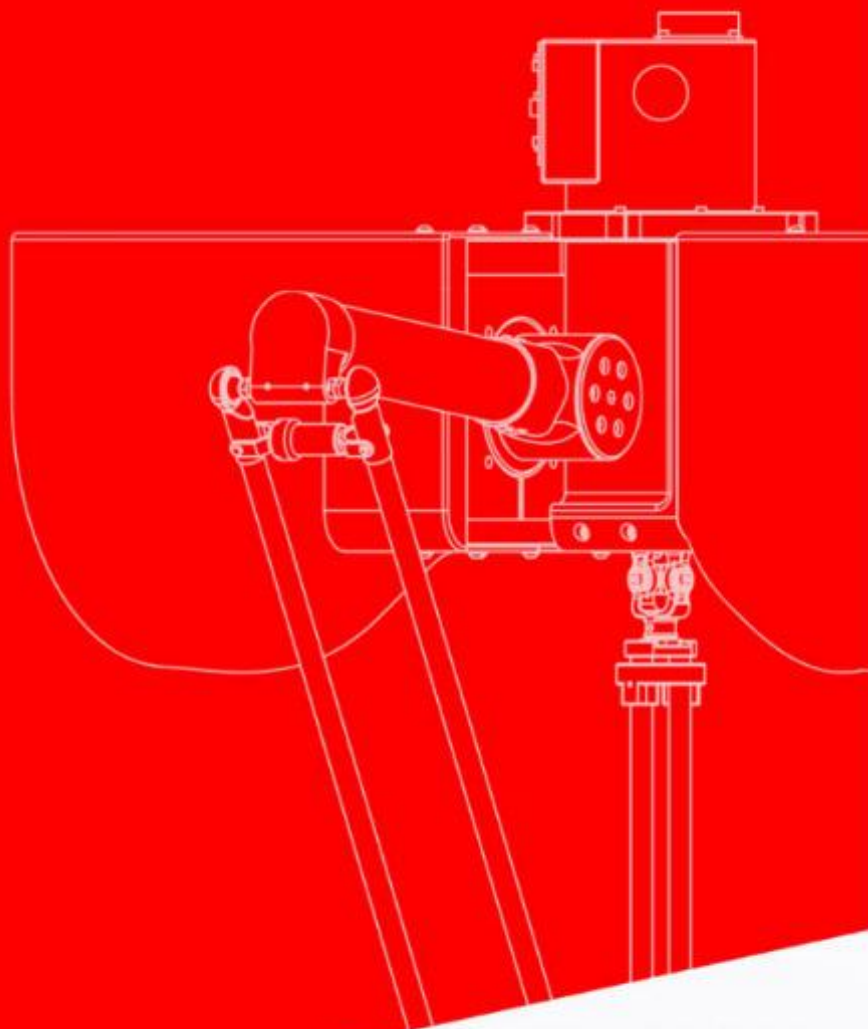


atomrobot®
阿童木机器人

阿童木机器人
并联销量遥遥领先



产品 使用说明书

Copyright 2022 阿童木机器人.保留所有权。

辰星（天津）自动化设备有限公司



Statement

This manual is applicable to atomrobot Delta model robots manufactured by Chenxing (Tianjin) Automation Equipment Co., Ltd.

Specifications and information relating to products in this manual are subject to change without notice. All statements, information and recommendations presented in this manual have been carefully handled, but cannot be guaranteed to be completely correct. The user is completely responsible for his application of any product. No responsibility is assumed for damages caused by this manual.

The interpretation of all contents of this manual belongs to Chenxing (Tianjin) Automation Equipment Co., Ltd.

This manual is not authorized to any party and may not be reproduced or copied in whole or in part in any way.

Copyright: Chenxing (Tianjin) Automation Equipment
Tel: 022-65181003
Service Hotline: 022-65181003
Address: No. 156 Nanhai Road, Binhai New District, Tianjin

Contents

Contents.....	1
Chapter1 Safety Precautions.....	2
1.1 General Safety Precautions.....	2
1.1.1 Robot System.....	2
1.1.2 Security Risks.....	2
1.1.3 Security Behavior.....	5
1.1.4 Emergency Stop.....	8
1.2 Robot Safety Precautions.....	8
1.2.1 Introduction of Security Signs.....	9
1.2.2 Potentially Fatal Danger.....	9
1.2.3 Possible Dangers of Test Operations.....	10
1.2.4 Electrical Dangers.....	10
Chapter2 Login Interface.....	12
Chapter3 Introduction to Main Interface.....	13
3.1 Operation Mode.....	13
3.2 Running Status.....	14
3.3 Single /Continuous Mode.....	14
3.4 Robot Enable.....	14
3.5 Emergency Stop Status.....	15
3.6 Speed Rate.....	15
3.7 Current Loader Name.....	15
3.8 Clear Alarm.....	15
3.9 Jog.....	16
Chapter4 Project Interface.....	17
4.1 Project.....	17
4.2 Program.....	17
4.3 Delete.....	18
4.4 Cancel.....	18
4.5 Open.....	18

4.6 Load.....	18
4.7 Unload.....	18
4.8 Other.....	18
4.8.1 Rename.....	19
4.8.2 Copy.....	19
4.8.3 Paste.....	19
4.8.4 View number.....	19
Chapter5 Program Interface.....	20
5.1 Teach.....	20
5.2 Execute Specified Line.....	21
5.3 New Instruction.....	21
5.3.1 New Expression.....	23
5.3.2 New Instruction.....	25
5.4 View Instruction Help.....	25
5.5 Modify Parameters.....	26
5.6 Edit.....	26
5.6.1 Copy.....	26
5.6.2 Cut.....	27
5.6.3 Paste.....	27
5.6.4 Delete.....	28
5.6.5 Disable / Enable.....	28
5.7 Load.....	28
Chapter6 Variable Interface.....	29
6.1 Teach.....	29
6.2 New.....	30
6.3 Variable Help.....	30
6.4 Edit.....	31
6.4.1 Copy.....	31
6.4.2 Paste.....	31
6.4.3 Rename.....	32

6.4.4 Delete.....	32
6.5 Other.....	32
6.5.1 View Track.....	32
6.5.2 Clear Unused.....	32
6.5.3 Delete Element.....	33
6.5.4 Insert Element.....	34
6.5.5 Expand.....	34
6.5.6 Collapse.....	34
6.6 Variables that the operator can manipulate.....	35
6.7 Load.....	35
6.8 Variable ordering.....	35
6.8.1 Type the sorting.....	35
6.8.2 Alphabetica the sorting.....	35
6.9 Variable Filtering.....	36
6.9.1 Type filtering.....	36
6.9.2 Name filtering.....	36
6.9.3 remove.....	36
6.10 Program storage structure versus variable value.....	36
6.11 Variable number.....	38
6.12 Modify program variables at run time.....	39
6.12.1 HMI controls modifying variables.....	40
6.12.2 TCP control modifies variables.....	40
Chapter7 I/O Monitoring Interface.....	43
7.1 Digital IO.....	43
7.1.1 Set the digital output value.....	43
7.2 Analog IO.....	44
7.2.1 Set the analog output value.....	44
7.3 Other.....	45
7.3.1 Set other output value.....	45
7.4 Custom Name.....	46

7.5 IO that the operator can manipulate.....	46
Chapter8 Conveyor.....	47
8.1 Conveyor Configuration.....	47
8.1.1 Encoder.....	48
8.1.2 Work Area.....	48
8.1.3 Conveyor Coordinate System.....	48
8.2 Teach.....	49
8.2.1 Teaching Method.....	49
8.2.2 Five Point Teach.....	49
8.2.3 Three Point Teach.....	54
8.2.4 Center of Circle Teach.....	54
8.2.5 Static Teach.....	56
8.2.6 Senior Teach.....	57
8.3 Data Buffer.....	60
8.4 Data History.....	61
8.5 Statistics.....	62
Chapter9 Function Block Interface.....	63
9.1 Zero Setting.....	63
9.1.1 Set axis zero.....	63
9.1.2 Set joint zero.....	65
9.2 Filter.....	66
9.2.1 Restore Default Configuration.....	67
9.3 Regional Monitoring.....	67
9.4 PLC Control.....	68
9.5 External Axis.....	69
9.5.1 Jog.....	69
9.5.2 New.....	70
9.5.3 Edit.....	70
9.5.4 External axis motion conditions and stop conditions.....	71
9.5.5 Send.....	71

9.5.6 Instruction execution sequence description.....	71
9.6 TCP Monitor.....	71
9.7 Tracking Parameters.....	73
9.7.1 Configuration Item Description.....	73
9.7.2 Configuration Button Description.....	73
9.8 Object Source Management.....	74
9.8.1 Camera.....	74
9.8.2 Sensor.....	79
9.8.3 Position Change.....	79
9.8.4 Virtual object source.....	79
9.9 Object Allot.....	80
9.10 Status Monitoring.....	82
9.11 Handwheel Function.....	82
9.12 Variable Cache Configuration.....	83
9.13 Yield.....	84
9.14 Drag and teach.....	85
9.14.1 Point teach.....	85
9.14.2 Trajectory teaching.....	86
9.15 Watchdog.....	87
Chapter10 Control Power.....	88
10.1 IO Control.....	88
10.1.1 Input / Output Configuration.....	88
10.1.2 Loader Configuration.....	89
10.2 TCP Control.....	90
10.2.1 TCP Communication Configuration.....	90
10.2.2 TCP Communication Protocol.....	91
10.3 ModbusTcp Control.....	91
10.3.1 Introduction.....	91
10.3.2 Packet Format.....	92
10.3.3 Robot Related Functions.....	96

10.3.4 Shared variable operation.....	102
Chapter11 Alarm Management.....	104
11.1 Current Alarms.....	104
11.2 Historical Alarms.....	104
Chapter12 System Interface.....	106
12.1 User Management Interface.....	106
12.2 Setting.....	106
12.2.1 Shut Down.....	107
12.2.2 Update System.....	107
12.2.3 Modification Date and Time.....	108
12.2.4 Communication IP settings.....	108
12.2.5 Lock Screen.....	108
12.2.6 Controller IP settings.....	108
12.2.7 MacLicense Setting.....	108
12.2.8 Control Power Setting.....	109
12.3 Import/Export.....	110
12.4 Version.....	111
Chapter13 Appendix I Instruction System Introduction.....	112
13.1 Motion Command.....	112
13.1.1 Ptp.....	112
13.1.2 Line.....	113
13.1.3 Circle.....	114
13.1.4 PtpRel.....	115
13.1.5 LineRel.....	116
13.1.6 LineAbs.....	117
13.1.7 ReturnHome.....	118
13.1.8 CustomPath.....	118
13.2 Tracking Function Instruction.....	118
13.2.1 WaitObject.....	118
13.2.2 IsArriveObject.....	119

13.2.3 ObjectDone.....	119
13.2.4 ObjectCancel.....	120
13.2.5 ObjectFinish.....	120
13.2.6 ObjectClear.....	120
13.3 Setting Instruction.....	121
13.3.1 SetDynamic.....	121
13.3.2 SetTransition.....	121
13.3.3 SetAcceleration.....	121
13.3.4 SetCartSys.....	122
13.3.5 SetTool.....	122
13.4 Input / Output Instruction.....	122
13.4.1 SetDout.....	122
13.4.2 SetVDout.....	122
13.5 Trigger Instruction.....	123
13.5.1 OnDistanceDO.....	123
13.5.2 OnPercentDO.....	123
13.6 Wait Instructions.....	124
13.6.1 Wait.....	124
13.6.2 WaitIsFinished.....	124
13.6.3 WaitTime.....	124
13.7 Process Control Instruction.....	125
13.7.1 IF.....	125
13.7.2 ELSIF.....	125
13.7.3 ELSE.....	125
13.7.4 WHILE.....	125
13.8 Assignment Instruction.....	126
13.8.1 :=.....	126
13.9 Monitoring Area Command.....	126
13.9.1 EnableWorkArea.....	126
13.9.2 DisableWorkArea.....	126

13.10 Palletizer instruction.....	127
13.10.1 ResetPalletizer.....	127
13.10.2 NextPalletizer.....	127
13.10.3 SetPalletizerNum.....	127
13.11 PLC Instructions.....	127
13.11.1 StartPLC.....	127
13.11.2 StopPLC.....	128
13.12 Communication Instruction.....	128
13.12.1 SendTcpData.....	128
13.13 Mathematical Operation Instruction.....	128
13.13.1 SIN.....	128
13.13.2 COS.....	128
13.13.3 TAN.....	128
13.13.4 ASIN.....	129
13.13.5 ACOS.....	129
13.13.6 ATAN.....	129
13.13.7 LN.....	129
13.13.8 EXP.....	129
13.13.9 ABS.....	130
13.13.10 SQRT.....	130
13.14 Operator.....	130
13.14.1 +.....	130
13.14.2 -.....	130
13.14.3 *.....	130
13.14.4 /.....	130
13.14.5 AND.....	131
13.14.6 OR.....	131
13.14.7 XOR.....	131
13.14.8 NOT.....	131
13.14.9 <.....	131

13.14.10 >.....	131
13.14.11 <=.....	131
13.14.12 >=.....	132
13.14.13 =.....	132
13.14.14 <>.....	132
13.14.15 ().....	132
13.14.16 [].....	132
13.15 Object Information.....	132
13.15.1 GetObjectId.....	132
13.15.2 GetObjectAttr.....	133
13.15.3 SetObjectAttr.....	133
13.15.4 GetObjectInfo.....	133
13.15.5 GetObjectOriginLocation.....	133
13.16 Conversion Instruction.....	134
13.16.1 IntToString.....	134
13.16.2 RealToString.....	134
13.16.3 BoolToString.....	134
13.16.4 StringToInt.....	134
13.16.5 StringToReal.....	135
13.16.6 StringToBool.....	135
13.17 Other instructions.....	135
13.17.1 GetCacheString.....	135
13.17.2 GetRobotCurrentEndJointPosition.....	136
13.18 Other instructions.....	136
13.18.1 GetCacheString.....	136
13.18.2 CustomAlarm.....	136
13.18.3 LineSearch.....	136
Chapter14 Appendix II Variable Types.....	138
14.1 Motion Variable.....	138
14.1.1 JointPosition.....	138

14.1.2 TcpPosition.....	139
14.1.3 JointDistance.....	140
14.1.4 TcpDistance.....	140
14.1.5 Dynamic.....	141
14.1.6 Transition.....	142
14.1.7 CartSys.....	142
14.1.8 Tool.....	143
14.2 Tracking variable.....	143
14.2.1 TargetObject.....	143
14.2.2 Conveyor.....	144
14.2.3 ObjectSource.....	145
14.2.4 ObjectSort.....	149
14.2.5 OverlapFilter.....	149
14.2.6 ObjectAllot.....	150
14.2.7 ConditionalControl.....	151
14.3 Regional variable.....	152
14.3.1 WorkArea.....	152
14.4 Input and Output Variables.....	153
14.4.1 Din.....	153
14.4.2 Dout.....	153
14.4.3 VDin.....	153
14.4.4 VDout.....	154
14.5 Basic Data Type Variable.....	154
14.5.1 INT.....	154
14.5.2 REAL.....	154
14.5.3 BOOL.....	154
14.5.4 STRING.....	154
14.6 Array variable.....	154
14.6.1 ARRAY OF INT.....	154
14.6.2 ArrayOfTcpPosition.....	155

14.7 Palletizer Variable.....	155
14.7.1 Palletizer.....	155
14.7.2 PalletizerData.....	157
14.7.3 SeniorPalletizer.....	159
14.8 Communication Variable.....	160
14.8.1 TcpConnect.....	160
14.8.2 HardTrigger.....	160
14.8.3 Alarm.....	161
14.8.4 AxisLimit.....	162
Chapter15 Appendix III Alarm Information.....	164
15.1 Serious Error.....	164
15.2 Error.....	166
15.3 Warning.....	171
Chapter16 Appendix IV Application example.....	173
16.1 Palletizing.....	173
16.2 SeniorPalletizer.....	174
16.2.1 SeniorPalletizer variables created.....	175
16.2.2 Example Configuration Description.....	175
16.2.3 Fast configuration.....	184
16.2.4 SeniorPalletizer example procedures.....	186
16.3 Production.....	186
16.4 Variable buffer area (realize the function of modifying variables during operation).....	187
16.5 Hard trigger.....	189
16.6 Area monitoring.....	192
16.7 Object Allot.....	195
16.8 External axis.....	197

Introduction

First of all, thank you for using Atomrobot:

This manual contains: Safety precautions, AtomPad operation instructions, Introduction to the command system, Alarm Information, Applications.

About this manual:

This manual is intended for manufacturers of robots that use this model, including those who install, adjust, and repair the robot. Anyone installing, adjusting, using, or repairing this robot must be trained by Chenxing (Tianjin) Automation Equipment Co., Ltd. and have carefully read this manual before doing any activities on the robot.

Chapter1 Safety Precautions

The safety precautions in this chapter are divided into two parts.

The first section contains general safety precautions that are generally applicable to all types of robots, as described in 1.2 General safety precautions.

The second part is about safety precautions for robots, which mainly introduces safety precautions about robot operation and usage, as described in 1.3 Robot Safety Precautions.

1.1 General Safety Precautions

1.1.1 Robot System

This section does not cover how to design and install the robot, nor does it cover peripheral equipment that affect the safety of the robot. For the protection of those using it, the robot should be designed to comply with the standards and laws of the region and country in which it is located.

Companies and individuals using Atomrobot should familiarize themselves with the standards and laws of their region and country, and install appropriate safety features to protect the robot's users. Users should be familiar with the instructions for using the robot system. However, even if the operator follows all the safety information given in the manual exactly, Atomrobot cannot guarantee that the operator will not be harmed in any way.

1.1.2 Security Risks

1.1.2.1 Introduction

This section contains information about the dangers that can occur in the installation and service work of the robot.

Security risks when installing, servicing robots.

1. Safety precautions regarding the robot are detailed in the Installation and Maintenance section.
2. The emergency stop button of the system must be in the easy to touch location, in case of accidents can stop the robot emergency.
3. The operator must be sure that the installed safety measures are available.
4. The operator must be trained to install and operate the maintenance robot.
5. The specifications of atomrobot's robot must also comply with the standards and legal rules of the region and country.

Non-voltage risks.

1. Safety area need to be identified prior to robot installation and need to be delineated prior to robot installation.
2. Protective measures or fences are needed to keep the operator outside the robot's working area (signs such as " Staff Only ", " Restricted Area ", "High Voltage Danger", etc. are placed in the relevant area).
3. No hanging objects above the robot to prevent dropping and damaging the robot and other equipment.
4. When disassembling the robot, be aware of objects that may fall from the robot and injure people.
5. Beware of burns from hot components in the electric control cabinet.
6. It is forbidden to use the robot as a ladder when servicing it and not to climb on it to prevent falling.
7. High temperatures in the reducer and splashing of high-temperature fluids can cause human injury.
8. It is strictly forbidden to move the shaft of the robot.
9. It is strictly forbidden to lean on the electric control cabinet or touch the buttons at will to prevent the robot from unpredictable action that may cause personal injury or equipment damage.

Integrator Security Considerations.

1. The integration supplier must confirm that all safety circuits are interlocked with the safety circuits of the external application.
2. The integration supplier must confirm that the emergency stop safety circuit is interlocked with the external application safety circuit.

Integrated Robotics

Security Risks	Specific description
High temperature components	Servo motors and Reducers generate high temperatures after long periods of operation, and touching these parts can easily result in burns. Robots warm up faster in high-temperature environments, and burns are more likely to occur.
Removing certain parts can easily cause the robot to collapse	Take measures to ensure that the robot does not collapse when certain parts are removed (e.g. when removing 1, 2 or 3 shafts motors, the active and driven arms need to be secured to prevent the robot from falling over)

Cable

Security Risks	Specific description
The cable is fragile and easily damaged	The cable is easily damaged by machinery, so special care should be taken when transporting, storing and using the cable to damage it, especially the connector part.

Motors Reducers

Security Risks	Specific description
The reducer is easily damaged by improper external forces	No matter when disassembling a motor reducer, or when installing a motor reducer. The reducer is prone to damage under excessive and improper external forces.

1.1.2.2 Pay Attention to the Parts of the Robot that tend to Heat Up

Many parts of the robot get hot during normal operation, especially the servo motor and reducer parts, and sometimes the parts near these two parts get hot, so touching these parts can cause burns. As the environmental temperature becomes higher, more of the robot's surface will become very hot and easy to cause burns.

Security measures:

1. Touch these parts with your hands before feeling the temperature of these parts with your hands close to these parts to prevent burns.
2. Wait enough time after a shutdown for the hot part to cool down before performing work such as repairs.

1.1.2.3 Safety Precautions Regarding Removal of Parts

Safe operation: Open the cover or protection device after confirming that the gears and other internal parts are no longer rotating or moving, and do not open the protection device while the gears, bearings, etc. are rotating.

Safety design: If necessary, use auxiliary devices to keep the internal parts that are no longer fixed from their original location.

1.1.2.4 Safety Precautions Regarding Pneumatics/Hydraulics

Introduction: This paragraph is about the safety of the pneumatic/hydraulic system accompanying the robot

Residual energy: After shutting down the air supply or hydraulic pump, there are residual gases/fluids in the pneumatic/hydraulic system that gas/ liquid has a certain amount of energy, to

take certain measures to prevent residual energy to cause harm to humans and equipment, in the maintenance of air pressure and hydraulic components before the residual energy in the system needs to be released.

Safety design: to prevent components from falling and hydraulic oil from flowing.

The need to install safety valves in case of accidents.

Need to prevent maintenance tools from dropping.

1.1.2.5 Risks during Operation

Introduction: Industrial robots are flexible systems that can be used in many industrial applications. All work must be operated by professionals and follow certain safety guidelines. Care must be taken at all times when operating.

Highly qualified operators: Industrial robots must be operated by professionals who are familiar with the entire system and understand the risks involved in each subsystem.

Risk of abnormal: If an abnormality occurs under normal working procedures, special care should be taken at this point.

1.1.2.6 Electrical Risks

Although in many cases it is necessary to turn on the power during troubleshooting. However, when actually servicing the robot, it is necessary to turn off the power and disconnect other power connections.

The robot's main power supply needs to be installed outside the robot's working range so that the operator can shut down the robot outside the robot's working range even if the robot goes out of control.

High voltage dangers that operators need to be aware of:

1. Power lines for servo motors.
2. Electrical dangers of power cables connecting fixtures and other devices. Fixtures, external devices, etc. It is possible that the robot's external devices may still be running after the robot is shut down, so the power cables of external devices can also be personally injured or the power cables damaged.

1.1.3 Security Behavior

1.1.3.1 Safety Measures

Fences and warning signs need to be installed around the robot's working area to ensure that the robot works safely, to prevent the entry of unoccupied people and to prevent the robot from injuring people.

Safety measures: Setting up safety measures requires considering that the workpiece held by the robot will cause injury to personnel if it is thrown off.

1.1.3.2 Fire Danger



Caution: A carbon dioxide extinguisher needs to be placed on site in case the robot system catches fire.

1.1.3.3 Emergency Disassembly Robot Arm

Description: In an emergency situation, any of the robot's arms get caught on the operator and need to be removed. (See Chapter 5, Maintenance for details on removal steps.) Smaller robot arms can be removed manually, but larger robots require the use of a crane or other equipment.

Secondary injury: Before releasing the joint holding brake, the mechanical arm needs to be fixed to ensure that the mechanical arm will not cause secondary injury to the trapped person under the effect of gravity.

1.1.3.4 Holding Brake Detection

Why testing: In normal operation, the holding brake usually wears out, which requires testing of the holding brake.

Check the holding brake steps:

1. Move the robot joints to the location where the joints are subjected to the maximum load.
2. Closing the robot so that the holding brake opens.
3. Marking of the joints.
4. See if the robot joints move after a while.

1.1.3.5 Safe Use of Teach



Caution: When enabled a PWR button on the oscillator, when pressed, servo motor on enable; when disconnected, servo motor off enable.

To ensure the safe use of the Teach, the following rules need to be followed.

1. The enable button must not be deactivated at any time.
2. Enable the need for timely disconnection during programming or testing.
3. Teachers are required to take the Teach with them when they enter the robot's work area so that others cannot move the robot without the programmer's knowledge.

Turn off enable promptly whenever the robot is temporarily stopped or when programming or testing.

1.1.3.6 Work within the Working Range of the Robot



DANGER: If work must be performed within the robot's working range, the following rules need to be followed

1. Mode is selected as manual mode before it can be on enable and disconnect other automatic controls such as computer control.
2. When the robot is in manual mode, the speed must be limited to 250 mm/s or less.
3. When the robot needs to be set to manual full speed, it should only be operated by professionals who have a full understanding of the risks.
4. Pay attention to the rotating joints of the robot to prevent hair and clothes from being caught in the joints.
5. Also be aware of other danger that may be caused by the movement of the robot, or other supplementary equipment.
6. Testing the motor holding brake for proper operation to prevent personal injury caused by abnormalities in the robot.
7. Consider contingency plans in the event that the robot suddenly moves toward the orientation it is in; ensure that hiding places are set up just in case.



DANGER: Do not stand under any robot arm under any circumstances in case the robot moves abnormalities or someone else moves to enable it.

1.1.4 Emergency Stop

Definition of Emergency Stop.

Emergency stop is independent of all robot electrical controls and can stop all robot movements.

An emergency stop means that all the power connected to the robot is disconnected, but the power to the holding brake on the servo motor is not. You must release the emergency stop button and turn the robot back on so that the robot can operate again.

Emergency stops in robotic systems need to be distinguished from.

1. A uncontrolled emergency stop that stops the robot by cutting power to all servo motors.
2. A controlled emergency stop, by giving the servo motor command to stop the robot, that way the robot can finish the path, when finished the path, the servo motor stop power supply.



Caution: The emergency stop can only be used when it is really needed and is really an emergency.



Caution: The emergency stop cannot be used for usual program stops, shutting down the robot, etc.

Emergency stop button


There are several emergency stop buttons in the robot system that can be used to stop the robot in an emergency. There is a red button on both the teach and the electrical control cabinet (shown below). Of course, users can also set their own emergency stop buttons as needed.




1.2 Robot Safety Precautions

Before using the machine (installation, operation, maintenance and service), please make sure to read and master this manual and other supplementary information, and start using the machine after you are familiar with all the equipment knowledge, safety knowledge and precautions. The safety precautions in this manual are divided into "Danger", "Caution", "Mandatory" and "Prohibited". The four categories are recorded separately.


1.2.1 Introduction of Security Signs

	Danger
There is a danger of serious injury or even death in case of misoperation.	

	Caution
Dangerous in case of misoperation, moderate injury, minor accident or object damage may occur.	

	Mandatory
Matters for which compliance is mandatory.	

	Prohibition
Absolutely prohibited matters.	

	Important
In order to ensure safe and effective operation, the user must comply with the matters that will be described in the relevant place.	

1.2.2 Potentially Fatal Danger

Introduction: Any running robot is a potentially deadly machine. When in operation, the robot may have unpredictable movements, and all movements have strong forces that may cause serious injury to people within the work area or damage to equipment.

Avoidance: Before preparing the robot for work, test the reliability of each safety measure (holding brake).

Safety measures include safety gates, holding brake, and safety lights.

Avoidance measures: Before turning on the robot, make sure there are staff only within the robot's working range.

1.2.3 Possible Dangers of Test Operations

Introduction: Because the robot needs to be disassembled for repair service work, there are several risks to consider for the first test job after the repair work is completed.

Measure: The following steps need to be followed for the first test after repair, installation, maintenance and other services.

1. Cleaning of all maintenance and installation tools on the robot and within the working area of the robot.
2. Installation of all security measures.
3. Ensure that everyone stands outside the safety of the robot.
4. When testing, special attention should be paid to the working condition of the repaired parts.



Caution: When the robot go through the program, pay special attention to potential interference dangers.

1.2.4 Electrical Dangers

Introduction: The electrical control cabinet is the hub of the control robot. Any misuse of the cabinet may result in electric shock and robot malfunction, which in turn may cause harm to people and equipment.

Dangers:

1. Never lean on an electrical or other control cabinet.
2. Do not press the operation keys randomly. Failure to do so may result in unanticipated movements of the robot, which may cause personal injury and equipment damage.
3. During operation, never allow non-staff to touch the electrical control cabinet. Failure to do so may cause the robot to produce unanticipated movements, which may result in personal injury and equipment damage.
4. Protective measures must be taken when wiring and piping between the electronic control cabinet, robot and peripheral equipment, such as pipe, line or cable through the pit or adding a protective cover to cover it so that it will not be stepped on by people or run over by forklifts.

Operators and other personnel may trip over lines, cables or piping and damage them, which may cause abnormal robot movements and result in personal injury or equipment damage.

5. When installing a tool on the robot, be sure to disconnect the power to the control cabinet and the tool, lock the power switch and hang a warning sign. If the power is turned on during the installation process, it may cause electric shock, or it may cause abnormal robot movement, which may result in injury.

6. Before operating the robot, press the emergency stop button on the front door of the control cabinet and the upper right side of the programmer to check if the "servo ready" indicator is off and to confirm that the power is off.

Chapter2 Login Interface

Select user → Enter password → Enter the system.

Users are divided into administrator, adjuster and operator according to their permissions. Administrator have the highest permissions and operator have the lowest permissions. The default password of the administrator is m, and the user password can be set by himself. Please refer to the user management interface for password setting.

User

Level
(0-highest authority; 1-medium authority; 2-lowest authority)

Password

-	1	2	3	4	5	6	7	8	9	0
.	q	w	e	r	t	y	u	i	o	p
_	Caps	a	s	d	f	g	h	j	k	l
:	z	x	c	v	b	n	m	X	Del	Ent

Figure 3.1 login interface

Chapter3 Introduction to Main Interface

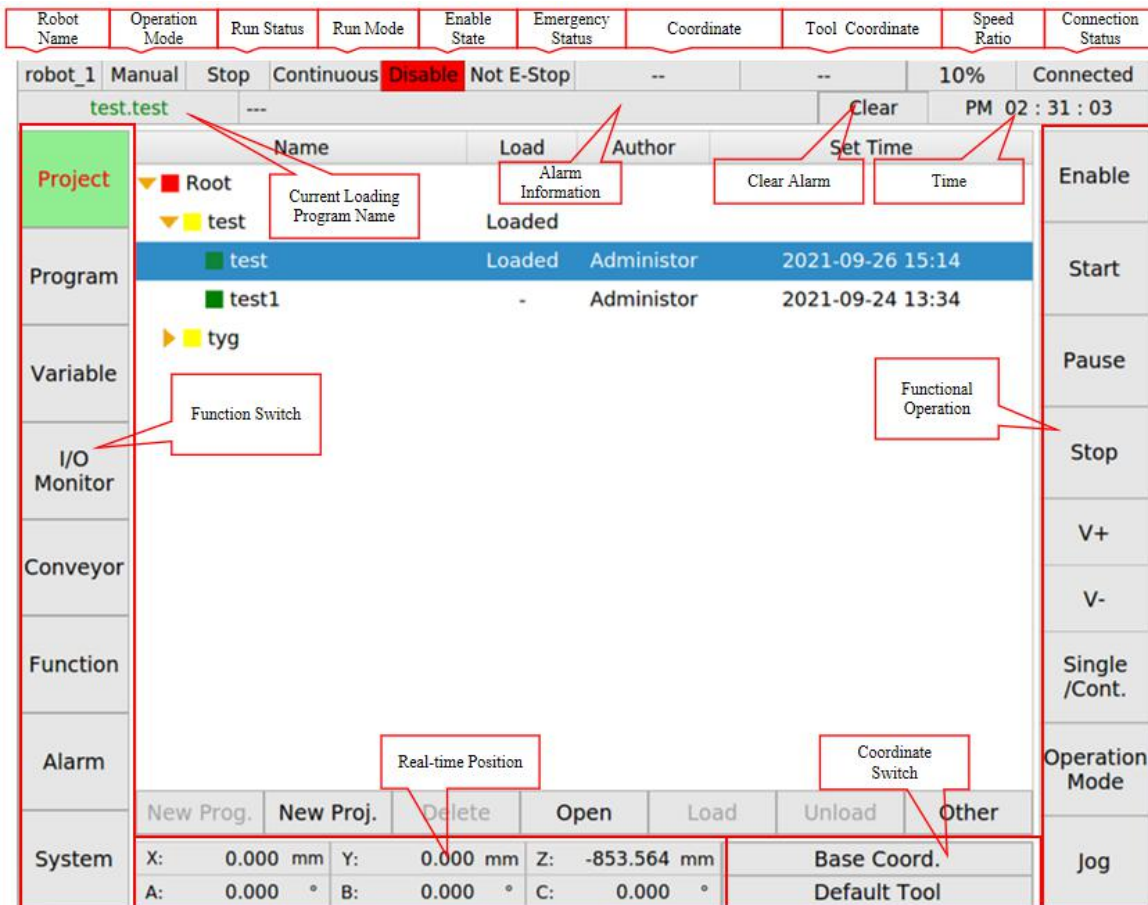


Figure 3.1 introduction to main interface

3.1 Operation Mode

Type: Manual mode, Auto mode.

Click the "operation mode" button to select the operation mode, as shown in the following figure

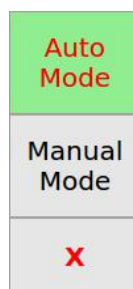


Figure 3.2 operation mode interface

3.2 Running Status

Type: Operation status, stop status and pause status.

Pause: Click the start button after pause, and the program will continue to run from the suspended location.

Stop: After stopping, set the current execution line of the program to the first line, and click the "start" button to start the program from the first line.

3.3 Single /Continuous Mode

Type: Single step mode, continuous mode.

Click the "Single/Cont." button to select the operation mode, as shown in the following figure



Figure 3.3 Single /Continuous Mode interface

3.4 Robot Enable

Enable state

Type: Enable state and non enable state.

Enable: The enable status displays "enable" and the background color is green.

Non enable: The enable status displays "non enable", and the background color is red.

Note: the robot can only run when it is enabled.

Enable operation

Manual mode: In manual mode, press the Yellow three-stage switch on the back of the teaching pendant to enable the robot shaft, release the switch to enable the robot shaft, or press the switch to the third stage to enable the robot shaft.

Automatic mode: In automatic mode, click the "enable" button in the upper right corner of the interface to enable the robot shaft, and click the "enable" button again to enable the robot shaft.

Note: the three-stage switch can be enabled only when it is in the second stage.

3.5 Emergency Stop Status

Type: Emergency stop status, non emergency stop status.

Emergency stop: The emergency stop status displays "E-Stop", and the background color is red.

Non emergency stop: The emergency stop status displays "Non E-Stop".

Note: the robot cannot run under the emergency stop state. In case of danger, please press the "emergency stop" button in time.

3.6 Speed Rate

Speed magnification range: 0-100%. When the dynamic parameters do not change, increasing the speed rate can improve the running speed of the manipulator.

Operation: Click the "Speed Rate" button to pop up the interface as shown in the figure below. There are five gears to select: 1%, 10%, 50%, 80% and 100%.

1%
10%
50%
80%
100%

Figure 3.4 speed multiplier gear

Note: V + / V - will have a certain time delay during the operation of the program.

3.7 Current Loader Name

Display format: Project name. Program name.

3.8 Clear Alarm

The "Clear" alarm button can clear some alarms.

When the alarm cannot be cleared, a dialog box that cannot clear the alarm will pop up.

3.9 Jog

Click the "Jog" button to pop up the jog operation interface as shown in the figure below.



Figure 3.5 joint jog



Figure 3.6 coordinate system jogging

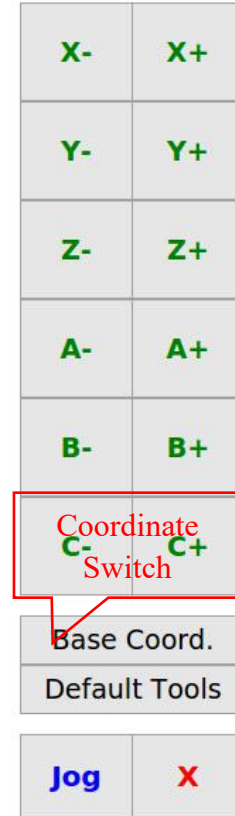


Figure 3.7 shaft jogging

"Jog" button can switch the jog mode: Joint jog (J1, J2, J3...), coordinate system jog (X, Y, Z, A, B, C) and shaft jog (D1, D2, D3...). When the coordinate system jog, the bottom coordinate button can switch the jog coordinate system, such as base coordinate system, world coordinate system and other user-defined coordinate systems. The user-defined coordinate systems include: the coordinate system of the current loader and the global coordinate system.

Pattern	Use Method
Manual Single Step	<p>When Joint jog: Click the button and the location changes by 0.1 degrees.</p> <p>When jog in the world coordinate system: Click the button once, and the location changes by 0.1mm.</p> <p>Shaft jog: Click the button and the location changes by 0.1 degrees.</p>
Manual Continuous	Press and hold the button to action, release the button to stop the movement.

Note: jog is only allowed in manual mode.

Chapter4 Project Interface

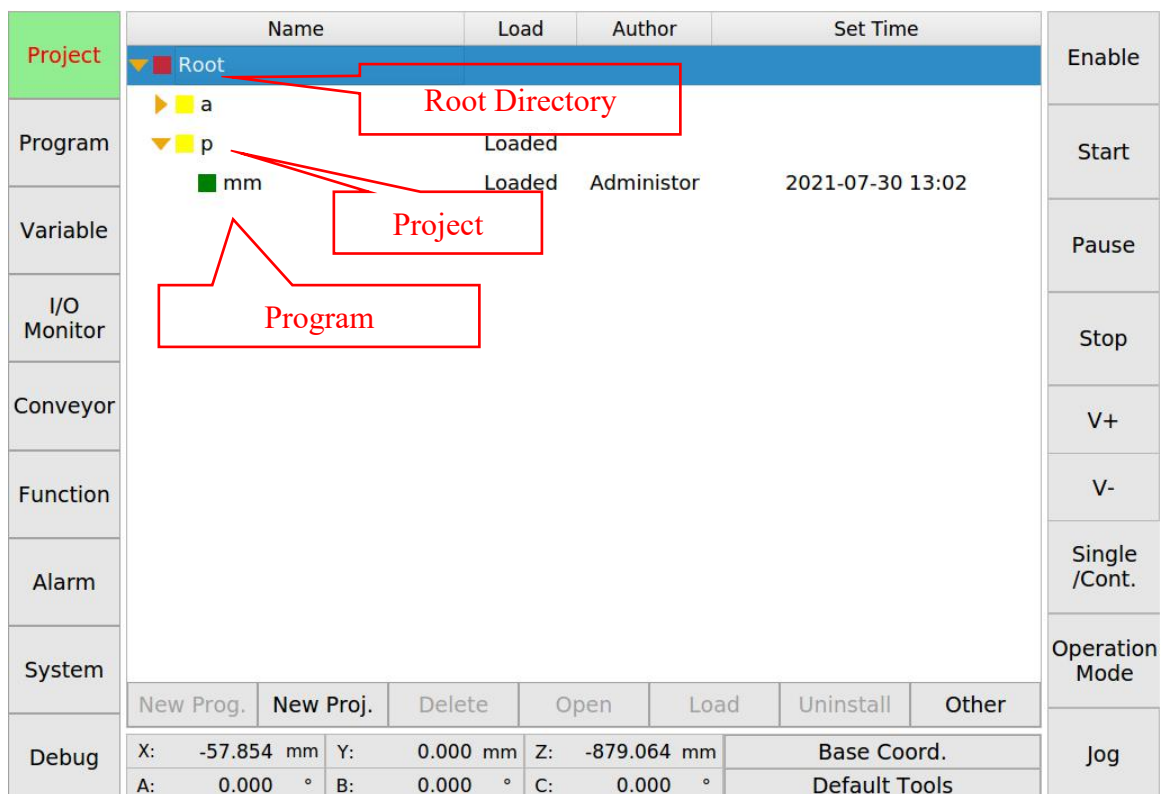


Figure 4.1 project interface

4.1 Project

Projects are used to manage programs and project variables. Each project can contain multiple programs and multiple project variables.

4.2 Program

Write control logic.

Pattern	Usage Method
Automatic Single	Click the "start" button to execute an instruction
Automatic Continuous	Click the "start" button to switch to the stop state after executing all instructions
Manual Single	Press and hold the "start" button to run the program. Release the button to switch to the pause state. If the execution of the last line of instructions of the program is completed when the button is released, switch to the stopped state

	Note: pressing and holding the "start" button all the time can only execute one instruction. You must release the "start" button and press and hold the "start" button again to execute the next instruction
Manual Continuous	Press and hold the "start" button to execute the program, and switch to the stop state after executing all instructions without releasing the "start" button

4.3 Delete

Delete project or program.

Loaded projects cannot be deleted; The programs under the loaded project cannot be deleted; Low privilege users cannot delete programs created by high privilege users.

4.4 Cancel

Cancels an item or program. If the program is deleted, the program can be cancelled. If an item is deleted, the item can be cancelled.

- Deleted items/programs can only be undone one step. If the project is deleted, the deleted project is automatically selected. If the program is deleted, the deleted program is automatically selected.
- If the name of the deleted project/program is the same as that of the project/program on the interface, the deleted project/program is automatically renamed and _new is added to the name of the deleted project/program.
- The program under the project is irretrievably loaded. It can be undone only after the project is uninstalled.
- After a program is deleted, if the program is renamed, the program is irrevocable.

4.5 Open

Open the program, you can edit the program, but you can't run the program.

4.6 Load

Loading project and program, loading project is to send the program under the project to the controller, and the controller executes the program in the loading project; The loaded program can be edited and executed, and only the loaded program can run.

4.7 Unload

Unload the project.

4.8 Other

Including rename, copy, paste, program wizard, view number.

4.8.1 Rename

Rename projects and programs. The loaded project and all programs under it cannot be renamed; Low privilege users cannot rename programs created by high privilege users.

4.8.2 Copy

Copy projects and programs.

4.8.3 Paste

Paste and copy the program; Paste the copied project.

4.8.4 View number

Project Numbering Rules:

The numbering starts from 1; when a new project is created, the numbering of the project is the maximum number of all current projects + 1.

For example: all existing 7 projects, the number of the new project is 8.

Program Numbering Rules:

Numbering starts from 1;

Take the project as the unit, that is, the program number is unique in the project, any project can have a program numbered 1, but there can only be one program numbered 1 in a project;

When creating a new program, the program number is the maximum value of all program numbers in the project to which the program belongs + 1.

View numbering operations:

Select Root to view the numbers of all projects; select an item to view the numbers of all programs in the project; select a program to view the numbers of the current program.

Chapter5 Program Interface

The program interface displays the currently loaded program, as shown in the figure. Line 8 is the currently executed line (the background color is green), and line 10 is the currently selected line (the background color is blue).

```

1  SetDynamic(_g.default_dyn);
2  SetCartSys(_g.default_world_cart_sys);
3  SetTransition(_g.default_transition);
4  SetAcceleration(PARABOLA_ACC);
5  SetTool(_g.default_tool);
6  ▼WHILE TRUE DO
7      Line(pos_1_h, dyn_max, trans_50);
8      Line(pos_1_l, dyn_max, trans_0);
9      Line(pos_1_h, dyn_max, trans_50);
10     Line(pos_2_h, dyn_max, trans_50);
11     Line(pos_2_l, dyn_max, trans_0);
12     Line(pos_2_h, dyn_max, trans_50);
13 END_WHILE;
14 >>>>>EOF<<<<<

```

Teach Set Line New Modify Parameter Edit Load

Figure 5.1 program interface

5.1 Teach

Teach variables or member variables that can be teach in the instruction (Please refer to teach section).

Operation steps

Select the instruction containing motion variables of JointPosition type, and click the "teach" button to pop up a dialog box. Select "Confirm" to teach, and select "Cancel" not to teach.

Select the instruction containing TcpPosition type variable or member variable, and click the "teach" button to pop up a dialog box, as shown in Figure 5.2. Select the reference coordinate system, select "Confirm" to teach, and select "Cancel" not to teach.

```
1 SetDynamic(_g.default_dyn);
2 SetCartSys(_g.default_world_cart_sys);
3 SetTransition(_g.default_transition);
4 SetAcceleration(PARABOLA_ACC);
5 SetTool(_g.default_tool);
6 WHILE TRUE DO
7   Line(pos_1_h,
8     Line(pos_1_l,
9       Line(pos_1_h,
10        Line(pos_2_h,
11         Line(pos_2_l,
12          Line(pos_2_h,
13        END_WHILE;
14 >>>>EOF<<<<<
```

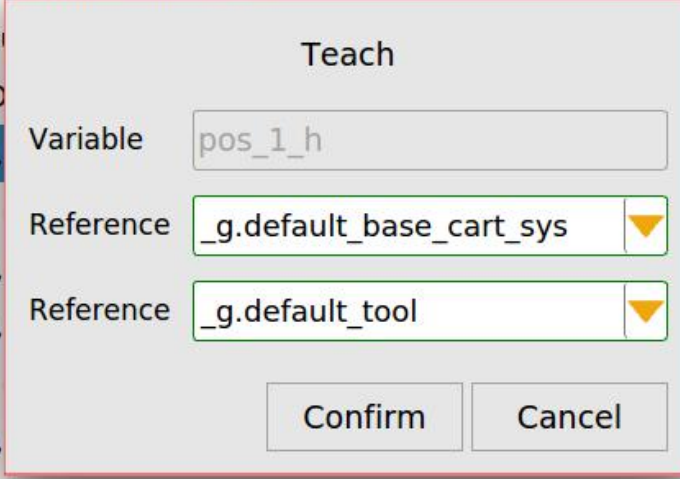
A dialog box titled "Teach" is overlaid on the code editor. It contains three input fields: "Variable" with the text "pos_1_h", "Reference" with a dropdown menu showing "_g.default_base_cart_sys", and another "Reference" with a dropdown menu showing "_g.default_tool". At the bottom of the dialog are two buttons: "Confirm" and "Cancel".

Figure 5.2 program teaching

5.2 Execute Specified Line

Sets the currently selected line as the current execution line.

Operation steps

Select the instruction to be executed and click the "set line" button. When you click the "start" button, the program starts from the set line.

5.3 New Instruction

New instructions are divided into new expressions and new instructions. The new expression method is for the following instructions: OnDistanceDO, OnPercentDO, ELSEIF, ELSE, WHILE and: =, etc.

At present, all instructions are displayed in three pages, as shown in Figure 5.3, Figure 5.4 and Figure 5.5.

Instruction Type				
Motion Ins.	Ptp	Line	Circle	PtpRel
	LineRel	LineAbs	ReturnHome	CustomPath
Track Ins.	WaitObject	IsArriveObject	ObjectDone	ObjectCancel
	ObjectFinish	ObjectClear		
Setting Ins.	SetDynamic	SetTransition	SetAcceleration	SetCartSys
	SetTool			
IO	SetDout	SetVDout	SetAout	SetVAout
	SetAoutValue	SetVAoutValue		
Wait Ins.	Wait	WaitIsFinished	WaitTime	
Process Ins.	IF	ELSIF	ELSE	WHILE
	LOOP			
				Description
<		Page 1	>	Confirm Cancel

Figure 5.3 new instruction 1

Instruction Type				
Math. Operator	SIN	COS	TAN	ASIN
	ACOS	ATAN	LN	EXP
	ABS	SQRT		
Assign Ins.	:=			
Trigger Ins.	OnDistanceDO	OnPercentDO		
Workarea Ins.	EnableWorkArea	DisableWorkArea		
Palletizer Ins.	ResetPalletizer	NextPalletizer	SetPalletizerNum	
Object Info.	GetObjectId	GetObjectAttr	GetObjOriPos	GetObjectParam
	SetObjectAttr	SetObjectParam	GetObjectInfo	
				Description
<		Page 2	>	Confirm Cancel

Figure 5.4 new instruction 2

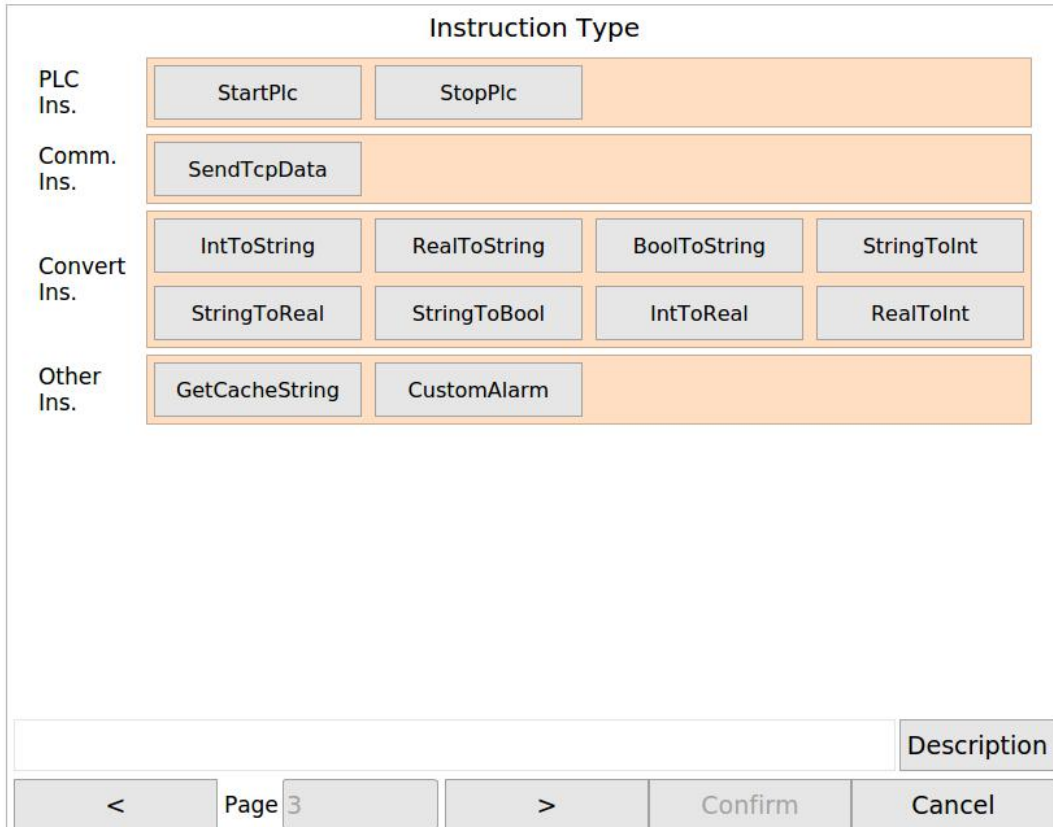


Figure 5.5 new instruction 3

5.3.1 New Expression

Operation steps

- (1) Click the "new" button and select the instruction type (taking: = instruction as an example), as shown in Figure 5.4;
- (2) Click "confirm" to enter the instruction editing page, as shown in Figure 5.6;
- (3) Click the "change parameter" button, and there are several options such as variable, instruction, number and string, as shown in Figure 5.7;
- (4) Select a variable, switch to the variable page, select an existing variable or a new variable, and click "Confirm";
- (5) Return to the edit page, modify other parameters, and click "Confirm" after completing the instruction parameters.

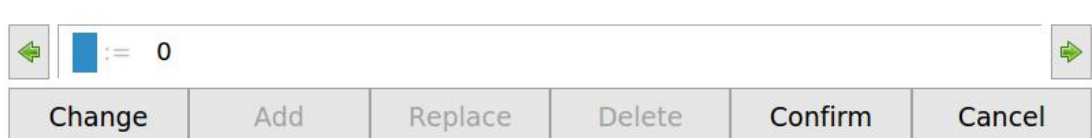


Figure 5.6 expression editing interface

Edit button instructions

Change parameters: There are variables, instructions, numbers and string selections, as shown in Figure 5.7;

Variable: Jump to the variables page and select existing variables or new variables as parameters;

Instruction: Jump to the instruction page and select the instruction as the parameter;

Number: Pop up the numeric keypad and input numbers as parameters;

Character string: Pop up the numeric keypad and enter the string as the parameter;

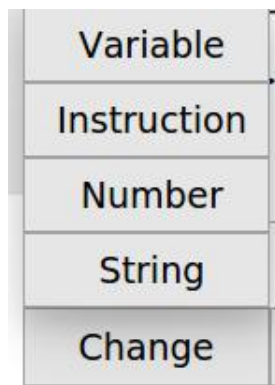


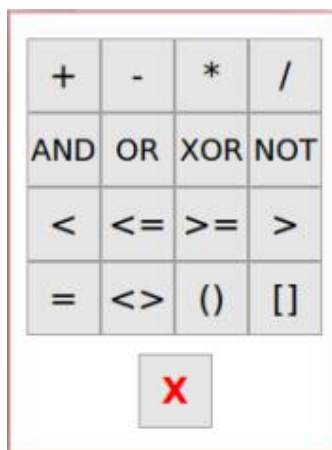
Figure 5.7 parameter change interface

Delete: Delete parameters and operators.

New operator: Insert the operator, as shown in Figure 5.8.

Replace operator: Change the operators in the expression, and the operators can only be switched at the same priority. The replaceable operators are as follows:

+And – replaceable; *And / replaceable; <, <=, >=And > replaceable; =And <> replaceable;



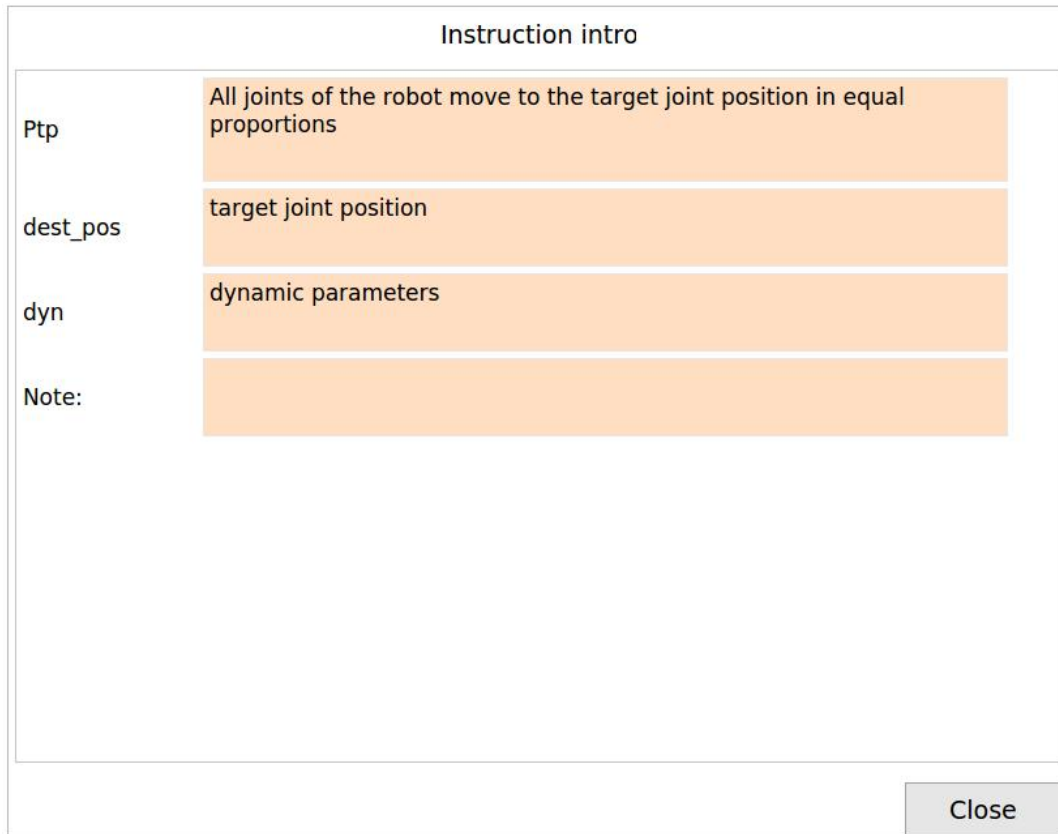


Figure 5.10 instruction introduction interface

5.5 Modify Parameters

Modify the variable or variable value referenced in the instruction.

Operation steps: Click the "modify parameter" button to pop up the instruction edit interface to modify the variables or expressions in the instruction.

5.6 Edit

5.6.1 Copy

Copies one or more selected instructions. Support copy within the same program and cross-program copy under the same project.

Note: when copying, only instructions with complete instruction structure can be copied; Such as if and end_If is a complete structure, only if is incomplete. As shown in Figure , if 7 lines is selected, it will prompt "the currently selected instruction is not a complete instruction".

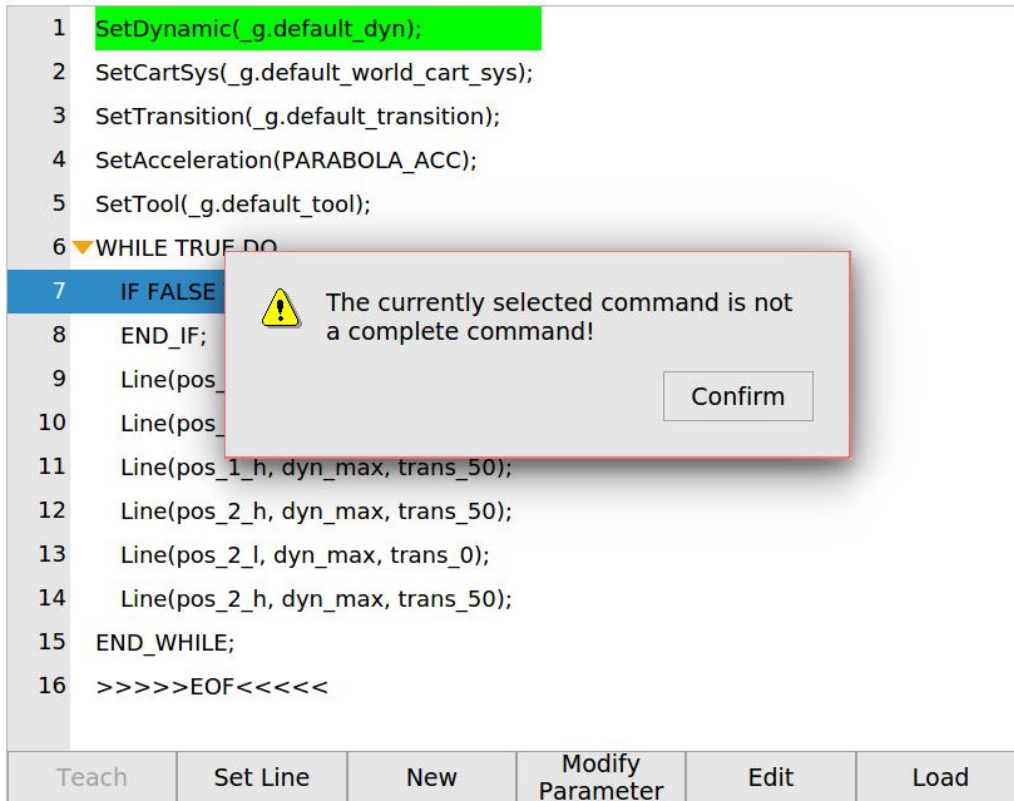


Figure 5.11 copy description interface

5.6.2 Cut

Cut one or more selected instructions. The precautions are the same as copying. Support cut within the same program and cross-program cut under the same project.

5.6.3 Paste

Paste the copied or cut instruction. The pasted instruction is inserted above the currently selected line. Incomplete instructions cannot be pasted. As shown in Figure , if you copy the instruction in line 10-13 and select the instruction paste in line 6, you will be prompted with "instruction logic is incomplete".

When copying instructions across programs, the program variables in the instructions need to be copied at the same time. The naming rules of program variables refer to the renaming of referenced variables during pasting.

Chapter6 Variable Interface

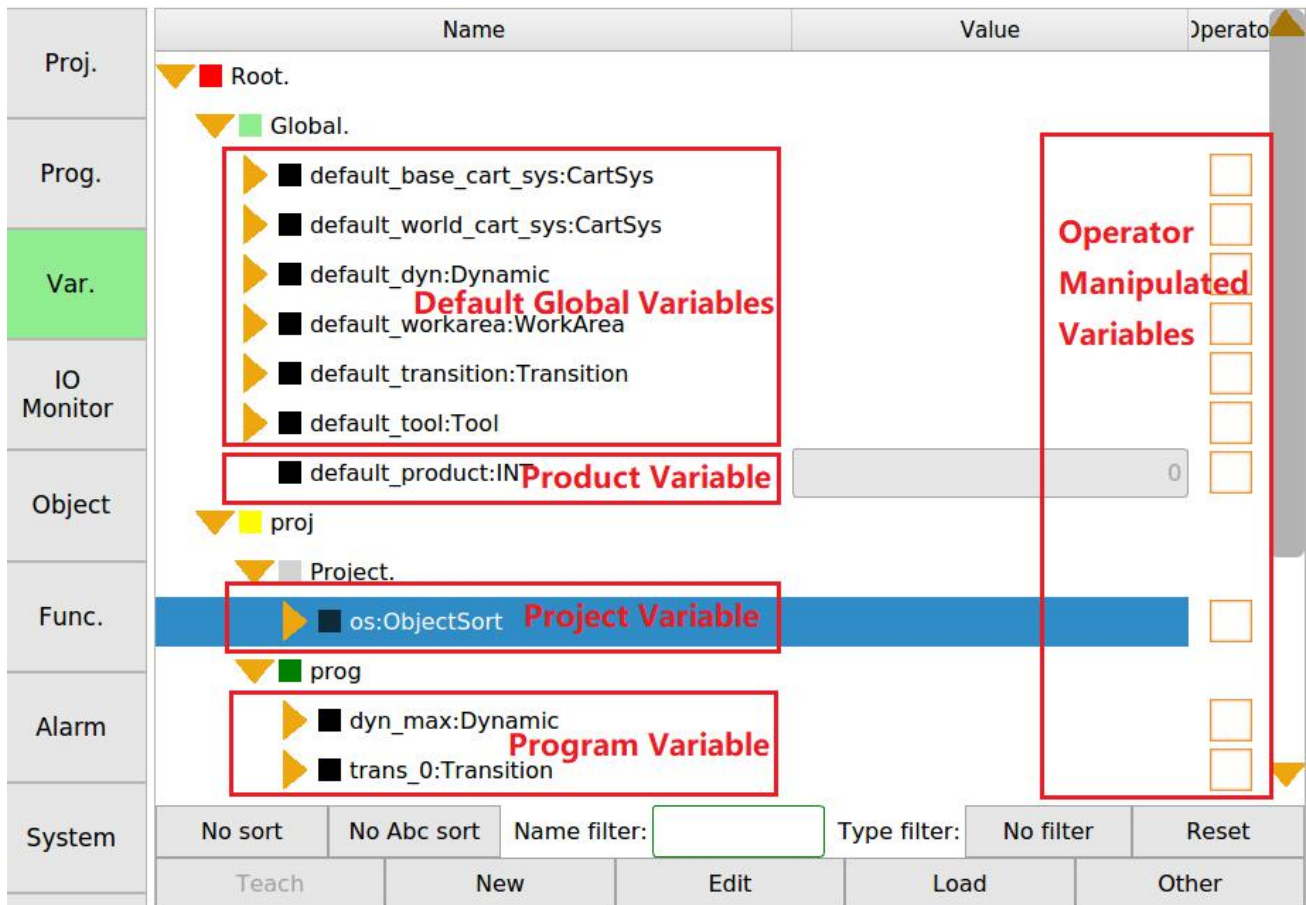


Figure 6.1 variable interface

Variable categories: global variables, project variables, program variables, global variables and 6 default global variables and one output variable as shown in Figure 6.1 Variable interface, the default global variables can only be used and cannot be changed.

Variable name format:

Variable custom name: variable type. Such as default_base_cart_sys: CartRefSys.

default_base_cart_sys: base coordinate system, the origin of the base coordinate system is at the center of the static platform.

default_world_cart_sys: The world coordinate system, which coincides with the base coordinate system.

6.1 Teach

Teach variable or teach variable member. Variable types that can be taught include cartsys, JointPosition, TcpPosition, these types can also be taught as members of other variables.

Such as cartsys variable; Cart_Sys in conveyor variable; Each element in the arrayofTcpPosition array can be taught.

Notes: when teaching TcpPosition type variable or member variable, you need specify the coordinate system of the teaching reference.

6.2 New

Operation steps

- (1) Click "new" to enter the variable type interface, as shown in Figure 6.2;
- (2) Select the variable and click “Confirm”.

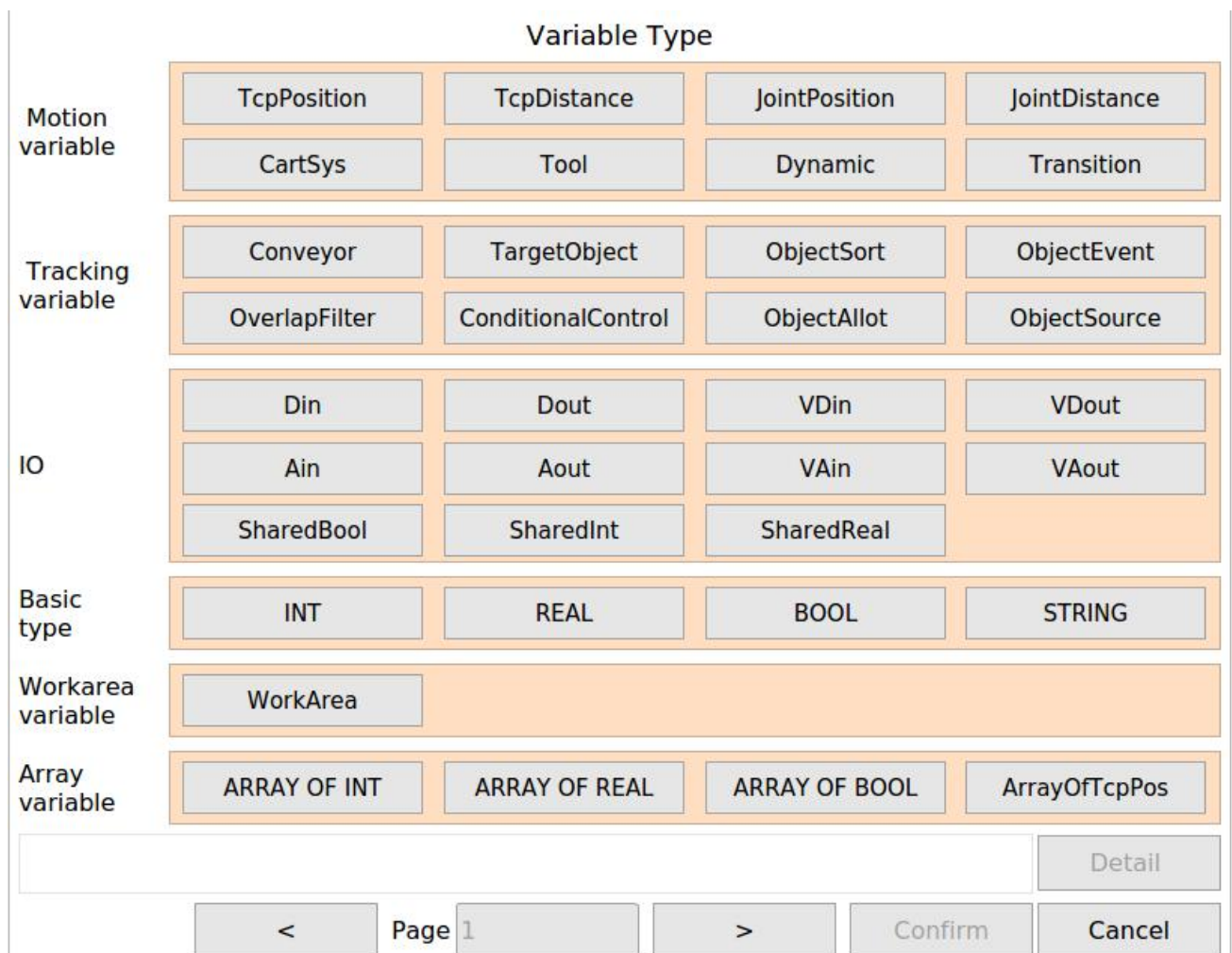


Figure 6.2 variable type interface

6.3 Variable Help

When creating a new variable, click the "detailed description" button in Figure 6.2 to jump to the help page of the variable. Take the targetobject variable as an example, as shown in Figure 6.3.

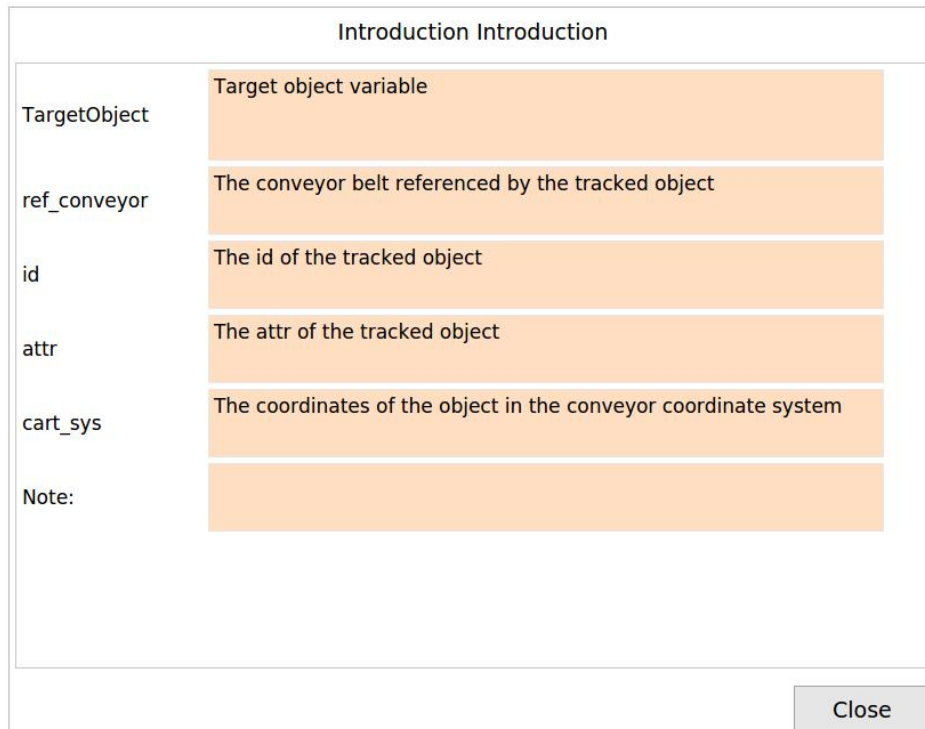


Figure 6.3 variable introduction interface

6.4 Edit

6.4.1 Copy

Copy a variable. It supports cross program replication, cross project replication, global variables can be copied as program variables, and program variables can be copied as global variables.

Note: when copying the conveyor variable, the object source parameter of the copied variable is empty.

6.4.2 Paste

Paste the copied variable.

Rename referenced variables when pasting

If the copy of the reference variable is included in the pasting process, and the reference variable name has been occupied in the current range, the system will rename the reference variable. Renaming rule: add a suffix to the original variable name `_New number`. For example:

The coordinate system variable FR1 in the program *test* refers to the coordinate system FR2;

When FR1 is copied to the program *mem*, FR2 will be copied at the same time. If the program *mem* contains a variable named FR2, FR2 will be renamed FR2_new;

If the program *mem* contains a variable named FR2_New, rename FR2 to FR2_New suffix is a number starting from 0;

6.4.3 Rename

Rename a variable.

6.4.4 Delete

Deletes the specified variable. Only variables that are not used by programs or other variables can be deleted.

6.5 Other

6.5.1 View Track

View the track of the arrayofTcpPosition type variable. Select the arrayofTcpPosition type variable and click the "track" button in "other", as shown in Figure 6.4.

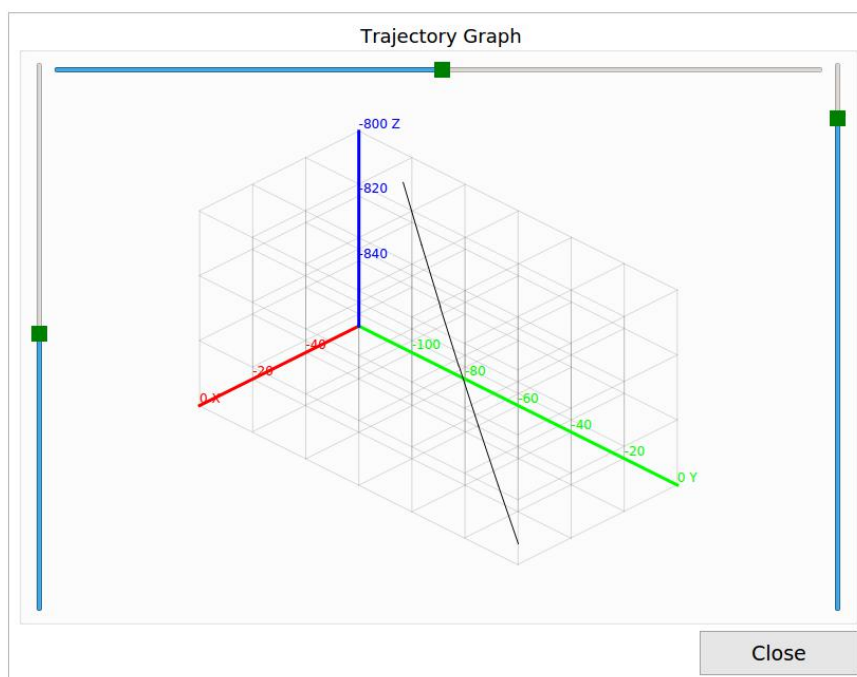


Figure 6.4 TCP array trace interface

6.5.2 Clear Unused

Clear unused variables.

Select the program to clear unused program variables; Check root to clear unused global variables.

You cannot clear unused variable types: WorkArea、ObjectSource、ObjectSort、ObjectAllot、ObjectEvent、ConditionalControl、OverlapFilter、TcpConnect、HardTrigger.

6.5.3 Delete Element

Delete array elements.

Delete step

Select the process array element and click "delete element" in "other" to pop up the delete array element dialog box, as shown in Figure 6.5. In which, the modification types are forward, current element and backward.

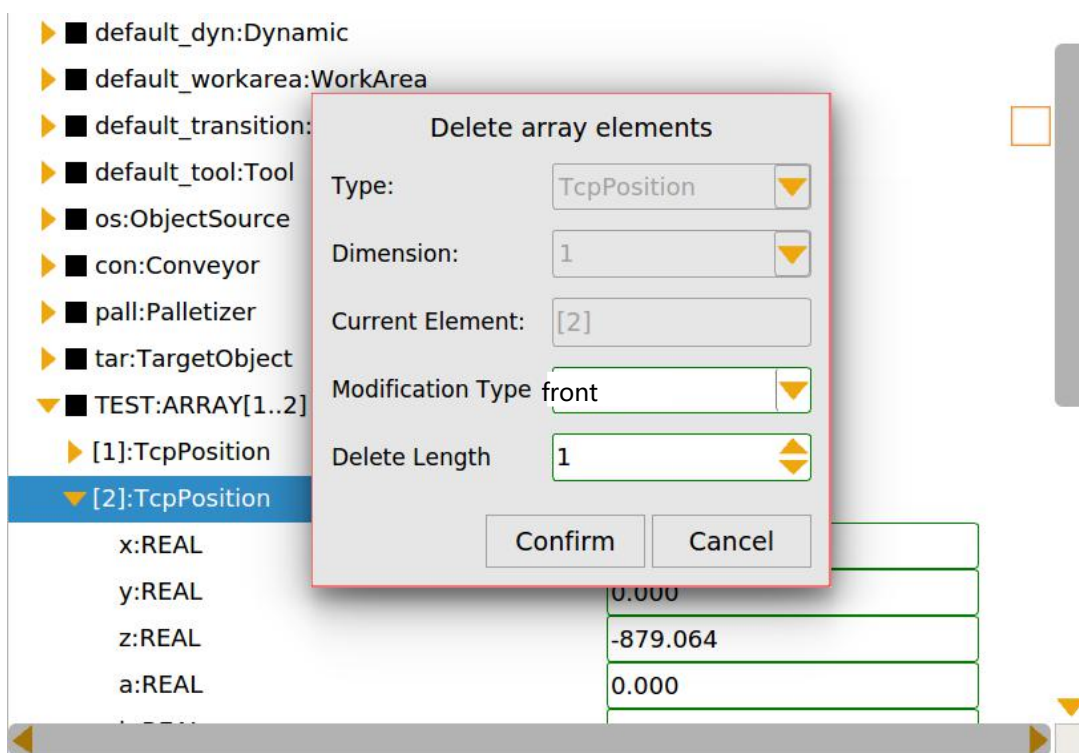


Figure 6.5 interface for deleting array elements

Delete Principle

- (1) When you select forward and backward deletion, the deletion length can be entered by yourself, and the current element is not included during deletion; When the current element is selected, the deletion length cannot be selected. Only the current element can be deleted;
- (2) When deleting forward, if the deletion length exceeds the previous number of current elements, adjust the deletion length and prompt whether to continue to delete;
- (3) When deleting backward, if the deletion length exceeds the number after the current element, adjust the deletion length and prompt whether to continue to delete;

(4) If one element remains after deleting, it is not allowed to be deleted, and a pop-up prompt will be displayed.

6.5.4 Insert Element

Insert array elements. Insert a specified number of elements before or after the currently selected element, as shown in Figure 6.6.

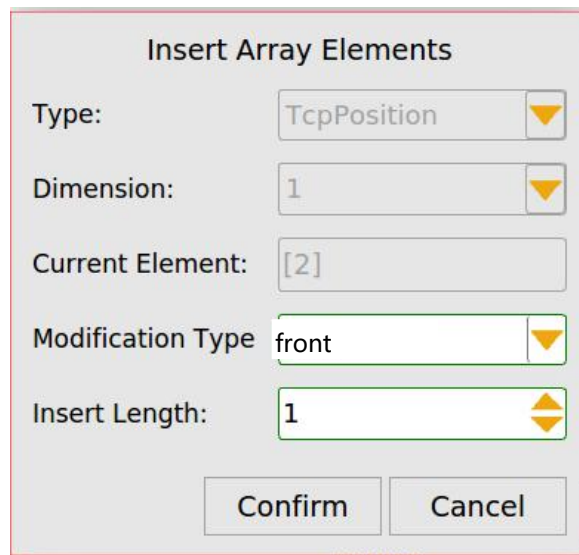


Figure 6.6 inserting array elements

6.5.5 Expand

Expand the display variables page node.

Expansion principle: only the next level nodes of the selected node are expanded.

There is a program *test* in the project *pp*, and there are variables in *test*. When *pp* is put away, click "expand", as shown in Figure 6.7.



Figure 6.7 expanded interface

6.5.6 Collapse

Collapse the display variables page node.

Collapse principle: collapse nested nodes such as the selected node, its child nodes and the child nodes of the child nodes.

Before Collapsing, as shown in Figure 6.8, click "Collapse" and then click the expansion icon in front of the item, as shown in Figure 6.7.

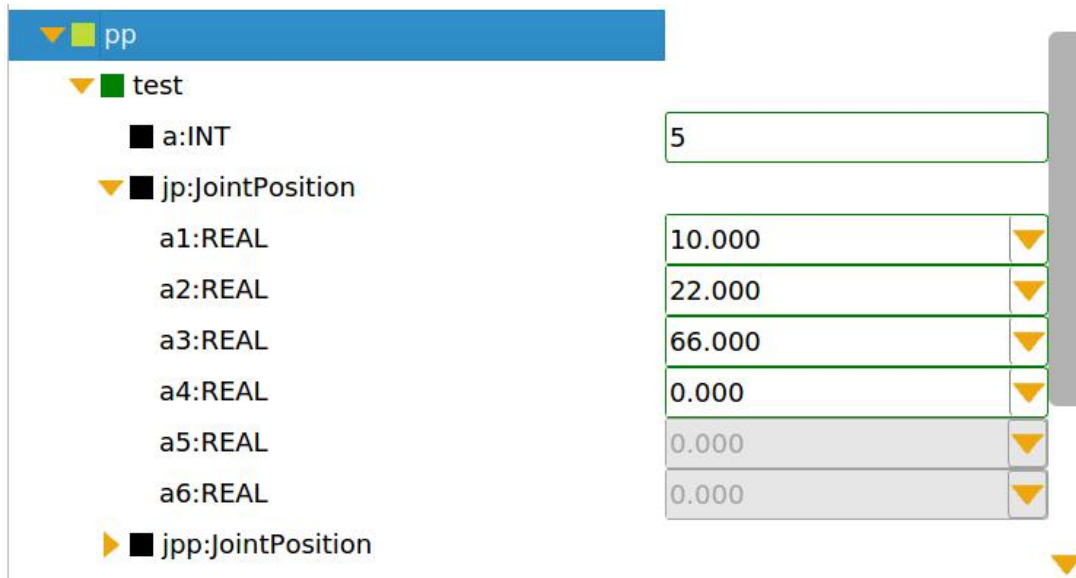


Figure 6.8 interface before folding

6.6 Variables that the operator can manipulate

If you need to view and modify the specified variable in the operator interface, tick the corresponding item.

6.7 Load

Reload the currently loaded project into the controller.

6.8 Variable ordering

Sort variables in all programs.

Note: The sorting function does not include default global variables. Regardless of the sorting method selected, the display position of default global variables remains unchanged. Type sort and alphabetic sort are mutually exclusive, that is, the two sorting methods cannot be used together.

6.8.1 Type the sorting

Sort variables in ascending alphabetical order by type, grouped together.

For example, the variables of Aout, REAL, and Alarm are sorted as Alarm, Aout, and REAL

6.8.2 Alphabetical the sorting

It is case insensitive and has ascending and descending order.

If the variable name list is ddd, aa, ab, def, ss, then aa, ab, ddd, def, ss is sorted

6.9 Variable Filtering

Filter variables according to different requirements. It can be divided into type filtering and name filtering. The two filters are an or relationship, that is, variables that meet the type filter and name filter requirements are displayed.

6.9.1 Type filtering

Filter display of variable type groups or specified variable types.

6.9.2 Name filtering

Displays a case-insensitive variable containing the name selection.

For example, if the variable name list is ABC, CDB, and AFDB, the user enters B in the name filter box to display the variables ABC and AFDB

6.9.3 remove

Restore the default display, that is, when the interface was first running.

6.10 Program storage structure versus variable value

The HMI is successfully connected to the controller.

First of all, the comparison of the storage structure of the program, such as variables, instruction changes. If they are inconsistent, the system prompts whether to load and replace them, as shown in the following figure.

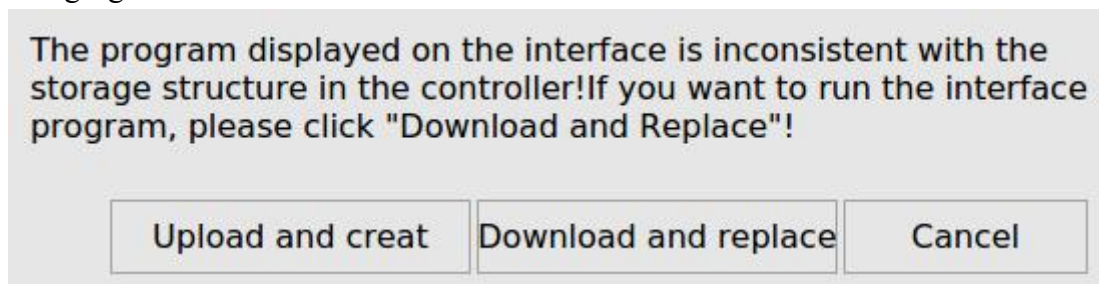


Figure 5.9 Comparison between HMI program and controller program structure

- "Upload and create": refers to uploading the project of the controller to the HMI and creating a new project that is the same as the uploaded project, so there is no need to compare the variables.
- "Download and replace" : the HMI program is loaded and replaced to the controller, without the need for variable value comparison.
- "Cancel": No operation is performed, and the lower and upper computers remain in the current state respectively.

Note: the robot always runs the program in the controller.

Second, if the program storage structure is consistent, the variable values are compared. If the variable values are inconsistent, a dialog box is displayed, as shown in the following figure.

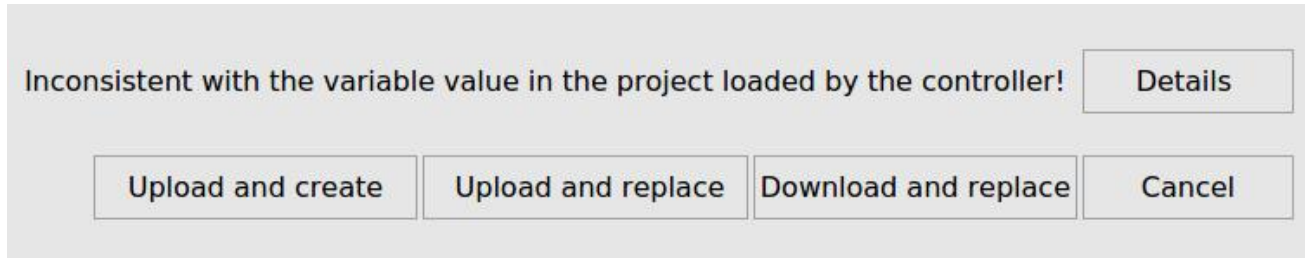


Figure 6.10 Compare the HMI variable value with the controller variable

- "Details", the window will display the values of controller variables and HMI variables in specific variables, as shown in the following figure.
- "Upload and create" refers to uploading the project in the controller to the HMI, and creating a new project in the HMI that is the same as the uploaded project.
- "Upload and replace" refers to the replacement of HMI variable values by variable values in the controller.
- "Download and replace" means that the HMI variable value replaces the variable value in the controller.
- "Cancel" means to do nothing.

Inconsistent with the variable value in the project loaded by the controller!

Variable name	Variable value in controller	Variable value in HMI
Global variable		
os.filter_enable	FALSE	TRUE

Upload and create Upload and replace Download and replace Cancel

Figure 6.11 Comparison of HMI variable values with controller variables - Details

6.11 Variable number

Variable numbers were added to support runtime modification of variables.

Variable number definition:

A numbered list of variables or variable members with a hierarchical structure, i.e. item number - program number - variable number - variable member number.

Variable numbering rules:

Numbering starts from 1;

Take the program or global or project as the unit, that is, the variable number is unique in the program or global or project, any program or global or project can have a variable numbered 1, but a program or global or project can only have a variable number 1. There is a variable numbered 1;

When creating a new variable, the variable number is the maximum value of all variable numbers in the program or global or project to which the variable belongs +1.

The project number and program number of global variables are 0; the program number of

project variables is 0; the number of variable members is fixed.

Table 6.1 Examples of variable numbers

Variable Category	Naming Rules
Global variables	Default product variables are numbered 0-0-7
Project variables	The program number to which the project variable belongs is 0, the project number to which the program where the variable belongs is 2, and the variable number is 2, the number is displayed as 2-0-2
Program variables	The program number to which the variable belongs is 1, the item number of the program to which the variable belongs is 2, and the variable number is 2, the number is displayed as 2-1-2

View variable operations:

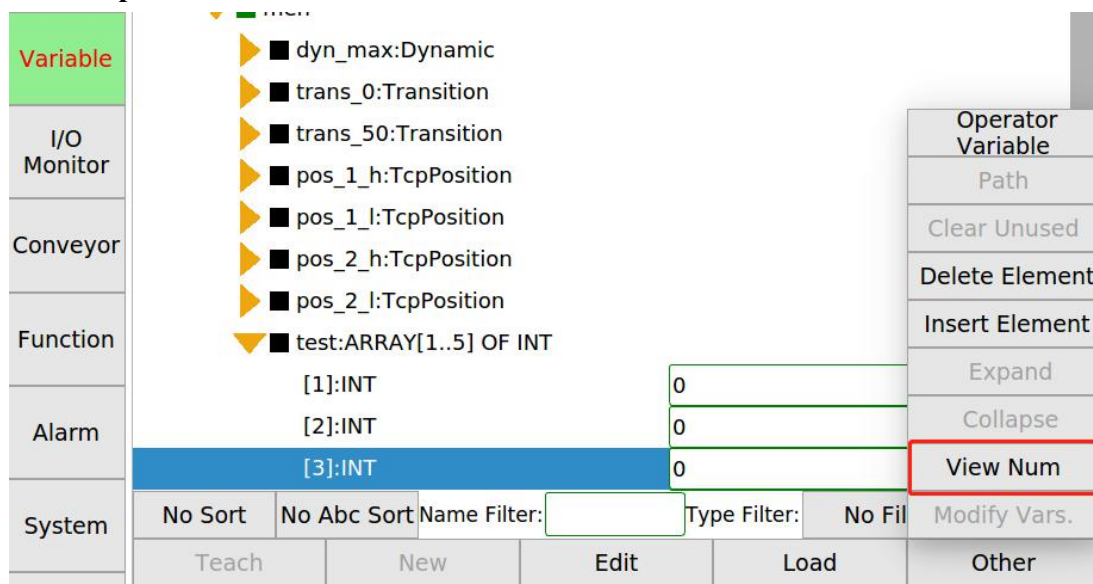


Figure 6.13 View variable numbers

6.12 Modify program variables at run time

Supports runtime modification of program variables, which can be modified under HMI or TCP control.

The following variable members are not supported for runtime modification:

- Read-only members cannot be modified;
- Reference members cannot be modified;
- Id and attr in TargetObject variables cannot be changed;

- The members of all variables in input and output types cannot be changed.
- Conveyor belt model and resolution

6.12.1 HMI controls modifying variables

Under HMI control, variables cannot be modified at the default runtime; If you need to modify, select "Others → Modify variables" to directly modify the values of all program variables, as shown in the following figure.

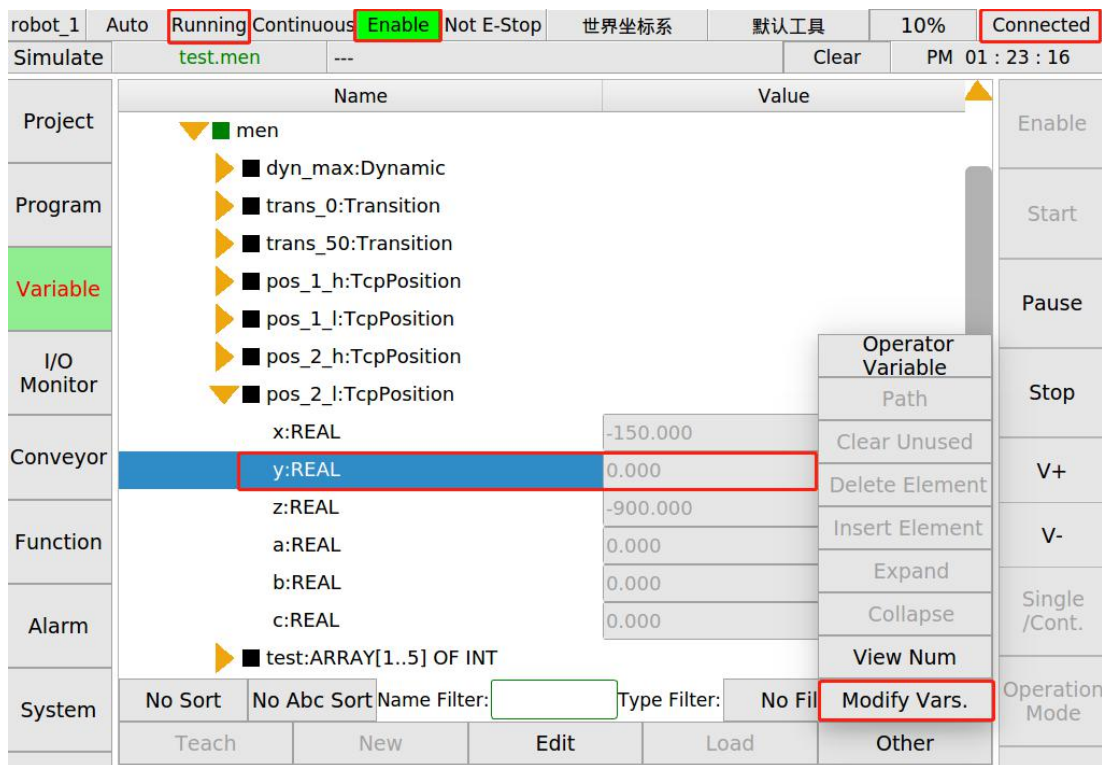


Figure 6.14 HMI controls modifying variables

6.12.2 TCP control modifies variables

TCP control variable can be modified by sending instructions to modify the value of the variable, as follows:

- (1) Select TCP control in the function block, select connection type as "server" and data type as "Simple String", as shown in the following figure.

The screenshot shows a configuration window titled "TCP Control". It contains the following elements:

- Connection Type:** A dropdown menu with "Server" selected.
- Data Type:** A dropdown menu with "Simple string" selected.
- IP:** A text input field containing "127.0.0.1".
- Port:** A text input field containing "60001".
- Instruction Logs:** A large, empty rectangular area for viewing logs.
- Buttons:** "Clear Logs" and "Save" buttons located at the bottom right.

Figure 6.15 Example of configuring TCP control

(2) TCP control can be realized when HMI is switched to TCP control mode and HMI control is obtained.

(3) You can view the number of variables through the methods in Chapter 6.1.1 and send fixed format instructions to modify the variable value. Single or multiple modifications are supported. The input values are separated by commas and end with semicolons. The punctuation is half width (English punctuation).

Single: [set_variable_of_number, number length, item number, program number, variable number, variable parameter number, value;]

More than one: [set_variable_list_of_number; n, number length, item number, program number, variable number, variable parameter number, value, Number length, item number, program number, variable number, variable parameter number, value;]

n: Modify the number of variables.

The serial number is 1:1 to 1-7-2. The serial length is 4.

The serial number is 2-1-7-2-3 and contains 5 characters.

After sending the command, the feedback value will be received in the TCP sender and THE TCP control part of the HMI function block respectively. If the feedback value is 1, the modification

is successful. If the feedback value is 0, the modification fails. The following table lists specific command examples.

Table 6.2 Example of sending instructions

Variable to be modified	Serial number	For example	
test.men. pos_2_1.y	1-2-7-1	Send	set_variable_of_number,4,1,2,7,1,15;
		Sender feedback	请求实际发送数据: set_variable_of_number,4,1,2,7,1,15; 设置变量值: 是否成功: 1
		HMI feedback	2:127.0.0.1:45666 Receive message:<set_variable_of_number,4,1,2,7,1,15;>! 2:127.0.0.1:45666 returned messages:<1,set_variable_of_number;>
test.men. pos_2_1.x test.men. pos_2_1.y	1-2-7-1	Send	set_variable_list_of_number,2,4,1,2,7,1,4,4,1,2,7,2,5;
	1-1-7-2	Sender feedback	请求实际发送数据: set_variable_list_of_number,2,4,1,2,7,1,4,4,1,2,7,2,5; 设置变量值: 是否成功: 1
	HMI feedback	2:127.0.0.1:45666 Receive message:<set_variable_list_of_number,2,4,1,2,7,1,4,4,1,2,7,2,5,;>! 2:127.0.0.1:45666 returned messages:<1,set_variable_list_of_number;>	

Chapter7 I/O Monitoring Interface

The I/O monitoring interface displays all I / O modules.

7.1 Digital IO

Digital IO			Analog IO		
Name	Custom Name	Value	Name	Custom Name	Value
▼ Actual Input					
X1		<input type="checkbox"/>	X2		<input type="checkbox"/>
X3		<input type="checkbox"/>	X4		<input type="checkbox"/>
X5		<input type="checkbox"/>	X6		<input type="checkbox"/>
X7		<input type="checkbox"/>	X8		<input type="checkbox"/>
X9		<input type="checkbox"/>	X10		<input type="checkbox"/>
X11		<input type="checkbox"/>	X12		<input type="checkbox"/>
X13		<input type="checkbox"/>	X14		<input type="checkbox"/>
X15		<input type="checkbox"/>	X16		<input type="checkbox"/>
▶ Actual Output					
▶ Virtual Input					
▶ Virtual Output					

Set VDout Restore VDout Display IO Set Dout Restore Dout

Figure 7.1 Digital IO monitoring interface

7.1.1 Set the digital output value

- (1) Click the value column and modify the check status of the check box to change the value of an output;
- (2) Click the "Set VDout" button to set all virtual digital outputs to high level;
- (3) Click the "Restore VDout" button to restore the IO level to the virtual digital output value.
- (4) Click the "Set Dout" button to set all actual digital outputs to high level;
- (5) Click the "Restore Dout" button to restore the IO level to the actual digital output value.

7.2 Analog IO

Digital IO		Analog IO	
Name	Value		Custom Name
▼ Actual Input			
AX1	0		
AX2	0		
AX3	0		
AX4	0		
AX5	0		
▼ Actual Output			
AY1	0	Set	
AY2	0	Set	
AY3	0	Set	
AY4	0	Set	
▶ Virtual Input			
▶ Virtual Output			

Display IO

Figure 7.2 Analog IO monitoring interface

7.2.1 Set the analog output value

- (1) Click the “Set” button in the value column;
- (2) Specify the setting value and click the “Confirm” button.

Variable Name: AY1

Custom Name:

Variable Value:

Figure 7.3 Analog output value setting interface

7.3 Other

Digital IO		Analog IO		Other	
Name	Value		Custom Name	Operator	
▶ Floating poin...					
▼ Floating poin...					
RealOutput1	0.000	Set		<input type="checkbox"/>	
RealOutput2	0.000	Set		<input type="checkbox"/>	
RealOutput3	0.000	Set		<input type="checkbox"/>	
RealOutput4	0.000	Set		<input type="checkbox"/>	
RealOutput5	0.000	Set		<input type="checkbox"/>	
RealOutput6	0.000	Set		<input type="checkbox"/>	
RealOutput7	0.000	Set		<input type="checkbox"/>	
RealOutput8	0.000	Set		<input type="checkbox"/>	
RealOutput9	0.000	Set		<input type="checkbox"/>	
RealOutputu...	0.000	Set		<input type="checkbox"/>	
RealOutputu...	0.000	Set		<input type="checkbox"/>	
RealOutputu...	0.000	Set		<input type="checkbox"/>	
RealOutputu...	0.000	Set		<input type="checkbox"/>	
RealOutputu...	0.000	Set		<input type="checkbox"/>	

Figure 7.4 Other monitoring interface

7.3.1 Set other output value

- (1) Click the “Set” button in the value column;
- (2) Specify the setting value and click the “Confirm” button.

Variable Name: RealOutput1

Custom Name:

Variable Value:

Figure 7.5 Other output value setting interface

7.4 Custom Name

Click the cell in the user-defined name column to set a user-defined name to facilitate the memory of an output or input function in the actual project application.

7.5 IO that the operator can manipulate

If you need to view and modify the specified IO in the operator interface, tick the corresponding item.

Chapter8 Conveyor

The conveyor interface is used to configure object source, teach conversion relationship, view data buffer, view data history and view statistical results.

Create a new conveyor variable, switch to the conveyor page, as shown in Figure 8.1

The screenshot shows the conveyor configuration interface. At the top, there is a dropdown menu for 'Conveyor:' with 'pp:cc' selected. Below it are five tabs: 'Conveyor' (highlighted in green), 'Teach', 'Buffer', 'History', and 'Count'. The main configuration area includes several fields and buttons:

- Encoder port:** Encoder0 (dropdown)
- Conveyor model:** LINE (dropdown)
- Encoder count:** [input field] inc
- Current conveyor velocity:** 0.000 mm/s
- Encoder accuracy:** 1.000 inc/mm
- Min. workarea:** 200.000 mm [Teach button]
- Max. workarea:** 600.000 mm [Teach button]
- Latest receiving distance:** 400.000 mm

Below the main configuration is a section titled 'Conveyor Coord.' with six input fields for coordinates:

- x: 0.000
- y: 0.000
- z: 0.000
- a: 0.000
- b: 0.000
- c: 0.000

Figure 8.1 conveyor interface

8.1 Conveyor Configuration

Current conveyor: Display the currently selected conveyor.

Belt selection: Select the conveyor, and the selection list contains the conveyor variables of the loader and the global conveyor variables.

The conveyor model has: line, disc and static. Different conveyor models support different teaching methods.

8.1.1 Encoder

Encoder code value: The code value of the current encoder.

Encoder interface: Name of equipment recording the location of conveyor, including encoder, external shaft and other equipment.

Encoder accuracy: Assuming that the encoder accuracy is 13inc/mm, it means that the encoder changes 13 code values, and the actual running distance of the conveyor is 1mm.

Note: encoder accuracy cannot be set to 0.

8.1.2 Work Area

(1) As shown in Figure 8.2, the working area is the working area that the robot can track, and the latest working area refers to the latest receiving area distance.

(2) In the case where no object is tracked, the object is not tracked when it is in the waiting area, and it needs to wait for the object to enter the preparation area to start tracking. If the object enters the latest work area during the tracking process, you can continue to track. When the object enters the super-boundary area, it will not continue to track the object, and the remaining tracking actions will be completed directly.

(3) If the object has not yet started to be tracked, and the object has entered the latest work area, the tracking of the object will be abandoned directly.

(4) When the speed of the manipulator is less than 2 times of the speed of the conveyor belt, it will prompt "the speed of the conveyor belt does not match the tracking speed". At this time, it is necessary to reduce the speed of the conveyor belt or increase the speed of the manipulator.

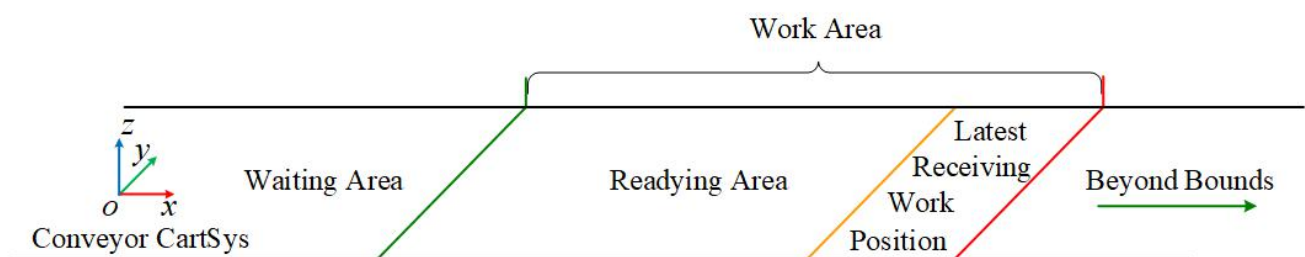


Figure 8.2 work area

8.1.3 Conveyor Coordinate System

Conveyor coordinate system: the coordinates of the conveyor in the world coordinate system. Calculate through five point teaching or three-point teaching (please refer to the teaching chapter for specific teaching process).

8.2 Teach

Teach is divided into five-point teach, three-point teach, circle center teach and static teach. Different conveyor models support different teaching methods.

8.2.1 Teaching Method

Teaching method supported by linear conveyor: five-point teach, three-point teach, senior teach.

Teaching method supported by disc conveyor: five-point teach, three-point teach, senior teach, circle center teach .

Teaching method supported by disc conveyor: static teach.

8.2.2 Five Point Teach

Scope of application

Use the vision system to detect the location of objects.

Teaching steps

(1) Select five points teaching;

(2) Teach the first point, as shown in Figure 8.3;

Place the object in a corner of the visual area and click the "empty workpiece" button;

Click the "workpiece capture" button to obtain the location of the object in the visual coordinate system and the encoder code value.

Visual position X:	<input type="text" value="0.000"/>	Workpiece get successfully!
Visual position Y:	<input type="text" value="0.000"/>	
Encoder position:	<input type="text" value="0"/>	
<input type="button" value="Get workpiece"/> <input type="button" value="Clear workpiece"/>		

Use the camera to inspect the workpiece to be tracked. For best monitoring results, the workpiece should be at the corner of the inspection area

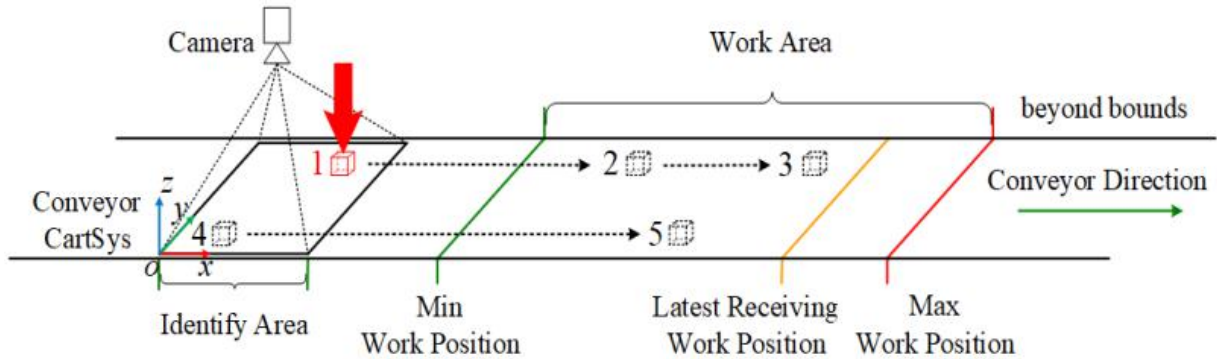


Figure 8.3 first point interface of five point teaching

Note: when there are multiple objects in the buffer, to prevent incorrect objects, please click "empty workpiece" first, and then click "workpiece grab".

(3) Teach the second point

Start the conveyor, move the object to the working area, and stop the conveyor;

Start the manipulator, align the end with the center of the object, and click the "teaching" button to obtain the current location of the manipulator.

Robot position X:	<input type="text" value="21.700"/>	Encoder position:	<input type="text" value="0"/>
Robot position Y:	<input type="text" value="-129.602"/>	Workpiece get successfully!	
Robot position Z:	<input type="text" value="-902.102"/>		
<input type="button" value="Teach"/>			

Start the conveyor to move the workpiece into the work area and teach it

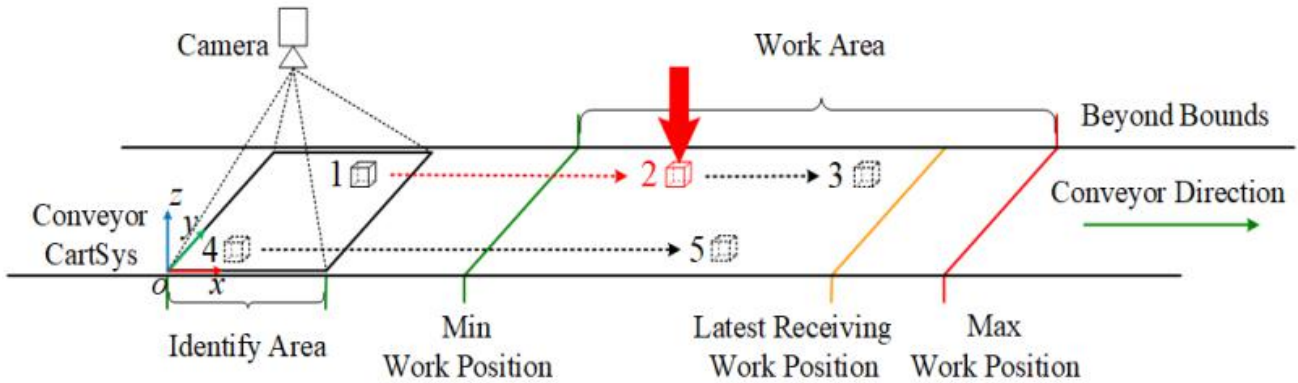


Figure 8.4 the second point interface of five point teaching

(4) Teach the third point

Start the conveyor, move the object to a location close to the maximum working area, and stop the conveyor;

Jog the manipulator, align the end with the center of the object, and click the "teach" button to obtain the current location of the manipulator.

Robot position X:	<input type="text" value="-42.274"/>	Encoder position:	<input type="text" value="0"/>
Robot position Y:	<input type="text" value="-39.789"/>	Encoder resolution:	<input type="text" value="0.000"/>
Robot position Z:	<input type="text" value="-884.138"/>	Teach succeed!	
<input type="button" value="Teach"/>			

Start the conveyor to move the workpiece close to the end of the work area and teach it

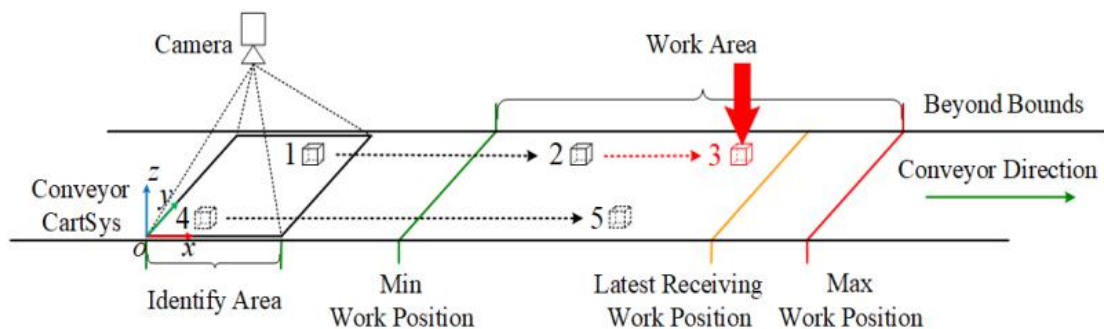


Figure 8.5 the third point interface of five point teaching

(5) Teach the fourth point

Place the object on the opposite corner of the first teaching point and click the "empty workpiece" button;

Click the "workpiece capture" button to obtain the location of the object in the visual coordinate system and the encoder code value.

Visual position X:	<input type="text" value="0.000"/>	Fetch workpiece succeed!
Visual position Y:	<input type="text" value="0.000"/>	
Encoder position:	<input type="text" value="0"/>	
<input type="button" value="Get Workpiece"/> <input type="button" value="Clear workpiece"/>		

Use the camera to inspect the second workpiece to be tracked. For best monitoring results, place the workpiece close to the diagonal of the over-inspection area

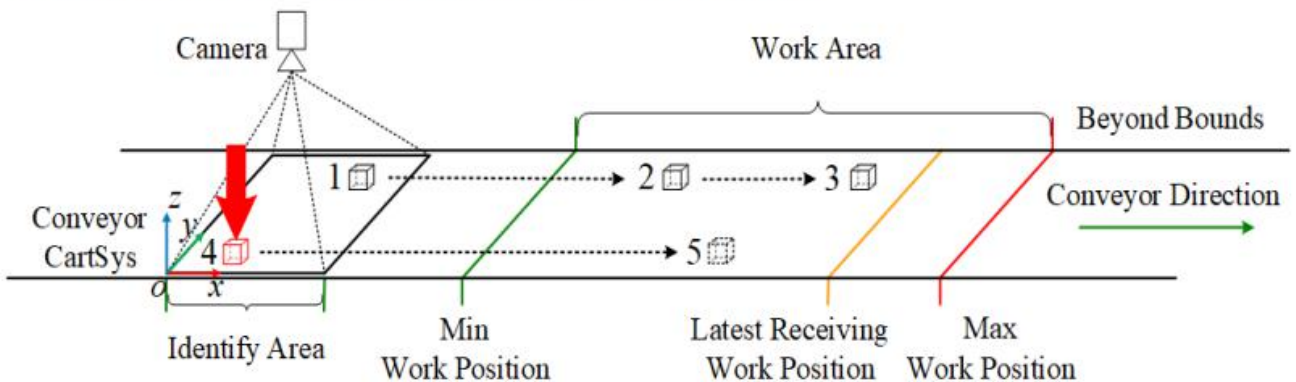


Figure 8.6 fourth point interface of five point teaching

(6) Teach the fifth point

Start the conveyor, move the object to the working area, and stop the conveyor;

Jog the manipulator, align the end with the center of the object, and click the "teach" button to obtain the current location of the manipulator.

Robot position X:	<input type="text" value="-82.886"/>	Encoder position:	<input type="text" value="0"/>
Robot position Y:	<input type="text" value="0.306"/>	Position error:	<input type="text"/>
Robot position Z:	<input type="text" value="-914.199"/>	Proportional error:	<input type="text"/>
<input type="button" value="Teach"/>		Teach succeed!	

Start the conveyor to move the workpiece into the work area and teach it

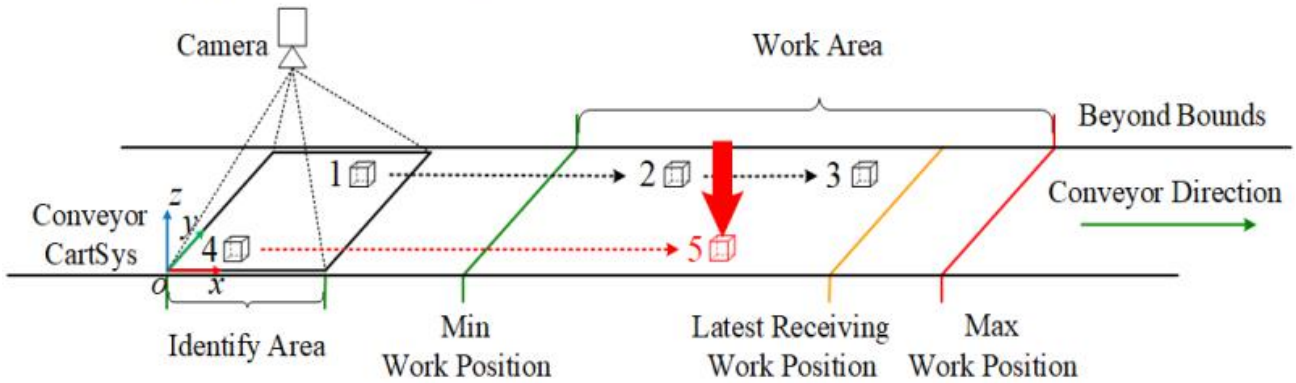


Figure 8.7 fifth point interface of five point teaching

Proportional error: the distance ratio between the teaching point and the actual point. The closer it is to 1, the more accurate the teaching is.

(7) Teach workspace

The working area is divided into the minimum value of the working area, the maximum value of the working area and the latest acceptance area distance. Refer to the workspace section.

Latest accept distance:	<input type="text" value="400.000"/>	
Min. workarea:	<input type="text" value="200.000"/>	<input type="button" value="Teach"/>
Max. workarea:	<input type="text" value="600.000"/>	<input type="button" value="Teach"/>

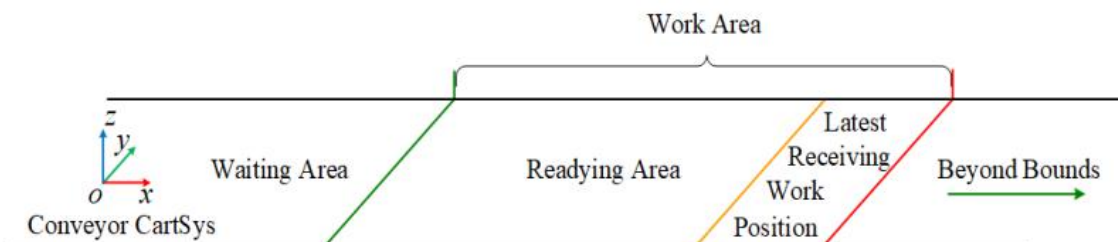


Figure 8.8 teaching workspace interface

Note: the z-direction inclination of the conveyor is not supported;

If an error occurs during teaching, the prompt message box will display a red error message; If the teaching is successful, the prompt message box will display a green success message.

8.2.3 Three Point Teach

Scope of application: Use sensors to detect the location of objects.

Teaching steps: Repeat steps (1) (2) (3) (4) (7) of the five point teaching.

Note: the z-direction inclination of the conveyor is not supported.

8.2.4 Center of Circle Teach

8.2.4.1 Scope of application

Teach of disc center location during disc tracking.

8.2.4.2 Teach steps

Teach methods: direct teaching and three-point teaching.

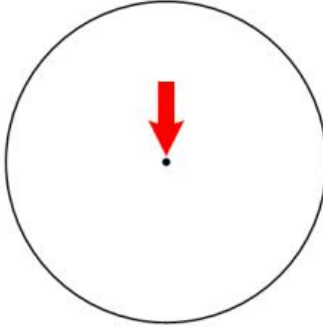
◆Direct teach steps

The direct teaching method is suitable for the known position of the center of the circle, teaching the center of the circle.

Teaching method of center of circle

Direct teach get circle center

Three point teach get circle center



	x	y	z	
Circle center	<input type="text"/>	<input type="text"/>	<input type="text"/>	Teach
Point 1	<input type="text"/>	<input type="text"/>	<input type="text"/>	Teach
Point 2	<input type="text"/>	<input type="text"/>	<input type="text"/>	Teach
Result:	<input type="text"/>	<input type="text"/>	<input type="text"/>	

Figure 8.9 center of circle teaching interface

(1) Move the robot align the center of circle;

(2) Select any “Teach” button to teach, After teaching, the displayed value of the result is the teaching value of the center of the circle.

◆Three point teach steps

The three-point teaching method is suitable for three points on a known circle,teaching the center of the circle.

Teaching method of center of circle

Direct teach get circle center

Three point teach get circle center

	x	y	z	
Point 1				Teach
Point 2				Teach
Point 3				Teach
Result:				

Figure 8.10 center of circle teaching interface

(1) Move the robot position to three points on the circle in turn, and click the “Teach” button respectively;

(2) After teaching, the displayed value of the result is the teaching value of the center of the circle.

Note: if an error occurs during teaching, the prompt message box will display a red error message; If the teaching is successful, the prompt message box will display a green success message.

8.2.5 Static Teach

Place the object under vision, obtain the location of each object and the end location of the manipulator corresponding to each object location, and at least two location points. The more location points, the more accurate the teaching is, and support up to nine location points.

Vision Coord. X(mm)	Visual Coord. Y(mm)	World Coord. X(mm)	World Coord. Y(mm)	World Coord. Z(mm)		
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	GetObj.	Teach
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	GetObj.	Teach

Result: Clear Compute +

Teaching Instructions:
To place objects visually, first empty the workpiece. Then click to acquire the object, align the end to the object, and click teach, that is, to obtain a set of correspondence between the vision and the end position. Repeat this at least twice, the more times, the more accurate the teaching will be.

Finish Cancel

Figure 8.11 static teaching interface

Note: it is recommended to click the "clear workpiece" button every time before obtaining the object to prevent error in the corresponding relationship of the object.

8.2.6 Senior Teach

■ Scope of application

Teach with a minimum of 3 and a maximum of 9 objects to determine the position of the object detected by the vision system.

■ Teaching steps

(1) Select Advanced Teaching, click the Get Object button to get the object list, you can delete the acquired objects, filter the objects to be taught, and click OK after filtering. The object list interface is shown in the figure below.

No.	Original data	
1		Delete

Figure 8.12 Advanced teaching object list interface

(2) Only add at least 3 objects. If the number of objects obtained through step (1) is less than 3, you need to repeat step (1)

(3) Teach the objects according to the serial numbers, as shown in Figure 8.13, teach the object with the serial number 3.

Empty Objects: Click Empty Objects to empty all objects on the conveyor belt.

Configuration: Click the configuration button to pop up the configuration interface, as shown in Figure 8.14.

Modify serial number: Modify the serial number of the object, the serial number is a number from 1-9, and cannot be repeated.

Cancel: Cancel the modification of the serial number of the object.

Confirm: Determine the serial number of the modified object.

No.	Point1	Point2	VisionPos.	
1			■	Delete
2			■	Delete
3			■	Delete

Serial number:

Original data:

Vision position

x y z

Encoder value:

Teaching point 1

x y z

Encoder

Teaching point 2

x y z

Encoder

Figure 8.13 Advanced teaching interface

(4) After the teaching in sequence is completed, click the Calculate button, and the configuration interface will pop up, as shown in the figure below.

Whether to keep the conveyor coordinate system

Whether to preserve the conveyor resolution

Whether to keep camera parameters

Whether to keep the camera image (x and y swapped)

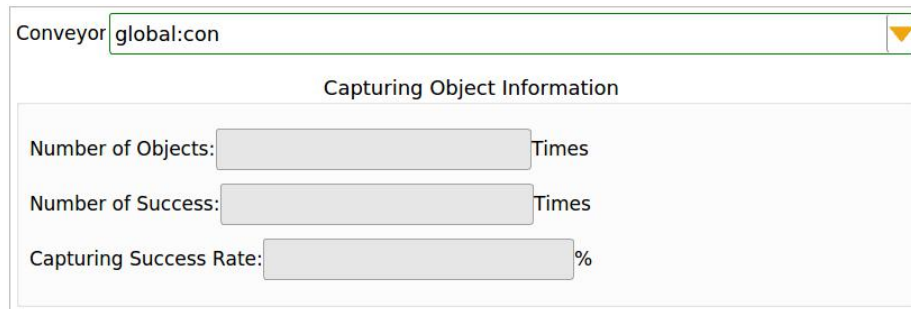
Figure 8.14 Configure interface

Whether to keep the conveyor belt coordinate system: If selected, the conveyor belt coordinate system information will keep the current value unchanged; otherwise, the coordinate system value determined by the teaching result will be retained.

Note: the data in the history record will not be saved after shutdown, but only the latest 1024 pieces of information received after the system is started. If the user clicks the clear all button, the record will continue after the clearing time, and the previous data will be cleared.

8.5 Statistics

View the success rate of the current tracking object, as shown in Figure 8.17.



Conveyor global:con

Capturing Object Information

Number of Objects: [] Times

Number of Success: [] Times

Capturing Success Rate: [] %

Figure 8.17 grab object information interface

Note: if you want to count the success rate at a certain time, you need to clear all the historical information in the data history interface.

Chapter9 Function Block Interface

The function block interface is shown in Figure 9.1.

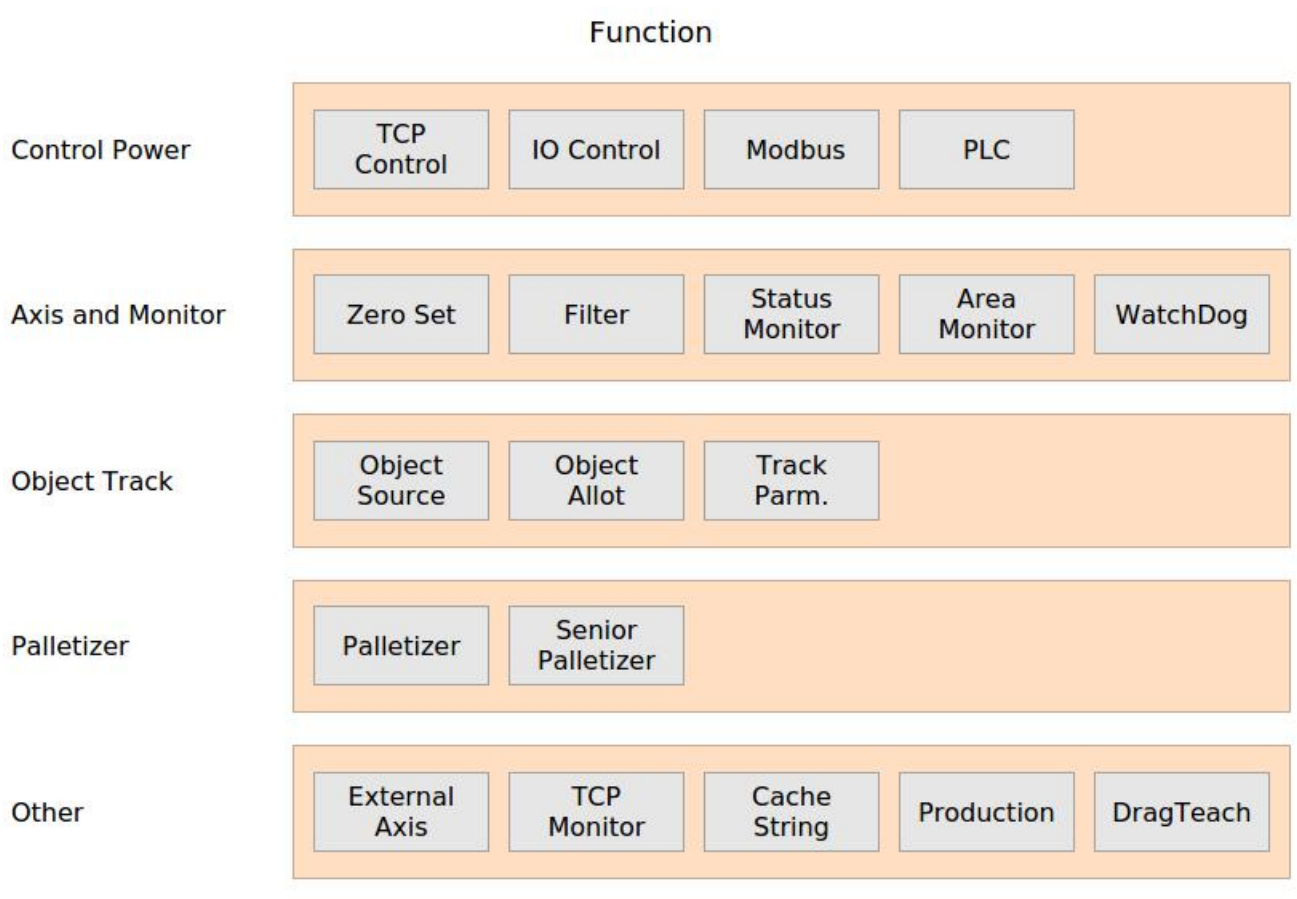


Figure 9.1 function block interface

9.1 Zero Setting

9.1.1 Set axis zero

It includes robot axis zero setting, external axis zero setting, encoder zero clearing and axis position setting. As shown in Figure 9.2.

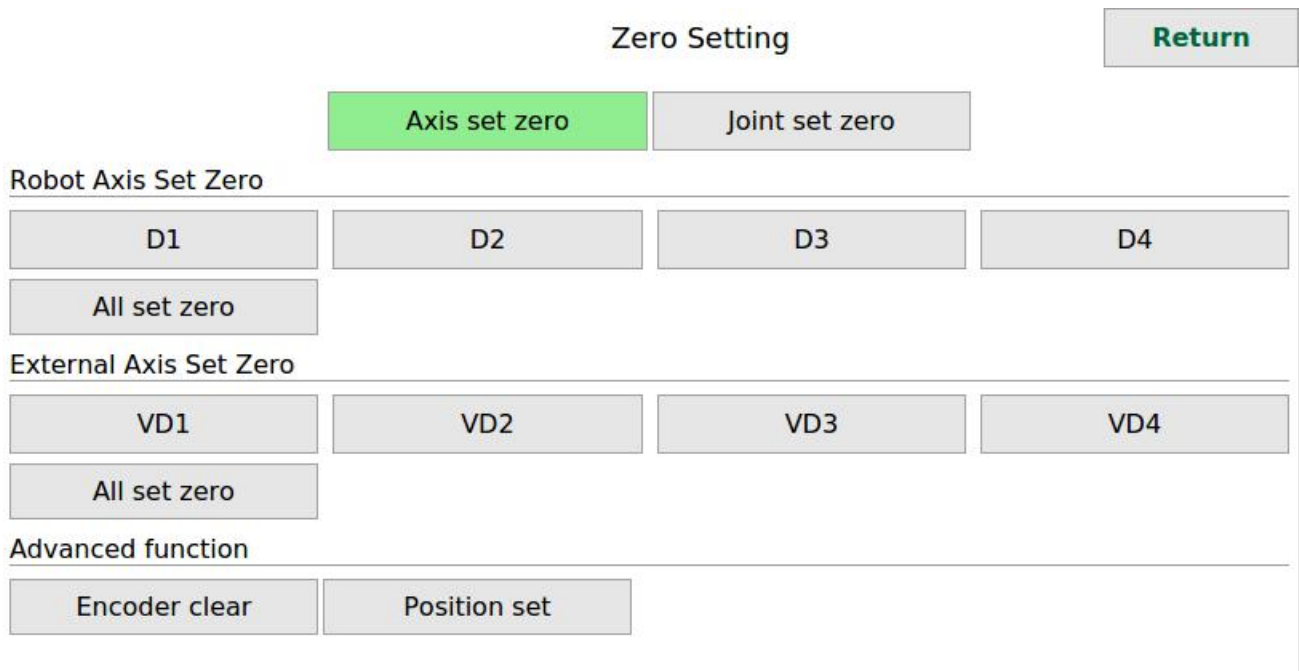


Figure 9.2 axis zero point setting interface

Robot axis zero setting

Single axis zero setting: such as clicking the button “D1”, set the current position of the robot axis “D1” to the “D1” zero;

Set all to zero: set the current position of all axes of the robot to the zero.

External axis zero setting

Single axis zero setting: such as clicking the button “VD1”, set the current position of the external axis “VD1” to the “VD1” zero;

Set all to zero: set the current position of all external axes to the zero.

Encoder reset

The encoder code value is cleared.

Note: encoder zeroing is only for module encoder, and slave encoder does not have this function.

Axis position setting

Sets the current position of the specified axis to the specified angle.

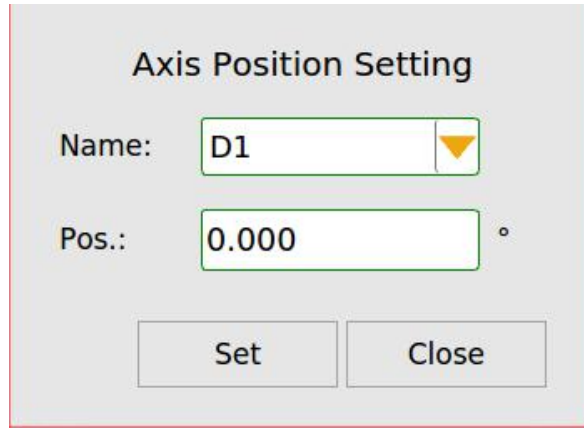


Figure 9.3 axis position setting

9.1.2 Set joint zero

It includes robot joint zero setting, external joint zero setting, encoder zero clearing and joint position setting. As shown in Figure 9.4.

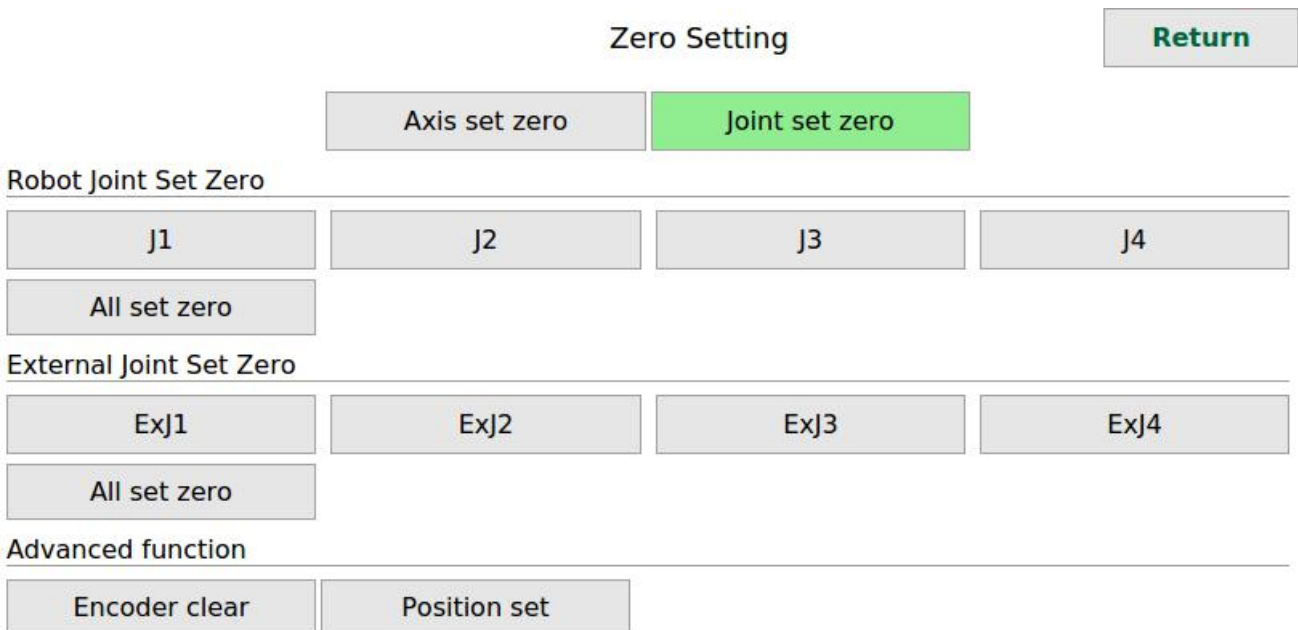


Figure 9.4 joint zero point setting interface

Robot joint zero setting

Single joint zero setting: such as clicking the button “J1”, set the current position of the robot axis “J1” to the “J1” zero;

Set all to zero: set the current position of all joint of the robot to the zero.

External joint zero setting

Single joint zero setting: such as clicking the button “ExJ1”, set the current position of the external axis “ExJ1” to the “ExJ1” zero;

Set all to zero: set the current position of all external joint to the zero.

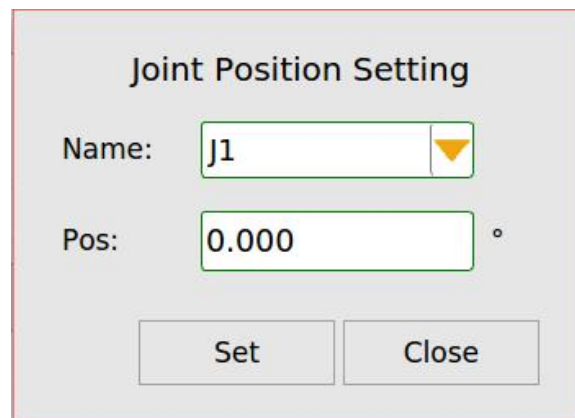
Encoder reset

The encoder code value is cleared.

Note: encoder zeroing is only for module encoder, and slave encoder does not have this function.

Joint position setting

Sets the current position of the specified joint to the specified angle.



Joint Position Setting

Name: J1

Pos: 0.000 °

Set Close

Figure 9.5 joint position setting

9.2 Filter

Filter types: mean and ZVD.

Mean has one parameter: filtering time; ZVD has two parameters: filtering time and damping rate. The interface is shown in Figure 10.4.

Note: the user can use the default parameters, if you need to modify it, please consult the supplier's technical staff.

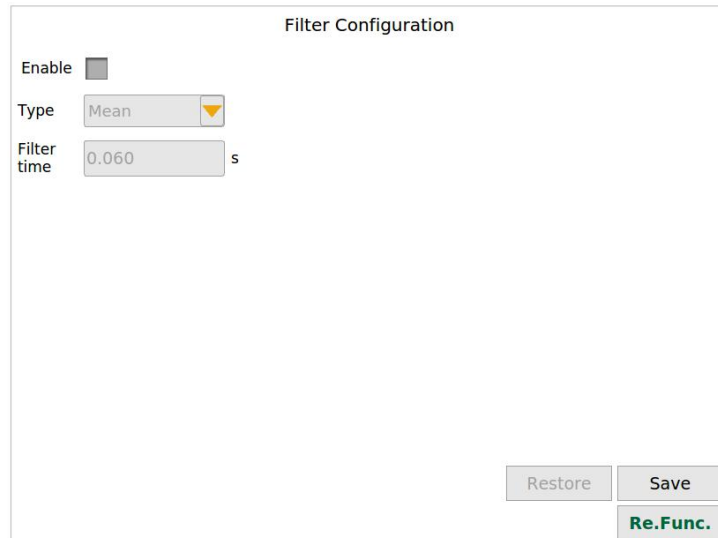


Figure 9.6 filter configuration interface

9.2.1 Restore Default Configuration

Restore default configuration: Restore the default filter configuration parameters of the system.

Note: the system default configuration itself cannot be modified, but it can be edited and saved. After saving, it can be used directly after enabling.

9.3 Regional Monitoring

Display area monitoring variables and configure the parameter values of area monitoring variables (refer to [WorkArea](#) variables for parameter details).

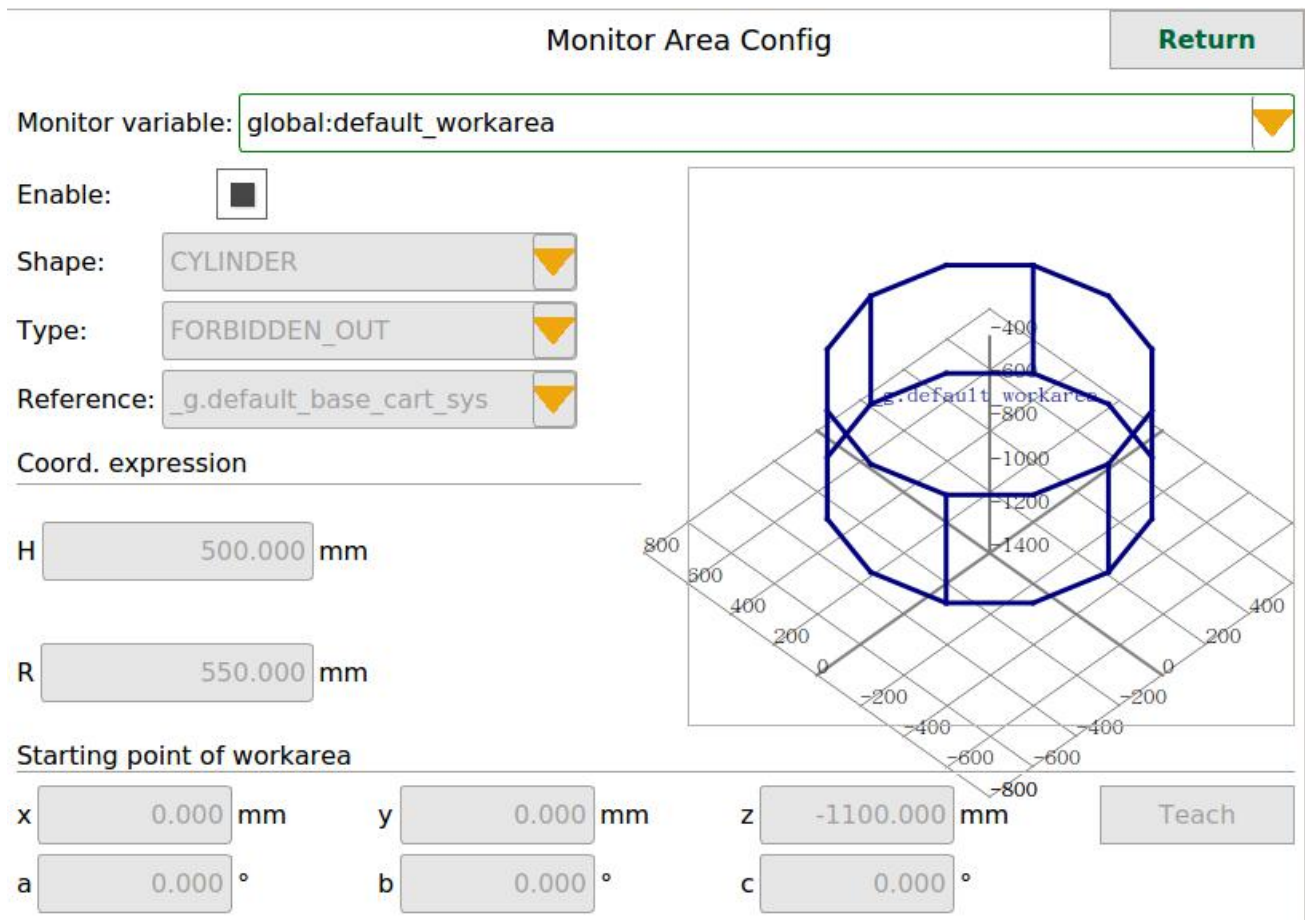


Figure 9.7 regional monitoring variable interface

9.4 PLC Control

Start PLC and stop PLC.

Current state: displays the current operating status of the PLC.

Startup and start: Configure whether the PLC is started.

Switch to plc interface: Click the button to switch to display the plc interface. If you want to return to the hmi interface, click the button to switch to hmi in the plc.

9.5 External Axis

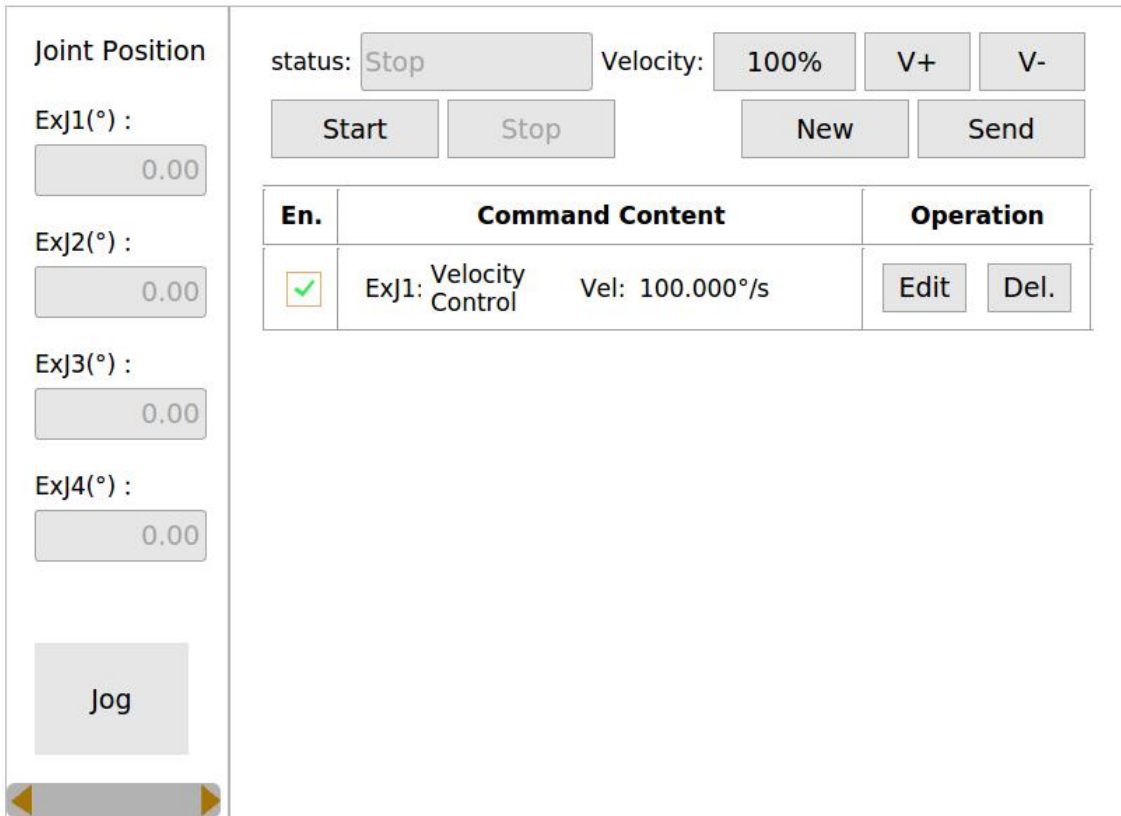


Figure 9.8 external axis interface

9.5.1 Jog

Click the "jog" button. When there are 4 external axes, the interface pops up, as shown in Figure 9.9, press the button and the axis starts to move, release the button and the axis stops moving, the axis movement speed is controlled by the speed of the external axis interface.



Figure 9.9 jog interface of external axis

9.5.2 New

Create a new external axis command.

9.5.3 Edit

Instruction Type: Speed Control

External shaft: Exj1

Relative Location:

Location: 0.000°

Speed: 0.000°/s

Acceleration: 0.000°/s²

Acceleration of acceleration: 0.000°/s³

Complete Dout: Y5

Trigger Conditions:

DIO: X1=True

Add

Edit

Delete

Stop condition:

DIO: X1=True

Add

Edit

Delete

Confirm Cancel

Figure 9.10 external shaft command editing interface

Instruction type

Location control: In the location control mode, the relative location check box is checked, and exj1 changes 10 degrees relative to the current location during movement; If the relative location check box is not checked, exj1 moves to a location of 10 degrees relative to the zero point of the shaft; The location can be set to a negative value.

speed control: Move according to the specified speed parameter.

Motion parameters

Motion parameter reference value: Speed (100), acceleration (1000) and acceleration (10000). The user can change the motion parameters as needed; The command running speed is controlled by the speed override of the external axis interface. If the speed override is modified to 100%, the speed still does not meet the requirements, and the motion parameters need to be adjusted appropriately .

Complete Dout

The output port will be set to true after the external axis instruction runs.

Run Dout

When the external axis command is running, the output port is set to true; when the external axis command is stopped, the output port is set to false.

9.5.4 External axis motion conditions and stop conditions

External axis motion conditions:

- (1) The command is enabled and sent successfully;
- (2) Click the “enable” and “start” button to make the external axis in the running and enabling state;
- (3) the trigger condition is true.

External axis stop condition: The command is disabled and sent successfully; Or move to the designated location; Or the stop condition holds; Or click the “stop” button.

9.5.5 Send

Click the "send" button, Send all external axis commands to the controller.If a new external axis command is created but not sent successfully,the new command will not take effect.

9.5.6 Instruction execution sequence description

For instructions controlling the same shaft, which instruction trigger condition is established first and which instruction is executed first, during the execution of this instruction, another instruction will not be executed after the trigger condition is established. Until the execution of the previous instruction is completed and the trigger condition of the other instruction mentioned above is still established, another instruction will be executed.

Commands controlling different axes do not affect each other.

9.6 TCP Monitor

Monitor the communication status corresponding to tcp.

Tcp Monitor Return

TCP list: ▼ Check data

Enable TCP:

TCP type: ▼

Port:

IP address:

Communication Status:

connection not enabled

Figure 9.11 TCP Monitor

TCP Communication List: Displays a list of loaded TCP communication variables.

Enable TCP: When checked, the connection is established; when unchecked, the connection is disconnected.

9.7 Tracking Parameters

It is used to configure the parameters related to tracking. There is a default configuration. The default configuration cannot be modified or deleted. The interface is shown in Figure 9.12. the user can use the default parameters.

Figure 9.12 default configuration interface of tracking parameters

9.7.1 Configuration Item Description

Configuration list: List of tracking parameter configuration names.

Current configuration: Displays the name of the current configuration.

Configuration Parameter

vel: Tracking speed, unit m/s;

acc: Tracking acceleration, unit m/s²;

jerk: Tracking acceleration, unit m/s³;

Tracking parameter type: PID and data;

PID parameters: Parameter 1, parameter 2, parameter 3.

9.7.2 Configuration Button Description

Delete button

Deletes the configuration displayed in the configuration list.

New button

Create a new configuration.

Save configuration button

Save the current configuration information.

9.8 Object Source Management

There are four kinds of object sources: camera, sensor, position change and virtual.

They have common parameters: filtering error and communication error.

Filter error: When two objects are within the error range, they are considered to be the same object, and the error value cannot be negative. The filter error value is generally the center distance of the two materials.

Communication Error: Error caused by communication.

9.8.1 Camera

The camera object source needs to be configured with communication mode and camera trigger mode.

Object Source Config Return

Config.List: pp:os ▼

Object source: Camera ▼ Enable: Visual Parm. Check

Comm. Error: 0.000 Filter error: 0.000

Comm. variable: _p.tcp_obj ▼ Enable

Comm. type: CLIENT ▼

IP: 192.168.100.101 Port: 4000

Trigger mode: Hard Trigger ▼

Trigger period: 100

Trigger Port: Y5 ▼

Restore
Save

Figure 9.13 object source configuration interface

9.8.1.1 Receive data communication configuration

Communication mode: At present, only TCP communication mode is supported, as shown in Figure 9.12.

Configure IP address and port number to receive object data sent by vision.

Comm. variable: _p.tcp_obj ▼ Enable

Comm. type: CLIENT ▼

IP: 192.168.100.101 Port: 4000

Figure 9.14 TCP communication configuration interface

Communication format: [X:703.17;Y:515.32;Z:-670.32;A:4.40;ATTR:-1;ID:6;INFO:Object], X、Y、Z、A is double type data, ATTR、ID is int type data, INFO is string type data(**INFO information cannot include : ; []**).

Among them, XY is the data that must be sent, others can be freely selected. such as [X:703.17;Y:515.32].

Support for scalable data: use : ; separate, if the test attribute is extended, the value is aa, the format is [X:703.17;Y:515.32;test:aa], The name and value of the extended data attribute cannot include : ; [].

IP: It is usually set to 192.168.100. X. the IP is the same as that of the visual device. It must be ensured that the X network segment cannot conflict with other devices.

Note: The controller is the client, the visual is the server, and the tcp communication is enabled. When the tcp communication is established, the data can be received. If the enable is turned off, the connection with the server will be disconnected..

9.8.1.2 Trigger mode

■ Network trigger

Network trigger: TCP communication mode triggers camera to take pictures;

Enable: enable enable, the connection of tcp communication is established; disable enable, the connection of tcp communication is disconnected;

Trigger cycle: The time period of periodically triggering the camera, in MS;

IP address and port number: TCP communication requires configuring the IP and port number for visual communication.

When the object source is enabled, it will actively establish communication with the vision system. After successful communication, it will periodically send trigger information to the vision to trigger the camera to take pictures.

Trigger mode: Network Trigger

Trigger period: 100

Comm. variable: _p.tcp_trigger Enable

Comm. type: CLIENT

IP: 192.168.100.102 Port: 4000

Trigger Info.: trigger

Figure 9.15 network trigger configuration interface

Note: generally, the IP is set to 192.168.100. X. it must be ensured that the X network segment cannot conflict with other devices.

■ Hard trigger:

The actual physical output IO triggers the camera to take pictures.

Trigger port: Used to trigger the output io of the camera.

Trigger cycle: Time period of periodically triggering the camera, unit: ms

When the object source is enabled, the output signal corresponding to the trigger port will be set to true periodically to trigger the camera to take pictures.

Trigger mode: Hard Trigger

Trigger period: 100

Trigger Port: Y5

Figure 9.16 hard trigger configuration interface

9.8.1.3 Visual parameter configuration

Camera Parm. config

Center x0 (pixels):	0.000	<input type="checkbox"/> a inverse:	<input type="checkbox"/> x and y swap
Center y0 (pixels):	0.000	Rotation angle(°):	0.000
x pixel ratio (pixels/mm):	1.000	x offset (mm):	0.000
y pixel ratio (pixels/mm):	1.000	y offset (mm):	0.000
<input type="checkbox"/> x reverse	<input type="checkbox"/> y reverse	a offset(°):	0.000

Instructions for use: When calculating camera parameters, calculate in the following order

1. Subtract x0, y0 from the original data
2. Divide the xy coordinates by the pixel ratio, and negate the coordinates that need to be reversed
3. Swap xy values that need to swap xy coordinates
4. Rotate the angle around the current origin, compensate the rotation angle
5. Translate the rotation offset from the current position to compensate for the object position

Figure 9.17 hard trigger configuration interface

Parameter Description:

- (1) x and y swap, corresponding to “type” in formula;
- (2) Center x0,y0, unit(pixel), corresponding to “x0 y0” in formula;
- (3) Rotation Angle, corresponding to “a” in formula;
- (4) xy offset, corresponding to “dx dy” in formula.

```

Formula:double x1 = ((type ? y_camera : x_camera) - x0) / kx;
double y1 = ((type ? x_camera : y_camera) - y0) / ky;
double x_con = x1 * cos(a * kPi / 180.0) - y1 * sin(a * kPi / 180.0);
double y_con = x1 * sin(a * kPi / 180.0) + y1 * cos(a * kPi / 180.0);
double a_con = a_camera + a * kPi / 180.0;
x_out = x_con + dx * cos(a_con) - dy * sin(a_con);
y_out = y_con + dx * sin(a_con) + dy * cos(a_con);
a_out = a_con + da * kPi / 180.0;

```

In Formula kx,ky =x,y reverse * xy Pixel Ratio.

9.8.2 Sensor

Configure the physical input port to which the sensor is connected. Specify the id and attr of the object generated by the sensor.

Figure 9.18 sensor configuration interface

9.8.3 Position Change

Position change refers to the generation of objects by different position changes. Specify the id and attr of the generated object.

Application scenario: The conveyor is controlled by an external shaft, and objects move back and forth on the conveyor.

Figure 9.19 location change configuration interface

9.8.4 Virtual object source

Generate virtual objects.

Figure 9.20 Virtual object source configuration interface

Virtual Object Parameters Meaning

Mode: Virtual mode, include Time and Distance.

Time: Generate objects by time;

Distance: Generate objects by position.

Each of the remaining parameters has a minimum and maximum value.

Each parameter is generated randomly between between minimum and maximum values when virtualized.

T: Virtual object interval time, effective when Mode is Time

D: Virtual object interval distance, effective when Mode is Distance

X, Y, A: The X coordinate, Y coordinate, rotation angle A of the object in the object source coordinate system

id: Generate the object ID

attr: Generate the object attr

9.9 Object Allot

Application scenario: It is applicable to the scene where tracking leakage cannot be accepted. If the incoming materials are dense, it can be considered to control the start and stop of the conveyor.

Object Allot Config Return

Config.List:

Enable allot: check status

Tcp variable: Enable tcp

Tcp type:

Port:

Object id:

Trigger IO:

Allot type:

Allot ratio:

Restore
Save

Figure 9.21 object Allot configuration interface

Configuration List: A list of object shunt configuration names.

Enable diversion: If checked, diversion is enabled; otherwise, diversion is not enabled.

View Status: View the connection status of the tcp between the current device and the offloaded device.

tcp variable: The variable name used for offloaded tcp communication.

Enable tcp: Select to establish tcp communication with the offloading device; otherwise, disconnect tcp communication.

tcp type: server and client, as the server sends object data to its corresponding client.

Port number: The port number of TCP communication between this machine and the next machine. After successful configuration, object data will be sent to the next machine.

Object id: The id of the object to be shunted.

Trigger IO: Trigger the signal of shunting action, and the rising edge is monitored to send all object data to be shunted to the next machine, if there is no trigger signal, the object will not send out.

Diversion type: RATIO: Divide proportionally (0-1); MAXIMUM: Divide by maximum capacity; RATIO_ MAXIMUM: Divide unfinished on a proportional basis; GROUPING: Divide by group.

9.10 Status Monitoring

Monitor the status of robot shaft, external shaft and unmapped shaft, enabling status, status word, control word, current code value read from driver, the target code value is the command value sent by the controller to the driver.

Return

Status Monitor

Robot Axis

External Axis

Unmapped Axis

Axis name:	<input type="text" value="D1"/>	<input type="checkbox"/> Enable	<input type="checkbox"/> Report an error	
Curr. encoder count:	<input type="text" value="23549"/>	Control word:	<input type="text" value="0"/>	
Target encoder count:	<input type="text" value="23549"/>	Status word:	<input type="text" value="0"/>	
Error code:	<input type="text" value="0"/>	Control mode:	<input type="text" value="0"/>	
Error message:	<input style="width: 100%;" type="text"/>			

Axis name:	<input type="text" value="D2"/>	<input type="checkbox"/> Enable	<input type="checkbox"/> Report an error	
Curr. encoder count:	<input type="text" value="-12538"/>	Control word:	<input type="text" value="0"/>	
Target encoder count:	<input type="text" value="-12538"/>	Status word:	<input type="text" value="0"/>	
Error code:	<input type="text" value="0"/>	Control mode:	<input type="text" value="0"/>	
Error message:	<input style="width: 100%;" type="text"/>			

Figure 9.22 status monitoring interface

9.11 Handwheel Function

Provide handwheel control function in six shaft mode.

Handwheel

Speed Ratio:	<input type="text" value="x0"/>	Current move code:	<input type="text" value="0"/> inc
Move direction:	<input type="text" value="--"/>	Current move distance:	<input type="text" value="0.000000"/> °
Status:	<input type="text" value="Stop"/>		

Figure 9.22 function interface of hand wheel status

current move code:Each time the handwheel is operated to control the code value to move in a certain direction.

Current move distance:current move code * resolution.

Handwheel

<input type="checkbox"/> Activate	<input type="checkbox"/> Encoder Code Value Reversed
Encoder: <input type="text" value=""/> ▼	x Direction: <input type="text" value="X5"/> ▼ <input type="button" value="false"/>
Angular Res.: <input type="text" value="0"/> "	y Direction: <input type="text" value=""/> ▼ <input type="button" value=""/>
Linear Res.: <input type="text" value="0"/> um	z Direction: <input type="text" value=""/> ▼ <input type="button" value=""/>
Response delay: <input type="text" value="0"/> ms	a Direction: <input type="text" value=""/> ▼ <input type="button" value=""/>
x1 times: <input type="text" value=""/> ▼ <input type="button" value=""/>	b Direction: <input type="text" value=""/> ▼ <input type="button" value=""/>
x10 times: <input type="text" value=""/> ▼ <input type="button" value=""/>	c Direction: <input type="text" value=""/> ▼ <input type="button" value=""/>
x100 times: <input type="text" value=""/> ▼ <input type="button" value=""/>	

Figure 9.24 function interface of hand wheel configuration

angle / line resolution:The actual distance/angle change corresponding to the encoder code value change 1 at x1 magnification, set according to user requirements;

response delay:If the response delay is 1ms, then the running state is set to stop state 1ms after the handwheel stops.

pose direction (x, y, z, a, b, c):To configure and control the input signals corresponding to each direction, refer to the hardware configuration for details.

magnification selection (x1, x10, x100):To configure the input signal corresponding to the speed multiplier, refer to the hardware configuration for details.

9.12 Variable Cache Configuration

Provides the data and custom name functions of the cache.

Index	Data	Custom Name
1	<input type="text"/>	<input type="text"/>
2	<input type="text"/>	<input type="text"/>
3	<input type="text"/>	<input type="text"/>
4	<input type="text"/>	<input type="text"/>
5	<input type="text"/>	<input type="text"/>
6	<input type="text"/>	<input type="text"/>
7	<input type="text"/>	<input type="text"/>
8	<input type="text"/>	<input type="text"/>
9	<input type="text"/>	<input type="text"/>
10	<input type="text"/>	<input type="text"/>

Figure 9.25 variable buffer configuration interface

This function can be used in conjunction with the GetCacheString command to realize the function of changing variables during operation. A total of 64 caches are supported.

9.13 Yield

Provides the use of variables to achieve the function of recording output. Variables must use global default product variable.

The image shows a software interface titled "Production Config". At the top right is a "Return" button. Below the title, there is a checked checkbox labeled "Enabled". Underneath, the text "Record variable:" is followed by a text input field containing "_g.default_product" and a dropdown arrow on the right. At the bottom right of the interface is a "Save" button.

Figure 9.26 variable buffer configuration interface

9.14 Drag and teach

9.14.1 Point teach

Use drag mode to teach variables. Variable types that support teaching are TcpPosition, JointPosition, CartSys, and the specified index in ARRAY OF TcpPosition.

DragTeach

Teaching method:

Current Teach Pos.

x:	<input type="text" value="0.000"/>	mm
y:	<input type="text" value="0.000"/>	mm
z:	<input type="text" value="0.000"/>	mm
a:	<input type="text" value="0.000"/>	°
b:	<input type="text" value="0.000"/>	°
c:	<input type="text" value="0.000"/>	°

Type:	<input type="text" value="TcpPosition"/>
Variable:	<input type="text" value="_g.fg"/>
Coord.:	<input type="text" value="_g.default_base_cart_sys"/>
Tool:	<input type="text" value="_g.default_tool"/>
<input type="button" value="Write Variable"/>	

Figure 9.27 Point teach

Teaching steps:

- (1) Click the "Drag Mode" button to switch to drag mode;
- (2) Drag the robot to the designated position;
- (3) Click "Write Variable".

9.14.2 Trajectory teaching

Use drag mode to teach a trajectory. The variable types that support teaching are ARRAY OF TcpPosition. When dragging, specify the collection period to record a position point, and the position points smaller than the filtering error will be filtered out;

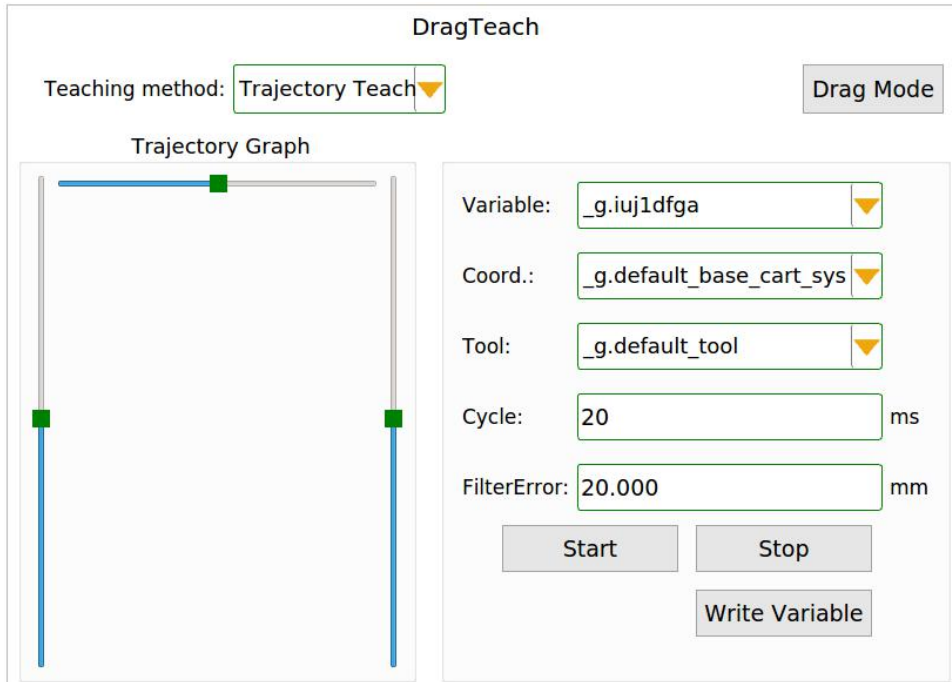


Figure 9.28 Trajectory teaching

Teaching steps:

- (1) Click the "drag mode" button to switch to drag mode;
- (2) Click the "Start" button to start dragging, and drag the robot to move according to the required trajectory;
- (3) Click the "Stop" button to stop dragging, and click "Write Variable".

9.15 Watchdog

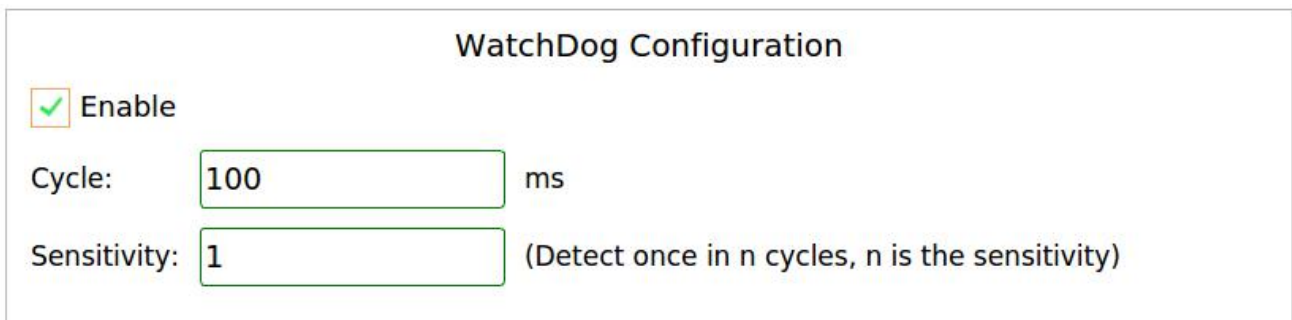


Figure 9.29 watch dog

Set as shown in Figure 9.28. After enabling, a heartbeat detection will be performed every 100ms. If no information is received for 30 consecutive cycles, the connection will be actively disconnected, and the user needs to check the network connection.

Chapter10 Control Power

Control power support: HMI, IO control, Modbus TCP and TCP.

10.1 IO Control

For input / output IO configuration and switching loader configuration, select IO control in the function block interface to enter the configuration interface.

10.1.1 Input / Output Configuration

Configure the input IO and output io of external mode, and display the value of current IO as true or false in real time. As shown in Figure 10.1.

Return

IO Config
Prog. Config

Input IO Config.

Enable:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>	Start:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>	<input type="checkbox"/> All Control Power Effective
E-stop:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>	Pause:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>	
Return home:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>	Stop:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>	
Clear error:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>	Shutdown:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>	

Note: [Emergency stop configuration] is not affected by the [All control rights effective] option, it is valid for all control rights!
 When enabling configuration IO control, it is synchronized with IO, and the rising edge switching of other

Output IO Config.

Fault signal:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>	Stop/Run:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>
Enable signal:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>	Pause signal:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>
Zero signal:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>	Start finished signal:	<input style="width: 90%;" type="text"/> ▼ <input style="width: 5%;" type="checkbox"/>

Note: [Output IO configuration function] is valid not only for IO control, but for all control power!

Restore
Save

Figure 10.1 input / output configuration interface

Enable: If the input IO of the lower enable is true, the robot will be enabled and the output IO of the enable signal will be true; conversely, if the input IO of the lower enable is false, the output IO of the lower enable and the enable signal will be false.

Emergency stop: when the input IO of emergency stop is pressed and true, the robot is in emergency stop state and the output IO of is true; On the contrary, when the input IO of emergency stop is pressed and false, the robot is in a non-emergency stop state.

Return Home: when the is enabled on the and in the non running state, when the input IO to return to zero is pressed, the robot returns to the zero location, and the output io of is true.

Clear Error: when there is alarm information and the input IO of error clearing is pressed, the output IO of is false after all alarms are cleared. If there are still alarms not cleared, the output IO of is still true.

Start program: in the enabled state, when the input IO of the startup program is pressed, the robot runs according to the loaded program, and the output IO of the is true.

Pause program: in the running state, when the input IO of the pause program is pressed, the robot pauses, and the output IO is true.

Stop program: in the running state, when the input IO of the stop program is pressed, the robot stops, and the output IO of the signal is false.

Shutdown: When the input IO of shutdown is pressed and it is true, the controller is turned off.

All control rights take effect: If the check box is checked, the input IO configuration takes effect under any control rights.

10.1.2 Loader Configuration

Configure the switching signal and switching success signal of the specified program.

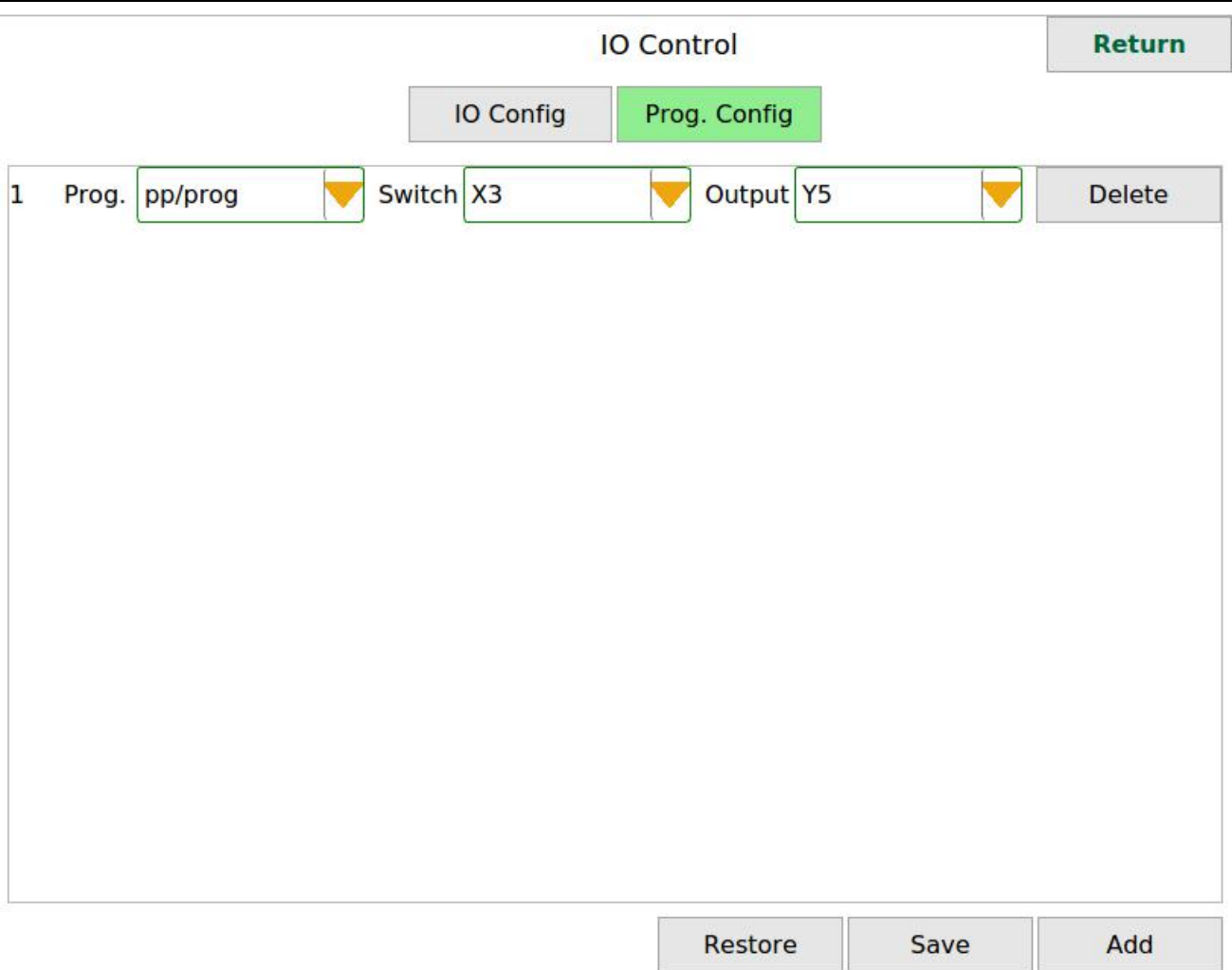


Figure 10.2 loader configuration interface

Switching signal: When the input IO is true, load the corresponding program.

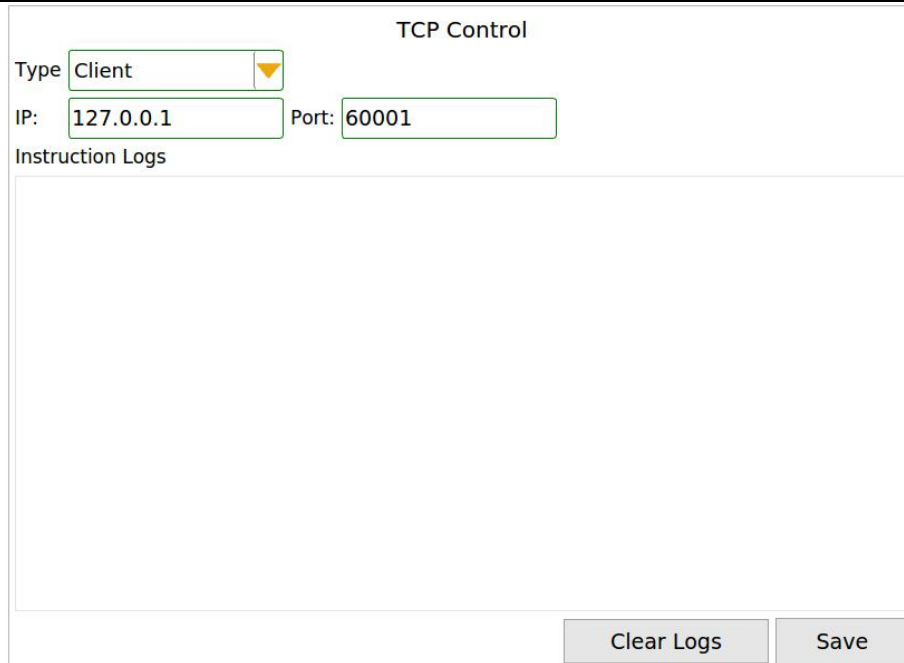
Output signal: When the program switching is successful, the output IO is true.

10.2 TCP Control

TCP control mode, the controller supports control through TCP communication. The requester of the control protocol can be either a TCP client or a TCP server.

10.2.1 TCP Communication Configuration

Select TCP control in the function block interface to enter the configuration interface.



The screenshot shows a web-based control interface for TCP communication. The title is "TCP Control". There are three input fields: "Type" (a dropdown menu showing "Client"), "IP:" (a text box containing "127.0.0.1"), and "Port:" (a text box containing "60001"). Below these is a large empty rectangular area labeled "Instruction Logs". At the bottom right, there are two buttons: "Clear Logs" and "Save".

Figure 10.3 TCP control interface

IP address: The IP address of the remote device. If it is local TCP communication, the IP address is 127.0.0.1.

Port number: The port number of TCP communication is the same as that of the remote device.

10.2.2 TCP Communication Protocol

JSON binary header: please refer to "TCP Control Protocol Version 1.0.0" for the control protocol.

Simple string: Please refer to "TCP Control Protocol (Simple String Version)" for the control protocol.

Byte: Please refer to "TCP Control Protocol (Simple Binary Version)" for the control protocol.

10.3 ModbusTcp Control

10.3.1 Introduction

Port: 50212;

All register addresses and numbers are parsed according to unsigned short;

All data are transferred and parsed in big-endian mode.

10.3.2 Packet Format

The robot control system serves as the server of TCP protocol. The client program serves as the client of TCP protocol.

Each data interaction adopts a fixed data packet format, which is composed of three parts: MBAP, function code and data.

MBAP

String	Length	Describe	Client	Server (Control System)
Transaction Identifier	2 Bytes	Uniquely identifies a packet	Specified by client	The server will copy this string and return it to the client, and the customer service side will judge whether the response is a response to a request according to this string. This string should be different each time the client sends a request.
Protocol Identifier	2 Bytes	Protocol ID, fixed to 0		
Length	2 Bytes	The number of bytes of all subsequent data in the entire packet	The client needs to calculate the length	The server recalculates the length based on the response
Unit Identifier	1 Bytes	Fixed to 0		

Function Code

The function code consists of 1 byte. The definition of each function code is shown below.

Data

The data sent is different according to different function codes. The data format of user-defined function code is described in the chapter "description of extended function code for realizing robot related functions".

◆Read coil (Digital output IO)

Function code and data of request packet:

Function code	1 Byte	0x01
Start address	2 Bytes	0x0000 to 0xFFFF
Number of coils	2 Bytes	1 to 2000(0x7D0)

Function code and data of response packet:

Function code	1 Byte	0x01
Number of bytes of subsequent data	1 Bytes	The value is the number of query coils divided by 8. If it cannot be divided, add 1
Coil status	n Bytes	Every 8 coil states are represented by 1 byte. Less than 8 also occupy 1 byte. The coil state of the low address is in the low significant bit of the byte, and the coil state of the high address is in the high significant bit of the byte

Function code and data of response packet in case of error:

Function code	1 Byte	0x01+ 0x80
Exception code	1 Bytes	0x01 or 0x02 or 0x03 or 0x04

◆Write coil (Digital output IO)

Function code and data of request packet:

Function code	1 Byte	0x0F
Start address	2 Bytes	0x0000 to 0xFFFF
Number of coils	2 Bytes	1 to 2000(0x7D0)
Number of subsequent bytes	1 Bytes	Calculation method: same as read coil
Coil status	n Bytes	Calculation method: same as read coil

Function code and data of response packet:

Function code	1 Byte	0x0F
Start address	2 Bytes	0x0000 to 0xFFFF
Number of coils	2 Bytes	1 to 2000(0x7D0)

Function code and data of response packet in case of error:

Function code	1 Byte	0x0F + 0x80
---------------	--------	-------------

Exception code	1 Bytes	0x01 or 0x02 or 0x03 or 0x04
----------------	---------	------------------------------

◆Read discrete input (Digital input IO)

Function code and data of request packet:

Function code	1 Byte	0x02
Start address	2 Bytes	0x0000 to 0xFFFF
Number of discrete inputs	2 Bytes	1 to 2000(0x7D0)

Function code and data of response packet:

Function code	1 Byte	0x02
Number of bytes of subsequent data	1 Bytes	Check the number of discrete inputs divided by 8. If it cannot be divided, add 1
Discrete input state	n Bytes	Every 8 discrete input states are represented by 1 byte. Less than 8 also occupy 1 byte

Function code and data of response packet in case of error:

Function code	1 Byte	0x02+0x80
Exception code	1 Bytes	0x01 or 0x02 or 0x03 or 0x04

◆Read register

Function code and data of request packet

Function code	1 Byte	0x03
Start address	2 Bytes	0x0000 to 0xFFFF
Number of registers	2 Bytes	1 to 125(0x7D)

Function code and data of response packet

Function code	1 Byte	0x03
Number of bytes of subsequent data	1 Bytes	2xN

Register value	2xN Bytes (n is the number of registers)	The value of a register, each of which occupies two bytes
----------------	---	---

Function code and data of response packet in case of error:

Function code	1 Byte	0x03+0x80
Exception code	1 Bytes	0x01 or 0x02 or 0x03 or 0x04

◆Write multiple registers

Function code and data of request packet

Function code	1 Byte	0x10
Start address	2 Bytes	0x0000 to 0xFFFF
Number of registers	2 Bytes	0x01 to 0x7B
Number of subsequent bytes	1 Bytes	The value is 2xn (n is the number of registers)
Register value	2xN Bytes	Set the value of the register. Each register occupies two bytes

Function code and data of response packet

Function code	1 Byte	0x10
Start address	2 Bytes	0x0000 to 0xFFFF
Number of registers	2 Bytes	0x01 to 0x7B

Function code and data of the response packet when an error occurs

Function code	1 Byte	0x10+0x80
Exception code	1 Bytes	0x01 or 0x02 or 0x03 or 0x04

10.3.3 Robot Related Functions

■ Extended function code description

65-72, 100-110 function codes can be customized.

The error response is returned in accordance with the Modbus

standard when an error occurs. When the operation is successful, only the function code is returned, and no data is returned.

Startup program: custom function code 65, no interactive data is required.

Stop program: custom function code 66, no interactive data is required.

Pause program: custom function code 67, no interactive data is required.

Return to zero: custom function code 68 does not require interactive data. It is not supported temporarily because it cannot be stopped

Set to zero: custom function code 69 needs to send a 2-byte unsigned integer. Indicates the index of the shaft to set the zero point, starting from 0.

Error clearing: custom function code 70 does not require interactive data.

Start jog: customize function code 71 and send index and direction information. Index represents the joint number or coordinate shaft. It starts from 0 and is represented by 2 bytes. It is parsed as an unsigned integer. Direction represents the direction, which is represented by 2 bytes. It is parsed as an unsigned integer. 0 represents the reverse direction and 1 represents the forward direction. Use Modbus TCP to switch the jog coordinate system.

Stop jog: custom function code 72, no interactive data required.

Upper enable: custom function code 100, no interactive data required.

Lower enable: custom function code 101, no interactive data required.

Set the speed percentage: customize the function code 102 and send a 2-byte unsigned integer. The minimum speed percentage is 1%. 0.1% is not considered.

Set single step continuous mode: Customize function code 103 and send 2-byte unsigned integer. 1 indicates single step mode and 2 indicates continuous mode.

Set the operation mode: the user-defined function code 104 needs to send a 2-byte unsigned integer. 1 indicates manual mode and 2 indicates automatic mode.

Switching program: the user-defined function code 105 needs to send 50 bytes of data. Each byte represents an ASCII code and needs to end with 0. The string represents the program name. Switches the current program to the program specified by the string.

Switch the jog coordinate system: the user-defined function code 106 needs to send 50 bytes of data, and each byte represents an ASCII code, which needs to end with 0. The string represents the coordinate system variable name. Switches the current coordinate system to the coordinate system specified by the string. Four coordinate systems are supported: world coordinate system, base coordinate system, joint coordinate system and shaft coordinate system. The string corresponding to the four coordinate systems is the world coordinate system: `_g.default_world_cart_Sys`, the corresponding index is 1. Base coordinate system: `_g.default_base_cart_Sys`, the corresponding index is 2. Joint coordinate system: `_Joint.sys`, the corresponding index is 3. Shaft coordinate system: `_Shaft.sys`, the corresponding index is 4. The jog coordinate system switched by MODBUS is only effective for Modbus control. Currently, only coordinate system switching through index is supported.

■ Coil, Discrete input address description

The coil and discrete inputs correspond to the actual digital physical quantity IO. The address is consistent with the drawing of electrical design. For example, when operating Y1, the physical output IO, the register address corresponds to 1. The same applies to input io.

Addresses up to 128 (not included) are for actual physical io use. addresses 128 and above are for virtual io use.

■ Input register address description

Because the current robot system does not provide analog support, addresses within 1000 are not used temporarily and are not supported. Addresses of 1000 and above are used to provide robot status query.

Read the following registers through the standard Modbus protocol function code to query the corresponding status of the robot.

◆Running status register

Register address: 1000

Number of registers: 1

Register data format: 2-byte unsigned integer, corresponding to unsigned short of C++. 1 indicates the running state, 2 indicates the suspended state, 3 indicates the stopped state, and 4 indicates the stopped state.

◆Error code register

Register address: 1001

Number of registers: 1

Register data format: 2-byte unsigned integer, corresponding to unsigned short of C++. 0 means there are no errors.

◆Enable register

Register address: 1002

Number of registers: 1

Register data format: 2-byte unsigned integer, corresponding to unsigned short of C++. 0 means not enabled and 1 means enabled.

◆Speed percentage register

Register address: 1003

Number of registers: 1

Register data format: 2-byte unsigned integer, corresponding to unsigned short of C++.

◆Single step continuous mode register

Register address: 1004

Number of registers: 1

Register data format: 2-byte unsigned integer, corresponding to unsigned short of C++. 1 indicates single step mode and 2 indicates continuous mode.

◆Operation mode register

Register address: 1005

Number of registers: 1

Register data format: 2-byte unsigned integer, corresponding to unsigned short of C++. 1 indicates manual mode and 2 indicates automatic mode.

◆Joint location information register

Register address: 1006

Number of registers: 12

Register data format: 12 registers have 24 bytes in total, and every 4 bytes represents a double type. Therefore, a total of 6 double type data represent the values of J1, J2, J3, J4, J5 and J6 in turn.

◆Cartesian coordinate system location information register

Register address: 1018

Number of registers: 12

Register data format: 12 registers have 24 bytes in total, and every 4 bytes represents a double type. Therefore, a total of 6 double type data represent the values of X, y, Z, a, B and C in turn.

◆Current program register

Register address: 1030

Number of registers: 25

Register data format: 25 registers, 50 bytes in total, can represent 49 ASCII characters at most, and the last string is 0. Used to get the name of the currently open program.

◆Current jog coordinate system register

Register address: 1055

Number of registers: 25

Register data format: 25 registers, 50 bytes in total, can represent 49 ASCII characters at most, and the last string is 0. Used to obtain the name of the current jog coordinate system.

■ Holding register address description

The address within 1000 (excluding 1000) is used to read or set the actual analog quantity io. Because the current robot system does not provide analog quantity support, it is not supported temporarily.

1000 and above addresses are used to provide robot shared variables.

■ Variable Shared Properties

Variable sharing means that under the control of modbus, the variable value of the selected variable can be updated to the control end, and the variable value can be modified at the control end and synchronized to the controller.

Note: when the manipulator is running, the change of variable value at the control end does not take effect.

■ Shared variable description

The operation of reading and writing shared variables is to read and write the holding register, and the corresponding address is 1000 and above.

Variables can only be written in a non running state.

Several known actions may reset the value of variables, including loading a project, loading a program, and switching a program. Therefore, when these actions are performed after writing variables, the written variable values will be overwritten.

■ Supported shared variables

Variable type	Total bytes	Attribute name	Offset address	Number of attribute bytes
INT	4	-	-	-
REAL	4	-	-	-
BOOL	2	-	-	-
JointPosition	24	a1	0	4
		a2	2	4
		a3	4	4
		a4	6	4
		a5	8	4
		a6	10	4
TcpPosition	24	x	0	4
		y	2	4
		z	4	4
		a	6	4
		b	8	4
		c	10	4
JointDistance	24	a1	0	4
		a2	2	4
		a3	4	4
		a4	6	4
		a5	8	4

		a6	10	4
TcpDistance	24	x	0	4
		y	2	4
		z	4	4
		a	6	4
		b	8	4
		c	10	4
Tool	24	x	0	4
		y	2	4
		z	4	4
		a	6	4
		b	8	4
		x	10	4
Dyn	48	vel_axis	0	4
		acc_axis	2	4
		dec_axis	4	4
		jerk_axis	6	4
		vel	8	4
		acc	10	4
		dec	12	4
		jerk	14	4
		vel_ori	16	4
		acc_ori	18	4
dec_ori	20	4		

		jerk_ori	22	4
Transition	8	trans_type	0	4
		trans_pos	2	4

10.3.4 Shared variable operation

In the Modbus module of the function block, you can add and delete shared variables and view the address of shared variables, which is easy to configure on the control side, as shown in the following figure.

Modbus Control
Return

Name	Addr.
▼ Global variable	
default_product(INT)	1000
▼ pp	
pd(INT)	1002
prog-dyn_max(Dynamic)	1004
prog-dyn_max(Dynamic)	1004
prog-dyn_max(Dynamic)	1028
prog-dyn_max(Dynamic)	1004
prog-dyn_max(Dynamic)	1028
prog-pos_2_h(TcpPosition)	1052

Add
Delete
Recover
Save

Figure 10.4 Modbus controls shared variable operations

- "Add" : Click "Add" to jump to the variable interface, select any variable as the shared variable, click "Save" button to set successfully and automatically assign the address.
- "Delete" : Select a variable and click "Delete" to cancel the shared variable attribute. If you click on a project or global variable, cancel the shared variable attribute for variables under that item.

- Restore: restore to the last saved state.

Chapter11 Alarm Management

This interface includes current alarm and historical alarm.

11.1 Current Alarms

Display the current alarm information, as shown in Figure 11.1.

Code	Time	Level	Description
● 6111	2021-10-14 13:42:43	Warning	When the motor is under enable down sta...
⊗ 4121	2021-10-14 13:34:32	Error	tcp communication failure 127.0.0.1:3000
⊗ 4433	2021-10-14 13:34:32	Error	_g.dsf variable does not choose configurati...

When the motor is under enable down state, it cannot respond to the "start" command

Display Delete Delete All

Figure 11.1 current alarms

Button description

Display: filter and display the current alarm information. All alarms, serious errors, general errors, warnings and messages can be displayed.

Delete: delete an alarm message.

Delete All: delete all alarms.

11.2 Historical Alarms

Display all historical alarm information, as shown in Figure 11.2.

Code	Time	Level	Description
✘ 4111	2021-08-03 14:57:55	Error	The emergency stop button was pressed!
🐛 2132	2021-08-03 14:56:03	Fatal e...	Exception information: The 7 command li...
✘ 4121	2021-08-02 19:14:01	Error	tcp communication failure The communi...

<	<	Page 0	>	>	Display
---	---	--------	---	---	---------

Figure 11.2 historical alarm interface

Button Explanation

Display: Filter and display historical alarm information. All alarms, serious errors, general errors, warnings and messages can be displayed.

|< and > |: Switch between page 0 and the last page.

< and >: Switch between next page and previous page.

Alarm prompt box: Select an alarm message to display the alarm message in the prompt box.

Chapter12 System Interface

12.1 User Management Interface

User password modification settings.

Authorized users can change the password of low authorized users and need to enter the original password.

User Manage	Setting	Import / Export	Version
User	Level	Status	Edit
Administrator	0	Logged	Change Password
Adjustor	1	- -	Change Password
Operator	2	- -	Change Password

Figure 12.1 user management interface

12.2 Setting

It includes shutdown, system update, system time setting, screen lock, automatic login setting, MacLicense configuration, controller IP setting, connection, disconnection, Control power, etc. As shown in Figure 12.2.

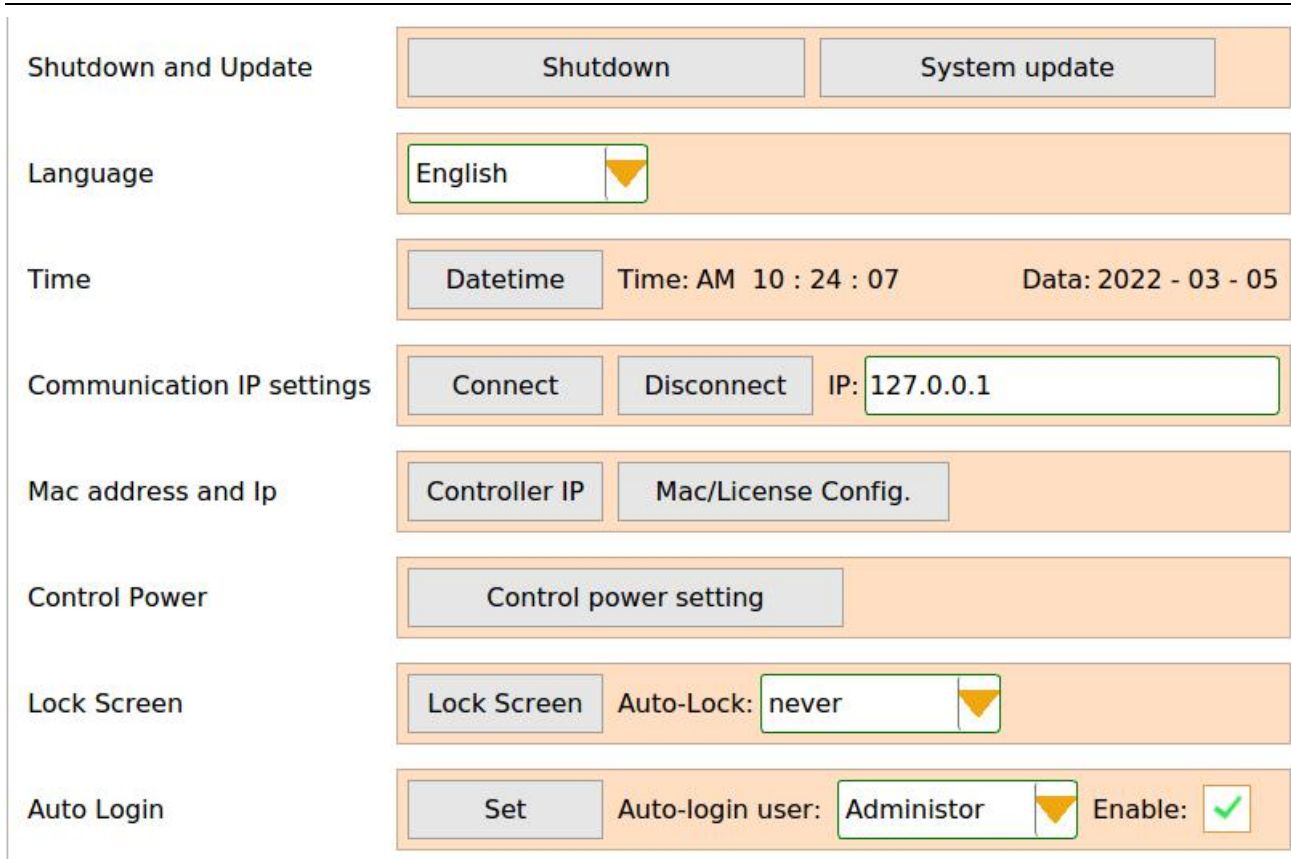


Figure 12.2 Setting interface

12.2.1 Shut Down

Click the shutdown button to restart the system or shut down.

Note: shut down the control cabinet before disconnecting the main power, otherwise the system may be damaged.

12.2.2 Update System

Step

(1) Insert the USB and click the "update system" button to pop up the dialog box for selecting the installation package *.tar.gz;

(2) Select the installation package and click "confirm" to pop up the dialog box of successful import of installation package;

(3) Click the "Confirm" button to enter the update system page, click the "Update System" button, and a progress bar will indicate the update status.

(4) Wait until the update is complete, and a pop-up prompt appears.

Note: the imported installation package name must be *installpackage.tar.gz*.

12.2.3 Modification Date and Time

Modify the system time.

12.2.4 Communication IP settings

IP Address: Set the IP address for TCP communication with the control system to be connected. Enter the IP address and click the "Connect" button, the current IP address will be saved and the connection will be created.

Restart the controller configuration: If the connection time is more than 10s and the connection is still not successful, the "Restart the controller configuration" button will appear, the controller configuration will be restarted, and the connection will be established.



Figure 12.3 Communication ip setting interface

12.2.5 Lock Screen

The user can set the screen lock time by himself. By default, never locks the screen.

If the screen lock time is set to 10s, and the interface is not operated for 10s, the screen is locked, and click the screen to wake up.

12.2.6 Controller IP settings

IP address: Set the IP address of the current control system network port. Enter the IP address and click the "Set" button.

Note: The IP address of each network port must not conflict.

12.2.7 MacLicense Setting

Click "MAC / license configuration" to pop up the interface, as shown in Figure 12.3.

Figure 12.4 MacLicense

MAC address used: MAC address specified by the current device.

License used: License matching MAC address.

Available MAC addresses: List of all MAC addresses for the device.

License: Select the license that matches the MAC address from the drop-down list.

Instructions: Explain how to set the MAC license.

12.2.8 Control Power Setting

Control power support: HMI, IO control, Modbus TCP, TCP. When switching control, select the control you want to switch and click “Confirm”.

Click the "Configure" button to jump to the corresponding configuration page.

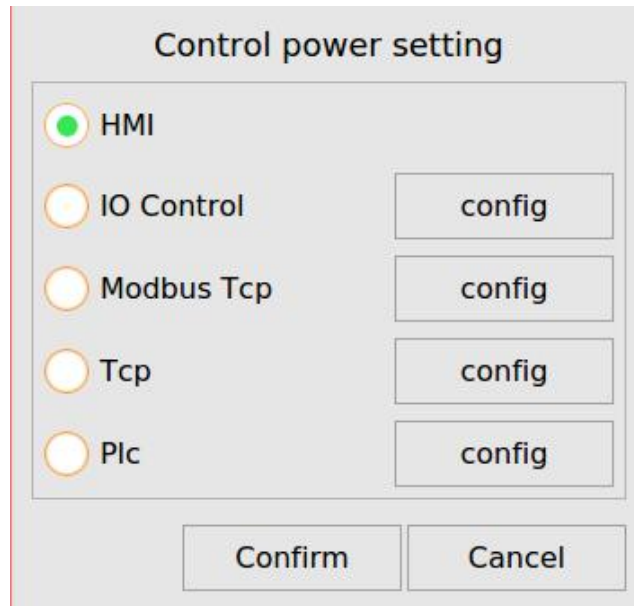


Figure 12.5 Control power setting interface

12.3 Import/Export

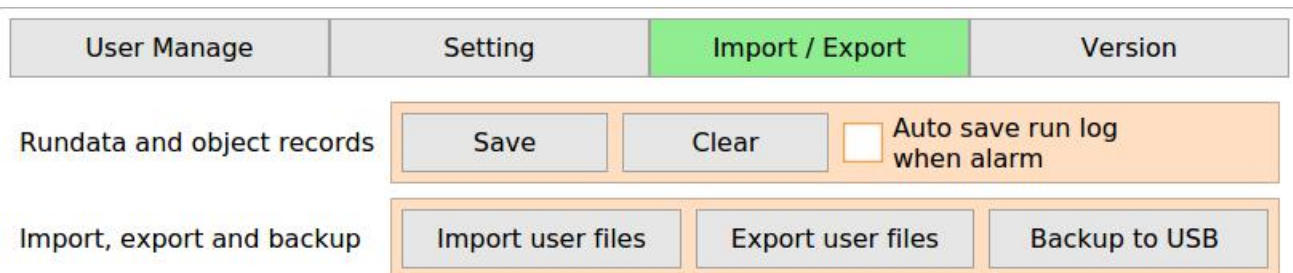


Figure 12.6 Import and export interface

■ Operating data and object record

Save: Output the running data and object records of the previous 30s.

Clear: Clear the currently saved running data and object records.

Automatically save the running log when alarming: check the box, and the running log will be automatically saved when the alarm is issued.

■ Import user files

Import the user file *.tar.gz to the system.

■ Export user files and logs

Export to local: Export user files and logs CX_User202009151634.tar.gz to the work directory.

Export to USB: Export user files and logs CX_User202009151634.tar.gz to USB.

User file naming rules: CX_User date (mm / DD / mm / min). Tar.gz

12.4 Version

Displays the current Atommotion version number.

The version number consists of three parts: Major version number - detailed version information date.

Chapter13 Appendix I Instruction System Introduction

13.1 Motion Command

13.1.1 Ptp

Instructions	Describe
Ptp	All joints of the robot reach the target joint location at the same time
parameter	describe
dest_pos: JointPosition	Target joint location
dyn: Dynamic	Robot dynamic parameters
Return value	describe
Void	No return value

Create a new Ptp command, the interface is shown in Figure 13.1.

Type	Value
▼ Ptp(,);	
dest_pos:JointPosition	<input type="text"/>
▼ dyn:Dynamic	Default
▶ :Dynamic	

Figure 13.1 PTP command interface

■ dest_pos Target joint location

The number of editable members of this variable is related to the robot model. When the robot is a four shaft model, a1, a2, a3 and a4 can be edited; If it is a three-shaft model, a1, a2 and a3 can be edited, and so on;

The parameter value indicates the location of the shaft relative to the zero point. If it is a rotating shaft, the unit is degrees; if it is a linear shaft, the unit is mm.

■ dyn Dynamic Parameters of Robot

Please refer to Dynamic. When using this parameter, you can use the system default global dynamic parameter variable or customize it.

13.1.2 Line

Instructions	Describe
Line	The end of the robot moves in a straight line to the target location at the specified speed.
parameter	describe
dest_pos: TcpPosition	Target end location
dyn: Dynamic	Robot dynamic parameters
transition: Transition	Robot motion smoothing parameters
Return value	describe
Void	No return value

Create a new Line command with the screen shown in Figure 13.2.

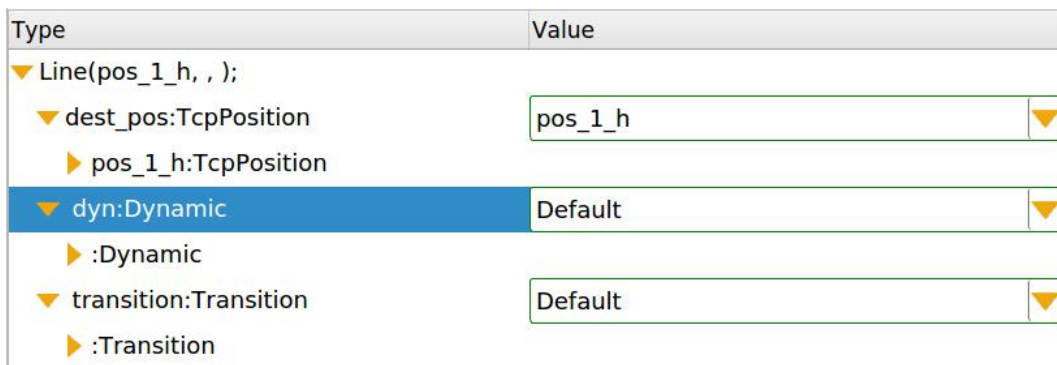


Figure13.2 Line command interface

■ dest_POS Target end location

The introduction of this parameter refers to TcpPosition. This parameter represents the location of the TCP point in the spatial coordinate system, xyz represents the location of the TCP point on the three axes of the reference coordinate system, and abc represents the gesture of the TCP point.

■ dyn Robot dynamic parameters

Refer to the Dynamic variable for the introduction of this parameter.

■ transition Robot motion smoothing parameters

The smoothing parameters are divided into No_TRANSITION and PERCENT_TRANSITION

NO_TRANSITION: Not smooth

PERCENT_TRANSITION: Percentage smoothing, value range 0-50%

13.1.3 Circle

Instructions	Describe
Circle	The end of the robot makes circular motion from the starting point, through the auxiliary point to the target point
parameter	describe
mid_pos: TcpPosition	Coordinates of arc auxiliary points
dest_pos: TcpPosition	Coordinates of arc end point
dyn: Dynamic	Robot dynamic parameters
transition: Transition	Robot motion approximation parameters
Return value	describe
Void	No return value

Note: The robot TCP end makes a full circle motion and must execute two circular arc motion commands; the starting position of this command is the target position of the previous motion command or the current robot TCP position.

Create a new circle command. The interface is shown in Figure 13.3.

Type	Value
▼ Circle(pos_1_h, pos_1_l, ,);	
▼ mid_pos:TcpPosition	pos_1_h ▼
▶ pos_1_h:TcpPosition	
▼ dest_pos:TcpPosition	pos_1_l ▼
▶ pos_1_l:TcpPosition	
▼ dyn:Dynamic	Default ▼
▶ :Dynamic	
▼ transition:Transition	Default ▼
▶ :Transition	

Figure 13.3 Circle command interface

■ mid_pos Auxiliary point location

This position point is used to assist the start and end points to form an arc, and the internal parameters are set the same as the line instruction's dest_pos target end position.

The target end position (dest_pos), dynamic parameter (dyn) and smoothing parameter (transion) are the same as Line instruction.

13.1.4 PtpRel

instructions	describe
PtpRel	All joints of the robot move synchronously from the reference point to offset the joint distance
parameter	describe
base_pos: JointPosition	Base point
dist: JointDistance	Offset joint distance
dyn: Dynamic	dynamic parameter
Return value	describe
Void	No return value

Create a PtpRel command with the interface shown in Figure 13.4.

Type	Value
▼ PtpRel(, , _g.default_dyn);	
▼ base_pos:JointPosition	Default ▼
▶ :JointPosition	
dis:JointDistance	▼
▼ dyn:Dynamic	_g.default_dyn ▼
▶ default_dyn:Dynamic	

Figure 13.4 Ptprel instruction interface

■ base_pos Base point

The reference point is the current location by default.

■ dist Offset joint distance

Dist by default, the offset is 0.

13.1.5 LineRel

Instructions	Describe
LineReal	The end of the robot moves in a straight line from the reference point to offset the spatial distance
parameter	describe
base_pos: TcpPosition	Base point
dist: TcpDistance	Offset space distance
dyn: Dynamic	dynamic parameter
transition: Transition	Smooth scale
Return value	describe
Void	No return value

Create a new LineRel command with the screen shown in Figure 13.5.

Type	Value
▼ LineRel(pos_1_h, , _g.default_dyn,);	
▼ base_pos:TcpPosition	pos_1_h ▼
▶ pos_1_h:TcpPosition	
dis:TcpDistance	▼
▼ dyn:Dynamic	_g.default_dyn ▼
▶ default_dyn:Dynamic	
▼ transition:Transition	Default ▼
▶ :Transition	

Figure 13.5 LineRel command interface

■ base_pos Base point

The description is the same as PtpRel

■ dist Offset

Dist by default, the offset is 0.

Void	There is no return value
------	--------------------------

13.1.6 LineAbs

Instructions	Describe
LineAbs	The end of the robot keeps other coordinates unchanged and moves to the target location in the specified direction
parameter	describe
x: REAL	Move to the target location in the X direction (by default, this direction does not move)
y: REAL	Move to the target location in the Y direction (by default, this direction does not move)
z: REAL	Move to the target location in the Z direction (by default, this direction does not move)
a: REAL	Move to the target location in direction a (by default, this direction does not move)
b: REAL	Move to the target location in direction B (by default, this direction does not move)
c: REAL	Move to the target location in direction C (by default, this direction does not move)
dyn: Dynamic	dynamic parameter
transition: Transition	Smooth scale
Return value	describe
Void	No return value

Create a new LineAbs command with the interface as shown in Figure 13.6, and set z to -900 to indicate the z-direction movement to the -900 position.

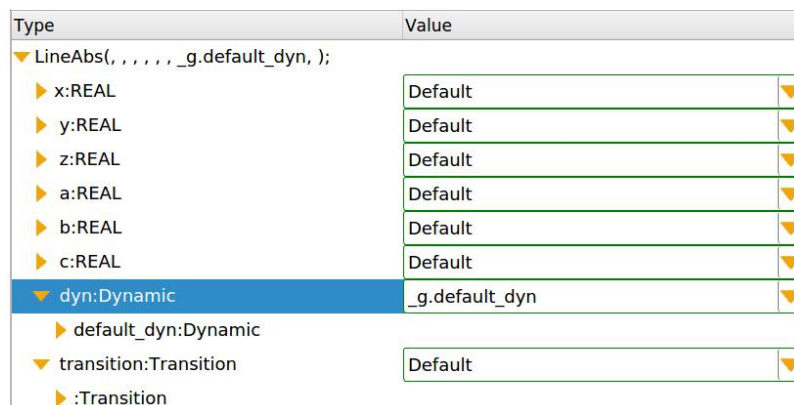


Figure 13.6 LineAbs command interface

13.1.7 ReturnHome

instructions	describe
ReturnHome	all shafts return zero
Return value	describe
Void	No return value

13.1.8 CustomPath

Instructions	Describe
CustomPath	The end of the robot moves in a custom trajectory
parameter	describe
dest_pos: ArrayOfTcpPosition	Custom track
dyn: Dynamic	dynamic parameter
transition: Transition	Smoothing parameters
Sampling_period: INT	Customize the sample rate of the track
Return value	describe
Void	No return value

Create a new CustomPath command, the interface is shown in Figure 13.7.

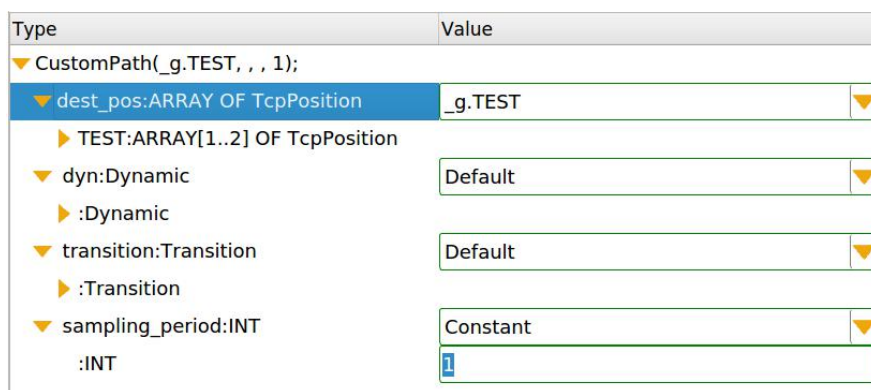


Figure 13.7 Custompath command interface

13.2 Tracking Function Instruction

13.2.1 WaitObject

Instructions	Describe
--------------	----------

WaitObject	Waiting for an object
parameter	describe
obj: TargetObject	Object to wait for
time: INT	Maximum waiting time, in MS; -1 means always waiting
Return value	describe
Void	No return value

After creating a new WaitObject command, the interface looks like Figure 13.8.

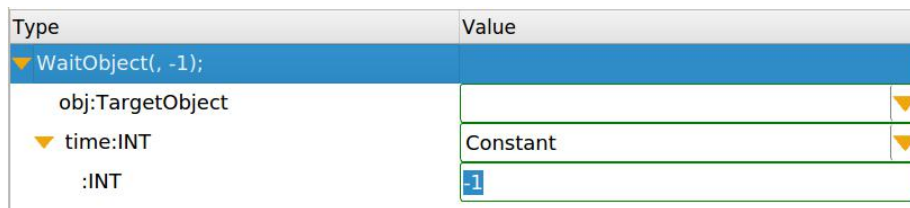


Figure 13.8 Waitobject instruction interface

■ Waiting time

When time is set to - 1, wait until the next instruction is executed after waiting for the object;

If the time is set to 20, the next instruction will be executed after the instruction has been executed for 20ms, regardless of whether it waits for the object or not.

13.2.2 IsArriveObject

instructions	describe
IsArriveObject	Determine whether the object has arrived
parameter	describe
obj: TargetObject	Target object to judge
Return value	describe
BOOL	When the object arrives, it returns true; otherwise, it returns false

13.2.3 ObjectDone

Instructions	Describe
ObjectDone	Cancels the active state of the current object and sets it to the completed state.
parameter	describe
obj: TargetObject	Object to be deactivated

Return value	describe
Void	No return value

13.2.4 ObjectCancel

Instructions	Describe
ObjectCancel	Cancels the active state of the current object and sets it back to the waiting state.
parameter	describe
obj: TargetObject	Object to be deactivated
Return value	describe
Void	No return value

13.2.5 ObjectFinish

Instructions	Describe
ObjectFinish	Deactivate the currently acquired object, and set it to completed to send, inactive, outgoing
parameter	describe
obj: TargetObject	The object to deactivate
Return value	describe
Void	No return value

13.2.6 ObjectClear

Instructions	Describe
ObjectClear	Empty all objects on a conveyor.
parameter	describe
con: Conveyor	Select empty for all conveyors to empty
Return value	describe
Void	No return value

13.3 Setting Instruction

All motion commands after the execution of the setting command use the parameters set by the setting command, except for the internal setting parameters of the command.

13.3.1 SetDynamic

Instructions	Describe
SetDynamic	Sets the default dynamic parameters used in instruction execution
parameter	describe
dyn: Dynamic	dynamic parameter
Return value	describe
Void	No return value

13.3.2 SetTransition

Instructions	Describe
SetTransition	Sets the default smoothing scale used in instruction execution
parameter	describe
transition: Transition	Smooth scale
Return value	describe
Void	No return value

13.3.3 SetAcceleration

Instructions	Description
SetAcceleration	Sets the default acceleration law used in command execution
parameter	describe
acceleration_type: AccelerationType	Acceleration law TRAPEZOID_ACC: T-shaped acceleration type PARABOLA_ACC: parabolic acceleration type SIN_ACC: sinusoidal acceleration type COS_ACC: cosine acceleration type
Return value	describe

Void	No return value
------	-----------------

13.3.4 SetCartSys

Instructions	Description
SetCartSys	Sets the coordinate system used in instruction execution
parameter	describe
cart_sys:CartSys	Coordinate system
Return value	describe
Void	No return value

13.3.5 SetTool

Instructions	Description
SetTool	Sets the tools used in instruction execution
parameter	describe
tool:Tool	tool
Return value	describe
Void	No return value

13.4 Input / Output Instruction

13.4.1 SetDout

Instructions	Describe
SetDout	Set the actual output port to true or false
Parameter	describe
dout: Dout	Output port to set
value: BOOL	Sets the value of the output port
Return value	describe
Void	No return value

13.4.2 SetVDout

Instructions	Describe
SetVDout	Set the virtual output port to true or false

parameter	describe
dout: VDout	Output port to set
value: BOOL	Sets the value of the output port
Return value	describe
Void	No return value

13.5 Trigger Instruction

13.5.1 OnDistanceDO

Instructions	Describe
OnDistanceDO	The trigger instruction executed after the last instruction is executed to a certain distance
parameter	describe
type: TriggerDistance	Trigger type From begin: triggered at a certain distance from the starting point; From: triggered at a certain distance from the end point;
distance: REAL	The set distance cannot be negative
time: INT	Delay triggering after the set distance is reached, unit: ms
action: Action	Expression executed after triggering, SetDout, SetVDout,:=
Return value	describe
Void	No return value

13.5.2 OnPercentDO

Instructions	Describe
OnPercentDO	Trigger instruction executed after a certain percentage of the previous instruction is executed
parameter	describe
percent: REAL	Set percentage
time: INT	Set the delay time after percentage arrival, in MS
action: Action	Expression executed after triggering: SetDout, SetVDout,:=
Return value	describe
Void	No return value

Note: the trigger command cannot be used alone. There must be a motion command before the trigger command, otherwise an error will be reported during operation.

13.6 Wait Instructions

13.6.1 Wait

Instructions	Describe
Wait	Wait for a condition to hold
parameter	describe
condition: BOOL	A condition: the value of a constant, variable, or the return value of an expression; The types of expressions are: getdvalue, getdoutvalue, isarriveobject and other operators
Return value	describe
Void	No return value

13.6.2 WaitIsFinished

Instructions	Describe
WaitIsFinished	Wait until the previous instruction is executed, and then continue to execute the subsequent instruction; This command cancels the smooth transition between the previous motion command and the next motion command.
Return value	describe
Void	No return value

13.6.3 WaitTime

Instructions	Describe
WaitTime	Wait a fixed time
parameter	describe
time: INT	Waiting time, when it is - 1, always waiting, unit: ms
Return value	describe
Void	No return value

13.7 Process Control Instruction

13.7.1 IF

Instructions	Describe
IF	Condition instruction, if the condition is met, execute the next step
parameter	describe
condition :BOOL	If condition: constant, variable value or return value of expression; The types of expressions are: getdvalue, getdoutvalue, isarriveobject and other operators
Return value	describe
Void	No return value

13.7.2 ELSIF

Instructions	Describe
ELSIF	Condition command, execute the next step if the elsif condition is met
parameter	describe
:BOOL	If condition: constant, variable value or return value of expression; The types of expressions are: getdvalue, getdoutvalue, isarriveobject and other operators
Return value	describe
Void	No return value

13.7.3 ELSE

Instructions	Describe
ELSE	Conditional instruction
Return value	describe
Void	No return value

13.7.4 WHILE

instructions	describe
WHILE	When the while condition is met, the loop executes

parameter	describe
:BOOL	The types of expressions are: getdinfile, getdoutvalue, isarriveobject and other operators
Return value	describe
Void	No return value

13.8 Assignment Instruction

13.8.1 :=

Instructions	Describe
:=	Assignment instruction
Return value	describe
Void	No return value

13.9 Monitoring Area Command

13.9.1 EnableWorkArea

Instructions	Describe
EnableWorkArea	Set the specified work area as enabled
parameter	describe
work_area: WorkArea	Work area to set
Return value	describe
Void	No return value

13.9.2 DisableWorkArea

Instructions	Describe
DisableWorkArea	Set the specified work area to disabled
parameter	describe
work_area: WorkArea	Work area to set
Return value	describe
Void	No return value

13.10 Palletizer instruction

13.10.1 ResetPalletizer

Instructions	Describe
ResetPalletizer	Reset the current location of palletizing
parameter	describe
palletizer: Palletizer	Stack to reset
Return value	describe
Void	No return value

13.10.2 NextPalletizer

Instructions	Describe
NextPalletizer	Set to next stacking location
parameter	describe
palletizer: Palletizer	Palletizing variable
Return value	describe
Void	No return value

13.10.3 SetPalletizerNum

Instructions	Describe
SetPalletizerNum	Sets the serial number of the current location of the palletizing
parameter	describe
palletizer: Palletizer	Palletizing variable to set
Num: INT	Location sequence number to set
Return value	describe
Void	No return value

13.11 PLC Instructions

13.11.1 StartPLC

Start the PLC program.

13.11.2 StopPLC

Stop the PLC program.

13.12 Communication Instruction

13.12.1 SendTcpData

Instructions	Describe
SendTcpData	Send TCP data
parameter	describe
Tcp_connect: TcpConnect	Specifies the tcpconnect variable to send
data:STRING	Data to send
Return value	describe
Void	No return value

13.13 Mathematical Operation Instruction

13.13.1 SIN

Instructions	Describe
SIN	Sinusoidal trigonometric function
parameter	describe
:REAL	Angle in degrees

13.13.2 COS

Instructions	Describe
COS	Cosine trigonometric function
parameter	describe
:REAL	Angle in degrees

13.13.3 TAN

Instructions	Describe
TAN	Tangent trigonometric function

parameter	describe
:REAL	Angle in degrees

13.13.4 ASIN

Instructions	Describe
ASIN	Inverse sine trigonometric function
parameter	describe
:REAL	Inverse sine trigonometric function parameters

13.13.5 ACOS

Instructions	Describe
ACOS	Inverse cosine trigonometric function
parameter	describe
:REAL	Inverse cosine trigonometric function parameters

13.13.6 ATAN

Instructions	Describe
ATAN	Arctangent trigonometric function
parameter	describe
:REAL	Arctangent trigonometric function parameters

13.13.7 LN

Instructions	Describe
LN	Natural logarithmic function
parameter	describe
:REAL	You need to find the value of the natural logarithm

13.13.8 EXP

Instructions	Describe
EXP	Exponential function based on e
parameter	describe

e: :REAL	Index of e
----------	------------

13.13.9 ABS

Instructions	Describe
ABS	Absolute value function
parameter	describe
:REAL	The number for which the absolute value is required

13.13.10 SQRT

Instructions	Describe
SQRT	Open square root function
parameter	describe
:REAL	A value that requires an open square root

13.14 Operator

13.14.1 +

Operator	Describe
+	Summation operator (support real and int types)

13.14.2 -

Operator	Describe
-	Difference operator (support real and int types)

13.14.3 *

Operator	Describe
*	Multiplication operator (support real and int types)

13.14.4 /

Operator	Describe
/	Quotient function (support real and int types)

13.14.5 AND

Operator	Describe
AND	And operator (bool type is supported)

13.14.6 OR

Operator	Describe
OR	Or operator (bool type supported)

13.14.7 XOR

Operator	Describe
XOR	XOR operator (bool type supported)

13.14.8 NOT

Operator	Describe
NOT	Non operator (bool type supported)

13.14.9 <

Operator	Describe
<	Less than operator

13.14.10 >

Operator	Describe
>	Greater than operator

13.14.11 <=

Operator	Describe
<=	Less than or equal to operator

13.14.12 >=

Operator	Describe
>=	Greater than or equal operator

13.14.13 =

Operator	Describe
=	Judge whether they are equal

13.14.14 <>

Operator	Describe
<>	Judge whether it is unequal

13.14.15 ()

Operator	Describe
()	The parenthesis operator in the expression is used to change the running priority

13.14.16 []

Operator	Describe
[]	The square bracket operator in the expression allows the user to get the array elements

13.15 Object Information

13.15.1 GetObjectId

Instructions	Describe
GetObjectId	Gets the ID of the object
parameter	describe
obj: TargetObject	Object to get ID
Return value	describe
INT	ID value of the object

13.15.2 GetObjectAttr

Instructions	DESCRIBE
GetObjectAttr	Get the attr of the object
parameter	describe
obj: TargetObject	Object to get attr
Return value	describe
INT	Attr value of object

13.15.3 SetObjectAttr

Instructions	Describe
SetObjectAttr	Set the attr of the object
parameter	describe
obj: TargetObject	Object to set attr
attr: INT	Attr to set
Return value	describe
Void	No return value

13.15.4 GetObjectInfo

Instructions	Describe
GetObjectInfo	Get object info
parameter	describe
obj: TargetObject	Object to get info
Return value	describe
STRING	Object information string

13.15.5 GetObjectOriginLocation

Instructions	Describe
GetObjectOriginLocation	Get the original location of the object
parameter	describe
obj: TargetObject	Object to get location
Return value	describe

dest_pos: TcpPosition	Original location of object
-----------------------	-----------------------------

13.16 Conversion Instruction

13.16.1 IntToString

Instructions	Describe
IntToString	Int type to string
parameter	describe
input: Int	Input int data
Return value	describe
STRING	character string

13.16.2 RealToString

Instructions	Describe
RealToString	Real type to string
parameter	describe
input: Real	Input int data
Return value	describe
STRING	character string

13.16.3 BoolToString

Instructions	Describe
BoolToString	Bool type to string
parameter	describe
input: Bool	Bool data input
Return value	describe
STRING	character string

13.16.4 StringToInt

Instructions	Describe
StringToInt	String to int type
parameter	describe

input: String	String data entered
Return value	describe
INT	Int value converted from string

13.16.5 StringToReal

Instructions	Describe
StringToReal	String to real type
parameter	describe
input: String	String data entered
Return value	describe
REAL	Real value converted from string

13.16.6 StringToBool

Instructions	Describe
StringToBool	String to bool type
parameter	describe
input: String	String data entered
Return value	describe
BOOL	Bool value converted from string

13.17 Other instructions

13.17.1 GetCacheString

Instructions	Describe
GetCacheString	Gets the string of the cache
parameter	describe
input: Int	Index of the cache (index value starts from 1)
Return value	describe
STRING	character string

13.17.2 GetRobotCurrentEndJointPosition

Instructions	Describe
GetRobotCurrentEndJointPosition	Obtain the feedback value of the current joint position of the robot end
Return value	describe
TcpPosition	Feedback value of the current joint position at the end of the robot

13.18 Other instructions

13.18.1 GetCacheString

Instructions	Describe
GetCacheString	Gets the string of the cache
parameter	describe
input: Int	Cache index (index value starts at 1)
Return value	describe
STRING	string

13.18.2 CustomAlarm

Instructions	Describe
CustomAlarm	Generates a custom alarm
parameter	describe
alarm: Alarm	Alarm information variable
Return value	describe
Void	

13.18.3 LineSearch

Instructions	Describe
LineSearch	The robot moves to the space target position in a straight

	line. When the conditions are met, the instruction is directly skipped and the next instruction is executed.
parameter	describe
dest_pos: TcpPosition	Target space position
condition: Bool	Trigger condition (true or false, supporting conditional statement judgment)
dyn: Dynamic	The dynamic parameters
Return value	describe
Void	

The following figure shows an example of LineSearch.

```

6   LineSearch(pos, TRUE, dyn_max);
7   LineSearch(pos, _g.dout.value, dyn_max);

```

Figure 13.9 LineSearch instruction application of example

In line 6, if the trigger condition is true, this instruction is directly skipped and line 7 is directly executed.

In line 7, when `_g.out.value` is false, it moves in a straight line to the spatial destination `pos:TcpPosition`; If the `_G.dout.value` external trigger is true during execution, the instruction is skipped and line 8 is executed.

Chapter14 Appendix II Variable Types

14.1 Motion Variable

14.1.1 JointPosition

Structural type	Member	Describe
JointPosition	a1: REAL	End location of the first joint in degrees
	a2: REAL	End location of the second joint in degrees
	a3: REAL	End location of the third joint in degrees
	a4: REAL	End location of the 4th joint in degrees
	a5: REAL	End location of the 5th joint in degrees
	a6: REAL	End location of the 6th joint in degrees

Create a new variable pos for the four-axis model robot, as shown in Figure 14.1.

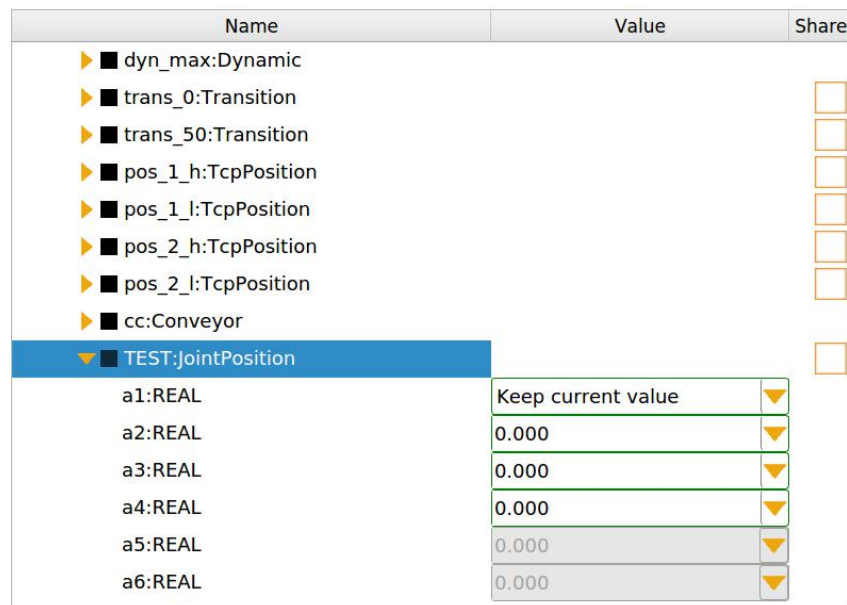


Figure 14.1 JointPosition variable interface

Each member has two options: value and maintain the current value. a1 is set to maintain the current value. During teaching, only the values of a2, a3 and a4 will be modified.

14.1.2 TcpPosition

Structural Type	Member	Describe
TcpPosition	x: REAL	X-shaft location in Cartesian coordinate system unit: mm
	y: REAL	Y-shaft location in Cartesian coordinate system unit: mm
	z: REAL	Z-shaft location in Cartesian coordinate system unit: mm
	a: REAL	Rotation angle of coordinate system around Z shaft (Euler angle) unit: degrees
	b: REAL	Rotation angle of coordinate system around X &apost;shaft (Euler angle) unit: degrees
	c: REAL	Rotation angle of coordinate system around Z &apost;shaft (Euler angle) unit: degrees

Create a new variable tcp_pos, as shown in Figure14.2.

Name	Value	Share
▶ ■ trans_0:Transition		<input type="checkbox"/>
▶ ■ trans_50:Transition		<input type="checkbox"/>
▶ ■ pos_1_h:TcpPosition		<input type="checkbox"/>
▶ ■ pos_1_l:TcpPosition		<input type="checkbox"/>
▶ ■ pos_2_h:TcpPosition		<input type="checkbox"/>
▶ ■ pos_2_l:TcpPosition		<input type="checkbox"/>
▶ ■ cc:Conveyor		<input type="checkbox"/>
▶ ■ TEST:JointPosition		<input type="checkbox"/>
▼ ■ TEST1:TcpPosition		<input type="checkbox"/>
x:REAL	Keep current value	▼
y:REAL	0.000	▼
z:REAL	0.000	▼
a:REAL	0.000	▼
b:REAL	0.000	▼
c:REAL	0.000	▼

Figure 14.2 TCP_POS variable interface

Where xyz represents the location and abc represents the attitude.

14.1.3 JointDistance

Structural Type	Member	Describe
JointDistance	a1: REAL	1st joint offset
	a2: REAL	2nd Joint offset
	a3: REAL	3rd joint offset
	a4: REAL	4th joint offset
	a5: REAL	5th joint offset
	a6: REAL	6th joint offset

Create a new variable joint_dis for the four-axis model robot, as shown in Figure 14.3.

Name	Value	Share
▶ ■ trans_50:Transition		<input type="checkbox"/>
▶ ■ pos_1_h:TcpPosition		<input type="checkbox"/>
▶ ■ pos_1_l:TcpPosition		<input type="checkbox"/>
▶ ■ pos_2_h:TcpPosition		<input type="checkbox"/>
▶ ■ pos_2_l:TcpPosition		<input type="checkbox"/>
▶ ■ cc:Conveyor		<input type="checkbox"/>
▶ ■ TEST:JointPosition		<input type="checkbox"/>
▶ ■ TEST1:TcpPosition		<input type="checkbox"/>
▼ ■ TEST2:JointPosition		<input type="checkbox"/>
a1:REAL	Keep current value	<input type="checkbox"/>
a2:REAL	0.000	<input type="checkbox"/>
a3:REAL	0.000	<input type="checkbox"/>
a4:REAL	0.000	<input type="checkbox"/>
a5:REAL	0.000	<input type="checkbox"/>
a6:REAL	0.000	<input type="checkbox"/>

Figure 14.3 joint_Dis variable interface

14.1.4 TcpDistance

Structural Type	Member	Describe
-----------------	--------	----------

TcpDistance	x: REAL	X-shaft offset
	y: REAL	Y-shaft offset
	z: REAL	Z-shaft offset
	a: REAL	A-shaft rotation offset
	b: REAL	B-shaft rotation offset
	c: REAL	Rotation offset in c-shaft direction

14.1.5 Dynamic

Structural Type	Member	Describe
Dynamic	vel_shaft: REAL	Percentage of the joint's actual running speed in its maximum speed
	acc_shaft: REAL	Percentage of the actual acceleration of the joint in its maximum acceleration
	dec_shaft: REAL	Percentage of joint deceleration in its maximum deceleration during actual operation
	jerk_shaft: REAL	Percentage of acceleration of joint in its maximum acceleration during actual operation
	vel: REAL	Actual running speed
	acc: REAL	Actual running acceleration
	dec: REAL	Actual running deceleration
	jerk: REAL	Actual running acceleration

	vel_ori: REAL	Actual operating attitude rotation angular velocity
	acc_ori: REAL	Actual operating attitude rotation angular acceleration
	dec_ori: REAL	Actual running attitude rotation angular deceleration
	jerk_ori: REAL	Actual running attitude angle plus acceleration

Note: the dynamic parameters are the maximum parameters allowed by the user. During actual operation, they need to be compared with the parameters configured in configure. If they exceed the parameters configured in configure, the parameters configured in configure shall prevail.

14.1.6 Transition

Structural Type	Member	Describe
Transition	trans_type: TransitionType	Smoothing type PERCENT_Transition: percent smooth, NO_Transition: not smooth
	trans_pos: REAL	Smoothing value, range 0-50%

14.1.7 CartSys

Structural Type	Member	Describe
CartSys	ref_cart_sys: REFERENCE TO CartSys	Reference coordinate system
	x: REAL	Offset of coordinate system origin in x-shaft direction relative to reference system
	y: REAL	Offset of coordinate system origin in Y-shaft direction relative to reference system
	z: REAL	Offset of coordinate system origin in z-shaft direction

		relative to reference system
	a: REAL	Rotation offset of coordinate system origin in a-shaft direction relative to reference system
	b: REAL	Rotation offset of coordinate system origin in b-shaft direction relative to reference system
	c: REAL	Rotation offset of coordinate system origin relative to reference system in c-shaft direction

14.1.8 Tool

Structural Type	Member	Describe
Tool	x: REAL	X location of end to tool conversion unit: mm
	y: REAL	Y location of end to tool conversion unit: mm
	z: REAL	Z location of end to tool conversion unit: mm
	a: REAL	End to tool transition a location in degrees
	b: REAL	End to tool converted B location in degrees
	c: REAL	End to tool converted C location in degrees

14.2 Tracking variable

14.2.1 TargetObject

Structural type	Member	Describe
TargetObject	ref_conveyor: REFERENCE TO Conveyor	Conveyor referenced by tracked object

	id: INT	Tracked object ID
	attr: INT	Tracked object attr
	Cart_sys: CartSys	The coordinates of the object in the conveyor coordinate system

14.2.2 Conveyor

Structural Type	Member	Describe
Conveyor	encoder: Encoder	Conveyor encoder code value interface
	resolution: REAL	Encoder resolution can be modified or calculated through teaching
	conveyor_model: ConveyorModelType	Belt type: straight, disc and static
	cart_sys: CartSys	The coordinates of the conveyor in the world coordinate system
	min_area: REAL	Minimum working area (objects can be grabbed only after entering the minimum working area)
	max_area: REAL	Maximum working area (if the object exceeds the maximum working area, give up grasping)
	latest_area: REAL	Latest work area (when grasping is planned, the object has exceeded the latest acceptance area, and the grasping is abandoned)
	ref_object_sort: REFERENCE	Sorting method: Specifies the sorting method of objects on the transfer. The sorting method may not be set

	TO ObjectSort	
--	---------------	--

14.2.3 ObjectSource

Structural Type	Member	Describe
ObjectSource	enable:BOOL	Whether to enable object source
	filter_error:REAL	Filter error: When two objects are within the error range, they are considered to be the same object, and the error value cannot be negative. The filter error value is generally the center distance of the two materials
	communication_error:REAL	communication error
	ref_conveyor:REFERENCE TO Conveyor	The object source spawns the conveyor belt where the object is located
	cart_sys:CartSys	The coordinates of the object source (vision) in the conveyor belt coordinate system
	source_type:ObjectSourceType	There are 4 types of object sources:
	Config:CameraSource	CAMERA: camera
	Config:SensorSource	SENSOR: sensor
	Config:PositionSource	POSITION: Position source
	Config:SimulateSource	SIMULATE: Simulate object source
	camera:Camera	Camera source configuration parameters

Structural Type	Member	Describe
CameraSource	period:INT	The time period for triggering the camera periodically, in ms
	ref_tcp:REFERENCE TO TcpConnect	Refer to the TcpConnect variable, which needs to configure the ip and port number for communication with the vision. Used to receive object data sent by vision.

	trigger_type:TriggerType	<p>There are 3 trigger types:</p> <p>NONE: not triggered</p> <p>HARD_TRIGGER: hard trigger</p> <p>TCP_SEND: TcpConnect triggered</p>
	trigger_io:IO	<p>The input types in io are: "din", "dout", "vdin", "vdout";</p> <p>Can be used when trigger_type is set to NONE or HARD_TRIGGER</p>
	ref_tcp_trigger:REFERENCE TO TcpConnect	<p>Refer to the TcpConnect variable, which needs to configure the ip and port number for communication with the vision.</p> <p>When the TcpConnect variable is enabled, it will actively establish communication with the vision system. After the communication is successful, it will periodically send trigger information to the vision to trigger the camera to take pictures.</p> <p>When trigger_type is set to TCP_SEND, ref_tcp_trigger can be used</p>
	tcp_trigger_message:STRING	<p>When the TCP_SEND type is triggered, the trigger information sent to the vision; when the trigger_type is set to TCP_SEND, tcp_trigger_message can be used</p>

Structural Type	Member	Describe
SensorSource	trigger_io:IO	Configure the physical input port to which the sensor is connected.
	id:INT	The input types in io are: "din", "dout", "vdin", "vdout";
	attr:INT	The id of the object generated by the sensor

Structural Type	Member	Describe
PositionSource	period:INT	communication cycle
	id:INT	Generate objects differently by position changes, specify the id of the generated object
	attr:INT	Generate objects differently by position changes, specify the attr of the generated object

Structural Type	Member	Describe
SimulateSource	id:INT	Generate the id of the virtual object
	attr:INT	Generate attr for virtual objects
	param:STRING	<p>param string format: Mode:Time;Tmin:1.000;Tmax:1.000;Dmin:100.000;Dmax:100.000;Xmin:0.000;Xmax:0.000;Ymin:0.000;Ymax:0.000;Amin:0.000;Amax:0.000</p> <p>Mode There are 2 types of virtual modes:</p> <p>Time: Generate objects by time</p> <p>Distance: Generate objects by location</p> <p>Each of the remaining parameters has a minimum value min and a maximum value max, which means that each parameter is randomly generated between the minimum and maximum values when virtualized</p> <p>T virtual object interval time, which takes effect when Mode is time</p> <p>D distance between virtual objects, effective</p>

		<p>when Mode is distance</p> <p>X, Y, A The x-coordinate, y-coordinate and rotation angle a of the object in the object source coordinate system</p>
--	--	--

Structural Type	Member	Describe
Camera	xy_switch:BOOL	<p>Axis Swap Type:</p> <p>true for xy exchange, false for no exchange</p>
	a_inverse:BOOL	Angle flip
	x0:REAL	pan to center x0
	y0:REAL	pan to center y0
	kx:REAL	x scaling
	ky:REAL	y scaling
	a:REAL	<p>Correct the rotation, the angle value is the rotation angle from the conveyor to the camera</p> <p>Value range: (-180.0, 180.0)</p>
	dx:REAL	Compensate for center offset, output dx on object coordinate system, position on conveyor belt
	dy:REAL	Compensate for center offset, output dy on object coordinate system, position on conveyor belt
da:REAL	<p>Compensate for center offset, output da on object coordinate system, position on conveyor belt</p> <p>Value range: (-180.0, 180.0)</p>	

14.2.4 ObjectSort

Structural Type	Member	Describe
ObjectSort	filter_enable: BOOL	Enable object filtering
	filter_type: ObjectSortType	Object filter type None: no filtering POSITION: filter by location and compare ID POSITION_Only: filter only by location, without comparing other information
	sort_enable: BOOL	Enable object sorting
	sort_type: ObjectSortType	sort order NONE: do not sort, strictly follow the object addition time, and grasp the earliest one first. DEFAULT_SORT: among all objects entering the work area, first grab and add the earliest. DIRECTION: sort in strict accordance with the direction of the conveyor, and grasp the conveyor at the front first
	max_search_num: INT	Maximum number of lookups
	direction: REAL	Direction relative to conveyor

14.2.5 OverlapFilter

Structural Type	Member	Describe
OverlapFilter	enable: BOOL	Enable stack filtering

	ref_conveyor: REFERENCE TO Conveyor	Conveyor for stacking filtration
	object_id: INT	Object ID to stack filter
	object_size_x: REAL	The length of an object in the X direction in the object coordinate system
	object_size_y: REAL	The length of an object in the Y direction in the object coordinate system

14.2.6 ObjectAllot

Structural Type	Member	Describe
ObjectAllot	enable: BOOL	Enable object shunting
	ref_conveyor: REFERENCE TO Conveyor	Conveyor for object shunting
	ref_tcp:REFERENCE TO TcpConnect	The referenced tcp communication variable is used to communicate with the shunting device and shunting the object
	object_id: INT	Object ID to divert
	allot_type: ObjectAllotType	Shunt type Ratio: proportional Allot (0-1) Maximum: split by maximum capacity RATIO_Maximum: uncompleted Allot based on proportional distribution. Grouping: shunting by group

	ratio: REAL	Proportional value of proportional shunt
	output_num: INT	Shunt by group, number of shunts per group
	total_num: INT	Shunt by group, total number of each group

14.2.7 ConditionalControl

Structural Type	Member	Describe
ConditionalControl	enable: BOOL	Enable condition control
	ref_conveyor: REFERENCE TO CONVEYOR	Conveyor under condition control
	out_type: OutType	Port type set after the condition is established OUT_Y: Actual output OUT_VY: virtual output
	port: Port	Condition controls the port set after the condition is established
	value: BOOL	Condition controls the port value set after the condition is established
	condition: ConditionType	Condition type of condition control OBJECT_NUM: controlled according to the number of objects OBJECT_POS: control according to the location of the farthest object
	operator: OperatorType	Operators for conditional judgment: Equal: equal to Unequal: not equal to

		<p>Greater: greater than</p> <p>Less: less than</p> <p>GREATER_ Equal: greater than or equal to</p> <p>LESS_ Equal: less than or equal to</p>
	object_num: INT	When the number of objects is controlled, the number of objects
	object_pos: REAL	When the farthest object location is controlled, the location of the object

14.3 Regional variable

14.3.1 WorkArea

Structural Type	Member	Describe
WorkArea	enable: BOOL	Is the monitoring area enabled
	protect_type: ProtectType	<p>WORK_AREA: the designated area is the work area;</p> <p>FORBIDDEN_AREA: the specified area is a prohibited work area</p>
	ref_cart_sys: REFERENCE TO CartSys	Reference coordinate system of monitoring area
	start_pos: TcpPosition	Starting point of monitoring area
	shape_type: ShapeType	<p>Shape of monitoring area</p> <p>Box: box</p> <p>Cylinder: cylinder</p>

	shape: Cylinder	<p>Cylinder members:</p> <p>H: Cylinder height</p> <p>R: Cylinder radius</p>
	shape: Box	<p>Box members:</p> <p>dx: a vertex of the box points to the vector on the x shaft in the diagonal direction</p> <p>dy: a vertex of the box points to the vector on the y shaft in the diagonal direction</p> <p>dz: a vertex of the box points to the vector on the z shaft in the diagonal direction</p>

14.4 Input and Output Variables

14.4.1 Din

Structural Type	Member	Describe
Din	port: Port	Port number
	value: BOOL	Port value

14.4.2 Dout

Structural Type	Member	Describe
Dout	port: Port	Port number
	value: BOOL	Port value

14.4.3 VDin

Structural Type	Member	Describe
VDin	port: Port	Port number
	value: BOOL	Port value

14.4.4 VDout

Structural Type	Member	Describe
VDout	port: Port	Port number
	value: BOOL	Port value

14.5 Basic Data Type Variable

14.5.1 INT

Data Type	Member	Describe
INT	int: INT	Int type data value

14.5.2 REAL

Data Type	Member	Describe
REAL	real: REAL	Real type data value

14.5.3 BOOL

Data Type	Member	Describe
BOOL	bool: BOOL	Bool type data value: true or false

14.5.4 STRING

Data Type	Member	Describe
STRING	string: String	String type data.

14.6 Array variable

14.6.1 ARRAY OF INT

Array	Array Element	Describe
ARRAY OF INT	[]: INT	Array elements are of type int

14.6.2 ArrayOfTcpPosition

Array	Array Element	Describe
ArrayOfTcpPosition	[]: TcpPosition	Array elements are of type TcpPosition

14.7 Palletizer Variable

14.7.1 Palletizer

Structural Type	Member	Describe
Palletizer	current_number: INT	Serial number of current stacking location (range 1 ~ max)_number)
	max_number: INT	No. of maximum stacking location
	is_full: BOOL	Full stamp, read-only parameter (current_number=max_number)
	is_empty: BOOL	Empty stack, read-only parameter (current)_number=1)
	current_object: ARRAY OF INT	Serial number of current workpiece in stacking sequence
	sequence: palletizerSequence	There are six combined stacking sequences. For example, YXZ indicates that the Y direction is placed first, then the X direction, and finally the Z direction
	palletizer_distance: ARRAY OF REAL	Offset distance in XYZ direction during stacking

palletizer_count: ARRAY OF INT	Number of pallets placed in XYZ direction
palletizer_location: TcpPosition	Stacking point location
first_location: TcpPosition	Location of the first point for stacking
entry_location: TcpPosition	The location of the stacking entry point, and the location entering the first point of stacking is the location of the entry point
entry_location_enable: BOOL	Enable palletizing entry point
front_location: TcpPosition	Point location before stacking
front_location_dx: REAL	Offset value of pre stacking point relative to X direction of stacking point
front_location_dy: REAL	Offset value of pre stacking point relative to Y direction of stacking point
front_location_dz: REAL	Offset value of pre stacking point relative to Z direction of stacking point
front_location_absolut:	Is the z-direction offset of the pre stacking point

	BOOL	relative to the stacking point an absolute value
	front_location_enable: BOOL	Enable pre palletizing point
	back_location: TcpPosition	Point location after stacking
	back_location_dx: REAL	Offset value of the post stacking point relative to the X direction of the stacking point
	back_location_dy: REAL	Offset value of post stacking point relative to Y direction of stacking point
	back_location_dz: REAL	Offset value of post stacking point relative to Z direction of stacking point
	back_location_absolute: BOOL	Is the z-direction offset of the post stacking point relative to the stacking point an absolute value
	back_location_enable: BOOL	Enable post palletizing point

14.7.2 PalletizerData

Structural Type	Member	Describe
PalletizerData	Name: STRING	The name of the palletizing data variable

	objects:ARRAY OF PalletizerObject	An array of palletized object variables to store palletized objects with different attributes
	layers:ARRAY OF PalletizerLayer	Palletizing an array of variables that hold palletizing layers of different attributes
	layer_index:ARRAY OF INT	An array of index values used for the palletizing layer
	layer_position: ARRAY OF TcpPosition	Array of stacking position points used

Structural Type	Member	Describe
PalletizerObject	Name: STRING	The name of the palletizing object
	objects_type: PalletizerObjectShape	Palletizing object type, including Box, Cylinder, Ball
	shape_data:ShapeData	The basic properties of stacking objects are length, width and height, radius and radius respectively
	object_position: TcpPosition	Object offset value

Structural Type	Member	Describe
PalletizerLayer	Name: STRING	Name of palletizing layer
	palletizer_size:ARRAY OF INT	Palletizing size is a one-dimensional array representation of length 2
	objects_index:ARRAY OF INT	List of palletized object index values

	palletizer_position: ARRAY OF TcpPosition	List of position points for palletizing objects
	front_position: ARRAY OF TcpPosition	A list of precursors to palletized objects
	back_position: ARRAY OF TcpPosition	A list of trailing points for palletized objects

14.7.3 SeniorPalletizer

Structural Type	Member	Describe
SeniorPalletizer	current_number: INT	Current palletizing point number (range 1~max_number)
	max_number: INT	The serial number of the last palletizing point
	is_full: BOOL	Full stomp is a read-only parameter (current_number = max_number)
	is_empty: BOOL	Empty stack, read-only parameter (current_number=1)
	object_type: INT	Type of object
	current_index: ARRAY OF INT	The index of the current object refers to the number of objects in which the object is located
	palletizer_position: TcpPosition	Position of palletizing point
	front_position: TcpPosition	The position to be passed before entering the palletizing point, e.g. the high point of the palletizing point

	back_position: TcpPosition	The position behind the palletizing point, the position that needs to be passed after leaving the palletizing point, e.g. the high point of the palletizing point
	off_position: TcpPosition	Stacking offset
	ref_palletizer:REFERENCE TO PalletizerData	Referenced palletizing data variable

14.8 Communication Variable

14.8.1 TcpConnect

Structural Type	Member	Describe
TcpConnect	enable: BOOL	Port number
	tcp_type: TcpType	SERVER: Server CLIENT: client
	ip: STRING	IP address
	Port: INT	Port number

14.8.2 HardTrigger

Structural Type	Member	Describe
HardTrigger	enable: BOOL	Enable hard trigger
	output_io: IO	Output IO port
	input_io: IO	Input IO port
	encoder: Encoder	Reference encoder

	<p>signal_type: TriggerSignalType</p>	<p>Input signal type</p> <p>IO_RISE: rising edge of IO</p> <p>TIME_PERIOD: time period</p> <p>DISTANCE: distance period</p> <p>LOCATION: fixed location</p>
	<p>signal_value: Real</p>	<p>Input signal parameters, time code value, period or fixed code value (MS or Inc)</p>
	<p>delay_type: TriggerDelayType</p>	<p>Delay signal type</p> <p>None: no delay</p> <p>TIME_Period: delay by a certain time</p> <p>Distance: delay by a certain distance</p>
	<p>delay_value: Real</p>	<p>Delay signal parameter, a fixed time or distance (MS or Inc) from the signal trigger time</p>
	<p>reset_type: TriggerResetType</p>	<p>Reset signal type</p> <p>TIME_Period: time period</p> <p>Distance: fixed distance</p> <p>IO_Down: falling edge</p>
	<p>reset_value: Real</p>	<p>Reset signal parameters from delay_Value a fixed time or distance (MS or Inc) from the end of the delay</p>

14.8.3 Alarm

结构体	成员	描述
Alarm	code:INT	Custom alarm code, value range 1-4095
	message:STRING	Custom alarm information
	level:INT	Custom alarm levels: 1= Information, 2= Warning, 3= general error, 4= fatal error

	operation:INT	How to operate the robot after the customized alarm: 0 = no operation is required when the alarm is reported, 1= stop and pause state is planned when the alarm is reported, 2= Stop and stop state is planned when the alarm is reported, 3= emergency stop is required when the alarm is reported, the speed plan will be discarded immediately stop and enter the stop state
--	---------------	---

14.8.4 AxisLimit

结构体	成员	描述
AxisLimit	enable:BOOL	Whether to enable variables
	axis:AxisType	Name of the axis to be detected
	pdo:PdoType	The Pdo type (actual position, actual speed, or actual torque) is handled by detecting the Pdo as an lvalue
	alarm:Alarm	When the conditions are met, the alarm will be triggered. When the alarm code is 0, the alarm will not be enabled by default
	output_io:IO	If the conditions are met, set the IO value for the specified output
	output_value:BOOL	Specifies whether the output IO can be given a value to set
	operator:OperatorType	Operators for conditional judgment: Is EQUAL to EQUAL:

		UNEQUAL: Not equal GREATER. LESS: LESS than GREATER_EQUAL: greater than or EQUAL to LESS_EQUAL: Less than or EQUAL to
	limit_value:REAL	Limit value, as an rvalue

Chapter15 Appendix III Alarm Information

15.1 Serious Error

Code	Describe	Solution
0x2111	Ethercat open license file failed	<p>The ethercat mac address configuration is abnormal</p> <ol style="list-style-type: none"> 1. Please check the master station communication network port wiring and mac address configuration is correct, turn off the industrial computer and restart 2.Contact supplier for solution
0x2112	EtherCAT master initialization failed	<ol style="list-style-type: none"> 1.Restart 2.Contact supplier for solution
0x2113	EtherCAT read XML config failed	<ol style="list-style-type: none"> 1.The XML file of the device is damaged 2.Contact supplier for solution
0x2114	EtherCAT error executing script	<ol style="list-style-type: none"> 1. The Ethernet communication network port is not connected. Check whether the network cable connection is normal and close
0x2115	EtherCAT failed to set license	<ol style="list-style-type: none"> 1.illegal license 2.Contact supplier for solution
0x2116	Ethercat unbind driver faild	<ol style="list-style-type: none"> 1.Contact supplier for solution
0x2117	ethercat tmp bus info file error	<ol style="list-style-type: none"> 1.Contact supplier for solution

0x2118	EtherCAT init file failed	1.Contact supplier for solution
0x2121	Master station communication connection timeout	1.Check whether the network cable connection order is consistent with the XML file 2.Check whether the actual number of devices is consistent with the XML file 3.Contact supplier for solution
0x2122	Master station communication interruption	1.The communication network cable of the master station is loose or dropped 2.Contact supplier for solution
0x2123	EtherCAT license is null	1.Contact supplier for solution
0x2131	Thread exception	1.Contact supplier for solution
0x2132	Unknown exception	1.Contact supplier for solution
0x2141	Failed to read configuration file	1.Contact supplier for solution
0x2142	Failed to write configuration file	1.Contact supplier for solution
0x2151	Emergency stop IO error	1. Check and reset the e-stop IO configuration 2.Contact supplier for solution
0x2152	Enable IO error	1. Check and reset the enabling IO configuration 2.Contact supplier for solution

15.2 Error

Code	Describe	Solution
0x4111	Emergency stop button pressed!	1. When the emergency stop button is pressed, release the emergency stop button
0x4112	Driver alarm	1. Use the debugging software or manual of the corresponding drive to view the details, or consult the drive manufacturer
0x4121	TCP communication failed!	1. The IP address or port number is not correct, modify ip address or port number
0x4122	TCP received data format verification failed!	1. Set the sending format according to the requirements of the control system
0x4133	Robot shaft torque overlimit!	<ol style="list-style-type: none"> 1. Check whether there is any command in the program that causes the shaft torque to exceed the limit 2. Set reasonable limit value according to actual demand
0x4135	Robot command exceeds the set limit!	<ol style="list-style-type: none"> 1. Check whether there is any command in the program that causes the shaft location to exceed the limit 2. After canceling the limit in the configuration interface, restore the robot to the normal location 3. Set reasonable limit value according to actual demand

0x4136	Robot location error, please check zero and shaft configuration parameters!	1. When the robot starts, it is in the impossible shaft location (the positive solution of the model fails), and check the zero point and shaft configuration parameters
0x4137	The actual location of the robot exceeds the working area!	1. Slowly jog back to the work area
0x4211	Robot model inverse solution failed	1. Please check the zero and shaft configuration parameters 2. Contact supplier for solution
0x4212	Robot model forward solution failed	1. Please check the zero and shaft configuration parameters 2. Contact supplier for solution
0x4213	Robot model joint inverse solution failure	1. Check the shaft and joint configuration parameters 2. Contact supplier for solution
0x4214	Robot model joint forward solution failure	1. Check the shaft and joint configuration parameters 2. Contact supplier for solution
0x4311	Instruction parsing error!	1. Instruction writing error or program parsing logic error, check the instruction or program

0x4312	Instruction syntax error!	1.Instruction writing error, check the instruction
0x4313	Instruction conversion error!	1.Instruction writing error or instruction conversion logic error, check the instruction or program
0x4314	Loader failed	1.The program does not exist or the project does not exist, reload the project or reload the program
0x4331	Speed planning failed!	<ol style="list-style-type: none"> 1. Check whether the set parameters are reasonable. 2.Conveyor belt speed does not match the tracking speed during tracking 3.Check if the path is reasonable 4.Contact the supplier for solution
0x4332	Jog failed	<ol style="list-style-type: none"> 1. Unable to meet the current jog demand, handle according to the detailed prompt 2. Contact supplier for solution
0x4333	Abnormal external shaft	<ol style="list-style-type: none"> 1. External shaft program error, reconfigure external shaft program 2. Contact supplier for solution
0x4338	executing instructions path planning failed	1.Check whether the parameters in the command are reasonable and reset
0x4351	Instruction execution error	<ol style="list-style-type: none"> 1. If an exception occurs during instruction execution, handle it according to the specific prompt 2. Contact supplier for solution

0x4352	There are no tracked objects	<ol style="list-style-type: none"> 1. The object related execution is executed without executing the instruction to obtain the object. Check the instruction logic 2. After the program is suspended, switch the execution line, resulting in the coordinate system not corresponding to the current execution. After clearing the object, execute the program from the beginning
0x4353	Too slow to track objects	<ol style="list-style-type: none"> 1. Slow down the conveyor or increase the dynamic parameters
0x4354	The object cannot be tracked before exceeding the boundary, please adjust the parameters	<ol style="list-style-type: none"> 1. Slow down the conveyor or increase the dynamic parameters 2. Check whether the command location in the object coordinate system is reasonable
0x4355	The command cannot be executed in a moving coordinate system	<ol style="list-style-type: none"> 1. The joint motion command cannot be commanded in the object coordinate system. Modify the command logic
0x4356	Robot out of safe area!	<ol style="list-style-type: none"> 1. The command setting value exceeds the working area limit. Modify the command value or set a reasonable working area

0x4358	Statement execution exception	<ol style="list-style-type: none"> 1. If an exception occurs during statement execution, handle it according to the specific prompt 2. Contact supplier for solution
0x4411	Encoder for conveyor does not exist	1.The encoder configured for the conveyor does not exist due to importing the program or modifying the hardware configuration. Reconfigure it
0x4421	The object source configuration is in use	1.The object source variable is using the object source configuration. Delete the object source variable or set the object source variable not to use the object source configuration to be deleted
0x4422	Object source configuration not found	1.The object source configuration is deleted, use another object source configuration or create an object source configuration
0x4423	Object source configuration IO exception	1. IO configuration error, reconfigure object source IO
0x4431	Object TCP server communication error	1. The IP address or port number is not configured correctly, or the port number has been used, reconfigure it
0x4433	Object shunt configuration not found	1.The specified object shunting configuration information is not found. Add object shunting configuration information or use another configuration

0x4434	Object shunting configuration IO exception	1. IO configuration error, reconfigure IO
0x4441	Condition control IO exception	1. IO configuration error, setting correct IO
0x4451	IO error	1. IO related errors shall be repaired according to the prompt information
0x4481	IOT server startup failed	1. If the TCP port is occupied, restart the control system
0x4621	An unknown exception occurred in Modbus TCP	1.The program code is wrong. You need to check the program and contact the supplier for solution
0x4622	Modbus TCP related error	1.If the function code is not supported or the data packet is wrong, eliminate the error according to the prompt
0x4623	Modbus TCP has no control power	1.The switching control is ModbusTcp
0x4633	TCP control error processing request	1.Contact supplier for solution
0x4700	PLC error	1.Contact supplier for solution

15.3 Warning

Code	Describe	Solution
------	----------	----------

0x6111	Unable to respond to start command	1.In the process of running, stop the program and start again 2.After confirming the upper enable, start running the program
0x6112	Forced switching of operation mode during operation	
0x6113	Cannot respond to a single step command	
0x6114	Cannot respond to continuous commands	
0x6115	The motor cannot be powered down during operation	
0x6116	Forced switching of operation mode in enabled state	
0x6117	The current system has an error level alarm and cannot respond to the "start" command	
0x6118	Virtual object error	
0x6119	Object quantity overflow	
0x6121	Out of working area when stopped	

Chapter16 Appendix IV Application example

16.1 Palletizing

Configure the parameter value of palletizing variable (refer to [Palletizer](#) variable for the meaning of parameter value).

■ Description of stacking procedure

```
20      SetCartSys(md);
21      Line(mdd.front_position,dyn,tra_50);
22      Line(mdd.palletizer_position,dyn,tra_50);
23      Line(mdd.back_position,dyn,tra_50);
24      ▼ IF mdd.is_full THEN
25          ResetPalletizer(mdd);
26      ▼ ELSE
27          NextPalletizer(mdd);
```

Figure 16.1 example of stacking procedure

图

Line 20: New palletizing coordinate system

Line 21: Move to the point before stacking

Line 22: Move to stacking point

Line 23: Move to the point after stacking

Lines 24 to 27: If the stacking is full, the stacking is reset; If it is not full, the next stacking point will be executed

Note:

(1) In the sample program, the point before palletizing, the point after palletizing, and the palletizing entry point are not set. Please refer to the description of palletizing variables to set according to your needs.

(2) The max_number member of the palletizing variable in the sample program is set to 60, which means that all palletizing points need to be filled as shown in the figure. Assuming that the last two points do not want to be placed, max_number is set to 58.

(3) If there are some points in the middle that do not want to be placed, please use the NextPalletizer command to update the serial number currently placed.

(4) The tracking parameters related to palletizing in this sample program are not set and are not for reference.

Graphical description corresponding to the sample program

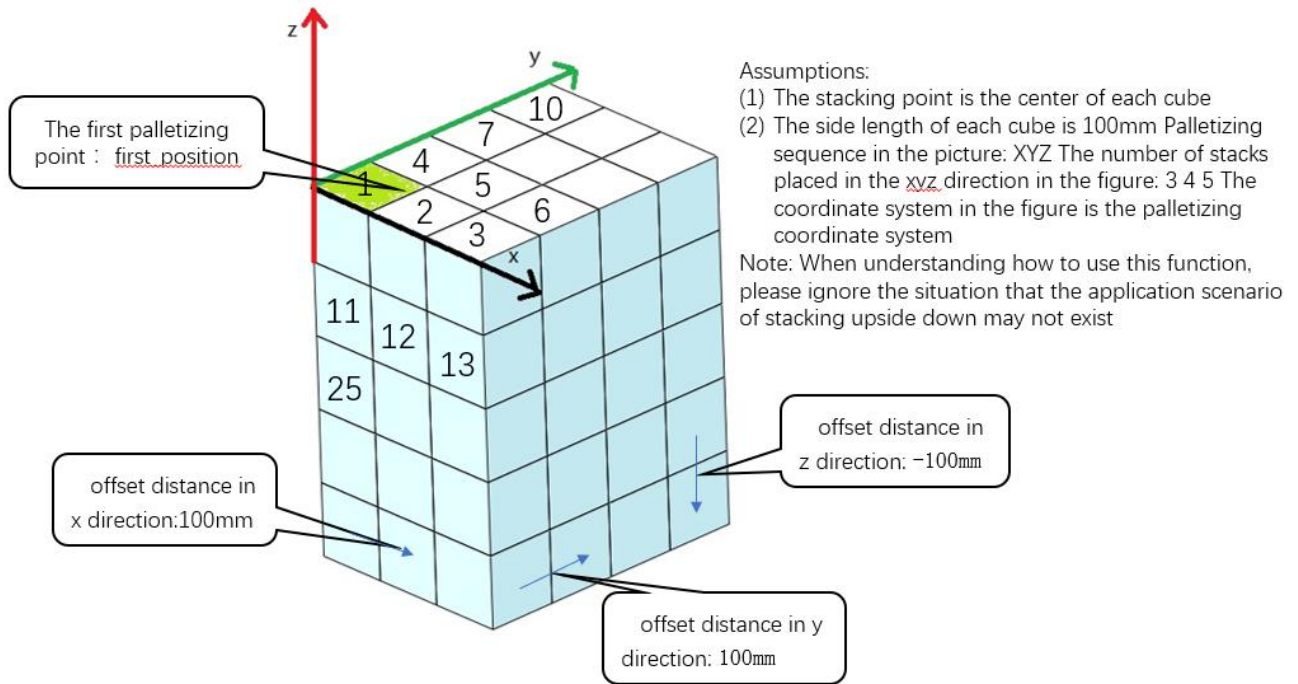


Figure 16.2 Palletizing diagram corresponding to the palletizing program

16.2 SeniorPalletizer

SeniorPalletizer function is to configure more complex palletizing, with the following characteristics:

- ① There are different kinds of objects in SeniorPalletizer variables, which can have different kinds of palletizing layers;
- ② Each object class has a unique name and can have different types and parameters; Object types can be added or deleted at will, and existing objects can be modified.
- ③ Each kind of stacking layer has a unique name, can be placed different kinds of objects, each object has a target point, front point, back point and offset point; Palletized layers can be added or deleted at will, and existing palletized layers can be modified.
- ④ In advanced stacking, the number of stacking layers can be customized, and the position and Angle of stacking layer can be set; You can select and configure it by palletizing layer name.

To sum up, the use of SeniorPalletizer needs to combine the creation of SeniorPalletizer variables, example configuration description and example program description, as shown below.

16.2.1 SeniorPalletizer variables created

SeniorPalletizer variables and PalletizerData variables need to be created in the variable interface. md :SeniorPalletizer refers to md_data :PalletizerData, as shown in the figure below.

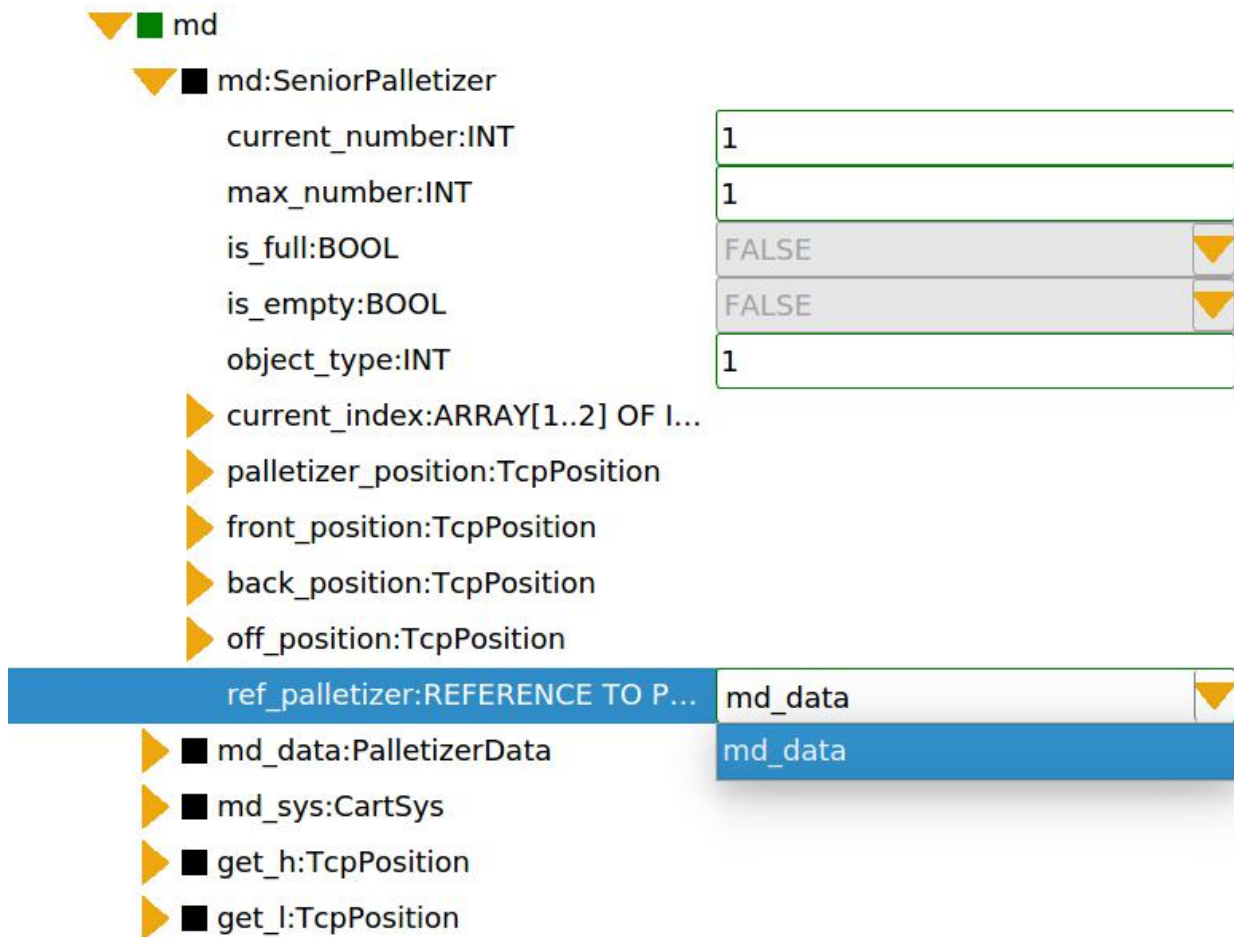


Figure 16.3 SeniorPalletizer variable creation example

16.2.2 Example Configuration Description

SeniorPalletizer variables are configured in function blocks, and can only be configured with global variables or SeniorPalletizer variables under loaded programs.

■ Parameter Configuration Instructions

There are four interfaces in total: SeniorPalletizer configuration interface, object configuration interface, single palletizing configuration interface, and palletizing queue configuration interface.

1. SeniorPalletizer configuration interface: used to display SeniorPalletizer variable information, as shown in the figure below.

Sen.Palle. test-md_test:md

Sen.Config. Obj.Config. Single Config. Palle. Config.

Full Pallets Current Nur 1 Current L 1 Obj.Num. 2

Empty Pallets Max. Num. 88 Current C 1 Current Obj 2

Senior palletizer variable-position parameter value

Tar.Pos. x 0.000 mm y 0.000 mm z 0.000 mm
a 0.000 ° b 0.000 ° c 0.000 °

For.Pos. x 0.000 mm y 0.000 mm z 0.000 mm
a 0.000 ° b 0.000 ° c 0.000 °

Bac.Pos. x 0.000 mm y 0.000 mm z 0.000 mm
a 0.000 ° b 0.000 ° c 0.000 °

Fast Config

Figure 16.4 md:SeniorPalletizer SeniorPalletizer configuration interface

"[Fast Configuration](#)" : Provides advanced palletizing features for quick configuration, which are described in the next section.

2. Object configuration interface: used to configure SeniorPalletizer variables with different types of objects, data type is PalletizerObject. Each object class has a unique name and can have different types and parameters; Object types can be added or deleted at will, and existing objects can be modified, as shown in the following figure.

Figure 16.5 tt:SeniorPalletizer Object configuration interface

- Object List: A configured object list. You can add or delete objects to the list by clicking Add and Delete buttons. You can configure an object on the page.
- "New" button: When creating a new object, you need to enter the name of the object. The name of the object must not be the same as other objects in the current object array, and follow the variable naming rules
- Delete button to delete an object.
- Select an object in the object list to modify the object shape, object properties, and object offset.

3. Single palletizing configuration interface: used to configure different palletizing layers of SeniorPalletizer variables. Data type is PalletizerLayer. Each palletizing type has a unique name and can be used for different kinds of objects. Each object has a target point, a front point, a back point and an offset point. Palletized layers can be added or deleted at will, and existing palletized layers can be modified. It is divided into three parts: palletizing layer management, palletizing object

management, real-time map management. Among them, real-time graph management is introduced in the next section, as shown in the figure below.

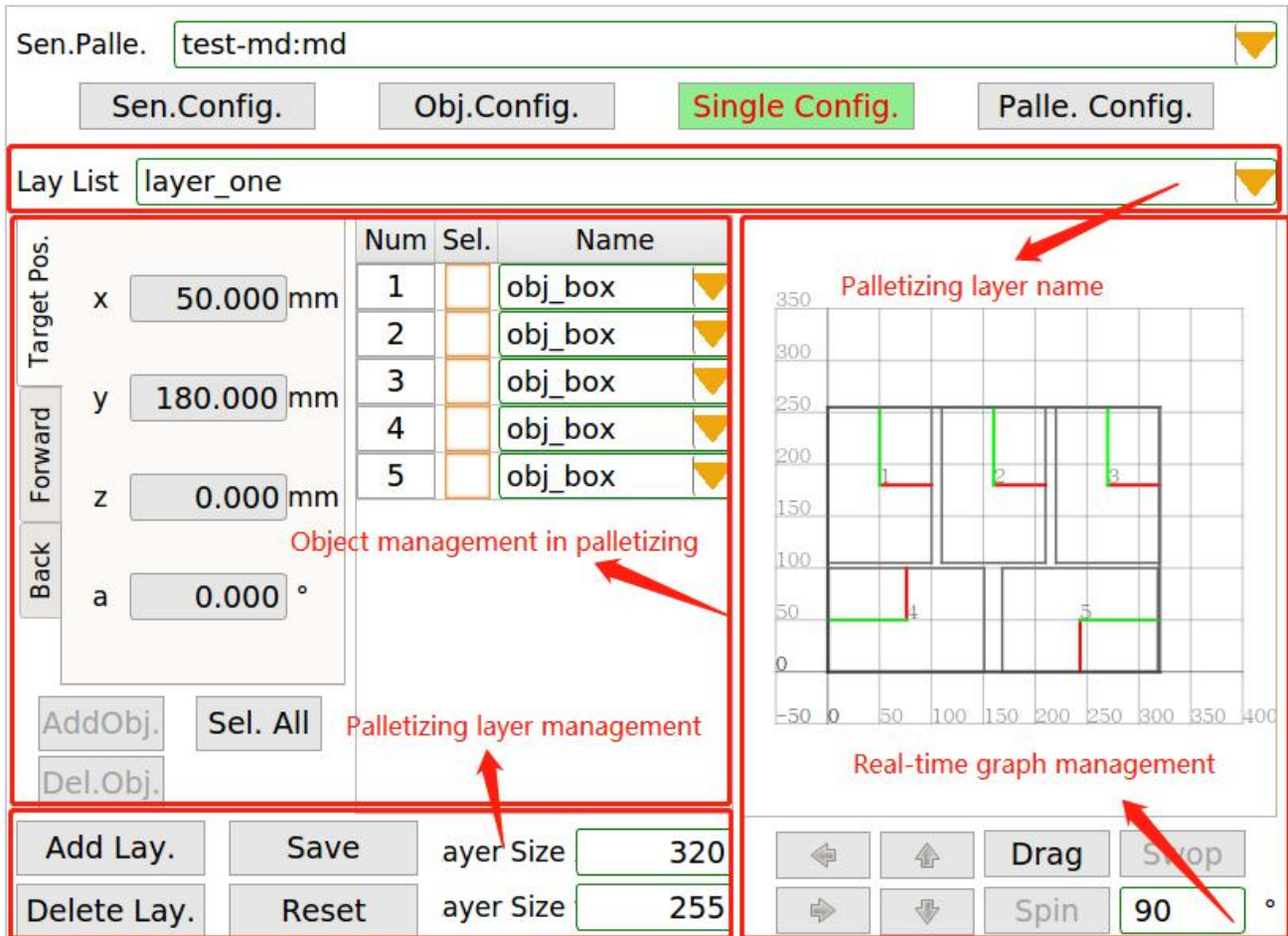


Figure 16.6 tt:SeniorPalletizer Single stack configuration interface

① Stacking layer management

- Layer list: Switch layers
- "New Layer" button: When creating a new layer, you need to enter the layer name. The layer name cannot be the same as other layer names in the current layer array, and the variable naming rules are observed. Immediately, no need to hit the save button.
- "Delete Layer" button: Delete a stack layer. Immediately, no need to hit the save button.
- "Save" button: Save the configuration of the object in the current modified layer. If the configuration is different from the saved configuration of the current layer, the font color of the save button will be red
- "Reset" button: Resets the object configuration of the current layer.

- "Layer size" : refers to the length of the current stacking layer in the X and y directions.

② Object management in palletizing layer

- "Add Object" and "Delete Object" buttons: Add or delete an object to the layer. After adding or deleting an object, it will be compared with the current saved configuration. If it is the same, it does not need to be saved. Different, save button turns red. You can delete multiple objects, add only one, and delete only the selected object. At least one object can be retained after deletion
- Target point, Front point, Back point: You can modify the target point, front point, back point, and offset point of the selected object. After the modification, the configuration is compared with the saved configuration. If the configuration is the same, you do not need to save the configuration.
- Object number: indicates the order in which objects are placed.
- Check box: Check or uncheck an object. If selected, the TCP point of the selected object is different, and the font of the edit box is red. The default font is black
- Object name: Object name, object name can be selected from the object name list.
- "All selected" or "Unselected" button: Used to select or unselect an object, which will affect the display of the TCP point value on the left.
- Static button and check box effect the color display of the left TCP point, specifically:

When an object is selected through the selection box, the TCP point on the left will display the corresponding position value. When multiple objects are selected, the value of the last object will be displayed by default if the button is selected. If selected in sequence with the checkbox, the last clicked object will be displayed. When unselected, the last selected object is displayed by default, and when not selected, the first object is displayed by default, and all TCP edit boxes cannot be edited.

4. Palletizing layer queue configuration interface: used to configure SeniorPalletizer layer queue, and can set the position and Angle of palletizing layer; You can select and configure by palletizing layer name, as shown in the following figure. It is divided into layer queue management and real-time graph management. The real-time graph management part is introduced in the next section.

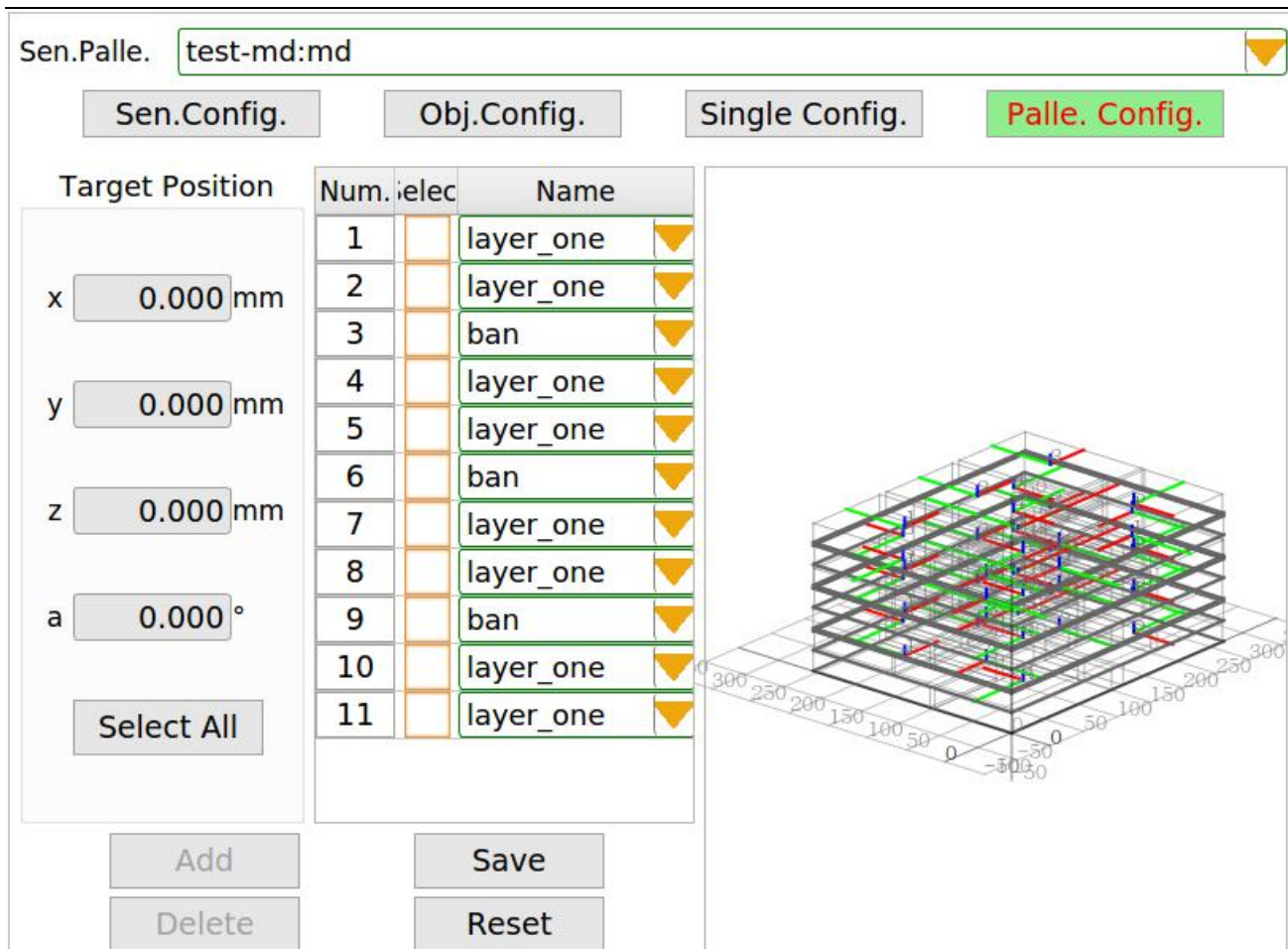


Figure 16.7 tt:SeniorPalletizer Palletizing queue configuration interface

■ Real-time map instructions

1. **Single layer palletizing configuration interface graphics**, the use of real-time graphics is divided into object coordinate management, object list management, object graphics management, object graphics button management four parts, as shown below.

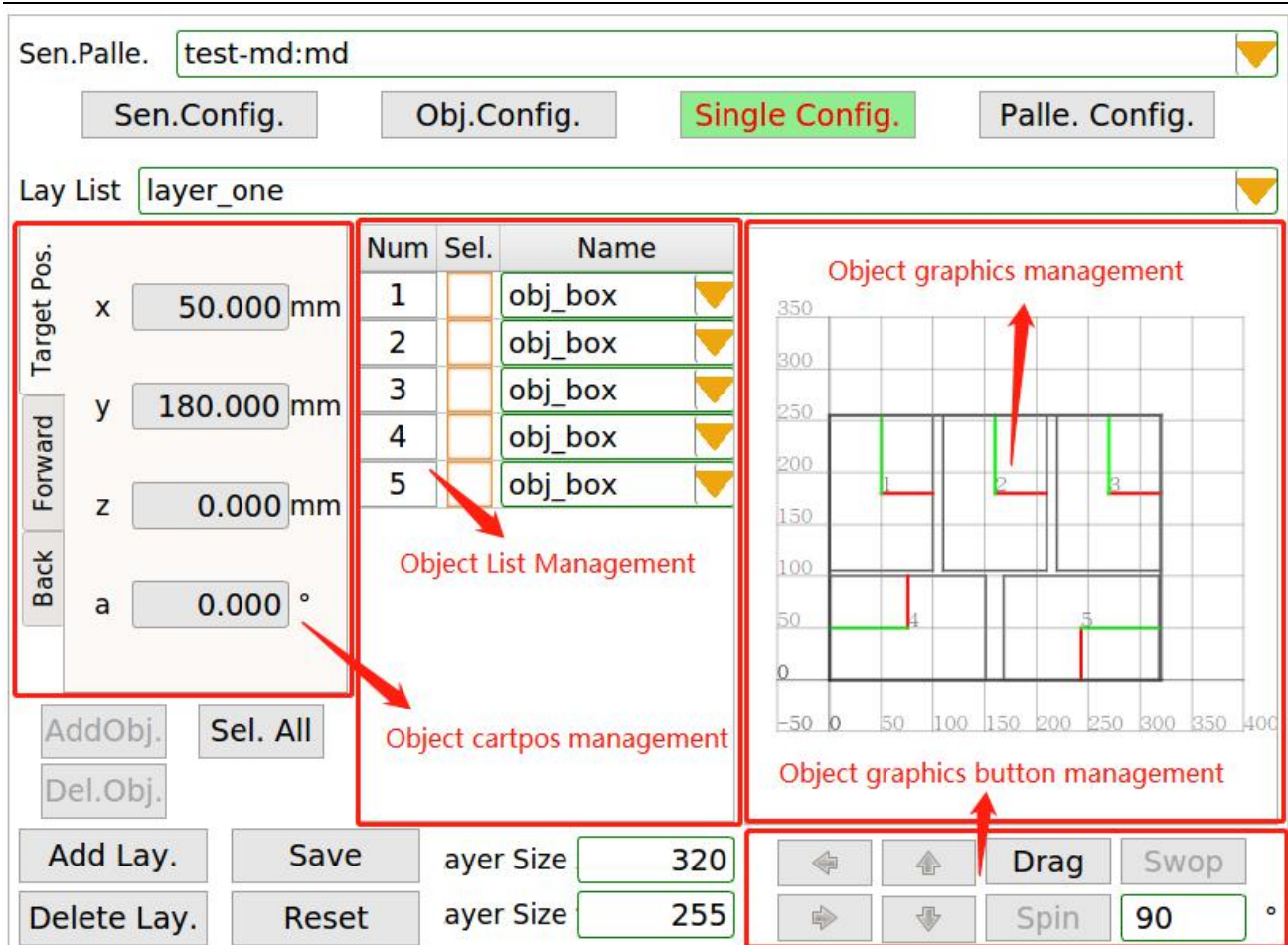


Figure 16.8 Single palletizing configuration interface graphics

- The default state is gray, and the selected state is green.
- Relation between actual coordinate system and real-time graph coordinate system:

Table 16.1 Real-time graph graphic description

	Describe
Actual coordinate system	Corresponding to the object coordinates management part, its target value affects the object graphics management part of the location of the graphics display
Real time graph coordinate system	This corresponds to the object Graphics management section
Corresponding relations between	The adaptive scale scales the actual coordinates to real-time graph coordinates

- Modify "Target Coordinates" : Modify "Target Coordinates" in the object coordinate management section. Click "Refresh Graphics" button to update the graphics display in the object graphics management section in real time.
- Status Selected: The object state in the object list management section corresponds to the object state in the object graph management section, and is updated in real time. If the object is selected in the object list management section, the corresponding object will change from gray to green. If you deselect it, it goes back to gray. Multiple selection is supported.
- "Layer size" : The size of the tray on which objects are placed, indicated by a black quadrilateral.
- Adaptive scaling: In the process of object movement, the graphics of object graphics management will adapt to the scale according to the size and position of the object to ensure that the object does not exceed the bounds.
- "Up, down, left and right" movement: the selected object can be moved up, down and left, and real-time modification of object coordinates management part of the object coordinates, support button long press operation.
- "Rotate" button: rotate the degree in the editable box clockwise.
- "Swap" button: Exchange the index of two objects, that is, exchange the object number in the object list management section.
- "Drag" button: The "drag" button supports the dragging operation only when it is in the pressed state. After being pressed, the state remains unchanged. Press again to cancel the dragging state. Only selected objects can be dragged. Dragging an animal changes the position of the graph and modifies the "target coordinates" in the object Coordinates Management section.

2. Palletizing layer queue configuration interface graphics. The use of real-time graphics can be divided into three parts: palletizing layer coordinate management, palletizing layer list management and palletizing layer queue graphics management, as shown in the figure below.

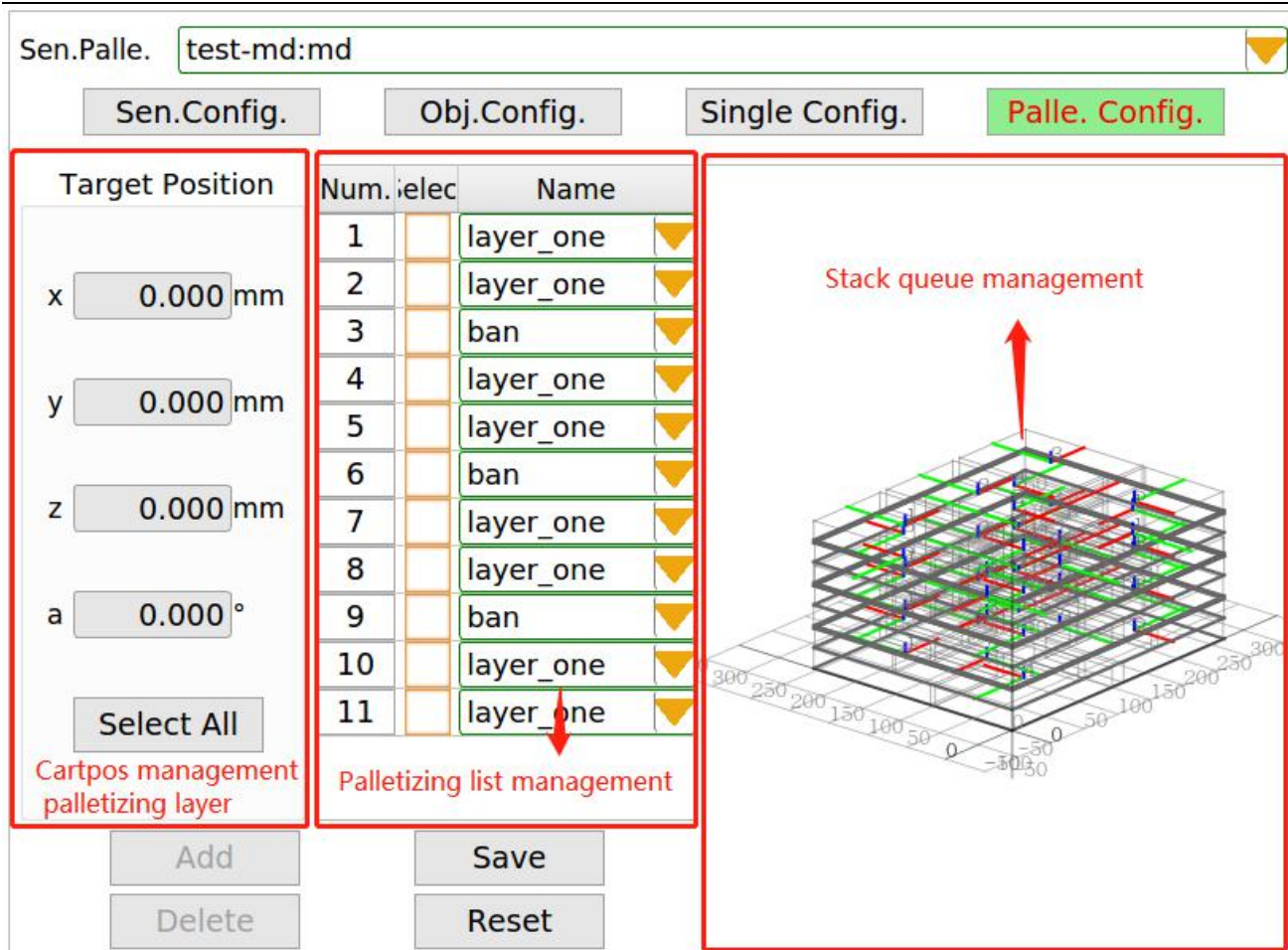


Figure 16.9 Palletizing queue configuration interface graphics

- 实际坐标系与实时图坐标系关系：

表 16.2 实时图图形描述

	Describe
Actual coordinate system	Corresponding to the coordinate management part of the palletizing layer, its target value affects the position display of the graphs in the queue graphics management part of the palletizing layer
Real time graph coordinate system	This corresponds to the palletizing queue graphics management section
Corresponding	The adaptive scale scales the actual coordinates to

relations between

real-time graph coordinates

- Modification of "Target Coordinates" : Modify "target position" in the coordinate management section of palletizing layer. Click "Refresh Graphics" button to update the graphics display of palletizing layer queue graphics management in real time.
- Status Selected: The palleting status in the palleting list management section corresponds to the palleting status in the palleting queue graphics management and is updated in real time.
- Adaptive scaling: part of the graphics in palletizing queue can adapt to the scaling according to the size and position of palletizing layer to ensure that the palletizing layer does not exceed the boundary.

16.2.3 Fast configuration

Quick Configuration has only one shape, but there are multiple layers, but the configuration of each layer is the same, as shown in the following figure.

Note: quick configuration will delete the original configuration, please use caution!!

Sen.Palle. test-mdd:mdd

Fast Config.

Palletizer Sequence: XY x positive dir y positive dir

Object length : 0.000 mm Object length : 0.000 mm Object length : 0.000 mm

Number and offset distance in X Y Z direction

number x	1	number y	1	number z	1
offset x	0.000 mm	offset y	0.000 mm	offset z	0.000 mm

Front-Pos. Config.

x	0.000 mm	y	0.000 mm	z	0.000 mm	a	0.000 °
---	----------	---	----------	---	----------	---	---------

Back Pos. Config.

x	0.000 mm	y	0.000 mm	z	0.000 mm	a	0.000 °
---	----------	---	----------	---	----------	---	---------

Figure 16.10 Palletizing queue configuration interface graphics

- "Quantity and offset distance in XYZ direction" : quantity XY refers to the quantity in XY direction respectively, quantity Z refers to the number of layers; Offset XYZ refers to the shift in the xyz direction of the coordinate system respectively.
- Front point configuration and Back point configuration: indicates the front and back points of all objects.

16.2.4 SeniorPalletizer example procedures

```
6   SetCartSys(md_cart_sys);
7   Line(md.front_position, dyn, tran_50);
8   Line(md.palletizer_position, dyn, tran_50);
9   Line(md.back_position, dyn, tran_50);
10  ▼ IF md.is_full THEN
11      ResetPalletizer(md);
12  ▼ ELSE
13      NextPalletizer(md);
14  END_IF;
```

Figure 16.11 SeniorPalletizer program example

Line 6: Switch to the palletizing system

Line 7: Move to the point before stacking

Line 8: Move to the palletizing point

Line 9: Move to the point behind stacking

Lines 10 to 14: If the stacking is full, the stacking will be reset; If not stamped, the next palletizing point will be executed.

Note:

(1) The pre-stacking point, post-stacking point and entrance point of the palletizing are not set in the sample program. Please refer to the description of palletizing variables to set according to requirements.

(2) If you do not want to place some points in the middle, please use the NextPalletizer or SetPalletizerNum command to change the serial number of the current placing.

(3) The tracking parameters related to excluding palletizing are not set in this example program, so it does not have reference.

16.3 Production

(1) New global int type variable

For example, int type variable named product

(2) Configure to use the product variable to record production

Select the variable in the function block - > production interface and enable it, as shown in the following figure:

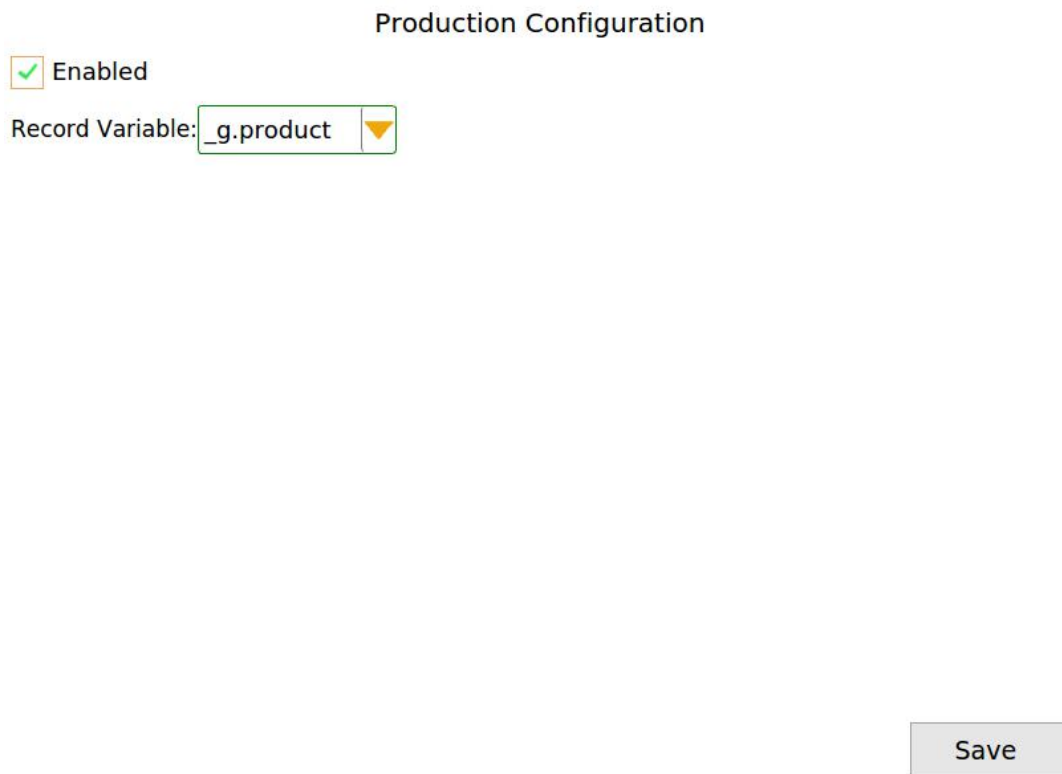


Figure 16.12 Production configuration interface

(3) Record the increase in production in the program

```
_g.product := _g.product + 1;
```

Program example

The sample program is the product program in the demo project. This program means that each time an object is placed, the production is considered to increase once, see line 28 of the program.

16.4 Variable buffer area (realize the function of modifying variables during operation)

The sample program is the cache in the demo project. The sample can change the xyz member of the pos_2_1 variable during operation.

■ Program example

```
1 SetDynamic(_g.default_dyn);
2 SetCartSys(_g.default_world_cart_sys);
3 SetTransition(_g.default_transition);
4 SetAcceleration(PARABOLA_ACC);
5 SetTool(_g.default_tool);
6 ▼WHILE TRUE DO
7   Line(pos_1_h, dyn_max, trans_50);
8   Line(pos_1_l, dyn_max, trans_0);
9   Line(pos_1_h, dyn_max, trans_50);
10  Line(pos_2_h, dyn_max, trans_50);
11  pos_2_l.x := StringToReal(GetCacheString(1));
12  pos_2_l.y := StringToReal(GetCacheString(2));
13  pos_2_l.z := StringToReal(GetCacheString(3));
14  Line(pos_2_l, dyn_max, trans_0);
15  Line(pos_2_h, dyn_max, trans_50);
16 END_WHILE;
17 >>>>EOF<<<<<
```

Figure 16.13 Buffer area program interface

■ Corresponding configuration of sample program

Variable Buffer Configuration

Index	Data	Custom Name	
1	-150	pos_2_l_x	<input type="checkbox"/>
2	0	pos_2_l_y	<input type="checkbox"/>
3	-875	pos_2_l_z	<input type="checkbox"/>
4			<input type="checkbox"/>
5			<input type="checkbox"/>
6			<input type="checkbox"/>
7			<input type="checkbox"/>
8			<input type="checkbox"/>
9			<input type="checkbox"/>
10			<input type="checkbox"/>

Figure 16.14 Variable buffer configuration interface

16.5 Hard trigger

Hard trigger is to generate a rising edge signal. Output_io is the output port, input_io is the input port, input_io is only useful when IO_RISE is selected. Hard trigger includes the following four types: IO_RISE, TIME_PERIOD, DISTANCE and POSITION.

■ Application example 1

(1) Configuration settings

▼ ■ hardtrigger:HardTrigger

enable:BOOL	TRUE
output_io:IO	dout1
input_io:IO	din1
encoder:Encoder	Encoder0
signal_type:TriggerSignalType	IO_RISE
signal_value:REAL	0.000
delay_type:TriggerDelayType	NONE
delay_value:REAL	0.000
reset_type:TriggerResetType	IO_DOWN
reset_value:REAL	0.000

Figure 16.15 Hard trigger configuration diagram

(2) Practical application:

The object is on the conveyor belt, the encoder records the distance, and the camera takes pictures. Corresponding to the above configuration parameter settings, the camera uses the IO_RISE trigger mode to take pictures. When din1 changes from false to true, dout1 generates a rising edge signal trigger, and when din1 changes from true to false, dout1 is reset.

Application example 2

(1) Configuration settings

▼ ■ hardtrigger:HardTrigger

enable:BOOL	TRUE
output_io:IO	dout1
input_io:IO	
encoder:Encoder	Encoder0
signal_type:TriggerSignalType	TIME_PERIOD
signal_value:REAL	300.000
delay_type:TriggerDelayType	TIME_PERIOD
delay_value:REAL	5.000
reset_type:TriggerResetType	TIME_PERIOD
reset_value:REAL	150.000

Figure 16.16 Hard trigger configuration diagram

(2) Practical application:

TIME_PERIOD hard trigger type. Corresponding to the above configuration settings, trigger once at 300ms, generate a rising edge signal with a delay of 5ms (dout1 changes from false to true), and reset after 150ms of high level (dout1 changes from true to false). As shown below.

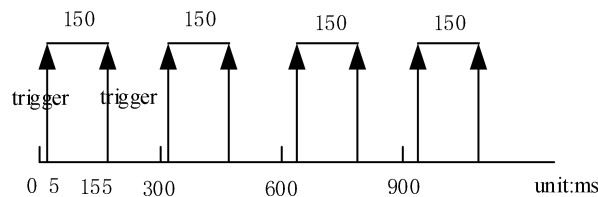


Figure 16.17 Example diagram of hard trigger

Application example 3

(1) Configuration settings

▼ ■ hardtrigger:HardTrigger	
enable:BOOL	TRUE
output_io:IO	dout1
input_io:IO	
encoder:Encoder	Encoder0
signal_type:TriggerSignalType	DISTANCE
signal_value:REAL	150.000
delay_type:TriggerDelayType	DISTANCE
delay_value:REAL	5.000
reset_type:TriggerResetType	TIME_PERIOD
reset_value:REAL	20.000

Figure 16.18 Hard trigger configuration diagram

(2) Practical application:

DISTANCE hard trigger type: Corresponding to the above configuration settings, every 150 code value is a trigger cycle, and a rising edge signal is generated after each cycle is delayed by 5 codes (dout1 changes from false to true), then reset after 20ms (dout1 changes from true to false)). as the picture shows:

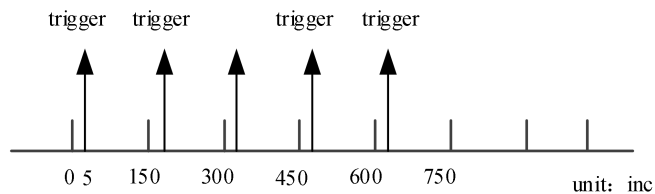


Figure 16.19 Example diagram of hard trigger

Application example 4

(1) Configuration settings

▼ ■ hardtrigger:HardTrigger

enable:BOOL	TRUE
output_io:IO	dout1
input_io:IO	
encoder:Encoder	Encoder0
signal_type:TriggerSignalType	POSITION
signal_value:REAL	2000.000
delay_type:TriggerDelayType	DISTANCE
delay_value:REAL	5.000
reset_type:TriggerResetType	DISTANCE
reset_value:REAL	30.000

Figure 16.20 Hard trigger configuration diagram

(2) Practical application:

POSITION hard trigger type. For example: in a carousel, a specific position is triggered. Corresponding to the above configuration settings, at a specific position, the encoder code is triggered when the encoder code value is 2000, and the rising edge signal is generated after the delay distance is 5 code values (dout1 changes from false to true), and reset after 30 code values (dout1 changes from true to false).

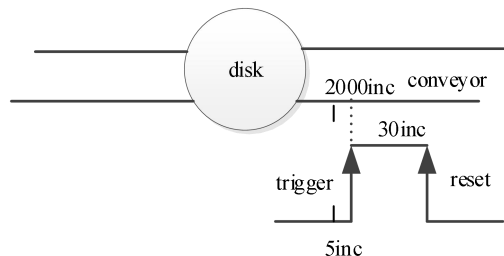


Figure 16.21 Example diagram of hard trigger

16.6 Area monitoring

(1) New area monitoring variable

For example, create a new area monitoring variable named workarea (see area monitoring for the configuration interface).

(2) Set area variable related parameter configuration (for specific parameter description, please refer to [Work Area](#))

Parameter configuration setting method:

(1) Starting point of working area:

Box shape: It is a corner point of a cube, which is the smallest point. (x is the smallest, y is the smallest, z is the smallest)

CYLINDER shape: is the center of the bottom surface.

(2) Coordinate expression:

BOX shape: dx: distance from the starting point along the x direction

dy: starting from the starting point, the distance along the y direction

dz: starting from the starting point, the distance along the z direction

CYLINDER shape:

H: the height of the cylinder (the distance along the positive direction of the z-axis)

R: the radius of the cylinder

Set the working area of the robot according to your needs. See example 2 for specific area settings.

(3) Workarea priority:

Prohibited Entry Area > Prohibited Leaving Area > Work Area

If the three areas exist at the same time, the robot must not enter the prohibited area to work, and then cannot leave the prohibited area, and finally work in the work area.

Application example 1 (WORK_AREA)

As shown in the figure, the cylinder is the working area of the robot.

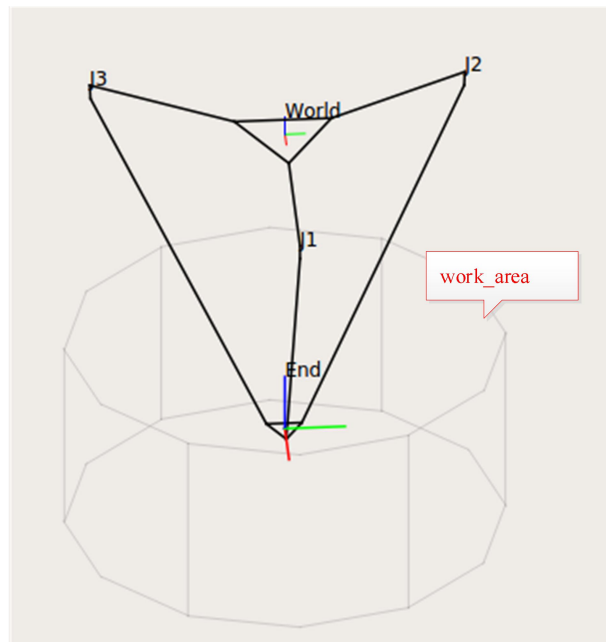


Figure 16.22 Example of working area

Application example 2 (FORBIDDEN_OUT)

As shown in the figure, the black rectangular parallelepiped area marked by the label is the forbidden_out area.

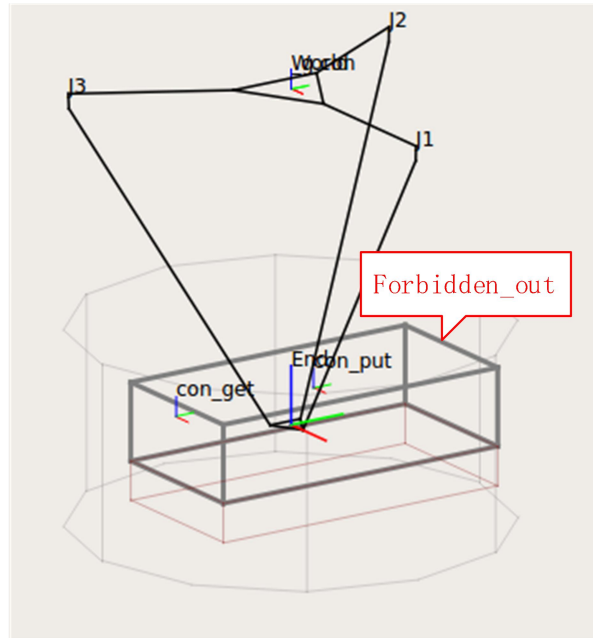


Figure 16.23 Example diagram of prohibiting leaving the work area

For example, the robot grabs or places objects on the conveyor belt, and the two conveyor belts and the area between them (the area in the dashed frame) are prohibited areas.

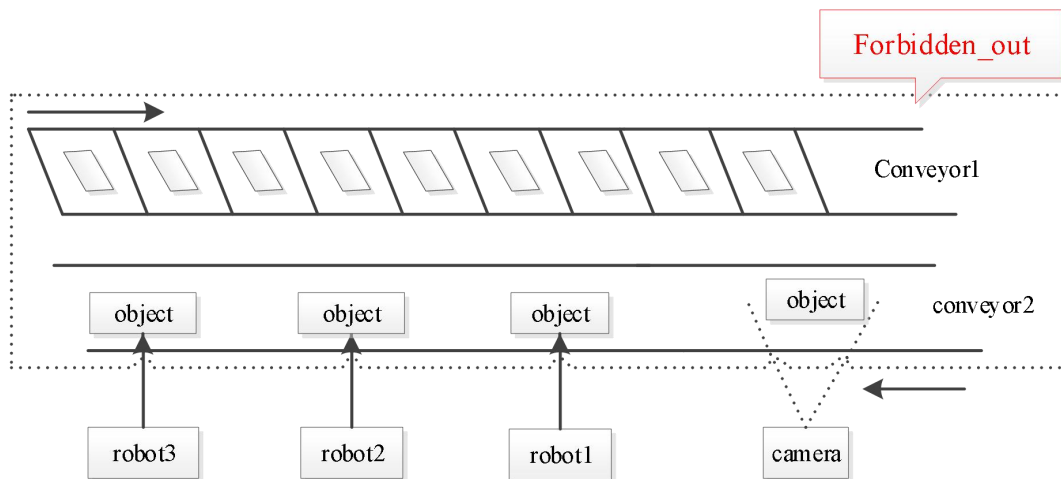


Figure 16.24 Application diagram of prohibiting leaving the work area

Application example 3 (FORBIDDEN_AREA)

Note: There is no entry area, and it is not in the no entry work area.

As shown in the figure, the red rectangular parallelepiped area marked by the label is a no-entry area.

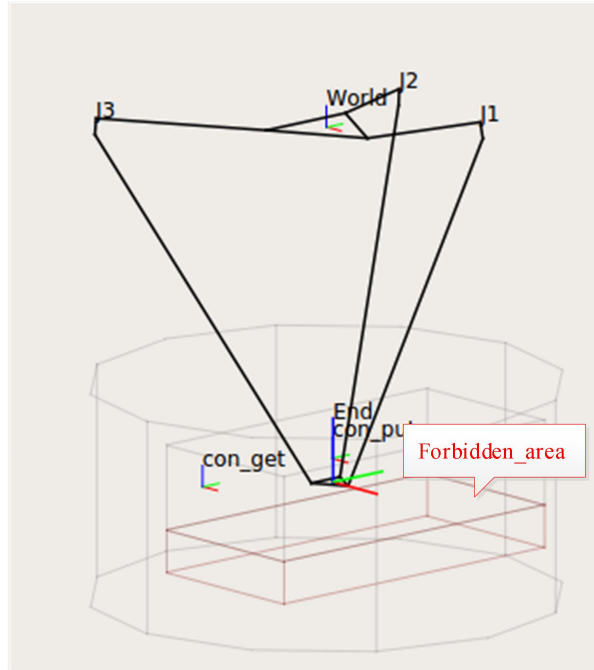


Figure 16.25 Example diagram of prohibited work area

For example, a conveyor belt with a grid, where the baffle of the grid (the red line) is the prohibited area of the robot.

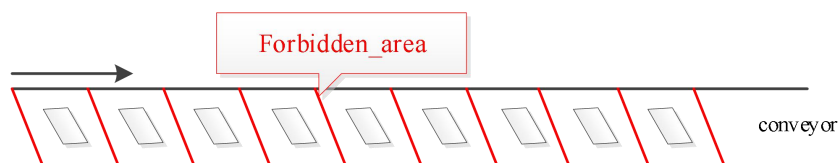


Figure 16.26 Application diagram of prohibited working area

16.7 Object Allot

To understand the function of object diversion, it is necessary to combine the definition of object diversion variable, the application example diagram and text description of object diversion.

(1) New object shunt variable

Conveyor belt variables are required to use object flow variables. For example: create a new conveyor belt variable named con, and then create a new object distribution variable called objectallot.

(2) Object shunt configuration parameter setting (for the specific meaning of the parameters, please refer to the object shunt variable ObjectAllot)

■ Application example 1 (single allot)

Taking MAXMUM as example, the robot 1 grabs the objects on the conveyor belt with its maximum capacity, and the rest is diverted to the robot 2 to grab the objects. Taking GROUPING as an example, set output_num to 2 and total_num to 5. Robot 1 grabs 3 of them, and the remaining 2 are diverted to robot 2. Take RATIO_MAXIMUM as an example, first divide the flow according to the proportion, and then divide the flow according to the maximum capacity. Assuming that the shunt ratio is X, robot 1 grabs objects in a scale of 1-X. If robot 1 has not completed the grab of 1-X ratio, it will be diverted to robot 2 to grab.

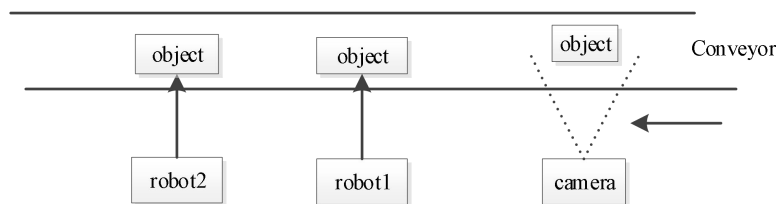


Figure 16.27 Example diagram of single shunt application

Application example 2 (double allot)

Take the RATIO type as an example. For grabbing conveyor belt b: Robot 2 divides the flow for robot 3. Assuming that the flow ratio is X, robot 2 grabs objects on the conveyor belt b according to the 1-X ratio, and places them on the conveyor belt a, and scales X. The objects in are diverted to the robot 3, and the robot 3 grabs the objects and places them on the conveyor belt a. For placing conveyor belt a: robot 3 divides grids for robot 2, assuming that the distribution ratio is Y, robot 3 places objects on the grids of conveyor belt a according to the 1-Y ratio, and divides the grids of Y ratio to robot 2. (Robot 2 is robot 3 shunting objects, and robot 3 is robot 2 shunting conveyor grid)

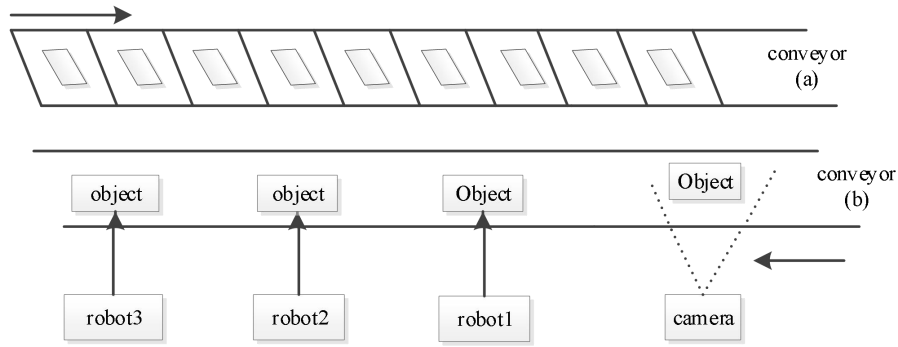


Figure 16.28 Example diagram of dual shunt application

16.8 External axis

The external shafts are conveyor belts, stop bars, and equipment with motors other than robots. To understand the function of the external axis, it is necessary to combine the configuration of the external axis (see the external axis) and the text description.

■ Application example (take conveyor belt as an example)

Instruction Type:	Vel.Control	▼	<p>Trigger Conditions:</p> <p>DIO: X1=True And</p> <p>Add</p> <p>Edit</p> <p>Delete</p> <p>Stop condition:</p> <p>DIO: X1=False And</p> <p>Add</p> <p>Edit</p> <p>Delete</p> <p>Confirm Cancel</p>
External axis:	ExJ2	▼	
Relative Position:	<input type="checkbox"/>		
Position:	0.000	°	
Velocity:	10.000	°/s	
Acceleration:	0.000	°/s ²	
Jerk:	0.000	°/s ³	
Finish Dout:	Y1	▼	
Run Dout:	Y2	▼	

Figure 16.29 Example diagram of external axis configuration

According to the above configuration, when x1 changes from false to true, a trigger signal is generated and the conveyor belt starts to move at a speed of 10°/s; when x1 changes from true to false, a reset signal is generated and the conveyor belt stops moving. When running Dout is true, the conveyor belt is in running status



微信公众号

atomrobot[®]

阿童木机器人

☎ 400-653-7789

🔍 www.tjchenxing.com

天津总部

辰星（天津）自动化设备有限公司

地址：天津滨海新区南海路156号

邮箱：sales@tjchenxing.com

Tianjin Headquarters

Chenxing (Tianjin) Automation Equipment Co., Ltd.

Address: No. 156 Nanhai Road, Binhai New District, Tianjin

Email: sales@tjchenxing.com

江苏子公司

辰星（苏州）自动化设备有限公司

地址：江苏省苏州市吴江经济技术开发区联福路139号

邮箱：maguosong@szchenxing.com

Jiangsu subsidiary

Chenxing (Suzhou) Automation Equipment Co., Ltd.

Address: No. 139 Lianyang Road, Wujiang Economic and Technological Development Zone, Suzhou City, Jiangsu Province

Province

Email: maguosong@szchenxing.com

东莞办事处

地址：东莞市虎门镇体育路555号

Dongguan Office

Address: No. 555 Tiyu Road, Humen Town, Dongguan City