User Manual
Operation Manual for Collaborative Robot System

User Manual
Operation Manual for Collaborative Robot System

V1.2

Suitable for Cobot Series

Ver: V7.5

The information in this Manual must not be considered as a commitment of Agilebot and may be changed without prior notice. Agilebot assumes no responsibility for errors (if any) in this Manual.

Except as expressly specified, nothing in this Manual shall be construed as any warranty or guarantee made by Agilebot for personal loss, property damage or specific applicability.

Agilebot assumes no responsibility for any accidents or indirect injuries caused by using this Manual or the product described therein.

This Manual and any parts thereof must not be reproduced or duplicated without written permission from Agilebot.

Additional copies of this Manual may be obtained from Agilebot.

The original language of this Publication is Chinese.

International standard units are adopted in this publication. GB means Chinese national standard.

Shanghai, China

Revision

Ver.	Date	Status
V1.0	Oct 25, 2023	Deactivated
V1.1	Feb 27,2024	Deactivated
V1.2	Feb 08,2025	Release

Table of Contents

SAFETY INSTRUCTIONS 11

1 PRODUCT OVERVIEW 19

1.1 ROBOT SYSTEM 19
1.2 ROBOT MODELS 20
1.3 CONTROLLER 21
1.3.1 Wired Handle 21
1.3.2 Controller Panel 21
1.3.3 Communication 22
1.3.4 I/O Signal 22
1.3.5 Robot Actions 22
1.3.6 E-stop Devices 23
1.3.7 Cobots Software Installation 23
1.3.8 Robot light strip 28
1.3.9 The buttons at the end cup 29
1.4 POWER-ON/OFF 32
1.4.1 Inspection before power-on 32
1.4.2 Setting Up C5A for the First Time 32
1.4.3 Power-on 37
1.4.4 Shutdown 38
1.5 INTRODUCTION TO OPERATION INTERFACE 40
1.5.1 Menu 40
1.5.2 Status bar 42
1.5.3 System information 43
1.6 MODE SWITCH 44
1.7 ROBOT TEACHING 46
1.8 VELOCITY CONTROL 52
1.9 SAFETY FUNCTIONS 53
1.9.1 Safety Limitations 53
1.9.2 Limit the posture of the tool 57
1.9.3 Tool Volume 59
1.9.4 Safety Plane 61
1.9.5 Safety I/O Signals SI/SO 62

2 SETTING OF ROBOT SYSTEM 65

2.1 I/O SIGNAL 65
2.1.1 Digital I/O 67
2.1.2 Group I/O 70
2.1.3 Special I/O 70
2.1.4 Tool I/O 72
2.1.5 Manual control of I/O 74
2.2 SETTING OF COORDINATE SYSTEM 77
2.2.1 Setting of tool coordinate system 78
2.2.2 Setting of user coordinate system 85
2.2.3 Setting of Remote TCP 89
2.2.4 Coordinate transformation function 91
2.3 SOFT LIMIT 95
2.3.1 Soft limit interface 95
2.4 PAYLOAD SETTING 97
2.4.1 Automatically measure the load 99
2.5 User level 109
2.6 GENERAL SETTING 114
2.6.1 Time/Language setting 114
2.6.2 Network Configuration 114
2.7 SYSTEM PARAMETERS 115
2.7.1 Type 115
2.7.2 Setting of general system variables 115

3 COMPOSITION OF PROGRAM 117

3.1 Program property 120
3.1.1 Program name 120
3.1.2 Comment 120
3.1.3 Program type 121
3.1.4 Program protection 121
3.2 Line number, end marker, arrow and parameter 122
3.3 Action instruction 123
3.3.1 Action type 123
3.3.2 Position data 125
3.3.3 Movement speed 127

3.3.4 Positioning type 128
3.3.5 Additional commands for actions 129
3.4 Register instruction 134
3.4.1 Number register instruction 134
3.4.2 Position register instruction 135
3.4.3 Position register element instruction 136
3.4.4 String register and string instruction 137
3.4.5 Motion register MR[i] instruction 138
3.4.6 Modbus special register instruction 138
3.5 I/O instruction 140
3.5.1 Digital I/O instruction 140
3.5.2 Tool I/O instruction 141
3.6 Logical instruction 142
3.6.1 IF instruction 142
3.6.2 SWITCH instruction 143
3.6.3 WHILE instruction 143
3.6.4 GOTO instruction 145
3.6.5 SKIP CONDITION instruction 145
3.7 Structure instructions 145
3.7.1 CALL - Call program instruction 146
3.7.2 WAIT - Waiting for conditions to meet 146
3.7.3 WAIT TIME - waiting time instruction 147
3.7.4 PAUSE - Pause instruction 148
3.7.5 ABORT - Abort instruction 148
3.7.6 RUN - Multithread instruction 149
3.7.7 Load - Dynamic program loading 150
3.7.8 Exec - Execute dynamic program loading 151
3.7.9 Unload - Dynamic program unload 151
3.8 Other instructions 153
3.8.1 TF_NO - Tool coordinate system instruction 153
3.8.2 UF_NO - user coordinate system instruction 153
3.8.3 J_POS - current joint coordinate instruction 153
3.8.4 L_POS - current Cartesian coordinate instruction 153
3.8.5 Payload_NO - payload setting instruction 154
3.8.6 Timer - timer instruction 154

3.8.7 Comment - Commend instruction 154
3.8.8 OVC - Overall velocity instruction 154
3.8.9 OAC - Overall acceleration instruction 154
3.8.10 LABEL - Label instruction 155
3.8.11 Socket - Socket instruction 155
3.8.12 Compound operation instruction 156
3.9 String instructions 160
3.9.1 StrLen - String length instruction 160
3.9.2 FindStr - String search instruction 160
3.9.3 SubStr - String subtraction instruction 160
3.10 Methods (functions) 161
3.10.1 SIN - Sine function 161
3.10.2 COS - Cosine function 161
3.10.3 TAN - Tangent function 161
3.10.4 ARCSIN - Anti-sine function 161
3.10.5 ARCCOS - Arccosine function 162
3.10.6 ARCTAN - Arctangent function 162
3.10.7 ABS - Absolute value function 162
3.11 VISION INSTRUCTION 162
3.11.1 VISION FIND instruction 162
3.11.2 VISION GET_QUANTITY instruction 162
3.11.3 MODELID GET MODEL ID instruction 163
3.11.4 GET_OFFSET instruction 163
3.11.5 VOFFSET instruction 163
3.11.6 Assignment of detected location information 163
3.11.7 Examples of Vision Program 164
4 PROGRAM CREATION AND EXECUTION 165
4.1 CREATE PROGRAM 165
4.1.1 Description of program operation interface 166
4.1.2 Copy program 169
4.1.3 Delete program 169
4.1.4 Choose program 169
4.1.5 Create action instruction 171
4.1.6 Modify motion instruction 174
4.1.7 Create additional instruction 174

4.1.8 Insert control instruction 175
4.2 PROGRAM SETTING 193
4.2.1 Setting of program launch mode 194
4.3 PROGRAM START POINT 199
4.4 EXECUTE PROGRAMS 200
4.4.1 Trigger the program in manual mode 200
4.4.2 Trigger the program in auto mode 200
4.4.3 Program pause and abort 201
4.4.4 Resume after pause 202
4.5 PROGRAM MONITORING 202
5 STATUS DISPLAY 204
5.1 REGISTER MANAGEMENT 204
5.1.1 Number register instruction 204
5.1.2 Motion register 204
5.1.3 Pose register 205
5.1.4 String register 207
5.1.5 Socket register 208
5.1.6 Modbus special registers 209
5.2 CURRENT POSITION 212
6 ROBOT COMMUNICATION SETTING 214
6.1 SLAVE CONFIGURATION 214
6.2 Master setting steps 216
7 SYSTEM BACKUP AND LOADING 220
7.1 BACKUP AND LOAD OBJECTS 220
7.2 DESCRIPTION OF OTHER FUNCTIONS 221
7.3 USER’S OPERATION MODE 221
8 PRACTICAL FUNCTIONS 224
8.1 PROGRAM OFFSET 224
8.1.1 General offset 225
8.1.2 Mirror offset 226
8.1.3 Circular array offset 227
8.2 BASIC SPACE ANTI-INTERFERENCE 227
8.3 REFERENCE POSE 231

8.4 TF ALIGNMENT FUNCTION....232
9 EVENT INFORMATION TABLE.... 236
9.1 EVENT DESCRIPTION....236
9.1.1 Relevant interfaces and function descriptions....237
9.2 HISTORY EVENT....238
9.2.1 Relevant interfaces and function descriptions....239
10 LIST OF SOCKET ERROR CODES.... 240

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 robot body, robot controller, wired handle, 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 robot without authorization from Agilebot Robotics Co., Ltd. Agilebot Robotics Co., Ltd. shall assume no responsibility for any damage to the 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 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.

For safety issues uncovered in this Manual, please refer to the Safety Manual.

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 permissions for operators

Operator

Operator's authorities include:

1) Turn on/off the robot.
2) Use the operating terminal 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 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.

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 controller IP
11) Setting of system time

Administrator (Admin)

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
 Danger	It indicates dangerous situations possibly resulting in serious injury or death to the user during incorrect operation.
 Warning	It indicates dangerous situations possibly resulting in mild or moderate personal injury or property damage during incorrect operation.
 Caution	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. Set peripheral devices outside the robot's range of motion as much as possible.
4. Arrange locks as needed to prevent personnel other than operators from turning on the power supply of the robot.
5. When conducting individual debugging of peripheral devices, it is important to disconnect the power supply of the robot in advance.
6. 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.
7. 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.
8. 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.
9. 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.
    1) Flammable environments
    2) Explosive environments
    3) High-radiation environment
    4) Water or high humidity environment
    5) 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
	Warning - electric shock
	Warning - hands pinching
	Beware of burns due to high temperature.
	Grounding

Stop Categories of the robot

The methods for stopping Collaborative robots include (According to IEC 60204-1):

Cat. 0 stop

- Stopping by immediate removal of power to the robot (i.e. an uncontrolled stop).

Cat. 1 stop

- A controlled stop with power available to the robot to achieve the stop and then removal of power when the stop is achieved (safety system will cut off the servo bus power 1 second after the robot stops or not.).

Cat. 2 Stop

- A controlled stop with servo bus power remaining available to the robot.

 Warning

Rrisk assessment of the entire system is required.

When establishing safety range, it is necessary to consider the stop distance and stop time. Do not enter the safety range of the robot or touch the robot when the system is in operation.

Safety functions

Overview to safety functions

The Collaborative robot is provided with the following safety features:

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

E-stop

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

- E-stop button with a wired handle

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 (Cat. 1).

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.

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 in a very urgent situation.

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 robots.

2. 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 Collaborative robot.

1.1 ROBOT SYSTEM

The Collaborative robot system is composed of:

- Robot body *
- Controller
- Wired handle
- Operating terminal **
- Robot connection cable
- Software system

The Collaborative robot system is shown in the figure:

* The robot body refers to the mechanical structure of the robot, which is a composed of such parts as motor, gearbox, encoder and drive mechanism.

** Operating terminal is a device used by the users for programming, setting and other operations.

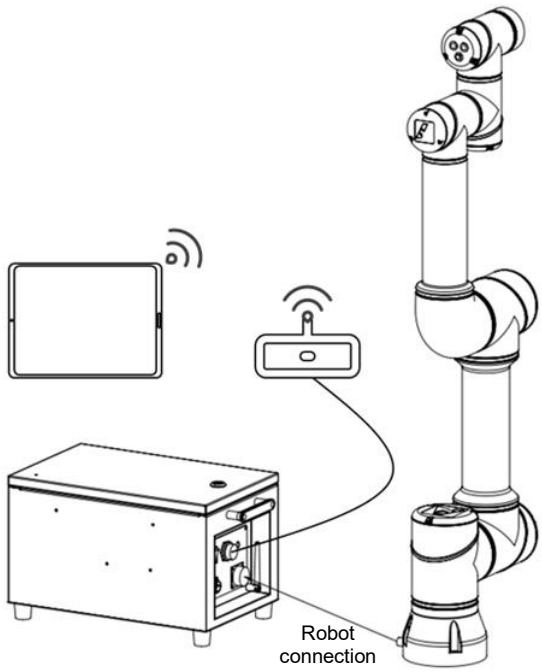


Fig. 1.1 Composition of Collaborative Robot System

1.2 ROBOT MODELS

Cobot Series

Product Categories		Product Series		Payload		Version	
GBT	Agilebot Industrial Robots	C	Cobots	5	5KG	A	1st Generation
				7	7KG		
				12	12KG		

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 Wired Handle

For a detailed introduction of wired handle, refer to the instructions for IRC-D6A Controller.

Fig. 1.3.1 Wired Handle

Color	The light on the Wired Handle		
	steady light	Flash (once per second)	Flash (twice per second)
Blue		1.The Servo on process 2.The Servo off process	
White	1.Servo off, no fault, and the Cobot software is not connected. 2.The robot arm has no available power supply: Failure\The system power is off.	Servo off, no fault, and the Cobot software is connected.	
Green	The program is running	The Servo on is completed.	
Yellow	Drag mode	1. The program is paused. 2. Type 2 stop.	
Red	1. Emergency stop 2. Alarm event level 5 - 10	The communication link of the main body is disconnected.	1. Alarm event level 11 2. Boot - up timeout

1.3.2 Controller Panel

There are status lamps, controller switch, communication interfaces, etc. on the controller panel. See the instructions for D6A Controller for detailed instructions.

1.3.3 Communication

The controller provides communication ports with peripheral devices. The IRC-D6A series controller provides 2 Ethernet ports.

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
- Special I/O
- Tool I/O

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 and action instruction based on the program.

- The action of the robot based on log feed or step feed is executed through the operating terminal. The action during jog feed is determined by the manual feed coordinate system and the speed multiplier.
- 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) and "MOVEC" (arc) (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 "MOVEL 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

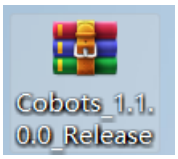
The robot is provided with the following e-stop devices:

- E-stop button on the wired handle.
- The controller has an e-stop signal and a safety door signal, both of which are Cat. 1 stop.

1.3.7 Cobots Software Installation

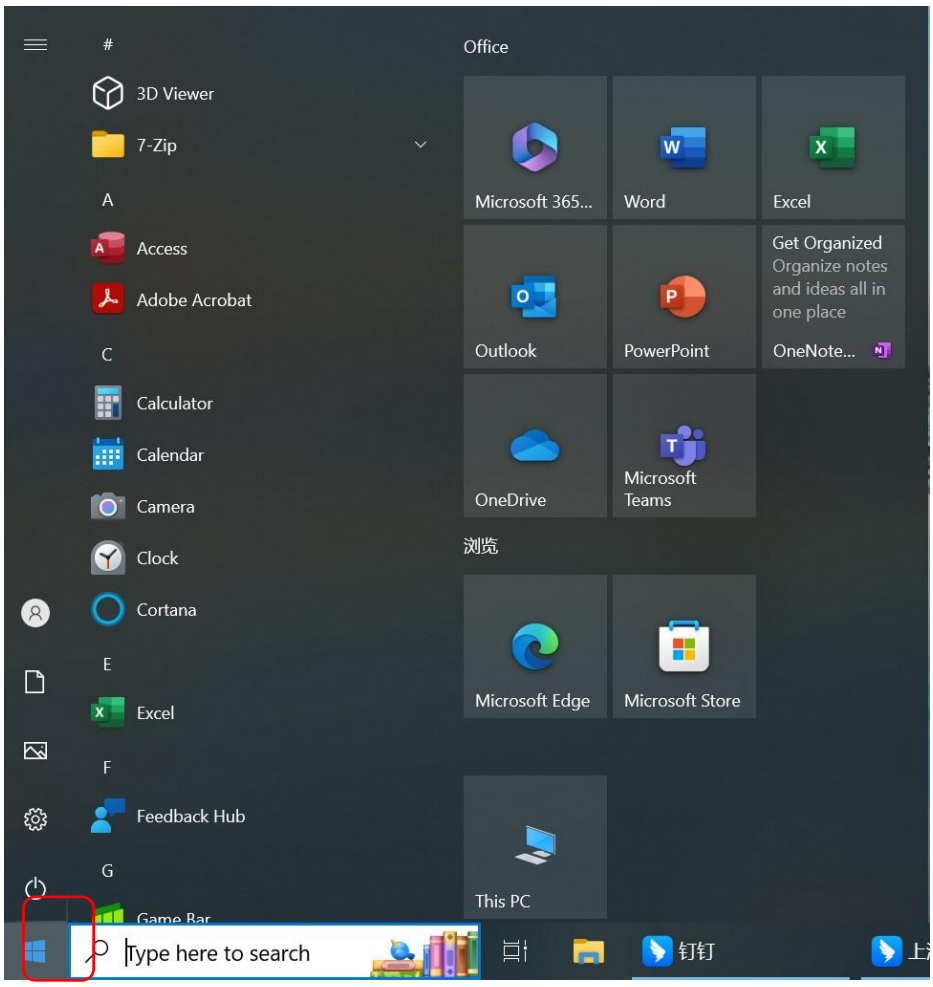
Preparation for software installation

1、 Prepare Cobots software installation package；

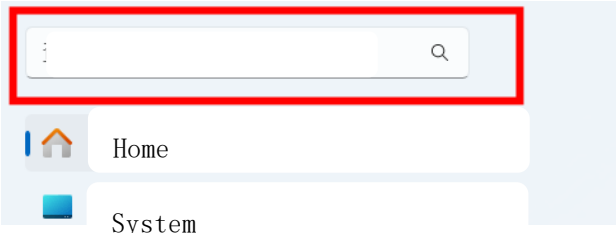


2、 Click  in the lower left corner of your computer desktop；

3、 Click Settings in the Start Menu；



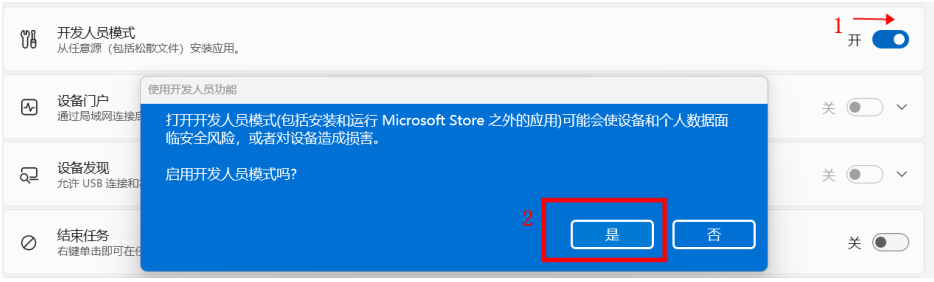
4、Enter Developer in the Find Settings field ，When finished select the corresponding option according to your computer system；



Win10 system：Developer Mode；

Win11 system：View Developer Drive Anti-Virus Behavior；

5、Open the Developer Mode option, a pop-up window will appear and you will be prompted to select “YES”；

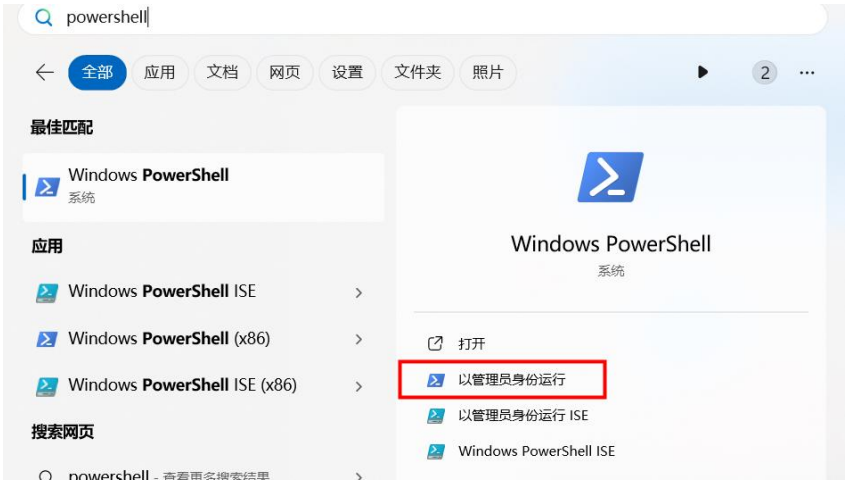


6、Click the Window Close button to exit the current window；

Software Installation Steps

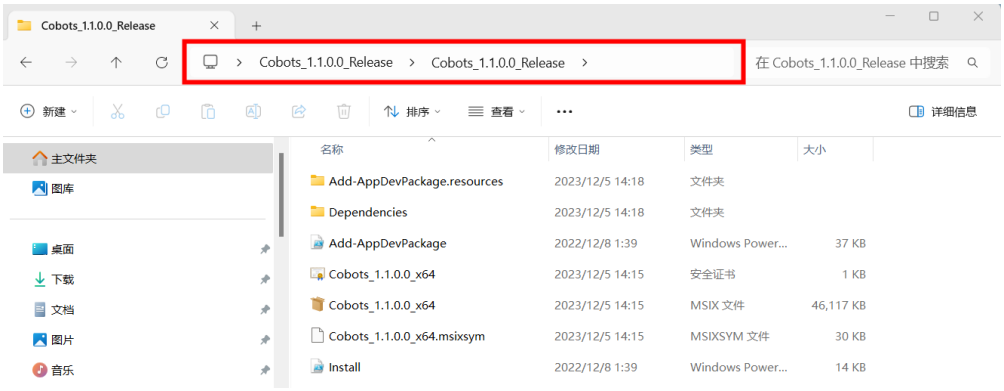
1、 Click  in the lower left corner of your computer desktop；

2、 Open the menu and type “PowerShell” ，Select Run as administrator when done;



3、 Go to the following screen：

4、 Open the Cobots software installation package and copy the path where the package is stored.;

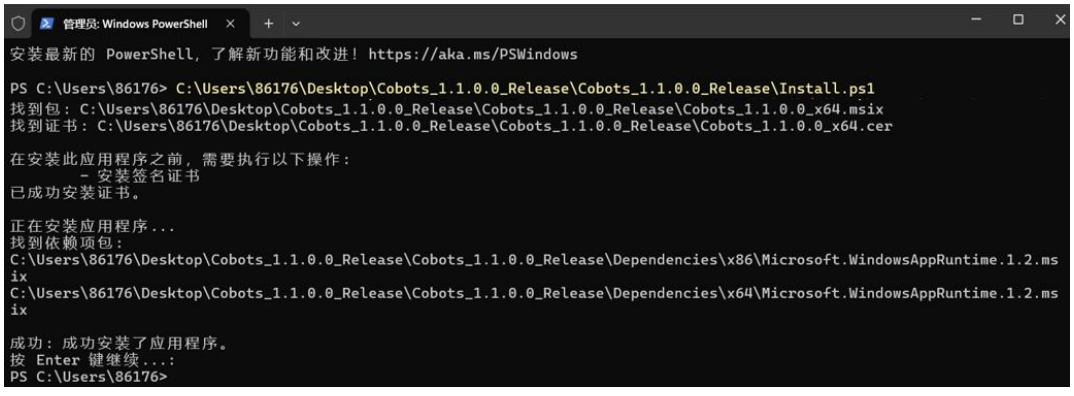


5、 In the command window, paste the path where the Cobots software package is stored and type “Install.psl ”and press Enter when you are done, if there are any exceptions, please refer to Chapter 3, Handling Software Installation Exceptions.;

```
PS C:\Users\21196> C:\Users\21196\Desktop\Cobots_1.1.0.0_Release\Cobots_1.1.0.0_Relea
se\Intall.psl
```

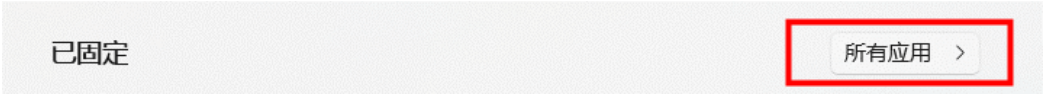
Note: C:\Users\21196\Desktop\Cobots_1.1.0.0_Release\Cobots_1.1.0.0_Release The above path is the path of the test installation computer, the installation of the actual path shall prevail.

6、 The command window shows that the application has been successfully installed, indicating that the installation is complete. Press Enter to exit the installation and click the Window Close button to close the command window.;

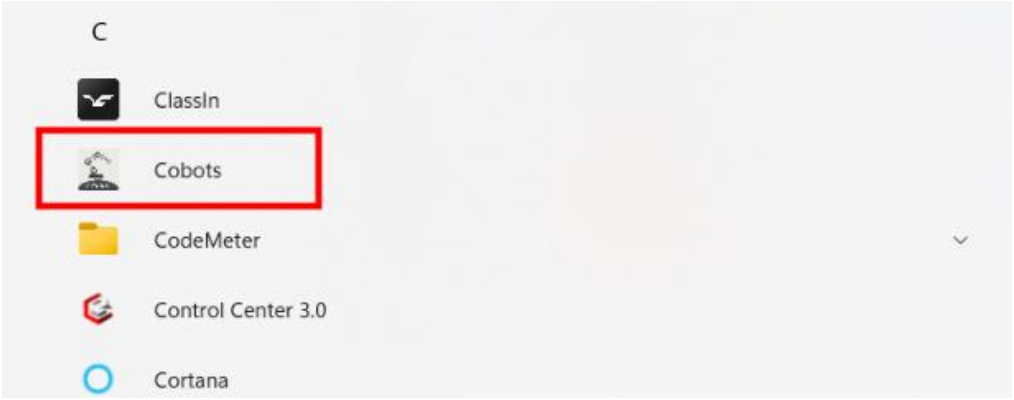


7、 Click  in the lower left corner of your computer desktop ;

8、 Click All Apps in the Start menu;;

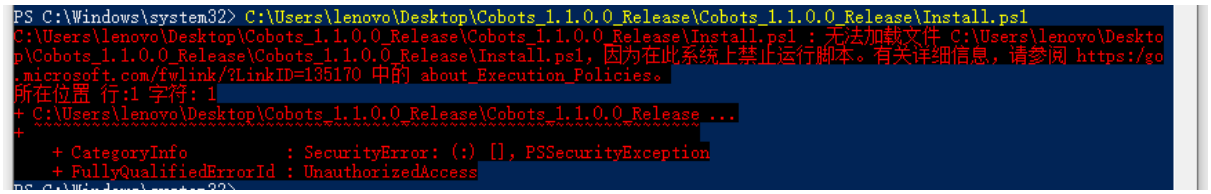


9、 Find the Cobots software in the Applications menu C-Series option to open and use it；



Q&A

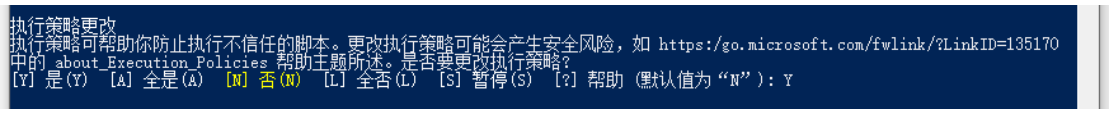
Q1：Unable to load file because running scripts is prohibited on this system.Follow the steps below when this error occurs;



Answer：

1) Type Set-ExecutionPolicy RemoteSigned in the command window and press Enter when you are done；

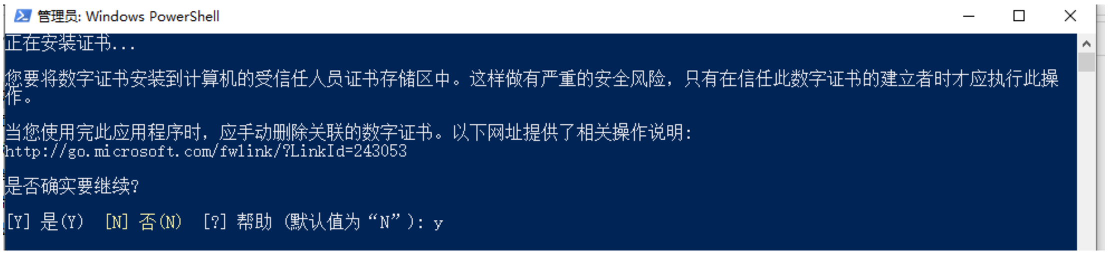
2) When the command window prompts Do you want to change the enforcement policy? type “Y” to complete and press Enter；



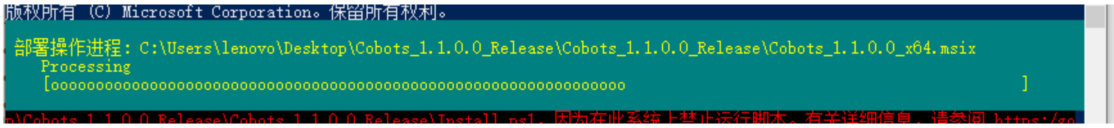
3) Open the Cobots software installation package and copy the path where the package is stored；

4) In the command window, paste the path where the Cobots software installation package is stored and type \Install.psl and press Enter when finished；

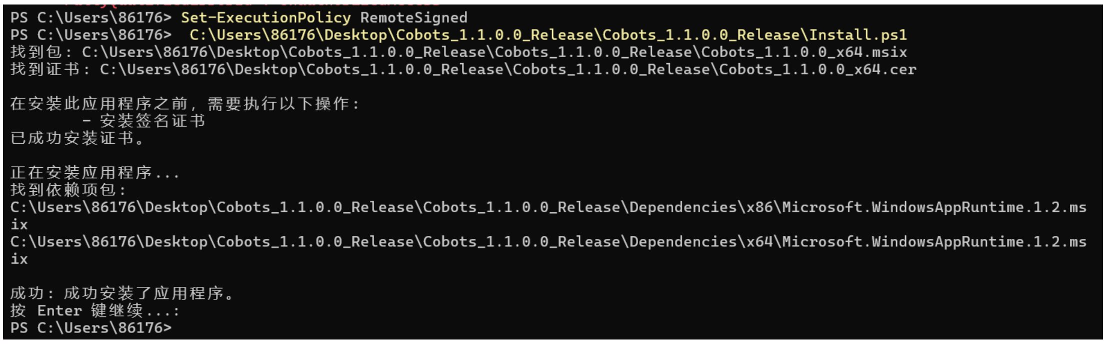
5) Do you really want to continue with the installation process when it pops up? Type Y to complete and press Enter;



6) A green box appears during the installation process to indicate that the installation is in progress, please wait；



7) The command window shows that the application has been successfully installed, indicating that the installation is complete. Press Enter to exit the installation and click the Window Close button to close the command window.;



8) Just follow the steps in Section 2 to open the software；

1.3.8 Robot light strip

The robot is equipped with indicator lights at its end effector.

The light ring on the 6th axis at the end effector is shown in the following figure, and the meaning of its color indications is shown in the following table.

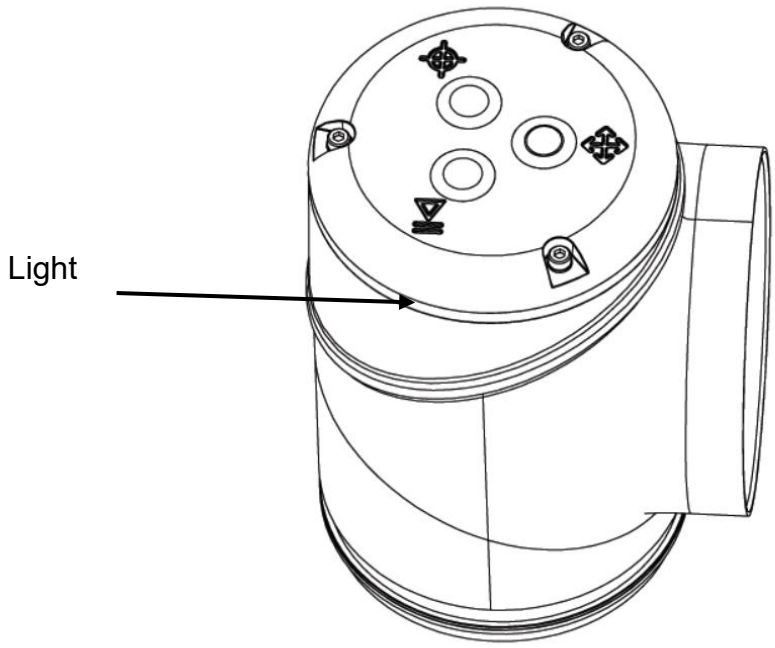


Fig.1.3 The light ring on the 6th axis

Color	The light ring on the 6th axis		
	steady light	**Flash (once per second)**	**Flash (twice per second)**
Blue		1.The Servo on process 2.The Servo off process	
White	Servo off, no fault, and the Cobot software is not connected.	Servo off, no fault, and the Cobot software is connected.	
Green	The program is running	The Servo on is completed.	
Yellow	Drag mode	1. The program is paused. 2. Type 2 stop.	
Red	1. Emergency stop 2. Alarm event level 5 - 10	The communication link of the main body is disconnected.	1. Alarm event level 11 2. Boot - up timeout
Off	The robot arm has no available power supply: 1.Failure 2.The system power is off.		

> Caution
>
> When the communication link of the robot body is disconnected and the arm communication is interrupted, the user can try to restart the control cabinet.

1.3.9 The buttons at the end cup

The robot is equipped with three buttons at its end effector. The buttons are the record position button , the drag button , and the pause/resume button respectively, as shown in the following figure.

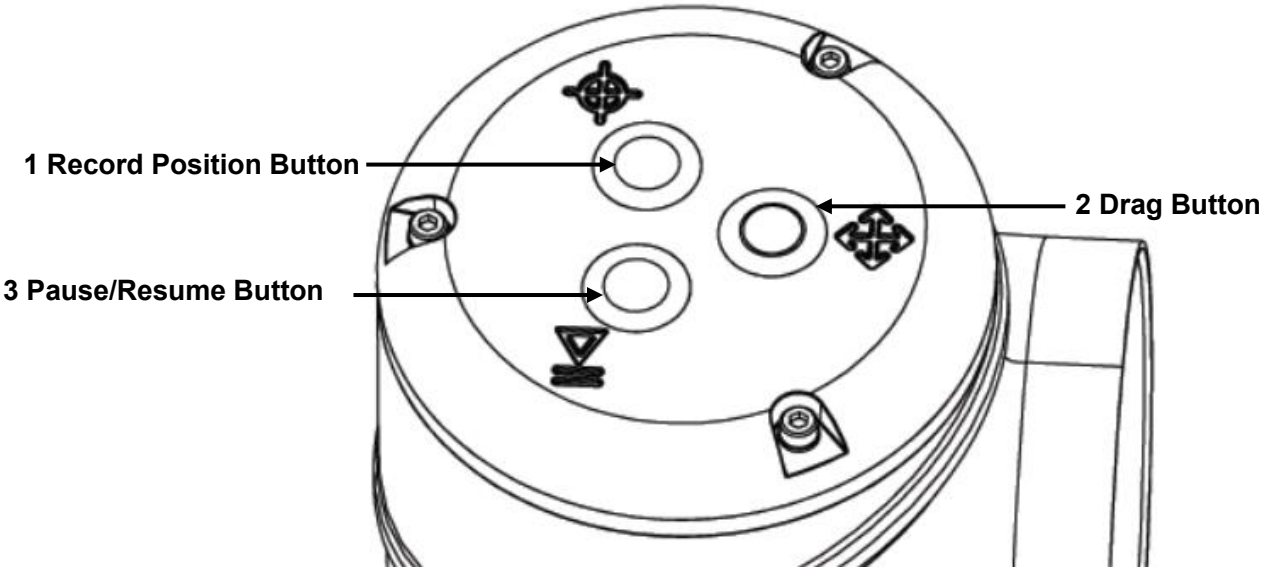


Fig.1.4 Button Position Diagram

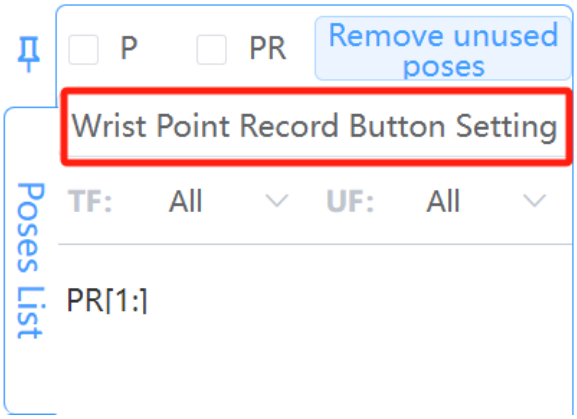
Record Position Button

The record position button needs to be used in conjunction with the robot operation software COBOT. When the record position button is pressed, the operation software will record the corresponding position.

Create in the Program Editing Page

Click the wrist position record button to set:

- The name and serial number format of the position can be modified.
- You can choose to create P or PR.



Create in the Pose Register Page

Click the wrist position record button to set:

- The name and serial number format of the position can be modified.
- You can choose to create PR

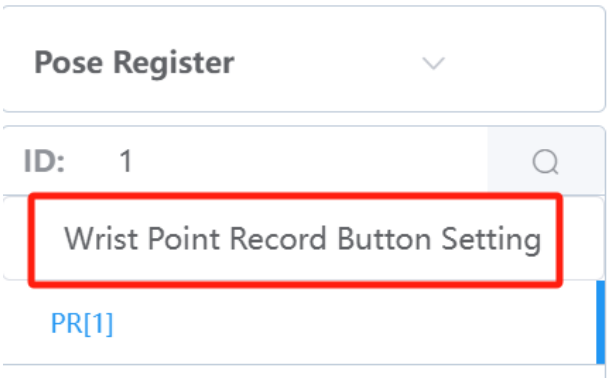


Fig.1.5 Wrist Point Record Button Setting

Drag Button

When the drag button is pressed, the robot enters the drag teaching mode. In this mode, users can directly drag the robot to the desired point. For teaching operations, please refer to 1.7 Robot Teaching.

Pause/Resume Button

In the manual mode, when the robot is running a program, pressing the pause/resume button can pause the robot's movement, and pressing it again can resume the movement.

> ⚠ Warning
>
> For the use of the drag button, users are required to fully assess the possible risks. It is necessary to ensure that parameters such as the installation posture of the robot, the end effector load and the Tool Center Point (TCP) are correctly set. Otherwise, it may cause personal injury or equipment damage.
> For the use of the pause/resume button, users are required to fully assess the possible risks. The sudden start and stop of the robot may cause personal injury or equipment damage.

1.4 POWER-ON/OFF

To power on/off, please refer to the instruction manual of the IRC-D6A control cabinet.

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:

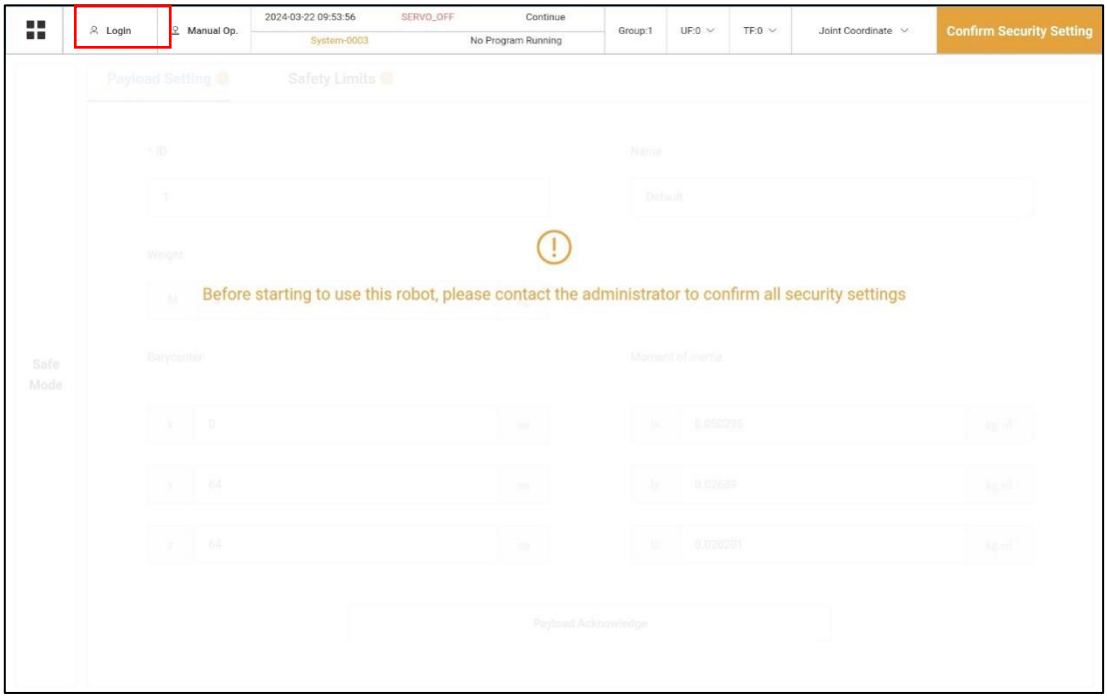
- Mechanical inspection: Check whether the base is completely fixed and whether the end-effector is firmly mounted.
- Wiring inspection: Check if various cables are wired correctly and firmly.
- Controller inspection: The controller (including power-off switch) should be mounted at a height convenient for operation and maintenance.

1.4.2 Setting Up C5A for the First Time

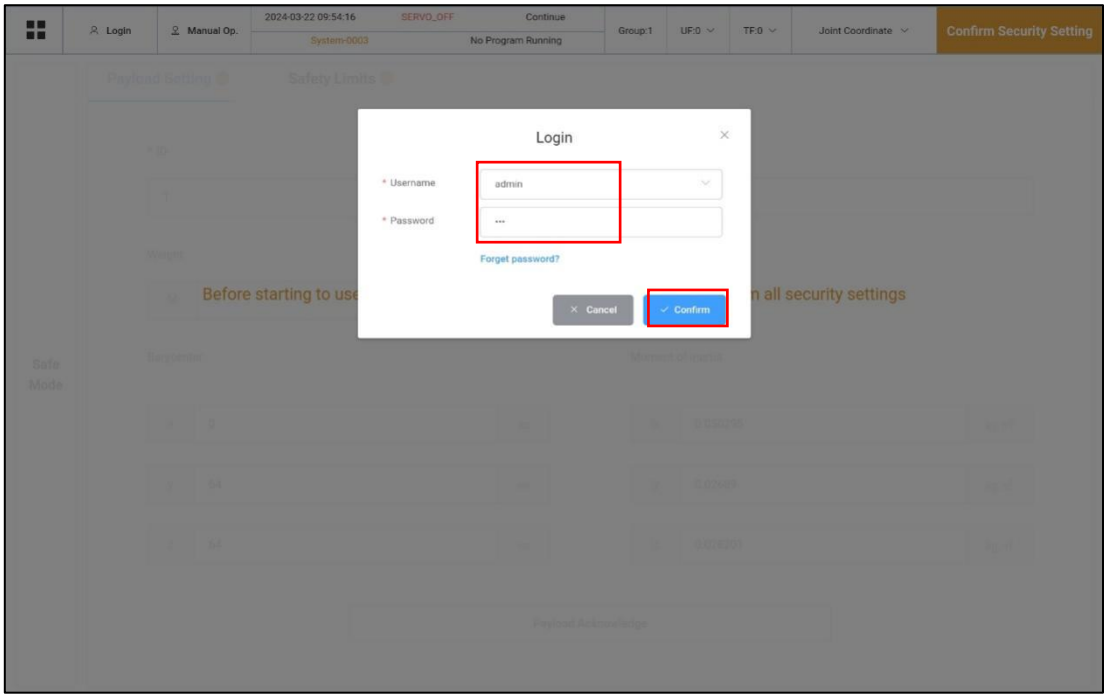
Collaborative robots have stricter safety limitations than industrial robots. Therefore, after the first power-on or the first power-on following a system update, the user needs to confirm the safety configuration before the robot can be safely enabled.

Payload Setting

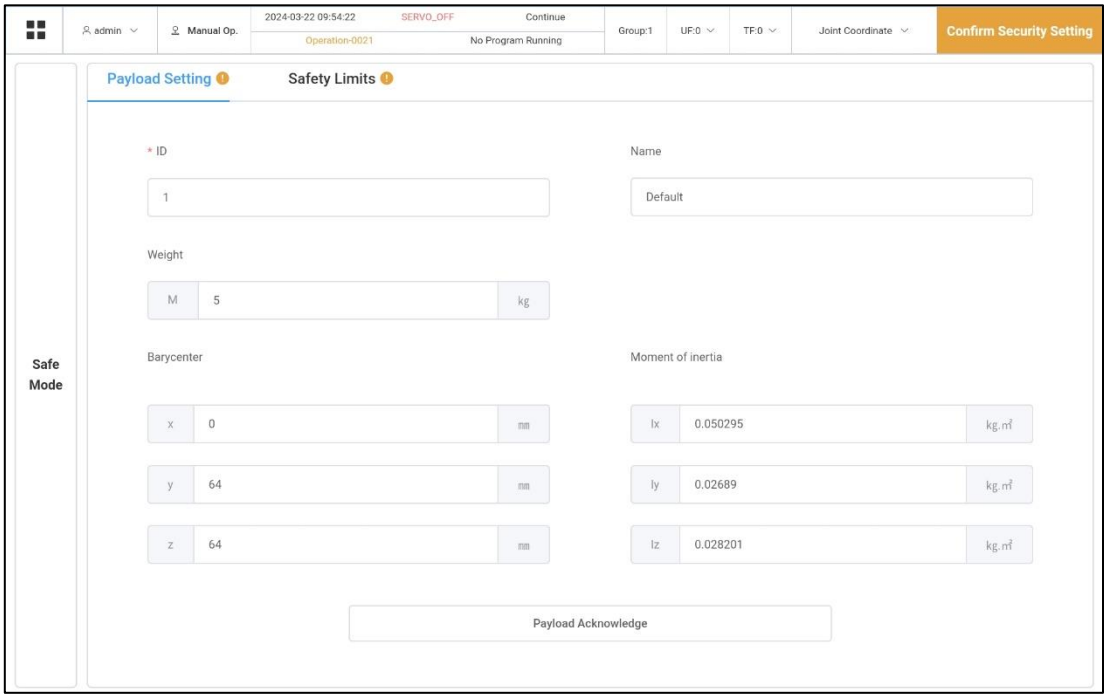
1.To start the robot system, connect to the robot using the Cobot software. The software will display the first boot security settings. Click the login button to access administrator rights.



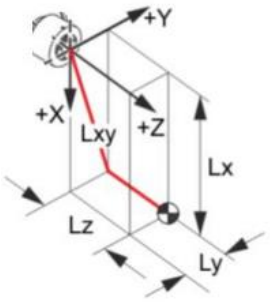
2.Select 'admin' as the username for administrator rights, enter '123' as the password, and click 'confirm'.



3. After confirmation, the system automatically enters the load setting interface.



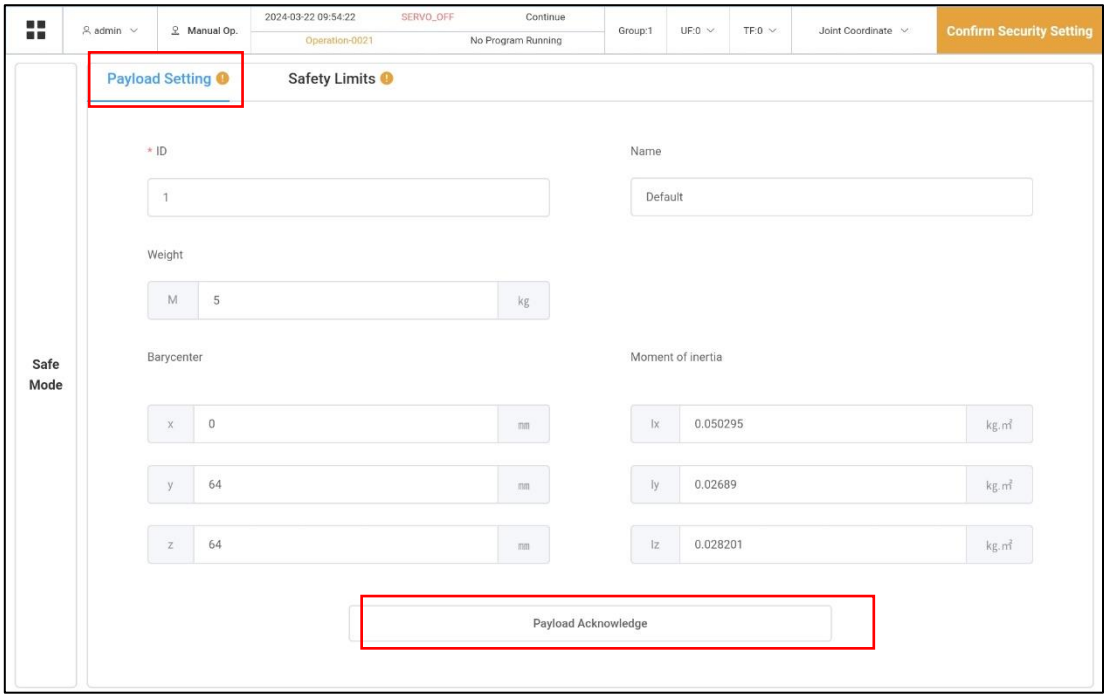
4.The mass of the load is denoted by M (kg), while X (mm), Y (mm), and Z (mm) represent the position of the centre of gravity of the load with respect to the centre of the flange. The moment of inertia of the load with respect to the direction of the X, Y, and Z coordinate systems are represented by Ix, Iy, and Iz, respectively. When the robotic load is treated as a mass point, the moment of inertia, Ix, Iy, and Iz, are written as 0.

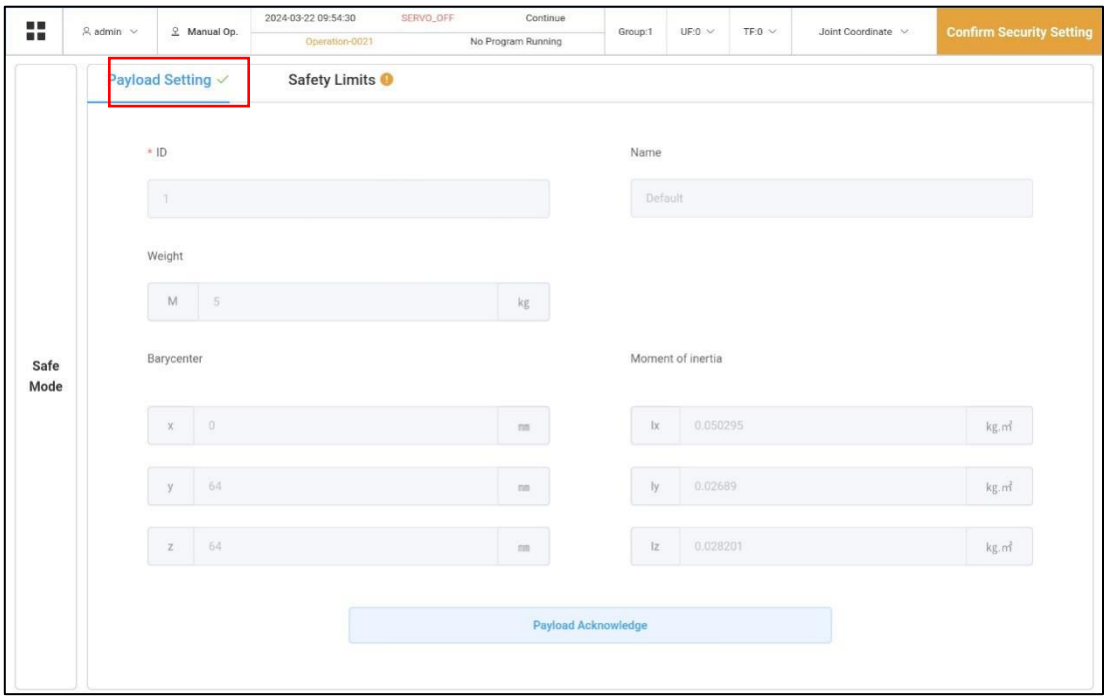


Description of payload parameters

PAYLOAD PARAMETERS		DESCRIPTION
ID		Payload Number
Name		Payload Name
Weight		Payload Weight
Barycenter	x	Offset in the X-direction with respect to the TCP point of the robot flange
	y	Offset in the Y-direction with respect to the TCP point of the robot flange
	z	Offset in the Z-direction with respect to the TCP point of the robot flange
Moment of inertia	Ix	A measure of the inertia of the load as it rotates around the X-axis.
	Iy	A measure of the inertia of the load as it rotates around the Y-axis.
	Iz	A measure of the inertia of the load as it rotates around the Z-axis.

5.To correctly load the parameters, click on Payload Acknowledge. If the payload is not confirmed, an exclamation mark " ! " will appear in the upper left corner. If the payload is confirmed, a tick "√" will appear in the same location. Once the load is confirmed, the payload setting is complete.

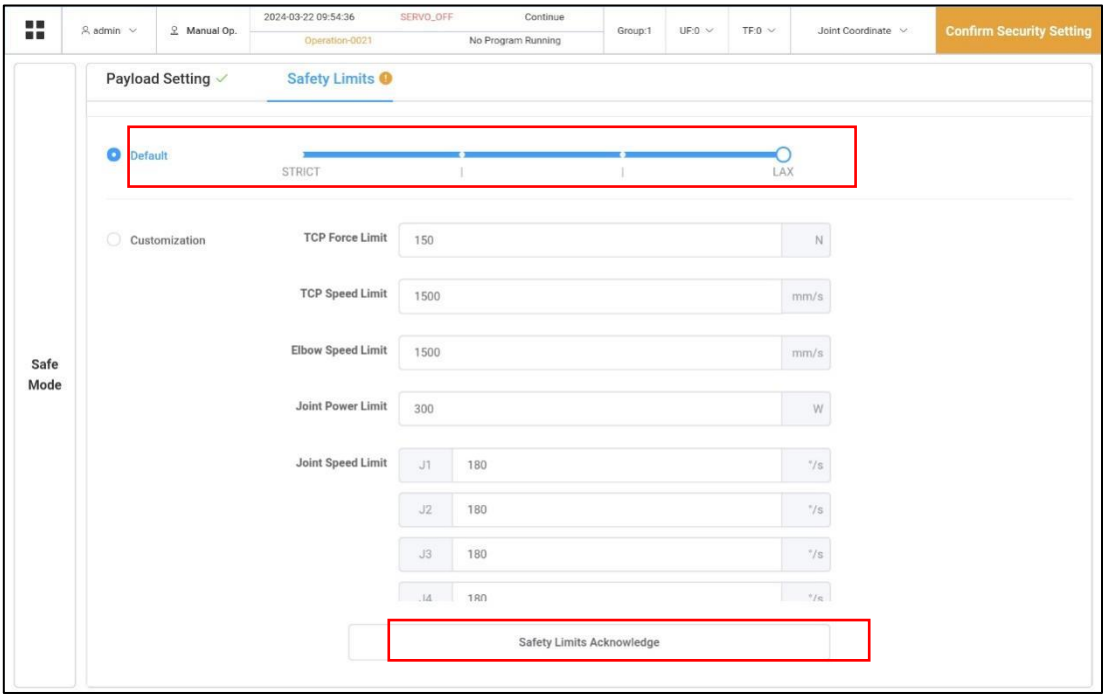




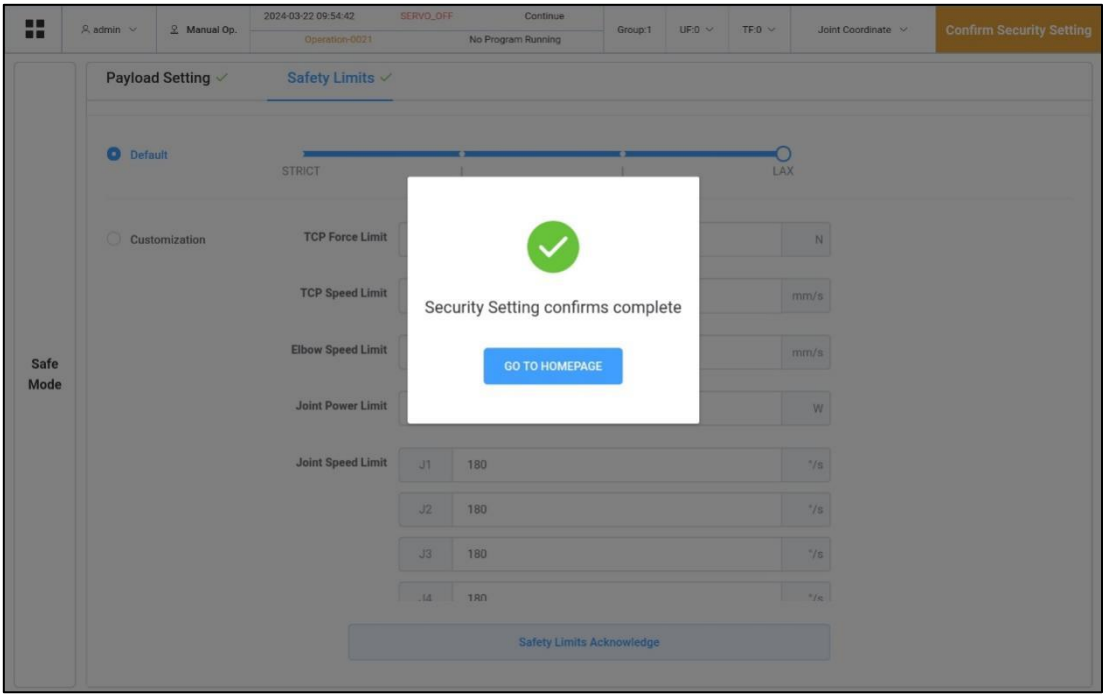
6. You can also use the default payload parameter values to complete the payload setting directly. After completing the setting for the first time, you can set the load again in "Menu Button" → "System Settings" → "Basic Settings" → "Load Setting". After completing the setting for the first time, set the load again in "Menu Button" → "System Settings" → "Basic Settings" → "Load Settings".

Safety Limits

1.When selecting the Default value, the parameter can not be changed. You can adjust the preset configuration through the safety restriction gear bar. After the configuration is completed, click on the Safety Limits Acknowledge to confirm.



2.Click to confirm the pop-up "Safety Limits Acknowledge", to complete the first-time use of the settings.



3. You can also click the Customization button to make specific settings for the security restrictions. After the configuration is completed, click on the Safety Limits Acknowledge to confirm.

Description of safety limits parameters

SAFETY LIMITS PARAMETERS		DESCRIPTION
TCP Force Limit		Limit the maximum force exerted by the robot tool in a clamping situation
TCP Speed Limit		Limit the maximum speed of the robot tool
Elbow Speed Limit		Limit the maximum speed of the robot's elbow
Joint Power Limit		Limit the mechanical work done by the robot
Joint Speed Limit	J1	Setting the upper limit of the speed of the J1 axis of the joint
	J2	Setting the upper limit of the speed of the J2 axis of the joint
	J3	Setting the upper limit of the speed of the J3 axis of the joint
	J4	Setting the upper limit of the speed of the J4 axis of the joint
	J5	Setting the upper limit of the speed of the J5 axis of the joint
	J6	Setting the upper limit of the speed of the J6 axis of the joint
Joint Soft Limit	J1	Setting the upper and lower limits of the soft limit of the joint J1 axis
	J2	Setting the upper and lower limits of the soft limit of the joint J2 axis
	J3	Setting the upper and lower limits of the soft limit of the joint J3 axis
	J4	Setting the upper and lower limits of the soft limit of the joint J4 axis
	J5	Setting the upper and lower limits of the soft limit of the joint J5 axis
	J6	Setting the upper and lower limits of the soft limit of the joint J6 axis

1.4.3 Power-on

Power-on sequence: Connect the power cord of the controller → Switch on the circuit breaker of the controller → Press the Power-on button on the handle → Wait for startup.

Connect the robot after 5-10s. The screen is as shown in Fig. 1.6.

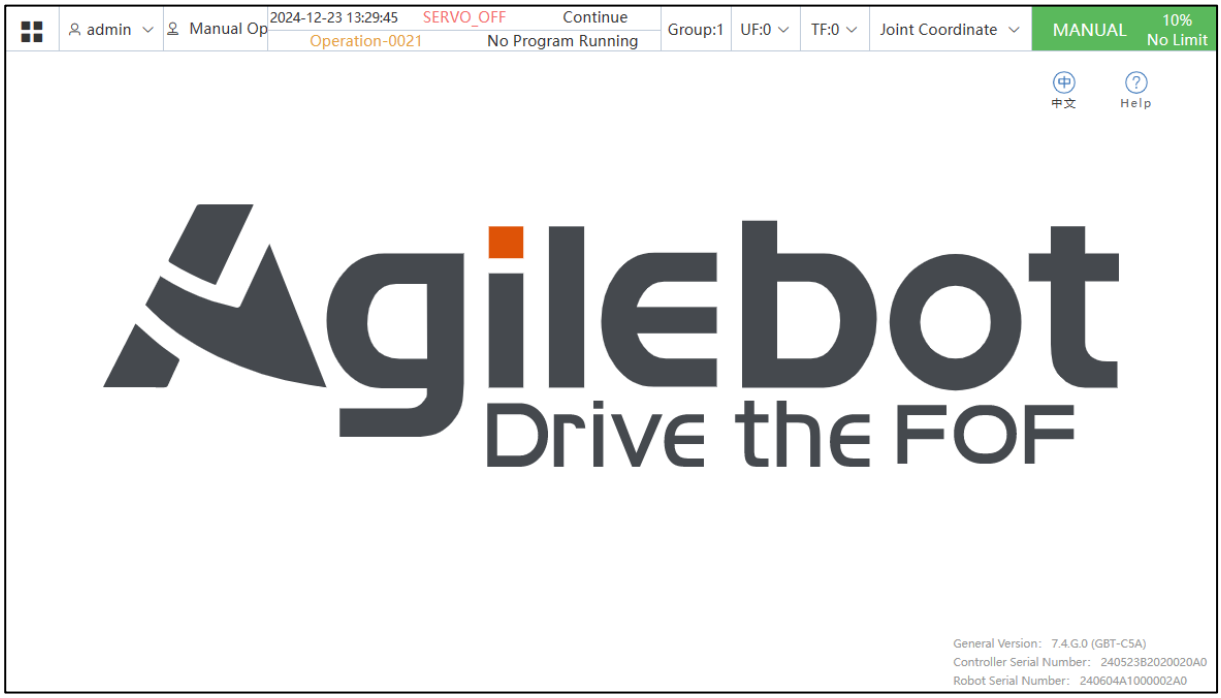


Fig. 1.6 Main Screen of Operation Interface

1.4.4 Shutdown

Shutdown sequence: First, ensure that the robot is in the stop state.

1) Enter the main menu and click "Power", as shown in Fig. 1.7.

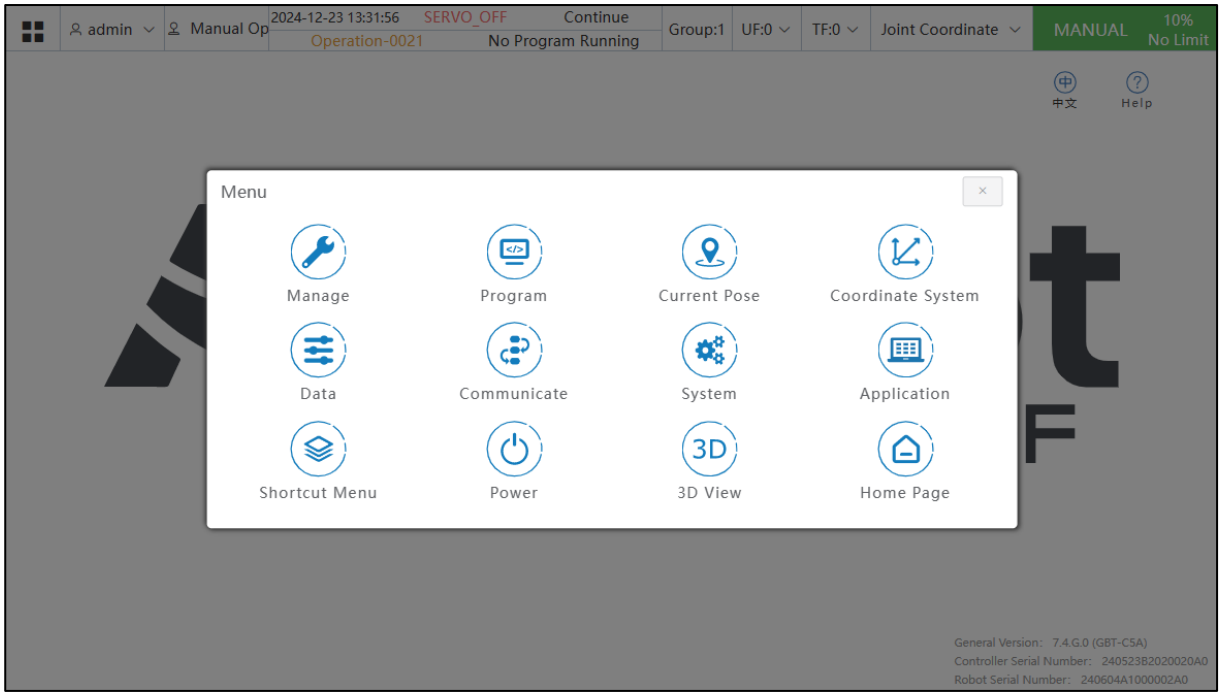
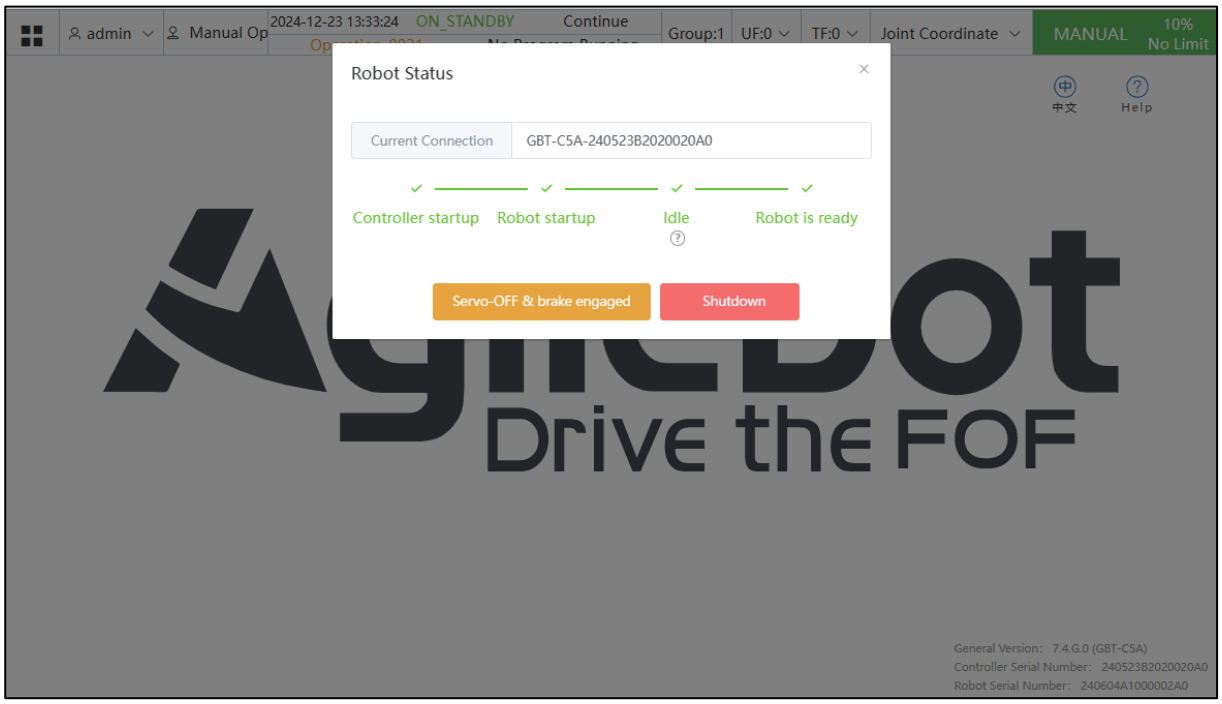
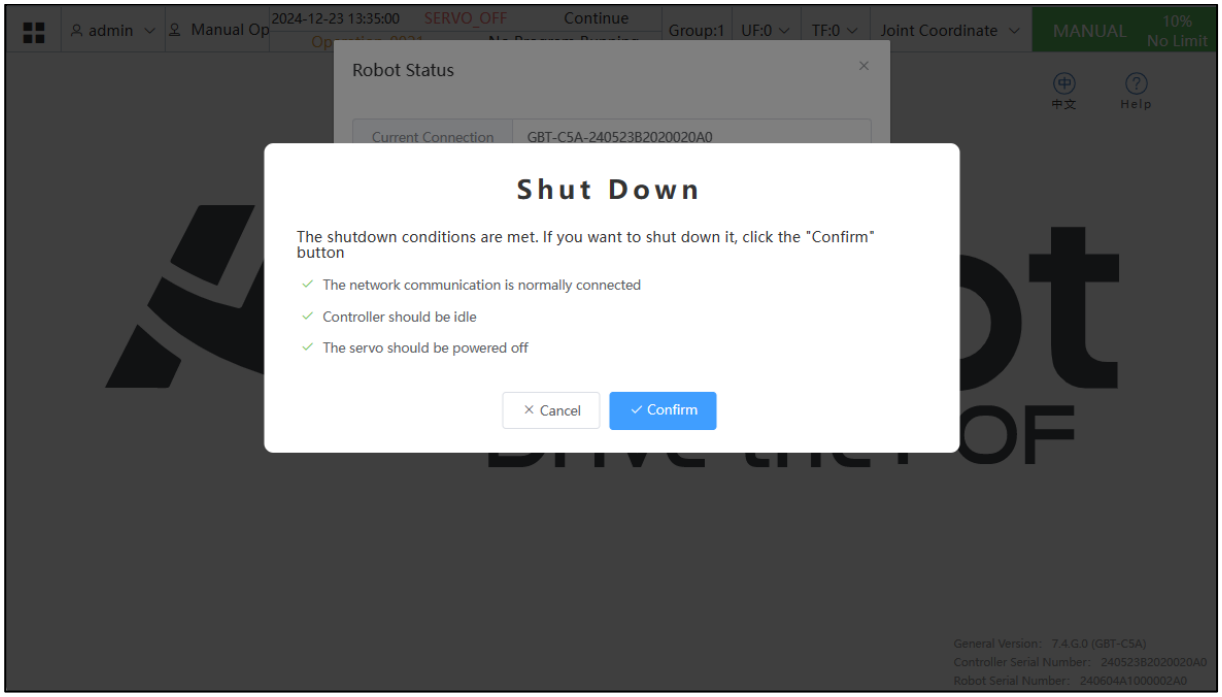


Fig. 1.7 Main Menu

2) Enter "Power" as shown in the following figure.



3) Click “Shutdown” as shown in the figure below.



4) Click “Confirm” to complete shutdown.

1.5 INTRODUCTION TO OPERATION INTERFACE

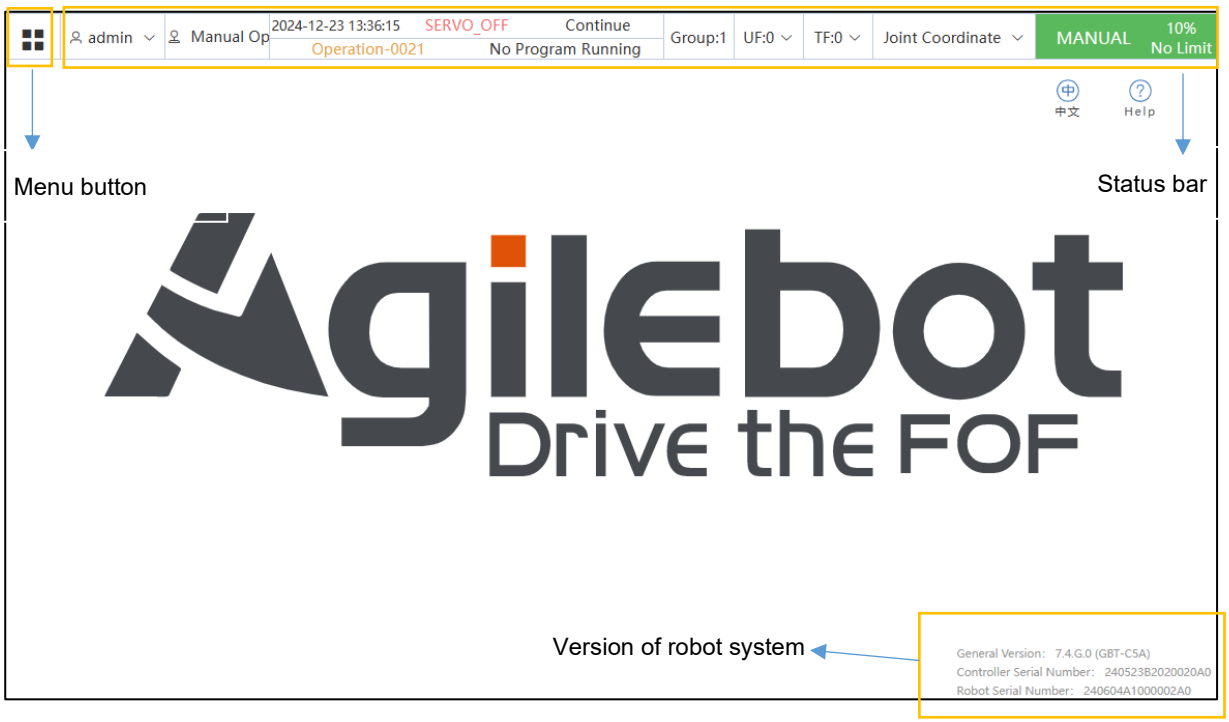
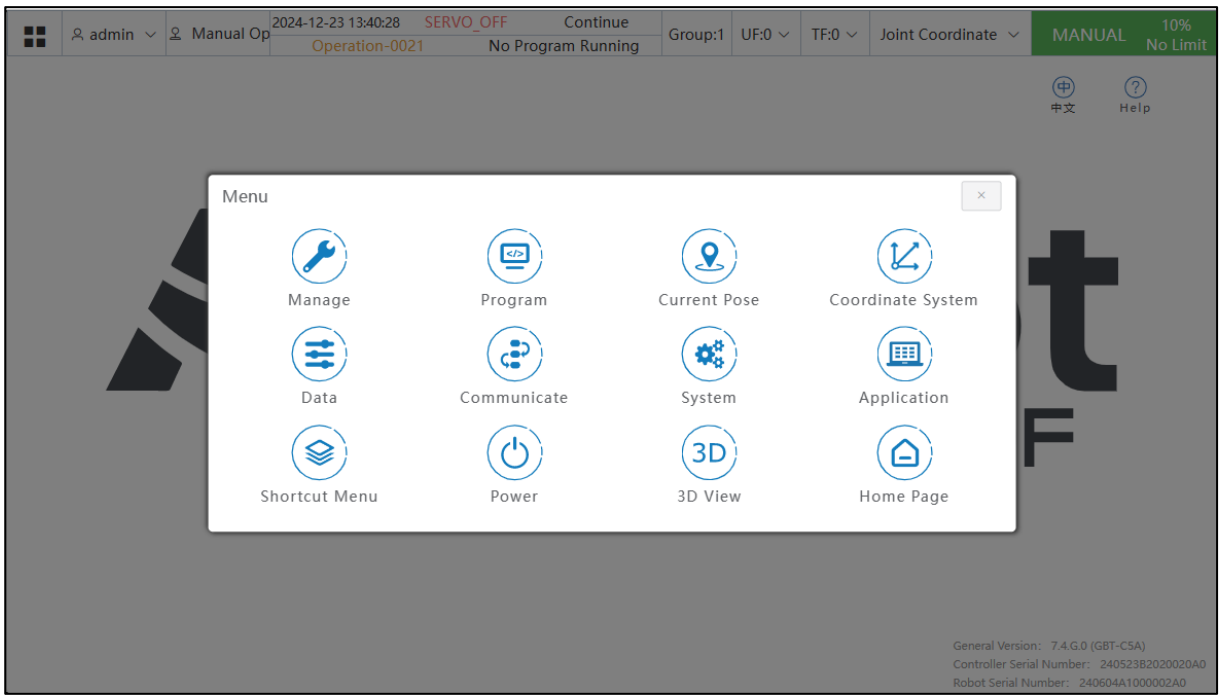


Fig. 1.8 Main Screen of Operation Interface

1.5.1 Menu

Click the "Menu" to pop up a menu list. The detailed menu list and sublist are as follows:



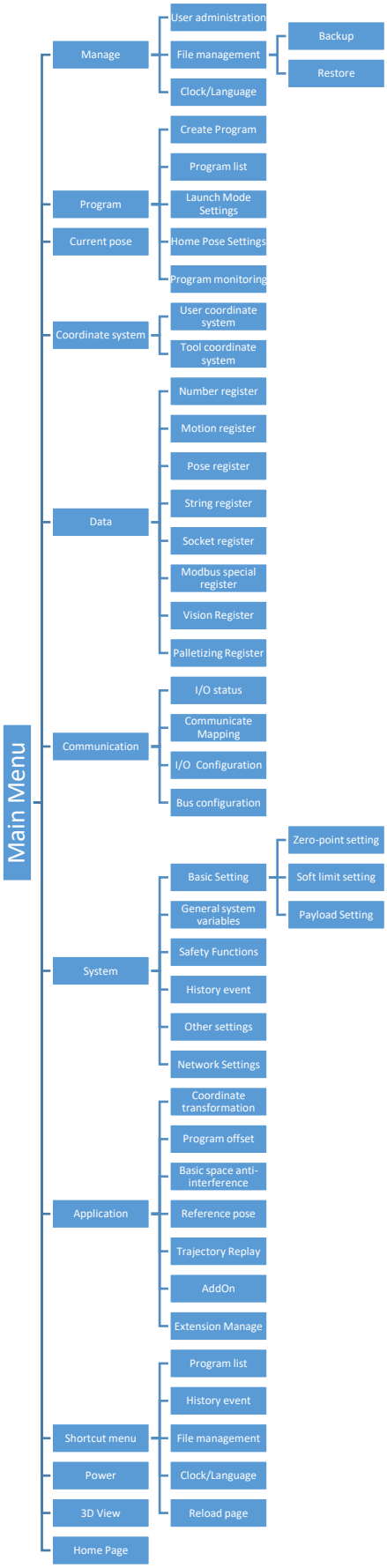


Fig. 1.10 Main Menu and Submenu List

1.5.2 Status bar

Fig. 1.11 Status Bar

Specific meaning of each component in the status bar:

Icons in status bar Form left to right	Specific meaning
1	User login (see Section 1.1 for specific authority definition)
2	Manual Op: Open the 3D interface to teach the robot.
3	Current system time
4	Robot's status
5	Program running mode
6	Alarm information
7	Program running status
8	Motion group
9	User coordinate system
10	Tool coordinate system
11	Current coordinate system
12	Operation mode
13	Setting of safety function
14	Overall velocity

1.5.3 System information

Path to detailed system version information: Home -> Help -> About

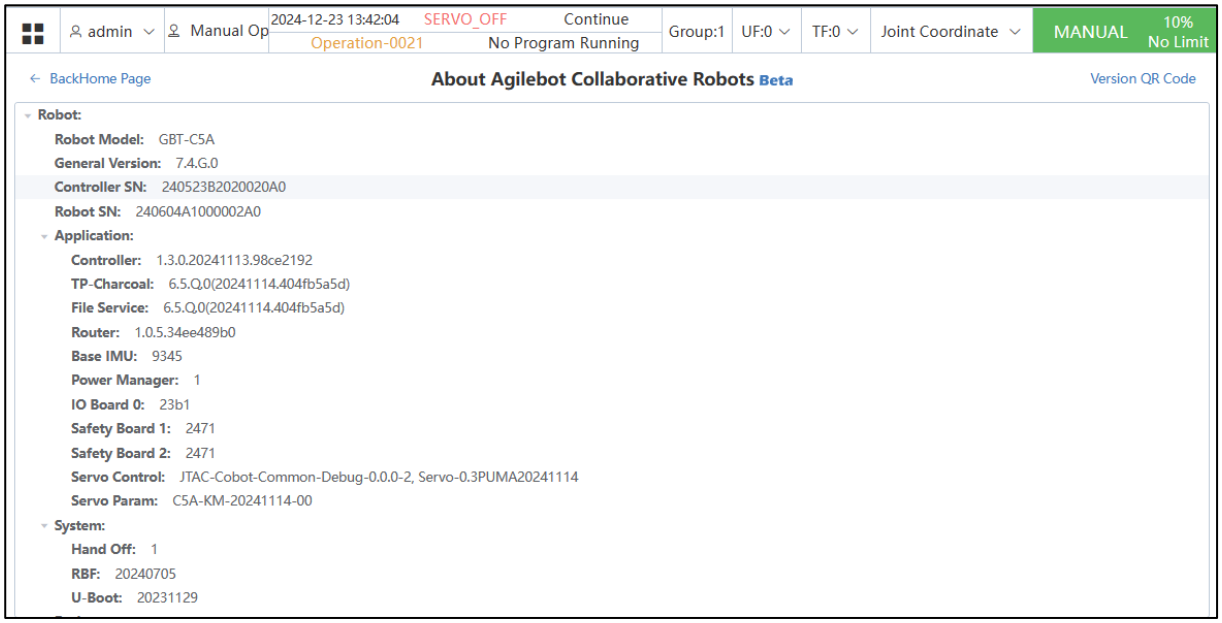


Fig. 1.12 Version of System Software

1.6 MODE SWITCH

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

Fig. 1.13 Mode Switch

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

1. **Manual 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.

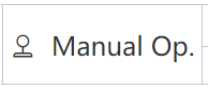
- Teach the robot.
- Debug executive programs, including positive-sequence single-step executive programs and reverse-sequence single-step executive programs.
- Edit and modify robot programs.
- Modify relevant configurations of the robot.
- Enter the drag mode.

1.7 ROBOT TEACHING

Teach the robot according to the joint coordinate system (please refer to 2.2 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 switch to L or M mode (manual mode),meuu 

or 

open 3D view.

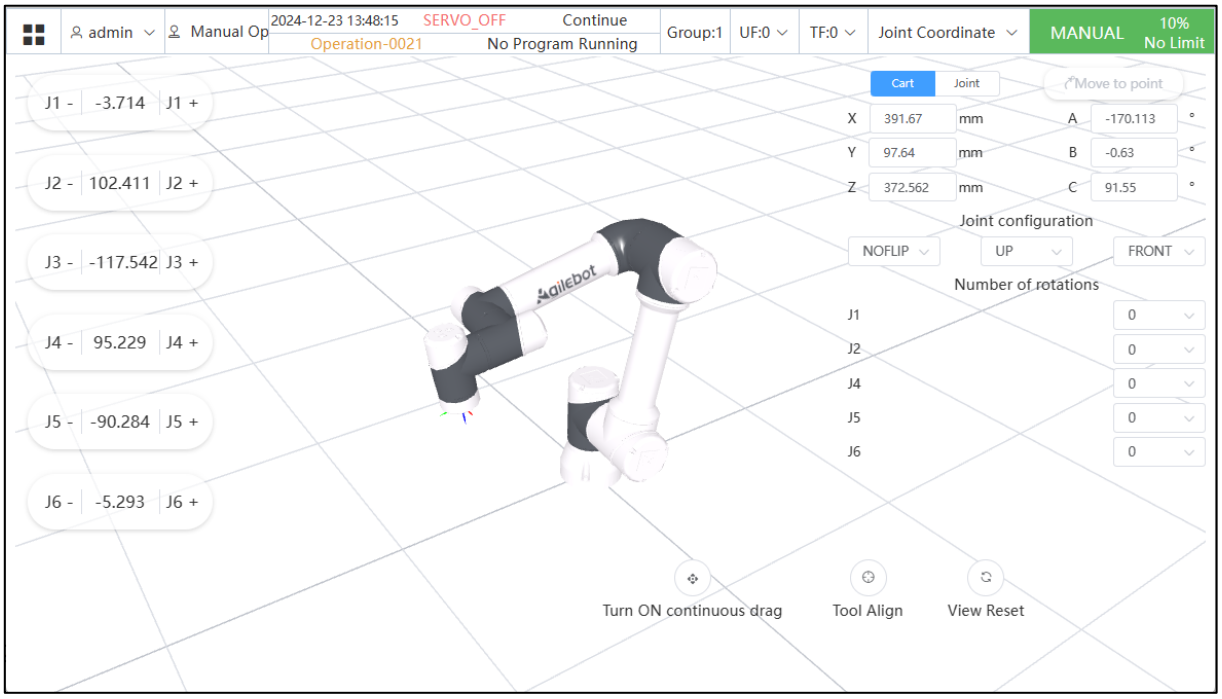


Fig. 1.14 3D view

2. In the status bar, select the reference coordinate system for robot operation as shown in Fig. 1.18.

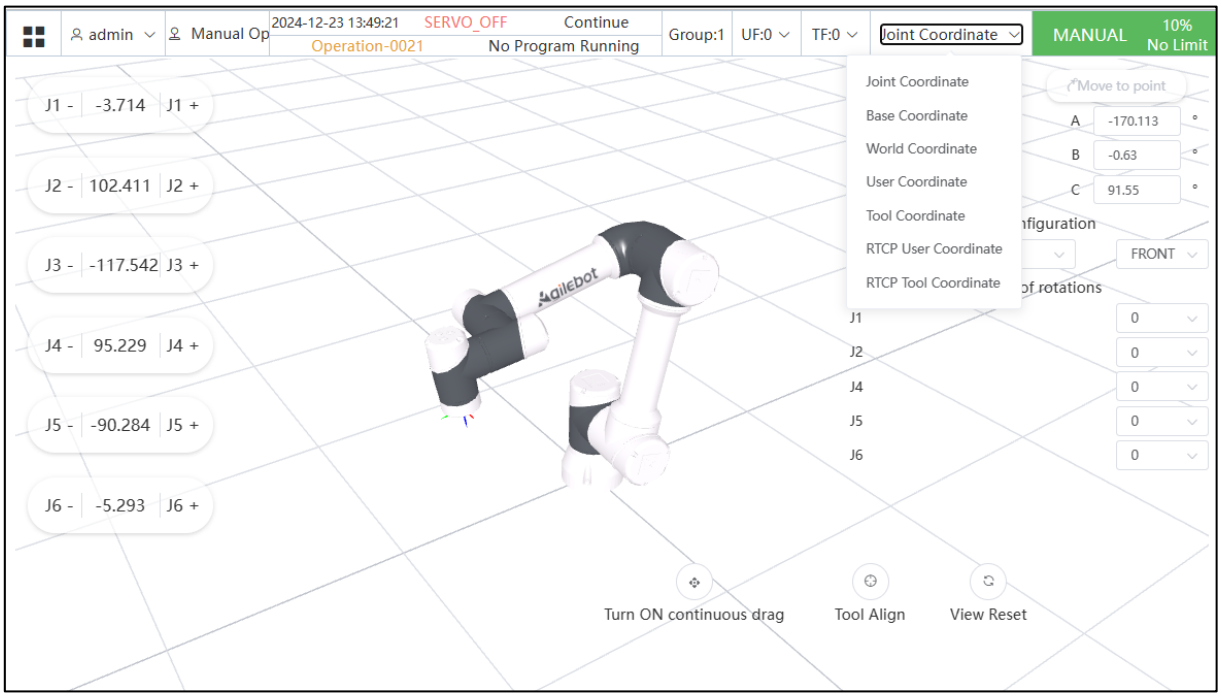


Fig. 1.18 Reference Coordinate System

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. Then, the coordinate system is activated successfully.

Fig. 1.15 Selection of User or Tool Coordinate System

4. The robot may move accordingly when a corresponding axis or XYZ direction is clicked.

5. Press the "Reset" button on the interface or the handle to clear the alarm and excite the robot. 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 and jog the corresponding position to move.

> ⚠ 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.

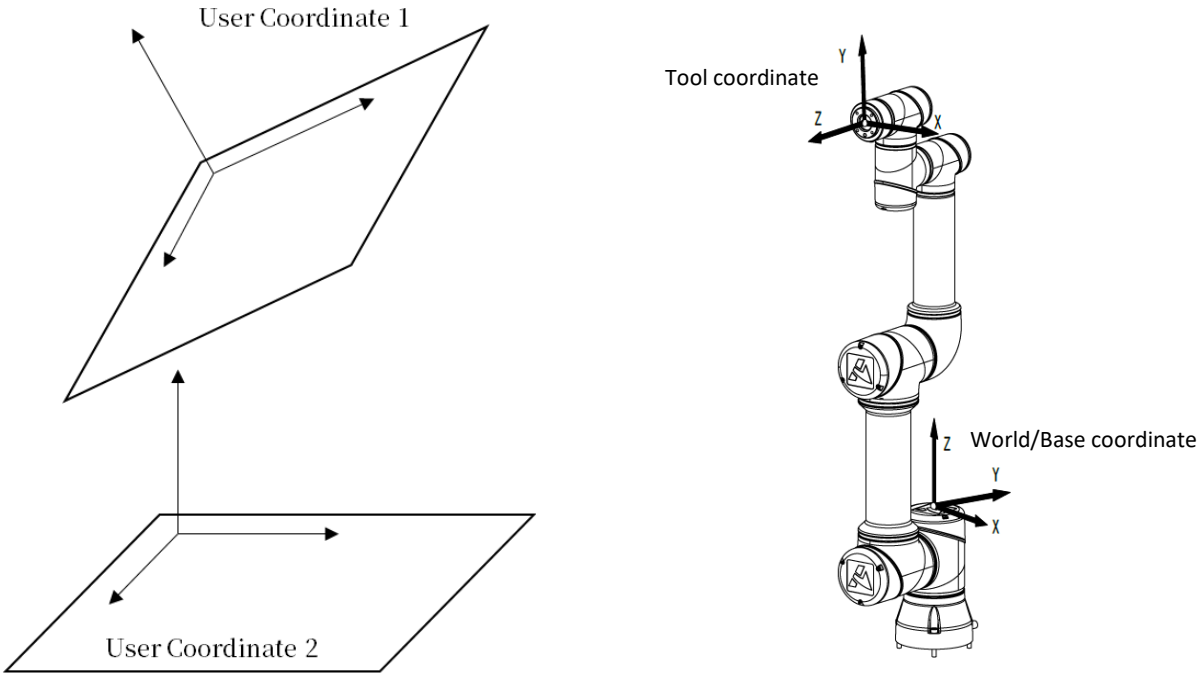
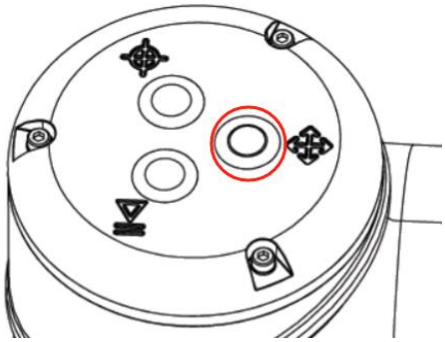


Fig. 1.16 Coordinate System of the Robot

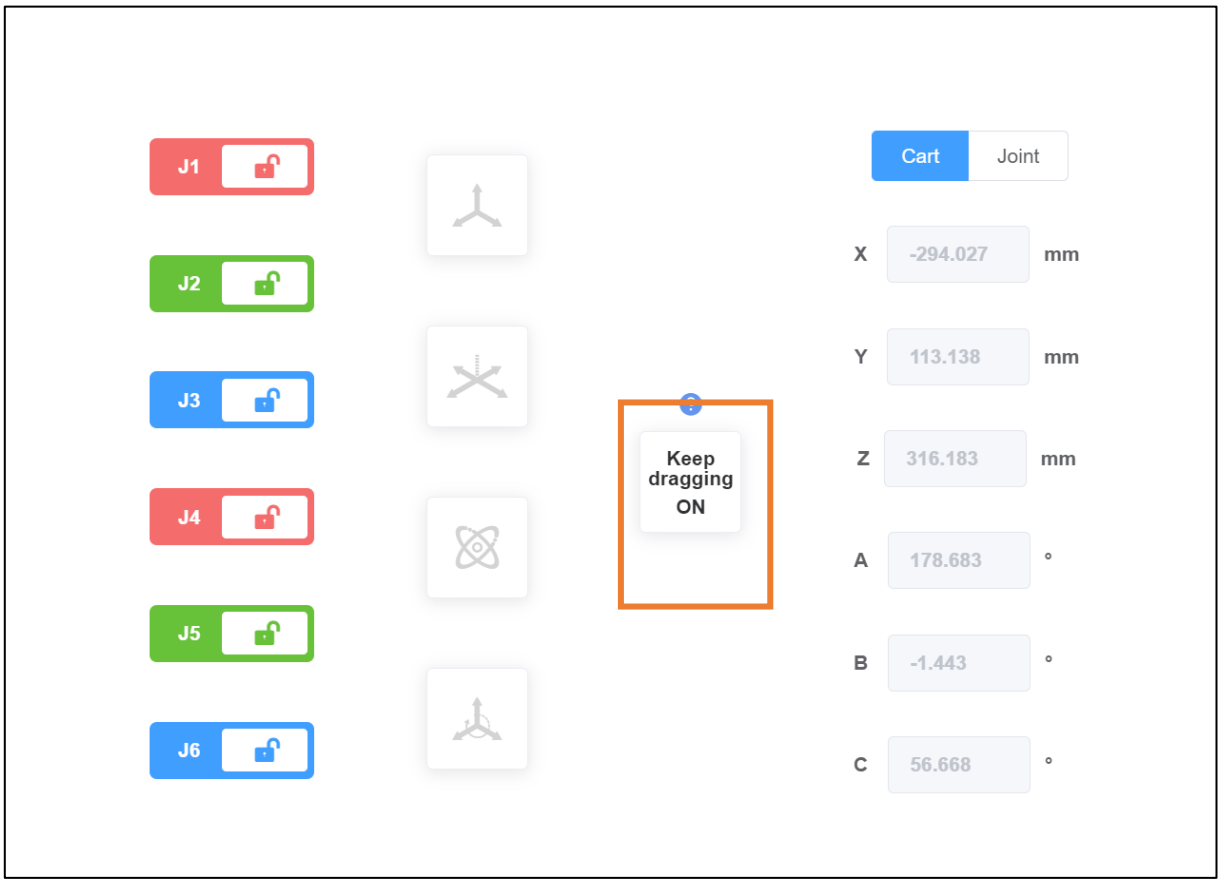
Drag Button

1.After the robot is powered on and enabled, hold down the drag button at the end of the robot, as shown in the following figure.



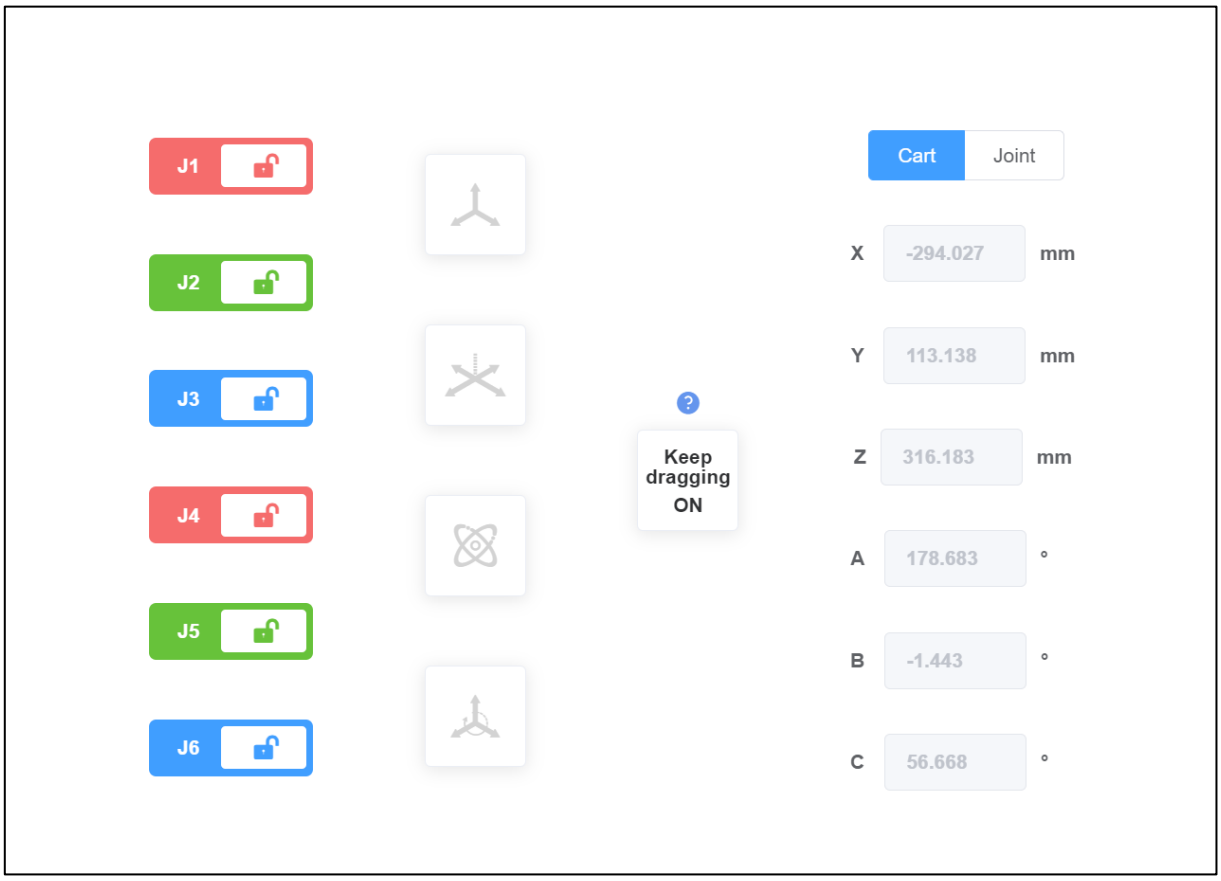
2.The light strip on the robot turns yellow and it enters the dragging interface, as shown below. At this time, the robot can be dragged.

- Continuous Dragging ON: Press the drag button once to enter the dragging interface. Press the same button again to exit the dragging state.
- Continuous Dragging OFF: Keep pressing the drag button to enter the dragging state.

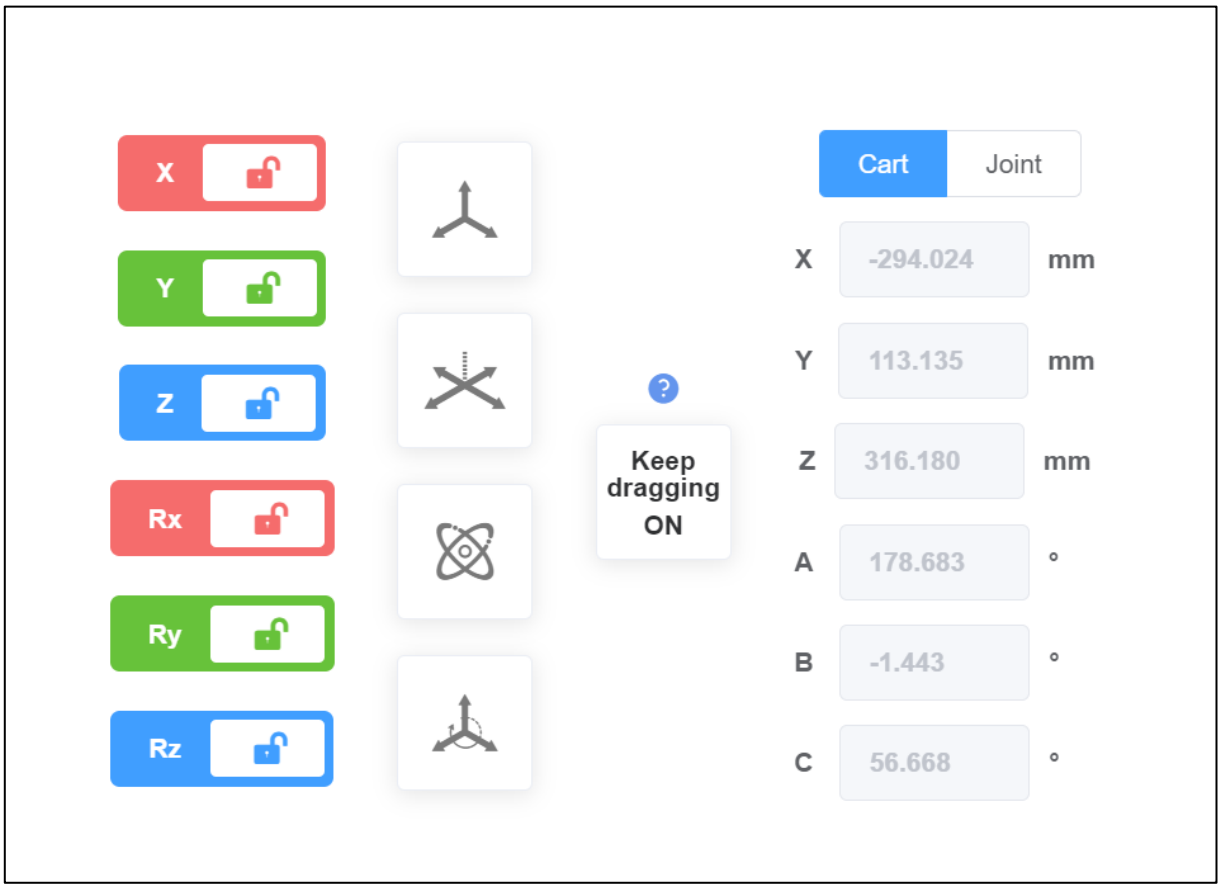


There are slight differences in entering the dragging interface under different coordinate systems:

Joint Coordinate System: Drag the robot's J1, J2, J3, J4, J5, and J6 joints, as shown in the following figure.



Rectangular Coordinate System: Drag the robot along X, Y, Z, Rx, Ry, and Rz axes, as shown in the following figure.



Control the positions of the tool in X, Y and Z directions.

Control the positions of the tool in X and Y directions.

Control the orientations of the tool in Rx, Ry and Rz directions.

Control the positions of the tool in X, Y and Z directions and the orientations of the tool in Rx, Ry and Rz directions.

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 switch.

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)

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)

When the speed is set manually, the robot can only move the corresponding step by setting the feed amount. The feed range is 0-10mm (the corresponding joint coordinate movement step length and the rotation step length of the attitude Angle are 0°-5°).

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

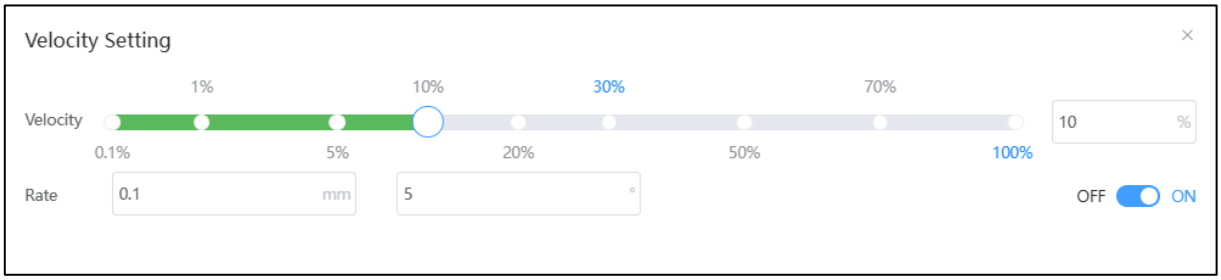


Fig. 1.21 Speed Setting Window

1.9 SAFETY FUNCTIONS

1.9.1 Safety Limitations

The safety limitations take effect immediately after the robot is powered on. Users or integrators need to complete a safety assessment based on the actual on - site situation and set the safety limitation values accordingly.

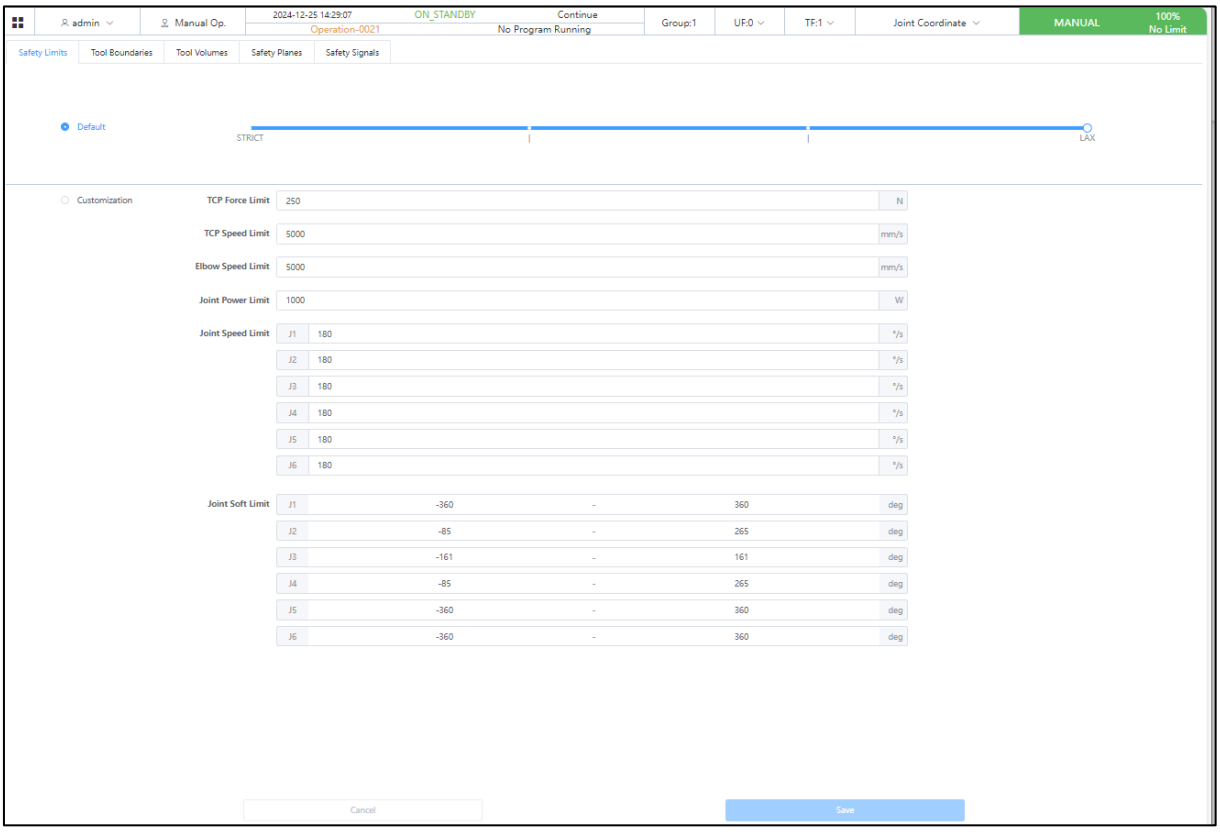


Figure 1.1 Safety Limitation Setting Interface

Safety Limitation Value Parameters:

Table 1.4.2 Explanation of Safety Limitation Parameters

Safety Limitation Parameter		**Explanation**
TCP Force Limit Value		Limits the maximum force exerted by the robot tool when
TCP Speed Limit Value		Limits the maximum speed of the robot tool.
Elbow Speed Limit Value		Limits the maximum speed of the robot's elbow.
Safety Limitation Parameter		**Explanation**
Joint Speed Limit Value	J1	Sets the upper limit of the speed of joint J1 axis.
	J2	Sets the upper limit of the speed of joint J2 axis.
	J3	Sets the upper limit of the speed of joint J3 axis.
	J4	Sets the upper limit of the speed of joint J4 axis.
	J5	Sets the upper limit of the speed of joint J5 axis.
	J6	Sets the upper limit of the speed of joint J6 axis.

Safety Limitation Parameter		Explanation
Joint Soft Limit	J1	Sets the upper and lower limits of the soft limit of joint J1
	J2	Sets the upper and lower limits of the soft limit of joint J2
	J3	Sets the upper and lower limits of the soft limit of joint J3
	J4	Sets the upper and lower limits of the soft limit of joint J4
	J5	Sets the upper and lower limits of the soft limit of joint J5
	J6	Sets the upper and lower limits of the soft limit of joint J6

Safety Limitation Value Entries that Users Can Set (The limitation values in this table are only for C5A)

	Limitation Item	Limitation Value				Safety State	Operation Permission
		Maximum Limitation	More Limitation	Moderate Limitation	Minimum Limitation		
Safety Limitation	CP Force (N)	100	120	150	250	Class 0 Stop	Administrator
	TCP Speed (mm/s)	250	750	1500	5000		
	Elbow Speed (mm/s)	250	750	1500	5000		
	Total Joint Mechanical Power (W)	80	200	300	1000		
Joint Speed Limitation(°/s)	J1	212					
	J2	212					
	J3	212					
	J4	212					
	J5	212					
	J6	212					
Joint Soft Limit (°)	J1	±360					
	J2	-85 to +265					
	J3	±165					
	J4	-85 to +265					
	J5	±360					
	J6	±360					

Warning

When the TCP approaches a singular position, it is necessary to consider whether the joints moving at high speeds are likely to cause harm to personnel. If there is such a risk, it is essential to limit the speed and force of the fast-moving joints that are prone to causing injuries under this working condition. Especially when approaching a singular pose during linear motion, it is crucial to ensure the safety of personnel and the stable operation of the equipment.

1.9.2 Limit the posture of the tool

It is used to limit the posture of the tool. In scenarios such as robot operations, when dragging the tool or programming, in order to avoid the tool pointing in directions that may cause danger, it is necessary to constrain its posture.

The rotation range of the +Z axis of the Tool Frame (TF) around the Tool Center Point (TCP) is limited within a cone with the TCP as the vertex. The axis of the cone is parallel to the +Z axis of the specified coordinate system (which can only be the User Frame (UF), the base coordinate system, or the world coordinate system), and the positive direction of the cone axis is the same as that of it. The range of the movable area is controlled by setting the half-angle of the cone.

Setting Steps:

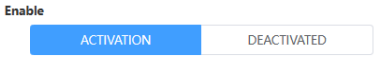
1、Click and in sequence 

、

、 

to enter the tool posture limitation configuration page.

2、Select the TF to be configured in the left tool list, and click on the right side

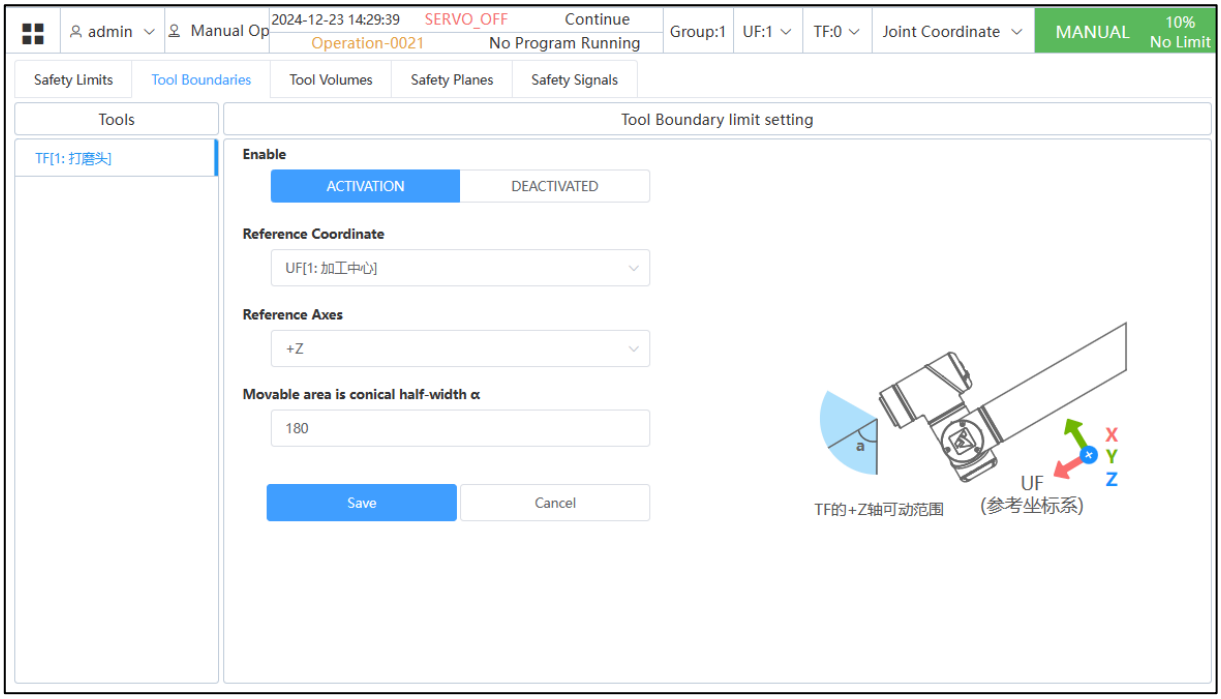


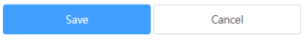
to activate the limitation.

3、Select the reference coordinate system 

and the reference coordinate axis

used to define the posture limitation cone. The reference coordinate axis is set to "+Z" by default.



4、Click 

to save the settings.h

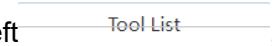
1.9.3 Tool Volume

The tool volume is a virtual volume block that completely encloses the outline of the actual tool. It is used to describe the size and position of the simplified tool, so that the robot can prevent collisions between the tool and its body in motion planning, or detect the position of the tool in space in real time, which is convenient for further judgment and control. For example, by setting the tool volume and the space where the tool is allowed to move, the tool can only move within the specified range.

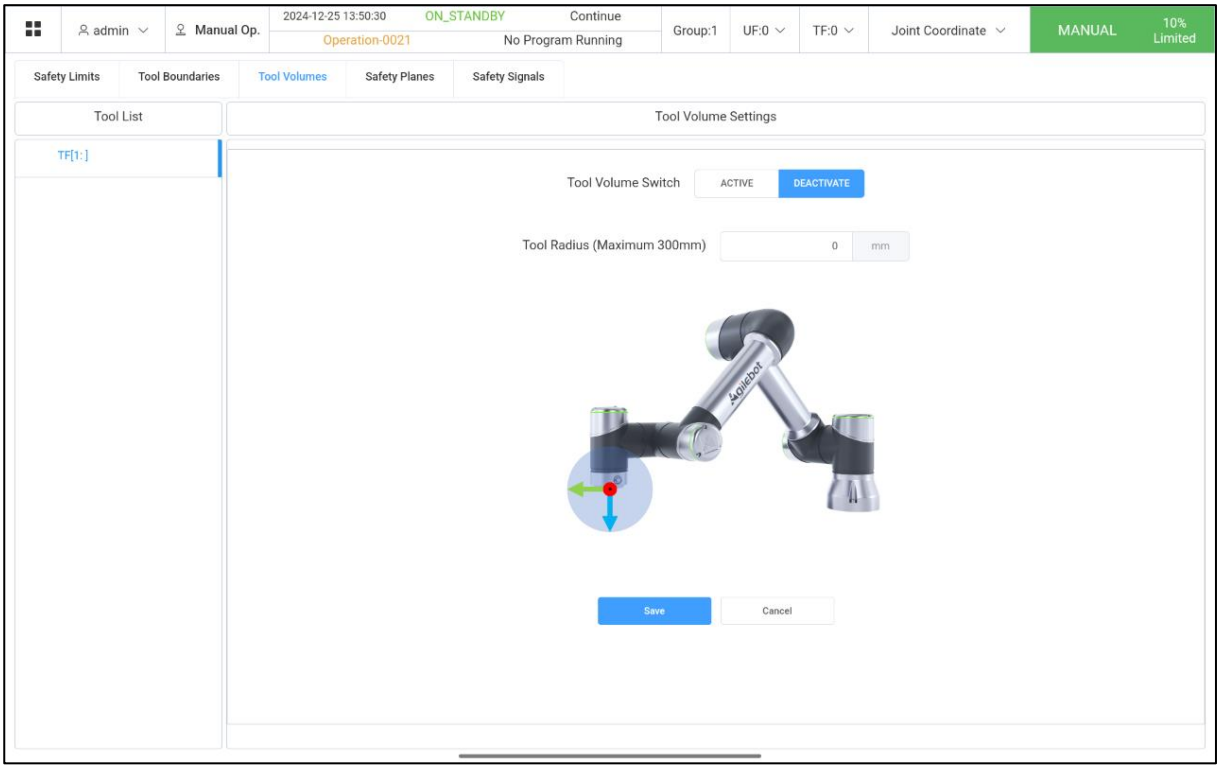
Setting Steps:

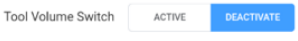
1、Click and in sequence 

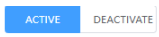
to enter the tool volume setting page.

2、In the left 

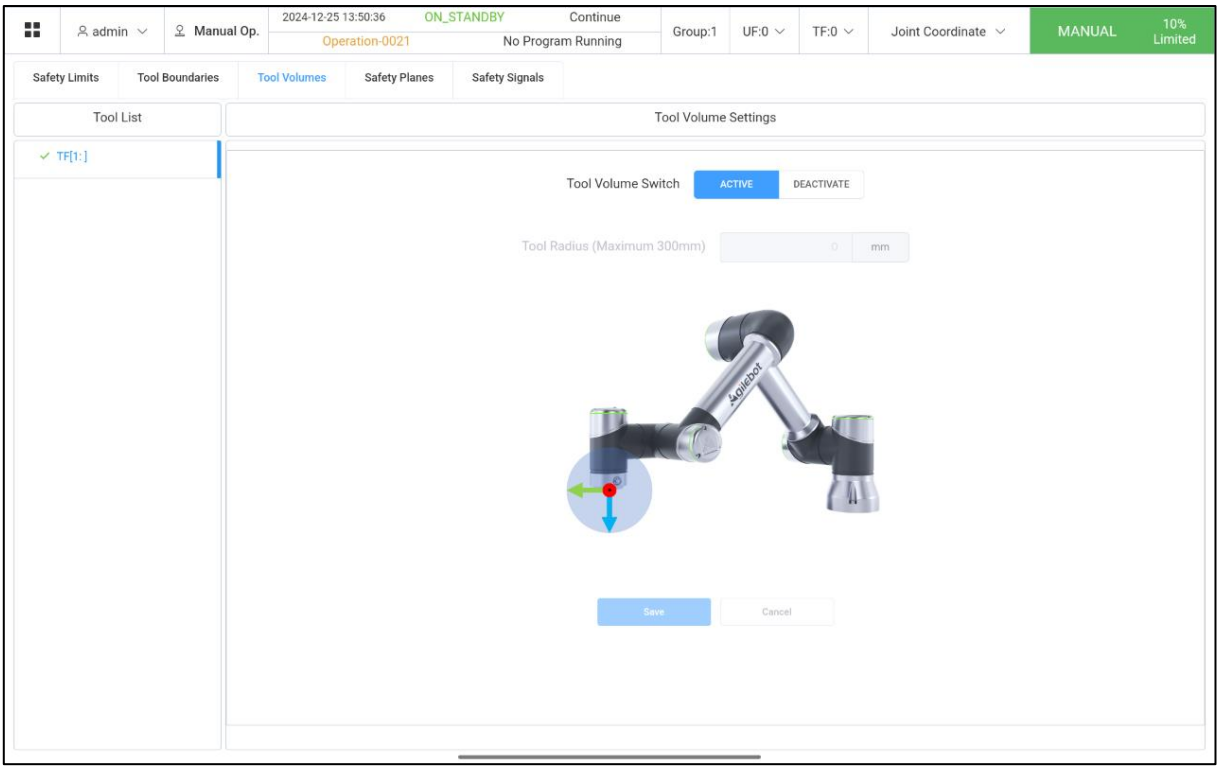
, select the TF for which the tool volume needs to be set. The maximum value of the tool radius in the figure is only for illustration purposes.



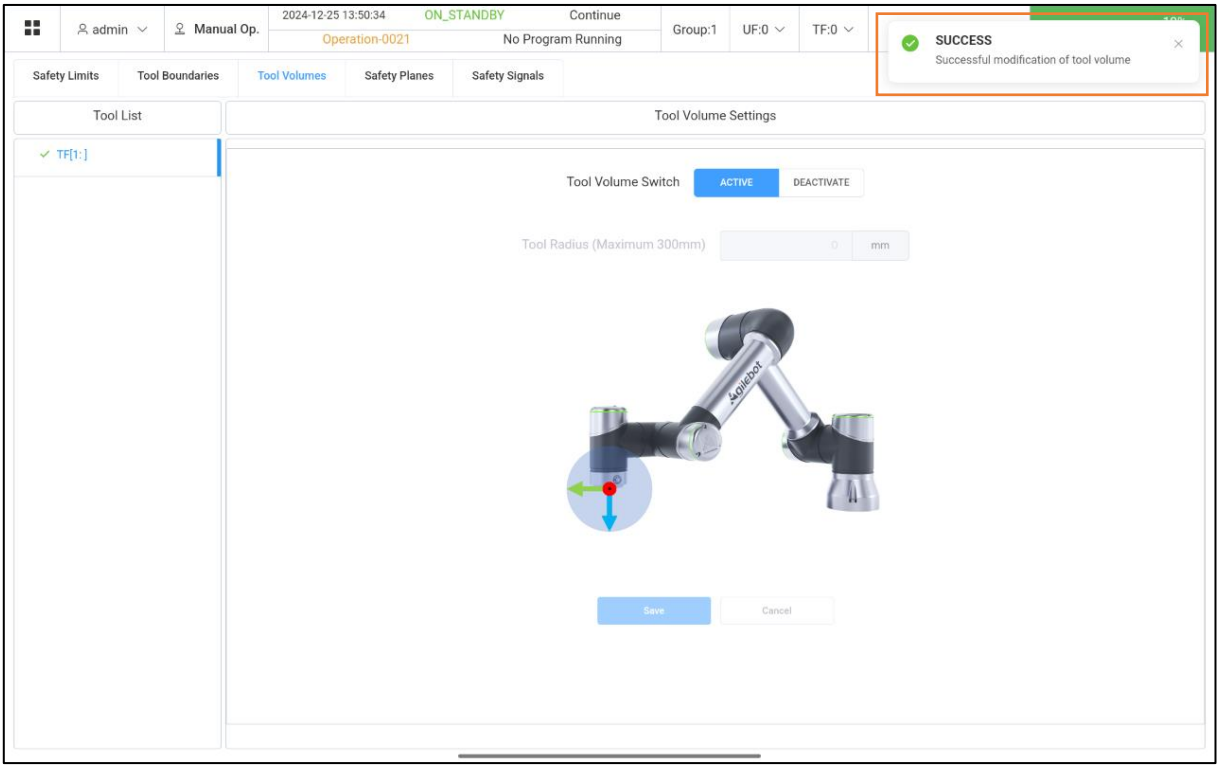
3、Change the on the right side 

to and set the radius of the tool sphere, and then click "Confirm" 

to complete the setting.



4、If the user does not click "Confirm" or "Cancel" after setting, Successful modification of tool volume.



1.9.4 Safety Plane

The safety plane is a virtual plane used to limit the working range of the robot tool or elbow (the third joint) or serve as a trigger condition. It is determined by three points in space. For example, the tool (tool volume) is not allowed to pass through this plane; or when the tool (tool volume) passes through this plane, the robot will slow down.

The safety plane has the following functions:

- Prohibit the Tool Center Point (TCP) or tool volume from passing through；
- *Prohibit the elbow (the third joint) from passing through;

Each safety plane is defined by teaching three points in space, and a maximum of eight safety planes can be defined.

The defined safety planes can be activated or deactivated. The establishment method of the safety plane is consistent with that of the User Frame (UF) three-point method. The Z-axis is the normal direction of the safety plane, and the positive direction of the Z-axis of each safety plane is the safety area where the robot can move freely.

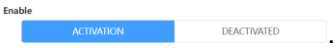
Setting Steps:

1、Click and in sequence 

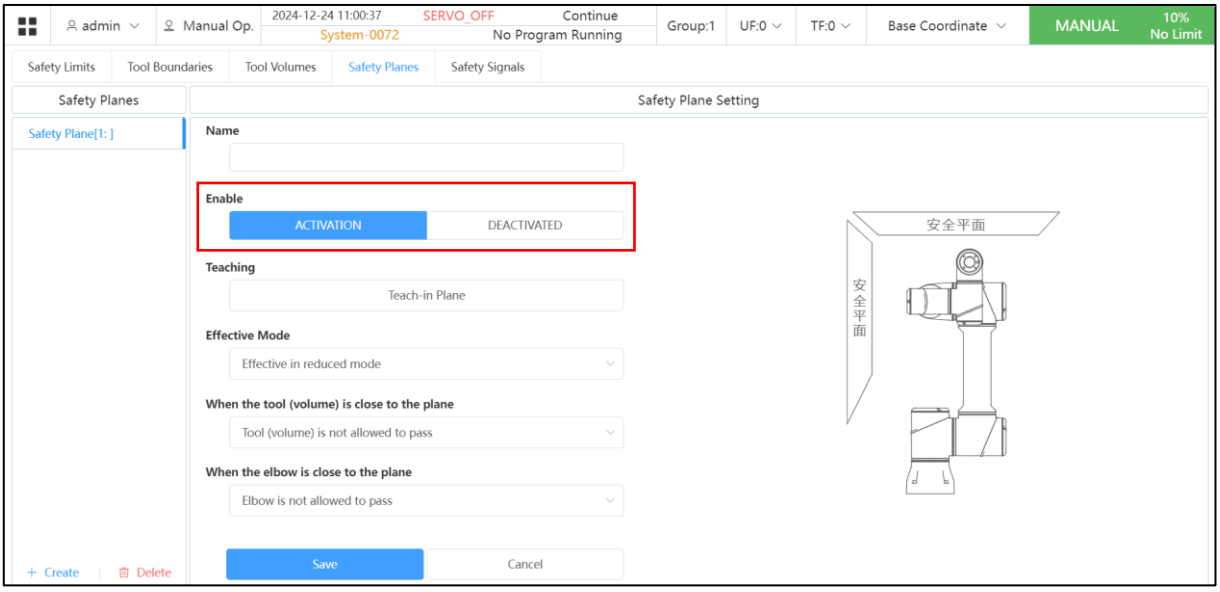
to enter the safety plane setting page.

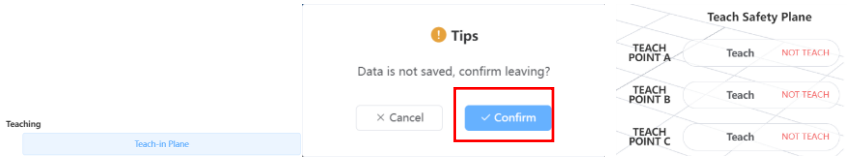
2、Click on the left 

to create a new safety plane, and then change 

to 

.



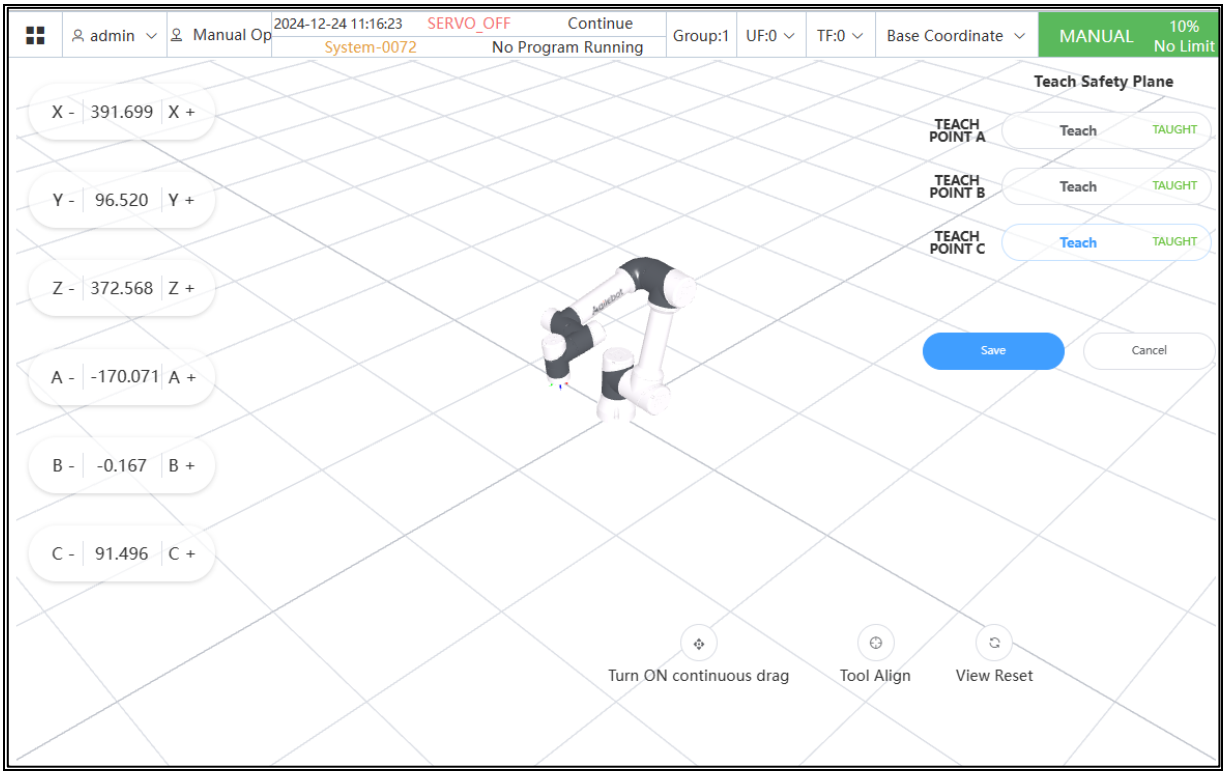
3. 

Click the three buttons in in sequence to teach 

three points, and then a safety plane can be taught. The teaching interface is similar to that of the "Virtual Jogging" interface. Among them, the "Name" cannot be changed. If the three teaching points are not modified, the newly created safety plane will be a plane that coincides with the base mounting surface by default. Click 

and 

to define the behaviors when the tool (or tool volume) and the elbow approach the plane.



4、If the user does not click "Confirm" or "Cancel" after setting.

1.9.5 Safety I/O Signals SI/SO

Safety I/O is part of functional safety and is a highly reliable I/O interface. Its signal lines exist in pairs to ensure that I/O signals can reliably enter and exit the controller through the safety I/O interface.

It is used to connect various sensors or safety devices to achieve diverse protection functions rather than simply stopping. For example:

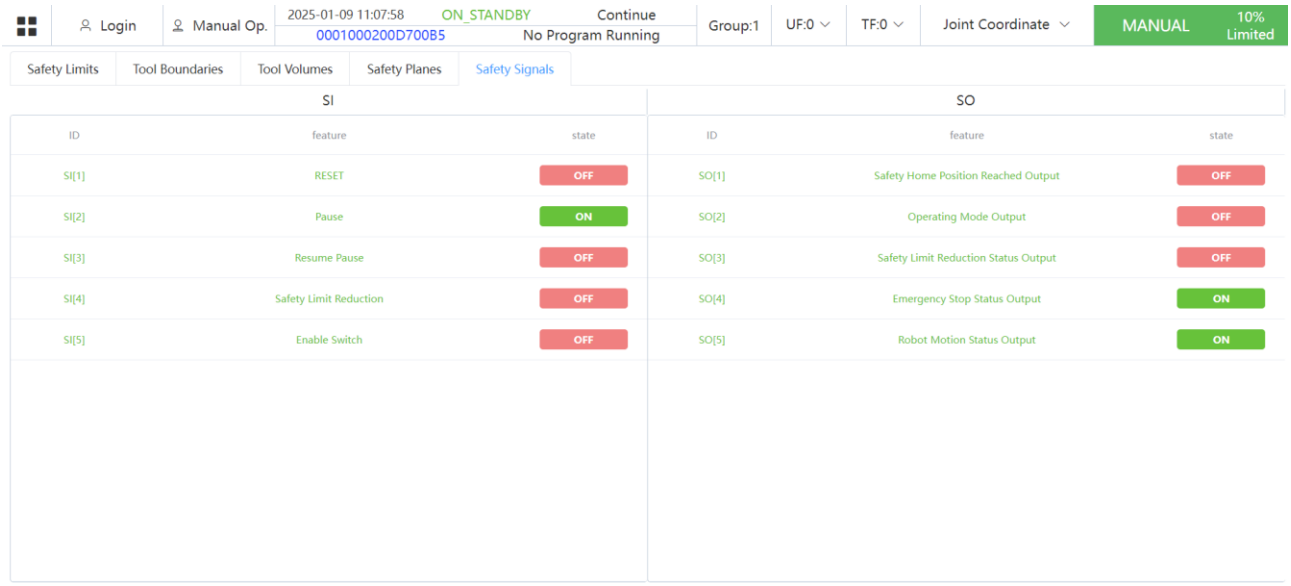
① When connected to a laser scanner, it can be set in the software that when the laser scanner is triggered, the robot will slow down by 50%. It can also be connected to a safety mat, and when the mat is triggered, the robot will execute a Category 2 stop.

② SO1 and SO2 are connected to a safety Programmable Logic Controller (PLC). It can be set in the software that when the robot is in motion, the guardrails around the robot will be locked. Meanwhile, SO3 and SO4 are connected to the protective stop interface of external devices, and when the robot stops moving, the external devices will also stop running.

It can be seen that multiple safety I/O interfaces are needed to achieve diverse protection effects.

Safety I/O Interface

The statuses of SI and SO can be viewed by entering from System and clicking Safety Functions Safety Signals, but they cannot be modified.



SI (Safety Digital Input)			
Serial Number	**Name**	**Functional Description**	**Fault Handling**
1	Reset	[Effective only in AUTO mode, used for remote enabling] The function is the same as the UI signal of the industrial robot: RESET, but the hardware interface is changed from a common I/O to a safety I/O. It is effective when a rising edge is detected.	If the safety system detects that the robot fails to complete the implementation of the corresponding function within the specified time, a Class 0 stop will be triggered.
2	Pause	When the input signal is "low", the program pauses and the robot executes a Class 2 stop.	

3	Resume Pause	[Effective only in AUTO mode, used for automatically resuming the robot's motion] When the input signal is "high", the program continues to run. Note: This signal can only allow a paused program to continue running, but it cannot start a program.	
4	Safety Limit Reduction	The function is not available	
5	Enable switch	A three - position enabling switch can be externally connected, and after connection, the function is consistent with the enabling switch of the industrial robot teach pendant.	

SO (Safety Digital Output)			
Serial Number	**Name**	**Functional Description**	**Fault Handling**
1	Safety Home Position Reached Output	When the robot has reached the set safety posture, the output is "high". The joint angle error does not exceed ±0.5°.	1. If the safety system detects that the robot is entering, in the "FAULT" state or exiting from the "FAULT" state, all safety outputs will be set to "low". 2. If the safety system detects that the actual movement of the robot exceeds the safety limits, all safety outputs will be set to "low".
2	Operating Mode Output	When the robot is in "AUTO" mode, the output is high; when the robot is in "MANUAL" or "Limit MANUAL" mode, the output is low.	
3	Safety Limit Reduction Status Output	The function is not available	
4	Emergency Stop Status Output	When the robot is in an emergency stop state or the emergency stop button is pressed, the output is "low", otherwise it is "high".	
5	Robot Motion Status Output	When the robot is in motion or is about to start moving, the output is "low"; when the robot has completely stopped, the output can change to "high".	

Caution

SI/SO cannot be set to bypass through software.

2 SETTING OF ROBOT SYSTEM

2.1 I/O 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.

Tool I/O

Tool I/O represents the I/O interfaces on the robot arm.

Robot I/O TDI[1]~TDI[2]

TDO[1]~TDO[2]

TAI[1]~TAI[2]

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).

The following table details the default sequence definition of each port on the IO board: (The actual function can be set according to the use scenario)

Signal Name	Signal Specification
UI1	Servo excitation locking
UI2	Pause signal
UI3	Reset signal
UI4	Program start/resume signal

Signal Name	Signal Specification
UI5	Program termination request signal
IO_0V	0V
UI6	Trigger signal
UI7	MPLCS Start signal
UI8	MPLCS Main program status signal, a total of 6 bits
UI9	
UI10	
IO_0V	**0V**
UI11	MPLCS Main program status signal, a total of 6 bits
UI12	
UI13	
UI14	Drag mode signal
DI1	User input signal
DI2	User input signal
IO_0V	0V
DI3	User input signal
DI4	User input signal
DI5	User input signal
DI6	User input signal
DI7	User input signal
IO_0V	**0V**
DI8	User input signal
DI9	User input signal
DI10	User input signal
DI11	User input signal
DI12	User input signal
IO_0V	0V
IO_0V	0V
IO_0V	0V
Default setting	**Signal specification**
DO_PS_IN6	**DO Power selection interface**
DO12	User output signal
DO11	User output signal
DO10	User output signal
DO9	User output signal
DO8	User output signal
DO_PS_IN5	**DO Power selection interface**
DO7	User output signal
DO6	User output signal
DO5	User output signal
DO4	User output signal
DO_PS_IN4	**DO Power selection interface**
DO3	User output signal
DO2	User output signal
DO1	User output signal
UO13	MPLCS Main program status feedback, a total of 6 bits

Signal Name	Signal Specification
DO_PS_IN3	**DO Power selection interface**
UO12	MPLCS Main program status feedback, a total of 6 bits
UO11	
UO10	
UO9	
DO_PS_IN2	**DO Power selection interface**
UO8	MPLCS Main program status feedback, a total of 6 bits
UO7	MPLCS Start signal
UO6	MPLCS Select request
UO5	Servo status signal
DO_PS_IN1	**DO Power selection interface**
UO4	The program is in execution
UO3	Alarm signal
UO2	Pause signal
UO1	Robot operating status/mode

For controller's signal interface and definition, refer to the Instructions for Controller.

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. 2.1.

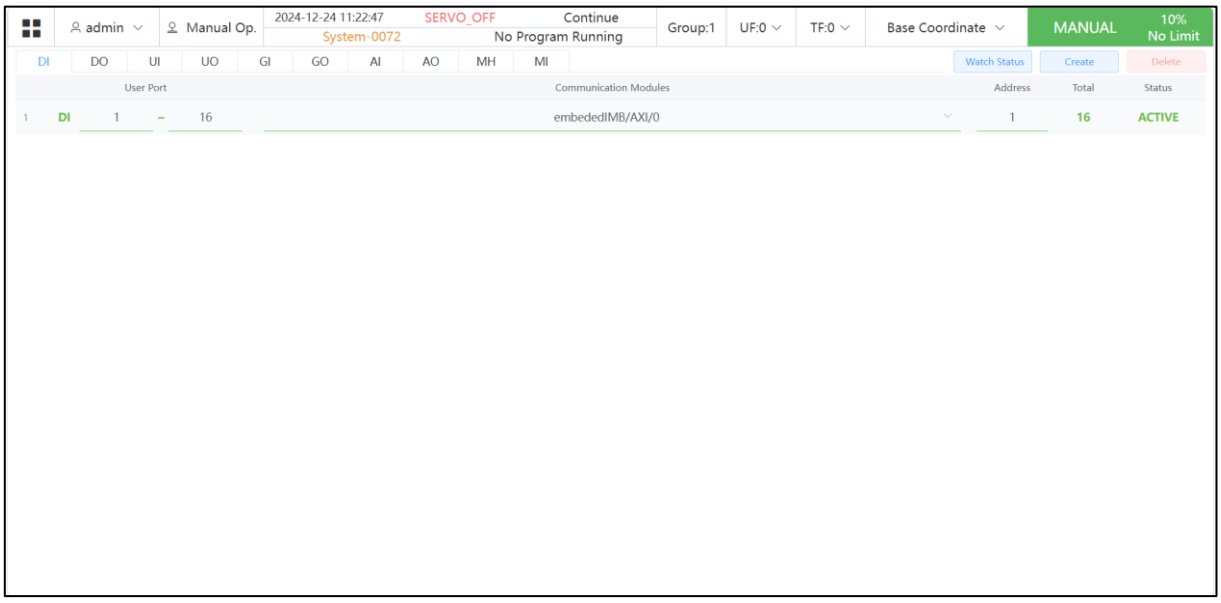


Fig. 2.1 I/O Mapping Interface

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:

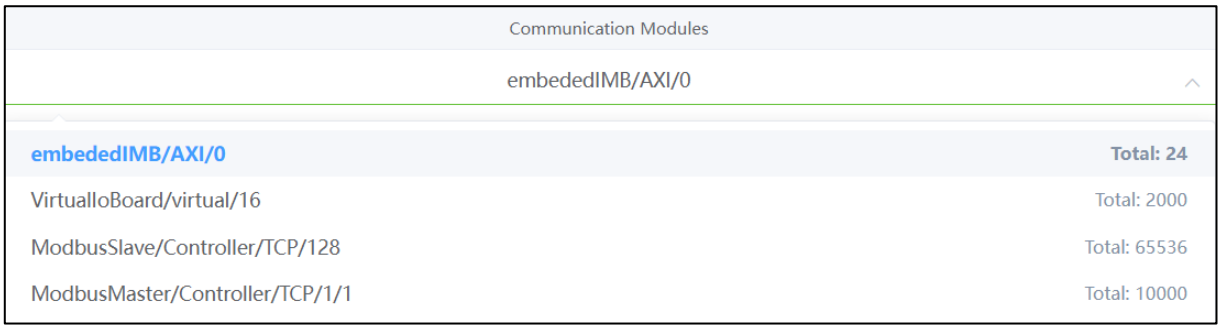


Fig. 2.2 Module Selection Interface

- Module 1 is an I/O adapter for the controller of a 6-axis robot.
- 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 I/O (GIGO) is a general digital signal used to aggregate multiple signal lines and conduct data exchange. The value of a group signal is expressed by a numerical value (decimal number). After being converted or inversely converted into a binary number, data is exchanged through the signal lines.

Mapping (Configuration) of I/O

Group I/O can define signal numbers as a group. It is possible to define 2 to 16 signals as a 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 software; or after pausing, it turns off the servo-holding brake to make a complete stop)	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 following processing: 1. Issue an alarm and then disconnect the servo power supply. 2. Instantly stop the robot (Cat. 0 stop) and suspend the execution of the program. 3. The servo cannot always be enabled. The bypass is ON.		UO[1]	CMDENBLE Allow peripheral devices to control the status signals of the robot.	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 switch 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.		UO[2]	Paused	"Paused" status signal. When the program execution status is "Paused", this signal is ON (i.e. the robot is paused).

List of UI/UO signals						
		The bypass is ON.				
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]	FAULT	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 Control Handle. 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 manually or automatically execute programs. The bypass is ON.		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”.

List of UI/UO signals						
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 "SMPLCS Simple Mode".
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 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.		UO[8]-UO[13]	Selection Confirm 1-6	It is only valid when the "Program Launch Mode" is set to "MPLCS" or "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.
UI[14]	Drag mode Signal	When the input level is high, the robot enters a dragable state. If the robot is in a state where dragging is not allowed, the input is ignored.				

2.1.4 Tool I/O

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

For the definition and usage of the tool IO interface, please refer to the manual of the collaborative robot body for details.

> **Warning**
>
> This equipment does not support the hot-swapping function. That is to say, when the equipment is in the powered-on and running state, it is prohibited to directly plug or unplug the end tool. If hot-swapping is forcibly carried out, it may lead to adverse consequences such as equipment failure, data loss, and interface damage, which will seriously affect the normal use and service life of the equipment. Therefore, when customers plug or unplug the end tool, they must turn off the power supply of the equipment first. After the equipment has completely stopped running, then carry out the plugging and unplugging operations of the end tool.

Tool I/O Configuration

Menu → Communication → I/O Configuration, and enter the tool I/O configuration interface, as shown in the figure below.

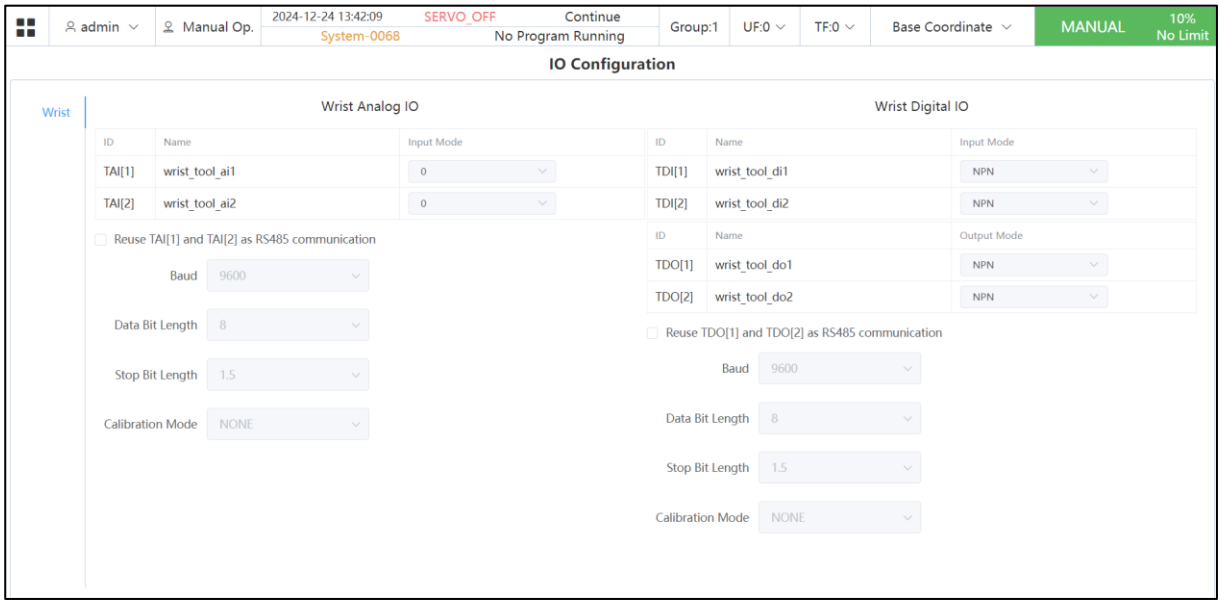


Fig. 2.3Tool I/O Configuration

Digital I/O

Digital Power Switching: Since the digital output interface and the dual-pin power mode are shared, if the dual-pin mode is selected, Digital 0 and Digital 1 will be directly disabled and only used for power supply.

Digital Input and Output Switching: In the single-pin mode, drop-down boxes can be set on the right side of Digital 0, Digital 1, Digital 2, and Digital 3 to select the input and output modes, and then another set of drop-down boxes on the right side can be used to select PNP and NPN (the push-pull function is reserved in hardware, and the software is not developed for the time being).

Analog I/O

Analog Input Switching: Select the mode of the tool's analog I/O. When "Analog Input" is chosen in the drop-down box, there can be drop-down boxes on the right side of Analog Quantity 0 and 1 to select between the current type and the voltage type.

Tool Communication

The two analog quantities can be multiplexed with the RS485 bus. Therefore, when the mode in the drop-down box of the tool's analog I/O is set to "Communication Configuration", the default configuration is the RS485 bus mode. Users can select and set values such as baud rate, parity check, and stop bits from the drop-down menu of the communication interface. Any change in the values will be immediately sent to the tool. If the values used are different from those of the tool, a warning will appear. The RS485 supports the use of Modbus RTU master and slave stations.

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.

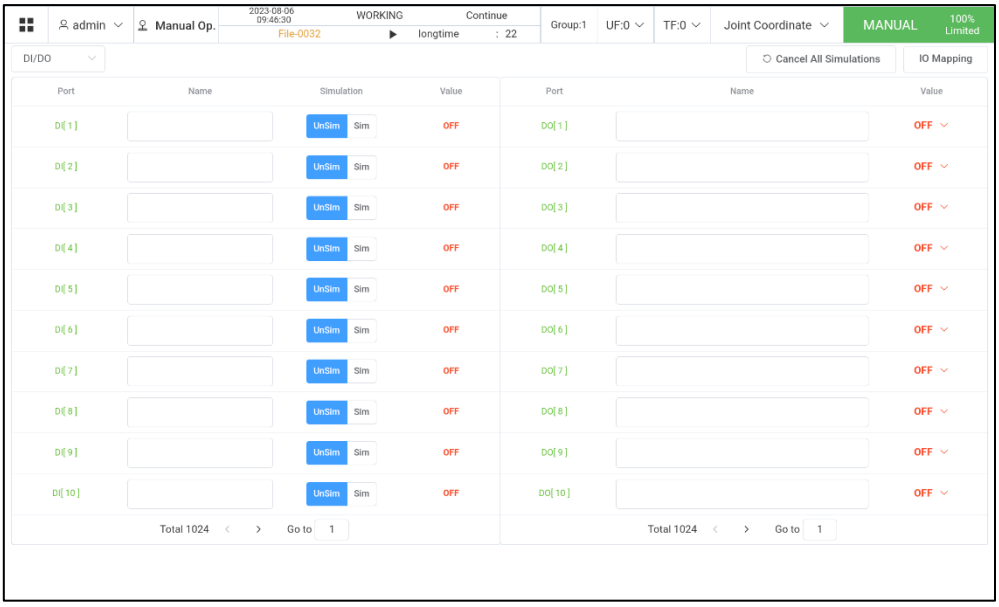


Fig. 2.4 I/O Status Screen

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.

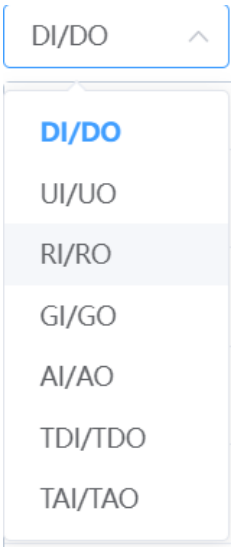


Fig. 2.5 I/O Type Switching Window

Forced output

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

Click the icon in the green 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.

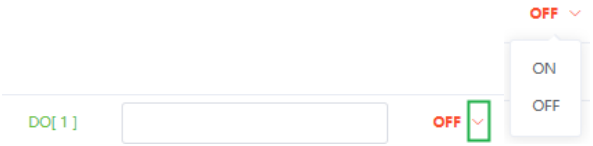


Fig. 2.6 DO Forced Output Window

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.

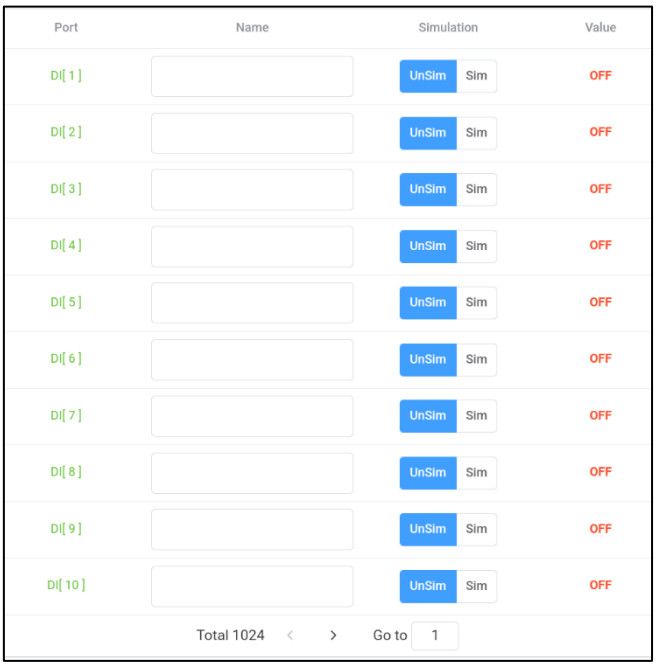


Fig. 2.7 DI Simulation Window

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. 2.8 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.

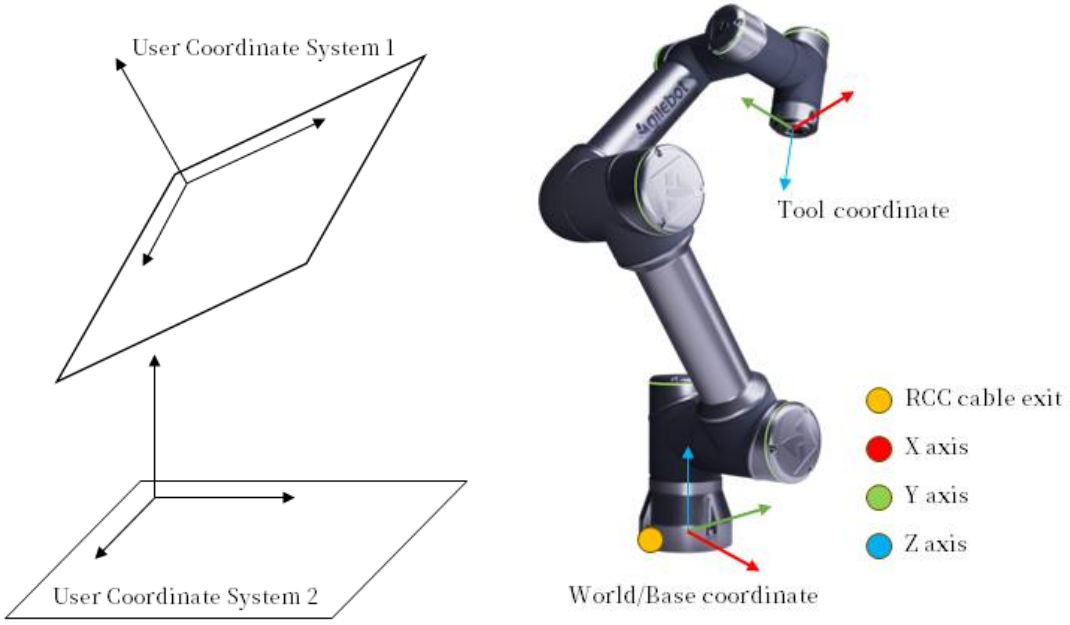


Fig. 2.9 Coordinate Systems of Robots

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(i.e. TF)

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(i.e. UF)

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 six-axis 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

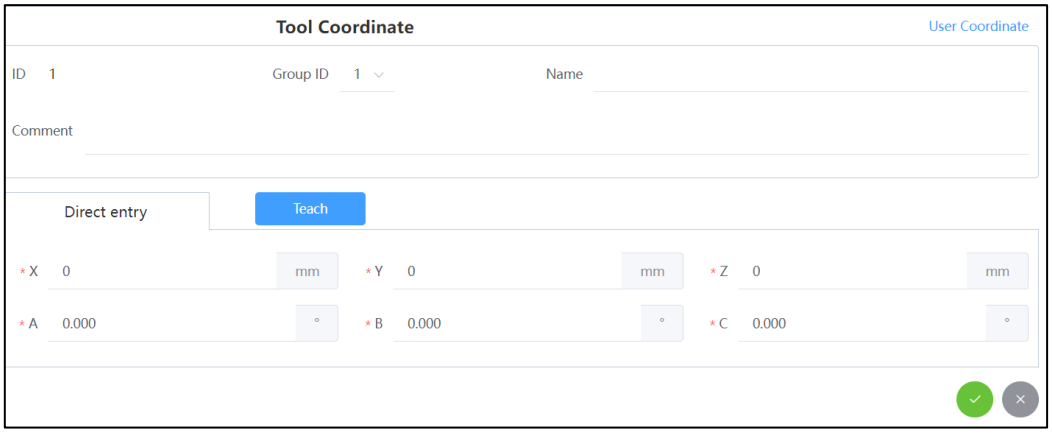


Fig. 2.10 Tool Coordinate System Setting Window

- 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.

Seven-point setting of tool coordinate system for robot (the four-point method is the same, but P1-P4 are used)

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.

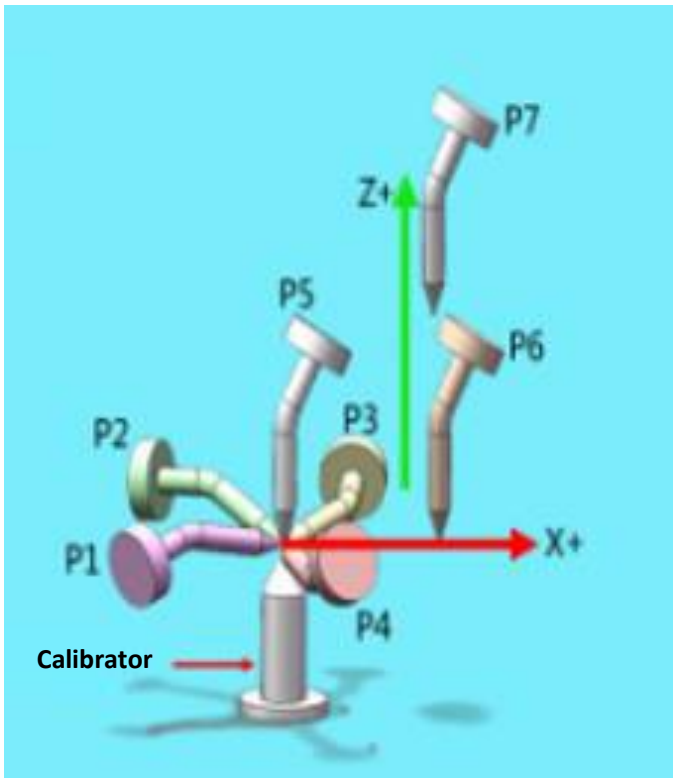


Fig. 2.11 7P Setting of Tool Coordinate Poses

Specific steps:

1. Click "Menu Button" → "Coordinate System" → "Tool Coordinate System" to enter the tool coordinate system setting interface, as shown in Fig. 2.12.

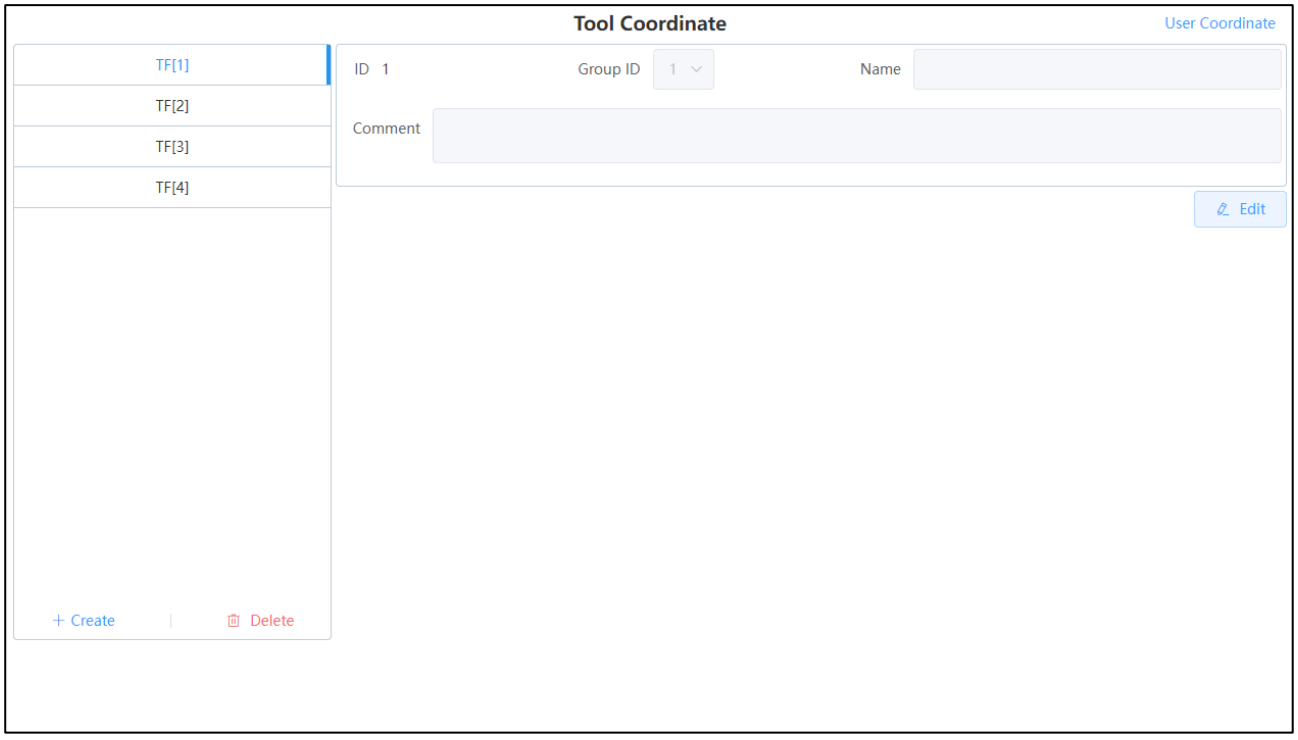


Fig. 2.12 Tool Coordinate System Setting Interface 1

2. Click "Edit" the interface is shown in Fig. 2.13.

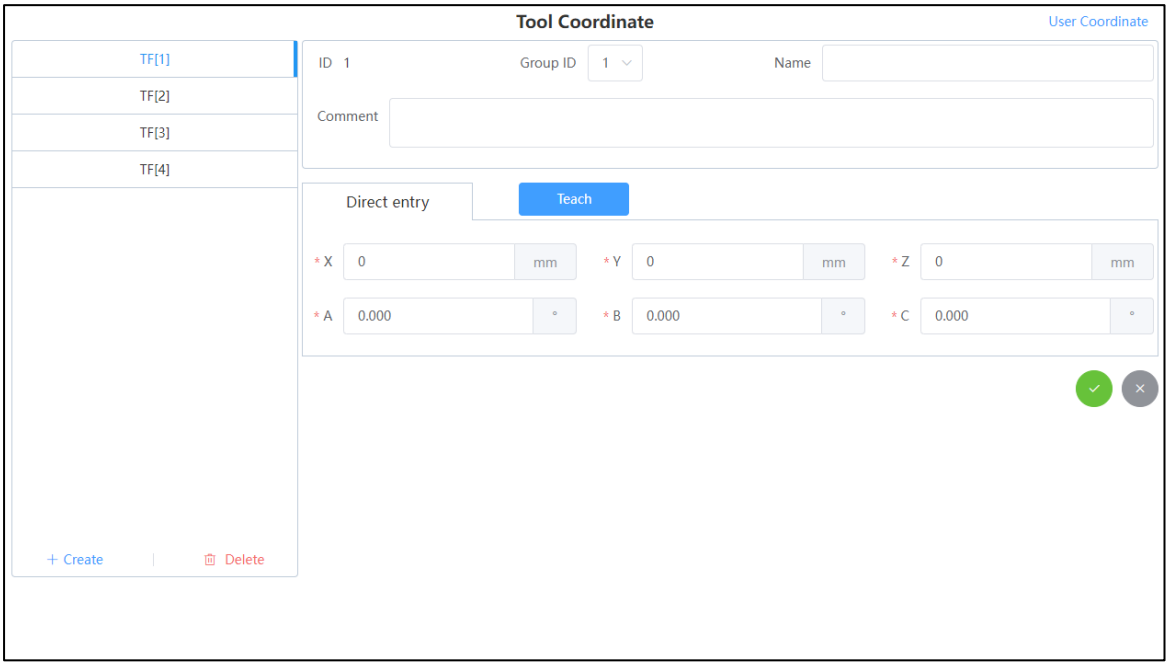


Fig. 2.13 Tool Coordinate System Setting Interface 2

2. Click "Teach" the interface is shown in Fig. 2.14.

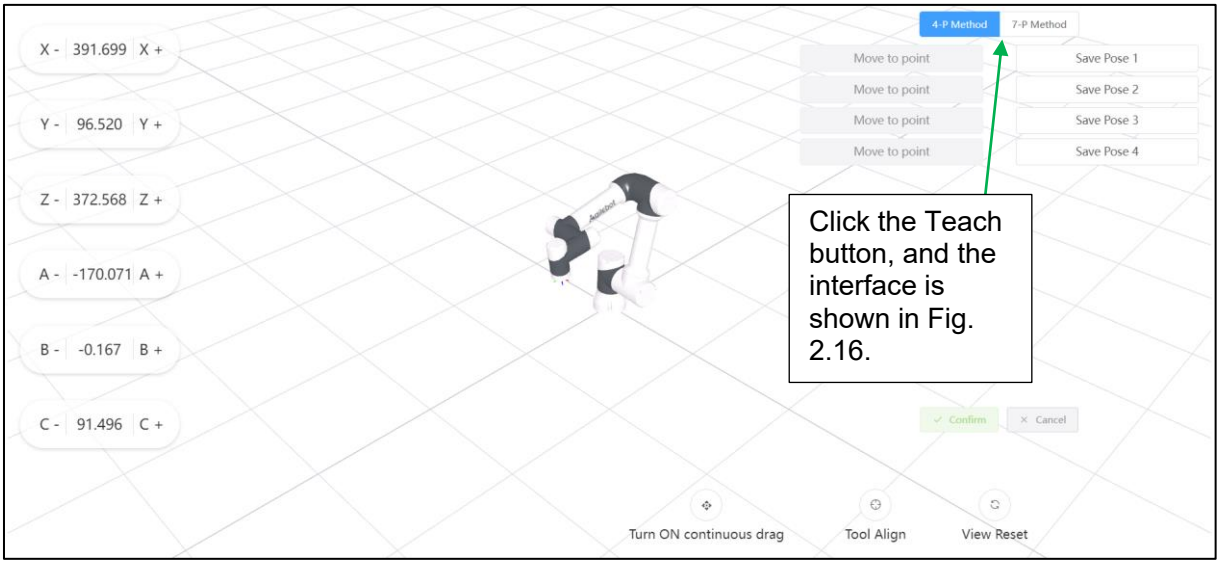


Fig. 2.14 Tool Coordinate System Setting Interface 3

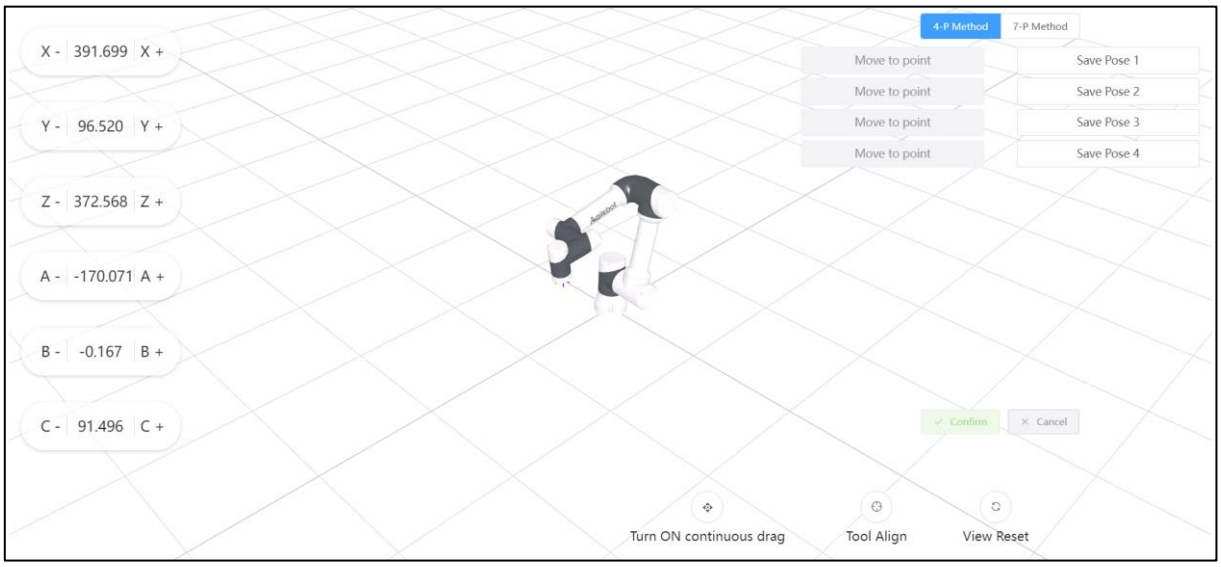


Fig. 2.15 Tool Coordinate System Setting Interface 4

4. Select the pose to be taught. After teaching of that pose, click "Save Pose", and the interface is shown in Fig. 2.16.

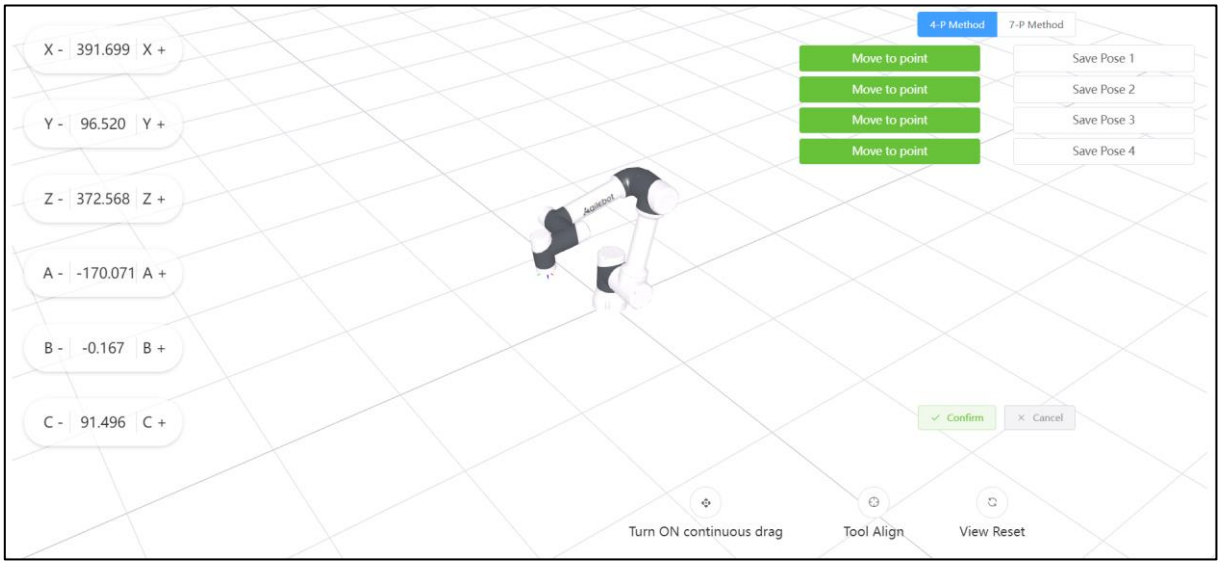


Fig. 2.16 Tool Coordinate System Setting Interface 5

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

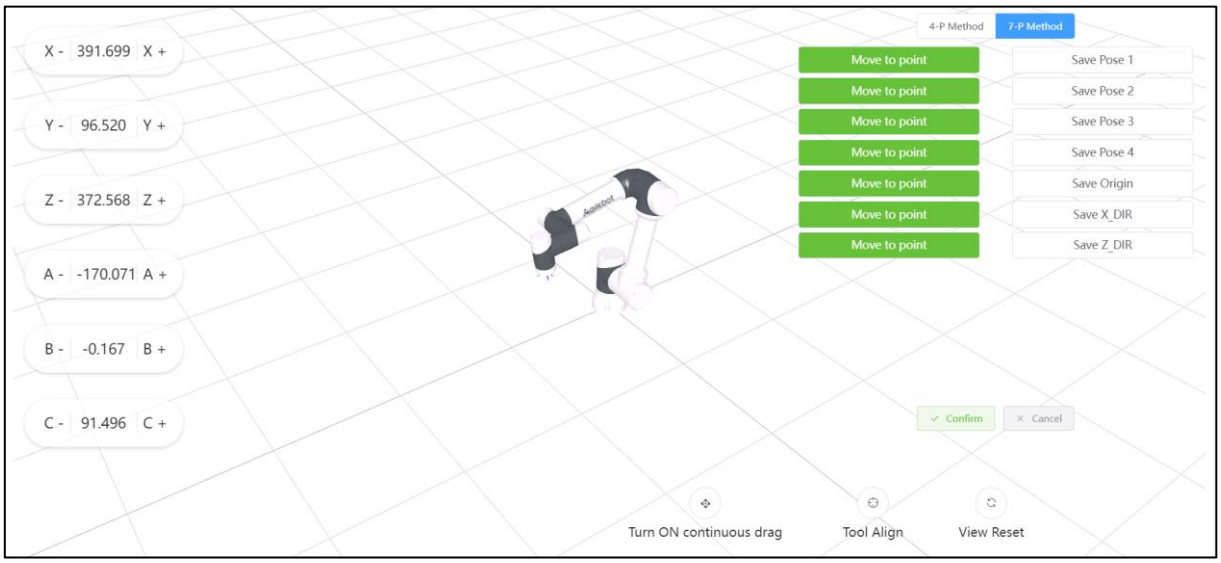


Fig. 2.17Tool Coordinate System Setting Interface 6

6. After teaching of 7 poses,.Please save after confirmation". At this time, click "Confirm"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.

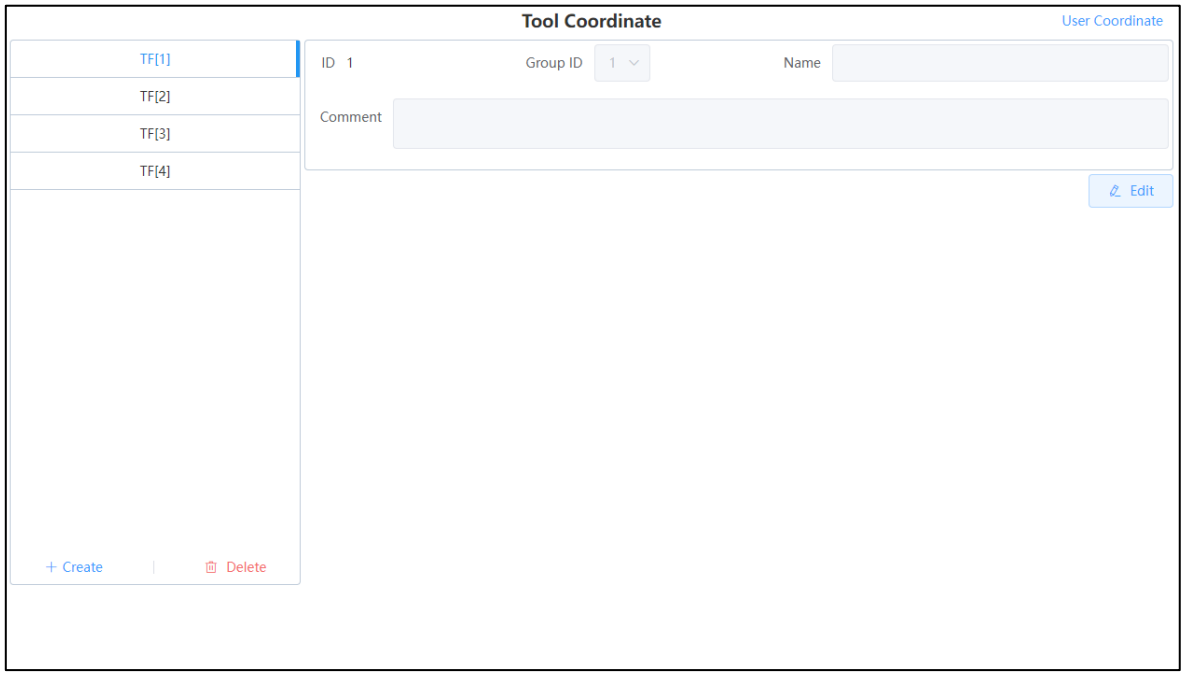


Fig. 2.18 Tool Coordinate System Setting Interface 7

4P setting of tool coordinate system for 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.

Setting of tool coordinate system by direct write method

As shown in Fig. 2.19, select the direct write method to write the tool coordinate data X, Y, Z, A, B, C.

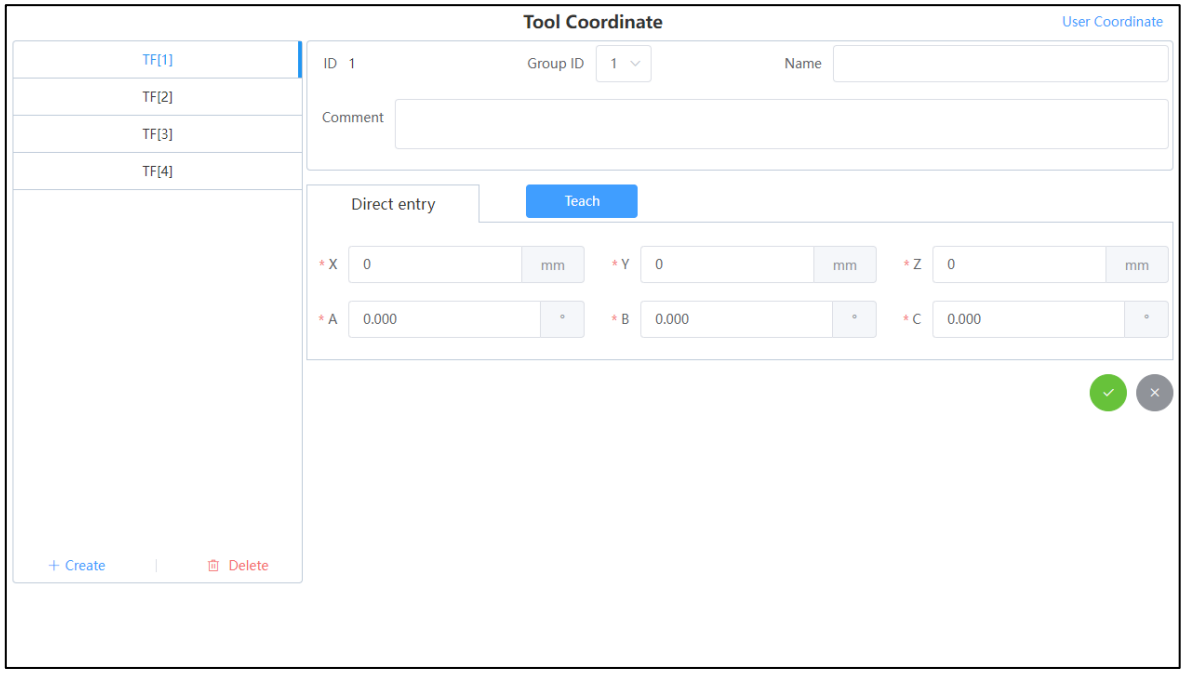


Fig. 2.19 Window of Direct Writing Method

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.

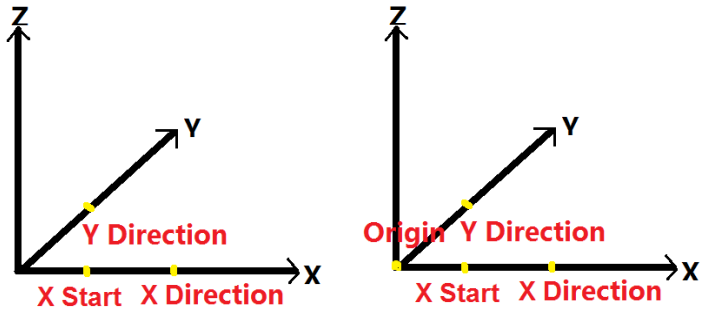


Fig. 2.20 3P Method Fig. 2.215 4P Method

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. 2.22, and Choose 3p/4p as shown in the figure below.

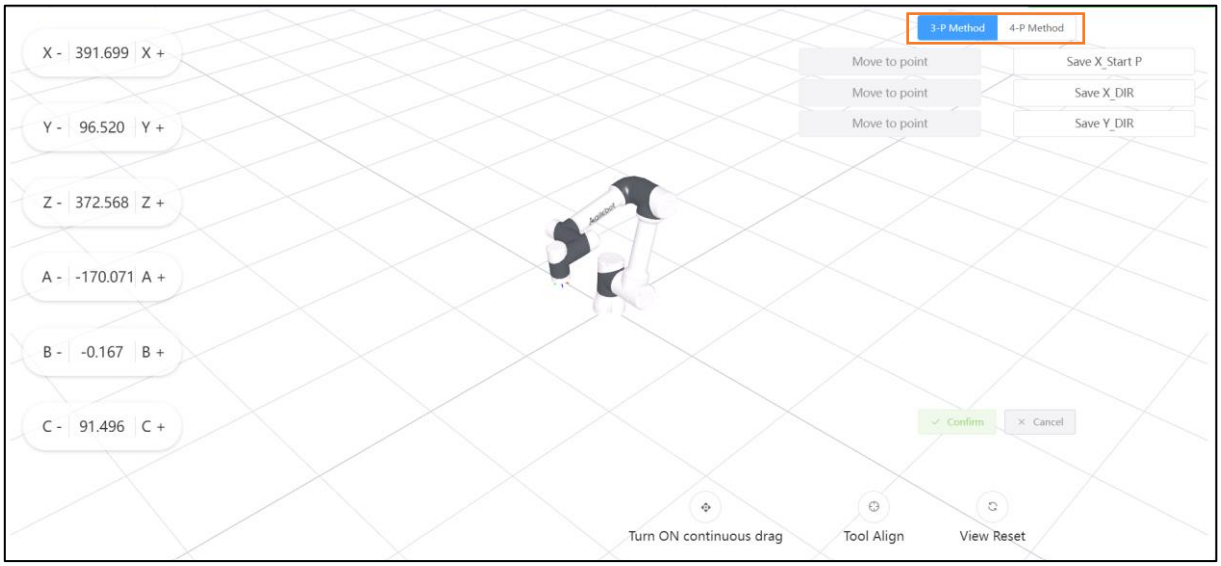


Fig. 2.22 User Coordinate System Setting Window

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. 2.23 “3P Method”/2.24 “4P Method”.

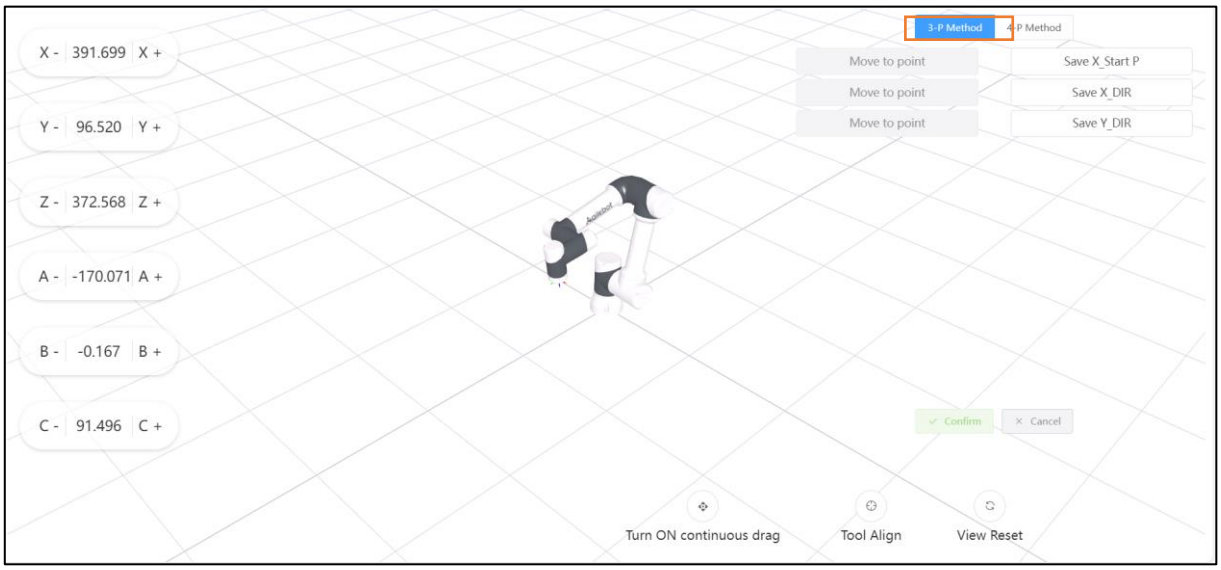


Fig. 2.23 3P Setting Window

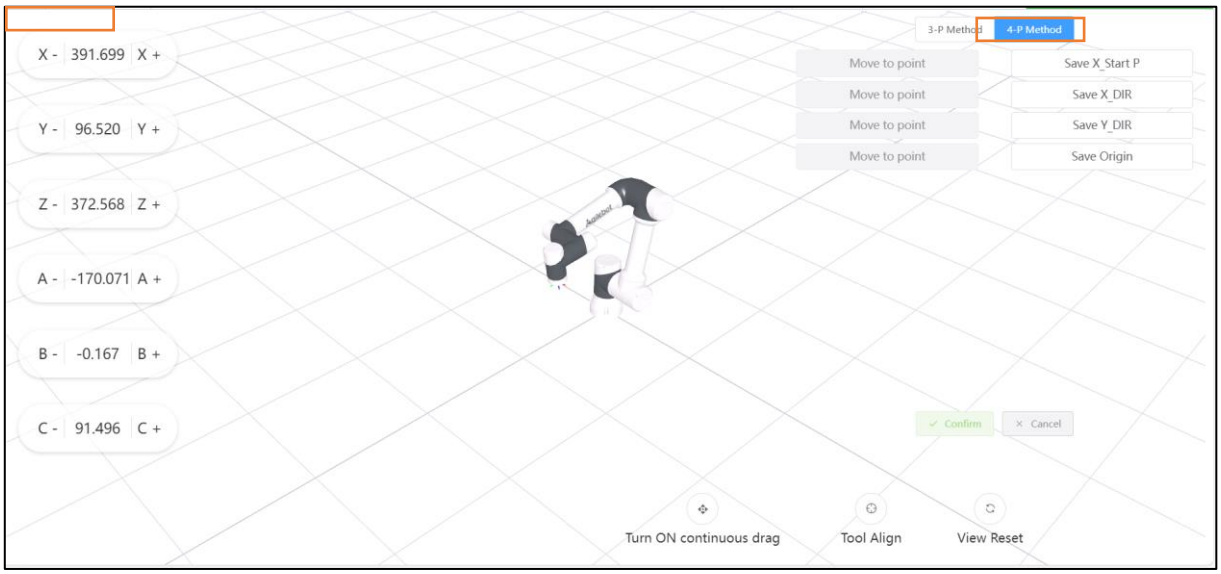


Fig. 2. 24 4P Setting Window

Direct write method:

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

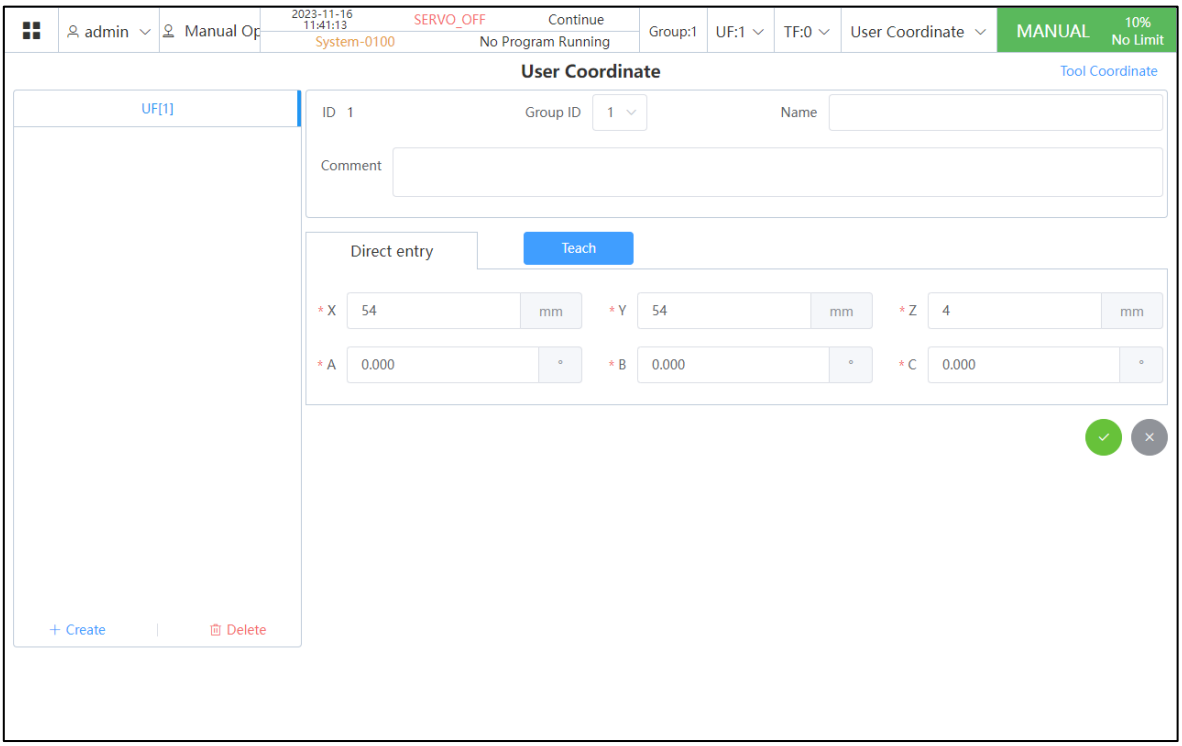


Fig. 2.295 Setting Window of Direct Writing Method

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.

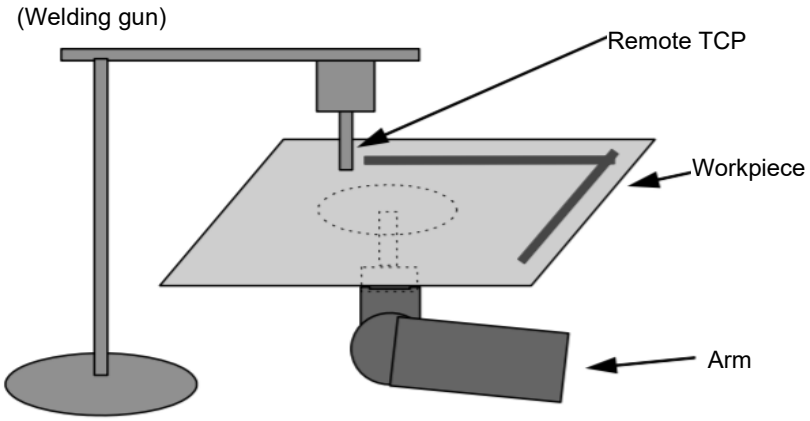


Fig. 2.26 Design Diagram of Remote TCP

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.

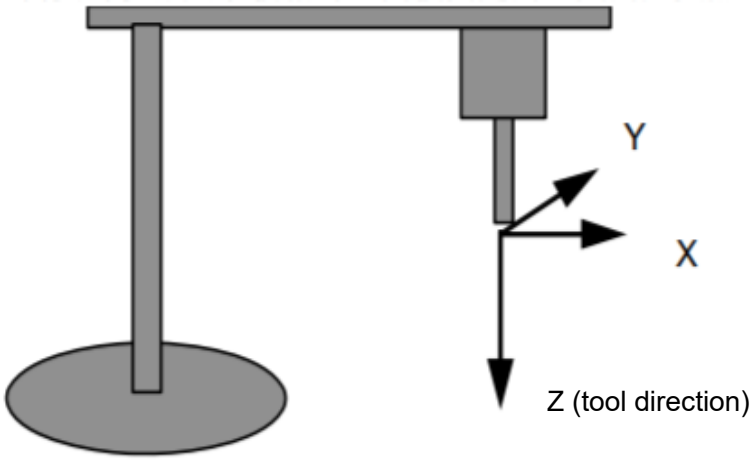


Fig. 2.27 Schematic Diagram of Remote TCP Coordinate System

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.

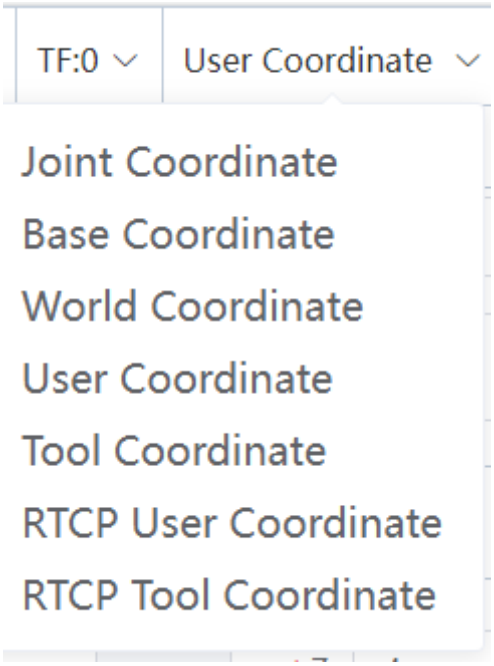


Fig. 2.28 Reference Coordinate System Menu

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. 2.29 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.

Caution

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. 2.30.

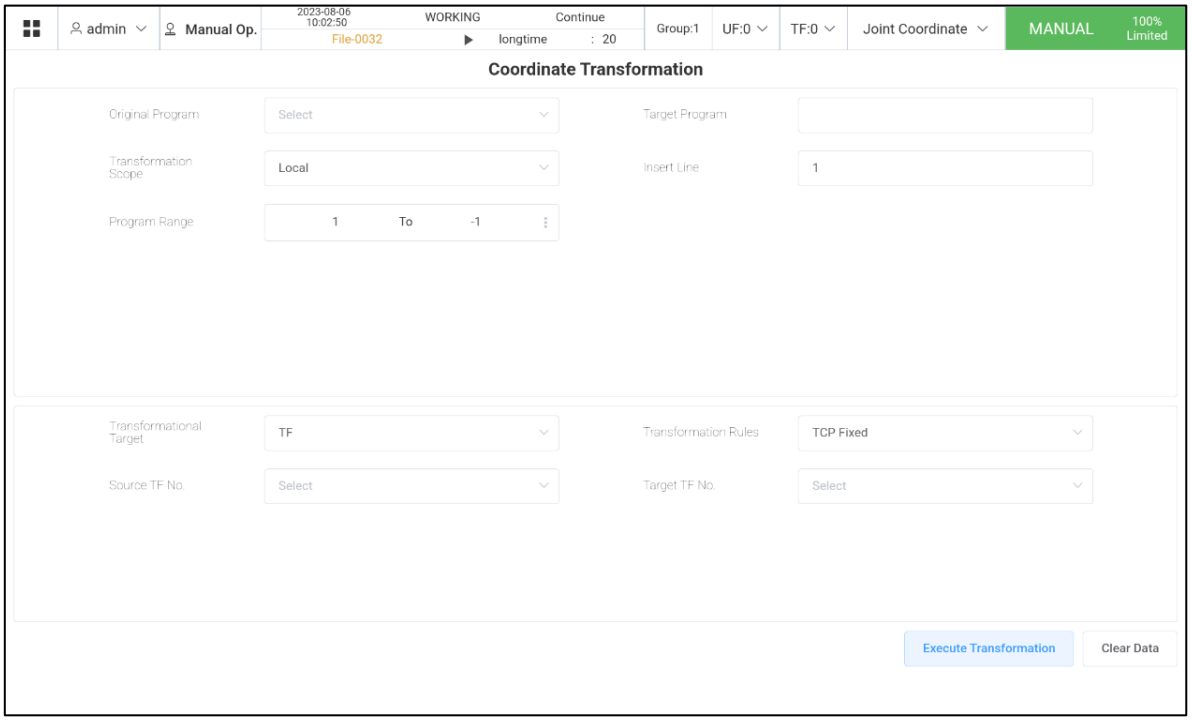


Fig. 2.30 Coordinate Transformation Interface

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.

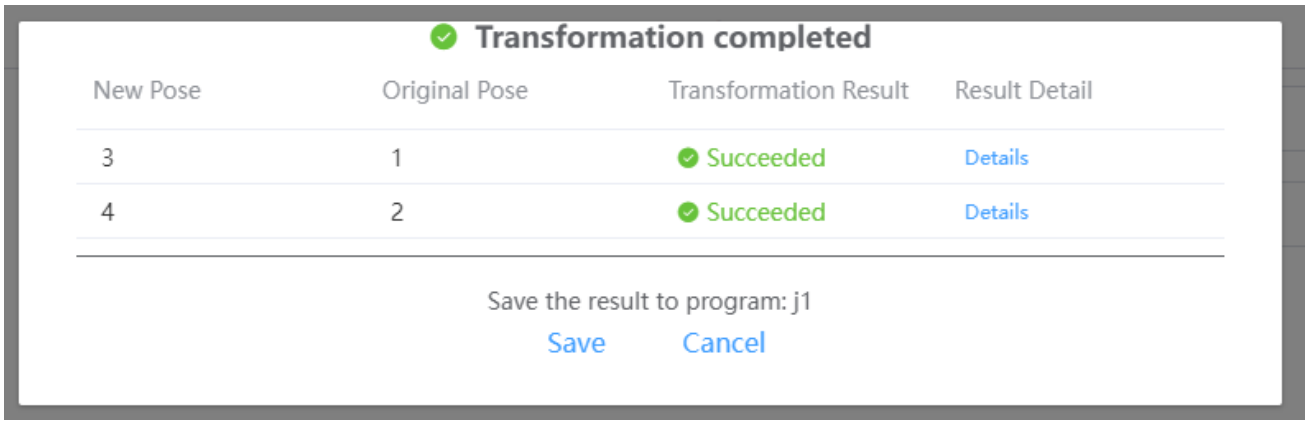


Fig. 2.31 Transformation Completion Window

3) Click “Save” to complete the transformation.

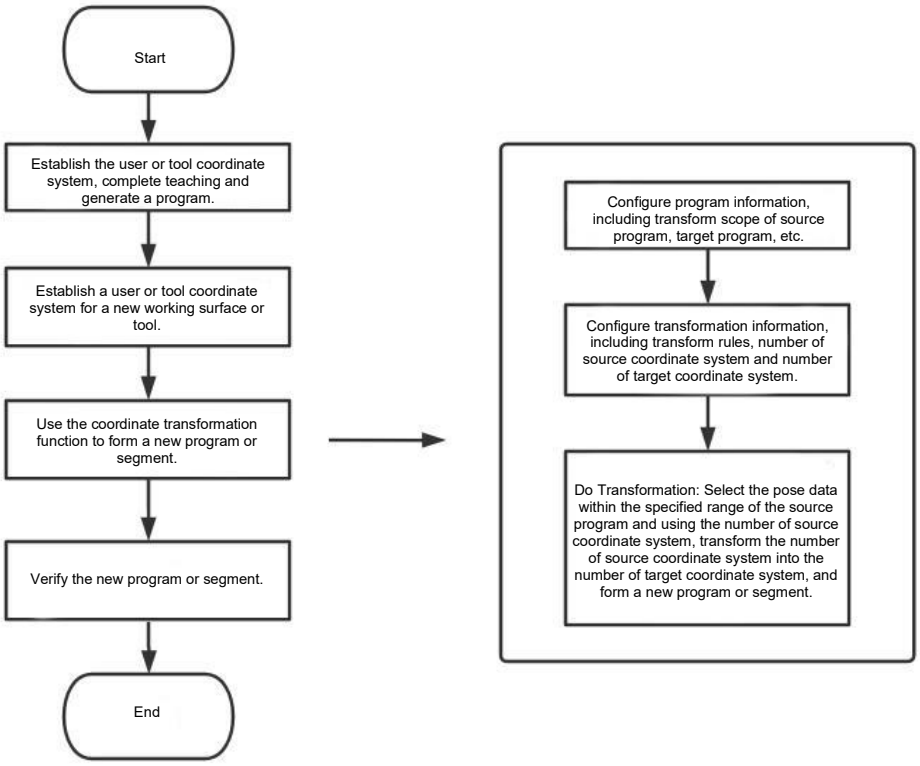


Fig. 2.32 Coordinate Transformation Flowchart

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 and the user is allowed to set the joint range of the soft limit according to the situation.

> 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 operating terminal. 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. 2.33.

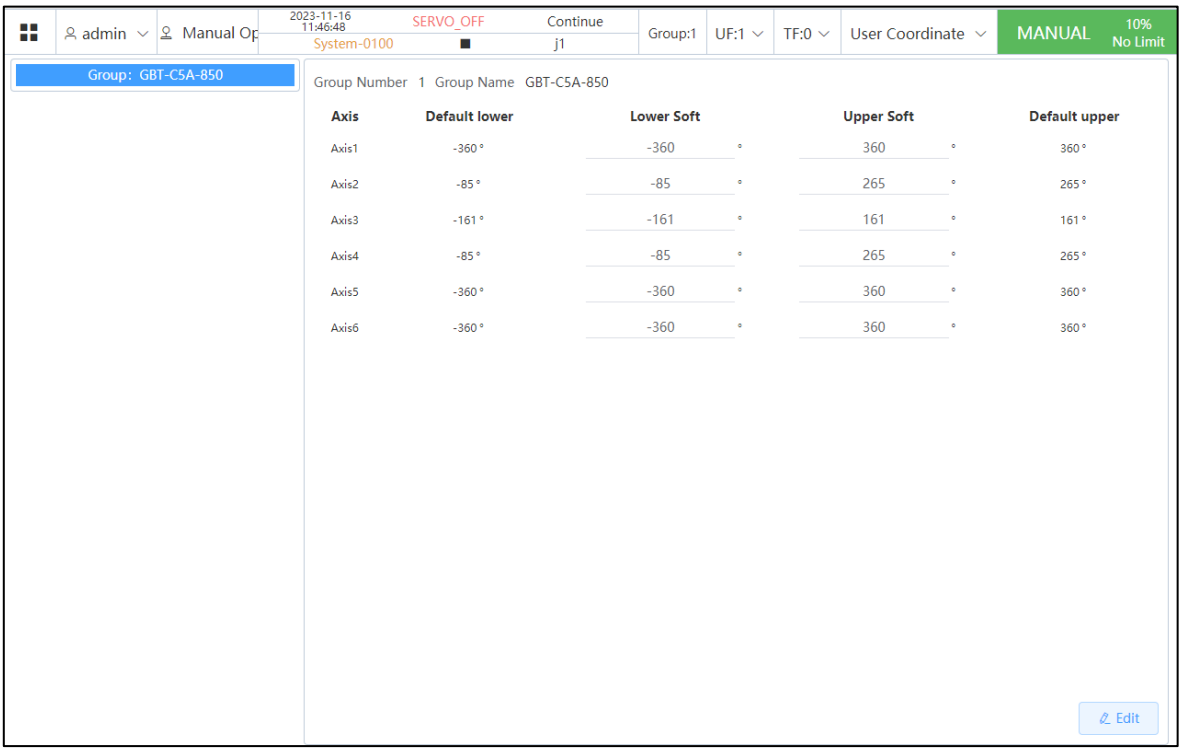


Fig. 2.33 Soft limit Setting interface

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 performance. (reduced 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. 2.34. The payload activated at default is set to [Payload: 0] and cannot be edited.

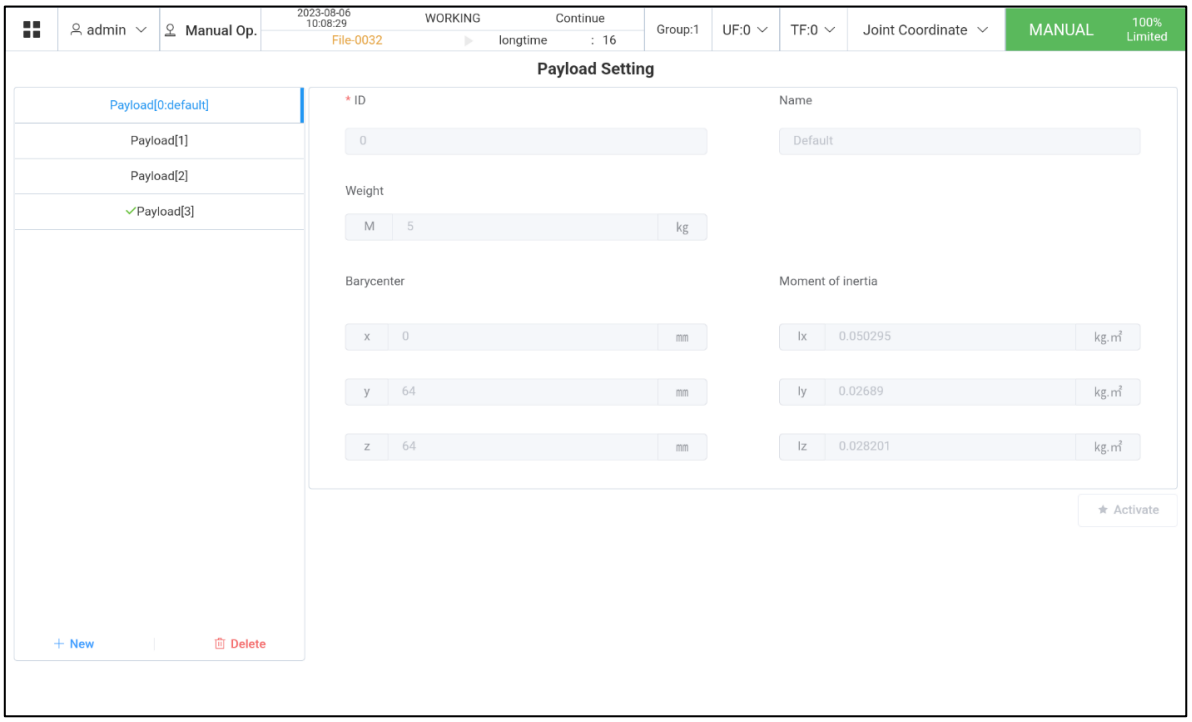


Fig. 2.34 Payload Parameter Interface

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

Fig. 2.35. - Cancel editing, - Save editing.

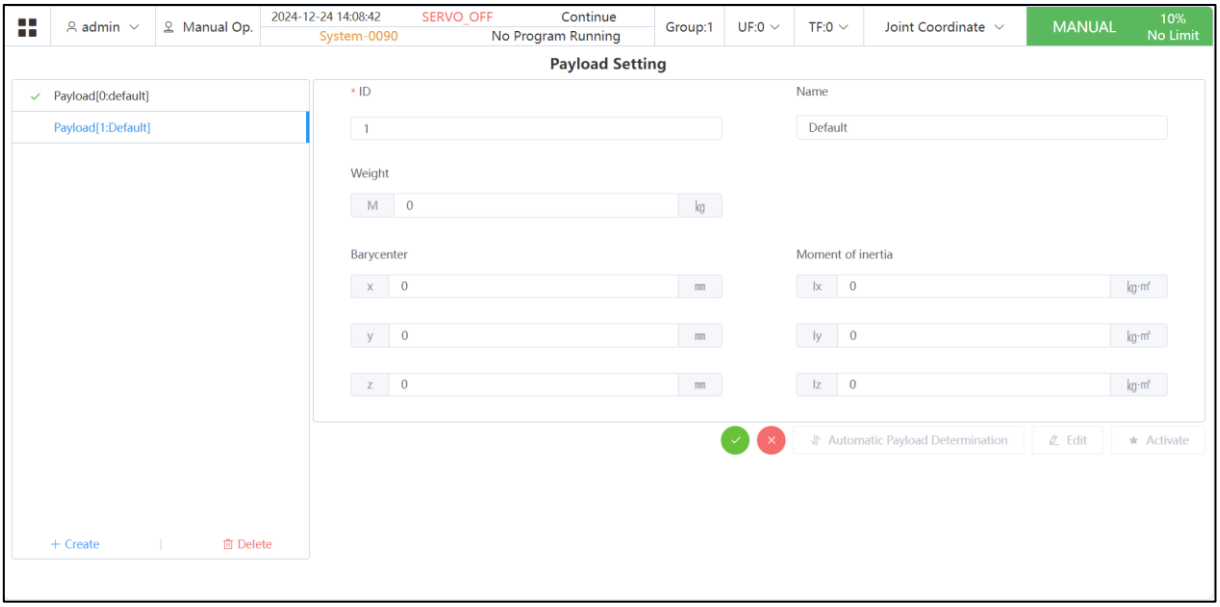


Fig. 2.35 Payload Parameter Interface

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.

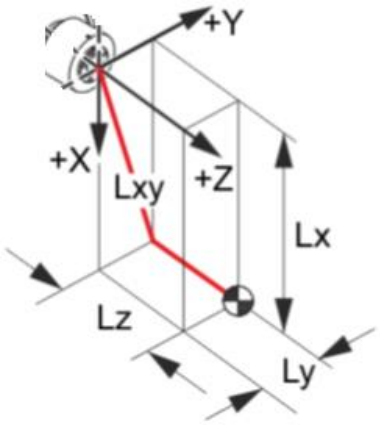


Fig. 2.36Payload Position Reference

3. With a "√" mark in front of the activated load.

2.4.1 Automatically measure the load

The automatic load measurement is an expansion of the original load setting function, and its location is at Load Setting/Automatic Measurement.

> **Caution**
>
> It is recommended that when using the automatic measurement function, the load mass is greater than 20% of the robot's maximum load.

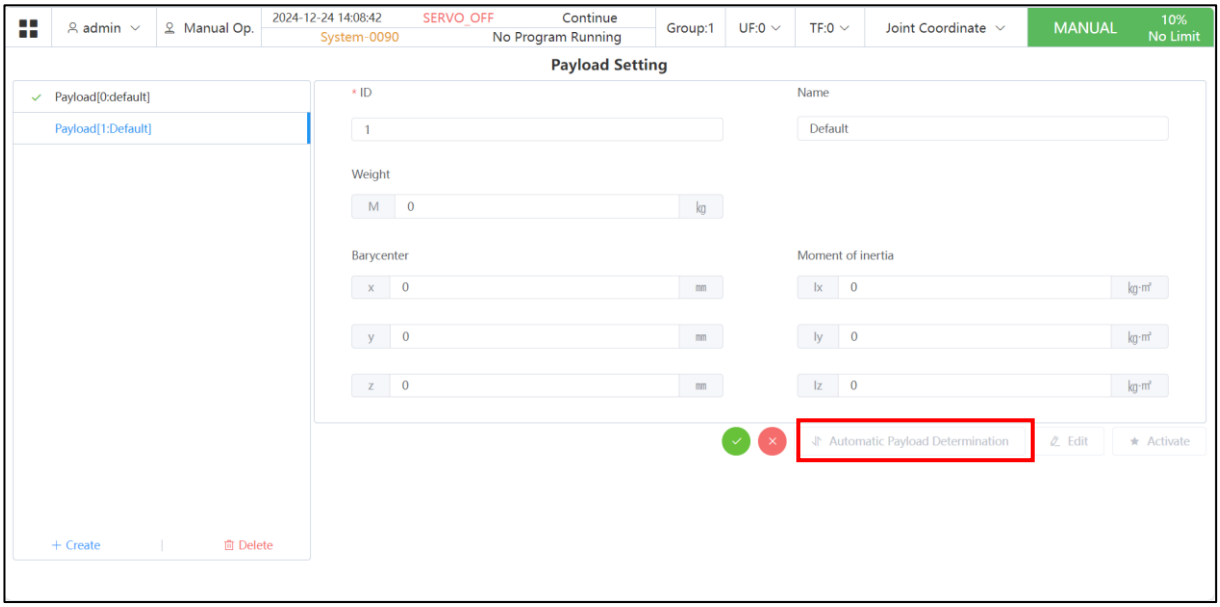


Fig. 2.37 Automatic Payload Determination

The steps are as follows:

1、Set the parameters and the 3rd axis.

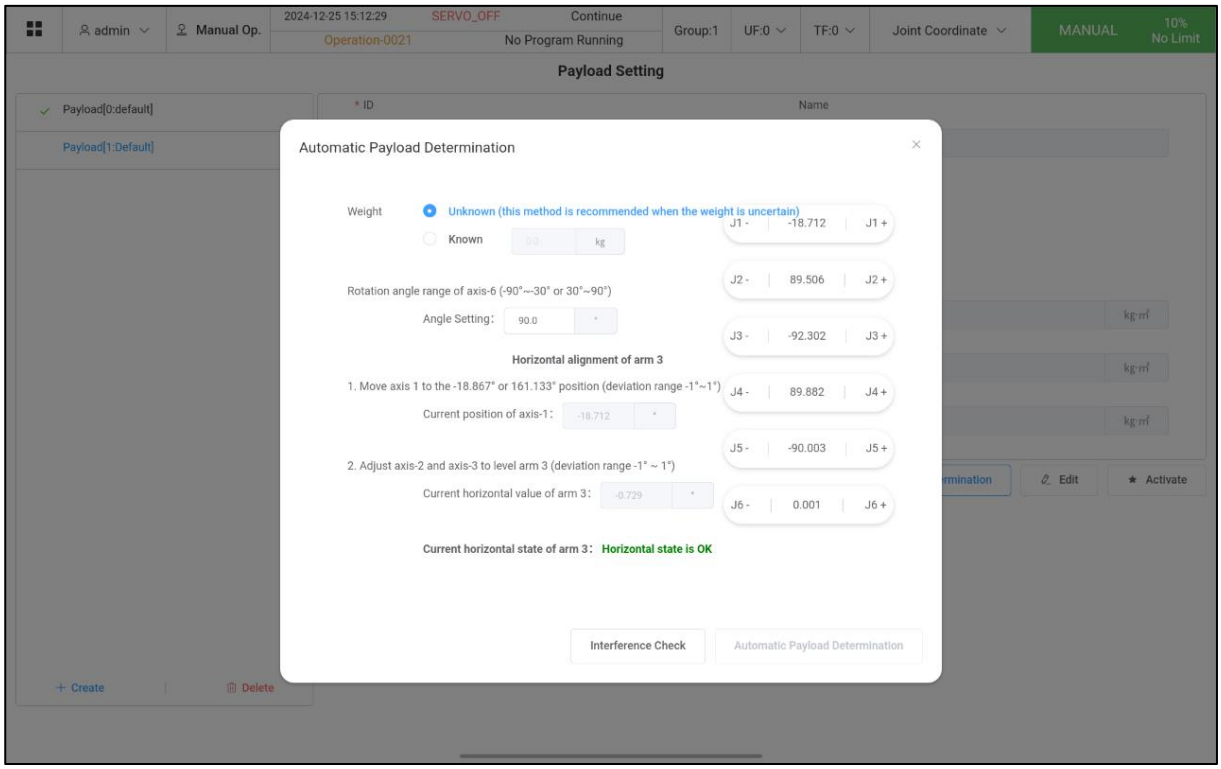


Fig. 2.38 parameters and the 3rd axis

Mass:

Unknown: Select this option when the specific value of the mass is unknown. It is recommended to choose this one in such a case;

Known: Select this option when the exact mass is known. The numerical value should be reserved to one decimal place, and there is no limit on the maximum value.

6-axis Angle:

Setting Range: -90° to -30° or 30° to 90°.

The numerical value should be reserved to one decimal place.

3-axis Horizontal Calibration (Teach the robot in manual mode):

Current Horizontal Value: Display the horizontal state of the forearm in real time. Taking the horizontal state as 0°, use the symbol “-” when it is downward and “+” when it is upward.

Current Horizontal State: When the horizontal value of the forearm is within the deviation range (currently it is -1° to 1°. Actually, a reasonable range will be given according to the results of research and development and debugging), the horizontal state will display

"Horizontal" and the 3-axis horizontal calibration is completed. Otherwise, it will display "Not Horizontal".

If there is an angle installation setting, the angle value will be automatically compensated.

2、Interference check

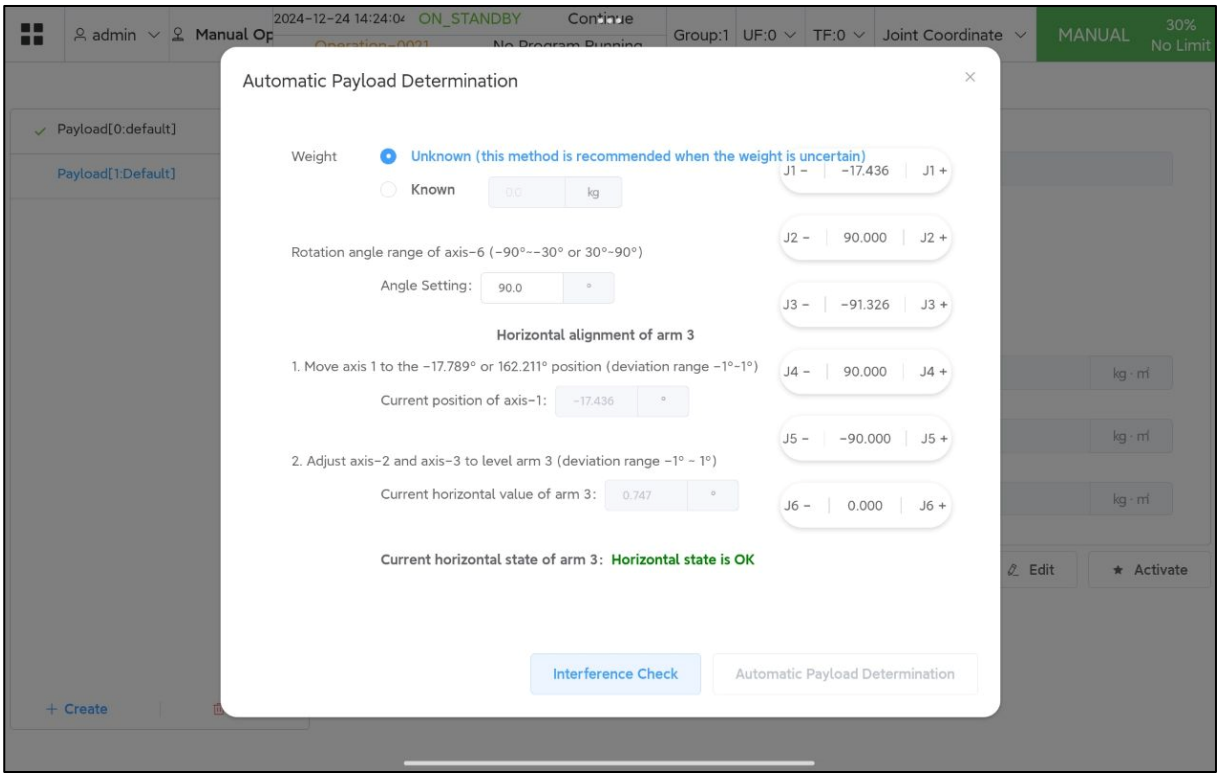


Fig. 2.39 Interference check

1）Interference Check: When the parameter settings are correct, the 3-axis is level, and in the manual mode, if all these conditions are met, the interference check option will be highlighted and can be clicked to proceed with the next operation (see the figure below). Otherwise, the interference check option will be displayed in grayscale and clicking on it will be ineffective.；

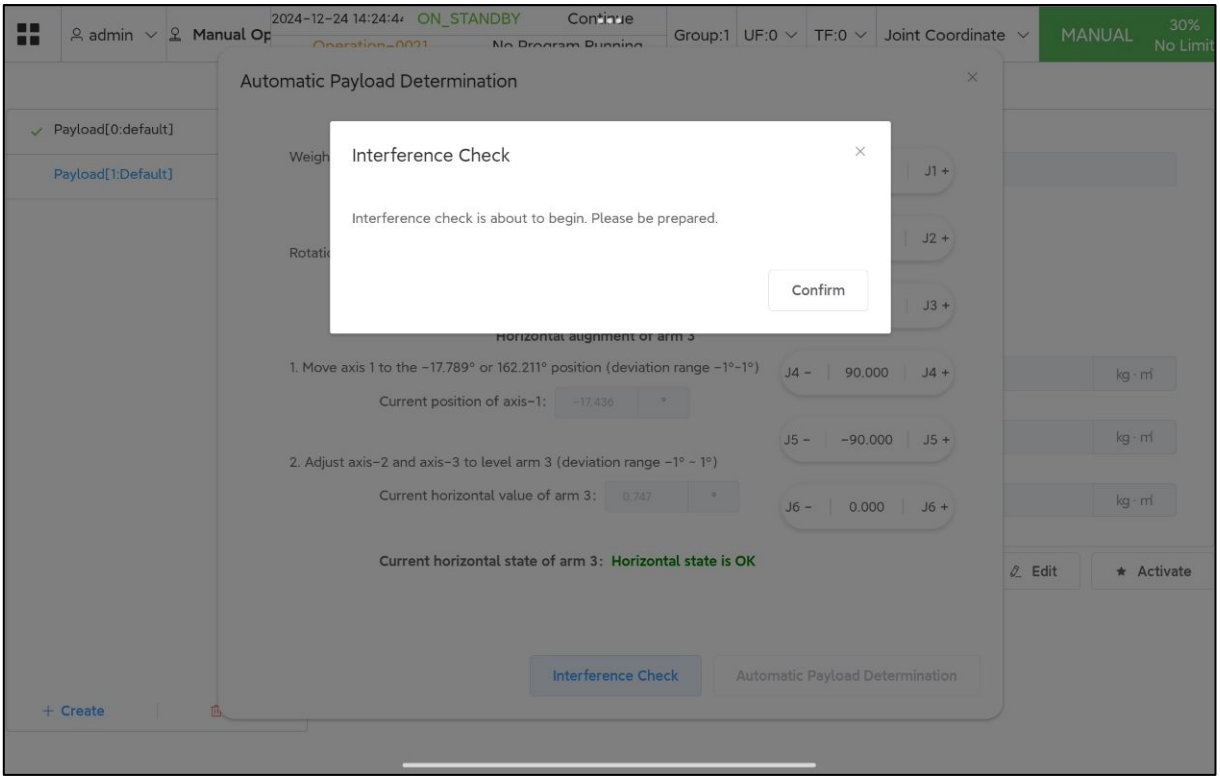


Fig. 2.40 Interference check Caution

3） X: Click it to cancel the ongoing "Interference Check".

3）Confirm: After the conditions for program operation in the manual mode are met, this button will be highlighted (otherwise, it will be displayed in grayscale and cannot be executed). Then click this button, and the robot will automatically conduct the interference check at a speed of 10%.

> ⚠ Caution
>
> The running trajectories of the interference check and the automatic measurement are the same.

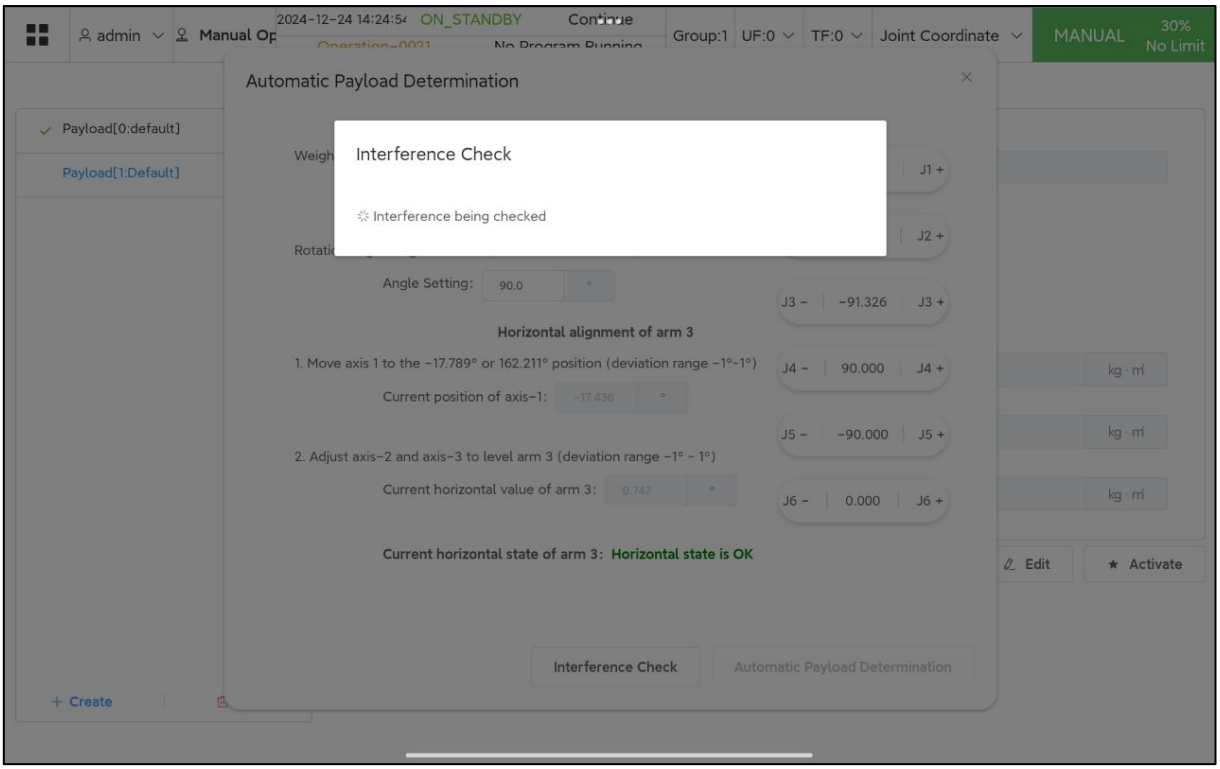


Fig. 2.41 Interference check

1）During the interference check: The robot moves slowly to conduct the interference check. During this process, the interference check can be stopped by releasing the deadman switch or pressing the emergency stop button.

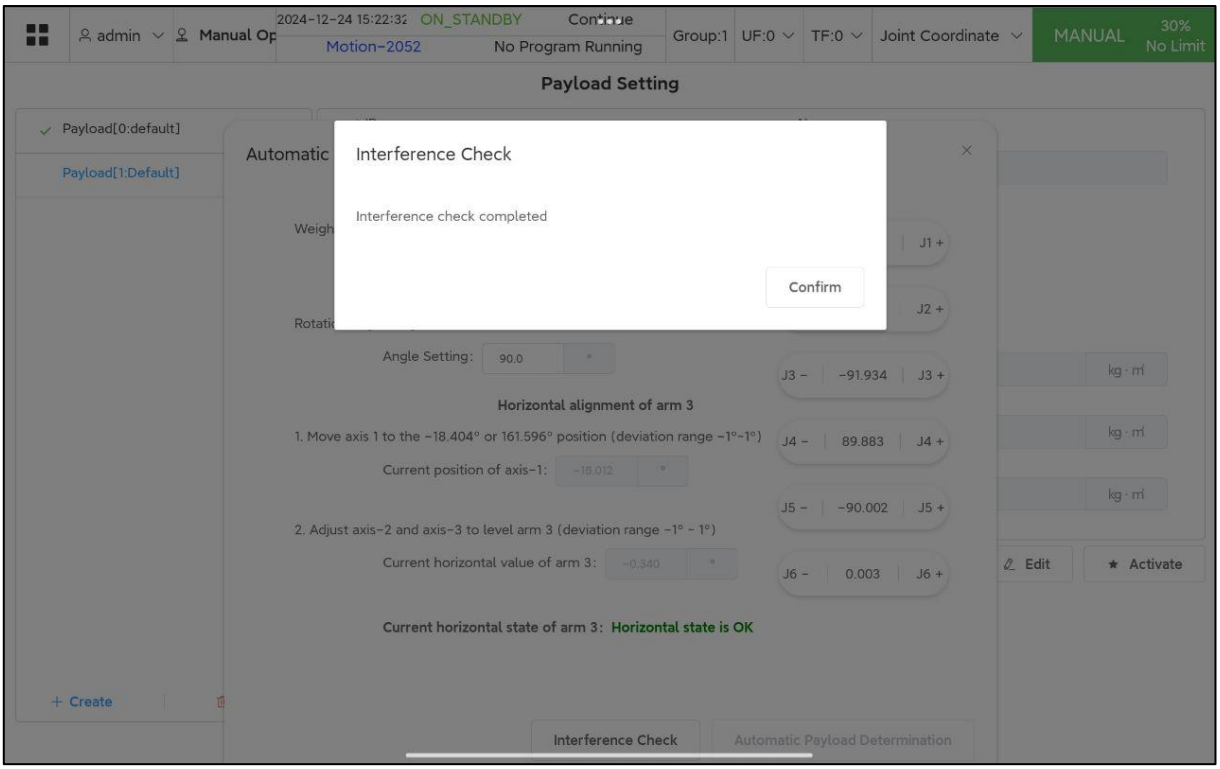


Fig. 2.42 Interference check completed

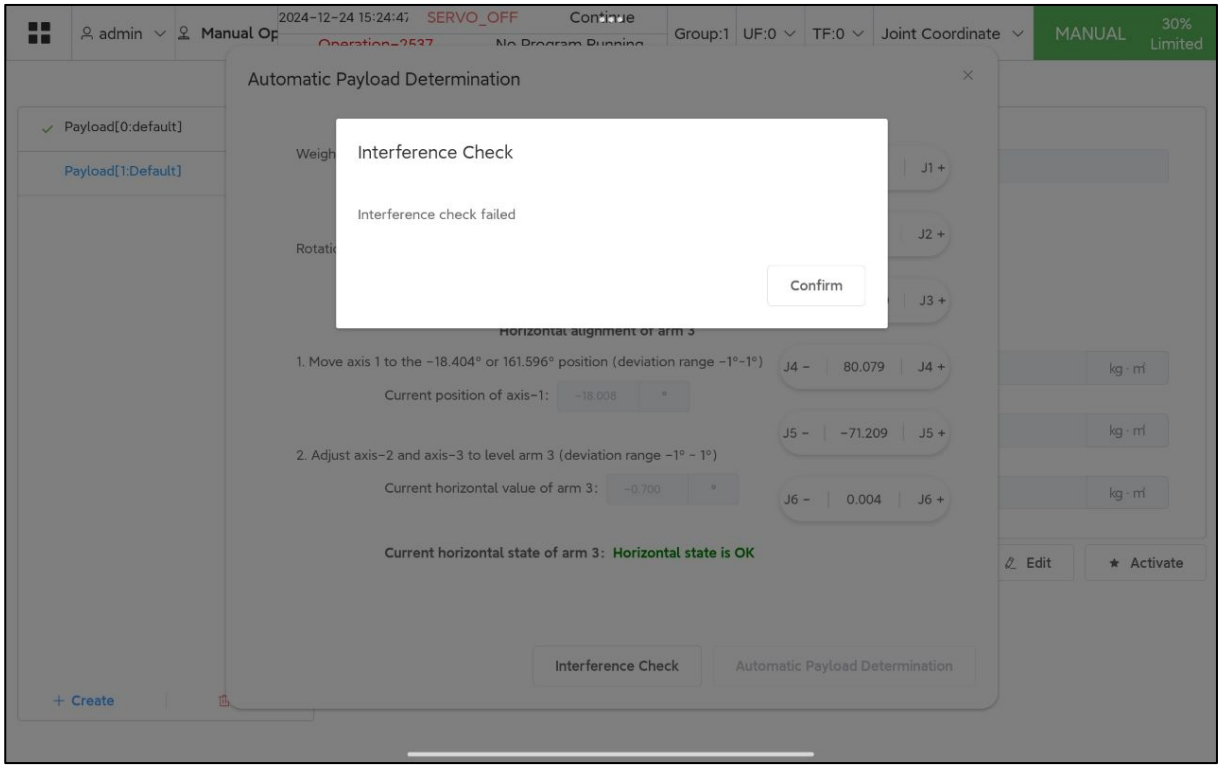


Fig. 2.43 Interference check failed

2）Confirm: The interference check is successful, with no interference. Click it and return to the previous interface.

Any interruption of the inspection actions during the interference check process is regarded as a failure of the interference check. Click it and return to the previous interface.

> ⚠ Caution
>
> When the interference check is completed, the robot will automatically reset to its initial position and posture.

3、Automatic measurement

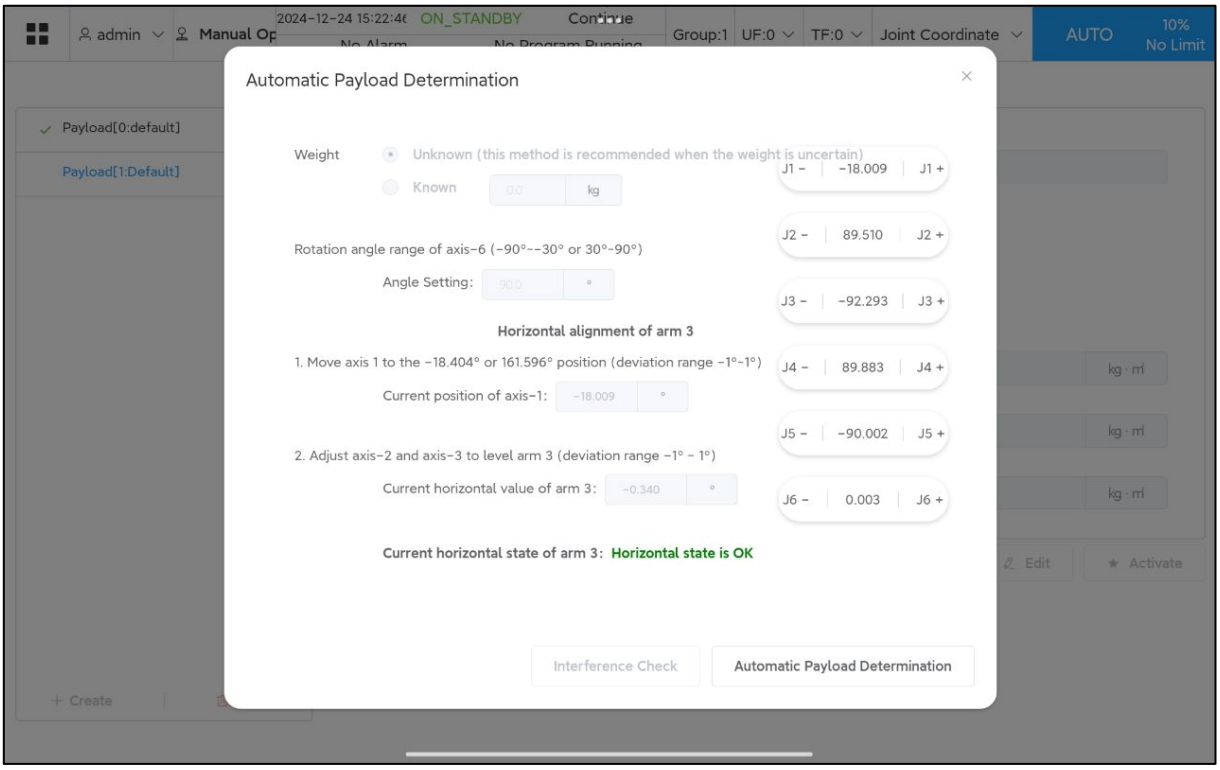


Fig. 2.44 Automatic Payload Determination

1）Automatic measurement: If the most recent interference check is successful and there is no change in parameters (it is sufficient to state that writable parameters remain unchanged and read-only parameters are at the horizontal level), switch to the automatic mode, and the automatic measurement option will be highlighted. Both the interference check and parameter settings will be locked and cannot be operated.

> Caution
>
> After the interference check is successful, if any configuration parameters are modified, the interference check needs to be carried out again. Otherwise, the automatic measurement in the automatic mode will not be highlighted and available either.

2）X: Click it to cancel the ongoing "Automatic Measurement".

3）Confirm: After the conditions for program operation in the automatic mode are met, click this button, and the robot will conduct the automatic load measurement.

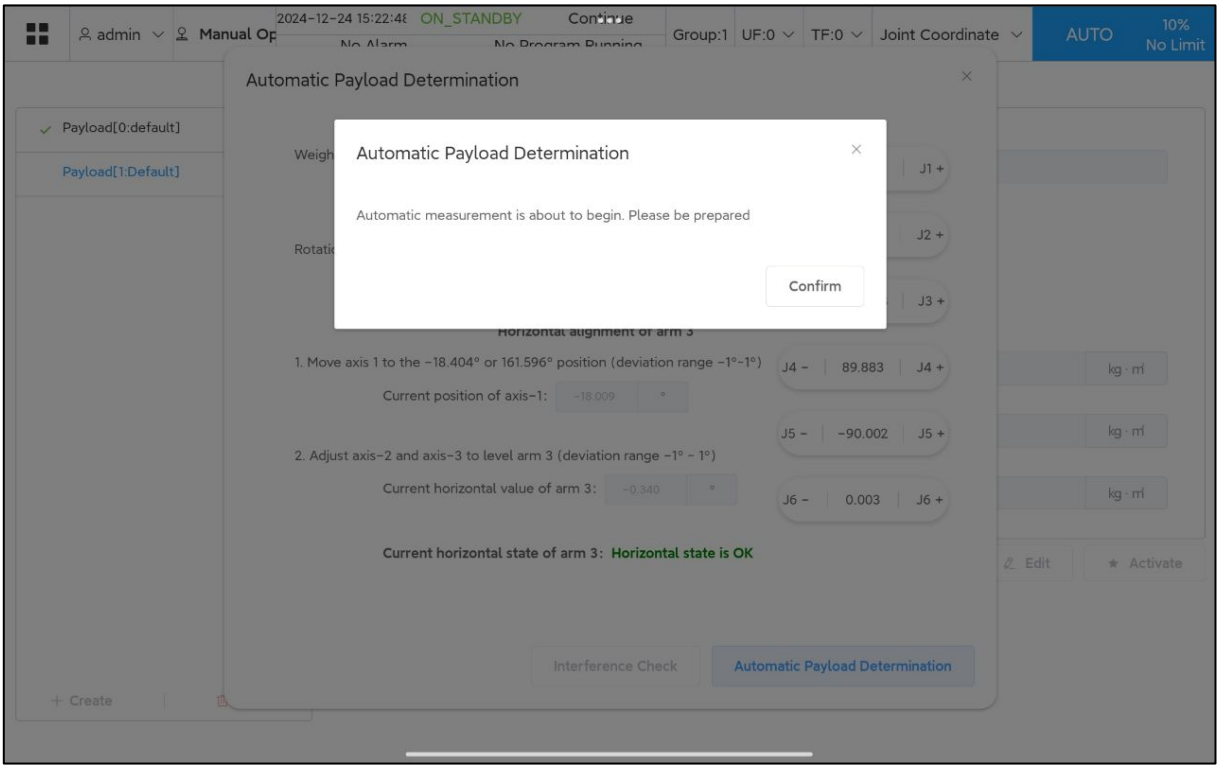


Fig. 2.45 Automatic Payload Determination

1）During the automatic measurement: The movement of the robot can be stopped in a timely manner by pressing the emergency stop button during this process.

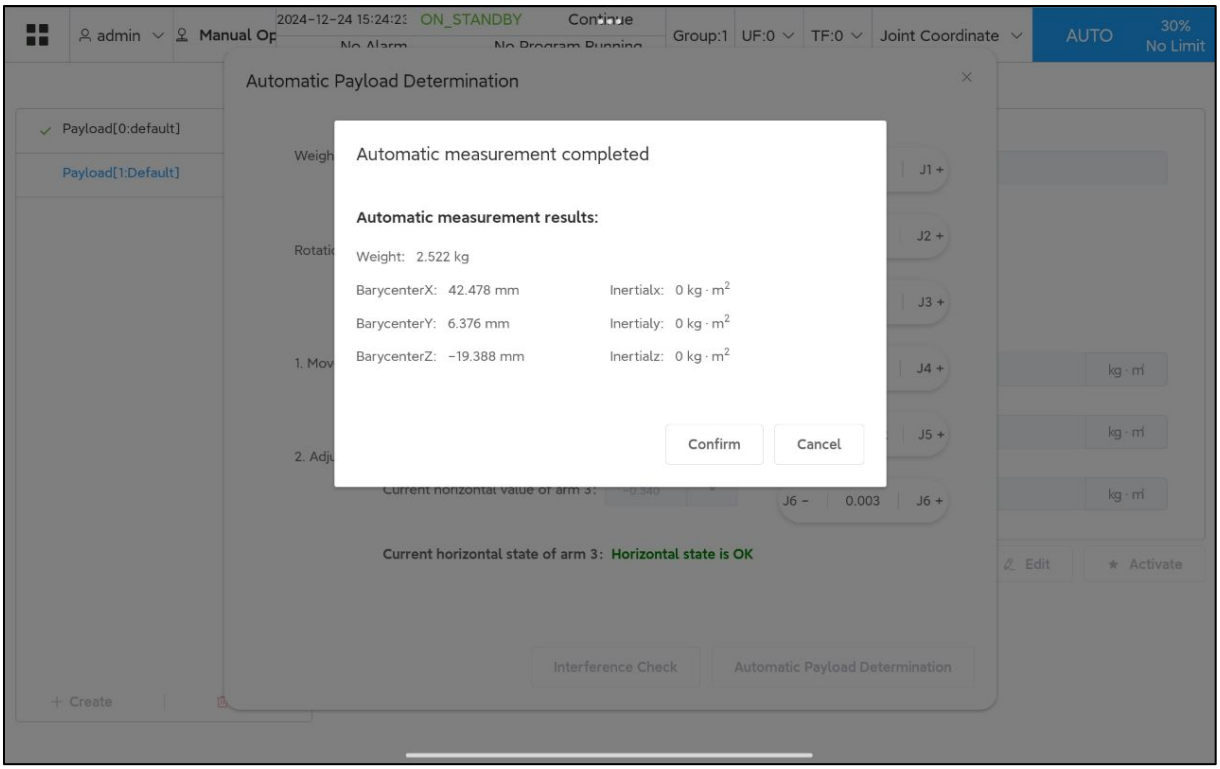


Fig. 2.47 Automatic measurement completed

2）Save: Confirm that the result is correct, update the parameters to the load setting interface and save them.

3）Cancel: Ignore the measurement result.

> ⚠ Caution
>
> When the interference check is completed, the robot will automatically reset to its initial position and posture.

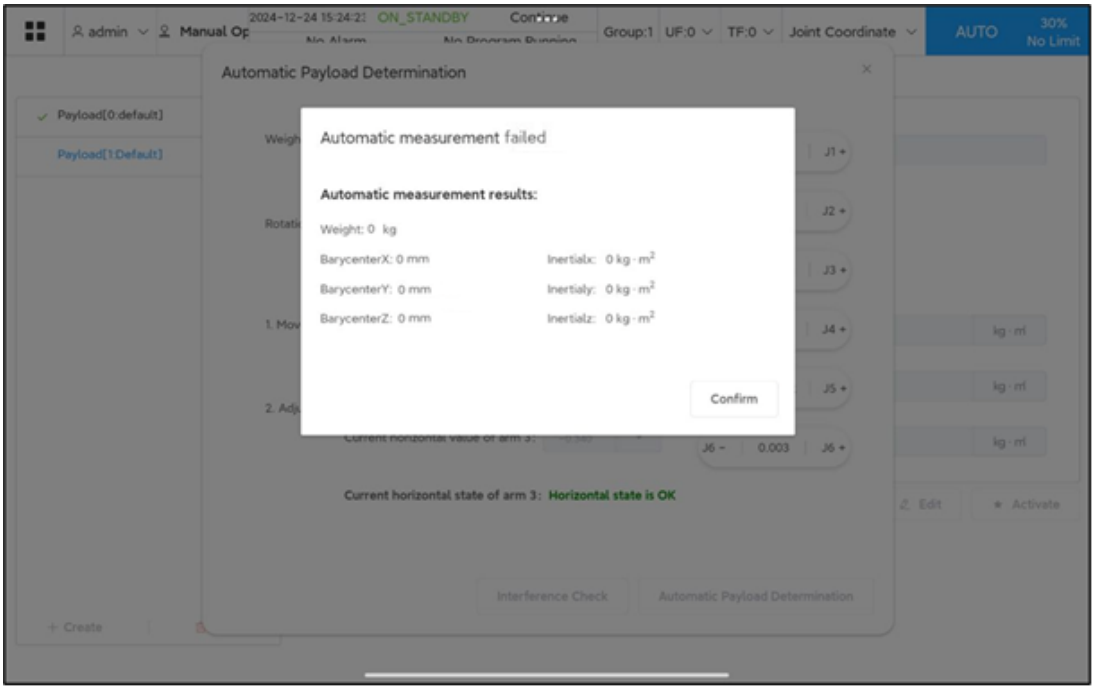


Fig. 2.48 Automatic measurement failed

1）Confirm: Ignore the measurement result.

> Caution
>
> If the automatic measurement fails, all the specific measurement result parameters will be displayed as 0.

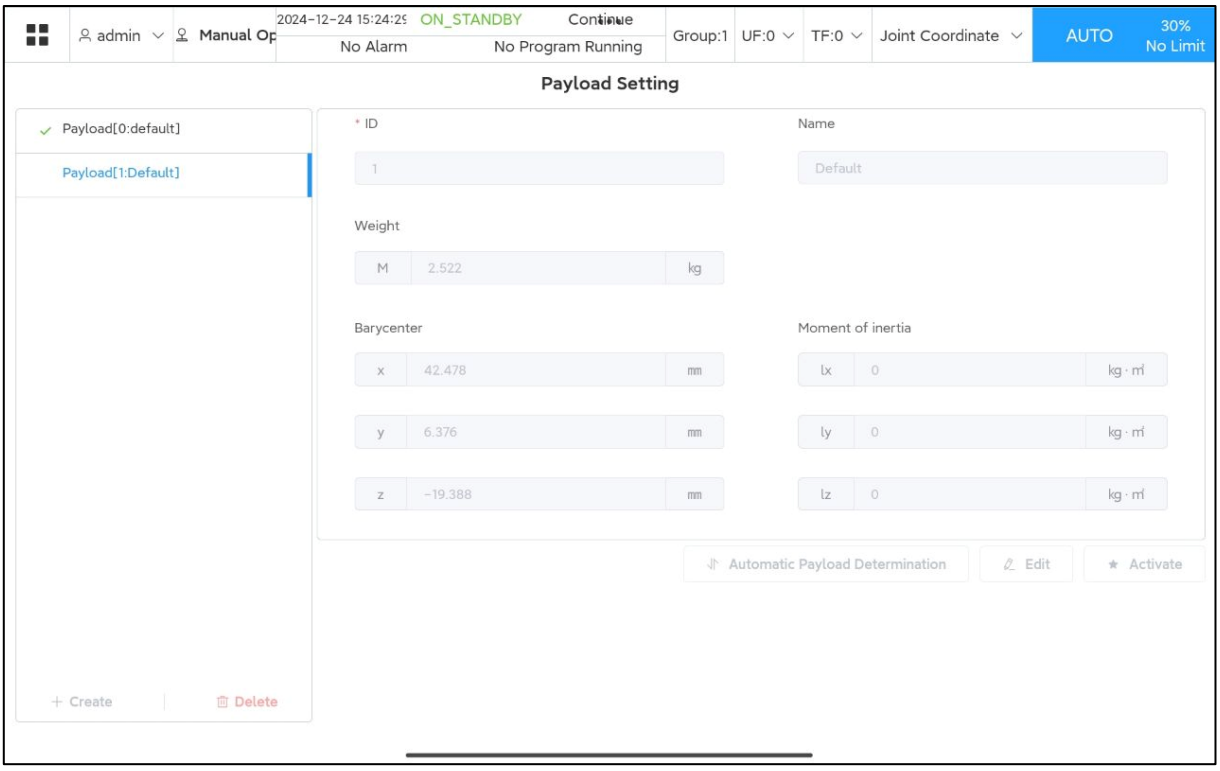


Fig. 2.49 The main interface for setting the load after the automatic measurement is completed

2.5 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:

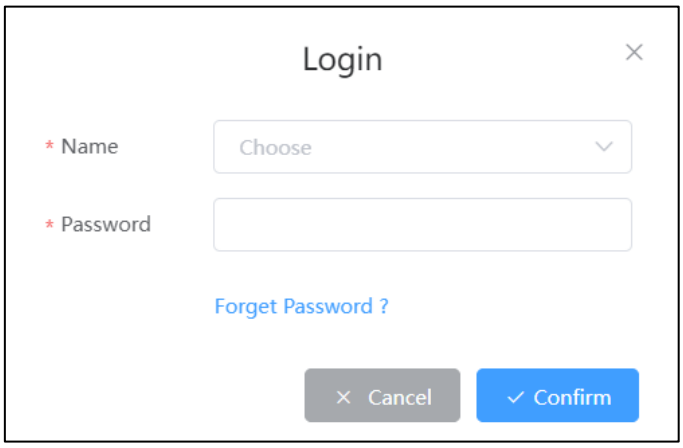


Fig. 2.50 Login Interface

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. 2.50.

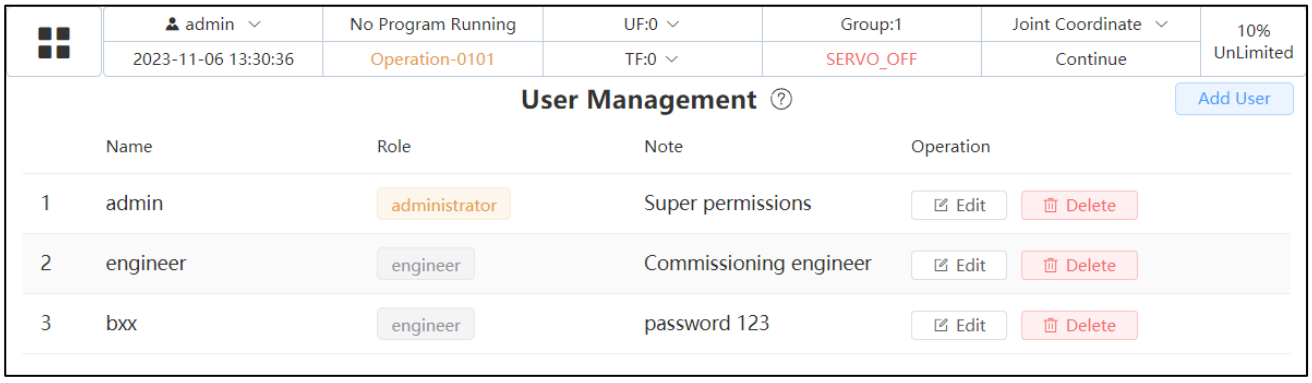


Fig. 2.51 User Operation Interface

3. Click "Edit" after the engineer to enter the “Edit User” interface, as shown in Fig. 2.51.

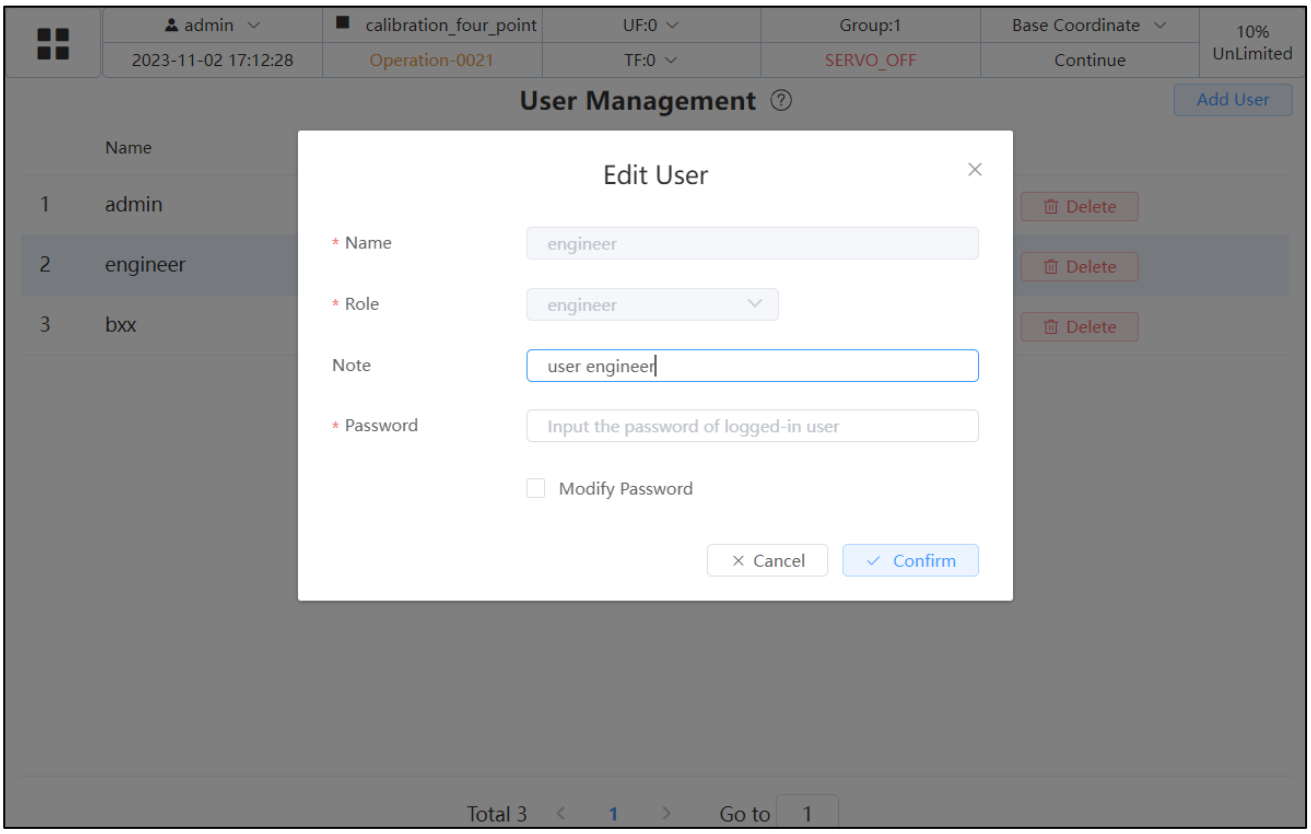


Fig. 2.52 “Edit User” Interface

4. Click ☐ before “Modify Password” to enter the password modification interface , as shown in Fig. 2.52.

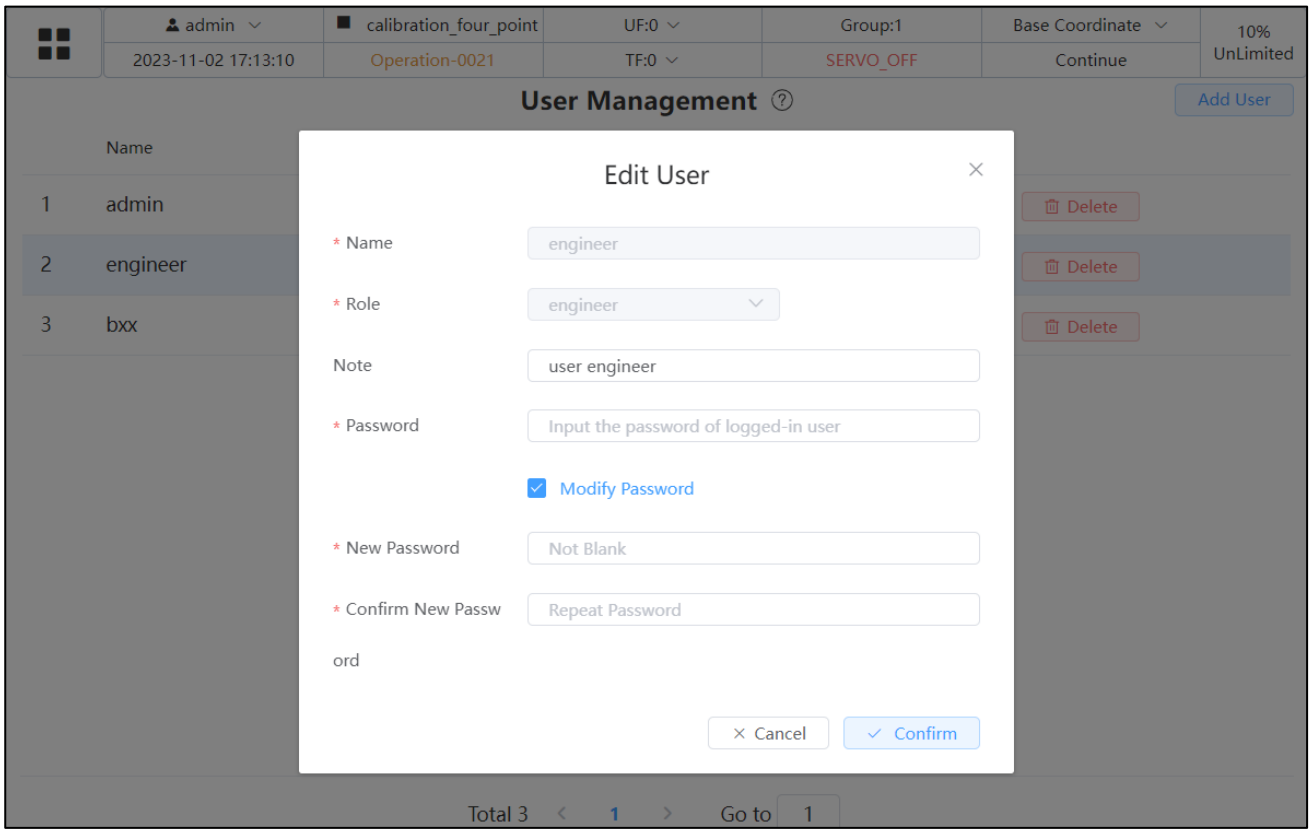


Fig. 2.53 “Modify Password” Interface

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

2.6 GENERAL SETTING

2.6.1 Time/Language setting

Successively click "Menu Button" → "Administration" → "Time/Language" to enter the screen as shown in Fig. 2.54.

The user can change current time and switch Chinese or English language.

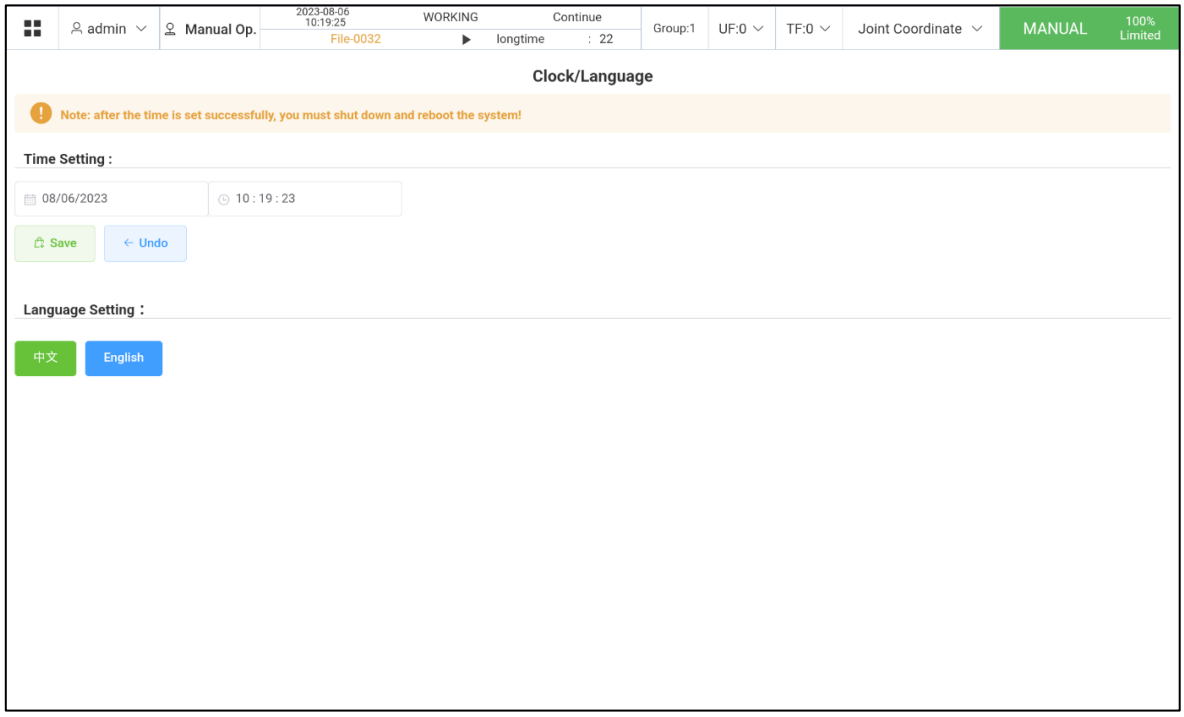


Fig. 2.54 Time/Language Setting Window

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

2.6.2 Network Configuration

In "Menu → System → Network Configuration", the IP address of the control cabinet can be set to obtain an IP address dynamically or set as a static IP. Please restart the system for the settings to take effect.

> **Caution**
>
> The 10.27.1.1/24 network segment is prohibited from being used on the IP side.

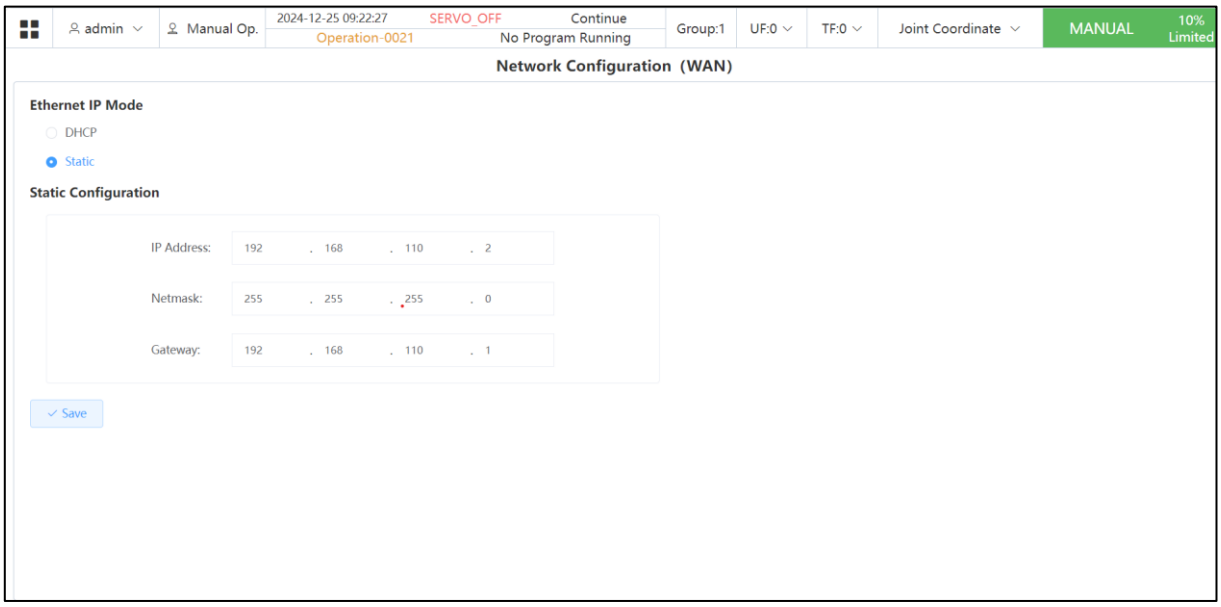


Fig. 2.55 Network Configuration(WAN)

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. 2.56.

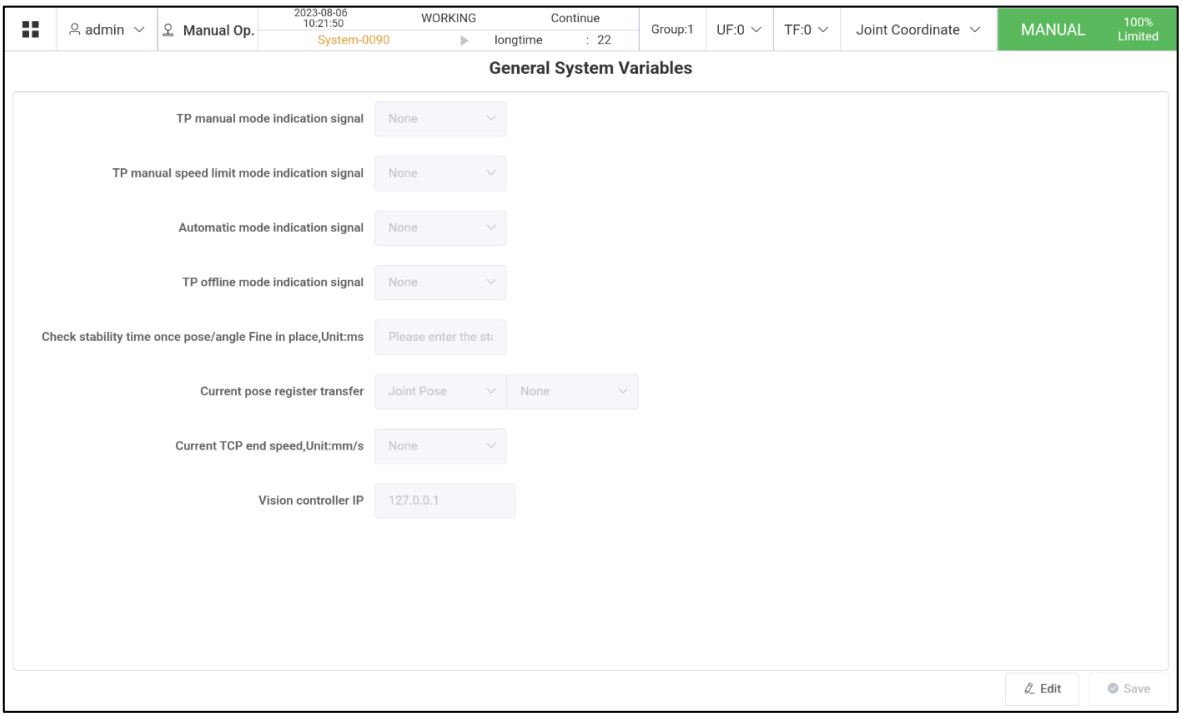


Fig. 2.56 System Variable Interface

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 mode switch is in the manual mode or off otherwise.
TP manual speed limit mode indication signal	Select the configured digital output signal. The signal status is ON when the mode switch is in the manual limited-speed mode or off otherwise.
Automatic mode indication signal	Select the configured digital output signal. The signal status is ON when the mode switch is in the auto mode or off otherwise.
TP offline mode indication signal	The collaborative robot is in off state.
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.

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.

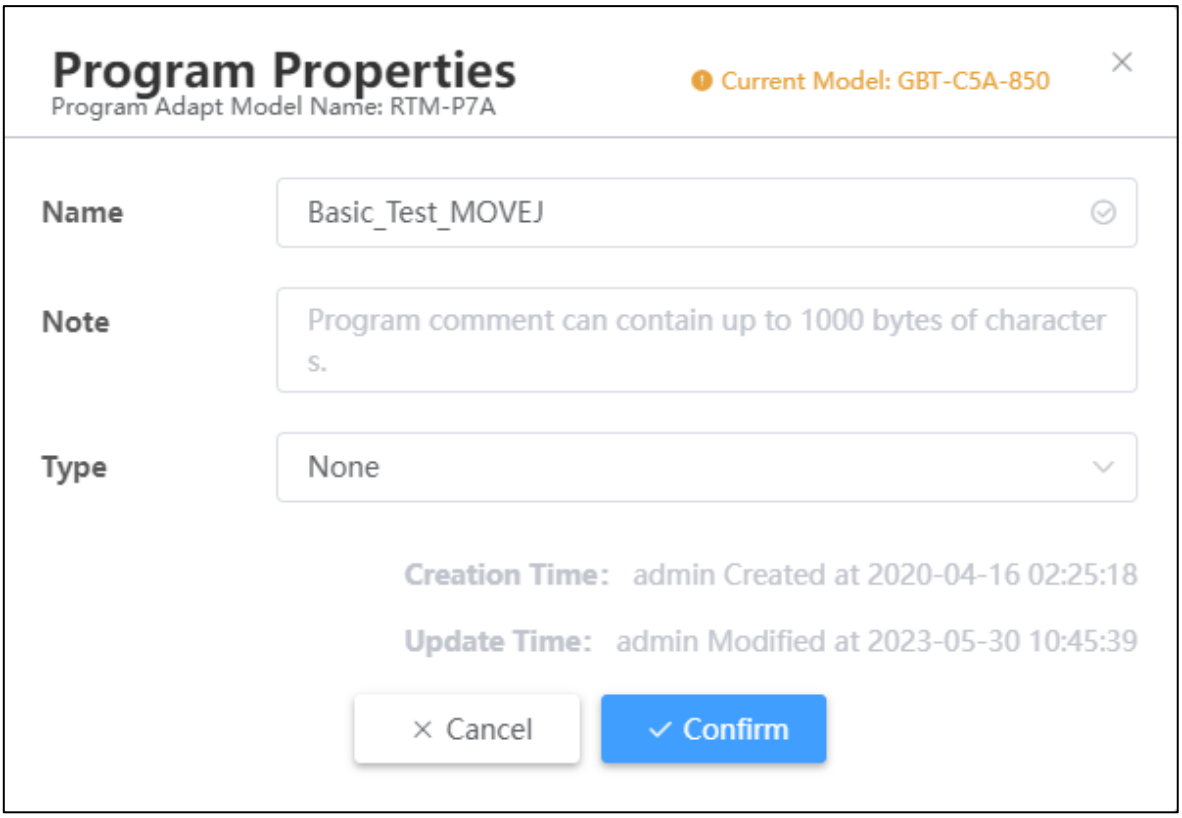


Fig. 3.1 Program Property Interface

The program property contains the following information:

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

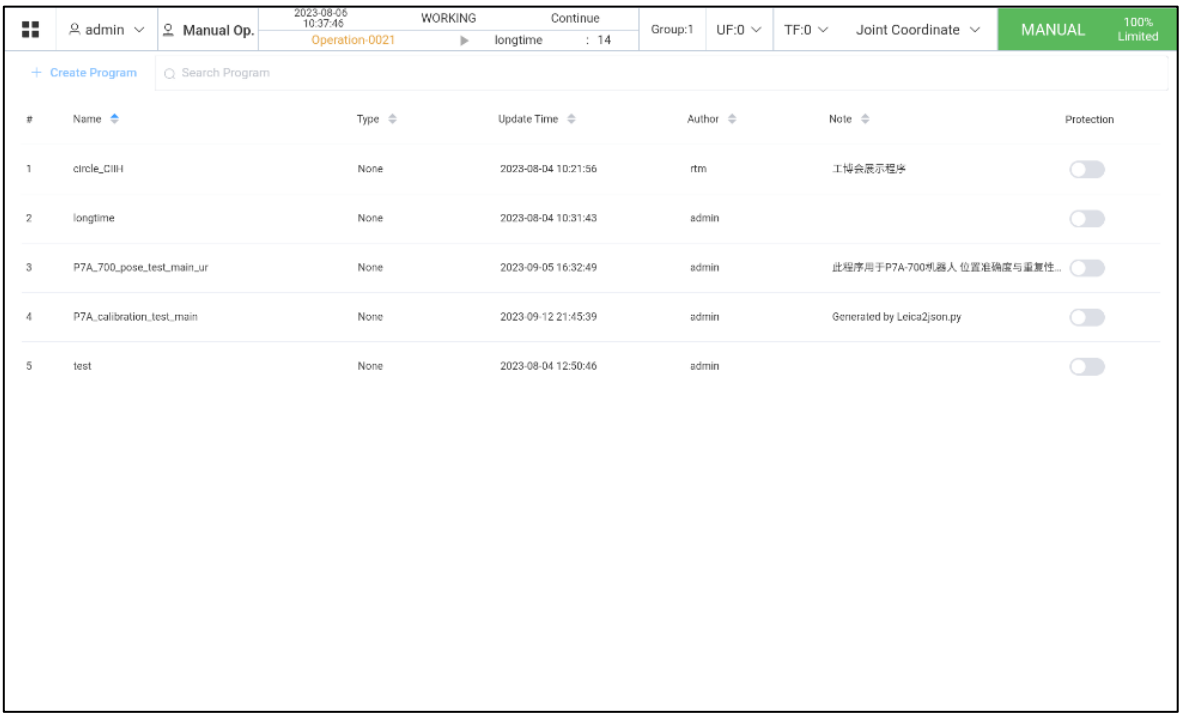


Fig. 3.2 Program List Interface

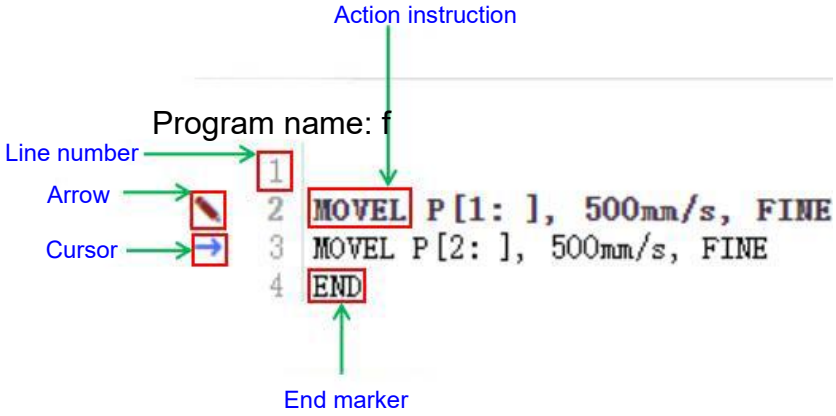


Fig. 3.3 Program Editing Statement Interface

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:

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

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.

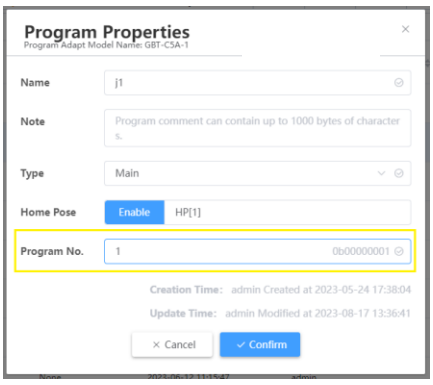


Fig. 3.4 Program Number Setting

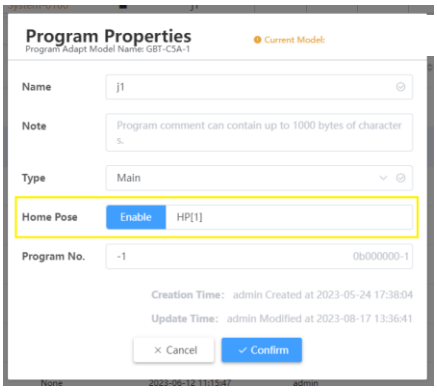


Fig. 3.5 Program Start Point Setting

Macro - Macro program:

During execution, a macro program can be called through macro start.

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. When the program type is selected as Macro, a list of program start points and program numbers is additionally shown on the program property interface as well. The programmer is allowed to choose whether to disable the "Program Start Point" function.

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).

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.

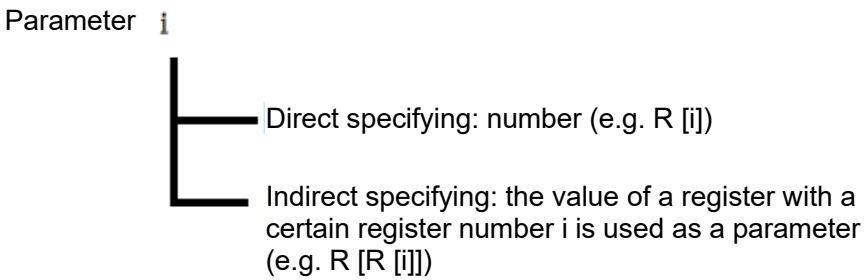


Fig. 3.6 Format of Parameter i

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)

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.

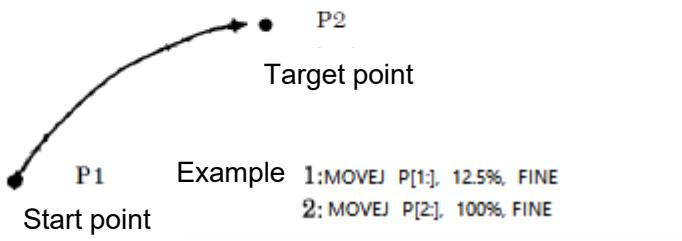


Fig. 3.7 Schematic Diagram of Joint Movement

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.

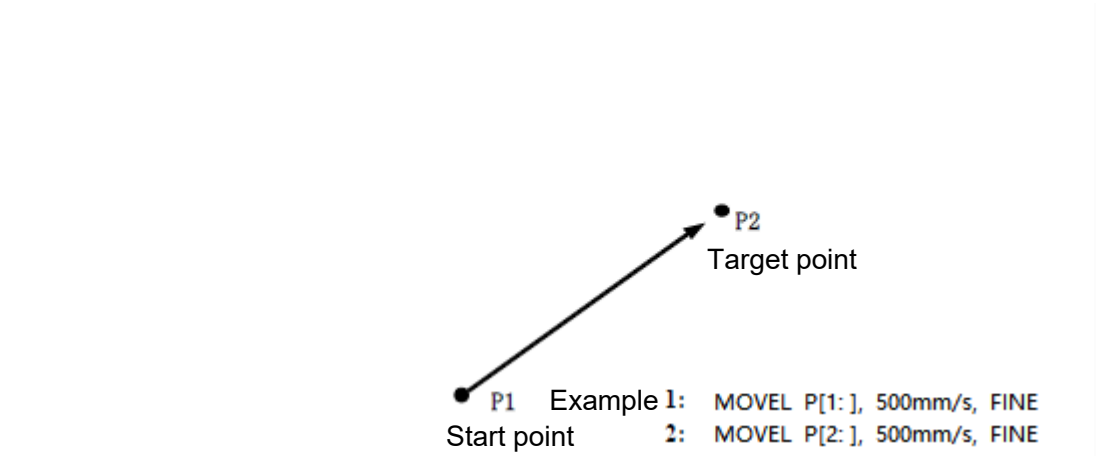


Fig. 3.8 Schematic Diagram of Linear Movement

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.

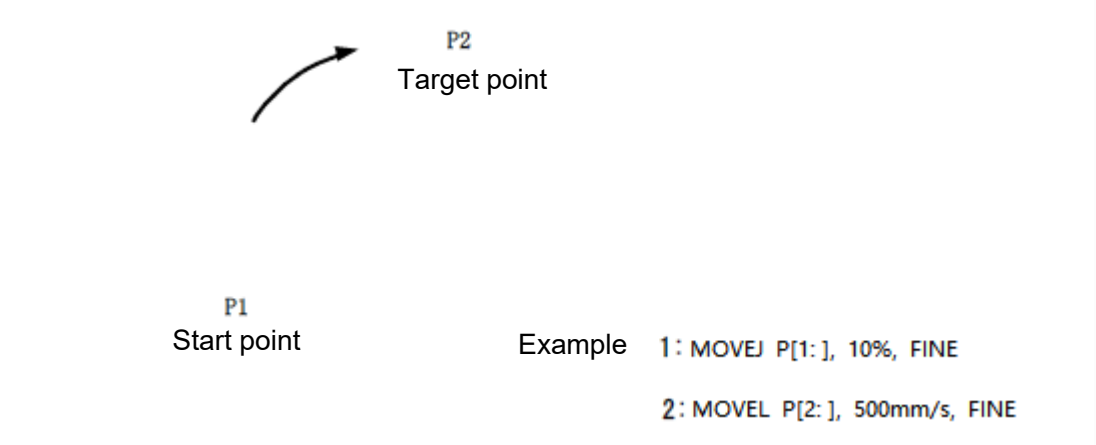


Fig. 3.9 Schematic Diagram of Rotary Movement

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.

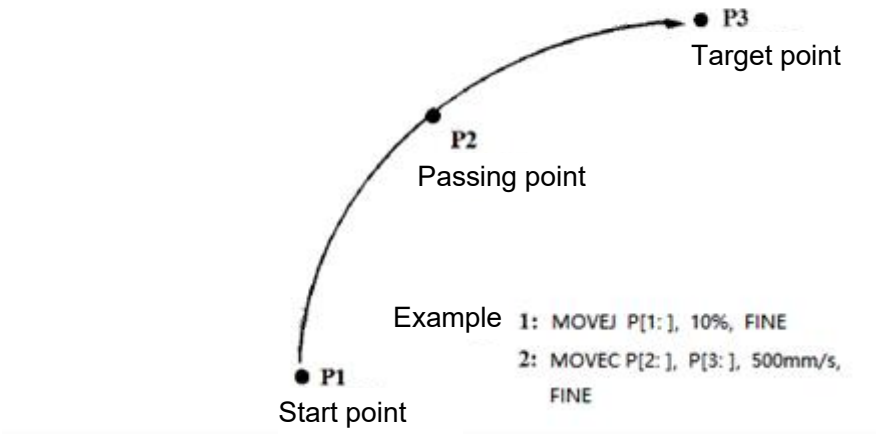


Fig. 3.10 Schematic Diagram of Arc Movement

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

It represents the rotation number of the axes (J3, J4, J6) for a six-axis 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%, SD50

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.

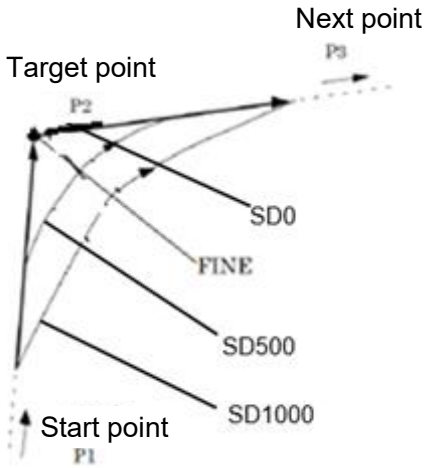


Fig. 3.11 Schematic Diagram of Corner Radius

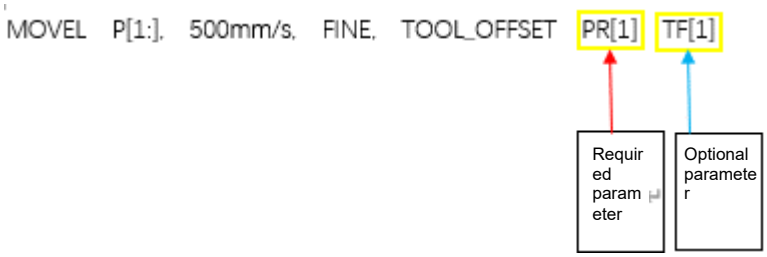
3.3.5 Additional commands for actions

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

- Tool offset instruction Tool_Offset

- Position offset instruction — Offset
- Trigger before instruction — TB
- Trigger after instruction — TA
- Distance-based trigger instruction — DB
- Motion parameter switch instruction — Swift
- Remote tool center point instruction — RTCP
- Skip instruction — SKIP

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

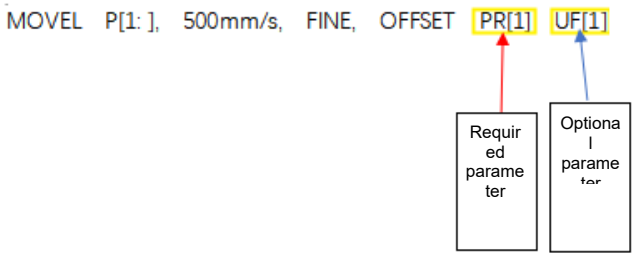
Tool offset instruction - The target position recorded in the position data causes the robot to move to a position only offsetting the offset amount specified in the tool offset condition. The offset condition is specified by the tool offset condition instruction.

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.

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.

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). 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.

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 0.01-1s. 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.

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.

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

Remote tool center point command (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.

> ⚠ Caution
>
> 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

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.

3.4 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

> 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 numerical 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.

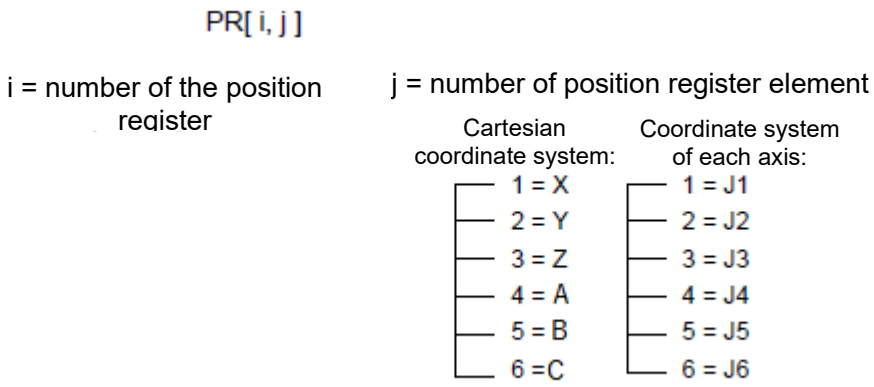


Fig. 3.11 Diagram for Position Register Element

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 MR[i] 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

3.5 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 Tool I/O instruction

The usage of the I/O instructions for Tools (TDI/TDO/TAI) is the same as that of the digital (DO/DI) I/O instructions. Just refer to the usage of the digital (DO/DI) I/O instructions.

3.6 Logical instruction

Logical instructions of Collaborative 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:

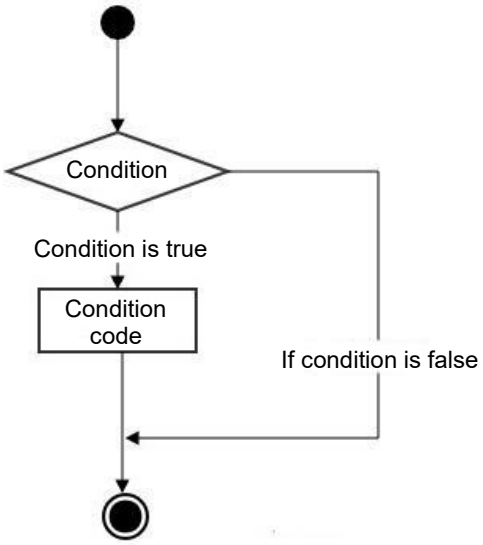


Fig. 3.12 Schematic Diagram of Conditional Statement Execution Process

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:

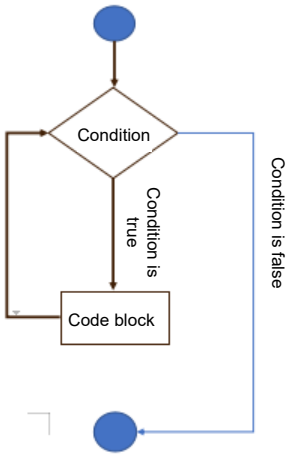


Fig. 3.14 Execution Process of WHILE Statement

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:

Program name: (WHILE)

```
1 WHILE DI[2: ] = DI[2: ]
2   MOVEJ P[1: ] 100% FINE
3   WHILE DI[1: ] <> 1
4     MOVEJ P[2: ] 100% FINE
5     MOVEL P[3: ] 4000mm/s FINE
6   END WHILE
7 END WHILE
8 END
```

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
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 D0[2]=0N, the program jumps directly from line 2 to line 4, and the execution continues down from line 4.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 and final position calls the subprogram Basic_Test_MOVEJ.

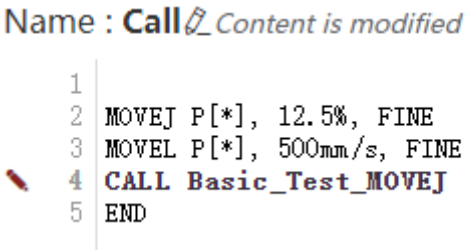


Fig. 3.16 Example of Call Instruction Programming

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.

```
1  R[1: default] = 5
2  MOVEJ P[1: ] 100% FINE
3  CALL gripperClose
4  WAIT 2 sec
5  MOVEJ P[2: ] 100% FINE
6  MOVEL P[3: ] 4000mm/s FINE
7  WAIT R[1] sec
8  CALL gripperOpen
9  MOVEJ P[1: ] 100% FINE
10 END PROG
```

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.

```
1  MOVEJ P[*] 100% FINE
2  MOVEL P[1: ] 4000mm/s FINE
3  MOVEC P[3: ] P[4: ] 4000mm/s FINE
4  IF R[1: default] = 100
5    PAUSE
6  END IF
7  MOVEJ P[5: ] 100% FINE
8  END PROG
```

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.

```
1  MOVEJ P[*] 100% FINE
2  MOVEL P[1: ] 4000mm/s FINE
3  MOVEC P[3: ] P[4: ] 4000mm/s FINE
4  # 自定义ERROR CODE = 10001
5  IF R[1: default] = 10001
6    ABORT
7  END IF
8  MOVEJ P[5: ] 100% FINE
9  END PROG
```

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)

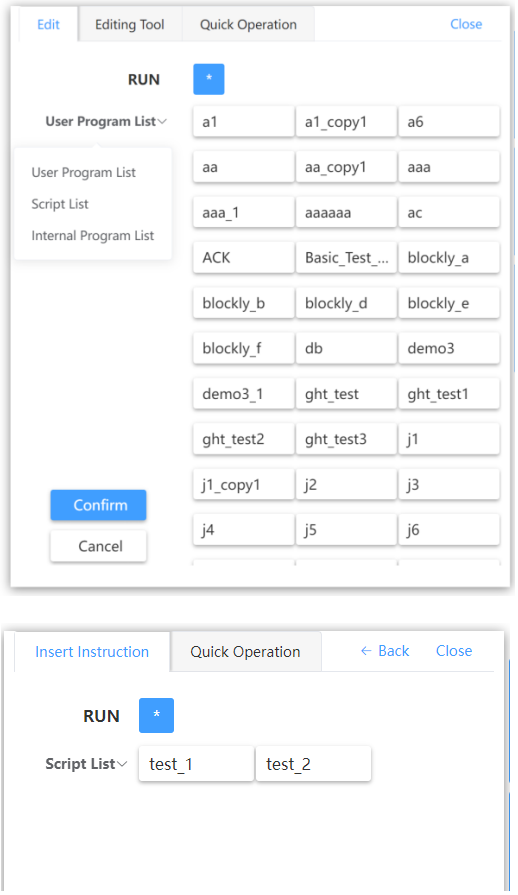


Fig. 3.20 Example of RUN Instruction Insertion

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.

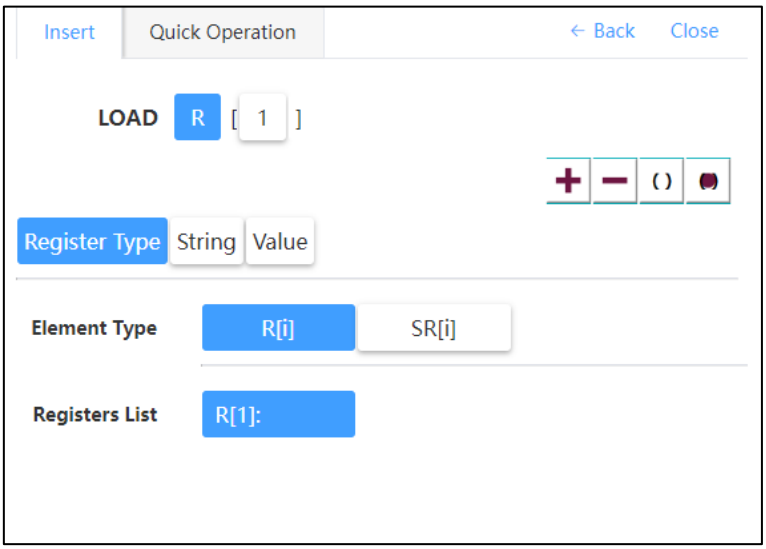


Fig. 3.21 Example of Load Instruction Insertion

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.

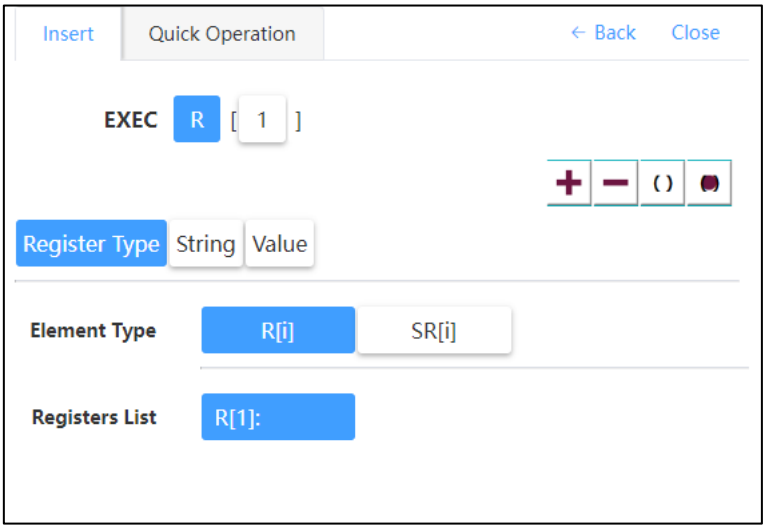


Fig. 3.22 Example of Exec Instruction Insertion

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.

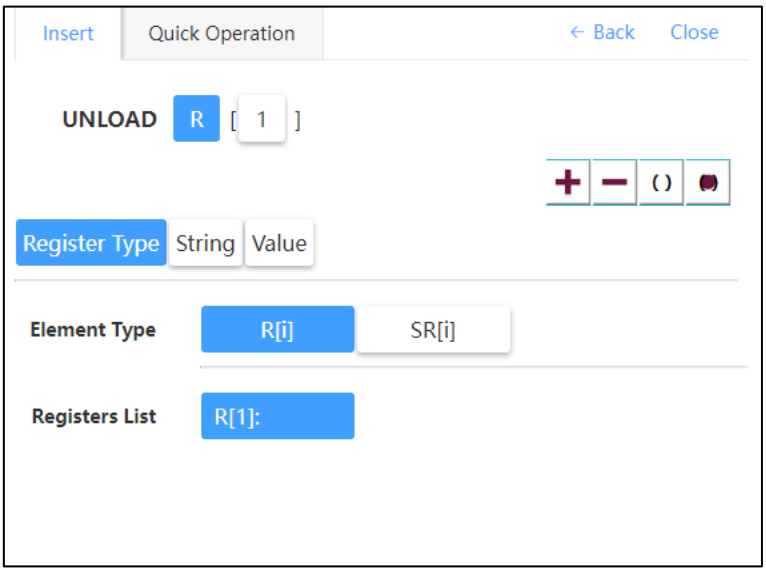


Fig. 3.23 Example of Unload Instruction Insertion

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

1. 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.

2. 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

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

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

3.8.11 Socket - Socket instruction

Overview:

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

SOCKET OPEN SK[*], parameter

The parameter SK [*] represents the current socket device used, and the optional parameter represents the statement execution result (see Chapter 10 Socket Error Code List) and is represented by a number register (R [*]).

Additional descriptions:

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

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

SOCKET CONNECT SK[*], parameter

The parameter SK [*] represents the current socket device used, and the optional parameter represents the statement execution result (see the Socket Error Code List) and is represented by a number register (R [*]).

SocketRecv: The SocketRecv instruction is used to read characters from a Socket and can specify a maximum length.

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

The parameter SK [*] represents the current socket device used, the parameter R [*] represents the acceptance length; SR [*] indicates that the received data exists in the string register (SR [*]), and the optional parameter 2 represents the statement execution result (see the Socket Error Code List) and is represented by a number register (R [*]).

Additional descriptions:

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

SocketSend: The SocketSend instruction is used to write a string to a Socket and send it to the opposite end.

SOCKET SEND SK[*], “”, parameter2

The parameter SK [*] represents the current socket device used, the parameter "" represents the content (string) to be sent, which can be a constant or a string register (SR [*]), and the optional parameter 2 represents the statement execution result (see the Socket Error Code List) and represented by a number register (R [*]).

Additional descriptions:

- Both Server and Client can call this instruction.
- Maximum length is 254 if the string to be sent is represented by SR.
- It is allowed to directly send a constant string or store the string in a string register and send the content by selecting that register.

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

SOCKET CLOSE SK[*], parameter

The parameter SK [*] represents the current socket device used, and the optional parameter represents the statement execution result (see the Socket Error Code List) and is represented by a number register (R [*]).

> ⚠ Caution
>
> Optional parameters may or may not be used.

3.8.12 Compound operation instruction

Compound operation instruction can combine various operators and data in assignment, conditional comparison and wait statements of TP program. The compound operation instruction supports the parenthesis “()”.

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

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

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

  Execute statement

  END IF

  In case of no parentheses in a sentence, it becomes a common operation instruction.
- WAIT (DO[1] = ON AND D0[2] = OFF) OR (R[1] = 1 OR R[2] = 2)

Data type

The following data types can be used in the compound operation instruction.

Type	Value	Data
Numerical value	Numerical values can be processed as data. Both integers and real numbers can be used.	Elements of registers, constants and position registers
Bool	The data can be set to any value of ON or OFF.	DI/O, UI/O, ON, OFF

Operator

The following operators can be used in the compound operation instruction.

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

- The data of numerical type can only be used for arithmetic operators.
- The output data of arithmetic operators is always numerical.

Operator	Operation
AND	Logical product of left and right sides
OR	Logical sum of left and right sides

- Logical operators are only applicable for Boolean data.
- The output data of the logical operator is always Boolean.

Operator	Operation
=	Return ON when left and right sides are equal. Return OFF when left and right sides are unequal.
<>	Return ON when left and right sides are unequal. Return OFF when left and right sides are equal.
<	Return ON when left side is smaller than right side. Return OFF when left side is greater than right side.
>	Return ON when left side is greater than right side. Return OFF when left side is smaller than right side.
<=	Return ON when left side is smaller than or equal to right side. Return OFF when left side is greater than right side.
>=	Return ON when left side is greater than or equal to right side. Return OFF when left side is smaller than right side.

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

The following shows the priority order of operators.

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

Conditional statement

The following are examples of compound operation instructions used 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 instruction

The following is an example of compound operation instructions used in wait instruction.

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

- Compound expressions can be specified in conditional statements of wait instruction.

- The result of a conditional statement must be Boolean.
- The wait instruction is waiting before the expression result becomes ON.

Addition and modification of compound operation instructions

For the instructions with compound operation function, the instruction editing interface provides symbol descriptions and the symbols in the following table are adopted to add and modify compound operation instructions.

List of Symbols Used for Addition and Modification of Compound Operation Instructions

	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.

3.9 String instructions

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 VISION INSTRUCTION

The program on the TP uses vision instructions. It is required to modify IP of the vision controller (PC) to ensure that the IP is in the same LAN segment as IP of the TP (robot controller).

	TP (robot controller)	Vision controller (PC)
IP	192.168.110.2	192.168.110.3
Mask	255.255.255.0	255.255.255.0

3.11.1 VISION FIND instruction

It is used to start a vision program.

Instruction format: VISION FIND + “Name of Vision Program”

E.g.: VISION FIND test starts the “test” of vision program.

3.11.2 VISION GET_QUANTITY instruction

It is used to obtain the number of vision detections.

Instruction format: VISION GET_QUANTITY + “Name of Vision Program” + R[i]

E.g.: VISION GET_QUANTITY test, R[1:] vision program test is to assign the detected number to the number register R [1].

3.11.3 MODELID GET MODEL ID instruction

This instruction assigns the model ID of the vision register to the number register R [i].

Instruction format: R[i] = VR[I, MODELID]

E.g.: R[1]=VR[1, MODELID]

3.11.4 GET_OFFSET instruction

It is used to obtain detection results from a vision program and store them in a specified vision register.

Instruction format: VISION GET_OFFSET + “Name of Vision Program” + VR[i] + GOTO LABEL[i]

Remark: Wait for image processing when the vision program has not finished processing.

When the vision program does not find the result, jump to the program-specified label according to GOTO.

The instruction can be executed repeatedly when the vision program finds multiple results.

E.g.: VISION GET_OFFSET test, VR[1], GOTO LABEL [1]

3.11.5 VOFFSET instruction

This instruction is used to offset the position of the robot with the data stored in the vision register.

Instruction format: …, VOFFSET, VR[i]

E.g.: MOVEL PR[1:], 500mm/s, FINE, VOFFSET VR[1]

3.11.6 Assignment of detected location information

Assign the detected position information of the vision register VR [i] to the pose register PR [i].

Instruction format: PR[i] = VR[i]

E.g.: PR[1] = VR[1]

3.11.7 Examples of Vision Program

Example 1: Detect multiple identical workpieces in one photography

```
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: Detect different workpieces by photography

```
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, VOFFSETVR[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

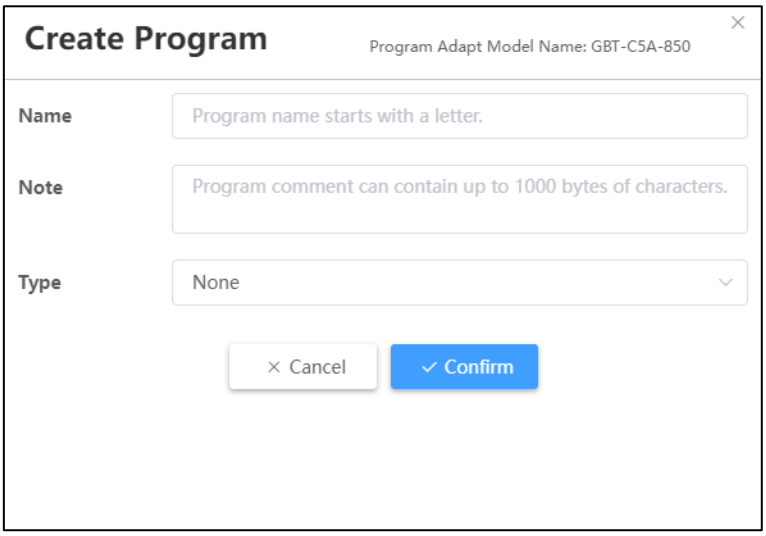


Fig. 4.2 Create Program Window

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.

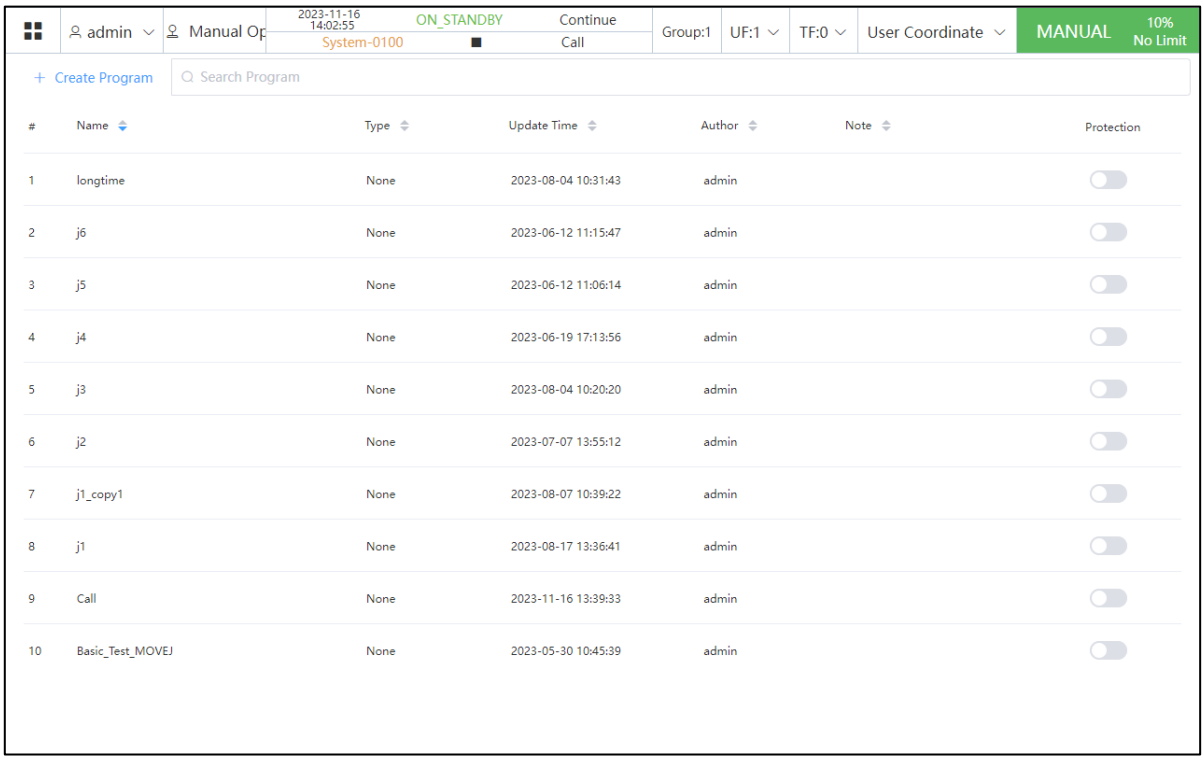


Fig. 4.3 Program List Interface

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.

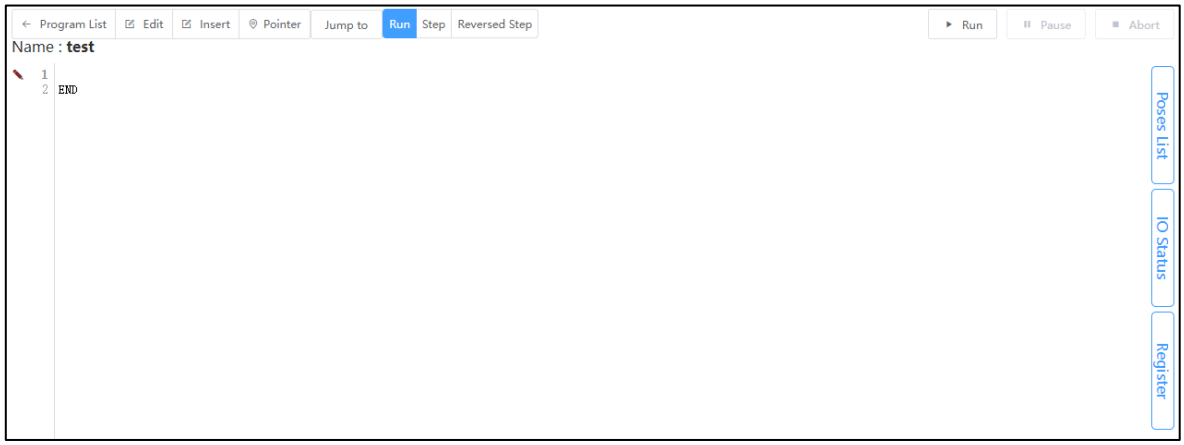
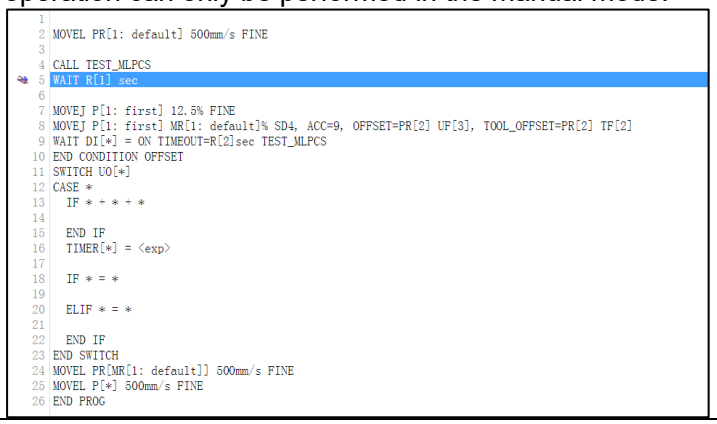


Fig. 4.4 Program Editing Interface

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.  1 2 MOVEL PR[1: default] 500mm/s FINE 3 4 CALL TEST_MLPCS 5 WAIT R[1] sec 6 7 MOVEJ P[1: first] 12.5% FINE 8 MOVEJ P[1: first] MR[1: default]% SD4, ACC=9, OFFSET=PR[2] UF[3], TOOL_OFFSET=PR[2] TF[2] 9 WAIT DI[*] = ON TIMEOUT=R[2]sec TEST_MLPCS 10 END CONDITION OFFSET 11 SWITCH UO[*] 12 CASE * 13 IF * + * + * 14 15 END IF 16 TIMER[*] = <exp> 17 18 IF * = * 19 20 ELIF * = * 21 22 END IF 23 END SWITCH 24 MOVEL PR[MR[1: default]] 500mm/s FINE 25 MOVEL P[*] 500mm/s FINE 26 END PROG
“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.

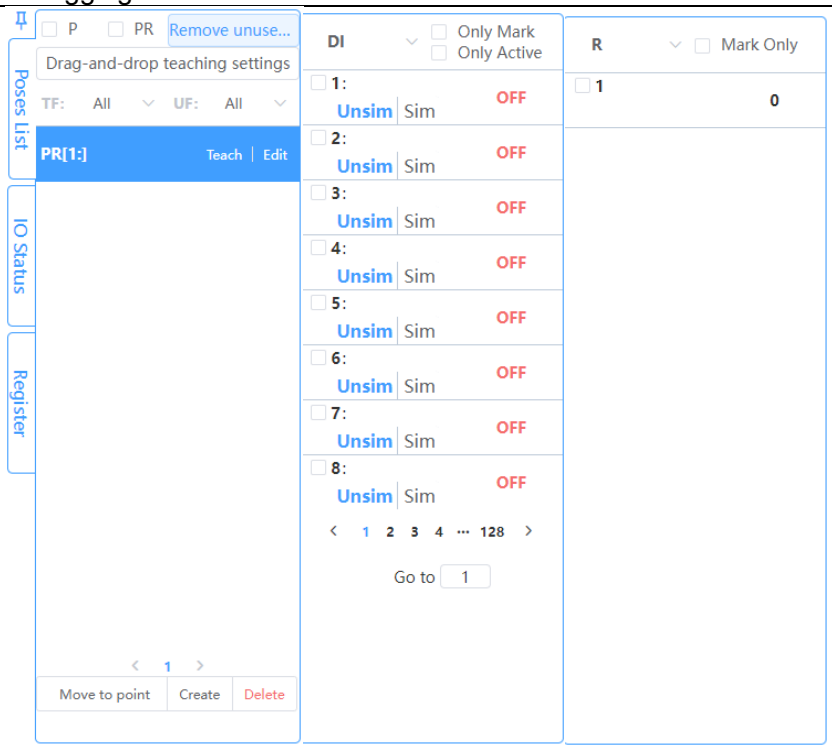
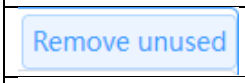


Fig. 4.5 shows the pose list, I/O status list and register list from left to right.

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.

Only Mark	Click here on the I/O status screen to mask the monitored I/O status information.
Only Active	Click here on the I/O status screen to mask unconfigured I/O states and only display I/O states successfully configured.
R	Click here on the register screen to switch to MR and SR register monitoring screens.
Mark Only	Click here on the I/O 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.

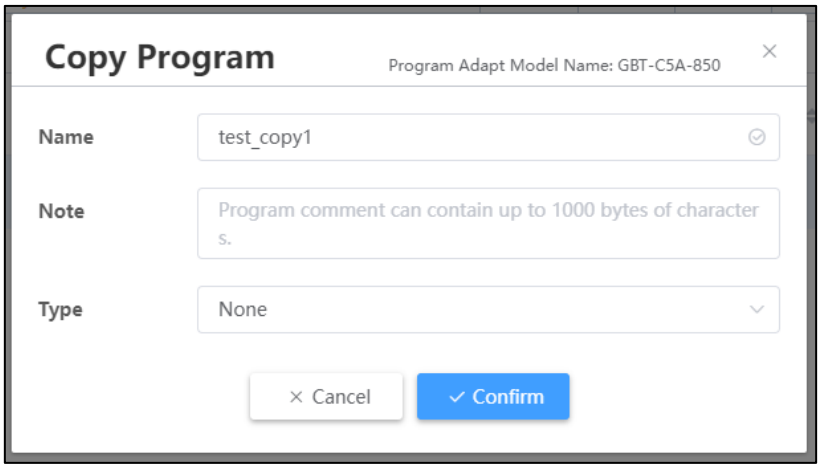


Fig. 4.6 Copy Program Window

4.1.3 Delete program

Choose the desired program in the program list interface and click "Delete" 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.

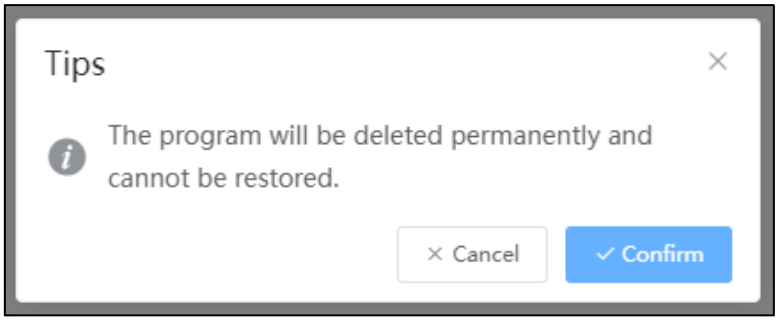


Fig. 4.7 Delete Program Window

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.

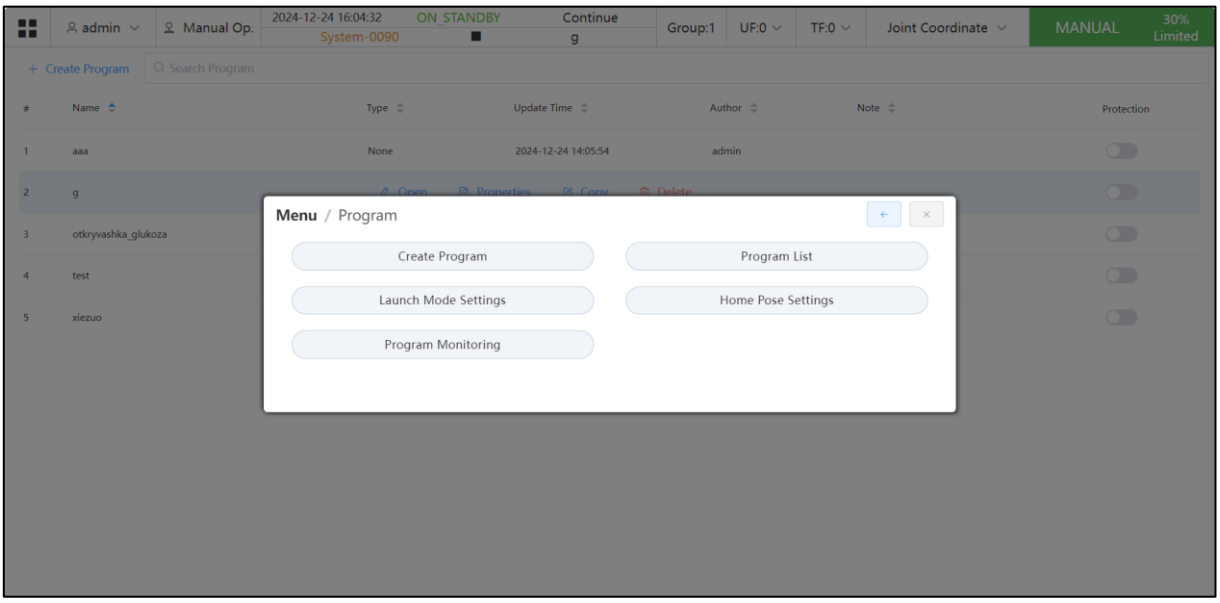


Fig. 4.8 Program Menu Window

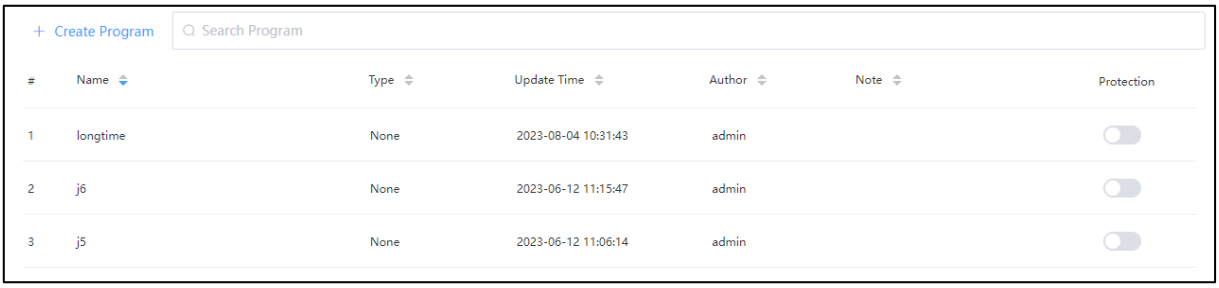


Fig. 4.9 Program List Window

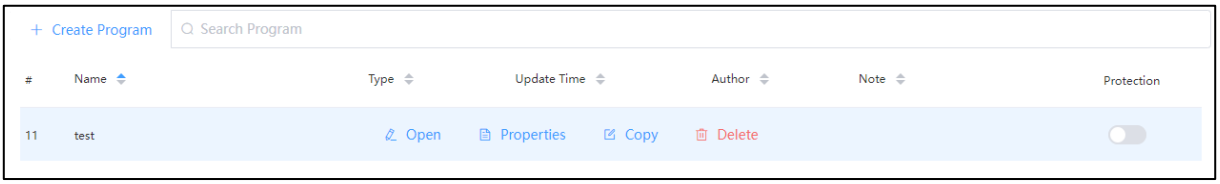


Fig. 4.1 Selected Program

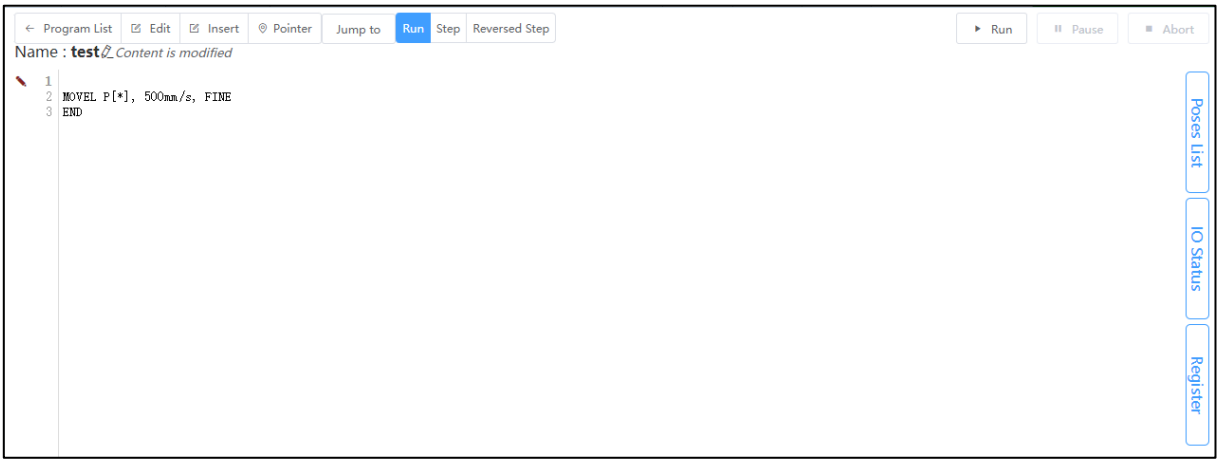


Fig. 4.11 Program Editing Window

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.

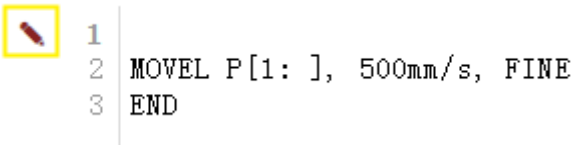


Fig. 4.12 Pen Cursor

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.

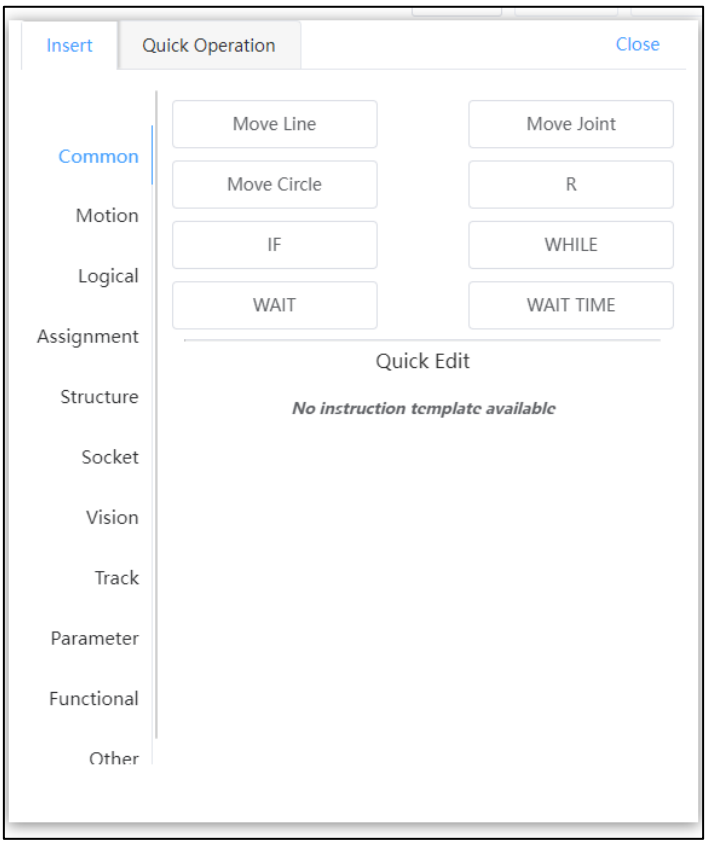


Fig. 4.13 Instruction Selection Window

3. Modify the contents of motion instruction according to the actual situation.

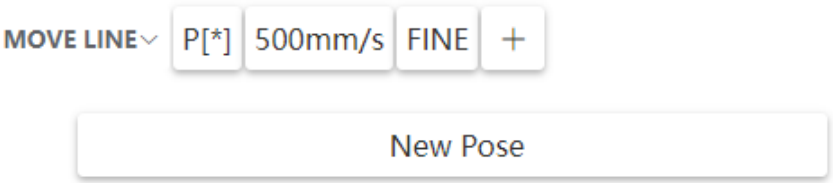


Fig. 4.14 Motion Instruction Editing Window

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).

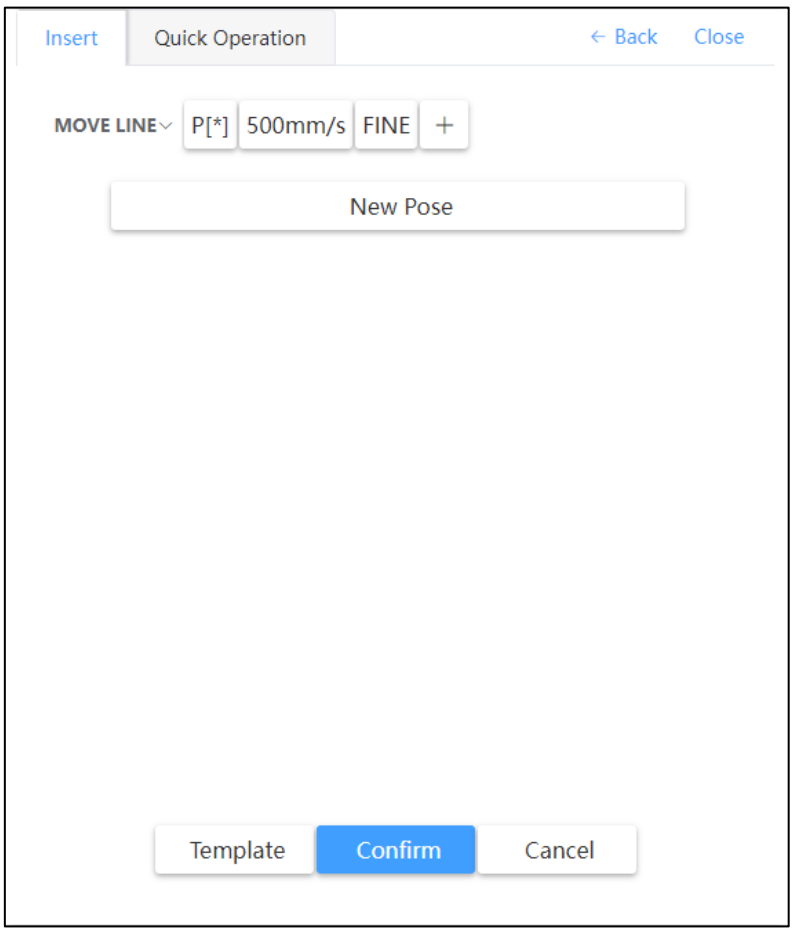


Fig. 4.15 Example of Continuous Motion Instruction

Additional descriptions:

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.

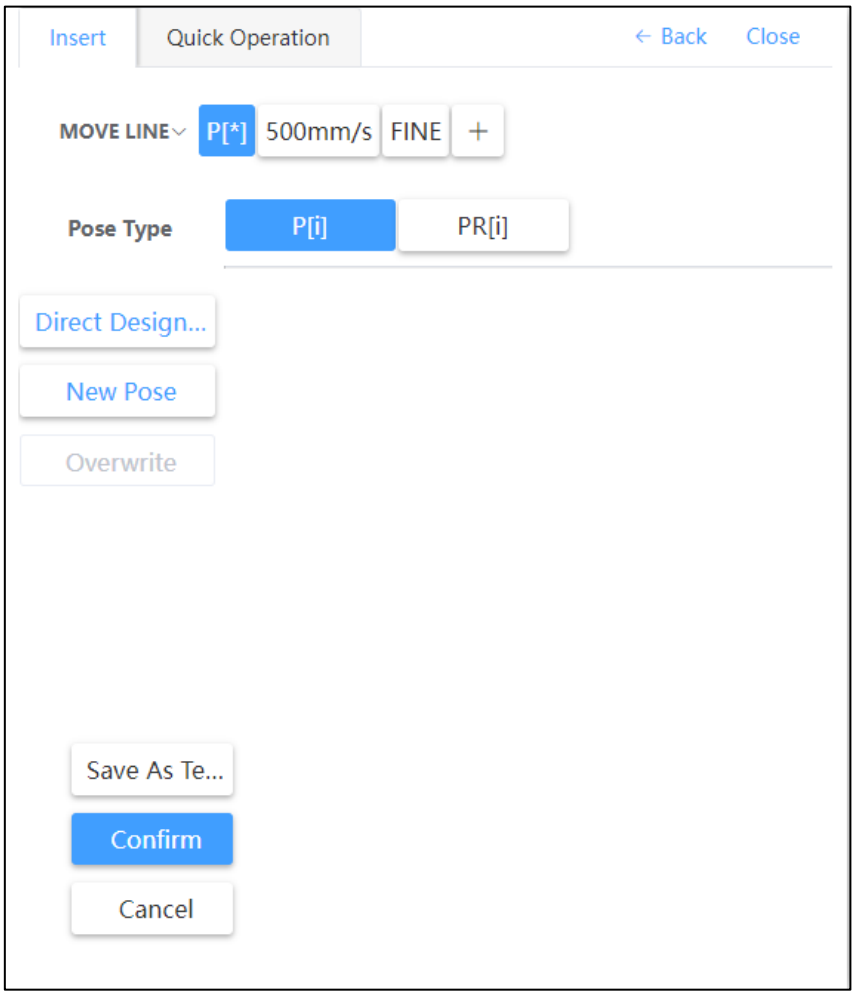
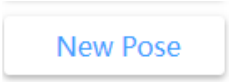


Fig. 4.16 Point or Position Register Creation Window

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, `MOVEL PR[MR[1: ]], 500mm/s, FINE` shows the indirect method. 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.

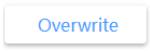
New Pose

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.

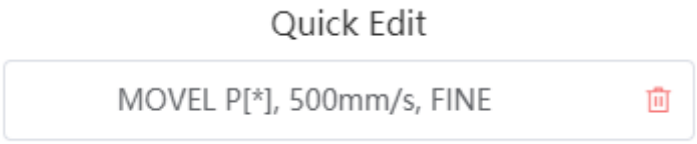


Fig. 4.17 Motion Instruction Template Window

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.

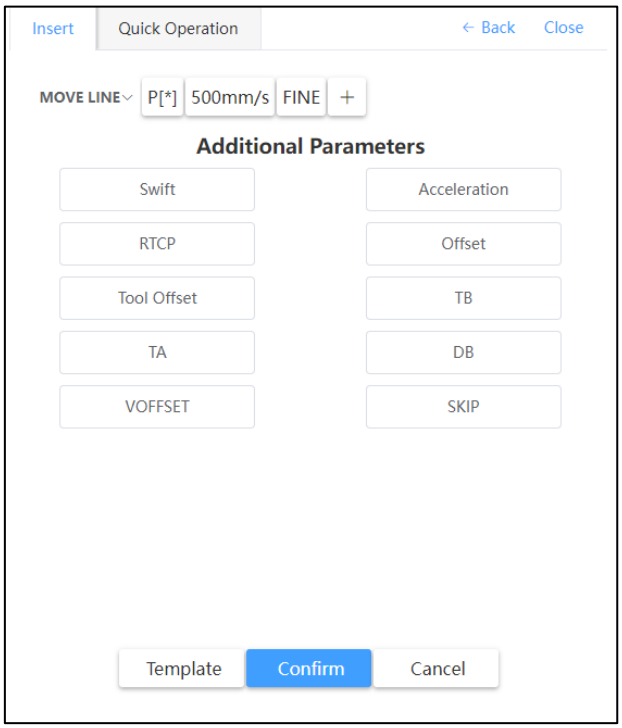


Fig. 4.18 Additional Instruction Editing Window

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.

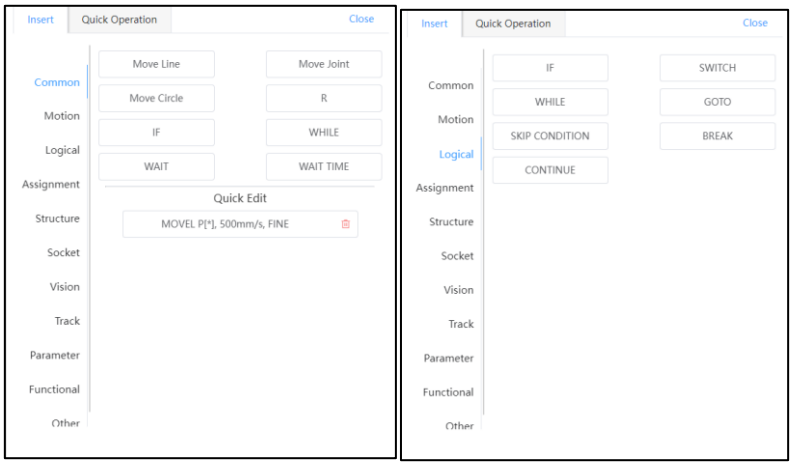


Fig. 4.18 Insert Instruction Window

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.

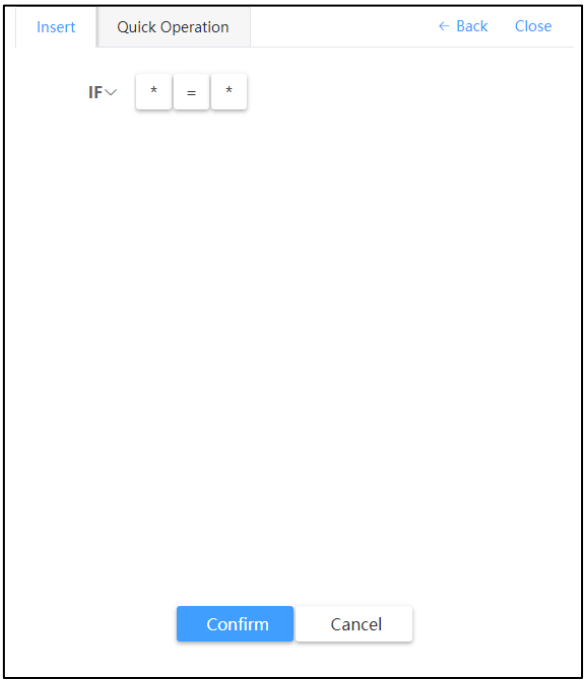


Fig. 4.19 Insert Instruction Window

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
```

```
IF R[1: default]  =  R[2: default]
  R[2: default] = 3
END IF
```

Fig. 4.20 Example of IF Instruction Program

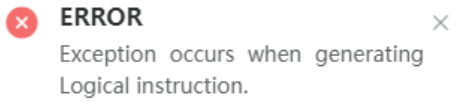
Additional descriptions of IF statement

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.

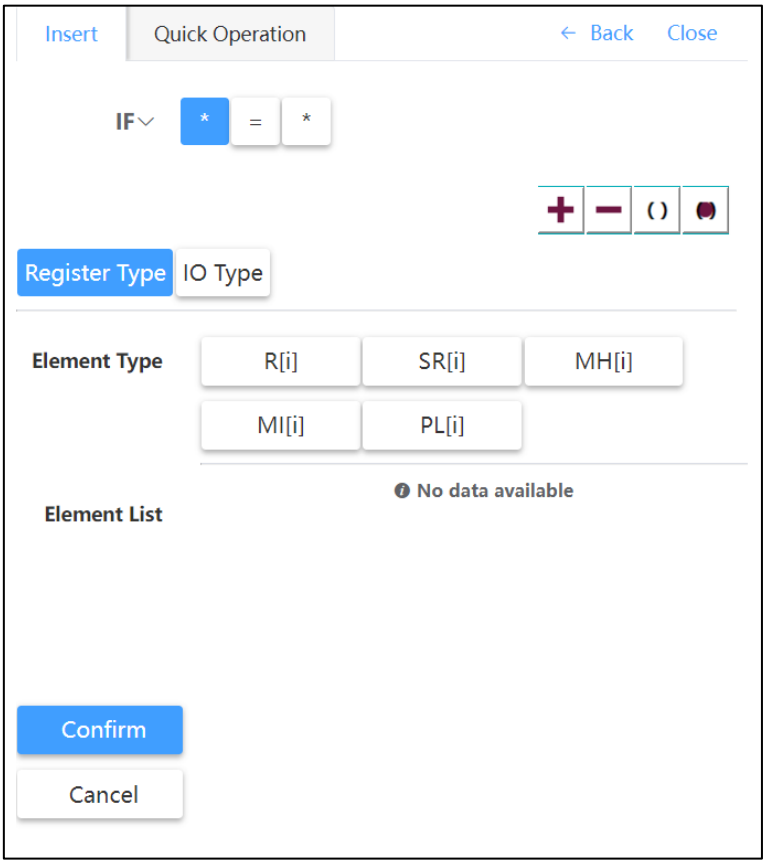


Fig. 4.22 IF Statement Parameter Editing Interface

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.

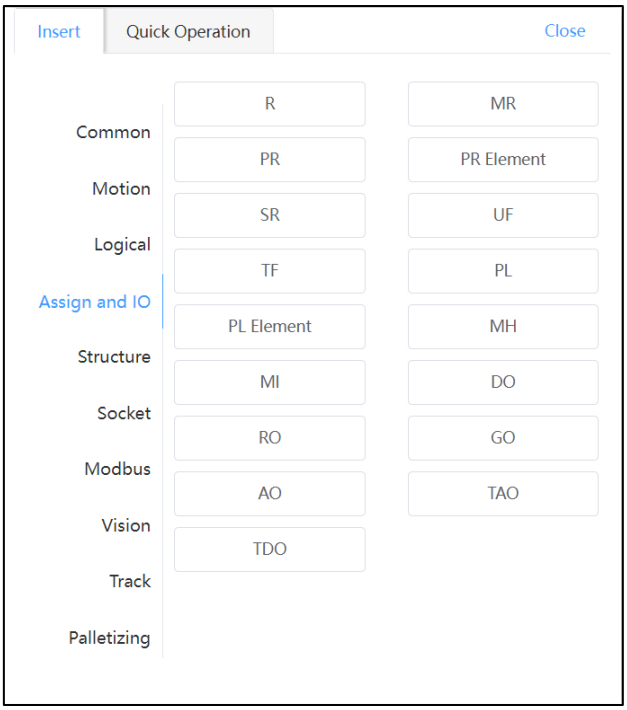


Fig. 4.23 “Assign and I/O" Instruction Interface

Click "R" to enter the R register instruction parameter filling interface, as shown in Fig. 4.24.

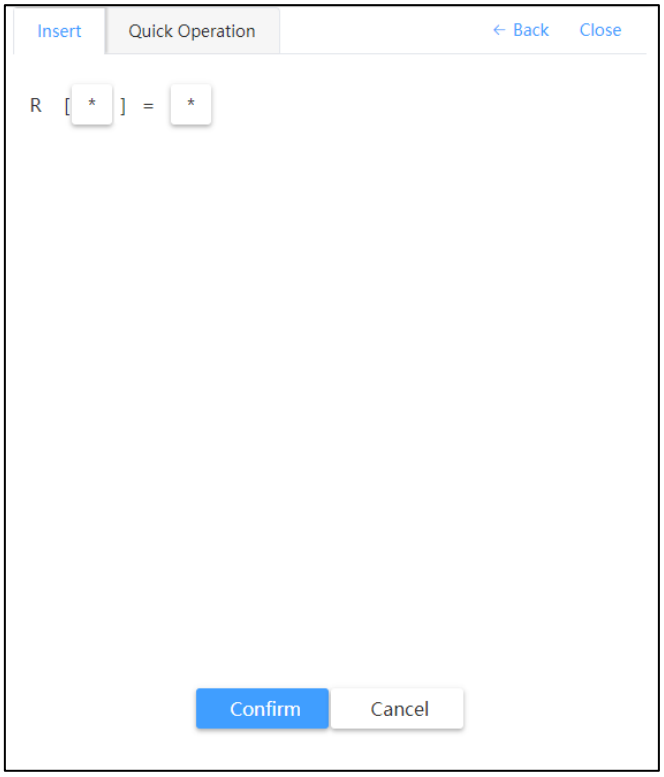


Fig. 4.24 “Assign and I/O" Instruction Interface

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.

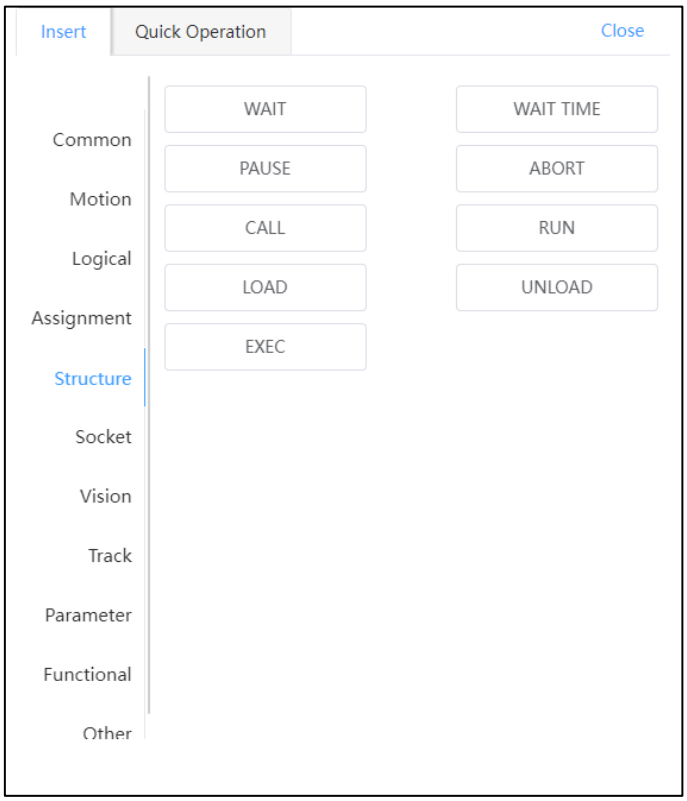


Fig. 4.25 Structure Instruction Interface

Click "WAIT" to enter the WAIT instruction parameter filling interface, as shown in Fig. 4.26.

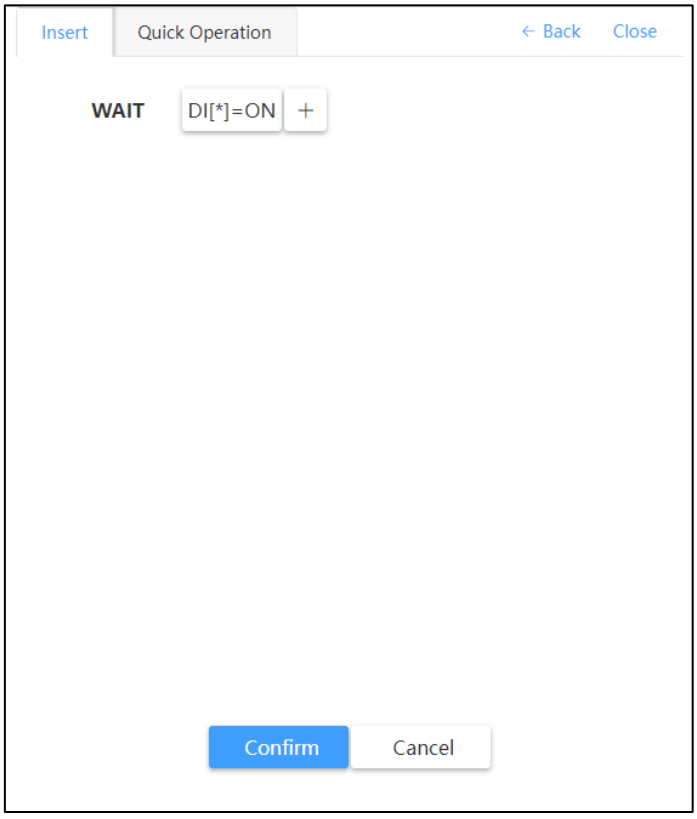
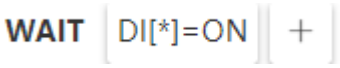


Fig. 4.26 WAIT Instruction Interface

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.

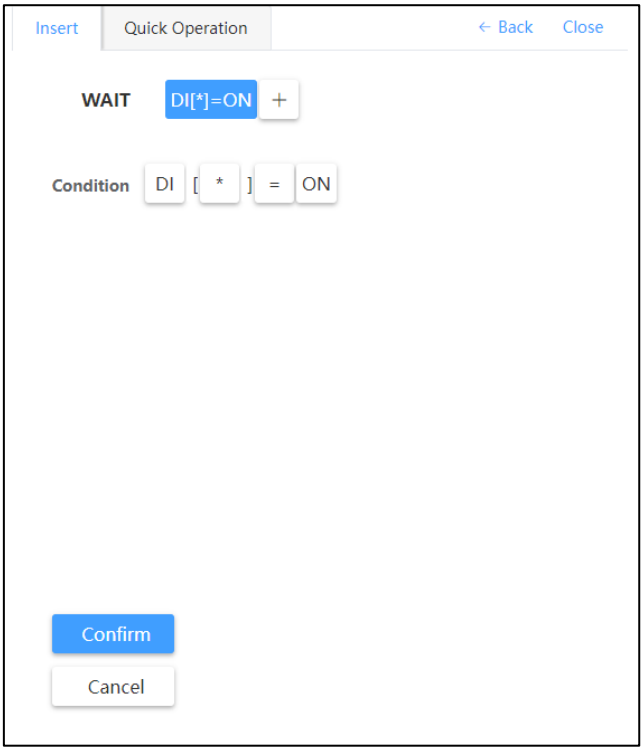
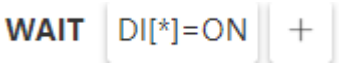


Fig. 4.27 WAIT Instruction Interface

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.

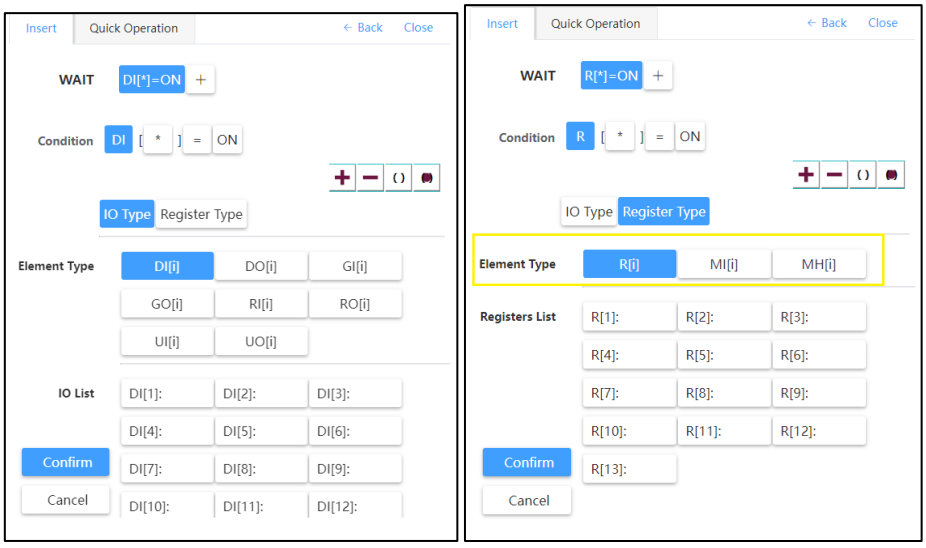
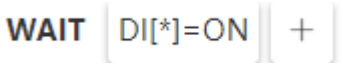


Fig. 4.28 WAIT Instruction Parameter Interface

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.

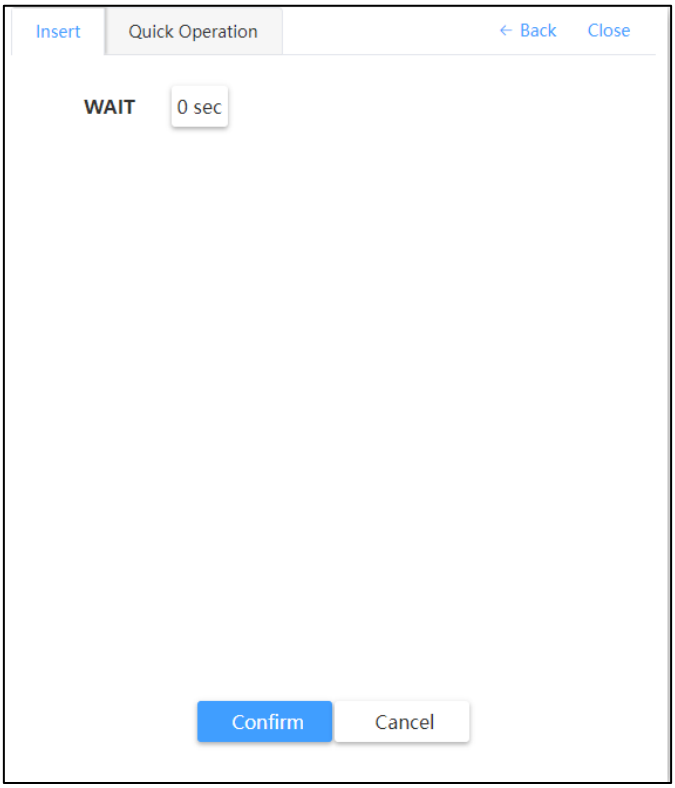


Fig. 4.29 WAIT TIME Instruction Parameter Filling Interface

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.

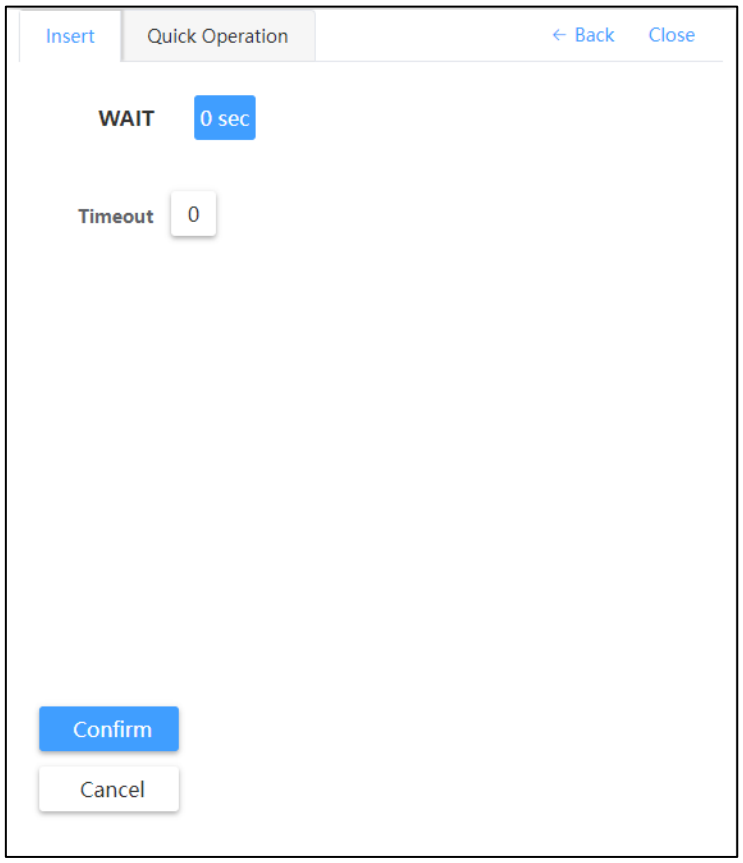


Fig. 4.30 WAIT TIME Instruction Parameter Filling Interface

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.

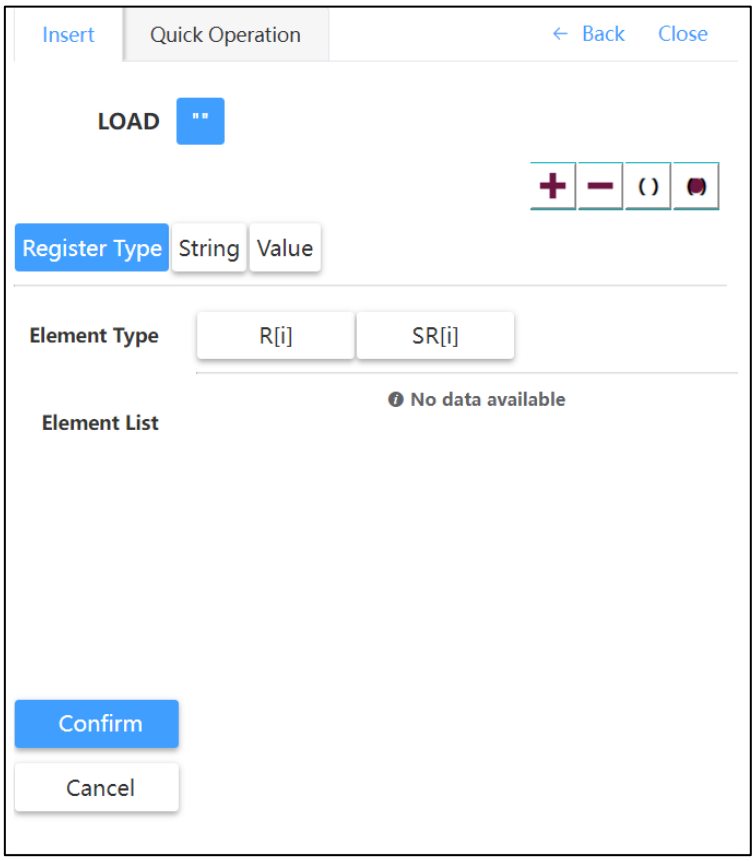


Fig. 4.31 LOAD Instruction Parameter Filling Interface

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.

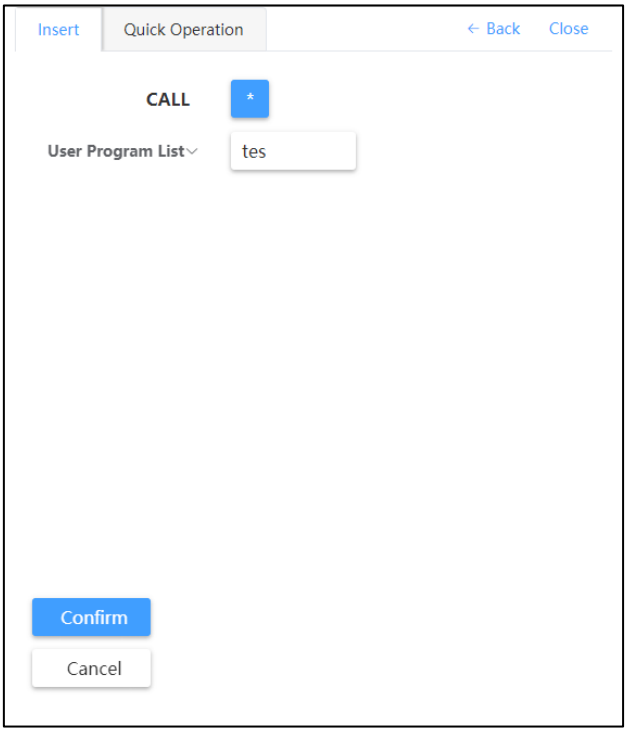


Fig. 4.32 CALL Instruction Parameter Filling Interface

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.

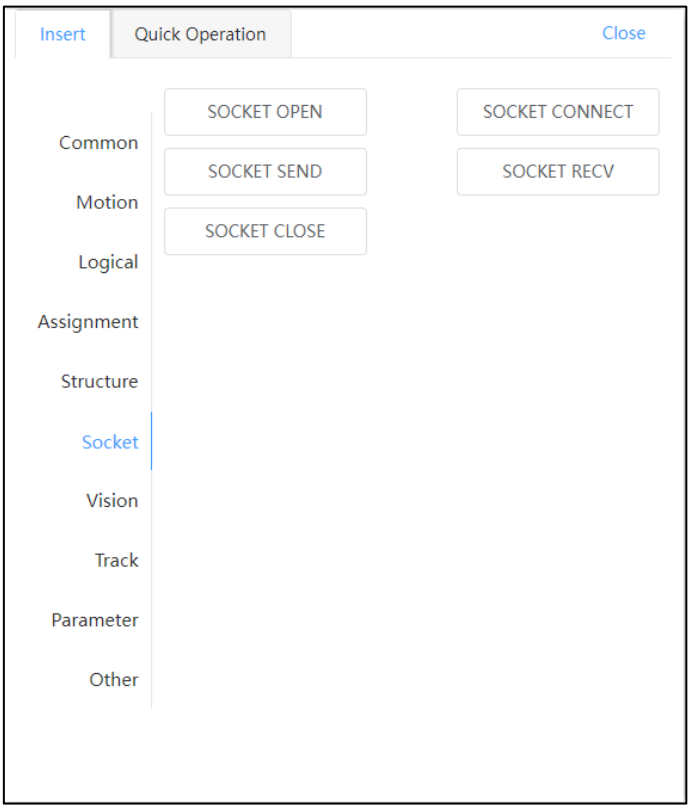


Fig. 4.33 SOCKET Instruction Interface

Click "SOCKET OPEN" to enter the SOCKET OPEN instruction parameter filling interface, as shown in Fig. 4.34.

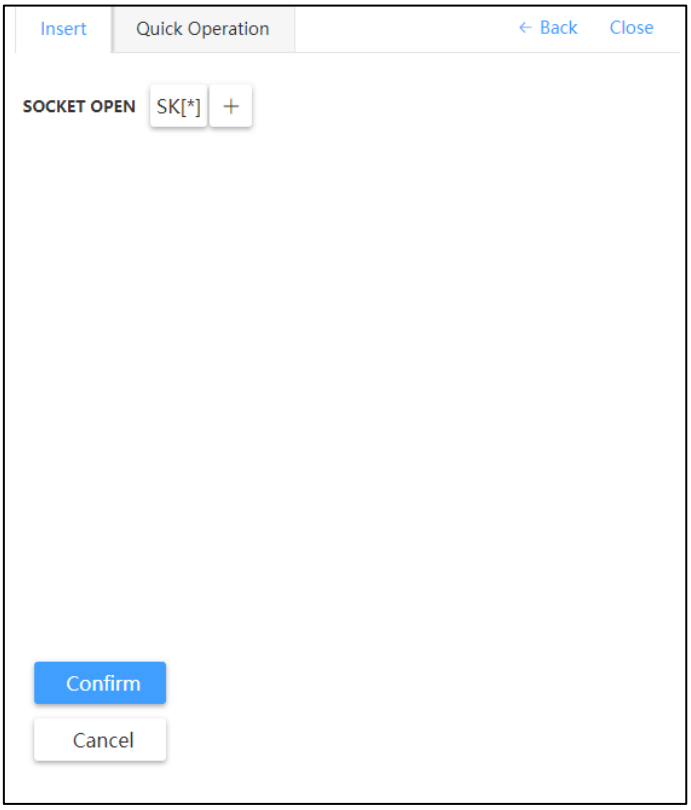


Fig. 4.34 SOCKET OPEN Instruction Parameter Filling Interface

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.

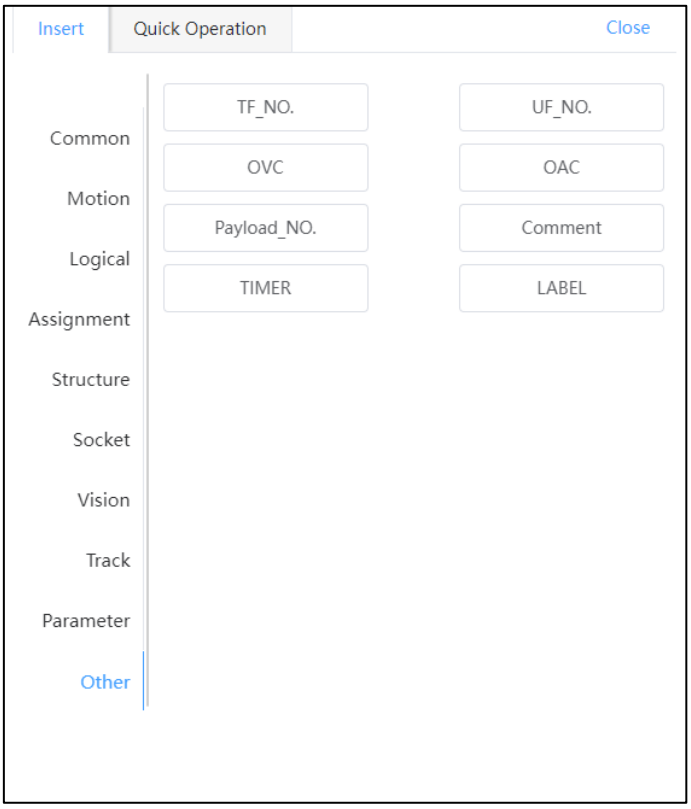


Fig. 4.35 Other Instruction Insertion Interface

Click "TF_NO" to enter the TF_NO instruction parameter filling interface, as shown in Fig. 4.36.

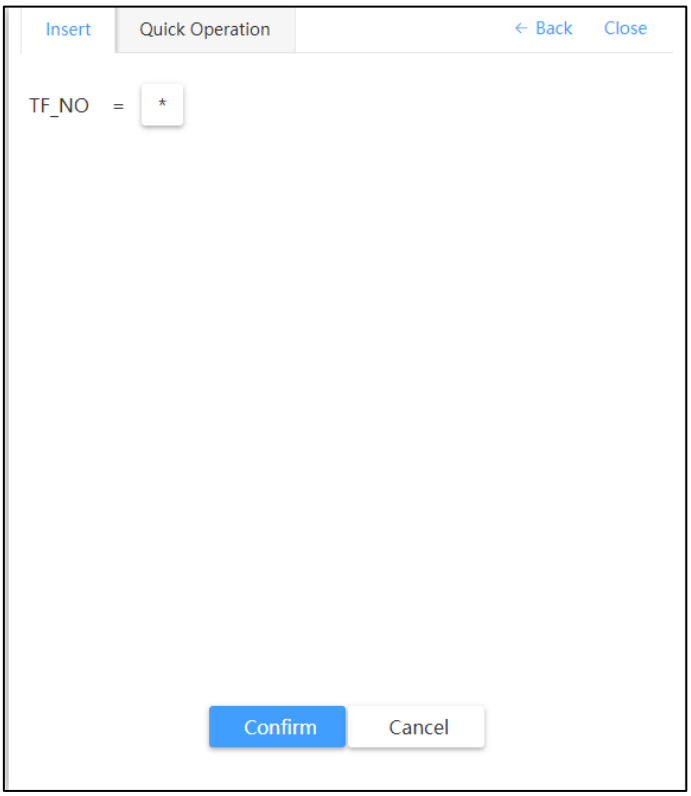


Fig. 4.36 TF_NO Instruction Parameter Filling Interface

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.

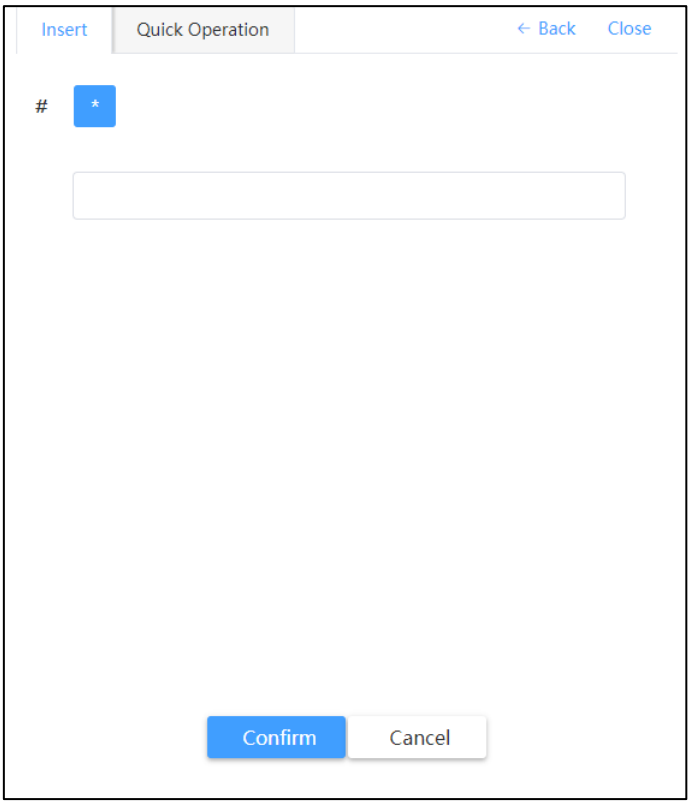


Fig. 4.37 Comment Instruction Parameter Filling Interface

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.

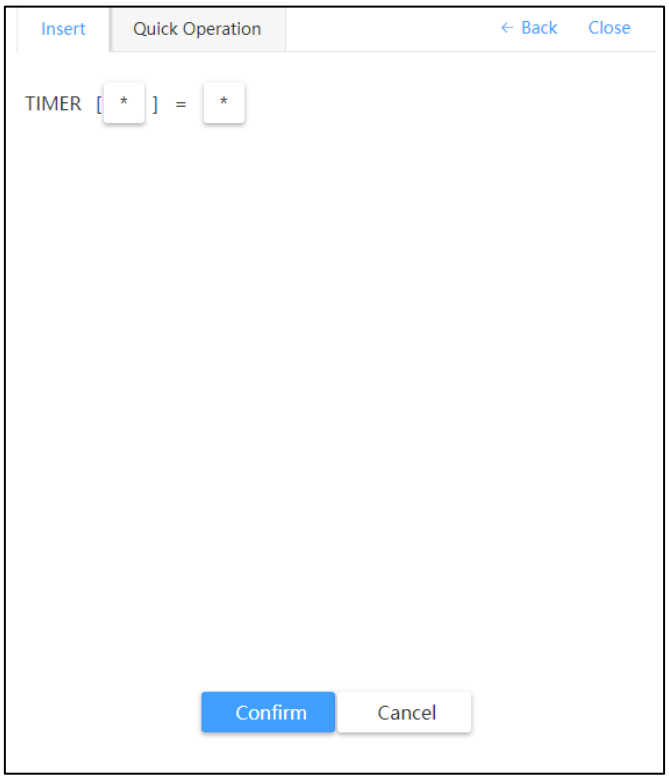


Fig. 4.38 TIMER Instruction Parameter Filling Interface

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.

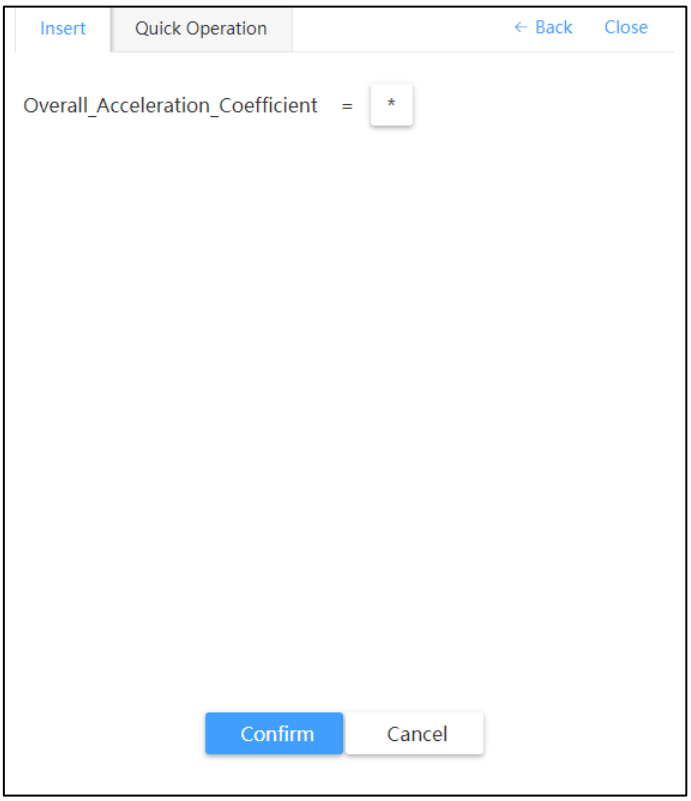


Fig. 4.39 OAC Instruction Parameter Filling Interface

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.

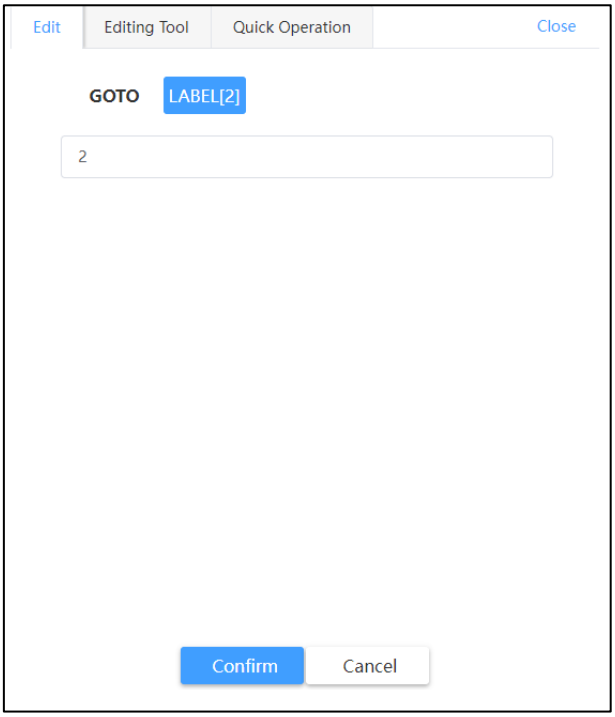


Fig. 4.41 GOTO Instruction Parameter Filling Interface

Fill in LABEL ID in the red 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.

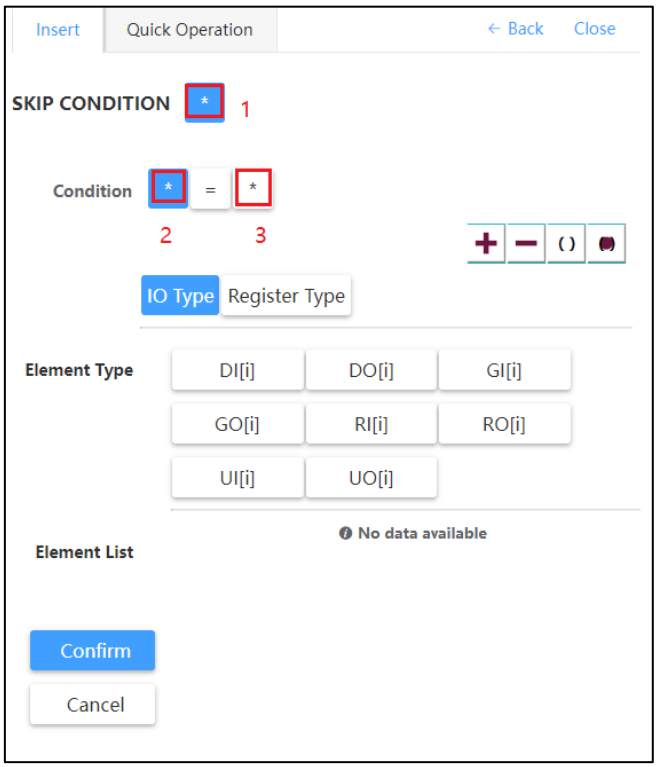


Fig. 4.42 SKIP CONDITION Instruction Parameter Filling Interface

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.

4.2 PROGRAM SETTING

Click "Menu Button" → "Program" to enter the interface as shown in Fig. 4.45. There are two settings: "Special Program Settings" and "Program Launch Mode Settings".

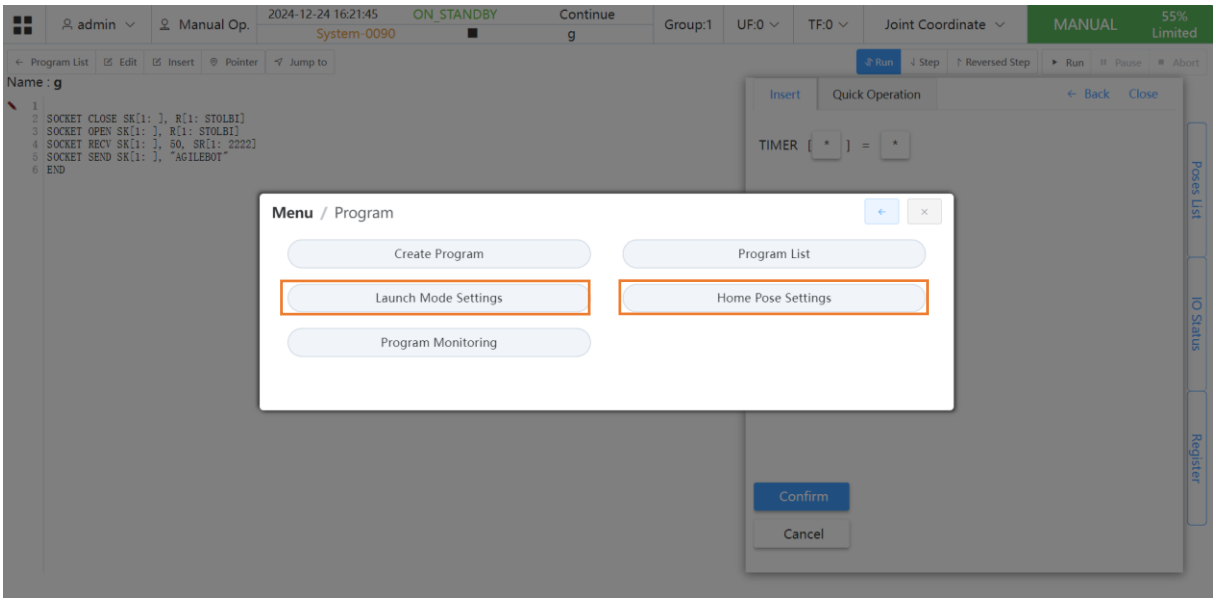


Fig. 4.45 Program Menu Window

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”.

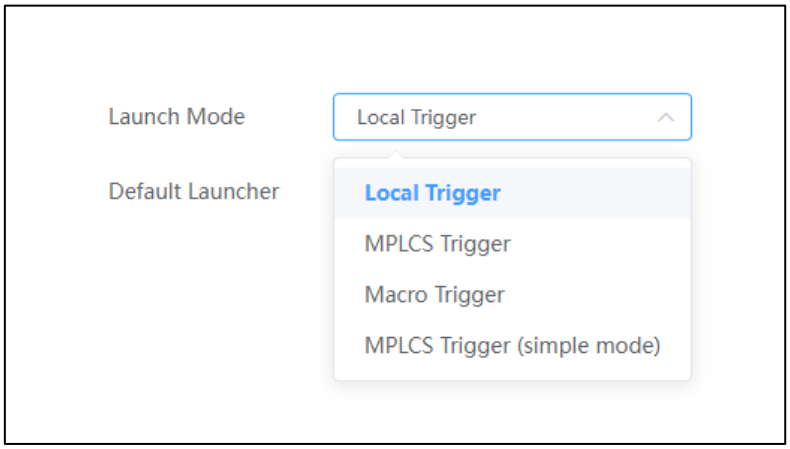


Fig. 4.47 Program Launch Mode Setting Interface

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:

1. When "MPLCS" is enabled, the system receives the Selection Strobe signal (UI[6]) from the upper computer to maintain a high level.
2. 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.
3. 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.
4. 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).
5. 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).
6. It triggers the main program if found. Meanwhile, it sends the UO[7] (MPLCS Start Done) to the upper computer.
7. Then, the upper computer turns off UI[6] and UI[8]-UI[13] signals.
8. 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:

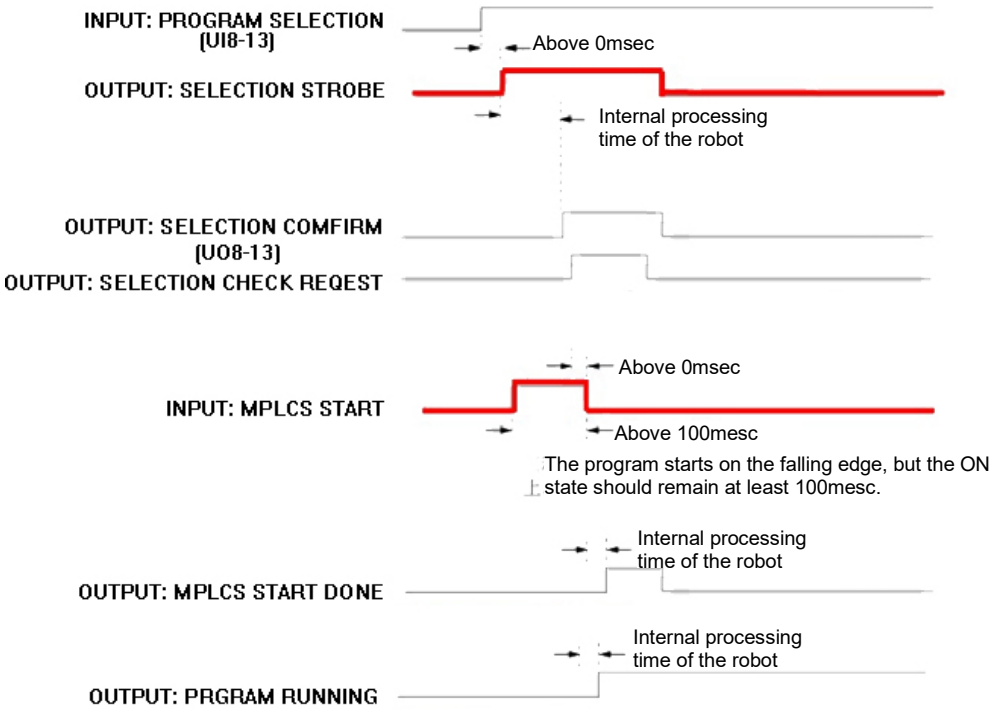


Fig. 4.48 MPLCS Timing Diagram

“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.

Macro Trigger

Macro Trigger is to trigger a macro program through pre-configured I/O, as detailed in Chapter 4.2.1.

Preconditions for Macro 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 “automatic program launch mode” is set as Macro Trigger.

Macro Trigger is applicable to macro programs.

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.

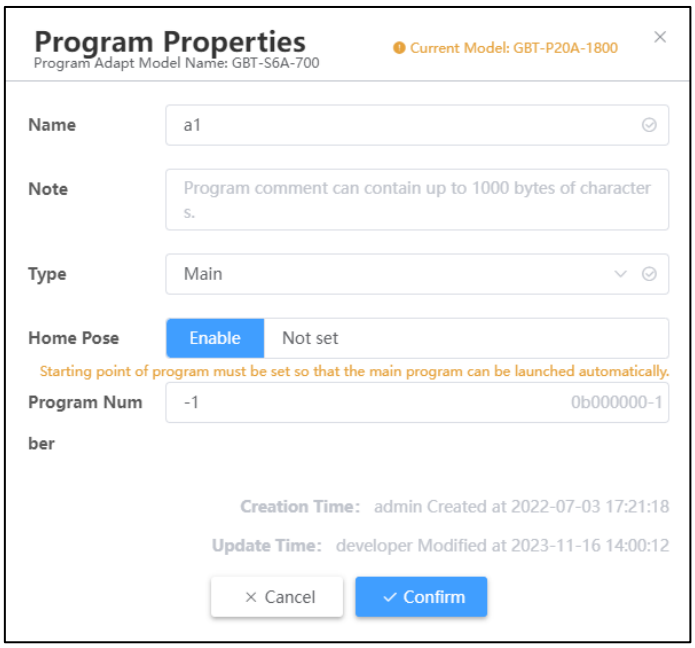


Fig. 4.49 Program Property Window

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 running mode switch must be in the auto mode.

4.3 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.

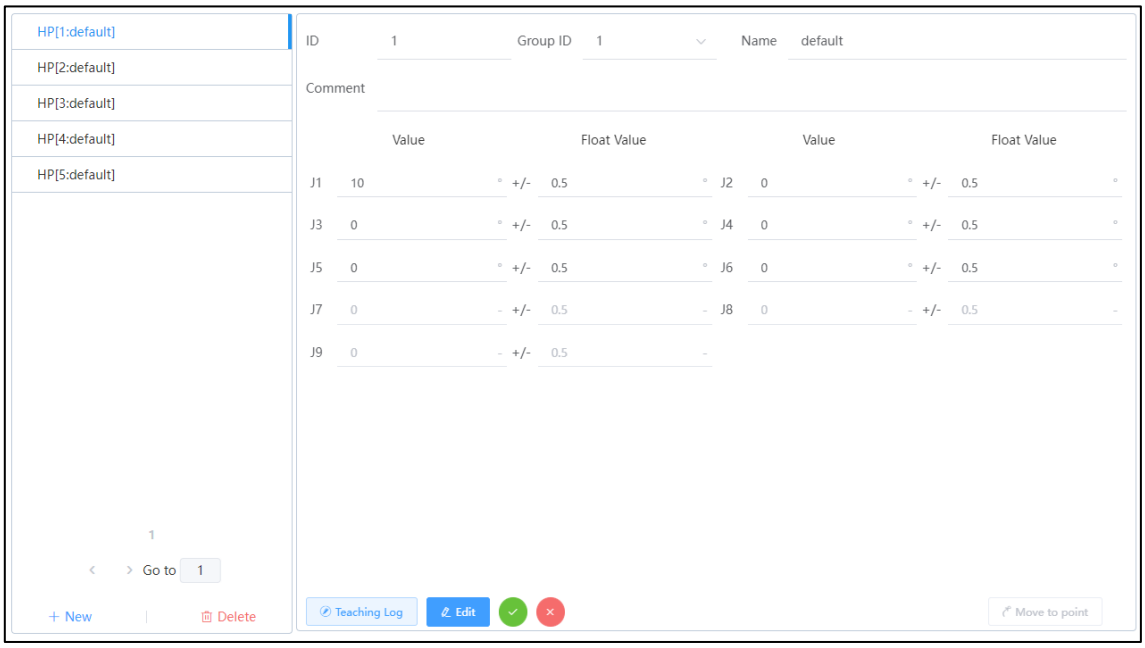


Fig. 4.50 Program Home Point Window

4.4 EXECUTE PROGRAMS

4.4.1 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 mode switch to the manual mode (manua or limit manual).
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)

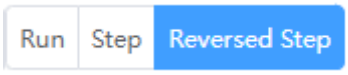


Fig. 4.51 Selection of Program Execution Modes

4. Select the statement from which line to execute the program. The selected statement is highlighted as shown in the following figure.

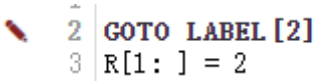


Fig. 4.52 Program Window

5. Click the "Arrow to Cursor". Execution will start from the selected line when the program is triggered.
6. Press the "Reset" button on the wired handle to clear alarm and other information and activate the robot (observe if the status lamp on the robot joint turns green).
7. Press the Start button to run the program.

4.4.2 Trigger 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:

1. 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.)
2. Set the mode switch to the auto mode (A).

3. Press the "Reset" button on the wired handle or send a “Restart” UI signal to the controller to clear alarm and other information and activate the robot (observe if the status lamp on the robot joint turns green).

4. Start the corresponding program according to the set program trigger mode (directly press the 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).

4.4.3 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, causing 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.

4.4.4 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.

4.5 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 Run to switch between "Run" and "Pause". “Run” means to execute the program and “Pause” means to pause the program. Click Abort to stop the program.

Status:

- Running

- Pause
- Abort

> Caution
>
> For Python script programs, Pause is invalid, while Abort is valid.

5 STATUS DISPLAY

5.1 REGISTER MANAGEMENT

The registers used by the Collaborative robot include number register (R), motion register (MR), pose register (PR), string register (SR), socket register and Modbus register.

5.1.1 Number register instruction

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.

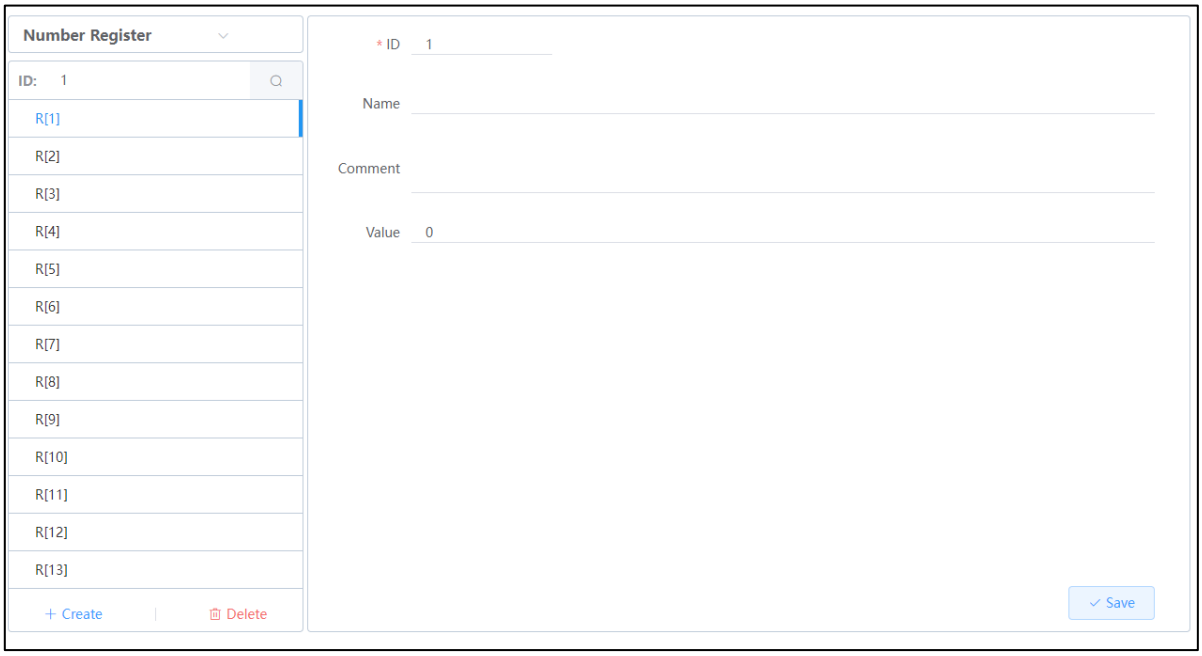


Fig. 5.1 Number Register Setting Interface

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 motion register screen.

Successively click "Menu Button" → "Data" → "Motion Register" to enter the motion register setting screen as shown in Fig. 5.2.

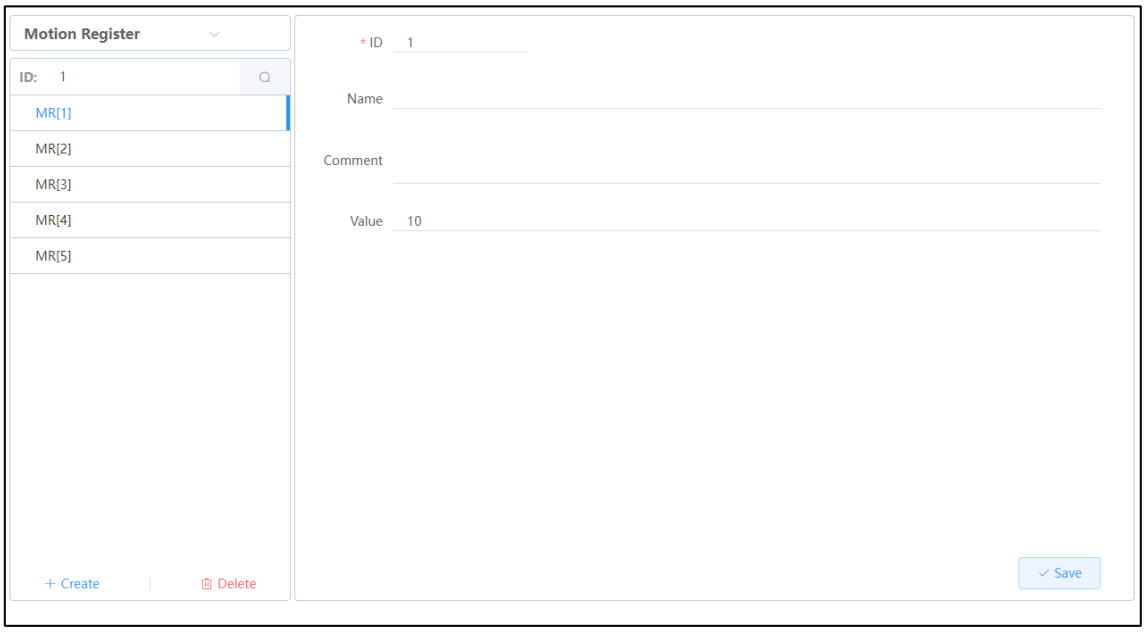


Fig. 5.2 Motion Register Setting Interface

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.

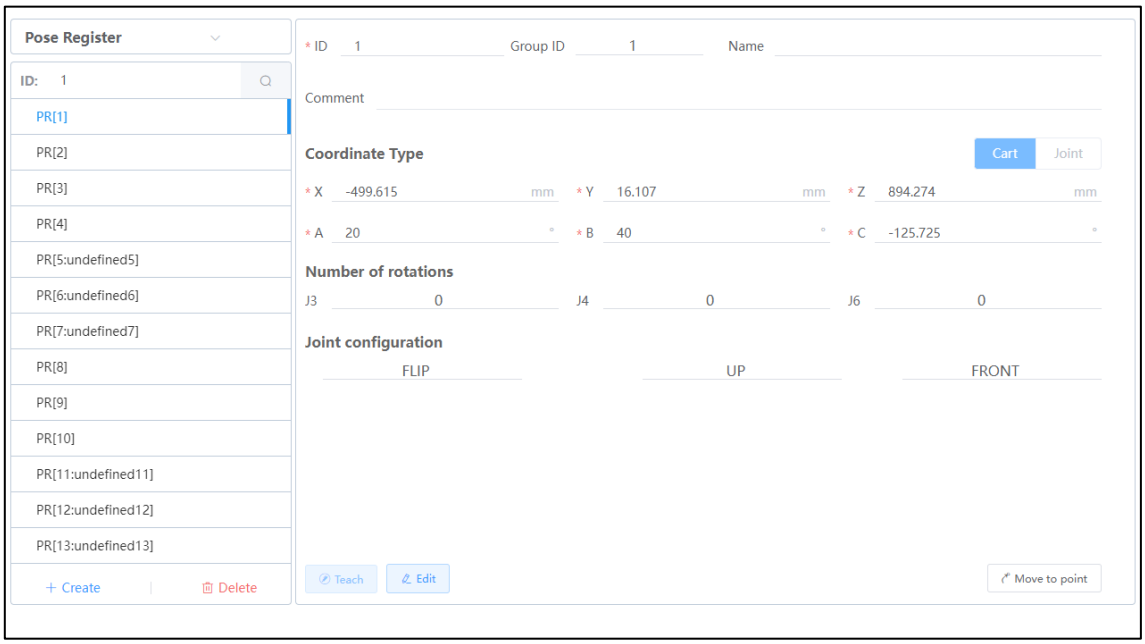


Fig. 5.3 Pose Register Setting Interface (Cartesian)

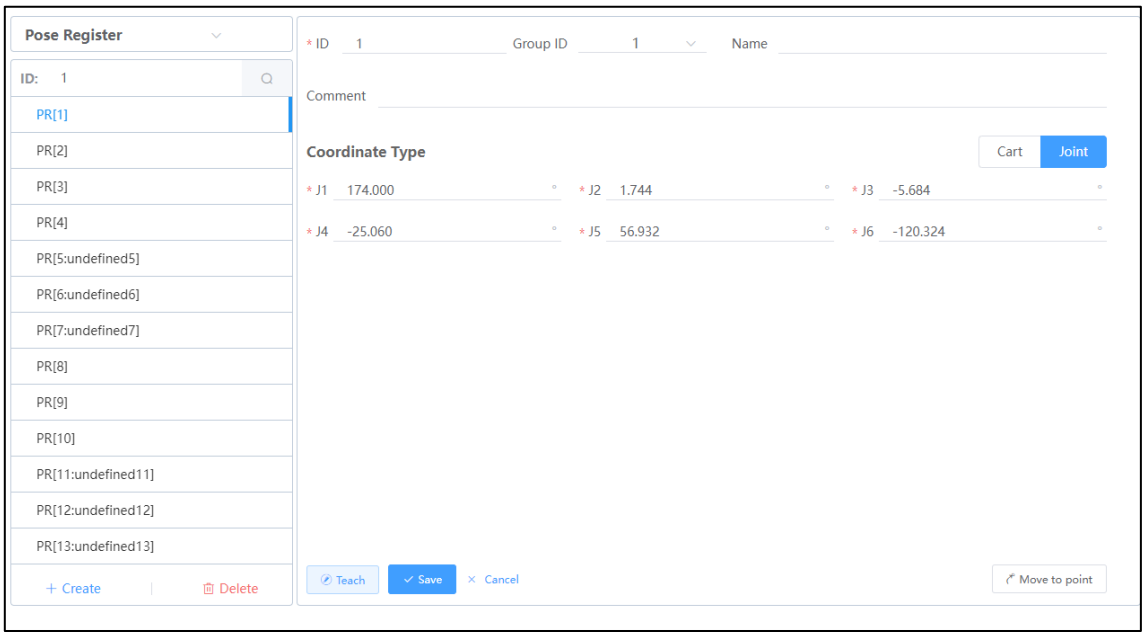
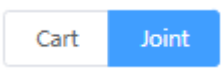


Fig. 5.4 Pose Register Setting Interface (Joint)

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.

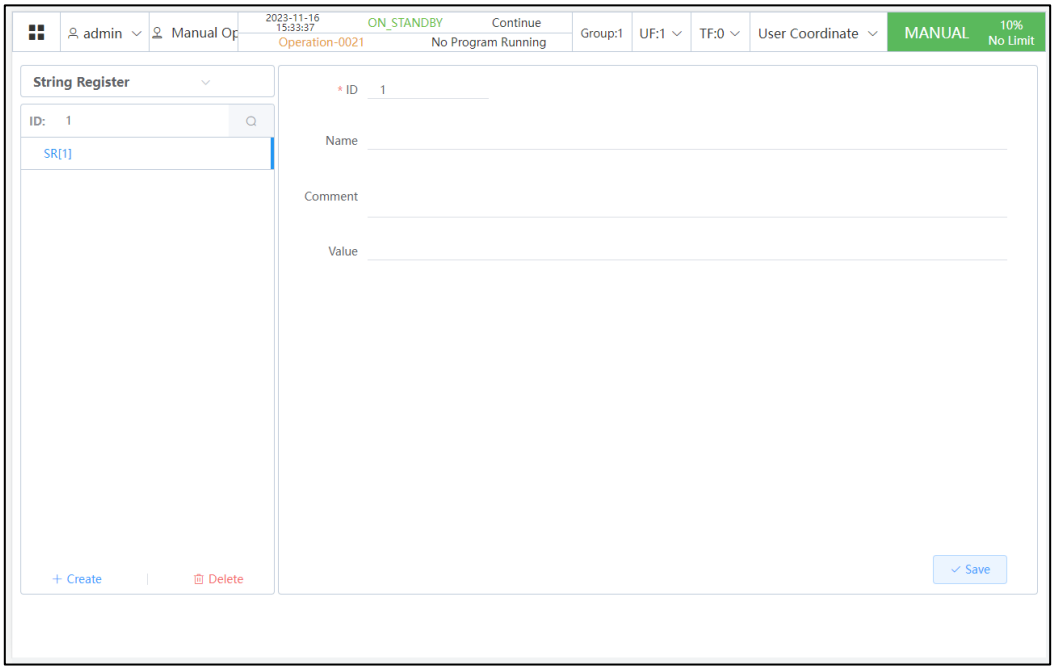


Fig. 5.5 String Register Setting Interface

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.

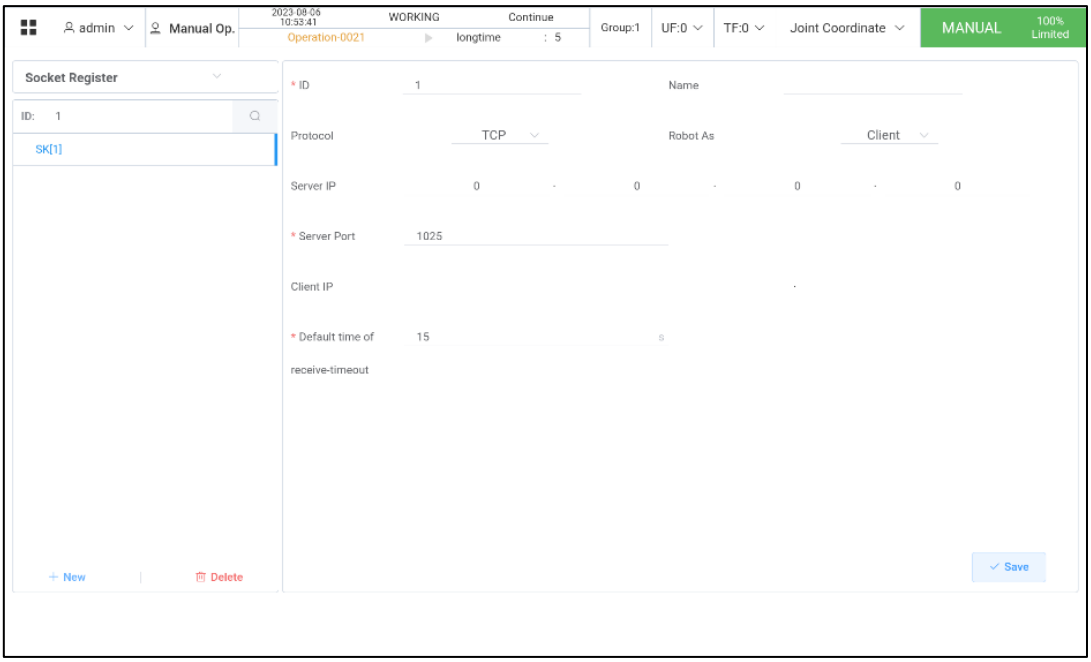


Fig. 5.6 Socket Register Setting Interface (Client)

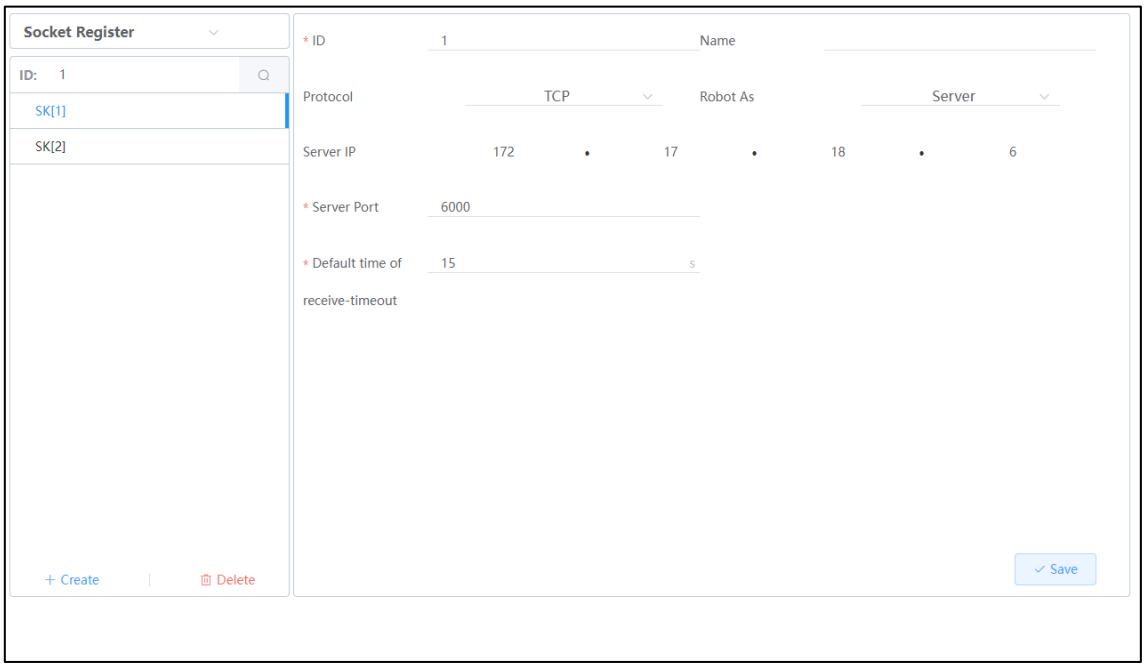


Fig. 5.7 Socket Register Setting Interface (Server)

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.

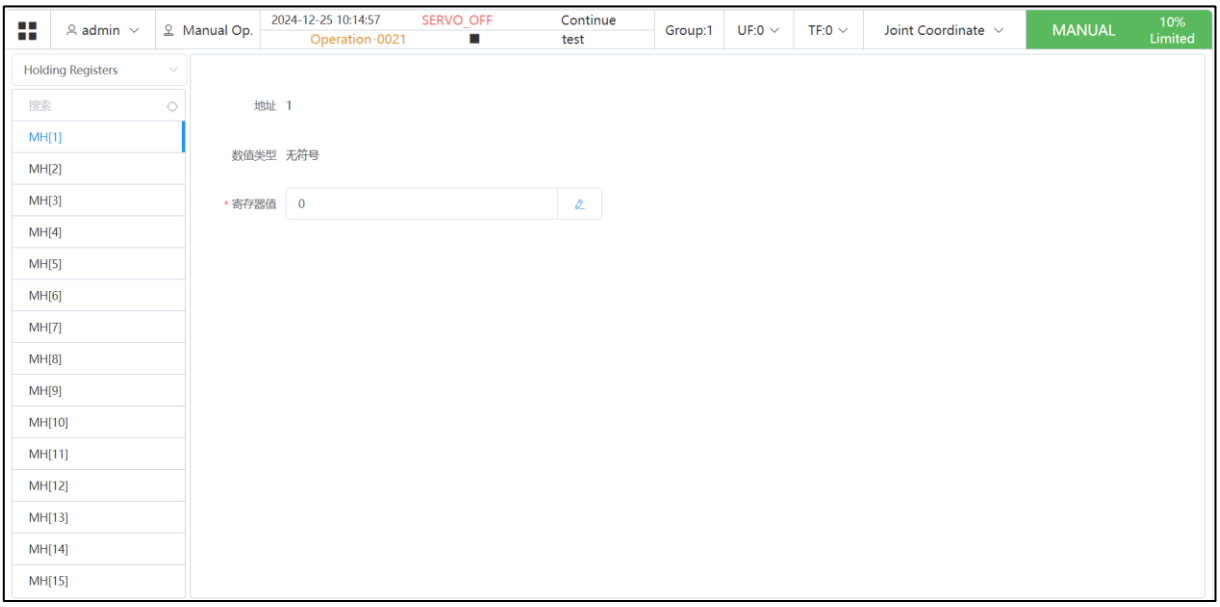


Fig. 5.8 Modbus Special Register Setting Screen

Click Holding Registers to switch between the monitoring interfaces of Input Registers and Holding Registers, as shown in Fig. 5.8.

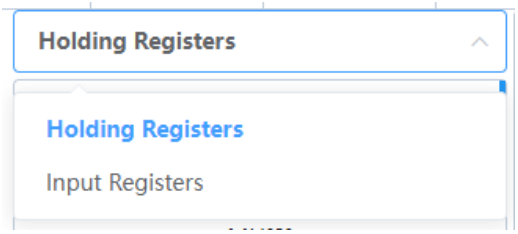


Fig. 5.8 Modbus Register Switching Window

I/O mapping and Modbus special registers

I/O mapping can be adopted to map Inputs/Coils in the Modbus special registers to DI/DO according to the following steps:

1. Successively click "Menu Button" → "Communication" → "Bus Configuration" to enter the interface shown in Fig. 5.9 and turn off the slave function (do not operate it if already turned off).

Fig. 5.9 Bus Configuration Window

2. Click "Function" to enter the interface shown in Fig. 5.10. Then, click "Edit" to configure the address and number of Inputs/Coils for the slave station. Maximum number of Inputs/Oils is 1024.

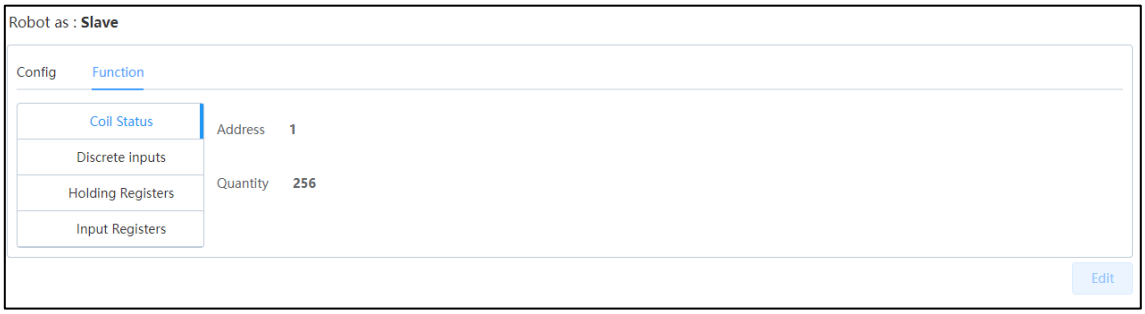


Fig. 5.10 Bus Register Signal Configuration Window

3. After configuration, click "Config" to return to the interface shown in Fig. 5.9 and activate the slave function.

4. Enter the I/O mapping screen to map the required IOs (Coil Status mapped to DI and Discrete inputs mapped to DO) (see Section 2.1.1 for specific operations)

5. Enter the I/O status screen to monitor and operate DI/DO.

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.

6

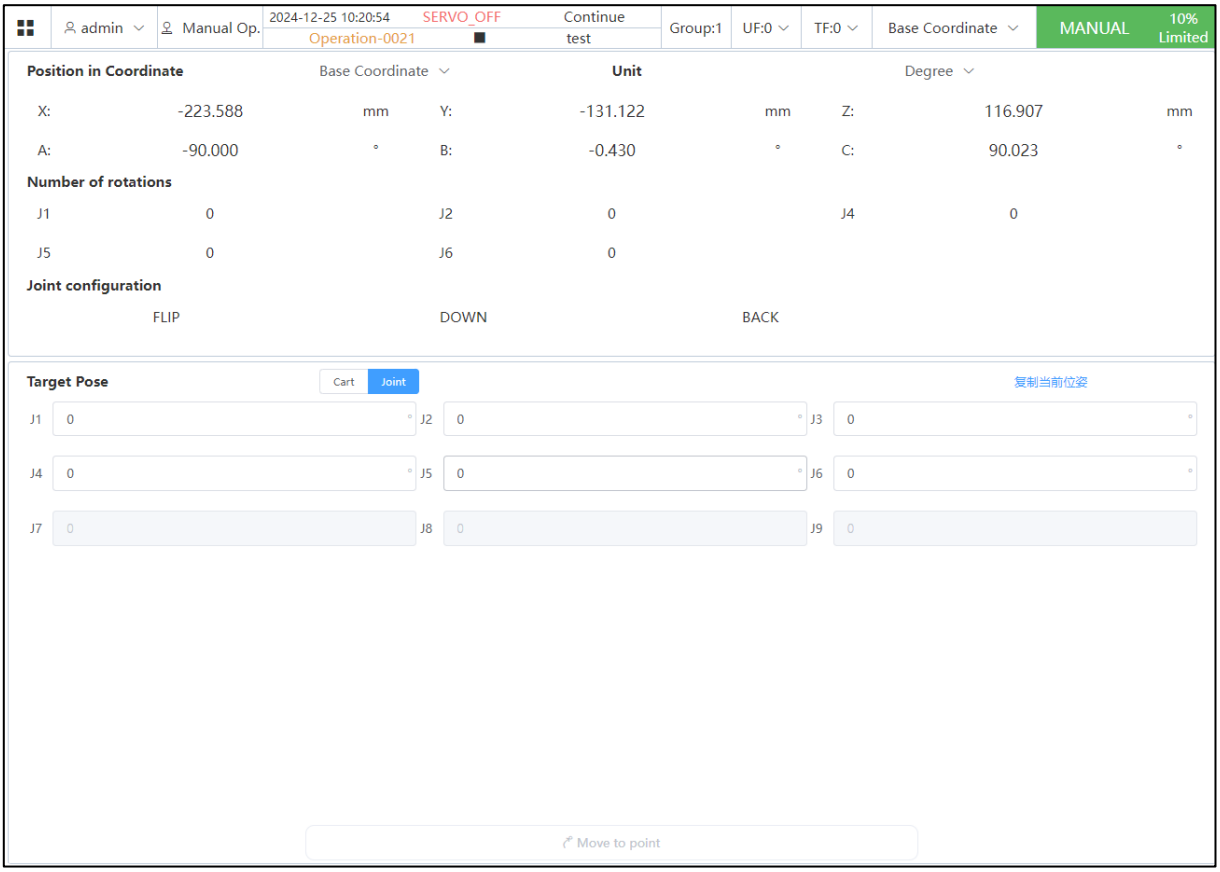


Fig. 5.11 Positions in Coordinate Interface (Cartesian)

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.

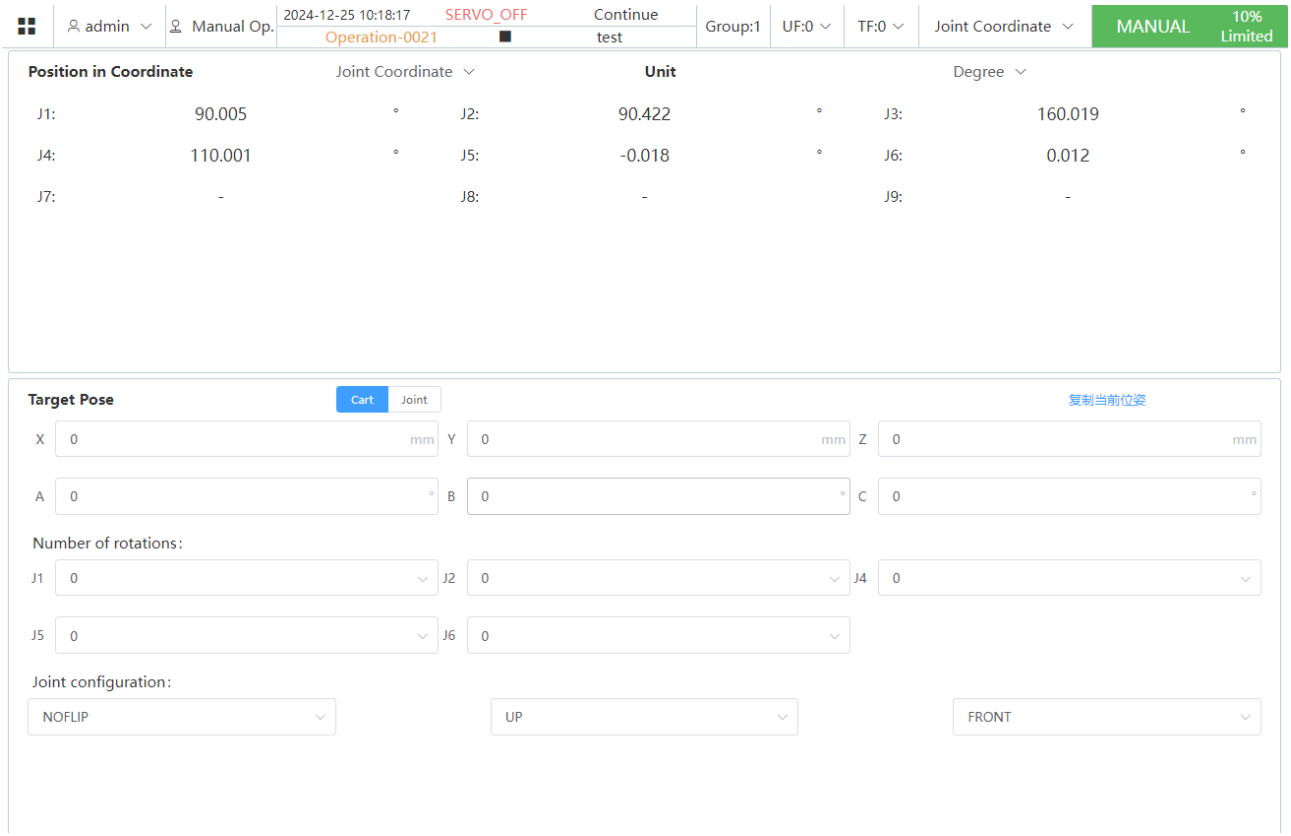
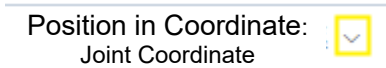


Fig. 5.12 Current Position Interface (Joint)

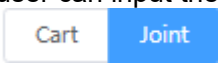
Click 

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).

6 ROBOT COMMUNICATION SETTING

Summary

Collaborative 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, Collaborative 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	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 Slave Station Setup Interface

Description of parameters:

- Channel: Controller/TCP
- IP Address: The IP address of the slave (robot)
- Port Number: The port used for communication with the master station
- Slave Response Delay: The latest response time specified for communication between the master and slave stations
- Data Type: Signed/Unsigned

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: The address and range are from 0 to 65,536.
   - Discrete Input Register: The address and range are from 0 to 65,536.
   - Holding Register Register: The address is 0 and the number is unlimited.
   - Input Register Register: The address is 0 and the number is unlimited.
4. After setting the register address and number, click "Config" 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 methods and practical cases about Modbus, please refer to the "GBT Modbus TCP Function Manual".

6.2 Master setting steps

Successively click "Menu Button" → "Communication" → "Bus Configuration" to enter the Modbus slave. configuration screen as shown in Fig. 6.1.

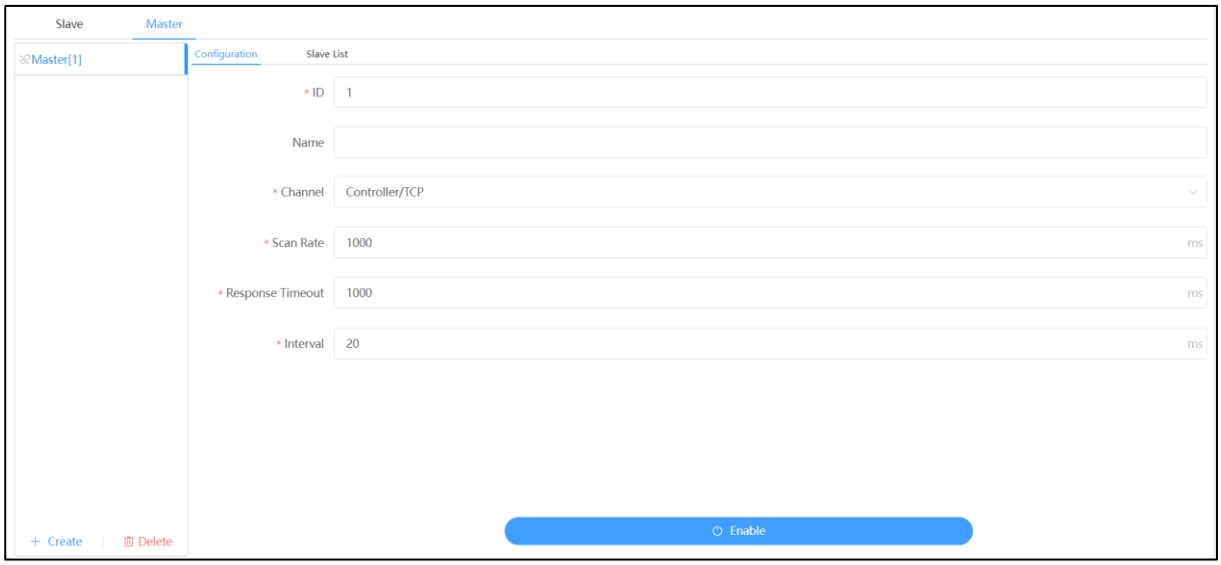


图 6.3 Master Station Setup Interface

Description of parameters:

- ID：ModbusID
- Name: Up to 128 bytes.
- Communication Channel: TCP, control cabinet RS485, or wrist RS485 are available.
- Communication Scan Rate: The polling cycle of the master station, with a default of 1,000 ms.
- Slave Station Response Time - out: The maximum response time specified when the master station communicates with the slave station.
- Scan Interval: The cycle of the master station polling each slave station, with a default of 20 ms.

> Caution
>
> Before the master station is enabled, it should be ensured that the communication with each slave station is normal. Otherwise, the enabling process will fail.

If TCP is used, the Controller/TCP mode should be selected, as shown in Figure 6.4.

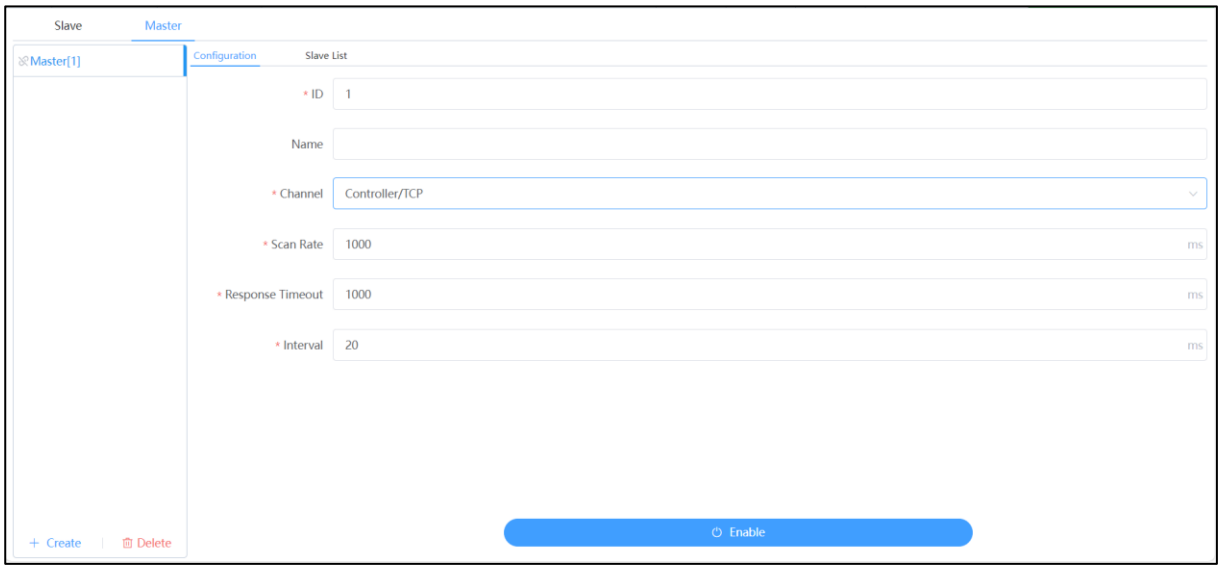


图 6.4 Controller/TCP Mode

If a TCP - to - RTU gateway is used, the Controller/TCP2RS485 mode should be selected, 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 should ensure the use of the transparent transmission mode.

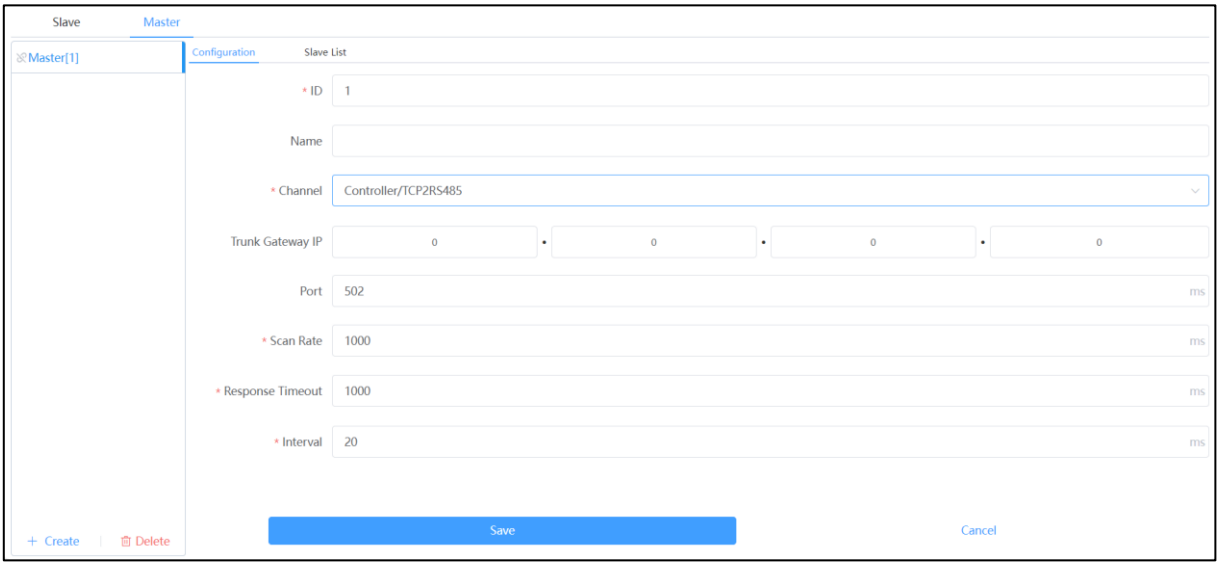


图 6.5 Controller/TCP2RS485 Mode

If TAI [1] and TAI [2] are multiplexed for RS485 communication, the Wrist/RS485_0 mode should be selected, as shown in Figure 6.6.

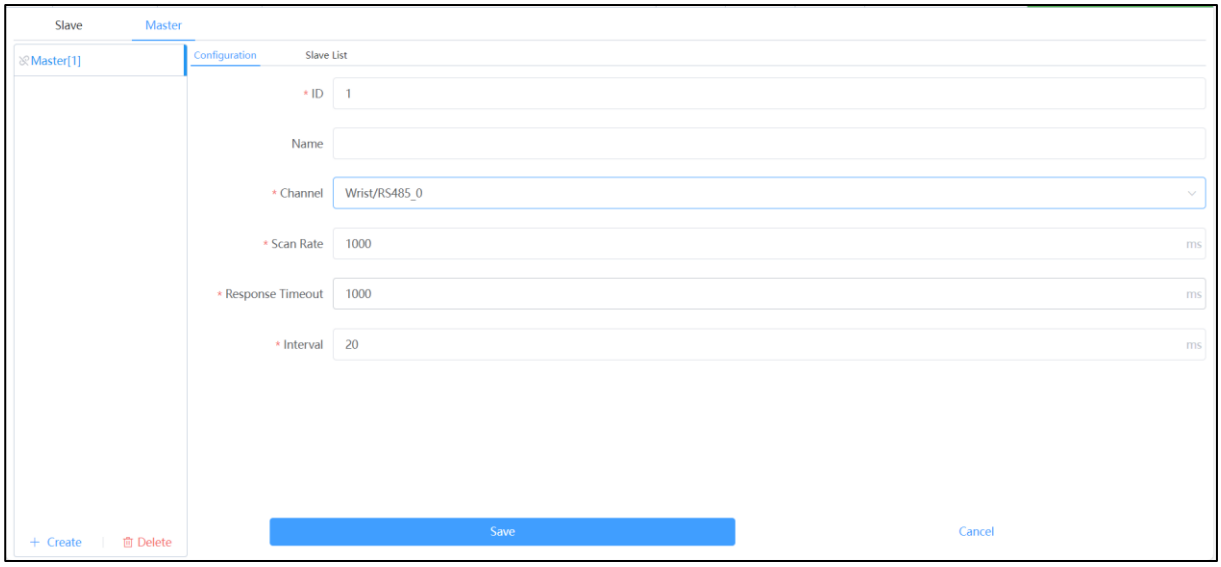


图 6.6 Wrist/RS485_0 Mode

If TDO[1] and TDO[2] are multiplexed for RS485 communication, the Wrist/RS485_1 mode should be selected, as shown in Figure 6.7.

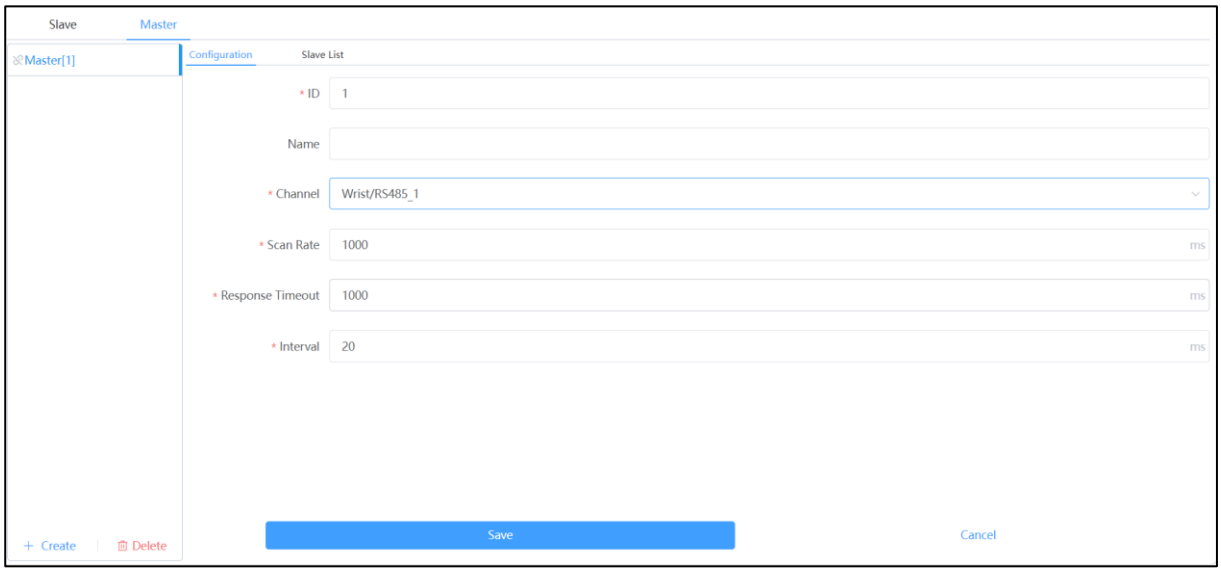


图 6.7 Wrist/RS485_1 Mode

In the slave station list of the master station, Modbus slave stations connected to the master station can be created. Up to 32 slave stations can be established. The following is the slave station list under the Controller/TCP mode:

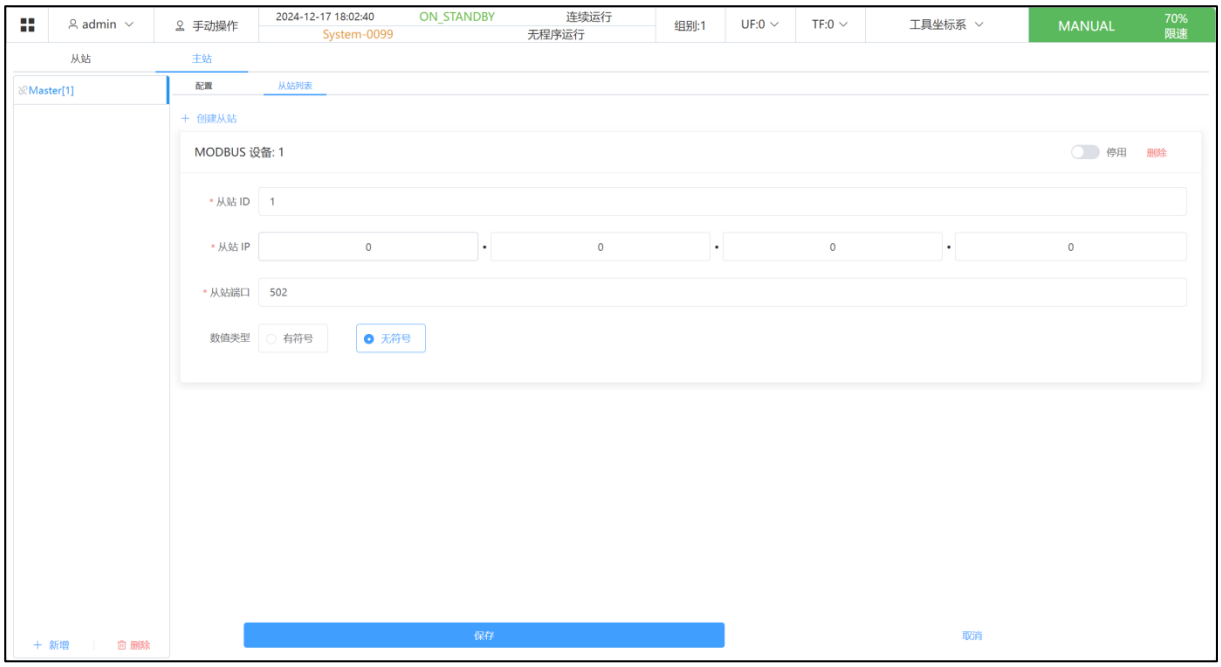


图 6.6 主站的从站列表

Description of parameters:

- Slave Station ID: The ID of the Modbus slave station.
- Slave Station IP: Up to 128 bytes.
- Slave Station Port: The port used for communication with the master station.
- Data Type: Signed/Unsigned.从站 ID：Modbus 从站 ID

For more detailed configuration, usage methods and practical cases about Modbus, please refer to the "GBT 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 BACKUP AND LOAD OBJECTS

Data

- User's program files
- Data files
- System setting files
- MCCP files
- Logs

Software

- Robot application software
- Robot motion control software

> Caution
>
> Exclude operating system and underlying drivers.

Selection of backup method:

A. Single file backup (file management interface): Choose a single file for backup and save it to a folder in the designated USB drive path.

B. 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.

C. All backup (file management interface): Back up all files and save them to a folder in the designated USB drive path.

D. 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:

A. Single file loading (file management interface): Designate a path and select a single file for loading.

B. Single type loading (file management interface): Designate a path, check a file type and load all files of that type in this directory.

C. All loading (file management interface): Choose a folder and load all recognizable robot files in this directory.

D. Mirror loading (file management interface): Choose the mirror zip package folder, select the mirror loading mode, decrypt and decompress the zip package and replace all files and installed software in the controller (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.

7.2 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 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 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 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.

7.3 USER'S OPERATION MODE

Backup:

1. 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.

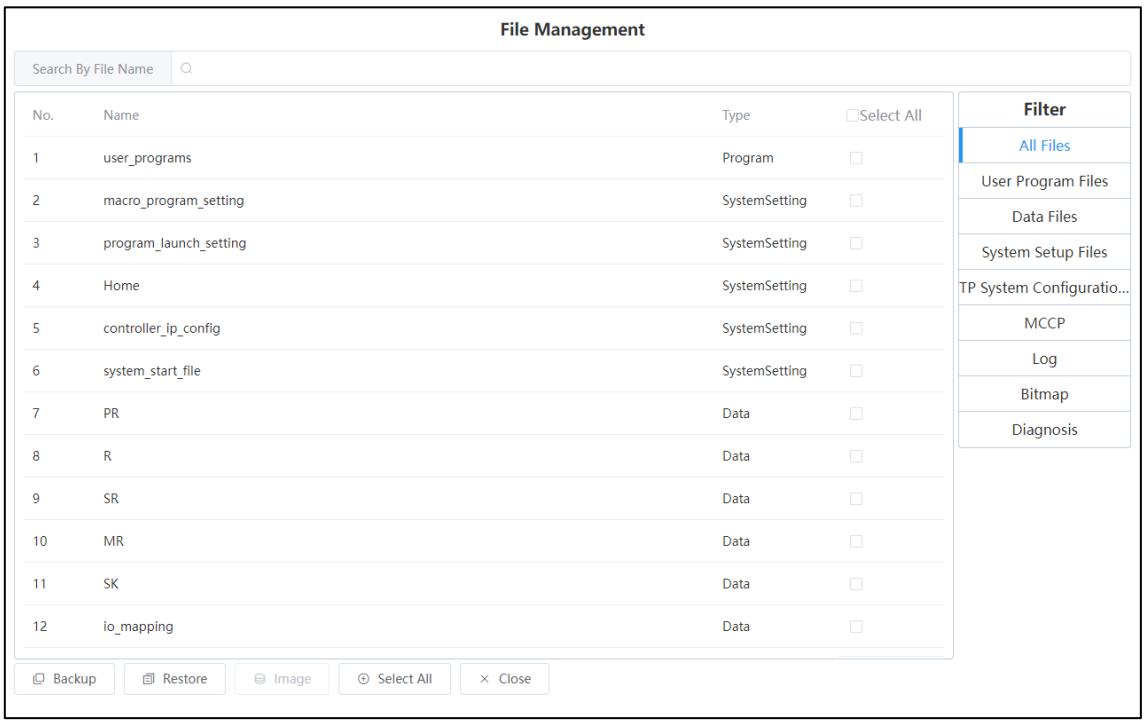


Fig. 7.1 File Management Interface

2. Click the square behind the target file, select the target file and then click the "Backup" button.

3. Select the path and specific operating directory, as shown in Fig. 7.2 and 7.3. If necessary, create or edit folders.

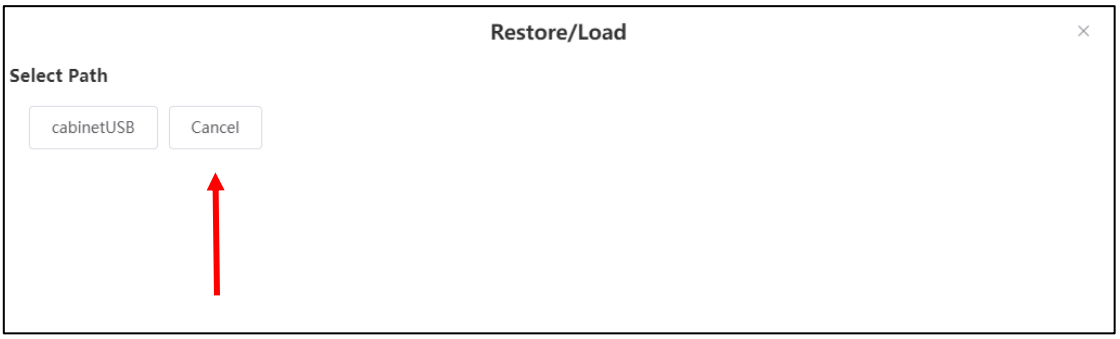


Fig. 7.2 Path Selection Interface

Fig. 7.3 File Directory Interface

4. Select the destination folder for backup and click "Start Backup".

5. Wait for the system to complete the execution.

Mirror:

1. Switch the robot to the SERVO_OFF state. Click "Menu Button" → "Management" → "File Management" → "Backup" and open the file management interface.

2. Click on the "Mirror" button.

3. Select the path and specific operating directory. If necessary, create or edit folders.

4. Select the destination folder for backup and click "Start Backup".

5. Wait for the system to complete the execution.

Restore/Load:

1. Switch the robot to the SERVO_OFF state. Click "Menu Button" → "Management" → "File Management" → "Restore/Load" and open the file management interface.

2. Select the path and specific operating directory.

3. Select the files to load and click “Start Restore”; wait for the system to complete the execution.

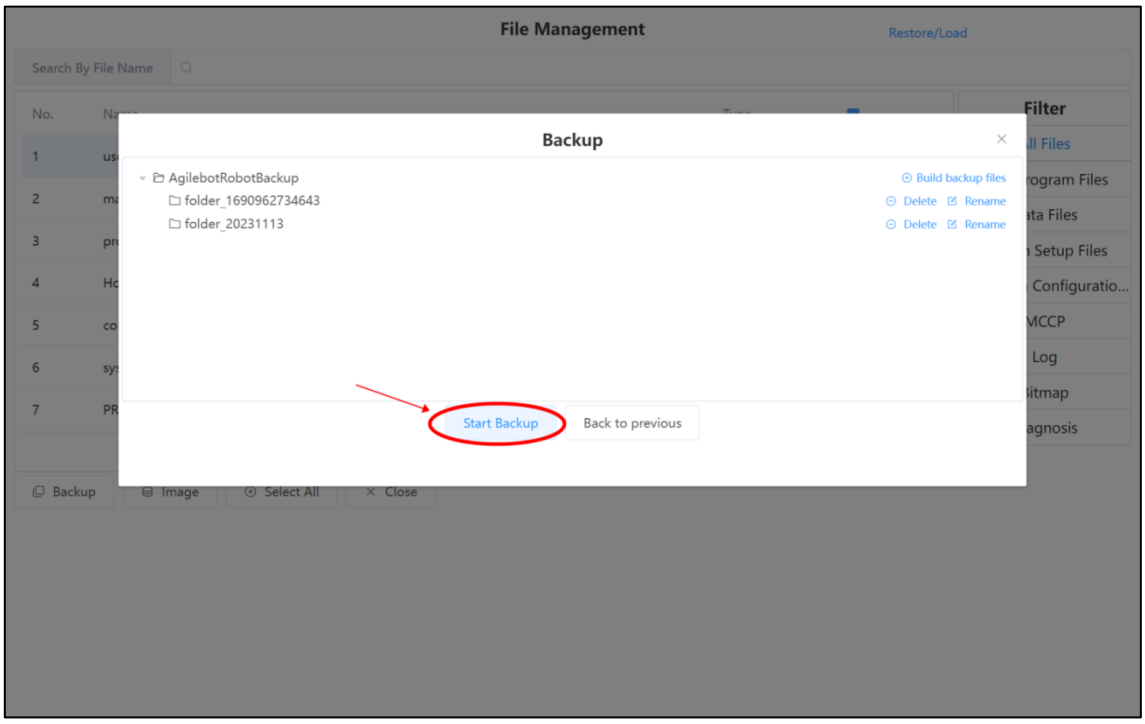


Fig. 7.4 Restore File Directory Interface

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 robot after mirror restore.
- No other software operations are allowed during loading or backup.

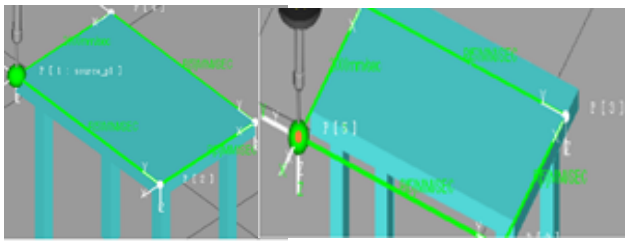
8 PRACTICAL FUNCTIONS

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

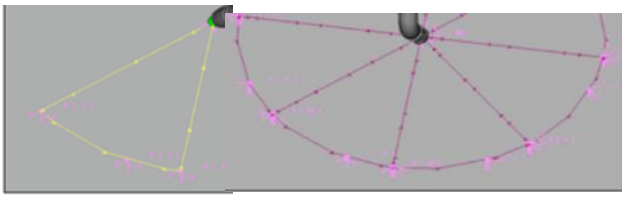


Fig. 8.2 Regular Repeating of Pose Data

Steps:

Click "Menu Button" → "Application" → "Program Offset" to enter the function setting interface as shown in Fig. 8.3;

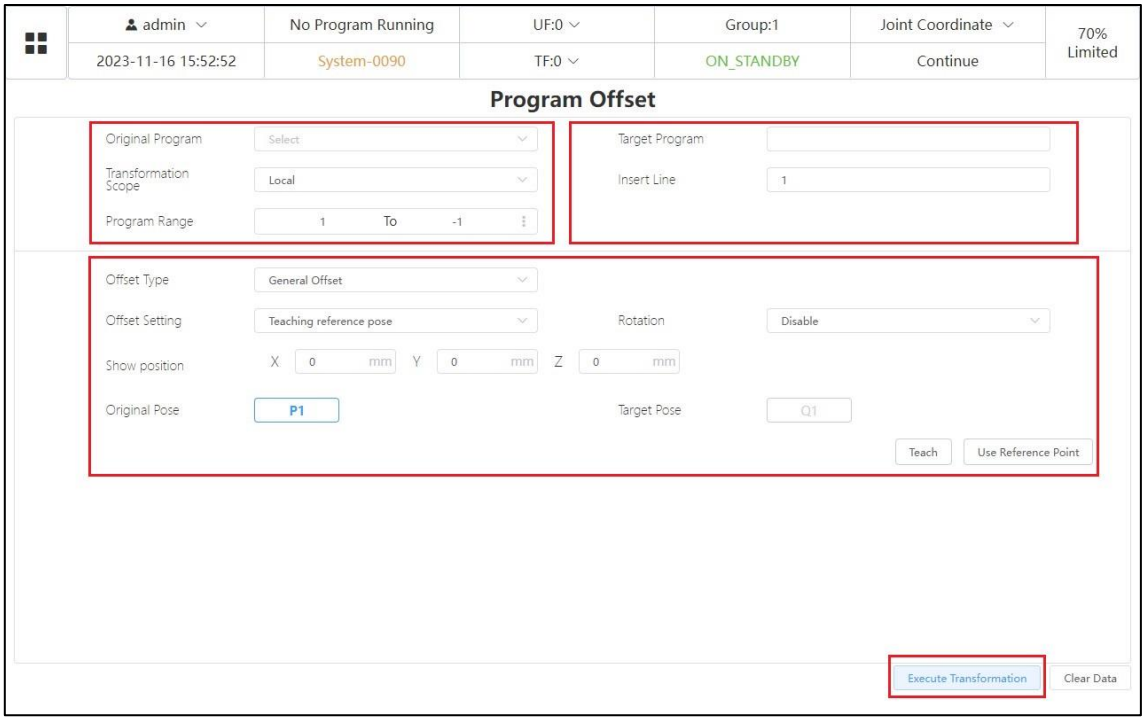


Fig. 8.3 Program Offset Setting Interface

2. 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";

3. 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";

4. Offset Type: Choose General Offset, Mirror Offset or Circular Array Offset in the "Offset Type".

5. After setting, click "Do Transformation" to generate a new point.

6. 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

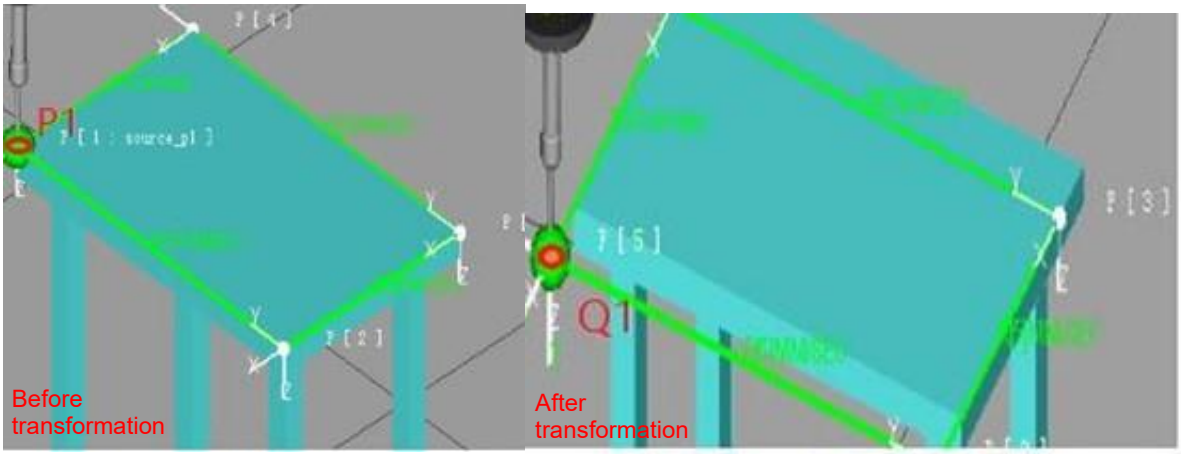


Fig. 8.4 General Offset

General offset with rotation

- Reference poses: 3 source poses and 3 target poses

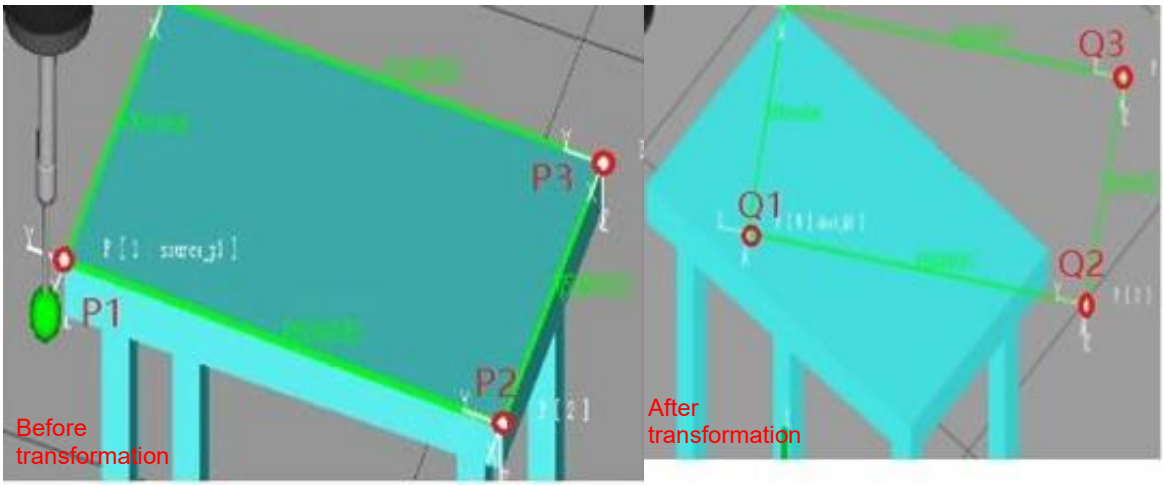


Fig. 8.5 General Offset with Rotation

8.1.2 Mirror offset

Mirror offset

- Reference poses: 1 source pose and 1 target pose

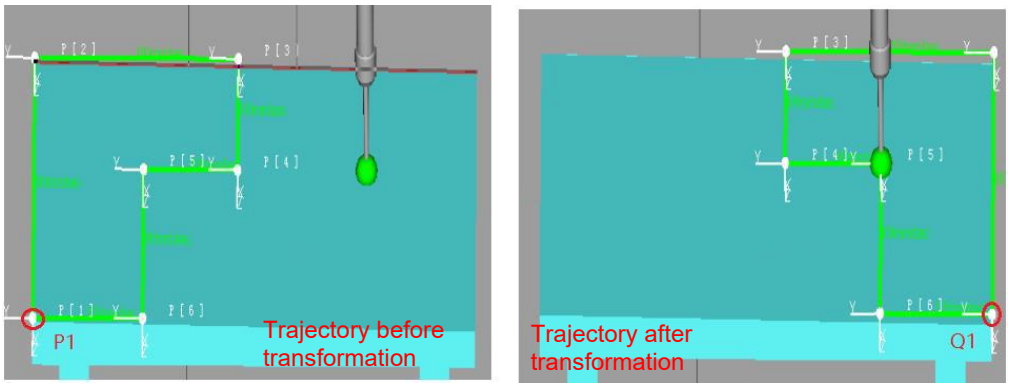


Fig. 8.6 Mirror Offset

Mirror offset with rotation

- Reference poses: 3 source poses and 3 target poses

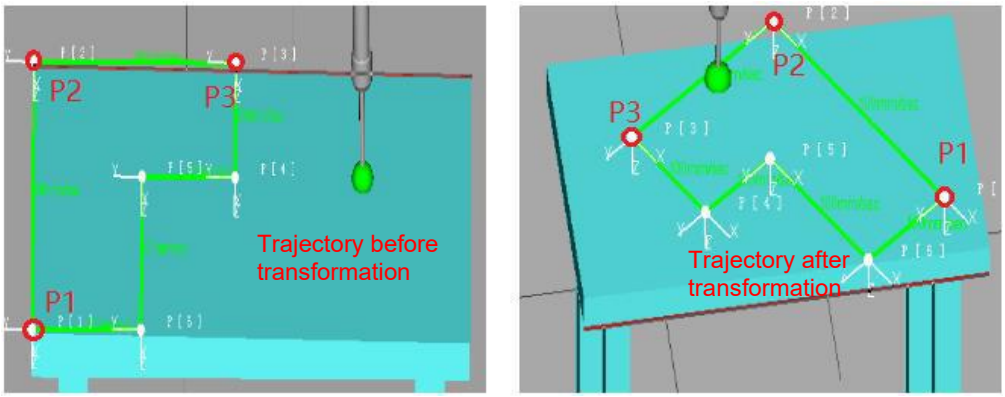


Fig. 8.7 Mirror Offset with Rotation

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.

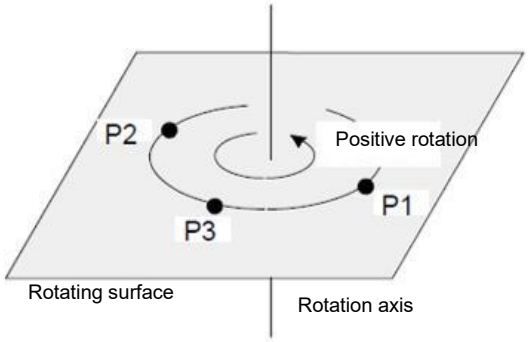


Fig. 8.8 No Rotation Axis Designated

Rotation axis designated

- Reference poses: 3 reference poses and 1 pose on the rotation axis

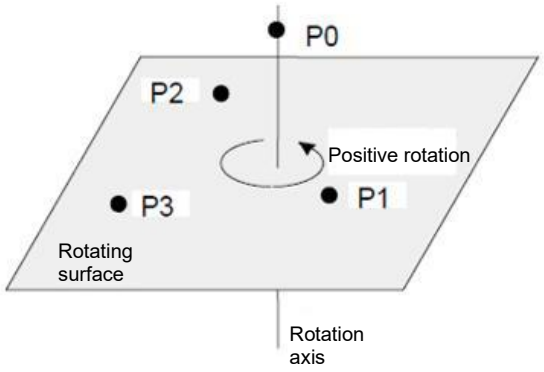


Fig. 8.9 No Rotation Axis Designated

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.

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.

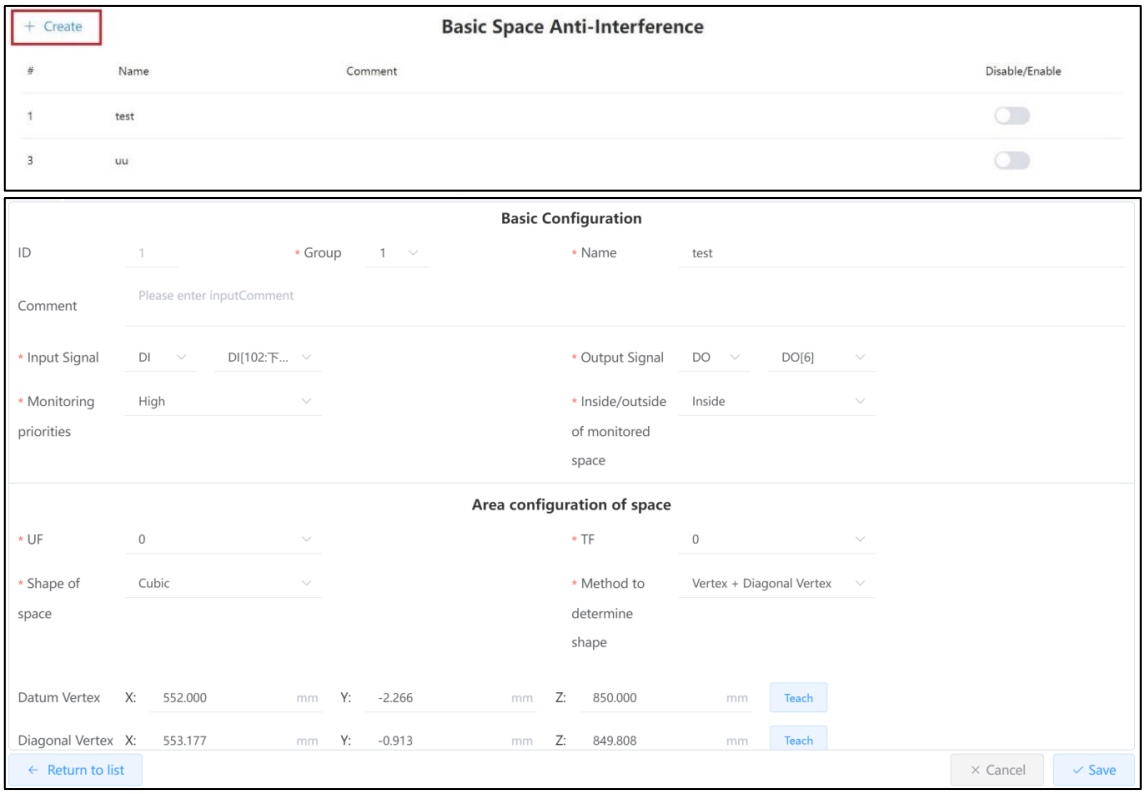


Fig. 8.10 Basic Space Anti-interference

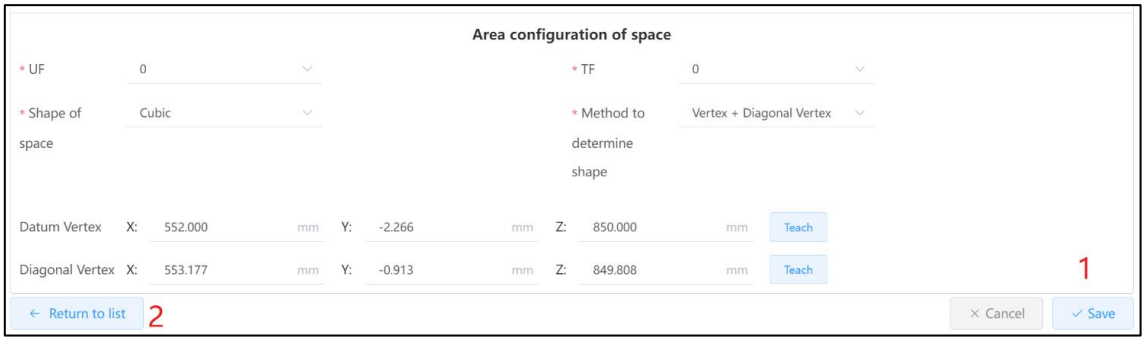


Fig. 8.11 Saving of Basic Space Anti-interference

2. After setting, click "Save" → "Return" → "Deactivate/Activate" to activate the interference region as shown in Fig. 8.11.

Introduction to basic configuration interface:

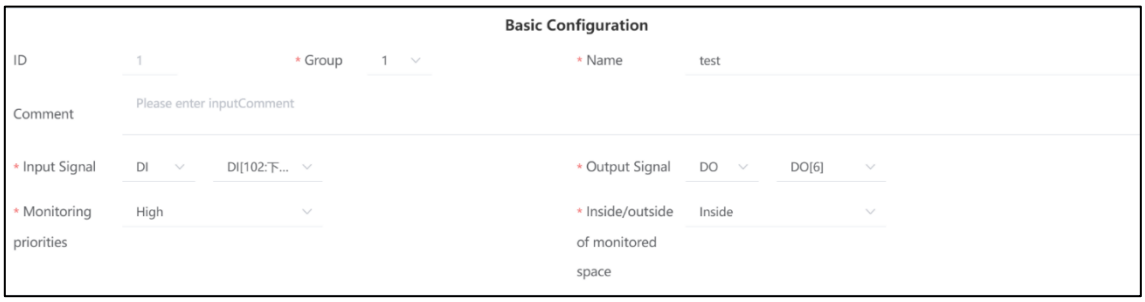


Fig. 8.12 Basic Configuration Interface

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.

Introduction to space region configuration interface:

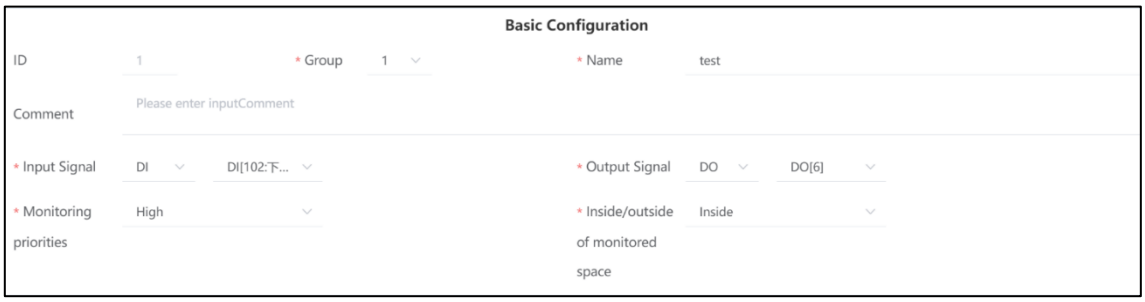


Fig. 13 Space Region Configuration Interface

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.

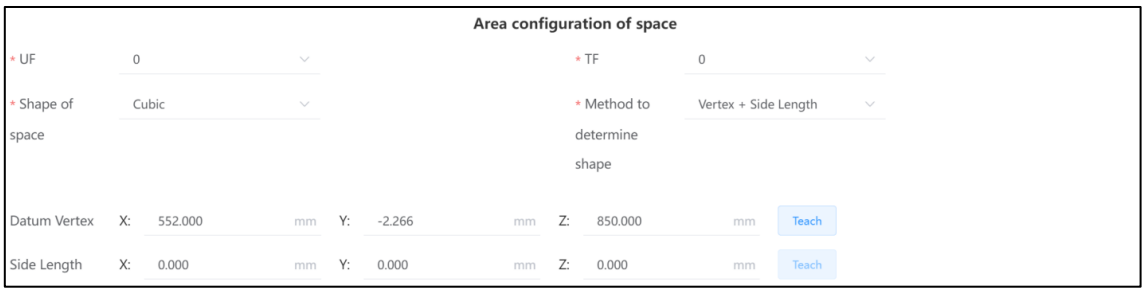


Fig. 8.14 Setting by Vertex + Side Length Method

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.

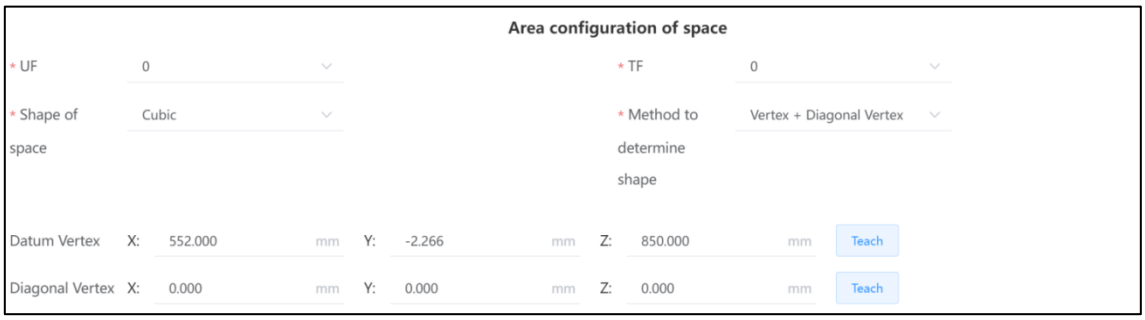


Fig. 8.15 Setting by Vertex + Diagonal Vertex Method

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.

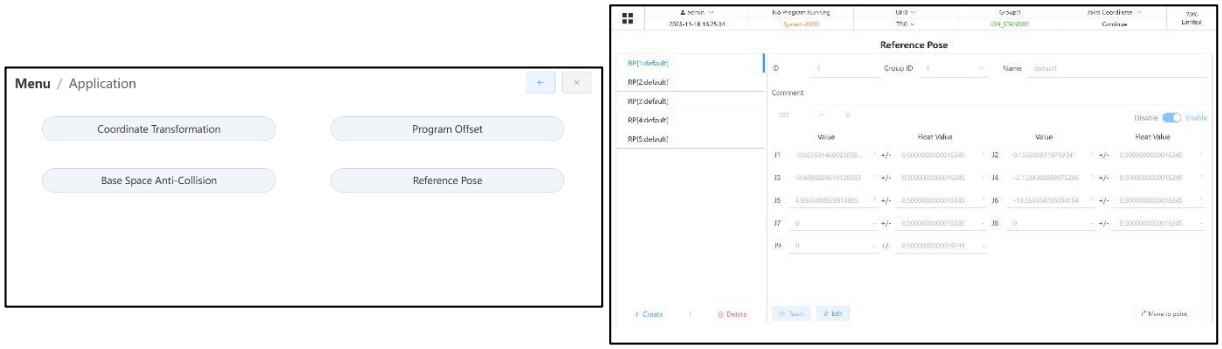


Fig. 8.16 Reference Pose Interface

Specific steps:

1. Click "Create" to create a reference pose 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.

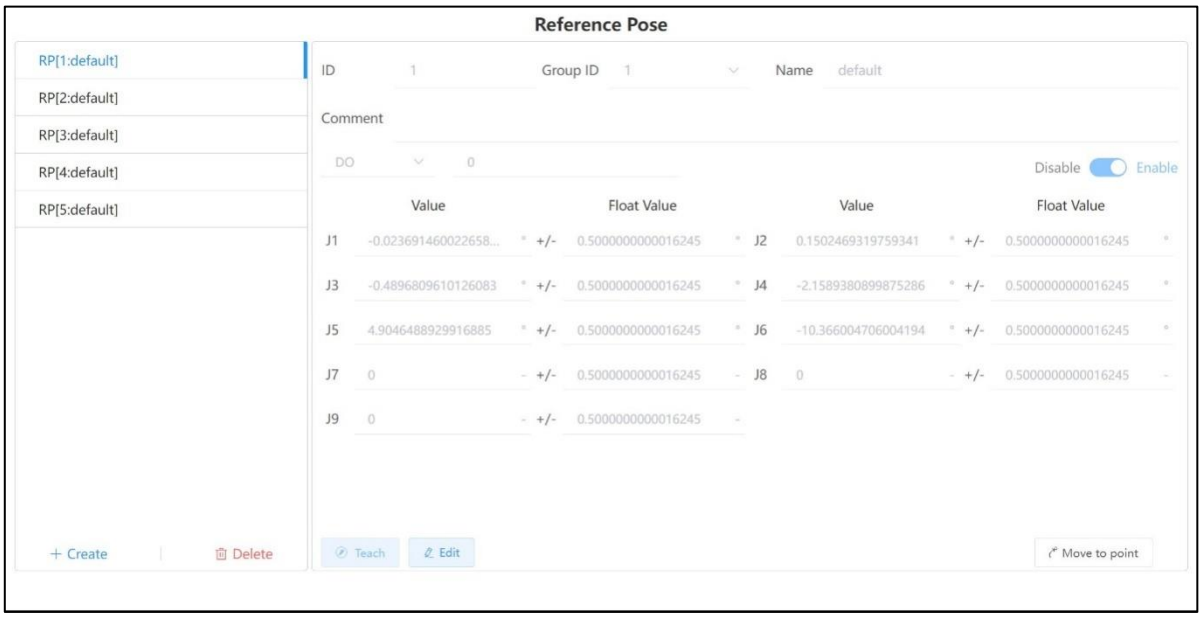


Fig. 8.17 Reference Pose Setting Interface

8.4 TF ALIGNMENT FUNCTION

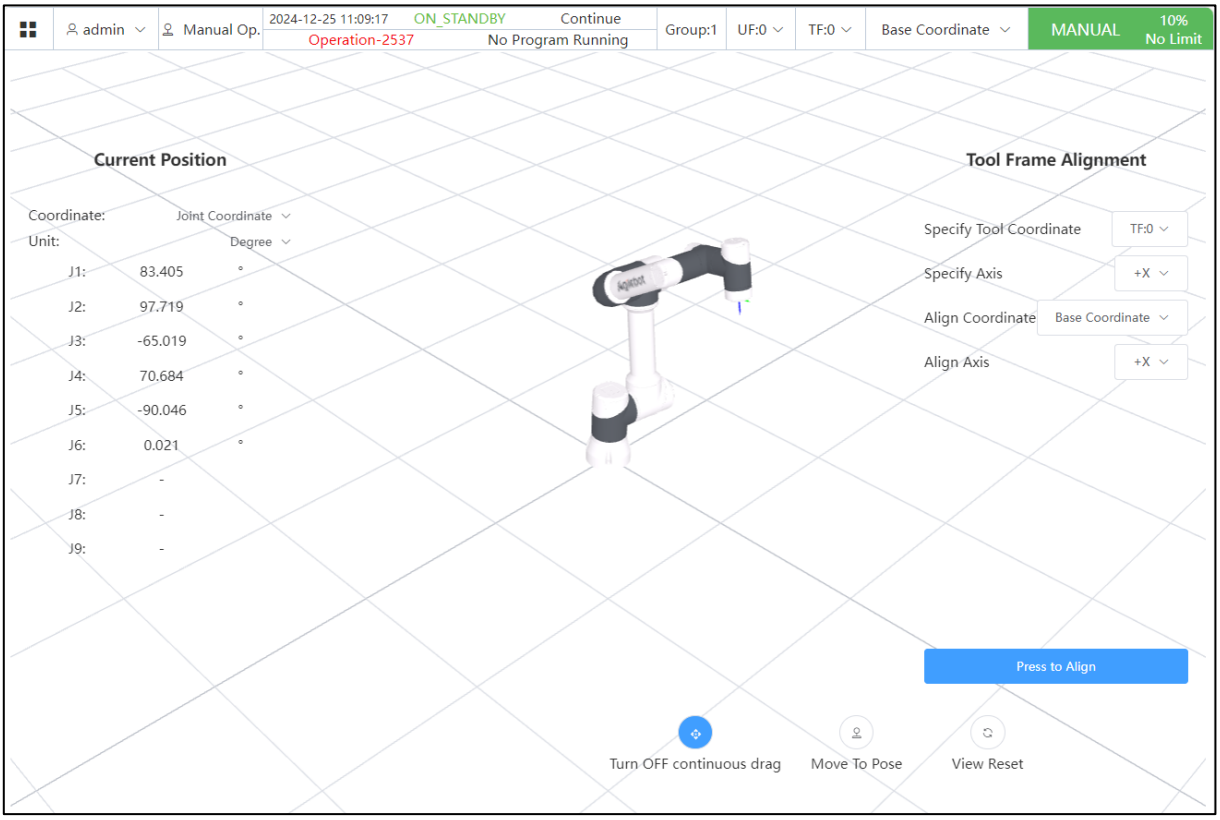
The main purpose of the TF alignment function is to facilitate users to quickly align the orientation of the current tool with a certain axis of a coordinate system (except TF) during the debugging process. Combined with the function of restricting the movement direction in the Cartesian space jogging and drag teaching (see the definition of the drag teaching function for details), it is convenient for subsequent teaching movements and achieves the effect of saving teaching time.

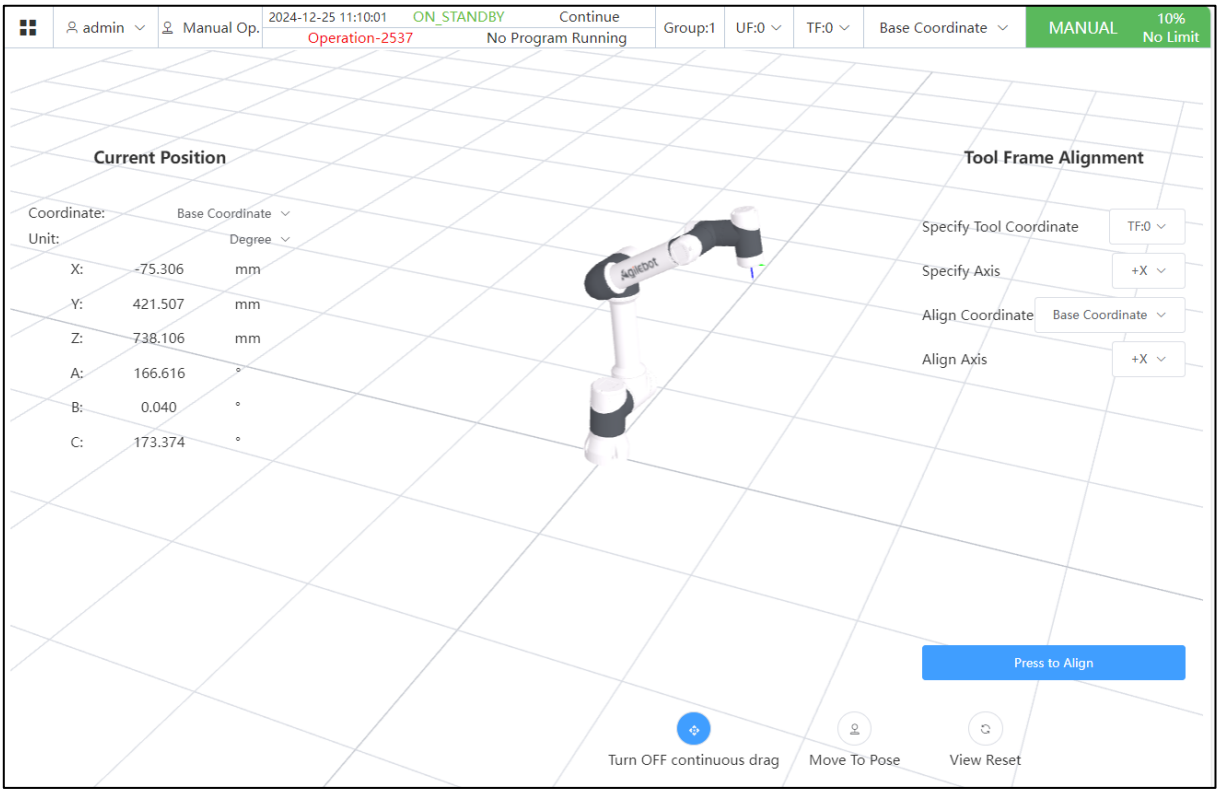
During alignment, only the posture changes (that is, only parameters A/B/C are adjusted).

Operation Steps:

1.In the mobile robot or point position editing interface, click the button 

to enter the TF alignment function interface.



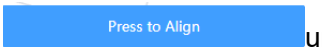


2. Set the TF that needs to be aligned and the axis of this TF.

3. Set the coordinate axis used as the alignment reference.;

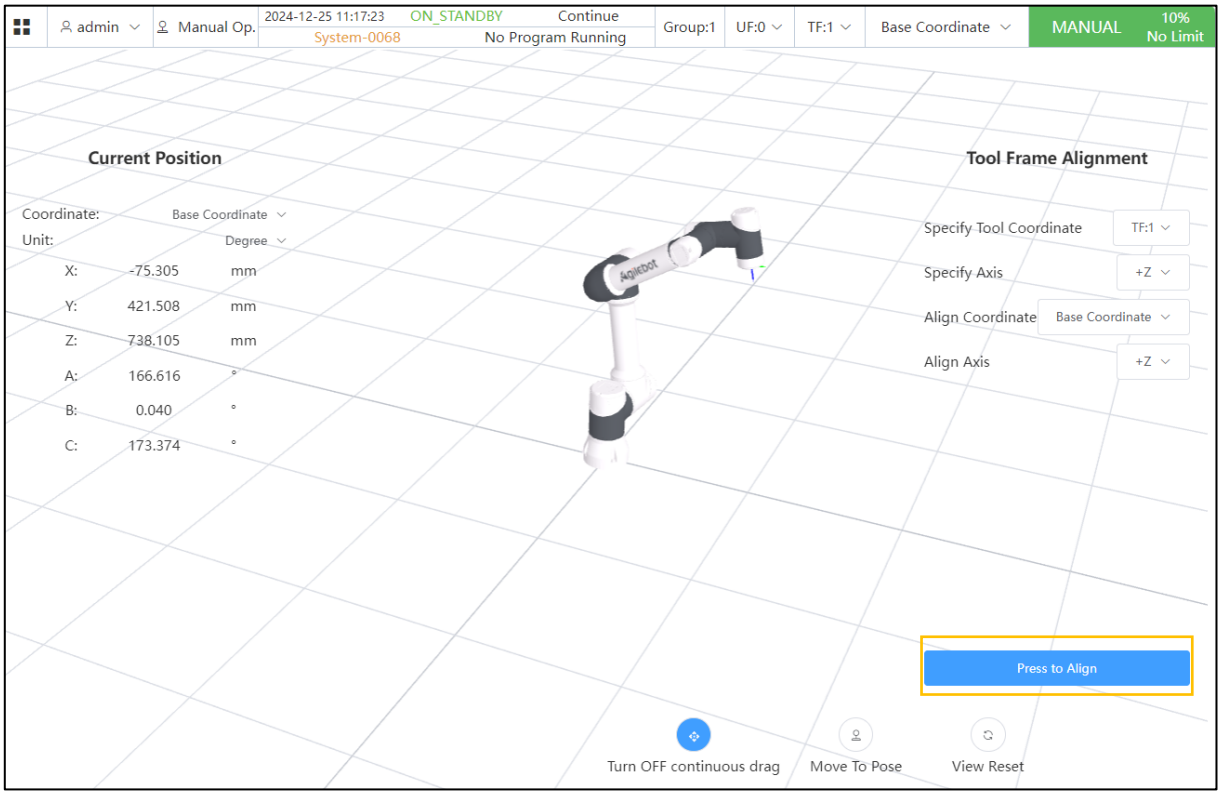
4. 

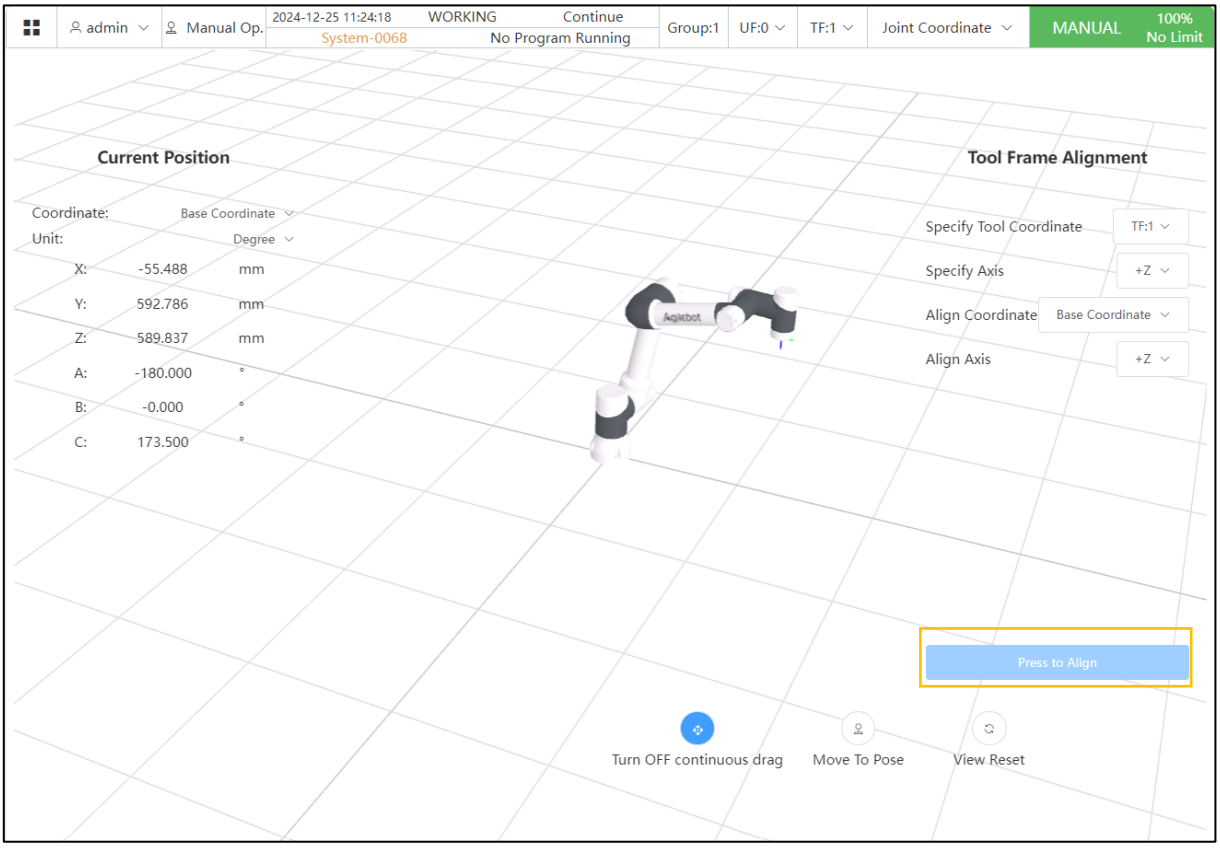
Adjust the speed of the TF during alignment through.;

5. Press and hold 

until the button changes to 

.




admin
Manual Op.
2024-12-25 11:24:18
WORKING
Continue
System-0068
No Program Running
Group:1
UF:0
TF:1
Joint Coordinate
MANUAL
100%
No Limit
Current Position
Coordinate:
Base Coordinate
Unit:
Degree
X: -55.488 mm
Y: 592.786 mm
Z: 589.837 mm
A: -180.000 °
B: -0.000 °
C: 173.500 °
Tool Frame Alignment
Specify Tool Coordinate
TF:1
Specify Axis
+Z
Align Coordinate
Base Coordinate
Align Axis
+Z
Press to Align
Turn OFF continuous drag
Move To Pose
View Reset

9 EVENT INFORMATION TABLE

9.1 EVENT DESCRIPTION

Agilebot robot event levels and corresponding behavior strategies, as shown in Fig. 9.1:

Behavior strategies corresponding to event 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	Unaffected	Pause	Local Overall	None
STOP.L STOP.G	Stop after deceleration	Power off (engage brakes)		Local Overall	Cat. 1 stop
SERVO1	immediately stop			Overall	Cat. 0 stop
ABORT.L ABORT.G	Stop after deceleration	Power off (engage brakes)	Abort	Local Overall	Cat. 1 stop
SERVO2	immediately stop	Power off (engage brakes)		Overall	Cat. 0 stop
SYSTEM				Overall	Cat. 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	Unaffected	Unaffected	Unaffected	/bush	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 is 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

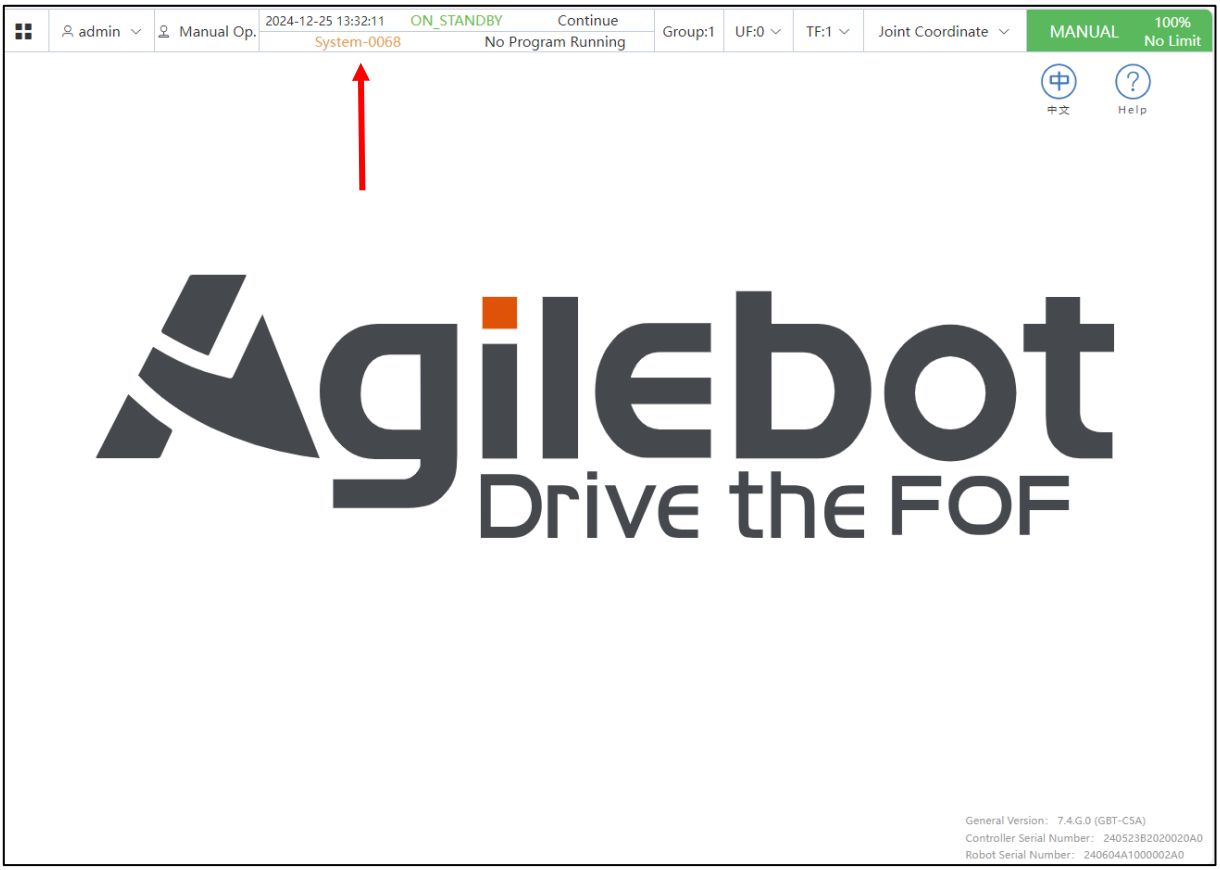


Fig. 9.2 Event Information Area in Status Bar

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.

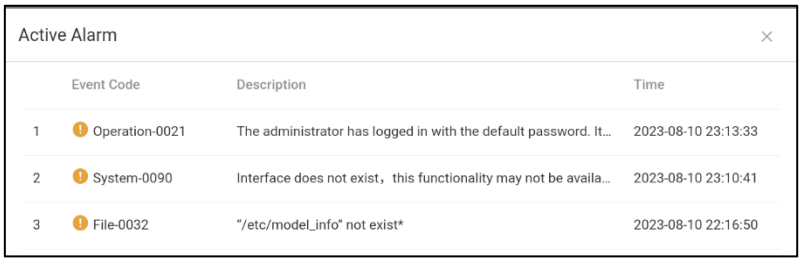


Fig. 9.3 Current Alarm Interface

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.

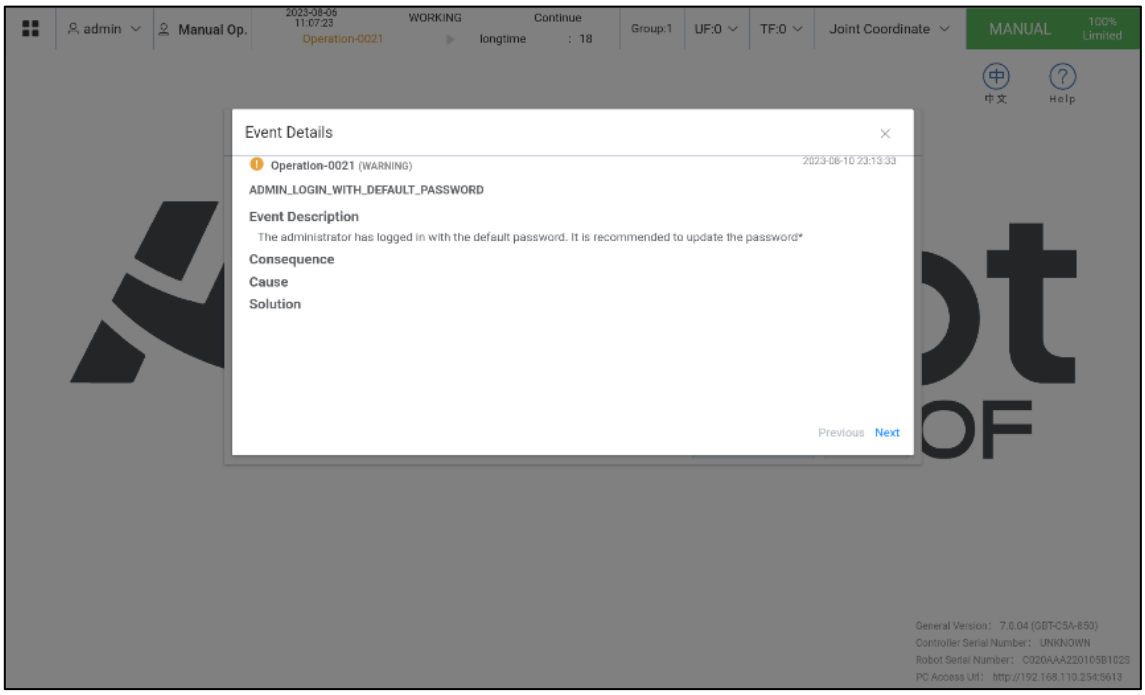


Figure 9.4 Event Detail Interface

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.

Send a Reset signal 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 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
>
> The storage size is limited 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

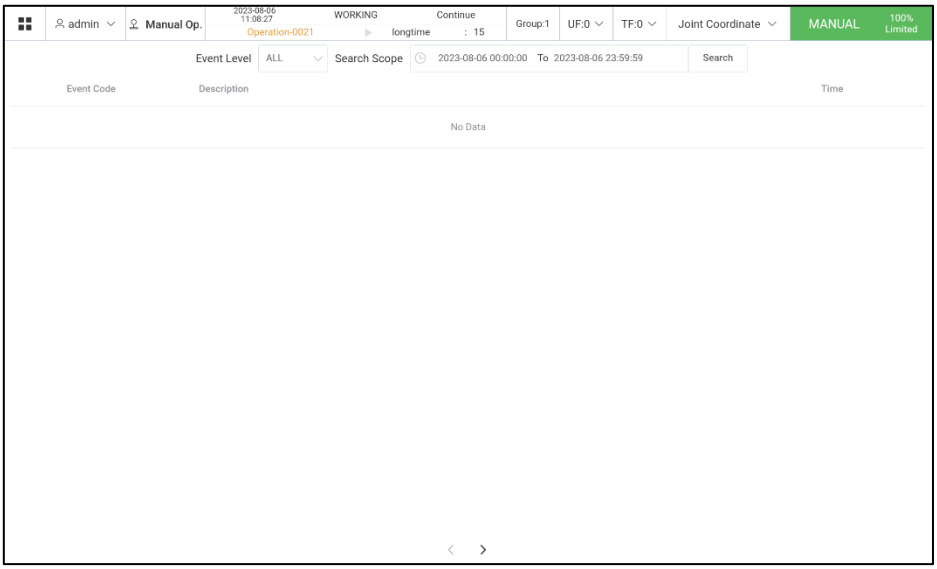


Fig. 9.5 Event Log/History Interface

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.

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):

Floor 8, Tower 6, Zhongjian Jinxiu Plaza, No. 50, Lane 308, Xumin Road, Qingpu District, Shanghai

Agilebot Operation and Technical Service Center:

Building 1, No. 338 Jiuye Road, Qingpu District, Shanghai

Service hotline: +86-21-5986 0805

Website: www.sh-agilebot.com