

H Series Robot Operation and Programming Instructions V.2.1



The update of this Instructions book is authorized and organized by Guangzhou Auctech Automation Technology Ltd. Without authorization or written permission from our company, no unit or individual has the right to modify or correct the content of this Instructions, and our company is not responsible for any customer losses caused by this.

In this Instructions book, we will strive to narrate various events related to it. Due to space limitations and product development positioning, it is not possible to provide a detailed description of all unnecessary or impossible events. Therefore, events not specifically described in this Instructions book can be considered as "impossible" or "not allowed" events.

We have checked whether the content of the printed materials is consistent with the hardware and software content described, and there may be inconsistencies. We will regularly proofread the content of printed materials and make necessary changes in future versions.

The copyright of this Instructions book belongs to Guangzhou Auctech Automation Technology Ltd. Any unit or individual who publishes or copies it is illegal and our company will hold them legally responsible.

Copyright © Guangzhou Auctech Automation Technology Ltd. All rights reserved.

Content

The introduction	1
1 Safety	3
1.1 General Safety rules	3
1.2 Instructions Description of Safety Responsibility	3
1.4 Precautions for robot and installation	5
1.5 Precautions for robot teaching	7
1.6 Precautions for robot operation	8
1.7 Notes on robot automatic operation	9
1.8 Precautions for operator operation	10
1.9 Safety features of robots	12
1.10 Common situation on site	13
2 System Hardware Composition	15
2.1 Ontology	15
2.2 Electric control system	15
2.2.1 Electric control cabinet	16
2.2.2 The controller unit	18
2.2.3 The drive unit	18
2.2.4 IO unit	19
2.2.5 The drive control unit	19
3 Common System Functions	26
3.1 Operation Interface	26
3.2 View the current location	37
3.3 IO situation display	38
3.4 AO situation display	39
3.5 Register variable	41
3.6 Model information configuration	42
3.7 Mechanical arm parameter information	43
3.8 Zero point calibration and recovery of zero point	45
3.9 Joint restriction	47
3.10 Spatial area configuration	48
3.10.1 Area configuration parameters	49
3.10.2 Regional Types and Response Methods	50
3.11 Calibration of the tool-workpiece coordinate system	55
3.11.1 Calibration of workpiece coordinate system using the three-point method	55
3.11.2 Calibration of tool coordinate system	56
3.12 System variables	65
4 Operation	69
4.1 Operation mode Introduction	69
4.2 Coordinate system type selection	70
4.3 Tool workpiece coordinate system selection	70
4.4 Speed ratio	72
4.5 Worldwide inching and inching operationInching	73
4.6 Moving to the point	74
5 Programming	76
5.1 Program structure	76
5.2 Program type	77
5.2.1 Daemon configuration function	78
5.3 Program Creation	81
5.4 Program Editing	82
5.5 Point online editing	83
5.6 Program execution	85
6 File management	87
6.1 Program file management	87
6.1.1 The import and export	87
6.1.2 File lock	89
6.2 Program data file management	90
6.3 User PLC file management	92

6.4 File management system parameters	93
6.5 Log and system snapshot files	95
7 System management	98
7.1 The user login	98
7.2 User rights configuration	101
7.3 Controller connection	102
7.4 Controller IP Configuration	103
7.4.1 IP description and configuration application	105
7.5 System authorization	107
7.6 System software restarts	107
7.7 System update.....	109
7.8 Switching System language.....	109
7.9 System version information	110
8 Application functions	111
8.1 Encoder-decoder function	111
8.2 One key back to zero function	112
8.3 Register Binary Bit Operations	114
8.4 External control signal configuration	115
8.4.1 iSAFE_DOOR Safety Gate Signal	117
8.5 External Runtime Program Configuration Operations	119
8.6 Reference point signal output function	121
8.7 Synergy Functions	123
8.7.1 Co-Configuration of the Translocator	123
8.7.1.2 Co-calibration	125
8.7.2 Collaborative teaching	131
8.7.3 Ground Track Synergy	132
8.8 Fallback function	133
8.9 Displaying global variable descriptions on program lines	138
8.10 Displaying the current state of the IO in a program line	139
8.11 Oscillator Backup Button Configuration	140
8.12 Calibrating the external shaft reduction ratio	142
8.13 External TCP Functions	143
8.13.1 Fixture Tool TCP Calibration	144
8.13.2 External tool TCP calibration	145
8.13.3 Demonstration and external TCP commands	147
8.14 Speed Configuration	147
8.14.1 Safe Speed Trigger and Recovery	147
8.14.2 Reset Speed Trigger and Recovery	151
8.14.3 Signal Speed Configuration	153
8.15 Path Keeping Function	157
8.16 Follower function	160
8.16.1 Introduction to Functions	160
8.16.2 Follow-on commands	161
8.16.3 Compensation Functions and MASK Options	162
8.16.4 Follower parameters	165
8.17 Craft Kits	166
8.18 Offline programming and PC debugging	166
9 Dynamics Functions	167
9.1 Mounting attitude setting	167
9.2 Ontology Friction Recognition Function	169
9.3 Load Configuration Functions	172
9.3.1 Dynamic load switching	172
9.3.2 Load Parameter Configuration	173
9.4 Collision detection function	177
9.4.1 Collision Detection Switch	177
9.5 Drag-and-drop teaching function	180
9.5.1 Drag-and-drop instructional settings	180
9.5.2 Drag Track Recording	181

9.6 Soft Float Function	188
9.6.1 Steps and Parameters Instructions	188
9.6.2 Soft Float Instructions	191
10 Communications	194
10.1 MODBUS communication	194
10.2 MODBUS Master	195
10.3 MODBUS slave	196
10.3.1 Configuration Interface	196
10.3.2 Protocol Instructions	197
10.4 SOCKET communication	210
10.4.1 SOCKET single client	210
10.4.2 SOCKET Server	211
10.4.3 SOCKET server-side one-to-many	213
11 Programming Instructions	215
11.1 summarize	215
11.2 Variable	217
11.2.1 Global variable	217
11.2.2 Program point variable	218
11.3 Point operation	218
11.4 R Register bit operation	219
11.5 Operator	219
11.6 Statement	221
11.6.1 Process statement	221
11.6.2 Conditional statement	222
11.6.3 Loop statement	224
11.7 Motion command	226
11.7.1 J joint motion command	227
11.7.2 L Linear motion command	227
11.7.3 C arc motion command	228
11.7.4 A Arc Motion Command	229
11.7.5 JUMP Gate Motion Command	230
11.7.6 REULMOVE Square hole machining motion command	231
11.8 Local motion parameters	232
11.9 Global motion parameters	233
11.9.1 Global speed parameter table	233
11.9.2 SMOOTH Flexibility level parameters	235
11.10 Trajectory control parameters	235
11.10.1 CNT_TYPE Smooth type parameter	235
11.10.2 CNT Smooth transition parameter	236
11.10.3 FINE Attribute	239
11.10.4 CR Smooth transition parameter	240
11.10.5 TURN TALARM End axis control parameters	241
11.10.6 WRISTJNT Singular parameters passing through the wrist	242
11.10.7 SEC Specify exercise time parameters	243
11.11 Sports Additional Attributes	244
11.11.1 INC Incremental motion attribute	245
11.11.2 OFFSET Position compensation attribute	245
11.11.3 TOOL_OFFSET Tool TCP Direction Compensation Attribute	247
11.11.4 SKIP Interrupt function attribute	248
11.11.5 PASS Skip waiting for motion execution instruction attribute	249
11.11.6 SET_TR Signal trigger function attribute	249
11.11.7 WAIT Determine the trigger signal attribute	251
11.12 Load setting	252
11.13 Collision detection	253
11.13.1 Collision detection switch	253
11.13.2 Collision detection sensitivity	253
11.14 IO Instruction	254
11.14.1 IO Variable instruction	254

11.14.2 VDI Virtual input instruction	254
11.14.3 PULSE Output pulse command	255
11.15 WAIT Waiting for instructions	256
11.16 WAIT TIME Delay instruction	256
11.17 Coordinate system instruction	257
11.17.1 UTOOL_NUM Tool coordinate system call instruction	257
11.17.2 UFRAME_NUM Instruction for calling the workpiece coordinate system	258
11.17.3 WORLDFRAME Command to call the world coordinate system	259
11.17.4 SET_TOOL Tool coordinate system value modification instruction	259
11.17.5 SET_FRAME Workpiece coordinate system value modification instruction	260
11.17.6 CALIUFRAme Calibration instruction for workpiece coordinate system	260
11.17.7 WCS_TEST Coordinate system detection instruction	260
11.18 Matrix arithmetic commands	261
11.18.1 MATMUL matrix multiplication instructions	261
11.18.2 MATINV matrix inverse command	262
11.19 Coordinate system conversion case	262
11.20 Co-motion commands	263
11.20.1 COORD_NUM synergy command	263
11.20.2 SYNCEXTAX Co-control Commands	264
11.21 EX_UTOOL External TCP Command	264
11.22 Process control instruction	265
11.22.1 MAINTASK foreground program name get command	265
11.22.2 RUNTASK background program name get command	265
11.22.3 RUN background program enable command	266
11.22.4 LOAD foreground program loading instruction	266
11.22.5 START programme start command	266
11.22.6 PAUSE programme pause command	267
11.22.7 END Program Abort Instruction	267
11.22.8 ABORT programme uninstallation command	268
11.23 VORD trim command	268
11.24 TIME Timing Command	269
11.25 JPOS LPOS Position Acquisition Command	269
11.26 TRACE conveyor tracking command	270
11.27 WEAVE swing command	270
11.28 SLAVE follow command	272
11.29 PALLET tray command	273
11.30 SAVE variable save command	274
11.31 PRINTF Information Printing Command	274
11.32 THROW Custom Alarm Commands	275
11.33 Maths function instructions	275
11.34 string instruction	277
11.34.1 TOSTR Convert String Instruction	277
11.34.2 TOVAL Convert Numeric Instruction	278
11.34.3 STRLEN get string length command	278
11.34.4 FINDSTR lookup string command	278
11.34.5 SUBSTR Intercept string instruction	279
11.34.6 string splicing	279
11.34.7 Character Cutting Case Procedure	280
11.35 SOCKET communication commands	280
11.35.1 OPENSOCKET creation command	281
11.35.2 CLOSESOCKET shutdown command	281
11.35.3 ISHANDLEOPEN judgement instruction	281
11.35.4 The CONNECT command	282
11.35.5 ACCEPT Monitor instructions	282
11.35.6 SEND send command	283
11.35.7 RECEIVE Receive Command	283
11.35.8 BRECBUFEMPTY Directive	284
11.35.9 SOCKETMONITOR monitor command	284

11.36 MODBUS Master Communication Command.....	285
11.36.1 OPEN_MODBUS_TCP Create Command.....	285
11.36.2CLOSE_MODBUS_TCP shutdown command.....	286
11.36.3OPEN_MODBUS_RTU Create Command.....	286
11.36.4CLOSE_MODBUS_RTU shutdown command.....	287
11.36.5READ_MDB_COILSTATUS read command.....	287
11.36.6WRITE_MDB_COILSTATUS Write Command.....	287
11.36.7READ_MDB_INPUTSTATUS read command.....	288
11.36.8 READ_MDB_INPUTREG read command.....	288
11.36.9 READ_MDB_HOLDREG read command.....	288
11.36.10 WRITE_MDB_HOLDREG Write Command.....	289
11.37 macro instruction.....	289
11.37.1 HANDPICK jaw commands.....	289
11.37.2SOFTFLOAT soft float command.....	291
12 Client service.....	293
12.1 Technical Support Enquiry.....	293
13 Appendix.....	294
13.1 Glossary.....	294
13.2 Global Motion Parameters Command Table.....	295
13.3 Table for assigning operating privileges of the demonstrator.....	296
13.4 List of external signals.....	301
13.5 SOCKET Error codes.....	305
13.6 System Signal Occupancy Table.....	306

The introduction

Introduction Thank you for buying Robot.

This manual describes how to safely and correctly using TV robot operation and programming.

Please read this manual and other related manuals carefully before using the robot, and be sure to understand and be familiar with what is explained in the manual.

Version

Relations

The software versions corresponding to this manual are as follows:

Equipment	Software/system version number
Demonstrator	HSPad3-V1.6. 11
Controller	HSC3-V1.6. 11
Drive and control in one	HSI_V1.6. 11

What to Read This manual is mainly for users of industrial robots. Please ensure that users have a basic knowledge of industrial robots.

Pictorial Description The following figures may appear in this specification and represent the following meanings:

Graphic	Instructions
	Prompt information, prompt to explain the precautions of operation.
	Warning information, warning of possible losses.
	Example messages that demonstrate specific applications of the concepts described in the examples.

Version	Revision date	Revision Notes
V2.0	2024/7/3	Supplementary Appendix 13.5, SOCKET Error Codes Correct the IPO mode selection instructions
V2.1	2024/7/4	Update the dynamics installation pose setting instructions Update the dynamics body friction identification instructions Update the dynamic load configuration description Update the dynamics drag demo setup instructions Update the drag track recording instructions

1 Safety

1.1 General Safety rules

Notes

It is the responsibility of the operator to ensure that the robot production environment complies with and complies with national and local laws, regulations and regulations regarding safety. Robots are different from other equipment, due to its fast speed, wide range of motion, complex posture changes and other characteristics in the production process, it is likely to bring personal injury or equipment accidents, so we must pay special attention to safety matters, so as not to cause unnecessary losses.

Before installation, operation and maintenance, please be sure to fully grasp the knowledge of robot equipment, safety information and all matters needing attention.

1.2 Instructions Description of Safety Responsibility

Notes

Robot is committed to providing reliable safety information, but cannot ensure the completeness of safety information. Even if the operation is carried out according to the safety precautions, it is not guaranteed that the robot will not cause personal and property losses in complex situations and uncertain factors.

1.3 Emergency stop button



Warnings:

- The emergency stop button is the only means to trigger the emergency stop state of the robot, and it is also the most important means to ensure the safety of the operator and equipment in an emergency. Therefore, when the robot is installed, taught, operated and operated, please make sure that the emergency stop button (including the external emergency stop button connected to the robot system by the integrator) can work normally.
- The normal working condition of the emergency stop button is that the device can enter the emergency stop state after pressing the emergency stop button, and the emergency stop state can be relieved after resetting the emergency stop button. Most reset operations of the emergency stop button are carried out by means of rotation, and the direction of rotation is marked on the surface of the button. There are also some buttons that are reset by pulling up.
- The operator must confirm that the emergency stop button is not abnormal before the robot can be used. All operators must strictly implement the regulations.

Indication of position

The emergency stop button location is shown below.

Demonstrator:



Electric control cabinet:





Tips:

After pressing the emergency stop button, an alarm message will pop up in the information window of the teacher. Operators need to reset the emergency stop button first, and click the [Alarm Confirmation] button to confirm the alarm information before they can continue to operate the robot.

1.4 Precautions for robot and installation

Notes on	● When considering robot safety, the robot body cannot be considered separately, but the whole robot production system environment should be considered comprehensively.
Matters	● Robots cannot perform production operations alone. Only after the end effector is installed and the peripheral equipment and equipment system are set up can the production operation be carried out. ● After installing the robot, be sure to set up a safety fence for the robot to prevent personnel from entering the robot working area during the operation of the robot, and to prevent accidents in which the control object is released or flown out due to the power supply or compressed air supply to the end effector is cut off. ● After setting up the safety fence, please be sure to place the robot's teaching device outside the safety fence. Only in this way can the movement of the whole robot be monitored. ● Make sure that no tools, fixtures, discarded objects or other obstacles are left in the robot's working area after the installation operation is completed. ● Before the robot is connected to the power supply, please confirm whether the voltage, frequency and cable specifications of the power supply meet the requirements. ● When installing the attached machine on the end-effector and manipulator, be sure to follow the size and number of bolts specified in the relevant manual, and use the torque wrench to tighten according to the specified torque. In addition, no rusty or grime bolts should be used. Non-specified fastening and imperfect methods can cause bolts to loosen, which may lead to major accidents.

- The load on the wrist and the manipulator must be controlled within the allowable weight. Failure to comply with the provisions of the allowed handling weight may lead to abnormal movements or premature damage of mechanical components.
- It is strictly prohibited to supply power, compressed air, welding cooling water outside the specification, which will affect the action performance of the robot, and cause abnormal action or failure, damage and other dangerous situations.
- Please ensure that the robot controller and peripheral equipment are properly grounded. The robot controller and peripheral equipment with interference power should be grounded separately. At the same time, if there is a source of electromagnetic interference outside the controller, noise filtering equipment should be installed on the power supply line of the robot control cabinet.
- Although the electromagnetic wave interference is related to its type or intensity, there is no perfect countermeasures with the current technology. In the robot operation, power on medium conditions, should comply with the operation precautions. The recorded data will be lost due to electromagnetic waves, other noise and substrate defects. Therefore, please regularly back up the program files and register files to external storage media such as U disk.
- Situations where robots are not allowed:

In combustible environments.

In an explosive environment.

Where there's a lot of radiation.

Water or high humidity.

1.5 Precautions for robot teaching

Before demonstration

- The teaching work should be completed by two people, one of them as a safety personnel, monitoring the whole teaching work, and stopping the robot in time through the safety device when danger is about to occur; The other person as a teaching staff, carefully and carefully complete the teaching task. The security personnel and the demonstration personnel must be the personnel with relevant training.
- Verify that the robot or peripheral equipment is not in a dangerous state and that nothing unusual has occurred.
- Confirm the range of motion of the robot, be sure not to get close to the robot or enter under the robot arm, to avoid the workpiece falling and endangering the life and safety of personnel.
- Confirm the safe passage of the teaching personnel so that they can safely evacuate in case of danger.
- Confirm the location and status of the emergency stop button (demonstrator, electric control cabinet).
- An emergency stop switch should be installed outside the safety fence where the entire movement range of the robot can be monitored, so that the safety personnel can easily and quickly press the emergency stop switch to stop the movement of the robot when they find the abnormal or dangerous movement of the robot.

In demonstration

- Whether safety personnel or teaching personnel, must always monitor the robot for abnormal movement, the robot and its surroundings for collision, clamping, extrusion risk.
- It should be confirmed that the procedure or steps are correct before proceeding with the operation. Editing procedures or steps incorrectly can lead to accidents.
- Always pay attention to the movements of the robot and do not work with your back to the robot. Even if the action of the robot is slow to react, it will lead to the possibility of accidents.

After the Demonstration

- Before confirming the teaching trajectory is correct, please ensure that there are no people and obstacles within the movement range of the robot. After confirming that there are no people and objects that may collide or interfere with the robot, run the teaching trajectory. At this time, the movement of the robot should be kept at low speed. When it is confirmed that the robot is correct, it can run at high speed.

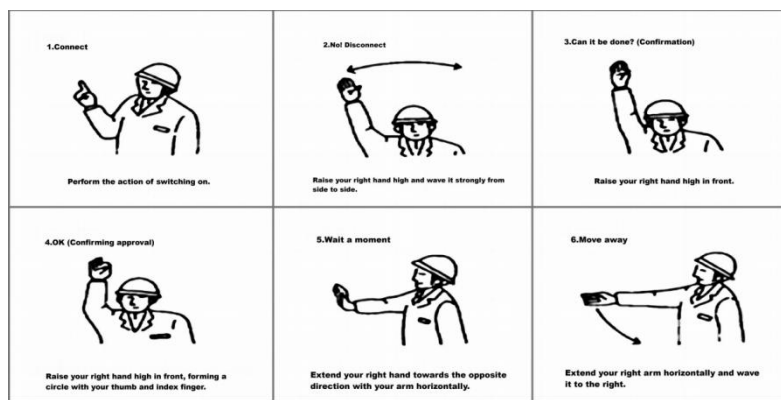
1.6 Precautions for robot operation

Notes on	Before starting the robot, please make sure that there are no people and items that
Matters	may interfere with the operation of the robot in the robot operation area.
	Before starting the robot, the function and name of each switch, display and signal of the robot should be clear, and ensure that the functions and equipment such as signal lights, external power sources, safety locks, emergency stop buttons are normal.
	If there are multiple operators in the robot application system, please make sure that the operators make clear to each other the signal of the robot to start operation, and let other operators determine that it is safe to start the robot.
	If the robot has not been running for a long time, please reduce the running rate of the robot and maintain the position where the emergency stop button can be pressed immediately to ensure that the robot can be stopped at any time.
	If the robot will produce a lot of waste, metal dust, small debris, etc., in the process of working, please cover the robot and robot controller with a suitable protective shell.
	According to the setting place and the work content, the operator should write the starting method, operation method, solution method when abnormal and other relevant operation regulations and check list of the robot, and operate in accordance with the operation regulations. Operating only with the memory and knowledge of the operator can lead to accidents due to forgetting and errors.

Industrial robot gesture method

In large robot application systems, there are usually multiple operators working collaboratively. When communicating at a distance, shouts should be avoided as far as possible, because the noise and other factors in the production environment may not convey the meaning correctly, resulting in accidents. The intention should be correctly communicated through industrial robot gestures and other ways on site.

An example gesture method is shown in the following figure:



1.7 Notes on robot automatic operation

Notes on Matters

- Be sure to confirm that the workpiece has been firmly grasped before the robot runs automatically. If the grip of the robot gripper is not enough, the robot may throw the workpiece out during operation. When the gripper is electromagnetically driven or pneumatically driven, design a corresponding failure protection system to ensure that the workpiece is not suddenly thrown out once the gripper loses its driving force.
- When the workpiece because of various circumstances, can not avoid being thrown out, to ensure that the safety fence, protective net, etc. can withstand the impact of the workpiece, can effectively protect personnel from injury.
- If the robot shows a fault and stops running, please check the fault shown and
- troubleshoise it according to the corresponding method to ensure that the robot is

in a sound state, and restart the robot in the correct recovery sequence.



Warnings:

Never assume that the robot is not moving because the robot may be waiting for an input signal to continue moving, or it may be temporarily stopped because of the rhythm of the work. The operator can check whether the robot is in the "running" state by looking at the robot motion state on the interface of the teacher.

1.8 Precautions for operator operation

Description of matters

- Operators must wear work clothes, safety helmets and safety shoes before working.
- Before entering the safety fence, please confirm that all safety measures are
- complete and functioning properly.
- When there are dust, oil and other dirt in the robot production area, please be sure to clean the environment in and around the working area of the robot first to avoid causing people to fall down or polluting the robot during maintenance.
- Only after cutting off the power supply, the operator can enter the action range of the robot to work.
- After running the robot for a period of time, all parts of the robot may heat up very high, especially the motor and reducer. Therefore, do not touch the above parts immediately after the robot stops.
- The operator should maintain the consciousness of escape at any time when working. It must be ensured that in an emergency, an immediate escape is possible.
- When switching on the power to the robot, make sure that there is no operator within the range of the robot.
- If the overhaul, repair, maintenance and other operations must be carried out in

the state of power, at this time, the operation should be carried out in groups of two people. One person should maintain a position where the emergency stop button can be pressed immediately, and the other person should remain alert and quickly operate within the range of the robot's action. In addition, the operation should be performed only after the retreat path is confirmed.

- Never enter the robot range of motion when the robot is in operation, even if the robot does not appear to be moving.
- Make sure that there are other robots around the robot and take care to avoid them.
- Before maintenance work, make sure there is enough space around the robot to avoid potential collision with peripheral equipment. Please keep the peripheral equipment in a fixed state to prevent the sudden movement of these equipment from causing danger.
- If the peripheral equipment has compressed air or water, cut off the supply and remove residual pressure before servicing.
- If it is necessary to replace parts, the replacement parts should only be those approved by Robot.
- Before removing the motor of any joint, please support and secure the robot arm with a suitable lifting device. After the motor is removed, the braking of the shaft will fail, and if it is not held in place by support, the shaft may fall down due to gravity, causing danger.
- After removing and replacing the motor, be sure to calibrate the zero before you can run the robot again.
- If you need to remove the motor, remove the motor after inserting the zero bolt and fixing the mechanical arm with a wooden block or crane to prevent it from falling off (the zero bolt and block are used to align with the original position, and can not be used to fix the machine). In addition, do not remove the motor

while the manipulator is supported by human hands.

- The balance spring device is in a compressed state inside under normal conditions, which is extremely dangerous. It is strictly forbidden to disassemble or decompose (only for models equipped with balance spring devices).
- Pay attention to the electrical components in the electric control cabinet. Even if it is in the state of power off, the energy retained in the device is still dangerous.
- Do not alter or modify the robot. We will not be responsible for any modification of the robot.

1.9 Safety features of robots

Description of Characteristics robots have the following safety characteristics, please be sure to fully understand and apply them to the design of safety measures.

- All emergency stop lines are hardware logic.
- The emergency stop button is installed on the teaching device and the electric control cabinet. If necessary, the emergency stop button can also be installed externally. Place these buttons where they are easily visible and easy to press during production operations.
- The motion speed and motion error of the robot will be monitored by the controller in real time. When the motion speed or position is out of tolerance, the robot will immediately stop and alarm the corresponding fault.
- For safety, the maximum speed of the robot will not exceed 250mm/s when the manual mode of the teaching device is used.
- The range of motion of each joint of the robot has been set at the factory. If necessary on site, the value of the soft limit of the joint can be modified, but it cannot exceed the limit of the mechanical limit.
- The teaching device provides user-defined use authority function, and the

operator can assign the corresponding authority to the field production personnel according to the field situation, in order to prevent the production personnel from performing some misoperations, resulting in personnel or property losses.

1.10 Common situation on site

Examples of situations

The majority of robot accidents are as follows:

- The robot performs automatic operation without confirming whether there is a person within the movement range of the robot.
- In the automatic operation state, a person enters the action range of the robot.
- The robot starts suddenly during the operation.
- They only pay attention to the robot in front of them and pay no attention to other robots.

The above accidents were caused by "neglecting safe operation steps" and "not expecting the robot to make sudden movements". In other words, they are all accidents caused by man-made unsafe behaviors such as "morning negligence" and "not following the prescribed steps". Therefore, operators must keep safety precautions in mind during their daily operations and must not be careless.

In addition, "emergencies" will also make the operator too late to implement "emergency stop", "escape" and other behaviors to avoid the accident, which is likely to lead to major accidents.

"Emergencies" generally have the following types:

- The low speed actions of the robot suddenly become high speed actions.
 - Other operators performed the operation.
 - Different procedures were started due to anomalies in peripheral equipment and program errors.
-

- Abnormal action due to noise, malfunction, defect, etc.
 - Misoperation.
 - A high speed action is executed when intended to be performed at a low speed.
 - The workpiece carried by the robot drops and scatters.
 - The workpiece is in the stop state of clamping and interlocking standby, and suddenly loses control.
 - An adjacent or behind robot executes the action.
-

**Action
countermeasures**

These are just a few examples. There are many forms of "contingencies." In most cases, it is not possible to "stop" or "run away" from a robot acting suddenly, so the following best countermeasures should be implemented to avoid such accidents:

- When the robot is not in use, measures such as "pressing the emergency stop button" and "cutting off the power" should be taken to make the robot unable to move.
- During robot action, configure safety personnel who can immediately press the emergency stop button to monitor the safety state.
- During the robot operation, the emergency stop button should be pressed immediately.
- During the action of the robot, the action of the robot should be paid attention to at all times, and the operation should not be carried out with the back of the robot. Even if the robot's action response is slow, it will lead to an accident.

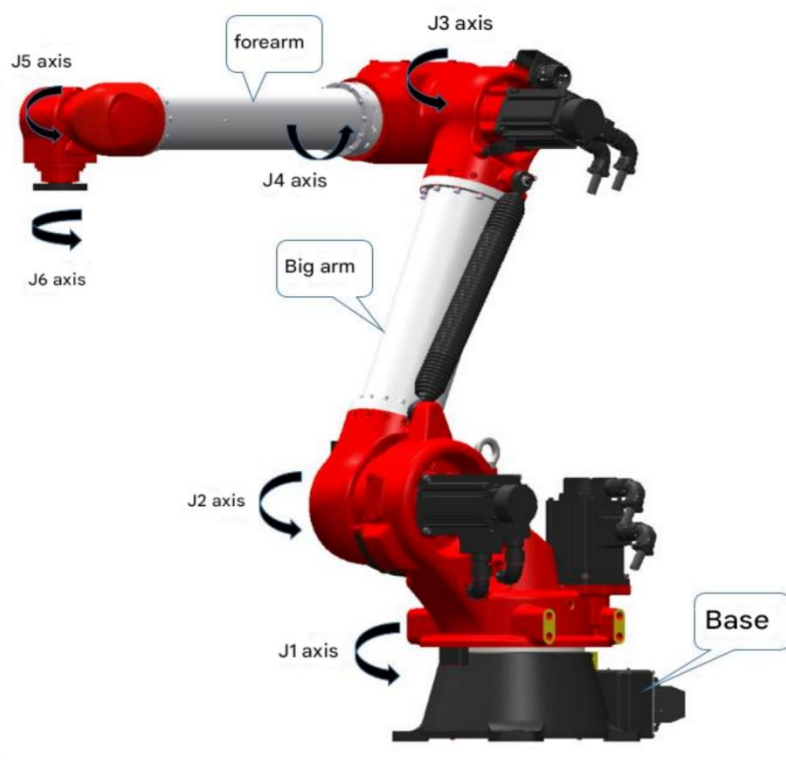
The above countermeasures must be fully understood and kept in mind by the operator and adhered to when working in the field.

2 System Hardware Composition

2.1 Ontology

Components The robot body is mainly composed of a base, a large arm, a small arm and each joint.

The body structure of a typical six-axis robot model is shown as follows:



2.2 Electric control system

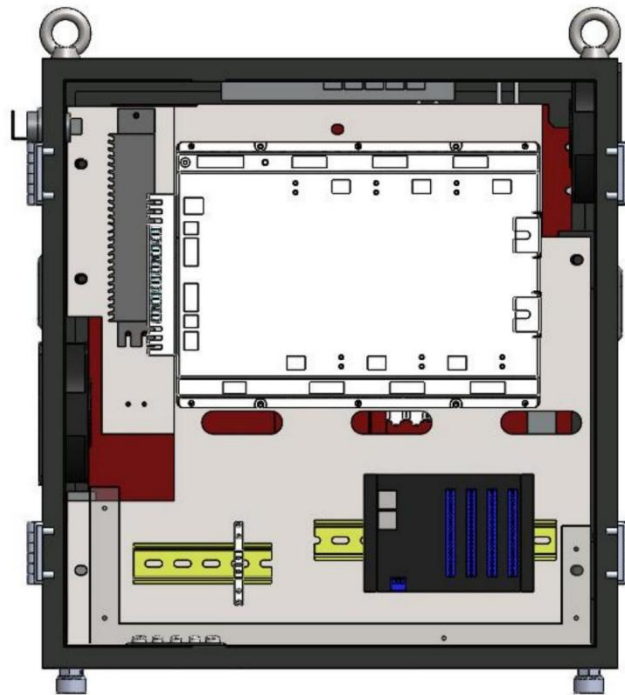
Core components The core components of the robot electronic control system mainly include: controller, servo driver, IO module, teaching device, power cable and coding cable, etc.

2.2.1 Electric control cabinet

Exterior in view design Electric control cabinet on the overall design adopts the design concept of intuitive, convenient, stable, in order to ensure the robot in use process is stable, reliable and safe operation, electric control cabinet adopts high hardness and durability of steel plate as the raw materials, with zinc plating, ensure that electric tank shell while meet the requirements of hardness, resistant to corrosion, wear resistance, stable and reliable protection.

The exterior and interior view of BR series general electric control cabinet are shown as follows:





Panel

The electric cabinet panel is shown below:

Description



Panel components from top to bottom are:

Emergency stop button: In case of emergency, tap this button to make the lock

hold the motor and cut off the servo signal.

- Power indicator: primary loop and secondary loop power indication.
- Alarm indicator: control system alarm indicator.
- Teaching switch: used to turn on or off the power of the teaching device.
- Power switch: used to turn on or off the power supply of the electric control cabinet..

2.2.2 The controller unit

Unit that IPC controller is equivalent to the human brain, all procedures and algorithm processing is completed in the IPC. IPC adopted an open, modular architecture, the embedded industrial computer as the platform, with a real-time Linux system, integrated with efficient robot motion control algorithm, provides the advanced fault diagnosis system. Benefit from the open fieldbus EtherCAT protocol, the system can support 64 largest axis, and support external axis function.

The controller is shown in the following figure:



2.2.3 The drive unit

Unit that The servo drive is used to control the servo motor, a controller is applied to the high precision positioning of transmission system. TV since the research of HSS - LDE - X6 is a fully functional, high performance servo drive, design and manufacture of innovative technology, with industry leading power density, with real-time Ethernet bus interface, using open fieldbus EtherCAT protocol, implementation and numerical

control device of high speed data exchange; With a high-resolution absolute encoder interface, it can be adapted to a variety of signal types of encoders.

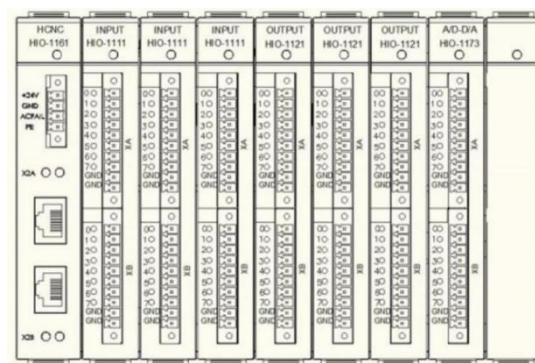
The HSS-LDE-X6 driver is shown in the following figure:



2.2.4 IO unit

Unit	TV itself IO unit using EtherCAT bus protocol, has the characteristics of high
Description	stability, high reliability. Products through strict anti-corrosion processing, has the input filter and power-fail protection function. IO extension module can be configured for any number of digital quantity input/output boards, supports both analog input and output.

IO equipment unit as shown in the figure below:



high-performance equipment. Unit 4, 6 axis motor inside the closed loop signal sampling input and output, three-phase drive axle has multiple IO and the communication interface, with sensors, PLC devices such as connectivity. Commonly used in 1 ~ 6 axis robot, small digital control equipment, automation equipment, etc.

Drive integration control unit as shown in the figure below:



2.3 Demonstrator

Instructions for Purpose

Teachers are handheld devices that manually move the robot, write programs, configure parameters, and check the status of the equipment. Teaching through the front-end ports and controller for data communication interaction. robot teaching device is usually referred to as "HSpad".

HSpad as shown in the figure below:

Hardware

Front instruction



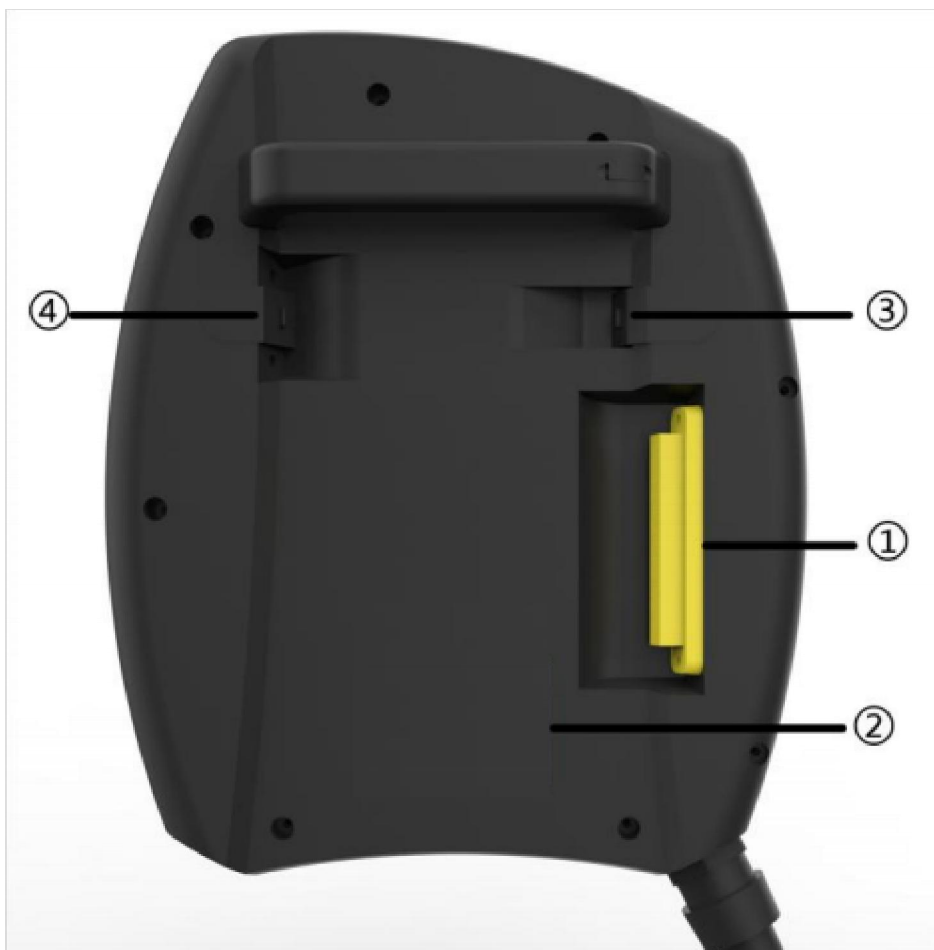
Label item	Instructions
1	Mode switch. Used to switch the robot operation mode. Use the steps: (1) clockwise twist switch, into the pattern set interface;

	(2) click the choose the required operating mode; (3) Turn the switch counterclockwise to exit the mode setting interface and complete the switching operation.
2	Emergency stop buttons. For fast to stop the robot's movement. Steps to use: (1) vertical shoot the button, complete stop operation; (2) clockwise unscrewed the button and stop button is reset; In column (3) confirmation of alarm information, complete reduction and stop the operation.
3	Press the Run button. Used to point and run the robot. This series of key six groups, each group were divided into positive and negative two direction key, in the corresponding robot corresponding axis of positive and negative direction. Use the steps: (1) Switch the operation mode of the robot to manual mode; (2) Press the safety switch on the back of the demonstrator to enable the robot; (3) presses the button, need some moving axis completion point move robot operation.
4	The manual rate adjustment buttons. Used to adjust the robot running speed automatically and external mode. Use the steps: (1) Switch the operation mode of the robot to automatic or external mode;

	(2) click the button, adjust speed operation.
5	Manual magnification adjustment button. Used to adjust the robot run in manual mode. Use the steps: (1) Switch the operation mode of the robot to manual mode; (2) Click the button to complete the speed adjustment operation.
6	Main menu button. Used to toggle the main menu and file navigator.
7	Pause the key. Used to pause a running program.
8	Stop pressing the button. To stop and uninstall program.
9	The back button. Used to roll back program lines. Steps to use: (1) Switch the operation mode of the robot to manual mode; (2) Press the safety switch on the back of the demonstrator to enable the robot; (3) Click the button to complete the operation of reverting the program line.
10	Run the keys. Used to run the loaded program.
11	Spare keys. Functions for customizing keys. The spare keys need to be configured before use.

Hardware

Instructions on the back



Label item	Instructions
1	Three-stage safety switch. Used in manual operation mode to make the robot can. The safety switch has three locations: (1) the initial position. (2) according to the first paragraph in the middle of the position. (3) Press to the full position of the second paragraph. When robot in manual mode, the user must press the safety switch and keep in the middle position, can let robot can keep the state exercise. Except for the middle position, which can be enabled, the other positions can be turned off. When the robot using automatic or external schema, safety switch

	doesn't work.
2	HSpad tag type.
3	The debug interface.
4	USB port. Often used to insert U disk for file copy, access and other related file operations.

- Hardware outlets instructions** Will teach on cable socket inserted into the electric control cabinet, can be in accordance with the following steps quickly inserted into the socket.
1. The first find teaching implement two lines logo on the cable socket, hold on both sides of the groove.



2. Check the electric control cabinet on the socket is with two lines.



3. If there are two vertical bar marks on the socket of the electric control cabinet, it can be directly aligned and inserted; If there are no two vertical bar marks on the socket of the electrical control cabinet, pinch the cable socket of the teaching device and insert it in reverse.

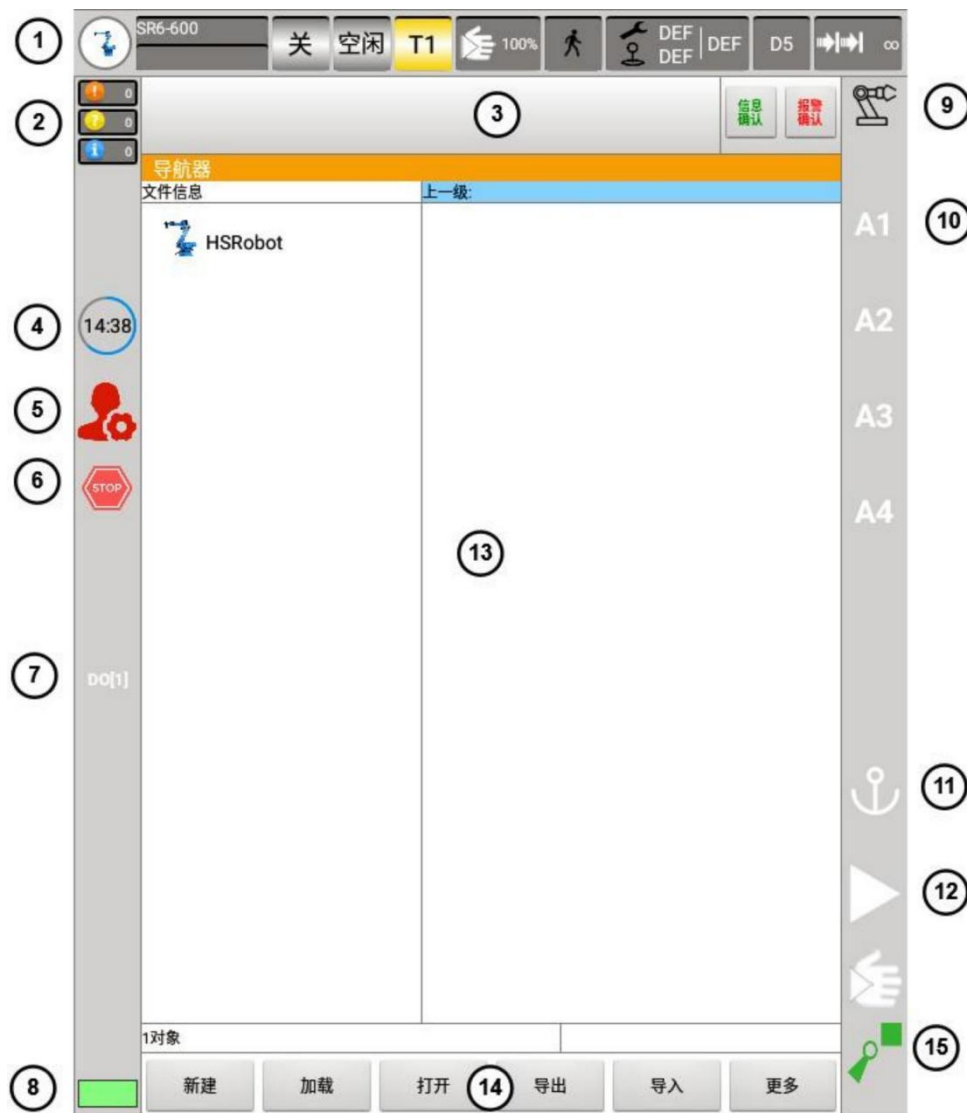
3 Common System Functions

3.1 Operation Interface

Main interface After the teaching device is turned on, the "HSC3-HSpad" software is opened by default.

Users can achieve the main control functions of the robot control system under this software.

The operation interface after the software is opened is shown in the figure below:



Tag entry	Instructions
1	The status bar. Used to display a series of current statuses of the robot. Clicking on the icons of various sub-function modules in the

	status bar allows for corresponding adjustments and settings.
2	Information Prompt Counter. Used to display how many messages of each type are waiting to be processed. Clicking on this icon will enlarge the message display interface.
3	Information Bar. Used to display information about the robot and the running program. When the teaching device receives multiple messages, the window defaults to only showing the last message prompt. Clicking on the window will display a list of messages. The list shows all pending messages. ● The "Details" button is used to display detailed information about the current message. ● The "Message Confirmation" button is used to acknowledge all messages except error messages. ● The "Alarm Confirmation" button is used to acknowledge all error messages.
4	Clock. Used to display the time.
5	User Quick Login Icon. Click the button to quickly jump to the user login interface. ● Blue indicates a production operator. ● Yellow indicates a debugging operator. ● Red indicates an administrator. ● Green indicates a custom user.
6	Robot Operation Status Indicator. Used solely for identification purposes.
7	Backup Key Indicator. Used solely for identification purposes.
8	Controller Network Status. ● The green display status indicates that the network connection is successful, the controller has been initialized, the teaching device is normally connected to the controller, and the robot can be controlled to move. ● The yellow display status indicates that the network connection is successful, but the controller has not been initialized, and the teaching device cannot control the robot to move.

	The red display status indicates that the network connection has failed, and the operator needs to check the network communication issue.
9	Coordinate System Type. Used to display the type of coordinate system currently used by the robot. Clicking on this icon allows you to switch the type of coordinate system.
10	Jogging Operation Indicator. Used solely for identification purposes. - If axis-related operation is selected, this series of icons will display the axis number (A1, A2, E1, E2, etc.). - If Cartesian operation is selected, this series of icons will display the coordinate system directions (X, Y, Z, A, B, C). Clicking on these icons will display the motion system group selection window. After selecting a group, the icon will change to display the corresponding icon for the selected group.
11	One-Click Home Return Icon. Used for quickly returning the robot to a safe point. Clicking on this icon allows the robot to return to a preset safe POINT according to the set home return program. For details on the configuration of the home return program, see the description in the section on the "One-Click Home Return Point Function".
12	Speed Multiplier Adjustment Indicator. - The triangle icon indicates non-manual mode. - The hand icon indicates manual mode. Used solely for identification purposes.
13	Main Functional Interface of the Software. Used to display the program interface, functional interfaces, etc.
14	Operation Menu Bar. Used for performing operations related to program files.
15	Collision Detection Prompt Indicator. Used to indicate whether the collision detection function is currently enabled in the system. - Gray indicates that collision detection is turned off. - Green indicates that collision detection is turned on. - Red indicates a collision alarm has occurred.

status bar

The status bar displays the current settings of the robot.

The status bar is shown in the figure below.



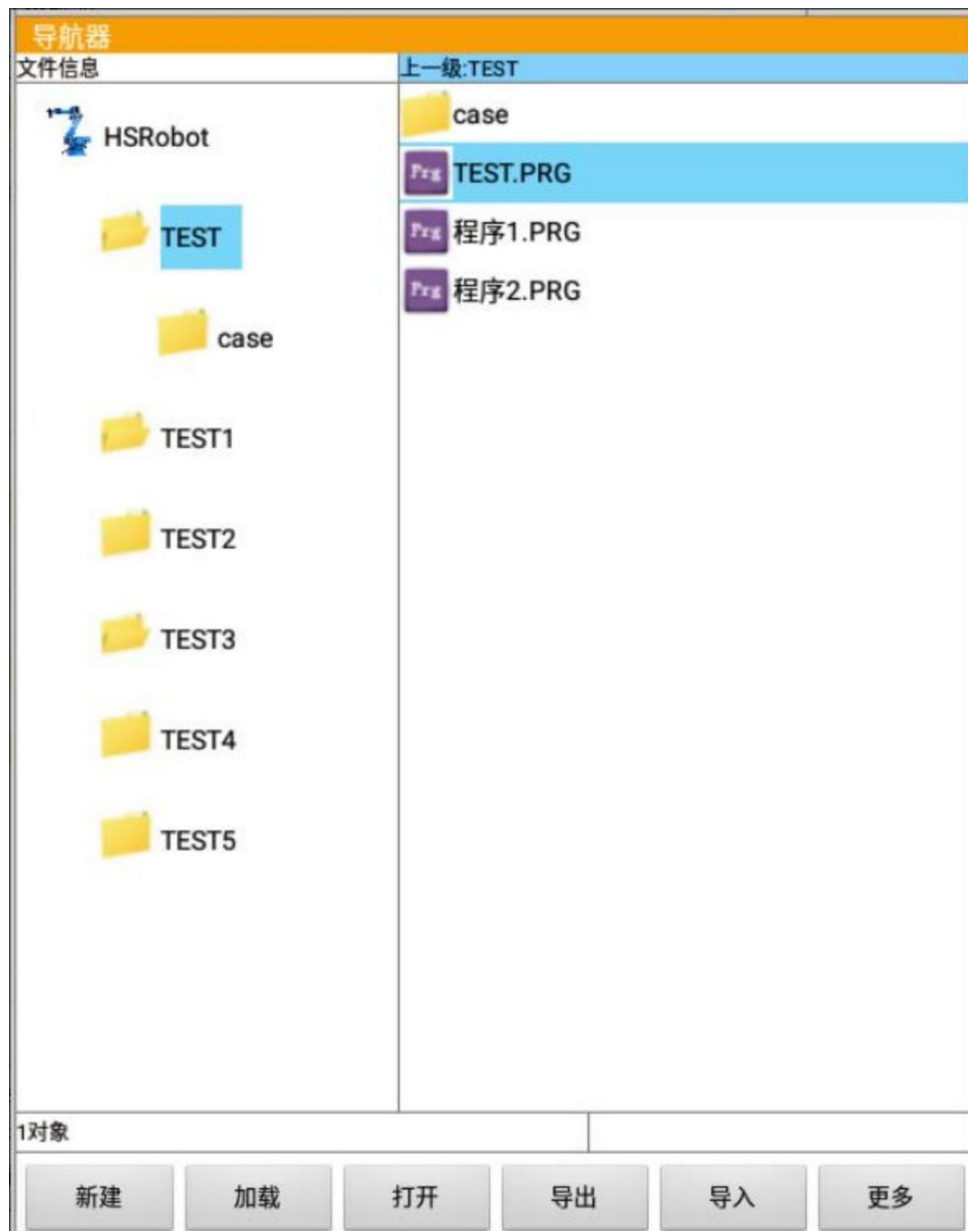
Tag entry	Instructions
1	Main Menu. Clicking on this icon allows you to switch between the main menu and the file navigator.
2	Machine Type and Program List Soft Key. This icon displays the current robot's machine type name and the loaded program information. Clicking on this icon opens the interface of the loaded program list, which shows the running program and the soft keys for program control. You can click on the program to enter the program interface or use the soft keys to perform corresponding operations on the running program.
3	Robot Enable Status. ● If the icon has a green background and displays the word "On," it means the robot is currently in the enabled state. ● If the icon has a gray background and displays the word "Off," it indicates the robot is currently in the disabled state. Clicking on this icon opens the enable settings window. In automatic mode, the robot can only be enabled or disabled through the "On" and "Off" buttons in this window. In manual and external modes, enabling or disabling the robot cannot be done through this window. The window displays the enable status and the safety switch status.
4	Robot and Program Operation Status. This icon displays the status of the program currently loaded in the system. If there is a hierarchical structure, the upper part indicates the status of the main program, and the lower part indicates the status of the background program. ● If the icon has a gray background and displays the word "Idle," it means no program is currently loaded. ● If the icon has a red background and displays the word "Ready," it means a program is loaded but not running. ● If the icon has a green background and displays the word "Running," it indicates a program is loaded and running normally. ● If the icon has a red background and displays the word

	"Error," it means a program is loaded but is in an abnormal state or the robot has encountered an exception. ● If the icon has a red background and displays the word "Pause," it indicates a program is loaded, running, but is currently paused. ● If the icon has a yellow background and displays the word "Wait," it means a program is loaded and is in an online editing state.
5	Operation Mode Status. This icon indicates the current operating mode of the robot. Manual T1 has a maximum TCP speed of 125 mm/s, and Manual T2 has a maximum TCP speed of 250 mm/s.
6	Robot Operating Speed Multiplier Status. This icon displays the current operating speed multiplier of the robot. This multiplier status can be set in two ways: (1) Clicking on this icon opens the speed multiplier adjustment window, where you can adjust the multiplier by 1% increments using the "+" and "-" buttons or by dragging the scroll bar left and right. (2) Through the manual speed multiplier adjustment buttons on the teaching device hardware.
7	Program Operation Mode Status. This icon shows the current method of program operation for the robot. In manual mode, clicking on this icon allows you to switch between "continuous" or "single-step" program operation.
8	Tool and Workpiece Number, Coordinated Group Number, Ground Rail Coordinated Jogging, IPO Mode Selection. This icon displays the current tool workpiece number, coordinated group number, ground rail coordinated jogging status, and IPO mode status of the robot. Clicking on this icon opens the settings window. Before loading the program, it shows the tool workpiece number selected by the operator. After successfully loading the program (without running), the system switches both the tool workpiece number and the coordinated group number back to the default values, thus displaying the default values (DEF). After running the program, if the program uses the tool workpiece coordinate system call instruction, it will display the tool workpiece coordinate number called in the program; if there is no call, it will

	remain at the default value. After unloading the program, the system will revert to the tool workpiece number called before loading.										
9	Robot Speed Grade or Load Status and Selection. This icon displays the current speed grade or dynamic load number selection of the robot. The system automatically determines the motion parameters enabled for the robot based on the model and displays them in the current status bar. When the icon shows the dynamic load number, clicking on the icon opens the selection settings window. Before loading the program, it shows the load number selected by the operator. After successfully loading the program (without running), the system switches the dynamic load number back to the default value, which is displayed as no load. After running the program, it will display the dynamic load number called in the program; if there is no call, it will remain as no load. After unloading the program, the system will revert to the dynamic load number called before loading.										
10	Robot Incremental Step Status. This icon displays the current moving distance of the robot when jogging by 1 unit. In manual mode, clicking on this icon opens the settings window for setting the jogging movement distance. The setting method is divided into persistent and non-persistent, with the non-persistent setting method supporting two ways: the scroll bar and fixed value buttons. The unit of this increment value changes according to the current coordinate system, being "." in joint coordinate system and "mm" in spatial coordinate system. Property Instructions: <table border="0"> <tr> <td>1 Continuous movement.</td><td>Close incremental manual</td></tr> <tr> <td>1 100mm / 10°</td><td>1 increment = 100mm or 10°.</td></tr> <tr> <td>1 10mm / 3°</td><td>1 increment = 10mm or 3°.</td></tr> <tr> <td>1 1 mm / 1°</td><td>1 increment = 1mm or 1°.</td></tr> <tr> <td>1 0.1mm / 0.005°</td><td>1 increment = 0.1mm or 0.005°.</td></tr> </table>	1 Continuous movement.	Close incremental manual	1 100mm / 10°	1 increment = 100mm or 10°.	1 10mm / 3°	1 increment = 10mm or 3°.	1 1 mm / 1°	1 increment = 1mm or 1°.	1 0.1mm / 0.005°	1 increment = 0.1mm or 0.005°.
1 Continuous movement.	Close incremental manual										
1 100mm / 10°	1 increment = 100mm or 10°.										
1 10mm / 3°	1 increment = 10mm or 3°.										
1 1 mm / 1°	1 increment = 1mm or 1°.										
1 0.1mm / 0.005°	1 increment = 0.1mm or 0.005°.										

Omniselector Navigator interface allows for the management of program files.

The navigator interface is shown in the figure below:



Navigator interface is generally divided into three areas:

- The file list on the left: Used to display the folder selected by the current operator.
- The directory list on the right: Used to display the program files in the folder selected by the current operator. The suffix for program files is ".PRG".
- The operation buttons at the bottom: Used for performing related operations on program files.

key	Instructions
new	Used for creating new programs and folders. There are two types of programs: ● Main Program: The group mask is "0", and motion

	instructions can be used. ● Background Program: The group mask is "*", and motion instructions cannot be used. When the operator creates a new program, the user can choose the type of program to be created.
load	Used to load the program. The master and background programs can be started or edited after being loaded. (1) Select the program in the navigator and press the load. (2) The program is displayed in the editor, and the selected program is loaded into the editor, always displaying with the corresponding open file and the running cursor. (3) In the editor, you can click "Start editing" to enter the online editing state of the program. The program can edit and input the response instructions only after loading. This feature supports online editing in manual mode. (4) Cancel loading program: click the "uninstall" button in the "Unload" program list or directly press the "Unload" button to unload the main program, but the background program cannot be uninstalled. If the program is running, the program must be stopped before canceling the program selection; not uninstalled under online editing, uninstall must switch to non-edited mode.
open	Used to open the program. After the program is opened, the program editing operation is available. After the switch interface is displayed, the button will be changed to the [Restore open] button, and click to open the open program. The program will be automatically saved immediately after the modification.
export	For export programs. When the operator exports the backup program, the file is exported to the export path set by the operator. This path can be set in the File [Program Data File Management] function module Path Configuration in the main menu.
import	For the import of the programs. When the operator imports the program, the system will find the program to restore under the import path set by the operator. This path can be set in the main menu for the file [Program Data File Management] function module Path Configuration.
More-Copy	Used for the replicate programs.

	You can copy the selected program.
More-Paste	For the paste program. You can paste the copied program.
More-delete	Delete is used to navigate through its selected file or folder.
More-Rename	For changes to the program name. You can change the name to the selected program.
More-Locking	For the locking program. The locked file cannot be renamed, deleted, or opened. The default lock password of the system is "hspad", which can be set in the function module of Configuration [Lock Password Settings] selected in the main menu.
More-multiple choices	Used to multiselect a list of programs.
More-full selection	For the full selection of the current program list.

Editor editor Program instructions can be edited in the interface.
Online editor interface is shown in the figure
below:

编辑器

P点号	类型	组号	UT	UF	CFG	值
1 跳转						

只读
TEST/程序1.PRGR
L1

更改
指令
备注
说明
自动滑屏
开始编辑
更多

The editor interface is largely divided into three areas:

- Top POINT information display area: displays information about the POINT variable P in the current program file.
- Middle instruction editing area: used to edit the current program instruction line.
- Operation button below: used for the program instructions.

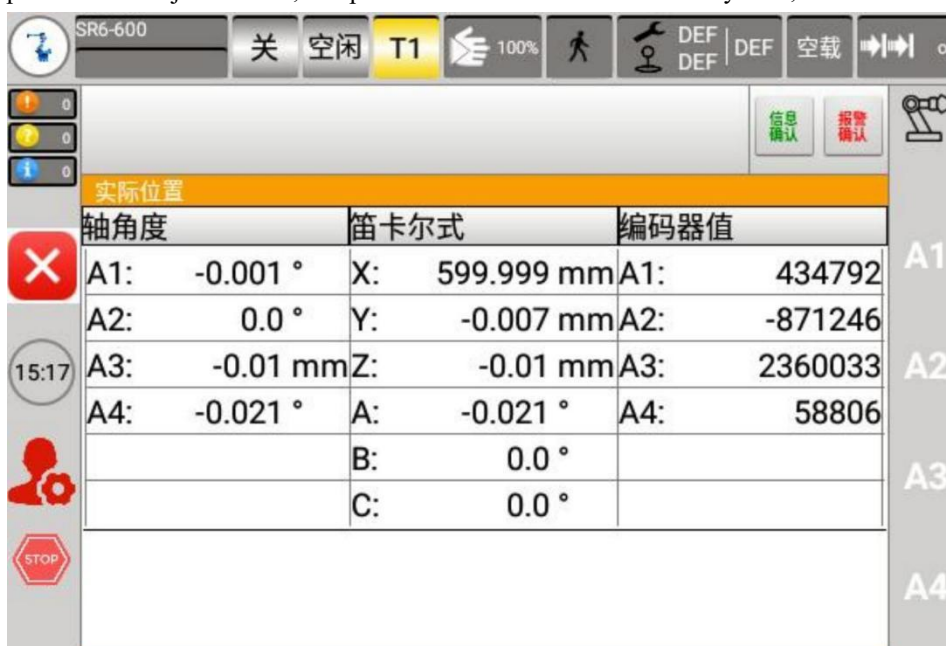
Operate the key	Instructions
change	Used to change the instructions. After selecting the instruction in the instruction edit area, click the button to enter the edit state of the instruction.
instruct	For adding new instructions. After selecting a line in the instruction edit area, click the button to select the new instruction. The new successful command is displayed below the current cursor highlight line.
comment	For the annotation instructions. After the instruction is selected in the instruction edit area, click the button to annotate the line instruction.

Instructions	Used for the additional annotation of the instructions, Instructions. After selecting a line in the instruction editing area, click the button to give the line a custom text Instructions.
Automatic slide screen / manual slide screen	Used to see if the program displays automatically. When the screen is selected, the screen shows the current line, and the user can slide the current program. This button is only available in the navigator interface by clicking the [Load] button.
Start editing	For the online editing programs. Click the button to enter the editing interface. After entering the editing interface, the button name is changed to [exit editing], and click to exit to the online editing interface. This button is only available in the navigator interface by clicking the [Load] button.
stop motion	For stopping the robot motion during a motion-to-point operation. This button is only available in the navigator interface by clicking the [Open] button.
More-macro references	For the program calling the macro instructions. Click this button to select the macro file within the system. The configuration macro file cannot be configured when the program is loaded and the program is opened.
More-revocation	To doing the previous step in the command edit area.
More-OpenInstructions	Note Instructions used to display the variable register. Click this button to make the instruction edit area display the variable register in the main menu- [display] - [variable list] function module Instructions.
More-This is in the On state	To display the input and output signal status. Click the button to enable the instruction editing area to display the status of the input and output signal in the program line in real time.
More-release wait	Waiting condition for relieving the WAIT instruction. When the program is in the waiting condition of the WAIT instruction, clicking the button enables the program to continue to

	the program without meeting the waiting condition of the WAIT instruction.
More-Hide the P-point list	To hide / display a list of P points. After the program loading enters the page, the P point list parameters are displayed by default. Click this button to hide the P point list.
More-navigation	For switching to the navigator interface.

3.2 View the current location

Instructions Users can view the position of the robot arm on the display interface, such as the position of the joint value, the position of the ortele coordinate system, etc.



- Axis Angle:
- Displays the current joint position values for each axis against the current model and current configuration.
- Cartesian:
- Displays the current position (X, Y, Z) and direction (A, B, C) of the robot TCP. If the external axis is currently configured, the joint position value of the external axis is displayed together here.
- The encoder value:
Displays the pulse value of the current shaft motor rotation.

During the robot operation, the actual position of each axis is updated in real time.

Steps of operation

Select the main menu to display the actual location.

3.3 IO situation display

Instructions

Reviewing the IO status of the system can obtain the status information of the current input and output IO of the system. The IO of the system has two real and virtual states, which can be switched through the "switch" button. For the real output IO and the virtual input-output IO, the value of the IO variable can be controlled through the "Value" button.



attribute	Instructions
IO number	Digital input / output IO number.
value	Input / output terminal values. If an input or output end is a TRUE, it is marked in red. Click value to switch value to TRUE or FALSE.
state	It indicates that the digital input / output terminal is the real IO or virtual IO, the real IO is displayed as REAL, and the virtual IO is displayed as VIRTUAL.
Instructions	Add a Instructions to the digital input / output end.
+100	Switch to the last 100 inputs or output ends in the display.
-100	Switch to the first 100 inputs or output ends in the display.

Toggle	Can switch between virtual and real input / output.
value	You can place the selected IO as either TRUE or FALSE.
Instructions	Add explanation Instructions to the digital input / output of the selected row, and click to change.
save	Save the IOInstructions.

Steps of operation

1. From the main menu, select the Display digital input / output terminal.
2. Click to select a specific input / output end and operate IO with the button on the right side of the interface.



warn:

The system will occupy the specific IO according to the number of entity IO. For details, please refer to the appendix section of the system signal occupancy table to view the IO signal that has been occupied by the system to avoid the occurrence of signal conflict.

3.4 AO situation display

Instructions

Reviewing the status of the system analog input / output (i. e. AO) can obtain the status information of the current input / output AO of the system, and the analog input / output system setting is set by the value of the voltage.



attribute	Instructions
order	Analog quantity of input / output serial number.
channel	Analog quantity: input / output AO number.
voltage	Analog quantity input / output terminal value.
Instructions	Add a Instructions to the input / output end of this analog quantity.
+100	Switch to the last 100 inputs or output ends in the display.
-100	Switch to the first 100 inputs or output ends in the display.

value	You can set the selected AO value. Range is from-24 to 24.
Instructions	Add explanation Instructions to the analog input / output of the selected row, and click to change.
save	Save the AOInstructions.

Steps of operation

1. From the main menu, select the Display analog input / output terminal.
2. Click select specific input / output and operate by setting parameter value AO.

3.5 Register variable

Instructions

Register variables are the global variables set by the system, with the following types:

- UT: the tool coordinate system register.
- UF: The base coordinate system register.
- R: Value register.
- SR: String register.
- JR: Joint-type coordinate register.
- LR: A Cartesian-type coordinate register.



View the register variable values Steps of operation

1. Select a list of variables in the main menu. A list of the register variables is displayed.
2. Clicking on the list of different registers displays the relevant variables.
3. You can change the page and save the register function.



Note:

After setting the register, click [save] to save. If not, it will be lost after power restart

3.6 Model information configuration

Robot information configuration instructions

The model information configuration function can configure the model and speed level parameters. Model information configuration has been set correctly before the factory, non-professionals do not modify without authorization. Re-switching the model will lose zero, limit and other parameters.



Step of operation

1. Under administrator permission, select Configure controller in the menu bar and enter the axis group configuration interface.
2. Click the "Modify" button.
3. Select the "Model" drop-down box option, and select the robot model.
4. Click the "OK" button.
5. Click the Modify the button. The robot can be given a switching speed level.
6. Select the Speed Level drop-down box option to select the speed level.
7. Click the OK " button.
8. Complete the setting after you restart the system.

warn:

After the model is successfully switched and the system is restarted, please recalibrate the zero point in time.

The robot enters the current robot nameplate number information through the presentation of the factory, and views the information of the current robot nameplate number in the version.



Steps of operation

1. Click the "modify" button.
2. Enter the factory number in the robot body.
3. Click the "OK" button to complete the setting.

3.7 Mechanical arm parameter information

Instructions

Robot parameters. You can view the detailed system parameters of the robotic arm shaft group.



Steps of operation

1. Select the information robot parameters in the menu bar and enter the robot parameters interface.
2. Click the option drop-down box option to display the details of the selection axis parameters.

3.8 Zero point calibration and recovery of zero point

Instructions

The robotic arm can only make the correct multiaxial linkage motion after the zero point calibration. If the robot moves directly without zero-point calibration, the robot cannot move normally according to the established trajectory expected by the operator. For zero calibration, it is necessary to move the joint axis of the robot to the mechanical zero POINT (that is, the zero scale position of each joint), input the initial position value, and click the [Calibration Zero] button.

Instructions



SR6-600 关 空闲 T1 100% DEF DEF DEF 空载

零点校准

轴数据校准:

轴	初始位置	补偿量
A1	0	0
A2	0	0
A3	0	0
A4	0	0

提示:
 1. 机器人出厂前已完成零点校准与标定。
 2. 切勿随意进行记录编码器与校准零点!
 3. 右侧示意图为机器人当前姿态零点位置{0,-90,180,0,0,0}

附加轴 恢复零点 校准零点



warn:

- The robot has carried out zero point calibration and rod length calibration compensation before leaving the factory, and the user does not need to calibrate under general circumstances.
- Different models of mechanical zero POINT position is not exactly the same, the initial position Angle value is not the same, if you want to calibrate can consult the relevant technical personnel.

Single axis calibration

In some cases, each joint axis of the robotic arm cannot be moved to zero due to space constraints, and only a single joint axis can be calibrated.

Steps of operation: Click the option in the "Initial Position" list of the joint axis to be calibrated, pop up the input box, enter the zero POINT data of the joint, and then click OK to complete the zero position calibration of the single axis.

Note that the [calibration zero] button is for calibrating all the axes of the robotic arm.

External axis calibration

The principle of external shaft calibration is the same. You only need to click the [external axis] button, switch to the external axis calibration interface, and then input the initial position value of zero point, then click the [Calibration zero] button.

How to recover the zero point quickly

1. Misconduct the [calibration zero] button:

Assuming that the user accidentally re-performed the zero point calibration, in fact, the system has saved the encoder value, directly click the [restore zero point] of the zero point interface.

2. accidental loss of zero data caused by switching models during system upgrade:

Get the zero-point data recorded at the factory from the manufacturer, and then re-import it.

3. Encoder is lost without power:

Each joint return card slot refers to the zero point, and then empty the multi-loop value of the encoder. Or each joint return slot refers to the zero, and then obtain the zero data from the factory and import it.

4. Situation of collider and jumping teeth:

The return slot of each joint refers to the zero calibration, and finally import the zero data recorded from the manufacturer.

3.9 Joint restriction

Instructions

By setting the soft limit of the joint axis of the robotic arm can be limited. The joint Angle has generally been set before the factory, and the user does not need to set it again.



warn:

- When setting the limit information, the value of the negative limit must be less than that of the positive limit.
- The robot must enable the limit switch and set the corresponding axis data, otherwise it may cause losses. The limit value is based on the zero POINT calculation of the robot, so the zero calibration is guaranteed correctly.

Steps of operation

1. Click in the menu bar to select the operational software limit switch.
2. Select a certain axis, edit the limit data in the table, and click the limit to enable [on / off] to open or close the enable limit.
3. After setting the soft limit information of all axes, click the [Save] button, and the limit setting will take effect immediately.



Note:

Click [external axis] to switch to the external axis for the external axis limit setting, the operation is the same as the internal axis.

3.10 Spatial area configuration

Instructions The space area function is a safety function that limits the space movement range of the robotic arm. Through the relevant configuration setting, the movement of the robotic arm cannot enter or exceed a certain space area. The spatial area configuration function requires setting the parameters and types of the area. After the configuration is completed and the area check function is enabled, the system will check and respond to the corresponding function.

3.10.1 Area configuration parameters

Instructions



区域配置

区域配置

类型: 无效区 ▼ 伸缩值: 0.0

处理模式: 不处理 ▼ ☐ 输出信号反转

区域原点 and 形状配置

基于UT[-1] UF[-1]	当前位置	工件号	默认 ▼	获取当前笛卡尔坐标
X: 71.603	A: 113.644	X	0.0 mm	
Y: 521.072	B: 0.0	Y	0.0 mm	
Z: -0.082	C: 0.0	Z	0.0 mm	

图示

形状: 圆柱形 ▼

半径: 0.0 mm

高: 0.0 mm

取消 确定

An Area (Area) refers to a spatial range. It has the following characteristics:

- (1) origin (Origin). Region relative to a reference point of a certain user coordinate system.
- (2) Shape (Shape). The region is usually based on a stereo of the origin and can be boxed and cylindrical.
- (3) Expansion value (Offset), to judge the allowable error when inside / outside the region. The expansion value can also be understood as the wall thickness of the area, the safety area is equivalent to increasing the wall thickness in, and the interference /

sharing area is equivalent to increasing out. In practical application, the expansion value should be larger than the theoretical value to increase the safety.

(4) Selection of workpieces, and determine the XYZ direction of the area according to the calibration selection of workpieces.



Note:

- The system cannot accurately stop at the edge of the expansion value, and the stop distance varies under the influence of the motion position and speed.

3.10.2 Regional Types and Response Methods

Instructions

The system sets up four region types, each of which has its own specific function.

Instructions for Regional Configuration Signal



序号	类型	共享区使能	区域输出
0	无效区	/	○
1	无效区	/	○
2	无效区	/	○
3	无效区	/	○
4	无效区	/	○
5	无效区	/	○
6	无效区	/	○
7	无效区	/	○

The system has 16 spatial areas, and an external output signal is configured for each area to indicating whether the robot position is inside / outside the area and whether the external output signal can be reversed.

- Signal reversal =true (open state), out of region output = ON.
- Signal reversal =false (closed state), into the region, area output = ON, outside the region is otherwise.

Steps of Operation

1. Select Configure Controller Configuration in the menu bar to enter the area configuration interface.
2. To select the configuration list, click the [Modify] button to enter the detailed configuration interface.

3. After the configuration is completed, select the Configuration controller configuration in the menu bar, and enter the input configuration and output configuration interface to configure the signals of the area configuration items.

Processing mode, Instructions

The processing mode is the response of the system according to the relationship between the actual position of the robot and the position of the set area (inside / outside). The processing mode includes:

- Don't deal with (IGNORE) .
- Report the wrong shutdown (ERROR STOP) .
- Warning shutdown (WARNING STOP) .

interference (Block Area

It means areas where the robot can not enter or stay.

Features: Under manual mode (T1, T2) point motion, inch motion, spatial motion / automatic mode (AUTO, external), the robot stops in the area. If the robot is already in the area, the area type needs to be switched to the invalid area, and then the robot moves outside the area before the robot can carry out subsequent movements.

Safe area (Work Area)

It is the area where the robot is not allowed to leave.

Features: Under the point motion, inch motion, spatial motion / automatic mode (AUTO, external) in manual mode (T1, T2), the robot is normal in the area and stops wrong outside the area. If the robot is already outside the area, the area type needs to be switched to the invalid area, and then the robot moves to the area, and the robot can carry out subsequent movements.

share (Share Area)

It is the area where the robot is not allowed to leave.

Features: Under the point motion, inch motion, spatial motion / automatic mode (AUTO, external) in manual mode (T1, T2), the robot is normal in the area and stops wrong outside the area. If the robot is already outside the area, the area type needs to be switched to the invalid area, and then the robot moves to the area, and the robot can carry out subsequent movements..

- In the enabling state, the robot position is normal outside the zone; follow the configured processing mode.
- In the non-enabled state, the shared area has no effect.

Features: Under the point motion, inch motion, spatial motion / automatic mode (AUTO, external) in manual mode (T1, T2), the robot is normal outside the area, and the error is stopped in the area. If the robot has issued an alarm in the area, it is

necessary to switch the area type to an invalid area or set the processing mode to "no processing", and then move the robot outside the area before the robot can carry out subsequent movement.

Invalid area

It is the failure of the current set area, and only the output signal detection of the area, without any error reporting and shutdown of the area.



Example:

If the robot is in a position and a region is to be generated at this position, the XYZ coordinates are {600, -100, 100}. If an interference zone is generated at the current position of the robot, the length is 200, the width is 100, the height is 50, and the scaling value is 5, then a box area generated at this position is (default in the positive direction, if the work coordinate is called, the direction of XYZ is determined).

X=[595,805]

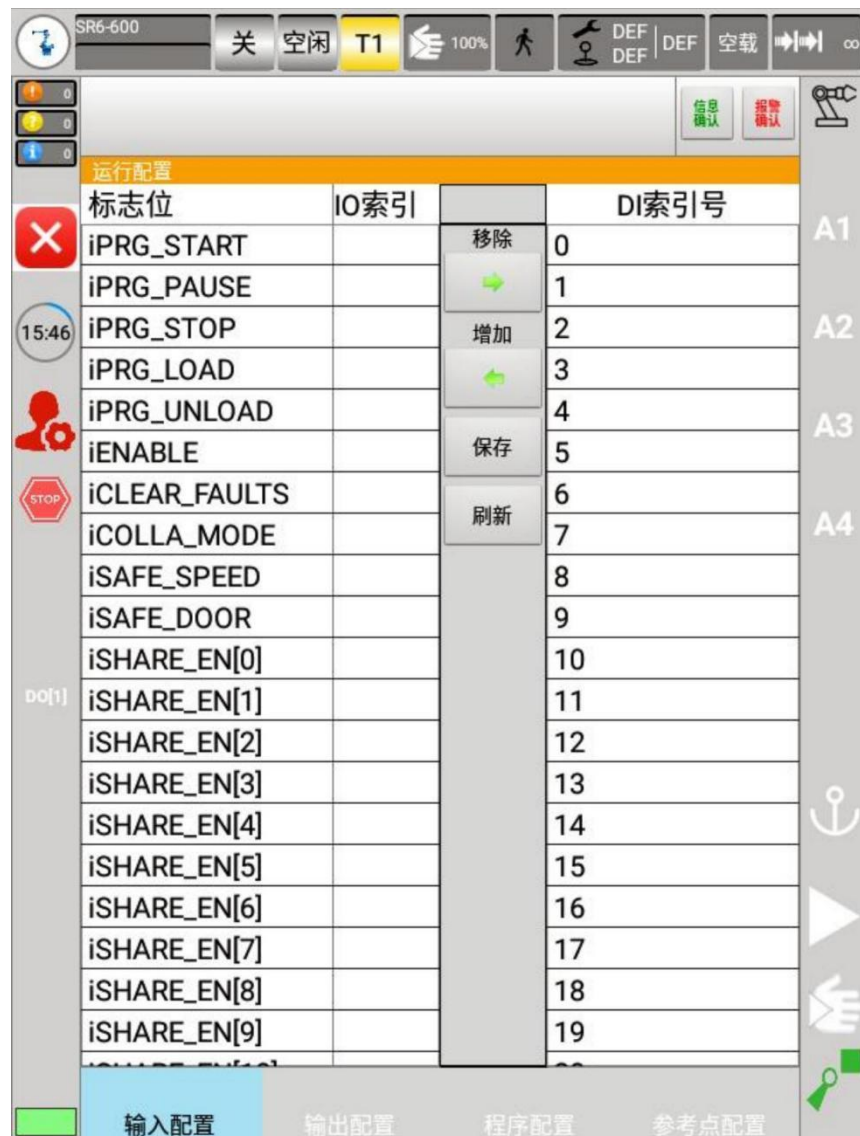
Y=[-105,5]

Z=[95,155]



Note:

- The three coordinate values of XYZ of the manipulator arm are not calculated to enter the interference zone / shared zone in the interference zone / sharing zone, and any of the three Z coordinates of the manipulator arm exceeds the value of the safe zone.
- When modifying the zone configuration, the drive enabling needs to be dropped before modification. The setting is invalid in the drive enabling state.



- There are 16 spatial regions set by the system, and an external output signal is configured for each region, see oAREA_OUT in Appendix 11.6 "List of External Signals", to use the functional area.



Each Shared area to configure an external input signal, see appendix

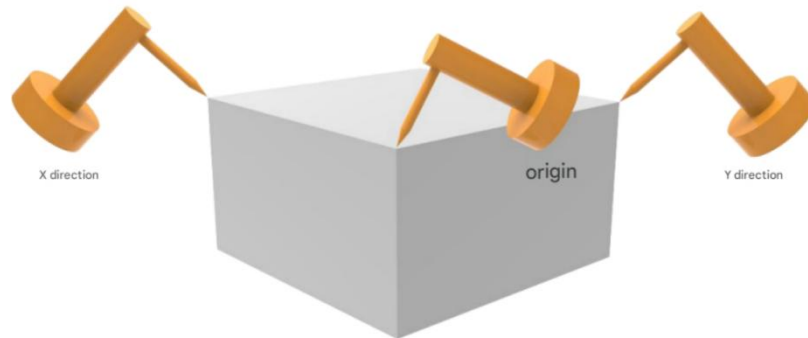
"iSHARED_EN" in "external signal list", for the use of the Shared area.

3.11 Calibration of the tool-workpiece coordinate system

Instructions Tool workpiece coordinate system calibration function is through the mobile manipulator mark certain point after the operation, the system based on the input point coordinate calculation. The coordinate system data obtained after calculation is saved in the UT or UF register.

3.11.1 Calibration of workpiece coordinate system using the three-point method

Instructions By specifying the origin, X direction and Y direction of the workpiece coordinate system, the workpiece coordinate system is calibrated.



Steps of operation

- 1 Select → Put into Operation → User workpiece Calibration from the menu.
2. Select the user workpiece number to be calibrated, and set the user workpiece name, which is the annotation of UF register.
3. Click on "start calibration" button.
4. Move to the origin of the base coordinate and click [Origin] to get the coordinates and record the origin coordinates.
5. Move to a point in the X direction of the calibration base coordinate, click [X direction] to get the coordinates and record the coordinates.
6. Move to a point in the Y direction of the calibration base coordinate, click the "Y

direction" to obtain the coordinate record coordinate.

7. Click on the "calibration" button, determine the program to calculate the calibration coordinate values.

8. Click the "Save" button to store the calibration value of the base coordinates.

9. Switch to the user coordinate system, select the calibration of artifacts, XYZ direction, according to the calibration of the direction.



Tips:

- Before calibration, the current working coordinate system value should be switched to the default value DEF, and then the manipulator should be moved for calibration.

3.11.2 Calibration of tool coordinate system

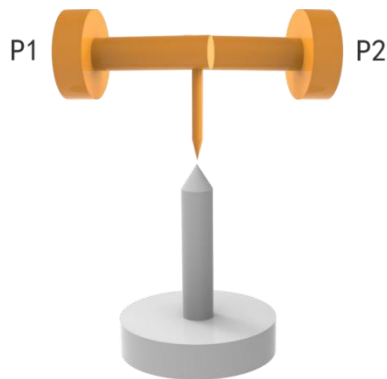
Instructions

- Tool coordinate system calibration is divided into four-axis robot tool coordinate system calibration and six-axis robot tool coordinate system calibration.
- Among them, four axis robot calibration tool coordinate system support 2 points, 4 points calibration, 20 points calibration; Six axis robot calibration tool coordinate system to support 4 points, six points calibration, 20 points calibration.

3.11.2.1 Four axis robot 2 point calibration tool coordinate system

Instructions

Will be the center of the robot end clamping tools from two different directions to move to a fixed reference point and to record the location of the mechanical arm. The system will calculate the TCP of the tool based on the two point position information recorded.



Steps of operation

1. Choose "Run" from the menu, then select "User Tool Calibration".
2. Select the user tool number to be calibrated and specify the user tool name if necessary.
3. Click on the "Start Calibration" button.
4. Move to reference point 1, click on "Reference Point 1", and record the coordinates.
5. Move to reference point 2, click on "Reference Point 2", and record the coordinates.
6. Click on the "Calibration" button to confirm that the program has calculated the calibration coordinates.
7. Click on the "Save" button to store the calibrated tool coordinate values.

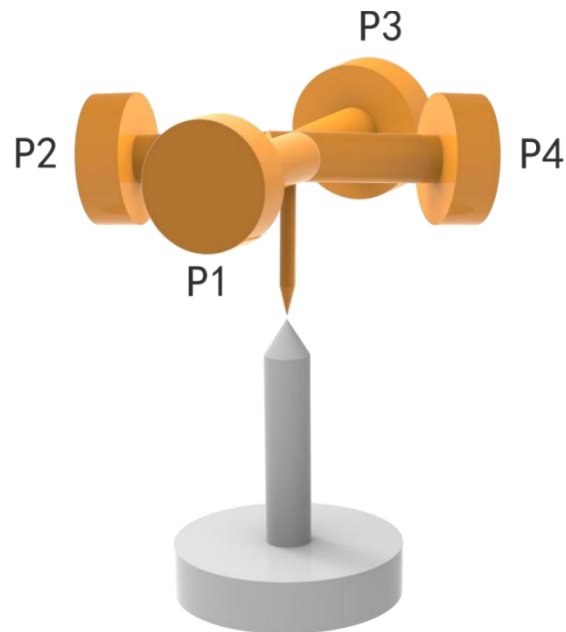
3.11.2.2 4-point calibration of quaxial robot tool coordinate system

Instructions

Move the center point of the tool held at the end of the robot from 4 different directions to a fixed reference point and record the current manipulator position. System will be according to the recorded information to calculate the four point tool of TCP.

The 4-point method calibrates the position, which indicates the offset distance between the fixture coordinate system and the center point of the flange. Therefore, in the marked tool coordinate system UT[x], only XYZ usually has a numerical value,

while ABC is 0.



Steps of operation

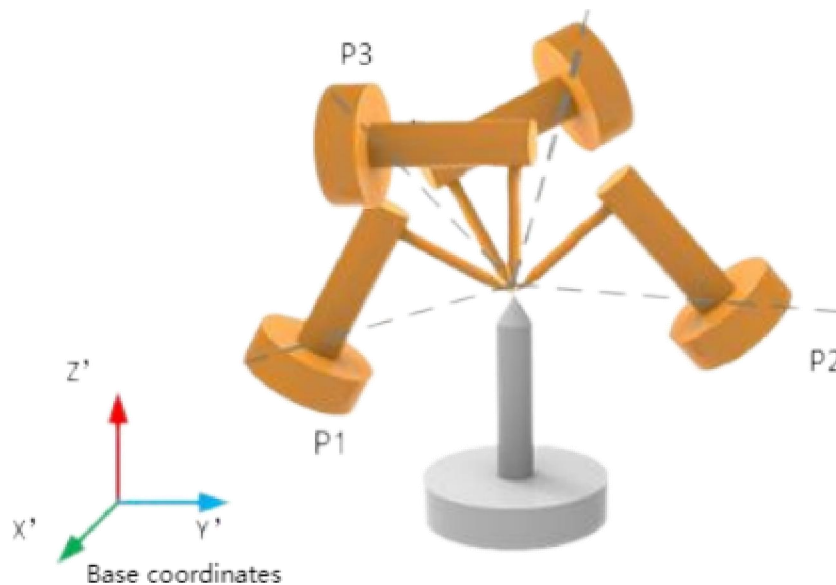
4 points operation flow and 2 points are similar, may refer to the four axis robot 2 point calibration tool coordinate system of the steps.

3.11.2.3 Calibration of six-axis robot tool coordinate system with 4-point method

Instructions

Move the center point of the tool held at the end of the robot from 4 different directions to a fixed reference point and record the current manipulator position. System will be according to the recorded information to calculate the four point tool of TCP.

The 4-point method calibrates the position, which indicates the offset distance between the fixture coordinate system and the center point of the flange. Therefore, in the marked tool coordinate system $UT[x]$, only XYZ usually has a numerical value, while ABC is 0.



Steps of operation

1. Select Put into Operation → User Tool Calibration from the menu.
2. User choose to calibration tool, can be set up user names.
3. Click the "Start Calibration" button.
4. Move to reference point 1 and click [Reference Point 1] to get the coordinates and record the coordinates.
5. Move to reference point 2 and click [Reference Point 2] to get the coordinate record coordinates.
6. Move to the reference point 3, click on the "reference point 3", access to coordinate record coordinate.
7. Move to the reference point 4, click on the "reference point 4", access to coordinate record coordinate.
8. Click the "Calibration" button to make sure the program calculates the calibration coordinates.
9. Click the "Save" button to store the calibration value of the tool coordinates.
10. The switch to the on the types of the coordinate system of the tool coordinate system, select the calibration tool, ABC direction, robot end tool TCP will around the workpiece rotation.

3.11.2.4 Six-axis robot tool coordinate system 6-point method calibration

Instructions

The 6-point method calibrates the tool coordinate system Similar to the 4-point method, the 6-point method can be used to calibrate the attitude of the tool coordinate system. Six point method first four reference points for calibration, the fifth and sixth,

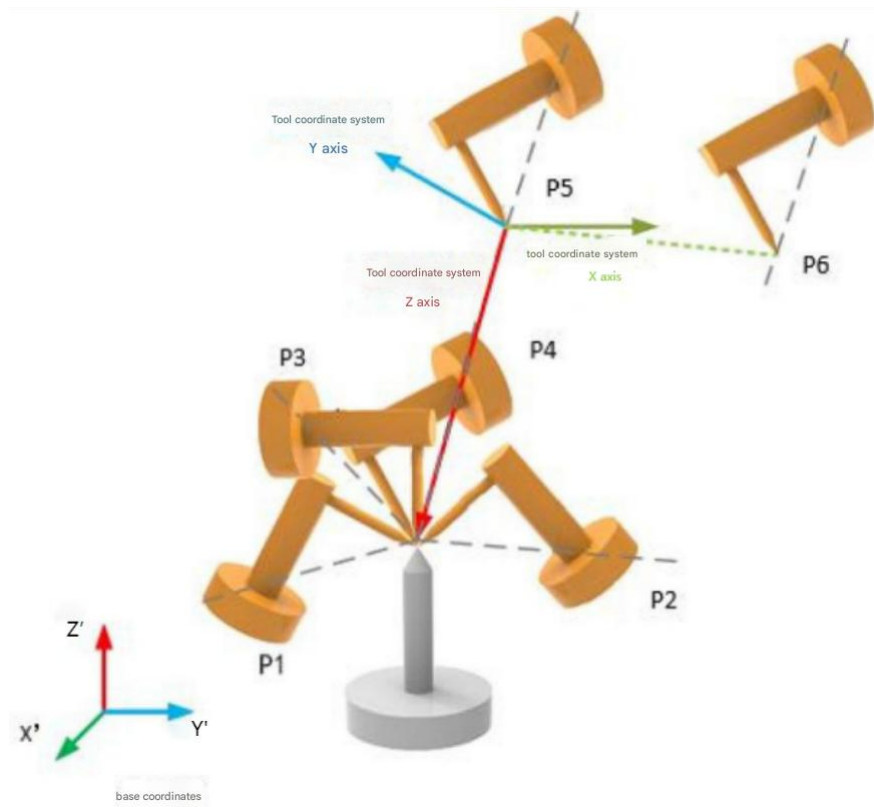
respectively the points on the tool is used to record the z axis and the points on the zx plane.

The essence of attitude calibration operation: In general, the user wants to Z axis calibration tool coordinate system for the mechanical arm in the direction of the flange center with the direction of the center of the clamp connection (below called the fixture axis direction), which, when use tools to coordinate the Z direction of the movement is moving along the fixture axis, so the calibration for the fifth point, Make sure that the invisible fixture axis (gray dashed line in the following figure) is collinear with the vector P5P4.

Can be added in the actual calibration reference determines the location of the invisible fixture axis, can also by properly selecting the fourth axis attitude to determine the direction of the points, 4 points, such as calibration fixture axis and basic coordinate system Z 'axle load, the calibration point 5, only need to select the points on the base coordinate system Z axis' (that is, only when the teaching points according to Z axis button).

Calibration points 6, must choose ZX plane of the above points, the system through the vector product $P5P4 \times P5P6$, calculate the Y axis tool coordinate system, finally calculate the X axis.

Note: When calibrating the fifth and sixth points, the posture of the manipulator must be consistent with the fourth point, that is, the posture remains unchanged.



Steps of operation

The operation process of the 6-point method is similar to the 4-point method, and the operation steps in the "Calibration of the 6-axis robot tool Coordinate System with 4-point Method" can be referred to.

3.11.2.5 Fixture mounting Angle A compensation

Method Notes Usually when installing a jig and fixture one edge and tool coordinate system X | Y axis is installation Angle tutu under (1), if the end of the tool coordinate system X axis and asked to clamp the edge of overlap tutu under (2), but there are two ways.

- Method 1: calculated by the fixture design personnel Delta_A are given, then add / minus the tool coordinate system of the original value, A new A value.
- Method 2: Calculated by reference.

First call calibration tool coordinate system, the end of the mechanical arm shaft back to zero, and the mechanical arm moves to A blank sheet at the end of the clamp on the desktop, and then draw A line along the current edge position on the blank paper,

record the mechanical arm current Descartes location value P0 (X0, Y0, Z0, A0, B0, C0), and then repeated "adjust the Angle of A value, Mobile X "operation until the mechanical arm tool TCP point X axis moving parallel to the direction of the straight line and blank paper direction, the record of mechanical arm Descartes location value P1 (X1, Y1, Z1, A1, B1, C1), $\Delta A = A1 - A0$.

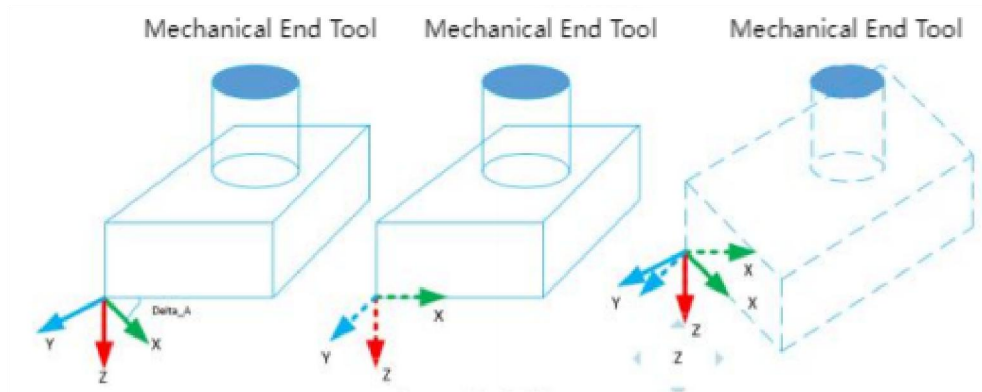
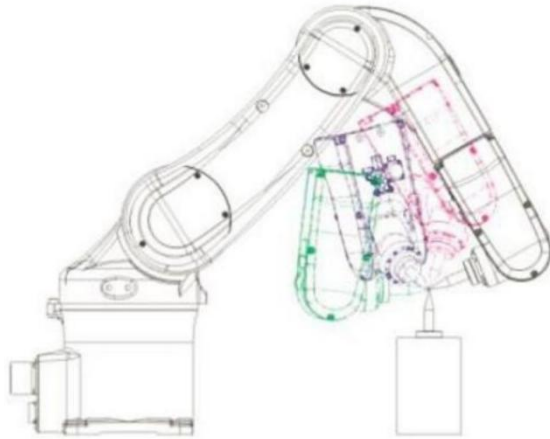


Figure 1: The tool coordinate system axis performs a rotation offset at the end of the tool by an angle ΔA .
Figure 2: Calibration – The tool coordinate system axis is aligned with the end of the tool.
Figure 3: ΔA is calculated through parameter objects and the standard rotation axis.

3.11.2.6 20-point calibration tool coordinate system

Function In TCP requiring high precision tools, can use 20 points calibration. 20 points
Description calibration logo set a high precision tool coordinate system, but because of the need to maintain high accuracy of TCP value, after calibration will modify existing at the same time the zero value of the joints.

The 20-point calibration is often used in situations where a collision changes the zero point and then reduces the TCP accuracy of the tool. The 20-point calibration can not only calibrate the TCP of the high-precision tool, but also correct the original joint zero value.

**Warning:**

The zero point will irreversibly change after calibration. Advice before formal for calibration, first select menu of "calibration file parameter management" for export calibration parameters operation. In other cases, do not operate this function.

Steps of operation



1. Select → Put into Operation → 20-point calibration from the menu.
2. Select the need calibration tool.
3. Approach the reference point by moving the robot in different poses (as different as possible for each pose) and get the point position information.
4. After obtaining more than 10 points, click the "Calibration" button, and the system will calculate the zero offset and the tool coordinate system value.
5. Click the [Save] button to save the relevant information.



Tip:

1. 20 point calibration, one 20 points just estimates. Usually only 10 points are needed to complete the calibration, but the more points are obtained, the more accurate the calibration result will be.
2. 20 points calibration will be to make the zero offset joints may cause point actual location of the program change.

- The 20-point calibration marks out a more accurate tool coordinate system by modifying the zero values of each axis of the system in order to ensure high accuracy.

3.12 System variables

Instructions Mechanical arm in the presence of related variables of the need to adjust the system operation process, the overall trajectory parameters adjustment, the current teaching is no the interface to manipulate, the need to adjust the variable Settings, etc., thus put forward "system variables" set interface, provides to the user to modify, Settings, and so on.



Vibration suppression function

The robot arm has the problems of jitter and unsteady trajectory speed. In order to solve the phenomenon, can be in after the completion of the trajectory planning, at the command output filter to commands (position and velocity), make its speed stability is achieved, but this may change order filtering track features, reduce the tracking accuracy, therefore need to be whether the parameter can be set to filter, filtering level, the location of the maximum deviation.

- Filter frequency setting: According to the different natural frequencies of different models, the filter frequency characteristics can be set, ranging from 0.00 to 10.00Hz, which can be accurately input to two decimal places.

- Filter Level Setting: Different levels correspond to different filtering effects. A higher value indicates a stronger filtering effect and reduced jitter (current maximum is 7).

Point flexibility

There is jitter when the robot moves with large loads. The compliance level is increased, and the user is supported to adjust the compliance of the point movement on the interface. If the compliance is too soft, it may lead to slow start and stop (long time to move to the point), and if the compliance is too small, it may shake. Users can set it in the interface according to their actual requirements.

**Dynamics
planning function**

Dynamic planning information consult with dynamic planning switch function, allows users to set switch parameters of dynamic adaptive planning.

**SCARA Angle
value**

The representation of system change the Angle of A SCARA type. The Angle A of the default SCARA models is changed to allow an excess of ± 180 degrees. The function interface can check whether the function is enabled or not.

- The value of Angle A is allowed to exceed $\pm 180^\circ$ when the current value of Angle A is displayed on the teacher
- Teaching record space point position, A value of the Angle $+ - 180^\circ$ is allowed
- Perform J LR [x], L LR [x] instruction, not specified by the shortest path and TURN attribute, but the movement to the current value of A horn or TURN TALARM and attribute failure, movement directly to the absolute value of A corner.

**Axis group
over wrist
singular
function**

Switch the default open axis group had wrist singularity function, mechanical arm cartesian point or linear motion to point near the wrist singularity, mechanical arm speed remains the same, 4, 6 axis does not flip through the axis can be the location of the $[4] = 0$.

**Joint flexible
compensation
function**

Flexible compensation is based on the implementation of the coupled dynamics, it overcomes the robot joint deformation under different load, the influence of open

flexible compensation can improve the robot absolute positioning accuracy and precision of trajectory.

Currently supported models have JR607 and JR6210.

Steps of operation

- 1 Click on the menu bar and select → Configuration → System Variables.
 2. According to the application requirements in terms of Settings, switch Settings take effect immediately, modify Settings click click save take effect immediately.
-

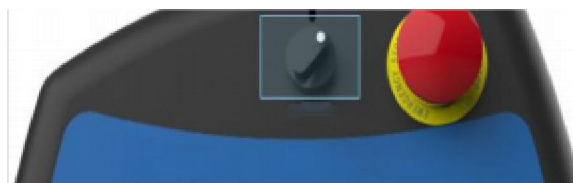
**Warnings:**

- 1 The user sets the parameters according to the application needs to avoid irreversible situations.

4 Operation

4.1 Operation mode Introduction

- Instructions** The system supports manual T1, manual T2, automatic run and external control modes. Mode switching can be done through the mode switching button on the top of the demonstrator.
- Manual T1 mode and manual T2 mode differ primarily in the speed limitations. The maximum TCP speed for T1 is 125mm/s, while for T2 it is 250mm/s. In manual mode, the teach pendant's physical buttons can be used to move the robotic arm for teaching purposes and to load PRG programs for debugging. During manual operation, the servo drive must be enabled by holding down the three-segment switch on the back of the teach pendant.
 - Automatic mode: In this mode, movement of the robotic arm using physical buttons on the teach pendant is not possible; only loading PRG programs can move the robot. At its highest speed setting, automatic mode operates at maximum motor speeds achievable. Servo drive enabling in automatic mode is done by clicking on the "Enable Status" icon located at top of teach pendant interface.
 - External control mode: This involves controlling robotic arm operations through IO signals. In external control mode, moving or running programs via teach pendant panel is not possible. Detailed instructions and configuration operations for external control signals are provided in Section 9.3 "External Control Signal Configuration," as well as Appendix 11.6 "Overview of External Control Signals."





4.2 Coordinate system type selection

Instructions You can specify the type of coordinate system you are currently using when running manually. Types are: the system of coordinate system axis coordinate system and world coordinate system, the base coordinate system, the user coordinate system (workpiece coordinate system), the tool coordinate system. Can click on the "coordinate system status" icon on the right side of the interface, to coordinate system of the switch.

- When using the axis coordinate system, the entity key on the right side of the teaching device corresponds to the J1-J6 axis of the mechanical arm. Press the positive and negative buttons of the corresponding axis, and the axis will move to the positive and negative direction.
- When using the world coordinate system, the entity key on the right side of the teacher corresponds to the direction of XYZABC. Press the positive and negative buttons in the corresponding direction, and the manipulator will move to the corresponding positive and negative directions.
- When using the base coordinate system, the entity key on the right side of the teacher corresponds to the direction of XYZABC. Press the positive and negative buttons in the corresponding direction, and the manipulator will move to the corresponding positive and negative directions.
- Using user coordinate system (workpiece coordinate system), the robot arm will move in the direction of the workpiece coordinate system currently used.
- When using the tool coordinate system, the mechanical arm along the current use of tool coordinate system specified in the direction of the movement.

4.3 Tool workpiece coordinate system selection

Instructions You can switch the current tool workpiece coordinate system, collaborative group, collaborative load number, and IPO mode on the operation interface.

using the default tool workpiece coordinate system, cooperative group, after the uninstall program, to load before calling tool, workpiece, collaborative group number.

Steps of operation

- 1 Touch the tool and workpiece coordinates status icon to open the "Activated Tool/workpiece Coordinates" window.
2. Select the desired tool and the workpiece coordinate system.

Instructions for IPO mode selection

The IPO mode includes two modes: flange mode and external tool mode. Flange mode refers to the handheld tool mode, where selecting the "tool coordinate system" for point-to-point movement will cause rotation around the TCP point at the end of the flange when operating ABC angles. External tool mode involves holding the workpiece, with the robot gripping it. When selecting the "tool coordinate system" for point-to-point movement, ABC angles will rotate around a pre-calibrated TCP point fixed on the ground.

Collaborative group selection

After setting the configuration parameters of the collaborative group by the robot controller, the external axis parameters are calibrated. Choose the corresponding coordination group can realize external control axis motion robot coordinated movement.

Load number selection

See the Load Configuration Function section for details.

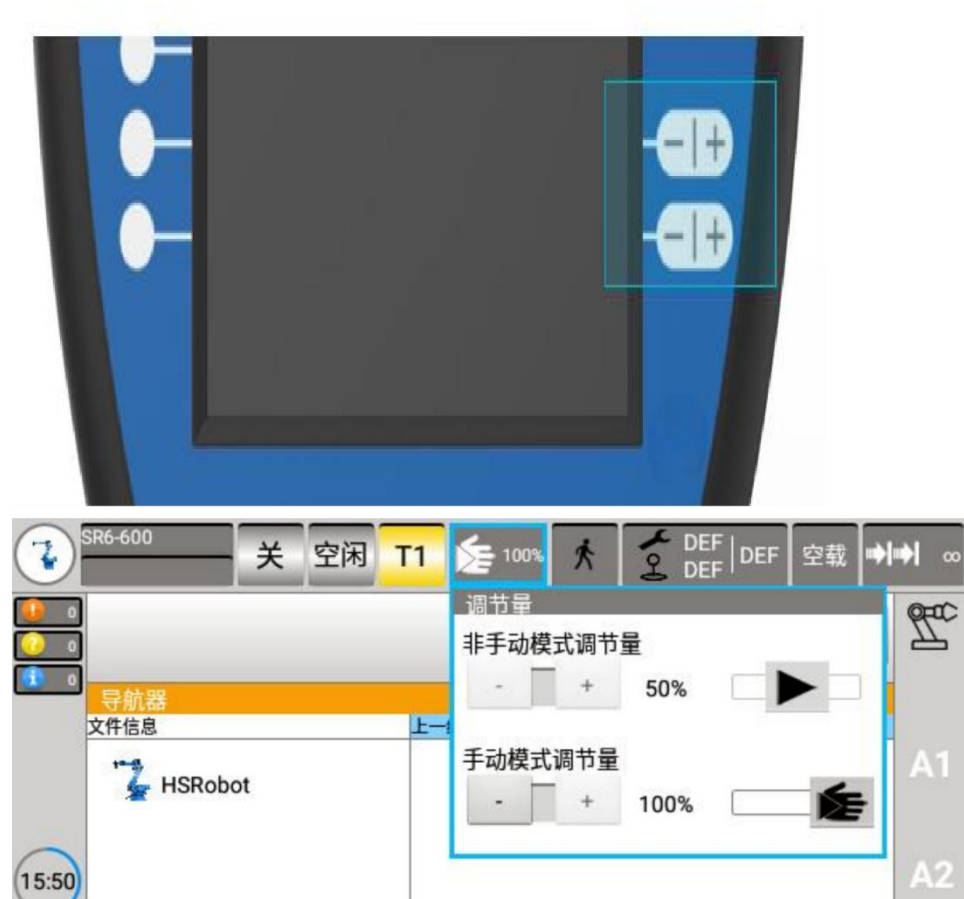


Tip:

When the loader before activation tool workpiece coordinate system, cooperative group for the user to select tool workpiece coordinate system, cooperative group. After the success of the loader (not running), activation of tool workpiece coordinate system, cooperative group switch to a default value DEF (regardless of the tool before loading the workpiece coordinate system, cooperative group is for the default value). After running the program, will show the program called tool workpiece coordinate system, cooperative group, the program has no call to coordinate system is

4.4 Speed ratio

Instructions By changing the speed ratio can adjust the speed of the robot run time. Speed rate expressed as a percentage, can through the teaching of physical ratio button or teach interface adjusted the ratio of the interface.

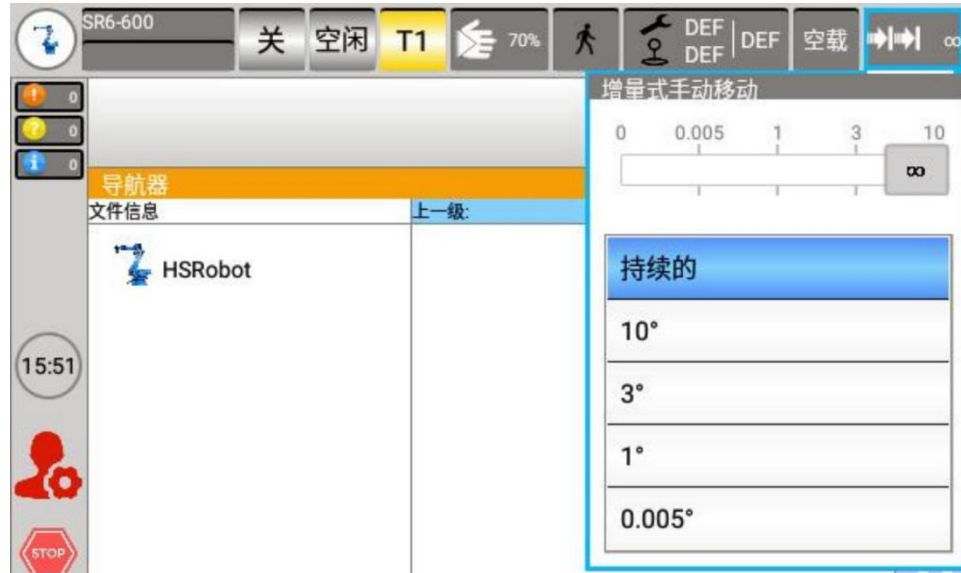


Steps of operation Touch ratio adjustment status icon, opening ratio adjustment window, press the corresponding button or drag ratio will be adjusted. Click on the plus or minus key system will be 100%, 75%, 50%, 30%, 10%, 3%, 1% interval setting ratio for the unit.

4.5 Worldwide inching and inching operation Inching

Worldwide point move

operation is in manual mode, the user holding down the teaching indicator button, a continuous movement of mechanical arm will stop until when the button is released. You need to select the specified action type as "continuous" in the interface below to carry out the point operation.



Steps of operation

1. Select the coordinate system of "Run key" in manual mode
2. Set the manual multiplier.

The following names will appear next to the Run key: A1 -

A6: corresponding robot marked number axis;

X, Y, Z: used for linear motion along the axis of the selected coordinate system; A, B, C: used for shaft rotate movement along the selected coordinate system.

3 Press and hold the safety switch while the enable is on.

4. Press the positive or negative run button, so that the robot movement in positive or reverse direction.

Inching instructions

Inching operation is in manual mode, the user holding down the teaching indicator button, the robot arm will move after the specified distance or Angle to stop movement. Need in the interface of the incremental manual mobile function interface can choose to specify the movement distance or Angle for inching operation. The inching operation is usually used for the positioning movement of the point at the same distance or the occasion of the need to adjust the teaching slightly.

4.6 Moving to the point

Instructions

The function is the accurate mechanical arm moves to show the function of the spot.

Steps of operation

In teaching is the main menu - show - the list of variables selected JR/LR variables, click "change", enter the target coordinates, press the safety switch, click on the "joint to point" or "straight to point", the robot can be run to the target point. If it is the program editing interface, select the motion command line, press the safety switch, and click "joint to the point" or "straight to the point", the function can also be realized.



名称	JR[0]
说明	
轴1	23.235
轴2	33.649
轴3	-50.824
轴4	-111.503



Tips:

- Support manual mode variable list JR/LR editing interface, the movement to "running" condition.
- Cartesian coordinate LR, with different forms, the manual "coordinates", manually enter the coordinates of the, need to be confirmed, to generate form, may be correct only point-to-point movement.
- Sports to have motion state display, below, left for the "movement", the right to "stop".



5 Programming

5.1 Program structure

Program description	● A PRG program is a collection of system instructions and statements that can be recognized and executed by the system. ● The system supports the calling of other PRG programs within a main program, with the main program being the caller and the called program being the subroutine. Subroutines can be invoked using the CALL instruction. ● Program names can include digits, letters, and underscores, but not Chinese characters or pure numbers. Additionally, they must be within 30 bytes in length and are case-sensitive.



Syntax examples:

```
CALL "name"
```

When the main program is executed here, it will jump into the program content of the execution subroutine, after the execution, and then return to the main program to continue to execute.

Structure instruction	The program is divided into three modules: axis group binding and initialization, variable definition declaration, and program body.

Application example as shown in the figure below:

```

<attr>
VERSION:0
GROUP:[0]
<end>
<pos>
P[0]{GP:0,UF:-1,UT:-1,JNT:[72.850820,-113.371212,203.060829,0.10
6136,87.945693,-2.565581,0.000000,0.000000,0.000000]};
P[1]{GP:0,UF:-1,UT:-1,JNT:[39.065293,-105.662922,199.062684,1.42
4838,84.706445,-36.946314,0.000000,0.000000,0.000000]};
P[2]{GP:0,UF:-1,UT:-1,JNT:[2.711088,-119.185497,207.760161,2.265
947,90.737979,-73.147851,0.000000,0.000000,0.000000]};
P[3]{GP:0,UF:-1,UT:-1,JNT:[0.004305,-89.999900,180.000319,0.0001
64,90.001140,-0.000676,0.000000,0.000000,0.000000]};
<end>

```

```
<program>
LBL[2]
J P[3]
J P[0]
J P[1]
J P[2]
J P[3]
GOTO LBL[2]
<end>
```

3

Label item	Instructions
1	Axis group of initialization, binding the business layer and shaft.
2	Define and declare point variables.
3	For the body of the program, add statement blocks.

5.2 Program type

Instructions

Programs are categorized into main programs and background programs. When creating a program, selecting "Group Mask" as "0" designates it as a main program, while choosing "*" designates it as a background program. The current version supports concurrent operation of foreground motion programs (main programs) and background programs. The teach pendant can manage the status of both programs and configure the automatic execution of background programs upon power-up. Additionally, program control instructions are provided to facilitate task state operations.



In the status bar, the program running status of the main program and the background program is supported to view and operate, and the user is supported to click into the program editor interface. The background program contains the "[*]" logo in the program editing interface.



Tips:

- Background programs can run independently
- Support background program control to pause the running of the main program, please see the [Program Control] instructions.

5.2.1 Daemon configuration function

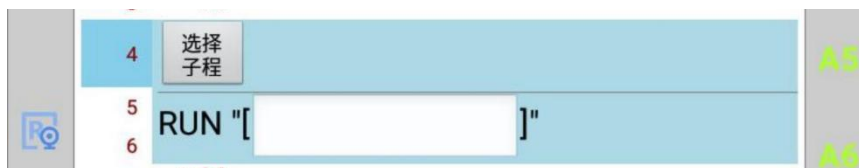
Instructions Background program configuration, the user can configure the power on automatically allow the background program function, background program can configure the controller to run the main program at the same time, can run the logic PRG program in the background at the same time, to achieve multi-thread function. The main program group mask is 0, and the multithreaded logic program group mask is *. It is also possible to make calls to the daemon by running the RUN daemon instruction in the main program.



Steps of operation

1. Select from the main menu → Configuration → Controller Configuration → Daemon Configuration to enter the Daemon configuration interface.
2. Click Select program to select the program that will run automatically when powered on.
3. Click on the function enabled to open the function configuration.
4. Set the number of 4ms program cycles, the default is 1.
5. After setting the cabinet restart, the function takes effect. If the selected program group mask is 0, the setting fails!

Instruction add



Steps

1. selected on the command line of a line to be inserted.
2. select Instruction → Program Control Instruction → RUN.
3. click the "select subroutines" button, and select the logic program.
4. Click the "OK" button.
5. Click the OK button in the action bar and add the RUN instruction to finish.

The constraint

- Background programs run by the following conditions:
- Enabling conditions. When a program at runtime, can make out, the main program finishes running. When enabled may continue to recover after the operation. Background the RUN the program won't end.
- Servo alarm. When a program at runtime, servo alarm, the main program will end. When the alarm is released, it can continue to resume operation. Background the RUN the program won't end.
- The state of the main program error, will affect the operation of daemon
- External signal output by the main program state output.
- External signal input: load the main program, but the background program will be loaded when the main program is loaded; PAUSE only for the main program effectively; Start, uninstall, and stop work for both programs.
- When switching to the background program interface, it is not controlled by the teacher keys (Pause, Uninstall, Back, Start).
- Background process is controlled by the program control soft keys, only the main program is controlled by program control soft keys.
- Pause in recovery after operation. Pause in operation, the main program and the daemon can independently control tentatively, can control continue to restore run independently.



Tip:

RUN when there is a logic program editing, without load, the system will automatically be issued to the latest program system; ABORT, END, including multiple threads PAUSE command only control the current program, does not affect the main program.

Up to allow only a set of logical.prg (background) and a set of robot and motion control (main) programs running at the same time.

- Configuration to electricity after running the program, load the main program contains instructions to add a backend application to RUN, will load failure.
- Perform the RUN daemon process, found that set group of mask to "*" in the program with sports instruction, the robot will stop operation and error.
- RUN after procedures must be in the running state, through the RUN command
- RUN has been running program. When performing a RUN command won't call the police.

RUN program alarm description: when the program execution, the main program or RUN programs appear alarm message, shown in teaching implement. For alarm priority, there is no main program and RUN program, the same priority shows the last alarm information.

5.3 Program Creation

Instructions	The list of system PRG files is as follows. In the navigator page, you can create and delete folders, create, modify and delete PRG program files. PRG program name can support the input of Chinese and English, numbers, underscores, the maximum input of 26 characters. Note: If you create a background logic program, the [group Mask] option should be selected *.

Steps of operation	1. In the directory structure is selected to create the file folder. 2. Click new. 3. Select the program or folder. 4. New applications need to choose corresponding set of mask: the main program to "0", the background program for "*". 5. Give the new file name cannot contain Spaces (name), and press the "ok" button.



5.4 Program Editing

Instructions After creating the program, navigate to the program list and select the desired program for editing. Click on the selected program and then click the "Load" button to enter loading mode. Next, click the "Start Editing" button to access online editing functionality. Once in online editing mode, select the program line where you want to insert new instructions, and then click on the "Instruction" button at the bottom of the teach pendant to choose and insert the required instruction.



Tip:

- Enter the online editing, robot and the running state of the program TAB shows yellow background and show the word "wait", said the current program has been loaded and in the online editing state. Program at this time, not to run, unloading operation, switch mode, need to exit the editor can be carried out only after overall program is running, such as unloading operation.
- The program is loading and running, and cannot be edited online.

5.5 Point online editing

Instructions Point online editing function system is in manual mode, the program is not running state "(pause, when the ready state) to modify the point, without having to uninstall

program operation. When starting the program online editing, the mechanical arm can be moved, and then record the new point position, and then resume the program operation, and the program will execute the movement according to the latest recorded point position.

**Steps of
operation**

1. The manual mode, select the need to modify the movement instruction program line (P, JR, LR are available).
2. Adjust the mechanical arm at location, click the record button, modify complete.



5.6 Program execution

Instructions In manual or automatic mode, can control via the teach.prg execution. The execution operations include loading, starting and running, pausing, stopping and unloading.

- Loading operation: Select the program to be loaded in the program list interface, and then click the "Load" button at the bottom of the teacher to load the program.
- Start and run: After the program is loaded, click the "Start" button in the teacher panel, and the program will start and run. If the program is not loaded, press the

"start" button will have no effect.

- Suspension: if you need to pause, only need to click the "pause" teaching instrument panel button. Program will be suspended.
- Stop, and uninstall: if you need to stop the program, only need to click the "stop" button teaching instrument panel. The program will stop running and automatic discharge.



Tip:

- Syntax error in system alarm code "L" line number for text file error, "SL:" line number for teaching device display programming mistake, "C:" the first "x" characters for the bank mistake.

6 File management

6.1 Program file management

Instructions PRG program files and folders can be copied, backed up to a USB drive, restored from a USB drive, deleted, and locked.

6.1.1 The import and export

Instructions

Backup restore operation

Backup and restore these RPG program from U disk on export and import operations.

Divided into set backup recovery path and backup file two operations.



Path set operation steps



1. Click on the main menu and select File -> Program Data File Management. In the interface, click the "Path Configuration" button to open a dialog box.
2. Set the export and import paths to either "udisk" (U disk) or the default path (root directory of U disk). Under "Debugger" permission, manual input of import and export paths is allowed.
3. In the navigator interface, select the file you wish to back up and click "Export" to complete the backup process. The backed-up file will be located in the designated path.
4. Within the navigator interface, clicking on "Import" will display files from the specified import path. Select your desired import option and click "OK" to complete the file restoration process.



Tips:

The program file storage system controller directory.

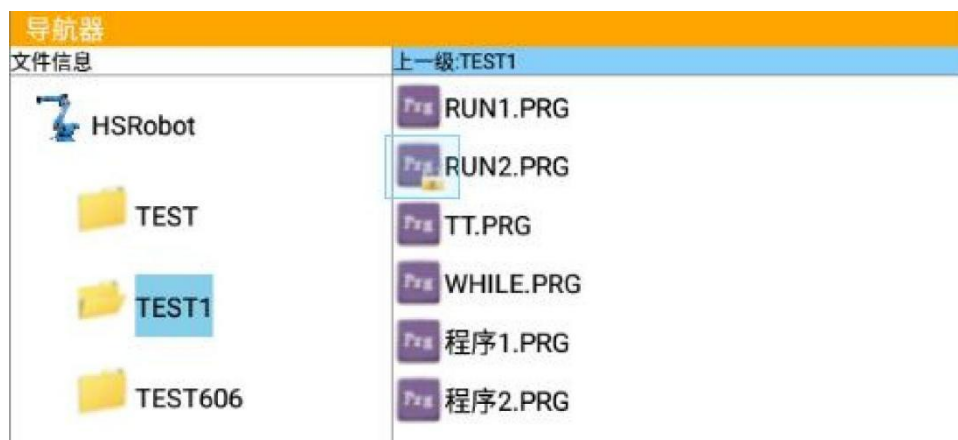
6.1.2 File lock

Instructions

The file lock feature is the action that sets a file to unchangeable. Passwords can be set up to protect.prg file.

1. The tag files in the directory structure.
2. Select more - > lock, click on the "lock" button prompt dialog box.
3. After completion of locking the selected file icon will display a lock style.

File lock operation



Cancel the lock operation

1. Choose has locked files in the directory.
2. Choose edit → lock, click on the "unlock" button prompt dialog box.
3. Enter unlock password, then click ok and unlock the currently selected file.

Change the unlock password

1. In the main menu configuration to teach device configuration, lock password.
2. Enter the original password and new password and then click the "ok" button to save the new password.



Tips:

Lock only for file operations, can not for folders.

The locked file cannot be renamed, deleted, or opened.

- 1 Unlock the initial password for "hspad".

6.2 Program data file management

Instructions Program data file management functions divided into two parts, access to register file and send the register file (variable list register file, IO documents), through U disk access controller register file, and check the file sent to the controller.

Steps of
operation



1. Select → File → Program Data File Management from the menu bar.
 2. Select the check box, choose the file need to import or export.
 3. Click on the "import file"/" export file "button.
 4. Click "confirm" button to confirm the operation.
-



Tip:

- Export to the program selected above data files stored in path regfile folder.
- To import program data files, you also need to put the sent text in the regfile folder selected above. (sending data file take effect after reboot the system).
- Before delivery programs data file values corresponding to 0, all will fail.

6.3 User PLC file management

Instructions

Can U disk backup of the control system of existing users of PLC program files, also can be introduced into the new user to update the PLC program files/recovery system environment.



Export file

1. Insert the USB drive and wait for successful recognition.
2. Navigate to the menu and select File -> User PLC File Management.

3. Click on the "Select Directory" button to specify the export path for the module.
4. Click on the "Export" button, and upon receiving a successful export prompt, locate the corresponding file in the chosen export directory.

The import file operations

1. Insert the U disk, after the U disk is successfully identified.
2. select the menu → File → User PLC file management.
3. Click the [Select File] button of the import module (select app file) to determine the file you want to import.
4. click the "select files" import module button to select the CRC file), sure you want to import the file.
- 5., click on the "import" button, prompt "import file successfully, please restart the controller power!" After power off and restart, the imported user PLC will take effect.



Tip:

- 1 When the network connection fails, you cannot import and export the user PLC.
- 1 PLC need to match the controller version, it is suitable for control version at the same level.

6.4 File management system parameters

Instructions

Can be derived by U disk of the control system of existing calibration parameters, dynamic identification of parameters, such as backup, also can import for calibration parameters, dynamic identification of parameters, such as reduction.



Steps of operation

- 1, inserted U disk, U disk after the success of the recognition
 - 2, select the menu → File → System parameters file management.
- Export:
- 3, click export module [select directory] button to select the directory), determine the choice.
 4. Under the interface of import and export system parameters, click the "Export"

button to complete the export of system parameters.

Import:

5, click the [Select File] button of the import module (select Calbratise.zip) and confirm the selection.

6. Import and export system parameters under the interface, click the "export" button, the prompt "import file successfully, please restart the controller power!" Power to restart after the import is complete.

6.5 Log and system snapshot files

Instructions	Log and system snapshot files help developers locate and fix problems quickly. Log and system snapshot information can be exported to a USB flash drive in the Teacher.



Steps of operation

1. Navigate to the menu and select Diagnostics -> Diagnostic Data Export.
2. Insert the USB drive into the teach pendant, choose the corresponding path, and click on "One-click Export".
3. Wait for a few minutes, and the log file will be sent to the designated folder on the USB drive.

**Tips:**

- The log can be sent to relevant personnel for problem analysis.
- Preserve the environment when issues arise and obtain the necessary files.
- Due to the large size of the log, retrieval may take some time. Your patience is appreciated.
- For non-collision problem diagnostic data export, please select "Use Simplified Data" to save time. "Simplified Data" export can fulfill most of the log data collection needs. The difference from non-simplified data is that it does not include system snapshots, i.e., collected data.
- Data export can be performed according to the requirements of relevant personnel.

7 System management

7.1 The user login

Instructions System setup user level three, respectively, the production, debugging engineer and administrator. In addition, users can also register the custom user in the system.

Different users with different permissions.

- Production: the operator user groups, the groups of users as the default user group.
- Debugging agent: debugging personnel groups of users, the user group owns most of the right to use the system function. The user through the password protection.
- Administrator: Final rights user group, this user has all the rights to use the system.
- The custom user: the user group of the right to use the function of the configuration by the user themselves.

The system default user level for production, if you need to change, can be performed in accordance with the following steps.

Steps of operation

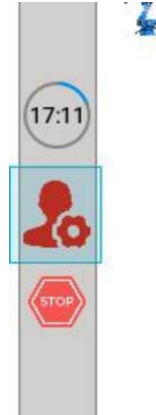


1. From the main menu, navigate to Configuration -> Teach Pendant Configuration -> User Group to display the current user group.
2. To switch to the default user group, click the "Standard" button. (If already in the default user group, the "Standard" option cannot be used.) To switch to another user group, select the desired group and press Login.
3. Debuggers and administrators need to enter a password for login and then confirm by clicking OK.
4. To modify a password: If it is necessary to change a user's password, select that user and press the "Password..." button.
5. In the password modification interface, input both the old and new passwords before clicking OK to complete the process of changing passwords.



Tips:

Click the "user shortcut login" icon of the operation interface to quickly enter the user login interface.



7.2 User rights configuration

Instructions User rights configuration function refers to limit a user according to the demand of the function of different operational matters, by configuring its operational matters, reduce system was wrong operation leads to the possibility of instability.



The configuration process

1. Registering and Logging in Users:

- In the Teach Pendant Configuration → User Group Login, click the "Register"

button and follow the steps to complete the registration process for a new user.

- After registration is completed, select the registered user for login on the login interface.

2. Customizing User Function Permissions:

- Log in with administrator privileges.
- Click the "Permission Configuration" button to access the permission configuration interface for customizing user operation permissions..

7.3 Controller connection

Instructions	Communication controller is the controller and the function of teaching device through the Ethernet connection, need to teach configuration on the controller's IP address and other information.

Steps of operation



1. Log in as the "Debugger" user (switch to Debugger or higher privileges).
2. Navigate to the selection menu bar → Communication → Teach Pendant Communication Configuration.
3. Enter the corresponding IP and port for the controller, then click the "Save" button.
4. Restart the teach pendant (select from the menu bar → System → Restart Teach Pendant, or manually close and reopen the "HSC3-HSpad" app).

7.4 Controller IP Configuration

Instructions Multiple controllers can be configured IP address or modify IP address.

Steps of operation



1. Log in as the "Debugger" user (switch to superuser or higher privileges).
2. Navigate to the menu bar and select → Communication → Controller IP Configuration.
3. Click the "Enable" button to configure the "IP Address" and "Subnet Mask", with provision for configuring up to two controller IPs.
4. After configuration is complete, click "save" button, the configuration take effect.
- 5, by clicking on the "enable" and "disable" button, can adjust its network status.

Tip:



- IP address can not be the same as the controller IP, such as: 10.10.56.213 configuration to 10.10.56.214 or with the old way n file new ip address conflict, such configuration will cause system module error.
- Pay attention to the IP address is 192.168.2 X segments have been system for internal use, please avoid to use.

7.4.1 IP description and configuration application

Standard cabinet controller IP description	Object	Default IP
	LAN0	IP : 10.10.56.214 Port: 23234 IP : 90.0.0.1 Port: 23234
	LAN1/ECAT	IP : 192.168.2.214 Port: 23234

Drive control cabinet controller IP description

Object	Default IP
Teacher connector	IP : 10.10.56.214 Port: 23234
MLAN	IP : 10.10.57.213 Port: 23234
LAN2	IP : 10.10.58.212 Port: 23234
LAN1	IP : 10.10.59.211 Port: 23234

Demonstrator IP

The teacher IP can be viewed and set through the "Ethernet" application on the teacher.

Note: the IP address and Gateway address you want to use the same network segment.

Object	
IP address	10.10.56.123
Netmask	255.255.255.0
Gateway address	10.10.56.1

7.5 System authorization

Instructions The controller needs to register after authorization to use.

Step of operation



in the USB flash drive.

- Put after registration file genAuthCode get U packing directory, inserted U disk, click the license button to obtain files to complete the registration.

7.6 System software restarts

Instructions

Software to restart, the control system of this function is not the same as power to restart. (partial scenarios need to restart the power to restart the system).

Steps of operation

Select "System" from the main menu, then choose "Restart System". In the dialog box that appears, select "OK" to restart the controller.



Tip:

Some Settings need to restart the system to take effect, such as: load, send register file, etc.

7.7 System upgrades

Instructions

You can version upgrade the system software through the teacher, and the upgrade package extension is .tar.gz.

Steps of operation

1. Insert the USB key with the upgrade package into the teacher.
2. Select → System → System Upgrade from the main menu to display the upgrade window.
- 3 Select the file to upgrade.
4. Click Send Update package.
- 5 Power off and restart the control system.



Tips:

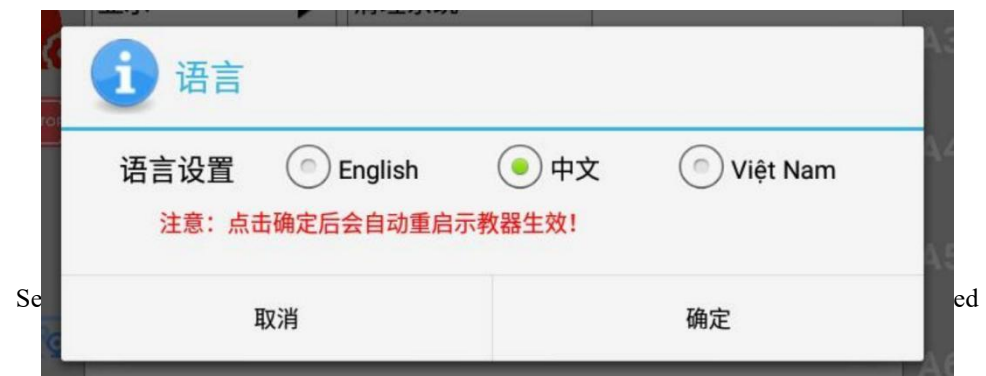
- Before upgrading the system, first return the robot to the zero position, power off and restart once or power on for the first time, do not do any operation before performing the upgrade operation, need to first [clean the system] to release disk space.
- If U disk to identify failure, can be set up to teach indicator system to select
- USB storage after uninstall SD card to install the SD card again.

7.8 Switching System language

Instructions

The language of the interface can be set on the teacher system.

Step of operation



language and click "OK". Restart the teach pendant to apply the changes.

7.9 System version information

Instructions

This feature can view the current system version information.

Steps of operation

1. Navigate to the main menu and select "Information" followed by "Version Information" to display the version details.
2. This allows you to view the current teach pendant, controller, and robot nameplate version information.

8 Application functions

8.1 Encoder-decoder function

Instructions Encoding function is R register mapped to output IO, the decimal number is converted to a binary number, and mapped to the corresponding IO process. For example DO [1] - DO R [1] and [4], which R [1] = 3 (binary 0011 b), then DO [1] = 1, DO [2] = 1, the rest of the DO is 0; Decoding function is to map the input IO to R register process, the process of binary decoding for the decimal number. For example, DI[1]-DI[4] maps to R[2], and the external input DI[2]=1, then the corresponding R[2] = 2(binary 0010b).

Step of
operation



1 Select → Configuration → Controller Configuration → Encode/Decode

Configuration from the main menu.

2. Click to select either Encoding Settings or decoding Settings.
3. Click the "Add" button to enter the Settings screen.
4. When the setup is complete, click the "OK" button to complete the setup.



Tips:

Encoding R [0] = 42, and associated DO [0 ~ 6], turn the decimal 42 binary is 0101010, DO the corresponding value of [0 ~ 6] for 0101010, if the associated DO [6] or more output signals, such as DO [7 ~ 10] also is 0.

8.2 One key back to zero function

Instructions

A key to zero function can be divided into back to zero program configuration and a key to zero operating two parts.

- Back to zero program configuration is a configuration back to zero movement.prg program execution.
- One-key back to zero operation is the system executes the back to zero motion program. When the back to zero signal is triggered, the system will automatically execute the program, so as to simplify the operation steps of the manipulator back to the set zero.

Examples of return to zero procedures:



```
LBL [X]
WAIT DI[10]=ON          'Guard: wait for the signal to be satisfied before
                        proceeding, otherwise block on this line forever
LR[1]=LPOS              'Get current position: assign to LR[1] register
IF LR[1][0]<200 ,GOTO LBL[1]
IF LR[1][0]>=200 ,GOTO LBL[2]
LBL [1]
```

L P[0]	‘return to zero
LBL [2]	
L P[1]	‘Return to zero transition point 1
L JR[0]	‘Getting to Zero

Back to zero program Configuration operation

- 1 Select Configuration → Controller Configuration → Back to Zero Program Configuration from the menu bar.
2. Click the "Configure Program" button and select the Back to zero program.
3. Click the "Save" button to complete the setup.

A key back to zero

1. The manual mode, the main program of free state, the open can.
2. Click the "One-click back to Zero" icon on the right to execute the back to zero program. (See the sample image below).



Tips:

- Back to zero execution preconditions for will open.
- The zero-back program must be executed when the main program is not loaded (idle).
- The execution of the zeroback program can only be executed in manual mode and automatic mode.
- Executed immediately back to zero movement under manual mode, automatic mode should be secondary to confirm.
- Back to zero, after the completion of program to write or modify must be loaded again will the latest program update to the controller.

8.3 Register Binary Bit Operations

Instructions

The system is adapted for use with PROFINET signaling process packages that support the binary digits of the R register value as a supplement to the IO signal. When this function is enabled, the value of the R register will be converted from decimal to binary, and then the first eight bits of the value will be changed to signal. That is, value 1 is signal ON and value 0 is signal OFF.

decimal	170	171
Bit sorting	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
	↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
binary	1 0 1 0 1 0 1 0	1 0 1 0 1 0 1 1

Operation

process

1. Install the PROFINET signaling process package.
2. Start the signal mapping function switch in the process package, and the system mapping corresponding register signal is enabled.
3. Use the corresponding signal instruction when programming.

Program Example:

```
IF R[0]=255 THEN 'That is, the signals from R[0].0 to R[0].7 are all ON
R[0].0=0 'Set the R[0].0 signal to OFF, at which time the value of R register
R[0] is changed from 255 to 254
R[1].1=1 'Setting the R[1].1 signal to ON
END IF
WAIT R[2].0=ON 'Wait for R[2].0 signal to be ON
```



Tip:

The installation of the PROFINET Signal Craft Kit occupies R[999] as a function enable switch, so please take care to program its use. Changing the value of R[999] register will affect the function enable.

Enable Function R[0] to R[199] are input signals and R[200] to R[399] are output signals, so please take care to program them for use.

Avoid using the R[998] register at the same time when this process pack is enabled.

Enabling profinet communication requires the system to be configured with PROFINET gateway hardware before operation.

8.4 External control signal configuration

Instructions

In the external mode (see the “Introduction to Operation Modes” section for mode descriptions), the robot arm can be controlled only by external signals. When using external control, you need to perform the system signal configuration operation before you can send the relevant signal to control the robot arm to perform the corresponding action.

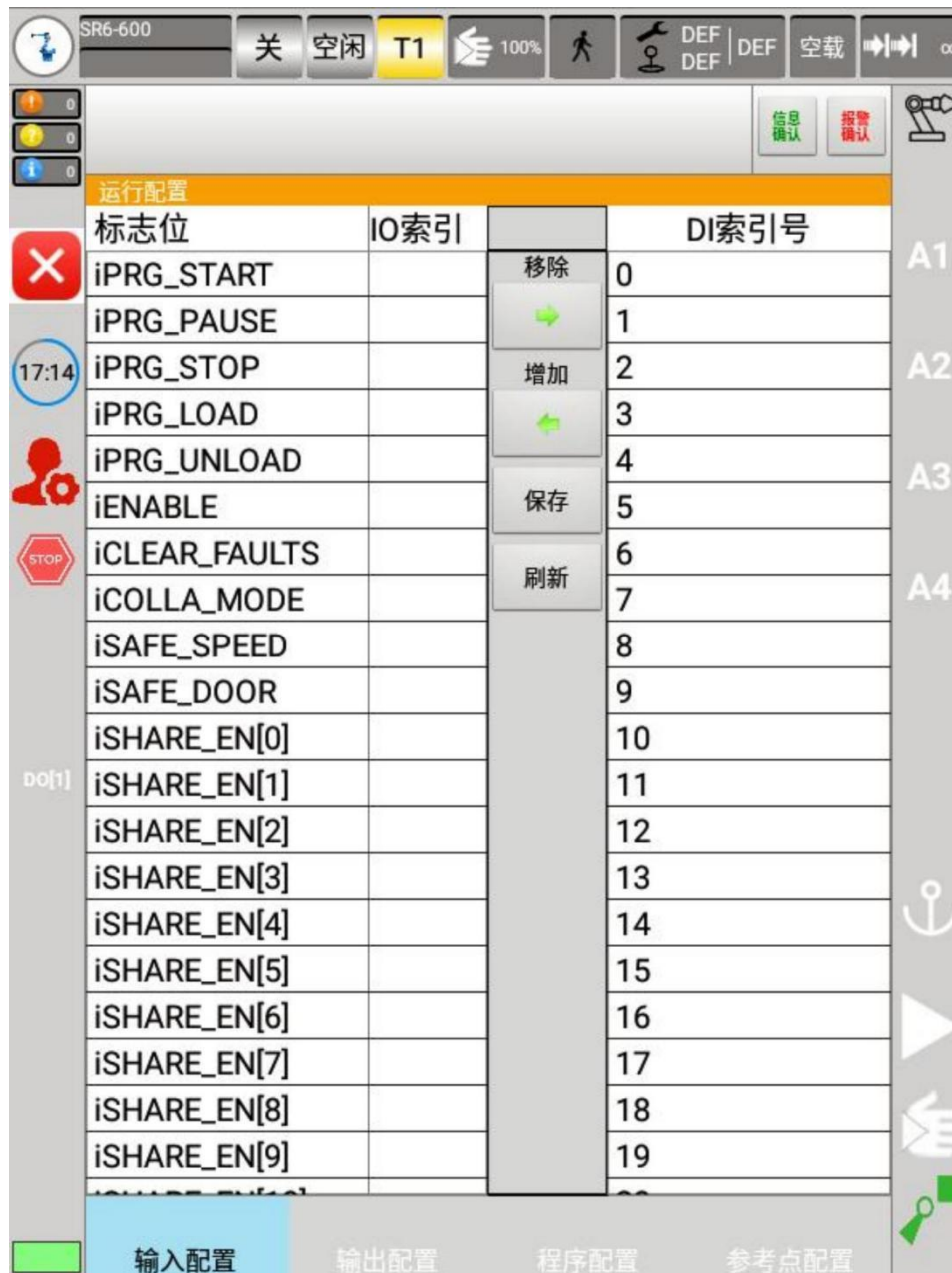
The “Signal Configuration Operation” is the process of mapping system signals to IO input/output indexes (i.e., binding functions to IOs). After the mapping relationship is established, programs can be executed by IO signals, and the robot status can be acquired.

The system is not configured for external signals in the factory settings, so the user needs to configure the mapping according to the actual needs.

The operations performed in external mode are enable, program load, program run, program pause, program stop, get program status, etc. The system provides many functional signals, see the chapter “List of External Signals” in the appendix for details.

Programs loaded and run in external mode need to be completed first through the “Program Configuration Operations” below.

Signal configuration operation



1. Select → Configuration→Controller Configuration→Running Configuration in the main menu to enter the Running Configuration interface.
2. Click “Input Configuration”, select “Flag Bit” on the left side of the screen, select the DI index number on the right side, and click to add “<-” to establish the mapping relationship, and click to remove “<->” to cancel the mapping relationship. -> Click “Remove” to cancel the mapping relationship.
3. Click “Save” to save the operation.
4. Click “Output Configuration”, similar to the above operation, you can set the output mapping relationship.



Tip:

- The running configuration can only be operated in manual mode T1/T2 with the main program not loaded (idle).
- Input Configuration: Bind the input signals to complete the system operation by triggering the input signals such as: enable, load, pause, run the program, clear the alarm and so on.
- Output Configuration: Binding output signals, used to display some statuses of the robot through IO outputs, such as: program status, enable status, current mode, area signals and so on.
- To view the detailed configuration information table, you can view the detailed signal information in the section “List of External Signals” attached to this document.

8.4.1 iSAFE_DOOR Safety Gate Signal

instructions The system supports the safety gate signal, after the user opens the safety gate, the robotic arm can receive the signal that the safety gate is open and pause the operation automatically; after the user enters the safety gate and finishes the transaction, he or she exits the working area and closes the safety gate, then presses the start button from the instructor or button box, the robotic arm continues to run the program. In the case of not closing the safety gate, the robot arm is not allowed to start the function signal of program running.



Tip:

- (1) When ISAFE_DOOR = OFF (when the safety gate is open), the program is suspended in both manual automatic and external modes.
- (2) The ISAFE_DOOR signal is interlocked with the program start signal.
 - When the safety door is open, i.e. ISAFE_DOOR=OFF, starting program execution is not allowed in any mode (including manual, automatic and external modes); however, JOG moving the robot arm and robot arm movement to the point in manual mode is allowed.
 - When the safety gate is closed, i.e., ISAFE_DOOR=ON, the program can be triggered to start operation by a pushbutton or the like.
- (3) The safety gate signal only affects the main program, not the run program.
- (4) Considering the version compatibility and realistic requirements, this signal is a configurable signal, the user can decide whether to configure this signal and the input IO port corresponding to the signal, if the signal is configured, due to the signal is normally closed, the debugging phase must ensure that there is a high level input.

- (5) According to the safety requirements. The safety door signal ISAFE_DOOR cannot be set as a virtual signal, this requires that the system internal implementation cannot be switched to a virtual state . Prevents misuse of the Oscillator|PC software.
- (6) The function of the system's program pause signal IPRG_PASUE is still valid and does not conflict with the function of the safety gate signal.

8.5 External Runtime Program Configuration Operations

Instructions In external mode, the robot arm can be controlled only by external signals. The system supports setting up the function of calling the system running program according to the R register information mapped by the program configuration.

Procedure



SR6-600 关 空闲 T1 100% DEF DEF 空载

信息确认 报警确认

运行配置

程序映射信息

程序映射寄存器 R[10] 修改

程序映射指定方式

☒ 列表序号 ☐ 程序名称

程序配置信息

序号	程序名	配置	删除
0	TEST1.PRG		
1			
2			
3			
4			
5			
6			
7			
8			
9			

+100 -100 保存

输入配置 输出配置 程序配置 参考点配置

1. Click the “Program Map Registers - Modify” button and enter the R register index number.
2. Click the “Program Map Register - OK” button to complete the program map register setting.

3. Cursor click to select any serial number, click “Configure” button. (There are 511 index numbers, 51 external programs can be configured at the same time).
4. The program selection screen pops up, click to select the program to run.
5. Click the OK button to complete the external program configuration.
6. Click “Save” on the action bar to complete the configuration of the external program.

**Tip:**

- After the external program configuration, if there is any modification of the program (including subroutines) or the first time loading, it is necessary to load it again in manual or automatic mode to send down the program to update to the controller.
- Assuming the program mapping register is R[1], when $R[1] = 2$, the external mode loads the program with index number 2, and when it is 0, it loads the program with external configuration index number 0. Different programs are loaded by the value of R register.
- After the configuration of external control signals is completed, the operation mode can be switched to external mode, and the robot can be debugged and verified through virtual IO trigger signals. In the interface of “Display-Input/Output-Digital Input/Output” in the menu bar, switch to Virtual state and click “Value” to operate the virtual signal.

8.6 Reference point signal output function

**Functional
description**

The reference point signal output function is a function for the system to output an oREF signal when the robot arm is located at a specific joint position. To use this function, the reference point position information must be configured and bound to the relevant output IO.

During the configuration operation, the JR register and the joint precision range value must be specified. When the robot is currently in the position of the specified JR register (e.g., JR[0]={0, -90, 180, 0, 90, 0}, the error between the current actual position value and the set value is within the precision range), the system outputs the position signal oREF.

The configuration operation of the reference point signal output function is described in the section “Reference Point Configuration Operation” below.

Reference point configuration operations



1. In “Configuration→Controller Configuration→Operation Configuration”, click “Reference Point Configuration” button to enter the reference configuration interface.
2. Click the “Add” button to set the “Reference Point Number”, “Register” index number, and “Accuracy”.
3. Click the “OK” button to complete the settings.



Tip:

Inputs DI [100~106] and DO [100~112] are used internally in the system, so be careful to avoid using them.

8.7 Synergy Functions

Functional Description	The Synergy function realizes cooperative movements of the robot and external axis groups. ● Cooperative control: Function in which the robot axis group and the external axis group work in coordination, and there is a cooperative interpolation motion between the two axis groups. ● Individual control: Function in which the robot and the external axis group work independently of each other. ● Main side: The active part of the combined motion of the axis groups. In cooperative control, when the master side moves, the slave side follows (keeping the relative positions of the master and slave side constant). Usually, the external shaft group is set as the master side. ● Slave: The slave part of the combined motion of the axes group. In cooperative control, when the master side moves, the slave side follows (keeping the relative position of the master and slave side constant); when the slave side is operated by pointing, the master side does not follow. Usually the robot axis group is set to the slave side. ● The synergy function is commonly used in the welding process where the robot arm and the shifter move together to complete the welding process. (The robot arm synchronizes its positional attitude change when the shifter moves to change its position). Please consult the relevant technical personnel for welding-related process operations. ● The process of using the synergy function is generally as follows: synergy configuration → synergy calibration → call the synergy group number for point teaching, and then run the PRG program to realize the synergy action.

8.7.1 Co-Configuration of the Translocator

8.7.1.1 Collaboration Group Configuration

Instructions	Co-group refers to the robot axis group and the external axis shifter after the combination of the whole, the shifter co configuration is the process of specifying the co-group number and configuring the external axes for co-motion with the robot. The co-group number is called in the PRG by the co-command COORD_NUM, and the system will perform co-motion after calling the co-command.

Procedure

变位机协同配置



协同组配置
协同标定

变位机协同配置

协同组号

0 ▼

轴序号	逻辑轴号	类型	操作	
0	未配置	未配置	修改	取消
1	未配置	未配置	修改	取消
2	未配置	未配置	修改	取消

刷新

保存

下一步

Label	Description
[Logical Axis Number]	Logical axis number indicates the index number of all axes inside the system (the system supports 9 axes of point information). 6-axis robotic arm's logical axis number is 0~5, and the logical axis number of external axes are listed in the order of 6, 7, 8 in sequence downwards.
[Type]	That is the type of external axis

	translators, axis types are rotary axis and linear axis.

1. Open the → Configuration → Controller Configuration → Positioner Collaboration Configuration in the menu bar.
2. Click the [Collaboration Group No.] drop-down box to select the serial number of the collaboration group.
3. Click the [Modify] button, select the [Logical Axis Number] drop-down box in turn, and select the corresponding external axis index number.
4. Click the [Type] drop-down box option, select the axis type (X ~ C, currently supports rotary axis type, select X can be).
5. Click the [Save] button, the cooperative group number and external axis logical axis number binding is complete.



Tip:

When performing a synergy group configuration, if two external axis translators are required to synergize, 2 logical axis numbers need to be configured.

8.7.1.2 Co-calibration

Instructions

When a robot axis group and an external axis translator group perform a cooperative action, their mutual positional relationship must be determined in advance. The process of determining this mutual position relationship is the calibration between the robot axis group and the external axis group.

Procedure

变位机协同配置



协同组配置
协同标定

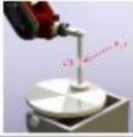
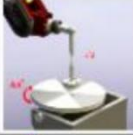
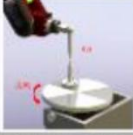
配置标定信息

协同组号

协同组名

标定方法

标定数据

	未记录
	未记录
	未记录

标定结果

1. Click [Next].
2. Click on the [Collaboration Group No.] drop-down box to select the serial number of the collaboration group. (This cannot be set arbitrarily, you need to select the group number of the collaborative configuration just set).
3. Click the [Collaboration Group Name] input box and enter the comment description of the collaboration group number.
- 4, select the [calibration method] drop-down box options, select the 5-point method (2 external axes in collaboration with the use of 5-point calibration, an external axis using the 3-point method calibration).
- 5, click the [Start Calibration] button, in order to obtain the mobile arm and the external axis, record the coordinates of the point (calibration method see below).
- 6, after the completion of the record, click [Calibration] button, the calibration results pop-up.
- 7, click [Save] button, complete the cooperative group calibration.



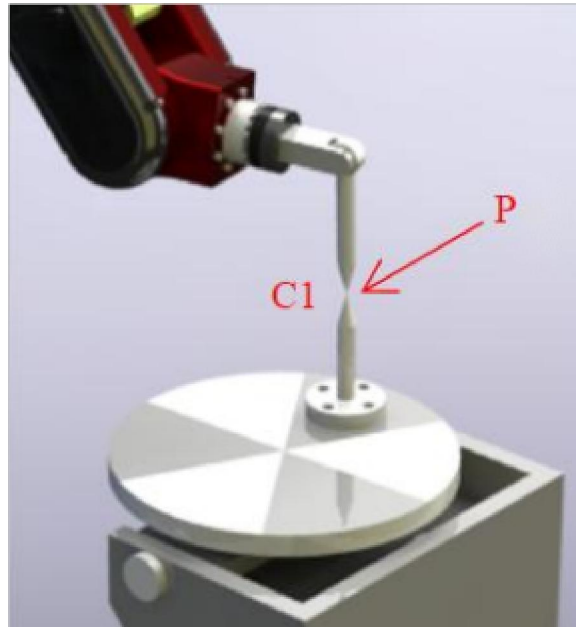
Tip:

The following conditions need to be met when performing co-calibration:

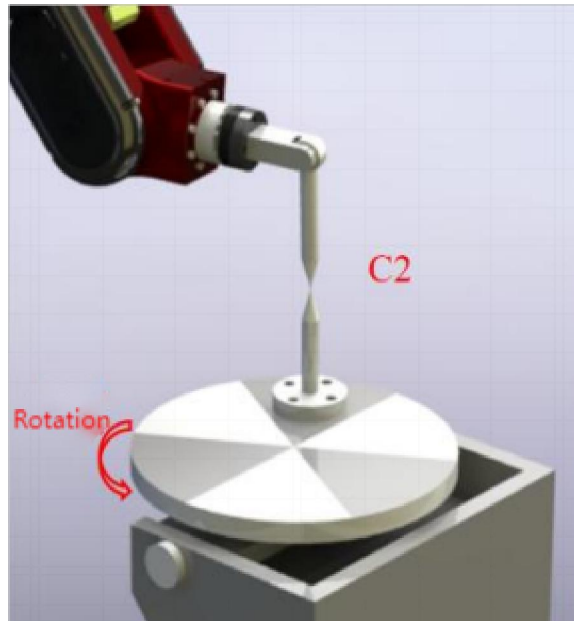
- The external axis has been zero calibrated.
- External axis limits are set.
- Perform tool calibration.
- Click the [Calibration Result] button to see if the current co-group number has been used.
- Calibration must be performed in the positive direction of the robot

**Coordinated
3-point
calibration
(external axis
group
calibration
when the
number of
external axes is 1)**

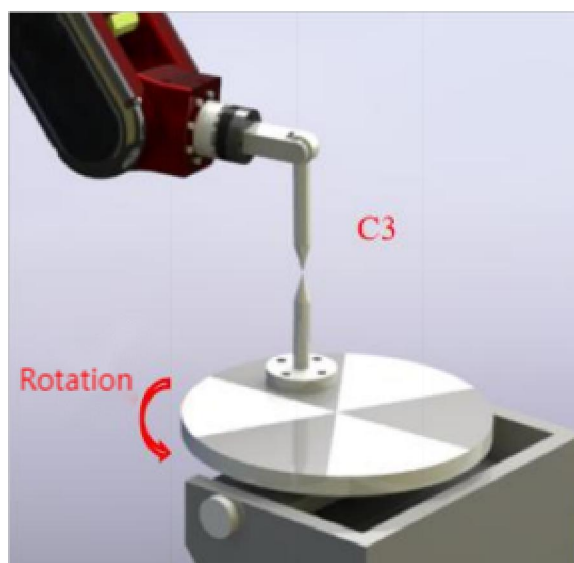
1. Determine an arbitrary point (point P) on the translocator's rotary table (as far away from the rotary table rotation as possible) Merge the control points of the robot with point P and record the 1st point C1.



2. Rotate the indexer shaft by any amount. There is no limit to the amount of rotation, but select 30° or more. Select positive direction calibration. Merge the controller point of the robot with the selected P point and record the 2nd point C2.



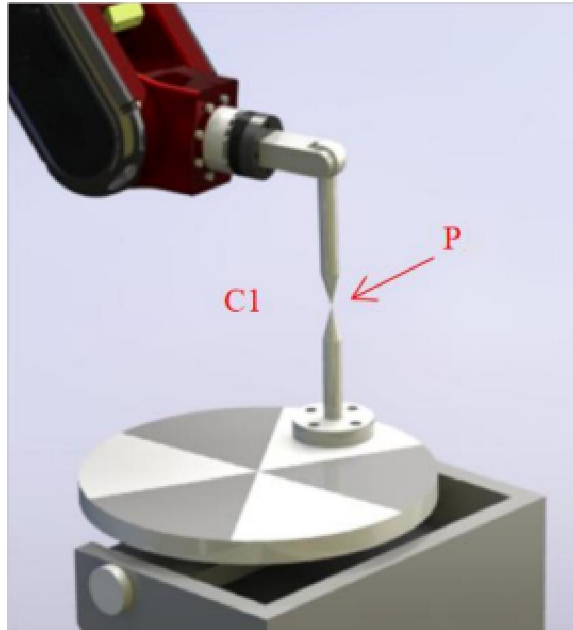
3. Rotate the translocator axis in the direction of rotation in 2. Merge the control point of the robot with the rotated P point and record the third point C3.



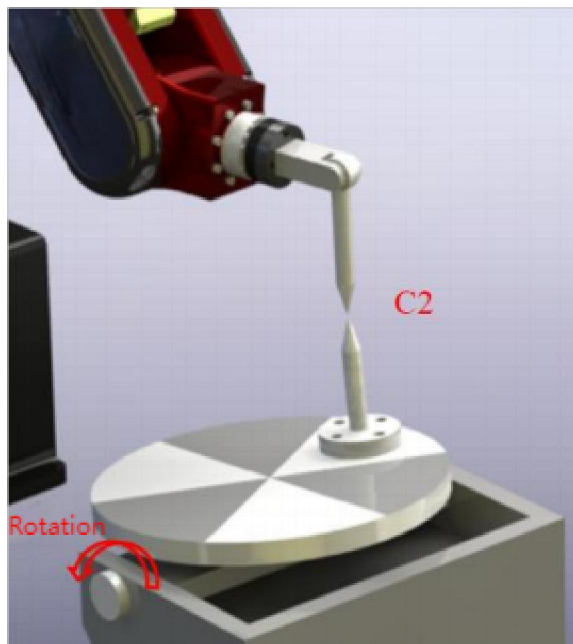
**Synergistic
5-point
calibration
(external axis
group
calibration
when the
number of
external axes is 2)**

1. Determine an arbitrary point (point P) on the turntable (as far as possible from the center of rotation of the turntable).

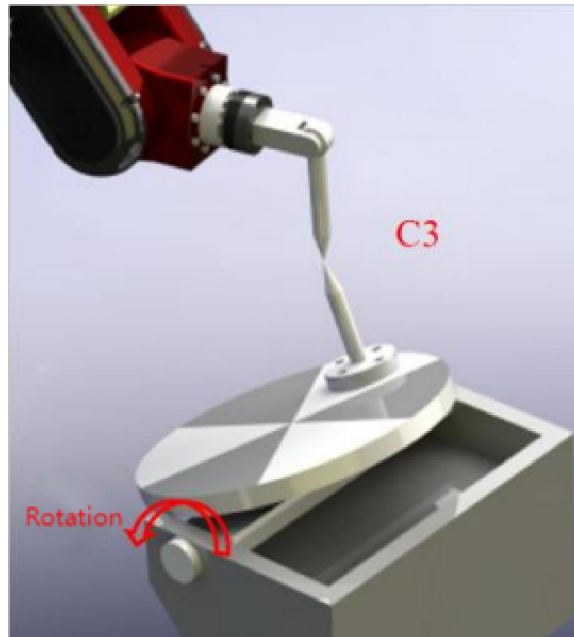
Make the 1st axis of the translator axis first pair horizontal to the earth, merge the controller point of the robot with point P, and record point C1.



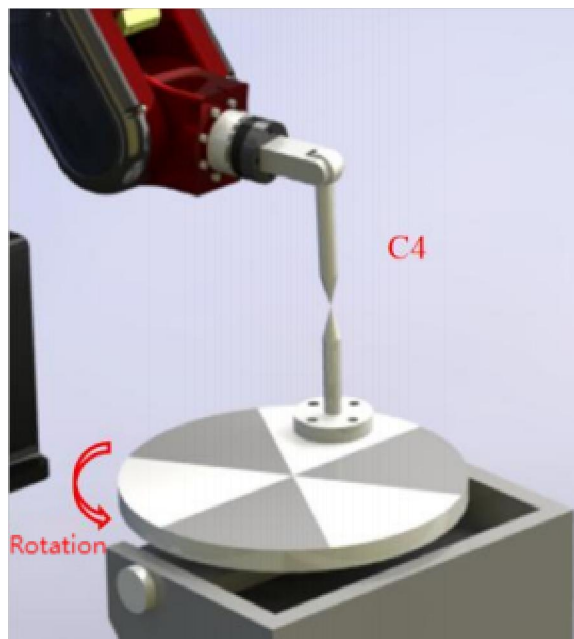
2. Rotate the 1st axis of the robot axis by approximately 30° , merge the controller point of the robot with point P, and record point C2.



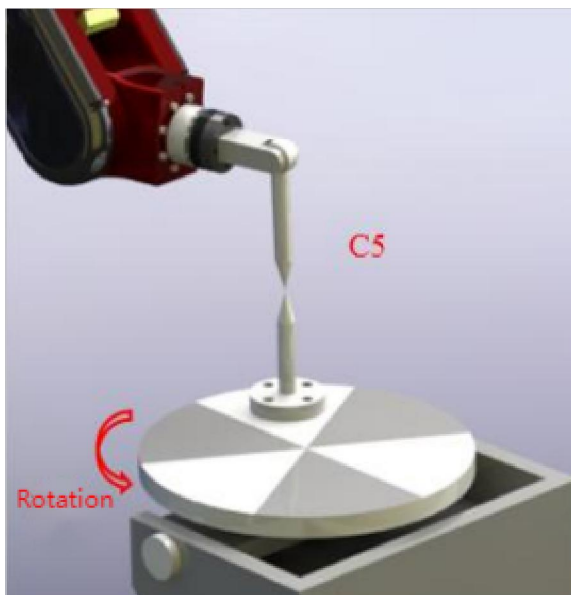
3. Rotate the 1st axis of the translocator shaft by approximately 30° further, merge the control point of the robot with point P and record point C3.



4. Return to the point C1 position and rotate the 2nd axis of the translocator shaft by approximately 30° . Merge the robot's control point with point P and record point C4.



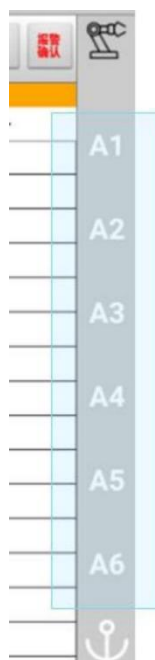
5. Rotate the 2nd axis of the translocator shaft by approximately another 30° . Merge the robot's control point with point P and record point C5.



8.7.2 Collaborative teaching

Instructions After the configuration of the cooperative group and the calibration of the cooperative group are completed, you can switch to the external axis. Then call the corresponding cooperative group number. After calling the cooperative group, the identifiers of the external axis E1~E2 will change to C1~C2, indicating that it is currently in cooperative state, and when the external axis is tapped, the robotic arm will cooperate with the variator to carry out cooperative movement.

Procedure



1. Right Axis Icons A1-A6, then click on the “Robot Axis” drop down box option. 2.

2. Click to select the “External Axis” option. 3.
3. Click the status bar tool artifact bar option, “Collaboration Group Selection” drop-down box option, after selecting the collaboration group number, the right axis will be displayed as C1~2 from E1~E2, at this time, tap C1 or C2, the robotic arm will collaborate with the shifter to carry out the collaborative motion.



Tip:

- When co-demonstrating, the program needs to be called to the corresponding co-group number. When co-motion is not used, the co-group number is set to the default, and the program uses the instruction to set COORD_NUM=-1.

8.7.3 Ground Track Synergy

Instructions

Ground rail synergy function, is the function that the robotic arm can move together with the ground rail synergistically. This function can be used at the same time with the Shifter Synergy Function to realize the operation of robotic arm + Ground Rail + Shifter synergistic welding. The simultaneous movement of robot arm and ground rail supports two forms:

- (1) Time-synchronized movement, i.e., the floor track and the robotic arm start and stop at the same time. The precision of the end of the robot arm cannot be guaranteed, and it is only suitable for production line handling.
- (2) Synergistic motion to ensure the trajectory accuracy and running speed of the end TCP. Suitable for welding and other high-precision requirements of the scene.

地轨协同配置

地轨配置信息

地轨轴号: 6 ▼ 取消标定

地轨坐标系标定

参考点1		#{370.0,-124.999,245.0,180.0,3.1E-6,180.0,0.0,0.0,0.0}	记录
参考点2			记录
参考点3			记录

标定结果
标定

标定验证

协同点动移动地轨轴时，如果机器人本体TCP点位置相对于固定的尖点是不变的，则地轨轴标定成功: 如果机器人本体 TCP 点相对于固定的尖点位置发生改变，须重新标定。

重置
保存



Tip:

- For detailed operation, please refer to “V1.6. 11_Mechanical Arm and Ground Rail Synergy Function User Manual” .
- The calibration of the ground rail coordinate system is to calibrate the world coordinate system, and the default call of the system is changed from the base coordinate system to the world coordinate system based on the calibration.
- To cancel the calibration of the world coordinate system, click the [Reset] button in the interface to confirm and restart the system to take effect.

8.8 Fallback function

Instructions

The fallback function means that after the robot arm is enabled to run the loaded program on the manual mode, at every click of the fallback button, the robot falls back to execute the instruction pointed to by the fallback pointer in the currently loaded program.



Tip:

If there is a back pointer displayed in the program, it means that the current scenario can be backed up, and if there is no back pointer displayed, it means that the current scenario cannot be backed up, and pressing the back button will prompt the corresponding alarm message.

The rules for each instruction under the fallback function are shown in the table below:

directives		Rollback processing method
Motion instructions (J, L, C, A, JUMP)		Moves to the target point recorded for the motion instruction indicated by the backward pointer with the motion attributes and motion mode of the last executed motion instruction.
Conditional instructions		A SELECT, CASE, or ELSE instruction may not be encountered to continue the backward step. The remaining conditional instructions are ignored.
process instruction	CALL	Ignore.
	GOTO、LBL	Backtracking may not be continued when this instruction is encountered.
cyclic instruction (FOR、END FOR、WHILE、END WHILE、BREAK)		Ignore.
IO instruction (DO、WAIT、PULSE)		Ignore.
program control instruction (PAUSE、ABORT、END)		Backtracking may not be continued when this instruction is encountered.
Other directives (THROW、RUN、SET_TR、SET_TOOL、SET_FRAME)		The fallback may not be continued by encountering the RUN instruction. Encountering the SET_TOOL, SET_FRAME directives may not perform backtracking. The remaining directives are ignored.

assignment instruction		Encountering UTOOL_NUM and UFRAME_NUM assignment directives may not continue the fallback. The remaining assignment instructions are ignored.
Motion Command Properties (VEL、VROT、CNT、ACC、DEC、SKIP、CNT_TYPE、COORD、SMOOTH、TURN、ALARM、PASS)		Ignore.
Motion instructions contain additional instructions	OFFSET	1. If this additional instruction is a direct position compensation instruction (Example: L P[0] ,OFFSET LR[2]) can be executed backward to move to the target point recorded by the motion instruction pointed to by the backward pointer plus the point position after the compensation amount with the motion attribute and motion mode of the last executed motion instruction. 2. If this additional instruction is a position compensation instruction (Example: L P[0] ,OFFSET), as long as you have run the conditional compensation instruction once (example: global assignment instruction OFFSET_CONDITION=LR[2]), you can execute the backward, if not executed, you can not execute the backward, the robot stops running and alarm.
	TOOL_OFFSET	1. If this additional instruction is a direct tool compensation instruction (Example: L P[0] ,TOOL_OFFSET LR[2]) can be executed

		backward, to the motion attributes and motion mode of the last executed motion instruction to the point position of the target point recorded by the motion instruction indicated by the backward pointer plus the compensation amount. 2. If this additional instruction is a position compensation instruction (Example: L P[0] ,TOOL_OFFSET), as long as you have run the conditional compensation instruction once (example: global assignment instruction TOOL_OFFSET_CONDITION=LR[2]), you can execute the fallback, if not executed, you can not execute the fallback, the robot stops running and alarm.
	INC	Backtracking cannot be continued when this instruction is encountered.
jump	Jump to motion commands	Both the sequence and fallback pointers point to the instruction on that line. Executing fallback takes the motion attributes and motion of the instruction pointed to by the fallback pointer back to the target point recorded for the instruction on that line, and when fallback is complete, the sequence pointer moves down and the fallback pointer moves up when execution is complete.
	Jump to instructions that are ignored by fall backs	The sequence pointer executes the instruction to be jumped, and the backward pointer points to the nearest motion instruction above the

		instruction to be jumped (provided that there is no instruction between the two instructions that is not executable for backward movement).
	Jump to an instruction that does not execute a fallback	The sequence pointer executes the instruction to be jumped, the backward pointer is not displayed, and the backward alarm is executed.
The point data recorded by the motion command does not correspond to the coordinate system of the tool workpiece currently in use on the display.		No fallback operation is allowed and an alarm is raised.



Tip: If the sequence pointer and the fallback pointer point to different programs, the display of the interface is based on the program where the sequence pointer is located.

You can only fallback in manual mode, fallback is single step by default.

8.9 Displaying global variable descriptions on program lines

Instructions Displaying global variable descriptions in program lines is an operation that allows you to see the contents of user-recorded descriptions on register lists and IO lists on the Oscillator program interface; this feature can be turned on and off, and the descriptions are not displayed when it is turned off.

Procedure



1. Open the program, click on the bottom button “More→Open Instructions”, you can see the corresponding instructions in the program after opening. The following is an example of the program after the instructions are turned on:

```
J JR[1:HOME]      'JR[1] is described in the register list as HOME
R[2]=R[1:number of times]-1  'R[2] is not described in the register list,
and R[1] is described as "times" in the register list.
DO[0: grab]=ON    'DO[0] is described in the IO list as "capture".
```

2. Open the program and click “More→Close Description”, and you cannot see the contents of the description after closing it. The following is an example of the program after closing the instructions:

```
J JR
R[2]=R[1]-1
DO[0]=ON
```



Tip:

It is not possible to change the description contents in the program line. If the register index/IO index/description contents are changed, the corresponding description contents are also changed in real time.

8.10 Displaying the current state of the IO in a program line

Instructions The function of displaying the current state of the IO in the program line refers to the function that allows you to see the real-time state of the corresponding IO variable on the Oscillator program interface, which can be turned on and off.

Procedure 1. Open the program, click the bottom button “More→Open Status”, after opening, you can see the corresponding IO status in the program. The following is an example of the program after the status is turned on:

```
DO[1:ON]=OFF      'DO[1] current state is ON, running this instruction
will adjust it to OFF
DO[2:OFF]=DI[1:ON]  'The current state of DI[1] is ON, the current state
of DO[2] is OFF, running this instruction will adjust the state of DO[2] to ON
```

2. Open the program, click the bottom button “More→Close Status”, you can not see the IO status after closing. The following is an example of the program after closing the description:

```
DO[1]=OFF
DO[2]=DI[1]
```



Tip:

- When opens the next program after one of the programs has been opened and closed, the status display of that program remains the same as the previous program, that is, all the programs that are affected by the button operation of the

previous program. It is off by default (including restarting after a power failure).
When loading a program.

- When co-existing with the IO description, follow the order of status display first, then the description.
- When the IO status changes, it needs to be updated synchronously.

8.11 Oscillator Backup Button Configuration

Instructions

The Oscillator provides 4 auxiliary buttons on the left side for user-defined key operations, which can be configured to output commands when the buttons are pressed.

There are 3 types of configurations, IO type, process package, and no configuration.

- IO type: shortcut key to operate the output IO value (when pressed, the IO value will be inverted and output according to the current state).
- Process Pack: shortcut key to open the process pack interface.
- No Configuration: disables the alternate key function.



Procedure

1. Under the authority of “Debugger”, select →Configuration→Demonstrator Configuration→Alternate Key Setting option from the main menu, and enter the interface of Alternate Key Setting.
2. Select the serial number and click the “Modify” button.
3. Click to select the “Function Type” drop-down box option (e.g., select IO type).
4. Enter the IO index number in the “DO Index” input box.
5. Click the “OK” button to complete the configuration.



Tip:

- IO type: Press the corresponding configuration auxiliary button to set the value to ON or OFF.

- Process Package: If the demonstration software has installed the process package, press the corresponding configuration auxiliary button to quickly open the process package interface (shortcut key to open the process package).
- IO auxiliary buttons can only be used in manual T1, T2 and automatic modes, and cannot be used in external modes, and the privilege should be set to “debugger” or above.

8.12 Calibrating the external shaft reduction ratio

Instructions The external axis reduction ratio calibration function is often used to calibrate the reduction ratio of an indexer for welding processes. Because of manufacturing errors in the indexer, there may be an error between the theoretical and actual values of the reduction ratio. The error will cause the accuracy of the robot and the indexer to decrease when they are controlled together, so it is necessary to calibrate and correct the reduction ratio (the external axis controls the indexer, so it is necessary to calibrate the reduction ratio of the external axis), and save the calibration results in the system axis group parameters. The calibration of the external axis reduction ratio calculates the reduction ratio of the external axis based on the center angle of the arc formed by the three reference points (P[1], P[2], and P[3]) and the actual rotational change of the position value of the indexer (difference between the encoder value of the P[3] point and the encoder value of the P[1] point). The robot arm must move the end TCP to each of the three reference points to calculate the center angle of the arc formed by the three points.

Procedure

1. Set the number of external axes (via the “Modify” button in the Main Menu→Controller Configuration→Axis Group Configuration→Number of Additional Axes, enter the number, and click “OK”).
2. Click Main Menu Selection → Put into Operation → External Shaft Reduction Ratio Calibration.
3. When the number of external axes is 1, the external axis number is 6; when the number of external axes is 2, the external axis numbers are 6 and 7; and when the number of external axes is 3, the external axis numbers are 6, 7, and 8, and the calibration is performed according to the actual selection of the corresponding number of external axes and external axis numbers.
4. Click the “Start Calibration” button to acquire the calibration points according to the calibration method.
5. After recording all the reference points, click the “Calibration” button, and the calibration result will be displayed in the pop-up box when the calibration is completed. Click “Save” button to save the calibration result to the controller.

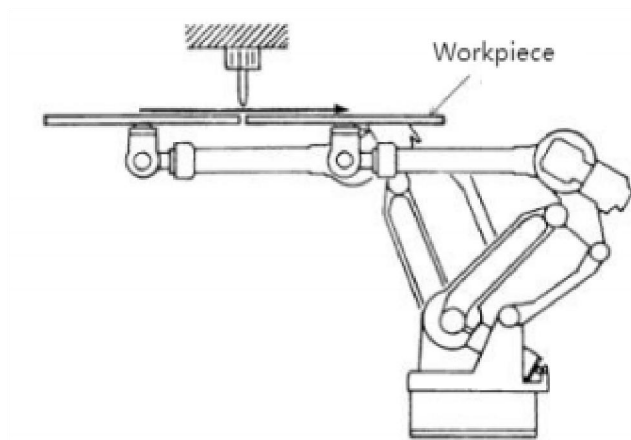
8.13 External TCP Functions

Instructions

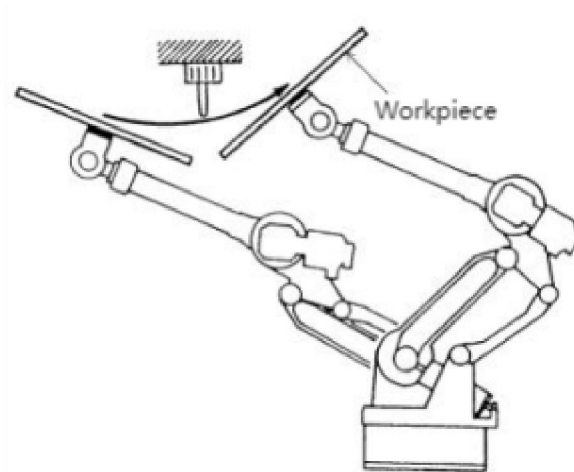
There are two general forms of robotic arms when machining objects.

1. The robot arm grips the machining object, but the center point of the tool is located at the end of the robot arm. When using the tool coordinate system, the robot arm moves along the XYZ direction of the tool coordinate system, and when rotating, it rotates around the center point of the tool, and the object rotates along with the center point at the end of the robot arm.
2. The robot arm grips the workpiece and moves it around a center point fixed on the floor (application scenarios fixed glue cans and sealing guns for gluing, fixed spot welders for electric welding). In this case, the center point of the tool is fixed on the outside of the robot arm. In this scenario, since the TCP of the arm's tool coordinate system is at the end of the flange, the rotation is around the center point of the end of the arm instead of the center point fixed on the floor, making point teaching troublesome and inaccurate.

The external TCP function is a function developed for the Form 2 scenario, where the robot is centered on the TCP fixed to the outside of the robot arm, and the robot grasps objects to move and rotate in all directions along the external TCP coordinate system. Before using the external TCP function, it is necessary to calibrate the external TCP and then call the external TCP instruction in the program.



(Upper figure)



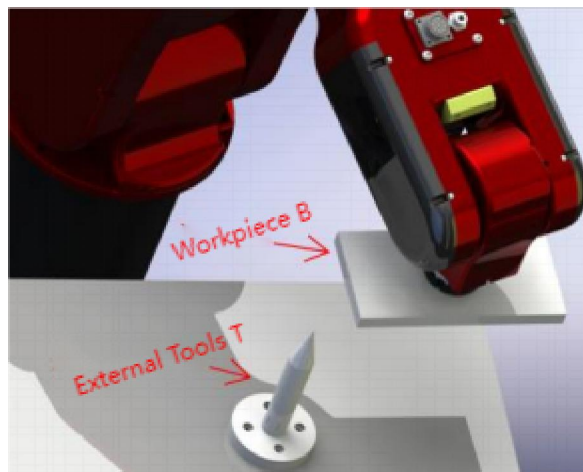
(Bottom)

Upper figure: When the attitude is maintained at a certain level, the center of the flange surface moves along a straight path.

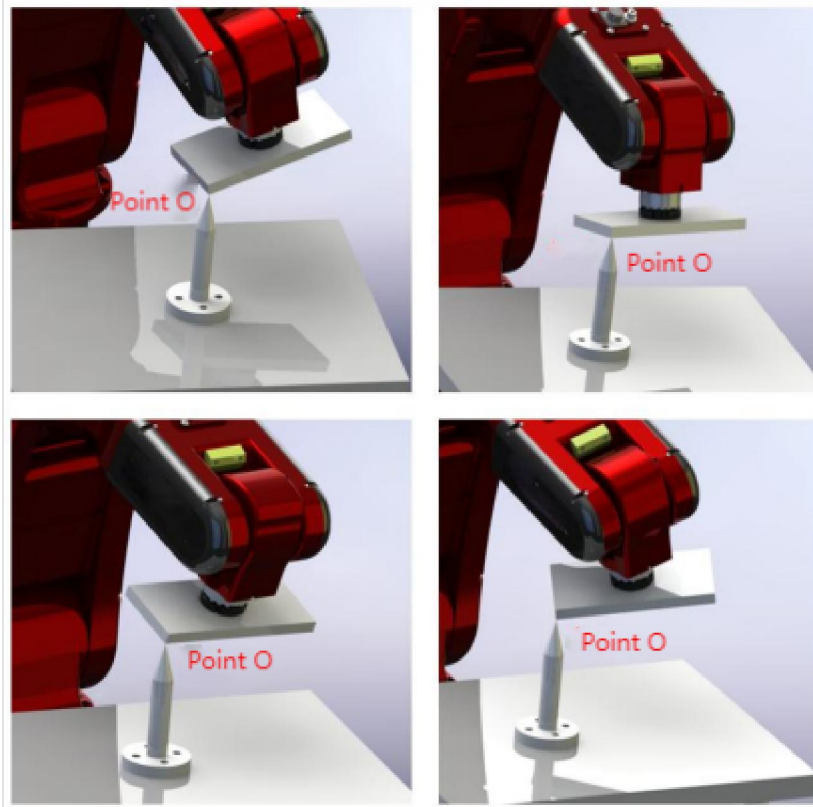
Bottom: When the attitude changes, the center of the flange surface moves along a circular path centered on the tip of the TCP.

8.13.1 Fixture Tool TCP Calibration

Procedure Prepare an external tool T and fix the workpiece B to the end of the robot.

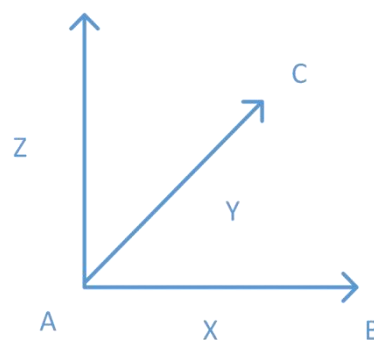


1. Select any point in the tool B for the tool 4-point calibration (as shown in the figure below, point O), the calibration is saved in the tool coordinates 0, in order to prepare for the subsequent access to the accurate world coordinate system.



8.13.2 External tool TCP calibration

Instructions Entering the calibration interface of the tool of the teach pendant, the user selects “external tool” as the tool type, and the acquisition of coordinates should be based on the first step of the calibration (e.g., tool 0) to be carried out. According to the order shown in the diagram, place the tool to the three points of ABC, record the position of the robot arm under the three points (click the [Get Coordinates] button), according to the three points of ABC to determine the coordinate system of the external tool TCP (the origin of the coordinate system is A, the X direction is AB, and the Y direction is AC). Finally, click [Calibration] and save.



Procedure

1. Select → Input operation → User tool calibration in the menu.
2. Select the user tool number to be calibrated and you can set the user tool name.
3. Click [User Tool Type] to select the external tool.
4. Click the [Start Calibration] button.
5. Move to the calibrated reference point 1 and click [Reference Point 1] to obtain the

coordinate record coordinates.

6. Move to the calibrated reference point 2 and click [Reference Point 2] to get the coordinate record coordinates.

7. Move to the calibrated reference point 3, click [Reference Point 3], get the coordinate record coordinates.

8. Click the [Calibration] button to make sure the program calculates the calibration coordinates.

9. Click the [Save] button to store the calibration value of the external tool coordinates.

8.13.3 Demonstration and external TCP commands

Demonstration of the procedure Set “IPO Mode Selection” to External Tool in the “Select Base Coordinate/Tool” of the Teach-In, move the XYZABC axis of the robot arm to teach and record the point

Instructions

position, which can be saved in the position variable P, JR/LR register of the program.

Use EX_UTOOL=ON, then call the UTOOL_NUM command to specify the value of

the external TCP.



Program Example:

```
EX UTOOL =ON
```

```
UTOOL_NUM =1
```

```
L P[1]
```

```
L P[2]
```

Set EX_UTOOL = OFF when turning off external TCP command calls.

8.14 Speed Configuration

8.14.1 Safe Speed Trigger and Recovery

Instructions

In practical applications, the robot automatically runs fast inside the safety fence, when the safety fence door is opened to enter the fence, in order to safely produce, the running speed of the robotic arm needs to be reduced to guard against unforeseen circumstances, when the fence door is closed, the robotic arm can be allowed to automatically return to the original running speed for production, or not automatically return to the running speed, but manually operated by the oscillator multiplier button for speed Multiplier modification.



- Speed multiplier after triggering, i.e., after configuring the corresponding IO signal to take effect, safe rate input value range [1, 100], configuring the corresponding IO signal to take effect or, if the robot motion rate is larger than the set safe rate value, it will automatically switch to the set safe speed.
- Automatically restore the original speed multiplier switch, on for the corresponding IO signal off, the robot will restore the original motion multiplier for motion. Off, the robot keeps the speed multiplication after the trigger signal for simulation.



Instructions for turning on the automatic restoration of the original speed multiplier function:

Prerequisites: Configure the speed multiplier $V_1=10\%$ after triggering, turn on the self-recovery function, and also configure the safety signal as: DI[200], V is the current speed.

Situation one:

DI[200]=ON, $V_2=100\%$

DI[200]=OFF 'Safe Speed Trigger in effect $V=V_1=10\%$

DI[200]=ON ' $V=V_2=100\%$ (speed recovery)

Situation two:

DI[200]=ON, ' $V_2=100\%$

DI[200]=OFF ' $V=V_1=10\%$

'Modified multiplier, $V_3=50\%$

DI[200]=ON 'Velocity recovery $V=V_2=100\%$



SR6-600

关 空闲 T1

100%

DEF DEF DEF 空载

DEF DEF DEF 空载

0

0

0

信息确认

报警确认

A1

A2

A3

A4

X

17:23

STOP

X

数字输入/输出端

IO号	值	状态	说明
25	○	REAL	
26	○	REAL	iSAFE_SPEED
27	○	REAL	
28	○	REAL	
29	○	REAL	
30	○	REAL	

输入端

输出端

+100

-100

切换

刷新

值

说明

保存

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引号
iCOLLA_MODE		22
iSAFE_SPEED	26	23
iSAFE_DOOR		24
iSHARE_EN[0]		25
iSHARE_EN[1]		27
iSHARE_EN[2]		28
iSHARE_EN[3]		29
iSHARE_EN[4]		30
iSHARE_EN[5]		31
iSHARE_EN[6]		32

移除

增加

保存

刷新

A1

A2

A3

A4

X

do[1]

运行配置

标志位	IO索引	DI索引

other operations to change the speed multiplier Vord, users can still change the speed multiplier via the Oscillator change multiplier or VORD command.

- In the case of multi-axis groups, only the speed multiplier change of the robot group is supported, and the case of external axis groups is not considered for the time being.
- The speed multiplier set when running the program should be lower than the current actual running speed, if the set speed is higher than the running speed, it will not respond.

Procedure

1. in the debugger above the user group.
2. Select “ Configuration → Controller Configuration → Speed Configuration” in the menu to enter the configuration page.
3. Set the parameters of “Speed multiplier after triggering” and “Whether to restore the original speed multiplier automatically” and save them. Select “Configuration Controller Configuration Running Configuration” in the menu, and select the corresponding DI index number to match the “iSAFE_SPEED” flag signal.

8.14.2 Reset Speed Trigger and Recovery

instructions

The reset speed function is that after clicking [Acknowledge Alarm] to reset the robotic arm after the system alarm, the system will automatically adjust the current multiplier to the set value.



- When configuring the function, you need to click the “Alarm Acknowledgement” button after the system throws an alarm for the function to take effect; if the system does not throw an alarm and clicks the “Alarm Acknowledgement” button directly, the function does not take effect.
- This function needs to configure the speed multiplier of the system after the alarm is acknowledged, and the configuration information is saved in the system in the form of a file.



Tip:

- The system default multiplier value for reset speed is 10.
- The Reset Speed Trigger function only works in Auto, External mode.

- Reset Speed Trigger does not have a mutually exclusive relationship with any other change of speed multiplier VORD operation, the user can still change the multiplier from the Teach-In or via the VORD command.
- When the speed is triggered, the teach pendant throws a message “Reset Speed Triggered” .
- Before triggering the reset speed, the current speed multiplier of the robot needs to be determined:
- If the current speed multiplier is greater than the configured reset speed multiplier, the function trigger can take effect after the function is configured, i.e. the reset speed can be triggered.
- Otherwise, the function trigger cannot take effect after the function is configured, that is, the reset speed cannot be triggered, and there is no error
- message.
- For the multi-axis group case, only the speed multiplier change of the robot group is supported, and the external axis group case is not considered for the time being.

Procedure

1. User groups above the debugger.
2. select the menu → Configuration → Controller Configuration → Speed Configuration, below the “reset speed” tab to enter the configuration page.
3. Set the parameter “Speed multiplier after alarm confirmation” and save it.
4. 4Select Function Enable to turn on the function.

8.14.3 Signal Speed Configuration

Instructions

In practice, the robotic arm will be enabled to run at different multipliers according to different application scenarios, and this feature supports user-defined configuration of the signal speed.



- The function needs to be enabled after the function is configured, otherwise the signal speed trigger function will not take effect.
- The function also needs to be configured with the following options:
- Trigger signal and trigger mode (ON/OFF)
- Speed multiplier of system operation after the signal is triggered
- Function Enable
- Function triggering uses a signal edge mechanism, where the configured IO signal edge triggers the robot arm to adjust the operating speed multiplier to the set value.
- Configuration information is saved as a file on the system.



Configuration note: “R[N].X=OFF/ON with value V”

- N is the R register number;
- X is the number of bits, also refers to the number of binary bits. The user can set the value of R[N] register, and the corresponding value data is converted from decimal to binary to control the output of signal speed;
- OFF/ON means signal switch;
- V is the multiplication rate;

Prerequisite: the corresponding signal speed has been configured

	配置项	值
1	R[150].1=ON	20

Scenario 1: Modify the signal configuration according to the instruction

‘Assuming the current multiplier is 50

R[150].1=OFF ‘Current speed multiplier is unchanged

R[150].1=ON ‘Current speed multiplier set to 20

Scenario 2: Modify the value of the R register in response to the signal speed

‘Assuming the current multiplier is 50

R[150]=60 ‘The current speed multiplier is unchanged (register data is 60, decimal to binary i.e. 0011 1100, the first bit is 0, i.e. OFF)

R[150]=70 ‘Current speed multiplier is set to 20 (register data is 70, decimal to binary i.e. 0100 0110, bit 1 is 1, i.e. ON)



Tip:

- The IO trigger signal can be a virtual signal.
- The signal speed trigger function only works in automatic, external mode.
- The IO trigger to change the speed multiplier is not mutually exclusive with any other operation to change the speed multiplier VORD, and the user can still change the multiplier from the Teach-In or via the VORD command.
- When the speed is triggered, the teach pendant will throw the message “Signal speed has been triggered”.
- For the multi-axis group case, only the speed multiplier of the robot group is supported, and the case of external axis group is not considered for the time being.

- Procedure**
1. in the debugger above the user group.
 2. Select menu→ Configuration→ Controller Configuration→ Speed Configuration, and enter the configuration page of “Signal Speed” tab at the bottom
 3. Set the “Signal Speed Configuration Item List”, click the corresponding list, and then select [Change] button to enter the pop-up window of signal configuration.

4. You can input [0,999] for the signal configuration register, [0,7] for the second input box, and [1,100] for the value of multiplier speed
5. Select Function Enable to turn on the function.

8.15 Path Keeping Function

Instructions	Path keeping function refers to the program pause to move the robot arm, in the start of the program run, the robot arm will first return to the position of the pause, and then the movement of the function. This function is turned on at the same time the program can be moved after the pause of the current implementation of the pointer, in the start of the program run, the robotic arm will move from the current position to the new pointer line specified in the target point, this time the trajectory of the change is relatively large, in order to prevent collision needs to be prompted to the user to confirm whether to start the start of the program execution.

Procedure



- 1 、 Select →Configuration→Controller Configuration→Path Keeping Function in the menu to enter the configuration page.
- 2 、 After setting the input position deviation allowable value, click Save to save the data to the controller, the position deviation value defaults to 50mm, and the attitude deviation angle defaults to 5°.
- 3 、 Select Function Enable to turn on the function.



Tip:

- The user needs to pay attention to the environment around the robot arm to avoid collision and other safety hazards during the path retracing process.

8.15.1 Execute the pointer change alert function

Instructions

After the program pause can move/jump the current execution pointer, in the start of the program run, the robotic arm will move from the current position to the target point specified in the new pointer line, there is the possibility of a large change in trajectory, in order to prevent collision in the start of the program prompts the user to confirm that whether or not to jump to the corresponding pointer to run the program.

Procedure

1. When the main program is paused, select other instructions and click [Jump] to jump the pointer position;
2. Users click the main program run button, the current pointer exists change pop-up window prompt:



3. Clicking the “OK” button starts program execution from the current pointer line; clicking the “Cancel” button does not start program execution.



Tip:

Please confirm the current position of the robotic arm and the point data of the current pointer pointing to the motion instruction after determining the pointer jump, and pay attention to the environment around the robotic arm to avoid collision and other safety hazards.

8.16 Follower function

8.16.1 Introduction to Functions

The follow function supports different application scenarios.

- (1) Scenario 1: Thermal moxibustion application.

Under the requirement of maintaining the height of the moxa head and the surface of the human body unchanged, during the movement of the robotic arm, the laser sensor measures the height value H of the moxa head from the surface of the human body, and then subtracts the laser sensor value h at the moment of the start of the follow-through, and the difference between the two ($H-h$) is sent down to the specified LR register as the compensation value. The compensated value is to compensate the current point value of TCP under the coordinate system of tool and workpiece.

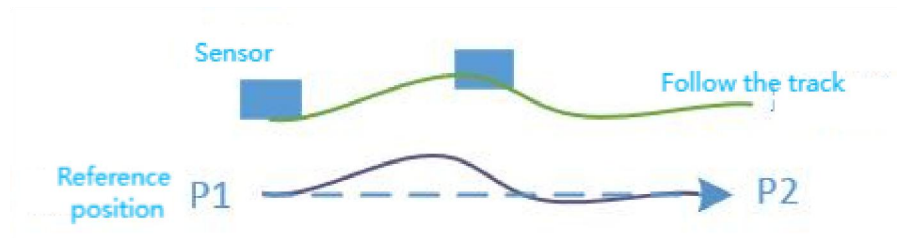


图 1-1 Schematic trajectory of thermal moxibustion application

- (2) Scenario 2: Welding arc tracking application.

When the arm uses the pendulum welding function, the pendulum welding trajectory can be shifted along the Y-axis aspect of the pendulum welding coordinate system as it moves along the target point (instead of compensating for points in the current tool workpiece coordinate system). The compensation values can be values in the six dimensions XYZABC.

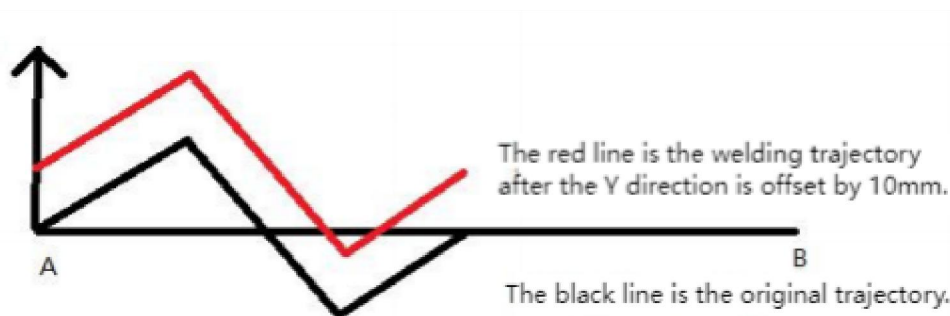


图 1-2 Welding application trajectory schematic

- (3) Scenario 3: Constant force sanding application.

Offline/demonstration programming generates the original motion trajectory, in the process of executing the original trajectory, the torque sensor senses the force in each direction in real time, the force control

algorithm calculates the compensation amount in each direction of XYZABC for each interpolation cycle and then sends it to the LR register, and the system superimposes the compensation amount during the interpolation. The compensation amount is in the form of accumulation, such as the first cycle of the force control algorithm sends the compensation value A1, then the robotic arm moves to P+A1; the second cycle of the force control algorithm sends the compensation value A2, then the robotic arm will move from the current position (P+A1), to the position of P+A1+A2.

(4) Scenario 4: Laser tracking application for interface scenarios.

After the host computer sends a point position through the interface, before the robot moves completely to the target point, the host computer updates a new point position, and the robot system moves according to the new point position without stopping in the middle.

8.16.2 Follow-on commands

The instruction format is as follows:

SLAVE MODE=1 CYCLE=2 LR=0 MASK=255

SLAVE: Indicates the statement is a follow function statement.

MODE: It means the function is open or closed. 1 means open the follower, when the follower function is open, the current position is used as the reference; 0 means close the follower.

CYCLE: It means the period of compensation data sampling, the value means the multiple of interpolation period. For example, CYCLE=2 means $4 \times 2 = 8\text{ms}$, and the current compensation data value is sent every 8ms.

LR: Indicates the LR register index value of the real-time compensation data value, the user writes the real-time calculated compensation value into this LR register, and the system carries out the follow function according to this value.

MASK: indicates the mask value. Since the requirements of each application site of the follow function are different, the function can be distinguished by the MASK value, and the meaning of the follow function is different under different MASK values. Specific MASK value meanings can be

found below.

Program Cases:

```
<program>
UTOOL_NUM = 1
    L P[0] VEL=150 ACC=200 DEC=200 CNT = 0      'Movement to the starting point, using the
                                                current point
as the reference point
SLAVE MODE=1 CYCLE=4 LR=0 MASK=255      'Turn on the
```

```

LP[1]
LP[2]
C P[3] P[4]
...
GOTO LBL[1]
SLAVE MODE=0 CYCLE=4 LR=0 MASK=255      'Turn off the follower
<end>

```

8.16.3 Compensation Functions and MASK Options

There are two main functional points of follower compensation:

- (1) Compensation direction: compensation along the base coordinate or workpiece coordinate system, compensation along the tool coordinate system, compensation along the swing coordinate system.
- (2) Compensation value: Compensation by relative or absolute value.

The following describes the functions of this function for different values of MASK.

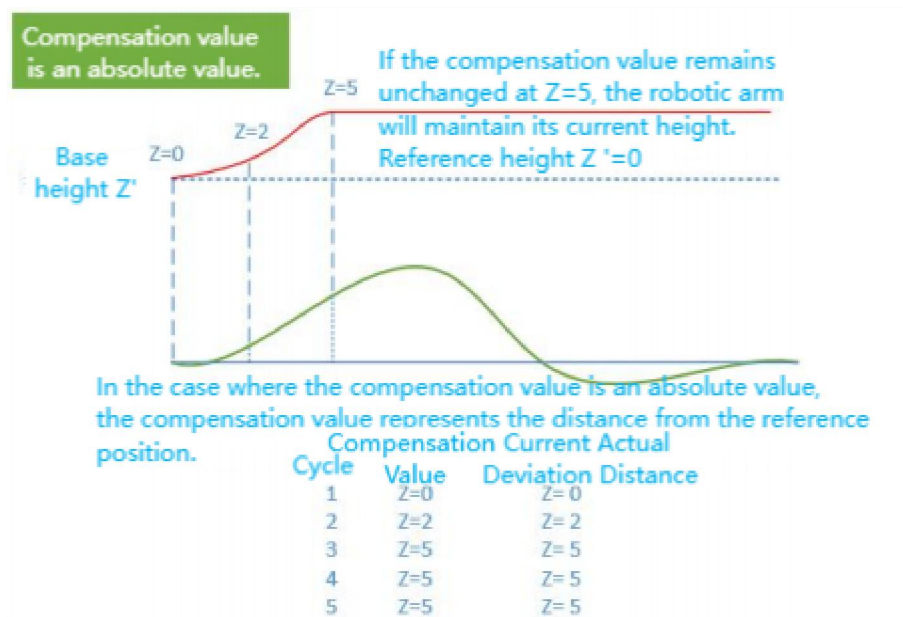
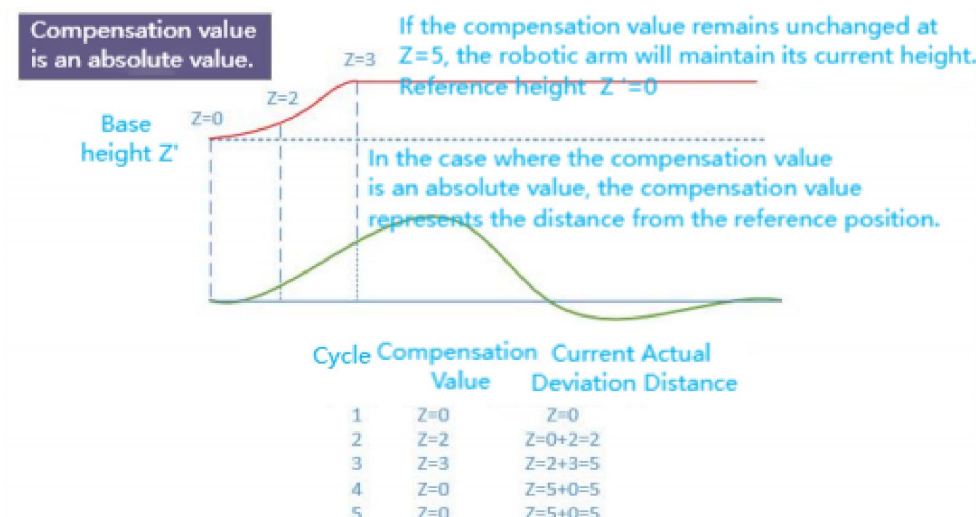


图 2-1 Explanation when the compensation value is an absolute value



2-2 Instructions when the compensation value is a relative value

(1) MASK=256

Compensation direction: Compensation along the direction of the pendulum welding coordinate system.

Meaning of compensation value: Absolute value of the change in the reference position relative to the pendulum coordinate system.

Function Instructions: real-time compensation of the value of the pendulum welding coordinate system, so that the pendulum welding trajectory for the overall offset. For example, when $LR[0]=\{0, 10, 0, 0, 0, 0, 0\}$, the original pendulum welding trajectory is shifted by 10mm along the Y-axis of the pendulum welding coordinate system as a whole, and the compensation value can be maintained unchanged when the specified follower position is reached. When the follower is turned off, the robot arm will keep the current position of the pendulum welding coordinate system. The arm will acquire the current value for compensation in real time while it is moving.

Available application scenarios: Arc welding tracking welding applications, etc.

Note: This feature must be used in conjunction with the internal interface to pass in the values of the pendulum welding coordinate system for each track segment, and is currently only integrated for use by internal developers in the welding team.

(2) MASK=255

Compensation direction: offset compensation along the direction of the tool coordinate system.

Compensation Value Meaning: the compensation value of tool TCP point, it is absolute value.

Function Instructions: Compensate the value of the end TCP point, if $LR[0]=\{0, 0, 20, 0, 0, 0, 0\}$, then the robot arm moves 20mm along the Z-axis (the center axis of the torch) of the tool coordinate system to compensate, in every interpolation cycle, the system will get the latest value of the compensation data $LR[0]$, and then move towards the compensation point, if the compensation value is If the compensation value is unchanged, the robot arm

will keep the current height and move toward the target point after reaching the compensation position.

Available application scenarios: laser welding applications, etc.

(3) MASK=254

Compensation direction: Compensation along the direction of the base coordinate or work coordinate system.

Meaning of compensation value: the absolute value of change relative to the base position.

Function Instructions: The position when the follower is turned on is the base position, and the compensation value corresponding to the base position is 0. If you write data into the compensation register, such as $LR[0]=\{0,0,30,0,0,0,0\}$, the robotic arm will move along the Z-axis of the workpiece coordinate system by 30mm under the current workpiece tooling coordinate system to perform the compensation, and in each interpolation cycle, the system will get the latest compensation data LR[0] value, and then move towards the compensation point. If the compensation value remains unchanged, the robot arm will keep the current height and move towards the target point after reaching the compensation position.

Available application scenarios: thermal moxibustion applications, etc.

(4) MASK=253

Compensation direction: offset compensation along the direction of the tool coordinate system.

Compensation Value: Relative value based on the last position change.

Function Instructions: The compensation value acts on the value of the end TCP point position, but the data of the compensation value requires the position data of each interpolation cycle.

① Application 1: For example, in the application of constant force grinding, the moment sensor senses the force in each direction in real time, and then the force control algorithm calculates the compensation amount of XYZABC six directions in each interpolation cycle and then sends it to the LR register, and the system superimposes the compensation amount in the interpolation. The compensation amount is in the form of accumulation, for example, if the first cycle force control algorithm sends compensation value A1, then the robot arm moves to $P+A1$; if the second cycle force control algorithm sends compensation value A2, then the robot arm will move from the current position ($P+A1$), to the position of $P+A1+A2$. The compensation position will be interpolated in one cycle, if the compensation value is too large, resulting in an interpolation cycle can not move to the point, then the alarm will be overspeed or over-acceleration, only the value of the compensation data LR [0] becomes 0, then the robotic arm will be maintained in the current position.

② Application 2: Constant force grinding in Z-direction, the moment sensor real-time according to the perceived force in Z-direction, real-time Z-axis force data sent to the LR register, the system will be

based on the actual force and the desired constant force to do a comparison, real-time adjustment of the Z-axis follower compensation value, so that the force is stably controlled in the desired force between the upper and lower floating 1N. The following position is also in the form of absolute value, for example, according to the current force calculation needs to move to the point A, then move to the position A, the next interpolation cycle calculation needs to move to the point B, then move to the B position.

Usable application scenarios: constant force grinding applications, etc.

In this type of application, the constant force algorithm is highly demanding, requiring discrete interpolation of errors and systematic superimposition of its downward position data. In this type of application, the position is sent in conjunction with codesys or the system's internal interface.

8.16.4 Follower parameters

The follow function must have some function configuration parameters set correctly in order to be used correctly. The name of the parameter file to be set is: group[0]_follow.data The contents of the file are as follows:

```

1  group_st:{
2      Follow_Up:{
3          FirLength:0,
4          Pmax:[200.00, 200.00, 200.00, 60.00, 60.000, 60.000, ],
5          Pmin:[-200.00, -200.00, -200.00, -60.00, -60.00, -60.00, ],
6          Vmax:[300.000, 300.00, 300.00, 60.00, 60.00, 60.00, ],
7          Amax:[4000.00, 4000.00, 4000.00, 300.00, 300.00, 300.00, ],
8          Jerk:[10000.00, 10000.00, 10000.00, 720.00, 720.00, 720.00, ],
9          FollowTool:[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, ],
10         LimitFlu:[0.40, 0.40, 0.40, 0.40, 0.40, 0.40, ],
11     },
12 },

```

Figure 2-3 Contents of the group[0]_follow.data file

The meaning of each parameter is as follows:

Pmax: Positive limit value for the six directions of XYZABC of the following compensation displacement amount.

Pmin: the negative limit value of the six directions of XYZABC for the amount of random compensation displacement.

Note: If Pmax[x] and Pmin[x] are 0 at the same time, it means that the follower in this dimension is closed. Users can set the limit values according to the actual engineering needs to provide safety protection.

Vmax: the maximum value of follower motion speed, can not be 0, this value is set too small may lead to slow follower response.

Amax: the maximum value of the acceleration of the following motion, can not be 0, this value is set too small may lead to the slow response of the following motion.

Jerk: the limitation value of acceleration of follower motion, can't be set to 0. Setting this value too

small may lead to slow response of follower motion.

Note: Users can adjust the speed parameter according to the actual engineering needs.

FollowTool: following motion to coordinate system value, used to prevent the current use of the to coordinate system to change the cause of possible anomalies, the parameter is not in effect, that is, allow the use of different to coordinate system.

LimitFlu: the minimum value of the following wavelength limitation, the change of the following compensation is not processed when it is less than this value, so as to reduce the jitter.

Through the workstation software HSR-Studio, in the menu bar-[Controller]-[Terminal Panel] function interface, use the following commands to set and query the follower parameters:

```
group[0].Follow_Up.Vmax[2]=150 'Set the Vmax[2] parameter to 150
```

```
group[0].Follow_Up 'query all parameters'
```

```
var.saveGp(0, "follow") 'Save the parameter file
```

8.17 Craft Kits

Instructions	AUCTECH Robotics supports belt line vision tracking, as well as palletizing, painting, welding and other processes. The relevant functions are added to the system in the form of process packages, for more information and use, please contact the relevant personnel.

8.18 Offline programming and PC debugging

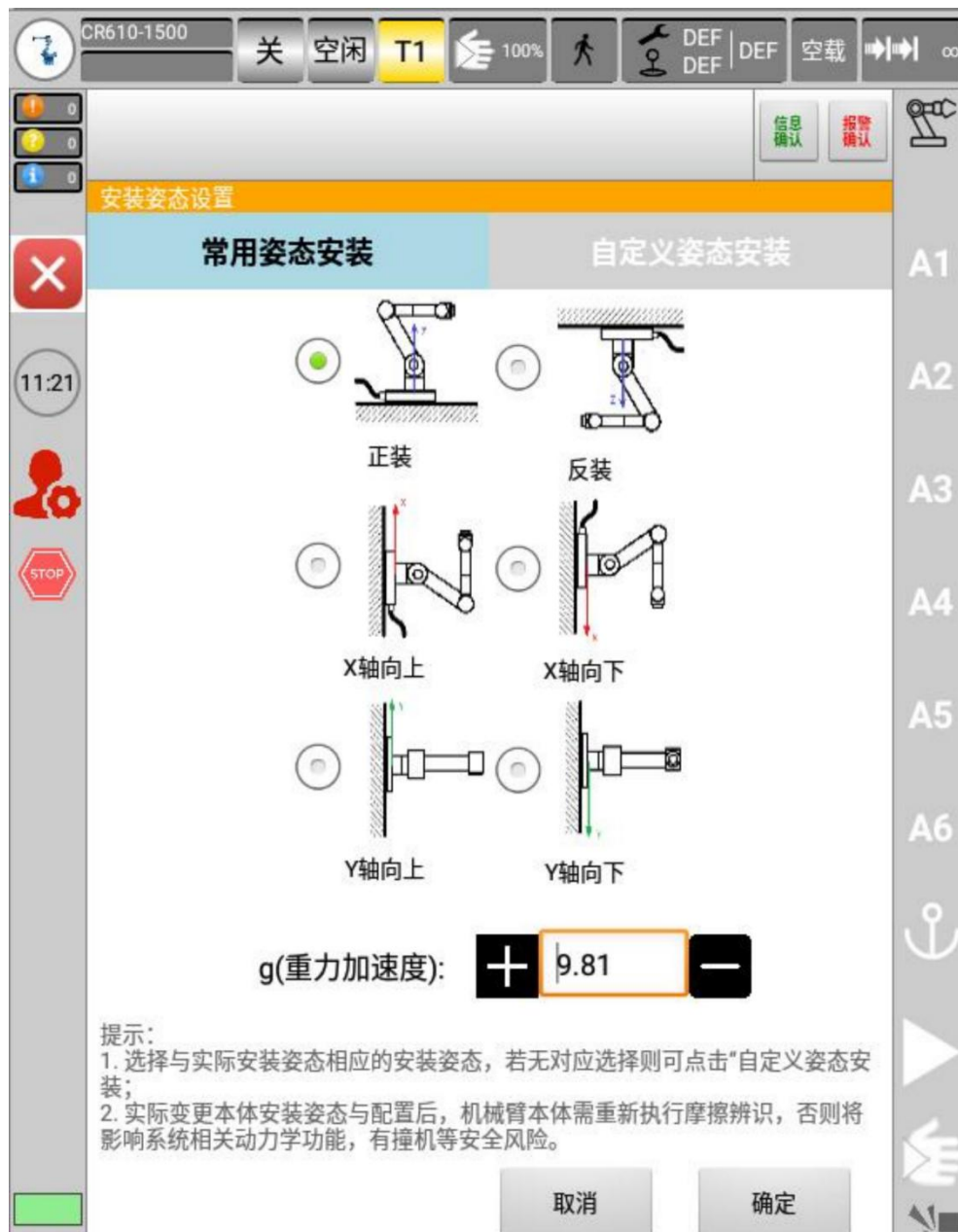
Instructions	AUCTECH Robotics supports offline programming function. Offline programming refers to the process of using offline programming software on a PC computer to generate the motion program of the robot arm according to the robot arm model and the imported processing object model. Offline programming can simplify some complex trajectory teaching programming tasks, often combined with the small line interpolation function, to achieve the effect of rapid completion of the teaching task. If you need to use offline programming software, please contact the relevant personnel. AUCTECH Robotics provides PC development and debugging software, which can realize controlling the JOG movement of robotic arm, writing PRG program, simulating and checking the running trajectory, etc. on PC computer, if you need to use PC development and debugging software, please contact the relevant personnel.

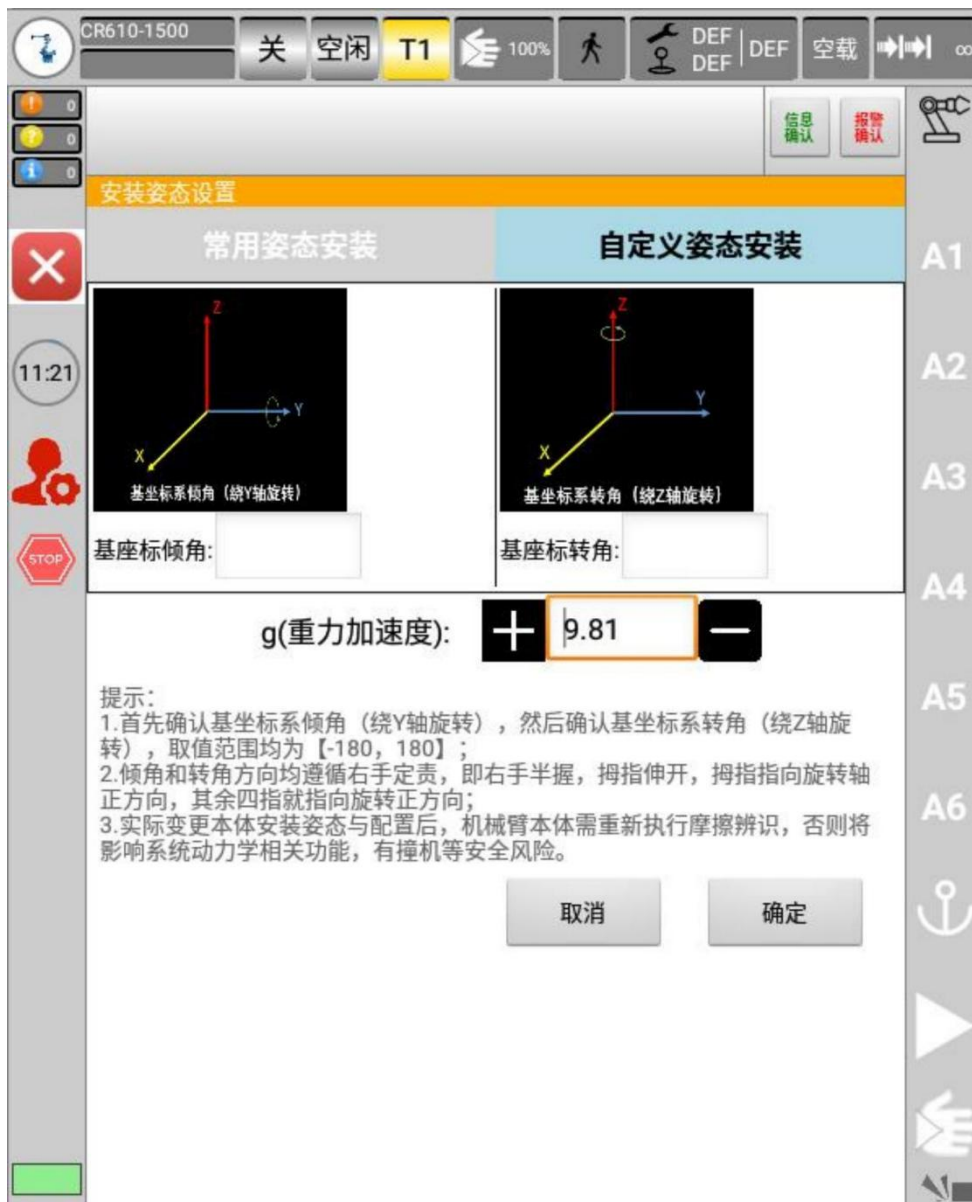
9 Dynamics Functions

9.1 Mounting attitude setting

Instructions

Normally the arm is mounted on a smooth table or floor, in which case no operation is required. If the arm is mounted in reverse, wall-mounted or side-mounted, and dynamics-related functions are used, the current mounting attitude needs to be set. The system provides the following configuration page.





Steps of operation

1. Select → Configuration → Dynamics Function → Mounting Attitude Setting in the menu to enter the setting page.
2. Select or set the mounting attitude according to the actual robot mounting attitude. You can first check whether there is a corresponding installation attitude in the “Common Attitude Installation” interface. If there is no corresponding installation attitude option, you can click to enter the “Custom Attitude Installation” interface to set up the custom attitude installation.
3. g is the local gravitational acceleration.
4. Click “OK” button to save the data to the system and restart the controller to take effect.

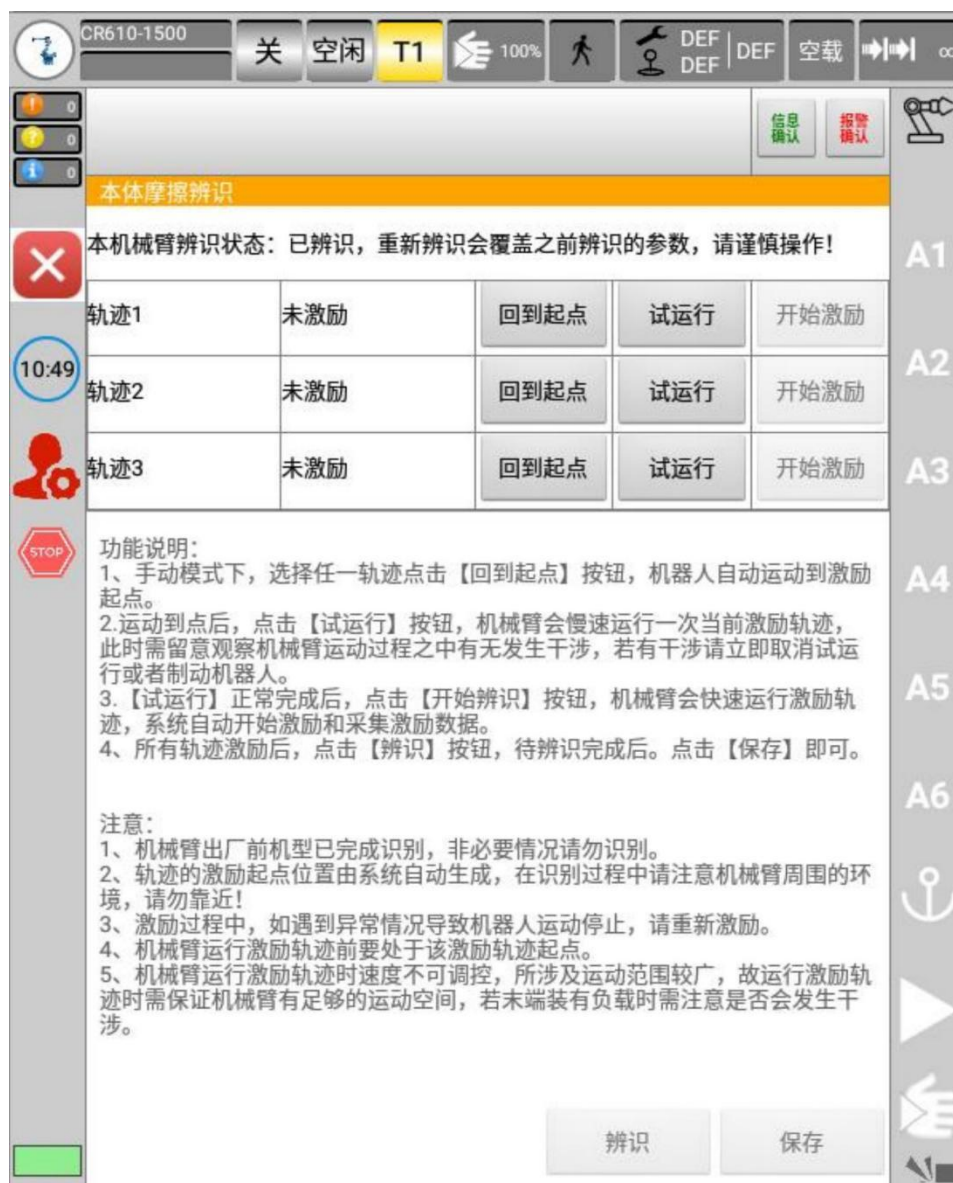
**Tips:**

1. Setting the mounting attitude is to improve the accuracy of the kinetic model and facilitate the assurance of the accuracy of the kinetic model. If you need to use it inverted, you should set the mounting attitude first when you install it inverted, and then recognize it. The safety attitude should also be set correctly before using the dynamics function.
2. After changing the mounting attitude, the arm body has to perform friction recognition again! Failure to do so will affect the function of the robotic arm system and can lead to safety risks such as crashes!

9.2 Ontology Friction Recognition Function

Instructions

Robot dynamics parameters are critical to the accuracy and reliability of the robot, and each parameter needs to be configured in the control system before it leaves the factory, without providing a recognition process to the outside world.



Steps of operation

1. Select → Configuration → Dynamics Function → Body Friction Recognition in the menu bar to enter the Body Friction Recognition page.
2. In manual mode, select any trajectory and click [Back to Start] button, the robot automatically moves to the excitation starting point.
3. After moving to the point, click [Trial Run] button, the system will run the incentive trajectory once at slow speed.
4. After the trial run ends normally, click the [Start Recognition] button, the system automatically starts to stimulate and collect stimulation data.
5. Same as the above steps to complete the remaining trajectory excitation.
6. After all the trajectories are stimulated, click the [Recognition] button, the system starts to recognize the dynamics parameters.
7. Click the [Save] button to save the recognition data into the controller.

**Attention:**

- The robot arm has been recognized before shipment, so please do not recognize it if it is not necessary.
 - The excitation starting position and track point position of the trajectory are automatically generated by the system. During the identification process, please pay attention to the environment around the robotic arm, pay attention to the possibility of motion interference, and please do not approach.
 - The excitation starting position of the trajectory is automatically generated by the system.
 - You can move to the starting point of the trajectory only in manual mode, and the trajectory can be stimulated in auto/manual mode.
 - During the excitation process, if the robot movement stops due to abnormal conditions (e.g. disabling, emergency stop, error, clicking the “stop excitation” button, etc.), the excitation fails and the current trajectory excitation data is cleared.
 - If the model does not perform this function of ontology recognition, the
-

Supported models

collaboration-related functions cannot be used.

Not all models are dynamically adapted, please ask the relevant technician for details.

9.3 Load Configuration Functions

Instructions

Load configuration is a function that lets the system know the current load on the robot by modifying the dynamics-related parameters, which can be obtained through self-entry or load recognition. Correctly setting the load and load parameters can improve the performance of the robot and effectively utilize the robot's dynamics-related functions, including collision detection and drag teaching functions. To avoid errors or undesirable effects, the robot should be load recognized and the load number selected before use.

9.3.1 Dynamic load switching

Instructions

Dynamic load switching is a function that allows the robot to reacquire the relevant load parameters under the current number by switching the load number, so as to know the current load situation of the robot, the system has 16 groups of load number data. Load switching can be accomplished in two ways, namely, directly through the teach pendant button and through the command to realize the switching.

Steps of operation

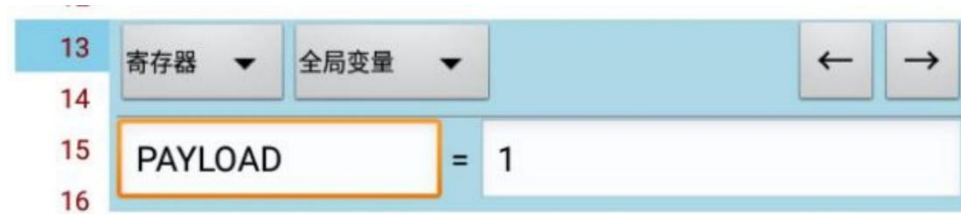
Mode one:



Click on the robot speed level or load status display box in the status bar of the Teach-Instructor, and when it is currently a kinetic load parameter, you can click on it to select the corresponding load number in the Load Number Selection drop-down box.

Mode 2: Set in the program by PAYLOAD assignment instruction.

1. Select the upper line of the instruction line to be inserted.
2. Select Command → Dynamics Command.
- 3, slide the drop-down box options to find the “PLAYLOAD ” option and click to select.
- 4, click the cursor to locate the right input box, and enter the value.
- 5, click “OK ” button to add the instruction is complete.



程序 Example:

PAYLOAD= 1 ‘Specify the current load with the number 1, which can be set within the range of -1 to 15. The default load is -1 empty load.



Attention:

- When the robot load changes, it should be switched to the corresponding load number in a timely manner, otherwise it may lead to a decrease in motion accuracy, false alarms of collision detection, or have an adverse effect on the drag force.
- The load number switching strategy for loading and unloading programs is the same as the tool artifact switching, and the current load after unloading the program is the same as the load used before loading.

9.3.2 Load Parameter Configuration

Instructions

Load parameter configuration is the setting of the load's mass, center of mass position, inertia tensor, and other parameters. There are two ways to operate to configure the load parameters as follows:

1. Load parameter configuration by manual editing: If the load parameter information is known, you can directly modify the load parameter value and save it to complete the load parameter configuration.

2. Load parameter configuration by manual editing: If the load parameter information is known, you can modify the load parameter value directly, save it and then finish the load parameter configuration.

Steps of operation for mode 1

Way 1: Load parameterization by manual editing



1. Select Configuration → Dynamics → Load Configuration in the menu bar.
2. Click the “Load No. Selection” drop-down box, select the corresponding load number to enter the parameter page (after selection, the information displayed on the page is the load parameter information of the load number, which is not the load number currently used by the robot).
3. Click the “Instructions” input box, you can make comments on the load number Instructions.
4. Select the corresponding load parameter input box, and fill in the corresponding load parameter value manually according to the actual load data.
5. Click the “Modify” button, and then click the pop-up “Save” button to save the modification result and complete the load parameter configuration.



Warning:

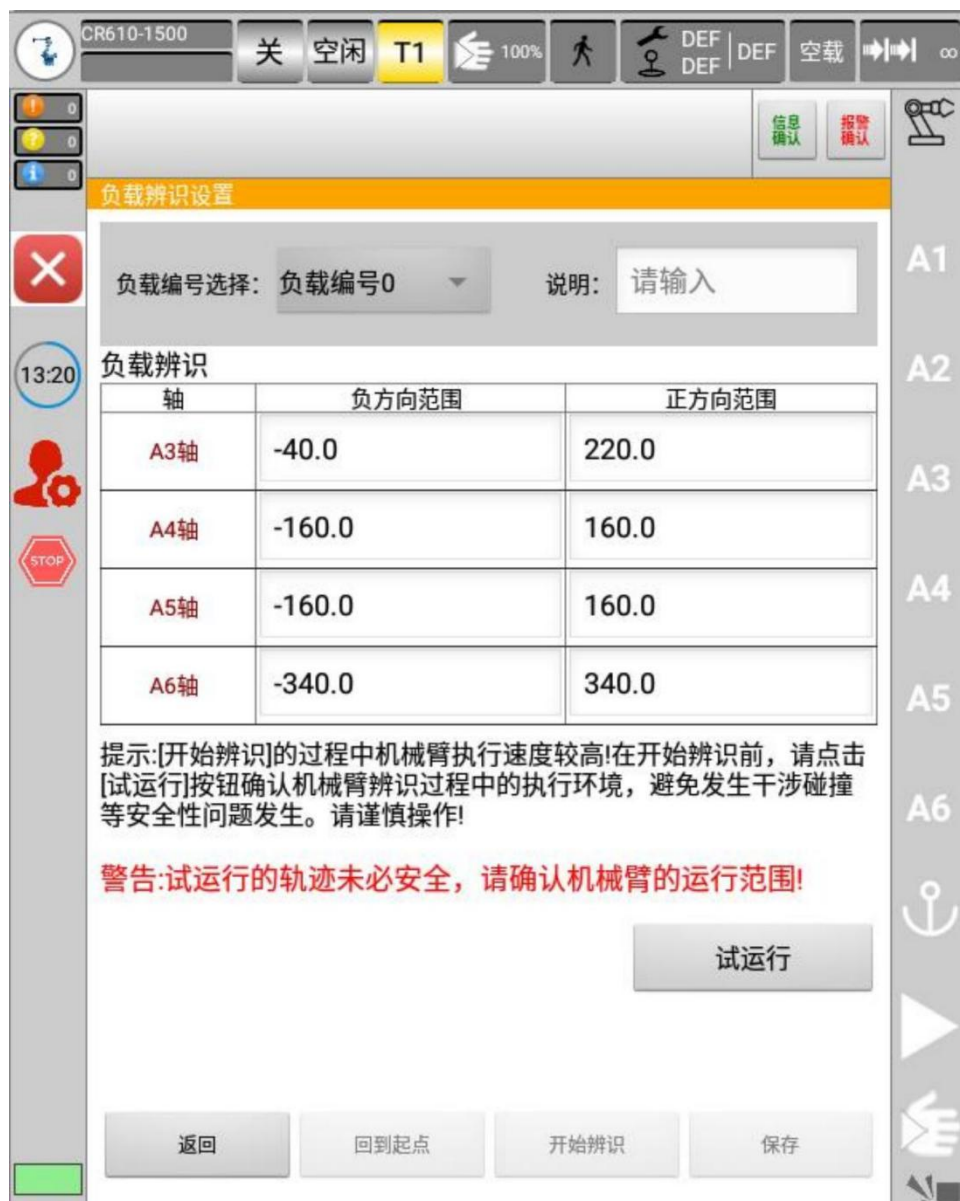
Since the BR robot is a model with internal rotation structure, users should pay attention to the following two cases if they manually modify the load recognition parameters:

1. Avoid robot third axis articulated axis close to 270° position
2. The robot's fourth axis articulated axis should be in the range of $0^{\circ} \sim 180^{\circ}$.

If the model is loaded, it is recommended to use the first solution. And pay attention to avoid the third axis close to 270° resulting in the presence of load and body interference.

Steps of Operation in Method 2

Mode 2: Automatic generation of load parameter information through load recognition.



CR610-1500 关 空闲 T1 100% DEF DEF 空载

信息确认 报警确认

负载辨识设置

负载编号选择: 负载编号0 说明: 请输入

负载辨识

轴	负方向范围	正方向范围
A3轴	-40.0	220.0
A4轴	-160.0	160.0
A5轴	-160.0	160.0
A6轴	-340.0	340.0

提示:[开始辨识]的过程中机械臂执行速度较高!在开始辨识前,请点击[试运行]按钮确认机械臂辨识过程中的执行环境,避免发生干涉碰撞等安全性问题发生。请谨慎操作!

警告:试运行的轨迹未必安全,请确认机械臂的运行范围!

试运行

返回 回到起点 开始辨识 保存

1. Select Configuration → Dynamics → Load Configuration in the menu bar.

2. Click the “Load No. ” drop-down box to select the corresponding load number (based on which load identification is performed).
3. Click “Enter Automatic Load Recognition” button to enter the load recognition setting interface.
4. Click “Negative Direction Range” or “Positive Direction Range” input box to reset the relative movement range of A3~A6 axes based on the current position (the default value of positive and negative direction range is automatically set according to the positive and negative soft limit range).
5. Click the “Trial Run” button to start the slow trial run trajectory to check whether the operating environment is safe.
6. Click on the “Start Recognition” button to start high-speed running to recognize the trajectory (the process belongs to the load recognition data collection process, the same trajectory repeated ten times).
7. After the identification is completed, click the “Save ” button to complete the load parameter configuration.

**Attention:**

Trajectory trial runs can only be performed in manual mode, while high-speed trajectory recognition can be performed in manual or automatic mode.

Before load recognition, the end of the robot must be hooked up to the appropriate load and the robot system must be in an idle state.

The collision detection function is automatically turned off during the high-speed run to recognize the track trajectory.

9.4 Collision detection function

Instructions Collision detection is a feature that enhances robot safety by quickly detecting and stopping robot motion when the robot is impacted by an external force. This function is automatically enabled from the moment the system is powered on, and is effective in manual, automatic, and external modes, and is a safeguard for maintaining safe robot operation.

9.4.1 Collision Detection Switch

Instructions Collision detection is automatically enabled from the moment the system is powered up after friction recognition, and can be briefly disabled by the program instruction COL_DETECT. The scope of this instruction is the same as that of the global variable.

Steps of operation

Program Example:

COL_DETECT = OFF ‘During program execution, collision detection function is turned off’



1. Select the upper line of the instruction line to be inserted.
2. Select Command → Dynamics Command.
- 3, slide the drop-down box options to find the “COL_DETECT ” option and click to select.
- 4, click the cursor position to the right input box.
- 5, click the “Other ” drop-down box options, find “OFF ” or “ON ” option and click.
- 6, click the “OK ” button to add the command is complete.



Attention:

After the function is briefly turned off by a program command, it will automatically switch to the on state when the program is in the non-running state such as uninstalling, reporting an error, or pausing, or when it finishes executing the program, or when it enters the online editing, etc.; and it will automatically switch to the off state when the program resumes running.

The collision detection system will be turned on automatically after the power is turned on for the model supported by the dynamics and after the robot has been recognized ontologically.

The current version of the collision detection response is deceleration and stop.

9.4.2 Collision Detection Sensitivity

Instructions

The larger the collision detection sensitivity value, the higher the detection sensitivity. The collision detection sensitivity can be modified in the “Collision Detection Configuration” interface of the Teachers, or through the COL_GUARD_ADJUST command in the program.

Steps of

Operation in Method One



1. Select Configuration → Dynamics Functions → Collision Detection Configuration in the menu bar.
2. Click the “Modify” button and enter a value in the “Collision Detection Sensitivity” input box, the input range is 1-100.

3. Click “OK” button to save the modified sensitivity value.

**Steps of
Operation in
Method 2**

Modified by the COL_GUARD_ADJUST command.

1. Select the upper line of the instruction line to be inserted.
2. Select Instruction → Dynamics Instruction.
3. slide the drop-down box to find the “COL_GUARD_ADJUST” option and click to select.
4. click the cursor position to the right input box, you can enter a range of 1-100 values; or click the “Register” drop-down box option, find the “R []” option and click, and in the R [] index box, enter the R register number.
5. click the “OK” button to add the instruction is complete.

Example 1:

```
COL_GUARD_ADJUST = 50      'Set the collision detection sensitivity to
50
```

Example 2:

```
COL_GUARD_ADJUST = R[1]    'Set the collision detection sensitivity value
to the value in the R [1] register
```



Attention:

- Only after the body friction identification can you enter the “Collision Detection Configuration” interface for configuration.
- The flag oCOL_STATE is added to the running configuration for IO configuration; oCOL_STATE outputs an ON signal when the collision detection is on, but no signal is output when it is off.
- The sensitivity value is also calculated based on the corresponding load number. Switching different loads should set the corresponding sensitivity value.
- The detection is based on the body friction identification and load parameters for collision detection, if the correct parameters are not filled in, the collision will be false.
- If the sensitivity of collision detection is set to 1, the system disables the function by default.

9.5 Drag-and-drop teaching function

Instructions	The operator directly drags the robot body for demonstration purposes, and adjusting the drag flexibility coefficient can improve the drag effect to some extent.

Steps of operation	1. Press the drag button on the end of the robot's six axes. 2. When the robot enable is turned on, the robot can move in any direction in which the hand applies an external force.



Attention:

- Drag is allowed only in manual mode.
- The robot must be recognized as a load, and the corresponding load number must be called up before using this function. Otherwise, the robot may fall off the arm if the recognition data is inaccurate, or other dangerous situations may occur.

9.5.1 Drag-and-drop instructional settings

Instructions	The Drag Teach setting supports Articulated Drag mode by default, and Cartesian Drag mode is only supported by some models (Cartesian Drag Configuration interface is displayed only for supported models). According to the choice of drag mode, set the corresponding drag flexibility. The larger the drag suppleness coefficient, the smoother the drag will be.

Steps of operation



1. Select → Configuration → Dynamics → Trailing Setting in the menu bar.
2. Click “Drag Function Switch” on the top to open the drag function.
3. Click “Modify” button to set the desired value in the corresponding axis dragging flexibility bar, the value range is 1-100.
4. Click “Confirm” to save the modified value of the drag flexibility coefficient, which will take effect immediately after the parameter is set.



Tip:

You can only enter the “Drag Show Setting” interface for configuration after you have performed the body friction identification.

9.5.2 Drag Track Recording

Instructions

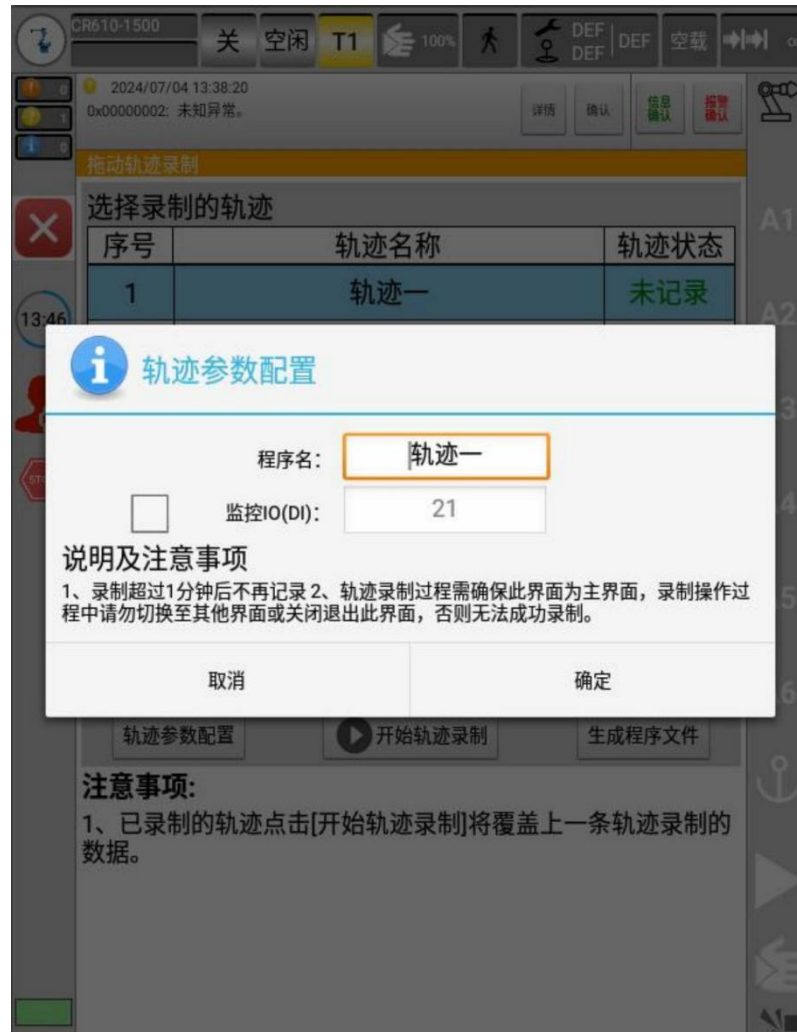
The user uses the collaborative robot drag function to drag the robot for trajectory teaching, the software will record the robot trajectory taught by the user, and then generate the corresponding robot program, which saves the points of the trajectory taught by the user, and the reproduction of the trajectory taught by the user can be accomplished by executing the program.

Steps of operation

1. Select the trajectory:



1. Select → Configuration → Dynamics → Drag Track Recording in the menu bar.
2. Select the trajectory to be recorded, the content box on the right will show the recording status of the current trajectory accordingly, and up to 8 trajectories can be recorded and saved at the same time. After selecting a track, click “Track Parameter Configuration” button and the following interface will pop up:

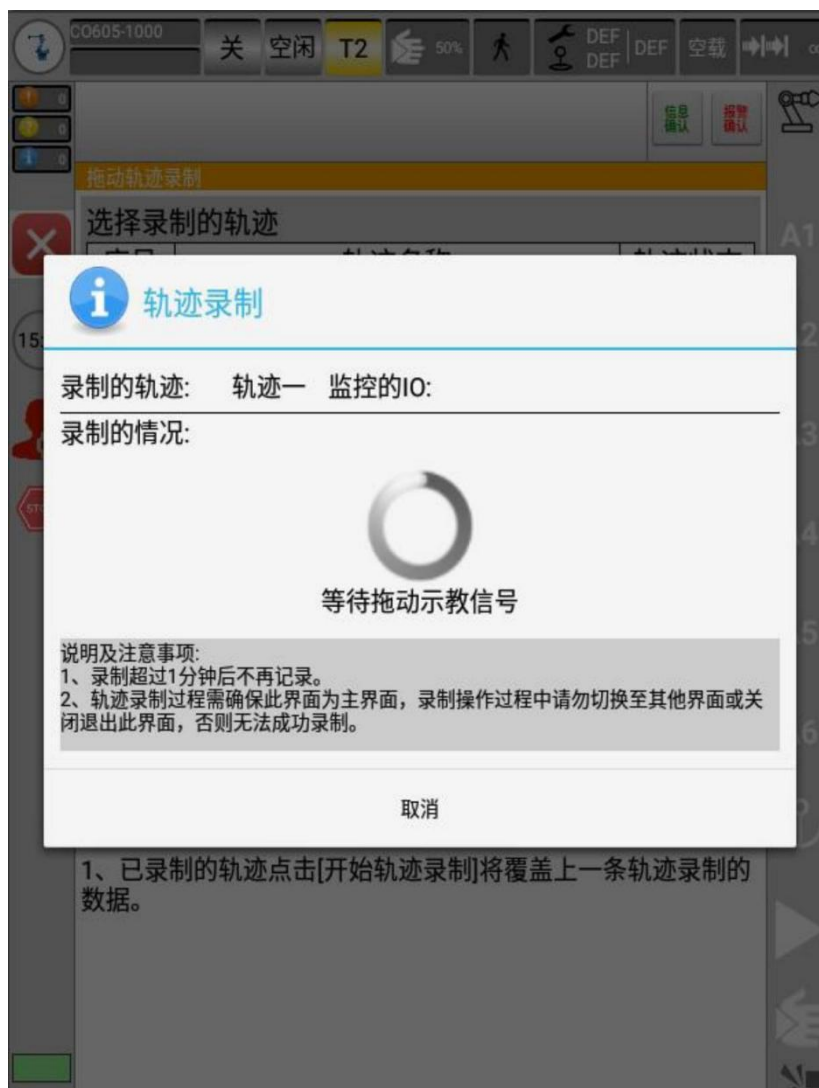


- Program name: the name of the recorded track, can be customized.
- Monitor IO(DI): the IO(input IO) recorded during the track recording process, i.e. record the change of this IO while dragging the recorded track, the current version only supports to record one IO. you can choose to record or not to record, checking the left box is to choose to record, unchecking is to choose not to record, the default is unchecked.

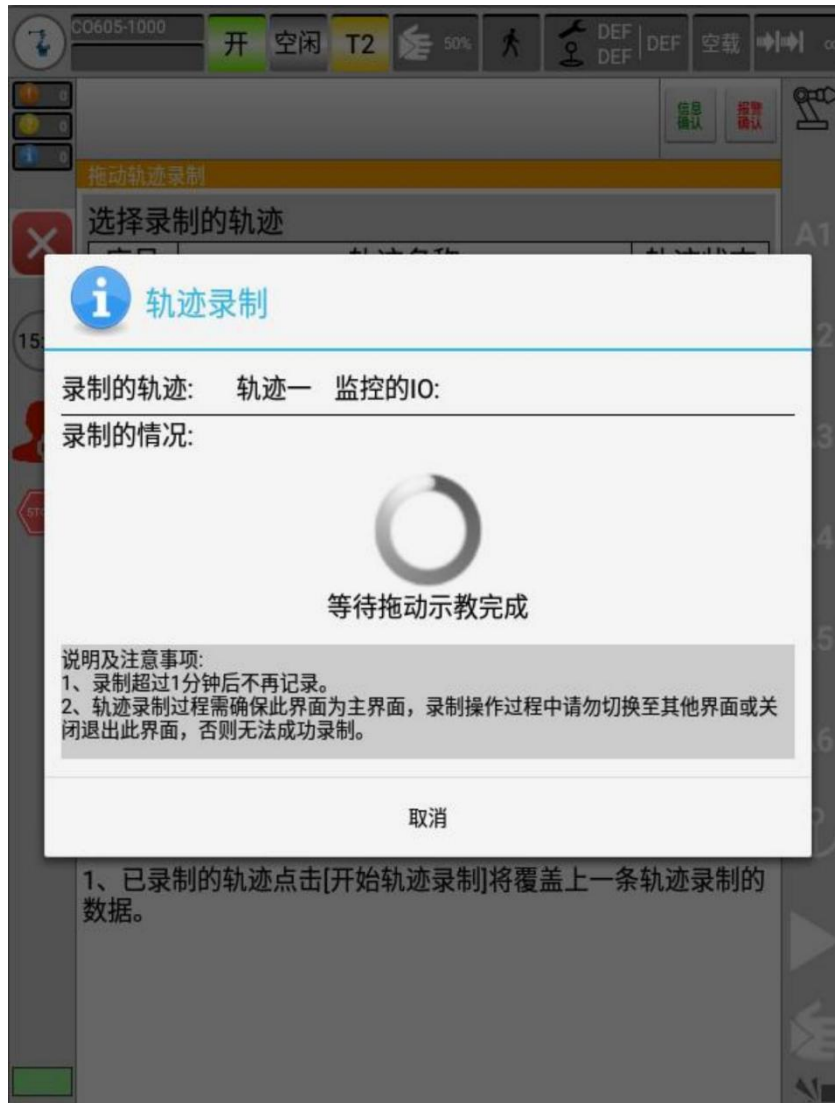
When the track status shows not recorded, Instructions has not recorded the track, so you can't click the "Generate Program File" button to generate the track, then you need to click the "Start Track Recording" button to start recording the track formally.

2. Drag the demonstration track:

Click the "Start Track Recording" button to enter the track recording interface.



The teaching status will be displayed in the Trace Teach interface. The above figure shows “Waiting for Drag Teach Signal” which means that the software is waiting for a trigger signal from the user. When the user presses the drag button on the end of the flange, the interface will change to “Waiting for drag teaching completion”, as shown below.

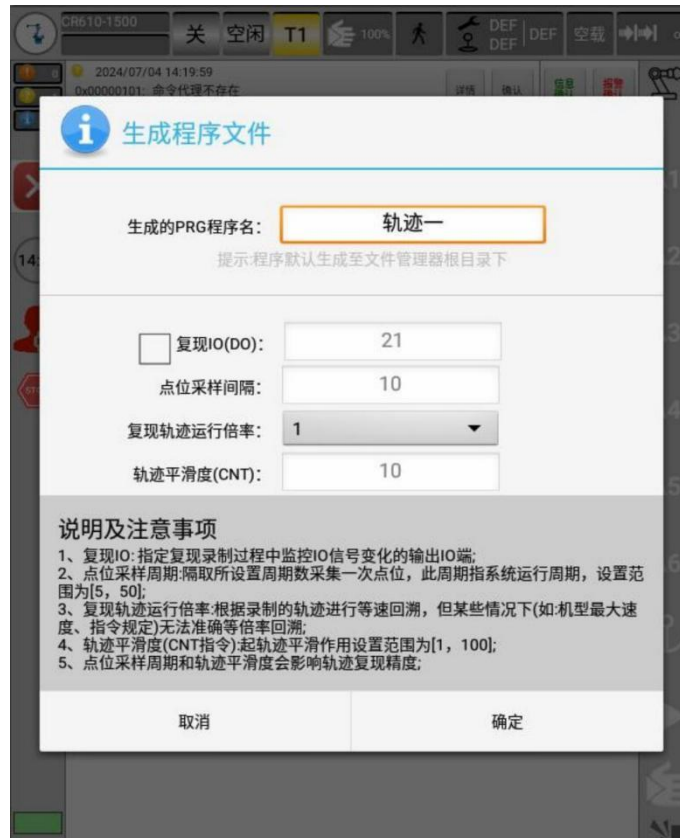


This page represents that the software has entered the trajectory teaching recording state, the software will record the robot trajectory taught by the user in real time. The maximum recording time is one minute, and the part of the trajectory that exceeds one minute will not be recorded. If the user wants to cancel the recording in the middle of the process, click the “Cancel ” button, the software will return to the previous interface, and the user can click the “Start Trajectory Recording ” button again to enter the trajectory recording interface. In the process of track recording, release the drag button to end the track recording automatically and return to the previous interface.

3. Generate trajectories:



After the track is successfully recorded, the track status will be changed to “Recorded”, at this time, you can click the “Generate Program File ”button for the next step of track program generation. After clicking it, the pop-up interface is as follows.



- Generated PRG program name: the final generated PRG program name, can be customized.
- Duplicate IO(DO): the IO(output IO) to be duplicated at the same time in the
- track duplication process, the current version only supports duplicating one IO, you can only check this option if you have selected the monitor IO in step 1 and recorded it successfully. You can choose to reproduce or not to reproduce, check the left box to choose to reproduce, uncheck the box to choose not to reproduce, the default is not to reproduce.
- Point Sampling Interval: take a point every few points when generating the
- trajectory, the default value is 10, that is, take a point every 10 points on the
- basis of the original trajectory to generate the trajectory, the larger the value is, the lower the accuracy of the trajectory recorded and reproduced, the setting
- range is [5, 50].
- Reproduction Estimated Running Multiplier: According to the recorded
- trajectory, the speed multiplier will be equalized, but in some cases, due to the condition limitations (such as the maximum speed of the model, the command regulations, etc.), it is impossible to accurately equalize the speed multiplier.
- Trajectory Smoothness: The setting range of this configuration item is 1-100, the larger the value, the higher the degree of trajectory smoothing (corresponding to the program CNT instruction).



Tip:

The drag signal for collaborative models is the drag button at the end of the flange.

When you repeat the recording of the same track, it will clear and overwrite the original data, and then a pop-up window will prompt “This track has been

recorded, re-recording will overwrite the original track data! ”. Please choose carefully.

Please be careful in choosing. After the track is recorded once, it can be

generated repeatedly, click “Generate Program File ” button to change the configuration items, then click “OK ” button again, and the system will

regenerate the track program according to the configured parameters.

9.6 Soft Float Function

Soft Float is a function that allows the robot arm to float along the direction of the external force when it is subjected to the external force according to the set float parameters. This function can effectively avoid the problem of material friction and material deflection caused by the inaccurate position of the end fixture to pick and place the material. Before using the soft float function, the arm must be properly calibrated for dynamics and friction parameters, and if a load is installed, the dynamics load parameters must be properly configured.

Currently the soft float can be adjusted by user parameters as well as algorithmic parameters. The user parameters include soft float type, soft float switch for each axis, flexibility, resilience, maximum speed, positive and negative limits, and other parameters. Algorithm parameters include gain, mass, stiffness, damping, minimum external force value, maximum external force value and other parameters.

In general, the adjustment of user parameters can meet the requirements of field process use. If the adjustment of the user parameters can not meet the requirements of the field process, you can contact our engineers to adjust the algorithm parameters.

9.6.1 Steps and Parameters Instructions

Steps of operation:

1. Upgrade the Soft Float Patch Pack.
2. Install the Soft Float Process Pack.
3. Configure the parameters in the Soft Float Process Pack screen.
4. Reference the Soft Float macro file in the PRG program using the “More” option.
5. Enter the appropriate Soft Float commands in the PRG program using manual commands.

Specific parameter configurations are set in the process package:

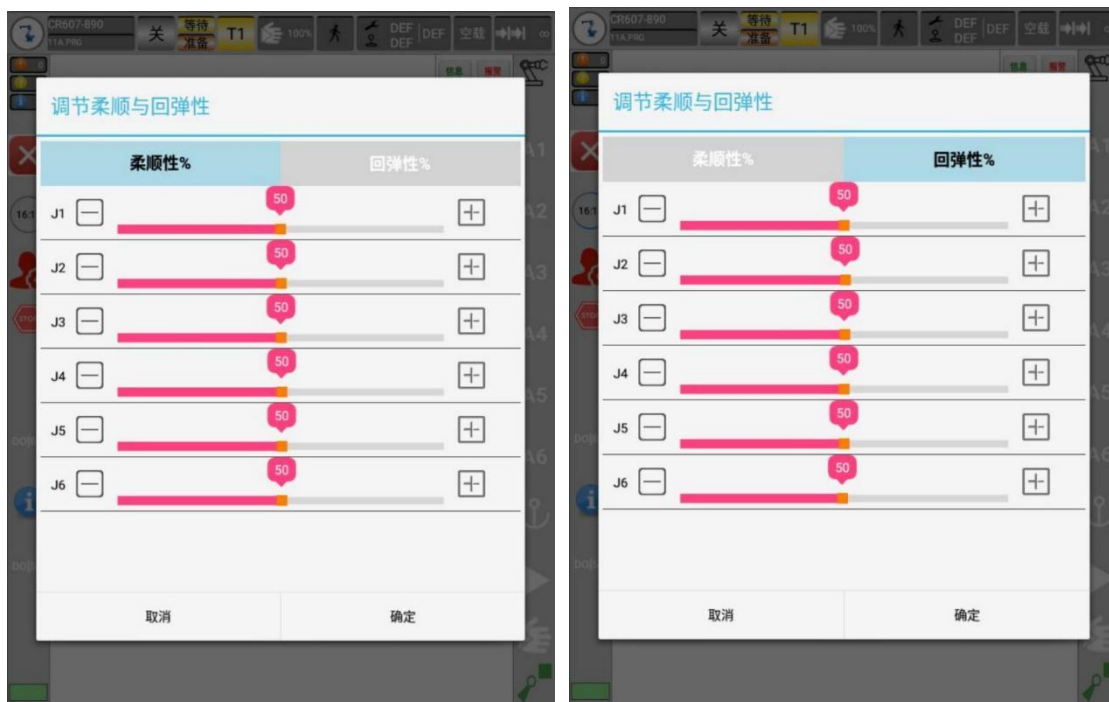


Figure 1-1 [Soft Float Process Package] Operation Interface

- (1) Soft Float No.: 0-8 groups of serial numbers; Soft Float provides a total of 8 groups of configuration items for the user to select the configuration. Users can make notes for each group of configuration items. (The system supports the soft-float function of the factory models have been configured by default, the number 0 and number 1 configuration items for user reference configuration)
- (2) Soft float number Instructions: you can enter the number of the note information
- (3) Float type selection: joint or space coordinate system can be selected, the coordinate system below the joint coordinate system displays the joint coordinate system, and the number of axes is displayed according to the model, and XYZABC is displayed when the space coordinate system is selected. The float type supports joint coordinate float (currently only supports six-axis joint float) and Cartesian space coordinate float (currently only supports XYZ float). When joint coordinates are selected, the

arm will float according to the external force on each axis; when spatial coordinates are selected, the arm will float according to the direction of the axes corresponding to the coordinate system it is using when the soft float function is turned on.

- (4) Parameter list: list. In order to display the current parameter list, in addition to each joint float switch, if you need to modify the list data, the user clicks on the [Modify] button, the current parameter list is entered into the modified state interface.
- (5) Float Switch: Enables or disables the soft float function for each axis/Cartesian space.
- (6) Modify: button. Click the Modify button to modify the information of the current list
- (7) Flexibility (%): button. When the parameter list interface is modified, the button is not clickable, on the contrary, it can be clicked, and the softness parameter adjustment window pops up by clicking on it, the softness is to adjust the soft float softness, which can be adjusted from 0 to 100%, and when it is adjusted to 0, the soft float function will be invalidated.
- (8) Resilience (%): button. Parameter list interface for the modified state of the button for the non-clickable state, and vice versa can be clicked, click to pop up the rebound parameters adjustment window, rebound that is, to adjust the stiffness of the soft float, can be adjusted from 0 to 100%, when adjusted to 0, the soft float to lose the spring properties.
- (9) Minimum external force value (N/Nm): the minimum external force value for triggering floating. Different models in different positions to open the soft float, the corresponding minimum external force value will be different, when the minimum external force value is set incorrectly, there may be a mechanical arm fluttering and so on, at this time, the user can increase the minimum external force value appropriately, after adjusting the parameter, the force that triggers the soft float of the mechanical arm also needs to be greater.
- (10) Maximum speed ($^{\circ}/s$): the maximum speed when floating. The maximum speed is the maximum speed that can be reached by the limitation of the arm when floating softly, in the process of floating softly, the maximum speed that can be reached by each joint is not large, and it is in the state of low-speed operation.
- (11) Negative Limit ($^{\circ}$): The maximum displacement of each axis to float in the negative direction, based on the position at which soft float is started.
- (12) Positive limit ($^{\circ}$): Maximum displacement of each axis to float in the positive direction based on the position at which soft float is started.



- Flexibility scroll bar: the data on the scroll bar is the value of the flexibility of the current axis, the 【-】 and 【+】 buttons on both sides of the scroll bar are for adjusting the precision, click to adjust 1%, the flexibility value can be set from 1 to 100%, 1% for the hardest, 100% for the most flexible.
- Resilience scroll bar: that is, the degree of rebound after the withdrawal of external forces, the data on the scroll bar for the current axis of the degree of rebound value, scroll bar on both sides of the 【-】 and 【+】 buttons for the adjustment of precision, click to adjust 1%, the degree of rebound after the withdrawal of external forces: 0 ~ 100%, 0 means no rebound, 100% that is, back to the initial position.

9.6.2 Soft Float Instructions

The soft float function is a macro instruction, `SOFTFLOAT<n>` specifies which group of float parameters to use, the value of `n` ranges from 0 to 7, and the grouping expression specifies the judgment condition to stop the soft float. the `TIMES` attribute specifies the time to float, and stops the soft float after the time has been reached; the `DI_NUM` and `DI_VALUE` specifies that the soft float stops after the state of this input IO is equal to `DI_VALUE`; the `R_NUM` and `R_VALUE` specifies that the soft float stops after the value of this R-register is equal to `R_VALUE`. `DI_NUM` and `DI_VALUE` specify that soft floating stops when the state of this input IO is equal to `DI_VALUE`; `R_NUM` and `R_VALUE` specify that soft floating stops when the value of this R register is equal to `R_VALUE`.

The stopping condition of soft float can be specified at the same time, and soft float is turned off as soon as one condition is satisfied.

The syntax is formatted as follows:

```
MAC SOFTFLOAT <n> [(TIMES=numerical value) | (DI-NUM=numerical value DIVNet=ON | OFF)
| (R_NUM=numerical value R_VALUE=numerical value)], OUT
```

Example of use Instructions:

Example 1:

MAC SOFTFLOAT 0 ,OUT ‘=Without specifying the stopping condition of Soft Float, Soft Float will be kept floating after it is turned on, and Soft Float will not be stopped until the program is uninstalled.

Example 2:

MAC SOFTFLOAT 0 TIMES=2000, OUT ‘Turn on soft float. Turn off soft float after 2S of float.

Example 3:

MAC SOFTFLOAT 0 DI_NUM=1 DI_VALUE=ON, OUT ‘Turn on soft float and block, wait for DO[1]=ON to turn off soft float

Example 4:

MAC SOFTFLOAT0 R_NUM=9 R_VALUE= 10.0, OUT ‘Turn on soft float and block, wait for R[9] = 10.0 to turn off soft float

Example 5:

MAC SOFTFLOAT 0 TIMES=2000 DI_NUM= 1 DI_VALUE=ON R_NUM= 1 R_VALUE= 10, OUT ‘ Enable soft float, specify multiple stop conditions at the same time, and exit soft float if only one condition is met.

The program example is as follows:

```
<PROGRAM>
  J P0 ‘The soft float can only be turned on after the robot arm moves to the point and stops.
  MAC SOFTFLOAT 0 TIMES=2000, OUT ‘Starts soft float with group 0 parameters and closes after
2 seconds of float time
  J P1 ‘Movement to point P1
<END>
```

Precautions

(1) The softfloat feature is only supported on models that have been adapted to the dynamics model. If

the model does not support the softfloat function, a warning will be thrown when executing the MAC SOFTFLOAT command, and there will be no floating effect.

(2) only correctly adapted to the soft float parameters (user parameters and algorithm parameters) to ensure the normal use of soft float, if you want to use this function but the model shows that the alarm is not adapted to the soft float parameters, please contact our engineers to assist in solving the problem. (There are only a few models that support soft float function by default)

(3) When Soft Float is on, the current tool, workpiece, and load level parameters are used.

(4) The SOFTFLOAT instruction blocks the motion smoothing of the previous motion instruction; the previous motion instruction will also block at the SOFTFLOAT instruction line when the PASS attribute is used, and the program Instructions are as follows

<PROGRAM>

J P1 CNT= 100,PASS ‘No smoothing effect, accurate movement to point P1.

MAC SOFTFLOAT 0 TIMES=2000, OUT ‘Starts soft float with group 0 parameters and closes after 2 seconds of float time

<END>

(5) The soft-float function is automatically turned off in the following cases: when the program is paused, stopped, or unloaded; when single-step debugging, arbitrary line debugging, or backing up; and when the system alarms. The system must throw a prompt after closing the soft-float function.

(6) The system internally limits the soft float to be the maximum value, and when the maximum speed value set by the user exceeds the maximum value set internally by the system, it moves at the maximum value set internally by the system.

(7) When using the `softfloat` command to specify which group of parameters to use, make sure that the `softfloat` process package is properly loaded with the corresponding setup parameters for the same group.

10 Communications

10.1 MODBUS communication

Modbus is a general-purpose serial communication protocol that works in a master-slave mode. There is only one master device in the whole network, and the other devices are slaves. The master is responsible for initializing the system communication settings and sending information to the slaves, who receive the information and respond to the master's query or make response actions according to the master's message. When the master does not send a request, the slave will not send out data by itself, and there is no direct communication between the slave and the slave.

Modbus supports two kinds of data transmission rules for serial port (RS485) and Ethernet port, i.e. Modbus RTU and Modbus TCP/IP.

In Modbus TCP mode, the master, also known as Client, takes the initiative to connect to the slave; the slave, also known as Server, usually opens a port and waits for the client (master) to connect. Standard Modbus TCP uses port 502 by default.

When using Modbus RTU, you have to set the serial communication parameters (serial port number, baud rate, parity, etc.), and all devices on a Modbus network must select the same transmission mode and serial port parameters.

In versions after V1.6.11. For the Modbus communication function, the robot arm can be both master and slave at the same time, and they do not affect each other. This is because the memory opened by the master and slave functions are independent of each other and can run simultaneously.

When the robot acts as a master, it uses the protocol-less generic Modbus function. The user calls the relevant Modbus function commands inside the program to perform communication and read/write operations on the slave data (sensors).

When the robot arm acts as a slave (server), the system provides Modbus slave control protocol for communication. Modbus slave control protocol is a kind of pre-defined control protocol within the robotic arm controller system, and each register has a clear meaning. The user writes the relevant values into the registers to control the robot arm to perform the corresponding operations. To use the Modbus slave control protocol, you need to enable the Modbus server function switch on the demonstrator/workstation software interface, and then wait for the host computer to connect to establish communication.

ModbusCmd slave control protocol function only supports ModbusTCP. If the end-user's device is a serial port, it must use the gateway to transfer.

The configuration operation and register data protocol of ModbusCmd Slave Control Protocol

function are described in the following sections.

10.2 MODBUS Master

When the robotic arm is a modbus master, the communication can be realized in the background program. The program example is as follows

```
<program>
LBL[0]
R[0]= OPEN_MODBUS_TCP(1," 10.10.56.112",502,0)      'establish
IF R[0]=-1 THEN
    THROW "MODBUSClient 1 was successfully created, but the connection to slave 1 failed! Reconnect
slave 1 when reading or writing registers "
    LEVEL=WARNING
END IF

LBL[1]
'*****Programming operation that writes the parameters R,DO,DI of the robot arm to the slave
station*****
'Write the DO value of the robot arm to the coil of slave 1.
R[0] = WRITE_MDB_COILSTATUS(1,0,3,DO[0])      'Robotic arm DO[0~2] --> Coil COIL[0~2] of the
slave station

'Write the R value of the robot arm to the coil of slave 1. Write the R register value to the Boolean coil and
the result is " 1 if not 0 " .
R[1]=3.2
R[2]=0
R[3]=-1
R[0]= WRITE_MDB_COILSTATUS(1,0,3,R[1]) 'Robotic arm R[1~3]--> Coil COIL[0~2] of slave,
COIL[0]=1,COIL[1]=0,COIL[2]=1 of slave

'Write the DI value of the robot arm to the coil of slave 1.
'Because in general there is no such need scenario. If you really need it, you can assign DI to R hosting first,
and then write it.
R[1]=DI[0]
R[2]=DI[1]
R[3]=DI[2]
R[0]= WRITE_MDB_COILSTATUS(1,0,3,R[1])      'Robotic arm DI[0~2] --> Robotic arm R[1~3]--> Coil
COIL[0~2] of slave station

'Write the value of the R register of the robot arm to the save register of slave 1,DATATYPE=0 means that
the input register of the slave is 16-bit INT type.
R[5]=5
```

```

R[6]=6
R[7]=7
R[0]= WRITE_MDB_HOLDREG(1,0,3,R[5],0)      'Robotic arm R[5~7]--> Coil HOLD_REG[0~2] of
slave station

*****Operation of reading register data from a slave station into the robotic arm system*****
'The value of 10 coils starting from 0 will be read from slave 1 and saved in the R,DO,VDI of the robot arm;
the user can choose any one of the saving methods
R[0]= READ_MDB_COILSTATUS(1,0,10,R[10])    'Coil COIL[0~9] of slave --> R[10~19] of robot arm
R[0]= READ_MDB_COILSTATUS(1,0,10,DO[10])    'Coil COIL[0~9] of slave --> DO[10~19] of robot
arm
R[0]= READ_MDB_COILSTATUS(1,0,10,VDI[33])    'Read PLC's COIL[0~9]--> Save in VDI[33~42]

'Read the value of slave 1, 10 discrete input coils from 0, and save it in the R,DO,VDI of the robot arm.
R[0]= READ_MDB_INPUTSTATUS(1,0,10,R[20])    'Discrete input InputStaus[0~9] from slave 1 -->
R[20~19] from robotic arm
R[0]= READ_MDB_INPUTSTATUS(1,0,10,DO[20])    'Discrete input InputStaus[0~9] from Slave 1 -->
DO[20~29] of robotic arm

'Reads the values of 3 input registers from slave 1, starting from 10, and saves them in the robot arm's R.
DATATYPE=0 means that the slave's input registers are 16-bit INT types.
R[0]= READ_MDB_INPUTREG(1,10,3,R[30],0)      'Slave 1 input register INPUT_REG[10~12] -->
R[30~32] of robot arm
'Read the value of slave 1, starting from 10, 5 save registers, saved in R of the robot arm.
R[0]= READ_MDB_HOLDREG(1,10,5,R[40],0)      'Save register of slave 1 HOLD_REG[10~14] -->
R[40~44] of robotic arm
WAIT TIME= 10      'Delay 10ms
GOTO LBL[1]
R[0]= CLOSE_MODBUS_TCP(1)      'Close the connection between the robot arm and slave 1
<end>

```

10.3 MODBUS slave

10.3.1 Configuration Interface

When the robot acts as a slave (server) and needs to communicate using the Modbus slave control protocol, you need to enable this function on the demonstrator/workstation software interface first.

Path of Oscillator Software Interface: Main Menu - [Communication] - [MODBUS Configuration].

MODBUS配置

MODBUS服务端功能开关: 开启

端口设置:

从站ID:

高低位: CDAB ▼

从站更新周期: **ms**

连接状态: 未连接

保存

注意:

- 1、保存成功需要重启系统才能生效!
- 2、只有MODBUS功能开关才能保存成功!

There are five configuration items on the configuration screen:

- (1) Modbus server function switch. Used to turn on and off the Modbus communication function.
- (2) Port Settings. Used to configure the communication port. The default value is 507. Setting the port does not allow conflicts with other ports that use the TCP/IP function.
- (3) Slave ID number. Used to configure the slave ID (SlaveID) of the robot arm. The default value is 1.
 1. Take communication with Siemens PLC as an example, if the PLC writer does not set the slave address of the function block, Siemens PLC may use the broadcast form, so the value of SlaveID must be set to 255. (If communication fails and the alarm slave address is incorrect, you can use the debugging assistant of your computer to connect with the PLC, check the actual message sent by the PLC to determine the actual slave id number, and then fill in the place.)
- (4) High-Low Bit. Used to configure the high and low bit order of a 32-bit single precision floating point data consisting of two 16-bit Modbus registers.
- (5) Data update cycle. Used to regulate how fast or slow the system data is synchronized. The default value is 50 in ms. the smaller the update period value, the faster the data update, but the greater the CPU performance burden on the controller.

10.3.2 Protocol Instructions

10.3.2.1 Coil Status Register CoilStatus

The CoilStatus register is readable and writable. Function codes: 0x01, 0x05, 0x0f.

Its data protocol is shown in Table 3-1.

Table 2-1 Coil status register data protocol

CoilStatus	Data	element	note

Address (10 scale)	Type		
0	Bit	DO[0]	Output IO
1	Bit	DO[1]	
2	Bit	DO[2]	
3	Bit	DO[3]	
...	Bit	...	
63	Bit	DO[63]	
64	Bit	VDI[0]	The value of the VDI variable, i.e. the value of the virtual input IO. If users need to load or run the program in external mode, they must first configure the IOs in the Run Configuration interface on the software interface of the Teachers/Workstations, and bind the IOs to the externally-controlled signals (here, it should be noted that the bound signals need to be virtual signals, i.e., the bound signals are those that are displayed as VIRTUAL on the Digital Input/Output interface). After the signal binding is completed, the state of the program can be controlled by writing the value of VDI[x] through ModbusCmd.
65	Bit	VDI[2]	
...	Bit	...	
96	Bit	VDI[32]	
...	Bit	...	

127	Bit	VDI[63]	

10.3.2.2 Discrete Input InputStatus

Discrete Input (InputStatus) read-only. Function code: 0x02.

Its data protocol is shown in Table 3-2.

Table 2-2 Discrete Input Data Protocol

Input Status Address (decimal system)	data type	content	notes
0	Bit	DI[0]	Real input IO status
1	Bit	DI[1]	
2	Bit	DI[2]	
...	Bit	...	
63	Bit	DI[63]	

10.3.2.3 Input Register InputReg

The InputReg is read-only. Function code: 0x04.

Its data protocol is shown in Table 3-3.

Table 2-3 Input Register Data Protocols

Input Status Address (decimal system)	data type	content	notes
1	byte	Alarm status	0: No alarm 1: Emergency stop alarm 2: Servo alarm 3: Forward over acceleration 4: Axis reaches limit 5: Forward looking over limit 6: Position unreachable 7: Other alarms

2	byte	Enable state	0: Not enabled 1: Enabled
3	byte	The current operating mode of the robot	0: None 1: Manual T1 mode 2: Manual T2 mode 3: Automatic mode 4: External mode
4	byte	Tool number currently called	Default: -1 to 15
5	byte	Tool number currently called	Default: -1 to 15
6	byte	The coordinate system of the current robot's point motion	0: None 1: Axis coordinate system 2: World coordinate system 3: Base coordinate system 4: Workpiece coordinate system (user coordinate system) 5: Tool coordinate system
7	(Double Word)	J1 Low position coordinates	Current joint position value
8	Floating Point Data	J1 Coordinate high position	
9	(Double Word)	J2 Low position coordinates	
10	Floating Point Data	J2 Coordinate high position	
11	(Double Word)	J3 Low position coordinates	
12	Floating Point Data	J3 Coordinate high position	
13	(Double Word)	J4 Low position coordinates	

14	Floating Point Data	J4 Coordinate high position	
15	(Double Word)	J5 Low position coordinates	
16	Floating Point Data	J5 Coordinate high position	
17	(Double Word)	J6 Low position coordinates	
18	Floating Point Data	J6 Coordinate high position	
19	(Double Word)	E1 Low position coordinates	
20	Floating Point Data	E1 Coordinate high position	
21	(Double Word)	E2 Low position coordinates	
22	Floating Point Data	E2 Coordinate high position	
23	(Double Word)	E3 Low position coordinates	
24	Floating Point Data	E3 Coordinate high position	
25	(Double Word)	X Low position coordinates	The Cartesian coordinate position value of the robot in the current tool/workpiece coordinate system
26	Floating Point Data	X Coordinate high position	
27	(Double Word)	Y Low position coordinates	
28	Floating Point Data	Y Coordinate high position	
29	(Double Word) Floating Point Data	Z Low position coordinates	
30		Z Coordinate high position	

31	(Double Word) Floating Point Data	A Low position coordinates	
32		A Coordinate high position	
33	(Double Word) Floating Point Data	B Low position coordinates	
34		B Coordinate high position	
35	(Double Word) Floating Point Data	C Low position coordinates	
36		C Coordinate high position	
37	(Double Word) Floating Point Data	E1 Low position coordinates	
38		E1 Coordinate high position	
39	(Double Word) Floating Point Data	E2 Low position coordinates	
40		E2 Coordinate high position	
41	(Double Word) Floating Point Data	E3 Low position coordinates	
42		E3 Coordinate high position	
43	(Double Word) Floating Point Data	J1 Low joint velocity	Current actual joint velocity value
44		J1 High joint velocity	
45	(Double Word) Floating Point Data	J2 Low joint velocity	
46		J2 High joint velocity	

47	(Double Word) Floating Point Data	J3 Low joint velocity	
48		J3 High joint velocity	
49	(Double Word) Floating Point Data	J4 Low joint velocity	
50		J4 High joint velocity	
51	(Double Word) Floating Point Data	J5 Low joint velocity	
52		J5 High joint velocity	
53	(Double Word) Floating Point Data	J6 Low joint velocity	
54		J6 High joint velocity	
55	(Double Word) Floating Point Data	E 1 Low joint velocity	
56		E 1 High joint velocity	
57	(Double Word) Floating Point Data	E2 Low joint velocity	
58		E2 High joint velocity	
59	(Double Word) Floating Point Data	E3 Low joint velocity	
60		E3 High joint velocity	
61	(Double Word)	J1 The actual current value of the joint is low	Actual value of current joint current
62	Floating Point Data	J1High actual joint current value	

63	(Double Word)	J2 The actual current value of the joint is low	
64	Floating Point Data	J2 High actual joint current value	
65	(Double Word)	J3 The actual current value of the joint is low	
66	Floating Point Data	J3 High actual joint current value	
67	(Double Word)	J4The actual current value of the joint is low	
68	Floating Point Data	J4 High actual joint current value	
69	(Double Word)	J5 The actual current value of the joint is low	
70	Floating Point Data	J5 High actual joint current value	
71	(Double Word)	J6 The actual current value of the joint is low	
72	Floating Point Data	J6 High actual joint current value	
73	(Double Word)	E1 The actual current value of the joint is low	
74	Floating Point Data	E1 High actual joint current value	
75	(Double Word)	E2 The actual current value of the joint is low	
76	Floating Point Data	E2 High actual joint current value	
77	(Double Word)	E3 The actual current value	

78	Floating Point Data	E1 The actual current value of the joint is low	
79	(Double Word)	Current end TCP movement speed	The synthesis speed ofXYZ
80	Floating Point Data		
81	(Double Word)	Current end TCP movement speed	The synthesis speed ofABC
82	Floating Point Data		
83	byte	Motion status of robot axis group	0: Initial value 1: Not enabled 2: Stop in progress 3: During exercise 4: Alarm
84	byte	Robot motion program status	0: None 1: Not loaded 2: Already loaded, in preparation 3: Running 4: Pause in progress 5: Error occurred

10.3.2.4 Hold Register

Hold registers are readable and writable. Function codes: 0x03, 0x06, 0x10.

The data protocol is shown in Table 3-4

Table 2-4 Maintain Register Data Protocol

HoldReg Address (10 scale)	data type	content	notes
0	Bit	Clear alarm	Rising edge trigger

1	Bit	Enable switch	0: Disconnect Enable 1: Up enable
2	Bit	Operation mode setting	0: None 1: Manual T1 mode 2: Manual T2 mode 3: Automatic mode 4: External mode
3	Bit	Robot manual operation speed	1 to 100 (unit:%)
4	Bit	Robot automatic running speed	1 to 100 (unit:%)
5	Bit	Point motion coordinate system setting	0: None 1: Axis coordinate system 2: World coordinate system 3: Base coordinate system 4: Workpiece coordinate system (user coordinate system) 5: Tool coordinate system
6	Bit	Perform jog based on the currently set coordinate system	0: Stop jogging 1: J1 (X) axis jog forward 2: J1 (X) axis jog in reverse direction 3: J2 (Y) axis jog forward 4: J2 (Y) axis jog reverse 5: J3 (Z) axis point motion positive direction 6: J3 (Z) axis point motion reverse 7: J4 (A) axis jog forward 8: J4 (A) axis jog in reverse direction 9: J5 (B) axis jog forward 10: J5 (B) axis jog in reverse direction

			11: J6 (C) axis jog forward 12: J6 (C) axis jog reverse 13: E1 external axis jog forward 14: E1 External axis jog reverse 15: E2 external axis jog forward 16: E2 External Axis Point Movement Reverse 17: E3 external axis jog forward 18: E3 External Axis Point Movement Reverse
7		Tool coordinate system number setting	0: No operation 1: Select tool coordinate -1 (default) 2: Select tool coordinate 0 3: Select tool coordinate 1 ... 17: Select tool coordinates 15
8		Setting of workpiece coordinate system number	0: No operation 1: Select Workpiece Coordinate-1 (default) 2: Select workpiece coordinate 0 3: Select workpiece coordinate 1 ... 17: Select workpiece coordinates 15
9	Bit	Retrieve the current joint position and save it to the specified JR [x] register	0: No operation 1: Retrieve joint coordinate values and store them in JR [0] 2: Retrieve joint coordinate values and store them in JR [1] ... 1000: Obtain joint coordinate values and store them in JR [999]

10	Bit	Retrieve the current Cartesian position and save it to the specified LR [x] register	0: No operation 1: Retrieve Cartesian coordinate values and store them in LR [0] 2: Retrieve Cartesian coordinate values and store them in LR [1] ... 1000: Obtain Cartesian coordinate values and store them in LR [999]
11	Bit	Joint motion to point, joint motion to designated JR [x] register	0: No operation 1: Joint motion to point JR [0] 2: Joint motion to point JR [1] ... 1000: Joint motion to point JR [999]
12	Bit	Straight line motion to point, straight line motion to designated LR [x] register	0: No operation 1: Linear motion to point LR [0] 2: Linear motion to point LR [1] ... 1000: Linear motion to point LR [999]
13	Bit	The index of the R register used by the program configuration module in the running configuration function	The index number of the program mapping register. For example, if it is R [10], fill in 10 here
14	Bit	The value of the R register used by the program configuration module in the running configuration function	The value of the program mapping register. For example, if R [10] is configured, when 1 needs to be written into R [10], fill in 1 here
15	Bit	reserve	
16	Bit	reserve	
17	Bit	reserve	

18	Bit	reserve	
19	Bit	reserve	
20	Bit	reserve	
21	byte	(Double Word) Floating Point Data	R[0] data-low
22	byte		R[0] data-high
23	byte	(Double Word) Floating Point Data	R[1] data-low
24	byte		R[1] data-high
...	byte	(Double Word) Floating Point Data	R[...] data-low
...	byte		R[...] data-high
59	byte	(Double Word) Floating Point Data	R[19] data-low
60	byte		R[19] data-high
61	byte	16 bit unsigned shaping	R[20](Only supports read and write within the range of 0-65535)
62	byte	16 bit unsigned shaping	R[21](Only supports read and write within the range of 0-65535)
63	byte	16 bit unsigned shaping	R[22](Only supports read and write within the range of 0-65535)
...	byte	16 bit unsigned shaping	R[...] (Only supports read and write within the range of 0-65535)
120	byte	16 bit unsigned shaping	R[79](Only supports read and write within the range of 0-65535)


Tip:

The robot arm can be master and slave at the same time.

CompatibilityInstructions: Since V1.6.11 version, we recommend users to use the way of programming instruction, modbus communication process package is supported in low version.

10.4 SOCKET communication

Socket communication can be implemented in the background program. The robotic arm can be used as a client or a server, and when used as a server it can be realized as a pair of multiple clients.

10.4.1 SOCKET single client

When the robotic arm is a socket client, the communication can be implemented in the background program. The program example is as follows:

```
<program
> R[0]=-1
R[0]=OPENSOCKET OPTIONS=0 HANDLE= 1    ' Create a socket with a client handle of 1.
IF R[0]<>0 THEN
THROW "Failed to create client  !" LEVEL=WARNING
GOTO LBL[100]      'The communication operation is terminated and program execution is completed.
END IF

LBL[2]
R[0] = CONNECT(1," 10.10.56.112",30000)    'Connecting to Visual, IP: 10.10.56.112 ,port number is
30000. do not set the timeout time
IF R[0]<>0 THEN    ' The connection was unsuccessful, probably because the visualization host server is
not yet open
WAIT TIME=3000    ' Delay 3s, then reconnect
GOTO LBL[2]      'Reconnecting the visualization host computer
END IF
SOCKETMONITOR(1,2) 'Setting up to throw only warnings after disconnection, PRG does not stop

R[1]=ISHANDLEOPEN(1)
WHILE R[1]=1
    R[2]= BRECBUFEMPTY(1)    ' Determine if there is data in the receive buffer
    IF R[2]=1 THEN          ' The RECEIVE function is called only when there is data in the
cache.
        RECEIVE(1,SR[0])    ' Receive the data sent by the vision and save it in SR[0].
    END IF
    IF SR[0]<>"" THEN        ' Visual send T1, 1.123,2.0,3.0, string; cut to SR[1]=T1 SR[2]=1.123
SR[3]=2.0 SR[4]=3.0
        R[0] = FINDSTR(SR[0],",")    ' SR[0]=T1, 1.123,2.0,3.0 R[0]=3
        SR[1]= SUBSTR(SR[0],0,R[0]-1)    ' Intercept T1 on the left, SR[1] = T1
```

```

        SR[0]= SUBSTR(SR[0],R[0]+1)          ' Intercept the right part of the string, after the
interception SR[0]=1.123,2.0,3.0
        R[0] = FINDSTR(SR[0],",")          ' SR[0]=1.123,2.0,3.0
        SR[2]= SUBSTR(SR[0],0,R[0]-1)      ' Intercept 1.123 on the left, SR[2] = 1.123
        SR[0]= SUBSTR(SR[0],R[0]+1)        ' Intercept the right part of the string, after the interception
SR[0]=2.0,3.0

        R[0] = FINDSTR(SR[0],",")          ' SR[0]=2.0,3.0
        SR[3]= SUBSTR(SR[0],0,R[0]-1)      ' Intercept left 1.123 SR[3] = 2.0
        SR[0]= SUBSTR(SR[0],R[0]+1)        ' Intercept the right part of the string, SR[0] = 3.0 after
interception
        SR[4]=SR[0]
        SR[0]=" "
        SEND(1,"OK")
        'convert to digital
' SR[4]=3.0 'empty
        'Give the visual uplink OK
        R[20]=TOVAL(SR[2])
        R[21]=TOVAL(SR[3])
        R[22]=TOVAL(SR[4])

    END IF
END WHILE
THROW "Server shutdown! Reconnect" LEVEL=WARNING
GOTO LBL[2]

LBL[100]
CLOSESOCKET HANDLE= 1          'Close Sockets
<end>

```

10.4.2 SOCKET Server

When the robotic arm is used as a Socket server, the communication can be implemented in the background program. The program example is as follows:

```

<program>
R[0]=OPENSOCKET OPTIONS=0 HANDLE= 1      'Create creates a server and listens for client
connections with handle 1.
IF R[0]<>0 THEN          ' If the creation of the server is unsuccessful, an alarm is thrown, and then the
program ends and the program changes to the READY state.
    THROW "Failed to create server!" LEVEL=ERROR
    GOTO LBL[100]        ' Ends the communication operation.
END IF
R[1]=-1                  ' R[1] is used to save the handle for communication with the PLC. The
unconnected state returned when the ISHANDLEOPEN() function is executed for the first

```

```
IF R[10]=0 THEN      ' . The PLC is not connected and listens for connections, if the PLC is already
connected, there is no need to listen in the call to the ACCEPT function.
```

```
    'Listen to the connection, if the PLC connects to the robotic arm, save the fd of communication
with the PLC in R[1], and return after 5S of listening.
```

```
R[0]=ACCEPT(1,"10.10.56.112",30000,R[1],5000)
```

```
END IF
```

```
R[10]=ISHANDLEOPEN(R[1])
```

```
IF R[10]=1 THEN      ' R[10]=1 indicates a successful connection to the PLC.
```

```
    R[11]= BRECBUFEMPTY(R[1])      ' Determine if there is data in the receive buffer
```

```
    IF R[11]=1 THEN      ' The RECEIVE function is called only when there is data in
the cache.
```

```
        RECEIVE(R[1],SR[0])      ' Receive data sent by the vision.
```

```
    END IF
```

```
    IF SR[0]<>"" THEN      ' Visual send T1, 1.123,2.0,3.0 string; cut to SR[1]=T1 SR[2]=1.123
SR[3]=2.0 SR[4]=3.0
```

```
        R[0] = FINDSTR(SR[0],",")      ' R[0]=3    SR[0]=T1, 1.123,2.0,3.0 R[0]=3
```

```
        SR[1]= SUBSTR(SR[0],0,R[0]-1)      ' Intercept T1 on the left, SR[1] = T1
```

```
        SR[0]= SUBSTR(SR[0],R[0]+1)      ' Intercept the right part of the string, after the
interception SR[0]=1.123,2.0,3.0
```

```
        R[0] = FINDSTR(SR[0],",")      ' SR[0]=1.123,2.0,3.0
```

```
        SR[2]= SUBSTR(SR[0],0,R[0]-1)      ' Intercept left 1.123 SR[2]=1.123
```

```
        SR[0]= SUBSTR(SR[0],R[0]+1)      ' Intercept the right part of the string, after the
interception SR[0]=2.0,3.0
```

```
        R[0] = FINDSTR(SR[0],",")      ' SR[0]=2.0,3.0
```

```
        SR[3]= SUBSTR(SR[0],0,R[0]-1)      ' Intercept 2.0 on the left, SR[3] = 2.0
```

```
        SR[0]= SUBSTR(SR[0],R[0]+1)      ' Intercept the right part of the string, SR[0] = 3.0 after
interception
```

```
        SR[4]=SR[0]      'SR[4]=3.0
```

```
        SR[0]=""      'empty
```

```
        SEND(R[1],"OK")      'Give the visual uplink OK
```

```
        'convert to digital
```

```
        R[20]=TOVAL(SR[2])
```

```
        R[21]=TOVAL(SR[3])
```

```
        R[22]=TOVAL(SR[4])
```

```
    END IF
```

```
END IF
```

```
WAIT TIME= 10      ' Add 10ms delay to dead loop
```

```
END WHILE
```

```
R[0]=CLOSESOCKET HANDLE=R[1]      ' After jumping out of the loop, close the socket for
communication with the PLC.
```

```
LBL[100]
```

```
CLOSESOCKET HANDLE=1      ' Close the listening socket
```

```
<end>
```

10.4.3 SOCKET server-side one-to-many

```
<program>
```

```
R[0]=OPENSOCKET OPTIONS=0 HANDLE=1      'Create creates a server and listens for client
connections with handle 1.
```

```
IF R[0]<>0 THEN
```

```
    THROW "Failed to create server!" LEVEL=WARNING
```

```
    GOTO LBL[100]      'Ends the communication operation.
```

```
END IF
```

```
R[1]=-1      'Reset first. and R[1] is used to save the communication handle of the first client.
```

```
R[2]=-1      'Reset first. and R[2] is used to save the communication handle of the second client.
```

```
WHILE 1=1      ' Loop listening code, client can disconnect and reconnect with server (robot arm)
```

```
    R[10]=ISHANDLEOPEN(R[1])
```

```
    IF R[10]=0 THEN      ' If it is not connected, it listens for connections. No need to listen and
accept the value of fd if it is already open normally, otherwise it will overwrite the value of R[1].
```

```
        ACCEPT(1,"10.10.57.213",30000,R[1],100)      ' The fd that communicates with the first client is
stored in R[1], and is returned after a timeout of 100ms.
```

```
    END IF
```

```
    R[10]=ISHANDLEOPEN(R[1])
```

```
    IF R[10]=1 THEN      ' R[10]=1 indicates that the handle is open and communication
with the first client is started.
```

```
        SEND(R[1],"hello world")
```

```
        R[11]= BRECBUFEMPTY(R[1])      ' Determine if there is data in the receive buffer
```

```
        IF R[11]=1 THEN      ' The cache receives the data only when it's available.
```

```
            RECEIVE(R[1],SR[1])
```

```
        END IF
```

```
        '...      ' Business logic judgment is done here, character cutting and so on.
```

```
    END IF
```

```
R[10]=ISHANDLEOPEN(R[2])
```

```
IF R[10]=0 THEN      'If it is not connected, it listens for connections. No need to listen and accept
the value of fd if it is already open normally, otherwise it will overwrite the value of R[2].
```

```
    ACCEPT(1,"10.10.57.213",30000,R[2],100)      ' The fd for communicating with the second client
is stored in R[2], and is returned after a listening timeout of 100ms.
```

```
    END IF
```

```
    R[10]=ISHANDLEOPEN(R[2])
```

```
    IF R[10]=1 THEN      ' R[10]=1 indicates that the handle is open and communication with
the second client is started.
```

```
        SEND(R[2],"hello world")
```



```
RECEIVE(R[2],SR[2])
'...      ' Business logic judgment is done here, character cutting and so on.
END IF
WAIT TIME=10      ' Delay 10ms
END WHILE
R[0]=CLOSESOCKET HANDLE=R[1]      ' After jumping out of the loop, close the socket that is
communicating with Client 1.
R[0]=CLOSESOCKET HANDLE=R[2]      ' After jumping out of the loop, close the socket that is
communicating with Client 2.
LBL[100]
CLOSESOCKET HANDLE=1      ' Close the listening socket
<end>
```

11 Programming Instructions

11.1 summarize

Terminology and Concepts:

Program file

A program, or PRG program, is a text file with a “.PRG ” extension. A program file is a collection of system function instructions and statements that can be recognized and executed by the system.

The program file name supports numbers, letters, underscores, and Chinese characters.

The program file name should be within 30 bytes, and the program name is case sensitive.

The program file structure is divided into three modules: document attribute module, variable declaration module, program body module. Specific structure and Instructions As shown in Figure 1-1 and Table 1-1.

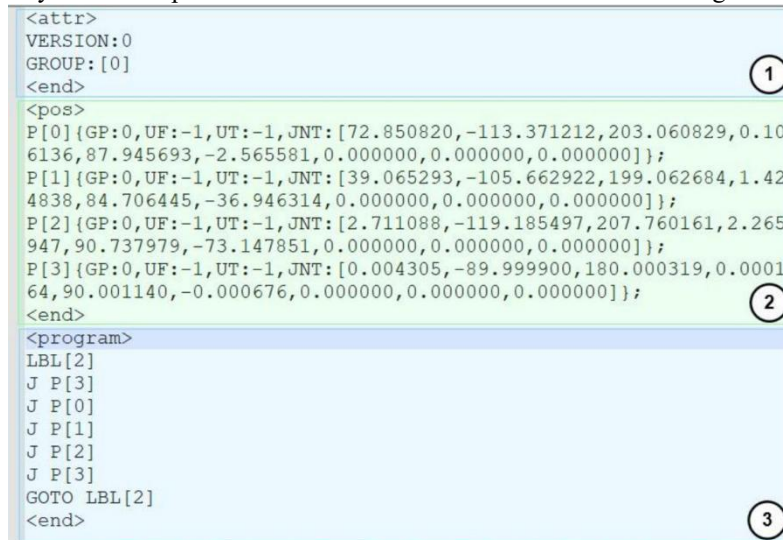


Figure 1-1 Schematic diagram of program file structure

Table 1-1 Instructions for Program File Structure Modules

Label Item	Instructions
1	Document attribute module. Used to record versions and the axis groups bound to them
2	Variable declaration module. Used to define and declare positional variables
3	Main module of the program. Used to add instructions and statements to

	implement programming logic

Main program and subroutines

The system supports PRG programs to call other PRG programs, with the main program as the caller and the called program as the subroutine. When calling, use the CALL instruction to make the call. Grammar example: CALL "program name". When the main program reaches this point, it will jump into the program content of the executing subroutine, and after execution, it will return to the main program to continue executing.

Front end program and backend program

The front-end program, also known as the main motion program, refers to a program with a group mask of 0 and motion instructions.

A background program, also known as a RUN program, refers to a program with a group mask of * that does not allow motion instructions and is only used for logical processing.

The system supports running both a front-end program and a back-end program simultaneously, and the two programs can run independently to meet the programming requirements of multitasking.

Loading

Loading refers to the operation of loading the contents of a program file into memory, and after the program is loaded, it can be executed.

Unload

Uninstalling refers to the operation of stopping the program from running and clearing its contents from memory.

Code annotation

Use single quotation marks ['] as comments for program lines in the program.

Line break connector

Use a reverse slash [\] as the line connector. The line connector is used when a single instruction line is too long to write, written at the end of the program line to connect adjacent lines of the program and concatenate them together to form a complete line of the program.

For example, there are two lines in the program:

```
A= B+
\ C+D
```

When executed, it is equivalent to one line of program: A=B+C+D.

Instruct

The instructions of the Type 3 control system can only be capitalized.

The system supports two basic data types: floating-point and string. Integer values will be implicitly

converted to floating-point values. Boolean and floating-point conversions follow the mandatory conversion rule of 'non-zero then 1'. Boolean numbers only support ON and OFF, and currently do not support TRUE and False.

11.2 Variable

11.2.1 Global variable

At present, the III system does not support custom variables, but the system predefines digital input/output IO, analog input/output IO, and six global register variables: R SR、JR、LR、UT、UF. Users can use these global registers in program programming for data computation and storage.

DI [x]: Digital input IO, boolean type, the system defines 512, namely DI [0]~DI [511]. VDI variables are virtual IO inputs used to switch the state of read-only variable DI [x] and implement virtual inputs.

DO [x]: Digital output IO, boolean type, the system defines 512, namely DO [0]~DO [511].

AI [x]: Analog input IO, float type, the system defines 16, namely AI [0]~AI [15].

AO [x]: Analog output IO, float type, the system defines 16, namely AO [0]~AO [15].

R [x]: A general-purpose double precision floating-point register variable, of type double, capable of accessing integer and floating-point data. The quantity is 2000, namely R [0]~R [1999]. The precision of variable values is up to 7 decimal places, and if exceeded, it will be rounded off.

SR [x]: A general-purpose string type register variable, each variable can store 512 characters, with a total of 100 characters, i.e. SR [0]~SR [99].

JR [x]: A universal joint point register variable used to access joint points. The quantity is 1000, namely JR [0]~JR [999].

LR [x]: A general-purpose spatial point register variable used for accessing spatial points. The quantity is 1000, namely LR [0]~LR [999].

UT [x]: Tool coordinate system register variable, used to store the values of the tool coordinate system. The quantity is 16, namely UT [0]~UT [15].

UF [x]: Workpiece coordinate system register variable, used to store the values of the workpiece coordinate system. The quantity is 16, namely UF [0]~UF [15].

Due to the fact that system variables are all global variables, the system will pre occupy them when implementing functions. Please refer to Appendix 32.2 "System Signal Occupancy Table" for details.

11.2.2 Program point variable

The program point is a point variable within the "<pos>...<end>" section of the PRG program. When the user obtains the point, the system will automatically add the point and record it in the section. The system supports two types of points: joint points and spatial points. The identification of joint points is JNT, and the identification of spatial points is LOC. When the program uses a spatial point, the tool part number specified in that point must be called, otherwise an alarm will be triggered indicating a mismatch in the coordinate system of the point.

The following example records P [0] as the joint point position and P [1] as the spatial point position:

```
<pos>
P[0]{GP:0,UF:-1,UT:-1,JNT:[3.126, -81.027, 29.643, 0.377, 82.950, -53.426, 0.0, 0.0, 0.0]};
P[1]{GP:0,UF:1,UT:1,CFG:[0, 0, 0, 0, 0, 0],LOC:[98.90, -994.70,296.30, -98.30, -3.60, -114.27, 0.0,
0.0, 0.0 ]};
<end>
<program>
J P[0]
UFRAME_NUM = 1
UTOOL_NUM= 1
L P[1]

...
<end>
```

11.2.2.1 Point variable component

The positional variable is a composite data structure, and the corresponding component can be obtained through the square brackets [] to manipulate the value of the corresponding bit. A positional variable contains data for 9 axes, therefore the subscripts are 0~8. The component data of the point is floating-point data, which supports operations such as addition, subtraction, multiplication, and division.

Example:

```
R[1]=JR[1][0]
LR[1][1]=R[0]*10
P[0][2]=P[0][2]+10
```

1.3 Point operation

Point variables can be understood as vectors, supporting operations such as addition, subtraction,

multiplication, division, and remainder of vectors.

Example:

$LR[1]=LR[2]+LR[3]$

The above statement adds the components of LR [2] and LR [3] and assigns them to the components of LR [1]. Similarly, support operations such as $LR [1]=LR [2] - LR [3]$, $LR [1]=LR [2] * LR [3]$, etc.

11.4 R Register bit operation

The R register stores an 8-bit signed integer, and users can obtain the value of each bit of the binary number corresponding to the integer through the dot. Due to the floating point type of the R register, if the actual value in the R register is not an integer, the system will default to rounding down to an integer before performing a bit operation.

Grammar format: $R[x].y$

Example:

IF $R[1].0=ON$ THEN 'Obtain the value of the 0th digit of R [1]

...

ENDIF

$R[1].0=0$ 'Set the 0th value of R [1] to 0

11.5 Operator

The operators supported by the current system are as follows:

Table 3-1 Operators Supported by the System

Operator category	Operator	Describe	Support data types	Priority	Example
Arithmetic operator	+	Addition operation	INT 、 DOUBLE、 STRING 、 POINT	3	$7=5+2$
	-	Subtraction operation	INT 、 DOUBLE、 POINT	3	$3=5-2$
	*	Multiplication	INT 、 DOUBLE、	2	$10=5*2$

		operation	POINT		
	/	Division operation	INT 、 DOUBLE、 POINT	2	2.5=5/2
	DIV	The integer whose quotient is obtained by division operation	INT 、 DOUBLE、 POINT	2	2=5 MOD 2
	MOD	Division operation takes remainder (modulo)	INT 、 DOUBLE、 POINT	2	1=5 MOD 2
Logical operator	AND	Logical union (one false is false)	BOOL	5	R[0]=R[1] AND R[2]
	OR	Logic or (one truth is true)	BOOL	5	R[0]=R[1] OR R[2]
	NOT	Logical inversion	BOOL	1	R[0]=NOT R[2]
Comparison operator	=	Equal to	INT 、 DOUBLE、 POINT	4	1=1
	◇	Not equal to	INT 、 DOUBLE、 POINT	4	1◇2
	<	Less than	INT 、 DOUBLE	4	1<2
	>	Greater than	INT 、 DOUBLE	4	2>1
	<=	Less than or equal	INT 、 DOUBLE	4	1<=1
	>=	Greater than or equal	INT 、 DOUBLE	4	1>=1

Tips:

- (1) When using logical operators, floating-point data is implicitly converted to Boolean data, i.e. 'non-zero then 1'.
- (2) Combine operators of the same priority from left to right.
- (3) NOT: The logical operation is reversed. The highest priority.

11.6 Statement

11.6.1 Process statement

11.6.1.1 LBL[] | GOTO LBL[]

The LBL [] tag statement is used to set the tag position, and the GOTO statement is used to jump the program to the specified tag position. If the GOTO keyword is to be used, the label LBL must be defined in the program first, and GOTO and LBL must be in the same program block.

Grammar format:

LBL “[”<unsigned int num>“”]

...

GOTO LBL “[”<unsigned int num>“”]

Example:

```
LBL[1]
J P[1] VEL=50
J P[2] VEL=50
GOTO LBL[1]
```

Execution Logic Manual: After the robot joints sequentially to P [1] and P [2], it executes GOTO

LBL [1], then jumps to LBL [1], and then joints sequentially to P [1] and P [2], continuously reciprocating these two points.

11.6.1.2 CALL

The CALL instruction is used for calling subroutines and executing their program content. Program calls support multiple layers of nesting, up to a maximum of 12 layers.

Grammar format:CALL <string name>

Example, Main Program MAIN PRG calls subroutineSON.PRG:

Here is the MAIN.PRG (Main Program)

<program>

```
J JR[1] VEL=50
J JR[2] VEL=50
CALL "SON.PRG" 'Call subroutine
J JR[3] VEL=50
```

<end>

Here is the SON.PRG (subroutine)

<program>

```
DO[1]=ON
```



```
WAIT TIME 500
DO[1]=OFF
<end>
```

Execution logic Instructions: The robot moves to JR [1], JR [2] in turn, then calls the subprogram, executes DO [1]=ON, after forced waiting for 500 milliseconds, executes DO [1]=OFF, then returns to the main program, the joint moves to JR [3], and completes program execution.

If the subroutine is in another file directory, when the CALL instruction is called, the file directory name must be added, such as CALL "FLOD1/SON. PRG"

11.6.2 Conditional statement

11.6.2.1 IF... GOTO LBL[]

If the condition is met, jump to the corresponding LBL tag for execution; If the condition is not met, the program blocks following IF will be executed sequentially.

Grammar format:

IF <bool expression>, GOTO LBL["<unsigned int num>"]



Example:

```
IF DI[1]=ON, GOTO LBL[1]    'If the condition is true, jump directly to LBL
[1] for execution; If the condition is false, execute in sequence downwards
J P[1] VEL=50
J P[2] VEL=50
LBL[1]
VDI[1]=OFF
.....
```

11.6.2.2 IF... CALL

If the condition is met, the code content of the subroutine will be called and executed before proceeding in sequence; If the condition is not met, execute the next line of instruction content of the IF instruction and ignore the subroutine call.

Grammar format:

IF <bool expression>, CALL<string name>



Example:

```
IF DI[1]=ON, CALL "TEST.PRG"
J JR[1] VEL=50
```

DO[1]=OFF

Execution Logic Manual:

If DI [1]=ON condition is met, execute TEST The program content of the PRG subroutine, after execution, proceeds to execute JJR [1] VEL=50 and subsequent

instructions; If the conditions are not met, ignore TEST The subroutine call of PRG directly executes JJR [1] VEL=50 and subsequent instructions.

11.6.2.3 IF... THEN... ENDIF

IF THEN needs to be used in conjunction with END IF instruction, IF... GOTO, IF... CALL instructions do not need to be used in conjunction with END IF. IF THEN can be used in conjunction with the ELSE instruction to execute the ELSE branch when the condition is not met.

Grammar format:

IF <bool expression> THEN

...

[ELS

E ...]

ENDIF



Example 1 , No ELSE branch:

IF DI[1]=ON THEN 'If DI [1]=ON is satisfied, execute the JP [1] instruction; otherwise, do not execute it

J P[1]

END IF

Example 2 , Used in conjunction with ELSE branch:

IF DI[1]=ON THEN 'If DI [1]=ON is satisfied, execute J P [1]; otherwise, execute J P [2]

J P[1]

ELSE

J P[2]

END IF

11.6.2.4 SELECT

The SELECT statement compares the value stored in the register selected by SELECT with the value after CASE. If they are equal, execute the CASE flow instruction in that row, and at the same time, the SELECT statement block will pop up. The CASE and ELSE instructions will no longer be executed; If they are not equal, the corresponding CASE for that line will not be executed, and the subsequent CASE option judgment will continue until the entire statement block is completed

Grammar format:

SELECT <num expression>

CASE<num expression>, (GOTO LBL“[”<unsigned int num>“]”)(CALL <string name>)

[{CASE<num expression>, (GOTO LBL“[”<unsigned int num>“]”)(CALL <string name>)}]

ELSE, (GOTO LBL“[”<unsigned int num>“]”)(CALL <string name>)]



Example:

```
SELECT R[0]
CASE 1, GOTO LBL[1]
CASE 2, GOTO LBL[2]
CASE 1, CALL"RT.PRG"      'This is a repeated CASE situation. IfR [0]=1,
then only execute the previous LBL [1] once
ELSE, CALL"HS.PRG"      'The ELSE instruction is optional, can be written or
not written
GOTO LBL[4]
LBL[1]
J P[1]
GOTO LBL[4]
LBL[2]
J P[2]
GOTO LBL[4]
LBL[3]
J P[3]
LBL[4]
```

Execution Logic Manual:

The above program is executed from top to bottom to match R [0]. When R [0]=1, the program moves to point P [1]; When R [0]=2, the program moves to point P2; When R

[0] is not equal to 1 or 2, the program executes the subroutine HS of the ELSE instruction part. The content of PRG is executed in descending order.

11.6.3 Loop statement

11.6.3.1 FOR... END FOR

The FOR statement needs to define the initial value, final value, and step value of a variable (BY expression is optional, if not written, it defaults to increasing by 1 each time). The system will determine whether the value of the loop variable is less than or equal to the final value. If it is less than or equal to the final value, the loop will be executed. Otherwise, the loop will exit. FOR and END FOR form a loop statement block.

Grammar format:

FOR<num expression1> TO<num expression2> [BY <num expression3>]

[<loop statements>]

END FOR



Example:

```
R[2]=0
FOR R[1]=0 TO 2 BY 1
R[2]=R[2]+1
END FOR
```

Execution Logic Manual:

The initial value of the loop body variable R [1] is 0, the final value is 3, and the step value is 1, so each loop increases by 1. When making the first judgment, R [1]=0, not greater than the final value, perform the first loop, R [2]=1; When making the second judgment, R [1]=1, not greater than the final value, proceed to the second loop, R [2]=2; When making the third judgment, R [1]=2, not greater than the final value, proceed to the third loop, R [2]=3; When making the fourth judgment, R [1]=3, greater than the final value, exits the loop.

So a total of four judgments were made and three cycles were repeated. After the program finishes running, R [1]=3, R [2]=3.



Tips:

If "dead loop" logic is required during the application process, it is recommended to add a WAIT TIME=50/100 delay instruction in the loop statement to avoid possible logic handling exceptions caused by program execution being too fast.

11.6.3.2 WHILE... END WHILE

The WHILE statement determines whether the loop has ended based on a conditional expression. When the condition is true, continue cycling; When the condition is false, exit the loop. The WHILE statement and END WHILE form a loop statement block.

Grammar format:

```
WHILE <bool expression>
[<loop statements>]
END WHILE
```



Example:

```
R[1]=0
WHILE R[1]<3
R[1]=R[1]+1
END WHILE
```

Execution Logic Manual:

When making the first judgment, R [1]=0, less than 3, perform the first loop; When making the second judgment, if R [1]=1 and is less than 3, proceed to the second loop; When making the third judgment, if R [1] is 2 and less than 3, proceed to the third loop; At the fourth judgment, R [1]=3, not less than 3, exits the loop. It cycled three

times in total.



Tips:

If "dead loop" logic is required during the application process, it is recommended to add a WAIT TIME=50/100 delay instruction in the loop statement to avoid possible logic handling exceptions caused by program execution being too fast.

11.6.3.3 BREAK

The BREAK statement is used to exit the FOR or WHILE loop.

Grammar format:

<BREAK>



Example:

```
WHILE R[1]=0
....      'program content
BREAK      'Break
END WHILE

FOR R[1]=0 TO 2 BY 1
....      'program content
BREAK      'Break
END FOR
```

11.7 Motion command

The motion command specifies the type of action for the robotic arm to move. There must be POINT information after the motion command, and the motion parameters of the segment can be specified. If not specified, the system's default motion parameters will be used. Additional motion attributes can be added after the motion command to control the logical operation of the motion segment.

Grammar format:

<Motion Type> <TargetPoint> [Motion Parameter][Additional Property]



Example:

```
J P[1] VEL= 100, OFFSET JR[1]
```



Tips:

When the target points of multiple consecutive lines of motion instructions are the same POINT, the system will forward process and ignore these repeated POINT.

11.7.1 J joint motion command

The joint motion J command starts from the current actual position of the robotic arm (or external axis) and moves the robotic arm (or external axis) to the target POINT position. The movement process does not involve trajectory and attitude control, meaning all joints start and stop simultaneously.

Grammar format:

J <TargetPoint> [Motion Parameter][Additional Property]



Example:

J P[1] VEL=50 ACC= 100 DEC=100 'Specify speed, acceleration, and deceleration

J P[2] 'Use system default parameters



Tips:

The target POINT of joint motion can be selected in two formats: joint point and spatial point, namely J JR [x] and J LR [x]. When executing J JR [x], it will move to the joint angle specified by the target point; When executing JLR [x], due to the multi solution problem of spatial POINT, there will be multiple joint inverse solutions. The system will select the values of joint axes 1, 3, and 5 based on the posture angle specified in the shape position CFG. For the rotation numbers of axes 4 and 6, the system defaults to the shortest path strategy, but users can use the value of the motion parameter TURN to specify the direction of rotation of the end axis.

11.7.2 L Linear motion command

The linear motion L command starts from the current position of the robotic arm and moves in a straight line to the target point, used in situations where trajectory control is required.

Grammar format:

L <TargetPoint> [Motion Parameter][Additional Property]



Example:

L P[1] VEL=50 ACC= 100 VROT=50 'Specify position speed, position acceleration, attitude rotation speed



Tips:

- (1) When moving in a straight line, the shape of the robotic arm cannot be changed. If the shape of the starting point is inconsistent with that of the target point, an alarm will be triggered (because shape changes often pass through singular points), and the motion will stop. But for situations where the wrist shape is inconsistent, the WRISTJNT command can be used to navigate through the wrist

singularity zone.

- (2) When moving in a straight line, the changes in position and posture are uniform and synchronized, so setting an unreasonable posture speed may affect the final movement rhythm.
- (3) When moving in a straight line, the target POINT has two formats for nodes and spatial points, namely L JR [x] and L LR [x]. When executing LJR [x], the spatial point corresponding to the joint position will be moved, so there may be a situation where the actual joint position of the robotic arm when it reaches the target point is inconsistent with the target POINT recorded in the program.

11.7.3 C arc motion command

The circular motion C command takes the current position as the starting point, MIDDLEPOINT as the midpoint, and TARGETPOINT as the endpoint, controlling the robot to perform circular trajectory motion in Cartesian space (three points form a circular arc). The arc command cannot draw a whole circle. If you want to draw a whole circle, you need to use two arc C commands.

Grammar format:

C <MiddlePoint> <TargetPoint>[Motion Parameter][Additional Property]



Example:

'A section of circular arc is written as follows

L P[1] 'Starting point of arc

C P[2] P[3]

'The whole circle is written as follows

L P[1]

C P[2]P[3]

C P[4]P[1]

The legend for the whole circle is as follows:

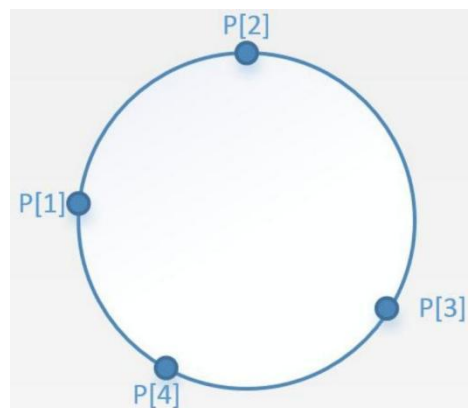


Figure 5-1 Schematic diagram of the whole circle POINT



Tips:

- (1) Circular motion is also a spatial motion that satisfies the constraint of linear motion L instruction, that is, the motion does not change the shape, and when the target point is a joint point, it will move to its corresponding spatial point, with synchronized position and attitude velocity.
- (2) The circular motion does not consider the attitude constraint at the midpoint, but only considers the attitude of the target point, that is, the attitude at the midpoint during the circular interpolation process may not necessarily be the same as during teaching.

Note: When opened in manual mode or online editing, the [Record] function in the arc C command is invalid. It needs to be modified by clicking on the command and moving the cursor to the input box for recording the required data to record POINT; The point function is to move the arc from the first point to the second point.

11.7.4 A Arc Motion Command

The multi segment arc A instruction can be used to implement multiple arcs of different radii, which are used to splice multiple irregular curves. The positions specified by three consecutive A instructions in the program line can form a circular arc. When using four consecutive A commands, it is possible to achieve arcs with radius X for the first three points and radius Y for the last three POINT points. POINT addition and deletion can be performed between two arcs, and different motion additional attributes can be set.

There is no pause between the movements of the two arcs.

Grammar format:

A <TargetPoint>[Motion Parameter][Additional Property]



Example:

```
J P[1] VEL = 100
A P[2] VEL = 500
A P[3] VEL = 500
A P[4]

A P[5]
L P[6]
```

Execution Logic Manual:

The execution trajectory of the above program is shown in Figure 5-2. When executing the first circular motion A command (line 2), the robot's motion mode is linear motion, that is, from the robot's current position P [1] to the target point P [2], from P [2] to P [3] (circular arc P2P3P4), from P [3] to P [4] (circular arc P3P4P5), from P [4] to P [5] (circular arc P4P5P3), and from P [5] to P [6].

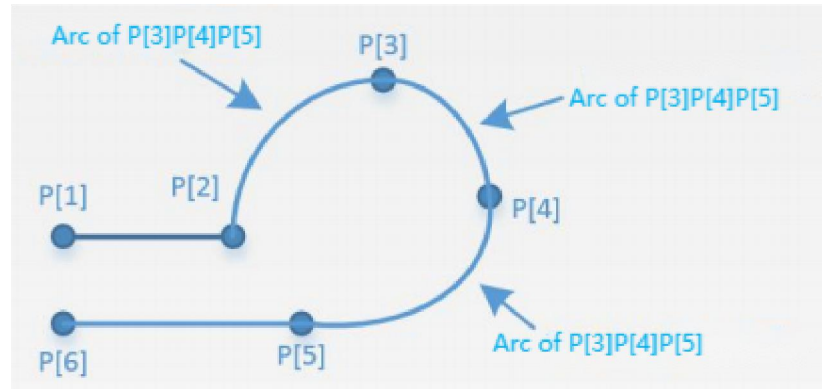


Figure 5-2 Schematic diagram of program trajectory for example A instruction



Tips:

When there are less than 3 consecutive A commands, the robotic arm will move in a straight line to the target point, where the A command is equivalent to the L command.

11.7.5 JUMP Gate Motion Command

The "gate shaped" motion JUMP instruction is to teach only one target point, and the robotic arm can move from the current position "gate shaped" to the target POINT. During motion, the robotic arm will perform three trajectory movements. The first trajectory is a vertical upward movement, the second trajectory is a joint movement on a specified plane, and the third trajectory is a vertical downward movement. The three trajectories are smoothly connected.

Grammar format:

```
JUMP <TargetPoint> LIMZ=<num expression> ARCH_A=<num expression>
ARCH_B=<num
expression>
```

LIMZ: represents the position value of the SCARA robotic arm's 3-axis (unit: mm), and the robotic arm performs joint motion on the z-plane corresponding to the height of this position value. Due to the value limit of SCARA 3-axis being [A3. PMIN, 0], this value is less than 0.

ARCH_S: represents the safe ascent distance (unit: mm), which means that the current position of the robotic arm rises the specified distance, then smoothly turns into the plane specified by the LIMZ parameter, and performs the second joint movement.

ARCH_D: represents the safe descent distance (unit: mm), which means that the robotic arm descends to the specified height and reaches the target point. The robotic arm smoothly turns out from the z-plane and then descends vertically to the target point.



Example:

```
JUMP P[1] LIMZ=-20 ARCH_A=50 ARCH_B=80
```

The execution trajectory of the above program is shown in Figure 5-3.

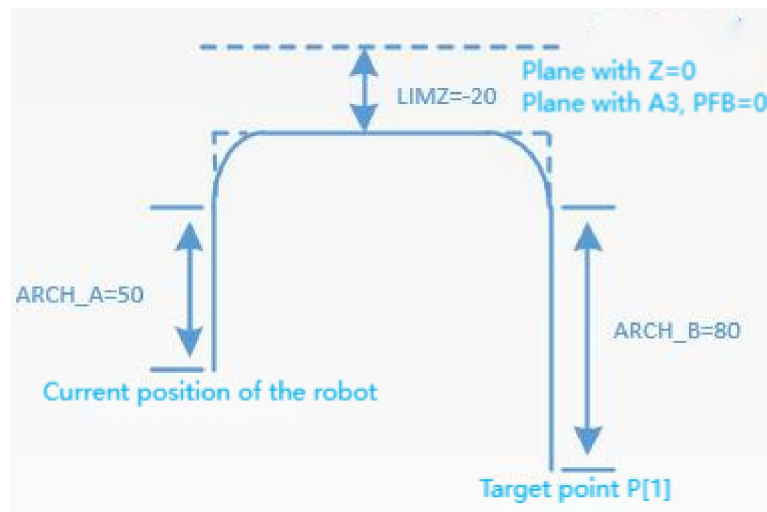


Figure 5-3 Schematic diagram of JUMP instruction example program trajectory



Tips:

The JUMP command can only be used on SCARA robotic arms and cannot be used on six axis robots.

11.7.6 REULMOVE Square hole machining motion command

The Lelo triangle motion can be used to achieve mixing operations inside square holes, requiring only one point to be taught, and mixing will almost stick to every corner. When using this command, a stirring fixture must be installed at the end of the robotic arm TCP. The end of the robotic arm TCP will spiral downwards while rotating, and spiral upwards after reaching the specified depth. After reaching the limit, the 6-axis will automatically reverse (i.e. forward and reverse within the limit). After reaching the designated time, the stirring action stops.

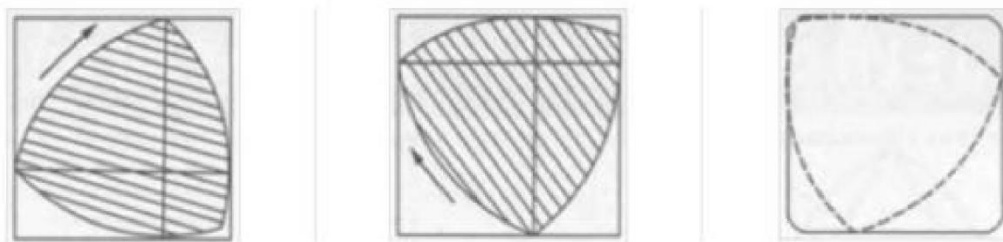


Figure 5-4 Schematic diagram of Lelo triangle motion

Grammar format:

REULMOVE SLEN=<num> DEPTH=<num> VEL=<num> FVEL=<num> TIME=<num>

SLEN: The side length value of the directional hole, measured in millimeters.

DEPTH: Depth of the hole, measured in millimeters.

VEL: End rotation speed, measured in degrees per second.

FVEL: Feed rate, measured in mm/s.

I TIME: The time of movement, measured in milliseconds.



Example:

```
L P[1]      'Move in a straight line to point P1, which is the center point of the
Lelot triangle
REULMOVE SLEN=20 DEPTH= 10 VEL= 10 FVEL= 10
TIME=3000   'Start stirring, timed for 3 seconds
```

11.8 Local motion parameters

Local parameters are motion parameters written after the instruction line, which control the characteristics of the trajectory segment. Local parameters are optional. If not written, the system uses default global parameters. If filled in, local parameters are used. The priority of local parameters is higher than that of global parameters.

Table 6-1 List of Local Motion Parameters of the System

Parameter	Meaning	Value Range	Example usage
VEL	Position speed	Floating positive integer. The J command corresponds to percentages ranging from 1 to 100, while the L/A/C command corresponds to spatial velocity values, measured in mm/s	J P[1] VEL= 100 L P[2] VEL=500
VROT	Attitude speed	Floating positive integer. Effective only for L/A/C commands, in degrees per second	L P[2] VEL=50
ACC	Acceleration	Floating positive integer. Unit is percentage	L P[2] ACC=50
DEC	Deceleration	Floating positive integer. Unit is percentage	L P[2] DEC=50
SMOOTH	Acceleration compliance level	Unsigned integer. The value range is 1~9, and the larger the value, the smoother the acceleration	L P[2] SMOOTH=50

Grammar format:

<Motion Type> <TargetPoint> {[Motion Parameter]}



Example:

```
J P[1] VEL= 100 ACC= 100   'VEL= 100 and ACC= 100 are local parameters
```

11.9 Global motion parameters

11.9.1 Global speed parameter table

The global motion parameters set the characteristics of all motion commands in the program during motion, and the effective range is the entire PRG program. By setting global motion parameters, it is possible to quickly adjust trajectory rhythms and other parameters without the need to input local parameters after each command line for control. The global motion parameters only take effect during the execution of this PRG and do not affect other PRG programs.

Table 7-1 List of Global Motion Parameters of the System

Parameter	Meaning	Value setting range	Default value
J_VEL	Joint speed (effective for J motion command)	1~100, in percentages, and divided according to the pattern: Manual T1:1~10 Manual T2:1~20 Automatic/External: 1~100	100
J_ACC	Joint acceleration ratio (effective for J motion commands)	1~200, expressed as a percentage	100
J_DEC	Joint reduction ratio (effective for J motion command)	1~200, expressed as a percentage	100
L_VEL	Linear velocity (linear L motion command)	According to the mode limitation: Manual T1: 1~125 mm/s Manual T2: 1~250 mm/s Automatic/External: 1~2400 mm/s, maximum value of 2400. The vtran value can be modified in the group speed parameter file group [0] -velocity.data file, which is different for each model	Maximum value
L_ACC	Linear acceleration ratio (linear L motion command)	1~200, expressed as a percentage	100
L_DEC	Linear deceleration ratio (linear L motion command)	1~200, expressed as a percentage	100
L_VROT	Linear attitude velocity (linear L motion command)	According to the mode limitation: Manual T1: $1 \sim 30^{\circ} / s$ Manual T2: $1 \sim 80^{\circ} / s$ Automatic/External: $1 \sim 20^{\circ} / s$, maximum	Maximum value

		value of 120. The value of vrot can be modified in the group speed parameter file group [0] -velocity.data file, which is different for each model	
C_VEL	Arc velocity (motion commands for arcs C and A)	According to the mode limitation: Manual T1: 1~125 mm/s Manual T2: 1~250 mm/s Automatic/External: 1-2400 mm/s, maximum value of 2400. The vtran value can be modified in the group speed parameter file group [0] -velocity.data file, which is different for each model	Maximum value
C_DEC	Arc reduction ratio (arc C, A motion command)	1~200, expressed as a percentage	100
C_ACC	Arc Acceleration Ratio (Arc C, A Motion Command)	1~200, expressed as a percentage	100
C_VROT	Arc attitude velocity (motion commands for arcs C and A)	According to the mode limitation: Manual T1: 1~30 °/s Manual T2: 1~80 °/s Automatic/External: 1~120 °/s, maximum value of 120. The value of vrot can be modified in the group speed parameter file group [0] -velocity.data file, which is different for each model	Maximum value
SMOOTH	Acceleration compliance level, can be used to adjust rhythm and jitter	Unsigned integer. The value range is 1~9, and the larger the value, the smoother the acceleration	5

Grammar format:

<Global Motion Parameter>=<Num Expression>



Example:

J_VEL= 100 'The following four are global parameters
J_ACC= 100
L_ACC= 120
SMOOTH= 1
J P[1]
J P[2]
L P[3]

11.9.2 SMOOTH Flexibility level parameters

To meet the adjustment requirements of rhythm and jitter in different scenarios, the system provides SMOOTH parameters to control the speed of acceleration changes in the robotic arm. SMOOTH value range: 1~9, default value 5. The larger the value, the smoother it is, and the slower the acceleration changes. SMOOTH=1 indicates that the acceleration changes the fastest during the start stop phase, which may cause shaking; SMOOTH=9 indicates that the acceleration changes the slowest and smoothest during the start stop phase.



Example:

```
SMOOTH=1    'Global parameters
J P[1]
J P[2]
J P[3] VEL= 100 ACC= 100 SMOOTH=5    'Local
parameter J P[4]
```

11.10 Trajectory control parameters

Trajectory control parameters are used to control the motion trajectory and motion strategy of the robotic arm. Similar to motion parameters, trajectory control parameters also have global and local distinctions. Users can choose trajectory parameters according to their actual application needs to meet application requirements.

Table 8-1 Trajectory Control Parameter List

Parameter	Meaning	Value Range	Default value
CNT_TYPE	Smooth type	Unsigned integer 0 1 2	0
CNT	Smooth parameter percentage	Floating point positive integer 0~100	0
CR	Smooth corner radius value, in millimeters	Floating point positive integer 1~100	0
TURN	End axis rotation control options	Unsigned integer 0 1 2 3 4	0
TALARM	End axis rotation control alarm switch	Unsigned integer 0 1	0

11.10.1 CNT_TYPE Smooth type parameter

CNTTYPE is a smooth type parameter. There are currently two different smoothing types in the system, and different values of CNTTYPE indicate the use of different smoothing types:

- (1) CNTTYPE=0 or 1 is the speed smoothing mode, and the robotic arm adopts a speed mixing method to "turn" into the second trajectory with the current speed as the initial condition.
- (2) CNTTYPE=2 is the trajectory smoothing mode, and the system uses Bezier curves to smoothly concatenate two trajectory segments.

The difference between the two smoothing methods is that the CNTTYPE=2 smoothing method does not

change the trajectory accuracy of the smooth transition segment in the following three situations:

- (1) Modify the speed multiplier (VORD) at which the program runs.
- (2) Modify the VEL speed parameter of the instruction.
- (3) Pause and restart during smooth trajectory segment.

Smooth speed has the characteristic of wide adaptability, and can have good smoothing effects on joint+line, joint+arc and other situations. However, changing the speed will change the smooth trajectory. Trajectory smoothing has the characteristic of not deforming the trajectory, but there may be constraints in some cases. Therefore, when debugging or pausing startup, it is required that the smoothed trajectory does not change, and trajectory smoothing can be used.

Grammar format:

<CNT_TYPE>=<0|1|2>



Example:

```
CNT_TYPE=1
CNT=100

J P[1]
L P[2] CNT=50

C P[3] P[4]
J P[5]
```



Tips:

- (1) When the CNTVNet instruction is not written to specify the smoothing type, the system defaults to the default value of 0 for CNTVNet, which is the speed smoothing method.
- (2) CNT_TPYE specifies the smoothness type, while CR and CNT specify the smoothness value. Therefore, both CR and CNT attributes can be used when velocity smoothing CNTTYPE=1 or trajectory smoothing CNTTYPE=2, and their meanings remain unchanged.

11.10.2 CNT Smooth transition parameter

CNT parameters can be used for smooth path stitching of two motion trajectories, reducing motion pauses. Its value is a percentage (ranging from 0 to 100). The meaning of CNT parameter value: the ratio of the time occupied by the smooth trajectory to half of the total motion time T of the trajectory segment. T is the total time required for the trajectory to move. Due to the positive correlation between the length of the trajectory segment and the running time t, the proportional relationship between the trajectory points at a certain moment and the total trajectory can be used to visually describe the CNT properties of the system.

Syntax format 1 (global parameters):

```
CNT=<1~100>
J P[1]
J P[2]
J P[3]
```

Syntax format 2 (local parameters):

J P[1]

J P[2] CNT=100

J P[3]



Example:

L P[A]

L P[B] CNT=50

L P[C]

Execution Logic Manual:

The trajectory path is $A \rightarrow B \rightarrow C$, as shown in Figure 8-1. Using the smooth transition function at point B, the robotic arm turns in at point M and out at point N. The attribute value of CNT means $CNT = BM / (AB/2) * 100$. Then:

$$CNT=30 = t_1 / (T/2)$$

$$CNT=50 = t_2 / (T/2)$$

$$CNT=100 = t_3 / (T/2)$$

The larger the CNT value, the smoother the trajectory of the robotic arm starts from the midpoint of the AB segment trajectory. When the CNT value is maximum, the system starts smoothing from the midpoint of the trajectory.

When $CNT=0$, it indicates that there is no smoothness, and the motion of the robotic arm will pass through point B, and the velocity at point B will decrease to 0 before continuing with the next stage of motion.

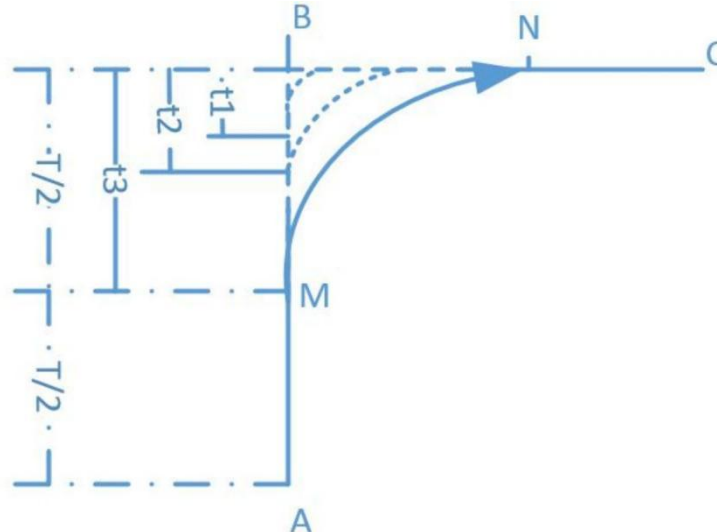


Figure 8-1 Schematic diagram of smooth trajectory of example program



Tips:

When the speed is smooth, the CNT value can be set reasonably to achieve good smoothing effect.


```
L P[1]
L P[2] CNT=20/50/100
L P[3]
```

- (1) When a long trajectory is shortened, the CNT value can be set to a smaller value.

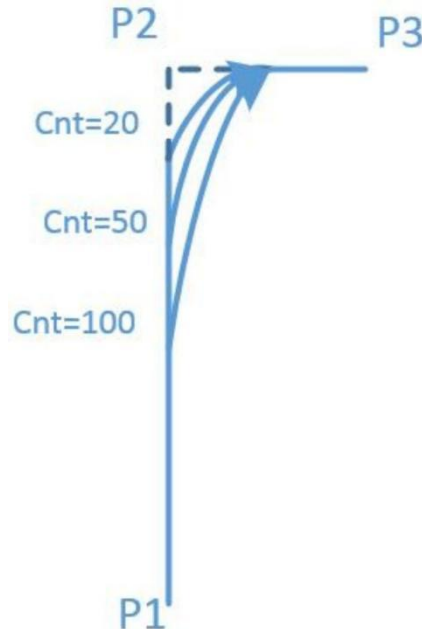


Figure 8-2 Schematic diagram of different CNTs for short and long trajectories

- (2) When a short trajectory is extended to a long trajectory, the CNT value should be set to a larger value.

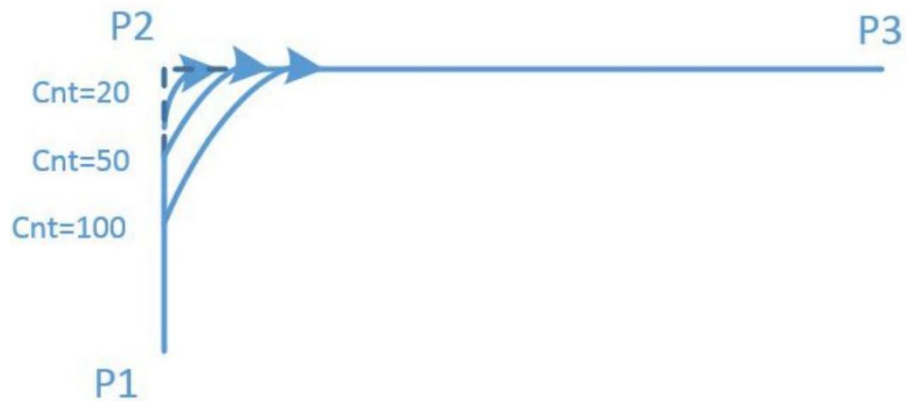


Figure 8-3 Schematic diagram of different CNTs for short and long trajectories



Tips:

- (1) The relationship between signal output and smooth transition points suggests:

```
L P[A]
```

```
L P[B] CNT=50
```

```
DO[1] = ON
```

```
J P[C]
```

'At this point, the signal is output at transition point M

- (2) The relationship between WAIT waiting conditions and smooth transitions suggests:

```
L P[A]
```

```
L P[B] CNT=50
```

WAIT DO[1] = ON 'If the DO [1] signal is ON before the transition point M, there is smoothing. If the WAIT condition is not met, it will stop at point P [B] and wait for the condition to be met. At this time, there is no smoothing action

```
J P[C]
```

- (3) Boundary Limit Instructions: Typically, when CNT reaches its maximum value, the trajectory transition starts from the midpoint of the trajectory.
- (4) Adaptive Instructions: When the trajectory is less than 5mm, the smooth planning performs adaptive processing, and the smooth trajectory first meets the requirements of high-speed smoothness. It is not strictly constrained by the current CNT smoothing value and may start smoothing from any point on the trajectory.
- (5) Pause Start Instructions: Pause during the smoothing process. When starting the motion, it will run directly to the final POINT position and will not follow the old smooth trajectory.
- (6) Does not support smoothing between J | L | A | C instructions and JUMP instructions. Example program instructions are as follows:

```
L P[1] CNT=50, PASS
```

'No smoothness, accurate point P [1]

```
JUMP P[2] LIMZ=-10 ARCH_A=40 ARCH_B=60
```

11.10.3 FINE Attribute

The FINE positioning attribute is used to accurately move the robotic arm to the target POINT position and stop, for precise control of the robotic arm to the point. Due to the delay of servo position feedback, when the position command controls the robotic arm to reach the target point, the actual position of the robotic arm may not be in place yet. Therefore, it is necessary to wait until it is determined that the robotic arm is in place before executing the next movement.

Grammar format:

<Motion Type><Target Point>[CNT=FINE]



Example:

```
L P[0]
```

L P[1] CNT=FINE 'First move in a straight line to P [0], then move

in a straight line to point P [1] and wait for the feedback position to be in place, then move in a straight line to point P [2]

L P[2]



Tips:

CNT=0 and CNT=FINE both indicate that the speed of the robotic arm at that point drops to 0. However, CNT=0 means that the commanded speed at that point drops to 0 but does not wait for servo feedback to reach 0 before starting the next stage of motion. CNT=FINE means moving to that point and waiting for position feedback to be in place.

11.10.4 CR Smooth transition parameter

In trajectory based applications such as gluing and polishing, there are geometric requirements for the smooth stitching of two trajectories, which require the ability to specify the position of the turning point. In this case, users can use the CR attribute function, which specifies the distance between the turning point and the target point during smooth transition, with a positive value in millimeters (mm). The larger the value, the larger the corner radius, and the farther the smooth trajectory's turning point is from the target point.

Grammar format:

<Motion Type><Target Point>[CR=<num expression>]



Example:

L P[1]
L P[2] CR=20
L P[3]

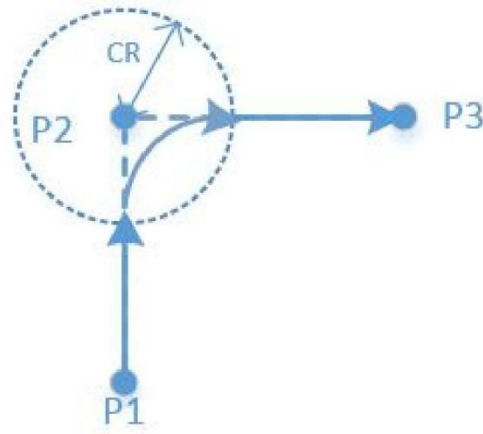


Figure 8-4 Schematic diagram of smooth trajectory of example program



Tips:

- (1) CR is a positive number, meaning the value is greater than 0. If the input is incorrect, an alarm must be triggered.
- (2) When the CR value is less than half of the short side, the actual radius value of the corner path is the arc of CR.
- (3) When the value of CR is greater than half of the short side, the actual corner radius is equal to half of the short side. However, due to velocity smoothing, the actual turning point may change depending on the velocity value when a long line segment is connected to a short line segment, resulting in corresponding changes in the smooth trajectory.
- (4) The CR attribute supports L, C, and A instructions, but does not support JUMP instructions. Writing it will not trigger an alarm, but it will have no effect.
- (5) In the case of two-point turning motion or three-point collinearity, the trajectory characteristics will cause the value of CR to be ineffective, and the motion will be precise to the point.
- (6) The CR and CNT attributes can be used when the velocity smoothing CNTTYPE=1 or the trajectory smoothing CNTTYPE=2. CNT_TPYE specifies the smoothing type, while CR and CNT specify the smoothness value.

11.10.5 TURN | TALARM End axis control parameters

TURN and TALARM are used to control the rotation direction of the end axes (6-axis of 6-joint robotic arm and 4-axis of SCARA) when the robotic arm executes JLR [x]. Due to the fact that the inverse solution of the system space points can correspond to multiple joint points, there are issues with form positioning and solution selection.

The meanings of the various options for the TURN parameter are as follows:

TURN=0, which means the direction with the smallest angle rotation change of the spatial equivalent

rotation axis. If the inverse solutions of axes 4 and 6 exceed the limit in this direction, the reverse solution will be automatically selected, which is the default option of the system.

TURN=1, which means the direction where the angle rotation of the spatial equivalent rotation axis changes the most is the opposite direction when TURN=0.

TURN=2, ensuring that the end axis is within $\pm 180^\circ$, without considering the rotation direction of the equivalent rotation axis. Under this option, there may be forward and reverse rotation of the end axis, but it will not exceed ± 180 degrees.

TURN=3, the end shaft can only rotate in the positive direction. If the joint of the end shaft rotating in the positive direction exceeds the limit, an alarm will be triggered.

TURN=4, The end axis can only rotate in the opposite direction. If the joint of the end axis reversing exceeds the limit, an alarm will be triggered.

The TALARM parameter refers to whether the system is allowed to automatically select the option value of TURN to ensure the normal operation of the system, rather than generating alarms.

The default value of TALARM is 1, indicating that the robotic arm can only operate according to the requirements of the TURN option. If it does not meet the requirements, the system will sound an alarm.

TALARM=0, The option to turn off TURN requires the system to automatically select the appropriate value of the TURN option to ensure that the movement of J LR [X] does not generate an alarm.



Example:

TURN = 0 'Method 1, Global Parameter Setting

TALARM = 0 'Method 1, Global Parameter Setting

J LR[x] TURN=2 TALARM=1 'Method 2: Use local parameter settings



Tips:

The TURN and TALARM options only take effect when J<space point>, and

have no effect when J<joint point> or L | C<space | joint>. That is, when J JR [x] TURN= 1 TALARM= 1, the system will not alarm, but the parameters have no actual meaning.

When the input of TURN and TALARM parameter values is illegal (such as

parameter values exceeding the range), the value change operation will not take effect.

11.10.6 WRISTJNT Singular parameters passing through the wrist

The WRISTJNT instruction can achieve passing through wrist singularities. When passing through the wrist singularity, the 4th and 6th axes do not flip, and the TCP end of the robotic arm maintains a straight trajectory, but the posture position will change. If this instruction is not used, the speed of the robotic arm will gradually decrease and then stop at the boundary of the singular interval when it approaches the wrist singularity.

Grammar format:

WRISTJNT=<num expression>



Example:

```
WRISTJNT= 1
L JR[1]    'JR[1][4]=-20
L JR[2]    'JR[2][4]=20
```

11.10.7 SEC Specify exercise time parameters

The SEC directive specifies the time for a robotic arm to perform a row of joint movements. The SEC instruction only applies to joint motion J instruction and is invalid for spatial motion L | A | C | JUMP.

The numerical unit of SEC instructions is milliseconds, with a range of 0 to [20, 32000]. If the SEC value exceeds the range during program execution, an alarm will be triggered stating that the value is out of range.

When SEC=0, it indicates that the joint motion instructions after the instruction line are not affected by the effect of SEC instructions.

When using SEC commands, the local velocity parameters of joint motion commands are invalid.

The time specified by the SEC directive is the time it takes to move at a speed of 100% magnification. If the magnification is adjusted, the actual time for the robot to complete the movement will change according to the adjusted magnification.

For example, if the specified beat is 1000ms, then when the magnification is 100%, the actual time for the robot to complete the movement is 1000ms; When the magnification is 50%, the actual time for the robot to complete the movement is 2000ms.

If the robotic arm can move to the target POINT within the specified SEC time, it will move within the specified time. Otherwise, an overspeed alarm will be reported.

A continuous motion command line with the same specified time is called a segment. In terms of algorithm, the system will perform global planning within each segment and use smooth transition effects between segments to ensure overall continuity of motion. The smooth transition effect is not affected by the CNT attribute values.



Example:

```
VORD= 100
J P[1]      'P [1] is not affected by the effectiveness of SEC directives
R[1]=1000
SEC=R[1]
L P[2] VEL=2000 'P [2] is not a joint motion command, SEC command has no
effect, and VEL=2000 is used during motion
SEC=50
J P[3] CNT=0    'During actual exercise, the specified rhythm is performed
with SEC=50, and CNT=0 setting is invalid
SEC= 10         'SEC instruction value exceeds the range, therefore an alarm
```

is triggered

```
J P[4] L_ACC=50
L P[5] L_VORT=100    'P[5] is not a joint motion command, therefore SEC
command is invalid
SEC=20
VORD=50
J P[6]                'If the magnification is 50% and SEC is 20, then the actual
movement will take 40ms to perform the specified rhythm movement
SEC=0
J P[7]
```

11.11 Sports Additional Attributes

The additional attribute of motion refers to the operation of controlling the robotic arm to change the target point, change the motion state, or output signals during the execution of motion. The instructions and order of effectiveness when using additional attributes for each sport simultaneously are shown in the following table:

Table 9-1 Detailed Description of Attributes

Name	The elements acted upon	Effective order	Notes
INC	Target point	1 (Highest)	(1) Cannot be used simultaneously with OFFSET/TOL_OFFSET (2) Cannot attach to C/A/JUMP motion command
OFFSET	Target point	1	(1) Cannot be used simultaneously with INC
TOOL_OFFSET	Target point/Coordinate System	1	(2) If these two instructions are used simultaneously, when calculating the motion POINT, first calculate the OFFSET offset value on the target point, and then calculate the TOOL-OFFSET tool compensation value
SKIP	Motion state	2	Motion interruption signal, judged when motion begins to execute
BIND_TR	Motion logic I/O signal	3	Output signal upon reaching the designated location
WAIT	Motion logic I/O signal	3	Wait for the signal to meet the requirements before moving to the

			target point when reaching the designated position
PASS	Instruction execution effect	4 (Minimum)	Independent use

11.11.1 INC Incremental motion attribute

Incremental motion is the movement of a robotic arm based on its current position, which means taking the value of the target POINT as the incremental value of the current robotic arm's movement. J P [x], INC, where the incremental value of P [x] can be a joint coordinate value or a Cartesian coordinate value.



Example:

J JR[0] 'Move to JR [0]

J JR[1], INC 'Perform incremental motion based on the previous point, moving to JR [0]+JR [1]

J JR[2], INC 'Perform incremental motion based on the previous point, moving to (JR [0]+JR [1]+JR [2])

Execution Logic Manual:

JR[0]={0,-90,180,0,90,0}, JR[1]={10,10,10,10,10,10}, JR[2]={-5,-5,-5,-5,-5,-5},

The execution process involves joint movement to point JR [0]. When executing J JR [1], INC will perform relative motion based on point JR [0], moving to point (JR [0]+JR [1]). The third row will move to point (JR [0]+JR [1]+JR [2]).

The POINT coordinate values for the three movements are:

JR[0]={0,-90,180,0,90,0},

JR[0]+JR[1]={0,-90,180,0,90,0}+{10,10,10,10,10,10}={10,-80,190,10,80,10},

JR[0]+JR[1]+JR[2]={0,-90,180,0,90,0}+{10,10,10,10,10,10}+

{-5,-5,-5,-5,-5,-5}={5,-85,185,5,75,5}



Tips:

When the incremental value of joint motion is the spatial POINT, the spatial POINT will be solved in reverse first, and then the joint solutions will be selected and overlaid before moving to the target point.

For example:

J JR[1]

J LR[1], INC 'LR [1] will first invert to joint values and then superimpose them

11.11.2 OFFSET Position compensation attribute

The position compensation instruction performs offset compensation on the position, depending on the type of compensation value. There are two forms of compensation. Joint compensation and spatial POINT

compensation. The compensation effect varies depending on the POINT type!

Form 1 is the case where the compensation value is joint, i.e. OFFSET JR [x]. The compensation effect is to add the compensated joint values to the joint values set at the target POINT after moving to the target point.

Form 2 is the case where the compensation value is spatial POINT, i.e. OFFSET LR [x]. The compensation effect is to offset the direction of the workpiece coordinate system when moving to the target point and placing it along the target POINT, which is equivalent to multiplying the current point matrix by the compensation value.



Point type Instructions for compensation values.

(1) The compensation value is the joint value, which is the effect instruction of OFFSET JR [x]

J JR[1], OFFSET JR[0] ‘Final position:JR[1]+JR[0]
J LR[0], OFFSET JR[0] ‘LR [0] is first solved as joint value P1, and the final position is P1+JR [0]

(2) The compensation value is a spatial value, which is the effect instruction of OFFSET LR [x]

L LR[1], OFFSET LR[0] ‘Offset compensation is performed along the current workpiece coordinate system, resulting in the final position LR [0] MATMUL LR[1].
L JR[0], OFFSET LR[0] ‘JR [0] is first solved as spatial point P1, and offset compensation is performed along the current workpiece coordinate system, resulting in the final position LR [0] MATMUL P1

There are two methods of compensation, global form and local form

(1) Global condition compensation instruction program Example:

OFFSET_CONDITION=JR[2] ‘Define global position compensation amount

J JR[0] ‘Correctly move to the JR [0] coordinate position
J JR[1],OFFSET ‘JR [1]+JR [2] move to a recalculated position

As shown in the above program, the program assigns JR [2] to the global position offset and sequentially moves it to JR [0] point (without OFFSET, no offset required). When executed to the last line, it will move to the offset compensated position, that is, JR [3]=JR [1]+JR [2] (with OFFSET, and the value of OFFSET-CODITION is JR [2]).

(2) Direct position compensation instruction program Example:

J JR[0] ‘Correctly move to the JR [0] coordinate position
J JR[1],OFFSET JR[2] ‘JR [1]+JR [2] move to a recalculated position

As shown in the above program, move sequentially to point JR [0] (without OFFSET, no offset required), and when executed to the last line, it will move to the offset compensation position, that is, JR [3]=JR [1]+JR [2] (with OFFSET JR [2], the direct offset is JR [2]).



Tips:

!! It should be noted whether the compensation value is in the form of joint points or spatial points, and the compensation effect varies depending on the POINT type!!

- (1) If method 1 is used and the OFFSETVNet instruction is not defined before the motion command J JR [1] OFFSET, an alarm will be triggered at runtime stating 'OFFSET value not set'.
- (2) If the JR [2] POINT information is modified midway during the execution of J JR [1] OFFSET or J JR [1] OFFSET JR [2], the offset will also change synchronously.
- (3) The coordinate representation of compensating POINT can be inconsistent with the target point (such as using joint coordinate system for the target point and Cartesian coordinate system for compensating POINT).
- (4) When compensating for the arc command, only the endpoint is compensated, and the transition point POINT of the arc is not compensated.

11.11.3 TOOL_OFFSET Tool TCP Direction Compensation Attribute

In practical applications, position compensation needs to be performed along the XYZ direction of the currently used tool coordinate system. When compensating along the tool coordinate system, only spatial POINT compensation is supported. There are also two compensation methods: global compensation and local compensation.

When compensating for spatial position, it is TOOL_OFFSET LR [x]. The compensation effect is the displacement along the coordinate axis of the tool coordinate system after moving to the target point. It is equivalent to performing a matrix right multiplication operation on the target POINT. Identify new target points.



The compensation value is a spatial value, which is the effect instruction of TOOL_OFFSET LR [x]

L LR[1], TOOL_OFFSET LR[0] 'Final position LR [1] MATMUL LR [0].

L JR[0], TOOL_OFFSET LR[0] 'JR [0] is first solved as spatial point P1, and offset compensation is performed along the current tool coordinate system, resulting in the final position P1 MATMUL LR [0].



There are two methods of compensation, global form and local form. There are two methods: one is to set the global tool compensation amount CONDITION and use TOOL-OFFSET to implement it, and the other is to directly specify the tool compensation amount after TOOL-OFFSET.

(1) Set global tool compensation amount Example:

TOOL_OFFSET_CONDITION=LR[2] 'Define global tool compensation amount

J JR[0] 'Directly move to JR [0]
J LR[1],TOOL_OFFSET 'Move to the (LR [1] MATMUL LR [2])
position.

(2) Directly specify the tool offset compensation amount Example:

J JR[0] 'Correctly move to the JR [0] coordinate
position
J LR[1],TOOL_OFFSET LR[2] 'Move to the (LR [1] MATMUL LR [2])
position.



Tips:

!!! Attention: Is the POINT of the compensation value a joint point or a spatial point? The compensation effect for joint points and spatial points is different!!!

- (1) The tool compensation quantity TOOL_SOFFSETVNet can only use the spatial POINT LR [x] variable.
- (2) If the TOOL-OFFSET-CODITION instruction is not defined before, the runtime alarm will sound "TOOL-OFFSET value not set".
- (3) When executing J LR [1] TOOL-OFFSET or J JR [1] TOOL-OFFSET LR [2], if the TOOL-OFFSET offset LR [2] POINT information is modified midway before executing the instruction, the offset will also change synchronously.
- (4) The coordinate representation of compensation POINT can be inconsistent with the target point (such as using joint coordinate system for the target point, and Cartesian coordinate system for compensation POINT).
- (5) When compensating for the arc command, only the endpoint is compensated, and the transition point POINT of the arc is not compensated.

11.11.4 SKIP Interrupt function attribute

The SKIP attribute can implement the function of interrupting the execution of the current motion instruction. During the execution of motion commands, if the configured triggering conditions are met, the robotic arm will immediately end the current motion and start executing the next line of program instructions. If the triggering condition is not met before the robot reaches the target point, the robot will complete the current motion line and execute the next line of instructions in the program after reaching the target point.



Example:

J P[1]
J P[2] SKIP DO[1]=ON 'When the DO [1]=ON condition is true, the P2 point
is skipped during runtime, and when it is false, the point is completed normally
J P[3]

11.11.5 PASS Skip waiting for motion execution instruction attribute

The PASS instruction means to skip and not wait for the current line of motion execution to complete. When the motion instruction starts executing, the non motion statements following the program also start executing at the same time, until the next motion instruction is encountered, blocking and waiting. Reasonable use of PASS attribute can achieve simultaneous execution of motion and logic, and multiple segments of motion can be smoothly spliced to improve production rhythm.



Example:

```
J P[0]
J P[1] CNT=50, PASS 'When starting the motion, DO [1]=ON and
subsequent non motion statements begin to execute, with a smooth transition between
P [1] and P [2]. If an alarm is triggered when starting execution, such as exceeding
the limit, the subsequent logical instructions will not be executed. Only after the
motion is
successfully started and executed, there will be a PASS operation for that segment of
motion
DO[1]=ON
R[1]=1
IF R[1]=0 THEN 'If the conditions are not met, do not enter the branch
and continue to execute downwards
LR[1][1]=1
END IF
IF R[1]=1 THEN 'Conditions are met to enter
LR[1][1] = 100
L P[2], PASS 'Block, wait until the motion of J P [1] is completed and the
motion of L P [2] begins to execute before the following logical statements are
executed
ENDIF
```



Tips:

- (1) The PASS attribute only supports J | L | A | C instructions and does not support JUMP instructions.
- (2) When a motion command has a PASS attribute, the execution pointer will block and wait for the current motion to complete before executing the next motion when encountering motion command J | L | C | A | JUMP.
- (3) The DELAY instruction is equivalent to a placeholder for empty motion, and the execution pointer will block on the DELAY instruction line.

11.11.6 SET_TR Signal trigger function attribute

The process of a robotic arm completing a motion trajectory is: stationary (starting point) - acceleration -

uniform speed - deceleration - stop (target point), with uneven speed during acceleration and deceleration. If the robotic arm outputs a signal or closes the signal after reaching the starting or target point, it will introduce trajectory segments with uneven speed, which will cause uneven coating in adhesive or welding applications and reduce product processing quality. Using SET_TR and BIND_TR signal triggers, signals can be turned on or off at specified trajectory segment positions.

Grammar format:

SET_TR[num], DS|DE|DC=<num>, <signal expression>

<Motion Type><Target Point>, BIND_TR[num1,num2...]

Parameter Instructions:

Starting point: Under normal circumstances, it is the target point of the previous motion command.

When the motion terminates, the calculation needs to be restarted from the current position of the robot as the starting point.

DS: The abbreviation for DISTANCE START, which refers to the distance from the starting point of the robot's motion.

DE: The abbreviation for DISTANCE END, which refers to the distance from the moving target point.

DC: The abbreviation for DISTANCE Complete, which refers to the percentage of completion of the entire distance.

Table 9-2 Detailed Parameter Manual

Trigger condition	DS=xx	DE=xx	DC=xx
Meaning	Robot TCP outputs signals when the distance from the starting point of robot operation exceeds xx mm	When the distance between the robot TCP and the target point of robot operation is within xx mm, output signal	Robot TCP has run the motion command line. When Robot TCP needs to move xx% of the entire distance, output a signal
Trigger condition	The robot leaves the starting point and the distance (instruction) between the robot and the starting point is greater than xx mm	The robot moves towards the target point, and the distance (instruction) between the robot and the target point is less than xx mm	The robot is moving towards the target point, and the percentage of the robot's TCP walking distance to the total length of the path is greater than xx
Output signal	When the robot meets the triggering conditions, signal output or shutdown is performed		

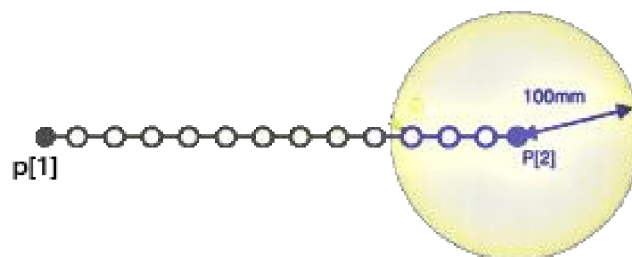


Figure 9-1 Schematic of Triggering POINT



Example:

```
SET_TR[0], DS=50, DO[4]=ON
SET_TR[1], DE= 100, DO[4]=OFF  'Set trigger number 1, specify the distance
of 100mm from the end point target, and output DO [4]=OFF

J P[0]
L P[1], BIND_TR[0]
L P[2], BIND_TR[1]  'P2POINT sets trigger 0 or 1, can set single or multiple
triggers, separated by commas
```

Execution Logic Manual:

In the example program, the first two lines set triggers 0 and 1, which means that at the end point 100mm, DO [5] outputs ON (note: when there is the same trigger number, execute according to the latest trigger number condition). Trigger 0 is bound at point P1, and trigger 1 is bound at point P2; The robotic arm moves linearly to point P [1], and then moves linearly from point P [1] to point P [2]. The system will periodically detect the position of the robot's TCP distance P1 and P2 points. When the distance P [1] is greater than 50mm, DO [4]=ON signal output; When the distance P [2] is less than 100mm, execute DO [4]=OFF.



Tips:

- (1) When the robot starts moving and meets the triggering conditions, it outputs the corresponding signal when it starts moving.
- (2) When CNT smoothing exists, the smooth transition does not pass through the target point, and there may be situations where the distance from the target point does not meet the triggering conditions, resulting in no signal output.
- (3) When the program pauses midway and resumes running, it will re evaluate the triggering conditions based on the current robot position as the starting point.
- (4) The specified distance signal output J joint command is invalid.
- (5) In actual operation, the specified distance signal output is related to the operating speed of the robotic arm, and there may be slight errors in the signal output point. This can be adjusted by reducing the speed of the output point or adjusting the specified distance value.

11.11.7 WAIT Determine the trigger signal attribute

During the operation of the robotic arm from the current point to the target point, it determines whether the triggering conditions are met at the set distance. If the conditions are met, the robotic arm continues to move forward to the target point. If not met, the robotic arm decelerates and stops, waiting for the signal conditions to be met before starting to run to the target point.

Grammar format:

<Motion Type><Target Point> WAIT <signal expression> <DS|DE|DC=num>

Table 9-3 Detailed Parameter Manual

Trigger condition	DS=xx	DE=xx	DC=xx
Meaning	Robot TCP outputs signals when the distance from the starting point of robot operation exceeds xx mm	When the distance between the robot TCP and the target point of robot operation is within xx mm, output signal	Robot TCP has run the motion command line. When Robot TCP needs to move xx% of the entire distance, output a signal
Trigger condition	The robot leaves the starting point and the distance (instruction) between the robot and the starting point is greater than xx mm	The robot moves towards the target point, and the distance (instruction) between the robot and the target point is less than xx mm	The robot is moving towards the target point, and the percentage of the robot's TCP walking distance to the total length of the path is greater than xx
Output signal	When the robot meets the triggering conditions, signal output or shutdown is performed		



Example:

```
J P[0]
L P[1] WAIT DI[1]=ON DE= 100    'Determine whether the signal of DI [1] is
ON at a distance of 100mm from the endpoint of P [1]. If it is ON, move to point P
[1], otherwise stop waiting.
L P[2]
```



Tips:

When the conditions are not met, the robotic arm decelerates and stops, then waits. At this time, the program is in a running state, and the robotic arm will automatically execute downwards after the signal is met.

11.12 Load setting

The PAYLOAD command is used to set the load number of the current end effector of the robotic arm. Load information includes information such as mass, center of mass, inertia, etc. The system has a total of 16 sets of load numbers, and users can hang different loads in practical applications, estimate and save them separately in each set of numbers.

Grammar format:

PAYLOAD=<-1~15>



Example:

```
PAYLOAD= 1
J P[0]
L P[2]
```



Tips:

- (1) When PAYLOAD=-1, it indicates no load.
- (2) Only robotic arms that support dynamic models can use the load estimation function.

11.13 Collision detection

11.13.1 Collision detection switch

The collision detection switch function is used to turn on and off the collision detection function. If the robotic arm supports the dynamic model and completes parameter identification, the collision detection function is enabled by default. If the user wants to disable collision detection for a certain trajectory during program execution, they can use this command to turn it off.

Grammar format:

COL_DETECT =<ON|OFF>



Example:

```
COL_DETECT=ON      'Collision detection is effective when moving the
trajectories of the two points below
J P[0]
J P[1]
COL_DETECT=OFF      'Collision detection is invalid when moving the
trajectory of the two points below
J P[2]
J P[3]
```

11.13.2 Collision detection sensitivity

The sensitivity of the collision detection function can be adjusted, and the higher the degree value, the higher the detection sensitivity.

Grammar format:

COL_GUARD_ADJUST = <1~100>


Tips:

When the collision detection sensitivity is set to 1, it indicates that it is very insensitive, and the system defaults to turning off the collision detection function.

11.14 IO Instruction

IO instructions are divided into IO status reading and output control, pulse signal output, etc. The system supports both digital and analog signals and has mapped them to predefined variables.

11.14.1 IO Variable instruction

Grammar format:

DO[num]=ON|OFF

AO[num]=<num expression>


Example:

```
J P[1]
DO[1]= ON           'Output [1] set to ON
DO[1]= OFF          'Output [1] set to OFF
DO[1]= DO[2]        'Output [2] is assigned as output [1]
DO[1]= DI[2]        'Input [2] is assigned to output [1]
AO[1]=1.5
IF DI[2]=ON THEN
  J P[2]
END IF
```

11.14.2 VDI Virtual input instruction

VDI is a virtual IO input, and when the system's IO switches to a virtual state, the value of VDI represents the status of that IO. Execute VDI [X]=ON | OFF, set the corresponding virtual DI signal to ON or OFF.

Grammar format:

VDI[num]=ON|OFF


Example:

```
J P[1]
VDI[1] = OFF        'Set [1] to OFF
VDI[1] = ON          'Set [1] to ON
VDI[1] = DO[2]       'Output [2] is assigned as virtual input [1]
VDI[1] = DI[2]       'Assign input [2] to virtual input [1]
```

11.14.3 PULSE Output pulse command

The PULSE instruction is used to control a certain DO [x] to output a pulse signal of a specified duration. After the PULSE instruction is executed, the program pointer starts to execute downwards without blocking and waiting for the pulse output to complete. The parameter unit for TIME is milliseconds.

Grammar format:

PULSE DO[unsigned int num] TIME=<unsigned int num>



Example 1:

```
J P[0] VEL=50
PULSE DO[4] TIME= 10000
J P[1] VEL=50
```

Execution Logic Manual:

The joint motion of the robotic arm reaches point P [0], at which point DO [4] is set to ON. The program does not wait and executes the next line of motion instructions normally. The joint motion reaches point P [1], and the program execution is completed. However, starting from the output of DO [4], after 10 seconds, the output of DO [4] is set to OFF.

Example 2:

```
DO[4]=ON
WAIT TIME= 1000
PULSE DO[4] TIME= 10000
```

Execution Logic Manual:

If there are both DO and PULSE instructions in the same time period, they should be unified and processed according to the principle of change first. DO [4] is set to ON, and then the program delays for 1 second. In the PULSE instruction, DO [4] is first set to ON, followed by OFF after 10 seconds, and the output ON duration is a total of 11 seconds.

Example 3:

```
PULSE DO[4] TIME= 1500
PULSE DO[4] TIME= 1000
```

Execution Logic Manual:

When two pulse instructions operate on the same IO simultaneously within the same time period, they are stacked in the time dimension and output as pulses with longer duration.

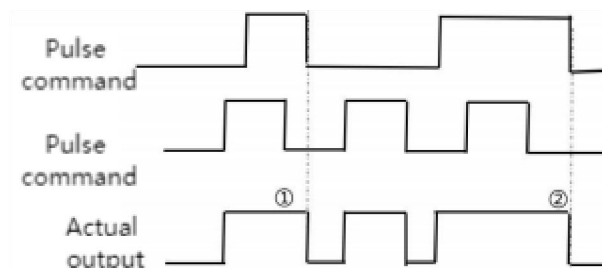


图 12-1 Pulse time superposition



Tips:

If the program is uninstalled before the output pulse ends, it will automatically reset to OFF position.

11.15 WAIT Waiting for instructions

The WAIT instruction is used to wait and determine if a condition is met. If the condition is not met, the program will block on this line until the condition is met before continuing execution. The WAIT instruction can be set to exceed the judgment.

Grammar format:

WAIT <bool expression> [TIMEOUT <int timeout> LBL <int nnum>]



Example 1:

```
WAIT DI[1]=ON      'When DI [1] is not equal to ON, the program keeps
blocking
J P[1] VEL=50
J P[2] VEL=50
```

Example 2:

```
J JR[1] VEL=50
WAIT DO[4]=ON TIMEOUT 500 LBL 2      'WAIT timeout instruction
J P[1] VEL=50
LBL [2]
J P[2] VEL=50
```

Execution Logic Manual:

The robotic arm joint moves to JR [1], and then waits for the DO [4]=ON condition to be met. If the DO [4]=ON condition is met within 500ms, the instructions will be executed in sequence, and the joints will move to P [1] and P [2] in turn. If the DO [4]=ON condition is not met within 500ms, the timeout will occur, and the pointer will jump to the LBL [2] line and continue to execute from that line, directly moving the joints to point P [2] without executing J P [1].

11.16 WAIT TIME Delay instruction

The function of the WAIT TIME instruction is to delay the execution of a program (task) in milliseconds. In the background RUN program or the WHILE statement block of an infinite loop, the WAIT TIME

instruction should be used for delay, otherwise there may be a problem of high CPU usage.



Example:

```
J P[1]
WAIT TIME= 1000    'After moving to P1 and waiting for 1 second for sleep,
VDI [1] is set to ON
VDI[1]=ON
J P[2]
```

11.17 Coordinate system instruction

The coordinate system instruction is used to call the coordinate system in the system, mainly to change the values of the coordinate system or make a coordinate transformation relationship effective. The coordinate system conversion diagram of the robotic arm is as follows:

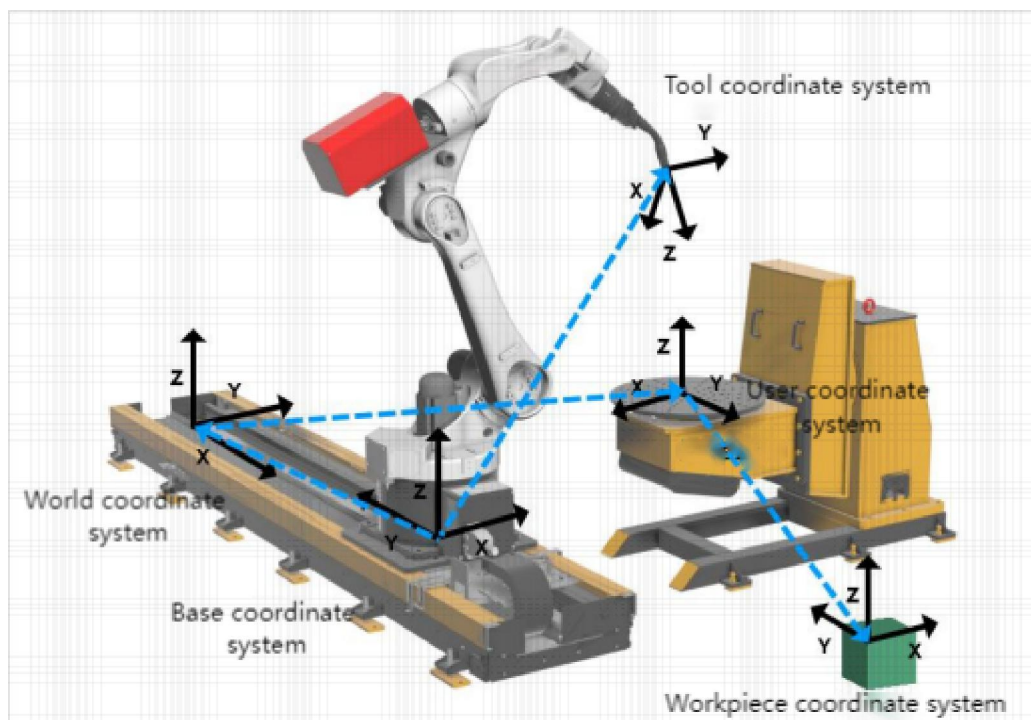


Figure 15-1 Coordinate System Conversion Relationship

11.17.1 UTOOL_NUM Tool coordinate system call instruction

This instruction is used to call the TCP tool coordinate system that must be used in the program. When executing a motion to a spatial POINT (where the spatial POINT information includes the value of UT), the system will determine whether the current spatial POINT is consistent with the tool coordinate system being used. If they are different, an alarm will be triggered stating that the POINT coordinate system is inconsistent. Therefore, before moving to a spatial POINT, the corresponding tool coordinate system must be called first. The value of UTOOL_UM is -1~15.

Grammar format:

UTOOL_NUM=<int expression>



Example:

UTOOL_NUM = 0 'Use the value of tool coordinate system 0

L P[1] VEL=50

L P[2] VEL=50

UTOOL_NUM = -1 'Switch the tool coordinate system to the default

value of -1

J P[3] VEL=50

J P[4] VEL=50



Switching the tool coordinate system to the default value of -1 prompts:

When recording POINT during teaching, the current tool coordinate system used will be saved in POINT, so the corresponding tool coordinate system must be called in the PRG program.

11.17.2 UFRAME_NUM Instruction for calling the workpiece coordinate system

This command is used to call the workpiece coordinate system to be used in the program. When executing a motion to a spatial POINT (where the spatial POINT information includes the value of UT), the system will determine whether the current spatial POINT is consistent with the currently used workpiece coordinate system. If they are different, an alarm will be triggered stating that the POINT coordinate system is inconsistent. Therefore, before moving to a spatial POINT, the corresponding workpiece coordinate system must be called first. The value of UFRAME_NUM is -1~15.

Grammar format:

UFRAME_NUM=<int expression>



Example:

UFRAME_NUM = 0 'Use the value of 0 in the workpiece coordinate system

L P[1] VEL=50

L P[2] VEL=50

UFRAME_NUM = -1 'Switch the workpiece coordinate system to the

default value of -1

J P[3] VEL=50

J P[4] VEL=50



Tips:

When recording POINT in teaching, the current workpiece coordinate system used

will be saved in POINT, so the corresponding workpiece coordinate system must be called in PRG program.

11.17.3 WORLDFRAME Command to call the world coordinate system

This instruction is used to activate or deactivate the world coordinate system. The world coordinate system coincides with the base coordinate without calibration, that is, both are located at the base position of the robotic arm. In the application of ground rail+robotic arm collaboration, it is often necessary to use the world coordinate system. For detailed information on the collaborative motion functions of the robotic arm and ground rail, please refer to documents such as the "Instructions for the Use of Robotic Arm and Ground Rail Collaborative Functions".

Grammar format:

WORLDFRAME=<ON|OFF>



Example:

```
WORLDFRAME=ON      'Enable world coordinate system
UFRAME_NUM=1       'Call Workpiece 1
UTOOL_NUM=1        'Call Tool 1
SYNCEXTAX=ON       'Ground rail collaborative motion activated
L P[0]
L P[1] VEL=100
WORLDFRAME=OFF     'Close the world coordinate system
J P[2] VEL=100     'Time synchronized motion, POINT is the call to UF1 in the
base coordinate system
```

11.17.4 SET_TOOL Tool coordinate system value modification instruction

The SET_TOOL instruction is used to modify the values of the tool coordinate system, and users can use the PRG program to modify the coordinate system of the current tool coordinate system.

Grammar format:

SET_TOOL <0~15> LR[unsigned int num]



Example:

```
UTOOL_NUM = 1      'Call tool coordinate system 1
SET_TOOL 1 LR[3]   'Modify the value of tool coordinate system 1 and assign
the data of LR [3] to tool coordinate system 1
L P[1]
```

11.17.5 SET_FRAME Workpiece coordinate system value modification instruction

The SET_FRAME instruction is used to modify the values of the workpiece coordinate system, and users can use the PRG program to modify the coordinate system of the current workpiece coordinate system.

Grammar format:

SET_FRAME<0~15> LR[unsigned int num]



Example:

```
UFRAME_NUM = 1      'Call workpiece coordinate system 1
SET_FRAME 1 LR[5]    'Modify the value of workpiece coordinate system 1 and
                    assign the data of LR [5] to workpiece coordinate system 1
L P[2]
```

11.17.6 CALIUFRAME Calibration instruction for workpiece coordinate system

The CALIUFRAME instruction is used to generate a workpiece coordinate system instruction through three points, and users can calibrate and assign values to the current workpiece coordinate system using the PRG program.

Grammar format:

<location var_uf>=CALIUFRAME(< location origin> ,<location x_pos> , <location y_pos >)



Example:

```
LR[1] = CALIUFRAME(LR[2],LR[3],LR[4])    '3-point method for calibrating
coordinate values
SET_FRAME 1 LR[1]                        'Set the calibrated coordinate values to
workpiece 1
L P[2]
```



Tips:

The workpiece coordinate system calibrated by the 3-point method is based on the OX axis. The UT and UF values of origin, x_pos, and y_pos should be consistent.

11.17.7 WCS_TEST Coordinate system detection instruction

When the PRG program is executed, if the currently used tool workpiece coordinate system and the tool coordinate system number UT recorded in the spatial POINT are different from the workpiece coordinate system number UF, the system will alarm "POINT coordinate system inconsistency". The function of the WCD_TEST instruction is to check whether the currently used coordinate system is consistent with the

coordinate system in the spatial POINT. WCD_TEST=0 means to turn off coordinate system detection and allow the currently used coordinate system to be different from the coordinate system in the spatial POINT. Grammar format:

WCS_TEST=0|1



Example:

```
UTOOL_NUM= 1
L P[1]      'P[1].UT= 1, The current tool number used is consistent with the tool
number in the spatial POINT, and the system does not alarm

L P[2]
UTOOL_NUM=2
L P[3]      'P[3].UT= 1 ,The current tool number used does not match the tool
number in the spatial POINT, and the system will sound an alarm

WCS_TEST=0   'Turn off detection
UTOOL_NUM=2
L P[3]      'P[3]. UT= 1 , The current tool number used does not match the tool
number in the spatial POINT, but coordinate detection is turned off and the system
does not alarm. It will move to the point under UT [1]
```

11.18 Matrix arithmetic commands

11.18.1 MATMUL matrix multiplication instructions

This instruction multiplies two chi-square matrices represented by POINT data and returns them. Note that the left and right operands of the two sides of the instruction cannot be exchanged (matrix multiplication does not satisfy the exchange law), i.e., whether a is multiplied right by b, or b is multiplied right by a, is determined by the user's input.

Grammar format:

<target var> = <location var1> MATMUL <location var2>



Example:

```
LR[1] = LR[1] MATMUL LR[2]      ' LR[1] right multiply by LR[2] and save in
LR[1]
```



Limit Instructions:

The UT,UF,CFG of the target point <target var> all use the value of <location var1>, and only the <location var2> POINT value is used in the multiplication, ignoring the coordinate system, morphometric component in the POINT data. The above example assumes LR[1]. UT= 1 , LR[1]. UF= 1, LR[2].UT=2, LR[2].UT=2 .

When multiplying, only the 'POINT value' of LR[2] is taken, without considering the coordinate system of that POINT. The LR[1] point after multiplication means it is

the point under UF1,UT1.

11.18.2 MATINV matrix inverse command

This instruction will invert the chi-square matrix represented by the POINT data and return it.

Grammar format:

<target var> = MATINV <location var1>



Example:

```
LR[1] = MATINV LR[2]
```



Restrictions on Instructions:

The UT,UF,CFG of the target point <target var> take the value of <location var1>.

11.19 Coordinate system conversion case



Case procedure 1: Transformation of space points under the workpiece coordinate system 1 to points under the base coordinate system.

```
UFRAME_NUM=1      ' Switch to use artefact 1. This instruction is not
                    ' required, and need not be called if artefact 1 has already been called in a previous line
                    ' of the program.

WAIT TIME=1        ' Waiting for motion to reach position

LR[1] = LPOS        ' Get the current actual position of the robot arm

LR[10]=UF[1]       ' LR[10] holds the values of the workpiece
                    ' coordinate system 1.

LR[2] = LR[10] MATMUL LR[1]  ' Transforms to a point in base coordinates .
```

Case Example Procedure 2: Transform the spatial points of the tool coordinate system 1 with respect to the base coordinates to the spatial points of the end flange with respect to the base coordinate system.

```
UTOOL_NUM = 1      ' Use of tool 1

WAIT TIME=1        ' Waiting for motion to reach position
```

```
LR[1] = LPOS LR[11]= UT[1]    ' Get points under tool 1
LR[12] = MATINV LR[11]    'Getting the value of tool 1

LR[2] = LR[1] MATMUL LR[12] ' Find the inverse matrix of tool 1 ' Transforms
back to end flange coordinates. lr[2] is the position of the end flange.
```

11.20 Co-motion commands

11.20.1 COORD_NUM synergy command

The COORD_NUM instruction is used to call the cooperative group number in the co-motion of the indexing machine. When you use the co ordinate function, you have to complete the configuration of the indexer axes, co ordinate group calibration and co ordinate trajectory teaching before you can call the co ordinate group in the programme and start the co ordinate motion. The value range of synergy group number (synergy coordinate system) is -1~4. The use of synergy function can be found in the operation and programming manual of Huazhou 3-type robotic arm.

Grammar format:

COORD_NUM=<int num expression>



Example 1:

```
COORD_NUM = -1    'No co-ordinated motion, the arm moves in a straight line
to point P[1].
```

```
L P[1]
COORD_NUM = 3    'Calls synergy group number 3 and starts the
translocator synergy motion
J P[2] VEL=50    'Synergistic movement of robotic arms and
displacements
J P[3] VEL=50
```

Example2:

```
COORD_NUM= 1    'Global parameter, using co-group 1
J P[1]
J P[2]          'Robotic arm and the first positioner working in tandem
COORD_NUM=-1    'Close Collaboration
J P[3]          'Movement of the robot arm to the transition point
J P[4] VEL= 100 ACC= 100 COORD=2    'Local parameter, call synergy group 2
to synergise with another transformer
```



Hints:

- (1) The program execution must call the co-group number correctly, i.e. it must call the co-group number used in the trajectory demonstration.
- (2) In the application of one robot arm + two translators, the co-motion supports local attributes for programming convenience.

11.20.2 SYNCEXTAX Co-control Commands

The simultaneous action of the robotic arm and the ground rail supports both forms:

- (1) Time synchronisation, i.e. the ground track and the robot arm start and stop at the same time.
- (2) Synergistic movement of the ground track and external axes ensures the trajectory accuracy and running speed of the end TCP.

The time synchronisation function cannot guarantee the accuracy of the end of the robotic arm and is only applicable to production line handling. The synergistic motion can ensure that the trajectory of TCP is straight line, which is suitable for welding and other high-precision requirement scenarios. SYNCEXTAX instruction is used to realise the ground track synergistic function to be turned on and off, and before using the ground track synergistic function, it is necessary to complete the configuration of the axis number of the ground track axes and the calibration of the coordinate system of the ground track first. For more details, please refer to 'Instruction Book of Robotic Arm and Ground Rail Synergy Function'.

Grammar format:

SYNCEXTAX=<ON|OFF>



Example:

```

WORLDFRAME=ON      'Use of geo-orbital coordinate system (world
coordinate system)
UTOOL_NUM= 1
UFRAME_NUM= 1
SYNCEXTAX=ON        'Synergistic movement of the earth's orbit
L P[0]
L P[1]
SYNCEXTAX=OFF       'Conclusion of the ground-orbit concerted motion
J P[2]               'time synchronous motion
    
```

11.21 EX_UTOOL External TCP Command

The robot arm holds the processed object and moves around a centre point fixed on the ground (e.g., a spray gun or welding torch fixed on the ground). At this time, the centre point of the tool is fixed on the outside of the robot arm rather than at the end of the arm flange, and the robot arm is holding the workpiece in its hand for movement. When using the external TCP function, the calibration of the external TCP must

be completed first, and then the external TCP command can be called in the PRG programme to control the movement of the robot arm.

Grammar format:

EX_UTOOL=<ON|OFF>



Example:

```
EX_UTOOL =ON      'Enabling External TCP
UTOOL_NUM = 1     'Calling calibrated external tools1
L P[1]
L P[2]
EX_UTOOL =OFF     'Enabling External TCP
L P[3]            'Use flange tool 1 to move to point P[3].
```

11.22 Process control instruction

The programme control instructions are used for programme execution control, including LOAD load, START start, PAUSE pause, END stop, ABORT unload, as well as RUN instruction, MAINTASK, RUNTASK instruction. Users can control the execution status of foreground and background programs through program instructions. For example, communicating with the host computer in the background programme and controlling the execution of the foreground programme according to the logic requirements of the host computer.

11.22.1 MAINTASK foreground program name get command

The MAINTASK instruction is used to get the name of the foreground programme, the path and suffix are already included in the name of the programme obtained, such as '/MD/QULIAO/TEST1.PRG '. If there is no programme in the current task channel, then the obtained name is an empty string.

Grammar format:

<string var>=MAINTASK



Example:

```
SR[1]=MAINTASK    'Get the name of the frontend application
PAUSE SR[1]       'Suspension of front office procedures
```

11.22.2 RUNTASK background program name get command

The RUNTASK instruction is used to get the name of the background programme, the path and suffix are already included in the name of the obtained programme, such as '/MD/TT1.PRG '. If there is no programme in the current task channel, the obtained name is an empty string.

Grammar format:

<string_var>=RUNTASK



Example:

```
SR[1]=RUNTASK      'Get the name of the backend application
PAUSE SR[1]        'Suspend background processes
```

11.22.3 RUN background program enable command

The RUN instruction is used to turn on the background programme to run. When the user uses the multitasking function, the user can configure the background programme to run automatically on power-up on the oscillator, or use the RUN instruction to turn on the background programme in the foreground motion programme, with the background logic programme group mask as *.

Syntax Format.

RUN <task name>

1 The programme name is the absolute path name



Example:

```
J P[1]
RUN "TEST/A1.PRG"    'Enable background program execution
```

11.22.4 LOAD foreground program loading instruction

This directive can only load foreground programs and throws a warning message if the current main program has already been loaded.

Grammar format:

LOAD <task name>

The programme name is the absolute path name



Example (written in a background RUN programme):

```
LOAD "A1.PRG"        'Load A1.PRG in the root directory
LOAD "TEST/A1.PRG"    'Loading A1.PRG in the TEST folder
```

11.22.5 START programme start command

The START instruction is used to start the execution of a foreground or background program that is in the state of 'Ready' | 'Pause', the instruction can be executed in the main program or RUN program, if the specified program is not in the state of 'Ready' | 'Pause', i.e., it cannot start the execution of the specified program, the system throws an alarm 'Cannot start xxx program execution'. If the specified program is not in the 'Ready' or 'Pause' state, it cannot start the specified program, and the system throws an alarm 'Cannot start xxx program execution'.

Grammar format:

START <task name>

The programme name is the absolute path name



Example (written in a background RUN programme):

```
SR[1]=MAINTASK      'Get the name of the frontend application
START SR[1]          'Initiate foreground program execution
```

11.22.6 PAUSE programme pause command

The PAUSE instruction is used to pause the program itself, or to pause the execution of the program with the specified program name, and can be used in the main program or in a RUN program.

The syntax is as follows:

PAUSE [programme name]



Example:

```
J P[1]
J P[2]
PAUSE      'Program execution reaches this line and the program status
changes to paused
J P[3]
```

11.22.7 END Program Abort Instruction

The END instruction is used to abort the execution of its own programme or another programme, the programme pointer returns to the first line of the programme and the task is in the READY state. This instruction can be used in the main program or in a RUN program.

Grammar format.

END [programme name]



Example:

```
J P[1]
J P[2]
END      'Program execution reaches this line, the program ends, and the
program pointer returns to the first line.
J P[3]
```



Hints:

- (1) If the END instruction is executed inside the CALL subroutine, the execution of the subroutine is aborted, and the subroutine will be jumped back to the main program to continue to the next execution (if the CALL is multi-layer nested, it will return to the previous one).

-
- (2) If the program specified by the END instruction has been loaded (i.e. in READY state), the execution of the END instruction will have no effect on the program; if the program specified by the END instruction is in RUN, PAUSE, or ERROR state, it can be aborted to return the task to READY state.
-

11.22.8 ABORT programme uninstallation command

The ABORT instruction is used to force the termination of a task and unload a programme. All movements controlled by the program's task are stopped. the ABORT instruction can abort its own program or other programs.

Grammar format.

ABORT [programme name]



Example:

```
J P[1]
J P[2]
ABORT      'Program execution reaches this line, stops and uninstalls the
program
J P[3]
```



Tip:

If the ABORT instruction is executed inside a CALL subroutine, it is terminating the subroutine and the main program.

11.23 VORD trim command

VORD is used to adjust the speed multiplier of the robotic arm, when the programme executes the VORD instruction, no matter which mode the robotic arm is in, the speed multiplier is set to the specified value and maintain this speed multiplier until the outside world modifies the multiplier or until the next VORD instruction modifies it. the VORD value ranges from 1 to 100, and the user can also get the value of the multiplier of the current running speed through this instruction.

Grammar format:

VORD=<num expression>

<var num>=VORD



Example:

```
VORD= 10      'Setting the VORD value
J P[1]
J P[2]
```

```
R[1]=VORD    'Get VORD value
```

11.24 TIME Timing Command

The TIME instruction is used to obtain the value of the system's clock, which can be used to calculate the time in ms for a programme to run.

Grammar format:

```
<var num> = TIME
```



Example:

```
R[1]=TIME
J P[1]

J P[2]
R[1]=TIME-R[1]
```

Execution Logic Manual:

The first line assigns the system time to R[1], then the robotic arm moves to point P[1] and P[2], and in the last line, the difference between the current time and the previous saved time is calculated and saved to the R[1] register, and by checking the value of the R[1] register, the length of the running time from P[1] to P[2] can be known.

11.25 JPOS | LPOS Position Acquisition Command

The JPOS|LPOS instructions obtain the current position of the robot and assign it to the corresponding register variables JR[x]|LR[x]. JPOS obtains the current position of the robot in joint coordinates; LPOS obtains the position of the robot in Cartesian coordinates under the current coordinate system of the tool artefact.

Grammar format:

```
<joint var> = JPOS
```

```
<location var> = LPOS
```



Example:

```
JR[1] = JPOS    'Record the current joint coordinate position in JR[1] register
LR[1] = LPOS    'Record the current Cartesian coordinate position in LR[1]
register
```



Hints:

When using the JPOS|LPOS instruction, the JPOS|LPOS instruction can only appear on the right side of an assignment to an equation, and JPOS can only pass values to the

JR register and LPOS can only pass values to the LR register.

11.26 TRACE conveyor tracking command

The system supports the sensor belt tracking function. To use the sensor belt tracking function, you must configure the parameters of the 7th axis and set the 7th axis as the encoder axis, and then perform the conveyor belt coordinate system calibration, trigger operation settings, and follow area settings. For details, see the Instructions for using the conveyor belt tracking function.



Example:

```
LBL[1]
L P[0]
TRACE NUM=0,SKIP LBL[3]  'Set up conveyor tracking, use group 0, jump to
LBL[3] on exception
TPABS= 1                'Ways to set the contour offset point to an absolute position
GET TP                  'Blocking the acquisition of the triggered location TP point
J TP VEL=800            'Joint movement to TP point, synchronisation
process L P[1]           'reclaim point
L P[2]                  'Rising point after pickup is complete
TRACE OFF               'off-track
L P[3]                  'feed point
GOTO LBL[1]
LBL[3]
TRACE OFF
GOTO LBL[1]
```

11.27 WEAVE swing command

Oscillating motion is an operation in which the end of the robot arm TCP performs a positively rotating oscillating trajectory. When using the swing function, the robot arm establishes a swing coordinate system and performs swing motion in the specified swing plane. Swing commands are divided into swing configuration commands, swing on and swing off commands.

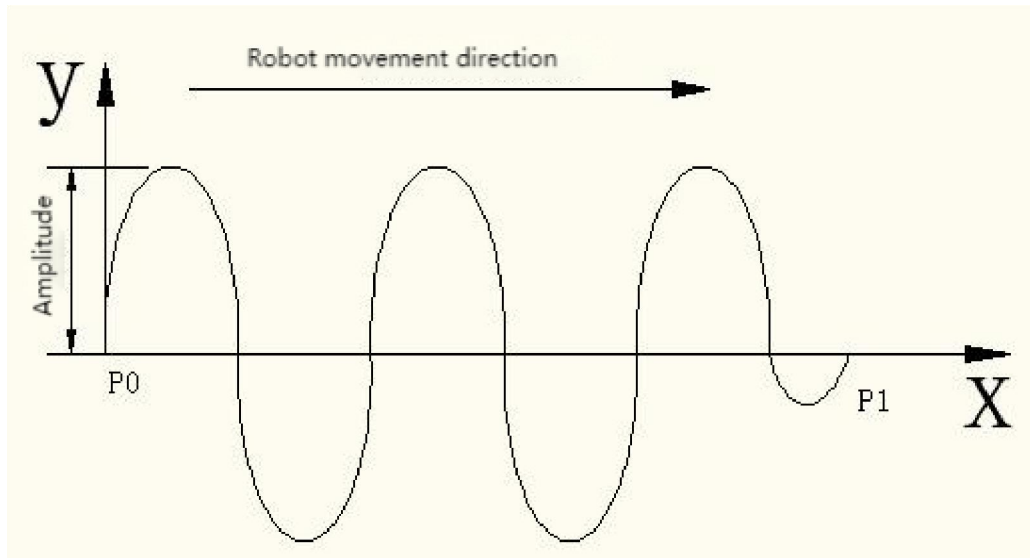


Figure 23-1 Schematic of WEAVE trajectory

Grammar format:

WEAVE NUM=<num> TYPE =<num> PLANE=<num> AMP= <num>

FRE=<num> WEAVE START NUM=<num>

WEAVE END

NUM: The serial number of the swing group is specified, and multiple groups of swing parameters can be configured, and then the group of swing parameters will be called when the programme starts the swing movement. Currently, the system supports 10 groups of swing parameters, i.e. the value of NUM is 0~9.

TYPE: Oscillating trajectory type, currently only positive spinning curves are supported, i.e. TYPE=0.

PLANE : The plane in which the oscillating trajectory of the robot arm is located. 1 denotes the XY plane; 2 denotes the YZ plane; 3 denotes the XZ plane.

AMP: The amplitude of the oscillation, i.e. the distance of the crest of the wave from the centre axis, in mm mm.

FRE : The frequency of the oscillation, i.e. how many positive spinning trajectories are travelled in 1 second, in Hz Hz.



Example:

```
WEAVE NUM=0 TYPE=0 PLANE= 1 AMP= 10 FRE=5      'Configure
parameters for swing group 0
J P[0]      'Movement to the starting point
WEAVE START NUM=0      'Oscillation start, start operation with set
oscillation parameter group 0
L P[1] VEL= 100      'oscillatory motion
J P[2]      'Normal execution of joint movements, no alarms
L P[3] VEL=200      'Oscillating motion, the value of the velocity vector in the
direction from P2 to P3 is 200mm/s
C P[4] P[5]      'oscillatory motion
WEAVE END      'end of swing
```

J P [6]

'Return to waiting point L P[1]

11.28 SLAVE follow command

The follow-me compensation function is the function of the robotic arm to adjust the position in real time during the movement according to the set compensation value. The follow-me compensation function is, in principle, a synthesis of the main movement and the follow-me movement, and the speed value of the movement command is agreed to be the speed value of the main movement, not the speed value of the final synthesised movement.

Grammar format:

SLAVE MODE=0|1 CYCLE=<num> LR=<num> MASK=255|256

MODE: Indicates the function is on or off, MODE value equals to 1 means turn on the follower, when the follower function is turned on, the current position will be used as the reference, MODE value equals to 0 means turn off the follower.

CYCLE: Indicates the period of compensation data sampling issued, the value means a multiple of the interpolation period. For example, CYCLE=2 means $4 \times 2 = 8\text{ms}$, and the current compensation data value is sent every 8ms.

LR: Indicates the LR register index value of the real-time compensation data value, the user calculates the compensation value in real time and writes it to this LR register, and the robotic arm carries out the follow function according to this value.

MASK : Indicates the mask value. Meaning of the follower function with different MASK values.
MASK=256 means position compensation in the direction of the swing coordinate system; MASK=255 means compensation in the direction of the tool coordinate system.



Example:

```

UTOOL_NUM = 1
L P[0] VEL= 150 ACC=200 DEC=200 CNT = 0 'Movement to the starting
point, using the current point as the reference point
SLAVE MODE= 1 CYCLE=4 LR=0 MASK=255 'switch on the
follow-through
LBL[1] 'trajectory
L P[1]
L P[2]
C P[3] P[4]
...
GOTO LBL[1]
SLAVE MODE=0 CYCLE=4 LR=0 MASK=255 'Turn off the follower

```

11.29 PALLET tray command

Tray command can be used for loading and unloading of 3C small materials, users only need to create a tray, and then enter the specified number of rows and columns to obtain the corresponding position, without the need to teach programming for each point.

Grammar format:

PALLET ID=<num> <P1> <P2> <P3> ROW=<nnum> COL=<num> LAY=<num> HEIGHT=<num>

ID: indicates the serial number of the tray, supports unsigned integer values ranging from 0 to 15.

P1: the origin of the tray; the line connecting P1 and P2 is the row direction of the tray; the line connecting P1 and P3 is the column direction of the tray.

ROW: indicates the number of rows, supports unsigned integer values ranging from 1 to 999.

COL: indicates the number of columns, supports unsigned integer values ranging from 1~999.

LAY: indicates the number of layers, supports unsigned integer values ranging from 1 to 999.

HEIGHT: indicates the layer height, supports 64-bit floating point type positive number value range.

Note that if the layer height exceeds the travelling distance of the robot arm, the alarm will be raised at the time of executing the movement to the pallet POINT that the POINT is unreachable. When defining a pallet, it does not detect whether the POINT is reachable or not.



Example:

```
PALLET ID=0 P[1] P[2] P[3] ROW=5 COL=4 LAY=2 HEIGHT=10
```

'Defines a 5-row, 4-column, 2-layer tray with an ID number of 0 and a height of 10mm per layer

```
LR[1]=PALLET ID=0 ROW=1 COL=2 LAY=1 'Get the POINT of tray 0,
```

row 1, column 2, level 1 and assign to LR[1]

```
L LR[1] 'Linear motion to LR [1]
```

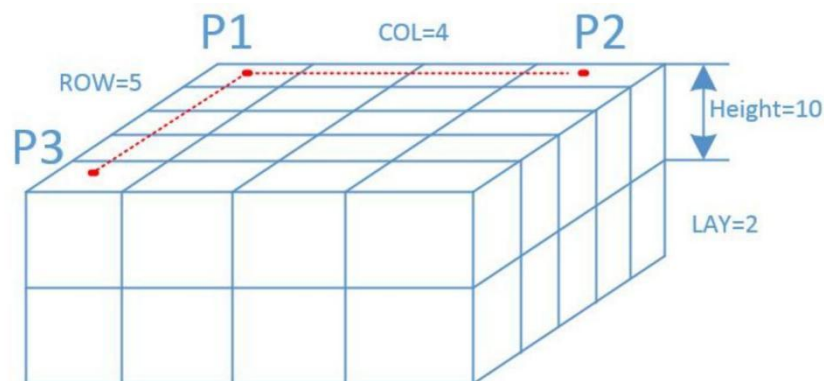


Figure 25-1 Pallet Schematic



Tips:

(1) When defining the pallet, it is required that the coordinate system of the tool artefacts used for the three points P[1], P[2], P[3] is required to be the same, if it is not

the same an alarm will be thrown.

(2) single row | single column of the tray, according to the value of ROW and COL to determine, do not need to consider P [1], P [2], P [3] a two-point coincidence between whether or not, for example: a single row of the tray (ROW = 1), this time, do not take into account the P3 point, that is, the P3 point can not be set up or set up and the P1 point coincides. Single row tray is the same.

Single row tray: PALLET ID= 1 P[1] P[2] P[3] ROW=1 COL=4 LAY=2 HEIGHT= 10

Single row tray: PALLET ID= 1 P[1] P[2] P[3] ROW=4 COL= 1 LAY=2 HEIGHT= 10

11.30 SAVE variable save command

The SAVE instruction can be used to save the corresponding register variable in the variable list.

Grammar format:

SAVE FILE=<file name>



Example:

```
SAVE FILE= "LR"      'Save LR register file
```



Tip:

The name of FILE supports files for register variables such as UT, UF, R, JR, LR, SR, etc., as well as saving configuration files for other functions of the system.

11.31 PRINTF Information Printing Command

This command prints display information directly on the Oscillator information display bar. When printing multiple messages, use a semicolon to separate them.

Grammar format:

PRINTF <string> <string|double> {;<string> [string|double]}



Example:

```
SR[1]="tom"
SR[2]="jerry"
R[1]=10
R[2]=20
PRINTF "height:" R[1]; "thickness:" R[2]      'Height:10;Thickness:20
PRINTF SR[1]
PRINTF SR[1] SR[2]      ' Syntax error, missing data item
```

11.32 THROW Custom Alarm Commands

The THROW instruction is used to throw user-defined alarm information. When the corresponding alarm level signal is thrown, the system will make corresponding operation according to the alarm level.

Grammar format:

THROW <string> LEVEL=FATAL|ERROR|WARNING|NOTE|INFO

FATAL: Critical unrecoverable error, the robot stops moving, the motion buffer is emptied, the programme enters an error state and the cache is emptied, unrecoverable (the programme cannot be resumed).

ERROR: Recoverable error, the robot stops moving, the programme enters an error state, but the motion buffer is not emptied, and the operation can be resumed.

WARNING: Warning, the system is in a critical state or the user operation may lead to undesirable consequences, such as prompt messages, will not cause the programme or movement to stop.

NOTE: NOTE, a general message giving guidance to the user.

INFO: Information, general system status information or user action records that are not prompted by the user.



Example:

THROW "Abnormal clamping of jaws" LEVEL=ERROR

11.33 Maths function instructions

Maths arithmetic function instructions are included: SQRT 、 SIN 、 COS 、 TAN 、 ASIN 、 ACOS 、 ATAN、 ATAN2 、 ABS 、 TRUNC 、 ROUND.

表 29-1 数学函数详细 Instructions

function name	Instructions	Range of independent variables	give an example	
SQRT	square root	$[0, +\infty]$	R[1]=4 R[2]= SQRT R[1]	R[2]=2
SIN	sine	$[-\infty, +\infty]$	R[1]=90 R[2]= SIN R[1]	R[2]=1
COS	cosine	$[-\infty, +\infty]$	R[1]=90	R[2]=0

			R[2]= COS R[1]	
TAN	tangent	Floating point numbers other than 90±180	R[1]=45 R[2]= TAN R[1]	R[2]=1
ASIN	inverse sine	[-1, 1]	R[1]=1 R[2]=ASIN R[1]	R[2]=90
ACOS	inverse cosine	[-1, 1]	R[1]=1 R[2]=ACOS R[1]	R[2]=0
ATAN	inverse tangent	$[-\infty, +\infty]$	R[1]=1 R[2]=ATAN R[1]	R[2]=45
ATAN2	inverse tangent	Except X=0 and Y=0	R[1]=-1 R[2]=0.5 ATAN2 R[1] R[2] , OUT R[3]	R[3]= -63.435
ABS	absolute value	$[-\infty, +\infty]$	R[1]=-100 R[2]=ABS R[1]	R[2]=100
TRUNC	Rounding (rounding off decimal values)	$[-\infty, +\infty]$	R[1]=5.5 R[2]=-5.2 R[3]=TRUNC R[1] R[4]=TRUNC R[2]	R[3]=5 R[4]=-5
ROUND	Rounding (to the nearest whole number)	$[-\infty, +\infty]$	R[1]=5.5 R[2]=-5.2 R[3]=ROUND R[1] R[4]=ROUND R[2]	R[3]=6 R[4]=-5
DIV	Left and right division operations to take integers	$[-\infty, +\infty]$	R[1]=5 R[2]=2 R[3]=R[1] DIV R[2]	R[3]=2

MOD	Left and right division operations to take the remainder	$[-\infty, +\infty]$	$R[1]=5$ $R[2]=-2$ $R[3]=R[1] \text{ MOD } R[2]$	$R[3]=1$



Hints:

- (1) The independent variable input and output in trigonometric functions are angle values.
- (2) If the denominator in the calculation result is 0, or the dependent variable exceeds the range will report an error.
- (3) Support and mathematical operators mixed operations, such as $R[2] = 30 + \text{ROUND } R[3]$, also supports the function instruction nested operations. For example, it supports $R[2]=\text{ABS SIN } 60$.
- (4) When using the ATAN2 instruction, it is necessary to take it out and use it separately as an independent programme line, store the result of the calculation directly into the R register after OUT, and then use the R register to participate in the mixed operation or make a judgement.

Example:

```

ATAN2 R[2] R[1], OUT R[3]
R[4]=R[3]*6+SQRT R[3]
IF R[3]>100 , GOTO LBL [1]

```

11.34 string instruction

String instructions can be used to convert, cut, splice, and lookup strings. The current system supports global string variables $SR[x]$, each $SR[x]$ variable can store 512 characters.

11.34.1TOSTR Convert String Instruction

The TOSTR instruction enables the conversion of values to strings.

Grammar format:

<string var> = TOSTR(<num expression>)



Example:

```
SR[1] =TOSTR(R[1])
```

Table 30-1 TOSTR Conversion

R[1] value	SR[1] value
1234	"1234"
34.152	"34.152"
23.123456789	"23.123456"



Tip:

Only 6 decimal places are retained when converting floating point numbers.

11.34.2 TOVAL Convert Numeric Instruction

The TOVAL instruction enables the conversion of a string to a numeric value.

Grammar format:

<num var> = TOVAL(<string expression>)



Example:

```
R[1] = TOVAL(SR[1])
```

Table 30-2 TOVAL Conversion

SR[1] value	R[1] value
"1234"	1234
"34.152"	34.152
"98abc"	98
"abcd"	0



Hints:

When SR[i] contains illegal characters, it will convert the correct part before the illegal characters, if all of them are illegal characters, the conversion result will be 0.

11.34.3 STRLEN get string length command

Use the STRLEN instruction to get the length of the string.

Grammar format:

<num var> = STRLEN(<string expression>)



Example:

```
R[1] = STRLEN(SR[1])
```

```
R[2] = STRLEN("hello")
```

Table 30-3 STRLEN Conversion

SR[1] value	R[1] value
"1234"	4
"34.152"	6
"a,,d;!p"	8
""	0

11.34.4 FINDSTR lookup string command

The FINDSTR instruction implements a string lookup function. It searches for the target string from left to right in the string, and returns the first occurrence if it is found, or 0 if the string is not found.

Grammar format:

<num var> = FINDSTR (<string1> ,<obj string2>)



Example:

R[1]=FINDSTR(SR[1], SR[2])

Table 30-4 FINDSTR Conversion

SR[1] value	SR[2] value	R[1] value
"hello world!"	"ell"	2
"hello world!"	"no"	0

11.34.5 SUBSTR Intercept string instruction

The SUBSTR instruction enables string interception.

Grammar format:

<string var> = SUBSTR(<obj string>, <start index>, [char length])

Obj string: the string to be intercepted, this item is mandatory.

start index: the start index of the string to be intercepted, when its value is equal to 0 or 1, it will start from the first position. If it is negative, the start position is counted backward from the last character, and then the length of the specified character is intercepted from that position in the downward direction.

char length: character length value, specifies the length of the character to be intercepted, is optional, if not written then the default return from the specified position after the start of all the characters, SUBSTR ('hello', 2) ---> return 'ello', SUBSTR ('hello', -2) ---> return 'lo'; if the value of the length of the character is more than the value of the length of the object string, then return all the object string.



Example:

SR[1]=SUBSTR (SR[2], R[1], R[2])

Table 30-5 SUBSTR Conversion

SR[1] value	SR[2] value	R[1] value	R[2] value
"ell"	"hello world!"	2	3
"world!"	"hello world!"	7	10
"!"	"hello world!"	-1	2
"ld"	"hello world!"	-3	2

11.34.6 string splicing

Grammar format:

<string var1> = <string expression1> + <string expression2>

Use the plus sign to perform string concatenation functions. The operands on both sides of the plus sign are required to be of the same string type.

11.34.7 Character Cutting Case Procedure

Example: Cut SR[0]='T1,1.123,2.0,3.0' to SR[1]=T1 SR[2]=1.123 SR[3]=2.0 SR[4]=3.0

```
R[0] = FINDSTR(SR[0],",")
```

```
SR[1]= SUBSTR(SR[0],0,R[0]-1) SR[0]= SUBSTR(SR[0],R[0]+1)
```

```
' SR[0]=T1,1.123,2.0,3.0 R[0]=3 ' Intercept T1 on the left ,
```

```
SR[1]=T1 ' Intercept the right part of the string
```

```
R[0] = FINDSTR(SR[0],",") ' SR[0]=1.123,2.0,3.0
```

```
SR[2]= SUBSTR(SR[0],0,R[0]-1) ' Intercept 1.123 on the left , SR[2]=1.123
```

```
SR[0]= SUBSTR(SR[0],R[0]+1) ' Intercept the right part of the string, after the interception
```

```
SR[0]=2.0,3.0
```

```
R[0] = FINDSTR(SR[0],",") ' SR[0]=2.0,3.0
```

```
SR[3]= SUBSTR(SR[0],0,R[0]-1) ' Intercept 1.123 on the left SR[3]=2.0
```

```
SR[0]= SUBSTR(SR[0],R[0]+1) ' Intercepting the right part of the string, SR[0] = 3.0 after interception
```

```
SR[4]=SR[0] 'SR[4]=3.0
```

11.35 SOCKET communication commands

The system supports the SOCKET communication function, which has two functional forms.

The first is the protocol-free SCOKET character communication, users can call the SOCKET function in the background programme to establish communication, and then complete the operation of sending and receiving data. The system supports network connection status monitoring, and will automatically alarm when communication is abnormal. This form of robotic arm can be used as a client or server; as a server allows multiple clients to connect and respond to multiple client data. The specific functions are described in the following chapters.

The second is the SOCKETCMD network terminal function. The system provides a set of defined

SOCKETCMD character stream protocol, the host computer sends corresponding strings to the robotic arm according to the protocol to control the robotic arm to enable, move to the point, load and run the program and other operations, or get the status data of the robotic arm, such as IO, current position, etc. For details, see the ‘V1.6. 11_SocketCmd Secondary Development Communication Protocol Manual_A01_CN’. For detailed functions, please refer to ‘ V1.6. 11_SocketCmd Secondary Development Protocol Manual_A01_CN

Instructions: In the following SOCKET function instructions, all of them can accept or not accept the return value of the function. That is, the following programming is correct

R[1]=OPENSOCKET OPTIONS=0 HANDLE= 1 ‘Receiving the return value

OPENSOCKET OPTIONS=0 HANDLE= 1 ‘Does not receive a return value.

11.35.1 OPENSOCKET creation command

Creates a socket. The syntax format is as follows:

INT OPENSOCKET OPTIONS=<int num> HANDLE=<int device number>

OPTIONS: 0 means the data is buffered until the buffer is full or timeout, this mode will be used to improve network utilisation; 1 means no delay, send the data immediately.

HANDLE: i.e. device handle, the value range is 1~255.

Function return value, create success: 0, create failure: different reasons for different return values, see the attached table for details, ‘SOCKET error code’.



Example:

OPENSOCKET OPTIONS=0 HANDLE= 1 ‘Does not receive a return value

R[0]=OPENSOCKET OPTIONS=0 HANDLE= 1 ‘Receiving the return value

11.35.2 CLOSESOCKET shutdown command

Closes the socket. The syntax format is as follows:

INT CLOSESOCKET HANDLE=<device number>

Function Return Value, Close Success: 0, Close Failure: Different return values for different reasons, see the attached table ‘SOCKET Error Code’ for details.



Example:

CLOSESOCKET HANDLE= 1 ‘Does not receive a return value

R[0]=CLOSESOCKET HANDLE= 1 ‘Receiving the return value

11.35.3 ISHANDLEOPEN judgement instruction

The ISHANDLEOPEN instruction is used to determine whether the socket is open and the communication connection is successful. A return of true indicates that the socket is open and the communication connection is successful, while a return of false indicates that the socket is not open or the communication is not connected. When converting a Boolean variable to floating point, 1.0 means true and 0 means false.

Grammar format:

BOOL ISHANDLEOPEN(int <device number>)



Example:

```
R[1]=ISHANDLEOPEN(1)
```

11.35.4 The CONNECT command

Connection server for situations where the robotic arm is acting as a client.

Grammar format:

INT CONNECT(int <device number>, string <ip address>, int <port number>, {int timeout})

device number: device handle, the socket created by opensocket.

ip address: IP address, IP address in string form.

port number: port number, 30000-65535.

timeout: timeout time in milliseconds, the system default connection timeout is 3000 milliseconds (3 seconds), and it will exit automatically after the timeout. This parameter is optional.

Function return value: connection success: 0; connection failure, timeout return: different reasons for different return values, see the attached table 'SOCKET error code' for details.



Example:

```
OPENSOCKET OPTIONS= 1 HANDLE=20
R[0]=CONNECT(20,"127.0.0.1",30000)
```



Tip:

The return value of the function is an optional parameter, the user can choose whether to receive the return value or not, if the timeout does not connect to the server then the function returns.

11.35.5 ACCEPT Monitor instructions

ACCEPT instruction is used in the case of a robotic arm as a server, the server side opens a socket (handle=1 in this case), and then blocks waiting for the client to connect (accept), when the client connects, the server side creates a new socket and returns to save it in fd, the server side uses this new socket to interact with data from the client that is already connected, and again blocks listening to wait for the next client to connect. The server uses this new socket to interact with the connected client and again blocks listening for the next client to connect. You can send information to the corresponding fd to achieve one-to-many communication.

The syntax is as follows:

INT ACCEPT(int <device handle>, string <ip>, int <port>, int& <fd>, {int timeout})

device handle: Device handle, a socket created with the opensocket command to listen for client connections.

ip: Local IP address string.

port: the port on which the machine receives data, 30000-65535.

fd: the handle of the newly created socket when the client connects, the server uses this new socket to interact with the connected client.

timeout: timeout in milliseconds, the default timeout is 3000 milliseconds. This parameter is optional.

Function return value: client connection is successful and return: 0; listening timeout return: different reasons for different return values, see the attached table 'SOCKET error code' for details.



Example:

```
OPENSOCKET OPTIONS=0 HANDLE= 1
R[0] = ACCEPT(1," 127.0.0.1",30000, R[1] )    'Use R[1] to receive connected
sockets
If
    Throw
Endif
SEND(R[1], "hello world")
```

11.35.6 SEND send command

The SEND instruction is used to send data, both server-side and client-side through this function. Since the maximum user buffer is 512 bytes, the maximum amount of data that can be sent at one time is 512 bytes, and it will be automatically truncated if it exceeds that amount.

Grammar format:

INT SEND(int <device handle>, string <data>)

device handle: Device handle, socket created using the opensocket command.

data: the string to be sent.

Function return value: send successfully returns the number of bytes sent successfully; send failure: different reasons for different return values, see the attached table 'SOCKET error code' for details.



Example:

```
OPENSOCKET OPTIONS=0 HANDLE=20
R[0]=SEND(20,"hello world")    'Sends hello world , returns 11 if successful.
SEND(20,SR[1])                'Sends the value inside the SR[1] register
variable
```

11.35.7 RECEIVE Receive Command

RECEIVE function is used to receive data, if receiving data is unsuccessful, such as handle is not created, then the function returns -1, timeout does not receive the data returns 0. Whether it is the server or the client to receive data through this function. Because the user buffer is 512 bytes, so the single reception of up to 512 bytes of data, if the host computer sends more than 512 data, must be received several times. Users can customise the format frame in engineering practice to ensure the reliability of communication.

Grammar format:

INT RECEIVE(int <device handle>, string& <data>, {int timeout})

device handle: device handle, socket created with opensocket command.

data: save the string to be retrieved.

timeout: timeout in milliseconds, the default receive timeout is 200 milliseconds. This parameter is optional.

Function return value: the number of bytes received is returned if the reception is successful; reception failure: different reasons for different return values, see the attached table 'SOCKET error code' for details.



Example:

```
OPENSOCKET OPTIONS=0 HANDLE=20
R[0]=RECEIVE(20, SR[1])    'Receive data and save in SR[1] variable
```

11.35.8 BRECBUFEMPTY Directive

The BRECBUFEMPTY instruction is used to determine if there is data in the receive buffer.

Grammar format:

INT BRECBUFEMPTY (INT HANDLE)

Function Return Value: 0 means the cache is empty; 1 means there is data in the cache.



Example:

```
OPENSOCKET OPTIONS=0 HANDLE=20
R[0]=BRECBUFEMPTY (20)    'Determine if there is data in the receive
buffer
```

11.35.9 SOCKETMONITOR monitor command

SOCKETMONITOR is used to set up asynchronous monitoring of the communication connection status, and the user can set up whether or not to alarm when the connection is disconnected. The system monitors the situation from [Connected] to [Disconnected]. When the connection is disconnected, the system automatically throws an alarm | warning.

Grammar format:

SOCKETMONITOR(int <device handle>, int <leve>)

device handle: device handle, use opensocket command to create socket.

level: response level, 1: alarm if disconnected, PRG programme stops running; 2: warning is thrown if disconnected, no alarm, PRG programme executes normally.



Example:

```
SOCKETMONITOR(10, 1)
```

11.36 MODBUS Master Communication Command

The system supports MODBUS communication function. When the robot arm is used as a MODBUS master, the user has to use commands to complete the read/write business logic inside the PRG; when the robot arm is used as a slave, the system provides ModbusCmd external control protocol, and the user only needs to turn on the ModbusCmd function switch on the UI of the demonstrator, and wait for the upper computer to connect to establish communication.

The robotic arm can be a master and a slave at the same time. Because the protocol-less universal Modbus function (in the form of commands) and the ModbusCmd external control protocol function are independent functions, the memory opened by the two functions is independent of each other. The robot arm can be used as a master to communicate and interact with the PLC and other devices on the production line using the Modbus function commands, and at the same time, the robot arm can be used as a slave to

communicate and interact with the touch screen using the ModbusCmd function, as described in the 'Instructions for the use of the Modbus Slave Control Protocol'.

When the robot arm acts as a master, the user needs to write the PRG programme and load and execute the relevant commands before the communication is established. The instruction is a triggered operation, and the user will only read or write the operation when the instruction is called and executed. Users can use while loop inside the background programme to achieve cycle read/write operation.

Instructions: All of the following function instructions may or may not accept the return value of a function. That is, all of the following programming is correct

R[1]=OPEN_MODBUS_TCP("10.10.56.100",502,1,0) 'Receiving the return value

OPEN_MODBUS_TCP("10.10.56.100",502,1,0) 'Does not receive a return value

11.36.1 OPEN_MODBUS_TCP Create Command

Create a MODBUS master, and the robot arm communicates with the slave as the master (client). In case of one-to-many, the corresponding IP address and port number must be called for communication separately, and the port number cannot be the same.

Grammar format:

INT OPEN_MODBUS_TCP(int <SlaveID>, string <IP>, int <Port>, int <HighLowType>)

SlaveID: Slave ID number, 1~255.

TCP/IP: IP address of the slave device (i.e. server).

Port: Port number of TCP communication connection, 502, 503, 504, 505, 506, etc. can be used.

HighLowType: the order of high and low bits, CDAB, then use 0 to indicate, ABCD, then use 1 to indicate.

The function returns a value of 0 for success, -1 for unsuccessful master creation, and -2 for unsuccessful slave connection. If there is an exception, the controller throws a warning message corresponding to the return value.



Example:

```
R[1]=OPEN_MODBUS_TCP(1,"10.10.56.100",502,0)  'Open and connect
slave 1
R[2]=OPEN_MODBUS_TCP(2,"10.10.56.101",503,0)  'Open and connect
slave 2
```

11.36.2CLOSE_MODBUS_TCP shutdown command

The robot arm acts as a master (client) to close the connection with the slave of this ID. The function returns a value, 0 for success, -1 for failure.

Grammar format:

INT CLOSE_MODBUS_TCP(int <SlaveID>)



Example:

```
R[1]= CLOSE_MODBUS_TCP(1)  'Close communication with slave 1.
```

11.36.3OPEN_MODBUS_RTU Create Command

When the robot arm communicates with multiple slaves as a master, the robot arm communicates with the slaves as a master (client). For one-to-many, the corresponding serial port number and slave station ID must be called up separately.

Grammar format:

INT OPEN_MODBUS_RTU(int<SlaveID>, int<SerialPort>, int<Baud>, int<DataBit>, int <StopBit>, int<ParityCheck>, int <HighLowType>)

SlaveID: ID number of the slave, 1~255.

SerialPort: serial port number.

Baud: Baud rate, can choose 9600, 115200, etc.

DataBit: Data bit, can be set to 5, 6, 7, 8.

StopBit: Stop bit, can be set to 1, 2.

ParityCheck: Parity check. 0 means none, 1 means odd check, 2 means even check.

HighLowType: the order of high and low bits, CDAB, then use 0 to indicate, ABCD, then use 1 to indicate.

The function returns a value of 0 for success, -1 for unsuccessful master creation, and -2 for unsuccessful slave connection. If there is an exception during creation, the controller throws a warning message corresponding to the return value.



Example:

```
R[1]=OPEN_MODBUS_RTU(1, 3,9600,8,1,0,1)  'As master, connect the
slave with ID= 1, using serial port 3.
R[1]=OPEN_MODBUS_RTU(2, 4,9600,8,1,0,1)  'As master, connect the slave
with ID=2, using serial port 4
```

11.36.4 CLOSE_MODBUS_RTU shutdown command

The robotic arm acts as a master and closes the MODBUS RTU communication with the slaves.

Grammar format:

INT CLOSE_MODBUS_RTU(int SlaveID)



Example:

R[1]=CLOSE_MODBUS_RTU(1) 'Disables communication with slave 1.

11.36.5 READ_MDB_COILSTATUS read command

The robotic arm acts as a master to read the coil data from the slaves.

Grammar format:

INT READ_MDB_COILSTATUS(int<SlaveId>, int<StartAddr>, int<CoilNum>, bool <Var>)

SlaveId: Slave address.

StartAddr: start address of coil state (0-09999).

CoilNum: number of coils to be read.

Var: IO type variable, used to store the value of the read coil, can use DO[x]|R[x] variable to save. The function returns 0 for a successful read and -1 for a failed read.



Example:

READ_MDB_COILSTATUS(1, 1, 2, R[1]) 'Read the values of the 2 coils of slave 1 starting from address 1 and store them in the R[1], R[2] variables

11.36.6 WRITE_MDB_COILSTATUS Write Command

The robotic arm serves as the master station to write data to the slave station coil.

Grammar format:

INT WRITE_MDB_COILSTATUS(int<SlaveId>, int<StartAddr>, int<CoilNum>, bool <Value>)

SlaveId: slave address.

StartAddr: start address of the coil (0-09999), unit bit.

CoilNum: number of coils to be written.

Value: value of bit type (0-255), set the coil value of ModBus slave according to the value of this variable.

The function returns 0 for a successful write and -1 for a failed write.



Example:

R[1]=1

R[2]=1

R[3]=0

WRITE_MDB_COILSTATUS(1, 0, 3, R[1]) 'Set the values in coil address 0, 1, and 2 of slave 1 to 1, 1, and 0, in that order.

11.36.7 READ_MDB_INPUTSTATUS read command

The robotic arm acts as a master reading the slave discrete coil data inputs.

Grammar format:

INT READ_MDB_INPUTSTATUS(int<SlaveId>, int<StartAddr>, int<Num>, bool <Var>)

SlaveId: Slave address.

StartAddr: start address of the coil state (0-09999).

Num: number of discrete inputs to be read.

Var: used to store the value of the read coil, can use DO[x]|R[x] register. The function returns 0 for a successful read and -1 for a failed read.



Example:

```
READ_MDB_INPUTSTATUS(1,0,2,R[10])    'Read the value of the 2 discrete
inputs from slave 1 starting at address 0 and store them in R[10], R[11]
```

11.36.8 READ_MDB_INPUTREG read command

The robotic arm acts as a master to read data from the slave input registers.

Grammar format:

INT READ_MDB_INPUTREG(int<SlaveId>, int<StartAddr>, int<Num>, dWord <Var>, int <DataType>)

SlaveId: Slave address.

StartAddr: start address of coil state (0-09999).

Num: number of input registers to be read.

Var: used to store the input registers read, can use R[x] variable.

DataType: data type of input register|holding register, use 0 to indicate 16-bit INT; use 1 to indicate 32-bit DINT; use 2 to indicate 32-bit FLOAT.

The function returns 0 for a successful read and -1 for a failed read.



Example:

```
R[1]=READ_MDB_INPUTREG(1,0,2,R[10],1)    'Read the values of the 2
input registers of slave 1 from address 0 and store them in R[10] and R[11].
```

11.36.9 READ_MDB_HOLDREG read command

The robot arm acts as a master to read the slave save register data

Grammar format:

INT READ_MDB_HOLDREG(int<SlaveId>, int<StartAddr>, int<Num>, dWord <Var>, int <DataType>)

SlaveId: slave address.

StartAddr: start address of the register (0~09999).

Num: number of read data.

Var: save the address of the holding register.

DataType: data type of input register|holding register, use 0 to indicate 16-bit INT; use 1 to indicate 32-bit DINT; use 2 to indicate 32-bit FLOAT.

The function returns 0 for a successful read and -1 for a failed read.



Example:

```
R[1]=READ_MDB_HOLDREG(1,0,2,R[10],1) 'Read the 2 elements starting
from 0 of slave 1 and store them in R[10], R[11]
```

11.36.10 WRITE_MDB_HOLDREG Write Command

The robot arm acts as a master to write to the slave to save the register data.

Grammar format:

```
INT WRITE_MDB_HOLDREG(int<SlaveId>, int<StartAddr>, int<Num>, dword <Value> ,
int
<DataType> )
```

SlaveId: slave address.

StartAddr: start address of the register (0~09999).

NUM: number of registers to be written.

Value: Write this value to the corresponding address of Modbus.

DataType: data type of input register|holding register, use 0 to indicate 16-bit INT; use 1 to indicate 32-bit DINT; use 2 to indicate 32-bit FLOAT.



Example:

```
R[10]=1
R[11]=1
R[12]=0

R[1] = WRITE_MDB_HOLDREG(1 , 0, 3 , R[10], 1) 'Write the data
R[10]=1, R[11]=1, R[12]=0 to the 3 elements after the start address of slave 1 is 0
```

11.37 macro instruction

11.37.1 HANDPICK jaw commands

This instruction is used for the gripper function application, which can control multiple output signals and monitor and judge multiple input signals at the same time to achieve the processing of signal linkage and inspection. The application for when the signal triggered to open or close the gripper jaws, and then judge

whether to open or close in place, not in place, then need to be alarmed output of the combination of functions of the process application.

Grammar format:

MAC HANDPICK DO=<num expression> VALUE=ON|OFF DI=<num expression> VALUE=ON|OFF T=<time num>, OUT;

DO: Output index number indicating the IO number associated with the gripper open/close action. The following VALUE option value indicates that the DO is equal to ON | OFF when the gripper is opened | closed.

DI: Input index number, indicates the IO number associated with detecting whether or not the gripper open/close action is in place, followed by a VALUE option value that indicates that the gripper open/close is in place when the DI is equal to ON|OFF.

T: timeout detection time, the unit is ms, that when DO = ON, the system waits for DI = ON signal to detect whether the signal in place, if T = 0, the system has been waiting for the signal, if T > 0, if beyond the set time clamping jaws are not yet in place, the system will issue an alarm, the alarm content for the 'clamping jaws to open or clamping abnormality'. If T=0, the system waits for the signal.



Instructions Example:

```
MAC HANDPICK DO=0 VALUE=ON DI=0 VALUE=ON T= 1000 ,
OUT; L P[0]
```

The effect of the above statement line is similar to the following program line:

```
DO[0]=ON
WAIT TIME= 1000
IF DI[0]=OFF THEN
THROW "Abnormal jaw opening or clamping" LEVEL=ERROR
ELSE
L P[0]
END IF
```



Program Example:

```
L P[0] VEL= 1500 CNT= 100 ACC=300 DEC=300 'Move to teach point P0
MAC HANDPICK DO=0 VALUE=ON DI=0 VALUE=ON T= 1000,
```

OUT; 'When the target point P[0] is reached, the 'open output signal' output signal DO[0] will be set to TRUE, waiting for the 'open detection signal' input

signal DI[0] to be set to TRUE, and the system will alarm if the waiting time exceeds 1 second. 'Jaw opening or abnormal clamping' .

```
R[0]=1 'R[0] assigned to 1
```

```
L P[1] VEL= 1500 CNT= 100 ACC=300 DEC=300 'Move to teach point P1
```

```
MAC HANDPICK DO=R[0] VALUE=ON DI=R[0] VALUE=ON T=2000,
```

OUT; 'When the target point P[1] is reached, the 'open output signal' output signal DO[R[0]] will be set to TRUE, wait for the 'open detection signal' input signal DI[R[0]] to be set to TRUE, and the system will alarm when the waiting time is

exceeded by 2 seconds. 'Jaw opening or abnormal clamping'.



Tip:

After the alarm is triggered, the alarm will be stopped, the alarm can be reset, the statement will be re-executed after a successful reset, and if the conditions are not met, the alarm will continue to be triggered.

11.37.2 SOFTFLOAT soft float command

The soft float function allows the arm to float in response to an external force in the direction of the force when the arm is subjected to an external force in the stationary state. The position of the arm at rest where the soft float function is turned on is called the reference position. After soft float is turned on, the soft float effect is determined by the parameter configuration item of float. The parameter configuration is done on the Oscillator UI.

The syntax format is as follows:

MAC SOFTFLOAT <n> [(TIMES=numerical value) | (DI_NUM=numerical value DI_VALUE=ON|OFF) | (R_NUM=numerical value R_VALUE=numerical value)], OUT

SOFTFLOAT<n>: specifies which group of float parameters to use, the value of n ranges from 0 to 7, and the grouping expression specifies the judgement condition to stop the soft float.

TIMES: specifies the time to float, and stops the soft float when the time is reached.

DI_NUM and DI_VALUE: It specifies the state of this input IO is equal to DI_VALUE to stop soft floating.

R_NUM and R_VALUE: Specifies that the soft float stops when the value of the R register is equal to R_VALUE.

The stop condition of soft float can be in various formats, such as time specified, IO state specified, R register value specified, etc., which can be written by the user according to the actual requirements.



Instruction Example:

Example1:

MAC SOFTFLOAT 0 ,OUT 'Without specifying the stopping condition of Soft Float, Soft Float will be kept floating after it is turned on, and Soft Float will not be stopped until the uninstallation process.

Example2:

MAC SOFTFLOAT 0 TIMES=2000, OUT 'Soft float is switched on and switched off after 2S of float, the

Example3:

MAC SOFTFLOAT 0 DI_NUM= 1 DI_VALUE=ON, OUT 'Enable soft float and block, wait for DI[1]=ON to disable soft float

Example4:

MAC SOFTFLOAT0 R_NUM=9 R_VALUE= 10.0, OUT 'Turn on soft

float and block, wait for R[9] = 10.0 to turn off soft float



Procedure Example:

J P[0] 'The soft float can only be switched on after the robot arm has moved to the point and stopped.

MAC SOFTFLOAT 0 TIMES=2000, OUT 'Starts soft float with group 0
parameters and closes after 2 seconds of float time

J P[1] 'Movement to point P1



Tips:

- (1) The softfloat function only supports models that have been adapted to the dynamics model. If the model does not support softfloat, a warning will be thrown when executing the MAC SOFTFLOAT command, and there will be no floating effect.
- (2) When soft float is enabled, the current tool, workpiece, and load level parameters must be used correctly.
- (3) In the following cases, the soft float function is automatically turned off: when the programme is paused, stopped or unloaded; when single-step debugging,
arbitrary line debugging, backward; when the system alarms. The system throws an alert when the soft float function is switched off.

12 Client service

12.1 Technical Support Enquiry

Introductory	This document provides information about the operation and handling of the robotic arm and can help you troubleshoot problems.

Information	The following information is required to provide technical support:

Description of the problem, including:

Steps of operation performed some time before the alarm fault. n

Information about the duration and frequency of the fault.

As much detailed information as possible about the hardware and software components of the overall system, such as:

Robot model, nameplate number

Controller model, factory number

System software name and version n

Diagnostic data package

uThe diagnostic data export in the system software allows you to consult your technician about exporting streamlined data.

nExisting installed plug-in packages

nProgram files for existing applications

nAdditional axis cases for existing configurations

13 Appendix

13.1 Glossary

name	Instructions
flangeless	1 The end of the arm, i.e. the end of the 6-axis robot, is flanged.
singularity	1 During the motion of a six-axis robot, if the angle of the fifth axis is 0 ($J5=0$) and the fourth axis is co-linear with the sixth axis, the angle of the two axes of the fourth and sixth axes will produce an infinite number of sets of solutions when the kinematic inverse solution is calculated. Such positions are called singularities in robotics. 1 In practice, when the six-axis robot moves in a spatial POINT coordinate system, it is most likely to pass through the singularity, and rapid rotation of the joints will occur, so it should try to avoid any singular POINT in the motion planning.
Euler's angle	1 Euler angles are a method of spatial coordinate attitude in a robotic arm, which determines the orientation of the coordinates through three angles of rotation. The Euler angles for the Wachovia III use the ZYX Euler angle type, which means that the object is first rotated by one angle around the z-axis, then by one angle around the rotated y-axis, and finally by one angle around the rotated x-axis.

13.2 Global Motion Parameters Command Table

Parameter name	Value range	Default value	MeaningInstructions
CNT_TYP E	0/1/2	0	Smoothing type parameters
CNT	0~100	0	Global Smooth Transition Parameters
J_VEL	The unit is percentage and is divided according to the mode: Manual T1: 1~10 Manual T2: 1~20 Automatic/External: 1~100	100	Joint velocity
J_ACC	1~200	100	Joint acceleration ratio in per cent
J_DEC	1~200	100	Joint reduction ratio in per cent
L_VEL	Limited according to mode: Manual T1: 1~125 mm/s Manual T2: 1~250 mm/s Auto/External: 1~2400 mm/s, max. 2400 but different for each model.	Maximum values	Linear velocity
L_ACC	1~200	100	Linear acceleration ratio in per cent
L_DEC	1~200	100	Linear reduction ratio in per cent
L_VROT	Limited according to mode: Manual T1: 1~30 °/s Manual T2: 1~80 °/s Auto/External: 1~120 °/s , max. 120 but different for each model.	Maximum values	Linear Attitude Velocity
C_VEL	Limited according to mode: Manual T1: 1~125 mm/s Manual T2: 1~250 mm/s Auto/External: 1~2400 mm/s, max. 2400 but different for each model.	Maximum values	Circular arc velocity
C_DEC	1~200	100	Circular reduction ratio in per cent
C_ACC	1~200	100	Circular acceleration ratio in per cent
C_VROT	Limited according to mode: Manual T1: 1~30 °/s Manual T2: 1~80 °/s Auto/External: 1~120 °/s , max. 120 but different for each model.	Maximum values	Circular Attitude Velocity
SMOOTH	1~9	5	Parameters of softness level

13.3 Table for assigning operating privileges of the demonstrator

First level menu	Secondary Menu	Third level menu	User group		
			Production staff	Debugger	administrators
File management	Programme data file management	—	√	√	√
	User PLC file management	—	View/Exportable	√	√
	System parameter file management	—	View/Exportable	√	√
File operation	newly built	—	Check out	√	√
	load	—	/	√	√
	save	—	√	√	√
	removing	—	/	√	√
	backing up	—	/	√	√
	resumption	—	/	√	√
	multiple choice	—	/	√	√
	rename	—	/	√	√
	lock	—	/	√	√
	Switch on/off Instructions	—	√	√	√
	On/Off status	—	√	√	√
	Deactivate Waiting	—	√	√	√
	optimisation	—	√	√	√
	turn on	—	√	√	√
configuration	Demonstrator Configuration	user group	√	√	√
		Alternate Key Configuration	Check out	√	√
		Lock Password Setting	—	√	√
		Robot Communication Configuration	Check out	√	√

	Controller Configuration	Running configuration	Check out	√	√
		Encoding/Decoding Configuration	Check out	√	√
		Regional configuration	Check out	√	√
		Controller IP Configuration	Check out	√	√
		Axis Group Configuration	Check out	Check out	√
		Co-group configuration	Check out	√	√
		Timeout setting	√	√	√
		Oscillating configuration	/	√	√
		Back to Zero Procedure Configuration	/	√	√
		Speed Configuration	Check out	√	√
		Back end Programming	Check out	√	√
		path-holding function	Check out	√	√
	Kinetic function	Mounting Attitude Setting	Check out	√	√
		Ontological friction identification	Check out	√	√
		Load Configuration	/	√	√
		Collision Detection Configuration	/	√	√
		Drag-and-drop instructional	/	√	√

		settings			
		Drag Track Recording	/	√	√
Demonstrate	Digital inputs/outputs	——	/	√	√
		——	/	√	√
	Analogue inputs/outputs	——	Check out	√	√
	Physical location	——	√	√	√
	Variable list	——	√	√	√
Diagnostic	Running log	——	Check out	√	√
	Diagnostic data export	——	√	√	√
Put into operation	User tool calibration	——	√	√	√
	User workpiece calibration	——	Check out	√	√
	Co-group calibration	——	Check out	√	√
	External shaft reduction ratio calibration	——	Check out	√	√
	20-point calibration	——	Check out	√	√
	Zero calibration	——	Check out	√	√
	Software Limit Switches	——	Check out	√	√
Information	Version information	——	Check out	√	√
	System Information	——	Check out	√	√
	Robot parameters	——	Check out	Check out	√
	Language Settings	——	/	/	

Systems	Reveal the Teachers	—	/	/	√
	Clearance system	—	/	/	√
	Shut down the system	—	/	/	√
	Reboot	—	/	/	√
	System upgrades	—	Check out	√	√
	Authorisation	—	Check out	√	√
Interface buttons	Activation button	—	Check out	√	√
	Pause button	—	√	√	√
	Uninstall button	—	√	√	√
	Rewind button	—	/	√	√
	Switching between manual and automatic modes	—	/	√	√
	Tool workpiece switching	—	√	√	√
	The way the programme runs	—	/	√	√
	Incremental manual movement setting	—	/	√	√
	Manual magnification adjustment	—	/	√	√
	Automatic magnification adjustment	—	/	√	√
	Select Coordinate System	—	/	√	√
	Programme Control Soft Buttons	—	/	√	√

	Auxiliary Button Operation		/	√	√
	Manual movement operation	—	/	√	√
	One button to return to zero	—	√	√	√
	—	—	/	√	√
Craft kit	Process package management	—		√	√
Programmes After loading	Online Editing	—	Check out	√	√
	Switching Mode	—	/	√	√

13.4 List of external signals

Table of signals in relation to modes of operation:

	Manual T1/T2 mode	Automatic mode	External mode
External Input Signal	Null	Valid (except loader signal iPRG_LOAD)	Efficiently
External Output Signal	Efficiently	Efficiently	Efficiently
Oscillator button	Efficiently	Partially valid	Partially valid

System input signal table:

Signal Name	Instructions	Effective method	range of action
iPRG_START	Start program signal.Starts the loaded user programme running.	Decline in effect along	Automatic mode, external mode
iPRG_PAUSE	Suspend programme signals.Suspends user programme operation.	Decline in effect along	Automatic mode, external mode
iPRG_STOP	Stop programme signal.Stops the user programme from running.	Decline in effect along	Automatic mode, external mode
iPRG_LOAD	Load program signals.Loads the specified user programme.	Rising edge in effect	External mode
iPRG_UNLOAD	Unload programme signal.Uninstalls the program in the ready state.	Decline in effect along	Automatic mode, external mode
iENABLE	System enable signal.	Rising Edge Enable, Set 0	Automatic

		Break Enable	mode, external mode
iCLEAR_FAULT S	Clear the error signal.	Rising edge in effect	Automatic mode, external mode
iSAFE_SPEED	Safety speed trigger signal. When triggered, it causes the arm to reduce its operating speed.	Decline in effect along	Automatic mode, external mode
iSAFE_DOOR	Safety door trigger signal	Decline in effect along	Automatic mode, external mode
iSHARED_EN[0]	Shared Area [0] Enable Switch	Rising Edge Enable, Set 0 Break Enable	Manual mode, automatic mode, external mode
iSHARED_EN[1]	Shared Area [1] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[2]	Shared Area [2] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[3]	Shared Area [3] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[4]	Shared Area [4] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[5]	Shared Area [5] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[6]	Shared Area [6] Enable Switch	Enable on rising edge, set to 0 to close	As above

iSHARED_EN[7]	Shared Area [7] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[8]	Shared Area [8] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[9]	Shared Area [9] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[10]	Shared Area [10] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[11]	Shared Area [11] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[12]	Shared Area [12] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[13]	Shared Area [13] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[14]	Shared Area [14] Enable Switch	Enable on rising edge, set to 0 to close	As above
iSHARED_EN[15]	Shared Area [15] Enable Switch	Enable on rising edge, set to 0 to close	As above

System output signal table:

Signal Name	Instructions	Note	Scope of action
oROBOT_READY	Robot ready signal. This signal is output when it is also satisfied that the system initialisation is complete, the user program is in the loaded state, and it is enabled.	The signal will not be output while the programme is running	All modes
oROBOT_STANDBY	Robotic arm standby signal: the signal is output when the system initialisation is completed, the robotic arm is not enabled, the main program is not loaded and there is no alarm.		All modes
oEMERGENCY_S	Emergency stop circuit on/off	This signal is ON when the	As above

TOP	status signal	emergency stop circuit is ON and OFF when the emergency stop button is pressed.	
oFAULTS	Error.		As above
oENABLE_STAT E	Enable state.		As above
oPRG_UNLOAD	The user program is in an unloaded state.	At the same moment in time, there is one and only one signal output from these signals	As above
oPRG_READY	User program loaded status.		As above
oPRG_RUNNING	User program running status.		As above
oPRG_ERR	User program alarm status.		As above
oPRG_PAUSE	The user programme is suspended.		As above
ols_Moving	The robot is in motion.		As above
oMANUAL_MOD E	The system is in manual mode.	At the same moment in time, there is one and only one signal output from these signals	As above
oAUTO_MODE	The system is in automatic mode.		As above
oEXT_MODE	The system is in external mode.		As above
oHOME	Currently at zero POINT		As above
oMD_ENABLED	MoDBuS Enable Switch	Reserved, not available at this time	As above
oMD_CONN	ModBus connection status	Reserved, not available at this time	As above
oCOL_STATE	Collision detection enable state		As above
oCOLLA_MODE	The system is in a collaborative mode		As above
oREF[0]	Reference point [0]		As above
oREF[1]	Reference point [1]		As above
oREF[2]	Reference point [2]		As above
oREF[3]	Reference point [3]		As above
oREF[4]	Reference point [4]		As above
oREF[5]	Reference point [5]		As above
oREF[6]	Reference point [6]		As above

oREF[7]	Reference point [7]		As above
oAREA_OUT[0]	Area [0] output signal		As above
oAREA_OUT[1]	Area [1] output signal		As above
oAREA_OUT[2]	Area [2] output signal		As above
oAREA_OUT[3]	Area [3] output signal		As above
oAREA_OUT[4]	Area [4] output signal		As above
oAREA_OUT[5]	Area [5] output signal		As above
oAREA_OUT[6]	Area [6] output signal		As above
oAREA_OUT[7]	Area [7] output signal		As above
oAREA_OUT[8]	Area [8] output signal		As above
oAREA_OUT[9]	Area [9] output signal		As above
oAREA_OUT[10]	Area [10] output signal		As above
oAREA_OUT[11]	Area [11] output signal		As above
oAREA_OUT[12]	Area [12] output signal		As above
oAREA_OUT[13]	Area [13] output signal		As above
oAREA_OUT[14]	Area [14] output signal		As above
oAREA_OUT[15]	Area [15] output signal		As above
WELD_GUN_COLLISION	Torch collision alarm	/	As above
WELD_ALARM	Welder Alarm	/	As above
WELD_MACHINE_ALARM	Welding equipment alarms	/	As above

13.5 SOCKET Error codes

Serial number	Error code	Error code resolution
1	-1	Handle does not exist
2	-2	Using connectionless handles
3	-3	Illegal ports
4	-4	Server Listening Failure 1

5	-5	Server Listening Failure 2
6	-6	Client connection failure
7	-7	The client has been connected
8	-8	Type Illegal
9	-9	Listen timeout
10	-100	The network is down.
11	-101	Network unreachable
12	-102	Network disconnected due to reset
13	-103	Software Causes Connection Termination
14	-104	The connection was reset by the other party.
15	-105	No available cache space
16	-106	Transmission Endpoint Connected
17	-107	Transmission endpoint not connected
18	-108	Cannot be sent again after the transmission endpoint is closed
19	-109	Too many references: can't splice
20	-110	Connection timeout
21	-111	Connection denied
22	-112	The mainframe is shut down.
23	-113	Unable to route to host
24	-114	The operation is already in process
25	-115	In-process operations
26	-125	Connection timeout

13.6 System Signal Occupancy Table

Serial number	Flag position	DO No.	Instructions
1	GC_oROBOT_READY	100	Robot readiness
2	GC_oFAULTS	101	Robot Alarm Status

3	GC_oENABLE_STATE	102	Robot Enable State
4	GC_oPRG_UNLOAD	103	Program uninstallation status
5	GC_oPRG_READY	104	Program readiness
6	GC_oPRG_RUNNING	105	Program operating status
7	GC_oPRG_ERR	106	Program error status
8	GC_oPRG_PAUSE	107	Suspended state of the programme
9	GC_oIS_MOVING	108	Robot Motion Status
10	GC_oMANUAL_MODE	109	Manual Mode Status
11	GC_oAUTO_MODE	110	Auto Mode Status
12	GC_oEXT_MODE	111	External Mode Status
13	GC_oREF_0	112	Reference 0 state
14	GC_oREF_1	113	Reference 1 state
15	GC_oREF_2	114	Reference 2 state

Serial number	Occupied functions	DO No.	Instructions
1	Coordination	0	Emergency stop
2	CO610 Strip Driver	16	Light Strip Drive Low
3		17	Light Strip Drive High
4	Socket communication	200	Communication 1 Communication Success Status
5		201	Communicating a Communicating Waiting for Connection Status
6		202	Communication 1 Communication Success Status
7		203	Communication 2 Communication Waiting for Connection Status
8		204	Communication 3 Communication Success Status
9		205	Communication 2 Communication Waiting for Connection Status
10	Codesys	511	Occupancy rate of a heartbeat

Serial number	Flag position	DI number	Instructions
1	GC_PRG_START_N	100	Triggering procedure
2	GC_PRG_PAUSE_N	101	Suspension of proceedings
3	GC_PRG_STOP_N	102	Stopping procedure
4	GC_PRG_LOAD_P	103	Loading Procedures
5	GC_PRG_UNLOAD_N	104	Uninstaller
6	GC_ENABLE	105	Enabling robots
7	GC_CLEAR_FAULTS_P	106	Clear Alarms
8	/	Default 129(virtual)/ 128(real)	Drag Trigger Input
9	/	Default 128(virtual)/ 129(real)	Drag-and-drop instructional trigger inputs

Serial number	Occupied functions	DI number	Instructions
1	Drag	300 (virtual)	Default Drag Trigger Input (Drag Button hardwired depending on model)
2		301 (virtual)	Default Teach-In Trigger Input (Drag Button hardwired depending on model)
3	CO605 Drag	16 (real) Default value	Drag the button to hardwire the trigger input (this IO is the servo's virtual IO, in descending order depending on the number of physical IOs configured)
4		17 (real) Default value	Teach-In Button Hardwired Trigger Input (this IO is a servo virtual IO, in descending order depending on the number of physical IOs configured)
5		18 (real) Default value	Teach-In Button Hardwired Trigger

			Input (this IO is a servo virtual IO, in descending order depending on the number of physical IOs configured)
6		19 (real) Default value	Teach-In Button Hardwired Trigger Input (this IO is a servo virtual IO, in descending order depending on the number of physical IOs configured)
7		16 (real) Default value	Teach-In Button Hardwired Trigger Input (this IO is a servo virtual IO, in descending order depending on the number of physical IOs configured)
8	CO610 Drag	17 (real) Default value	Teach-In Button Hardwired Trigger Input (this IO is a servo virtual IO, in descending order depending on the number of physical IOs configured)
9		18 (real) Default value	Teach-In Button Hardwired Trigger Input (this IO is a servo virtual IO, in descending order depending on the number of physical IOs configured)
10		19 (real) Default value	Teach-In Button Hardwired Trigger Input (this IO is a servo virtual IO, in descending order depending on the number of physical IOs configured)
11		6 (real)	Drag button hardwired trigger input
12	CR605/CR607 Drag	7 (real)	Teach-In Button Hardwired Trigger Input
13	CO610	0	Emergency stop

Guangzhou Aucotech Automation Technology Ltd

Add: Room 903, Building E1, Design City, No. 7, Hexian North Street, Helong Street, Baiyun District, Guangzhou City, CHINA

Tel: +86 20 84898493

E-mail: info@aucotech.com.cn

Web: www.aucotech.com.cn