Product Manual

Robot System Operation Manual

Product Manual

Robot System Operation Manual

V2.3

Suitable for IRC series controllers

Ver.: V7.6

The teaching pendant system supports six languages:
Chinese/English/Korean/Russian/Vietnamese/Japanese

This manual is designed to provide you with product-related supporting information, and its content may be updated at any time without prior notice.

Agilebot strives to ensure the accuracy and timeliness of the manual's content, but the information contained therein does not constitute any express warranty or commitment by Agilebot.

Please note that Agilebot shall not assume any liability for any damages (including but not limited to direct, indirect, incidental, or consequential damages) that may arise from the use, reference, or reliance on the content of this manual.

Under no circumstances shall any part of this manual be reproduced or disseminated without Agilebot's prior written permission.

If you need additional copies of this manual, please contact Agilebot.

We apologize for any inconvenience caused during the use of this manual!

Note: This publication is based on the Chinese version.

This publication uses the International System of Units throughout, and GB represents the Chinese National Standard.

Agilebot Robotics Co., Ltd.

Shanghai, China

Revised

Ver.	Date	Status
V1.0	Feb. 25, 2023	Canceled
V1.1	Apr 3, 2023	Canceled
V1.2	Jul 13, 2023	Canceled
V2.0	Jul 28, 2023	Canceled
V2.2	Jul 24, 2025	Draft
V2.3	May 22,2026	Released

Safety instructions

It is necessary to read and understand the contents described in this chapter before using robots.

In this Manual, the robot system refers to an integrated system integrating the industrial robot and its controller, teach pendant, cables, software and other accessories. So, it is required to fully consider the safety precautions of the user and the system.

Nobody is allowed to modify the industrial robot without authorization from Agilebot Robotics Co., Ltd. Agilebot Robotics Co., Ltd. shall assume no responsibility for any damage to the industrial robot or its components due to the use of any other components (software, tools, etc.) not provided by Agilebot.

Agilebot Robotics Co., Ltd. assumes no responsibility for any consequences caused by misuse of the industrial robot. The misuse includes:

- Use the robot beyond the specified parameter range
- Use it as a carrier for humans or animals
- Use it as a climbing tool
- Use it in explosive environments
- Use it without safety protection

Besides safety precautions in this chapter, this Manual contains other safety instructions, which must be followed as well.

Definition of user

The operators are defined as follows:

- Operator

  Perform power-on/off operation on the robot.

  Start the robot program from the panel board.

- Robot Engineer

  Operate the robot.

  Perform teaching and programming debugging of the robot within the safety fence.

- Maintenance Engineer

Operate the robot.

Perform teaching of the robot within the safety fence.

Carry out maintenance (repair, adjustment, replacement) operations on the robot.

The “Operator” is not allowed to enter the safety fence.

The "Robot Engineer" and "Maintenance Engineer" can carry out operations within the safety fence.

The operations within the safety fence include handling, setting, teaching, adjustment, maintenance, etc.

To carry out the operations within the safety fence, it is necessary to receive professional training on the robot.

When operating, programming and maintaining the robot, the operator, programmer and maintenance engineer must give a safety warning and wear at least the following protective articles.

- Work clothes suitable for operations
- Safety shoes
- Safety helmets

System authority for operators

Operator

Operator’s authorities include:

1. Turn on/off the robot.
2. Use the handheld teach pendant to teach the robot; select, debug, run, start, pause and abort the programs.
3. Switch the currently loaded TF/UF and modify overall velocity parameters through the above status bar on the teach pendant screen.
4. Allow operations, e.g. moving to the target point.
5. Review alarms and reset regular alarms.
6. Perform the operations of I/O status interface and register interface.

Warning

1. The “Operator” is not allowed to enter the safety fence.
2. The operations within the safety fence include handling, setting, teaching,

adjustment, maintenance, etc.

3. To carry out the operations within the safety fence, it is necessary to receive professional training on the robot.

4. When operating, programming and maintaining robots, the operator, programmer and maintenance technicians must be careful and wear protective equipment, such as work clothes, safety shoes and safety helmets suitable for relevant work.

Robot engineer

The authority of the robot engineer includes:

1. All authorities of the operators

2. Setting of robot's zero point, setting of soft limit, establishment and editing of coordinate system

3. I/O configuration and management

4. Communication configuration

5. Creation, editing, revision, deletion and other robot program management functions

6. Creation and setting of various registers

7. Management function of robot program attributes

8. Setting of program launch mode

9. Backup and loading of files

10. Setting of IP/TP IP address of controller main board

11. Setting of system time

Administrator

The authorities of administrator include:

1. All authorities of the operator and robot engineer

2. Software installation and upgrading

3. Management of programmer roles, which can be added, deleted or edited

Definition of safety records

This Manual includes safety warnings to ensure personal safety of the users and avoid any damage to the machine tool and describes them with "Danger" and "Warning" in the main text based on their importance in safety.

In addition, relevant additional descriptions are described as "Caution".

Before use, the user must thoroughly read the precautions described in "Danger", "Warning" and "Caution".

Identification	Definition
	It indicates dangerous situations possibly resulting in serious injury or death to the user during incorrect operation.
	It indicates dangerous situations possibly resulting in mild or moderate personal injury or property damage during incorrect operation.
	It provides additional descriptions outside the scope of danger or warning.

Please read this Manual carefully and keep it secure for easy reference at any time.

Safety of operator

When the robot operates automatically, it is first necessary to ensure the safety of the operator. It is quite dangerous to enter the motion range of the robot during its automatic operation. Measures should be taken to prevent the operator from entering the motion range of the robot.

General precautions are listed below. Please take appropriate measures to ensure the safety of the operator.

1. All operators using the robot system should pass the training courses provided by Agilebot Robotics Co., Ltd.
2. During the operation, it is possible that the robot is waiting for a start signal and is about to start even if it appears to have stopped. Even in such cases, it should be considered that the robot is in motion.
3. Make sure to set up safety fences and gates around the robot system.
4. Set peripheral devices outside the robot's range of motion as much as possible.
5. Arrange locks as needed to prevent personnel other than operators from turning on the power supply of the robot.
6. When conducting individual debugging of peripheral devices, it is important to disconnect the power supply of the robot in advance.
7. When handling or mounting the robot, make sure to follow the correct method shown by Agilebot Robotics Co., Ltd. The operation in the wrong way may lead to

overturning of the robot, causing injury to the operator.

8. After mounting, make sure to operate the robot at a low speed for the first time. Then, gradually raise the speed and confirm if there are any abnormalities.

9. When using the robot for operation, it is important to confirm that there are no persons inside the safety fence in advance. Meanwhile, check for potential hazards and make sure to eliminate potential hazards (if any) before operation.

10. Do not operate the robot in the following situations. Otherwise, it may pose an adverse effect to the robot and also cause serious injuries to the operator.

    a. Flammable environments

    b. Explosive environments

    c. High-radiation environment

    d. Water or high humidity environment

    e. When connecting various stop signals of peripheral devices and the robot, make sure to confirm the stop operation to avoid incorrect connections.

Safety warning label

Both the robot and the controller bear several safety and information labels, which contain important information related to the product. This information is very useful for all persons operating the robot system, e.g. during mounting, maintenance or operation.

The safety labels are only graphical and applicable to all languages.

 Caution

It is required to observe the safety and health signs on the product label. In addition, it is also necessary to comply with the supplementary safety information provided by the system builder or integrator.

Sign	Description
	An electric shock may occur if the internally energized parts of the controller are touched when powered on.

	Operation against the instructions may result in an accident of personal injury or product damage. This is a warning message applicable to certain functional requirements.
	Grounding sign of controller
	Keep your hand away from moving parts, otherwise your hand or fingers may get stuck between the axis and the cover. The robots equipped with telescopic covers do not pose the risk of pinching hands or fingers. Therefore, they do not have this label.
	Never enter the work area while the robot is moving. Otherwise, the robot may collide with the operator. This is very dangerous and may cause serious safety issues.
	Beware of burns due to high temperature.

Stop of the robot

The methods for stopping Agilebot robots include:

E-stop (equivalent to Class 0 stop of IEC 60204-1)

- Power off, namely disconnecting the servo power supply to stop the robot instantly.
- The robot is immediately braked and the servo power supply is disconnected as the same time. The robot stops immediately.

Protective stop (equivalent to Class 1 stop of IEC 60204-1)

- Controlled stop, namely the stopping method by disconnecting the servo power supply after the robot is decelerating.
- Control the robot to slow down and disconnect the servo power supply after 1s.

The stop distance and time of the controlled stop are longer than those of the uncontrolled stop. When the controlled stop is adopted, it is necessary to conduct a thorough risk assessment and analysis of the whole system for the stop distance and time are longer.

Strategies for e-stop and protective stop of the robot

The following three stop functions are specified according to GB 5226.1-2008.

Class 0: Stop (and uncontrolled stop) by simply cutting off the power of the mechanical braking mechanism.

Class 1: Controlled stop, namely apply a power to the mechanical braking mechanism to achieve the stop and remove the power after the stop.

Class 2: Controlled stop, namely applying stored kinetic energy to the mechanical actuator.

The protective stop of the Agilebot robot belongs to Class 1 stop mode.

The e-stop of the Agilebot robot belongs to Class 0 stop mode.

Please see the table below for details.

	E-stop	Protective stop
Occasion	There is a fast and accessible passage for the operator.	It is determined by the rules of safety distance.

Start	Manual	Auto or manual
Performance of safety system	Class 3 in GB/T 16855.1-2008, or determined by risk assessment	Class 3 in GB/T 16855.1-2008, or determined by risk assessment
Reset	Only manual	Manual or auto
Operating frequency	Infrequent; it is used in emergency situations.	Variable; it is used in each cycle or infrequently.
Function	Remove all hazardous energy sources.	Control the protected danger.

Safety functions

Overview to safety functions

The Agilebot robot is provided with the following safety features:

- E-stop
- Enabling device
- External safety device connector
- External e-stop button connector
- External isolator connector
- External limit/stop device connector

E-stop

The Agilebot robot is provided with the following emergency stop devices:

- Emergency stop button on the upper right of the teach pendant
- E-stop button on controller (only available on IRC-I8A-S controller)
- External e-stop device (input signal)

It is required to press this device in case of danger or emergency. The robot should have the following response when the emergency stop device is pressed:

The robot stops in the form of uncontrolled stop (Class 0).

First rotate the e-stop device for unlocking and then turn on the servo power to continue the operation.

Warning

If possibly causing a danger, the tools or other devices connected to the robot must be integrated into the e-stop circuit of the robot. Otherwise, it may cause death or serious bodily injury to the operator or damage to any property.

Enabling device

The enabling device of Agilebot robot is an enabling button located on the back of the teach pendant. It has three positions:

- Unpressed
- Middle
- Fully-pressed (alarm position)

Caution

The robot can be operated in manual mode only when the enabling button is held in the middle position. During teaching operation of the robot or program operation under manual mode, the release of the enabling button may trigger Class 1 stop mode; fully pressing of the enabling button may trigger Class 0 stop mode.

External safety connector

1. **External e-stop connector**

Every workstation, which may control the start of a robot or trigger a hazardous situation, must be provided with an external e-stop device, which is under the responsibility of the system integrator. At least one external e-stop device must be mounted to ensure that an e-stop device is available even when the control TP is unplugged.

The external e-stop device is connected through a safety connector provided to the customer on the controller and is not included in the supply scope of industrial robots.

2. **External isolator connector**

The system integrator must use protective devices to prevent personnel from entering hazardous areas of industrial robots. This protective device is an isolator and must meet the following requirements:

    - Comply with the requirements of EN ISO 14120.

- Can prevent personnel from entering dangerous areas or crossing it easily.
- Do not have or cause any danger.
- Must maintain a specified distance from all hazardous locations.

3. **External limit/stop device connector**

It is a device designed to prevent the robot from getting out of the working range. Do not have or cause any danger.

1. Product overview

This chapter provides an overview of basic composition and various devices of the Agilebot robot.

1.1 Robot system

The robot body is a mechanical body composed of such parts as motor, gearbox, encoder and drive mechanism.

The Agilebot robot system is composed of:

- Robot body
- Controller
- Teach pendant
- Motor and encoder cables
- Software system

The Agilebot robot system is shown in the figure:

1 - Robot body 2 - Controller 3 - Teach pendant

Fig. 1.1 Composition of SCARA Robot

1- Robot body 2 - Controller 3 - Teach pendant

Fig. 1.2 Composition of Six-axis Robot

1.2 Robot models

SCARA Series

Product Categories		Product Series		Payload		Version		Reach		Z-Axis Stroke		Branch Version	
GBT	Agilebot Industrial Robots	S	SCARA Robot	3	3KG	A	1st Generation	400	400mm	Blank	Standard stroke	Blank	Standard Version
		CS	Cylindrical SCARA Robot	6	6KG			420	420mm	.3	300mm	C	Cleanroom Version
				10	10KG			500	500mm				
				20	20KG			600	600mm				
								700	700mm				
								800	800mm				
								1000	1000mm				

PUMA Series

Product Categories		Product Series		Payload		Version		Reach		Branch Version	
GBT	Agilebot Industrial Robots	P	PUMA Robot	7	7KG	A	1st Generation	700	721mm	Blank	Standard Version
				20	20KG	B	2nd Generation	900	901mm	C	Cleanroom Version
								1800	1805mm		

Robot Controller Naming Rules

Product Categories		Technical Features		Standard Axis		Version		Controller Type	
IRC	Industrial Robot Controller	I	Integrated All-in-One Solution	4	4 Axes	A	1st Generation	Blank	Standard
		D	Drive Distributed	6	6 Axes	B	2nd Generation	S	Small Type
				8	8 Axes			C	Compact Type

1.3 Controller

1.3.1 Teach pendant

The teach pendant is an operating device managing the interface between application software and the user. The teach pendant is connected to the controller through a cable. The appearance of the teach pendant is shown in Fig. 1.2.

Enable device

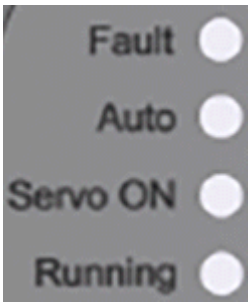
Fig. 1.3 Front and Back of Teach Pendant

Caution

The USB port on the top of the teach pendant cannot be used for backup and restore. Please use the USB port on the controller.

The functions of buttons on the teach pendant are explained in the following table:

Icon	Function
	On/Off button of teach pendant: Press and hold this button to start the teach pendant after the controller is powered on. Release it when the orange lamp of the button is on. After the robot is started up, the On/Off button lamp changes from orange to green.

	Mode selector: Rotate the key to switch among Manual limit speed mode (L), Manual maximum speed mode (M) and Auto mode (A).
	E-stop button: used to brake the robot in emergency situations.
	Status lamps: • Fault status lamp: It is red when the robot has an alarm message and does not light up in case of no alarm. • Auto status lamp: It is green when the robot is in auto mode; otherwise it does not light up • Servo ON status lamp: It is green when the servo motor of the robot is powered on; otherwise it does not light up. • Running status lamp: It is green and flashes when the robot runs a program.
	Press this button to enter current program screen.
	Press this button to enter current position screen.
	Press this button to enter the I/O monitoring screen.
	Press this button to enter the number register screen.
	Press this button to enter the pose register screen.

	Press this button to enter the payload setting screen.
	Press this button to return to the homepage of teach pendant.
	Start button: start and debug robot programs.
	Pause/stop button: pause program operation in execution status; stop program execution in the pause state.
	Jog key: It is used for jog feed; press the jog button to move the robot when there is no alarm, the enable device is kept at the middle gear and the mode selector is in manual mode.
	Confirm the event or error and excite the robot.
	Turn on the servo in auto mode.
	Turn off the servo in auto mode.
	The Multiplier key is used to change the speed multiplier.
Enable device	It has three gears: Unpressed, Middle Holding and Fully-pressed and is used to excite the robot. Note: The servo motor of the robot is powered on in the Middle gear rather than Unpressed and Fully-pressed.

1.3.2 Controller panel

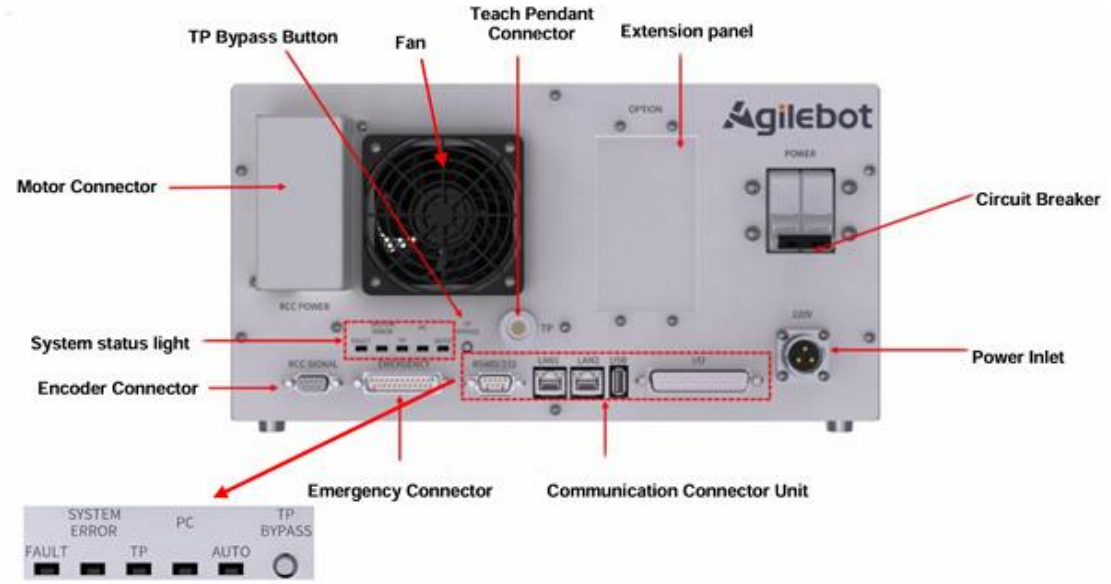


Fig. 1.4 Connectors on IRC-I4A-C Controller Panel

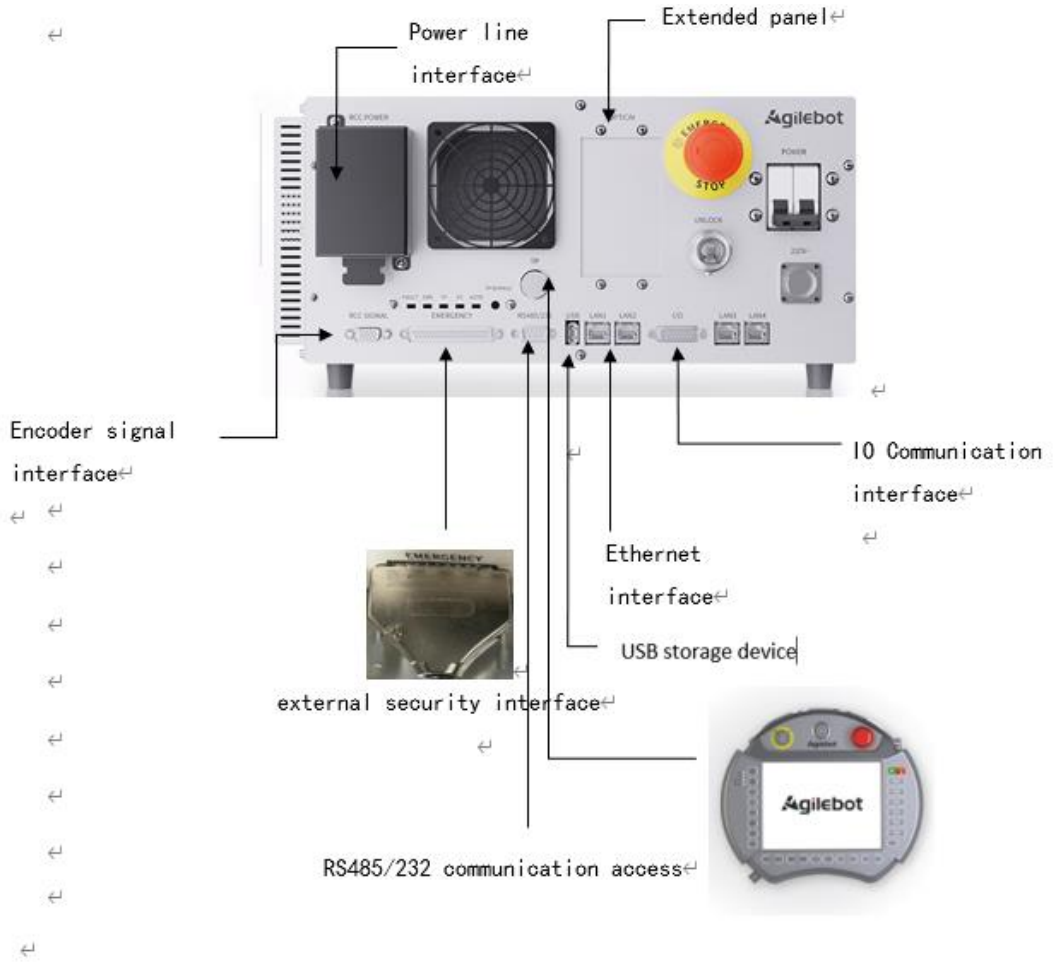


Fig. 1.5 Connectors on IRC-I6A-C Controller Panel

Warning

Except for the TP (teaching pendant) interface, all interfaces on the control cabinet do not support hot plugging. Plugging or unplugging while powered on may cause equipment damage or pose safety risks!

1.3.3 IRC-I4A-C controller without the teach pendant mode

IRC-I4A-C series controllers are not standardly provided with a teach pendant and can be operated by using the virtual teach pendant software "Compass". The system status lamps and TP BYPASS buttons in Fig. 1.4 are described in the table below:

System status lamp and button	Function
FAULT lamp	It is red when the robot reports an error and does not light up if it is normal.
SYSTEM ERROR lamp	The system error lamp lights up when a high-level error occurs, namely a serious error that cannot be cleared or reset by the rest button or rest signal (usually system level).
TP lamp	The light is in green when the controller is inserted into the teach pendant. Otherwise, it does not light up.
PC lamp	The light is in green when the virtual teach pendant software “Compass” is used for operation. Otherwise, it does not light up.
AUTO lamp	The light is in green when the operating mode of the robot is auto run. Otherwise, it does not light up.
TP BYPASS lamp	Press this button to switch the robot to the auto mode after the teach pendant is pulled out, if the robot is in the manual mode before the teach pendant is pulled out.

The teach pendants of IRC-I4A-C series controllers support hot swapping. When the teach pendant is pulled out, the Fault lamp is in red. The robot can be used normally by external reset or by resetting through virtual teach pendant software "Compass". If a teach pendant is inserted into the IRC-I4A-C series controller and it is desirous to operate the robot through Compass, it is required to switch the teach pendant to the auto mode so that Compass can be connected to the controller and have operation authorities; after Compass has the operation authority, the teach pendant loses the operation authority. At this time, the PC lamp is in green, while the TP lamp does not light up; if the teach pendant is in manual mode, Compass cannot be connected to the controller and the teach pendant has operation authority. If no teach pendant is inserted on the IRC-I4A-C series controller, it can be directly connected to the IRC-I4A-C series controller through Compass.

1.3.4 I/O signal

For I/O (I/O signals), universal and special signals can be used to enable the robot control system and external devices to transmit and receive data. The universal signals (user-defined signals) are controlled by programs and used for communication with external devices. The special signals (system-defined signals) are controlled by the system.

I/O can be divided into the following types.

- Digital I/O
- Group I/O
- Special I/O
- Robot I/O

In order to facilitate I/O wiring, the controller is equipped with an external I/O adapter for customers.

1.3.5 Robot actions

For the action of the robot, the motion of the tool center point from the current position to the target position is regarded as an action instruction.

The robot controller uses the motion control system comprehensively controlling the trajectory, acceleration/deceleration, positioning and speed of the robot.

The robot controller can divide multiple axes into multiple action groups for control (multi-action function). The respective action groups are independent of each other, but can synchronously cause the robot to act simultaneously.

There are two types of robot actions: log feed from the teach pendant and action instruction based on the program.

- The action of the robot based on log feeding is executed through the buttons of the teach pendant. The action during jog feed is determined by the manual feed coordinate system and the speed.
- The action of the robot based on action instructions is determined by the position data, action type, positioning type, movement speed, speed multiplier and so forth specified in the action instruction.

The action types include "MOVEJ" (joint), "MOVEL" (straight), "MOVEC" (arc) and door type motion (see Section 3.3.1 for details), which can be selected to operate the robot. When "MOVEJ" is selected, the tool center point moves nonlinearly between two teach points. When "L is selected, the tool center point moves linearly between two teach points. When "MOVEC" is selected, the tool center point passes through the transition point from the starting point to the target point and its motion is controlled in an arc manner. There are two positioning types for the tool center point: "FINE" (positioning) and "SD" (corner radius).

1.3.6 E-stop devices

The robot is provided with the following e-stop devices:

- E-stop button on the upper right of the teach pendant
- E-stop button on controller (available on IRC-I6A-C、IRC-I6A controller)
- External e-stop device (input signal)

The robot stops in any case when the e-stop button is pressed or an external e-stop signal is input.

1.4 Power-on/Power-off

This section mainly explains how to start/stop the robot control system.

1.4.1 Inspection before power-on

It is necessary to ensure mounting correctness to ensure the safety of the robot and the operator during operation. The following steps should be checked for the mounting of the robot:

- Mechanical inspection: Check whether the base is completely fixed and whether the end-effector is firmly mounted.
- Wiring inspection: Check if the motor and encoder cables are wired correctly and firmly.

- Controller inspection: The controller (including power-off switch) should be mounted at a height convenient for operation and maintenance. The recommended mounting height is between 0.6m-1.9m.

1.4.2 Power-on

Power-on sequence: Connect the power cord of the controller and switch the circuit breaker to the Off state . Power on the controller, switch on the circuit breaker and press the Power-on button of the teach pendant. When the yellow lamp of the button is on, release and wait for startup. After normal startup and self-check, the screen shown in Fig. 1.6 is displayed.

Fig. 1.6 Main Screen of Teach Pendant

1.4.3 Shutdown

Shutdown sequence:

1. Press the power switch button on TP until the screen shown in Fig. 1.7 appears, and then click Confirm.

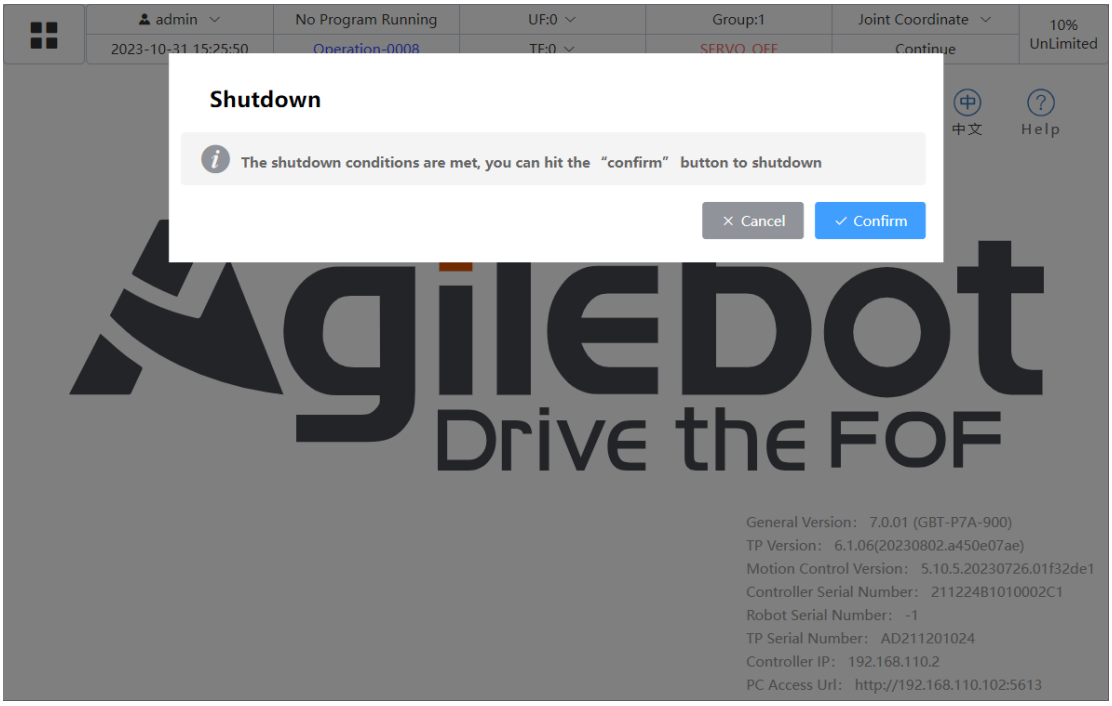


Fig. 1.7 Shutdown Window

2. When the AC-DC Power Supply lamp on TP changes from on to off, it indicates that TP has been turned down. At this time, disconnect the circuit breaker of the controller to complete the shutdown.

Caution

Power on only after the controller status lamp is completely off.

1.5 Introduction to operation interface

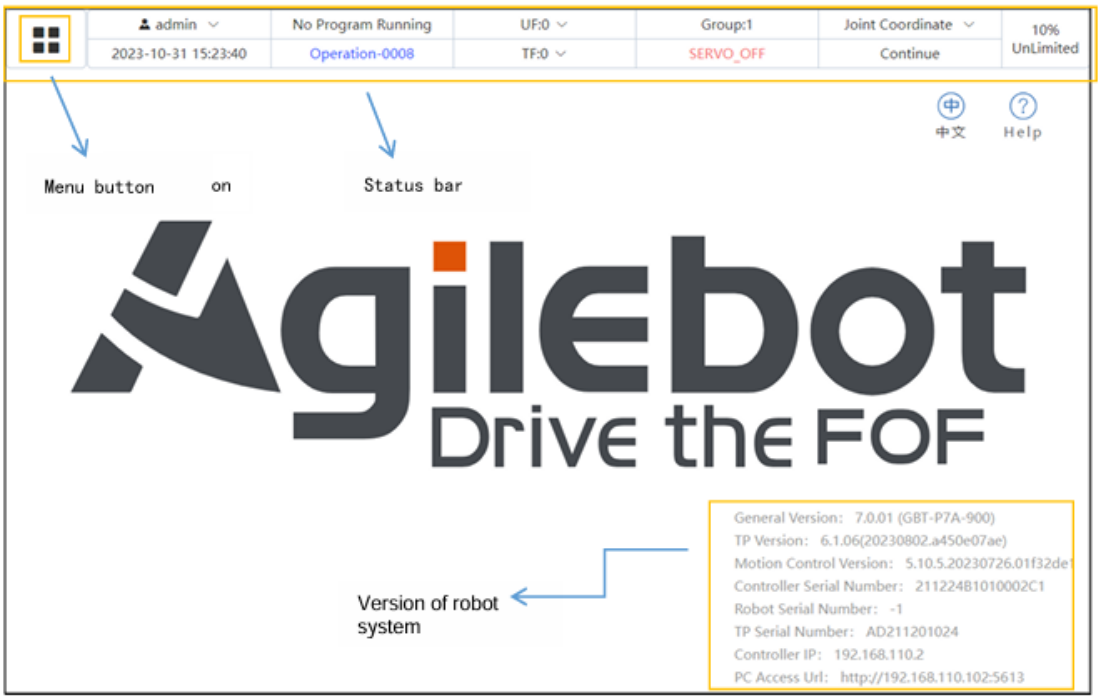


Fig. 1.8 Main Screen of Teach Pendant

1.5.1 Menu

Click the "Menu Button" in the upper left corner of Fig. 1.8 to pop up a menu list. The detailed menu list and sublist are as follows:

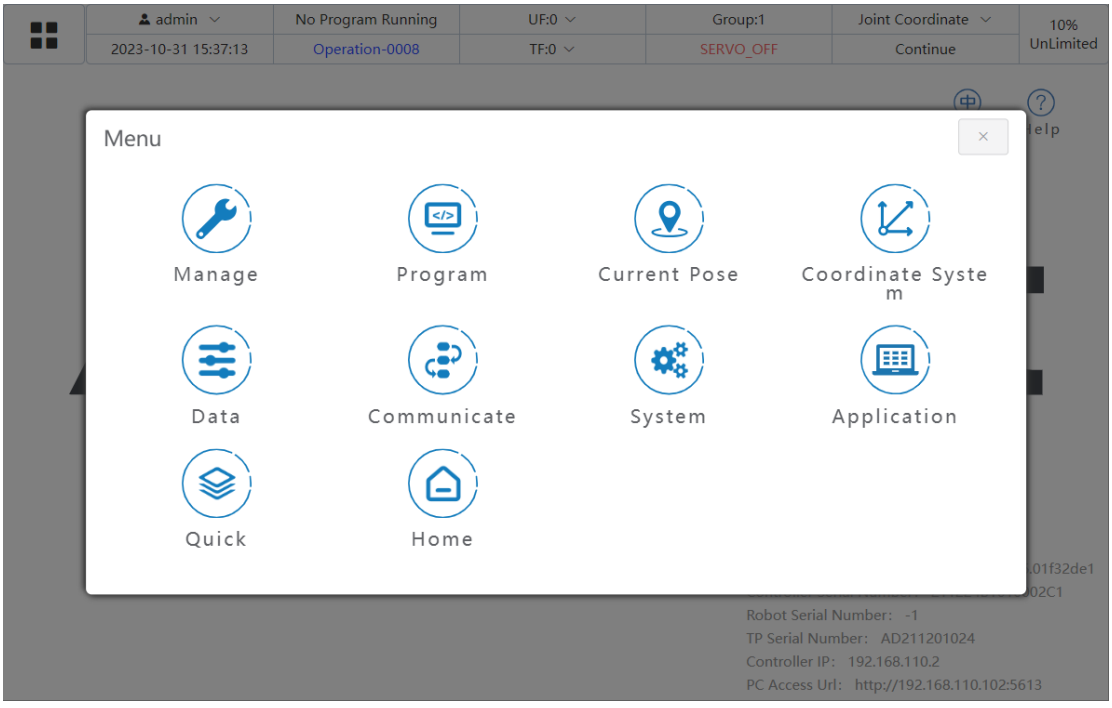


Fig. 1.9 Menu List

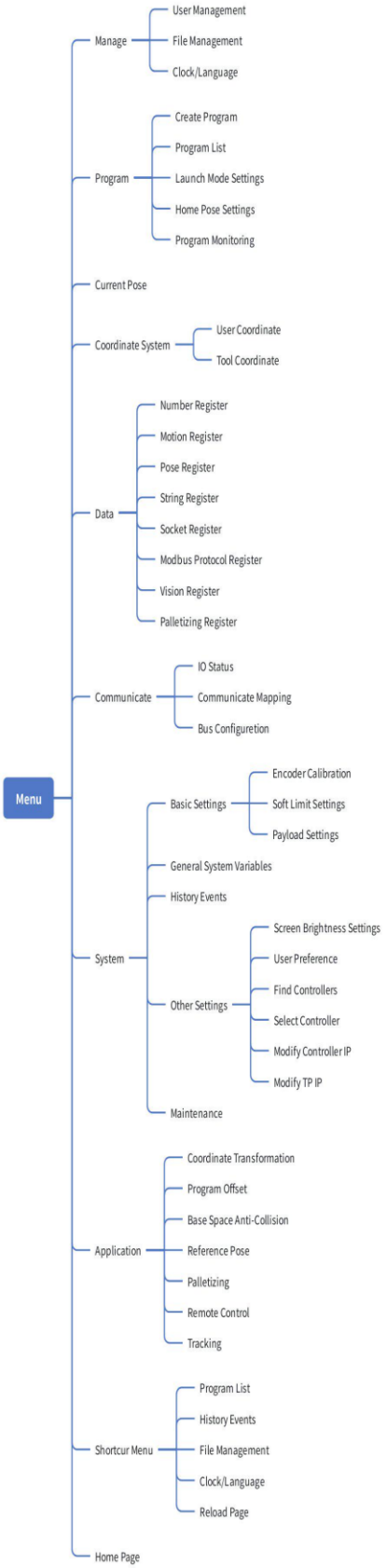

Menu
Manage
User Management
File Management
Clock/Language
Program
Create Program
Program List
Launch Mode Settings
Home Pose Settings
Program Monitoring
Current Pose
Coordinate System
User Coordinate
Tool Coordinate
Data
Number Register
Motion Register
Pose Register
String Register
Socket Register
Modbus Protocol Register
Vision Register
Palletizing Register
Communicate
IO Status
Communicate Mapping
Bus Configuretion
System
Basic Settings
Encoder Calibration
Soft Limit Settings
Payload Settings
General System Variables
History Events
Other Settings
Screen Brightness Settings
User Preference
Find Controllers
Select Controller
Modify Controller IP
Modify TP IP
Maintenance
Application
Coordinate Transformation
Program Offset
Base Space Anti-Collision
Reference Pose
Palletizing
Remote Control
Tracking
Shortcur Menu
Program List
History Events
File Management
Clock/Language
Reload Page
Home Page

Fig. 1.10 Main Menu and Submenu List

1.5.2 Status bar

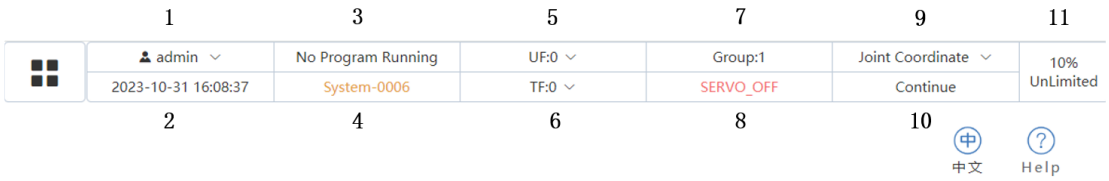


Fig. 1.11 Status Bar

The following table provides specific meaning of each component in the status bar in detail:

Numbers and icons in status bar	Specific meaning
1	User login (see Section 1.1 for specific authority definition)
2	Current system time
3	Display the name of current running program. The current status (run, pause, stop) is on the left side of the program name, while the rightmost number represents the line number of current program. ■ : Program stopped ❚❚ : Program paused ▶ : Program running
4	Alarm events (see Chapter 8)
5	Switching/activation of user coordinate system number (see Section 1.7)

6	Switching/activation of tool coordinate system number (see Section 1.7)
7	Display of motion group, with “1” representing the robot body.
8	Display of servo status, with "ON_STANDBY" indicating that the servo is powered on and "SERVO_OFF" indicating that the servo is not powered on.
9	Switching of current coordinate system (see 1.7)
10	Display of program running mode (continuous run, single-step run)
11	Current overall velocity, which can be adjusted by clicking (see Fig. 1.21 for the interface after clicking)
	It is used to view detailed version information and common problems of the robot control system (see Section 1.5.3).
	Switch of TP Chinese and English 、Vietnamese 、Korean 、Japanese 、Russian interfaces

1.5.3 System information

As shown in Fig. 1.8, current brief version information of the control system is displayed in the lower right corner of the main interface. Click on the Help About in the upper right corner to view more detailed system software version information as shown in Fig. 1.12.

← BackHome

About Agilebot

Version QR Code

	Name	Version Number
1	**Robot Model**	**GBT-P7A-900**
2	**General Version**	**7.0.01**
3	Controller	5.10.5.20230726.01f32de1
4	TP-Charcoal	6.1.06(20230802.a450e07ae)
5	File Service	6.1.06_20230802.a450e07ae
6	Hand Off	19628104
7	IO Board 0	0
8	RBF	21622161
9	Safety Board	21040917
10	Servo Control	JTAC-P7A-Debug-0.12.0-0, Servo-8.7PUMA20221216
11	Servo Param	P7A-900-20220901-00
12	U-Boot	21
13	TP SN	AD211201024
14	Controller SN	211224B1010002C1
15	Body SN	-1

Fig. 1.12 Version of System Software

1.6 Mode selector

The mode selector is a key switch mounted on TP, as shown in Fig. 1.13. The mode selector is used to select the most suitable robot mode based on its action conditions and usage. There are three operation modes: M (Manual) - manual maximum speed mode, L (Limit manual) - manual limit speed mode and A (Auto) - auto mode.

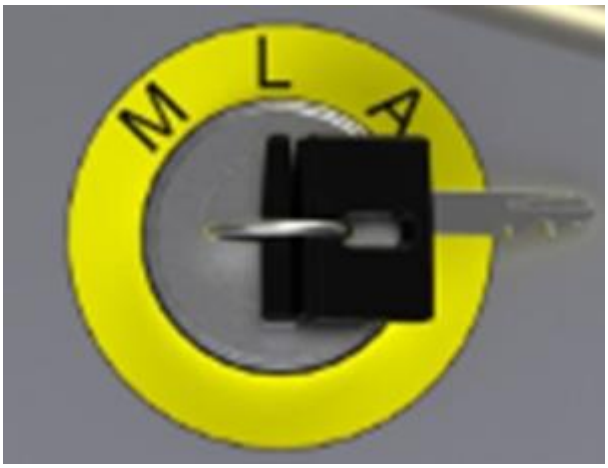


Fig. 1.13 Mode Selector

Switch the operation modes when no program is running. Remove the key from the switch to fix the switch in that position.

1. **Manual maximum-speed mode (M)**

This is a mode for robot program debuggers or operators (hereinafter referred to as the user) to debug the motions of the robot. In this mode, the user can mainly perform the following operations:

- Teach the robot.
- Debug executive programs, including positive-sequence continuous executive programs, positive-sequence single-step executive programs, and reverse-sequence single-step executive programs.
- Edit and modify robot programs.

In this mode, the user is mainly prohibited from performing the following operations.

- Start and execute robot programs through external signals.

2. **Manual limit speed mode (L)**

This is a manual mode where the robot has a limit speed, and its purpose is the same as the manual mode. However, it is only required to adjust and maintain the robot's motion speed below 250mm/s or 18.5°/s to prevent accidents caused by excessive speed during manual operation.

In this mode, the user can mainly perform the following operations:

- Teach the robot.
- Debug executive programs, including positive-sequence continuous executive programs, positive-sequence single-step executive programs, and reverse-sequence single-step executive programs.
- Edit and modify robot programs.

In this mode, the user is mainly prohibited from performing the following operations:

- Start and execute robot programs through external signals.

In this mode, the following restrictions are posed on the motion speed of the robot during program debugging or execution:

- The motion speed of the Cartesian motion instruction is always below 250mm/s.
- The motion speed of the joint instruction is always below 18.5°/s.
- The speed is limited based on the teaching speed at 100% magnification. Therefore, at a teaching speed of 2000mm/s, the speed is limited to 250mm/s if the magnification is 100% and to 125mm/s if the magnification is 50%. Thus, the speed can be further slowed down by lowering the magnification.

3. **Auto mode (A)**

This is the mode for automatic operation of the robot during normal operation. In

this mode, the robot obtains the program information or program number to be executed through communication or I/O and then executes it.

In this mode, the user can mainly perform the following operations:

- Execute the robot program through the launch mode selected in the "Program Launch Mode".

In this mode, the user is mainly prohibited from performing the following operations.

- Teach the robot.
- Debug executive programs, including positive-sequence single-step executive programs and reverse-sequence single-step executive programs.
- Edit and modify robot programs.
- Modify relevant configurations of the robot.

1.7 Robot teaching

Teach the robot according to the joint coordinate system (please refer to 2.3 Coordinate System Settings for the contents of coordinate system).

The single axis operation of the robot is able to independently operate the robot arm to move in either the positive or negative direction of each axis. Specific operation process is as follows:

1. Turn the robot mode selector to L or M mode (manual mode).
2. In the TP status bar, select the reference coordinate system for robot operation as shown in Fig. 1.14.

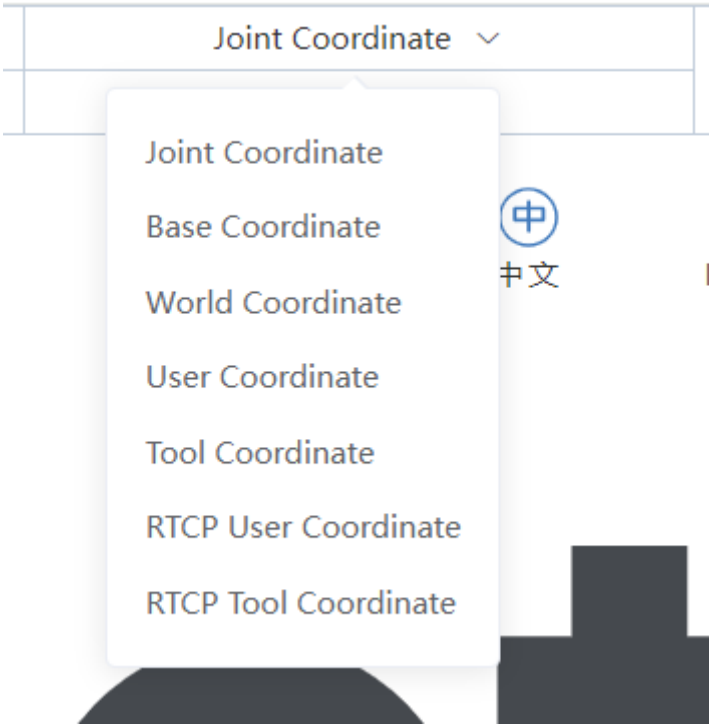


Fig. 1.14 Reference Coordinate System

3. If the reference coordinate system for robot operation is a user or tool coordinate system (if not, ignore this step), it is also necessary to activate the user or tool coordinate system in the status bar. If the coordinate system is activated successfully, a prompt window pops up as shown in Fig. 1.16.

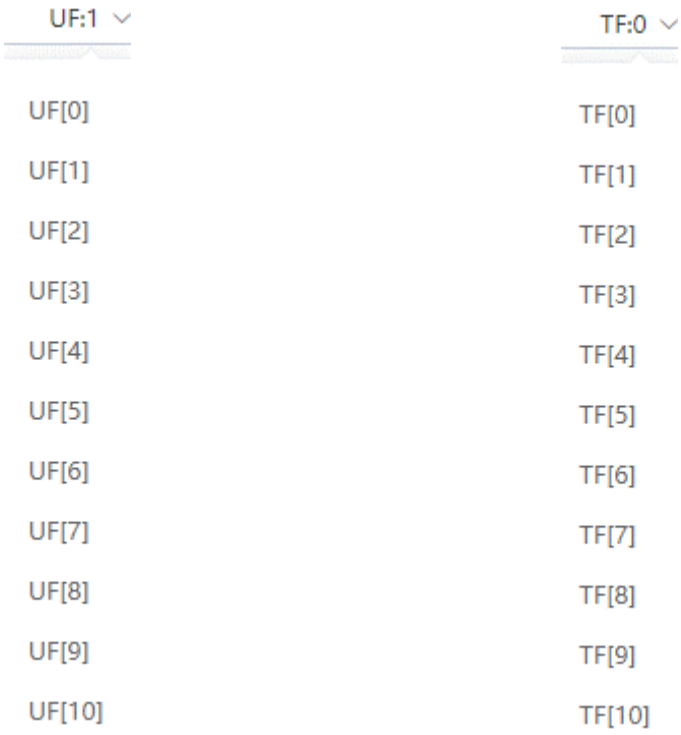


Fig. 1.15 Selection of User or Tool Coordinate System

4. Turn the Enable button on the back of TP to the middle gear, as shown in Fig. 1.17.

Fig. 1.17 TP Enable Button

5. Press the "Reset" button on TP to clear the alarm and observe if the status lamp "SERVO ON" turns green, indicating that the robot servo has been successfully powered on.

6. Press the Jog button to teach the robot movement, as shown in Fig. 1.18 below.

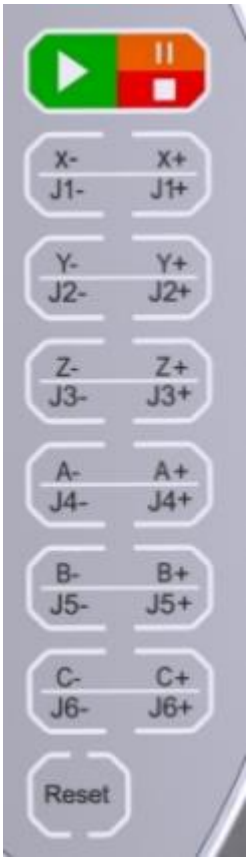


Fig. 1.18 Jog Button of Robot

Caution

When teaching in the joint coordinate system, click the Jog button to control the rotation of each robot axis.

When teaching in the Cartesian coordinate system, jog Buttons X, Y and Z to control the tool coordinate origin of the robot to move along Axes X, Y and Z of the Cartesian coordinate system; jog Buttons A, B and C to control the tool coordinate origin to rotate around Axes X, Y and Z of the Cartesian coordinate system.

Teach the robot with world or base coordinate system.

The base coordinate system is located on the robot base and defined in the factory. The world coordinate system is defined at default to coincide with the base coordinate system. During teaching, the origin of the selected tool or flange coordinate system moves or rotates in the direction of the world or base coordinate system.

Teach the robot with user coordinate system.

During teaching, the origin of the selected tool or flange coordinate system moves or rotates in the direction of the user coordinate system.

Teach the robot with tool coordinate system.

During teaching, the selected tool coordinate system moves or rotates in the direction of the current tool coordinate system. The tool coordinate system spatially changes with its origin. Therefore, every teaching action may change the spatial position and pose of the origin or direction of the tool coordinate system.

Teach the robot with RTCP user/tool coordinate system.

Jogging under external TCP is similar to that in the user coordinate system. They move in X, Y and Z directions of the reference coordinate system. The difference between the jogging pose in external tool coordinate system and that in user coordinate system: for the jogging pose in the user coordinate system, the TCP point rotates around the user coordinate system. for the jogging pose in the external tool coordinate system, the TCP point rotates around the external tool coordinate system.

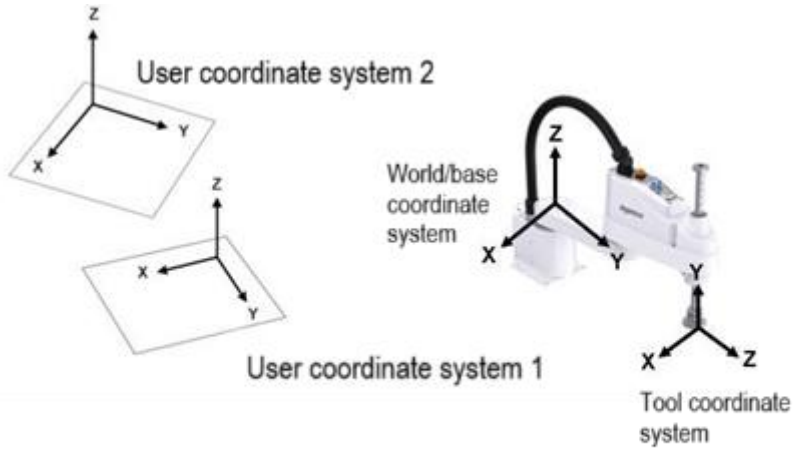


Fig. 1.19 Coordinate System of SCARA Robot

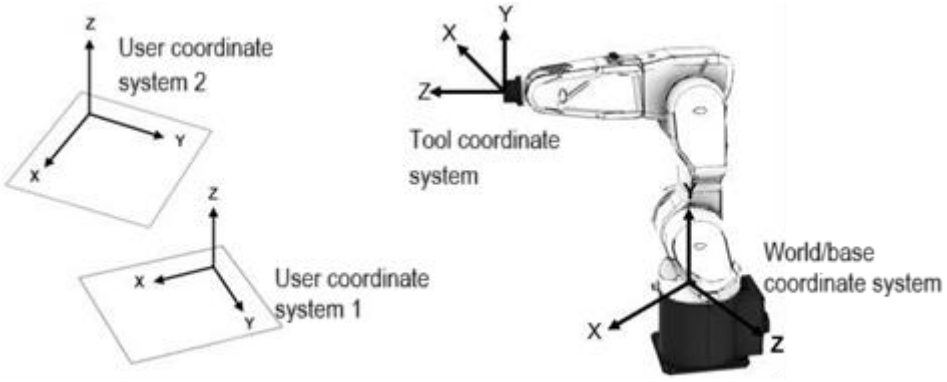


Fig. 1.20 Coordinate System of Six-axis Robot

1.8 Velocity control

Overall velocity control

The overall velocity control function is applicable to the following operation modes: manual limit speed mode, manual mode and auto mode.

After power-on, the overall velocity coefficient is always 10%, regardless of the position of the robot mode selector.

The user can manually input velocity values or pull the velocity slider to adjust the velocity.

- Under running instruction: velocity limit of the robot = instruction velocity * overall velocity coefficient % (maximum speed is 250mm/s or 18.5 °/s in manual limit speed mode)
- During teaching: velocity limit of the robot = 250mm/s * overall velocity coefficient % or 18.5 °/s * overall velocity coefficient %

When the robot is in motion, the adjusted overall velocity takes effect when the next motion instruction is parsed and executed.

Step teaching (using feed rate)

The step teaching function is applicable to: manual (limit speed) mode and manual maximum speed mode. The feeding motion control function is not available in the automatic mode. The overall velocity becomes 10% whenever the operation mode is switched to the auto mode.

After the step teaching function is enabled, the robot only moves at the feed rate of the corresponding step size each time the robot movement button is pressed. The feed rate is divided into two levels: 0.05mm and 1mm (the corresponding joint-coordinate motion step and pose-angle rotation step are tentatively set at 0.5° and 2°). If switching to the auto mode, the user is not allowed to click this button.

Note: When the step motion is planned, the given speed limit is fixed at 250mm/s.

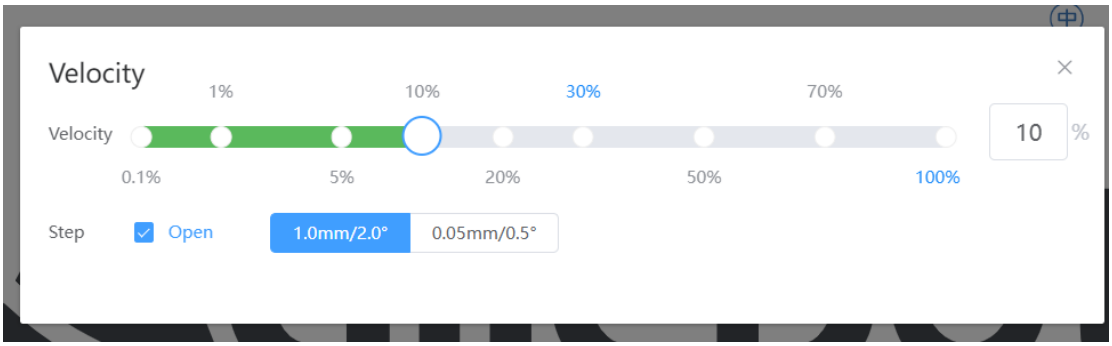


Fig. 1.21 Speed Setting Window

2. Setting of robot system

2.1 I/0 signal

I/O (I/O signal) is an electrical signal used by the robot to communicate with peripheral devices, e.g. end-effector and external devices. It consists of universal I/O and special I/O.

Universal I/O

Digital I/O D I[i]/DO[i]

[i] of I/O represents the logical number of the signal number.

Special I/O

The special I/O is an I/O whose purpose has been determined.

System I/O U I[i]/UO[i]

[i] of I/O represents the logical number of the signal number.

Robot I/O

The robot I/O represents the I/O interfaces on the PUMA robot; The SCARA robot does not have this signal interface.

Robot I/O RI[i]/RO[i]

[i] of I/O represents the logical number of the signal number.

I/O mapping

The universal I/O (DI/O) and special I/O (UI/O, etc.) are referred to as logical signals.

In contrast, the actual I/O signal points are referred to as physical signals. To specify physical signals, use the I/O mapping function to specify the I/O module and use the signal number (physical number) within the I/O module to specify each signal.

Physical number

The physical number refers to the signal number in the I/O module. Express the physical number as follows.

• Digital input signal: Input Port1, Input Port2...

• Digital output signal: Output Port1, Output Port2...

It is required to establish an association between physical signals and logical signals in order to control the I/O signal points on the robot controller. This association is called I/O mapping (configuration).

• For digital I/O and system I/O, the I/O allocation can be changed and the association between physical and logical signals can be redefined.

DI/DO adapter

An I/O adapter is provided to facilitate the wiring and use of I/O signals.

The wiring method of the IRC-I4A-C controller adapter should be suitable for the type of controller. The NPN controller adapter should be wired according to the NPN method, while the PNP controller adapter according to the PNP method. The IRC-I4A-C controller has the control ports of 16 outputs and 24 inputs; it is connected to the adapter with a special signal cable. The appearance of IRC-I4A-C adapter is shown in the following figure.

Fig. 2.1 External I/O Board

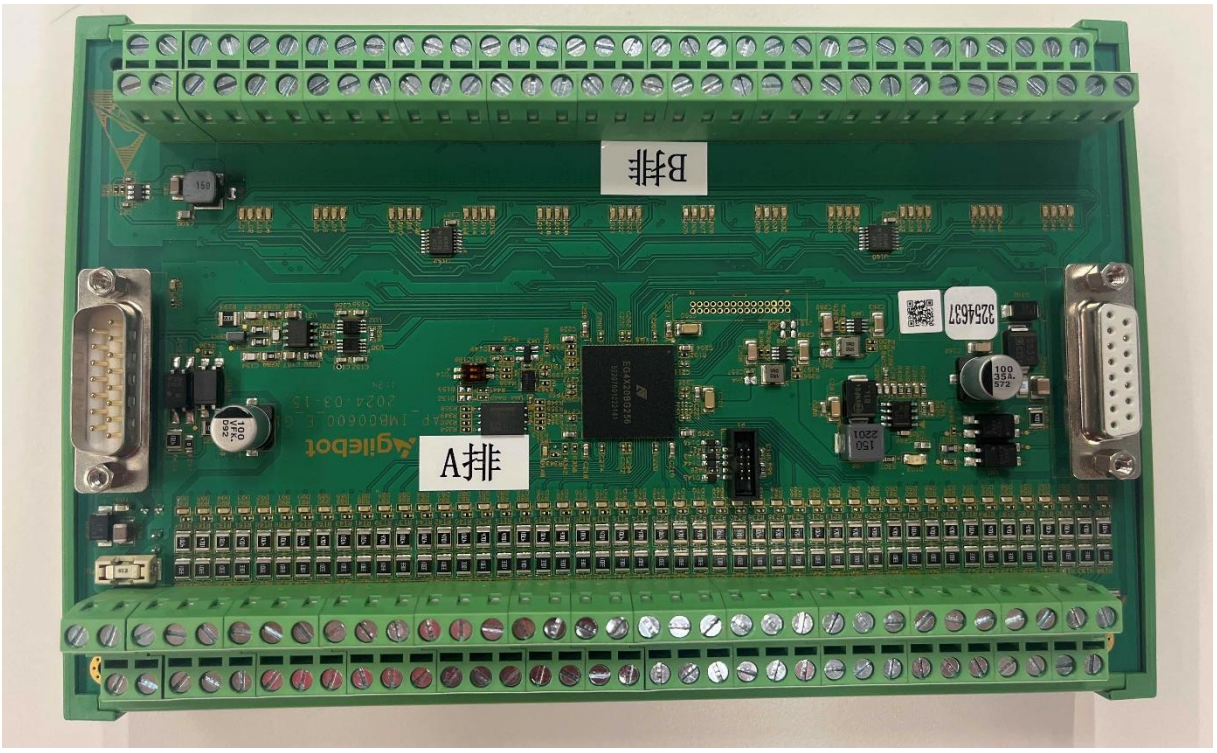
The following table provides a detailed explanation of the default wire sequence definitions for each port on the I/O board: (actual functions can be set according to the operating scenario).

Port No.	Functional/physical number	Port No.	Functional/physical number
01	DI 1-8 common port	26	Input Port16
02	Input Port1	27	Output Port6
03	Input Port2	28	Output Port7
04	Input Port3	29	Output Port8
05	Input Port4	30	Output Port9

06	Input Port5	31	Output Port10
07	Input Port6	32	Output Port11
08	Input Port7	33	NC
09	Input Port8	34	DI 17-24 common port
10	Output Port1	35	Input Port17
11	Output Port2	36	Input Port18
12	Output Port3	37	Input Port19
13	Output Port4	38	Input Port20
14	Output Port5	39	Input Port21
15	DO_PS_IN 1	40	Input Port22
16	DO 1-8 common port	41	Input Port23
17	24V	42	Input Port24
18	DI 9-16 common port	43	Output Port12
19	Input Port9	44	Output Port13
20	Input Port10	45	Output Port14
21	Input Port11	46	Output Port15
22	Input Port12	47	Output Port16
23	Input Port13	48	DO_PS_IN 2
24	Input Port14	49	DO 9-16 common port
25	Input Port15	50	0V

The IR control cabinet IRC-I6A-C has specially developed corresponding I/O adapter boards for the 48-channel DI/DO of the I/O board to assist customers, as shown in the figure below:

Block A、B



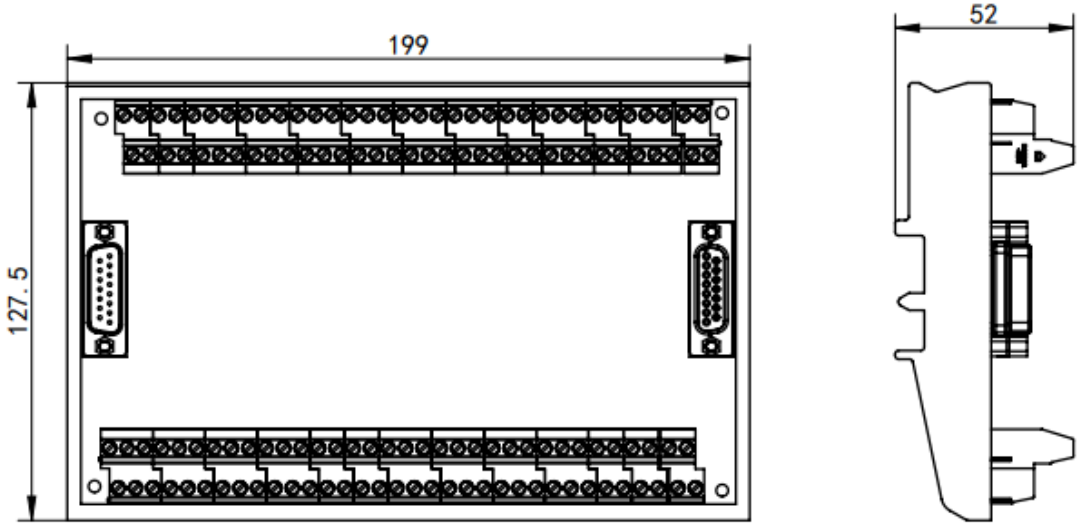


Fig. 2.2 External I/O Board

The I/O adapter board has a total of 4 terminal blocks, with terminal block A being the DI interface and terminal block B being the DO interface. In addition, there is a terminal block for 24V and 0V, each with 4 pins. The I/O board IMB provides a total of 48 DI channels and 48 DO channels.

For detailed setting operations, please refer to the teach pendant's operation manual. This manual only provides instructions on the DI/DO circuit connections.

The pin definitions for row A are shown in the table below:

Port No. of terminal block A	Functional	Port No. of terminal blockA	Function
01	Input Port1	33	Input Port25
02	Input Port2	34	Input Port26
03	Input Port3	35	Input Port27
04	Input Port4	36	Input Port28
05	Input Port5	37	Input Port29
06	Input Port6	38	Input Port30
07	Input Port7	39	Input Port31
08	Input Port8	40	Input Port32
09	Input Port9	41	Input Port33
10	Input Port10	42	Input Port34
11	Input Port11	43	Input Port35
12	Input Port12	44	Input Port36
13	COM_1_12	45	COM_25_36
14	24V	46	24V
15	0V	47	0V
16	0V	48	0V
17	Input Port13	49	Input Port37
18	Input Port14	50	Input Port38
19	Input Port15	51	Input Port39
20	Input Port16	52	Input Port40

Port No. of terminal block A	Functional	Port No. of terminal blockA	Function
21	Input Port17	53	Input Port41
22	Input Port18	54	Input Port42
23	Input Port19	55	Input Port43
24	Input Port20	56	Input Port44
25	Input Port21	57	Input Port45
26	Input Port22	58	Input Port46
27	Input Port23	59	Input Port47
28	Input Port24	60	Input Port48
29	COM_13_24	61	COM_37_48
30	24V	62	24V
31	0V	63	0V
32	0V	64	0V

DI Interface — Pin Definition Table for Row A

The pin definitions for row B are shown in the table below:

Port No. of terminal block B	Functional	Port No. of terminal block B	Function
01	0V	33	0V
02	0V	34	0V
03	24V	35	24V
04	24V_In_1-12	36	24V_In_25-36
05	Output Port1	37	Output Port25
06	Output Port2	38	Output Port26
07	Output Port3	39	Output Port27
08	Output Port4	40	Output Port28

09	Output Port5	41	Output Port29
10	Output Port6	42	Output Port30
11	Output Port7	43	Output Port31
12	Output Port8	44	Output Port32
13	Output Port9	45	Output Port33
14	Output Port10	46	Output Port34
15	Output Port11	47	Output Port35
16	Output Port12	48	Output Port36
17	0V	49	0V
18	0V	50	0V
19	24V	51	24V
20	24V_In_13-24	52	24V_In_37-48
21	Output Port13	53	Output Port37
22	Output Port14	54	Output Port38
23	Output Port15	55	Output Port39
24	Output Port16	56	Output Port40
25	Output Port17	57	Output Port41
26	Output Port18	58	Output Port42
27	Output Port19	59	Output Port43
28	Output Port20	60	Output Port44
29	Output Port21	61	Output Port45
30	Output Port22	62	Output Port46
31	Output Port23	63	Output Port47
32	Output Port24	64	Output Port48

DI Interface — Pin Definition Table for Row B

2.1.1 Digital I/O

Digital I/O (DI/DO) is a standard digital signal for data exchange from peripheral devices by processing input/output signal lines of the I/O units. To put it correctly, it belongs to a universal digital signal. There are two digital signal values: ON and OFF.

I/O mapping (configuration)

Digital I/O can redefine the physical numbers of signal lines and set the following items. For details on I/O allocation, please refer to "2.1 I/O Signals".

The starting port assigns physical numbers to logical numbers to achieve signal point mapping. Thus, the initial physical number can be specified for the allocation.

Caution

The physical number specifies the input/output pins on the I/O module. The logical number is assigned to the physical number. So, a signal can be regarded as a unit to change the allocation.

The starting port of the physical number is set according to actual needs.

Allocation of digital I/O

Successively click "Menu Button" → "Communication" → "I/O Mapping" to enter the screen as shown in Fig. 2.3.

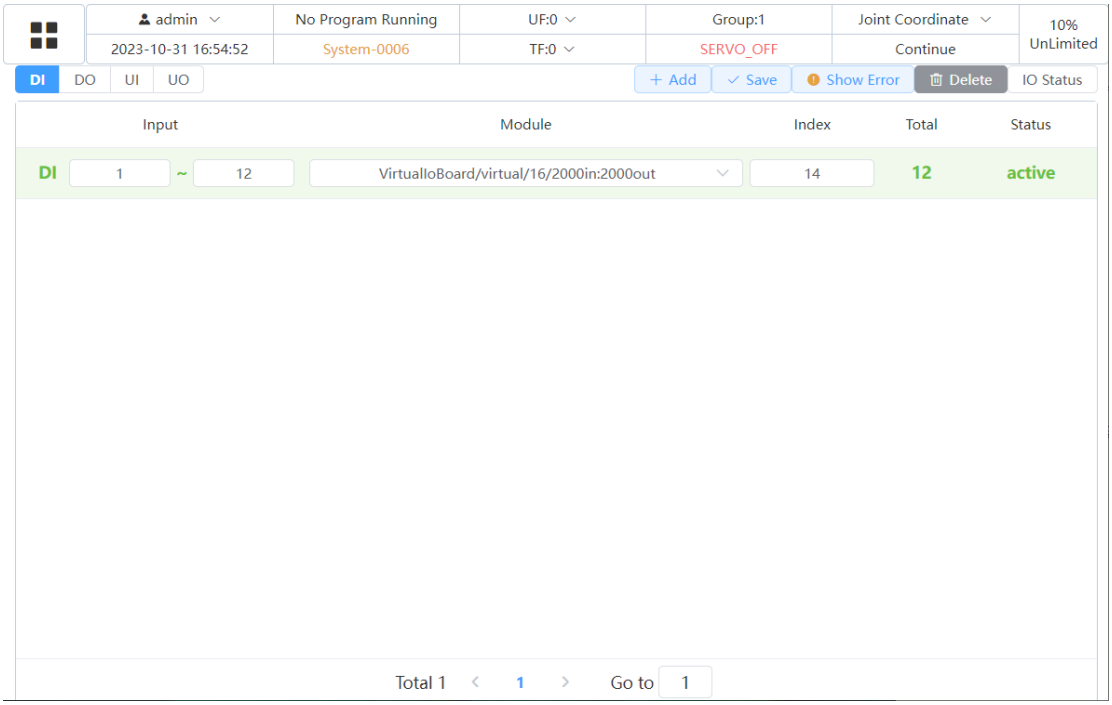


Fig. 2.3 I/O Mapping Interface

Specific meanings of the I/O mapping interface are as follows:

- I/O type: The user can choose it according to the I/O type to be managed and mapped. Currently, the mappable I/O includes DI/DO/UI/UO;
- Input: Display UO/DO or UI/DI (DI in the diagram) according to the "I/O Type" selected by the user. Choose the logical signal ID for I/O mapping here; (method: Set the entire segment, namely, simply input start and end IDs of the logical I/O to be set. For example, in order to set 12 DIs, simply enter 1 at the position of start ID and 12 at the position of end ID.) (Note: The ID segments of the same I/O type cannot overlap.)
- Module number: It allows the user to select the communication module recognized in the current system, as shown in the figure below:

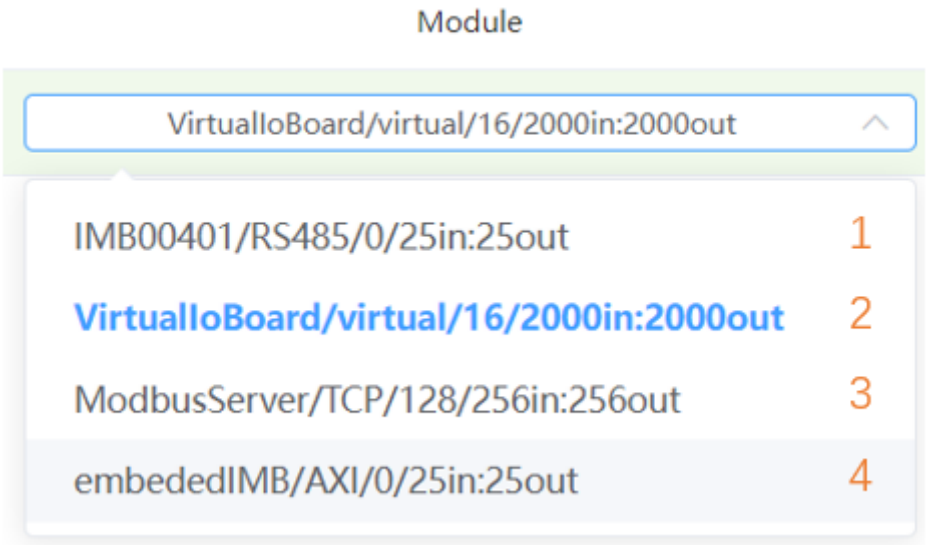


Fig. 2.4 Module Selection Interface

- Module 1 is the I/O adapter of the controller used by a six-axis robot. For the SCARA robot, the module name changes from IMB00401/RS485/0/25in:25out to embedediMB/AXI/024in:16out.
- Module 2 is a virtual device I/O for the user to perform I/O mapping without actual physical I/O modules to facilitate program debugging.
- Module 3 is a modbus tcp module, which can map logical signals to the modbus register; e.g. Mapping DI to Coil Status and DO to Discrete inputs in the modbus register.

Caution

In the module number drop-down list, only communication modules and Virtual I/O Devices recognized in the current system are available for the user to choose. Virtual I/O Device is a virtual I/O device (2000 In/2000 Out) for the user to perform I/O

mapping without actual physical I/O modules to facilitate program debugging.

- Start Port: It is the physical port, on the I/O module, corresponding to the first logical I/O of the input logical I/O segment to be configured. Example: The required I/O segment is DI1-DI5. If the user chooses the corresponding physical port as 3, the relationship is that DI1-DI5 correspond to physical ports "Input Port3-Input Port7" one-to-one. Namely, if the physical port "Input Port3" has a signal input, the DI1 status is ON; so do the states of DI2-DI5.
- Total number: It is the number of I/O automatically calculated based on start and end IDs (number = end number - start number + 1); (for example, DI2-DI5 corresponding to the I/O number of 5-2+1=4). It is read only and cannot be operated by the user.
- Status: Display the status of each entry in the I/O mapping; there are four types: Active, Unfindable, Modifying and Invalid.
- Active means that the I/O configuration is in an active state (successfully configured and saved).
- Unfindable indicates that the physical module cannot be found (this situation specifically refers to the original active state caused by the disconnection of the physical module).
- Modifying/To be Saved represents the modified I/O configuration, which is legal but has not been saved yet.
- Invalid indicates that the configured content of the entry is invalid and contains missing items (illegal) of errors or address conflicts. The user cannot successfully save it (other entries in the Modifying/To be Saved state can be saved).
- Add: Click this button to create a new I/O configuration entry.
- Save: Click to save the mapped content.
- Delete: Click to delete the currently-selected I/O configuration entry.
- Display error: Display current error and make modifications according to its prompt content.
- I/O status: Click here to switch to the I/O status interface

Output

Set the output value of the robot through program execution or manual operation.

See Section 2.2 for forced output and simulated input of signals.

Warning

The controller controls peripheral devices through signals. Forced output/simulated input may pose adverse effects on the safety of the system in some cases. Never perform forced output or simulated input before confirming the usage and function of the signal.

2.1.2 Group I/O

Group IO (GIGO) is a general-purpose digital signal used to aggregate multiple signal lines and perform data exchange. The value of a group signal is expressed by a numerical value (in decimal), which is converted to or reversed from a binary number for data exchange via signal lines.

I/O Mapping (Configuration)

Group IO can define signal numbers as a single group. It is possible to define 2 to 16 signals as one group

2.1.3 Special I/O

System I/O (UI/UO) is a special signal whose purpose has been determined in the system, namely system I/O signal. These signals are connected from the I/O module to the remote controller and peripheral devices and the robot is controlled externally. Refer to Section 2.1.1 for mapping contents of special I/O.

The special I/O, namely system I/O, is described in the following:

List of UI/UO signals						
UI[1]	Servo_Enable Servo enable signal (it can be used as an alarm signal of instantaneous stop peripheral	Servo_Enable is usually ON. When the peripheral upper computer does not want the robot to move or when power is switched on, it is switched to OFF. It is used for safety locking. In the OFF state, the system performs the		UO[1]	CMDENBLE Allow peripheral devices to control the status signals of	ON indicates that peripheral device control is enabled, while OFF means that peripheral device control is disabled. Output high level when the following conditions are met: 1. UI[5] is ON. 2. The mode

	software ; or after pausing, it turns off the servo-holding brake to make a complet e stop)	following processing: 1. Issue an alarm and then disconnect the servo power supply. 2. Instantly stop the robot (Class 0 stop) and suspend the execution of the program. 3. The servo cannot always be enabled. The bypass is ON.			the robot.	selector is in "Auto" mode. 3. UO[3] is OFF.
UI[2]	Pause_ Request	Pause signal. It is usually ON. In the OFF state, the system performs the following processing: It is planned to slow down and stop the executing action and to suspend the execution of the program. The bypass is ON.		UO[2]	Paus ed	"Paused" status signal. When the program execution status is "Paused", this signal is ON (i.e. the robot is paused).
UI[3]	Reset Alarm reset signal	Release the alarm, power on the servo and effectively generate a Reset request at a high level.		UO[3]	FAUL T	When an alarm occurs in the system, this alarm signal is output and can be reset by RESET. Note: This signal is not output when the system issues a warning type alarm.

UI[4]	Start & Restart Program launch/resume signal	Start or restart the program (depending on whether the program status is “Aborted” or “Pause”) and its function is the same as the Start button on TP. Take the effective falling edge to start or restart the program.		UO[4]	Program Running Program running signal	ON indicates that the program is running; OFF indicates that no program is running.
UI[5]	Abort Program Program abort signal	Request to terminate a program in execution or paused state. It is usually ON. In the OFF state, the system performs the following processing: 1. The alarm bar indicates a program abort request and the program enters the abort mode. If the program is still running, immediately stop the robot's action and then abort the program. It is similar to an "aborted" alarm. 2. Allow to enable and teach the servo, but not to		UO[5]	Servo Status	This signal is set to high level when the robot operation status is "Working", "On Standby" or "Servo ON". It is at lower level under “Servo-OFF”.

		manually or automatically execute programs. The bypass is ON.				
UI[6]	Selection Strobe Trigger signal	It is only valid when the "Program Launch Mode" is set to "MPLCS" or "MPLCS Simple Trigger". Read the trigger signal for selecting the program to be executed. When it is ON, read the input of Program Selection 1-6 and select the program to be executed. Note: This signal is ignored when a program is executing (running or paused).		UO[6]	Selection Check Request	It is only valid when the "Program Launch Mode" is set to "MPLCS" or "MPLCS Simple Trigger".
UI[7]	MPLCS Start	It is only valid when the "Program Launch Mode" is set to "MPLCS" or "MPLCS Simple Trigger". It is a start signal of program number selection.		UO[7]	MPLCS Start Done	It is only valid when the "Program Launch Mode" is set to "MPLCS" or "MPLCS Simple Trigger".
UI[8]-UI[13]	Program Selection 1-6	It is only valid when the "Program Launch Mode" is set to "MPLCS" or "MPLCS Simple		UO[8]-UO[13]	Selection Confirm 1-6	It is only valid when the "Program Launch Mode" is set to "MPLCS" or

		Trigger". The 6-digit binary number of the program number is converted to a decimal number, which is the start number of the main program to be executed.				"MPLCS Simple Trigger". After receiving the Selection Strobe signal, the robot controller may read the status of UI[8]-UI[13] and feed it back to the upper computer for confirmation.

2.1.4 Robot I/O

Robot I/O is a digital signal used as end-effector I/O by the robot. The end-effector I/O is used after being connected to the connector attached to the robot's wrist.

The end-effector I/O (I/O interface on robot body) consists of the universal signals of 6 inputs and 6 outputs.

RI [1~6] input

RO [1~6] output

The I/O interfaces on the body is on four axis arms, with two interfaces, EE1 and EE2.

Pin No.	Function
01	RO1
02	RO2
03	RO3
04	RO4
05	RI1
06	RI2

07	I/O_24V
08	I/O_0V
09	/

Definition of EE1 aviation plug

Pin No.	Function
01	RO5
02	RO6
03	RI3
04	RI4
05	RI5
06	RI6
07	I/O_24V
08	I/O_0V
09	/

Definition of EE2 aviation plug

Caution

There is no I/O interface on the body of SCARA robot. Namely, the SCARA robot does not have RO/I.

2.1.5 Manual control of I/O

The manual control of I/O involves signal interaction with peripheral devices before executing the program.

The manual control of I/O refers to the following entries.

- Forced output
- Simulated input

Successively click "Menu Button" → "Communication" → "I/O Status" to enter the I/O status screen as shown in the following figure.

I/O type

Fig. 2.5 I/O Status Screen

In this interface, it is possible to view the status of I/O, force the output signal and simulate the input signal of digital I/O. Write the name of the signal (supporting Chinese) in the "Name" box of the above interface.

Choose the I/O type of manual control

Click the I/O type to choose the I/O type to be operated, as shown in the following figure.

Fig. 2.6 I/O Type Switching Window

Forced output

For the forced output, manually switch the digital output signal to ON/OFF.

Click the icon in the yellow box of the I/O status interface below, and NO/OFF will pop up. At this time, choose the status to be forced to complete the forced output.

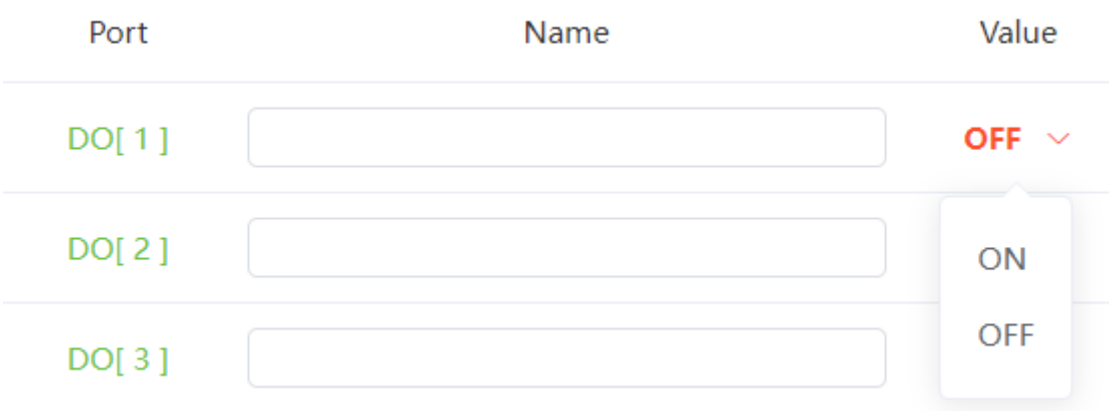


Fig. 2.7 DO Forced Output Window

Input signal simulation

Provide the input signal simulation function for the convenience of actual debugging. Click "sim" to enable the simulation function and "UnSim" to disable the simulation function. As shown in the figure below, the ON/OFF operation can be performed on the input signal of the enabled simulation function; click the "Cancel All Simulations" button to cancel all simulated signals.

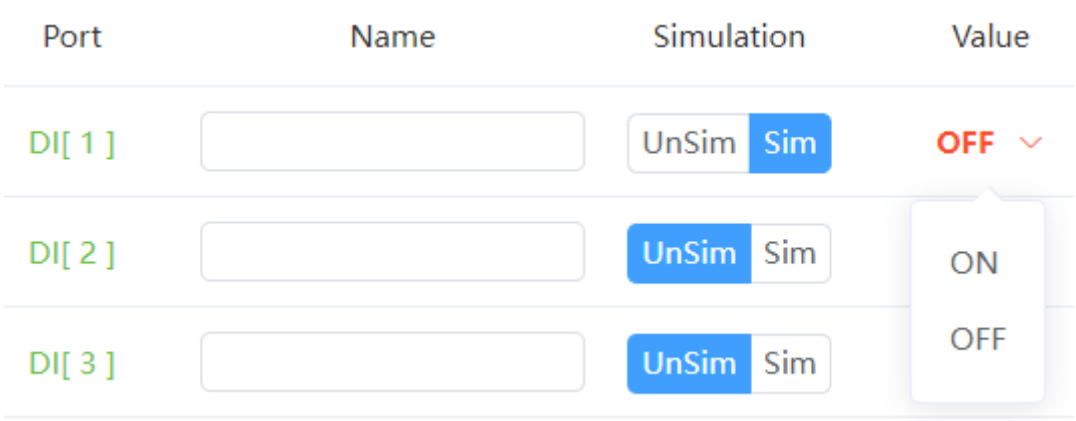


Fig. 2.8 DI Simulation Window

Bypass function of special signal UI

UI has a bypass function. When the bypass function is not activated, the UI value should be its true value (the bypass function can be only operated on UI[1]/UI[2]/UI[5]).

Fig. 2.9 System Signal Window

2.2 Setting of coordinate system

The coordinate system is a position indicator system defined on the robot or space to determine the position and pose of the robot. The robot has multiple coordinate systems as follows.

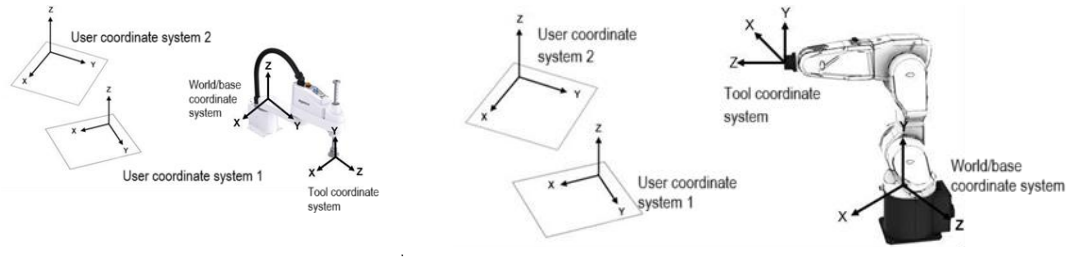


Fig. 2.10 Coordinate Systems of Robots

Joint coordinate system

The joint coordinate system is set in the joints of the robot. In the joint coordinate system, the positions and poses of the robot are determined based on the joint coordinate system on the base side of each joint.

Cartesian coordinate system

In the Cartesian coordinate system, the positions and poses of the robot are defined by the coordinates X, Y and Z from the origin in space to the origin (tool center point) on the tool side, as well as the rotation angles A, B and C relative to the tool side around the X, Y and Z axes in space.

Tool coordinate system

This is the coordinate system defining the position of the tool center point (TCP) and the tool pose. The tool coordinate system must be defined in advance. When not, the default is the end flange coordinate system.

World coordinate system

It is the standard Cartesian coordinate system fixed in space and coinciding with the base coordinate system.

Base coordinate system

It is the standard Cartesian coordinate system fixed to the base of the robot and the user coordinate system is defined accordingly.

User coordinate system

The user coordinate system is a Cartesian coordinate system defined by the user for each workspace. If not, it is replaced by the world/base coordinate system.

2.2.1 Setting of tool coordinate system

The tool coordinate system is a Cartesian coordinate system defining the tool center point (TCP) and the tool pose. In the tool coordinate system, TCP is usually taken as the origin and the tool direction as Axis Z.

The tool coordinate system is composed of the position of the tool center point (TCP) (X, Y, Z) and the tool pose (A, B, C).

The position of the tool center point (TCP) is defined by coordinates x, y and z relative to the default coordinate system of the tool center point.

The tool pose is defined by the pose of the tool coordinate system relative to the end flange coordinate system.

The tool coordinate system is defined on the coordinate system setting screen. It is allowed to define 10 tool coordinate systems, which can be switched according to the situation.

Advantages of tool coordinate system:

- It can allow the tool to move linearly in the direction of the tool coordinate system.
- The tool can rotate around the tool coordinate system, the position of the tool's working point is kept unchanged while the direction is changing, so that the robot pose is changing accordingly.
- The velocity set by the program is the actual velocity of the tool working point rather than the default velocity at the end of the flange.
- The default tool coordinate system is the end flange coordinate system.

Setting method

- 4P method
- 7P method (only applicable to PUMA robots)
- Direct write method

Caution

Before setting the tool coordinate system, it is required to switch the current coordinate system to the base coordinate system and teach the poses under the base coordinate system.

Introduction to tool coordinate system interface

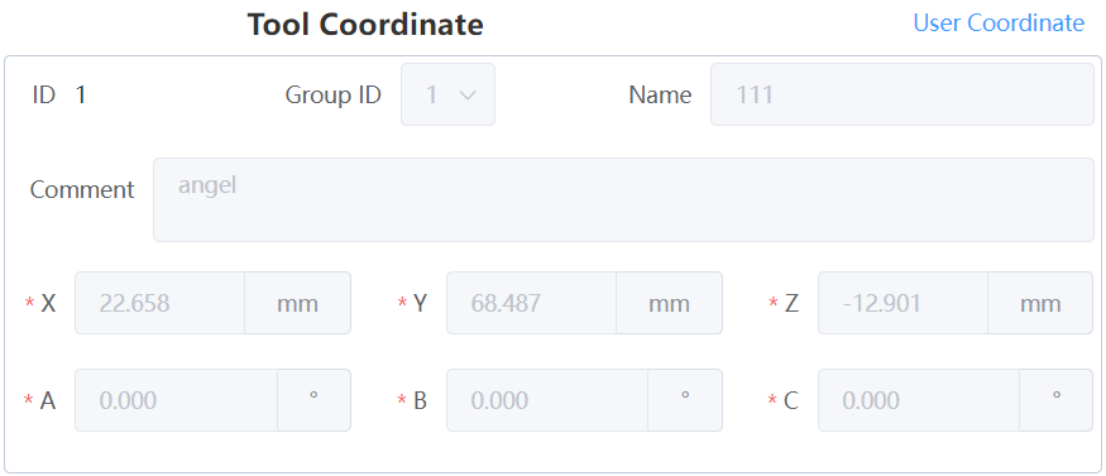


Fig. 2.11 Tool Coordinate System Setting Window

- User coordinate system: Click it to switch to the user coordinate system setting interface.
- ID: number of the currently selected tool coordinate system
- Group ID: motion group, currently supporting the robot body, with the body Group ID represented by 1.
- Name: It is allowed to enter the name of tool coordinate system.
- Note: Notes can be made for the tool coordinate system.

7P setting of tool coordinate system for PUMA robot

The 7P method is adopted to set the tool coordinate poses as shown in Fig. 2.12. The control poses P1-P5 should always contact with the sharp points of the calibrator and the poses of P1-P4 should be as different as possible. When teaching P5, the end of the tool must be linear with the calibrator. P6 is used to determine the direction of tool coordinate X and P7 to determine the direction of tool coordinate Z. During teaching, P7 can move a distance towards tool coordinate Z based on P6.

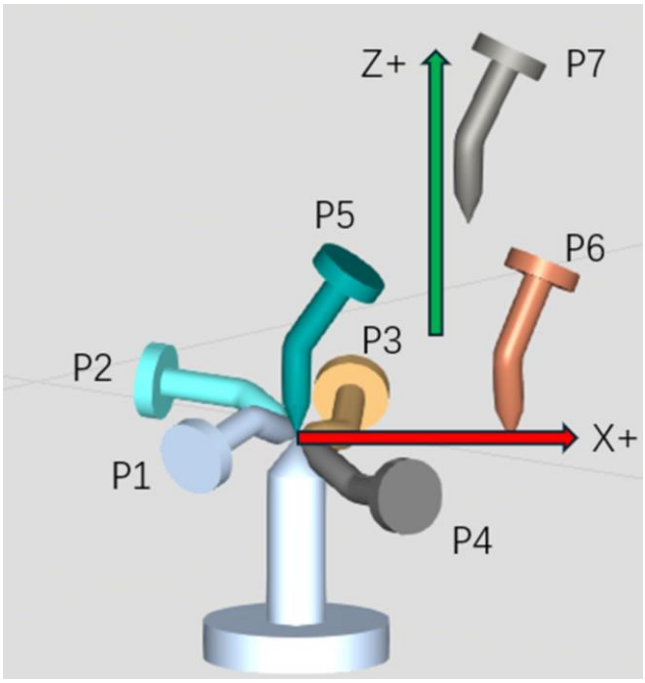


Fig. 2.12 7P Setting of Tool Coordinate Poses

Specific steps:

1. Click "Menu Button" → "Coordinate System" → "Tool Coordinate System" to enter the tool coordinate system setting interface, as shown in Fig. 2.13.

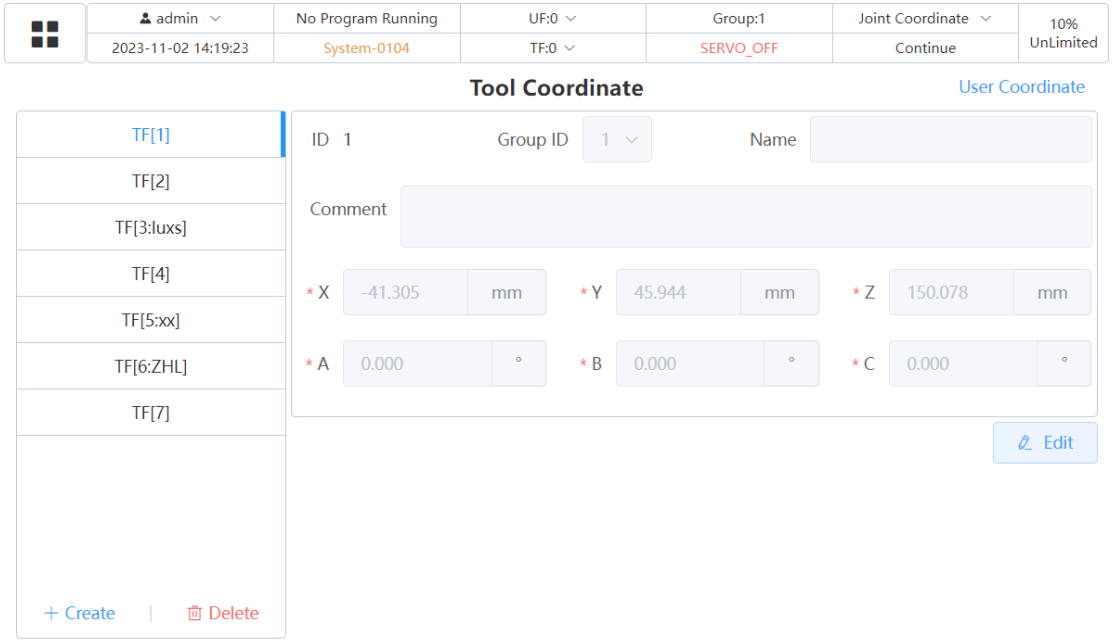


Fig. 2.13 Tool Coordinate System Setting Interface 1

2. Click "New" or select the tool coordinates to be set → click "Edit", and the interface is shown in Fig. 2.14.

Click here to select the 7P method, and the interface is shown in Fig. 2.15.

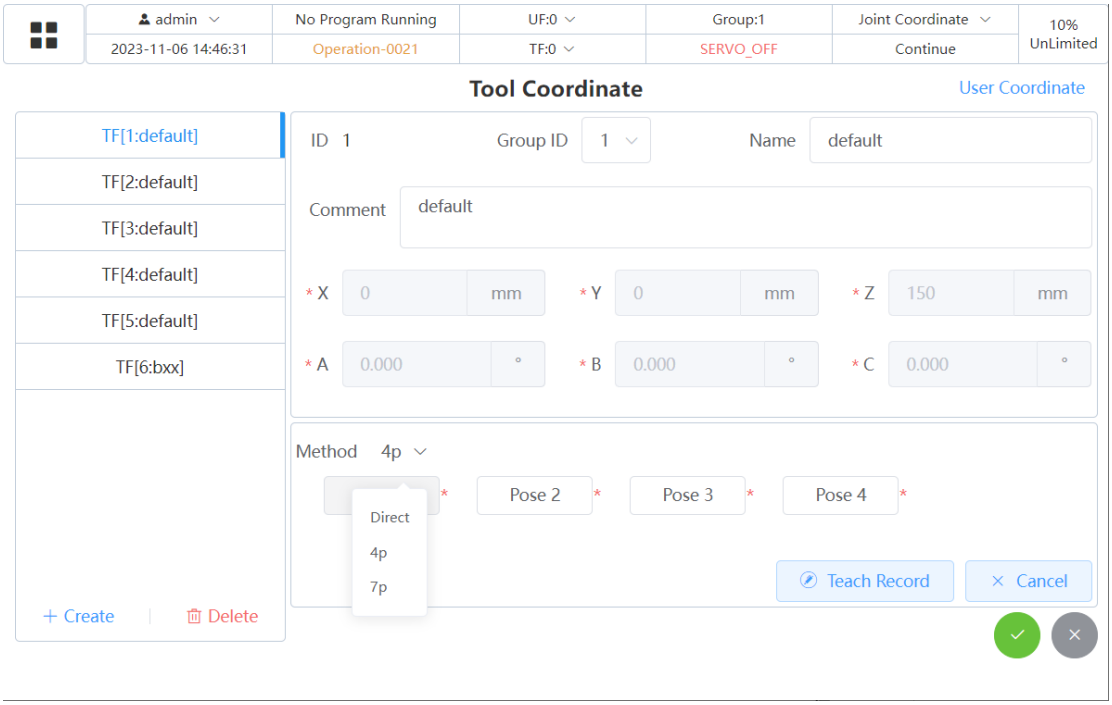


Fig. 2.14 Tool Coordinate System Setting Interface 2

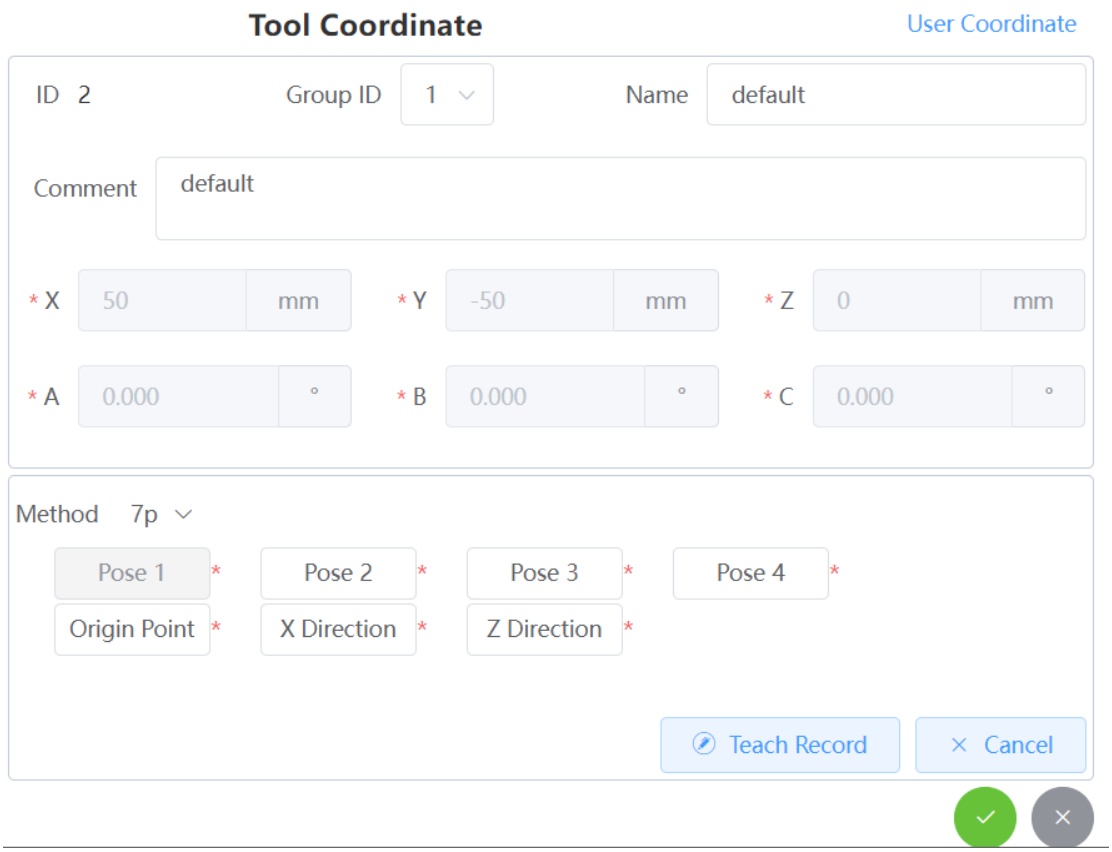


Fig. 2.15 Tool Coordinate System Setting Interface 3

3. Select the pose to be taught. After teaching of that pose, click "Teach Record", and the interface is shown in Fig. 2.16.

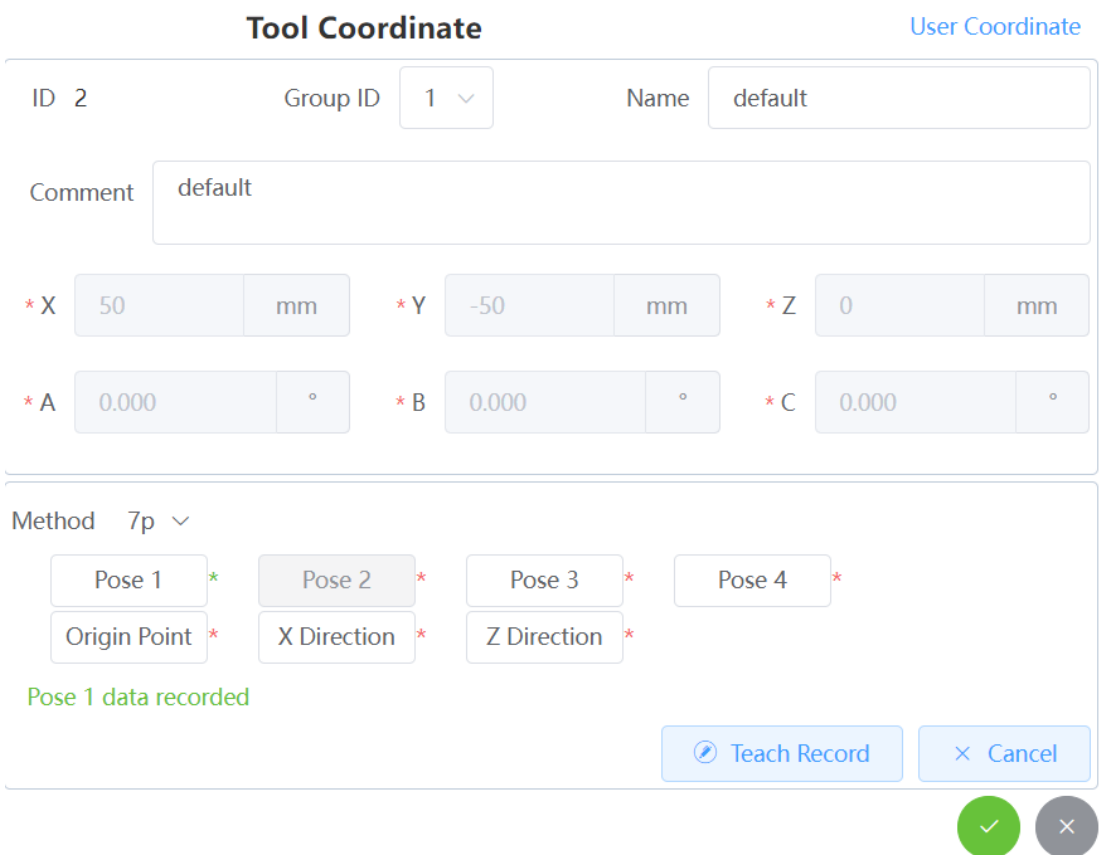


Fig. 2.16 Tool Coordinate System Setting Interface 4

After successful pose recording, a prompt of "Pose x data recorded" will be displayed and the next pose can be taught at this time.

After pose recording, "*" may change from red to green.

4.Teach 7 poses in sequence according to the requirements of Fig. 2.12.

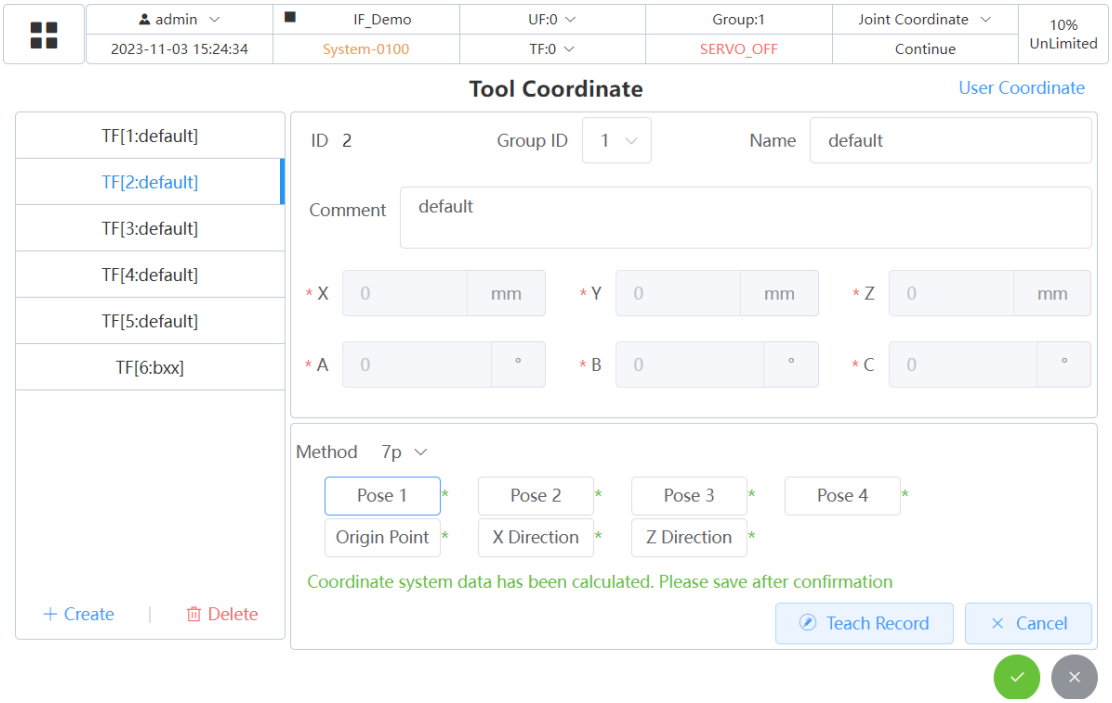


Fig. 2.17 Tool Coordinate System Setting Interface 5

5. After teaching of 7 poses, a prompt will be displayed stating "Coordinate system data has been calculated. Please save after confirmation". At this time, click in Fig. 2.17, and it may prompt "Coordinate system saved successfully", as shown in Fig. 2.18. At this time, the tool coordinate system has been set.

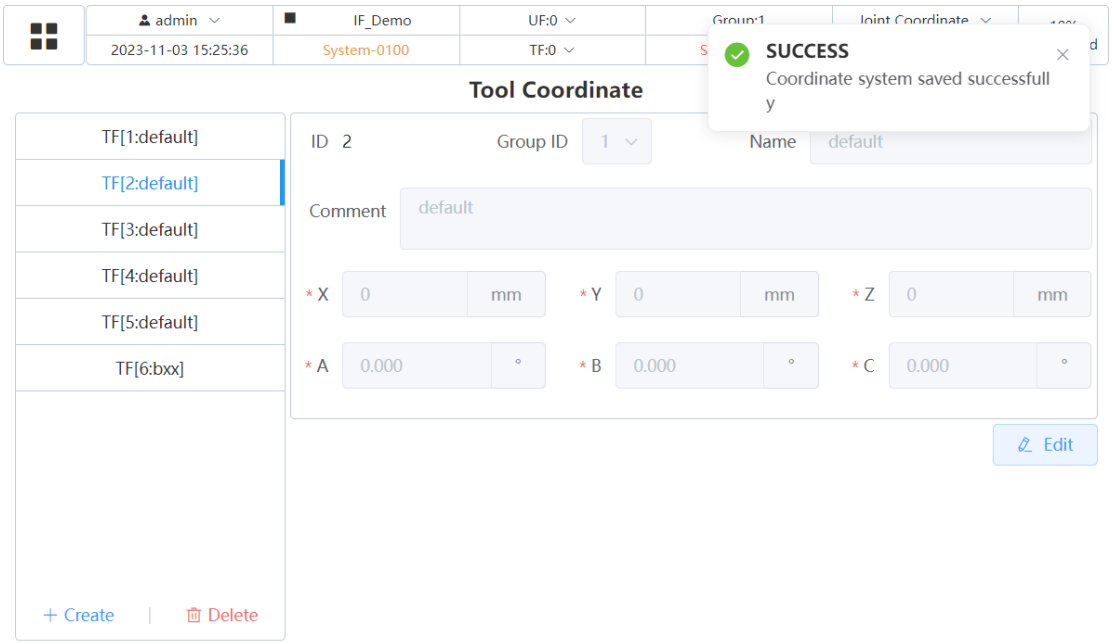


Fig. 2.18 Tool Coordinate System Setting Interface 6

4P setting of tool coordinate system for PUMA robot

4P setting steps: Enter the tool coordinate system setting interface, select the 4P method and teach P1-P4 in Fig. 2.12 according to the pose requirements. Please refer to the 7P method for specific operating steps.

4P setting steps of tool coordinate system for SCARA robot

1. Click "Menu Button" → "Coordinate System" → "Tool Coordinate System" to enter the interface as shown in Fig. 2.19, and then select the 4P method.

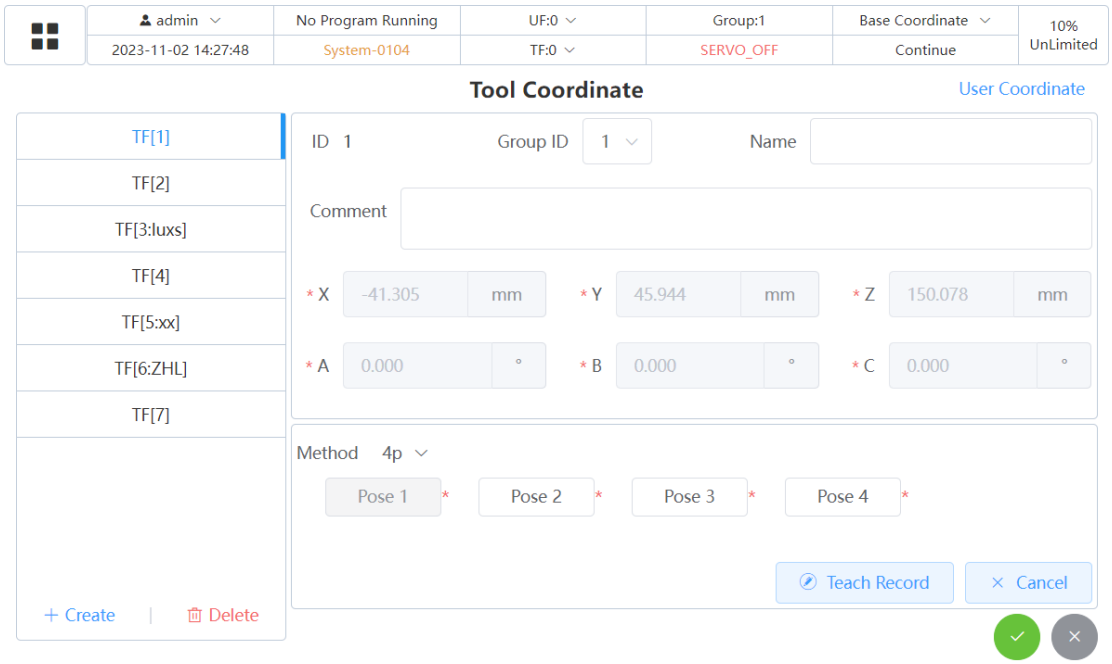


Fig. 2.19 Tool Coordinate System Setting Interface

2. As shown in Fig. 2.20, select the "base coordinate system" as the operating coordinate system, move the tool along four different pose directions to the same reference pose as shown in Fig. 2.21, and record the poses separately.

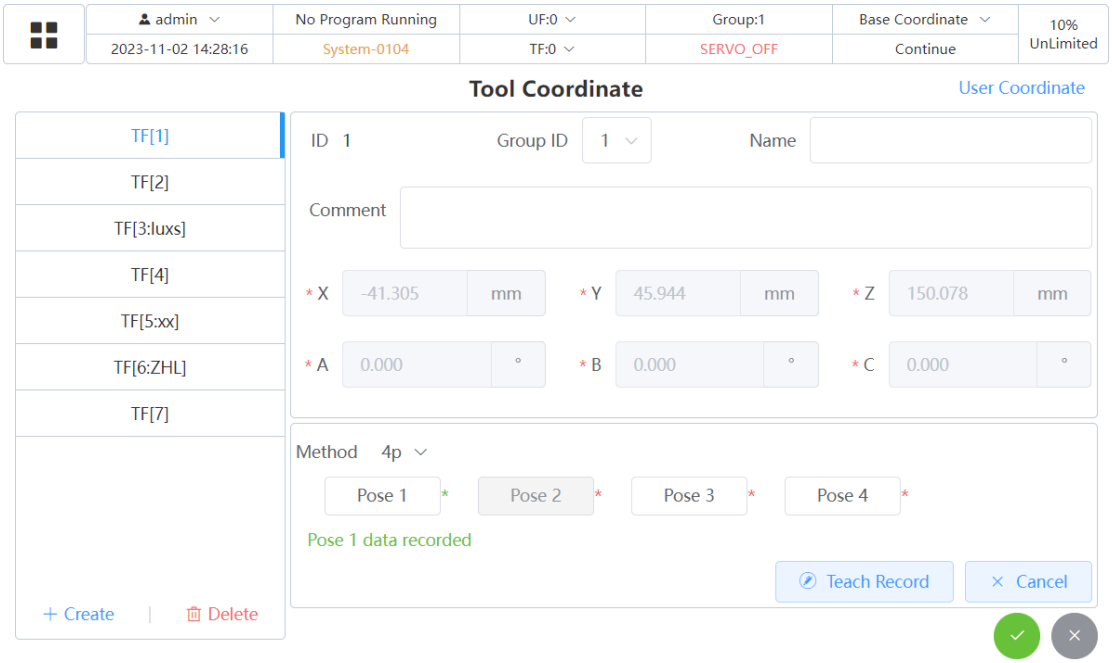


Fig. 2.20 Tool Coordinate System Setting Interface

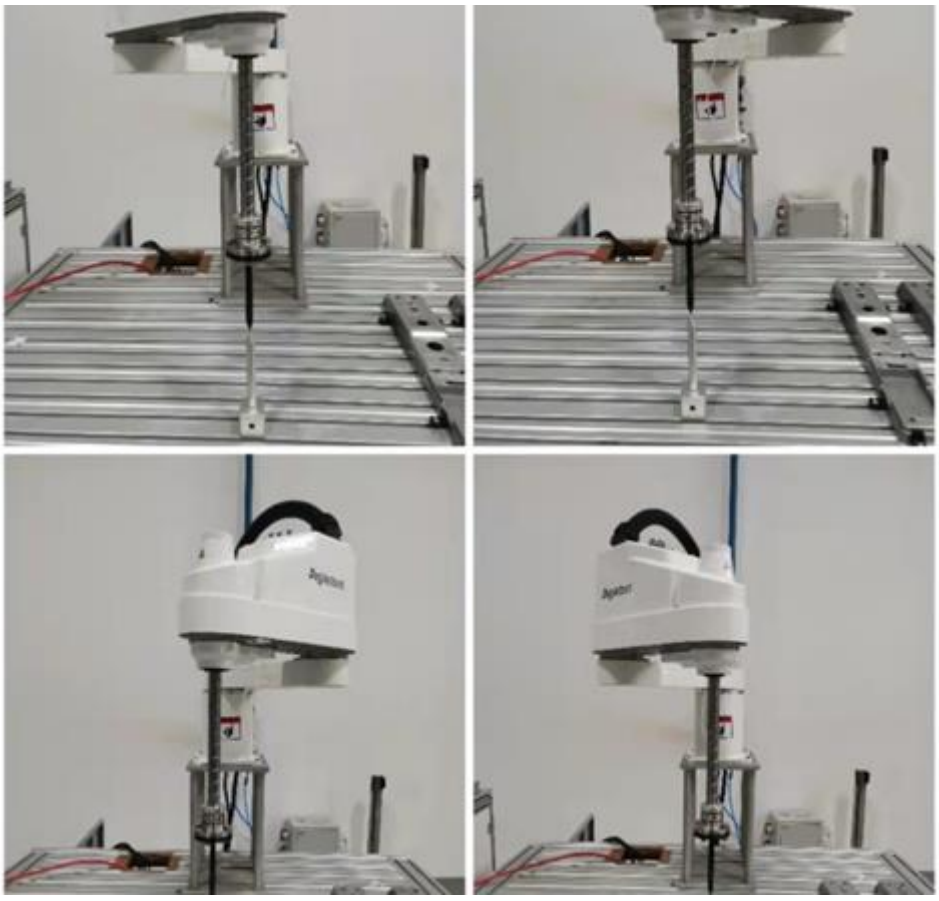


Fig. 2.21 Reference Poses for 4P Method

1. After pose recording, the tool coordinate system is automatically calculated as shown in Fig. 2.22. Click to record the coordinate system settings.

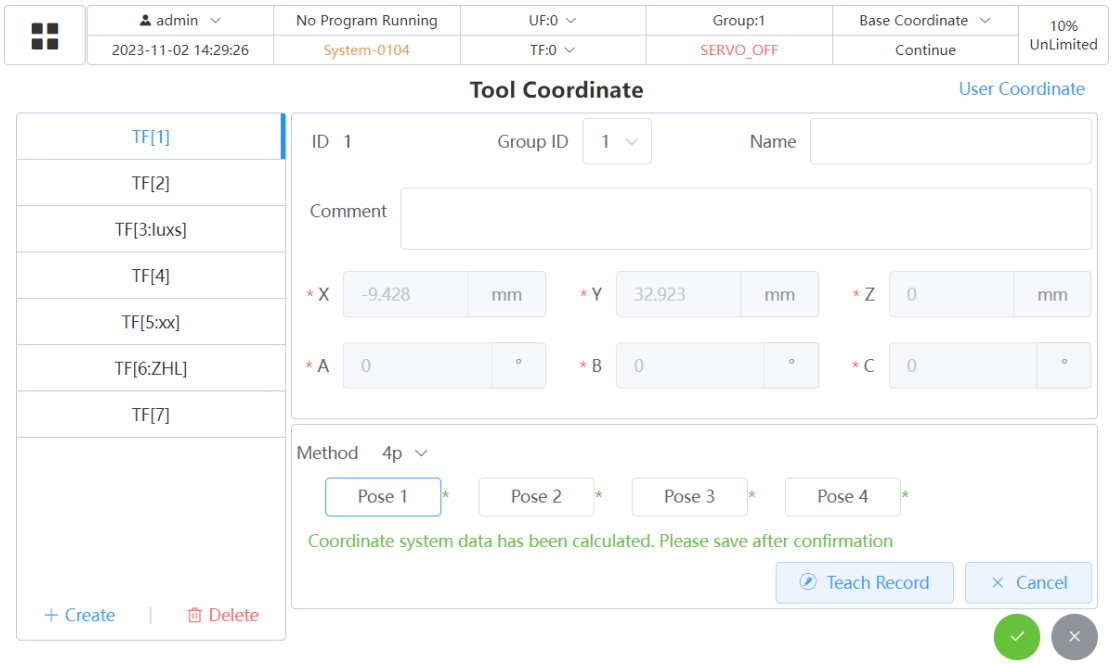


Fig. 2.22 Click “Save” after Teaching

Setting of tool coordinate system by direct write method

As shown in Fig. 2.23, select the direct write method to write the tool coordinate data X, Y, Z, A, B, C.

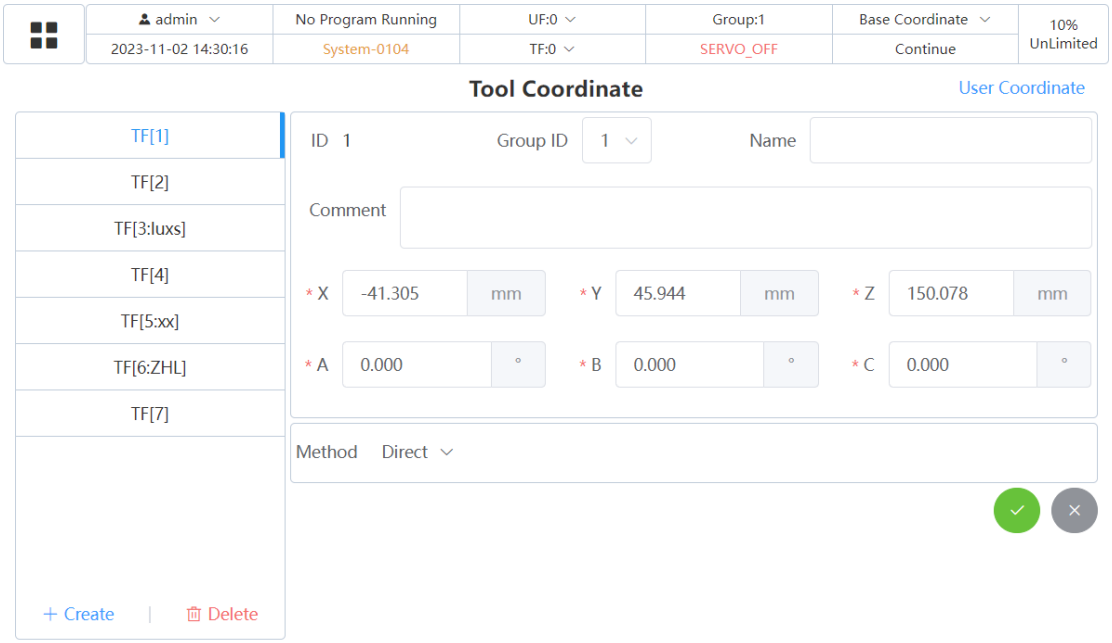


Fig. 2.23 Window of Direct Writing Method

2.2.2 Setting of user coordinate system

The user coordinate system is a Cartesian coordinate system defined based on the user's workspace.

When not defined, the user coordinate system is replaced by the world coordinate system.

The user coordinate system is defined by the position of the origin relative to the base coordinate system (X, Y, Z) and the rotation angles around X, Y, and Z axes (A, B, C).

The user coordinate system is used when setting and executing pose registers and executing position compensation instructions. In addition, the position in the program can also be taught based on user coordinates. For setting of the position register, please refer to Section 5.1.3. For execution of position compensation instructions, refer to 3.3.5 Additional Action Instructions.


Caution

In the case of teaching in joint form, changing the user coordinate system may not affect the position variables and position registers. However, please note that position variables and position registers are affected by the user coordinate system in the case of teaching in Cartesian form.

When the user coordinate system is defined on the coordinate system setting screen, 10 coordinate systems can be defined and switched according to the situation.

Advantages of user coordinate system:

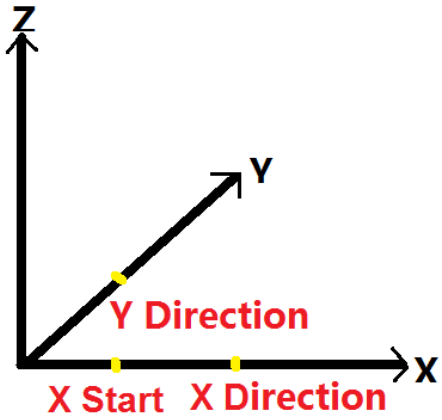
- Manually move the tool coordinate system along the edge of the working surface or workpiece.
- Perform pose teaching with the user's base coordinates as references. If the workpiece must be moved for its fixture has been moved (for example), it is required to simply reset the user coordinate system to the same position on the fixture and not to teach the program points again.
- The default user coordinate system coincides with the base coordinate system.

The data format of the user coordinate system is: [X 0mm, Y 0mm, Z 0mm, A 0°, B 0°, C 0°].

Setting method:

1. 3P method

As shown in Fig. 2.24, the first point "X Start" is the starting point of the X-axis; the second teaching point "X Direction" is a point in the X-axis direction; the third teaching point "Y Direction" is a point in the Y-axis direction. After clicking "Save", the system may automatically calculate XYZABC parameters related to the user coordinate system.



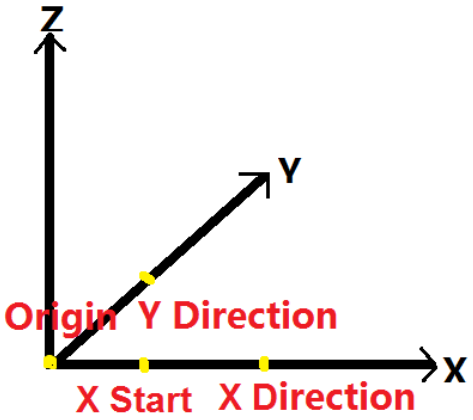


Fig. 2.24 3P Method Fig. 2.25 4P Method

2. 4P method

As shown in Fig. 2.25, the first point "X Start" is the starting point of the X-axis; the second teaching point "X Direction" is a point in the X-axis direction; the third teaching point "Y Direction" is a point in the Y-axis direction; the fourth point "Coord Origin" is the origin position of the user coordinate system (which can be any point in space). After clicking "Save", the system may automatically calculate XYZABC parameters related to the user coordinate system.

3. Direct write method

Steps of 3P/4P method:

1. Click "Menu Button" → "Coordinate System" → "User Coordinate System" to enter the interface as shown in Fig. 2.26, and Choose 3p/4p as shown in the figure below.

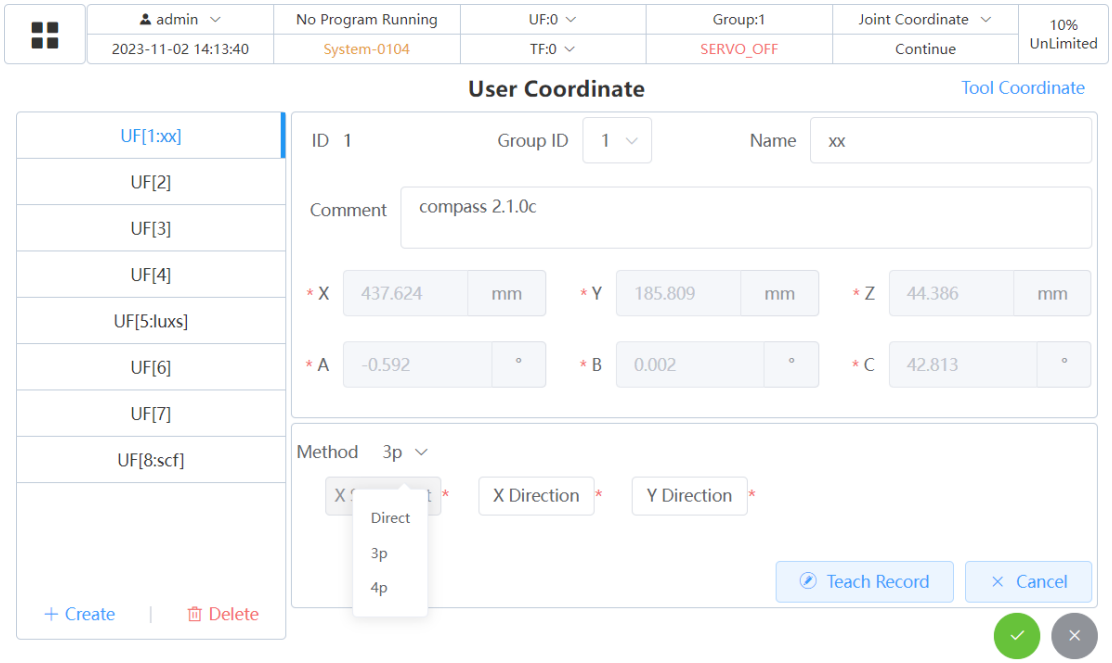


Fig. 2.26 User Coordinate System Setting Window

2. Record the poses according to the description of the 3P/4P method. After all points are recorded, the user coordinate system can be automatically calculated, as shown in Fig. 2.27 “3P Method”/2.28 “4P Method”.

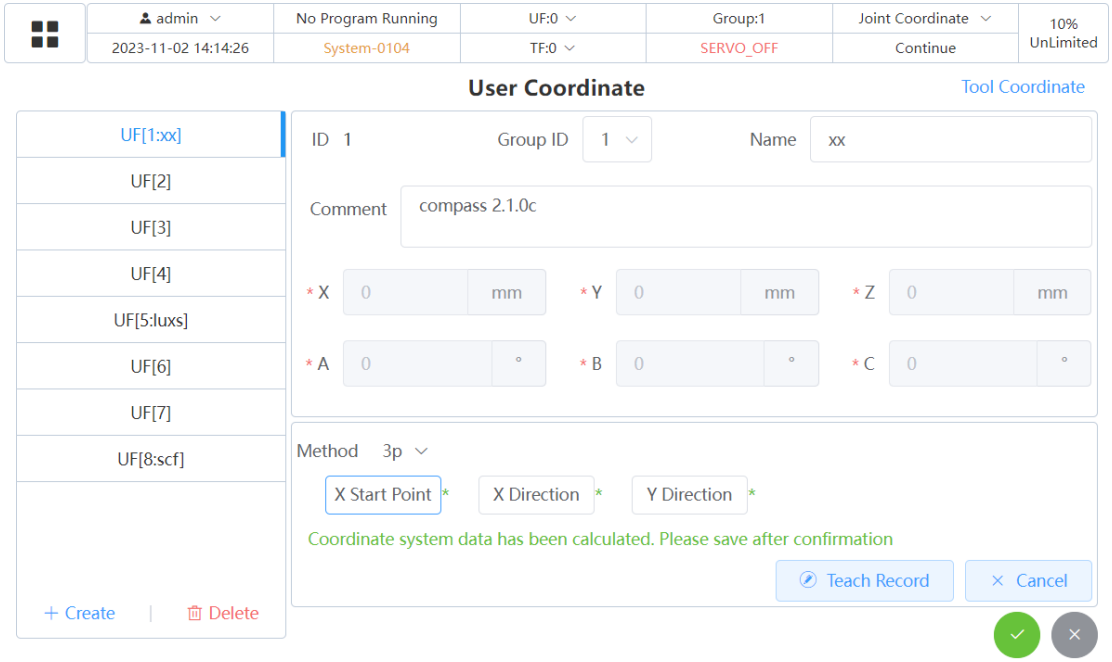


Fig. 2.27 3P Setting Window

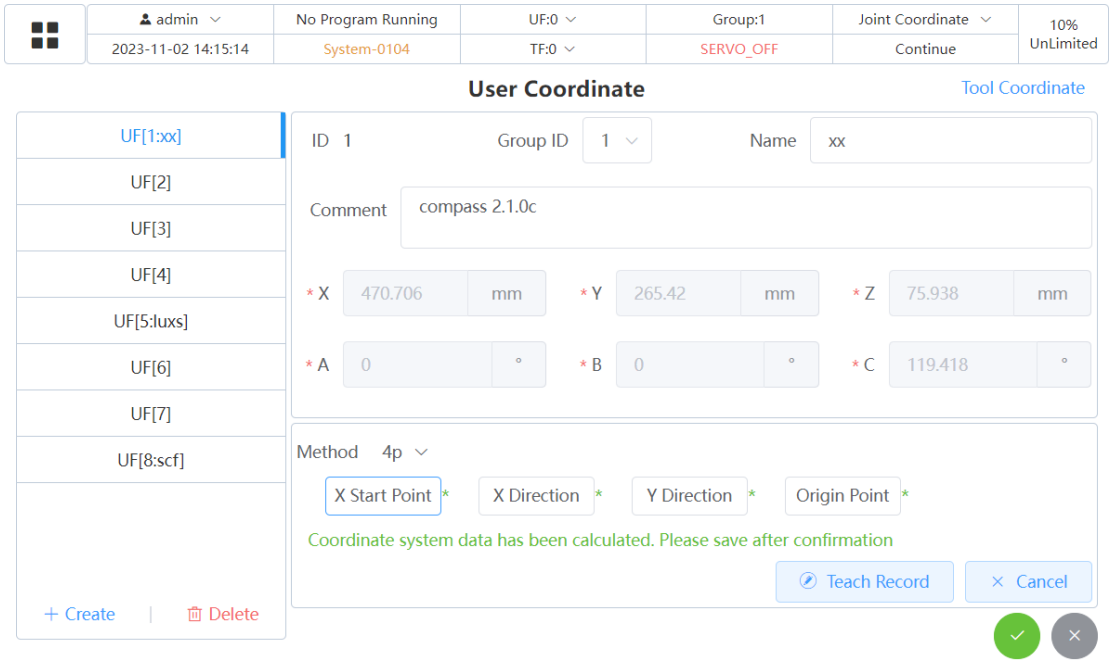


Fig. 2.28 4P Setting Window

Direct write method:

In the user coordinate system interface, choose the direct write method (Direct) to write the date of the user coordinate system.

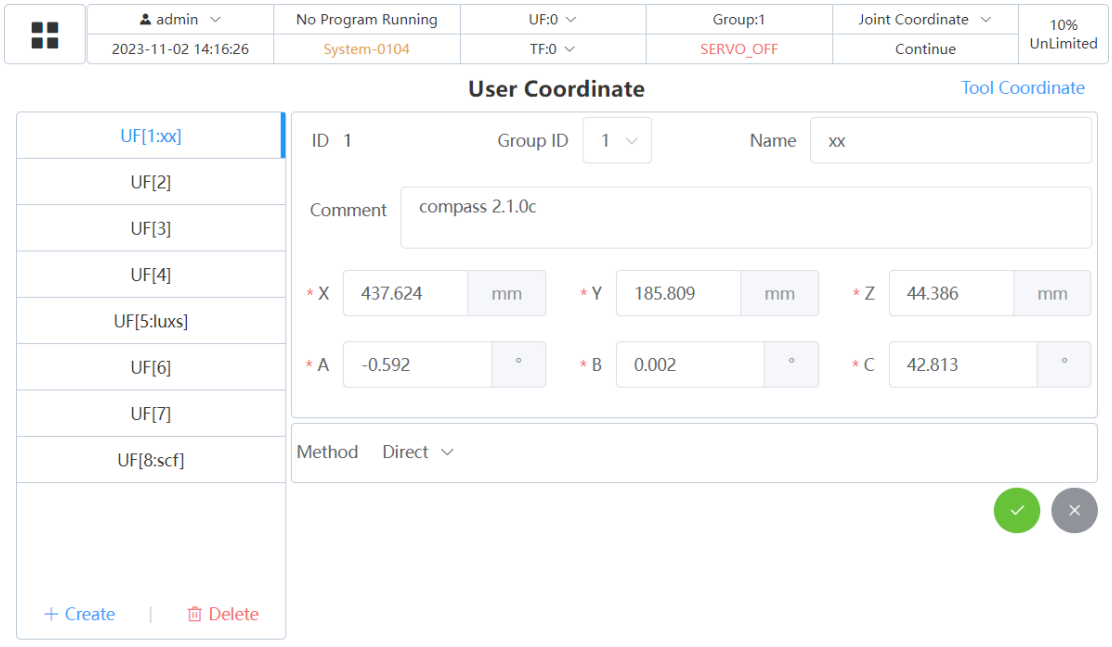


Fig. 2.29 Setting Window of Direct Writing Method

2.2.3 Setting of Remote TCP

Overview to remote TCP:

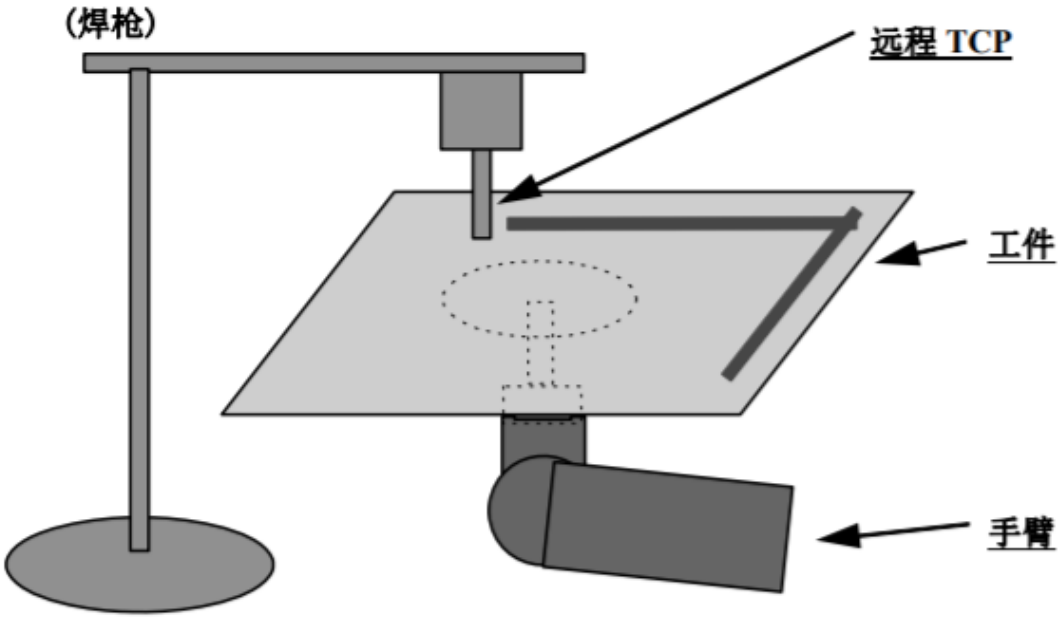
The remote TCP function can be adopted to move the workpiece fixed on the robot relative to the TCP fixed on the ground (i.e. remote TCP), so that the workpiece can perform linear, arc and other actions relative to the tool fixed on the ground so as to complete machining.

(Welding gun)

Remote TCP

Workpiece

Arm



远程 TCP 功能利用例(焊封应用)

Fig. 2.30 Design Diagram of Remote TCP

To use the remote TCP function, it is necessary to teach the robot for protruding positions relative to the tool fixed on the ground.

The setting method of the RTCP coordinate system is consistent with that of the user coordinate system, of which the direct write method/3P method/4P method can be adopted and the RTCP coordinate system can be set directly in the configuration page of the user coordinate system.

Z (tool direction)

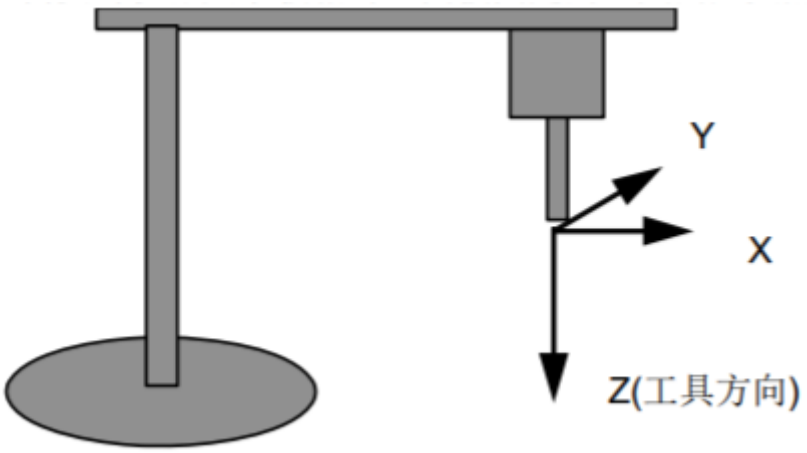


Fig. 2.31 Schematic Diagram of Remote TCP Coordinate System

When performing jogging teach based on the RTCP coordinate system, namely parallel movement/rotation around the remote TCP, it is necessary to select "RTCP/Tool" or "RTCP/User" from the coordinate system drop-down menu in the page status bar and choose corresponding user and tool coordinate systems.

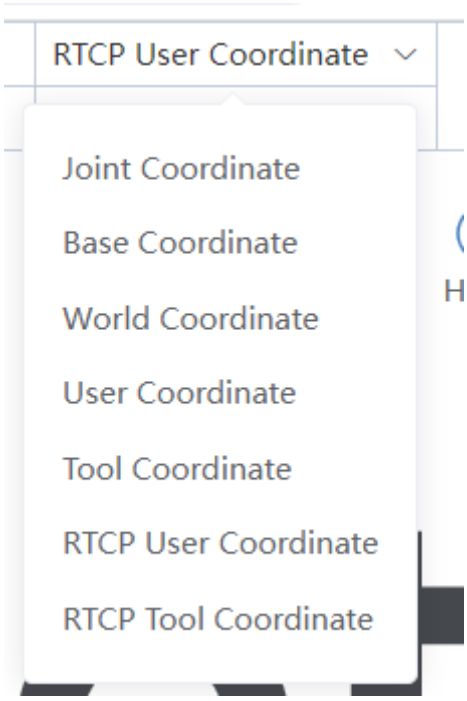


Fig. 2.32 Reference Coordinate System Menu

The characteristics and differences between "RTCP/Tool" and "RTCP/User":

Coordinate system	Center point	Equal movement/rotation direction

RTCP/tool	RTCP	Selected tool coordinate system
Tool coordinate system	TCP	Selected tool coordinate system
RTCP/user	RTCP	Selected user coordinate system

Fig. 2.33 References of Remote TCP Movement Directions

When executing remote TCP actions, it is necessary to add an additional instruction RTCP of remote tool action in the instructions. When the RTCP additional instruction is used as the motion instruction, the user coordinate system adopted for pose recording may become the RTCP coordinate system. In case of no RTCP additional instruction, it still maintains the user coordinate system.

RTCP cannot be used during joint movement.

Example:

Relative to remote TCP, move the workpiece to P[1] at a relative velocity of 2000mm/s:

MoveL P[1] 2000mm/s Fine RTCP

Relative to remote TCP, move the workpiece from P[1] to P[2] at a relative velocity of 2000mm/s:

MoveC P[1] P[2] 2000mm/s Fine RTCP

2.2.4 Coordinate transformation function

Overview to coordinate system transformation:

The coordinate transformation function is as follows: select the instruction in a certain range in the program, convert the tool or user coordinate system used in the pose data according to the motion statement of the recorded pose data and then use the transformed program statement as a new program or segment. Before transformation, different transformation rules can be selected to change or maintain coordinate values in the pose data, so that the robot can move to a new actual position based on the new coordinate system and the actual position of the robot can also be kept consistent with that before transformation. Namely, the angles of each joint remain unchanged.

Application scenario:

Arm A previously used is damaged and should be replaced with Arm B. However, Arm B is 1cm shorter than Arm A. In this case, the tool coordinate system of Arm B can be established first. Then, the tool coordinate transformation rules can be configured and executed. Finally, the robot can also reach the position reachable by Arm A when executing the transformed program.

When it is necessary to copy the working surface and its trajectory, a user coordinate system can be established based on the new working position. Then, the user coordinate transformation rules can be configured and executed. Finally, when the robot executes the transformed program, its relative trajectory in the new user coordinate system can be kept consistent with that in the original working surface.

Setting of coordinate system transformation:

1. Click "Menu Button" → "Application" → "Coordinate Transformation" to enter the coordinate transformation interface, as shown in Fig. 2.34.

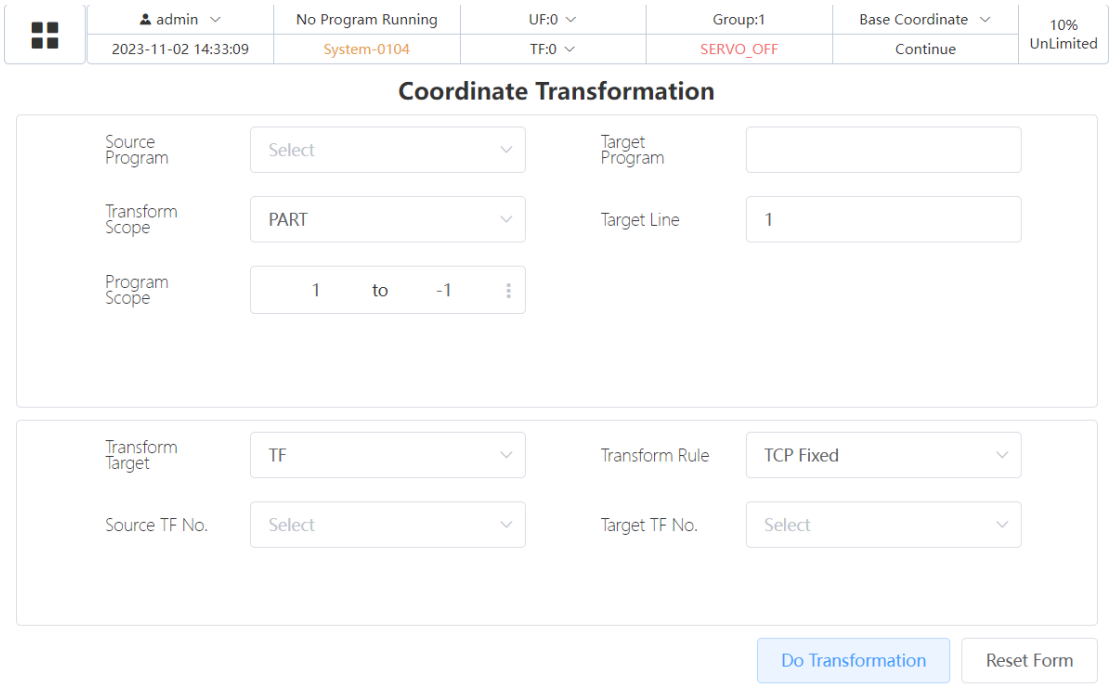


Fig. 2.34 Coordinate Transformation Interface

Description of coordinate transformation parameters:

- Source Program: A program requiring coordinate transformation
- Target Program: Name the new program after transformation here or use the original program name.
- Transform Scope: Choose All to transform the whole program or Local to

transform the specified program scope.

- Program Scope: Used in conjunction with transform scope.
- Target Line: Insert the line of the target program.
- Transform Target: There are two types: user coordinate system and tool coordinate system.
- Transform Rule: If the transform target is the user coordinate system, there are two transform rules: Pose Data Fixed and Pose Data Changed.
- If the transform target is a tool coordinate system, the transform rules are TCP Fixed and Robot Fixed.
- For specific transform rules, see the detailed description of transform rules in the following text.
- Number of tool/user coordinate system before transform: The number of the tool/user coordinate system used for the pose in the original program.
- Number of tool/user coordinate system after transform: The number of the tool/user coordinate system to be used in the target program.

2. After setting of the above parameters, click "Do Transformation", and the following pop-up window will appear.

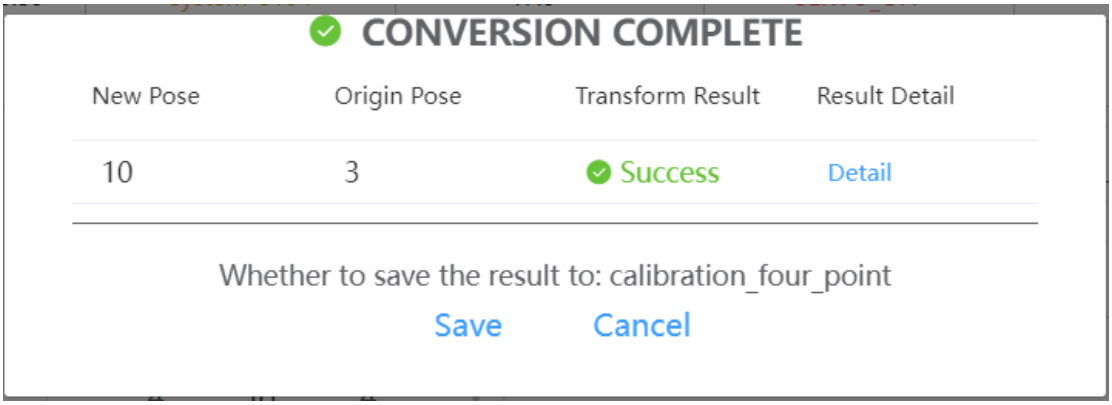


Fig. 2.35 Transformation Completion Window

3. Click “Save” to complete the transformation.

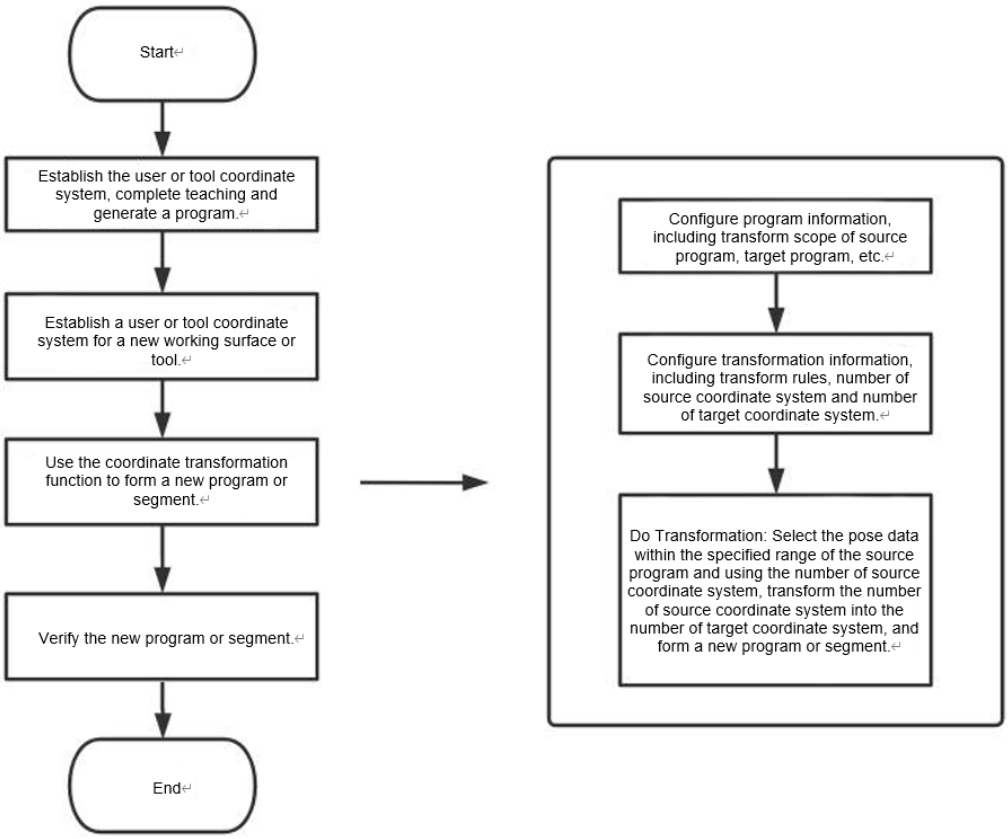


Fig. 2.36 Coordinate Transformation Flowchart

Caution

Position and pose values in pose data:

Only perform offset calculation for P[] in the program, while the pose register PR[] remains unchanged and does not involve in offset calculation.

After offset calculation, the pose P[] expressed based on the Cartesian coordinate system is still a Cartesian coordinate value and the pose P[] expressed based on the joint coordinate system is still a joint coordinate value.

When the P[] after offset calculation falls outside the motion space, it is saved as an untaught value if the P[] is expressed based on the joint coordinate system; the offset value is directly saved if the P[] is expressed in a Cartesian coordinate system.

Transformation rules in tool coordinate transformation:

- TCP Fixed: Before and after transformation, the "Tool coordinate system number after transformation" directly replaces the "Tool coordinate system number before transformation" and the pose coordinate value remains unchanged in the pose data.

Typical application scenario: It is used to replace damaged tools. This rule can ensure that the new tool can still reach the working position of the original tool when the transformed program is executed.

- Robot Fixed: Before and after transformation, the "Tool coordinate system number after transformation" directly replaces the "Tool coordinate system number before transformation" in the pose data. However, the pose coordinate values are recalculated and generated and their calculation rule is that the joint angle of the robot remains unchanged relative to the original pose. Typical application scenario: It mainly lets the robot to avoid certain path positions, but does not care about the position of TCP.

Transformation rules in user coordinate transformation:

- Pose Data Fixed: Before and after transformation, the "User coordinate system number after transformation" directly replaces the "User coordinate system number before transformation" and the pose coordinate value remains unchanged in the pose data. Typical application scenario: When the original working surface is displaced or a working surface should be copied, a new user coordinate system can be established based on the changed working surface (a user coordinate system has already been established based on the original working surface). Then, the rule is used to execute the user coordinate transformation. Finally, when the robot executes the transformed program, it can ensure that the path of the robot remains unchanged relative to the new working surface.

- Pose Data Changed: Before and after transformation, the "User coordinate system number after transformation" directly replaces the "User coordinate system number before transformation" in the pose data. However, the pose coordinate values are recalculated and generated and their calculation rule is that the joint angle of the robot remains unchanged relative to the original pose. Typical application scenario: Actual position of the target workpiece has not changed, but the benchmark for the workpiece has changed.

2.3 Soft limit

The software limits and protects the reach of each axis joint of the robot, so that its software protection range is smaller than the hard limit of the model at default so as to avoid frequent hard collisions during normal operation.

The user is allowed to set the joint range of the soft limit according to the situation (but it is always smaller than the mechanical hard limit range of the model).

Warning

1. Never rely solely on the movable range of the joint to control the motion reach of the robot. The limit switch and hard limit should be used simultaneously. Otherwise, it may cause personal injury or equipment damage.

2.Please adjust the hard limit to meet software adjustment. Otherwise, it may cause personal injury or equipment damage.


Caution

The change in the movable range of the joint has an impact on the motion reach of the robot. To avoid malfunctions, it is necessary to reconsider the impact of changing the movable range of each axis. If the change of the movable range is not considered sufficiently, it may lead to unexpected results, e.g. alarm at the previously taught position.

Upper limit:

It indicates the upper limit value of the joint's movable range. It is the movable range in the positive direction.

Lower limit:

It indicates the lower limit value of the joint's movable range. It is the movable range in the negative direction.

Function description:

The joint range of the soft limit can be set through the TP application end. The setting method is to modify it in the input box with an accuracy of 0.001°. Null values and characters are prohibited. It is not allowed to fill in a soft limit range value exceeding the factory default range (the default range is displayed in the soft limit setting interface).

2.3.1 Soft limit interface

Steps for setting soft limit of the joint:

Successively click "Menu Button" → "System" → "Basic Setting" → "Soft Limit Setting" to enter the screen as shown in Fig. 2.37.

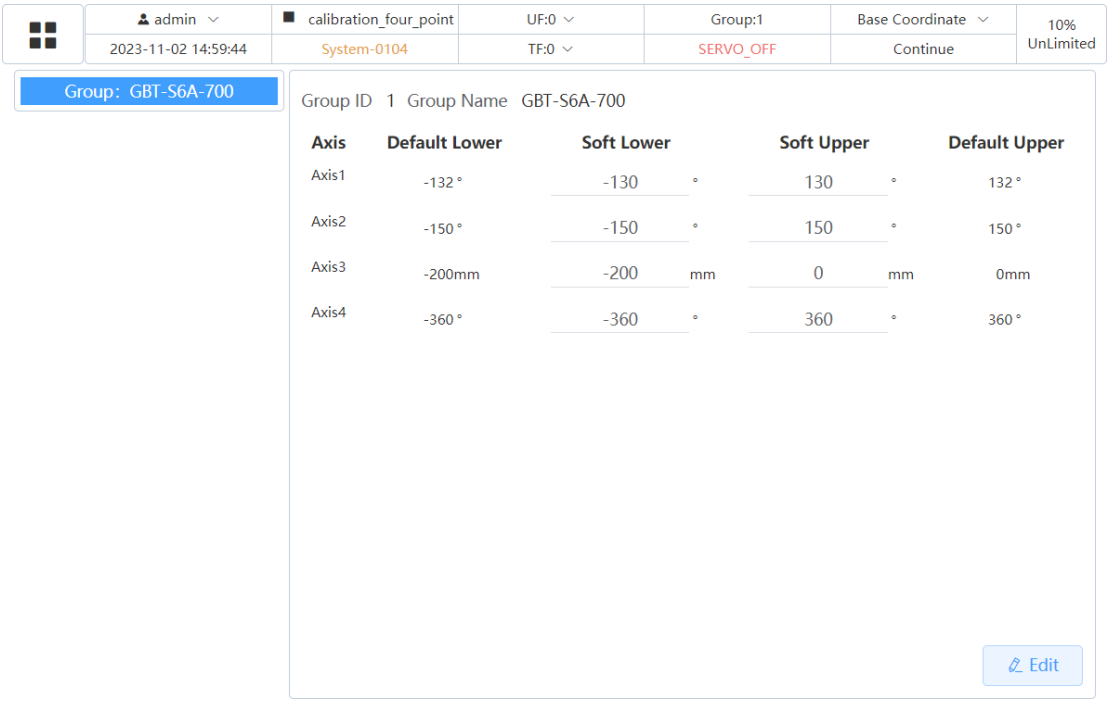


Fig. 2.37 Soft limit Setting interface

Through this interface, the user can set the soft limit of the robot. However, the soft limit set by the user must be less than or equal to the default soft limit set at the factory.

- Axis: It indicates the information of all axes under the motion group.
- Soft Lower: It is the lower axis limit. It is not necessarily a negative number, but must be less than or equal to the positive limit.
- Soft Upper: It is the upper axis limit. It is not necessarily a positive number, but must be greater or equal to the negative limit.

Caution

Click "Edit" to modify upper and lower limits of the soft limit. In the input box of positive and negative limits, the system may automatically change to the upper limit of the positive limit specified at the factory if the user fills in a value greater than the specified positive limit at the factory; the system may automatically change to the lower limit of the negative limit specified at the factory if the user fills in a value less than the specified negative limit. After change, click "Save" to make it effective immediately.

2.4 Payload setting

Summary of payload setting:

The payload setting is to set relevant information of the payload (weight, barycenter, etc.) mounted on the robot.

The following effects can be achieved by setting payload information appropriately.

- Improve motion performances (lower vibration, better cycle time, higher accuracy, etc.).
- Effectively utilize relevant dynamic functions. (Improve the collision detection function, gravity compensation function and other properties.)

If a greater error is found in payload information, it may lead to great vibration or incorrect detection of collisions. In order to more effectively use the robot, the user is recommended to appropriately set payload information of the devices arranged on robots, workpieces or arms.

The payload information can be set on the "Payload Setting Screen". 10 payloads can be set on this screen. Multiple payloads can be set in advance. Then, the payloads can be changed by simply switching their numbers. In addition, the payload number can be switched in the program through program instructions (see Section 3.8.5).

The robot can achieve optimal control accuracy and stability under corresponding payloads by selecting or adjusting internal control parameters for different payloads.

The payload is set in the following steps:

1. Successively click "Menu Button" → "System Setting" → "Basic Setting" → "Payload Setting" to enter the screen as shown in Fig. 2.38. The payload activated at default is set to [Payload: 0] and cannot be edited.

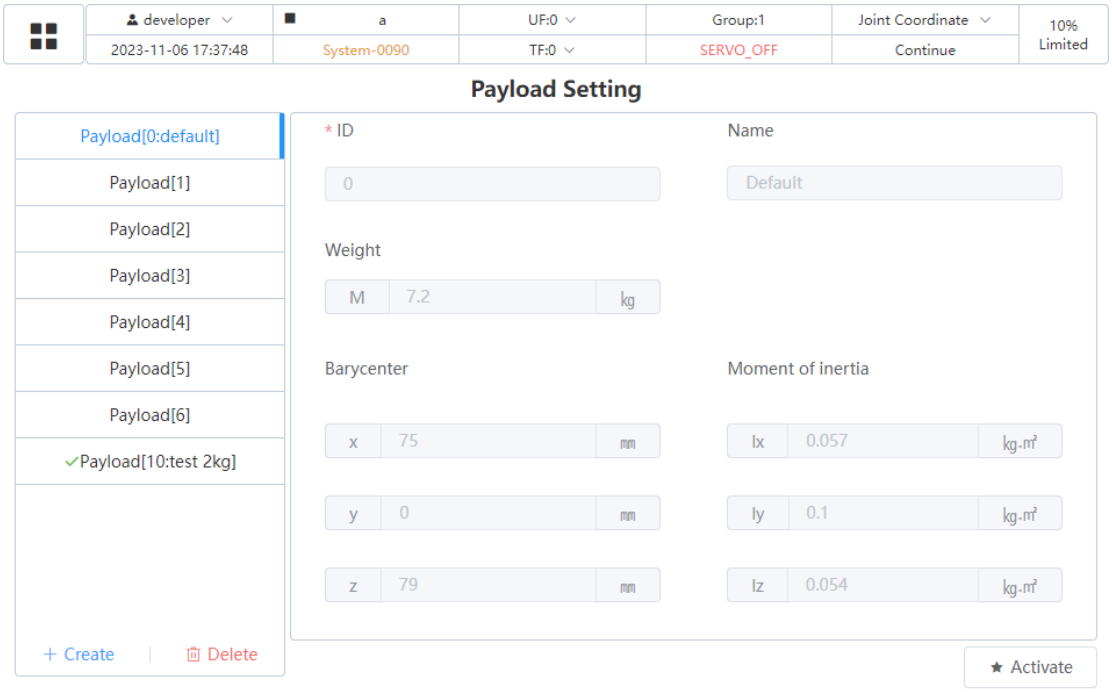


Fig. 2.38 Payload Parameter Interface

2. Click “New” and set a new payload. Click "Edit" to manually input payload data, as shown in Fig. 2.39.

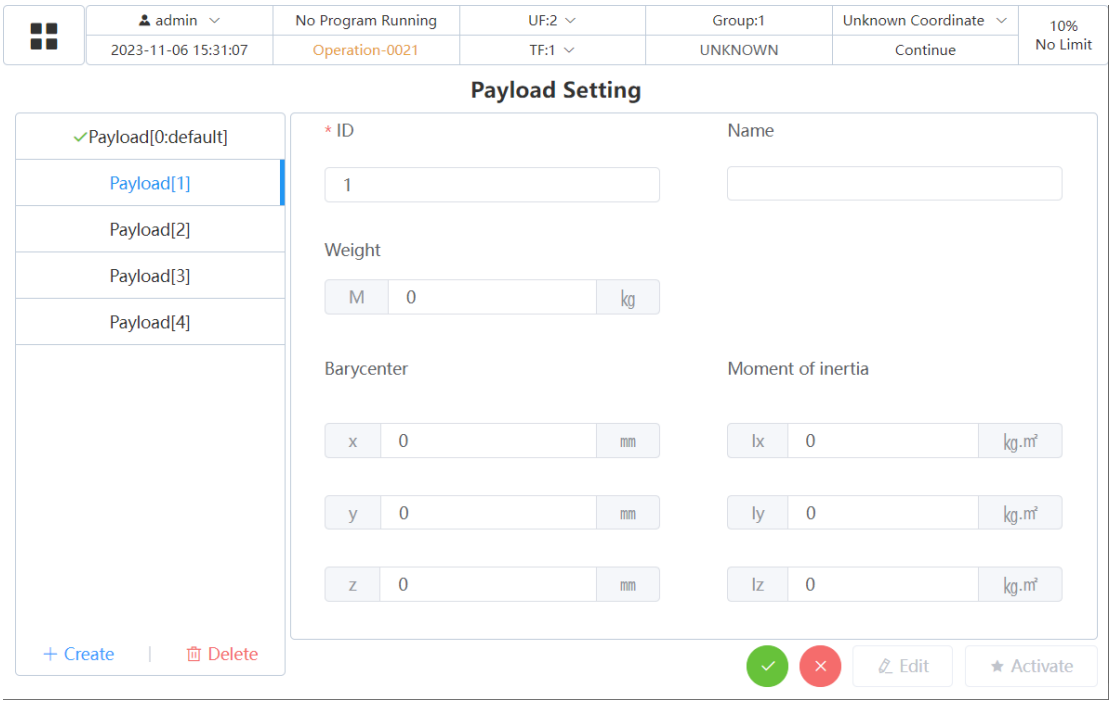


Fig. 2.39 Payload Parameter Interface

Among others, M (kg) is the mass of the payload, X (mm), Y (mm) and Z (mm) are the barycenter positions of the payload relative to the flange center as shown in Fig. 2.39, and Ix, Iy and Iz are the rotational inertia of the payload relative to X, Y and Z coordinates. When the payload of the robot is taken as a mass point, the moment of inertia Ix, Iy and Iz are written as 0.

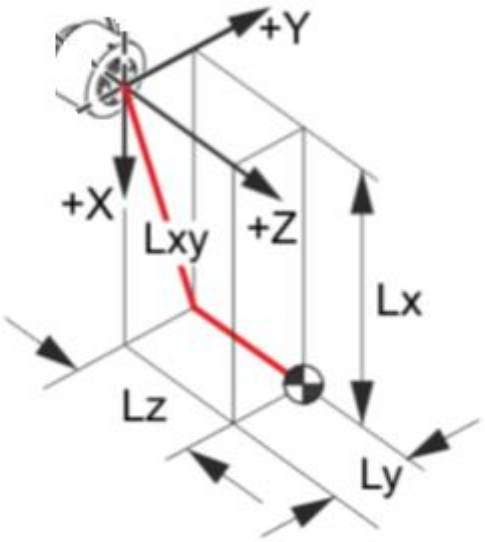


Fig. 2.40 Payload Position Reference

3. The setting of activated payload is shown in Fig. 2.40, with a "√" mark in front of the activated load.

2.5 Zero calibration

Zero calibration is an operation associating the angle of each robot joint with the pulse count.

The zero calibration operation is to obtain the pulse count corresponding to the zero-point position.

The “zero calibration” is completed before ex-factory. It is unnecessary to perform zero calibration in daily operations. However, zero calibration should be performed in the following situations.

Please contact us for performing high-precision calibration in the following situations: the motor, pulse encoder or reducer is replaced, or the battery used for pulse count backup is depleted.

Warning

The data of the robot and the pulse encoder, including zero calibration data, are saved through their respective backup batteries. The battery depletion may cause data loss. The batteries in the controller and mechanism should be replaced regularly. When the battery voltage drops, the system may give an alarm to notify the user - please replace the battery timely.

Zero calibration method

- General calibration method
- Direct writing method of zero encoding data

2.5.1 General calibration method

Select one or several axes and record their current readings as new zero data in the parameter file of the robot's Flash. The recording objects include main axis and additional axes of the robot (if any). It is possible to calibrate a single axis. (For example, if a user moves a robot to coincide the zero scale of a certain axis and then uses this function to achieve zero calibration of the robot.)

It is required to perform zero calibration when the loss of zero calibration data for a specific axis is caused by the voltage drop of the battery for the rear pulse counter or the replacement of the pulse encoder. Select the general calibration method and check multiple axes or a single axis for calibration. Check “complete” and click the “calibration” button to complete the calibration.

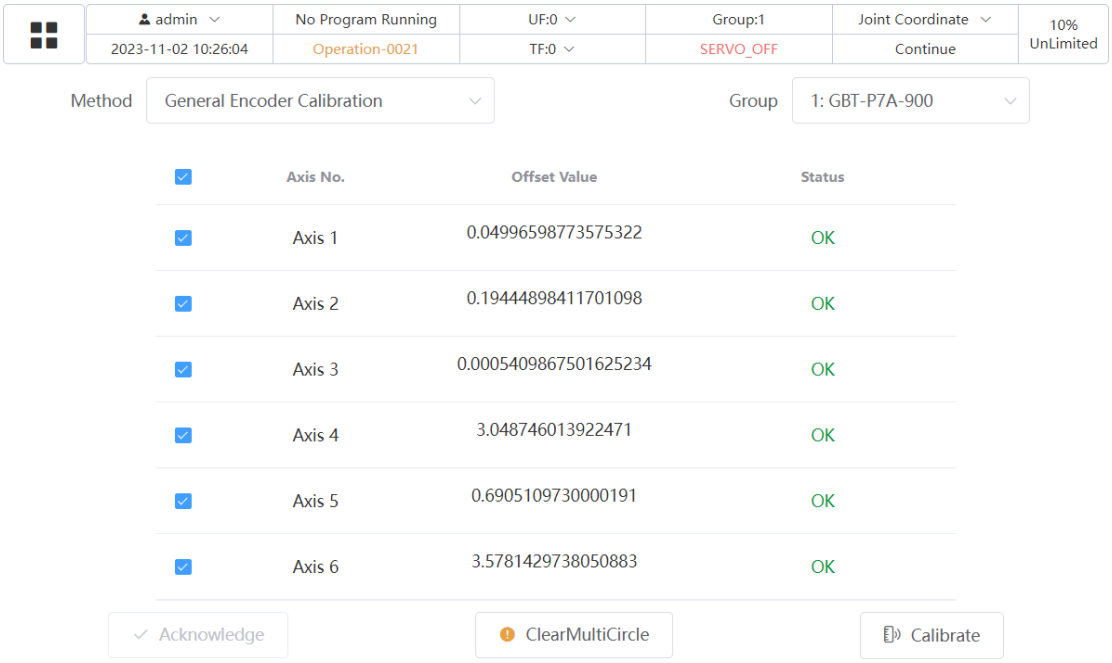


Fig. 2.41 Interface of General Calibration Method

The steps for general calibration method are as follows:

1. For a PUMA robot, align the zero-point reference marks of all axes. For a SCARA robot, align the zero-point reference marks of Axes 1, 2 and 4, but raise Axis 3 to the highest position.

2. Successively click "Menu Button" → "System" → "Basic Setting" → "Zero-point Setting" to enter the screen as shown in Fig. 2.42.

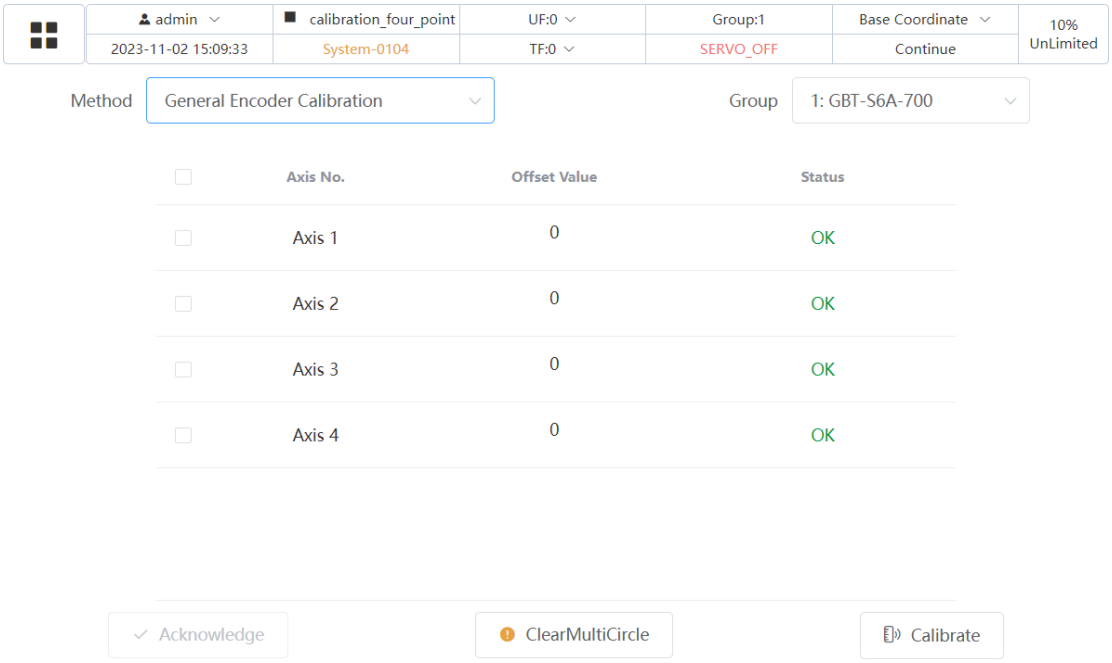
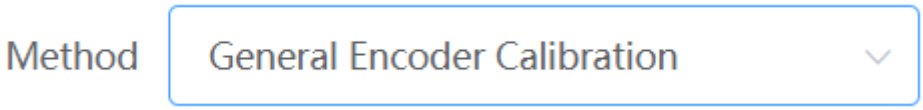


Fig. 2.42 Zero-point Status Screen

3. Click the calibration method in the upper left corner of the zero-calibration screen and then click "General Calibration Method".



4. Click the box in front of the axis number to select all axes to be calibrated as shown in 2.43; alternatively, select an axis to be calibrated separately as shown in Fig. 2.44.

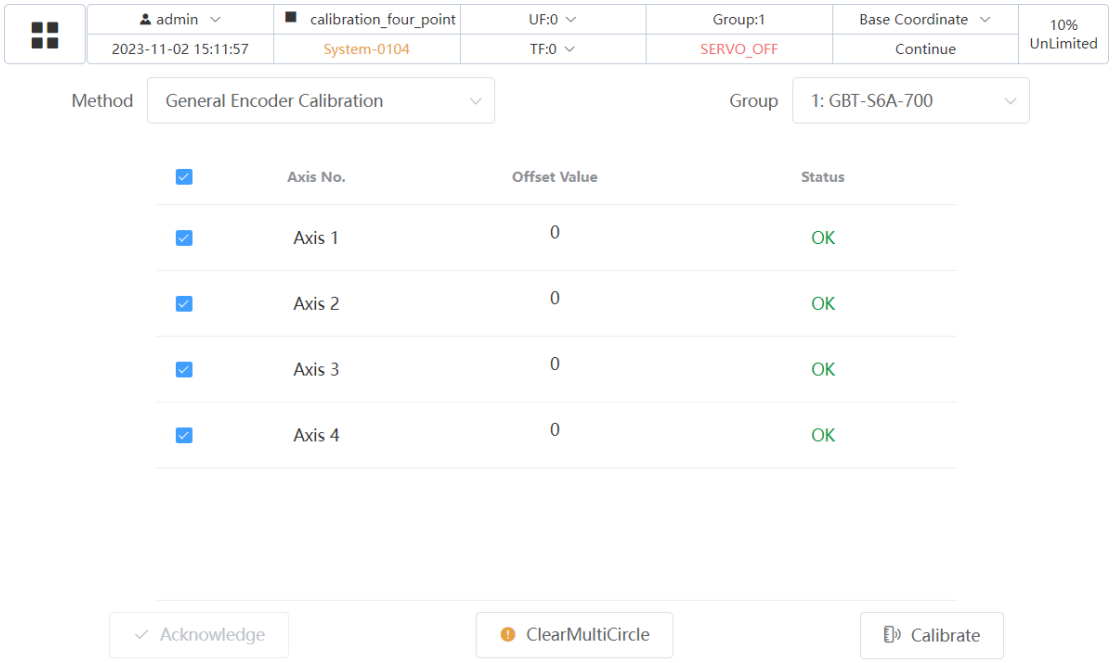


Fig. 2.43 Selection of Multi-axis Screen

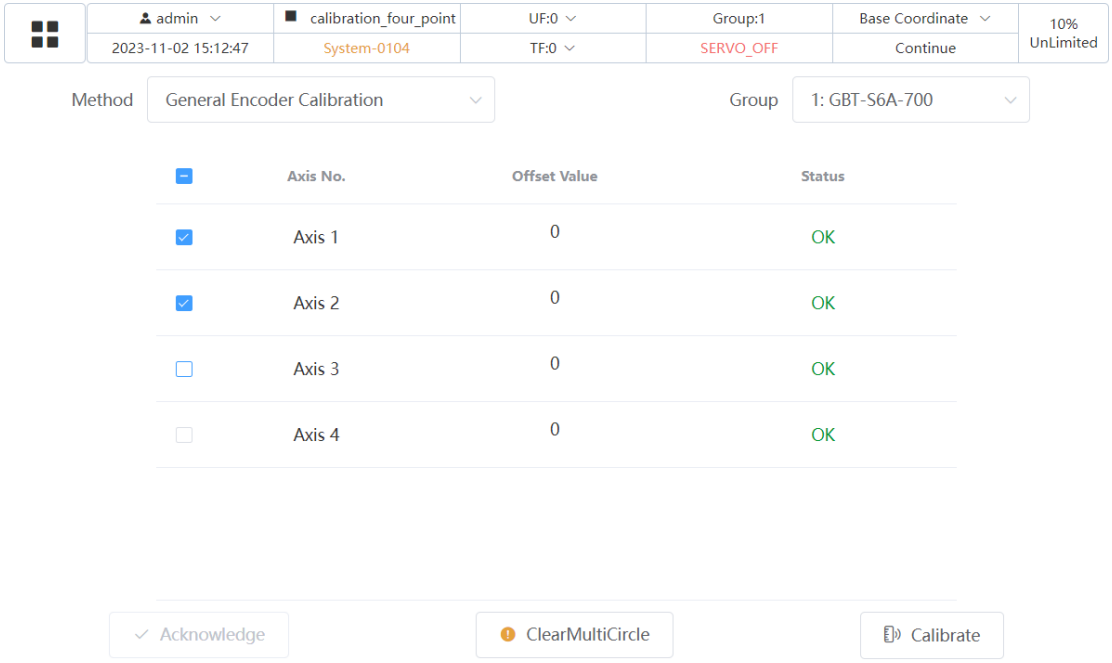


Fig. 2.44 Selection Single-axis Screen

5. After selecting the axis to be calibrated, click the "Calibration Button". If the “Calibration success" message pops up, the zero-point status of the calibrated axis may change to "Unsave". shown in Fig. 2.45.

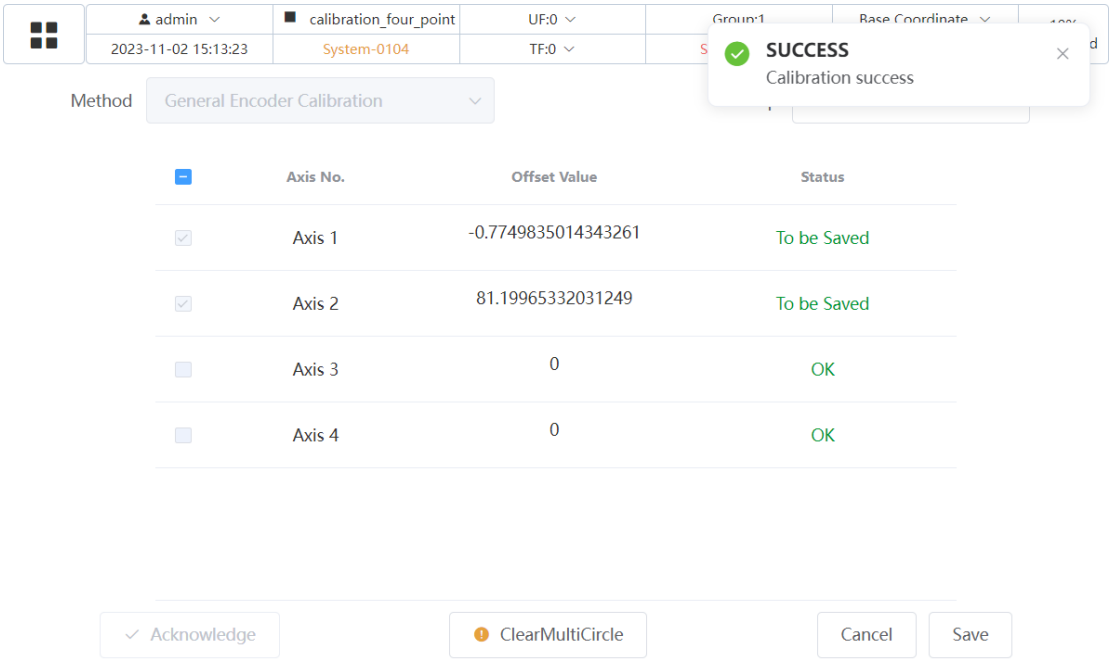


Fig. 2.45 Status Changes to "Unsave"

6. Click “Save” to complete the calibration.

2.5.2 Direct input encoder calibration

As for the direct input encoder calibration, the zero calibration data can be directly entered into the system variables. This operation is used in the situations where zero calibration data is lost while pulse data is still maintained.

The steps for direct input encoder calibration are as follows:

1. Choose the "Direct Input Encoder Calibration" as the calibration method, click on the check box in front of the axis number, and select the axis to be calibrated.

2. Input zero-point data within the offset (note: the zero-point data corresponding to the offset is labeled on the base of the PUMA robot; not for the SCARA robot). Then click on "Calibrate" and finally click "Save".

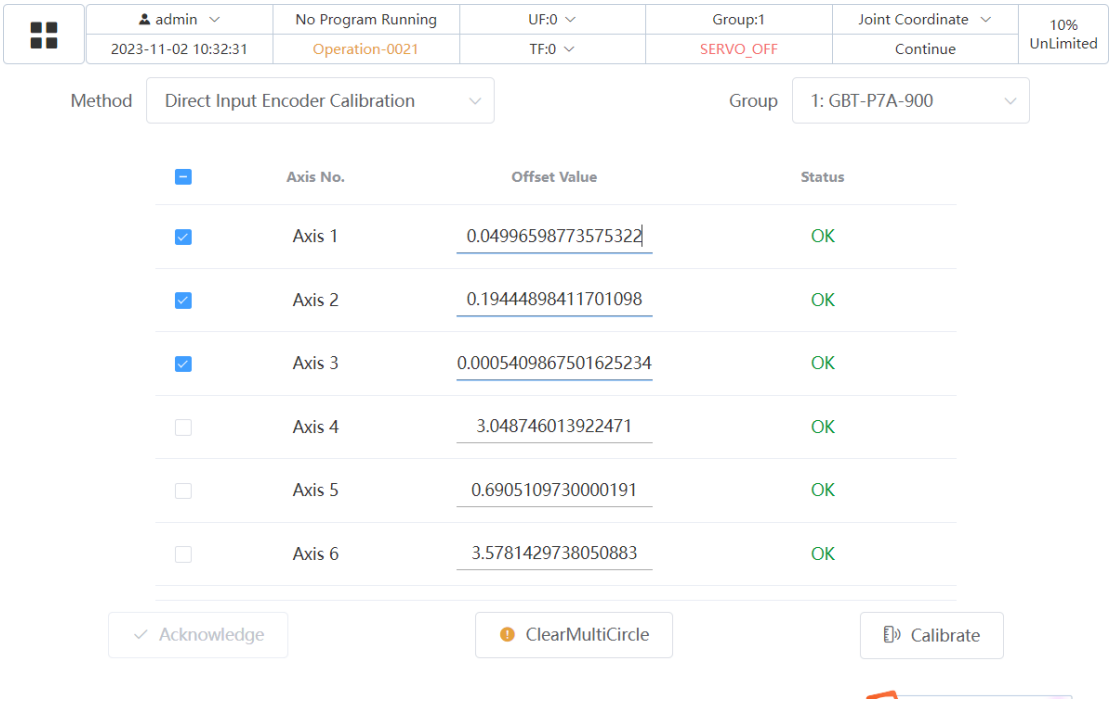


Fig. 2.46 Window of Direct Writing Method

2.5.3 “Temporary Shield Error” function key

When a zero-point error is found on an axis, press the "Temporary Shield Error" button (but do not select a specific axis) to shield the axis reporting an error. So, the alarm axis of the robot can jump out of the alarm state and move on again (limited to joint teaching motion).

The “Temporary Shield Error” function key is only effective for axes with a "zero-point loss flag", while other axes continuously use the original "zero-point data".

After performing a new zero calibration on the problematic axis and saving the calibration data, the "zero calibration shield" flag on that axis disappeared. When the "zero calibration shield" flags on all axes disappear, the robot can resume its normal motion mode.

2.5.4 “Reset Encoder/Clear Encoder Battery Error” function key

It is necessary to clear the multi-circle of the encoder after the encoder is powered off or the robot body is reassembled. At this time, an "encoder battery error" alarm may appear generally. Note that the general calibration method or direct input method can

be only used in the case of zero-point loss alarm.

Please note that the "Reset Encoder" function key (as shown in Fig. 2.47) can only be pressed when all axes of the robot are in mechanical zero position (as this function key is facing all axes of the robot). Otherwise, subsequent calibration may not achieve optimal accuracy.

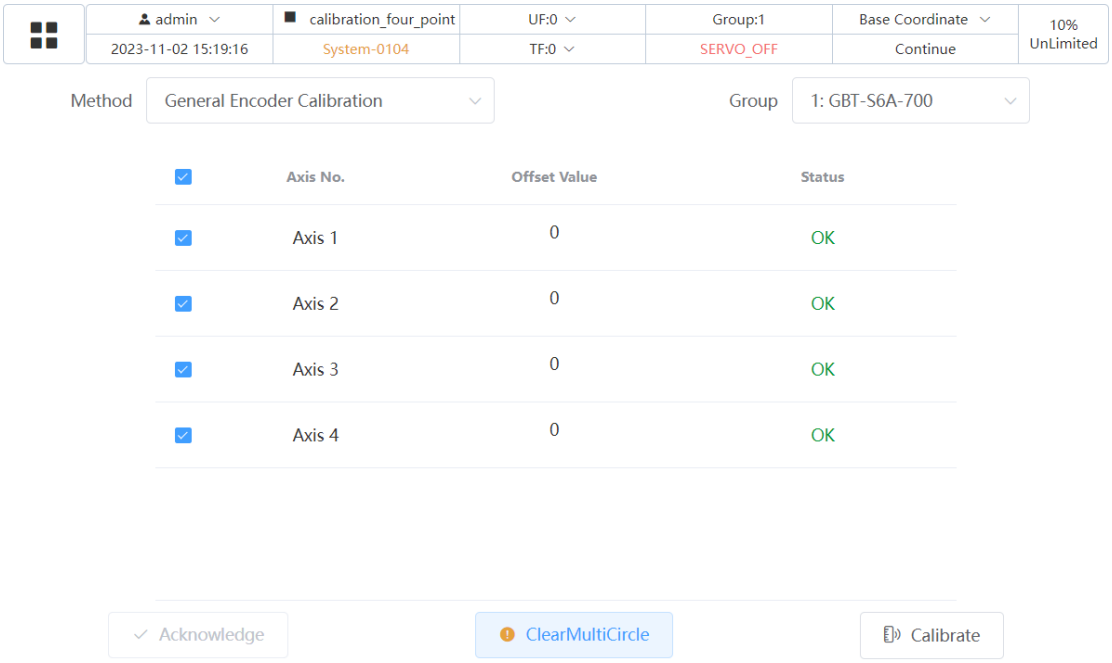


Fig. 2.47 Zero Status Window

After pressing this button, a warning dialog box may pop up, as shown in Fig. 2.48.

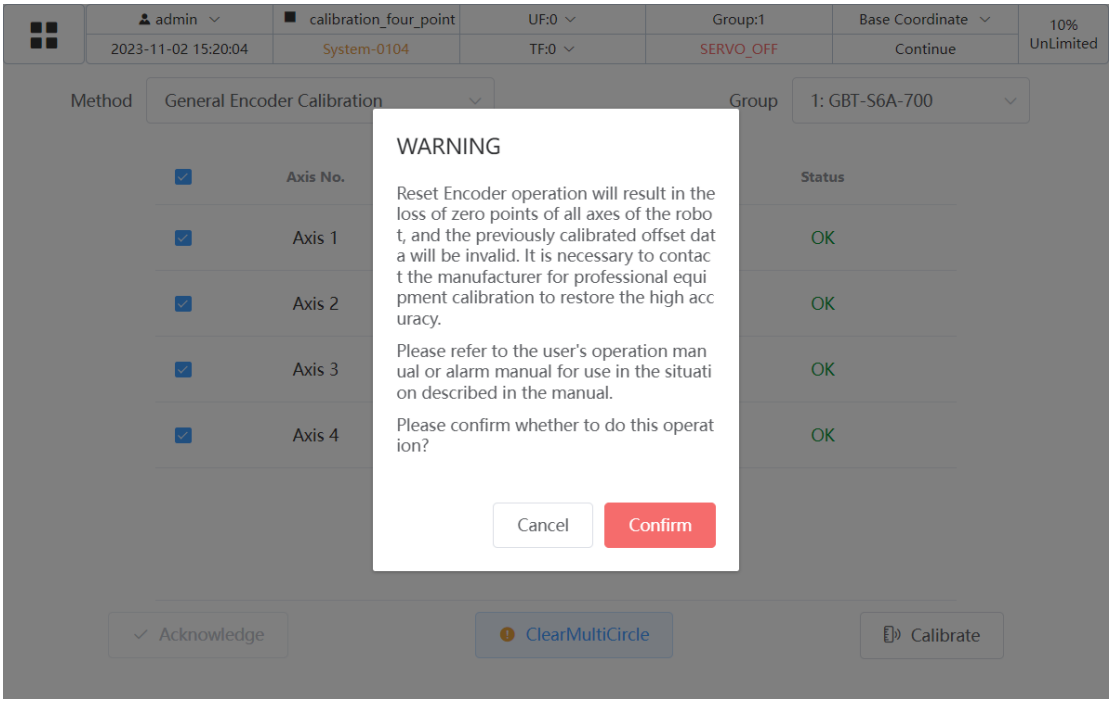


Fig. 2.48 Reset Warning Window

After confirming the operation, clear the multi-circle data of all encoders and set the zero-point loss flag regardless of whether the previous zero-point was lost. The user is required to perform zero-point calibration.

Unreasonable clearing of multi-circle data may affect the accuracy of the robot and should be done carefully. If the user accidentally clear multi-circle values, it may lead to a decrease in the accuracy of the robot. Then, a dedicated calibrator should be used to recalibrate and restore the accuracy of the robot. If necessary, it is recommended to contact the robot manufacturer to perform the calibration.

2.5.5 Description of zero-point calibration scenarios

Scenarios where it is necessary to press the "ClearMmultiCircle" function key:

- Encoder battery is depleted (in general, a battery undervoltage alarm is shown on the alarm bar, similar to: "Servo-4108 Axis 1 Encoder Battery Error, etc."). At this point, after replacing the battery, the user should firstly clear the multi-circle information of the encoder and then perform a new zero calibration.
- When replacing the motor, it is necessary to unplug the battery cable, and the encoder may lose power as well. So, after motor replacement and before zero calibration, it is also necessary to clear the multi-circle of the encoder first.

Scenarios where it is unnecessary to press the "ClearMmultiCircle" function key:

- A zero-point loss alarm or zero-point abnormality warning may appear. In this case, in case of a zero-point loss alarm, a zero-point calibration can be performed; in case of a zero-point abnormality warning, a pop-up of "Please manually confirm whether the zero point is normal" may appear when performing zero-point calibration. Click the Normal option, the warning will automatically disappear and the system will return to normal.

2.6 General setting

2.6.1 Time/Language setting

Successively click "Menu Button" → "Administration" → "Time/Language" to enter the screen as shown in Fig. 2.49.

The user can change current time and switch Chinese or English、Vietnamese、Korean、Japanese、Russian language.

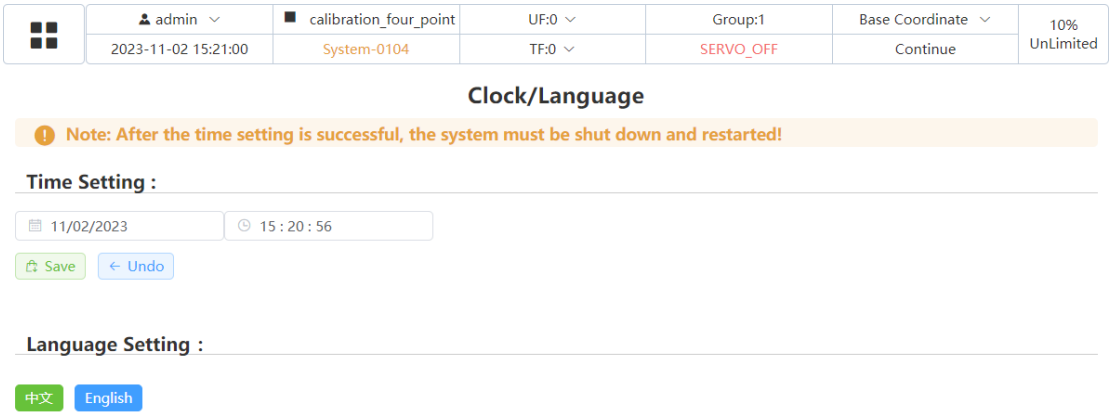


Fig. 2.49 Time/Language Setting Window

Caution

The time setting can be completed by shutting down and restarting the system.

2.6.2 Brightness setting

Successively click "Menu Button" → "System" → "Other Settings" → "Brightness Setting" to enter the screen as shown in Fig. 2.50.

The user can choose the brightness of the screen (default from 1-94).

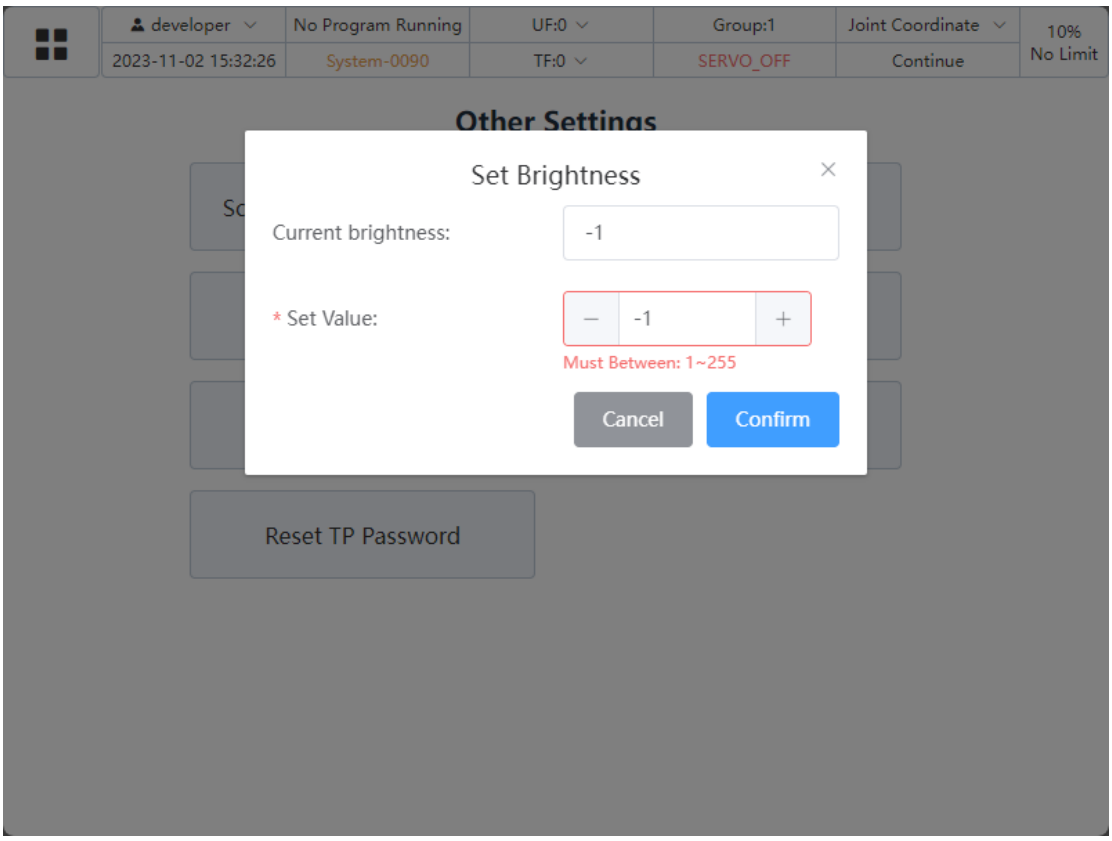


Fig. 2.50 Brightness Setting Window

2.6.3 Automatic screen-off

The screen is automatically off if the user does not operate for a certain period of time (default 10 min).

After screen off, the user can click the "Lock Screen" button to unlock the screen, which will light up again.

The user can also modify the automatic screen-off time in "Menu Button ->System ->Other Settings ->Preferences".

2.6.4 Modify IP

When being deployed independently and not connected to the subnet on the site, the

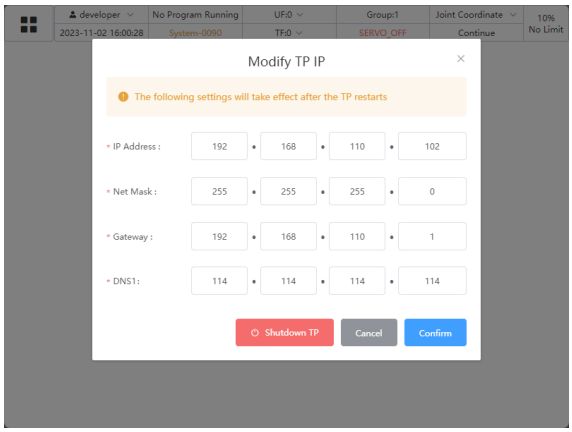
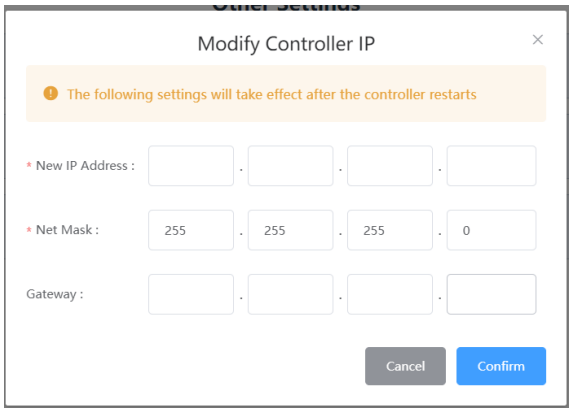
robot can work directly by default IP.

When the robot is integrated into a certain subnet on the site, it may be necessary to modify IP of the controller and TP based on the subnet's IP range.

When a new TP is replaced, the self-test may fail during the first startup due to incorrect IP configuration of the controller paired with TP. At this moment, there is a button to select a controller on the interface. The user can click it to reconfigure IP of the controller to which TP is connected.

Please contact the manufacturer in case of any problems.

Steps to modify IP:

Modify TP IP to an idle IP in the subnet: Successively click "Menu Button" → "System" → "Other Settings" → "Modify TP IP" to enter the "Modify TP IP" screen.	
Modify controller IP to an IP in the subnet: Successively click "Menu Button" → "System" → "Other Settings" → "Modify Controller IP" to enter the "Modify Controller IP" screen.	

Caution

If IP is modified to a different subnet, it is recommended to first modify the controller IP and then TP IP. Otherwise, they are on different subnets, possibly leading to the situation where the connection cannot be reached.

The robot controller and TP should be restarted after IP modification.

2.6.5 Find controller

This function is used to find which controller is in the same net as TP and to connect the controller (if found). This function is often available when a TP is compatible with multiple controllers.

Successively click "Menu Button" → "System" → "Other Settings" → "Find Controller" to enter the "Find Controller" screen as shown in Fig. 2.51.

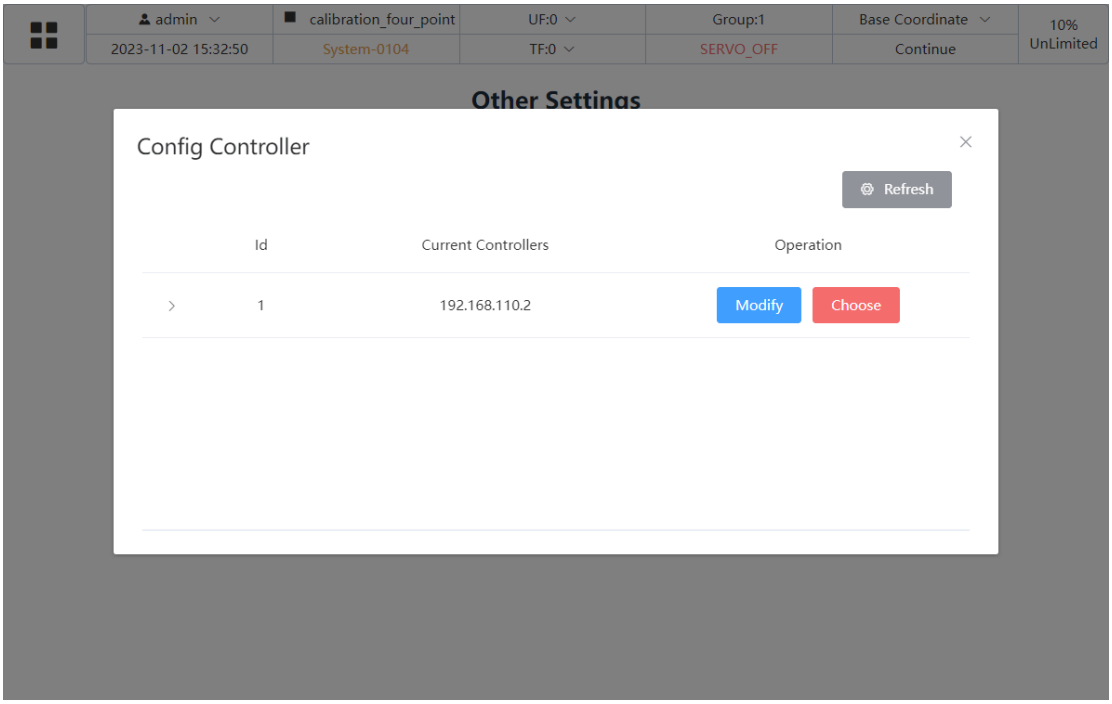


Fig. 2.51 Find Controller Interface

Click "Refresh" in the upper left corner of Fig. 2.51 to refresh the number of controllers in the current net. Click "Modify" to modify IP of the controller. Then, click "Connect" to connect TP to the corresponding controller.

2.6.6 Choose controller

This function is used to select a controller in the same net as TP. It is often available when a TP is compatible with multiple controllers.

Successively click "Menu Button" → "System" → "Other Settings" → "Choose Controller" to enter the "Choose Controller" screen as shown in Fig. 2.52. Enter the controller IP you want in the "Target Controller" and click "Confirm".

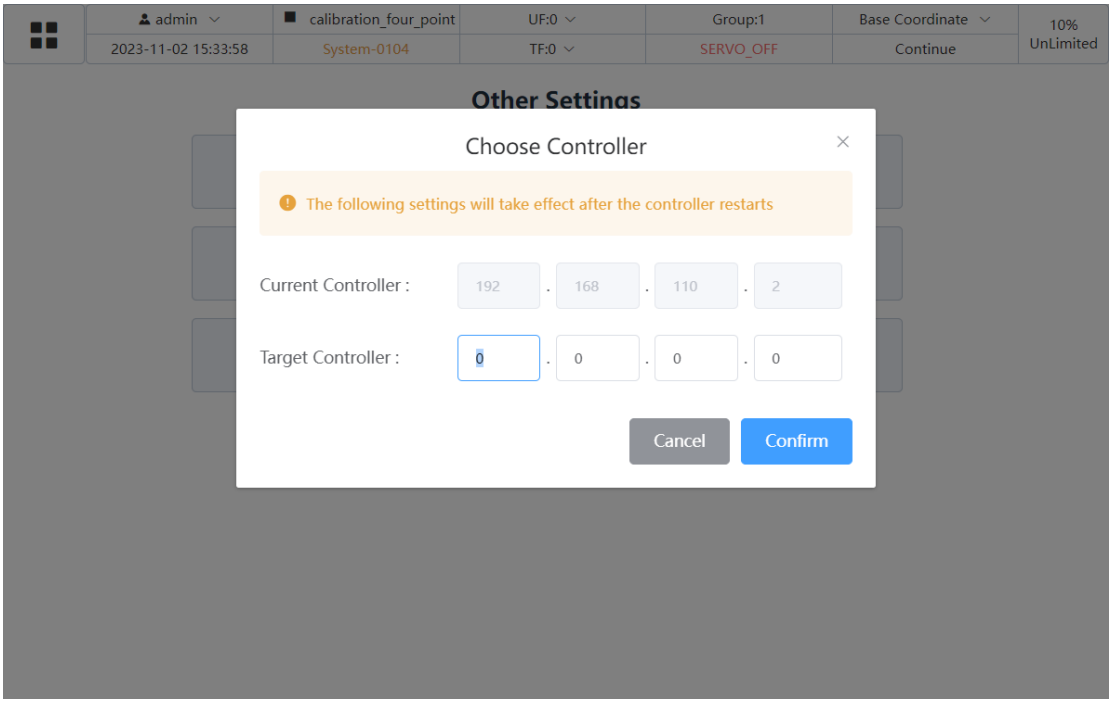


Fig. 2.52 Choose Controller Interface

2.7 System parameters

2.7.1 Type

- Model parameters: They are only related to the product model and derived from the design parameters generated by design and development. They may be mechanical parameters, controller parameters or verified performance parameters. The model parameters of the same model have the same format and numerical values, unless the design of the model has been changed.
- Robot parameters: They are physical parameters only related to a specific product or certain parameters not accurately calculated by the design theory. They are caused inevitably by inconsistencies in actual production and manufacturing and their values are different for each robot.
- User parameters: External interfaces are provided to allow the users to frequently change parameter values (the users are end users or function developers. Not all parameters are open to end users). These parameters are targeted towards application and based on application demands.
- Temporary parameters: They are parameter data temporarily recorded to meet certain mechanisms or software functions in the system. If being saved after power down, these parameters are generally recorded in NV-RAM. Otherwise, they are

stored in the memory. They are only related to the state of the robot at a certain moment, but unrelated to physical properties, algorithm variables or user settings of the robot.

2.7.2 Setting of general system variables

Successively click "Menu Button" → "System" → "General System Variables" to enter the screen as shown in Fig. 2.53.

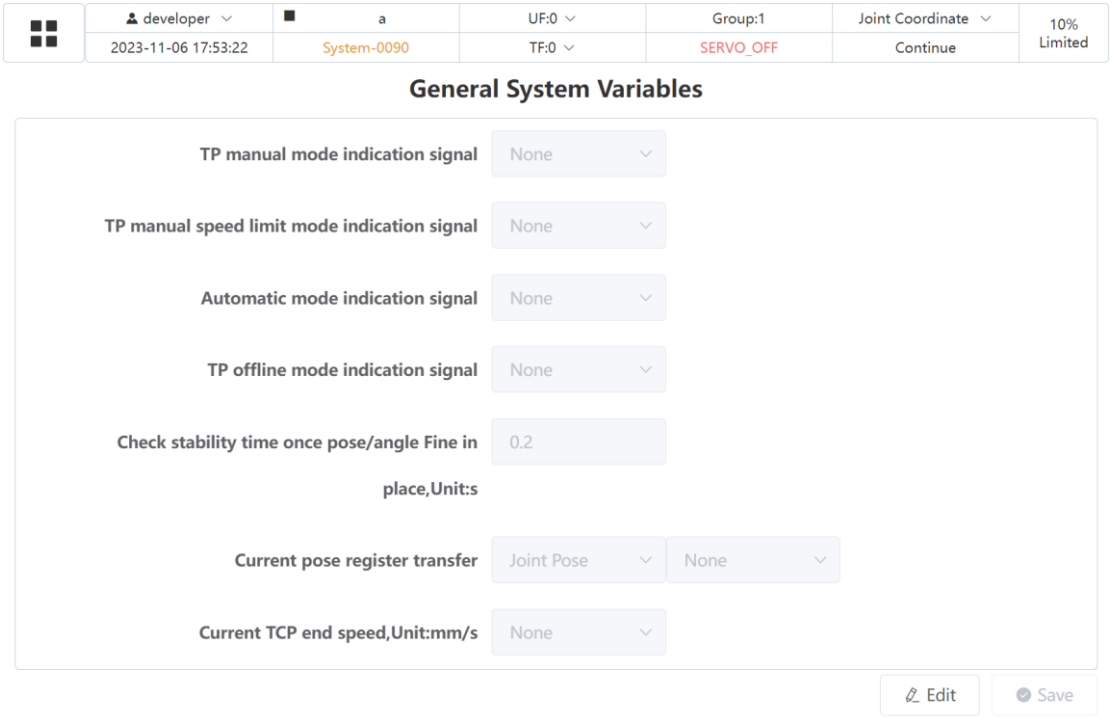


Fig. 2.53 System Variable Interface

Description of general system variable parameters:

Contents of general system variables	Meaning
TP manual mode indication signal	Select the configured digital output signal. The signal status is ON when the TP mode selector is in the manual maximum speed mode or off otherwise.
TP manual speed limit mode indication signal	Select the configured digital output signal. The signal status is ON when the TP mode selector is in the manual limit speed mode or off otherwise.

Automatic mode indication signal	Select the configured digital output signal. The signal status is ON when the TP mode selector is in the auto mode or off otherwise.
TP offline mode indication signal	The controller is not equipped with a TP.
Check stability time once pose/angle Fine in place	Check stability time once pose/angle Fine in place, unit: s
Current pose register transfer has two types: joint transfer and Cartesian transfer.	When joint transfer is adopted, the data in the configured PR register is the current joint data of the robot. When Cartesian transfer is adopted, the data in the configured PR register is the current pose data of the robot. The PR register contains all PR parameters, including pose parameters, number of rotations, and joint configuration. The data in this PR register will be updated every 8ms.
Current TCP end speed	The configured register will represent current TCP speed, which is updated every 8ms.

2.8 User level

Some functions and settings can only be accessed by the users with appropriate levels. There are three user levels in the system:

- Operator
- Engineer
- Administrator (Admin)

Default password:

Except for the operator, every level needs a password to activate the system. Different users have different operation authorities. (The operation authorities of different users can be found in the section of system authorities for operators in the Safety Instructions)

The operator does not have a default password. The default passwords for other users are shown in the table below.

User type	Default password
Robot engineer	(This password can be modified under admin privilege.)
Admin	123

The steps to log in as a user are as follows:

1. Click "Login" in the status bar of TP to enter the login interface, as shown in the following figure:

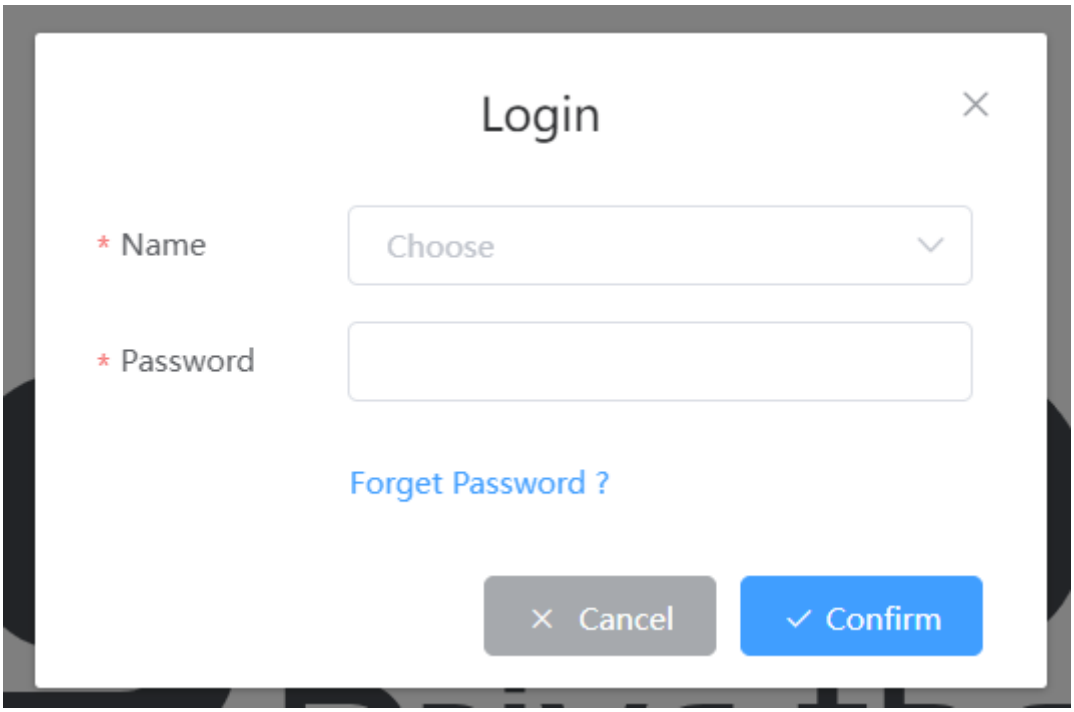


Fig. 2.54 Login Interface

2. Click “Choose” to select the type of user to log in.

3. Enter the user password of corresponding type and click "Confirm" to complete the login.

Steps for robot engineer to modify the password

1. Log in as the admin.

2. Successively click "Menu Button" → "User Management" to enter the user management interface, as shown in Fig. 2.55.

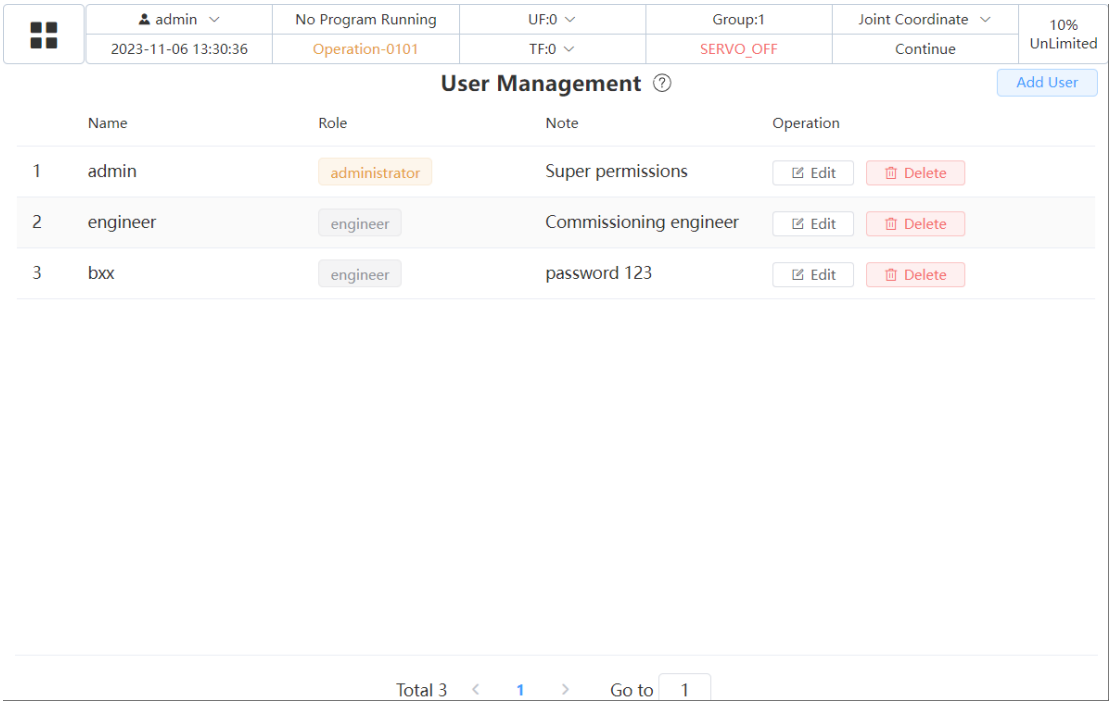


Fig. 2.55 User Operation Interface

3. Click "Edit" after the engineer to enter the “Edit User” interface, as shown in Fig. 2.56.

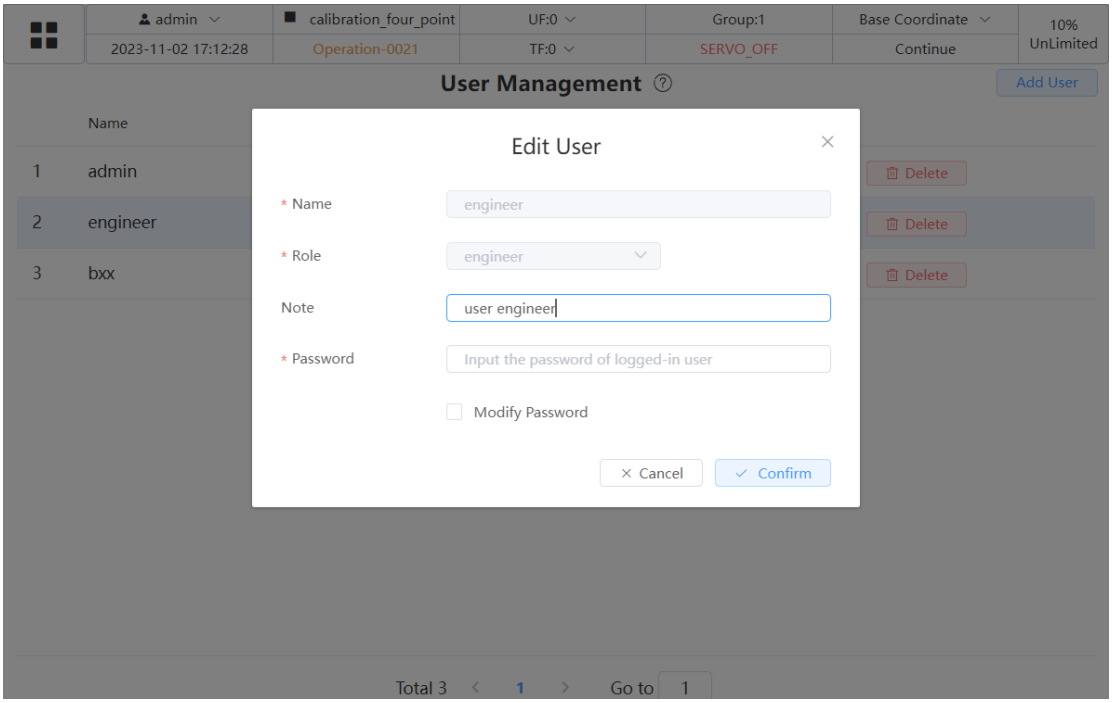


Fig. 2.55 “Edit User” Interface

4. Check the "Change Password" option to enter the password modification interface. , as shown in Fig. 2.57.

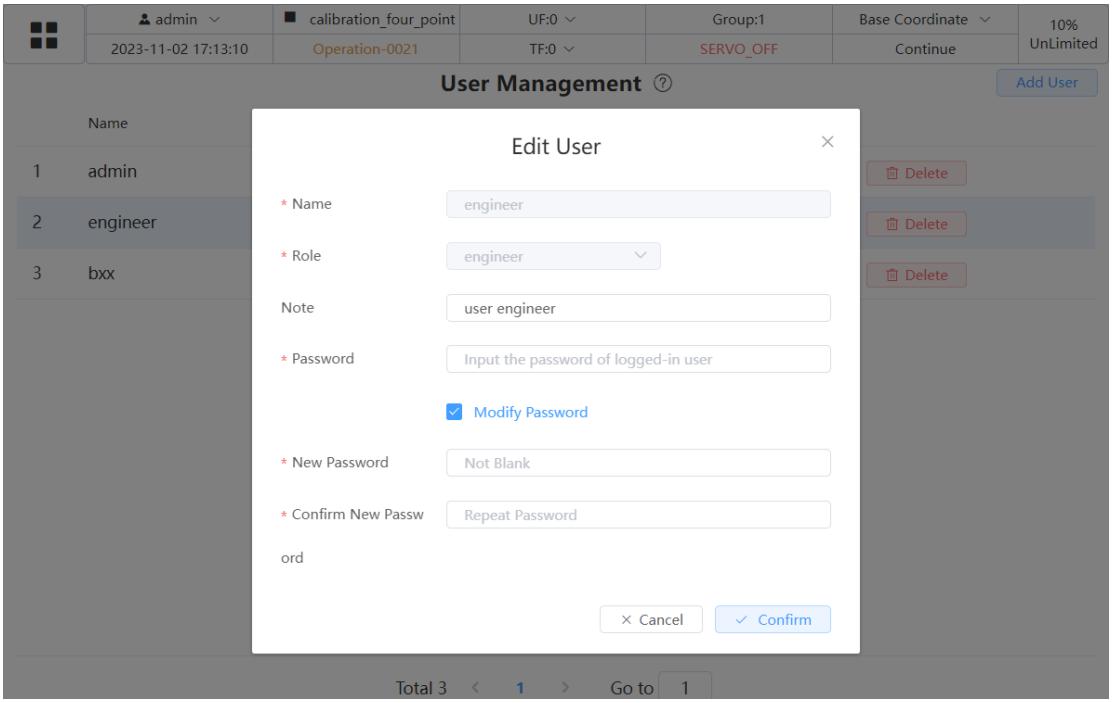


Fig. 2.55 “Modify Password” Interface

5. Enter 123 in the “Password” field, then enter “New Password” and “Confirm New Password”, and click “Confirm” to complete the password modification.

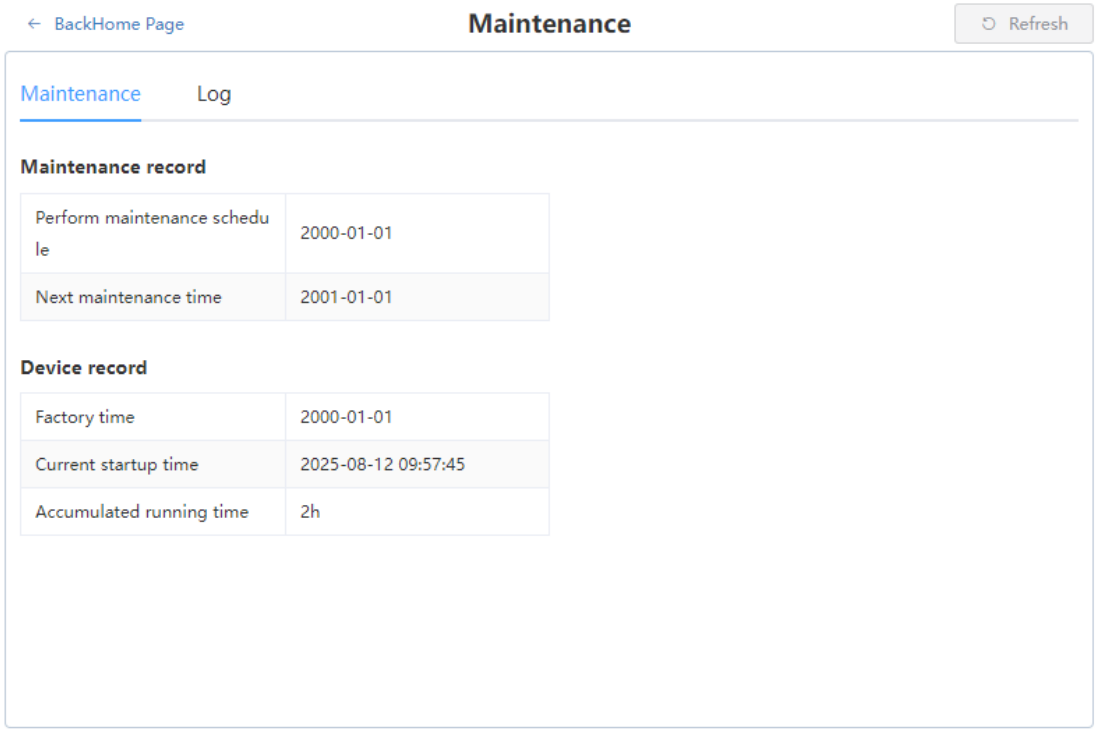
2.9 Maintenance

Menu → System → Maintenance, enter the equipment maintenance information interface.

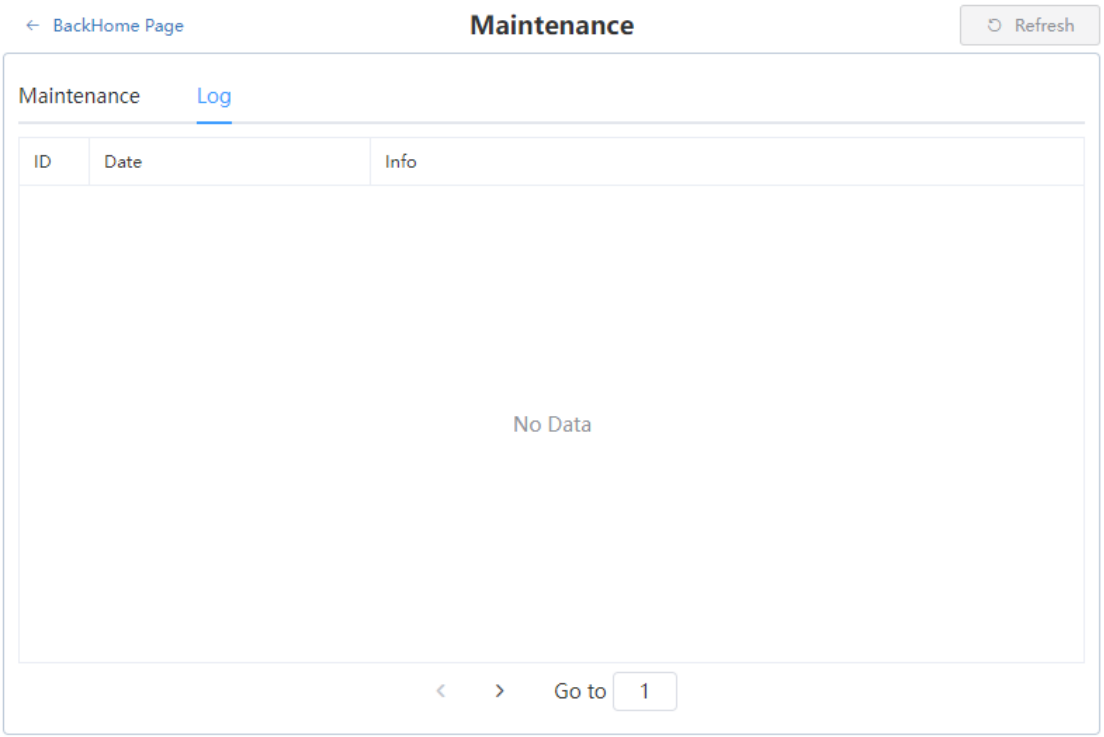
Maintenance Information

Maintenance records: Record the time of maintenance execution and the time of next maintenance.

Equipment records: Display the factory time, the startup time of this time, etc.



Log



3. Composition of program

This chapter describes the composition and instructions of the program.

A robot application consists of instructions and other accompanying information required by the robot to perform tasks and recorded by the user.

In addition to program information describing how the robot performs tasks, the program also includes detailed information on defining properties.

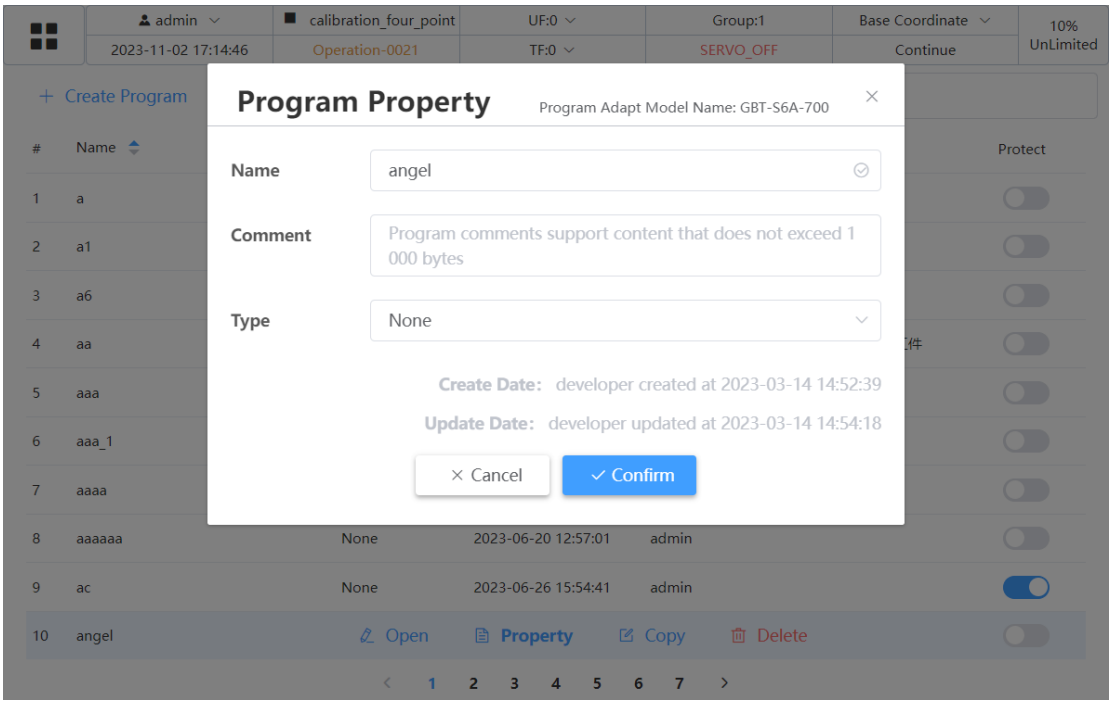


Fig. 3.1 Program Property Interface

The program property contains the following information:

Program name, program comment, program type, create date and update date

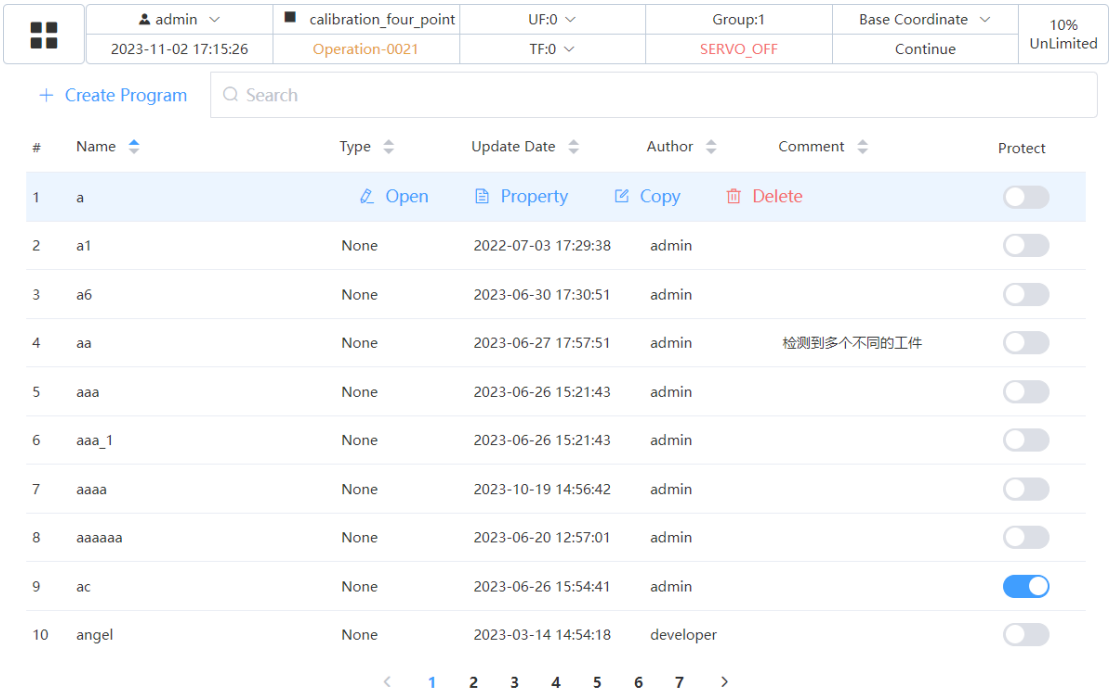


Fig. 3.2 Program List Interface

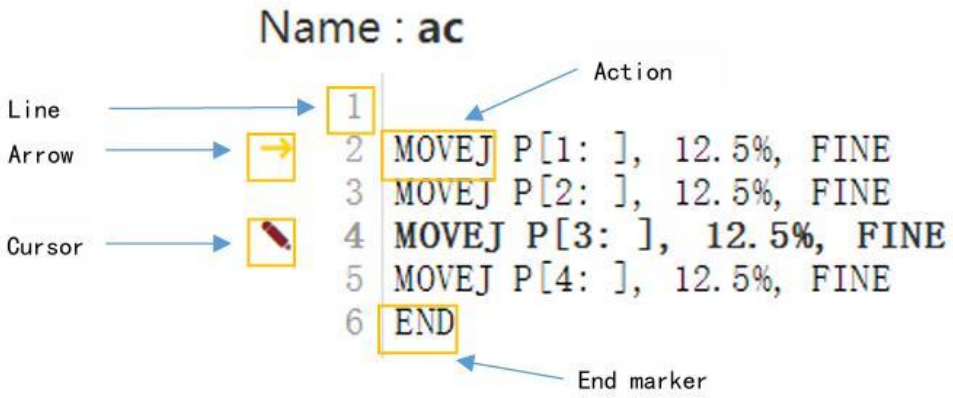


Fig. 3.3 Program Editing Statement Interface

The program is composed of the following information:

- Line number assigned to each program instruction.
- Action instruction instructing the robot to move in which direction and by which means.
- Program instruction containing the following contents.
  - Register instruction for storing numerical data.
  - Position register instruction for storing robot position data.
  - I/O (input/output) instructions for sending signals to and receiving signals

from peripheral devices.

- Transfer instructions (IF, WITCH, CALL) for changing the flow direction of the program when the defined conditions are met.
- Wait instruction for postponing program execution.
- Annotations added to the program.
- Other instructions

- The end marker indicates that no more instruction is left in the program.

3.1 Program property

The program properties are set on the program property screen. "Program property" is displayed when it is selected on the program list screen.

3.1.1 Program name

The program names are used to distinguish programs stored in the memory of the controller. It is not allowed to create more than 2 programs with the same name in the same controller.

Length

The length of the program name consists of 1 to 60 characters. The program name must be unique to the program.

Characters allowed

Letters: English letters.

Number: 0~9. The program name cannot start with a number.

Mark: Only _ (underline) allowed. @ and * cannot be used.

Contents

The program must be named in a way clarifying its purpose and function.

3.1.2 Comment

A comment can be added to the program name when a new program is created. The comment is used to describe additional information to display along with the program name on the program list interface.

Length

The length of the program comment consists of 1 to 1000 characters.

Characters allowed

Any character, number, symbol or Chinese character can be used.

Contents

The program comment must be described in a way clarifying the program purpose and function.

3.1.3 Program type

There are program types as follows:

- Main - The program started as the main program from TP or an external device.
- None - The program called as a subprogram from the working program and used to execute specific tasks.

Main - Main program:

It can be called by Local Trigger, MPLCS and MPLCS Simple Trigger. When the program type is selected as Main or Macro, a list of program start points and program numbers is additionally shown on the program property interface. The programmer is allowed to choose whether to disable the "Program Start Point" function. As shown in Fig. 3.5, the program number should be set only when the program type is selected as Main. Please see Section 4.3 for details of the program starting points; additional descriptions of Section 4.2.2 for details of

program numbers.

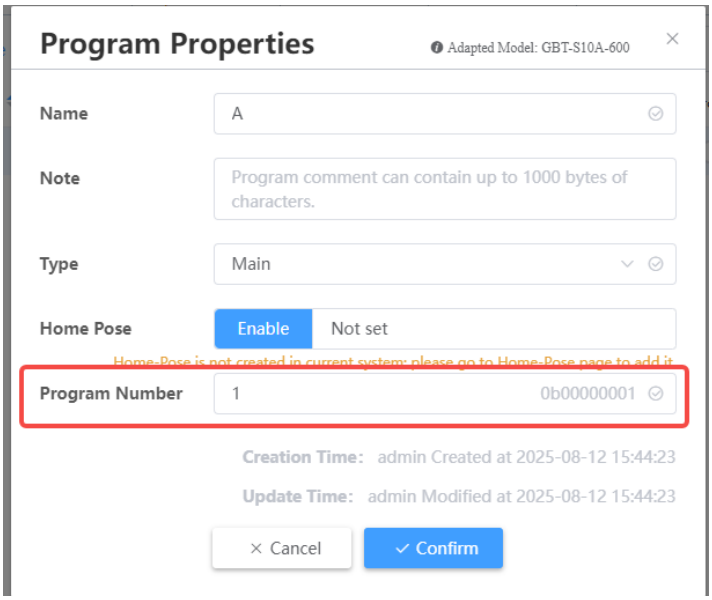


Fig. 3.4 Program Number Setting

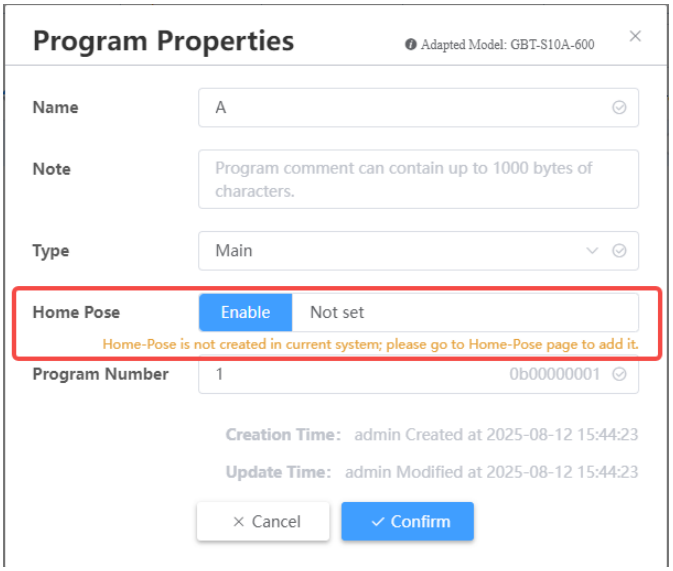


Fig. 3.5 Program Start Point Setting

None - General program:

There are no special restrictions or distinctions. In the auto mode, it cannot be started by any means other than Local Trigger mode (manually pressing the start button). However, it can be called by other programs.

3.1.4 Program protection

Write protection can be used to specify whether the program can be changed or deleted.

When program protection is enabled, it is not allowed to add data to the program or modify the program. After program commissioning, turn on program protection in the program list interface in order to prevent oneself or others from rewriting the program (Fig. 3.2 Program List Interface).

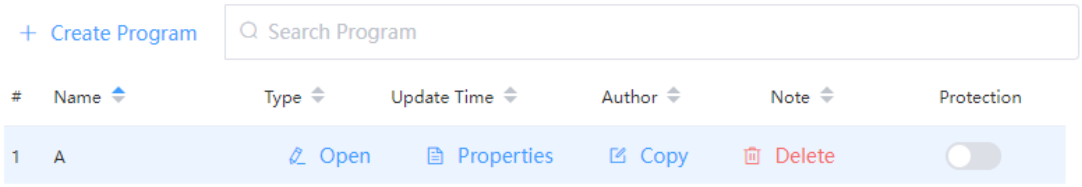


Fig. 3.2 Program List Interface

3.2 Line number, end marker, arrow and parameter

Line number

The line number is automatically inserted next to each instruction added to the program. When the instruction is deleted, the program automatically reassigns the number so that the initial line is always Line 1 and the second line is Line 2.

When the program is changed, the line number can be used to specify which line should be used as the object for moving, deleting or specifying the scope through the cursor.

In addition, the cursor can also be moved to the target line number by clicking "Jump To" in the program editing screen.

End marker

The end marker (END) is automatically displayed after the last instruction in the program. As new instructions are added, the end marker of the program can move towards the bottom of the screen while maintaining its position on the last line of the program.

When executing the last instruction to the end marker, the program ends and the arrow stops at the end marker.

Arrow

The line number pointed by the arrow indicates that the program has been executed to this line.

This chapter will explain program instructions required for program creation/modification in the following.

Programs are created/modified on the program editing screen (see Section 4.1 for program creation and modification).

Parameter i

The parameter i is the exponent used in the specification of control instructions (program instructions other than action ones). The parameter can be specified directly or indirectly. Direct specifying is usually to specify integers within the range of 1 to 32767. The range of values varies depending on the type of instruction used. Indirect specifying is to specify the register number.

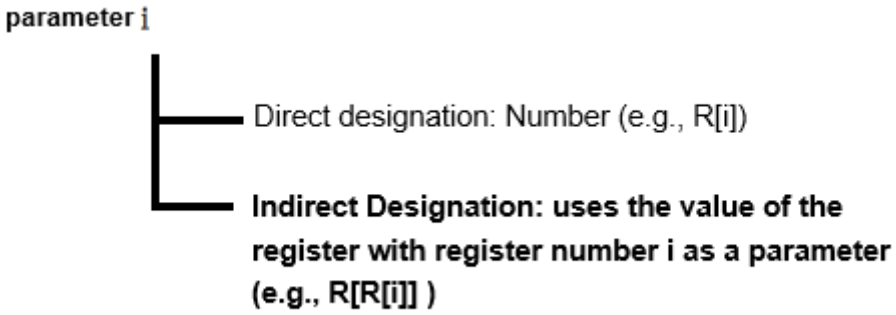


Fig. 3.6 Format of Parameter i

3.3 Action instruction

The action instruction refers to the instruction causing a robot to move towards a designated position in the workspace at a specified velocity and method. The contents specified in the action instruction are as follows. The instruction format is shown in Fig. 3.3.

- Action type - Specify trajectory control towards the specified position.
- Position data - Teach the robot where it will move.
- Movement speed - Specify the movement speed of the robot.
- Positioning type - Specify whether to locate at the specified position.
- Additional instruction - Specify the execution of an additional instruction in an action.

It is required to teach action instructions (see Section 4.1.5 for creation of action instructions and Section 4.1.6 for modification of action instructions).

3.3.1 Action type

Action type specifies travel trajectory towards the specified position. Action types

include: joint action without trajectory/pose control (MOVEJ), linear action with trajectory/pose control, and arm action.

- Joint action (MOVEJ)
- Linear action (MOVEL)
- Arc action (MOVEC)
- Jump action (only applicable to SCARA robots)

Joint action (MOVEJ)

MOVEJ is a basic method of moving a robot to a designated position. The robot accelerates and moves along all axes simultaneously, and then decelerates and stops simultaneously. The movement trajectory is usually non-linear. Record the type of action when teaching the end point. Joint movement velocity (%) is the percentage relative to maximum movement velocity. The pose of the moving tool is uncontrolled.

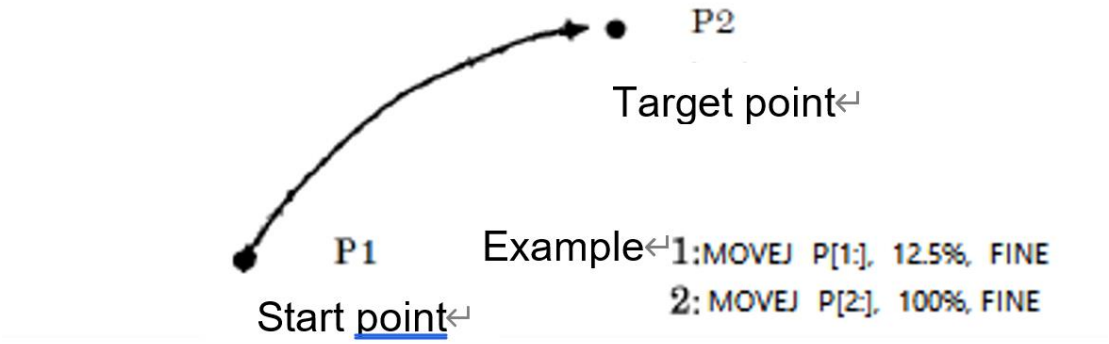


Fig. 3.7 Schematic Diagram of Joint Movement

Linear action (MOVEL)

MOVEL is a movement method that the movement trajectory of the tool center point from the start point to the end point is controlled linearly. Record the type of action when teaching the end point. The unit of linear movement velocity is mm/s. The pose of the moving tool is controlled after the poses of start and target points are separated.

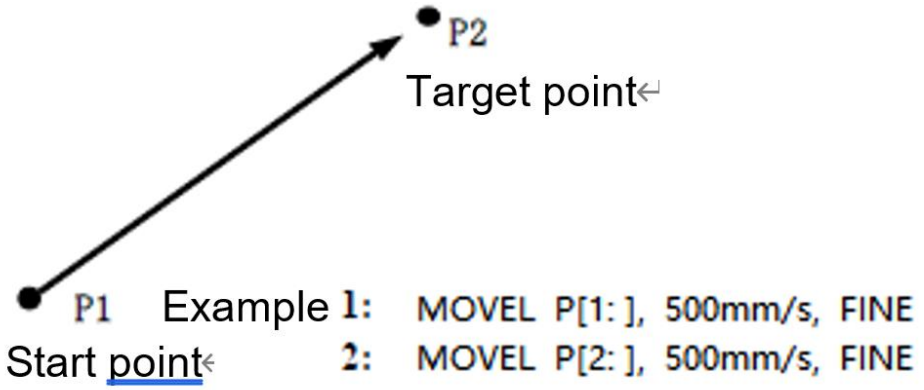
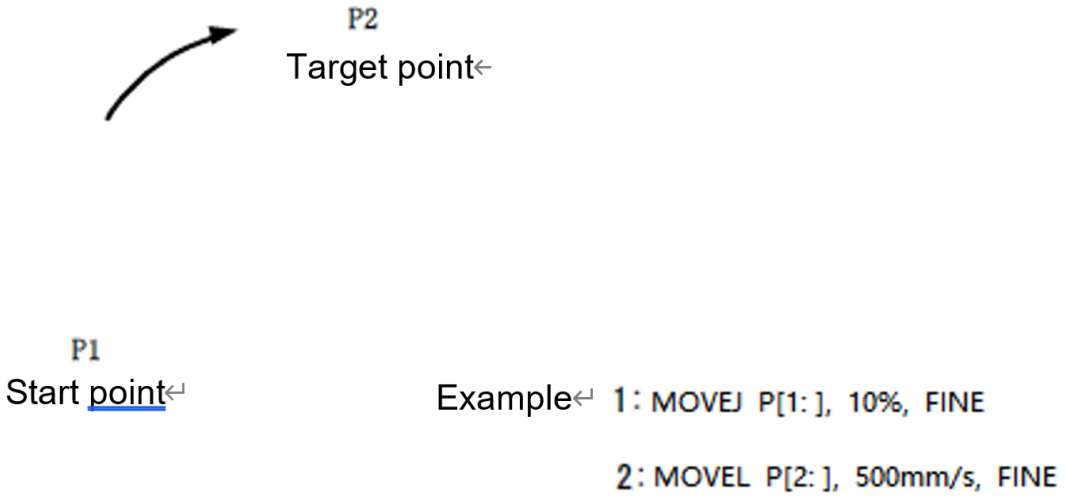


Fig. 3.8 Schematic Diagram of Linear Movement

The rotation action is a movement method that the linear action is adopted to rotate the pose of a tool from the start point to the end point around the tool center point. The pose of the moving tool is controlled after the poses of start and target points are separated. The movement trajectory (when the tool center point is moving) is controlled linearly.



Example

Fig. 3.9 Schematic Diagram of Rotary Movement

Arc action (MOVEC)

MOVEC is a movement method that the movement trajectory of the tool center point is controlled in an arc manner from the start point, via the passing point, to the end point. The passing point and target point are taught in an instruction. The unit of arc

movement velocity is mm/s. The pose of the moving tool is controlled after the poses of start, passing and target points are separated.

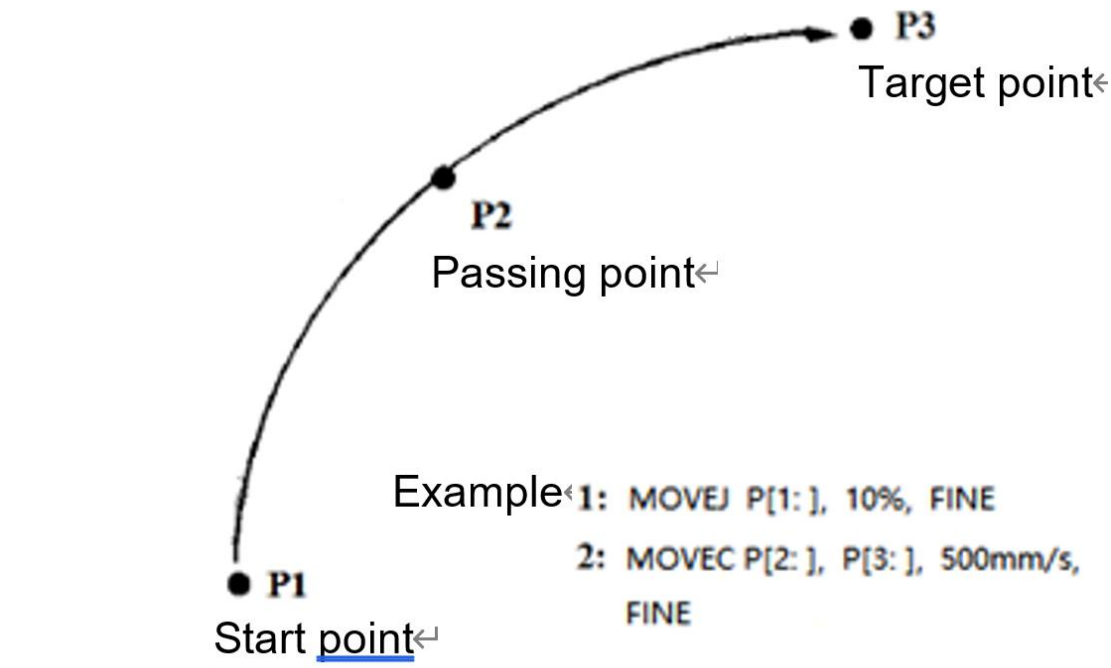


Fig. 3.10 Schematic Diagram of Arc Movement

JUMP action

The main motion mode of Jump is to lift Axis Z to a specified height, keep the moving trajectory directly above the target point and then lower it to the target point for control.

JUMP

The JUMP instruction is to direct the robot to perform a Jump motion (first vertically ascending, then horizontally moving and finally vertically descending). This instruction takes the current point of the robot (referring to the endpoint of the previous instruction during continuous automatic operation) as the starting point of JUMP and the point specified in the statement as the endpoint of JUMP. MOVEJ is adopted in the whole process.

Instruction format:

JUMP P [i], speed1, speed2, positioning type, LimZ, additional conditional instruction 1, additional conditional instruction 2

Description of statement parameters:

- P[i]: It represents the target position and can be specified directly by P[i] or

indirectly by PR[i].

- Speed1: The speed unit is mm/s. It can be directly specified by a constant or indirectly specified by MR[i]. It refers to the speed of rise or fall.
- Speed2: With the speed multiplier from 1 to 100%, it refers to the speed of horizontal movement.
- Positioning type: There are two types: SD and FINE, as detailed in Section 3.3.4.
- LimZ: It represents the upper limit of Coordinate Z of the robot in Jump motion and can be directly specified by a constant or indirectly specified by MR[i].
- Additional conditional instruction 1: optional (omitted). The instruction “Approach” has the general form of ARCH (Arch1, Arch2). Arch1: It can be an R register or a number. It can be used to specify the transfer distance (vertical distance from the starting point) before a horizontal action through the Jump instruction. (Unit: mm) Arch2: It can be an R register or a number. It can be used to specify the approach distance (vertical distance from the target point) for completely ending the horizontal movement through the Jump instruction. (Unit: mm)
- Additional conditional instruction 2: optional (omitted). The instruction has Offset, Tool Offset, TB, TA and DB for choosing (see Section 3.3.5 for details).

Example of instruction programming:

JUMP P[1:], 500mm/s, 12.5%, SD20, LimZ 130, ARCH (30 50), Offset

JUMP3

JUMP3 is an instruction requiring the robot to perform a jump motion (used to combine 2 linear actions and 1 joint action). It is applicable to SCARA robots. It contains a starting point and three designated points, of which adjacent two points form 3 line segments in sequence. The points specified in the linear motion statement of the first and last segments and in the PTP motion statement of the middle segment are used as the endpoints of the JUMP3 instruction.

Instruction format:

JUMP3 P[i], P[j], P[k], speed1, speed2, positioning type, additional conditional instruction

Description of statement parameters:

- P[i]: It is a transfer point higher than current position and can be specified directly by P[i] or indirectly by PR[i].
- P[i]: It is an approach point above the target coordinate point and can be specified directly by P[i] or indirectly by PR[i].
- P[k]: It is a target coordinate point and can be specified directly by P[i] or indirectly by PR[i].

- Speed1: The speed unit is mm/s. It can be directly specified by a constant or indirectly specified by MR[i]. It refers to the speed of Axis-Z rise or fall.
- Speed2: With the speed multiplier from 1 to 100%, it refers to the speed from P[i] to P[j].
- Positioning type: There are two types: SD and FINE, as detailed in Section 3.3.4.
- Additional conditional instruction: optional (omitted). The instruction has Offset, Tool Offset, TB, TA and DB for choosing (see Section 3.3.5 for details).

Example of instruction programming:

JUMP3 P[1:], P[2:], P[3:], 500mm/s, 12.5%, SD20, Offset

JUMP3CP

JUMP3CP is an instruction requiring the robot to perform a jump motion (used to combine 3 linear actions). It is applicable to SCARA robots. It contains a starting point and three designated points, of which adjacent two points form 3 linear segments in sequence.

Instruction format:

JUMP3CP3 P[i], P[j], P[k], speed, positioning type, additional conditional instruction

Description of statement parameters:

- P[i]: It is a transfer point higher than current position and can be specified directly by P[i] or indirectly by PR[i].
- P[i]: It is an approach point above the target coordinate point and can be specified directly by P[i] or indirectly by PR[i].
- P[k]: It is a target coordinate point and can be specified directly by P[i] or indirectly by PR[i].
- Speed: The speed unit is mm/s. It can be directly specified by a constant or indirectly specified by MR[i]. It refers to the speed of Axis-Z rise or fall.
- Positioning type: There are two types: SD and FINE, as detailed in Section 3.3.4.
- Additional conditional instruction: optional (omitted). The instruction has Offset, Tool Offset, TB, TA and DB for choosing (see Section 3.3.5 for details).

Example of instruction programming:

JUMP3CP P[1:], P[2:], P[3:], 500mm/s, SD20, Offset

3.3.2 Position data

Position data stores the position and pose of the robot. When action instructions are taught, position data is also written into the program.

Position data includes: joint coordinates based on joint coordinate system and Cartesian coordinates represented by tool positions and poses in the workspace.

Cartesian coordinate

The position data based on Cartesian coordinates is defined through four elements: TCP position (origin of tool coordinate system) in the Cartesian coordinate system, tool pose, form, and Cartesian coordinate system used.

The world or user coordinate system is used in the Cartesian coordinate system. The selection of these Cartesian coordinate systems will be discussed later in this section.

Position and pose

- Position (x, y, z): The TCP position (origin of tool coordinate system) in a Cartesian coordinate system is represented by 3D coordinates.
- Pose (A, B, C) - It is represented by the rotation angle around Axes X, Y and Z in the Cartesian coordinate system.

Form

Form refers to the pose of the main body of the robot. There are multiple forms meeting the conditions of Cartesian coordinates (X, Y, Z, A, B, C). To determine the form, it is necessary to specify the joint configuration and rotation number for each axis.

Joint configuration

The joint configuration represents the configuration of wrist and arm. It is to specify which side the control points of wrist and arm are located relative to the control surface. The robot is located at a singular point (special pose) when the control points are overlapping with each other on the control surface. The robot cannot operate for there are infinite forms based on specified Cartesian coordinates on the singular point.

- The robot cannot work at positions located at singular points. (In some cases, the most obtainable form can be selected for operation.) In this case, teaching can be performed through joint coordinates.
- In straight lines/arcs, the robot cannot pass through singular points on the path (joint configuration cannot be changed). In this case, joint movement is adopted.

Rotation number

The number of rotation represents the rotation number of the axis (J4) for a SCARA robot and of the axes (J3, J4, J6) for a PUMA robot. These axes return to the same positions after a rotation. So, it is required to specify the number of rotations. The rotation number of each axis is 0 at a position of 0°.

In the case of programmed linear and arc actions, the robot moves towards the target

point in the closest pose leaving from the starting point. At this moment, the rotation number at the target point is automatically selected. So, the actual rotation number of the robot at the target point may differ from that shown in the position data in some cases.

Verify Cartesian coordinate system

The verification of the Cartesian coordinate system is to detect which coordinate system number is used when the positional data based on Cartesian coordinates are reproduced.

If 0-10 is specified in the number of tool coordinate system and in the number of user coordinate system, the coordinate system number during teaching should be the same as that during reproduction. Otherwise, the robot collision may occur. The coordinate system number is written into the position data during position teaching. To change the written coordinate system number, it is required to change (activate) the coordinate function by the tool/user and then teach the point again.

Number of tool coordinate system (UT)

The number of tool coordinate system is specified by the number of the end flange or tool coordinate system. The coordinate system on the tool side is determined accordingly.

- 0: The default tool coordinate system (end flange coordinate system) is used.
- 1-10: The tool coordinate system with the specified number is used.

Number of user coordinate system (UF)

The number of user coordinate system is specified by the number of the world or user coordinate system. The coordinate system of the workspace is determined accordingly.

- 0: The base coordinate system is used.
- 1-10: The user coordinate system with the specified number is used.

Position variable - P[i]

The position variable is a standard variable for position data storage. The position data is automatically recorded when the action instruction is taught.

The following Cartesian coordinate system and number are used when Cartesian coordinates are taught.

- Coordinate system with currently selected tool coordinate system number (UT=0-10).
- Coordinate system with currently selected user coordinate system number (UF=0-10).

The following Cartesian coordinate system and number are used during

reproduction.

- Coordinate system with taught tool coordinate system number (UT=0-10).
- Coordinate system with taught user coordinate system number (UF=0-10).

Position register - PR[i]

The position register is a universal variable used to store position data. (For position teaching of position register, see Section 5.1.3)

The following Cartesian coordinate system and number are used when Cartesian coordinates are taught.

- Coordinate system with currently selected tool coordinate system number (UT=0-10).
- Coordinate system with currently selected user coordinate system number (UF=0-10).

The following Cartesian coordinate system and number are used during reproduction.

- Coordinate system with taught tool coordinate system number (UT=0-10).
- Coordinate system with taught user coordinate system number (UF=0-10).

Caution

Due to no UF/TF in the PR register, it represents the pose of the specified TF under the currently activated UF during program execution. So, switching between different UF/TF may result in different positions of the actual robot in space.

P represents local pose data, including UF/TF. If P is used in the motion instruction but inconsistent with UF_No and TF_No, the program cannot continuously execute downwards and it is required to modify UF_No and TF_No to make them consistent with P record.

Position number - P[i]

Names (containing 31 characters at most) can be added to the position number and the position register number.

Example: MOVEJ P[1:point1], 10%, FINE

MOVEJ PR[1:point2], 10%, FINE

3.3.3 Movement speed

The robot's movement speed can be specified in the motion instruction. The movement speed is limited by the speed multiplier during program execution. The speed multiplier ranges from 1 to 100%. The unit specified in the movement speed varies according to the type of action taught by the action instruction.

MOVEJ P[1:], 10%, FINE

When the action type is MOVEJ, the ratio of relative maximum movement speed can be specified in the range of 1-100%.

MOVEL P[2:], 500mm/s, FINE

When the action type is MOVEL or an arc action, the unit is mm/s and the ratio is specified between 1 and 4000 mm/s.

Specify the speed through the motion register (MR).

It is allowed to specify the speed through the motion register. Thus, the movement speed of the action instruction can be specified after it is calculated in the motion register.

Caution

This function can be used to freely change the robot's movement speed through the motion register. Therefore, sometimes it may cause the robot to act at unexpected speeds according to the specified motion register value. When this function is adopted, it should be noted to fully confirm the specified motion register value during teaching/operation.

Display of action instruction for the movement speed specified through the motion register

- MOVEJ P[1:], MR[i:]%, FINE
- MOVEL P[2:], MR[i:]mm/s, FINE
- MOVEC P[3:], P[4:], MR[i:]mm/s, FINE

3.3.4 Positioning type

The end-of-action method can be defined for the robot in the action instruction based on the positioning type. There are two positioning types under standard circumstances.

- FINE positioning type
- SD positioning type

FINE positioning type

MOVEJ P[1:], 10%, FINE

According to the FINE positioning type, the robot stops at the target position (positioning) and then moves towards the next target position.

SD positioning type

MOVEJ P[1:], 10%, SD 50

According to the SD positioning type, the robot approaches the target position and proceeds to the next position with no complete stop at that position.

The extent that a robot approaches the target position is defined by a value between 0 and 1000.

When 0 is defined, the robot moves at the closest position to the target and proceeds to the next position with no complete stop at that position.

The greater the SD value, the lower the deceleration of the robot near the target position and the farther away the robot from the target position.

Caution

In the case of programmed wait and other instructions after the action instruction of SD is specified, the robot may stop on the trajectory of the corner and execute this instruction under standard settings.

When multiple actions are continuously performed with short distance and fast speed in the SD mode, even the SD value of 1000 may cause the robot to slow down.

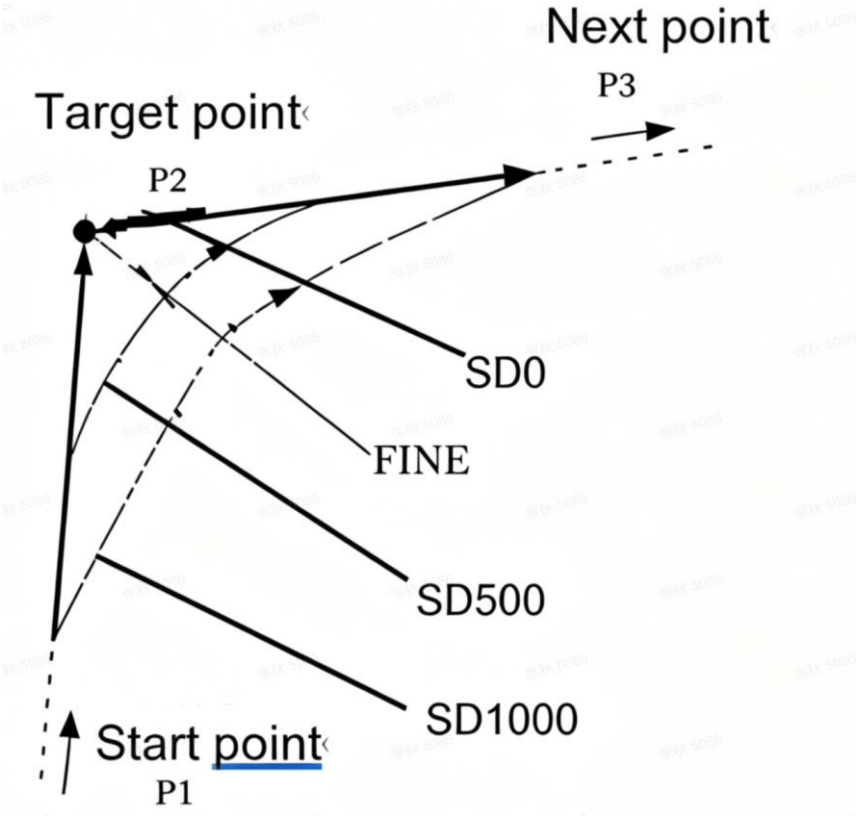


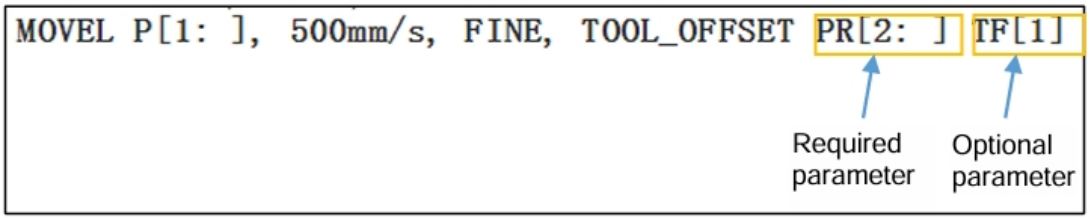
Fig. 3.11 Schematic Diagram of Corner Radius

3.3.5 Additional instructions for actions

Additional instructions cause a robot to perform specific tasks during its actions. There are additional instructions as follows.

- Tool offset instructionTool_Offset
- Position offset instructionOffset
- Trigger before instructionTB
- Trigger after instructionTA
- Distance-based trigger instructionDB
- Motion parameter switch instructionSwift
- Remote tool center point instructionRTCP
- Skip instructionSKIP

3.3.5.1 Tool offset instruction (tool center point)



Required parameter: It represents offset with the type of PR register.

Optional parameter: Tool coordinate system as offset reference

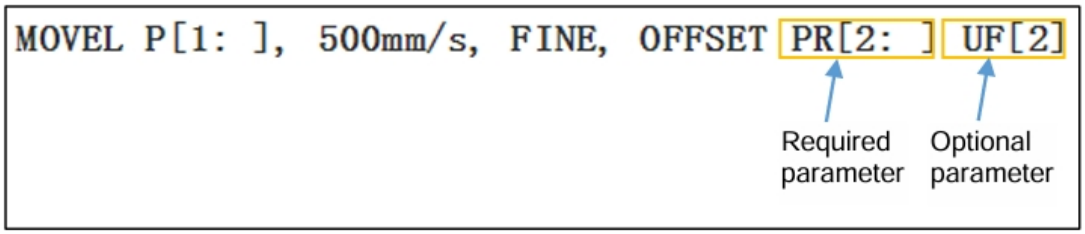
Tool offset instruction - The target position recorded in the position data causes the robot to move to a position only offsetting the offset amount specified in the tool offset condition. The offset condition is specified by the tool offset condition instruction.

The tool offset conditions specify the following elements:

- The position register specifies the direction and amount of offset.
- The tool coordinate system is used when offsetting.
- The current tool coordinate system is used if a tool coordinate system is not specified as the offset reference. If a tool coordinate system is specified as the offset reference, the offset is based on the specified coordinate system.

If the basic motion instruction is MoveJ (the offset object point, i.e. target P[n] in the Move instruction, is executed in joint form), it will be transformed into joint coordinates to execute the offset on the joint even if the expression specifying the offset PR[x] is in a Cartesian coordinate system. However, the transformation process is executed based on the offset reference specified by the TF[i] parameter.

3.3.5.2 Position offset instruction (Offset)



Required parameter: It represents offset with the type of PR register.

Optional parameter: User coordinate system as offset reference

Position offset instruction - The target position recorded in the position data causes the robot to move to a position only offsetting the offset amount specified in the position offset condition. The offset condition is specified by the position offset condition instruction.

The position offset conditions specify the following elements:

- The position register specifies the direction and amount of offset.
- The offset of the joint is used when the position data is the joint coordinate.
- When the position data is a Cartesian coordinate, the user coordinate system specified by the optional parameter is used as the offset reference. If not specified, the current user coordinate system (UF) is used.

In addition to PR, a specific point information can also be used as a required parameter in the position offset instruction. C_VEC (X, Y, Z, A, B, C) or J_VEC (J1, J2, J3, J4, J5, J6) directly specifies the offset. Among them, the data is of Cartesian type in C_VEC and joint type in J_VEC.

If the basic motion instruction is MoveJ (the offset object point, i.e. target P[n] in the Move instruction, is executed in joint form), it will be transformed into joint coordinates to execute the offset on the joint even if the expression specifying the offset PR[x] is in a Cartesian coordinate system. However, the transformation process is executed based on the offset reference specified by the UF[] parameter.

3.3.5.3 Trigger before instruction (TB)

Perform signal output and call the subprogram before the specified time when the robot action ends (excluding motion instructions. If the subprogram contains motion instructions, an error will be reported with the error code "Program-2328"). The specified time range is 0.01-30s. The input smaller than 0.01 will not be saved successfully (except for 0).

Format:

- Action statement + TB + specified time + CALL program name
- Action statement + TB + specified time + output signal

DO[i] can be selected as the output signal.

Example:

- MoveJ P[1] 100% FINE TB 1.00S CALL PROGRAM

  The robot moves to P1 through joint motion, and calls the program in the first 1s before reaching P1.

- MoveJ P[1] 100% FINE TB 1.00S DO[1]=ON

  The robot moves to P1 through joint motion, and sets DO[1] to ON in the first 1s before reaching P1.

3.3.5.4 Trigger after instruction (TA)

Perform signal output and call the subprogram after the specified time when the robot action ends (excluding motion instructions. If the subprogram contains motion

instructions, an error will be reported with the error code "Program-2328"). The specified time range is 0.01-30s. The input smaller than 0.01 will not be saved successfully (except for 0).

Format:

- Action statement + TA + specified time + CALL program name
- Action statement + TA + specified time + output signal

DO[i] can be selected as the output signal.

Example:

- MoveJ P[1] 100% FINE TA 1.00S CALL PROGRAM

  The robot moves to P1 through joint motion, and calls the program in the first 1s after reaching P1.

- MoveJ P[1] 100% FINE TA 1.00S DO[1]=ON

  The robot moves to P1 through joint motion, and sets DO[1] to ON in the first 1s after reaching P1.

3.3.5.5 Distance-based trigger instruction (DB)

When the robot TCP moves to a distance within a specified range from the target position of the motion instruction, this instruction allows the moving robot to execute signal output or call subprogram (excluding motion instructions).

Format:

- Action instruction + DB specified distance + CALL program name
- Action instruction + DB specified distance + Signal output

DO[i] can be selected as the output signal.

Parameter:

- Specified distance: When entering a spherical range centered around the target point of the motion instruction, TCP executes the trigger instruction. The radius of this spherical range is determined by the specified distance.

Example:

- MOVEL P[1] 100% FINE DB 100.00mm, DO[3]=ON

  The robot moves towards P1 in a linear motion. Set DO[3] to ON when the robot's TCP enters a space with P1 as the spherical center and a radius of 100mm.

- MOVEL P[2] 100% FINE DB 50.00mm， CALL A

  The robot moves towards P1 in a linear motion. Call subprogram when the robot's TCP enters a space with P1 as the spherical center and a radius of 100mm.

3.3.5.6 Motion parameter switch instruction (Swift)

SWIFT instruction: It is used to switch motion parameters (acceleration, acceleration, etc.) to meet faster beat requirements.

Example:

- MOVEJ P[1] 50% FINE, SWIFT

3.3.5.7 Remote tool center point coordinate system instruction (RTCP)

When executing remote TCP actions, it is necessary to add an additional instruction RTCP of remote tool action in the instructions. The RTCP coordinate system is essentially the same as the user coordinate system and the data is also the same. Only when the RTCP additional instruction is added into the motion instruction, the user coordinate system adopted for pose recording may become the RTCP coordinate system. In case of no RTCP additional instruction, it still maintains the user coordinate system. This change is reflected in the algorithm processing in the control system. No clear distinction is made between the RTCP coordinate system and the user coordinate system on the configuration page.

RTCP cannot be used during joint movement.

Example:

Relative to remote TCP, move the workpiece to P[1] at a relative velocity of 2000mm/s:

MoveL P[1] 2000mm/s Fine RTCP

Relative to remote TCP, move the workpiece from P[1] to P[2] at a relative velocity of 2000mm/s:

MoveC P[1] P[2] 2000mm/s Fine RTCP

3.3.5.8 Skip instruction (SKIP)

The SKIP instruction is used to directly jump from the current line containing the SKIP instruction to the target instruction line or program when the jump condition set by the SKIP CONDITION instruction (see Section 3.6.5) is met. It is often used in conjunction with the motion instruction SKIP CONDITION.

Instruction format: action instruction + SKIP GOTO LABEL [i]/CALL + program name

Example:

1 SKIP CONDITION DO[2]=ON

2 MOVEL P[1], 200 mm/s, FINE, SKIP GOTO LABEL[1]

3 MOVEJ P[2], 12.5%, FINE

4 LABEL[1]

5 MOVEJ P[3], 12.5%, FINE

The program starts executing from the first line. When DO[2]=ON, the program jumps to the fourth line after the second line ends and continuously executes downwards from the fourth line. When DO[2]=OFF, the program continuously executes downwards from the current line after the second line ends.

3.3.5.9 Acceleration Multiplier Command (ACC)

The ACC only acts on a single motion command. When the motion command does not use the ACC additional command, the global acceleration proportional coefficient is used for motion planning. When the motion command uses the ACC additional command, the ACC additional command is used as the acceleration proportional coefficient for planning.

Format: ACC 80%

Parameters:

- Acceleration ratio: Acceleration ratio percentage, ranging from 10% to 100% (modification of this value is only allowed with manufacturer permission!).

Example:

MOVEL P[1], 500 mm/s, FINE, ACC 80%

3.4 Register instruction

The register instructions perform arithmetic operations on registers. There are several types of registers.

- Number register instruction
- Position register instruction
- Position register element instruction
- String register and string instruction

- Motion register instruction
- Modbus special register instruction
- Palletizing Register instruction

Caution

All registers can be called directly through numerical values, or indirectly through combinatorial operation of number registers. They are called direct specifying method and indirect specifying method.

The expression examples are as follows: PR[R[20]], SR[20+R[2]], R[R[1]+R[2]]

As for the register operations, the following polynomial operations can be performed.

Example

R[2]=R[3]-R[4]+R[5]-R[6]

R[10]=R[2]*100/R[6]

3.4.1 Number register instruction

A number register is a variable used to store an integer or decimal value. 1500 number registers are available under standard circumstances.

R [i]=(value), where i is the number of the number register (1-1500)

The value can be:

- Constant
- R[i]: Value of register R[i]
- PR[i, j]: Value of position PR element [i, j]
- DI[i]: Digital input signal
- DO[i]: Digital output signal
- UI[i]: System input signal
- UO[i]: System output signal
- String
- Method: SIN, COS, etc. (see Section 3.10 for specific methods)

The arithmetic symbols of a number register include addition (+), subtraction (-), multiplication (*), division (/), remainder (MOD), and integer part of the quotient (DIV).

Example:

R[i] = (value) + (value)

R[i] = (value) + (value) instruction is to substitute the sum of two values into a number register.

R[i] = (value) - (value)

R[i] = (value) - (value) instruction is to substitute the difference between 2 values into a number register.

R[i] = (value) * (value)

R[i] = (value) * (value) instruction is to substitute the product of two values into a number register.

R[i] = (value) / (value)

R[i] = (value)/ (value) instruction is to substitute the quotient of 2 values into a number register.

R[i] = (value) MOD (value)

R[i] = (value) MOD (value) instruction is to substitute the remainder of 2 values into a number register.

R[i] = (value) DIV (value)

R[i] = (value) DIV (value) instruction is to substitute the integer part of the quotient of 2 values into a number register.

R[i] = (x-(x MOD y))/y

3.4.2 Position register instruction

The position register instructions perform arithmetic operations on position registers. The position register instruction can perform substitution, addition and subtraction processing.

A position register is a variable used to store position data (X, Y, Z, A, B, C). 200 position registers are available under standard circumstances.

PR[i] = (value)

PR[i] = (value) instruction is to substitute the position data into the position register. Where, i is the number of the position register (1-200).

The value can be:

- PR[i]: Value of position register [i]
- P[i]: Value of teaching position [i] in the program
- L_POS: Cartesian coordinate of current position (see Section 3.8.4)

- J_POS: Joint coordinate of current position (see Section 3.8.3)

Example:

PR［1］= L_POS

PR［2］= PR［3］

PR［3］= P[1:]

PR[4] = PR[R[1]]

3.4.3 Position register element instruction

The position register element instruction performs arithmetic operations on position registers.

In PR[i, j], i represents the number of the position register and j represents the number of the position register element. The position register element instruction can perform substitution, addition and subtraction processing, and is described in the same way as the number register instruction.

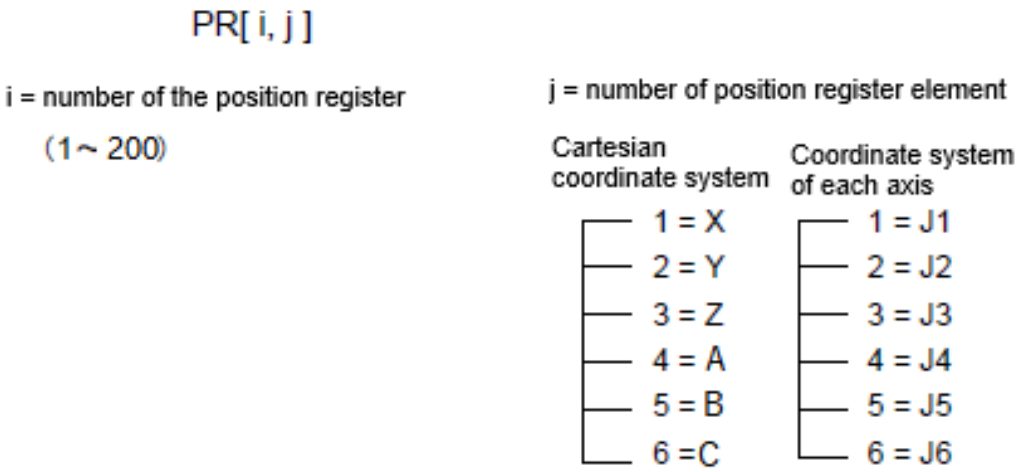


Fig. 3.11 Diagram for Position Register Element

PR[i, j] = (value)

PR[i, j] = (value) instruction is to substitute the element value of the position data into the position register element.

Where, i is the number of the position register (1-200).

The value can be:

- Constant
- R[i]: Value of register [i]

- PR[i, j]: Value of position PR element [i, j]
- MI[i]: Modbus register (hold type)
- MH[i]: Modbus register (input type)

Refer to Section 3.4.6 for Modbus registers.

Example:

PR［1, 1］= R［3］

PR［4, 3］= 324.5

PR[i, j] = (value) + (value)

PR[i, j] = (value) + (value) instruction is to substitute the sum of two values into a position register element.

PR[i, j] = (value) - (value)

PR[i] = (value) - (value) instruction is to substitute the difference between 2 values into a position register element.

PR[i, j] = (value) * (value)

PR[i, j] = (value) * (value) instruction is to substitute the product of two values into a position register element.

PR[i, j] = (value) / (value)

PR[i, j] = (value) / (value) instruction is/ substitute the quotient of two values into a position register element.

PR[i, j] = (value) MOD (value)

PR[i, j] = (value) MOD (value) instruction is to substitute the remainder of two values into a position register element.

PR[i, j] = (value) DIV (value)

PR[i, j] = (value) DIV (value) instruction is to substitute the integer part of the quotient of 2 values into a position register element.

PR［4, 3］= PR［1, 3］- 3.528

3.4.4 String register and string instruction

For string registers, a maximum of 255 characters can be stored in each register. Maximum number of string registers is 300.

SR[i] = (value)

SR[i] = (value) instruction is to substitute string data into the string register.

The value can be: Constant, (number register R[i], string register SR[i]), String,

Method

It can be transformed from numerical data to string data. Retain 6 decimal places and round off excess parts.

It can be transformed from string data to numerical data. The transformation value is the numerical value before the first character in the string.

Example SR[i]=R[j]

Value of R[j]	Result of SR[j]
R[j]=1234	SR[j]=1234
R[j]=12.34	SR[j]=12.34
R[j]=5.123456789	SR[j]=5.123457

Example R[i]=R[j]

Value of SR[j]	Result of R[i]
SR[j]=1234	R[i]=1234
SR[j]=12.34	R[i]=12.34

SR[i]=(value) (operator) (value)

SR[i]=(value) (operator) (value) instruction is to combine two values and substitute the result of the operation into a string register.

In each operation, the data type depends on the (value) to the left of the (operator).

If the value on the left is string data, the strings are combined with each other.

If the value on the left is numerical data, arithmetic operations are performed. At this point, if the (value) on the right is a string, the numerical value before the first character is used for the operation.

The operator is: + (combine)

Example SR[i]=R[j]+SR[k]

Values of R[j] and R[k]	Result of SR[j]
R[j]=123.456+SR[k]=345.678	SR[j]=456.134

R[j]=456+SR[k]=1abc2	SR[j]=457

Example SR[i]=SR[j]+R[k]

Values of SR[j] and R[k]	Result of SR[j]
SR[j]=123.+R[k]=456	SR[j]=123.456
SR[j]=def+R[k]=81573	SR[i]=def81573

3.4.5 Motion register instruction

A motion register can be understood as a motion variable specific to the motion instruction. It is convenient for debugging the speed of many identical motion instructions. The calculation operation of the MR register is the same as the number register R. The MR register can only be used and appears in motion instructions. The MR register only supports integer type and a decimal (if any) in the calculation assignment is rounded off.

MR[i] = value

There are two data types for values: constant and register (R/PR[i,j]/MI[i]/MH[i]).

Example:

MR[1:]=500

MOVEL P[1:], MR[1:]mm/s, FINE

The robot moves towards P1 in a linear motion at a speed of 500mm/s.

3.4.6 Modbus special register instruction

The Modbus special register is used in Modbus communication to communicate with the Modbus master station. 120 holding registers and 120 input registers are available under standard circumstances.

MH/I[i] =(value), where i is the number of the Modbus special register (1-120).

The value can be:

- Constant
- R[i]: Value of register R[i]
- PR[i, j]: Value of position PR element [i, j]
- MI[i]: Value of input register

- MH[i]: Value of holding register

3.4.7 Palletizing Register Command

The palletizing register command is used to directly read, modify, and calculate the values of palletizing registers. Its specific application methods are as follows.

3.4.7.1 Palletizing Register Command

Only successfully established, configured, and verified palletizing process projects can index specific PL registers.

Assignment forms:

PL[i] = [R, C, L] or PL[i] = PL[j]

3.4.7.2 Palletizing Register Element Command

It is a command used to directly read, modify, and calculate the value of a certain element in the palletizing register. Its specific application methods are as follows.

Command form:

PL[i, R]

Assignment forms:

PL[i, R] = MR[i] or positive integer

Palletizing register elements, whose values are R (row), C (column), L (layer), are positive integers. The maximum value depends on the system support (limited by the row, column, and layer values of the specific palletizing configuration during use, with a value range of [1, 255]).

Examples:

MR[2] = 3

PL[1, R] = MR[2]

PL[1, C] = MR[2]

PL[1, L] = 5

3.4.7.3 Palletizing Registers Used for Condition Judgment

To facilitate users to further refine program logic control according to actual application scenarios and meet production process requirements, palletizing registers

can be used for condition judgment in programs to control palletizing stacking based on palletizing registers:

Example 1:

When the row, column, and layer values are 1, 2, 3, execute...

IF PL[1] = [1,2,3]

...

Example 2:

When it is the 1st row and even layers, execute...

...

ELSE IF PL[1] = [1,2-0,*]

...

In condition judgment, the expression form of elements in [R, C, L] is consistent with the form of "表述 of setting path conditions". Arithmetic operations such as addition, subtraction, multiplication, and division are not allowed, and register operations are not supported. For example, the following situations are not allowed: [2+3, *, *], [2/3, *, *], [R[1], *, *], [MR[1]-1, *, *], etc.

3.5 I/O instruction

The I/O instruction can achieve signal interaction with peripheral devices (e.g. PLCs) and control application processes by reading the input signal (DI/UI/RI) status and changing the output signal (DO/RO) status.

Caution

Before use of I/O signals, it is required to map logical numbers to physical numbers. (For mapping of I/O, see Section 2.1.1)

3.5.1 Digital I/O instruction

Digital input (DI) and digital output (DO) are input/output signals that the user can control.

R[i] = DI[i]

The R[i] = DI[i] instruction is to store the state of the digital input (ON=1, OFF=0) in a register.

Example:

R[1]= DI[1]

R[R[3]]= DI[R[4]]

DO[i] = ON/OFF

The DO[i] = ON/OFF instruction is to turn on or off the specified digital output signal.

Example:

DO[1] = ON

DO[R[3]] = OFF

DO[i] = PULSE, [time]

The DO[i] = PULSE, [time] instruction is to switch on within the specified time and output the specified digital output; the unit of time is hour and second. What is the range of pulse output time?

Example:

DO[2] = PULSE, 0.2 s

DO[R[3]] = PULSE, 1.2 s

DO [i]=R[i]

The DO [i]=R[i] instruction is to turn on or off the specified digital output signal according to the value of the specified register. It is turned off if the value of the register is smaller than or equal to 0 and turned on if it is greater than 0.

Example:

DO［1］= R［2］

DO［R［5］］= R［R［1］］

3.5.2 Group I/O Commands

Group Input (GI) and Group Output (GO) signals group several digital input/output signals, allowing control of these signals with a single command.

R[i] = G[i]

The R[i] = GI[i] command converts the binary value of the specified group input signal to a decimal value and assigns it to the specified register.

Examples are as follows:

R[1] = GI[1]

R[R[3]] = GI[R[4]]

GO[i] = (Value)

The GO[i] = (Value) command outputs the value after binary conversion to the specified group output.

Examples are as follows:

GO[1] = 0

GO[R[3]] = 32767

GO[i] = R[i]

The GO[i] = R[i] command converts the value of the specified register to binary and outputs it to the specified group output.

Examples are as follows:

GO[1] = R[2]

GO[R[5]] = R[R[1]]

3.5.3 Robot I/O instruction

The usage of robot (RO/RI) I/O instruction is the same as that of digital (DO/DI) I/O instruction. Sp, please refer to the usage of digital (DO/DI) I/O instruction.

3.6 Logical instruction

Logical instructions of Agilebot robots include IF, SWITCH and WHILE instructions. The logical instructions can also be understood as conditional control instructions, of which the code block to be executed is determined based on the execution result (True or False) of one or more statements. The execution process of conditional statements can be easily understood through the following figure:

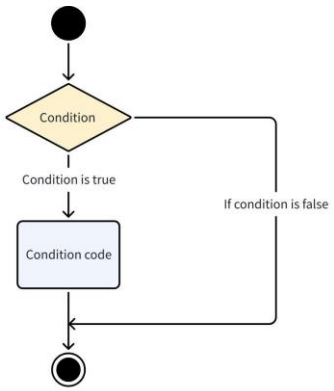


Fig. 3.12 Schematic Diagram of Conditional Statement Execution Process

3.6.1 IF instruction

Instruction format:

IF Condition 1

Statement block 1

ELSE IF Condition 2

Statement block 2

ELSE

Statement block 3

END IF

- Execute the statement block 1 if Condition 1 is true.
- Judge Condition 2 if Condition 1 is false.
- Execute the statement block 2 if Condition 2 is true.
- Execute the statement block 3 if Condition 2 is false.

Condition types judged by IF

- Register type (number register R[i], string register SR[i])
- I/O type (DO/UO/DI/UI)

3.6.2 SWITCH instruction

The SWITCH instruction is usually used to handle the situation where a judgment variable has multiple different values, in order to simplify the program and improve its readability.

Instruction format:

SWITCH subject

CASE pattern_1

Action_1

BREAK

CASE pattern_2

Action_2

BREAK

DEFAULT

Action_3

BREAK

BREAKEND SWITCH

Execute the statement after DEFAULT when all CASEs cannot be matched.

Execute the corresponding Action_x when the values of subject and pattern_x are the same.

Subject can be register (R[*], SR[*]), I/O type, constant or method.

Pattern can be register (R[*], SR[*]), I/O type, constant, method or string.

The BREAK statement is a jump action.

Caution

CASE must be used in conjunction with BREAK.

The example of the Switch logical instruction programming is shown in the following figure:

```
2 SWITCH R[10: default]
3 CASE 1
4   R[11: default] = R[10: default]  +  1
5   BREAK
6 CASE 2
7   R[11: default] = R[10: default]  +  2
8   BREAK
9 END SWITCH
```

Fig. 3.13 Switch Programming Example

3.6.3 WHILE instruction

The WHILE instruction, also known as loop instruction, is generally used to control programs or actions to be executed repeatedly.

Instruction format

WHILE judgment conditions

Execute Statement 1

CONTINUE

Execute Statement 2

BREAK

Execute Statement 3

END WHILE

When the loop instruction WHILE is executing, the robot runs until the judgment condition is not met. Otherwise, after the BREAK statement, it jumps out of the loop instruction and executes the instruction after END WHILE. The difference between Continue statement and Break statement is that the Continue statement only ends the current loop rather than the whole loop; the Break statement ends the whole loop process and does not judge whether the loop conditions are met.

The execution flowchart of WHILE is as follows:

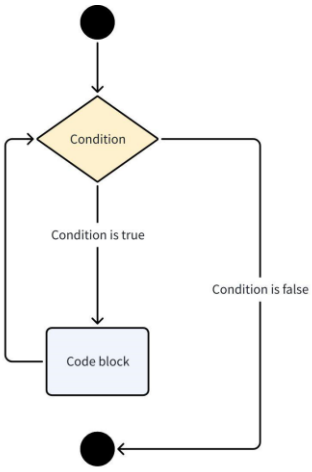


Fig. 3.14 Execution Process of WHILE Statement

Type of judgement condition

- Register type (number register R[i], string register SR[i])
- I/O type (DO/UO/DI/UI)

The example of WHILE instruction programming is shown in the following figure:

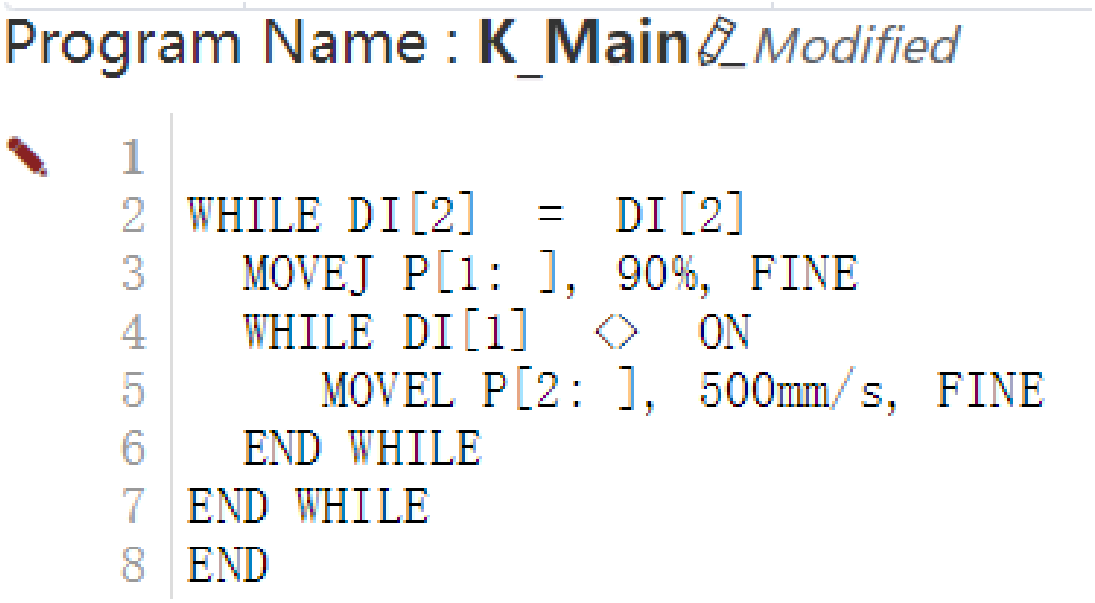


Fig. 3.15 Example of WHILE Instruction Programming

3.6.4 GOTO instruction

The GOTO instruction is used to jump to the specified label position in the same program and continuously execute the program downwards from the specified label position. The GOTO instruction can only be used after a label instruction LABEL has been inserted (see Section 3.8.11).

Instruction format:

GOTO LABEL[i]

Example:

1 MOVEL P[1], 200 mm/s, FINE

2 LABEL[1]

3 MOVEL P[1], 200 mm/s, FINE

4 MOVEJ P[2], 12.5%, FINE

5 GOTO LABEL[1]

6 MOVEJ P[3], 12.5%, FINE

When the program is executed from Line 1 to Line 5 (GOTO LABEL [1]), it may jump to Line 2 (LABEL [1]) and continue executing from Line 2.

3.6.5 SKIP CONDITION instruction

The SKIP CONDITION instruction is to set jump conditions and is often used in conjunction with the additional action instruction SKIP (see Section 3.3.5). It directly jumps from current line containing the SKIP instruction to the target instruction line or program when the jump condition is met.

Instruction format: SKIP CONDITION (I/O type/register type)

Example:

1 SKIP CONDITION DO[2]=ON

2MOVEL P[1], 200 mm/s, FINE, SKIP GOTO LABEL[1]

3 MOVEJ P[2], 12.5%, FINE

4 LABEL[1]

5 MOVEJ P[3], 12.5%, FINE

The program starts executing from the first line. When DO[2]=ON, the program jumps to the fourth line after the second line ends and continuously executes downwards from the fourth line. When DO[2]=OFF, the program continuously executes downwards from the current line after the second line ends.

3.7 Structure instructions

The structure instructions are used to control the program execution process, including CALL, WAIT, WAIT TIME, PAUSE, ABORT, RUN, LOAD, UNLOAD and EXEC instructions.

3.7.1 CALL - Call program instruction

Instruction format: Call + program name. It is used to call a subprogram, form nesting, improve program reusability and readability and reduce programming workload. For example, in the process of robot application, it is required to control the movement of the end fixture at different positions.

As shown in the following figure, the program initialization calls the subprogram calibration_fou_point.

Program Name : K_Main Modified

```
1
2 CALL calibration_four_point
3 WHILE DI[2]  =  DI[2]
4   MOVEJ P[1: ], 90%, FINE
5   WHILE DI[1]  <>  ON
6     MOVEL P[2: ], 500mm/s, FINE
7   END WHILE
8 END WHILE
9 END
```

Fig. 3.16 Example of Call Instruction Programming

3.7.2 WAIT - Waiting for conditions to meet

Instruction format: WAIT + Condition + TIMEOUT + SKIP/Waring/Called Program

Description of parameters:

The available data types for conditions include I/O type (DI/DO/UI/UO) and register type (R[i]/ MI[i]/MH[i]).

TIMEOUT:

Optional parameters (not-required parameters). Its meaning is the waiting time (s). It can be specified by a constant or number register.

When the WAIT instruction does not use the additional condition TIMEOUT: Execute the WAIT instruction. Only when the conditions are met can the program continue to execute downwards. Otherwise, it will wait until the conditions are met.

When the WAIT instruction uses the additional condition TIMROUT: Execute the WAIT instruction. The program continues to execute downwards if the condition is met within the specified waiting time or pauses at the WAIT instruction if the condition is not met within the specified waiting time.

SKIP/Waring/Called Program:

Optional parameters (not-required parameters). In case the optional parameter TIMEOUT is used, one of the optional parameters can be selected from three parameters SKIP/Waring/Called Program.

If the SKIP parameter is used:

The program continues to execute downwards if the condition is met within the specified waiting time or skips the current line (WAIT instruction line) and continues to execute downwards if the condition is not met within the specified waiting time.

If Waring is used:

Execute the WAIT instruction. The program continues to execute downwards if the condition is met within the specified waiting time or pauses at the WAIT instruction if the condition is not met within the specified waiting time.

Execute the WAIT instruction if the called program is used. The program continues to execute downwards if the condition is met within the specified waiting time or executes the called program and then continues to execute downwards if the condition is not met within the specified waiting time.

The following figure is an example of the wait instruction programming.

```
7  WAIT R[1: default]  =  1
8  WAIT R[1: default]  =  1 TIMEOUT=2sec
9  WAIT R[1: default]  =  1 TIMEOUT=1sec SKIP
10 WAIT R[1: default]  =  1 TIMEOUT=1sec Warning
11 WAIT R[1: default]  =  1 TIMEOUT=1sec A
12 END
```

Fig. 3.17 Example of WAIT Instruction Programming

3.7.3 WAIT TIME - waiting time instruction

Instruction format:

WAIT TIME + waiting time (s).

WAIT TIME can be specified by a constant or number register.

Before the waiting time ends, the robot stops executing instructions until the waiting time ends. The waiting time can be a numerical value or a register.

The programming example is shown in the following figure.

Program Name : **K_Main** *Modified*

```
1  CALL calibration_four_point
2  WHILE DI[2]  =  DI[2]
3    WAIT 0.5 sec
4    MOVEJ P[1: ], 90%, FINE
5    WHILE DI[1]  <>  ON
6      WAIT 1 sec
7      MOVEL P[2: ], 500mm/s, FINE
8    END WHILE
9  END WHILE
10 END
```

Fig. 3.18 Example of WAIT Instruction Programming

3.7.4 PAUSE - Pause instruction

Instruction format: Pause

When the Pause instruction is encountered, the program is paused and the robot immediately plans a deceleration trajectory and stops its motion. The program state enters a paused state. Press the Start button again to continue execution.

The following figure is an example of the PAUSE instruction programming.

Program Name : **K_Main**

```
1  CALL calibration_four_point
2  WHILE DI[2]  =  DI[2]
3    IF DI[3]  =  ON
4      PAUSE
5    END IF
6    MOVEJ P[1: ], 90%, FINE
7    WHILE DI[1]  <>  ON
8      MOVEL P[2: ], 500mm/s, FINE
9    END WHILE
10 END WHILE
11 END
```

Fig. 3.19 Example of PAUSE Instruction Programming

3.7.5 ABORT - Abort instruction

Format: ABORT

Mandatory end instruction: End program execution, immediately brake the servo motor of the robot and apply the holding brake, and power off the servo bus. After the mandatory end instruction, the program enters a termination state and the cursor stops at the current line.

The following figure is an example of the ABORT instruction programming.

Program Name : K_Main Modified

```
1  CALL calibration_four_point
2  WHILE DI[2]  =  DI[2]
3    IF DI[3]  =  ON
4      ABORT
5    END IF
6    MOVEJ P[1: ], 90%, FINE
7    WHILE DI[1]  <>  ON
8      MOVEL P[2: ], 500mm/s, FINE
9    END WHILE
10 END WHILE
11 END
```

Fig. 3.20 Example of ABORT Instruction Programming

3.7.6 RUN - Multithread instruction

RUN instruction can be used to run multiple programs simultaneously.

Format: RUN + program name

Program type: User program (XML and JSON), Script program (PYTHON)

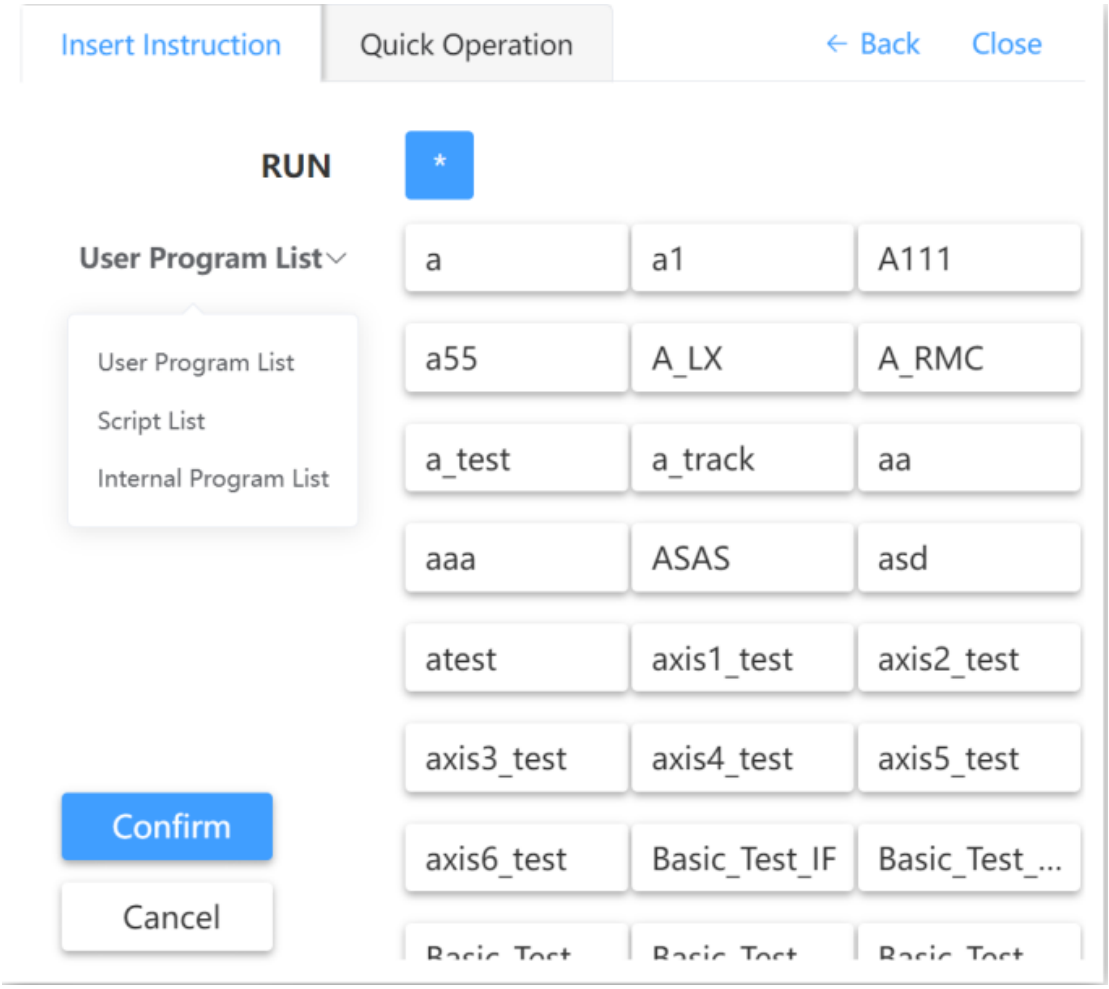


Fig. 3.20 Example of RUN Instruction Insertion

Caution

Notes:

- If the RUN instruction is used in the Main program to call the Pick program, Main is called the "caller", Pick is called the "callee", and the one called by the "callee" is also called the "callee" (the same below).
- The CALL instruction only executes a program at a time. However, the RUN instruction allows multiple programs to be executed simultaneously. It does not wait for the completion of the "callee" before executing the "caller".
- When the same program is called by RUN for multiple times, subsequent runs may not be responded to and there is an INFO event if the first execution is not completed.
- Only the caller can write motion instructions. If the callee has motion instructions,

an alarm may be sent out with the alarm code "program-2328".

Reset button:

- For the caller program, the callee program will also be reset.

Pause/abort:

- If the caller is paused/aborted, all callees may be paused/aborted as well.

Single step & positive sequence:

- Single step is only available when all programs are paused or aborted.
- When the caller is executing in single step & positive sequence, the callee, after being executed, may follow the caller to perform in single step & positive sequence line by line.
- If single step & positive sequence is adopted in the callee, only the callee is executing in single step & positive sequence line by line, but the caller will not follow it.

Single step & reverse sequence:

- Ignore the RUN instruction and do not execute it.

Status bar:

- Program name and line number: The running program (if any) is displayed; otherwise, the last program run or opened is displayed.
- Robot's operation status:

If the caller is still present, the status depends on the caller. If not, the status is WORKING if one of the callees is running or ON-STANDBY otherwise.

UI:

- Start/Restore: Control all programs.
- Stop: Control all programs.
- Abort: Control all programs.

UO:

- Pause status: similar to the description of robot's operation status.
- Program Running: similar to the description of robot's operation status.

3.7.7 Load - Dynamic program loading

Definition: Dynamically loading programs

Format: Load + program (e.g. "pick")

In this instruction, the program may specify register types (R [i]/SR [i]), strings and constants.

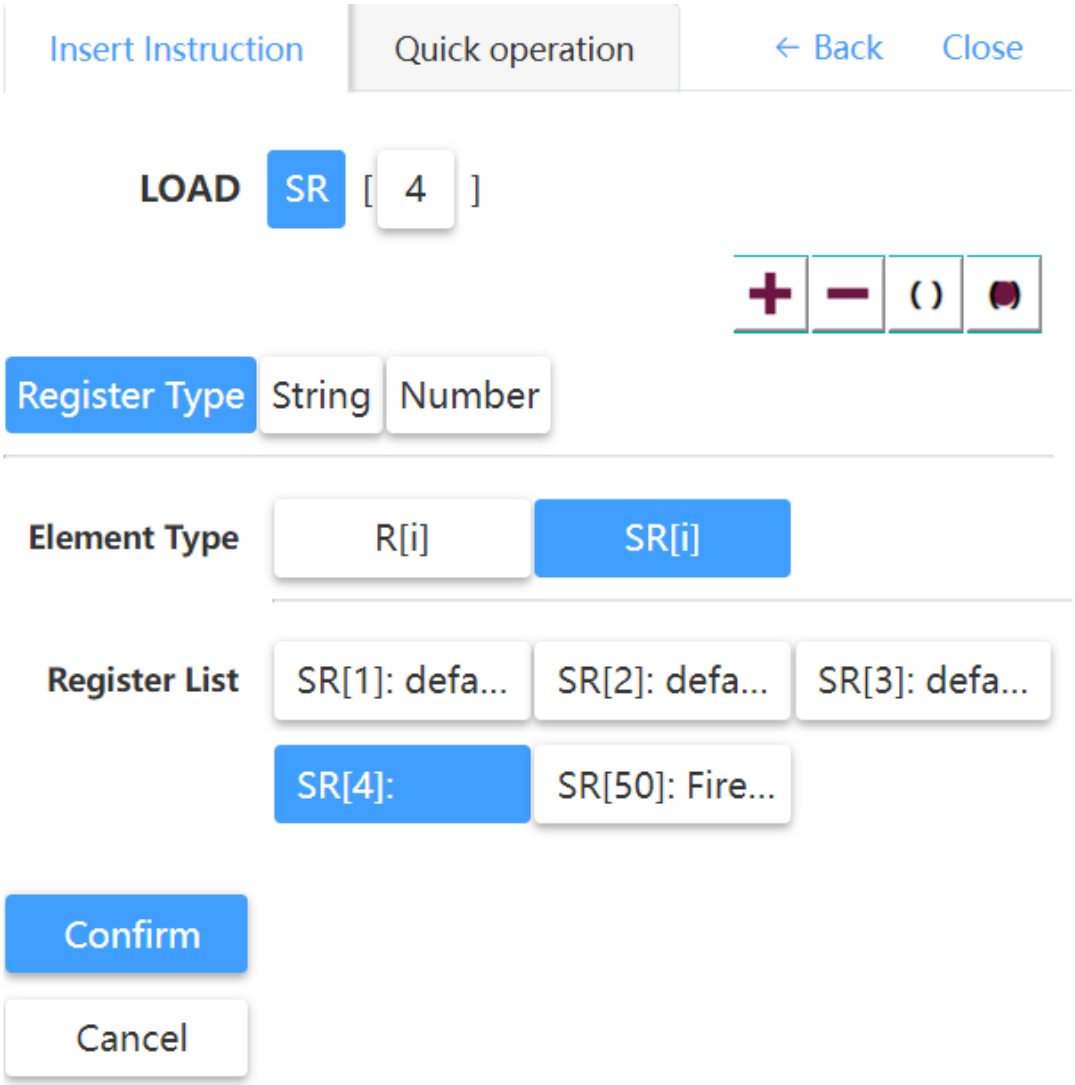


Fig. 3.21 Example of Load Instruction Insertion

3.7.8 Exec - Execute dynamic program loading

Definition: Execute programs

Format: Exec + program (e.g. "pick")

In this instruction, the program may specify register types (R [i]/SR [i]), strings and constants.

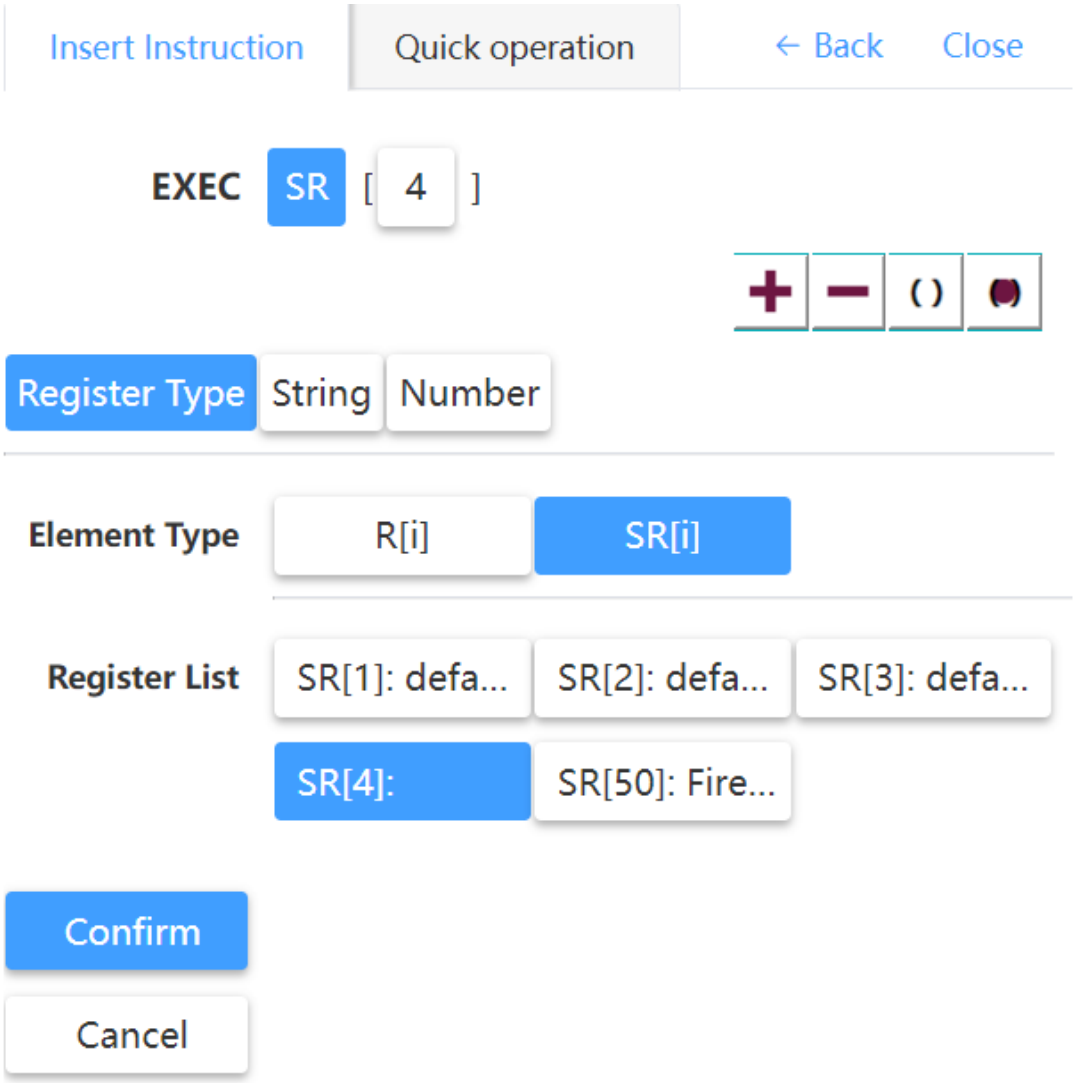


Fig. 3.22 Example of Exec Instruction Insertion

3.7.9 Unload - Dynamic program unload

Definition: Unload programs

Format: Unload + program (e.g. "pick")

In this instruction, the program may specify register types (R [i]/SR [i]), strings and constants.

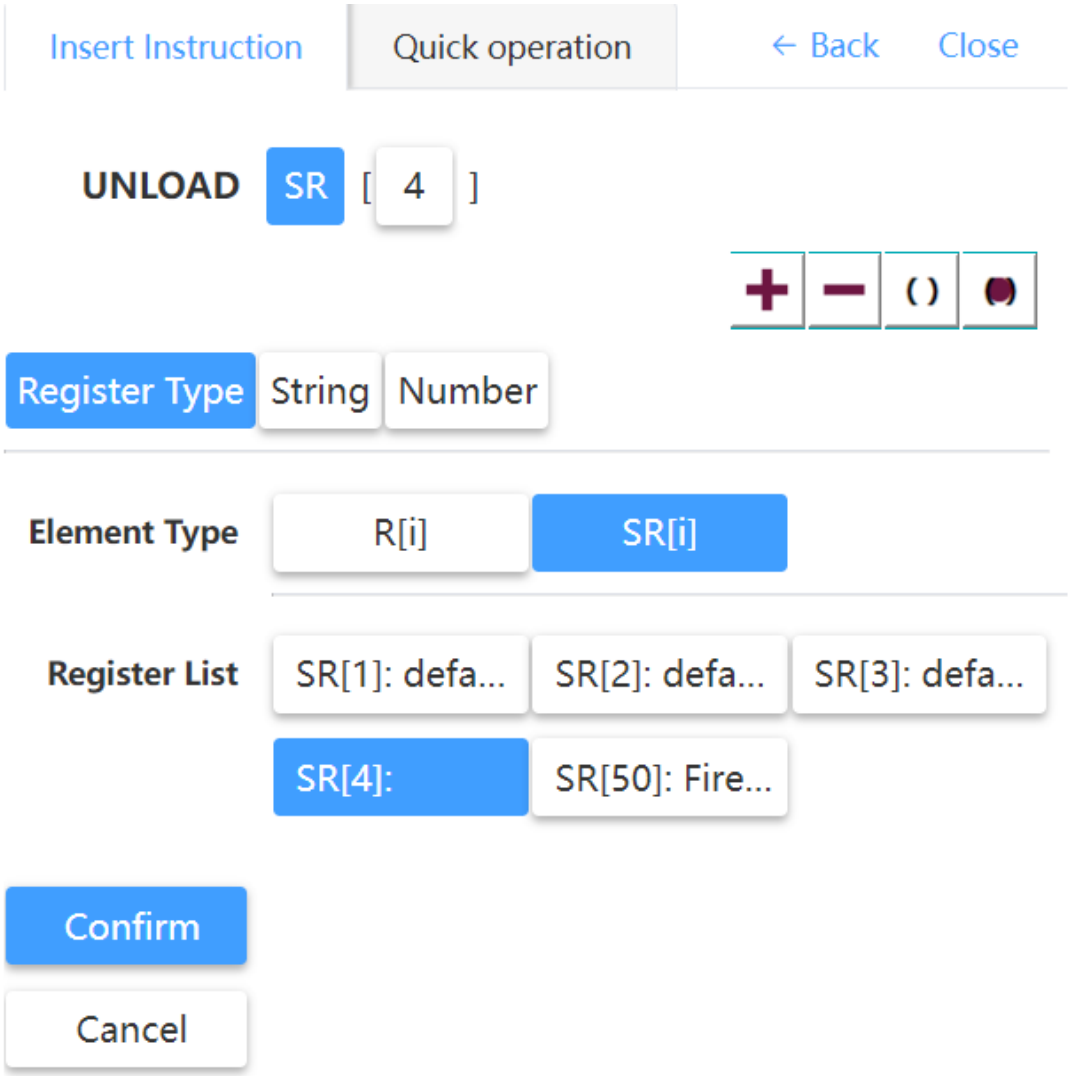


Fig. 3.23 Example of Unload Instruction Insertion

3.8 Other instructions

3.8.1 TF_NO - Tool coordinate system instruction

It is to switch tool coordinate systems in programs and is generally used in conjunction with UF_NO.

Instruction format: TF_No = tool coordinate system number; the data type of the "Tool Coordinate System Number" can be selected from two types: register type (R) and constant.


Caution

Due to no UF/TF in the PR register, it represents the pose of the specified TF under the currently activated UF during program execution. So, switching between different UF/TF may result in different positions of the actual robot in space.

P represents local pose data, including UF/TF. If P is used in the motion instruction but inconsistent with UF_No and TF_No, the program cannot continuously execute downwards and it is required to modify UF_No and TF_No to make them consistent with P record.

3.8.2 UF_NO user coordinate system instruction

It is to switch user coordinate systems in programs and is generally used in conjunction with TF_NO.

Instruction format: TF_No = user coordinate system number; the data type of the "User Coordinate System Number" can be selected from two types: register type (R) and constant.


Caution

Due to no UF/TF in the PR register, it represents the pose of the specified TF under the currently activated UF during program execution. So, switching between different UF/TF may result in different positions of the actual robot in space.

P represents local pose data, including UF/TF. If P is used in the motion instruction but inconsistent with UF_No and TF_No, the program cannot continuously execute downwards and it is required to modify UF_No and TF_No to make them consistent with P record.

3.8.3 J_POS current joint coordinate instruction

It indicates the current position in the joint coordinate system and is often used in conjunction with the pose register PR.

Example

PR[i] = J_POS Assign current position in the robot's joint coordinate system to the

position register PR [i].

3.8.4 L_POS current Cartesian coordinate instruction

It indicates the current position in the Cartesian coordinate system and is often used in conjunction with the pose register PR.

Example

PR[i] = J_POS Assign current position in the robot's joint coordinate system to the position register PR [i].

3.8.5 Payload_NO - payload setting instruction

It is used to switch the payload number in the program. For example, during the application, only the payload of the fixture is left at the end of the robot arm before the workpiece is grasped, but a combined payload of the fixture and the workpiece is on the end of the robot arm after the workpiece is grasped. Then, the payload data before and after grasping are recorded with different payload numbers.

Instruction format: PAYLOAD_NO = load record number; the data type of "Payload Record Number" can be selected from two types: register type (R) and constant.

3.8.6 Timer - timer instruction

Program Timer instruction: it is used to start or stop a program timer. The start or stop of the timer can be controlled by its status. The global timer supports multi-program (task) sharing.

Instruction format

a. Timer[i] = value There are four data types of values, namely status (Start/Stop/Reset), register (R), constant and method (please refer to Section 3.10).

When the value is the status

Timer[i]=Start; the time starts.

Timer[i]=Stop; the time stops.

Timer[i]=Reset; the timer resets (value resetting)

In [i],i represents the timer number, which can be specified directly by a constant or indirectly by the number register R.

b. Assignment: R[i]=Timer [i] (provided that Timer must have a defined state; if not, Timer is null and cannot be assigned).

3.8.7 Comment - Commend instruction

The comment instruction is used to add comments to a program to improve its readability, and this comment has no impact on program operation.

Format: #(Comment text)

The comment text supports all data types.

3.8.8 OVC - Overall velocity instruction

It is used to modify the overall velocity.

Instruction format: Overall_Velocity_Coefficient = velocity percentage。 There are two data types for “velocity percentage”: number register type (R) and constant.

3.8.9 OAC - Overall acceleration instruction

It is used to modify the overall acceleration.

Instruction format: Overall_Acceleration_Coefficient = acceleration percentage. There are two data types for “acceleration percentage”: number register type (R) and constant.

3.8.10 LABEL - Label Instruction

Defines program labels, often used in conjunction with the GOTO instruction (see Section 3.6.4) and the SKIP CONDITION instruction (see Section 3.6.5).

Format: LABEL[i : Comment]; where i is the label number, and Comment is the label comment information.

3.8.11 Socket - Socket Instruction

Instruction Overview:

SocketOpen: Use the SocketOpen instruction to create a Server and wait for Client connections.

SOCKET OPEN SK[*], parameter

The parameter SK[*] represents the currently used socket device, and the optional parameter indicates the execution result of the statement (see Chapter 10 Socket Error Code List), expressed by a numerical register (R[*]).

Supplementary Notes:

- SocketOpen is a prerequisite for using other socket commands. Please call the SocketOpen instruction first before performing subsequent socket operations.

SocketConnect: Use the SocketConnect instruction to create a Client and connect to a Server.

SOCKET CONNECT SK[*], parameter

The parameter SK[*] represents the currently used socket device, and the optional parameter indicates the execution result of the statement (see Socket Error Code List), expressed by a numerical register (R[*]).

SocketRecv: The SocketRecv instruction is used to read characters from the Socket, and the maximum length can be specified.

SOCKET RECV SK[*], R[*], SR[*], parameter2

The parameter SK[*] represents the currently used socket device, the parameter R[*] represents the receiving length; SR[*] represents the received data stored in the string register (SR[*]), and the optional parameter2 indicates the execution result of the statement (see Socket Error Code List), expressed by a numerical register (R[*]).

Supplementary Notes:

- Both Server and Client can call this instruction.
- The length of the received string: the maximum value is 254.
- The "default receive timeout" parameter configured in the SK register in the instruction is valid for this instruction.

SocketSend: The SocketSend instruction is used to write a string to the Socket for sending to the opposite end.

SOCKET SEND SK[*], "", parameter2

The parameter SK[*] represents the currently used socket device, the parameter "" represents the content to be sent (string), which can be a constant or a string register (SR[*]), and the optional parameter2 indicates the execution result of the statement (see Socket Error Code List), expressed by a numerical register (R[*]).

Supplementary Notes:

- Both Server and Client can call this instruction.
- If the string to be sent is represented by SR, the maximum length is 254.
- You can directly send a constant string, or store the string in a string register and send the content by selecting the register.

SocketClose: The SocketClose instruction is used to close the connection.

SOCKET CLOSE SK[*], parameter

The parameter SK[*] represents the currently used socket device, and the optional parameter indicates the execution result of the statement (see Socket Error Code

List), expressed by a numerical register (R[*]).

Caution

Optional parameters can be used or not, depending on actual needs. Parameters other than optional ones are mandatory.

3.8.12 Compound Operation Instruction

The compound operation instruction can combine various operators and data in assignment statements, conditional comparison statements, and wait instruction statements of TP programs. The compound operation instruction supports parentheses "()".

The compound operation instruction can be used in register instructions, IF instructions, WAIT instructions, and WHILE instructions.

The compound operation instruction is specified in parentheses as shown below.

- IF DI[1] = (DO[2] AND DO[3])

Execution statement

END IF

When there are no parentheses in the sentence, it becomes a normal operation instruction.

- WAIT (DO[1] = ON AND D0[2] = OFF) OR (R[1] = 1 OR R[2] = 2)

Data Types

The compound operation instruction can use the following data types.

Type	Value	Data
Numerical	Can handle numerical values as data. Both integers and real numbers can be used.	Registers, constants, elements of position registers
Boolean	Data can be set to either ON or OFF	DI/O, UI/O, ON, OFF

Operators

The compound operation instruction can use the following operators.

Operator	Operation
+	Addition operation of the left and right sides
-	Subtraction operation of the left and right sides
*	Multiplication operation of the left and right sides
/	Division operation of the left and right sides
MOD	Remainder of the division operation of the left and right sides
DIV	Integer part of the quotient of the division operation of the left and right sides

- Arithmetic operators can only use numerical type data.
- The output data of arithmetic operators is always numerical.

Operator	Operation
AND	Logical AND of the left and right sides
OR	Logical OR of the left and right sides

- Logical operators are only used for boolean data.
- The output data of logical operators is always boolean.

Operator	Operation
=	Returns ON when the left and right sides are equal. Returns OFF when the left and right sides are not equal.
<>	Returns ON when the left and right sides are not equal. Returns OFF when the left and right sides are equal.

<	Returns ON when the left side is less than the right side. Returns OFF when the left side is greater than the right side.
>	Returns ON when the left side is greater than the right side. Returns OFF when the left side is less than the right side.
<=	Returns ON when the left side is less than or equal to the right side. Returns OFF when the left side is greater than the right side.
>=	Returns ON when the left side is greater than or equal to the right side. Returns OFF when the left side is less than the right side.

- "=" and "<>" can be used in both numerical data and boolean data.
- "<", ">", "<=", and ">=" can only be used in numerical data.

The following table shows the priority order of operators.

Priority	Operators
High	*, /, DIV, MOD
	+, -
Medium	<, >, <=, >=
	=, <>
Low	AND
	OR

Conditional Statements

The following shows examples of using compound operation instructions in conditional instructions.

IF (R[1] = (PR[1,6] + R[1]) * R[2])

IF (DI[1] AND (DI[2] OR DI[3]))

- Compound expressions can be used in conditional statements.
- The result of a conditional statement must be boolean.

Wait Instructions

The following shows examples of using compound operation instructions in wait instructions.

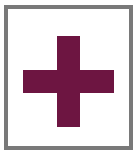
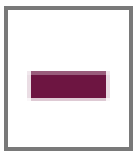
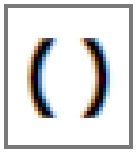
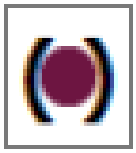
WAIT (DI[1] = ON AND (DI[2] = ON OR DI[3] = ON))

- Compound expressions can be specified in the conditional statements of wait instructions.
- The result of the conditional statement must be boolean.
- The wait instruction waits until the result of the expression becomes ON.

Addition and Modification of Compound Operation Instructions

For instructions with compound operation functions, the symbol description of the instruction editing interface is used, and the following table symbols are used for adding and modifying compound operation instructions.

Symbol Table for Addition and Modification of Compound Operation Instructions

	represents adding an expression
	represents removing an expression (there must be at least one, that is, there must be at least one item on the right side of the equation)
	represents adding parentheses
	represents removing parentheses

3.9 String instruction

3.9.1 StrLen - String length instruction

It is used to obtain the length of a string and bring the result into a register. It is commonly used as R[i]=StrLen (value).

The data type of the value is usually a constant string or a string register (SR).

Example:

If R[i] = StrLen ("666"), the value of R[i] is 3.

If R[i] = StrLen (SR[i]), where SR[i]=111111, the value of R[i] is 6.

3.9.2 FindStr - String search instruction

R [i]=FindStr (Value 1, Value 2); the data type of Value 1 and Value 2 is usually a constant string or a string register (SR).

Value 1 represents the "object string", Value 2 represents the "string to search for", and R[i] represents the position of the string to search for in the object string. From left to right, the position of the first character is 0 in the object string, and so on. When the searched string is not found, the value of R[i] is -1.

Example:

R[i] = FindStr ("123456", "1") The value of R[i] is 0.

R[i] = FindStr ("123456", "0") The value of R[i] is -1.

3.9.3 SubStr - String subtraction instruction

SR[i]=SubStr (Value 1, Value 2, Value 3).

Value 1 represents the "object string", Value 2 represents the "start point position", and Value 3 represents the "string length to be subtracted". Among them, the data type of Value 1 is usually a constant string or a string register (SR), and the data types for Values 2 and 3 are usually constants or number registers (R). SR [i] represents the subtracted string.

SR[i]=SubStr (Value 1, Value 2, Value 3) instruction is to subtract a partial string from an object string and substitute its result into a string register. The subtracted string is determined according to the start point position of the first character of the object value and the string length to be subtracted.

Example:

SR[i]=SubStr ("P123456", 0, 1) The value of SR[i] is "P".

3.10 Methods (functions)

3.10.1 SIN - Sine function

Sin (Sine) is used to calculate the sine value of an angle.

Basic example

The following example describes the function Sin.

R[1:]=Sin(*)

R [1:] is to obtain the sine value of *. Range of sine values = (-1, 1).

3.10.2 COS - Cosine function

Cos (Cosine) is to calculate the cosine value based on the angle of relevant data type “num”.

Basic example

The following example describes the function Cos.

R[1:]=Cos(*)

R[1:] is to obtain the cosine value of *. Range of cosine values = (-1, 1).

3.10.3 TAN - Tangent function

Tan (Tangent) is used to calculate the tangent value of an angle.

Basic example

The following example describes the function Tan.

R[1:]=Tan（*）

R [1:] is to obtain the tangent value of *.

3.10.4 ARCSIN - Anti-sine function

ArcSin is used to calculate the anti-sine value of an angle.

Basic example

The following example describes the function ArcSin.

R[1:]= ArcSin(0.5)

R [1:] is to obtain the ArcSin value (30) of 0.5.

3.10.5 ARCCOS - Arccosine function

ArcCos is used to calculate the arccosine value of an angle.

Basic example

The following example describes the function ArcCos.

R[1:]= ArcCos(-0.5)

R [1:] is to obtain the ArcCos value (120) of -0.5.

3.10.6 ARCTAN - Arctangent function

ArcTan is used to calculate the arctangent value of an angle.

Basic example

The following example describes the function ArcTan.

R[1:]= ArcTan(1)

R [1:] is to obtain the ArcTan value (45) of 1.

3.10.7 ABS - Absolute value function

Abs is to obtain absolute values, namely positive values of digital data.

Basic example

The following example describes the function Abs.

R[1:]=Abs(R[2:])

Specify R[1:] as the absolute value of R[2:].

3.11 CollisionDetect Command

3.11.1 CollisionDetect

It is used to set the collision detection ON/OFF.

Instruction format: CollisionDetect ON/OFF, GROUP *, AXIS *; where, GROUP and AXIS parameters are optional.

Optional parameter GROUP *: Motion group; the parameter of the robot motion group is 1 (currently, it does not support external axis configuration).

Optional parameter AXIS *: Axis, which represents one of Axes 1-6 of the robot when the motion group is 1 (select the axis according to the actual situation).

When the optional parameter is not used, it indicates that collision detection is turned on/off for all motion groups.

Basic example

The following example describes the CollisionDetect instruction.

CollisionDetect ON, GROUP 1, AXIS 1; It indicates the collision detection of Axis 1 of the robot.

CollisionDetect ON; It indicates the collision detection of all axes of the robot.

3.11.2 CollisionRange

It is used to set the collision detection deviation. The greater the deviation value of collision detection, the more sensitive the robot's collision detection; vice versa.

Instruction format: CollisionRange * %, GROUP *, AXIS*; where, GROUP and AXIS parameters are optional.

Optional parameter GROUP *: Motion group; the parameter of the robot motion group is 1 (currently, it does not support external axis configuration).

Optional parameter AXIS *: Axis, which represents one of Axes 1-6 of the robot when the motion group is 1 (select the axis according to the actual situation).

When the optional parameter is not used, it indicates that the collision detection sensitivity of all motion groups is *%.

Basic example

The following example describes the CollisionRange instruction.

CollisionRange 30%, GROUP 1, AXIS 1; It indicates that the collision detection sensitivity of the robot's Axis 1 is 30% of maximum sensitivity.

CollisionRanget 30%; It indicates that the collision detection sensitivity of all robot axes is 30% of maximum sensitivity.

3.12 Palletizing Instructions

Palletizing instructions are a general term for a collection of instructions and their accompanying information, including:

Instruction	Description
Palletizing Format Instruction	Calculates the position of the current palletizing point based on the value of the palletizing register and the palletizing mode, matches the corresponding path according to path conditions, and rewrites the position data of the palletizing action instruction.

Palletizing Action Instruction	Action instructions that use approach points/pick-and-place points/exit points as position data.
Palletizing End Instruction	Calculates the next stacking point and rewrites the value of the palletizing register.

The three instructions must exist simultaneously in the same program (main program) to take effect. If any of the instructions is placed in a program called by CALL or RUN, it will be ignored and cannot be executed. The assignment operation of the PL register must be completed before the palletizing format instruction; otherwise, problems such as collision between the robot and the stack or damage to the workpiece due to excessively high position when releasing the gripper may occur. Other instructions can be inserted arbitrarily between the palletizing format instruction and the section number instruction without affecting the execution of the palletizing function.

3.12.1 Palletizing Format Instruction

Both the palletizing format instruction and the end instruction are used to identify the effective range of palletizing instructions, and they exist in pairs. The palletizing format instruction only displays parameters describing the characteristics of the pallet type, which is convenient for reading and cannot be modified.

Instruction Form:

PALLETIZING[i: NAME], B, LOAD, R, C, L

Parameter	Description
B	Types of stack forms: B, E.
i	Palletizing instruction ID, range: [1,30] (integer).
NAME	Palletizing name, inherited from the name of the specific palletizing process in "Menu - Application - Palletizing".
LOAD	Palletizing/unloading identifier: value "LOAD" means palletizing; "UNLOAD" means unloading.
R, C, L	Values of rows, columns, and layers, N ∈ [1,255].

Example:

PALLETIZING [1: Test], B, UNLOAD, 3, 4, 5

It represents a type B stack with 3 rows, 4 columns, and 5 layers, with ID 1, name "Test", and mode as type B unloading palletizing process.

3.12.2 Approach Point/Pick-and-Place Point/Exit Point Action Instructions

These are used to record the point data obtained through teaching. These points will be offset according to the taught initial stacking posture, and the offset vector is determined by the RCL self-increment or self-decrement rules and the generated stacking points. In the case of multiple path styles, the path style of the [R,C,L] point uses the path style that meets the matching path conditions. Palletizing action instructions only support MOVEJ (default) and MOVEL.

The approach point, pick-and-place point, and exit point instructions cannot exist independently outside the effective range of the palletizing format instruction and the palletizing end instruction.

Instruction Form:

PAL [i, A_x]

PAL [i, BTM]

PAL [i, R_x]

Parameter	Description
PAL […]	Palletizing-specific posture register (similar to P in effect).
i	Bound to a specific palletizing process, range: N ∈ [1,30].
A_x	Approach point ID, where "A" is a fixed prefix and "x" is a positive integer. The number of points is the same as the number of approach points set in the corresponding process.
BTM	Stacking point, only one per path point.
R_x	Exit point ID, where "R" is a fixed prefix and "x" is a positive integer. The number of points is the same as the number of

	exit points set in the corresponding process.

Example:

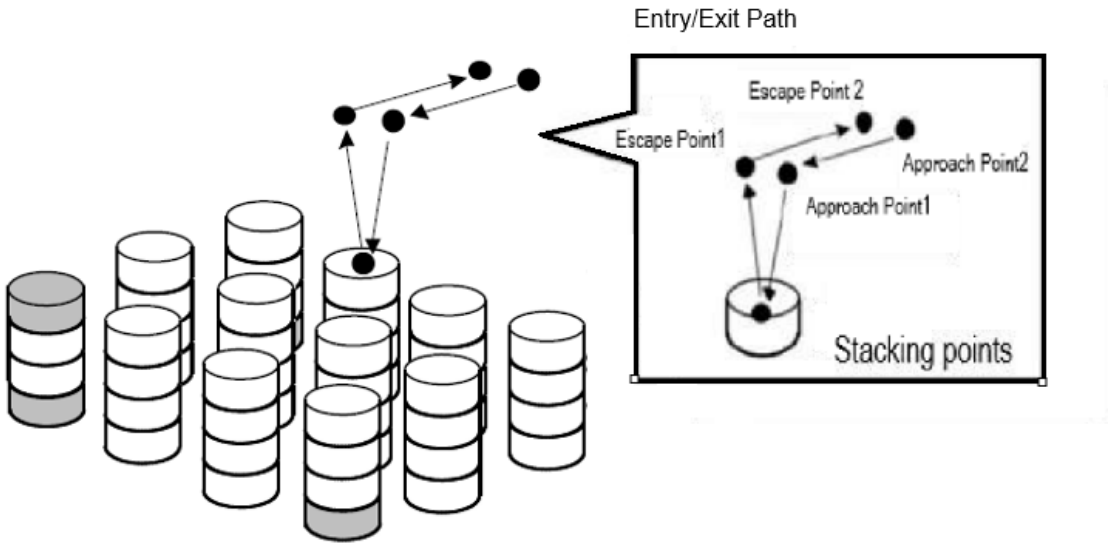
MOVEJ PAL[1，A_2]，90%，SD

MOVEL PAL[1，A_1]，500mm/s，SD

MOVEL PAL[1，BTM]，500mm/s，FINE

MOVEL PAL[1，R_1]，500mm/s，FINE

MOVEL PAL[1，R_2]，500mm/s，FINE



Path composed of approach point, stacking point, and exit point

It means moving to the obstacle-free approach position first, then to the stacking point, releasing the workpiece after arrival, and finally moving to the obstacle-free exit point, from which the robot can return safely.

Caution

In actual use, points can be skipped according to the actual situation. For example, for a path with 2 approach points and 3 exit points, [0,2] approach points and [0,3] exit points can be used. For example:

MOVEJ PAL[1，A_1]，500mm/s，SD

MOVEL PAL[1，BTM]，500mm/s，FINE

MOVEL PAL[1，R_1]，500mm/s，SD

MOVEL PAL[1，R_3]，500mm/s，FINE

3.12.3 Palletizing End Instruction

It is used to identify the effective range of the palletizing instruction and exists in pairs with the palletizing format instruction. At the same time, when the program runs to this instruction, the value of the palletizing register will be updated according to the self-increment (self-decrement) rules configured in the palletizing format instruction. The attributes of this instruction are subordinate to the palletizing format instruction and cannot be configured independently.

Instruction Form:

PALLETIZING_END [i: NAME]

Parameter	Description
i	Palletizing instruction ID, automatically set when configuring the palletizing format instruction, cannot be changed independently.
NAME	Palletizing instruction name, inherited from the name of the specific palletizing process in "Menu - Application - Palletizing".

Example:

PL[1] = [2,2,2]

PALLETIZING [1: Test]，B，LOAD，3，4，5

MOVEJ PAL[1，A_1]，90%，SD

MOVEL PAL[1，BTM]，500mm/s，FINE

MOVEL PAL[1，R_1]，500mm/s，FINE

PALLETIZING_END [1: Test]

It indicates that the palletizing order is row -> column -> layer, in positive sequence, with a step value of 1. For rows 1-5, the value of PL[1] is [2,2,2]; when row 6 ends,

the value of PL[1] is [3,2,2].

Note:

Multiple palletizing processes can be used in the same program and nested according to the program. For example:

PALLETIZING [1: Test]，B，LOAD，2，10，5

MOVEJ PAL[1，A_1]，90%，SD

MOVEL PAL[1，BTM]，500mm/s，FINE

MOVEL PAL[1，R_1]，500mm/s，FINE

……

PALLETIZING [2: Test]，B，LOAD，3，10，5

MOVEJ PAL[2，A_2]，90%，SD

MOVEJ PAL[2，A_1]，90%，SD

MOVEL PAL[2，BTM]，500mm/s，FINE

MOVEL PAL[2，R_1]，500mm/s，FINE

MOVEL PAL[2，R_2]，500mm/s，FINE

……

PALLETIZING_END [1: Test]

PALLETIZING_END [2: Test]

3.13 Vision Instructions

To use vision instructions in the TP program, you need to modify the IP address of the vision controller to ensure that the IP address of the vision controller (PC) and the IP address of the TP (robot control cabinet) are in the same local area network segment.

	TP (Robot Control Cabinet)	Vision Controller (PC)

IP Address	192.168.110.2	192.168.110.3
Subnet Mask	255.255.255.0	255.255.255.0

3.13.1 VISION RUN_FIND Instruction to Start a Vision Program

Used to start a vision program.

- Soft trigger instruction format: VISION RUN_FIND + 'vision program name'
- Hard trigger instruction format: VISION RUN_FIND

3.13.2 VISION GET_QUANTITY Instruction to Obtain the Number of Vision Detections

Used to obtain the number of vision detections.

Instruction format: VISION GET_QUANTITY + 'vision program name' + R[i]

Example: VISION GET_QUANTITY test, R[1:]

The number of detections by the vision program "test" is assigned to the numerical register R[1].

3.13.3 MODELID Instruction to Obtain Template ID

This instruction assigns the template ID of the vision register to the numerical register R[i].

Instruction format: R[i] = VR[I, MODELID]

Example: R[1] = VR[1, MODELID]

3.13.4 GET_OFFSET Instruction to Obtain Detection Position

Used to obtain detection results from the vision program and store them in the specified vision register.

Instruction format: VISION GET_OFFSET + 'vision program name' + VR[i] + GOTO LABEL[i]

Description:

- If the vision program has not finished processing, it waits until image processing is completed.
- If the vision program does not detect anything, it jumps to the specified label in the program according to GOTO.
- This instruction can be executed repeatedly when the vision program detects multiple results.

Example: VISION GET_OFFSET test, VR[1], GOTO LABEL [1]

3.13.5 VOFFSET Vision Position Compensation Instruction

This instruction compensates the robot's position with data stored in the vision register.

Instruction format: …, VOFFSET, VR[i]

Example: MOVEL PR[1:], 500mm/s, FINE, VOFFSET VR[1]

3.13.6 Instruction to Assign Detected Position Information

Assigns the position information of the detected vision register VR[i] to the posture register PR[i].

Instruction format: PR[i] = VR[i]

3.13.7 Examples of Vision Programs

Example 1: Detecting multiple identical workpieces in one photo

Label[1]

VISION RUN_FIND test

VISION GET_QUANTITY test, R[1:]

IF R[1] = 0, GOTO LABEL [1]

While R[1] > 0,

VISION GET_OFFSET test, VR[1], GOTO LABEL [1]

MOVEL PR[1:], 500mm/s, FINE, VOFFSET VR[1] R[1] = R[1] - 1

End

Example 2: Detecting different workpieces in a photo

```
Label[1]
VISION RUN_FIND  test
VISION GET_OFFSET  test，  VR[1]，  GOTO LABEL [1]
R[1] = VR[1，  MODELID]
Switch R[1]:
Case 1:   MOVEL PR[1]，  500mm/s，  FINE，  VOFFSET  VR[1]
Case 2:      MOVEL PR[2]，  500mm/s，  FINE，  VOFFSET VR[1]
End
```

4. Program creation and execution

The creation, modification, deletion and other operations of the programs must be carried out in the manual mode but not in the auto mode.

4.1 Create Program

There are two ways to create a program:

1. Click "Menu Button" → "Program" to enter the interface as shown in Fig. 4.1. Click "Create Program", and a new program interface will pop up as shown in Fig. 4.2. After filling in program name and comments and selecting the program type on this interface, click "Confirm" to complete the program creation.

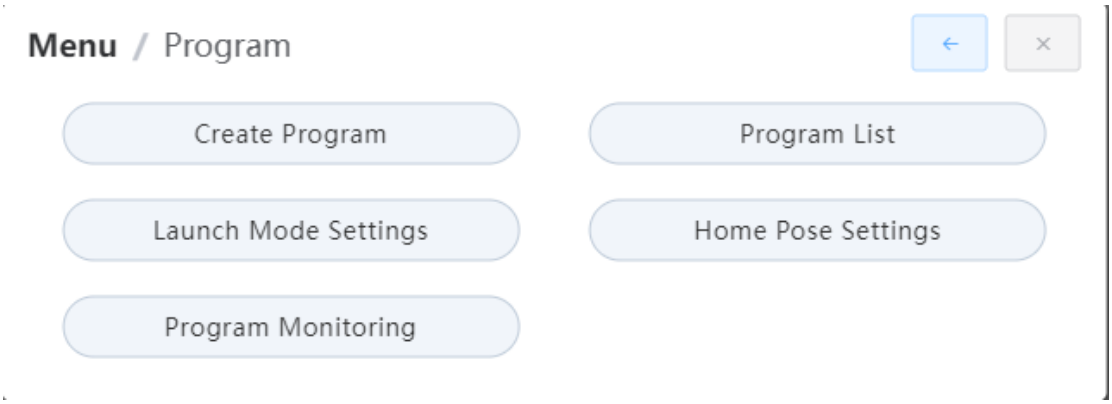


Fig. 4.1 Program Menu Interface

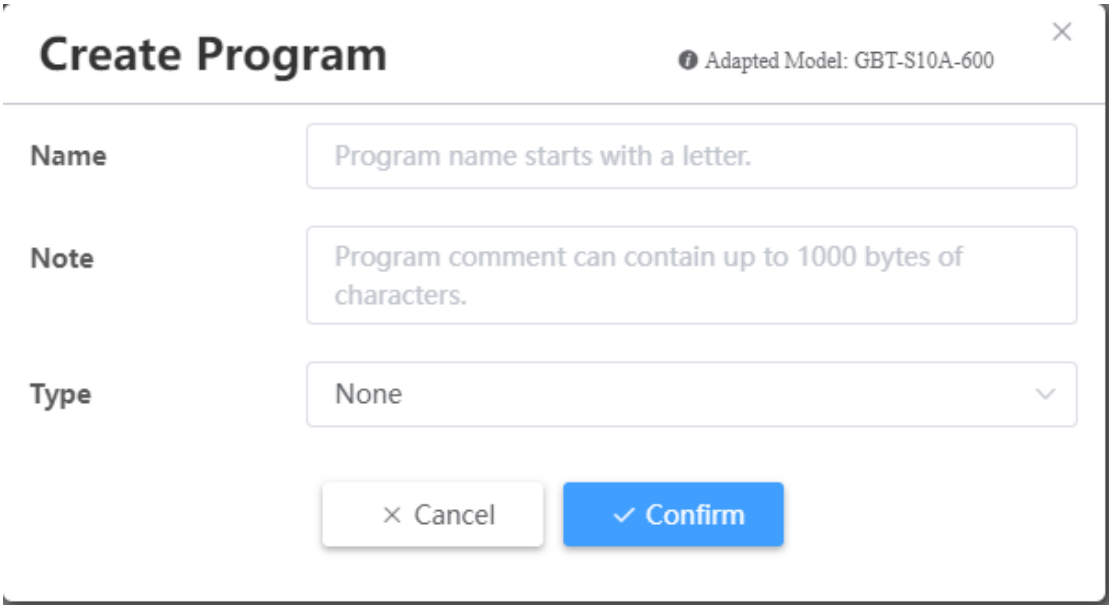


Fig. 4.2 Create Program Window

2. In the second way, click "Menu Button" → "Program" → “Program List” to enter the interface as shown in Fig. 4.3. Click "New Program", and an interface will pop up as shown in Fig. 4.2. After filling in program name and comments and selecting the program type on this interface, click "Confirm" to complete the program creation.

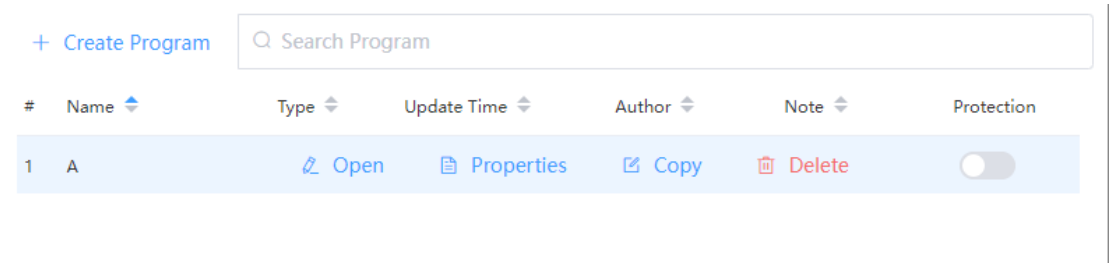


Fig. 4.3 Program List Interface

4.1.1 Description of program operation interface

Open the newly created program shown in Fig. 4.4. There are only blank lines and "END" line.

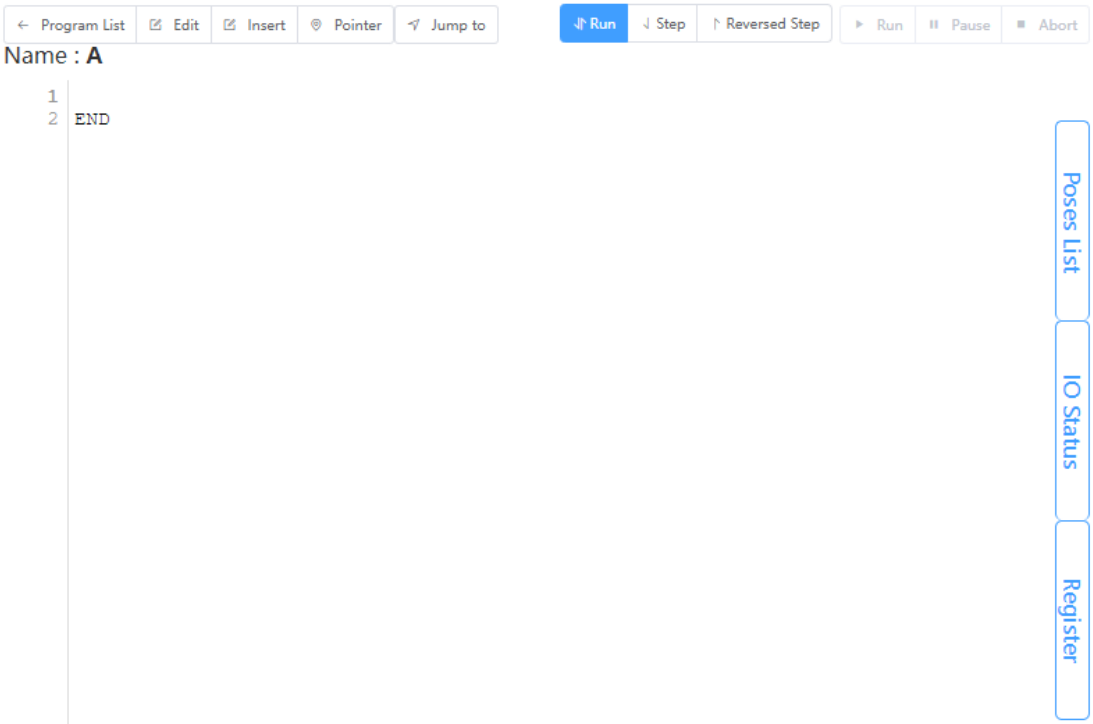


Fig. 4.4 Program Editing Interface

Description of buttons:

“Back”	Return to the program list interface.
“Modify instruction”	Edit existing program instructions: Move up/down: Move the current program line instruction before or after the previous line; Copy/Cut: Copy or cut current program line instruction to the clipboard; Paste: Paste the content on the clipboard to the next line of the current program line; Undo/Redo: Cancel or redo the previous operation; Disable/Enable: Disable or enable the current program line instruction. It may also be conveniently used for debugging. Add a blank line: Add a new blank line on the next line of the current program line.
“Insert Instruction”	Insert a new instruction, such as "common instructions, motion instruction, logical instruction, assignment instruction, I/O instruction, structure instruction and other instructions".

“Arrow to Cursor”	Move the line number of initial running program to the line number pointed by the cursor, for example, the initial program line number is no longer the first line, but the fifth line in the following figure. The execution cursor is an arrow. It is located at the far left of the program line number, indicating the program line number. This operation can only be performed in the manual mode.
“Jump To”	Indicate the corresponding program line number in the input program. Click "go" to move the cursor to the program line number.
“Continuous”	It is selected at default, indicating that the program is executed continuously in a positive order until the selected program is completed or paused/aborted.
“Single step & positive sequence”	Execute the programs one by one with a single step and in a positive sequence. Press it once to run an instruction at a time. After execution, the program execution status remains "paused". This operation can only be performed in the manual mode.
“Single step & reverse sequence”	Execute the programs one by one in a reverse sequence. Press it once to run an instruction at a time. After execution, the program execution status remains "paused". Only motion instructions rather than control instructions can be executed in the reverse sequence. This operation can only be performed in the manual mode.
“Pose”	Usually, the pose list is hidden on the right side of the program editing interface, as shown in Fig. 4.5; it only appears when the user clicks on it. The pose list contains private P and global PR of the program. After these points are changed during programming or modified on the PR register interface, the pose list interface should also be changed accordingly; vice versa. In addition, when the servo is powered on, press and hold the "Move" button in the bottom left corner to slowly move the robot to the pose after it is selected in the pose list interface.

“I/O Status”	Usually, this interface is hidden on the right side of the programming interface, as shown in Fig. 4.5; it only pops out when the user clicks on it. This interface has the same contents as and share data with the "Communication → I/O Status" interface. It is to facilitate the user to monitor I/O status during programming and debugging.
“Register”	Usually, this interface is hidden on the right side of the programming interface, as shown in Fig. 4.5; it only pops out when the user clicks on it. This interface has the same contents as and share data with relevant register interface. It is to facilitate the user to monitor the register status during programming and debugging.

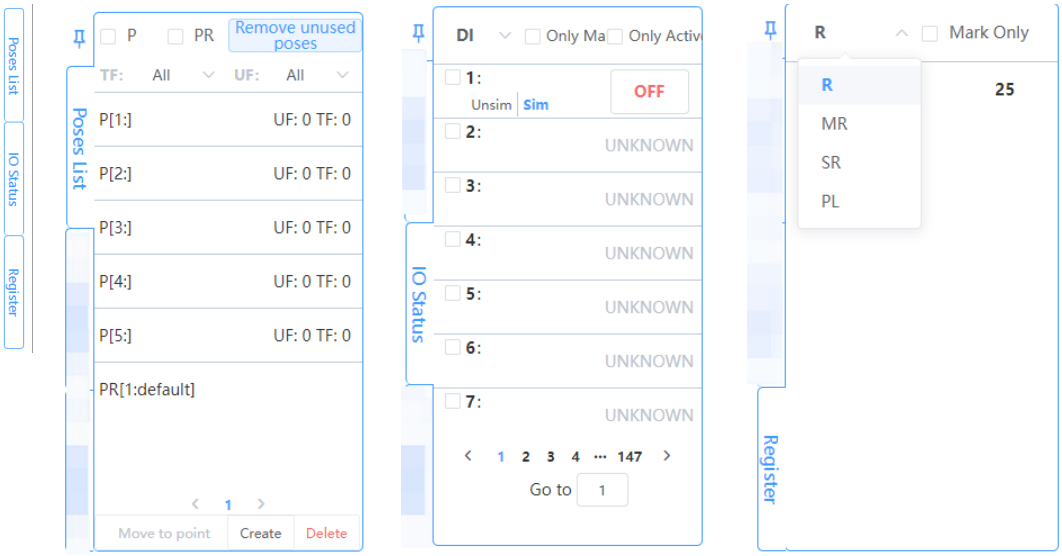
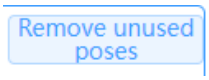
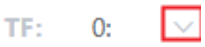
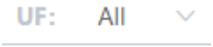
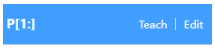
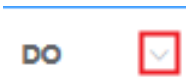
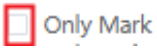
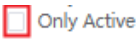


Fig. 4.5 shows the pose list, I/O status list and register list from left to right.

Description of function keys on pose, I/O status and register interfaces:

Function key	Specific function
	Click here to lock the "Pose", "I/O Status", and "Register" interfaces in the program editing screen. Click again to unlock them. It facilitates monitoring of relevant data status during debugging.
	Click here in the upper left corner of the pose screen to only display P variable poses.

	Click here directly above the pose screen to only display PR variable poses.
	Click here in the upper right corner of the pose screen to delete P and PR poses you want to delete, or to clear all poses with one click.
	Click here on the pose screen to select which tool coordinate system is used as the reference for the established poses.
	Click here on the pose screen to select which user coordinate system is used as the reference for the established poses.
	Choose the desired pose in the pose screen, where you can re-teach pose data and manually edit pose data.
“Move to point”	After the servo is powered on in the robot's manual mode, choose the pose to be moved in the pose screen and click here to move the robot to the target pose.
“Create”	Click here in the pose screen to create new pose data for the robot.
“Delete”	Click here on the pose screen to delete the pose data you want.
	Click here on the I/O status screen to switch between the I/O types to be monitored.
	Click here on the I/O status screen to mask the monitored I/O status information.
	Click here on the I/O status screen to mask unconfigured I/O states and only display I/O states successfully configured.
	Click here on the register screen to switch to MR and SR register monitoring screens.
	Click here on the register status screen to mask the monitored register status.

4.1.2 Copy program

Select the desired program in the program list interface and click "Copy Program" as shown in Fig. 4.6. The user can copy the currently selected program and redefine the name, comments and type of the copied program. (Note: The program name cannot be the same as the name in the current program list) Copy Program is only open at or above the programmer level.

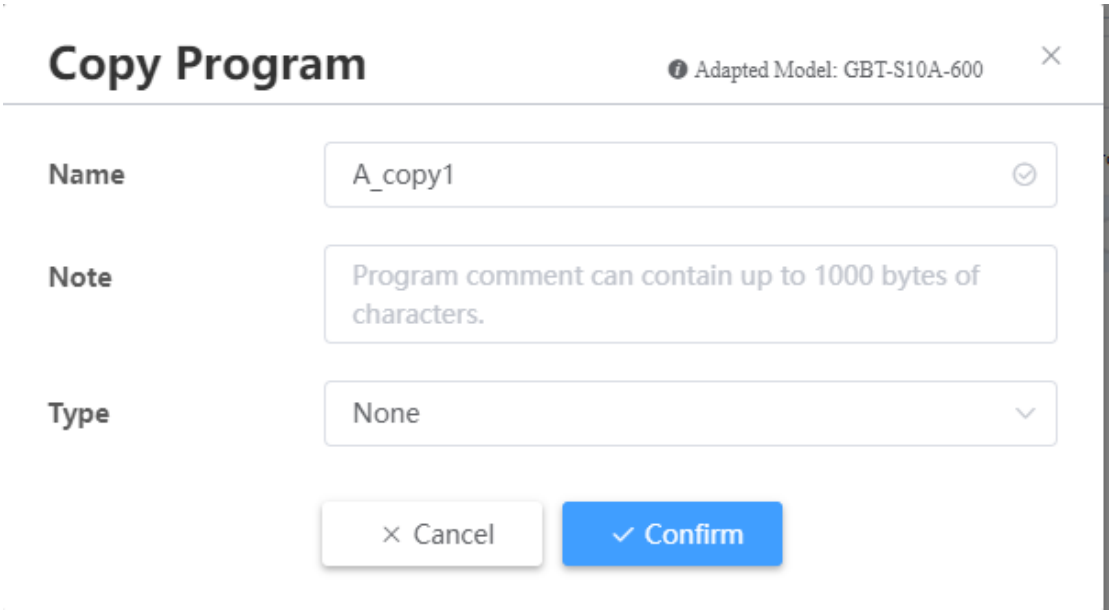


Fig. 4.6 Copy Program Window

4.1.3 Delete program

Choose the desired program in the program list interface and click "Delete Program" to show the interface shown in Fig. 4.7. Then, click "Confirm" to permanently delete the program, which cannot be restored. Delete Program is only open to the levels above programmer.

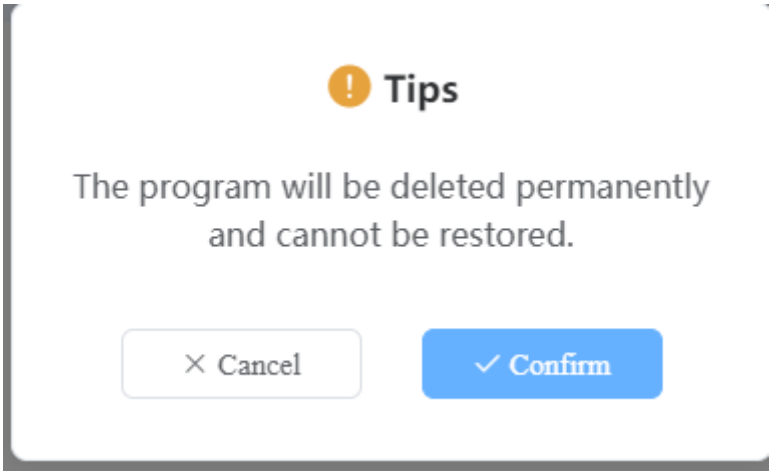


Fig. 4.7 Delete Program Window

4.1.4 Choose program

Click "Menu Button" → "Program" to enter the interface as shown in Fig. 4.8. Click "Program List" to enter the program list interface as shown in Fig. 4.9. Click the desired program as shown in Fig. 4.10. Then, click "Open Program" to enter the current program editing interface as shown in Fig. 4.11.

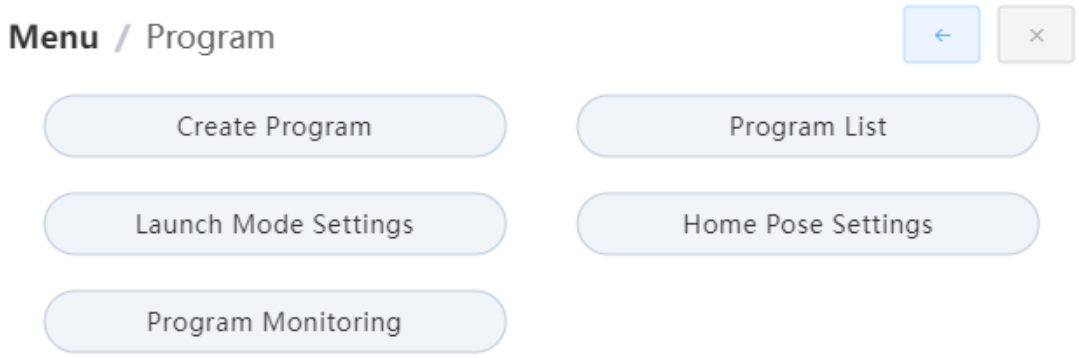


Fig. 4.8 Program Menu Window

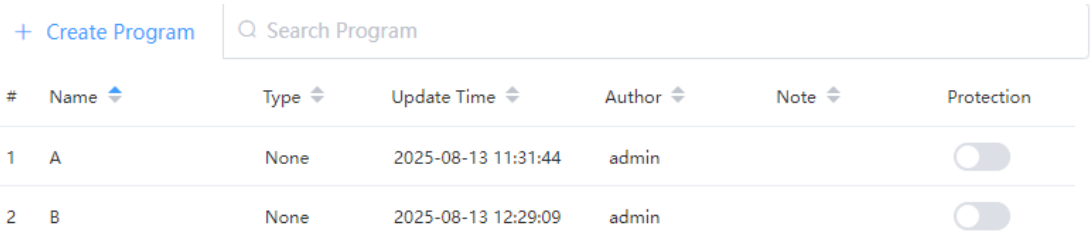


Fig. 4.9 Program List Window

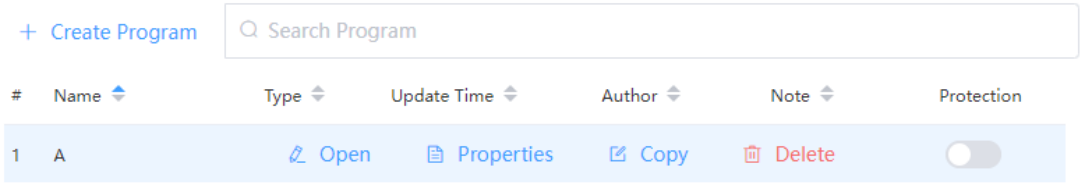


Fig. 4.10 Selected Program

Fig. 4.11 Program Editing Window

4.1.5 Create action instruction

Insert an action instruction according to the following steps:

1. On the program operation interface, click and select the program line to be

inserted into the program position and select the program line. Then, a cursor may display in front of the program line number, as shown in Fig. 4.12.

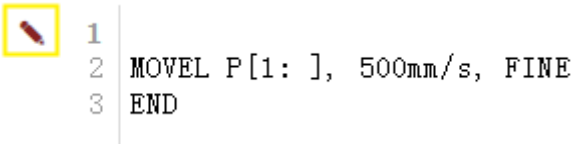


Fig. 4.12 Pen Cursor

2. Click on "Insert Instruction" in the program operation interface to enter the instruction selection interface, as shown in Fig. 4.13 → select the desired motion instruction (e.g. Move Line) to enter the motion instruction editing interface.

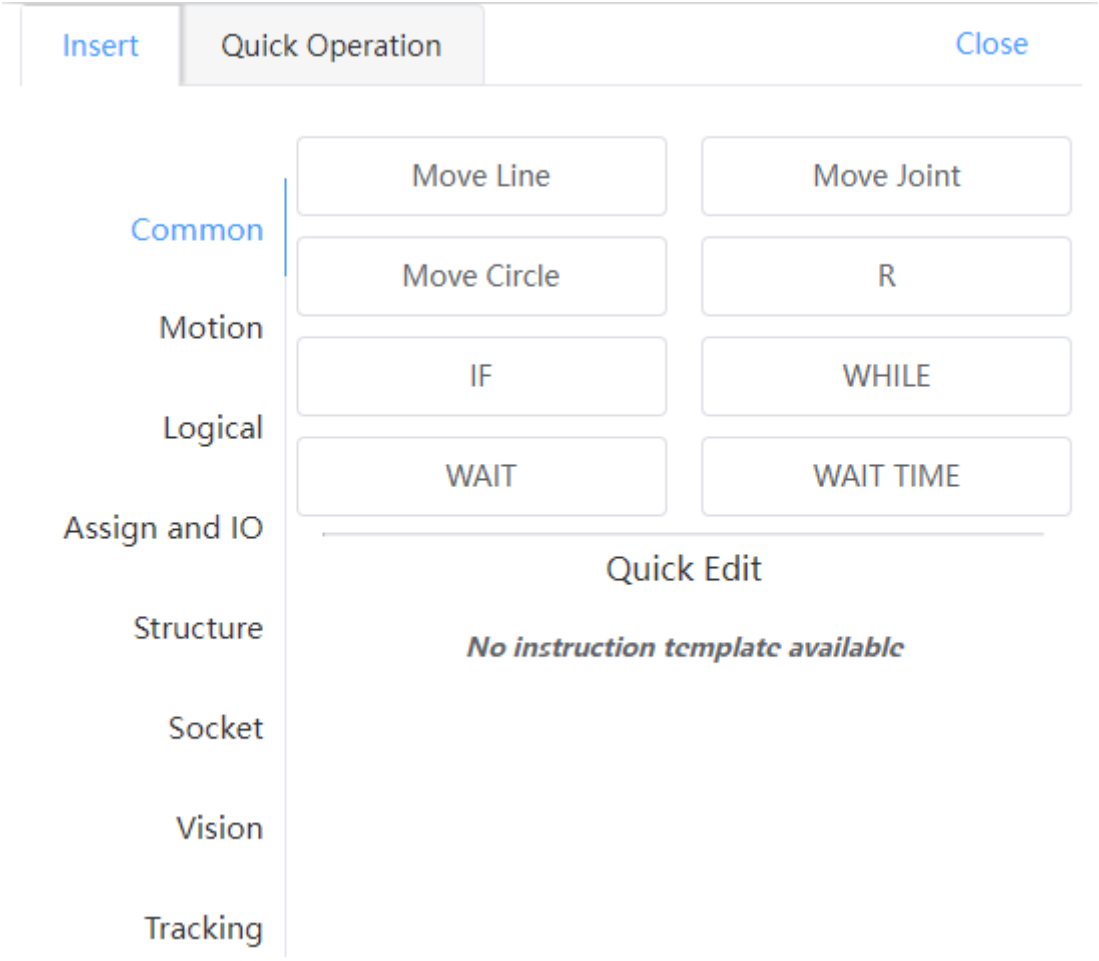


Fig. 4.13 Instruction Selection Window

3. Modify the contents of motion instruction according to the actual situation.

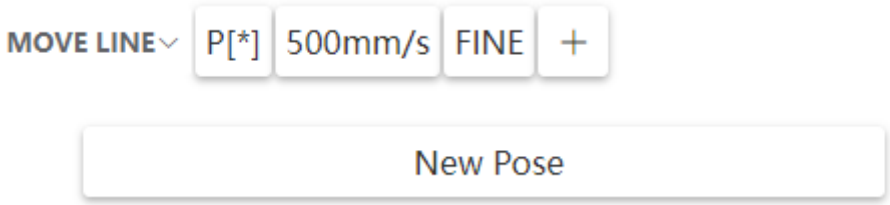


Fig. 4.14 Motion Instruction Editing Window

4. Teach the robot to the desired target position, click on new pose in Fig. 4.14, and "*" in P [*] will display specific numbers as shown in Fig. 4.15. Click "Confirm" to complete the insertion of motion instructions (insertion steps are the same for MOVEJ and MOVEC instructions).

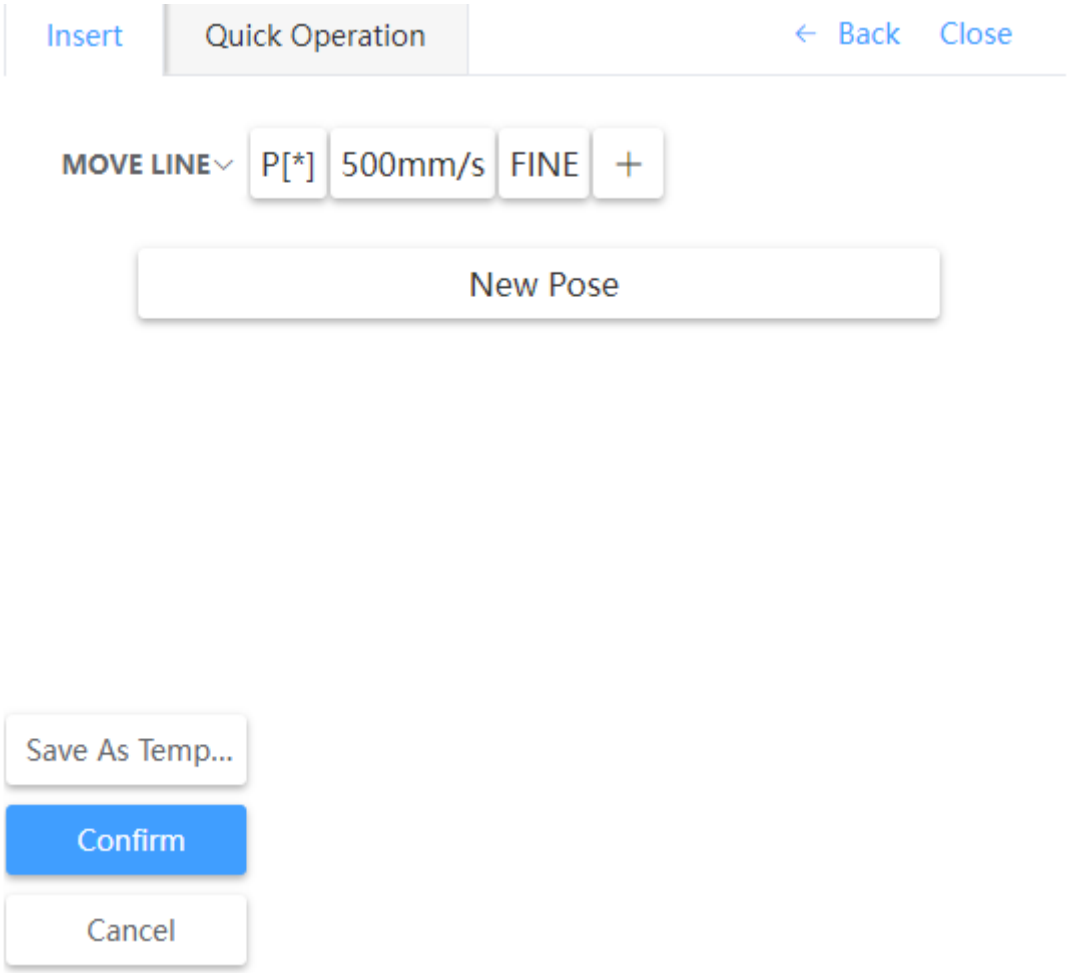


Fig. 4.15 Example of Continuous Motion Instruction

Additional descriptions:

5. Click on the motion instruction editing interface to switch between motion instructions (MOVJ, MOVEC). In order to edit P, click "P [*]" in Fig. 4.14 to display the following screen.

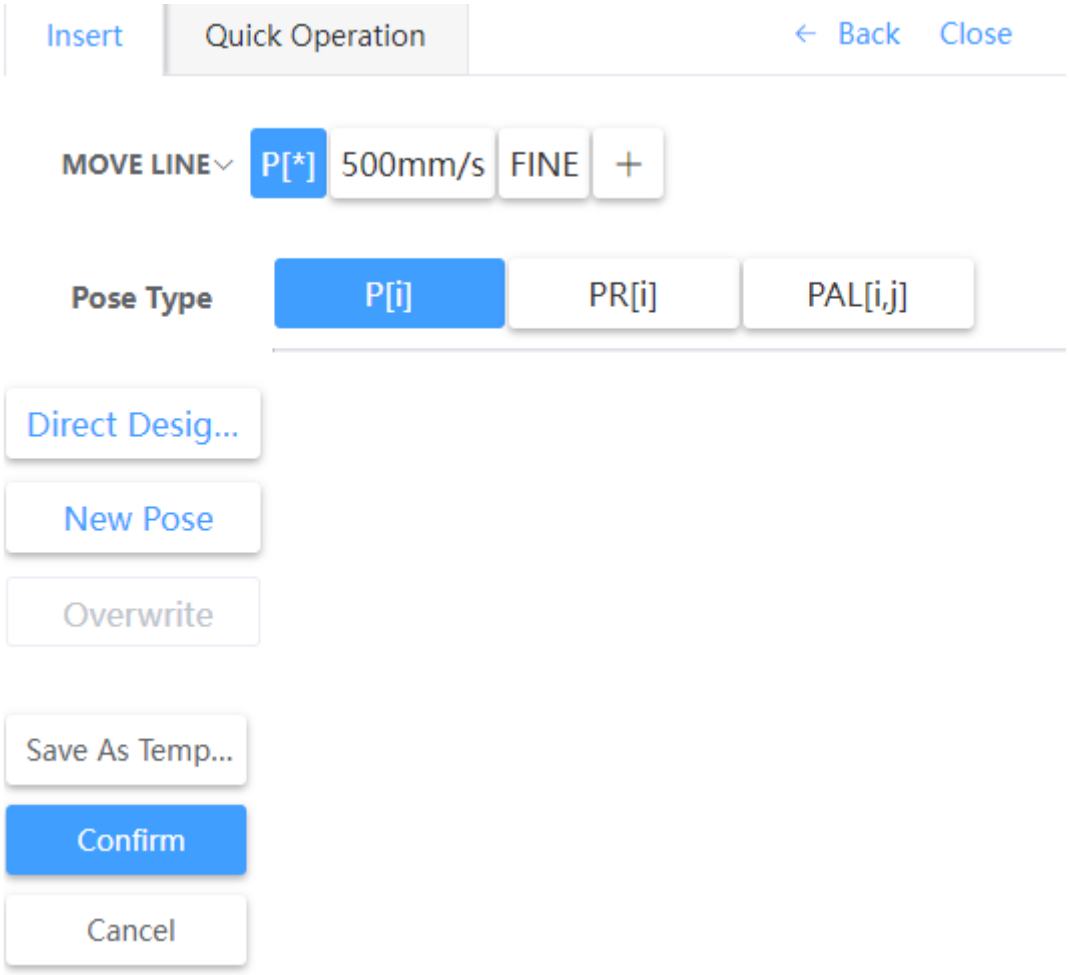


Fig. 4.16 Point or Position Register Creation Window

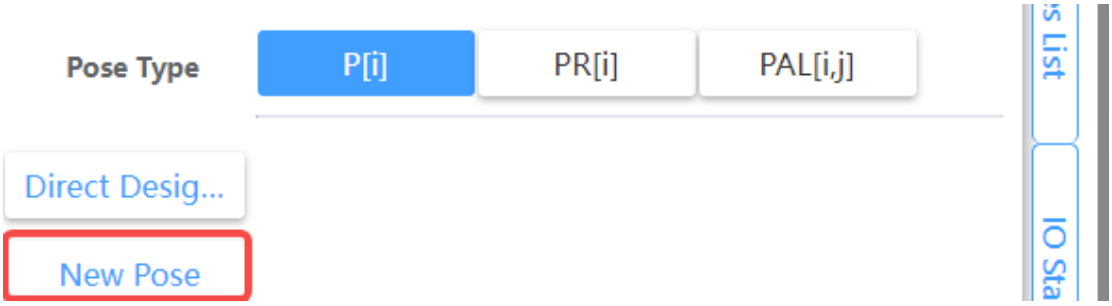
There are two "P[*]" position types: P [i] and PR [i]. P [i] can be understood as a local variable, namely, P [i] in each program can exclusively be used in that program. PR [i] can be understood as an overall variable and shared for all programs. They should be selected according to actual needs.

Parameter i can be specified only directly when P [i] is used or indirectly when PR [i] is used. When Parameter i can be indirectly specified, i in PR [i] can be an MR register, for example, shows the indirect method.

```
MOVEL PR[MR[1: ]], 500mm/s, FINE
```

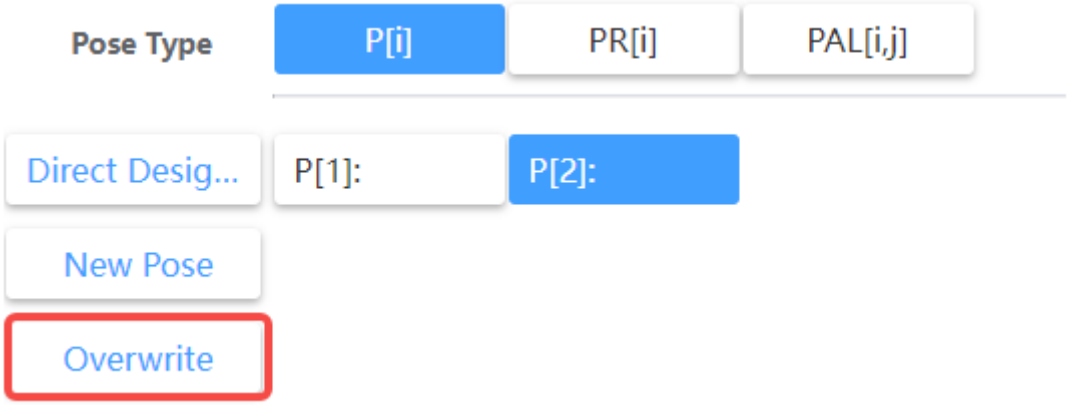
The speed parameter can also be specified by the MR register. Click the speed parameter in the motion instruction editing interface to modify it. Click the positioning type parameter "FINE" to modify the positioning type.

New Pose



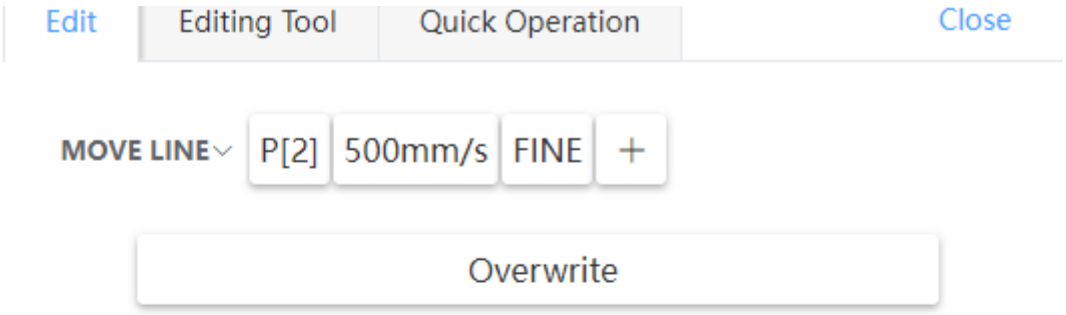
Click to create P or PR.

Record Pose



It is in light color “” if the pose to be taught or modified again is not selected and in deep color “” after the pose has been selected for teaching. Then, click “Record Pose” to update pose data.

Save as Template



After clicking on "Save as Template", the following screen will appear on the "Common Instructions" and "Motion Instructions" interfaces.

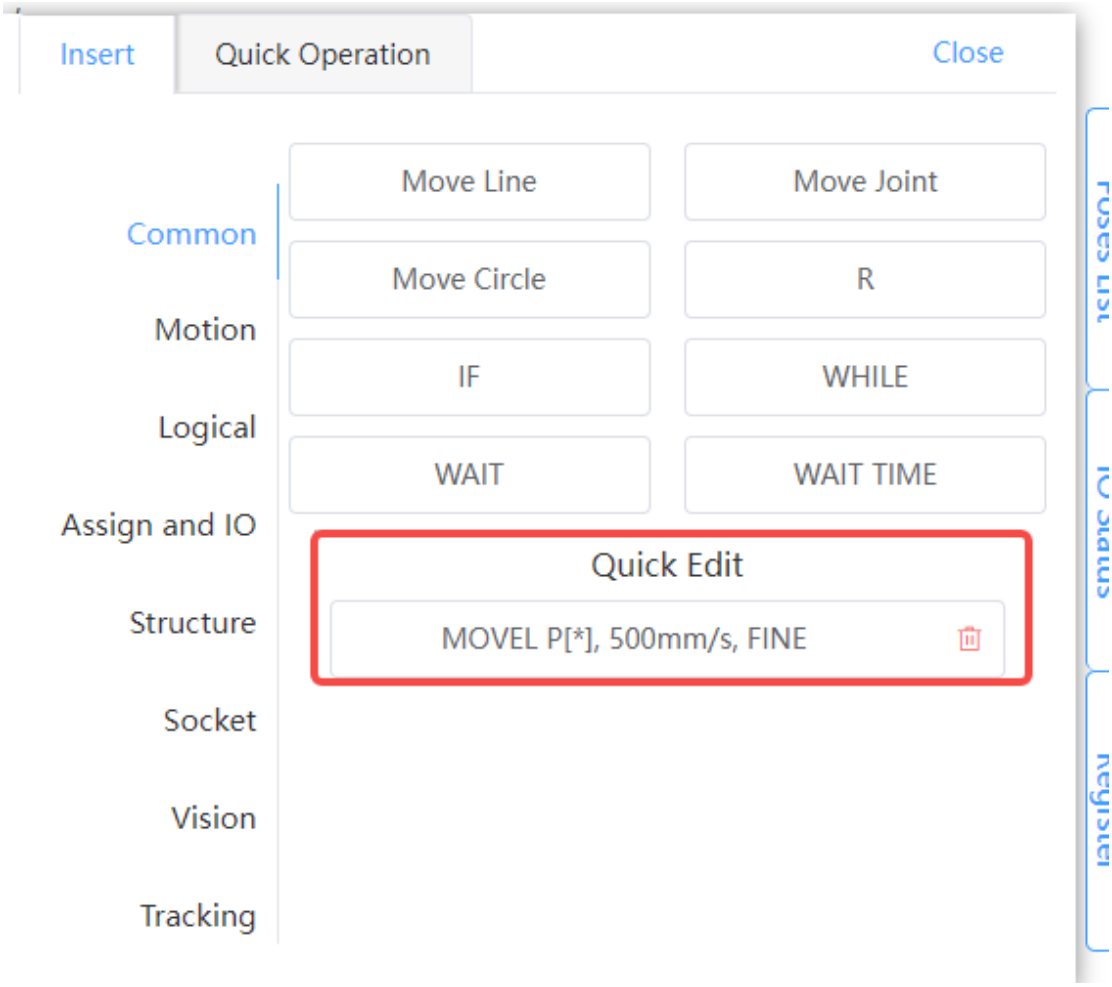


Fig. 4.17 Motion Instruction Template Window

Click the motion instruction statement in Quick Edit to quickly insert motion instructions into the program.

4.1.6 Modify motion instruction

Select the motion instruction to be modified in the program editing interface, and a cursor will appear in front of the selected statement line, as shown in Fig. 4.12.

After clicking "Edit Instruction", the following screen will appear.

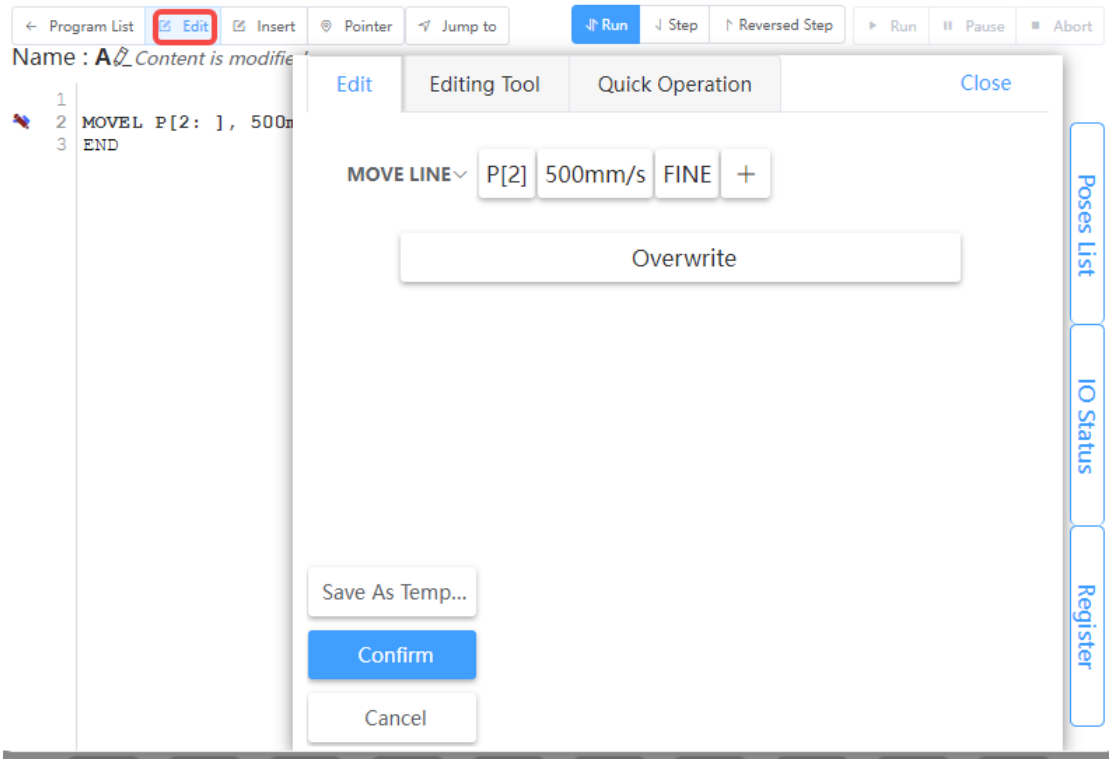


Fig. 4.17 Motion Instruction Editing Window

Refer to the “create instruction” steps in Section 4.1.5 to modify it.

4.1.7 Create additional instruction

Based on the previous section, click "+" after the motion instruction in the editing interface, and the following screen will appear.

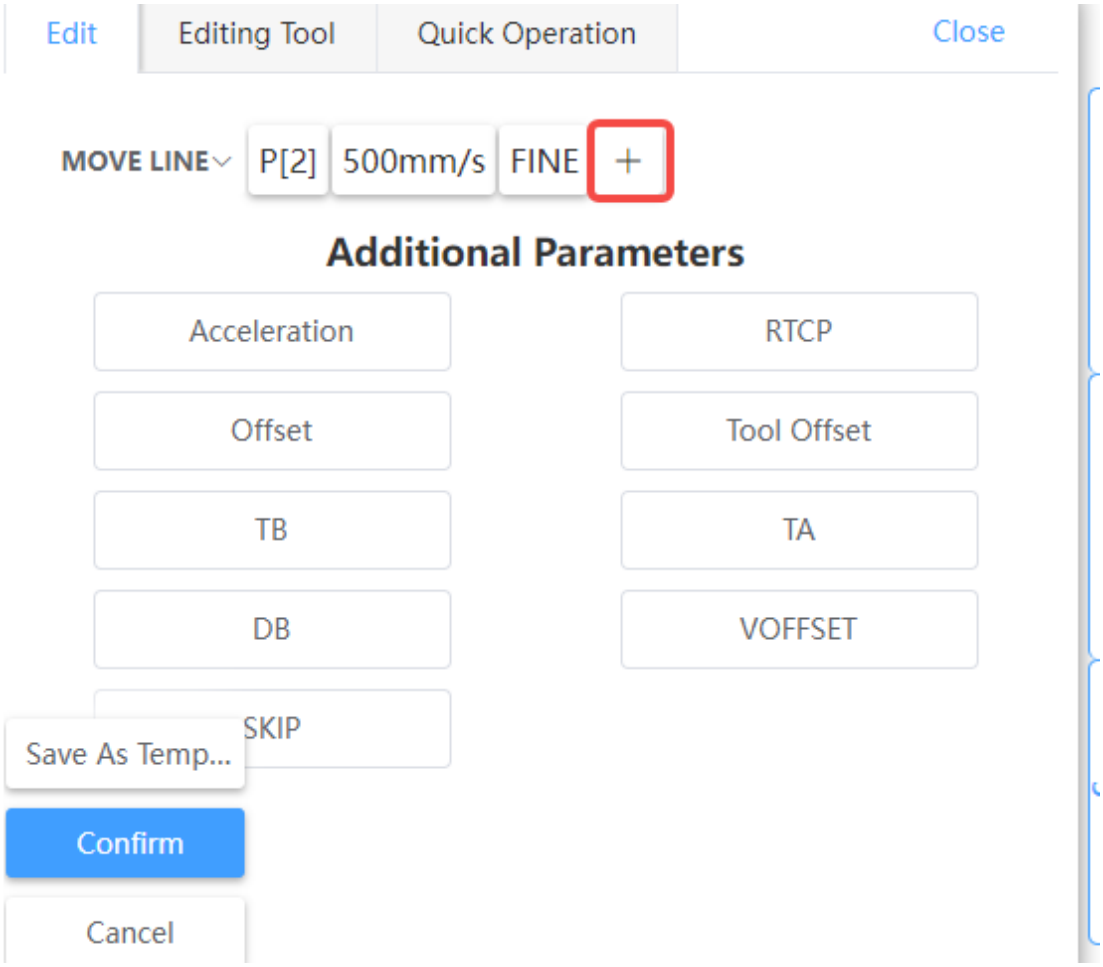


Fig. 4.18 Additional Instruction Editing Window

Select the required additional parameters. Refer to Section 3.3.5 for instructions on additional parameters.

4.1.8 Insert control instruction

Control instructions are a general term for program instructions (other than action instructions) used on the robots.

Insert If instruction

After clicking "Insert Instruction" in the program editing interface, enter the instruction selection interface. Then, click "Common Instruction" or "Logical Instruction" as shown in the following figure.

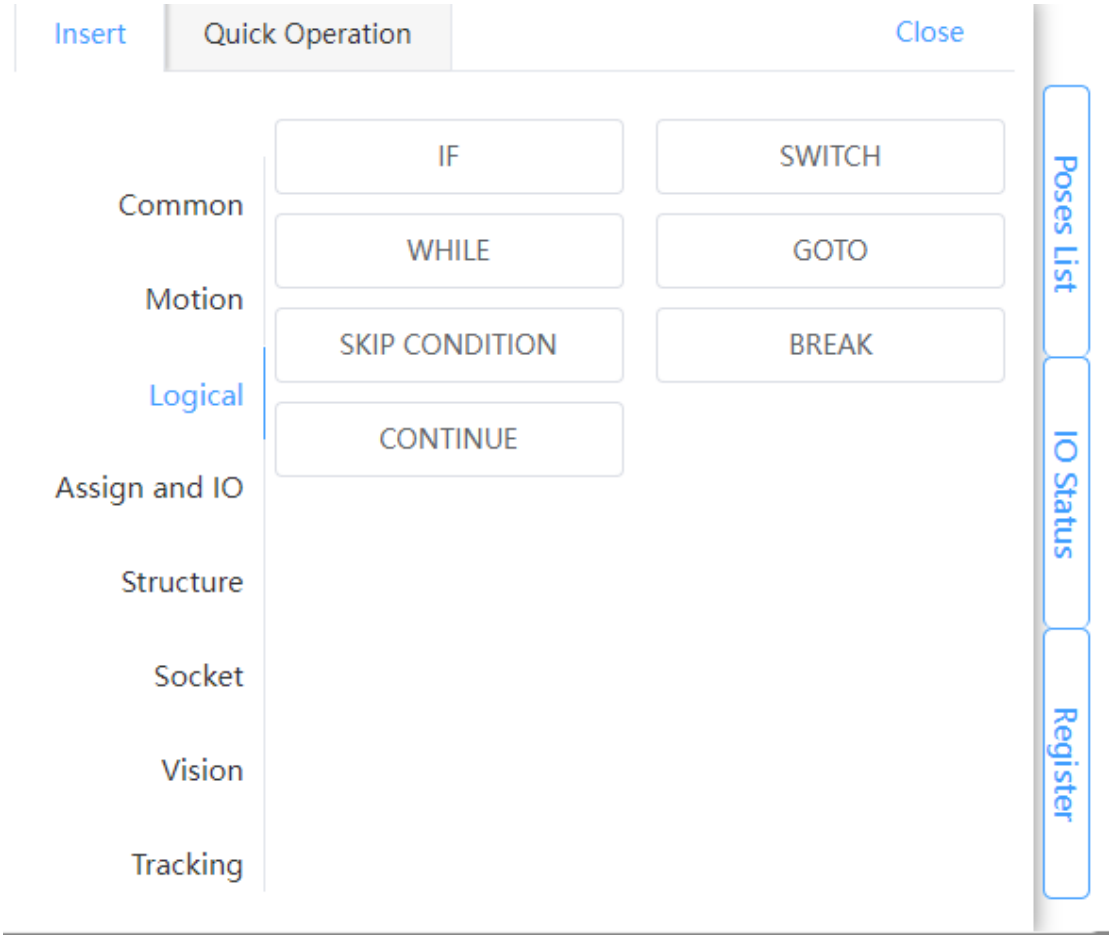


Fig. 4.18 Insert Instruction Window

Select IF to enter the IF statement editing interface as shown in Fig. 4.19. Click "*" to add corresponding parameters and then click "Confirm" to complete the addition of IF statement. See additional descriptions of the IF statement for addition of parameters.

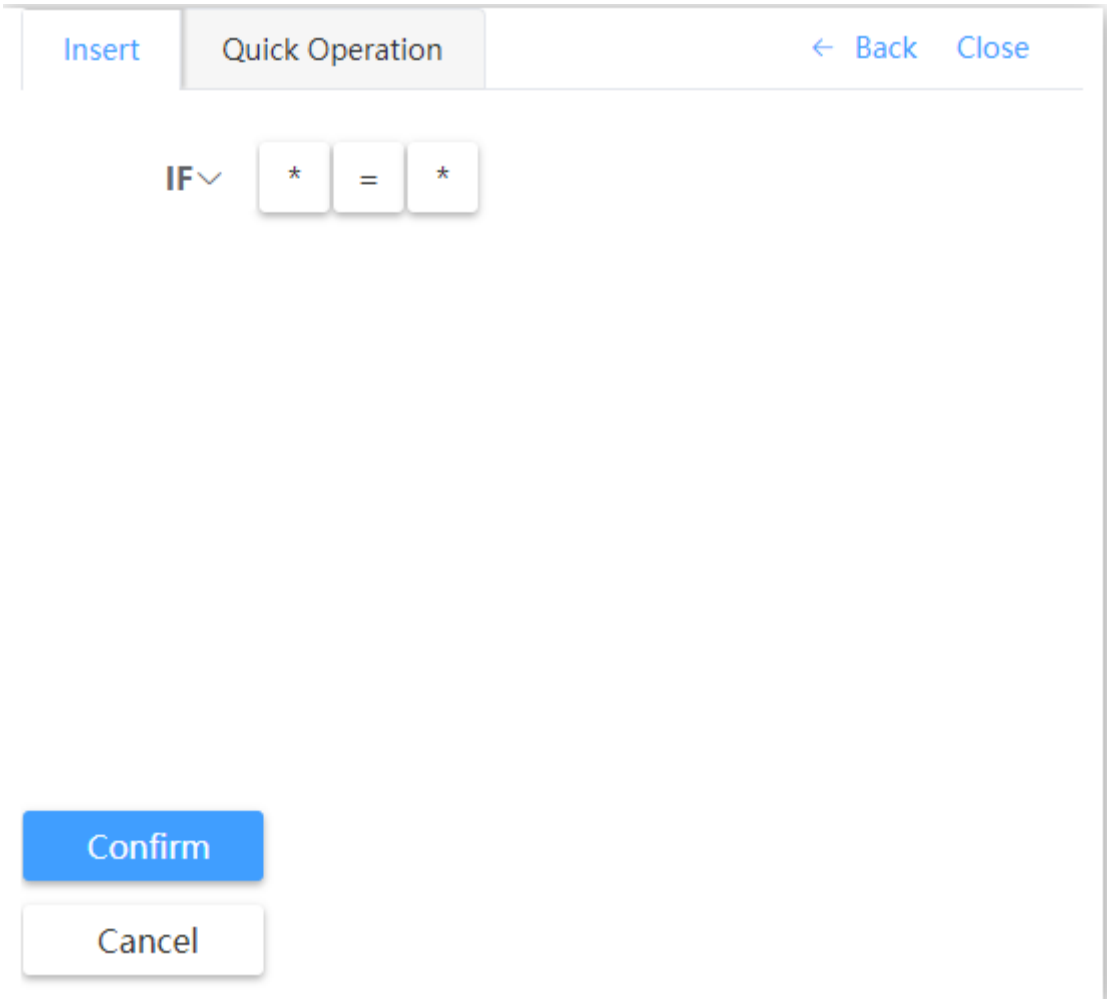


Fig. 4.19 Insert Instruction Window

An example of IF statement programming is shown in Fig. 4.20:

```
IF R[1: default]  =  R[2: default]
  R[2: default] = 1
ELIF R[2: default]  =  R[3: default]
  R[2: default] = 2
ELSE
  R[2: default] = 3
END IF
```

Fig. 4.20 Example of IF Instruction Program

Additional descriptions of IF statement

In the IF statement editing interface, click to switch among IF, ELSE IF and ELSE, as

shown in the following figure.

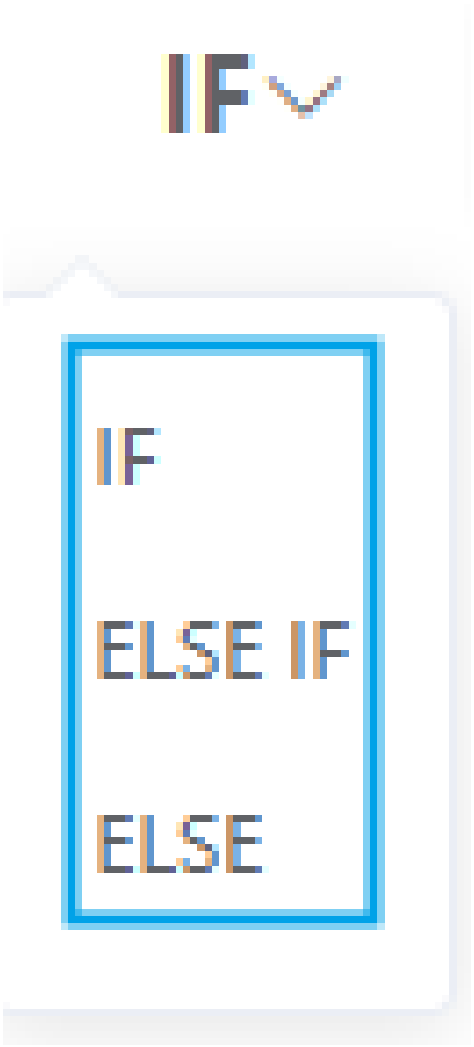
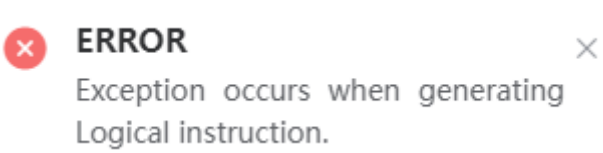


Fig. 4.21 IF Instruction Switching Interface

Before inserting ELSE IF or ELSE, it is necessary to insert IF. Otherwise, ELSE IF or ELSE cannot be inserted successfully and error prompts will appear as well.



Click "*" in the IF statement interface in Fig. 4.22 to display the IF statement parameter editing interface for adding judgment conditions.

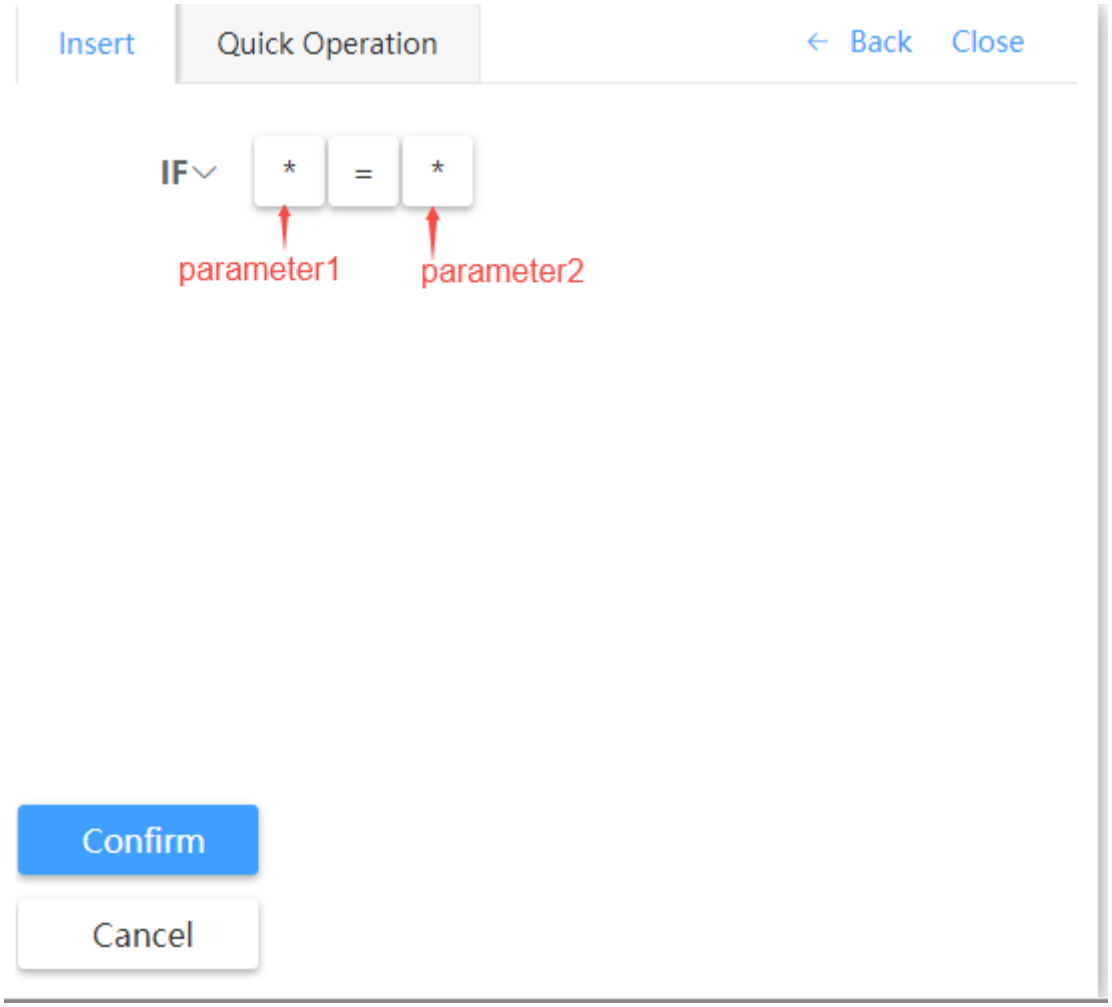


Fig. 4.22 IF Statement Parameter Editing Interface

Parameter 1 include registers and I/O, of which the register type includes number register R [i] and the I/O types include special I/O and digital I/O.

Parameter 2 includes register, I/O and I/O status, of which the register type includes number register R [i], the I/O type includes special I/O and digital I/O, and the I/O status includes ON and OFF.

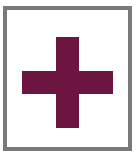
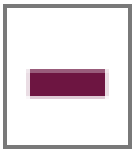
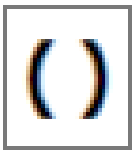
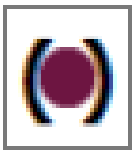
Element type

It refers to optional register and I/O type for Parameter 1 or Parameter 2.

Element list

It refers to specific elements for register and I/O type of Parameter 1 or Parameter 2. Only elements created or configured are displayed in this list.

Explanation of symbols in the detailed interface of IF statement

	Add an expression.
	Remove an expression (i.e. at least one expression, namely, at least a term to the right of the equation)
	Add parentheses.
	Remove parentheses.

Click "=" to switch operators in the IF statement editing interface. For the types of replaceable operators and their usage methods, please refer to 3.8.11 Compound Operation Instructions.

Insert SWITCH instruction

Please refer to the insertion method of IF instruction.

Insert WHILE instruction

Please refer to the insertion method of IF instruction.

Insert register and I/O instructions

See Section 3.4 for description of register instructions.

Insert number register - R[i]

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Assign and I/O" to switch to the "Assign and I/O" instruction interface, as shown in Fig. 4.23.

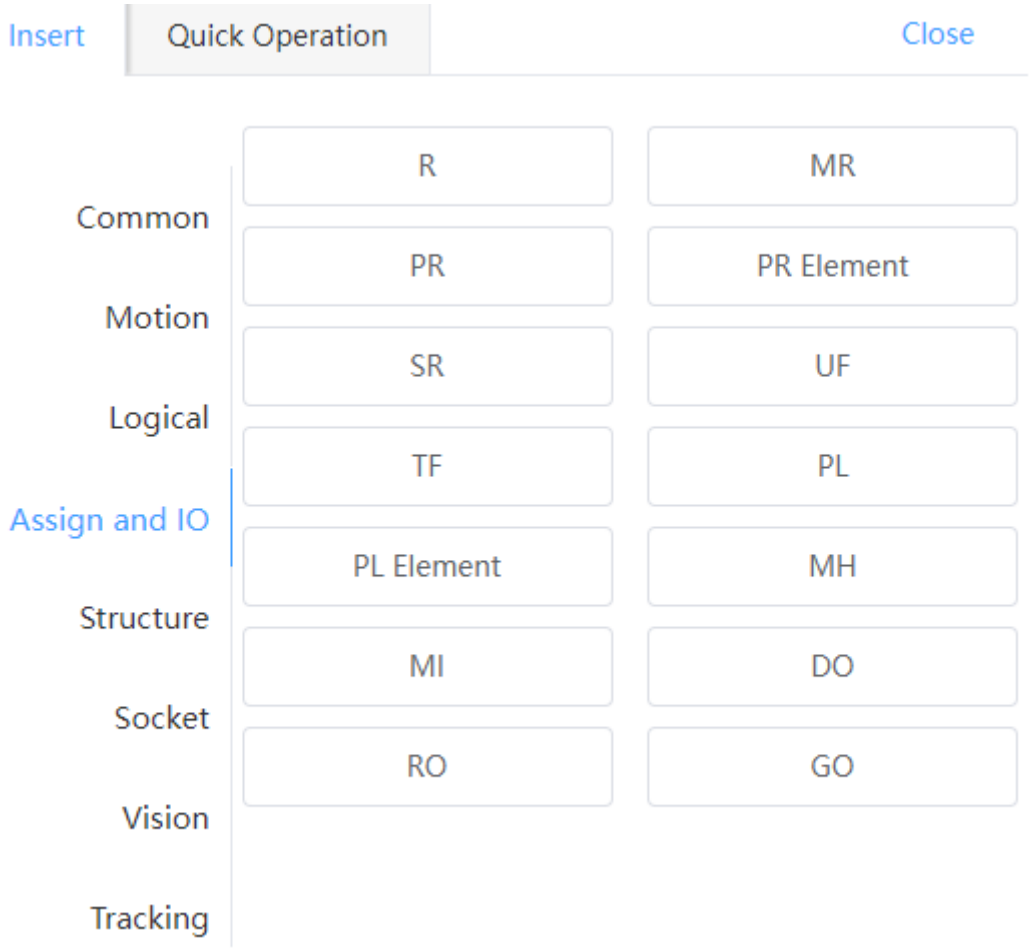


Fig. 4.23 “Assign and I/O" Instruction Interface

Click "R" to enter the R register instruction parameter filling interface, as shown in Fig. 4.24.

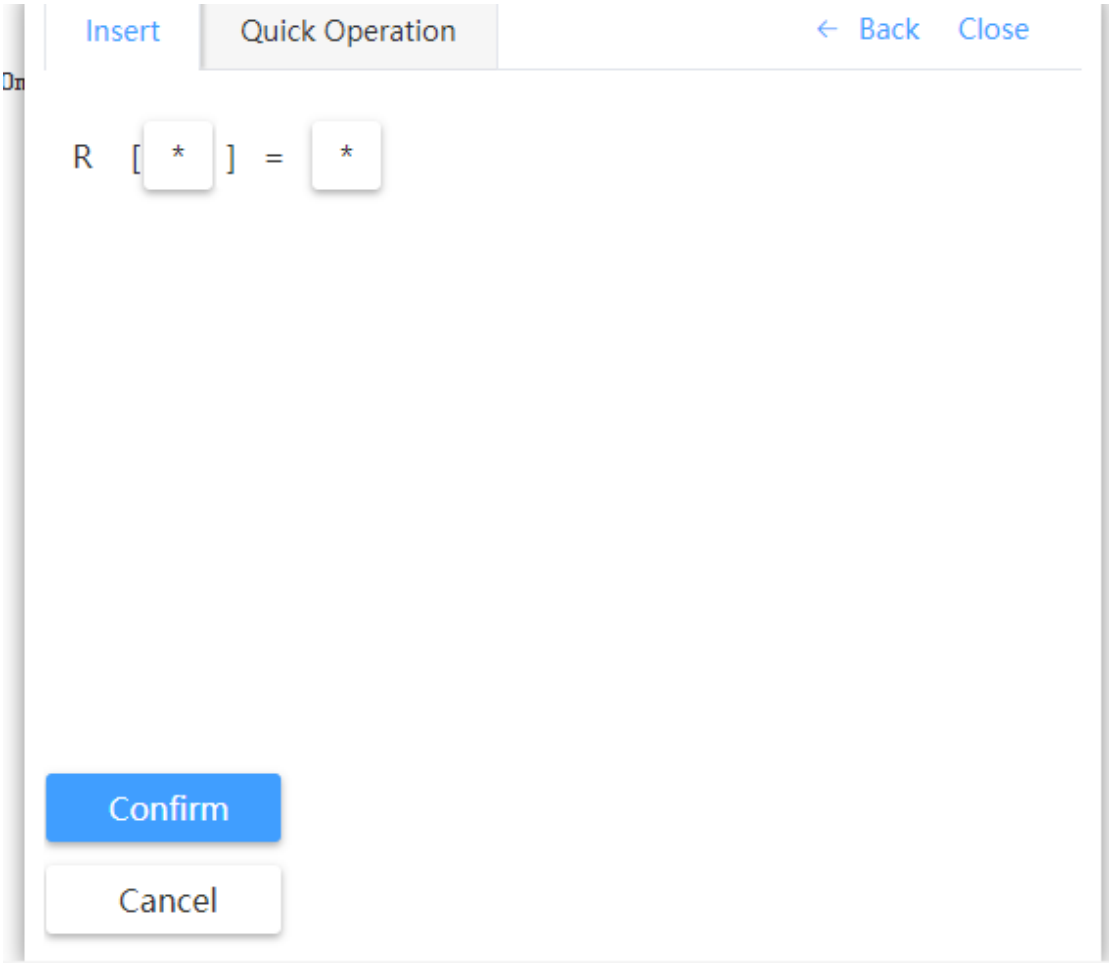


Fig. 4.24 “Assign and I/O" Instruction Interface

Click "*" to fill in the parameters. After that, click "Confirm" to complete instruction insertion. See Section 3.4.1 for description of R register parameters.

Caution

The insertion steps for other registers are similar to those for I/O (DO/RO) and R registers. Please refer to the insertion steps for registers.

Insert WAIT instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Structure Instructions" to switch to the "Structure Instructions" interface, as shown in Fig. 4.25.

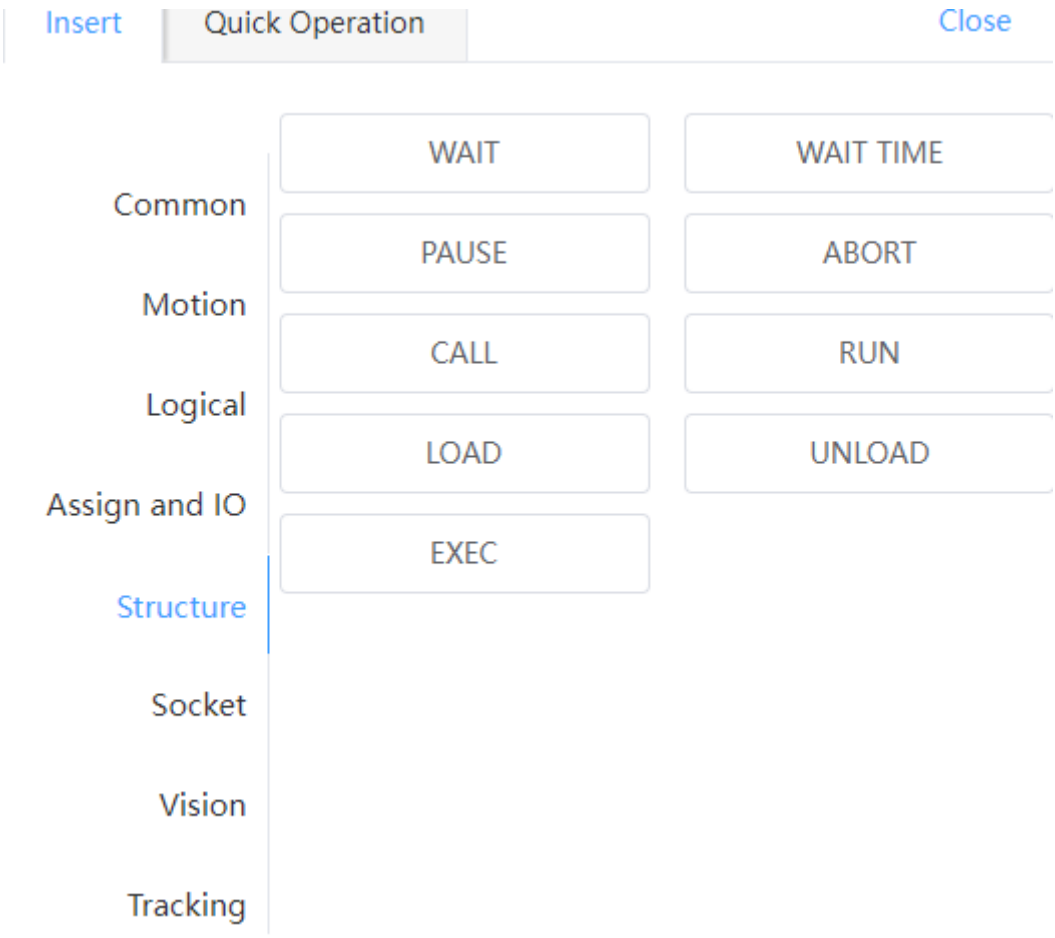


Fig. 4.25 Structure Instruction Interface

Click "WAIT" to enter the WAIT instruction parameter filling interface, as shown in Fig. 4.26.

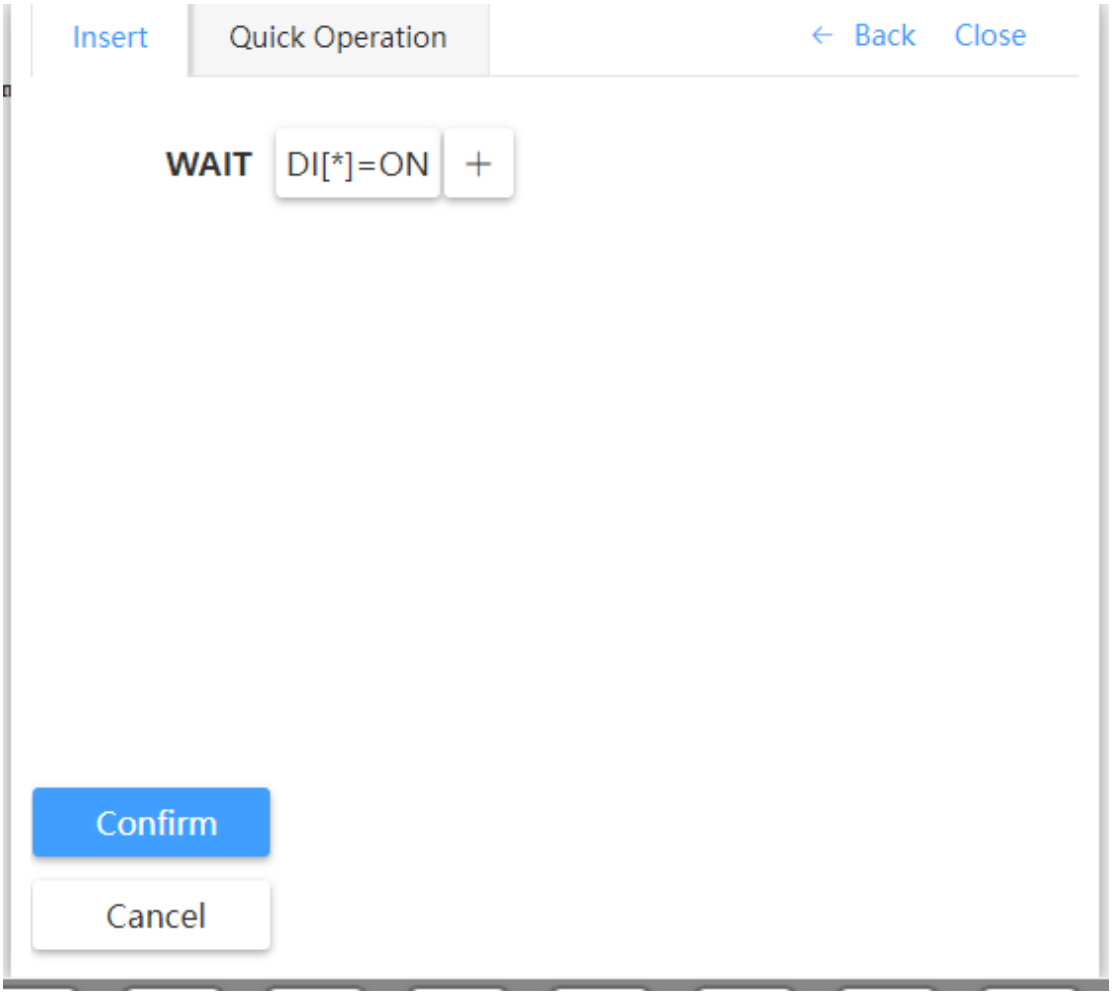


Fig. 4.26 WAIT Instruction Interface

Click "DI[*]=ON" to set the parameters. After that, click "Confirm" to complete instruction insertion.

Additional descriptions

When inserting the WAIT instruction, the default expression is . Click "DI[*]=ON" to enter the interface shown in Fig. 4.27, where you can modify the expression.

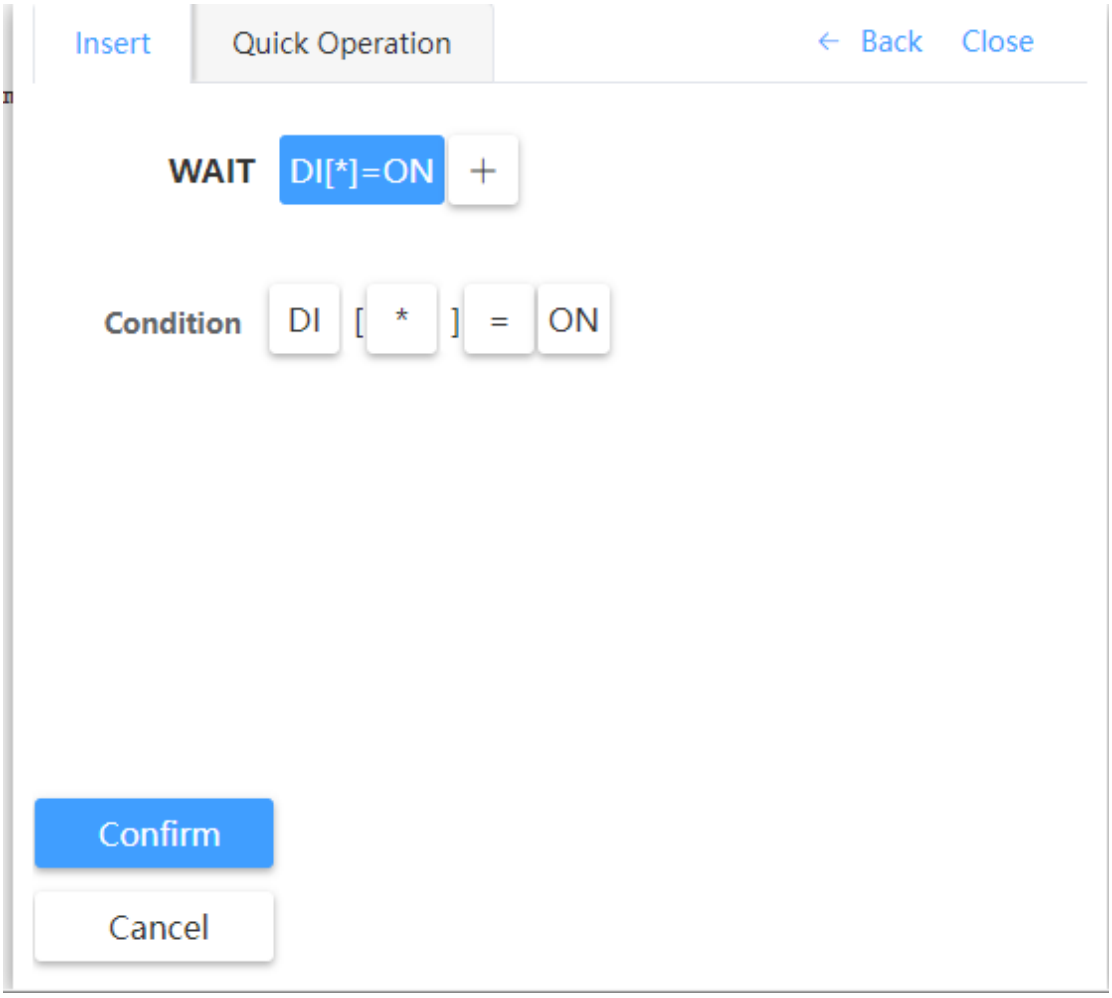


Fig. 4.27 WAIT Instruction Interface

If you want to change the element type to the left of "=" in the expression in the above figure, click "DI" to choose available element types of I/O type and register type. The I/O type includes (DI/DO/UI/UO) and the register type includes (R/MI/MH), as shown in Fig. 4.28.

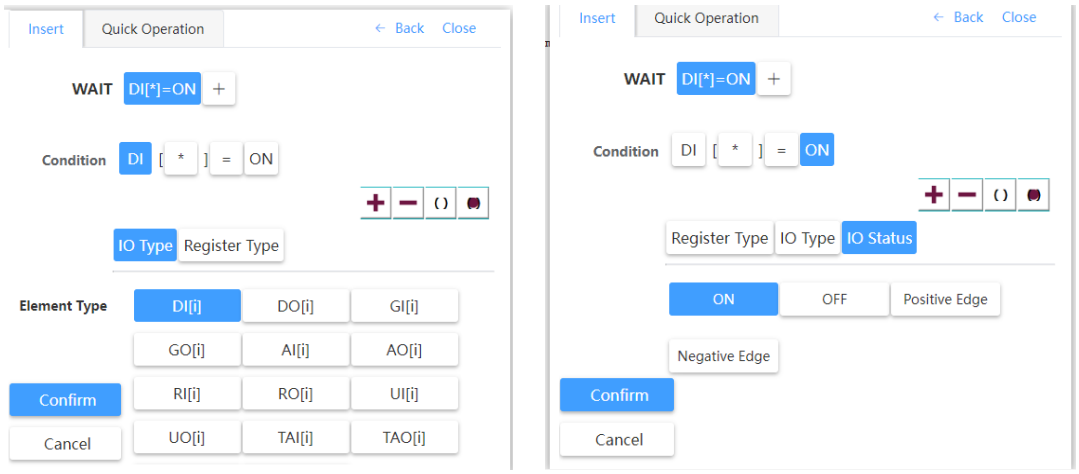


Fig. 4.28 WAIT Instruction Parameter Interface

The value of the element type to the right of "=" in the expression varies along with the element type to the left of "=". After setting, click "Confirm". Refer to Section 3.5 for I/O type values and Section 3.4 for register type values.

Insert WAIT TIME instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Structure Instructions" to switch to the "Structure Instructions" interface, as shown in Fig. 4.25.

Click "WAIT TIME" to enter the WAIT TIME instruction parameter filling interface, as shown in Fig. 4.29.

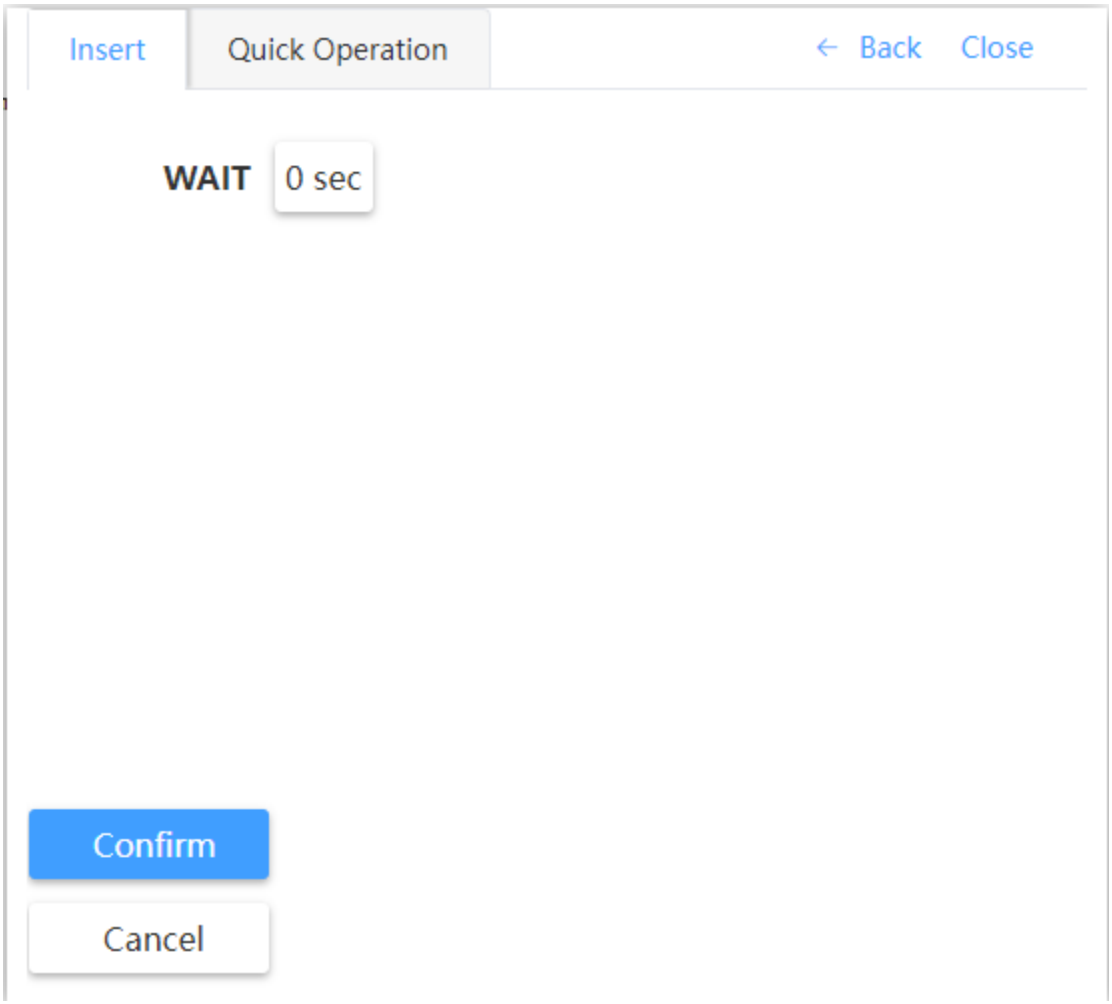


Fig. 4.29 WAIT TIME Instruction Parameter Filling Interface

Click "0 sec" to set the parameters. After that, click "Confirm" to complete instruction insertion.

Additional descriptions

Click "0sec" in Fig. 4.29 to pop out the interface shown in Fig. 4.30.

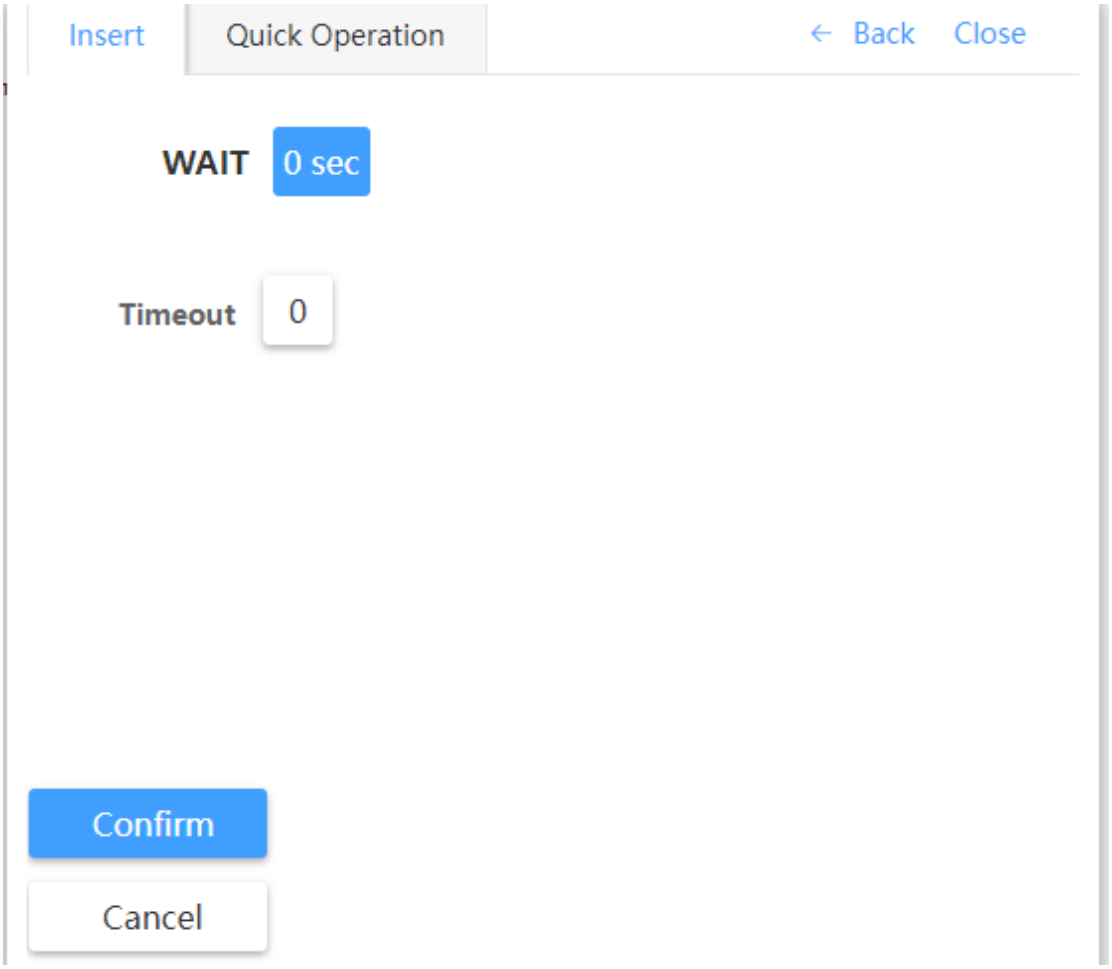


Fig. 4.30 WAIT TIME Instruction Parameter Filling Interface

Click "0" in Fig. 4.30 to set the timeout time. There are two data types for the timeout time: constant and number register (R).

Insert LOAD instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Structure Instructions" to switch to the "Structure Instructions" interface, as shown in Fig. 4.25.

Click "LOAD" to enter the LOAD instruction parameter filling interface, as shown in Fig. 4.31.

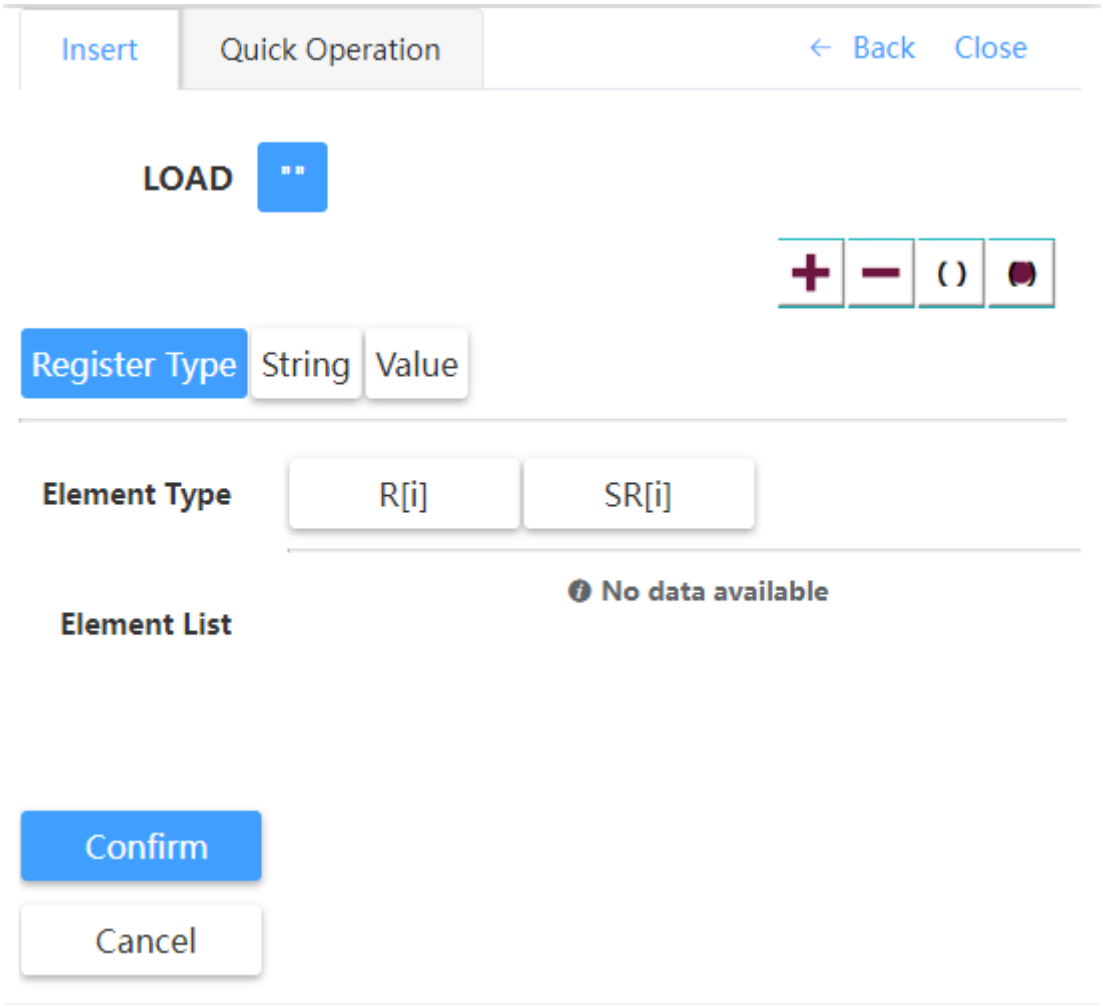


Fig. 4.31 LOAD Instruction Parameter Filling Interface

Click the double quotes to set the parameters. After that, click "Confirm" to complete instruction insertion.

Additional descriptions

Click on "" after "LOAD" to select data types, such as register type (R, SR), string and constant.

Insert EXEC instruction

Please refer to the insertion of EXEC instruction.

Insert UNLOAD instruction

Please refer to the insertion of EXEC instruction.

Insert PAUSE instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Structure Instructions" to switch to the "Structure Instructions" interface, as shown in Fig. 4.25.

Click "PAUSE" to complete the insertion of the PAUSE instruction.

Insert ABORT instruction

Please refer to the insertion of ABORT instruction.

Insert CALL instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Structure Instructions" to switch to the "Structure Instructions" interface, as shown in Fig. 4.25.

Click "CALL" to enter the CALL instruction parameter filling interface, as shown in Fig. 4.32.

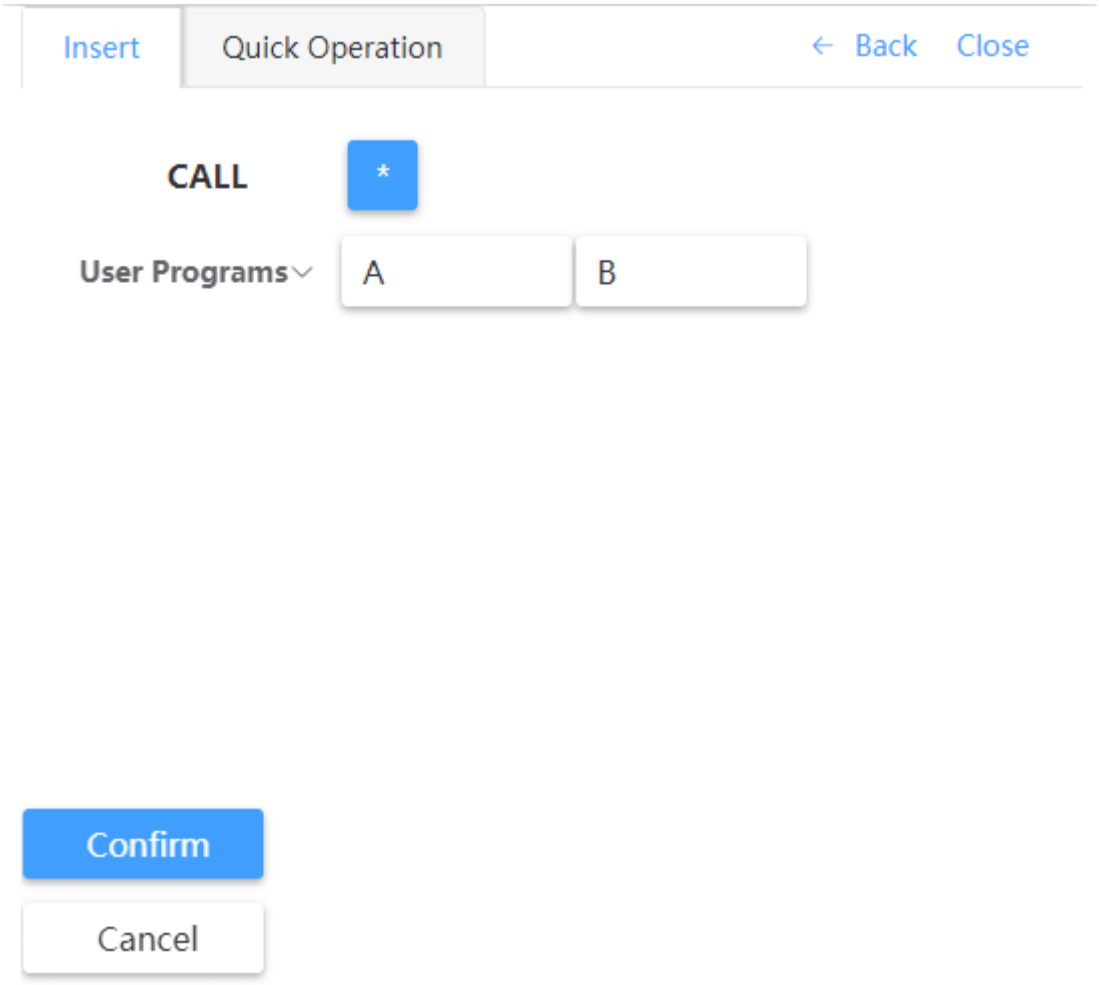


Fig. 4.32 CALL Instruction Parameter Filling Interface

Select the program to be called on the interface shown in Fig. 4.32 and click "Confirm" to complete the insertion of the CALL instruction.

Insert SOCKET OPEN instruction

Click "Insert Instruction" on the program operation interface to enter the instruction

selection interface → Click "SOCKET" to switch to the "SOCKET" interface, as shown in Fig. 4.33.

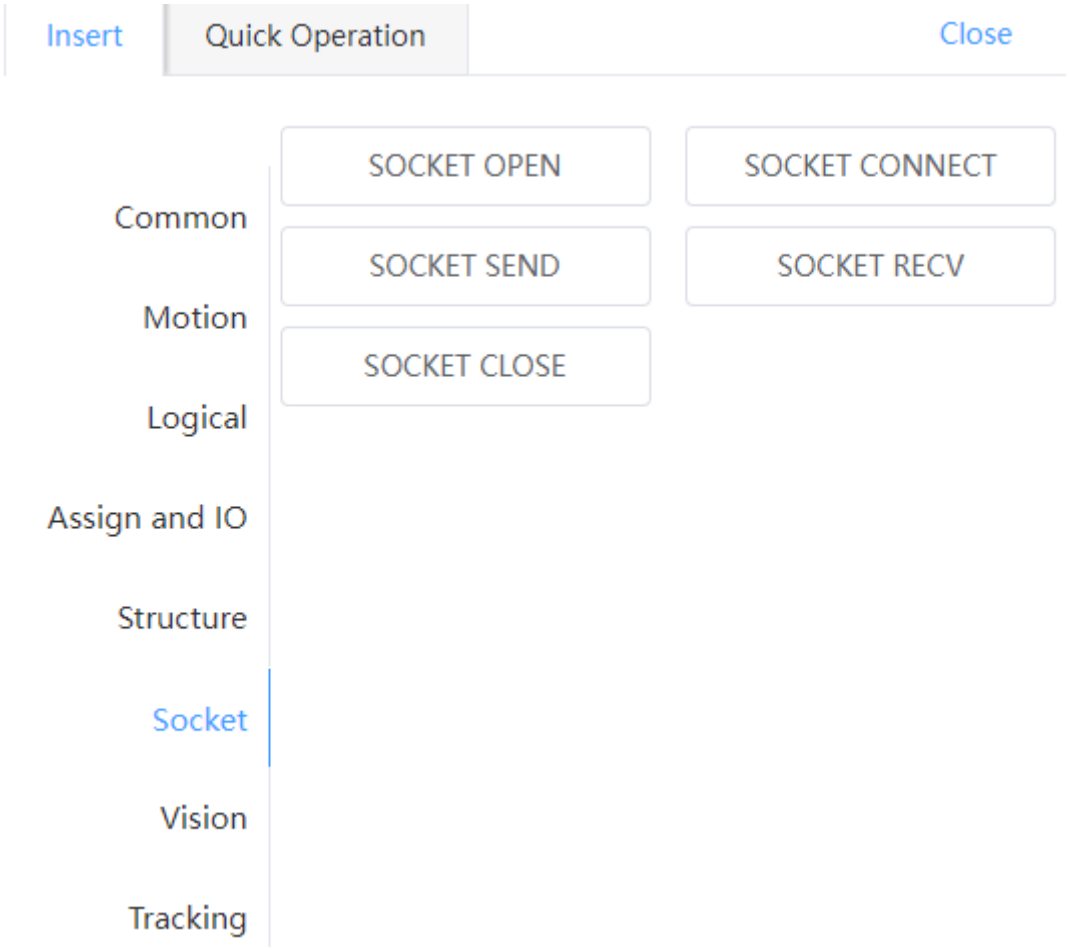


Fig. 4.33 SOCKET Instruction Interface

Click "SOCKET OPEN" to enter the SOCKET OPEN instruction parameter filling interface, as shown in Fig. 4.34.

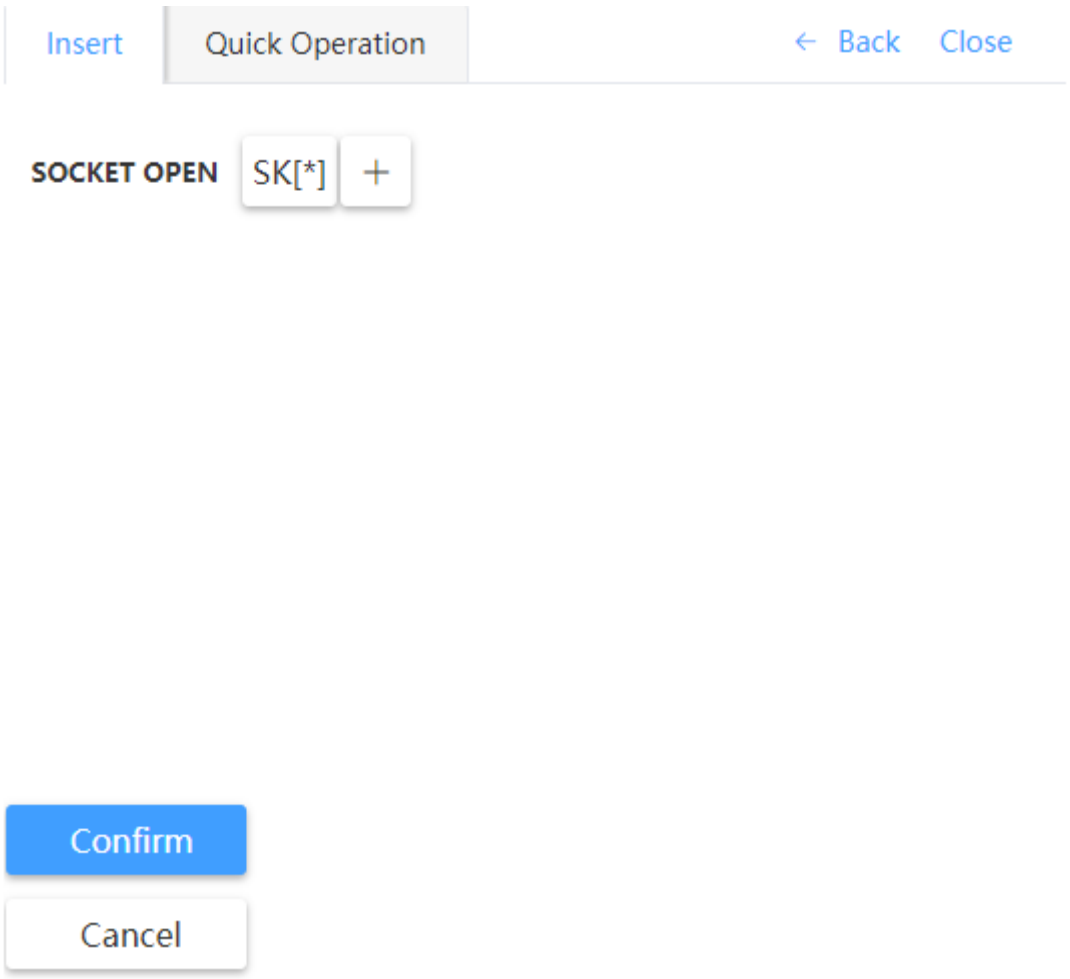


Fig. 4.34 SOCKET OPEN Instruction Parameter Filling Interface

Click "SK[*]" to select the socket device; click "+" to add additional parameters (return parameters). After that, click "Confirm" to complete the insertion of the SOCKET OPEN instruction.

For other SOCKET instructions, please refer to the insertion of SOCKET OPEN instruction.

Insert TF_NO instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Other Instructions" to switch to the "Other Instructions" interface, as shown in Fig. 4.35.

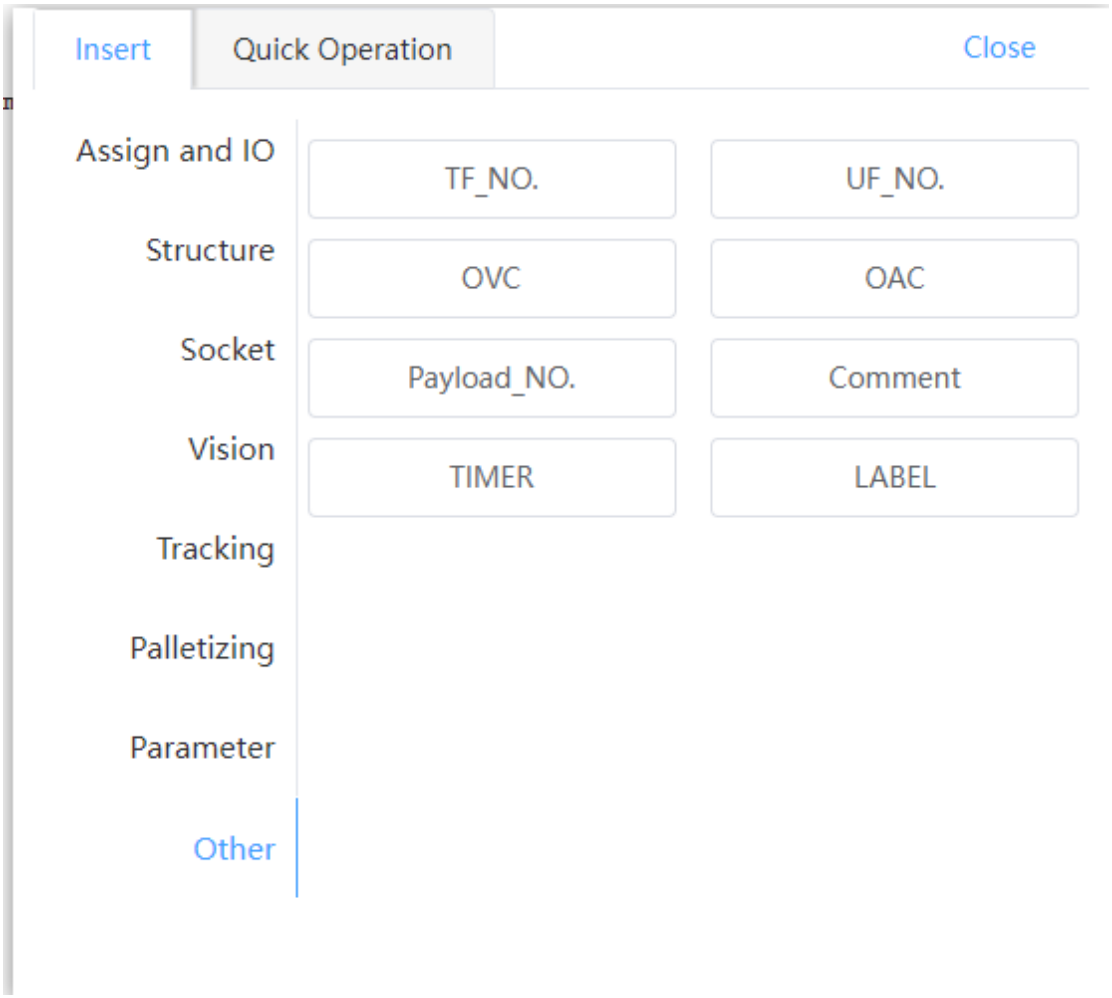


Fig. 4.35 Other Instruction Insertion Interface

Click "TF_NO" to enter the TF_NO instruction parameter filling interface, as shown in Fig. 4.36.

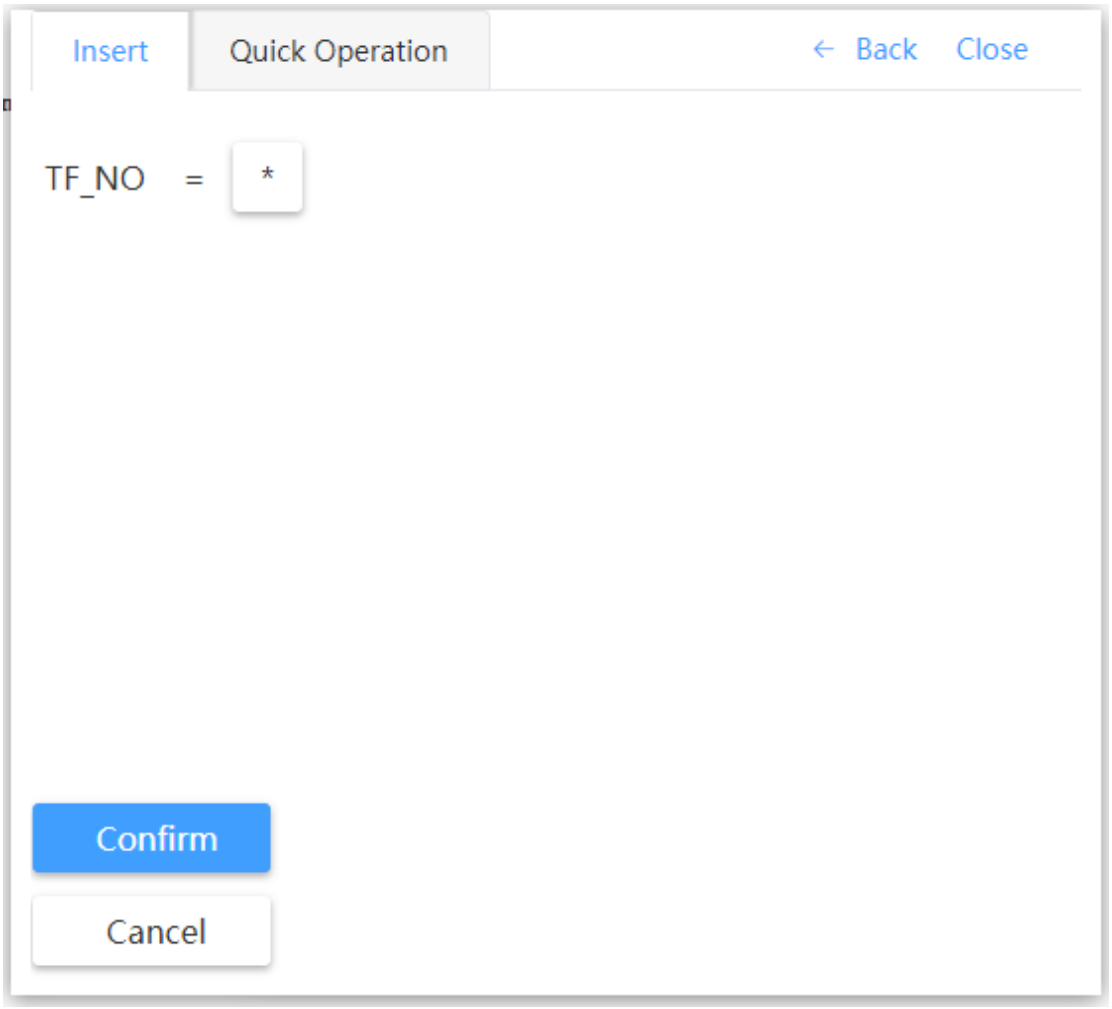


Fig. 4.36 TF_NO Instruction Parameter Filling Interface

Click "*" on the interface of Fig. 4.36 to set the tool coordinate system number. After that, click "Confirm" to complete the insertion of TF_NO instruction.

For UF_ NO & Payload_NO and OVC instructions, please refer to the insertion of TF_NO instruction.

Insert Comment instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Other Instructions" to switch to the "Other Instructions" interface, and slick "Comment" to enter the Comment instruction parameter filling interface, as shown in Fig. 4.37.

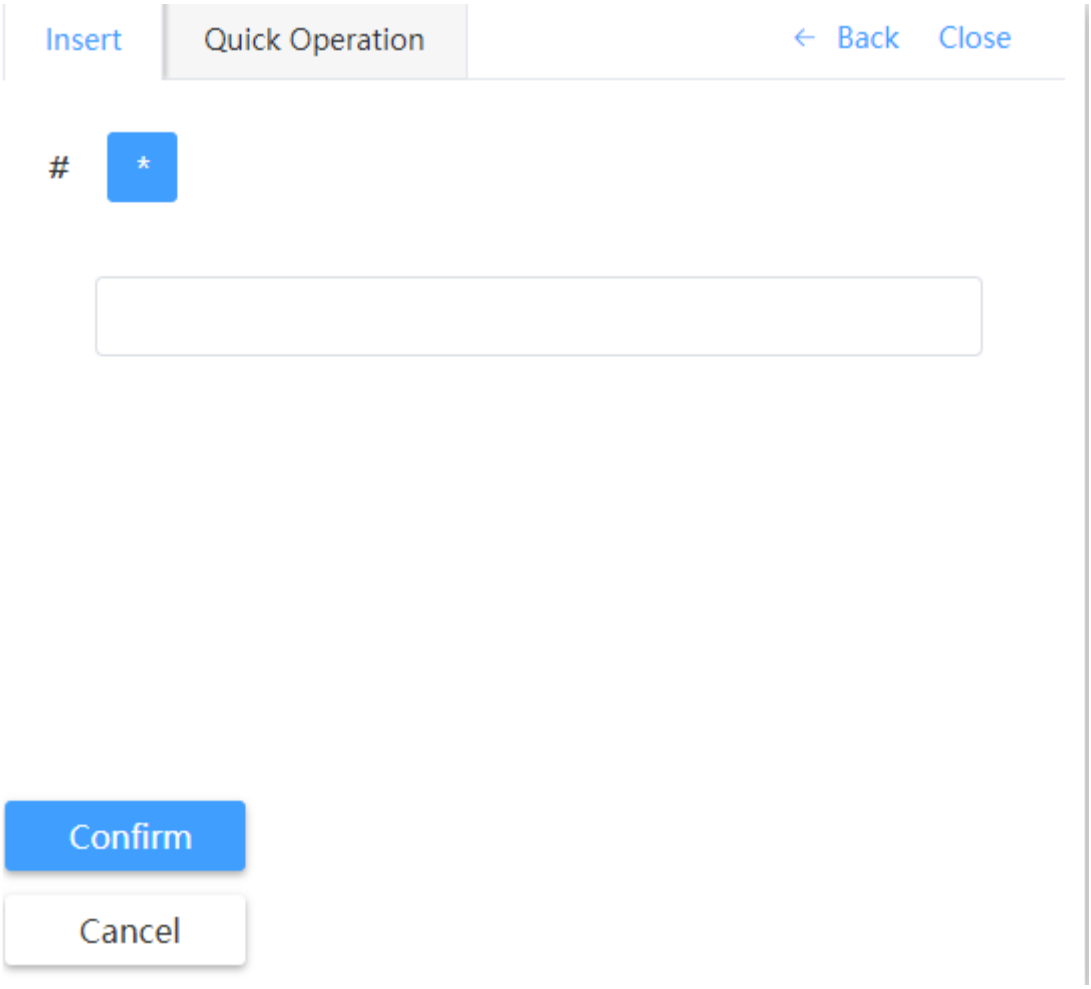


Fig. 4.37 Comment Instruction Parameter Filling Interface

Fill in the comment data in the rectangular bar in Fig. 4.37. After that, click "Confirm" to complete the insertion of the Comment instruction.

Insert TIMER instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Other Instructions" to switch to the "Other Instructions" interface, and slick "TIMER" to enter the TIMER instruction parameter filling interface, as shown in Fig. 4.38.

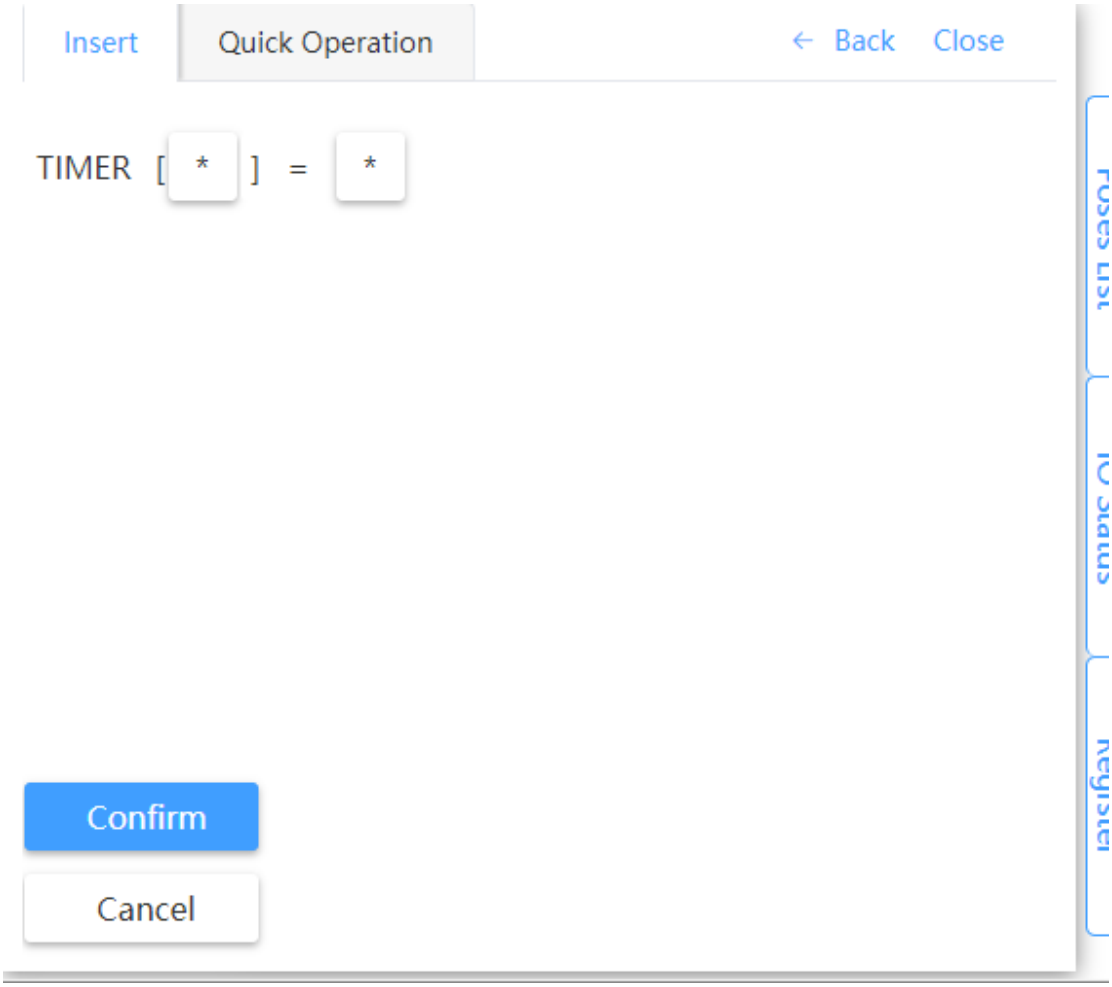


Fig. 4.38 TIMER Instruction Parameter Filling Interface

Click "*" in Fig. 4.38 to set the timer number and value (refer to Section 3.8.6 for timer parameters). After that, click "Confirm" to complete the insertion of the TIMER instruction.

Insert OAC - overall acceleration instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Other Instructions" to switch to the "Other Instructions" interface, and slick "OAC" to enter the OAC instruction parameter filling interface, as shown in Fig. 4.39.

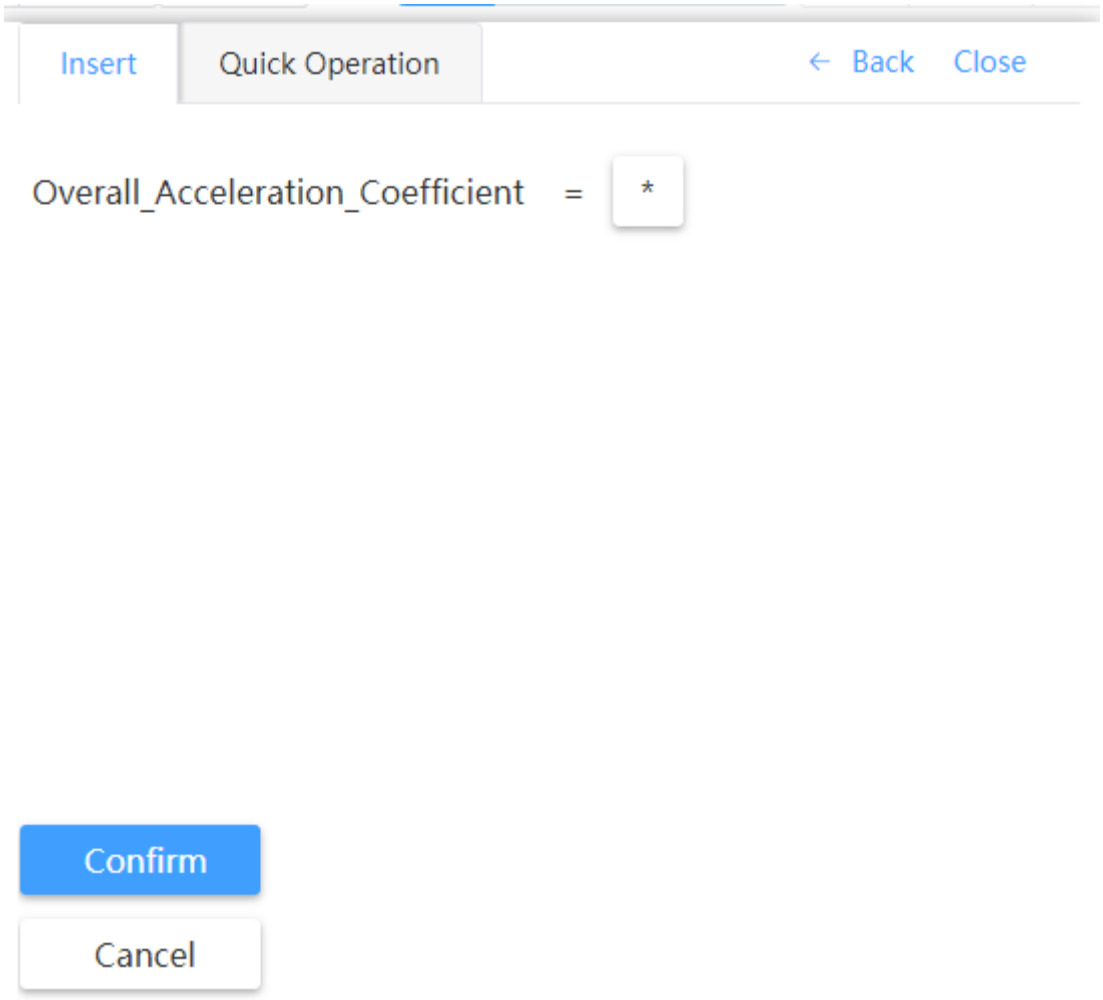


Fig. 4.39 OAC Instruction Parameter Filling Interface

Click "*" in Fig. 4.39 to set the parameters. After that, click "Confirm" to complete OAC instruction insertion.

Insert LABEL instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Other Instructions" to switch to the "Other Instructions" interface, and slick "LABE" to enter the LABE instruction parameter filling interface, as shown in Fig. 4.40.

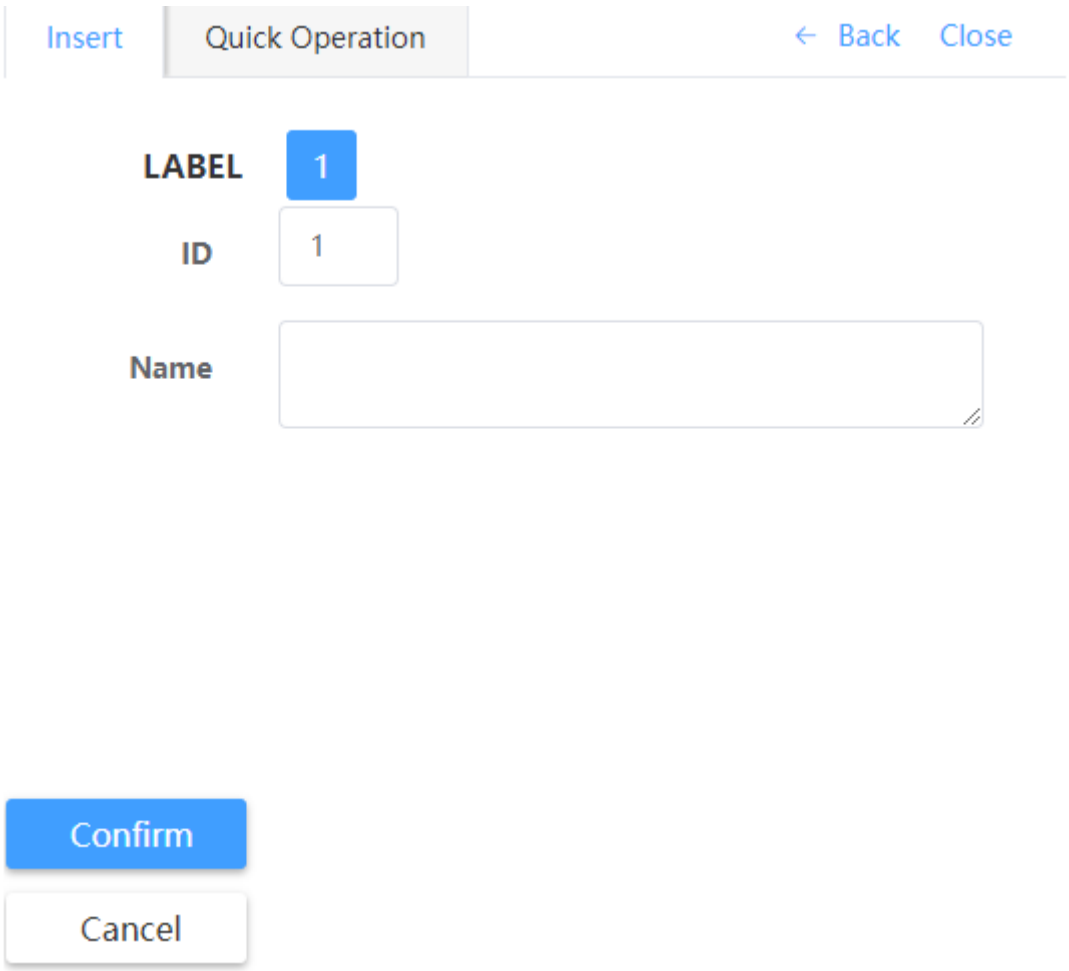


Fig. 4.40 LABEL Instruction Parameter Filling Interface

Set ID number in the parameter filling box after ID in Fig. 4.40. After that, click "Confirm" to complete the insertion of the LABEL instruction. ID is the label number and the label can be named in the name field.

Insert GOTO instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Logical Instructions" to switch to the "Logical Instructions" interface, and slick "GOTO" to enter the GOTO instruction parameter filling interface, as shown in Fig. 4.41.

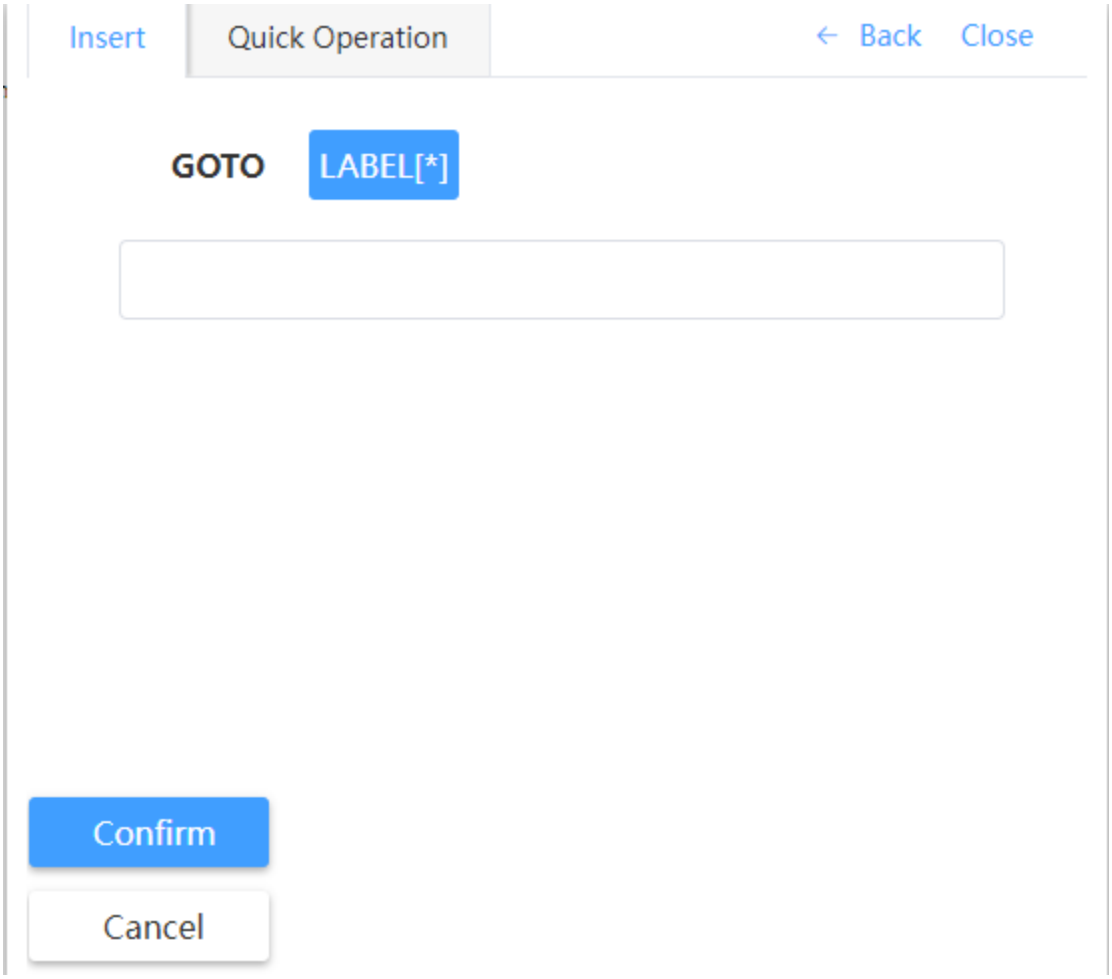


Fig. 4.41 GOTO Instruction Parameter Filling Interface

Fill in LABEL ID in the yellow box of Fig. 4.41. After that, click "Confirm" to complete the insertion of the GOTO instruction.

Insert SKIP CONDITION instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Logical Instructions" to switch to the "Logical Instructions" interface, and slick "SKIP CONDITION" to enter the SKIP CONDITION instruction parameter filling interface, as shown in Fig. 4.42.

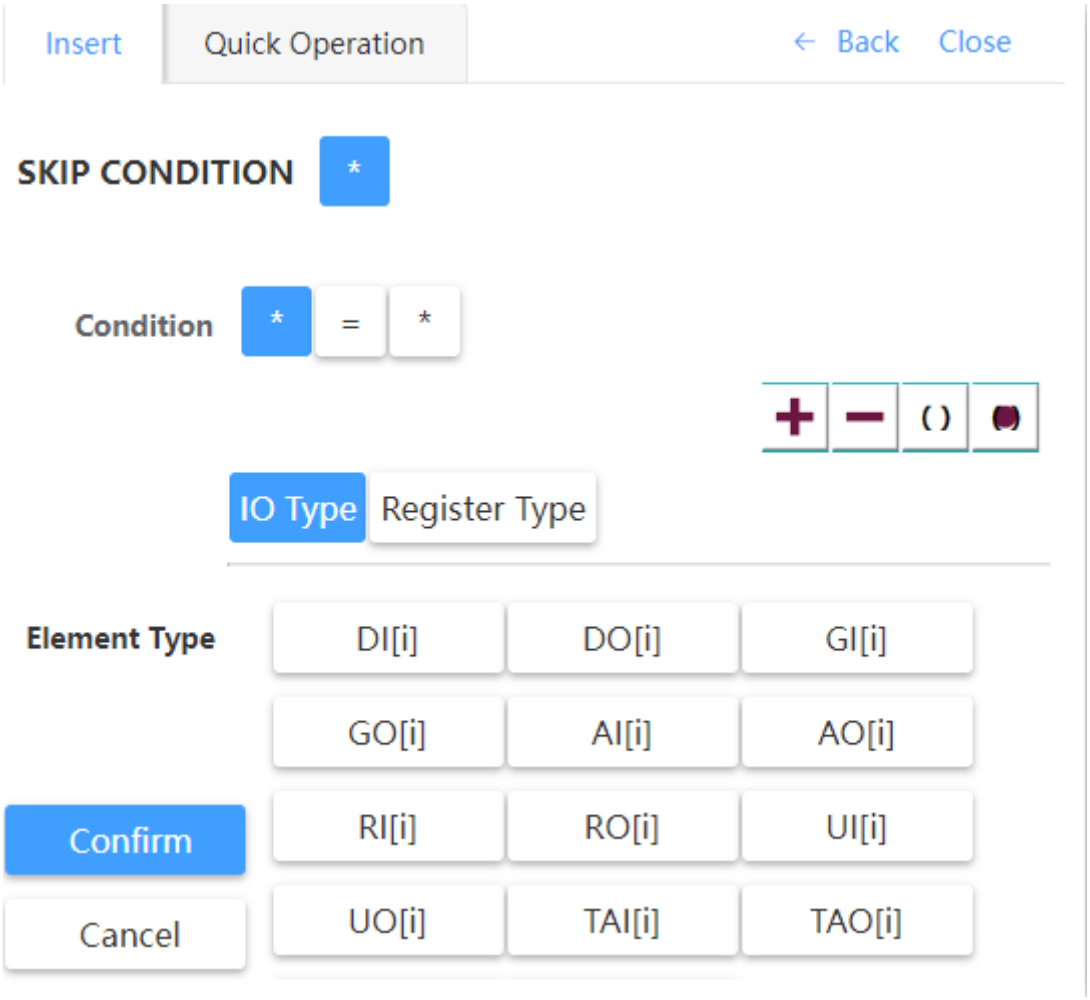


Fig. 4.42 SKIP CONDITION Instruction Parameter Filling Interface

Click Position 1 in Fig. 4.42 to display the conditional parameter setting interface. Optional parameter types include I/O type and register type at Position 2, and register type, I/O type, numerical value and I/O status at Position 3. Click "=" between Positions 2 and 3 to select the desired operator and Boolean operator. After setting, click "Confirm" to complete the insertion of the SKIP CONDITION instruction.

Insert CollisionDetect instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Parameter Setting" to switch to the "Parameter Setting" interface, and slick "CollisionDetect" to enter the CollisionDetect instruction parameter filling interface, as shown in Fig. 4.43.

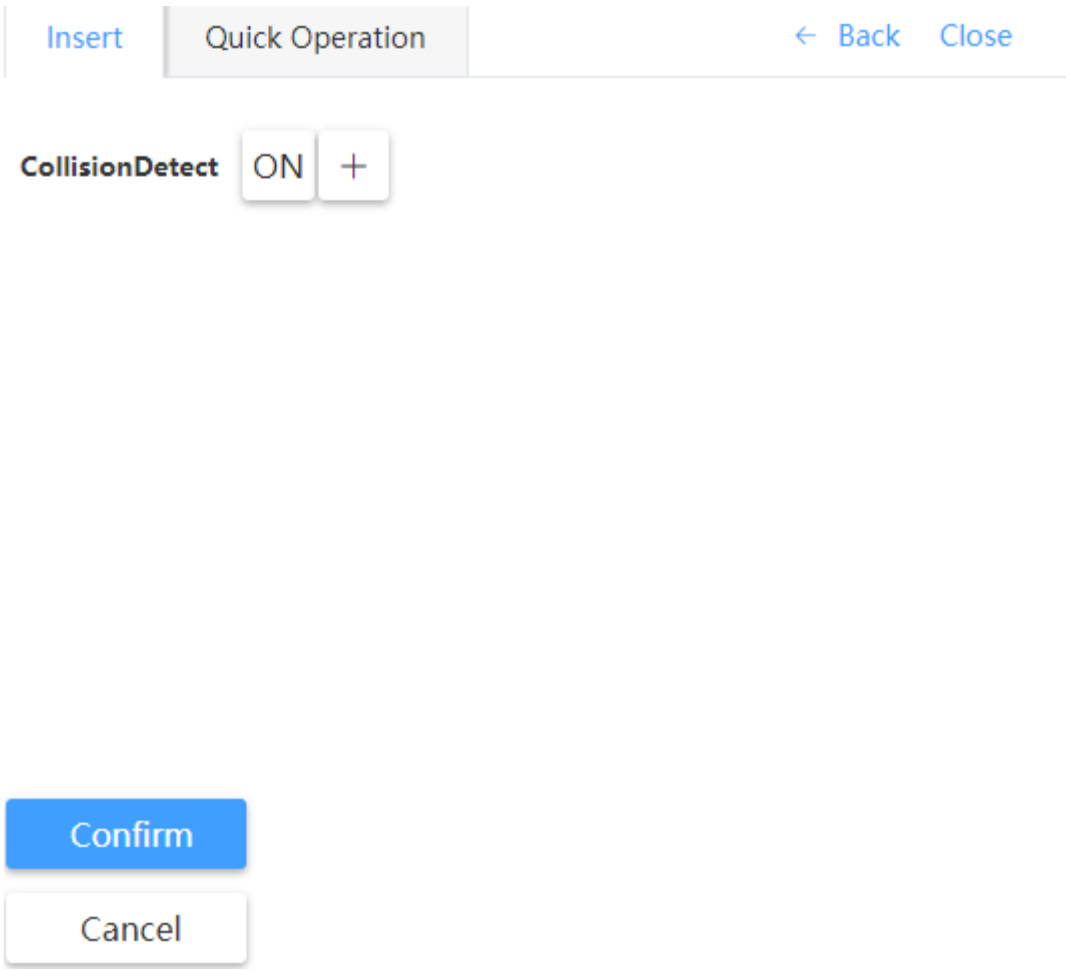


Fig. 4.43 CollisionDetect Instruction Parameter Filling Interface

In Fig. 4.43, Click "+" to add additional parameters Group and Axis and click "ON" to switch the collision detection status to ON or OFF. After setting, click "Confirm" to complete the insertion of the CollisionDetect instruction.

Insert CollisionDetect instruction

Click "Insert Instruction" on the program operation interface to enter the instruction selection interface → Click "Parameter Setting" to switch to the "Parameter Setting" interface, and slick "CollisionRange" to enter the CollisionRange instruction parameter filling interface, as shown in Fig. 4.44.

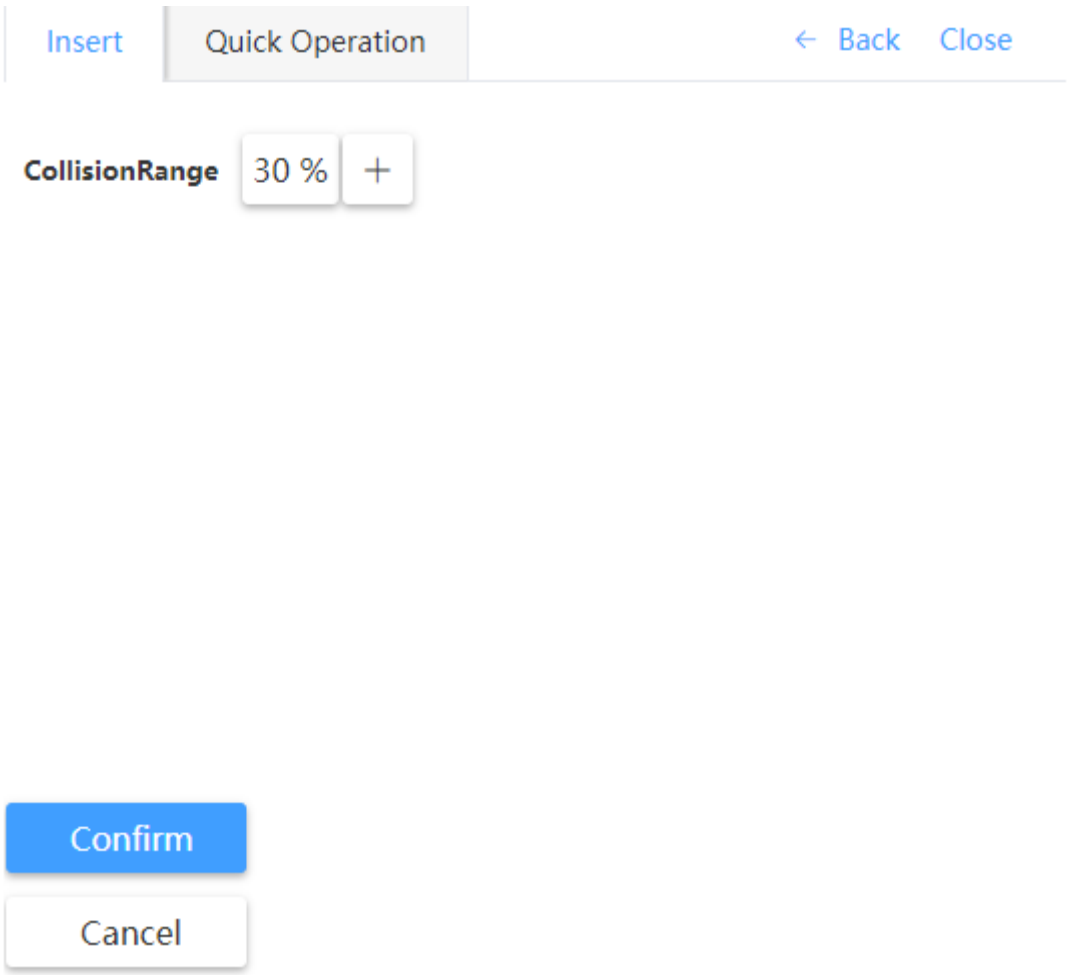


Fig. 4.44 CollisionRange Instruction Parameter Filling Interface

In Fig. 4.44, Click "+" to add additional parameters Group and Axis and click Position 1 to set the deviation range value. After setting, click "Confirm" to complete the insertion of the CollisionRange instruction.

4.2 Program setting

Click "Menu Button" → "Program" to enter the interface as shown in Fig. 4.45. This is settings: "Program Launch Mode Settings".

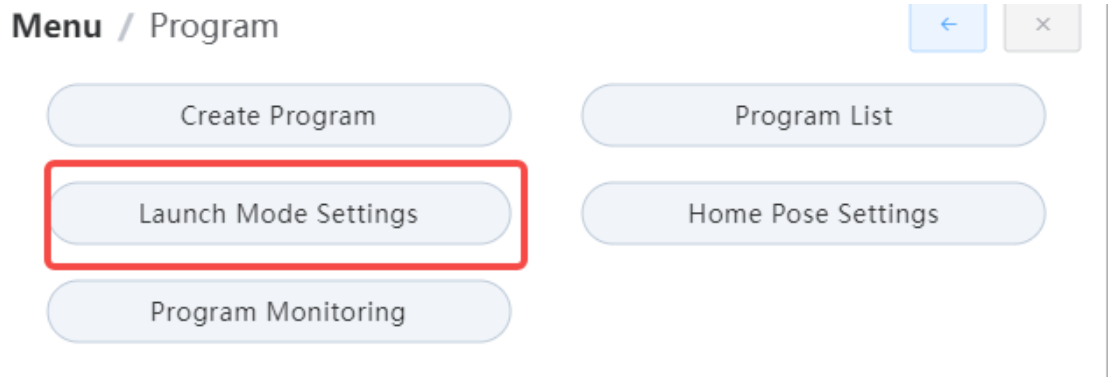


Fig. 4.45 Program Menu Window

4.2.1 Setting of program launch mode

When the operation mode is the auto mode, the program has four launch modes: Local Trigger, MPLCS, Macro Trigger and MPLCS Simple Trigger.

The user can click "Menu Button" - "Program" - "Program Launch Mode Setting" in sequence to enter the program trigger mode setting interface as shown in Fig. 4.47. Specify one of these four trigger modes to launch the robot program. The default mode of the system is “Local Trigger”.

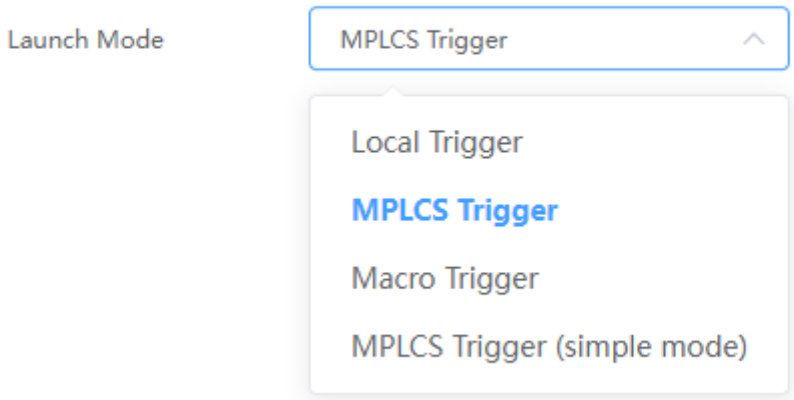


Fig. 4.47 Program Launch Mode Setting Interface

Local Trigger

Local Trigger is that the user selects and opens the desired program (i.e. enter the program editing interface), and clicks the Start button on the TP, and the program runs continuously from the first line.

“Local Trigger” is applicable to all types of programs.

Preconditions for “Local Trigger”

- The operation mode of the robot is the "auto mode".
- The running status of the robot is "On-Standby".
- The program execution status is “Finished”.
- The “program launch mode” is set as Local Trigger.
- An executable program has been loaded (open it to enter the program editing interface).

MPLCS

MPLCS is to trigger the main program by its trigger number.

The trigger number of the main program is in decimal.

The process of MPLCS is as follows:

6. When "MPLCS" is enabled, the system receives the Selection Strobe signal (UI[6]) from the upper computer to maintain a high level.
7. As long as the UI[6] signal remains at a high level, the system may read the level states (0/1) of six UI signals from UI[8] to UI[13] to form a 6-bit binary number.
8. The system may also provide feedback on the 6-bit binary number read by itself through 6 UOs (Selection Confirm) and maintain a high-level signal of Selection Check Request (UO[6]) to the upper computer for checking the request.
9. After receiving UOs [6], the upper computer checks if the outputs of six UOs are consistent with the given values. If consistent, it sends an MPLCS Start to the robot system (UI[7]) and disconnects it after a certain time (this time must be longer than 100ms).
10. After obtaining UI[7], the robot system converts the 6-bit binary number into a decimal number and follows this number to find main programs with the same main program number. (If not found, it reports an alarm event).
11. It triggers the main program if found. Meanwhile, it sends the UO[7] (MPLCS Start Done) to the upper computer.
12. Then, the upper computer turns off UI[6] and UI[8]-UI[13] signals.
13. Once detecting that the UI[6] signal disappears, the robot system resets UO [6], UO [8] - UO [13] and UO [6] signals.

Caution

At present, considering the I/O limit of the standard I/O board, the UI Program Selection and UO Selection Confirm used for MPLCS start only need 6 bits to form

a 6-bit trigger code.

The specific time sequence can be referred to Fig. 4.48:

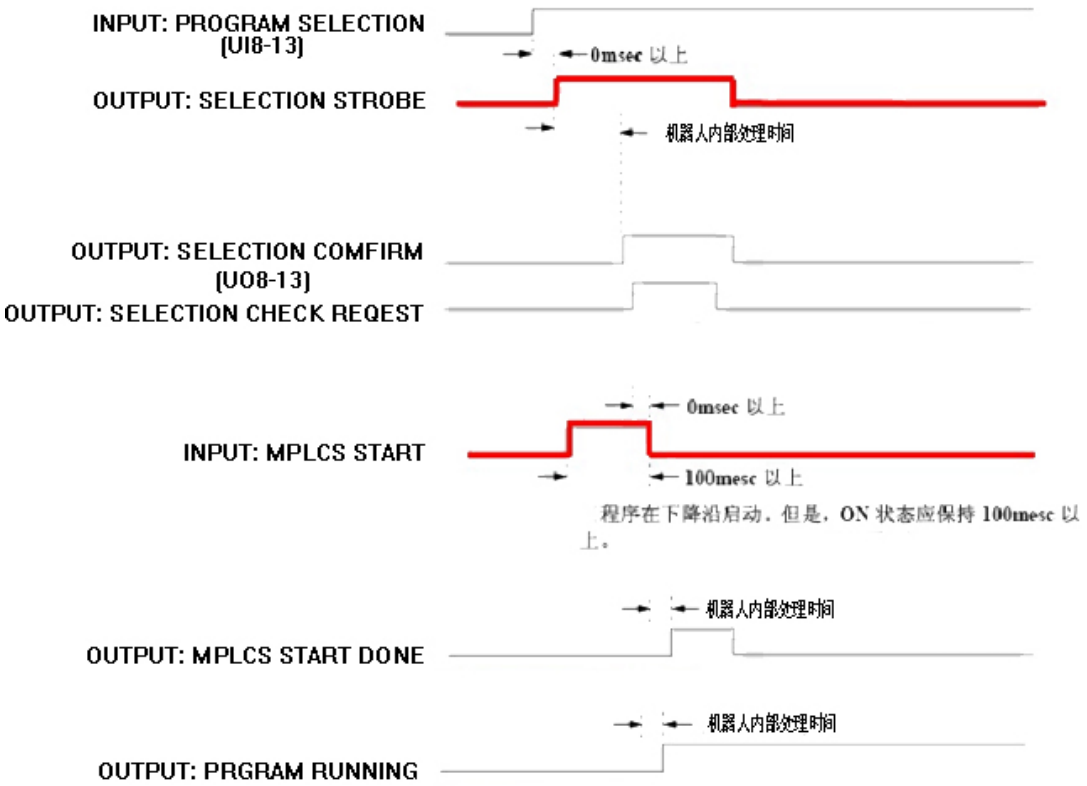


Fig. 4.48 MPLCS Timing Diagram

“MPLCS” is applicable to main programs with the attribute “Main”.

Preconditions for “MPLCS”

- The operation mode of the robot is the "auto mode".
- The running status of the robot is "On-Standby".
- The program execution status is “Finished”.
- The “program launch mode” is set as MPLCS.

MPLCS Simple Trigger

The process of MPLCS mode is relatively complex. In many scenarios, such a complex process is unnecessary (ordinary integration users also do not want too complex timing logic). So, the MPLCS Simple Trigger mode is provided. Compared to the complete mode, the simple mode simplifies the process of feeding back the received UI[8] - UI[13] signals for external confirmation.

- The robot can find the START signal when it is in the "MPLCS Simple Trigger Mode" and the corresponding trigger conditions are met.
- It is considered necessary to start the program when the START signal shows a

falling edge (from high level to low level). Then, UI[8] 1~UI[13] are read and combined into a decimal number as the program code. It tries to find the corresponding program to start and reports an alarm if not found.

- Other mechanisms, e.g. pause and stop, are the same as the complete mode.

MPLCS Simple Trigger Mode is applicable to main programs with the attribute “Main”.

Preconditions for “MPLCS Simple Trigger Mode”

- The operation mode of the robot is the "auto mode".
- The running status of the robot is "On-Standby".
- The program execution status is “Finished”.
- The “program launch mode” is set as MPLCS Simple Trigger Mode.

Additional descriptions

1. When the program type is selected as Main, the functions of “program start point” and “program number” are additionally shown on the program property interface. The programmer is allowed to choose whether to disable the "Program Start Point" function as shown in Fig. 4.49. Please refer to Section 4.3 for details.

2. The program number ranges from 1 to 255. However, since only 6 system signals are used to call Main programs and the maximum decimal number composed of 6 signals is 64, the maximum program number for external calls to the Main program is 64.

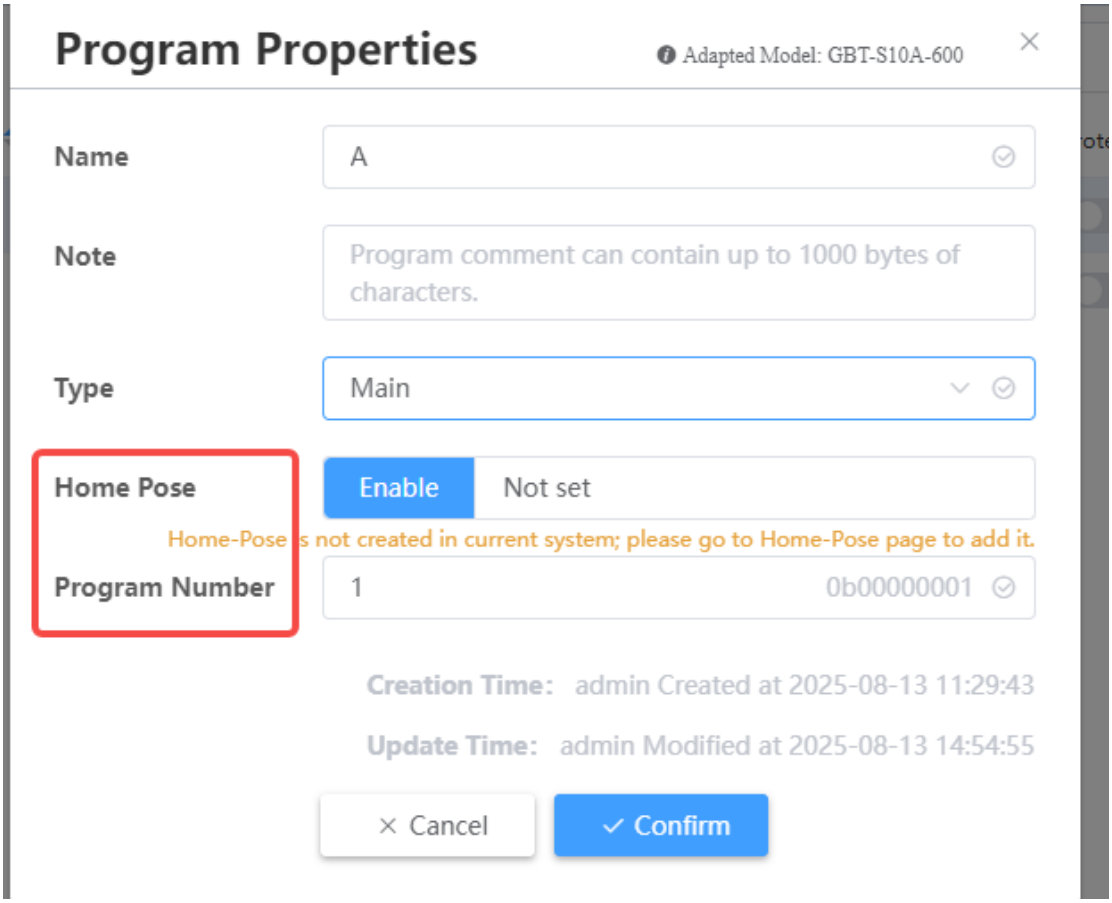


Fig. 4.49 Program Property Window

3. None - general programs: There are no special restrictions or distinctions. In the auto mode, it cannot be started by any means other than Local Trigger mode (manually pressing the start button). However, it can be called by other programs.

4. Macro - Macro program: During execution, a macro program can be called through macro start.

5. Introduction to macro program launch mode: The user can set it in the special program settings (macro settings) interface and each macro program corresponds to a DI one by one. A macro program can be started through pre-set I/O.

6. To start the main program through "MPLCS" or “MPLCS Simple Trigger", the TP mode selector must be in the auto mode.

4.2.2 Program start point

The user can set the program start point (i.e. home point) for the automatically running program in order to prevent unpredictable danger or injury when the program starts automatically in the auto mode, for the robot's current position is not at the

program's start point planned by the programmer and the first path planned is different from the debugging process and does not meet the programmer's expected path to the first target point. Successively click "Menu Button" → "Program" → "Start Point" to enter the screen as shown in Fig. 4.50.

The elements of program start point include:

- J1-6 is the joint information for this pose. Unit: angle.
- ID: The ID number of the pose can be modified.
- Comment: support 128 bytes.
- Name: support 32 bytes.
- Float value: The positive/negative floating range during verification of Home point. Unit of display layer: angle. The allowed range is 999.999 with a default value of 0.5.

The user can add or delete the home point of the program and record the current point as the home point by clicking the "Teaching Log" button at the bottom of the page. Click the "Edit" button to change the value and floating value of the robot axis. In order to quickly teach the robot to the recorded program home point, the user can use the "Move to Point" button to move the robot to the home position.

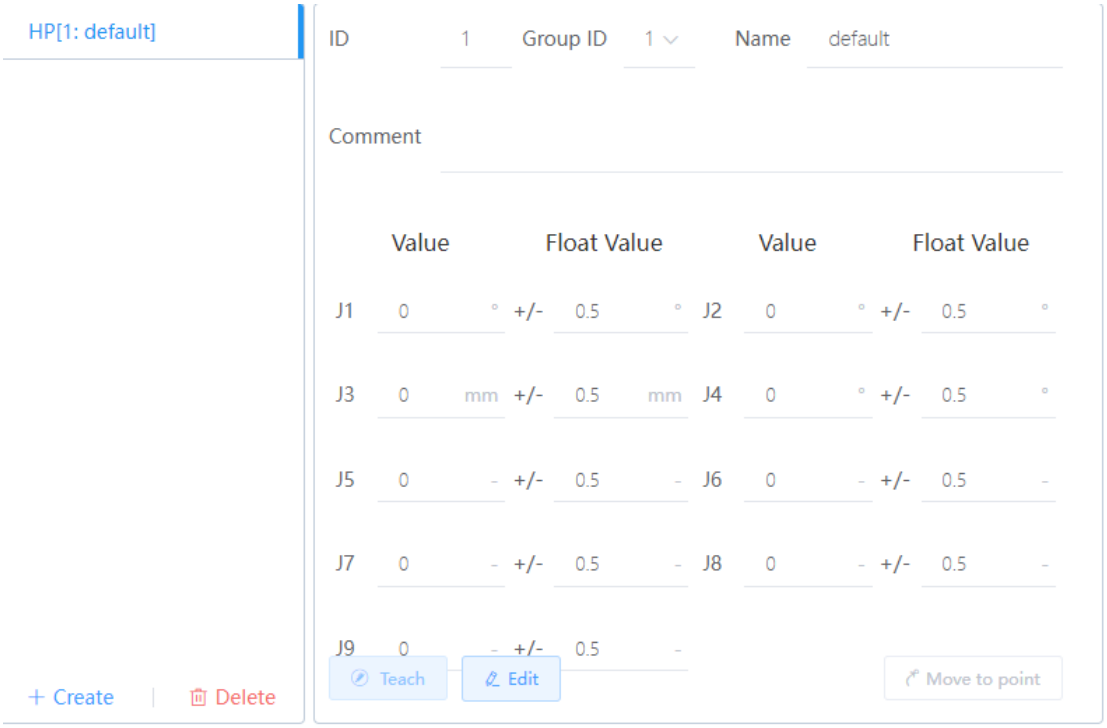


Fig. 4.50 Program Home Point Window

4.3 Execute programs

4.3.1 Trigger the program in manual mode

Executing a program in manual mode is also known as "program debugging and execution". The robot program can be executed according to the following steps:

1. Set the TP mode selector to the manual mode (manual maximum speed or manual limit speed).

2. Open the program to be executed (refer to Section 4.1.4 for program opening).

3. Choose how to execute the program (continuous, single step & positive sequence, single step & reverse sequence)

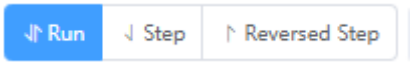


Fig. 4.51 Selection of Program Execution Modes

4. Select the statement from which line to execute the program. The selected statement is highlighted as shown in the following figure.

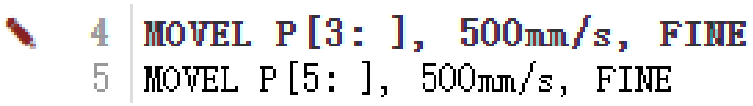


Fig. 4.52 Program Window

5. Click the "Arrow to Cursor". Execution will start from the selected line when the program is triggered.

6. Press the Enable button on the back of TP and keep it at the middle gear.

Fig. 4.53 Enable Button

7. Press the "Reset" button on TP to clear alarm and other information and activate the robot (observe if the TP status lamp "Servo ON" turns green).

8. Press the Start button to run the program.

4.3.2 Execute the program in auto mode

The user can only execute programs continuously in auto mode.

Moreover, in the auto mode, the program must be executed through the "automatic trigger mode" specified by the system settings or resumed from the pause status.

Only when the status is in "ON-standby" in the robot's running status bar (and the program status is not in "Paused" or "Running") can it be started through the automatic program trigger mode.

In the auto mode, simply press the "Start" button or send a "Restart" UI signal to the controller to resume the execution of a paused program.

Multiple ways can be adopted to start the program in the auto mode. Please refer to Section 4.2.2 for details.

The robot program can be executed according to the following steps:

9. Open the program to be executed (Do not open the program when it is called

through the program number or macro instruction. The corresponding program should be opened only during manual execution.)

10. Set the mode selector to the auto mode (A).

11. Press the "Reset" button on TP or send a UI signal of "Restart" to the controller to clear alarm and other information and activate the robot (observe if the TP status lamp "Servo ON" turns green).

12. Start the corresponding program according to the set program trigger mode (directly press the TP Start button, send the DI trigger signal with the specified macro or launch the program according to the timing sequence specified by the main program trigger mode).

4.3.3 Program pause and abort

Pause

The robot's program is paused for some reason during operation. After pause, the program arrow stops at the executing program line.

The following actions may cause pauses:

- When the program is running, the user clicks the Pause button; or the robot receives an external Pause signal.
- There will be alarms causing "pause" behavior, such as "STOP", "Pause", "Servo 1" and other level alarms. When a program is executed in single step or reverse order, it may enter a pause state after each instruction is executed.
- A “Pause” instruction is executed during program execution.

Caution

The alarm levels of STOP and Servo 1 can cause the abortion of the motion. However, they only cause the program to pause. For example, triggering the safety door signal can cause a "STOP" alarm. Although the motion stops instantly, the program turns to a pause state.

Abort

There are two abort states: program abort and motion abort.

Program abort

Program abort: The robot's program can be paused for some reason during operation. After abort, the system maintains the sequence number of the instruction line just being executed. Please note: normal completion of the program or execution of abort instruction is also considered as a special case of program abort.

If a subprogram called by a higher-level program is aborted during execution, the information returned to the higher-level program may be lost and the subprogram cannot be independently started and executed again or return the information to the higher-level program after execution.

The following actions may cause program abort:

- The user clicks the Pause button twice. The interface will pop up a box indicating whether to stop current program. Click “Confirm”.
- An alarm possibly causing "program abort" occurs.
- AN “End” instruction is executed during program execution.
- A “Abort” instruction is executed during program execution.

Motion abort

Motion abort: The general motion abort of the robot refers to power-off of the servo motor and application of the brake. It is not required to distinguish whether the robot is moving, decelerating or stationary at this time, but the servo is powered off and the brake is applied directly.

Generally, motion abort may not necessarily cause program abort, but may result in program pause (e.g. when the e-stop button is pressed); program abort does not necessarily trigger motion stop as well. It is possible that the program has ended and the robot is still running and waiting for the instruction to execute another program.

The following actions may cause motion abort:

- The user clicks the Pause button twice. The interface will pop up a box indicating whether to stop current program. Click “Confirm”.
- An alarm possibly causing "motion abort" occurs.
- A “Abort” instruction is executed during program execution.

4.3.4 Resume after pause

When the program execution status of the robot is "Pause", the user can click the Start button to resume the program start if the robot's running status is "On-Standby" and no alarm above "Pause" level is valid.

4.4 Program monitoring

Successively click "Menu Button" → "Program" → "Program Monitoring" to enter the program monitoring screen.

The program monitoring screen is as shown in the figure below:

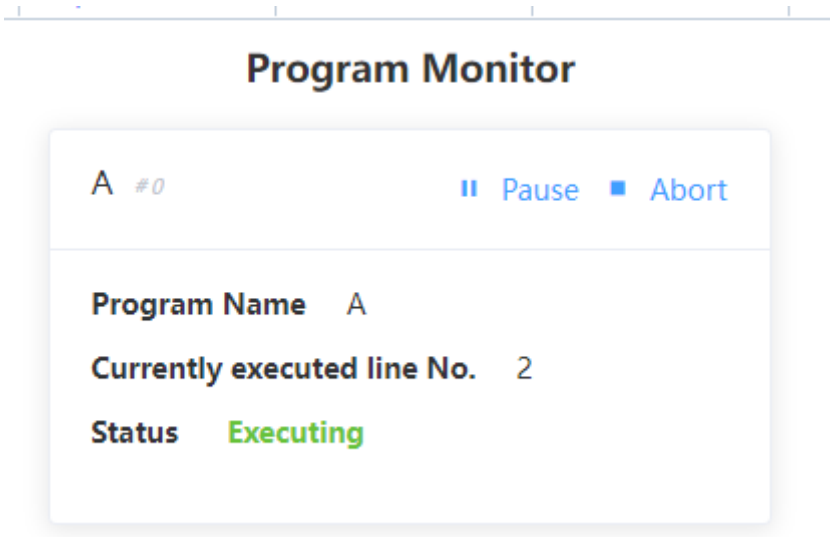


Fig. 4.54 Program Monitoring Interface

Click to switch between "Run" and "Pause". “Run” means to execute the program and “Pause” means to pause the program. Click to stop the program.

Status:

- Running
- Pause
- Abort (then, the program will disappear)

Caution

For Python script programs, Pause is invalid, while Abort is valid.

5. Status display

5.1 Register management

The registers used by the Agilebot robot include number register (R), motion register (MR), pose register (PR), string register (SR), socket register and Modbus register.

5.1.1 Number register

The number register (R[i]) is displayed and set on the number register screen.

Successively click "Menu Button" → "Data" → "Number Register" to enter the number register setting screen as shown in Fig. 5.1.

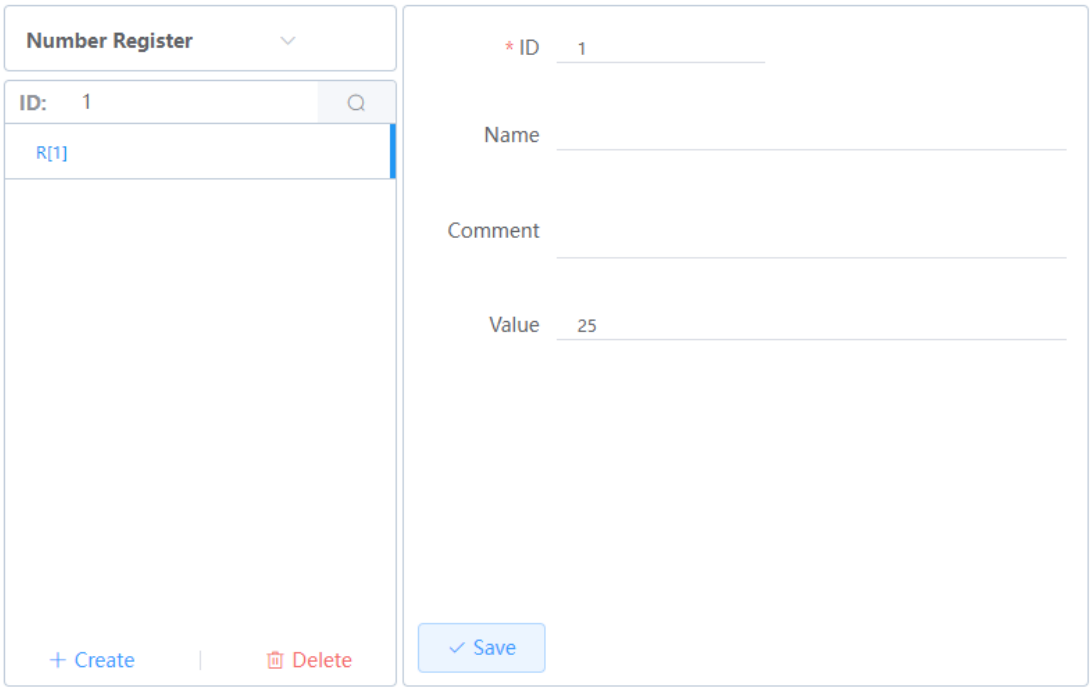


Fig. 5.1 Number Register Setting Interface

It is allowed to create or delete registers on the number register setting interface.

Description of number register parameters

- ID: Register number.
- Name: Appoint the name of number register here. The name can be composed of characters less than 31 bytes, but is not allowed to contain: #$%^&*()@+-=/\,:;'"'"'|<>~{}
- Comment: Add a comment to the number register here. The comment may contain up to 128 bytes of characters.
- Value: Set the register value here. Range: ±99999999.999.

After setting of the above parameters, click “Save”.

5.1.2 Motion register

The motion register (MR[i]) is displayed and set on the number register screen.

Successively click "Menu Button" → "Data" → "Motion Register" to enter the motion register setting screen as shown in Fig. 5.2.

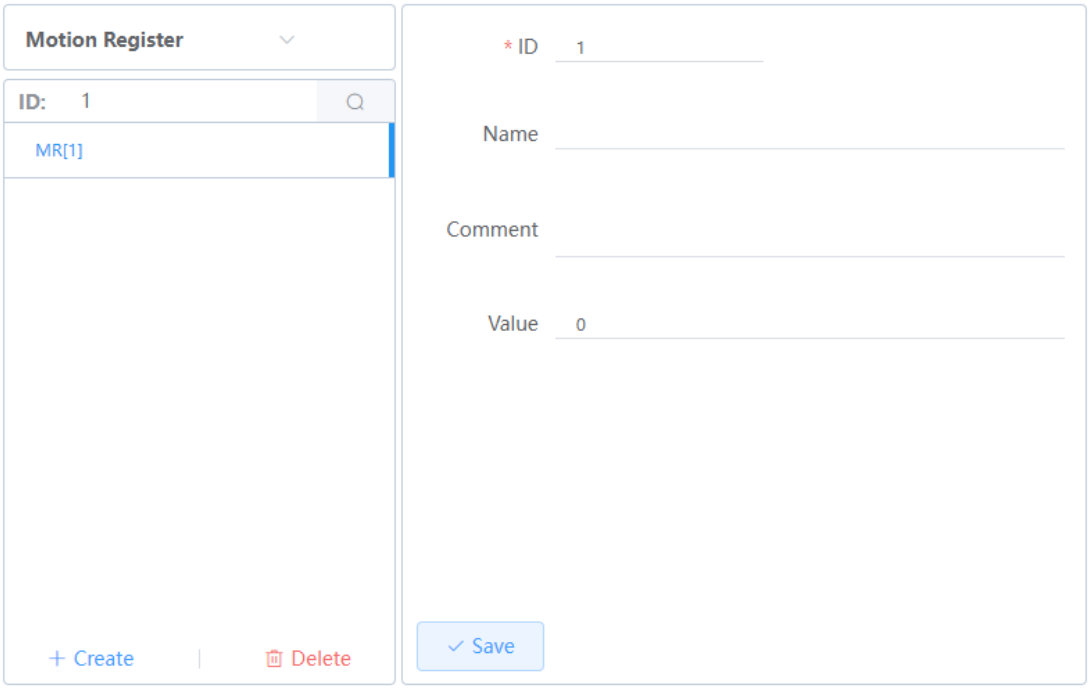


Fig. 5.2 Motion Register Setting Interface

It is allowed to create or delete registers on the motion register setting interface.

Description of motion register parameters

- ID: Register number.
- Name: Appoint the name of motion register here. The name can be composed of characters less than 31 bytes, but is not allowed to contain: #$%^&*()@+-=/\,.:;'"|<>~{}
- Comment: Add a comment to the motion register here. The comment may contain up to 128 bytes of characters.
- Value: Set the register value here. Range: an integer within the range of ±99999999.

After setting of the above parameters, click “Save”.

5.1.3 Pose register

The pose register (PR[i]) is displayed and set on the pose register screen.

Successively click "Menu Button" → "Data" → "Pose Register" to enter the pose register setting screen as shown in Fig. 5.3 and Fig. 5.4.

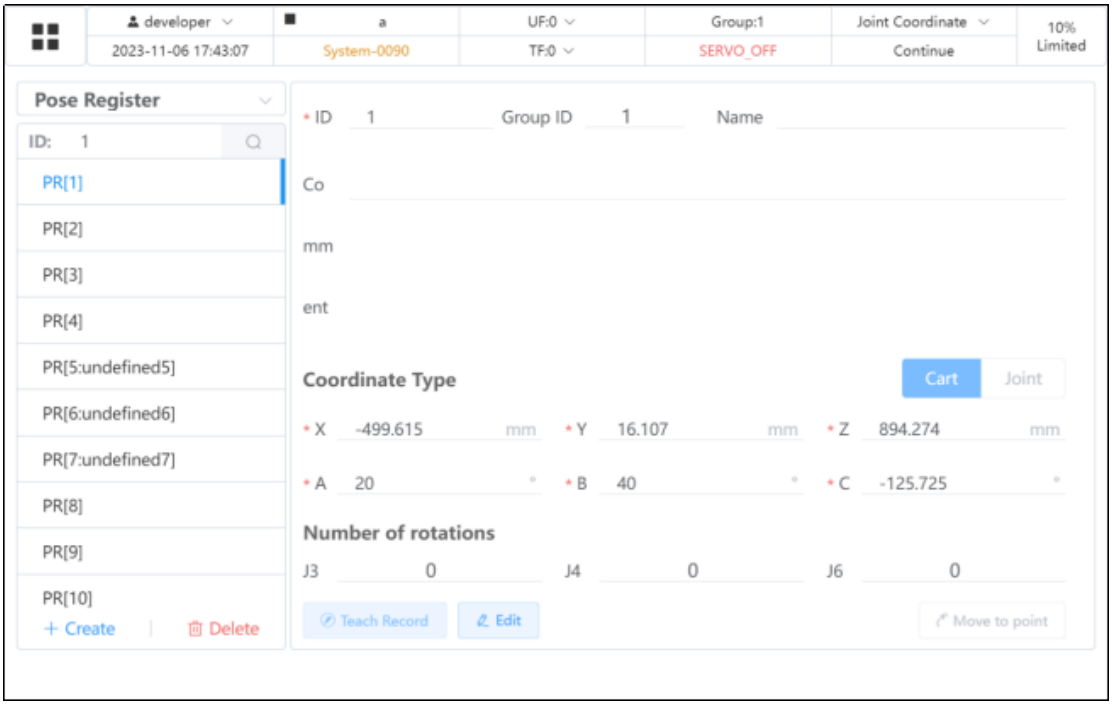


Fig. 5.3 Pose Register Setting Interface (Cartesian)

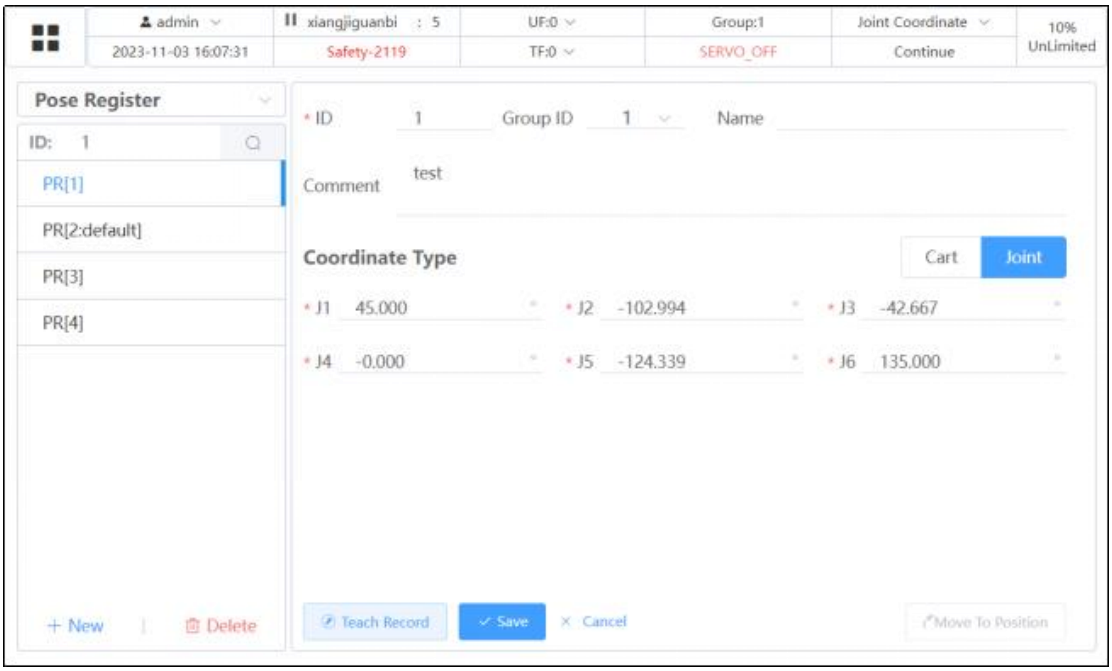


Fig. 5.4 Pose Register Setting Interface (Joint)

It is allowed to create or delete registers on the pose register setting interface; perform data switching between Cartesian coordinate system (Fig. 5.3) and joint coordinate system (Fig. 5.4) by .

Description of pose register parameters

- ID: Register number
- Group ID: "1" represents the robot body, while other values represent external axes. Currently, only "1" is supported for body representation.
- Name: Appoint the name of motion register here. The name can be composed of characters less than 31 bytes, but is not allowed to contain: #$%^&*()@+-=/\,.:;'"""|<>~{}
- Comment: Add a comment to the motion register here. The comment may contain up to 128 bytes of characters.
- Value: Set the register value here. Range: an integer within the range of ±99999999.
- The values (X, Y, Z, A, B, C) in the Cartesian coordinate system represent poses of the specified TCP in the specified user/base/world coordinate system.
- The values (J1, J2, J3, J4, J5, J6) in the joint coordinate system represent the angles of each robot joint.

Writing of coordinates

- Current robot coordinates can be recorded by the "Teach Record" button (this value does not include the information of user and tool coordinate systems). So, when the PR pose register is used in the program, it is necessary to use UF_No and TF_No instructions to specify the user or tool coordinate system in advance. Spatial positions represented by the PR register may not necessarily be the same under different user or tool coordinate systems. Current coordinate system is used if no coordinate system is specified in the program.
- It is also allowed to fill in coordinate values by clicking the "Edit" button. After that, click "Save" to complete the filling.

In the manual mode and with the servo powered on, click "Move to Point" on the pose register interface. Then, the robot will move to the point in the pose register.

5.1.4 String register

The string register (SR[i]) is displayed and set on the string register screen.

Successively click "Menu Button" → "Data" → "String Register" to enter the string register setting screen as shown in Fig. 5.5.

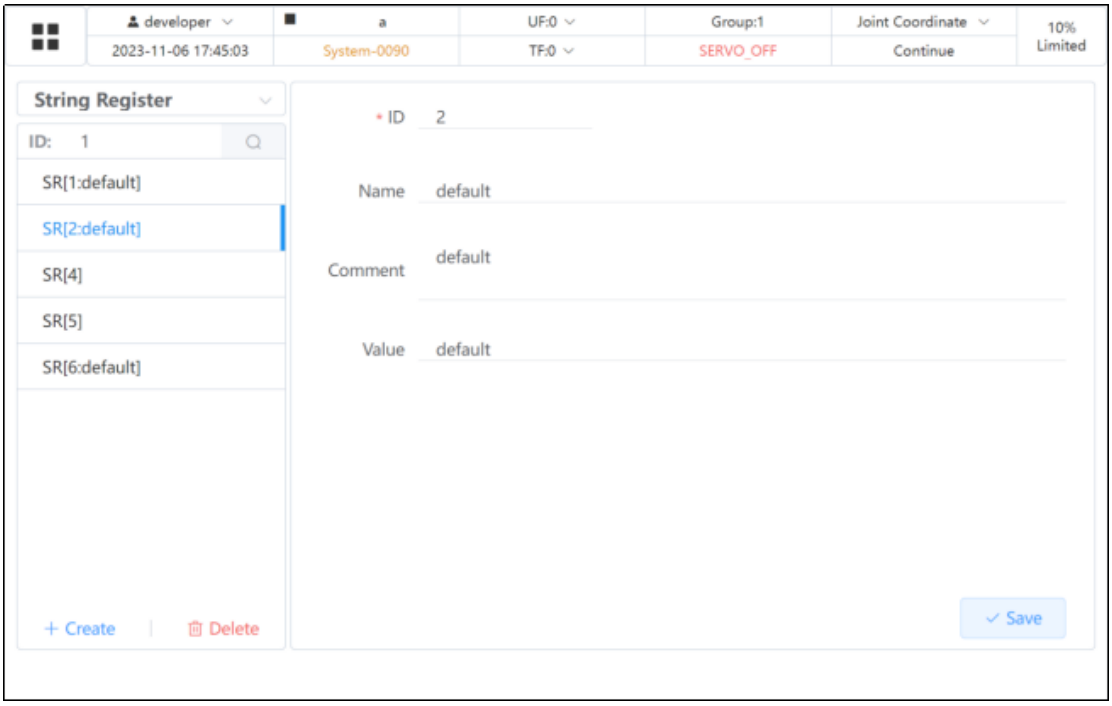


Fig. 5.5 String Register Setting Interface

It is allowed to create or delete registers on the string register setting interface.

Description of string register parameters

- ID: Register number.
- Name: Appoint the name of string register here. The name can be composed of characters less than 31 bytes, but is not allowed to contain: #$%^&*()@+-=/\,.:;'"'"'|<>~{}
- Comment: Add a comment to the string register here. The comment may contain up to 128 bytes of characters.
- Value: Set the register value here. It can contain 255 bytes at most.

After setting of the above parameters, click “Save”.

5.1.5 Socket register

It is used in conjunction with socket instructions (see Section 3.8.10 for socket instructions).

Overview

The robot acts as Socket Client or Server to perform socket communication with other devices.

Operation process:

On the Socket register page, configure the robot as a Client or Server parameter,

such as communication protocol, IP, port, etc.

Call the Socket program instruction when writing a program.

Position

Successively click "Menu Button" → "Data" → "Socket Register" to enter the socket register setting screen as shown in Fig. 5.6 and Fig. 5.7. TCP and UDP can be selected as functional protocols. The robot can be used as Server or Client, of which configuration items are different.

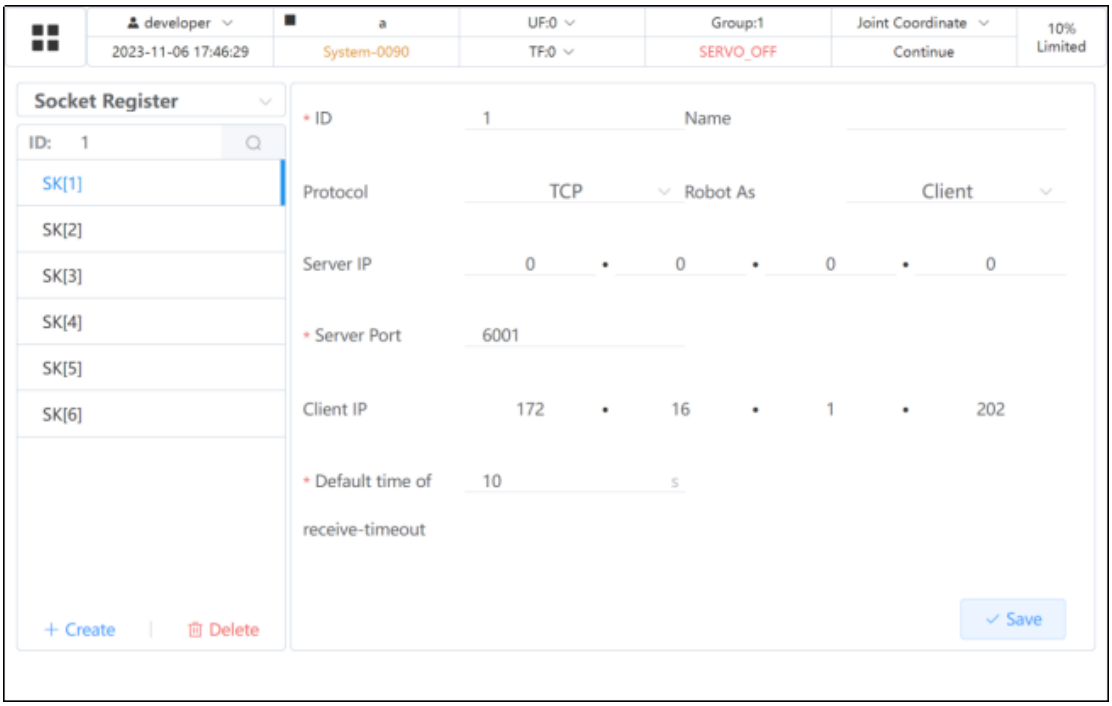


Fig. 5.6 Socket Register Setting Interface (Client)

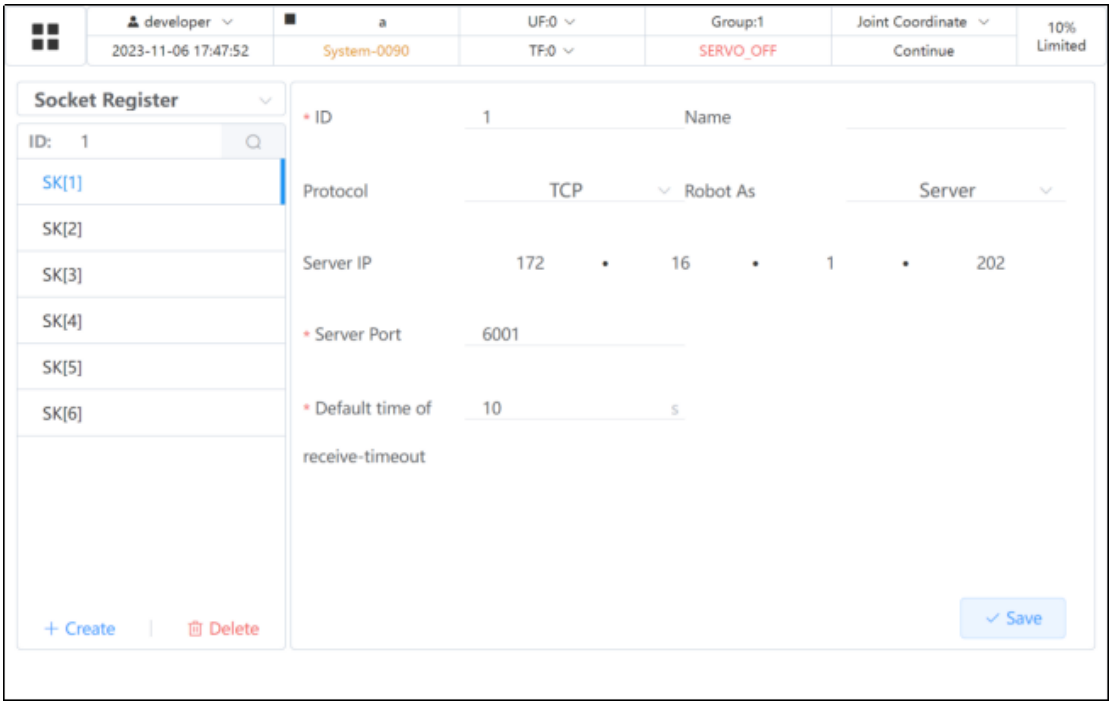


Fig. 5.7 Socket Register Setting Interface (Server)

Description of socket register parameters

- ID: Register number.
- Name: Appoint the name of string register here. The name can be composed of characters less than 31 bytes.

Robot as Server (as shown in Fig. 5.7):

- Server IP represents the IP of the robot.
- The server port represents the port number used by the Socket. It is used when the Client is connected.
- The default time of receive-timeout represents the waiting time when data is received from the Socket. It will not wait beyond this time. The return value of the corresponding program instruction for receive-timeout is 999.

Robot as Client (as shown in Fig. 5.6):

- Server IP represents the IP of the server to be connected by the robot on the opposite end. It is input by the user.
- Server Port represents the port of the server to be connected by the robot on the opposite end. It is input by the user.
- Client IP represents the IP of the robot (which cannot be changed in the interface shown in Fig. 5.7).
- The default time of receive-timeout is the same as the Server.

5.1.6 Modbus special registers

They can be used to monitor Modbus registers and currently support monitoring of slave Input Registers and Holding Registers.

Among them, Holding Registers are represented by MH[i] and Input Registers by MI[i], where i is the sequence number. Total number of each register can be configured to 120 at most.

Successively click "Menu Button" → "Data" → "Modbus Special Register" to enter the Modbus special register setting screen as shown in Fig. 5.8.

Caution

Before entering the Modbus Register screen, the Modbus slave function must be activated. Otherwise, an error may occur when entering the Modbus Register page.

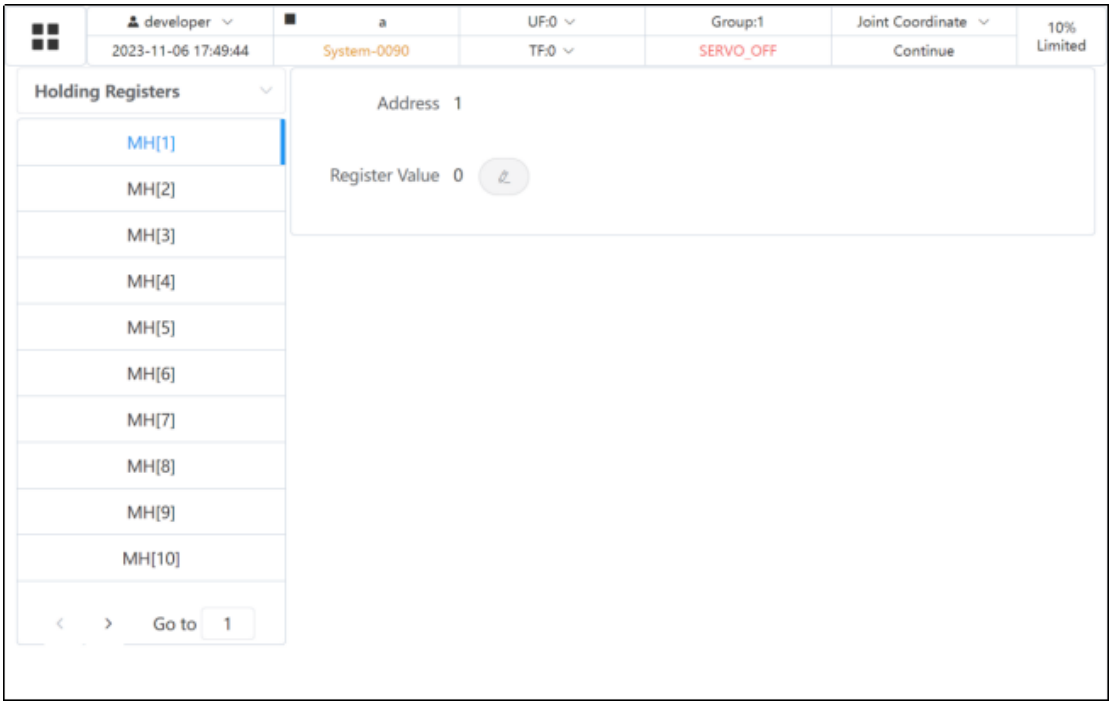


Fig. 5.8 Modbus Special Register Setting Screen

Click to switch between the monitoring interfaces of Input Registers and Holding Registers, as shown in Fig. 5.8.

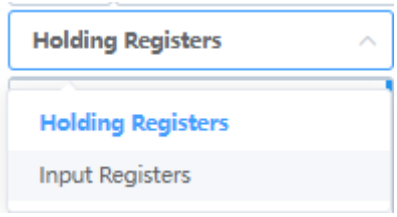


Fig. 5.8 Modbus Register Switching Window

I/O mapping and Modbus special registers

IO Mapping and Modbus Dedicated Registers

IO mapping can be used to map Inputs/Coils in Modbus dedicated registers to DI/DO. The steps are as follows:

1. Click "Menu Button" → "Communication" → "Bus Configuration" in sequence to enter the interface shown in Figure 5.9.

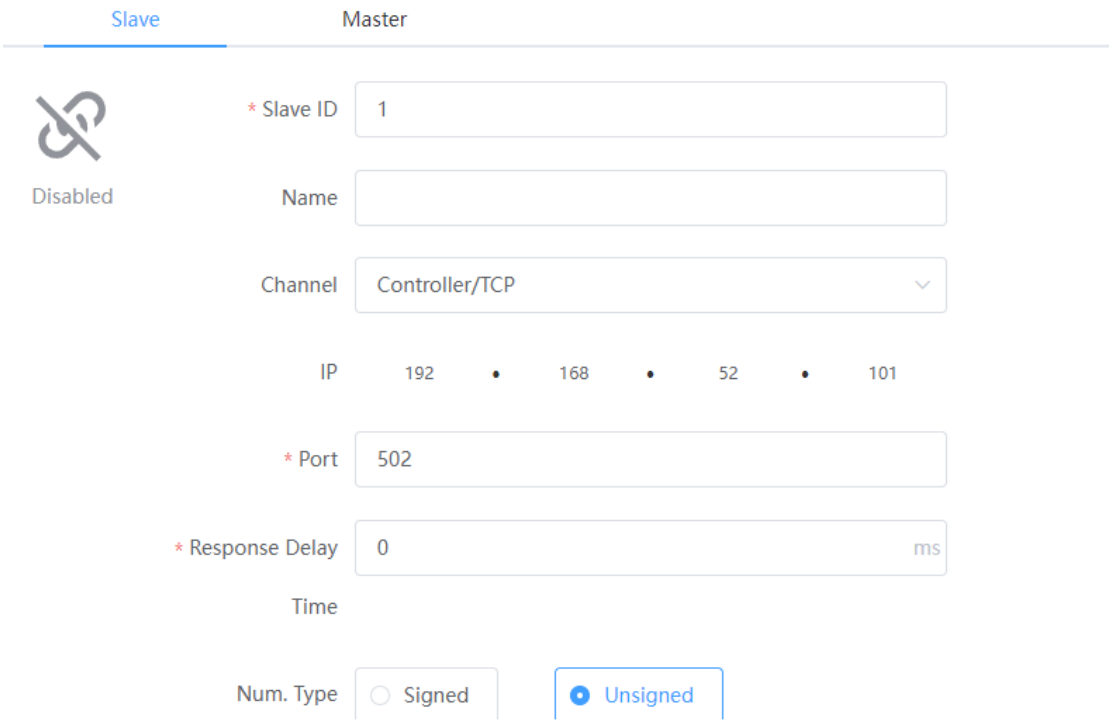


Figure 5.9 Bus Configuration Window

2. Enter the IO mapping page and map the required IO (Coil Status to DI, Discrete inputs to DO) (for specific operations, see Section 2.1.1).

3. Enter the IO status page to monitor and operate DI/DO.

*For more detailed configuration, usage methods, and practical cases of Modbus, please refer to the *Jiebot Modbus TCP Function Manual*.

5.1.7 Palletizing Registers

Palletizing registers are dedicated registers for palletizing functions. The maximum number of palletizing registers allowed to be set in the system configuration is 30. A palletizing register is automatically created (deleted) when the corresponding palletizing process is activated (deactivated) and cannot exist independently. Palletizing registers 1-30 correspond to palletizing processes 1-30 respectively.

A palletizing register is denoted as: PL[i:NAME]

The data mode contains 3 elements: [R, C, L]

Parameter	Description
i	The ID number of the palletizing register, representing the palletizing process from 1 to 30.
NAME	The name of the palletizing process, default is "default".
R	The data of the palletizing row, which cannot exceed the row configuration of the palletizing process.
C	The data of the palletizing column, which cannot exceed the column configuration of the palletizing process.
L	The data of the palletizing layer, which cannot exceed the layer configuration of the palletizing process.

That is, "i" in PL[i] ranges from 1 to 30, so PL[1] ~ PL[30] correspond to the palletizing processes of PALLETIZING[1] ~ PALLETIZING[30]. PL registers are global registers and visible to all programs. In principle, PL registers are not allowed to be operated independently in programs that do not use the corresponding palletizing process; the values of PL registers are retained after power-off.

Click "Menu Button" → "Data" → "Palletizing Registers" in sequence to enter the palletizing register setting interface, as shown in Figure 5.11.

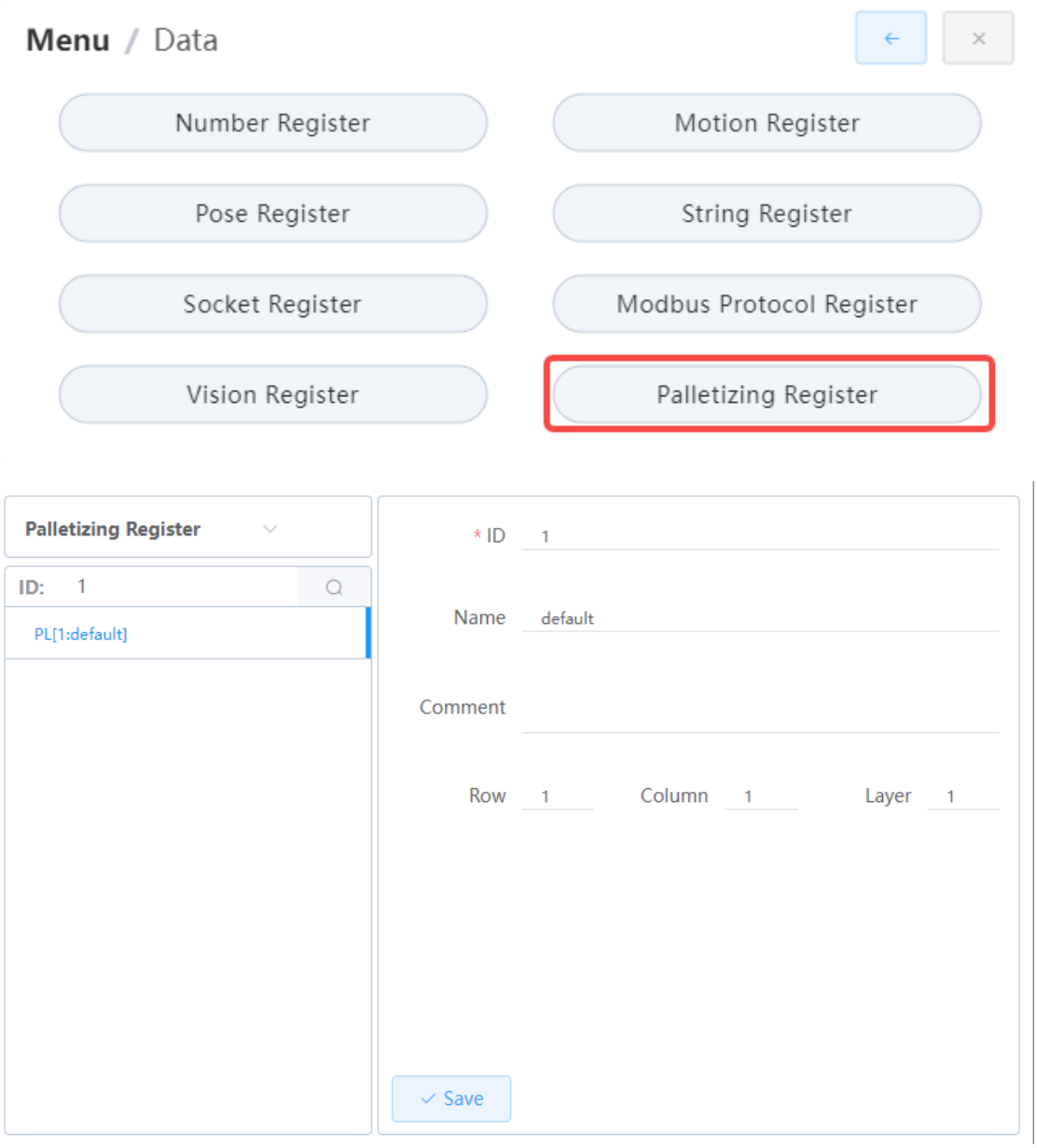


Figure 5.11 Palletizing Register Setting Interface

5.1.8 Vision Registers

Click "Menu" → "Data" → "Vision Registers" to enter the interface shown in Figure 3.1.

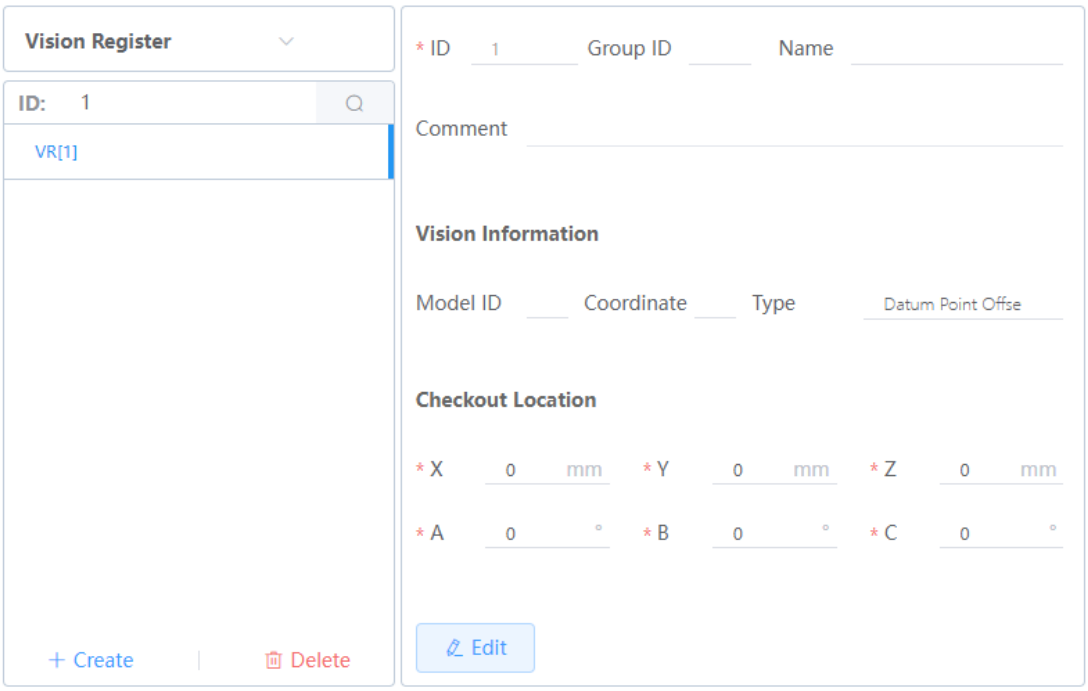


Figure 3.1 VR Register Setting Interface

Basic Information

- ID: The ID of the register
- Group ID: The motion group of the register, default is 1
- Name: A remark name for the register, default is empty

Vision Information

- Template ID: The template ID of the detected workpiece
- Coordinate System: The coordinate system number of the compensation data
- Type: The compensation type of the detected workpiece coordinate value
- Datum Point Offset: Position correction based on the compensation data of the user coordinate system, with the coordinate system number being the user coordinate system
- User Coordinate System: The absolute position of the detected workpiece, with the coordinate system number being the user coordinate system

5.2 Current position

It is the current position interface (as shown in Fig. 5.11) mainly used to provide the

user with real-time information about the robot's current pose during robot teaching; and provide the user with a convenient function to move to a confirmed target point.

This interface consists of two parts: Positions in Coordinate (current position) (Area 1 in Fig. 5.11) and Go To Position (target position) (Area 2 in Fig. 5.11) of the robot.

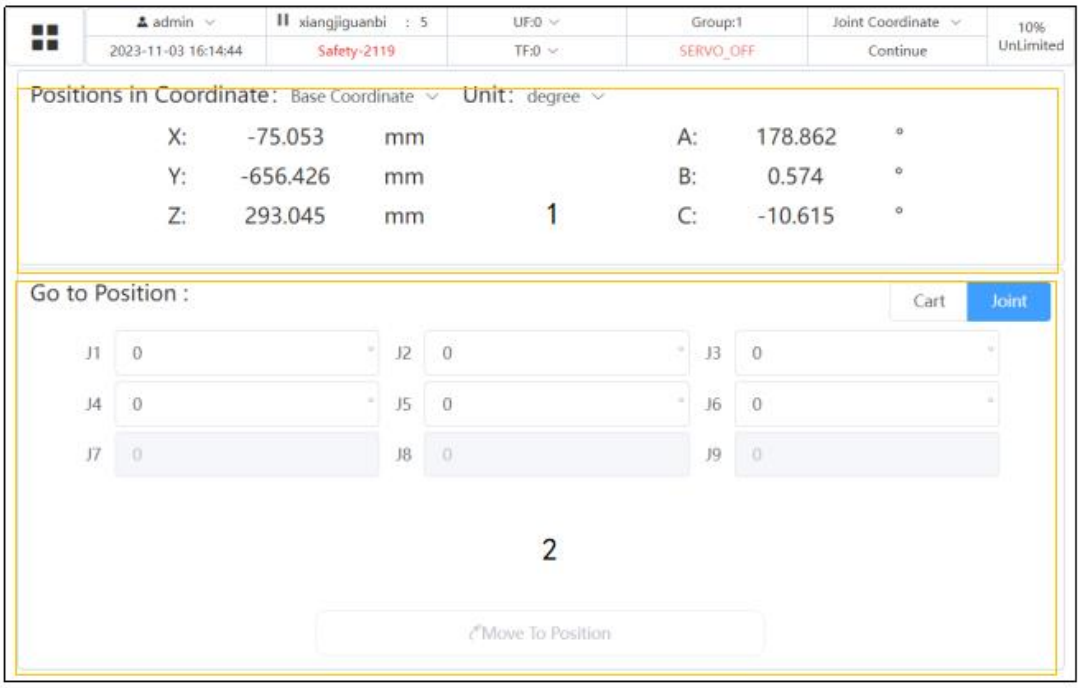


Fig. 5.11 Positions in Coordinate Interface (Cartesian)

Description of Positions in Coordinate area

The Positions in Coordinate area displays the current pose information of the robot, so that the user can conveniently observe the robot's pose in real-time during the teaching process. The displayed data is shown in Fig. 5.11 when the system coordinate system is selected as the Cartesian space coordinate system and in Fig. 5.12 when the joint coordinate system is selected.

Fig. 5.12 Current Position Interface (Joint)

Click to choose world coordinate system, base coordinate system, joint coordinate system or user coordinate system. If the world, base or user coordinate system is selected, the coordinates of current robot's TCP in the corresponding coordinate system will be displayed. If the joint coordinate system is selected, joint coordinates of current robot will be displayed. (It is independent of the "Current Coordinate Selection" on the status bar).

Click to convert angle units: angle system or radian system.

Description of “Go To Position” area

The user can input the specified target point (it is allowed to input Cartesian or joint coordinate data and click to switch data types). After that, the robot is enabled to power on. Press and hold the "Move to Position" button so that the robot can slowly move to the target point in a relative motion mode (MoveJ).

Caution

During the movement, the user must hold the "Move to Position" button until the robot reaches the target point. Otherwise, the robot may immediately slow down and stop.

6. Robot communication setting

Summary

Agilebot robots are provided with the Modbus TCP protocol, so that the customers can communicate with peripheral devices through buses.

Modbus TCP is an open master/slave application communication protocol. Currently, Agilebot robots can only support the slave function. 4 transmission data formats are as follows:

Function	Type	Authority	Robot mapping
Discrete output (coils)	Single bit	Read and write	Digital Input
Discrete input (coils)	Single bit Single bit	Read only	Digital output
Input Registers	16-bits word	Read only	MI register
Holding Registers	16-bits word	Read and write	MH register

6.1 Slave configuration

Successively click "Menu Button" → "Communication" → "Bus Configuration" to enter the Modbus slave configuration screen as shown in Fig. 6.1.

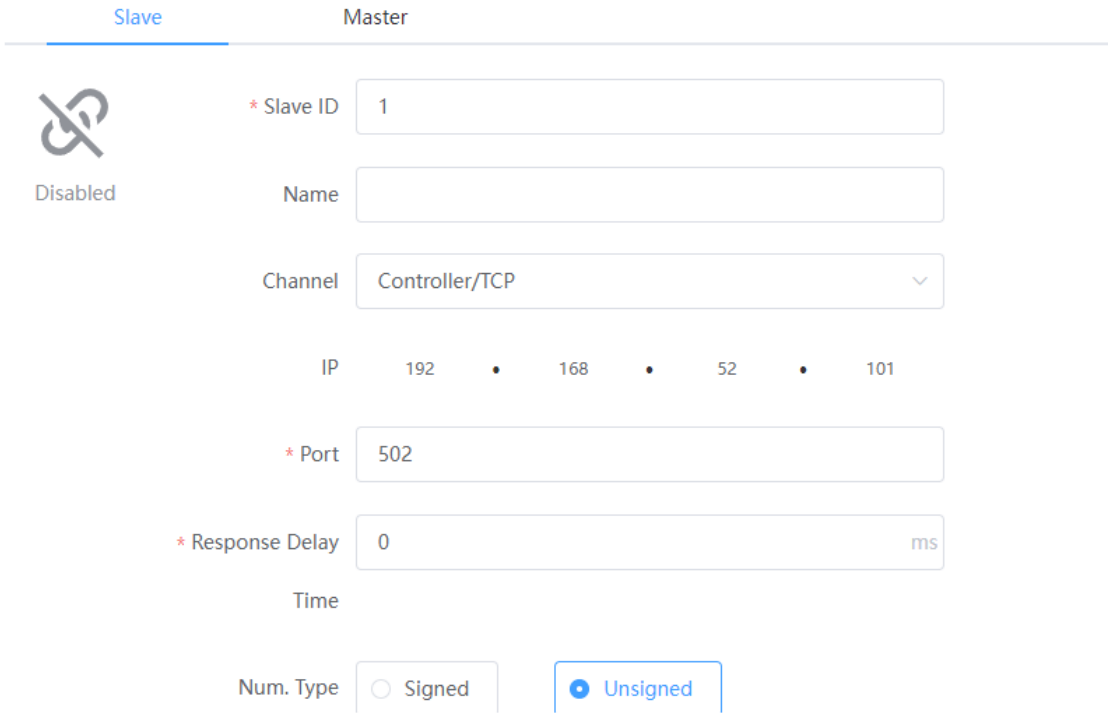


Fig. 6.1 Current Position Interface

Description of parameters:

- IP address: Slave (robot) address
- Port number: The port used for communication with the master station
- Name: Slave Name
- Response delay: The latest response time specified for communication between master and slave stations

Slave setting steps

1. In Fig. 6.1, it is necessary to first turn off the Enable state if the slave function is enabled. Then, the configured IP address must be in the same domain as the robot (the default IP of the robot is 192.168.110.2) and the name and response delay of the slave station cannot be kept blank.

2. Click "Function" in Fig. 6.1 to enter the register setting interface shown in Fig. 6.2.

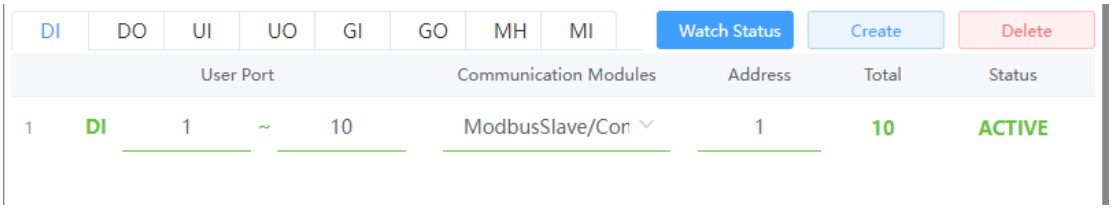


Fig. 6.2 Bus Register Signal Setting Screen

3. Set register address and number according to actual needs as shown in Fig. 6.2.

- Coil register: address and range of0-65536
- Discrete Input register: address and range of 0-65536
- Holding Register: address and range of 0-9999
- Input Register: address and range of 0-9999

4. After setting the register address and number, click "Eable" to return to the interface in Fig. 6.1 and to enable the configuration as shown in Fig. 6.3.

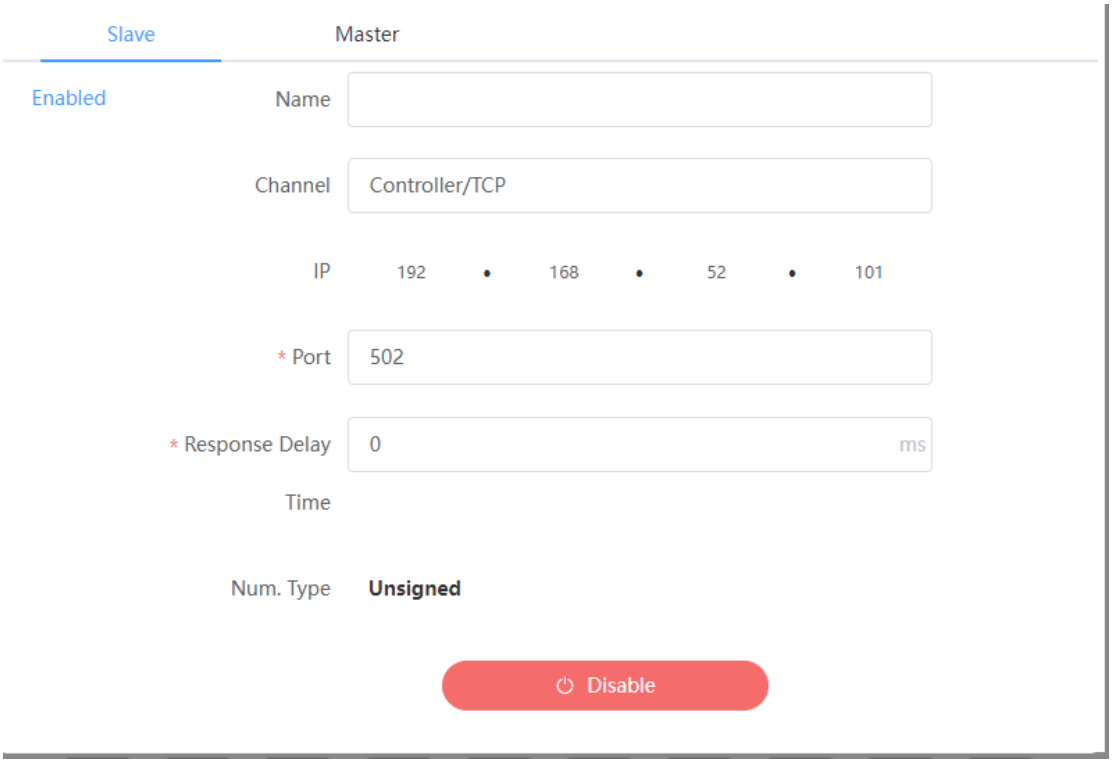


Fig. 6.3 Current Position Interface

- For more detailed configuration, usage and practical cases of Modbus, please refer to the *Agilebot Modbus TCP Function Manual*.

6.2 Master configuration

Click "Menu Button → Communication → Bus Configuration → Master Station" in

sequence to enter the Modbus master station configuration interface as shown in the figure.

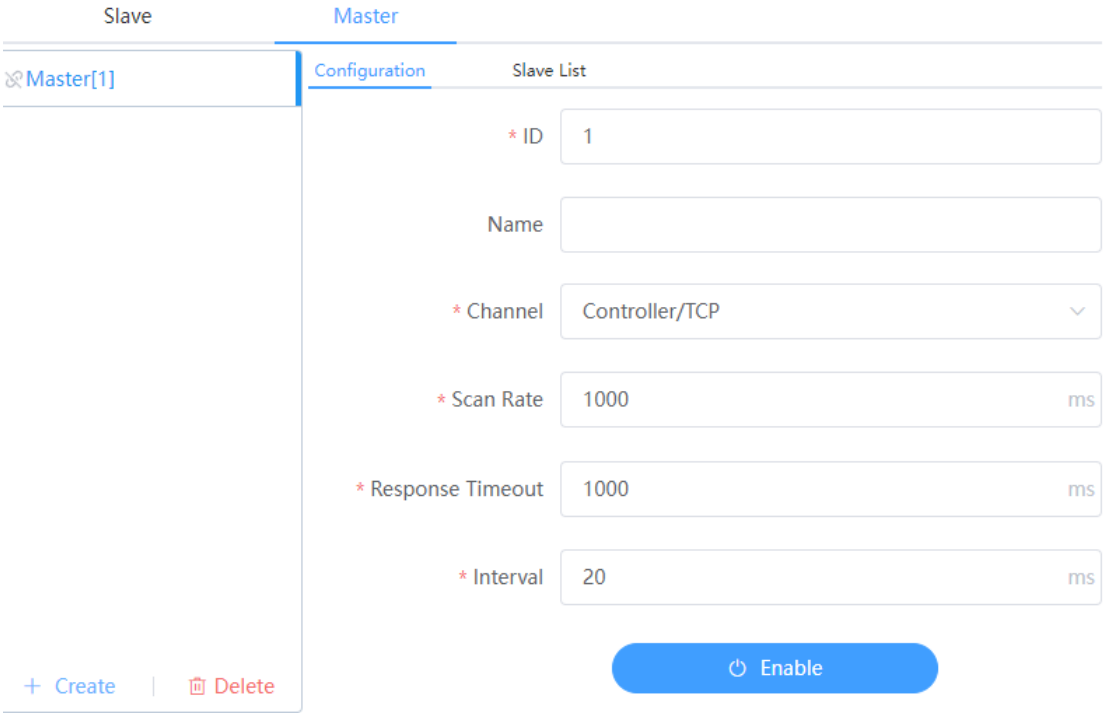


Figure 6.3 Master Station Interface

Parameter Description:

- ID: Modbus ID
- Name: Maximum 128 bytes
- Communication Channel: Optional Controller/TCP, Controller/TCP2RS485 modes
- Communication Scan Rate: Master station polling cycle, default 1000 ms
- Slave Response Timeout: The maximum response time specified when the master station communicates with slave stations
- Scan Interval: The cycle for the master station to poll each slave station, default 20 ms

Caution

Before enabling the master station, ensure that all slave stations communicate normally; otherwise, the enabling will fail.

If TCP is used, select the Controller/TCP mode, as shown in Figure 6.4.

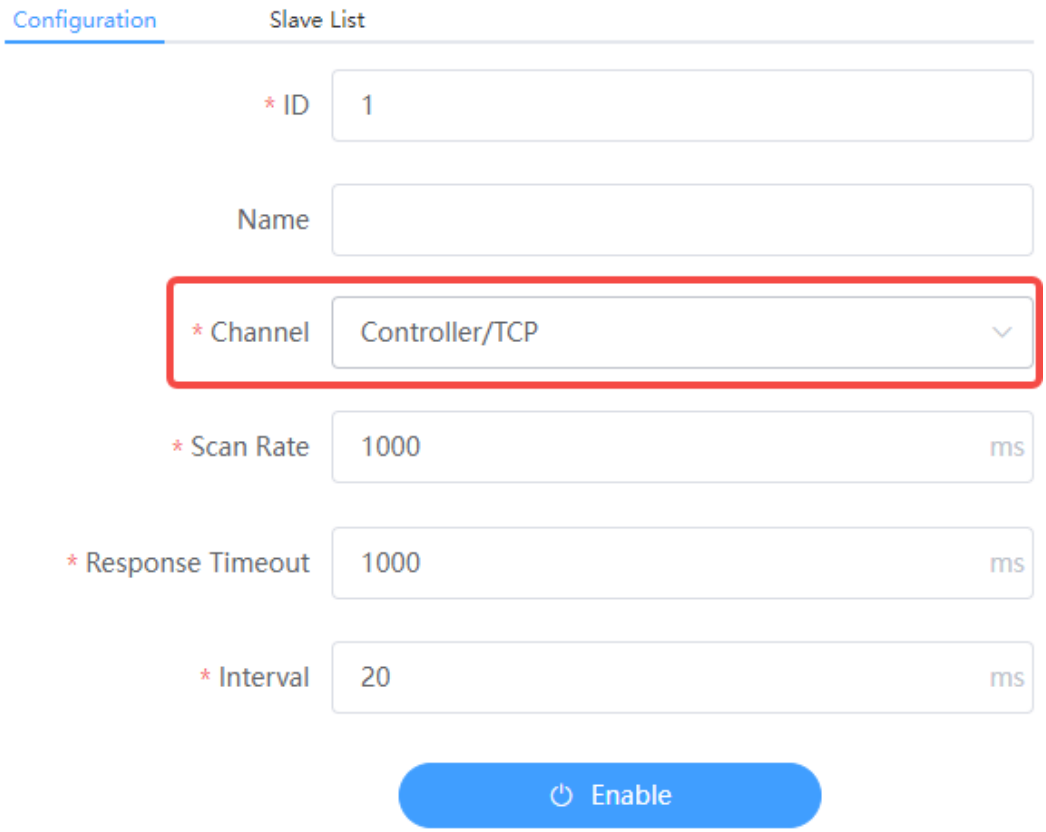


Figure 6.4 Controller/TCP Mode

If a TCP-to-RTU gateway is used, select the Controller/TCP2RS485 mode, as shown in Figure 6.5. For this item, the IP address and port of the relay conversion module need to be filled in additionally, and the module must use the transparent transmission mode.

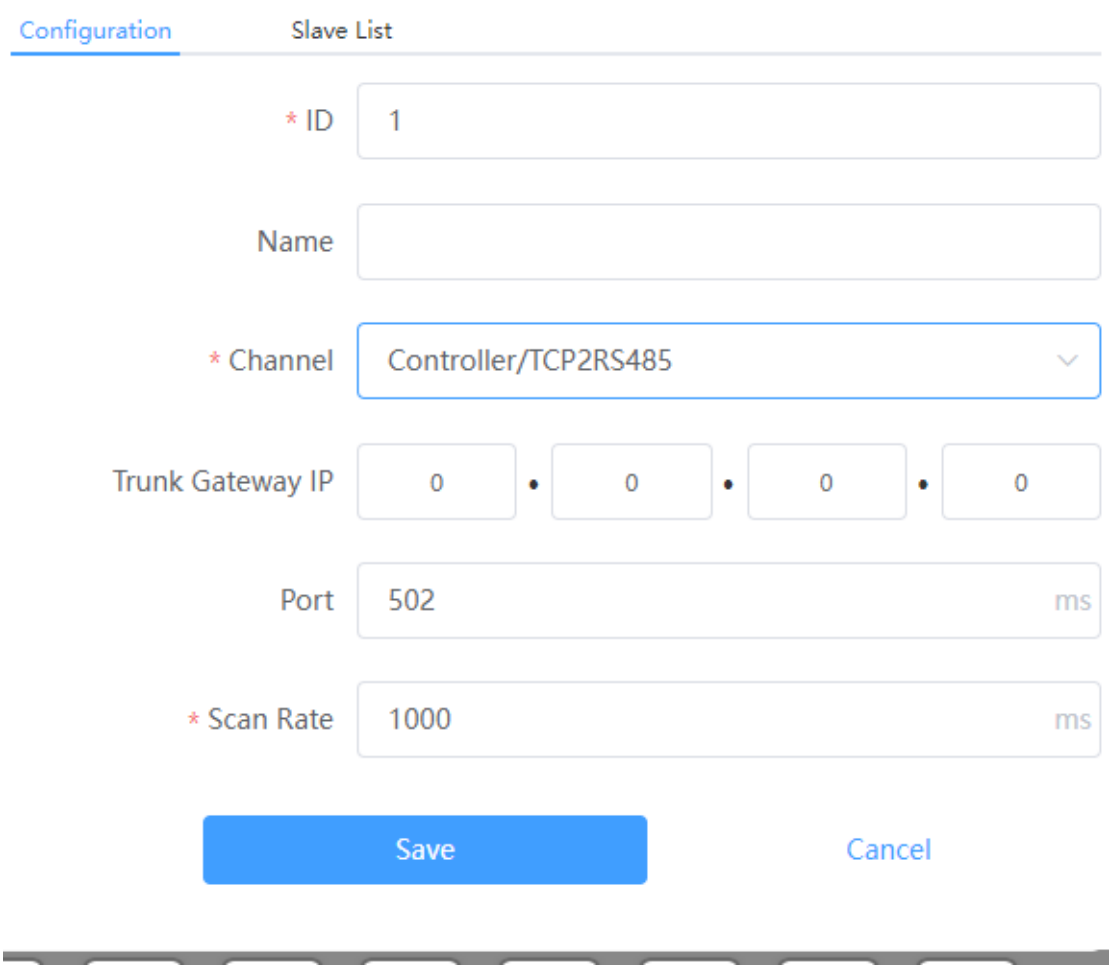


Figure 6.5 Controller/TCP2RS485 Mode

In the slave station list of the master station, you can create Modbus slave stations connected to the master station, and a maximum of 32 slave stations can be established. The following is the slave station list in Controller/TCP mode:

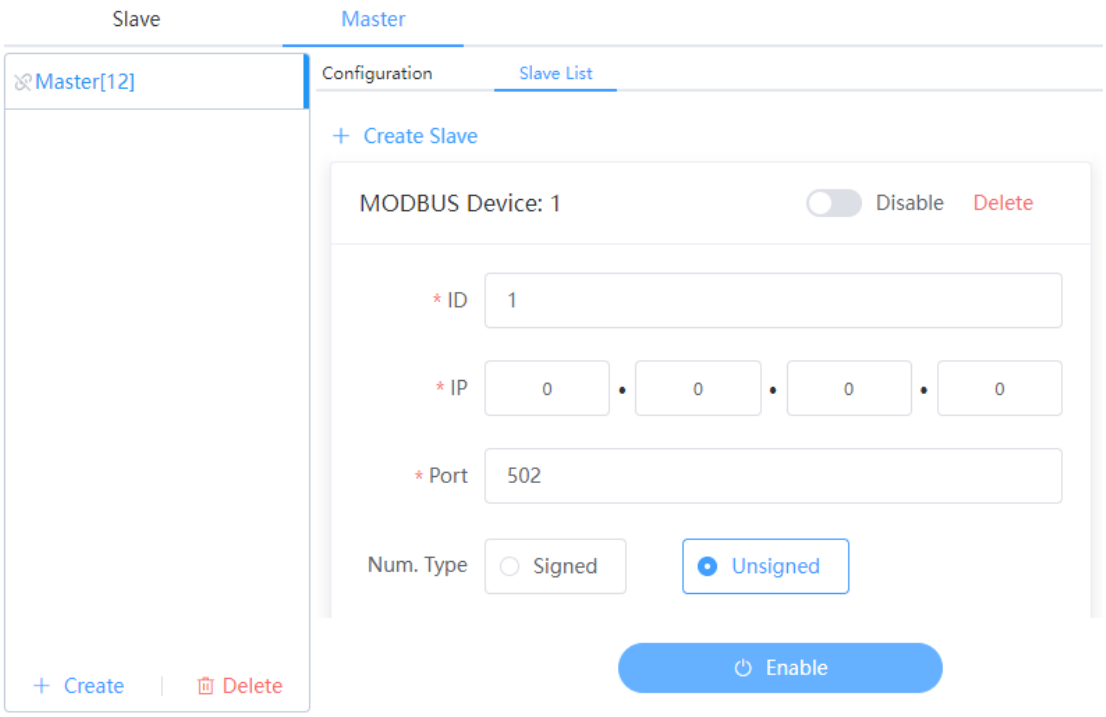


Figure 6.8 Slave Station List of Master Station

Parameter Description:

- Slave ID: Modbus slave ID
- Slave IP: IP address of the slave station
- Slave Port: Port used for communication with the master station
- Value Type: Signed/Unsigned

*For more detailed configuration, usage methods, and practical cases of Modbus, please refer to the *Jiebot Modbus TCP Function Manual*.

7. System backup and loading

- Prevent accidental damage or loss of programs or software during use or transmission.
- Before program modification, make a backup of the source program for convenient recovery.
- Overall coverage or loading of robot software, programs and files.

7.1 Allowable storage devices

Storage devices in FAT32 format, with USB-A interface, USB2.0 protocols and 8G-32GB capacity, such as USB flash drives, mobile hard drives, etc.

The recommended USB drive brands and models are shown in the table below:

Brand	Model	Capacity
Kingston	DTXM	32G
SanDisk	CZ73	32G

Caution

The USB memory has security features and the product requiring password authentication when accessing to the drive cannot be used.

7.2 Backup and load objects

Data

- User's program files
- Data files
- System setting files
- TP config files
- MCCP files
- Logs

Software

- Robot application software
- Robot motion control software

Caution

Exclude operating system and underlying drivers.

Selection of backup method:

1. Single file backup (file management interface): Choose a single file for backup and save it to a folder in the designated USB drive path.

2. Single type backup (file management interface): Choose a file type, back up all files of this type and save them to a folder in the designated USB drive path.

3. All backup (file management interface): Back up all files and save them to a folder in the designated USB drive path.

4. Mirror backup (file management interface): Mirror all contents listed: all files, robot application software and robot motion control software, and save them in zip or other form to a designated USB drive.

Selection of load method:

1. Single file loading (file management interface): Designate a path and select a single file for loading.

2. Single type loading (file management interface): Designate a path, check a file type and load all files of that type in this directory.

3. All loading (file management interface): Choose the folder on the USB drive and load all recognizable robot files in this directory.

4. Mirror loading (file management interface): Choose the mirror zip package in the folder on the USB drive, select the mirror loading mode, decrypt and decompress the zip package and replace all files and installed software in the controller and TP (not involving operating system). Generally, this method is used to restore software to a certain version or recover a debugged robot engineering application as a whole.

7.3 Description of other functions

- Path browsing: It provides path browsing and selection functions for backup and restore/loading. It is allowed to open the folder in the path of Cabinet-USB\AgilebotRobotBackup in the USB drive until you select the desired subdirectory.

- File recognition: During backup and restore/loading, only mirror files (.zip) and common folders in the path of Cabinet-USB\AgilebotRobotBackup can be recognized. It is required to ensure that this directory only contains files related to the robot system.

- Folder edit: It is allowed to create, delete and edit the named folders in the directory of Cabinet-USB\AgilebotRobotBackup.

- Overwrite prompt function: When restoring/loading to the controller, please confirm whether to overwrite the files with the same names (if any). The options include "Confirm to Overwrite", "Skip", "Cancel Backup/Load (previously overwritten files cannot be restored)" and there is a check box "Perform the same operation for subsequent files".

- Progress confirm: display backup or loading progress.

7.4 User's operation mode

Backup:

1. Insert the USB drive into the USB port on the robot controller (TP USB port does not support backup operation).

2. Switch the robot to the SERVO_OFF state. Click "Menu Button" → "Management" → "File Management" → "Backup" and open the file management interface, as shown in Fig. 7.1.

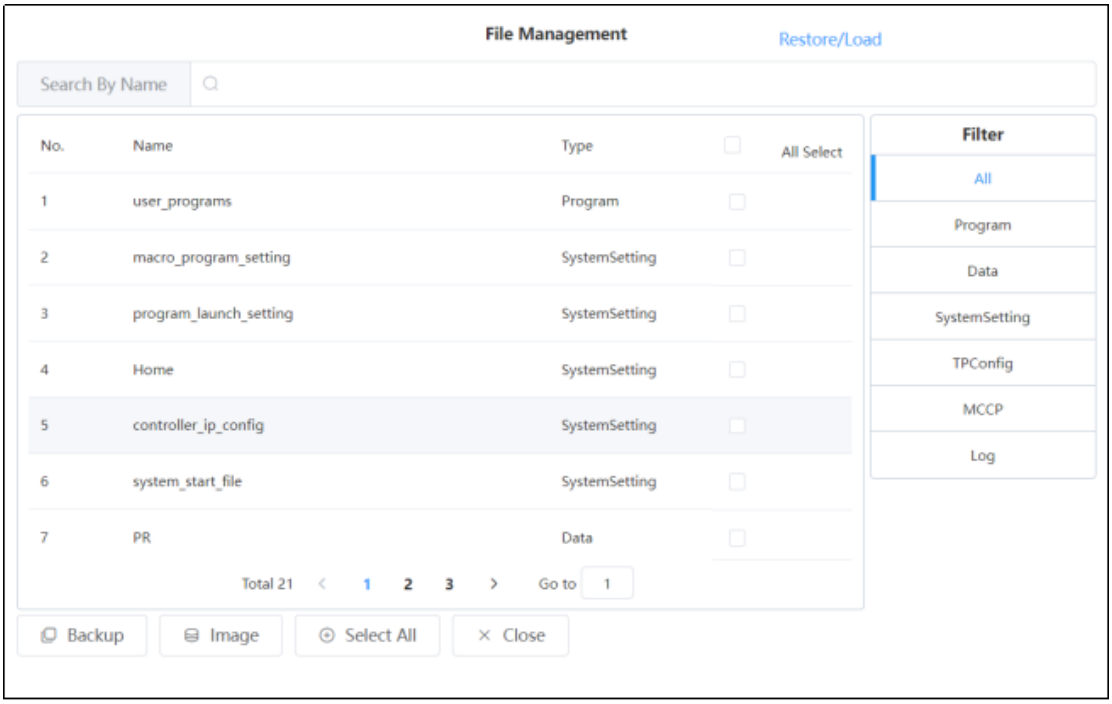


Fig. 7.1 File Management Interface

3. Click the square behind the target file, select the target file and then click the "Backup" button.

4. Select USB path and specific operating directory, as shown in Fig. 7.2 and 7.3. If necessary, create or edit folders.

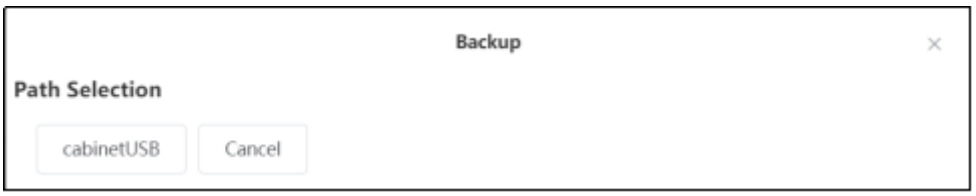


Fig. 7.2 USB Path Selection Interface

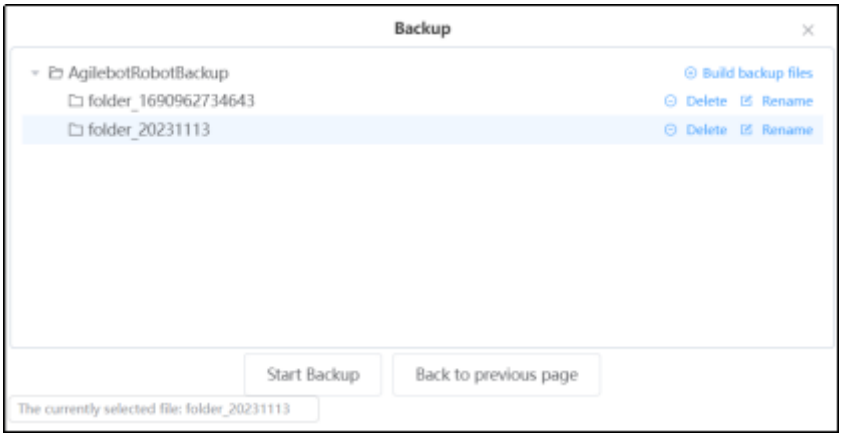


Fig. 7.3 File Directory Interface

5. Select the destination folder for backup and click "Start Backup".

6. Wait for the system to complete the execution.

Mirror:

1. Insert the USB drive into the USB port on the robot controller (TP USB port does not support backup operation).

2. Switch the robot to the SERVO_OFF state. Click "Menu Button" → "Management" → "File Management" → "Backup" and open the file management interface.

3. Click on the "Mirror" button.

4. Select USB path and specific operating directory. If necessary, create or edit folders.

5. Select the destination folder for backup and click "Start Backup".

6. Wait for the system to complete the execution.

Restore/Load:

1. Insert a USB.

2. Switch the robot to the SERVO_OFF state. Click "Menu Button" → "Management" → "File Management" → "Restore/Load" and open the file management interface.

3. Select USB path and specific operating directory.

4. Select the files to load and click “Start Restore”; wait for the system to complete the execution.

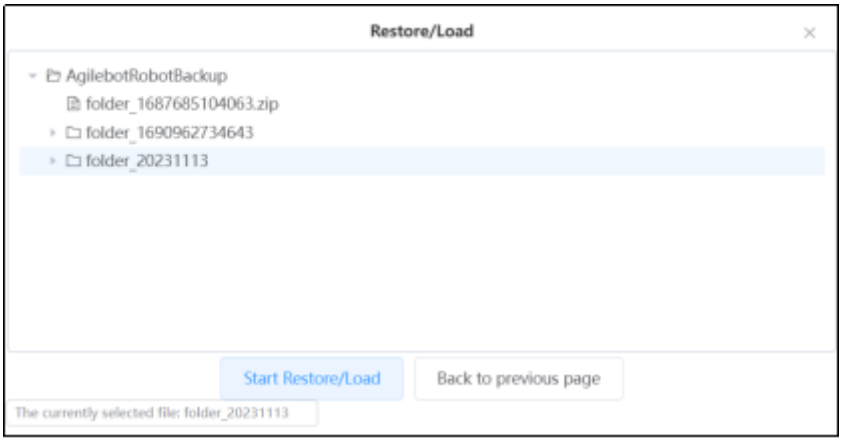


Fig. 7.4 Restore File Directory Interface

Precautions

- Authority: This operation can be performed under admin.
- Execution conditions: The robot is in the manual mode, does not perform any motion instructions (regardless of teaching or programming) or is idle.
- It is necessary to restart the controller and TP after mirror restore.
- No other software operations are allowed during loading or backup.

8. Practical functions

8.1 Program offset

For a certain range of action statements in the program, the recorded poses in motion statements are uniformly transformed according to pre-specified rules and then the transformed program is treated as a new program or segment.

Offset type

- General offset: parallel or parallel-rotation offset
- Mirror offset: symmetrical offset relative to the designated symmetry plane
- Circular array offset: Angular offset around the rotation axis on the same circumference

Typical application scenarios

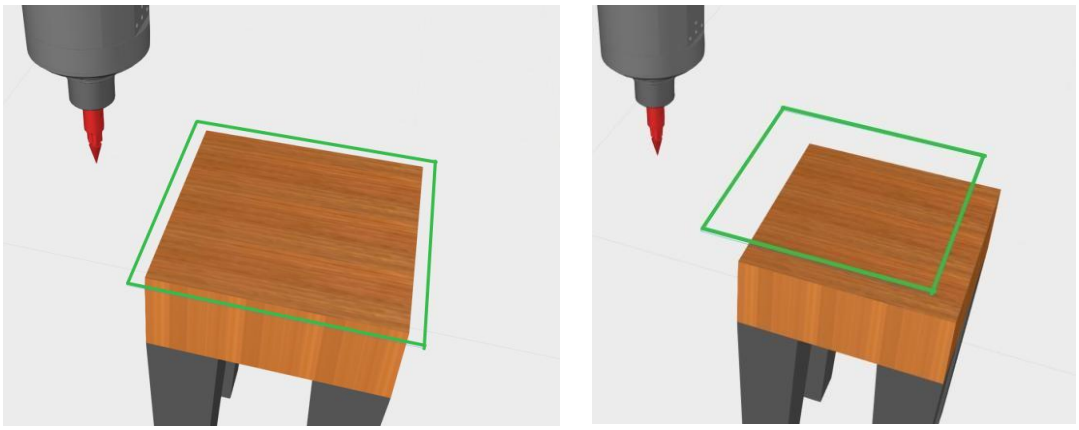


Fig. 8.1 Pose Data Offset Based on Certain Pattern

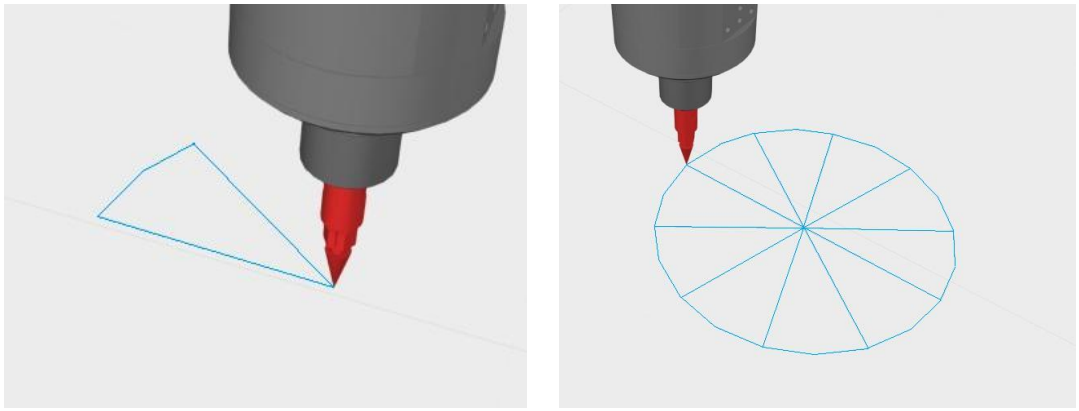


Fig. 8.2 Regular Repeating of Pose Data

Steps:

Click "Menu Button" → "Application" → "Program Offset" to enter the function setting interface as shown in Fig. 8.3;

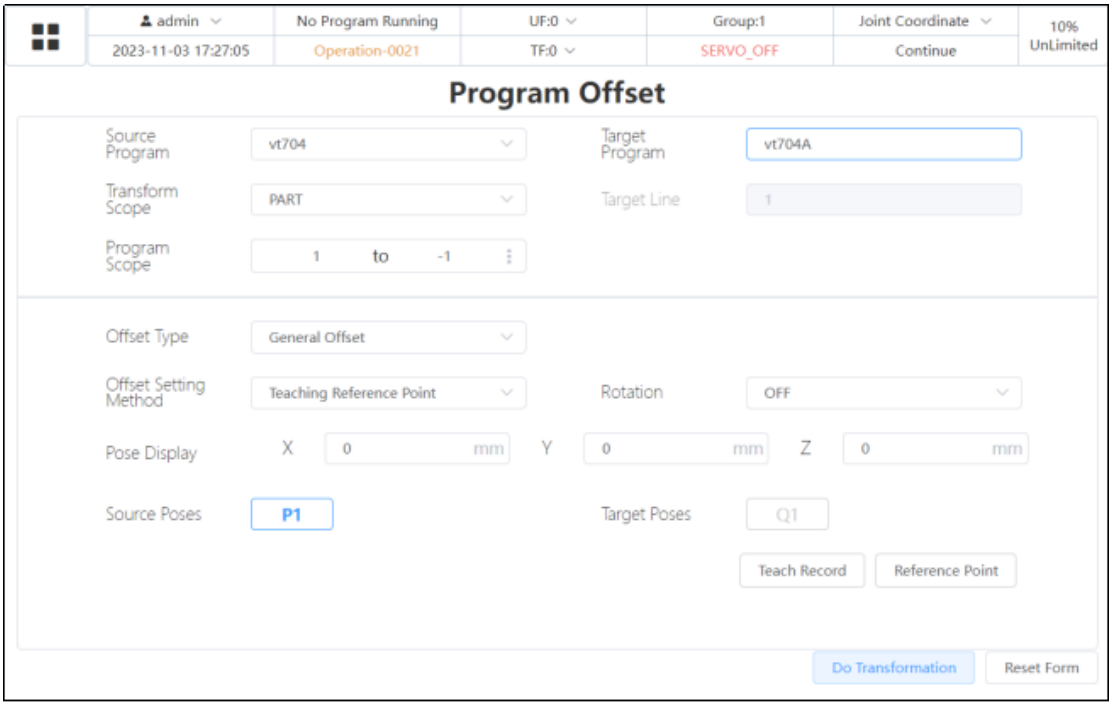


Fig. 8.3 Program Offset Setting Interface

5. Source Program: Choose the program name to be offset in the "Source Program" field. If offsetting a certain range in the Source Program, choose "Part" in the "Transform Scope" and enter the specified line number in the "Program Scope";

6. Target Program: Enter a new program name in the "Target Program" field, and the offset instructions will be generated in the new program. If an existing program name is entered, it is required to specify the line number to be inserted into the existing program in the "Target Line";

7. Offset Type: Choose General Offset, Mirror Offset or Circular Array Offset in the "Offset Type".

8. After setting, click "Do Transformation" to generate a new point.

9. Click “Save” to generate a new program.

Precautions:

- Offset calculation is only allowed for P[] rather than PR[].
- Cart and Joint points are kept unchanged after offset.
- The offset of Joint point fails if beyond the soft limit after offset. This point is used as the value of the untaught log in the copied program.
- If being offset according to the rules, Cart points are not used in reachability judgment.

8.1.1 General offset

General offset

- Directly input X, Y and Z offsets.
- Reference poses: 1 source pose and 1 target pose

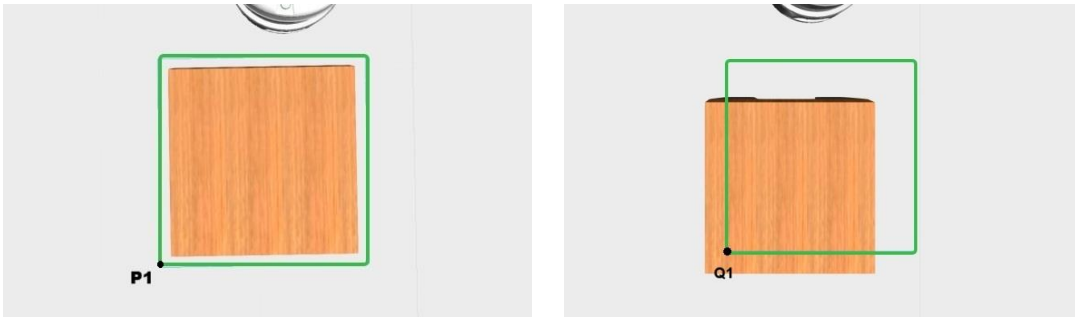


Fig. 8.4 General Offset

General offset with rotation

- Reference poses: 3 source poses and 3 target poses

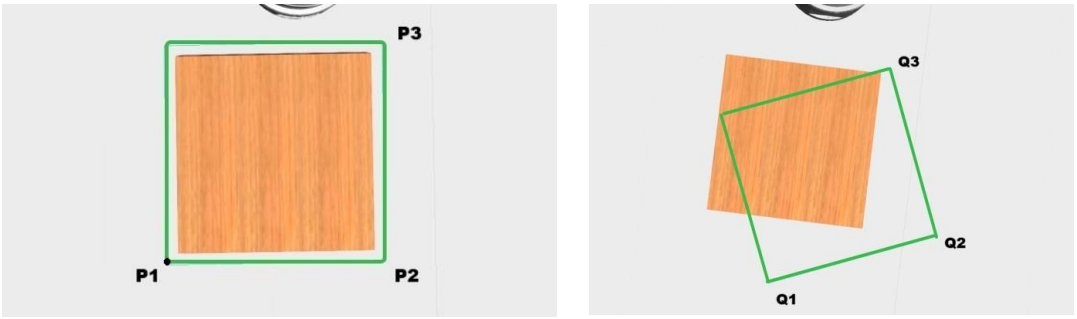


Fig. 8.5 General Offset with Rotation

8.1.2 Mirror offset

Mirror offset

- Reference poses: 1 source pose and 1 target pose

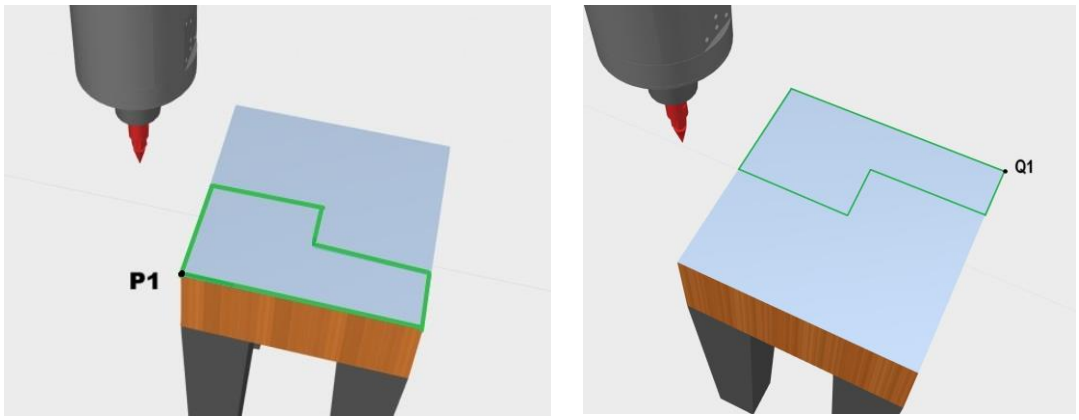


Fig. 8.6 Mirror Offset

Mirror offset with rotation

- Reference poses: 3 source poses and 3 target poses

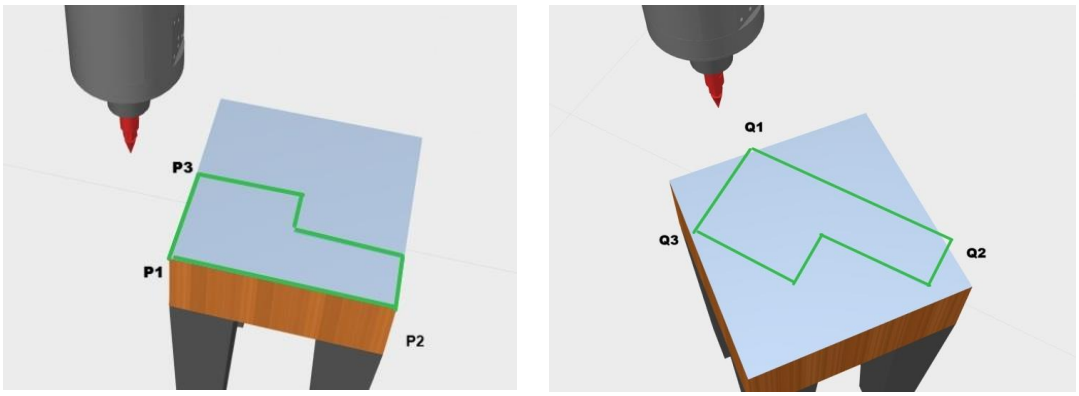


Fig. 8.7 Mirror Offset with Rotation

8.1.3 Circular array offset

No rotation axis is designated.

- Reference poses: 3

Rotating surface: form a rotating surface automatically by 3 points

Rotation axis: The center of a circle formed by three points is perpendicular to that of the rotating surface.

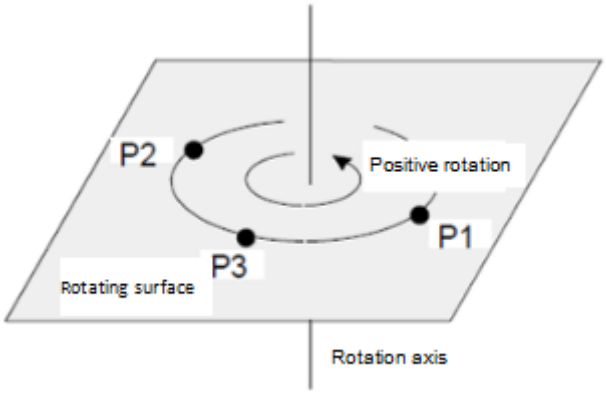


Fig. 8.8 No Rotation Axis Designated

Rotation axis designated

- Reference poses: 3 reference poses and 1 pose on the rotation axis

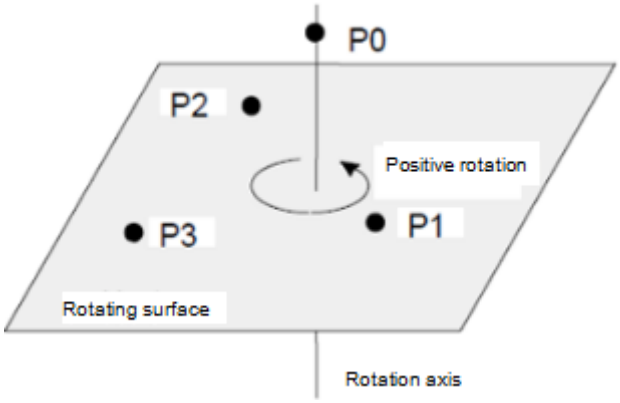


Fig. 8.9 No Rotation Axis Designated

8.2 Basic space anti-interference

Interference prevention function in basic space: When other robots or peripheral devices enter the predetermined interference region, the robot may automatically stop until it is confirmed that other devices have moved from the interference region, even if a motion command is issued to the robot to enter the interference region. Then, the stop state is released and the robot automatically restarts the action. Meanwhile, corresponding outputs can be set. The robot can determine the position of the TCP point in real-time. When TCP enters the interference region, the designated DO state is set to OFF, informing peripheral devices that the robot has entered the interference region. Corresponding DO will be set to ON after TCP leaves the interference region.

8.2.1 Setting steps:

1. Click "Menu Button" → "Application" → "Reference Pose" → "Basic Space Anti-interference" → "New" to enter the configuration interface, as shown in Fig. 8.10.

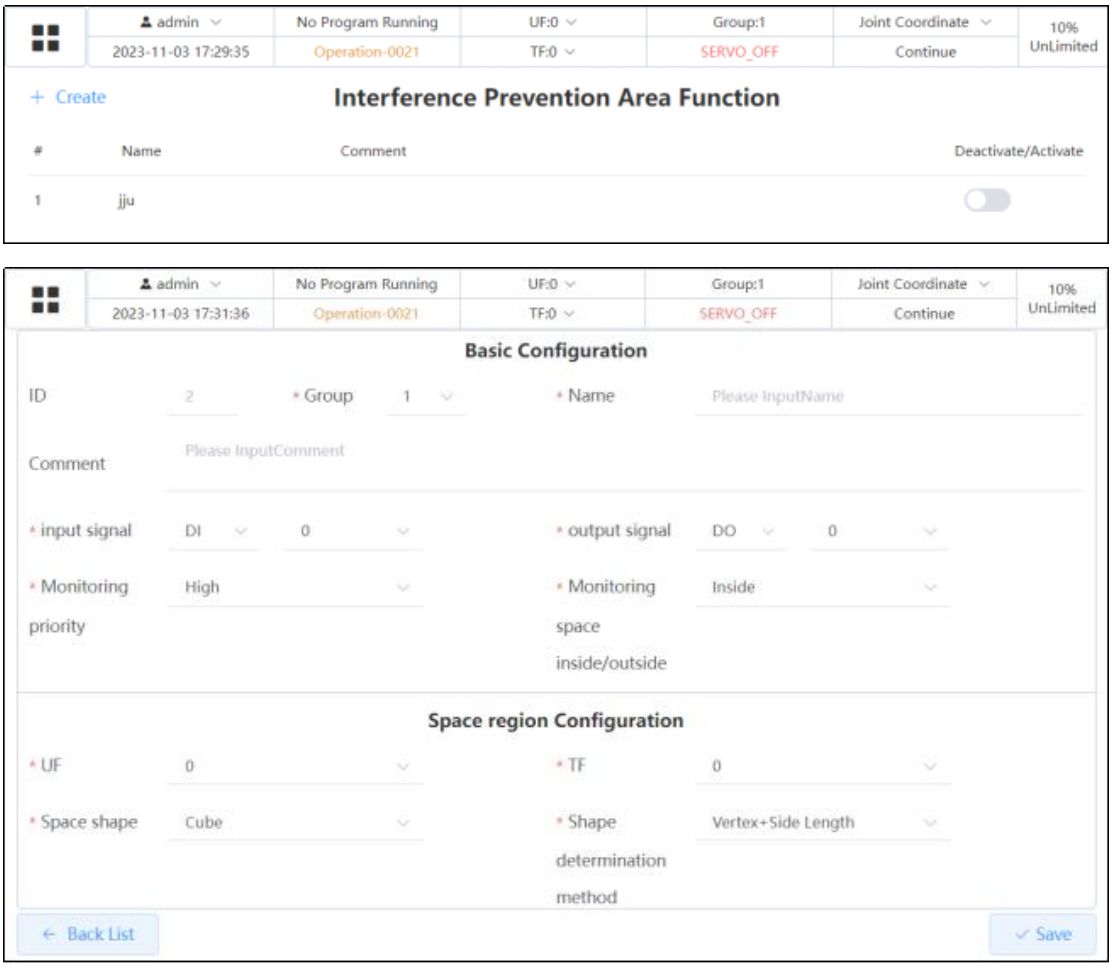


Fig. 8.10 Basic Space Anti-interference

2. After setting, click "Save" → "Return" → "Deactivate/Activate" to activate the interference region as shown in Fig. 8.11.

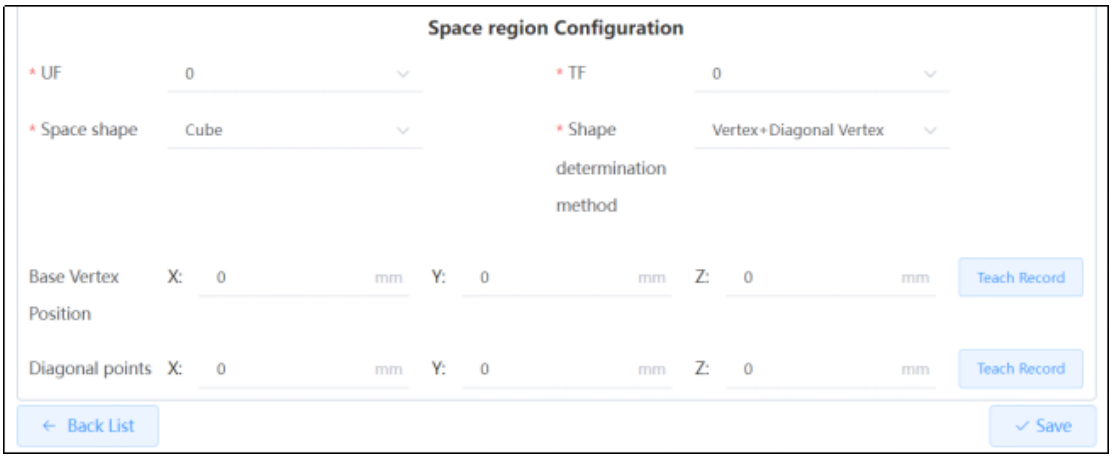


Fig. 8.11 Saving of Basic Space Anti-interference

8.2.2 Introduction to basic configuration interface:

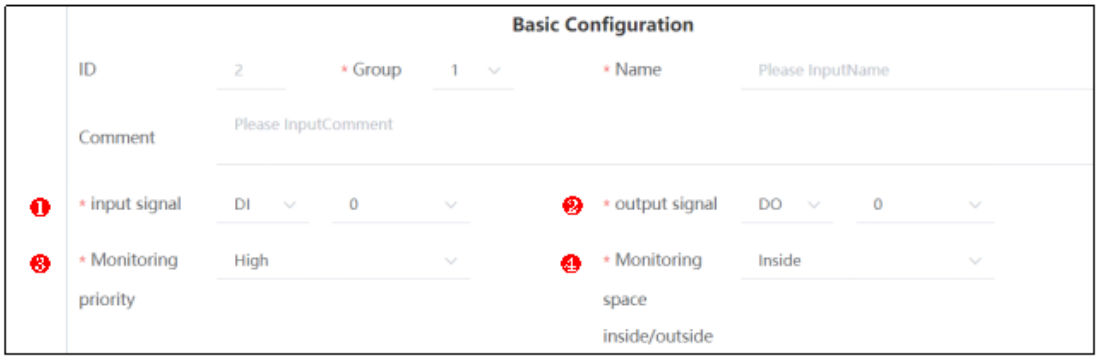


Fig. 8.12 Basic Configuration Interface

1. Input signal:

Set the input signal (customized signal) to input whether other robots or external devices have entered the interference region.

When the input signal is disconnected and the robot tries to enter the interference region, the robot gets into a hold state. When the input signal is connected, the hold state is released and the system automatically restarts.

Caution

The robot decelerates and stops from the moment it enters the interference region from the center point of the tool. So, the actual stop position of the robot is where it enters the interference region.

2. Output signal:

Set the output signal (customized signal). The output signal is disconnected when TCP exists in the interference region and connected when TCP is outside the region.

- The output signal is connected under the safety status (outside the interference region).
- The output signal is disconnected under the hazard status (inside the interference region).

3. Monitoring priority:

Make clear which robot has priority in entering the interference region when this function is used on two robots and these robots attempt to enter the interference region simultaneously. It should be designed that the robot on the "high" (priority) side enters the interference region first, performs and completes the operation and exits the interference region, and then the robot on the "low" (non-priority) side can enter the interference region. In addition, 2 robots must be arranged with different

settings from the other side.

4. Monitoring space (inside/outside):

Appoint the inside or outside of the set cut as the interference region.

8.2.3 Introduction to space region configuration interface:

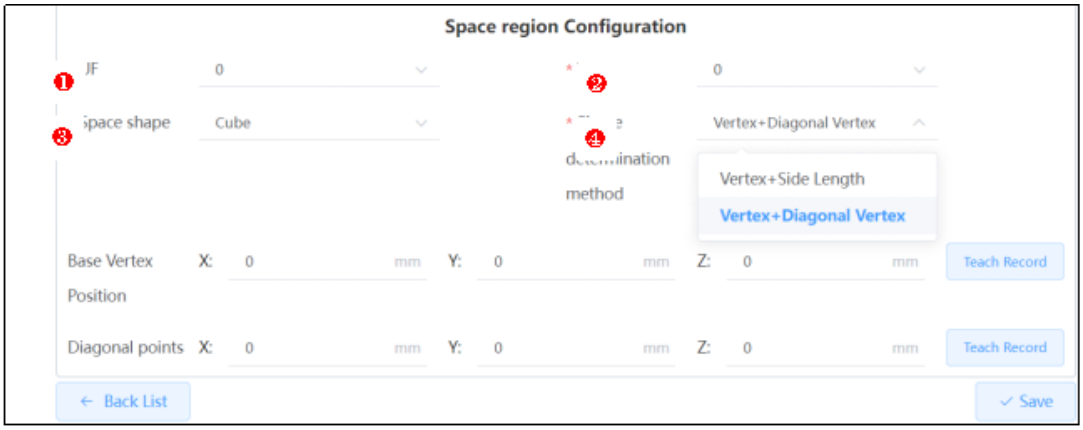


Fig. 13 Space Region Configuration Interface

Contents of space region configuration	Description
UF	Choose the user coordinate system.
TF	Choose the tool coordinate system.
Space shape	Cube at default
Shape determination method	Vertex + Side Length: The length from the base vertex to the side of the cube along Axes X, Y and Z of the user coordinate system (each side of the cube must be parallel to the coordinate axis of the user coordinate system). Vertex + Diagonal Vertex: The interference region is a cube with reference and diagonal vertices.

Setting steps for Vertex + Side Length method:

1. Set vertices of the interference region. Move the currently activated TCP of the robot to the position set as the vertex of the interference region, and click on "Teach Record". Then, the current TCP position is set as a vertex of the interference cube. It

is also allowed to directly input the coordinates X, Y and Z.

2. Specify side length of the cube along Axes X, Y and Z of current user coordinate system from vertices in the side length column.

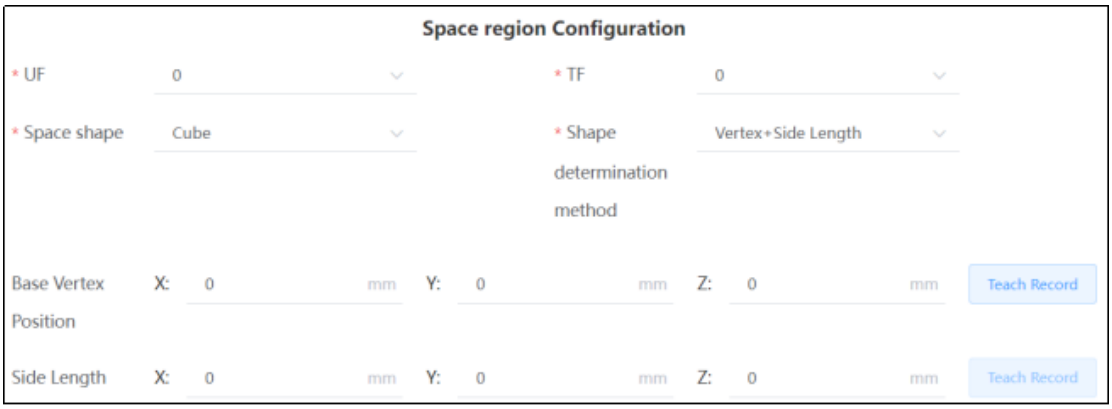


Fig. 8.14 Setting by Vertex + Side Length Method

Setting steps for Vertex + Diagonal Vertex method:

1. Move the currently activated TCP of the robot to the position set as the **vertex** of the interference region, and click on "Teach Record". Then, the current TCP position is set as a vertex of the interference cube. It is also allowed to directly input the coordinates X, Y and Z.

2. Move the currently activated TCP of the robot to the position set as the **diagonal vertex** of the interference region, and then click on "Teach Record". It is also allowed to directly input the coordinates X, Y and Z.

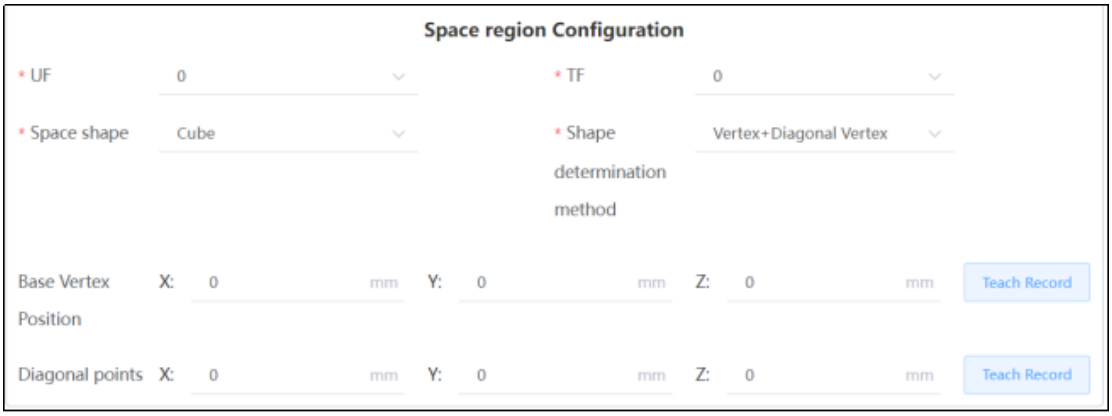


Fig. 8.15 Setting by Vertex + Diagonal Vertex Method

8.3 Reference pose

The reference pose is one or several pre-set specific poses. After this function is enabled, the system may check in real time whether the current joint angle of the robot is within a certain range of the set reference pose (the range can be set) and a

specified signal is output. This function can usually be used to inform peripheral devices that the robot is in a specific safe position or whether it is at a safe pose before starting the robot program.

It is allowed to set 10 reference poses.

Enter the Setting Screen:

Successively click "Menu Button" → "Application" → "Reference Pose" to enter the reference pose setting interface as shown in Fig. 8.16.

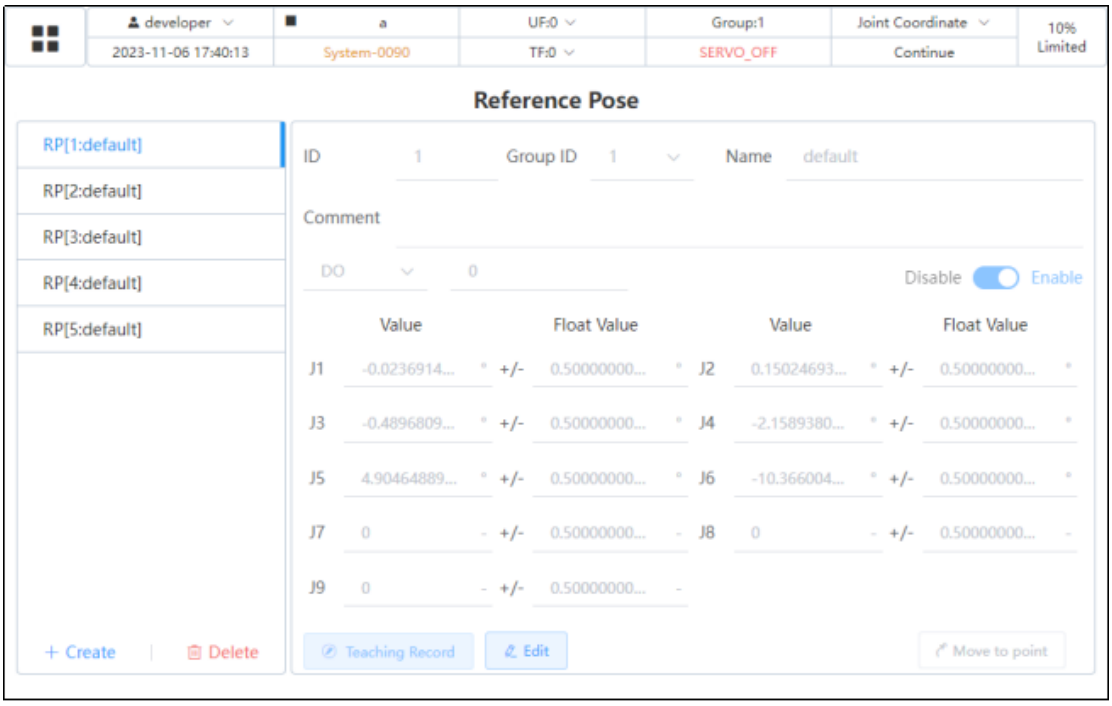


Fig. 8.16 Reference Pose Interface

Specific steps:

1. Click "Create" to create an RP.
2. Click "Edit" to set the parameters.
3. Set the digital output signal when the tool is in the reference pose, which can be selected as DO or RO. Take care to avoid duplication with other signals.
4. There are 2 methods for teaching reference poses: teach to record the current pose and directly input coordinates of the reference pose. Enter coordinates on the left and the allowable error range on the right. Never set the float value as 0, for it should basically be above 0.1.
5. Press the Activate button after setting and click to save the changes.

Fig. 8.17 Reference Pose Setting Interface

8.4 Palletizing and Depalletizing

8.4.1 Description of Palletizing Function

Palletizing refers to placing workpieces in an orderly manner layer by layer according to a certain sequence.

By teaching the placement action of one workpiece, specifying the approach and exit points, picking path mode, arrangement mode, and overlapping mode, the placement actions of all workpieces can be taught easily. In addition, the task of unloading the placed workpieces in the reverse order is called depalletizing.

Users can process all target workpieces on the pallet with minimal code or by selecting instructions.

8.4.2 Structure and Types of Palletizing

To achieve various stacking effects in palletizing, key path styles and route point information are required.

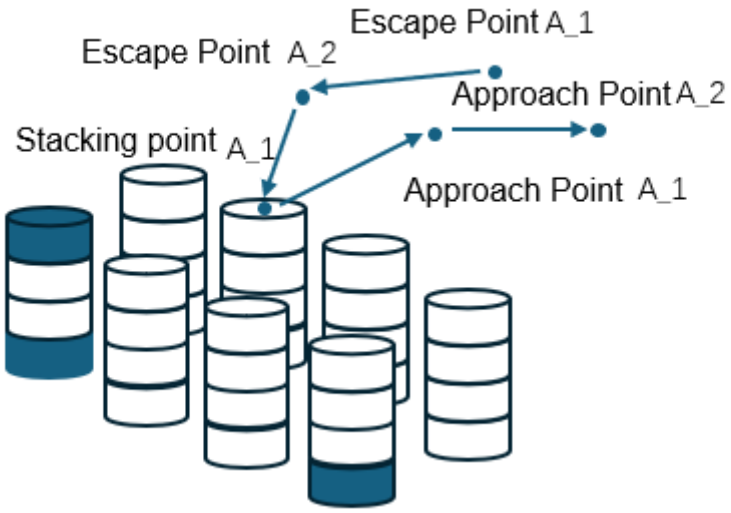


Figure 8.19 Schematic Diagram of Palletizing Path Styles

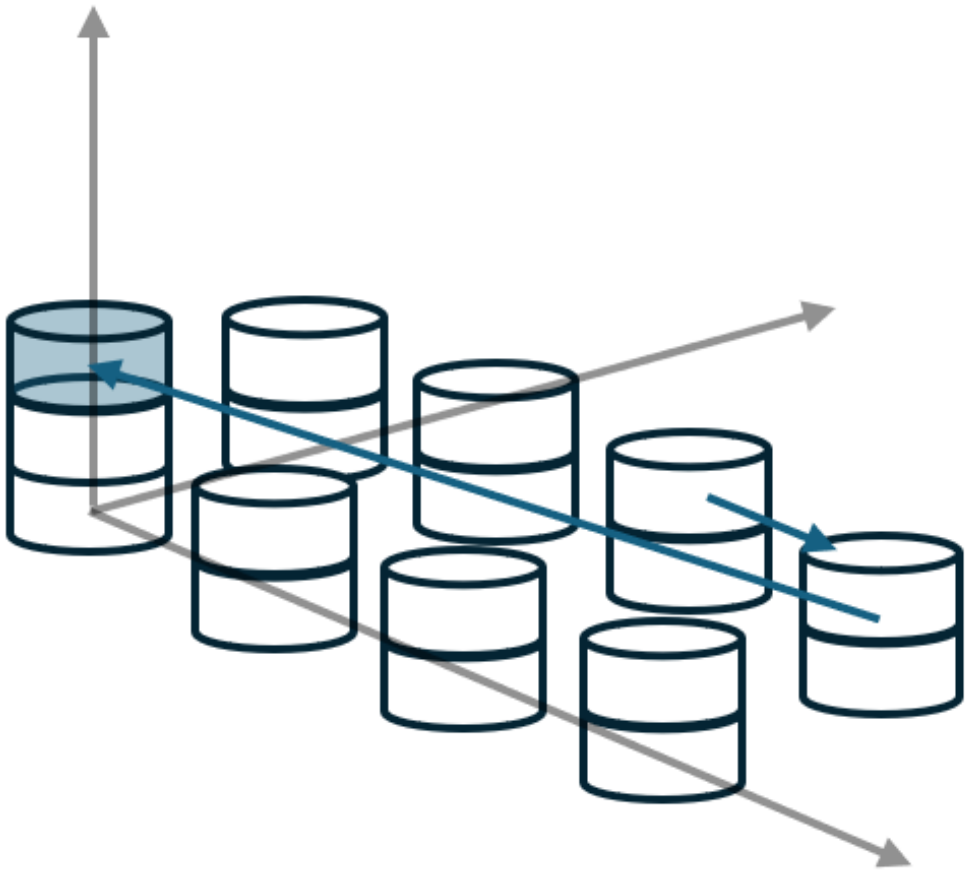
The palletizing function can realize workpieces with a fixed posture, all being [1,1,1] (the starting point of the pallet stack), and the palletizing/depalletizing types with a bottom shape of straight line/parallelogram/trapezoid and one or more entry and exit routes.

Figure 8.20 Schematic Diagram of Palletizing Shape and Projection of Workpiece Direction

Users input or teach the position of the first workpiece placed on the pallet stack, as well as the position of the last workpiece in the row, column, and layer directions respectively, and fill in the number of workpieces in the row, column, and layer of the pallet stack. The software generates a filled lattice based on these parameters.

According to the combination of rows, columns, and layers, there are 6 combinations in total, which are used for the operation order during palletizing/depalletizing. The following figure shows the stacking order in the sequence of row -> column -> layer. The starting point [1,1,1] of the pallet itself, and the directions of rows and columns are not fixed; they are controlled by the order of the taught points. The first taught point is [1,1,1], the second point determines the row direction, the third point

determines the column direction, and the fourth point determines the layer direction.



[3,2,2] → [4,2,2] → [1,1,3]

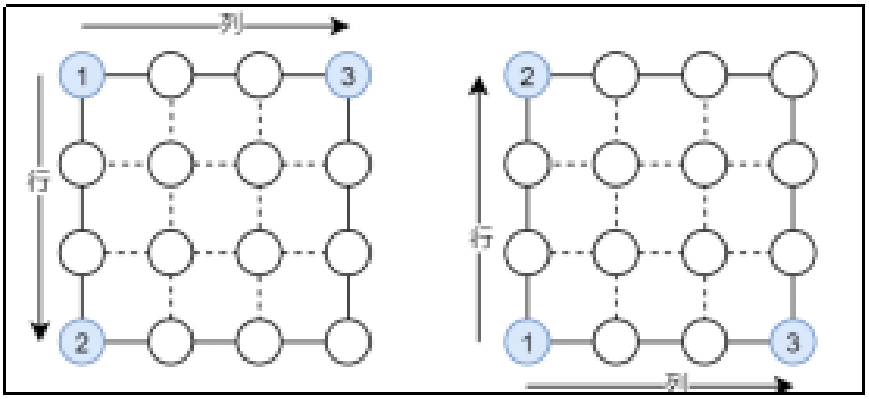


Figure 8.21 Schematic Diagram of Palletizing Order

Taking type B as an example, when teaching the pallet shape, it is necessary to teach the origin, row end, column end, highest point, and auxiliary points to set the pallet type. The following figure shows a 4-row, 2-column, 4-layer pallet without auxiliary points, requiring teaching points [1,1,1], [1,2,1], [1,1,4], [4,1,1]. When the taught points of rows, columns, and layers of the pallet shape overlap, subsequent repeated points do not need to be taught. For example:

- If the number of rows/columns/layers = [1,2,1], the points to be taught are: [1,1,1], [1,2,1];
- If the number of rows/columns/layers = [4,1,1], the points to be taught are: [1,1,1], [4,1,1];
- Example 4: If the number of rows/columns/layers = [4,2,1] without auxiliary points, the points to be taught are: [1,1,1], [4,1,1], [1,2,1];

When the shape of the first layer is a trapezoid, auxiliary points can be used to assist in teaching the pallet shape, and the filled points are filled in proportion:

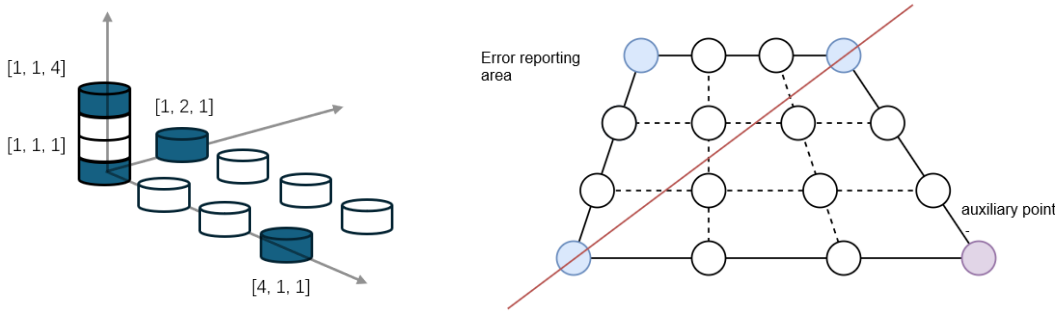


Figure 8.22 Schematic Diagram of Palletizing Auxiliary Points

Note:

Auxiliary points should fall on the right side or lower side of the red line in the following figure.

8.4.3 Palletizing Direction and Step

The palletizing/depalletizing direction affects the starting position of palletizing/depalletizing, as well as the magnitude of the change (fixed value of addition or subtraction) and the direction of change of the pallet point values during row, column, and layer stepping.

The following table shows the influence of the palletizing/depalletizing direction on the default initial position of palletizing/depalletizing and the next palletizing/depalletizing direction (without human intervention).

Basic Information		**Initial values**			**Next direction**
Stack Posture	Orientation	line	column	layer	/

Palletizing	Positive order	1	1	1	Row/column/layer values grow from small to large
	Reverse order	Total rows	Total columns	1	Row/column values decrease from large to small
Depalletizing	Positive order	Total rows	Total columns	Total layers	Row/column values decrease from large to small
	Reverse order	1	1	Total layers	Row/column/layer values grow from small to large

The step value affects the change (fixed value of addition or subtraction) of the pallet point values during row, column, and layer stepping in palletizing/depalletizing. When the addition or subtraction of the step value in the row/column/layer direction exceeds the maximum value of the row/column/layer, it is considered that the palletizing/depalletizing in the current row/column/layer direction is completed, and the palletizing/depalletizing action in the next row/column/layer direction can be performed.

For a 2-row, 2-column, 2-layer pallet with the order of row -> column -> layer for palletizing/depalletizing, the operation is performed in the following order:

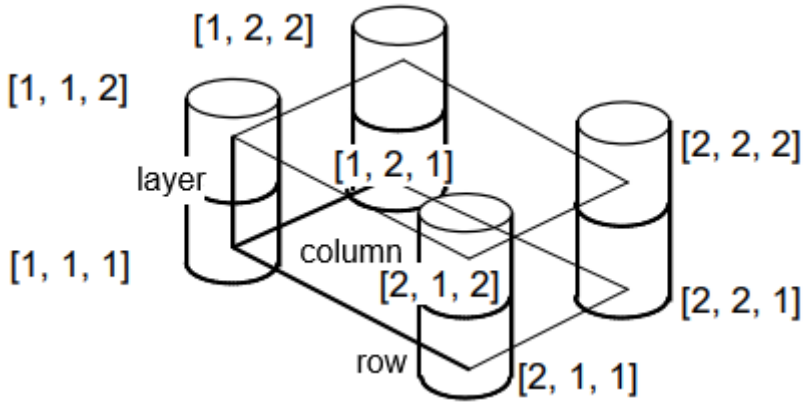


Figure 8.23 Schematic Diagram of Palletizing Auxiliary Points

Operation order table for 2-row, 2-column, 2-layer palletizing/depalletizing:

Palletizing (row->column->layer)	Depalletizing (row->column->layer)

Forward order, step value: 1	Reverse order, step value: 1	Forward order, step value: 1	Reverse order, step value: 1
[1,1,1]	[2,2,1]	[2,2,2]	[1,1,2]
[2,1,1]	[1,2,1]	[1,2,2]	[2,1,2]
[1,2,1]	[2,1,1]	[2,1,2]	[1,2,2]
[2,2,1]	[1,1,1]	[1,1,2]	[2,2,2]
[1,1,2]	[2,2,2]	[2,2,1]	[1,1,1]
[2,1,2]	[1,2,2]	[1,2,1]	[2,1,1]
[1,2,2]	[2,1,2]	[2,1,1]	[1,2,1]
[2,2,2]	[1,1,2]	[1,1,1]	[2,2,1]
[1,1,1]	[2,2,1]	[2,2,2]	[1,1,2]
…	…	…	…

Operation order table for 4-row, 2-column, 2-layer palletizing/depalletizing:

Palletizing (row->column->layer)		Depalletizing (column->row->layer)	
Forward order, step value: 2	Reverse order, step value: 2	Forward order, step value: 3	Reverse order, step value: 3
[1,1,1]	[4,2,1]	[4,2,2]	[1,1,2]
[3,1,1]	[2,2,1]	[3,2,2]	[2,1,2]
[1,2,1]	[4,1,1]	[2,2,2]	[3,1,2]
[3,2,1]	[2,1,1]	[1,2,2]	[4,1,2]
[1,1,2]	[4,2,2]	[4,2,1]	[1,1,1]

[3,1,2]	[2,2,2]	[3,2,1]	[2,1,1]
[1,2,2]	[4,1,2]	[2,2,1]	[3,1,1]
[3,2,2]	[2,1,2]	[1,2,1]	[4,1,1]
[1,1,1]	[4,2,1]	[4,2,2]	[1,1,2]
…	…	…	…

8.4.4 Palletizing Interface Configuration

To use the palletizing function, first configure the palletizing process engineering. Enter the path "Menu - Application - Palletizing Process" and complete the configuration according to the flow chart of the palletizing process engineering, as shown in the following figure:

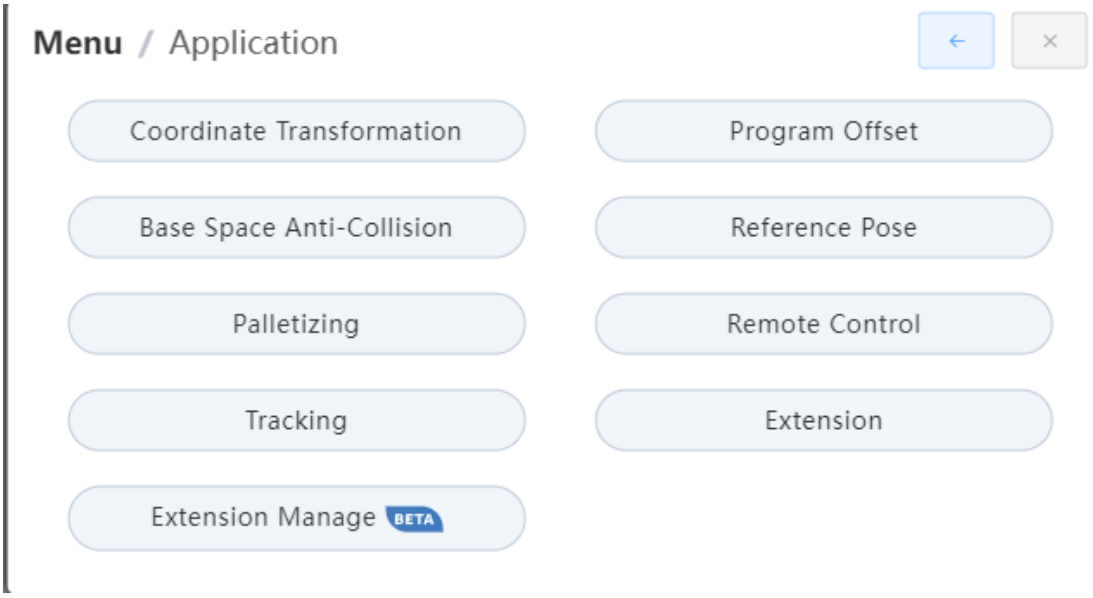


Figure 8.24 Schematic Diagram of Palletizing Process Entry and Configuration Flow

8.4.4.1 Palletizing Configuration

The palletizing process engineering can be configured only in manual mode (M, L) and when no user program resides in the memory (no user program is in a paused or running state); otherwise, only viewing is allowed, and no modifications can be made.

Enter the path "Menu - Application - Palletizing" and click "New" to create a

palletizing process engineering, as shown in the following figure:

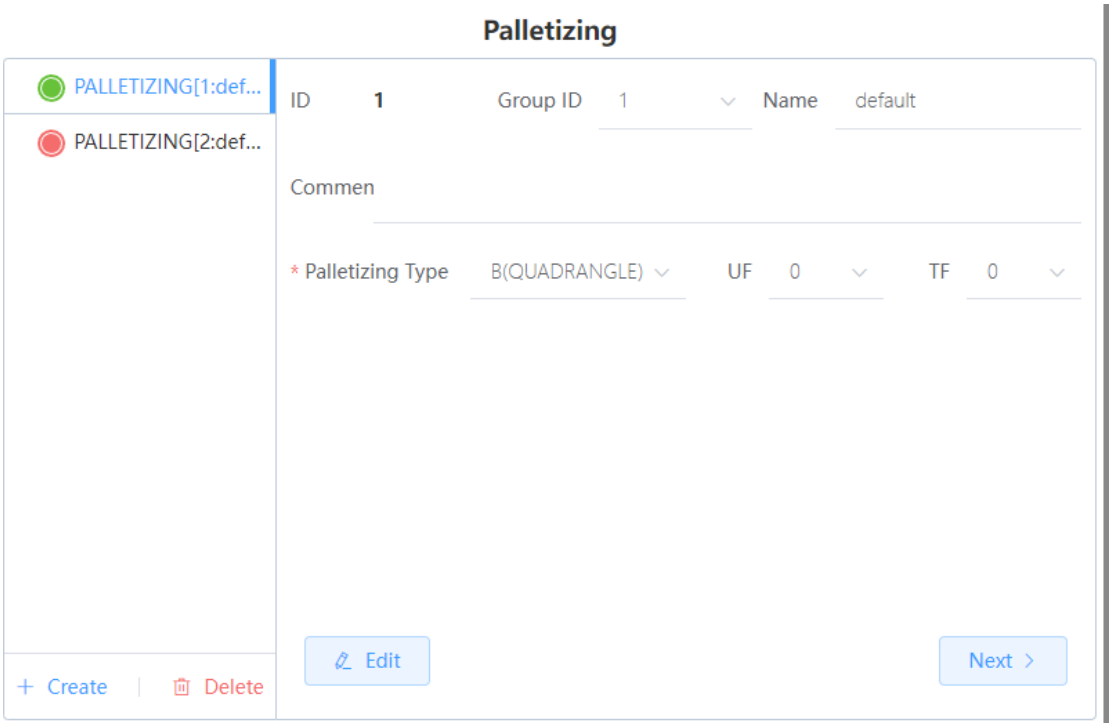


Figure 8.25 Pallet Shape and General Settings Page of Palletizing Process

Clicking "New" can add a palletizing process engineering, with a maximum of 30 process engineerings (serial numbers 1-30) supported, and the default name is "default".

At this point, you can click "Next" to view the parameters of each page, and click "Edit" to input or modify the current parameters. After modification, you can save all current data through the "Save" button, or discard all data and restore the state before editing through the "Undo" button.

8.4.4.2 Basic Data Input

Basic data corresponds to the basic palletizing information in Sections 8.4.2 and 8.4.3. Modifying the basic information will affect the validity of the taught points and the pallet shape in subsequent steps.

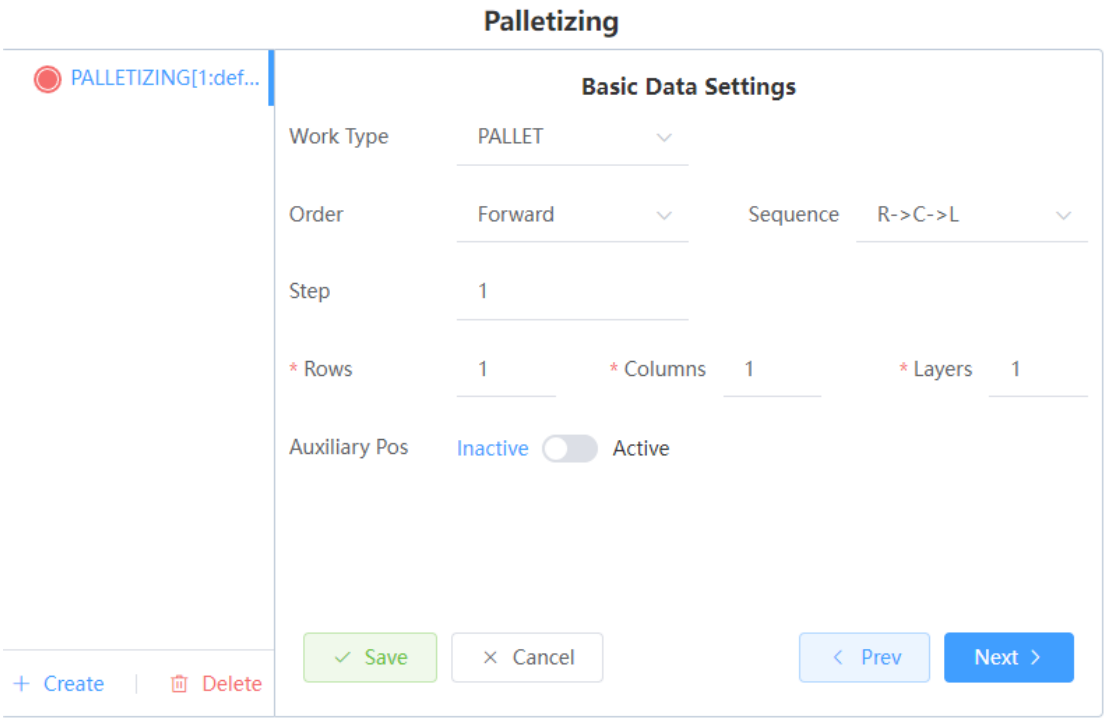


Figure 8.26 Basic Data Setting Page of Palletizing Process

Palletizing/depalletizing type (): The type is palletizing or depalletizing, default is palletizing:

- Parameter 1: Palletizing (PALLET);
- Parameter 2: Depalletizing (DEPALL);

Palletizing/depalletizing direction (): Used to determine the starting position and palletizing/depalletizing direction during palletizing or depalletizing:

- Parameter 1: Forward order ()
- Parameter 2: Reverse order ()

Palletizing/depalletizing order (): The order of rows (R), columns (C), and layers (L) during palletizing and depalletizing, with a total of 6 parameter combinations, default is row -> column -> layer (R->C->L):

- Parameter 1: Row -> Column -> Layer (R->C->L);
- Parameter 2: Row -> Layer -> Column (R->L->C);
- Parameter 3: Column -> Row -> Layer (C->R->L);
- Parameter 4: Column -> Layer -> Row (C->L->R);
- Parameter 5: Layer -> Row -> Column (L->R->C);
- Parameter 6: Layer -> Column -> Row (L->C->R);

8.4.4.3 Teaching Pallet Shape

Teaching pallet shape interface for type B pallet:

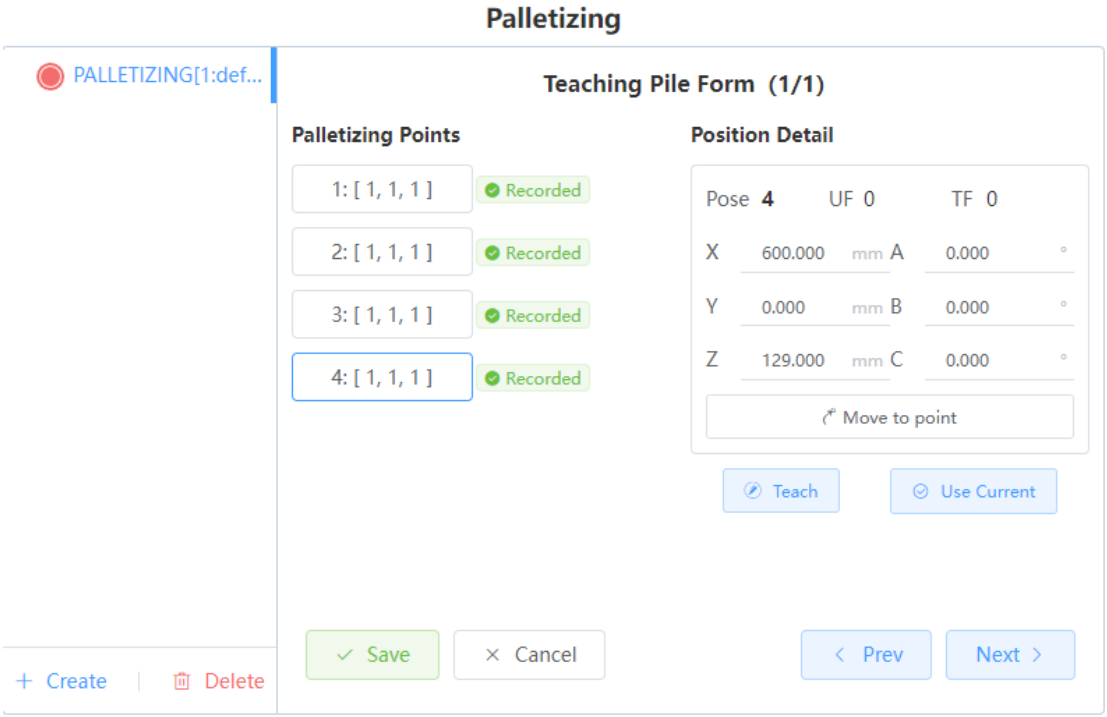


Figure 8.27 Pallet Style Setting Page of Palletizing Process

Teaching pallet shape (1/1): There is only one teaching style for type B palletizing, so it is displayed as (1/1). The posture of the pallet shape teaching point must be consistent with [1,1,1]; otherwise, a pop-up error will appear, and the teaching of this point is invalid. The form of palletizing points can refer to the point display of the user coordinate system.

When the auxiliary point in 8.4.4.2 is disabled: Teach 3 bottom points and 1 top point in order; when the auxiliary point is enabled: Teach 4 bottom points and 1 top point in order, as shown in the following figure:

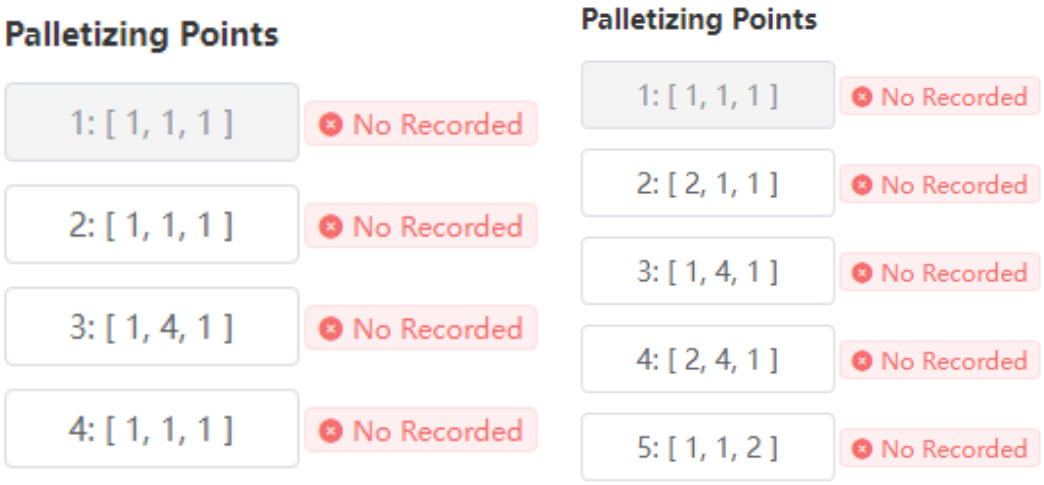


Figure 8.28 List of Teaching Points with and without Auxiliary Points in Palletizing Process

8.4.4.4 Setting Path Styles

It is necessary to set path conditions for each path to judge the path selection when handling specific points, and teach the corresponding path points; each path style can have 0 to 8 approach points or exit points. When executing the palletizing/depalletizing of a specific pallet point, the path style with successful condition matching will be called.

Add a path style in order, default is [*,*,*], and three types of parameters can be filled in. The priority is: row = column = layer, direct specification > remainder specification > fuzzy matching:

Direct specification	1 to 255 integers	Specific row/column/layer position
Remainder specification method	m-n integers	For example, 3-1 means that the row/column/layer value of the pallet point divided by 3 with a remainder of 1 indicates a successful match
Fuzzy matching	*	Indicates any row/column/layer of the pallet point

Click the red button to enter the teaching page, and it will be displayed as a blue

button after teaching is completed.

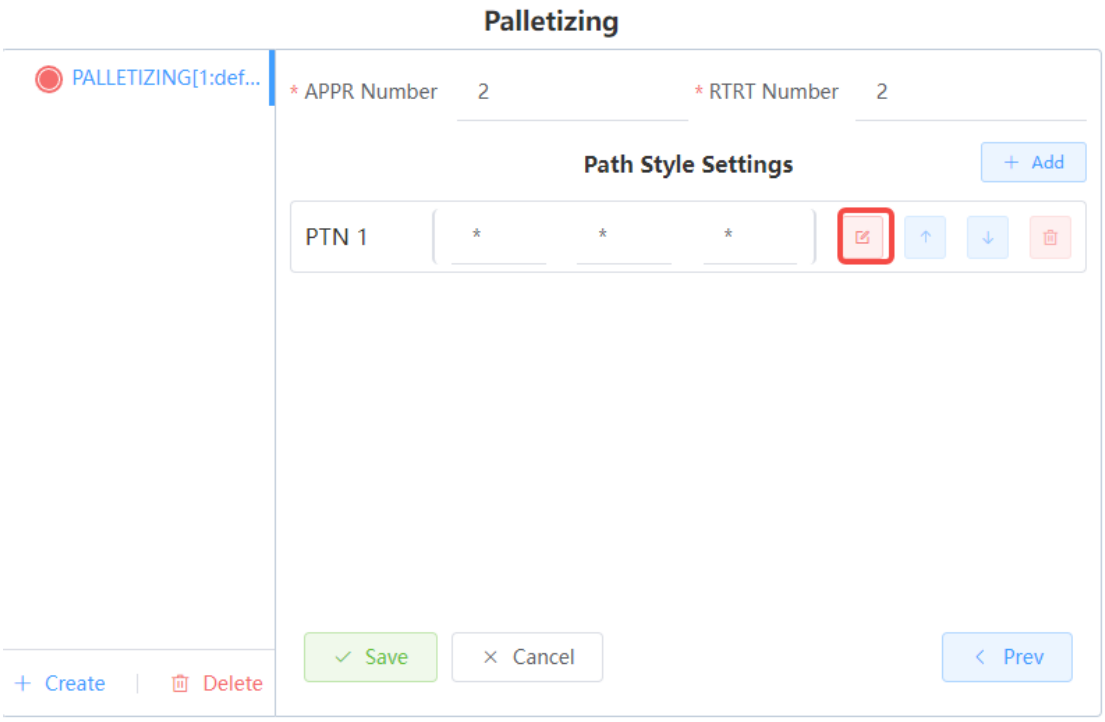


Figure 8.29 Path Style Setting Page of Palletizing Process

According to the number of approach points and exit points, the corresponding number of approach points A_i and exit points R_i will be provided during path style editing. All points need to be filled in completely before confirmation.

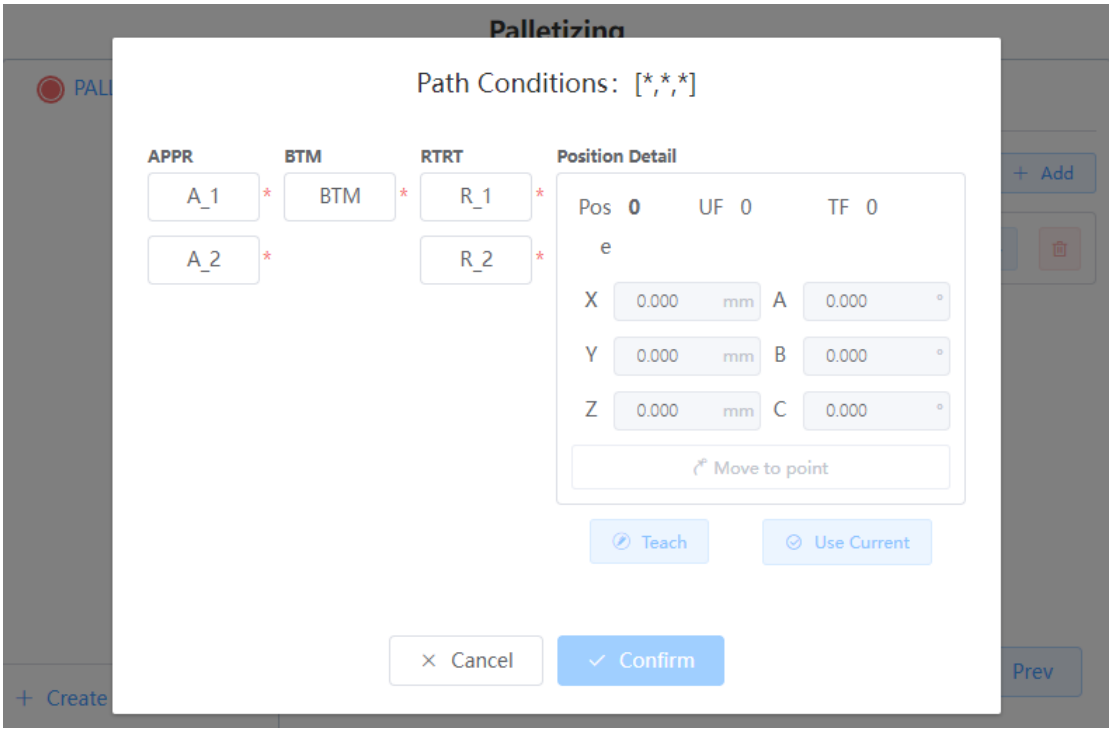


Figure 8.29 Path Condition Setting Page of Palletizing Process

8.4.4.5 Teaching Path Points

Path styles that have not been taught completely are highlighted in red, and those that have been taught completely are highlighted in blue;

In the same path style, the path conditions have no relationship with the taught path points. Modifying path conditions or teaching paths does not affect each other; only during the operation of the user program, the relevant path points are indexed according to the matched path conditions;

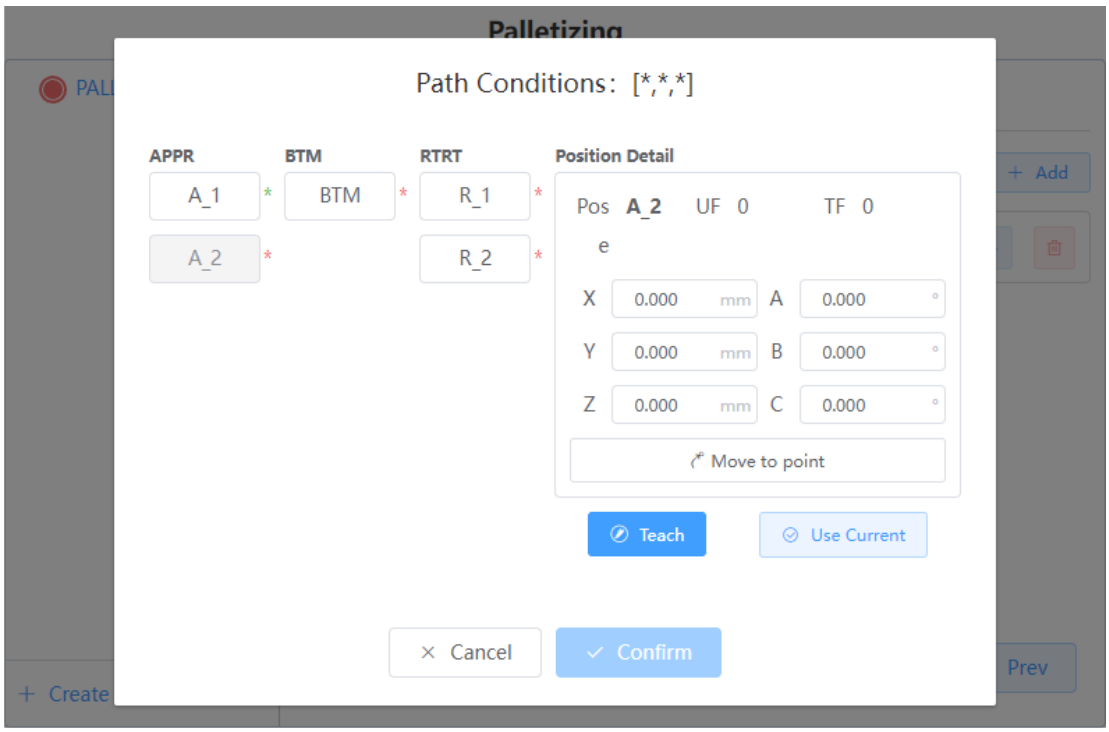


Figure 8.30 Path Style Setting Page of Palletizing Process

Note:

There is only one pick-and-place point. In 8.4.4.4, if the number of approach points and/or exit points is increased, the original point data in the path conditions of this section will be retained, and points will be added in order, with all points turning red; if the number of approach points and/or exit points is reduced, deletion starts from the largest serial number, the original point data that is not deleted is retained, and all points turn red to prompt for confirmation and saving.

Finally, click "Complete" to overwrite the historical data; otherwise, the configured parameters will be discarded according to the confirmation option in the pop-up window when leaving the current PALLETIZING[X:YYY].

8.4.5 Palletizing Program

8.4.5.1 Inserting/Editing Palletizing Registers

Click "Insert Instruction" → "Assignment and IO" → "PL" in sequence to enter the palletizing register page, as shown in the following figure.

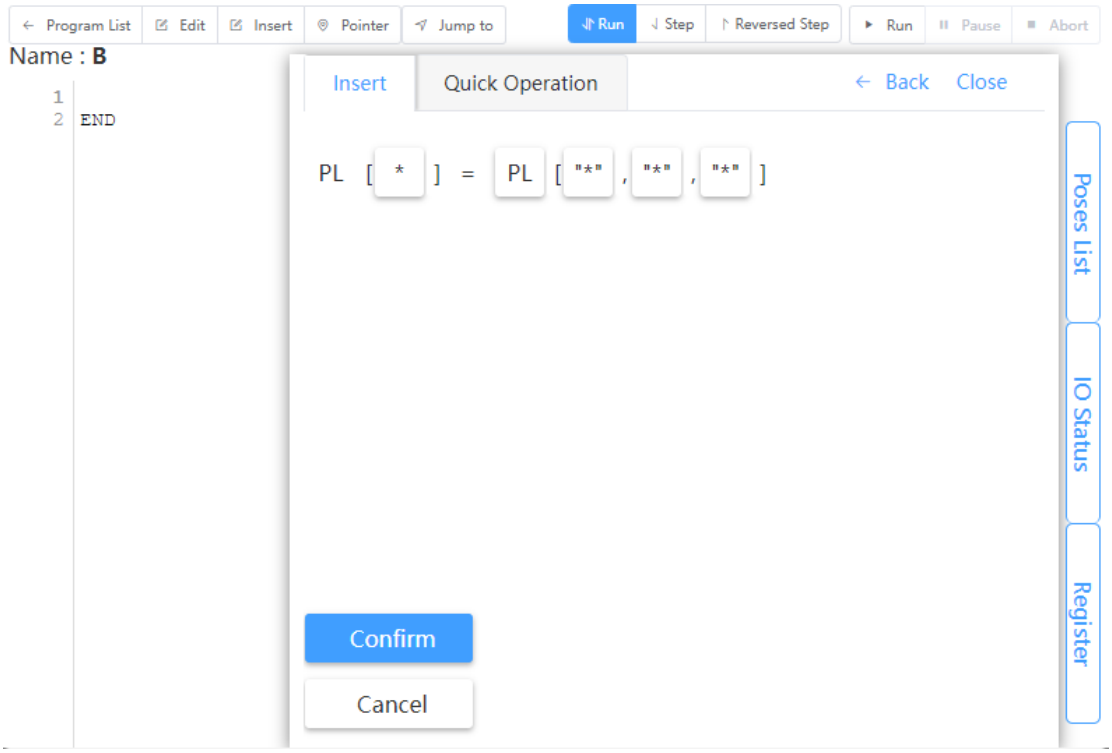


Figure 8.31 Path Style Setting Page of Palletizing Process

PL[*] can only select the index of the completed palletizing process engineering; the index of PL[*] for uncompleted palletizing process engineering is invisible; when assigning values, the values of I, J, K cannot exceed the row, column, and layer value configurations of the bound palletizing process engineering; when editing palletizing registers, refer to the insertion and assignment rules of PL[*] in 3.4.7 Palletizing Registers.

8.4.5.2 Inserting Palletizing Format Instructions

Click "Insert Instruction" → "Palletizing Instruction" → "PALLETIZING" in sequence to enter the palletizing instruction selection page, as shown in the following figure.

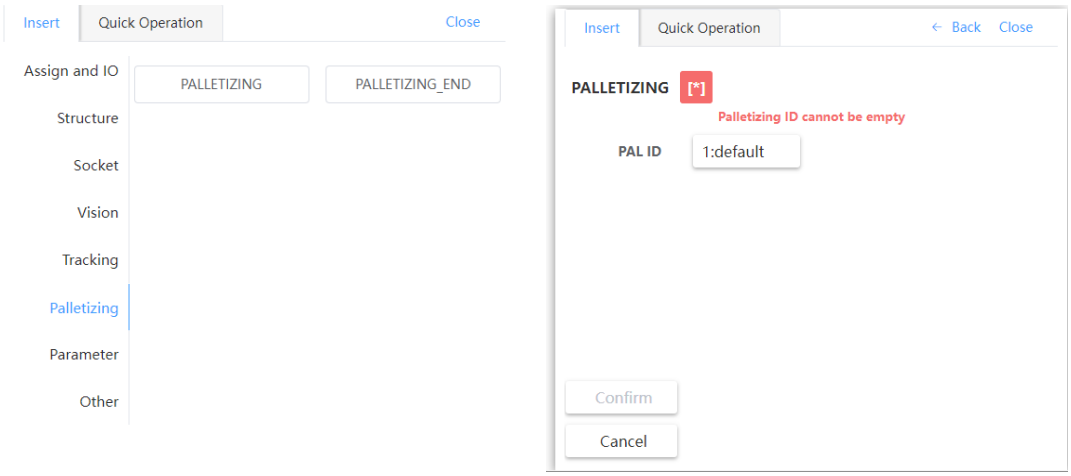


Figure 8.32 Path Style Setting Page of Palletizing Process

PALLETIZING [*] can only select the index of the completed palletizing process engineering; the index of PALLETIZING [*] for uncompleted palletizing process engineering is invisible.

8.4.5.3 Inserting Palletizing Action Instructions

After the selected index is determined, the action instruction can be inserted by clicking "Insert Instruction" → "Palletizing Instruction" → "PALLETIZING" in sequence. The default action instruction format is:

MOVEJ PAL[x，xx]，30%，FINE：

As shown in the following figure:

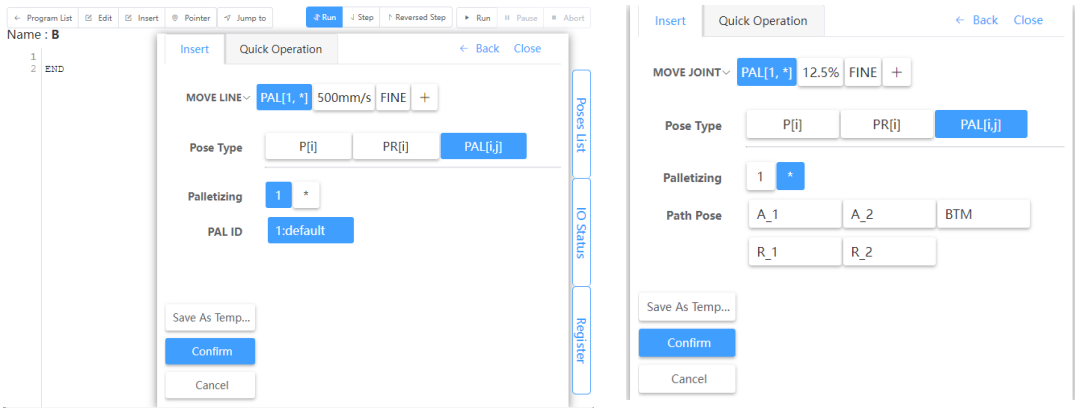


Figure 8.33 Path Style Setting Page of Palletizing Process

When fine-tuning the workpiece posture for palletizing/depalletizing, it can be realized by using the OFFSET of the palletizing action instruction. If combined with path

conditions, the workpiece posture of a specific row/column/layer or specific [I, J, K] can be further refined; example:

IF PL[1] = [1,1,1]

MOVEL PAL[1，BTM]，500mm/s，FINE

ELSE IF PL[1] = [2,*,*]

MOVEL PAL[1，BTM]，500mm/s，FINE，OFFSET PR[1]

Click "Insert Instruction" → "Palletizing Instruction" → "PALLETIZING" in sequence to enter the palletizing instruction selection page, as shown in the following figure, and select the PALLETIZING_END instruction.

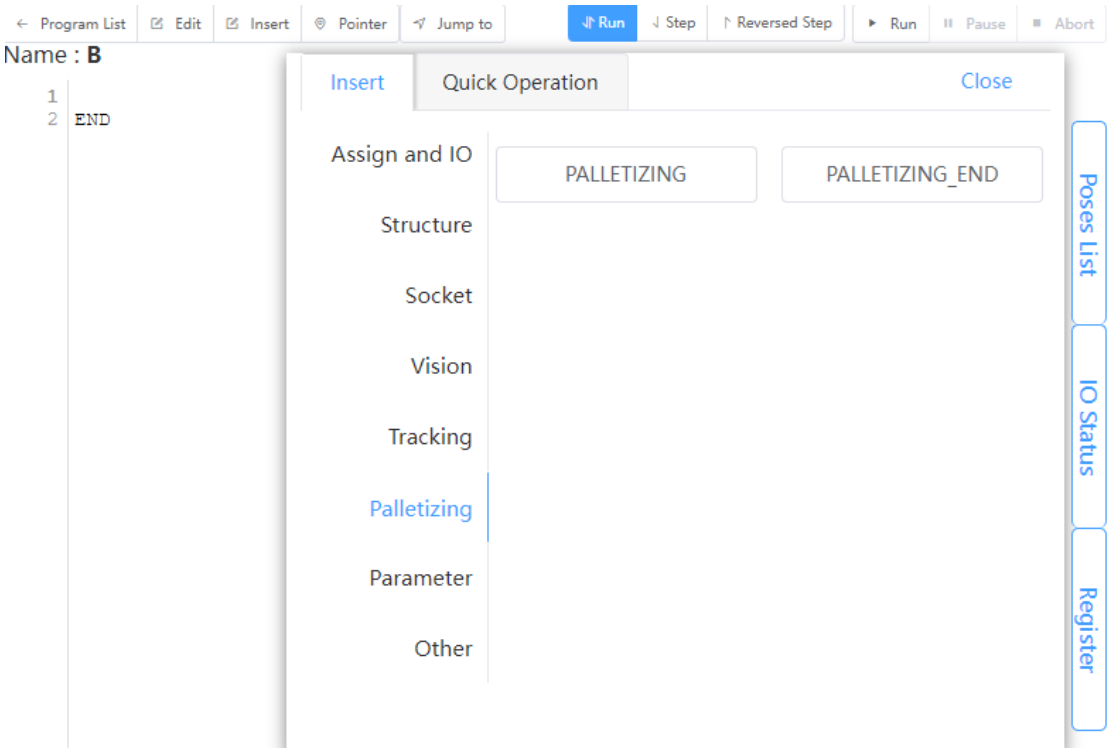


Figure 8.34 Path Style Setting Page of Palletizing Process

8.4.5.4 Pseudocode Example

#Initialization

DO[1: Close Vacuum] = OFF

MOVEL P[1: Initial Position], 500mm/s, FINE

// Specify the starting position of palletizing and initialize the palletizing register

PL[1] = [1, 1, 1]

```
WHILE 1
#Feeding Path
MOVEL P[2: Feeding Position], 500mm/s, FINE
DO[1: Open Vacuum] = ON
MOVEL P[3: Safe Movement Position], 500mm/s, FINE
#Start of Palletizing Instruction
PALLETIZING [1: Test], B, LOAD, 3, 4, 5
#Move to Safe Palletizing Position
MOVEJ PAL[1, A_2], 30%, SD10
MOVEL PAL[1, A_1], 500mm/s, FINE
#Palletizing
MOVEL PAL[1, BTM], 500mm/s, FINE
#Release Gripper
DO[1: Close Vacuum] = OFF
Move to Safe Exit Position
MOVEL PAL[1, R_1], 500mm/s, SD10
MOVEJ PAL[1, R_2], 50%, FINE
#End of Palletizing, palletizing register increments by step value
PALLETIZING_END [1: Test]
#Move to Safe Return Position
MOVEJ P[4: Safe Return Point], 500mm/s, FINE
MOVEJ P[1: Initial Position], 500mm/s, FINE
IF PL[1] = [3,4,5]
#Palletizing is full, jump
GOTO LABEL[1:]
END IF
END WHILE
LABEL[1:]
END
```

8.4.6 Precautions

When teaching the palletizing process engineering, no user program can be running or paused;

The three instructions (palletizing format instruction, approach point/pick-and-place point/exit point action instruction, and palletizing end instruction) can only function if they exist in the same program. Even if only one instruction is copied to a subprogram, it will not work properly;

In programs containing palletizing instructions, the coordinate system transformation function cannot be used;

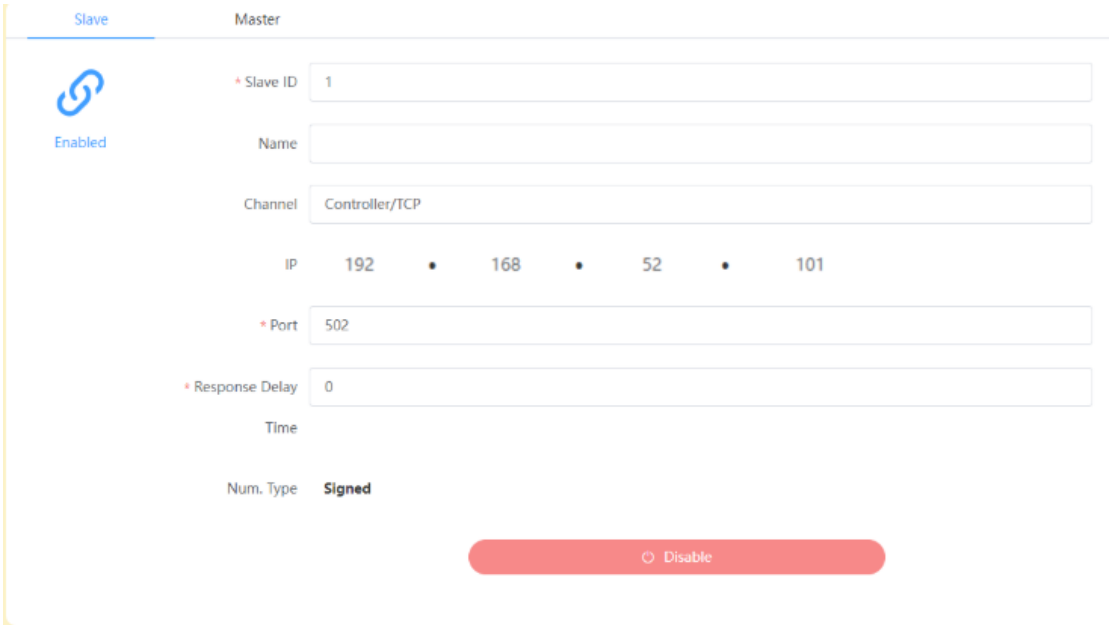
8.5 Remote control

Remote control is suitable for scenarios where the teach pendant is not used. The host computer can enable functions such as servo activation, excitation, start/resume/pause programs, modify the global speed, jog for teaching, record the current position, and modify the coordinate system through mapping digital I/O or bus I/O.

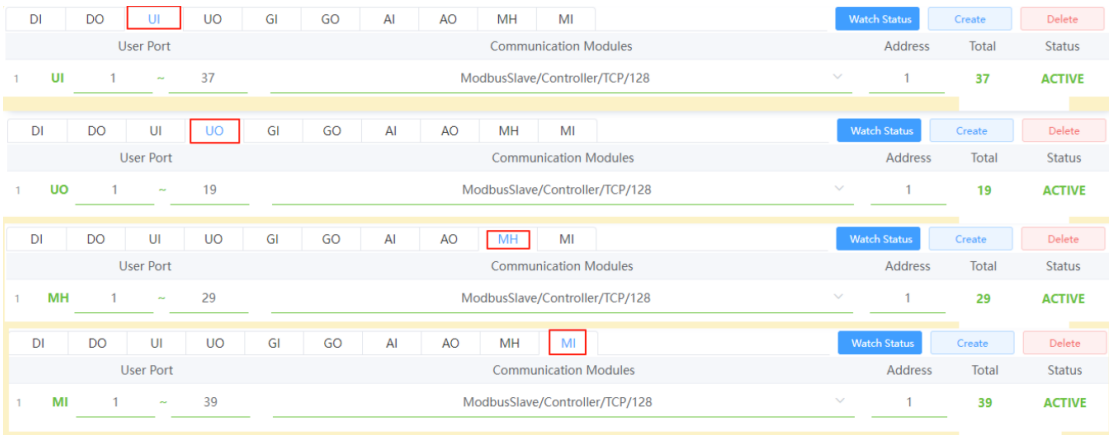
Bus Configuration and I/O Mapping

Before configuring the remote control function mapping, the bus should be configured first, and the addresses of dedicated I/O and MH/MI registers should be allocated.

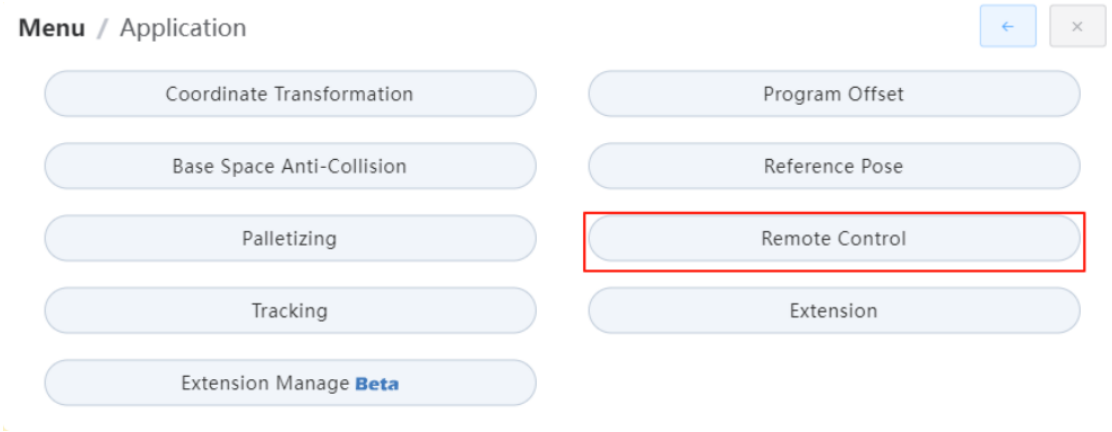
Configuration path: Menu/Communication/Bus Configuration. Select the signed data type and click Enable.



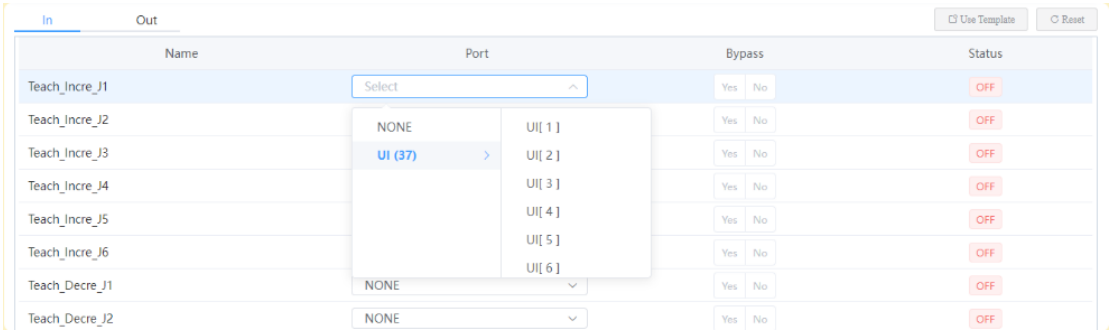
Allocate UI/UO and MH/MI addresses.



After allocating the address space, you can enter the mapping configuration of specific functions by sequentially entering Menu/Application/Remote Control.



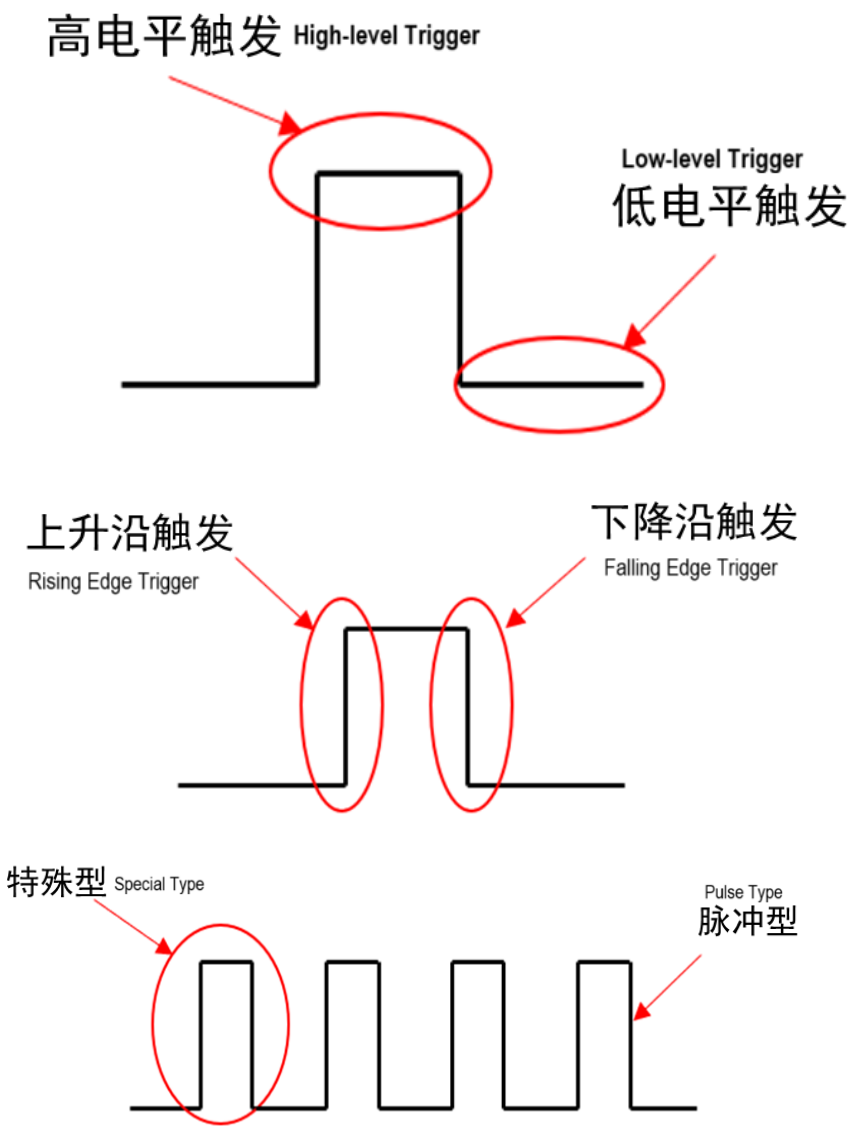
You can configure the dedicated I/O (UI/UO) or registers (MH/MI) for function or data mapping through options.



8.5.1 Signal Types

Considering factors such as safety, the remote control provides 6 types of level signals:

Serial Number	Signal Type	Description
1	Low-level Trigger	Trigger the corresponding operation by providing a normal low level.
2	High-level Trigger	Trigger the corresponding operation by providing a normal high level.
3	Rising Edge Trigger	First provide a low level, then a high level to trigger the corresponding operation. It is recommended to have an interval of more than 2 ms.
4	Falling Edge Trigger	First provide a high level, then a low level to trigger the corresponding operation. It is recommended to have an interval of more than 2 ms.
5	Special Type	A high-level trigger mode from the rising edge to the falling edge, suitable for the manual teaching scenario. Provide a high level when pressed and a low level when released.
6	Pulse Type	A trigger mode with periodic switching between high and low levels, suitable for scenarios such as detecting the online status of the device to prevent the device from false death.



8.5.2 Input Address Mapping Table

Serial Number	Name	Signal Function	Signal Type	Robot Signal Address	Remarks
1	Servo Enable	Servo Activation	Special Type	UI[1]	In application, a high level must be continuously provided.

2	Pause Request	Pause	Falling Edge Trigger	UI[2]	The machine can run only when the normal state is high level.
3	Reset	Alarm Reset	Rising Edge Trigger	UI[3]	This operation needs to be carried out under the premise that "Servo Enable" is ON.
4	Start&Restart	Program Start/Resume	Falling Edge Trigger	UI[4]	Generate a falling edge by providing a pulse signal.
5	Abort Program	Program Termination Request	Falling Edge Trigger	UI[5]	In application, a high level must be continuously provided. Similar to UI2, it is an event signal. When it is OFF, there is a risk of system freeze caused by multiple trigger signals.
6	Selection Strobe	Trigger	High-level Trigger	UI[6]	Used to record the program number. The system reads the signals of Program Selection 1 - 6 during the high-level duration of the trigger signal.
7	MPLCS Start	Main Program Number Start	Falling Edge Trigger	UI[7]	Used to execute the program. The ON state should be maintained for more than 100 ms, and then when it changes to OFF, the system recognizes the falling edge and the program starts.
8	Program Selection 1	Program Number	High-level	UI[8]	

			Trigger		
9	Program Selection 2	Program Number	High-level Trigger	UI[9]	
10	Program Selection 3	Program Number	High-level Trigger	UI[10]	
11	Program Selection 4	Program Number	High-level Trigger	UI[11]	
12	Program Selection 5	Program Number	High-level Trigger	UI[12]	
13	Program Selection 6	Program Number	High-level Trigger	UI[13]	
14	Drag	Drag Enable	High-level Trigger	UI[14]	Only for collaborative robots.
14	Speed Incre	Speed Increase V+	Rising Edge Trigger	UI[15]	The global speed increases by 5% each time.
15	Speed Decre	Speed Decrease V-	Rising Edge Trigger	UI[16]	The global speed decreases by 5% each time.
16	Goto Home	Return to Origin	Special Type	UI[17]	The ID of the home point needs to be provided. After this operation is executed, the current direct motion mode will change to MovJ.

17	Switch Teach Manual Mode	Switch to Manual Teaching Mode	Rising Edge Trigger	UI[18]	
18	Switch Teach Manual Limit Mode	Switch to Manual Speed-limited Teaching Mode	Rising Edge Trigger	UI[19]	
19	Switch Auto Mode	Switch to Auto Mode	Rising Edge Trigger	UI[20]	
20	Teach Incre J1	Teaching J1/X +	Special Type	UI[21]	
21	Teach Incre J2	Teaching J2/Y +	Special Type	UI[22]	
22	Teach Incre J3	Teaching J3/Z +	Special Type	UI[23]	
23	Teach Incre J4	Teaching J4/A +	Special Type	UI[24]	
24	Teach Incre J5	Teaching J5/B+	Special Type	UI[25]	
25	Teach Incre J6	Teaching J6/C+	Special Type	UI[26]	
26	Teach Decre J1	Teaching J1/X –	Special Type	UI[27]	
27	Teach Decre J2	Teaching J2/Y –	Special Type	UI[28]	

28	Teach Decre J3	Teaching J3/Z –	Special Type	UI[29]	
29	Teach Decre J4	Teaching J4/A –	Special Type	UI[30]	
30	Teach Decre J5	Teaching J5/B-	Special Type	UI[31]	
31	Teach Decre J6	Teaching J6/C-	Special Type	UI[32]	
32	Write Cur Point To Pr Reg	Write the Current Position to the Current PR Variable	Special Type	UI[33]	
33	Write input Point To Pr Reg	Write the Input Position to the Current PR Variable	Special Type	UI[34]	
34	Move To Pr Point	Directly Move to the Current PR Variable Position	Special Type	UI[35]	
35	Read Pr Reg	PR Variable Read Success	Rising Edge Trigger	UI[36]	
36	Bus Heartbeat	Bus Communication	Pulse Signal	UI[37]	Used to determine the online status of both parties and switch to the

		Heartbeat Signal			PC mode. If the input heartbeat is not configured or the input heartbeat pulse does not change within 5 s, it is considered offline.
37	Home Id	ID of the Home Point		MH1	For this item and the following items, the IO group or MH register mapping should be selected.
38	Cur Pr Reg No	ID of the Current PR Variable		MH2	
39	Pr Arm Param 1	Arm Parameter 1 Written to the PR Variable		MH3	Configuration of the J5 joint of a 6-axis robot 1: Wrist flips downward -1: Wrist flips upward
40	Pr Arm Param 2	Arm Parameter 2 Written to the PR Variable		MH4	Configuration of the J2 joint of a 4-axis Scara robot 1: Arm on the right -1: Arm on the left
41	Pr Arm Param 3	Arm Parameter 3 Written to the PR Variable		MH5	Configuration of the J3 joint of a 6-axis robot 1: Arm above (elbow above the line connecting the wrist and the 2-axis) -1: Arm below (elbow below the line connecting the wrist and the 2-axis)
42	Pr Arm Param 4	Arm Parameter 4 Written		MH6	Configuration of the J1 joint of a 6-axis robot 1: Arm in the

		to the PR Variable			front -1: Arm in the back
43	Pr Rotate Param 1	Rotation Parameter 1 Written to the PR Variable		MH7	Number of rotations of the 1-axis 1: 180° ~ 530° 0: -179° ~ 179° -1: -539° ~ -180°
44	Pr Rotate Param 2	Rotation Parameter 2 Written to the PR Variable		MH8	Number of rotations of the 2-axis 1: 180° ~ 530° 0: -179° ~ 179° -1: -539° ~ -180°
45	Pr Rotate Param 3	Rotation Parameter 3 Written to the PR Variable		MH9	Number of rotations of the 3-axis 1: 180° ~ 530° 0: -179° ~ 179° -1: -539° ~ -180°
46	Pr Rotate Param 4	Rotation Parameter 4 Written to the PR Variable		MH10	Number of rotations of the 4-axis 1: 180° ~ 530° 0: -179° ~ 179° -1: -539° ~ -180°
47	Pr Rotate Param 5	Rotation Parameter 5 Written to the PR Variable		MH11	Number of rotations of the 5-axis 1: 180° ~ 530° 0: -179° ~ 179° -1: -539° ~ -180°
48	Pr Rotate Param 6	Rotation Parameter 6 Written to the PR Variable		MH12	Number of rotations of the 6-axis 1: 180° ~ 530° 0: -179° ~ 179° -1: -539° ~ -180°
49	Pr COORDIN	Coordinate System Written to		MH13	0 selects the joint coordinate system 1 selects the Cartesian

	ATE	the PR Variable			coordinate system
50	Pr Point J1	J1/X Coordinate Written to the PR Variable		MH14 - MH15	
51	Pr Point J2	J2/Y Coordinate Written to the PR Variable		MH16 - MH17	
52	Pr Point J3	J3/Z Coordinate Written to the PR Variable		MH18 - MH19	
53	Pr Point J4	J4/A Coordinate Written to the PR Variable		MH20 - MH21	
54	Pr Point J5	J5/B Coordinate Written to the PR Variable		MH22 - MH23	
55	Pr Point J6	J6/C Coordinate Written to the PR Variable		MH24 - MH25	
56	Manual Frame	Coordinate System		MH26	Modifying the coordinate system will affect the display of the current

		Selection			position output, that is, the same position has different representations in different coordinate systems. 0 Joint coordinate system 1 Base coordinate system 2 World coordinate system 3 User coordinate system 4 Tool coordinate system 5 RTCP user coordinate system 6 RTCP tool coordinate system
57	Tool Frame	Tool Coordinate System Number Selection		MH27	Modifying the user coordinate system will affect: the result of writing the current position to the PR in the Cartesian coordinate system, the final position and motion trajectory of jogging to the PR point, and the output result.
58	User Frame	User Coordinate System Number Selection		MH28	Modifying the user coordinate system will affect: the result of writing the current position to the PR in the Cartesian coordinate system, the final position and motion trajectory of jogging to the PR point, and the output result.
59	Global Speed	Global Speed Setting		MH29	1 - 10,000 represents 0.01% - 100%

8.5.3 Output Address Mapping Table

Serial Number	Name	Signal Function	Robot Signal Address	Remarks
1	CMDENBLE	Status signal allowing the peripheral device to control the robot	UO[1]	Output high level when the following conditions are met: 1) The robot's operating status is "On-Standby". 2) It is in the "Auto" mode. 3) The program execution mode does not select "Single-step Execution" or "Reverse Execution". (When this signal is high, it means that the program can be started or paused and resumed using the "Program Start Method in Auto Mode". Specifically, it depends on whether it is currently in the Paused state.)
2	Paused	"Paused" status signal	UO[2]	This signal is ON (i.e., the robot is paused) when the program execution status is in the "Paused" state.
3	FAULT	Alarm signal	UO[3]	The alarm signal is output when an alarm occurs in the system and can be reset through RESET. Note: When the system issues a warning alarm (Warning), this

				signal is not output.
4	Program Running	Program is running signal	UO[4]	When it is ON, it indicates that the program is running, that is, the program execution status is "Executing".
5	Servo Status	Servo status signal	UO[5]	This signal is set to high level when the robot's operating status is "Working", "On-Standby", or "Servo-ON", and set to low level when it is "Servo-OFF".
6	Selection Check Request	Program selection confirmation	UO[6]	Only valid when the "Program Start Method in Auto Mode" is set to "MPLCS".
7	MPLCS Start Done	Main program number start completed	UO[7]	Only valid when the "Program Start Method in Auto Mode" is set to "MPLCS".
8	Selection Confirm 1	Program 1 selection confirmation	UO[8]	Only valid when the "Program Start Method in Auto Mode" is set to "MPLCS". After receiving the Selection Strobe signal, the robot controller will read the status of UI[5] - UI[10] and feed it back to the host for confirmation.
9	Selection Confirm 2	Program 2 selection confirmation	UO[9]	

10	Selection Confirm 3	Program 3 selection confirmation	UO[10]	
11	Selection Confirm 4	Program 4 selection confirmation	UO[11]	
12	Selection Confirm 5	Program 5 selection confirmation	UO[12]	
13	Selection Confirm 6	Program 6 selection confirmation	UO[13]	
14	Bus Heartbeat	Bus communication heartbeat signal	UO[14]	Pulse signal
15	Write Pr Reg Status	PR variable write success or failure	UO[15]	Output ON indicates that the PR variable is written successfully, and OFF indicates that the PR variable is written failed; before the write is triggered, the previous state is maintained.
16	Teach Manual Mode	Manual teaching mode output	UO[16]	
17	Teach Manual Limit	Manual speed-limited teaching mode output	UO[17]	
18	Auto Mode	Auto mode output	UO[18]	

19	Robot Moving	Robot is moving signal	UO[19]	Output ON when the robot is moving and OFF when it reaches the specified position.
20	Pr Arm Param 1	Arm parameter 1 read from the PR variable	MI1	
21	Pr Arm Param 2	Arm parameter 2 read from the PR variable	MI2	
22	Pr Arm Param 3	Arm parameter 3 read from the PR variable	MI3	
23	Pr Arm Param 4	Arm parameter 4 read from the PR variable	MI4	
24	Pr Rotate Param 1	Rotation parameter 1 read from the PR variable	MI5	
25	Pr Rotate Param 2	Rotation parameter 2 read from the PR variable	MI6	
26	Pr Rotate Param 3	Rotation parameter 3 read from the PR variable	MI7	
27	Pr Rotate	Rotation	MI8	

	Param 4	parameter 4 read from the PR variable		
28	Pr Rotate Param 6	Rotation parameter 5 read from the PR variable	MI9	
29	Pr Rotate Param 6	Rotation parameter 6 read from the PR variable	MI10	
30	Pr Coordinate	Coordinate system read from the PR variable	MI11	
31	Pr Point J1	J1/X coordinate read from the PR variable	MI12 - MI13	
32	Pr Point J2	J2/Y coordinate read from the PR variable	MI14 - MI14	
33	Pr Point J3	J3/Z coordinate read from the PR variable	MI16 - MI17	
34	Pr Point J4	J4/A coordinate read from the PR variable	MI18 - MI19	
35	Pr Point J5	J5/B coordinate read from the	MI20 - MI21	

		PR variable		
36	Pr Point J6	J6/C coordinate read from the PR variable	MI22 - MI23	
37	Cur Point J1	Current position J1/X coordinate	MI24 - MI25	
38	Cur Point J2	Current position J2/Y coordinate	MI26 - MI27	
39	Cur Point J3	Current position J3/Z coordinate	MI28 - MI29	
40	Cur Point J4	Current position J4/A coordinate	MI30 - MI31	
41	Cur Point J5	Current position J5/B coordinate	MI32 - MI33	
42	Cur Point J6	Current position J6/C coordinate	MI34 - MI35	
43	Global Speed	Current global speed	MI36	1 - 10,000 represents 0.01% - 100%
44	Manual Frame	Current coordinate system	MI37	
45	Tool Frame	Current tool coordinate system	MI38	

		number		
46	User Frame	Current user coordinate system number	MI39	

8.6 Secondary Development

Agilebot Robot Development Center: https://dev-docs.sh-agilebot.com/

SDK

The Robot SDK (Software Development Kit) is a collection of tools designed to help developers build robot applications more efficiently. It typically includes resources such as Application Programming Interfaces (APIs), code examples, documentation, and debugging tools, enabling developers to control the robot's hardware, perceive the environment, or implement intelligent decision-making without writing code from scratch.

ROS

The Robot's ROS (Robot Operating System) development environment is a set of open-source frameworks and tools specifically designed for robot software development. Its core goal is to simplify the construction, testing, and deployment of robot systems.

Plugins

The plugin system is a software secondary development platform provided by Agilebot Robots. It allows you to focus on your specific business scenarios, breaking the limitation of fixed functions in traditional robots. Through the modular components and interfaces provided by the platform (such as motion control commands, register call interfaces, UI interaction components, etc.), you can quickly build your own application framework. Thus, you can obtain the necessary reference materials such as code examples and guides, and by using these resources, you can develop your plugin applications faster and with higher quality.

9. Alarm list

9.1 Description of alarm event

Agilebot supports the following classification of alarm event levels and corresponding behavior strategies, as shown in Fig. 9.1:

Behaviors and security strategies corresponding to alarm levels					
Event level	Robot motion	Servo bus power supply	User program	Scope	Safe stop strategy corresponding to alarms
INFO	Unaffected	Unaffected	Unaffected	...	None
PAUSE.L PAUSE.G	Stop after deceleration	Not break (release)	Pause	Local Overall	None
STOP.L STOP.G	Stop after deceleration	Break		Local Overall	Type: Class 1 stop
SERVO1	Instantaneous stop			Overall	Type: Class 0 stop
ABORT.L ABORT.G	Stop after deceleration	Break (braking)	Abort	Local Overall	Type: Class 1 stop
SERVO2	Instantaneo	Break (braki		Overall	Type: Class 0 stop

SYSTEM	us stop	ng)		Overa ll	Type: Class 0 stop It is a major issue related to the system. After this alarm occurs, all robot operations should be prohibited. After it is solved, it is necessary to shut down, restart and initialize the system.
Warning	Unaff ected	Unaff ected	Unaff ected	...	None

Fig. 9.1 Event Levels and Strategies

Description of current alarm event:

Events with a level above "Info" (exclusive) are defined as alarms.

"Current alarm" is a valid alarm not successfully reset yet, and the corresponding processing behavior of its "event level" may continuously generate restrictions or warnings on the system. Accordingly, the alarm, which has been reset, is no longer a current valid alarm but a past alarm. Such an alarm is not shown in the list of "Current Alarms" but in the "Event History".

The current alarm is displayed in the "Event Information" area of the UI status bar. It is used to remind the user whether there are still active alarms in the current system, so that the user can conveniently maintain and debug the robot.

Please note the following points:

- Only current events with the highest level are displayed in the event information bar.
- Events with "Info" or higher levels are displayed in the "Event Information" status bar.
- There is also an independent interface for displaying all current alarms, but the events at the "Info" level are not displayed in this interface (for Info is not actually an alarm).

9.1.1 Relevant interfaces and function descriptions

Fig. 9.2 Event Information Area in Status Bar

The alarm event status bar is shown in Fig. 9.2. The user can click event information area on this status bar to enter the current alarm interface as shown in Fig. 9.3.

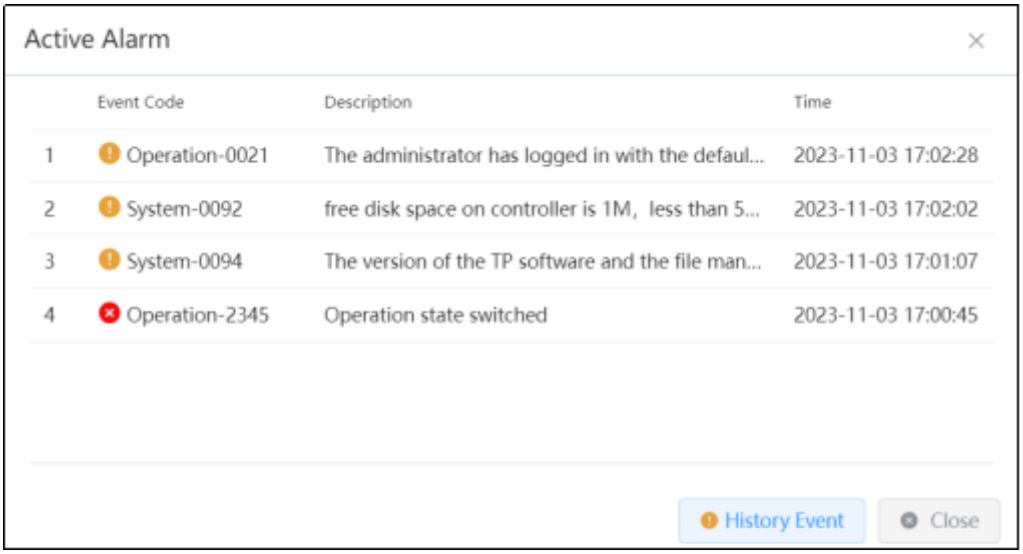


Fig. 9.3 Current Alarm Interface

The current alarm interface is to display all current alarms present in the system. They are sorted in chronological order from current to past. The user can click any alarm to enter its detailed description interface.

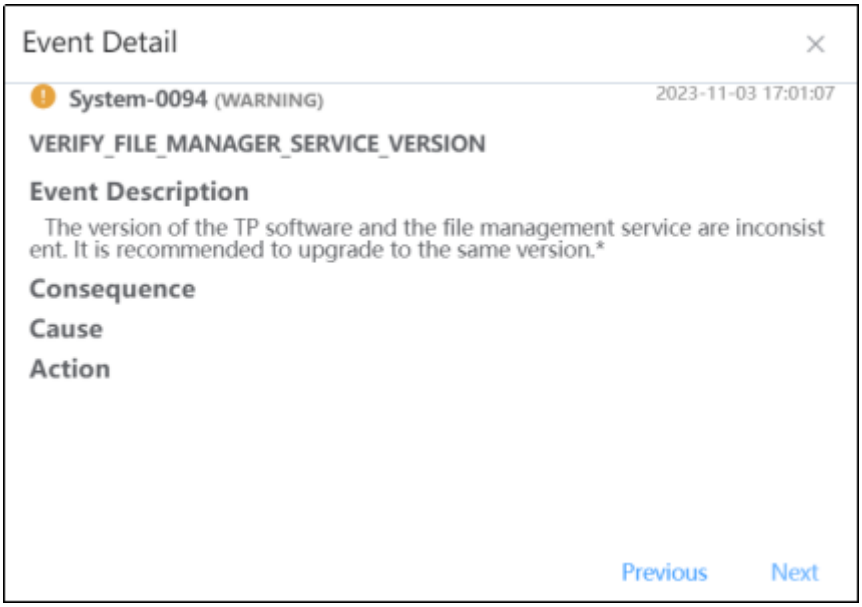


Figure 9.4 Event Detail Interface

Fig. 8.4 shows an event detail interface for current alarm - single alarm. This interface provides specific and detailed description of a certain alarm, including code, level, name, description, consequence, cause and action. The user can switch among different alarms by Previous/Next. The switched list is related to the alarms on current alarm interface.

A Reset signal may be sent to the controller for resetting. After the controller completes resetting (successful or failed), the "Current Alarm" interface is refreshed. The alarms (if any) after resetting are displayed in the current alarm interface in chronological order. Without alarm, an empty page is shown and no alarm is displayed on the status bar of the TP interface.

The user can also click the History Event button on the current alarm interface to enter the History Event interface.

9.2 *History event*

Overview

Historical events record user's operations on the robot system as well as alarms, prompts and status transitions inside the system, for example, a series of events, such as user login, alarm, pause, motor power-on, contactor release, robot entering manual speed limit mode, reset, calibration, etc.

Caution

Temporary storage size is 250M for an Event Log. Beyond this limit, a new log will

overwrite the oldest one from the current time.

9.2.1 Relevant interfaces and function descriptions

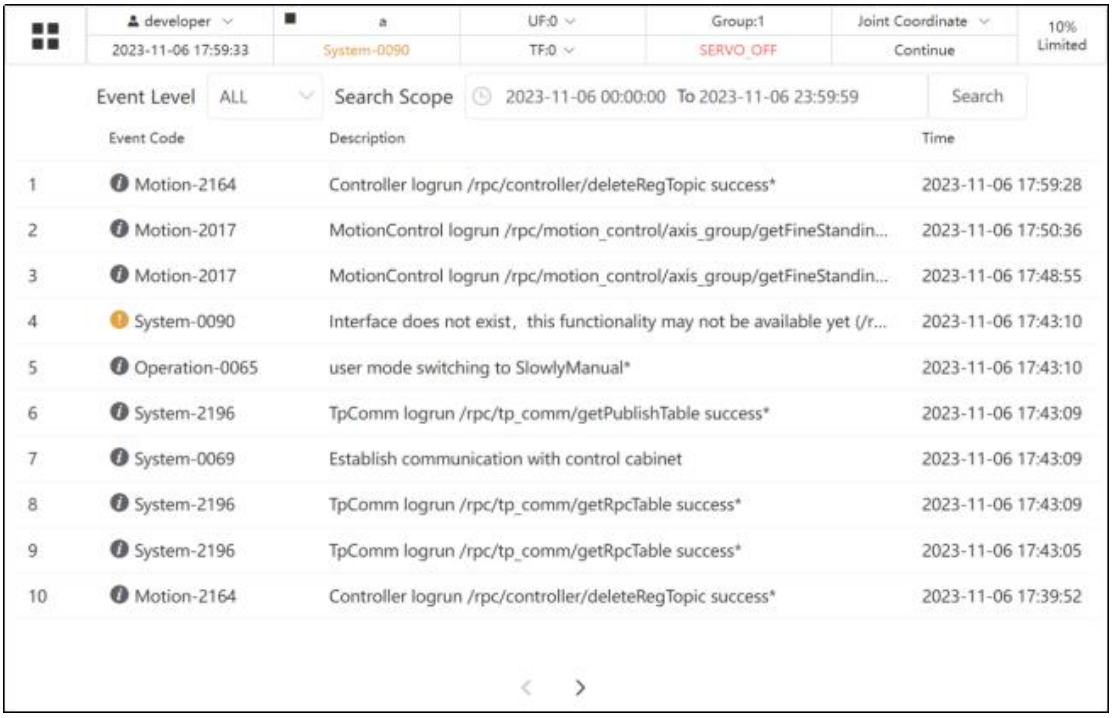


Fig. 9.5 Event Log/History Interface

Successively click "Menu Button" → "Shortcut Menu" → "History Event" to enter the history event log interface as shown in Fig. 8.5. This interface lists all events in chronological order from present to past.

The user can filter them through specific filters. There are two filters. One is based on alarm category or level selected by the user and the displayed contents are related to the category selected by the user; another is a time filter, of which the user can choose to display only event logs in a certain period of time.

If the user clicks an event entry in the event log, it will also automatically enter its detailed description interface.

10. List of SOCKET error codes

This chapter introduces the meanings of optional parameters in the SOCKET instruction (see Section 3.8.12)

Parameter	Meaning
0	Success
1	Operation not permitted
2	No such file or directory
3	No such process
4	Interrupted system call
5	Input/output error
6	No such device or address
7	Argument list too long
8	Exec format error
9	Bad file descriptor
10	No child processes
11	Try again
12	Out of memory
13	Permission denied
14	Bad address
15	Block device required
16	Device or resource busy
17	File exists
18	Cross-device link

19	No such device
20	Not a directory
21	Is a directory
22	Invalid argument
23	File table overflow
24	Too many open files
25	Not a typewriter
26	Text file busy
27	File too large
28	No space left on device
29	Illegal seek
30	Read-only file system
31	Too many links
32	Broken pipe
33	Math argument out of domain of func
34	Math result not representable
35	Resource deadlock would occur
36	File name too long
37	No record locks available
38	Invalid system call number

39	Directory not empty
40	Too many symbolic links encountered
41	Operation would block
42	No message of desired type
43	Identifier removed
44	Channel number out of range
45	Level 2 not synchronized
46	Level 3 halted
47	Level 3 reset
48	Link number out of range
49	Protocol driver not attached
50	No CSI structure available
51	Level 2 halted
52	Invalid exchange
53	Invalid request descriptor
54	Exchange full
55	No anode
56	Invalid request code
57	Invalid slot
58	Resource deadlock would occur

59	Bad font file format
60	Device not a stream
61	No data available
62	Timer expired
63	Out of streams resources
64	Machine is not on the network
65	Package not installed
66	Object is remote
67	Link has been severed
68	Advertise error
69	Srmount error
70	Communication error on send
71	Protocol error
72	Multihop attempted
73	RFS specific error
74	Not a data message
75	Value too large for defined data type
76	Name not unique on network
77	File descriptor in bad state
78	Remote address changed

79	Can not access a needed shared library
80	Accessing a corrupted shared library
81	.lib section in a.out corrupted
82	Attempting to link in too many shared libraries
83	Cannot exec a shared library directly
84	Illegal byte sequence
85	Interrupted system call should be restarted
86	Streams pipe error
87	Too many users
88	Socket operation on non-socket
89	Destination address required
90	Message too long
91	Protocol wrong type for socket
92	Protocol not available
93	Protocol not supported
94	Socket type not supported
95	Operation not supported on transport endpoint
96	Protocol family not supported
97	Address family not supported by protocol
98	Address already in use

99	Cannot assign requested address
100	Network is down
101	Network is unreachable
102	Network dropped connection because of reset
103	Software caused connection abort
104	Connection reset by peer
105	No buffer space available
106	Transport endpoint is already connected
107	Transport endpoint is not connected
108	Cannot send after transport endpoint shutdown
109	Too many references: cannot splice
110	Connection timed out
111	Connection refused
112	Host is down
113	No route to host
114	Operation already in progress
115	Operation now in progress
116	Stale file handle
117	Structure needs cleaning
118	Not a XENIX named type file

119	No XENIX semaphores available
120	Is a named type file
121	Remote I/O error
122	Quota exceeded
123	No medium found
124	Wrong medium type
125	Operation Canceled
126	Required key not available
127	Key has expired
128	Key has been revoked
129	Key was rejected by service
130	Owner died
131	State not recoverable
132	Operation not possible due to RF-kill
133	Memory page has hardware error
999	Recv timeout

Contact us

Agilebot Robotics Co., Ltd. (Shanghai Headquarters):

Address:7F, Building T1, Hongqiao Wanchuang Center, Lane 500, Xinlong Road, Minhang District, Shanghai, China

Agilebot Operation and Technical Service Center:

Building 1, No. 338 Jiuye Road, Qingpu District, Shanghai

Agilebot Dongguan Branch:

Room 801,Building 9, No. 14 Industrial South Road, Songshan Lake Park, Dongguan City,Guangdong Province, China

Service hotline: +86-21-5986 0805

Website: www.sh-agilebot.com