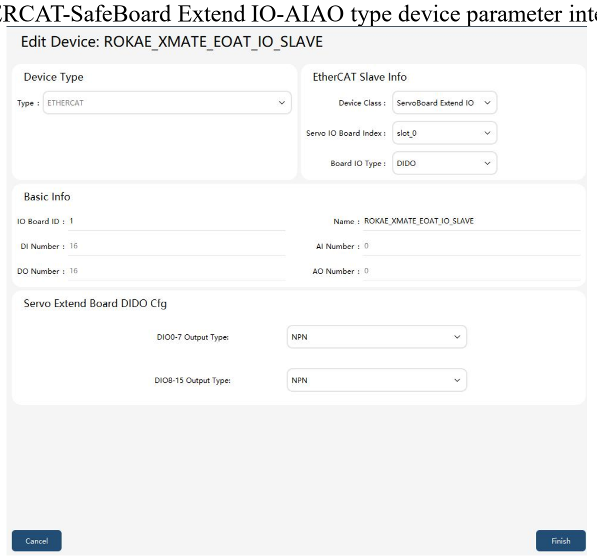
ETHERCAT-Slave IO type device parameter interface



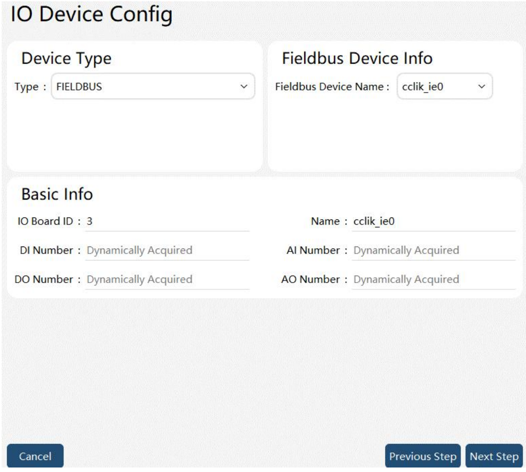
ETHERCAT-SafeBoard Extend IO-DIDO type device parameter interface



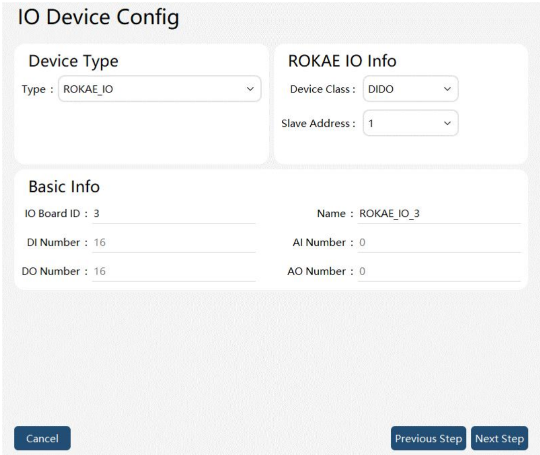
ETHERCAT-SafeBoard Extend IO-AIAO type device parameter interface



ETHERCAT-ServoBoard Extend IO-DIO type device parameter interface



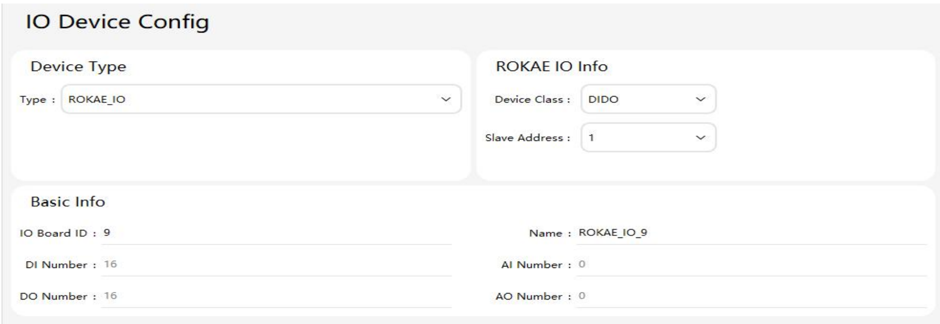
FIELDBUS type device parameter interface



I/O device configuration parameters interface

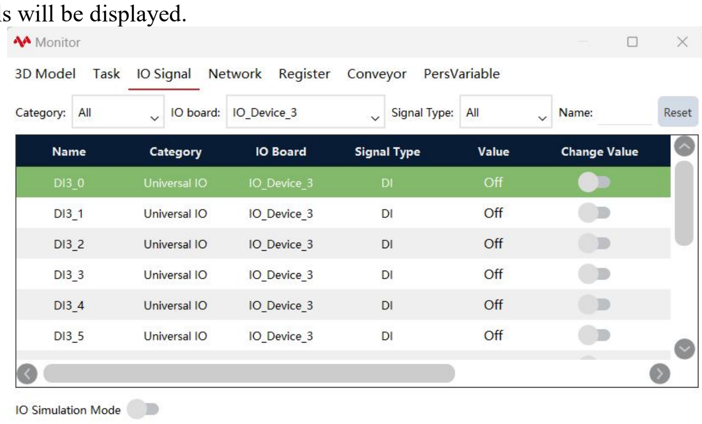
Parameter	Explanation
Device type	EtherCAT and FIELDBUS are optional. EtherCAT refers to IO expansion with the EtherCAT bus and expansion modules. The expansion modules can only serve as slaves, and the slave address needs to be configured.
EtherCAT slave information - Device type	SafeBoard IO, SafeBoard Extend IO, xPanel IO, Slave IO, and ServoBoard Extend IO are optional. SafeBoard IO refers to the DI and DO on the robot safeboard. SafeBoard Extend IO refers to the expansion IO on the robot safeboard, generally the expansion IO on the safeboard in the XBC_5 controller cabinet. xPanel IO refers to the DI and DO of the end-effectors of the cobots. Slave IO refers to the EtherCAT expansion module. ServoBoard Extend IO refers to the extended IO of the robot servo in the XBC_6 control cabinet.
EtherCAT slave information - IO board type	When SafeBoard Extend IO is selected in EtherCAT slave information - Device type, the option IO board type appears for selection of the safeboard expansion IO board. DIO16_1, DIO16_4, DIO16_5, DIO16_6, AIAO4_2, and AIAO4_3 are optional. The last digit of the option refers to the address of the safeboard expansion IO board. Select the safeboard expansion IO board that is actually connected. In addition to manual editing by the user, the controller can automatically identify and add the safeboard expansion IO board. When ServoBoard Extend IO is selected in EtherCAT slave information - Device type, the servo IO board position option will appear. Slots 1-6 correspond to the actual expansion slot positions on the control cabinet. Currently, only DIO is supported for the IO board type. In addition to manual editing by the user, the controller can automatically identify and add servo extended IO during startup.
EtherCAT slave information - Slave address	The slave address of the expansion module in the EtherCAT bus topology. It should not conflict with the address of the safeboards, joints, or the cobot end-effectors.
FIELDBUS - Bus device name	The custom name when a bus connection is created on the Bus Device page. It is used to associate with the Bus Device.
ROKAE_IO device type	DIDO and AIAO are optional, corresponding to the digital and analog IO

	boards developed by ROKAE. Please note that it shall be set according to the actual IO board.
ROKAE_IO slave address	It refers to the address of self-developed IO board accessed and ranges from 1 to 15 without repetition. Please note that it shall be set according to the dialing address of the actual IO board, otherwise, there may be an abnormal state.
IO board serial number	A virtual IO board is generated for each IO device configuration for the control system to classify and manage the IO boards internally. The IO board serial number is the unique number for virtual IO board management.
Name	The custom name of the virtual IO board. It is used for filtering in Status Monitoring -> IO Signal.
Number of digital inputs	Number of DIs.
Number of digital outputs	Number of DOs.
Digital IO Configuration	When ServoBoard Extend IO is selected in EtherCAT slave information - Device type and the IO board type is set to DIO, the digital IO configuration option will appear. The output type of DIO0-7 and DIO8-15 can be configured as either NPN or PNP. Note: This configuration requires a controller restart to take effect.
Number of analog inputs	Number of AIs.
Number of analog outputs	Number of AOs.
Analog IO Configuration	When SafeBoard Extend IO is selected in EtherCAT slave information - Device type and AIAO4_2 or AIAO4_3 is selected in IO board type, the option Analog IO Configuration appears. Each analog channel can be configured as voltage type or current type.



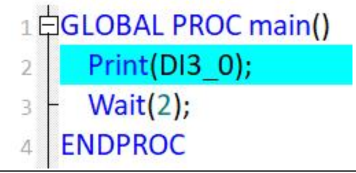
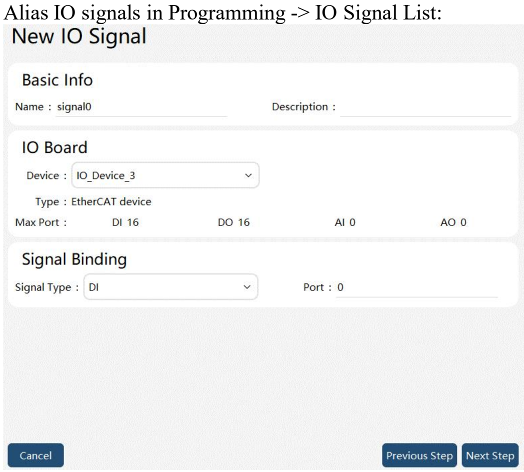
ROKAE_IO type device parameter interface

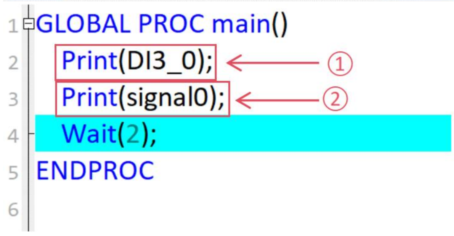
Monitor the created DI, DO, AI, and AO in Status Monitoring -> IO Signal. The IO signals can be filtered by Virtual IO Board Name. Only the IO signals currently configured on the virtual IO board will be displayed. You can also filter the signals by signal type. Only a certain type of DI, DO, AI, and AO signals will be displayed.



After the virtual IO board is configured, a default name will be generated for the IO signal. There are two ways to use RL: one is to use the default name; the second is to alias and use a new name in RL.

Use the default name	The board IO_Device_3 generates the DI3_X IO signals by default. As shown in the figure, DI3_0 is processed directly in the RL program.

<table>
<tr><td></td><td>
</td></tr>
<tr><td>Alias</td><td>Alias IO signals in Programming -> IO Signal List:


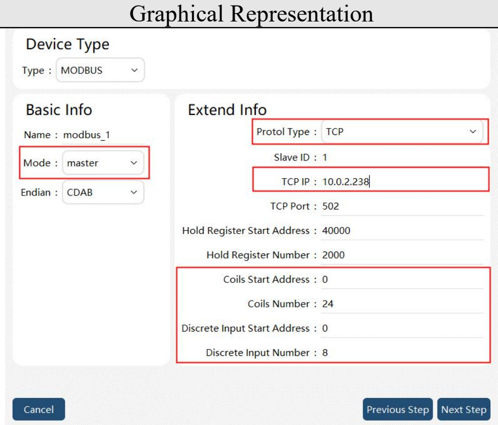


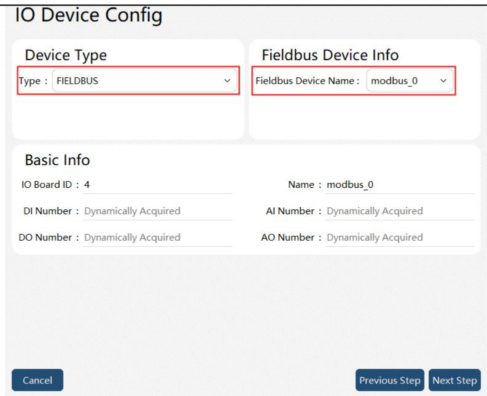
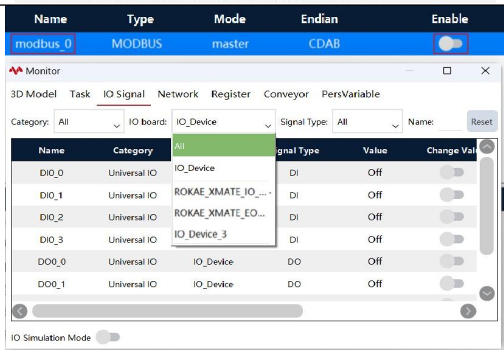
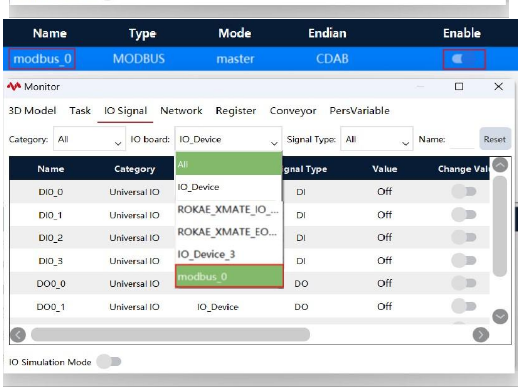
The statements ① and ② in the above figure have the same effect.</td></tr>
</table>

10.6.3Modbus expansion IO example

When real IO signals are required to interact with external devices, it is recommended to use an adapter module, which is connected to the control cabinet. You can contact ROKAE to obtain recommended Modbus IO modules. The module is a Modbus TCP slave and controls the robot through the coil function. The robot needs to be configured as a Modbus Master with the coil function enabled.

According to the configuration method of the field bus and expansion IOs, configure Bus Device first and then configure IO Device when using Modbus expansion module to expand real IOs.

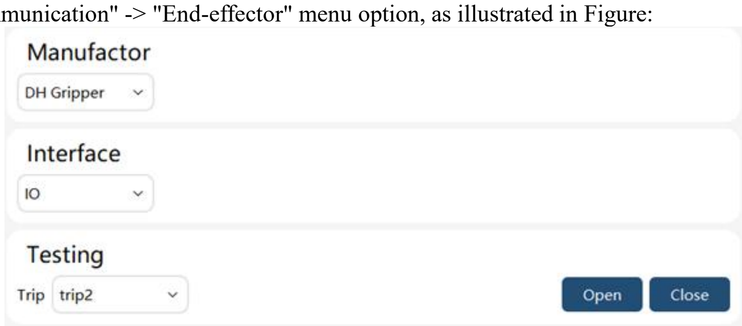
<table>
<tr><th>Step</th><th>Graphical Representation</th><th>Explanation</th></tr>
<tr><td>1. Configure Bus Device.</td><td>
</td><td>In master-slave mode, select "Master" to use the robot as a master.
Slave ID represents the slave ID of the Modbus IO module.
When the robot serves as the master, the IP is filled in with the IP of the Modbus IO module, and the port number is filled in with the port number of the Modbus IO module.
Next: After the parameters are configured, click "Next" to complete the configuration.</td></tr>
</table>

2. Configure IO Device Configuration.		Select Device Type as "FIELDBUS". Bus device name: Select the device name defined in the Bus Device configuration. Basic information: Digital I/O and analog I/O configured in Bus Device configuration will be obtained automatically. No configuration is required.
3. Enabling function and status monitoring.	 	After the Bus Device and IO Device Configuration are just configured, the configured expansion IOs will not be displayed in the status monitoring because the bus connection is not enabled yet. To use these IOs, enable the bus connection to correctly establish connection and communication between the controller cabinet and the expansion IO modules. As shown in the left figure, after the bus device is enabled, the connection between the controller and the expansion modules is established and works properly. Now the IO of the bus device modbus_0 is displayed in the status monitoring of the IO signals, and the number of DIs and the total number of DIs and DOs remains the same as the bus device configuration.

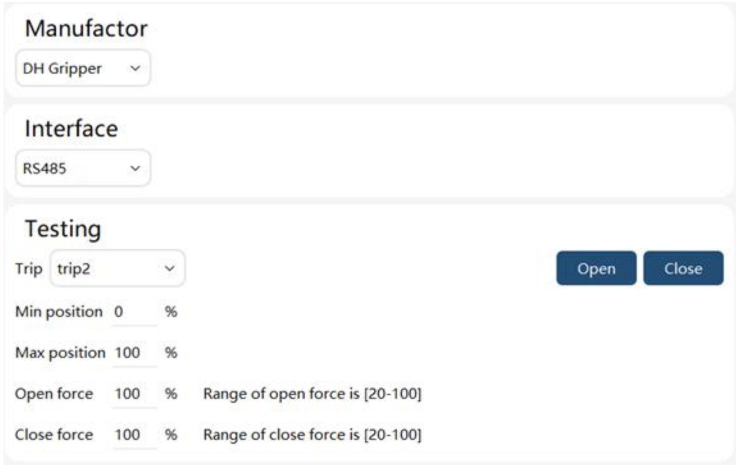
10.7End-effector

The xCore control system enables the manipulation of DH grippers via end-effector, with the end-effector interface supporting IO communication and RS485 communication. This functionality is exclusively applicable to the collaborative robot xMate ER series.

On the HMI main interface, access to the End-effector settings interface can be obtained through the menu "Communication" -> "End-effector" menu option, as illustrated in Figure:



Configuration of end-effector IO ports



Configuration of end-effector RS485 ports

The relevant parameter settings are explained as follows:

Parameter setting	Value/Description
Manufacturer	Choose DH grippers.
Interface	Communication protocol, optional controllable via IO or RS485.
Path	It includes two sets of travel attributes trip1 and trip2, which contain the opening/closing position and force.
Maximum position	Maximum opening position, unit: percentage. Only RS485 is supported.
Minimum position	Minimum opening position, unit: percentage. Only RS485 is supported.
Supporting force	The force used when the gripper is opened, unit: percentage. Only RS485 is supported.
Gripping force	The force used when the gripper is closed, unit: percentage. Only RS485 is supported.

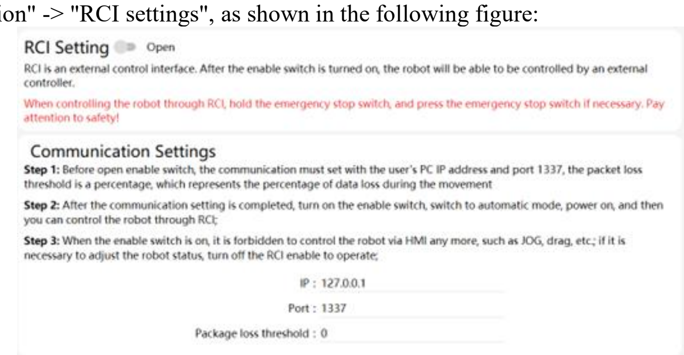
After setting the parameters, you can use the "Open" and "Close" buttons to turn the gripper on or off.

Note: RS485 supports setting of the gripper trip parameter. For IO control, the trip parameters can only be set through the DH communication adapter.

10.8RCI settings

RCI is an external control interface, and the RCI communication setting is required before use. Only collaborative models support this feature.

On the HMI main interface, you can enter the RCI settings interface through the menu "Communication" -> "RCI settings", as shown in the following figure:



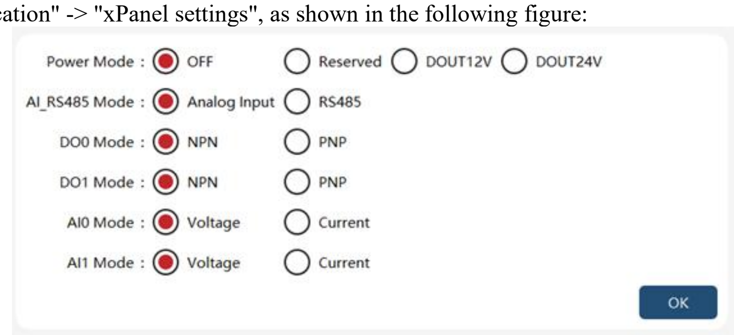
The parameters that need to be set are shown in the table below:

Parameters	Explanation
IP	If the user PC is directly connected to the robot via network cable, the IP address of the user PC should be in the same network segment as the IP of the robot; if the user PC is connected to the robot via wireless or router, the user PC should be in the same LAN as the robot.
Port	The port number is set to 1337 by default.
Packet loss threshold	The packet loss threshold is in percentage, which represents the packet loss rate during RCI communication. For example, when the packet loss threshold is set to 10, it means that the packet loss rate during RCI usage should not exceed 10%. The packet loss threshold is recommended to be set between 10–20.

Refer to the *RCI User Manual* for detailed RCI usage and routines.

10.9xPanel settings

xPanel settings are available to set the mode of the robot's end-effector, which is only applicable to the xMate CR and SR collaborative robots.
On the HMI main interface, you can enter the xPanel settings interface through the menu "Communication" -> "xPanel settings", as shown in the following figure:



The parameters that can be set are shown in the table below:

Parameters	Explanation
External power supply mode	Set the external power supply mode at the end
Analog input or RS485 mode selection	Choose to use analog input or RS485 serial port at the end
Digital output DO0 mode	Set the corresponding terminal DO output to NPN or PNP mode
Digital output DO1 mode	Set the corresponding terminal DO output to NPN or PNP mode
Analog input AI0 mode	Set the corresponding analog input to voltage or current type
Analog input AI1 mode	Set the corresponding analog input to voltage or current type

After setting the required parameters, click the "OK" button, and the settings will take effect.
Note: The voltage or current type of the external analog input signal should be consistent with the corresponding analog input mode, otherwise, unexpected errors may be caused.

10.10Electric gripper and suction cup

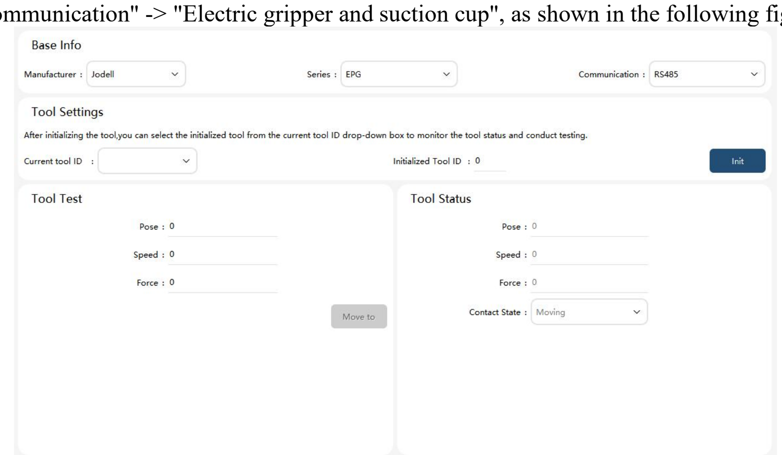
10.10.1Overview

xMate CR, ER, and SR robots support end RS485 communication and are currently compatible with multiple electric grippers and suction cups. This interface is mainly used for configuring and testing adapted electric grippers and suction cups.
Note:

- The function is only applicable to xMate ER, CR, and SR robots;
- The old version of the robot's end board may not be compatible with this function. Please contact the ROKAE to upgrade the board firmware;
- Before using the xMate CR model, please confirm that the end parameters of the xPanel are configured correctly;

10.10.2Configurations

On the HMI main interface, you can enter the Electric gripper and suction cup interface through the menu "Communication" -> "Electric gripper and suction cup", as shown in the following figure:



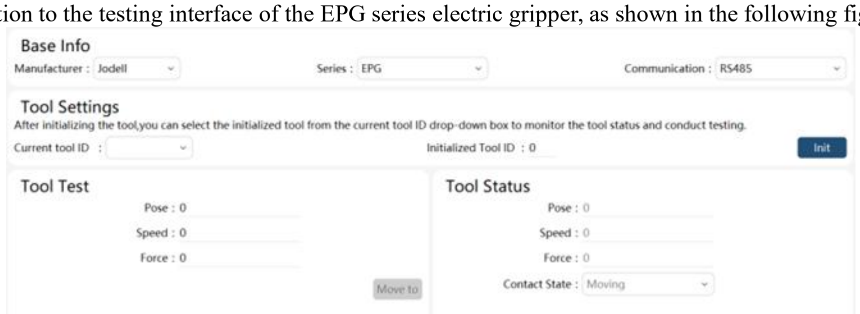
The parameter description for setting "Basic Information" is as follows:

Parameter	Explanation
Manufacturer	Jodell, Robustmotion, Robotiq, and DH;

Series	Jodell, supports EPG series electric grippers and EVS series suction cups; Robustmotion, supports RM-RMG and RM-C series electric grippers; Robotiq, supports the 2F85 series; DH, supports the PGI series;
Communication	Only RS485 interface is supported;

10.10.2.1Jodell electric grippers

The "Basic Information" section allows the Manufacturer to select Jodell as the preferred option, while the Series option enables them to choose EPG. Consequently, the HMI will seamlessly transition to the testing interface of the EPG series electric gripper, as shown in the following figure:

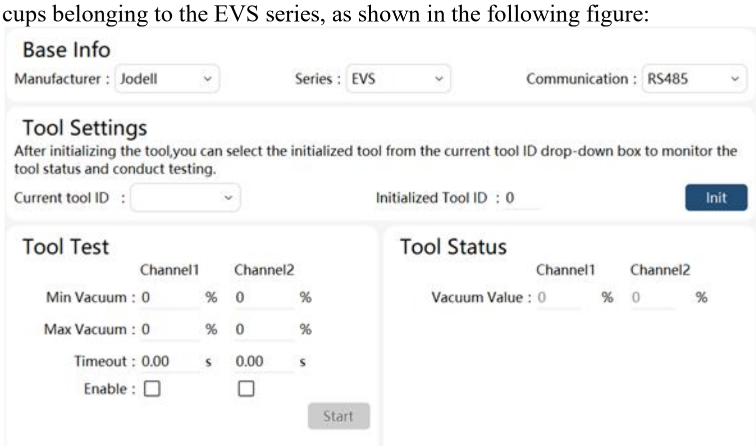


Initialization: To test an electric gripper, enter the ID of the gripper and click the "Initialize" button. If the gripper successfully detects and initializes, it indicates that the hardware connection and communication are functioning properly, allowing users to proceed or utilize it for further operations.
Tool testing: After initialization, click the "Move to" button to control the electric gripper's movement to a specified position with designated velocity and force. If the gripper reaches the desired position or encounters objects with predetermined force, it will halt its motion accordingly while displaying its contact detection status on the testing interface.
The relevant parameter settings are explained as follows:

Parameter	Value/Description
Tool ID	Enter the ID of the electric gripper. This ID is the electric gripper ID set in the Jodell Robotics debugging software
Tool position	Set the position of the electric gripper, with a range from 0 to 255
Tool velocity	Set the velocity of the electric gripper, with a range from 0 to 255
Tool torque	Set the torque of the electric gripper, with a range from 0 to 255

10.10.2.2Jodell suction cup

The "Basic Information" section includes the selection of Jodell as the Manufacturer and EVS as the Series. Consequently, the HMI will automatically switch to the testing interface specifically designed for suction cups belonging to the EVS series, as shown in the following figure:



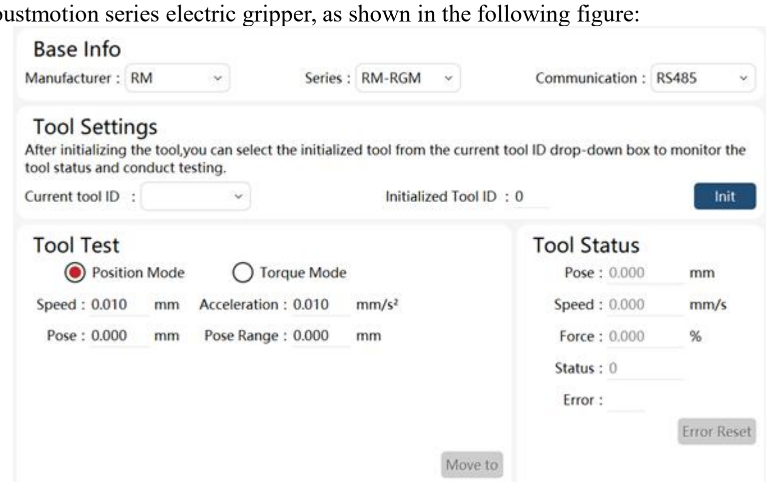
Initialization: To test a suction cup, first input the ID of the suction cup and click on the "Initialize" button. If the software prompts successful initialization, it indicates that the hardware connection and communication of the suction cup are functioning properly, allowing users to proceed to the next step or utilize it further.
Tool testing: After initialization, adjust the suction cup parameters as required. Once all parameters have been entered, click on the "Setup" button to conduct a test on the suction cup.
The parameter settings are explained as follows:

Parameter	Value/Description
Channel selection	The suction cup supports two channels. The user can choose the effectiveness of the two channels at will

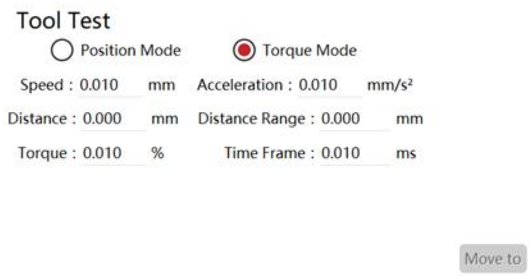
Minimum vacuum	Set the target vacuum level of the suction cup. The suction cup stops working when the inside vacuum level reaches this value
Maximum Vacuum	Set the target vacuum level of the suction cup. The suction cup starts working when the inside vacuum level is greater than this value
Timeout period	Times out when the minimum vacuum level specified is not reached in the specified time

10.10.2.3Robustmotion electric gripper

The Manufacturer selects Robustmotion in the "Basic Information" section, and chooses either RM-RGM or RM-C for the Series. As a result, the HMI will automatically switch to the test interface of the Robustmotion series electric gripper, as shown in the following figure:



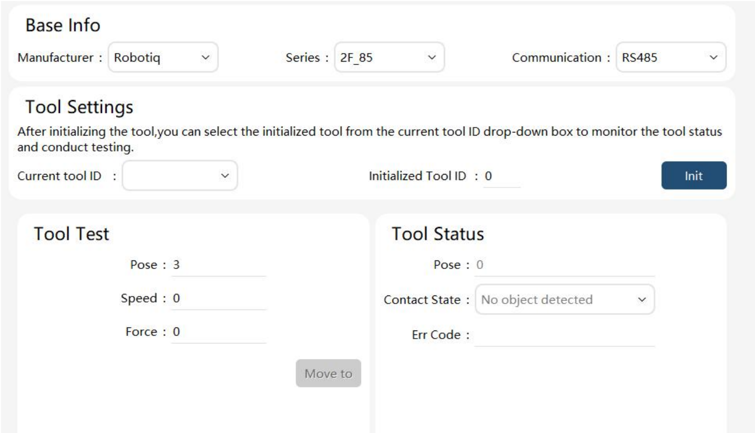
Initialization: To test an electric gripper, it is necessary to input the ID of the device and click on the "Initialization" button. If the electric gripper is properly configured and communication is established, initialization will be successful, enabling further testing and usage.
Current tool ID: This displays a list of initialized device IDs. By selecting a device with its corresponding ID, users can view its status and conduct tests.
Tool state: After successful initialization, the Robustmotion electric gripper can display various tool states including position (mm), speed (mm/s), torque (%), in-place state (1: in place), and error alarms (error code information).
Tool test: The running condition of Robustmotion electric grippers can be tested in position mode or torque mode. According to the manufacturer's instructions, position mode should be used for testing grip release while torque mode should be used for grip closure.
Position mode: Parameters that can be set include absolute positioning position (mm), speed (mm/s), positioning range (considering in-place error range, mm), and acceleration (mm/s^2), as shown above.
Torque mode: The parameters that can be set include distance (relative distance, mm), speed (mm/s), force (%), acceleration (mm/s2), positioning range (in-place error range considered, mm), and time range; as shown below:



The electric gripper can move according to the set mode and parameters if clicking on the "Move to" button.

10.10.2.4Robotiq 2F_85 electric gripper

In the "Basic Information", set the Manufacturer to Robotiq and the Series to 2F_85, and the HMI will switch to the testing interface of the Robotiq 2F_85 electric gripper, as shown in the following figure:

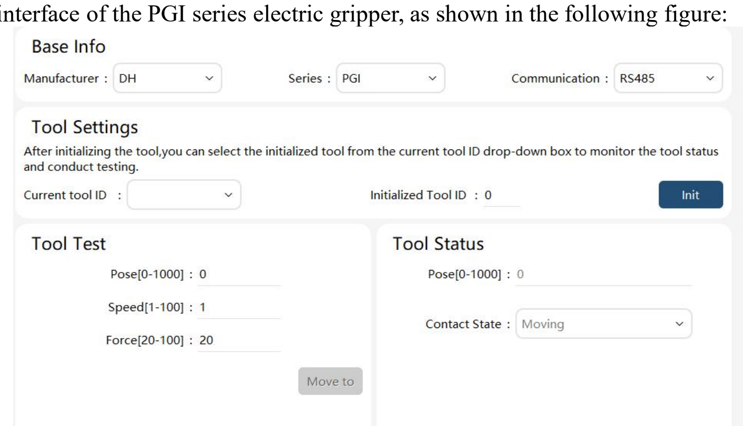


Initialization: To test an electric gripper, enter the ID of the gripper and click the "Initialize" button. If the gripper successfully detects and initializes, it indicates that the hardware connection and communication are functioning properly, allowing users to proceed or utilize it for further operations.
Tool testing: After initialization, click the "Move to" button to control the electric gripper's movement to a specified position with designated velocity and force. If the gripper reaches the desired position or encounters objects with predetermined force, it will halt its motion accordingly while displaying its contact detection status on the testing interface.
The relevant parameter settings are explained as follows:

Parameter	Value/Description
Tool ID	Enter the ID of the electric gripper
Tool position	Set the position of the electric gripper, with a range from 3 to 255
Tool velocity	Set the velocity of the electric gripper, with a range from 0 to 255
Tool torque	Set the torque of the electric gripper, with a range from 0 to 255

10.10.2.5DH electric gripper

The "Basic Information" section allows the Manufacturer to select DH as the preferred option, while the Series option enables them to choose PGI. Consequently, the HMI will seamlessly transition to the testing interface of the PGI series electric gripper, as shown in the following figure:



Initialization: To test an electric gripper, enter the ID of the gripper and click the "Initialize" button. If the gripper successfully detects and initializes accompanied by executing one opening and closing action, it indicates that the hardware connection and communication are functioning properly, allowing users to proceed or utilize it for further operations. Note: In the initialization process, the opening and closing of the electric gripper determines its operating range, and the electric gripper will move within the operating range.
Tool testing: After initialization, click the "Move to" button to control the electric gripper's movement to a specified position with designated velocity and force. If the gripper reaches the desired position or encounters objects with predetermined force, it will halt its motion accordingly while displaying its contact detection status on the testing interface.
Note: If the electric gripper grasps an object during movement, it will not continue to move even if the object is removed, and will only move again after a new target position is re-set.
The electric gripper has five statuses, namely 0 (the electric gripper is moving), 1 (the electric gripper does not grasp the object), 2 (the electric gripper grasps the object), 3 (the object falls), and 4 (no content is displayed, that is, the current ID is not initialized successfully).

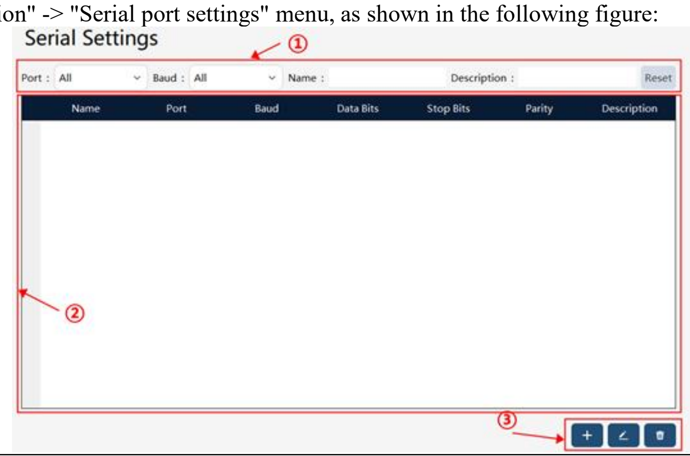
The relevant parameter settings are explained as follows:

Parameter	Value/Description
Tool ID	Enter the ID of the electric gripper. This ID is the one set in the DH debugging software
Tool position	Set the position of the electric gripper, with a range from 0 to 1000 in % (determined in the initialization process)
Tool velocity	Set the speed of the electric gripper, with a range from 0 to 100 in %
Tool torque	Set the torque of the electric gripper, with a range from 20 to 100 in %

10.11Serial port settings

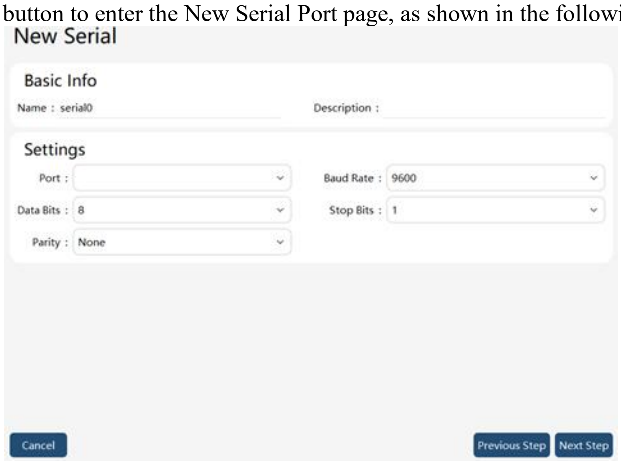
Users can utilize serial ports for communication with external devices. The utilization of serial ports necessitates hardware equipment support. The XBC5 control cabinet of industrial robots features a dedicated RS-232 serial port on the cabinet body. Alternatively, users can leverage the reserved USB interface in the control cabinet and employ the USB to RS-232 interface module for serial port communication. However, this functionality is not supported by collaborative robots due to the absence of relevant hardware interfaces.

On the HMI main interface, users can access the serial port settings interface via the "Communication" -> "Serial port settings" menu, as shown in the following figure:



①	Serial port display filtering.
②	List of serial port device.
③	Serial port edit buttons, from left to right, are Create, Edit, and Delete.

Click the "Create" button to enter the New Serial Port page, as shown in the following figure:



Before using the serial port, the parameters that need to be configured are as follows:

Parameter	Explanation
Name	The custom name to be used as the unique identifier in RL to use the serial port resources. Note: The serial port name is subject to the name conflict restriction in the project. It should not be identical with the existing network identifiers in the project or the existing identifiers of other serial ports.
Port	System port. The control system lists all the serial port resources detected (including the USB-to-USB ports) for users' selection and use.
Baud rate	1200/2400/4800/9600/19200/38400/57600/115200 are optional.
Data bit	5/6/7/8 bits are optional.

Stop bit	1/1.5/2 bits are optional.
Parity bit	Odd parity/Even parity/Mark parity/Space parity/None parity are optional.

After configuring the parameters, click the "Next" button to complete the serial port configuration. At this time, use the serial port in the RL program. The serial port function includes a series of commands, please refer to the detailed description of serial port command in the RL command.
Note: Please try to ensure that the parameter settings on both ends of the serial port communication are consistent, otherwise, it may cause abnormal data transmission and reception.

10.12Encoder

This function is part of the conveyor belt tracking function. For detailed usage, please refer to the *Conveyor Belt Tracking Function User Manual*.

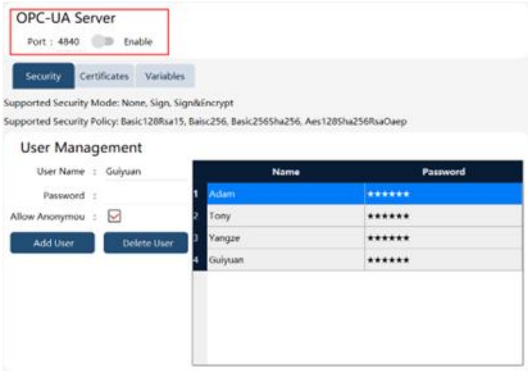
10.13OPC-UA

10.13.1Overview

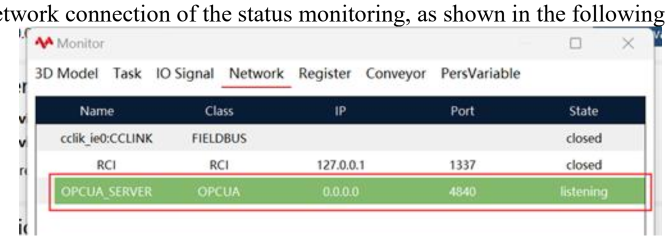
The OPC-UA of the xCore control system currently supports the xCore controller as the server of OPC-UA communication, and supports all mandatory nodes and some optional nodes in the *OPC 40010-1 OPC UA for Robotics, Part 1: Vertical Integration* standard by default. For a detailed model introduction, please refer to the relevant parts of Appendix OPC-UA. In addition, it also supports user-defined variables and event upload functions.

10.13.2Open and close

On the HMI main interface, enter the OPC-UA configuration interface through the menu "Communication" -> "OPC-UA", as shown in the figure below. You can enter the port of the OPC-UA service in the "Port" edit box above, and turn on or off the OPC-UA service through the "Enable" button.

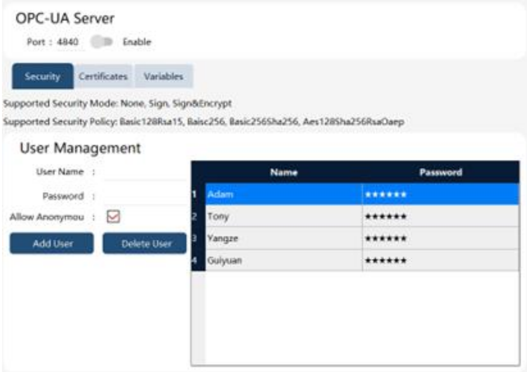


After the OPC-UA service is opened normally, you can see that the OPC-UA service is in a listening state in the network connection of the status monitoring, as shown in the following figure.



10.13.3Safety

Click the "Safety" tab on the OPC-UA configuration interface to enter the safety configuration interface, as shown in the figure below:



This interface is mainly used to explain and configure OPC-UA server safety policies.

1. None, Sign, and Sign & Encrypt are supported by default. Select the required mode when the client connects.
2. Four safety policies are supported by default: Basic 128Rsa15, Basic 256, Basic 256Sha256, and Aes128Sha256RsaOaep. Just select the required policy when the client connects.
3. Tick the "Allow Anonymous" check box to allow the client to log in anonymously. If it is not checked, the client can only log in with the user in the list on the right.
4. Enter the user name and password, click the "New User" button, and you can add a new user to the list on the right.
5. Select the user in the list on the right, and click the "Delete User" button to delete unwanted users.

10.13.4Certificate

Click the "Certificate" tab on the OPC-UA configuration interface to enter the certificate configuration interface, as shown in the figure below:

1. By default, if the OPC-UA server does not import the certificate and private key, the control system will use the self-generated certificate and private key.
2. Click the "Import Certificate" button to import the server certificate, and click the "Import Private Key" button to import the server private key. Both the imported certificate and private key need to be in der format, and ensure that the certificate and private key match. Additionally, it should be noted that the URL of the imported server certificate must be: urn:xcore.opcua.server.
3. Click the "Import" button on the right to import trusted client certificates, and click the "Delete" button to delete the selected client certificates.

10.13.5Custom variable configuration

Click the "Variable" tab on the OPC-UA configuration interface to enter the custom variable configuration interface, as shown in the figure below:

The OPC-UA communication of the xCore control system supports custom variable function, supports four types of variables: bool, int, double, and string, and supports configuring whether the

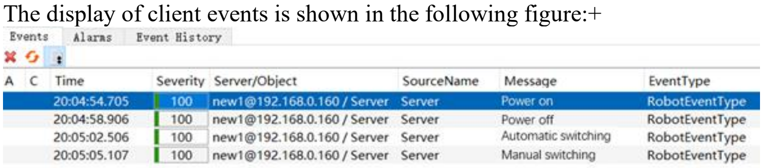
client writeable properties. The specific configuration functions are as follows:
1. Fill in the attributes such as "Name", "Description", "Type", "Writable", and "Initial Value" on the left side of the page, and click the "Create" button to add a custom variable to the list on the right. A maximum of 128 custom variables are supported.
2. Click the "Delete" button to delete the selected variable in the list on the right. Click the "Clear" button to remove all variables from the list on the right.
3. Check the "Enable Monitoring" check box under the variable list on the right to turn on variable monitoring, and "Current Value" will display the value of the variable in the controller in real time. Click the "Modify Value" button to modify the current value of the selected variable.
The client can find the node of the custom variable under the CustomVariables node in the Robotics model, and perform read and write operations. On the robot side, the user can use the ReadOpcUaVarByName and WriteOpcUaVarByName commands to read or modify OPC-UA custom variables in the RL program. For detailed usage methods, please refer to the corresponding instructions in the RL command chapter.

10.13.6Event

The OPC-UA server supports notifying the OPCUA client of some state changes of the robot through event. The currently supported reporting events are shown in the following table.

Event	Severity
Power on/off	100
Manual/automatic switching	100
Program running/stop	100
Emergency stop triggered	600
Safety gate stop triggered	600
Protection stop triggered	600
Collision alarm	600

The display of client events is shown in the following figure:+



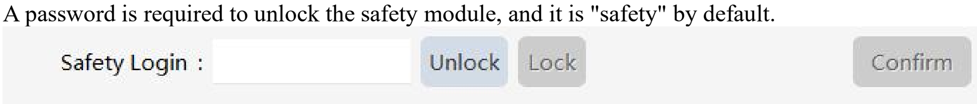
11Safety

11.1Introduction to this chapter

This chapter mainly introduces the settings of xCore safety related functions.

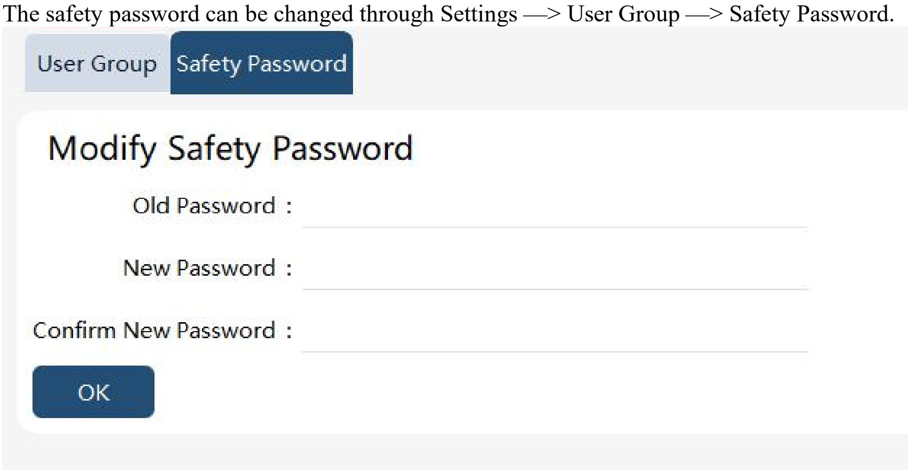
11.2Safety password

A password is required to unlock the safety module, and it is "safety" by default.



1、 To access the safety module, the user must enter the password and click "Unlock" to operate the safety interface.
2、 After unlocking, the user can enter other interfaces of the safety module without re-entering the password.
3、 Re-unlocking is required when the user switches back to the safety module from other modules.
4、 After modifying the settings of the safety module, the user needs to click the "OK" button to confirm the safety parameters.
5、 Regardless of the user permission, operators and other low-permission users are allowed to modify the safety module parameters after logging in.

The safety password can be changed through Settings —> User Group —> Safety Password.



11.3Joint limit

11.3.1Highlights

The joint limit monitors the parameters of robot joints. When the joint exceeds the threshold, the robot will immediately stop running, and the RSC robot will enter a safe stop state.
The joint limit mainly includes joint position limit, joint velocity limit, joint torque limit, and joint power limit. Each limit can be configured with two parameters for users to determine the threshold based on the current mode (normal mode or reduced mode).
The user has the flexibility to enable or disable specific functionalities as required.

11.3.2Joint position

11.3.2.1Highlights

The joint position limit is used to set the maximum motion range of each joint at the software level to avoid interference or collision between the robot and peripheral equipment.
During the drag process, the joint angles are also protected by the joint position limit. Drag near the joint position limit will give the manipulator a rebound force against the direction of the joint position limit. The range of the drag rebound force is within 10° of the upper and lower joint position limits set by the HMI interface. Assuming that the joint 1 position limit is −170° to 170°, then the range of the drag rebound force is [−170° to −160°] and [160° to 170°].

①	The "Enable" switch controls whether the joint position limit is enabled;
②	The lower limit of joint positions in normal mode shall be less than the upper limit.
③	The joint position limit in reduced mode shall be less than or equal to that in normal mode.
④	They are the maximum and minimum limits of joint positions for the robot.

Warning

1. The joint position limit shall not exceed the mechanical hard limit available to the robot.
2. When the servo firmware supports the maximum range of mechanical motion for collaborative robots, the joint position limit can be set to the maximum range of mechanical motion for each joint.

11.3.2.2Handling for moving beyond the joint position limit

In some rare cases, the robot may move beyond the joint position limit, such as triggering an emergency stop when moving to the limit, and exceeding the joint position limit when executing STOP 0.

In xCore V2.1 and earlier versions, when the robot has one or more joints outside the joint position limit, it will be unable to jog or run programs. At this point, it is necessary to first cancel the joint position limit, then jog the out-of-limit joint back within the joint position limit, and finally enable the joint position limit again.

In xCore V2.2 and later versions, for non-RSC robots, when the robot moves beyond the joint position limit, it is allowed to jog the robot back within the joint position limit.

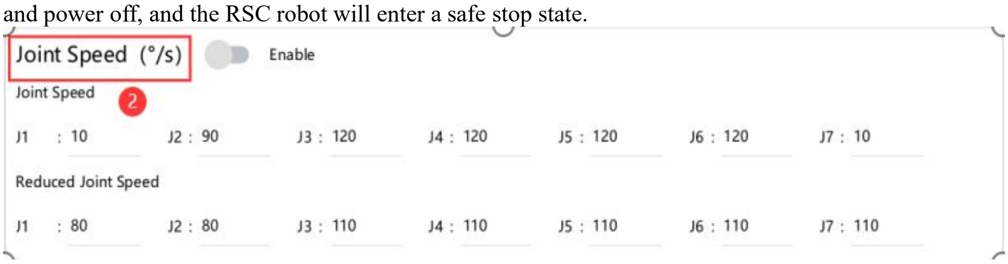
For RSC robots, when the robot moves beyond the joint position limit, it will enter the safe stop state. At this point, it is necessary to first cancel the joint position limit, then click "emergency reset", jog the out-of-limit joint back within the joint position limit, and finally enable the joint position limit again.

Warning

Cancellation of the joint position limit can only be used to jog the out-of-limit joint back within the normal range when the robot joint exceeds the joint position limit, and the program is unable to run when the joint position limit is canceled.

11.3.3Joint velocity

Joint velocity limit: The joint velocity limit can be turned on/off by an enable switch. When it is enabled, the angular velocity of robot joints will be monitored in real time. Depending on the current mode (normal mode or reduced mode), different monitoring parameters are used to determine the threshold. If any joint angular velocity exceeds the threshold, the robot will immediately plan to stop and power off, and the RSC robot will enter a safe stop state.

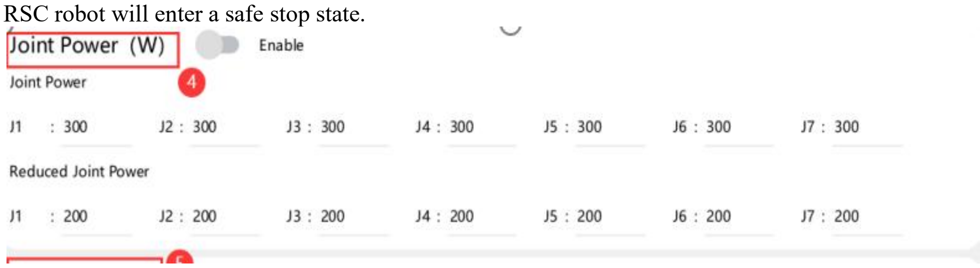


11.3.4Joint torque

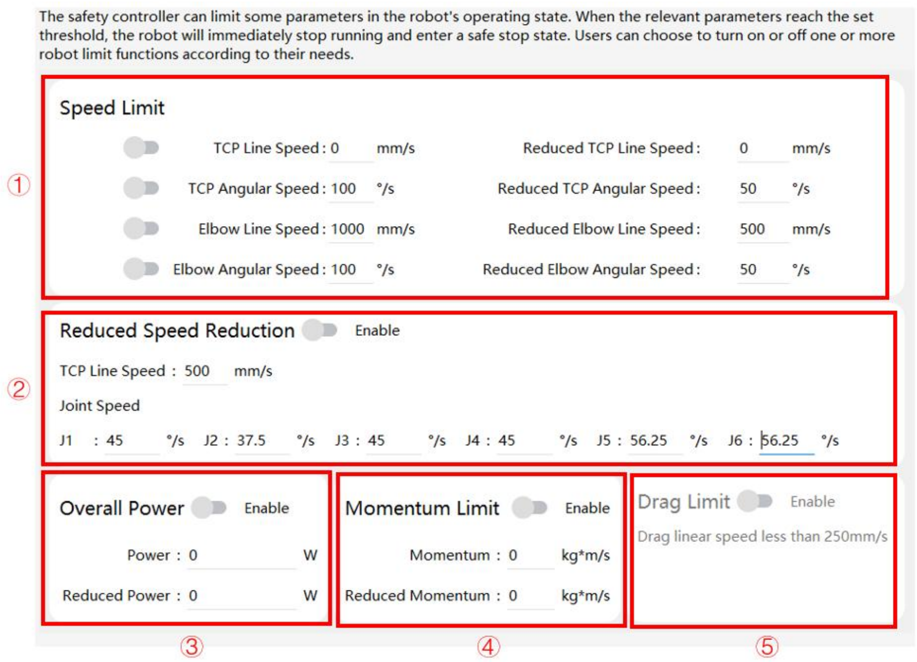
Joint torque limit: The joint torque limit can be turned on/off by an enable switch. When it is enabled, the torque of robot joints will be monitored in real time. Depending on the current mode (normal mode or reduced mode), different monitoring parameters are used to determine the threshold. If any joint torque exceeds the threshold, the robot will immediately plan to stop and power off, and the RSC robot will enter a safe stop state.

11.3.5Joint power

Joint power limit: The joint power limit can be turned on/off by an enable switch. When it is enabled, the power of robot joints will be monitored in real time. Depending on the current mode (normal mode or reduced mode), different monitoring parameters are used to determine the threshold. If any joint power exceeds the threshold, the robot will immediately plan to stop and power off; and the RSC robot will enter a safe stop state.



11.4Robot limits



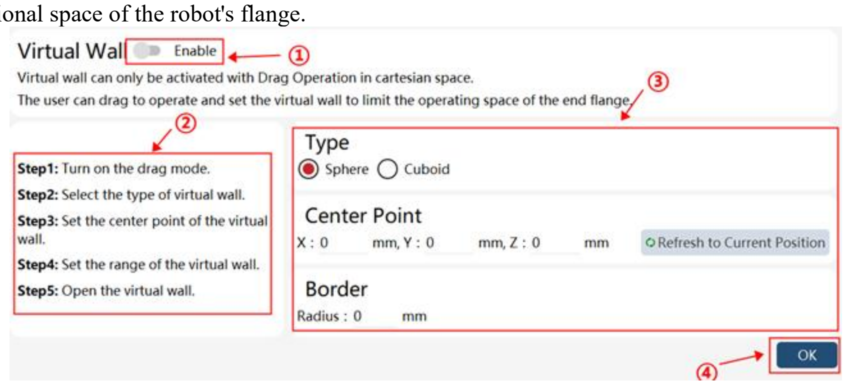
①	Velocity limit: The velocity limit covers TCP linear velocity, TCP angular velocity, elbow linear velocity, and elbow angular velocity. Additionally, each velocity limit can be turned on/off by an independent enable switch. When it is enabled, the velocity of robots will be monitored in real time. Depending on the current mode (normal mode or reduced mode), different monitoring parameters are used to determine the threshold. For example, "TCP linear velocity" is used as the threshold in normal mode, and "reduced TCP linear velocity" is used as the threshold in reduced mode. When any monitored value exceeds the threshold, the robot will immediately plan to stop and power off, and the RSC robot will enter a safe stop state.
②	Reduced velocity: If the user turns on the reduced velocity, in the reduced mode, the

	robot will move at the set TCP velocity and joint velocity.
③	Robot power limit: The robot power limit can be turned on/off by an enable switch. When it is enabled, the power of robots will be monitored in real time, and different monitoring thresholds will be used based on the current mode (normal mode or reduced mode). If the robot power exceeds the threshold, the robot will immediately plan to stop and power off, and the RSC robot will enter a safe stop state.
④	Momentum limit: The momentum limit can be turned on/off by an enable switch. When it is enabled, the momentum of robots will be monitored in real time, and different monitoring thresholds will be used based on the current mode (normal mode or reduced mode). If the momentum exceeds the threshold, the robot will immediately plan to stop and power off, and the RSC robot will enter a safe stop state.
⑤	Drag velocity limit: The drag velocity limit can be turned on/off by an enable switch, which is special for RSC robots. When it is enabled, if the drag velocity of the collaborative robot exceeds 250 mm/s, the robot will be stopped and powered off instantly to enter a safe stop state.

11.5Virtual wall

11.5.1Highlights

The virtual wall is specifically designed to confine the working area at the end of the flange in the Cartesian space (translation only) drag scene of the xMate collaborative robot. As users approach this virtual barrier, they will encounter a reactive force exerted by it.
The typical usage scenario involves medical professionals utilizing xMate collaborative robots as auxiliary tools for surgical operations through dragging actions. In order to enhance safety and prevent any potential misoperations, establishing a virtual wall becomes crucial to restrict the operational space of the robot's flange.



①	The "Enable" switch controls whether the virtual wall is enabled; Click “OK” to take effect;
②	Introduction to the steps for using the virtual wall function;
③	Virtual wall types, including sphere and cuboid; Virtual wall parameters, including center points; Note: The parameters set within the "boundary" must be greater than a value, such as 200 mm or above for the cuboid, and 100 mm or above for the sphere.

Note: In the extreme case of excessive drag force and speed, the robot may exceed the range of the virtual wall, and the system will provide corresponding prompts.

11.6Collision detection

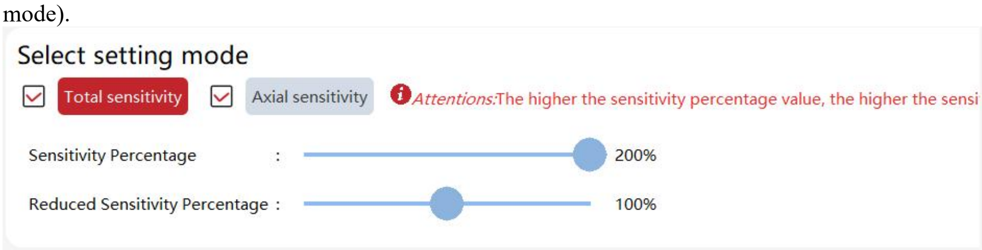
11.6.1Highlights

Collision detection is a passive function that relies on the estimation of the robot's dynamic model. It enables timely identification of unexpected collisions with the external environment during robot operation, allowing for prompt implementation of pre-set measures to mitigate any potential damage.

11.6.1.1Setting mode

The "whole setting" and "single joint setting" are available, and at least one of them shall be checked.
According to different setting modes, the sensitivity of the whole robot or single joint can be adjusted. The higher the percentage, the higher the sensitivity, and the easier it is for the robot to detect collisions. The factory default sensitivity is set to 100%, which can be adjusted by the user according to their needs.
Different sensitivity thresholds will be used based on the current mode (normal mode or reduced

mode).

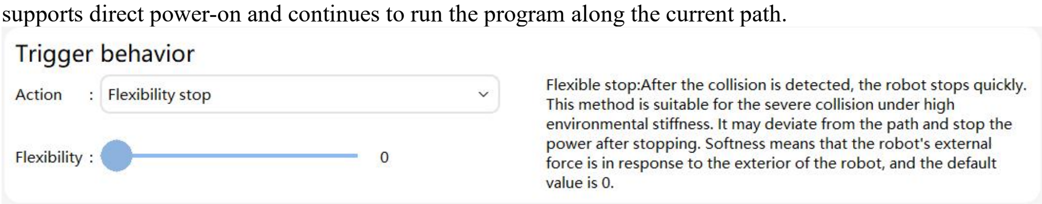


11.6.1.2Impact limit

Impact limit (including TCP impact and elbow impact): The impact limit can be turned on/off by an enable switch (collision detection is also turned on). When it is enabled, the TCP impact and elbow impact of the robot will be monitored in real time. Depending on the current mode (normal mode or reduced mode), different monitoring parameters are used to determine the threshold.
When any monitored value exceeds the threshold, the robot will enable the trigger behavior of collision detection, and the RSC robot will enter a safe stop state.

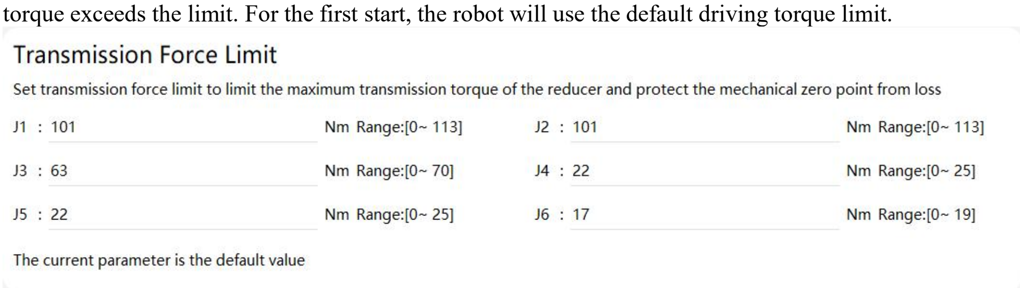
11.6.1.3Trigger behavior

The trigger behavior only includes a soft stop.
Soft stop: a collision detection stop method for robots and high stiffness environments. The greater the soft, the faster the response of the robot, and the greater the load of the robot joint; soft generally uses the default 0. After a safe stop, the robot will automatically power off. In this state, the robot supports direct power-on and continues to run the program along the current path.



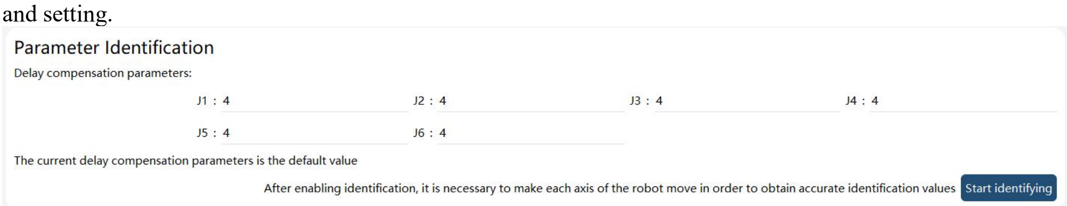
11.6.1.4Driving torque limit

The driving torque limit is used to limit the maximum driving torque of the reducer and protect the important parts of the driving chain and the mechanical zero.
The driving torque limit is available for collision protection*. When the controller detects that the driving torque exceeds the limit, the robot will trigger an error message indicating that the driving torque exceeds the limit. For the first start, the robot will use the default driving torque limit.

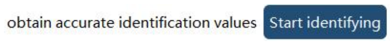
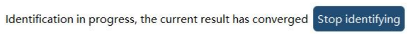
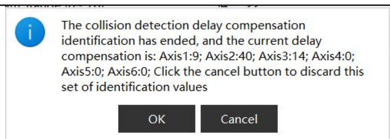


11.6.1.5Parameter identification

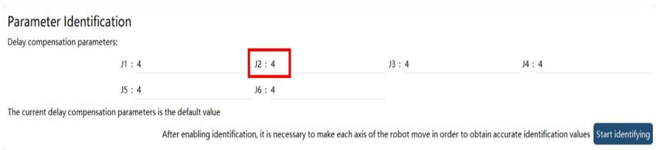
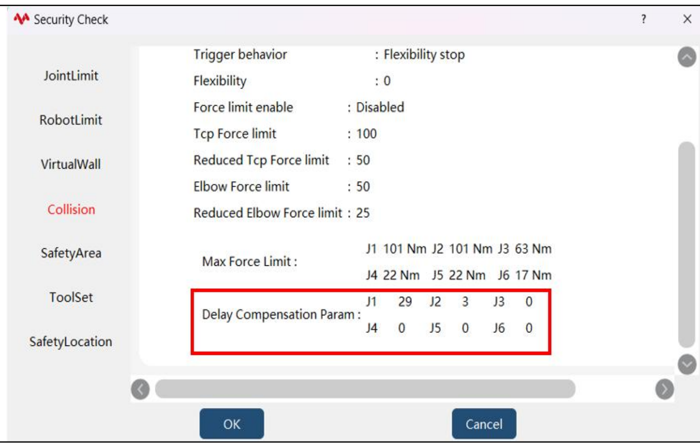
Collision detection parameter identification is used to identify and set the internal parameters of the collision detection algorithm to improve the accuracy of impact monitoring, reduce the probability of false alarms, and optimize collision detection performance.
Collision detection parameter identification supports "delay compensation parameter" identification and setting.



The detailed steps for enabling collision detection parameter identification are as follows:

Step	Graphical Representation	Explanation
1. Disable collision detection and collision protection.		The user needs to turn off the collision protection on the production interface.
1. Click the "Start Identification" button.		The dynamic feedforward needs to be enabled in advance for the collision detection parameter identification.
2. Switch to automatic mode and power on.		
3. Run the RL program for identification.		The user can use a dedicated identification program or any RL program that has collision detection.
4. After the identification result converges, stop the identification.		
5. Save the identification result and set the delay compensation parameters.		The delay compensation parameters take effect immediately after clicking "OK".

The detailed steps for manually setting delay compensation parameters are as follows:

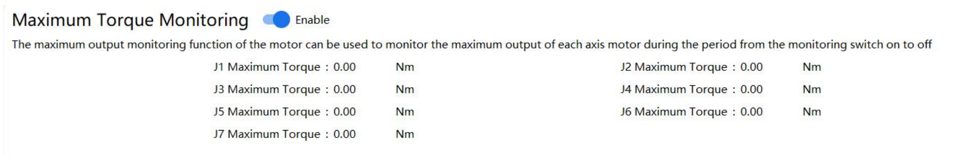
Step	Graphical Representation	Explanation
1. Input delay compensation parameters.		The manual settings of delay compensation parameters cannot exceed the range.
2. Click the "OK" button to confirm parameters.		

Note

1. In special cases, if it is necessary to identify collision detection parameters during production, collision detection and collision protection can be enabled during identification (not recommended).
2. To obtain accurate identification results of each joint, all joints in the identification program need to move as far as possible and as fast as possible.
3. The results of collision detection parameter identification are not fixed values with slight differences.

11.6.1.6Maximum output torque monitoring

The maximum output monitoring is used to monitor the maximum output torque of the motor of each joint during the period from enabling to disabling. Users can adjust the driving torque limit of each joint according to the maximum output torque monitoring parameters.



Note

The maximum output torque monitoring is disabled by default after the controller is restarted.

11.6.2Notes

1. During-program execution, if the robot collides with external devices while moving at high speed and the collision force exceeds a certain threshold, triggering an alarm and stopping the servo driver, the robot can only resume operation after clearing the collision, restarting itself, and resetting the servo alarm.
2. Incorrect sensitivity mode selected may cause a false collision alarm. Please select different sensitivity thresholds for each application scenario.
3. The collision detection sensitivity is affected by the robot hardware, and there are differences in sensitivity thresholds between different robots. Currently, the three sensitivity modes only provide a set of nominal values. The user with higher requirements for collision detection sensitivity can fine-tune the sensitivity of each axis based on specific application scenarios through the single-axis setting or adjust the detection sensitivity online through RL commands.
4. After collision detection and safety monitoring are triggered, a pop-up window will appear, and you must click "Confirm" to manually clear the alarm before continuing to run.
5. Collision detection is enabled by default at the factory for collaborative robots.
6. For the description of the collision protection*, see the user manual of the production interface.

Warning

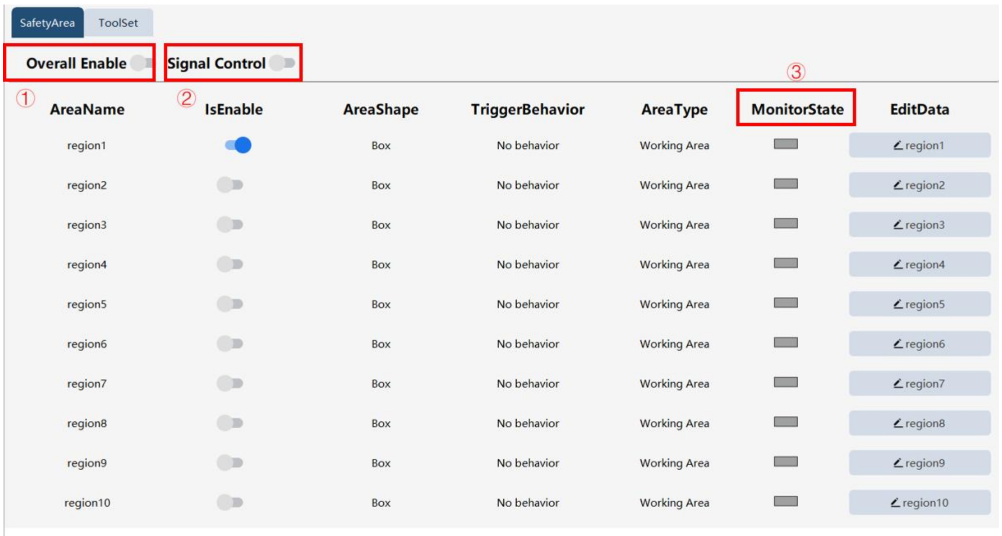
Before using collision detection, the user must ensure that the following parameters are set correctly. Otherwise, the controller may fail to calculate the correct output torque, resulting in a false alarm.

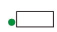
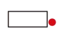
1. Robot model
2. Robot installation method
3. Load information (tool)
4. Mechanical and sensor zeros
5. Robot body parameters

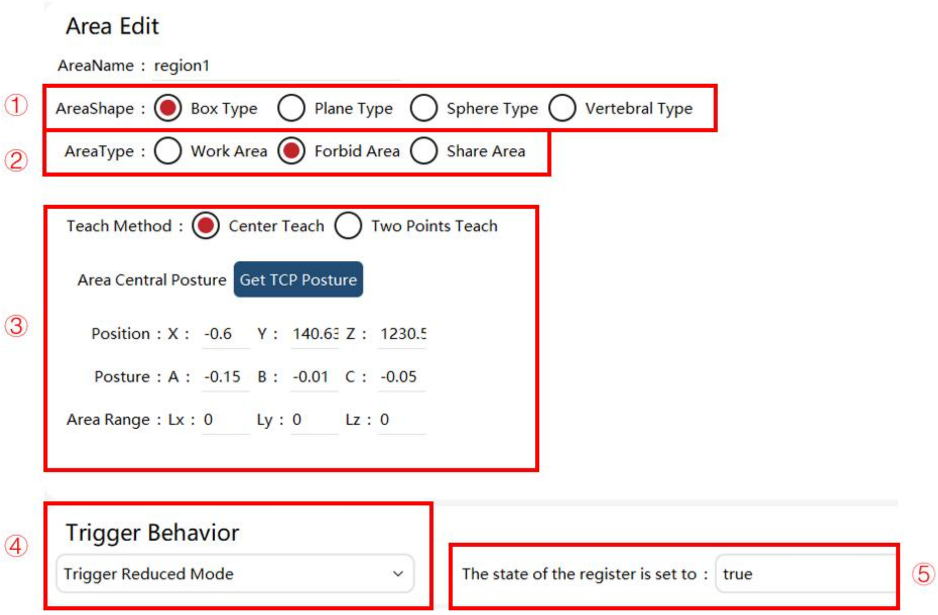
11.7Safe region

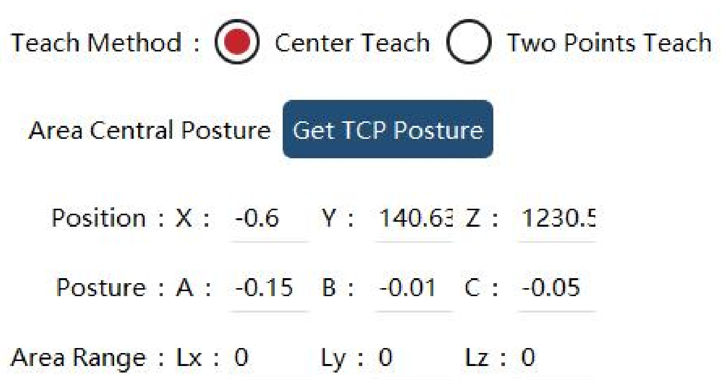
11.7.1Highlights

Safe regions are used to set the behavior of the end-effector and elbow in and out of a region.
The user can define several safe regions in the space (currently supports up to 10). When the robot enters and exits the safe region, it selectively triggers the preset safety behavior, and automatically modifies the register value (binding the register function code of the safe region).
The safe region retraction function is described in 11.7.3.



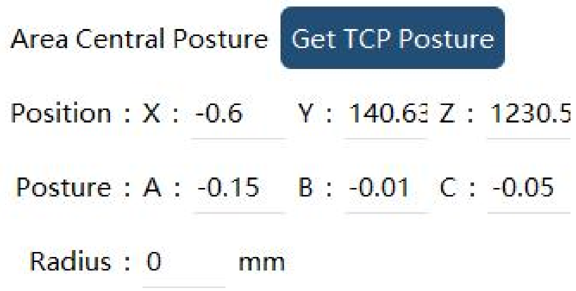
<table>
<tr><td>①</td><td>"Overall Switch": Turns on or off the safe region function, when this switch is off, all safe regions are invalid.</td></tr>
<tr><td>②</td><td>"Signal Control": After opening, a register signal can be used to control whether a certain safe region is turned on.
Each safe region supports two ways to set whether it is turned on: one is through HMI Settings, and the other is through Register Settings (register function code "enable _ safe _ region01 ~ enable _ safe _ region10", with bool or int16 type, and read-only).
Note: When the "Signal Control" is turned on, enabling/disabling the safe region is determined by the bound register, and the button under the "Enable/Disable" bar on the HMI will not take effect. When the "Signal Control" is turned off, the state of each safe region is determined by the "Enable/Disable" on the HMI.</td></tr>
<tr><td>③</td><td>Region monitoring: The relationship between the tool checked and regions are displayed.
<table>
<tr><td>Icon</td><td>Explanation</td></tr>
<tr><td></td><td>Region disabled</td></tr>
<tr><td></td><td>Inside the working region, and non-triggered</td></tr>
<tr><td></td><td>Outside the forbidden region, and non-triggered</td></tr>
<tr><td></td><td>Outside the working region, and non-triggered</td></tr>
<tr><td></td><td>Inside the forbidden region, and triggered</td></tr>
</table></td></tr>
</table>



<table>
<tr><td>①</td><td>Different shapes for region selection support different types of region settings, as shown in the table below.
<table>
<tr><th>Region Shape</th><th>Region Type</th></tr>
<tr><td>Cuboid</td><td>Working region, forbidden region, and shared region</td></tr>
<tr><td>Plane</td><td>Working region and forbidden region</td></tr>
<tr><td>Sphere</td><td>Working region, forbidden region, and shared region</td></tr>
<tr><td>Cone</td><td>Forbidden region for the inner part of the cone and working region for the outer part of the cone</td></tr>
</table>
Shared region: For a shared region, it is required to bind DIDO. If the DI is true, it indicates that the region is occupied and the robot will pause and wait outside the region when it is about to enter the region. When the DI is false, it indicates that the region is unoccupied, the robot will continue to move into the shared region, and the DO is set to true. It is unnecessary to set the trigger behavior of the shared region. When the robot is about to enter the occupied shared region, its behavior is to pause and wait. When the robot has entered the occupied shared region, its behavior is to slow down and stop at maximum capability.</td></tr>
<tr><td>②</td><td>Region teaching:
For cuboid regions, there are two teaching methods, namely center point teaching and two-point teaching.
Center point teaching: Click to get the current TCP pose (TCP relative to the base frame) to determine the center point pose of the region, and then manually set the length, width, and height of the cuboid.
Two-point teaching: Teach two points (two points on the cuboid diagonal) to determine the cuboid region, and then click "Confirm Point 1" —> "Confirm Point 2" —> "Confirm".


The plane region is determined by teaching or manually inputting the position (X, Y, Z),</td></tr>
</table>

<table>
<tr><td></td><td>and only manual input is available for the vector temporarily. The Z-axis direction is the safe region.


The sphere region is determined by teaching or manually inputting the center point pose and inputting the radius.


The sphere region is determined by teaching or manually inputting the center point pose and inputting the radius and height.


</td></tr>
<tr><td>③</td><td>Trigger behaviors include no behavior, normal mode enabled, reduced mode triggered, reduced mode enabled, and normal/reduced mode enabled.
No behavior: the robot has no specific action;

Normal mode enabled: In the normal mode, when the tool is about to enter the forbidden region, the planning to stop will be triggered; and when the tool has entered the forbidden region, the maximum capacity to slow down and stop will be triggered.

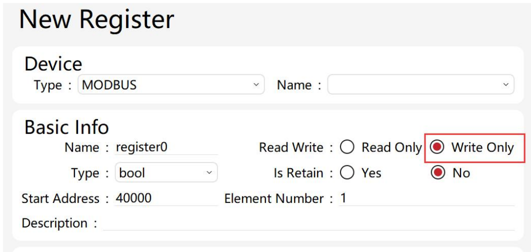
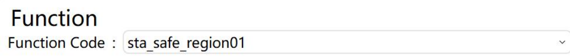
Reduced mode triggered: When the tool enters the forbidden region, the reduced mode will be triggered.

Reduced mode enabled: In the reduced mode, when the tool is about to enter the forbidden region, the planning to stop will be triggered; and when the tool has entered the forbidden region, the maximum capacity to slow down and stop will be triggered.

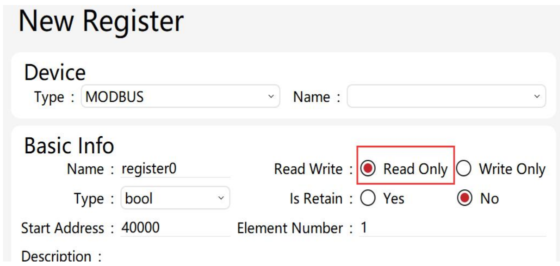
Normal/Reduced mode enabled: In the normal/reduced mode, when the tool is about to enter the forbidden region, the planning to stop will be triggered; and when the tool has entered the forbidden region, the maximum capacity to slow down and stop will be triggered.</td></tr>
<tr><td>④</td><td>The state of the region-bound register after triggering includes True/False.
True: the register output signal is true when the safe region triggers a safety behavior, and vice versa; (output true when entering the forbidden region).
False: the register output signal is false when the safe region triggers a safety behavior, and vice versa; (output false when entering the forbidden region).</td></tr>
</table>

11.7.2Association of safe region and register

11.7.2.1Safe region status output

Step	Graphical Representation	Explanation
1. First, create a new register, and select the type as write-only;		The left figure is for example only;
2. Select the function code "sta_safeRegion01–sta_safeRegion10", indicating binding the triggering status of the corresponding safe region to the current newly-created register.		

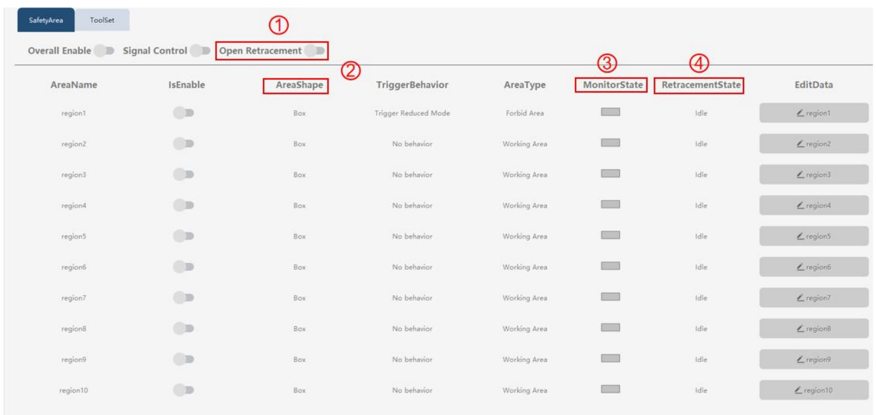
11.7.2.2Register control safe region enable

Step	Graphical Representation	Explanation
1. First, create a new register, and select the type as read-only;		The left figure is for example only;
2. Select the function code "enable_safe_region01–enable_safe_region10", indicating binding the control switch of the corresponding safe region to the current newly-created register;		

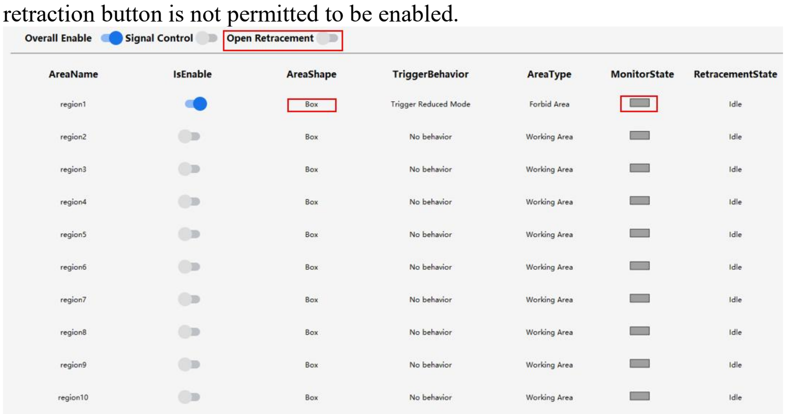
11.7.3Safe region retraction function

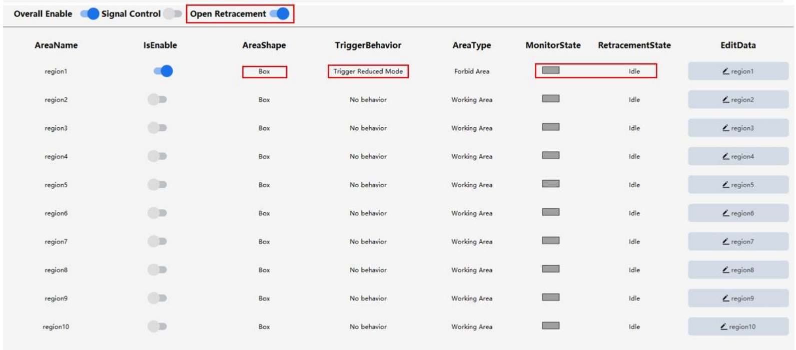
11.7.3.1Retraction function introduction

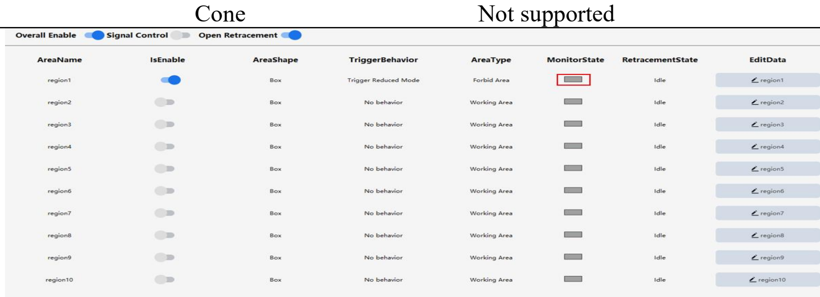
The safe region retraction function refers to the capability that when the robot is within a forbidden region, activating the safe region retraction button will directly move the robot out of the forbidden region without requiring deactivation of all safe regions.

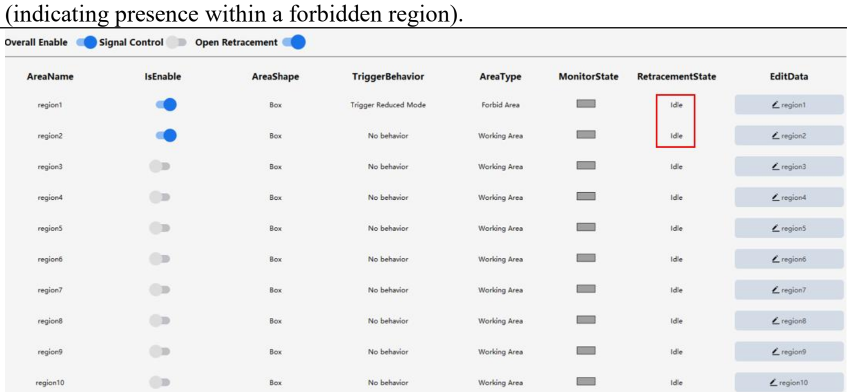


①	The button is the retraction activation button. When region monitoring shows the current position is within the forbidden region of a planar safe region and the region is in an

<table>
<tr><td></td><td>active mode (normal mode active, reduced mode active, or both normal and reduced modes active), the retraction button is allowed to be enabled. In all other cases, the retraction button is not permitted to be enabled.


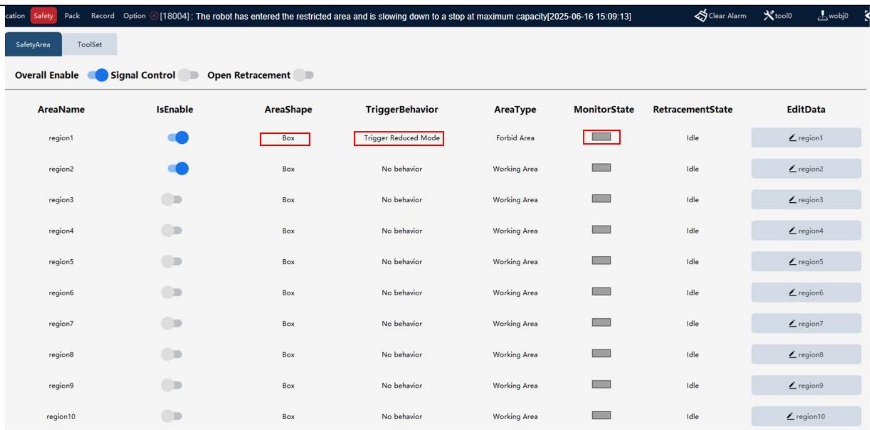
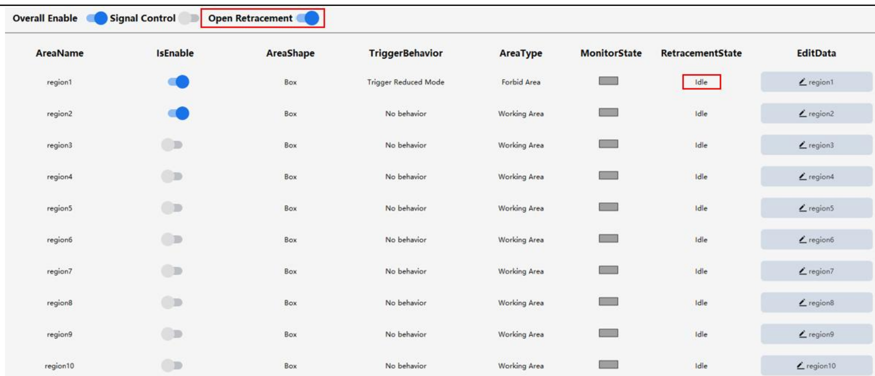
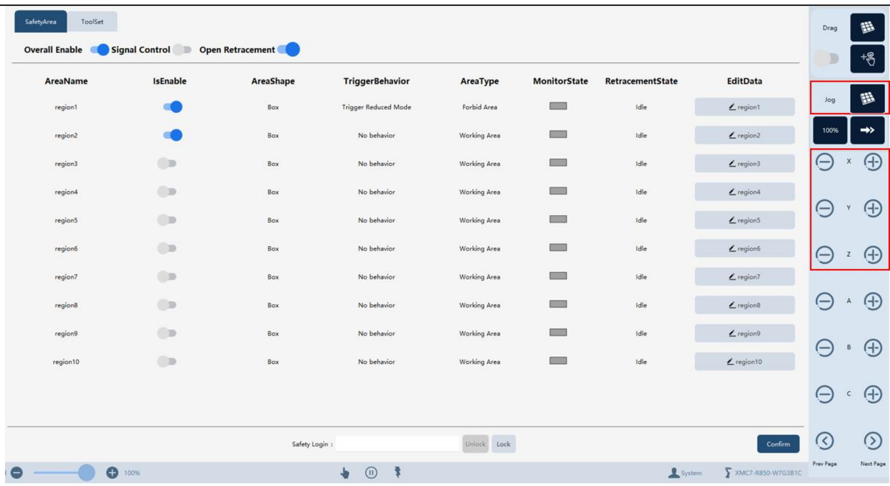


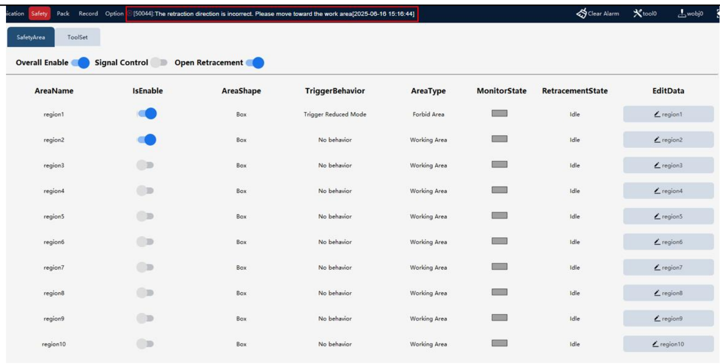
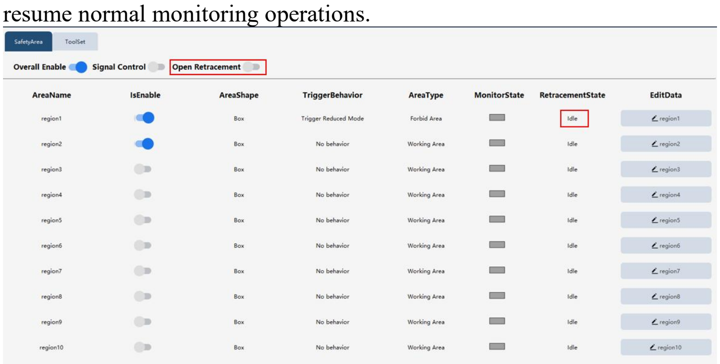
Upon completion of retraction for all planar safe regions, the retraction button will automatically deactivate</td></tr>
<tr><td>②</td><td>Among all region shapes, only planar safe regions support the retraction function
<table>
<tr><th>Region Shape</th><th>Support Safe Region Retraction</th></tr>
<tr><td>Cuboid</td><td>Not supported</td></tr>
<tr><td>Plane</td><td>Supported</td></tr>
<tr><td>Sphere</td><td>Not supported</td></tr>
<tr><td>Cone</td><td>Not supported</td></tr>
</table></td></tr>
<tr><td>③</td><td>

The retraction button can only be enabled when region monitoring displays red (indicating presence within a forbidden region).</td></tr>
<tr><td>④</td><td>

The retraction status is divided into two types: one is "In Retraction", indicating that the current region is undergoing retraction, where multiple regions can simultaneously be in retraction; the other status is "Idle", indicating that the current region is either waiting or does not require retraction.</td></tr>
</table>

11.7.3.2Retraction function operation procedure

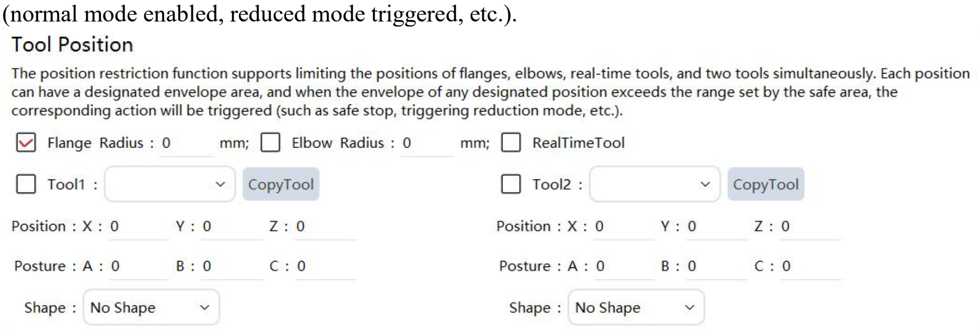
①	 When the robot enters a planar forbidden region, the system will prompt that it has entered the forbidden region and all motion operations will be disabled. If you want to move the robot out of the forbidden region at this time, you can turn on the retraction button to initiate retraction.
②	 After activating the retraction button, the status changes to "In Retraction" and power-on operations become available. For RSC robots, after retraction activation, you need to additionally click the emergency stop reset via the HMI interface; after the reset, power-on operations can be initiated.
③	 After power-on, jog operation is permitted only in the direction away from the forbidden region. Note: Retraction only permits jog operation (all other motion operations are prohibited), and jog is restricted to Cartesian X/Y/Z linear movements (all other motion modes are forbidden). If the jog direction is not oriented away from the forbidden region, the robot will prohibit motion and report an error as follows:

<table>
<tr><td></td><td>
</td></tr>
<tr><td>④</td><td>When retraction is completed, the retraction button will automatically deactivate and the retraction status will change to "Idle". Upon completion of retraction, all safe regions will resume normal monitoring operations.

</td></tr>
</table>

11.8Tool setting

11.8.1Tool position

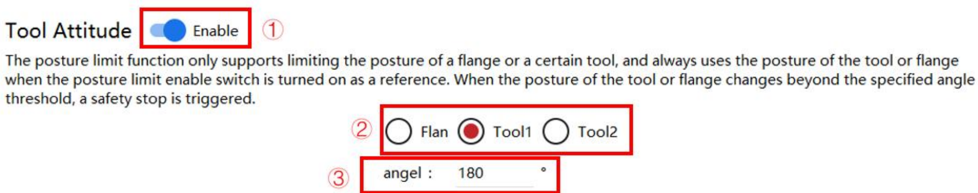
The tool position limit is available to limit the positions of flanges, elbows, real-time tools, and two fixed tools simultaneously. An envelope can be specified for each position. When the envelope of any position exceeds the setting of the safe region, the behavior of the safe region will be triggered (normal mode enabled, reduced mode triggered, etc.).



Tool envelope: The tool envelope includes three shapes, namely no envelope, cuboid, and sphere.

Real-time tool: When RL runs motion commands, the real-time tool is the tool in the command. When there is no motion command, the real-time tool is the tool selected on the upper right of the HMI. The envelope of the real-time tool can be set when editing the tool (global tool list in the frame calibration and tool list in the project).

11.8.2Tool orientation



①	The "Enable" switch can turn on/off orientation limits;
②	The orientation limit function only monitors one object at a time, and can select one from flange, tool 1, or tool 2;
③	Angle: When the orientation function is enabled, the orientation of the selected object is used as a reference, and a cone formed according to the set angle is used as the allowable

	range of the attitude; when the attitude of the selected object exceeds the range of the cone, a safe stop will be triggered;

11.9Safety position

11.9.1Highlights

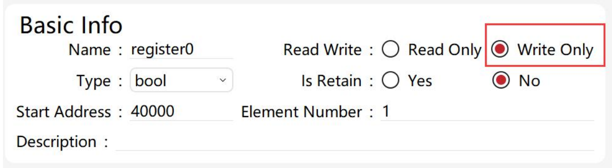
The safety position function refers to the binding register outputting a signal indicating the robot's presence in the predetermined safety position. Through this function, users can ascertain the relative positioning of the robot with respect to the safety position.

xCore control system supports up to 8 safety positions with joint angles as reference. Each safety position corresponds to a register function code (type: bool or int16, read/write: write only, sta_safe_jnt_pos1~sta_safe_jnt_pos8). When the current joint angle of the robot and the joint angle set for a safety position are within the allowable error, the value of the register to which the corresponding register function code for the safety position is bound to will be modified automatically (when within the allowable error of the safety position, if the register type is bool, the register value is true; if the register type is int16, the register value is 1).

The safety Home is special for RSC robots, and a safety position can be checked as the safety Home. After it is checked, a safety DO signal can be output if each joint of the robot reaches the set range. If none is checked, the safety Home is disabled.

①	Enable: A safety position can be enabled or not.
②	Safety Home: A safety position can be checked as a safety Home.
③	No.: After clicking, the user can set the parameters for the safety position on the right side.
④	"Joint Coordinate" corresponding to safety positions; It can be manually updated; you can also click "Update Position" to update the current joint position data of the robot;
⑤	The "Allowable Error" corresponding to the safety position, when the current joint angle and "Joint Coordinate" of the robot are less than the "Allowable Error", the robot is considered to be in the safety position;

11.9.2Association of safety position and register

Step	Graphical Representation	Explanation
1. First, create a new register, and select the type as write-only;		The left figure is for example only;
2. Select the function code "sta_safe_jnt_pos1−sta_safe_jnt_pos 8", indicating binding the feedback status of the corresponding safety position to the current newly-created register.		

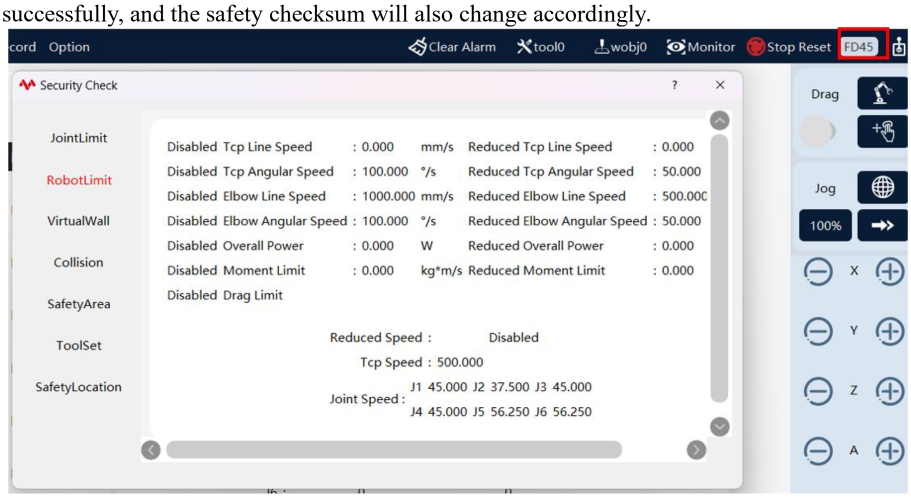
11.10Safety checksum

To modify the safety settings, click the "OK" button at the lower right of the interface and confirm the settings after the safety checksum.

The "Safety Checksum" icon displays a combination of four digits of "number + letter" to allow the user to understand the status of safety-related settings. When there is a change in safety-related settings, it will automatically calculate and generate a new combination of four digits of "number + letter".

After clicking the icon, the current safety settings will be available, including the joint limit, robot limit, virtual wall, collision detection, safe region, tool settings, and safety position.

After modifying the parameters of these items, clicking the "OK" button will trigger a pop-up window displaying the safety checksum. After clicking the "OK" button, the safety parameters will be set successfully, and the safety checksum will also change accordingly.

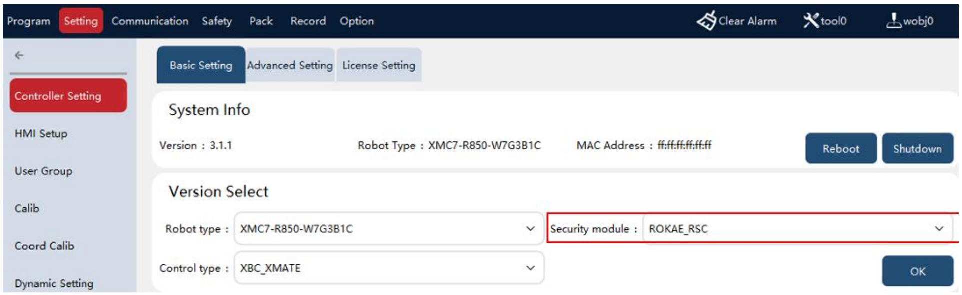


11.11Safety controller

The xCore control system can be optionally equipped with an RSC safety controller, which is a safety module that complies with functional certification and performs various internal safety-related calculations and protections. The safety functions of the xCore control system are processed in parallel, forming a dual safety architecture.

To ensure the data and parameter integrity of the safety controller, real-time data transmission adopts the FSoe communication mechanism for accurate transmission, while non-real-time data utilizes a secure synchronization mechanism with a synchronization time of 5s−10s.

For robots equipped with safety controllers, the safeboard type is ROKAE_RSC as depicted in the figure below.



11.11.1Changes after equipping safety controllers

In addition to the functions displayed on the subsequent safety controller configuration interface, there are several changes in the use of robots equipped with safety controllers.

11.11.1.1Changes to robot motor state

"Safety stop state" is added to the robot state to indicate the safety state caused by the limits of the safety controller.

Safe stop state		The robot is in a safe stop state, which means that the safety controller detects that the work or communication is abnormal, or a parameter exceeds the safety threshold set by the safety controller, and the robot cannot be powered on.

11.11.1.2Added robot reset

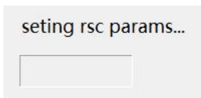
When the robot is in any of the "emergency stop state", "safety gate state" or "safe stop state", to reset it to the "power-off state", you must complete the following 2 steps:
Step 1: Eliminate the operation or condition that triggers the above three states, such as rotating the emergency stop button to "OFF" position, clearing the safety gate trigger signal, removing the safety overrun factor, or disabling the corresponding safety limit;
Step 2: Click "Reset" button on the interface, and the robot will reset to the "power-off state".

11.11.1.3Changes to the safety gate logic

For robots without safety controllers, when the robot is in automatic mode and receives the signal of safety gate closed, the robot will be powered off immediately.
For robots with safety controllers, when the robot is in automatic mode and receives a signal of safety gate closed, the RL program will be suspended, and the robot will not be powered off. In this situation, the robot is unable to run the RL program or step through the RL program. If you want to restore the robot's status, you can: execute the signal to disconnect the safety gate, and click on the "Reset" signal on the HMI.

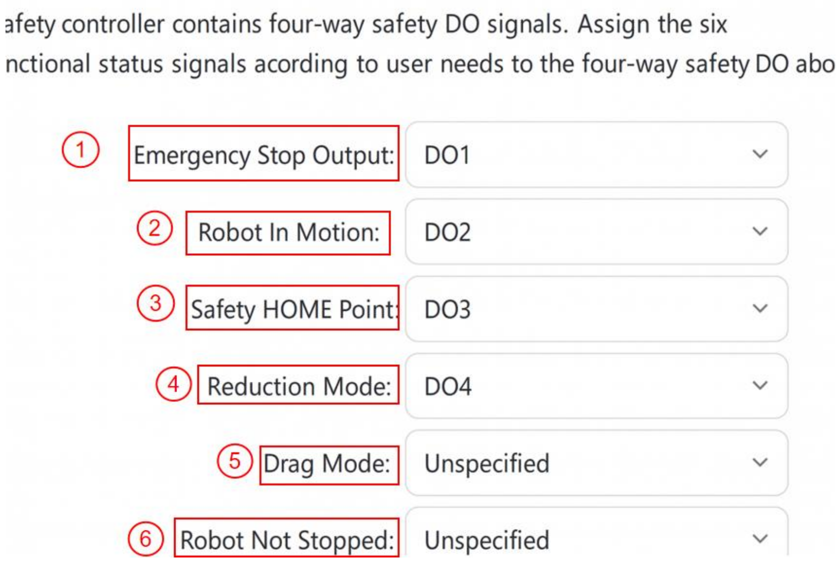
11.11.1.4Time difference between zero calibration and friction parameter setting

The zero information and the friction parameter information of the robot need to be synchronized to the safety controller to ensure the basic safety restriction function of the safety controller to be used normally. Therefore, when the user performs zero calibration or sets friction parameters, the controller will actively synchronize the updated parameters with the safety controller, which takes about 5s–10s to wait. At this point, the interface is as shown in the figure below, and the user is unable to operate and use the robot.



11.11.2Safety DO configuration

The safety controller has four channels of safety DO signals. The user can map several safety state signals to the four safety DOs.



①	E-stop output: When the robot is in the E-stop state, the output is true. Otherwise, it is false.
②	Robot in motion: When the robot RL program is in operation, the output is true, but when it is not in operation, the output is false.
③	Safety Home point: When the robot is within the range of the safety Home point of safety controller, the output is true. Otherwise, it is false.
④	Reduced mode: When the robot is in reduced mode, the output is true. Otherwise, it is false.
⑤	Drag mode: The robot is in drag mode.
⑥	Robot still on: Provided that the robot RL program is in operation (i.e., "Robot in motion" is true), if the robot joints are in motion, the output is true. Otherwise, it is false.

12Process Package

The xCore Control System not only offers impeccable core functionalities, but also encompasses an extensive array of advanced features. In relation to the process packages listed below, we provide exclusive documentation. Please feel free to reach out to us if you require any further assistance.

12.1Conveyor belt tracking

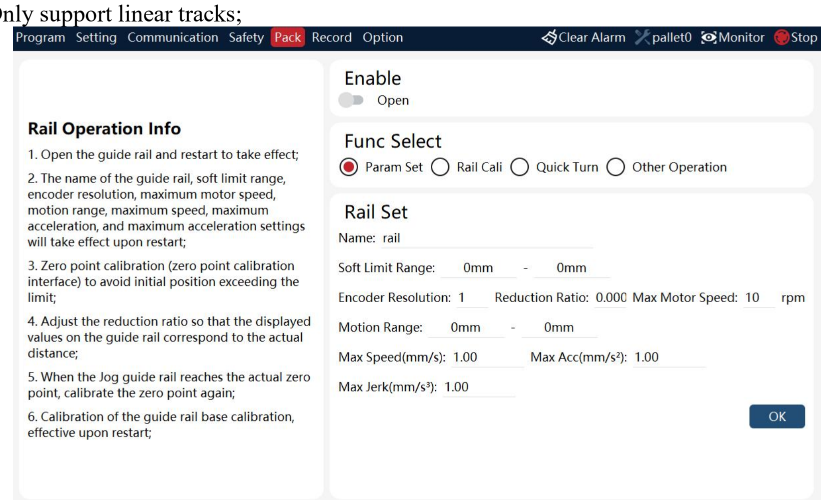
Main features and limitations:

- Support linear conveyor belt tracking;
- Support two positioning/triggering methods: photoelectric and 2D vision;
- Support industrial six-axis robots; and
- xMate collaborative robots only support the versions with control cabinets.

12.2Track

Main features and limitations:

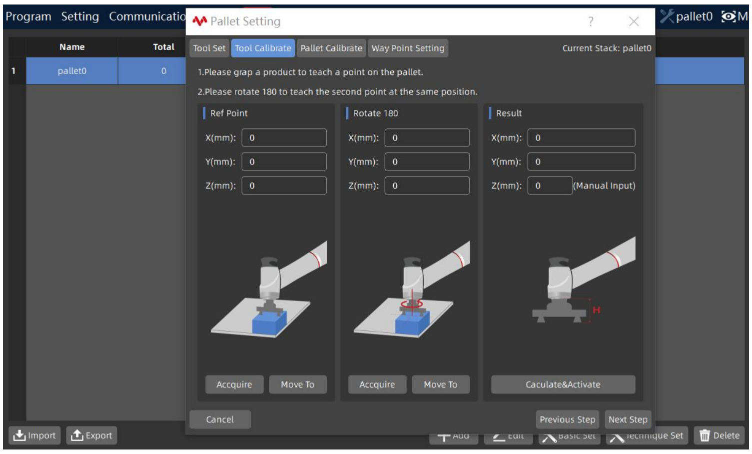
- Only support linear tracks;



12.3General stacking

Main features and limitations:

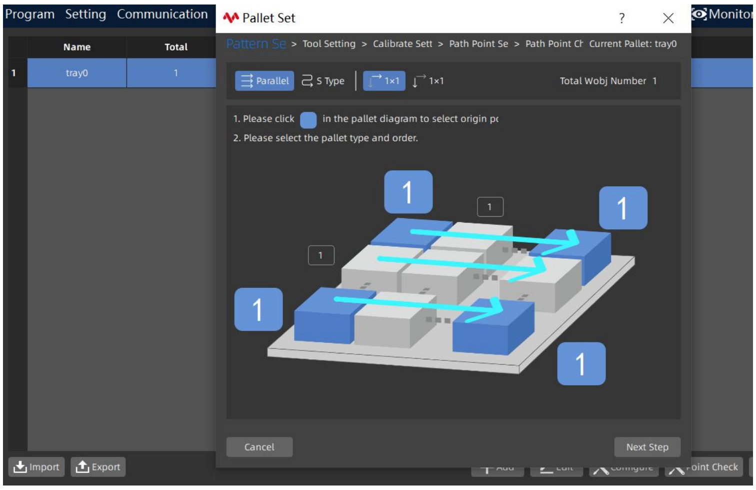
- Support the creation of up to 100 stacking processes;
- Support preset stack patterns, including matrix overlapping, criss-cross, and rotating, and support custom patterns; and
- One stacking process supports the creation of up to 100 plane layouts and up to 50 layers, and one plane layout supports the creation of up to 200 work objects.



12.4Tray

Main features and limitations:

- Support up to 100 tray processes;
- Support custom stacking patterns of robot, including parallel pattern and S-shaped pattern; and
- One tray process supports up to 16 plane layouts, and one plane layout supports up to 999 work objects.



12.5PV typesetting

Please contact ROKAE for more information.

12.6PV inserting

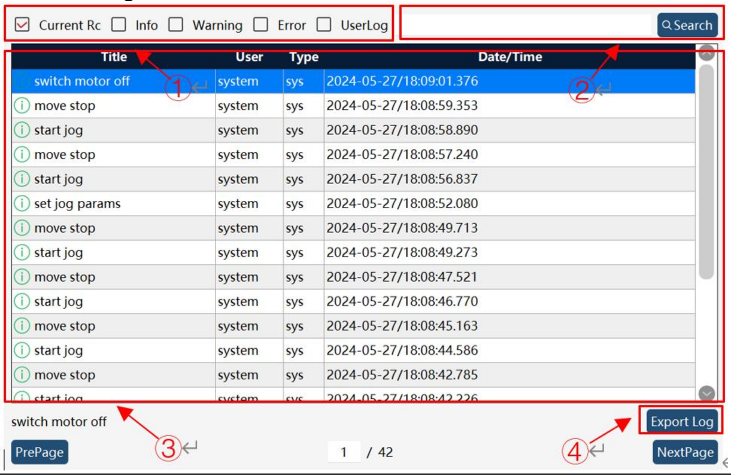
Please contact ROKAE for more information.

13Log

The log module includes eight interfaces, namely HMI logs, controller logs, operation logs, log timelines, internal logs, diagnostic settings, hardware status, and diagnostic data monitoring.

13.1HMI logs

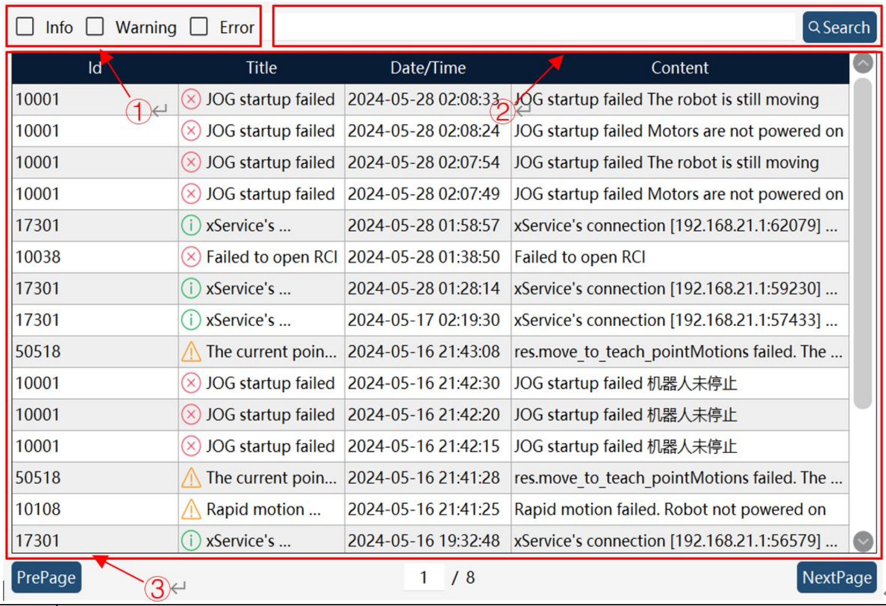
The HMI logs interface displays the current user's HMI operation log information, and this interface includes filtering criteria area, search area, display page, etc. The user can click "Export Log" to export the desired HMI log information.



①	Filtering criteria area, where the user can choose to view only the controller logs of the current controller or the ones connected to the HMI, as well as select the log level for further filtering.
②	Search area, where the logs can be searched by the input of keywords.
③	Log display page, where log title, user, type, and generation time are displayed. The previous and next pages can be switched by clicking on "Previous/Next" buttons.
④	The "Export Log" button can be clicked to export the log to a CSV format file.

13.2Controller logs

The controller logs interface displays the controller log information of the current robot, and this interface includes filtering criteria area, search area, and display page. The User can view the controller log information here.



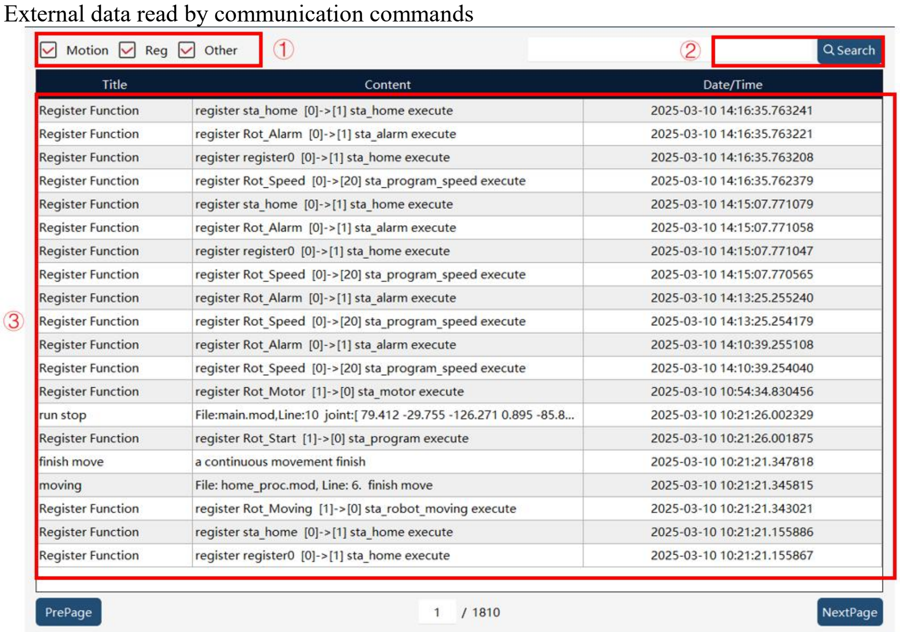
①	Filtering criteria area, where the user can select log level for further filtering.
②	Search area, where the logs can be searched by the input of keywords.
③	Log display page, where basic information such as log number, title, generation time, and content is displayed. The user can press Previous/Next to switch between

	the previous and next pages.

13.3Operation logs

Compared with controller logs, operation logs mainly record the key commands that the robot has executed, including the following information:

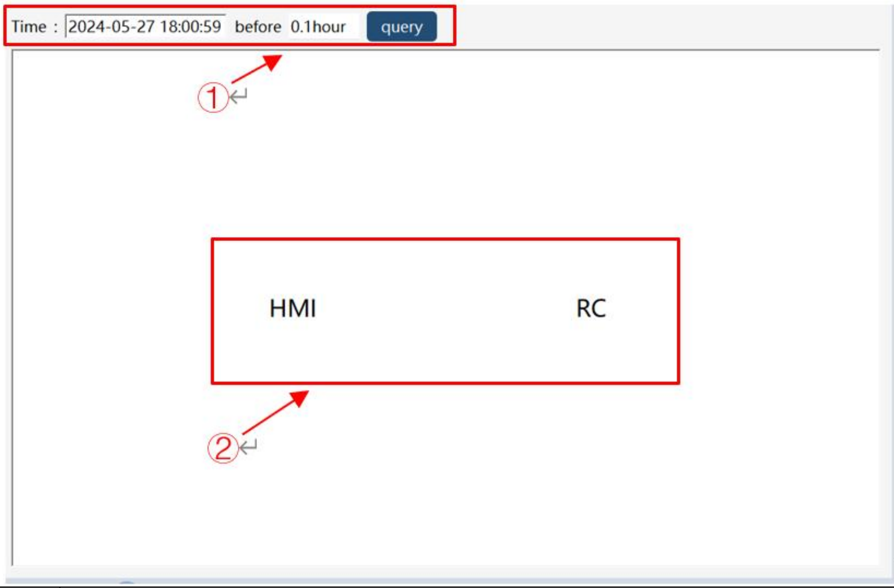
1. Start and stop of the robot and execution of motion commands
2. Execution of two logical commands IF-ELSE and WAIT UNTIL
3. Execution of commands SetDO\PulseReg\PulseDO
4. JOG, drag, and regain path
5. Emergency stop, pause, and start
6. Register status codes output externally, and control codes received from external controllers
7. Changes in safe regions
8. Triggering of functions such as collision detection and diagnostic data saving
9. External data read by communication commands



①	Filtering criteria area, where the user can select log level for further filtering.
②	Search area, where the logs can be searched by the input of keywords.
③	Log display page, where basic information such as title, generation time, and content is displayed. The user can press Previous/Next to switch between the previous and next pages.

13.4Log timeline

The log timeline interface displays both HMI and controller log information in chronological order, and this interface includes search area and display page, allowing the user to search for HMI logs and controller logs at different time periods.



①	Search area, where the user can search logs with a specific time range
②	Log display page, where HMI logs are displayed on the left and RC logs on the right

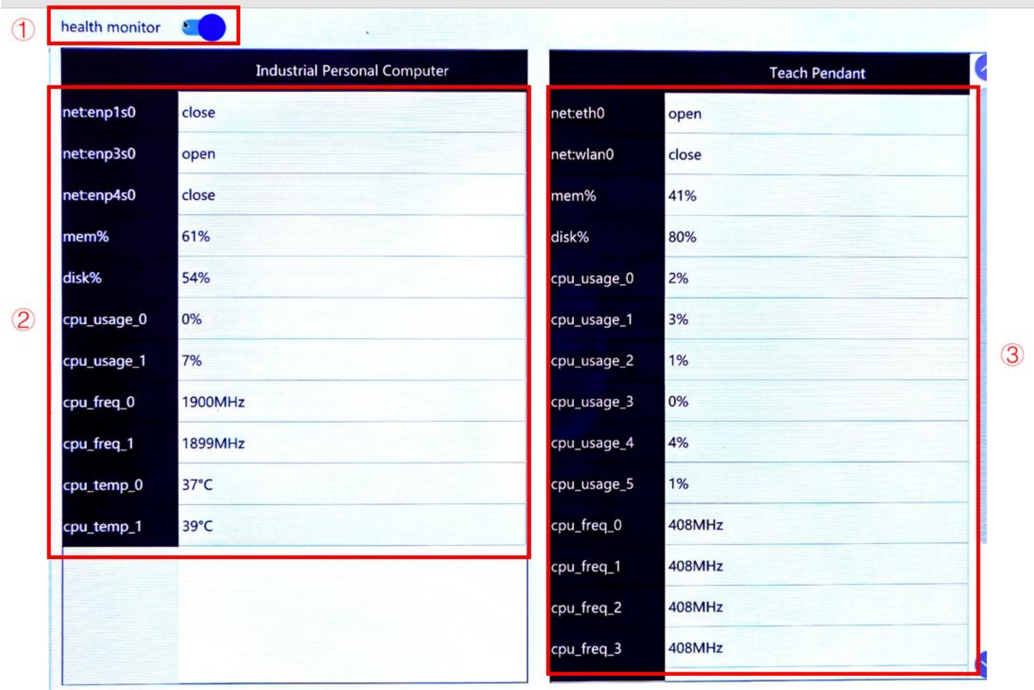
13.5Internal logs

The internal logs interface has functions such as providing a basis for troubleshooting technical issues, robot malfunctions, etc. and pinpointing the cause of problems, and this interface mainly includes search area and display page.

①	Internal logs search area
②	Internal logs display page

13.6Hardware status

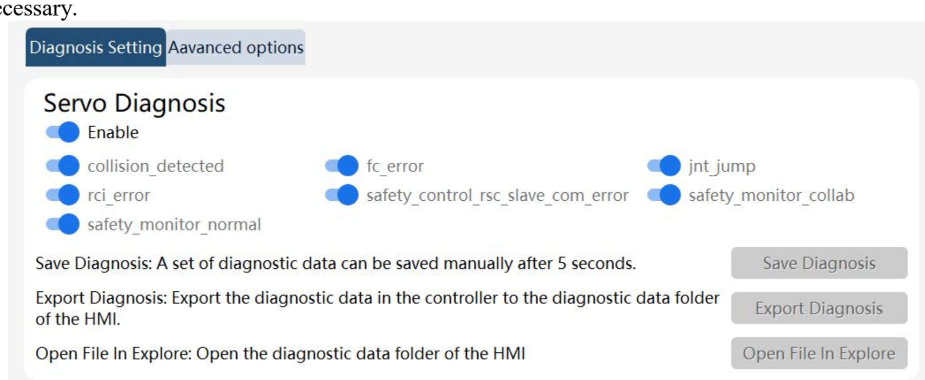
It is used to monitor the hardware status of the IPC and the teach pendant, including memory usage, disk usage, network adapter status, CPU frequency, temperature, and usage. The interface consists of an enable switch, an IPC hardware status display area, and a teach pendant hardware status display area.

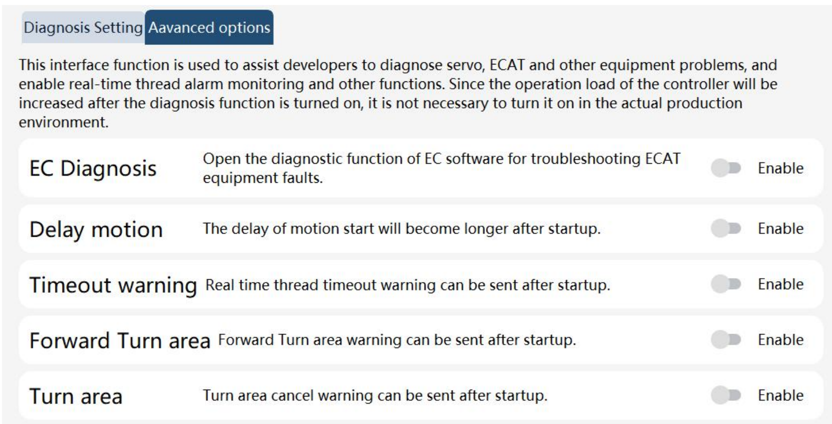


①	Hardware status monitoring switch
②	IPC hardware status display area
③	Teach pendant hardware status display area

13.7Diagnostic setting

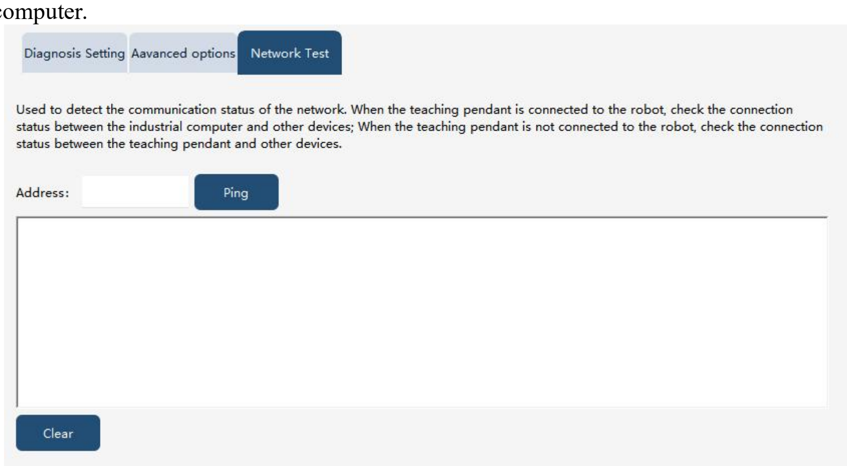
This interface is used to assist developers with the diagnosis of the servo, ECAT, and other equipment, and enable real-time thread alarm and monitoring and other functions. Since enabling the diagnostic function will increase the workload of the controller, do not turn it on in actual production unless necessary.





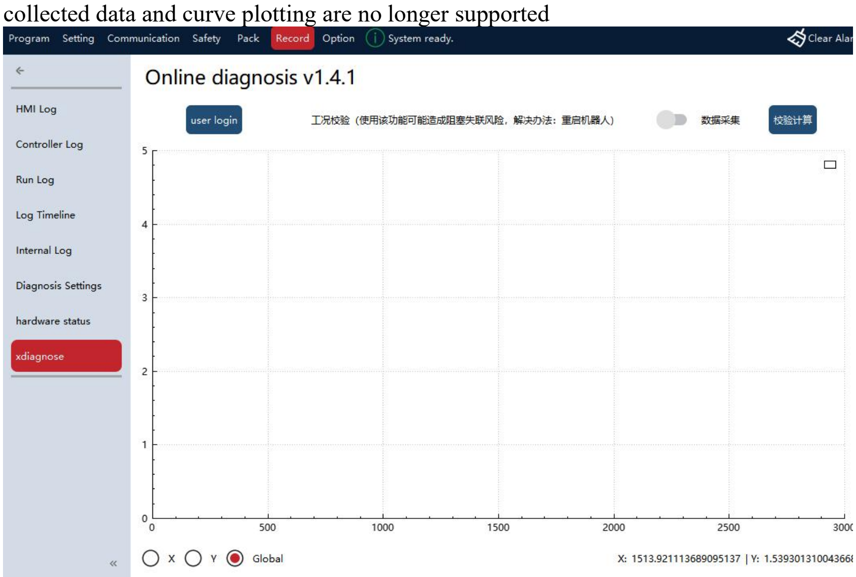
Servo diagnosis	The servo diagnostic module is used to save the data errors in the servo. Click the Save button. The diagnostic data can be exported after 5s.
EC diagnosis	The EC software diagnosis function can be used to assist in troubleshooting ECAT devices.
Timeout warning	It aims to send real-time thread timeout warnings after enabled.
Turning zone	It reports a turning zone warning after it is enabled.
Delay motion	The delay in motion will be prolonged after it is enabled.
Lookahead turning zone	It reports a lookahead turning zone warning after it is enabled;

The network test is used to check the communication status between the industrial control computer of the robot and other devices, verifying whether the network is functioning properly. When the teach pendant is connected to the controller, it performs the controller network test function, which checks the network connection between the controller's industrial control computer and other devices. When the teach pendant is not connected to the controller, it performs the teach pendant network test function, which checks the network connection between the teach pendant and the industrial control computer.



13.8Working condition verification

The working condition verification function assists developers and field personnel in collecting machine status data during RL program execution for analyzing machine issues. This function is supported in the PC version of the HMI but not in the teach pendant HMI. In xCore 3.1, the export of collected data and curve plotting are no longer supported



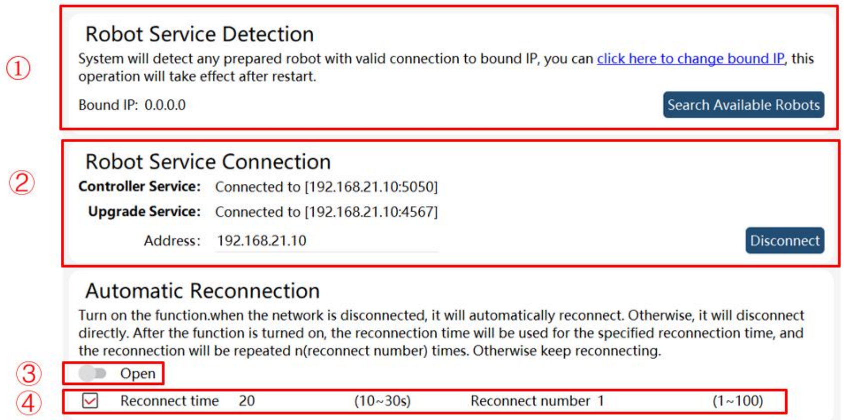
This function is located in the diagnostic data monitoring page of the log module, where users can start data collection by clicking the data acquisition button during RL operation, stop collection by clicking the button again after task completion, and then click verification calculation to obtain the working condition analysis results.

1	0.010497	Normal	1.839600	Normal	0.000000	Normal
2	0.508205	Normal	56.309925	Normal	0.000000	Normal
3	0.187171	Normal	21.128640	Normal	0.000000	Normal
4	0.069840	Normal	5.497049	Normal	0.000000	Normal
5	0.005934	Normal	0.495526	Normal	0.000000	Normal
6	0.001251	Normal	0.721062	Normal	0.000000	Normal

14Options

The options page mainly involves robot connection, software upgrade, import and export, and feature demonstration. Note: Using a USB drive with excessively large capacity may result in slow copying speeds, which can affect backup and export performance.

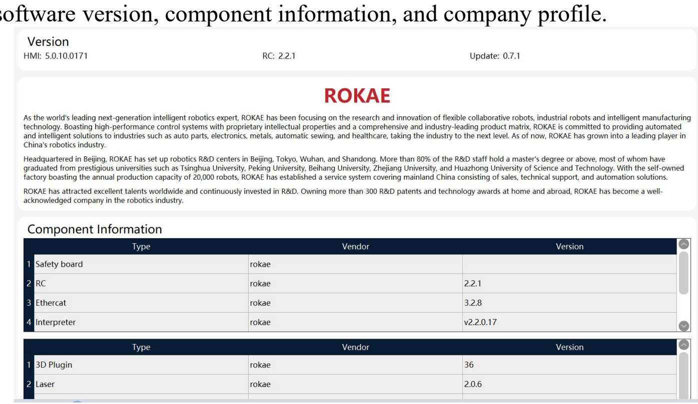
14.1Connect



①	Search for available robots: Search for all robots in the same LAN (except direct connection). When the robot is connected, it will be displayed that the controller service and the update service are both connected. If the robot can not be found when Robot Assist and the robot are in the same LAN, or the real-time position of the robot is not displayed on the 3D interface after connecting the robot, the user can enter the "Settings" - "HMI Settings" interface and select the IP address assigned to the LAN in the Bound IP Address drop-down box to solve the above two problems.
②	Display the connection status, and the user can select to disconnect it.
③	Enable the function of automatic reconnection: When the network between the robot controller and Robot Assist is disconnected, Robot Assist will try reconnection automatically and will stop trying after the preset reconnection time.
④	There are two ways for reconnection: (1) Check the checkbox: the total duration of reconnection = single duration * the number of reconnections. The user can specify a single duration and the number of connections and within the total reconnection duration, Robot Assist will attempt to perform automatic reconnections. (2) Uncheck the checkbox: Robot Assist will keep trying to reconnect the controller.

14.2About ROKAE

It introduces software version, component information, and company profile.



14.3Software upgrade

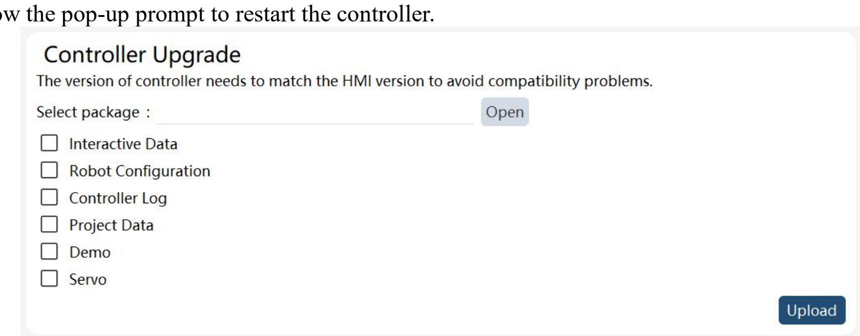
14.3.1Controller upgrade

Through the controller upgrade function, it is possible to: (1) upgrade the controller software version and (2) restore data.

Upgrade controller version: Select the upgrade package, click "Upload", and the interface will prompt "Uploaded successfully". Follow the pop-up prompt to restart the controller.

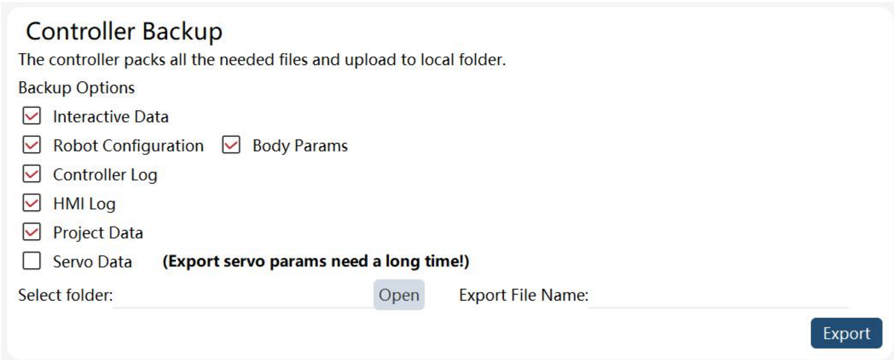
Restore data: Select the data package to be restored, check the data to be restored, and click "Upload".

Follow the pop-up prompt to restart the controller.



14.3.2Controller backup

This function can achieve data backup of the controller. Check the data for backup, click "Open" to select the backup directory, and then click "Export". The exported file is an encrypted file in RPA format.



Robot configuration: controller settings - system configuration; calibration - zero point voltage value of torque sensor; body parameters - overload coefficient; motion parameters - parameters and safety control; etc.

Native parameters: calibration — encoder values, angle calibration settings, and base frame; dynamic settings — (third order) friction coefficient; body parameters — DH parameters, reduction ratio, etc.

Interactive data: dynamic settings - feedforward and constraint switch; motion parameters - stopping distance, Search, and minimum radius of turning zone, Conf, force control parameters, xPanel configuration, quick adjustment, teach pendant mode, communication, and safety.

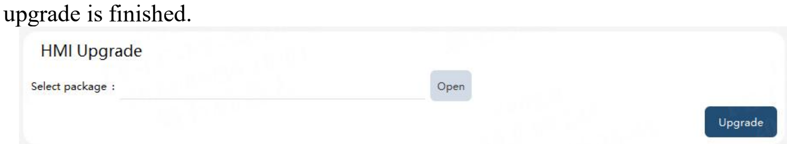
Controller logs: version information files, user configuration files, controller logs, and project logs. Only the logs generated in the last three days can be exported, and more logs can be exported in the log module on the export page.

HMI logs: syslog, mirror version description file, and version information file of the teach pendant.

System logs: IPC syslogs, interfaces files, and rc.local files.

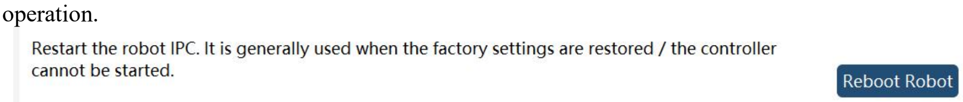
14.3.3HMI upgrade

Only for xPad2, and it aims to upgrade the teach pendant HMI software. Click "Open", select the HMI software upgrade package in the USB drive directory, and click "Upgrade" to start the HMI upgrade process. After the HMI upgrade is completed, the HMI software will start automatically and the HMI upgrade is finished.



14.3.4Restart robot

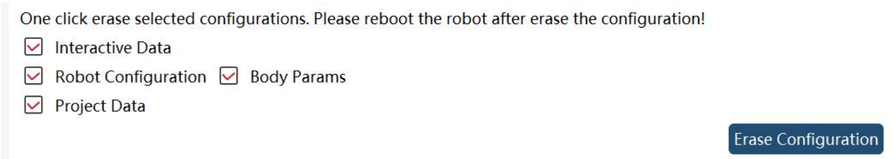
It aims to restart the IPC system, and the upgrade service connection needs to be established for this operation.



14.3.5Erase configuration

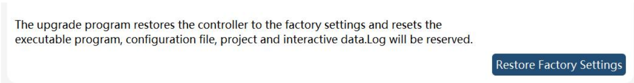
It aims to erase custom configurations, robot configurations, body parameters, project data, etc. For

instructions on the above data, please refer to the "Controller Backup" section. Performing this function will still retain the relevant operation logs of the control system. Please use it with caution.
Check the content to be erased, click "Erase Configuration", and manually restart the robot to take effect.
Note: The upgrade service connection needs to be established for this operation.



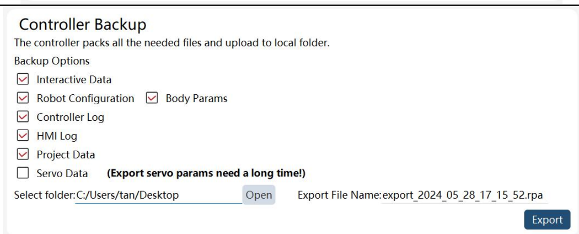
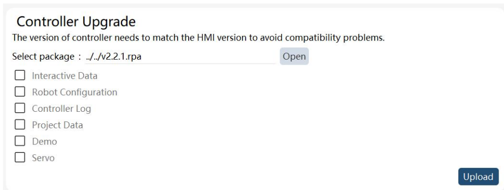
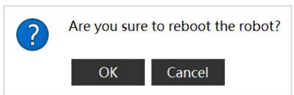
14.3.6Erase all configurations

It aims to restore the control system to its factory default state. This function will reset configuration files, project data, and interactive data within the control system but still retain relevant logs. Please use it with caution.
Note: The upgrade service connection needs to be established for this operation.



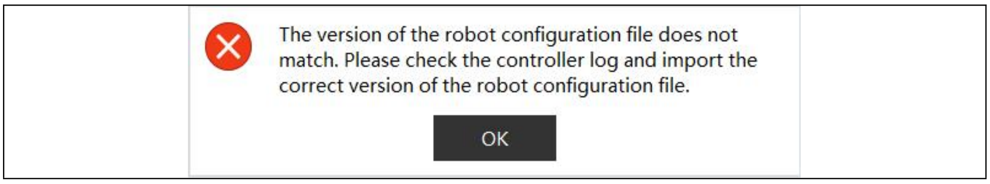
14.3.7Example of control system upgrade

When the user wants to upgrade the control system, contact ROKAE to obtain the new version of the control system installation package. Refer to the operation steps below:

Step	Picture	Explanation
1. Prepare the installation package.		
2. Open the HMI and connect to the controller and the upgrade service.		
3. Before upgrading, it is recommended to back up the control system to avoid the loss of important data. You can check the backup content in the "Backup" option and select "Export Folder" to export the backup.		This step is not mandatory.
4. Select the installation package, and the upgrade option will be configured according to the installation package. Click "Upload".		Do not select "Interactive Data", "Robot Configuration", "Controller Logs", "Project Data", "Demo Case", or "Servo" in the "Controller Upgrade" option.
5. After successful upload, the HMI will prompt to restart the robot and upon the restart, the control system upgrade is completed.		

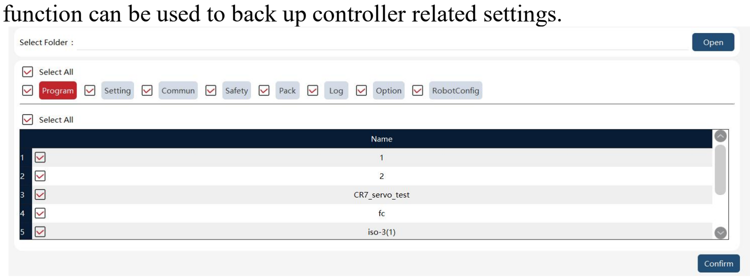
Note

1. When the controller version does not match the HMI version, the HMI will display real-time log information on the top status bar, with the content stating "Version mismatch. Recommend HMI version: [xxx]".
2. When the controller version does not match the configuration file version, the HMI will pop up a warning in the middle after booting up, as shown in the following figure. At this time, any robot motion operation will be disabled, and the correct version of the configuration file needs to be upgraded according to the controller log information.



14.4Export

The export function can be used to back up controller related settings.

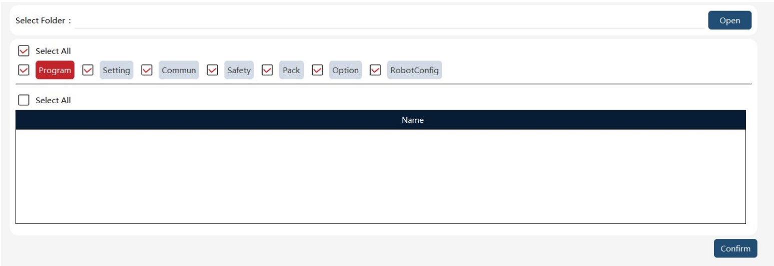


As shown in the figure, select the target folder, select the content to be exported, and click "OK". The available export contents are as follows:

Function Module	Export Content
Programming	Specific individual project
Setting	Controller settings - other settings
	HMI settings - basic settings, theme, and workspace directory
	User group- UserLogin
	Dynamic settings - dynamic feedforward and dynamic constraint
	Global Coordinate System - Global Tools, Global Workpieces, Global Coordinate System
	Force control parameters - force control parameters, force control model, drag optimization
	Motion parameters - motion parameters, AccSet, safety control, Search commad, minimum turning zone, default Conf
	Error code filtering
	Custom buttons
Communication	System IO
	External communication
	IO device
	Bus devices
	End-effector
	RCI settings
	Serial port settings
	Encoder
Safety	Joint limitations
	Robot restrictions
	Collision detection
	Safe region
	Safety position
	Virtual Wall
Process Package	Conveyor belt
Log	Controller logs
	HMI logs
	Project logs
	Sys logs
	Backup Engineering
	Select all controller logs
Options	Connect
Robot configuration	Model file

14.5Import

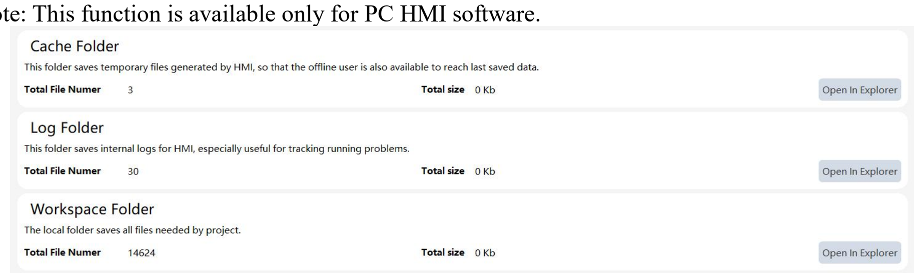
The import function can import controller settings.



Select the zip package you want to import, and after opening the package, the configuration items contained in the zip package will be automatically checked while the configuration items that are not contained will be grayed out.
The user can choose specific import content as the case may be. Note: After importing, the control system needs to be restarted to take effect.

14.6File manager

It is intended to quickly open several folders involved in the Robot Assist software.
Note: This function is available only for PC HMI software.

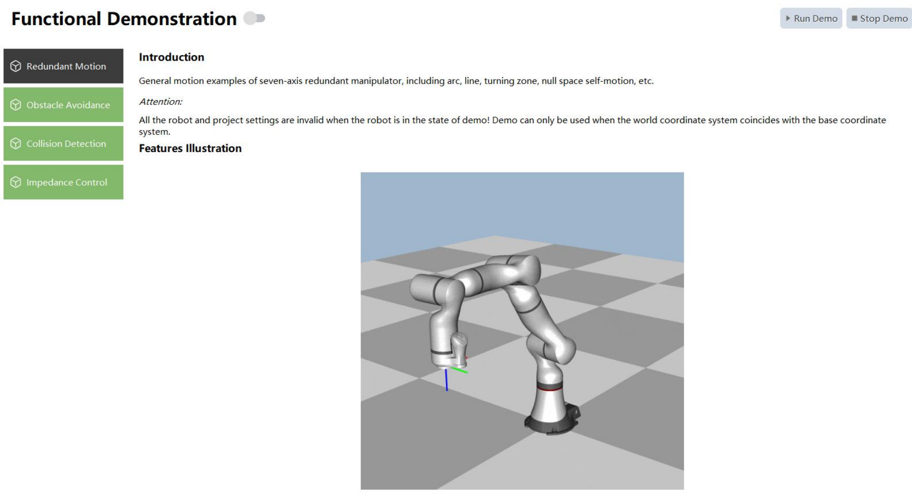


Cache folder: Store the cache of Robot Assist.
Log folder: Store Robot Assist logs. The logs in the folder are consistent with the internal logs on the diagnostic interface. You can click here to enter the folder for log copying.
Workplace folder: Store robot project files.

14.7Demos

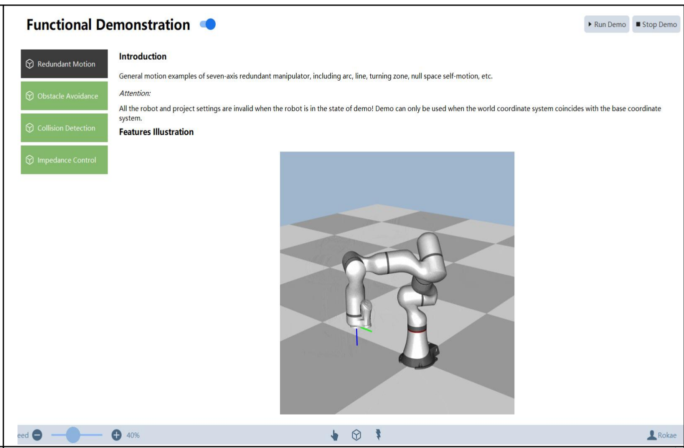
14.7.1Seven-axis redundant motion

It is used for the demonstration of seven-axis redundant motion, including circular motion, linear motion, turning, and null-space motion, and it supports xMate ER PRO.

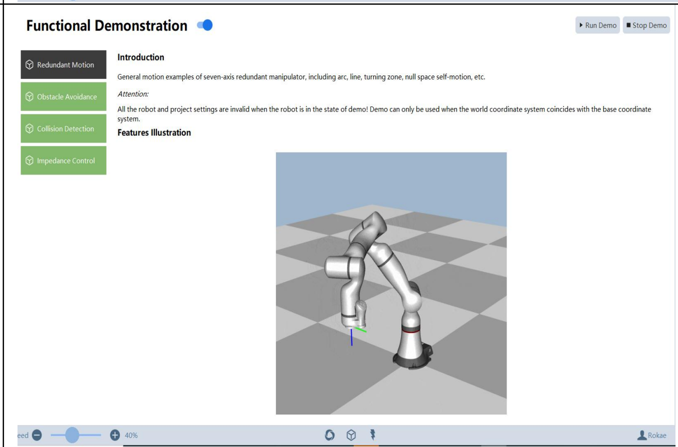


Step	Graphical Representation	Explanation

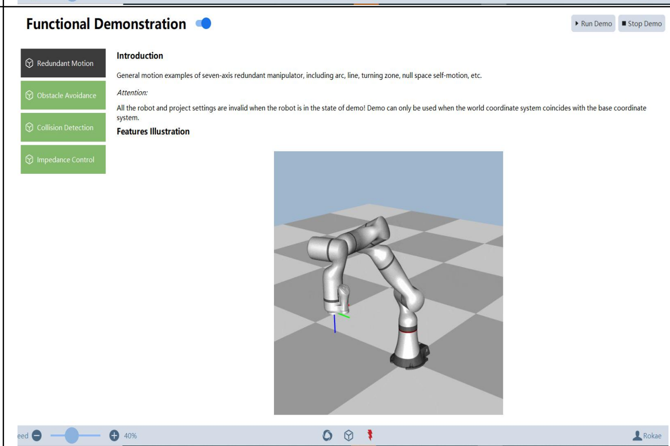
1. Select Option - Demo - Seven-axis redundant motion and enable Demo to enter the demonstration mode.



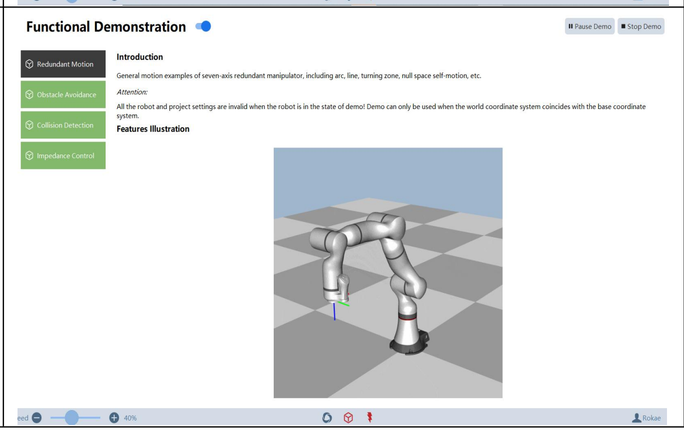
2. Click the mode switching button  on the bottom status bar and switch to Auto Mode.



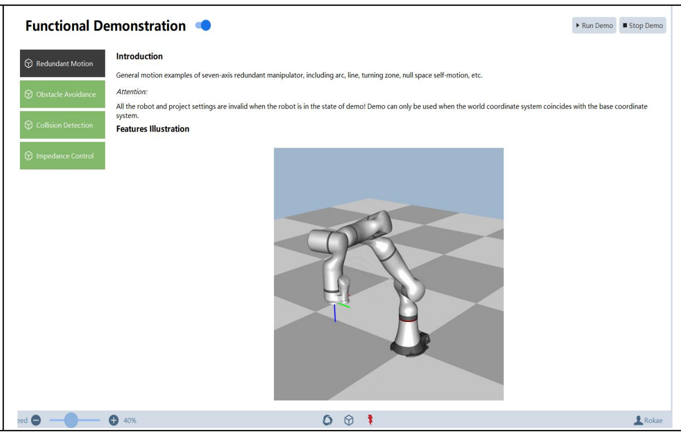
3. Click the Power-on button  on the bottom status bar to power on the robot.



4. Click "Play Demo" in the upper right corner.

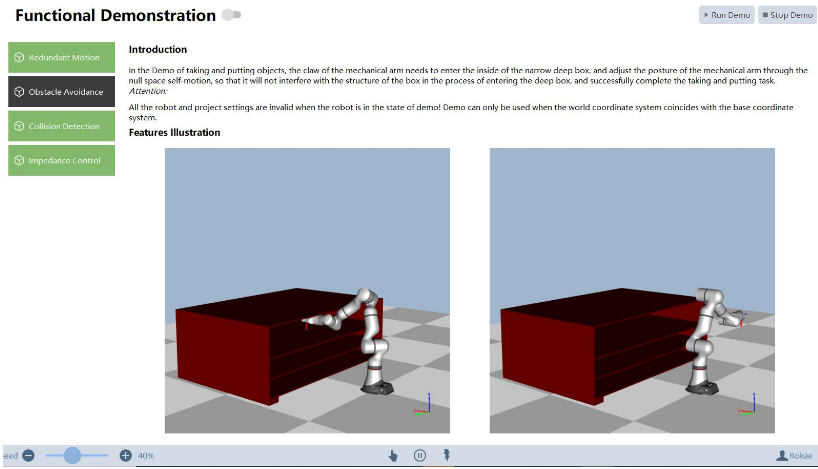


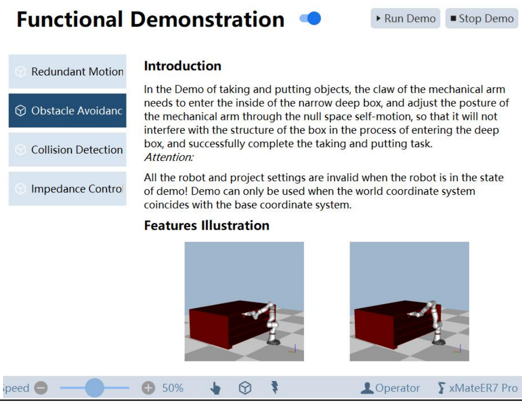
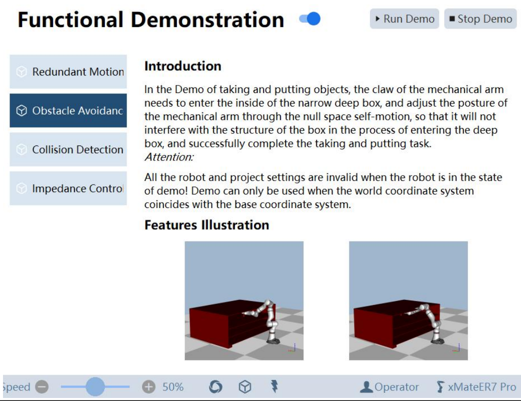
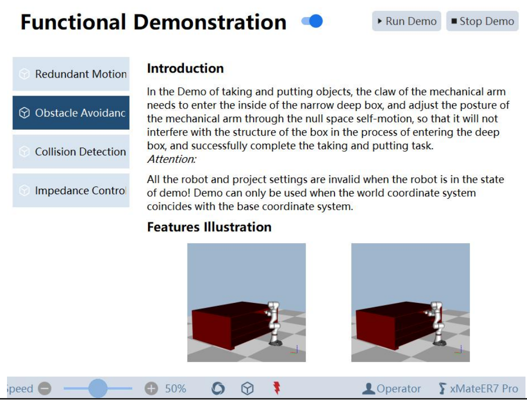
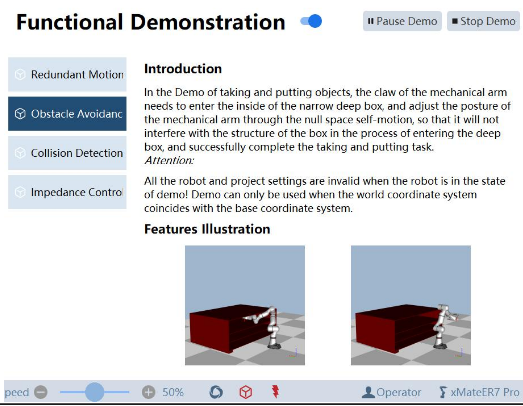
5. After playing the demo, click "Stop Demo" in the upper right corner.

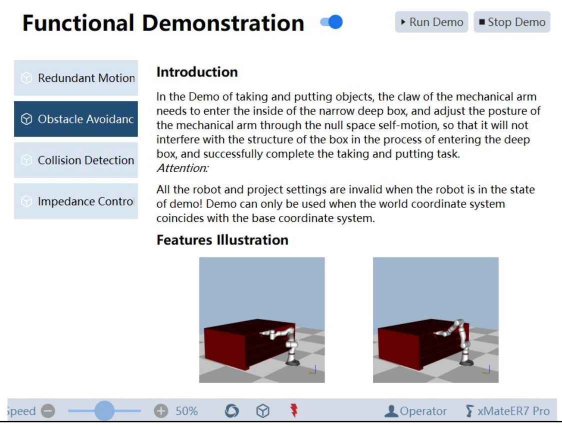


14.7.2Obstacle avoidance

When the demonstration manipulator enters a narrow and deep box, it will not interfere with the box structure by adjusting its orientation through null-space self-motion, which enables the robot to perform the pickup and delivery task successfully.



<table>
<tr><th>Step</th><th>Graphical Representation</th><th>Explanation</th></tr>
<tr><td>1. Select Option - Demo - Obstacle avoidance and enable Demo to enter the demonstration mode.</td><td>
</td><td></td></tr>
<tr><td>2. Click the mode switching button  on the bottom status bar and switch to Auto Mode.</td><td>
</td><td></td></tr>
<tr><td>3. Click the Power-on button  on the bottom status bar to power on the robot.</td><td>
</td><td></td></tr>
<tr><td>4. Click "Play Demo" in the upper right corner.</td><td>
</td><td></td></tr>
</table>

5. After playing the demo, click "Stop Demo" in the upper right corner.	

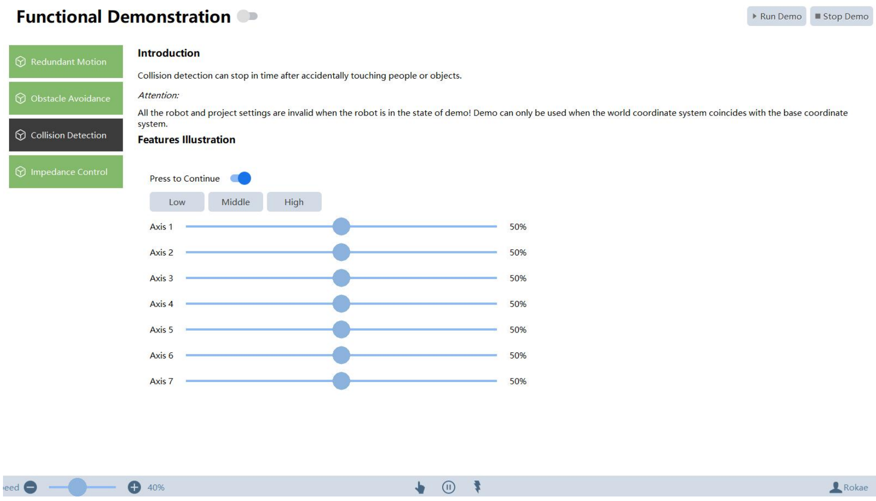
14.7.3Collision detection

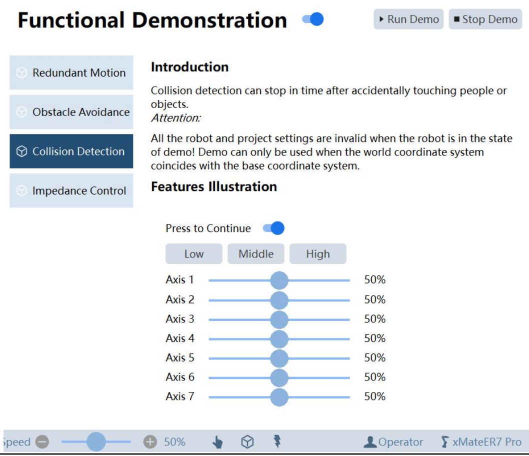
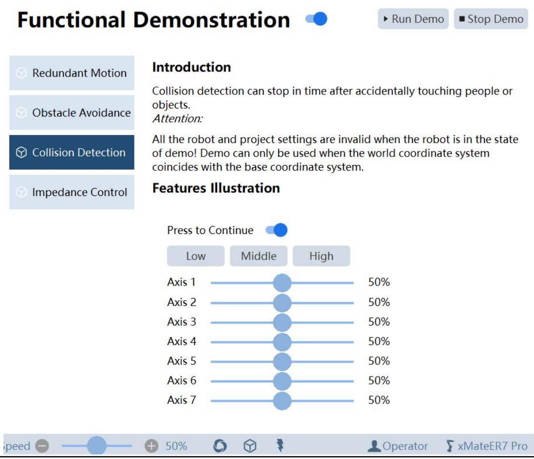
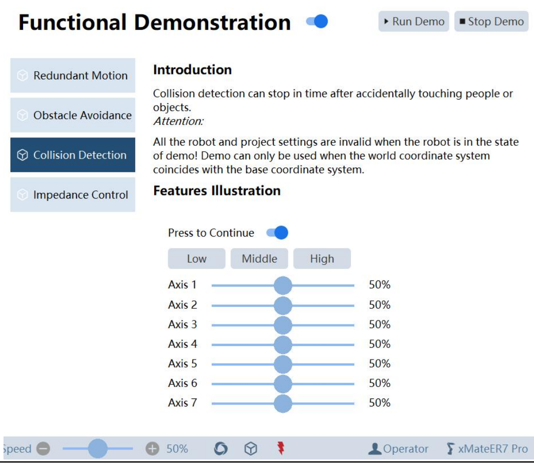
It aims to demonstrate the function of collision detection.

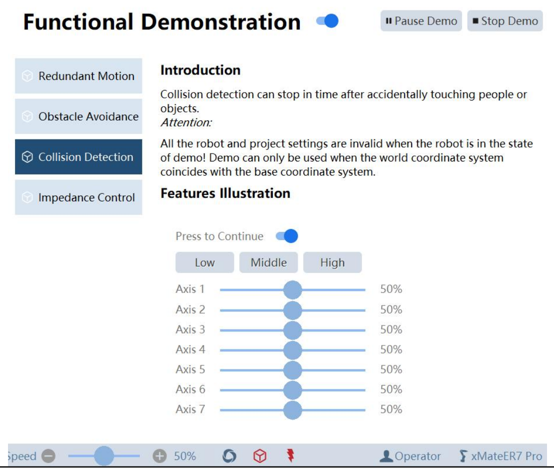
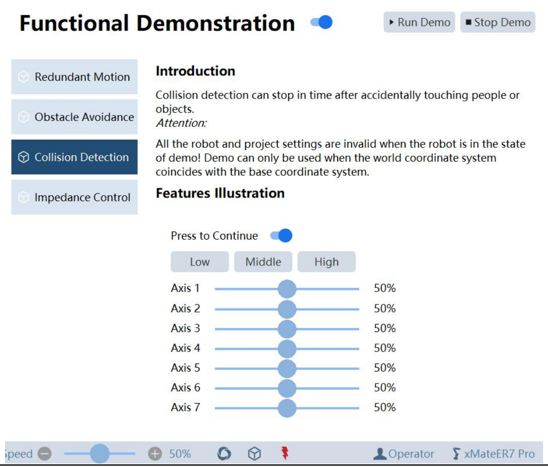
Collision detection sensitivity settings: Support the single-axis setting and the high, medium, or low setting.

When the robot detects a collision, it stops softly. The press to continue is unavailable in version 3.0 collision detection.

When using the demo collision detection function, the collision protection* is temporarily turned off if the collision protection* function is turned on.

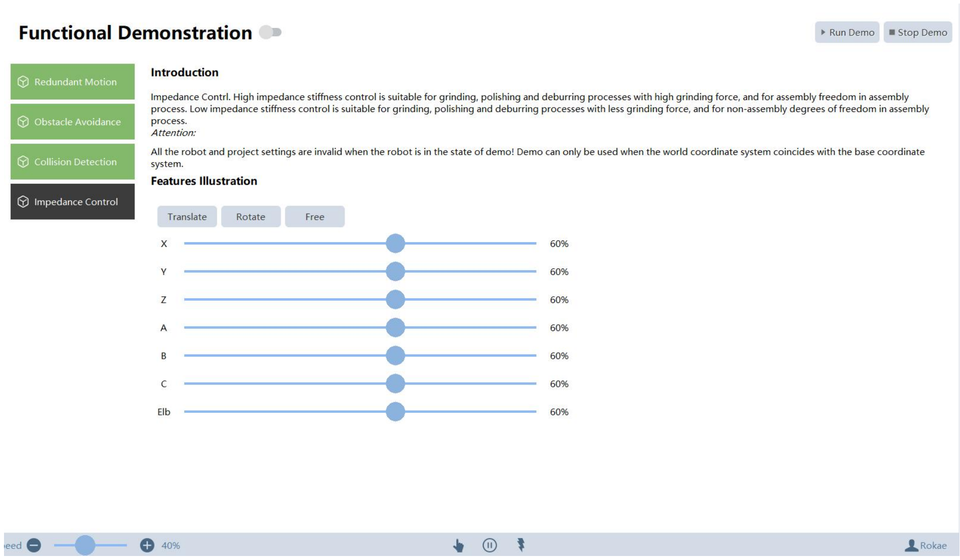


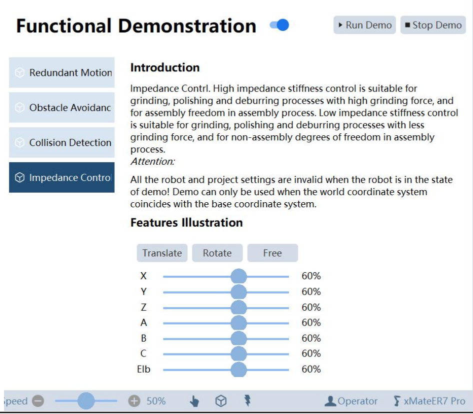
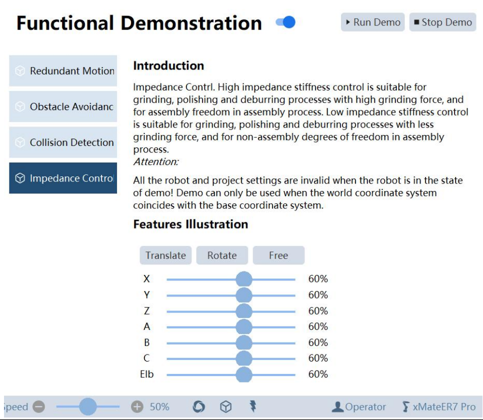
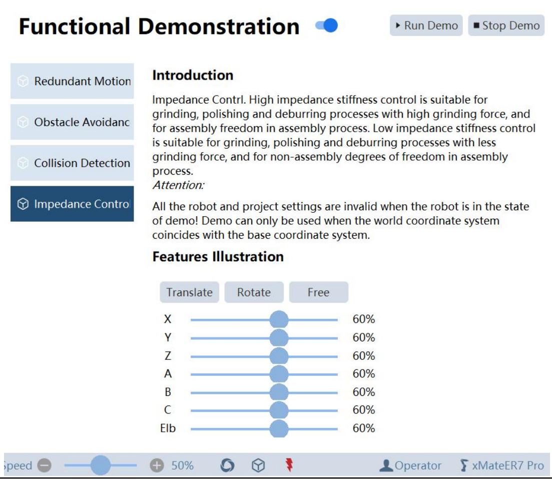
Step	Graphical Representation	Explanation
1. Select Option - Demo - Collision detection and enable Demo to enter the demonstration mode.		
2. Click the mode switching button  on the bottom status bar and switch to Auto Mode.		
3. Click the Power-on button  on the bottom status bar to power on the robot.		

4. Click "Play Demo" in the upper right corner.		
5. After playing the demo, click "Stop Demo" in the upper right corner.		To adjust the threshold of collision detection sensitivity during a demonstration, click "Stop Demo" first and replay the demo after adjustment.

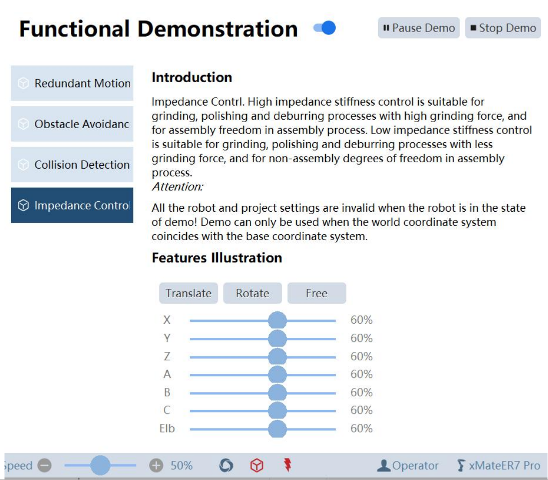
14.7.4Compliance demo

It aims to display the compliance control effect of xMate under different stiffness and spatial conditions.

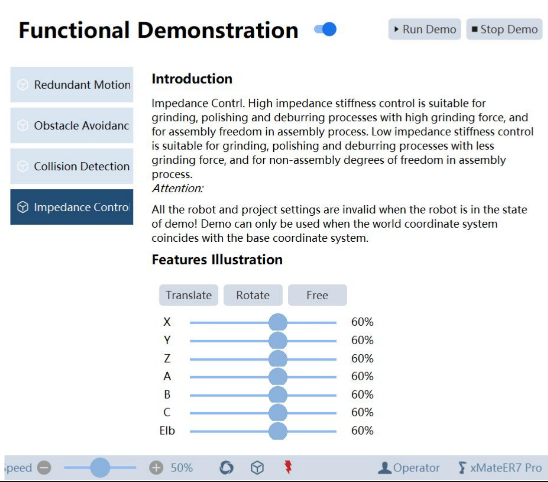


Step	Graphical Representation	Explanation
1. Select Option - Demo - Compliance demo and enable Demo to enter the demonstration mode.		
2. Click the mode switching button  on the bottom status bar and switch to Auto Mode.		
3. Click the Power-on button  on the bottom status bar to power on the robot.		

4. Click "Play Demo" in the upper right corner.



5. After playing the demo, click "Stop Demo" in the upper right corner.



To adjust the stiffness threshold during the Compliance demo, click "Stop Demo" first and replay the demo after adjustment.

Warning

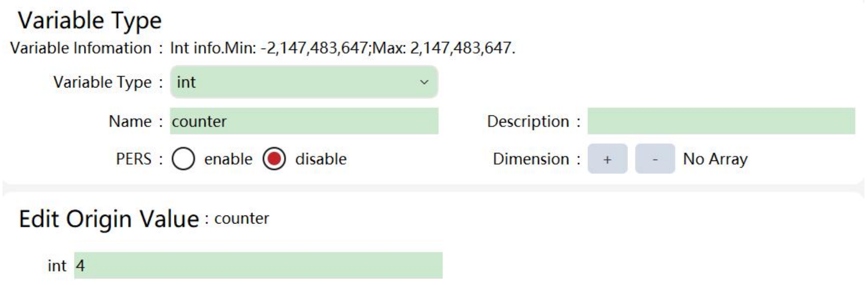
During a demonstration, all configurations of the robot have failed. Please note the following:

1. The base frame of the robot coincides with the world frame by default.
2. There is no load at the robot end-effector by default. Otherwise, the demonstration of collision detection, compliance, and other functions will be affected.

15RL Commands

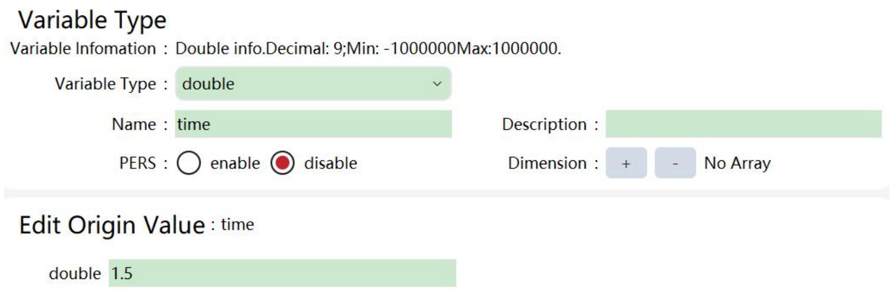
15.1Variable Type

15.1.1Int

<table>
<tr><td>Explanation</td><td>The range of the integer int variable is -2147483647–2147483647. It is recommended that the value is within the specified range. If the value is in excess of the range, it will be assigned randomly, and the maximum value range must not be exceeded when using it.</td></tr>
<tr><td>Example</td><td>In the variable list:


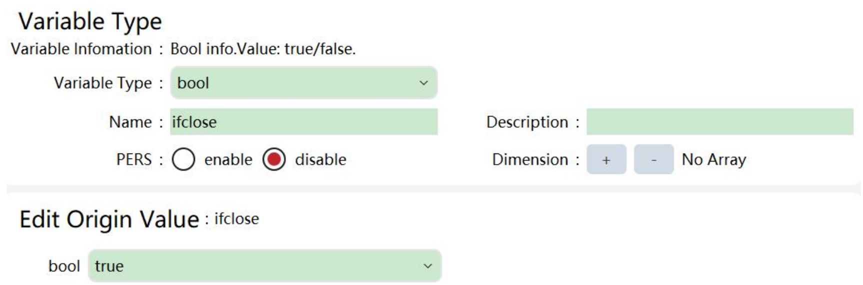
It represents the data counter that defines an integer global variable type, and its initial value is 4.</td></tr>
</table>

15.1.2Double

<table>
<tr><td>Explanation</td><td>Floating-point numbers, are stored using 8 bytes. Do not exceed the value range when using them.</td></tr>
<tr><td>Example</td><td>In the variable list:


It represents the local variable time that defines a floating point, and its initial value is 1.5.</td></tr>
</table>

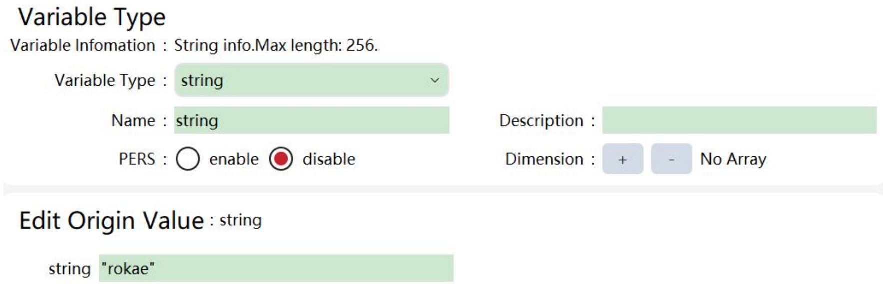
15.1.3Bool

<table>
<tr><td>Explanation</td><td>The variable bool is mainly used for status or logic judgments. The value is true or false.
When it is assigned an int or double value, non-zero takes the value of true and zero takes the value of false.</td></tr>
<tr><td>Example</td><td>In the variable list:


It indicates that a bool type global variable ifClose is defined and the initial value is true.</td></tr>
</table>

15.1.4String

Explanation	String-type variables consist of multiple letters or numbers. Note: They must be placed in double quotation marks "" at the time of defining in RL text.
Example	In the variable list:

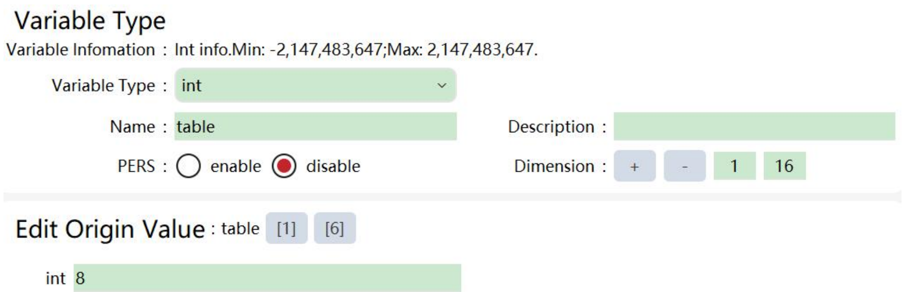
<table>
<tr><td></td><td>


It indicates that a string variable name is defined and initialized to "rokae".

String type variables support the "+" operation for string concatenation, which can achieve string concatenation.
Example:
name = "Rok" + "ae"
It means that the variable name is assigned to "Rokae".
</td></tr>
</table>

15.1.5Array

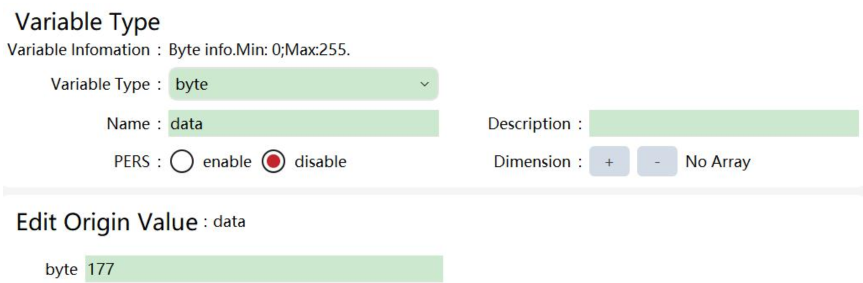
<table>
<tr><td>Explanation</td><td>An array is a collection of variables of the same type, either one-dimensional or multi-dimensional. The elements in the array are accessed using subscripts. The subscript of each dimension begins with 1. The total length of the array should not exceed 1000.</td></tr>
<tr><td>Example</td><td>
In the variable list:



It indicates that a two-dimensional array that contains 16 integer variables is defined. The value of the sixth element of line 1 is assigned to 8.
</td></tr>
</table>

15.1.6byte

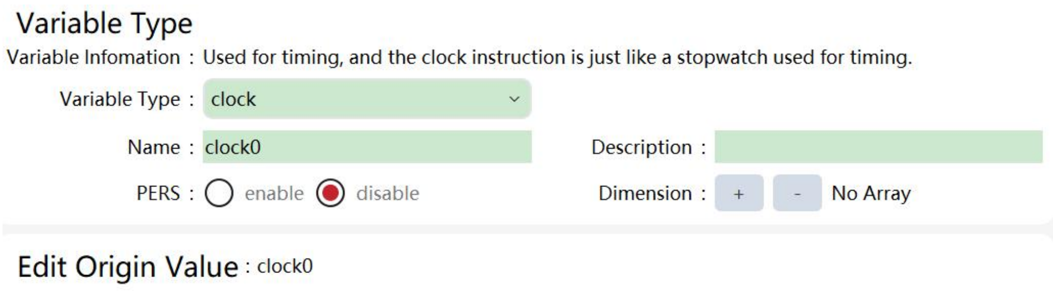
<table>
<tr><td>Explanation</td><td>byte represents the unsigned byte in RL language, same as unsigned char in C++.
The value range is 0−255, and negative values are not allowed. It is generally used in SocketSendByte instruction.
When the byte variable's value exceeds 255, it is automatically truncated, keeping only the lower 8 bits of the value, e.g. var byte data2=288, and the value of data2 is 32 after truncation.</td></tr>
<tr><td>Example</td><td>
In the variable list:



It defines a byte variable data, which has a value of 177.
</td></tr>
</table>

15.1.7clock

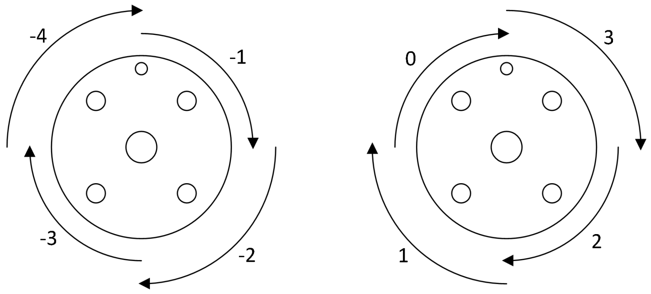
<table>
<tr><td>Explanation</td><td>The clock is used for timing, and clock-related commands are just like a stopwatch used for timing.
The time accuracy of clock type storage is 0.001s, and the maximum time interval is 45 days (i.e., 45 x 24 x 3600 seconds).</td></tr>
</table>

Example	In the variable list:  The following example shows how to use variable clock: Example 1: ClkStart (clock1); ClkStop (clock1); interval=ClkRead(clock1); ClkReset (clock1); Interval (pre-declared double variable) reads the interval between ClkStart and ClkStop, in seconds (s).

15.1.8Implicit type conversion

Explanation	Currently, during data setup in the variable lists, data types are restricted. Values that do not match the variable type cannot be successfully entered, thus avoiding implicit type conversion.
Example	When defining the integer counter in the variable list, no decimals, only integers, can be entered.

15.1.9Confdata

Explanation	The confdata (Robot Configuration Data) is used to define the morphological configuration data that corresponds to the spatial target point. Since the robot mostly uses a rotary joint, any one of the joints exhibits the same status at 1° and 361°. Therefore, after the form of robot is selected, other measures are required to deal with the multiple-loop problem of the joint. For example, when the joint angle is between 0° and 90°, the quadrant number is 0. When the joint angle is between 90° and 180°, it is marked as 1 and is increased/decreased by 1 for every 90°. When the angle is negative, the corresponding number of quadrants is also negative, as shown in the following figure (left: negative joint angle; right: positive joint angle). For robot joints, the angle increases when rotating anticlockwise and decreases when rotating clockwise. In the figure below, the joint angle decreases as the joint rotates clockwise, and the corresponding confdata changes as -1->-2->-3->-4 and 3->2->1->0, respectively. For the xMate CR and SR collaborative robots, we directly record the rounded-down values of each joint angle in confdata. In addition, the same Cartesian space target point corresponds to different inverse kinematics, so it is necessary to use confdata to specify the form to be selected. 
Definition	7 parameters are needed to complete the confdata, including: **cf1**, data type: int, the quadrant that corresponds to the Axis 1 angle or the rounded-down joint angle. **cf2**, data type: int, the quadrant that corresponds to the Axis 2 angle or the rounded-down joint angle. **cf3**, data type: int, the quadrant that corresponds to the Axis 3 angle or the rounded-down joint angle. **cf4**, data type: int, the quadrant that corresponds to the Axis 4 angle or the rounded-down joint angle. **cf5**, data type: int, the quadrant that corresponds to the Axis 5 angle or the rounded-down joint angle. **cf6**, data type: int, the quadrant that corresponds to the Axis 6 angle or the rounded-down joint angle. **cf7**, data type: int, the quadrant that corresponds to the Axis 7 angle or the rounded-down joint angle. **cfx**, data type: int, the configuration number of the form used by the robot, ranging from 0 to 8.
Example	There are different inverse kinematics for the same end-effector Cartesian space pose. The values of cfx from 0 to 8 represent each group of inverse kinematic solutions, which are explained in detail as follows. For 6-axis industrial robots, xMateER collaborative robots, and 7-axis robots:

cfx	Wrist center is on Axis 1...	Wrist center on the lower arm…	Wrist angle is...
0	Front	Front	Positive
1	Front	Front	Negative
2	Front	Rear	Positive
3	Front	Rear	Negative
4	Rear	Front	Positive
5	Rear	Front	Negative
6	Rear	Rear	Positive
7	Rear	Rear	Negative

For xMateCR collaborative robots (cfx=0 represents the solution that is closer to the joint angle represented by cf1–6):

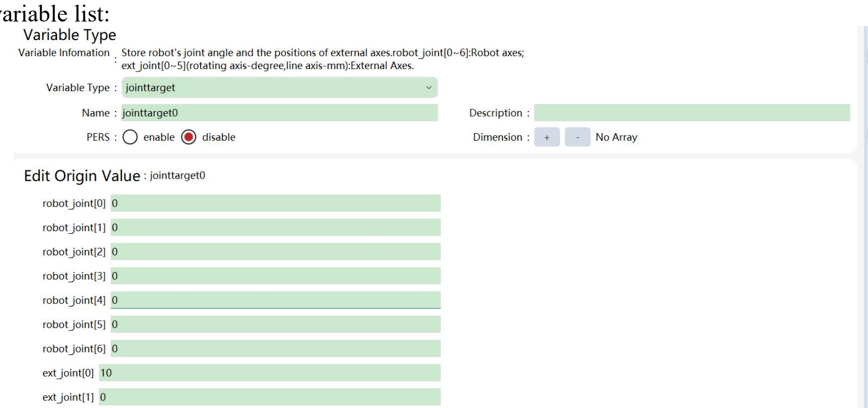
cfx	Wrist center is on Axis 1...	Wrist center on the lower arm…	Axis 5 angle is...
0			
1	Front	Front	Negative
2	Front	Front	Positive
3	Front	Rear	Negative
4	Front	Rear	Positive
5	Rear	Front	Negative
6	Rear	Front	Positive
7	Rear	Rear	Negative
8	Rear	Rear	Positive

For xMateSR collaborative robots, cfx is always 0, indicating the solution that is closer to the joint angle represented by cf1–6.

For 3-axis industrial robots and 4-axis industrial robots:

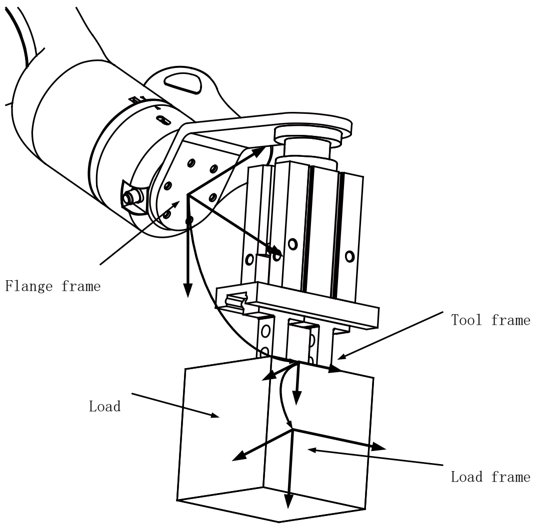
cfx	Wrist center on the lower arm…
0	Front
1	Rear

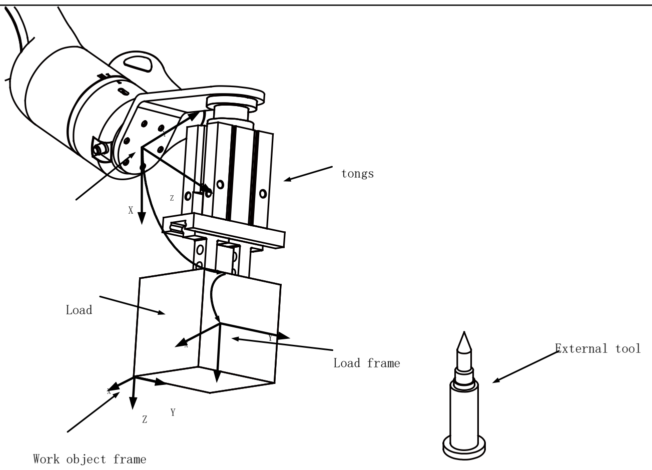
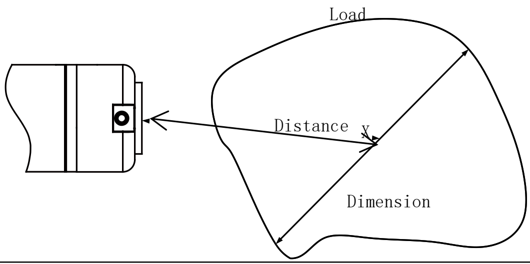
15.1.10jointtarget

<table>
<tr><td>Explanation</td><td>To store the robot’s joint angle and the positions of external axes.</td></tr>
<tr><td>Definition</td><td>robax, Robot Axis, data type: double, containing 7 members of double type, which store the angle of the robot's 7 joints, in Degree.
extax, External Axis, data type: double, containing 6 members of double type, which can store the position of up to 6 external axes. If the external axis is a rotation axis, the unit is Degree; if the external axis is a linear axis, the unit is mm.</td></tr>
<tr><td>Example</td><td>In the variable list:


The above command defines a point named "jointtarget0" in the joint space. Except that the Axis 6 is 90 degrees, the other axes of the robot are all 0 degrees. The first external axis is set to 10 degrees or 10 mm, depending on the type of external axis; the remaining external axes are set to zero.</td></tr>
<tr><td>Structure assignment</td><td>jointtarget j1 = J:{-268.649031, 321.536626, 259.344893, 55.143011, 66.111070, 169.543340, 29.916387}{EJ 0,0,0,0,0,0};
Note:
1. 1. J:{...} represents the angles (in degrees) of the robot's 7 joints, with each value being of type double.
2. {EJ ...} defines the position information of 6 external axes, also of type double. In this example, all external axis values are set to 0.</td></tr>
</table>

15.1.11load

Explanation	The variable type load is used to store the dynamic parameters of the robot's load. There are two main types of robot loads: ● The tool or work object itself installed at the end-effector of the robot; ● Objects that the tool picks up/sucks up. The variable load does not support individual creation. It can only be manually modified in the tool calibration interface as a member of the tool-type variables or automatically modified by the control system using the load identification function. By defining the dynamic parameters of the load correctly, the robot can achieve optimal performance. The wrong definition may lead to the following consequences: ● The robot cannot maximize the ability to use the servo system, resulting in degraded performance. ● The accuracy of the path is reduced, and the positioning error increases. ● Overloading of mechanical components results in a reduction in life or damage.
Definition	In the xCore system, the load is treated as a rigid body. There are four parameters for describing the load: **mass**, data type: double, the mass of the load, in kg. **cogx**, the offset of the center of mass in the X-direction, data type: double, if the tool is mounted on the robot, cogx records the offset of the center of mass in the X direction of the tool frame; if the external tool function is used, the cogx records the offset of the center of mass of the load held by the gripper in the X direction of the work object frame. **cogy**, the offset of the center of mass in the Y-direction, data type: double, if the tool is mounted on the robot, cogy records the offset of the center of mass in the Y direction of the tool frame; if the external tool function is used, the cogy records the offset of the center of mass of the load held by the gripper in the Y direction of the work object frame. **cogz**, the offset of the center of mass in the Z-direction, data type: double, if the tool is mounted on the robot, cogz records the offset of the center of mass in the Z direction of the tool frame; if the external tool function is used, the cogz records the offset of the center of mass of the load held by the gripper in the Z direction of the work object frame. **q1~q4**, quaternions, to record the direction of the principal axis of inertia of the load, data type: double; When the tool is mounted on the robot, the orientation of the principal axis of inertia is described in the tool frame. See the figure below for details:  When using an external tool, the direction of the principal axis of inertia is described in the work object frame. See the figure below:

	 **Ix**, inertia x, data type: double, the inertia of the load along the x-axis, in kgm2. **Iy**, inertia y, data type: double, the inertia of the load along the y-axis, in kgm2. **Iz**, inertia z, data type: double, the inertia of the load along the z-axis, in kgm2. Note: Correctly defining the load inertia helps to improve the robot's movement accuracy, especially when handling large objects. If ix, iy, iz are set to zero, the load will be treated as a mass. Usually, if the distance from the center of mass of the load to the flange center point is smaller than the maximum size of the load itself, the load inertia should be defined, as shown in the following figure: 

15.1.12orient

Explanation	It is used to store the orientation information of the frame or space rigid body. Variables of orient type do not support individual creation or modification and are only used as member variables of some variables.
Definition	The RL language system uses quaternions to represent orientations, so there are a total of 4 components expressed as follows: **q1**, data type: double, the 1st component of the quaternion. **q2**, data type: double, the 2nd component of the quaternion. **q3**, data type: double, the 3rd component of the quaternion. **q4**, data type: double, the 4th component of the quaternion.
Remarks	We usually describe the orientation of the rigid body by using the rotation matrix. The quaternion is another way to describe orientation more concisely. The four components of the quaternion satisfy the following relationship:$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$ The rotation matrix and the quaternion can be converted to one another. It is supposed that there is a rotation matrix R, $R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ then: $q_1 = \frac{\sqrt{r_{11} + r_{22} + r_{33} + 1}}{2}$ \\ $q_2 = \frac{\sqrt{r_{11} - r_{22} - r_{33} + 1}}{2}$ $\quad signq_2 = sign(r_{32} - r_{23})$ $q_3 = \frac{\sqrt{r_{22} - r_{11} - r_{33} + 1}}{2}$ $\quad signq_3 = sign(r_{13} - r_{31})$

	$q_4 = \frac{\sqrt{r_{33} - r_{11} - r_{22} + 1}}{2}$ $\quad signq_4 = sign(r_{21} - r_{12})$

15.1.13pos

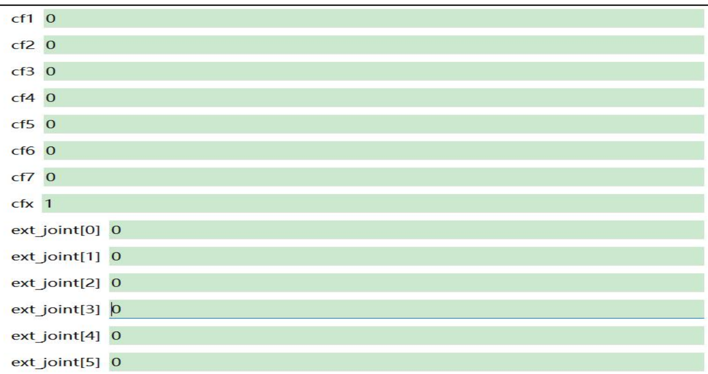
Explanation	It is used to store location information in 3D space. Variables of pos type do not support individual creation or modification and are only used as member variables of some variables.
Definition	The RL language system describes three-dimensional space using the Cartesian frame, so the pos variable has three components: x, y, and z. **X**, data type: double, the X coordinate of the position. **Y**, data type: double, the Y coordinate of the position. **Z**, data type: double, the Z coordinate of the position.

15.1.14pose

Explanation	It is used to store the position and orientation of Cartesian space.
Definition	**X**, data type: double, the X coordinate of the position. **Y**, data type: double, the Y coordinate of the position. **Z**, data type: double, the Z coordinate of the position. **Q1**, data type: double, the 1st component of the quaternion. **Q2**, data type: double, the 2nd component of the quaternion. **Q3**, data type: double, the 3rd component of the quaternion. **Q4**, data type: double, the 4th component of the quaternion.
Structure assignment	pose pose_obj = PE:{{X,Y,Z},{Q1,Q2,Q3,Q4}}; Refer to the above definitions for parameter meanings. Example: pose pose_obj = PE:{{100,100,100},{1,0,0,0}};

15.1.15robtarget

Explanation	It aims to store Cartesian positions and orientations of 3D space, which is used for MoveJ, MoveL, MoveC, and MoveT commands. Because of the multi-solvability of the inverse kinematics of the robot, the robot can arrive in many different forms for the same target pose. In order to specify the configuration form, the robtarget variable also contains the robot configuration data. Variables of the robtarget type are automatically created when the motion command is inserted by auxiliary programming. Manually changing the internal value of the variable may lead to non-correspondence between the Pose and ConfData, and the robot cannot execute the motion command normally. Note: The use of Cartesian positions and orientations in robot programs is defined in the work object frame. If the work object used in the end is not the same as that used during the initial programming, the robot's motion will deviate from the desired path. Therefore, it shall be confirmed that the changes in work object will not cause danger in the following two cases: ● Use the "Modify Command" function to change the wobj parameter of the command; ● The actual work object used is different from the one used in the program commands. Improper use can result in personal injury or equipment damage!
Definition	**Trans**, spatial position, data type: pos, the position offset stored in the reference frame. **Rot**, orientation, data type: orient, the orientation stored in the reference frame. **Conf**, Robot Configuration Data, data type: confdata, to save the configuration data of the robot. Please refer to confdata for details. **Extax**, External Axis, data type: double, containing 6 members of double type, which can store the position of up to 6 external axes. If the external axis is a rotation axis, the unit is Degree; if the external axis is a linear axis, the unit is mm.
Example	

<table>
<tr><td></td><td>

A Cartesian space pose named p1 with the position and orientation (in quaternions) as shown above is defined. The elbow is 10°, and the angles of the Axis 1, 3, 5, and 7 are between 0 and 90°. The robot belongs to the first group of morphological configurations (see confdata for details), and all external axes are in zero.</td></tr>
<tr><td>Structure assignment</td><td>robtarget rob1 = p:{{849.572593, -347.654631, 35.341636},{-0.361604, 0.078279, 0.640346, -0.673106}, -1.058584}{cfg 1,2,3,4,5,6,7,8}{EJ 1,2,3,4,5,6};
Explanation: p:{{Space Position Trans}, {Orientation Rot}, {Arm Angle}}{cfg: Robot Configuration Data Conf}{EJ: External Axis Information Extax};</td></tr>
</table>

15.1.16signalxx

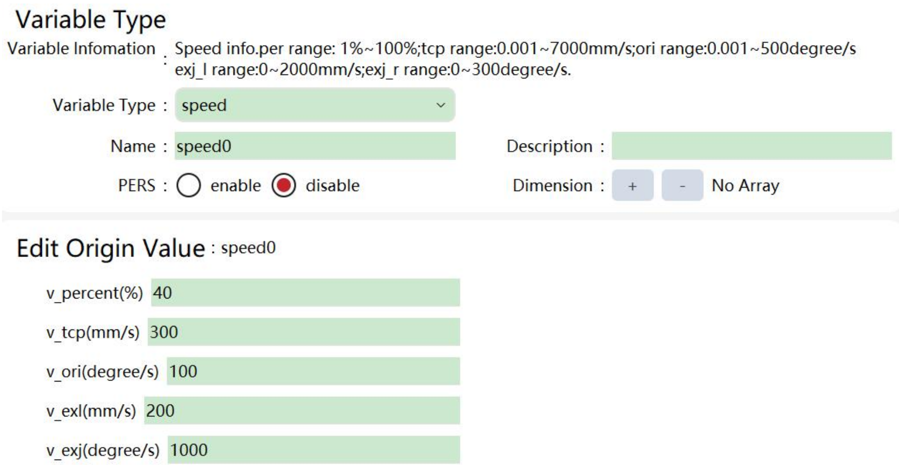
<table>
<tr><td>Explanation</td><td>signalxx type variables are used to describe I/O signals.
All signalxx type variables need to be defined in the "Input/Output" and then used in the program. Direct declaration in the program is not supported.
signalxx currently only supports digital input and output, including the following variable types:
<table>
<tr><th>Variable Type</th><th>Used to describe...</th><th>Explanation</th></tr>
<tr><td>signaldi</td><td>Digital input signal</td><td>The value is True or False, and only indicates the status</td></tr>
<tr><td>signaldo</td><td>Digital output signal</td><td>The value is True or False and is assigned to output</td></tr>
<tr><td>signalgi</td><td>Digit group input signal</td><td>A segment of continuous physical input port is defined as a binary number that can be converted to decimal for use in RL. It supports up to 16 DIs to constitute the group input. Therefore, the value of signalgi ranges from 0 to (2^n -1), with n as the number of DI points contained in group input</td></tr>
<tr><td>signalgo</td><td>Digit group output signal</td><td>A segment of continuous physical output port is defined as a binary number that can be converted to decimal for use in RL. It supports up to 16 DOs to constitute the group output. Therefore, the value of signalgo ranges from 0 to (2^n -1), with n as the number of DO points contained in the group output</td></tr>
</table>
The signaldo and signalgo types contain only signal references and can be assigned using separate commands (e.g., SetDO, SetGO, etc.).
signaldi and signalgi can be used to directly obtain the value of the corresponding input signal in the program.

Note:
● It is not supported to define/declare variables of type signalxx in the program. If such usage occurs, the program will report an error. Before using variables of signalxx type, please configure them in the IO signal list.
● The scope of the signalxx variable is System, and its priority, when compared with other scope types, is System > GLOBAL > LOCAL.
● If the variables declared in the Signal of the IO configuration interface and in the RL programs have the same name, the variable of scope in a lower level will be selected.</td></tr>
<tr><td>Example</td><td>Example 1
//Use the state of the digital input as a criterion for judgment
IF（di1 == true）
do something…
ENDIF

Example 2
//Use the state of the digital input as a criterion for judgment. For example, if the definition group input gi2 maps the first three bits of the 1st byte of Profinet IO, then when the values of bit0 to bit2 are 0, 1, and 1, the value of gi2 is 110 (6 after being converted to int). The same goes for group output (signalgo) as well.
IF（gi2 == 6）
   do something…</td></tr>
</table>

	endif

15.1.17speed

<table>
<tr><td>Explanation</td><td>It is used to define the speed of the robot and the external axes.
For users' convenience, the system presets the commonly used speed variables, which can be directly selected through auxiliary programming. For details, please refer to Insert Command.</td></tr>
<tr><td>Definition</td><td>The speed-type variable contains 5 member variables: Joint Velocity Percentage, TCP Linear Velocity, Orientation Velocity, External Axis Linear Velocity, and External Axis Angular Velocity.
Joint Velocity Percentage, data type: double, to specify the motion speed when the joint movement command is applied, applicable to the commands MoveAbsJ and MoveJ, with the value ranging from 1% to 100%.
TCP Linear Velocity, data type: double, to define the linear velocity of the TCP, with the value ranging from 0.001 mm/s to 1000000 mm/s.
Orientation Velocity, data type: double, to define the rotation speed of the tool, with the value ranging from 0.001 degrees/s to 1000000 degrees/s.
External Axis Linear Velocity, data type: double, to define the motion speed of the external linear axis, with the value ranging from 0 mm/s to 1000000 mm/s.
External Axis Angular Velocity, data type: double, to define the motion speed of the external rotary axis, with the value ranging from 0 degrees/s to 1000000 degrees/s.</td></tr>
<tr><td>Example</td><td>In the variable list:


The image above shows a definition of a speed variable named speed0, in which the joint rotation speed is 40% of the maximum allowable speed, the TCP linear speed is 300 mm/s, the space rotation speed is 100°/s, the external axis angular velocity is 1000°/s, and the external axis linear velocity is 200 mm/s.</td></tr>
<tr><td>Structure assignment</td><td>speed speed0 = v:{40,300,100,1000,200};
Explanation: The image above shows a definition of a speed variable named speed0, in which the joint rotation speed is 40% of the maximum allowable speed, the TCP linear speed is 300 mm/s, the space rotation speed is 100°/s, the external axis angular velocity is 1000°/s, and the external axis linear velocity is 200 mm/s.</td></tr>
</table>

Name	Joint Velocity Percentage	TCP Linear Velocity	Orientation Velocity	External Axis Angular Velocity	External Axis Linear Velocity
v5	1%	5 mm/s	200°/s	1000°/s	5000 mm/s
v10	3%	10 mm/s	200°/s	1000°/s	5000 mm/s
v25	5%	25 mm/s	200°/s	1000°/s	5000 mm/s
v30	5%	30 mm/s	200°/s	1000°/s	5000 mm/s
v40	5%	40 mm/s	200°/s	1000°/s	5000 mm/s
v50	8%	50 mm/s	200°/s	1000°/s	5000 mm/s
v60	8%	60 mm/s	200°/s	1000°/s	5000 mm/s
v80	8%	80 mm/s	200°/s	1000°/s	5000 mm/s
v100	10%	100 mm/s	200°/s	1000°/s	5000 mm/s
v150	15%	150 mm/s	200°/s	1000°/s	5000 mm/s
v200	20%	200 mm/s	200°/s	1000°/s	5000 mm/s
v300	30%	300 mm/s	200°/s	1000°/s	5000 mm/s

v400	40%	400 mm/s	200°/s	1000°/s	5000 mm/s
v500	50%	500 mm/s	200°/s	1000°/s	5000 mm/s
v600	60%	600 mm/s	200°/s	1000°/s	5000 mm/s
v800	70%	800 mm/s	200°/s	1000°/s	5000 mm/s
v1000	100%	1000 mm/s	200°/s	1000°/s	5000 mm/s
v1500	100%	1500 mm/s	200°/s	1000°/s	5000 mm/s
v2000	100%	2000 mm/s	200°/s	1000°/s	5000 mm/s
V3000	100%	3000 mm/s	200°/s	1000°/s	5000 mm/s
v4000	100%	4000 mm/s	200°/s	1000°/s	5000 mm/s
v5000	100%	5000 mm/s	200°/s	1000°/s	5000 mm/s
v6000	100%	6000 mm/s	200°/s	1000°/s	5000 mm/s
v7000	100%	7000 mm/s	200°/s	1000°/s	5000 mm/s
vmax	100%	infinite	200°/s	1000°/s	5000 mm/s

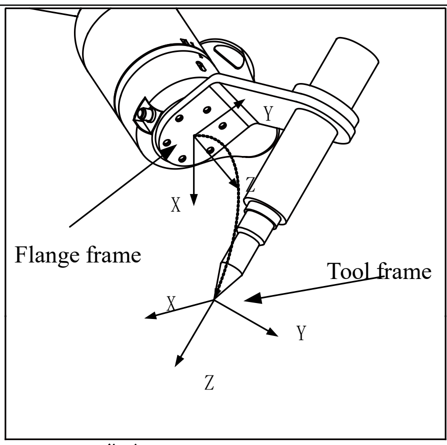
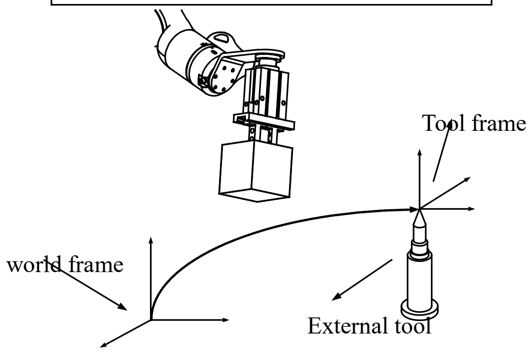
The system predefines some common speed variables, as shown in the following table.

Note

All space rotation speeds in the system's pre-defined speed variable are 200°/s. If there are special requirements on the rotation speed of the end-effector of the robot, a new speed variable can be defined for use according to the process requirements.

15.1.18tool

Explanation	The tool-type variables are used to record tool parameters, including TCP, orientation, and dynamic parameters of the tools used by the robot. The robot uses tools to interact with the outside world, so the tool variable will affect the motion of the robot from the following aspects: Only the TCP will move according to the programmed path and speed. When the robot executes a pure spatial rotation, only TCP will remain motionless; The motion path and speed specified during programming refer to the path and speed of the tool frame relative to the work object frame. Therefore, replacing a well-calibrated tool or work object does not affect the shape and speed of the path; When using external tools, the speed of programming refers to the speed of a work object (relative to external tools). Note that when using the external tool, tframe in the tool-type variable will record the zero position and orientation offset of the external tool, while tload will record the dynamic parameters of the gripper that is installed at the end-effector of the robot for grasping work object. The data of the tool-type variable is stored in the database. When the program is loaded, it is read by the program editor from the database. Therefore, do not try to modify the tool-type variable directly in the program editor, and thus the unpredictable errors will be avoided. If you need to modify the tool-type variable, please modify it through the calibration interface. See the Calibration of the tool frame for details. Be sure to correctly define the dynamic parameters of the end-effector load of the robot, including the tool itself and the two parts of the object captured by the tool. The wrong definition may lead to the following consequences: ● The robot cannot maximize the ability to use the servo system, resulting in degraded performance; ● The accuracy of the path is reduced, and the positioning error increases; Overloading of mechanical components results in a reduction in life or damage.
Example	**Robhold**, data type: boot, to define whether the tool is installed on the robot. True indicates that the tool is installed on the robot. False indicates that the tool is not installed on the robot and an external tool is being used. When making a jog or executing a program, only one of the robhold parameters can be True in the tool/work object combination used at the same time. That is, if the robhold of the tool is True, the corresponding work object robhold must be false, and vice versa; otherwise, the robot will prompt an error, and it is impossible to make a jog or execute the corresponding program command. **Tframe**, Tool Frame, data type: pose, to record the tool frame of the tool used, including: TCP represents the offset in the x, y, and z directions relative to the robot end-effector flange frame, in millimeters. The orientation offset of the tool frame relative to the flange frame is expressed in quaternion. See the following figure for details:

	  **Tload**, dynamic parameters of the tool, data type: load, to record the dynamic parameters of the tool. For the common tool, tload describes the dynamic parameters of the entire tool. For external tools, tload describes the dynamic parameters of the gripper used by the robot (holding the work object). For general tools installed on the robot, the load parameters include: The mass of the tool (weight), in kg; The center of gravity of the tool, described in the flange frame, in millimeters (mm); the direction of the principal axis of inertia, described in the flange frame; and The inertia magnitude of the tool along the principal axis of inertia, in kgm2. If all inertia components are defined as 0 kgm2, the tool is treated as a Point Mass. Note: When using the external tool function, the TCP and orientation are defined relative to the world frame. If the robot is using an external tool, then the tload member is used to record the dynamic parameters of the gripper installed on the robot. The meaning of the specific parameters remains unchanged. Please note that the tload members only define the dynamic parameters of the gripper used by the robot (holding the work object). The dynamic parameters of the gripped work object are not included. To ensure that the robot performs optimally under all circumstances, you need to define two tool variables to handle this situation: ● A tool saves all parameters of the gripper itself; ● Another tool saves all parameters of the gripper + gripped work object; The use of different tools in the motion command would help implement the switching function with or without load.
Structure assignment	tool tool0 = {whether the tool is handheld,{{tool position},{tool orientation quaternion}},{mass,{center of gravity X,center of gravity Y,center of gravity Z},{load orientation quaternion},inertia ix,inertia iy,inertia iz}}; whether the tool is handheld: true, handheld; false, external. Example:tool tool0 = {true,{{0,0,0},{1,0,0,0}},{0,{0,0,0},{1,0,0,0},0,0,0}};

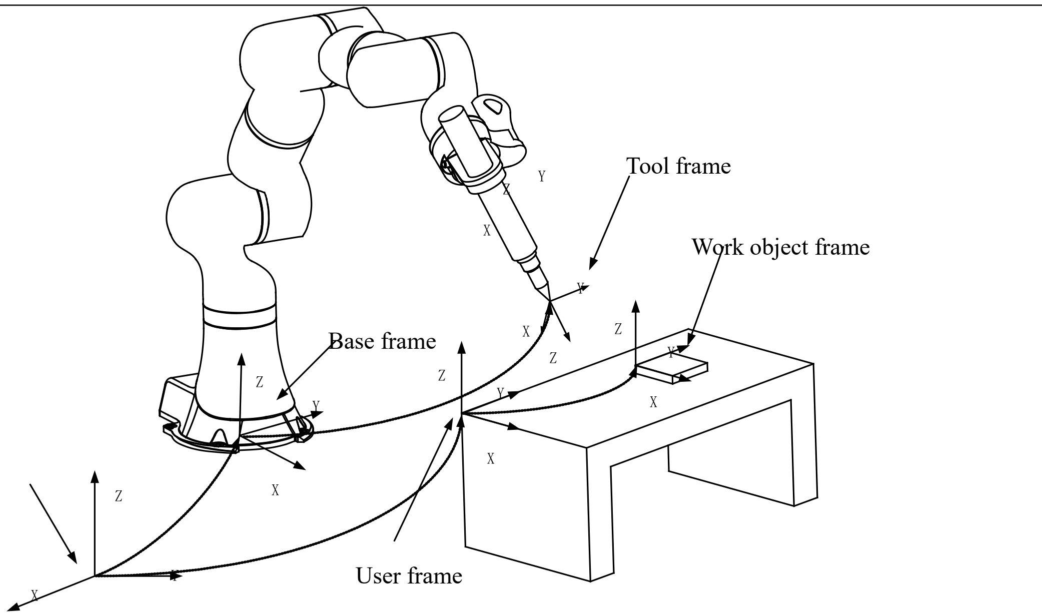
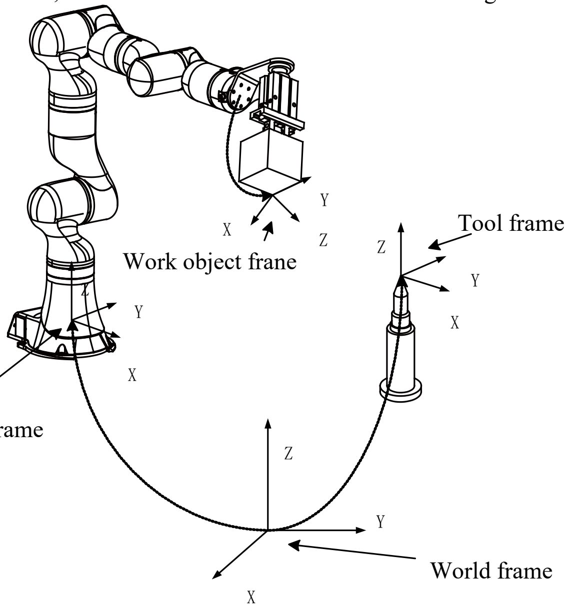
15.1.19Trigdata

Explanation	trigdata is used to store information data about the trigger events during robot motion, including trigger conditions and trigger actions. The trigger condition is usually reaching a specified location on the path; the trigger action can be setting IO, setting variables, etc. Variables of type trigdata cannot be defined by the assignment operator and can only be defined by a specific RL command, so the information stored in each trigdata variable depends on the Trig command as used, for example, the TrigIO, etc.

	Then, it can be used by the corresponding movement commands TrigL, TrigC, TrigJ, etc.
Example	The following example shows how to use the trigdata: Example 1 VAR trigdata gripopen; TrigIO(gripopen,0.5,do1,true); TrigL(p1,v500,gripopen,fine,tool1);

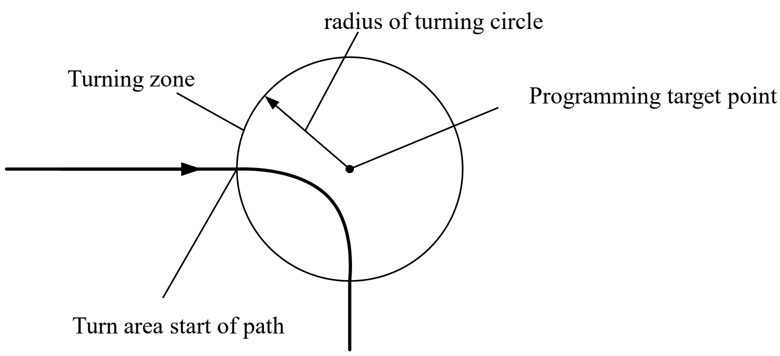
15.1.20wobj

Explanation	wobj is an abbreviation for Work Object. Work object refers to an object processed, handled, or transported by a robot. All the positions used in the motion command are defined in the work object frame (if no work object frame is specified, it defaults to the world frame. The world frame can be seen as a wobj0). There are several benefits in doing this: ● The location of many processing points can be obtained from the design drawing of the work object and used directly; ● When the robot is reinstalled or the work object is moved, you only need to re-calibrate the work object frame to reuse the previous program and avoid reprogramming. ● With a suitable sensor provided, vibrations or slight movements of the work object can be automatically compensated. Under normal circumstances, if you do not define a specific work object frame, the control system will then regard the world frame as the default work object frame wobj0. However, when using external tools, the work object frame must be defined because the programming path and speed refer to the path and speed of the work object, rather than the tool. Usually, the work object frame is defined relative to the user frame, but if the user does not specify a user frame, the work object frame is defined by default relative to the world frame. For details, see the Robot's frames. The work object actually consists of two frames, the user frame and the work object frame. Inserting a user frame at the upper layer of the work object frame is to support the situation where multiple identical work objects need to be machined. For an explanation of the defining relationships of the relevant coordinates, see the explanation of oframe in the "Definitions" section.
Definition	**Robhold**, to define whether the work object is mounted on the robot. True indicates that the work object is mounted on the robot and the external tool is currently being used. False indicates that the work object is not mounted on the robot and the normal tool is currently being used. **Ufprog**, User Frame Programmed, data type: bool, to define whether the user frame is fixed or moving. True indicates that the user frame is fixed, False indicates that the user frame is moving, e.g., to define whether it is on an external positioner or another robot. This value is mostly used when the robot is required to coordinate its movement with the positioner or other robots. **Ufmec**, User Frame Mechanical Unit, data type: string, the mechanical unit name is used to specify which mechanical unit the user frame is bound to; it is useful only if ufprog is false. **Oframe**, Work Object Frame, data type: pose, to store the origin and orientation of the work object frame. **uframe_id**, User Frame ID, data type: int, to store the id of the user frame. The corresponding user frame can be found by id. When using normal tools (non-external tools), the frame definition chain is as follows: The work object frame is defined relative to the user frame; the user frame is defined relative to the world frame.

	 When using external tools, the frame definition chain is as follows: The work object frame is defined relative to the user frame; the user frame is defined relative to the flange frame. 
Structure assignment	wobj wobj0 = {Robhold, Ufprog, Ufmec, Oframe, uframe_id, { mass, {center of gravity X, center of gravity Y, center of gravity Z}, {load orientation quaternion}, inertia ix, inertia iy, inertia iz }}; Refer to the "Definition" section of this table for parameter meanings. Example:wobj wobj0 = {false,true,"robot",{0,0,0},{1,0,0,0},0,{0,{0,0,0},{1,0,0,0},0,0,0}};

15.1.21zone

Explanation	The zone variable is used to define how a certain motion ends, or to define the size of the turning zone between two motion trajectories. For the same target point of robot commands, there are two processing methods in the motion command: ● When it is processed as a stop-point, the robot will move to the target point and reach the target point at a speed of 0 before continuing to execute the next command; ● When it is processed as a transition point, the robot will not move to the target point but will start proceeding to the next target point at a place that is several millimeters away from such a target point. The turning path will deviate from the programmed path. We call the transition area between the two trajectories a turning zone. See the following figure for details:



The size of the turning zone cannot exceed half of the path length. If it is exceeded, the system will automatically reduce the turning zone to half the total path length. The use of turning zones prevents the robot from starting and stopping frequently, significantly reducing the cycle time.

Note:
In some special cases, the turning zone will be canceled. The system will report the log "Corner Path Failed". Possible causes are as follows

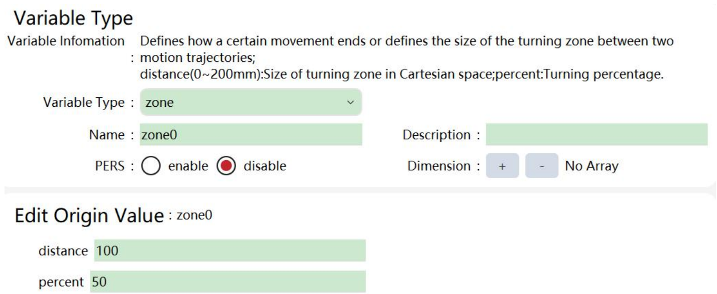
- Turning zone length too small (0.01 mm/0.00001 rad);
- At least one of the two trajectories is too short (1 mm/0.001 rad);
- The two trajectories are nearly parallel and the direction of motion is opposite;
- The two trajectories perform pure rotation with the motion axis reversed. Such that only the end-effector axis rotates forward in the previous trajectory, and only the end-effector axis rotates reverse in the latter trajectory.
- When a warning for "Turning Zone Canceled" is generated, the program automatically treats the affected command target point as a stop-point.
- In addition to the special cases above, all logic commands will cancel the turning zone of the previous motion command.

Definition

Joint space trajectories and Cartesian space trajectories define turning zones with different parameters.
The variable contains two parts: distance and percent.
Distance, size of turning zone in Cartesian space, data type: double; it is used for the commands MoveL, MoveC, and MoveT to define the size of the turning zone for Cartesian space trajectories, that is, when the robot moves to a point with a distance of several millimeters to the target point, it starts to move to the next target point, in millimeters, with the value ranging from 0 to 200 mm.
Percent, turning percentage, data type: double; it is used for MoveJ and MoveAbsJ, indicating how far it is to the target angle when starting turning. 100% represents half the value of the entire rotation angle. For command MoveL with pure space-rotation, the parameter Percent is used instead of Distance.

Example

In the variable list:



A zone variable is defined, in which the size of the Cartesian turning zone is 100 mm and the size of the joint space turning zone is 50%.

Structure assignment

zone z1 = s:{Distance,Percent};
Refer to the "Definition" section of this table for parameter meanings.
Example:zone z1 = s:{1,1};

The system predefines some common turning zone variables, as shown in the following table.

Name	Size of turning zone in Cartesian space	Turning percentage
fine	0 mm	0%
z0	0.3 mm	0.15%
z1	1 mm	1%
z5	5 mm	3%
z10	10 mm	5%
z15	15 mm	8%
z20	20 mm	10%

z30	30 mm	15%
z40	40 mm	20%
z50	50 mm	25%
z60	60 mm	30%
z80	80 mm	40%
z100	100 mm	50%
z150	150 mm	75%
z200	200 mm	100%

15.1.22torqueinfo

Explanation	It is used to describe the forces and torques applied to the robot; It includes joint space torque information and Cartesian space torque information;
Definition	**joint_torque**, data type: joint space torque information; **cart_torque**, data type: Cartesian space torque information; **joint_torque.measure_torque**, data type: double array, information of measured force in the joint space and the torque applied to each axis measured by the force sensor; **joint_torque.external_torque**, data type: double array, information of external force in the joint space, and information of the torque applied to each axis measured by the controller based on the robot model and measured force; **cart_torque.m_force**, data type: double array, force in all directions (xyz) in the Cartesian space; **cart_torque.m_torque**, data type: double array, torque in all directions (xyz) in the Cartesian space;

Example	The following example shows how to use variable torqueinfo:
	Example 1
	TorqueInfo tmp_info = GetEndtoolTorque(tool1, wobj1);
	//Obtain the information architecture of the torque applied to the tool at the end-effector of the robot in the case of tool1 wobj1
	…
	print(tmp_info.joint_torque.measure_torque);
	print(tmp_info.joint_torque.external_torque);
	//Print the measured force and external force of each axis
	…
	print(tmp_info.cart_torque.m_torque);
	//Print Cartesian space torque
	…
	print(tmp_info.cart_torque.m_force[1]);
	print(tmp_info.cart_torque.m_torque[1]);
	//Print information of force and torque in X direction

15.1.23SocketServer

Explanation	A Socket TCP server is established on the controller to listen for connections initiated by external devices as the client. This server is only used to listen for connection requests and multiple connections are supported. When a connection is established, a new SocketConn object is generated for communication. Note: ● Do not create (OpenDev) and destroy (CloseDev) server resources too often as it requires time for system resource application and release. It is recommended to keep at least a 500 ms time interval between creating and destroying resources; otherwise, system resources will be overloaded and cause problems. ● This command only creates a server resource object, and the server creation is not completed. The server needs to enter the listening state via OpenDev and SocketAccept. ● The server supports multiple connections.
Definition	**Ip**, data type: string; the control system uses the ip parameter to match the network interface controller (NIC) and uses the corresponding NIC for network listening. If this parameter is set to "0.0.0.0", it means listening for the connections of all NICs. In most cases, it can be set to "0.0.0.0". **Port**, data type: int, listening port. When an external client initiates a connection, specify the value of the server port set for this purpose. **Name**, data type: string, the unique identifier of the server used in the RL program. It is unique within the project and can be shared between multiple tasks without naming conflicts.

Example	Example 1
	SocketServer ss = {"192.168.0.160", 8090, "svr"};
	//Only listen for NIC with ip set to 192.168.0.160
	SocketConn conn = SocketAccept("svr");
	
	Example 2
	SocketServer ss = {"0.0.0.0", 8090, "svr"}; //Monitor all robot network cards
	SocketConn conn = SocketAccept("svr");

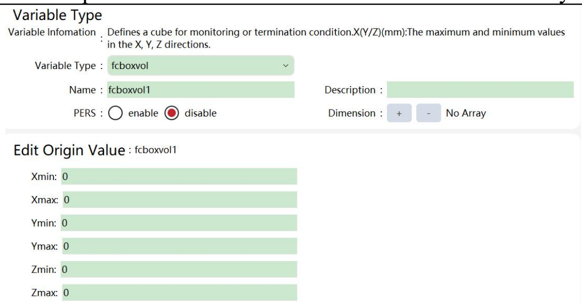
15.1.24SocketConn

Explanation	Socket TCP connection object, used for communication to external devices. There are two types: ● The robot, as a client, initiates a connection and communication through the object to the TCP server of the external device. ● The robot acts as a server for communication connections to the counterpart device generated when a connection is initiated by a TCP client of the external device. When multiple TCP client connections are initiated by different external devices, one connection is generated for each connection.
Definition	Ip, data type: string; when the robot is used as a client, this parameter indicates the ip of the external device's server. When the robot is used as a server, this parameter indicates the ip of the external client when a connection is established by the external device. **Port**, data type: int, listening port. When the robot initiates a connection, the server port of the external device should be specified. **Name**, data type: string, the unique identifier of the connection used in the RL program. It is unique within the project and can be shared between multiple tasks among connections and between connection and server. Server names should not conflict within the project. **Cache**, data type: int, size of the cache, indicating max data received that can be cached; it can be left blank. 1 by default. **Suffix**, data type: string, terminator, indicating the end of a message; it can be left blank. "\r" by default. **Attr**, data type: string, connection attribute. ● "incoming": local server, connected by the opposite-end client. ip and port identify the client information. ● "outgoing": local client, connected to the external server. ip and port identify the opposite-end server connected. ● "" and others: unavailable connection, indicating that the connection has not been opened or unestablished connection has been found. **State**, data type: string, current communication connection status; closed: connection closed; established: connection established and working properly. Note: ● When used as a client, the ip and port information should be set by the user. When used as a server, the ip and port information should be automatically obtained from the accept command. Do not modify these two values easily after the connection is established, unless you are very clear about the use of these two values to avoid errors in program logic and operation. ● suffix can be reset at any time and can take effect until the next read. Use this feature with caution, as it can cause communication data errors. suffix should be set before communication and should not be modified again.

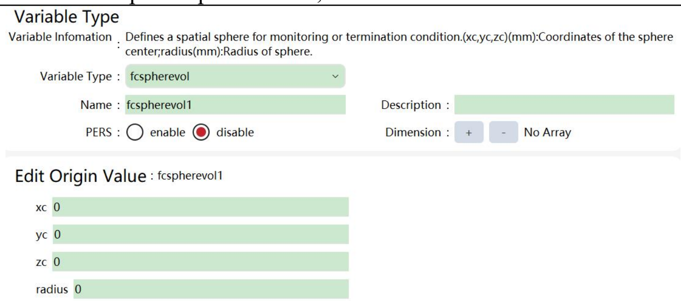
Example	Example 1
	//Server ip "192.168.0.202", port 8090, connection name "clt", cache default to 1, and suffix default to "\r"
	SocketConn scnn1 = {"192.168.0.202", 8090, "clt"};
	
	Example 2
	//Server ip "192.168.0.203", port 8091, connection name "clt1", cache 2, and suffix default to "\r"
	SocketConn scnn2 = {"192.168.0.203", 8091, "clt1", 2};
	Example 3
	//Server ip "192.168.0.204", port 8092, connection name "clt2", cache 2, and suffix "\n"
	SocketConn scnn3 = {"192.168.0.204", 8092, "clt2", 2, "\n"};
	
	Example 4
	//Used as server, connection established by the external device
	//Server ip "192.168.0.204", port 8092, connection name "clt2", cache 2, and suffix "\n"
	SocketConn conn = SocketAccept("svr1");
	Print(conn.ip); //External device IP
	Print(conn.port); //Port of the external device to establish the connection
	Print(conn.cache); //Buffer queue for receiving messages
	Print(conn.suffix); //Sending and receiving suffix

15.1.25FCBoxVol

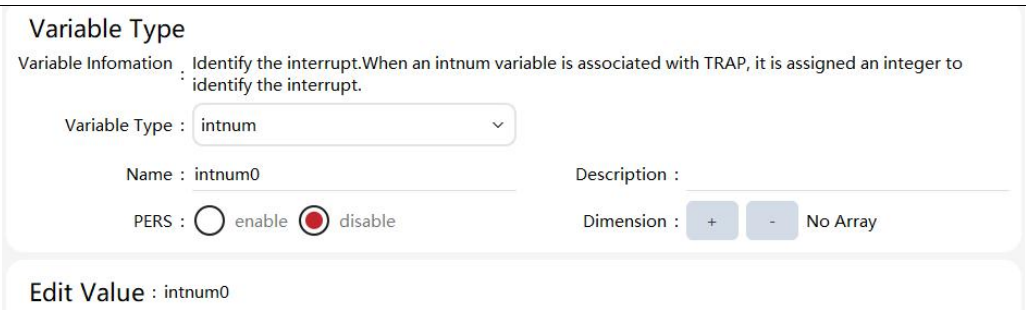
Explanation	It is used to define a spatial cuboid for position monitoring or termination conditions after force control is enabled.
Definition	Xmax represents the coordinate value of the cuboid boundary in the positive x direction; Xmin represents the coordinate value of the cube boundary in the negative x direction; Ymax represents the coordinate value of the cuboid boundary in the positive y direction; Ymin represents the coordinate value of the cube boundary in the negative y direction;

	Zmax represents the coordinate value of the cuboid boundary in the positive z direction; Zmin represents the coordinate value of the cube boundary in the negative z direction;
Example	
Structure assignment	FcBoxVol fcboxvol0 = FCBV:{Xmin,Xmax,Ymin,Ymax,Zmin,Zmax}; Refer to the "Definition" section of this table for parameter meanings. Example:FcBoxVol fcboxvol0 = FCBV:{0,1,2,3,4,5};

15.1.26FCSphereVol

Explanation	It is used to define a spatial sphere for position monitoring or termination conditions after force control is enabled.
Definition	Xc: the coordinate value of the center of the spatial sphere in the x direction; Yc: the coordinate value of the center of the spatial sphere in the y direction; Zc: the coordinate value of the center of the spatial sphere in the Z direction; Radius: the spatial sphere radius;
Example	
Structure assignment	FcSphereVol fc = FCSV:{xc,yc,zc,radius}; Refer to the "Definition" section of this table for parameter meanings. Example:FcSphereVol fc = FCSV:{1,2,3,4};

15.1.27intnum

Explanation	Used as an interrupt identifier. An intnum variable can only identify one interrupt.
Definition	
Example	

15.2Basic variable and structure

All variable types supported by the RL command. The indivisible types, including int, double, bool, and string are basic variables (also known as primary variables), which are the foundation of all variable types. Combined by certain rules, the variable types are called structures.

15.2.1Composition of structure

The combination rules for structures generally combine data with physical significance abstractly.
Example:

- The structure pos combines three doubles into a position (xyz) in three-dimensional space.

- The structure orient combines four doubles into a quaternion that describes the orientation.
- The structure pose combines position (pos) and orientation (orient) into a pose parameter that describes the robot position.

15.2.2Use of structure

Structures, serving as parameters for commands, can be performed in finer ways based on the scenarios. Its data can be modified directly via the specified RL commands.
Example 1:
Robtarget structure consists of: space position (pos), orientation (orient), configuration data (confdata), and external axes (double array). Their names are trans (pos), rot (orient), conf (confdata), extax (double), and users can access the structure members directly in the RL function via their names.
robtarget rob1 = ... // variable list or user-customized Cartesian variable
rob1.trans.x + = 20 // add the x of point position to 20
// In the structure definition of trans (pos), it contains three variables of x, y, and z
// The x of the last visit to rob is therefore rob1.trans.x
print (rob1.trans) // print the position data only

Example 2:
The following is available for the wobj frame:
// Taking default wobj0 as an example
wobj0.robhold // work object handheld (bool).
wobj0.ufprog // user frame programmed (bool, rarely used).
wobj0.ufmec // user frame mechanical unit usually for plating lines and tracking (string).
wobj0.oframe // work object frame pose
wobj0.oframe.x // work object frame pose x
wobj0.oframe.y // work object frame pose y
wobj0.oframe.z // work object frame pose z
wobj0.q1 // work object frame pose quaternion
wobj0.uframe_id// work object-related user frame id

Other complex structures can also refer to this method for structure access.

15.3Function

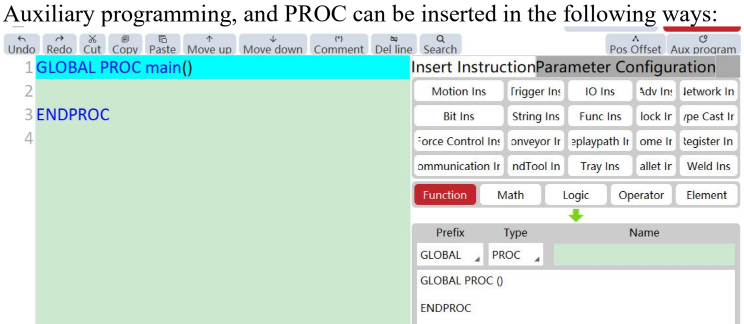
Use of functions can simplify the code structure, improve the readability and reuse rate of code. The user can define the program segment as a new function that needs to be executed frequently so that it can be conveniently called in the main program at any time.

15.3.1Function definition

15.3.1.1PROC

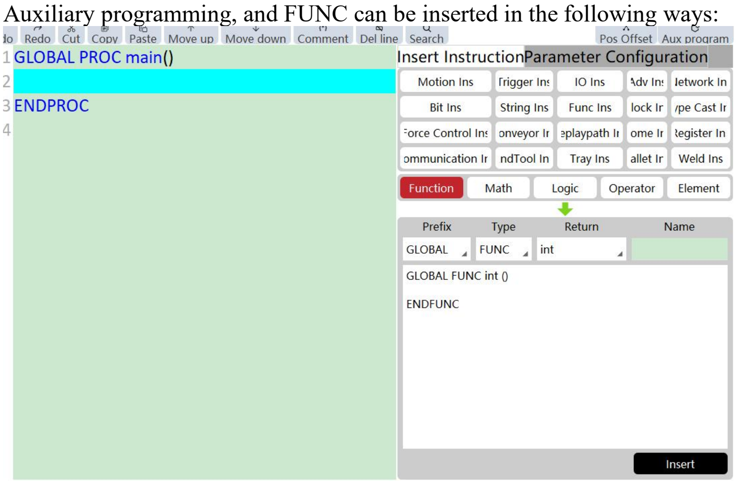
PROC represents a function with no return value, defined as:
SCOPE PROC RoutineName()
…
…
//do something
…
…
ENDPROC
Where:
1. SCOPE is the function scope, which supports both the GLOBAL and LOCAL;
2. PROC is the defining keyword for functions with no return value;
3. RoutineName is the function name. The naming rules are the same as the variable naming rules. For details, see the Variable naming rules.

Auxiliary programming, and PROC can be inserted in the following ways:



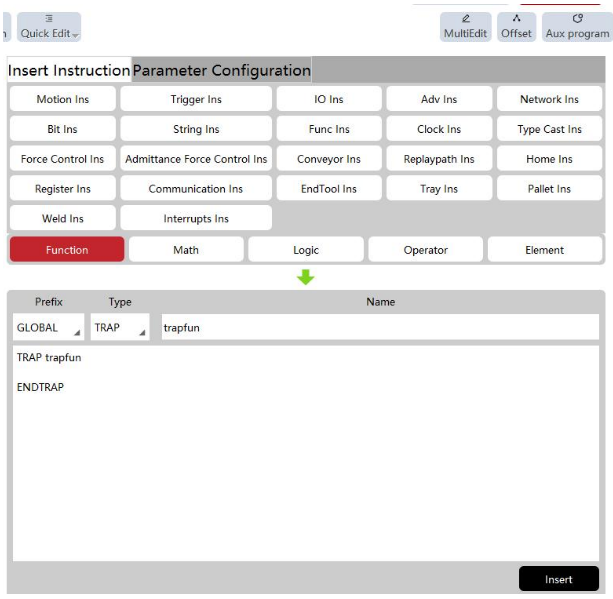
15.3.1.2FUNC

FUNC is a function with a return value, defined as:
SCOPE FUNC RET RoutineName()
…
…
//do something
…
…
ENDFUNC
Where:
1. SCOPE is the function scope, which supports both the GLOBAL and LOCAL;
2. FUNC is the defining keyword for functions with no return value;
3. RET is the return value type;
4. RoutineName is the function name. The naming rules are the same as the variable naming rules. For details, see the Variable naming rules.
Auxiliary programming, and FUNC can be inserted in the following ways:



15.3.1.3TRAP

TRAP is an interrupt function with no return value, defined as:
TRAP TrapName()
…
…
//do something
…
…
ENDTRAP
Where:
1. TRAP is the definition keyword for interrupt functions with no return value;
4. TrapName is the function name. The naming rules are the same as the variable naming rules. For details, see the Variable naming rules.
Auxiliary programming, and TRAP can be inserted in the following ways:



15.3.2Function call

When calling a function, enter the function name directly in the program editor, for example:
RoutineName()
Note:

- Only other GLOBAL-level functions in this project or LOCAL-level functions in this module file can be called. Recursive calls are not supported. Cross calls between two sub-functions is also not supported.
- Calling a function is treated as a separate program command in the compiler.
- It is not allowed to define a function in a function.

15.4Commands

15.4.1Variable type conversion

15.4.1.1ByteToStr

Explanation	It is used to convert byte-type data to string-type data in a specified format.
Definition	Return value, data type: string, the converted string-type data. **ByteToStr (BitData [\Hex] | [\Okt] | [\Bin] | [\Char]);** **BitData**, data type: byte, the byte-type data to be converted; convert by decimal by default. **\Hex**, identifier, converted in hexadecimal. **\Okt**, identifier, converted in octonary. **\Bin**, identifier, converted in binary. **\Char**, identifier, converted according to Ascii character format.

Example	Example 1
	VAR byte data1 = 122
	VAR string str1
	str1 = ByteToStr(data1); //”122”
	str1 = ByteToStr(data1 \Hex); //”7A”
	str1 = ByteToStr(data1 \Okt); //”172”
	str1 = ByteToStr(data1 \Bin); //”01111010”
	str1 = ByteToStr(data1 \Char); //”z”
	Define byte-type variable data1 and assign it with 122, convert data1 to string-type data: 122 by decimal; 7A by hexadecimal;172 by octal; 01111010 by binary; and z by character.

15.4.1.2DecToHex

Explanation	It is used to convert a decimal number to a hexadecimal number.
Definition	Return value, data type: string, the hexadecimal data obtained from the conversion, represented by 0-9, a-f, A-F. **DecToHex(str);** **str**, data type: string, the decimal data to be converted, represented by 0-9.
Attention	Data range from 0 to 2147483647 or 0 to 7fffffff.

15.4.1.3DoubleToByte

Explanation	It is used to convert a double-type variable or a double array to a byte array.

Definition	Return value, data type: byte array, the byte array obtained from the conversion, each double data is converted to 8 byte-type data. **DoubleToByte(dou1);** **dou1**, data type: double, the double-type variable to be converted.

15.4.1.4DoubleToStr

Explanation	It is used to convert a double-type variable to a string.
Definition	**DoubleToStr(Val, Dec);** **Val1**, data type: double, the double-type variable to be converted. **Dec**, data type: string, the number of decimal places to be kept.
Attention	The maximum number of decimal places is 15 digits.

15.4.1.5HexToDec

Explanation	It is used to convert a hexadecimal number to a decimal number.
Definition	Return value, decimal Integer data obtained from the conversion, represented by 0-9. **HexToDec(str);** **str**, data type: string, the hexadecimal data to be converted, represented by 0-9, a-f, A-F.
Attention	Data range from 0 to 2147483647 or 0 to 7fffffff.

15.4.1.6IntToByte

Explanation	It is used to convert an int-type variable or an int array to a byte array.
Definition	Return value, the byte array obtained from the conversion, each int data converted to four byte data. Data type: byte array. **IntToByte(int1);** **int1**, data type: int or int array, the int-type variable or int array to be converted.
Attention	Data range from -2147483647 to 2147483647.

15.4.1.7IntToStr

Explanation	It is used to convert integer to string.
Definition	Return value, the string obtained from the conversion. **IntToStr(int1);** **int1**, data type: int, the integer to be converted.
Attention	Data range from -2147483647 to 2147483647.

15.4.1.8EulerToQuaternion

Explanation	It is used to convert Euler angle to quaternion.
Definition	Return value, the conversion result, 0 means successful, others mean abnormal. **EulerToQuaternion (type,A,B,C,q1,q2,q3,q4);** Type, Euler angle order type, including EULER_XYZ and EULER_ZYX. **A,B,C**, the Euler angle to be converted. Data type: double **q1~q4**, the quaternion obtained from the conversion. Data type: double

15.4.1.9QuaternionToEuler

Explanation	It is used to convert a quaternion to an Euler angle.
Definition	Return value, the conversion result, 0 means successful, others mean abnormal. **QuaternionToEuler (type,q1,q2,q3,q4,A,B,C);** Type, Euler angle order type, including EULER_XYZ and EULER_ZYX. **q1~q4**, the quaternion to be converted. Data type: double **A,B,C**, the Euler angle to be converted. Data type: double

15.4.2Motion commands

15.4.2.1MoveAbsJ

Explanation	MoveAbsJ (Move Absolute Joint) is used to move the robot and the external axis to a position defined by the angle of the axis for rapid positioning or moving the robot to a precise axis angle. All axes move synchronously and the end-effector of the robot moves along an irregular curve. Please be aware of the risk of collision. The tool parameter used in the MoveAbsJ instruction would not affect the end position of the robot, but the tool parameters are still being used by the controller for dynamics calculations.
Definition	**MoveAbsJ (ToJointPos， Speed， Zone， Tool， [Wobj]);**

	The parameter in [ ] is optional and can be omitted. **TojointPos**, To Joint Position, data type: jointtarget, the target angle and position value of the robot and the external axis. **Speed**, Move Speed, data type: speed, to specify the motion speed of the robot when it executes MoveAbsJ, including the translation speed of the robot end-effector, the rotation speed, and the motion speed of the external axis. **Zone**, Turning Zone, data type: zone, to define the size of the turning zone for the current trajectory. **Tool**, data type: tool, the tool used when executing the trajectory. The command MoveAbsJ calculates the motion speed and the size of the turning zone using the tool's TCP data. **[Wobj]**, Work Object, data type: wobj, the work object used when executing this trajectory. When the tool is installed on the robot, this parameter can be ignored; when using external tools, this parameter must be specified, and the robot will calculate the motion speed and the size of the turning zone by using the data saved in wobj.
Example	Example 1 MoveAbsJ (j10, v500, fine, tool1): The robot moves along an irregular path at a velocity of v500 to the absolute joint angle as defined by j10 using tool1, with a turning zone of 0. Example 2 MoveAbsJ (startpoint， v1000， z100， gripper， phone); The robot moves along the irregular path to the absolute joint angle defined by the startpoint at a velocity of v1000 in the work object frame phone by using the gripper, with a turning zone of 100 mm.

15.4.2.2MoveJ

Explanation	MoveJ (Move The Robot By Joint Motion) is used to move the robot from one point to another when the motion trajectory of the robot end-effector is not required. All axes move synchronously and the end-effector of the robot moves along an irregular curve. Please be aware of the risk of collision. The biggest difference between the commands MoveJ and MoveAbsJ is that the given target point format is different. The target point of MoveJ is the spatial pose of the tool (TCP) rather than the joint axis angle.
Example	**MoveJ (ToPoint， Speed， Zone， Tool， [Wobj]);** The parameter in [ ] is optional and can be omitted. **ToPoint,** target pose, data type: robtarget, the target position described in the Cartesian space. **Speed,** Move Speed, data type: speed, to specify the motion speed of the robot when it executes MoveJ, including the translation speed of the robot end-effector, the rotation speed, and the motion speed of the external axis. **Zone**, Turning Zone, data type: zone, to define the size of the turning zone for the current trajectory. Tool, data type: tool, the tool used when executing the trajectory. The command MoveJ calculates the motion speed and the size of the turning zone using the tool's TCP data. [Wobj], Work Object, data type: wobj, the work object is used to execute the trajectory. When the tool is installed on the robot, this parameter can be ignored; when using external tools, this parameter must be specified, and the robot will calculate the motion speed and the size of the turning zone by using the data saved in wobj.
Example	Example 1 MoveJ (p30， v100， z50， tool1); The robot moves the TCP along the irregular path to the target point defined by p30 at a velocity of v100 by using tool1, with a turning zone of 50 mm. Example 2 MoveJ (endpoint， v500， z50， gripper， wobj2); The robot moves the TCP along the irregular path to the target point defined by the endpoint at a velocity of v500 in the work object frame wobj2 by using the gripper, with a turning zone of 50 mm.

15.4.2.3MoveL

Explanation	It is used to move the TCP along a straight line to a given target position. When the starting and ending orientations are different, the orientation will be rotated synchronously with the position to the endpoint. Since the translation and rotation speeds are specified separately, the final motion time of the MoveL command depends on the change time of orientation, position, and elbow (whichever is longer) in order not to exceed the specified speed limit. Therefore, when performing certain trajectories (for example, small displacements but with large changes in orientation), if the robot is moving at a significantly slower or faster speed, check whether the rotation speed setting is reasonable. When you need to keep the TCP stationary by only adjusting the tool orientation, you can achieve this by specifying the starting point and endpoint for MoveL with the same position but with a different orientation.

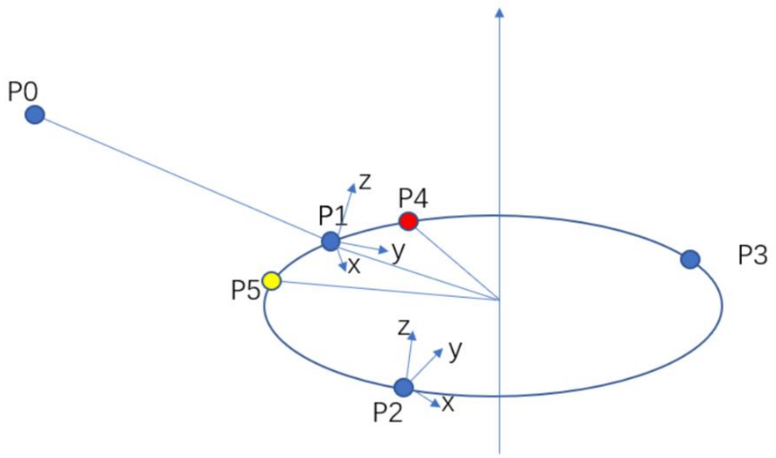
Definition	**MoveL （ToPoint， Speed， Zone， Tool， [Wobj]）;** The parameter in [ ] is optional and can be omitted. **ToPoint,** target pose, data type: robtarget, the target position described in the Cartesian space. **Speed**, Move Speed, data type: speed, to specify the motion speed of the robot when it executes MoveL, including the translation speed of the robot end-effector, the rotation speed, and the motion speed of the external axis. **Zone**, Turning Zone, data type: zone, to define the size of the turning zone for the current trajectory. Tool, data type: tool, the tool used when executing the trajectory. The speed in the command refers to the tool's TCP speed and rotation speed. **[Wobj]**, Work Object, data type: wobj, the work object is used to execute the trajectory. When the tool is installed on the robot, this parameter can be ignored; when using external tools, this parameter must be specified, and the robot will calculate the motion speed and the size of the turning zone by using the data saved in wobj.
Example	Example 1 MoveL (p10， v1000， z50， tool0); The robot moves the TCP along the straight path to the target point defined by p10 at a velocity of v1000 by using tool0, with a turning zone of 50 mm. Example 2 MoveL(endpoint， v500， z50， gripper， wobj2); The robot moves the TCP along the straight path to the target point defined by the endpoint at a velocity of v500 in the work object frame wobj2 by using the gripper, with a turning zone of 50 mm.

15.4.2.4MoveC

Explanation	MoveC (Move Circle) is used to move the TCP along the arc through the middle auxiliary point to the given target position. When the starting and ending orientations are different, the orientation will rotate synchronously with the position to the end position. The orientation at the auxiliary point does not affect the arc motion process. Since the translation and rotation speeds are specified separately, the final motion time of the MoveC command depends on the change time of orientation, position, and elbow (whichever is longer) in order not to exceed the specified speed limit. Therefore, in certain trajectories (for example, small displacements but with large changes in orientation), if the robot is moving at a significantly slower or faster speed, check whether the rotation speed setting is reasonable.
Definition	**MoveC（AuxPoint， ToPoint， Speed， Zone， Tool， [Wobj]）;** The parameter in [ ] is optional and can be omitted. **AuxPoint**, Auxiliary Point, data type: robtarget, the position of the auxpoint described in the Cartesian space, used to determine the size of the arc and the direction of motion. The orientation of this point does not affect the execution of the final trajectory. **ToPoint,** target pose, data type: robtarget, the target position described in the Cartesian space. **Speed**, Move Speed, data type: speed, to specify the motion speed of the robot when it executes MoveC, including the translation speed of the robot end-effector, the rotation speed, and the motion speed of the external axis. **Zone**, Turning Zone, data type: zone, to define the size of the turning zone for the current trajectory. Tool, data type: tool, the tool used when executing the trajectory. The speed in the command refers to the tool's TCP speed and rotation speed. **[Wobj]**, Work Object, data type: wobj, the work object is used to execute the trajectory. When the tool is installed on the robot, this parameter can be ignored; when using external tools, this parameter must be specified, and the robot will calculate the motion speed and the size of the turning zone by using the data saved in wobj.
Example	Example 1 MoveC(p10， p20， v1000， z50， tool0); The robot moves the TCP along the arc, passing through p10 to the target point defined by p20 at a velocity of v1000 by using tool0, with a turning zone of 50 mm. Example 2 MoveC (auxpoint， endpoint， v500， z50， gripper， wobj2); The robot moves the TCP along the arc, passing through the auxpoint to the target point defined by the endpoint at a velocity of v500 in the work object frame wobj2 by using the gripper, with a turning zone of 50 mm.

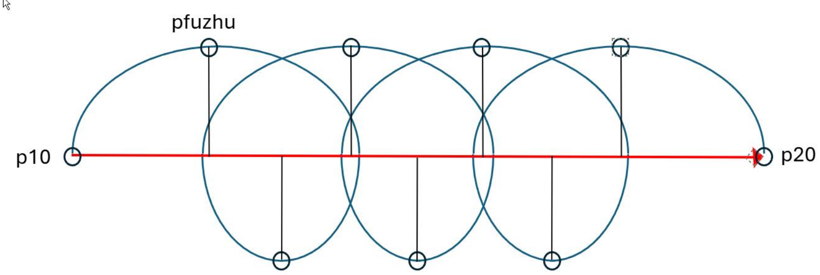
15.4.2.5MoveCF

Explanation	MoveCF (Move Circle Full) is used to move and rotate the Tool Center Point (TCP) along a circular path

	defined by a start point and two auxiliary points at a set full-circle motion angle. The destination of the MoveCF motion is defined by the full-circle motion angle, and the two auxiliary points are only used to locate the space position of circular paths. During the full-circle motion, the orientation will change in the way specified by parameters, and the orientation of auxiliary points has no effect on the orientation during the full-circle motion. The final motion time of MoveCF is only defined by the position variation time. Therefore, the motion velocity is checked for reasonable setting with relatively small radius, thus avoiding excessively fast orientation rotation.
Definition	**MoveCF (AuxPoint1, AuxPoint2, RunDeg, RotType, Speed, Zone, Tool, Wobj);** The parameter in [ ] is optional and can be omitted. **AuxPoint**, Auxiliary Point, data type: robtarget, the position of two auxiliary points described in Cartesian space. It is used to define the position, size, and motion direction of arc, and the orientation of two auxiliary points has no effect on the motion. **RunDeg**, full-circle motion angle, data type: double, value range: -359-3600 (note the soft limit of robot end axis), the angle of full-circle motion, used to define the circle center angle. It can be negative, namely drawing the circle in an opposite direction. **RotType**, orientation variation type, data type: char. Optional parameters (orientation variation types of full-circle motion): "ConstPose", "RotAxis", "FixedAxis", and please choose one of the three according to your needs: ● "ConstPose": full-circle motion of constant orientation, during which the orientation will remain unchanged at the starting point. ● "RotAxis": full-circle motion of moving axis rotation, during which the orientation is determined by the starting point orientation and the circle position, and the orientation changes one circle around the center axis of the circle. Due to the large orientation variation during the motion, the angle of starting end axis should be set reasonably, thus avoiding exceeding the soft limit during motion (the last joint in particular). ● "FixedAxis": full-circle motion of fixed axis rotation, during which the orientation is determined by the starting point orientation and the circle position, and the orientation changes around the center axis of the circle but does not rotate around its own Z-axis. Speed, Move Speed, data type: speed, to define the TCP speed when the robot executes the MoveCF. The orientation rotation speed is related to the full-circle path length, and not defined by speed parameters. Therefore, the parameters should be set reasonably to avoid excessively fast rotation of end axis. **Zone**, Turning Zone, data type: zone, to define the size of the turning zone for the current trajectory. Tool, data type: tool, the tool used when executing the trajectory. The speed in the command refers to the tool's TCP speed. **[Wobj]**, Work Object, data type: wobj, the work object is used to execute the trajectory. When the tool is installed on the robot, this parameter can be ignored; when using external tools, this parameter must be specified, and the robot will calculate the motion speed and the size of the turning zone by using the data saved in wobj.
Example	 Example 1 MoveJ (P1, v1000, z100, tool1); MoveCF (P2, P3, 360,"RotAxis", v100, z50, tool1); The starting point P1, and the auxiliary points P2 and P3 jointly define the space circular trajectory. Taking P1->P2->P3 as the positive direction, move 360° along the circle trajectory and return to the P1. During the motion, based on the orientation of starting point P1, rotate accordingly around the center axis of circular trajectory. Example 2

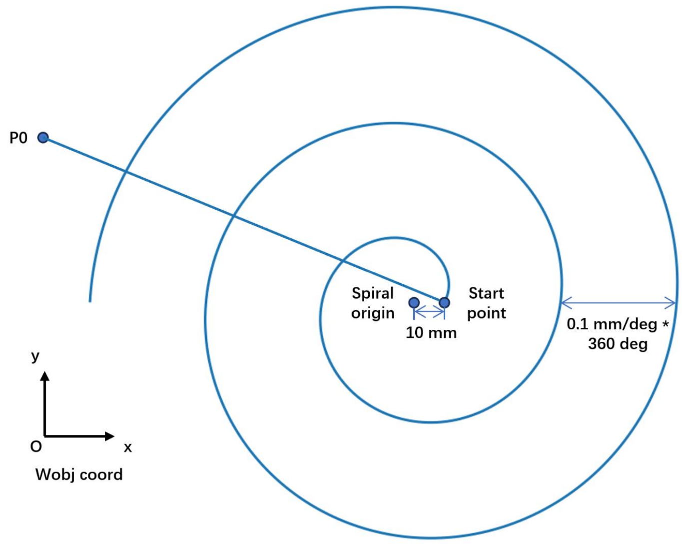
	MoveJ (P1, v1000, z100, tool1); MoveCF (P2, P3, -330,“ConstPose”, v100, z50, tool1); Taking P1->P2->P3 as the positive direction, move -330° along the circle trajectory and return to the P5. Maintain the same orientation as P1 during the motion. Example 3 MoveJ (P1, v1000, z100, tool1); MoveCF (P3, P2, 30,“FixedAxis”, v100, z50, tool1); Taking P1->P3->P2 as the positive direction, move 30° along the circle trajectory to the P4. During the motion, based on the orientation of starting point P1, rotate accordingly around the center axis of the circular trajectory but not around its own Z-axis.

15.4.2.6MoveT

Explanation	MoveT (Move trochoid) is used to move the TCP to a given target position through rotary stepping with a trochoid passing through auxiliary points. When the starting and ending orientations are different, the orientation will rotate synchronously with the position to the end orientation. The orientation at the auxiliary point does not affect the spiral motion process. Since the translation and rotation speeds are specified separately, the final motion time of the MoveT command depends on the change time of orientation, position, and elbow (whichever is longer) in order not to exceed the specified speed limit. Therefore, in certain trajectories (for example, small displacements but with large changes in orientation), if the robot is moving at a significantly slower or faster speed, check whether the rotation speed setting is reasonable.
Definition	**MoveT (AuxPoint, ToPoint, Radius, Step, Speed, Zone, Tool, [Wobj]);** The parameter in [ ] is optional and can be omitted. **AuxPoint**, Auxiliary Point, data type: robtarget, the position of the auxpoint described in the Cartesian space, used to determine the size of the arc and the direction of motion. The orientation of this point does not affect the execution of the final trajectory. **ToPoint,** target pose, data type: robtarget, the target position described in the Cartesian space. **Radius**, cycloid radius, data type: double, radius of trochoid advance, in mm **Step**, step length, data type: double, step length of trochoid advance, in mm. **Speed**, Move Speed, data type: speed, to specify the motion speed of the robot when it executes MoveT, including the translation speed of the robot end-effector, the rotation speed, and the motion speed of the external axis. **Zone**, Turning Zone, data type: zone, to define the size of the turning zone for the current trajectory. Tool, data type: tool, the tool used when executing the trajectory. The speed in the command refers to the tool’s TCP speed and rotation speed. **[Wobj]**, Work Object, data type: wobj, the work object is used to execute the trajectory. When the tool is installed on the robot, this parameter can be ignored; when using external tools, this parameter must be specified, and the robot will calculate the motion speed and the size of the turning zone by using the data saved in wobj.
Example	Example 1 MoveL (p10, v1000,fine, tool0); MoveT (pfuzhu, p20, 150, 50, v1000, z50, tool0); The robot moves TCP to draw a trochoid that passes p10 in an arc at a velocity of v1000. With a trochoid radius of 150 mm and a step of 50 mm, the TCP finally arrives at the target point defined by p20, with a turning zone of 50 mm. 

15.4.2.7MoveSP

Explanation	MoveSP (Move Spiral) is used to draw an Archimedean spiral line on the center point TCP of the tool in a plane parallel to the work object frame xy, according to the specified initial radius, rotation increment, total rotation angle, and rotation direction. During the motion process, the orientation changes linearly to the specified orientation at the target point. Note: When the MoveSP command pauses midway and continues to run, it will regenerate the path starting from the current point and no longer continue with the previous path.

Definition	**MoveSP (Point, Radius, Radius_step, Angle, Direction, Speed, Zone, Tool, [Wobj]);** The parameter in [ ] is optional and can be omitted. Point, target point, data type: robtarget, the orientation of Cartesian point position only used as that of the endpoint of the spiral line. Radius, initial radius, data type: double, the initial radius of spiral line, in mm, required to be no less than 0 mm. The center point position of the spiral line is the radius distance of the current position of TCP moving in the negative direction of the x-axis of the work object frame. **Radius_step**, rotation increment, data type: double, the rotation increment of spiral line, in mm/deg, required to be not less than 0.0001 mm/deg. **Angle**, total rotation angle, data type: double, the total rotation angle of spiral line, in deg, required to be no less than 0.1 deg and no more than 3600 deg. Direction, rotation direction, data type: int, 0: clockwise, 1: counterclockwise. **Speed**, Move Speed, data type: speed, to specify the motion speed of the robot when it executes MoveSP, including the translation speed of the robot end-effector and the rotation speed. **Zone**, Turning Zone, data type: zone, to define the size of the turning zone for the current trajectory. The spiral line does not currently support turning zones, and the system will automatically cancel the turning zones before and after the spiral line. Tool, data type: tool, the tool used when executing the trajectory. The speed in the command refers to the tool's TCP speed and rotation speed. **[Wobj]**, Work Object, data type: wobj, the work object is used to execute the trajectory. When the tool is installed on the robot, this parameter can be ignored; this command only supports external work objects. The TCP motion plane is parallel to the xy plane of the work object frame.
Example	Example 1 MoveL (Start_point, v1000, fine, tool1); MoveSP (Point, 10, 0.1, 900, 1, v100, fine, tool1); The robot uses tool1, and TCP starts moving in a straight line Start_point from P0 at a speed of v1000. TCP performs Archimedean spiral line motion at a speed of v100, with the center point located 10 mm in the negative direction of the x-axis of the work object frame from Start_point, and the rotation direction is counterclockwise when viewed from the z-axis of the work object frame. The radius increases by 0.1 mm for each 1° of rotation and stops after a total rotation of 900°. 

15.4.2.8SearchL

Explanation	SearchL (Search Liner) is used to search the position when moving the TCP along a straight line. During the movement, the robot will monitor a digital input (DI) signal or a read-only register. When the signal status monitored matches the trigger mode, the robot immediately reads the current position. The command can be used when the tool fixed to the manipulator is a probe used for surface detection. Use SearchL command to obtain the outline coordinates of the work object. The command can only be used for motion tasks.
Definition	**SearchL ([action,] [signal_type], [trigger_mode,] save_rob, target_rob, Speed, Tool ,Wobj);** The parameter in [ ] is optional and can be omitted. **Action**, action after triggering DI, data type: keyword, blank: no stop ● \Stop: quick stop, which may cause the robot to deviate from the path, without speed limits. \PStop: planned stop. The robot will stop on the specified path, without speed limits **signal_type**, data type: keyword, blank: DI signal ● \DI: DI signal ● \Reg: register signal

	signal_name, data type: DI signal or register, a signal that the SearchL command triggers a specific behavior, using a user-defined DI signal or a register created in the Communication —> Register **trigger_mode**, DI signal trigger mode, data type: keyword, blank: posedge triggering by default ● \Flanks: edge triggering (posedge/negedge) ● \Posflank: posedge triggering ● \Negflank: negedge triggering ● \Highlevel: high-level triggering ● \Lowlevel: low-level triggering For numeric registers, 0 indicates a low level and non-0 indicates a high level save_rob, data type: robtarget, to save the point position of the position data when the robot triggers the signal **target_rob**, data type: robtarget, target point position of linear motion **Speed**, Move Speed, data type: speed, to specify the motion speed of the robot when it executes Search, including the translation speed of the robot end-effector, the rotation speed, and the motion speed of the external axis. Tool, data type: tool, the tool used when executing the trajectory. The speed in the command refers to the tool's TCP speed and rotation speed. **Wobj**, Work Object, data type: wobj, the work object is used to execute the trajectory. When the tool is installed on the robot, this parameter can be ignored; when using external tools, this parameter must be specified, and the robot will calculate the motion speed and the size of the turning zone by using the data saved in wobj.
Example	Example 1 SearchL(di0, save_rob, target_rob, v500, tool0); The robot uses tool0 and TCP moves at a speed of v500 towards target_rob in a straight line. If di0 jumps to high during the motion, the robot's coordinate information at the time of the signal jump is recorded in save_rob. Example 2 SearchL(\PStop, di0, \Lowlevel, save_rob, target_rob, v500, tool0); The robot uses tool0 and TCP moves at a speed of v500 towards target_rob in a straight line. If di0 jumps to low during the motion, the robot immediately has a planned stop and the robot's coordinate information at the time of detecting to be low is recorded in save_rob. Example 3 SearchL(\PStop, \Reg, register0, save_rob, target_rob, v500, tool0, wobj1); The robot moves the TCP towards the target_rob in a straight line at a velocity of v500 in wobj1 by using the tool0. If register0 changes from 0 to non-0 during the motion, the robot immediately has a planned stop and the robot's frame information is recorded in save_rob.

15.4.2.9SearchC

Explanation	SearchC (Search Circle) is used to search for a position when moving the TCP along a circle. During the movement, the robot will monitor a digital input (DI) signal or a read-only register. When the signal status monitored matches the trigger mode, the robot immediately reads the current position. The command can be used when the tool fixed to the manipulator is a probe used for surface detection. Use SearchC command to obtain the outline coordinates of the work object. The command can only be used for motion tasks.
Definition	**SearchC ([action,] di, [trigger_mode,] save_rob, aux_rob, target_rob, Speed, Tool [,Wobj]);** The parameter in [ ] is optional and can be omitted. **Action**, action after triggering DI, data type: keyword, blank: no stop ● \Stop: quick stop, which may cause the robot to deviate from the path. But the robot stops quickly. Only available when the speed is below v100 ● \PStop: planned stop. The robot will stop on the specified path, without speed limits **signal_type**, data type: keyword, blank: DI signal ● \DI: DI signal ● \Reg: register signal **signal_name**, data type: DI signal or register, a signal that the SearchC command triggers a specific behavior, using a user-defined DI signal or a register created in the Communication —> Register. **trigger_mode**, DI signal trigger mode, data type: keyword, blank: posedge triggering by default ● \Flanks: edge triggering (posedge/negedge) ● \Posflank: posedge triggering ● \Negflank: negedge triggering ● \Highlevel: high-level triggering ● \Lowlevel: low-level triggering For numeric registers, 0 indicates a low level and non-0 indicates a high level save_rob, data type: robtarget, to save the point position of the position data when the robot triggers the signal

	aux_rob, data type: robtarget, auxiliary point of circular motion **target_rob**, data type: robtarget, target point position of circular motion **Speed**, Move Speed, data type: speed, to specify the motion speed of the robot when it executes Search, including the translation speed of the robot end-effector, the rotation speed, and the motion speed of the external axis. Tool, data type: tool, the tool used when executing the trajectory. The speed in the command refers to the tool's TCP speed and rotation speed. **[Wobj]**, Work Object, data type: wobj, the work object is used to execute the trajectory. When the tool is installed on the robot, this parameter can be ignored; when using external tools, this parameter must be specified, and the robot will calculate the motion speed and the size of the turning zone by using the data saved in wobj.
Example	Example 1 SearchC (di0, save_rob, aux_rob, target_rob, v500, tool0); The robot uses tool0 and TCP moves at a speed of v500 towards target_rob in a circle after passing auxiliary point aux_rob. If di0 jumps to high during the motion, the robot's coordinate information at the time of the signal jump is recorded in save_rob. Example 2 SearchC (\PStop, di0, \Flanks, save_rob, target_rob, v500, tool0); The robot uses tool0 and TCP moves at a speed of v500 towards target_rob in a straight line after passing auxiliary point aux_rob. If di0 jumps from low to high or from high to low during the motion, the robot immediately has a planned stop and the robot's coordinate information at the time of the signal jump is recorded in save_rob. Example 3 SearchC (\Reg, register0, \Flanks, save_rob, target_rob, v5, tool0); The robot moves TCP towards the target_rob in a straight line through passing aux_rob at a velocity of v5. If register0 changes from 0 to non-0 during the motion, the robot immediately has a planned stop and the robot's frame information at the time of the signal jump is recorded in save_rob.

15.4.3Trigger command

15.4.3.1TrigIO

Explanation	TriggIO is used to set a trigdata as an output I/O trigger during the motion. Digital output DO and digital group output GO are supported.

Definition	**TrigIO (TrigData,Distance,RefStart,SignalName,Value);**
	TrigData, data type: trigdata, a variable used to store the trigger data set by this TrigIO;
	Distance, data type: double, non-negative (negative numbers are treated as 0), to define the location offset of the trigger event on the path. Whether the location offset is relative to the path start or end is defined by RefStart;
	RefStart, data type: bool, to define whether the trigger position is relative to the start point (true) or the end point (false);
	SignalName, data type: signaldo or signalgo, the signal name of the digital output or digital group output associated with this defined IO event, which must be an output signal that has been set correctly; //add
	Value, data type: bool or int, to define the target value of the output signal when an IO event is triggered. The data type of the given value should match the SignalName type.

Example	Refer to the TrigL example

15.4.3.2TrigReg

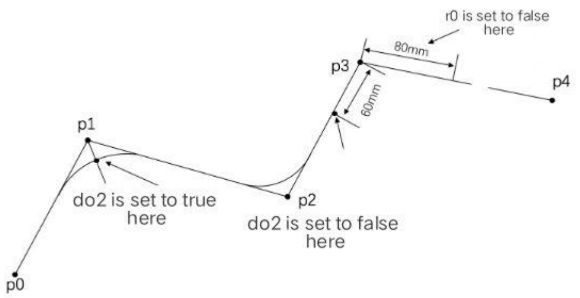
Explanation	TrigReg is used to set a trigdata to modify the register value during the motion; register types supported include int16, bool, float, and bit.
Definition	**TrigReg (TrigData,Distance,RefStart,RegName,Value);** **TrigData**, data type: trigdata, a variable used to store the trigger data set by this TrigIReg; **Distance**, data type: double, non-negative (negative numbers are treated as 0), to define the location offset of the trigger event on the path. Whether the location offset is relative to the path start or end is defined by RefStart; **RefStart**, data type: bool, to define whether the trigger position is relative to the start point (true) or the end point (false); **RegName**, the register name, and the data type is not available. Note: Registers can not be created in RL. The user needs to create new registers through "Robot -> Communication -> Register"; **Value**, data type: int16, bool, float, or bit, to define the target value of the register when a register modification event is triggered. The data type of the given value should match the RegName type; if the value specified by the user mismatches with the register type, the type will be transformed automatically;
Example	Refer to the TrigL example

15.4.3.3TrigVar

Explanation	TrigVar is used to set a trigdata to modify the RL variables during the motion, and the type of RL variables includes bool, byte, int, and double.
Definition	**TrigVar (TrigData,Distance,RefStart,VarName,Value);** **TrigData**, data type: trigdata, a variable used to store the trigger data set by this TrigVar; **Distance**, data type: double, non-negative (negative numbers are treated as 0), to define the location offset of the trigger event on the path. Whether the location offset is relative to the path start or end is defined by RefStart; **RefStart**, data type: bool, to define whether the trigger position is relative to the start point (true) or the end point (false); **VarName**, name of the RL variable to be modified, without any data type; **Value**, data type: bool, byte, int, and double, used to define the target value of the RL variable when a RL variable modification event is triggered. The data type of the given value should match the VarName type, and if the value specified by the user mismatches with the RL variable type, the type will be transformed automatically.
Example	Refer to the TrigL example

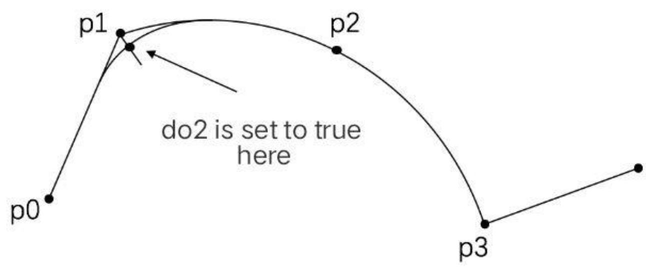
15.4.3.4TrigL

Explanation	Like MoveL, TrigL is a command to perform linear motion in space. The difference is that TrigL can perform predefined operations at several specified positions during the motion; the two commands are the same in the number and meaning of other parameters.
Definition	**TrigL (ToPoint,Speed,Trigger,Zone,Tool,[Wobj]);** **ToPoint**, target pose, data type: robtarget, the target pose described in the Cartesian space; **Speed**, type: speed, to specify the motion speed of the robot when it executes MoveL, including the translation speed of the robot end-effector, the rotation speed, and the motion speed of the external axis; **Trigger**, trigger condition and action, data type: trigdata; trigdata must be the trigdata processed with TrigX command; otherwise, the compiler will report an error when coming to this line. **Zone**, turning zone, data type: zone, to define the size of turning zone for the current trajectory; **Tool**, type: tool; **[Wobj]**, work object, type: wobj, the work object used when executing this trajectory. When the tool is installed on the robot, this parameter can be ignored. When using external tools, this parameter must be specified, and the robot will calculate the motion speed and the size of turning zone by using the data stored in wobj.

	Example 1
	VAR trigdata tc1
	VAR trigdata tc2
	VAR trigdata tc3
	...
	//Set tc1, tc2, tc3
	TrigIO (tc1,0,true,do2,true);
	TrigIO (tc2,60,false,do2,false);
	TrigVar (tc2,60,false,Bool0,false); //Bool0 is a RL variable in bool type
	TrigReg (tc3, 80, true, r0, false); //r0 is a bool type register
	...
	//Motion
	MoveL (p1,v500,z50,tool1);
	TrigL (p2,v500,tc1,z50,tool1);
	TrigL (p3,v500,tc2,fine,tool1);
	TrigL (p4,v500,tc3,fine,tool1);
	

15.4.3.5TrigC

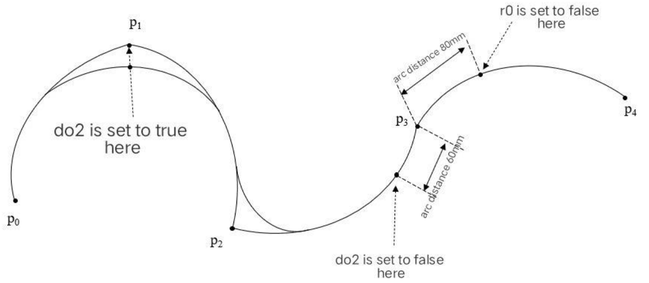
Explanation	TriggC is similar to MoveC in that it is a command to execute circular motion. The difference is that TriggC can perform predefined operations at several specified positions during the motion; the two commands are the same in the number and meaning of other parameters.
Definition	**TrigC (AuxPoint,ToPoint,Speed,Trigger,Zone,Tool,[Wobj]);** **AuxPoint**, Auxiliary Point, data type: robtarget, the target pose described in the Cartesian space; **ToPoint**, target pose, data type: robtarget, the target pose described in the Cartesian space; **Speed**, type: speed, to specify the motion speed of the robot when it executes MoveL, including the translation speed of the robot end-effector, the rotation speed, and the motion speed of the external axis; **Trigger**, trigger condition and action, data type: trigdata; trigdata must be the trigdata processed with TrigX command; otherwise, the compiler will report an error when coming to this line. **Zone**, turning zone, data type: zone, to define the size of turning zone for the current trajectory; **Tool**, type: tool; **[Wobj]**, work object, type: wobj, the work object used when executing this trajectory. When the tool is installed on the robot, this parameter can be ignored. When using external tools, this parameter must be specified, and the robot will calculate the motion speed and the size of turning zone by using the data stored in wobj.

Example	Example 1
	VAR trigdata tc1
	...
	//Set tc1
	TrigIO (tc1,0,true,do2,true);
	...
	//Motion
	MoveL (p1,v500,z50,tool1);
	TrigC (p2,p3,v500,tc1,fine,tool1);
	

15.4.3.6TrigJ

Explanation	The forms of TrigJ and MoveJ are exactly identical, and both are commands that execute joint space motion. The difference is that TrigJ can execute predefined operations at several specified positions during the motion, with no difference in the number and meaning of other parameters. When the robot executes a MoveJ command, the trajectory of tcp is usually an arc. When triggering the Trigger signal, the distance of TrigJ is calculated as per the arc traveled by the tcp, see the example for details.
Definition	**TrigJ (ToPoint,Speed,Trigger,Zone,Tool,[Wobj]);** **ToPoint**, target pose, data type: robtarget, the target pose described in the Cartesian space; **Speed**, data type: speed, to specify the motion speed of the robot when it executes MoveL, including the robot joint speed, translation speed of the robot end-effector, rotation speed, and motion speed of the external axis; **Trigger**, trigger condition and action, data type: trigdata; trigdata must be the trigdata processed with TrigX command; otherwise, the compiler will report an error when coming to this line; **Zone**, turning zone, data type: zone, to define the size of turning zone for the current trajectory; **Tool**, type: tool; **[Wobj]**, work object, type: wobj, the work object used when executing this trajectory. When the tool is installed on the robot, this parameter can be ignored. When using external tools, this parameter must be specified, and the robot will calculate the motion speed and the size of turning zone by using the data stored in wobj.
Example	Example 1 VAR trigdata tc1; VAR trigdata tc2; VAR trigdata tc3; ... //Set tc1, tc2, tc3 TrigIO (tc1,0,true,do2,true);

	TrigIO (tc2,60,false,do2,false);
	TrigReg (tc3, 80, true, r0, false); //r0 is a bool type register
	...
	//Motion
	MoveJ (p1,v500,z50,tool1);
	TrigJ (p2,v500,tc1,z50,tool1);
	TrigJ (p3,v500,tc2,fine,tool1);
	TrigJ (p4,v500,tc3,fine,tool1);



15.4.4Force control commands

15.4.4.1CalibSensorError

Explanation	Calibrate the torque sensor zero
Definition	No parameters, and can be used directly.
Example	Example 1 FcInit (Tool1, Wobj0, 0); FcStart(); CalibSensorError();
Attention	● The torque sensor zero calibration is only applicable to collaborative robots, which is unavailable for industrial robots. ● This interface can be called for calibration when the manipulator is in a static state at any pose. However, a correct load must be set, otherwise an error may be reported.

15.4.4.2FcInit

Explanation	It is used for initialization before the force control is enabled, such as setting the work object, tool, and force control frame.
Definition	**FcInit (Tool, Wobj, ForceFrameRef);** **Tool**, data type: pose, the tool used for force control. The origin of the force control frame is the TCP of the tool (the orientation is the same as the orientation of the frame selected in the third parameter). Note that all adapter flanges used need to be included in the definition of the tool. **Wobj**, data type: pose, the work objects used for force control. Many force control functions are defined relative to the work object frame, such as the orientation of the force control frame, the search mode, and termination conditions. This parameter is Wobj0 by default. **ForceFrameRef**, data type: int, to define the frame to which the force control frame is relative. It supports: ● 0: world frame ● 1: work object frame ● 2: Tool frame ● 3: Base frame The default value is the world frame (0).
Example	Example 1 FcInit (Tool1, Wobj0, 0); Initialize force control, and define the tool1 and work object wobj0 used when force control is enabled, and the definition of force control frame in relative to the world frame.
Attention	FcInit is not allowed to be called again between FcInit and FcStop.

15.4.4.3SetControlType

Explanation	It is used to set the impedance control type.
Definition	**SetControlType (ctrl_type);** **ctrl_type**, data type: int, impedance control type. It supports: ● 0: joint impedance ● 1: Cartesian impedance

Example	Example 1 FcInit (Tool1, Wobj0, 0); SetControlType(0); Set joint impedance as the impedance control mode after executing FcInit.
Attention	The impedance type can only be set after executing FcInit and before executing FcStart. After setting the impedance control type, it should be operated with proper impedance stiffness. If the setting control type is joint impedance, users should proceed to use the SetJntCtrlStiffVec interface to set the joint impedance stiffness. If only Cartesian impedance stiffness is set, the Cartesian stiffness setting does not take effect when the robot moves. As it is both joint impedance and Cartesian impedance stiffness, only the joint impedance stiffness setting takes effect when the robot moves, with both joint impedance and Cartesian impedance stiffness set to 0 by default.

15.4.4.4SetCartNsStiff

Explanation	It is used to set the null-space impedance stiffness
Definition	**SetCartNsStiff(cart_ns_stiff);** **cart_ns_stiff**, data type: double, Cartesian null-space impedance stiffness, range: 0–4, in N.m/rad.
Example	Example 1 FcInit (Tool1, Wobj0, 0); SetControlType(1); SetCartNSStiff(2); Set Cartesian impedance as the impedance control mode and the null-space impedance stiffness as 2.
Attention	This interface can only be called after executing SetControlType 1, that is, setting Cartesian impedance as the impedance control mode. If not, the null-space impedance parameters will not be set successfully.

15.4.4.5SetJntCtrlStiffVec

Explanation	It is used to set the joint impedance stiffness
Definition	**SetJntCtrlStiffVec(jnt1_stiff，jnt2_stiff，jnt3_stiff，jnt4_stiff，jnt5_stiff， jnt6_stiff，jnt7_stiff);** **Jnt1_stiff**, data type: double, impedance stiffness of joint 1, in N.m/rad. **Jnt2_stiff**, data type: double, impedance stiffness of joint 2, in N.m/rad. **Jnt3_stiff**, data type: double, impedance stiffness of joint 3, in N.m/rad. **Jnt4_stiff**, data type: double, impedance stiffness of joint 4, in N.m/rad. **Jnt5_stiff**, data type: double, impedance stiffness of joint 5, in N.m/rad. **Jnt6_stiff**, data type: double, impedance stiffness of joint 6, in N.m/rad. **Jnt7_stiff**, data type: double, impedance stiffness of joint 7, in N.m/rad. //When setting up a 6-axis robot, this parameter defaults to 0.
Example	Example 1 FcInit(Tool1, Wobj0, 0); SetControlType(0); SetJntCtrlStiffVec(1500,1500, 1500,1500,100,100,100); Set the joint impedance as the impedance control mode and the impedance stiffness of joints 1–7 as 1500, 1500, 1500, 1500, 100, 100, 100, respectively.
Attention	This interface can only be called after executing SetControlType 0, that is, setting joint impedance as the impedance control mode. If not, the joint impedance parameters will not be set successfully.

Maximum stiffness of each axis of the collaborative model (N.m/rad)

	J1	J2	J3	J4	J5	J6	J7
ER3P	6000	6000	6000	1000	1000	1000	1000
ER7P	6000	6000	6000	1000	1000	1000	1000
ER3	3000	3000	3000	300	300	300	
ER7	3000	3000	3000	300	300	300	
SR3	3000	3000	3000	300	300	300	
SR4	3000	3000	3000	300	300	300	
SR5	3000	3000	3000	300	300	300	
CR7	6000	6000	6000	1000	1000	1000	
CR12	20000	20000	20000	3000	2500	2500	
CR17	20000	20000	20000	2500	2500		
CR17/25	20000	20000	20000	2500	2500		
CR18	20000	20000	20000	3000	2500	2500	
CR20	20000	20000	20000	3000	2500	2500	
CR25	20000	20000	20000	2500	2500		

15.4.4.6SetCartCtrlStiffVec

Explanation	It is used to set the Cartesian impedance stiffness
Definition	**SetCartCtrlStiffVec trans（stiff_x，trans_stiff_y，trans_stiff_z，rot_stiff_x，rot_stiff_y，rot_stiff_z);** **trans_stiff_x**, data type: double, Cartesian impedance force stiffness in the X-direction, in N/m. **trans_stiff_y**, data type: double, Cartesian impedance force stiffness in the Y-direction, in N/m.

	trans_stiff_z, data type: double, Cartesian impedance force stiffness in the Z-direction, in N/m. **rot_stiff_x**, data type: double, Cartesian impedance torque stiffness in the X-direction, in N.m/rad. **rot_stiff_y**, data type: double, Cartesian impedance torque stiffness in the Y-direction, in N.m/rad. **rot_stiff_z**, data type: double, Cartesian impedance torque stiffness in the Z-direction, in N.m/rad.
Example	Example 1 FcInit (Tool1, Wobj0, 0); SetControlType (1); SetCartCtrlStiffVec(1000， 1000， 1000， 100， 100， 100); Set Cartesian impedance as the impedance control mode and the impedance force stiffness in X/Y/Z direction as 1000, and the impedance torque stiffness as 100.
Attention	This interface can only be called after executing SetControlType 1, that is, setting Cartesian impedance as the impedance control mode. If not, the Cartesian impedance parameters will not be set successfully.

Maximum stiffness of each axis of the collaborative model, in N/m and N.m/rad

	trans_x	trans_y	Trans_z	rot_x	rot_y	rot_z
ER3P	6000	6000	6000	1000	1000	1000
ER7P	6000	6000	6000	1000	1000	1000
ER3	3000	3000	3000	300	300	300
ER7	3000	3000	3000	300	300	300
SR3	3000	3000	3000	300	300	300
SR4	3000	3000	3000	300	300	300
SR5	3000	3000	3000	300	300	300
CR7	6000	6000	6000	1000	1000	1000
CR12	18000	18000	18000	2500	2500	2500
CR17/25	18000	18000	18000	2500	2500	2500
CR18	18000	18000	18000	2500	2500	2500
CR20	18000	18000	18000	2500	2500	2500

15.4.4.7SetJntTrqDes

Explanation	Set the desired torque of the joint
Definition	**SetJntTrqDes (tau_d1,tau_d2,tau_d3,tau_d4,tau_d5,tau_d6,tau_d7);** **tau_d1**, data type: double, the desired torque of joint 1, range: -30–30, in N.m. **tau_d2**, data type: double, the desired torque of joint 2, range: -30–30, in N.m. **tau_d3**, data type: double, the desired torque of joint 3, range: -30–30, in N.m. **tau_d4**, data type: double, the desired torque of joint 4, range: -30–30, in N.m. **tau_d5**, data type: double, the desired torque of joint 5, range: -30–30, in N.m. **tau_d6**, data type: double, the desired torque of joint 6, range: -30–30, in N.m. **tau_d7**, data type: double, the desired torque of joint 7, range: -30–30, in N.m. //When setting up a 6-axis robot, this parameter defaults to 0.
Example	Example 1 FcInit (Tool1, Wobj0, 0); SetControlType (0); FcStart(); SetJntTrqDes (5,5,5,5,5,5,5); FcStop(); Set the desired torque of all joints to 5 N.m.
Attention	This interface can only be called after executing FcStart and before executing FcStop. If not, the desired joint torque will not be set successfully.

15.4.4.8SetCartForceDes

Explanation	Set the desired Cartesian force/torque
Definition	**SetCartForceDes (force_x，force_y，force_z，torque_x，torque_y，torque_z);** **force_x**, data type: double, the desired Cartesian force in the X-direction, range: -60–60, in N. **force_y**, data type: double, the desired Cartesian force in the Y-direction, range: -60–60, in N. **force_z**, data type: double, the desired Cartesian force in the Z-direction, range: -60–60, in N. **torque_x**, data type: double, the desired Cartesian torque in the X-direction, range: -10–10, in N.m. **torque_y**, data type: double, the desired Cartesian torque in the Y-direction, range: -10–10, in N.m. **torque_z**, data type: double, the desired Cartesian torque in the Z-direction, range: -10–10, in N.m.
Example	Example 1 FcInit (Tool1, Wobj0, 0); SetControlType(1); FcStart(); SetCartForceDes(0,0,5,0,0,0); FcStop(); Set the desired Cartesian force/torque. Set the desired force in the z-direction to 5 N.

Attention	This interface can only be called after executing FcStart and before executing FcStop. If not, the desired Cartesian force/torque will not be set successfully.

15.4.4.9SetSineOverlay

Explanation	Set the sine overlay rotating around a single axis
Definition	**SetSineOverlay(line_dir, amplify, frequncy, phase, bias);** **line_dir**, data type: int, overlay reference axis. It supports: ● 0: x-axis as the reference direction ● 1: y-axis as the reference direction ● 2: z-axis as the reference direction **Amplify**, data type: double, overlay amplitude, in N.m. **Frequncy**, data type: double, overlay frequency, in Hz. **Phase**, data type: double, overlay phase, range: −3.14 to 3.14, in rad. **Bias**, data type: double, overlay offset, range: −10 to 10**,** in N.m.
Example	Example 1 FcInit(Tool1, Wobj0, 0); SetControlType(1); SetSineOverlay(0，10，5，3.14，2); Set rotary overlay around x-axis (0), amplitude: 10 N.m, frequency: 5 Hz, phase: 3.14 rad, and offset: 2 N.m.
Attention	This interface can only be called after executing SetControlType 1, that is, setting Cartesian impedance as the impedance control mode, and before executing StartOverlay. If not, the sine overlay will not be set successfully.

Upper limit of collaborative model parameters:

	Maximum overlay amplitude	Maximum overlay frequency
ER3P	10	5
ER7P	10	5
ER3	10	5
ER7	10	5
SR3	5	5
SR3-C	5	5
SR3-A	5	5
SR4	5	5
SR4-C	5	5
CR7	10	5
CR12	10	5
CR17/25	10	5
CR18	10	5
CR20	10	5

15.4.4.10SetLissajousOverlay

Explanation	Set the Lissajous overlay within a plane
Definition	**SetLissajousOverlay(plane, amplify_one, frequncy_one, amplify_two, frequncy_two, phase_diff);** **Plane**, data type: int, overlay reference plane. It supports: ● 0: XY plane as the reference plane ● 1: XZ plane as the reference plane ● 2: YZ plane as the reference plane **amplify_one**, data type: double, amplitude of overlay in Direction 1, range: −20 to 20, in N.m. **frequncy_one**, data type: double, frequency of overlay in Direction 1, range: 0−5, in Hz. **amplify_two**, data type: double, amplitude of overlay in Direction 2, range: −20 to 20, in N.m. **frequncy_two**, data type: double, frequency of overlay in Direction 2, range: 0−5, in Hz. **phase_diff**, data type: double, phase deviation between overlays in two directions, range: −3.14 to 3.14, in rad.
Example	Example 1 FcInit (Tool1，Wobj0，0); SetControlType (1); SetLissajousOverlay （0，5，2.5，10，5，3.14); Set Lissajous overlay within the xy plane (0). The amplitude and frequency are 5 N.m and 2.5 Hz in the x-direction, and 10 N.m and 5 Hz in the y-direction. The phase deviation between the y-direction and x-direction is 3.14 rad.
Attention	This interface can only be called after executing SetControlType 1, that is, setting Cartesian impedance as the impedance control mode, and before executing StartOverlay. If not, the overlay will not be set

	successfully.

15.4.4.11SetLoad

Explanation	Set the load information used by the force control module.
Definition	**SetLoad(m,rx,ry,rz,Ixx,Iyy,Izz);** **M**, data type: double, load mass, in kg, range: 0–25; **Rx**, data type: double, the position of the load's center of mass on the x-axis of the flange frame, in mm, range: (-300, 300); **Ry**, data type: double, the position of the load's center of mass on the y-axis of the flange frame, in mm, range: (-300, 300); **Rz**, data type: double, the position of the load's center of mass on the z-axis of the flange frame, in mm, range: (-300, 300); **Ixx**, data type: double, the inertia of the load's center of mass along the x-axis, in kg*mm^2, range: (0, 100000); **Iyy**, data type: double, the inertia of the load's center of mass along the y-axis, in kg*mm^2, range: (0, 100000); **Izz**, data type: double, the inertia of the load's center of mass along the z-axis, in kg*mm^2, range: (0, 100000);
Example	Example 1 FcInit (Tool1, Wobj0, 0); FcStart(); SetLoad (1,0,0,10,0.001,0.001,0.0001); Set the end-effector load as follows: the mass is 1 kg, the component of the center of mass in the flange frame is 0, 0, and 10 mm, and the inertia of the load relative to the load's center of mass frame is 0.001 kg*mm^2, 0.001 kg*mm^2, and 0.0001 kg*mm^2, respectively.
Attention	The interface can only be called after executing FcStart. If not, the load parameters will not be set successfully.

15.4.4.12FcStart

Explanation	It is used to enable force control. It switches the robot from pure position control to force control
Definition	No parameters, and can be used directly.
Example	Example 1 FcInit (Tool1, Wobj0, 0); FcStart(); Enable force control through FcStart after executing FcInit. The robot is now in force control mode.
Attention	This interface is called after executing FcInit. Before calling the instruction, the robot mechanical zero, force sensor zero, and load information should be set correctly, and the body parameters are identified correctly. Otherwise, the effectiveness of the force control function will be affected or even disabled.

15.4.4.13FcStop

Explanation	It is used to stop force control. The robot will switch from force control to position control. Executing this command will automatically stop all overlays internally.
Definition	No parameters, and can be used directly.
Example	Example 1 FcInit (Tool1, Wobj0, 0); FcStart(); FcStop(); It is used to stop force control. The robot will switch from force control to position control. Executing this instruction clears all force control states.
Attention	This interface is called after executing FcStart, and it will clear the force control state, such as force control load information, impedance parameters, overlay, and desired force. To enable force control again, FcInit should be executed again.

15.4.4.14StartOverlay

Explanation	It is used to enable the overlay set before
Definition	No parameters, and can be used directly.
Example	Example 1 FcInit (Tool1, Wobj0, 0); SetControlType (1); SetSineOverlay (0，10，5，3.14，2); SetLissajousOverlay (0，5，2.5，10，5，3.14); FcStart(); StartOverlay();

	Start the superposition of overlays set before. In the example, these overlays include the sine overlay around the x-axis and the Lissajous overlay within xy plane.
Attention	The interface can only be called after executing FcStart. If not, the sine overlay will not be set successfully.

15.4.4.15PauseOverlay

Explanation	Pause the overlay
Definition	No parameters, and can be used directly
Example	Example 1 FcInit (Tool1, Wobj0, 0); SetControlType (1); SetSineOverlay (0， 10， 5， 3.14， 2); FcStart(); StartOverlay(); PauseOverlay(); Pause the overlay
Attention	The interface can only be called after executing StartOverlay.

15.4.4.16RestartOverlay

Explanation	Restart the paused overlays
Definition	No parameters, and can be used directly.
Example	Example 1 FcInit (Tool1, Wobj0, 0); SetControlType(1); SetSineOverlay(0， 10， 5， 3.14， 2); FcStart(); StartOverlay(); PauseOverlay(); RestartOverlay(); Restart the overlays
Attention	The interface can only be called after executing PauseOverlay. This interface is used in conjunction with PauseOverlay to restart paused overlays.

15.4.4.17StopOverlay

Explanation	Stop the overlays
Definition	No parameters, and can be used directly
Example	Example 1 FcInit (Tool1, Wobj0, 0); SetControlType(1); SetSineOverlay(0， 10， 5， 3.14， 2); FcStart(); StartOverlay(); StopOverlay(); Stop the overlays.
Attention	The calling of the interface is of practical value can only after executing StartOverlay.

15.4.4.18FcCondForce

Explanation	It is used to define termination conditions related to contact force
Definition	**FcCondForce(xmin, xmax, ymin, ymax, zmin, zmax, IsInside, TimeOut);** **Xmin**, to define the lower limit of the force limit in the X-direction. It indicates the maximum value in the negative X-direction if the value is negative. The unit is N and the default value is negative infinity. Data type: double **Xmax**, to define the upper limit of the force limit in the X-direction. It indicates the minimum value in the negative X direction if the value is negative. The unit is N and the default value is positive infinity. Data type: double **Ymin**, to define the lower limit of the force limit in the Y-direction. It indicates the maximum value in the negative Y direction if the value is negative. The unit is N and the default value is negative infinity. Data type: double **Ymax**, to define the upper limit of the force limit in the Y-direction. It indicates the minimum value in the negative Y direction if the value is negative. The unit is N and the default value is positive infinity. Data type: double **Zmin**, to define the lower limit of the force limit in the Z-direction. It indicates the maximum value in the negative Z direction if the value is negative. The unit is N and the default value is negative infinity. Data

	type: double **Zmax**, to define the upper limit of the force limit in the Z-direction. It indicates the minimum value in the negative Z direction if the value is negative. The unit is N and the default value is positive infinity. Data type: double **IsInside**, to define whether the internal/external restriction condition is true. Data type: bool TimeOut, to define the timeout period in seconds, range: 1−600. Data type: double
Example	Example 1 FcInit (Tool1, Wobj0, 0); FcStart(); FcCondForce (-100， 100， -100， 100， -100， 100， true， 60); Define a termination condition. The condition is true when the contact force is within plus or minus 100 N in the x/y/z-axis direction of the force control frame, and terminates when it exceeds 100 N. The timeout period is 60 seconds.
Attention	This interface can only be called after executing FcStart and before executing FcStop. If not, the termination conditions of the contact force will not be set successfully.

15.4.4.19FcCondPosBox

Explanation	It is used to define termination conditions related to contact location
Definition	**FcCondPosBox(SupvFrame, Box, IsInside, Timeout);** **SupvFrame**, to select which coordinate system is defined relative to the monitored spatial body. The frame is derived by converting a work object frame onto a frame. The conversion of the frame is defined by pose. By default, pose0 is used. That is, the work object frame is used without using any conversion. Data type: pose. **Box**, to define a cuboid. Data type: fcboxvol **IsInside**, to define whether the internal/external restriction condition is true. Data type: bool TimeOut, to define the timeout period in seconds, range: 1−600. Data type: double
Example	Example 1 FcInit (Tool1, Wobj0, 0); FcStart(); VAR fcboxvol box1 = fcbv:{-100.0, 100.0, -200.0, 200.0, -300.0, 300.0}; VAR pose pose1 = pe:{{0, 0, 0},{1, 0, 0, 0}}; FCCondPosBox (pose1, box1, false, 60); Define a termination condition. The termination condition is triggered when the robot TCP enters the defined cuboid or waits more than 60 seconds.
Attention	This interface can only be called after executing FcStart and before executing FcStop. If not, the termination conditions of the cuboid location will not be set successfully.

15.4.4.20FcCondTorque

Explanation	It is used to define termination conditions related to contact torque.
Definition	**FcCondTorque(xmin, xmax, ymin, ymax, zmin, zmax, IsInside, TimeOut);** **Xmin**, to define the lower limit of the torque limit in the X-direction. It indicates the maximum value in the negative X-direction if the value is negative. The unit is N.m and the default value is negative infinity. Data type: double **Xmax**, to define the upper limit of the torque limit in the X-direction. It indicates the minimum value in the negative X-direction if the value is negative. The unit is N.m and the default value is positive infinity. Data type: double **Ymin**, to define the lower limit of the torque limit in the Y-direction. It indicates the maximum value in the negative Y-direction if the value is negative. The unit is N.m and the default value is negative infinity. Data type: double **Ymax**, to define the upper limit of the torque limit in the Y-direction. It indicates the minimum value in the negative Y-direction if the value is negative. The unit is N.m and the default value is positive infinity. Data type: double **Zmin**, to define the lower limit of the torque limit in the Z-direction. It indicates the maximum value in the negative Z-direction if the value is negative. The unit is N.m and the default value is negative infinity. Data type: double **Zmax**, to define the upper limit of the torque limit in the Z-direction. It indicates the minimum value in the negative Z-direction if the value is negative. The unit is N.m and the default value is positive infinity. Data type: double **IsInside**, to define whether the internal/external restriction condition is true. Data type: bool TimeOut, to define the timeout period in seconds, range: 1−600. Data type: double
Example	Example 1 FcInit (Tool1, Wobj0, 0); FcStart(); FcCondTorque (-10， 10， -10， 10， -10， 10， true， 60); Define a termination condition. When the contact torque is greater than 10 N.m in any direction of the

	force control frame, or the time exceeds 60s, the termination condition is triggered.
Attention	This interface can only be called after executing FcStart and before executing FcStop. If not, the termination conditions of the contact torque will not be set successfully.

15.4.4.21FcCondWaitWhile

Explanation	It is used to activate the previously defined termination conditions and wait until these conditions become False or timeout in the current line.
Definition	No parameters, and can be used directly
Example	Example 1 FcInit (Tool1, Wobj0, 0); FcStart(); FcCondTorque (-10， 10， -10， 10， -10， 10， true， 60); FcCondForce (-100， 100， -100， 100， -100， 100， true， 60); FcCondWaitWhile(); Activate the termination conditions. The program blocks at the current position and waits for the termination conditions to be triggered.
Attention	It can be used after the force control termination conditions are defined.

15.4.4.22FcMonitor

Explanation	It is used to enable or disable the force control module protection monitor. Force control protection monitor refers to the use of user-set protection parameters by the controller in impedance mode to limit the speed, momentum, power and other states of the robot, in order to achieve protection in impedance mode.
Definition	FcMonitor (On); enable the force control module protection monitor, and the user-set protection parameters take effect during the impedance motion. FcMonitor (Off); disable the force control module protection monitor. The user-set protection parameters are not effective in impedance motion, and the controller will use default protection parameters to limit the robot's motion status.

Example	Example 1
	FcInit (tool0,wobj0,0);
	SetControlType (0);
	SetFcJointVelMax (1.0, 1.0, 1.0, 0.5, 0.5, 0.5, 0);
	SetFcJointEnergyMax (100, 100, 100, 100, 100, 100, 0);
	FcMonitor (On);//Enable the force control module protection monitor
	FcStart();
	…
	FcMonitor (Off);//Disable the force control module protection monitor
	…
	FcStop();

Attention	● It is recommended to set the protection parameters first when using impedance mode, and then use FcMonitor On to enable the force control module protection monitor. ● If the protection parameters are not set or if FcMonitor Off is used to disable the force control module protection monitor, the controller will take effect with the default protection parameters. ● The default protection parameters have significant limitations, making it easy to trigger force control protection errors when using impedance mode.

15.4.4.23GetEndToolTorque

Explanation	It is used to get the current robot torque
Definition	**GetEndToolTorque(Tool, Wobj [, RefType]);** The parameter in [] can be ignored. Return value, torque information, data type: TorqueInfo **Tool**, the information of the tool currently in use. Data type: Tool **Wobj**, the information of the work object currently in use. Data type: Wobj **RefType**, reference frame relative to the torque, data type: Int ● 0: default, torque information of the end-effector relative to the world frame ● 1: torque information of the end-effector relative to the flange frame ● 2: torque information of the end-effector relative to the tool frame

Example	TorqueInfo tmp_info = GetEndtoolTorque(tool1, wobj1);
	//Obtain the information architecture of the torque applied to the tool at the end-effector of the robot in the case of tool1 wobj1
	…
	Print(tmp_info.joint_torque.measure_torque);
	Print(tmp_info.joint_torque.external_torque);
	//Print the measured force and external force of each axis

	…
	Print(tmp_info.cart_torque.m_torque);
	//Print Cartesian space torque
	…
	Print(tmp_info.cart_torque.m_force[1]);
	Print(tmp_info.cart_torque.m_torque[1]);
	//Print information of force and torque in X direction

15.4.4.24SetFcJointVelMax

Explanation	It is used to set the maximum axis velocity during impedance motion.
Definition	**SetFcJointVelMax jnt1(vel，jnt2_vel，jnt3_vel，jnt4_vel，jnt5_vel， jnt6_vel，jnt7_vel);_** **Jnt1_ vel**, data type: double, the maximum velocity of Joint 1, in rad/s. **Jnt2_ vel**, data type: double, the maximum velocity of Joint 2, in rad/s. **Jnt3_ vel**, data type: double, the maximum velocity of Joint 3, in rad/s. **Jnt4_ vel**, data type: double, the maximum velocity of Joint 4, in rad/s. **Jnt5_ vel**, data type: double, the maximum velocity of Joint 5, in rad/s. **Jnt6_ vel**, data type: double, the maximum velocity of Joint 6, in rad/s. **Jnt7_ vel**, data type: double, the maximum velocity of Joint 7, in rad/s.
Example	SetFcJointVelMax(1.0,1.0,1.0,1.0,0.5,0.5,0.5);

Maximum velocity of each joint of the collaborative robot, in rad/s

	J1	J2	J3	J4	J5	J6	J7
ER3P	4.7	4.7	4.7	4.7	6.2	6.2	6.2
ER7P	4.7	4.7	4.7	4.7	6.2	6.2	6.2
ER3	4.7	4.7	4.7	6.2	6.2	6.2	
ER7	4.7	4.7	4.7	6.2	6.2	6.2	
SR3	4.7	4.7	4.7	6.2	6.2	6.2	
SR4	4.7	4.7	4.7	6.2	6.2	6.2	
SR5	4.7	4.7	4.7	6.2	6.2	6.2	
CR7	4.7	4.7	4.7	6.2	6.2	6.2	
CR12	4.7	4.7	4.7	6.2	6.2	6.2	
CR17	4.7	4.7	4.7	6.2	6.2		
CR18	4.7	4.7	4.7	6.2	6.2	6.2	
CR20	4.7	4.7	4.7	6.2	6.2	6.2	
CR25	4.7	4.7	4.7	6.2	6.2		

15.4.4.25SetFcCartVelMax

Explanation	It is used to set the maximum Cartesian velocity during impedance motion.
Definition	**SetFcCartVelMax(vel_x，vel_y，vel_z，vel_a，vel_b， vel_c);** **Vel_x**, data type: double, maximum linear velocity in the X direction, in m/s. **Vel_y**, data type: double, maximum linear velocity in the Y direction, in m/s. **Vel_z**, data type: double, maximum linear velocity in the Z direction, in m/s. **Vel_a**, data type: double, maximum angular velocity around the X axis, in rad/s. **Vel_b**, data type: double, maximum angular velocity around the Y axis, in rad/s. **Vel_c**, data type: double, maximum angular velocity around the Z axis, in rad/s.
Example	SetFcCartVelMax (1.0,1.0,1.0,0.5,0.5,0.5);

Maximum Cartesian velocity of the collaborative robot, in m/s, /rad/s

	Vel_x	Vel_y	Vel_z	Vel_a	Vel_b	Vel_c
ER3P	2.0	2.0	2.0	4.7	4.7	4.7
ER7P	2.0	2.0	2.0	4.7	4.7	4.7
ER3	2.0	2.0	2.0	4.7	4.7	4.7
ER7	2.0	2.0	2.0	4.7	4.7	4.7
SR3	2.0	2.0	2.0	4.7	4.7	4.7
SR4	2.0	2.0	2.0	4.7	4.7	4.7
SR5	2.0	2.0	2.0	4.7	4.7	4.7
CR7	2.0	2.0	2.0	4.7	4.7	4.7
CR12	2.0	2.0	2.0	4.7	4.7	4.7
CR17	2.0	2.0	2.0	4.7	4.7	4.7
CR18	2.0	2.0	2.0	4.7	4.7	4.7
CR20	2.0	2.0	2.0	4.7	4.7	4.7
CR25	2.0	2.0	2.0	4.7	4.7	4.7

15.4.4.26SetFcJointMomentumMax

Explanation	It is used to set the maximum angular momentum of joints during impedance motion.
Definition	**SetFcJointMomentumMax(jnt1_moment，jnt2_moment，jnt3_moment，jnt4_moment，jnt5_moment， jnt6_moment，jnt7_moment);** **Jnt1_moment**, data type: double, maximum angular momentum of Joint 1, in kg*m/s. **Jnt2_moment**, data type: double, maximum angular momentum of Joint 2, in kg*m/s. **Jnt3_moment**, data type: double, maximum angular momentum of Joint 3, in kg*m/s. **Jnt4_moment**, data type: double, maximum angular momentum of Joint 4, in kg*m/s. **Jnt5_moment**, data type: double, maximum angular momentum of Joint 5, in kg*m/s. **Jnt6_moment**, data type: double, maximum angular momentum of Joint 6, in kg*m/s. **Jnt7_moment**, data type: double, maximum angular momentum of Joint 7, in kg*m/s.
Example	SetFcJointMomentumMax (0.1, 0.1, 0.1, 0.1, 0.055, 0.055, 0.055);

Maximum angular momentum of each joint of the collaborative robot, in kg*m/s

	J1	J2	J3	J4	J5	J6	J7
ER3P	1.0	1.0	1.0	1.0	0.55	0.55	0.55
ER7P	2.0	2.0	1.0	1.0	0.55	0.55	0.55
ER3	1.0	1.0	1.0	0.55	0.55	0.55	
ER7	2.0	2.0	1.0	0.55	0.55	0.55	
SR3	0.55	0.55	0.55	0.2	0.2	0.2	
SR4	1.0	1.0	0.55	0.2	0.2	0.2	
SR5	1.0	1.0	0.55	0.2	0.2	0.2	
CR7	2.0	2.0	1.0	0.55	0.55	0.55	
CR12	3.5	3.5	2.0	1.0	0.55	0.55	
CR17	7.0	7.0	5.5	3.5	2.0		
CR18	3.5	3.5	2.0	1.0	0.55	0.55	
CR20	7.0	7.0	3.5	2.0	1.0	1.0	
CR25	7.0	7.0	3.5	2.0	1.0		

15.4.4.27SetFcJointEnergyMax

Explanation	It is used to set the maximum power of joints during impedance motion.
Definition	**SetFcJointEnergyMax (jnt1_ energy，jnt2_ energy，jnt3_ energy，jnt4_ energy，jnt5_ energy，jnt6_ energy，jnt7_ energy);** **Jnt1_ energy**, data type: double, maximum power of Joint 1, in kg.m2/s3. **Jnt2_ energy**, data type: double, maximum power of Joint 2, in kg.m2/s3. **Jnt3_ energy**, data type: double, maximum power of Joint 3, in kg.m2/s3. **Jnt4_ energy**, data type: double, maximum power of Joint 4, in kg.m2/s3. **Jnt5_ energy**, data type: double, maximum power of Joint 5, in kg.m2/s3. **Jnt6_ energy**, data type: double, maximum power of Joint 6, in kg.m2/s3. **Jnt7_ energy**, data type: double, maximum power of Joint 7, in kg.m2/s3.
Example	SetFcJointEnergyMax (100, 100, 100, 100, 100, 100, 100);

Maximum power of each joint of the collaborative robot, in kg*m2/s3

	J1	J2	J3	J4	J5	J6	J7
ER3P	2500.0	2500.0	2500.0	2500.0	1500.0	1500.0	1500.0
ER7P	4000.0	4000.0	3000.0	3000.0	2000.0	1000.0	1000.0
ER3	2500.0	2500.0	2500.0	1500.0	1500.0	1000.0	
ER7	4000.0	4000.0	3000.0	2000.0	1000.0	1000.0	
SR3	1500.0	1500.0	1500.0	600.0	600.0	600.0	
SR4	2500.0	2500.0	1500.0	600.0	600.0	600.0	
SR5	2500.0	2500.0	1500.0	600.0	600.0	600.0	
CR7	4000.0	4000.0	3000.0	2000.0	1000.0	1000.0	
CR12	8000.0	8000.0	5000.0	3500.0	1500.0	1500.0	
CR17	8000.0	8000.0	5000.0	1500.0	1500.0		
CR18	4000.0	4000.0	3000.0	2000.0	1000.0	1000.0	
CR20	16000.0	16000.0	8000.0	4500.0	2500.0	2500.0	
CR25	16000.0	16000.0	8000.0	4500.0	2500.0		

15.4.5Drag and replay

15.4.5.1ReplayPath

Explanation	It is used to replay the recorded trajectory using drag teaching. You can control the running rate during replay. Note: If the velocity of the recorded drag trajectory is too high and the replay rate is set too high, it is easy

	to trigger servo alarms and damage the machine. It is recommended to gradually increase the replay rate from low velocity.
Definition	**ReplayPath(path [, rate] [, wobj/tool]);** **Path**, data type: path, type of drag and replay path, which is defined in the path list generated by drag teaching. **Rate**, data type: double, replay percentage, range: 0.01−3.00. 0.01 means replay at 1% running rate when dragging; 1.00 at 100% running rate; and 3.00 at 300% running rate. **wobj/tool**, data type: tool/work object, to specify the end-effector for the replay command to be a tool or work object. During the replay, the robot will change the replay control parameters according to the tool of the corresponding device to improve the operating stability
Example	Example 1 ReplayPath(path , 1, tool1); Use the original running rate to record and replay.

15.4.6IO commands

15.4.6.1SetDO

Explanation	It is used to set the value of a digital output signal. If the command is performed after the motion command, it will not interrupt the turning zone and be triggered at the end of the motion command trajectory or at the starting point of the turning zone. See Example 2 for specific usage.
Definition	**SetDO(DoName，Value);** **DoName**, data type: signaldo, to determine the name of the DO signal that needs to change state, which must be a variable that has already been defined in the input/output interface. **Value**, data type: bool, the target state of the DO signal, and only true and false are available.
Example	Example 1 SetDO(do2，true); The digital output point corresponding to do2 is set at a high level. Example 2 Scenario 1: IO commands between two motion commands with turning zones is adopted MoveL(p1，v1000，z50，tool0); SetDO(do2，true); MoveL(p2，v1000，z50，tool0); At this point, the SetDO command is triggered at the starting point of the turning zone from p1 to p2. Scenario 2: There are no longer motion commands after using the IO command with the turn area motion command MoveL (p1， v1000， z50， tool0); SetDO (do2， true); ....(There are no subsequent motion commands or commands to interrupt the turning zone) At this point, the SetDO command is triggered to execute after the motion command reaches p1. Scenario 3: Motion command does not include the turning zone MoveL (p1， v1000， fine， tool0); SetDO (do2， true); ....(Regardless of subsequent commands) At this point, the SetDO command is triggered to execute after the motion command reaches p1.

15.4.6.2SetAllDO

Explanation	It is used to set the value of all digital output signals. If the command is performed after the motion command, it will not interrupt the turning zone and be triggered at the end of the motion command trajectory or at the starting point of the turning zone. See Example 2 of SetDO for specific usage.
Definition	**SetAllDO(Value);** **Value**, data type: bool, the target state of the DO signal, and only true and false are available.
Example	Example 1 SetAllDO (true); Set all digital output voltages to a high level, except DO bound with system function.

15.4.6.3SetGO

Explanation	It is used to set the value of a group. If the command is performed after the motion command, it will not interrupt the turning zone and be triggered at the end of the motion command trajectory or at the starting point of the turning zone. See Example 2 of SetDO for specific usage.

Definition	**SetGO(GoName，Value);** **GoName**, data type: signalgo, to determine the name of the go signal that needs to change value, which must be a variable that has already been defined in the input/output interface. **Value**, data type: int, the target value of go signal. Note: The maximum supported value is 2,147,483,648 (2^{31}).
Example	Example 1 SetGO (go3，8); Set the value of a set of physical ports corresponding to go3 as 8.

15.4.6.4SetAO

Explanation	It is used to set the value of an analog output signal. If the command is performed after the motion command, it will not interrupt the turning zone and be triggered at the end of the motion command trajectory or at the starting point of the turning zone. See Example 2 of SetDO for specific usage.
Definition	**SetAO(AoName，Value);** **AoName**, data type: signalao, to determine the name of the ao signal that needs to change value, which must be a variable that has already been defined in the input/output interface. **Value**, data type: double, the target value of the ao signal.
Example	Example 1 SetAO (ao3，5.123); Set the value of a set of physical ports corresponding to ao3 as 5.123.

15.4.6.5PulseDO

Explanation	It is used to generate a pulse of the DO signal. If the command is performed after the motion command, it will not interrupt the turning zone and be triggered at the end of the motion command trajectory or at the starting point of the turning zone. See Example 2 of SetDO for specific usage.
Definition	**PulseDO ([\High,] [length,] signal);** [\High], when the command is executed, regardless of the current state, the signal state is always set to high (1). **[length]**, to specify pulse length: 0.001-2000s. Default to 0.2s when missing. Data type:double or int **signal**, the signal to generate the pulse. Data type:signaldo
Attention	If SetDO/SetGO is executed during PulseDO, PulseDO will be invalid and SetDO/SetGO will be executed.

15.4.6.6PulseReg

Explanation	It is used to specify a register to generate a pulse signal for a specified time and restore the initial value of the register after the end of the time. If this command is performed after the motion command, it will not interrupt the turning zone but will be triggered at the end of the motion instruction trajectory or at the starting point of the turning zone. See Example 2 of SetDO for specific usage.
Definition	**PulseReg (Register, Value, Time);** **Register**, name of the register to generate the pulse signal, data type: Bit/Bool register **Value**, to specify the value of the pulse signal, data type: Bool. **Time**, the duration of the pulse signal in seconds, with a limit range of [0.001, 10.0]. Data type: double
Attention	If WriteRegByName or register equal assignment is executed during PulseReg, the valid value of the register will take effect depending on the last executed command. But the initial value before executing PulseReg will be restored after the time period specified by PulseReg ends.

15.4.7Communication commands

In the RL program, the robot can communicate with external devices through both Ethernet and serial ports. A unified set of commands is designed for resource management and data sending and receiving, which ensures consistent use experience.

Command set	TCP client	TCP server	Serial port
OpenDev	Y	Y	Y
SocketAccept	N/A	Y	N/A
CloseDev	Y	Y	Y
SendString	Y	Y	Y
SendByte	Y	Y	Y
ReadBit	Y	Y	Y
ReadByte	Y	Y	Y
ReadDouble	Y	Y	N/A
ReadInt	Y	Y	N/A
ReadString	Y	Y	Y
GetSocketConn	Y	N/A	N/A
GetSocketServer	N/A	Y	N/A

GetBufSize	N/A	N/A	Y
ClearBuffer	Y	Y	Y

15.4.7.1OpenDev

Explanation	It is used to open a listening server, initiate a connection as a client, and open a serial port resource, depending on the object indicated by the parameter. When opening the SocketServer object, the robot will initiate resource and complete port binding and port listening. When opening the SocketConn object, the robot will act as a TCP client and try to connect to the external server according to the preset ip and port. When opening the serial port resource, the serial port will be initialized according to the window parameters and communication conditions will be provided.
Definition	**OpenDev(name);** **name**, data type: string, the name of the client object or server object or serial port resource.

Example	Example 1
	SocketConn scnn3 = {"192.168.0.200", 8090, "clt1", 2, "\n"};
	Try
	OpenDev("clt1") // Try to connect to the remote server. If the connection is successful, the attr of clt1 will be modified to outgoing automatically.
	string readstr = ReadString(30, "clt1");
	..... // Logic processing of readstr
	string sendstr = "hello server！ ";
	SendString(sendstr , "clt1"); //Use clt1's client connection to send data
	... // A series of code
	catch(ERROR e); // ERROR error type, including the file that generated the error, line number, error code, and error content
	... // A series of exception handling
	Endtry
	
	Example 2
	SocketServer listener1 = {"192.168.0.200", 8090, "svr1"};
	global pers bool exit = false;
	try
	OpenDev("svr1"); //Bind port, listening port
	while(exit != true)
	SocketConn conn = SocketAccept("svr1"); // Client connected via blocking receive
	Endwhile
	catch(ERROR e);
	... // A series of exception handling
	Endtry
	If an error is reported, the control system will throw an exception and report the cause of the error. If the exception is not caught by the try block, the control system will stop the program.

15.4.7.2SocketAccept

Explanation	It is used to block wait for client connections to arrive and complete client connection. This command is only used when the robot is acting as a TCP server. This command is only used when the robot is acting as a TCP server.
Definition	Return value, data type: SocketConn, after an external device connects to the robot as a TCP client, the control system generates a communication object that is used by the RL program to control communication read and write. **SocketConn conn = SocketAccept(name);** name, data type: string, the name of the SocketServer object that has been prepared and opened successfully using OpenDev.

Example	Example 1
	SocketServer listener1 = {"192.168.0.200", 8090, "svr1"};
	global pers bool exit = false;
	try
	OpenDev("svr1") //Bind port, listening port
	while(exit != true)
	SocketConn conn = SocketAccept("svr1"); // Client connected via blocking receive
	conn.name = "client1"; // Important! Give the communication connection a name, otherwise, it will be difficult to read and write data by name
	conn.suffix = "\n"; // Optional, set the packet terminator
	Endwhile
	catch(ERROR e)
	... // A series of exception handling

	Endtry If an error is reported, the control system will throw an exception and report the cause of the error. If the exception is not caught by the try block, the control system will stop the program.
Attention	● The command will block the current task, so the correct way to use it is in multitasking. There is a low-priority task continuously receiving and generating the communication connection object SocketConn independently. ● The command returns a connection operation object and has the ip and port information of the client connection, which can be used by other parts of the program. The returned connection object is a SocketConn structure with a name randomly assigned by the system. After getting the connection object, please change the name of the connection object to avoid connection loss. ● The server supports multiple connections.

Note

To ensure the stability of the robot's motion control, the control system allocates only a portion of its computational resources to network communication functions. When the robot acts as a socket server listening for connections, if it receives extremely frequent network connection requests or data streams resembling a "DDoS attack", this may cause the robot's network connections to external devices (such as the teach pendant and other equipment) to disconnect or result in operational lag.

Network connection or data interaction frequency must remain below 1 per millisecond (1/ms).

15.4.7.3CloseDev

Explanation	It is used to close the resource, which can be used to close the TCP communication connection, TCP listening server, or serial port resource.
Definition	**CloseDev(name);** name, data type: string, SocketConn connection, listening server SocketServer object, or serial port resource used for communication.

Example	Example 1
	SocketConn scnn3 = {"192.168.0.200", 8090, "clt1", 2, "\n"};
	Try
	   OpenDev("clt1");
	   string readstr = ReadString(30, "clt1");
	   ..... // Logic processing of readstr
	   string sendstr = "hello server！ ";
	   SendString (sendstr, "clt1"); //Use clt1's client connection to send data
	   ... // A series of code
	catch(ERROR e);
	   ... // A series of exception handling
	endtry
	CloseDev("clt"); //Close the socket client at last, regardless of whether an error occurs.
	
	Example 2
	SocketServer listener1 = {"192.168.0.200", 8090, "svr1"};
	global pers bool exit = false;
	try
	   OpenDev("svr1"); //Bind port, listening port
	   while(exit != true)
	   SocketConn conn = SocketAccept("svr1"); // Client connected via blocking receive
	   conn.name = "client1"; // Important! Give the communication connection a name, otherwise, it will be difficult to read and write data by name
	   conn.suffix = "\n"; // Optional, set the packet terminator
	   Endwhile
	   catch(ERROR e)
	   ... // A series of exception handling
	Endtry;
	CloseDev("client1"); //Close communication with external TCP client. Important!
	CloseDev("svr1"); //Close the listening server
	
	In Example 2, there are two network objects, and you must close the communication connection first and then the server object, otherwise it will generate a state of incomplete resource release (TCP TIME_WAIT state).
	If the robot has established multiple communication connections with external devices when it acts as a server, you need to close these communication connections in order before closing the server.

	In the case of incomplete resource release, the control system needs to be restarted. However, there is no need to worry too much, as there is redundancy in the number of resources allowed in the control system; this ensures the program runs properly after a small number of resources are occupied. However, it is necessary to avoid a large number of resources being occupied due to incorrect use.

15.4.7.4SendString

Explanation	It is used to send a string outwards. It can be sent through the network or serial port, depending on the hardware resource represented by the identifier in the parameter.
Definition	**SendString(StringData， name);** **StringData**, data type: string, the string data to be sent. name, data type: string, the name of the hardware resource used to send the data. It can be the SocketConn object with an established TCP communication connection or the serial port resource successfully opened.
Example	Example 1 SendString(“Hello World”， “Socket0”); Send Hello World string outwards through Socket0. Socket0 is the SocketConn type that has been defined and successfully connected. Example 2 VAR String str1 =“Hello World”; SocketSendString(str1， “Serial1”); Sends the string Hello World stored in str1 outwards via Serial1. Serial1 is a defined and successfully opened serial port.

15.4.7.5SendByte

Explanation	It is used to send a byte outwards. It is very useful when sending ASCII characters.
Definition	**SendByte(ByteData, name);** **ByteData**, data type: int, byte, or byte array, to send an unsigned byte or array from 0 to 255, mainly used for ASCII codes. **Name**, data type: string, the name of the socket or serial port to send data.
Example	Example 1 SendByte(13, “socket0“); Send a carriage return through Socket0. Example 2 VAR byte data1 = 13; SendByte(data1, “serial0“); First define a byte variable data1, which is actually a carriage return. Then send the data outwards through serial0. Example 3 VAR byte data2[2] = {13,17}; SendByte(data2, “socket0”); Send an array variable byte data2 through socket0. Sent all in the array. Example 4 VAR byte data2[2] = {13,17,20}; SendByte(data2[2], “socket0”); Sends a byte variable of data2[2] through socket0, which represents the 2nd element of the array. The value 17 of data2[2] will be sent without sending any other elements.

15.4.7.6ReadBit

Explanation	The control system receives data by bit. 1) Received by TCP through network communication. The externally sent data should end with the terminator configured by SocketConn. 2) Received by serial communication. The external device only needs to send the data, with no requirement on the terminator.
Definition	Return value, data type: bool array, to store the received bit data using a bool array. Each bit corresponds to a bool member. **Ret = ReadBit(BitNum, TimeOut, name);** **BitNum**, data type: int, the number of bits that need to be read. The size should be an integer multiple of 8. **TimeOut**, data type: int, timeout period, in s, ranging from 0 to 86400 and 60s by default. **name**, data type: string, the name of the communication connection SocketConn or the serial port. **Ret**, data type: bool array, received data. The first element of the array indicates the lowest bit.

Example	Example 1 bool groupio[16]; groupio = ReadBit(16, 60, “Socket0”); 16 bit data is read by the ReadBit command and stored in a bool array named groupio with a timeout period of 60 seconds. Assume that the external device sends ASCII characters, 95 + terminator, the robot receives "95". As the hexadecimal values of "9" and "5" are 0x39 and 0x35 respectively, the data received by the user is 0x3935. At this time, the groupio array from [1] to [16] is 1001 1100 1010 1100. The [1] is the low bit of the data, which matches with 0x3935.

15.4.7.7ReadByte

Explanation	It is used to receive data with a certain number of bytes. Note that the data needs to be separated by commas.
Definition	Return value, data type: byte array, to store the received data using a byte array. **Ret = ReadByte(ByteNum, TimeOut, name);** **ByteNum**, data type: int, the number of bits that need to be read. The size should be an integer multiple of 8. **TimeOut**, data type: int, timeout period, in s, ranging from 0 to 86400 and 60s by default. **name**, data type: string, the name of the communication connection SocketConn or the serial port. **Ret**, data type: byte array, received data.
Example	Example 1 byte rets[6] = {0,0,0,0,0,0}; rets = ReadByte(6,60,"clt1"); 6-byte data is read and stored in a bool array named rets with a timeout period of 60 seconds.
Attention	Note that bytes from external devices need to be separated by commas, e.g. send "1,2,3,4,5,6". When sending data via TCP, the data should end with the pre-defined terminator. When sending data via serial port, the terminator is not required.

15.4.7.8ReadDouble

Explanation	It is used to receive double-type data via Socket. The sent data should end with the pre-defined terminator. Note that this command is only valid for TCP network communication and when robots act as the client/server, but not for serial ports.
Definition	Return value, data type: double array, to store the received data using a double array. **Ret = ReadDouble(DoubleNum, TimeOut, name);** **DoubleNum**, data type: double, the number of doubles to be read, up to 4,096. **TimeOut**, data type: int, timeout period, in s, ranging from 0 to 86400 and 60s by default. **name**, data type: string, the name of the Socket used to receive the data.
Example	Example 1 double dd[10]; dd =ReadDouble(10, 60, “Socket0”) ; Read 10 double-type data and store them in a double array named dd with a timeout period of 60 seconds.

15.4.7.9ReadInt

Explanation	It is used to receive int-type data via Socket. Externally sent data must end with the pre-defined terminator. Note that this command is only valid for TCP network communication and when robots act as the client/server, but not for serial ports.
Definition	Return value, data type: int, to store the received data using an int array. **Ret = ReadInt(IntNum, TimeOut, name);** **IntNum**, data type: int, the number of int to be read, up to 4,096. **TimeOut**, data type: int, timeout period, in s, ranging from 0 to 86400 and 60s by default. **name**, data type: string, the name of the Socket used to receive the data.
Example	Example 1 int ii[10]; ii = ReadInt(10, 60, “Socket0”) ; 10 int data are read and stored in an int array named ii with a timeout period of 60 seconds.

15.4.7.10ReadString

Explanation	It is used to read a string and return it. Externally sent data should end with the pre-defined terminator.
Definition	Return value, data type: string, to store the received string. **Ret = ReadString(TimeOut, name, [len]);** **TimeOut**, data type: int, timeout period, in s, ranging from 0 to 86400 and 60s by default. **name**, data type: string, the name of the socket or serial port to receive data.

	len, data type: int, optional parameter, only used when reading through the serial port. Since the terminator is not defined in the serial port, it is necessary to specify the length before successful reading and parsing.
Example	Example 1 VAR String str1 str1 = ReadString(60, “Socket1”); Receive a string from Socket1 and store it in str1 with a timeout period of 60 seconds. Network communication. Example 2 VAR String str1 str1 = ReadString(60, “serial0”,5); Receive a string for a length of 5 bytes from serial0 and store it in str1 with a timeout period of 60 seconds. Serial port communication.

15.4.7.11GetSocketConn

Explanation	It is used to find the socket attribute set object using the socket connection name. The result obtained by this command can be used for judgment and processing logic. It should be used only as a read-only object. This command is only applicable to communication connections (including robot as client, or as a server which has been connected to the channel for communication), not for listening servers and serial ports.
Definition	Return value, data type: SocketConn, the socket attribute object found by given name. **Ret = GetSocketConn(name);** **name**, data type: string, the name of the communication connection SocketConn. **Ret**, data type: SocketConn, the socket attribute object found by given name.

Queryable properties	Query method	Meaning and example
ip address	ret.ip	String, e.g. "192.168.0.161"
Port number	ret.port	integer, e.g. 8090
Attribute	ret.attr	Robot as server: "incoming". Robot as client: "outgoing". If the connection is not established: "" or other value, usually blank
Cache size	ret.cache	1−100
Name	ret.name	In the given example, it is "client0"
Connection state	ret.state	closed, establish

Example	Example 1 SocketConn ret= GetSocketConn(“client0”); Find SocketConn object with the name "client0". You can use ret to get the attributes of this connection, including the ip address, port number, communication terminator, and connection state.

15.4.7.12GetSocketServer

Explanation	Find the corresponding server attribute set object with the user-defined name. The result obtained by this command can be used for judgment and processing logic. It should be used only as a read-only object. This command is only applicable to listening servers (SocketServer objects), not to communication connections (including robot as client, or as a server which has been connected to the channel for communication) and serial ports.
Definition	Return value, data type: SocketServer, the server attribute object found by given name. **Ret = GetSocketServer(name);** **Name**, data type: string, the name of communication connection SocketServer. **Ret**, data type: SocketServer, the socket attribute object found by given name.

Queryable properties	Query method	Meaning and example
ip address	ret.ip	String, e.g. "192.168.0.161"
Port number	ret.port	integer, e.g. 8090
Name	ret.name	In the example above, it is "svr1"
Connection state	ret.state	closed, listening, error

Example	Example 1
	SocketServer listener1 = {"192.168.0.200", 8090, "svr1"};
	OpenDev("svr1"); //Bind port, listening port
	//Get the SocketServer object using the connection identifier "svr1", at this time ret will copy all the states of listener1 in Task 1
	SocketServer ret= GetSocketServer("svr1");
	if(ret.state == "listening") //Use SocketServer's attr attribute to judge if listening is in underway
	//Logic processing
	endif

15.4.7.13GetBufSize

Explanation	It is used to get the amount of data not read in the buffer of the serial port, in bytes. The command is only applicable to the serial port, not to the TCP server and the client.
Definition	Return value, data type: int, the amount of unprocessed data in the buffer, in bytes. **Ret = GetBufSize(name);** **name**, data type: string, the name of the serial port resource. **Ret**, data type: int, the amount of unprocessed data in the buffer, in bytes.
Example	Example 1 OpenDev("serial0"); int a = GetBufSize("serial0"); Print(a);

15.4.7.14ClearBuffer

Explanation	Clear the connected buffer, and any unread data will be lost. The serial port and socket data are supported. Data that has been split by the terminator and data that has not been split will be cleared.
Definition	**ClearBuffer(name);** **name**, data type: string, name of the link.
Example 1	**Assuming that the terminator is \r, two copies of data have been received and one copy of data is being received. After executing this command, all the data in the buffer will be cleared, and the RL program can only read the re-sent data after clearing** **123456789\r** **Abcdefg\r** **mmmmmm**

15.4.7.15ReadOpcUaVarByName

Explanation	It is used to read the value of OPC-UA custom variables by name.
Definition	**ReadOpcUaVarByName(name, value);** **name**, data type: string, the name of OPC-UA custom variables. **value**, data type: bool/byte/int/double/string, to store the value of OPC-UA custom variables that are read. If the type of the value does not match the type of the OPC-UA custom variable, it will be automatically converted. Note: When converting string to the value type, it is always 0. No return value.
Example	Example 1 int value = 0; ReadOpcUaVarByName(“int_var”, value); Print(value);

15.4.7.16WriteOpcUaVarByName

Explanation	It is used to modify the value of OPC-UA custom variables by name.
Definition	**WriteOpcUaVarByName(name, value);** **name**, data type: string, the name of OPC-UA custom variables. **value**, data type: bool/byte/int/double/string, the modified value of OPC-UA custom variables. If the type of the value does not match the type of the OPC-UA variable, it will be automatically converted. Note: When converting string to the value type, it is always 0.
Example	Example 1 int value = 0; WriteOpcUaVarByName(“int_var”, value); WriteOpcUaVarByName(“int_var”, 123);

15.4.8Network command

15.4.8.1SocketCreate (expired)

Explanation	Establish a Socket connection. By using the Socket instruction, the RL program can obtain data from an external device or send out program data. The RL language supports the simultaneous establishment of multiple different Sockets for the connection of multiple external devices. Different names may be used to distinguish between the different Sockets. The Socket instruction is based on the TCP/IP protocol, so theoretically any external device that supports TCP/IP can communicate with the RL program to exchange data. All data sent to the RL Socket instruction (i.e. data received using the SocketRead series of instructions) should end with a carriage return. All data before the receipt of the carriage return will be merged into the same data processing. When using the Socket function, the robot controller only supports

	connection to an external server as a client. Up to 10 Socket connections are supported. Note that this command is marked as "expired" for it is used in xCore Control System version 1.3. It is still valid in higher versions, but no longer maintained. Further use is not recommended.
Definition	Return value, data type: bool, return true if created successfully and false if failed **SocketCreate（"ip_Address", Port, "Name", [Cache] [, "Terminator"]);** ip_Address, data type: string, to define the ipv4 address that needs to be connected to the server. The double quotation marks shall be used to include it. **Port**, data type: int, to define the server port number. **Name**, data type: string, to define the name of a new Socket. Different names must be specified between different Sockets. **Cache**, data type: int, to define the size of the Socket cache. The communication data is stored in the cache queue and can be omitted. **Terminator**, data type: string, to define the terminator type of socket communication, which can be omitted, default to "\r".

Example	Example:
	if (SocketCreate("10.0.6.11",8080,"S1",10,"\r"))
	// Successful creation
	else
	// Error handling
	endif

Attention	Due to the limitation of the TCP/IP protocol resource release mechanism, do not call the commands SocketCreate and SocketClose frequently. Otherwise, the program may run incorrectly. To avoid frequent calls to the SocketCreate and SocketClose commands in loop mode, it is best to add a time delay between the two commands, e.g. SocketClose("S1"); wait (0.1); SocketCreate("10.0.6.11",8080,"S1",10,"\r");

15.4.8.2SocketClose (expired)

Explanation	It is used to close the Socket. Note that this command is marked as "expired" for it is used in xCore Control System version 1.3. It is still valid in higher versions, but no longer maintained. Further use is not recommended.
Definition	**SocketClose（"SocketName");** **SocketName**, data type: string, the name of Socket to be closed.
Example	Example 1 SocketClose("Socket0");
Attention	Do not use the SocketClose command directly after the SocketSend series of commands. Failure to do so may result in data transmission failures. Use the SocketClose command after receiving the confirmation messages.

15.4.8.3SocketSendString (expired)

Explanation	It is used to send a string outwards via Socket. Note that this command is marked as "expired" for it is used in xCore Control System version 1.3. It is still valid in higher versions, but no longer maintained. Further use is not recommended.
Definition	**SocketSendString（StringData，"SocketName");** **StringData**, data type: string, the string data to be sent. **SocketName**, data type: string, the name of the Socket used to send the data.
Example	Example 1 SocketSendString（"Hello World"，"Socket0"); Send Hello World string outwards through Socket0. Example 2 VAR String str1 ="Hello World" ; SocketSendString（str1，"Socket0"）; Send the str1 stored string via Socket0.

15.4.8.4SocketSendByte (expired)

Explanation	It is used to send a byte outwards through the Socket. It is very useful when sending ASCII characters. Note that this command is marked as "expired" for it is used in xCore Control System version 1.3. It is still valid in higher versions, but no longer maintained. Further use is not recommended.
Definition	**SocketSendByte(ByteData, "SocketName") ;** **ByteData**, data type: int, byte, or byte array, to send an unsigned byte or array from 0 to 255, mainly used

	for ASCII codes. **SocketName**, data type: string, the name of the Socket used to send the data.
Example	Example 1 SocketSendByte(13, “socket0“); Send a carriage return through Socket0. Example 2 VAR byte data1 = 13; SocketSendByte(data1, “socket0“); First define a byte variable data1, which is actually a carriage return. Then send it outwards through socket0. Example 3 VAR byte data2[2] = {13,17} ; SocketSendByte(data2, “socket0”) ; Send an array variable byte data2 through socket0.

15.4.8.5SocketReadBit(expired)

Explanation	It is used to receive data by Bit through the Socket. Externally sent data must end with a carriage return. Note that this command is marked as "expired" for it is used in xCore Control System version 1.3. It is still valid in higher versions, but no longer maintained. Further use is not recommended.
Definition	Return value, data type: bool, to store the received bit data using a bool array. Each bit corresponds to a bool member. **SocketReadBit(BitNum, TimeOut, “SocketName”);** **BitNum**, data type: int, the number of bits that need to be read. The size should be an integer multiple of 8. **TimeOut**, data type: int, timeout period, in s, ranging from 0 to 86400 and 60s by default. **SocketName**, data type: string, the name of the Socket used to receive the data.
Example	Example 1 bool groupio[16]; groupio = SocketReadBit(16, 60, “Socket0”); 16 bit data is read by the SocketReadBit command and stored in a bool array named groupio with a timeout period of 60 seconds.

15.4.8.6SocketReadDouble(expired)

Explanation	It is used to receive double-type data via Socket. Externally sent data must end with a carriage return. Note that this command is marked as "expired" for it is used in xCore Control System version 1.3. It is still valid in higher versions, but no longer maintained. Further use is not recommended.
Definition	Return value, data type: double, to store the received data using a double array. **SocketReadDouble(DoubleNum, TimeOut, “SocketName”);** **DoubleNum**, data type: double, the number of doubles to be read, up to 30. **TimeOut**, data type: int, timeout period, in s, ranging from 0 to 86400 and 60s by default. **SocketName**, data type: string, the name of the Socket used to receive the data.
Example	Example 1 double dd[10]; dd = SocketReadDouble(10, 60, “Socket0”); Read 10 double-type data using the SocketReadDouble command and store it in a double array named dd with a timeout period of 60 seconds.

15.4.8.7SocketReadInt(expired)

Explanation	It is used to receive int-type data via Socket. Externally sent data must end with a carriage return. Note that this command is marked as "expired" for it is used in xCore Control System version 1.3. It is still valid in higher versions, but no longer maintained. Further use is not recommended.
Definition	Return value, data type: int, to store the received data using an int array. **SocketReadInt(IntNum, TimeOut, “SocketName”);** **IntNum**, data type: int, the number of int to be read, up to 30. **TimeOut**, data type: int, timeout period, in s, ranging from 0 to 86400 and 60s by default. **SocketName**, data type: string, the name of the Socket used to receive the data.
Example	Example 1 int ii[10]; ii = SocketReadInt(10, 60, “Socket0”); 10 int data is read by the SocketReadInt command and stored in an int array named ii with a timeout period of 60 seconds.

15.4.8.8SocketReadString(expired)

Explanation	It is used to read a string from Socket and return it. Externally sent data should end with a carriage return. Note that this command is marked as "expired" for it is used in xCore Control System version 1.3. It is still valid in higher versions, but no longer maintained. Further use is not recommended.
Definition	Return value, data type: string, to store the received string. **SocketReadString(TimeOut, “SocketName”);** **TimeOut**, data type: int, timeout period, in s, ranging from 0 to 86400 and 60s by default. **SocketName**, data type: string, the name of the Socket used to receive the data.
Example	Example 1 VAR String str1 str1 = SocketReadString(60, “Socket1”); Receive a string from Socket1 and store it in str1 with a timeout period of 60 seconds.

15.4.9Logic commands

15.4.9.1Return

Explanation	Function return. When the program encounters a RETURN command, if the program is currently in a subroutine, the program will return to the previous function. If the program is currently in the main function, the program ends directly.

15.4.9.2Wait

Explanation	The program waits for a period of time ranging from 0 to 2147484 seconds.
Example	Example 1 Wait (2); It indicates waiting for 2s.

15.4.9.3WaitUntil

Explanation	It is used to wait for a certain condition to be established; the timeout sign is set to true if it is exceeded, and after the waiting, the program proceeds to the next execution.
Definition	**WaitUntil(cond,\MaxTime,\TimeFlag)** **Cond**, bool type logic expression **MaxTime**, timeout and optional parameter, in seconds, with int or double type **TimeFlag,** timeout flag, set to true if timeout occurs with optional parameter, using the bool type variable
Example	Example 1 WaitUntil (di2 == true); … It means waiting until the di2 signal value is true before executing the following statements. Example 2 WaitUntil (di2 == true,5); … It means waiting until the di2 signal value is true. If the waiting time is over 5s and the di2 signal is still false, the following statement is executed. Example 3 Bool flag = false; WaitUntil (di2 == true, 5,flag); … It means waiting until the di2 signal value is true. If the waiting time is over 5s and the di2 signal is still false, the flag is set to true, and the following statement is executed. If the di2 turns to true within 5s, the flag is set to false. The flag can be used for subsequent logic judgment.

15.4.9.4Break

Explanation	It is used to jump out of the current loop, and is used in the WHILE loop in the RL language. When the WHILE loop is executed to Break, regardless of WHILE's CONDITION, it will jump out from the WHILE loop directly.

Example	Example 1
	VAR int counter = 0;
	WHILE(1)
	    IF(counter == 5)
	        break;
	    Endif

	counter++;
	ENDWHILE
	The program will jump out of the WHILE loop when the counter is 5.

15.4.9.5IF…Else if…Else

Explanation	Conditional judgment command

Example	Example 1
	IF(condition1)
	//a
	Else if (condition2)
	//b
	Else if (condition3)
	//c
	Else
	//d
	Endif
	Execute logic a when condition1 is true, logic b when condition2 is true, and so on.

15.4.9.6Goto

Explanation	The Goto command allows the pointer to jump to the marked command

Example	Example 1
	int a = 0;
	int b = 9;
	Goto end;
	print(a);
	end:;
	print(b);
	Define two variables a and b, then use the print function to print two statements. Use the Goto statement to force a jump to the end marker position of the print b statement, at which point the print of a will not be executed.

15.4.9.7For

Explanation	It is used to define a loop control structure that executes a specified number of times.

Example	Example 1
	For(int i from 1 to 10)
	    Print(“i = %d\n”, i);
	endfor
	This program prints i 9 times from 1 to 10 by adding 1 each time in sequence.
	
	Example 2
	For(int i from 1 to 10 step 3)
	    Print(“i = %d\n”, i);
	Endfor
	This program prints i 3 times from 1 to 10 by adding 3 each time in sequence.
	
	Supplementary explanation:
	Continue and Break can be used to control the For flow. See the Continue and Break commands for details.

15.4.9.8Continue

Explanation	Exit this loop. Continue executing the commands from the beginning of the loop, but just end the loop without exiting from the loop body.

Example	Example 1
	VAR int count = 0
	WHILE(1)
	count++
	IF(count == 1)
	Continue
	Else
	Break;
	MoveAbsJ(j10， v500， fine， tool1);
	Endif
	ENDWHILE

	The code for MoveAbsJ will not be executed.

15.4.9.9Inzone

Explanation	It is used with SetDO or modbus, cclink, and other IO operations or commands; this command can ensure that the signal is triggered at a defined point position, instead of being triggered earlier by the lookahead pointer.
Example	MoveL p1 MoveL p2 Inzone SetDO(dox, true); print(123); EndInzone MoveL p3; Supplementary explanation: In the example, an Inzone command is used. After the interpreter looks ahead to Inzone, instead of executing this command immediately, it generates an additional function which includes SetDo and print commands. This additional function takes effect when the motion command move p2 is completed. 1. If there is a turning zone between the two motion commands p2 and p3, the additional function will be executed at the moment when the robot reaches the turning zone 2. If there is no turning zone, the additional function will be executed at the moment the robot reaches p2

15.4.9.10While

Explanation	While loop allows you to write a loop control structure that keeps executing before conditions are met.

Example	Example 1
	int count = 0;
	while(count < 10)
	count++;
	print(count);
	endwhile
	This program enables a loop that counts by 1 from 0 to 10 and prints.
	Supplementary explanation: Continue and Break can be used to control the While flow. See the Continue and Break commands for details.

15.4.9.11Pause

Explanation	It is used to pause the program. The program enters the pause state after the previous statement of the pause statement is executed. The program must be resumed by clicking on the teach pendant or running the signal through an external program.
Attention	This command does not support auxiliary programming for the moment

15.4.9.12try/catch

Explanation	The try-catch command is an error handling mechanism in RL language. If an error occurs between the try and the catch commands, the program will convert the execution error into an error message set "e" and continue running from the catch-end try code block

Definition	**try**
	// do something
	catch(error e)
	print(e);
	endtry
	
	For example, reading data from a network link is a command that is likely to fail, but at this point, the user does not want the robot to stop. Instead, the user can use try-catch to capture errors and process them through RL programming
	Description
	error type description: error is a structure consisting of four parameters, namely
	● file: string (error occurred in file name)
	● line: int (error line)
	● num: int (error code)
	● reason: string (error reason)
	The error structure can be printed directly through the print command.
	// error data
	...

	catch(error e)
	print(e.line);
	print(e.num);
	print(e.reason);
	print(e);
	endtry

Example	Example 1
	ReadOnce:;
	Try
	Double xyz[3] = ReadDouble(3, timeout, socketname);
	Robtarget_0.trans.x = xyz[1];
	Robtarget_0.trans.y = xyz[2];
	Robtarget_0.trans.z = xyz[3];
	
	MoveL (Robtarget_0, v2000, fine, tool0);
	Catch(error e)
	SendString("Recv rob xyz error", socketname);
	Goto ReadOnce;
	endtry
	
	This program supports a simple application scenario. The communication command ReadDouble is used to read a three-dimensional array from the TcpSocket as the xyz parameter of the motion point position, and then the MoveL command is called to move to the corresponding Cartesian point.
	If the try/catch command is not used and the point position received from the TcpSocket is wrong, the robot will report "out of range" or "planning error" and stop the program.
	If the try/catch command is used, the motion command error is still reported, but the program does not stop. Instead, it jumps to the code segment between catch and endtry and handles the error as desired by the user. In this example, SendString tells Socket the point position error received by the host, and the host decides how to handle the error and calls the goto command to re-execute ReadDouble and wait for the next position.
	
	Example 2
	re_read:;
	try
	opendev("conn_name");
	string_res = readstring("conn_name");
	catch (error e)
	if (e.num == xxx)
	// A certain manageable error does not pause
	goto re_read;
	else
	print(e);
	Pause;
	endif
	endtry

Attention	Note: Force control commands cannot trigger try-catch

The error types and standard error codes that try/catch can process:

Classification	Error command	Explanation	error.num	error.reason
Default error		Commands without dedicated error codes	-1	Unknown error
Serial port related command		When the serial port does not exist	-1	Unknown error
Motion related command	MoveXX, Search, TrigL, etc. AccRamp, HomeSet, and other motion parameter settings	Tool and work object errors in motion coordinates Motion speed error Motion load error Beyond the motion range Planning error Encounter singularity, etc.	-1	Unknown error
Network command	OpenDev	Network link port error	-1	Unknown error
	All network commands	RL-operated connection for external communication	-1	Unknown error
Calculation and logic commands	CalcJoinT CalcRobt CRobT CJointT CLKSTOP	Internal error of controller	-1	Unknown error

	GOTO			
Peripheral control (Jodell series) (RM series)	JodellGripInit JodellSuckInit JodellSuckStatus RMRGMGripPosMove RMRGMGripTrqMove RMRGMGripStatus RMRGMResetErr RMCGripPosMove RMCGripTrqMove RMCGripStatus RMCResetErr RMRGMGripInit RMCGripInit	Peripheral communication abnormal	-1	Unknown error
Laser control	All laser commands	Laser welding OFF	-1	Unknown error
Stacking control	TrayUpdate TrayCount PalletUpdate PalletLayerCount PalletWobjCount SolarVisionExec	Data sending and receiving error with the host computer	-1	Unknown error
Register control	ReadRegByteByName	Data reading failed	-1	Unknown error
Axis 4 locking	SingAreaLockAxis4	Pose error, unable to activate the Axis 4 locking function	-1	Unknown error
Internal error of interpreter	Parameter type and quantity errors of most commands		-1	XXX parameter error
Internal error of interpreter			0	
Network command Serial port command	OpenDev	Connect to server failed	101	OpenDevConn failed
	OpenDev	Robot failed to start as server	102	OpenDevServer failed
	GetSocketConn	SocketConn not established	103	GetSocketConn failed, connection absent
	GetSocketConn	Obtaining the name of SocketConn structure is a server	104	GetSocketConn failed, the object is SocketServer
	GetSocketServer	GetSocketServer failed, server absent	105	GetSocketServer failed, server absent
	OpenDev	Error in connection enabled input parameter, there is no connection matched in the variable list	106	OpenDev failed, non-existent object is used
	SocketAccept	Input parameter is not a server name	107	SocketAccept (server) requires a server name
	GetSocketConn	Error in obtaining the name of SocketConn structure	108	GetSocketConn(conn), non-existent SocketConn
	GetSocketServer	Error in obtaining the name of SocketConn structure	109	GetSocketConn (server), non-existent SocketServer
	ReadBit	Command input parameter error	110	ReadBit must read integer multiples of 8
	ReadDouble	Command input parameter error	111	ReadDouble exceeds the preset range (0, 4096]
	ReadInt	Command input parameter error	112	ReadInt exceeds the preset range (0, 4096]
	ReadByte	Command input parameter error	113	ReadByte exceeds the preset range (0, 4096]
	ReadBit ReadDouble ReadInt ReadByte ReadString	Input time is too long	114	ReadXX time exceeds the preset range (0, 86400]
	ReadBit ReadDouble ReadInt ReadByte ReadString	Connection disconnected or data read error	115	Read failed
	ReadDouble	Beyond the limit time	116	ReadDouble timeout
	ReadInt	Beyond the limit time	117	ReadInt timeout
	ReadString	Beyond the limit time	118	ReadString timeout
	ReadBit	Beyond the limit time	119	ReadBit timeout
	ReadByte	Beyond the limit time	120	ReadByte timeout
	SendString	Command timeout or connection disconnected	121	SendString timeout or connection disconnected
	SendByte	Command timeout or connection disconnected	122	SendByte timeout or connection disconnected
Conveyor belt tracking	WaitObj	When executing the command, work object is out of the start window and cannot be tracked	123	Out StartWindow
	WaitObj	Waiting for tracking work object timeout	124	Out WaitTime
	WaitObj	Repeated tracking of work object	125	Connected Twice
	Possible occurrence	Tracking process exceeds the	126	Out MaxDistance

	after tracking enabled	working area and throws an exception		

15.4.9.13SwitchCase

Explanation	SwitchCase, like the IF command, controls the flow control based on the input variable conditions. RL interpreter will compare the variables in the Case field in order based on the input variable (condition). If the two variables are equal, the interpreter will enter the code branch of the corresponding Case and stop comparing and entering other code branches. If all conditions are not met, it will enter the Default branch; If no Case condition matches and there is no Default branch, it will enter no branch and the Switch command ends; Multiple conditions can be input for the Case command (see command structure Case C1, C12, C13 and example 1).
Definition	**Switch(condition)** **Case C1,C12,C13:** **Functions1()** **Case C2:** **Functions2()** **Default:** **DefaultFunction();** **EndSwitch**

Example	Example 1
	reg_int is a register variable, the host (PLC) will update the value of the variable through relevant register protocols (e.g. modbus, cclink). The production project expects the robot to execute the corresponding function branch (e.g. a blocked trajectory) according to the value of the register. If the register inputs 1, 2, and 3, then function A will be executed; if the register inputs 4, 5, and 6, then function B is executed. If the above conditions are not met, function C will be executed in the Default branch.
	Switch(reg_int)
	Case 1,2,3:
	FunctionsA();// The robot follows point positions related to function A
	Case 4,5,6:
	FunctionsB();// The robot follows point positions related to function B
	Default:
	FunctionC();// Execute function C if without specified input
	EndSwitch

15.4.10Home command

15.4.10.1Home

Explanation	It is used to make the robot return to the set Home through joint space motion
Definition	The command includes no input parameters
Example	Example 1 HomeSet(0,30,0,60,0,90,0); Home; Use the HomeSet command to set the Home and then the Home command to move the robot to the drag pose in the joint space.
Attention	Home pose setting must be enabled on the Robot Setup > Quick Turn interface or through the HomeSet command before the Home command can be used, otherwise, an error is reported.

15.4.10.2HomeSet

Explanation	It is used to set the robot's Home in the joint space
Definition	**HomeSet (axis1,axis2,axis3,axis4,axis5,axis6,axis7);** **Axisx**, data type: Double, to set the angle of home on each axis
Example	Example 1 HomeSet(0,30,0,60,0,90,0); Home; Use the HomeSet command to set the Home and then the Home command to move the robot to the drag pose in the joint space.

15.4.10.3HomeSetAt

Explanation	It is used to obtain the setup data of the robot's Home
Definition	**HomeSetAt(index);** Return value, data type: double, joint angle, in ° **Index**, data type: int, to get the joint angle of the specified axis at Home. When the index is 0, return if

	HomeSet is enabled, 1 means enabled, and 0 disabled.
Example	Example 1 HomeSet (0,30,0,60,0,90,0); double angle2 = HomeSetAt(2) angle2 Get the joint angle of joint 2 at 30°.

15.4.10.4HomeDef

Explanation	Determine if the Home is set
Definition	**HomeDef()** Return value, data type: bool, true: Home already set, false: Home not set

15.4.10.5HomeSpeed

Explanation	Set the running speed of Home command
Definition	**HomeSpeed Speed**
Example	Example 1 HomeSpeed(v1000); Home(); Set the Home speed to V1000. Then the Home command moves the robot to Home at the speed of V1000.

15.4.10.6HomeClr

Explanation	Clear Home setting
Example	Example 1 HomeClr(); Clear Home set in the program. After clearing, the Home command cannot be executed.

15.4.11Math command

15.4.11.1Sin

Explanation	sin() is used to calculate the sine of parameter x and return the result.
Definition	**double sin(double x)** **x** in radians; Return value: Return the calculated result between -1 and 1.

15.4.11.2Cos

Explanation	cos() is used to calculate the cosine of parameter x and return the result.
Definition	**double cos(double x)** **x** in radians; Return value: Return the calculated result between -1 and 1.

15.4.11.3Tan

Explanation	tan() is used to calculate the tangent of parameter x and return the result.
Definition	**double tan(double x)** **x** in radians; Return value: Return the tangent of parameter x.

15.4.11.4Cot

Explanation	cot() is used to calculate the cotangent of parameter x and return the result.
Definition	**double cot(double x)** **x** in radians; Return value: Return the cotangent of the parameter x.

15.4.11.5Asin

Explanation	asin() is used to calculate the arcsine of parameter x and return the result.
Definition	**double asin(double x)** **Parameter x** ranges from -1 to 1, beyond which error will be reported. Return value: Return the calculated result between -PI/2 and PI/2, in radians.

15.4.11.6Acos

Explanation	acos() is used to calculate the arccosine of parameter x and return the result.

Definition	**double acos(double x)** Parameter **x** ranges from -1 to 1, beyond which error will be reported; Return value: Return the calculated result between 0 and PI, in radians.

15.4.11.7Atan

Explanation	atan() is used to calculate the arctangent value of parameter x and return the result.
Definition	**double atan(double x)** Return value: Return the calculated result between -PI/2 and PI/2.

15.4.11.8Sinh

Explanation	tanh() is used to calculate the hyperbolic tangent of parameter x and return the result.
Definition	**double sinh(double x)** The mathematical definition is: (exp(x) - exp(-x))/2; Return value: Return the hyperbolic sine of parameter x.

15.4.11.9Cosh

Explanation	cosh() is used to calculate the hyperbolic cosine of parameter x and return the result.
Definition	**double cosh(double x)** The mathematical definition is: (exp(x)+exp(x))/2; Return value: Return the hyperbolic cosine of the parameter x.

15.4.11.10Tanh

Explanation	tanh() is used to calculate the hyperbolic tangent of parameter x and return the result.
Definition	**double tanh(double x)** The mathematical definition is: sinh(x)/cosh(x); Return value: Return the hyperbolic tangent of parameter x.

15.4.11.11Exp

Explanation	exp() is used to calculate e to the x power, which is the e^x value, and return the result;
Definition	**double exp(double x)** Return value: Return the result of e to the x power.

15.4.11.12Ln

Explanation	ln() is used to calculate the logarithm value of x at the base of e and return the result.
Definition	**double ln(double x)** Function description: Find the natural logarithm of x, ln(x), x > 0; Return value: Return the natural logarithm value of parameter x.

15.4.11.13log10

Explanation	log10() is used to calculate the logarithm value of x at the base of 10, and return the result.
Definition	**double log10(double x)** Where x>0; Return value: Return the natural logarithm value of parameter x at the base of 10.

15.4.11.14pow

Explanation	pow() is used to calculate x to the y power, which is the xy value, and return the result;
Definition	**double pow(double x, double y)** Return value: Return the result of x to the y power.

15.4.11.15sqrt

Explanation	sqrt() is used to calculate the square root of parameter x and return the result.
Definition	**double sqrt(double x)** The parameter x must be positive; Return value: Return the square root of parameter x.

15.4.11.16ceil

Explanation	ceil() will return the minimum integer value no less than parameter x, and the result will be returned in the double type.

Definition	**double ceil(double x)** Return value: Return a minimum integer value not less than the parameter x.

15.4.11.17floor

Explanation	floor() will return the maximum integer value not greater than the parameter x, and the result will be returned in the double type.
Definition	**double floor(double x)** Return value: Return the maximum integer value not greater than the parameter x.

15.4.11.18abs

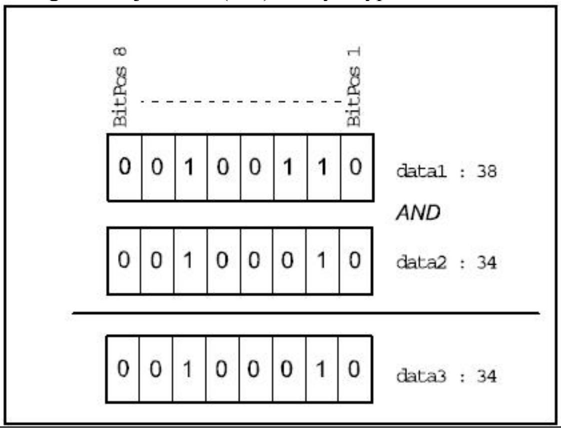
Explanation	Find the absolute value of x, |x|;
Definition	**int abs(int x)/double abs(double x)** Return value: When the input parameter is of int type, the output is also of int type. When the input parameter is of double type, the output is also.

15.4.11.19rand

Explanation	To generate an integer random number;
Definition	**rand()** Return value: An integer random number, ranging from 0 to 2147483647.

15.4.12Bit operation

15.4.12.1BitAnd

Explanation	BitAnd is used to generate logical conjunction (and) for byte type data. See table below: 
Definition	Return value, data type: byte, the result returned by performing logical conjunction of two byte-type data. **BitAnd (BitData1, BitData2);** **BitData1**, data type: byte, the byte data 1 to be processed. **BitData2**, data type: byte, the byte data 2 to be processed.
Example	Example 1 VAR byte data1 = 34; VAR byte data2 = 38; VAR byte byte3 = BitAnd(data1, data2); //34 Define the byte-type variable data1 and data2. assign them with the value of 34 and 38, respectively; perform logical conjunction on data1 and data2, the returned value of 34 is assigned to byte3.

15.4.12.2BitCheck

Explanation	It is used to check whether a bit in a byte-type data is 1. If so, returns true, otherwise, false.
Definition	Return value, data type: bool, true indicates the bit is assigned to 1, false indicates the bit is assigned to 0. **BitCheck (BitData, BitPos);** **BitData**, data type: byte, the byte data to be processed. **BitPos**, data type: int, the position of byte to be operated, ranging from 1 to 8.
Example	Example 1 VAR byte data1 = 130; VAR bool b1 = BitCheck(data1, 8) //true; Definite byte data1 and assign it with 130, check if the 8th bit of data1 is 1 and return true if so.

15.4.12.3BitClear

Explanation	To set a certain bit of byte- or int-type data to 0. The bit starts from 1.
Definition	**BitClear(BitData | IntData, BitPos);** **BitData**, data type: byte, the byte data to be processed. **IntData**, data type: byte, the byte data to be processed. **BitPos**, data type: int, the position of the bit to be operated, ranging from 1 to 8 for byte data and 1 to 32 for int data.
Example	Example 1 VAR byte data1 = 255; BitClear data1 1 //254; BitClear data1 2 //252; Define byte-type variable data1 and assign it with 255, Perform BitClear on data1, set the first bit to 0, and 254 is returned, set the second bit to 0, and 252 is returned.

15.4.12.4BitLSh

Explanation	It is used to perform logical left shift on byte-type data.
Definition	Return value, data type: byte, the byte data obtained by performing the left-shift operation. **BitLSh (BitData, ShiftSteps);** **BitData**, data type: byte, the byte data to be processed. **ShiftSteps**, data type: int, the bits selected for the left shift, ranging from 1 to 8.
Example	Example 1 VAR int left_shift = 3; VAR byte data1 = 38; VAR byte data2; data2 = BiLSh(data1, left_shift) //48; Define byte-type variable data1, and assign it with 38, perform 3 bits left shift on data1, and 48 is returned.

15.4.12.5BitNeg

Explanation	It is used to perform logical negation on byte-type data.
Definition	Return value, data type: byte, the byte data obtained by performing the logical negation. **BitNeg (BitData);** **BitData**, data type: byte, the byte data to be processed.
Example	Example 1 VAR byte data1 = 38; VAR byte data2; data2 = BitNeg(data1) //217; Define byte-type variable data1, and assign it with 38, perform logical negation on data1, and 217 is returned.

15.4.12.6BitOr

Explanation	It is used to perform logical disjunction (or) on byte-type data.
Definition	Return value, data type: byte, the byte data obtained by performing the logical disjunction. **BitOr (BitData1, BitData2);** **BitData1**, data type: byte, the byte data 1 to be processed. **BitData2**, data type: byte, the byte data 2 to be processed.
Example	Example 1 VAR byte data1 = 39; VAR byte data2 = 162; VAR byte data3; data3 = BitOr(data1, data2); //167 Define the byte-type variable data1 and data2, assign them with the value of 39 and 162, respectively; perform logical conjunction on data1 and data2, and 167 is returned.

15.4.12.7BitRSh

Explanation	It is used to perform the logical right shift on byte-type data.
Definition	Return value, data type: byte, the byte data obtained by performing the right-shift operation. **BitLSh (BitData, ShiftSteps);** **BitData**, data type: byte, the byte data to be processed. **ShiftSteps**, data type: int, the bits selected for the right shift, ranging from 1 to 8.
Example	Example 1 VAR int right_shift = 3; VAR byte data1 = 38; VAR byte data2;

	data2 = BiRSh(data1, right_shift); //4 Define byte-type variable data1, and assign it with 38, perform 3 bits right shift on data1, and 4 is returned.

15.4.12.8BitSet

Explanation	It is used to set a certain bit of byte- or int-type data to 1. The bit starts from 1.
Definition	**BitSet (BitData | IntData, BitPos);** **BitData**, data type: byte, the byte data to be processed. **IntData**, data type: byte, the byte data to be processed. **BitPos**, data type: int, the position of the bit to be operated, ranging from 1 to 8 for byte data and 1 to 32 for int data.

Example	Example 1
	VAR byte data1 = 0;
	BitSet (data1,1); //1
	BitSet (data1,2); //3
	Define byte-type variable data1 and assign it with 255, Perform BitSet on data1, set the first bit to 1, and 1 is returned, set the second bit to 1, and 3 is returned.

15.4.12.9BitXOr

Explanation	It is used to perform logical exclusive or on byte-type data.
Definition	Return value, data type: byte, the byte data obtained by performing the logical disjunction. **BitXOr (BitData1, BitData2);** **BitData1**, data type: byte, the byte data 1 to be processed. **BitData2**, data type: byte, the byte data 2 to be processed.
Example	Example 1 VAR byte data1 = 39; VAR byte data2 = 162; VAR byte data3; data3 = BitOr(data1, data2) ;//133 Define the byte-type variable data1 and data2, assign them with the value of 39 and 162, respectively; perform logical exclusive or on data1 and data2, and 133 is returned

15.4.13String operations

15.4.13.1StrFind

Explanation	It is used to find the position of a particular set of characters in the string from a specific location.
Definition	Return value, data type: int, the location of the first matching character. If the location is not found, the length of the returned string is added by 1. **StrFind (Str ChPos Set [\NotInSet]);** **Str**, data type: string, the string to be searched. **ChPos**, data type: int, the starting position, starting from 1, if the location is off the boundary, an error is reported. **Set**, data type: string, the character set to be matched. **[\NotInSet]**, identifier, to identify the character that cannot be matched in the character set.

Example	Example 1
	VAR int found;
	found = StrFind("Robotics", 1, "aeiou"); //2
	Match from the first character "R", and find the second character "o" in the character set "aeiou", return matching location 2.
	found = StrFind("Robotics", 1, "aeiou" \NotInSet); //1
	Match from the first character "R", and find the first character "R" is not in the character set "aeiou", return matching location 1.

15.4.13.2StrLen

Explanation	It is used to obtain the length of the string.
Definition	Return value, data type: int, the current string length, which is longer than or equal to 0. **StrLen (Str);** **Str**, data type: string, the string that requires the calculation of string length.
Example	Example 1 VAR int num; num = StrLen("Robotics"); //8 The length of the string "Robotics" is 8.

15.4.13.3StrMap

Explanation	It is used to back up a string, all characters in it are replaced according to the specified mapping relationship. The mapped characters correspond one by one according to their position, and the characters that are not mapped remain the same.
Definition	Return value, data type: string, the replaced string. **StrMap (Str, FromMap, ToMap);** **Str**, data type: string, the original string. **FromMap**, data type: string, the index of the mapping. **ToMap**, data type: string, the value of the mapping.

Example	Example 1
	VAR string str;
	str = StrMap(“Robotics”, “aeiou”, “AEIOU”) //RObOtIcs;
	Map the string "Robotics", and "aeiou" is respectively mapped to "AEIOU".
	
	Use restrictions: FromMap and ToMap have to match with each other and have to be of the same length.

15.4.13.4StrMatch

Explanation	It is used to search in a string, starting at the specified location, search for a particular format or a string, and return the matched location.
Definition	Return value, data type: int, the position of the first character of the matched string, and if there is no match, the string length plus one is returned. **StrMatch (Str, ChPos, Pattern);** **Str**, data type: string, the string to be searched. **ChPos**, data type: int, the starting position, and if the location exceeds the length range of the string, an error is reported. **Pattern**, data type: string, the format string to match.
Example	Example 1 VAR int found; Found = StrMatch(“Robotics”, 1, “bo”) //3; Search from the first character for "bo" and find a match at the third position, position 3 is returned.

15.4.13.5StrMemb

Explanation	It is used to check whether a character in a string belongs to a specified character set.
Definition	Return value, data type: bool, true indicates that the character in the string belongs to the specified character set. Otherwise, false is returned. **StrMemb (Str, ChPos, Set);** **Str**, data type: string, the string to be checked. **ChPos**, data type: int, the position of the character to be checked; if it exceeds the range of the string, an error is reported. **Set**, data type: string, the character set to be matched.

Example	Example 1
	VAR bool memb;
	memb = StrMemb(“Robotics”, 2, “aeiou”) //true;
	The second character o is a member of the character set "aeiou" and true is returned.

15.4.13.6StrOrder

Explanation	It is used to compare two strings and return the Boolean value.
Definition	Return value, data type: bool, when str1<=str2, return true; otherwise, false. **StrOrder (Str1, Str2);** **Str1**, data type: string, the first string value. **Str2**, data type: string, the second string value.

Example	Example 1
	VAR bool le;
	le = StrOrder(“FIRST”, “SECOND”); //true;
	le = StrOrder(“FIRSTB”, “FIRST”); //false

15.4.13.7StrPart

Explanation	It is used to truncate a part of a string to generate a new string.
Definition	Return value, data type: string, the truncated string, truncating a string from a specified location with a specified length. **StrPart (Str, ChPos, Len);** **Str**, data type: string, the original string of a truncated string. **ChPos**, data type: int, the starting position, and if it exceeds the range of the string, an error is reported.

	Len, data type: int, the length for truncating.

Example	Example 1
	VAR string part;
	part = StrPart(“Robotics”, 1, 5); //Robot
	Truncate the string for a length of 5 bits from position 1 to get "Robot".

15.4.13.8StrSplit

Explanation	It is used to split a string into an array of strings by specifying a separator
Definition	Return value, data type: string array, the array of strings obtained by splitting **StrSplit (Str [, separator]);** **Str**, data type: string, the original string to be split. **Separator**, data type: string, a separator. All characters in the string are considered as a separator and can be defaulted. If no separators exist, space can be considered as the default separator.
Example	string str_arr[4] = StrSplit("test1,test2;test3\test4", "\,;"); The string is split into four substrings (test1 test2 test3 test4). Use restrictions: ● An error is reported when the input string is blank. ● If the split results do not match the length of the defined string, an error is reported.

15.4.13.9StrToByte

Explanation	StrToByte can convert a string into byte type data
Definition	Return value, data type: byte, the conversion result of a string. **StrToByte (Str, [trans]);** **Str**, data type: string, the string to be converted. **Trans**, data type: enumeration, the mathematical binary format of the string. Available parameters include \Bin (binary), \Okt (octal), \Hex (hexadecimal), \Char (character), and the default (no parameter, decimal)
Example	Example 1 Byte NumBin = StrToByte(“10”, \Bin); Byte NumOkt = StrToByte(“10”, \Okt); Byte NumBin = StrToByte(“10”); Byte NumHex = StrToByte(“10”, \Hex); The string "10" is converted to byte numbers in binary, octal, decimal, and hexadecimal in order, and the results are 2, 8, 10 and 16. Example 2 Byte NumChar = StrToByte(“0”, \Char); The character "0" is converted to 48 according to the conversion relationship between characters and ASCII. Use restrictions: An error will be reported when the input string does not conform to the specified data format.

15.4.13.10StrToDouble

Explanation	StrToDouble can convert a string into double type data
Definition	Return value, data type: double, the conversion result of a string. **StrToDouble (Str);** **Str**, data type: string, the string to be converted.
Example	Example 1: Double NumDouble = StrToDouble(“3.1415926”); Convert string "3.1415926" into double type data. Use restrictions: An error will be reported when the input string does not conform to the specified data format.

15.4.13.11StrToInt

Explanation	StrToInt can convert a string into Int type data
Definition	Return value, data type: Int, the conversion result of a string. **StrToDouble (Str);** **Str**, data type: string, the string to be converted.
Example	Example 1 Int NumInt = StrToInt(“99”); Convert string "99" into Int type data.

	Use restrictions: An error will be reported when the input string does not conform to the specified data format.

15.4.13.12StrToDoubleArray

Explanation	StrToDoubleArray can convert a large number of strings double into the type of double array data
Definition	Return value, data type: Int, to determine whether the conversion is abnormal. -1: error, 0: normal. **StrToDoubleArray(output, input, spilit);** **Output**, data type: double array, the output of conversion results **Input**, data type: str string, the input of strings **Spilit**, data type: str string, a delimiter of Double strings
Example	Example 1 string tmp_ss = "1,2,3,4,5,6,7"; double db_arr[10]; StrToDoubleArray (db_arr, tmp_ss, ","); // result db_arr = {1,2,3,4,5,6,7,0,0,0} Use restrictions: The string allows for an extra terminator at the end In case of conversion failure or including illegal characters, it will report an error, that is, "The input string is not in Double form after segmentation" The data volume of double array should be greater than or equal to that of the data in the string; otherwise, it will report an error, that is, "The input array size is insufficient, or the array is not a one-dimensional array"

15.4.14Operators

15.4.14.1Basic operators

15.4.14.1.1Arithmetic operators

Arithmetic operators include:

Operators	Application
+	Plus
-	Minus
*	Multiply
/	Divide
%	Modular arithmetic
--	Decrement
++	Increment

Arithmetic operators support data types of bool, byte, int, and double, and if different types of variables are added, subtracted, multiplied, and divided, they will trigger implicit conversion.

The examples for arithmetic operators are as follows:
Example 1
VAR int a = 1;
VAR int b = 2;
VAR int c = -b;//Negate
VAR int ac = a * c; //Multiplication

Example 2
The two operators ++ and --, also known as unary operators, are operators that operate on an operand. RL does not distinguish between pre and post increment or decrement:
x = n++;   //Means to add n by 1 and assign the n value to x
x = --n;   //Means to subtract n by 1 and assign the new value to x

Example 3
Implicit conversion results of addition, subtraction, multiplication, and division of different types of variables:

Type 1	Type 2	Result
bool	bool	bool
bool	byte	byte

bool	int	int
bool	double	double
byte	byte	byte
byte	int	int
byte	double	double
int	int	int
int	double	double
double	double	double

15.4.14.1.2Logical operators

Logical operators support the operation of the basic data types, including

Operators	Application
&&	Logical conjunction
||	Logical disjunction
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to
!	Take logical negation

Logic and && expressions are true if the results on both sides are true, and the logic or || expression is true if one of the conditions of the two sides is true.

Example 1

The examples for other logical operators are as follows:

```
VAR int res = 1;
while(res < 3)        //Compare to determine whether res is less than 3
res++;
endwhile
di5 = !di6; //Take logical negation
VAR int counter = 4;
while(di7&&di8)    //Calculate logical conjunction
if(counter == 5)        //Whether it equals to
        break;
    endif
endwhile
```

15.4.14.1.3Assignment operators

Assignment operators include:

Operators	Application
=	Assignment
+=	Addition assignment
-=	Subtraction assignment
*=	Multiplication assignment
/=	Division assignment
%=	Modulus assignment

The examples for assignment operators are as follows

```
VAR int num1 = 3;
VAR int num2 = 4;
num1 += num2;            //Equivalent to num1 = num1 + num2, then num1 = 7.
```

```
num1 -= num2;        //Equivalent to num1 = num1 – num2, then num1 = -1.
num1 *= num2;        //Equivalent to num1 = num1 * num2, then num1 = 12.
num1 /= num2;        //Equivalent to num1 = num1 / num2, then num1 = 0.
num1 %= num2;        //Equivalent to num1 = num1 % num2, then num1 = 3.
```

All assignment operations of variables support implicit conversion. When the data types on the left and right sides of the assignment operation are inconsistent, the interpreter will attempt to trigger an implicit conversion to enable the program to continue running. When the conversion fails, the program will report an error and stop.
Bool, Byte, Int, and Double can be converted to each other. IO and register variables are special forms of the above four variables, and if they are used for assignment operations, they can also trigger implicit conversions.
If the return value of the function belongs to the above four variables, it can also be used as the right value of the assignment operation for assignment calculation.

Example 1

```
int tmp_num = 10.5;    // 10
bool tmp_bool = 1;     // true
tmp_bool = 0;          // false
double tmp_d = 999;    // 999.0
```

Example 2

```
// Register variables can be directly used to modify ordinary variables
double tmp_num = register0;
// Register variables can be directly used for conditional judgment
WaitUntil(register0 == 10);
```

Example 3

```
int mem_ret = StrMemb(“Robotics”, 2, “aeiou”);
// The return value of StrMember is of type bool. If it is necessary to use an int type to receive the return value,
// The controller will not report an error but will perform an implicit conversion
// true -> 1 ，   false-> 0
```

15.4.14.1.4Other operators

Operators	Application
()	Parentheses
.	Dot operator

The examples for the operators are as follows:
Example 1

```
VAR int num = arr[1];        //Assign the first element of the array to num
VAR int num2 = (1+2)*3;      //Using parentheses can change the order of operations, the value of num2 here is 9
```

Example 2
Define a robtarget variable pt1

```
pt1.trans.x = 200;           // Change the x coordinate of the pt1 point to 200 using the "." operator
```

Use restrictions:
The "." operator does not support modifications to the A, B, C members of robtarget variables.

15.4.14.2Operation priority

Priority	Operators	Use form	Combination direction
1	( )	(Expression)/function name (formal parameter list)	
	.	Variable name.	
2	-	-Expression	From right to left
	++	++ Variable name/Variable name ++	

	--	--Variable name/Variable name --	
	!	!Expression	
3	/	Expression / Expression	From left to right
	*	Expression * Expression	
	%	Integer expression / Integer expression	
4	+	Expression + Expression	From left to right
	-	Expression - Expression	
5	>	Expression > Expression	From left to right
	>=	Expression >= Expression	
	<	Expression < Expression	
	<=	Expression <= Expression	
6	==	Expression == Expression	From left to right
	!=	Expression != Expression	
7	&&	Expression && Expression	From left to right
8	||	Expression || Expression	From left to right
9	=	Variable = Expression	From right to left
	/=	Variable /= Expression	
	*=	Variable *= Expression	
	%=	Variable % = Expression	
	+=	Variable += Expression	
	-=	Variable -= Expression	

15.4.15Clock commands

15.4.15.1ClkRead

Explanation	It is used to read the value of the clock.
Definition	Return value, data type: double, to return the time interval between the stop time of the clock or the current time and the start of the clock. The accuracy is 0.001s. **ClkRead (Clock);** **Clock**, data type: clock, name of the clock.
Example	Example 1 VAR clock clock1; ClkStart(clock1); ClkStop(clock1); VAR double interval=ClkRead(clock1); interval stores the time interval between start and stop of clock1.

15.4.15.2ClkReset

Explanation	It is used to reset a clock. ClkReset guarantees that the count is 0 before using a clock.
Definition	**ClkReset (Clock);** **Clock**, data type: clock, name of the clock.
Example	Example 1 VAR clock clock1; ClkReset (clock1); Reset clock1.

15.4.15.3ClkStart

Explanation	It is used to start a clock. When a clock starts, it will continue to count until the clock stops or the program resets. The clock will continue to operate after the program stops or the robot is powered off.

Definition	**ClkStart (Clock);** **Clock**, data type: clock, name of the clock.
Example	Example 1 VAR clock clock1; ClkStart (clock1); Declare clock1, and start clock1.

15.4.15.4ClkStop

Explanation	It is used to stop a clock. When the clock stops, it stops counting. After the clock stops, it can be read for the interval, restarted, or reset.
Definition	**ClkStop (Clock);** **Clock**, data type: clock, name of the clock.
Example	Example 1 VAR clock clock1; ClkStart (clock1); … ClkStop (clock1); Stop clock1.

15.4.16Advanced commands

15.4.16.1RelTool

Explanation	It is used to translate or rotate the spatial position in the tool frame as specified by the current command. Main difference from Offs: Offs is the offset relative to the work object frame, and RelTool is the offset relative to the tool frame.
Definition	Return value, data type: robtarget, to return the new pose after the offset. **RelTool(Point, XOffset, YOffset, ZOffset, Rx, Ry, Rz [, Tool, Wobj]);** **Point**, data type: robtarget, the point to be offset, or the initial point of the offset command. **XOffset**, data type: double, offset in the x-direction of the tool frame. **YOffset**, data type: double, offset in the y-direction of the tool frame. **ZOffset**, data type: double, offset in the z-direction of the tool frame. **Rx**, data type: double, the rotation angle around the x-axis of the tool frame. **Ry**, data type: double, the rotation angle around the y-axis of the tool frame. **Rz**, data type: double, the rotation angle around the y-axis of the tool frame. **Tool**, data type: tool, contain tool frame information describing the Point position. **Wobj**, data type: wobj, contain work object frame information describing the Point position.

Example	Example 1
	p2=RelTool(p1,100,0,30,20,0,0);
	Since no tool and work object is specified, tool0 and wobj0 are used by default. Offset point p1 by 100 mm in the x-direction, 0 mm in the y-direction, and 30 mm in the z-direction on the tool frame, and then rotate 20 degrees around the x-axis. Last assign the new target point position to p2.
	
	Example 2
	p2=(RelTool(p1,100,0,30,20,0,0), tool5, wobj6);
	Offset point p1 by 100 mm in the x-direction, 0 mm in the y-direction, and 30 mm in the z-direction on the tool5 tool frame, and then rotate 20 degrees around the x-axis. Last assign the new target point position to p2.
	
	Example 3
	MoveL (RelTool(p1, 100,0,30,20,0,0), v4000, fine, tool2, wobj4);
	RelTool is used along with the Move command. As no tool or work object frame is specified, the tool and wobj of the Move command will be used. Offset point p1 by 100 mm in the x-direction, 0 mm in the y-direction, and 30 mm in the z-direction on the tool2 tool frame, and then rotate 20 degrees around the x-axis.

Attention	Auxiliary programming is not supported for the optional parameters (Tool and Wobj) of this command. The 5-axis models of the xMate CR series may have many unreachable points when using the RelTool command due to the lack of one orientation DOF. It is necessary to determine the offset based on the characteristics of reachable orientations (e.g. translation of points is usually reachable where the flange remains parallel to the base).

15.4.16.2Offs

Explanation	The position offset function, which is used to offset a point in the work object frame specified in the current command by a distance and return the position value of a new point. The translation offset is represented by x, y, and z, and the orientation rotation offset is represented by Rx, Ry, and Rz.

Definition	Return value, data type: robtarget, the new pose after the offset. **Offs（Point， XOffset， YOffset， ZOffset [, Rx, Ry, Rz] ）** **Point**, data type: robtarget, the point to be offset, or the initial point of the offset command. **XOffset**, data type: double, offset in the x-direction of the work object frame. **YOffset**, data type: double, offset in the y-direction of the work object frame. **ZOffset**, data type: double, offset in the z-direction of the work object frame. **Rx**, data type: double, the rotation angle around the x-axis of the work object frame. **Ry**, data type: double, the rotation angle around the y-axis of the work object frame. **Rz**, data type: double, the rotation angle around the z-axis of the work object frame.
Example	Example 1 p11=Offs(p10,100,200,300); Have the point p10 offset 100 mm in the x-direction, offset 200 mm in the y-direction, offset 300 mm in the z-direction of the work object frame, and assign the position of the new target point to p11.
Attention	This command does not support auxiliary programming for the moment.

15.4.16.3ConfL

Explanation	After the command is enabled, the following two items are checked: 1. Check whether the actual joint angle (i.e., cf1−7) reached through the Cartesian path is largely different from the teaching of the target point. If the difference is too large, the log will display the prompt, and the user consider whether to reach the target point in such a way. 2. Check whether the actual joint configuration (i.e., cfx value) reached through the Cartesian path is consistent with the teaching of the target point, and if not, an error will be reported and the movement will stop.

Example	Example 1
	ConfL(on);;
	MoveL (p1, v1000 ….);
	//If the J6 angle at the P1 teaching position is -5°, the J6 angle may be 355° after the MoveL command is executed (the difference between the two is +-360°, that is, they are the same position). At this time, the log will prompt "The difference between the actual point of the turning angle and the set point is too large", but the movement will not stop.
	…
	ConfL(off);
	MoveL (p1, v1000 ….);
	//After the conf check of the Cartesian path is disabled, the robot can move to P1, but the cf1−7 and cfx values of confdata for the actual point and teaching point are not checked.

Attention	1. The ConfL command corresponds to Cartesian motion commands such as MoveL and MoveC, without impact on the MoveJ and MoveAbsj, or the conf setting of "Move to". 2. In the case of executing Cartesian motion commands such as linear motion and circular motion, the robot moves towards the target point with the pose most similar to that in the starting point. At this point, the angle of the target point will be automatically selected, so the actual turning angle of the robot at the target point will be different from the angle at the teaching position in some cases. When the command is enabled, a prompt is given for this situation.

15.4.16.4ConfJ

Explanation	The Cartesian frame corresponds to a set of conf parameters (cf1−7, cfx). The conf data corresponding to the Cartesian coordinate points manually changed or written by the user may be incorrect, which makes it impossible for the controller to resolve the path of the target point. But in some scenarios, the user cares only about the robot's TCP position rather than the orientation. In this case, ConfJ Off can be used to remove conf limitations of the point and the controller can try to calculate the inverse kinematics closest to the starting point of the path (the calculation may fail, resulting in a failure of the motion command). Users can read the introduction to confdata for details of conf.

Example	Example 1
	p1.trans.x = ….;
	MoveJ (p1, v1000 ….);
	//Only the frame is modified, not the confdata parameters. This command is likely to cause the execution to fail.
	…
	ConfJ(Off);
	MoveJ (p1, v1000 ….);
	//After the conf check is disabled, the robot can move to P1, but the orientation is uncertain.

Attention	The ConfJ command corresponds to MoveJ, without impact on the other motion commands, or the conf setting of "Move to".

15.4.16.5Conf

Explanation	Effect of unified ConfL and ConfJ on/off: Conf on: equivalent to ConfL on and ConfJ on Conf off: equivalent to ConfL off and ConfJ off
Example	Example 1 Conf(on); MoveL (p0, v1000 …); MoveJ (p1, v1000 ….); //Effect equivalent to the following commands: ConfL (on); ConfJ (on); MoveL (p0, v1000 …); MoveJ (p1, v1000 ….);

15.4.16.6VelSet

Explanation	The VelSet command allows for adjusting maximum motion speed for smoother motion when the robot is handling fragile objects. Instead of being constant, the maximum velocity of each joint keeps changing with load, body orientation, and other factors when the robot is moving. The VelSet command scales the maximum velocity capability curve for a specific task path, and the scaled maximum velocity capability curve is also a changing curve.
Definition	**VelSet (gain);** **gain**, data type**:** int, the maximum velocity capacity is specified in percentage, ranging from 1% to 100%, where 100% means the maximum acceleration. The robot reports an error when going over the limit.
Example	Example 1 VelSet (50); Set the maximum velocity capability to half of the robot's default maximum velocity.
Attention	1. The VelSet command only affects the motion commands of the corresponding RL project, instead of JOG, move to, rapid motion, and other non-project functions. 2. The VelSet function will interrupt the turning zone. Please do not insert VelSet commands between the motion commands that require a turning zone. 3. The difference between the VelSet command and the program running rate adjustment slide: the program running rate adjustment slide modifies the user's expected velocity, for example, motion command V4000, under 50% slide control, equals a user's expected velocity of V2000. But if the robot is at its limits, the actual maximum velocity of this motion command is only V1000, then the actual motion velocity of the robot does not change regardless of whether the velocity slide is at 50% or 100% because both V2000 and V4000 are above V1000. Changing the expected velocity during this range will not impact the actual execution velocity; on the contrary, VelSet 50 does not change the user's expected velocity but reduces the actual maximum velocity of the motion command by 50% during the motion planning process. Under the same motion command, the actual motion velocity of the robot will be cut to half from V1000 to V500. The user should identify the difference between these two functions. 4. Speed automatically reverts to the default (100%) during the following operations: ● RL program is reset manually (PP to Main) ● A new RL program is loaded

15.4.16.7AccSet

Explanation	The AccSet command allows for adjusting acceleration for smoother movement when the robot is handling fragile objects.
Definition	**AccSet (acc, ramp);** **Acc**, data type: int, the acceleration is specified as a percentage of the system preset value, ranging from 30% to 100%, where 100% means the maximum acceleration, beyond which the robot will stop and report an error. Ramp, data type: int, the Jerk is specified as a percentage of the system preset value, ranging from 30% to 100%, where 100% means the maximum jerk, beyond which the robot will report an error.
Example	Example 1 AccSet (50,50); Acceleration and jerk are set to half of the default.
Attention	Acceleration automatically reverts to the default (100%) during the following operations: ● RL program is reset manually (PP to Main) ● A new RL program is loaded

15.4.16.8MotionSup

Explanation	It is used to turn on and off Collision Detection.

Definition	**MotionSup(type [, level, event]);** **Type**, data type: keyword, On: turn on, Off: turn off **Level**, data type: int, the additional parameter for MotionSup On to modify the collision detection sensitivity percentage, range: [1,200] **Event**, data type: string, the additional parameter for MotionSup On to set the behavior after collisions ● "softstop" indicates it stops compliantly
Example	Example 1 MotionSup(On); //... other commands MotionSup(Off); After enabling collision detection, users can execute other commands and turn off collision detection by MotionSup Off after the completion of commands Example 2 MotionSup(On, 200, “softstop”); Users enable collision detection and set the detection sensitivity percentage to 200% and the behavior triggered to a compliant stop after detecting a collision.

15.4.16.9MotionSupPlus

Explanation	MotionSupPlus (Motion Supervision Plus) is used to adjust the robot's joint collision detection sensitivity in the RL program at any time.
Definition	**MotionSupPlus(x1,x2,x3,x4,x5,x6,x7);** **x1 to x7**, the collision detection sensitivity in % for joints 1-7, respectively.
Example	Example 1 MotionSupPlus(5,20,7,20,6,20,5); Indicates the sensitivity of the 7 joints to be 5, 20, 7, 20, 6, 20, 5, respectively. Note: For 6-axis robots, 7 parameters should be set too, where the first 6 parameters correspond to joints 1-6. This command is available for cobots and six-axis industrial robots, but not three- and four-axis industrial robots.

15.4.16.10MotionSupJointTrq

Explanation	MotionSupJointTrq (Motion Supervision Joint Torque) Motion supervision is used to adjust the driving torque limits of robot joints at any time in the RL program.
Definition	**MotionSupJointTrq(x1,x2,x3,x4,x5,x6,x7);** **x1 to x7 indicates the driving torque limits of joint 1 to joint 7 respectively, in N.m.**
Example	Example 1 MotionSupJointTrq (100,91,59,14,14,14,5); It indicates that the driving torque limits of 7 joints are 100 N.m, 91 N.m, 59 N.m, 14 N.m, 14 N.m, 14 N.m, and 5 N.m, respectively. Note: For 6-axis robots, 7 parameters should be set too, where the first 6 parameters correspond to joints 1-6. This command is available for cobots and six-axis industrial robots, but not three- and four-axis industrial robots.

15.4.16.11BreakLookAhead

Explanation	This command informs the control system to cancel the lookahead and force the cancellation of the turning zone between the previous motion command and the next motion command. The robot TCP will move to the target point position of the previous motion command and then move to the next point without the turning zone. The program pointer will also wait for the TCP to move to the target point position of the previous motion command before continuing the lookahead scan.
Definition	The command includes no parameters and no return value
Example	Example 1 MoveL(P1,v1000,z50,tool0); BreakLookAhead MoveL(P2,v1000,z50,tool0); MoveL(P3,v1000,z50,tool0); 1) The turning zone of point P1 is set to z50. Because of the BreakLookAhead command, the lookahead and the turning zone will be canceled, and the robot TCP will move exactly to point P1 and then to P2. There is no BreakLookAhead command between P2 and P3, so the robot will look ahead at P2 and pass the z50 turning zone before moving to P3. 2) The BreakLookAhead command has the same effect as the wait 0 command.

15.4.16.12GetRobotMaxLoad

Explanation	It is used to get the maximum load value of the current robot model.
Definition	**Ret = GetRobotMaxLoad();** **Ret**, return value, data type: int, maximum payload

Example	int maxload = GetRobotMaxLoad();
	print(maxload);
	With xMate 7 as an example, return 7.

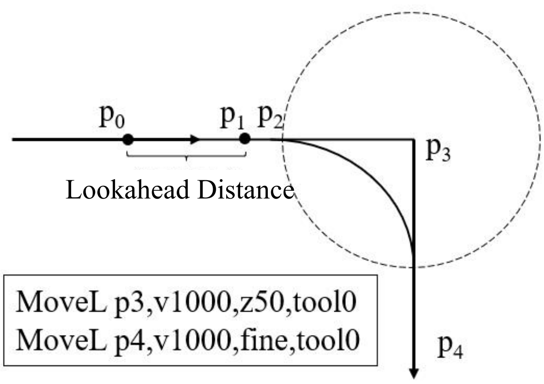
15.4.16.13GetRobotState

Explanation	It is used to get the current operating state of the control system. Use the 4-byte bit information to represent the state of the control system, including fault, emergency stop, safety gate, operation mode, servo mode, and motion state, as shown in following table.
Definition	**Ret = GetRobotState();** **Ret**, return value, data type: byte array, use four-byte types to represent the robot state.

Example	Example 1
	byte st[4] = GetRobotState();
	print(st);
	Return {0,5,0,0}. According to the table, the current state is: no fault, motor powered on, automatic mode, robot motion state, servo is in position mode.

S/N	State bits	Meaning
1	Byte[1].bit[1]	1: Control system is not authorized
2	Byte[1].bit[2]	1: Control system recoverable faults
3	Byte[1].bit[3]	1: Control system fatal error
4	Byte[1].bit[4]	1: Servo system failure
5	Byte[1].bit[5]	1: Servo system fatal failure
6	Byte[1].bit[6]	1: Emergency stop
7	Byte[1].bit[7]	1: Safety gate stop
8	Byte[1].bit[8]	Reserved
9	Byte[2].bit[1]	Power-on state, 0: motor is not powered on; 1: motor is powered on
10	Byte[2].bit[2]	Robot motion state, 0: idle; 1: in motion
11	Byte[2].bit[3]	Operation mode, 0: manual mode; 1: automatic mode
12	Byte[2].bit[4]	Servo mode, 0: position mode; 1: torque mode
13	Byte[2].bit[5]	Reserved
14	Byte[2].bit[6]	Reserved
15	Byte[2].bit[7]	Reserved
16	Byte[2].bit[8]	Reserved
17	Byte[3]	Reserved
18	Byte[4]	Reserved

15.4.16.14AutoIgnoreZone

Explanation	It is used to specify whether to allow the control system to automatically ignore the turning zone.
Definition	**AutoIgnoreZone (true/false);** **true**: Allow the control system to automatically ignore the turning zone (This is also the default state of the control system); **false**: Do not allow the control system to automatically ignore the turning zone  As shown above: The robot runs two MoveL commands with a z50 turning zone in between. During the motion, the robot needs lookahead from its current position for smooth and safe motion. For example, when the robot moves to p0, it looks ahead to P1. In this process, the control system pre-processes the

	information between the two points.
	As the robot moves forward, the lookahead end point also moves forward. At a certain point, the lookahead end point p1 coincides with p2, the start point of the turning zone. If the control system has received the second motion command, it can generate a turning zone properly and control the robot to move along the predetermined trajectory; if the control system fails to receive the second motion command, it cannot generate the turning zone, and it will process the turning zone according to the AutoIgnoreZone command status. See below for the logic:
	AutoIgnoreZone true: Instead of waiting for the second motion command, the control system will cancel the turning zone and control the robot to move directly toward P3.
	AutoIgnoreZone false: The control system will wait for the second motion command, during which the robot will slow down until the turning zone trajectory is generated. If the robot fails to receive the second motion command when reaching P2, the robot will stop moving and report an error through HMI.
	The failure of the robot to receive the second motion command timely is often a result of too many non-motion commands between two motion commands, e.g.:
	MoveL p3,v1000,z50,tool0
	For(int i from 1 to 10000)
	    printf("i = %d\n", i)
	endfor
	MoveL p4,v1000,fine,tool0
	Many print commands are added between two motion commands, and it takes a long time for the control system to receive the second motion command after the first one is processed.

Example	Example 1 AutoIgnoreZone(true); MoveL(p3,v1000,z50,tool0); MoveL(p4,v1000,fine,tool0); Allow the control system to automatically ignore the turning zone Example 2 AutoIgnoreZone(false); MoveL(p3,v1000,z50,tool0); MoveL(p4,v1000,fine,tool0); Do not allow the control system to automatically ignore the turning zone

15.4.16.15MotionWaitAtFinePoint true/false

Explanation	When the robot is stationary and the user clicks Start, the control system will look ahead a certain distance according to the lookahead parameter before starting the robot. This command sets whether the robot starts moving immediately when the lookahead coincides with a fine point. Fine point: the target point without a turning zone, i.e. a target point with the turning zone parameter set to fine.
Definition	**MotionWaitAtFinePoint(true/false);** MotionWaitAtFinePoint true: The control system controls the start of the robot strictly according to the lookahead parameters. The robot only starts to move when the lookahead distance reaches the set value of the lookahead parameter or the lookahead of all motion commands is completed. In this state, the control system can guarantee the set lookahead distance. MotionWaitAtFinePoint false: The control system does not strictly follow the lookahead parameters, and the robot starts moving immediately when the lookahead coincides with the fine point. In this state, the robot can still start smoothly when the program logic gets extremely complicated, but the lookahead distance cannot be guaranteed. Default: MotionWaitAtFinePoint false
Example	Example 1 MotionWaitAtFinePoint(true); MoveL(p1,v1000,fine,tool0); MoveL(p2,v1000,fine,tool0); MoveL(p13v1000,fine,tool0); MoveL(p1,41000,fine,tool0); MoveL(p1,51000,fine,tool0); When the control system looks ahead to p1, it does not start the robot immediately, but checks whether the current lookahead distance has reached the set length before deciding whether to start the robot. Example 2 MotionWaitAtFinePoint(false); MoveL(p1,v1000,fine,tool0); MoveL(p2,v1000,fine,tool0);

	MoveL (p3,v1000,fine,tool0); MoveL (p4,v1000,fine,tool0); MoveL (p5,v1000,fine,tool0); When the control system looks ahead to p1, it immediately starts the robot, instead of checking whether the current lookahead distance has reached the set length.

15.4.16.16IgnoreOverride

Explanation	In scenarios where welding, gluing, and other processes have strict requirements for the motion speed along the path, the command is developed with the hope of the motion speed of the process section not being affected by the global speed. This command can temporarily block the influence of the rate slider on motion commands, so that a specific segment of motion commands and trajectories is not affected by the global speed. This command supports performance in both automatic and manual modes. In manual mode, the motion speed is limited by v250. If the speed exceeds v250, it moves at v250. In automatic mode, it moves at the desired speed.
Definition	**IgnoreOverride(On/Off);** IgnoreOverride On indicates that subsequent motion commands are not affected by the rate slider, and IgnoreOverride Off has the opposite effect.

Example	Example 1
	MoveJ (p1, v1000 ….);//Affected by slider speed
	IgnoreOverride(On);
	MoveJ (p2, v1000 ….);//Unaffected by slider speed
	MoveJ (p3, v1000 ….);//Unaffected by slider speed
	IgnoreOverride(Off);
	MoveJ (p4, v1000 ….);//Affected by slider speed
	
	Example 2 (manual mode)
	IgnoreOverride(On);
	MoveJ (p2, v1000 ….);//Unaffected by slider speed and moving at the speed of v250
	MoveJ (p3, v100 ….);//Unaffected by slider speed and moving at the speed of v100
	IgnoreOverride(Off);
	
	Example 3 (automatic mode)
	IgnoreOverride(On);
	MoveJ (p2, v1000 ….);//Unaffected by slider speed and moving at the speed of v1000
	MoveJ (p3, v100 ….);//Unaffected by slider speed and moving at the speed of v100
	IgnoreOverride(Off);

Attention	Affected motion commands: MoveAbsJ, MoveJ, MoveL, MoveC, MoveCF, MoveT, SearchL, SearchC, TrigL, TrigC, and TrigJ. The command is not immediately executed and does not interrupt the turning zone. It can only be used in motion tasks and cannot be used in Inzone; otherwise, an error will be reported.

15.4.16.17SingAreaLockAxis4

Explanation	This command indicates the use of locking the 4-axis method to avoid robot wrist singularities.
Definition	**SingAreaLockAxis4(on/off);** SingAreaLockAxis4 on indicates the enabling of the 4-axis locking to avoid wrist singularity function. It should be noted that this function can only be enabled when the 4-axis of the robot is at 0° or ±180°. It is necessary to ensure that the 4-axis is at the target point of the previous motion command of SingAreaLockAxis4 on or that the 4-axis of the robot is already at the above angle to ensure normal program operation, otherwise, an error will be reported. **Off**, the function is turned off. Note: The Cartesian motion command between SingAreaLockAxis4 on and SingAreaLockAxis4 off adopts a special interpolation method for its pose, without changing the motion angle of the 4-axis. Any motion command attempting to change the 4-axis angle will cause an error. At the same time, this command is designed as a blocking command, which will interrupt the turning zone between the front and rear motion commands of SingAreaLockAxis4. The current version is applicable to industrial standard six-axis series (XB, NB models) and collaborative xMateCR, xMateSR series (excluding 5-axis models). This command is not supported when full DH compensation is enabled.
Example	Example 1 MoveAbsJ(p1,v1000,z50,tool0); SingAreaLockAxis4(on); MoveL(p2,v1000, z50,tool0); MoveL(p3,v1000, z50,tool0);

	SingAreaLockAxis4(off); MoveL(p5,v1000, z50,tool0); Point position p1 needs to ensure that the 4-axis angle is 0° or ±180°. When running to SingAreaLockAxis4, the 4-axis locking for wrist singularity avoidance function is enabled. The MoveL p2 and MoveL p3 will adopt the special interpolation method for their poses to maintain the 4-axis angle unchanged. SingAreaLockAxis4 off indicates to close the function.

15.4.16.18SpeedRefresh

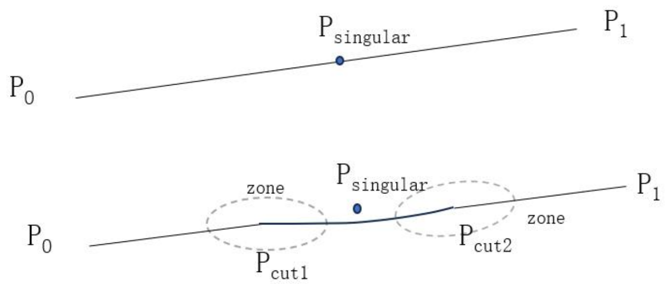
Explanation	It is used to override the speed value in the current motion program task.
Definition	**SpeedRefresh(override);** **override,** data type: int, value range: [1%, 100%], if the speed override value exceeds the limit range, the robot will report an error.
Example	Example 1 SpeedRefresh(70); It indicates the current speed override value is set to 70%
Attention	1. When executing pptomain, pptofunc, cursor movement, reloading the project, or entering/exiting demo mode, the speed value set through SpeedRefresh will be cleared and restored to the program running speed set on the teach pendant interface. 2. Priority note: If a speed value is set through SpeedRefresh, the speed specified by this command shall prevail; if the robot's operating speed is manually adjusted after the SpeedRefresh command is executed (e.g., by dragging the speed slider on the teach pendant), the manually adjusted speed shall prevail thereafter. 3. The SpeedRefresh command takes effect in turning zones. 4. The SpeedRefresh command can set a speed value that exceeds the maximum program speed limit in manual mode or the initial maximum program speed in automatic mode. Use this command with caution. (When running an RL program in Automatic mode, if pause is clicked and switched to manual mode, then the "Run" button is clicked to continue execution, the program running speed at this time will take the smaller value between the "program speed limit in manual mode" and the "speed value set through SpeedRefresh") 5. The speed override value set through SpeedRefresh will not be immediately completed, since there will be a certain time lag between issuing commands and the speed impact on the physical robotic arm.

15.4.16.19CSpeedOverride

Explanation	It indicates the current speed override value that users read
Definition	**Ret = CSpeedOverride();** **Ret** Data type: int The value range is 1%−100% of the speed override value

Example	Example 1
	int override = CSpeedOverride();
	print(override);
	If the current speed override value is 70%, it will return to 70

15.4.16.20SingAreaJointWay

Explanation	It indicates the use of joint space trajectory interpolation to avoid singularities in Cartesian commands. The Cartesian motion command between SingAreaJointWay on and SingAreaJointWay off will be automatically detected by the control system for any singularity. If the trajectory does not contain singularities, it will move in the same way as a normal trajectory. If it contains singularities, it will move in a unique pattern specific to the mode. See below for details:  As shown in the above figure, for the Cartesian trajectory P0P1 with singularities, the control system detects the singularity Psingular and adds two points, Pcut1 and Pcut2, around the singularity Psingular, to

	the original trajectory. The original trajectory is divided into three parts: P0Pcut1, Pcut1Pcut2, and Pcut2P1. Among them, P0Pcut1 and Pcut2P1 are still in the original Cartesian trajectory, but Pcut1Pcut2 uses a joint space trajectory (MoveAbsJ) instead of the original trajectory, so that it can traverse the singularity. The three trajectory segments are smoothly transitioned using a turning zone, and the turning radius of the turning zone can be set, which is the zone parameter in the command. The motion of robot near singularities usually involves a large range of joint angles, so when using this command, it is necessary to confirm whether the robot's motion trajectory meets the requirements. Note: The current version is applicable to industrial standard six-axis series (XB, NB models).
Definition	**SingAreaJointWay(on/off,zone);** **on/off,** to indicate that the joint interpolating singularity is enabled or disabled. **Zone**, to indicate the radius of the turning zone for the three trajectory segments P0Pcut1, Pcut1Pcut2, and Pcut2P1 after cutting, as shown in the above figure, referring to the definition of the turning zone.
Example	Example 1 MoveAbsJ(p1,v1000,z50,tool0); SingAreaJointWay(on,50); MoveL(p2,v1000, z50,tool0); MoveL(p3,v1000, z50,tool0); SingAreaJointWay(off); MoveL(p5,v1000, z50,tool0); In the above commands, SingAreaJointWay on,50 enables singularity avoidance and specifies an internal turning radius of 50 mm for singularity avoidance. SingAreaJointWay off disables singularity avoidance, and the motion commands in between will use the method of joint interpolating singularity avoidance for motion.

15.4.16.21SingAreaWrist

Explanation	This command indicates using sacrifice orientation to avoid singularities in Cartesian commands. The Cartesian motion commands between SingAreaWrist on and SingAreaWrist off both use sacrifice orientation to move. In this case, the robot tool follows the correct and precise trajectory motion, but the shape of the robot's wrist will be altered. When the singularity is not traversed, the above situation will also occur. The robot uses sacrifice orientation to move, and the wrist orientation of the robot may have a large range of motion. Therefore, when using this command, it is necessary to confirm whether the robot's motion trajectory meets the requirements. Note: The current version is applicable to industrial standard six-axis series (XB, NB models) and collaborative xMateCR, xMateSR series. This command is not supported when full DH compensation is enabled.
Definition	**SingAreaWrist(on/off, limit);** **on/off,** to indicate that the sacrifice orientation for singularity avoidance is enabled or disabled. **Limit**, the value in degrees that represents the maximum allowable sacrifice orientation.
Example	Example 1 MoveAbsJ (p1,v1000,z50,tool0); SingAreawrist (on,30); MoveL (p2,v1000, z50,tool0); MoveL (p3,v1000, z50,tool0); SingAreaWrist off; MoveL (p4,v1000, z50,tool0); In the above commands, SingAreaWrist on,30 enables singularity avoidance and specifies the maximum sacrifice orientation of 30 degrees for singularity avoidance. SingAreaWrist off disables singularity avoidance, and the motion commands in between will use the method of singularity avoidance for motion. Please note that the turning zone between p1 and p2 and between p2 and p3 can be generated normally, while the turning zone between p3 and p4 cannot be generated.
Attention	It can only be used for linear motion, not for curved motion. When using the sacrifice orientation for singularity avoidance function, the teaching point will be changed when it is within the singularity range. The wrist orientation during motion may sometimes differ from the taught orientation, not only at the teaching points where singularities are traversed, but also potentially at subsequent teaching points. The motion of the orientation near the singularity may differ between single-step motion and continuous motion.

15.4.16.22SetRobotJointsMaxAcc

Explanation	It is used to dynamically modify the maximum acceleration of the robot joint. When the robot needs to magnify the maximum acceleration of the joint to raise the takt and increase the robot's motion speed, the maximum acceleration of each joint can be set by this command. The effect of this command is consistent

	with the "maximum joint acceleration" under the motion parameters set on HMI, and this command is available for a specific motion.
Definition	**SetRobotJointsMaxAcc(jointval1, jointval2, jointval3, jointval4, jointval5, jointval6, jointval7);** **jointval1−7**, data type: double/int, maximum acceleration of the joint, unit: °/s^2.
Example	Example 1 SetRobotJointsMaxAcc (80000.0, 70000.0, 70000.0, 150000.0, 150000.0, 20000.0, 20000.0);
Attention	When there is a turning zone between the two motion commands, the turning zone is generated according to the constraints of the smaller one between the maximum joint accelerations.

15.4.16.23SetRobotJointsMaxJerk

Explanation	It is used to dynamically modify the maximum jerk of the robot joint. When the robot needs to magnify the maximum jerk of the joint to raise the takt and increase the robot's motion speed, the maximum jerk of each joint can be set by this command. The effect of this command is consistent with the "maximum joint jerk" under the motion parameters set on HMI, and this command is available for a specific motion.
Definition	**SetRobotJointsMaxJerk(jointval1, jointval2, jointval3, jointval4, jointval5, jointval6, jointval7);** **jointval1−7**, data type: double/int, maximum jerk of the joint, unit: °/s^3.
Example	Example 1 SetRobotJointsMaxJerk (30000.0, 27000.0, 27000.0, 50000.0, 40000.0, 60000.0, 60000.0);
Attention	When there is a turning zone between two trajectory segments, the turning zone is generated according to the constraints of the smaller one between the maximum joint accelerations;

15.4.16.24ResetJointKineLimit

Explanation	It is used to reset the maximum speed and maximum acceleration of the joint to the original value, with SetRobotJointsMaxAcc and SetRobotJointsMaxJerk. After it takes effect, the effect of the above commands will be cleared.
Definition	**ResetJointKineLimit();** No parameters.
Example	Example 1 ResetJointKineLimit();

15.4.16.25SetTransmissionOverloadParams

Explanation	It is used to dynamically modify the driving overload coefficient of the robot. When the robot needs to magnify the driving overload coefficient of the joint properly to raise the takt and increase the robot's motion speed, the driving overload coefficient of each joint can be set by this command. The effect of this command is consistent with the "overload coefficient" under the body parameters set on HMI, and this command is available for a specific motion.
Definition	**SetTransmissionOverloadParams (jointval1, jointval2, jointval3, jointval4, jointval5, jointval6, jointval7);** **jointval1−7**, data type: double/int, driving overload coefficient of the joint.
Example	Example 1 SetTransmissionOverloadParams (0.95, 0.95, 0.95, 0.95, 1.5, 0.95, 1.0);
Attention	When there is a turning zone between two trajectory segments, the turning zone is generated according to the constraints of the smaller one between the transmission overload coefficients;

15.4.16.26ResetTransmissionOverloadParams

Explanation	Used to restore the robot's transmission overload coefficient to its original value; to be used in conjunction with the **SetTransmissionOverloadParams command**; the effect of this command is cleared after taking effect;
Definition	**ReSetTransmissionOverloadParams();** No parameters.
Example	Example 1 ReSetTransmissionOverloadParams();

15.4.16.27SetAccRampTime

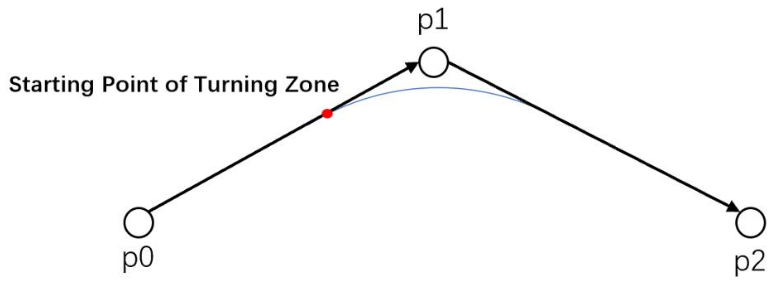
Explanation	It is used to set the acceleration ramp time, that is, the time it takes for the robot to increase its acceleration from a minimum to a maximum. The smaller the value, the faster the robot accelerates, and vice versa.
Definition	**SetAccRampTime (ramptime);** **ramptime**, data type: double/int, acceleration ramp time, range: [0.02, 0.5], unit: s.
Example	Example 1 SetAccRampTime (0.15);

15.4.16.28ResetAccRampTime

Explanation	It is used to reset the acceleration ramp time, with the SetAccRampTime.
Definition	**ResetAccRampTime ();** **No parameters;**
Example	Example 1 ResetAccRampTime();

15.4.16.29SetVarValue

Explanation	It is used to assign the variables in the turning zone, and the assignment is not triggered for lookahead.
Definition	SetVarValue(var1, var2); Assign var2 to var1. var1: data type: byte, int, double, bool, and writable register variable with no function code bound. var2: data type: byte, int, double, bool, IO variable, register variable, function return value, and expression.

Example	Example 1
	
	
	//The robot is currently located in the p0.
	MoveL(p1);
	SetVarValue(var1, var2); // Assign var2 to var1 at the start point of the turning zone
	MoveL(p2);

Attention	The implicit conversion occurs during the assignment.
	int a=1;
	double b=20.22;
	SetVarValue(a, b);
	For example, after the assignment, the variable a is 20.

15.4.16.30SetStopAccRampTime

Explanation	It is used to set the acceleration ramp time during the final stop phase. For target points without a turning zone, this is the time it takes for the robot to increase its acceleration from a minimum to 0 during the final stop phase. The smaller the value, the faster the robot stops, and vice versa. If the acceleration ramp time is greater than the acceleration ramp time during the final stop phase, the robot will use the longer acceleration ramp time for stopping.
Definition	**SetStopAccRampTime(ramptime);** **ramptime**, data type: double/int, acceleration ramp time during the final stop phase, range: [0.01, 1], unit: s.
Example	Example 1 SetAccRampTime(0.15);

15.4.16.31ResetStopAccRampTime

Explanation	It is used to reset the acceleration ramp time during the final stop phase, with the SetStopAccRampTime.
Definition	**ResetStopAccRampTime();** **No parameters;**
Example	Example 1 ResetStopAccRampTime();

15.4.16.32MotionSupJointTrq

Explanation	The motion supervision is used to adjust the maximum output torque of each joint in the RL program at any time.
Definition	**MotionSupJointTrq (J1,J2,J3,J4,J5,J6,J7);**
Example	Example 1 MotionSupJointTrq(50,50,70,50,60,60,50);

15.4.16.33PathRecStart

Explanation	Start recording the robot's path. It is a stop lookahead command. After executing PathRecStart to initiate path recording, the supported motion types for recording include MoveL, MoveJ, MoveAbsJ, MoveC, TrigL, TrigC, and TrigJ. During path recording, the execution of MoveC and TrigC commands cannot be interrupted; otherwise, the target points cannot be recorded. If MoveC or TrigC commands are interrupted, an error will be reported and execution will stop.
Definition	**PathRecStart();**
Example	MoveL(p1,v1000,z50,tool0,wobj0); PathRecStart(); MoveL(p2,v1000,z50,tool0,wobj0); When lookahead reaches PathRecStart, it stops lookahead and waits for the motion before PathRecStart to complete before executing PathRecStart. Motion commands after this command will be recorded.

15.4.16.34PathRecStop

Explanation	Stop recording the robot's path and clear the recorded path data. It is a stop lookahead command.
Definition	**PathRecStop();**
Example	MoveL(p1,v1000,z50,tool0,wobj0); PathRecStop(); MoveL(p2,v1000,z50,tool0,wobj0); When lookahead reaches PathRecStop, it stops lookahead and waits for the motion before PathRecStop to complete before executing PathRecStop. After this command, path recording will cease and the recorded path data will be cleared.

15.4.16.35PathRecBwd

Explanation	Make the robot move backward along the recorded path. It is a stop lookahead motion command, and the motion of PathRecBwd will not be recorded. If there are commands that interrupt lookahead between motion instructions, when executing the recorded path, the turning zone of the trajectory before the lookahead interrupting command will be set to z0. After executing PathRecStart to initiate path recording, if an interrupt is triggered, the PathRecBwd command in the interrupt must precede other motion commands; otherwise, an error will be reported and execution will stop. PathRecBwd cannot be interrupted. If an interrupt is triggered while executing PathRecBwd, an error will be reported and execution will stop.
Definition	**PathRecBwd();**
Example	MoveAbsJ(j1,v1000,z50,tool_weld); PathRecStart(); MoveL(p1,v1000,z50,tool0,wobj0); MoveAbsJ(j2,v1000,z50,tool_weld); PathRecBwd(); MoveL(p2,v1000,z50,tool0,wobj0); Two motion commands are recorded after PathRecStart. When executing PathRecBwd, the robot will perform path backtracking, moving from j2→p1→the robot's position when executing PathRecBwd (j1)

15.4.16.36PathRecFwd

Explanation	Make the robot move backward to the position where PathRecBwd was executed. It is a stop lookahead motion command, and the motion of PathRecFwd will not be recorded. If there are commands that interrupt lookahead between motion instructions, when executing the recorded path, the turning zone of the trajectory before the lookahead interrupting command will be set to z0. After executing PathRecStart to initiate path recording, if an interrupt is triggered, the PathRecFwd command in the interrupt must not be followed by any motion commands; otherwise, an error will be reported and execution will stop. PathRecFwd cannot be interrupted. If an interrupt is triggered while executing PathRecFwd, an error will be reported and execution will stop.
Definition	**PathRecFwd();**
Example	MoveAbsJ(j1,v1000,z50,tool_weld); PathRecStart(); MoveL(p1,v1000,z50,tool0,wobj0); MoveAbsJ(j2,v1000,z50,tool_weld); PathRecBwd(); PathRecFwd(); When executing PathRecBwd, the robot performs path backtracking [j2->p1->robot's position when executing PathRecBwd (j1)]. At this point the robot is at position j1. When executing PathRecFwd, the robot will move from j1->p1->j2 back to position j2.

15.4.16.37GetRecStartStatus

Explanation	Get whether path recording is enabled. The return value is of bool type
Definition	**GetRecStartStatus();**

Example	bool b1 = GetRecStartStatus(); Get whether path recording is enabled. true if enabled, false if disabled.

15.4.16.38SetMaxMotionJerk

Explanation	Set the value of the robot's underlying otgc jerk. The default value is 300. Increasing this value can achieve faster start/stop effects. The value will be restored to default after pptomain;
Definition	**SetMaxMotionJerk(Num);** **Num:** Data type: int/double, greater than 100
Example	Example 1 SetMaxMotionJerk(10000);

15.4.16.39VibSuppression

Explanation	Command to set the vibration suppression function switch. During Pptomain, the vibration suppression function reverts to the state set in the HMI interface (see the vibration suppression function in the Dynamic settings module in 9.8).
Definition	**VibSuppression(Type);** **Type**, data type: keyword, on: turn on, off: turn off
Example	VibSuppression(on); MoveAbsJ(p1,v500,fine,tool1); MoveAbsJ(p2,v500,fine,tool1); MoveAbsJ(p3,v500,fine,tool1); VibSuppression(off); MoveAbsJ(p4,v500,fine,tool1); When executing the VibSuppression(on) command, the controller enables the vibration suppression function. The movements at subsequent points p1−p3 will be affected by the vibration suppression function, where robot vibrations during start/stop are suppressed, resulting in improved trajectory accuracy. When executing the VibSuppression(off) command, the controller disables the vibration suppression function.

15.4.17Function commands

15.4.17.1CRobT

Explanation	It is used to get the robot pose. When using this function, you need to give the names of the tool and the work object. Return the pose of the specified tool frame, the current axis configuration information, and the external axis position. Note: When using CRobT, the robot should be in the stop state, i.e. the turning zone of the motion statement before CRobT should be set as fine.
Definition	Return value, data type: robtarget, return the current robot position, orientation, axis configuration data, and external axis information. **CRobT(Tool， Wobj);** **Tool**, data type: tool, the tool used when calculating the position. **Wobj**, data type: wobj, the work object used when calculating the position.
Example	Example 1 p2 = CRobT(tool1， wobj2);

15.4.17.2CJointT

Explanation	CJointT is used to read the current angle of the robot axes and external axes. Note: When using CJointT, the robot should be in the stop state, i.e. the turning zone of the motion command before CRobT should be set as fine.
Definition	Return value, data type: jointtarget, rotation axis unit: degree; linear axis unit: mm, to return the current angle value of the robot axes and the external axes. **CJointT();**
Example	Example 1 VAR jointtarget j2; j2 = CJointT();

15.4.17.3CalcJointT

Explanation	It is used to calculate the corresponding joint angle based on the specified robtarget variable
Definition	Return value, data type: jointtarget, to return the positions of joint angle and external axes corresponding to the input position. The joint angle is in Degrees (°), the external axis of the straight line is in millimeters (mm), and the rotation of the external axis is in Degrees. **CalcJointT(Rob_Target，Tool，Wobj);** Rob_Target, data type: robtarget, the specified Cartesian space target point. Please note that the tool and

	work object used in the definition of this point should be consistent with the tool/work object used in the CalcJointT command, otherwise, it may lead to results error. Tool, data type: tool, the tool to be used when calculating the joint angle. Note that it needs to be the same as the one used when defining the robtarget used. Wobj, data type: wobj, the work object to be used when calculating the joint angle. Note that it needs to be the same as the one used when defining the robtarget used.
Example	Example 1 jpos2 = CalcJointT(pt1, tool1,wobj2); Calculate the joint angle when tool1 reaches the pt1 and assign it to jpos2. pt1 is defined under wobj2.

15.4.17.4CalcRobt

Explanation	It is used to calculate the corresponding Cartesian space pose based on the specified joint angle.
Definition	Return value, data type: robtarget, to return the Cartesian space pose of a given joint angle. **CalcRobt(Joint_Target，Tool，Wobj);** **Joint_Target**, data type: jointtarget, the given joint angle for calculating Cartesian space pose. Tool, data type: tool, the tool used when calculating Cartesian space pose. Wobj, data type: wobj, the work object used when calculating the Cartesian space pose.
Example	Example 1 pt1 = CalcRobT(jpos1, tool2,wobj); Calculate the Cartesian pose based on jpos1 and assign it to pt1. pt1 is the pose described by the tool frame tool2 in the work object frame wobj1.

15.4.17.5Print

Explanation	It is used to print and output the user-defined content to the teach pendant, and the user can then use this function to debug the program. The input parameters of the Print function are special, the number of input parameters is unlimited, but there must be at least one, and each parameter must be a defined variable or a constant. The system converts these variables into strings and concatenates them in series, and finally outputs them to the debug window of the program editor. (When the debug window is closed, the information of the print is still recorded, with the maximum number of 500)
Definition	**Print(var1， var2，……);**

Example	Example 1
	counter = 0;
	while(true)
	        counter++;
	        Print(“counter = ”,counter);
	endwhile
	After the program is executed, the HMI's program debug window will print the following information:
	counter = 1
	counter = 2
	counter = 3
	counter = 4
	……
	
	Note: When you need to output a string, you can use double quotation marks "" to include the characters you want to display, but nested double quotation marks in double quotation marks are not supported.

15.4.17.6Print_f

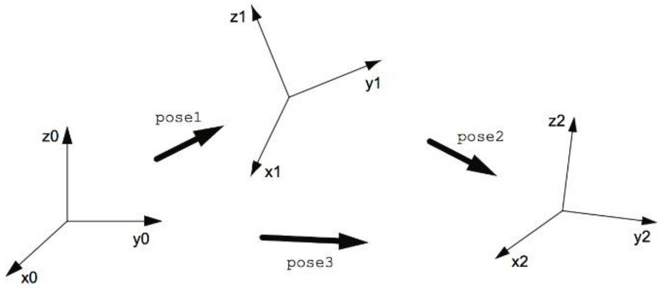
Explanation	The Print_f function is similar to the Print command, but it outputs the parameters to a specific log, and users can use this function to record key data and key events of the program. The backup function enables you to export and view the data. The input parameters of the Print_f function are unlimited, but there must be at least one, and each parameter must be a defined variable or a constant. The system converts these variables into strings and concatenates them in series, and finally outputs them to the log.
Definition	**Print_f(var1， var2，……);**

Example	Example 1
	counter = 0;
	while(true)
	        counter++;
	        Print_f(“counter = ”,counter);
	        //Other functions

endwhile
After the program is executed, the user log records the changes to the counter, which is available in the exported rl_log.

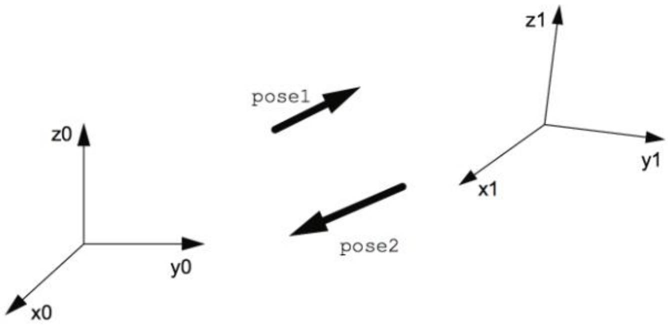
```
g3log: created log file at: Sat Nov 23 12:54:56 2024
        Running on machine: ubuntu
        LOG format : [level | mmdd hh:mm:ss.uuuuuu] msg

[RLINFO|11/23 12:55:02.632768 11058][xMateCR7][demo_step_back]task0:main.mod:MAIN[5]->counter = 1
[RLINFO|11/23 12:55:03.637821 11058][xMateCR7][demo_step_back]task0:main.mod:MAIN[5]->counter = 2
[RLINFO|11/23 12:55:04.645517 11058][xMateCR7][demo_step_back]task0:main.mod:MAIN[5]->counter = 3
[RLINFO|11/23 12:55:05.649153 11058][xMateCR7][demo_step_back]task0:main.mod:MAIN[5]->counter = 4
[RLINFO|11/23 12:55:06.661848 11058][xMateCR7][demo_step_back]task0:main.mod:MAIN[5]->counter = 5
[RLINFO|11/23 12:55:07.675333 11058][xMateCR7][demo_step_back]task0:main.mod:MAIN[5]->counter = 6
[RLINFO|11/23 12:55:08.677965 11058][xMateCR7][demo_step_back]task0:main.mod:MAIN[5]->counter = 7
[RLINFO|11/23 12:55:09.682166 11058][xMateCR7][demo_step_back]task0:main.mod:MAIN[5]->counter = 8
[RLINFO|11/23 12:55:10.686953 11058][xMateCR7][demo_step_back]task0:main.mod:MAIN[5]->counter = 9
[RLINFO|11/23 12:55:11.693370 11058][xMateCR7][demo_step_back]task0:main.mod:MAIN[5]->counter = 10
```
……

15.4.17.7PoseMult

Explanation	PoseMult is used to calculate the product of two pose changes
Definition	pose3 = PoseMult(pose1， pose2); **pose1**, data type, pose, pose 1. **pose2**, data type, pose, pose 2. **pose3**, return value, data type: pose, the result of pose product.
Example	 pose1 represents the pose of frame 1 relative to frame 0, and pose2 the pose of frame 2 relative to frame 1. Pose3, the pose of frame 2 relative to frame 0 can be calculated through the following method: VAR pose pose1; VAR pose pose2; VAR pose pose3; ... pose3 = PoseMult(pose1, pose2);

15.4.17.8PoseInv

Explanation	PoseInv is used to calculate the inversion of a pose change.
Definition	pose2 = PoseInv(pose1); **pose1**, data type, pose, input pose; **pose2**, data type, pose, return value.
Example	pose1 represents the pose of frame 1 relative to frame 0, and pose 2 the pose of frame 0 relative to frame 1. If pose1 is known, pose2 can be calculated through the following method: VAR pose pose1; VAR pose pose2; ... pose2 = PoseInv(pose1);

15.4.17.9GetRobABC

Explanation	Get the Euler angle orientation ABC of the Cartesian space point P; the rotation sequence: the initial frame (the work object frame selected in the motion command) first rotates around its own X axis, then around its Y axis, and last around its Z axis
Definition	**double db_arr[3] = GetRobABC(Point [, A, B, C]);** **Point**, data type: robtarget, the Cartesian point position used when calculating the position. **A, B**, C, data type: double, the return value of the Euler angle orientation for the Cartesian point position. **Return value**, data type: double-type three-dimensional array, the return value of the Euler angle orientation for the Cartesian point position.

Example	Example 1
	point0 is a Cartesian point position variable. To convert the Euler angle orientation of the variable to a Double variable of RL, use the following RL command
	VAR double Rob_A;
	VAR double Rob_B;
	VAR double Rob_C;
	// Assign the Euler angle of point0 to Rob_A|B|C
	GetRobABC(point0, Rob_A, Rob_B, Rob_C);
	
	Example 2
	point0 is a Cartesian point position variable. To generate an array of temporary variables to store the Euler angles of the Cartesian point position, use the following RL command
	double db_arr[3] = GetRobAbc(point0);

15.4.17.10SetRobABC

Explanation	Get the orientation of the Cartesian space point P based on the Euler angles ABC entered; the rotation sequence: the initial frame (the work object frame selected in the motion command) first rotates around its own X axis, then around its Y axis, and last around its Z axis.
Definition	**SetRobABC(Point , A, B, C);** **Point**, data type: Cartesian point position, the Cartesian point position whose orientation to be modified. **A, B, C**, data type: double, to set the Euler angle orientation for the Cartesian point position, in °.
Example	Example 1 point0 is a Cartesian point position variable. Rotate the point position around the fixed axis of X, Y, and Z to 30°, 60°, 90° respectively. SetRobABC(point0, 30, 60, 90);

15.4.17.11RotRobABC

Explanation	Rotate the Euler angles from the existing orientation of the Cartesian space point P based on the Euler angles ABC entered; the rotation sequence: the initial frame (orientation of the point P) first rotates around its own X axis, then around its Y axis, and last around its Z axis. The input angles ABC are added to the existing Euler angles.
Definition	**RotRobABC(Point , A, B, C);** **Point**, data type: Cartesian point position, the Cartesian point position whose orientation to be modified. **A, B, C**, data type: double, to set the Euler angle orientation for the Cartesian point position, in °.
Example	Example 1 point0 is a Cartesian point position variable. Rotate the point position around X, Y, and Z to 30°, 60°, 90°. RotRobABC(point0, 30, 60, 90);

15.4.17.12OpMode

Explanation	It is used to obtain the current operating mode of the robot

Definition	**ret = OpMode();** **ret**, return value, data type: int, 0: undefined, 1: automatic mode, and 2: manual mode.

Example	Example 1
	int mode = OpMode();
	print(mode);
	It returns to 1 if it is currently in automatic mode, and returns to 2 if in manual mode.

15.4.18Register commands

15.4.18.1ReadRegByName

Explanation	It is used to read the value of the corresponding register according to the register name
Definition	**ReadRegByName(RegData， Value);** **RegData**, data type: readable register variable, Setup -> Communication -> Register interface function, register variable. **Value**, data type: bool/int/double, the register data will be written into Value, and if the register variable type mismatches with the interpreter variable, the format will be converted automatically
Example	Example 1 int tmp_int; ReadRegByName(modbus_int_read[6], tmp_int); Read the data named modbus_int_read with subscript 6 into tmp_int variable

15.4.18.2WriteRegByName

Explanation	It is used to read the variable value of the corresponding register according to the name of the register. If the command is performed after the motion command, it will not interrupt the turning zone and be triggered at the end of the motion command trajectory or at the starting point of the turning zone. See Example 2 of SetDO for specific usage.
Definition	**WriteRegByName(RegData， Value);** **RegData**, data type: writable register variable, Setup -> Communication -> Register interface function, register variable. **Value**, data type: bool/int/double, the register data will be written into Value, and if the register variable type mismatches with the interpreter variable, the format will be converted automatically
Example	Example 1 WriteRegByName(modbus_int_write[6], 200); Write the data of INT 200 to the register corresponding to modbus_int_write[6].

15.4.18.3ReadRegByteByName

Explanation	It is used to read the value of the corresponding byte of the register according to its name
Definition	**ReadRegByteByName(RegData， Value， byteFlag);** **RegData**, data type: readable or writable int16/int32 type register, Communication -> Register interface function, register variable. **Value**, data type: byte type variable, the byte value corresponding to the register is read into Value which must be a variable of byte type. **byteFlag**, byte flag, with a value range of 1−4 where 1 is LSB and 4 is MSB.
Example	Example 1 byte tmp_value; ReadRegByteByName(modbus_reg, tmp_value,1); The first byte of the register named modbus_reg is read into the tmp_value variable

15.4.18.4WriteRegByteByName

Explanation	It is used to write the value of the corresponding register byte according to the name of the register. If the command is performed after the motion command, it will not interrupt the turning zone and be triggered at the end of the motion command trajectory or at the starting point of the turning zone. See Example 2 of SetDO for specific usage.
Definition	**WriteRegByteByName(RegData， Value， byteFlag);** **RegData**, data type: writable int16/int32 type register, Communication -> Register interface function, register variable. **Value**, data type: byte, the Value data is written into the corresponding byte of the register **byteFlag**, byte flag, with a value range of 1−4 where 1 is LSB and 4 is MSB.
Example	Example 1 WriteRegByteByName(modbus_reg, 200,2); 200 data is written into the second byte of modbus_reg.

15.4.19End-effector commands

15.4.19.1JodellGripInit

Explanation	Initialization command of Jodell electric gripper
Definition	**JodellGripInit (ID,wait_time);** **ID**, data type: Int variable, to establish communication and initialize Jodell electric gripper, parameter ID. **Wait_time**, data type: Int variable, to wait for the initialization to complete, wait time threshold, report error on timeout, in s.
Example	

15.4.19.2JodellGripMove

Explanation	Motion command of Jodell electric gripper
Definition	**JodellGripMove (ID,Pos,Vel,Trq);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Pos**, data type: Int variable, target position, unitless, range: 0−255. **Vel**, data type: Int variable, electric gripper velocity, unitless, range: 0−255. **Trq**, data type: Int variable, force detected by electric gripper operation, unitless, range 0−255.
Example	

15.4.19.3JodellGripStatus

<table>
<tr><td>Explanation</td><td>It is used to obtain the status of Jodell electric gripper</td></tr>
<tr><td>Definition</td><td>JodellGripStatus (ID,Pos,Vel,Trq,Contact);
ID, data type: Int variable, the gripper ID that obtains the movement status of the gripper.
Pos, data type: Int variable, to obtain the electric gripper’s current position, unitless, range: 0−255.
Vel, data type: Int variable, to obtain the electric gripper’s velocity, unitless, range: 0−255.
Trq, data type: Int variable, to obtain the electric gripper’s torque, unitless, range: 0−255.
Contact, data type: Int variable, to obtain the electric gripper’s state, unitless, range 0-255, where bit6-7 indicate whether the electric gripper detects an object.
<table>
<tr><th>Bit</th><th>Name</th><th>Value/Description</th></tr>
<tr><td>0</td><td>gAct</td><td>0: the electric gripper is being reset; 1: the electric gripper is in the enabling state</td></tr>
<tr><td>2</td><td>gMode</td><td>0: the parameter control mode; 1: the parameterless control mode</td></tr>
<tr><td>3</td><td>gGTO</td><td>0: stop; 1: moving to the target position</td></tr>
<tr><td>4-5</td><td>gSTA</td><td>0: the electric gripper is being reset or in the inspection state; 1: being activated; 2: not used; 3: activation completed</td></tr>
<tr><td>6-7</td><td>gOBJ</td><td>0: fingers are moving to the specified position; 1: fingers stop due to contact with an object when opening to reach the specified position; 2: fingers stop due to contact with an object when closing to reach the specified position; 3: fingers reach the specified position, but no object is detected.</td></tr>
</table></td></tr>
<tr><td>Example</td><td></td></tr>
</table>

15.4.19.4JodellSuckInit

Explanation	Initialization command of Jodell suction cup
Definition	**JodellSuckInit(ID);** **ID**, data type: Int variable, to initialize the suction cup of this ID and detect if the suction cup of this ID is connected correctly.
Example	

15.4.19.5JodellSuckSet

Explanation	The command for Jodell suction cup to operate. When this command is given, the suction cups immediately start operating according to the set parameters
Definition	**JodellSuckSet(ID,CH1_enable,CH1_VacMin, CH1_VacMax, CH1_Waittime, CH2_enable, CH2_VacMin, CH2_VacMax, CH2_Waittime);** **ID**, data type: Int variable, the ID of the suction cup being controlled. **CH1_enable**, data type: Int variable, whether the first channel of the suction cup is working or not. 1: working; 0: not working. **CH1_VacMin**, data type: Int variable, the minimum vacuum level of the first channel of the suction cup, range: 0−255. 0 means pure vacuum, and a value over 100 **means releasing the suction cup; stop pumping when the actual vacuum level is lower than this threshold;** CH1_VacMax, data type: Int variable, the maximum vacuum level of the first channel of the suction cup, range: 0−255. 0 means pure vacuum, and a value over 100 means releasing the suction cup; start pumping when the actual vacuum level is higher than this threshold;

	CH1_Waittime, data type: Double variable, timeout value of the first channel of the suction cup; **CH2_enable**, data type: Int variable, whether the second channel of the suction cup is working or not. 1: working; 0: not working. **CH2_VacMin**, data type: Int variable, the minimum vacuum level of the second channel of the suction cup, range 0−255. 0 means pure vacuum, and a value over 100 **means releasing the suction cup; stop pumping when the actual vacuum level is lower than this threshold;** CH2_VacMax, data type: Int variable, the maximum vacuum level of the second channel of the suction cup, range 0-255. 0 means pure vacuum, and a value over 100 means releasing the suction cup; start pumping when the actual vacuum level is higher than this threshold; **CH2_Waittime**, data type: Double variable, timeout value of the second channel of the suction cup;
Example	

15.4.19.6JodellSuckStatus

<table>
<tr><td>Explanation</td><td>It is used to obtain the status of Jodell suction cup</td></tr>
<tr><td>Definition</td><td>JodellSuckStatus(ID,Vac1,Contact1,Time_Err1,Vac2,Contact2,Time_Err2);
ID, data type: Int variable, the ID of the suction cup whose status is to be obtained.
Vac1, data type: Int variable, current vacuum level of the suction cup’s first channel obtained, range: 0−100.
Contact1, data type: Int variable, current status of the suction cup’s first channel obtained, range: 0−255, where bit6-7 indicates whether the an object is detected. See the table below for status details.
<table>
<tr><th>Bit</th><th>Name</th><th>Value/Description</th></tr>
<tr><td>0</td><td>gAct</td><td>0: the electric suction cup is not enabled; 1: the electric suction cup is enabled</td></tr>
<tr><td>2</td><td>gMode</td><td>0: the automatic control mode; 1: the advanced control mode</td></tr>
<tr><td>3</td><td>gGTO</td><td>0: adjustment stopped; 1: the pressure or vacuum is being adjusted</td></tr>
<tr><td>4-5</td><td>gSTA</td><td>0: the electric suction cup is not activated; 1 & 2: the electric suction cup is not used; 3: the electric suction cup is activated</td></tr>
<tr><td>6-7</td><td>gOBJ</td><td>0: below the minimum air pressure; 1: work object detected and minimum pressure value reached; 2: work object detected and maximum pressure value reached; 3: no object detected, object lost or detached.</td></tr>
</table>
Time_Err1, data type: Int variable, whether the suction cup's first channel obtained triggers a timeout alarm.
Vac2, data type: Int variable, current vacuum level of the suction cup’s second channel obtained, range: 0−100.
Contact2, data type: Int variable, current status of the suction cup’s second channel obtained, range: 0−255, where bit6-7 indicates whether an object is detected. See the table above for status details.
Time_Err2, data type: Int variable, whether the suction cup's second channel obtained triggers a timeout alarm.</td></tr>
<tr><td>Example</td><td></td></tr>
</table>

15.4.19.7RMRGMGripInit

Explanation	It is the command to initialize Robustmotion RM-RGM series electric grippers.
Definition	**RMRGMGripInit(ID);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper.
Example	

15.4.19.8RMCGripInit

Explanation	It is the command to initialize Robustmotion RM-C series electric grippers.
Definition	**RMCGripInit (ID);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper.
Example	

15.4.19.9RMRGMGripPosMove

Explanation	It is the Motion command for the position mode of the Robustmotion RMRGM series electric gripper.
Definition	**RMRGMGripPosMove(ID,Pos,Vel,Acc,PCheck);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Pos**, data type: Double variable, target position (mm) with a setting range of -2000.0−2000.0. **Vel**, data type: Double variable, running speed of electric grippers (mm/s) with a setting range of 0.01−1000.0. **Acc**, data type: Double variable, running acceleration of electric grippers (mm/s^2), with a setting range of 0.01−2000.0. **PCheck**, data type: Double variable, positioning range (mm), with a setting range of 0.01−10.0.

Example	

15.4.19.10RMCGripPosMove

Explanation	It is the Motion command for the position mode of the Robustmotion RMC series electric gripper
Definition	**RMCGripPosMove(ID,Pos,Vel,Acc,PCheck);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Pos**, data type: Double variable, target position (mm) with a setting range of -2000.0−2000.0. **Vel**, data type: Double variable, running speed of electric grippers (mm/s) with a setting range of 0.01−1000.0. **Acc**, data type: Double variable, running acceleration of electric grippers (mm/s^2), with a setting range of 0.01−2000.0. **PCheck**, data type: Double variable, positioning range (mm), with a setting range of 0.01−10.0.
Example	

15.4.19.11RMRGMGripTrqMove

Explanation	It is the Motion command for the torque mode of the Robustmotion RMRGM series electric gripper.
Definition	**RMRGMGripTrqMove(ID,Pos,Vel,Acc,Trq,PCheck,TCheck);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Pos**, data type: Double variable, target distance (mm) with a setting range of -2000.0−2000.0. **Vel**, data type: Double variable, running speed of electric grippers (mm/s) with a setting range of 0.01−1000.0. **Acc**, data type: Double variable, running acceleration of electric grippers (mm/s^2), with a setting range of 0.01−2000.0. **Trq**, data type: Double variable, positioning range (N.m.), with a setting range of 0.01−100.0. **PCheck**, data type: Double variable, positioning range (mm), with a setting range of 0.01−10.0. **TCheck**, data type: Double variable, time range (mm), with a setting range of 0.01−1000.0.
Example	

15.4.19.12RMCGripTrqMove

Explanation	It is the Motion command for the torque mode of the Robustmotion RMC series electric gripper.
Definition	**RMCGripTrqMove (ID,Pos,Vel,Acc,Trq,PCheck,TCheck);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Pos**, data type: Double variable, target distance (mm) with a setting range of -2000.0−2000.0. **Vel**, data type: Double variable, running speed of electric grippers (mm/s) with a setting range of 0.01−1000.0. **Acc**, data type: Double variable, running acceleration of electric grippers (mm/s^2), with a setting range of 0.01−2000.0. **Trq**, data type: Double variable, positioning range (N.m.), with a setting range of 0.01−100.0. **PCheck**, data type: Double variable, positioning range (mm), with a setting range of 0.01−10.0. **TCheck**, data type: Double variable, time range (mm), with a setting range of 0.01−1000.0.
Example	

15.4.19.13RMRGMGripStatus

Explanation	It is the acquisition state command of Robustmotion RM-RGM series electric grippers
Definition	**RMRGMGripStatus (ID,Pos,Vel,Trq,Reach,Err);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Pos**, data type: Double variable, electric gripper position, in mm. **Vel**, data type: Double variable, running speed of electric grippers, in mm/s. **Trq**, data type: Double variable, output torque of electric gripper (%), in %. **Reach**, data type: bool variable, to indicate whether the electric grippers are in place. **Err**, data type: Int variable, error code for electric grippers.
Example	

15.4.19.14RMCGripStatus

Explanation	It is the state acquisition command of Robustmotion RMC series electric grippers.
Definition	**RMCGripStatus (ID,Pos,Vel,Trq,Reach,Err);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Pos**, data type: Double variable, electric gripper position, in mm. **Vel**, data type: Double variable, running speed of electric grippers, in mm/s. **Trq**, data type: Double variable, output torque of electric gripper (%), in %. **Reach**, data type: bool variable, to indicate whether the electric grippers are in place.

	Err, data type: Int variable, error code for electric grippers.
Example	

15.4.19.15RMRGMResetErr

Explanation	It is the error command for electric grippers resetting of Robustmotion RM-RGM series.
Definition	**RMRGMResetErr (ID);** **ID**, data type: Int variable, the ID of the electric grippers that need to be reset.
Example	

15.4.19.16RMCResetErr

Explanation	It is the error command for electric grippers resetting of Robustmotion RMC series.
Definition	**RMCResetErr(ID);** **ID**, data type: Int variable, the ID of the electric grippers that need to be reset.
Example	

15.4.19.17RobotiqGripInit

Explanation	It is the command to initialize Robotiq 2F_85 series electric grippers.
Definition	**RobotiqGripInit(ID);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper.
Example	

15.4.19.18RobotiqGripGetStatus

Explanation	It is the command to get the status of Robotiq 2F_85 series electric grippers.
Definition	**RobotiqGripGetStatus(ID，Pos，gOBJ，Err);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Pos**, data type: Int variable, position of electric grippers, range: 0−255. **gOBJ**, data type: Int variable, contact status of electric grippers. 0 indicates the electric gripper is in motion without contacting any object; 1 indicates object contact occurred during the gripper opening process; 2 indicates object contact occurred during the gripper closing process; 3 indicates the electric gripper reached the specified position without contacting any object. **Err**, data type: Int variable, error code for electric grippers.
Example	

15.4.19.19RobotiqGripMove

Explanation	It is the command to move Robotiq 2F_85 series electric grippers.
Definition	**RobotiqGripMove(ID，Pos，Vel，Trq);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Pos**, data type: Int variable, position of electric grippers, range: 3−227. **Vel**, data type: Int variable, velocity of electric grippers, range: 0−255. **Trq**, data type: Int variable, output torque of electric grippers, range: 0−255.
Example	

15.4.19.20DhGripInit

Explanation	It is the command to initialize PGI-140-80 series electric grippers.
Definition	**DhGripInit(ID Time_wait);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Time_wait**, data type: double variable, unit: s, range: 0−10.
Example	

15.4.19.21DhGripGetStatus

Explanation	It is the command to get the status of DH PGI-140-80 series electric grippers.
Definition	**DhGripGetStatus(ID，Pos，gOBJ);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Pos**, data type: Int variable, position of electric grippers, range: 0−1000. **gOBJ**, data type: Int variable, contact status of electric grippers, 0: the electric gripper is moving; 1: the electric gripper arrives at the target position; 2: the electric gripper grasps the object; 3: the object falls; 4. the current ID is not initialized successfully.
Example	

15.4.19.22DhGripMove

Explanation	It is the command to move DH PGI-140-80 series electric grippers.
Definition	**DhGripMove(ID，Pos，Vel，Trq);** **ID**, data type: Int variable, the gripper ID that controls the movement of the gripper. **Pos**, data type: Int variable, position of electric grippers, range: 0−1000. **Vel**, data type: Int variable, velocity of electric grippers, range: 1−100. **Trq**, data type: Int variable, output torque of electric grippers, range: 20−100.
Example	

15.4.20Interrupt commands

15.4.20.1IRegister

Explanation	Register an interrupt, determining whether it can be triggered once, whether it can be debugged (triggered during single-step or single-step pause), as well as the interrupt number, trigger signal, and trigger type. . One trigger source can only be bound to one interrupt function.
Definition	**IRegister([\SINGLE,][\DEBUG,]int_num,** signal**,** trigger_type**, trap_function);** **[] indicates optional parameters** **int_num,** data type: interrupt number variable, used as an interrupt identifier signal**,** register signal/DI signal trigger_type**,** trigger method DI signal trigger methods: ● \Posflank: posedge triggering ● \Negflank: negedge triggering ● \Highlevel: high-level triggering ● \Lowlevel: low-level triggering Note: High/low level triggering will continuously trigger interrupts Register (int\bool\bit\byte) trigger methods: 0 represents low level, non-0 represents high level **trap_function:** interrupt function name **\SINGLE,** single trigger (optional) **\DEBUG,** debuggable (optional), adding this parameter allows interrupts to be triggered during single-step or single-step pause states
Example	IRegister \SINGLE,\DEBUG,intnum0,DI1_0,\Posflank,"trapfun1" Register an interrupt where when DI signal DI1_0 has a posedge change, execute trapfun1 function and stop responding to this interrupt after one trigger; this interrupt can be triggered during single-step and single-step pause states; intnum0 serves as an identifier for this interrupt, used for IEnable, IDisable, and GetTrapData commands

15.4.20.2IEnable

Explanation	Activate an interrupt disabled by IDisable, lookahead triggered (activated by default after IRegister)
Definition	**IEnable (int_num);** **int**_num, data type: interrupt number variable, used as an interrupt identifier
Example	IEnable(intnum0); Activate interrupt corresponding to intnum0, can be triggered by bound signal source after activation

15.4.20.3IDisable

Explanation	Disable an interrupt, lookahead triggered. Note that, for an interrupt configured with single-trigger mode, if it receives an interrupt signal after being disabled by IDisable, it will still be treated as completion of the single trigger
Definition	**IDisable(int_num);** **int**_num, data type: interrupt number variable, used as an interrupt identifier
Example	IDisable(intnum0); Disable the interrupt corresponding to intnum0; after being disabled, it cannot be triggered by the bound signal source

15.4.20.4GetTrapData

Explanation	Get information about an interrupt, lookahead triggered.
Definition	**GetTrapData (int_num,str);** **int**_num, data type: interrupt number variable, used as an interrupt identifier **Str,** data type: string receiving interrupt information
Example	GetTrapData (intnum0,string0); Get interrupt information corresponding to intnum0, store information in string0

16Appendix

16.1Details of user permission

Category	Function	Operator	Teacher	Programmer	Admin	System
Project related and program editing	Project configuration (create, import, export, load, restore, save as, set default project)	N	N	Y	Y	Y
	View project (including program and object data such as IO, variables, and predefined parameters)	Y	Y	Y	Y	Y
	Edit project (including object settings such as program editing and tools)	N	N	Y	Y	Y
	Teach point positions (including RL editor interface, variable list point type, point list)	N	Y	Y	Y	Y
Robot motion and program running	Switch between automatic/manual mode	Y	Y	Y	Y	Y
	Power on/off	Y	Y	Y	Y	Y
	Start/Stop program	Y	Y	Y	Y	Y
	Switch program loop mode	Y	Y	Y	Y	Y
	Adjust program running speed	N	Y	Y	Y	Y
	Single-step program debugging	N	Y	Y	Y	Y
	Jog/Drag the robot	N	Y	Y	Y	Y
Status monitoring	View runtime data	Y	Y	Y	Y	Y
	Set IO Signal	N	Y	Y	Y	Y
	Set register value	N	Y	Y	Y	Y
	Variable monitoring	N	Y	Y	Y	Y
Setting	Controller settings - basic settings	N	N	Y	Y	Y
	Controller settings - advanced settings	N	N	Y	Y	Y
	Controller settings - authorization settings	N	N	N	Y	Y
	HMI settings - basic settings	N	N	N	Y	Y
	HMI settings - Teach Pendant mode	N	N	Y	Y	Y
	User group	Y	Y	Y	Y	Y
	Zero calibration	N	N	Y	Y	Y
	Calibration of the base frame	N	N	Y	Y	Y
	Dynamic settings	N	N	Y	Y	Y
	Body parameters	N	N	Y	Y	Y
	Motion parameters	N	N	Y	Y	Y
	Force control parameters	N	N	Y	Y	Y
	Quick adjustment	N	Y	Y	Y	Y
	Electronic nameplate	N	N	Y	Y	Y
	Error code alarm filtering	N	N	Y	Y	Y
	Custom buttons	N	N	Y	Y	Y
Communication	System IO	N	N	Y	Y	Y
	External communication	N	N	Y	Y	Y
	IO device	N	N	Y	Y	Y
	Bus devices	N	N	Y	Y	Y
	Register	N	N	Y	Y	Y
	End-effector	N	N	Y	Y	Y
	RCI settings	N	N	Y	Y	Y
	xPanel configuration	N	N	Y	Y	Y
	Electric gripper and suction cup	N	N	Y	Y	Y
	Serial port settings	N	N	Y	Y	Y
	Encoder	N	N	Y	Y	Y
	OPC-UA	N	N	Y	Y	Y
Safety	Soft limit	N	N	Y	Y	Y
	Virtual wall	N	N	Y	Y	Y
	Collision detection	N	N	Y	Y	Y
	Safe region	N	Y	Y	Y	Y
	Safety monitor	N	N	Y	Y	Y
	Collaboration mode	N	N	Y	Y	Y
	Safety position	N	Y	Y	Y	Y
Process	Conveyor belt	N	N	Y	Y	Y

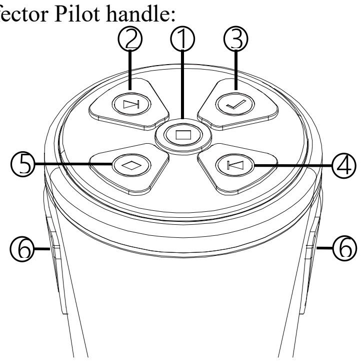
Package	Track	N	N	Y	Y	Y
Log	Log query	Y	Y	Y	Y	Y
	Diagnostic setting	N	N	N	N	Y
Options	Connect/About/Demo	Y	Y	Y	Y	Y
	Software upgrade - controller upgrade/other settings	N	N	N	Y	Y
	Software upgrade - HMI upgrade/controller backup	Y	Y	Y	Y	Y
	Export	Y	Y	Y	Y	Y
	Import	N	N	N	Y	Y
	File manager	N	N	N	Y	Y

16.2Introduction of collaborative robot's end-effector handle

16.2.1ER series

The xMate ER series robot end-effector integrates a Pilot handle with an intelligent interactive panel. In Drag Mode, the buttons on the Pilot handle can be used for quick point position teaching and continuous trajectory teaching, providing better human-machine interaction.

Definition of buttons on end-effector Pilot handle:

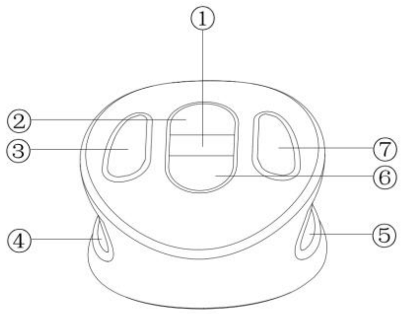


S/N	Definition
①	It is used to update the teaching point with the current pose, start/stop trajectory recording
②	Next
③	Add the midpoint/track to the list and confirm/cancel the pop-up prompt.
④	Previous
⑤	Delete the point position/trajectory in the list, cancel pop-up window prompts
⑥	In Drag Mode, press the two enabling buttons at the same time to activate the drag function

16.2.2CR series

The xMate CR series robot end-effector integrates an xPanel handle with an intelligent interactive panel. In Drag Mode, the buttons on the Pilot handle can be used for quick point position teaching and continuous trajectory teaching, providing better human-machine interaction.

Definition of buttons on end-effector xPanel handle:



S/N	Definition
①	Add the midpoint/track to the list and confirm the pop-up prompt
②	Move forward
③	Delete the point/track in the list and cancel the pop-up prompt
④	In Drag Mode, press the two enabling buttons at the same time to activate the drag function
⑤	

⑥	Move backward
⑦	Update the teach point with the current pose, confirm the pop-up prompt, and start/stop trajectory recording.

16.3Point position and path teaching (based on the collaborative robot's end-effector handle)

16.3.1Point position teaching

Turn on the drag enabling switch on the operation panel, and the robot is powered on automatically and enables Drag Mode. The following operations can be performed through Robot Assist and the robot end-effector drag handle:

Step	Explanation
1. Create/load a project and enter the Point List interface;	The end-effector buttons only respond when the current page of Robot Assist is Point List or Path List.
2. Press the two enabling buttons on the end-effector handle at the same time, drag the robot to any position, and release the drag enabling button. Press the Add Point button on the end-effector handle.	A new teaching point of the current pose is added to the end of the Point List, and the cursor is now at the new teaching point;
3. Press the Previous/Next button on the end-effector handle	Move the cursor to the previous/next point in the Point List and select the point;
4. Select a point to update in the Point List, drag the robot to another position, and release the drag enabling button. Press the Update Point button on the end-effector handle;	A pop-up window prompt will appear when you try to update a point position. If you press the OK button on the end-effector handle, the selected point position will be updated from the current pose. If you press the Cancel button on the end-effector handle, the pop-up window will be closed, and the selected point position will remain unchanged;
5. Select a point to delete in the Point List. Press the Delete Point button on the end-effector handle and confirm.	A pop-up window prompt will appear when you try to delete a path. If you press the OK button on the end-effector handle, the selected path will be deleted from the Path List. If you press the Cancel button on the end-effector handle, the pop-up window will be closed, and the selected path will remain on the Path List;

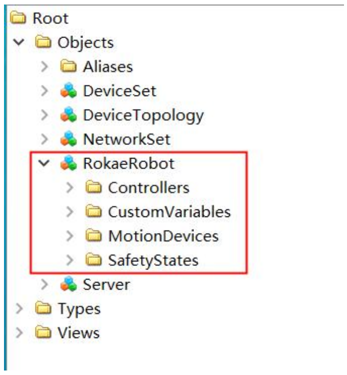
16.3.2Path teaching

Turn on the drag enabling switch on the operation panel, and the robot will be powered on automatically and enable the drag mode. Then, perform the following operations through Robot Assist and the end-effect drag handle:

Step	Explanation
1. Create/Load a project, move the robot to any start position, and enter the Path List interface.	The end-effector buttons only respond when the current page of Robot Assist is Path List or Path List.
2. Press the Add Path button on the end-effector handle;	A new path is added to the end of the Path List, and the cursor is now at the new path;
3. Press the Previous/Next button on the end-effector handle;	Move the cursor to the previous/next path in the Path List and select the path;
4. Select a path in the Path List to start recording. Press the Start Trajectory Recording button on the end-effector handle and press the two enabling buttons on the end-effector handle at the same time to drag the robot for trajectory recording.	The trajectory recording starts after the Start Trajectory Recording button is pressed. Press the Stop Trajectory Recording button to stop recording, and the trajectory is saved automatically.
5. Select a path to delete in the Path List. Press the Delete Path button on the end-effector handle and confirm.	A prompt window will pop up when you try to delete a path. If you press the OK button on the end-effector handle, the selected path will be deleted from the Path List. If you press the Cancel button on the end-effector handle, the prompt window will be closed, and the selected path will remain on the Path List.

16.4OPC-UA Robotics model

The OPC-UA server of the xCore controller system defaults to supporting all mandatory options specified in the standard of OPC 40010-1 OPC-UA for Robotics, Part 1: Vertical Integration. The top-level directory is as follows:

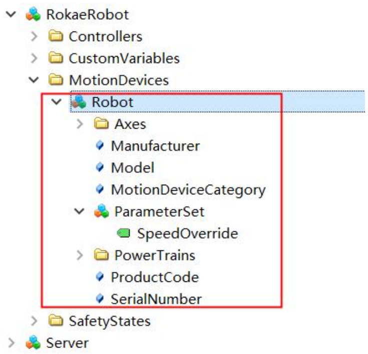


Type	Browse Name	Description
MotionDeviceSystem	RokaeRobot	Each server has an instance of MotionDeviceSystem type, named "RokaeRobot", placed under the Objects node;

The child nodes of MotionDeviceSystem are as follows:

Type	Browse Name	Description
MotionDevices	MotionDevices	A container that can accommodate MotionDeviceType instances
Controllers	Controllers	A container that can accommodate MotionDeviceType instances
SafetyStates	SafetyStates	A container that can accommodate ControllerType instances
CustomVariables	CustomVariables	A container that accommodates custom variables. This node is an extension of the Robotics standard model, and all user-defined variables can be found under this node

16.4.1MotionDevices model

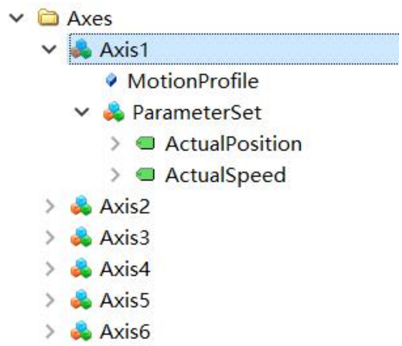


Type	Browse Name	Description
MotionDevice	Robot	The instance of MotionDeviceType, named "Robot"

The child nodes of MotionDevice are as follows:

Type	Browse Name	Description
MotionDeviceCategory	MotionDeviceCategory	The type of motion equipment specified in ISO 8373, which is set to ARTICULATED_ROBOT, that is, joint robot
Manufacturer	Manufacturer	Manufacturer
Model	Model	Robot model
ProductCode	ProductCode	Product number, currently not supported
SerialNumber	SerialNumber	Serial number, currently not supported
ParameterSet	ParameterSet/SpeedOverride	Program speed, RL program speed: 1−100%

16.4.1.1Axes child nodes

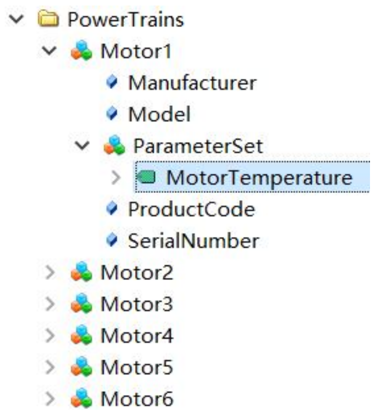


Type	Browse Name	Description
Axis	Axis%1	1. Each instance corresponds to an axis, and a 6-axis robot corresponds to 6 instances 2. Name according to axis number, such as Axis1, Axis2

The Axes child nodes are as follows:

Type	Browse Name	Description
MotionProfile	MotionProfile	Axis type, generally: 1 (ROTARY)
ParameterSet	ParameterSet/ActualPosition	Current axis position, in degrees
	ParameterSet/ActualSpeed	Current axis speed, in degrees/s

16.4.1.2PowerTrains child nodes

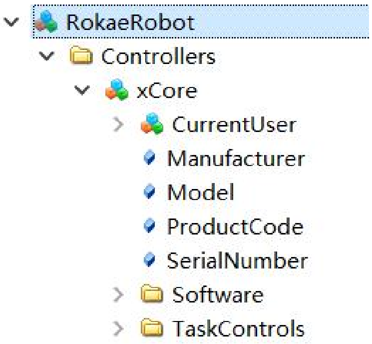


Type	Browse Name	Description
PowerTrain	Motor%1	1. Each instance corresponds to an axis power unit, and a 6-axis robot corresponds to 6 instances, including 6 MotorType instances 2. Name according to axis number, such as Motor1, Motor2

Motor child nodes are as follows:

Type	Browse Name	Description
Manufacturer	Manufacturer	Manufacturer, currently not supported
Model	Model	Robot model, currently not supported
ProductCode	ProductCode	Product number, currently not supported
SerialNumber	SerialNumber	Serial number, currently not supported
ParameterSet	ParameterSet/MotorTemperatur	Motor temperature, currently not supported

16.4.2Controllers model

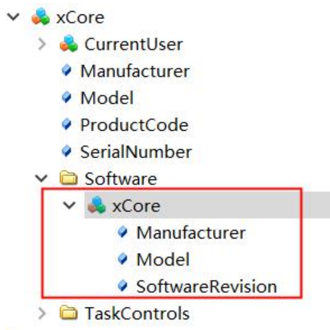


Type	Browse Name	Description
ContorllerType	xCore	1. Each robot contains an instance of ContorllerType, named xCore

ContorllerType child nodes are as follows:

Type	Browse Name	Description
Manufacturer	Manufacturer	Manufacturer
Model	Model	Robot model
ProductCode	ProductCode	Product number, currently not supported
SerialNumber	SerialNumber	Serial number, currently not supported
CurrentUser	CurrentUser	Currently not supported
Software	Software	A container containing SoftwareType instances
TaskControls	TaskControls	A container containing TaskControlType instances

16.4.2.1Software child nodes

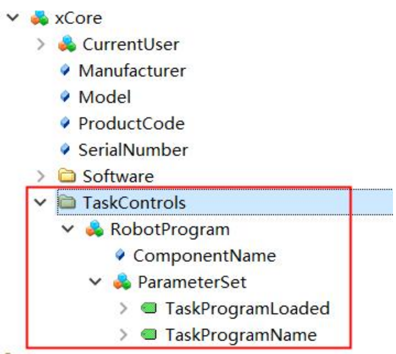


Type	Browse Name	Description
Software	xCore	xCore control system

ContorllerType child nodes are as follows:

Type	Browse Name	Description
Manufacturer	Manufacturer	Manufacturer
Model	Model	Robot model
SoftwareRevision	SoftwareRevision	Software version number, such as 2.1.2

16.4.2.2TaskControls child nodes

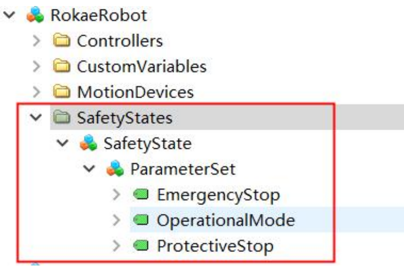


Type	Browse Name	Description
TaskControl	RobotProgram	TaskControlType instance named RobotProgram

ContorllerType child nodes are as follows:

Type	Browse Name	Description
ComponentName	ComponentName	Null
ParameterSet	ParameterSet/TaskProgramName	Display the current RL project name
	ParameterSet/TaskProgramLoaded	If the project has been loaded, display true, and if there is no project, display false

16.4.3SafetyStates

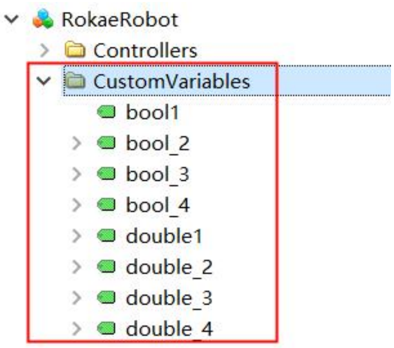


Type	Browse Name	Description
SafetyState	SafetyState	SafetyStateType instance named SafetyState

SafetyState child nodes:

Type	Browse Name	Description
ParameterSet	ParameterSet/EmergencyStop	Null
	ParameterSet/OperationalMode	Operation mode, enumeration values: 0-Other, 1-Manual Reduced Speed, 2-Manual High Speed, 3-Automatic, 4-Automatic External At present, the manual and automatic modes in the control system correspond to 1 and 3 in the above enumeration values respectively
	ParameterSet/ProtectiveStop	Internal safety gate stop and protection stop status of the controller

16.4.4CustomVariables



The custom variables configured in section 11.13 can be found under this node, and clients can interact with controller data by reading and writing variables under this node.

17Troubleshooting

17.1Control System Error Codes

17.1.11XXXX

Code	Description	Possible Reasons	Solution
10000	Error in parsing HMI request packet	Incorrect protocol of HMI request packet	Please check if the HMI and the control system version matches
10001	JOG startup failed	1. Not in the Manual mode; 2. Robot not powered on; 3. Robot in motion; 4. Not in the Position mode	Please make sure that the robot is in the Manual mode and powered on
10002	Quick adjustment startup failed	1. Not in the Manual mode; 2. Robot not powered on; 3. Robot in motion; 4. Not in the Position mode	Please make sure that the robot is in the Manual mode and powered on
10003	Mechanical zero calibration failed	1. Not in the Manual mode; 2. Robot in motion; 3. Not in the Position mode	Please make sure that the robot is in the Manual mode and not in motion
10004	Mechanical zero calibration succeeded	None	None
10005	Sensor zero calibration failed	1. Not in the Manual mode; 2. Robot in motion; 3. The robot is not around the mechanical zero; 4. Not in the Position mode	Jog the robot to the mechanical zero and make sure that the robot is in the Manual mode and not in motion
10006	Sensor zero calibration succeeded	None	None
10007	Failed to reboot the controller		Please stop the robot motion and the RL program
10008	Failed to clear encoder alarm	Servo encoder fault	Restart robot. If it's still present, contact ROKAE Technical Support
10009	Encoder alarm cleared successfully	None	None
10010	Failed to switch to the Manual mode	Mode switching is not allowed when the robot is in motion; or the robot is already in the Manual mode	Stop the motion and try again
10011	Failed to switch to the Automatic mode	Below are some possible causes: 1. The robot is in motion; 2. The robot is in emergency stop; 3. Drag is already enabled; 4. Automatic mode is already enabled	1. Not in motion; 2. Restore from emergency stop; 3. Drag is disabled
10012	Power-On Condition Check Failure	1. Not in correct operation mode; 2. Robot in emergency stop state; 3. Robot in torque control mode; 4. Shutdown signal received; 5. Servo in critical fault state; 6. Controller initialization incomplete	1. Switch to correct operation mode; 2. Reset emergency stop state; 3. Switch robot to position mode; 4. Check servo faults; 5. Wait for controller initialization to complete
10013	Power-off failed	The robot is in motion or it is not in	Stop the robot motion

		the Automatic mode	
10014	Failed to enable drag!	Drag cannot be enabled in the following cases: 1. Automatic mode; 2. Powered on; 3. Not in the Position mode; 4. In motion; 5. Not executing a routine task; 6. Safety monitoring or safety monitor triggered!	Switch to the Manual mode, keep the power off, and switch to the Position Mode, and then try again; or try again after reboot.
10015	Drag enabled successfully	None	None
10016	Failed to disable drag!	Wrong operating mode	1. If the drag is enabled through RCI, it should be disabled through RCI; 2. Restart the robot and try to recover.
10017	Drag disabled successfully!	None	None
10018	Failed to update the virtual wall	1.The set area is too small; 2.The robot's current position is outside the set area; 3. Drag mode is not enabled	1. Expand the virtual wall boundary; 2. Move the robot end-effector to the set area; 3. Enable the Drag mode
10019	Virtual wall updated successfully	None	None
10020	New teach pendant connection rejected	Teach pendant connection exists	Disconnect existing connection
10021	Socket-server connection timeout	Connection timeout	1. Please check the device connection; 2. Please check whether the server is working properly
10022	Wrong format of data received by the socketread instruction	Network error	
10023	IP address and port number for creating the socket are occupied	RL program or external communication socket has already used the IP address and port number. Two sockets may not use the same IP address and port number	Set different IP address or port numbers
10024	The created socket name is duplicated	The created socket name is duplicated	Change another socket name
10025	The socket fails to receive data	1. Network connection error; 2. Server did not send data in time; 3. Wrong terminator; 4. Wrong data type	1. Check the network connection; 2. Check whether the server sends data; 3. Check whether the terminator matches; 4. Check the type of data sent
10026	socket disconnected	None	None
10027	socket connected	None	None
10028	SocketRead data length does not match the set length	The number of data received does not match the set number	Send the correct number of data
10029	Failed to set input in the simulation mode		

10030	Failed to input signal in the simulation mode	1. Simulation mode is not activated; 2. The signal does not exist	1. Activate the Simulation mode; 2. Check input signal setting
10031	Simulation GI signal failed	1. Simulation mode is not activated; 2. The signal does not exist	1. Turn on the simulation mode; 2. Check GI signal setting
10032	Failed to set the output signal	1.The signal does not exist; 2.DO signal has been set as system output	1.Check the output signal setting; 2. Simulation is not bound as the system output signal
10033	Failed to set GO signal	1. The signal does not exist; 2. The set value is beyond the allowable range	Check the GO signal setting
10034	Error in parsing RCI packet	Unable to parse for wrong message length	Please check the RCI instruction format
10035	RCI parameters saved successfully	None	None
10036	RCI opened successfully	None	None
10037	RCI closed successfully!	None	None
10038	Failed to open RCI	1. The IP address cannot be same as the robot address (192.168.0.160) and local host (127.0.0.1); 2. Wrong format of IP address; 3. Port number already occupied; 4. Robot in motion; 5. Not in the Position mode	1. Please set the allowed IP address and port number; 2. Stop the robot motion
10039	Failed to close RCI	1. Robot in motion; 2. Not in the Position mode	Stop the robot motion and try to close again
10040	Failed to drag	There is a large deviation between the feedback and the model torque, and the drag can not be activated	1. Check whether the current Tool setting is consistent with the actual situation and whether the set tool mass center is reasonable; 2. Check the monitoring window to see whether the robot coordinate system and pose consistent with the actual situation; 3. Confirm that the current robot model and RD parameters are consistent with the actual parameters; 4. Try to return to the mechanical zero and zero the sensors before dragging; 5. For more detail, refer to the Drag Fault Troubleshooting Manual
10041	Connect the client to RCI	None	None
10042	Client disconnected from RCI	Client disconnection detected	Please check the client
10043	RCI is not responding. Please check auto and power-on status		
10044	Body parameter identification	None	None

	completed. Please reboot the robot		
10045	Load parameters are identified successfully	None	None
10046	Failed to update collision detection parameters		
10047	Enable collision detection	None	None
10048	Disable collision detection	None	None
10049	Load identification failed. The load exceeds Robot's rated load	The load of installed tool or workpiece exceeds the robot's rated load	Use a tool or workpiece within the robot's rated load range
10050	Load identification failed. Wrong load identification result.	Exception in sensor torque feedback	Check the sensor torque
10051	Failed to set joint position limit	1. The set angle is beyond the robot’s mechanical limit; 2. Not in the Manual mode; 3. Robot in motion; 4. Not in the Position mode	1. Switch to the Manual mode and stop the robot motion; 2. Confirm the robot's mechanical limit and set the angle within the range
10052	Joint position limit set successfully	None	None
10053	Correction of robot hard limit	None	None
10054	Correction of joint position limit	None	None
10055	Failed to set servo filter parameters	Failed to read servo filter data	Restart the robot. If the issue persists, please contact technical support.
10059	This model of controller does not support upgrading to version 3.1 or higher!	The command filter for the high-payload model has not been enabled.	Keep the controller at the pre-upgrade version. If there are any issues, please contact technical support.
10060	Setting of Maximum joint velocity over the limit	1. The joint velocity range in the Collaboration mode is 0-15 degrees/s; 2. The maximum axial velocity of each axis cannot be exceeded in the Non-Collaboration mode	Please check and set the value within the range
10061	Setting of Maximum TCP velocity over the limit	1. The TCP velocity range in the Collaboration mode is 0-250; 2. The maximum TCP velocity cannot be exceeded in the Non-Collaboration mode	Please check and set the value within the range
10062	Setting of Joint torque over the limit	The maximum joint torque of each axis is 3 times of the rated torque	Please check and set the value within the range
10063	Drag out of virtual range,virtual wall failure	Dragging exerts excessive external torque	1. Expand the virtual wall boundary; 2. Reduce the drag torque; 3.

			Increase virtual wall stiffness
10064	The virtual wall takes effect again	After leaving the virtual wall then enter the virtual wall again ,the virtual wall takes effect again	no repair
10065	Sync NTP failed	NTP service not installed; Failed to sync time with server	Install NTP service; Make sure NTP server is running
30060	Joint velocity over the limit		
30061	TCP velocity over the limit		
30062	Joint torque over the limit		
30063	Momentum over the limit		
30064	Total joint power over the limit	Total joint power exceeds the value set in the safety monitor	1. Slow down the motion speed; 2. Reduce the load and inertia; 3. Turn off the safety monitor
30065	Dual encoder position deviation over the limit		
30066	Dual channel data deviation of the torque sensor exceeds the limit		
10067	Total power over the limit	The maximum power of each axis is the maximum joint torque (Nm) × maximum joint velocity (radians/sec), where: 1. The maximum joint velocity in the Collaboration mode is 15 degrees/s; 2. The maximum axial velocity of each axis cannot be exceeded in the Non-Collaboration mode	Please check the parameters and their unit
10068	Industrial robots do not support pausing when collision detected for the moment. Please Switch to stopping		
10070	Enter the safety zone and stop the motion	None	None
10071	Enter the safety zone and start the Collaboration mode	None	None
10072	Trigger the Reduced mode and slow down the motion	None	None
10073	Trigger the secondary Collaboration mode and slow down the motion	None	None
10074	Exit the Reduced mode	None	None
10075	Trigger the Reduced mode	None	None
10078	Trigger limit during RCI operation	1. Joint or Cartesian space position over the limit; 2. Joint velocity over	Please check the torque instruction and the initial status of the robot

		the limit; 3. Joint or motor torque over the limit; 4. Large deviation between torque instruction and actual torque; 5. Robot in a singular position	
10079	Force control module protection triggered, force control mode exited	See the "content"	1. Check if the robot status is normal in force control mode; 2. Set proper force control protection parameters
10080	Failed to enable drag: joint position over the limit	Current position of the robot over the joint position limit	Jog the robot to a position within the joint position limit
10081	socket failed to send data	Network failure cause	Check the network problem
10082	socket data reception timeout	Network failure or code logic error	1. Check the code logic; 2. Check whether data is properly received and sent at the other end of the network
10083	Connection failed with external communication as a client. Trying to reconnect	1. Wrong IP address and port number; 2. Server device not properly started; 3. Abnormal device connection	1. Check whether the set IP address and port number are correct; 2. Reopen the server
10084	Connection failed with external communication as a server	1. Wrong IP address and port number; 2. Server device not properly started; 3. Abnormal device connection	1. Set the IP address to blank or 0.0.0.0; 2. The port number cannot be set to 0
10085	Socket creation failed. The same name as the external communication name	The socket in the RL program cannot have the same as external communication sockets	Use a name different from those of external communication sockets when creating a socket
10086	Socket creation failed. The socket name exceeds 30 bytes	Socket name too long	Shorten the socket name
10087	Failed to open the external communication: $arg	$arg	Process based on the cause prompt
10088	socket connected	None	None
10090	Safety zone set successfully	None	None
10091	Failed to set the safety zone	1. Not in the Manual mode; 2. Robot in motion; 3. Safety Zone behavior is set to "Trigger collaboration mode" but the Collaboration mode is not turned on	1. Switch to the Manual mode; 2. Stop the robot motion; 3. To trigger the Collaboration mode, please turn it on first
10092	Collaboration mode set successfully	None	None
10093	Failed to set the Collaboration mode	1. Not in the Manual mode; 2. Robot already powered on; 3. Robot in motion; 4. Parameters set over the limit	1. Make sure that the robot is in the Manual mode, stopped, and powered off; 2. Check the parameter setting of the Collaboration mode
10094	Safety monitor set successfully	None	None
10095	Failed to set the safety monitor	1. Not in the Manual mode; 2. Robot already powered on; 3. Robot in	Make sure that the robot is in the Manual mode, stopped, and

		motion	powered off
10096	Failed to set the collision detection trigger action	The trigger behavior is not supported	Please modify the trigger behavior
10097	DH parameters are saved successfully. Please restart the robot for these parameters to take effect.	None	None
10098	In this version, only part of the linkage parameters can be modified		
10099	Failed to save RD parameters. RD parameter changes over the limit	RD parameter changes over the limit	The limit range of RD parameter changes is +/- 50
10100	IP address and port number for creating the socket are occupied	RL program or external communication socket has already used the IP address and port number. Two sockets may not use the same IP address and port number	Please use different IP address or port numbers
10101	socket ip:port error	Duplicate or wrong IP address already used by the RL program and external communication sockets	Change the Socket to a free IP:PORT of the host or check the network settings
10102	Socket port error	Wrong port number already used by the RL program and external communication sockets	Change the socket port to a free port number from 0 to 65535
10103	SocketSendString failed	Connection not established or network error	Please check the network connection or use a connected socket
10104	SocketSendByte failed	Connection not established or network error	Please check the network connection or use a connected socket
10105	Failed to parse socket data	Data sent against the rules	Send data that conform to the rules
10106	Rapid motion failed. Not in the Manual mode	Wrong robot status. Not in the Manual mode	Switching to the Manual mode
10107	Rapid motion failed. Program is running	Wrong robot status. Program is running	Pause the program
10108	Rapid motion failed. Robot not powered on	Wrong robot status. Robot not powered on	Robot enables Power On
10109	Starting point not set. Error in Home instruction	Starting point not set. Error in Home instruction	Set the starting point through the Setup > Basic Settings > Quick Adjustment interface
10110	Calibration angle set successfully	None	None
10111	Failed to set the calibration angle	Calibration angle over the limit	Set the angle within the limit

10112	End-effector quick adjustment failed. Please select the appropriate adjustment type	End adjustment is used for fine-tuning the end-effector pose. Calculation may fail for large-scale end-effector adjustment	1. Check the robot's current pose and tool/work object parameters, and select the appropriate type of end-effector adjustment; 2. The robot is already in the corresponding pose; 3. The specified pose is out of the range of motion; 4. The specified pose may move the robot to a singularity
10114	Duplicate serial port names. Please enter again	When creating a new serial port or modifying the name of a serial port, the name should be different from the system IO, the socket name or existing serial port names	Check the system IO, socket name and existing serial port names, and re-enter the new serial port name
10115	Serial port created successfully	None	None
10116	Failed to create the serial port	1. Serial port hardware connection disconnected; 2. The serial port socket does not exist	1. Please check the serial port hardware connection; 2. Delete the serial port variable from the variable list and recreate one
10117	Failed to close serial port	1. Serial port hardware connection disconnected; 2. The serial port socket does not exist	1. Please check the serial port hardware connection; 2. Click pptomain and run again
10118	Serial port closed successfully	None	None
10119	Serial port does not exist	The serial port does not exist	Please create a serial port in the Communication - Serial Port Setting interface
10121	Serial port failed to send string	1. Serial port hardware connection disconnected; 2. The serial port socket does not exist	1. Please check the serial port hardware connection; 2. Delete the serial port variable from the variable list and recreate one, and ensure that the serial port communication is proper and available
10122	Serial port failed to read byte	Data read cannot be converted to byte type	Please check the content sent
10123	Serial port failed to send byte	1. Serial port hardware connection disconnected; 2. The serial port socket does not exist	1. Please check the serial port hardware connection; 2. Delete the serial port variable from the variable list and recreate one, and ensure that the serial port communication is proper and available
10124	Serial port failed to clear buffer	1. Serial port hardware connection disconnected; 2. The serial port socket does not exist	1. Please check the serial port hardware connection; 2. Click pptomain and run again
10125	Serial port failed to obtain the	1. Serial port hardware connection	1. Please check the serial port

	buffer length	disconnected; 2. The serial port socket does not exist	hardware connection; 2. Click pptomain and run again
10126	serial data reception timeout	Network failure or code logic error	1. Check the code logic; 2. Check whether data is properly received and sent at the other end of the network
10127	serial failed to receive data	1. Abnormal serial port connection; 2. Opposite end failed to send data in time; 3. Mismatch of parameters configured for both ends of the serial port	1. Check the serial hardware connection; 2. Check whether the opposite end has sent data; 3. Check the parameters configured for both ends of the serial port
10128	xDiagnose version too old. Unable to view the data	xDiagnose version too old	Can use xDiagnose version 0.3.8 or above
10129	The wait time of the conveyor belt for the work object over the limit	1. The conveyor belt stops but the tracking is still on; 2. The photoelectric switch functions abnormally and it cannot capture the work object trigger signal	1. Check whether the conveyor belt is moving properly; 2. Check whether the photoelectric switch captures the trigger signal properly
10130	Work object beyond the startup window		
10131	Work object beyond the working area	Work object beyond the working area. Unable to keep tracking	Please adjust the work object position
10132	Conveyor belt speed below the threshold	The conveyor belt stops moving or the encoder is disconnected	Please check whether the conveyor belt stops moving or the encoder is disconnected
10133	Failed to switch to the hot-swapping mode	Hot-swapping is not supported by the model, control cabinet or safety board firmware	1. Check whether the current robot model is an industrial model with the XBC5 control cabinet or a collaborative CR model; 2. Check whether the robot safety board is a mini board and the firmware is upgraded to 2.0; 3. Check whether the ENI file is correctly configured
10134	Failed to switch to the hot-swapping mode	1. Safety board failed to switch the status correctly; 2. Communication error	Please contact the supplier for support
10135	Switched to hot-swapping mode successfully	None	None
10136	Rapid motion failed. Angle over the mechanical limit	Joint angle over the mechanical limit	Modify the angle value and make sure it is within the mechanical limit
10137	Work object on the conveyor belt have been associated twice	Execute WaitWobj again when DropWObj is not executed	Please check the instruction. The same work object should be associated only once
10138	Failed to execute Waitwobj	Caused by continuous execution in the	Single-step execution in the Manual

	instruction	Manual mode	mode or continuous execution in the Automatic mode
10139	There are no associated work objects	1. There is no work object being tracked; 2. There is no conveyor belt with the specified name	Please place the work object on the conveyor belt and trigger the photoelectric switch. The status monitoring shows whether the work object is in the correct position
10140	Abnormal friction force identification result. Use nominal friction coefficient		Contact the Control Team
10141	The mass input is out of range		The mass input should be between 0 and the maximum load
10142	Load-free identification not performed. Loaded identification stops		Please perform load-free identification first and then loaded identification
10143	Failed to set speed before moving to a point position or force control identification		
10144	This model only modifies the friction force after kinetic identification. It is recommended to perform friction force identification directly.		
10145	PCB 3/4 axis robots do not support the Reduced mode for the moment. There is no trigger action	PCB 3/4 axis robots do not support the Reduced mode for the moment	
10146	The Robot does not support dynamic identification	The Robot does not support dynamic identification	1.Please click the friction identification button
10147	Function authorization failure	See the content	Please contact the administrator of licensing
10148	The axis max speed exceeds the limit, data won't be saved	See the content	Please edit the axis max speed setting on HMI
10149	Socket disconnected, pause program according to the socket setting	Socket disconnected	Check socket connection status
10150	Socket disconnected, pause program and power down motors according to the socket setting	Socket disconnected	Check socket connection status
10151	Failed to move to point	Invalid to set "handheld" or "external" for both the tool and the wobj	Select the correct tool and wobj
10152	Failed to save DH parameters.	DH parameter changes over the limit	The limit range of DH parameter

	DH parameter changes over the limit		changes is: length +/- 50mm , angle +/- 10°
10153	Calibration failed. user frame id error	Calibration failed. user frame id error	Select the correct user frame system and recalibrate.
10154	The robot is in a safe retraction state and cannot perform the current operation.	The robot is in a safe withdrawal state and is not allowed to perform the current operation.	After the jog robot has safely retracted in manual mode, try again.
10200	Electronic nameplate does not exist	1. The body has no electronic nameplate; 2. Hardware damage of the electronic nameplate; 3. The serial port of electronic nameplate configured incorrectly	1. Check the hardware connection of the electronic nameplate serial port; 2. Check the electronic nameplate hardware; 3. Check the serial port configuration of the electronic nameplate
10201	Error in electronic nameplate data reading	1. Wrong protocol address of the electronic nameplate; 2. Hardware damage of the electronic nameplate	1. Check whether the protocol matches; 2. Check the electronic nameplate hardware
10202	Error in electronic nameplate data writing	1. Wrong protocol address of the electronic nameplate; 2. Hardware damage of the electronic nameplate	1. Check whether the protocol matches; 2. Check the electronic nameplate hardware
10203	Electronic nameplate data mismatch	1. The robot body has been changed; 2. The control cabinet has been changed	Data overwrite after confirmation
10204	Encoder battery voltage too low	1. Running for too long; 2. Encoder battery is damaged	Replace the encoder battery
10205	The duration of the robot motion exceeded the warranty period	Not maintained in time	Regular maintenance
10206	Data overwritten successfully	None	None
10207	Electronic nameplate data overwrite not allowed	1. Electronic nameplate does not exist; 2. Model, ID or hardware version do not match	Please check the electronic nameplate according to the log
10208	RC data overwrite not allowed	1. The controller model does not match with the body; 2. Data has not been burnt to the electronic nameplate or wrong data is burnt	1. Select the correct model and restart the controller; 2. Burn the correct data to the electronic nameplate
10209	Socket communication failed	1.IP and port are already used by external communication 2.The socket name has the same name as the external communication, the default SYS_SOCKET name is already used by the external communication.	1.Change the port or IP of the socket 2.Change the name of the socket
10210	Switching precision compensation state not allowed	Zero point data error	Check zero point data

13013	Emergency stop triggered	None	Manually resume emergency stop
13014	Safety gate opened	None	Manually close the safety gate
13015	Start to calculate body parameter identification	None	None
13016	Error in program lexical or grammar check	Grammatical error in RL program	Please check the RL program
13017	Program PP_to_main failed	1. Task destroyed; 2. Project file not loaded; 3. main function missing	1. Reinitialize the task; 2. Reload the project; 3. Check if there is a main function in the file
13018	Program PP_to_func failed	1. Task destroyed; 2. Symbol table not yet established for the task; 3. The function to be jumped to does not exist	1. Reinitialize the task; 2. Check if there is a grammatical error in the function and recreate the symbol table; 3. Check if the function exists
13019	Program PP_to_line failed	1. Task destroyed; 2. Symbol table not yet established for the task; 3.The program PP_to_line jump is only allowed inside the same PROC/FUNC	1. Reinitialize the task; 2. Check if there is a grammatical error in the function and recreate the symbol table; 3.Please check if the jump is inside the same PROC/FUNC
13020	All RL tasks have been stopped. Please check the error message or click PPToMain	1. The single loop mode task is finished; 2. When an error occurs, the task will stop	1. Check the program for logical errors; 2.Click pptomain to run again
13021	Wrong base coordinate system. Failed to set	The controller can not parse the instruction to set the base coordinate system, possibly because the controller version is not compatible with the HMI	Please check whether the control system version matches with HMI
13022	Next step select unloaded Tasks.		Check the corresponding task in the project's task list and execute pptomain.
13030	RSC Detects that the Robot Exceeds the Limit of Power	The robot power exceeds the limit, or the power of configuration in RSC is too low, or the RSC fails	Check whether the RSC robot power limit range is reasonable, or close the RSC robot power limit
13031	RSC Detects that the Robot Exceeds the Limit of Momentum	The robot momentum exceeds the limit, or the momentum of configuration in RSC is too low, or the RSC fails	Check whether the RSC robot momentum limit range is reasonable, or close the RSC robot momentum limit
13032	RSC Detects that the Robot Exceeds the Limit of Elbow or TCP Force	The robot Elbow or TCP force exceeds the limit, or the Elbow or TCP force of configuration in RSC is too low, or the RSC fails	Check whether the RSC robot Elbow or TCP force limit range is reasonable, or close the RSC robot Elbow or TCP force limit
13033	RSC Detects that the Robot Exceeds the Limit of TCP Velocity	The TCP velocity of robot exceeds the limit, or the TCP velocity of configuration in RSC is too low, or the RSC fails	Check whether the RSC robot TCP velocity limit range is reasonable, or close the RSC robot TCP velocity limit

13034	RSC Detects that the Robot Exceeds the Limit of Position	The position of robot exceeds the limit, or the RSC fails	Check whether the RSC robot position limit range is reasonable, or close the RSC robot position limit
13035	RSC Detects that the Robot Exceeds the Limit of Posture	The posture of robot exceeds the limit, or the RSC fails	Check whether the RSC robot posture limit range is reasonable, or close the RSC robot posture limit
13036	RSC Detects that the Robot Exceeds the Limit of Collision Force	The collision force of robot exceeds the limit, or the RSC fails	Check whether the RSC robot collision force limit range is reasonable, or close the RSC robot collision force limit
13037	RSC Detects that the Robot Exceeds the Limit of Joint Power	The joint power of robot exceeds the limit, or the RSC fails	Check whether the RSC robot joint power limit range is reasonable, or close the RSC robot joint power limit
13038	RSC Detects that the Robot Exceeds the Limit of Joint Position	The joint position of robot exceeds the limit, or the RSC fails	Check whether the RSC robot joint position limit range is reasonable, or close the RSC robot joint position limit
13039	RSC Detects that the Robot Exceeds the Limit of Joint Velocity	The joint velocity of robot exceeds the limit, or the RSC fails	Check whether the RSC robot joint velocity limit range is reasonable, or close the RSC robot joint velocity limit
13040	RSC Detects that the Robot Exceeds the Limit of Joint Torque	The joint torque of robot exceeds the limit, or the RSC fails	Check whether the RSC robot joint torque limit range is reasonable, or close the RSC robot joint torque limit
13041	RSC Detects that the Communication of joint is Abnormal	The joint communication of robot is abnormal, or the RSC fails	Try to reboot the robot, or contact the manufacturer to check the hardware failure
13042	RSC Detects that the Operation of joint is Abnormal	The joint operation of robot is abnormal, or the RSC fails	Try to reboot the robot, or contact the manufacturer to check the hardware failure
13043	The Communication between RSC and Controller is error	The communication between RSC and controller is error	Try to reboot the robot, or contact the manufacturer to check the hardware failure
13044	Failed to Synchronize Sdo Data Between Controller and RSC	Failed to synchronize sdo data between controller and RSC	Try again, or contact the manufacturer to check the hardware failure
13045	Succeeded in Synchronize Sdo Data Between Controller and RSC		
13046	The security limit of RSC is exceeded when dragging	When dragged, TCP speed exceeds 250mm/s, triggering safe stop	
13047	RSC Initialization Parameter	The basic security parameters of RSC	Hard restart the robot or control

	Checks are Inconsistent.	are inconsistent with those of the master controller	cabinet
13048	The communication between the RSC and the master controller enters a secure data state (0x08)	The communication between the RSC and the master controller enters a secure data state (0x08)	
13049	The parameter of joint position limit for RSC is invalid	The set angle is beyond the robot’s mechanical limit	Confirm the robot's mechanical limit and set the angle within the range
13050	The parameter of safety home position for RSC is invalid	The set angle is beyond the robot’s mechanical limit	Confirm the robot's mechanical limit and set the angle within the range
13051	Safety gate closed		
13052	Safety gate is opened, and the RL program cannot be continued	Safety gate is opened	Reset the safety gate, and try again
13053	RSC trigger Flying speed protect	Robot experience flying speed	
13057	RSC detected a short circuit fault in channel		Please check if there is any wiring error. After the fault is restored, power off and restart
13058	RSC detects inconsistent dual-channel signals of safety DI signals	RSC safety DI signal dual channel signal is inconsistent	Please check and repair the wiring of the corresponding DI signal, then reset it to 0, and then set it to 1 at the same time to recover the fault
13059	RSC detected a dual machine communication failure		If it persists, it is recommended to replace the RSC
13060	RSC detected MCU address fault		If it persists, it is recommended to replace the RSC
13061	RSC detected inconsistent output data fault	The data information output by the host and slave computers is inconsistent	If it persists, it is recommended to replace the RSC
13062	RSC detected voltage exceeding the limit	The voltage of the power supply is not between 20~30V	Check RSC power supply voltage
13063	RSC detected algorithm library malfunction		
13064	RSC detected that Bamboo is not running properly		Power off and restart. If the issue persists after restarting, it is recommended to replace the RSC
13065	RSC detects FSOE internal communication failure		Re power on for testing. If the issue persists, it is recommended to replace the RSC
13066	RSC detected communication failure in FSOE department		Re power on for testing. If the issue persists, it is recommended to

			replace the RSC
13067	RSC detected abnormal communication from station		
13068	RSC detected that the slave station is not in the FSOE data state		
14001	Network disconnected	None	None
14002	Network connection closed	None	None
14003	Network connection established	None	None
14004	Network connection monitoring enabled	None	None
14005	Reconnecting network	None	None
14006	Reconnecting network	None	None
14010	Write is not allowed for bound registers	The register is bound to the system functional register. Write operation in RL program is not allowed	Please use another register or unbind this register
14011	Failed to open fieldbus device. Corresponding Ethercat slave not found	No Ethercat slave is configured or there is an exception in the device linking	Import and configure Ethercat slave
14012	The register is readonly register, Write operation is not allowed	The register is readonly register, Write operation is not allowed	Please use writeonly register， or modify the register attribute
14020	Fieldbus device opened successfully	None	None
14021	Failed to open fieldbus device	See the "content"	Please check the fieldbus device configuration and make sure that the corresponding device is correctly connected
14022	Fieldbus device closed successfully	None	None
14023	The serial port is already occupied by the fieldbus device	The serial port is already used by modbus RTU fieldbus device	Close the fieldbus device or use another serial port
14030	Error in profinet fieldbus model configuration	The data model selected for the slot does not match the PLC	Reconfigure the data model of the slot
14031	Error in holding register file	Error in crc validation for holding register files. Files are corrupted	Delete holding register file
14032	Failed to open fieldbus device.SDO setup failed	SDO setup failed	Check if SDO initialization settings meet device requirements
14501	Unable to set system DO	DO signal is a system IO signal and cannot be set	Please use other output signals or unbind this signal in the system IO
14502	Failed to set GO signal	The value set is beyond the valid range of the signal	Please check the value

14503	Failed to set AO signal	The value set is beyond the valid range of the signal	Please check the value
14504	Failed to set PulseDO. Time out of range	Wrong pulse time, [0.001, 2000] S range	Re-enter the pulse time
14505	Failed to set PulseReg. Time out of range	Wrong pulse time, [0.001, 10] S range	Re-enter the pulse time
14510	Failed to load the output signal	The mapped physical port is in conflict with other signals or system output signals	1. Check if the output port is occupied; 2. Reconfigure the system IO
14511	Failed to set the system input signal. There is a conflict	Violation of system input configuration rules. Possible causes: 1. Duplicate system IO is configured; 2. The corresponding IO is already occupied by the RL project; 3. IO signal does not exist	1. Configure independent system IO; 2.Modify the system IO occupied by the RL project; 3.Open or create IO device
14512	Failed to set the system output signal due to a conflict	Violation of system output signal configuration rules. Possible causes: 1. Duplicate system IO is configured; 2. The corresponding IO is already occupied by the RL project; 3. IO signal does not exist	1. Configure independent system IO; 2.Modify the system IO occupied by the RL project; 3.Open or create IO device
14521	Failed to initialize Ethercat IO device. Corresponding Ethercat slave not found	No Ethercat slave is configured or there is an exception in the device linking	Import and configure Ethercat slave
14530	The status of the safety panel expansion IO device has changed	Hardware damage or abnormal linking for the expansion IO device	Please check the hardware status and link status of the expansion IO devices on the safety panel
14531	The status of the safety panel expansion IO device has changed	New expansion IO device connected	None
14532	IO device status has changed	IO device configuration connected	None
14533	IO device status has changed	IO device configuration connected	None
14534	IO device status has changed	IO device connection error	None
14535	The imported register variable uses a bus device that does not exist on the current machine	The bus device is not configured on the current machine	Manually edit the register variable and change the device name to a bus device that exists on the current machine
14536	Empty signal name	Used an empty signal	Use the signal with the normal name
14537	Signal type invalid	An invalid type of signal was used	Use the correct signal type
14538	Signal mapping device not existed	The signal is bound to a non-existent device	Bind the signal to the correct device
14539	Signal mapping device is disabled	The device for signal mapping is disabled	1.Bind other normal devices 2.Re-enable disabled devices
14540	New group signal mapping	The relationship between the starting	The group signal end port number

	invalid, Port mapping error	port and ending port of the group signal is incorrect	should be greater than the start port number; 2.The port number of the signal should not exceed the device port range
14541	New group signal mapping invalid, Port number should not be greater than 32	Too many signal ports in the group	The number of signal ports in the group should not exceed 32
14542	New signal mapping invalid, Port mapping error	Signal port setting error	1.Change the starting and ending ports of the signal to make them consistent; 2.The port number of the signal should not exceed the device port range
14543	New signal mapping invalid, Output physical port is already used	The port used by the signal is occupied	1. Configure independent system IO; 2.Modify the system IO occupied by the RL project; 3.Open or create IO device
15000	Failed to execute drag path playback	Execute drag path playback. No available drag data in the buffer	Record the drag path again
15001	Failed to save the drag data	No available drag data in the buffer	Please record the drag path again
15002	Failed to execute drag path playback	Drag path record deleted accidentally	Record the drag path or import the path again
15003	Failed to execute drag path playback. Error in drag file data	Playback path from another type of robots	Import the correct drag playback path or record the trajectory again
15004	Failed to save the drag data	Failed to drag the replay serialized data to the hard disk file	Please record the drag path again or restart the robot
15005	Drag data saved successfully	None	None
15006	Execute drag path playback	None	None
15007	Start recording the drag playback data	None	None
15008	Stop recording the drag playback data	None	None
15009	Stop executing drag path playback	None	None
15010	Failed to record path data: too few path waypoints	Too few drag path waypoints	Please record the drag path again
15011	Failed to record path data: joint angle over the limit	Joint in the drag path exceeds the joint position limit	Please record the drag path again
15012	Failed to record path data: joint velocity over the limit	Joint velocity in the drag path exceeds the limit	Please record the drag path again
15013	Failed to record the path data: the speed is not zero when recording ends	The speed is not zero when path recording ends	Click end recording after the robot stops
15014	Failed to record path data: The	The speed is not zero when recording	Make sure the robot is stopped

	speed is not zero when recording starts	starts	before recording starts
15015	Failed to record the path data: motor not powered on	Motor not powered on	Robot Power-On
15016	Path data recorded successfully	None	None
15017	Execute drag path playback. Failed to read path	Failed to parse the drag path data. The path data may be tampered	Import the path or record the trajectory again
15018	Path playback. The speed set exceeds the limit	The playback speed set is too fast	Decrease the path playback speed
15019	Waypoints in the path playback exceed the joint position limit	Playback path waypoint exceeds the joint position limit	Adjust the current joint position limit
15020	Path playback is not allowed when the track is turned on.	Path playback is not allowed when the track is turned on.	Path playback after closing the tarck.
17001	Register failed to read data	1. The register does not exist; 2. The register does not match the variable type; 3. The array subscript is out of range	Please check the register settings
17002	Register failed to write data	1. The register does not exist; 2. The register does not match the variable type; 3. The array subscript is out of range	Please check the register settings
17003	modbus failed to read the input register		
17004	Failed to load register configuration	Failed to parse register configuration file (registers.json). See the "content" for reasons	Please try to reconfigure the register or erase the configuration
17005	modbus communication failed, modbus link not established		
17006	modbus configuration saved successfully	None	None
17007	External communication configuration saved successfully	None	None
17008	Failed to parse modbus configuration	Error in modbus register configuration	Check the modbus register configuration
17009	Failed to load the bus device configuration file	Failed to parse the bus device configuration file (fieldbus_device.json). See the "content" for reasons	Please try to reconfigure the bus device or erase the configuration
17100	Failed to turn on cclink. cclink gateway module not retrieved	cclink gateway module not configured or an exception in the device linking	Import and configure cclink gateway module
17101	cclink already turned on	None	None

17102	Failed to turn on cclink		
17103	cclink already turned off	None	None
17104	Successful communication with CC-LINK IEF Basic master	None	None
17105	Communication with CC-LINK IEF Basic master disconnected	None	None
17200	There is no any set of servo params files that meet the rules in the controller	Each function of servo parameter switching can only be used if there is at least one set of servo params in the controller	None
17201	Power on failure, power on is prohibited during servo parameter switching	Power on failure, power on is prohibited during servo parameter switching	None
17202	It is prohibited to switch servo parameters while robot is power on	It is prohibited to switch servo parameters while robot is power on	None
17203	Successfully switched servo parameters	None	None
17204	Servo parameter switching mismatch	None	None
17205	Electroplating line visual socket, IP cannot be empty or '0.0.0.0', port cannot be empty	None	None
17206	Quick adjustment of custom pose beyond limit.	None	None
17207	There is an external input that is rejected.	None	None
17300	RL program customization stopped and cannot continue running	After pptomain 、 pptoline or pptofunc,start run program	None
17301	xService's connection disconnected	1.Connection between HMI and xCore disconnected unexpectedly; 2.SDK as client disconnect this connection	1.On HMI disconnect with xcore and reconnect it.
17310	Servo parameter acquisition failed		
17311	Successfully obtained servo parameters		
17312	Servo parameter download failed		
17313	Successfully downloaded servo parameters		
17314	Some fields in cfg are missing	Cfg file is incomplete	please update the cfg file

17315	Fail to change IP	See content	Enter the appropriate IP
17316	Rail zero calibration failed	1. Not in the Manual mode; 2. Robot in motion; 3. Not in the Position mode	Please make sure that the robot is in the Manual mode and not in motion
17317	Rail zero calibration succeeded	None	None
17318	Soft estop state does not allow power on	There is currently a read-only register bound to ctrl_soft_estop function code and low level, no power on allowed	Reset the register bound with the ctrl_soft_estop function code
17319	Soft emergency stop triggered	none	Resume soft emergency stop by register
17320	Motion commands status is inconsistent with the operation	See the content	Do PPtoMain or reset cache
17500	The driver is in a critical error state. It's not allowed to clear alarm and power on.	The driver is in a critical error state, hardware may in error state.	Try to power off and restart the control system; Contact the manufacturer's technical support.
17400	OpcUA Variable read failed	1.The variable is not existed; 2.The variable type is different from the parameter	Please check the RL command and the OpcUA variable configuration
17401	OpcUA Variable write failed	1.The variable is not existed; 2.The variable type is different from the parameter	Please check the RL command and the OpcUA variable configuration
17402	Soft calibration failed	1. Not in the Manual mode; 2. Robot in motion; 3. Not in the Position mode	Please make sure that the robot is in the Manual mode and not in motion
17403	Axis zero calibration successfully		
17404	Axis soft calibration successfully		
17405	Soft calibration successfully		
17406	Axis Soft calibration failed	1. Not in the Manual mode; 2. Robot in motion; 3. Not in the Position mode	Please make sure that the robot is in the Manual mode and not in motion
17407	Opening rail failed, please close the safety area first		
17408	Safety area open failed	When the rail is opened, the safety area cannot be opened	
17507	Conveyor start tracking failed..	None	None
17508	Conveyor stop tracking failed..	None	None
17509	Conveyor calibration failed, the number of points in the X direction is not 3.	Please check the calibration process	None
17510	Conveyor calibration failed, transmission ratio is 0.	1.The conveyor belt is not turned on,2.Encoder value abnormality	None
17511	The safety stop parameters are set incorrectly	The safety stop parameters may be set to zero	Please set The safety stop parameters to proper values
17512	During the tracking process of	None	None

	the conveyor belt, it is prohibited to execute ordinary motion commands		
17513	The conveyor belt stopped during tracking and cannot continue running	None	After pptomain, rerun the program
17520	OPC UA server start failed	1.Port is wrong or occupied; 2.Error Configuration	1.Use the right port; 2.Change the configuration
17521	HMI link	None	None
17522	HMI Motor	None	None
17523	HMI Motor	None	None
17524	PP_To_Main	None	None
17525	PP_To_Line	None	None
17526	PP_To_Func	None	None
17527	Reload Project	None	None
17528	Run Project	None	None
17529	Project Forward	None	None
17530	Project Back	None	None
17531	Manual mode switching successfully	None	None
17532	Automatic mode switching successfully	None	None
17600	Turn off maximum torque monitoring		
17700	None	None	
30067	joint power over the limit	None	None
30068	TCP angular speed over the limit	None	None
30069	elbow speed over the limit	None	None
30070	elbow angular speed over the limit	None	None
30071	moment over the limit	None	None
30072	tool attitude over the limit	None	None
18000	Robot exits the shared area, freeing the shared area from occupation	None	None
18001	Robots enter the shared area, and the shared area is occupied	None	None
18002	The robot has entered the occupied shared area and is slowing down and stopping at maximum capacity	None	None
18003	The robot is about to enter the	None	None

	occupied shared area, slowing down and pausing, waiting for the shared area to be released		
18004	The robot has entered the prohibited zone and has slowed down to a maximum capacity to stop	None	None
18005	Robots are about to enter the prohibited zone, plan to stop	None	None
18006	Shared area failure, robot continues to operate	None	None
18007	When switching the reduction mode, the current joint axis angle has exceeded the reduced joint position	None	Please reset the reasonable reduction joint position
18008	Cannot change the dimension of the array	None	Do not modify array dimensions
18009	controller logs are corrupted,using backup logs	may be caused by a power failure or abnormal restart	no need to fix

17.1.23XXXX

Code	Description	Possible Reasons	Solution
31001	Configuration Parameter Error of EtherCAT Master	Mismatched EtherCAT configuration files	Re-import the configuration files, and restart the control system
31002	Failed to Read EtherCAT Authorization Files	The authorization file does not exist or the read and write permissions are incorrect	Re-import the authorization file
31003	EtherCAT Authorization Failure	EtherCAT not Authorized	Please check the EtherCAT authorization code and try to reauthorize it on the EtherCAT authorization interface. After the authorization is successful, restart the robot to take effect
31004	EtherCAT Master Configuration Failure	The master configuration fails due to mismatched EtherCAT configuration files	Re-import the configuration files, and restart the control system
31005	DC Configuration Failure	The configuration fails due to mismatched EtherCAT configuration files	Re-import the configuration files, and restart the control system
31006	DCM Configuration Failure	The configuration fails due to mismatched EtherCAT configuration files	Re-import the configuration files, and restart the control system
31007	Bus Scan Failure	The configuration fails due to mismatched EtherCAT configuration	Re-import the configuration files, and restart the control system

		files	
31008	The Number of Configured Slaves Does not Match that of Scanned Slaves	1. The configuration fails due to inconsistency between the EtherCAT configuration and the actual network topology; 2. Hardware failure in the EtherCAT network from the slave equipment	1. Re-import the configuration files, and restart the control system; 2. Contact the manufacturer's technical support
31009	Failed to Enable the EtherCAT Bus	Due to errors in EtherCAT bus startup process, some slaves cannot switch to OP mode correctly, resulting in bus failure	Try to restart the control system; contact the manufacturer's technical support
31010	Internal Axis Servo Initialization Failure	The drive malfunctions internally	Check whether the servo drives are disconnected from each other
31011	IO Slave Initialization Failure	1. The IPC and general IO modules are disconnected from each other; 2. General IO modules and safety IO modules are disconnected from each other	Restart the control system after reconfiguring the IO signal according to the failure reason
31012	Tailboard Slave Initialization Failure	The IPC and the general tailboard slave module is disconnected from each other, or the tailboard slave hardware fails	Check the hardware connection, or replace the tailboard slave hardware
31013	Safety Board Slave Failure	The IPC and the safety board slave modules are disconnected from each other	Check hardware connections
31014	Abnormal Communication with Slave Devices	1. The communication between IPC and slave module is interrupted; 2. The slave device malfunctions	1. Confirm whether the EtherCAT debug cable is unplugged from the robot side; 2. Check the hardware connections and restart the control system; 3. Please contact the technical support
31015	Servo Alarm	The servo drive sends an alarm, and the specific reason for the alarm needs to find the corresponding drive manual according to the error code reported by the servo	1. Find the corresponding drive manual according to the error code, and handle it according to the manual guidance; 2. Contact the technical support
31016	Servo alarm cleared	None	None
31017	Ethercat is not Authorized, and the Trial Period Ends	Ethercat is not authorized and can only be used for one hour	Please contact the technical support to ask for the Ethercat authorization code and authorize on the EtherCAT authorization interface. After the authorization is successful, restart the robot to take effect
31018	Abnormal Connection between	1. The cable between the slave	1. Please check whether the cables

	Slave Device and Controller	device and the controller malfunctions; 2. The slave malfunctions	between the slave devices fail; 2. Please contact the technical support
31019	Failed to Initialize the CClink Gateway Slave	The hardware connection is abnormal or the hardware malfunctions	Check hardware connections
31020	Ethercat IO Slave Module not Adapted	The Ethercat IO slave module that has not been adapted is used, therefore the controller has no information about the current IO slave. Users need to ensure that the IO module is configured and used correctly	Since the PDO setting of the IO module that has not been adapted is unknown, there is a certain risk in direct use. Please use it with caution. Generally, digital IO modules with less than 64 channels can be used directly; however, the analog IO module is not recommended to be used directly because the controller is unknown to its analog quantity type, range and accuracy parameters, and the PDO settings of various equipment manufacturers may also be different. Please contact the technical support for adaptation before using it
31021	Safety Board Slave Initialization Failure	The safety board firmware version configured by the eni file is inconsistent with the actual firmware version	Check whether the safety board firmware version of the eni file is consistent with the actual firmware version
31022	Tailboard slave Input Current Signal Overload	When the tailboard slave channel AI is set to current mode, make sure that the actual input is current and does not exceed the limit parameter, otherwise the hardware may be damaged	Please disconnect the input signal of the corresponding channel and enter the xpanel configuration interface to set as Voltage Mode
31023	Tailboard slave Input Current Signal Overload	The xpanel setting has been completed, and the overload is recovered	To continue using the current mode, ensure that the input current is not overloaded
31024	xPanel terminal slave station hardware test failed	xPanel terminal slave station hardware test process is incorrect,hardware connection is incorrect or hardware is faulty	Please confirm the xPanel test process is correct. If the fault persists,contact R&D for troubleshooting
31025	Safety Board Type Configuration Error	The safety board type of the configuration is inconsistent with the actual safety board type	Check whether the safety board type of the configuration is inconsistent with the actual safety board type
31026	Initialization of internal axis servo drive failed	The server drive version in the ENI file of the master configuration is inconsistent with the server drive	Replace the main site ENI file to confirm that the correct ENI file is used.

		configuration scanned	
31027	Unable to read the primary ENI file correctly	The master ENI file was not found in the system	Re upgrade the ENI file corresponding to the current model
31028	EtherCAT failed to scan the bus, and the scan slave information does not match the slave information in the configuration ENI file	The configuration of slave information or topology in the ENI file of the master station is inconsistent with the scanned EtherCAT slave station	Replace and upgrade the correct ENI file
31029	EtherCAT scan bus failed with cross wiring from the slave station	EtherCAT slave station wiring error	Please check whether the wiring of EtherCAT slave station is correct
31030	External Axis Servo Initialization Failure	1. An unconnected external axis, such as a guide rail, was mistakenly opened 2.The drive malfunctions internally	According to the cause of the error, close the external axis that was opened incorrectly or check if the connection with the external axis is interrupted
31031	Initialization of external axis servo drive failed	The server drive version in the ENI file of the master configuration is inconsistent with the server drive configuration scanned	Replace the main site ENI file to confirm that the correct ENI file is used.
31032	The ESI file and controller version of the safety board RSC do not match	The controller version is 3.0, and the ESI file of the safety board RSC does not match the controller version. Some of RSC's safety functions will not function properly.	We need to upgrade the ESI file of the RSC security board and the ENI file of the main control
32001	Power-On Failure	Servo faults, STO circuit open, power-on aborted, etc.	1. Confirm operation procedure; 2. Check servo status; 3. Clear servo alarms before power-on retry
32002	Power-on Failed, STO not Connected	None	None
32003	Blocking of Communication Thread between Controller Master and Slave Device	The communication between the controller and the slave device is abnormal due to software reasons	1. Soft restart of controller; 2. Contact the technical support
32004	EtherCAT Thread Blocked, Scheduling Timeout	Internal system error	Restart control system
32005	The EtherCAT Thread was Blocked and Timed Out over 5000 Times in a Row	Internal system error	Restart control system
32006	Position Command Rejected	The position command to be sent to the servo has a big jump, and the generated speed exceeds the maximum speed of the motor, which may cause danger. To be safe, perform the power-off operation	Re-power on to operate

32007	Ethercat Thread Event Execution Timeout	Internal alarm	Internal state record, which can be ignored
32008	Error while writing Servo Zero	Only support CR joints, confirm whether it is a CR robot and whether the firmware is correct	None
32009	write servo zero successfully		
32010	Encoder Battery Voltage Low Warning	Encoder Battery Voltage Insufficient	Please replace encoder battery in time
32011	Encoder Battery Voltage Low Warning Cleanup		
35001	Short Circuit of the Drive	Output to output, output to ground, internal PWM bridge error	Troubleshoot circuit problems and eliminate short circuit
35002	Over-Temperature of Drive	The internal temperature of the drive reaches the set value	Reduce the drive temperature below the set value
35003	Over-Voltage of Drive	The bus voltage exceeds specified voltage limit	The bus voltage is restored within the specified voltage range
35004	Under-Voltage of Drive	The bus voltage is below the specified voltage	The bus voltage is restored within the specified voltage range
35005	Over-Temperature of Encoder Motor	The motor over-temperature switch shows the over-temperature error	Restore the temperature switch to normal state
35006	Encoder Feedback Error	1. 5V output inside the drive is over-current; 2. The resolver or analog encoder is not wired; 3. The level exceeds the error range; 4. The incremental encoder differential signal is not wired	1. The encoder power supply is restored to the specified voltage range; 2. The feedback signal is restored to the specified level range; 3. The differential signal is connected well
35007	Drive Phase Error	The phase angle based on the encoder fails to match with the switch state of the HALL. This error occurs only when the brushless motor is configured to be sinusoidal. This error does not occur during the resolver feedback or when the HALL correction function is stopped.	The phase angle based on the encoder is consistent with the switching state of the HALL
35008	Drive Reached the Current Limit	Motor overload or abnormal circuit	1. Check whether the motor is overloaded; 2. Check the circuit; 3. Contact the technical support
35009	Drive Reaching Voltage Limit	1. The set speed is too high; 2. The motor is abnormal	1. Check whether the set speed is abnormal; 2. Check whether the motor is abnormal; 3. Contact the technical support
35010	Power-on of Drive at Positive Limit	The drive is over positive limit	Restore the drive limit
35011	Power-on of Drive at Negative	The drive is over negative limit	Restore the drive limit

	Limit		
35012	Drive Following Difference out of Tolerance	Beyond the following error set by the user	Check whether the upper instruction fails and check the robot state
35013	Position Counter in Place	Internal error of drive	Try to restart. If the problem is not solved, please contact the technical support
35014	Suitable for Fault without Other Emergencies	Internal error of drive	Try to restart. If the problem is not solved, please contact the technical support
35015	Node Error	Internal error of drive	Try to restart. If the problem is not solved, please contact the technical support
35016	Command Error of Drive	1. This error is not accurate and can be ignored; 2. There is no PWM or other command signals	Recover to input signal
35100	Under-Voltage of the Drive DC Bus	The servo detects that the bus voltage is less than the set under-voltage protection threshold in real time possibly due to insufficient bus supply voltage	1. Check whether the power supply voltage of the robot is normal; 2. Check whether the drive capacitance is normal; 3. Replace the servo drive
35101	Over-Voltage of Drive DC Bus	The servo detects that the bus voltage is higher than the set over-voltage protection threshold in real time, possibly because: 1. The bus power supply voltage is too high; 2. The robot decelerates too quickly; 3. The handling of base power supply is abnormal	1. Check the power supply voltage of the robot; 2. Check the deceleration of the robot; 3. Check the robot power management loop
35102	Over-Current of Drive Motor	The amplitude of servo real-time detection current vector is larger than the set safety protection threshold, possibly because the current is raised up due to the sudden stop of the motor during operation	1. Check whether the three-phase line and 48V power line of the motor are connected correctly; 2. Check whether the motor deflection angle is correct
35103	Over-Load of Drive Motor	The joint motor continuously exceeds the rated torque protection threshold range, possibly because of the servo closed-loop control. If the actual position of the motor cannot track the given position, the torque current will be too large, and the error will be reported when the duration reaches a certain degree	1. Check whether the three-phase line and band-type brake cable of the motor are connected correctly; 2. Check whether the motor deflection angle is correct
35104	Drive Motor Magnetic over Limit	Determine whether the excitation	Update the servo drive to the latest

		current amplitude is less than the set value, otherwise, an error will be reported. For field weakening control, it is necessary to change the excitation current to reach a higher speed. At present, the bus voltage of the motor adapted by low-voltage servo is enough, without requiring field weakening, so this error report has been blocked	firmware version and update the servo drive to the latest parameter version
35105	Stall Alarm of Drive	If the joint speed tracking error is greater than the set threshold and lasts for more than 1s, an error will be reported. Stall is reported in the start-up stage, and the motor needs to find the deflection angle again for normal operation. When stall is reported during motor operation, it is necessary to check hardware configuration, software version and parameters	1. Check whether the three-phase line and band-type brake cable of the motor are connected correctly; 2. Check whether the motor deflection angle is correct; 3. Check the hardware configuration; 4. Check the servo software version
35106	Out of Tolerance of Drive Position	The joint position tracking error exceeds the set threshold, so it is necessary to check whether the hardware and software are configured correctly with suitable version	1. Check whether the wiring harness is connected correctly; 2. Check whether the joint encoder is normal; 3. Replace joint hardware, motor, reducer and drive; 4. Confirm whether the instructions issued by the master station are reasonable
35107	Zero Current Alarm of the Drive	The zero current value needs to exceed the set threshold, and the current is abnormal when it is not enabled	Update the servo drive to the latest firmware version, update the servo drive to the latest parameter version and replace the servo drive
35108	Phase A Over-Current of the Drive	1. The phase A current of the motor exceeds the sampling range of the drive current; 2. The locked rotor for motor; 3. The load is too large or the speed is too fast; 4. The resistance of joint mechanism is large	1. First check whether the harness is connected correctly; 2. Check whether the motor deflection angle is correct; 3. Check whether the band-type brake is opened correctly when enabling; 4. When the load is too much, reduce the speed and check whether the fault disappears; 5. Observe whether the fault joint during operation makes abnormal noise due to excessive resistance
35109	Phase B Over-Current of the	1. The phase B current of the motor	1. First check whether the harness is

	Drive	exceeds the sampling range of the drive current; 2. The locked rotor for motor; 3. The load is too much or the speed is too fast; 4. The resistance of joint mechanism is high	connected correctly; 2. Check whether the motor deflection angle is correct; 3. Check whether the band-type brake is opened correctly when enabling; 4. When the load is too much, reduce the speed and check whether the fault disappears; 5. Observe whether the fault joint during operation makes abnormal noise due to excessive resistance
35110	Phase C Over-Current of the Drive	1. The phase C current of the motor exceeds the sampling range of the drive current; 2. The locked rotor for motor; 3. The load is too much or the speed is too fast; 4. The resistance of joint mechanism is high	1. First check whether the harness is connected correctly; 2. Check whether the motor deflection angle is correct; 3. Check whether the band-type brake is opened correctly when enabling; 4. When the load is too much, reduce the speed and check whether the fault disappears; 5. Observe whether the fault joint during operation makes abnormal noise due to excessive resistance
35111	IGBT Over-Current of the Drive	1. The motor current exceeds the set threshold; 2. The locked rotor for motor; 3. The load is too much or the speed is too fast; 4. The resistance of joint mechanism is high	1. First check whether the harness is connected correctly; 2. Check whether the motor deflection angle is correct; 3. Check whether the band-type brake is opened correctly when enabling; 4. When the load is too much, reduce the speed and check whether the fault disappears; 5. Observe whether the fault joint during operation makes abnormal noise due to excessive resistance
35112	Object Overflow of the Drive		
35113	Image Overflow of the Drive		
35114	Thread H Timeout of the Drive		
35115	Thread M Timeout of the Drive	The servo software scheduling is abnormal, and the program has not been completed within the specified time, possibly because the firmware version or parameter version is not the latest one	Update the servo drive to the latest firmware version, update the servo drive to the latest parameter version and replace the servo drive
35116	Thread L Timeout of the Drive		
35117	Thread C Timeout of the Drive		
35118	Interrupt and Crash of the Drive	The servo software scheduling is	Update the servo drive to the latest

		abnormal, and the program has not been completed within the specified time, possibly because the firmware version or parameter version is not the latest one	firmware version, update the servo drive to the latest parameter version and replace the servo drive
35119	Main Task Crash of the Drive		
35120	Illegal Hardware of the Drive		
35121	Timeout in Debug Version of the Drive		
35122	Object ID Timeout of the Drive		
35123	Thread ID Error of the Drive		
35124	PV Variable ID Error of the Drive		
35125	VV Variable ID Error of the Drive		
35126	VU Variable ID Error of the Drive		
35127	Data Type Error of the Drive		
35128	Existence of Data Object of the Drive		
35129	Non Existence of Data Queue of the Drive		
35130	Full Data Queue of the Drive		
35131	Empty Data Queue of the Drive		
35132	Application Version Error of the Drive		
35133	Parameter Verification Error of the Drive		
35134	Parameter Number Error of the Drive		
35135	PN Parameter ID Error of the Drive		
35136	UN Parameter ID Error of the Drive		
35137	FN Parameter ID Error of the Drive		
35138	SPI Communication Error of the Drive	The communication between DSP and parameter memory is abnormal, possibly due to the firmware version or the parameter version is not the latest one; or the servo drive hardware is abnormal	Update the servo drive to the latest firmware version, update the servo drive to the latest parameter version and replace the servo drive
35139	E2P Overflow of the Drive	Too many parameters are read by the parameter memory, possibly due to	Update the servo drive to the latest firmware version, update the servo

		the firmware version or the parameter version is not the latest one; or the servo drive hardware is abnormal	drive to the latest parameter version and replace the servo drive
35140	Empty Dynamic Memory of the Drive	Logic error occurs in the servo software operation, possibly due to the firmware version or the parameter version is not the latest one	Update the servo drive to the latest firmware version, update the servo drive to the latest parameter version and replace the servo drive
35141	Instruction Code Error of the Drive	Logic error occurs in the servo software operation, possibly due to the firmware version or the parameter version is not the latest one	Update the servo drive to the latest firmware version, update the servo drive to the latest parameter version and replace the servo drive
35142	Instruction Length Error of the Drive	Logic error occurs in the servo software operation, possibly due to the firmware version or the parameter version is not the latest one	Update the servo drive to the latest firmware version, update the servo drive to the latest parameter version and replace the servo drive
35143	Meter Channel Number Error Of the Drive		
35144	OSCI Channel Number Error Of the Drive		
35145	Static Mode Error of the Drive	Error occurs in the servo operation mode, possibly due to the servo firmware and parameters are not the latest version; or the servo parameters are not successfully written	Update the servo drive to the latest firmware version and replace the servo drive
35146	Dynamic Mode Error of the Drive	Error occurs in the servo operation mode, possibly due to the servo firmware and parameters are not the latest version; or the servo parameters are not successfully written	Update the servo drive to the latest firmware version and replace the servo drive
35147	Mode Change Error of the Drive	Error occurs in the servo operation mode, possibly due to the servo firmware and parameters are not the latest version; or the servo parameters are not successfully written	Update the servo drive to the latest firmware version and replace the servo drive
35148	IPM Alarm of the Drive	It is possibly because the servo firmware and parameters are not the latest version	Update the servo drive to the latest firmware version and replace the servo drive

35149	Module Overheating Alarm of the Drive	The joint temperature exceeds the temperature range for safe and reliable operation	Carry out corresponding cooling treatment
35150	Disconnection of the STO Switch of the Drive	The servo has been detecting the STO signal level, which is inconsistent with the setting logic, so an error is reported, possibly due to the robot enabling handle is not effective	1. Press the manual enabling handle; 2. Re-plug the STO cable; 3. Replace the STO cable and the servo drive
35151	Encoder Error (0x7371)	The QEP encoder reports an error and the ABZ cable is disconnected, possibly due to the servo has not been upgraded to the latest firmware and parameter version	Update the servo drive to the latest firmware version and update the servo drive to the latest parameter version
35152	Encoder Error (0x7372)		
35153	Encoder Error (0x7373)		
35154	Encoder Error (0x7374)		
35155	Encoder Error (0x7375)		
35156	Encoder Error (0x7376)		
35157	Encoder Error (0x7377)		
35158	Encoder Error (0x7378)	Encoder chip reports an error: Biss2 communication abnormal	1. Check the joint encoder cable; 2. Replace the encoder; 3. Replace the encoder cable; 4. Replace the servo drive
35159	Encoder Error (0x7379)	Encoder chip reports an error: Biss1 communication is abnormal, and the encoder decoding chip cannot be found	1. Check the encoder cable at the joint motor side; 2. Replace the encoder; 3. Replace the encoder cable; 4. Replace the servo drive
35160	Encoder Error (0x737A)	1. CRC error of encoder communication; 2. Encoder communication is abnormal	1. Re-plug the encoder cable; 2. Replace the encoder; 3. Replace the encoder cable; 4. Replace the servo drive
35161	Encoder Error (0x737B)	The encoder communication is abnormal: CRC verification of encoder passed, but encoder chip reported an error	1. Re-plug the encoder cable; 2. Replace the encoder; 3. Replace the encoder cable; 4. Replace the servo drive
35162	Encoder Error (0x737C)	1. Single encoder hardware fault; 2. The encoder battery is exhausted.	1. If a single encoder reports an error and cannot be recovered after restart, it is necessary to contact after-sale maintenance. 2. All axes report errors at the same time. Contact the after-sale maintenance to replace the encoder battery.
35163	TZ Trigger		

35164	Internal Error (0xFF91)		
35165	Upper Enabling Current Loop is Set as 0 (0xFFA2)		
35166	Unable to Track the Master Station Position Command (0x8612)		
35201	5v of Over-Current of Secondary Encoder		CDHD can provide a maximum current of 250 mA to the secondary encoder. Check whether the encoder is short-circuited, and whether the encoder drawing exceeds the current limit
35202	Over-Current	1. Drive or motor fault; 2. The robot load exceeds the limit; 3. Send collision, etc.	Check whether the motor connection is short-circuited and whether there is overshoot in the current circuit
35203	Foldback of the Motor	1. Drive or motor fault; 2. The robot load exceeds the limit; 3. Send collision, etc.	Check the specifications of the drive-motor. This fault may occur if the capacity (power) of the motor is too small.
35204	Foldback of the Drive		Check the specification of the drive-motor. This alarm may occur if the capacity (power) of the drive is too small. Check whether the commutation angle is correct (i.e., whether the rectification is balanced)
35205	Invalid Current Sensor Compensation	The drive malfunctions	Restart. If the fault still exists, the driver may need to be repaired. Please contact the technical support
35206	Disconnection of Motor Phase	The drive malfunctions	Check the wiring of the motor phase
35207	Output Over-Current Detection	The drive malfunctions	Verify the correct wiring of the digital output and ensure that the output circuit is not shorted.
35208	Over-Voltage		Check whether regenerative resistance is required
35209	Under-Voltage Fault		Check whether the main AC voltage power supply is connected to the drive and turned on. The under-voltage limit can be read by the UVTHRESH command
35210	Regenerative Over-Current		Increase the regenerative resistance
35211	STO Fault		Check whether the STO connector (P1) is correctly connected
35212	Vbus Measurement Circuit Failure		Restart. If the fault still exists, the driver may need to be repaired.

			Please contact the technical support
35213	Bus AC Power Off		Please check the wiring or contact the technical support
35214	Regenerative Resistance Overload		Check whether regenerative resistance characteristics are suitable for this application
35215	Overheating of Integrated Power Module		Check whether the ambient temperature exceeds the drive specification. Otherwise contact the technical support.
35216	Control Panel Overheating		Check whether the ambient temperature exceeds the drive specification. Otherwise contact the technical support.
35217	Temperature Sensor Failure		Restart the power supply. If the problem persists, please contact the technical support
35218	Power Level Overheating		Check whether the ambient temperature exceeds the drive specification. Otherwise contact the technical support.
35219	Motor Overheating Fault		Check whether the drive is properly configured (using THERMODE,THERMTYPE,THERMTHRESH and THERMTIME), and if necessary, check whether the motor temperature sensor is properly connected to the drive. If the drive is correctly configured and wired, check whether the motor specification is too small.
35220	Internal Power Supply Out of Range		The drive may need to be repaired. Please contact the technical support
35221	Out of Range for 5v		It may happen during power failure. In other cases, please contact the technical support.
35222	Power EEPROM Fault		Contact technical support.
35223	Control EEPROM Fault		Contact technical support.
35224	CAN Supply Fault		The driver may need to be repaired, please contact the technical support.
35225	Self-Test Failure		Contact technical support.
35226	Parameter Memory Verification Failure		Reconfigure the drive, or download the parameter set and save the parameters. If the problem persists,

			please contact the technical support.
35227	Writing to Flash Memory Failure		Contact technical support.
35228	Fieldbus Speed Exceeds the Limit		Enable the drive and send a valid position command
35229	Not Configured		Execute CONFIG after setting drive parameters
35230	FPGA Config Failed		Contact technical support.
35231	Motor Setup Failed		Check phase and motor wiring. Make sure the correct feedback type is selected. Check MOTORSETUPST for hints.
35232	Phase Find Failed		Check whether the motor feedback type and the phase-finding parameters are set correctly for the application.
35233	FPGA Version Mismatch		Update either FPGA version or driver version.
35234	Emergency Stop Issued		Turn off the input.
35235	Fieldbus Version Mismatch		Make sure the correct version has been downloaded to the drive.
35236	ESI Version Mismatch		Make sure the correct version has been downloaded to the drive.
35237	BiSS-C Encoder Internal Error		Refer to BiSS-C Encoder User Manual.
35238	HIPERFACE Encoder Data Error		Enter the command HSAVE 1 using the drive software.
35239	ESI Manufacturer Mismatch		Make sure the correct manufacturer data has been downloaded to the drive.
35240	Index Line Break		Check whether the drive is configured for working with the index signal (using MENCTYPE), and check if the index signal is connected.
35241	Power Brake Load is Open		
35242	Short Circuit of the Power Brake		
35243	Stall Fault		Remove the stall condition, and take care to prevent stall conditions.
35244	Secondary Feedback Index Break		Check whether the drive is configured for working with the index signal on the secondary encoder, and check if the index signal is connected.
35245	Secondary Feedback A/B Line		Check whether all signals from the

	Break		secondary encoder are properly connected to the drive.
35246	Pulse and Direction Input Line Break		Check whether all signals to the P&D inputs are properly connected to the drive.
35247	Power Brake Fault		Replace the motor brake.
35248	Motor Runaway Condition Detected		Correct MPHASE setting. Activate and improve the phase find process.
35249	Feedback Communication Error, Encoder Communication Disconnected	1. The circuit connection between the robot body and the control cabinet is poor. 2. The encoder or drive may be faulty	Check whether the feedback device is wired correctly. Check whether the correct encoder type (MENCTYPE) is selected.
35250	Nikon Encoder Operational Fault		Check whether the feedback device is wired correctly. Check whether the correct encoder type (MENCTYPE) is selected.
35251	Tamagawa Init Failed		Check whether the wiring to the encoder is correct.
35252	A/B Line Break		Refer to the section Sine Encoder and Resolver Diagnostics. Check whether all signals from the primary feedback device are properly connected to the drive.
35253	Invalid Halls		Check whether the Hall signals are all properly connected. While turning the motor, read the Halls state (using HALLS) to see which signal is not connected. If the feedback type is Tamagawa, check whether the feedback wiring is correct.
35254	Absolute Encoder Battery Low-Voltage	The battery is a consumable that must be replaced regularly. To purchase, please contact the manufacturer.	Replace the battery and then reset the drive. If the battery is replaced while the drive is on, the position information is retained.
35255	Phase-Locked Loop Synchronization Failed		Check for controller synchronization signal. Check the cable connection and wiring.
35256	Encoder Simulation Frequency Too High		Check the parameters used for setting up the equivalent encoder output. If using a sine encoder, check the ENCOUTRES parameter settings.
35257	Tamagawa Abs Operational Fault		Check the battery voltage and feedback wiring. Make sure the motor did not move at a high velocity

			during encoder initialization.
35258	Custom Absolute Encoder Operational Fault		Check the battery voltage and feedback wiring. Make sure the motor did not move at a high velocity during encoder initialization.
35259	Differential Halls Line Break		Make sure HALLSTYPE matches the Hall sensors in use (single-ended or differential). Check whether all signals from the differential Hall sensors are properly connected to the sensor.
35260	Encoder Phase Error		Set MENCAQBFILT to 0 to remove the filter on A and B signals. If problem persists, it may be due to a faulty encoder.
35261	AqB Commutation Fault		If a fault occurs shortly after motion begins, check MENCRES settings. If a fault occurs after some time it is likely due to EMI noise. Improve the installation. Make sure ground is connected. Make sure shield is connected on feedback and motor cables.
35262	SensAR Encoder Fault		Use command SRVSNSINFO to identify the fault.
35263	Sine Feedback Communication Fail		Check whether the data and clock signals to the EnDat encoder are connected properly. The cable must be shielded.
35264	A/B Out of Range		Refer to the section Sine Encoder and Resolver Diagnostics. Check the amplitudes of the sine and cosine signals.
35265	Sankyo Absolute Encoder Fault		Check the battery voltage and feedback wiring. Make sure the motor did not move at a high velocity during encoder initialization.
35266	Sine Encoder Quadrature Fault		Check the feedback device wiring. Make sure the correct encoder type (MENCTYPE) is selected.
35267	Sin/Cos Calibration Invalid		Re-execute the sine/cosine calibration process.
35268	Feedback 5V Over-Current		The CDHD can source a maximum current of 250 mA to the primary

			encoder. Check for short-circuit at the encoder. Check if the encoder is drawing more than the current limit.
35269	Resolver Initialization Failed		Check resolver wiring and gain value.
35270	Endat2X Feedback Fault		Reset the encoder including encoder power off.
35271	Fieldbus Cable Disconnected		Reestablish the connection between controller and drive.
35272	Fieldbus Control Command Lost		Clear the fault and allow the controller to send new commands.
35273	CAN Heartbeat Lost		Reconnect master and slave, and power cycle the drive.
35274	Drive Locked		Contact technical support.
35275	EtherCAT Packet Loss		Make sure the EtherCAT master (controller) sends the packets within the time defined (by the master).
35276	Torque Feedback Out of Limit		
35277	Unstable Current Loop		Check and modify current controller settings.
35278	Velocity Over-Speed Exceeded		Check whether VLIM is set to match the application requirements. Using velocity loop tuning, check for excessive overshoot.
35279	Exceeded Maximum Velocity Error		Change drive tuning to improve velocity tracking, or increase VEMAX to allow a greater velocity error.
35280	Exceeded Maximum Position Error		Change drive tuning to improve position tracking, or increase PEMAX to allow a greater position error.
35281	Secondary Feedback Position Mismatch		Increase SFBPETHRESH, STBPETIME.SFBPEMAX, or improve position tuning.
35282	Excessive PE Value		Check tuning.
35283	CAN/EtherCAT State Not Operational		Make sure the controller does not switch to a lower state of communication while the drive is disabled.
35284	Internal Error		Contact technical support.
35285	Motor Plate Read Failed		Reconnect the feedback device. Make sure the motor type nameplate data is present.
35286	SAVE and Power Cycle Required		SAVE and then cycle power to the

			drive.
35287	Realtime Overload Fault		Contact technical support.
35288	PFB Off Checksum Invalid		If required by the application, home the machine.
35289	PFB Off Data Mismatch		If required by the application, home the machine.
35290	No PFB Off Data		If required by the application, home the machine.
35291	Pulse Train Frequency Too High		Reduce the frequency of the gearing pulses commanded from the controller.
35301	Short Circuit of the Drive	i. The U, V, and W outputs of the drive are short-circuited; ii. The drive is disturbed, which causes the DI signal to be abnormal. This is a false alarm; 1. The ground wire is not connected well; 2. The parameter setting of the current loop regulator is not suitable, causing current oscillation and interference; iii. The drive is damaged (such as IGBT short circuit, abnormity in detecting circuit by the current).	i. Check the U, V, and W wiring of the drive (for example, after disconnecting the motor power cable, observe whether the drive still reports a short-circuit fault. It must be done under the premise that the motor's band-type brake is disconnected to ensure mechanical safety); ii. Check the drive IGBT with a multimeter to confirm whether it is short-circuited; iii. Standardize the wiring, especially ground wire; iv. Adjust the parameters of the current loop; v. Replace the drive.
35302	Drive Output Short Circuited to Ground	i. The U, V, and W outputs of the drive are short circuited to ground; ii. The drive is damaged (such as abnormity in detecting circuit by the current).	i. Check the U, V, and W wiring of the drive; ii. Replace the drive.
35303	Abnormal Encoder Data	i. Hiperface encoder fault; ii. Encoder wiring error; iii. The internal AD calibration parameters of the drive are abnormal.	i. Replace the encoder; ii. Check the encoder wiring and ensure that it is correct; iii. Replace the drive.
35304	Rotor Positioning Error	i. The setting of position loop, velocity loop, and current loop regulator parameters is unreasonable; ii. The motor parameters are set incorrectly; iii. The parameter 0x20D2 is set too small; iv. The parameter 0x2003 is set incorrectly; v. The peripheral wiring of the drive is incorrect (such as the motor power cable and the motor encoder cable);	i. Check the peripheral wiring of the drive and ensure that it is correct. ii. Adjust the parameters of the position loop, velocity loop, and current loop regulator parameters of the drive, and ensure that the motor parameters are set correctly; iii. Re-detect the rotor position compensation angle; iv. Increase the setting value of parameter 0x20D2; v. Replace the

		vi. The internal circuit of the drive is abnormal; vii. The parameter 0x2207 static balance torque compensation value is not set properly.	drive; vi. Reduce the setting value of parameter 0x2207 static balance torque compensation value.
35305	Abnormal Motor's Band-Type Brake	i. The motor's band-type brake itself is abnormal; ii. When the motor is operating at high velocity, the servo suddenly turns OFF; iii. The servo parameters 0x2233 and 0x20D2 are set too small; iv. A short circuit occurs in the motor's band-type brake circuit.	i. Replace the motor's band-type brake; ii. Increase parameters 0x2233 and 0x20D2; iii. Check the motor's band-type brake circuit.
35306	EtherCAT PDO Configuration Error	i. PDO is configured incorrectly.	i. Correct the EtherCAT PDO configuration.
35307	Abnormal Encoder Internal Communication	a. A fault occurs in the encoder; b. The motor encoder wiring is abnormal (such as line break, the shielded twisted pair cable is not used, and it is coupled with the motor power cable); c. The ground wire of the drive is not reliably connected; d. Strong interference sources exist around the drive.	a. Check the wiring of the motor encoder and ensure that the wiring is standard and correct; b. Add magnetic rings to the encoder cable and the motor power cable; c. Reliably connect the ground wire of the drive; d. Replace the motor encoder; e. Remove the strong interference sources around the drive, or independently supply power to the drive and the surrounding strong interference sources; f. Add a line filter to the input power supply of the drive.
35308	Encoder Type Change	i. The encoder type has been changed.	i. Re-power on or soft reset the drive.
35309	Drive Phase U Over-Current	i. The parameter setting of the current loop regulator is unreasonable, leading to the current control oscillation; ii. The motor parameters are set incorrectly; iii. The internal current sampling circuit of the drive is abnormal.	i. Adjust the parameters of the current loop regulator; ii. Set motor parameters correctly; iii. Replace the drive.
35310	Drive Phase V Over-Current	i. The parameter setting of the current loop regulator is unreasonable, leading to the current control oscillation; ii. The motor parameters are set incorrectly; iii. The internal current sampling circuit of the drive is abnormal.	i. Adjust the parameters of the current loop regulator; ii. Set motor parameters correctly; iii. Replace the drive.

35311	Drive Phase W Over-Current	i. The parameter setting of the current loop regulator is unreasonable, leading to the current control oscillation; ii. The motor parameters are set incorrectly; iii. The internal current sampling circuit of the drive is abnormal.	i. Adjust the parameters of the current loop regulator; ii. Set motor parameters correctly; iii. Replace the drive.
35312	DC Bus Over-Voltage	i. The input power supply voltage of the drive is too high; ii. The dynamic braking energy is excessive when the motor stops quickly; 1. The deceleration is excessive when the motor stops; 2. The wiring of the dynamic braking resistor is incorrect; 3. The value of the dynamic braking resistance is too high; iii. The internal voltage sampling circuit of the drive is abnormal; iv. The internal dynamic braking circuit of the drive is abnormal.	i. Adjust the input power supply of the drive to the permissible range; ii. Reduce the motor deceleration when stopping; iii. Check the wiring of the dynamic braking resistor and ensure that it is correct; iv. Properly reduce the resistance of the dynamic braking resistor (the resistance shall not be lower than the minimum allowable value), and increase the power of the dynamic braking resistor.
35313	Control Power Supply Under-Voltage	i. The 24V control power supply is abnormal; ii. The wiring of the 24V control power supply is incorrect, such as poor wiring; iii. The load of the 24V control power supply is excessive; iv. The internal circuit of the drive is abnormal.	i. Check the wiring of the 24V control power supply and ensure that it is reliable; ii. Check the load of the 24V control power supply and ensure that the capacity of the 24V control power supply can meet the load consumption under all working conditions. iii. Replace the 24V control power supply; iv. Replace the drive.
35314	Drive Continuous Overload	i. The motor load is excessive; 1. The actual mechanical load is excessive; 2. There is jamming in the mechanical load; 3. The motor's band-type brake is not released; ii. The motor acceleration and deceleration time is too short; iii. The internal current sampling circuit of the drive is abnormal; iv. The band-type brake circuit of the drive is abnormal.	i. Reduce the actual mechanical load of the motor; ii. Increase the motor acceleration and deceleration time; iii. Check the transmission mode of mechanical load to ensure that there is no jamming or other abnormal phenomena; iv. Check the wiring of the motor's band-type brake to ensure reliable wiring; v. Replace the motor; vi. Replace the drive.
35315	Encoder Wiring Error	i. For CDA8 V1 products, pin 8 and pin 15 of the encoder terminal connector are not short-circuited; ii.	i. For CDA8 V1 products, pin 8 and pin 15 of the encoder terminal connector are short-circuited; for

		For CDA8 V2 products and CDR series products, pin 6 and pin 8 of the encoder terminal connector are not short-circuited; iii. The encoder cable is poorly wired.	CDA8 V2 products and CDR series products, pin 6 and pin 8 of the encoder terminal connector are short-circuited, and reliable encoder cable wiring shall be ensured.
35316	CPU Overload	i. The drive operation is disturbed; ii. The internal circuit of the drive is abnormal; iii. The data collection of DriveStarter is excessive.	i. Standardize the peripheral wiring of the drive and add anti-interference measures; ii. Replace the drive; iii. Close some DriveStarter data collection channels.
35317	Drive Output Phase Loss	i. Line break and poor wiring occur in the U, V, and W outputs of the drive; ii. The motor impedance is excessive; iii. The internal current sampling circuit of the drive is abnormal.	i. Check the U, V, and W wiring of the motor and ensure that it is reliable; ii. Replace the motor (or disable drive output phase loss detection); iii. Replace the drive.
35318	Instantaneous Overload of the Drive	i. The motor load is excessive; ii. The internal temperature sampling circuit of the drive is abnormal; iii. The operating environment temperature of the drive exceeds the permissible operating range; iv. The drive operation is disturbed (such as out of sync); v. The motor acceleration and deceleration are set too large, and the acceleration and deceleration time are set too short.	i. Reduce the actual mechanical load of the motor; ii. Standardize the peripheral wiring of the drive and add anti-interference measures; iii. Reduce the ambient temperature, such as improving the cooling conditions of the cabinet; iv. Replace the drive.
35319	Abnormal External Communication Transmission of the Encoder	a. The motor encoder wiring is abnormal (such as line break, the shielded twisted pair cable is not used, and it is coupled with the motor power cable); b. The ground wire of the drive is not reliably connected; d. Strong interference sources exist around the drive.	a. Check the wiring of the motor encoder and ensure that the wiring is standard and correct; b. Add magnetic rings to the encoder cable and the motor power cable; c. Reliably connect the ground wire of the drive; d. Remove the strong interference sources around the drive, or independently supply power to the drive and the surrounding strong interference sources; e. Add a line filter to the input power supply of the drive.
35320	Abnormal External Communication Reception of the Encoder	a. The motor encoder wiring is abnormal (such as line break, the shielded twisted pair cable is not used, and it is coupled with the	a. Check the wiring of the motor encoder and ensure that the wiring is standard and correct; b. Add magnetic rings to the encoder cable

		motor power cable); b. The ground wire of the drive is not reliably connected; d. Strong interference sources exist around the drive.	and the motor power cable; c. Reliably connect the ground wire of the drive; d. Remove the strong interference sources around the drive, or independently supply power to the drive and the surrounding strong interference sources; e. Add a line filter to the input power supply of the drive.
35321	Drive Hardware Over-Current	i. There is jamming or locking in the mechanical load; ii. The rotor compensation angle is set incorrectly; iii. Encoder wiring error; iv. The current loop regulator parameters are set unreasonably, resulting in current control oscillation; v. Motor parameter setting error (wire resistance, wire inductance, counter electromotive force, etc.); vi. The internal current detection circuit of the drive is abnormal; vii. The drive's band-type brake circuit is damaged, without 24V output; viii. The motor's band-type brake is damaged; ix. The motor load is too large, or the motor acceleration and deceleration are set too large, and the acceleration and deceleration time are set too short; x. The setting of 0x60B2 torque compensation value or 0x2207 static balance torque compensation value is unreasonable.	i. Check the transmission mode of mechanical load to ensure that there is no jamming or other abnormal phenomena; ii. Re-detect the rotor compensation angle; iii. Check the wiring of the motor encoder and ensure that the wiring is standard and correct; iv. Adjust the parameters of the current loop regulator; v. Set motor parameters correctly; vi. Replace the drive; vii. Replace the motor; vii. Check the transmission mode of mechanical load to ensure that there is no jamming or other abnormal phenomena; viii. Confirm whether the mechanical design is reasonable; optimize motor acceleration and deceleration, and extend the acceleration and deceleration time; ix. Optimize the dynamics model of the upper controller and the torque compensation value of the given 0x60B2; or reasonably reset the 0x2207 static balance torque compensation value.
35322	Abnormal Band-Type Brake Circuit of the Drive	i. The drive's band-type brake output is short circuited; ii. Excessive band-type brake output current of the drive causes over-temperature; iii. The drive's band-type brake output is open circuit; iv. The internal detection circuit of the drive is abnormal.	i. Check the wiring of the drive's band-type brake output and ensure it is correct and reliable; ii. Replace the drive.
35323	Abnormal Resolver Circuit of the	i. The internal resolver circuit of the	i. Correctly set the resolver

	Drive	drive is abnormal; ii. The resolver parameter setting of the drive does not match the actual resolver.	parameters of the drive; ii. Replace the drive.
35324	Control Mode Setting Error	i. When the servo is enabled, the controller sets the control mode that the drive does not support (see the object dictionary 0x6502 for the control mode supported by each product).	i. Before enabling the servo, set the controller to the correct control mode.
35325	Input Phase Loss Fault	i. The input power supply of the drive is poorly wired; ii. The drive servo parameter "power circuit setting" is set to three-phase input, but the actual power supply input is single-phase; iii. The front end uses an electronic transformer, which has abnormal harmonics and cannot be identified by the servo.	i. Check the input power wiring of the drive and ensure that it is reliable; ii. Set the servo parameter "power circuit setting" correctly; iii. Add a filter to the front end of the servo drive.
35326	DC Bus Under-Voltage	i. The input power supply voltage of the drive is too low; ii. The internal voltage sampling circuit of the drive is abnormal; iii. Servo parameter 0x202C is set incorrectly, 220V power supply is set as 380V power supply; iv. The input power cord of the drive is disconnected.	i. Adjust the input power supply of the drive to the permissible range; ii. Replace the drive; iii. Set 0x202C drive parameters correctly; iv. Check the input power cord wiring of the drive.
35327	Inverter Power Module Over-Temperature	i. The motor load is excessive; ii. The internal temperature sampling circuit of the drive is abnormal; iii. The operating environment temperature of the drive exceeds the permissible operating range.	i. Reduce the actual mechanical load of the motor; ii. Reduce the ambient temperature, such as improving the cooling conditions of the cabinet; iii. Replace the drive.
35328	Dynamic Braking Overload	i. The motor performs frequent quick stop operations, resulting in excessive dynamic braking energy; ii. The servo parameters "resistance of the dynamic braking resistor" and "power of the dynamic braking resistor" are set incorrectly.	i. Correctly set the servo parameters "resistance of the dynamic braking resistor" and "power of the dynamic braking resistor"; ii. Change the operating conditions of the motor to avoid frequent quick stop operations of the motor, such as extending the stop time of the motor.
35329	Continuous Overload of Motor	i. The motor load is excessive; 1. The actual mechanical load is excessive; 2. There is jamming in the mechanical load; 3. The motor's	i. Reduce the actual mechanical load of the motor; ii. Increase the acceleration and deceleration time when the motor is running; iii. Check

		band-type brake is not released; ii. The motor acceleration and deceleration time is too short; iii. Motor parameters are set incorrectly; iv. The internal current sampling circuit of the drive is abnormal; v. The band-type brake circuit of the drive is abnormal; vi. The selected motor type is not suitable and the power is too small (for example, high-power drives with small-power motors, which operate at high velocity with full load for a long time).	the transmission mode of mechanical load to ensure that there is no jamming or other abnormal phenomena; iv. Check the wiring of the motor's band-type brake to ensure reliable wiring; v. Check the motor parameters to ensure that they are set correctly (such as the rated current and thermal time constant of the motor); vi. Replace with a high-capacity motor; vii. Replace the drive.
35330	Rectifier Power Module Over-Temperature	i. The motor load is excessive; ii. The internal temperature sampling circuit of the drive is abnormal; iii. The operating environment temperature of the drive exceeds the permissible operating range	i. Reduce the actual mechanical load of the motor; ii. Reduce the ambient temperature, e.g. improve the heat dissipation conditions of the cabinet; iii. Replace the drive.
35331	Motor U – Phase Instantaneous Overload	i. The motor load is excessive; 1. The actual mechanical load is excessive; 2. The mechanical load leads to jamming or locking; 3. The motor's band-type brake is not released; ii. The motor acceleration and deceleration time is too short; iii. The rotor offset angle is set incorrectly; iv. The motor parameter is set incorrectly. v. The internal current sampling circuit of the drive is abnormal; vi. The drive's band-type brake circuit is abnormal; vii. Inadequate capacity of the motor model selected; viii. Poor or detached contact of one phase of the motor power line.	i. Reduce the actual mechanical load of the motor; ii. Increase the acceleration and deceleration time durations during motor operation; iii. Check the transmission mode of mechanical load to ensure that there is no jamming or other phenomena; iv. Re-check the rotor offset angle; v. Check the wiring of the motor's band-type brake to ensure reliable wiring; vi. Check the motor parameters to ensure that they are set correctly (such as the rated current, fast overload protection threshold, and fast overload protection time duration of the motor); vii. Change to a high – capacity motor; viii. Replace the drive; ix. Check whether the wiring of motor power line is reliable.
35332	Motor V – Phase Instantaneous Overload	i. The motor load is excessive; 1. The actual mechanical load is excessive; 2. The mechanical load leads to jamming or locking; 3. The motor's band-type brake is not released; ii.	i. Reduce the actual mechanical load of the motor; ii. Increase the acceleration and deceleration time durations during motor operation; iii. Check the transmission mode of

		The motor acceleration and deceleration time is too short; iii. The rotor offset angle is set incorrectly; iv. The motor parameter is set incorrectly. v. The internal current sampling circuit of the drive is abnormal; vi. The drive's band-type brake circuit is abnormal; vii. Inadequate capacity of the motor model selected; viii. Poor or detached contact of one phase of the motor power line.	mechanical load to ensure that there is no jamming or other phenomena; iv. Re-check the rotor offset angle; v. Check the wiring of the motor's band-type brake to ensure reliable wiring; vi. Check the motor parameters to ensure that they are set correctly (such as the rated current, fast overload protection threshold, and fast overload protection time duration of the motor); vii. Change to a high－capacity motor; viii. Replace the drive; ix. Check whether the wiring of motor power line is reliable.
35333	Motor W－Phase Instantaneous Overload	i. The motor load is excessive; 1. The actual mechanical load is excessive; 2. The mechanical load leads to jamming or locking; 3. The motor's band-type brake is not released; ii. The motor acceleration and deceleration time is too short; iii. The rotor offset angle is set incorrectly; iv. The motor parameter is set incorrectly. v. The internal current sampling circuit of the drive is abnormal; vi. The drive's band-type brake circuit is abnormal; vii. Inadequate capacity of the motor model selected; viii. Poor or detached contact of one phase of the motor power line.	i. Reduce the actual mechanical load of the motor; ii. Increase the acceleration and deceleration time durations during motor operation; iii. Check the transmission mode of mechanical load to ensure that there is no jamming or other phenomena; iv. Re-check the rotor offset angle; v. Check the wiring of the motor's band-type brake to ensure reliable wiring; vi. Check the motor parameters to ensure that they are set correctly (such as the rated current, fast overload protection threshold, and fast overload protection time duration of the motor); vii. Change to a high－capacity motor; viii. Replace the drive; ix. Check whether the wiring of motor power line is reliable.
35334	Abnormal Communication of the Power Supply Unit Module	i. Poor connection of the control cable between the power supply unit module and the motor module; ii. The control port between the power supply module and the motor module is burnt due to short circuit of the motor's band-type brake cable or external 24V (STO) to ground.	i. Check the control cable connection between the power supply unit module and the motor module and ensure the wiring is reliable; ii Check for short circuit to ground on the motor's band-type brake cable or external 24V (STO); iii. Replace the drive.
35335	Hardware STO1 Triggered	i. STO1 is triggered or is poorly wired.	i. Check STO1 wiring and make sure the wiring is reliable; ii. Verify that the STO1 circuit (e.g. emergency stop

			switch) is not triggered.
35336	Hardware STO2 Triggered	i. STO2 is triggered or is poorly wired.	i. Check the STO2 wiring and make sure the wiring is reliable; ii. Verify that the STO2 circuit (e.g. emergency stop switch) is not triggered.
35337	Abnormal STO Wiring	i. STO1/STO2 is poorly wired.	i. Check STO1/STO2 wiring and make sure the wiring is reliable.
35338	Drive External Fault	i. Fault of other axes. ii. Abnormal internal circuit of the drive.	i. Check other axes and make sure they are free of fault; ii. Such fault can be disabled by modifying the servo parameter "Fault operation switch"; iii. Replace the drive.
35339	Excessive Position Tracking Error	i. The mechanical load of the motor leads to jamming or locking, so that the motor cannot operate; ii. The planned acceleration for the target position value of the host computer is too high; iii. The servo parameters 0x6065 and 0x6066 are too small; iv. Unreasonable setting of the drive regulator parameters leads to unsatisfactory position tracking performance; v. The internal circuit of the drive is abnormal.	i. Check the transmission mode of mechanical load to ensure that there is no jamming or other issue; ii. Appropriately reduce the planned acceleration for the target position value of the host computer; iii. Appropriately increase the setting value of the servo parameters 0x6065 and 0x6066; iv. Optimize regulator parameters to improve position tracking performance; v. Replace the drive.
35340	Position Control Overflow	i. The actual or target value of the position exceeds the maximum permissible range.	i. Execute the "encoder multi-turn zeroing" command and ensure that the motor operating range does not exceed the maximum permissible range; ii. If it is necessary to operate the motor in a large range, the unlimited position control mode can be enabled through the servo parameter "position control switch".
35341	Excessive Velocity Tracking Error	i. The mechanical load of the motor leads to jamming or locking, so that the motor cannot operate; ii. The setting values of the servo parameters 0x20A3 and 0x20A4 are too small; iii. Unreasonable setting of the drive regulator parameters leads to unsatisfactory velocity tracking performance; iv. The internal circuit of the drive is abnormal.	i. Check the transmission mode of mechanical load to ensure that there is no jamming or other phenomena; ii. Appropriately increase the setting values of the servo parameters 0x20A3 and 0x20A4; iii. Optimize the regulator parameters to improve velocity tracking performance; iv. Replace the drive.

35342	Control Cycle Parameter Setting Error	i. Unreasonable setting of the EtherCAT communication cycle, position control cycle and velocity control cycle.	i. Set the EtherCAT communication cycle, position control cycle and velocity control cycle correctly.
35343	EEPROM Writing Failed	i. The internal circuit of the drive is abnormal; ii. The drive is interfered	i. Replace the drive; ii. Re-power on the drive and improve the anti-interference measures of the drive.
35344	Origin Searching Failed	i. Unreasonable parameter setting of origin searching (objects 0x6098, 0x6099, 0x609A); ii. The motor is already in the limit switch trigger state when the origin searching is activated; iii. Switch to non-HM mode during origin searching.	i. Set the origin searching parameters correctly (objects 0x6098, 0x6099, 0x609A); ii. Ensure that the motor is not in the limit switch trigger state when the origin searching is activated.
35345	Illegal EtherCAT Bus Instruction	i. The EtherCAT communication state machine is incorrectly matched with the control word time sequence.	i. The host computer correctly handles the EtherCAT communication state computer and control word time sequence.
35346	Abnormal DriveStarter Communication	i. The drive commissioning cable is disconnected or is poorly contacted. ii. The communication of the drive commissioning serial port is interfered.	i. Check the wiring of the drive commissioning cable and ensure reliable connection; ii. Replace isolated serial port commissioning cable; iii. Strengthen anti-interference measures for the commissioning cable, such as adding magnetic rings, reliable grounding of commissioning computer, and provide power supply for the debugging computer and the drive separately.
35347	Abnormal Communication of the EtherCAT Bus	i. The EtherCAT communication is interfered; ii. Disconnection or poor contact of EtherCAT network cables; iii. Insufficient real-time performance of the host computer; iv. Mismatch between the underlying DC synchronization mechanism of the EtherCAT master of the host computer and the drive requirements; v. The internal circuit of the drive is abnormal.	i. Optimize EtherCAT communication wiring and strengthen anti-interference measures, such as using Category 5E shielded twisted pair network cables and ensuring reliable grounding of the controller; ii. Check the connection of EtherCAT network cables to ensure reliable connection; iii. Change to the host computer with stronger real-time performance, or extend the EtherCAT communication cycle; iv. Appropriately increase the servo parameter 0x20D3 setting value; v.

			Modify the underlying DC synchronization mechanism of the EtherCAT master of the host computer to ensure that the RxPDO data sent by the host computer is at least 130 μs ahead of the DC synchronization signal; vi. Replace the drive.
35348	The Position Exceeds the Hardware Limitation	i. Limit switch input triggered.	i. Check the state of the limit switch and ensure that it is not triggered.
35349	Positive Software Limitation	i. The actual position value exceeds the threshold set by the servo parameters 0x2004 and 0x2005.	i. Appropriately increase the setting value of the servo parameters 0x2004 and 0x2005; ii. Operate the motor within the range specified by the servo parameters 0x2004 and 0x2005; iii. If users do not want to use this function, they can disable the software limitation detection function through the servo parameter "Position control switch".
35350	Negative Software Limitation	i. The actual position value exceeds the threshold set by the servo parameters 0x2004 and 0x2005.	i. Appropriately increase the setting value of the servo parameters 0x2004 and 0x2005; ii. Operate the motor within the range specified by the servo parameters 0x2004 and 0x2005; iii. If users do not want to use this function, they can disable the software limitation detection function through the servo parameter "Position control switch".
35351	Excessive Power-On Position Deviation	i. After the drive is powered off, the motor position is shifted; ii. For motor encoders with batteries, the external battery is unavailable or the battery is under-voltage.	i. For motor encoders with batteries, ensure the battery is connected and the battery voltage is normal; ii. If users don't want to use this function, they can set the servo parameter 0x200E to 0, and disable the detection function of excessive power-on position deviation.
35352	Power-On Position Control Overflow	i. For motor encoders with batteries, the external battery is unavailable or the battery is under-voltage; ii. The drive is powered off in any of the following control mode: the unlimited position control mode,	i. Re-power on the drive after the encoder multi-turn zeroing command is executed; ii. In the unlimited position control mode, if users do not want to use this function, they can modify the servo parameter "Position

		velocity mode, or torque mode, and the position has exceeded the permissible range.	control switch" to disable the power-on position control overflow detection function.
35353	Encoder Battery Under-Voltage Fault	i. The encoder is not connected with an external battery or the battery is poorly wired; ii. The encoder battery is under-voltage.	i. Check the battery wiring of the encoder to ensure reliable wiring; ii. Replace the battery; iii. If a Panasonic or Tamagawa encoder is connected, the encoder multi-turn zeroing command needs to be executed for individual versions; iv. If users do not want to use this function, they can modify the servo parameter 0x2009 to disable the encoder battery under-voltage detection function.
35354	The Motor Exceeds the Velocity Limit	i. Unreasonable setting of drive regulator parameters leads to large speed tracking overshoot; ii. Poor wiring of the encoder; iii. The encoder data transmission is interfered; iv. The encoder is damaged; v. The internal circuit of the drive is abnormal.	i. Optimize regulator parameters to improve speed tracking performance; ii. Check the encoder cable connection to ensure reliable wiring; iii. Strengthen the anti-interference measures of encoder cables, such as adding magnetic rings, using shielded twisted pair cables, and realizing reliable grounding; iv. Replace the encoder; v. Replace the drive.
35355	Excessive Voltage Limit Position Tracking Error	i. The mechanical load of the motor is jammed so that the motor cannot operate; ii. The planned acceleration of the host computer's target position value is too high; iii. The servo parameters 0x6065 and 0x6066 are too small. iv. Unreasonable setting of drive regulator parameters leads to unsatisfactory position tracking performance; v. The input power supply voltage of the drive is too low; vi. The internal circuit of the drive is abnormal.	i. Check the transmission mode of mechanical load to ensure that there is no jamming or other phenomena; ii. Appropriately reduce the planned acceleration of the host computer's target position value; iii. Appropriately increase the setting value of the servo parameters 0x6065 and 0x6066; iv. Optimize regulator parameters to improve position tracking performance; v. Ensure that the input power supply voltage of the drive is within the specified range; vi. Replace the drive.
35356	Encoder Over-Speed Fault	i. Unreasonable setting of drive regulator parameters leads to large speed tracking overshoot; ii. Poor wiring of the encoder; iii. The encoder data transmission is interfered; iv. The encoder is	i. Optimize regulator parameters to improve speed tracking performance; ii. Check the encoder cable connection to ensure reliable wiring; iii. Strengthen the anti-interference measures of encoder cables, such as

		damaged; v. The internal circuit of the drive is abnormal; vi. When the servo motor is enabled, an external force rotates the motor shaft.	adding magnetic rings, using shielded twisted pair cables, and realizing reliable grounding; iv. Replace the encoder; v. Replace the drive; vi. Check the mechanical load at the end of the motor shaft to ensure that the motor shaft is not subjected to gravity or external mechanical forces.
35357	Operation Error of Position Planning	i. Unreasonable setting of position planning parameters, such as the target position value and the planned target deceleration (0x6084).	i. Set position planning parameters correctly.
35358	Multi-Axis Synchronization Exception	i. The internal circuit of the drive is abnormal.	ii. Replace the drive.
35359	EtherCAT Bus Synchronization Exception	i. Unreasonable setting of the servo parameter 0x20D3; ii. The EtherCAT master synchronization mode is incorrectly configured.	i. Set the servo parameter 0x20D3 correctly; ii. Correctly configure the synchronization mode of the EtherCAT master.
35360	EEPROM Version Change	i. The drive firmware is upgraded.	i. Re-power on the drive.
35361	Motor Overload Alarm	i. The motor load is excessive; 1. The actual mechanical load is excessive; 2. The mechanical load leads to jamming; 3. The motor's band-type brake is not released; ii. The motor acceleration and deceleration time is too short; iii. The motor parameter is set incorrectly; iv. The internal current sampling circuit of the drive is abnormal; v. The drive ’ s band-type brake circuit is abnormal.	i. Reduce the actual mechanical load of the motor; ii. Prolong the acceleration and deceleration time durations during motor operation; iii. Check the transmission mode of mechanical load to ensure that there is no jamming or other phenomena; iv. Check the wiring of the motor's band-type brake to ensure reliable wiring; v. Check the motor parameters to ensure that they are set correctly (such as the rated current and thermal time constant of the motor); vi. Change to a high-capacity motor. Vii. Replace the drive.
35362	Speed Limit Alarm	i. In the speed mode, the target speed exceeds the maximum speed of the motor; ii. In the position mode, the planned speed exceeds the maximum speed of the motor; iii. Servo parameter setting is unreasonable.	i. Reduce the target or planned speed; ii. Change to a motor with higher maximum speed; iii. Reset the maximum planned speed of the parameter 0x607F according to the actual situation.
35363	DC Bus Under-Voltage Alarm	i. The input power supply voltage of the drive is too low; ii. The internal voltage sampling circuit of the drive is abnormal.	i. Adjust the input power supply of the drive to the permissible range; ii. Replace the drive.

35364	Control Mode Setting Unsupported	i. (Delete when the servo motor is enabled) The controller sets the control mode that the drive does not support (see the object dictionary 0x6060 and 0x6502 for the control mode supported by each product); ii. The control mode is not specified after the communication between the controller and the servo motor is established.	i. Before enabling the servo, set the controller to the correct control mode.
35365	Effective Parameters for Re-Power-On Changed	i. Servo parameters for re-power-on modified	i. Re-power on the drive
35366	Encoder Battery Under-Voltage Alarm	i. The encoder is not connected with an external battery or the battery is poorly wired; ii. The encoder battery is under-voltage; iii. The encoder battery cables are wired reversely or short circuited to ground due to the damaged sheath.	i. Check the battery wiring of the encoder to ensure reliable wiring; ii. Replace the battery; iii. If a Panasonic or Tamagawa encoder is connected, the encoder multiturn zeroing command needs to be executed for individual versions; iv. If users do not want to use this function, they can modify the servo parameter 0x2009 to disable the encoder battery under-voltage detection function.
35367	Drive Internal Alarm	i. The internal circuit of the drive is abnormal.	i. Replace the drive; ii. Contact the after-sales for technical support.
35368	Mechanical Zero Uncalibrated	i. The encoder battery failed due to under-voltage, and the servo parameter 0x2009.Byte3 is set to "Encoder battery under-voltage fault is detected and zero uncalibrated is informed"; ii. There is a fault of excessive deviation of the power-on position and the user determined that the mechanical zero is lost; iii. The motor features a single-turn absolute encoder and the driver failed to execute the origin searching instruction.	i. Zero the drive.
35369	Encoder receiving external communication alarm	a. The motor encoder wiring is abnormal (such as line break, the shielded twisted pair cable is not used, and it is coupled with the motor power cable); b. The ground wire of the drive is not reliably	Encoder receiving external communication alarm

		connected; d. Strong interference sources exist around the drive.	
35370	Encoder Sending External Communication Alarm	a. The motor encoder wiring is abnormal (such as line break, the shielded twisted pair cable is not used, and it is coupled with the motor power cable); b. The ground wire of the drive is not reliably connected; d. Strong interference sources exist around the drive.	a. Check the wiring of the motor encoder and ensure that the wiring is standard and correct; b. Add magnetic rings to the encoder cable and the motor power cable; c. Reliably connect the ground wire of the drive; d. Remove the strong interference sources around the drive, or independently supply power to the drive and the surrounding strong interference sources; e. Add a line filter to the input power supply of the drive.
35371	Encoder internal communication alarm	Encoder internal communication alarm	a. Check the wiring of the motor encoder and ensure that the wiring is standard and correct; b. Add magnetic rings to the encoder cable and the motor power cable; c. Reliably connect the ground wire of the drive; d. Replace the motor encoder; e. Remove the strong interference sources around the drive, or independently supply power to the drive and the surrounding strong interference sources; f. Add a line filter to the input power supply of the drive.
35372	Software Limitation Alarm	i. The actual or target position value exceeds the threshold set by the servo parameters 0x2004 and 0x2005.	i. Appropriately increase the setting value of the servo parameters 0x2004 and 0x2005; ii. Operate the motor within the range specified by the servo parameters 0x2004 and 0x2005; iii. Reduce the target position value so that it falls within the range specified by the servo parameters 0x2004 and 0x2005; iv. If users do not want to use this function, they can disable the software limitation detection function through the servo parameter "Position control switch".
35373	AD Correction Coefficient	i. The drive is not subjected to AD	i. Reset the drive AD correction

	Invalid Alarm	correction.	coefficient
35374	Abnormal position planning parameter alarm	i. Unreasonable setting of position planning parameters.	i. Set position planning parameters correctly.
35375	Excessive power-on position deviation alarm	i. The motor position deviates after the drive is powered off.	i. The drive executes the fault reset command; ii. Re-power on or perform the soft reset of the drive.
35401	Short Circuit of the Drive	1. The U/V/W output cable of the drive is short-circuited, or short-circuited to ground; 2. The U/V/W output cable of the motor is short-circuited, or short-circuited to ground; 3. An internal cable of the drive is short-circuited, or short-circuited to ground; 4. False alarm is caused because the drive is interfered	1. If a short circuit occurs between the cable's UVW phases, or between the cable's U/V/W and grounding, dispose or replace the cable; 2. If a short circuit occurs between the motor's UVW phases, or between the motor's U/V/W cable and grounding, replace the motor; 3. If faults still occur after disconnecting the drive U/V/W output wiring, replace the drive; 4. Improve the electromagnetic environment of the equipment by standardizing wiring and cabling, increasing the cross-sectional area of the grounding wire and adding magnetic rings.
35402	Excessive U-Phase Output Current	1. The parameter setting of the current loop regulator is unreasonable, leading to the current control oscillation; 2. The motor parameters are set incorrectly; 3. The internal current sampling circuit of the drive is abnormal.	1. Adjust the parameters of the current loop regulator; 2. Set motor parameters correctly; 3. Replace the drive
35403	Excessive V-Phase Output Current	1. The parameter setting of the current loop regulator is unreasonable, leading to the current control oscillation; 2. The motor parameters are set incorrectly; 3. The internal current sampling circuit of the drive is abnormal.	1. Adjust the parameters of the current loop regulator; 2. Set motor parameters correctly; 3. Replace the drive
35404	Excessive W-Phase Output Current	1. The parameter setting of the current loop regulator is unreasonable, leading to the current control oscillation; 2. The motor parameters are set incorrectly; 3. The internal current sampling circuit of the drive is abnormal.	1. Adjust the parameters of the current loop regulator; 2. Set motor parameters correctly; 3. Replace the drive
35405	Drive Hardware Over-Current	1. Excessive motor load or motor	1. Check and handle the mechanical

		acceleration and deceleration setting values, and too short acceleration and deceleration time durations set; 2. The rotor offset angle is set incorrectly, and fails to meet the rotor positioning error detection condition; 3. Abnormal jumps occur in the encoder feedback; 4. The current loop regulator parameters are set unreasonably, resulting in current control oscillation; 5. Motor parameters are incorrectly set (wire resistance, wire inductance, counter electromotive force, rotor inertia, etc.); 6. Current detection circuit inside the drive is abnormal or the drive's band-type brake circuit is damaged, without 24V output; 7. The motor's band-type brake is damaged; 8. The torque offset value or static balance offset value is not set properly.	load driving to ensure that there is no jamming or other phenomena; 2. Re-detect the rotor offset angle; 3. Check the motor encoder wiring and ensure that it is standard and correct; 4. Adjust current loop regulator parameters; 5. Set motor parameters correctly; 6. Replace the drive; 7. Replace the motor; 8. Optimize the host controller dynamics model, and optimize the given value or set the value according to the actual load.
35406	Drive Output Short Circuited to Ground	1. The drive U/V/W output cable is short-circuited to ground; 2. The motor U/V/W cable is short circuited to ground; 3. There is a short circuit inside the drive or a short circuit to ground.	1. If a short circuit occurs between the cable's U/V/W and grounding, repair or replace the cable; 2. If a short circuit occurs between the motor's U/V/W cable and grounding, replace the motor; 3. If faults still occur after disconnecting the drive U/V/W output wiring, replace the drive.
35407	DC Bus Over-Voltage	1. Excessive input power supply voltage of the drive; 2. Excessive dynamic braking energy when the motor stops quickly; 3. The dynamic braking resistor is not connected or wired incorrectly; 4. The over-high resistance value of the dynamic braking resistor; 5. Internal malfunction of the drive.	1. Adjust the input power supply of the drive to the permissible range; 2. Reduce the motor deceleration when stopping; 3. Correct the wiring of dynamic braking resistor to ensure correct wiring; 4. Reduce the resistance value of the dynamic braking resistor appropriately (the resistance value cannot be lower than the minimum permissible value) and increase the power of the dynamic braking resistor; 5. Replace the drive.
35408	DC Bus Under-Voltage	1. Excessively low input power	1. Adjust the input power supply of

		supply voltage of the drive; 2. Abnormal voltage sampling circuit inside the drive; 3. The drive power loop is set incorrectly, and the 220 V power supply is set to 380 V power supply; 4. The input power supply cord of the drive is disconnected.	the drive to the permissible range for normal working; 2. Replace the drive; 3. Set the drive power loop in consistent with the actual power supply; 4. Check and handle the wiring of the input power supply cord of the drive to ensure that the wiring is correct and secure.
35409	Power Module Over-Temperature	1. The motor load is excessive; 2. The internal temperature sampling circuit of the drive is abnormal; 3. The operating environment temperature of the drive exceeds the permissible range.	1. Reduce the actual mechanical load of the motor; 2. Replace the drive; 3. Reduce the ambient temperature, such as improving the heat dissipation conditions of the cabinet.
35410	CPU1 Watchdog Overflow	Internal malfunction of the drive.	Replace the drive.
35411	CPU2 Watchdog Overflow	Internal malfunction of the drive.	Replace the drive.
35412	Dynamic Braking Resistor Overload	1. The frequent quick shutdown of the motor leads to excessive dynamic braking energy; 2. The power setting of the dynamic braking resistor is inconsistent with that of the actual resistor.	1. Change the operating conditions of the motor to avoid frequent quick shutdown of the motor, such as extending the stop time of the motor. Or replace the dynamic braking resistor with one of a higher power; 2. Set the power of dynamic braking resistor correctly, with the value set in consistent with the actual power of the dynamic braking resistor.
35413	Continuous Overload of Motor	1. Excessive motor load; 2. Over-short acceleration and deceleration time durations of the motor; 3. Incorrect setting of motor parameters; 4. Abnormal release action of the band-type brake; 5. Wrong motor model with a smaller power (such as the high-power drive loaded with the small-power motor runs at full load and high speed for a long time); 6. Abnormal internal current sampling circuit of the drive.	1. Reduce the actual mechanical load of the motor to ensure that the machinery is not jammed; 2. Prolong the acceleration and deceleration time durations during motor operation; 3. Check the motor parameters to ensure that the motor parameters are set correctly (such as the rated current and thermal time constant of the motor); 4. Check the line of the band-type brake of the motor to ensure normal action of the band-type brake; 5. Change to a high－capacity motor; 6. Replace the drive
35414	Excessive Position Tracking Error	1. Excessive motor load; 2. Inappropriate control parameters; 3. Abnormal release action of the band-type brake; 4. Too small	1. Reduce the actual mechanical load of the motor to ensure that the machinery is not jammed; 2. Optimize the control parameters and

		threshold or time duration for judging excessive position tracking error.	enhance the corresponding performance of the servo; 3. Check the line of the band-type brake of the motor to ensure normal action of the band-type brake; 4. Appropriately increase the threshold or time duration for judging excessive position tracking error.
35415	Positive Software Limitation	Position feedback value exceeds (positive software limitation value + positioning completion threshold).	The range of motion should not exceed the setting value for positive software limitation. If the positive software limitation function is not needed, it can be prohibited by the parameter position control switch.
35416	Negative Software Limitation	Position feedback value exceeds (negative software limitation value - positioning completion threshold).	The range of motion should not exceed the setting value for negative software limitation. If the negative software limitation function is not needed, it can be prohibited by the parameter position control switch.
35417	Encoder Data Overflow	In position mode, the encoder multiturn value exceeds the actual encoder multiturn bits when unlimited position control is not enabled.	Perform the encoder multiturn zeroing operation, or enable the unlimited position control mode, or work in a non-position mode (torque mode or speed mode).
35418	CPU1 Operation Fault	1. Operational malfunction of the drive firmware; 2. Internal malfunction of the drive.	1. Upgrade the drive firmware; 2. Replace the drive.
35419	CPU2 Operation Fault	1. Operational malfunction of the drive firmware; 2. Internal malfunction of the drive.	1. Upgrade the drive firmware; 2. Replace the drive.
35420	CPU1 Memory Fault	1. Operational malfunction of the drive firmware; 2. Internal malfunction of the drive.	1. Upgrade the drive firmware; 2. Replace the drive.
35421	CPU2 Memory Fault	1. Operational malfunction of the drive firmware; 2. Internal malfunction of the drive.	1. Upgrade the drive firmware; 2. Replace the drive.
35422	CPU Memory Conflict	1. Operational malfunction of the drive firmware; 2. Internal malfunction of the drive.	1. Upgrade the drive firmware; 2. Replace the drive.
35423	Magnetic Pole Positioning Error	1. The setting value of the rotor position compensation angle of the motor is inconsistent with the detected value; 2. The detection	1. Re-detect the motor rotor position compensation angle and set it correctly; 2. Appropriately increase the sensitivity setting value of rotor

		sensitivity for rotor positioning fault is too low; 3. The setting of static balance torque compensation value is inconsistent with the actual load; 4. The wrong wiring of the motor leads to the change of rotor phase angle; 5. The motor malfunction leads to the change of rotor phase angle; 6. The gravity load causes rotation of the motor at the instant when the servo is enabled, and the speed exceeds the threshold of rotor positioning fault detection sensitivity.	positioning fault detection; 3. Set the static balance torque compensation value correctly according to the actual load; 4. Correct wiring, and re-detect the motor rotor position compensation angle; 5. Replace the motor; 6. Set the static balance torque compensation value correctly according to the actual load.
35424	Abnormal Encoder Data	1. Abnormal encoder data; 2. Encoder cable sequence error or poor contact; 3. Abnormal encoder data due to noise interference.	1. Replace the motor or encoder; 2. Correct the wiring sequence or reinforce the wiring; 3. Improve the electromagnetic environment of equipment by standardizing wiring and routing, increasing the cross-sectional area of grounding wire and adding magnetic rings.
35425	Abnormal encoder communication	1. Abnormal encoder data; 2. Encoder cable sequence error or poor contact; 3. Abnormal encoder data due to noise interference.	1. Replace the motor or encoder; 2. Correct the wiring sequence or reinforce the wiring; 3. Improve the electromagnetic environment of equipment by standardizing wiring and routing, increasing the cross-sectional area of grounding wire and adding magnetic rings.
35426	Encoder Communication Timeout	1. Abnormal encoder data; 2. Encoder cable sequence error or poor contact; 3. Abnormal encoder data due to noise interference.	1. Replace the motor or encoder; 2. Correct the wiring sequence or reinforce the wiring; 3. Improve the electromagnetic environment of equipment by standardizing wiring and routing, increasing the cross-sectional area of grounding wire and adding magnetic rings.
35427	Encoder Internal Malfunction 1	Abnormal encoder internal state	Soft reset of the encoder after zeroing or restart the drive
35428	Malfunction of Other Drive Axes	1. Malfunction of other axes; 2. Abnormal internal circuit of the drive	1. Check other axes and reset those reported fault to ensure that other axes are free of fault, and such fault can be prohibited from detection by the parameter 0x2094; 2. Replace the

			drive
35429	Control Encoder Over-Speed	1. The encoder position feedback value variates excessively in a position sampling period, and exceeds 1.3 times of the highest speed of the motor; 2. Encoder malfunction; 3. Abnormal encoder data due to noise interference.	1. Optimize motor parameters and control parameters. The maximum motor set speed is usually not less than the actual maximum motor speed; 2. Check the encoder settings and wiring; 3. Improve the electromagnetic environment of equipment by standardizing wiring and routing, increasing the cross-sectional area of grounding wire and adding magnetic ring.
35430	Drive Continuous Overload	1. Excessive motor load or acceleration/deceleration time set is too short; 2. Actual mechanical load is excessive or it's jammed; 3. The motor's band-type brake is not released; 4. An exception in the motor or motor's band-type brake; 5. Internal malfunction of the drive.	1. Reduce the actual mechanical load or increase the motor acceleration and deceleration time; 2. Check the transmission mode of mechanical load to ensure that there is no jamming; 3. Check the band-type brake wiring and ensure that it is reliable; 4. Replace the motor; 5. Replace the drive.
35431	Drive Output Phase Loss	1. The line break and poor wiring occur in the U, V and W outputs of the drive; 2. The motor impedance is excessive; 3. The internal current sampling circuit of the drive is abnormal.	1. Check the wiring of motor U, V and W and ensure that it is reliable; 2. Replace the motor or turn off the drive output phase loss detection function; 3. Replace the drive.
35432	Motor Stall	1. Unreasonable setting of drive regulator parameters leads to large speed tracking overshoot; 2. Abnormal encoder data due to electromagnetic and noise interference; 3. Abnormal encoder data due to damaged encoder; 4. Abnormal internal circuit of the drive.	1. Optimize regulator parameters; 2. Improve the electromagnetic environment of equipment by standardizing wiring and routing, increasing the cross-sectional area of grounding wire and adding magnetic ring; 3. Replace the motor or encoder; 4. Replace the drive.
35433	Excessive Current Tracking Error	1. Unreasonable setting of drive regulator parameters leads to large speed tracking overshoot; 2. Abnormal encoder data due to electromagnetic and noise interference; 3. Abnormal encoder data due to damaged motor; 4. Abnormal internal circuit of the	1. Optimize regulator parameters; 2. Improve the electromagnetic environment of equipment by standardizing wiring and routing, increasing the cross-sectional area of grounding wire and adding magnetic ring; 3. Replace the motor; 4. Replace the drive.

		drive.	
35434	Abnormal Target Position Value	1. In CSP mode, the difference between the target position value and the actual position value exceeds the threshold set for the position tracking error when the servo is enabled; 2. In CSP mode, the target trajectory acceleration exceeds the threshold set for the maximum acceleration when the motor is running, and the difference between the target position value and the actual position value exceeds the threshold set for the position tracking error.	1. Check and confirm that the target position value and the actual position value are normal, so that the difference does not exceed the set threshold for the position tracking error; 2. Check and confirm that the target position value is normal, or appropriately increase the set threshold for maximum acceleration or the position tracking error.
35435	Encoder Power-On Data Overflow	The feedback value of the position during drive power-up is beyond the range allowed by the encoder.	Soft reset the encoder after zeroing or restart the drive.
35436	Target Position Value Overflow	Target position value exceeds the maximum permissible range when unlimited position control is disabled in the position mode.	Perform the encoder multiturn zeroing operation, or enable the unlimited position control mode, or work in a non-position mode (torque mode or speed mode).
35437	Abnormal Motor's Band-Type Brake	1. There is an exception in the motor band-type brake, and braking fails; 2. The servo is suddenly turned off when the motor is running at high velocity and the braking time is too long; 3. The set braking time for the motor band-type brake is shorter than the actual braking time; 4. The detection sensitivity for rotor positioning fault is too low.	1. Replace the motor; 2. Optimize the process logic control to avoid sudden servo OFF when running at high speed; 3. The braking time set for the motor band-type brake should be not less than the actual braking time; 4. Properly increase the detection sensitivity for rotor positioning fault.
35438	Control Power Supply Under-Voltage	1. The 24V control power supply is abnormal; 2. The wiring of the 24V control power supply is incorrect or poorly connected; 3. The load of the 24V control power supply is excessive; 4. The internal circuit of the drive is abnormal.	1. Replace the 24V control power supply; 2. Check the wiring of the 24V control power supply and ensure that it is reliable; 3. Check the load of the 24V control power supply and ensure that the capacity of the 24V control power supply can meet the load consumption under all working conditions; 4. Replace the drive.
35439	STO1 Triggered	STO1 is triggered or is poorly wired.	Check the STO wiring to ensure that it is reliable and not triggered.
35440	STO2 Triggered	STO2 is triggered or is poorly wired.	Check the STO wiring to ensure that

			it is reliable and not triggered.
35441	Positive Hardware Limit Switch Triggered	One-way motion to the mechanical limit causes hardware limit triggering.	The fault can be cleared directly through motion in the opposite direction until the mechanical limit is restored. Be careful to plan the position so that the hardware limit is not exceeded.
35442	Negative Hardware Limit Switch Triggered	One-way motion to the mechanical limit causes hardware limit triggering.	The fault can be cleared directly through motion in the opposite direction until the mechanical limit is restored. Be careful to plan the position so that the hardware limit is not exceeded.
35443	The Motor Exceeds the Velocity Limit	1. The actual motor speed feedback value exceeds 1.1 times the maximum motor speed; 2. The encoder is abnormal.	1. Optimize motor parameters and control parameters. The maximum motor speed set is usually not less than the actual maximum motor speed; 2. Check the encoder settings and wiring.
35444	Emergency Stop Input Switch Triggered	Emergency stop input switch is triggered or is poorly wired.	Check the emergency stop input switch wiring to ensure that it is reliable and not triggered.
35445	Torque Monitoring Windup Fault	1. The motor load is excessive and it exceeds the torque monitoring alarm threshold; 2. The torque saturation monitoring threshold set is too low.	1. Reduce the actual mechanical load of the motor or increase the motor acceleration and deceleration time; 2. Increase the torque saturation monitoring threshold set. When the threshold is set to 0, the fault will not be detected.
35446	Excessive Velocity Tracking Error	1. Excessive motor load; 2. Inappropriate control parameters; 3. Abnormal release action of the band-type brake; 4. Too small threshold or time duration for judging excessive velocity tracking error.	1. Reduce the actual mechanical load of the motor to ensure that the machinery is not jammed; 2. Optimize the control parameters and enhance the corresponding performance of the servo; 3. Check the line of the band-type brake of the motor to ensure normal action of the band-type brake; 4. Appropriately increase the threshold or time duration for judging excessive velocity tracking error.
35447	Short Circuit of the Drive 2	1. The U/V/W output cable of the drive is short-circuited, or short-circuited to ground; 2. The	1. If a short circuit occurs between the cable's UVW phases, or between the cable's U/V/W and grounding,

		U/V/W output cable of the motor is short-circuited, or short-circuited to ground; 3. An internal cable of the drive is short-circuited, or short-circuited to ground; 4. False alarm is caused because the drive is interfered	dispose or replace the cable; 2. If a short circuit occurs between the motor's UVW phases, or between the motor's U/V/W cable and grounding, replace the motor; 3. If faults still occur after disconnecting the drive U/V/W output wiring, replace the drive; 4. Improve the electromagnetic environment of the equipment by standardizing wiring and cabling, increasing the cross-sectional area of the grounding wire and adding magnetic rings.
35448	Origin Searching Failed	1. Unreasonable parameter setting of origin searching; 2. The motor is already in the limit switch trigger state when the origin searching is activated; 3. Switch to non-HM mode during origin searching.	1. Set correct parameters of origin searching; 2. The sure that the motor is not in the limit switch trigger state when the origin searching is activated; 3. Do not switch to non-HM mode during origin searching.
35449	EtherCAT Process Data Error	The PDO set value is beyond the target's allowable range.	The PDO set value is within the target's allowable range.
35450	Illegal EtherCAT Bus Instruction	EtherCAT communication state machine is incorrectly matched with the control word time sequence.	The host computer correctly handles the EtherCAT communication state computer and control word time sequence.
35451	EtherCAT Communication Cycle Error	1. EtherCAT communication cycle is shorter than the servo control cycle; 2. EtherCAT communication period is not set to an integer power of 2 for 250 μs.	1. Adjust the EtherCAT communication cycle or the servo control cycle so that the communication cycle is longer than the servo control cycle; 2. Set the EtherCAT communication cycle to an integer power of 2 for 250 μs.
35452	Operation Error of Position Planning	1. Target position value cache overrun when running PP mode under EtherCAT control; 2. Internal malfunction of the drive.	1. Optimize the EtherCAT master control process to reduce the number of target position value caches, typically no more than 4; 2. Replace the drive.
35453	Illegal EtherCAT Synchronization Mode	1. EtherCAT communication DC mode is incorrectly configured; 2. DC mode is not activated for EtherCAT communication.	1. Configure EtherCAT communication DC mode correctly; 2. Activate the DC mode for EtherCAT communication.
35454	Target Position Value Beyond the Set Range	The target position value exceeds the set range when the unlimited position	Set the target position value to a value between the lower limit and the

		control mode is enabled or disabled.	upper limit of the position range or use the normal unlimited position mode.
35455	Motor U – phase Instantaneous Overload	1. Excessive motor load; 2. Over-short acceleration and deceleration time durations of the motor; 3. Incorrect setting of motor parameters; 4. Abnormal release action of the band-type brake; 5. Wrong motor model with a smaller power (such as the high-power drive loaded with the small-power motor runs at full load and high speed for a long time); 6. Abnormal internal current sampling circuit of the drive; 7. The fast motor overload protection threshold and protection time duration set are too small.	1. Reduce the actual mechanical load of the motor to ensure that the machinery is not jammed; 2. Prolong the acceleration and deceleration time durations during motor operation; 3. Check the motor parameters to ensure that the motor parameters are set correctly (such as the rated current and thermal time constant of the motor); 4. Check the line of the band-type brake of the motor to ensure normal action of the band-type brake; 5. Change to a high – capacity motor; 6. Replace the drive; 7. Increase the fast motor overload protection threshold and protection time duration appropriately.
35456	Motor V – phase Instantaneous Overload	1. Excessive motor load; 2. Over-short acceleration and deceleration time durations of the motor; 3. Incorrect setting of motor parameters; 4. Abnormal release action of the band-type brake; 5. Wrong motor model with a smaller power (such as the high-power drive loaded with the small-power motor runs at full load and high speed for a long time); 6. Abnormal internal current sampling circuit of the drive; 7. The fast motor overload protection threshold and protection time duration set are too small.	1. Reduce the actual mechanical load of the motor to ensure that the machinery is not jammed; 2. Prolong the acceleration and deceleration time durations during motor operation; 3. Check the motor parameters to ensure that the motor parameters are set correctly (such as the rated current and thermal time constant of the motor); 4. Check the line of the band-type brake of the motor to ensure normal action of the band-type brake; 5. Change to a high – capacity motor; 6. Replace the drive; 7. Increase the fast motor overload protection threshold and protection time duration appropriately.
35457	Motor W – phase Instantaneous Overload	1. Excessive motor load; 2. Over-short acceleration and deceleration time durations of the motor; 3. Incorrect setting of motor parameters; 4. Abnormal release action of the band-type brake; 5. Wrong motor model with a smaller	1. Reduce the actual mechanical load of the motor to ensure that the machinery is not jammed; 2. Prolong the acceleration and deceleration time durations during motor operation; 3. Check the motor parameters to ensure that the motor parameters are set

		power (such as the high-power drive loaded with the small-power motor runs at full load and high speed for a long time); 6. Abnormal internal current sampling circuit of the drive; 7. The fast motor overload protection threshold and protection time duration set are too small.	correctly (such as the rated current and thermal time constant of the motor); 4. Check the line of the band-type brake of the motor to ensure normal action of the band-type brake; 5. Change to a high - capacity motor; 6. Replace the drive; 7. Increase the fast motor overload protection threshold and protection time duration appropriately.
35458	Dynamic Braking Overload	The interval between two adjacent dynamic braking stops is too short when the motor is running.	If there is a dynamic braking stop when the motor is running, the interval must be at least 360*actual speed^2 /rated speed^2 (s).
35459	Internal Malfunction of the Drive	Internal malfunction of the drive.	Replace the drive.
35460	Abnormal Limit Switch	The limit switch is triggered or is poorly wired.	Check limit switch wiring to ensure that it is reliable and not triggered.
35461	Abnormal Communication of the EtherCAT Bus	1. The EtherCAT communication is interfered; 2. Disconnection or poor contact of EtherCAT network cables; 3. Insufficient real-time performance of the host computer; 4. Mismatch between the underlying DC synchronization mechanism of the EtherCAT master of the host computer and the drive requirements; 5. Internal malfunction of the drive.	1. Improve the electromagnetic environment of equipment by standardizing wiring and routing, increasing the cross-sectional area of grounding wire and adding magnetic ring; 2. Check the connection of EtherCAT network cables to ensure reliable connection; 3. Change to the host computer with stronger real-time performance, or extend the EtherCAT communication cycle, or increase the timeout detection sensitivity; 4. Modify the underlying DC synchronization mechanism of the EtherCAT master of the host computer to ensure that the SM2 event of the host computer is at least 125 μs ahead of the DC synchronization signal; 5. Replace the drive.
35462	Interface Encoder Resolution Change	Interface encoder resolution is changed.	Re-power on or perform the soft reset of the drive.
35463	Encoder Overheating	1. The actual encoder temperature is too high; 2. The encoder is abnormal.	1. Bring down the encoder operating temperature to within the allowable range; 2. Replace the motor or encoder.
35464	Encoder Battery Under-Voltage	1. Encoder battery voltage too low;	1. Replace the encoder battery; 2.

	Fault	2. The encoder battery is poorly wired.	Check and handle the battery wiring to ensure that the battery wiring is correct and secure.
35465	Control Mode Setting Error	When the servo is ON, the control mode set is not supported by the drive, such as NM, VL or IP, or the control mode is set to PV or PT under EtherCAT control.	When the servo is ON, set the control mode supported by the drive.
35466	Excessive Power-On Position Deviation	When the drive is powered on, the position is different from the last position saved during power-off and it exceeds the set threshold.	Check whether the mechanical position has been changed. Clear the alert after confirming that the mechanical zero point is normal.
35467	Abnormal Encoder Acceleration	1. Abnormal encoder data; 2. Encoder cable sequence error or poor contact; 3. Abnormal encoder data due to noise interference.	1. Replace the motor or encoder; 2. Correct the wiring sequence or reinforce the wiring; 3. Improve the electromagnetic environment of equipment by standardizing wiring and routing, increasing the cross-sectional area of grounding wire and adding magnetic rings.
35468	Motor Rotor Locked	1. The mechanical load leads to jamming or locking; 2. The motor's band-type brake is not released.	1. Check and handle the mechanical load transmission to ensure that there is no jamming or locking; 2. Check and handle the band-type brake circuit to ensure the band-type brake can be released properly.
35469	EEPROM Data Write Error	Internal malfunction of the drive.	Replace the drive.
35470	Axis EEPROM Data Read Error	Internal malfunction of the drive.	Replace the drive.
35471	Band-type Brake Control Circuit Error	1. The motor band-type brake is short circuited or poorly wired; 2. There is a short circuit or poor wiring inside the band-type brake; 3. Internal malfunction of the drive.	1. Check the wiring of the drive's band-type brake output and ensure it is correct and reliable; 2. Replace the motor; 3. Replace the drive.
35472	CPU1 Overload	1. The drive operation is subject to noise interference; 2. Excessive data collected by the commissioning software; 3. The internal circuit of the drive is abnormal.	1. Improve the electromagnetic environment of equipment by standardizing wiring and routing, increasing the cross-sectional area of grounding wire and adding magnetic ring; 2. Close some of the data collection channels of the commissioning software; 3. Replace the drive.
35473	CPU2 Overload	1. The drive operation is subject to noise interference; 2. Excessive data	1. Improve the electromagnetic environment of equipment by

		collected by the commissioning software; 3. The internal circuit of the drive is abnormal.	standardizing wiring and routing, increasing the cross-sectional area of grounding wire and adding magnetic ring; 2. Close some of the data collection channels of the commissioning software; 3. Replace the drive.
35474	CPU1 Handshake Failed	1. Operational malfunction of the drive firmware; 2. Internal malfunction of the drive.	1. Upgrade the drive firmware; 2. Replace the drive.
35475	DriveMaster Communication Timeout	1. The drive's commissioning cable is disconnected or poorly wired; 2. The drive's commissioning serial communication is interfered.	1. Check the wiring of the drive commissioning cable and ensure a reliable connection 2. Improve the electromagnetic environment of the equipment by using isolated serial port commissioning cable, standardizing wiring and cabling, increasing the cross-sectional area of the grounding wire, and adding magnetic rings.
35476	ESC Configuration EEPROM Error	Internal malfunction of the drive.	Replace the drive.
35477	ESC Internal Access Error	Internal malfunction of the drive.	Replace the drive.
35478	Servo Enabling not Ready	1. When the servo is ON, the drive is in actual motor and virtual encoder mode; 2. When the servo is ON, the encoder communication is disconnected; 3. When the servo is ON, the motor speed is higher than 30rpm; 4. When the servo is ON, the STO status is not lifted; 5. When the servo is ON, the DC bus voltage is too low and the charging relay is not closed; 6. When the servo is ON, the dynamic braking status is not lifted; 7. Internal error of drive.	1. Check the drive motor mode to ensure that the drive is in actual motor and actual encoder mode when the servo is ON; 2. Check the encoder communication status to ensure that the encoder communication is normal when the servo is ON; 3. Check the motor operation status to ensure that the motor is still when the servo is ON; 4. Check the STO status to ensure that the STO status ends when the servo is ON; 5. Check the DC bus voltage status to ensure that the DC bus voltage meets the enable threshold when the servo is ON and the charging relay is closed; 6. Check the dynamic braking status to ensure that the dynamic braking status ends when the servo is ON; 7. Replace the drive.
35479	CPU2 Handshake Failed	1. Operational malfunction of the	1. Upgrade the drive firmware; 2.

		drive firmware; 2. Internal malfunction of the drive.	Replace the drive.
35480	CPU1 Main Task Timeout	1. The drive operation is subject to noise interference; 2. Excessive data collected by the commissioning software; 3. The internal circuit of the drive is abnormal.	1. Improve the electromagnetic environment of equipment by standardizing wiring and routing, increasing the cross-sectional area of grounding wire, and adding magnetic ring; 2. Close some of the data collection channels of the commissioning software.
35481	DC Bus Charging Relay Error	Charging relay inside drive malfunctions.	Replace the drive.
35482	CPU Internal Error	1. Operational malfunction of the drive firmware; 2. Internal malfunction of the drive.	1. Upgrade the drive firmware; 2. Replace the drive.
35483	Actual Position Value Overflow	Actual position value exceeds the maximum permissible range when unlimited position control is disabled in the position mode.	Perform the encoder multiturn zeroing operation, or enable the unlimited position control mode, or work in a non-position mode (torque mode or speed mode).
35484	Encoder Internal Error 2	Encoder internal status error.	Soft reset the encoder after zeroing or restart the drive.
35485	Encoder Internal Error 3	Encoder internal status error.	Soft reset the encoder after zeroing or restart the drive.
35486	Excessive Position 2 Following Error	1. Excessive motor load; 2. Inappropriate control parameters; 3. Abnormal release action of the band-type brake; 4. Too small threshold or time duration for judging excessive position 2 following error.	1. Reduce the actual mechanical load of the motor to ensure that the machinery is not jammed; 2. Optimize the control parameters and enhance the corresponding performance of the servo; 3. Check the line of the band-type brake of the motor to ensure normal action of the band-type brake; 4. Appropriately increase the threshold or time duration for judging excessive position 2 following error.
35487	STO Wiring Error	STO1/STO2 triggered or poorly wired.	Check the STO wiring to ensure that it is reliable and not triggered.
35488	Excessive Velocity 2 Following Error	1. Excessive motor load; 2. Inappropriate control parameters; 3. Abnormal release action of the band-type brake; 4. Too small threshold or time duration for judging excessive velocity 2	1. Reduce the actual mechanical load of the motor to ensure that the machinery is not jammed; 2. Optimize the control parameters and enhance the corresponding performance of the servo; 3. Check

		following error.	the line of the band-type brake of the motor to ensure normal action of the band-type brake; 4. Appropriately increase the threshold or time duration for judging excessive velocity 2 following error.
35489	Abnormal Main Power Input	1. The power input power supply of the driver is poorly wired; 2. The driver power circuit is set as three-phase input, but the actual power supply input is single-phase; 3. Electronic transformer is used in the front end, and the harmonic of electronic transformer is abnormal.	1. Check the power input power wiring of the driver and ensure that the wiring is reliable; 2. Correctly set the driver power circuit, and the set value is consistent with the actual power supply input; 3. Wire according to the transformer manual, and install a filter at the front end of the servo driver if necessary.
35490	Motor Band Brake Disconnection	1. The motor holding brake is not connected or has poor contact; 2. Motor holding brake is abnormal; 3. Drive internal exception.	1. Check and handle the motor band brake wiring to ensure that the wiring is correct and firm; 2. Replace the motor; 3. Replace the drive.
35491	Coprocessor Communication Exception	1. Encoder cable sequence error or poor contact; 2. Encoder data is abnormal due to noise interference.	1. Correct the wiring sequence or reinforce the wiring; 2. Improve the electromagnetic environment of equipment by standardizing wiring and wiring, increasing the sectional area of grounding wire, and adding magnetic ring.
35492	Abnormal Change of Encoder AB Signal	1. Encoder cable sequence error or poor contact; 2. Encoder data is abnormal due to noise interference.	1. Correct the wiring sequence or reinforce the wiring; 2. Improve the electromagnetic environment of equipment by standardizing wiring and wiring, increasing the sectional area of grounding wire, and adding magnetic ring.
35493	Rectifier Module Overheating	1. Abnormal temperature sampling circuit inside the driver; 2. Driver operating environment temperature is outside the allowable operating range.	1. Replace the drive; 2. Decrease the ambient temperature, for example, improve the radiation conditions of cabinets.
35494	Radiator overheating	1. Abnormal temperature sampling circuit inside the driver; 2. Driver operating environment temperature is outside the allowable operating range.	1. Replace the drive; 2. Decrease the ambient temperature, for example, improve the radiation conditions of cabinets.
35495	Motor Overheating	1. The motor load is too large; 2. The	1. Reduce the actual mechanical load

		operating environment temperature of the motor is too high to be allowed. 3. Error in setting thermocouple resistance value for motor overheat protection; 4. abnormal temperature sensor of the motor; 5. Drive internal exception.	of the motor to ensure that the machine is free from jamming; 2. Enhance the heat dissipation of the motor to ensure that the operating environment temperature is within the allowable range; 3. Correctly set the setting value of thermocouple resistance for motor overheat protection; 4. Replace the motor; 5. Replace the drive.
35496	Incremental Encoder Z Signal Exception	1. The encoder's own data is abnormal; 2. Wrong encoder cable sequence or poor contact; 3. Abnormal encoder data due to noise interference.	1. Replace the motor or encoder; 2. Correct the wiring sequence or reinforce the wiring; 3. Improve the electromagnetic environment of the equipment by standardizing wiring and wiring, increasing cross-section area of grounding wire, and adding magnetic rings..
35497	Abnormal Energy Consumption Brake Circuit	Energy consumption brake selection for setting the server parameters is inconsistent with the actual connection of the energy consumption brake resistance.	Correctly set the selected servo parameters for energy consumption braking to match the actual wiring of the energy consumption braking resistance.
35498	CPU Overheat	1. Abnormal temperature sampling circuit inside the driver; 2. Driver operating environment temperature is outside the allowable operating range.	1. Replace the drive; 2. Decrease the ambient temperature, for example, improve the radiation conditions of cabinets.
35499	Power Failure of Main Power Supply	1. When the driver servo is ON, the power supply fails; 2. Abnormal power failure detection of main power supply due to noise interference; 3. The servo parameter main power failure detection time is set too small.	1. Check the power supply and wiring of the driver to ensure that the power supply is normal and the connection is reliable; 2. Improve the electromagnetic environment of equipment by standardizing wiring and wiring, increasing the sectional area of grounding wire, and adding magnetic ring; 3. Properly increase the setting value of servo parameter main power failure detection time.
35500	Abnormal Diagnosis of STO1 Circuit	1. STO1 triggering or poor wiring; 2. Driver internal exception.	1. Check the STO wiring to ensure that it is reliable and not in the triggered state; 2. Replace the drive.
35501	Abnormal Diagnosis of STO2 Circuit	1. STO2 triggering or poor wiring; 2. Driver internal exception.	1. Check the STO wiring to ensure that it is reliable and not in the

			triggered state; 2. Replace the drive.
35502	Hall Signal is Abnormal	1. Hall sensor signal is abnormal; 2. Hall sensor wiring sequence error or poor contact; 3. Hall signal is abnormal due to noise interference.	1. Replace the motor or Hall sensor, or prohibit Hall signal detection; 2. Correct the wiring sequence or reinforce the wiring; 3. Improve the electromagnetic environment of equipment by standardizing wiring and wiring, increasing the sectional area of grounding wire, and adding magnetic ring..
35503	Abnormity of Encoder AB Signal Under Phase	1. Hall sensor or AB encoder signal is abnormal; 2. Hall sensor or AB encoder wiring sequence error or poor contact; 3. Hall or AB encoder signal is abnormal due to noise interference.	1. Replace the motor or hall sensor and encoder; 2. Correct the wiring sequence or reinforce the wiring; 3. Improve the electromagnetic environment of equipment by standardizing wiring and wiring, increasing the sectional area of grounding wire, and adding magnetic ring.
35504	Drive Internal Exception 2	Drive internal exception.	Replace driver.
35505	The robot cfg is incorrect and the robot is not allowed to use it.	The robot cfg is incorrect.	Use the correct robot cfg.
35506	The soft limit exceeds the hard limit.	The soft limit exceeds the hard limit.	The soft limit exceeds the hard limit.
35507	The configuration file has STOP_STO_TIME field and stop1 time greater than 760ms, but the firmware version of the security board is too lower	The configuration file has STOP_STO_TIME field and stop1 time greater than 760ms, but the firmware version of the security board is too lower	Upgrade the firmware of the security board
35508	Failed to set the value of STOP_TIME to Saftey board	Hardware failure	Check hardware
35509	Failed to set the value of STOP_TIME！	The value of STOP_TIME is too large	The value of STOP0_TIME should be < manu STO_TIME-30,The value of STOP1_TIME should be < auto STO_TIME-60.
35600	STO Switch Activation Failure	Robot enabling handle not effective	Press manual enable handle; Reconnect STO cables; Replace STO lines; Replace servo drive
35601	Brake Voltage Anomaly	Hardware brake voltage output exceeds standard	Verify firmware & parameter versions; Replace drive board
35602	Drive Unit Overheat	Joint temperature exceeds operational limits	Perform cooldown process; Update firmware; Replace servo drive
35603	Primary Encoder Error	Motor-side encoder malfunction	Firmware validation; Recalibration; Replace joint assembly

35604	Secondary Encoder Error	Joint-side encoder anomaly	Firmware verification; Recalibration; Replace joint
35605	Encoder Disturbance	Line noise/drive board failure or EMI interference	Inspect connections; Replace drive board; Verify firmware
35606	Current Overload - Phase A	Phase A current exceeding threshold	Check: Cable connections/Motor alignment/Brake status/Load parameters
35607	Current Overload - Phase B	Phase B current exceeding threshold	Check: Cable terminations/Position calibration/Brake function/Load profile
35608	Current Overload - Phase C	Phase C current surpassing limit	Verify: Wiring integrity/Motor offset/Brake operation/Load conditions
35609	IGBT Protection Triggered	Current exceeding preset threshold	Inspect: Load conditions/Deceleration parameters/Mechanical resistance
35610	Motor Overload Protection	Excessive torque or motor stall	Check: Wiring/Acceleration settings/Winding resistance/Brake condition
35611	Current Sensing Anomaly	Current detection anomaly in disabled state	Update firmware/parameters; Replace drive unit
35612	Low DC Voltage Condition	Insufficient bus voltage supply	Verify power input; Replace drive/power boards
35613	High DC Voltage Condition	Voltage surge/Regenerative deceleration anomaly	Inspect power supply; Adjust deceleration profile
35614	Position Tracking Fault	Tracking error exceeding threshold	Verify connections/Hardware integrity/Command rationality
35615	Communication Link Failure	EtherCAT communication failure	Check cabling; Validate firmware compatibility
35616	Angle Alignment Fault	Wiring/parameter configuration error	Inspect: Cable routing/Firmware version/Calibration parameters
35617	Encoder Count Overflow	Firmware mismatch/EEPROM corruption	Upgrade firmware; Replace drive controller
35618	Encoder Count Overflow	Firmware mismatch/EEPROM corruption	Upgrade firmware; Replace drive controller
35619	RSC firmware version is incompatible with the controller version	RSC firmware version is incompatible with the controller version	Check the RSC firmware version and upgrade it to the latest compatible version
35620	Motor short-circuit (overcurrent) protection	1.Excessive current triggers the current protection mechanism	1.Check for any severe impacts; 2.Inspect for potential phase-to-phase short circuits in the motor
35621	Power Board Bus Under-voltage	1. Power cable disconnected or insufficient supply voltage; 2. Power board hardware failure;	1. Check power connection; 2. Inspect servo driver hardware;

35622	Power Board Bus Over-voltage	1. Incorrect power cable polarity causing overvoltage; 2. Power board hardware failure;	1. Verify power input configuration; 2. Inspect power board components;
35623	Main Relay Feedback Abnormal	1. Power board hardware malfunction;	1. Check power board circuitry;
35624	Excessive Leakage Current	1. Short circuit in body wiring harness or power cables; 2. Power board hardware failure;	1. Inspect cable insulation integrity; 2. Test power board functionality;
35625	Brake System Overload	1. Brake circuit open; 2. Brake mechanical failure; 3. Power board or servo driver module defect;	1. Check wiring continuity; 2. Test brake mechanism; 3. Diagnose driver module;
35626	User Power Overcurrent	1. External device wiring error or short circuit; 2. Exceeding rated power capacity;	1. Verify peripheral connections; 2. Confirm device specifications;
35627	EEPROM Write Failure	1. EEPROM component failure;	1. Replace power board;
35628	EEPROM Read Failure	1. EEPROM component failure;	1. Replace power board;
35629	Power Relay Malfunction	1. Power board hardware malfunction;	1. Inspect relay contacts;
35630	CR35 Heat Exchanger Error	1. Heat exchanger wiring short; 2. Power board failure;	1. Check thermal management system wiring; 2. Test power board;
35631	48V Power Undervoltage	1. Joint resistance abnormally high; 2. Power supply misconfiguration;	1. Check joint mechanical operation; 2. Verify power input;
35632	48V Power Overcurrent	1. Excessive mechanical load or acceleration;	1. Optimize motion parameters;
35633	CR35 Bleeder Circuit Error	1. Power board component failure;	1. Replace power board;
35634	CR35 Bleeder Overcurrent	1. Bleeder resistor failure;	1. Replace bleeder module;
35635	CR35 Bleeder MOS Short	1. Power board hardware failure;	1. Replace power board;
35636	Bleeder I^2T Protection Triggered	1. Power board thermal overload;	1. Check cooling system;
35637	Board Communication Error	1. Power board hardware defect;	1. Replace communication module;
35638	FPGA Watchdog Reset	1. Servo driver control board failure;	1. Replace control board;
35639	FPGA Leakage Alert	1. Ground fault in cabinet wiring;	1. Perform insulation resistance test;
35640	Encoder Interpolation Fault	1. Verify encoder hardware integrity;	
35641	Encoder Multi-turn Count Error	1. Encoder battery disconnection; 2. Low battery voltage;	1. Check battery connectors; 2. Measure battery voltage;
35642	Abnormal Encoder Data	1. Verify encoder cable routing; 2. Check encoder magnetic gap;	
35643	Encoder Comms Lost	1. Encoder cable disconnection;	1. Re-seat encoder connectors;
35644	Encoder Overspeed Error	1. Loose encoder connections; 2. Magnetic gap deviation;	1. Secure cable terminations; 2. Adjust encoder alignment;
35645	Sensor CRC Failure	1. Sensor data frame CRC mismatch;	1. Inspect sensor wiring;
35646	Null Sensor Data	1. All-zero data packets received;	1. Check sensor power supply;
35647	Velocity Tracking Error	1. Significant deviation between calculated and actual joint speed;	1. Verify sensor installation; 2. Calibrate speed parameters;

35648	Zero-point Calibration Fault	1. Zero calibration data mismatch;	1. Recalibrate sensor;

17.1.34XXXX

Code	Description	Possible Reasons	Solution
41410	ForceObs limit exceeded	The observed force（Fx、Fy、Fz） in the Cartesian space exceeds the limit	Check the configuration and torque sensor status, modify the program to decrease the total external force acting on the robot
41411	TorqueObs limit exceeded	The observed torque （Mx、My、Mz） in the Cartesian space exceeds the limit	Check the configuration and torque sensor status, modify the program to decrease the total external force acting on the robot
41412	Position limit exceeded	The position in the JOINT space exceeds the limit	Change current position a few increments to allowed area
41413	Position limit exceeded	The position in the Cartesian space exceeds the limit	Change current position a few increments to allowed area
41414	Velocity limit exceeded	You cannot proceed without removing the causes of this error	Reduce speed, use fine point, increase AccSet, avoid singularity, inc. dynamic resolution
41415	Velocity limit exceeded	You cannot proceed without removing the causes of this error	Reduce speed, use fine point, increase AccSet, avoid singularity, inc. dynamic resolution
41416	Singularity Problem	The destination or current position is too close to singularity	Change destination position a few increments. During jogging, use axis by axis. During program execution, use MoveAbsJ
41417	Execution Error	No operation will be possible until after correcting the fault.	Verify that the joint position is within allowed region
41418	FC supervision error	The default force supervision has triggered because the programmed or measured external forces are larger than the safety limit for the robot type	Check the configuration and modify the program to decrease the total external force acting on the robot
41419	FcInit Error	The TCP length exceeds limitation, or the current reference coordinate system is NOT valid	Review the limits for the tool and detailed definitions in the Product manual
41420	Force control activation failed	See the "content"	1. Verify current tool (Tool) settings match actual configuration and ensure proper tool mass/center of gravity settings; 2. Check monitoring window to confirm robot coordinate system and pose match actual status; 3. Confirm current robot model and RD parameters match actual

			configuration; 4. Try returning to mechanical zero position and perform sensor reset before dragging; 5. Ensure no external force is applied when enabling drag; 6. For more detail, refer to the Drag Fault Troubleshooting Manual.
41421	Failed to pause the force control	The robot force control is NOT executed	Command execute only after starting FC
41422	Failed to restart FC	The robot force control is not paused	Pause or Stop the FC before restarting the force control tasks
41423	Failed to set SensorUseMod	The force control is uninitialized	Check the configuration and initialize the FC
41424	Calibration Failed	The dynamic compensation is executed, and a manual calibration is NOT allowed	To use the manual calibration, please change the sensor usage mode to calibration
41425	Failed to set the sine overlay	The overlay is paused in the non-impedance control mode, or the command is NOT allowed when the program is executing	Ensure that it is in the impedance control mode and that the overlay is stopped
41426	Failed to set the Lissajous overlay	The overlay is paused in the non-impedance control mode, or the command is NOT allowed when the program is executing	Ensure that it is in the impedance control mode and that the overlay is stopped
41427	Failed to start the overlay	The overlay is not set, or it is not in the Cartesian impedance control mode	Check control mode command and set the overlay parameters before restarting.
41428	Failed to pause the overlay	The overlay is not started, or it is not in the Cartesian impedance control mode	Please ensure that the overlay is started and it is in the Cartesian impedance control mode
41429	Failed to restart the overlay	The overlay is not paused, or it is not in the Cartesian impedance control mode	Pause the current FC mission and check control mode command. Switch to cartesian impedance control before restarting.
41430	Failed to set ControlType	The impedance control mode is uninitialized	Stop the current FC mission and initialize control parameter again
41431	Initialization error of JntImpedanceFC	Not in the joint impedance control mode; the initialization command is not executed	Execute the Joint impedance control mode before restarting FC
41432	Initialization error of CartImpedanceFC	Not in the Cartesian impedance control mode; the initialization command is not executed	Execute the Cartesian impedance control mode before restarting FC
41433	Failed to set NullSpaceFC	It is not in the Cartesian impedance control mode, or the impedance is	Check the control mode and try again. Do not restart system until a

		uninitialized	valid force calibration or cartesian impedance mode is made.
41434	Failed to set JntForceDes	It is not in the joint impedance control mode, or the impedance is uninitialized	Check the control mode and try again. Do not restart system until a valid force calibration or joint impedance mode is made.
41435	Failed to set CartForceDes	It is not in the Cartesian impedance control mode, or the impedance is uninitialized	Check the control mode and try again. Do not restart system until a valid force calibration or cartesian impedance mode is made.
41436	Switch Command not allowed	Cannot switch impedance mode when in FC Execution state	Stop the current FC mission and initialize control parameter
41437	Failed to activate the virtual wall function	The robot flange is beyond the virtual wall, which does not meet the virtual wall constraints	Please move the robot flange into the virtual wall range and then activate the virtual wall
41438	Failed to activate the virtual wall function	The virtual wall can be activated only in the drag mode	Reactivate the virtual wall after the drag mode is enabled
41439	Failed to activate the virtual wall function	The current state of the robot does not support the virtual wall function	
41440	Deactivate the collision detection	The collision detection is not required in the impedance mode	None
41441	Reactivate the collision detection	The collision detection is deactivated automatically when impedance mode is enabled. Now, the collision detection is automatically activated when the impedance mode is disabled	None
41442	Failed to activate impedance control	When the impedance mode is enabled, the robot does not meet the joint position limitation requirement; the impedance mode cannot be enabled	Please ensure that the robot is within the joint position limitation when the impedance mode is enabled for the robot
41443	Parameter Error in FCStiffSetting	The impedance stiffness set exceeds the theoretical maximum value and has been replaced by the default maximum value	Please reset the impedance stiffness value within a reasonable range
41444	Parameter Error in FCStiffSetting	The impedance stiffness value set is abnormal and does not possess physical significance	Please reset the impedance stiffness value within a reasonable range
41445	Identification Calculation Finished	Identification Calculation Finished, results satisfy the standards.	Please reboot after the second identification process
41446	Identification Exception	Exception occurs during Identification, some results will be replaced by default standards	Results will be replaced by default standards, please follow up hardware qualification
41447	Failed to drag	There is a large deviation between the feedback and the model torque, and the drag can not be activated	1. Check whether the current Tool setting is consistent with the actual situation and whether the set tool

			mass center is reasonable; 2. Check the monitoring window to see whether the robot coordinate system and pose consistent with the actual situation; 3. Confirm that the current robot model and RD parameters are consistent with the actual parameters; 4. Try to return to the mechanical zero and zero the sensors before dragging; 5. For more detail, refer to the Drag Fault Troubleshooting Manual
41448	Failed to initialize the force control	Fcinit is not allowed to be powered off during force control	Click pptomain and run again
41449	Failed to initialize the force control	Fc Commands from last time are not finished, new command will be refused.	Calling Fc Commands too quickly, please increase the time interval between FcStop and FcInit for next time.
41450	Failed to drag	The voltage of the sensor is abnormal,and the drag can not be activated.	Please check the status of sensors.
41451	Force control protection triggered, controller power off	See the "content"	None
41452	Collision detection turned off		
41453	Collision detection reopened		
41454	ForceControl Stopped When PowerOff		
41455	Dont Check Path Deviation		
41456	Resume in Postion Control Mode		
41457	Please Restart Force Control		
41458	The time setting cannot be negative or greater than 600		Please check the time format
41459	In force control mode, unsupported instruction		
41460	The virtual wall area setting is too small		
41462	Calibrate Sensor failed，please check the load setting	The bias between dynamics model trq and sensor measured is too large	Please check the load setting
41461	TCP length is too large(norm<=0.15m),the ref point of force control will be setted at flange		
41463	Drag enable prohibited in		

	workpiece handling mode		
41464	Failed to drag	The range of the joint position limitation is to small to open drag mode	Check the joint position limitation, make sure the the range of the soft limit is beyond 25°
41465	Calibration failed due to excessive deviation between sensor torque and theoretical torque		Check load configuration
41470	Force control protection parameters are set to default values		
41471	Force control protection parameters set to user-defined values		
41472	SetFcJointVelMax execution failed	Invalid command parameters	Check and correct SetFcJointVelMax parameters
41473	SetFcCartVelMax execution failed	Invalid command parameters	Check and correct SetFcCartVelMax parameters
41474	SetFcJointMomentumMax execution failed	Invalid command parameters	Check and correct SetFcJointMomentumMax parameters
41475	SetFcJointEnergyMax execution failed	Invalid command parameters	Check and correct SetFcJointEnergyMax parameters
41476	Program execution rejected due to force control protection trigger	Force control protection activated	Restart program execution from pptomain
41477	In the command parameters, the lower limit exceeds the upper limit.	The command parameters are invalid.	Please check the command parameter settings and correct the parameters.
41478	Sensor identification failed!	Before starting the identification, the robot is not at the mechanical zero point	Please manually jog to the mechanical zero point and then start the identification
42001	Admittance motion exceeds safe range	Exceeds safe range	1. Please check the safety range parameters. 2. Please check if the admittance control parameters are reasonable.

17.1.45XXXX

Code	Description	Possible Reasons	Solution
50000	Blending canceling	The threshold for canceling the blending is exceeded	1. Increase the angle between the two trajectories; 2. Increase the length of the two trajectories; 3. Increase the zone
50001	The controller status is abnormal	The controller status is abnormal	Run PPtoMain to reset projects or reload programs

50002	Exceeds range of motion	1. The target point exceeds the range of motion of the robot. 2. The target point is the singular position in the Cartesian coordinate system	1. Check the target point position. 2. Move the robot by using its joints. 3. Check CONFDATA configuration
50003	Two adjacent target points are too close	Two adjacent target points are too close	Check whether two adjacent Move commands use the same target point
50004	The start point of arc is too close to the end point. Failed to generate arc. Trajectories will be ignored	The start point of arc is too close to the end point. Failed to generate arc. Trajectories will be ignored	1. Adjust the distance between the point positions of the start point and the end point of the arc
50005	The start point of arc is too close to the auxiliary point. Failed to generate arc	The start point of arc is too close to the auxiliary point. Failed to generate arc	1. Adjust the distance between the point positions of the start point and the auxiliary point of the arc
50006	The end point of arc is too close to the auxiliary point. Failed to generate arc	The end point of arc is too close to the auxiliary point. Failed to generate arc	1. Adjust the distance between the point positions of the end point and the auxiliary point of the arc
50007	The end point of arc is too close to the auxiliary point. Failed to generate arc	If the distance between any two of the start point, the auxiliary point, and the end point is too short, the angle between them will be very small	1. Adjust the distance between the point positions of the start point, the end point, and the auxiliary point of the arc
50008	The start point, the auxiliary point, and the end point are on the same straight line. Failed to generate arc	The start point, the auxiliary point, and the end point are on the same straight line. Failed to generate arc	1. Adjust the distance between the point positions of the start point, the end point, and the auxiliary point of the arc
50009	The radius of the arc is too small. Failed to generate arc	The distance between the start point, the end point, and the auxiliary point of the arc are too short	1. Adjust the distance between the point positions of the start point, the end point, and the auxiliary point of the arc
50010	Conditions for generating an arc are not met. Failed to generate arc	The distance between the start point, the end point, and the auxiliary point of the arc are too short, or the start point, the auxiliary point, and the end point of the arc are on the same line	1. Adjust the distance between or the orientation of the point positions of the start point, the end point, and the auxiliary point of the arc
50011	The start point of the trochoid is too close to the end point. Failed to generate trochoid	The start point of the trochoid is too close to the end point. Failed to generate trochoid	1. Adjust the distance between the point positions of the start point and the end point of the trochoid. 2. Conditions for generating a trochoid are as follows: the specified radius is more than 1 mm; the specified feed is more than 1 mm; the trochoid length is more than the sum of two radii and one and a half feeds. Note that the trochoid length depends on the

			distance between the start point, the auxiliary point, and the end point
50012	The start point of the trochoid is too close to the auxiliary point. Failed to generate trochoid	The start point of the trochoid is too close to the auxiliary point. Failed to generate trochoid	1. Adjust the distance between the point positions of the start point and the auxiliary point of the trochoid. 2. Conditions for generating a trochoid are as follows: the specified radius is more than 1 mm; the specified feed is more than 1 mm; the trochoid length is more than the sum of two radii and one and a half feeds. Note that the trochoid length depends on the distance between the start point, the auxiliary point, and the end point
50013	The end point of the trochoid is too close to the auxiliary point. Failed to generate trochoid	The end point of the trochoid is too close to the auxiliary point. Failed to generate trochoid	1. Adjust the distance between the point positions of the end point and the auxiliary point of the trochoid. 2. Conditions for generating a trochoid are as follows: the specified radius is more than 1 mm; the specified feed is more than 1 mm; the trochoid length is more than the sum of two radii and one and a half feeds. Note that the trochoid length depends on the distance between the start point, the auxiliary point, and the end point
50014	The start point, the auxiliary point, and the end point of the trochoid are on the same straight line. Failed to generate trochoid	The start point, the auxiliary point, and the end point of the trochoid are on the same straight line. Failed to generate trochoid	1. Adjust the distance between the point positions of the start point, the auxiliary point, and the end point of the trochoid. 2. Conditions for generating a trochoid are as follows: the specified radius is more than 1 mm; the specified feed is more than 1 mm; the trochoid length is more than the sum of two radii and one and a half feeds. Note that the trochoid length depends on the distance between the start point, the auxiliary point, and the end point
50015	The specified radius is too small. Failed to generate trochoid	The specified radius is too small. Failed to generate trochoid	1. Increase the specified radius. 2. Conditions for generating a trochoid are as follows: the specified radius is more than 1 mm; the specified feed is more than 1 mm; the trochoid length is more than the sum of two radii and one and a half feeds. Note that the trochoid length depends on the distance between the start point, the auxiliary point, and the end point

50016	The specified feed is too small. Failed to generate trochoid	The specified feed is too small. Failed to generate trochoid	1. Increase the specified feed. 2. Conditions for generating a trochoid are as follows: the specified radius is more than 1 mm; the specified feed is more than 1 mm; the trochoid length is more than the sum of two radii and one and a half feeds. Note that the trochoid length depends on the distance between the start point, the auxiliary point, and the end point
50017	The specified radius of the trochoid is less than the feed. Failed to generate trochoid	The specified radius of the trochoid is less than the feed. Failed to generate trochoid	1. The specified radius should not be less than the specified feed. 2. Conditions for generating a trochoid are as follows: the specified radius is more than 1 mm; the specified feed is more than 1 mm; the trochoid length is more than the sum of two radii and one and a half feeds. Note that the trochoid length depends on the distance between the start point, the auxiliary point, and the end point
50018	The trochoid is too short. Failed to generate trochoid	The radius and feed are specified, if the distance between the start point and the end point is short, a trochoid will not be generated	1. Reduce the specified radius or feed. 2. Increase the distance between the start point and the end point. 3. Conditions for generating a trochoid are as follows: the specified radius is more than 1 mm; the specified feed is more than 1 mm; the trochoid length is more than the sum of two radii and one and a half feeds. Note that the trochoid length depends on the distance between the start point, the auxiliary point, and the end point
50019	Internal error generating path	Internal error generating path	1. Re-adjust the target point position, pose, and arm angle. Note that the arm angle only needs to be considered for 7-axis robots
50020	The set tool coordinate system cannot match the features of the current model. Failed to generate the path	The specified tool coordinate system does not support the model	1. The tool coordinate system of the current model must meet the following two requirements – the method for holding: hand-held, and position: x=0, y=0
50021	Exceeds range of motion	1. The target point with the specified confdata has no solution	1. Use the instruction ConfJ off to cancel confdata; 2. Re-teach the point position
50022	Illegal Instructions	1. Illegal Instructions	1. Re-teach the point position; 2. Restart the robot
50023	Singularity errors	1. The target point has no teaching, or its conf information is	Modify the target point conf

		incorrect	
50024	The incorrect trajectory type can make it impossible for the robot to translate along the specified direction, please re-teach the point position	The current robot can only translate on the xz plane of the base coordinate, and cannot deviate from the xz plane of the base coordinate system	Re-teach the point position
50025	The incorrect trajectory type can make it impossible for the robot to rotate along the specified direction, please re-teach the point position	While the current robot rotating, its rotation axis must be in parallel with the y-axis of the base coordinate system	Re-teach the point position
50026	Waiting for the next motion instruction to be analyzed for too long,automatically cancel the turning area	Insert too many non-motion commands between two motion commands,automatically cancel the turning area	1. Run the RL command "AotoIgnoreZone Off",The system will not automatically cancel the turning zone. 2.Simplify the non-motion instruction between two motion instructions
50027	Track length less than minimum turning radius, automatic path splicing, integrated turning area.	1.The trajectory needs to be connected to the turning area both front and back, but the length of the trajectory is less than twice the radius of the minimum turning area; 2.The track is set to connect to a turning area, but the track length is less than the minimum turning area radius	This function can make the movement smoother. If you want to turn off this function, you can set the minimum turning area radius to 0
50028	The spiral's initial radius is less than zero. Failed to generate spiral	The spiral's initial radius is less than zero. Failed to generate spiral	1. Increase the spiral's radius step. 2. Conditions for generating a spiral are as follows: the initial radius is not less than 0; the radius step is more than 0.0001 mm/deg; the cumulative rotation angle is between 0.1 and 3600 deg;
50029	The spiral's radius step is too low. Failed to generate spiral	The spiral's radius step is too low. Failed to generate spiral	1. Increase the spiral's radius step. 2. Conditions for generating a spiral are as follows: the initial radius is not less than 0; the radius step is more than 0.0001 mm/deg; the cumulative rotation angle is between 0.1 and 3600 deg;
50030	The spiral's cumulative rotation angle is illegal. Failed to generate spiral	The spiral's cumulative rotation angle is illegal. Failed to generate spiral	1. Increase the spiral's cumulative rotation angle. 2. Conditions for generating a spiral are as follows: the initial radius is not less than 0; the radius step is more than 0.0001 mm/deg; the cumulative rotation angle is between 0.1 and 3600 deg;

50031	Trajectory error. Spiral trajectory does not support handheld work objects	Spiral trajectory does not support handheld work objects	Replace with handheld tools
50033	Wrong trajectory,the endpoint deviates from lock axis angle	The endpoint deviates from lock axis angle(0,180,-180)	Adjust the endpoint or close lock axis
50034	Unable to reach the target in the locked axis state or 5-axis robot.Please change the target	Unable to reach the target in the locked axis state or 5-axis robot.Please change the target	Unable to reach the target in the locked axis state or 5-axis robot.Please change the target
50035	Unable to generate trajectory of identification in the locked axis, Please close lock axis	trajectory couldn't be generated correctly in the locked axis state	Please close lock axis
50036	5 axis robot unable to do load identification	5 axis robot unable to do load identification	none
50040	Unable to do identification because the soft limit is not enabled	The soft limit is not enabled	Please enable the soft limit
50041	Unable to do identification because the joint position limitation is incorrectly set	The joint position limitation is incorrectly set	Please set the joint position limitation correctly
50042	The stating point of this trajectory is incorrect，this robot needs to keep flange parrallel to the base	The stating point of this trajectory is incorrect， this robot needs to keep flange parrallel to the base	Re demonstrate teaching points
50043	The target point of this trajectory is incorrect，this robot needs to keep flange parrallel to the base	The target point of this trajectory is incorrect， this robot needs to keep flange parrallel to the base	Re demonstrate teaching points
50044	The back direction is wrong. Please move towards the work area direction.	The back direction is wrong. Please move towards the work area direction.	Please move towards the work area direction.
50101	The joint angle exceeds limit	The axis-angle motion exceeds the motion range	1. Cancel the joint position limitation 2. Manually move each axis of the robot to the normal working range
50102	During the lookahead of the trajectory, encounter singularities	There are trajectories across singularities	Please avoid singularities (refer to the manual for the relevant information): 1. Re-teach the point position, and change the target point; 2. Or change the Cartesian space motion instruction to the joint space motion instruction
50103	Position incompatibility	Inability to move to the target point of the given ConfData	1. Change the target point ConfData; 2. Change to MoveJ or MoveAbsJ

		through the planning of the Cartesian space	
50104	During the lookahead, the joint torque exceeds the limit	1.The load value is set too high, exceeding the robot's load-bearing capacity; 2. The friction coefficient of the robot is too high. 3. The electric overload coefficient or transmission overload coefficent of the robot is too small.	1.Check if the load values match the actual situation. 2. Check parameters such as friction coefficent,motor overload coefficient,transmission overload coefficient; 3. Try to change the type of the instruction, for instance, change the Cartesian space instruction to the joint space instruction;
50105	Lookahead points are not continuous	1. Program logical problems; 2.Dynamic parameters errors, there are parameters not covered in the reasonable range	1. Modify program logic; 2. Check dynamic parameters; 3. Change the point position or the blending
50106	The staring point of the trajectory is greater than the ending point one, and the trajectory is unreasonable	Program logical problems	For program logical problems, modify the point position of the program, or modify its blending
50107	Generate the incorrect trajectory of the blending	Internal Error	Cancel the blending, change the size of the blending or the target point
50108	The new trajectory was not obtained in time, and the motion stopped	System failure	
50109	The empty queue of the trajectory makes it impossible to execute the planning	Set trajectory errors	Avoid generating the blending between the two reversed trajectories or reduce the length of the blending
50110	Unable to get the initial position of the program	Unable to get the initial position of the program	1. Modify program logic; 2. Modify the size of the blending of the first trajectory
50111	The positions of the two trajectories cannot connect	The positions of the two trajectories cannot connect	1 Re-teach the point position; 2. Modify the size of the blending
50112	Incorrect entered poses	Incorrect entered poses	1. Re-teach the pose, 2. If the current model is 3-axis or 4-axis robot, please check whether the entered pose matches the features of the current model
50113	[WristSing]the change of robot posture is over the limitation	[WristSingthe change of robot posture is over the limitation	Please avoid singularities (refer to the manual for the relevant information):1. Re-teach the point position, and change the target point 2. Reset larger posture limitation 3. Try different type singular avoidence
50114	The joint angle exceeds limit	The axis-angle motion exceeds the motion range	1. Cancel the joint position limitation 2. Manually move each axis of the robot to the normal working range
50115	There are singularities	During the lookahead, the	Please avoid singularities (refer to the manual

	during the lookahead of the trajectory. Please avoid the singularities	trajectory goes across the singularity	for the relevant information): 1. Re-teach the point position, and change the target point; 2. Or change the Jog Mode and try to use Joint space Jog; 3. Or change the Cartesian space motion instruction to the joint space motion instruction
50116	During the lookahead of the trajectory, encounter singularities	The joint of trajectories is a singularity	Please avoid singularities (refer to the manual for the relevant information): 1. Re-teach the point position, and change the target point; 2. Add wait0 after the incorrect trajectory
50117	[WristSing]During the lookahead, the trajectory goes across the shoulder/elbos singularity	[WristSing]During the lookahead, the trajectory goes across the shoulder/elbos singularity	Please avoid singularities (refer to the manual for the relevant information):1. Re-teach the point position, and change the target point 2. Try different sing type
50118	[WristSing]During the lookahead of the trajectory, the end pos dose not match required value	[WristSing]During the lookahead of the trajectory, the end pos dose not match required value, unable get a certain path	Please avoid singularities (refer to the manual for the relevant information):1. Re-teach the point position, and change the target point 2. Try different sing type
50119	Unable to open the singular avoidance when using track	Unable to open the singular avoidance when using track	close the track
50120	[WristSing]Search end point joint failed	[WristSing]Search end point joint failed. Probabaly because the end point is singular point or approach position limitation.	Please avoid singularities (refer to the manual for the relevant information):1. Re-teach the point position, and change the target point 2. Try different sing type
50121	[WristSing]Search path point joint failed	[WristSing]Search path point joint failed. Probabaly because the end point is singular point or approach position limitation.	Please avoid singularities (refer to the manual for the relevant information):1. Re-teach the point position, and change the target point 2. Try different sing type
50122	Unable to open the singular avoidance when using fc model	Unable to open the singular avoidance when using fc model	close the fc model
50123	Singular avoidance doesn't support recent path type, please change to cartesian linear path	Singular avoidance doesn't support recent path type, please change to cartesian linear path	Please change path type to cartesian linear
50124	The angle between the upper and lower arms exceeds the safe range	The angle between the upper and lower arms exceeds the safe range	Please change target point
50125	The angle between the upper and lower arms exceeds the safe range	The angle between the upper and lower arms exceeds the safe range	Please change motion type
50126	During the lookahead, joint max speed exceeds limit	1.The joint speed is set too small	1.Increase the maximum joint speed
50127	During the lookahead, joint	1.The joint acceleration is set too	1.Increase the maximum joint acceleration

	max acceleration exceeds limit	small	
50128	During the lookahead, joint max jerk exceeds limit	1.The joint jerk is set too small	1.Increase the maximum joint jerk
50201	The Cartesian path encounters the unreachable point	The Cartesian path encounters singularities	Change the trajectory, or move to the target position through the joint space
50202	The Cartesian path encounters illegal poses	The target point in the Cartesian path cannot match the configuration of the current robot	Re-teach the target point, and change it the teaching point if the target point is manually entered
50203	Path planning errors	Internal Error	
50204	Insufficient sampling points during the process of planning	1. There are too many logic judgments or too much calculation of point positions inserted between the two motion instructions; 2. The state of the IPC is unstable, please check whether there is the non-xCore program operating in the robot	1. Modify the RL project, and restart its operation; 2. Close the Linux background program of the non-xCore control system
50205	Too much difference in adjacent instruction points during motion planning	Controller error, there is a great difference in adjacent instruction points during motion planning, exceeding the limit	1. Re-teach the target point position of the trajectory; 2.Try to change the type of the instruction, for instance, change MoveL to MoveJ; 3. Add wait0 after the incorrect trajectory to refresh the state
50206	Incorrect internal calculation of path planning	The calculation of controller planning is incorrect	1. Re-teach the target point position of the trajectory; 2.Try to change the type of the instruction, for instance, change MoveL to MoveJ; 3. Try to change the size of the blending of the trajectory and the expected speed; 4. Add wait0 after the incorrect trajectory to refresh the state
50207	Not stopped at end point of path	Incorrect planning, and stop somewhere in the middle of the robot path	Change the specified motion parameters (the size of the blending, and the position of the target point), or change the pose of the target point, or change the motion instruction (such as change MoveL to MoveJ)
50208	Trajectory internal error	Trajectory internal error	1. Expand or reduce the blending. 2. Teach point positions again
50209	Failed to stop within the specified stopping distance	The specified stopping distance is too short	Increase the stopping distance
50210	Path turn angle deviation exceeds limit	Actual Cartesian path turn angle differs significantly from programmed value	1. Use 'ConfL off' command to bypass this warning; 2. Reteach target point with turn angle closer to starting point
50211	Constraint violation detected	1. Load setting exceeds robot	1. Verify actual load matches settings; 2.

	during planning while stationary	capacity; 2. Excessive friction coefficient; 3. Motor/servo overload coefficient too small; 4. Planning calculation error	Check friction/motor/transmission overload coefficients; 3. Try changing command type (e.g., switch from Cartesian to joint space); 4. Contact technical support
50301	The thread for motion planning is blocked. Execution timed out	The thread for motion planning is blocked. Execution timed out	Use PPtoMain again to run projects or restart the controller
50302	The thread for motion planning is blocked. Scheduling timed out	System failure	Use PPtoMain again to run projects or restart the controller
50303	The thread for motion planning is blocked. Consecutive timeout over 5000 times	System failure	Use PPtoMain again to run projects or restart the controller
50304	No sufficient commands are sent from the planner to the EtherCAT thread	System failure	
30400	Robot stopped due to collision. Check the robot operating environment and confirm that the staff and devices are safe before restart	Robot collision	1.Check the robot operating environment, confirm that the staff and devices are safe, and power on the robot, before restart
30401	Robot stopped due to collision. Check the robot operating environment and confirm that the staff and devices are safe before restart	Robot collision	1. Check the robot operating environment, confirm that the staff and devices are safe, and power on the robot, before restart; 2. Check whether the current Tool setting is consistent with the actual situation and whether the set tool mass center is reasonable
30402	Robot stopped due to collision. Check the robot operating environment and confirm that the staff and devices are safe before restart	Robot collision	1. Check the robot operating environment, confirm that the staff and devices are safe, and power on the robot, before restart; 2. Check whether the current Tool setting is consistent with the actual situation and whether the set tool mass center is reasonable
30403	Robot stopped due to collision. Check the robot operating environment and confirm that the staff and devices are safe before restart	Robot collision	1. Check the robot operating environment, confirm that the staff and devices are safe, and power on the robot, before restart; 2. Check whether the current Tool setting is consistent with the actual situation and whether the set tool mass center is reasonable
50401	Set max torque limit failed	The max torque limit is out of range, set parameter failed	Set correct max torque limit
50501	Tool and work object	Set "handheld" for both work	Select correct tool and wobj

	coordinate system settings conflict	objects and tools	
50502	Tool and work object coordinate system settings conflict	Set "external" for both work objects and tools	Select correct tool and wobj
50503	Tool and work object coordinate system settings conflict	Set "handheld" for both work objects and tools	Select correct tool and wobj
50504	Tool and work object coordinate system settings conflict	Set "external" for both work objects and tools	Select correct tool and wobj
50506	Category 1 internal mode error of motion modules	Internal error	Restart robot. If the problem persists, contact technical support
50507	Category 2 internal mode error of motion modules	Internal error	Restart robot. If the problem persists, contact technical support
50508	Failed to reset module status	Internal error	Restart robot. If the problem persists, contact technical support
50509	Failed to set motion start point	Internal error	Restart robot. If the problem persists, contact technical support
50510	Failed to generate trajectories	Internal error	Restart robot. If the problem persists, contact technical support
50511	Failed to execute motion command.	Current robot state does not permit to execute any movement.	1.Interrupt current project and restart it from beginning 2.Reboot robot. If the problem persists, contact technical support
50512	Number of axes mismatches	Software error	Please contact the technical support
50513	Invalid speed settings	Software error	Please contact the technical support
50514	Invalid step length settings	Software error	Please contact the technical support
50515	Invalid settings of coordinate systems	Software error	Please contact the technical support
50516	Invalid settings of motion axes	Software error	Please contact the technical support
50517	End-effector rapid adjustment failed. Teach point positions again	1. End-effector rapid adjustment failed due to special or extreme positions	Teach point positions again within the normal working range of robots
50518	The current point position may be the target point position	The current point position may be the target point position	None
50519	Failed to generate trajectories. Unqualified target points	Target point positions may be out of robot working range	Teach point positions again within the normal working range of robots
50520	JOG mode error. Robot cannot translate in the specified direction. Modify the JOG mode	The current robot can only translate on the xz plane of the base coordinate, and cannot deviate from the xz plane of the	It is recommended that to choose the base coordinate system or the joint coordinate system when modifying the JOG mode

		base coordinate system	
50521	JOG mode error. Robot cannot rotate in the specified direction. Modify the JOG mode	The rotating axes of the robot must be parallel to the Y-axis of the base coordinate system, or the robots rotate along the Z-axis of the flange coordinate system	It is recommended that to choose the base coordinate system or the joint coordinate system when modifying the JOG mode
50522	Trajectory error. 4-axis PCB robots do not support handheld work objects and external tools	4-axis PCV robots do not support handheld work objects and external tools	Replace with handheld tools
50523	The robot already reached end point of trajectory	When you click Next, the robot have reached the end point of the trajectory. The robots will not move	Confirm whether you want to stop the robot at the end point of the trajectory. To move the robot to the end point of the next trajectory, click Next
50524	JOG failed to open, missing key motion parameters	The current robot configuration file lacks key motion parameters and does not allow JOG	Confirm that the configuration file is updated
50525	The axis4 angle is not 0 or 180 degrees,please jog j4 to 0 or 180 degrees firstly	The robot's axis4 angle is not 0 or 180 degrees,please jog j4 to 0 or 180 degrees firstly	The robot's axis4 angle is not 0 or 180 degrees,please jog j4 to 0 or 180 degrees firstly
50526	The flange is not parallel to the base,Please jog J4 and Ry first to make the flange parallel to the base	The flange is not parallel to the base，it is not allowed to jog x y z	Please jog J4 and Ry first to make the flange parallel to the base
50527	In the locked state, it is not supported to hold the workpiece	In the locked state, it is not supported to hold the workpiece	it is supported to hold the tool
50528	The flange is not parallel to the base,Please jog Ry to make the flange parallel to the base	The flange is not parallel to the base，it is not allowed to jog x y z	Please jog Ry to make the flange parallel to the base
50529	Invalid load settings	Invalid load settings	Please check the load setting
50530	The button is being clicked too frequently.	The previous motion has not yet fully stopped, so the current motion cannot be initiated	Increase the time interval between consecutive motions to prevent overly rapid clicks.
50531	Rapid end-effector adjustment failed, please select the correct tool or the correct coordinate system	1. The tool is an external tool, and the adjustment is based on a coordinate system that is the robot's base coordinate system	Select the correct tool or the correct coordinate system
50532	Turn off precision compensation state	When the precision compensation state is enabled, singular avoidance and parallel pedestal jog cannot be enabled	Turn off precision compensation state
50601	Dynamics disabled.	Torque feedforward disabled.	Enable torque feedforward

	Resetting the torque feedforward may cause a sudden power-on jitter	Torque feedforward reset	
50602	Friction identification		
50603	VirbrationSuppression Failed		
50604	GravityCompensation Failed		
50605	Open Collision Detection delay compensation parameters identification failed, torque feedforward is closed	torque feedforward is closed	Open torque feedforward
50701	Command data lost during transmission	Command data lost during transmission	Restart the program
50702	The configuration file dose not match the controller version;	The configuration file does not match the controller version;	Please download the latest version of the configuration file on the SW website and upgrade it to the controller;
50801	Trajectory type is not supported in weaving mode	Weaving mode does not support joint-space trajectory, pure rotation trajectory or trochoid trajectory	Do not use cartesian-space trajectory, pure rotation trajectory or trochoid trajectory
50802	When the network is disconnected or unstable, the robot will stop moving.	During JOG, motion to or quick adjustment, stop the robot movement to prevent machine collision due to network disconnection or network instability.	Please check that the network connection is normal, ensure that the network environment is stable, and retry the JOG, motion to, or quick adjustment operation.

17.1.56XXXX

Code	Description	Possible Reasons	Solution
60000	RL instruction parameter error	RL instruction parameter error	Modify the instruction and enter the correct parameter.
60001	SocketReadBit parameter error	RL encounters an error while running.	Parameters of SocketReadBit must be multiples of 8.
60002	SocketReadBit failed to read data.	RL encounters an error while running.	SocketReadBit failed to read data (data length mismatch).
60003	SocketReadDouble illegally reads data.	RL encounters an error while running.	SocketReadDouble illegally reads data.
60004	SocketReadDouble failed to read data.	RL encounters an error while running.	SocketReadDouble failed to read data (data length mismatch).
60005	Failed to load RL project.	1. rsync configuration file is incorrect or missing. 2. The hmi version does not match. 3. The RL project file is accidentally	1. Configure the correct operating environment. 2. Check the corresponding HMI version. 3. Contact technical support.

		modified. 4. Network failure.	
60006	The robot is running. pptomain failed.	Robot is running, pptomain is not allowed.	Press Pause or Emergency Stop to shut down the robot.
60007	The joint position limitation is not enabled. Single-step debugging is not allowed.	The joint position limitation is not enabled. Single-step debugging is not allowed.	Open joint position limitation settings in the setting interface.
60008	Running speed synchronization failed. Cancel the next step.	Running speed synchronization failed. Cancel the next step.	Pause and restart the controller.
60009	RL is running or the actuator encounters an error. The next step is rejected.	RL is running or the actuator encounters an error. The next step is rejected.	If RL is running, wait for RL to stop or click Pause; If the actuator encounters an error, click pp_to_main to re-execute.
60010	Failed to start interpreter, unable to perform the next step.	Failed to start interpreter, unable to perform the next step.	Restart the controller or contact technical support.
60011	An error occurred during RL task execution. The task stopped.	An error occurred during RL task execution. The task stopped.	Switch to the corresponding task to check the logic error near the program pointer, and after eliminating the error, pptomain will run again.
60012	GetSocketConn failed.	The connection does not exist or has been disconnected.	Execute this command after SocketConnect is connected.
60013	GetSocketServer failed.	The connection does not exist or has been disconnected.	Execute this instruction after SocketServer is in listening state.
60014	Unable to start running the RL program.	The program failed to start running because: 1) the robot is not powered on; 2) the robot is running.	Make sure the robot is powered on and is not running.
60015	Reload project failed，variables are too much.	1.Project variable overlimit 14000.	Project variable overlimit
60100	Parameter error of HexToDec	RL encounters an error while running.	The string entered by HexToDec must be a hexadecimal integer.
60101	StrToByte parameter error	RL encounters an error while running.	The string entered by StrToByte(string, \Hex) must be a hexadecimal integer.
60102	StrToByte parameter error	RL encounters an error while running.	The string entered by StrToByte(string) must be a decimal integer.
60103	StrToByte parameter error	RL encounters an error while running.	The string entered by StrToByte(string, \Okt) must be an octal integer.
60104	StrToByte parameter error	RL encounters an error while running.	The string entered by StrToByte(string, \Bin) must be a binary integer.
60105	StrToByte parameter error	RL encounters an error while running.	The string entered by StrToByte(string, \Char) must be ASCII characters.
60106	The result of StrToByte overflows.	RL encounters an error while running.	The string data entered by StrToByte(string HEX) overflows.
60107	The result of StrToByte overflows.	RL encounters an error while running.	The string data entered by StrToByte(string DEC) overflows.
60108	The result of StrToByte overflows.	RL encounters an error while running.	The string data entered by StrToByte(string OKT) overflows.

60109	The result of StrToByte overflows.	RL encounters an error while running.	The string data entered by StrToByte(string BIN) overflows.
60110	The result of StrToByte overflows.	RL encounters an error while running.	The string data entered by StrToByte(string CHAR) overflows.
60111	The range of the StrPart string exceeds the limit.	RL encounters an error while running.	The range of the StrPart string exceeds the limit.
60112	The StrMatch parameter is too large to find.	RL encounters an error while running.	The StrMatch parameter is too large to find.
60113	The value range of BitPos parameter of BitCheck instruction exceeds the limit.	RL encounters an error while running.	The value range of BitPos parameter of BitCheck instruction exceeds the limit.
60114	The value range of BitPos parameter of BitClear instruction exceeds the limit.	RL encounters an error while running.	The value range of BitPos parameter of BitClear instruction exceeds the limit.
60115	The value range of ShiftSteps parameter of BitLsh instruction exceeds the limit.	RL encounters an error while running.	The value range of ShiftSteps parameter of BitLsh instruction exceeds the limit.
60116	The value range of ShiftSteps parameter of BitRsh instruction exceeds the limit.	RL encounters an error while running.	The value range of ShiftSteps parameter of BitRsh instruction exceeds the limit.
60117	The value range of BitPos parameter of BitSet instruction exceeds the limit.	RL encounters an error while running.	The value range of BitPos parameter of BitSet instruction exceeds the limit.
60198	Command failed		
60199	Command executed successfully		
60200	An error occurs during FcInit execution.	An error occurs during FcInit execution.	Modify the FcInit instruction to the correct instruction parameter.
60201	An error occurs during FcStart execution.	An error occurs during FcStart execution.	Modify the FcStart instruction to the correct instruction parameter.
60202	An error occurs during FcPause execution.	An error occurs during FcPause execution.	Modify the FcPause instruction to the correct instruction parameter.
60203	An error occurs during FcRestart execution.	An error occurs during FcRestart execution.	Modify the FcRestart instruction to the correct instruction parameter.
60204	An error occurs during FcStop execution.	An error occurs during FcStop execution.	Modify the FcStop instruction to the correct instruction parameter.
60205	An error occurs during ClearFcError execution.	An error occurs during ClearFcError execution.	Modify the ClearFcError instruction to the correct instruction parameter.
60206	An error occurs during SetControlType execution.	An error occurs during SetControlType execution.	Modify the SetControlType instruction to the correct instruction parameter.
60207	An error occurs during SetJntCtrlStiffVec execution.	An error occurs during SetJntCtrlStiffVec execution.	Modify the SetJntCtrlStiffVec instruction to the correct instruction parameter.
60208	An error occurs during SetCartCtrlStiffVec execution.	An error occurs during SetCartCtrlStiffVec execution.	Modify the SetCartCtrlStiffVec instruction to the correct instruction

			parameter.
60209	An error occurs during SetCartNSStiff execution.	An error occurs during SetCartNSStiff execution.	Modify the SetCartNSStiff instruction to the correct instruction parameter.
60210	An error occurs during SetLoad execution.	An error occurs during SetLoad execution.	Modify the SetLoad instruction to the correct instruction parameter.
60211	An error occurs during StartOverlay execution.	An error occurs during StartOverlay execution.	Modify the StartOverlay instruction to the correct instruction parameter.
60212	An error occurs during StopOverlay execution.	An error occurs during StopOverlay execution.	Modify the StopOverlay instruction to the correct instruction parameter.
60213	An error occurs during PauseOverlay execution.	An error occurs during PauseOverlay execution.	Modify the PauseOverlay instruction to the correct instruction parameter.
60214	An error occurs during SetSineOverlay execution.	An error occurs during SetSineOverlay execution.	Modify the SetSineOverlay instruction to the correct instruction parameter.
60215	An error occurs during SetLissajousOverlay execution.	An error occurs during SetLissajousOverlay execution.	Modify the SetLissajousOverlay instruction to the correct instruction parameter.
60216	An error occurs during SetJntTrqDes execution.	An error occurs during SetJntTrqDes execution.	Modify the SetJntTrqDes instruction to the correct instruction parameter.
60217	An error occurs during SetCartForceDes execution.	An error occurs during SetCartForceDes execution.	Modify the SetCartForceDes instruction to the correct instruction parameter.
60218	An error occurs during RestartOverlay execution.	An error occurs during RestartOverlay execution.	Modify the RestartOverlay instruction to the correct instruction parameter.
60219	An error occurs during SetSensorUseType execution.	An error occurs during SetSensorUseType execution.	Modify SetSensorUseType to the correct parameter.
60220	An error occurs during CalibSensorError execution.	An error occurs during CalibSensorError execution.	Modify CalibSensorError to the correct parameter.
60221	An error occurs during FcCondForce execution.	An error occurs during FcCondForce execution.	Modify the FcCondForce instruction to the correct instruction parameter.
60222	An error occurs during FcCondPosBox execution.	An error occurs during FcCondPosBox execution.	Modify the FcCondPosBox instruction to the correct instruction parameter.
60223	An error occurs during FcCondTorque execution.	An error occurs during FcCondTorque execution.	Modify the FcCondTorque instruction to the correct instruction parameter.
60224	An error occurs during FCCondWaitWhile execution.	An error occurs during FCCondWaitWhile execution.	Modify the FCCondWaitWhile instruction to the correct instruction parameter.
60225	An error occurs during FcCondOrient execution.	An error occurs during FcCondOrient execution.	Modify the FcCondOrient instruction to the correct instruction parameter.
60226	An error occurs during FcCondPosSphere execution.	An error occurs during FcCondPosSphere execution.	Modify the FcCondPosSphere instruction to the correct instruction parameter.
60227	An error occurs during FcCondPosCylinder execution.	An error occurs during FcCondPosCylinder execution.	Modify the FcCondPosCylinder instruction to the correct instruction parameter.
60228	An error occurs during FcCondTcpSpeed execution.	An error occurs during FcCondTcpSpeed execution.	Modify the FcCondTcpSpeed instruction to the correct instruction parameter.

60229	Inverse kinematics of CalcJointT failed.	The conf error of Cartesian coordinates	Modify the entered Cartesian coordinates.
60300	ReplayPath playback rate exceeded limit.	RL encounters an error while running.	ReplayPath playback rate exceeded limit.
60400	RL call hierarchy exceeds limit.	RL encounters an error while running.	RL call hierarchy exceeds limit.
60500	Cannot load multiple motion tasks.	Cannot load multiple motion tasks.	Assign only one task as the motion task.
60501	Predecessor task error	The predecessor task cannot be itself.	Modify predecessor tasks to other tasks.
60502	Motion task is not allowed to set predecessor task.	Motion task is not allowed to set predecessor task.	Motion task is not allowed to set predecessor task.
60503	The motion instruction cannot hold the workpiece and the tool at the same time.	The motion instruction cannot hold the workpiece and the tool at the same time.	Modify the instruction for the tool and wobj parameter.
60504	Invalid Search instruction input signal	The signal type is neither a DI signal nor register	Use the correct DI signal or register.
60505	The Search instruction failed to record the point position.	The tool and the workpiece are simultaneously hand-held or external.	Modify parameters and use tools and workpieces at different positions.
60506	An error occurs in the Search instruction DI signal.	The DI signal does not exist or has been set as the system input.	Modify the parameter and use the DI signal that is not set as the system input.
60507	SearchL \Stop instruction, the speed is greater than v100.	The robot can be stopped quickly only when the speed is less than or equal to v100.	Use a smaller speed parameter.
60508	SearchC \STOP instruction, the speed is greater than v100.	The robot can be stopped quickly only when the speed is less than or equal to v100.	Use a smaller speed parameter.
60509	Wrong program state, cannot perform the next step.	1. Cannot reach the motion instruction point; 2. Move through a singular point;	1. Check the list of point positions; 2. Optimize the motion trajectory; 3. Click PPtoMain.
60510	The RL instruction runs too slowly.	Tcp < 1 μ m/s or ori < 1e-6° /s or jnt < 1%;	The motion instruction adopts a proper speed parameter.
60511	The drag playback path is missing/does not exist.	The drag playback path is missing/does not exist.	Record/use a valid playback path.
60512	Drag playback function error	Drag playback function error	Check whether the drag playback function is effective.
60513	HomeSet instruction range error.	Setting the home point angle exceeds the limit.	Modify the angle of the Home point to be within the hard limit range of the robot.
60514	Saving Home parameter failed.	Robot configuration module error	Try restarting the robot or contact the manufacturer.
60515	HomeSetAt acquisition failed.	The axis specified by HomeSetAt does not exist.	Modify the parameters of HomeSetAt.
60516	The HomeClr instruction failed.	Robot configuration module error	Try restarting the robot or contact the

			manufacturer.
60517	Wrong number of HomeSet parameters	Wrong number of HomeSet parameters	Modify the number of parameters to that of the robot axes.
60518	Wrong number of Hordr parameters	Wrong number of Hordr parameters	Modify the RL file and transfer the correct Hordr parameters.
60519	Hordr parameter value error	Hordr parameter value error	Modify the RL file and transfer the correct Hordr parameters.
60520	HordrAt parameter value error	HordrAt parameter value error	Modify the RL file and transfer the correct Hordr parameters.
60521	GetEndtoolTorque tool is not compatible with the workpiece.	The tool and the workpiece are simultaneously hand-held or external.	Modify the GetEndtoolTorque instruction and transfer the correct tool, wobj parameter.
60522	GetEndtoolTorque end coordinate system type error	Undefined end coordinate system type	Modify the GetEndtoolTorque instruction and transfer the correct type parameter of coordinate systems: 0-2.
60523	AccSet parameter exceeds the limit range of 30% ~ 100%.	AccSet acceleration and jerk are limited to 30% ~ 100%.	Modify the AccSet instruction parameter.
60524	Quaternion parameter error	The sum of squares of quaternions of variables should be equal to 1.	Modify the corresponding quaternions to correct value.
60525	OpenDev port error	Available ports [0, 65535]	Modify the OpenDev port to the correct value.
60526	Starting program failed due to robot is in error state. Please check recently error record.	Starting program failed due to robot is in error state. Please check recently error record.	1.Fix RL-Language error or logic error; 2.PPtoMain and restart program
60527	Trigger instruction parameter error	The trigdata set by TrigVar instruction does not exist	Create the target var to modify and set the trigdata with TrigVar instruction.
60528	Trigger instruction parameter error	The DO/GO/register variable set by trigdata does not exist; or the output signal is not set; or the register is not writable	Create the output signal (or a writable register) and set it with TrigIO/TrigReg instruction.
60529	Failed to start RL program. The robot is busy.	1. The robot is running and cannot be started repeatedly; 2. The RL file is being parsed and cannot be started.	Wait for the register function code "sta_robot_is_busy" to turn to 0 before sending the start instruction
60530	Userframe's name is duplicated	The input userframe is duplicated by exist userframe, this userframe may work incorrectly	Please check the userframe list and make sure which frame is needed and delete the wrong frame
60531	Tray data update failed	1. Tray name or workpiece No. Error. 2. Failed to obtain data from tray module. 3. Tray variable value update	1. Check whether the parameter input in the "TrayUpdate" function is incorrect. 2. Contact after-sale.
60532	Failed to obtain workpiece quantity of tray	1.Tray name error. 2. Failed to obtain data from tray module.	1. Check if tray name error occurs. 2. Contact after-sale.
60533	Failed to update palletizing data	1. The stacking name or workpiece	Check whether the parameter input in the

		number is wrong or the layer number is wrong. 2. Failed to obtain data from the palletizing module. 3. Stack variable value update failed.	"PalletUpdate" function is incorrect
60534	Failed to get the number of palletizing layers, the name $arg does not exist	Wrong palletizing name	Check the "PalletLayerCount" function input parameter
60535	Failed to get artifact quantity	Wrong palletizing name or layer number	Please check the "PalletWobjCount" function input arguments
60536	There is no pers value in database, using default value	There is no pers value in database, using default value	There is no pers value in database, using default value
60537	Invalid tool or wobj or userframe	Tool or wobj load error, possible reasons: 1. Tool or tool_load has wrong quaternion parameter; 2. Wobj or its related userframe has wrong quaternion parameter; 3. Wobj failed to relate userframe, need to re-edit wobj.	Edit and update related tools or wobjs or userframes
60538	Movement trajectory param is error	Movement trajectory param is error	Please reset the correct motion parameters.
60600	Failed to obtain parameters during laser welding	Failed to obtain laser welding configuration parameters. Procedure	1. Check whether the laser welding process parameter table is created. 2. Check whether the corresponding parameters in the laser welding process parameter table are correct. 3. An internal laser welding error occurs. Contact the manufacturer
60601	Laser welding setup parameters failed	Error in laser welding setup parameters	1. Check the laser welding port configuration to confirm whether the register or IO configuration of the error signal is wrong
60602	After the laser welding single step, please perform the PPToMain operation first before continuing the operation	After a single step of laser welding, the switch is not allowed to continue running	After a single step of laser welding, it is not allowed to switch to continue running, please perform PPToMain operation first
60603	Laser welding does not support running from the cursor	Laser welding does not support running from the cursor	Laser welding does not support running from the cursor
60604	The laser welding process file does not exist	The laser welding process file does not exist	Please check that the process file in the laser welding instruction has been created
60605	The current moving target point coincides with the laser welding start or end point	The current moving target point coincides with the laser welding start or end point, and the laser welding timing cannot be triggered	1. Check the target point of the error motion command so that the target point does not coincide with the current robot position

		normally	
60606	The laser welding function is turned off, do not use the laser welding command	The laser welding function switch is off	1.Turn on the laser welding function switch 2.Delete the laser welding instruction
60607	Unable to respond to external program start signal, there is currently an error alarm in the controller, which needs to be manually cleared before starting	There is an error alarm in the current controller	Manually clear the alarm before starting
60608	Failed to switch project through socket instruction	1.Switch to non-existent projects, 2.There are non semi static tasks running	1.Switch existing projects, 2.Stop the running task before proceeding with project switching
60609	Successfully switched project through socket instruction	Successfully switched project	Successfully switched project
60610	Currently, there are registers or system IO with program pause function do not reset and do not allow program startup	Currently, there are registers or system IO with program pause function do not reset	Reset the registers and system IO bound with program pause function
60611	Robot is saving diagnose data, can't start run program	Robot is saving diagnose data, can't start run program	Wait 10s and restart again
60612	Configure tool and wobj are conflict, using the last correct configuration	The tool-wobj configured in the upper right corner of the teach pendant (HMI) cannot be calculated by the robot. They are both rob-hold or not rob-hold	Correctly configure the rob-hold tool or rob-hold wobj to avoid conflicts
60613	Configure tool and wobj are conflict and robot cannot move	The tool-wobj configured in the upper right corner of the teach pendant (HMI) cannot be calculated by the robot. They are both rob-hold or not rob-hold	Correctly configure the rob-hold tool or rob-hold wobj to avoid conflicts
60614	Joint position limitation is disabled, can't start run program	Joint position limitation is disabled	Enable joint position limitation
60615	Soft shutdown is triggered, can't start run program	Soft shutdown is triggered	It is not suggested to start program at this state
60700	Command execution of Jodell device failed	The command execution fails due to the abnormal communication between the terminal and the Jodell device	Check the hardware connection and retry the execution
60701	Command execution of RM device failed	The command execution fails due to the abnormal communication between the terminal and the RM device	Check the hardware connection and retry the execution
60702	Command execution failure for opening fourth axis lock	Failure to meet the fourth axis locking opening conditions	Please check if the current angle of the fourth axis is 0 ° or 180 °

		resulting in opening failure	
60703	Command execution failure for closing fourth axis lock	Failure to meet the fourth axis locking closing conditions resulting in closing failure	
60704	Command execution failure for opening SingAreaWrist	Failed to open SingAreaWrist due to not meeting the opening conditions	Please check whether the openning conditions are met
60705	Command execution failure for closing SingAreaWrist	Failed to close SingAreaWrist due to not meeting the closing conditions	
60706	Command execution failure for opening SingAreaJointWay	Failed to open SingAreaJointWay due to not meeting the opening conditions	Please check whether the openning conditions are met
60707	Command execution failure for closing SingAreaJointWay	Failed to close SingAreaJointWay due to not meeting the closing conditions	
60708	Robots that are not in standard six axis configuration or CR six axis configuration are not allowed to use the SingAreaLockAxis4 command	Robots that are not in standard six axis configuration or CR six axis configuration are not allowed to use the SingAreaLockAxis4 command	Do not use the SingAreaLockAxis4 command or replace the robot
60709	Robots that are not in standard six axis configuration or CR, ER six axis configuration are not allowed to use the SingAreaWrist command	Robots that are not in standard six axis configuration or CR, ER six axis configuration are not allowed to use the SingAreaWrist command	Do not use the SingAreaWrist command or replace the robot
60710	Robots with non-standard six axis configurations are not allowed to use the SingAreaJointWay command	Robots with non-standard six axis configurations are not allowed to use the SingAreaJointWay command	Do not use the SingAreaJointWay command or replace the robot
60711	The SingAreaLockAxis4 command is not allowed in non motion tasks	The SingAreaLockAxis4 command is not allowed in non motion tasks	Move the SingAreaLockAxis4 command to a motion task
60712	The SingAreaWrist command is not allowed in non motion tasks	The SingAreaWrist command is not allowed in non motion tasks	Move the SingAreaWrist command to a motion task
60713	The SingAreaJointWay command is not allowed in non motion tasks	The SingAreaJointWay command is not allowed in non motion tasks	Move the SingAreaJointWay command to a motion task
60714	Unable to resume continuous operation, a special emergency stop or collision detection has occurred	1.Emergency stop planning timeout, controller anomaly; 2.Emergency stop or collision detection occurred during the process of returning to the path	PPToMain and restart
60715	Device initialization execution	Check init success	Init successful， motion control can be

	sucessful		performed
60716	Device initialization execution failed	Hardware model mismatch or incorrect connection	Check the connection or if the tool model matches
60717	Command execution of Dh device failed	The command execution fails due to the abnormal communication between the terminal and the Dh device	Check the hardware connection and retry the execution
60800	Unsupported move types	The conveyor wobj only supports MoveL and MoveC motion types	Change to MoveL or MoveC
60801	Conveyor belt, rail, and positioner do not support backward step.	Conveyor belt, rail, and positioner do not support backward step.	Use pptoline or jog instead of backward step.
60802	The current controller state does not allow mode switching.	The current controller state does not allow mode switching.	Try pptoline to reset advance pointer
60803	Task is not a motion task, can not stepback	Task is not a motion task, can not stepback	Choose a motion task for debug in RL editor
60804	Task was finished, can not stepback	Task was finished, can not stepback	Try pptomain or pptoline instead of stepback
60805	Task was finished or the task pointer was in the status that can't change work mode	Task was finished or the task pointer was in the status that can't change work mode	Try pptomain or pptoline to debug
60806	Trajectory first command failed in stepback mode	Trajectory first command failed in stepback mode	Try reset teaching point position
60807	Trajectory move failed in stepback mode	Movement command's arguments was invalid	1.Try reset teaching point position 2.Check RL program's input or logic
60900	Unsupported tool or wobj	The tool and wobj cannot be both handheld or external at the same time	
60901	Robotiq 2F_85 Init failed	1.Abnormal communication between the end and Robotiq device resulted in instruction execution failure;2、Robotiq device ID Error	1.Check the hardware connection and execute it again;2.Enter the correct ID and execute again
60902	Robotiq 2F_85 Get status failed		
61000	Torque threshold setting failed	Condition not met, torque threshold setting failed	
61001	Collision detection not enabled, MotionSup On command cannot be used	Collision detection not enabled, MotionSup On command cannot be used	Enable collision detection
61002	Error getting envelope information	Incorrect getting of envelope information	
61003	Obtaining the nickname of the pallets failed	Function stack number error	Please enter the correct number number
61004	Failed to trigger interrupt. Task	Task does not exist.	

	does not exist.		
61005	Failed to trigger interrupt. Non sports tasks do not trigger interrupts.	Non sports tasks do not trigger interrupts.	Please set interrupts in the exercise task.
61006	Failed to trigger interrupt. There are interrupts that have been triggered but not executed.	There are interrupts that have been triggered but not executed.	Please trigger another interrupt after one interrupt is completed
61007	Failed to trigger interrupt. Single step mode, \DEBUG option not selected, this interrupt will not trigger.	Single step mode, \DEBUG option not selected, this interrupt will not trigger.	If you want to trigger an interrupt in single step mode, please select the \DEBUG option.
61008	Failed to trigger interrupt, in continuous mode pause state, the interrupt is not triggered.	Continuous mode - paused state, the interrupt is not triggered..	If you want to trigger an interrupt in pause mode, select the \ \ DEBUG option and switch to single step mode
61009	Failed to trigger interrupt. Task is resetting, this interrupt will not trigger.	Task is resetting, this interrupt will not trigger.	
61010	Failed to trigger interrupt. Task is executing interrupt, this interrupt will not trigger.	Task is executing interrupt, this interrupt will not trigger.	Please trigger another interrupt after one interrupt is completed
61011	Failed to trigger interrupt. The interrupt function does not exist.	The interrupt function does not exist.	Please check if the interrupt function exists.
61012	Failed to trigger interrupt. Interpreter coroutine is full.	Interpreter coroutine is full.	
61013	Failed to trigger interrupt. Please trigger an interrupt during task execution.	Task runs to the endproc.	Please trigger an interrupt during task execution.
61014	Failed to trigger interrup. Interrupt is turned off, note: if triggered once, it should not respond.	Interrupt closed by IDisable	Please trigger the interrupt after IEable
61015	An instruction that is not allowed to be executed within the interrupt function has been executed	An instruction that is not allowed to be executed within the interrupt function has been executed	Do not execute instructions that are not allowed within interrupt functions
61016	Satisfy waituntil during interruption.	Satisfy waituntil during interruption.	
61017	Trigger interrupt failed		
61018	The program failed to start due to not being synchronized with the controller.	The program changes in the HMI have not been synchronized to the controller, and starting the program via system IO, register function code, or external communication is prohibited.	Please click pptomain or perform a reload operation via the teach pendant before executing the program startup operation.

61019	Inconsistent tool usage caused the recipe to fail during execution	The tool configuration used in the project was changed, but the recipe file was not synchronized and updated	Update the recipe to match the tool used in the project, and then rerun the program
61020	Vibration suppression command execution failed	This model does not currently support the vibration suppression function	
61030	Execution of motion command failed	PathRecStart has been executed and path recording has been enabled. This instruction cannot be recorded	Cannot execute unrecorded motion commands between PathRecStart and PathRecStop
61031	Record path related instruction execution failed		
61032	Failed to update palletizing data	1. The stacking name or workpiece number is wrong or the layer number is wrong. 2. Failed to obtain data from the palletizing module. 3. Stack variable value update failed.	Check whether the parameter input in the "PalletUpdateByUniversal" function is incorrect
61033	Failed to get the number of palletizing layers, the name $arg does not exist	Wrong palletizing name	Check the "PalletLayerCountByUniversal" function input parameter
61034	Failed to get artifact quantity	Wrong palletizing name or layer number	Please check the "PalletWobjCountByUniversal" function input arguments

ROKAE

ROKAE
400-010-8700
www.rokae.com
sales@rokae.com