ITRI

Mechanical arm

Operation Manual

Robot Arm Manual

Robot Application Software

(corresponding to version 3.0.2.2)

Industrial Technology Research Institute Robot Application Systems Division

2013/10/3

Safety warning sign

1. Do not enter the robot's range of motion during robotic operation or movement.
   Do **NOT** enter workspace of the robot arm during operation.
2. The range of movement of the robot arm should be clearly marked to prevent accidental entry by personnel.
   Workspace should be clearly marked to prevent people entering.
3. Turn off the power before setting or adjusting the robot arm.
   Before setting up or adjusting the arm, turn off robot arm.
4. When performing the robot arm operation, please confirm the emergency switch position. When an emergency occurs, press and immediately Stop.
   Make sure the emergency button reachable by your hands. When Emergency occurs, push the button immediately.
5. When teaching the robot arm, please confirm the surrounding environment to avoid interference.
   Before using the robot arm, make sure the surrounding Environment is clear and free of collision.
6. Do not operate the robot arm in a damp place.
   Do not operate the robot arm under a wet environment.
7. Do not make changes to system parameters without the instructions of the engineer.
   Do not alter the system parameters.
8. Do not make any hardware changes without the instructions of the engineering staff.
   Do not alter the setting of the hardware.
9. Do not turn off the main power supply before the normal shutdown.
   Before you normally stops the robot, do not turn off the main power.
10. The operator should read the manual carefully to avoid misuse.
    You should read this manual before using the robot.

Witness

Page 5

Page 8

Page 10

Figure

Page 17

Chapter 1 Mechanical Arm Hardware Inspection and Mainte

A. Implement daily inspections and regular inspections

(1) Be sure to carry out daily inspections and periodic inspections, and confirm whether the robots and related machines have before the operation. abnormal situation. If there is an abnormality, please perform repairs and other necessary measures immediately.

(2) Please record the specific contents of regular inspection and implementation of maintenance, and keep it for more than 3 years.

B. Check the items before starting the arm every day.

No.	Checkpoint	Inspection method	power supply status	Judging criteria	Determining bad follow-up
1	**power cord**	Visually, hand grasping	OFF	No loose, dirty dirt	Insert it correctly
2	**Control box connection line, Cable and plug**	Visually, hand grasping	OFF	No loose, dirty Scale, damage, crack mark	Correctly inserted, damaged or replaced Components
3	**safety fence**	Visual guardrail	OFF	Off position	Correctly pull the guardrail
4	**liquid crystal display**	Visual inspection	ON	normal display	Repair or replacement
5	**keyboard and mouse**	Trial operation	ON	Normal effect	Repair or replacement
6	**controller signal light**	visual	ON	Light	Repair or replacement

Page 18

7	**emergency switch**	Trial switch	ON	emergency stop	Repair or replacement
8	**security door or guardrail**	Touch safely Off, visual barrier	ON	emergency stop	Repair or replace, properly pull Guardrail

Rotating 2, 3, 4,

	Motor brakes are released				
9	**turn off**	5 axes, press loose switch	OFF	Motor shaft is loose	Repair or replacement
10	**arm body appearance**	Visual, hand-touch	OFF	No dirt, damage Injury, crack	Repair or replacement
11	**motor exterior**	Visual, hand touch	ON	Abnormal temperature rise or shock	Observe, repair or replace

C. Quarterly inspection items

No.	Checkpoint	Inspection method	power supply status	Judging criteria	Follow-up
1	**Arm body fixing screw** **wire**	Torque wrench Trial lock M10 screw wire	OFF	Need up to 65 N-m	Lock
2	**Control box cooling fan** **And filter**	Visual inspection	OFF	Normal operation, no Dirt	Repair or remove dirt

3	**pneumatic directional valve**	I/O point operation	ON	Normal ON/OFF	repair or replacement

The figure below shows the mechanical arm body fixing screw configuration and air pressure valve.

Figure 1 Robot arm fixing screw

Figure 2 Mechanical arm air valve

D. Check items every two years

No.	Checkpoint	Inspection method	Judging criteria	Follow-up
1	**rechargeable battery**	OFF status	Direct replacement	Encoder position data is stored, straight Replace the new battery
2	**J4, J5 timing belt**	visual inspection	OFF	Tooth wear or lack tooth Replacement of new products

The figure below illustrates the replacement of the rechargeable battery.

Figure 3 Mechanical arm battery replacement

The figure below shows the location of the rechargeable battery in the robot body.

Figure 4 Mechanical arm battery body position

The following figure illustrates the replacement of the timing belt in the robot body.

Figure 5 Fifth shaft belt replacement

Figure 6 Fourth Axis Belt Replacement

E. Troubleshooting the robot arm

(1) If the arm is faulty, loosen the motor brake, manually move each axis to the normal position, and then use the Home function.

Revert to the operating origin.

(2) If the arm failure is serious and the machine origin is ran away, please contact the manufacturer for disposal.

Chapter II Mechanical Arm Control Box

Controller

A. Appearance of electric control box

Controller Outlook

Figure 8 shows the size of the electric control box, which indicates the length, width and height of the electric control box, in mm. Show belov

It shows the appearance of the electric control box and the functions of each button and socket.

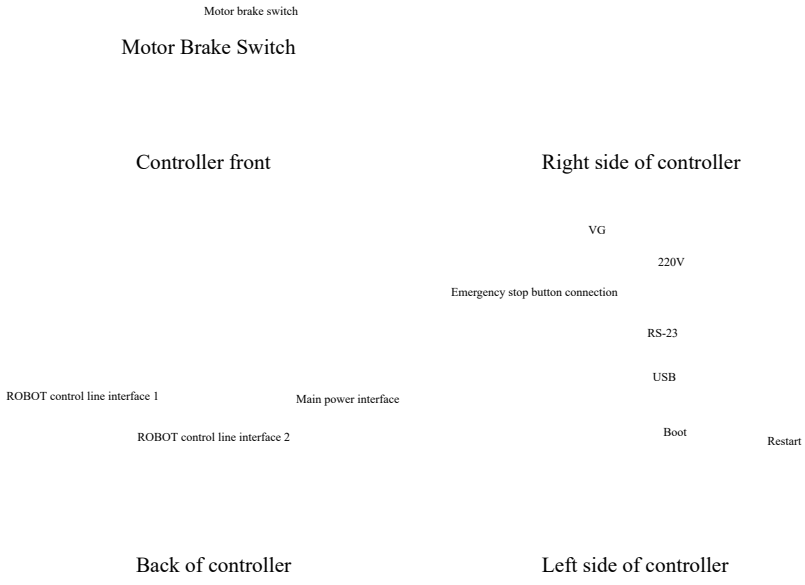

Figure 7 Appearance of the electrical control box

back

460

480

Right side

left

350

positive

Figure 8 Electric Control Box Size

B. Inside the electric control box

Figure 9 shows the internal component layout of the electrical control box, which is represented by each viewpoint. **Wrong! Find no To the reference source.** Shows what the inside of the electrical control box actually looks like.

Inside the left side **Inside the main entrance** Inside right

Left side **Front door** **Right side** **Behind**

Figure 9 Internal diagram of the electrical control box

Inside the controller Inside the controller

Right side of controller Remote control/teacher

Figure 10 Electrical control box internal view

C. Power on

How to turn on the cotroller

● Check whether the power plug is loose before starting the power supply.

● The input power of this controller is 3-phase 220V AC power supply, the maximum current requirement is 30 amps, before starting.

Please confirm that the power is correct.

➢ **Startup steps: (Note: Some versions indicate that the automatic start, you can skip this step)**

Procedures

(1) Open the controller door panel:

Refer to Figure 11

Press the door handle and open the active door, as shown in Figure 11.

(2) Pull the main power switch in the upper left corner up:

Refer to Figure 12

Pull the main switch up and turn on the power, as shown in Figure 12.

(3) Start the computer

Refer to Figure 13

Press the ON button to turn on the computer, as shown in Figure 13.

(4) Wait for the computer to complete the boot process, about 2 to 5 minutes.

Wait for 2-5 minutes for
The computer to complete
Boot process.

(5) Click on ITRI RAS to start using (boot automatically start).

Press the door handle, open

Open door

Press this button
And open the door.

Figure 11 Opening the controller door panel

Turn on this switch
(Main Power)

Move the main switch up

Pull on, turn on the power

Figure 12 Push the main power switch in the upper left corner up

Press the ON button,
Turn on the computer

Press "ON" button

Figure 13 Start the computer diagram

D. Input point configuration

● The common reference point (COM) of the input point has a voltage of +24V, and the maximum current of a single point must not exceed 0.1A.

From the first point to the 48th point, the system is used internally. Do not change it arbitrarily. From 49th to 64th.

For external use.

➢Configuration **steps:**

(1) Turn on the main power activity door:

Press the door handle and open the active door, as shown in Figure 14.

(2) Turn off the controller main power:

Pull the main switch down and turn off the power, as shown in Figure 15.

(3) Open the electric control box movable door:

Press the door handle and open the active door, as shown in Figure 16.

(4) Wiring in accordance with EMI electromagnetic protection standards

Input point configuration area (I49~I64), as shown in Figure 17.

Press the door handle, open
Open door

Figure 14 Turning on the main power active door

Page 27

Lower the main switch

Pull, turn off the power

Figure 15 Shutdown of the controller main power supply

Press the door handle, open

Open door

Figure 16 Opening the door of the electric control box

Input point configuration

District (I49～I64)

Figure 17. Input point configuration diagram

E. Output point configuration

● The voltage of the common reference point (COM) of the output point is 0V, and the maximum current of a single point must not exceed 0.1A.

Page 28

From the first point to the 48th point, the system is used internally. Do not change it arbitrarily. From 49th to 64th.

For external use.

➢Configuration **steps:**

(1) Turn on the main power activity door:

Press the door handle and open the active door, as shown in Figure 14.

(2) Turn off the controller main power:

Pull the main switch down and turn off the power, as shown in Figure 15.

(3) Open the electric control box movable door:

Press the door handle and open the active door, as shown in Figure 16.

(4) Wiring in accordance with EMI electromagnetic protection standards:

Output point configuration area (O49 ~ O64), as shown in Figure 18.

Output point configuration area

(O49～O64)

Figure 18 Output point configuration diagram

Chapter III Robotic Software Control

Software

A. Start using the ITRI RAS operating software

How to use ITRI RAS robot arm operating software

After the robot is set up, you want to start using the RAS system. First you need to find the following figure and double-click it.

You will see the main operation screen (automatically started when the preset is turned on).

Double click this icon on
The desktop.

Figure 19 RAS start pattern

Press the ESC key on the keyboard to exit the program. You can press ESC to quit the program.

In addition, double-click the image below to connect directly to the system program folder for advanced user operations or to modify the system.

Figure 20 Link RAS System Folder

B. Introduction to software functions

Here are the basic functions of the software. The main control blocks are divided into four parts, which are as follows:

The window of the software is divided into 4 parts:

1. Program operation: responsible for the operation of the robotic arm program, including file processing and operation, suspension,
Program Operation (Only used when you code inside this software.)
End and other command buttons.

2. Status window display: displays the current system status and contains an error message when an exception occurs.
State Window: Shows current system state and error messages.

3. Function operation main screen: Each function operation screen varies with the function selection.
Main operation window: Varies with the function you choose.

4. Function operation menu: Select the button for different function operation, a total of 8 functions, each function is as follows:
Function menu: Select different functions in this software.
There are 8 functions:

a. Operation screen: It is responsible for displaying the simulation screen of the operation and the execution program and system status.
Shows the simulation of the arm and points teached before.

b. Programming: The robotic programming language can be written to manipulate the robot arm.
You can write program with the provided functions here. However,in
The assignment, we will write program outside the software.

c. Motion control: basic motion operations, including axis movement, linear movement, tool mode movement, and
You can move the robot here by rotating each axis or specify
Points in Cartesian space.
You can also teach coordinate points for use in programming.

d. IO setting: IO communication is used as input and output.
I/O Configuration: Do not chnge any setting in it.

e. Vision Module: This module can be used to perform visual positioning and object detection.
Visual Module: Not used in our case.

f. Communication module: This module can be used to communicate with external controllers, and the communication protocol is used.
Communication Module: Set the controller to communicate with
Devices or programs outside this software via Ethernet or RS232.
Ethernet network or RS232 serial transmission.

g. System and Exception Handling: Contains a history of system parameter settings, system restarts, and exceptions.
System error handler: The controller can be reboot here.

The detailed picture can be seen below (for example, type A):

Program operation Status display window, contains

Figure 21 Robot Control Software

C. System Status Window

Please refer to Figure 22

On the main function [Run Screen], we can see that there is a window on the left, which details the robot arm.

The operational information, as shown in the figure, includes the following:

1. Command sending angle (unit: degree): refers to the angle value that the software specified hardware should move to, and the actual editing

The encoder return value will have a slight error, and the normal error amount is +1~-1 pulse wave, which is +0.01~-0.01 degrees.

2. Encoder return angle (unit: degree): The angle value actually returned by the motor encoder.

3. Robot position: The position of the robot arm is converted according to the angle of the software output command.

4. System status display list (unit: mm): displays historical system status information, including system initialization,

System restart, exception information, teaching information, robotic language output information (MSG command), external communication

Information and more.

For details, please refer to the following figure:

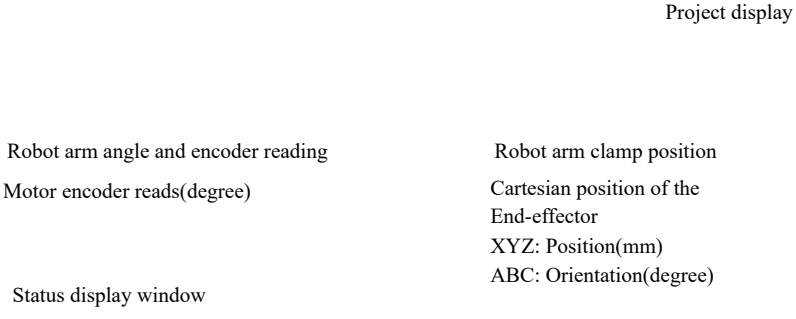

Figure 22 Robot arm status display

Page 32

D. Basic motion function operation

In this section we will cover the basic robotic arm functions, including motion, teaching, and coordinate settings.

After the robot is turned on, we can perform basic arm movements as follows:

1. Open the main program, click the [Motion Control] tab button, the main screen of motion control appears, as shown below

Single axis mode: (Figure 24)
Rotating each axis
Position mode: (Figure 25)
Moving the end-effector
In Cartesian space.
Tool mode: (Figure 26)
Also moving in Cartesian
Space, but reference to
The tool frame.

Speed acceleration coordinate status display

Motion control button

Motion control button

Teaching point operation button

Speed control button

You can change
Speed here

Status display

Teaching point list

List of teaching point

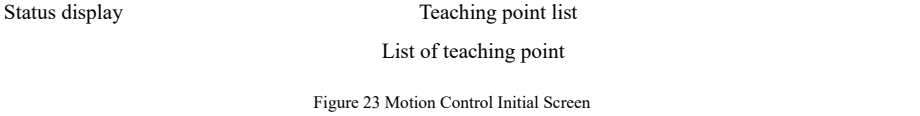

Figure 23 Motion Control Initial Screen

The motion control buttons have the following three modes of operation depending on the reference coordinate system:

(1) Point-to-point movement of multiple axes according to the axis coordinate system (the most commonly used coordinate system): Single-axis m

(2) According to the earth coordinate system (also a very common coordinate system, three-dimensional space card coordinate concept): position n

(3) According to the tool coordinate system (in special cases, the coordinate system based on the tool coordinate): Tool mode

The following figure illustrates the three coordinate system relationships (described in Type A and Type D arms, respectively)

The top-down coordinate system is the axis coordinate, the earth coordinate, and the tool coordinate:

Page 33

J5

J3

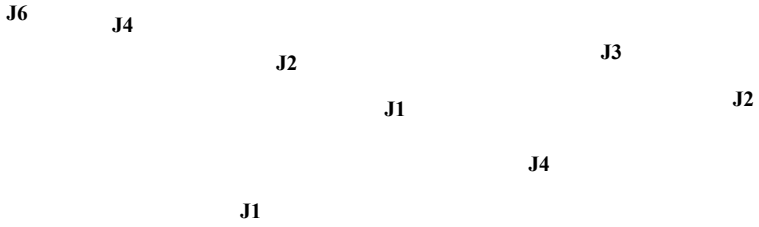

Figure 24 Axis coordinate system

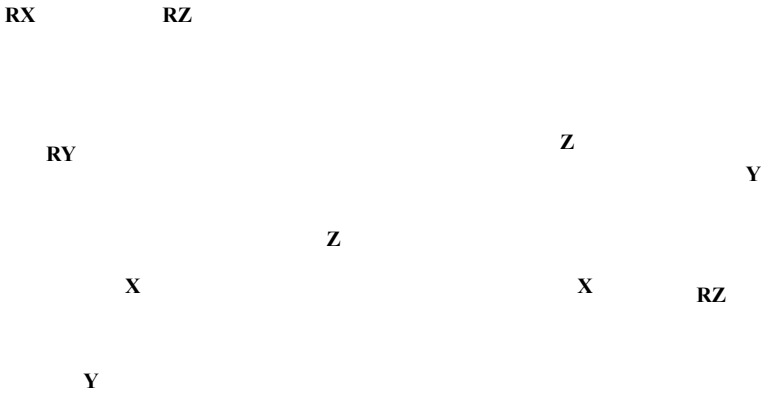

Figure 25 Earth coordinate system

Figure 26 Tool coordinate system

2. In the motion operation interface (right screen) we can divide into four parts, respectively

(1) Motion control button section

(2) Velocity acceleration state and coordinate display part

(3) Teaching point operation button

(4) List of teaching points

In the above four parts, we can press the motion control button to change the posture and speed of the robot arm, and from

The right status display interface knows the current status, and teaches and changes through the teaching point operation and the list interface.

more. Switching motion reference frame

Figure 27 Motion Control Button Interface

Change speed and speed ratio

Speed and speed ratio display

Add (subtract) speed time display

Currently selected coordinate system

Figure 28 Speed acceleration state and coordinate display interface

Page 35

Add/overwrite: Add current pose to a teaching point.
(A window will prompt to let you set the name of the point.)
Delete: Delete the selected teaching point.
Move quickly to the point: Point-to-point move to the teaching point.(Page 37)
Move straight to the point: Linear move to the teaching point. (Page 37)
Move to Home: Home the robot.

Figure 29 teach point operation button

After the button is checked, the button is executed.

Confirmation inquiry will be made during the trip.

Figure 30 List of teaching points

And right click on the teaching point list to call the teaching point operation shortcut button, including point-to-point fast moving,

Straight line movement, point sorting, point modification, etc.

3. Click [Move to Home Point] to return to the Home action. If the action is normal, the A-type D-type robot arm

The posture should be as follows (Note: If the system parameter Home point has been changed, the posture may change).

Note: Moving to the custom origin is the user-returned Home method, which will be explained in the following sections.

Axis in a straight line

Z pose

Figure 31 Robotic Arm Point Position

(due to the angle of shooting and the real

May be slightly different)

4. Perform arm single-axis motion, click the tab [single-axis mode], click the up or down arrow to move, and move

During the process, the positive direction of each axis moves as shown in the figure below.

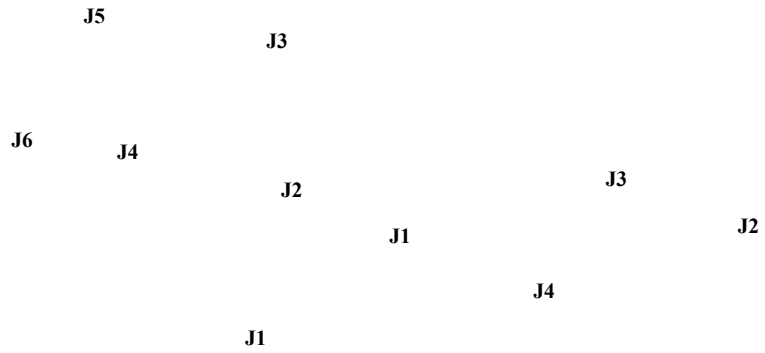

Figure 32 The arm moves axially (the direction indicated by the arrow is positive)

5. Perform a linear motion of the arm, click on the tab [Location Mode], click on the direction to move, and the D-arm is in the right direction.

The structure is called, so the diagonal direction of the first axis is the positive Y axis, and the direction of each card coordinate is as shown below.

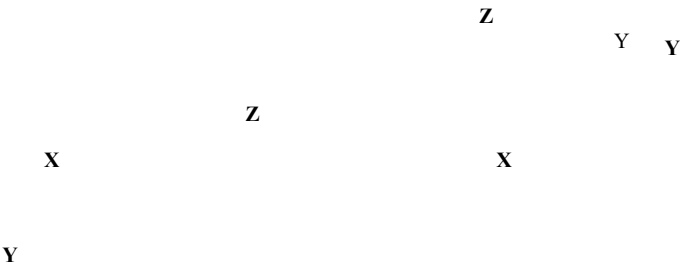

Figure 33 The robot arm moves in a straight line (the direction indicated by the arrow is positive)

6. Move the arm tool coordinate, click the tab [Tool Mode], click the direction to move, the same shift

During the movement, if the path is a normal period, this attention, D-arm tool coordinates and large

The coordinates of the ground are the same, and the coordinates of each tool are as shown below.

TX

TY

TZ

TZ

TX **TY**

Figure 34 Robot arm tool coordinate moves linearly (the direction indicated by the arrow is positive)

7. Next we will set the speed, switch the tab and find that there are two speed settings, one is called
In velocity setting, we can find two kinds of velocity, one
PtPSpeed (point-to-point moving speed), another called LineSpeed (linear moving speed), these two
Is called PtPSpeed(Point-to-point speed); the other is Line
The difference in speed is in the type of motion, so there are two types of motion:
Speed. They are differ in the way the robot moves. There are
Two categories of movement in the robot arm:

a. Point-to-point movement = fast move to point = Point-to-point Movement

b. Linear movement = Linear Movement

The point-to-point movement adopts the fast moving mode, and the interpolation point is not performed during the movement, and the moving path
Point-to-point movement does not include interpolation in
Its track. The moving track cannot guranteed to be linear.
Straight line, but no singularity problem, so this moving method is usually used during the big stroke movement, and the point
It is also free of singularity, so most long travel take
The way. The PtPSpeed is the percentage of the max motor
The speed calculation method of the point-to-point movement is calculated in percentage according to the motor setting maximum RPM (speed), fo
Rotation angular velocity. For example, PtPSpeed = 10 means
That the point-to-point movement is conducted with 10% of
Max motor angular velocity.

PtP Speed = 10, which means that the point-to-point motion motor moves at 10% speed, so the axis movement also belongs to

Point to point to move quickly.

The linear movement adopts the interpolation movement mode, that is, the interpolation point between the target point and the starting point, whicl
Linear movement interpolate many subgoals on the line
Conecting the start point and the destination.
The path is a straight line, but there are singular points. Avoid manipulating the singular position in the arm operation.
It ensures the robot arm to move linearly, but may encounter
Singularity when computing inverse kinematics.
When you reach the singular position, your arm will stop moving. Therefore, this movement is usually used on small paths of detailed operations.
The singular point stops the robot.
Is ususally adopted in short travel of the end-effecter.
Way, and how its speed is calculated in terms of how many millimeters to advance per second, such as Line Speed = 10, ie
Line Speed is linear speed of the end-effecter in mm/s.
For example, Line Speed = 10 means linear speed = 10/mms.
It means that the linear movement speed is 10mm/s, and the linear motion of the tool coordinate also belongs to this setting.

The difference between the two movement modes can be seen in Figure 35 below. The red points are the starting point and the ending point respect
Figure 35 shows the difference between these 2 movements.
In the direction of movement, the blue point is the interpolation point in the middle. Since there is an interpolation point in the middle of the linear

The moving path tends to be straight, while the point-to-point moving is no.

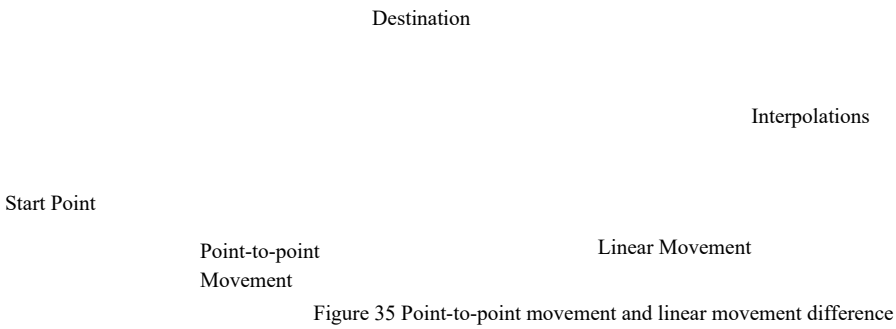

Figure 35 Point-to-point movement and linear movement difference

8. There is also a speed setting that can be set regardless of the mode, that is, Speed Rate

Ratio), so the speed percentage setting can even dynamically adjust the speed of the robot arm. The formula is [real

Interval speed = = [speed] * [speed percentage], the three speed settings can be seen in Figure 36 below.

Figure 36 Motion Control Speed Settings

E. Teaching point operation

In the teaching point operation section, the corresponding button interface is divided into four parts:

(1) Operation of variables and teaching points with coordinate setting

(2) Position movement confirmation

(3) Return to the origin

The operation of variables and teaching points with coordinates Position movement confirmation Return to origin

Figure 37 teach button function division

In the left interface teaching point operation, three types of variables can be taught, namely numerical values, strings, and seats.

Punctuation, here to introduce the operation of the coordinate point, use the axis movement or linear movement, and click the [Add / Overwrite] bu

Teach points, teach two points and use the [Fast Move to Point] button or the [Line Move to Point] button to confirm the teaching

Point, in addition, when moving, we can also directly click the axis button to directly set the value, add teaching points such as

under:

(1) Move the robot arm to the position to be taught

(2) Click the [Add/Overwrite] button

(3) Enter the point name

(4) Press OK

The method of deleting the teaching points is as follows:

(1) Select the point to be deleted in the teaching point list

(2) Click the [Delete] button

The method of overwriting the teaching points is as follows:

(1) Move the robot arm to the position to be overwritten

(2) Select the point to be overwritten in the list of teaching points

(3) Click the [Add/Overwrite] button

(4) You can see that the selected point has been entered in the dialog box, press OK directly.

In addition, in terms of location movement confirmation, it is divided into fast moving to point (corresponding to Robot Language is MOVJ).

Move the line to the point (corresponding to the Robot Language to MOVL) and move the two commands

The method is different. You can use these two buttons to complete the path confirmation after teaching. Therefore,

It is important to understand how these two mobile methods differ. According to the previous section, MOVJ is moving from point to point.

Page 41

MOVL moves in a straight line, so moving quickly to the point is the axis parameter for each axis to move to the teach point, while the line

Move to the point where the interpolation line moves to the position of the teach point.

The location move confirmation method is as follows:

(1) Select the point to be deleted in the teaching point list

(2) Click the [Fast Move to Point] or [Line Move to Point] button

In addition, in the teaching point system, we can teach three kinds of variables, namely 1. numerical value, 2. string, 3.

Coordinate points, and the most common ones are coordinate points. The following explains the functions of each field in the teaching list and

significance:

1. No.: The number corresponding to the memory location of each point in the software. It can teach up to 500 points.

2. Name: The name of each variable, which is used to identify variables in the arm control language.

3. Type: The variable type of teaching. In the arm control language, the string variable needs to be prefixed with $, coordinates.

Point variables need to be prefixed with *, and the coordinate system selected for teaching will also be indicated in this column.

4. Value: The content value whose variable type is a numeric value

5. String: The variable type is the content value of the string

6. J1~J8: Axis 1~Axis 8 with variable type as coordinate point Angle value, X, Y, Z, A, B, C, U, V: variable type

Coordinate value of the coordinate point

Figure 38 teach point format

Right click on the list of teaching points to see the reordering function, which can redefine the point of the list.

Arrangement, the following arrangement:

(1) sort by number

(2) Sort by number name

(3) Reverse sort by number name

(4) Sort by variable name

(5) Reverse sort by variable name

Figure 39 Sorting method

Sorting 1, 2, 3,..., a, b, c...

Reverse sorting 15, 14, 13..., z, y, x...

F. Coordinate setting

In the coordinate setting, there are mainly the following three settings.

(1) Workpiece coordinate setting: Set the workpiece coordinates we define.

(2) Tool coordinate setting: Set the length and angle of the tool extended by the method.

(3) Each axis limit solution setting: Set the inverse operation to not find an approximate solution, and the value range is limited to -180~+180 degr

The workpiece coordinate system, as the name suggests, uses the workpiece as the reference point as the coordinate system, in the card coor

The origin of the coordinates is the starting point of the system. Therefore, in some cases, the origin of the geodetic coordinates of the original card

Intuitive, therefore, we use the re-set coordinate origin to coincide with the workpiece origin, so we are on the ground

The description of the target will also be more intuitive. In the [Motion Control] project, we click [Coordinate Settings] to redefine the work.

Coordinates, as shown in Figure 40 coordinates.

Click the button to find that the system will ask you to modify which group of coordinates settings (0~9), which is provided by the system.

10 sets of coordinates for setting, so that when writing Robot Language, you can use CHANGEAXIS

Let's change the coordinate system, and the teaching point will also prompt you to use the group coordinate system to teach you to avoid users.

I found a bit of confusion when I tested the teaching point or when I wrote the program.

Figure 40 coordinates setting

After changing the coordinates setting, press the [X] in the upper right corner to complete the setting.

The workpiece coordinate setting method is as follows:

Use the position status display value of the robot arm to establish the origin of the workpiece coordinate, and design various methods to accurately

The workpiece coordinate origin is at the position of the original robot arm geodetic coordinates. Therefore, input this precise position to the param

In the material, the workpiece coordinate setting can be completed, and the same tool coordinate setting method is also the same. For example setti

The origin of the new workpiece coordinate is the position of the original coordinate X=10, Rx=40, and the end point of the tool is the extension of

X=100, Z=100, rotated 45 degrees. In addition, each coordinate setting is linked to each Robot file.

Together, a single file uses a single setting.

G. Keyboard operation mode

Press F1 in the motion control interface to switch the keyboard operation mode, which can be controlled by the keyboard.

The arm is easy for the user to teach. As shown in the figure below, the current speed and the selected control mode will be displayed.

The shortcut description is displayed on the right side.

◎ Attention! After switching to keyboard operation mode, no other operations can be performed except for jumping out of this mode.

Figure 41 Keyboard manipulation mode

The keyboard shortcut can be used to select the operation mode. The selected mode will display different colors and the left text.

The window will remind the user which mode to switch to. The following instructions are displayed:

(1) Use the up and down keys to increase or decrease the movement

(2) Q, A key changes speed (you need to switch to single-axis mode to change PtP Speed, and cut to position mode)

Change Line Speed)

(3) W, S key changes the speed ratio (Speed Rate)

(4) J key switches to single axis mode

(5) P key switches to position mode

(6) T key switches to tool mode

(7) 1, 2, 3... key switching mode, for example

In single axis mode, 1 = first axis, 2 = second axis...

In position mode, 1=X axis, 2=Y axis...

And so on

(8) H key moves to the origin, you need to keep pressing the button to move, release and stop immediately

(9) K key moves quickly to the point, you need to continue to press the button to move, release immediately stop

(10) L key moves straight to the point, you need to continue to press the button to move, release and stop immediately

(11) Page Up/ Page Down, select points (up/down)

(12) Shift + two buttons simultaneously, you can add or overwrite teaching points

H. I/O (input point/output point) control

In addition to the motion function, the robot needs to be matched with the surrounding devices to complete the work.

Such as jaws, solenoid valves, fixture locks, visual identification, etc., so the external device communication on the software provides

Two interfaces are [I/O Control] and [Ethernet Network Communication], and I/O Control is

It is the most commonly used interface in sequence control. We can use I/O to open and close the jaws, or use I/O to work with the PLC.

The controller communicates, and the relevant communication protocol is designed according to the user's needs, which is quite convenient.

First open the [I/O Settings] interface (take the A-type robot arm as an example), you can see the following **error!** Can **not find the parameters**

According to the source. At the input point, the robot arm provides I48~I55 with 8 signal inputs, and o48~o55 total 8

Point air valve signal output, and provide 8 points of unwired signal input from I56~I63 (need to be wired by the user)

And o56~o63 a total of 8 unwired signal output (need to be wired by the user). In addition to providing contacts,

You can see that there are many I/O points for system signal processing I/O points.

Figure 42 Input/Output Point Control (Left: Input Point, Right: Output Point)

In the input point window, we can observe the state of the input point. In the simulation mode, you can directly click the analog input.

In, and in the output point window, we can directly click the button to perform I/O signal output.

I. System absolute value encoder zero correction

In the I/O setting interface, the absolute value encoder is used to clear the zeroing operation with a specific output point, as shown below.

When the robot arm is operated for a long time, it causes mechanical wear, accuracy is inaccurate, or a collision occurs.

If the mechanism is slightly deformed, the zero point calibration procedure must be carried out. The mechanism calibration method needs to be car

Step out the software and execute as follows:

◎ Attention! This action will seriously affect the behavior of the robot arm, and the effect of the action is not fully understood.

Do not follow the steps below.

1. The A-arm disassembles the second, fourth and fifth-axis housings of the robot arm and corrects them with a calibration rod.

D-arm uses a level to correct uniaxial levels

2. Release the brakes or use the motion control buttons to move the arm to the calibration zero and fine tune the correction.

3. After the A-arm inserts the calibration rod correctly into the calibration hole, the correction positioning is completed.

4. The system needs to first Servo OFF, click the output point o36 [Soft OFF] or press the emergency switch.

5. Confirm that the robot arm position is calibrated at the calibration origin.

6. Then select the output point o34 [absolute encoder clear], at which point the absolute encoder has been cleared.

7. The system restarts, or click the [Real Machine Restart] button in the [System/Exception Handling] item, and the encoder will read after restartir

The values should all be zero.

8. Click [Move to Home] to complete the correction if you move to Home correctly.

Figure 43 Absolute Encoder Clear Point & Soft OFF Point

J. Operation screen and simulation screen

In the operation screen, we will display the system status window as well as the simulation screen and program execution commands.

Column, the program execution command column will actually reverse the location and content of the actual program execution, and the simulatio

Simultaneously operate a 3D model with a real robotic arm for operator reference and in simulation mode

It can be used as a basis for manipulating the robot arm. By simulating the screen, the user can write the program offline in advance.

Use the simulation mode to pre-determine whether the motion meets the requirements and whether there is interference problem, and the execution

The method can refer to the appendix. The following figure lists the content sub-items of the operation screen, including a status display window o

And the upper right robot arm simulation screen, and the teaching point screen on the lower right.

Figure 44 Operation screen

The operation of the analog screen can refer to the following:

(1) Left mouse button pressed for horizontal movement

(2) The middle mouse button is pressed to zoom

(3) Right click on the mouse to rotate

For the analog screen operation button as shown below, other operations can be performed.

Figure 45 Analog Screen Operation Button

among them,

(1) Command display: switch to display the teaching point list or execute time command below

(2) Viewing angle of the screen: switching the perspective of the analog screen

(3) Path display: whether the switch shows a red moving path when the robot arm moves

(4) Teaching point display: Whether the switch displays all teaching points (yellow arrow) on the simulation screen, and when clicking this point,

This point becomes a red arrow in the analog screen (the direction indicated by the arrow is the Z direction)

Figure 46 teaches the graphics displayed in the simulation screen

(5) Model loading: 3D models can be externally loaded, and the file type is STL.

Through the model loading, you can more intuitively observe the action relationship of the robot arm in the virtual environment.

How to build an STL model with SolidWorks.

(1) First open the SolidWorks part drawing or combination drawing file to be converted, as shown in the following figure.

(2) Add a new standard in the place where you want to set the setting as the origin. The following coordinate system 1

Figure 47 SolidWorks Sample Image

(3) Click Save as new file and change the archive type to STL.

Page 50

(4) Select "Output as" in the output option → Binary

(5) Select the "resolution" in the output option → good

(6) Select the coordinate system 1 just created in the output coordinate system in the output option.

(7) If it is a combination key, please check "Store all components of the assembly in a single file"

Figure 48STL Output Options

(8) Finally click OK and save to complete the STL file

The generated STL model is now imported into the arm simulation screen. Click on the model to load and you can see

A new window pops up, and right click on the window, the shortcut option will pop up, click [leave]

Leave the model loading interface as shown below.

◎ Attention! After opening the model loading interface, no other operations can be performed except leaving this interface.

Location setting Color setting

Figure 49 Model loading interface

Click [Load] and select the model you just created to load the model. If you want to delete the model, just

Right click on the model and select Delete.

The model loads the input field below to change the existing position and color of the model, and the preset model is loaded into the machin

The peripheral module of the arm (ie relative to the earth coordinates), if the model is a jaw, please right click on the model

The model can be attached to the end effector relative to the tool coordinates.

The figure below shows the simulation screen after loading the sample model.

Figure 50 Simulation screen after loading the STL model

K. Visual Module Introduction

This operating software provides a simple visual module that can be used as a basic object tracking function with a camera. The main processing part is run with the robotic programming language, and the human interface is mainly used for display and To capture the image, we can directly capture the image of the object and save it by clicking the left mouse button on the image. And can do basic image processing.

Switching to the visual module screen, you can find that the central picture is black, that is, the interface of the image display. Right click on it and select [Open Camera], you can see the picture captured by the camera, as shown below.

Figure 51 Vision Module Camera Capture Screen

In the figure, we use the central circular hole as the object to be tracked. Click [Close Camera] to stop the image and slide it. After pressing the left button to select the center hole, release it, and the storage window will pop up, and the image area will be selected. The file type is bmp, as shown below.

Page 53

Figure 52 Framed visual objects

Name the file test.bmp and store it in a specific location. The example is stored in D:\qt_project\ITRI

RAS\Win32\Release\test.bmp.

Next, you will use Robot Language for visual tracking, which will be explained in subsequent chapters.

In the "Compose" section, follow the steps first.

(1) Open a new project (click on the top left corner [Build a new project]), select the file type.irb

(2) Switch to [Programming Interface] and write the following code

```
VS_PATTEACH 0 D:\qt_project\ITRI RAS\Win32\Release\test.bmp
VS_OPEN
WHILE 1
VS_QUERYFRAME
VS_PATMATCH 0 1
VS_SHOW
SLEEP 100
LOOP
VS_CLOSE
```

(3) Detailed code code screen as shown below

Page 54

Figure 53 Visual Object Detection Code

(4) After the program is executed, you can see the tracking result under the [Vision Module] interface.

Figure 54 Visual Object Tracking Results

L. System parameters and anomaly history

When we click on the [System] project, we can enter the modification interface of the system parameters. Here we first

The operating mode is defined as follows:

User mode

2. Engineer mode

The robot state also has two modes:

Simulation mode

2. Real machine connection mode

In the simulation mode, the system only simulates the movement of the robot arm, the actual arm does not move, and the I/O does not advance.

Actual action, but the actual machine mode is connected with the actual robot arm. Click on the right under this interface.

The upper corner button switches the mode directly.

Figure 55 Mode Switch Button

In addition, the operating software here is divided into two modes for the user, and the two modes have

The permissions are different. The user mode pointer does not work for the human-machine interface operated by the general user.

Compilation and modification of parameters, while engineer mode is no. In the design phase, engineers use engineer modules

Exercise and teach points, and write robotic programming language, with full operating rights,

After the test path is running correctly, the robot arm will be handed over to the field staff for operation.

If the operator misuses, we will switch the operating software to user mode, in which the user can only watch the operation.

Turn the screen, perform input point output point control, and watch system parameters and exceptions.

Modification and movement, only the operation, pause, stop and other buttons and modify user data, mode jump

The mode jumps by inputting an instruction.

Page 56

Press the F11 button of the keyboard during any screen, and the inquiry window will pop up as shown below.

Figure 56 Ask to change the appearance window

Enter simple to jump to user mode

Enter itriq300 to jump to engineer mode

After changing the appearance, it will remain the last set mode when it is next turned on.

Figure 57 User Mode

In addition, the system parameters are divided into the following four types of parameters.

(1) Robot system basic parameters

(2) Robot kinematics DH parameters

(3) MCC axial control card parameters

(4) I/O table parameters

Figure 58 System Parameters

To change the content, just select the parameter and double-click the left button in the right column to modify it. Please be sure to click after modif

The archive restart can only be modified.

◎ Attention! Modifying the parameters will change the operating state and mode of the robot arm. Please understand the parameters clea

Meaning avoids unexpected errors.

The contents and definitions of each parameter are as follows:

Robot system basic parameters	definition
Pixel Mode	Set the software resolution, set value -1 for automatic detection
Auto Communication Mode	Whether to automatically turn on communication when the operation software is initialized and executed
Auto Communication IP	Set the IP address of the automatic communication to be turned on
Auto Communication Port	Set the port (connection port) where automatic communication is turned on.
Auto Communication Com	Set the RS232 connector for automatic communication to open
Auto Communication Type	Set the way to communicate automatically using Ethernet or RS232
Auto Communication Buffer Size	Set the communication data register size
Auto Load Project Name	Depending on the file that is automatically opened at startup

Page 58

Robot kinematics DH parameter definition	
Kinematics	Types of kinematics: (0) Type A, (1) Type D
Mechanical a1～a8	D-H parameter content
Mechanical d1～d8	D-H parameter content
Mechanical dPe～dRf	Parallel mechanism parameter setting
MCC axial control card parameters	definition
System Mode	System operation mode: (0) real machine connection, (1) simulation mode
Number Of Joints	Number of mechanical arm axes
Absolute Encoder Mode	Absolute encoder type
Card Index	Axis control card number
Card Type	Axis control card form: (3) 6 axes, (4) 8 axes
Joint 1~8 Initial Degree	Home point angle of each axis
RIO Set	RIO interface number
RIO Input Swap	Is the RIO input point reversed?
RIO Output Swap	Is the RIO output point reversed?
Driver 1~8 Alarm IO ID	Drive abnormal IO contact number
Driver 1~8 Alarm IO Swap	Drive abnormal IO contact is the opposite
E-STOP Alarm IO ID	Emergency stop IO contact number

Page 59

E-STOP Alarm IO Swap	Emergency stop IO contact is the opposite
Motor 1~8 PosToEncoderDir	Motor running angle and encoder direction: (0) in phase, (1) inverting
Motor 1~8 PPR	Motor running a pulse wave number

Motor 1~8 RPM	Maximum speed of motor operation (unit: rev / min)
Motor 1~8 Pitch	Motor running unit=1
Motor 1~8 GearRatio	Reduction ratio of each shaft
Motor 1~8 HighLimit	Upper limit of each axis (unit: circle)
Motor 1~8 LowLimit	Lower limit of each axis (unit: circle)
Motor 1~8 PulseMode	Pulse mode
Motor 1~8 ABSwap	Whether each axis encoder is AB inverted
Motor 1~8 InputRate	Input ratio of each axis encoder
Motor 1~8 Absolute Encoder Type	Absolute encoder type for each axis
Motor 1~8 Absolute Encoder Para 1	Absolute encoder parameter 1 for each axis
Motor 1~8 Absolute Encoder Para 2	Absolute encoder parameters for each axis 2
Motor 1~8 Absolute Encoder Para 3	Absolute encoder parameters for each axis 3
Motor 1~8 Absolute Offset Degree	Absolute encoder initial offset for each axis (actual display value = encoder read value – offset value)
Motor 1~8 Absolute Swap Degree	Whether the absolute encoder reading value of each axis is reversed
Motor 1~8 Encoder Swap Degree	Whether the axis encoder reads the value in reverse

Page 60

Motor 1~8 Pulse Width	Pulse width of each axis
I/O Version	System I/O list version
J6/J5 Ratio	A type arm J5/J6 bevel gear ratio
Run Input	Execute the trigger input point, -1 means not enabled
Stop Input	Stop triggering the input point, -1 means not enabled
Pause Input	Pause trigger input point, -1 means not enabled
E-Stop Output	Linked to the emergency stop output point, -1 means not enabled
Singular Avoid Method	Singular position avoidance method, -1 means not enabled

I/O table parameters	definition
Input 0~63	Input point name
Output 0~63	Output point name

In addition, in the [Exception] item, the time point at which the abnormality occurred and the abnormal situation are recorded, and the upper right t

[History Clear] or [Abnormal Reset] can be performed, and [Abnormal Reset] can only restore non-driver exceptions.

Normally, if the exception is a drive exception, the system restart mode can only be used to clear the exception.

The exception categories are listed below:

Exception name	definition
1st to 8th axes reach the limit	Axis 1~8 reach the limit position
Error error	Interpolation error

Axis control card error XXXXX	The axis control card is abnormal. For details, see the axis control card exception list.
Drive exception 1~8	Axis 1~8 drive Alarm

M. Communication module and communication operation

In robotic arm control, it is often necessary to communicate with external devices, vision systems, master control systems, and

He has a robotic arm, etc., and the system can be commanded via an Ethernet network or RS232 serial transmission.

Communication, communication mode is controlled remotely by transmitting the robotic programming language, such as Figure 59

The communication method can produce a production line system with multiple machines working together:

Figure 59 Robot Arm Control System

The above method uses Remote Control to control the robot arm by sending commands or data.

Operation is very common on production line systems.

Figure 60 Production Line Scheduling Control

The above picture shows a production line scheduling system in which the central controller processes the data or dispatches commands between tl

As can be seen from the above figure, we can have 6 robots for Robot1~Robot6 by central control computer.

The arm gives the order, and according to the type of command issued, it is divided into command control and data control, as the name suggests, l

The control is controlled directly by the robotic language, and no Robot Language is executed on the robot arm.

On the contrary, the data control only passes the data to the robot arm, and the arm itself is already performing the Robot Language task.

Through the data transfer to get the adjustment of the relevant actions, so according to the two control methods and transmission format is divided

under:

Communication method:

(1) Ethernet Server
(2) Ethernet Client
(3) RS232

Control Method:

(1) Command communication mode

(2) Data communication mode

Figure 61 Types and methods of communication

The command communication mode is controlled by passing the robot language (Robot Language).

Is a standard string, meaning that there is an empty character (NULL or \0) at the end of the string, taking the GOHOME command as an example.

The central control computer will send 7 characters:

(1) G, (2)O, (3)H, (4)O, (5)M, (6)E, (7)\0

The robot arm operation software judges a single single command by discriminating the ending character and transmits the command

Page 63

Will be interpreted by the interpreter, after the interpretation is completed, and then stored in the interpretation schedule waiting to be executed, so

Will wait until the last command is completed, and the movement is the same, as follows

(1) First movement command\0

(2) Second motion command\0

Figure 62 Two strokes have pauses

The overall motion situation will be as shown in Figure 62, with pauses between the two movements.

To achieve a non-stop motion, it is like writing a robotic language in the [programming] project.

The two commands are separated by a line break, as in the following way

(1) The first motion command\n

(2) Second motion command\0

Writing a C programming language is:

Char string = "first motion command \n second motion command \0";

Figure 63 There is no pause between two motion commands

The movement will be smoother and without pauses, as shown in Figure 63.

Page 64

When the control computer sends a command, the robot arm operation software will analyze whether the command is correct or not.

If the process is the correct robotic language, the IRA will be returned. If it is the wrong syntax, the ERR will be returned.

The computer determines whether the command sent an error by returning the value.

The data communication mode is the same as the command communication mode. The difference is that the data is transmitted.

Not a command, and the main program can execute and receive data at the same time, mainly through the network / serial

The material is transferred to the scratchpad in the controller, and the next data transfer will continue to be stacked until the top

Layer data is taken away, data will be sent to the top, and the data format passed is numerical data and can be transmitted more.

Parameters, parameters and parameters are separated by a comma "," as follows

10.2, 20.3, -100

That is, a piece of data is transmitted with 3 parameters of 10.2 and 20.3 and -100 respectively.

The stacking form of the register is as follows

1st data	Value 1	Value 2	...	Value m
2nd data	Value 1	Value 2	...	Value m
...				
Nth data	Value 1	Value 2	...	Value m

Figure 64 Controller data stream stacking

To get the data, you only need to use STACK_GET in the robot code to get the top level.

It can be.

Use a web server test software here (additional need to contact the engineer), we can benefit

Use this software to connect to the machine and test the network communication function, and the IP address of the local connection is

127.0.0.1, the test software fixed port is 4000.

Figure 65 Network Communication Native Software

When the software is executed, the Server connection will be automatically opened, and the client will be connected. The Send at the top can be us

Enter the command to be sent (Enter button to send), the middle Recv displays the last message received, the list below shows

Show history received messages and current software status.

The following figure shows the communication between the robot arm and the connection test software:

Figure 66 Command Communication Control

Similarly, the robotic arm can also perform data communication control. Add the following code and take the data.

Communication connection, as shown in Figure 67.

STACK_CLEAR first clears the scratchpad to avoid remaining data in the scratchpad, used within the program

The loop continuously obtains data through STACK_GET, and the interpreter will stay when the scratchpad has no data.

In STACK_GET, the line waits for the data to come in. Once the data is available, the MSG outputs the data content.

The result is shown in Figure 68.

Figure 67 Data Communication Code

Figure 68 Data Communication Execution Result

Similarly, when passing data, you need to add 『\0』 characters as the end of each data (same as the command stream format).

In addition, using RS232 serial transmission is the same result, we can use command flow control, or

Data is transmitted through the data stream, and the biggest difference between the two modes is that the command communication control can onl

Control, you can't execute the robot language with this machine at the same time; the data communication control is the opposite, data communicat

The main information is dropped into the scratchpad for the machine to execute the robot language. Therefore, data communication is only require

To establish a connection, the data can be transferred to the scratchpad even if the native robot program is in a non-executable state.

N. Robotic programming language writing interface

The key to manipulating the robotic arm to move it flexibly is the arm control language. This software design

Simple and easy to understand programming language, so that even users without procedural design can easily write out

The entire robotic arm controls the language, and the operating interface is introduced here first.

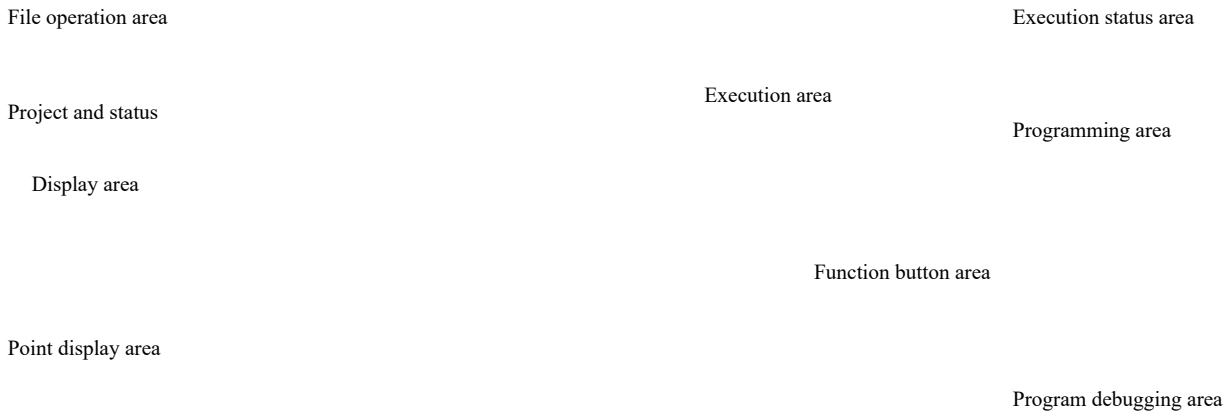

Figure 69 programming interface

Here to explain the robotic language type, divided into

(1) Single file (.irb)

(2) Project (.pir)

The Robot Language execution core is the .irb file, and the project contains two .irb files.

When a file is executed, the program only executes the instructions in the file. When executing as a project, in addition to the program writing inter

Will be divided into two interfaces, the main program and the subprogram. When executing, the two irb files are executed synchronously, so

Please confirm the task needs first, avoid writing the motion command in the main program and the subprogram to cause the robot arm

Action.

The purpose of the project is that the main program can be used for motion control, while the subprogram is used for peripheral I/O and com

The control and monitoring can be separated and synchronized when the user writes the Robot Language. Two more irb

The files share the same list of teaching points, so the data can be transmitted through the list of teaching points, as long as the teaching point

Page 68

The [numerical] variable is declared in the list to share the value.

In addition, when the [Program Writing] item is pressed, the file operation that was originally darkened in the upper left corner of the softwa

The button can be pressed. Note that only the file processing button in the program writing section will be enabled, so that the user can

Three functions of building a new project, opening a project, and saving a project. The preset is automatically read after the robot software is execu

Project, the project file name is default.pir, the software arm of the software control language exclusive file name is irb (ITRI

Robot), the file structure contains the following information.

(1) Robotic language content

(2) Teaching points of sports teaching

(3) Coordinate system setting

So each irb has its own running content, teaching points, and coordinate system. The project file has

Two irbs, two of which are different from the Robot Language, teaching points and coordinate systems

All the same.

The figure below explains the execution of the operating area.

Figure 70 Performing the action area button

Its features include the following:

1. Execution: After the program is written, click Execute to start executing the programming language, but before the execution, the software soluti

The translator will automatically archive and run the debugger. If the debugger has a syntax error, it will detect the error.

If it is displayed on the debug list, the robot arm will not actually be run to ensure safety. And the debug list will

Page 69

The error line number is displayed and corrected by the program writer.

2. Pause: During the execution of the program, the program execution command line of the main screen will immediately display the software inter

The executed command, when the pause is pressed, the robotic program interpreter will immediately pause, and the arm movement will also stand.

Pause pause, the system status list will also display the pause information, and when you press the execution, the robot arm will continue

Execute the command, so we can use this function to stop the robot arm and stop it, let the personnel

You can temporarily enter the active space of the robot arm to work or repair the surrounding equipment, and then leave the following

Continued to run the original half of the work.

3. Stop: The robot stops and the interpreter stops working.

4. Single step: We can use this function to perform single step exercise during the test writer phase.

Step by step to confirm whether the action is normal, the function will only execute one line at a time, and can be paused

Then start stepping again, or you can click to execute directly in the single-step execution state.

make. (Single stepping will also be debugged first and will only be available when opening a single irb file).

In addition, the function button part is organized, and the programming language written above is automatically typeset, and another

One is the sprite, the wizard function is enabled. We can query all the robotic programming languages and have simple

Instructions and command input format, you can use the sprite to directly insert instructions in the place where you want to insert the instruction.

The user does not need to memorize the language grammar when writing the arm programming language, and it is more convenient to use.

If you don't want to use the execution to test the program function when you open the project, because the main and subprograms will be executed

Then press [Single Execution] of the function operation button.

Figure 71 Project and automatic alignment

Figure 72 Program Language Insert Wizard

Click the left mouse button at the point where you want to insert the command, click on the sprite, and then fill in the field contents, then

Click Insert and the sprite will automatically insert a complete command for us.

In addition, the maximum number of lines in a robotic language single file is 1000 lines, and the maximum number of variables is 500.

The number of macros is 100. A detailed explanation of the control language will be given below.

O. Robot arm control language

Robot Arm Control Language

The control languages are mainly divided into the following:

There are many functions and instructions in This Language. Since we write C/C++ program Outside the software, we only focus on motion Commands

(1) Variable declaration and four arithmetic operations

(Please jump to page 89)

DIM

DIM	Variable name	As	Variable type
			1. Number
			2. String
			3. Point

The DIM instruction can declare three types: number, string, and point.

The arm programming language adopts a hierarchical structure, and the variables in each level are independent, that is, the variables of life and dea

Like the programming language, as explained in the following figure, WHILE and IF instructions will separate the levels, that is, the variables are

The scope is only in its own WHILE area or IF area and its sub-layers. From the figure below we can see that the change

The effective range of the number.

1 Main level

(declaration variables A, B, C)

2 1st IF level

3 2nd IF level (declaration variable G)

4 3rd IF level (declaration variable I)

5 The first WHILE level (declaration variable J)

6 2nd WHILE level (declaration variable D, E)

7 4th IF level

8 5th IF level (declaration variable K)

9 The third WHILE level (declaration variable F)

10 4th WHILE level

11 5th WHILE level (declaration variable H)

1. The variables A, B, and C are recognized at any level.

2. The variable D, E is only known at the level 6, 9.

3. The variable F is only known at level 9.

4. The variable G is only known at the level 3, 7, 8, 10, 11.

5. The variable H is only known at level 11.

6. Variable I only knows level 4.

7. Variable J is only known at level 5.

8. The variable K is only known at level 8.

The declared numerical variables can be directly mathematically operated through four arithmetic expressions, such as the following program:

```
DIM X As Number
X = 2*( 3+5 ) – 7
```

That is, X is calculated as X = 9. Also can be written as follows:

```
DIM X As Number
DIM Y As Number
X = 2*( 3+5 ) – 7
Y = 2 + X * 2
```

Then Y is calculated as Y=20. Note that the four operations are added and subtracted before being multiplied and divided.

In addition, it cannot be written in the following form:

```
DIM X As Number
X = 10 = 2*3 ← has two equals
X += 1                ← no such way
```

Variable declarations can be declared in Chinese, for example:

```
DIM, I am a variable, As Number.
I am a variable = 10
```

The above is acceptable.

SETSTRING

SETSTRING	Set variable	String content

Strings can be directly assigned with equal signs as values, but in the SETSTRING mode, the following examples:

SETSTRING $STR string

The above is to set the STR = string, be careful to add the $ preamble before the string variable.

SETPOINT_DEG

SETPOINT_DEG	Set variable					point			
SETPOINT_DEG	Set variable	J1	J2	J3	J4	J5	J6	J7	J8

SETPOINT_POS

SETPOINT_POS	Set variable					point			
SETPOINT_POS	Set variable	X	Y	Z	A	B	C	U	V

SETPOINT_DEG sets the coordinate point (defined from the angle), which means that the angle data is transmitted through the forward kinematics

Calculate the location data, because the coordinate point data contains: 1. angle, 2. position, two types of data, so in

When setting the coordinate point variable, we must consider whether it is the angle conversion position method or the position conversion angle m

And can directly give point variables or give angle information for each axis, for example:

SETPOINT_DEG *NEW_POINT *OLD_POINT

The above formula sets the OLD_POINT coordinate point to NEW_POINT, and the setting calculation method is changed from angle

Count as a position, and note that the coordinate point needs to be preceded by a *.

SETPOINT_DEG *P1 45 20 20 30 30 30 ← A type 6 axis

SERPOINT_DEG *P1 45 20 45 180 ←D type 4 axis

The coordinate point setting can also be performed in the above manner.

SETPOINT_POS sets the coordinate point (defined from position), this setting happens to be just with the SETPOINT_DEG command

Instead, the point data is set from the positional back angle by inverse kinematics, such as the following:

SETPOINT_POS *NEW_POINT *OLD_POINT

SETPOINT_POS *NEW_POINT 100 200 300 0 90 180

◎ Attention! When using the SETPOINT command, you should not teach points or singularities that the robotic arm cannot reach.

Position to avoid danger.

(2) Mathematical operation function

SQRT

SQRT	Numerical value	Set variable

Calculate the square root value, such as

SQRT 4 X

The calculation result is X = 2, and the above formula can be regarded as √4 = X, in addition

A = 100

SQRT AB

The result is B = 10.

POW

POW	Base	index	Set variable

POW2

POW	index	Set variable

POW3

POW	index	Set variable

POW5

POW	index	Set variable

POW10

POW	index	Set variable

Calculate the number of powers, for example,

POW 7 2 X

The calculation result is X = 14, the above formula can be regarded as X = 7^2, and

POW2 10 X

Calculate the number of powers with base 2, and the result is X = 1024. The above equation can be regarded as X = 2^10.

The POW2, POW3, POW5, and POW10 instructions have the same usage.

The mathematical formulas listed above can also be brought into the calculation of variables, such as

A = 2

POW10 AB

The calculation result is B = 100, and the above formula can be regarded as B = 10^A.

LOG

LOG	Numerical value	Set variable

LOG10

LOG10	Numerical value	Set variable

Calculate the log value, which is the natural logarithm (exponential) base LOG, or a 10-bit base

LOG10, examples are as follows

LOG 100 X

LOG10 100 Y

SIN

SIN	angle	Set variable

Calculate the sin trigonometric function, the unit of the incoming angle is degrees, such as

SIN 30 X

The result is X = 0.5. The above equation can also be regarded as X = sin (30 degrees). In addition, variables can be passed in.

A = 60

SIN AB

The above formula means B = sin(A).

COS

COS	angle	Set variable

Calculate the cos trigonometric function, the unit of the incoming angle is degrees, such as

COS 60 X

The result is X = 0.5. The above equation can also be regarded as X = cos (60 degrees). In addition, variables can be passed in.

A = 60

COS AB

The above formula means B = cos(A).

TAN

TAN	angle	Set variable

Calculate the tan trigonometric function, the unit of the incoming angle is degrees, such as

TAN 30 X

The result is X = 0.557735. The above equation can also be regarded as X = tan (30 degrees). In addition, variables can be passed in.

A = 60

TAN AB

The above formula means B = tan(A).

ASIN

ASIN	Numerical value	Set variable

Calculate the arc sin inverse trigonometric function, calculate the unit as degrees, and note that the range of the input range is +1~-1.

Exceeding will produce a calculation error, and the range of the solution is +90~-90, such as

ASIN 0.5 X

The result is X = 30. The above equation can also be regarded as X = asin(0.5). In addition, variables can be passed in.

A = 0.75

ASIN AB

The above formula means B = asin(A).

ACOS

ACOS Numerical value Set variable

Calculate the arc cos inverse trigonometric function, calculate the unit as degree, and note that the input range is +1~-1

Between, more than will produce a calculation error, and the range of the solution is +0~+180, such as

ACOS 0.5 X

The result is X = 60. The above equation can also be regarded as X = acos(0.5). In addition, variables can be passed in.

A = 0.75

ACOS AB

The above formula means B = acos(A).

ATAN

ATAN Numerical value Set variable

Calculate the arc tan inverse trigonometric function, calculate the unit as degree, and note that the range of the input range is +∞~-∞

Between, and the range of the solution is +90~-90, such as

ATAN 0.5 X

The calculation result is X = 26.565, and the above formula can also be regarded as X = atan (0.5). In addition, variables can be passed in.

A = 0.75

ATAN AB

The above formula means B = atan(A).

ATAN2

ATAN2 Bottom X High Y Set variable

Calculate the arc tan inverse trigonometric function, calculate the unit as degree, and note that the range of the input range is +∞~-∞

Between, and the range of the solution is +180~-180, such as

ATAN2 1 1 X

The result is X = 45. The above equation can also be regarded as X = atan2(1,1). In addition, variables can be passed in.

A = 0.75

B = 1.25

ATAN ABC

The above formula means C = atan(A, B).

DEG2RAD

DEG2RAD	Numerical value	Set variable

RAD2DEG

RAD2DEG	Numerical value	Set variable

The conversion of the diameter and the angle, the DEG2RAD is the angle to the diameter, and the RAD2DEG is the angle to the angle.

For example, we have to convert 30 degrees into a diameter for X to write

DEG2RAD 30 X

Conversely, the angle of the angle can be written as

RAD2DEG 3.14 X

CEIL

CEIL	Numerical value	Set variable

Perform ceil math operations, such as X=ceil (30.2) can be written as

CEIL 30.2 X

FLOOR

FLOOR	Numerical value	Set variable

Perform floor math operations, such as X=floor(2.23) can be written as

FLOOR 2.23 X

RAND

RAND	Random range R	Set variable

Perform a random number operation with a random range of 0~R. For example, we want to get a random integer from 0 to 100.

Written

RAND 100 X

X can get random integers, noting that the values obtained each time are different.

ABS

ABS	Numerical value	Set variable

Perform absolute value operations on variables, such as B = abs(A) can be written as

ABS AB

EXP

EXP	Numerical value	Set variable

Perform a natural logarithm (exponential) operation on the variable, such as B = exp(A) can be written as

EXP AB

(3) Basic processing functions

MSG

MSG	Message content

A message can be displayed in the system status display list window, for example

MSG ITRI Robot

The result message window will display

If you enter Chinese

MSG ITRI robot

The same result

And the command can display the variable value, and the variable value can be found by using the address symbol &, the following example show

As a result, the coordinate point P results, as well as the string S content:

```
DIM X As Number

DIM S As String
```



```
DIM P As Point

X= 10*2+3*5

SETSTRING ITRI $S

SETPOINT_DEG *P 0 0 45 -45 0 10

MSG Number= &X

MSG String= &S

MSG Point= &P
```

The result of the execution is

Point Because the data is long, we can pull the data displayed in the back.

The MSG command can also display multiple data at the same time, such as the display of the above program.

```
MSG &S = &X
```

The output is

PAUSE

PAUSE

Pause, motion, and interpreter commands are paused. Equivalent to the pause button, you need to manually press the execute button to continue.

STOP

STOP

Stop, motion and interpreter commands stop. Equivalent to the stop button, stop the program immediately.

SLEEP

SLEEP Pause time (unit: millisecond ms)

Pause the interpreter to pause at a set time, noting that the motion command will not pause.

SLEEP 1000

The above example is a pause of 1 second (that is, 1000 milliseconds).

CLC

CLC

Clear the contents of the system status display list window.

FILE_READNUMBERS

FILE_READNUMBERS File name Value...

Read the data file from the outside, the file must be placed in the same folder as the robotic language file.

To read multiple values, but in order, for example

FILE_READNUMBERS Data.txt N1 N2 N3 N4 N5

The above formula reads 5 values from Data.txt and stores them in the N1~N5 variables.

And the data structure of Data.txt needs to be changed separately for each digital data.

Figure 73 Data format for reading data files

As can be seen from the above, each piece of data is separated, and the result of reading is

N1 = 1.23 N2 = 4.25 N3 = 3.32 N4 = 0.11 N5 = 0.56

FILE_WRITENUMBERS

FILE_WRITENUMBERS File name Value...

In contrast to FILE_READNUMBERS, the data is written, and the written file is stored in the machine.

The arm program language file can store multiple data under the same folder, but it will be overwritten each time it is executed.

The original data file, examples are as follows:

N1 = 0.11

N2 = 0.23

N3 = 4.96

SETSTRING Data.txt $FileString

FILE_WRITENUMBERS $FileString N1 N2 N3

The above example uses the string FileString to create a file name and write N1, N2, N3 three variables

In the file.

GETTIME

GETTIME day second

Get the system time, the resulting data is stored in two variables, the first variable gets the date, now

The day is 9/20, and the date data obtained is 20, and the second variable gets the number of seconds after the time is added.

For example, the read time is 13:34:29,115, and the data obtained will be 13*60*60.

+ 34*60 + 29 + 115*0.001 = 48869.115 seconds. Examples are as follows

GETTIME nDay nSeconds

EXE

EXE Program path name parameter…

Executing an external program is equivalent to the winexec function of the c instruction. The example is as follows

EXE C:\ExProg 10 20

%

% annotation

All text after the % symbol is annotated.

(3) Basic register

INTREG_SET

INTREG_SET	Scratchpad number	Set value

Internal register setting, the value data can be temporarily stored in the internal register, the register size = 100, examples as follows

INTREG_SET 1 2.02

Set the internal scratchpad 1 value to 202.

INTREG_GET

INTREG_GET	Scratchpad number	Get the value

The internal register reads, the value can be read from the internal register, the register size = 100, examples are as follows

INTREG_GET 1 X

Read the internal scratchpad 1 value to X.

EXTREG_SET

EXTREG_SET	Scratchpad label	Set value

External register settings, the value data can be temporarily stored in the external register, examples are as follows

EXTREG_SET Num1 2.02

Set the external register Num1 value to 202.

EXTREG_GET

EXTREG_GET	Scratchpad label	Get the value

External register read, the value can be read from the external register, examples are as follows

EXTREG_GET Num1 X

Read the external register Num1 value to X.

(4) Discriminant and loop

IF

IF Discriminant

When the discriminant is established, if the discriminant is satisfied, the discriminant is executed below, and a new discriminant is added.

One level, the maximum number of levels of the interpreter is 25*25. If the interpreter exceeds the interpreter, an error will be reported, and each Il

Need to cooperate with the ENDIF directive, examples are as follows

IF X > 2+3*5

X = X+1

ENDIF

If X is larger than 2+3*5, increase X by 1.

ENDIF

ENDIF

The end of the discriminant.

WHILE

WHILE Discriminant

Looping is performed. If the discriminant is established, it will enter the loop and execute below. Adding a loop will add a layer.

Level, the maximum number of levels of the interpreter is 25*25. If the interpreter is exceeded, an error will be reported, and each WHILE instruct

With the LOOP instruction, when the instruction runs to LOOP after executing the contents of WHILE, the program will

Go back to the discriminant of WHILE and perform the judgment again, until the judgment is not established, and then jump out of execution.

WHILE X<10

X= X+1

LOOP

When X is smaller than 10, add X to 1 and discriminate until X is greater than 10 before jumping out of the loop.

LOOP

LOOP

Page 89

End of the loop.

(5) Exercise command Motion commands

SETPTPSPEED

SETPTPSPEED Point-to-point speed (unit: %)

Set the point-to-point movement speed, which is calculated as a percentage of the maximum speed of each axis.

Set point-to-point speed. The speed is represented by the
Percentage of max speed of each axis. (0~100%)

SETLINESPEED

SETLINESPEED Linear speed (unit: mm / sec mm / s)

Set the linear movement speed at a speed of mm/s.

Set line speed, speed when moving in Cartesianspace, in mm/s.

GETPTPSPEED

GETPTPSPEED Point-to-point speed (unit: %)

The point-to-point moving speed is obtained, and the speed is calculated as a percentage of the maximum speed of each axis.

Get point-to-point speed. The speed is represented by the
Percentage of max speed of each axis. (0~100%)

GETLINESPEED

GETLINESPEED Linear speed (unit: mm / sec mm / s)

The linear movement speed is obtained, and the speed is mm/s.

Get line speed, speed when moving in Cartesianspace, in mm/s.

SETSPEEDRATE

SETSPEEDRATE Speed ratio (unit: %)

Set the speed ratio, for example PtP Speed = 10, and the speed ratio Speed Rate = 150, so the actual point

The speed to the point is changed to 15, and the same linear speed setting is also true.

Set speed rate in percentage. The actual point-to-point
Speed and actual line speed is point-to-point speed and
Line speed multiplied by the speed rate. For example:

Point-to-point speed = 10% Actual point-to-point speed = 10% * 150% = 15%
Line speed = 10 mm/s ==> Actual line speed = 10mm/s * 150% = 15mm/s
Speed Rate = 150 (%)

Page 90

MOVJ

Point-to-point move by specify teaching point or joints' angles.

MOVJ point Teaching point
MOVJ J1 J2 J3 J4 J5 J6 J7 J8

The axis moves the motion command. Perform fast point-to-point axis movement without linear interpolation during the process. There are two kir

The first type of direct input point coordinates, the second type of input shaft angle value, the following is an example (take 6-axis as an example)

MOVJ *P1

MOVJ 10 10 20 -25 X 90

You can use variable inside the software as input.

MOVJ 10*2 X+1 Y/3 20.1 70 80

In addition, when inputting multiple axes, if the single axis is not intended to move, you can enter the # symbol to skip the input and keep the origi

If you don't want to move some of the axes, enter # in that axis.

Values, such as moving only two or three axes can be written as follows

For example, if you want to rotate only 2nd and 3rd axes:

MOVJ # 20 30 # # # # # # #

In addition, you can also enter only the first few axis values, the axis that is not input, the system will automatically fill 0, for example

MOVJ 10 20 30

This instruction is equivalent to

MOVJ 10 20 30 0 0 0

MOVP Point-to-point move by specifying teaching point or Pose in Cartesian space.

MOVP	point Teaching point							
MOVP	X	Y	Z	A	B	C	U	V

Move the motion command from point to point. Make a point-to-point position to move quickly, without linear interpolation during the process.

There are two modes of command, the first type of direct input point coordinates, the second type of input position value, examples are as follows (

Page 91

Axis as an example)

MOVP *P1

MOVP 100 200 320 -25 X 90

MOVP 10*2+100 X+12 Y*3 60 -70 180

In addition, if you do not intend to move in one direction when entering the position, you can enter the # symbol to skip the input and keep it in pla

Set value, for example, only move XYZ to write the following

Same as MOVJ here

MOVP 100 120 330 # # # # #

In addition, you can also enter only the first few position values, the position will not be entered, the system will automatically fill 0, for example

MOVP 100 200 300

This instruction is equivalent to

MOVP 100 200 300 0 0 0

MOVL

Linear move by specifying teaching point or pose in Cartesian space.

MOVL	point							
MOVL	X	Y	Z	A	B	C	U	V

The position moves the motion command in a straight line. Straight line movement, linear interpolation during the process, pay attention to when n

Avoid singular points, there are two ways of command, the first type directly input point coordinates, the second type of input position value,

**BEWARE: There may be singularity in linear moving.
The example is as follows (take the 6-axis as an example)

MOVL *P1

MOVL 100 200 320 -25 X 90

MOVL 10*2+100 X+12 Y*3 60 -70 180

In addition, if you do not intend to move in one direction when entering the position, you can enter the # symbol to skip the input and keep it in pla

Set value, for example, only move XYZ to write the following

Same as MOVJ in this part

MOVL 100 120 330 # # # # #

In addition, you can also enter only the first few position values, the position will not be entered, the system will automatically fill 0, for example

MOVL 100 200 300

This instruction is equivalent to

MOVL 100 200 300 0 0 0

MOVT Linear move in tool frame.
(In other word: relative movement to the curernt pose.)

MOVT	dX	dY	dZ	dA	dB	dC	dU	dV

The tool coordinates move the motion command in a straight line. Perform a point-to-point position to move quickly, and linear interpolation will b

It should be noted that the moving path does not pass through the singular point, but only the position change amount can be input, and the moving

The effector coordinates are moved as shown in the following example (taking the 6-axis as an example)

MOVT 10 0 0 -25 X 10

MOVT 10*2-100 X+12 Y*3 10 -10 0

In addition, it is also possible to input only the first few position change values, and the position change amount that is not input, the system will au

E.g

MOVT 10 20 30

This instruction is equivalent to

MOVT 10 20 30 0 0 0

MOVA Curve movement. Specify the cartesian position of point 1 (P1) And target pose (P2), the robot will move to point 2(P2) along a curve passing P1.

MOVA	Point 1			Point 2			
MOVA	P1x	P1y	P1z	P2x	P2y	P2z	P2a P2b P2c

Perform circular motion, and interpolate during the movement. The movement method is to draw an arc from the current point through the middle.

Point 1, and finally to the target point 2, as shown in the following figure, Remarks: Intermediate point 1 does not need to set the rotation posture

Figure 74 Circular motion command MOVA movement mode (blue arrow is the direction of the robot arm, black arrow is the movement path)

There are two input methods for this command. The first type directly inputs two points, and the second input point 1 has the XYZ coordinate valu

And the position coordinates of point 2, the following examples

MOVA *P1 *P2

MOVA 100 200 200 XYZABC

Note that the intermediate point 1 position cannot overlap with the current position point.

**Beware: P1 cannot be as same as the starting point.

MOVC Circle movement. Specify the cartesian position of point 1 (P1) And target pose (P2), the robot will move to point 2(P2) along The arc passing P1.

MOVC	Point 1			Point 2		
MOVC	P1x	P1y	P1z	P2x	P2y	P2z

Perform a circular motion, interpolate during the movement, and move the way from the current point to the arc through the middle

Point 1, then pass the middle point 2, and finally return to the starting point, as shown in the following figure, remarks: intermediate point 1 and po

No need to set the rotation posture

Page 94

Figure 75 Circular motion command MOVC movement mode (blue arrow is the direction of the robot arm, black arrow is the movement path)

There are two input methods for this command. The first type directly inputs two points, and the second input point 1 has the XYZ coordinate valu

And the position coordinates of point 2, the following examples

MOVC *P1 *P2

MOVC 100 200 200 XYZ

Note that the intermediate point 1, intermediate point 2 position cannot overlap with the current position point.

GOHOME

Go to home pose by point-to-point movement.

GOHOME

Move to the Home point set by the system and use the GOHOME command to move the system to Home quickly.

Point, the movement is like MOVJ, the process is not interpolated.

ROTATEX Rotate the tool frame against the X axis of global frame (base frame). Input angle ranges from -180 to 180.

ROTATEX Angle (unit: degree)

Rotate the earth coordinate X axis at the tool point position, that is, the Rx axis. Note that the command angle value cannot exceed

+180 or below -180 degrees, examples are as follows.

ROTATEX 90

ROTATEX RX

Page 95

ROTATEY Rotate the tool frame against the Y axis of global frame (base frame). Input angle ranges from -180 to 180.

ROTATEY Angle (unit: degree)

Rotate the geodetic Y axis at the tool point position, that is, the Ry axis. Note that the command angle value cannot exceed

+180 or below -180 degrees, examples are as follows.

ROTATEY 90

ROTATEY RY

ROTATEZ Rotate the tool frame against the Z axis of global frame (base frame). Input angle ranges from -180 to 180.

ROTATEZ Angle (unit: degree)

Rotate the coordinate coordinate Z axis at the tool point position, that is, the Rz axis. Note that the command angle value cannot exceed

+180 or below -180 degrees, examples are as follows.

ROTATEZ 90

ROTATEZ RZ

WAITMOTION Wait for the previous motion command ends; then Execute next commands.

WAITMOTION

Wait for the pen motion command to end before continuing with the following program.

WAIT Wait for some time; then execute next commands.
Ex: "WAIT 500" means waiting for 500ms.

WAIT Time (ms ms)

The motion waits (pauses) for a period of time (ms) before continuing with the following movements.

Page 96

BLEND Smooth the route of concecutive motions.
BLEND ON to smoothe the route.

BLEND Switch (ON/OFF)

Set whether the continuous motion is smooth. If the switch is turned on, the motion will be smoothed. If the switch is turned off, the motion will nc

Smoothing, the two will have the following differences.

If you have 2 motions,
As shown in Figure 76
Left graph, the robot
Will move like the right
Graph if BLEND is ON.
BLEND ON can make the
Movement more smooth
And quicker, but the
End-effector is not able
To arrive the midpoint
Accurate.

Figure 76 Left: no smoothing; right: smoothing

Smoothing will cause no motion to decelerate to zero between the two motion commands, making the motion look continuous and smooth.

Just like when the car makes another turn, if there is no smoothing, the car will decelerate to 0 at the turning point and then accelerate to the corner

The advantage of moving is that the movement process ensures that the position has reached the position. The downside is that every movement ha

0 (stationary) action, there will be a sense of pause; and smoothing is like the car is slightly slowed down at the turning point and then immediately

Speeding in the direction of the corner, it is necessary to make a smooth turn and move the path will change, can not move at right angles, the bene

It is that the motion looks smooth, and the downside is that the middle of the motion path point cannot ensure the arrival of the positioning. This Bl

After the setting is turned off, the subsequent motion commands will be changed, and the preset BLEND is on, so usually I

We will use the motion command with the WAITMOTION command to achieve the same effect as BLEND OFF. example

As below:

In default, BLEND is ON. You can use
WAITMOTION to achieve the same effect
As BLEND OFF.

BLEND OFF

MOVL *P1 =

MOVL *P2

Can be changed to the following meaning

MOVL *P1

WAITMOTION

MOVL *P2

HOLD Pause every motion of the robot until CONTINUE is executed.

HOLD

The motion is paused until the call to CONTINUE motion continues.

CONTINUE Resume the robot motion paused by HOLD command.

CONTINUE

The exercise continues and needs to be used with the HOLD instruction.

GETMOTIONSTATUS

GETMOTIONSTATUS Return status

Get the current state of motion, the return value represents the following:

Get motion status of the arm. There are 4 return values:
0) Running (Running)

1) Motion stop (STOP)

2) Motion pause (HOLD)

3) Exercise Waiting (WAIT)

SETACCTIME Set acceleration time in ms.

SETACCTIME	Time (unit: millisecond ms)

SETDECTIME Set Deceleration time in ms.

SETDECTIME	Time (unit: millisecond ms)

Set the motion acceleration (ACC) and deceleration (DEC) time. Each motion mode includes an acceleration segment and a deceleration segment.

The command sets the acceleration section (ACC) and the deceleration section (DEC) time. If the acceleration period is too long, the motion accele

When the speed has not been accelerated to the target speed, and the target position is reached, the motion will stop. The following figure shows th

The curve explains it.

Speed

Time

Acceleration Time Deceleration Time

Figure 77 Acceleration section and deceleration section

An example of writing is as follows:

SETACCTIME 300

SETDECTIME 300

SETACCTIME ACC

GETACCTIME Get acceleration time in ms.

GETACCTIME	Get time (unit: millisecond ms)

GETDECTIME Get Deceleration time in ms.

GETDECTIME	Get time (unit: millisecond ms)

The acceleration and deceleration time set by the speed curve is obtained as an example.

GETACCTIME ACC

GETDECTIME DEC

SETACCTYPE Set acceleration type

SETACCTYPE	Set the type of acceleration curve

SETDECTYPE Set Deceleration type

SETDECTYPE	Set the type of deceleration curve

Set the acceleration section and deceleration section curve types. There are two settings as follows, and they are explained graphically.

There are 2 types:
0) Type T T Type (Figure 78)

1) S type S Type (Figure 79)

Figure 78 T (ladder) curve

Figure 79 S-curve

Examples are as follows

SETACCTYPE 0

SETDECTYPE 1

GETACCTYPE

GETACCTYPE	Get the type of acceleration curve

GETDECTYPE

GETDECTYPE	Get the type of deceleration curve

Get the acceleration section, deceleration section curve type, and return it as a numerical value. The example is as follows

GETACCTYPE ACCtype

GETDECTYPE DECtype

GETDEG Get current degrees of joints.

GETDEG	point							
GETDEG	J1	J2	J3	J4	J5	J6	J7	J8

Get the angle value of each rotary axis command, there are two ways to write, the first one directly saves the value into one point, the second straig

The result is stored in each numerical variable. For the A-type 6-axis arm, only 6 variables need to be filled, and the 4 axes are the same.

Take the 6 axis as an example:

GETDEG *MyPoint

GETDEG J1 J2 J3 J4 J5 J6

GETPOS Get current tool frame positions

GETPOS	point							
GETPOS	X	Y	Z	A	B	C	U	V

Get the position value of each coordinate axis command, there are two ways to write, the first one directly saves the value into one point, the secon

The result is stored in each numerical variable. The variable may not be filled. If it is not filled, only the coordinate value of the filled mark will be

Take the 6 axis as an example:

GETPOS *MyPoint

GETPOS XYZ

GETPOS XYZABC

GETENCDEG

GETENCDEG	point							
GETENCDEG	J1	J2	J3	J4	J5	J6	J7	J8

Obtain the actual encoder angle value of each rotary axis, there are two ways to write, the first one directly saves the value into a point, the first

The two directly store the results in individual numerical variables. For the A-type 6-axis arm, only 6 variables need to be filled, 4

The same axis, taking the 6 axis as an example:

GETENCDEG *MyPoint

GETENCDEG J1 J2 J3 J4 J5 J6

GETENCPOS

GETENCPOS	point							
GETENCPOS	X	Y	Z	A	B	C	U	V

Obtain the position value of the actual encoder reading conversion of each coordinate axis. There are two ways to write the value. The first type dir

One point, the second directly stores the result in each numeric variable. Variables can be filled, if not filled, only back

Pass the coordinate value of the fill, taking the 6 axis as an example:

GETENCPOS *MyPoint

GETENCPOS XYZ

GETENCPOS XYZABC

GETJ1

GETJ1	Get the angle

GETJ2

GETJ2	Get the angle

GETJ3

GETJ3	Get the angle

GETJ4

GETJ4	Get the angle

GETJ5

GETJ5	Get the angle

GETJ6

GETJ6	Get the angle

GETJ7

GETJ7	Get the angle

GETJ8

GETJ8	Get the angle

Get the coordinate axis command position value (J1~J8).

GETX

GETX	Get position value

GETY

GETY	Get position value

GETZ

GETZ	Get position value

GETA

GETA	Get position value

GETB

GETB	Get position value

GETC

GETC	Get position value

Page 104

Get the command position value, X, Y, Z, A, B, C position.

Note that the J1~J8 or XYZABC values obtained by the above commands are the number of commands given to the drive by the system.

Value, and to get the actual ENC value you must use the GETENCDEG instruction, but the command value and actual

The value of the error is very small under normal conditions, so the two values are almost the same.

SETGLOBALAXIS

SETGLOBALAXIS X Y Z Rx Ry Rz

Workpiece coordinate setting, changing the origin and direction of the current geodetic coordinates, the translational rotation of the new coordinate

The steps are to turn Rx first, then Ry, then Rz, and then XYZ.

As shown above, red is the new coordinate and black is the system coordinate. If you want to define the original coordinates (100, 200, 200) as

The new origin, the program needs to be written

SETGLOBALAXIS 100 200 200 0 0 0

If the rotation is included as follows

Page 105

Then the program needs to be written

SETGLOBALAXIS 100 200 200 -90 0 0

Note that the workpiece coordinate setting is to turn first and then pan.

SETTOOLAXIS

SETTOOLAXIS X Y Z Rx Ry Rz

Tool coordinate setting, change the current tool coordinate terminal position and direction, translation rotation of new coordinates and old coordina

Quantity, the steps are to first shift XYZ and then turn to Rx and then to Ry and then to Rz.

As shown above, the tool is extended from the original terminal by 200mm, and the program is written as

SETTOOLAXIS 0 0 200 0 0 0

If the angle of the terminal machining is rotated by 45 degrees,

Then the program is written

SETTOOLAXIS 0 0 200 45 0 0

Note that the tool coordinates are translated first and then rotated.

(6) I/O operations

INPUT

INPUT	Point ID	Get the value

Read the input point change. When using this instruction, the program will not wait for the end of the exercise and then execute. The example is as

INPUT 10 i10

OUTPUT

OUTPUT	Point ID	Switch ON/OFF

Set the output point status. When using this command, the program will "wait" for the end of the motion and then execute it. The example is as foll

OUTPUT 10 ON

OUTPUT 11 OFF

OUTPUT 12 1

OUTPUT 13 0

Remarks: Before the 2.0.6.0 version of the operating software, the interpreter will stop at this line and wait for the motion to stop.

The version puts this execution command into Que and waits for execution, and the interpreter will not get stuck in this line.

OUTPUT2

OUTPUT2	Point ID	Switch ON/OFF

Set the output point status. When using this command, the program will not "wait" for the end of the exercise and then execute. The example is as 1

OUTPUT2 10 ON

OUTPUT2 11 OFF

OUTPUT2 12 1

OUTPUT2 13 0

WAITINPUTON

WAITINPUTON	Point ID

Wait until the input point is ON to continue execution. The example is as follows.

WAITINPUTON 20

WAITINPUTOFF

WAITINPUTOFF Point ID

Wait until the input point is OFF to continue execution. The example is as follows.

WAITINPUTOFF 20

(7) Simulation operation

SIMU_NORMAL

SIMU_NORMAL

Set the 3D model to display as a standard view.

SIMU_FRONT

SIMU_FRONT

Set the 3D model to display the front view.

SIMU_BACK

SIMU_BACK

Set the 3D model to display the back angle.

SIMU_LEFT

SIMU_LEFT

Set the 3D model to display as the left perspective.

SIMU_SIDE

SIMU_SIDE

Set the 3D model to display as a side view.

SIMU_TOP

SIMU_TOP

Set the 3D model to display as the top angle of view.

SIMU_BOTTOM

SIMU_BOTTOM

Set the 3D model to display as the bottom angle.

SIMU_ZOOM

SIMU_ZOOM X zoom Y zoom Z zoom

Set the 3D model to display the zoom ratio, 0.5 for 50%, as shown in the following example.

SIMU_ZOOM 0.5 1 0.5

SIMU_ADDMODAL

SIMU_ADDMODAL Numbering tool? XYZ Rxyz RGB path

Externally load the 3D model, the second parameter is set as a tool, if it is a tool, it will follow the robot after loading.

Move and move, the example is as follows.

SIMU_ADDMODAL 0 1 100 200 100 0 0 0 255 255 0 C:\mod.stl

SIMU_DELMODAL

SIMU_DELMODAL Numbering

Remove the external load 3D model, as shown in the following example.

SIMU_DELMODAL 0

(8) Communication operation

STACK_GET

STACK_GET	Get the value

Get the data from the communication (client data flow mode), the data passed to the pipe

By stacking, the command gets the oldest data.

STACK_CLEAR

STACK_CLEAR

Clear the value data register passed from the communication (client data stream mode).

NET_CNT

NET_CNT	IP	Port

Networking (client mode) connection is equivalent to operating the [Connect] button on the HMI.

NETS_CNT

NETS_CNT	Port

Networking (sever mode) is equivalent to operating the [Connect] button on the HMI.

SERIAL_CNT

SERIAL_CNT	ComPort

Performing a serial transmission connection is equivalent to operating the [Connect] button on the HMI.

NET_GETDEG

NET_GETDEG

Get the current command angle through the network (client command stream mode), the values obtained are separated by commas,

For example, the 6 axes are received, for example, 10.01, 20, 30.22, 30, 11.1, and 22.34 are received to indicate that 6 axes are received, and the an

Do not: axis 1 = 10.01, axis 2 = 20, axis 3 = 30.22, axis 4 = 30, axis 5 = 11.1, axis 6 = 22.34.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NET_GETPOS

NET_GETPOS

Get the current command location through the network (client command stream mode), the values obtained are separated by commas,

For example, the 6-axis is received, for example, 10.01, 20, 30.22, 30, 11.1, and 22.34 are received, indicating the position of the received card, its

They are X=10.01, Y=20, Z=30.22, A=30, B=11.1, and C=22.34, respectively.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NET_GETENCDEG

NET_GETENCDEG

Obtain the current actual encoder angle through the network (client command stream mode), and the obtained value is divided by a comma.

On, taking 6 axes as an example, for example, receiving 10.01, 20, 30.22, 30, 11.1, 22.34 means receiving 6 axes,

The angles are: axis 1 = 10.01, axis 2 = 20, axis 3 = 30.22, axis 4 = 30, axis 5 = 11.1, axis

6=22.34.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NET_GETENCPOS

NET_GETENCPOS

Obtain the current actual encoder position through the network (client command stream mode). The obtained value is divided by a comma.

Open, taking the 6-axis as an example, for example, receiving 10.01, 20, 30.22, 30, 11.1, 22.34 means receiving the card coordinates,

The positions are X=10.01, Y=20, Z=30.22, A=30, B=11.1, C=22.34.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NET_GETNUM

NET_GETNUM Numerical variable name

Get the variable value of the current program through the network (client command stream mode). For example, the command through the network

DIM X As Number

X=10*2+3-7*8

NET_GETNUM X

You can use this command to get the value of the variable X.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NET_GETRUNSTATUS

NET_GETRUNSTATUS

The status of the current program is obtained via the network (client command stream mode). There are three states:

1) Executing

0) Termination

-1) Suspended

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NET_MSG

NET_MSG MSG content

Get the MSG through the network (client command stream mode), just like the MSG instruction, but the result is not the output.

The system status displays the window, but the information content is transmitted over the network.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NET_GETINPUT

NET_GETINPUT Input point status

Get the current input point status via the network (client command stream mode).

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NET_GETOUTPUT

NET_GETOUTPUT Output point status

Get the current output point status through the network (client command stream mode).

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NETS_GETDEG

NETS_GETDEG

Get the current command angle through the network (sever command flow mode), the values obtained are separated by commas, to 6

For example, if the axis is received, for example, 10.01, 20, 30.22, 30, 11.1, and 22.34 are received, and 6 axes are received.

It is: axis 1 = 10.01, axis 2 = 20, axis 3 = 30.22, axis 4 = 30, axis 5 = 11.1, axis 6 = 22.34.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NETS_GETPOS

NETS_GETPOS

Get the current command location through the network (sever command flow mode), the values obtained are separated by commas, to 6

For example, if the axis is received, for example, 10.01, 20, 30.22, 30, 11.1, and 22.34 are received, and the card coordinates are received.

Others are X=10.01, Y=20, Z=30.22, A=30, B=11.1, C=22.34.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NETS_GETENCDEG

NETS_GETENCDEG

Obtain the current actual encoder angle through the network (sever command flow mode), and the value obtained is divided by a comma.

On, taking 6 axes as an example, for example, receiving 10.01, 20, 30.22, 30, 11.1, 22.34 means receiving 6 axes,

The angles are: axis 1 = 10.01, axis 2 = 20, axis 3 = 30.22, axis 4 = 30, axis 5 = 11.1, axis

6=22.34.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NETS_GETENCPOS

NETS_GETENCPOS

Obtain the current actual encoder position via the network (sever command stream mode). The value obtained is divided by a comma.

Open, taking the 6-axis as an example, for example, receiving 10.01, 20, 30.22, 30, 11.1, 22.34 means receiving the card coordinates,

The positions are X=10.01, Y=20, Z=30.22, A=30, B=11.1, C=22.34.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NETS_GETNUM

NETS_GETNUM Numerical variable name

Get the variable value of the current program through the network (sever command stream mode). For example, the command through the network

DIM X As Number

X=10*2+3-7*8

NETS_GETNUM X

You can use this command to get the value of the variable X.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NETS_GETRUNSTATUS

NETS_GETRUNSTATUS

The status of the current program is obtained via the network (sever command stream mode). There are three states:

1) Executing

0) Termination

-1) Suspended

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NETS_MSG

NETS_MSG MSG content

Obtain the MSG through the network (sever command stream mode), just like the MSG instruction, but the result is not the output.

The system status displays the window, but the information content is transmitted over the network.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NETS_GETINPUT

NETS_GETINPUT Input point status

Get the current input point status through the network (sever command flow mode).

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

NETS_GETOUTPUT

NETS_GETOUTPUT Output point status

Get the current output point status through the network (sever command stream mode).

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

SERIAL_GETDEG

SERIAL_GETDEG

Get the current command angle through the network (UART command stream mode), the values obtained are separated by commas,

For example, the 6 axes are received, for example, 10.01, 20, 30.22, 30, 11.1, and 22.34 are received to indicate that 6 axes are received, and the an

Do not: axis 1 = 10.01, axis 2 = 20, axis 3 = 30.22, axis 4 = 30, axis 5 = 11.1, axis 6 = 22.34.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

SERIAL_GETPOS

SERIAL _GETPOS

Get the current command position through the network (UART command stream mode), the values obtained are separated by commas,

For example, the 6-axis is received, for example, 10.01, 20, 30.22, 30, 11.1, and 22.34 are received, indicating the position of the received card, its

They are X=10.01, Y=20, Z=30.22, A=30, B=11.1, and C=22.34, respectively.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

SERIAL_GETENCDEG

SERIAL _GETENCDEG

Obtain the current actual encoder angle through the network (UART command stream mode). The obtained value is divided by a comma.

On, taking 6 axes as an example, for example, receiving 10.01, 20, 30.22, 30, 11.1, 22.34 means receiving 6 axes,

The angles are: axis 1 = 10.01, axis 2 = 20, axis 3 = 30.22, axis 4 = 30, axis 5 = 11.1, axis

6=22.34.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

SERIAL_GETENCPOS

SERIAL _GETENCPOS

Obtain the current actual encoder position via the network (UART command stream mode). The value obtained is divided by a comma.

Open, taking the 6-axis as an example, for example, receiving 10.01, 20, 30.22, 30, 11.1, 22.34 means receiving the card coordinates,

The positions are X=10.01, Y=20, Z=30.22, A=30, B=11.1, C=22.34.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

SERIAL_GETNUM

SERIAL _GETNUM Numerical variable name

The variable value of the current program is obtained through the network (UART command stream mode). For example, the command through th

DIM X As Number

X=10*2+3-7*8

SERIAL _GETNUM X

You can use this command to get the value of the variable X.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

SERIAL_GETRUNSTATUS

SERIAL _GETRUNSTATUS

The status of the current program is obtained via the network (UART command stream mode). There are three states:

1) Executing

0) Termination

-1) Suspended

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

SERIAL_MSG

SERIAL _MSG	MSG content

Get the MSG through the network (UART command stream mode), just like the MSG instruction, but the result is not the output.

The system status displays the window, but the information content is transmitted over the network.

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

SERIAL_GETINPUT

SERIAL _GETINPUT	Input point status

The current input point status is obtained via the network (UART command stream mode).

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

SERIAL_GETOUTPUT

SERIAL _GETOUTPUT	Output point status

The current output point status is obtained via the network (UART command stream mode).

Note: When the command is sent as a high-order command, the program interpreter will be given the highest priority and will not be the s

Execution of the robotic arm program interferes.

(9) Visual commands

VS_OPEN

VS_OPEN			
VS_OPEN	Specify camera open number (first 0, second 1, and so on)		
VS_OPEN	Specify the opening number	Specified image length	Specify image width

Turn on the camera. There are three input methods, the first one directly enters the command, and the system itself searches for the open code.

Number and resolution, the other two are the specified number and the length and width of the specified image resolution. This command is runnin

The process will wait until the camera is turned on and then proceed to the next example.

VS_OPEN

VS_OPEN 0

VS_OPEN 0 640 480

VS_OPEN2

VS_OPEN2	Specify camera ID (custom assignment, different from the open number)			
VS_OPEN2	Specify camera ID (custom assignment, and enabl		Specify camera open number (first 0, second)	
	Different numbers)		Taiwan 1, and so on)	
VS_OPEN2	Specify camera ID	Specify the opening numb	Specified image length	Specify image width

Turn on multiple cameras at the same time. There are three input methods, the first one directly enters the command and the camera ID, by the syst

The system searches for the opening number and the resolution, and the other two are the specified opening number and the specified image resolu

The length and width of the degree, this command will be executed after the camera is turned on during the running process, for example, to open t

Camera, a number 0, a number 1 can be written as:

VS_OPEN2 0 0

VS_OPEN2 1 1

The above instructions are used by the system to capture the image resolution. If you want to specify the resolution, you can write

VS_OPEN2 0 0 640 480

VS_OPEN2 1 1 1024 768

VS_CLOSE

VS_CLOSE

Turn off the camera, corresponding to the VS_OPEN command.

VS_CLOSE2

VS_CLOSE2	Specify camera ID (custom assignment, different from the open number)

Turn off the camera that is turned on with multiple cameras, and specify the camera that is turned off corresponding to the VS_OPEN2 command.

ID, such as turning off camera 1.

VS_CLOSE2 1

VS_QUERYFRAME

VS_QUERYFRAME

Get a video from the camera, the command must be used after VS_OPEN, and the camera is turned on successfully

After that, you can get an image from it, such as

VS_OPEN

VS_QUERYFRAME

VS_CLOSE

The above instruction step is to turn on the camera, then acquire another image, and finally turn off the camera.

VS_QUERYFRAME2

VS_QUERYFRAME2 Specify camera ID (custom assignment, different from the open number)

Get a video from the specified camera ID, which must be used after VS_OPEN2, and photography

After the machine is successfully turned on, you can get an image from it, such as

VS_OPEN2 0 1

VS_QUERYFRAME2 0

VS_CLOSE2 0

The above instruction step is to turn on the camera 0, then obtain a video from the camera 0, and finally the camera

0 is off.

VS_SHOW

VS_SHOW

The image acquired from the camera is displayed in the [Vision Module] window. This command must be used in

VS_QUERYFRAME After getting the image, like

VS_OPEN

VS_QUERYFRAME

VS_SHOW

VS_CLOSE

The above instruction step is to turn on the camera, and then obtain a video from the camera, and then take the captured image.

The image is displayed on the visual module window, and finally the camera is turned off.

VS_SHOW2

VS_SHOW2 Specify camera ID (custom assignment, different from the open number)...multiple groups

The image obtained from the specified camera ID is displayed in the [Vision Module] window. This command must be used in

VS_QUERYFRAME2 After the image is acquired, the command can specify to display multiple sets of cameras simultaneously.

The image sequence is arranged in the order specified by the order, such as

VS_OPEN2 0 0 640 480

VS_OPEN2 1 1 800 600

VS_QUERYFRAME2 0

VS_QUERYFRAME2 1

VS_SHOW2 0 1

VS_CLOSE2 0

VS_CLOSE2 1

The above instruction steps are to turn on cameras 0 and 1, and then take a video from cameras 0 and 1, then

Then display the images acquired by the two cameras on the same visual module window from left to right in the order of the camera.

0 Camera 1 and finally camera 0 and 1 are turned off. To display the opposite order, write it as

VS_SHOW2 1 0

VS_PATTEACH

VS_PATTEACH Object number File full path name

Page 125

Reading an image from a file and teaching it as an image object for subsequent object search (VS_PATMATCH),

Written

VS_PATTEACH 0 C:\ITRI\RobotPattern1.bmp

The above is to teach the RobotPattern1.bmp file as the image object 0.

VS_PATMATCH

VS_PATMATCH Object number Search ranking

Search for image objects, and obtain images from the previous one to perform object search tasks. This command can specify objects.

Number and search ranking, ranking points are divided into high and low points, the highest score in the first place, the second highest in the secon

Push, such as

VS_PATTEACH 0 C:\ITRI\RobotPattern1.bmp

VS_QUERYFRAME

VS_PATMATCH 0 1

The above is to teach the image object 0 first, then get a video from the camera, and then the image object

Look for, specify the search name as single 1, the result is as follows

The image search content will be framed by a red box, and the image name to be found will be displayed on it and included in the right side.

The ranking number is displayed in the arc, the first name is 0, the second name is 1... and so on, and the small brackets are displayed.

The image is searched for the score, and the score above is 91.64.

Page 126

VS_PATMATCH2

VS_PATMATCH2 Specify camera ID Object number Search ranking

Perform the search for the image object of the specified camera, and obtain the image from the previous one to perform the object search task.

Let VS_PATMATCH specify the object number and search ranking, and additionally specify the camera ID.

The same ranking scores are divided into high and low points, the highest score in the first place, the second highest in the second place... and so on

VS_ PATTEACH 0 C:\ITRI\RobotPattern1.bmp

VS_QUERYFRAME2 1

VS_PATMATCH2 1 0 2

The above is to teach the image object 0 first, then obtain a video from the camera 1, and then the image object

Search for pieces, specify the top two for the search.

VS_SAVE

VS_SAVE File full path name

For image storage, this command will save the currently acquired image, and if there is a specified ROI, the image will be saved.

The small and content will be like the area selected by the ROI. The examples are as follows.

VS_SAVE C:\ITRI\RobotImage1.bmp

VS_PATPOS

VS_PATPOS Object number Get image X value Get the image Y value

Obtain the value of the object position found after searching for the VS_PATMATCH object. You can specify the information you want to know.

The image object number, and the instruction can only obtain the image object that is searched once. Return XY value is shadow

Like the Pixel value, such as.

DIM Xpos As Number

DIM Ypos As Number

VS_QUERYFRAME

VS_PATMATCH 0 1

VS_PATPOS 0 Xpos Ypos

The above code is part of the code, not complete code (sign and ignore not to write VS_OPEN,

VS_CLOSE and other instructions)

VS_PATSCORE

VS_PATSCORE Object number Get image score

Get the score of the object found after searching for the VS_PATMATCH object. You can specify the image to be known.

Like the object number, and the instruction can only get the image object that is searched for once, for example.

DIM Score As Number

VS_QUERYFRAME

VS_PATMATCH 0 1

VS_PATSCORE 0 Score

The above code is part of the code, not complete code (sign and ignore not to write VS_OPEN,

VS_CLOSE and other instructions)

VS_PATPOS2

VS_PATPOS2 Object number Object ranking Get image X value Get the image Y value

Obtain the value of the object position found after searching for the VS_PATMATCH object. You can specify the information you want to know.

The image object number, and the command can specify the image object of the ranking. Return Pixel with XY value as image

Value, such as the second.

DIM Xpos As Number

DIM Ypos As Number

VS_QUERYFRAME

VS_PATMATCH 0 3

VS_PATPOS2 0 2 Xpos Ypos

The above code is part of the code, not complete code (sign and ignore not to write VS_OPEN,

VS_CLOSE and other instructions)

VS_PATSCORE2

VS_PATSCORE2 Object number Object ranking Get image score

Get the score of the object found after searching for the VS_PATMATCH object. You can specify the image to be known.

Like the object number, and the instruction can specify the image object of the ranking, such as the second name.

DIM Score As Number

VS_QUERYFRAME

VS_PATMATCH 0 3

VS_PATSCORE2 0 2 Score

The above code is part of the code, not complete code (sign and ignore not to write VS_OPEN,

VS_CLOSE and other instructions)

VS_MSG

VS_MSG X Y size R G B message

The MSG command of the image, like the MSG command, but the output object is not the system status display window, but

On the image, such as

VS_MSG 100 100 20 255 0 0 ITRI Robot

The above formula writes an ITRI Robot string with a red size of 20 at the (100, 100) position.

VS_MSG2

VS_MSG2 Specify camera ID X Y size R GB message

The MSG command of the image, like the MSG command, but the output object is not the system status display window, but

On the image, this command can additionally specify the camera ID, such as

VS_MSG2 1 100 100 20 255 0 0 ITRI Robot

The above formula writes an ITRI Robot string with a red size of 20 on the camera 1 (100, 100).

VS_SETROI

VS_SETROI X Y W H

Set the image ROI, which is the area of interest of the image. For example, we only need to perform image processing on a certain area.

To set the ROI, you must call VS_RESETROI, otherwise the image ROI will always be set as above, and the command will

Affects all visual instructions. The following example sets the image ROI and then retrieves the image.

VS_SETROI 0 0 100 100

VS_SAVE C:\ITRI\RobotROI1.bmp

Figure 80 Original image and set ROI

VS_SETROI2

VS_SETROI2 Specify camera ID X Y W H

Set the image ROI, which is the area of interest of the image. This command is the same as VS_SETROI.

Specify the camera ID additionally.

VS_RESETROI

VS_RESETROI

Reset the image ROI.

VS_RESETROI2

VS_RESETROI2 Specify camera ID

Reset the image ROI and specify the camera ID.

VS_PEN

VS_PEN Brush width R G B

Set the brush for image drawing to set the width and color, such as

VS_PEN 2 255 0 0

VS_LINE

VS_LINE X1 Y1 X2 Y2

The image can be drawn in a straight line, and the VS_PEN setting brush must be called before calling this command.

(X1, Y1)

(X2, Y2)

Examples are as follows

VS_PEN 2 255 0 0

VS_LINE 100 200 300 300

VS_LINE2

VS_LINE2 Specify camera ID X1 Y1 X2 Y2

Line drawing of the image of the specified camera ID can be made, and VS_PEN must be called before calling this command.

Set the brush, the same as VS_LINE, the example is as follows.

VS_PEN 2 255 0 0

VS_LINE2 1 100 200 300 300

VS_RECTANGLE

VS_RECTANGEL X Y width high

You can make a rectangular drawing of the image, and you must call the VS_PEN setting brush before calling this command.

(X, Y)

width

high

Examples are as follows

VS_PEN 1 0 0 255

VS_RECTANGLE 10 10 200 100

VS_RECTANGLE2

VS_RECTANGEL2 Specify camera ID X Y width high

You can make a rectangular drawing of the image of the specified camera ID, and you must call VS_PEN before calling this command.

Set the brush, the same as VS_RECTANGLE, the example is as follows.

VS_PEN 1 0 0 255

VS_RECTANGLE2 0 10 10 200 100

VS_CIRCLE

VS_CIRCLE CX CY radius

You can round the image and call the VS_PEN setting brush before calling this command.

(CX, CY)

Examples are as follows

VS_PEN 1 0 0 255

VS_CIRCLE 10 10 200

VS_CIRCLE2

VS_CIRCLE2 Specify camera ID CX CY radius

You can make a circular drawing of the image of the specified camera ID, and you must call VS_PEN before calling this command.

Set the brush, the usage is the same as V_SCIRCLE, the example is as follows.

VS_PEN 1 0 0 255

VS_CIRCLE2 2 10 10 20

VS_PATMATCH3

VS_PATMATCH3 Object number Rotation interval angle

Perform target image object comparison search (including rotation), in addition to finding the position, you can simultaneously find the rotation an

Call VS_QUERYFRAME to get an image before using this command, for example

VS_QUERYFRAME

VS_PATMATCH3 0 10

The above instructions are for an alignment every 10 degrees.

VS_PATMATCH4

VS_PATMATCH4 Specify camera ID Object number Rotation interval angle

Perform the target image object comparison search (including rotation) of the specified camera ID, except for finding the position

When looking for the rotation angle, you must call VS_QUERYFRAME2 to get an image before using this command.

E.g

VS_QUERYFRAME2 1

VS_PATMATCH4 1 0 10

The above instructions are for an alignment every 10 degrees.

VS_PATROT

VS_PATROT Object number Get the angle of rotation

Obtain the image rotation angle after the VS_PATMATCH3 and VSPATMATCH4 commands are judged.

DIM Rot As Number

VS_PATROT 0 Rot

VS_ROTATE

VS_ROTATE Rotation angle

Perform image rotation processing and rotate clockwise to positive. The example is as follows

VS_ROTATE 45

Page 135

Original image After rotating 45 degrees

The image above shows the image after the original image has been rotated 45 degrees.

VS_ROTATE2

VS_ROTATE2 Specify camera ID Rotation angle

Perform image rotation processing for the specified camera ID, clockwise rotation is positive, usage is VS_ROTATE

Same as the example below

VS_ROTATE2 1 45

VS_GRAY

VS_GRAY

VS_GRAY2

VS_GRAY2 Specify camera ID

Perform (specify camera ID) image grayscale processing.

Page 136

Original image Gray scale

VS_THRESHOLD

VS_THRESHOLD Threshold

VS_THRESHOLD2

VS_THRESHOLD2 Specify camera ID Threshold

Perform (specify camera ID) image Threshold processing.

Original image THRESHOLD processing

VS_CANNY

VS_CANNY Threshold

VS_CANNY2

VS_CANNY2 Specify camera ID Threshold

Perform (specify camera ID) image Canny processing.

Original image Canny processing

VS_ERODE

VS_ERODE repeat times

VS_ERODE2

VS_ERODE2	Specify camera ID	repeat times

Perform (specify camera ID) image erosion processing.

Original image Erosion treatment

VS_DILATE

VS_DILATE	repeat times

VS_DILATE2

VS_DILATE2	Specify camera ID	repeat times

Perform (specify camera ID) image expansion processing.

Original image Expansion treatment

VS_LAPLACE

VS_LAPLACE	Sigma value

VS_LAPLACE2

VS_LAPLACE2	Specify camera ID	Sigma value

Perform (specify camera ID) image Laplacian processing.

Original image Laplace processing

VS_HARRISCORNER

VS_HARRISCORNER size

VS_HARRISCORNER2

VS_HARRISCORNER2 Specify camera ID size

Perform (specify camera ID) image HARRIS corner detection processing.

corner

corner

Original image HARRIS corner processing

VS_SUM

VS_SUM Ch1 Ch2 Ch3 Ch4

VS_SUM2

VS_SUM2 Specify camera ID Ch1 Ch2 Ch3 Ch4

Perform (Specify Camera ID) image pixel sum calculation to calculate 4 channel values, as shown in the following example.

DIM CH1 As Number

DIM CH2 As Number

Page 140

DIM CH3 As Number

DIM CH4 As Number

VS_SUM CH1 CH2 CH3 CH4

VS_CALIBRATION

VS_CALIBRATION	Long corner points	Short corner points	Number of times	Interval ms

VS_CALIBRATION2

VS_CALIBRATION2	Specify camera ID	Long corner points number	Short corner points number	Number of times	Intervals Ms

Perform (specify camera ID) image correction, input the long side short corner points of the grid version, and image correction

The number of times the material is taken and the interval between each time is completed. After the calibration is completed, the relevant data wil

The image correction mode is not turned on until VS_UNDISTORTMAP is called. The example is as follows.

VS_CALIBRATION 6 8 10 500

Note: Calling this command does not require calling VS_OPEN to open the camera. This command only performs the calibration procedure.

Usually only need to be corrected once.

VS_UNDISTORTMAP

VS_UNDISTORTMAP

VS_UNDISTORMAP2

VS_UNDISTORTMAP2	Specify camera ID

(Specify camera ID) Image correction mode is turned on, which can be used for cameras that have been corrected.

Page 141

Image correction mode is enabled, and image correction is performed with VS_UNDISTORFRAME.

VS_UNDISTORTFRAME

VS_UNDISTORTFRAME

VS_UNDISTORTFRAME2

VS_UNDISTORTFRAME2 Specify camera ID

Perform (specify the camera ID) image correction, perform image correction on the current frame, and call this command first.

Correct the mode VS_UNDISTORTMAP and select an image VS_QUERYFRAME, then

Can be corrected, for example, we have to get the corrected image can be written as:

VS_UNDISTORTMAP

VS_OPEN

VS_QUERYFRAME

VS_UNDISTORTFRAME

VS_SHOW

VS_CLOSE

The above command first turns on the image correction mode (just one time), then turns on the camera to capture the image, then

Call VS_UNDISTORTFRAME for correction, and finally display and end.

(10) Macro use

MACRO

MACRO Variable 1 Variable 2 Variable 3 ... Variable N

The robotic language main program can call the subprogram (macro) through the interconnection between the IRB files, for example

The main program file name is MAIN.irb, and the subprogram (macro) file name is MACRO1.irb, then I

We can put two files in the same folder, and the software will automatically recognize the subprogram (macro).

Executed by MAIN.irb calling MACRO1.irb.

In the MACRO1.irb file, the program contents are as follows

MSG This is macro file.

GOHOME

We executed the MSG command in the subroutine and GOHOME back to the HOME movement.

And in MAIN.irb, we write

MSG This is main file.

MACRO1

We executed the MSG command and the call subroutine MACRO1 in the main program.

Simply use the subroutine and write your own common macros to make the code more concise.

In addition, if we want to call a macro while passing variables, we can make Add(a, b) similar to C++.

The effect, how to write a macro? First we must first open a new file and store the file name Add, then in

The [Motion Control] window teaches two numeric variables as the incoming variables. As shown below, select the variable type.

[Value], then click [Add] to add the numeric variable a and the numerical variable b, and the initial value is 0.

Generate number 0 variable a, and number 1 variable b, type is numeric, initial value is 0, note here

The numbers of a and b cannot be changed arbitrarily because the numbering of the numbers 0, 1, 2, 3... and the subprogram (macro) are passed.

The order of the variables is related, so the first variable passed by our Add(a, b) function is a, the corresponding

The number is 0, the second variable is b, the corresponding number is 1, then we want the Add subroutine

(macro) returns the result of the calculation back, so we must also declare a numeric variable for the program to return.

Value, so we add the third variable c as shown below

Finally, we write the following code in Add.irb

c = a+b

Very simple line of code, so we will complete the writing of the subprogram (macro), try to add MSG &c

Execute to see, we can know that the output is 0, because a=0, b=0, a+b=0, we are currently

The main program has not yet called the subprogram (macro), so both a and b have not been assigned. There is another one to pay attention to here

The variable declared in the subroutine cannot be the same as the main program variable name, otherwise an error will be generated.

After writing the subroutine (macro), write the main program, first open a robot called CallAdd.

IRB file, then write the following code

DIM result As Number

Add 1 2 &result

MSG result = &result

The above code we call Add subroutine (macro), then calculate the result = 1+2 in the subroutine, here

Did you find it? The variable value to be returned needs to be preceded by the variable "&" symbol, which is in the subprogram.

In the calculation, a = 1, b = 2, result = c, the following is the program execution result.

In addition, the subprogram (macro) can also pass strings and coordinate points, but note that the use of these two variables requires

Plus the preamble, examples are as follows

MACRO1 Num $ITRI *Point

The above command calls the MACRO1.irb subroutine (macro) and passes the value Num, the string ITRI, and the point.

Point.

Page 145

Chapter 4 Robot Arm Sample Code

A. Variables and four arithmetic operations

```
% Dimentions

DIM Var1 As Number

DIM Var2 As Number

DIM Var3 As Number          Declare values, strings, coordinate points variables

DIM Var As Number

DIM Str As String

DIM Pt As Point

% Variables Initial

Var1 = 2

Var2 = 5                    Given numerical initial value

Var3 = 11

SETSTRING $Str This_is_a_string          Given string initial value

GETDEG *Pt                  Given the initial value of the coordinate point

% Calculate

Var = Var1 * (3+2*Var2) / (Var3 +1 )     Arithmetic

% Show Result
```

Page 146

```
MSG Var = &Var , Str = &Str
```

Display window output results in system status

```
MSG Pt = &Pt
```

The following is the program execution result

B. Mathematical function use

```
% Dimentions

DIM a As Number

DIM b As Number

DIM c As Number

DIM d As Number

DIM e As Number

% Math

a = 1+3*4/5

SIN 30 b

ATAN2 4 3 c

RAND 100 d

POW 10 2 e

% Print
```

Declare variables

Perform four arithmetic operations and use mathematical functions


```
MSG a = &a

MSG b = &b

MSG c = &c

MSG d = &d

MSG e = &e
```

Display results in the system status window

The following is the program execution result

C. Discriminant and loop use

```
% Dimention

DIM x1 As Number

DIM x2 As Number

DIM x3 As Number

DIM x4 As Number

DIM x5 As Number

DIM x6 As Number

DIM x7 As Number

% Calculated
```

Declare variables


```
X1 = 1

WHILE     X1<=10

    MSG number & x1

    X2 = 2*3+(3*(4-1)) / (x1+1)

    LOG10 20+x1 x3

    DEG2RAD 180-x1 x4

    CEIL     22.357575*x1 x5

    X6 = x1/10

    IF     X6 >= 0.5

        X7 = x1+x2+x3+x4+x5+x6

    ENDIF

    % Print
```

Given initial value

WHILE loop, x1 variable is constantly behind

+1, until the time is greater than 10, jump out of the loop

IF discriminant, if x6 is greater than or equal to 0.5, enter the following execution

```
MSG x1 = &x1
MSG x2 = &x2
MSG x3 = &x3
MSG x4 = &x4
MSG x5 = &x5
MSG x6 = &x6
MSG x7 = &x7
```

Display results in the system status window


```
MSG -
SLEEP 10000
X1 = x1 +  1
LOOP
```

bel Bro gram is Res ted f the

D. Basic Motion Control Commands

```
% Dimention  n
DIM j1 As Nu Umber
DIM j2 As Nu Umber
DIM j3 As Nu Umber
DIM j4 As Nu Umber
```

An Rep the Deg of

```
DIM j5 As Nu Umber

DIM j6 As Nu Umber
```



```
DIM x As Number

DIM y As Number

DIM z As Number

DIM a As Number

DIM b As Number

DIM c As Number

DIM pt As Point

% Set Value

GETDEG j1 j2 j3 j4 j5 j6

GETPOS xyzabc

% Print Positions

MSG x= &x , j1= &j1

MSG y= &y , j2= &j2

MSG z= &z , j3= &j3

MSG a= &a , j4= &j4

MSG b= &b , j5= &j5

MSG c= &c , j6= &j6

GETPOS *pt

% Motion Speed
```

Declare position coordinate variable

Get the current axis angle value and coordinate position value

Display data in the system status window

```
SETPTPSPEED 30
```

Set point-to-point speed and linear movement speed

```
SETLINESPEED 50

% Motion Control

GOHOME

WAITMOTION

MOVJ j1+20 j2 j3 j4/2+10 j5 j6

MOVP xyzabc

WAITMOTION
```

Motion command

```
MOVL x+30 yzabc
```

(MOVJ, MOVP, MOVL, MOVT, GOHOME)

```
WAIT 1000

MOVL *pt

WAITMOTION

MOVT 0 0 30 0 0 0

GOHOME

% Print
```

End of the run in the system status window

```
MSG Run Stop
```

E. Basic I/O control

```
% Dimention

DIM i1 As Number
```

Declare the state variable that stores the Input point (input point)


```
DIM i2 As Number
```

Declare the state variable that stores the Input point (input point)

```
WHILE i1 == 0

    INPUT 0 i1

    INPUT 1 i2

    IF i2 == 1
```

Get the status of input points 0, 1, and give the status value to i1, i2

```
        MSG Input 1 on , output 1 on
        OUTPUT 1 ON
    ENDIF
    IF i2 == 0
        MSG Input 1 off , output 1 off
        OUTPUT 1 OFF
    ENDIF
    SLEEP 500
LOOP
```

If input point 2 is ON, output point 1 is enabled.

If input point 2 is OFF, output point 1 can be turned OFF.

F. EXTREG use

```
EXTREG_SET TESTREG 30
DIM X As Number
EXTREG_GET TESTREG X
MSG X= &X
```

Set the TESTREG value to 30

Give the TESTREG value to X

The following is the result of the execution

G. The camera basically captures images

```
% Camera Open
VS_OPEN
WHILE 1 == 1
    % Get Frame
    VS_QUERYFRAME
    % Show Image
    VS_SHOW
    % buf time
```

Turn on the camera (automatically find)

Capture images and display

```
    SLEEP 100
LOOP
%Close Camera
VS_CLOSE                Turn off the camera
```

H. Basic image processing

```
% Camera Open
VS_OPEN                 Turn on the camera (automatically find)
WHILE 1 == 1
```



```
    VS_QUERYFRAME
    VS_SHOW
    SLEEP 100
    %
    VS_QUERYFRAME
    VS_THRESHOLD 30
    VS_SHOW
    SLEEP 100
    %
    VS_ QUERYFRAME
    VS_CANNY 10                 Basic image processing
    VS_SHOW
                                (THRESHOLD, CANNY, LAPLACE)
    SLEEP 100
    %
    VS_ QUERYFRAME
    VS_LAPLACE 0.3
    VS_SHOW
```

```
    SLEEP 100
LOOP
```



```
%Close Camera
VS_CLOSE          Turn off the camera
```

I. Object comparison tracking

```
VS_PATTEACH 0 D:\Pattern.bmp          Perform image object teaching
VS_OPEN
WHILE 1==1
    VS_QUERYFRAME
    VS_PATMATCH 0 1          Perform image object comparison tracking
    VS_SHOW
LOOP
VS_CLOSE
```

J. Turn on multiple cameras

```
VS_OPEN2 0 0
                    Turn on camera 0 and camera 1
VS_OPEN2 1 1
WHILE 1==1
    VS_QUERYFRAME2 0
    VS_QUERYFRAME2 1          Capture two camera images and display
    VS_SHOW2 0 1
```

```
LOOP

VS_CLOSE2 0                    Turn off two cameras

VS_CLOSE2 1
```

K. Movement suspension and continuation

```
SETPTPSPEED 10

MOVJ 0

GOHOME

MOVJ 0

MSG 1

WAITMOTION

HOLD

SLEEP 1000                     Pause for 1 second and continue

CONTINUE

GOHOME

MOVJ 0

GOHOME

MSG 2
```


L. Using camera calibration

```
VS_OPEN

VS_UNDISTORTMAP                Turn on camera correction

WHILE 1==1
```

```
    VS_QUERYFRAME
    VS_UNDISTORTFRAME        Correction of the previous image
    VS_SHOW
LOOP
VS_CLOSE
```

M. Communication register read

```
DIM X As Number
DIM Y As Number
STACK_CLEAR
WHILE 1
    STACK_GET XY             Read the communication register data
    MSG X= &X Y= &Y
    SLEEP 500
LOOP
```

N. Communication data flow

```
DIM X As Number
DIM Y As Number
WHILE 1
    X=X+1
    RAND 100 Y
```



```
    MSG &X &Y
    NET_MSG &X , &Y          Send data through the Internet Client
    SLEEP 500
LOOP
```

Appendix 1 Operation steps

1 Make sure the power supply box is turned off.

OK closed

2 Determine the completion of the wiring action (the wiring has a foolproof measure)

Main power connector

(Please determine the voltage)

Page 160

Screen power

Screen signal

Emergency stop button

Keyboard, mouse

After picking up

3 Turn on the power box power

Please refer to Chapter 2 for this section.

4 Start using software operation

Please refer to Chapter 3 for this section.

Page 161

Appendix 2 Boot Error Number Meaning

Numbering	significance
1001	The kinematic library initialization failed (the kinematic parameters are set incorrectly)
1002	Kinematic library failed to load (missing RobotKinematics.dll file)
1003	Motion control library initialization failed
	1. There is a problem with the axis card
	2. The emergency switch is not released
	3. Absolute encoder read failed
	4. Drive exception
	5. I/O exception
1004	Motion control library failed to load (missing RobotMCCL.dll file)
1005	Simulation library initialization failed
	1. Missing glut32.dll file
	2. Missing simulation model STL file
1006	Simulation library failed to load (missing RobotSimulation.dll file)
1007	Communication library initialization failed (communication parameter setting is incorrect)
1008	Communication library failed to load (missing iCommunications.dll file)
1009	Visual library initialization failed (missing cv200.dll, cxcore200.dll,

Page 162

	Highgui200.dll)
1010	Visual library failed to load (missing RobotVision.dll file)
1011	License account error
1012	License password is incorrect

1013	License verification error

Appendix 3 Interpreter Error Messages

Error number	error name	Description
0	normal	
1	The interpreter could not find the relevant instruction	Not in instruction
2	Variable names cannot be associated with system commands	Variable names are the same as the Robot Language command
	the same	Stacking, generating errors
3	IF~ENDIF or WHILE~LOOP,	IF corresponds to ENDIF
	The number corresponds incorrectly	WHILE corresponds to LOOP
4		

	The command is not used correctly	The amount of arguments passed to the command is incorrect.
5	Found an error under this command layer	Such as subprogram, IF, WHILE layer
6	Command level error	Level exceeds limit (25*25 layers)
7	Contains undefined variable names	Variables are only valid in this layer area