艾创科技
AITRON TECH

Operation Software Manual

Version：

V1.0

Please ensure that this manual reaches the end user of the product

Table of Contents

1. Notes for Safe Operation

Except as may be expressly stated anywhere in this manual, nothing herein shall be construed as any kind of guarantee or warranty by AITRON for losses, damages to persons or property, fitness for a specific purpose or the like.

In no event shall AITRON be liable for incidental or consequential damages arising from use of this manual and products described herein.

1.1. Forbidden operations

1 Combustion environment

2 Possible explosion environment

3 Radio interference environment

4 In water or other liquids

5 Transport people or animals 6 Never climb

7 Other

1.2. Safety operating rules

1.2.1. Teaching and manual robot

1 Be sure not to use gloves when operate the teaching pendant and the operation panel.

2 Be sure to use a lower speed rate when operating in teaching mode operating such that the control opportunity of the robot is increased.

3 Be sure to consider the moving tendency of the robot when press the start button on the teaching-programming pendant.

4 Be sure to consider how to avoid the moving trajectory of the robot in advance and there is no interference for the moving route.

5 Be sure that the area around the robot must be clean, free of oil, water and impurities, etc.

1.2.2. Production operation

1 Be sure to know all the tasks that the robot will perform according to the setting program before when run the machine.

2 Be sure to know the position and the state of all the switches, sensors, and control signals that will move the robot.

3 Be sure to know the position of emergency stop buttons on the robot control cabinet and peripheral control equipment. Prepare to use these buttons in case of an emergency.

Never judge that a robot has completed its program by there is not any movement because it is likely that the robot is waiting for the input signal to keep it moving.

2. Installlation

2. 1. Installation of teaching pendant

The connecter of the cable of teach pendant is shown in the figure below.

Please connect the male connector to the female connector on the panel of cabinet.

Figure 2.1 Interface at the end of teaching box line

2. 2. Installation of control cabinet

Installation environment

Woring temperature: 0 to 45℃

Be sure enough space for heat dissipation. I

Install in the place with little oscillation (under 0.5G oscillation). Specially, be sure to keeping away from punch and other equipment.

No direct sunshine light, humidity, and water.

No corrosive, flammable or explosive liquids or gases in the area.

It must be a place with little oil and dust. The pollution of installing places is registered as PD2.

NRC series products are installed in the cabinet and need to be installed in the final system. The final system should provide the corresponding fire protection shell, electrical protection shell, and mechanical protection shell, etc, and meet the local laws and regulations and relevant IEC standards, as shown in the figure.

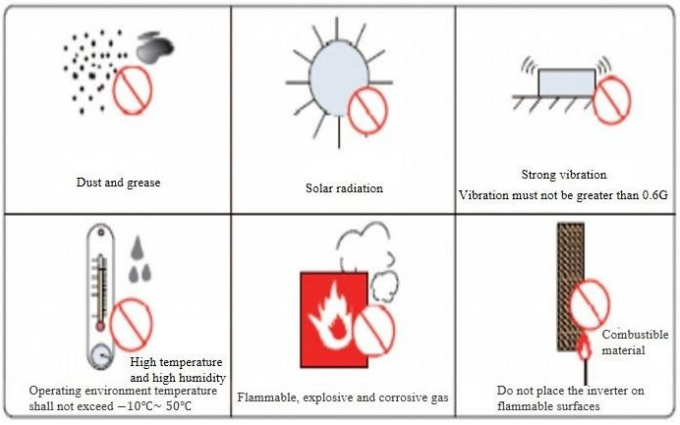


Figure 2.2 Installation environment

Installation location

Install the control cabinet outside of the motion of the robot (outside the safeguarding).

Install the control cabinet in a location from which the robot is easily visible.

Install the control cabinet in a location from which you can easily inspect it when the door is open.

Install the control cabinet at least 500mm from the nearest wall to allow maintenance access.

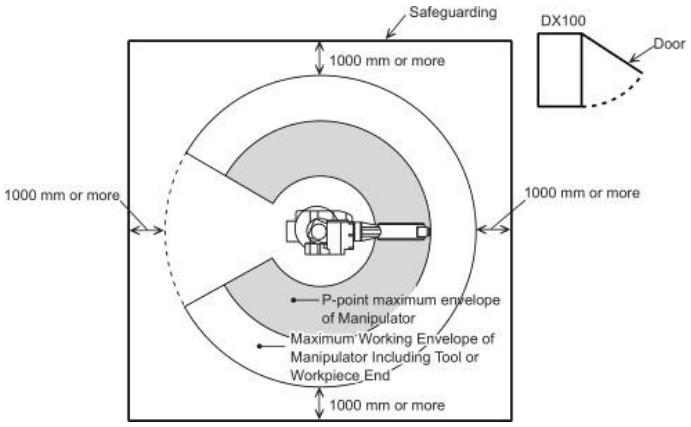


Figure 2.3 Installation location

2.2.1. Cable requirements

1、 Cable classification

Level 1: Sensitive signal (low voltage analog signal, high-speed encoder signal, high speed communication signal, positive and negative 10V analog signal, low speed 422, 485 signal, digital input and output signal).

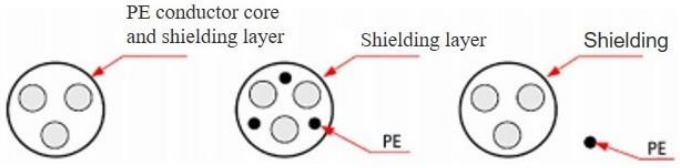
Level 2: Interference signal (low voltage power supply, contactor control line, motor line with recorder, high voltage AC power line, motor line without recorder).

2、 Recommend to use screened balanced cables as the input and output main circuit cables.

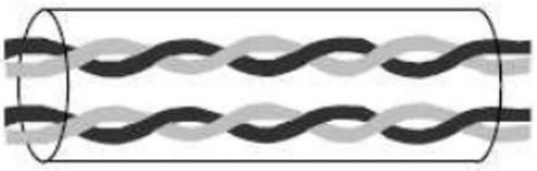
Compared with four-core cables, the use of screened balanced cables can reduce the electromagnetic radiation of the whole conduction system.

Recommend to use screened balanced cables (shown in Figure 1.9) as the power cable.

Recommend to use shielded twisted pair cables (shown in Figure 1.10) as the signal cable. Note: Recommend to use shielded twisted pair cables as the digital signal cables.



Schematic diagram of symmetrical shielded cable



Schematic diagram of double stranded shielded cable

Figure 2.4 Cable type

1 Recommend to use shielded communication cables (shown in Figure) as the communication cables.

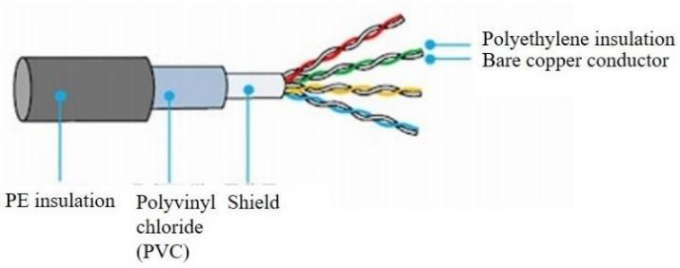


Figure 2.5 Schematic diagram of shielded communication cables

Be sure that the RJ45 Cat.6 crystal head is connected right.

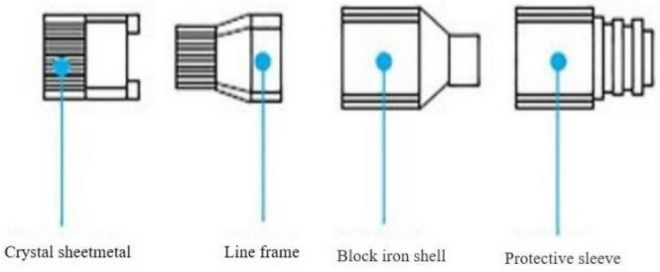


Figure 2.6 Diagram of crystal head with shielded metal shell

2.2.2. Cable design requirements

1.Be sure that the power cables is far from all control cables and encoder cables.

2.Be sure that the motor power cables, input power source cables, and control circuit cables are not wired in the same slot.

3.Be sure that the motor cables and the control circuit cables are not long-distance parallel wiring to avoid the electromagnetic interference.

4.Be sure at least 100 mm space distance between different grades of cables in a same slot. Different grades of cables are arranged separately. When long distance cables are wired in the same direction, be sure at least 100 mm space distance between different grades of cables. The metal part of the controller is directly connected to the back plate by using the conductor as the back plate (using the zinc plate which is not sprayed). Be sure the separation of different grades of cables. If the cables of different grades must be crossed, be sure the crossing of 90 degrees.

2.2.3. Grounding Requirements

Be sure that the grounding end is grounded, otherwise possibly causing injury from electric shock or misoperation due to the interference.

1.Grounding requirements for power lines, as shown in the Figure.

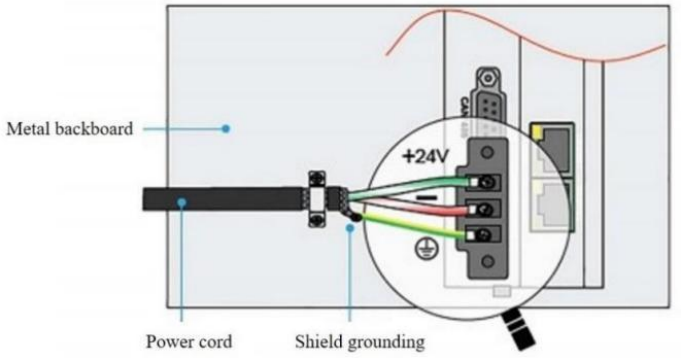
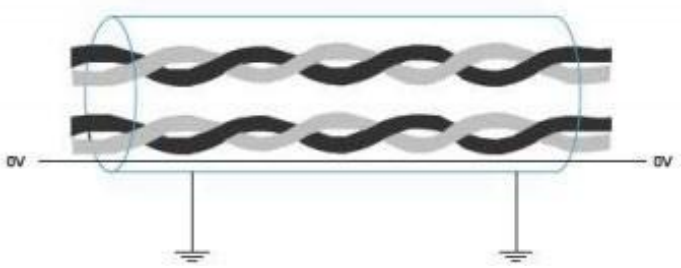


Figure 2.7 Grounding requirements for power lines

Differential signal line (CAN/RS485/RS422) uses shielded twisted pair cables. The shielding layer must be linked to 0V at both ends of the cable, as shown in the figure.



2.2.4. Wiring Notes

- Figure 2.8 Twisted pair shielded cable
- Wiring and inspection must be performed only by professionals with appropriate skills.
- The system must be grounded reliably. Be sure that the grounding resistance is less than 4 ohms and the grounding line is not replaced by the neutral line (zero line).
- Be sure that the wiring is correct and tightened. Incorrect wring may cause system failure or unexpected consequences.
- Be sure that the surge snubber diode is connected correctly to the system in accordance with the specified direction, otherwise the product will be damaged.
- Before plugging in/out or opening the product crate, make sure to cut off the power supply.
- The signal and power lines are not wiring in the same pipeline if possible (at least 30 mm space distance).
- For signal line, encoder (PG) feedback line, use multi-stranded wire and multi-core stranded shielded wire. For wiring length, the longest instruction input line is 3m and the longest PG feedback line is 20m. The signal line of the code line is a group of twisted pair wires, the power line is a group of twisted pair wires, and the battery line is a group of twisted pair wires.
- Do not frequently turn ON/OFF the power supply. When the continuous operation of ON/OFF power supply is required, be sure less than once per minute. Since there is capacitance in the power supply part of the servo unit, frequent ON/OFF will result in the performance degradation of the main circuit components of the servo unit.
- Confirm the power and voltage of the switching power supply in the control system. Be sure that the

voltage of controller, teaching-programming pendant, and IO module is not less than 50W. The specific power supply depends on the load of IO module.

- Suggest that the switching power supply for servo system and control system are used separately to prevent the occurrence of servo interference on the control system.

  1.Super six types of shielded wire are needed for the control system and servo connection.

  2.If an axis corresponds to a servo, the wire needs to be connected in the order of the axes.

  3.Please connect in the order of controller-servo-IO board.

The connection definition diagram of teaching-programming pendant

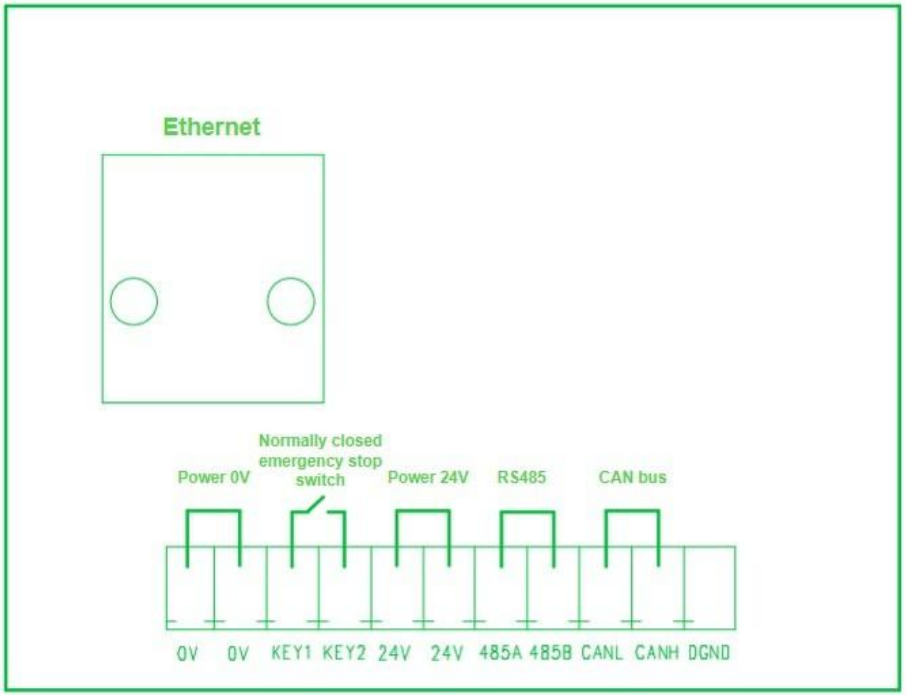


Figure 2.9 the connection definition diagram of teaching-programming pendant

3. Configuration steps for a new robot

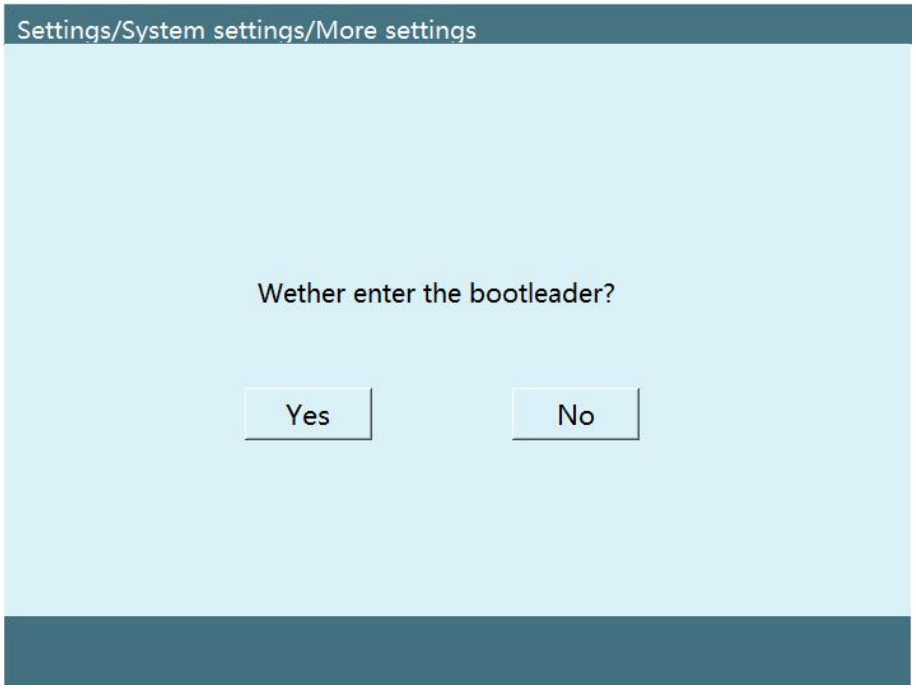
1.The robot and servo information of controller must be configured appropriate before operation with robot, otherwise the controller will report "unable to connect servo" and stop.

Note: It may takes a few minutes for new configuration affects. You will see “disconnected” warning information shown in teach pendant. It will recover automatically after the configuration steps finish.

Item	Description
The number of robots	1-4
Robot type	Scara,6 dof,…
Servo type	Servo driver product information
External axis type	
External axis servo type	
IO moduals	

It’s recommend to use “install wizard” for new user. The wizard is in "Settings-System Settings- Other Settings"

2.Once the configuration of the robot is completed and restarted, then set the key parameters of the robot, such as robot parameters, DH parameters, Cartesian parameters, etc. After then, perform power up and other operations.

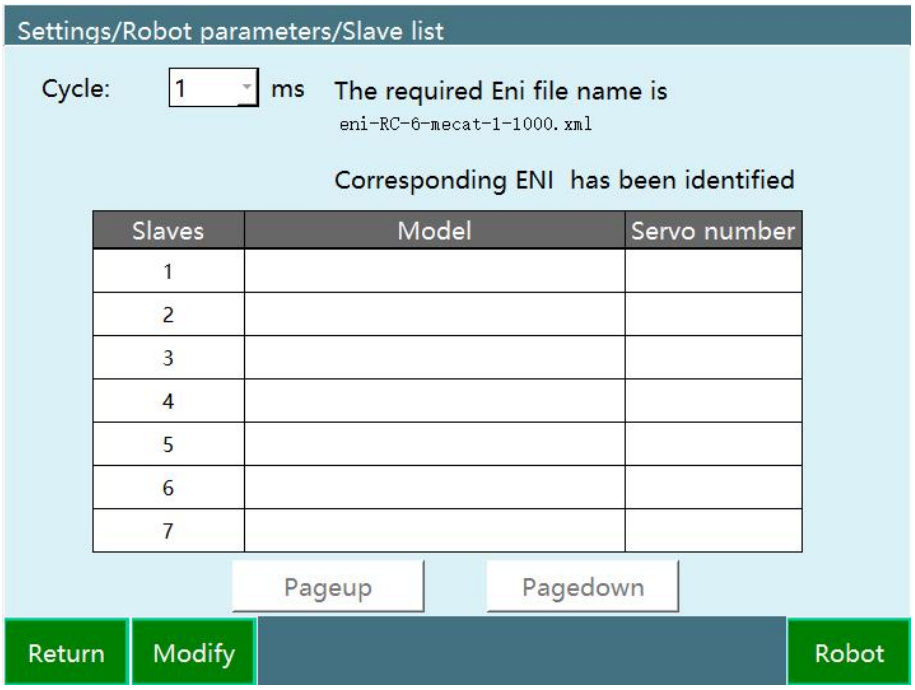


If you want to configure the parameters manually instead of using the configuration wizard, here are the complete parameters configuration steps:

Switch permission to “Administrator” and the default password is 123456.

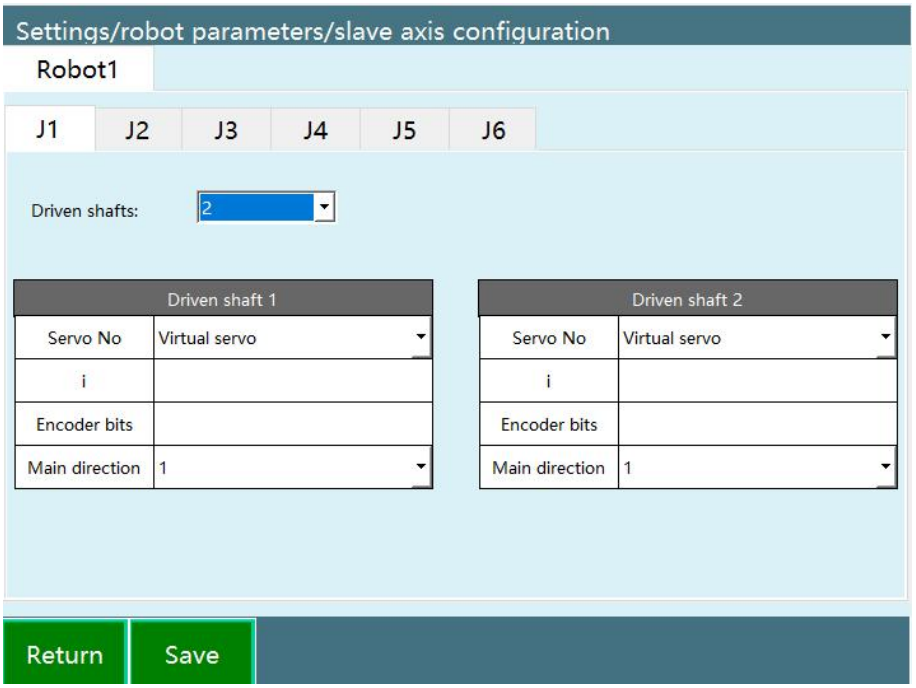
Under the menu "Setting-Robot Parameters-Robot Configuration", configure the number of robots, the communication cycle of robots, the type of robots, and the type of servo (Make sure that the type of robots is correct, otherwise the robot will not move normally)

1. The servo list displays the number of servo models read after the current controller is turned on. This interface can set the communication cycle.

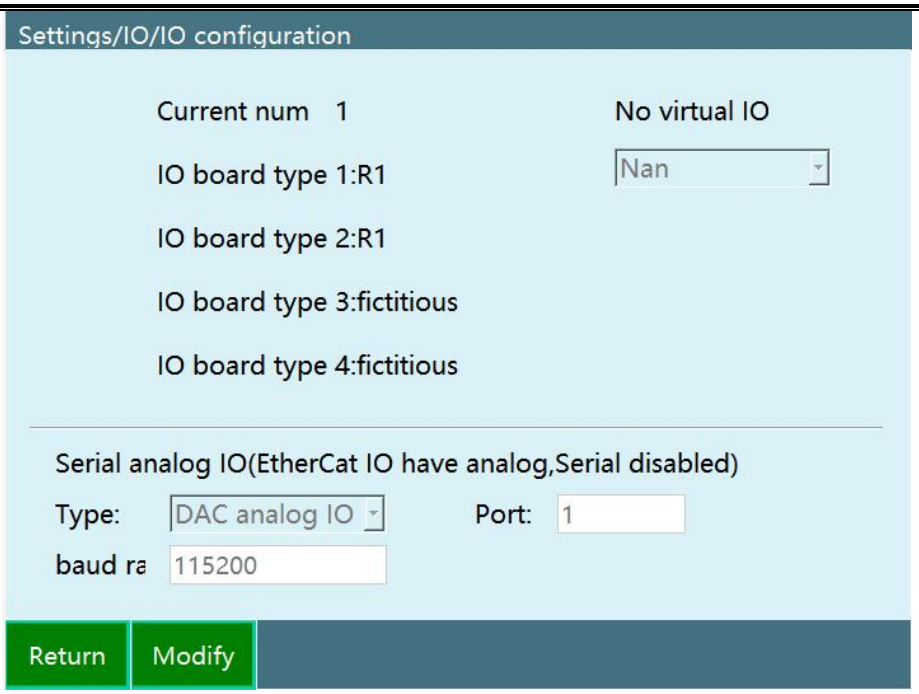


Robot servo configuration can configure the number of robots, robot type, number of external axes, and servo options.

Slave axis setting, you can set the number of slave axis and slave axis servo.



3.In "Settings-IO-IO configuration", configure the serial port analog IO type and the number of virtual IOs. Normal EtherCAT IO does not need to be set;

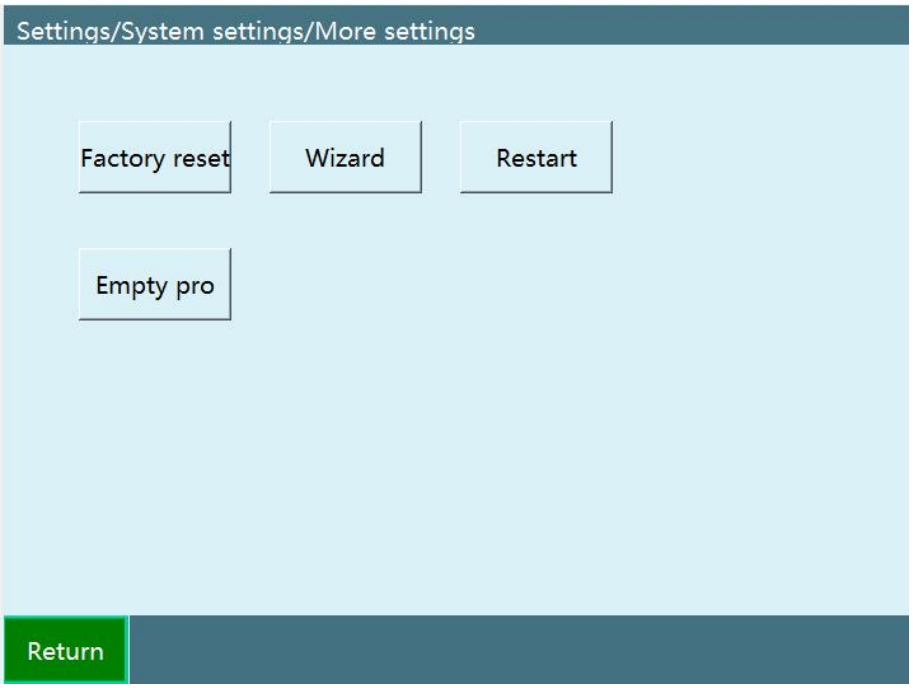


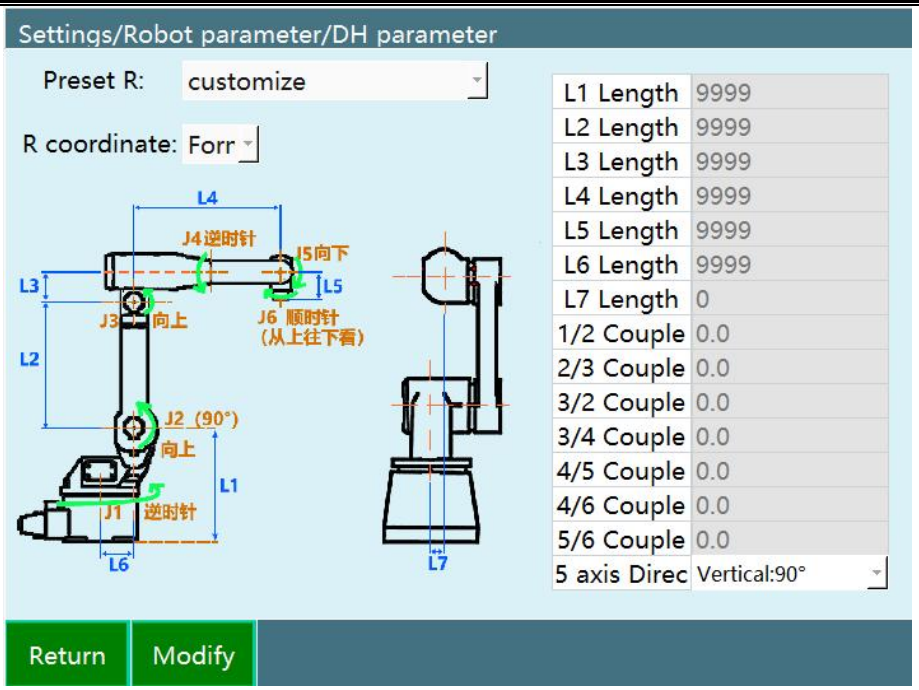
Note: When using Huatai IO, the ENI file is slightly different.

4.Restart the system (The modification of the robot configuration becomes effective after restarting);

5.In the DH parameter interface， we provide the preset function for robot. If your robot type is in the dropdown list, the setting of each parameter is quick and easy through the preset function.

6.The selection of robot coordinate is based on assembly (flip: Cartesian coordinates, tool coordinates, user coordinates are consistent with the operating habits of forward erection)

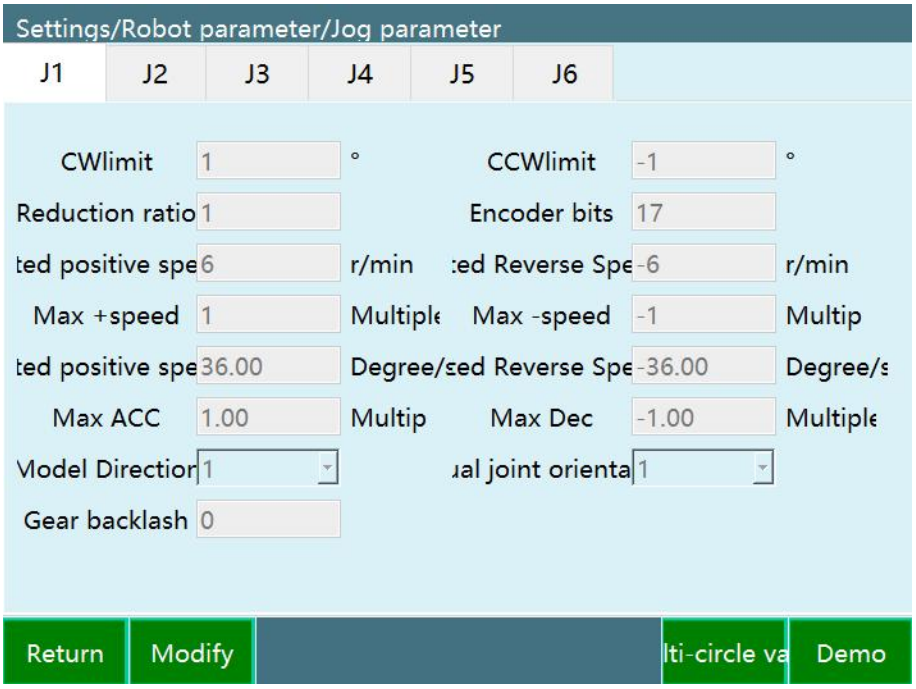




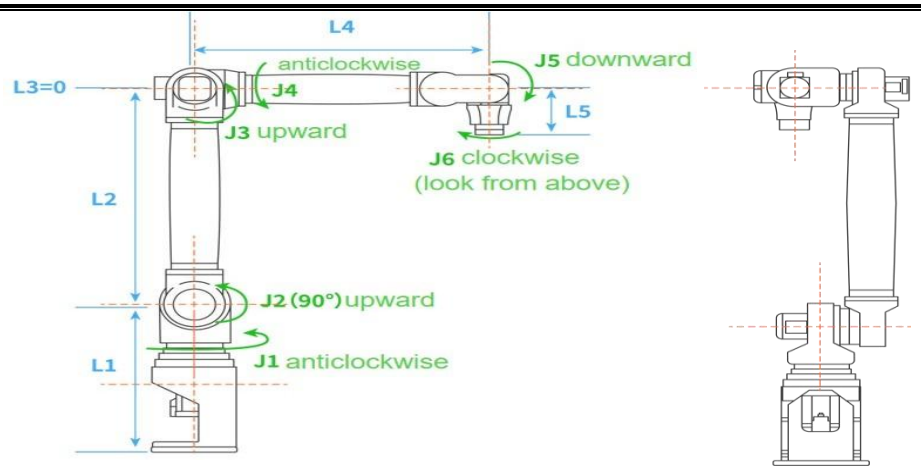
7.In the DH parameter interface, after clicking “preset robot” button on the top left corner, you can select the type of robot that has been adapted. After selecting, the DH parameters and joint parameters of the robot are automatically filled out.

8.If select to preset robot, the zeros must be modified manually.

9.If there is no such robot type in the option, please fill in the parameters manually according to the following steps.



10.In the menu "Settings-Robot parameters-Jog parameters", fill out all parameters and set the limit of each joint from -3333 to 3333; (Please click individually each axis of the robot to check whether the positive direction of each axis of the robot is correct or not)



Robot type	shaft	Positive direction
Six axis	J1	anticlockwise
	J2	upward
	J3	upward
	J4	anticlockwise
	J5	downward
	J6	clockwise
Four axis SCARA	J1	anticlockwise
	J2	anticlockwise
	J3	downward
	J4	clockwise
Four shaft palletizing	J1	anticlockwise
	J2	upward
	J3	upward
	J4	anticlockwise
Four shaft joint	J1	anticlockwise
	J2	upward
	J3	upward
	J4	upward
Five-axis joint	J1	anticlockwise
	J2	upward
	J3	upward
	J4	anticlockwise
	J5	downward(
Two axis SCARA	J1	anticlockwise
	J2	anticlockwise

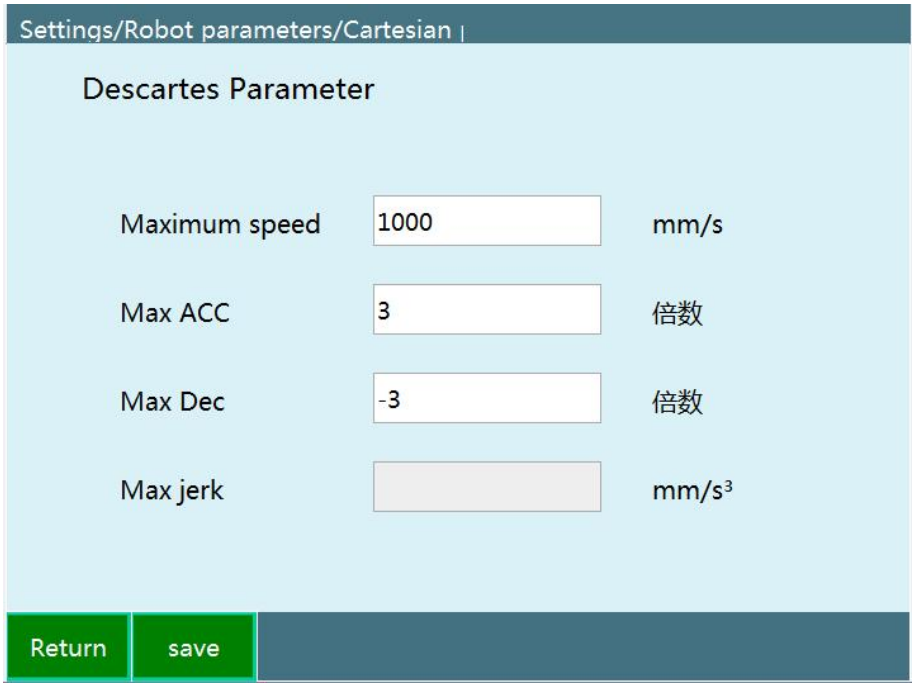
Three axis SCARA	J1	anticlockwise
	J2	anticlockwise
	J3	upward
One axis	J1	anticlockwise
Four axis SCARA special-shaped	J1	upward
	J2	anticlockwise
	J3	anticlockwise
	J4	clockwise

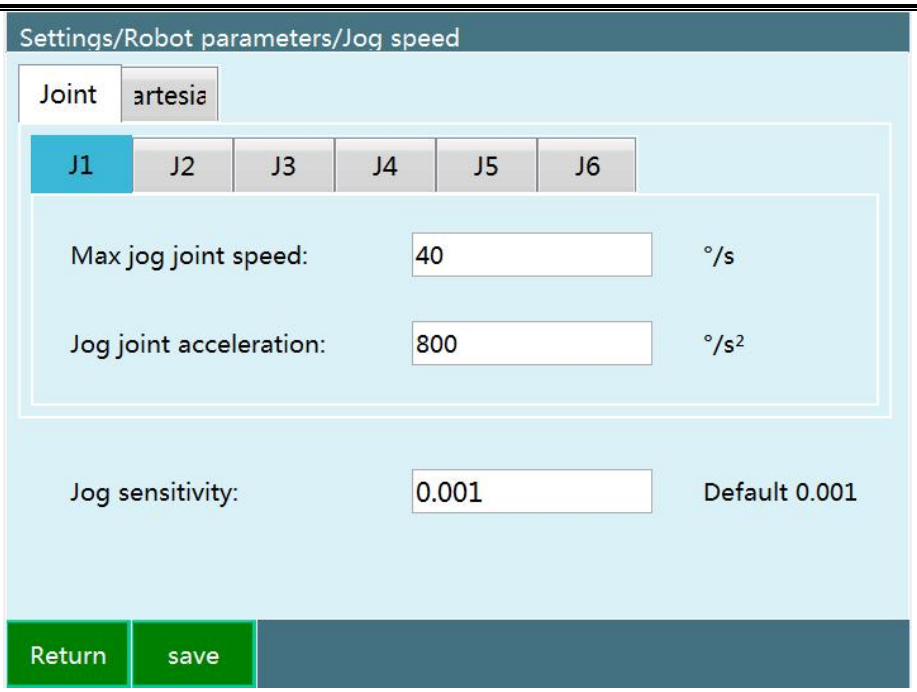
11.In the menu "Setting-Robot Parameters-Zero Position", set the zero of the robot. If five-axis is vertically downwards, select "Five-axis Vertical" in the last line of the DH parameter interface. If five-axis is horizontal, select "Five-axis Horizontal".

12.In the menu "Setting-Robot Parameters-Joint Parameters", the limit position of each axis joint is set according to the specific working environment.

13.In the menu "Setting-Robot Parameters-DH Parameters", fill in the parameters according to the actual parameters of the robot. The acceleration and deceleration can be set to 4-6 times of the maximum positive speed and the maximum negative speed.

14.Check whether the Cartesian parameters, manual speed and operation parameters are correct or not.




Settings/Robot parameters/Jog speed
Joint
artesia
J1
J2
J3
J4
J5
J6
Max jog joint speed:
40
°/s
Jog joint acceleration:
800
°/s²
Jog sensitivity:
0.001
Default 0.001
Return
save

4. The basics of robotics

4. 1. Control Groups and Coordinate Systems

4. 1. 1. Coordinate Systems

The following coordinate systems can be used to operate the robot，as shown in the following figure.

Joint Coordinates：

Each axis of the robot moves independently.

Cartesian Coordinates：

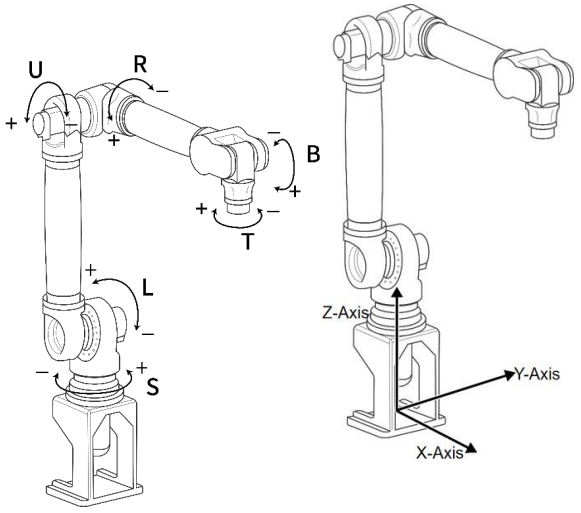
The tool tip of the robot moves parallel to any of the X-, Y-, and Z-axes. A, B and C rotate around X, Y and Z axes respectively. The Euler angle used in this system is XYZ.

Tool Coordinates

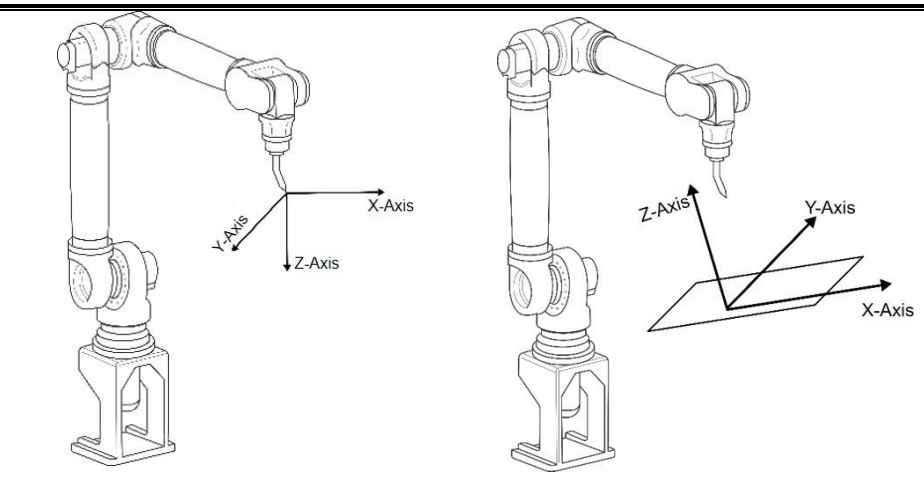
The effective direction of the tool mounted in the wrist flange of the robot is defined as the Z-axis. The origin of the coordinate system is defined at the tip of the tool, and the tip of the body moves in parallel according to the coordinates. TA, TB and TC rotate around TX, TY and TZ axes respectively.

User Coordinates

The XYZ-cartesian coordinates are defined at any point and angle. The tool tip of the body moves parallel to the Coordinates of them.



Joint Coordinates Cartesian Coordinates



Tool Coordinates User Coordinates

Figure 4.1 several coordinate systems

4.2. Coordinate Systems and Axis Operation

4.2.1. Joint Coordinates

When operating in joint coordinates mode, the axes of the robot move independently. When pressing the axis operation key that the robot does not have, it does not do any action.

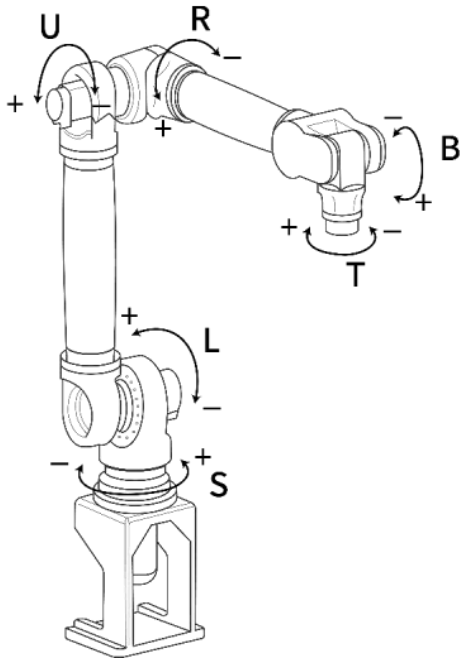


Figure 4.2 Joint Coordinates

4.2.2. Axis Motion in Joint Coordinates

Axis Name		Axis operation	Action
Basic axis	S axis	S+/S-	The body rotates left and right

	L axis	L+/L-	Anterior and posterior movement of lower arm
	U axis	U+/U-	Upper and lower arm movements
Wrist axis	R axis	R+/R-	Wrist rotation
	B axis	B+/B-	Up and down movement of wrist
	T axis	T+/T-	Wrist rotation

4.2.3. Cartesian Coordinates

In the cartesian coordinates, the manipulator moves parallel to the X-, Y-, or Z-axes. The motion of each axis is described in the figure below.

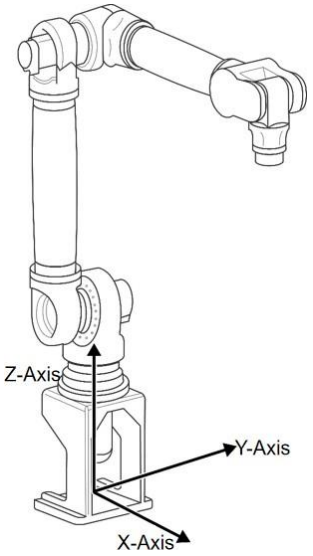


Figure 4.3 Cartesian Coordinates

Moves parallel to X- or Y-axis Moves parallel to Z-axis

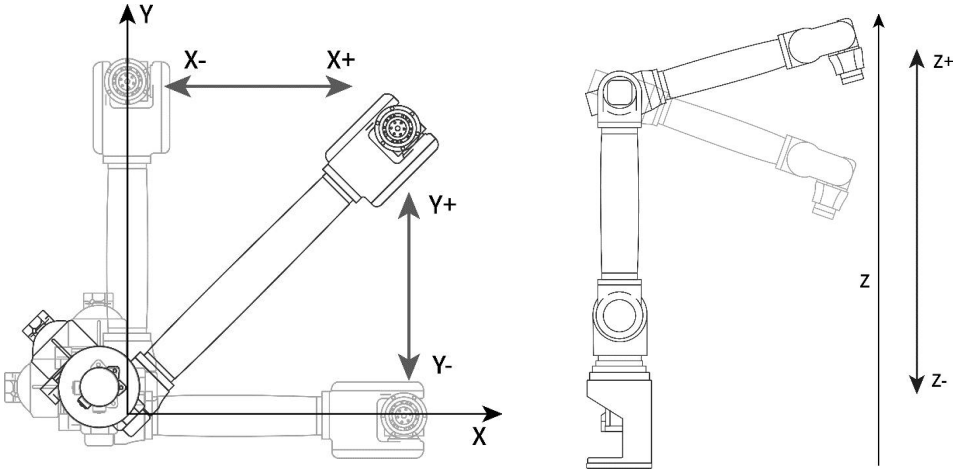


Figure 4.4 Cartesian coordinates axis operation

4.2.4. Axis Motion in Cartesian Coordinates

Axis Name		Axis operation	Action
Basic axis	X axis	X+/X-	Parallel movement along X-axis
	Y axis	Y+/Y-	Parallel movement along Y-axis
	Z axis	Z+/Z-	Parallel movement along Z-axis
Attitude axis	A axis	A+/A-	Rotate around the X-axis
	B axis	B+/B-	Rotate around the Y-axis
	C axis	C+/C-	Rotate around the Z-axis

4.2.5. Tool Coordinates

In the tool coordinates, the manipulator moves parallel to the X-, Y-, and Z-axes, which are defined at the tip of the tool.

The tool coordinates are defined at the tip of the tool, assuming that the effective direction of the tool mounted on the manipulator wrist flange is the Z-axis.

Therefore, the tool coordinates axis direction moves with the wrist. The motion of each axis is described in the figure below.

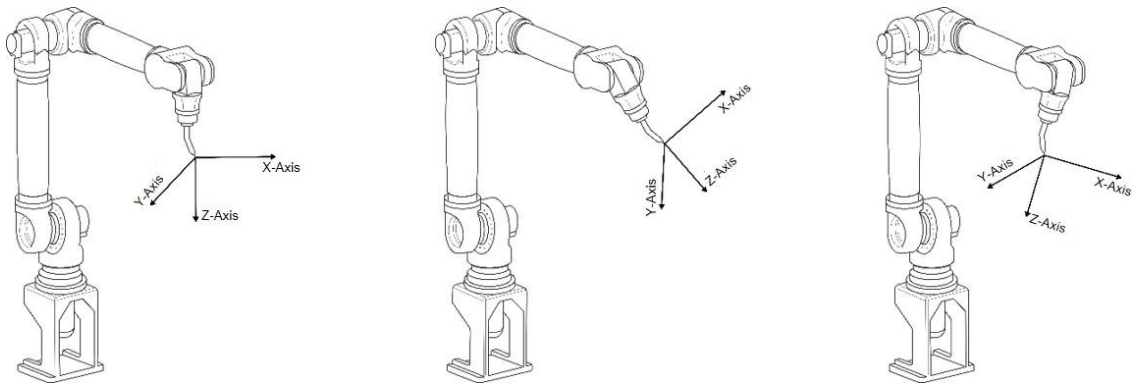


Figure 4.5 Tool Coordinates

In the tool coordinates motion, the manipulator can be moved using the effective tool direction as a reference regardless of the manipulator position or orientation.

These motions are best suited when the manipulator is required to move parallel while maintaining the tool orientation with the workpieces.

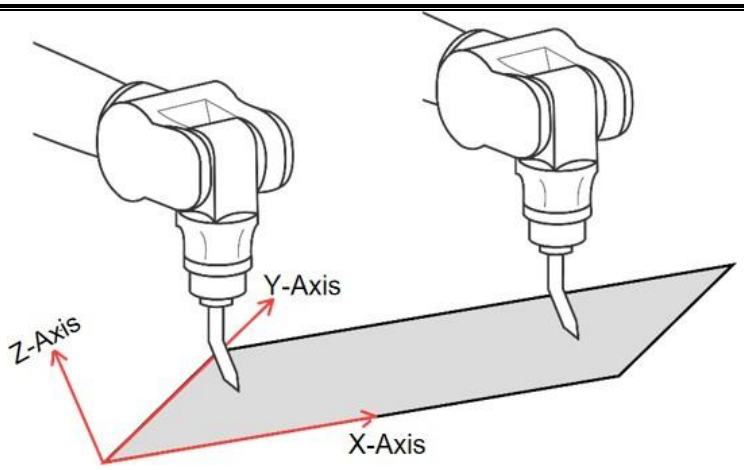


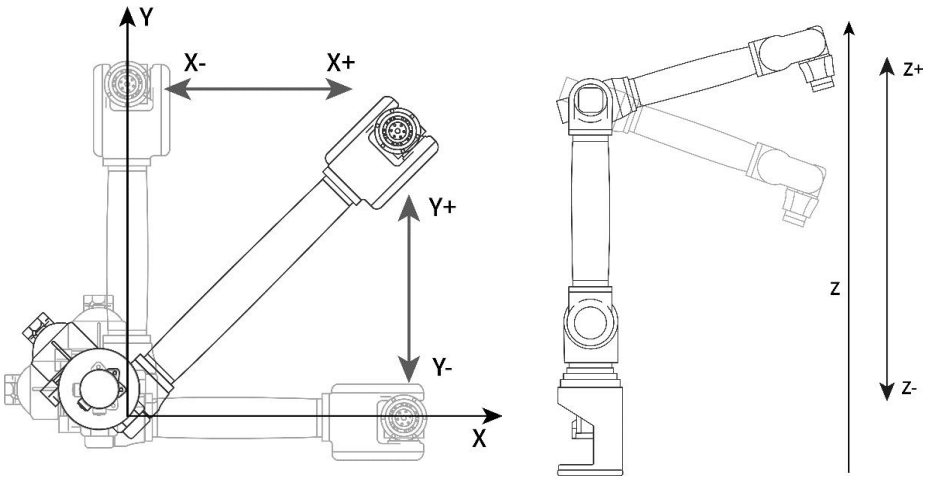
Figure 4.6 Tool coordinates axis operation

4.2.6. Axis Motion in Tool Coordinates

Axis Name		Axis operation	Action
Basic axis	TX axis	TX+/TX-	Parallel movement along TX-axis
	TY axis	TY+/TY-	Parallel movement along TY-axis
	TZ axis	TZ+/TZ-	Parallel movement along TZ-axis
Attitude axis	TA axis	TA+/TA-	Rotate around the TX-axis
	TB axis	TB+/TB-	Rotate around the TY-axis
	TC axis	TC+/TC-	Rotate around the TZ-axis

4.2.7. User Coordinates

In the user coordinates, set any angle in any position which is the range of the manipulator motion. The manipulator moves parallel to each axis of the coordinates which are set by the user. It is shown in the figure.



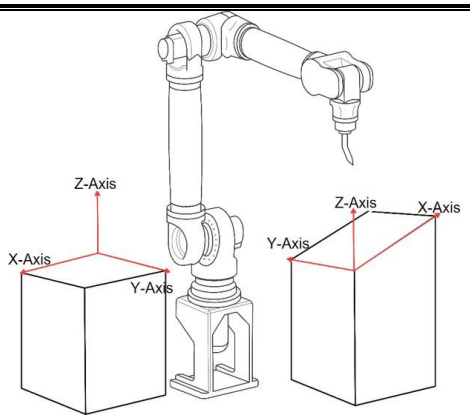


Figure 4.7 User Coordinates

4.2.8. Axis Motion in User Coordinates

Axis Name		Axis operation	Action
Basic axis	UX axis	UX+/UX-	Parallel movement along UX-axis
	UY axis	UY+/UY-	Parallel movement along UY-axis
	UZ axis	UZ+/UZ-	Parallel movement along UZ-axis
Attitude axis	UA axis	UA+/UA-	Rotate around the UX-axis
	UB axis	UB+/UB-	Rotate around the UY-axis
	UC axis	UC+/UC-	Rotate around the UZ-axis

4.2.9. Examples of User Coordinate Utilization

The user coordinate settings allow easy teaching in various situations. For example:

When there are multiple fixture platforms: manual operation can be simplified by setting the user coordinates for each fixture.

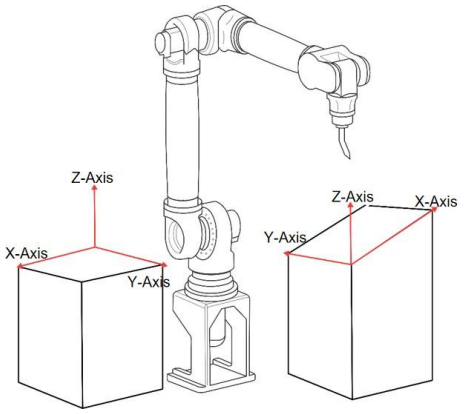


Figure 4.8 Multiple fixture platforms

When performing arranging or stacking operations, the incremental value for shift can be easily programmed by setting user coordinates on a pallet.

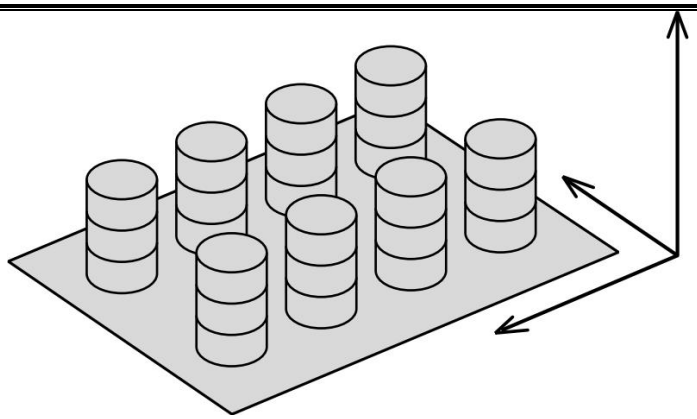


Figure 4.9 Arrangement and placing job

When performing conveyor tracking operations, the moving direction of the conveyor is specified.

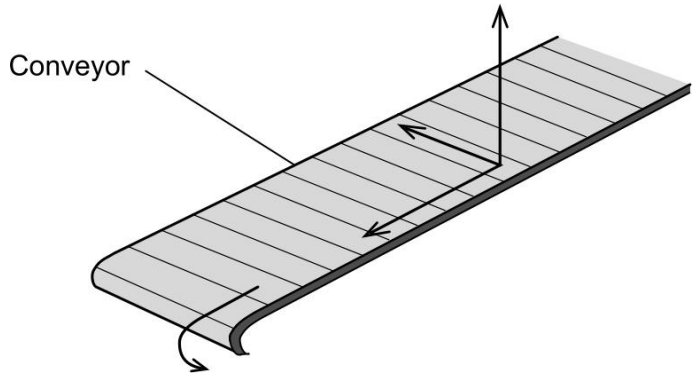


Figure 4.10 specify the direction of movement of the conveyor belt

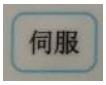
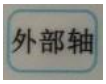
4. 3. External Axis

After using the external axis button to switch to the external axis, you can jog and teach the external axis; the external axis only supports joint jog.

Axis Name	Axis operation	Action
O1 axis	J1+/J1-	1-axis rotating motion of external axis
O2 axis	J2+/J2-	2-axis rotating motion of external axis
O3 axis	J3+/J3-	3-axis rotating motion of external axis
O4 axis	J4+/J4-	4-axis rotating motion of external axis
O5 axis	J5+/J5-	5-axis rotating motion of external axis

5. Introduction of button and interface of demonstrator

5.1. The T30 teach pendant physical buttons

	Switch current servo state
	Switch the current robot. (Only available in multi-machine mode)
	Switch between the current robot and external axis. (Only available when there is an external axis)
	Home button
	Recovery site button
	The error is cleared after the servo reports an error. (Only valid in teaching mode)）
	Reserved

Down side

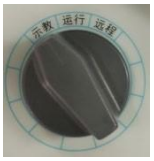
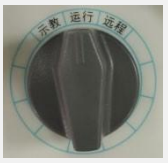
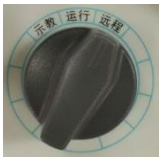
	Switch between sequential execution or reverse execution when single-step running program in teaching mode.
	Run the program step by step in the teaching mode.
	Reduce the teaching or running speed.
	Increase the teaching or running speed.
	Switch tool hand (reserved).
	Switch whether to execute the program in a single step in the teaching mode in order or in reverse order.

Right side

	Pause the program in run mode

	Start program in run mode
	When teaching, the corresponding axis runs in the negative direction
	When teaching, the corresponding axis runs in the positive direction

Key switch

	On the left, switch to teaching mode
	In the middle, switch to run mode
	On the right, switch to remote mode

Emergency button

	Press emergency stop

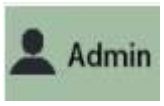
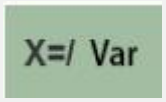
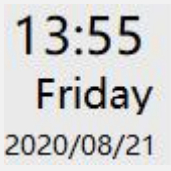
Wheel knob

	Program interface selection to switch the previous line and the next line

5.2. Introduction to Operating System

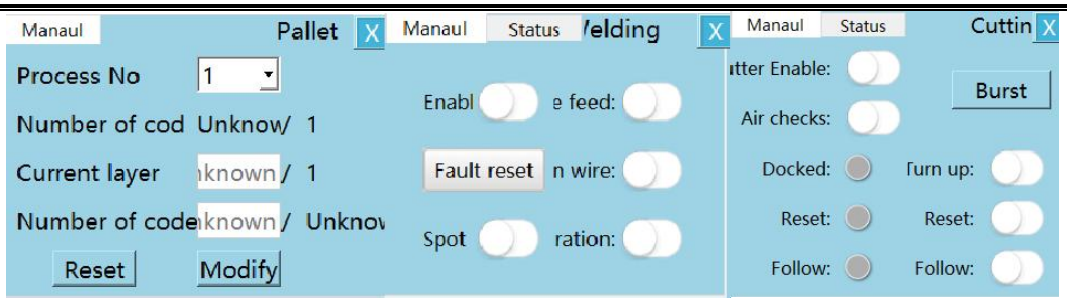
5.2.1. Basic description

This section provides an overview of the various parts of the program interface. The left side are the function keys and the other functions are shown in the table.

	Open the administrator/technician/operator switch interface.
	Open the robot function setting interface.
	Open the robot process selection interface.
	Open the robot variable setting interface.
	Open the robot status view interface.
	Open the project preview interface.
	Open the program command interface.
	Open the error log interface.
	Open the robot real-time display interface.
	Date and time display.

5.2.2. Status Introduction

The top of the program is the status bar, which displays the status of the robot.



Mode status: Switch by rotating {Mode Select Key}, includes Teach Mode, Remote Mode, and Play Mode.

Servo status: After starting the program, press the {Mot} key, then switch "Servo Ready" status to "Servo Stop" status.

In the teach mode, when press the {DEADMAN} key or run the program is run in the "reproduction mode" or the "cycle mode", the servo status is switched to the "servo run" status.

Program status: the current status of running program. When run program with single-step in tach mode or run program in "reproduction mode", "cycle mode", the status of program is switched to “running”.

Manual speed: Increases or reduces the manual speed by pressing {V+} {V-}, at the bottom of the teaching-programming pendant

Increase speed: Each time {V+} is pressed, the manual speed changes in the following order: inch 1% → inch 2% → low 5% → low 10%-medium 25% → medium 50% → fast 75%-fast 100% Reduce Speed: Each time {V-} is pressed, the manual speed changes in the following order: fast 100% → fast 75% → medium 50% → medium 25% low 10% → low 5% → inch 2% → inch 1% Robot status: Press {Rob} key at the bottom of the teaching-programming pendant.

There are two status "Robot a" and "Robot b" respectively.

Tool status: Press {Jog} key at the bottom of the demonstrator.

There are nine states of "tool 1", "tool 2", "tool 3", "tool 4", "tool 5", "tool 6", "tool 7", "tool 8" and "tool 9" respectively.

Process pattern: {In Teach Mode} Switched manually.

There are four states: "General", "Welding", "Stacking" and "Laser Cutting" respectively.

Frame system: Switched by pressing {Coordinate System Switch} on the left side of the teaching-programming pendant.

There are four coordinates:

“ joint coordinate system”， “Cartesian coordinate system” , “tool coordinate system” and “ user coordinate system ” respectively.

6. Robot teaching and running

6. 1. Robot Preparing

6. 2. Start up and Safety Confirmation

This section mainly describes the start up before the teach operation and the method to be sure the safety measures.

6. 2. 1. Start Up

Operation steps:

1.Check whether the connecting wires of servo, controller and teaching-programming pendant components are well connected.

2.Turn the main power switch on the cabinet panel to the ON position, and the main power is connected.

3.Press the green servo start button on the cabinet panel.

6. 2. 2. Safety confirmation

For safety reasons, please make sure that the emergency stop key is normal before teach.

Use confirmation of emergency stop button:

Before the robot is used, please make sure the emergency stop key on the control cabinet and teaching-programming pendant respectively. When pressed, the servo power supply is disconnected.

1.Press the emergency stop key on the control cabinet and teaching-programming pendant;

2.Be sure that the servo power is off, the teaching-programming pendant displays the servo error, and the control cabinet servo error lamp is on;

3.Clear the servo error, the servo error lamp in the control cabinet is off, and the "servo stop" is displayed on the teaching-programming pendant;

4.After be sure it is normal, press the {MOT} key on the teaching-programming pendant to make the servo in the servo preparation status;

6. 3. Preparation for Teaching-programming Pendant

After the teaching-programming pendant is started up and the servo is be sure to have no error, the teaching-programming pendant is be sure to be in the teach mode. If not, the key is rotated to select the mode and the teaching-programming pendant is switched to the teach mode. Press {MOT} key (servo preparation) on the teaching-programming pendant, and the "servo status" column at the top of

the program interface displays as "servo ready" and flashes 

.Only in “servo-ready” status can the robot be possible!

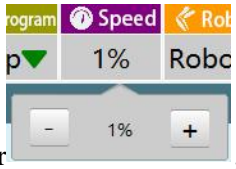
Press the orange {DEADMAN} key on the back of the teaching-programming pendant lightly, and the sound of the robot being powered up will be heard, and the "servo status" column will be displayed as the green "servo running", indicating that the servo power is successfully connected.

6.4. Point operation

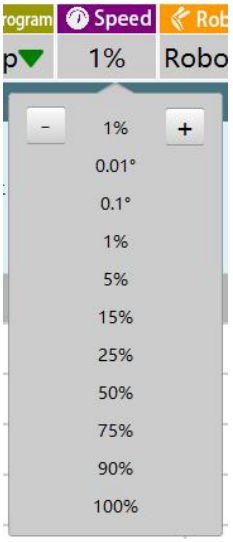
This section mainly describes the related matters of manual operation by using the physical keys of the teaching-programming pendant in the teach mode. It includes the definition and setting of coordinate system, the method of manual operation, the speed setting and the be sure of each status during manual operation. It takes a lot of practice to become proficient.

6.4.1. Teach Speed Control

In the teach mode, the motion speed of the manually manipulation robot is modified by pressing the {V+} (speed increase) key or {V-} (speed decrease) key on the hand-held manipulation teaching-programming pendant. For each press the manual speed changes in the following order, which is be sure by the speed display in the status area.

You can also click on the speed item in the status bar 

, which will bring up a drop-down menu. Clicking {-} and {+} can also add or subtract speed. Click on the middle number will pop up the speed option, you can quickly select several commonly used speed.



Speed increase: press the {V+} (speed increase) key at the bottom of the teaching-programming pendant. Each press, the manual speed will change in the following order:

Inching motion 0.01°→inching motion 0.1°→1\%→5\%→10\%→15\%→25\%→50\%→75\%→100\%

Speed down: press the {V-} (speed down) key at the bottom of the teaching-programming pendant. Each press, the manual speed will change in the following order:

High 100\%→high 75\%→mid 50\%→mid 25\%→low 15\%→low 10\%→low 5\%→micro motion 1\%→inching motion 0.1°→inching motion 0.01°

Inching motion: inching motion speed under the joint coordinate system is 0.01 ° and 0.1 ° two grades. In the rectangular, tool and user coordinate system, there are two grades of 0.1mm and 1mm. The teach speed is

based on the percentage, and the actual speed is the percentage in the status bar multiply the maximum speed of the point movement. The maximum speed of point movement is set in the setting-robot parameter-point movement interface, Please refer to the chapter of robot setup for detailed parameters and setting methods.

6.4.2. Description and Switching of Coordinate System

There are four coordinate systems in this product, namely joint coordinate system, rectangular coordinate system, tool coordinate system and user coordinate system.

- All points in the joint coordinate system are the angle values of the joint axis of the robot relative to the mechanical zero of the axis.
- Rectangular coordinate system is also known as the "base coordinate system". All points are the coordinate value (unit mm) of the robot tip (center of flange) relative to the center of the robot base.
- All points in the tool coordinate system are the coordinate value (unit mm) of the tool tip (TCP point) carried by the robot relative to the center of the robot base. See the chapter on Tool and User Coordinates for its definition and usage.
- The user coordinate system is also known as the " workpiece coordinate system", and all points are the coordinate values (unit mm) of the tool tip (flange center) of the robot relative to the origin of user coordinate system (without tool). See the chapter on Tool and User Coordinates for its definition and usage.
- In the teach mode, press the {Coordinate System Switch} key in the physical keys area on the left side of the teaching-programming pendant. Every time the key is pressed, the coordinate system will be switched in the following order and be sure by the display of the top status bar.
- You can also click on the coordinate system column of the status bar to pop up the coordinate system selection menu and switch by clicking on the corresponding coordinate system.

Joint→rectangle→tool→user

6.5. Point operation

1.To perform the pointing operation of the robot, the following steps are specified:

2.Starting up.

3.Check whether the emergency stop key is intact or pressed.

4.Press the {MOT} key of the teaching-programming pendant to be sure that the servo status is "servo preparation".

5.Select the coordinate system you want to use.

6.Adjust to the appropriate speed.

7.Press the {DEADMAN} key (the orange key on the back of the teaching-programming pendant), and do not release.

8.Use the keys in the physical key area on the right side of the teaching-programming pendant to operate the robot to move.

9.Release the {DEADMAN} key.

6.6. Programming

This section will mainly introduce the operation of the instructions of this product. It includes the operation of creating, modifying, deleting, copying and renaming programs, inserting, modifying, deleting and copying instructions, as well as the specific function description of each instruction, and provides specific examples. If you want to master it skillfully, you need to use it many times.

6.6.1. Program New/Open/Delete/Rename/Copy

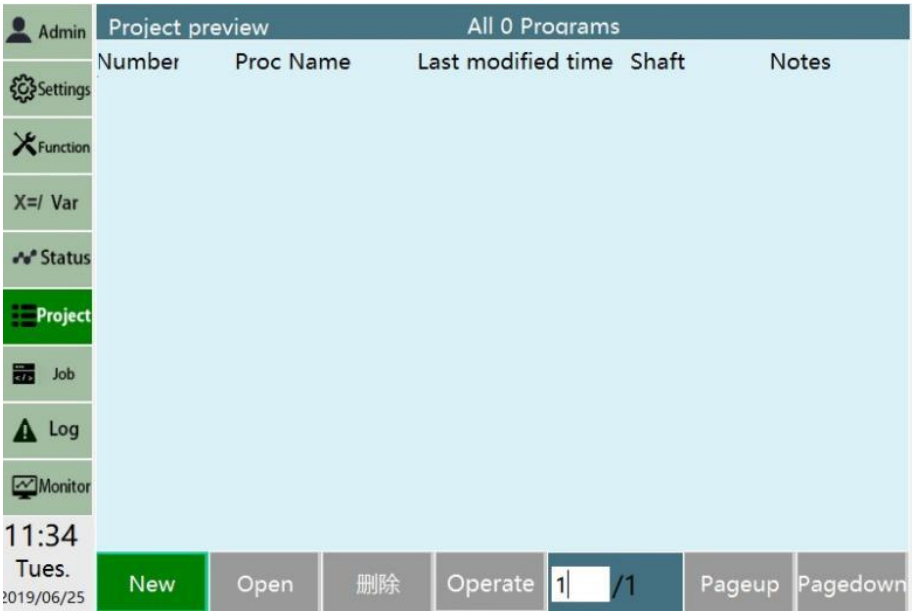
Users need to enter the program interface and use the bottom button to perform related operations to insert/modify/delete/copy/rename instructions of the program.

6.6.2. New Program

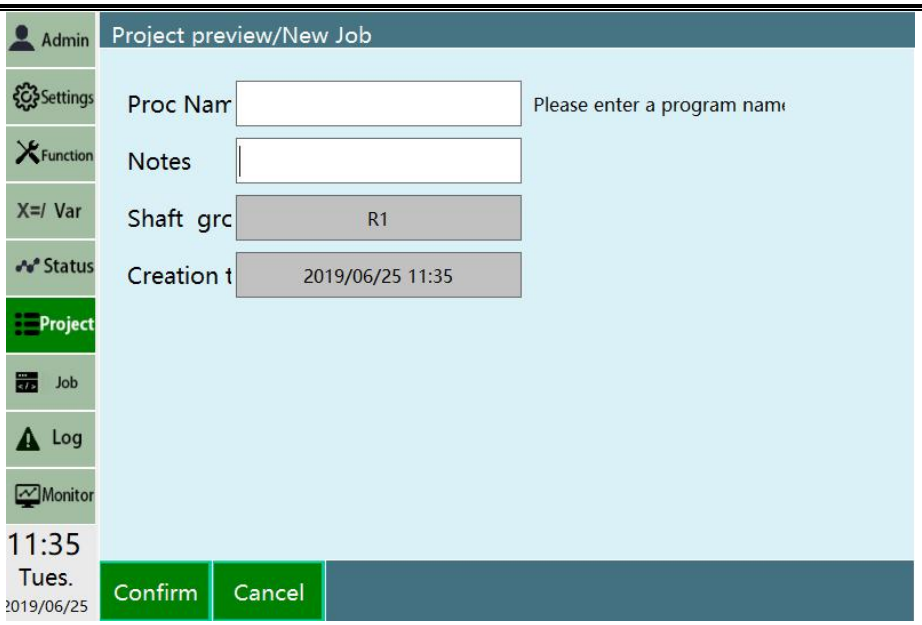
New programs need to be created by clicking on the {New} button at the bottom of the program interface.

The new program is under the selected program. The relevant steps are as follows：

1.Enter the program interface；



2.Enter the corresponding program name and other parameters in the pop-up "Program Creation" window.



2. Click on the {OK} button at the bottom, the program is created successfully, and jump into the new program. If you want to cancel the new program, click on the {Cancel} button.

CAUTION
The program name must be a string of two or more characters beginning with a letter. The program name cannot be the name of an existing program.

6. 6. 3. Program Open

To open an existing job file, the user needs the following steps:

1.Open the "engineering" interface;

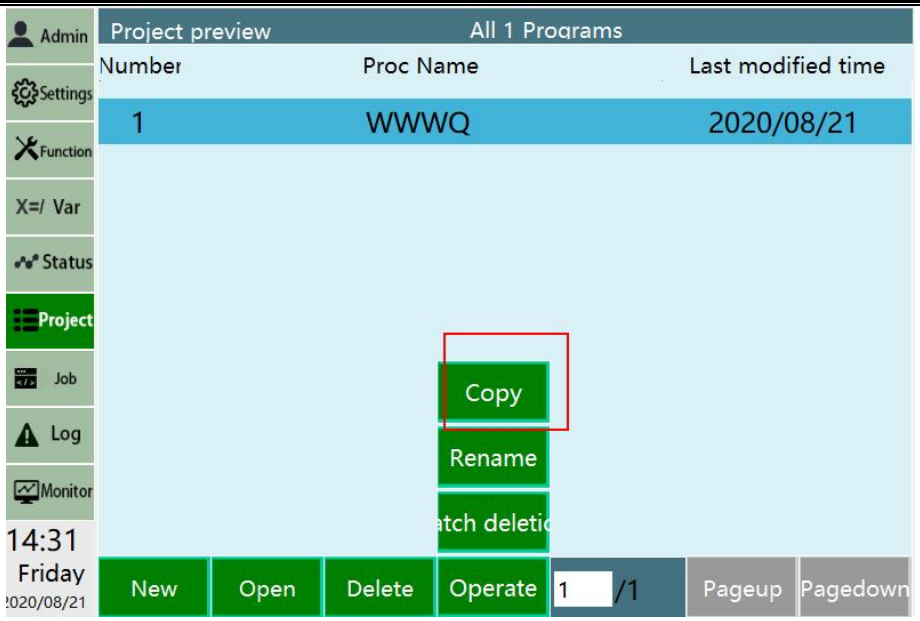
2.Select the program you want to open.

3.Click on the {Open} button at the bottom. The program was successfully opened.

6. 6. 4. Program Copy

To copy an existing job file (which can only be copied as a whole), users need to do the following steps:

1.Select the program to copy;



2.Click on the {Operation} button at the bottom and then click on the {Copy} button.

3.Click on {OK} otherwise {Cancel}; you can also change the name of the file.

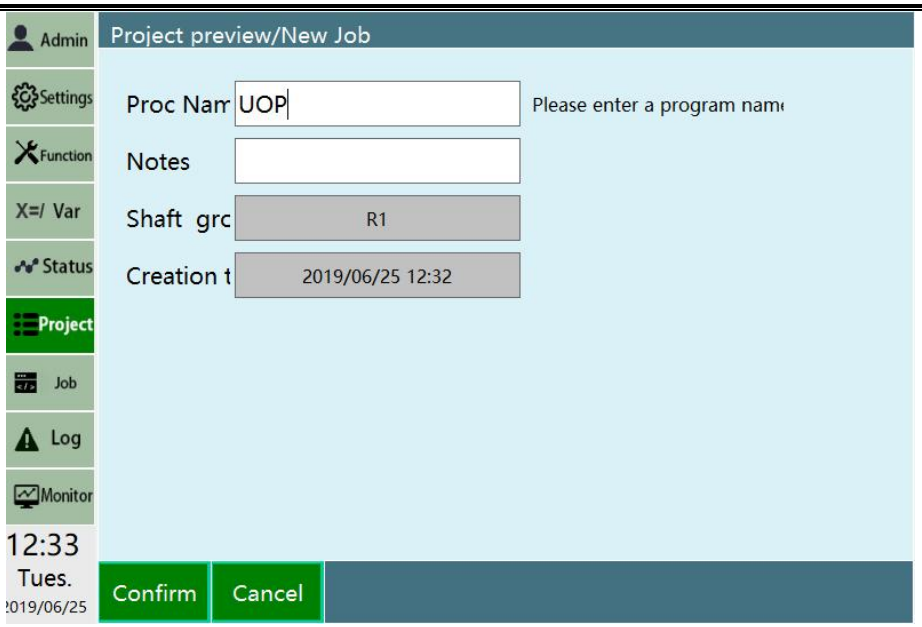
6. 7. Program Rename

The rename operation can modify the name of the selected program. The operation steps are as follows:

1.Select the program you want to rename.

2.Click on {Operation}, and then click on {Rename}.

3.Enter the name you want to modify in the pop-up window.



3. Click on the {OK} button. If you want to cancel the renaming operation, click on the {Cancel} button.

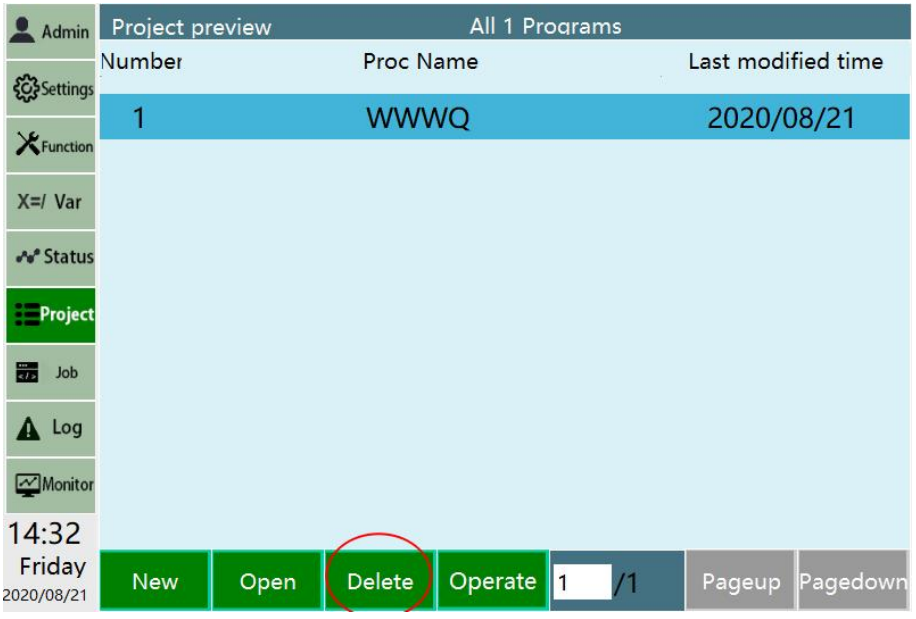
小心 CAUTION
The program name of a renamed program cannot be the name of an existing program.

6. 7. 1. Program Delete

Delete operation can delete the selected program. The relevant steps are as follows:

1.Select the program you want to delete.

2.Click on the {Delete} button;



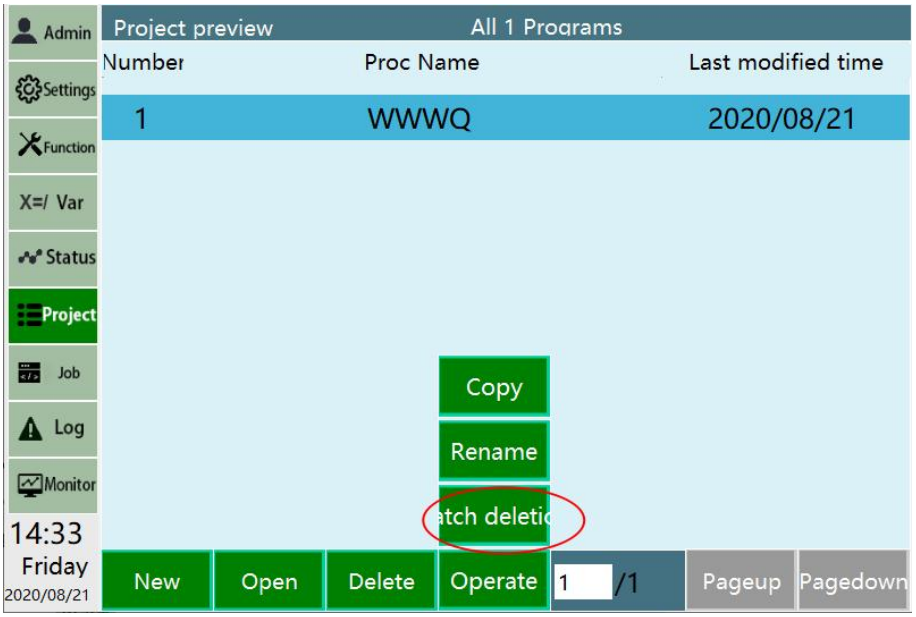
3.Click on the {OK} button in the pop-up window. If you want to cancel the deletion operation, click on the {Cancel} button.

6.7.2. Batch Delete

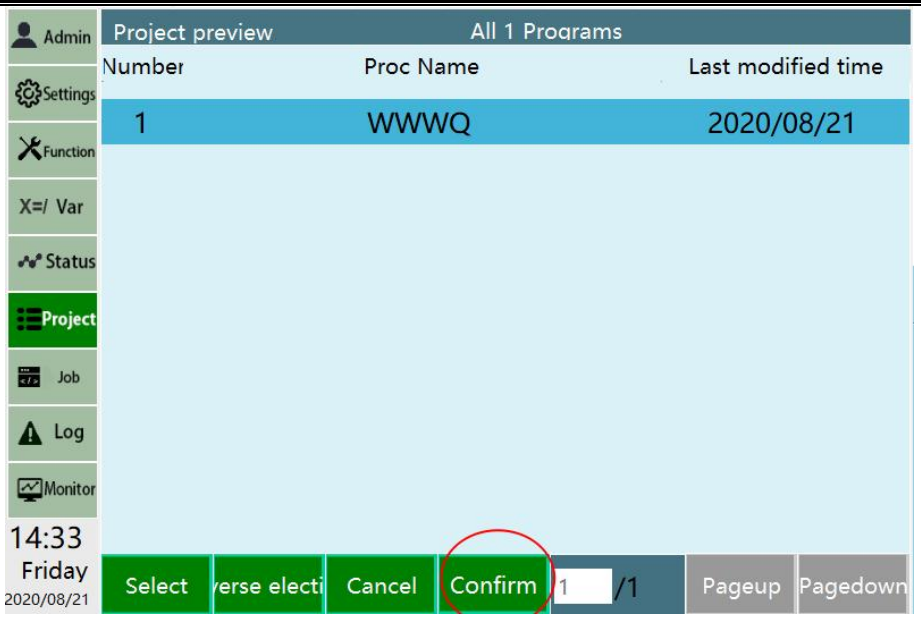
Batch deletion can delete more than one program file at a time. The method of use is as follows:

1.Enter the engineering interface;

2.Click on the operation of the bottom menu bar - {Batch Deletion} button;



3.Select the program files that need to be deleted (only the files on the current page can be selected, but not on the previous or next page). Click on the {Select All} button to select all the program files on this page;



4.Click on the {Confirmation Button} button and then pop up the confirmation box and click on the {Confirmation} button to delete the batch successfully.

6. 8. Instruction Operation

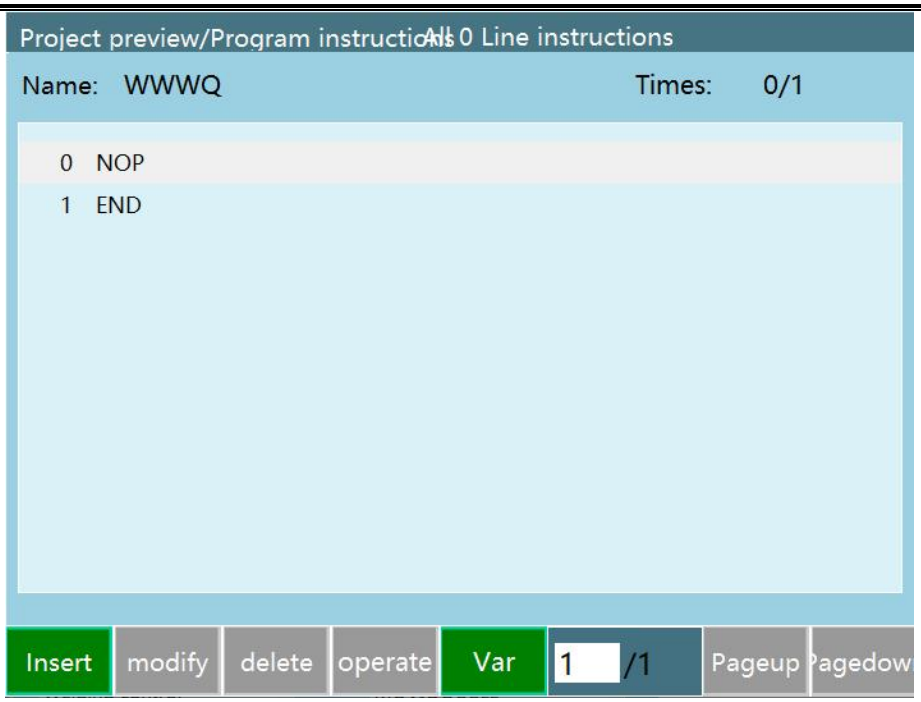
Users need to enter the program preview interface to perform instruction-related operations such as insertion/modification/deletion by using the bottom button.

6. 8. 1. Insertion

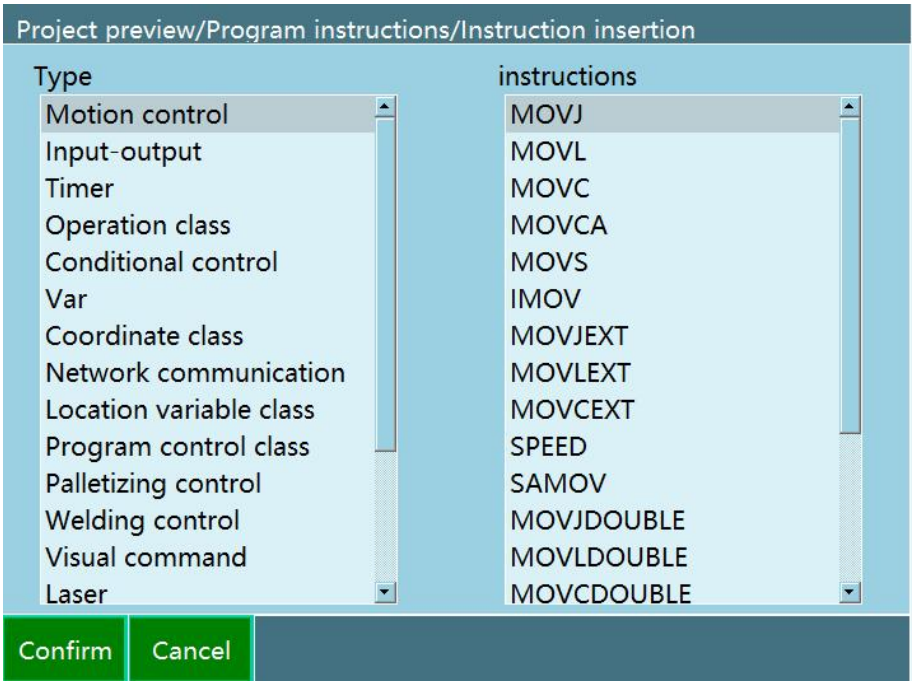
The insertion of instructions needs to be operated by using the {Instruction Menu} button at the bottom of the program preview interface.

Inserted instructions are below the selected instruction line The relevant steps are as follows:

1.Enter the program preview interface;

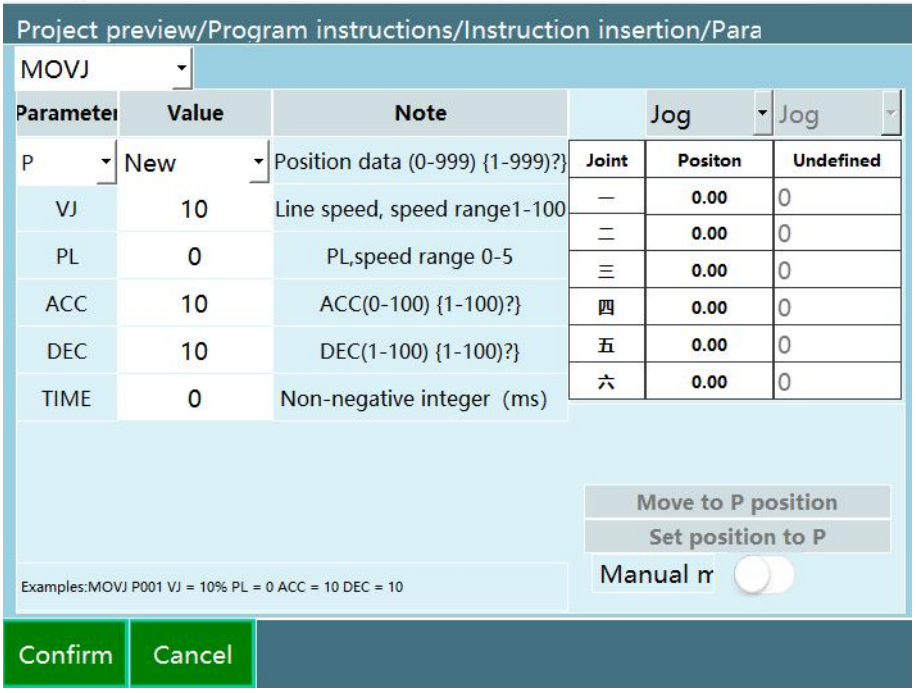


2.Click on the {Insert} button to pop up the instruction type menu;



3.Click on the type of instructions needed to insert instructions, such as motion control classes, as shown in the figure;

4.Click on the instructions you need to insert, such as MOVL, as shown in the figure;



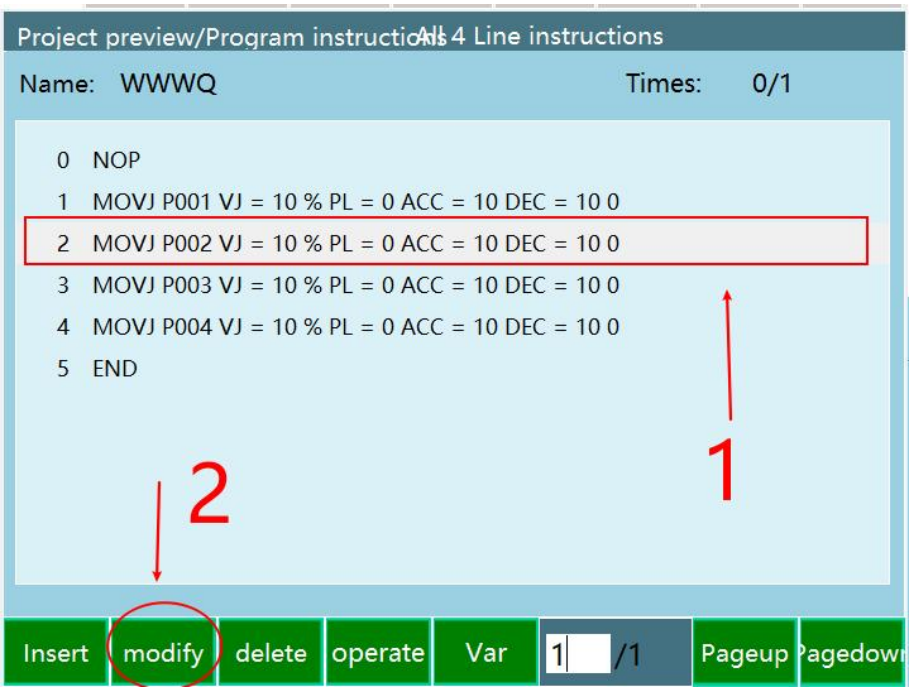
5.Set the relevant parameters of the inserted instructions;

6.Click on the {Confirmation} button at the bottom of the program;

6.8.2. Instruction Modification

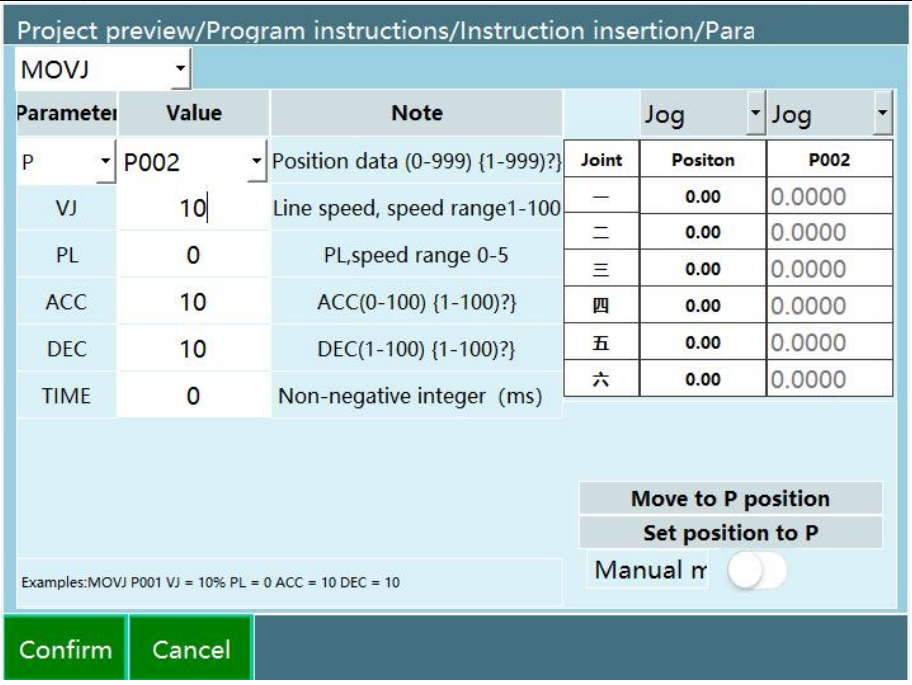
Users can easily modify the parameters of the inserted instructions by using the "modify" command. The steps to modify the instruction parameters are as follows:

1.Selected inserted rows (except NOP rows and END);



3.Click on the {Modify} button at the bottom of the program.

4.Modify the relevant parameters;



5.Click on {OK} button at the bottom when the modification is complete;

6.Successful instruction modification.

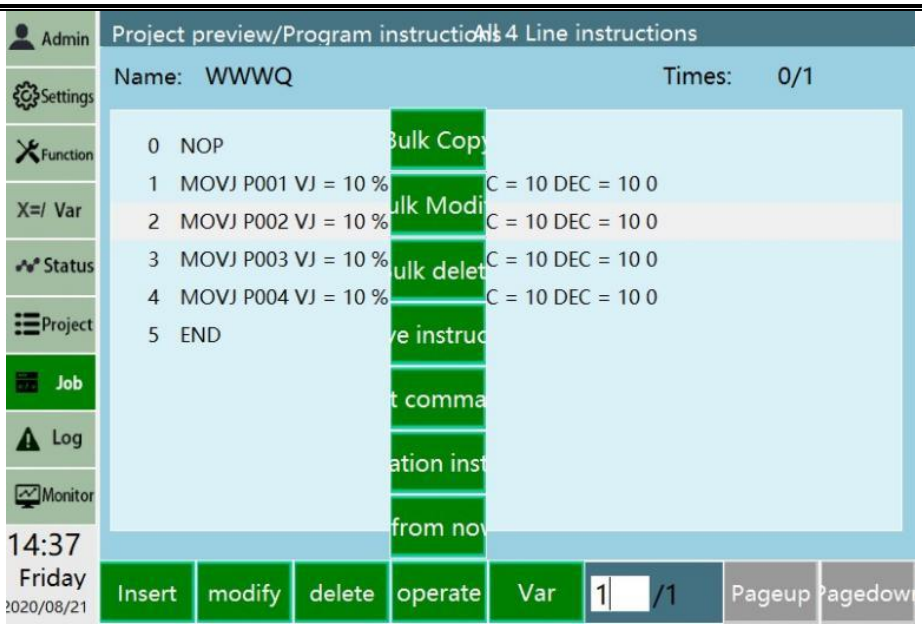
CAUTION
The program name of a renamed program cannot be the name of an existing program.

6.8.3. Batch Copy

Users can replicate the required instructions to the designated place through the "batch copy" operation. The steps are as follows:

1.First click on the bottom "Operation" button 

:



2.Select the required instructions:

3.Click on the {Confirm Copy} button, pop up the button below, and fill in the position you pasted:

6. 9. Instruction Description (Instruction Specification)

This section mainly describes the functions of the instructions and the functions and specifications of the relevant parameters. Provide some examples of specific application scenarios.

6. 9. 1. Motion Control Class

Motion control instructions include MOVJ, MOVL, MOVC, IMOV, NOVCA, MOVJEXT, MOVLEXT, MOVCEXT and other instructions.

When inserting these motion control instructions, if the P-point is selected as the new one, a new P-variable will be created automatically at the same time, and the current robot position will be written into the variable. Running the command runs to the position where the robot inserts the command.

The target points of all motion instructions in this system use position variables, local position variables are P, global position variables are G. See the chapter of position variable for the method of using specific position variable.

The functions and scope of each instruction and related parameters are as follows:

- MOVJ

When the robot moves to the target point, it is used in the non-trajectory constrained interval. If the joint is used to interpolate the teaching robot axis, the moving command is MOVJ.

For safety reasons, usually, use joint interpolation to teach the first step. The default speed is VJ = 10, which is the maximum speed of 10\%.

MOVJ	function	Moving to teach position by joint interpolation	
	parameters	position data, base axis position data, tool axis position data	Display in the interface
		VJ= reproduction speed	VJ：1-100

		PL= positioning level	PL：0~5
		NWALL	
		UNTIL	
		ACC= acceleration adjustment ratio	ACC：1-100
		DEC= deceleration adjustment ratio	DEC：1-100
	use example	MOVJ P001 VJ=10% PL=2 ACC=10 DEV=10	

- MOVL

Use the linear trajectory to move in the program point of the linear interpolation teach. If linear interpolation is used to teach the robot axis, the mobile command is MOVL.

Linear interpolation is often used in welding operations.

When linear interpolation is used, the wrist attitude of the robot remains unchanged.

MOVL	function	Moving to teach position by joint interpolation	
	parameters	position data, base axis position data, tool axis position data	Display in the interface
		V= reproduction speed	V：2-9999
		PL= positioning level	PL：0~5
		NWALL	
		UNTIL	
		ACC= acceleration adjustment ratio	ACC：1-100
		DEC= deceleration adjustment ratio	DEC：1-100
	use example	MOVL P001 V=100 PL=2 ACC=10 DEV=10	

- MOVC

The robot moves through three dotted circles taught by arc interpolation.

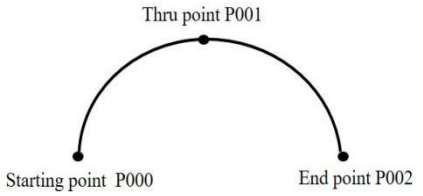
If arc interpolation is used to teach the robot axis, the mobile command is MOVC.

The starting point of the first arc of a single arc and a continuous arc can only be MOVJ or MOVL.

- Single arc

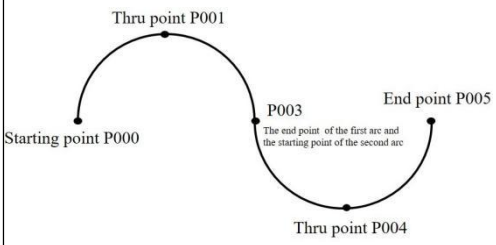
When there is only one arc, as shown in the table below, three points of P1-P3 are taught by arc interpolation.

If P0 before entering the arc is taught by joint interpolation or linear interpolation, the trajectory of P0-P1 will automatically become a straight line.

	point	Interpolation method	command
	P000	joint linear	MOVJ MOVL
	P001-P002	arc	MOVC

■ Continuous arc

As shown in the table below, when there are more than two consecutive arcs whose curvature changes, the arcs will eventually be separated one by one. Therefore, as shown in Figure 4, join the points of joint and linear interpolation at the connection point between the former arc and the latter arc.

	point	Interpolation method	command
	P000	Joint linear	MOVJ MOVL
	P001-P002	arc	MOVC
	P003-P004	arc	MOVC

MOVC	function	Arc interpolation moves to the target position. The three-point arc method is adopted. The first point before the arc is the first point, and the two MOVCs are the middle point and the target point. Note: The first motion control class instruction of the job file cannot be MOVC.	
	parameters	position data, base axis position data, tool axis position data	Display in The interface
		V= reproduction speed	V：2-9999
		PL= positioning level	PL：0~5
		NWALL	
		UNTIL	
		ACC= acceleration adjustment ratio	ACC：1-100

		DEC= deceleration adjustment ratio	DEC：1-100
	use example	MOVC P001 V=100 PL=2 ACC=10 DEV=10	

- IMOV

IMOV	function	Move by the joint position or linear interpolation from the current position according to the set incremental value distance	
	parameters	B= position data	BF：base coordinates RF：robot coordinates TF：tool coordinates UF：user coordinates
		V= reproduction speed	V：2-9999
		PL= positioning level	PL：0~5
		user coordinates	Display B parameter status
		UNTIL	
		ACC= acceleration adjustment ratio	ACC：1-100
		DEC= deceleration adjustment ratio	DEC：1-100
	use example	IMOV B001 V=100 PL=2 ACC=10 DEV=10	

- MOVS

In welding, cutting, welding, primer painting and other jobs, if the use of free curve interpolation, for irregular curve workpiece teaching jobs can be easy.

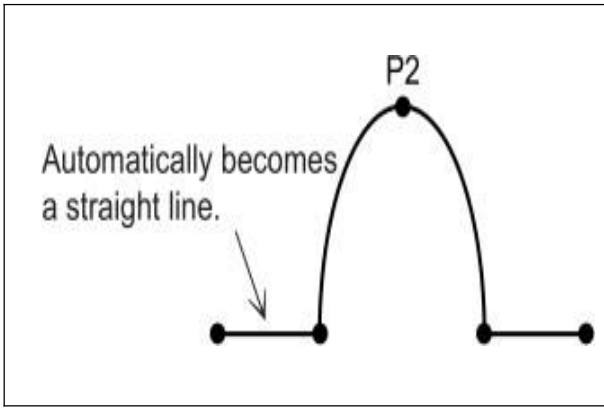
The trajectory is a parabola passing through three points,.

If the free curve is used to interpolate the teaching robot axis, the moving command is MOVS.

■ Single free curve

As shown in the table below, the three points of teaching P1-P3 are interpolated with free curve.

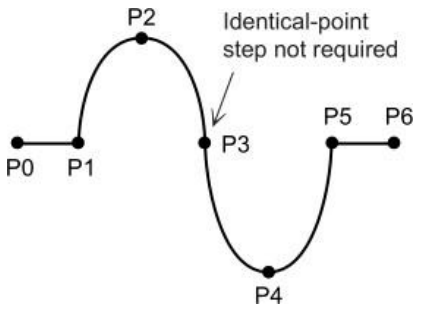
If joint interpolation or linear interpolation is used to teach the P0 point before entering the free curve, the trajectory of P-P1 will automatically become a straight line.



point	Interpolation method	command
P000	Joint linear	MOVJ MOVL
P001-P003	free curve	MOVS
P004	Joint linear	MOVJ MOVL

■ Continuous free curve

The trajectory is established by coincidence parabola synthesis.。

	point	Interpolation method	command
	P000	Joint linear	MOVJ MOVL
	P001-P005	free curve	MOVS
	P006	joint	MOVJ MOVL

Unlike arc interpolation, the joint of two free curves cannot be the same or have no other instructions.

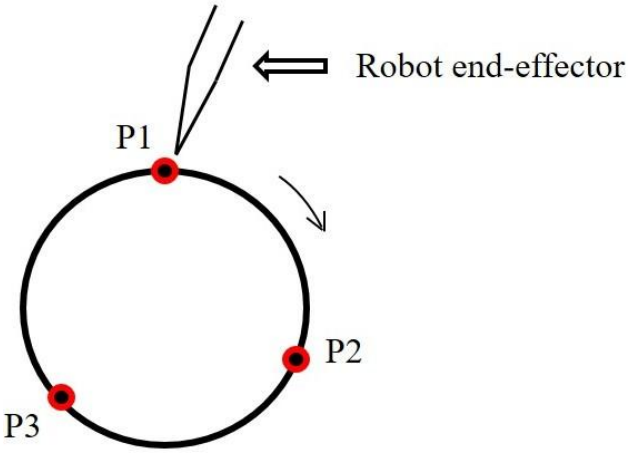
Establishment of synthetic trajectory in the case of coincidence parabola.

MOVS	function	Move to the teaching position in the form of free curve interpolation.	
	parameters	position data, base axis position data, tool axis position data	Display in the interface
		V= reproduction speed	V：2-9999
		PL= positioning level	PL：0~5
		NWALL	
		UNTIL	
		ACC= acceleration adjustment ratio	ACC：1-100
		DEC= deceleration adjustment ratio	DEC：1-100
	use example	MOVS P001 V=100 PL=2 ACC=10 DEV=10	

- MOVCA

To teach the robot to walk a complete circle, the mobile command is MOVCA. Instruction insertion premise

Click on the {Tool} button in the status bar above and select the tool that has been calibrated before.



Insertion steps, a total of four instructions.

{movca}Click on the {Insert}, click on the {Coordinate Switching Class}, select SWITCHTOOL, and select the tool number previously calibrated.

Move to any point of the circle you want to draw as shown in Figure P1, click on the {Insert}, click on

the {Motion Control Class}, and select {movj} or {movl};

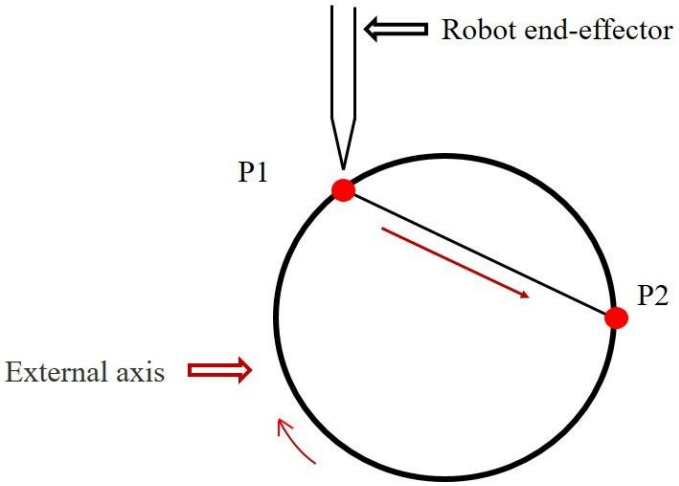
Move to any point of the circle you want to draw as shown in Figure P2 (unlike the point in Step 2). Click on the {Coordinate System} button in the upper status bar, select the "Tool" coordinate system, click on the {Insert}, click on the {Motion Control Class}, and select {movca}.

Move to any point of the circle you want to draw as shown in Figure P3 (unlike the point in Step 2 or 3). Click on the {Coordinate System} button in the upper status bar, select the "Tool" coordinate system, click on {Insert}, click on {Motion Control Class}, and select .

MOVCA	function	Based on the principle of determining a circle by three points, draw a circle. The three-point circle drawing method is adopted. The first point is in front of the circle and the two MOVCAs are in the middle of the circle. Note: The first motion control class instruction of the job file cannot be MOVCA.	
	parameters	position data, base axis position data, tool axis position data	Display in the interface
		V= reproduction speed	V：2- 9999
		PL= positioning level	PL：0~5
		NWALL	
		UNTIL	
		ACC= acceleration adjustment ratio	ACC：1-100
		DEC= deceleration adjustment ratio	DEC：1-100
	use example	MOVCA P001 V=100 PL=2 ACC=10 DEV=10	

- MOVJEXT

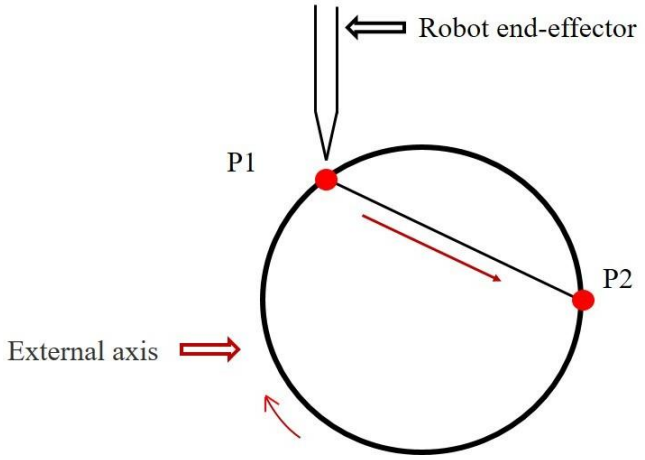
The robot moves to the teaching position by means of joint interpolation, and the external axis is compensated by joint difference.



MOVJEXT	function	The robot moves to the teaching position by means of joint interpolation,and the external axis is compensated by joint difference.	
	parameters	position data, base axis position data, tool axis position data	Display in the interface
		VJ= reproduction speed	VJ：1-100
		PL= positioning level	PL：0~5
		NWALL	
		UNTIL	
		ACC= acceleration adjustment ratio	ACC：1-100
		DEC= deceleration adjustment ratio	DEC：1-100
	use example	MOVJEXT P001 VJ=10% PL=2 ACC=10 DEV=10	

- MOVLEXT

The robot moves to the teaching position by linear interpolation, and the external axis moves by joint difference compensation.

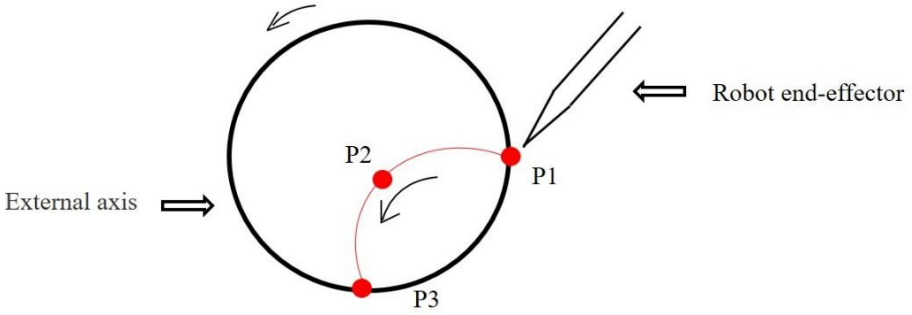


MOVLEXT	function	The robot moves to the teaching position by arc interpolation, and the external axis is compensated by joint difference.

	parameters	position data, base axis position data, tool axis position data	Display in the interface
		V= reproduction speed	V：2- 9999
		PL= positioning level	PL：0~5
		NWALL	
		UNTIL	
		ACC= acceleration adjustment ratio	ACC:1-100
		DEC= deceleration adjustment ratio	DE：1-100
	Use example	MOVL P001 V=100 PL=2 ACC=10 DEV=10	

- MOVCEXT

The robot moves to the teaching position by arc interpolation, and the external axis is compensated by joint difference.



MOVCEXT	function	The robot moves to the teaching position by arc interpolation, and the external axis is compensated by joint difference.	
	parameters	position data, base axis position data, tool axis position data	Display in the interface
		V= reproduction speed	V：2-9999
		PL= positioning level	PL：0~5
		NWALL	
		UNTIL	
		ACC= acceleration adjustment ratio	ACC：1-100

		DEC= deceleration adjustment ratio	DEC：1-100
	use example	MOVCEXT P001 V=100 PL=2 ACC=10 DEV=10	

- SAMOV

Robots move to a set absolute position by joint interpolation.

If you don't want to move an axis, leave it blank at its coordinates. (Do not fill in 0)

SAMOV	function	Robots move to a set absolute position by joint interpolation.	
	parameters	B=position data	BF：base coordinates RF：robot coordinates TF：tool coordinates UF：user coordinates
		V= reproduction speed	V：2-9999
		PL= positioning level	PL：0~5
		user coordinates	Display B parameter status
		UNTIL	
		ACC= acceleration adjustment ratio	ACC：1-100
		DEC= deceleration adjustment ratio	DEC：1-100
	Use example	SAMOV B001 V=100 PL=2 ACC=10 DEV=10	

- SPEED

The movement speed of all the motion instructions below the SPEED instruction is as follows: the instruction speed * the speed of the upper status bar * the percentage of SPEED.

SPEED	function	setting global speed	
	parameters	globalspeed（%）	speed percentage： 1-200
	use example	SPEED 200	

6.9.2. Input and Output Classes

- DIN

Write the input status of the current digital IO input port into a variable.

DIN	function	Read the digital input status into a variable.	
	parameters	variable sources	INT、DOUBLE、BOOL、GINT、GDOUBLE

		variable name	1-999	
		input number	IN# IGH# IG#	group number 1-16 group number 1-4 group number 1-2
	use example	DIN A001 IN#（1）		

- DOUT

Set the output value of the current digital IO.

DOUT	function	Move to the teaching position in the form of free curve interpolation.	
	parameters	output mode	1 channel output（OT#) 4 channel output（OGH#） 8 channel output（OG#）
		variable sources	manual selection INT DOUBLE BOOL GINT GDOUBLE GBOOL
		variable name	1-999
	use example	DOUT OT#（1） I001	

- PULSEOUT

PULSEOUT	function	Control pulse output	
	parameters	frequency	1-100000
		number	Integers greater than 0
	use example	PULSEOUT RATE=100 SUM=100	

6.9.3. Timer Class

- TIMER

Timing

TIME R	function	delay	
	parameters	time	0-9999s
	use example	TIMER=100s	

6.9.4. Conditional Control Class

Conditional motion control class instructions include JUMP, IF, ELSEIF, WAIT, WHILE and other instructions. The functions and scope of each instruction and related parameters are as follows：

- JUMP

JUMP instructions should be combined with LABEL. When the JUMP instructions are executed, when the program meets the requirements, jump to the designated LABEL to execute the program below LABEL.

JUMP	function	When conditions are met, jump to the specified LABEL		
	parameters	label name	LABEL	
		judging condition	parameter type	INT DOUBLE BOOL etc
			Parameter name	0-999 integer
			comparison modes	==、<、>、<=、>=、!=
			variable value source	customization or other variables
			new parameter	value
			Source parameters	existing variables
	use example	JUMP *D1 WHEN（A001=4）		

- CALL

When executing the CALL instruction of job file A, jump to the job file B to which the CALL instruction refers. After the execution of job file B, jump back to job file A, and continue to execute the next instruction of CALL instruction.

When the last instruction in the job file is CALL, after executing the job file B to which the CALL instruction refers, the job file A is jumped back and the program stops.

CALL	function	Call a program with a specified name
	parameter	program name
	Use example	CALL JOB1

- IF

IF judgment statement, judging whether the condition is valid, if it is valid, run the program between IF and ENDIF, otherwise jump out.

	function	Judging whether the condition is valid or not, it runs if it is valid, otherwise it will jump out.		
		judging condition	parameter type	INT DOUBLE BOOL etc
			Parameter name	0-999 integer
			comparison modes	==、<、>、<=、>=、!=
			new parameter	value
			source parameters	existing variables
	Use example	IF（I003 == 1）		

- ELSEIF

When the judgment condition of IF is not valid, if the condition of ELSEIF is valid, run the program between ELSEIF and ENDIF, if not, jump out.

LSEI F	function	When the judgment condition of IF is not valid, if the condition of ELSEIF is valid, run the program between ELSEIF and ENDIF, if not,jump out.		
		judging condition	parameter type	INT DOUBLE BOOL etc
			parameter name	0-999 integer
			comparison modes	==、<、>、<=、>=、!=
			new parameter	value
			Source parameters	existing variables
	use example	ELSEIF（I003 == 1）		

- ELSE

Run the program between ELSE and ENDIF when the judgment condition of IF is not valid.

- WAIT

WAIT waits for instructions when conditions are not met; otherwise it does not wait.

WAIT	function	WAIT waits for instructions when conditions are not met;otherwise it does not wait.		
	parameters	judging condition	parameter type	INT DOUBLE BOOL etc
			parameter name	0-999 integer
			comparison modes	==、<、>、<=、>=、!=
			variable value source	customization or other variables
			new parameter	value
			Source parameters	existing variables
			TIMER	waiting time
	use example	WAIT （I001 == 2） T=2s		

- LABEL

Need to be used in conjunction with JUMP. JUMP instructions jump to the LABEL instructions.

LABEL	function	Label where needed, JUMP call	
	parameters	label name	Composition of letters and numbers,no more than 8 digits
	use example	LABEL *M1	

- WHILE

When the conditional statement is valid, the WHILE loop is executed, otherwise the WHILE is skipped.

WHILE	function	When the conditional statement is valid, the WHILE loop is executed, otherwise jump out.		
	parameters	judging condition	parameter type	INT DOUBLE BOOL etc
			Parameter name	0-999 integer

			comparison modes	==、<、>、<=、>=、!=
			variable value source	customization or other variables
			new parameter	value
			source parameters	existing variables
	use example	WHILE (I001==1)		

6.9.5. Arithmetic Operations Class

Arithmetic operation instructions include ADD, SUB, MUL, DIV, MOD and other instructions. The functions and scope of the instructions and related parameters are as follows.

- ADD

ADD	function	Data 1 is added to data 2, and the results are stored in data 1. Format:: ADD <Data 1> <Data 2>		
	parameters	Data1	BOOL INT DOUBLE GINT GDOUBLE GBOOL	Data 1 is a variable
		Data2	Constant BOOL INT DOUBLE GINT GDOUBLE GBOOL	Data 2 is a constant or variable
	use example	ADD I002 3		

- SUB

SUB	function	Data 1 is subtracted from data 2, and the results are stored in data 1.Format: SUB <Data 1> <Data 2>

	parameters	Data1	BOOL INT DOUBLE GINT GDOUBLE GBOOL	Data 1 is a variable
		Data2	Constant BOOL INT DOUBLE GINT GDOUBLE GBOOL	Data 2 is a constant or variable
	use example	SUB I002 3		

- MUL

	function	Data 1 is multiplied by data 2, and the results are stored in data 1.Format: MUL <Data 1> <Data 2>		
MUL	parameters	Data1	BOOL INT DOUBLE GINT GDOUBLE GBOOL	Data 1 is a variable
		Data2	Constant BOOL INT DOUBLE GINT GDOUBLE GBOOL	Data 2 is a constant or variable
	use example	MUL I002 3		

- DIV

DIV	function	Data 1 is divided by data 2, and the results are stored in data 1. Format: DIV <Data 1> <Data 2>

	parameters	Data1	BOOL INT DOUBLE GINT GDOUBLE GBOOL	Data 1 is a variable
		Data2	Constant BOOL INT DOUBLE GINT GDOUBLE GBOOL	Data 2 is a constant or variable
	use example	SUB I002 3		

- MOD

MOD	function	Data 1 is divided by data 2, and the remainder is stored in data 1.Format: MOD <Data 1> <Data 2>		
	parameters	Data1	BOOL INT DOUBLE GINT GDOUBLE GBOOL	Data 1 is a variable
		Data2	Constant BOOL INT DOUBLE GINT GDOUBLE GBOOL	Data 2 is a constant or variable
	use example	SUB I002 3		

6.9.6. Welding Control Class

- ARCON

ARCON	function	welding start

	parameter	ARCON statement	Welding process number
	Use example	ARCON #3	

ARCOFF Welding end statement

- ARCSET

ARCSET	function	welding settings	
	parameters	Data1	V= voltage value
		Data2	V= current value
	use example	ARCSET V=10 A=10	

- WVON

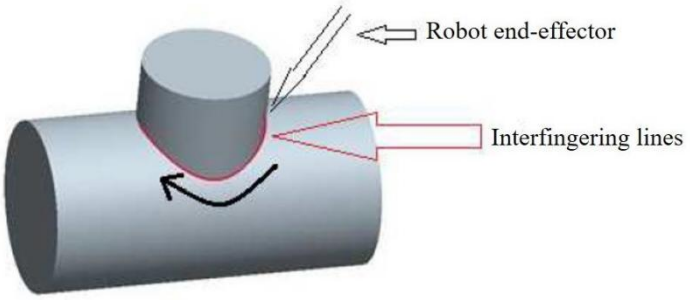
WVON	function	pendulum welding start	
	parameters	welding process number	Process number
	use example	WVON #3	

- WVOFF

End of pendulum welding

- CIL

To walk along a intersecting line (this article refers specifically to the intersecting line of two cylinders), the move command is CIL.



IL	function	Completion of a intersection line based on three-point motion	
	parameters	position data, base axis position data, tool axis position data	Display in the interface
		V= reproduction speed	V：2-9999
		PL= positioning level	PL：0~5
		ID	1-3
		UNTIL	

		ACC= acceleration adjustment ratio	ACC：1-100
		DEC= deceleration adjustment ratio	DEC：1-100
	use example	CIL P000 V=500mm/s PL=2 ACC=1 DEC=1 ID=1	

6.9.7. Palletizing Control Class

- PALLET

Declare the program as a palletizing instruction, usually inserted in the head of the program.

If the palletizing program stops at half of the palletizing, please modify the PALLET instructions and fill in the number of currently coded workpieces at the number of workpieces, and the program will automatically continue the palletizing.

PALLET	function	pendulum welding start	
	parameters	welding process number	process number
		palletizing type	palletizing unloading
		number of eorkpieces coded	0-9999
	Use example	PALLET ID=4 TYPE=0 NUM=1	

- PALENTER

Linear interpolation is used to run to the entry point of the workpiece in the palletizing process, and the position is set at the position parameters of the palletizing process.

PALENTER	function	Running to the entry point of palletizing in the form of linear interpolation	
	parameters	palletizing process number	process number
		V speed	V：2-9999
		PL= positioning level	PL：0~5
		ACC= acceleration adjustment ratio	ACC：1-100
		DEC= deceleration adjustment ratio	DEC：1-100
	use example	PALENTER ID=1 V=500mm/s PL=2 ACC=1 DEC=1	

- PALSHIFT

Linear interpolation is used to run to the workpiece auxiliary point of the palletizing process, and the position is set at the position parameters of the palletizing process.

PALSHIFT	function	Running to the palletizing auxiliary point in the form of linear interpolation	
	parameters	palletizing process number	Process number
		V speed	V：2-9999
		PL= positioning level	PL：0~5
		ACC= acceleration adjustment ratio	ACC：1-100

		DEC= deceleration adjustment ratio	DEC：1-100
	use example	PALSHIFT ID=1 V=500mm/s PL=2 ACC=1 DEC=1	

- PALREAL

Linear interpolation is used to run to the placement point of the palletizing process, and the position is set at the position parameters of the palletizing process.

PALREAL	function	Running to the palletizing point in the form of linear interpolation	
	parameters	palletizing process number	process number
		V speed	V：2-9999
		PL= positioning level	PL：0~5
		ACC= acceleration adjustment ratio	ACC：1-100
		DEC= deceleration adjustment ratio	DEC：1-100
	use example	PALREAL ID=1 V=500mm/s PL=2 ACC=1 DEC=1	

- PALEND

Determine whether the palletizing is completed, and if so, set a BOOL variable to 1.

PALEND	function	Determine whether the palletizing is completed, and if so,set a BOOL variable to 1.	
	parameters	palletizing process number	process number
		variable name	BOOL variable name
	use example	PALEND ID=1 A001	

6.9.8. Variable Class

Variable class instructions include INT, DOUBLE, BOOL, SETINT, SETDOUBLE, SETBOOL instructions. The functions and scope of the instructions and related parameters are as follows：

- INT

Defining and assigning a local INT variable requires inserting instructions into the program header.

INT	function	Define local INT variables and assignment	
		variable name	0-999

	parameters	variable value source	Constant INT DOUBLE BOOL GINT GDOUBLE GBOOL
		new parameter	constant
		source parameter	existing variable name
	use example	INT I001=1	

Defining and assigning a local DOUBLE variable requires inserting instructions into the program header.

DOUBLE	function	Define local DOUBLE variables and assignment	
	parameters	variable name	0-999
		variable value source	Constant INT DOUBLE BOOL GINT
			GDOUBLE GBOOL
		new parameter	constant
		source parameter	existing variable name
	use example	DOUBLE D001=1	

- BOOL

Defining and assigning a local BOOL variable requires inserting instructions into the program header.

BOOL	function	Define local DOUBLE variables and assignment	
	parameters	variable name	0-999

		variable value source	Constant INT DOUBLE BOOL GINT GDOUBLE GBOOL
		new parameter	constant
		source parameter	existing variable name
	use example	BOOL A001=1	

- SETINT

  Assignment to INT variables.

SETINT	function	Assign INT variables.	
	parameters	variables	INTGINT
		variable value source	Constant INT DOUBLE BOOL GINT GDOUBLE GBOOL
		new parameter	constant
		source parameter	existing variable name
	use example	SETINT I001=1	

- SETDOUBLE

  Assignment to DOUBLE variables.

SETDOUBLE	function	Assignment to DOUBLE variables.	
	parameters	variables	INTGINT
		variable value source	Constant INT DOUBLE BOOL GINT GDOUBLE GBOOL

		new parameter	constant
		source parameter	existing variable name
	use example	SETDOUBLE D001=1	

- SETBOOL

Assignment to BOOL variables.

SETBOOL	function	Assignment to BOOL variables.	
	parameters	variables	INT GINT
		variable value source	Constant INT DOUBLE BOOL GINT GDOUBLE GBOOL
		new parameter	constant
		source parameter	existing variable name
	use example	SETBOOL A001=1	

- FORCESET

In the process of running the program, the global variable value in the current cache is written into the variable file.

FORCESET	function	In the process of running the program, the global variable value in the current cache is written into the variable file.	
	parameters	variable type	GINT GDOUBLE GBOOL
		variable name	variable name
	use example	FORCESET GI001	

6.9.9. Coordinate transformation class

Coordinate transformation class includes two kinds of transformation: SWITCHTOOL and SWITCHUSER.

- SWITCHTOOL

When the robot is running, it is necessary to use the command to switch the tool coordinate system after replacing the tool with the wrist.

SWITCHTOOL	function	Switching tool coordinate system during program running

	parameters	tool coordinate number (1-3) & none
	use example	SWITCHTOOL 1

- SWITCHUSER

When the workbench is replaced during the running of the robot, the user coordinate system needs to be switched with this instruction.

SWITCHUSER	function	Switching user coordinate system during program running
	parameters	user coordinate number (1-5) & none
	use example	SWITCHUSER 1

6. 10. Program Running

The program can run in three modes, including "single step", "running", "remote", which correspond to "teach mode", "play mode" and "remote mode". Users can switch between "teach mode", "reproduction mode" and "circular mode" by using the mode selection key on the left of the teaching- programming pendant.

Figure Mode selection key

6. 10. 1. Teach Mode

In the teach mode, it can complete the point operation of the robot, job file programming, system parameter setting and other operations. In the process of job file programming, "STEP" function can be used to operate the job file step by step.

6. 10. 2. Be sure trajectory using STEP

After the user has selected the inserted command line, he can only run the selected command line by pressing the {DEADMAN} key and clicking on the {STEP} key in the physical key area at the bottom of the teaching-programming pendant to operate the job file in one step \ textbf {(do not release the {DEADMAN} key during the robot movement)} single step operation.

STEP running speed = instruction speed * speed ratio of the status bar above. The specific steps are as follows：

1.Select the instruction line to perform a one-step operation.

2.Press the {DEADMAN} key and the robot will power up.

3.Press the {STEP} key, and the robot executes the command of the selected line and stops after execution.

4.The selected line is automatically moved down, and press the {STEP} key again if you want to run the next line of instruction in one step.

6.10.3. Play Mode

In the play mode, you can click on the {Number of Runs} button in the lower left corner to set the number of runs of the program. Run once by default. Clicking on the }{Loop Run} button in the pop- up window can make the program run indefinitely.

In the play mode, the number of running times and the total number of running times are displayed above the program, and the format is "number of times run / total number of times run".

During the running process, the number of runs can be modified. After the modification, the robot stops after running the set number of times. For example, if the original setting is run 200 times and 156 times have been run, the number of times the setting is set to 3 times, the robot stops after continuing to run three times. \\

Run speed = command speed * speed ratio of the status bar above.

6.11. Remote Mode

Remote mode supports two kinds of peripheral equipments, digital IO and Modbus touch screen. When the controller is activated after the teaching-programming pendant is removed, it will automatically enter the remote mode. The equipment priority is: Modbus > Digital IO. When two peripheral equipments are connected, the digital IO enable can be controlled through the Modbus touch screen.

6.11.1. Reservation Mode

The reservation mode uses digital IO to control the running of the program. The mechanism is to set (reserve) in advance in the remote mode, through the program to be started by IO, the number of runs and number it. After switching to remote mode, the set program is sorted by IO signal. After pressing the {Running} button, the program will run according to the sorted program and the number of runs. After all the programs have been run, the operation stops. If you need to run again, you need to reorder.

If a single program is required to run indefinitely, the number of times it runs is set to 0 at the time of reservation.

The steps of the reservation procedure are as follows：

1.Enter Settings - Remote Program Settings；

2.Set up five programs for reservation and the number of runs；

3.Set the functions of each IO input port in the IO-IO function, where Program 1 - Program 5 corresponds to the sorting function of the five programs in the remote program setting interface;

4.Switch to remote mode；

5.Give the IO corresponding to the serial number of the program a high level (set to high level valid) lasting 2 seconds, then release it, and the program enters the queue；

6.If you want to cancel the sorting of a program after the sorting is completed, then give the IO corresponding to the program serial number a high level (set to high level valid) which lasting 2 seconds and release again；

7.Give the corresponding IO port a rising edge to start the program (set to high level valid), and

the system starts to run according to the number of programs in the queue；

8.Sorting and canceling queues can also be performed during the run.

If the reservations are switched on, the first reservations program will start running after the reservations.

6.11.2. Modbus Program

This function is run by using Modbus touch screen equipment. The mechanism is to run the program in Modbus equipment by setting the program in advance, and input the program serial number through Modbus touch screen after switching to remote mode.

The setting steps of the Modbus program are as follows：

1.Enter the Settings-Modbus program；

2.The maximum number of programs can be set up to 300；

3.After setting up, switch to remote mode；

4.Fill in the serial number of the program to be run in the running of the program in the Modbus touch screen, click on the {Running}, and the program will start running.

7. Tool and User Coordinates

7. 1. Tool Calibration

7. 1. 1. Tool coordinate system

Flange Center: The origin of the default tool coordinate system, flange center point to flange positioning hole direction is + X direction, vertical flange outward direction is + Z direction, finally according to the right hand rule can determine the Y direction. The new tool coordinates are derived from the relative default tool coordinates.

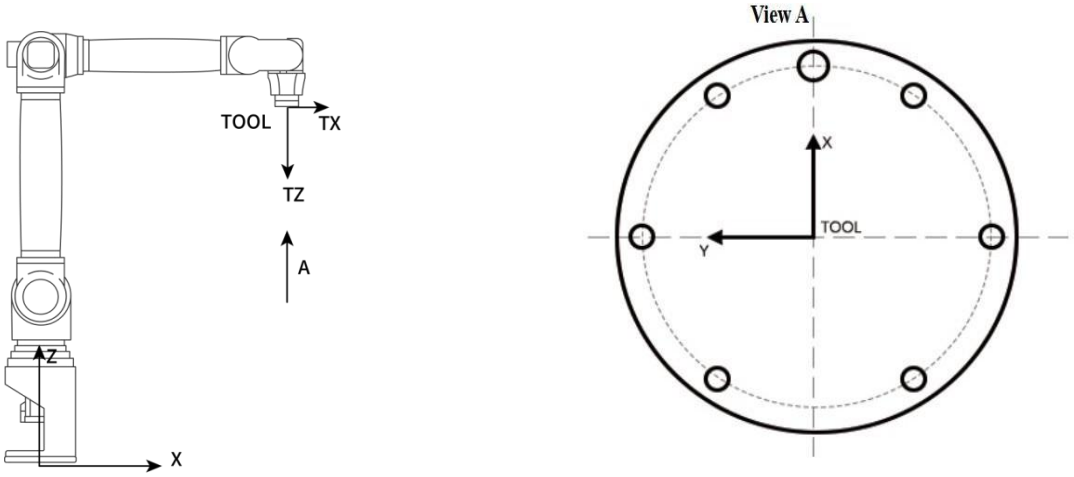


Figure 7.1 Tool coordinate system and flange

7. 1. 2. TCP: TOOL CENTER POINT

Robot trajectory and speed: refers to the trajectory and speed of TCP points

TCP is generally set in the center of the gripper, the end of the welding wire, the nose of the static arm of spot welding and so on.

In order to describe the position of an object in space, it is necessary to fix a coordinate system on the object, and then determine the position and attitude of the coordinate system (origin position and three coordinate axis attitude), namely, seven DOFs are needed to describe the position and attitude of the rigid body. For industrial robots, the end flange mounting tool is needed to operate. In order to determine the position and attitude of the tool, a TCS(tool coordinate system) is bound to the tool.

The origin of TCS is TCP (Tool Center Point). In trajectory programming of robots, the position and attitude of TCS in other coordinate systems should be recorded and executed in the program.

Industrial robots usually define a TCS in advance. The XY plane of TCS is bound to the flange plane of the sixth axis of the robot. The origin of TCS coincides with the center of flange.Obviously TCP is in the flange center. ABB robots call TCP as tool0 and REIS robots call it as _ tnull. Although the default TCP can be used directly, in practice, such as welding, the user usually defines the TCP point to the tip of the wire (actually the position and attitude of the welding welding torch tool coordinate system in the tool0 coordinate system). Then the position recorded in the program is the position of the welding wire tip, and the attitude recorded is the attitude of the torch revolving around the welding wire tip.

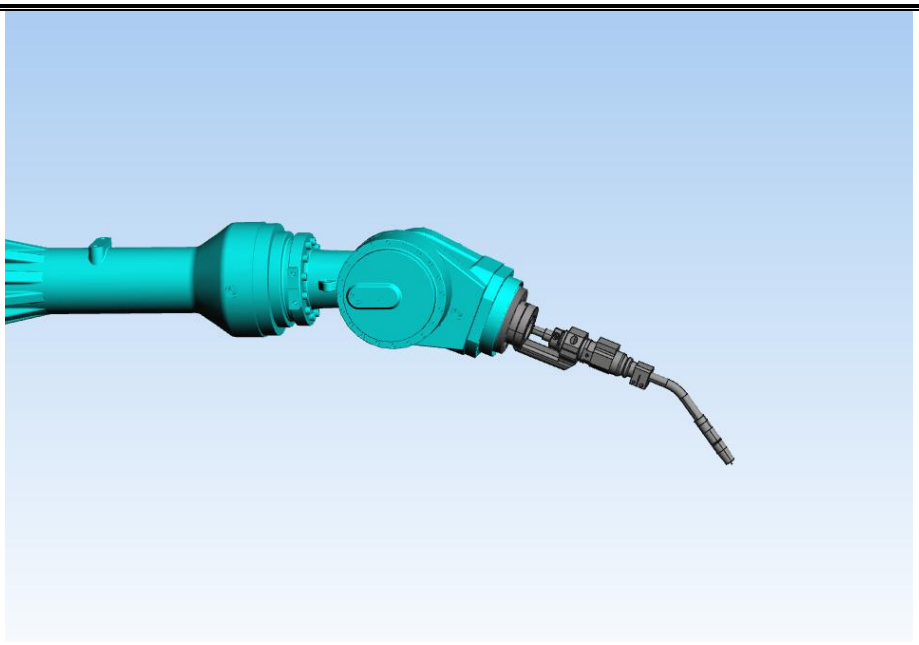


Figure 7.2 Industrial robot welding

Thinking

From thinking 1, we know that the tool coordinate system is a research object in motion, but what role does it play in the actual debugging process? Thinking about how the attitude and position of the grippers in Figure 1 and Figure 2 are adjusted?

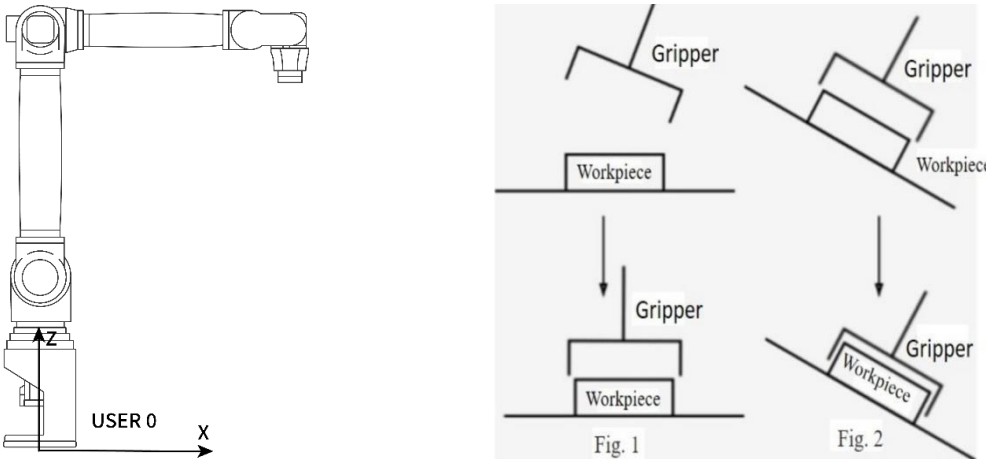


Figure 7.3 Claw posture and position

Conjecture: Two conjectures can be drawn from thinking:

Conjecture 1: If the gripper in Figure 1 has a rotating point, the gripper can be selected directly around the rotating point.

Conjecture 2: If one of the grippers in Figure 2 moves in the forward direction, it can move directly past.

Conclusion: The function of establishing tool coordinate system is as follows:

1.Establish the TCP point of the tool (i.e. tool center point) to facilitate the adjustment of tool status.

2.Determine the direction of tool feed to facilitate tool position adjustment.

7.2. Tool Coordinate System Characteristics

The new tool coordinate system is obtained by changing relative to the default tool coordinate system. The position and direction of the new tool coordinate system always maintain an absolute position and attitude relationship with the flange, but they are always changing in space.

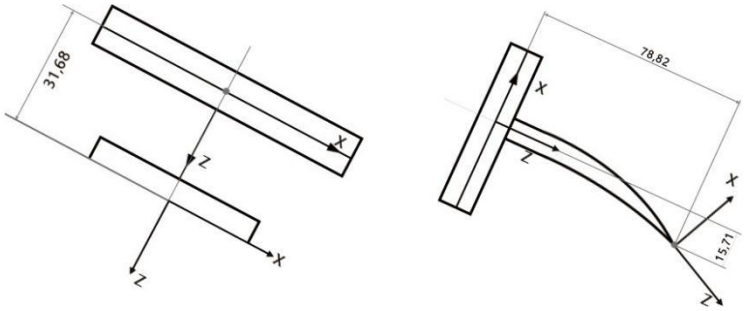
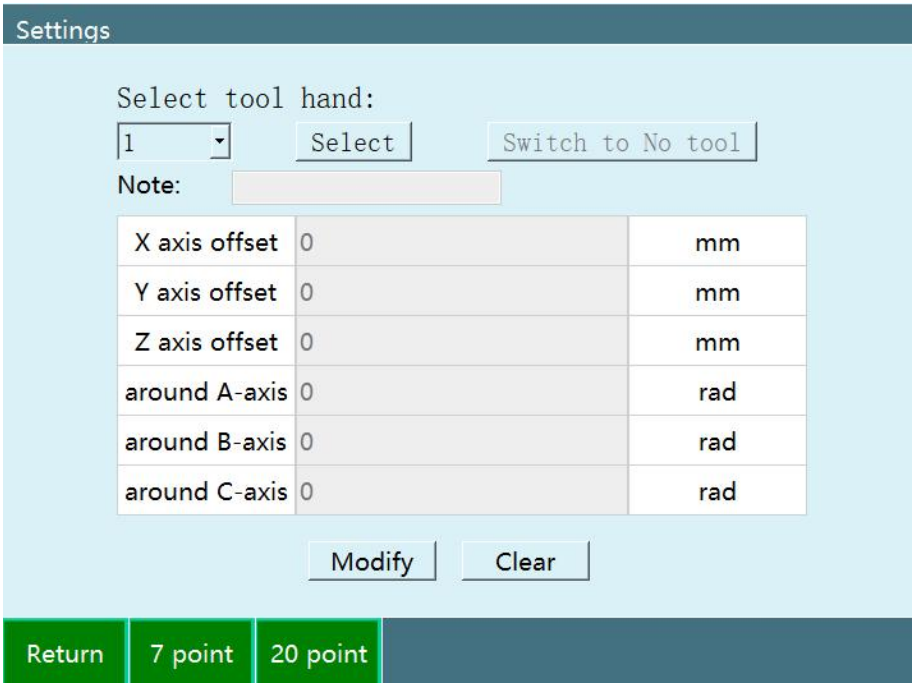


Figure 7.4 New tool coordinate system

7.3. Tool Parameter Setting

Click on the {Tool Calibration} in the Settings to enter the tool calibration interface, as shown in the figure.



If there are detailed parameters of the tool, in this interface, users can directly fill in the relevant parameters of the tool end offset without seven-point calibration.

When entering this interface, the size parameters of the tool saved in the controller will be read automatically (default items are 0). If you change the tool hand, please fill them in again.

Detailed parameter setting steps are as follows:

1.Open the tool calibration interface, the following table is an introduction to each parameter:

parameters	action

X-axis direction migration	The migration length (mm) of the tool end relative to the center of the flange along the Cartesian coordinate system X-axis
Y-axis direction migration	The migration length (mm) of the tool end relative to the center of the flange along the Cartesian coordinate system Y-axis
Z-axis direction migration	The migration length (mm) of the tool end relative to the center of the flange along the Cartesian coordinate system Z-axis
Migration about axis A	The migration angle (°) of the tool end relative to the center of the flange around the X-axis of the Cartesian coordinate system
Migration about axis B	The migration angle (°) of the tool end relative to the center of the flange around the Y-axis of the Cartesian coordinate system
Migration about axis C	The migration angle (°) of the tool end relative to the center of the flange around the Z-axis of the Cartesian coordinate system

2.Click on the {Modify} button.

3.Fill in the parameters corresponding to the tool, in which the functions of the parameters are shown in the table above.

4.After be sure that it is correct, click on the {Save} button and and the setting is successful.

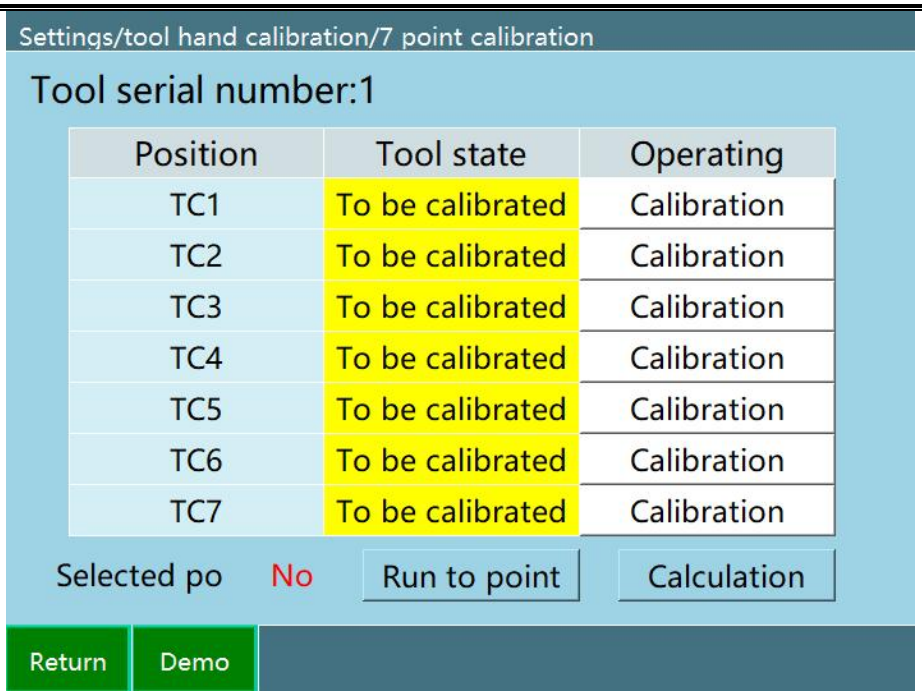

CAUTION

Click on the {Clear} button to clear the parameters that have been filled in.

If you click on {Back} button or {Seven-Point Calibration} button at the bottom operating area during parameter setting process, then jump to the corresponding interface, and unsaved settings will not be retained.

7. 4. Seven-Point Calibration

Click on the {Seven-Point Calibration} button at the bottom to enter the seven-point calibration interface, as shown in the figure.



Without the detailed parameters of the tool, TCP calibration can be carried out, and the size parameters of the tool can be calculated automatically. The specific calibration steps are as follows:

1.Now take the pen tip as the reference point and make sure that the reference point is fixed, as shown in the figure below.

2.Put the end of the tool perpendicular to the reference point and click on the corresponding{Calibration} button of the interface “TC1”, as shown in the figure below.

3.TC2 Calibration: Switch the robot to a position with the end facing the reference point and click on the corresponding {Calibration} button on the line, as shown in the figure below.

4.TC3 Calibration: Switch the robot to a position with the end facing the reference point and click on the corresponding {Calibration} button on the line, as shown in the figure below.

5.TC4 Calibration: Switch the robot to a position with the end facing the reference point and click on the corresponding {Calibration} button on the line, as shown in the figure below.

6.TC5 Calibration: Switch the robot to a position with the end facing the reference point and click on the corresponding {Calibration} button on the line, as shown in the figure below.

7.TC6 Calibration: On the basis of TC5, move any distance along the negative direction of Cartesian coordinate system X axis, and click on the corresponding {Calibration} button of the line, as shown in the figure below.

8.TC7 Calibration: On the basis of TC6, move any distance along the negative direction of Cartesian coordinate system Y axis, and click on the corresponding {Calibration} button of the line, as shown in the figure below.

9.Click on {Run To This Point} to see if the calibration is accurate.

10.Click on the {Calibration} button and the calibration is successful.

If you are not satisfied with a certain point in the calibration process, you can click on the corresponding {Cancel The Calibration} button of the line to cancel the calibration, and then calibrate the point again after

canceling the calibration.

Click on the {Demonstration} button at the bottom to open the demonstration interface and explain how to calibrate the tool.

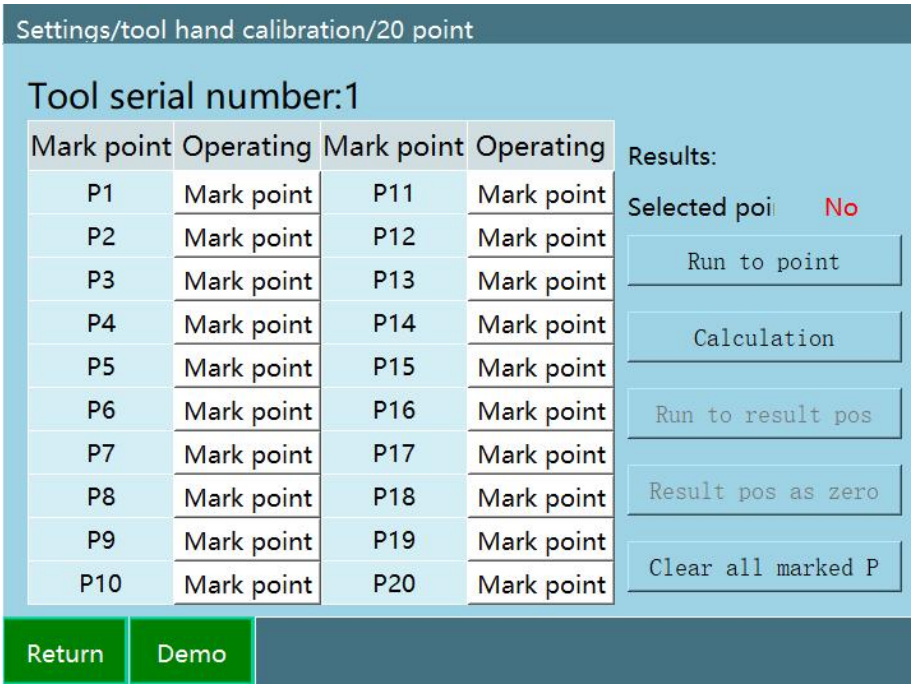
Click on the {Back} button at the bottom to return to the tool calibration interface.

7.5. Twelve/Fifteen-Point Calibration

Twelve/Fifteen/Twenty-Point Calibration share a calibration interface, and the first fifteen points of calibration are the fifteen-point calibration method.

Twelve-Point Calibration means that Fifteen-Point calibration does not mark the last three points (thirteen-fifteen),and the calibration result is only the offset of the XYZ axis direction of the tool hand, without the value of rotation around ABC.

Click on the {Twenty-Point Calibration} button at the bottom to enter the Twenty-point calibration interface, as shown in the figure.



The specific calibration steps are as follows:

1.Find a reference point (the tip of the calibration cone is the reference point) and make sure that this reference point is fixed.

2.Start inserting the position point, every time you insert a point, click [Mark this point] to insert 15 points

Specific steps are as follows:

1）The first point the robot returns to the zero point, and the robot tip is aligned with the tip of the calibration cone through Cartesian coordinates to calibrate the first point;

2）The second point on the basis of the first point, rotate C by 180 degrees through the Cartesian coordinate system; align the tip to calibrate the second point;

3）The third point the robot returns to the zero point, and aligns the tip of the robot with the tip of the

calibration cone through the Cartesian coordinate system; calibrate the third one;(Same as the first point)

4）The fourth point on the basis of the third point, using the rectangular coordinate system to make B-, the degree is at 30°-60°, and the tip is aligned to calibrate the fourth point;

5）The fifth point On the basis of the fourth point, make B+，J5>-90° through the rectangular coordinate system, and align the tip of the robot with the tip of the calibration cone to calibrate the fifth point.

6）The sixth point select the first point and move the robot to the first point. On the basis of the first point, use the Cartesian coordinate system to do B+, J5>-90°, and align the tip to calibrate the sixth point;

7） Seventh point on the basis of the first point, use the rectangular coordinate system to do B-, J5>-90°, and align the seventh point with the tip;

8） The eighth point On the basis of the seventh point, use the rectangular coordinate system to do A+, rotate 90°, J5>-90°, and align the tip to calibrate the eighth point;

9） The ninth point on the basis of the seventh point, use the rectangular coordinate system to do A- and rotate 90°, J5>-90°, and align the tip to calibrate the ninth point;

10 ） The tenth point the robot returns to the first point and moves the five axes through the joint coordinate system to make the five axes move up,J5<-90°, align the tip and calibrate the tenth point;

11） The eleventh point On the basis of the tenth point, use the rectangular coordinate system to do A+, rotate 90°, J5<-90°, and align the tip to calibrate the eleventh point;

12） The twelfth point on the basis of the eleventh point, use the rectangular coordinate system to do A-, rotate 90°, J5<-90°, and align the tip to calibrate the twelfth point;

13）The thirteenth point when the robot returns to the zero position, adjust the robot posture so that the tip of the end tool of the robot faces downwards, and align the

calibration tip with the calibration cone to calibrate the thirteenth point;

14）The fourteenth point ,on the basis of the thirteenth point, use the Cartesian coordinate system to do X-, the robot move a certain distance, and click directly to calibrate the fourteenth point.

15） The fifteenth point on the basis of the fourteenth point, use the rectangular coordinate system to do Y+ to make the robot move a certain distance, and click directly to calibrate the fifteenth point;

After marking, click {Calculate}.

{Cancel the calibration}:If you are not satisfied with a certain point in the calibration process, you can click on the corresponding {Cancel The Calibration} button of the line to cancel the calibration, and then calibrate the point again after canceling the calibration.

{Run to that point}:After calibrating a point, you can click [Run to this point], and the robot will run to that point.

{Mark the result position as zero}:Set the position after calibration compensation as the zero position of the current robot.

{Clear all calibration points}:The calibration point will be saved in the controller. The calibration result will be cleared only after clicking to cancel calibration, clear all calibration points, and switch tools to enter

the calibration interface manually.

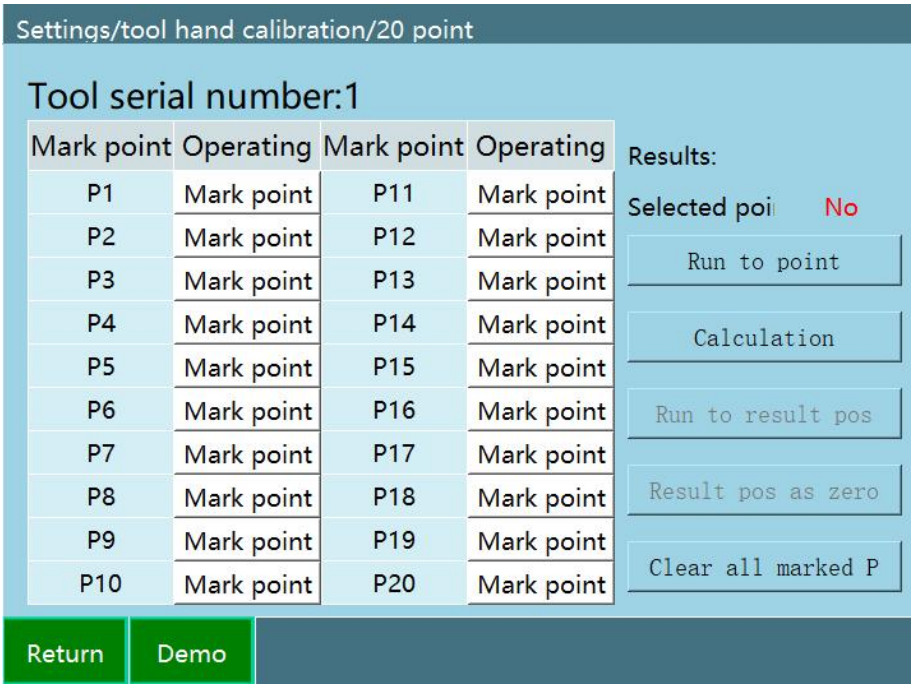
CAUTION
The attitude of each point should be in any direction as far as possible. If the attitude is rotated in a certain direction, sometimes the precision is inaccurate. Please keep the reference point fixed during calibration, otherwise the calibration error will increase.

Click on the {Back} button at the bottom to return to the "tool calibration" interface.

7.6. Twenty-Point Calibration

Twelve/Fifteen/Twenty-Point Calibration share a calibration interface, and calibrate all twenty points is to use the Twenty-Point Calibration method.

Click on the {Tool Calibration} button at the bottom of the interface to enter the "twenty point calibration" interface, as shown in the figure.



The specific calibration steps are as follows:

Find a reference point (the tip of the pen is the reference point) and make sure that the reference point is fixed.

Start inserting position points. For each insertion point, click on {Mark The Point}, insert twenty points, and the greater the attitude difference of each point, the better.

Manufacturer recommendations:

In the calibration step, the first point of tool hand posture is vertically downward, the second point moves the A+ axis, the third point moves A+, the fourth point moves A+, the fifth point moves A-, the sixth point moves A-, and the seventh point moves A-, move B+ at the eighth point, move B+ at the ninth point, move B+

at the tenth point, move B- at the eleventh point, move B- at the twelfth point，

move B- at the thirteenth point, and move the other points mainly C-axis forms a double across-shaped layout calibration

After completing the twenty-point mark, click on the {Calculate}.

{Cancel The Calibration}:If you are not satisfied with a certain point in the calibration process, you can click on the corresponding {Cancel The Calibration} button to cancel the calibration, and then calibrate the point again after canceling the calibration.

{Run To This Point}:{Run To This Point} can be clicked after each calibration point , and the robot will run to that point.

{Mark The Result Position As zero Point}: Set the position after calibration and compensation to the zero position of the current robot.

{Clear all calibration points}:The calibration point will be saved in the controller, and the calibration result will be cleared only after clicking to cancel calibration, clear all calibration points, and switch tools to enter the calibration interface manually.


CAUTION

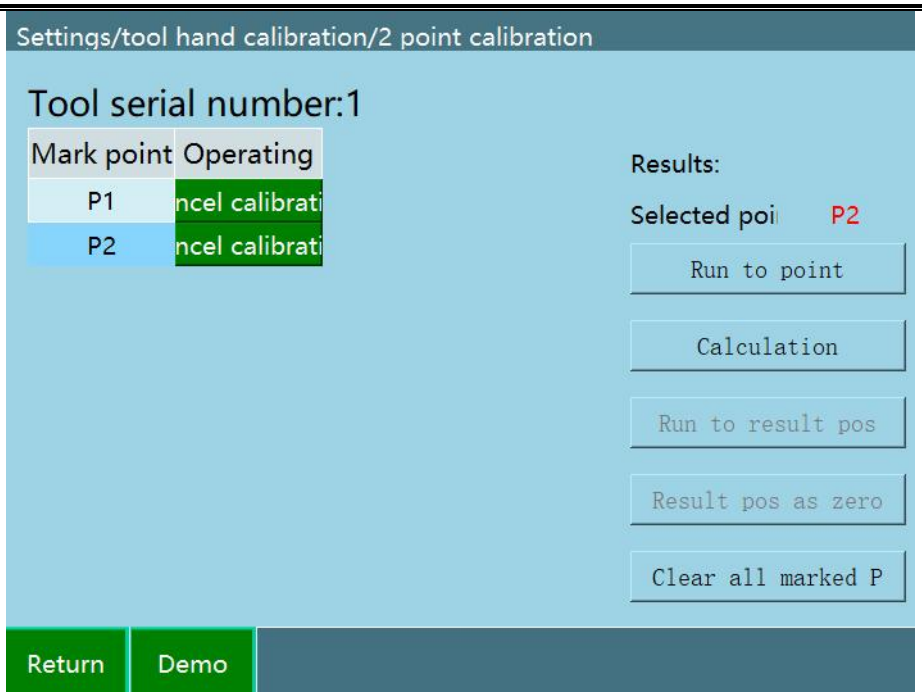
The attitude of each point should be in any direction as far as possible. If the attitude is rotated in a certain direction, sometimes the precision is inaccurate.

Please keep the reference point fixed during calibration, otherwise the calibration error will increase.

7.7. Two-Point Calibration

Two-point calibration supports 4-axis SCARA and 4-axis palletizing.

Click the [two point calibration] button at the bottom of the {tool hand calibration} interface to enter the "two- point calibration" interface, as shown in the figure.



The specific calibration steps are as follows:

1.Find a reference point (the pen tip is the reference point), and make sure that this reference point is fixed.

2.When you start to insert a point, click [Mark this point] for each point you insert, and insert two points. The larger the difference in posture of each point, the better.

3.After completing the two points mark, click [Calculate].

If you are not satisfied with a certain point in the calibration process, you can click on the corresponding {Cancel The Calibration} button to cancel the calibration, and then calibrate the point again after canceling the calibration.

{Run To This Point} can be clicked after each calibration point , and the robot will run to that point. Move the robot to another position, and then click on the {Run To The Position Of Calculation Result}, the robot moves to the original calibration position, which is equivalent to the zero position of the robot.

{Mark The Result Position As zero Point}: Set the position after calibration and compensation to the zero position of the current robot.

{Clear all calibration points}:The calibration point will be saved in the controller, and the calibration result will be cleared only after clicking to cancel calibration, clear all calibration points, and switch tools to enter the calibration interface manually.

CAUTION

The attitude of each point should be in any direction as far as possible. If the attitude is rotated in a certain direction, sometimes the precision is inaccurate.

Please keep the reference point fixed during calibration, otherwise the calibration error will increase.

Click on the {Demo} button at the bottom to open the "demo" interface and explain how to calibrate the tool.

Click on the {Back} button at the bottom to return to the "tool calibration" interface.

7.8. User Coordinates

7.8.1. The Function of User Coordinate System

Definition: Default user coordinate system: The default user coordinate system User 0 coincides with the rectangular coordinate system. The new user seating system is based on the default user coordinate system changes.

Thinking: From Think 1, we know that the user coordinate system is a reference object in motion, but what role does it play in the actual debugging process?

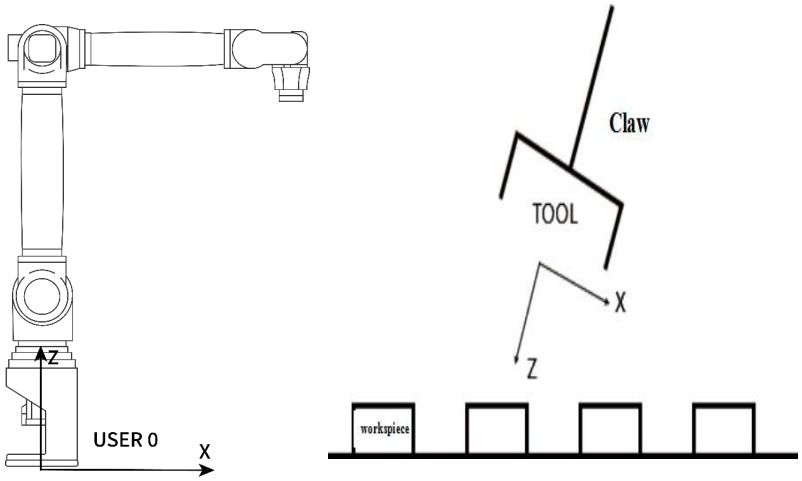


Figure 7.5 Non-tilting worktable

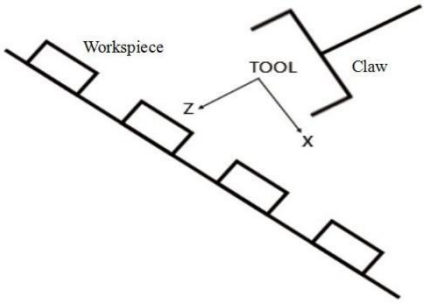


Figure 7.6. Inclined worktable

Conjecture: As can be seen from Figure 4, it will be difficult to debug the position of each workpiece using the default user coordinate system User 0 or Cartesian coordinate system, but it will be more convenient if two directions of a coordinate system are parallel to the worktable.

Conclusion: The function of user coordinate system

Determine the reference coordinate system;

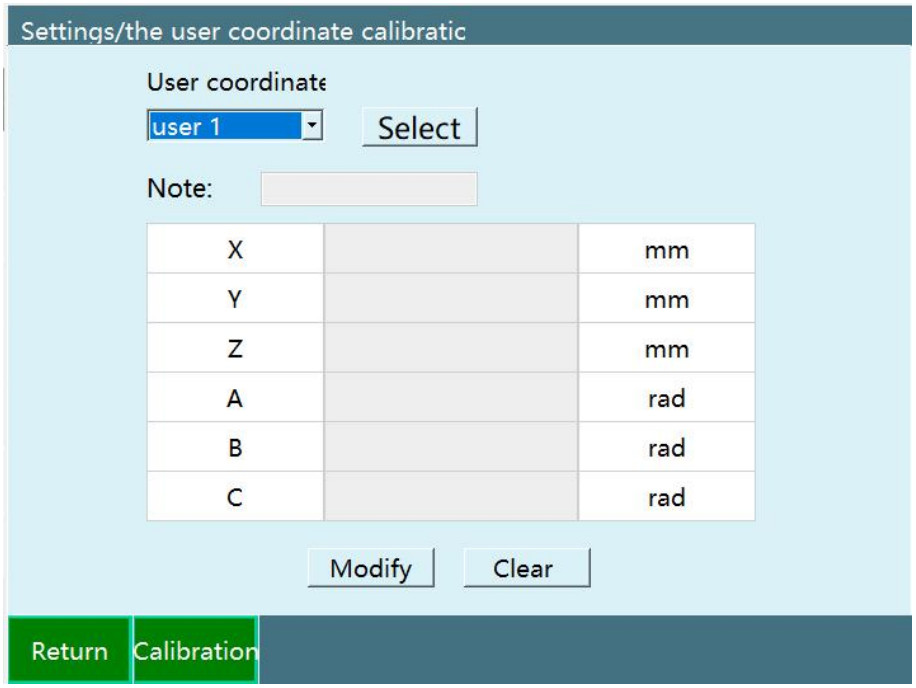
1.Determine the direction of motion on the worktable for easy debugging.

2.Characteristics of user coordinate system

The new user coordinate system is obtained by changing the default user coordinate system User 0. The position and attitude relative space of the new user coordinate system are unchanged.

7.8.2. User Coordinate Parameter setting

Click on the {User Coordinate Calibration} button in the "Settings" interface to enter the "user coordinate" interface, as shown in the figure.



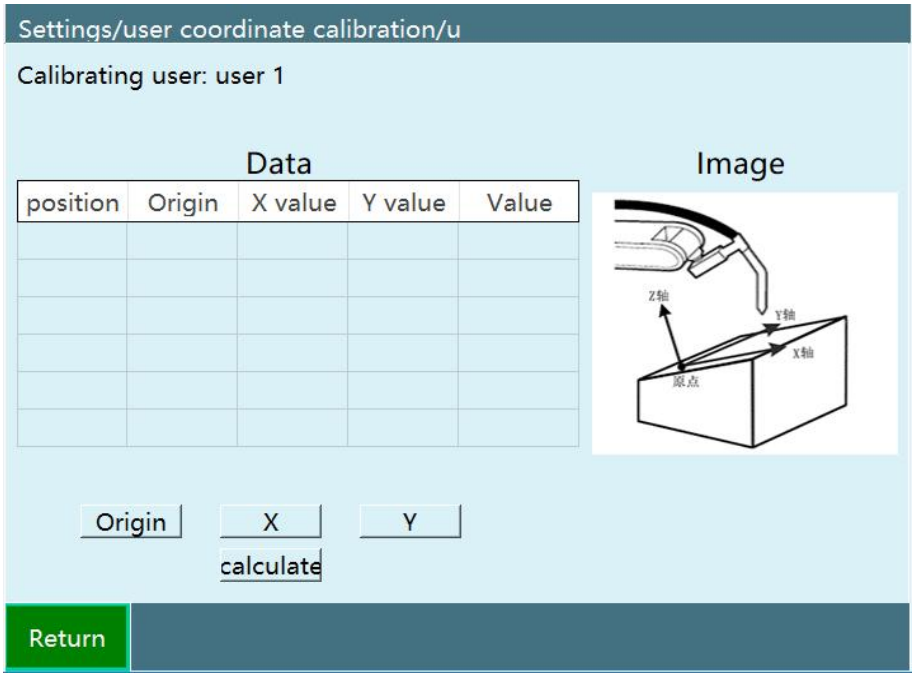
The parameters of the user coordinates are as follows

parameters	function
X value	Migration of user coordinate origin from the X-axis direction of robot base origin
Y value	Migration of user coordinate origin from the Y-axis direction of robot base origin
Z value	Migration of user coordinate origin from the Z-axis direction of robot base origin
A value	The rotation angle (radian) of the user coordinate system relative to the Cartesian coordinate system around the X-axis direction
B value	The rotation angle (radian) of the user coordinate system relative to the Cartesian coordinate system around the Y-axis direction
C value	The rotation angle (radian) of the user coordinate system relative to the Cartesian coordinate system around the Z-axis direction

If you have exact values, please fill in directly. Note that the three values of ABC are radian.

7.8.3. User Coordinate System Calibration

Click on the {User Calibration} button at the bottom of the "User Coordinate Calibration" interface to enter the "User Calibration" interface, as shown in the figure.



The calibration of user coordinate system should follow the following steps:

1.Move the tip of the robot to the position expected of the origin of user coordinate system, and click on the {Calibrate Origin} button.

2.Move the robot at any distance relative to the origin of the user coordinate system to the position expected to be the positive direction of the X-axis of the user coordinate system, and click on the

3.{Calibration X-axis} button.

Move the robot at any distance relative to the origin of user coordinate system to the position expected to be the positive direction of user coordinate system Y-axis, and click on the

{Calibration Y-axis} button.

CAUTION

If the Y axis of user coordinate system is not calibrated accurately, the system will automatically compensate.

Click on the {Back} button at the bottom of the interface to return to the user coordinate calibration interface.

8. Numerical Variable

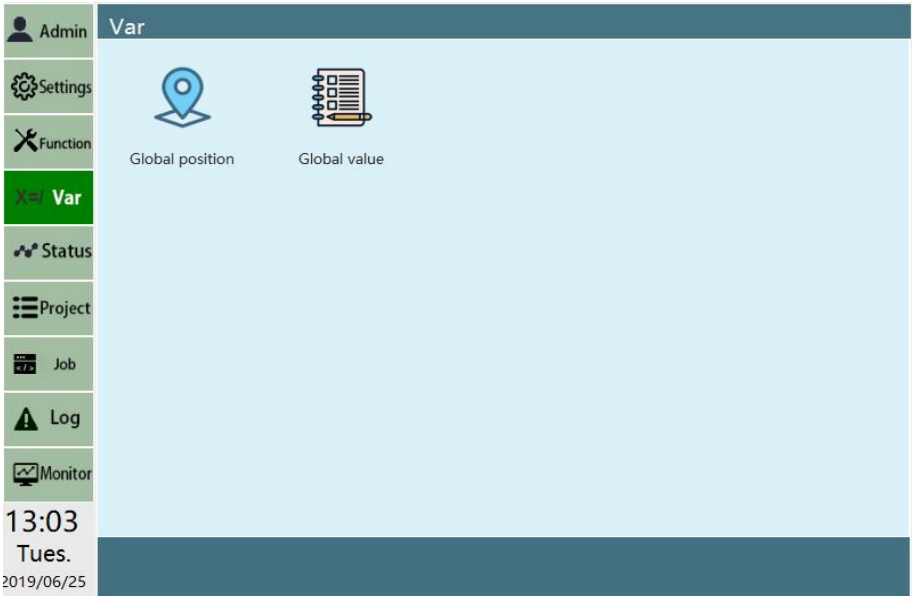
This chapter mainly describes the variables of the control system.

8.1. Variable Name

	type	Quantity	Example
Global numerical variables	Global Integer Variable、GINT	Each job file can contains 999	GI001
	Global Floating Point Variable 、GDOUBLE		GD001
	Global Bool Variable、GBOOL		GB001
Lobal numerical variables	Lobal Integer Variable、INT	Each job file can contain 999	I001
	Lobal Floating Point Variable 、DOUBLE		D001
	Lobal Bool Variable、BOOL		B001

8.2. Global Numerical Variables

The global value variable is a variable that can act on all robots and all programs, such as program AA of Robot 1 and program BB of Robot 2, which can use the same global value variable at the same time. This section will mainly explain the use of the global variable interface, as well as the use of position and numerical variables.



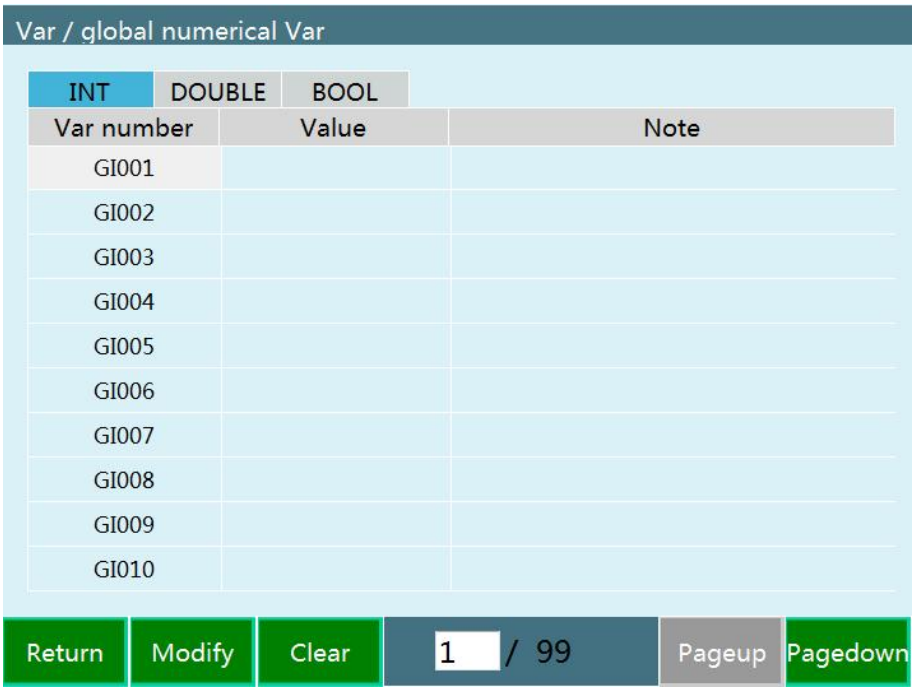
8.2.1. Global Value

Imagine how tedious it is for a robot to complete a process with so many instructions. If we insert instructions and set variables every time, we add value variables to call them. For example, "WHILE (INT001 = 10)... END (WHILE)" instructions, there are many procedures in which a robot completes a certain process, we directly call your pre-set value variables.

At the same time, global value variables can be used to transfer information between the main program,

the called subroutine and the background program for logical judgment.

Value variables store values, including integer variables , double variables and boolean variables.



8.2.2. Global Bool Variable

Global Bool variables store bytes. In this interface, the values and annotations of each variable can be modified. The significance of each parameter is as follows：

The name of the variable is the number of the variable, and the name of the global Bool variable is GAxxx.

The value is the value of the variable, and the range of values of Bool variables is "0/1".

Annotations are user-defined annotations for the variable, which facilitate users to mark the role of the variable, ranging from arbitrary values, and can be in Chinese.

8.2.3. Global Integer Variable

Global integer variables store integer. In this interface, the values and annotations of each variable can be modified. The significance of each parameter is as follows：

The name of the variable is the number of the variable, and the name of the global integer variable is GIxxx.

The value is the value of the variable, and the range of an integer variable is an integer.

Annotations are user-defined annotations for the variable, which facilitate users to mark the role of the variable, ranging from arbitrary values, and can be in Chinese.

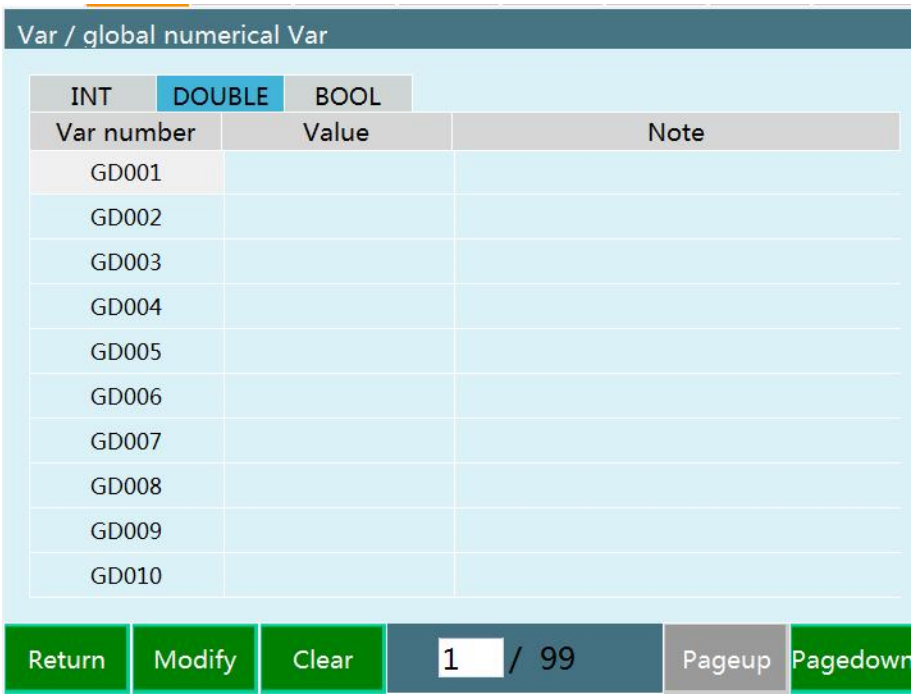
8.2.4. Global Floating Point Variable

Global real variables store real numbers. In this interface, the values, contents and annotations of each variable can be modified. The significance of each parameter is as follows:

The name of the variable is the number of the variable, and the name of the global real variable is GDxxx.

The value is the value of the variable, and the range of the floating-point variable is real.

Annotations are user-defined annotations for the variable, which facilitate users to mark the role of the variable, ranging from arbitrary values, and can be in Chinese.



Click on the data type you want to modify, select the variable name, and click on {Modify}, then you can modify the values and comments. Then click on {Save}. Click on {Clear} to clear the data you choose.

8. 3. Use of Global Numerical Variable

8. 3. 1. Define Global Value Variable

Define variables before using them. The methods for defining variables are as follows:

1.Click on the {Variable} button on the left to enter the variable interface;

2.Click on the global value variable;

3.Select the corresponding variable number and click on the {Modify} button;

4.Fill in the required values at the values and notes;

5.Variables that are not manually defined are defaulted to zero.

8. 3. 2. Assign Values to Global Variable by Calculating Instructions

Global variables can be calculated by ADD, SUB, MUL, DIV and MOD instructions. Note: Global Bool variables cannot be calculated !

8. 3. 3. ADD

Add operation (+).

Formula: variable type (variable name) = variable type (variable name) + variable value source (parameter)

To calculate global integer or global value variables, select GINT or GDOUBLE at the variable type. If the source of variable value is customized, the parameters can be manually filled in at the "new parameters". It can also be used for other variable values.

Case 1:

Premise: GI001=1 Instruction: ADDGI0011

Significance:GI001=GI001 Result: GI001=2

Case 2:

Premise: GI001=1 GI002=2

Instruction: ADD GI001 GI002

Significance: GI001=GI001+GI002 Result: GI001=3

8.3.4. SUB

Subtraction operation (-)

Formula: variable type (variable name) = variable type (variable name)- variable value source (parameter)

To calculate global integer or global value variables, select GINT or GDOUBLE at the variable type.

If the source of variable value is customized, the parameters can be manually filled in at the "new parameters". It can also be used for other variable values.

Case :

Premise:GD001=3.4 Instruction: SUB GD001 1.1 Significance: GI001=GI001-1.1

Result: GD001=2.3

8.3.5. MUL

Multiply operation(*)

Formula: variable type (variable name) = variable type (variable name) * variable value source (parameter)

To calculate global integer or global value variables, select GINT or GDOUBLE at the variable type. If the source of variable value is customized, the parameters can be manually filled in at the "new parameters". It can also be used for other variable values.

Case :

Premise: GD001=3.4 GI001=2

Instruction: MUL GD001 GI001

Significance: GD001=GD001*GI001 Result: GD001=6.8

8.3.6. DIV

Division operation (DIV)

Formula: variable type (variable name) = variable type (variable name) DIV variable value source (parameter)

To calculate global integer or global value variables, select GINT or GDOUBLE at the variable type. If the source of variable value is customized, the parameters can be manually filled in at the "new parameters". It can also be used for other variable values.

Case :

Premise: GD001=3.4 GI001=2

Instruction: DIV GD001 GI001 Significance: GD001=GD001÷GI001 Result: GD001=1.7

8.3.7. MOD

Remainder operation（MOD）

Formula: variable type (variable name) = variable type (variable name) MOD variable value source (parameter)

To calculate global integer or global value variables, select GINT or GDOUBLE at the variable type. If the source of variable value is customized, the parameters can be manually filled in at the "new parameters". It can also be used for other variable values.

Case :

Premise: GD001=14 GI001=3

Instruction: MOD GD001 GI001 Significance: GD001=GD001 MOD GI001 Result: GD001=2

8.3.8. Assign Values Directly to Global Variables

Through SETBOOL, SETINT and SETDOUBLE instructions, the value of variables can be changed directly when the program is running.

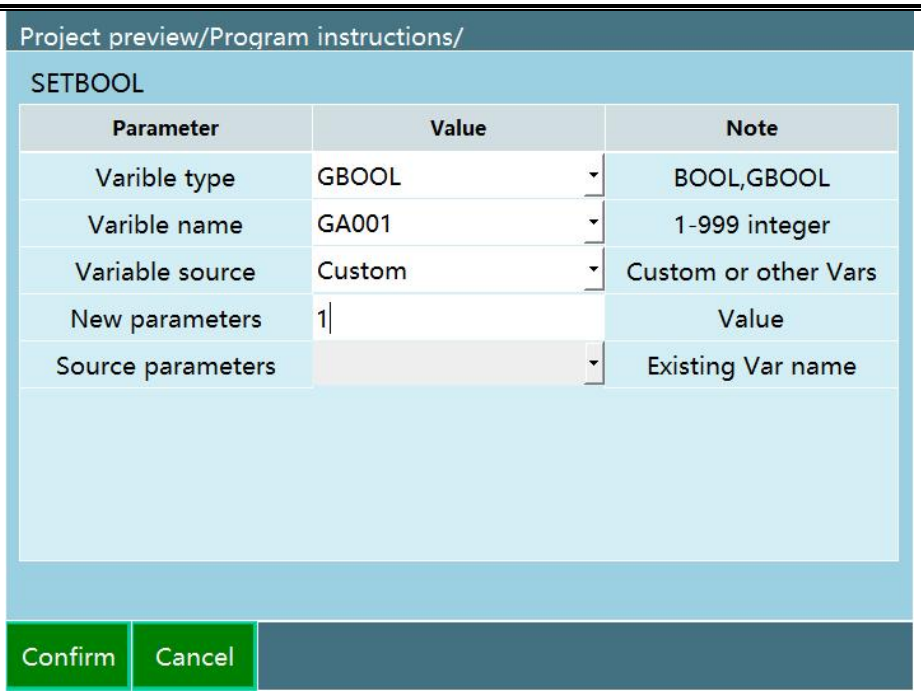
1.In the program, click on the {Insert} button；

2.Select "variable class "；

3.To change the global BOOL variable, select the SETBOOL instruction and click on {OK}；

4.Select "GBOOL" at the type of variable; select the previously defined global BOOL variable; variable value source selected "customized". Fill in the value that needs to be changed at the new parameter, and if you need to change the value of the variable to 1, fill in 1 here.

For example, if you need to change the value of GA001 variable to 1 when running the program, fill in the parameters as shown in the figure below.



SETINT and SETDOUBLE are used to set INT and DOUBLE type variables respectively, the usage is the same as above.

8.3.9. Use Global Variables to Count

In the process of running the program, all calculation and assignment operations are to change the values in the cache, and the values in the "variable-global value" interface will not be modified, that is, when the program stops running, the values of all global variables will be restored.

To count a loop process (such as a WHILE inner loop), you can use the FORCESET instruction. Using scene: There is a process between a WHILE and ENDWHILE instruction. There is an ADD GI001 1 instruction in the process, that is, every time a loop is made between WHILE and ENDWHILE, the value of GI001 variable is increased by one, that is, the number of execution times of the process is increased by one. After the program stops running, the value of GI001 is reduced to 0, so the number of operation times of the process can not be seen.

Solution: Insert a FORCESET GI001 instruction after the Add GI001 1 instruction. When the program is finished, the value of GI001 can be seen in the "variable-global value" interface, which represents the number of times the program runs.

Click on the {Insert} button in the "program" interface;

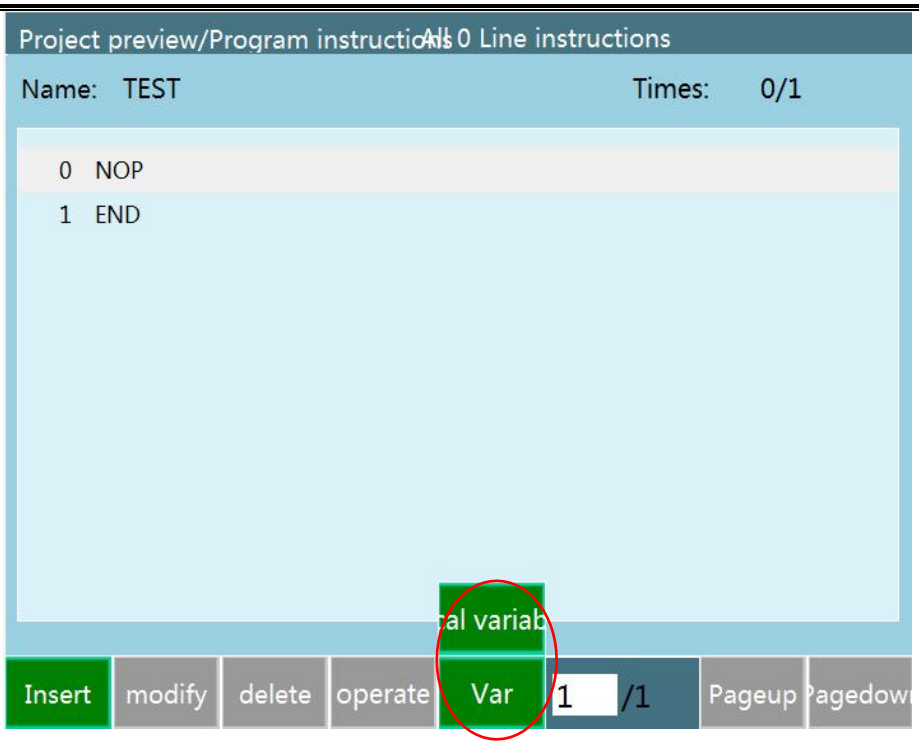
Select "variable class "-"FORCESET", and click on {OK};

Select the variable type. If you want to change the global integer variable, select GINT and select "GI001" for the variable name;

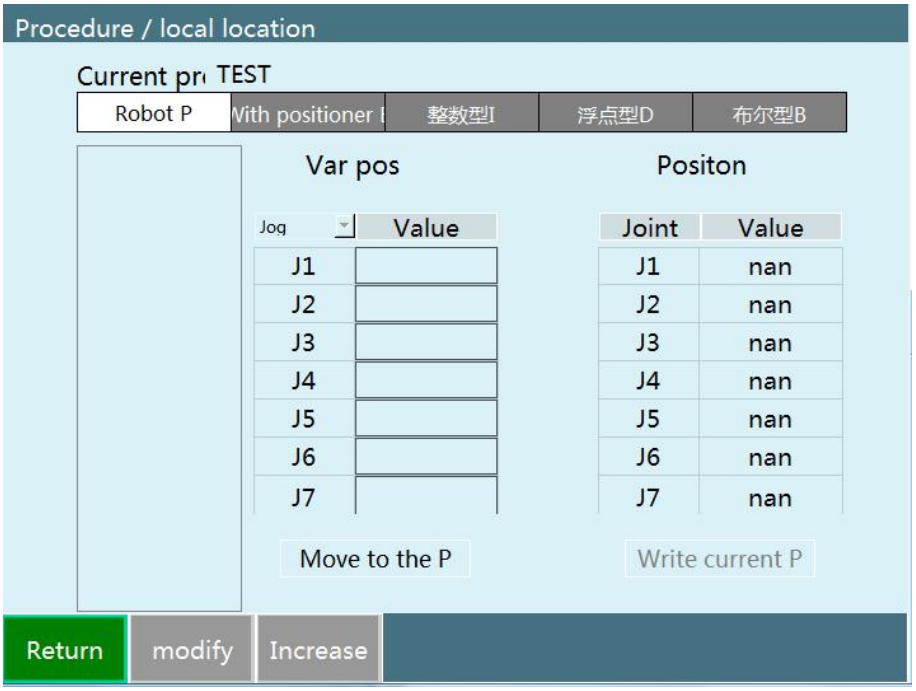
Click on the {Insert} button to complete.

8.4. Local numerical variables

Local variables can only be used for the defined program itself, such as variables of program A can not be used in program B.



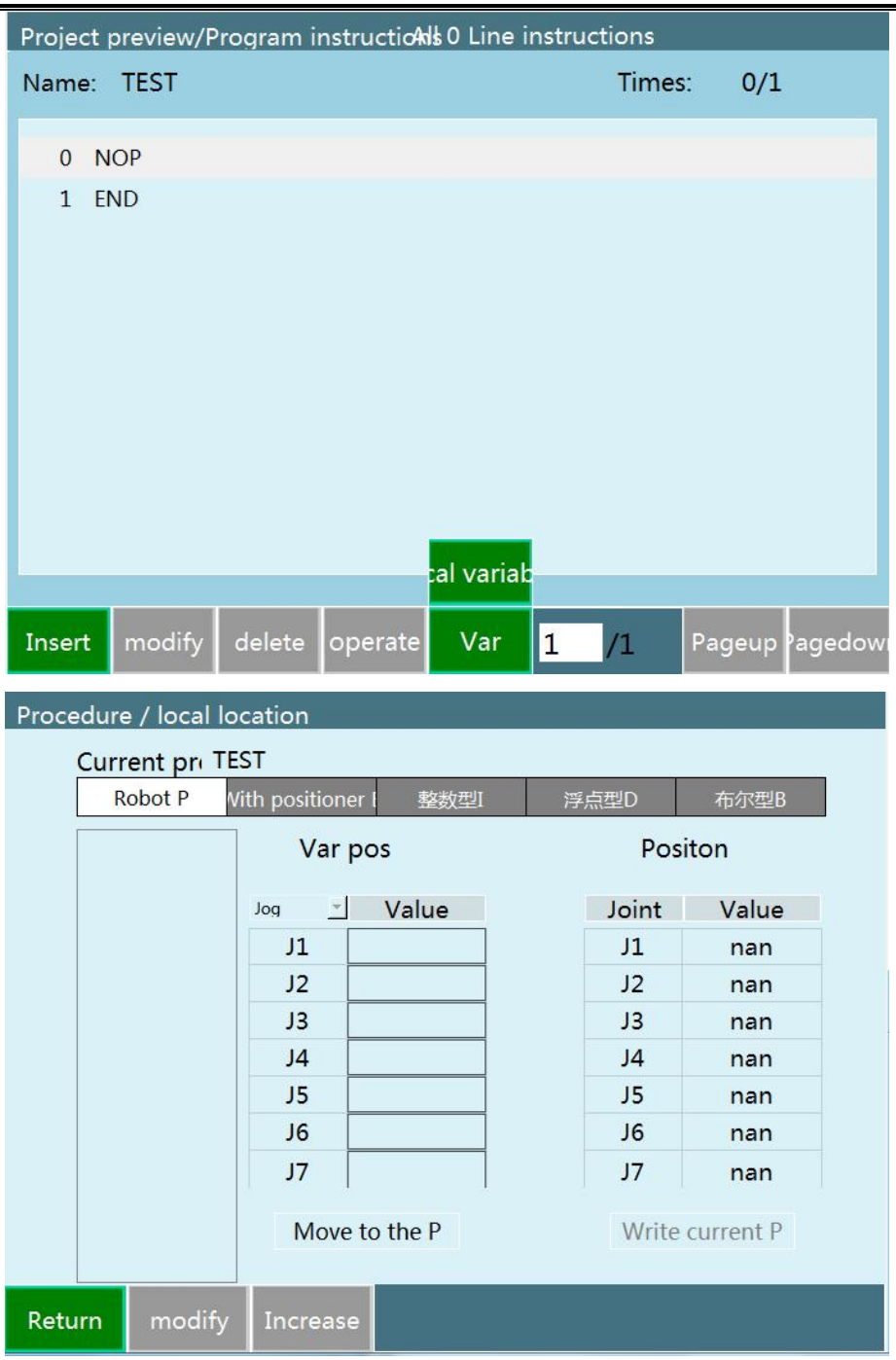
Numerical variables store values, including integer variables, real number variables, and Boolean variables.All defined local numerical variables can only be used in the current program, and other programs and background programs cannot be used.



8. 5. Use of local variables

Define Local Variables

Defining local variables is different from defining global variables. To define a local variable, you need to click the variable-local variable page setting on the program page.



8.5.1. Int I

Local integer variables are used to store integer variables. The variable name is Ixxx.

The default value is 0. When you need to modify, select the variable name to be modified, enter the value, and click Save.

8.5.2. Floating Point Variable D

Local real variables are used to store real variables. The variable name is Dxxx.

The default value is 0. When you need to modify, select the variable name to be modified, enter the value, and click Save.

8.5.3. Bool variable B

Local Bool variables are used to store Bool variables. The variable name is Bxxx.

The default value is 0. When you need to modify, select the variable name to be modified, enter the value, and click Save.

8.5.4. Assignment of Local Variables Using Calculation Instructions

The method of calculating and assigning local variables using the ADD, SUB, MUL, DIV, and MOD instructions is the same as the calculation method for global variables.

8.5.5. Assign Values Directly to Variables

The method of directly assigning a local variable using the SETINT, SETDOUBLE, and SETBOOL instructions is the same as the method of directly assigning a global variable.

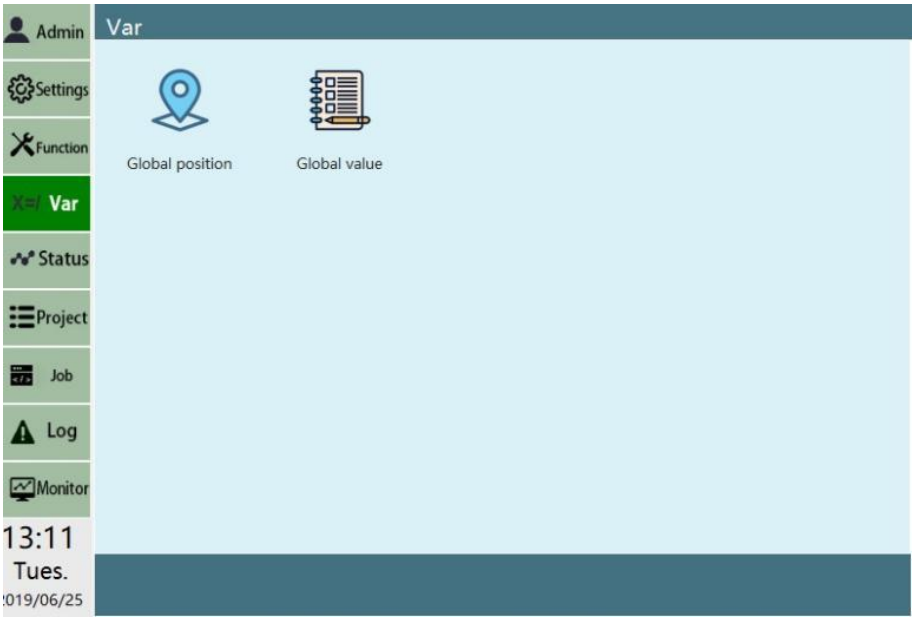
9. Position variables

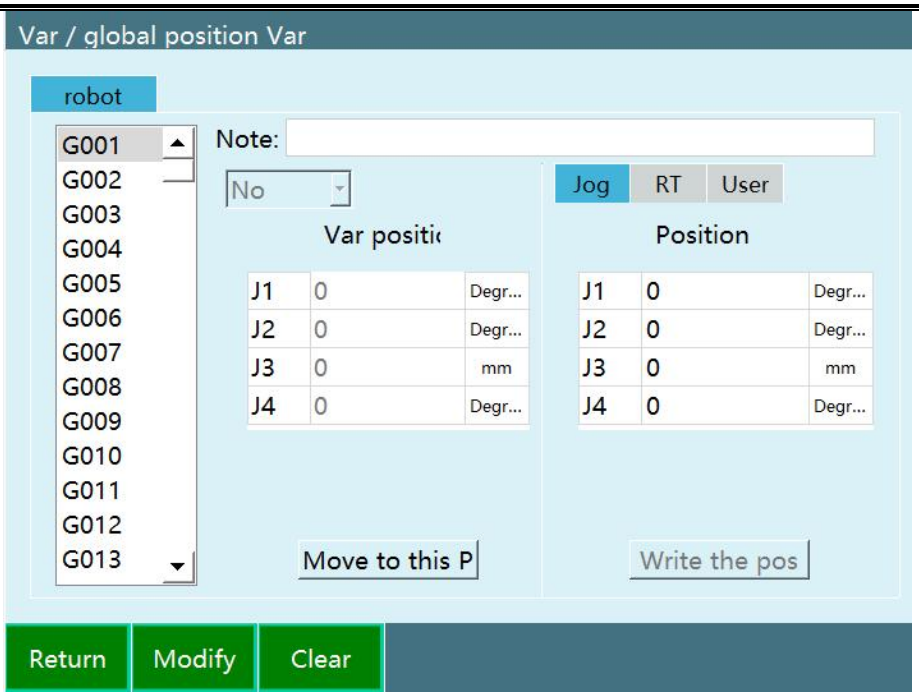
This chapter mainly describes the variable settings of the control system.

The global position variable	Global location、G	G001
The local location variable	Local locationP point	P001
	Local locationEpoint	E001
	Local locationS point (IMOV)	S001
	Local locationR point(SAMOV)	R001

9.1. Global position variable

The global position variable (G) is available in all job files for a robot. Defining global position variables needs to be done on the “variables - global position” interface.





The definition method of global position variable is as follows：

Enter the "variable"-"global numerical" interface;

Select variables that need to be defined, such as G001;

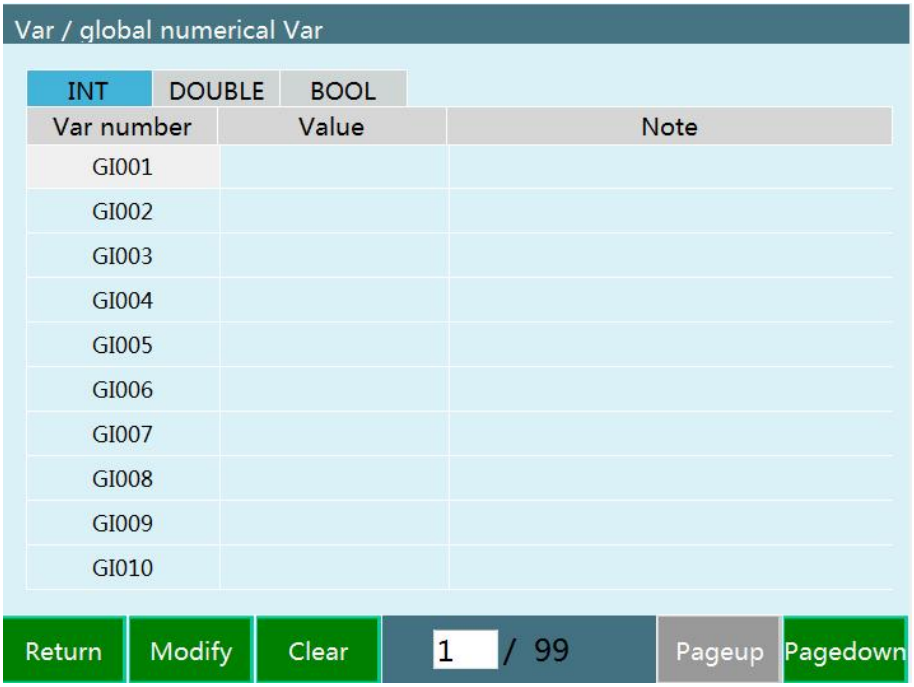
Teach the robot to the position that needs to be defined, and switch the coordinate system to the required coordinate system, such as rectangular coordinate system;

Click on the {Modify} button;

Click on the {Record Current Point} button;

Click on the {Save} button.

The global numerical variables (G) is available in all job files for a robot. Defining global position variables needs to be done on the "variables - global numerical" interface.



1.Enter the "variable"-"global position" interface;

2.Select variables that need to be defined, such as Integer Variable;

3.Then select the defined variable name. Such as GI001

4.Click on the {Modify} button

5.Enter the value after the selected variable name and the necessary remarks

6.Click on the {Save} button.

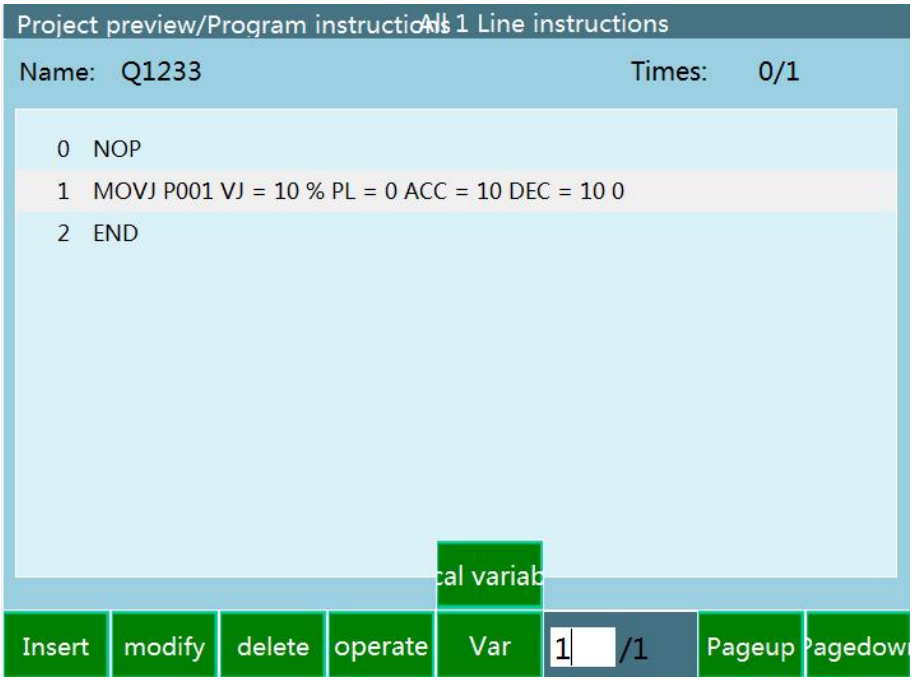
9. 2. Local position variables

The local location variable (P) can only be used for a single job file and cannot be used across all job files.

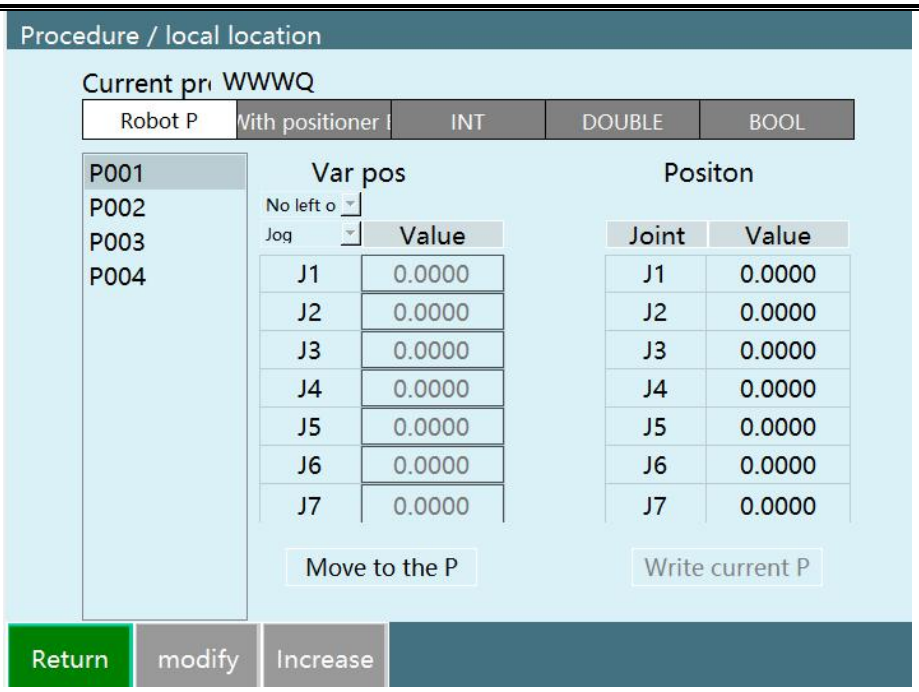
The definition of the local position variable is automatically defined only when the "new" variable is selected when a motion class instruction such as MOVJ, MOVL, MOVC is inserted.

View local location variables

1.Click Program-Variables-Local Variables to enter the local variable viewing interface



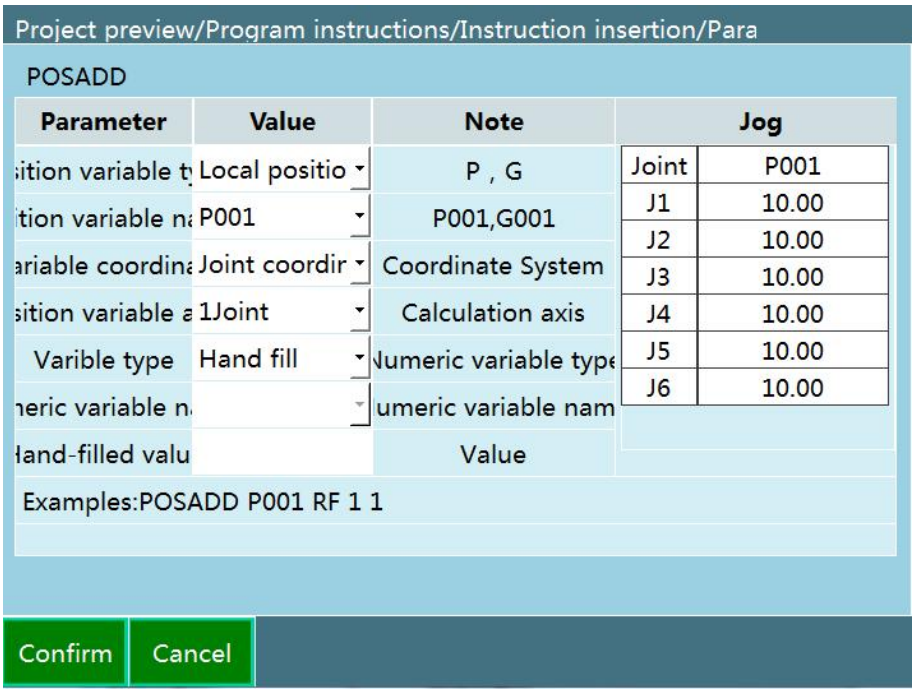
2.It can perform functions on local position variables, modify points, add points, run to that point, write current position, etc.



9.3. Use of Position Variable Calculation Class Instructions

9.3.1. POSADD Instruction

The position variable addition operation (+), which can add the value of the single axis of the position variable (global, local), and then assign it to the axis.



The variable name of the position variable can be a value variable, such as 1001=50, then P$1001 is P1001。

This instruction can add a single axis of a position variable in any coordinate system, regardless of the coordinate system in which the position variable is inserted, but it will be converted to the original coordinate system in assignment. For example, if the second axis of P001 variable is added, the P001 coordinate is in the joint coordinate system (0,0,0,0,0,0). You need to add 10 to the Z axis for this point.

Convert P001 to Cartesian coordinates (500,0,1000,0,0,0), then add 10 to the Z axis, i.e (500,0,1010,0,0,0), and finally convert to joint coordinates (0,-1,1,0,1,0) and assign this value to P001.

Formula: position variable = position variable {Coordinate System (Axis)} + value variable or number

To calculate a global integer or global value variable, select GINT or GDOUBLE at the variable type.

If the source of variable value selects to fill in manually, the parameters can be filled in manually at the "new parameters". It can also be used for other variable values.

9.3.2. POSSUB Instruction

The position variable subtraction operation (-), which can subtract the value of the single axis of the position variable (global, local), and then assign it to the axis.

The meaning and method of this instruction are similar to the POSADD instruction.

Formula: position variable = position variable {Coordinate System (Axis)} - value variable or number.

To calculate a global integer or global numeric variable, select GINT or GDOUBLE at the variable type.

If the source of variable value selects to fill in manually, the parameters can be filled in manually at the "new parameters". It can also be used for other variable values.

9.3.3. POSSET Instruction

Position variable assignment, which can assign the value of position variable (global, local) on a single axis directly.

The meaning and method of this instruction are similar to the POSADD instruction. Formula: position variable {Coordinate System (Axis)} = value variable or number.

To calculate a global integer or global numeric variable, select GINT or GDOUBLE at the variable type.

If the source of variable value selects to fill in manually, the parameters can be filled in manually at the "new parameters". It can also be used for other variable values.

9.3.4. READPOS Instruction

Read the position variable coordinate instruction, which can read the coordinate value of the position variable into the numerical variable.

When the coordinate value of "current position" is selected to read, the coordinate value is read when the robot runs to that position.

Formula: value variables (I, D.GI, GD) = position variables {Coordinate System (Axis)}

9.3.5. USERFRAME_SET Instruction

Modify the user coordinate system instruction, which allows the user to modify the value of an axis of the user coordinate system parameters. After modification, all points using user coordinates are migration.

For example, P001, P002 and P003 all use user coordinate system 1 and insert USERFRAME_SET instruction to add 10 to the X parameter of user coordinate system 1, then the position variables of P001, P002 and P003 are migration by 10mm to the X axis.

9.3.6. TOOLFRAME_SET Instruction

Modify the tool coordinate command.This command can modify the value of one axis of the tool coordinate system.After modification, the trajectory in the program used will change with the modification of

the tool coordinate system value.For example, the original tool offset is (0,0,200,0,0,0).Use this command to modify the offset of the Z-axis direction to 100, and the center position of the 6-axis flange will be offset down by 100mm during operation.If it is changed to the corresponding tool hand, the tool hand with the Z-axis offset of 200mm is changed to the tool hand with the Z- axis offset of 100mm, and the tip position remains unchanged.

9.3.7. COPYPOS Instruction

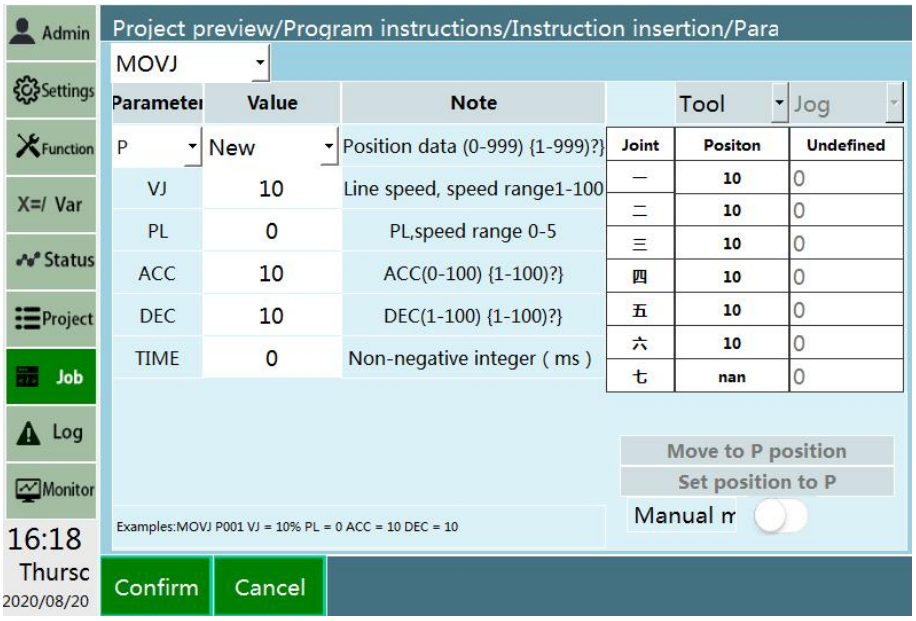
Copy point instruction.Copy the current position, local position variable, global position variable, etc. to another local or global position variable.

For example, copy the current position to the local position variable.Source location variable type: current location, source location variable name: not selected,Target position variable type: local position variable, target position variable name: P001.

9.3.8. 4-axis SCARA robot left and right hand

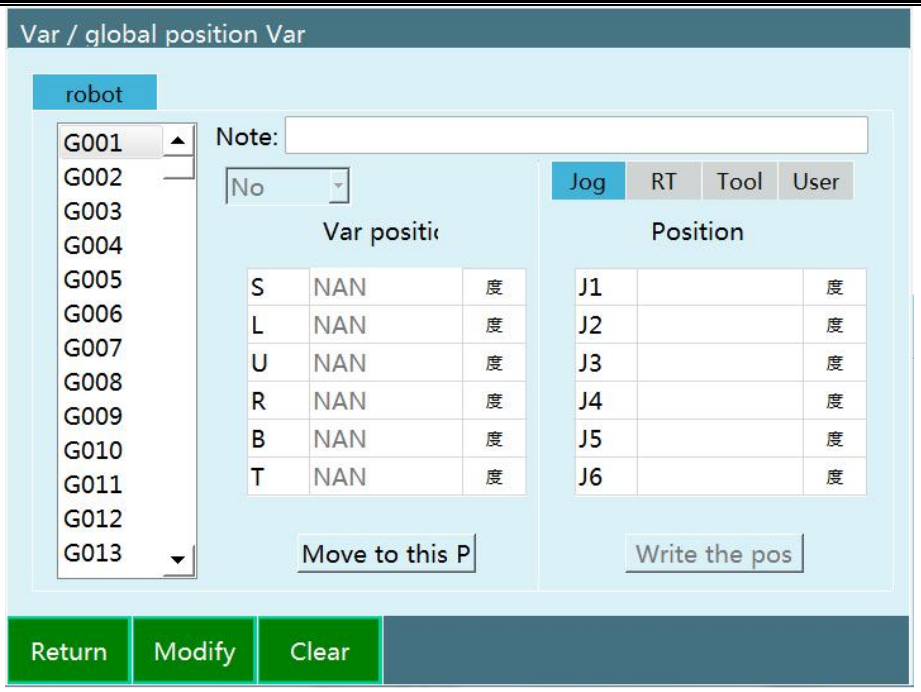
Using left and right hands is generally used to compress the robot's moving space or avoid obstacles.Generally, we only choose the rectangular coordinate system to set the left and right hands, and the judgment method is based on the direction of the two axes.The left and right hand function can only be used for 4-axis robots.

The command setting interface can choose left and right hands,When the setting is completed, you need to click the [Manual Modification] button, and then click OK to complete.



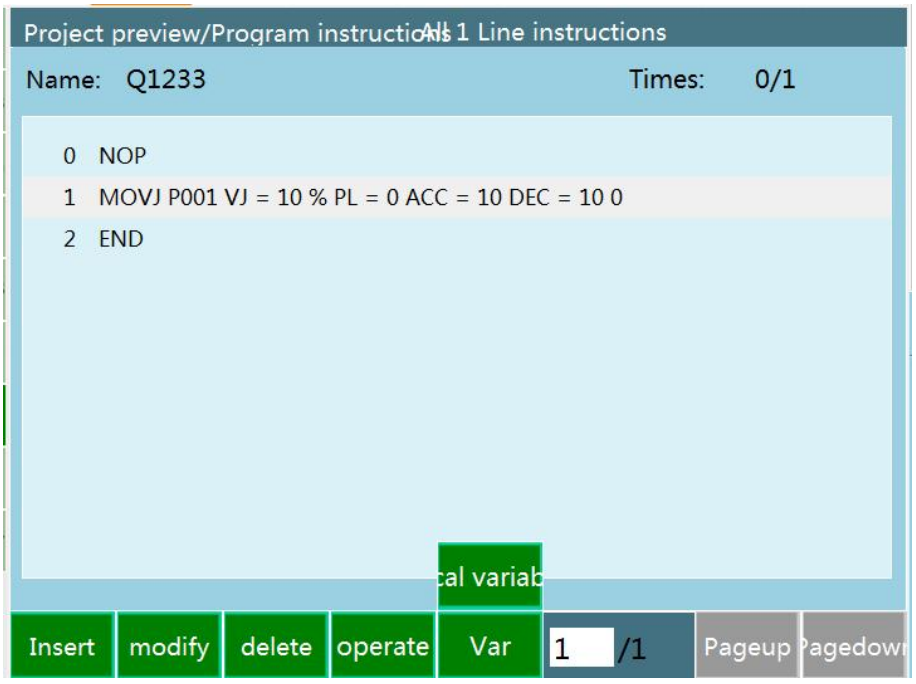
9.3.9. Global variable settings for left and right hands

Click [Variables]-[Global Variables],Click the drop-down menu.

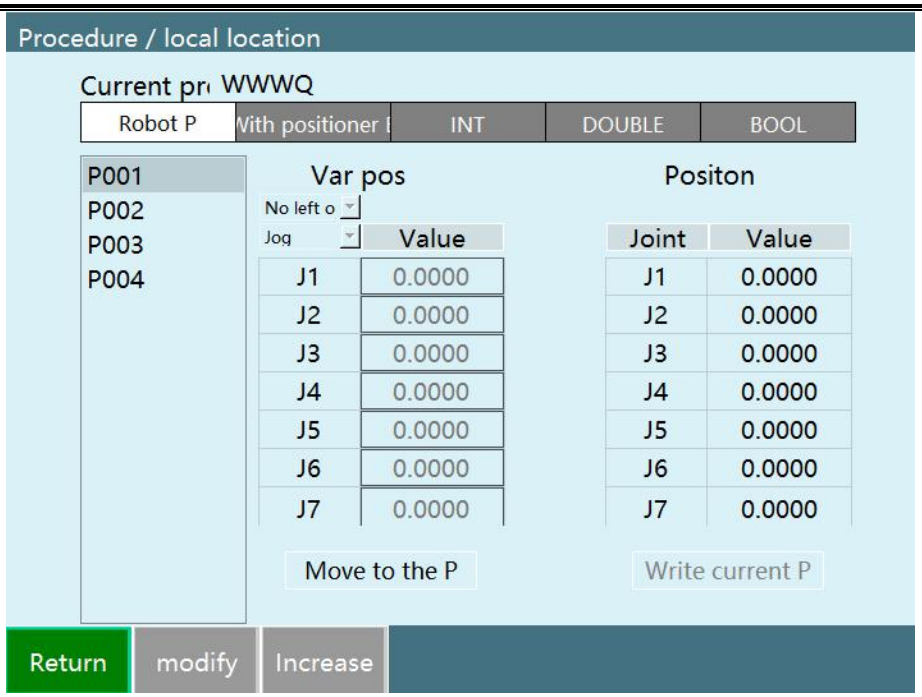


Local variable setting left and right hand

Click [Program], select a program to open, select [Variables]-[Local Variables] at the bottom



Click on the drop-down arrow at the top and select left and right hands



In the command parameter setting interface, you can select parameters to set local variables

10. Use of Conditional Judgment Class Instructions

Conditional judgment class instruction includes CALL, IF, WHILE, WAIT, JUMP and other instructions.

10.1. Instruction Description

10.2. CALL

CALL instruction is used to call a subroutine.

In this system, there is no distinction between the main program and the subroutine when establishing the program. When one program calls another program, the called program is the subroutine.

The two programs cannot call each other, that is, after program A calls program B, program B cannot call program A.

Parameter	Meaning
Program name	The program name of the called program

Case

Premise: Two programs, Job1 and Job2, have been established, and CALL instructions have been inserted into Job1.

Instruction: CALL [Job2] Meaning: Call subroutine Job2

Process: When the instruction of Job1 runs to the CALL instruction, the program jumps to the program Job2. After running all the instructions of the program Job2, the program jumps back to the next line of the CALL [Job2] instruction of the program Job1 and continues to run.

10.3. IF

If the condition of IF instruction is satisfied, the instruction between IF and ENDIF is executed. If the condition of IF instruction is not satisfied, then jump to ENDIF instruction and continue to run the instruction under ENDIF, and the instruction between IF and ENDIF is not executed.

The judgment condition of IF is (comparison number 1 comparison method comparison number 2), for example, comparison number 1 is 2, comparison number 2 is 1, comparison method is ">", then 2 > 1, judgment condition is valid; if the comparison method is "<" or "==", judgment condition is not valid.

IF instruction can be used alone or in combination with the ELSEIF and ELSE instruction. Note that ELSEIF and ELSE instructions cannot be used separately from IF instruction !

Note that when the beginning of the program is IF and the last action is ENDIF instruction, please insert a 0.1 second TIMER (Delay) instruction above or below the IF instruction.

Otherwise, if the condition of the IF instruction is not satisfied, the program will crash.

When IF instruction is inserted, ENDIF instruction will be inserted at the same time. When deleting IF instruction, please note that the corresponding ENDIF instruction is also deleted. Otherwise, the program will not be able to executed.

Another IF instruction or other conditional judgment class instruction such as WHILE and JUMP can be nested in the instruction.

Parameter	Meaning
Parameter type	The type of the comparison number 1,the input value of a variable or a number or an analog
Parameter name	If the type of the previous selection is a variable (INT, DOUBLE, BOOL, GINT, GDOUBLE, GBOOL), here is the variable name of comparison number 1 If the type selected in the previous item is the input value (DIN, AIN), then here is the port number for digital or analog input
Comparison method	== equal to < less than more than <= less than or equal to >= more than or equal to != not equal to
Variable value source	The type of the comparison number 2, customize or input values of variables or numbers or analogs
New parameter	If the type of the previous selection is custom, it is not optional here.If the type of the previous selection is a variable (INT, DOUBLE, BOOL, GINT, GDOUBLE, GBOOL), here is the variable name of comparison number 1.If the type selected in the previous item is the input value (DIN, AIN), then here is the port number for digital or analog input.
Source parameter	If the variable value source is selected as custom, the value of comparison number 2 is filled in directly here.

Case 1

Premise: Global variables or local variables have been defined, such as GI001=8 Instructions: IF (GI001<9)

Other instructions, such as MOVJ, etc. ENDIF

Meaning: If GI001<9, run the instruction between IF and ENDIF, if it is not satisfied, it will not run.

Process: Because GI001=8<9, the condition is valid, the instruction between IF and ENDIF is run, and the instruction following ENDIF is continued after running.

Case 2

Premise: Global variables or local variables have been defined, such as GI001=5, D001=8.88 Instructions: IF(GI001>=D001)

Other instructions, such as MOVJ, etc. ENDIF

Meaning: If GI001>=D001, run the instruction between IF and ENDIF, if it is not satisfied, it will not run.

Process: Because GI001 = 5, D001 = 8.88, 5 < 8.88, the condition is not valid, and the instruction between IF and ENDIF will not be run. The program jumps to the next line of instruction under ENDIF and continues to run.

Case 3

Premise: An external IO equipment is connected, such as the input value of port 10 of digital IO is 1.

Instructions: IF (DIN10=1)

Other instructions, such as MOVJ, etc. ENDIF

Meaning:If the input value of the digital IO port 10 is equal to 1, the instruction between IF and ENDIF is run, but if it is not satisfied, it will not run.

Process: Because the input value of port 10 of digital IO is 1,that is, DIN10 = 1, the condition is satisfied. After running the instructions between IF and ENDIF, the instructions under ENDIF are continued.

10.4. ELSE

The ELSE instruction must be inserted between IF and ENDIF, but only one ELSE instruction can be embedded in an IF instruction.

When the judgment condition of the IF is valid, the instruction between the IF and the ELSE is executed, and the next line of the jump to the ENDIF instruction continues to run, instead of running the instruction between ELSE and ENDIF.

When the judgment condition of the IF is not valid, it will jump to the instruction running between ELSE and ENDIF, instead of running the instruction between IF and ELSE.

Note that when you delete IF instructions, you need to delete the corresponding ELSE and ENDIF instructions, otherwise the program will not run.

Case 1

Premise: Global variables or local variables have been defined, such as GI001=8 Instructions: IF (GI001<9)

Other instructions 1, such as MOVJ, etc. ELSE

Other instructions 2, such as MOVJ, etc. ENDIF

Meaning: If GI001<9, the instruction 1 between IF and ELSE is run, and if it is not, the instruction 2 between ELSE and ENDIF is run.

Process: Because GI001=8<9, the condition is valid, the instruction between IF and ELSE is run, and the instruction following ENDIF is continued after running.

Case 2

Premise: Global variables or local variables have been defined, such as GI001=5, D001=8.88 Instructions: IF(GI001>=D001)

Other instructions 1, such as MOVJ, etc. ELSE

Other instructions 2, such as MOVJ, etc. ENDIF

Meaning: If GI001>=D001, the instruction 1 between IF and ELSE is run, and if it is not, the instruction 2 between ELSE and ENDIF is run.

Process: Because GI001 = 5, D001 = 8.88, 5 < 8.88, the condition is not valid. Instruction 2 between ELSE and ENDIF will be run, and then the instructions under ENDIF will continue to run.

10.5. ELSEIF

The ELSEIF instruction must be inserted between IF and ENDIF. An ELSE instruction or multiple ELSEIF instructions can also be inserted between ELSEIF and ENDIF.

When the IF condition is satisfied, the instructions between ELSEIF and ELSEIF and ENDIF will be ignored, only the instructions between IF and ELSEIF will be run, and then jump to the next line of instructions under ENDIF to continue running.

When the condition of IF is not satisfied, it will jump to ELSEIF instruction to judge the condition of ELSEIF. If it is satisfied, it will run the instruction between ELSEIF and ENDIF, and then continue to run the instruction under ENDIF. If it is not satisfied, it will jump directly to one line of instruction under ENDIF to continue to run.

If multiple ELSEIFs are nested in IF and ENDIF, the first ELSEIF judgment condition is judged when the judgment condition of IF is not valid, and if it is, the instructions between the first ELSEIF and the second ELSEIF are run; if not, the second ELSEIF judgment condition is judged, and so on. Note that when you delete IF instructions, you need to delete the corresponding ELSEIF and ENDIF instructions, otherwise the program will not run.

Case 1

Premise: Global variables or local variables have been defined, such as GI001=8 Instructions: IF (GI001<9)

Other instructions 1, such as MOVJ, etc. ELSEIF (GI001>7)

Other instructions 2, such as MOVJ, etc. ENDIF

Meaning: If GI001<9, the instruction 1 between IF and ELSEIF is run. If it is not satisfied, the judgment condition of ELSEIF is judged. If it is satisfied, the other instruction 2 is run. If it is not satisfied, the instruction jumped to the ENDIF continues to run.

Process: Because GI001=8<9, the condition is valid, the instruction between IF and ELSEIF is run, and the instruction following ENDIF is continued after running.

Case 2

Premise: Global variables or local variables have been defined, such as GI001=5, D001=8.88 Instructions: IF(GI001>=D001)

Other instructions 1, such as MOVJ, etc. ELSEIF (D001<9)

Other instructions 2, such as MOVJ, etc. ENDIF

Meaning: If GI001>=D001, the instruction 1 between IF and ELSE is run. If it is not satisfied, the judgment condition of ELSEIF is judged. If it is satisfied, other instruction 2 is run. If it is not satisfied, the instruction jumped to ENDIF continues to run.

Process: Because GI001 = 5, D001 = 8.88, 5 < 8.88, the condition is not valid, the condition of ELSEIF is judged, because D001 = 8.88 < 9, if the condition is valid, the other instruction 2 is run.

Case 3

Premise: Global variables or local variables have been defined, such as GI001=5, D001=8.88 Instructions: IF(GI001>=D001)

Other instructions 1, such as MOVJ, etc. ELSEIF(D001>9)

Other instructions 2, such as MOVJ, etc. ELSE

Other instructions 3, such as MOVJ, etc.

ENDIF

Meaning: If GI001>=D001, the instruction 1 between IF and ELSE is run. If it is not satisfied, the judgment condition of ELSEIF is judged. If it is satisfied, other instruction 2 is run. If it is not satisfied, the instruction jumped to ENDIF continues to run.

Case 3

Premise: Global variables or local variables have been defined, such as GI001=5, D001=8.88 Instructions: IF(GI001>=D001)

Other instructions 1, such as MOVJ, etc. ELSEIF(D001>9)

Other instructions 2, such as MOVJ, etc. ELSE

Other instructions 3, such as MOVJ, etc.

ENDIF

Meaning: If GI001>=D001, the instruction 1 between IF and ELSE is run. If it is not satisfied, the judgment condition of ELSEIF is judged. If it is satisfied, other instruction 2 is run. If it is not satisfied, the instruction jumped to ENDIF continues to run.

Process: Because GI001 = 5, D001 = 8.88, 5 < 8.88, the condition is not valid, the condition of ELSEIF is judged, because D001 = 8.88 < 9, if the condition is valid, the other instruction 3 is run.

Case 4

Premise: Global variables or local variables have been defined, such as GI001=5, D001=8.88 Instructions: IF(GI001>=D001)

Other instructions 1, such as MOVJ, etc. ELSEIF(D001>9)

Other instructions 2, such as MOVJ, etc. ELSEIF (GI001<6)

Other instructions 3, such as MOVJ, etc. ELSEIF (GI001>4)

Other instructions 4, such as MOVJ, etc. ENDIF

Meaning: If GI001>=D001, the instruction 1 between IF and ELSE is run. If it is not satisfied, the judgment condition of the first ELSEIF is judged. If D001>9 is satisfied, other instructions 2 are run. If not, the second ELSEIF is judged. The judgment condition is that if GI001<6, other instructions 3 are run, if not, the third ELSEIF is judged, and so on.

Process: Because GI001 = 5, D001 = 8.88, 5 < 8.88, then the condition is not valid, judging the ELSEIF judgment condition, because D001 = 8.88 < 9, the condition is not valid, judging the second ELSEIF, GI001 = 5 < 6, if the condition is valid, then run the other instruction 3, and then jump to the instruction under ENDIF to continue running.

Other instructions 1,

Such as MOVJ, etc. ELSEIF(D001>9)

Other instructions 2, such as MOVJ, etc. ELSEIF (GI001<6)

Other instructions 3, such as MOVJ, etc. ELSEIF (GI001>4)

Other instructions 4, such as MOVJ, etc. ENDIF

Meaning: If GI001>=D001, the instruction 1 between IF and ELSE is run. If it is not satisfied, the judgment condition of the first ELSEIF is judged. If D001>9 is satisfied, other instructions 2 are run. If not, the second ELSEIF is judged. The judgment condition is that if GI001<6, other instructions 3 are run, if not, the third ELSEIF is judged, and so on.

Process: Because GI001 = 5, D001 = 8.88, 5 < 8.88, then the condition is not valid, judging the ELSEIF judgment condition, because D001 = 8.88 < 9, the condition is not valid, judging the second ELSEIF, GI001 = 5 < 6, if the condition is valid, then run the other instruction 3, and then jump to the instruction under ENDIF to continue running.

10.6. WHILE

When the condition of WHILE instruction is satisfied, the instruction between WHILE and ENDWHILE will be run circularly. If the judgment condition is not satisfied before running to the WHILE instruction, it will jump to the ENDWHILE instruction instead of the instruction between WHILE and ENDWHILE when running to the WHILE instruction; if the judgment condition becomes unsatisfactory during the process of running the instruction between WHILE and ENDWHILE, it will continue to run until running to the ENDWHILE line, and it will not circulate but continue to run the instruction under ENDWHILE.

The judgment condition of WHILE is (comparison number 1 comparison method comparison number 2), for example, comparison number 1 is 2, comparison number 2 is 1, comparison method is ">", then 2>1, the judgment condition is valid; if the comparison mode is "" <" or "==", the judgment condition is not valid.

Note that inserting the WHILE instruction will also insert the ENDWHILE instruction. To delete the WHILE instruction, delete the corresponding ENDWHILE instruction at the same time, otherwise the program will not run.

When the program starts with WHILE and the last instruction is ENDWHILE, insert a 0.3 second TIMER (Delay) instruction at the beginning or end of the program. Otherwise, when the condition of WHILE instruction is not satisfied, the program will crash.

When instructions in WHILE do not have motion instructions or may fall into a dead cycle in some cases, please insert a 0.3 second TIMER (Delay) instruction between WHILE and ENDWHILE. Otherwise, when the conditions of WHILE instructions are satisfied, the program may crash.

The WHILE instruction can be used to nest multiple judgment instructions such as WHILE, IF or JUMP at the same time.

Parameter	Meaning

Parameter type	The type of the comparison number 1,the input value of a variable or a number or an analog
Parameter name	If the type of the previous selection is a variable (INT, DOUBLE, BOOL, GINT, GDOUBLE, GBOOL),then here is the variable name of comparison number 1.If the type of the previous selection is input value (DIN, AIN), then here is the port number of the digital input or analog input.
Comparison method	== equal to < less than more than <= less than or equal to >= more than or equal to != not equal to
Variable value source	The type of the comparison number 2, customize or input values of variables or numbers or analogs
New parameter	If the type of the previous selection is custom, it is not optional here. If the type of the previous selection is a variable (INT, DOUBLE, BOOL, GINT, GDOUBLE, GBOOL), here is the variable name of comparison number 1. If the type selected in the previous item is the input value (DIN, AIN), then here is the port number for digital or analog input.
Source parameter	If the variable value source is selected as custom, the value of comparison number 2 is filled in directly here.

Case 1

Premise: The variable GI001=1has been defined. Instruction:WHILE (GI001<2)

Other instructions ENDWHILE

Meaning: When GI001 < 2, other instructions between WHILE and ENDWHILE are circulated. Until the condition is not valid, the instructions running to ENDWHILE will not be recycled, but continue to run the instructions under ENDWHILE.

Process: Because GI001 = 1 < 2, other instructions between WHILE and ENDWHILE are circulated. Until the condition is not valid, the instructions running to ENDWHILE will not be recycled, but continue to run the instructions under ENDWHILE.

Case 2

Premise: The variable GI001=1, D001=7 has been defined. Instruction: WHILE (GI001<2)

Other commands 1, MOVJ, etc. WHILE (D001<10)

Other instructions 2, MOVJ, etc. ADD D001 1

ENDWHILE

Other instructions 3

ADD GI001 1 ENDWHILE

Meaning: When GI001 < 2, all instructions between WHILE and ENDWHILE will be run circularly. When running to WHILE (D001 < 10), D001 < 10 will be judged. If it is valid, other instructions 2 and ADD instructions will be run circularly until D001 > = 10, jump out of the intermediate WHILE instructions, continue to run other instructions 3 and ADD instructions, and then recycle until GI001 > = 2 jumps out. WHILE.

Process: Initial GI001 = 1 < 2, D001 = 7 < 10, so the judgment conditions of the two WHILE instructions are all valid at first, and the other instructions 2 and ADD instructions between WHILE (D001 < 10) and intermediate ENDWHILE will be circulated. D001 = 10 will be added to D001 once per loop. After three loops, D001 = 10 will be added. The intermediate judgment conditions will not be valid. Continue to run other instructions 3 and ADD GI001 1 instructions, GI001 plus 1 once per loop, and G after running 1 time. I001 = 2, the condition is not valid, continue to run the instructions under ENDWHILE.

Other instructions 2, MOVJ, etc. ADD D001 1
ENDWHILE

Other instructions 3
ADD GI001 1 ENDWHILE

Meaning: When GI001 < 2, all instructions between WHILE and ENDWHILE will be run circularly. When running to WHILE (D001 < 10), D001 < 10 will be judged. If it is valid, other instructions 2 and ADD instructions will be run circularly until D001 > = 10, jump out of the intermediate WHILE instructions, continue to run other instructions 3 and ADD instructions, and then recycle until GI001 > = 2 jumps out. WHILE.

Process: Initial GI001 = 1 < 2, D001 = 7 < 10, so the judgment conditions of the two WHILE instructions are all valid at first, and the other instructions 2 and ADD instructions between WHILE (D001 < 10) and intermediate ENDWHILE will be circulated. D001 = 10 will be added to D001 once per loop. After three loops, D001 = 10 will be added. The intermediate judgment conditions will not be valid. Continue to run other instructions 3 and ADD GI001 1 instructions, GI001 plus 1 once per loop, and G after running 1 time. I001 = 2, the condition is not valid, continue to run the instructions under ENDWHILE.

10.7. WAIT

WAIT is waiting, you can select whether there is waiting time. When the "TIME" option is not checked, the WAIT instruction will remain waiting until the judgment condition is valid. If the "TIME" option is checked, the next instruction will continue to run after waiting for the parameter for a long time. If the condition becomes valid while waiting, the next instruction is executed immediately.

Parameter	Meaning
Parameter type	The type of the comparison number 1,the input value of a variable or a number or an analog
Parameter name	If the type of the previous selection is a variable (INT, DOUBLE, BOOL, GINT, GDOUBLE, GBOOL),then here is the variable name of comparison number 1. If the type of the previous selection is input value (DIN, AIN), then here is the port number of the digital input or analog input.

Comparison method	== equal to < less than more than <= less than or equal to >= more than or equal to != not equal to
Variable value source	The type of the comparison number 2, customize or input values of variables or numbers or analogs
New parameter	If the type of the previous selection is custom, it is not optional here. If the type of the previous selection is a variable (INT, DOUBLE, BOOL, GINT, GDOUBLE, GBOOL), here is the variable name of comparison number 1. If the type selected in the previous item is the input value (DIN, AIN), then here is the port number for digital or analog input.
Source parameter	If the variable value source is selected as custom, the value of comparison number 2 is filled in directly here.
TIME	Options, if not selected, wait forever until the condition is valid. If you selected, you can fill in the waiting time (seconds). After the waiting time, even if the condition is still invalid, it will jump to the next line and continue to run.

Whether continuous	If you select "Yes", the PL of the previous instruction and the PL of the next instruction can be continuous when the conditions are met before running the instruction. If you choose otherwise, PL will be interrupted.

Case

Premise: The variable GI001=1 has been defined. Instruction: WAIT(GI001==2)T = 2

Meaning: When GI001 is not equal to 2, the program stays in this instruction and waits, but after waiting more than two seconds it will no longer wait, jumping to the next program to continue running. If the condition is satisfied during the waiting process, jump to the next line immediately to continue running.

Process: Because GI001 is not equal to 2, the program stays in this instruction to wait, but after waiting more than two seconds it will no longer wait, jumping to the next program to continue running.

10.8. LABEL

The LABEL instruction must be used in conjunction with the JUMP instruction. The LABEL instruction alone is meaningless.

Parameter	Meaning
Tag name	The tag name of the LABEL instruction has been inserted, option
Analyzing conditions	Option, if selected, you can set the judgment condition. If it is not selected, it will jump directly after running to JUMP.
Parameter name	If the type of the previous selection is a variable (INT, DOUBLE, BOOL, GINT, GDOUBLE, GBOOL), then here is the variable name of comparison number 1. If the type of the previous selection is input value (DIN, AIN), then here is the port number of the digital input or analog input.
Comparison method	== equal to < less than more than <= less than or equal to >= more than or equal to != not equal to
Variable value source	The type of the comparison number 2, customize or input values of variables or numbers or analogs
New parameter	If the type of the previous selection is custom, it is not optional here. If the type of the previous selection is a variable (INT, DOUBLE, BOOL, GINT, GDOUBLE, GBOOL), here is the variable name of comparison number 1. If the type selected in the previous item is the input value (DIN, AIN), then here is the port number for digital or analog input.
Source parameter	If the variable value source is selected as custom, the value of comparison number 2 is filled in directly here.
Parameter	Meaning
Tag name	A string starting with a character, with a maximum length of 8 characters

10.9. JUMP

JUMP is used for jumps and must be used in conjunction with the LABEL (label) instructions. JUMP can set whether there is a judgment condition or not. When set to no judgment condition, running to the instruction will jump directly to the corresponding LABEL instruction and continue to run the next line of instructions of LABEL.

When set to have a judgment condition, if the condition is satisfied, jump to the LABEL instruction line; if the condition is not satisfied, ignore the JUMP instruction and continue to run the next line of the JUMP instruction.

LABEL tags can be inserted above or below JUMP, but it cannot be jumped across programs.

The label name of LABEL must be two or more characters beginning with the letter.

Inserting LABEL tags has no effect on the running of programs, but it should conform to the rules of program running, such as not inserting on MOVC instructions or on local variable definition instructions.

Case 1

Premise: The variable GI001=1has been defined. Instruction:WHILE (GI001<2)

Other instructions ENDWHILE

Meaning: When GI001 < 2, other instructions between WHILE and ENDWHILE are circulated. Until the condition is not valid, the instructions running to ENDWHILE will not be recycled, but continue to run the instructions under ENDWHILE.

Process: Because GI001 = 1 < 2, other instructions between WHILE and ENDWHILE are circulated. Until the condition is not valid, the instructions running to ENDWHILE will not be recycled, but continue to run the instructions under ENDWHILE.

Case 2

Premise: The variable GI001=1, D001=7 has been defined. Instruction: WHILE (GI001<2)

Other commands 1, MOVJ, etc. WHILE (D001<10)

Other instructions 2, MOVJ, etc. ADD D001 1

ENDWHILE

Other instructions 3

ADD GI001 1 ENDWHILE

Meaning: When GI001 < 2, all instructions between WHILE and ENDWHILE will be run circularly. When running to WHILE (D001 < 10), D001 < 10 will be judged. If it is valid, other instructions 2 and ADD instructions will be run circularly until D001 > = 10, jump out of the intermediate WHILE instructions, continue to run other instructions 3 and ADD instructions, and then recycle until GI001 > = 2 jumps out. WHILE.

Process: Initial GI001 = 1 < 2, D001 = 7 < 10, so the judgment conditions of the two WHILE instructions are all valid at first, and the other instructions 2 and ADD instructions between WHILE (D001 < 10) and intermediate ENDWHILE will be circulated. D001 = 10 will be added to D001 once per loop. After three loops, D001 = 10 will be added. The intermediate judgment conditions will not be valid. Continue to run other instructions 3 and ADD GI001 1 instructions, GI001 plus 1 once per loop, and G after running 1 time. I001 = 2, the condition is not valid, continue to run the instructions under ENDWHILE.

10.10. The UNTIL

UNTIL instruction is used to jump out during a movement. That is, during one movement of the robot, pause and start the next process. When the condition is satisfied, regardless of whether the current robot is running or not, immediately pause and start an instruction under the ENDUNTIL instruction.

The judgment condition of UNTIL is (comparison number 1 comparison method comparison number 2), for example, comparison number 1 is 2, comparison number 2 is 1, comparison method is ">", then 2>1, the judgment condition is valid; if the comparison mode is "" <" or "==", the judgment condition is not valid.

Note that the ENDUNTIL instruction is inserted at the same time as the UNTIL instruction is inserted. To delete UNTIL instructions, delete the corresponding ENDUNTIL instructions at the same time, otherwise the program will not run.

Parameter	Meaning
Parameter type	The type of the comparison number 1,the input value of a variable or a number or an analog
Parameter name	If the type of the previous selection is a variable (INT, DOUBLE, BOOL, GINT, GDOUBLE, GBOOL),then here is the variable name of comparison number 1. If the type of the previous selection is input value (DIN, AIN), then here is the port number of the digital input or analog input.
Comparison method	== equal to < less than more than <= less than or equal to >= more than or equal to != not equal to
Variable value source	The type of the comparison number 2, customize or input values of variables or numbers or analogs
New parameter	If the type of the previous selection is custom, it is not optional here. If the type of the previous selection is a variable (INT, DOUBLE, BOOL, GINT, GDOUBLE, GBOOL), here is the variable name of comparison number 1. If the type selected in the previous item is the input value (DIN, AIN), then here is the port number for digital or analog input.
Source parameter	If the variable value source is selected as custom, the value of comparison number 2 is filled in directly here.

Case

Premise: The variable GI001=1 has been defined. Instructions: UNTIL (GI001<2)

Other instructions ENDUNTIL MOVJ P003

Meaning: When running "other instructions" between UNTIL and ENDUNTIL, if GI001 becomes a value of < 2, the current action is suspended and the MOVJ P003 instruction is jumped to; if GI001 is always > 2, the MOVJ P003 instruction is run after running other instructions.

10.11. CRAFTLINE

Process skip instruction. Use with special process. Use with special craft skipping

Parameter	Meaning
New parameter	Fill in the number of lines in the special process program

10.12. CMDNOTE

Instructions comments. You comment contan use this instruction to add comments to the appropriate position of the program for easy debugging

Parameter	Meaning
Comment content	Support Chinese and English

10.13. POS_REACHABLE

The judgment instruction is reached. Used to judge whether the target point can be reached. If the point can be reached, the variable is set to 1, otherwise it is set to 0

Parameter	Meaning
Location variable name	P point and G point can be selected
Exercise type	Can choose MOVJ, MOVL
State is stored in variable type	Can be stored in BOOL, GBOOL

Example Prerequisite: BOOL variable A001 has been defined, and position variable P001 has been defined Command: POS_REACHABLE MOVJ P001 A001 Meaning: Determine whether to use MOVJ interpolation to run to P001 position. A001 value of 1 means reachable, A001 value of 0 means unreachable.	
State is stored in variable name	BOOL, GBOOL variable name

10.14. CLKSTART

The CLKSTART instruction is used for timing. Run this command to start timing and record the time in a local or global DOUBLE variable.

Parameter	Meaning
Serial number	The serial number of the timer can be counted separately by using 32 timers at the same time.
Stored variable type	Store the timed time into the local DOUBLE variable or the global GDOUBLE variable.
Save the variable name	The variable name of the variable where the time is stored.

10.15. CLKSTOP

The CLKSTOP instruction is used to stop the timing of the timer corresponding to the serial number. The value stored in the variable will not return to zero after stopping.

Parameter	Meaning
Serial number	The serial number of the timer to stop timing.

10.16. CLKRESET

The CLKRESET instruction is used to reset the timer corresponding to the serial number to zero. If this command is not used, the next time the CLKSTART command is run, the time will be accumulated.

Parameter	Meaning
Serial number	The serial number of the timer to be reset to zero.

11. Background task

11.1. limit

Currently, only the following commands are supported in the background task program:

category	instruction	content
Input and output class	DIN	IO input
	DOUT	IO output
	AIN	Analog input
	AOUT	Analog output
	READ_DOUT	Read output
Timer class	TIMER	Delay
Operation class	ADD	plus
	SUB	Less
	MUL	Multiply
	DIV	except
	MOD	mold
	SIN	Sine
	COS	Cosine
	ATAN	Arctangent
	LOGICAL_OP	logic operation
Condition control class	IF	in case
	ELSEIF	Otherwise if
	ELSE	otherwise
	WAIT	wait
	WHILE	Inner loop
	LABEL	label
	JUMP	Jump
	CLKSTART	Timing begins
	CLKSTOP	Time ends
	CLKRESET	Timer reset

Variable class	SETINT	Assigned integer
	SETDOUBLE	Assign floating point
	SETBOOL	Assign Boolean
Communication class	SENDMSG	send data
	PARSEMSG	Analytical data
	READCOMM	Read
	OPENMSG	Open data
	CLOSEMSG	Close data
	PRINTMSG	Output Data
	MSG_CONN_ST	Get information connection status
Position variable class	USERFRAME_SET	User coordinate modification
	TOOLFRAME_SET	Tool coordinate modification
	READPOS	Read point
	POSADD	Point plus
	POSSUB	Point minus
	POSSET	Point change
	COPYPOS	Replication point
Coordinate switching class	SWITCHUSER	Switch user coordinates
Program control class	PAUSERUN	Suspended
	CONTINUERUN	Keep running
	STOPRUN	Stop running
	RESTARTRUN	Rerun

11.2. Note

Note: Press the pause button in running mode, and IO pause in remote mode only pauses the main program, not background tasks.

11.3. Background task programming

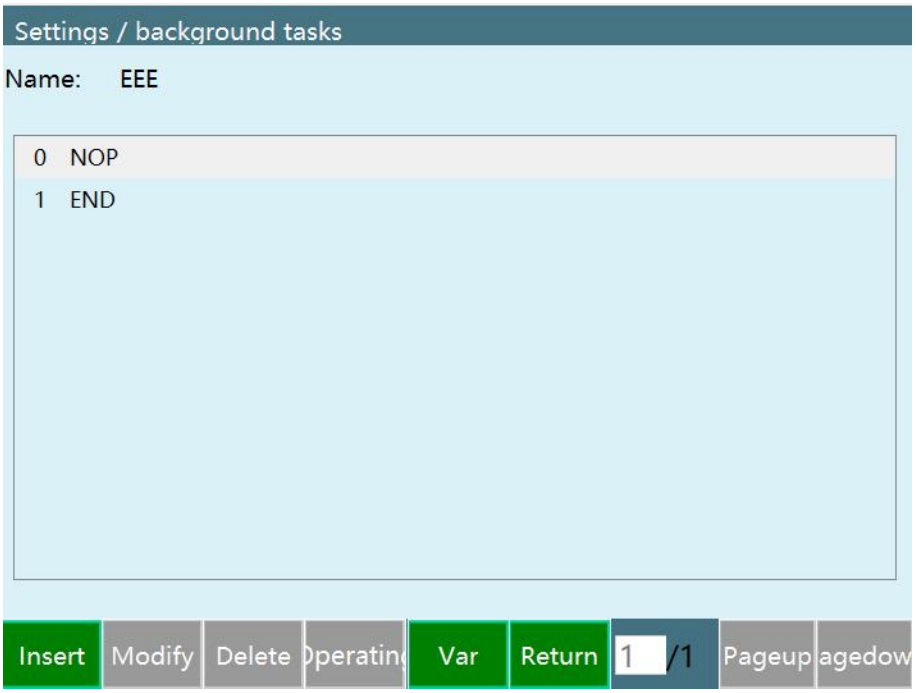
The program that needs to run in the background task needs to be carried out in "Settings- Background Task", and its programming is the same as writing ordinary programs.

11.3.1. Notice

It is best to insert a delay of 0.2s in the WHILE loop and the last line of the entire program.

When editing the background task program, to debug, only the "STEP" single-step operation mode is provided. To run the debugging as a whole, please insert the PTHREAD_START instruction in the main program and run it for debugging.

The background task is started and executed only once. It can be used in conjunction with WHILE instruction if loop judgment is needed.



11.4. Main program programming

If you want to run background tasks in the main program, you need to insert the PTHREAD_START (start thread) instruction in the program. To exit the background task, insert the PTHREAD_END (exit thread) instruction.

The background task only starts to run after running PTHREAD_START, and the background program does not pause when the main program is paused.

Conditions for stopping background tasks:

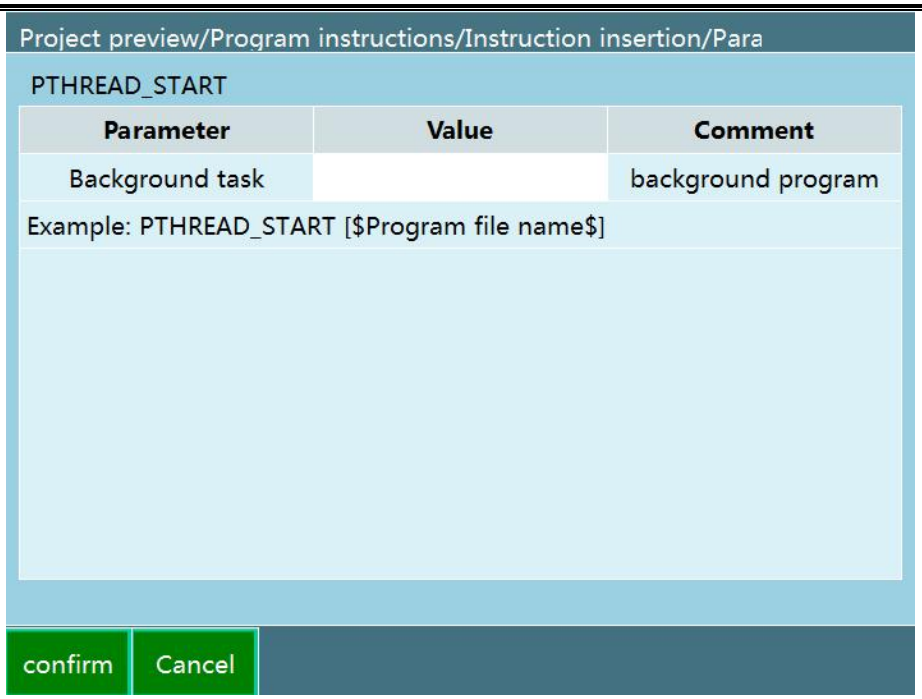
1.The program runs to the PTHREAD_END instruction;

2.The program stops and the robot stops enabling.。

11.4.1. PTHREAD_START（Start thread）

Run the PTHREAD_START command to start the background task. This instruction is located in the program control instruction.

When inserting the instruction, click the [value] input box to automatically pop up the established background task list, select the background task to be run, and click the [OK] button to select the program.

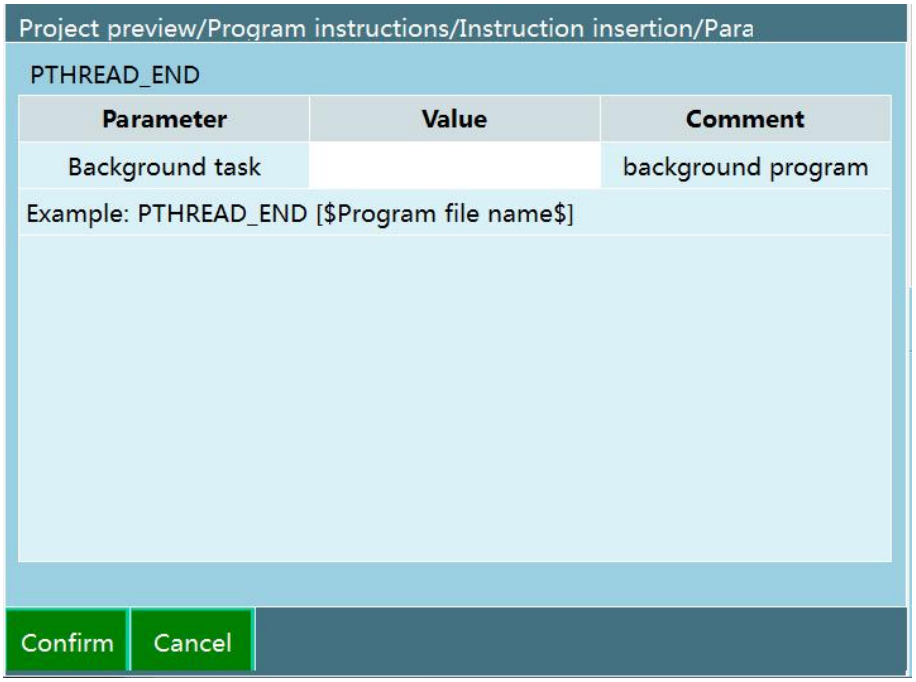


When running the main program to the PTHREAD_START instruction, the background task is started.

11.4.2. PTHREAD_END（Close thread）

Running the PTHREAD_END instruction will exit the corresponding background task that has been running.

Its insertion and modification method is the same as PTHREAD_START.



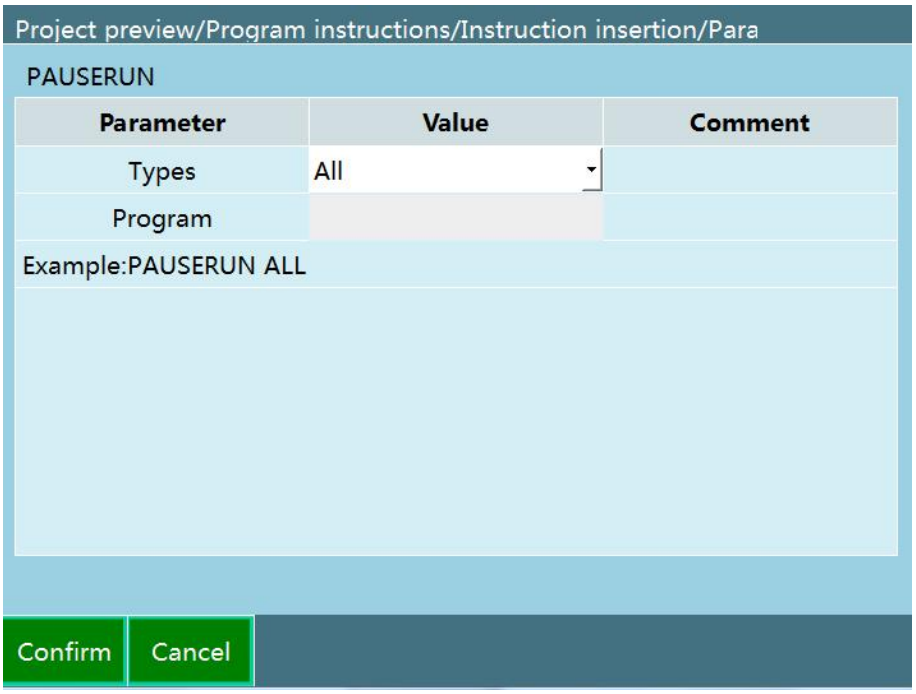
11.4.3. PAUSERUN （Pause thread）

Running the PAUSERUN instruction will suspend all tasks, main programs, or background tasks.

When inserting the instruction, click the [Value] drop-down box of the type, select the type of control, and click the [Value] input box of the program, a list of established background tasks will pop up

automatically, select the background task to be run, and click the [OK] button , The program will be selected, and it is not selectable for all and main programs.

Note: Pressing the stop key of the teaching box only pauses the main program.

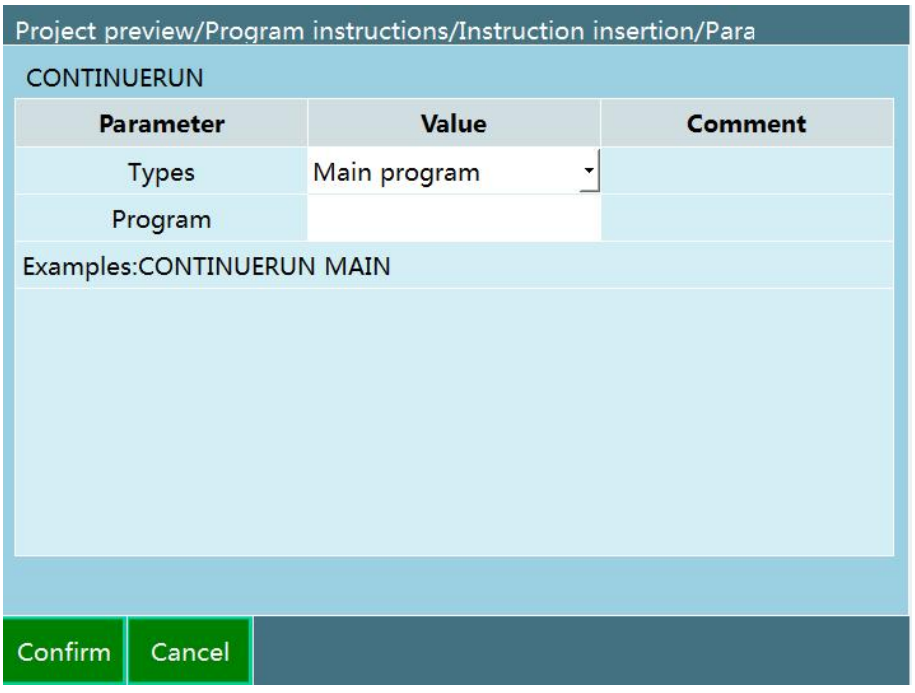


When the running program reaches the PAUSERUN instruction, all the set tasks, main program, or background tasks will be suspended.

11.4.4. CONTINUERUN（Continue thread）

Running the CONTINUERUN instruction will continue to run the main program or background tasks.

When inserting the instruction, click the [Value] drop-down box of the type, select the type of control, and click the [Value] input box of the program, a list of established background tasks will pop up automatically, select the background task to be run, and click the [OK] button . The program will be selected, and the main program cannot be selected.

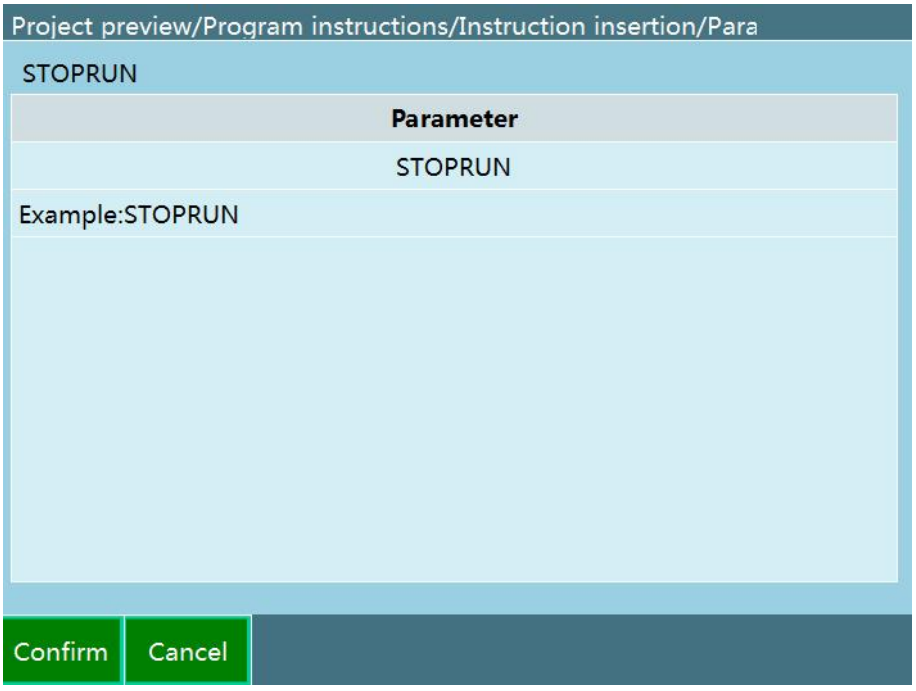


When the program is running, when it reaches the CONTINUERUN instruction, the main program or

background task will continue to run.

11.4.5. STOPRUN （Stop running）

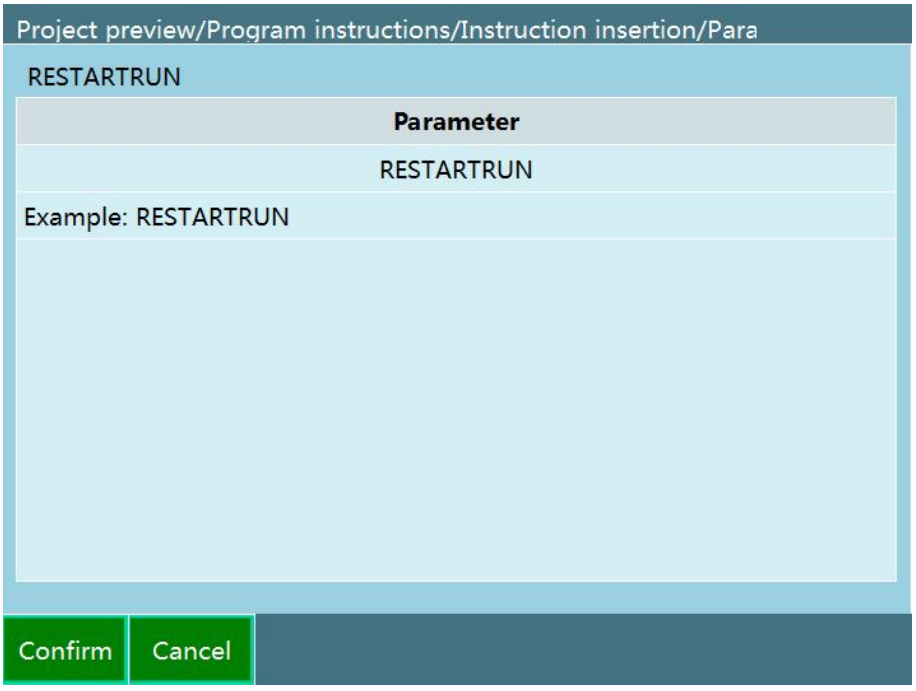
Running the STOPRUN instruction will stop all tasks from running.



The instruction can be inserted directly by clicking to confirm, no need to set parameters.

11.4.6. RESTARTRUN（Rerun）

Running the RESTARTRUN instruction will rerun all tasks.



The instruction can be inserted directly by clicking to confirm, no need to set parameters.