TT ROBOT
天太机器人

USER MANUAL

MDCS-510 DRIVE & CONTROL INTEGRATED SYSTEM

Version V1.0

USER MANUAL OF MDCS-510 DRIVE & CONTROL INTEGRATED SYSTEM

V1.0

History

Version	Amendment	By	Date
V1.0	New Create	Wang Peng	2022-1-12

Alert	Please make sure that you have read this manual carefully or received sufficient professional training before using it, otherwise there may be mechanical or personal hazards!
Electric Shock	Do not place the system in a place where liquid and metal debris will splash. If environmental conditions cannot be avoided, a protective cover must be set up.

Content

Chapter 1 Specification

1.1 Specification parameters

Model	MDCS-510				
Input voltage	AC 200V~240V, 50/60Hz Single Phase				
Max. Input	3.0 KW				
Max. Output	J1 750W	J2 750W	J3 750W	J4 750W	J5 750W
Temperature	-25°C~40°C				
Humidity	20%~80% (non-condensing)				
Digital I/O		Input	Output		
	Qty	32points	32points		
	Type	NPN/PNP (Dial switch)	Fixed output 0V		
	Current	N/A	Leaky output≤167mA		
Communication Port	USB Port	1pcs (version upgrade)			
	Serial comm	RS485×1, RS232×1			
	Net comm	RJ45×1 (Software: 4routes)			
	Fieldbus	Modbus RTU/TCP Slave mode EtherCAT×1(non-support so far)			
Teach Pendant	Screen	8'' TFT			
	Resolution ratio	800*600			
	Touch type	Resistance type			
	Mode switch	Switch type (Manual, Automatic, Remote)			
	Emergency stop	1			
	Enabled switch	3-shift type 1pc			
	External port	USB×1			
Weight	Controller	About 8 Kg			
	Teach Pendant	855g			
Size	Controller	L367×W210×H256mm			
	Teach Pendant	L270×W194×H59mm			

Chapter 2 Controller introduction

2.1 Controller 2D drawing

MDCS-510 controlling system structure and size as below:

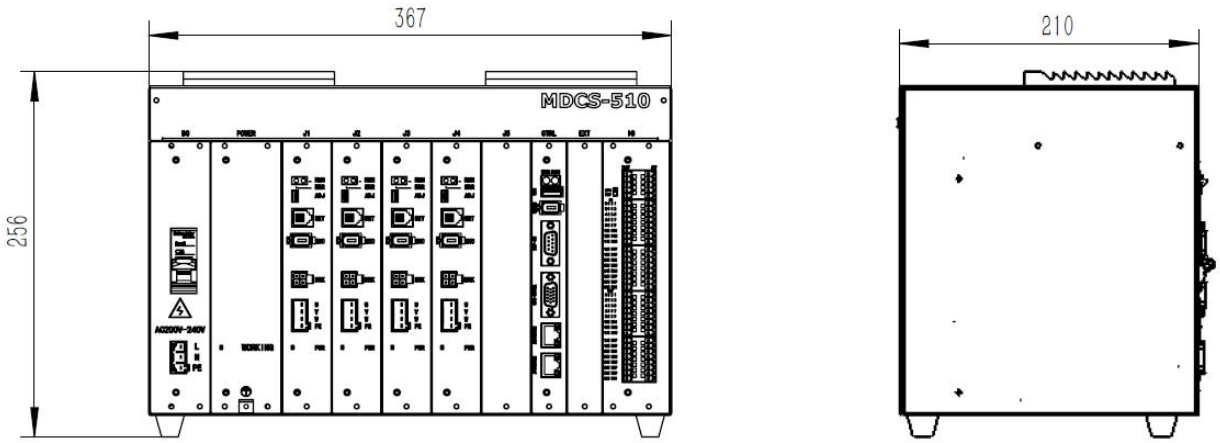


FIG. 2-1 Outer frame size drawing

Alert	When put the control box: 1.Reserve enough connection space on the interface panel, around >50mm; 2.Do not block the cooling fan on the side of the control cabinet, and reserve space > 100mm.

2.2 System terminal description

System terminal panel layout as below picture 2-2 shown:

FIG. 2-2 terminal panel layout

Name/marks	Function	Name/marks	Function
ON<->OFF	Power switch	EXTIO	Extended I/O port
AC220V	Power terminal	RS485	Serial interface
ADJ	Servo adjusting terminal	TEACH-PAD	Teach Pendant terminal
SET	Servo Ext board interface	ETHERNET	network port
ENC	Motor encoder interface	ETHERCAT	BI (non-support)
BRK	(J3)Braking wiring port	NPN-PNP	Input level dial-up
U V W PE	Motor power wiring port	RUN	Running indicator
USB	software upgrade port	ERR	Alarming indicator
24V	Power 24V	PWR	Servo power indicator
0V	Power 0V	⏚	Earthing port

Chapter 3 Wire Connection definition

3.1 Digital Input Schematic diagram

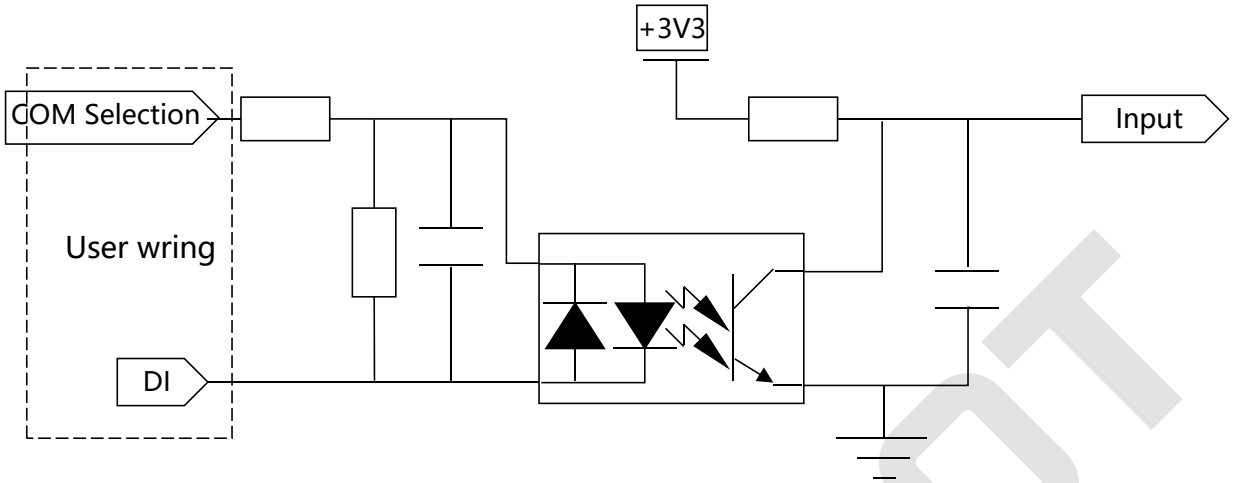


FIG. 3-1 Digital Input Schematic diagram

Alert	NPN type input device, please switch the switch to NPN side; PNP type switch to PNP side; Whether the system internal 24V power supply, the actual measurement shall prevail.

3.2 Digital Output Schematic diagram

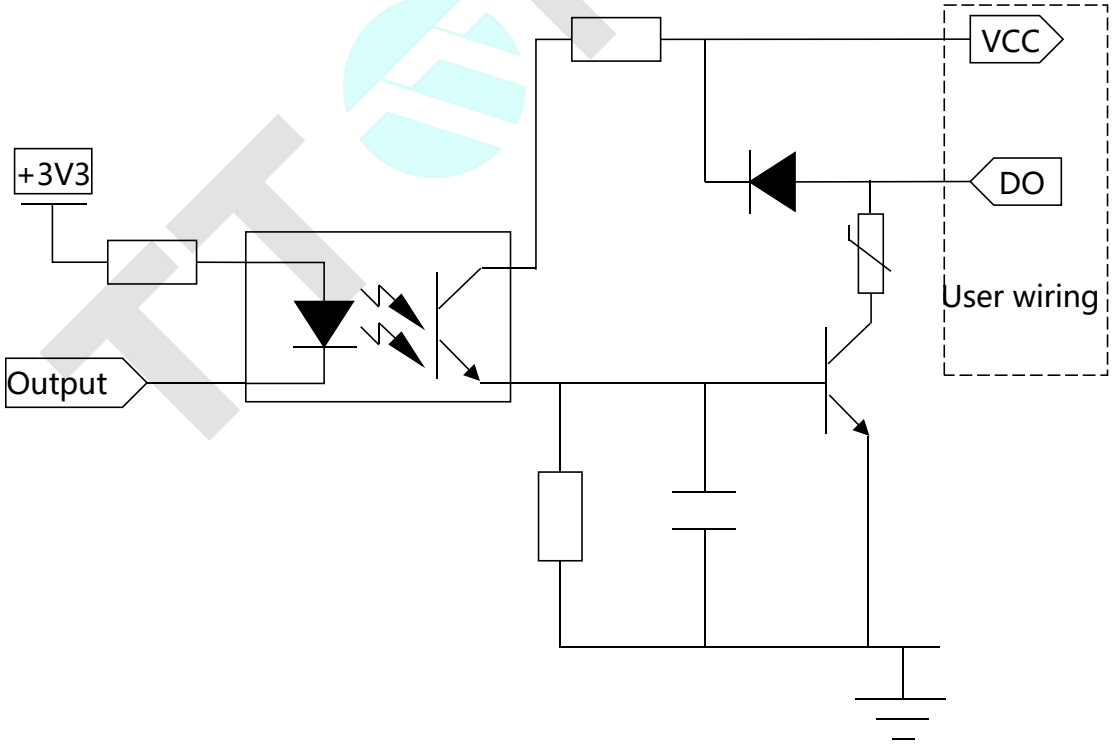


FIG. 3-2 output circuit

3.3 Local I/O input output terminal

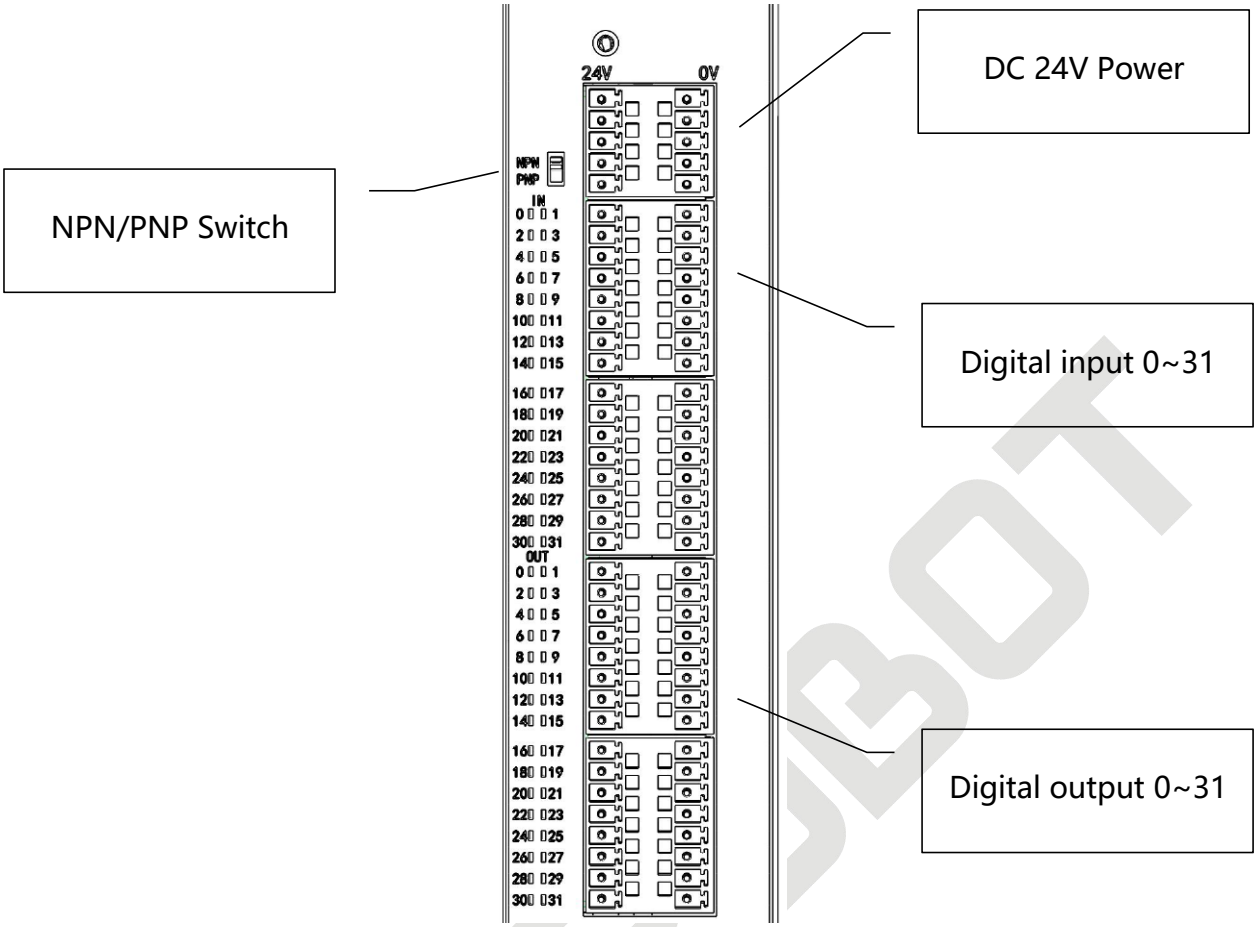


FIG. 3-3 Local I/O Layout

Terminal	Description
Switch power supply access terminal	1. I/O inner no power, user need wiring outer power (Pre-install on special order) 2. Input port, output port, sharing power
Input terminal	1. Input port total 32pcs 2. The input port can select the input level as NPN or PNP through the level property dip switch
Output terminal	1. Output port total 32pcs 2. The output port is specified as 0V output

3.4 EXT IO terminal connection definition

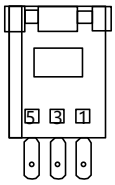


FIG.3-4-1 RS232/485 terminal wiring drawing

Pin definition EXTIO (1394 female)			
1	Internal occupy	**4**	EXT485_A-
2	EXT485_A+	**5**	blank
3	Internal occupy	**6**	blank

Tips	About EXTIO ports: 485+ and 485- constitute the 485 I/O expansion bus respectively. Extended I/O devices connect to the 485 I/O expansion bus respectively.

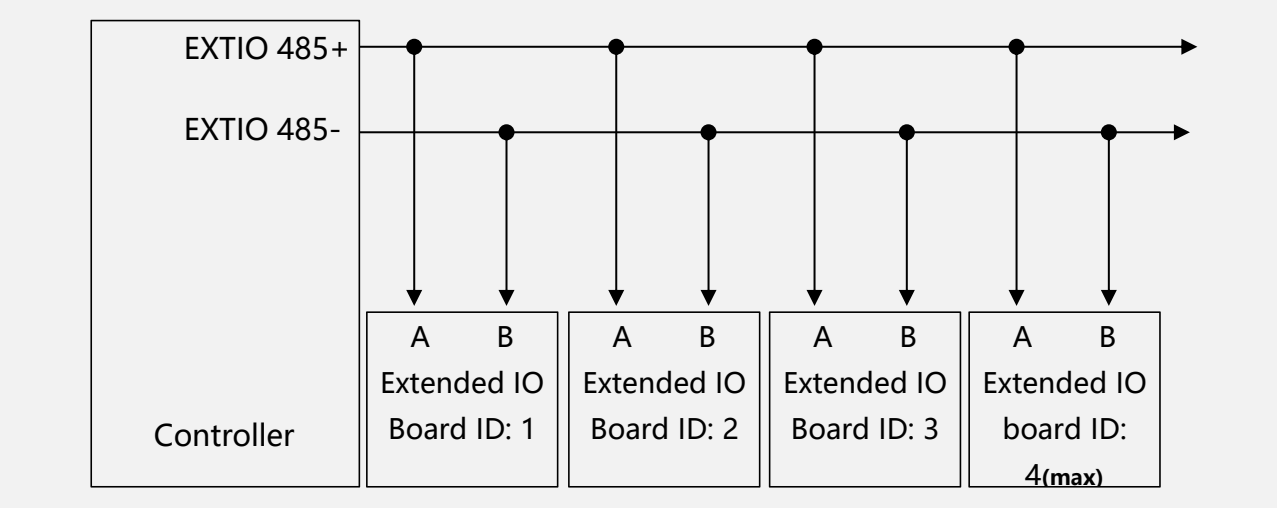


FIG.3-4-2 Extended bus diagram

3.5 Extended I/O Board connection definition (option)

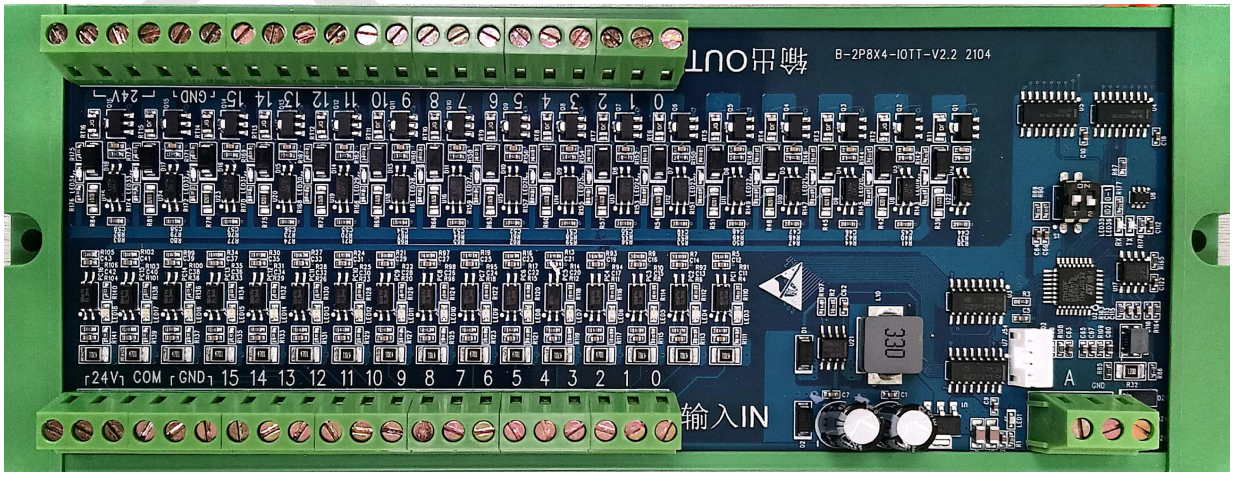


FIG. 3-5 extended IO board

3.5.1 Connection terminal description

Terminal	Description
24V	1.Users need to configure an additional 24V power supply to supply power to the expansion I/O board 2.Input and output between 24V and 0V inner is connected 3.Do not use the system IO 24V power supply for the expansion I/O board. The design power of the dc 24V power supply does not include the consumption of the expansion I/O board.
COM	1.NPN type (active low level) input device, COM terminal shorted 24V 2.PNP type (high level effective) input device, COM terminal shorted 0V; 3.The output is fixed at 0V, independent of this terminal.
GND	1.0V of external 24V power supply 2.Input and output 0V is connected inside the extended board.
0....15	Input and output wiring pins
B A GND	Communication pin, B to 485-, A to 485+ This system 485 communication adopts 2-wire system, and GND does not need wiring

3.5.2 Installation Description

Size	Description
Length	218mm
Width	86mm
Installation way	Suitable for 35mm standard guide rail, buckle type installation

Tips	For details about how to use extended I/O, see 7.2.4 Extended I/O details of setting instruction.

3.6 RS232/485 terminal connection definition

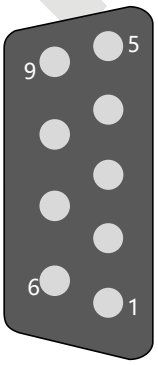


FIG. 3-6 RS232/485 terminal connection

RS232/485(DB9-F) Pin definition			
1	1#485+	**6**	1#485-
2	232RX	**7**	blank
3	232TX	**8**	blank
4	blank	**9**	blank
5	GND		

3.7 HIMI terminal connection define

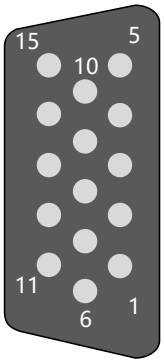


FIG. 3-7 HMI terminal connection

HMI(DB15-3R-F) Pin definition					
1	NET-RX+	**6**	NET-TX+	**11**	NET-TX-
2	HMI485+	**7**	NET-RX-	**12**	blank
3	IN4	**8**	HMI485-	**13**	blank
4	IN2	**9**	IN3	**14**	blank
5	GND	**10**	IN1	**15**	24V

3.8 Definition of connecting motor connector wiring

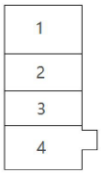


FIG. 3-8-1 Motor power line wiring diagram

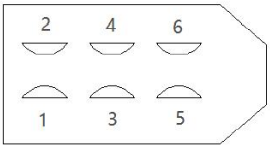


FIG.3-8-1 Motor Encoder line wiring diagram

HMI(DB15-3R-F) Pin definition	
1	U
2	V
3	W
4	PE
1	5V
2	D+
3	0V
4	D-

3.9 BRK motor brake terminal wiring definition

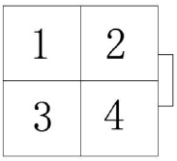


FIG.3-9 BRK Motor brake wiring diagram

HMI(DB15-3R-F) Pin definition	
1	BRK-
2	BRK+
3	24V
4	Blank

3.10 Earthing access attention

3.10.1 Earthing access standard

a. The length of the controller grounding screw to the grounding terminal should not exceed 20cm

b. Ground earthing cable specification: not less than 0.75 square yellow and green wire

3.10.2 Earth wiring illustration

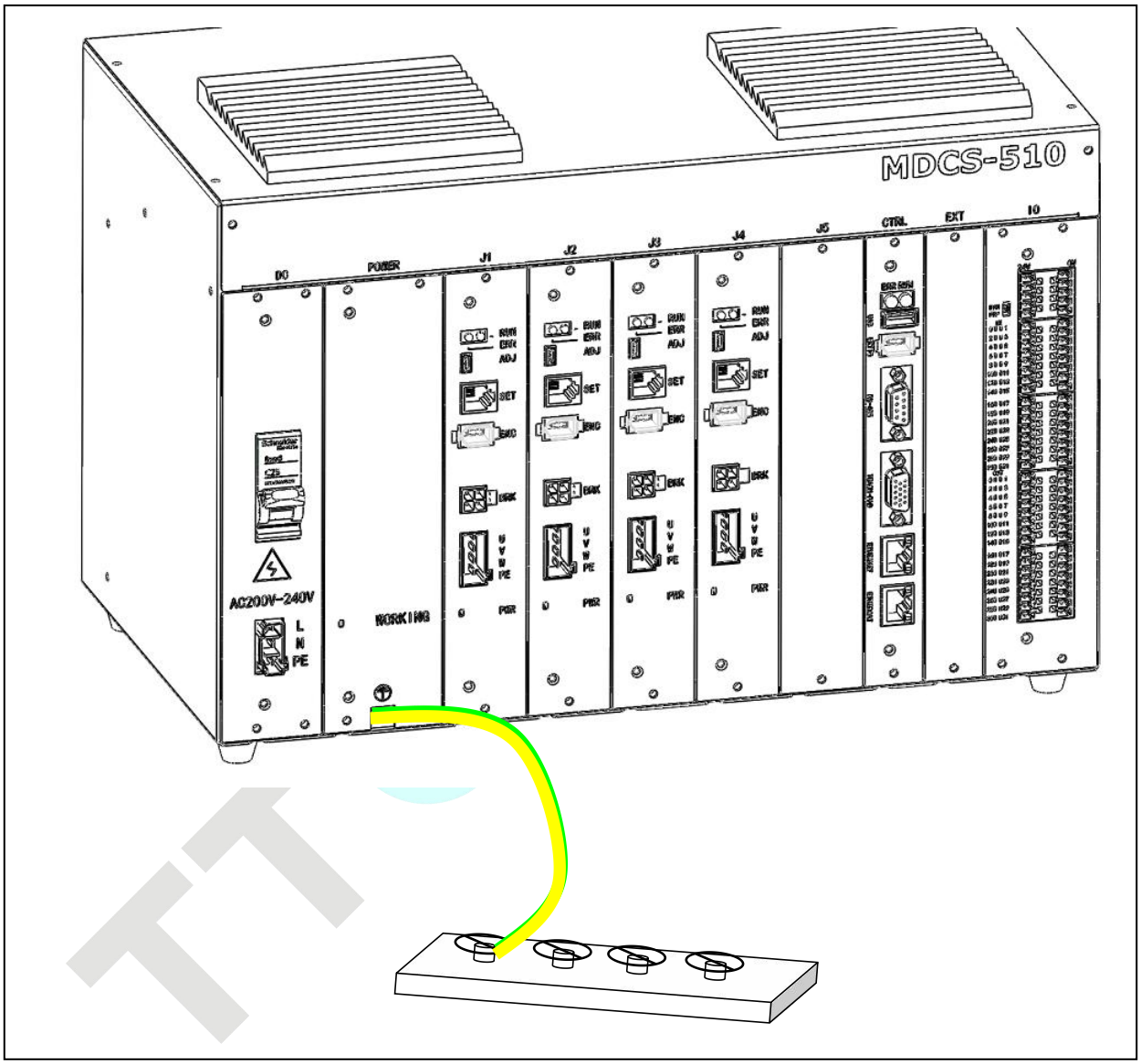


FIG. 3-10 Controller earth wiring illustration

Alert	Ensure the system is reliably grounded; otherwise, communication-related faults may occur.

Chapter 4 Teach Pendant introduction

4.1 Teach Pendant layout

As photo shown 4-1, Front side from left to right: indicator, screen, buttons, mode switch and emergency stop.

FIG. 4-1 Teach Pendant front side

As photo shown 4-2, equipped with USB Port and touch pen at the bottom.

FIG. 4-2 Teach Pendant bottom side

As FIG. 4-3 shown, equipped with manual robot enabled switch at the back side (yellow one on the picture)

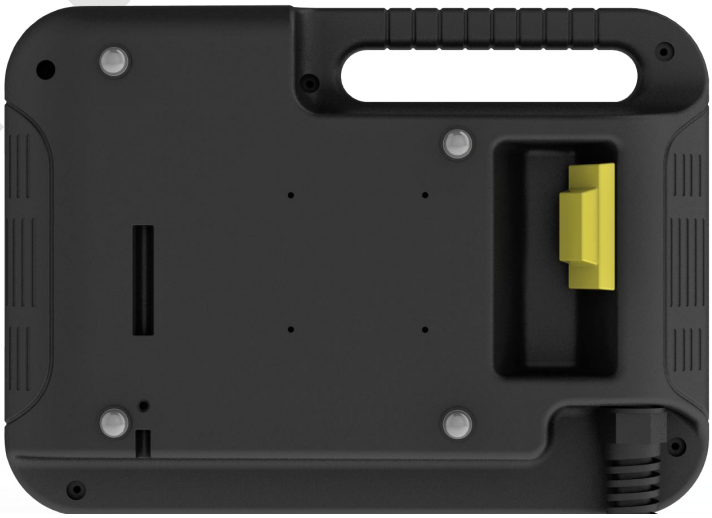


FIG. 4-3 Teach Pendant back side

4.2 Teach Pendant control switch function description

Diagram	Switch name	Function description	
	Mode switch	To control Robot at which mode	TEACH: manual mode
			PLAY: Auto mode
			REMOTE: Remote mode
	Emergency stop	Used when stopping the robot in an emergency	
	Manual enabled	Power on the motor, manual mode effective	
	-negative direction inching	Manual joint movement mode, correspond to the negative directions of axis J1-J6	
		Manual Cartesian movement mode, correspond to the negative directions of X, Y, Z, A, B, and C	
	+ Positive direction inching	Manual joint movement mode, correspond to the positive directions of axis J1-J6	
		Manual Cartesian movement mode, correspond to the positive directions of X, Y, Z, A, B, and C	
	Auto enabled	Power on the motor,, Auto mode effective	
	Start/Checking	Manual mode, press and hold, proceed the existing program.	
		Auto mode, continuous or single-step mode start the program	
	Pause	Manual or auto mode, pause the program running	
	Alarming clear	clear appeared alarm If alarm still exists, afraid cannot clear all alarm	
	Increase velocity	Manual/Auto/remote mode speed up	
	Reduce velocity	Manual/Auto/Remote mode speed cut	

Chapter 5 Software interface using introduction

5.1 Interface area introduction

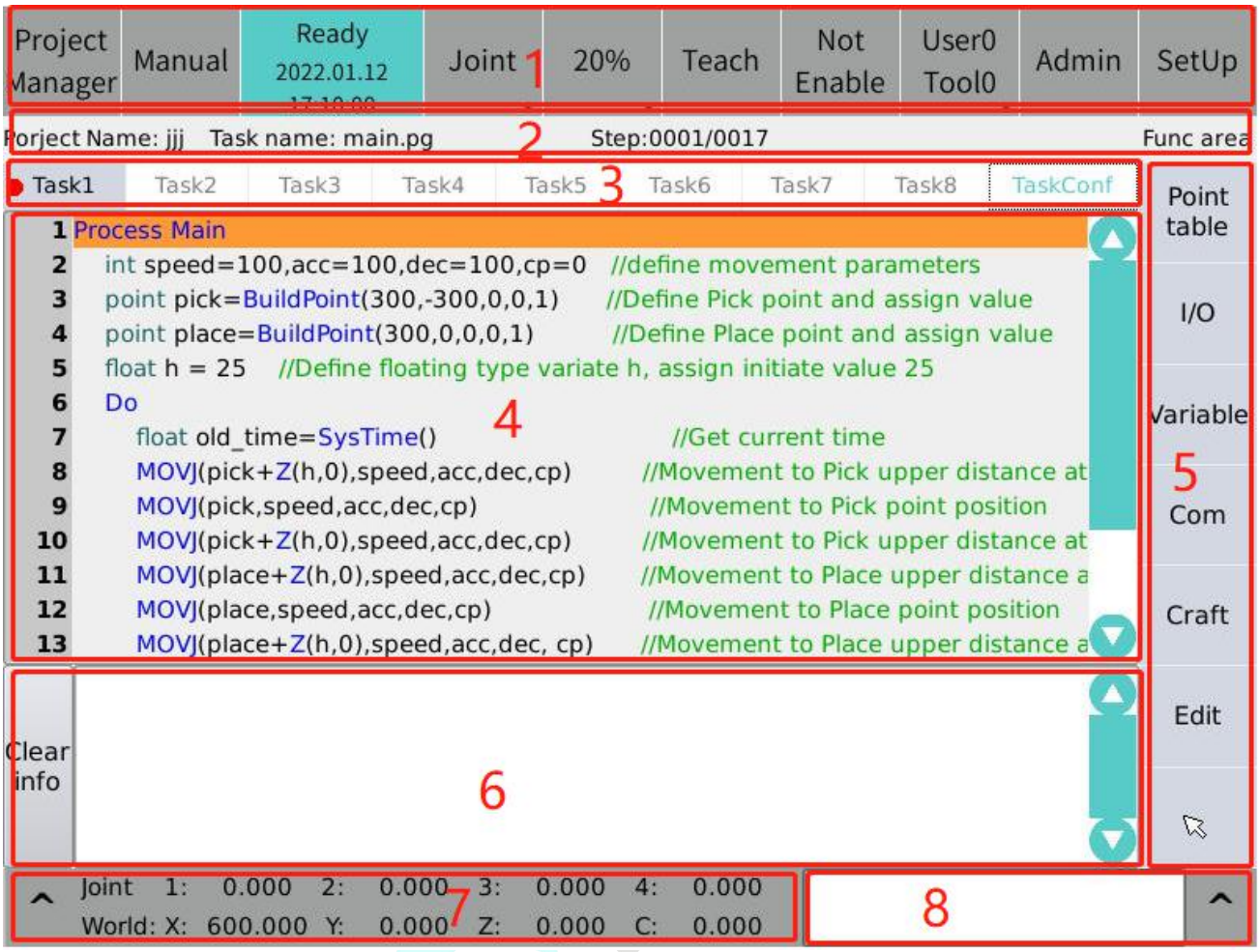


FIG. 5-1 Main interface

Area	Type	Function description			
1	System menu	Project Manager	Button, the entrance of project management (manual)	Manual	Status light, system working mode
			Switch, new create, delete, change, import/export etc.		Manual/Auto/remote
		Ready 2022.01.12 17:10:08	Status light, system working status	Joint	Switch to jog mode under manual mode
			Ready, running, pause, emergency stop, alarming		Joint, world, user, tool
		20%	Button, manual 0.01mm/degree~100	Teach	Button, record the location to the appointed place
			Auto 1~100		refer to 5.2.5.2. point position modification

		Not Enable	Status light, motor power on or not	User0 Tool0	Button, manual user/tool shift
			Not Enable Motor not power on Enable motor power on		user ID: 0~15 Tool ID: 0~9
		Admin	Button, switch right of using	SetUp	Button, setting entrance (manual)
			Operator, technician, administrator		Coordinate, parameter, upgrade, backup etc.
2	Navigation	Indicate the current interface content or file information			
3	Parallel task	Task1	single click to check the program content	TaskConf	Button, multiple-task configuration entrance (manual)
			stop/pause-red, running-green		refer to 5.2.3 project task configuration
		Task2	white frame means not assigned task		
4	Program display	1 Proces 2 float sp 3 speed=	Program content: Grey bar stands for row number, Orange strip means current execution, blue font is variable type definition, red font is grammar/syntax		
			⚠ Manual mode non-edit status, row number can select but cannot modify program		
			Auto mode cannot select row, row number refresh based on actual execution		
5	Project menu	Point table	Point Position management entrance	I/O	I/O monitor entrance
			Only for viewing under auto mode refer to 5.2.5 point position table		⚠ input: The simulation on: selection on off input: The simulation off: Per actual received output: manual: user select on off output: auto: system control
		Variable	Modbus instant communication variable	Com	Communication configuration entrance

		Variable	checking entrance	Communication	auto mode can review only
		Craft	closed state	Edit	Program edit entrance Refer to 5.3.1 program edit
		Continuity	Single-step/continuous switch		
6	Printing output	Printing program Print( ) function output result		Clear info	Empty printing info auto clear (scroll up 100 lines covered)
7	real-time monitor	Joint	display the real-time joint value	World:	display the real-time world/user coordinate value
		Single click	Tune/call out axis position, motor, encoder, IO etc. monitoring information		
8	The man-machine communication	Real-time output of man-machine communication information			

Tips	Figure 5-1 shows the interface in manual mode. In automatic mode, the interface is the same as that in manual mode, but the corresponding buttons are disabled or the modification is restricted for query only

5.2 Project management

5.2.1 Project files management

FIG. 5-2-1 Project management main page

Button icon	Description
NewPro	New create project, set project name per the new create dialogue box
Load	Loading project, load and reset project per the selected one
Copy	Copy the selected and indicate to input the duplicate copy's name.
ReName	Rename the selected project name, project under running cannot change
Delete	Delete the selected project, project under running cannot delete
ViewContent	Open project content interface, details refer to 5.2.2
Local<->USB	Open the project import/export interface, details refer to 5.2.4
Refresh	Refresh all projects
Close	Exit project management, if click project management can exit too.

5.2.2 Project contents interface

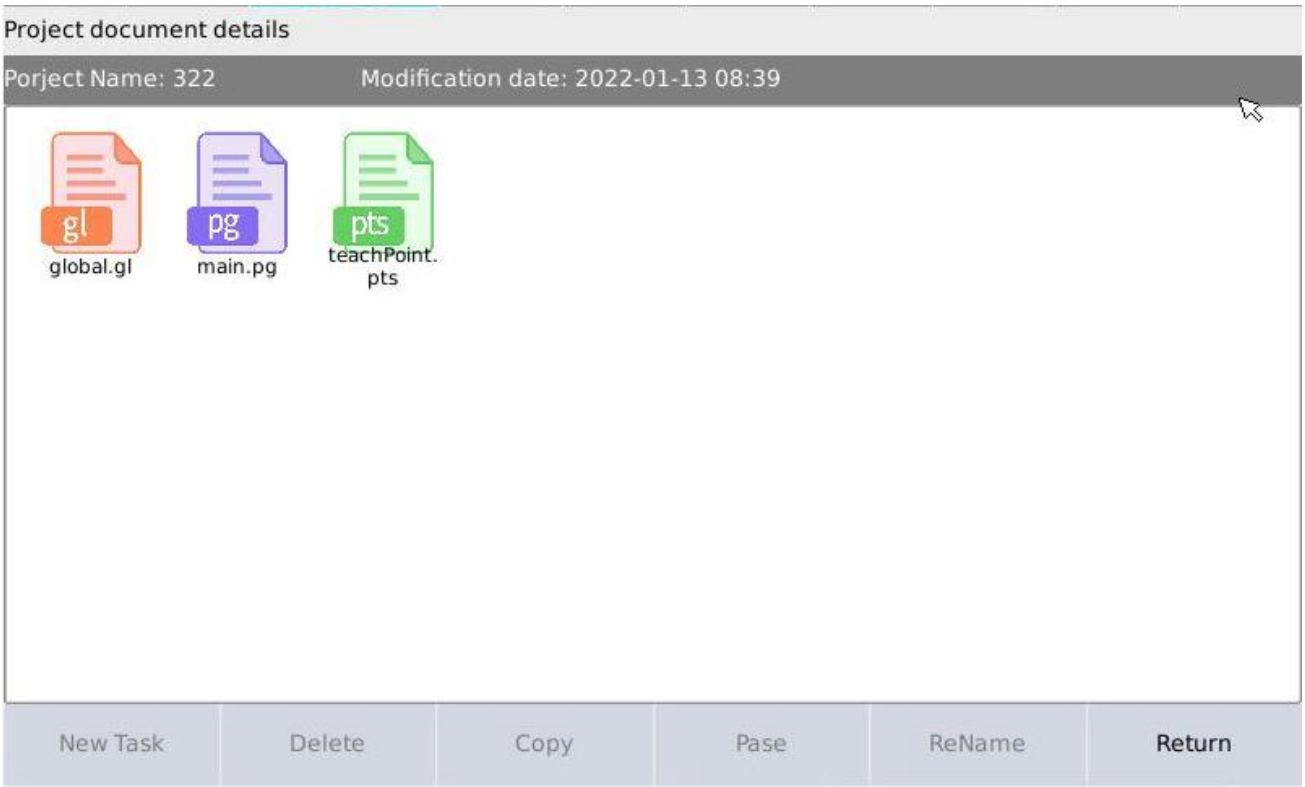


FIG. 5-2-2 Project contents interface

Button icon	Description
New Task	New create foreground or background program on the current project
Delete	Delete the selected contents. Some situation cannot delete, practical tips prevail
Copy	Copy the selected program or other contents into “clipboard”
Pase	Paste “clipboard” contents, this operation can across projects
ReName	Rename the selected file. Some situation cannot rename, practical tips prevail
Return	Exit the current interface, return to the previous “project management”

Attention	After changing the project program, manually click “Load Project” to load the project again. Some operations in this column need to be performed on the task management interface

5.2.3 Project task configuration

5.2.3.1 Project task configuration main page(multiple-task configurator)

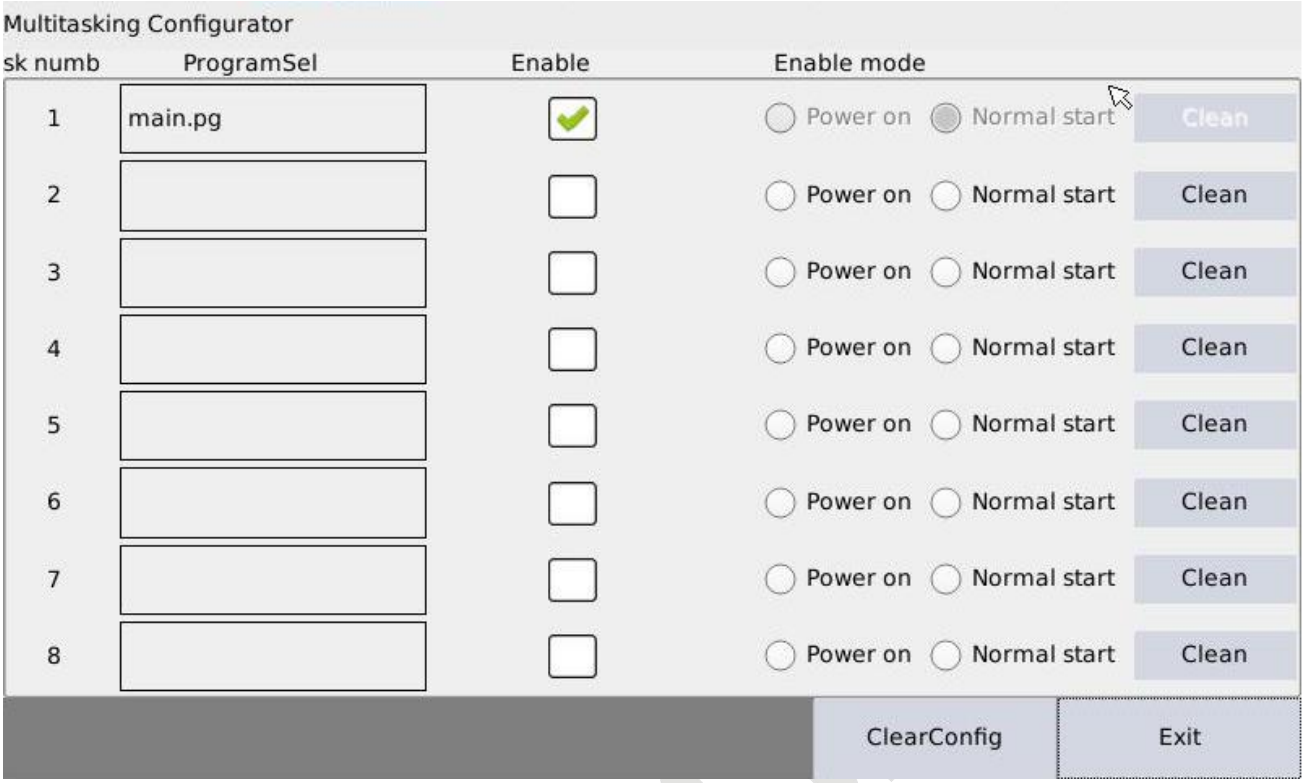


FIG. 5-2-3-1 project task configuration interface

Button icon	Description
main.pg	Start program selecting dialog box, details refer to 5.2.3.2
✔	whether or not to use the left side program
Power on	After clicking, the machine is on and into the system, and left side program initiate; Or common boot mode
Normal start	After clicking, left side program initiate along with main task button; Or boot mode
Clean	Clear left side configuration program, and the program itself cannot delete
ClearConfig	Clear all configuration
Exit	Exit and back to the manual interface main page; before exiting, it reminds if need to save the current configuration or not, it comes into effective after confirmed.

5.2.3.2 Program select dialog box

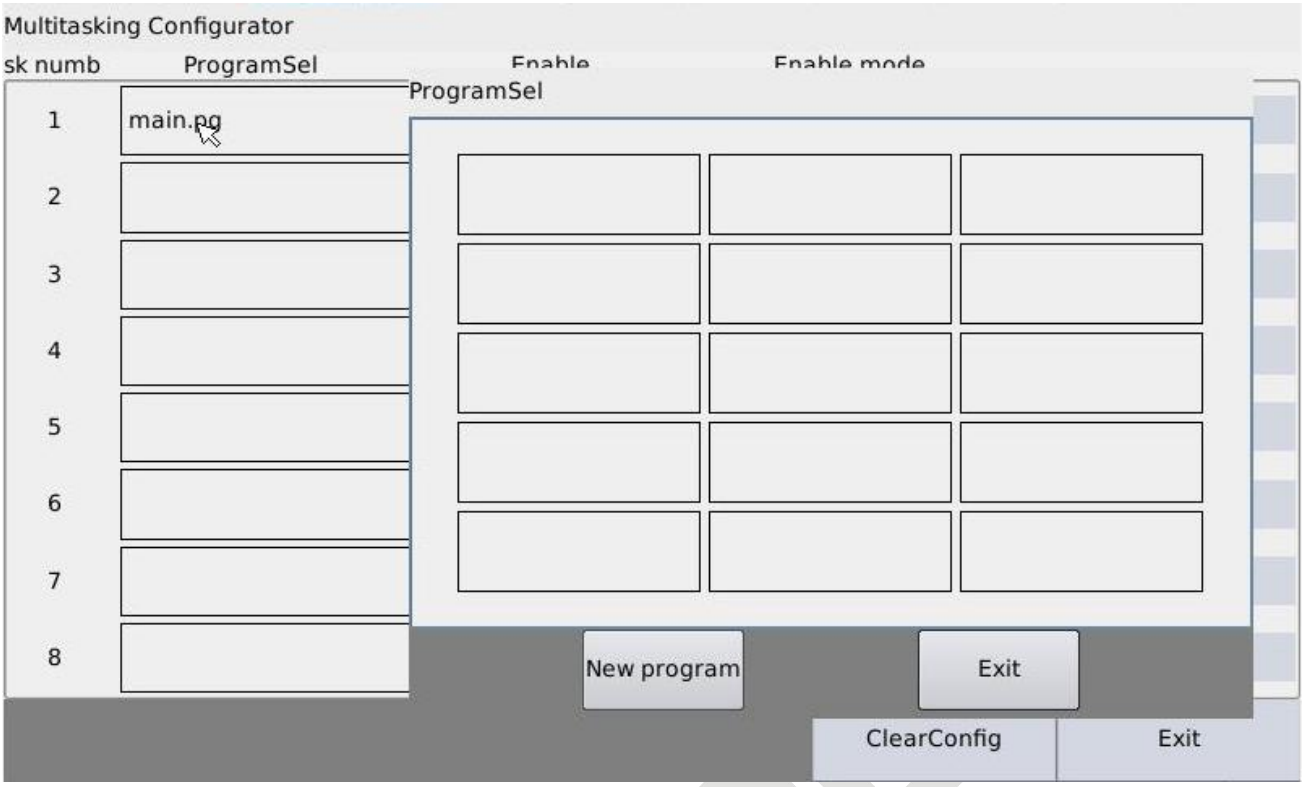
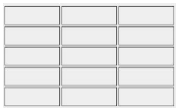


FIG. 5-2-3-2 Program select dialog box

Button icon	Description
	Being existed main program/background program displayed in the list accordingly, select the corresponding name.
New program	If no background program to be selected, but user want to configure task, click this.
Exit	Exit the current interface, back to the previous.

Tips	Main program suffix ".pg", background program suffix ".pgb" One project file system may contain many main program and background program, but after allocation or configuration, max only 1 main program and other 7 background program into effective. Details refer to 6.1.1 project files structure

5.2.4 Program import / export interface (USB Driver)

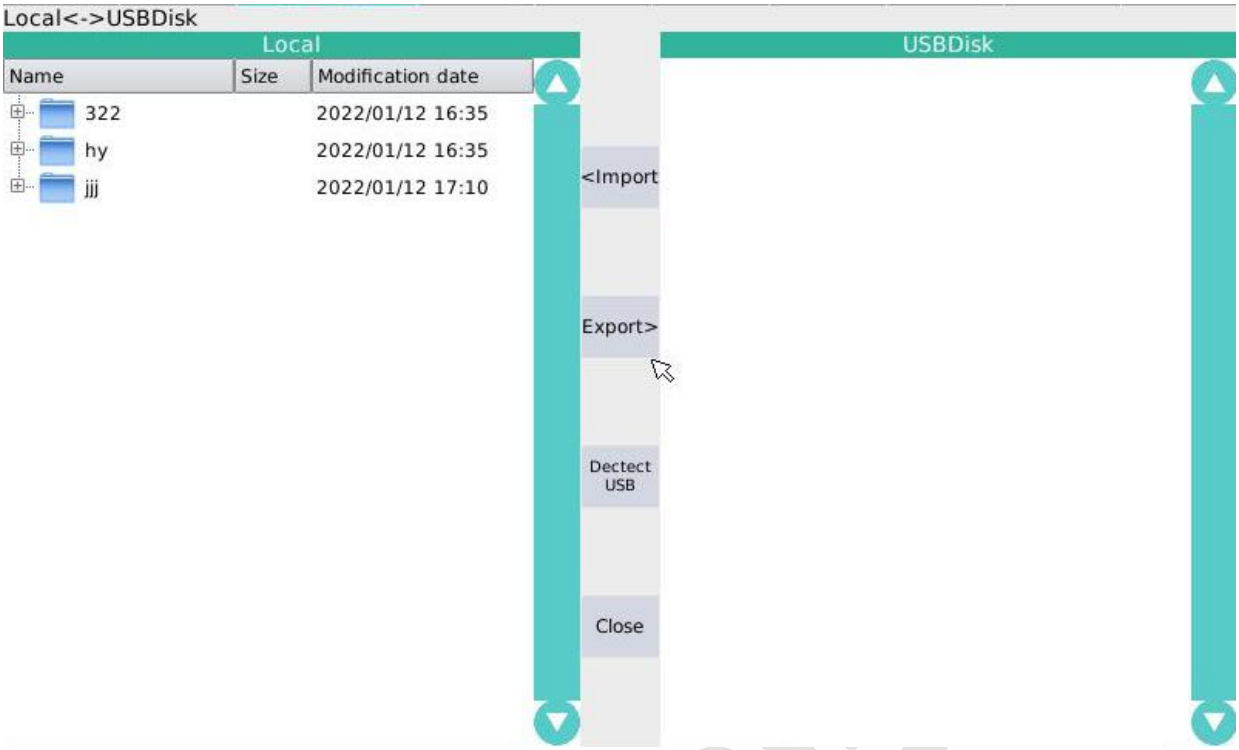


FIG. 5-2-4 Project import/export interface

Button icon	Description
<Import	Import USB driver selected project into the system(local)
Export>	Export the system (local) selected project into USB driver
Dectect USB	Manually check if USB driver exists, it takes about 1-2 seconds in the case of normal connection.
Close	Exit current interface back to the main page of the project management

Tips	If not detect out the outer USB, click button “USB checking”, if still not found USB, re-start Teach Pendant or the system. When system detect out USB, corresponding tips would indicate on the bottom right corner.
Tips	Importing project, when it’s multiple type, need to re-configure multiple task, refer to 5.2.3
Alert	**Do not try to import non-project folders on a USB flash drive as project files, unknown errors may occur or crash system file.**

5.2.5 Point Table

5.2.5.1 Point table main page

FIG. 5-2-5-1 Point table main interface

Button icon	Description
Point	Display current executed point serial number
P00001	Target position number, selected and click [To here], [Save], [Clear]in program reference, and can use Pn(2). (Only read operation in program, cannot edit) . For example, disx=Pn(2).x or MOVJ(Pn(2),100,100,100,0)
Remark Click here to add o...	Display or modify/cancel selected point position name, after selected, expand the keyboard, allow to enter max.10 pcs characters
Import.csv	Import USB driver teachpoint.csv file into the system project point position table
Export.csv	Export project point position table and save it as teachpoint.csv name into the USB driver
MOVJ	MOVJ/MOVL/JUMP 3 movement mode choices available
LimH 200.000	Selected JUMP movement mode into effective, each time when re-enter the point position table, Axis J3 upper limited position will be used as the initial value. You can change it manually before motion,

	but the value should not be lower than the current position or the target position
MovePoint	In the form of joint interpolation (MOVJ/MOVL/JUMP optional) movement reach the target position, need to manually enabled ON state.
Save	Save the teach or modified point position value to the system.
Clear	Clear current setup position.
Pre Page	Back to the previous page, total 10 point positions on each page
1 /1000	When input numbers quickly go to the multiples of 10 page.
Next Page	Turn to next page, total 10 point positions on each page
Exit	Exit the current interface back to the main page

5.2.5.2 Point Position modification

5.2.5.2.1 teach point table

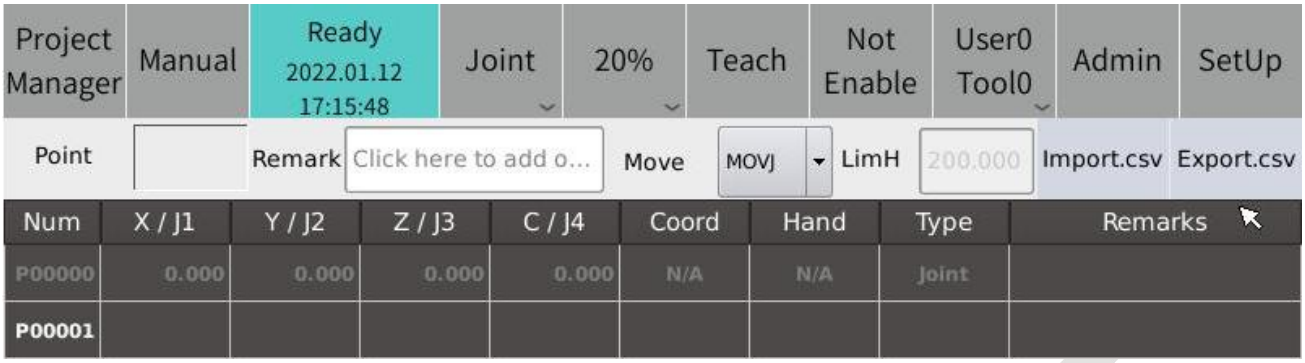


FIG. 5-2-5-2-1 Point position table

Point teach method:

1 Selected S/N, Orange highlight on the existing row

2 Click system menu of teaching button, coordinate values stored based on the current movement mode, user coordinates, and other states

Current state	Description
Joint	Save as each axis joint data, "type" show JOINT automatically
World	Save Cartesian (XYZ...) data, "type" show Cartesian automatically
Tool	Save Cartesian (XYZ...) data, "type" show Cartesian automatically
User	Save Cartesian (XYZ...) data, "type" show Cartesian automatically
User0 Tool0	When saving data using Cartesian, coordinate serial number selected by the current user would be recorded into the coordinate system meanwhile.
Num	Range of 0-1000, quoted Pn (numbers) in program
Hand	Based on current posture generated automatically, Cartesian coordinate can be modified left or right by clicking

3 Select Save, end teaching work

Skills	When in teaching location, it may cause slight movement if the motor is not enabled, the position can be locked if the motor is enabled manually.

5.2.5.2.2 Modify point position data

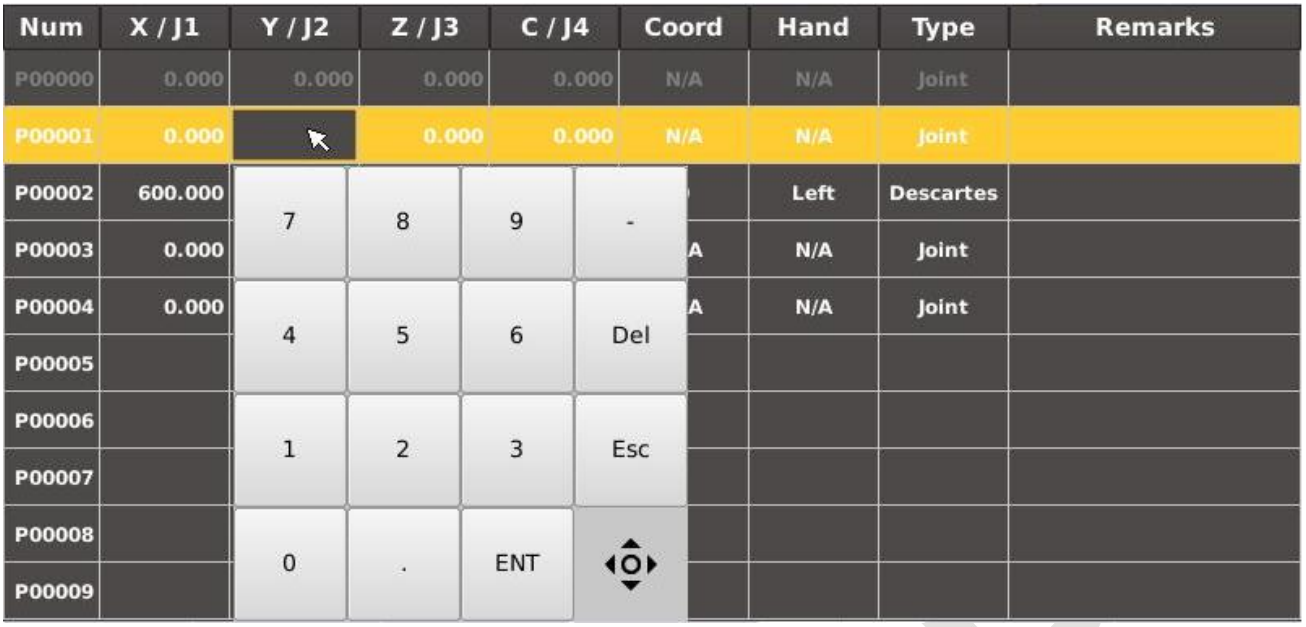


FIG. 5-2-5-2-2 modify point position data

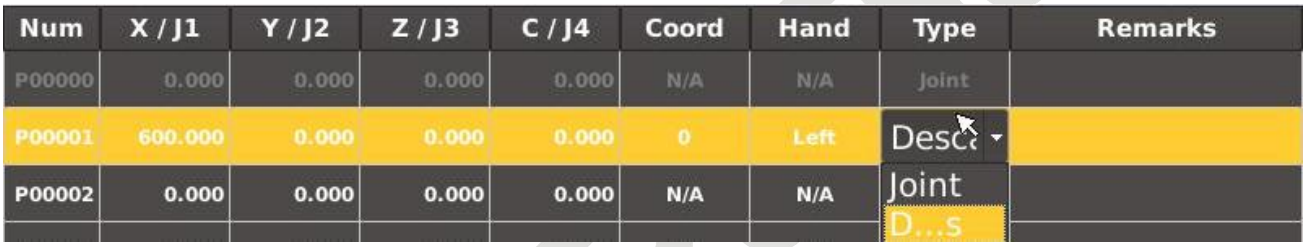


FIG. 5-2-5-2-3 modify coordinate type

Tips	Not all Cartesian coordinate values can be reached. Please check whether each joint is within the soft limit after modifying the values. Refer to the Robot operation range diagram. Wrong data setup, the point position cannot be reached.
Tips	The point position quoted in the program is “Pn(s/n)”. Each project has a point table file and can only access its own point table file.
Alert	P00000 cannot modify, it only for reach here through others. It represents Robot mechanical zero. Mechanical zero indicates that the robot arm is completely stretched. When in motion, please fully consider whether there is obstacle around.

5.2.6 I/O Monitor panel

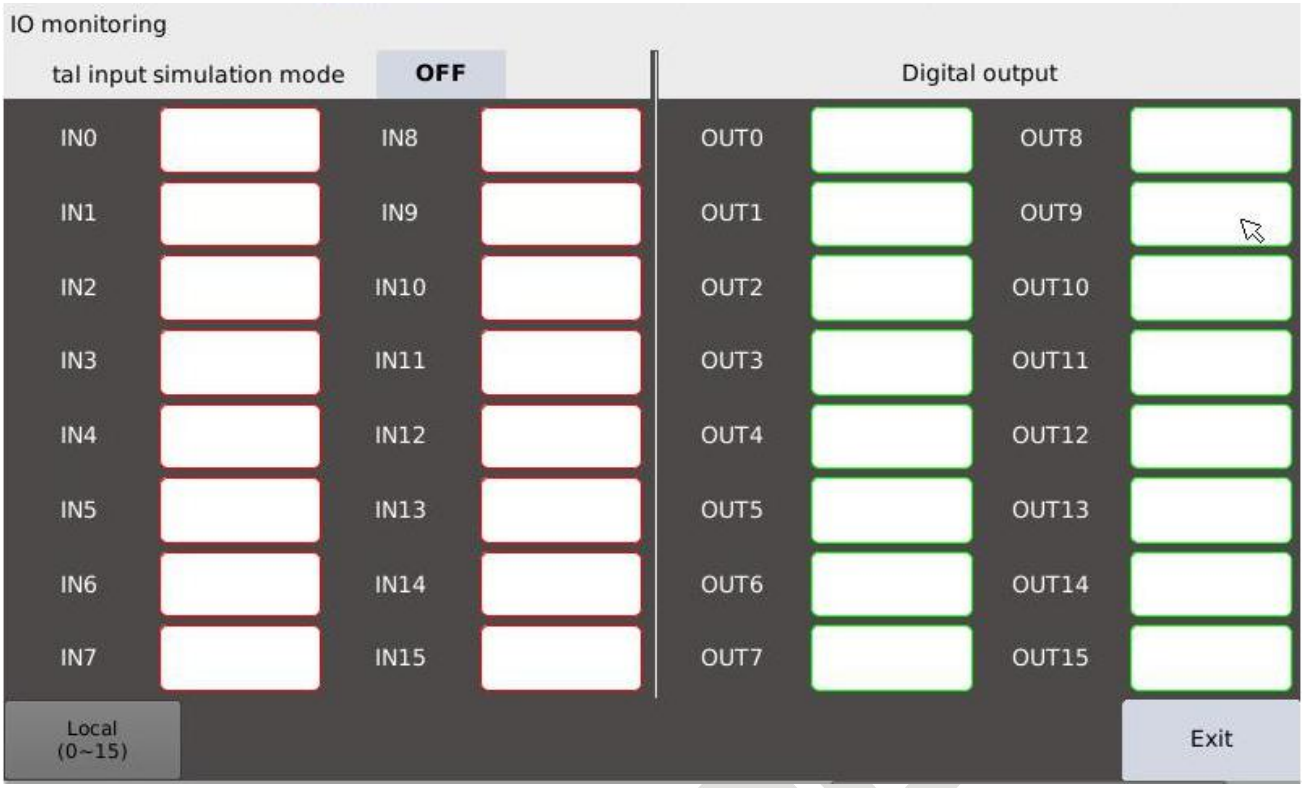


FIG. 5-2-6 I/O Monitor panel

Button icon	Description	
OFF	Simulation function off (default)	
ON	Simulation function ON	Simulation is for input signal only
		Start on manual or auto mode
		Not receive real input signal anymore once start on, only use the simulated input signal.
	input	When the display is red, it means that the input signal is checked Under the simulation function on, click the button simulation input switch, and when the working mode is switched to automatic, it is still in the simulation state.
	Output	Manual mode, click button to control output switch ON/OFF. When the working mode switch to auto mode, the clicked state will not be changed. If equipped with emergency stop reset function, the crresponding signal reset when the emergency signal triggered.
		Auto or Remote mode: click invalid, only controlled by program.
Local (0~15)	Selected local output group, 0~15 or 16~31, extended I/O would show right side if it is applied.	
Exit	Exit current interface pad.	

Skills	Simulation input can solve the problem of no actual input signal while checking program execution logic issue.

5.2.7 Variable (Interface)

Global variable introduction

This interface referred variable as the system defined Modbus real-time communication systematic overall variable, int 32 and float 32 each 100 units respectively (non-data-holding on power-off) GInt32 and Gfloat32 each 50 units (data-holding on power-off). For a detailed introduction to variable classification, please refer to 6.1.2 variable type and scope.

Global variable - power down does not hold

	+0	+1	+2	+3	+4
I0	0	0	0	0	0
I5	0	0	0	0	0
I10	0	0	0	0	0
I15	0	0	0	0	0
I20	0	0	0	0	0
I25	0	0	0	0	0
I30	0	0	0	0	0
I35	0	0	0	0	0
I40	0	0	0	0	0
I45	0	0	0	0	0

Int Type | Float Type | GInt Type | GFloat Type | Pre Page | Next Page | Exit

FIG. 5-2-7 variable monitor interface

5.2.7.1 Interface description

The above FIG.+0~4 means the added unit numbers on the left side address, for example: If want to check I39 current value, first go to find I35, then +4, thus the location of the vertical and horizontal intersections stands for the value of I39.

If input manually, click corresponding intersections, input the corresponding value on the expanded keyboard to get it done and sent the data immediately;

5.2.7.2 Modbus custom variable read and write operate

When using program instruction to operate address reading and writing, refer to 6.2.7 communication instruction MIRead, MFRead, MIWrite, MFWrite etc. to read and write operation per address serial number, unit read write can quote I/F/GI/GF connect S/N.

5.2.7.2.1 Write operation

Case 1:

```
Set I39 value as 10
I39=10 also can write MIWrite(39,10)      // non-data-holding on power-off
GI0 = 1 also can write: GIWrite(0,1)       // data-holding on power-off
```

Case 2:

```
Reset I0~I99 as 0
int i
For i=0 To 99
        MIWrite( i , 0 )
Next
```

5.2.7.2.2 Read operation

Case 1:

```
wait I20 value as 15
Function Wait(int addr,int val)// Waiting variable I equals to some value's function
        Print("waiting I",addr,"=",val)//Printing the message being waited.
        Do While MIRead(addr) != val
                Delay(5)
        Loop
FunctionEnd
Wait( 20 , 15 ) //Call for custom function Waite ()
```

Case 2:

```
Judge I20 whether or not equals to 15
If I20 == 15 Then     //Judge now I20 whether or not equals to 15
        ...
Else
        ...
EndIf
```

Case 3:

Based on the value sent by master station to do the XYZC off-set, the master station send XYZC off-set value to F40~F43 respectively

MOVJ(standby,speed,acc,dec,cp)

Break() // Interrupt continuous motion, prevent reading wrong time incorrect value

MOVJ(labelingPos+X(F40,0)+Y(F41,0)+Z(F42,0)+C(F43,0),speed,acc,dec,cp)

Tips	**If user not using Modbus, these addresses can be used also as global variable.**

5.2.8 Communication

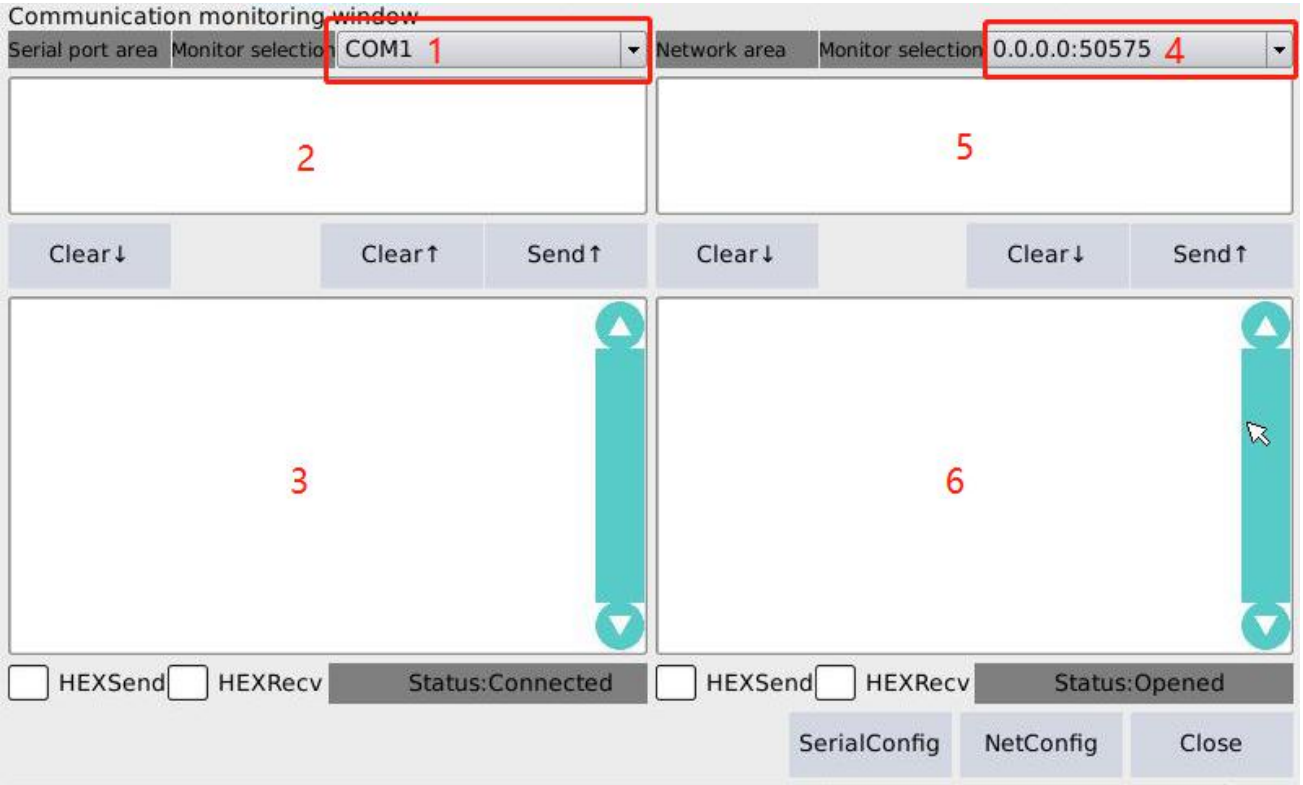


FIG. 5-2-8-1 Communication monitor interface

Button icon	Description
1	Serial port monitor selection, COM1/COM2
2	Serial port sending area, click and expand keyboard, as FIG.5-2-8-2 communication sending interface
3	Serial port receive send data monitor
4	Network port monitor select
5	Network port sending area, click and expand keyboard
6	Network port receive send data monitor
HEXSend	Tick and send in HEX format
HEXRecv	Tick and receive in HEX format
SerialConfig	Configuration on channels of serial port
NetConfig	Configuration the channels of network port
Close	Exit the current interface

5.2.8.1 Serial port communication configuration

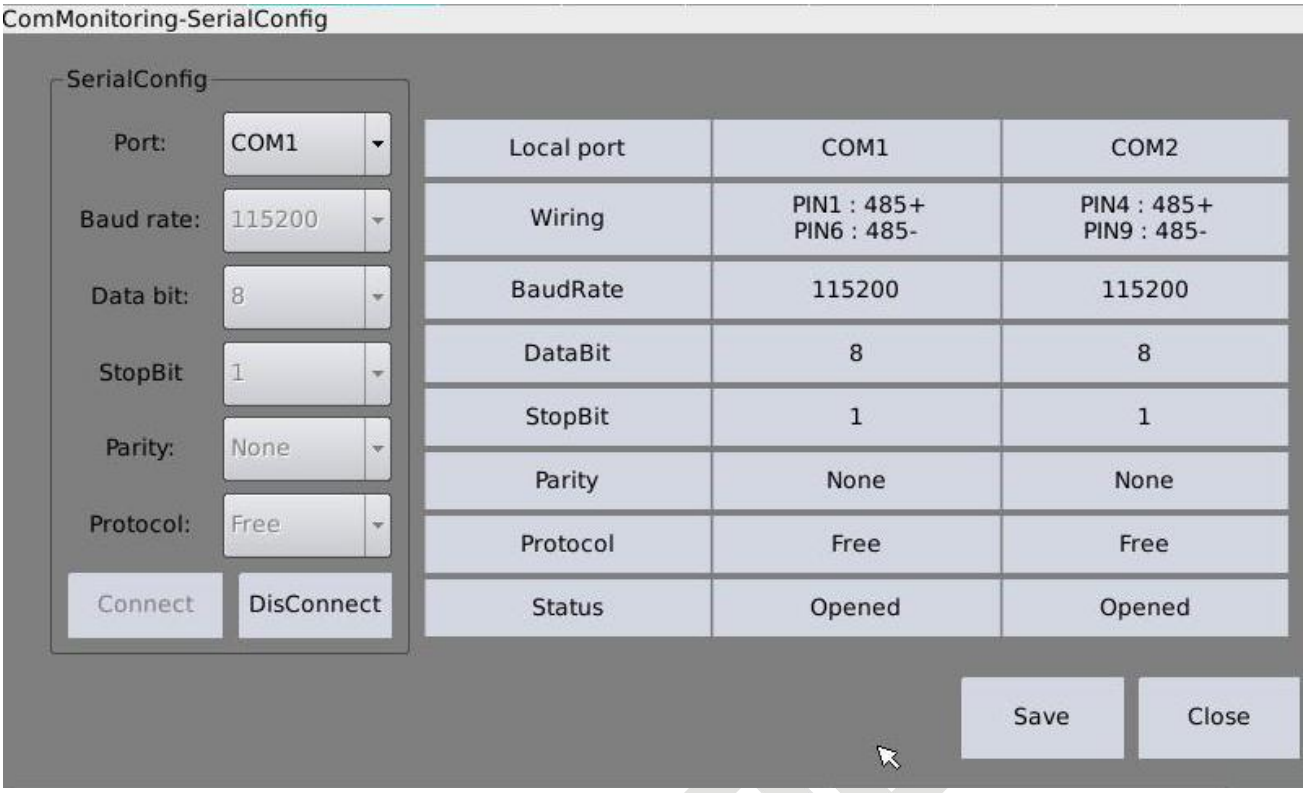


FIG. 5-2-8-1-1 Serial port communication configuration interface

Description

Current hardware support COM1, COM2 available

COM1, when set the configuration as Modbus protocol, COM1 is slave station mode, address is 1 as default.

COM2 only self-defined protocol function at present

Details refer to <Chapter 7, Communication function introduction>.

5.2.8.2 Network communication configuration

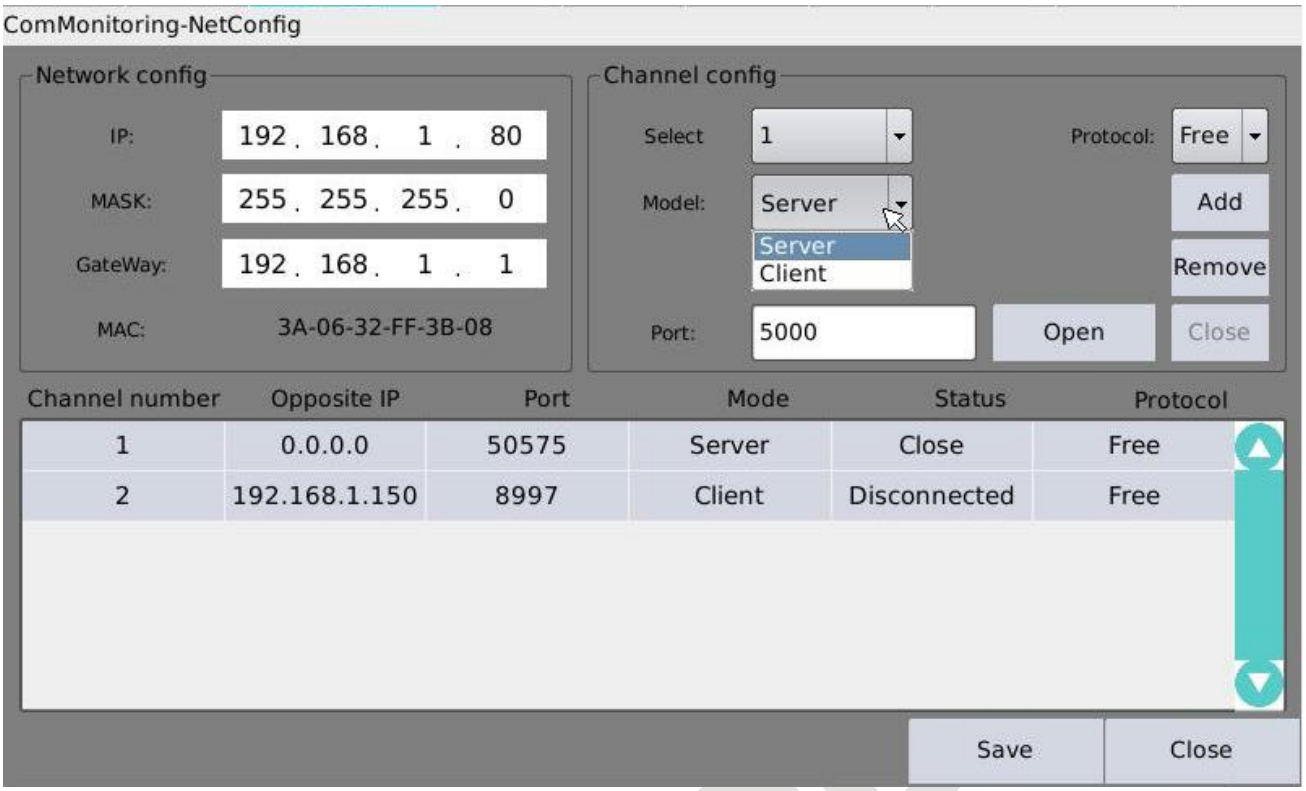


FIG. 5-2-8-2 Network communication configuration interface

Description

The Network config means use controller's local IP at EtherNet port.

Channels number 1-4 in valid. Details refer to <Chapter 7 Communication function introduction>.

5.2.9 Craft

Closed state.

5.2.10 Program edit

5.2.10.1 Program edit main page

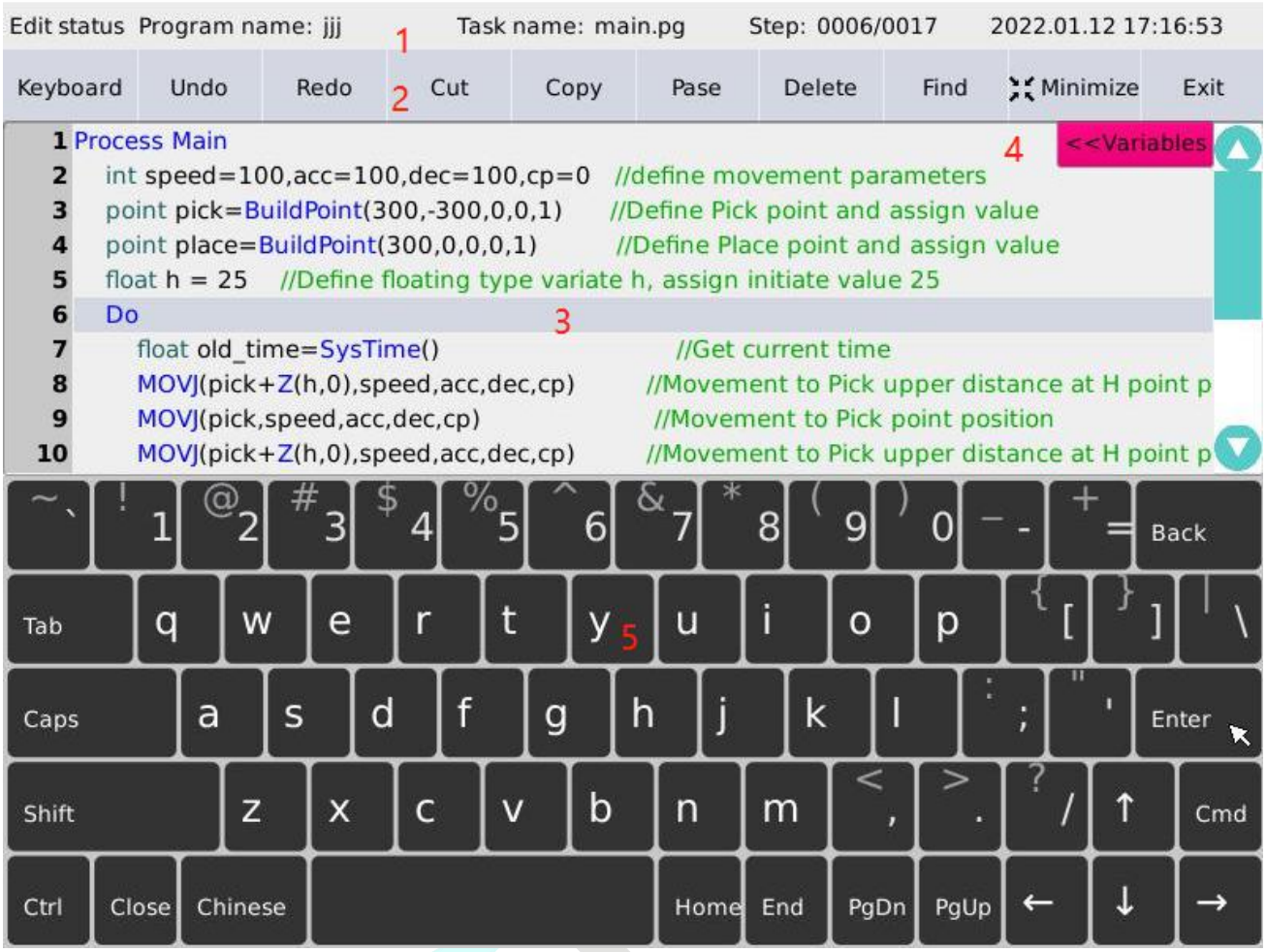


FIG. 5-2-10-1 program edit main interface

Area	Type	Function description
1	Navigating	Display current file info and system time
2	Edit tool	Edit file common used tool buttons, like copy, past, cancel etc.
3	Edit window	Program: Grey column means row number, orange strip stands for current execution, light blue is fonts variable type definition, blue font is grammar/syntax
4	Variables	Expand global variable edit window, details refer to 5.2.10.3 global variable
5	Keyboard	Direct input Character or instruction called for, details refer to 5.2.10.2 instruction keyboard

Skills	Browse to more lines of programs by turning off keyboard. Ctrl+X, C and V on the keyboard can realize cut, copy and paste. Select multiple rows and press Tab to indent them as a whole. Hold down Shift and then press Tab to retract them.

5.2.10.2 Instruction keyboard

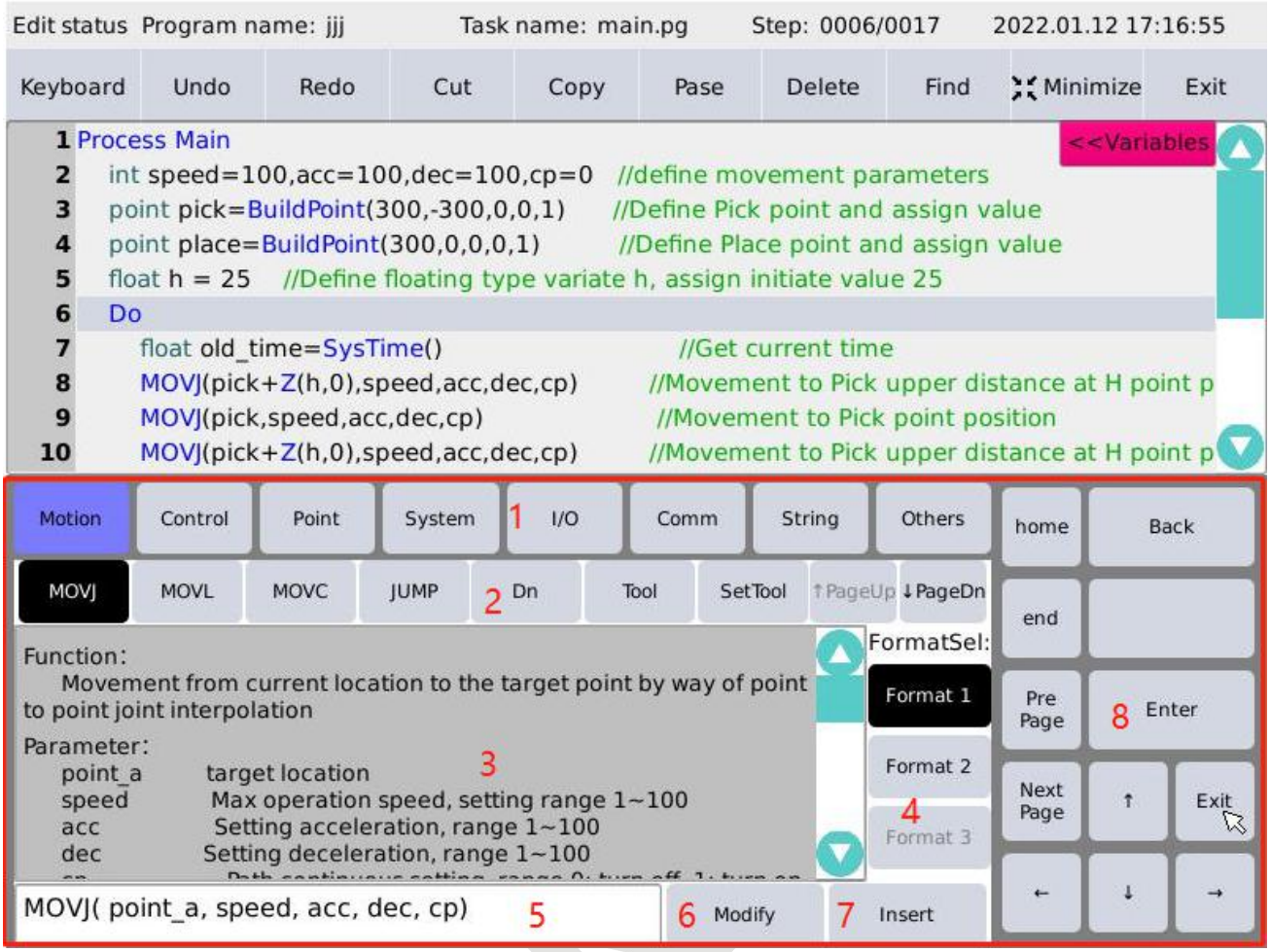


FIG. 5-2-10-2-1 open instruction keyboard interface

Area	Type	Function description
1	Instruction classification	Classify per the purpose of instruction
2	Specific instruction	Display type 1 all instruction, browse by scrolling up and down
3	Instruction introduction	Describe instruction' function
4	Format choice	Same instruction but different parameter realize different function
5	Instruction browsing	Preview of the to be inserted instruction content
6	Parameter modification	Through parameter modify button, open the dialog box of the modify parameter, details refer to 5-2-10-2 instruction parameters modify After modify done, Press Insert to finish the instruction inserting at the specified cursor
7	Insert button	Insert the contents of the preview window into the program

5.2.10.2.1 Instruction parameter modification

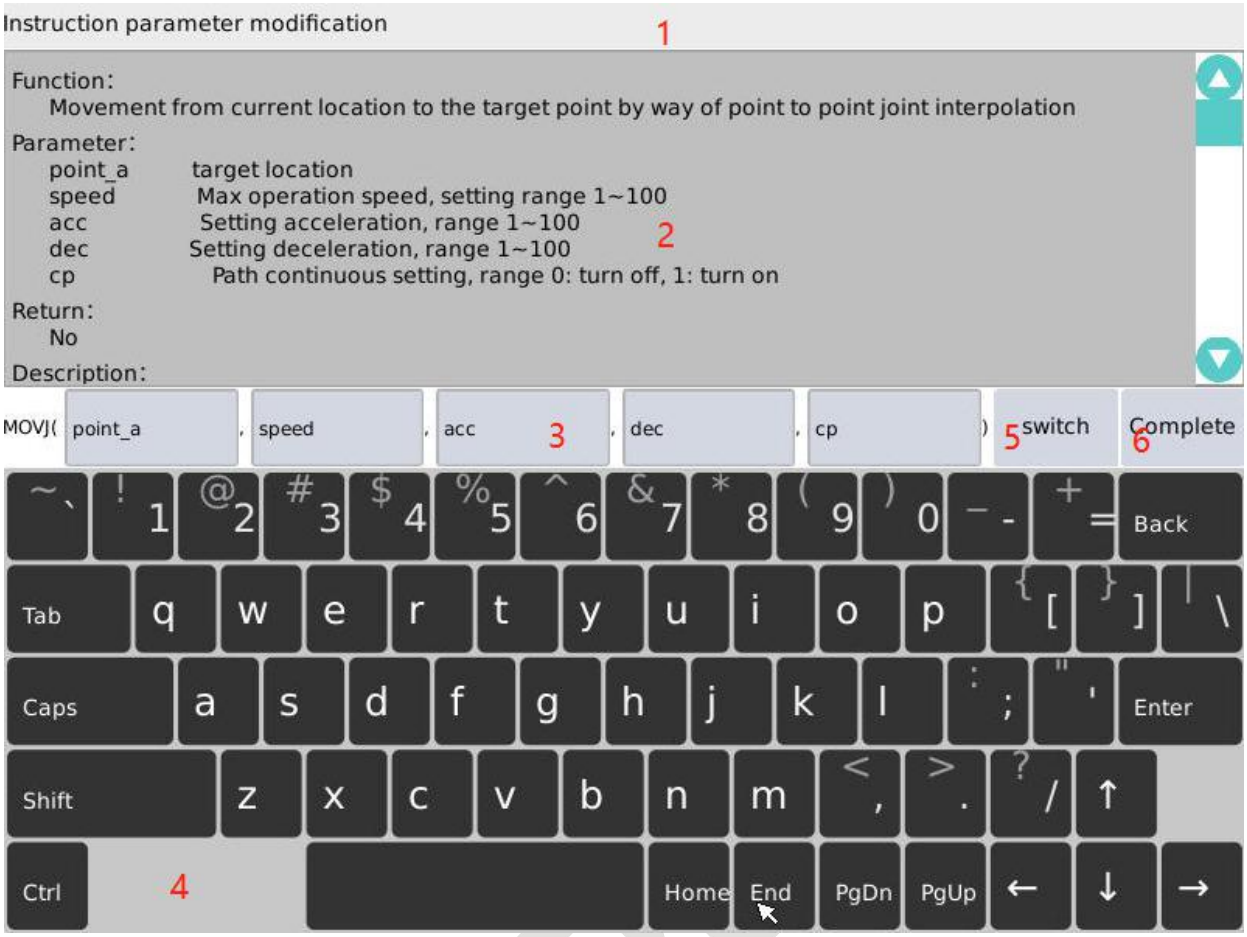


FIG. 5-2-10-2-2 instruction keyboard parameter modify

Area	Type	Function description
1	Navigating	Display current location
2	Instruction introduction	Describe instruction' function
3	Parameter modify dialog box	One by one parameter modify
4	Keyboard	Keyboard used for parameter modification
5	Parameter shift	Shift parameter form, currently only under sports instruction point_a shift to Pn(0)
6	Modification done	After modification done, exit current interface, refresh the modified parameters instruction to preview as show on FIG.5-2-10-2-3

FIG. 5-2-10-2-3 touch modify done to preview

5.2.10.3 Global variables (project level)

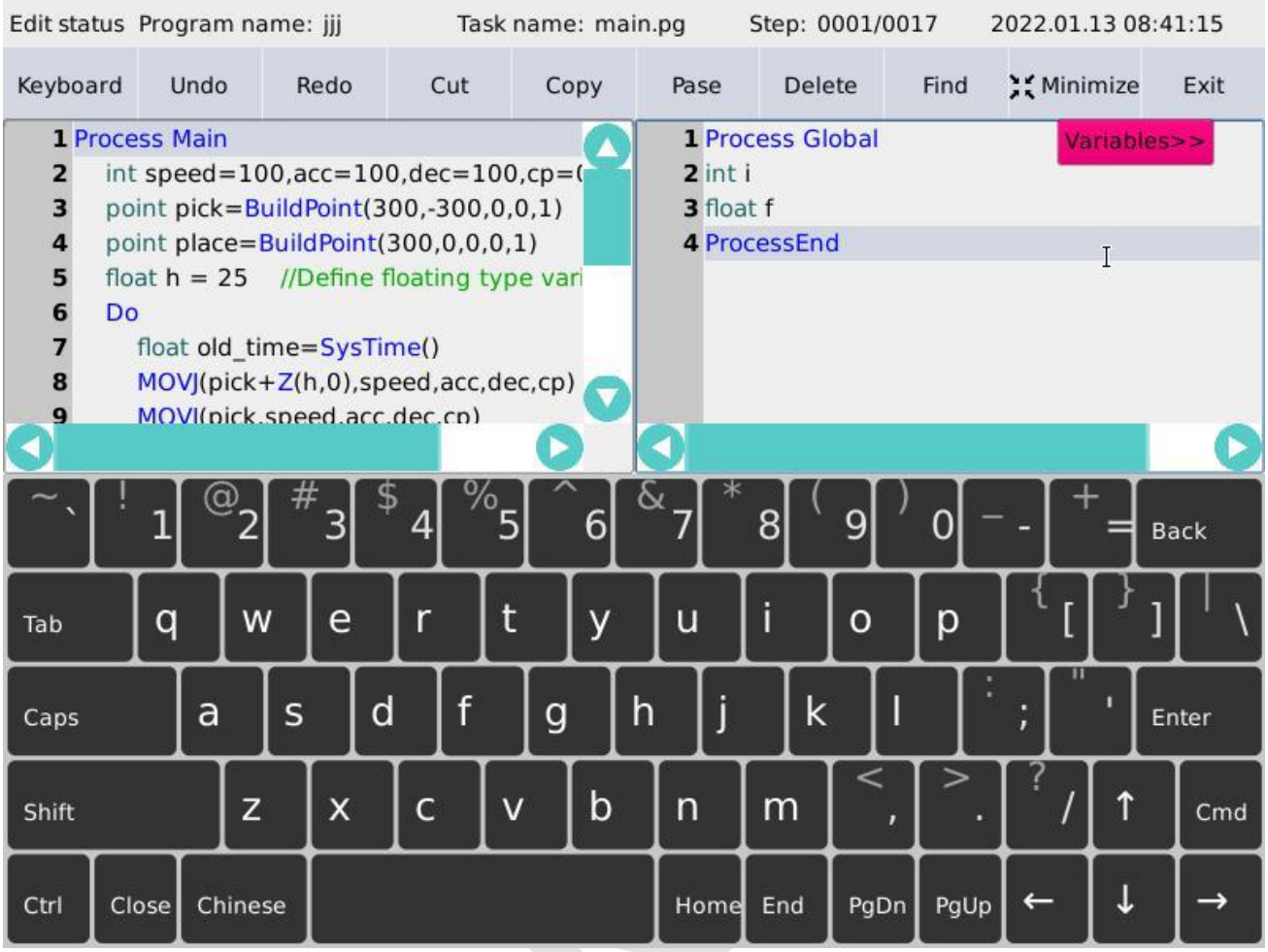


FIG. 5-2-10-3 Global variables

Button icon	Description
	Expand global variables window
	Shut off global variables window

Tips	Global variable (suffx .gl) always begins with “Process Global” keyword, ends with “ProcessEnd”. Between these two keywords declare global variable, which will work at the whole project.
Alert	This window can be expanded in any program edit state, and be careful not to declare variables with the same name. Be careful not to change the keywords of opening and closing when editing the program.

5.3 Setting

Touch [Setup] enter the setting main page, as FIG.5-3

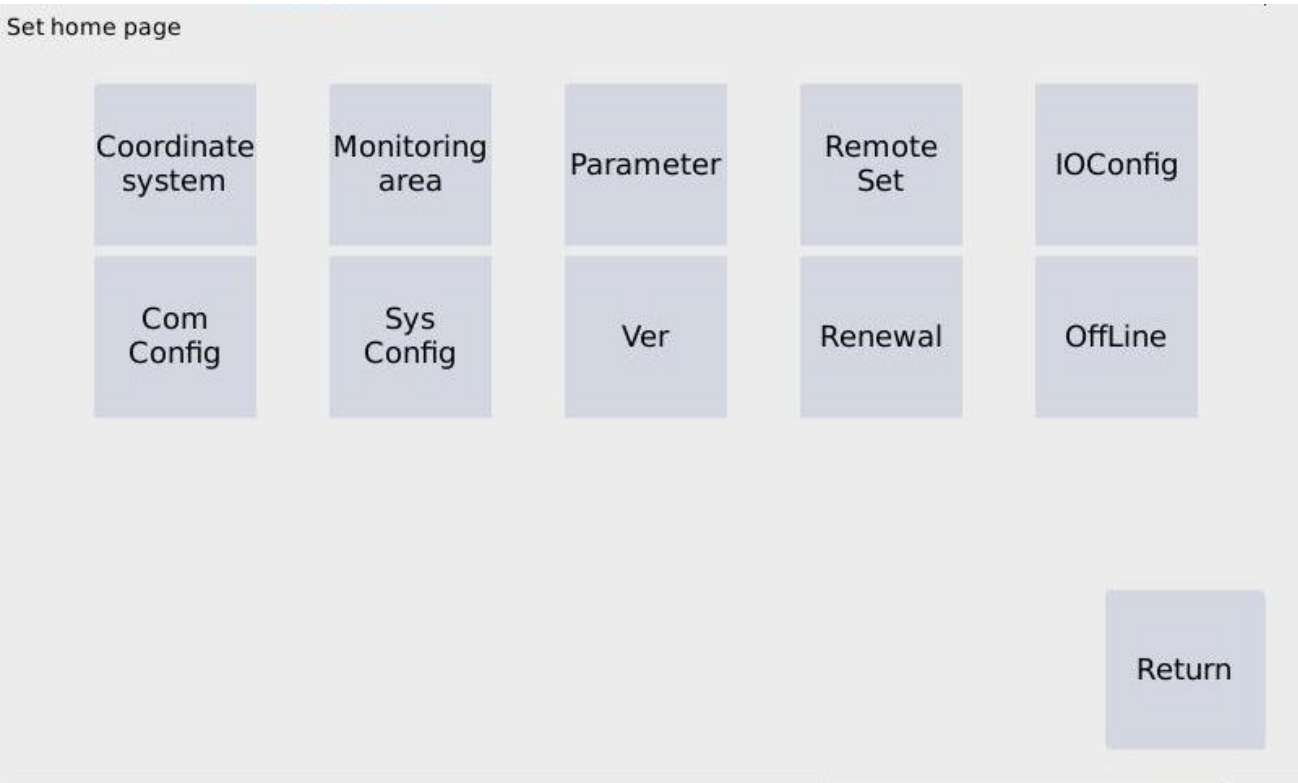


FIG. 5-3 setup main interface

Button icon	Description
Coordinate system	User coordinate system, tool coordinate system setting
Monitoring area	TCP End area monitor: cuboid or circular column obstacle area, monitor area setting
Parameter	SCARA structure parameters (DH parameters), Servo parameters, Operation & control parameters, Advanced parameters
Remote Set	Remote designated launching project, start-stop input &output configuration
IOConfig	Extended IO number setup, IO name configuration, import export configuration
Com Config	Serial port communication and internet communication configuration

Sys Config	System time, key tone buzzer, startup picture setting
Ver	Software version(included controlling system and Teach Pendant) check, Navigator HMI, compile front-end software upgrade
Renewal	Trial period over, remove locked entrance
OffLine	Teach Pendant off line button

5.3.1 Coordinate system

5.3.1.1 User coordinate system

User coordinate system defined on workpiece, also can be called workpiece coordinate system. Within Robot motion allowed range of any location to set any angle of Axis X, Y, Z.

1. User coordinate system interface

FIG. 5-3-1-1 user coordinate system

Button icon	Description
Set	Enter into setting interface, refer to picture 5-3-1-2 user coordinate system teaching
Clean	Clear selected S/N contents
Save	Save modified change
Return	Exit current interface, back to the previous menu

Alert	S/N#0 equals to world coordinate, cannot be modified In this interface can direct modify coordinate system calculated value, unless operator clear understand its meaning, or use system provided setting method to modify

2. User coordinate system teaching setting

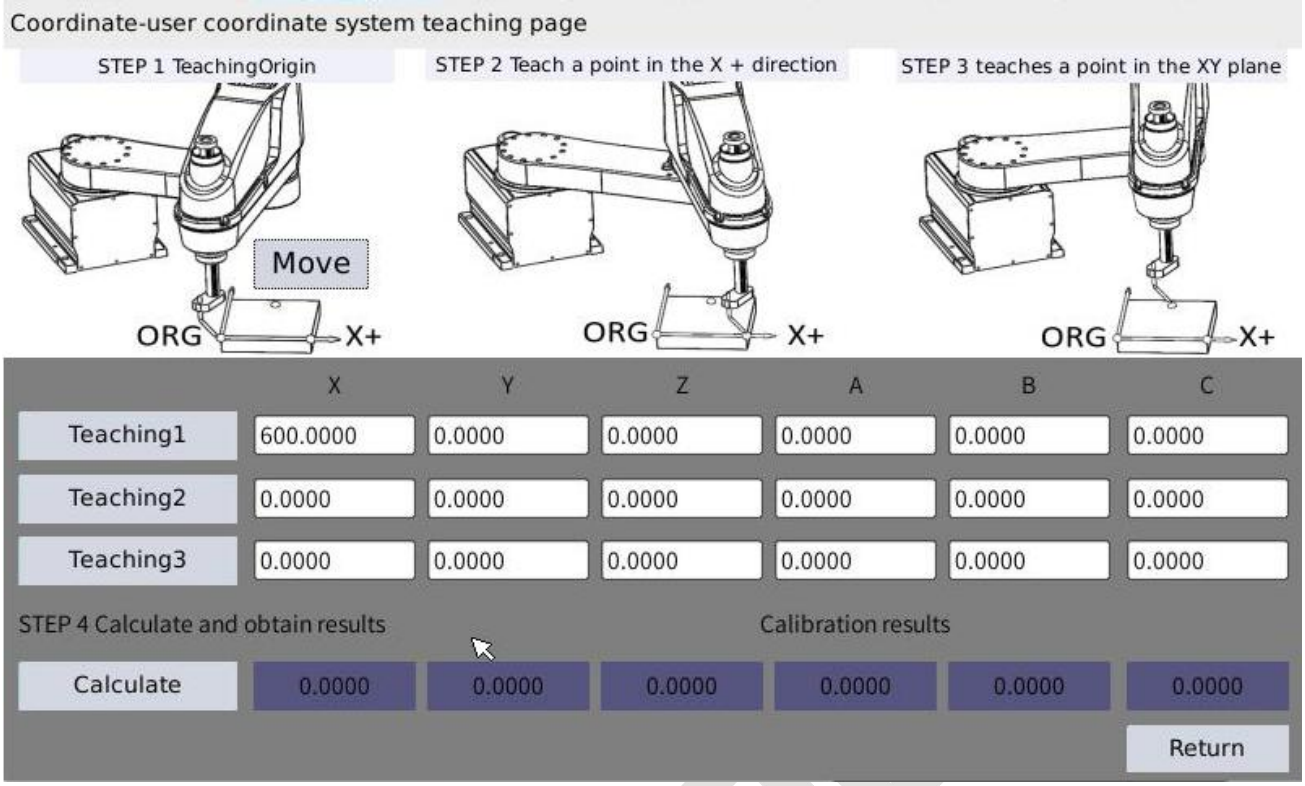


FIG.5-3-1-2 user coordinate system Teaching

Button icon	Description
Teaching1	teaching user coordinate system original point
Teaching2	Teach user coordinate system X + direction point
Teaching3	Teach user coordinate system XY plane Y + direction any point
Calculate	The calibration result calculated by three-point method
Return	Exit current interface, back to the previous menu
Move	After teaching shown beside the chart, move robot by joint interpolation method to this location, after enabled connection it can move

Tips	Before click movement button, turn on manual enabled button. During movement, Robot movement can be interrupted when releasing manual enabled key or press emergency stop button

5.3.1.2 Tool coordinate system

Tool coordinate system installed on the Robot end, original point and direction keeps changing along with its location and angle, and this coordinate system actually is based on basic coordinate system by means of revolving and changing location

1. Tool coordinate system check and modify interface

FIG. 5-3-1-3 Tool coordinate system

Button icon	Description
Set	Enter into setting interface, refer to FIG.5-3-1-4 tool coordinate system teaching.
Clean	Delete selected S/N contents
Save	Save modified change
Return	Exit current interface, back to the previous menu

Alert	S/N#0 equals to default coordinate system, cannot be modified In this interface it can direct modify coordinate system calculated value, unless operator understand its meaning, or use system provided setting method to modify

2. Tool coordinate system teaching

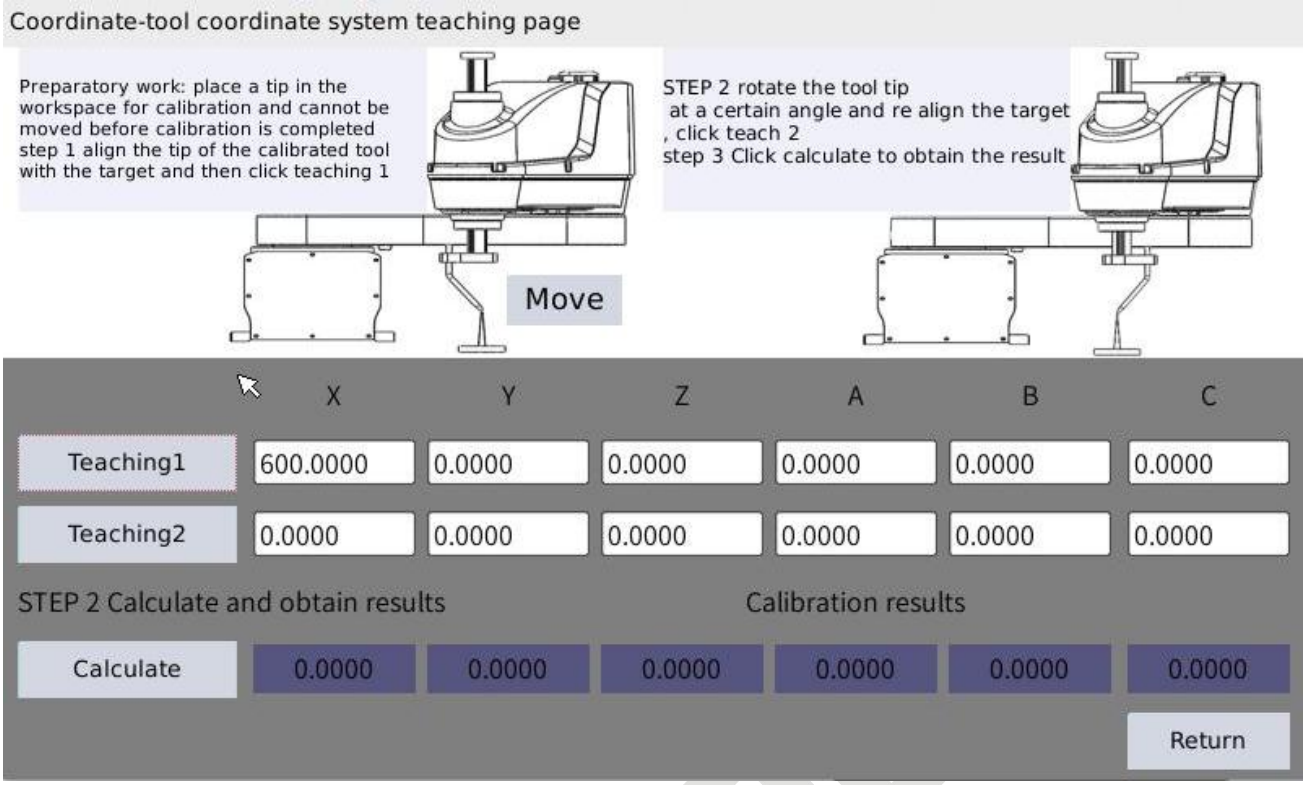


FIG. 5-3-1-4 Tool coordinate system teaching

Button icon	Description
Teaching1	Teaching 1st location
Teaching2	Let tool tip to teaching 2nd location
Calculate	Calculate the calibration result
Return	Exit existing interface, back to the previous menu
Move	Move robot to this point, after enabled key pressed continuously it can move

Tips	Before click Movement button, manual enabled key should be pressed continuously. While during the movement, robot can be interrupted by releasing the manual enabled key or pressing down emergency stop button

5.3.2 Monitor area

Introduction	Description
Function introduction	1. Real time determine whether robot is in the monitor area, if it is, output or shut off corresponding signal and variable per the set value. 2. Set obstacle area, not let Robot enter, independent adjust speed cut boundary width. 3. This machine provides max.8 monitor area. 4. To avoid entering into obstacle area caused non movement, individually set some certain area monitor in valid or not valid. 5. Monitor area shape can be cuboid or cylinder style. 6. When internal use only, can use internal virtual signal, using I/O when exchange external equipment. If need communication, to send out internal virtual signal via communicated way, it's not suggested to do this, as communication reduces real time, it may risk crash; So only real time requirement is not high occasion to use.
Application scenarios	1. Robot has public operation space with other equipment, that's there is interference in the motion space of robot and external equipment 2. There is interference between Robot movement space and external object (fixed). 3. When Robot is turned on, it determines the current location is within which monitor interval, and plan different returning path back to standby point.

5.3.2.1 Monitor interval setting

1. Monitor interval check setting status

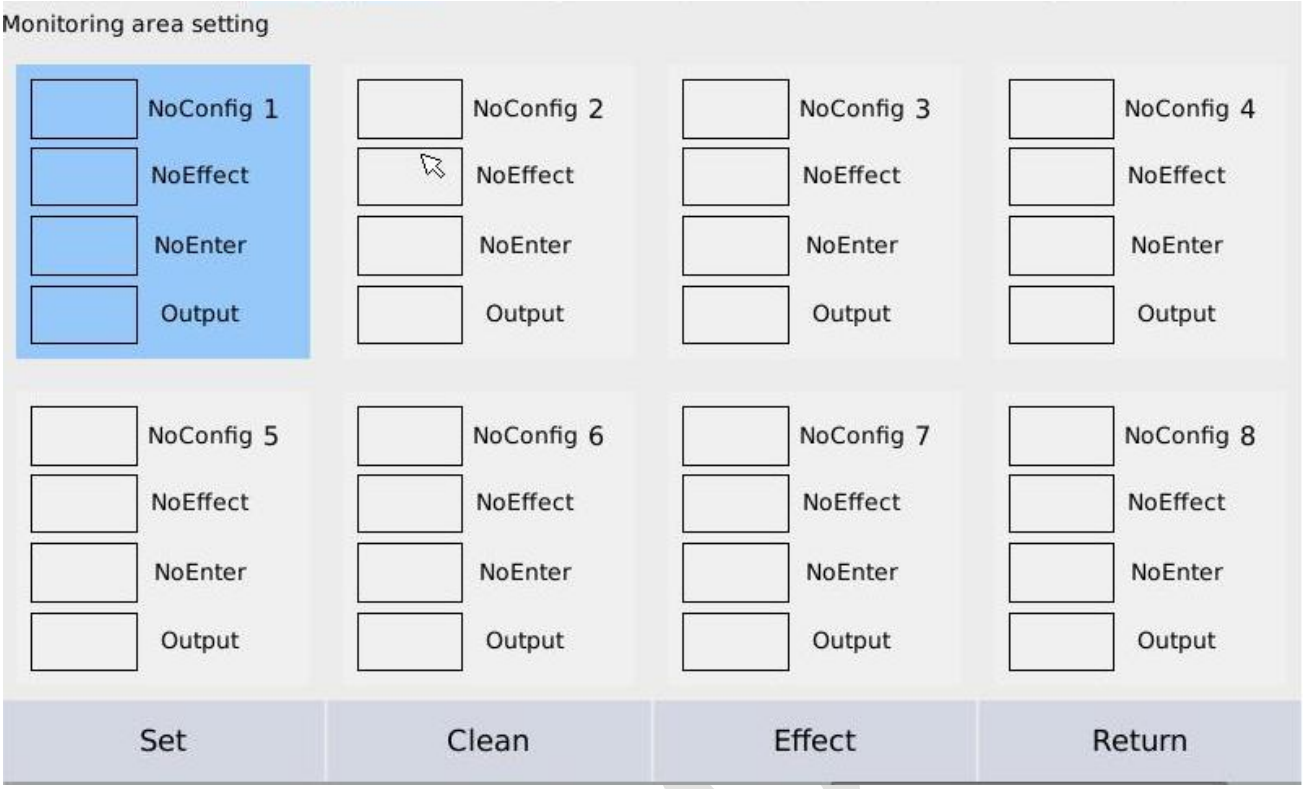


FIG. 5-3-2-1 Monitor interval setting main page

Button icons	Description
Set	Open monitor interval setting, as FIG. 5-3-2-2 monitor interval details setting
Clean	Clear corresponding S/N configuration contents
Effect	Effective: Controller starts monitoring if TCP end arrived at the target area or not Effective cancel: Controller stops monitoring if TCP end arrived at the target area
Return	Exit this interface back to the previous menu

2. Monitor interval setting

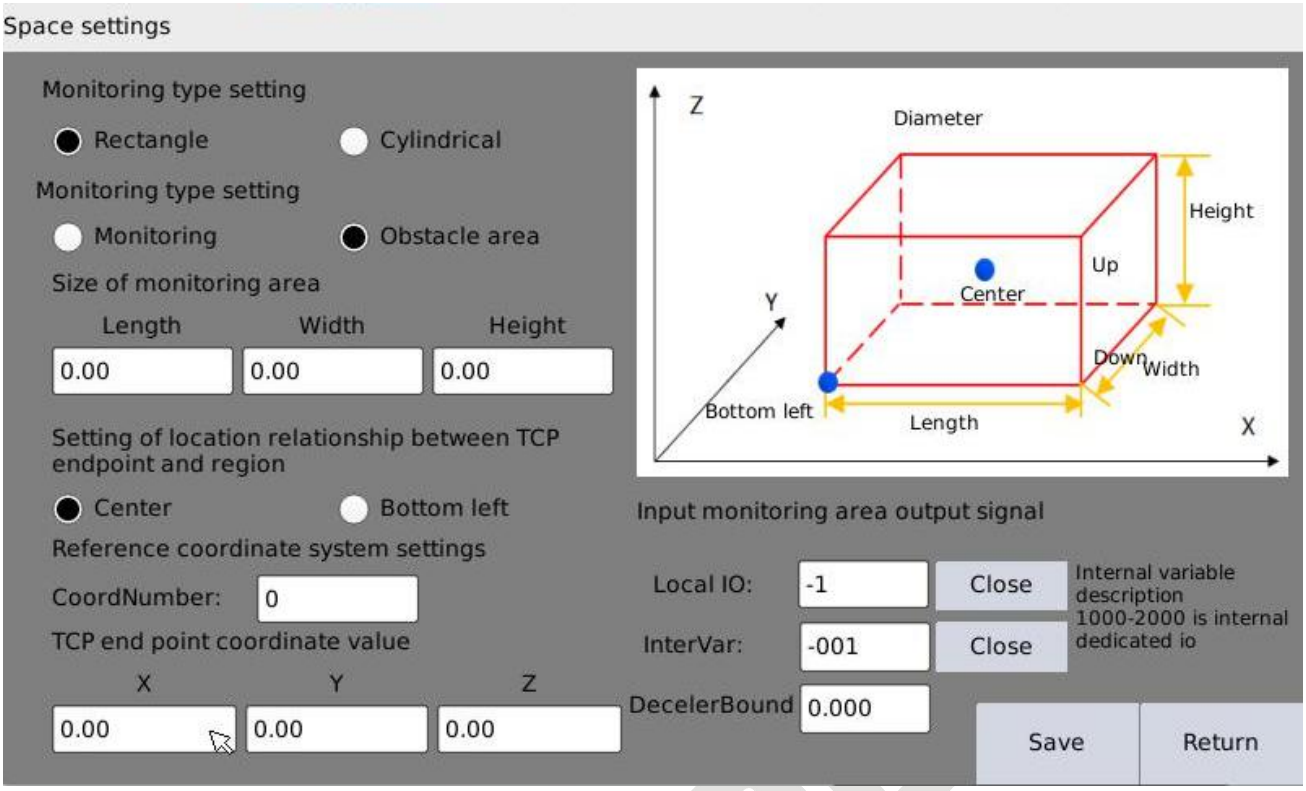


FIG. 5-3-2-2 Monitor interval details setting

Button icon	Description	
Close	OFF: TCP end point output is false in monitor/obstacle area ON: TCP end point output is True in monitor/obstacle interval	
0.00	Expand keyboard, input or modify parameter	
	Length	Reference Length of Axis X direction, Unit: mm
	Width	Reference Length of Axis Y direction, Unit: mm
	Height	Reference Length of Axis Z direction, Unit: mm
	Coordinate system S/N	Reference coordinate system S/N, world coordinate as default
	TCP end point coordinate value	It represents spatial location at reference coordinate system S/N. It can capture the current location value through "Teach" button on main interface above.
	Local IO	Effective local output terminal (Include extended I/O if applied)
	Internal variable	1000~2000, use ReadDI instruction get the status
Save	Save setting value	
Return	Exit current interface, back to previous menu	

5.3.3 Parameters

5.3.3.1 DH parameter

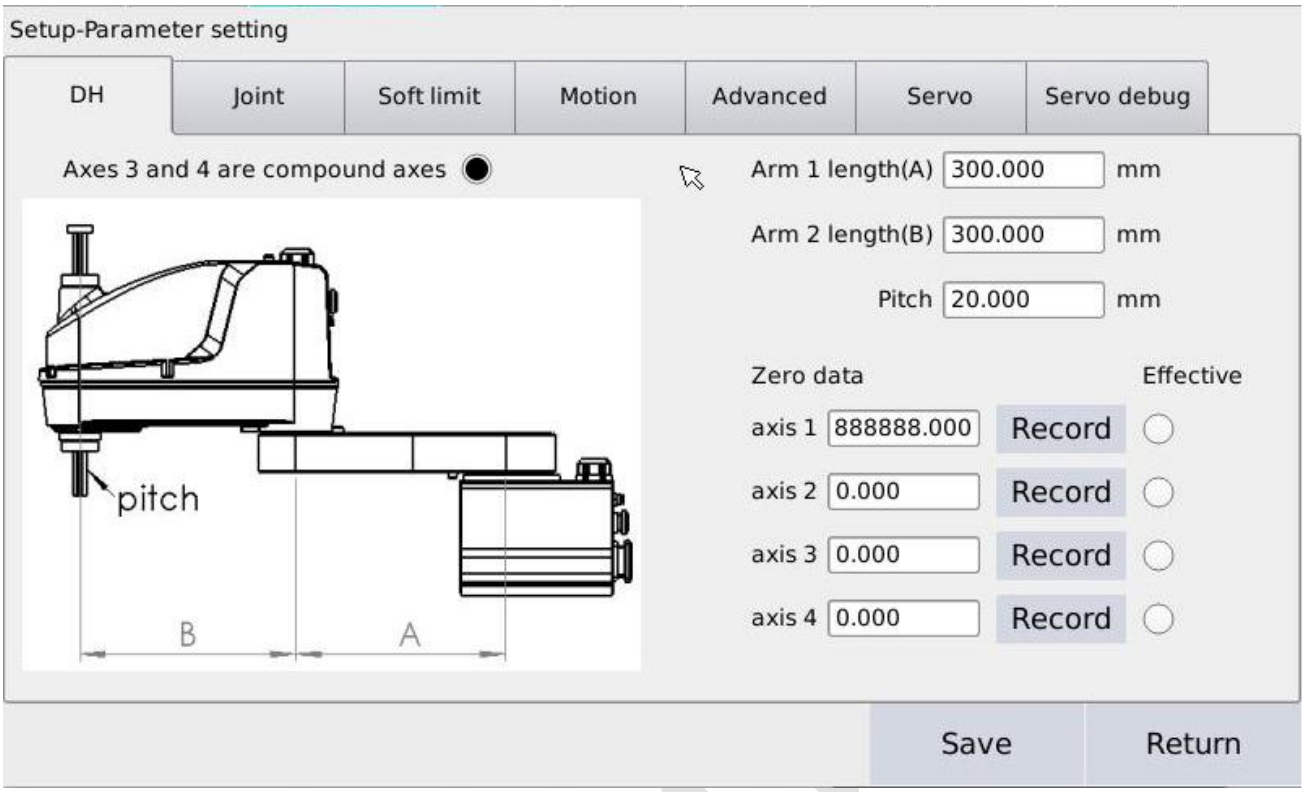


FIG. 5-3-3-1 DHL parameter setting

Button icon	Description
Axes 3 and 4 are compound axes	If Axle 3 or 4 separate, cancel this tick
Arm 1 length(A) 300.000 mm Arm 2 length(B) 300.000 mm Pitch 20.000 mm	Corresponding parameters to be filled
Zero data Effective axis 1 888888.000 Record axis 2 0.000 Record axis 3 0.000 Record axis 4 0.000 Record	After axle point recorded down, read encoder location value is current mechanical zero value **Danger** Make sure you understand the meaning or implication of this parameter, improper modify may risk crash!

5.3.3.2 Joint parameter

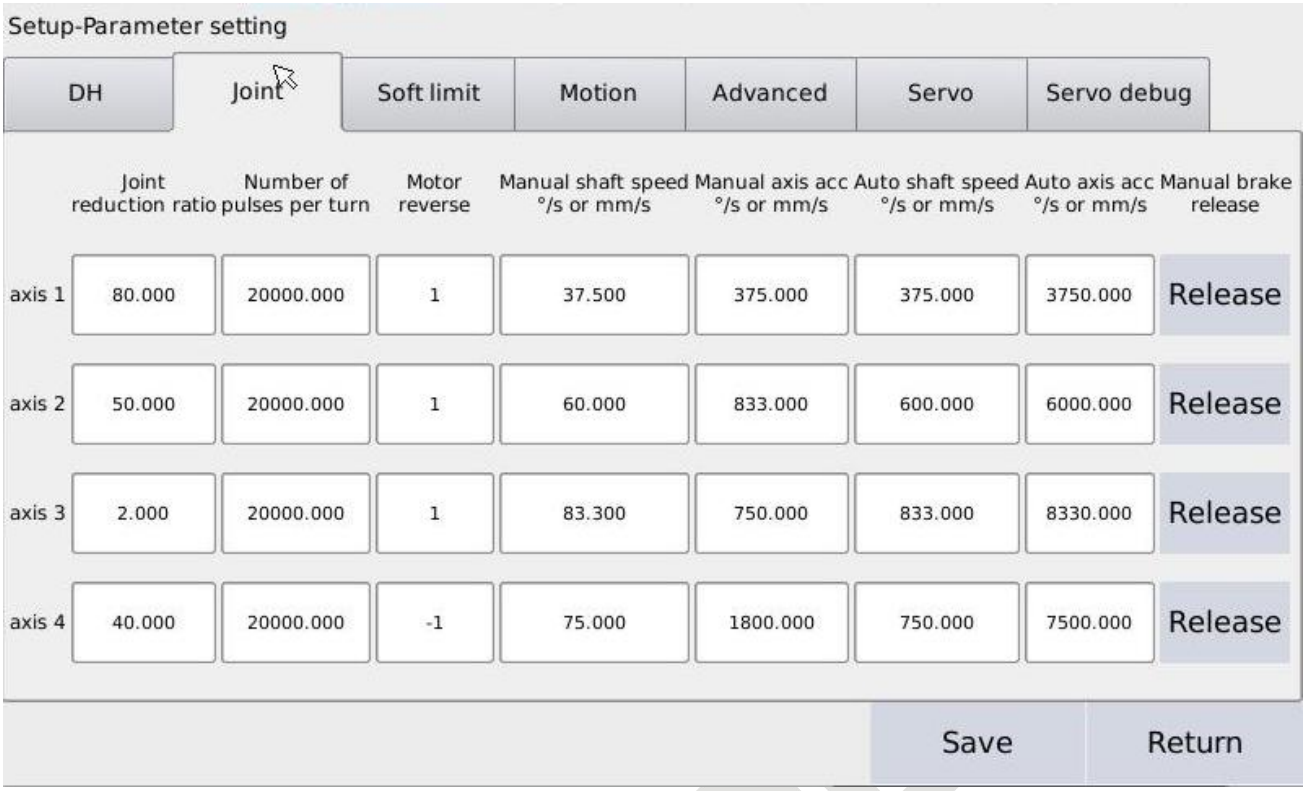


FIG. 5-3-3-2 Joint parameters setting

Button or icon	Description
Release	Press down release axis motor's brake
Joint reduction ratio Number of pulses per turn Motor reverse	This parameter is matched by the factory **Warning** **Make sure you understand the meaning or implication of this parameter, Improper modify may risk speed reducer worn out untimely or motor burnt down ! Any modify should be at the Technician's support.**
Manual shaft speed °/s or mm/s Manual axis acc °/s or mm/s Auto shaft speed °/s or mm/s Auto axis acc °/s or mm/s	It affects Robot interpolation motion speed and accel eration at axes joint. **Warning** **Make sure you understand the meaning or implication of this parameter, Improper modify may risk speed reducer worn out untimely or motor burnt down ! Any modify should be at the Technician's support.**

5.3.3.3 Software limit setting

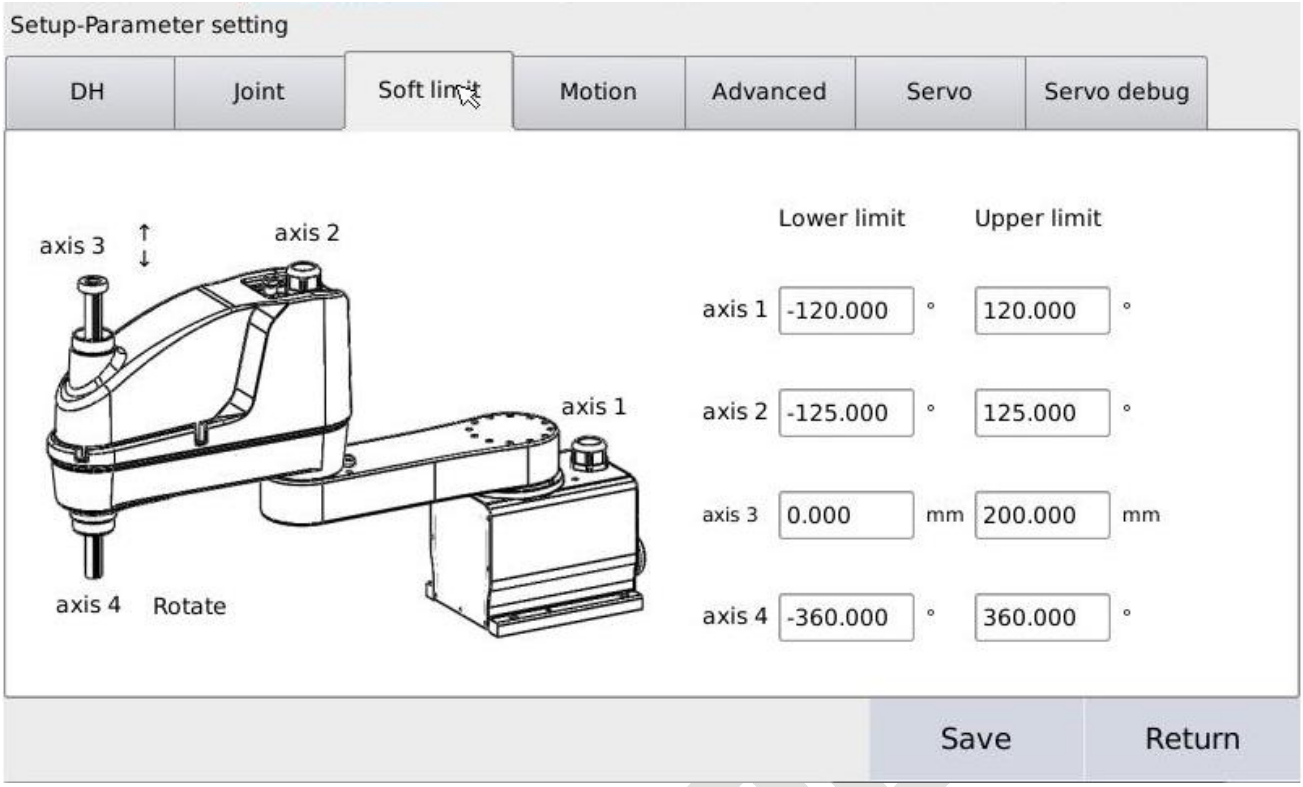


FIG. 5-3-3-3 software limit parameter setting

Button or icon	Description
Lower limit	Affect Robot max operation range **Warning**: **Make sure you understand the meaning or implication of this parameter, Improper modify may risk speed reducer worn out untimely or motor burnt down ! Any modify should be at the Technician's support.**
Upper limit	

Skill	Turn off the servo enable to open the coordinate monitoring, push the robot (The 3rd axis's brake needs to be released), and observe the maximum operating range of each joint through the monitoring window. On this basis, reduce the limit on both sides by 3~5, which can effectively avoid the risk of collision.

5.3.3.4 Movement parameter

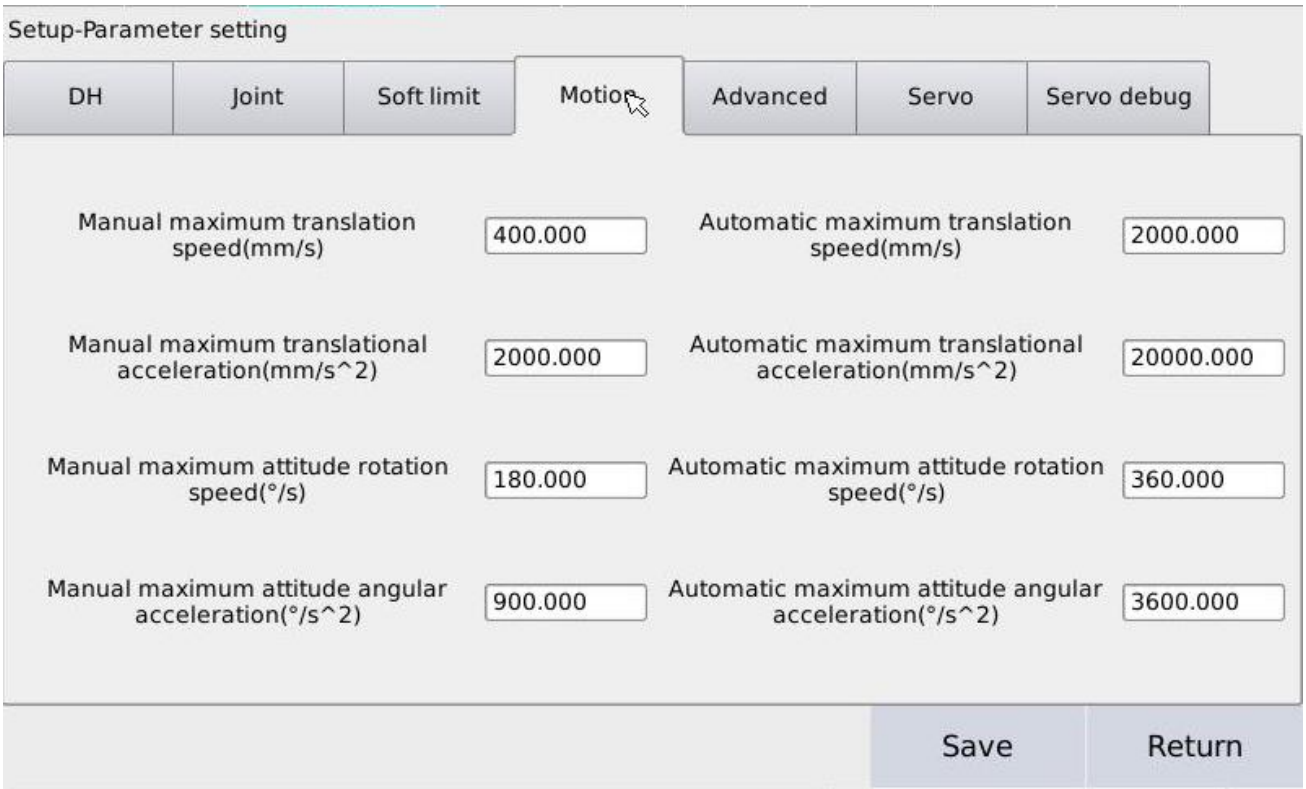


FIG. 5-3-3-4 Movement parameter setting

Button or icon	Description
translation	Affect Robot linear interpolation velocity
rotation	Affect Robot posture interpolation velocity

5.3.3.5 Advanced parameter

Setup-Parameter setting

DH | Joint | Soft limit | Motion | Advanced | Servo | Servo debug

Num	Name	Value	Remarks
0	Robot type(Number of axes)	4	4:SCARA;6:six-axe robot; 7:seven-axe robot;
1	Joint 1 direction	1	
2	Joint 2 direction	1	
3	Joint 3 direction	1	
4	Joint 4 direction	-1	
5	Joint 5 direction	1	
6	Joint 6 direction	1	

Save | Return

FIG. 5-3-3-5 advanced parameter setting

This table contains all the parameters mentioned above, modify requires caution.

The factory code required to contact sales after service.

Warning	**Make sure you understand the meaning or implication of this parameter, Improper modify may risk speed reducer worn out untimely or motor burnt down! Any modify should be at the Technician's support.**

5.3.3.6 Servo parameters

Setup-Parameter setting

DH | Joint | Soft limit | Motion | Advanced | Servo | Servo debug

Num	Name	Value	Properties
0	Software version	0	ReadWrite
1	Motor model	0	ReadWrite
2	Frequency doubling molecule	0	ReadWrite
3	Frequency doubling denominator	0	ReadWrite
4	Encoder model	0	ReadWrite
5	Pole zero value	0	ReadWrite
6	Reset absolute encoder turns alarm	0	ReadWrite

axis 1 | axis 2 | axis 3 | axis 4 | Save | Return

FIG. 5-3-3-6 Servo parameter setting

Warning	**Make sure you understand the meaning or implication of this parameter, Improper modify may risk speed reducer worn out untimely or motor burnt down! Any modify should be at the Technician's support.**

5.3.3.7 Servo adjust

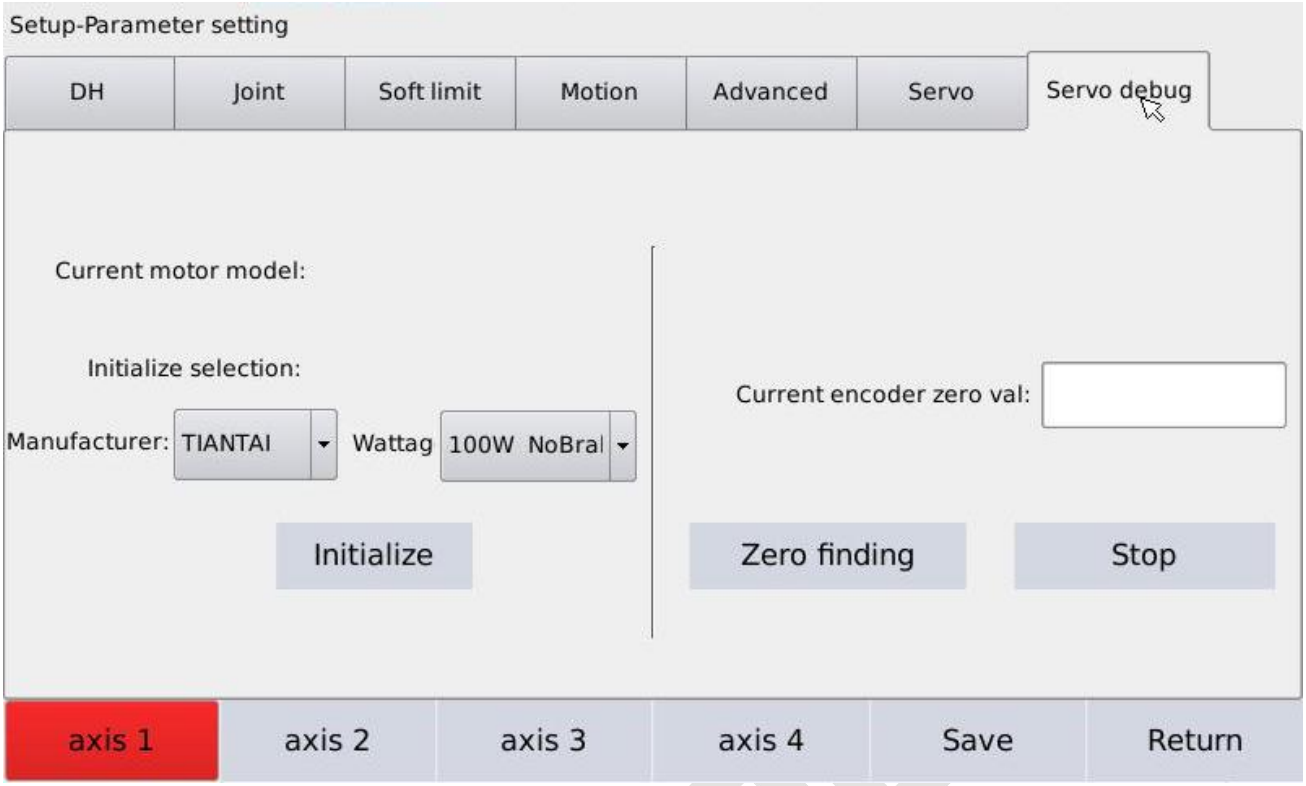


FIG. 5-3-3-6 Servo parameters setting

This page is used for initial configuration of servo parameters. Do not set the parameters arbitrarily unless necessary.

Warning	Make sure you understand the meaning or implication of this parameter, Improper modify may risk speed reducer worn out untimely or motor burnt down! Any modify should be at the Technician's support.

5.3.4 Remote configuration

5.3.4.1 Remote function introduction

Remote function: when Teaching key turn to remote mode, turn on the Robot not through Teach Pendant related button but via dedicated remote IO or Modbus communication to control Robot running, pause, reset, emergency stop etc. Movement, hence called external control. When the input criteria meet the requirements, the preset specified program file will be automatically opened to start execution, regardless of whether the preset file is currently open.

5.3.4.2 Remote motion sequence chart

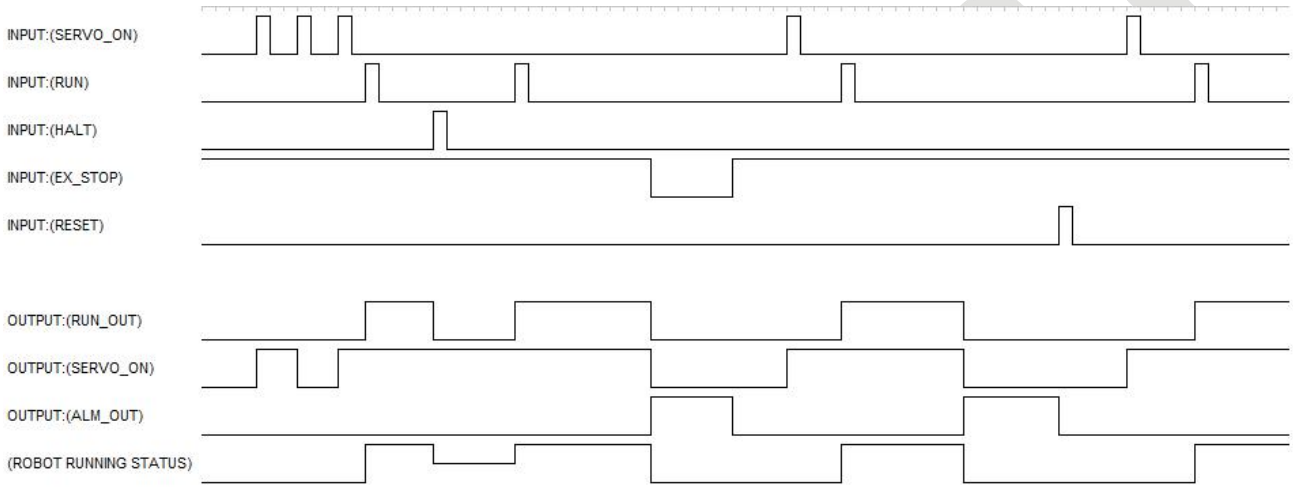


FIG. 5-3-4-2remote input out status sequence chart

Description:

1.Input signal is effective for edge except emergency stop. The continuous input period is valid only once at the beginning and invalid thereafter

2.Output signal is continuous output, ON/OFF status integrated control by input and output status.

3.When Robot in halted state, input enable signal once, and its status will be reversed once, that's disable and enable status switch. While Robot in running or pausing, input enable signal determine invalid, that's if you input enable signal enable failure.

4.Under disable status input running signal Robot cannot move, program cannot operate. But if tick auto enable option exceptional.

5.Input pause/emergency signal Robot would take 50~500ms (Movement speed varies slightly.) to full stop from the movement ; and activating signal is forbidden within this period.

6.Input pause signal on the program running period, Robot stop movement and its program remains pause state, not be enabled. Program pointer stops at the existing row (Non-running complete, movement instruction is interrupted, non-movement instruction would be non-executed.) Once re-enter activating signal, the program continue the execution from the program existing pointer lies on.

7.When emergency stop in effective, Robot disconnect enable immediately, stop the program, cursor stops at the existing row, alarm signal output meantime. During all this emergency stop effective period, input enable signal, Robot not respond; Reset emergency stop, alarm eliminate automatically.

8.Reset emergency stop, whether input reset signal or not, program pointer would reset to the 1st row.

9 .While Robot on regular operation, if alarm arise from non emergency stop, then enable disconnect, Robot stop the running, input reset signal to eliminate the alarm, program pointer reset to the 1st row meanwhile.

5.3.4.3 Remote configuration interface

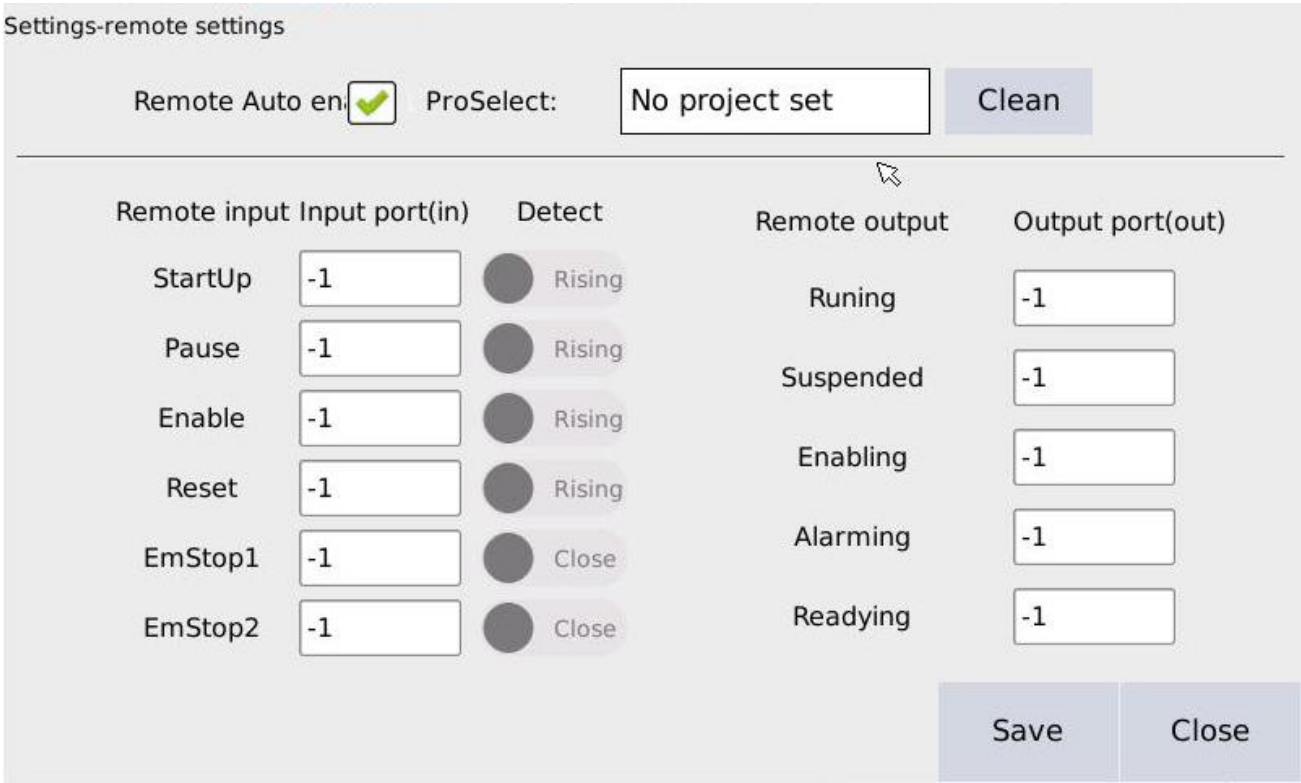


FIG. 5-3-4-3 remote configuration interface

Function	Description
Remote Auto en	Tick and save, while external activating signal was given, auto enable, no need to give separate enable signal.
ProSelect:	Touch box pops up project selection screen, set project to be called in remote mode. If not selected, project started as currently loaded by the auto mode.
Clean	Delete already configured remote project.
-1	Configured remote input/output signal No.
Rising	Switch rising edge or falling edge for detecting.
Close	Switch Normal-Close or Normal-Open for E-stop detecting.

Remark:

Emergency stop is suggested to set as NC, to prevent not stop imminently when the input signal is disconnected unexpectedly.

5.3.5 IO configuration

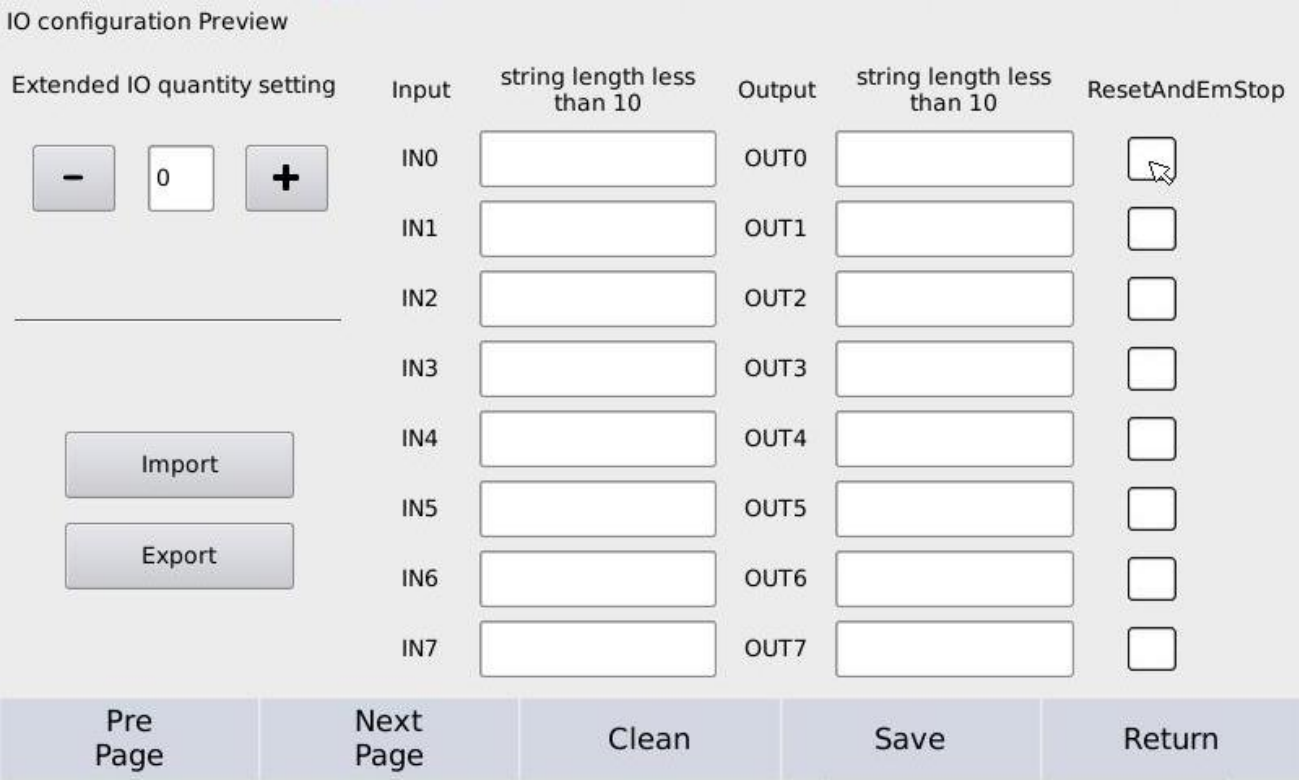


FIG. 5-3-4-3 IO configuration interface

Function	Description
Extended I/O quantity setting	Setting extended IO quantity, range 0~4, 0 means not using extension. Setting will be in effective after restart the system.
Remark	Remark IO name, associate the name on the I/O monitor page. It is irrelevant to the actual program variable name.
Reset after emergency stop	After tick, the selected output port change to OFF status one time when emergency stop be triggered.
Export	Read the I/O configuration file from the controller and export it to the USB flash drive.
Import	Read the I/O configuration file from the USB flash drive and import it to the controller
Clean	Clean specified port remark name
Save	Save whole configuration file into the controller

5.3.6 Communication setting

Details refer to <Chapter 7 Communication function introduction>.

5.3.7 Version

5.3.7.1 Backup to upgrade

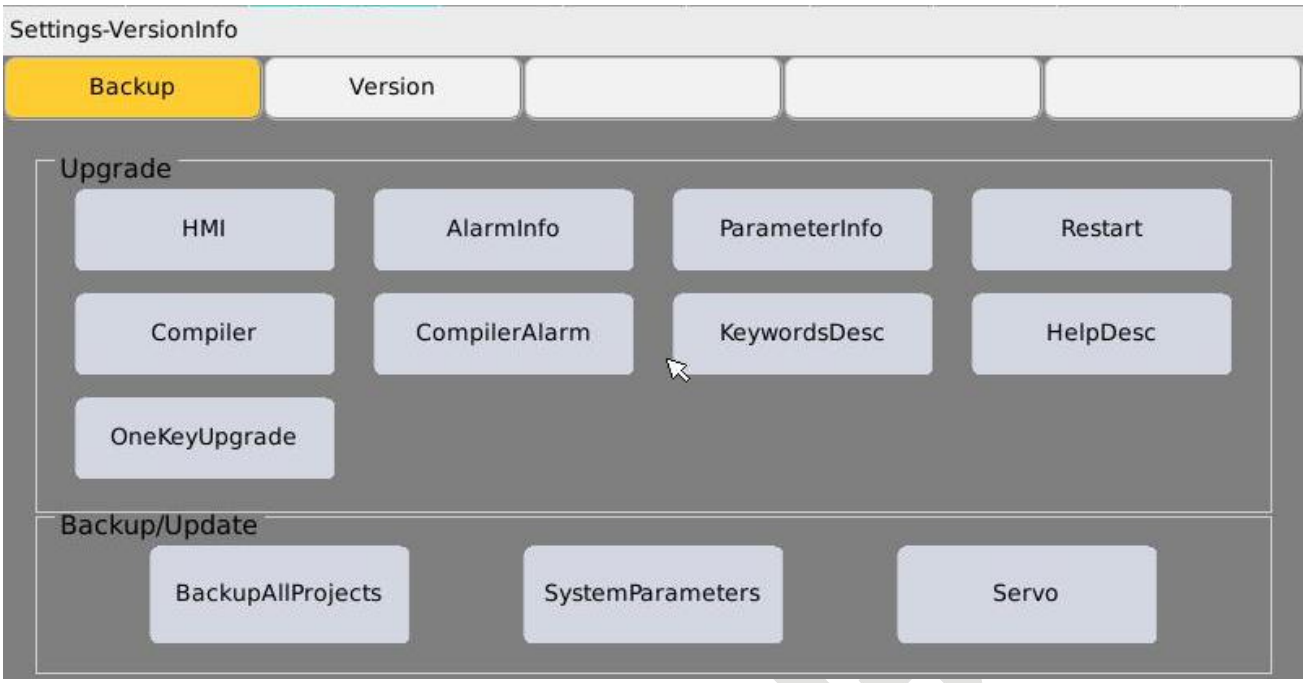


FIG. 5-3-7-1 upgrade backup interface

Warning	**Use with the support of technicians.**

5.3.7.2 Version

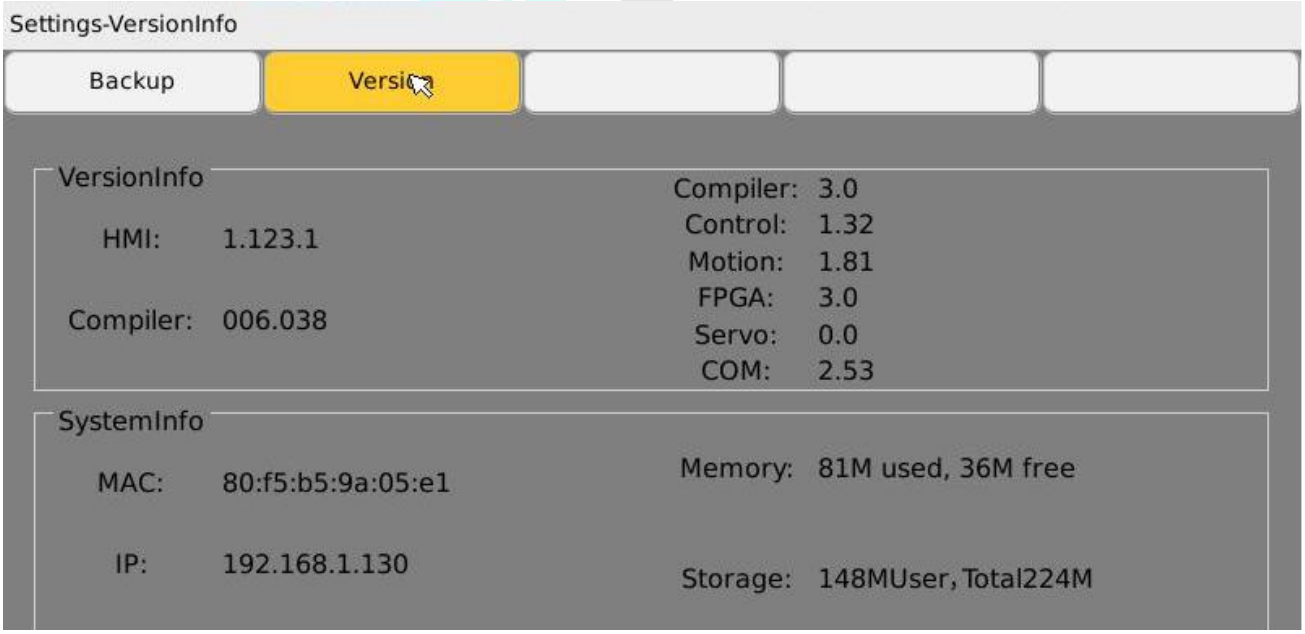


FIG. 5-3-7-2 version viewing interface

Display current system & Teach Pendant software version.

5.3.8 System setting

5.3.8.1 Time / Screen calibration

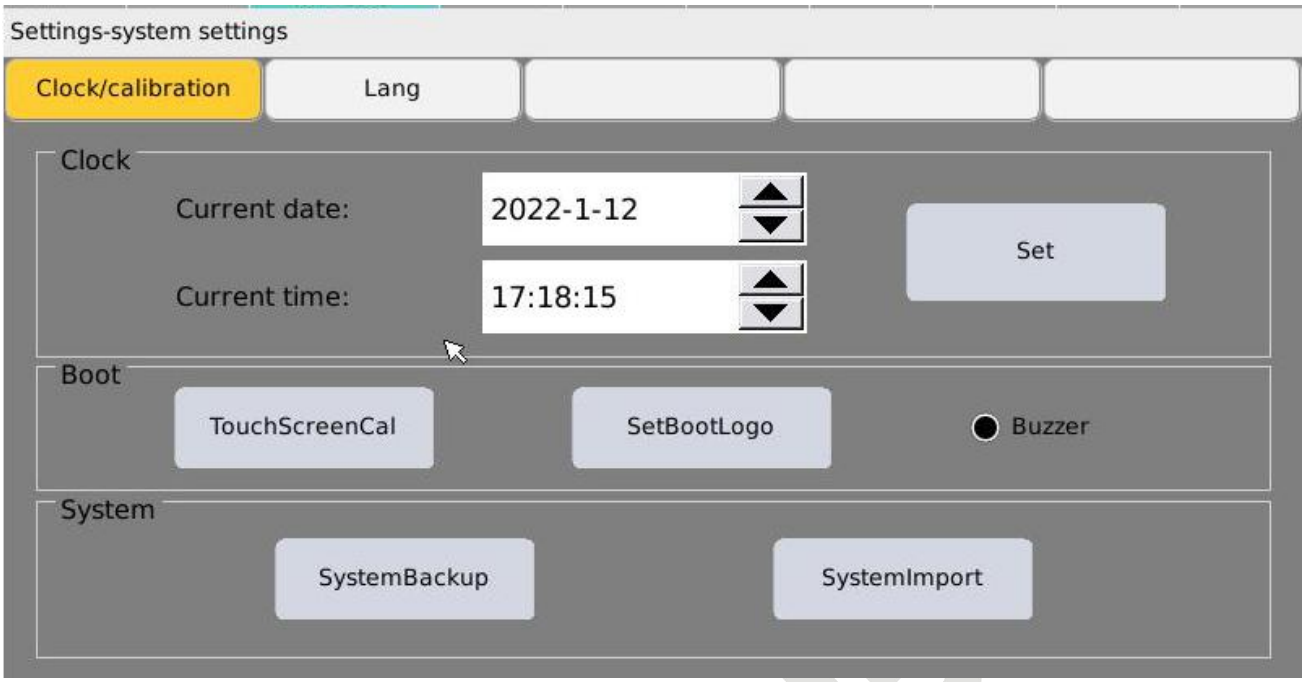


FIG. 5-3-8-1 system time setting interface

Setting system date and time, and can select ON/OFF buzzer.

5.3.8.2 Language

FIG. 5-3-8-2 language setting interface

Tips	**After change language, user need restart the system.**

5.3.9 Renewal

When renewal alarm pops up, liaise with factory for renewal code, then it will return back to service.

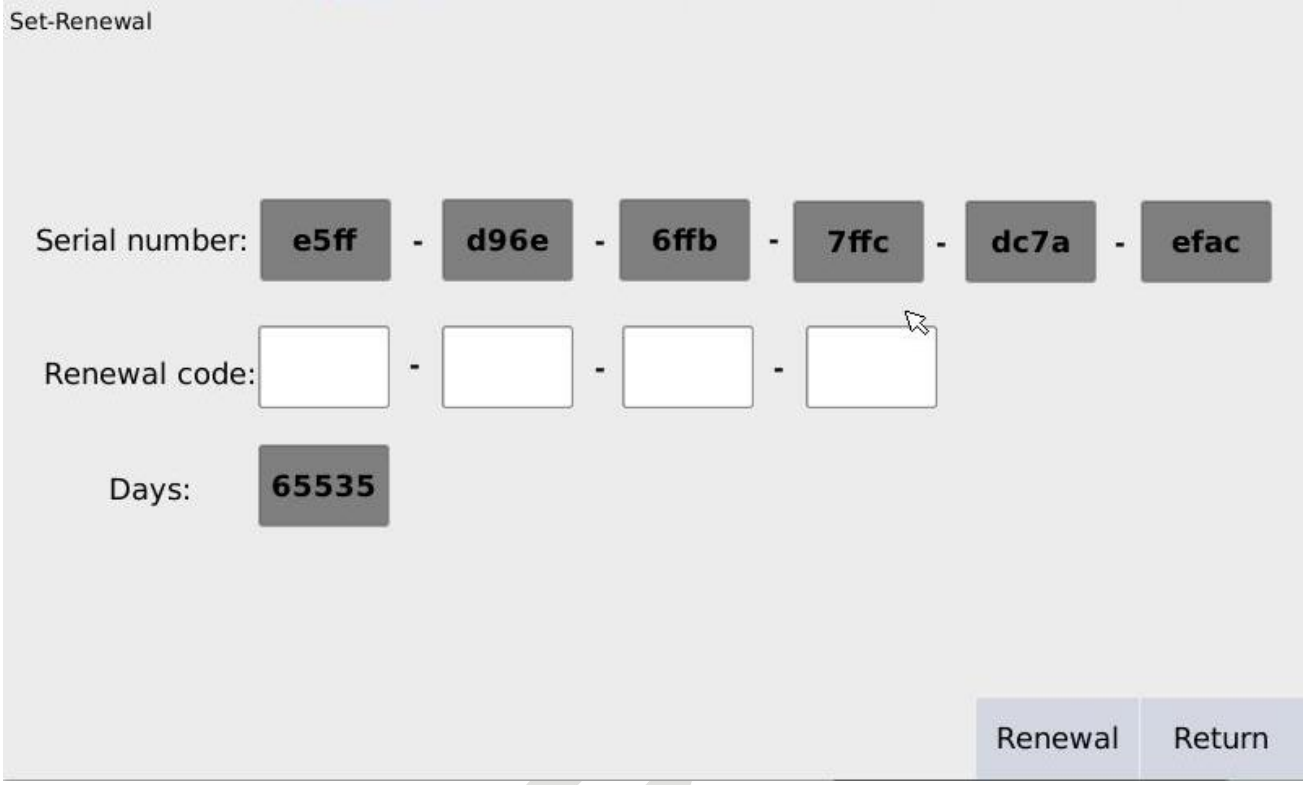


FIG. 5-3-9 renewal setting interface

5.3.10 Teaching pendant offline

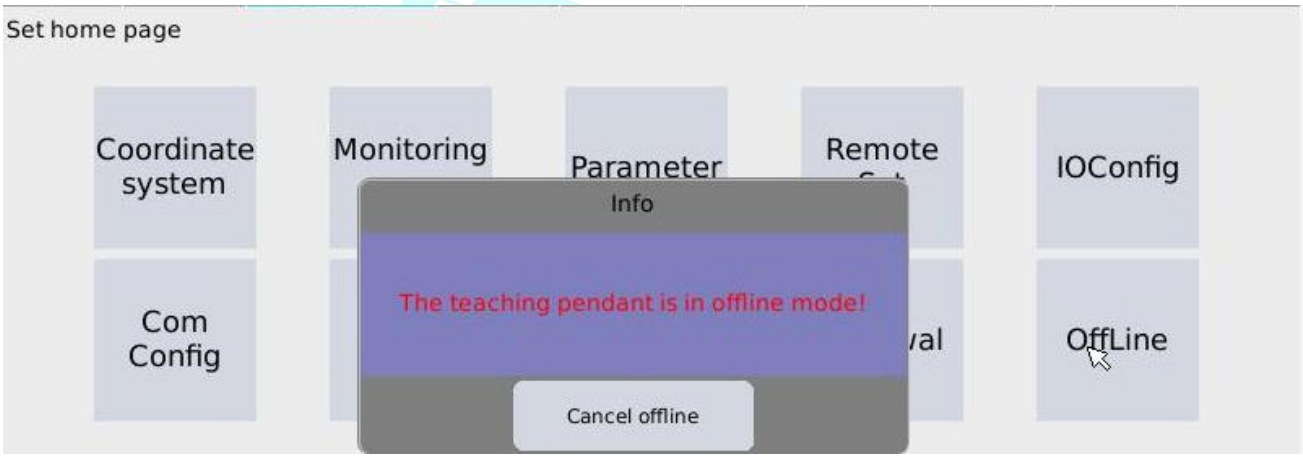


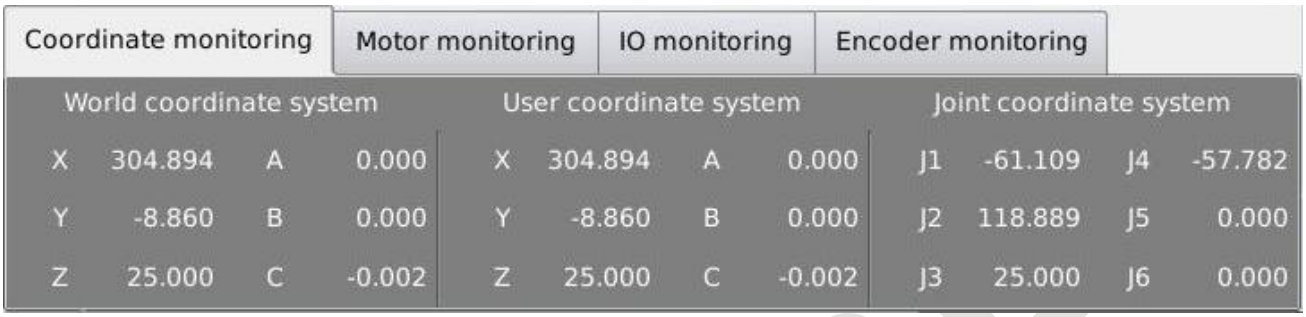
FIG. 5-3-10 off line mode page

When controller is under offline status, teach pendant can unplug or plug. The offline means teach pendant disconnect with the controller, to realize hot unplug.

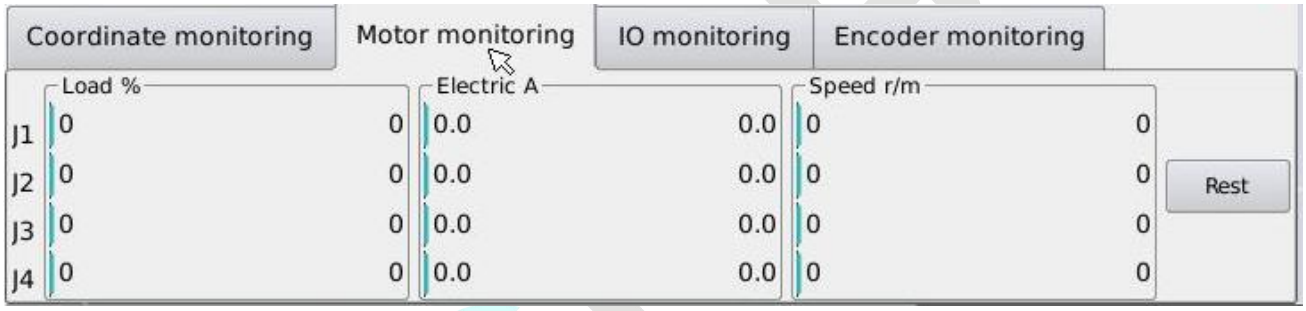
Alert	When the Teach Pendant unplug at offline mode and re-inserted, plug contact time preset in this system is 5 seconds, within this period connect/disconnect like this reconnect, there won't be any exceptional, but when detect out of this time period, determine reconnect will occur emergency stop.

5.4 Real-time info monitor

5.4.1 Coordinate system values monitor



5.4.2 Motor monitor



5.4.3 IO monitor

5.4.4 Robot axes motor encoder

Tips	**This encoder values stands for Robot axes motor encoder values.**

5.5 History warning info

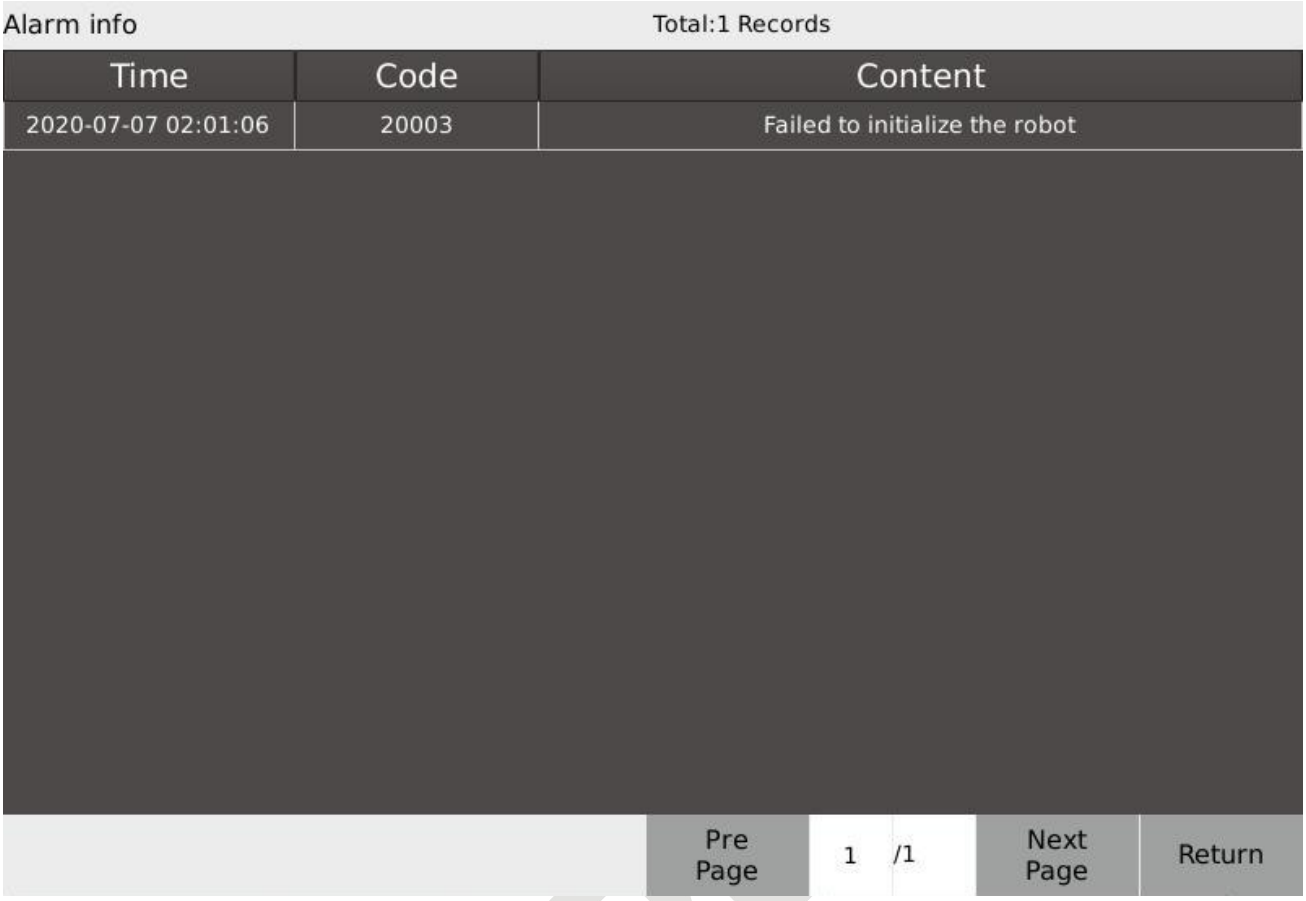


FIG. 5-3-3-10 History warning info

Remark:

After each startup can only record the exceptional what happened this time.

Chapter 6 Program system introduction

6.1 Programming syntax

6.1.1 Project file structure

Type	Suffix	Min. Qty	Max. Qty	Format	Description
Main task	pg	1	1	Process Main ... ProcessEnd	Allow many .pg file in the project, but only configure 1 file as a main task to run. 1 project must have one main task.
Background program	pgb	0	7	Process Sub ... ProcessEnd	Allow many exist in project, but only can select 7 background program to run.
Global variable	gl	1	1	Process Global ... ProcessEnd	Stored 1 file, can edit from any one configured project (main task or background task), Global variable file, its content is valid only to this project.
Point table	pts	1	1		Store max. 10000 points, each project equip one point position table

6.1.2 Variable type and scope

Class	Data type	Scope	Others
System	Integer	All project	I0~I99 (non-power-failure on hold type) GI0~GI49 (power-failure on hold type) Modbus instant communication
	Float	All project	F0~F99 (non-power-failure on hold type) GF0~GF49 (power-failure on hold type) Modbus instant communication
Project	System provided all type	Current project	Power-failure on hold
Task	System provided all type	Current project	Single task common form
Modules	System provided all type	Current modules	Different modules, variable name can be repeat(Defined in Function custom functions)

6.1.3 Custom variable declare type

Keyword	Data type	Min. value	Max. value	Others
int	Integer	-32768	32767	
uint	Unsigned integer	0	65535	
float	Float	N/A	N/A	Precision 0.000001
string	String	1	128	Length range
point	Point	N/A		Fixed type:(float,float,float,float,unit)

6.1.3.1 Variable naming scheme

1. Must begin with Alphabet/letter, can be followed by letters, digits, and underscores (_)

2. Upper and lower case letters stands for different variable,

int A, a

stands for declare 2 different variable.

6.1.3.2 Array

1. The type declaration is followed by [ ] to specify the array capacity, max 65535. For example: int arr[20] , abc

2. Array index value range: 0~capacity minus 1.

For example:

abc = arr[2]

That means assign the 3rd element of the arr array to variable abc.

Tips	All variables declared in global variables are global (valid in current project), besides this, all other variables declared are local

6.1.4 Custom function

6.1.4.1 Format description

Use the keyword "Function" in your program to define your self-defined function.

Declare format as follow:

```
Function  User_defined_name(type definition, parameters variable,...)
        User defined method
        (omit) Return returning value// being executed direct exit Function
        Other operation
FunctionEnd
```

Rules of use

Name:

Begins with English letter, followed by the underscore, digit, letter (uppercase or lowercase letters);

Location in document:

Declare well before calling

Rules of calling

Called after Function declare, quoted user defined name and ( )

Details of using refer to the example below.

Max. Declare qty:

50 pcs

Tips	Define a custom function in the main task or background task, it is equivalent to a local variable. Follow the rule of "define first, use later", otherwise, an error will be reported. Function between tasks cannot access to each other. Define function to the global variable, each task can access.

6.1.4.2 Example of Non movement type user defined function

Case 1:

```
//User defined an add function
Function myadd ( float a , float b )
        Return a+b
FunctionEnd
```

Usage in the program as follows, for instance, write in the main program:

```
Process Main
    Function myadd ( float a , float b )
        Return a+b
    FunctionEnd
    float i = myadd(100.2 , 2.5)
    Print("calculation result i = " , i)
ProcessEnd
//Use user defined function myadd to calculate, i return value is 102.7
```

Case 2:

```
//User defined location calculation
//4 point calculation define
Function GetposAtRight(point posa,point posb,point posc,point posd,int i,int j,int row,int col)
    float xa=posa.x , ya=posa.y , za=posa.z , ca=posa.c
    float xb=posb.x , yb=posb.y , zb=posb.z
    float xc=posc.x , yc=posc.y , zc=posc.z
    float xd=posd.x , yd=posd.y , zd=posd.z
    float resualt_x=((row-i)*(col-j)*xa+i*(col-j)*xb+i*j*xc+(row-i)*j*xd)/(row*col)
    float resualt_y=((row-i)*(col-j)*ya+i*(col-j)*yb+i*j*yc+(row-i)*j*yd)/(row*col)
    float resualt_z=((row-i)*(col-j)*za+i*(col-j)*zb+i*j*zc+(row-i)*j*zd)/(row*col)
    point rtn = BuildPoint(resualt_x,resualt_y,resualt_z,ca,1)//right hand side
    Return rtn
FunctionEnd
```

6.1.4.3 Examples of custom functions with movement types

Case 1: Get label program

```
Function GetLable()
        point feederPos
        If I80 ==2 Then
                feederPos = feederPos1
        EndIf
        If I80 == 3 Then
                feederPos = feederPos2
        EndIf
        MOVJ(feederPos+Z(30,0),speed,acc,dec,cp)
        Open(vacuum)
        MOVJ(feederPos,speed,acc,dec,cp)
        Delay(500)
        MOVJ(feederPos+Z(30,0),speed,acc,dec,cp)
        Break()
FunctionEnd
...
GetLable() //Call GetLabel() function
```

6.1.5 Keyword list

Keyword list					
int	MOVJ	Zlimit	Open	Close	BuildPoint
uint	MOVL	Delay	Pulse	ReadDI	Matrix
float	MOVC	Rand	WaitDI	ReadDO	U2U
string	Dn	And	Or	Not	Where
point	JUMP	SysTime	Print	Pause	X
Do	Exit	While	Util	Loop	Y
If	EndIf	Else	ElseIf	Then	Z
For	To	Step	Next	GoTo	C
Switch	EndSwitch	Case	Default		.x
Power	Sin	Cos	Tan	Pn	.y
Abs	Asin	Acos	Atan		.z
OpenCOM	CloseCOM	ReadCOM	WriteCOM		.c
CheckCOM	SetCOM			Tool	.j1
OpenNet	CloseNet	ReadNet	WriteNet	User	.j2
CheckNet	SetNet			SetTool	.j3
StrAscValue	StrAscChar			SetUser	.j4
StrLen	StrLeft	StrRight	StrMid	StrTrim	StrFind
StrReplace	StrSlicer	StrEmpty	StrCat	StrFormat	StrData
StrCmp	StrnCmp	StrToupper	StrTolower	StrToString	StrToValue
Process	Main	Sub	Global	ProcessEnd	Function
FunctionEnd		MIRead	MFRead	MIWrite	MFWrite
		GIRead	GFRead	GIWrite	GFWrite

With the system software update, part keyword list may adjust, the actual provided instruction list prevails.

6.1.6 Operation symbol list

Operation symbol list					
Operation symbol	**Action**	**Operation symbol**	**Action**	**Relational Operation symbol**	**Action**
+	add	**//**	note	**>**	greater
-	minus	**( )**	function	**<**	smaller
*****	multiply	**[ ]**	array symbol	**==**	equal
/	divide	**:**	jump label	**>=**	greater than or equal to
%	remainder	**,**	parameter separation	**<=**	smaller than or equal to
=	valuation	**.**	quoted coordinate member	**!=**	unequal to
		"…"	**…**string		
		!…!…!	Parallel instruction		
		\n	new line		
		$b	binary		
		$o	octonary		
		$x	hex		
		%X	At "StrFormat" instruction, use %04X stands for 0x0000 format.		

6.2 Instruction description

6.2.1 Movement

Instruction Name	Function Introduction
MOVJ	(Format 1) Movement from current location to the target point by way of point to point joint interpolation
MOVJ	(Format 2) Assign some certain axis movement to the target location (angle or mm)
MOVL	Moves from the current position to the target in a straight line interpolation manner
MOVC	Starting from the current position, according to the current position, the path position and the target position calculation to reach the target point by circular interpolation
JUMP	To reach the target point through gate type joint interpolation (vertical direction gate type, horizontal direction not fixed)
Dn	Setting parallel instruction and synchronization of starting execution account for total trip % value
Tool	Select the tool number and use it in subsequent movements until it is switched to another tool
SetTool	Modify the data of the tool number
User	Select user number and get it applied into the exist, until changed by another.
SetUser	Modify the data of the user number

MOVJ

MOVJ(point_a, speed, acc, dec, cp)

Function: Movement from current location to the target point by way of point to point joint interpolation

Parameter:

point_a	target location
speed	Max operation speed, setting range 1~100
acc	Setting acceleration, range 1~100
dec	Setting deceleration, range 1~100
cp	Path continuous setting, range 0: turn off, 1: turn on

Return: N/A

Description: Path is not fixed mode, there is enough room to avoid interference

Example: Case 1

```
int speed=100,acc=100,dec=100,cp=0          //define movement parameters
Do
float old_time=SysTime()                    //get current time
MOVJ(Pn(1),speed,acc,dec,cp)                //movement to the 1# point
MOVJ(Pn(2),speed,acc,dec,cp)                //movement to the2# point
MOVJ(Pn(3),speed,acc,dec,cp)                //movement to the 3# point
MOVJ(Pn(2),speed,acc,dec,cp)                //movement to the 2# point
                                            // Recapture the current time, minus the prior time, to reach the consumption time
Print("Program cost time",(SysTime()-old_time)/1000,"s" )
Loop
```

Case 2

```
int speed=100,acc=100,dec=100,cp=0          //define movement parameters
point pt_a=Pn(4)                            //define point type variable pt_a, assign the variable its initial value with the 4# point in the point table
point pt_b=Pn(5)                            //define point type variable pt_b, assign the variable its initial value with the 5# point in the point table
point pt_c=Pn(6)                            //define point type variable pt_c, assign the variable its initial value with the 6# point in the
```

```
                                        point table
Do
float old_time = SysTime()              //get current time
MOVJ(pt_a , speed , acc , dec , cp)     //Movement to "pt_a" point position, equivalent
                                        to move 4# point in the point table
MOVJ(pt_b , speed , acc , dec , cp)     //Movement to "pt_b" point position, equivalent
                                        to move 5# point in the point table
MOVJ(pt_c , speed , acc , dec , cp)     //Movement to "pt_c" point position, equivalent
                                        to move 6# point in the point table
MOVJ(pt_b , speed , acc , dec , cp)     //Movement to "pt_b" point position, equivalent
                                        to move 5# point in the point table
                                        // Recapture the current time, minus the prior
                                        time, to reach the consumption time
Print("Program cost time",(SysTime()-old_time)/1000,"s" )
Loop


Case 3
int speed=100,acc=100,dec=100,cp=0      //define movement parameters
point pick=BuildPoint(300,-300,0,0,1)   //Define Pick point and assign value
point place=BuildPoint(300,0,0,0,1)     //Define Place point and assign value
float h = 25                            //Define floating type variable h, assign initiate
                                        value 25
Do
float old_time=SysTime()                //Get current time
MOVJ(pick+Z(h,0),speed,acc,dec,cp)      //Movement to Pick upper distance at H point
                                        position
MOVJ(pick,speed,acc,dec,cp)             //Movement to Pick point position
MOVJ(pick+Z(h,0),speed,acc,dec,cp)      //Movement to Pick upper distance at H point
                                        position
MOVJ(place+Z(h,0),speed,acc,dec,cp)     //Movement to Place upper distance at H point
                                        position
MOVJ(place,speed,acc,dec,cp)            //Movement to Place point position
MOVJ(place+Z(h,0),speed,acc,dec, cp)    /Movement to Place upper distance at H point
                                        position
```

// Recapture the current time, minus the prior time, to reach the consumption time

Print("Program cost time",(SysTime()-old_time)/1000,"s")

Loop

Reference: Pn , BuildPoint , Matrix , Where , X , Y , Z , C , Dn ,Break

MOVJ

MOVJ(n, angle_pos, speed, acc, dec, cp)

Function:	Assign some certain axis movement to the target location (angle or mm)	
Parameter:	n	Target axis number (1,2,3,4)
	angle	Absolute position of target (1-2-4 axis Angle,3 axis mm)
	speed	Max. Operation speed, setting range 1~100
	acc	Acceleration setting , range 1~100
	dec	Deceleration setting , range 1~100
	cp	Path continuous setting, range 0: turn off, 1: turn on
Return:	N/A	
Description:	Clearly know the target axle operation would crash or not	
Example:	int speed=100,acc=100,dec=100,cp=0	//Define movement parameter
	MOVJ(1,0,speed,acc,dec,cp)	//Other axes remains still, axis 1 movement to 0 position individually
	float target=30	// Define axes 3 to reach target, 30mm (mm no need to write)
	MOVJ(3,target,speed,acc,dec,cp)	//Other axes remains still, axle 3 movement to 30mm position individually
Reference:	Dn , Break	

MOVL

MOVL(point_a, speed, acc, dec, cp)

Function: Moves from the current position to the target in a straight line interpolation manner

Parameter:

point_a	Target position
speed	Max. Operation speed, setting range 1~100
acc	Acceleration setting , range 1~100
dec	Deceleration setting , range 1~100
cp	Path continuous setting, range 0: turn off, 1: turn on

Return: N/A

Description:

1.The singularity is near the joint Angle of axis J2 which is close to 0 degree. The linear motion in this range may cause the joint overspeed alarm, to avoid as possible to select this point.

2.A straight line between the current point and the target point will cause a soft limit alarm if it passes through the soft limit area and should be avoided as much as possible

3.Speed and acceleration and deceleration parameters are relative values, and affected by parameter-motion parameter-automatic maximum translational speed and automatic maximum translational acceleration

4.Actual maximum linear motion speed calculation method, automatic maximum translational speed× speed× multiplier

Example:

```
int speed=100,acc=100,dec=100,cp=0          //define motion parameter
MOVL(Pn(1),speed,acc,dec,cp)                //Move to the point set in the motion point table
point point_a=Pn(2)                         //Define point position and assign 2#point of point position table as value
MOVL(point_a,speed,acc,dec,cp)              //Movement to point_a
point_a=BuildPoint( 250,450,-50,20,1)       // Changing the data of point_a does not affect the recorded values of Pn(2)
MOVL(point_a,speed,acc,dec,cp)              //Movement to the updated point position
```

Reference: Pn , BuildPoint , Matrix , Where , X , Y , Z , C , Dn ,Break

MOVC

MOVC(point1, point2, speed, acc, dec, cp)

Function: Starting from the current position, according to the current position, the path position and the target position calculation to reach the target point by circular interpolation

Parameter:

point1	passing position
point2	target position
speed	Max. Operation speed, setting range 1~100
acc	Acceleration setting range 1~100
dec	Deceleration setting range 1~100
cp	Path continuous setting, setting range 0: turn on lubricity; 1: turn off lubricity

Return: N/A

Description: 1.Path generated based on 3-point drawing circle rule, the arc segment path should not pass the soft limit

2.Attention required for teaching location, the sequence of passing point and destination point affect the motion path, not of exchange property

Example:

```
int speed=100,acc=100,dec=100,cp=0        //define motion parameter
MOVJ(Pn(1),speed,acc,dec,cp)              //Reach P1 point
MOVC(Pn(2),Pn(3),speed,acc,dec,cp)        //Set from P1, through P2 P3 arc, reach to P3 point finally
```

Reference: Pn , BuildPoint , Matrix , Where , X , Y , Z , C , Dn ,Break

MOVC(point1, point2, angle, speed, acc, dec, cp)

Function: The circle was calculated according to the current position, passing position and target position. The direction of the current point-passing point-target point was taken as the Angle positive direction, and the current position was taken as the Angle starting point and the target Angle as the end point to make circular interpolation

Parameter:

point1	passing position
point2	target position
angle	Target angle, relative to the current position (calculated from the starting point), in degrees, with a value range of - 360 ~ 360
speed	Max. Operation speed, setting range 1~100
acc	Acceleration setting range 1~100
dec	Deceleration setting range 1~100
cp	Path continuous setting, setting range 0: turn on; 1: turn off

Return: N/A

Description:

1.Path generated based on 3-point drawing circle rule, the arc segment path should not pass the soft limit

2.When the angle range setting exceeds ± 360, the movement stops at 360 (or - 360)

3.Attention required for teaching location, the sequence of passing point and destination point affect the motion path, not of exchange property

Example:

```
int speed=100,acc=100,dec=100,cp=0          //define motion parameter
MOVJ(Pn(1),speed,acc,dec,cp)
MOVC(Pn(2),Pn(3),360,speed,acc,dec,cp)      //Set from P1, through P2 P3 arc, reach to P3 point finally
```

Reference: Pn , BuildPoint , Matrix , Where , X , Y , Z , C , Dn ,Break

JUMP

JUMP(point_a, h1, h2, h3, speed, acc, dec, cp)

Function: To reach the target point through gate type joint interpolation (vertical direction gate type, Horizental direction not fixed)

Parameter:

point_a	Target position
h1	Lift height in mm from current position towards Z+
h2	Lift height in mm from target position towards Z+
h3	The maximum z-direction limit (absolute height) passed by the movement process, in mm
speed	Max. Operation speed, setting range 1~100
acc	Acceleration speed, setting range 1~100
dec	Deceleration setting range 1~100
cp	Path continuous setting, in valid here, always move to the target point position

Return: N/A

Description:

1.The upper limit set at least above the current position and the target point plus the lift height respectively;

2.If the height set is lower than the starting point or the height increased by the target point, the lift or descent height will be reduced;

3.When height limit setting a little bigger than raised or before descent height, it maybe more effective;

4.The upper limit value cannot exceed axle J3 software limit upper limit;

5.Others refer to MOVJ

6.Current starting position higher than upper limit would cause alarm, need to lower down the current position or enhance upper limit (Specific analysis with actual application)

Example: Case 1

```
int speed=100,acc=100,dec=100,cp=0          //Define motion parameter
point pick=BuildPoint(300,-300,0,0,1)       //Define pick point and assign value
point place=BuildPoint(300,0,0,0,1)         //Define place point, and assign value
point standby=BuildPoint(300,0,0,0,1)       //Define standby point, and assign value
float h=0.5,zlim=25                         //Define h as height enhanced, zlim means axle Z absolute upper limit
MOVJ(standby,speed,acc,dec,cp)              //Movement to standby point
```

```
Do
float old_time=SysTime()                    //Capture current time
JUMP(pick,h,h,zlim,speed,acc,dec,cp )       //Jump motion(gate type) to pick point
JUMP(place,h,h,zlim,speed,acc,dec,cp )      //Jump motion (gate type) to place point
                                            // Recapture current time and minus the prior
                                            time, reach out the consumption time
Print("Program cost time",(SysTime()-old_time)/1000,"s" )
Loop
```

Reference: Pn , BuildPoint , Matrix , Where , X , Y , Z , C , Dn ,Break

Dn

Dn(per)	
Function:	Setting parallel instruction and synchronization of starting execution account for total trip % value
Parameter:	per 0~100 integer value, unit %, no need to write unit
Return:	N/A
Description:	1.Cannot use individually 2.After applied on motion instruction, format: !MOVJ(Pn(1),speed,acc,dec,cp)!Dn(20);Open(2)! 3.If involves into many parallel operation, use ';' to separate between!...! 4.JUMP type motion, total trip not account while raising and descent 5.If the jump motion does not include horizontal motion, calculate the motion percentage from the descent.
Example:	!MOVJ(Pn(1),speed,acc,dec,cp)!Dn(20);Open(2);Dn(80);Close(2)! //Open 2# port at 20% position and close 2# port at 80% position during movement to P1 point
Reference:	MOVJ , MOVL , MOVC , JUMP , Open , Close

Tool

Tool(tool_num)

Function: Select the tool number and use it in subsequent movements until it is switched to another tool

Parameter: tool_num 0~8 Tool number 0~8

Return: N/A

Description: 1.In special cases, the point in use is pure joint point, and the tool number does not take effect

2.Tool (0) indicates that no external tool is attached, that is, the default tool with all offsets of 0 is the rotation center of the 4th axis (rotation axis). When other tool numbers 1-8 are selected, translate the rotation center to the top of the tool according to the registered tool data;

3. If tool (0) is not added in the program, the system will execute all teaching points or user build points in the mode of tool (0) by default;

4.When the program is executed from line 1, the system will initialize to tool (0);

Example:

```
int speed=100 , acc=100 , dec=100 , cp=0      //define motion parameter
point point_a,point_b                         /define point_a,point_b
point_a=BuildPoint ( 250 , 450 , 50 , 20 , 1 )  //Set point_a data
point_b=BuildPoint ( 260 , 440 , 30 , 25 , 1 )  //Set point_b data
Do
MOVJ ( point_a , speed , acc , dec , cp )     //The first cycle moves to point according to point_a with tool 0#.If it is executed again, the tool will move to point_a with tool 1#.If you want to eliminate this difference, insert tool (0) before the instruction, you can use tool (0) every time you execute this line of instruction.
Tool(1)                                       //Select tool 1 until it is changed again
MOVL ( point_a , speed , acc , dec , cp )     //Move to point_ a with tool 1#.
MOVJ ( Pn(1) , speed , acc , dec , cp )       //Tool 1# does not work if the 1# point is a joint type
MOVJ ( Pn(2) , speed , acc , dec , cp )       //Tool 1#  work if the 2# point is a descartes type
Loop
```

Reference: SetTool , MOVJ , MOVL

SetTool

SetTool(tool_num, xpos, ypos, zpos, cpos)

Function:	Modify the data of the tool number	
Parameter:	tool_num	Tool number: 1~9
	xpos	Absolute value of tool number X
	ypos	Absolute value of tool number Y
	zpos	Absolute value of tool number Z
	cpos	Absolute value of tool number C
Return:	N/A	
Description:	Tool(0) is flange center tool (of no any offset value), cannot modify	
Example:	SetTool(1,10,10,0,0)	//modify 1# tool coordinate system, data: X=10,Y=10,Z=0,C=0
	Tool(1)	//Apply 1# tools below
Reference:	Tool	

User

User(user_num)

Function: Select user number and get it applied into the exist, until changed by another.

Parameter: user_num User number: 0~15

Return: N/A

Description: 1, In special cases, the user number does not take effect when the point in use is a point referenced in the point table

```
User( 2 )
MOVJ( Pn( 1 ) , speed , acc , dec , cp )      // User (2) not affect on this position
point abc = BuildPoint( 23, 45 , 25 , 0 , 0 )
MOVJ( abc ,speed , acc , dec , cp )           //User (2)Being effected point position value
2, User( 0 )                                  //stands for world coordinate system,no offset value
User( 2 )
......
User( 0 )                                     //Switch user back to world coordinate system
```

3.Point generated by "BuildPoint( )" , after "User (xx)" or "Too(xx)", different user or tool number, final position maybe different.

```
point abc = BuildPoint( 23, 45 , 25 , 0 , 0 )
User( 2 )
Tool( 1 )
MOVJ( abc ,speed , acc , dec , cp )           /User (2) Tool (1) being effected point position
Tool( 2 )
MOVJ( abc ,speed , acc , dec , cp )           //New position when user (2) tool (2) takes effect
```

4, To let point position change with different user coordinate system, use U2U instruction to realize.

```
User( 2 )
MOVJ( Pn( 1 ) , speed , acc , dec , cp )      //User (2) no affect on this location
point abc = U2U( 2 , 0 , Pn( 1 ) )            //Switch Pn(1) into User 2 point position
MOVJ( abc ,speed , acc , dec , cp )           //User (2) being effected point position
```

Example: Refer to the above described

Reference: SetUser , Tool , SetTool , U2U

SetUser

SetUser(user_num, xpos, ypos, zpos, cpos)

Function:	Modify the data of the user number	
Parameter:	user_num	User number: 0~15
	xpos	User X absolute value
	ypos	User Y absolute value
	zpos	User Z absolute value
	cpos	User C absolute value
Return:	N/A	
Description:	User 0 is Robot DH modular corresponding World coordinate system, cannot modify.	
Example:	SetUser(1 , 10 ,10 ,0 ,0)	
	User(1)	
Reference:	User	

6.2.2 Control

Instruction Name	Function Introduction
Do...Loop	Looping execution the contents between Do and Loop unconditionally
Do While...Loop	When the condition is true, loop execution the contents between Do and Loop
Do Until...Loop	When condition is true, terminate loop execution the contents between Do and Loop, otherwise continuous looping execution
Exit Do	Direct exit current nesting level Do looping
Exit For	Direct exit current nesting level For looping
If...Then...Else...EndIf	Determine whether the conditional expression is true, and execute the content after the branch condition is true
For...Next	Repeat execution For...Next a series of statement per the assigned number of times
Switch...Case...EndSwitch	By enumerated variable value, when detecting one of Case value established, select execution of the corresponding instruction, no more detect after the execution is done
GoTo label	Jumps unconditionally from the current line to the line with the specified label number

Do

Do... ...Loop	
Function:	Looping execution the contents between Do and Loop unconditionally
Parameter:	N/A
Return:	N/A
Description:	1. When need to exit looping, use "Exit Do" statement to exit current level loop

Example:	Do
	//Loop execution
	Delay(1)
	Loop

Reference:	Exit Do , Delay , While , Until

Do

Do While condition... ...Loop	
Function:	When the condition is true, loop execution the contents between Do and Loop
Parameters:	condition Condition expression: as sta= = 1
Return:	N/A
Description:	1. When need to exit looping, use "Exit Do" statement to exit current level loop 2.It is recommended to add Delay(10) between the dead loops that are prone to not executing other instruction to prevent the machine from grinding to crash.

Example:	Do While 1 //Loop execution
	Delay(1)
	Loop

Reference:	Exit Do , Delay , While , Until

Do

Do Until condition... ...Loop	
Function:	When condition is true, terminate loop execution the contents between Do and Loop, otherwise continuous looping execution
Parameter:	condition Condition expression: as sta = = 1
Return:	N/A
Description:	1.When need to exit looping, use "Exit Do" statement to exit current level loop

Example:	Do Until sta==0 //Loop execution
	Delay(1)
	Loop

Reference:	Exit Do , Delay , While , Until

Exit

Exit Do	
Function:	Direct exit current nesting level Do looping
Parameter:	N/A
Return:	N/A
Description:	When need to exit Do looping, use "Exit Do" statement to exit current level loop

Example:	Do While 1 //Looping executed
	Delay(1)
	If sta ==1 Then
	Exit Do //When sta=1, exit current nesting level looping
	EndIf
	Loop

Reference:	Exit Do , Delay , While , Until

Exit

Exit For	
Function:	Direct exit current nesting level For looping
Parameter:	N/A
Return:	N/A
Description:	When need to exit For looping, use "Exit For" statement to exit current level loop

Example:	For i=0 To 15 Step 1 //Looping executed
	Delay(1)
	If sta ==1 Then
	Exit For //When sta=1, exit current nesting level looping
	EndIf
	Next

Reference:	For , To , Step , Next , Delay

If

If condition Then... ...Else... ...EndIf

Function: Determine whether the conditional expression is true, and execute the content after the branch condition is true

Parameter: condition expression: like sta == 1 , sta > 1 And sta < 3 , a < 5 Or b < 5

Return: N/A

Description:

1.When under Else situation and condition is true, execute between Then and Else, and when false, execute between Else and EndIf.

2.When there is no Else situation and the condition is true, execute between Then and EndIf, when false, exit determination.

3.Else disprove branch condition, [Default]not proceed the false condition.

4.ElseIf attach one condition, when If determine false, enter into this determination, [Default]not proceed the false condition.

5.This instruction can be checked only once, if it requires for repeated checking, needs to be embedded for use in loops.

6."==" means judgment and "=" means assignment. If user write "If a = 1 then ...", it will be judged as true every time. Pay attention to it when programming.

Example:

Case 1

```
If sta == 1 Then                    //Determine sta whether or not =1
statement1                          //When condition is true, execute statement 1 contents

Else
statement2                          //When condition is false, execute statement 2 contents
EndIf                               //Exit determine
```

Case 2

```
If sta == 1 Then                    //Determine sta whether or not =1
statement1                          //When condition is true, execute statement 1 contents
ElseIf sta == 2 Then                //Determine sta whether or not =2
statement2                          //When condition is true, execute statement 1 contents
ElseIf ...  Then                    //Determine sta = other
...
EndIf                               //Exit determine
```

Reference: And , Or , Not , Exit , GoTo ,Do , If

For

For i=d1 To d2... ...Next

Function: Repeat execution For...Next a series of statement per the assigned number of times

Parameter:

i	Numeric type variable
d1	Starting value
d2	Ending value
Step	1 Each loop variable increase or decrease value, [Default] mode, i+1

Return: N/A

Description: The next statement of the For statement (d1 to d2 Step x...compare d1+x with d2) is executed after the "Next" command is reached. The counter variable (variable name) is incremented only by the value specified by the increment value (Step increment value). If no increment value is set, the counter increases by "1" each time. The counter variable (variable name) is compared to the final value. If the counter variable is less than or equal to the final value, the next statement of the "For" command is re-executed. If the counter variable (variable name) is greater than the final value, execution branches to For... Outside the Next loop, proceed to the Next command of the "Next" command.

Example:

```
Case 1
For i = 1 To 10
MOVJ( Pn(i) ,speed , acc , dec , cp )      //Movement from the current position to
                                           position in the proper order of 1~10#  in the
                                           point table
Next
Case 2
For i = 10 To 1 Step                       -1
MOVJ( Pn(i) ,speed , acc , dec , cp )      // Movement from the current position to
                                           position in the proper order of 10~1#  in the
                                           point table
Next
Case 3
int n = 0
For n=0 To 15
Close(n)                                   //Turn off the output of ports 0 ~ 15 in turn
Next
```

Reference: Do...Loop

Switch

Switch var...Case d1... ...Default... ...EndSwitch

Function: By enumerated variable value, when detecting one of Case value established, select execution of the corresponding instruction, no more detect after the execution is done.

Parameter:

var	Numerical type or text string type variable
d1	Determine variable whether or not equals to enumerated value 1
d2	Determine variable whether or not equals to enumerated value 2
...	...
Default	If the above enumerated values are not met, execute this operation. [Default] If the above values listed are not met, no operation is executed.

Return: N/A

Description: The same as If statement, this instruction can be checked only once, if it requires for repeated checking, needs to be embedded for use in loops.

Example:

```
Switch var
Case d1                    //var=d1
statement1                 //Condition established exit after execution
Case d2                    //var=d2
statement2                 //Condition established exit after execution
Case ...
...
Default                    //var not equal to above listed value
statement_default          //Condition established exit after execution
EndSwitch
```

Reference: If...Then...Else

GoTo

GoTo label	
Function:	Jumps unconditionally from the current line to the line with the specified label number
Parameter:	label Statement to jump here
Return:	N/A
Description:	1.Before using GoTo statement, labels need to be declared in advance. 2.Label location can be front or back. 3.Label can be defined, but need to add ":"back.
Example:	... label: //Destination label ... ... GoTo label //Jumping to destination label
Reference:	Do , For , If , Switch

6.2.3 Point operation

Instruction Name	Function Introduction
Pn	Index data from point table
Where	Return the position of TCP
Matrix	Build a parallelogram (including a rectangle) array of points per the provided reference point, the total number of points equal to the number of points in the 1st direction X the number of points in the 2nd direction
Matrix	Reference an array point with the specified array number
U2U	Convert a position point in the source coordinate system to a coordinate point in the target coordinate system
BuildPoint	Build a point position data
BuildJPoint	Build a point position data (joint type)
BuildX	Change coordinate X based on the reference point, return a new position point
BuildY	Change coordinate Y based on the reference point, return a new position point
BuildZ	Change coordinate Z based on the reference point, return a new position point
BuildC	Change coordinate C based on the reference point, return a new position point
GetPointX	Return X coordinate component data from a point
GetPointY	Return Y coordinate component data from a point
GetPointZ	Return Z coordinate component data from a point
GetPointC	Return C coordinate component data from a point
GetPointJ1	Return J1 axis angle data from a point
GetPointJ2	Return J2 axis angle data from a point
GetPointJ3	Return J3 axis height data from a point
GetPointJ4	Return J4 axis angle data from a point

Pn

Pn(num)	
Function:	Index data from point table
Parameter:	num Point data index range 0~9999
Return:	Point position data
Description:	1.The datas have 2 forms in the point position table, joint or cartesian type. 2.When the referenced joint type data is used directly by the motion command, ignore the "User( ) and Tool( )" instruction above. 3.When the referenced cartesian type data is used directly by the motion command, ignore the User() but execute Tool() instruction. 4.When the location points are built from the customer BuildPoint() instruction, execute both User () and Tool () reference system as specified. 5.Pn(0) presents robot mechanical zero, other point position is for teaching.
Example:	point pt_new=Pn(1) //Declare the point type variable pT_new, and then assign the point table number P00001 to pt_new
Reference:	MOVJ , MOVL , MOVC , User , Tool , BuildPoint , Tool ,User , U2U

Where

Where( )	
Function:	Return the position of TCP
Parameter:	N/A
Return:	Point position data
Description:	1.Defalut return world coordinate system cartesian style. 2.Use .X .Y .Z .C or .J1 .J2 .J3 .J4 take the coordinate value of this point 3.In order to prevent the instruction from being executed in advance and the robot position has not arrived when obtaining information, add a "break()" statement in the previous sentence to avoid this situation;
Example:	pt_now=Where() //Assign current position to pt_now
Reference:	.x , .y , .z , .c , GetPointX , GetPointY , GetPointZ , GetPointC , Break

Matrix

Matrix(1, home_pos, dir1_pos, dir2_pos, row, col)

Function: Build a parallelogram (including a rectangle) array of points per the provided reference point, the total number of points equal to the number of points in the 1st direction X the number of points in the 2nd direction

Parameter:

1	Array number: 1~200
home_pos	Array starting point
dir1_pos	1st direction end
dir2_pos	2nd direction end
row	1st direction total points number>=1, row and col cannot be as 1 simultaneously
col	2nd direction total points number>=1, row and col cannot be as 1 simultaneously

Return: N/A

Description: Allow to build space quadrilateral, that is: the point plane is not necessarily parallel to the XY plane

Example:

```
/*------Copy below Gi Gj declare into the global variable-------
int Gi=0, Gj=0
-------------------------------------------------------------*/
int speed=100,acc=100,dec=100,cp=0          //Define motion parameter
int m=4, n=3, i=0, j=0                      //Set direction 1 has m pcs, direct 2 has n pcs
point org=BuildPoint(300.000, 000.000, -    //Setting org point
020.000, 000.000, 0)
point pt1=BuildPoint(300.000, 300.000, -    //Setting direction 1 destination
020.000, 000.000, 0)
point pt2=BuildPoint(400.000, 000.000, -    //Setting direction 2 destination
020.000, 000.000, 0)
Matrix(1,org,pt1,pt2,m,n)                   //Build m x n code 1 array based on the
                                            reference point
Do
For i = Gi To m-1
Gi = i                                      //Save current value to global variable Gi
For j = Gj To n-1
```

```
Gj = j                                          //Save current value to global variable Gj
float h=5, zlim=10, z=25
/*
MOVJ(Matrix(1,i,j)+Z(z,0),speed,acc,dec,cp)     //Successively movement to 1# matrix point +z
MOVJ(Matrix(1,i,j),speed,acc,dec,cp)            //Successively movement to 1# matrix point
MOVJ(Matrix(1,i,j)+Z(z,0),speed,acc,dec,cp)     //Successively movement to 1# matrix point +z
*/
JUMP(Matrix(1,i,j),h,h,zlim,speed,acc,dec,cp)   //Successively movement to 1# matrix point
Break()
Next
Gj = 0                                          //Loop ending, reset variable to initial value
Next
Gi = 0                                          //Loop ending, reset variable to initial value
Loop
```

Reference: BuildX , BuildY , BuildZ , BuildC

Matrix

Matrix(1, i, j)		
Function:	Reference an array point with the specified array number	
Parameter:	1	Array number: 1~200
	i	From direct 1 serial number, begins with 0
	j	From direct 2 serial number, begins with 0
Return:	Point position data	
Description:	Serial number begins with 0, until total number value minus 1 ends	
Example:	Details refer to Matrix format 1	
Reference:	BuildX , BuildY , BuildZ , BuildC ,MOVJ , MOVL	

U2U

U2U(target_user_num, source_user_num, point_a)

Function: Convert a position point in the source coordinate system to a coordinate point in the target coordinate system

Parameters:

target_user_num	Target User number: 0~15
source_user_num	Source user number: 0~15
point_a	Location point under Source user number

Return: Point position data

Description: 1.Points created by BuildPoint default to points in the world coordinate system, corresponding source_user_num is 0 .

2.Joint point position in the point position table, corresponding source_user_num is 0.

Example: pt_new=U2U(1,2,point_a) //Convert coordinate 2 point_a into coordinate 1 point position value, and assign pt_new.

Reference: User , BuildPoint , Pn

BuildPoint

BuildPoint(xpos, ypos, zpos, cpos, hand)		
Function:	Build a point position data	
Parameter:	xpos	Build point X coordinate
	ypos	Build point Ycoordinate
	zpos	Build point Z coordinate
	cpos	Build point C coordinate
	hand	Left hand 0, right hand 1
Return:	Point position data	
Description:	Point data created in the program does not contain the user coordinate system option,	can be set by the User instruction, if User is not specified, the default is world coordinates, that is: User(0).
Example:	pt_new=BuildPoint(100,110,120,130,0)	//Build point pt_new (xyzc = 100,110,120,130, left hand) and assign the result value to pt_new
Reference:	MOVJ , MOVL , MOVC , Pn , User , Tool	

BuildJPoint

BuildJPoint(j1, j2, j3, j4)

Function:	Build a point position data (joint type)	
Parameter:	j1	Build point j1 coordinate
	j2	Build point j2 coordinate
	j3	Build point j3 coordinate
	j4	Build point j4 coordinate
Return:	Point position data (Joint type)	
Description:	This point position is not affected by program User,Tool instruction.	
Example:	pt_new=BuildJPoint(-40,50,3,20)	//Build joint coordinate J1, J2, J3, J4 to -40°,50°,3mm,20°.
Reference:	MOVJ , MOVL , MOVC , Pn , User , Tool	

BuildX

BuildX(point_a, 0, dis)		
Function:	Change coordinate X based on the reference point, return a new position point	
Parameter:	point_a	Reference point
	0	Relative relation (1, direct alternate)
	dis	The value in the X direction relative to the point to be converted, unit mm
Return;	Point position data	
Description:	This instruction has the same effect on Point+X (dis , 0/1)	
Example:	point_a_new=BuildX(point_a , 0 , 25)	//Increase the value of point_a by 25mm in the positive X direction and assign the result value to pt_new
Reference:	MOVJ , MOVL , MOVC ,Pn , User , Tool	

BuildY

BuildY(point_a, 0, dis)		
Function:	Change coordinate Y based on the reference point, return a new position point	
Parameter:	point_a	Reference point
	0	Relative relation (1, direct alternate)
	dis	The value in the Y direction relative to the point to be converted, unit mm
Return;	Point position data	
Description:	This instruction has the same effect on Point+Y (dis , 0/1)	
Example:	point_a_new=BuildY(point_a , 0 , 25)	//Increase the value of point_a by 25mm in the positive Y direction and assign the result value to pt_new
Reference:	MOVJ , MOVL , MOVC ,Pn , User , Tool	

BuildZ

BuildZ(point_a, 0, dis)		
Function:	Change coordinate Z based on the reference point, return a new position point	
Parameter:	point_a	Reference point
	0	Relative relation (1, direct alternate)
	dis	The value in the Z direction relative to the point to be converted, unit mm
Return;	Point position data	
Description:	This instruction has the same effect on Point+Z (dis , 0/1)	
Example:	point_a_new=BuildZ(point_a , 0 , 25)	//Increase the value of point_a by 25mm in the positive Z direction and assign the result value to pt_new
Reference:	MOVJ , MOVL , MOVC ,Pn , User , Tool	

BuildC

BuildC(point_a, 0, dis)		
Function:	Change coordinate C based on the reference point, return a new position point	
Parameter:	point_a	Reference point
	0	Relative relation (1, direct alternate)
	dis	The value in the C direction relative to the point to be converted, unit mm
Return;	Point position data	
Description:	This instruction has the same effect on Point+C (dis , 0/1)	
Example:	point_a_new=BuildC(point_a , 0 , 25)	//Increase the value of point_a by 25mm in the positive C direction and assign the result value to pt_new
Reference:	MOVJ , MOVL , MOVC ,Pn , User , Tool	

GetPointX

GetPointX(point_a)	
Function:	Return X coordinate component data from a point
Parameter:	point_a point type
Return:	X coordinate component of point position data
Description:	1.This instruction has the same effect on point_a.x 2.Note that the received value to be defined as a float data type
Example:	ax=GetPointX(point_a) // Return X component of point_a, assign the result value to ax variable
Reference:	.x , .y , .z , .c , Where

GetPointY

GetPointY(point_a)	
Function:	Return Y coordinate component data from a point
Parameter:	point_a point type
Return:	Y coordinate component of point position data
Description:	1.This instruction has the same effect on point_a.y 2.Note that the received value to be defined as a float data type
Example:	ay=GetPointY(point_a) // Return Y component of point_a, assign the result value to ay variable
Reference:	.x , .y , .z , .c , Where

GetPointZ

GetPointZ(point_a)	
Function:	Return Z coordinate component data from a point
Parameter:	point_a point type
Return:	Z coordinate component of point position data
Description:	1.This instruction has the same effect on point_a.z 2.Note that the received value to be defined as a float data type
Example:	az=GetPointZ(point_a) // Return Z component of point_a, assign the result value to az variable
Reference:	.x , .y , .z , .c , Where

GetPointC

GetPointC(point_a)	
Function:	Return C coordinate component data from a point
Parameter:	point_a point type
Return:	C coordinate component of point position data
Description:	1.This instruction has the same effect on point_a.c 2.Note that the received value to be defined as a float data type 3.Unit is angle system.
Example:	ac=GetPointC(point_a) // Return C component of point_a, assign the result value to ac variable
Reference:	.x , .y , .z , .c , Where

GetPointJ1

GetPointJ1(point_a)

Function: Return J1 axis angle data from a point

Parameter: point_a point type

Return: J1 axis angle of point position data

Description: 1.This instruction has the same effect on point_a.j1.

2.Note that the received value to be defined as a float data type.

3. Returning value is angle system.

Example: Case 1

```
float  jointpos=GetPointJ1(point_a)        //Return J1 axis angle data from point_a, assign the result value to "jointpos".
```

Case 2

```
float jointpos=point_a.j1                  //Same as the above example.
```

Reference: .j1 , .j2 , .j3 , .j4 , Where

GetPointJ2

GetPointJ2(point_a)	
Function:	Return J2 axis angle data from a point
Parameter:	point_a point type
Return:	J2 axis angle of point position data
Description:	1.This instruction has the same effect on point_a.j2. 2.Note that the received value to be defined as a float data type. 3. Returning value is angle system.
Example:	Case 1 float jointpos=GetPointJ2(point_a) //Return J2 axis angle data from point_a, assign the result value to "jointpos". Case 2 float jointpos=point_a.j2 //Same as the above example.
Reference:	.j1 , .j2 , .j3 , .j4 , Where

GetPointJ3

GetPointJ3(point_a)

Function: Return J3 axis height data from a point

Parameter: point_a point type

Return: J3 axis angle of point position data

Description: 1.This instruction has the same effect on point_a.j3.

2.Note that the received value to be defined as a float data type.

3. Returning value unit is mm.

Example: Case 1

```
float  jointpos=GetPointJ3(point_a)        //Return J3 axis height data from point_a, assign the result value to "jointpos".
```

Case 2

```
float jointpos=point_a.j3                  //Same as the above example.
```

Reference: .j1 , .j2 , .j3 , .j4 , Where

GetPointJ4

GetPointJ4(point_a)	
Function:	Return J4 axis angle data from a point
Parameter:	point_a point type
Return:	J4 axis angle of point position data
Description:	1.This instruction has the same effect on point_a.j4. 2.Note that the received value to be defined as a float data type. 3. Returning value is angle system.
Example:	Case 1 float jointpos=GetPointJ4(point_a) //Return J4 axis angle data from point_a, assign the result value to "jointpos". Case 2 float jointpos=point_a.j4 //Same as the above example.
Reference:	.j1 , .j2 , .j3 , .j4 , Where

6.2.4 System

Instruction Name	Function Introduction
SysTime	Return system present time, unit ms
Pause	System pause
Delay	Wait for some period then to continue the moving
Print	Print information
Break	After the execution of the previous motion instruction is completed, then execute the instruction after Break, to prevent the next non-IO or Delay instruction from being executed in advance
Power	Calculate the value of a number to the N-th power
Sqrt	Calculate the square root of a number
Abs	Calculate the absolute value of a number
Sin	Calculate Sine of an Angle
Cos	Calculate Cosine of an angle
Tan	Calculate Tangent of an Angle
Asin	Calculate Arcsine of a value and return the
Acos	Calculate inverse cosine of a value and returns the corresponding Angle value
Atan	Calculate inverse tangent of a value and returns the corresponding Angle value
RTA	Convert radians to angles
ATR	Convert angles to radians

SysTime

SysTime( )	
Function:	Return system present time, unit ms
Parameter:	N/A
Return:	A value in ms unit
Description:	The general usage is that the time difference of consumption is obtained by subtracting the time obtained successively

Example:	float old_time = SysTime() //Get current time
	...
	// Get the current time again and subtract the previous time to get the elapsed time between the two
	Print("Program cost time",(SysTime()-old_time)/1000,"s")

Reference:	Print

Pause

Pause( )	
Function:	System pause
Parameter:	N/A
Return:	N/A
Description:	Program is being paused, to continue the execution through keypad or external command

Example:	MOVJ(put , speed , acc , dec , cp)
	Open (3)
	Delay(300)
	Close (3)
	Pause() //When the program runs to this line, pause execution .
	MOVJ(put_up , speed , acc , dec , cp)

Reference:	MOVJ , MOVL , MOVC , Open , Close , Pulse

Delay

Delay(time)	
Function:	Wait for some period then to continue the moving
Parameter:	time To be waited time, unit ms
Return:	N/A
Description:	1This instruction followed by the motion instruction would interrupt the continuous motion here 2.Insert Delay(0) in an infinite loop to prevent program execution from getting stuck
Example:	Delay(300) //When the program runs at this point, wait 300ms before moving down
Reference:	MOVJ , MOVL , MOVC , Do...Loop , Break

Print

Print("print_string")

Function: Print information

Parameter: "print_string" Content to be printed

Return: N/A

Description: Printing contents include all types, use "," to separate datas

Example: Print("abc=",abc) //Printing result, abc=3

Reference: StrFormat , SysTime

Break

Break( )	
Function:	After the execution of the previous motion instruction is completed, then execute the instruction after Break, to prevent the next non-IO or Delay instruction from being executed in advance
Parameter:	N/A
Return:	N/A
Description:	1.This instruction followed by the motion instruction would interrupt the continuous motion here. 2.This applies to the scenario where the following operations can be performed only after a certain point is completely reached(at least in logical). 3.No need to add Break() ,when Delay or IO operation behind motion.

Example:	MOVJ(put , speed , acc , dec , cp)
	Break ( )
	gla = gla +1 //The variable gla is an engineering variable, which involves other execution in the background. It can be calculated only after it is in place.
	Open (3)
	Delay(300)
	Close (3)

Reference:	MOVJ , MOVL , MOVC , Open , Close , Pulse

Power

Power(num, pow_n)		
Function:	Calculate the value of a number to the N-th power	
Parameter:	num	base number
	pow_n	index
Return:	N-th power result	
Description:	Be careful to define reasonable received values, typically of type float	
Example:	float rtn = Power(2,3)	//Returning value 8.000000
	Abs(Power(2 , 0.5))	//Returning value is the absolute value of 2 squared
Reference:	Abs , Sqrt	

Sqrt

Sqrt(num)	
Function:	Calculate the square root of a number
Parameter:	num The number of being requested
Return:	Square root result, take positive value
Description:	Be careful to define reasonable received values, typically of type float
Example:	float rtn = Sqrt(4) //Returning value is 2.000000
Reference:	Power

Abs

Abs(abs_num)		
Function:	Calculate the absolute value of a number	
Parameter:	abs_num	The number of being requested
Return:	Seek the result value after absolute value	
Description:	Note that the receiving unit is defined as the appropriate type int or float	
Example:	Abs(-1.23)	//Result is 1.23
Reference:	Power	

Sin

Sin(degree)		
Function:	Calculate Sine of an Angle	
Parameter:	degree	Angular value
Return:	Sin value	it is float type
Description:	Note that the receiving unit is typically defined as a float of the appropriate type	
Example:	Sin(30)	//Result is 0.5
Reference:	Cos , Tan , Asin , Acos , Atan	

Cos

Cos(degree)		
Function:	Calculate Cosine of an angle	
Parameter:	degree	angular value
Return:	Cosine value	it is float type
Description:	Note that the receiving unit is typically defined as a float of the appropriate type	
Example:	Cos(60)	//Result is 0.5
Reference:	Sin , Tan , Asin , Acos , Atan	

Tan

Tan(degree)		
Function:	Calculate Tangent of an Angle	
Parameter:	degree	Angular unit, floating point type
Return:	Tangent value	it is float type
Description:	Note that the receiving unit is typically defined as a float of the appropriate type	
Example:	Tan(45)	//Result is 1.0
Reference:	Sin , Cos , Asin , Acos , Atan	

Asin

Asin(value)		
Function:	Calculate Arcsine of a value and return the	
Parameter:	value	To be calculated value
Return:	Corresponding angle of Sin, angular unit,	it is float type
Description:	Note that the receiving unit is typically defined as a float of the appropriate type	
Example:	Asin(0.5)	//Result is 30
Reference:	Sin , Cos, Tan , Acos , Atan	

Acos

Acos(value)	
Function:	Calculate inverse cosine of a value and returns the corresponding Angle value
Parameter:	value To be calculated value
Return:	Corresponding angle of Cosine, angular unit, it is float type
Description:	Note that the receiving unit is typically defined as a float of the appropriate type
Example:	Acos(0.5) //Result is 60
Reference:	Sin , Cos, Tan , Asin , Atan

Atan

Atan(value)	
Function:	Calculate inverse tangent of a value and returns the corresponding Angle value
Parameter:	value To be calculated value
Return:	Corresponding angle of Tangent, angular unit, it is float type
Description:	Note that the receiving unit is typically defined as a float of the appropriate type
Example:	Atan(1) //Result is 45
Reference:	Sin , Cos, Tan , Asin , Acos

RTA

RTA(value)	
Function:	Convert radians to angles
Parameter:	value To be calculated value (Radian)
Return:	Converted angular value, it is float type
Description:	Note that the receiving unit is typically defined as a float of the appropriate type The system default internal processing is Angle value, and this function is used when the data connected to the external system is in radians.
Example:	float rtn=RTA(3.1415926) //Result is 180
Reference:	ATR , Sin , Cos, Tan , Asin , Acos , Atan

ATR

ATR(value)	
Function:	Convert angles to radians
Parameter:	value To be calculated value (Angle)
Return:	Converted Radian value, it is float type
Description:	Note that the receiving unit is typically defined as a float of the appropriate type he system default internal processing is Angle value, and this function is used when the data connected to the external system is in radians.
Example:	float rtn=ATR(180) //Result is 3.141593
Reference:	RTA , Sin , Cos, Tan , Asin , Acos , Atan

6.2.5 I/O

Instruction Name	Function Introduction
Open	Open a port, and remain output status until it's closed
Close	Close a port output
Pulse	Make a terminal output at the specified time, then shut off the terminal
ReadDI	Reads a number of port numbers and returns the status value of the input value, encoded in decimal bits
ReadDO	Reads a number of port numbers and returns the status value of the output value, encoded in decimal bits
WaitDI	Read input terminal signal status and determine the condition is true, if it's true then move down to execute, or else always determine the input signal
WaitDI	Read input terminal signal status and determine the condition is true at the specified time, when it's true then move down to execute, or else wait for the timing to complete before executing down, and return to corresponding value

Open

Open(port_num)

Function: Open a port, and remain output status until it's closed

Parameter: port_num Port number: 0~31

Return: N/A

Description: Open a terminal for a while then close this terminal, if no need to wait for then use Pulse to alternate;

Open multiple terminal, use ","to separate, as Open (1,2,3); Currently maximum Qty of terminal you can open is 16 pcs one time;

Example:

```
Case 1
For i = 0 To 15                    //Open local 0~15 output terminal
Open(i)
Next
Case 2
Open( 1 )                          //Open local 1# output terminal
Delay( 1000 )
Close( 1 )                         //Close local 1# output terminal
Delay( 1000 )
```

Reference: Close , Pulse , WaitDI , ReadDI , ReadDO , Delay

Close

Close(port_num)

Function: Close a port output

Parameter: port_num Port number: 0~31

Return: N/A

Description: Open a terminal for a while then close this terminal, if no need to wait for , use Pulse to alternate;

Close multiple terminal, use ","to separate, as Close (1,2,3); Currently maximum Qty of terminal you can close is 16 pcs one time;

Example: Case 1

```
For i = 0 To 15                    //Close local 0~15 output terminal
Close(i)
Next
Case 2
Open( 1 )                          //Open local 1# output terminal
Delay( 1000 )
Close( 1 )                         //Close local 1# output terminal
Delay( 1000 )
```

Reference: Open , Pulse , WaitDI , ReadDI , ReadDO , Delay

Pulse

Pulse(port_num, time)

Function: Make a terminal output at the specified time, then shut off the terminal

Parameter:

port_num	Local output terminal number 0~31
time	Output holding time, unit ms

Return: N/A

Description: The output time of this instruction overlaps the time consumed by subsequent instruction until the output is completed

In the last sentence or in the execution of switching mode, pause, etc., the signal will be executed according to the set time

Example: MOVJ(Pn(1) , speed , acc , dec , cp)

Pulse(3 , 500)

MOVJ(Pn(2) , speed , acc , dec , cp)

Case explanation, Movement to Pn(1) and end, open 3# output for 500ms then close, open Pn(2) point position motion simultaneously

Remarks, Smooth transition between Pn(1) and Pn(2) of the above execution will be interrupted. If smooth trajectory is required, use parallel instruction

Reference: Open , Close , WaitDI , ReadDI , ReadDO , Delay , Dn

ReadDI

ReadDI(port_num)

Function: Reads a number of port numbers and returns the status value of the input value, encoded in decimal bits

Parameter: port_num If there are too many parameters, use ", " to separate them.

Return: Value

Description: 1.Encoding order Codes in the order in which they appear in parentheses, the ones that come first are in the low order.

2.Input terminals can be different from the order of 0,1,2,3... like it can be 3,1,2,0.

3.Support maximum 20 input point encoder.

Example:

```
int a = ReadDI(0,1,2,3)
Switch a
Case $b1010  ...                    //Binary notation for the constant 10, it can be also written as Case 10, and at this point the corresponding N0,IN1,IN2,IN3 input status is OFF,ON,OFF,ON respectively
EndSwitch
If ReadDI(1) == 0 Then              //if IN1 input status is OFF
...
EndIf
```

Reference: Open , Close , Pulse , WaitDI , ReadDO , Delay

ReadDO

ReadDO(port_num)

Function: Reads a number of port numbers and returns the status value of the output value, encoded in decimal bits

Parameter: port_num If there are too many parameters, use ", " to separate them.

Return: Value

Description: 1.Encoding order Codes in the order in which they appear in parentheses, the ones that come first are in the low order.

2.Output terminals can be different from the order of 0,1,2,3... like it can be 3,1,2,0.

3.Support maximum 20 output point encoder.

Example:

```
int a = ReadDO(0,1,2,3)
Switch a
Case $b1001                    //Binary notation for the constant 9, it can be also written as Case 9, and at this point the corresponding OUT0,OUT1,OUT2,OUT3 output status is  ON,OFF,OFF,ON respectively.
...
EndSwitch
If ReadDO(1) == 0 Then         //If OUT1 output status is OFF
...
EndIf
```

Reference: Open , Close , Pulse , WaitDI , ReadDI , Delay

WaitDI

WaitDI(port_num, 1)

Function: Read input terminal signal status and determine the condition is true, if it's true then move down to execute, or else always determine the input signal

Parameter:

port_num	Input port number 0~31
1	Status: 0, OFF 1, ON

Return: N/A

Description: If signal status never meet the conditions, it keeps on waiting, if wait for a while only, refer to other format

Example: ...

```
WaitDI(10, 1)        //Wait for input terminal number 10 ON
WaitDI(8, 0)         //Wait for input terminal number 8 OFF
```

Reference: Open , Close , Pulse , ReadDI , ReadDO , Delay

WaitDI

WaitDI(port_num, 1, time)

Function:	Read input terminal signal status and determine the condition is true at the specified time, when it's true then move down to execute, or else wait for the timing to complete before executing down, and return to corresponding value	
Parameter:	port_num	Input port number 0~31
	1	Switch status: 0,OFF 1, ON
	time	waiting time
Return:	0	Received signal within time
	1	Still no signal when time is up
Description:	Increase the waiting time to avoid the application waiting forever	
	Do not set waiting time too long, increase processing of returned values in time	
Example:	int a = WaitDI(4,1,1000)	//Program execution to this line wait for output 4# terminal ON, if ON immediately (within <1000ms), then program move down execution at once, a=0; if exceeding 1000ms still OFF, then program move down execution as well, a=1.
Reference:	Open , Close , Pulse , ReadDI , ReadDO , Delay	

6.2.6 Communication

Instruction Name	Function Introduction
OpenCom	Open serial port
CloseCom	Close serial port
ReadCom	Return the ASCII data read by the serial port(string) per the terminal number
WriteCom	Write data to serial port
CheckCom	Return the length of the string or number of bytes received by the serial port
ClrCom	Clear serial port receive buffer
SetCom	setting serial port info
OpenNet	Open network port
CloseNet	Close network port
ReadNet	Return ASCII data read by network port per channel number (string)
WriteNet	Write data to network port
CheckNet	Return the string length or number of bytes received by the network port
ClrNet	Clear network port receive buffer
SetNet	Set network port info
MIRead	Read Int type modbus address corresponding value
MFRead	Read float type modbus address corresponding value
MIWrite	Write Int type modbus address corresponding value
MFWrite	Write float type modbus address corresponding value
GIRead	Read Int type global variable address corresponding value
GFRead	Read float type global variable address corresponding value
GIWrite	Write into Int type global address corresponding
GFWrite	Write into float type modbus address corresponding value

OpenCom

OpenCom(com_num)		
Function:	Open serial port	
Parameter:	com_num	Terminal number: 1-2
Return:	-2	Parameter error
	1	Open success
Description:	Communication parameter setting reference communication---serial port setting	

Example:

```
int sn = 1                          //com number range (1-2)
int BaudRate=115200                 //Baud rate/*115200,57600,38400,19200,9600,4800,2400,1200*/
int Databit=8                       //data bit  8
int Stopbit=1                       //stop bit 1
string Parity="N"                   //Parity selection("N"Parity none)
int res=0
string read_data
SetCom( sn, BaudRate, Databit, Stopbit, Parity)
OpenCom(sn)
Do
/*
res = -2 parameter error
res = -1 serial port not open
*/
res = CheckCom(sn)
If res == -2 Then
Print("parameter error")
ElseIf res == -1 Then
Print("no running opencom")
ElseIf res > 0 Then                 //If send and receive in hexadecimal, change below 0 into 2
read_data = ReadCom(sn , 0)         //Read serial port sn data and save into read_data variable
```

```
WriteCom(sn , 0 ,read_data)          //Send read_data through channel sn
                                     //Print(read_data)
EndIf
Delay(5)
Loop
```

Reference: CloseCom , SetCom

CloseCom

CloseCom(com_num)		
Function:	Close serial port	
Parameter:	com_num	Terminal number: 1-2
Return:	-2	Parameter error
	1	Close success
Description:	N/A	
Example:	CloseCom(1)	//Close 1# serial port
	Operation details refer to OpenCom	
Reference:	OpenCom	

ReadCom

ReadCom(com_num)

Function: Return the ASCII data read by the serial port(string) per the terminal number

Parameter: com_num Terminal number: 1-2

Return: string read from serial port

Description: N/A

Example: recvstr = ReadCom(1) //string read from serial port 1#.

Operation details refer to OpenCom

Reference: WriteCom

ReadCom(com_num, code)

Function: Return the ASCII data read from the serial port(ASCII,Octal,Hex format can be choice)

Parameter: com_num 1-2 Terminal number:1-2

code Encode format: 0, stands for ASCII ,1, stands for Octal, 2,stands for Hex

Return: string read from serial port(According to code value)

Description: 1. ReadCom(1,0) equals to ReadCom(1), means receive ASCII data

2. When receive Octal,Hex byte value, return string, as: Sender send0x00,0x1F,0x01 by Hex, then recvstr="00 1F 01"

3.Confirm the sending format of the sender when using, if the receive format is not matched which could cause display showed garbled or receive datas out of the agreement

Example: string recvstr = ReadCom(1,2) //Read 1# serial port received Hex data

string recvstr = ReadCom(2,0) //Read 2# serial port received ASCII data

Operation details refer to OpenCom

Reference: WriteCom

WriteCom

WriteCom(com_num, str_write)		
Function:	Write data to serial port	
Parameter:	com_num	Terminal number: 1-2
	str_write	string data to be sent
Return:	-2	parameter error
	0	data-sending success
	1	data-sending failure
Description:	N/A	
Example:	string sendstr = "ABCDEFG"	//Define sending contents
	WriteCom(1, sendstr)	//Send “ABCDEFG” string to 1# serial port
	Operation details refer to OpenCom	
Reference:	ReadCom	

WriteCom(com_num, code, str_write)

Function:	Write data to serial port based on different encoder format	
Parameter:	com_num	Terminal number: 1-2
	code	Encoder format: 0, stands for ASCII ,1, stands for Octal, 2, stands for Hex
	str_write	String data to be sent
Return:	-2	parameter error
	0	data-sending success
	1	data-sending failure

Description: 1.WriteCom(1, 0 , str_write) equals to WriteCom(1,str_write) , means write data WriteCom to COM 1# terminal in ASCII format

2.Before writing hexadecimal data, convert the hexadecimal data to a byte string and send it as a string. If the format is not correct, the format is wrong will be alarmed.

Example:

```
string sendstr = "ABCDEFG"                //Define sending contents
WriteCom(1 , 0 , sendstr)                 //Send "ABCDEFG"string to 1# serial port
string str_write = "05 1F 3D 42"          //Define string content as Hex format
WriteCom(1 , 2 , str_write)               //Write 0x05 0x1F 0x3D 0x42 data to 1# serial
                                          port in Hex format
```

Operation details refer to OpenCom

Reference: ReadCom

CheckCom

CheckCom(com_num)		
Function:	Return the length of the string or number of bytes received by the serial port	
Parameter:	com_num	Terminal number: 1-2
Return:	>0	Return string length or bytes numbers
	-1	Serial port not open
	-2	Serial port is being occupied by others
Description:	1. like: Sender sends 0x00,0x1F,0x01 in Hex format, then the returning value is 3, means total 3 bytes	
	2.like: Sender sends “http” in ASCII code, then the returning value is 4, means string length is 4	
Example:	int chk= 0	//Define received numbers
	chk = CheckCom(1)	//Check received numbers, further process the corresponding content based on the received quantity
	Operation details refer to OpenCom	
Reference:	ReadCom , WriteCom	

ClrCom

ClrCom(com_num)		
Function:	Clear serial port receive buffer	
Parameter:	com_num	Terminal number: 1-2
Return:	-2	parameter error
Description:	N/A	
Example:	ClrCom(1)	//Clear 1# channel buffer
Reference:	CheckCom , ReadCom	

SetCom

SetCom(com_num, baud_rate, data_bit, stop_bit, parity)		
Function:	setting serial port info	
Parameter:	com_num	Terminal number(numerical type): 1~2
	baud_rate	Baud rate(numerical type): 115200,57600,38400,19200,9600,4800,2400,1200
	data_bit	Data bit (numerical type): 5 6 7 8
	stop_bit	Stop bit (numerical type): 1 2
	parity	Parity(string type): Specify "O" for odd numbers, Specify "E" for even numbers,Specify "N" if no parity
Return:	-2	Parameter error
	1	Setting success
Description:	N/A	
Example:	SetCom(1 , 115200 , 8 , 1 , "N")	//Setting 1# channel serial port
	Operation details refer to OpenCom	
Reference:	OpenCom , CloseCom	

OpenNet

OpenNet(net_num)

Function: Open network port

Parameter: net_num Channel: 1~4

Return: -2 parameter error

1 Open success

Description: Setting on the communication interface or manual operation, actual results are subject to instruction execution

Example: Case 1

```
                                        //Set local sn channel as the client
string Server_Ip="192.168.1.150"        //Server IP
int Server_port=5000                    //Server port
int sn=1                                //Connection channel

int res=0
string read_data                        //Read data
SetNet(1,sn,Server_Ip,Server_port)      //Set sn channel as the client
OpenNet(sn)
Do
/*
res=-5  Initialization
res=-4  network disconnected
res=-3  non-execution opennet(sn)
res=-2  parameter error
res=-1  network open, but no connection was established with the other party
res>=0  data length
*/
res = CheckNet(sn)
If res == -5 Then
Print("Ethernet restarting ")
ElseIf res == -4 Then
```

```
Print("line out")
ElseIf res == -3 Then
Print(" no running opennet funciton")
ElseIf res == -2 Then
Print("parameter error")
ElseIf res == -1 Then
Print("no ESTABLISHED")
ElseIf res > 0 Then                    //If send and receive in Hex, change below 0 to 2
read_data=ReadNet(sn,0)                //Save data to read_data
Print(read_data)
WriteNet(sn,0,read_data)               //Send read_data variable saved data to channel
                                       sn
EndIf
Delay(5)
Loop

Case 2
                                       //Set channel sn as service
string Local_Ip="192.168.1.80"         //Local IP address
int Local_port=5000                    //Local port
int sn=1                               //Channel
int res=0
string read_data                       //Read data
SetNet(0,sn,Local_Ip,Local_port)       //Set channel sn as local service end
OpenNet(sn)
Do
/*
res=-5   Initialization
res=-4   Network disconnected
res=-3   opennet(sn) non-execution
res=-2   Parameter error
```

```
res=-1   Network open, no connection was established with the other party
res>=0  Data length
*/
res = CheckNet(sn)
If res ==-5 Then
Print("Ethernet restarting ")
ElseIf res ==-4 Then
Print("line out")
ElseIf res ==-3 Then
Print(" no running opennet funciton")
ElseIf res == -2 Then
Print("parameter error")
ElseIf res ==-1 Then
Print("no ESTABLISHED")
ElseIf res > 0 Then                    //If send and receive in Hex, change below 0 to 2
read_data=ReadNet(sn,0)                //Print(read_data) Sa
Print(read_data)
WriteNet(sn,0,read_data)               //Send read_data variable saved data to channel
                                       sn
EndIf
Delay(5)
Loop
```

Reference: CloseNet

CloseNet

CloseNet(net_num)		
Function:	Close network port	
Parameter:	net_num	Channel number: 1~4
Return:	-2	Parameter error
	1	Close success
Description:	Setting on the communication interface or manual operation, actual results are subject to instruction execution	
Example:	CloseNet(1)	//Close 1# channel network
Reference:	OpenNet	

ReadNet

ReadNet(net_num)

Function: Return ASCII data read by network port per channel number (string)

Parameter: net_num Channel number: 1~4

Return: read data

Description: 1.communication interface or manual operation, actual results are subject to instruction execution.

2.Can not use when network not open.

Example: recvstr = ReadNet(1) //Receive channel 1# network contents save to string variable recvstr.

Operation details refer to OpenNet

Reference: WriteNet

ReadNet(net_num, code)

Function: Return ASCII data read by network port per channel number (ASCII, OCTAL, HEX format choice)

Parameter: net_num Channel number: 1~4

code Encoder format: 0 stands for ASCII, 1 stands for Octal, 2 stands for Hex

Return: read data

Description: 1.ReadNet (1,0) equals to ReadNet(1), means receive ASCII data

2.When receive Octal,Hex byte value, return string, like: Sender sen d0x00,0x1F,0x01 by Hex, then recvstr="00 1F 01"

3.Confirm the sending format of the sender when using, if the receive format is not matched which could cause display showed garbled or receive data out of the agreement.

Example: string recvstr = ReadNet(1,2) //Read 1# serial port received Hex data

string recvstr = ReadNet(2,0) //Read 2# serial port received ASCII data

Operation details refer to OpenNet

Reference: WriteNet

WriteNet

WriteNet(net_num, str_write)		
Function:	Write data to network port	
Parameter:	net_num	Channel number: 1-4
	str_write	Data to be sent
Return:	-2	parameter error
	0	Sent data succeed (network port not open)
	1	Sent data failure
Description:	1.Setting on the communication interface or manual operation, actual results are subject to instruction execution.	
	2.Can not use when network	not open.
Example:	string sendstr = "ABC"	//Define sending contents
	WriteNet(1, sendstr)	//Send string “ABC” to 1# channel
Reference:	ReadNet	

WriteNet(net_num, code, str_write)

Function:	Write data to network port based on different encoder format	
Parameter:	net_num	Channel number: 1-4
	code	Encoder format: 0 stands for ASCII 1 stands for Octal, 2 stands for Hex
	str_write	Data to be sent
Return:	-2	Parameter error
	0	Sent data succeed (network not open)
	1	Sent data failure

Description: 1.WriteNet (1,0,str_wire) equals to WriteNet(1,str_write), means write data to 1# connection channel in ASCII format

2.Before writing hexadecimal data, convert the hexadecimal data to a byte string and send it as a string. If the format is not correct, the format is wrong will be alarmed.

Example:

```
string sendstr = "ABCDEFG"          //Define sending contents
WriteNet(1 , 0 , sendstr)           //Send "ABCDEFG" string to 1# serial port
string str_write = "05 1F 3D 42"    //Define string contents in Hex format
WriteNet(1 , 2 , str_write)         //Write 0x05 0x1F 0x3D 0x42 to 1# connection channel in Hex format
```

Operation details refer to OpenNet

Reference: ReadNet

CheckNet

CheckNet(net_num)		
Function:	Return the string length or number of bytes received by the network port	
Parameter:	net_num	Channel number: 1-4
Return:	>0	Return the string length or number of bytes
	-1	Network port opened, but not established the connection with other party
	-2	Parameter error
	-3	opennet(sn) not executed
	-4	Network disconnected
	-5	Initialization
Description:	1.Setting on the communication interface or manual operation, actual results are subject to instruction execution.	
	2.Can not use when network not open.	
	3.Like: Sender send 0x00,0x1F,0x01 in HEX, then returning value is 3, means 3bytes.	
	4.Like: Sender send “http”in ASCII code, then returning value is 4, means 4bytes.	
Example:	int chk= 0	//Define received number
	chk = CheckNet(1)	//Check received numbers, further process the corresponding content based on the received quantity.
	Operation details refer to OpenNet	
Reference:	ReadNet , WriteNet	

ClrNet

ClrNet(net_num)		
Function:	Clear network port receive buffer	
Parameter:	net_num	Channel number: 1-4
Return:	N/A	
Description:	1.Setting on the communication interface or manual operation, actual results are subject to instruction execution. 2.Cannot use when network not open.	
Example:	ClrNet(1)	//Clear 1# Channel buffer.
Reference:	ReadNet , WriteNet	

SetNet

SetNet(type, net_num, IP, port)		
Function:	Set network port info	
Parameter:	type	Channel type: 0 Service end, 1 Client end
	net_num	Channel number:1-4
	IP	IP address: “192.168.1.150”
	port	Port number: 1000~65535
Return:	-2	Parameter error
	1	Setting success
Description:	1.Setting on the communication interface or manual operation, actual results are subject to instruction execution. 2.Such as Interface set as A, SetNet instruction set as B, during the instruction execution period and afterwards execute in B setting. 3.When set this machine as service end, do not use port number: 502,8080.	
Example:	SetNet(1 , 1,"192.168.1.150", 5000)	//This machine 1# channel set as Client, set the other party server IP and port number.
	Operation details refer to OpenNet	
Reference:	OpenNet , CloseNet	

MIRead

MIRead(I_num)		
Function:	Read Int type modbus address corresponding value	
Parameter:	I_num	Variable I number: I0~I99
Return:	int type value	
Description:	1.Can check or set related value on the variable interface. 2.Not support power-failure saving.	
Example:	MIRead(1)	//Read int type modbus code I1 corresponding value, Equivalent to I1
Reference:	MFRead	

MFRead

MFRead(F_num)		
Function:	Read float type modbus address corresponding value	
Parameter:	F_num	Variable F number: F0~F99
Return:	float type value	
Description:	1.Can check or set related value on the variable interface. 2.Not support power-failure saving.	
Example:	MFRead(1)	//Read float type modebus number I1 corresponding value, Equivalent to I1
Reference:	MIRead	

MIWrite

MIWrite(I_num, intValue)		
Function:	Write Int type modbus address corresponding value	
Parameter:	I_num	Variable I number: I0~I99
	intValue	32 - bit signed integer value
Return:	N/A	
Description:	1.Can check or set related value on the variable interface. 2.Not support power-failure saving.	
Example:	MIWrite(1,-100)	//Write -100 to int type modebus number I1 corresponding address, Equivalent to I1=-100
Reference:	MFWrite	

MFWrite

MFWrite(F_num, floatValue)		
Function:	Write float type modbus address corresponding value	
Parameter:	F_num	Variable F number: F0~F99
	floatValue	32-bit float type value
Return:	N/A	
Description:	1.Can check or set related value on the variable interface. 2.Not support power-failure saving.	
Example:	MFWrite(1,123.456)	//Write 123.456 to float type modbus number F1 corresponding address, Equivalent to F1=123.456
Reference:	MIWrite	

GIRead

GIRead(I_num)		
Function:	Read Int type global variable address corresponding value	
Parameter:	I_num	Variable GI number: GI0~GI49
Return:	integer type value	
Description:	1.Can check or set related value on the variable interface. 2.Support power-failure saving. 3.Support modbus read & write.	
Example:	GIRead(1)	//Read integer type global variable number GI1 corresponding value, equivalent to GI1
Reference:	GFRead	

GFRead

GFRead(F_num)		
Function:	Read float type global variable address corresponding value	
Parameter:	F_num	Variable GF number: GF0~GF49
Return:	float type value	
Description:	1.Can check or set related value on the variable interface. 2.Support power-failure saving. 3.Support modbus read & write.	
Example:	GFRead(1)	//Read float type global variable number GF1 corresponding value, equivalent to GF1
Reference:	MIRead	

GIWrite

GIWrite(I_num, intValue)		
Function:	Write into Int type global address corresponding	
Parameter:	I_num	Variable GI number: GI0~GI49
	intValue	32 - bit signed integer value
Return:	N/A	
Description:	1.Can check or set related value on the variable interface. 2.Support power-failure saving. 3.Support modbus read & write.	
Example:	GIWrite(1,-100)	//Write -100 into GI1 corresponding address, equivalent to GI1 = -100
Reference:	MFWrite	

GFWrite

GFWrite(F_num, floatValue)		
Function:	Write into float type modbus address corresponding value	
Parameter:	F_num	Variable GF number: GF0~GF49
	floatValue	32-bit float type value
Return:	N/A	
Description:	1.Can check or set related value on the variable interface. 2.Support power-failure saving. 3.Support modbus read & write.	
Example:	GFWrite(1,123.456)	//Write 123.456 into GF1 corresponding address, equivalent to GF1 = 123.456
Reference:	MIWrite	

6.2.7 String

Instruction Name	Function Introduction
StrLen	Count and return string total number of characters
StrLeft	Return string left side specified number string
StrRight	Return the specified number of strings to the right of the string
StrMid	Return a string at the specified position in the middle of the string
StrTrim	Trim string head and tail specified content, return remains of string
StrFind	Find specified string position among the string being searched, and return this position
StrReplace	Find specified string position among the string being searched, and replace this position string
StrEmpty	Empty string variable value
StrCat	Concatenate multiple strings into one string
StrFormat	Concatenate multiple numerical value in format into one string
StrData	Slicer one string in format into multiple numerical value
StrSlicer	Slicer one string in specified string into multiple string
StrCmp	Compare two string sizes
StrnCmp	Compare the contents of two strings of specified length from the left
StrToupper	Converts small characters in a string to uppercase
StrTolower	Converts uppercase letters in a string to lowercase letters
StrToString	Converts numeric value content to string
StrToValue	Convert string content to numeric value
StrAscValue	Get the ASCII value of the first character of the string
StrAscChar	Get the character corresponding to the ASCII value

StrLen

StrLen(str)	
Function:	Count and return string total number of characters
Parameter:	str String to be counted
Return:	Numerical value
Description:	Numerical value starts with 1

Example:	string str = "abcdefg"
	int cnt
	cnt = StrLen(str)
	Print("str1 included character numbers is" , //Printing result is 7
	cnt)

Reference:	StrLeft , StrRight , StrMid , StrFind , StrTrim , Print

StrLeft

StrLeft(str, n)		
Function:	Return string left side specified number string	
Parameter:	str	String to be processed
	n	Character number to be returned
Return:	String	
Description:	Starting with the first one on the left, this operation does not change the string being manipulated	
Example:	string str = "abcdefg" , str1	
	str1 = StrLeft(str , 3)	
	Print("str1 = " , str1)	//str content is abc
Reference:	StrLen , StrRight , StrMid , StrFind , StrTrim , Print	

StrRight

StrRight(str, n)

Function:	Return the specified number of strings to the right of the string	
Parameter:	str	String to be processed
	n	Character number to be returned
Return:	String	
Description:	Start with 1st on the right	
Example:	string str = "abcdefg" , str1	
	str1 = StrRight(str , 3)	
	Print("str1 = " , str1)	//str content is efg
Reference:	StrLeft , StrLen , StrMid , StrFind , StrTrim , Print	

StrMid

StrMid(str, n1, n2)

Function:	Return a string at the specified position in the middle of the string	
Parameter:	str	String to be processed
	n1	The starting position of string
	n2	The end position of string
Return:	String	
Description:	Counting is 1 from the first character of the string	
Example:	string str = "abcdefg" , str1	
	str1 = StrMid(str , 3 , 4)	
	Print("str1 = " , str1)	//str content is cd
Reference:	StrLeft , StrRight , StrLen , StrFind , StrTrim , Print	

StrTrim

StrTrim(str1, str2, str3)

Function: Trim string head and tail specified content, return remains of string

Parameter:

str1	String to be processed
str2	The string to trim the header
str3	The string to trim the tail

Return: String

Description: The parts that do not need to be removed are represented by NULL

Example:

```
string srcStr="Image[X:123.123;Y:234.234;A:345.345]ImageDone"
string begins="Image"
string ends="ImageDone"
temps = StrTrim ( srcStr , begins , ends )        //temps=[X:123.123;Y:234.234;A:345.345]
```

Reference: StrLeft , StrRight , StrMid , StrFind , StrLen , Print

StrFind

StrFind(str1, str2, n)

Function:	Find specified string position among the string being searched, and return this position	
Parameter:	str1	String to be found
	str2	String keyword
	n	Specify the number of occurrences
Return:	number of occurrences	
Description:	The value of n starts at 1, indicating the position from which the first occurrence is to be found	
Example:	charstart = StrFind (temps , "[" , 1)	//temps=[X:123.123;Y:234.234;A:345.345],after the operation charstart=1
Reference:	StrLeft , StrRight , StrMid , StrLen , StrTrim , Print	

StrReplace

StrReplace(str, str1, str2, n)		
Function:	Find specified string position among the string being searched, and replace this position string	
Parameter:	str	The string variable to be replaced and stored as the target after the operation
	str1	String to be found
	str2	Replacement value
	n	Specifies the number of occurrences to replace,[Default] Indicates all replacement
Return:	N/A	
Description:	The value of n starts at 1, indicating the position from which the first occurrence is to be found	
Example:	StrReplace (temps , "abc" , "")	//temps value 123abc456, after operation, temps value is 123456
Reference:	StrFind , Print	

StrReplace

StrReplace(str, str1, str2)		
Function:	Find specified string position among the string being searched, and replace this position string	
Parameter:	str	The string variable to be replaced and stored as the target after the operation
	str1	String to be found
	str2	Replacement value
	n	Specifies the number of occurrences to replace,[Default] Indicates all replacement
Return:	N/A	
Description:	The value of n starts at 1, indicating the position from which the first occurrence is to be found	
Example:	StrReplace (temps , "abc" , "")	//temps value 123abc456, after operation, temps value is 123456
Reference:	StrFind , Print	

StrEmpty

StrEmpty(str)	
Function:	Empty string variable value
Parameter:	str String to be processed
Return:	N/A
Description:	This operation returns no value and changes the contents of the manipulated variable
Example:	string str = "abcdefg" int cnt StrEmpty(str) cnt = StrLen(str) Print("str = " , str , ", cnt =" ,cnt) //str =,cnt=0
Reference:	StrReplace , Print

StrCat

StrCat(str1, str2, ...)

Function:	Concatenate multiple strings into one string	
Parameter:	str1	String 1 to be concatenated
	str2	String 2 to be concatenated
	...	String to be concatenated
Return:	String	
Description:	Concatenate the parameters in the order in which they appear	
Example:	string str1 = "abc" , str2 = "defg" , str	
	str = StrCat(str1 , str2)	
	Print("str = " , str)	//str =abcdefg
Reference:	StrFormat , Print	

StrFormat

StrFormat(format, n1, n2, ...)

Function: Concatenate multiple numerical value in format into one string

Parameter:

format	Format, details refer to Description
n1	1st value
n2	2nd value
...	3rd or 4th ....value

Return: String

Description: 1.Enter the character style you want to roll out in the format, like string format="X:%f;Y:%f;A:%f".

2. Where %f is the format of the value.

3.Commonly used numeric value format representation, Floating point is type %f, integer %d, hexadecimal %x, binary 2 %b, floating point %s.

4.Specially,when the numeric type is floating point, you can specify the number of digits to display, like %3.3f,stands for xxx.xxx.

Example: Print (StrFormat(" charstart= %d, charnum= %d" , charstart , charnum))

//Printing content charstart= 1, charnum= 31

Reference: StrCat , StrData , StrSlicer , Print

StrData

StrData(str, format, n1, n2, ...)

Function: Slicer one string in format into multiple numerical value

Parameter:

str	String to be processed
format	Format
n2	2nd value
...	N-th...value

Return: N/A

Description: 1.Enter the character style you want to roll out in the format, like string format="X:%f;Y:%f;A:%f".

2. Where %f is the format of the value.

3.Commonly used numeric value format representation, Floating point is type %f, integer %d, hexadecimal %x, binary 2 %b, floating point %s.

Example: string format="[X:%f;Y:%f;A:%f]"

```
StrData( s1, format , x , y , angle )      //s1 content [X:123.123;Y:234.234;A:345.345]
Print(x , y , angle)                       //after operation:x=123.123000 y=234.234000
                                           angle=345.345000
```

Reference: StrFormat , StrSlicer , Print

StrSlicer

StrSlicer(str, str1, n, str2)

Function: Slicer one string in specified string into multiple string

Parameter:

str	String to be processed
str1	Delimiter, or string
n	at the beginning of the array, if omitted, it means all.
str2	Save variable to array after splitting

Return: N/A

Description: When the segmentation result is greater than 1, specified string array underlined as the starting store, this prevents multiple partitioned data from not being stored

Example:

```
string temps , s1 , s2 , strtemp_arr[3]
StrSlicer ( s2 , ";" , 1, strtemp_arr[0] )
StrSlicer ( s2 , ";" , 2, strtemp_arr[1] )
StrSlicer ( s2 , ";" , 3, strtemp_arr[2] )
```

or

```
string temps , s1 , s2 , strtemp_arr[3]
StrSlicer ( s2 , ";" strtemp_arr[0] )
                                    //s2 content X:123.123;Y:234.234;A:345.345
                                    //after operation
                                    //strtemp_arr[0] ="123.123"
                                    //strtemp_arr[1] ="234.234"
                                    //strtemp_arr[2] ="345.345"
```

Reference: StrFormat , StrData , Print

StrSlicer(str, str1, str2)

Function: Slicer one string in specified string into multiple string

Parameter:

str	String to be processed
str1	Delimiter, or string
n	at the beginning of the array, if omitted, it means all.
str2	Save variable to array after splitting

Return: N/A

Description: When the segmentation result is greater than 1, specified string array underlined as the starting store, this prevents multiple partitioned data from not being stored

Example:

```
string temps , s1 , s2 , strtemp_arr[3]
StrSlicer ( s2 , ";" , 1, strtemp_arr[0] )
StrSlicer ( s2 , ";" , 2, strtemp_arr[1] )
StrSlicer ( s2 , ";" , 3, strtemp_arr[2] )
```

or

```
string temps , s1 , s2 , strtemp_arr[3]
StrSlicer ( s2 , ";" strtemp_arr[0] )
```

```
//s2 content X:123.123;Y:234.234;A:345.345
//after operation
//strtemp_arr[0] ="123.123"
//strtemp_arr[1] ="234.234"
//strtemp_arr[2] ="345.345"
```

Reference: StrFormat , StrData , Print

StrCmp

StrCmp(str1, str2)

Function:	Compare two string sizes	
Parameter:	str1	String 1
	str2	String 2
Return:	Value difference	
	0	equal
	>0	String 1>String2
	<0	String1<String2
Description:	Compare the first character size	
Example:	While StrCmp (temps , "") != 0	
		//temps [X:123.123;Y:234.234;A:345.345]
		//The result after comparison is 91!=0, condition is established then enter the loop, remark: "["character ascii code value is 91
Reference:	StrnCmp , StrAscValue , StrAscChar , Print	

StrnCmp

StrnCmp(str1, str2, n)

Function:	Compare the contents of two strings of specified length from the left	
Parameter:	str1	String 1
	str2	String 2
	n	Specifies the length of the content from the left
Return:	Value difference	
	0	equal to
	>0	String1>String2
	<0	String1<String2
Description:	Compare multiple character size	
Example:	string a="abcde", b="abccdefg"	
	StrnCmp(a , b , 3)	//abc=abc,difference is 0
	StrnCmp(a , b , 4)	//abcd>abcc,difference is 1
Reference:	StrCmp , StrAscValue , StrAscChar , Print	

StrToupper

StrToupper(str)

Function: Converts small characters in a string to uppercase

Parameter: str String

Return: String

Description: N/A

Example: string a="AbcD1e" , s

s = StrToupper(a) //s="ABCD1E"

Reference: StrTolower , Print

StrTolower

StrTolower(str)	
Function:	Converts uppercase letters in a string to lowercase letters
Parameter:	str String
Return:	String
Description:	N/A
Example:	string a="AbcD1e" , s s = StrTolower(a) //s="abcd1e"
Reference:	StrToupper , Print

StrToString

StrToString(n)	
Function:	Converts numeric value content to string
Parameter:	n String to be processed
Return:	String
Description:	Values can be integer or floating point
Example:	string str float fx = 345.2 str = StrToString(fx) Print("str = " , str) //str =345.2
Reference:	StrToValue , Print

StrToValue

StrToValue(str)

Function: Convert string content to numeric value

Parameter: str String to be processed

Return: Numeric value

Description: 1.Values can be integer or float type.

2.This operation does not change the content of the original string.

Example:

```
string str = "345.256"
float fx
fx = StrToValue( str )
Print( "fx = " , fx )                    //fx=345.256000
```

If don't know whether a source string can be converted to a value, you can use the following custom function to determine

```
Function IsDigit(string s)
                                        //Test whether string can be converted to number
    int i,asc,flag=0
    If StrLen(s) < 1 Then
        Return 0                        //No data
    EndIf
    For i = 1 To StrLen(s)
        asc = StrAscValue(StrMid(s,i,1))
        If asc == 46 Then       //.
            flag = flag + 1
            If i == 1 Or flag > 1 Then
                Return 0
            EndIf
        ElseIf asc < 48 Or asc > 57 Then
            Return 0
        EndIf
    Next
    Return 1
```

```
FunctionEnd
```

Reference: StrToString , StrData , Print

StrAscValue

StrAscValue(str)

Function: Get the ASCII value of the first character of the string

Parameter: str String

Return: Numeric Value

Description: The ASCII value of the first character

Example: int a = StrAscValue(12ax) //a = 49 ,"1" ASCII numeric value is 49

Reference: StrnAscChar , Print

StrAscChar

StrAscChar(n)	
Function:	Get the character corresponding to the ASCII value
Parameter:	n Value, 0-128
Return:	Character
Description:	Do not set values beyond boundaries
Example:	string a = StrAscChar(49) //a ="1" ,"1" ASCII numeric value is 49
Reference:	StrnAscChar , Print

6.2.8 Others

Instruction Name	Function Introduction
int	Define the variable type as an integer
float	Define variables as float type
string	Define variable as string variable
point	Define the variable type as point type variable
Function	Define a method (function block/Function), can be called when in use
ServoOn	Used to set the motor enable of the robot to ON
ServoOff	Used to set the motor enable of the robot to OFF
GetServoStatus	Return the Robot all axes motor enable state
PowerOn	Start the Robot main program (Task 1)
PowerOff	Stop the main robot program (Task 1)
SysHalt	Pause the robot main program (Task 1)
SysReset	Reset the robot main program (Task 1)
GetSysStatus	Return to the working state of the robot in task 1
GetError	Return the alarm serial number of the robot
SetSysRatio	Set the run rate of the main robot program (task 1)
GetSysRatio	Get the run rate of the main robot program (task 1)

int

int	
Function:	Define the variable type as an integer
Parameter:	N/A
Return:	N/A
Description:	Int must be followed by a space. If multiple variables of the same type are used, separate them with separator ","
Example:	int a = 100 , b = -1 , c
Reference:	float , string , point

float

float	
Function:	Define variables as float type
Parameter:	N/A
Return:	N/A
Description:	Float must be followed by Space. If multiple variables of the same type are used, separate them with separator ","
Example:	float af = 1.23 , bf = -1.45 , cf
Reference:	int , string , point

string

string	
Function:	Define variable as string variable
Parameter:	N/A
Return:	N/A
Description:	String must be followed by Space. If multiple variables of the same type are used, separate them with separator ","
Example:	float as = "abc" , bs = "a:" , cs
Reference:	int , float , point

point

point	
Function:	Define the variable type as point type variable
Parameter:	N/A
Return:	N/A
Description:	Point must be followed by Space. If multiple variables of the same type are used, separate them with separator ","
	Generally, point a variable can be directly assigned value.
Example:	point posa = Pn(1) , posb = Where() , posc
Reference:	int , float , string

Function

Function Name( ).....FunctionEnd	
Function:	Define a method (function block/Function), can be called when in use
Parameter:	Configure as required
Return:	Return as required
Description:	If you want to define a method that takes parameters, use "()" to indicate it(type 1 Variable name 1, type 2, variable 2...)This parameter is entered as a variable parameter when the method is called; If the method to be defined takes no parameters, also need to add parentheses; Generally, point a variable can be directly assigned value.
Example:	Function myName() ... FunctionEnd
Reference:	int , float , string , point

ServoOn

ServoOn( )	
Function:	Used to set the motor enable of the robot to ON
Parameter:	N/A
Return:	N/A
Description:	ServoOn()used to set the motor power supply to ON (enable ON), And release the brake on all axes

Example:	If GetServoStatus() == 0 Then
	ServoOn() //Enable all axes
	EndIf

Reference:	GetServoStatus , ServoOff

ServoOff

ServoOff( )	
Function:	Used to set the motor enable of the robot to OFF
Parameter:	N/A
Return:	N/A
Description:	ServoOff( ) used to set the motor power supply to OFF (enable OFF), and regain the brake on all axes (if yes)
Example:	ServoOff() //Disable close all axes
Reference:	ServoOn

GetServoStatus

GetServoStatus( )

Function: Return the Robot all axes motor enable state

Parameter: N/A

Return: 0: All axes disabled currently

1: All axes enabled currently

Description: Used for when the motor status of enable needs feedback, for example, the external control device needs to know the robot enable status

Or for its own need to control the motor enable.

Example:

```
If GetServoStatus() == 0 Then
ServoOn()                    //Enable all axes
EndIf
```

Reference: ServoOn , ServoOff

PowerOn

PowerOn( )	
Function:	Start the Robot main program (Task 1)
Parameter:	N/A
Return:	N/A
Description:	For background communication task control main program start;

Example:	If StrCmp(sdata[0], "STARTRUN") == 0 Then
	//STARTRUN
	PowerOn()
	EndIf

Reference:	PowerOff , SysHalt , SysReset

PowerOff

PowerOff( )	
Function:	Stop the main robot program (Task 1)
Parameter:	N/A
Return:	N/A
Description:	For background communication task control main program stop;

Example:	If StrCmp(sdata[0], "STOP") == 0 Then
	//STOP
	PowerOff()
	EndIf

Reference:	PowerOn , SysHalt , SysReset

SysHalt

SysHalt( )	
Function:	Pause the robot main program (Task 1)
Parameter:	N/A
Return:	N/A
Description:	For background communication task control main program halt.

Example:	If StrCmp(sdata[0], "HALT") == 0 Then
	//HALT
	SysHalt()
	EndIf

Reference:	PowerOn , PowerOff , SysReset

SysReset

SysReset( )	
Function:	Reset the robot main program (Task 1)
Parameter:	N/A
Return:	N/A
Description:	Used for background communication task control main program reset (background program does not reset);

Example:	If StrCmp(sdata[0], "RESET") == 0 Then
	//RESET
	SysReset()
	EndIf

Reference:	PowerOn , PowerOff , SysHalt

GetSysStatus

GetSysStatus( )

Function: Return to the working state of the robot in task 1

Parameter: N/A

Return:

0:	In the reset
1:	In the emergency stop
2:	In the alarm
3:	Ready
4:	In service
5:	In the pause
6:	Single step operation
7:	Single step suspended

Description: Used for the situation that needs to feedback the running state of the main task, for example, the external control device needs to know the robot enabling state;

Or for its own need to control the motor enable.

Example:

```
If getsysterm == 1 And GetSysStatus() == 4 Then
WriteNet(sn,"STARTRUN,SUCCESS")
getsysterm = 0
EndIf
```

Reference: PowerOn , PowerOff , SysHalt , SysReset , GetError

GetError

GetError( )		
Function:	Return the alarm serial number of the robot	
Parameter:	N/A	
Return:	0:	No alarming
	>0:	Specific alarm number(An integer ranging from 0 to 65535.)
Description:	Used to query the specific alarm number when in the alarm status	

Example:	If GetError() > 0 Then
	If SysTime()-timer1 > 2000 Then
	WriteNet(sn,StrCat("ERROR,",StrToString(GetError())))
	timer1 = SysTime()
	EndIf
	EndIf	

Reference:	GetServoStatus	

SetSysRatio

SetSysRatio(ratio)

Function: Set the run rate of the main robot program (task 1)

Parameter: ratio Range: 1~100

Return: N/A

Description: The system select the nearest gear based on the proximity principle, usually used for socket communication control.

Example:

```
If IsDigit(sdata[1]) == 1 Then          //IsDigit() is a custom function, check whether it is a numeric string
SetSysRatio(StrToValue(sdata[1]))       //Ratio set by the user
Else
I0 = 1000                               //1000 The upper computer sending value err
EndIf
```

Reference: GetSysRatio

GetSysRatio

GetSysRatio( )	
Function:	Get the run rate of the main robot program (task 1)
Parameter:	N/A
Return:	Range: 1~100
Description:	The system select the nearest gear based on the proximity principle, usually used for socket communication control.

Example:	If StrCmp(sdata[0], "GETSYSRATIO") == 0 Then
	WriteNet(sn,StrCat("GETSYSRATIO,",StrToString(GetSysRatio()),",SUCCESS"))
	EndIf

Reference:	SetSysRatio

Chapter 7 Communication introduction

7.1 Communication introduction

The MDCS-510 system provides two physical interface modes for communication: serial port (RS-485/232) and Ethernet. The detailed functions are shown in the following table:

Terminal	Description					
RS-485	**location**	Db-9 identified as RS-485 on the system interface panel				
	Wiring definition		Type	Pin A	Pin B	Pin C
		COM1	RS485	1#485+	6#485-	-
		COM2	RS232	2#232RX	3#232TX	5#GND
	Function mode	Serial port communication	Custom protocol		Port	COM1,COM2
			Modbus-RTU		Port	COM1
		Remote control	Modbus-RTU		Port	COM1
	Modbus-RTU	Function	Serial port communication or remote control		I/O extended	
		Working	Slave station preset in the system. User no need to set		No master station function	
		Station address	System address: 1 configured by advanced parameter 159#		No master station function	
	Factory setting	Baud rate: 115200, digit byte: 8, stop byte: 1, parity: none				
Ethernet	**Port number**	1pc, **Exchange devices are required for multiple connections**				
	Connection Qty	4routs(Custom protocol)+ 2routs(Modbus protocol)				
	Function mode	Network	Custom protocol		4routs	
			Modbus-TCP		2routs(built-in)	
		Remote control	Modbus-TCP		2routs built-in,	
	Modbus-TCP setting	Working: Slave station Station address: 1(default), can be changed with the 159# advanced parameter, range: 1~250				
	Factory setting	IP: 192.168.1.80 Slave station address: 1				

7.2 How to use serial port (RS-485/232) and EXTIO (option)

7.2.1 Wiring

MDCS-510 provides below serial port:

One line (COM1) is RS-485 two-wire connection mode, the maximum transmission rate is 115200 baud.

Another line (COM2) is RS232 mode, users need to connect RX,TX, and GND according to port pin definition.

Before use, users should follow the wiring mode and required function mode provided in <7.1 Communication Function Introduction >, connect the communication lines between the equipment and the system

Alert	Ensure that the sequence of 485+ and 485- / 232RX and 232TX is correct when connecting cables, otherwise, communication failure may occur!

7.2.2 Custom protocol mode (COM1/COM2)

1)Open “Communication” interface -”serial port setting”

2)Communication parameter setting

Includes baud rate, data bit, stop bit, check mode and other Settings, if any differences, you can modify the parameters of either party to make them consistent.

3)Save setting

Click “Save” in valid at once.

Before clicking save, users can manually open it to test sending and receiving data, also not open, to control whether to open by the program command.

4)Test method for serial port communication

Manual tests can use the communication monitor window. Test the communication results with the command runtime (automatically), print instruction can be used with the print window display.

7.2.3 Modbus-RTU mode (COM1)

1)The Modbus function is enabled by default in the latest version, user can skip this step

2)Enter “communication”-”serial port setting”-select “COM1”Set communication “Baud ratio”, “data byte”, “stop byte”, “checking”, “agreement”--Modbus slave

3)Click “Open”, start effective

Remark: It takes effect immediately after being opened, external master station can communicate with the system immediately, and still works after a reboot, until user press “close” button.

4)Click---”save”

Tips	Ensure that the communication parameters of both parties are set to the same value, otherwise, data sending and receiving exceptions may occur!
Tips	Save the parameters after any configuration, in case the reboot does not take effect. See Appendix B Modbus Address table for information about the local address.

7.2.4 Extended I/O setting (EXTIO)

1)Set IO configuration parameters

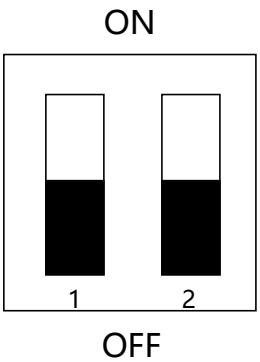
Select “ set”-”IO configuration”, select the number of I/O boards that you want to expand in the number of extended I/ OS box on the bottom, range: 0~4, 0 means cannot use.

Click “Save” key and exit, system will automatically complete the settings after saving the settings, effect after the operating system restart.

Tips	**The extended I/O interface of the MDCS-510 is transferred from the COM1 interface of the DC-415 to the EXTIO interface dedicated for I/O, for details, see 3.4 EXTIO Port wiring Definition**

2)Extended module station numbers introduction

Set the DIP switch of the extension module (Slave station No.), if only one expansion board is used, adjust the DIP switch to OFF, indicates that the current station number is 1, If there are multiple expansion boards corresponding to, the station of 2.3.4 should be set as follows:



Dip 1	Dip 2	Station	I/O Serial range
OFF	OFF	1	32~47
OFF	ON	2	48~63
ON	OFF	3	64~79
ON	ON	4	80~95

After the DIP switch is set, the extended module takes effect only after restarted.

3) Position of the DIP switch

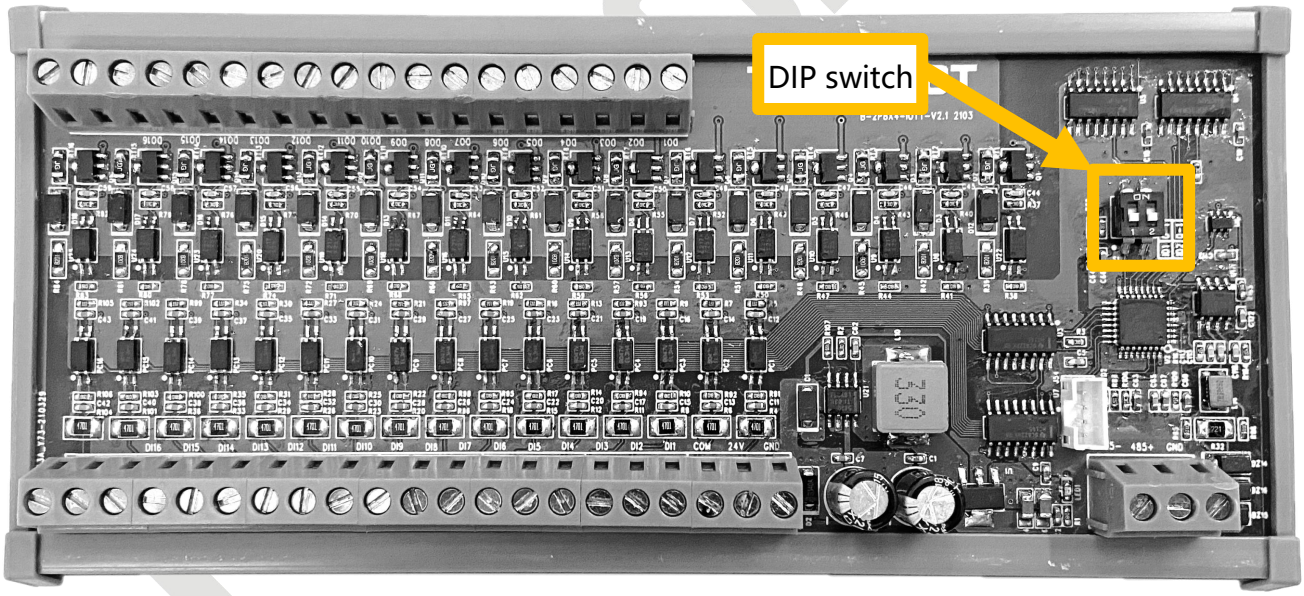


FIG. 7-2-2-3 I/O extended board plane

4)Extended I/O board user wiring

4.1)Communication connection between extended I/O board and control system

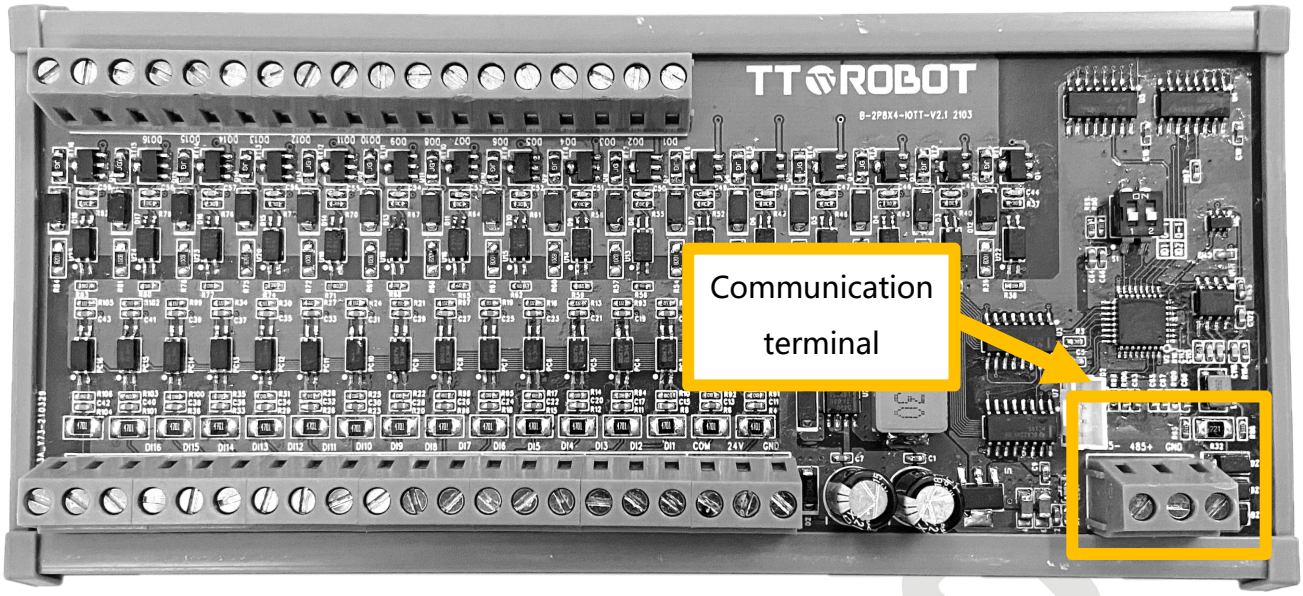


FIG. 7-2-2-4 I/ extended board plane

As shown in the figure above, you need to prepare two cables, wire diameter 0.2mm2 or larger, twisted pair is preferred; You can also use a direct line instead, the length cannot exceed 2m.

One end is connected to control box COM1 port, the attached DB9 (male head) needs to be welded. The other is 485+ connected to the middle (ID 485+/A), 485- connect to left side (ID 485-/B);

4.2)Subscriber Line Access

This method is similar to the system I/O connection method, for details, see < Extended I/O Board connection definition (option)>.

4.3)How to Use

For example, Output single coil solenoid valve with OUT32----, connect 24V to the positive pole of the solenoid valve, connect solenoid valve 0V to the first terminal on the output side (pin no. 0)

After the connection, open the indicator I/O - expansion /IO - manual output OUT32, Observe whether the solenoid valve works.

4.4)Indicator description

The V2.2 extended I/O board has an indicator, it can directly reflect the status of each channel signal input or output;

Other indicators are described as follows:

LED1 : Power indicator, it always lights when there is electricity supply;

LED2 : Expanded I/O board heartbeat indicator, Alternately on or off for 1 second, means the extended I/O board program is started.

RX : The message is received from the controller, flash fast during normal operation, but when no message is received, light is off.

TX : Indicates the message sent to the controller, flash fast during normal operation, unsent message indicator is off.

7.2.5 Serial port communication examples (Custom protocol)

```
Process Main
int sn = 2
int BaudRate=115200
                    /*115200,57600,38400,19200,9600,4800,2400,1200*/
int Databit=8
int Stopbit=1
string Parity="N"
int res=0
string read_data
SetCom( sn, BaudRate, Databit, Stopbit, Parity)
OpenCom(sn)
Do
    /*
      res = -2 Parameter error
      res = -1 Serial port not open
    */
    res = CheckCom(sn)
    If res == -2 Then
        Print("parameter error")
    ElseIf res == -1 Then
        Print("no running opencom")
    ElseIf res > 0 Then
        read_data = ReadCom(sn) //save the Read Data
        Print(read_data)
        WriteCom(sn,read_data)
    EndIf
    Delay(5)
Loop
ProcessEnd
```

Tips	If Modbus-RTU is used, users do not need to write transceiver programs. For details, see<5.2.7.2 Modbus custom variable read and write operate>

7.3 How to use Ethernet

7.3.1 Custom protocol description

The MDCS-510 supports a maximum of 6 external connections at the same time, 2 of them are Modbus-TCP or PC software connection channels fixed in the system; The remaining 4 channels provide free protocol connection channels for users(Display the highest 4 channels in the interface)

7.3.2 Custom protocol mode (TCP/IP)

1)Configure the network communication interface.

a. Engineer enter into channel--network configure.

b. In the network configuration group set IP, sub-net mask, and default gateway.

c. Select 1 channel in the channel configuration, set the local mode as Server/client.

It may be configured to IP addresses and ports, depending on the mode selected.

d. After Channel setting done, click---Add, to complete channel setting.

e. Click---"Save", so that the next boot will still take effect.

2)See the communication example below, select robot as server or client mode.

7.3.3 Modbus-TCP mode configuration

7.3.3.1 Set network port info

Set IP address to 192.168.1.80,255.255.255.0, 192.168.1.1

Save after the setting is done.

7.3.3.2 Set Modbus function enable

Modbus is enabled by default after the system start.

7.3.3.3 Switch the mode key to remote mode

Modbus external controls take effect only in this mode.

In this mode, the buttons on the Teach Pendant respond only to emergency stop, V+, V-, enable button and mode on key.

7.3.3 Network port communication example (Free agreement Robot as client end)

```
Process Main
//Set local SN channel as client
string serverIp="192.168.1.150"     //Server IP
int server_port=5000                //Server port number
int sn=1                            //Channel number
int res=0
string read_data
SetNet(1,sn, serverIp, server_port)   //Set local SN channel as client
OpenNet(sn)
Do
   /*
      res=-5 initialization
      res=-4 network line disconnected
      res=-3 opennet(sn)nnot executed
      res=-2 Paramter error
      res=-1 Network already open, but not established
      res>=0 data lenght/bytes
   */
    res = CheckNet(sn)
    If res == -5 Then
           Print("Ethernet restarting ")
    ElseIf res == -4 Then
        Print("line out")
    ElseIf res == -3 Then
        Print(" no running opennet funciton")
    ElseIf res == -2 Then
        Print("parameter error")
    ElseIf res == -1 Then
        Print("no ESTABLISHED")
    ElseIf res > 0 Then
        read_data=ReadNet(sn)       //Save data to read_data
        Print(read_data)
        WriteNet(sn,read_data)
    EndIf
    Delay(5)
Loop
ProcessEnd
```

7.3.4 Network port communication example (Custom protocol Robot as server)

```
Process Main
//Set sn channel as service end
string ServerIp="192.168.1.66"      //Server IP
int Server_port=6098                //Server port number
int sn=1                            //Channel number
int res=0
string read_data
SetNet(0,sn,ServerIp,Server_port)   //Set Channel sn as server
OpenNet(sn)
Do
   /*
      res=-5 Initialization
      res=-4 Network disconnected
      res=-3 opennet(sn) not executed
      res=-2 Parameter eror
      res=-1 Network already open, but not established
      res>=0 Data length/bytes
   */
    res = CheckNet(sn)
    If res ==-5 Then
           Print("Ethernet restarting ")
    ElseIf res ==-4 Then
        Print("line out")
    ElseIf res ==-3 Then
        Print(" no running opennet funciton")
    ElseIf res == -2 Then
        Print("parameter error")
    ElseIf res ==-1 Then
        Print("no ESTABLISHED")
    ElseIf res > 0 Then
        read_data=ReadNet(sn)       //Save data to read_data
        Print(read_data)
        WriteNet(sn,read_data)
    EndIf
    Delay(5)
Loop
ProcessEnd
```

Appendix

A Common alarm

Error code	Error message
1	Unrecognized symbol
2	The value input format is incorrect
3	The processing file does not exist
4	Unsupported syntax format
10	Initialize parameters with default Settings
11	Failed to open parameter file
12	Parameter file data read is abnormal
13	Failed to create parameter file
14	Failed to create parameter file
15	Parameter file CRC check error
16	The stored parameter value is incorrect
17	Item parameter change
18	Parameters are added
30	The initial coordinate system is set by default
31	Failed to open the coordinate system file
32	The data read from the coordinate system file is abnormal
33	Failed to create the coordinate system file
34	Failed to write the coordinate system file
35	The CRC check of the coordinate system file is incorrect
40	Initialize I/O configuration with default Settings
41	Failed to open the IO configuration file
42	The I/O configuration file is incorrectly read
43	The I/O configuration file failed to be created
44	Failed to write the I/O configuration file
45	The CRC check of the I/O configuration file is incorrect
50	Initialize area monitoring data with default Settings
51	Failed to open the area monitoring configuration file
52	Failed to read the area monitoring configuration file
53	Failed to create the area monitoring configuration file
54	Description Failed to write the area monitoring configuration file
55	The CRC check of the area monitoring configuration file is incorrect
60	Failed to record the area monitoring reference point
70	Initialize teaching with the default Settings
71	Failed to open teaching file
72	teaching file data read is abnormal
73	Failed to create teaching file
74	teaching failed to apply for storage space
75	Failed to write teaching file

76	CRC check error in teaching file
80	Failed to convert joint coordinates to Cartesian coordinates in teaching
81	Teach pendant Cartesian coordinate conversion to joint coordinate failed
82	Teaching User coordinate system conversion failed
83	Teaching invalid
84	Failed to record teaching
100	The initial project configuration information is incorrect
101	Error reading current project information
102	No project is currently loaded
103	Engineering loading failure
104	Remote mode project not found
200	1st Axis TT-Link real-time communication bus is abnormal
201	2nd Axis TT-Link real-time communication bus is abnormal
202	3rd Axis TT-Link real-time communication bus is abnormal
203	4th Axis TT-Link real-time communication bus is abnormal
1000	Unrecognized symbol
1001	The value input format is incorrect
1002	The processing file does not exist
1003	Unsupported syntax format
1004	The task content is empty
1005	Failed to save intermediate code file
1006	File start flag does not match task level
1007	Input parameter error
1008	modbus Global variables cannot be declaration typed
2000	Error building syntax tree
2001	Variable duplicate declaration
2002	Variable declaration initialization type error
2003	Variable is not declared to be used
2004	Operand type error
2005	Wrong assignment type
2006	Assignment operation left value type error
2007	Unsupported operator type
2008	An undeclared reference to array
2009	Array name variable is declared repeatedly
2010	The number of initialization rows in a point array is greater than the number of elements in the array
2011	The value of the initialization row of the point array is not equal to 6
2012	Array declaration initialization value type error
2013	Incorrect data type for fetching point
2014	Label redefinition
2015	The GoTo tag is not declared
2016	Label registration failed
2017	The expression statement format is incorrect

2018	Case expressions support only constants
2019	The number of Do nested layers (20) exceeds the upper limit
2020	The number of If nested layers (20) exceeds the upper limit
2021	The number of Switch nested layers (20) exceeds the upper limit
2022	The number of For nested layers (20) exceeds the upper limit
2023	Only global variable declarations are supported
2024	The number of arrays initialized does not match the array size
2025	Variable declaration error
2026	Array declaration error
2027	Statement does not have a matching terminator
2028	Number of Function nesting layers exceeds the upper limit (10)
2100	Unsupported function calls
2101	Incorrect number of function arguments
2102	The function parameter type is incorrect
2103	Function parameters out of range
2104	No movement commands are allowed in sub-tasks
2105	Incorrect function format
2106	Failed to register a custom function
2107	Error assigning object to function
2108	The number of returned values of a custom function exceeds the upper limit
2109	The number of values returned by a custom function does not match the number of left value lists
2110	Instruction functions do not support multiple assignments
2111	Custom functions are declared repeatedly
2112	"Unsupported parallel execution functions
2113	Custom functions are not supported during parallel execution
2114	The Dn instruction supports only constant parameters
2115	(PowerEn\GetError) Incorrect command executed in main task (PowerEn\GetError)
2200	Array initialization value type error
2201	The number of elements in an array declaration must be a positive integer
2202	Array initialization value and array size are inconsistent
2203	Array names are declared repeatedly
2204	The array name is undeclared
2205	Elements in an array reference must be positive integers
2300	Label redefinition
5000	The divisor cannot be 0
5001	Temporary variable subscript > Upper limit (10)
5002	Error obtaining non-point numeric value
5003	Error getting point value
5004	Numeric type error
5005	Variable subscript >Upper limit
5006	Error saving value

5007	Error jumping target
5008	The opcode type is incorrect
5009	String label >Upper limit
5010	Error starting from the specified row location
5011	The array element type is incorrect
5012	Error getting array element
5013	Switch The condition type is incorrect
5014	The case determine condition type is incorrect
5015	Error assigning a non-point variable
5016	Error assigning the point variable
5017	Error getting temporary variable value
5018	Error saving non-point temporary variable value
5019	Error saving value of temporary variable of type point
5020	Coordinate conversion error
5021	Cartesian coordinate offset error
6000	Function parsing error
6001	Number of function call parameters exceeds the upper limit(10)
6002	The number of function call arguments matches incorrectly
6003	Teach point not exist
6004	Unsupported function references
6005	Unsupported print type
6006	The function parameter type is incorrect
6007	The function returns an assignment type error
6008	he string length is smaller than the interception length
6009	String does not match numeric value
6010	Error obtaining Matrix point location
6011	Error saving temporary variable of type point
6012	Error getting function parameter type
6013	Error getting function argument content
6014	Error getting user coordinate value
6015	The parameter axis number is out of range
6016	The number of points in the array direction must be >= 2
6017	Custom function self-nesting depth exceed limit (> 10)
6018	A custom function failed to apply for stack space
6019	The return value of a custom function multiple assignment does not match the number of assignment objects
6020	The string format does not match
6021	Incorrect formatting string usage
6022	The value of a function parameter is out of range
6023	The point-position coordinate system for creating Matrix is inconsistent
6024	Error saving string
7000	Failed to apply for Memory Space
7001	The current data is empty

7100	Error identifying intermediate code file
7101	Error loading global variable file
8000	Undefined alarm
10000	System abrupt stop
10001	Servo is not enabled
10002	Invalid operation
10003	System not ready
10004	Servo enable shutdown response times out
10005	Emergency stop shutdown response times out
10006	Global variable task startup timeout
10007	The global variable task compiles the back-end response time out
10008	The startup mode of the main task is abnormal. Check the project file
10010	After the trial period expires
12000	Operation control response timed out
12001	Compiling the front-end response timed out
12010	Task 1 compiling response timed out
12011	Task 2 compiling response timed out
12012	Task 3 compiling response timed out
12013	Task 4 compiling response timed out
12014	Task 5 compiling response timed out
12015	Task 6 compiling response timed out
12016	Task 7 compiling response timed out
12017	Task 8 compiling response timed out
12020	The logical response of task 1 timed out
12021	The logical response of task 2 timed out
12022	The logical response of task 3 timed out
12023	The logical response of task 4 timed out
12024	The logical response of task 5 timed out
12025	The logical response of task 6 timed out
12026	The logical response of task 7 timed out
12027	The logical response of task 8 timed out
18000	TCP enters the obstacle zone
20001	Robot type error
20002	Path type error
20003	Failed to initialize the robot
20004	Disallow form conversion
20005	joint exceed software limit
20006	Axis 1 exceed software limit
20007	Axis 2 exceed software limit
20008	Axis 3 exceed software limit
20009	Axis 4 exceed software limit
20010	Axis 5 exceed software limit

20011	Axis 6 exceed software limit
20012	Axle 1 over-speed
20013	Axle 2 over-speed
20014	Axle 3 over-speed
20015	Axle 4 over-speed
20016	Axle 5 over-speed
20017	Axle 6 over-speed
20018	Pulse too large
20019	The motion time is 0
20020	Failed to calculate the velocity of the uniform section
20021	Acceleration time is 0
20022	Constant velocity time is 0
20023	Deceleration time is negative
20024	Position is too rush
20025	Time is too rush
20026	Wrong track type
20027	One dimensional direction is wrong
20028	Wrong robot form type
20029	The operation control parameters are incorrect
20030	The wrong direction
20031	Operation data frame cache is full
20032	Operation data frame cache is empty
20033	Waiting for control data frame timed out
20034	The central Angle is too small
20035	Failed to calibrate tool coordinate system
20036	Pulse sending failure
20037	Failed to get data frame
20038	root failure
20039	Vector uniformization failed
20040	Matrix inversion failed
20041	The vector dimension is 0
20042	Vector dimension too high
20043	Speed parameter error
20044	joint exceed software limit
20045	joint 1 exceed software lower limit
20046	joint 2 exceed software lower limit
20047	joint 3 exceed software lower limit
20048	joint 4 exceed software lower limit
20049	joint 5 exceed software lower limit
20050	joint 6 exceed software lower limit
20051	Joint motion command format is incorrect
20052	Failed to map the axis number
20053	Coupling parameter error

20054	Time out waiting for servo in position
20055	Motor encoder value communication failure
20056	Failed to calculate the center of a circle
20057	Failed to get the tool coordinate system
20058	Failed to get the user coordinate system
20059	Joint 1 servo timed out of position
20060	Joint 2 servo timed out of position
20061	Joint 3 servo timed out of position
20062	Joint 4 servo timed out of position
20063	Joint 5 servo timed out of position
20064	Joint 6 servo timed out of position
20065	Waiting to read interpolation data timed out
20066	The tool coordinate system parameter A is A non-zero value or B is non-zero value
20067	Error percentage of operation control completion
20068	The target point type is incorrect
20069	The z-axis of user coordinate system is not parallel to the z-axis of world coordinate system
20070	Speed parameter error
20071	Wrong acceleration parameter
20072	The command number of the operation data frame is out of order
20073	Smoothing parameter error
20074	Vector Angle parameter error
20075	Wrong number of axis 4 turns
20076	Error in calculating the number of turns of axis 4
20077	The number of data frames in the operation cache is incorrect
20078	Failed to switch the tool setting
20079	Acceleration over
20080	Motion time is negative
20081	Gate motion parameters are incorrect
20082	Iterative failure
20083	The serial number of gate motion target is incorrect
20084	Failed to calculate attitude Angle C
20085	User coordinate parameters are incorrect
20086	Time error
20087	The compiler side controls the data frame parameter error
20088	CP smoothing is not supported
20089	The interpolated data cache is empty
20090	The current Z height is greater than the gate motion limit Z height
25701	The Cartesian displacement is zero
25702	Joint position displacement is zero
25703	Velocity value ignored

25704	Idle executes the pause command
25705	Idle executes the abrupt stop command
25706	Running Status execute Running command
25707	Running Status execute reset command
25708	Emergency Stop execute running command
25709	Emergency stop execute Pause command
25710	Emergency stop execute abrupt stop command
25711	Pause status execute pause command
25712	Pause status execute abrupt stop command
30000	Axis 1 servo drive alarm
30001	Axis 1 servo VCE alarm< AL001 >
30002	Axis 1 servo over-voltage< AL002 >
30003	Axis 1 servo under-voltage< AL003 >
30004	Axis 1 servo current over-current< AL004 >
30006	Axis 1 servo current sampling channel is abnormal< AL006 >
30007	Axis 1 servo over-speed< AL007 >
30008	Axis 1 servo position is out of tolerance< AL008 >
30009	Axis 1 servo user torque overload< AL009 >
30016	Axis 1 servo brake is abnormal< AL010 >
30017	Axis 1 servo drive time out< AL011 >
30020	Axis 1 servo overheating< AL014 >
30021	Axis 1 servo overloading< AL015 >
30022	Axis 1 servo drives over load< AL016 >
30023	Axis 1 servo motor over load< AL017 >
30032	Axis 1 servo line saving initialization error /UVW initialization error< AL020 >
30033	Axis 1 servo absolute value encoder communication error< AL021 >
30034	Axis 1 servo absolute encoder CRC check error< AL022 >
30035	Axis 1 servo encoder battery alarm< AL023 >
30036	Axis 1 servo absolute encoder cycles overflow< AL024 >
30037	Axis 1 servo absolute encoder alarm< AL025 >
30038	Axis 1 servo absolute coordinate loss
30039	Axis 1 servo absolute encoder clear number failure warning
30040	Axis 1 servo PUU overflow alarm< AL028 >
30048	Axis 1 servo emergency stop< AL030 >
30049	Axis 1 servo forward limit trigger< AL031 >
30050	Axis 1 servo reverse limit trigger< AL032 >
30053	Axis 1 servo repeatedly assign the same function to different DI< AL035 >
30056	Axis 1 servo position command direction signal is abnormal< AL038 >
30064	Axis 1 servo initialization parameter failure (EEP)< AL040 >
30065	Axis 1 failed to obtain fault record in EEPROM< AL041 >
30080	Axis 1 serial communication failure< AL050 >
30081	Axis 1 serial communication timeout failure< AL051 >
31000	Axis 2 servo drive alarm

31001	Axis 2 servo VCE alarm< AL001 >
31002	Axis 2 servo over-voltage< AL002 >
31003	Axis 2 servo under-voltage< AL003 >
31004	Axis 2 servo current over-current< AL004 >
31006	Axis 2 servo current sampling channel is abnormal< AL006 >
31007	Axis 2 servo over-speed< AL007 >
31008	Axis 2 servo position is out of tolerance< AL008 >
31009	Axis 2 servo user torque overload< AL009 >
31016	Axis 2 servo brake is abnormal< AL010 >
31017	Axis 2 servo drive time out< AL011 >
31020	Axis 2 servo overheating< AL014 >
31021	Axis 2 servo overloading< AL015 >
31022	Axis 2 servo drives over load< AL016 >
31023	Axis 2 servo motor over load< AL017 >
31032	Axis 2 servo line saving initialization error /UVW initialization error< AL020 >
31033	Axis 2 servo absolute value encoder communication error< AL021 >
31034	Axis 2 servo absolute encoder CRC check error< AL022 >
31035	Axis 2 servo encoder battery alarm< AL023 >
31036	Axis 2 servo absolute encoder cycles overflow< AL024 >
31037	Axis 2 servo absolute encoder alarm< AL025 >
31038	Axis 2 servo absolute coordinate loss
31039	Axis 2 servo absolute encoder clear number failure warning
31040	Axis 2 servo PUU overflow alarm< AL028 >
31048	Axis 2 servo emergency stop< AL030 >
31049	Axis 2 servo forward limit trigger< AL031 >
31050	Axis 2 servo reverse limit trigger< AL032 >
31053	Axis 2 servo repeatedly assign the same function to different DI< AL035 >
31056	Axis 2 servo position command direction signal is abnormal< AL038 >
31064	Axis 2 servo initialization parameter failure (EEP)< AL040 >
31065	Axis 2 failed to obtain fault record in EEPROM< AL041 >
31080	Axis 2 serial communication failure< AL050 >
31081	Axis 2 serial communication timeout failure< AL051 >
32000	Axis 3 servo drive alarm
32001	Axis 3 servo VCE alarm< AL001 >
32002	Axis 3 servo over-voltage< AL002 >
32003	Axis 3 servo under-voltage< AL003 >
32004	Axis 3 servo current over-current< AL004 >
32006	Axis 3 servo current sampling channel is abnormal< AL006 >
32007	Axis 3 servo over-speed< AL007 >
32008	Axis 3 servo position is out of tolerance< AL008 >
32009	Axis 3 servo user torque overload< AL009 >
32016	Axis 3 servo brake is abnormal< AL010 >

32017	Axis 3 servo drive time out< AL011 >
32020	Axis 3 servo overheating< AL014 >
32021	Axis 3 servo overloading< AL015 >
32022	Axis 3 servo drives over load< AL016 >
32023	Axis 3 servo motor over load< AL017 >
32032	Axis 3 servo line saving initialization error /UVW initialization error< AL020 >
32033	Axis 3 servo absolute value encoder communication error< AL021 >
32034	Axis 3 servo absolute encoder CRC check error< AL022 >
32035	Axis 3 servo encoder battery alarm< AL023 >
32036	Axis 3 servo absolute encoder cycles overflow< AL024 >
32037	Axis 3 servo absolute encoder alarm< AL025 >
32038	Axis 3 servo absolute coordinate loss
32039	Axis 3 servo absolute encoder clear number failure warning
32040	Axis 3 servo PUU overflow alarm< AL028 >
32048	Axis 3 servo emergency stop< AL030 >
32049	Axis 3 servo forward limit trigger< AL031 >
32050	Axis 3 servo reverse limit trigger< AL032 >
32053	Axis 3 servo repeatedly assign the same function to different DI< AL035 >
32056	Axis 3 servo position command direction signal is abnormal< AL038 >
32064	Axis 3 servo initialization parameter failure (EEP)< AL040 >
32065	Axis 3 failed to obtain fault record in EEPROM< AL041 >
32080	Axis 3 serial communication failure< AL050 >
32081	Axis 3 serial communication timeout failure< AL051 >
33000	Axis 4 servo drive alarm
33001	Axis 4 servo VCE alarm< AL001 >
33002	Axis 4 servo over-voltage< AL002 >
33003	Axis 4 servo under-voltage< AL003 >
33004	Axis 4 servo current over-current< AL004 >
33006	Axis 4 servo current sampling channel is abnormal< AL006 >
33007	Axis 4 servo over-speed< AL007 >
33008	Axis 4 servo position is out of tolerance< AL008 >
33009	Axis 4 servo user torque overload< AL009 >
33016	Axis 4 servo brake is abnormal< AL010 >
33017	Axis 4 servo drive time out< AL011 >
33020	Axis 4 servo overheating< AL014 >
33021	Axis 4 servo overloading< AL015 >
33022	Axis 4 servo drives over load< AL016 >
33023	Axis 4 servo motor over load< AL017 >
33032	Axis 4 servo line saving initialization error /UVW initialization error< AL020 >
33033	Axis 4 servo absolute value encoder communication error< AL021 >
33034	Axis 4 servo absolute encoder CRC check error< AL022 >
33035	Axis 4 servo encoder battery alarm< AL023 >
33036	Axis 4 servo absolute encoder cycles overflow< AL024 >

33037	Axis 4 servo absolute encoder alarm< AL025 >
33038	Axis 4 servo absolute coordinate loss
33039	Axis 4 servo absolute encoder clear number failure warning
33040	Axis 4 servo PUU overflow alarm< AL028 >
33048	Axis 4 servo emergency stop< AL030 >
33049	Axis 4 servo forward limit trigger< AL031 >
33050	Axis 4 servo reverse limit trigger< AL032 >
33053	Axis 4 servo repeatedly assign the same function to different DI< AL035 >
33056	Axis 4 servo position command direction signal is abnormal< AL038 >
33064	Axis 4 servo initialization parameter failure (EEP)< AL040 >
33065	Axis 4 failed to obtain fault record in EEPROM< AL041 >
33080	Axis 4 serial communication failure< AL050 >
33081	Axis 4 serial communication timeout failure< AL051 >

B Modbus communication address

Group	Definition	Add. type	Value	Read/ write	Start address-decimal	Start address-Hex	Length
Sys status	System working mode	unsigned short(16b)	0x00: initialization 0x01: remote auto 0x02: remote manually 0x03: remote DNC	Read only	24576	0x6000	1
Sys status	System status	unsigned short(16b)	0x00: initialization/reset 0x01: abrupt stop 0x02: alarm 0x03: ready 0x04: running 0x05: pause	Read only	24577	0x6001	1
Sys status	Servo status	unsigned short(16b)	0x00: Initialization/servo off 0x01: servo enable on	Read only	24578	0x6002	1
Sys status	Current project number	unsigned short(16b)	0: No project currently 1~99: current in use project	Read only	24579	0x6003	1
Sys status	Current alarm number	unsigned short(16b)	0~99	Read only	24580	0x6004	1
Sys status	Current alarm number	unsigned short(16b) ×10	Alarming number	Read only	24581	0x6005	10
Sys control	System control command	unsigned short(16b)	0x01: Reset 0x02: Start 0x03: Pause 0x04: read alarm/clear alarm	Read/Write	24592	0x6010	1
Sys control	Servo enable command	unsigned short(16b)	0x01: servo enable ON<->OFF;	Read/Write	24593	0x6011	1
System control	Servo monitor reset command	unsigned short(16b)	0x01: Clear servo monitoring data record	Read/Write	24594	0x6012	1
Sys control	System mode switch command	unsigned short(16b)	0x01: Switch to auto mode 0x02: Switch to manual mode	Read/Write	24595	0x6013	1
Sys control	Ratio change command	unsigned short(16b)	0x01: ratio+ 0x02: ratio-	Read/Write	24596	0x6014	1
Sys control	Auto ratio	unsigned short(16b)	0~16: 1%,2%,5%,10%,15%,20%,25%,30%,35%,40%,45%,50%,60%,70%,80%,90%,100%	Read only	24597	0x6015	1

Sys control	Manual ratio	unsigned short(16b)	0~19: 0.01mm,0.1mm,1mm, 1%,2%,5%,10%,15%,20%,25%,30%,35%,40%,45%,50%,60%,70%,80%,90%,100%	Read only	24598	0x6016	1
Sys control	Manual mode	unsigned short(16b)	0x00: joint 0x01: world 0x02: tool 0x03: user	Read/Write	24599	0x6017	1
Sys control	Project operation instruction	unsigned short(16b)	1: loading specified No. project	Read/write	24600	0x6018	1
Sys control	Target project number	unsigned short(16b)	1~99	Read/Write	24601	0x6019	1
Real-time data	current user coordinate system number	unsigned short(16b)	0~15	Read/write	24624	0x6030	1
Real-time data	Current tool coordinate system number	unsigned short(16b)	0~8	Read/write	24625	0x6031	1
Real-time data	Current running row number	unsigned short(16b)	0~49999	Read only	24626	0x6032	1
Real-time data	Current operational command execution progress	unsigned short(16b)	0~100	Read only	24627	0x6033	1
Real-time data	real-time world coordinate	float(32b)×6	X,Y,Z,A,B,C	Read only	24640	0x6040	2*6
Real-time data	Real-time current user coordinates	float(32b)×6	UX,UY,UZ,UA,UB,UC	Read only	24656	0x6050	2*6
Real-time data	Real-time joint coordinate	float(32b)×6	J1,J2,J3,J4,J5,J6	Read only	24672	0x6060	2*6
Servo monitor	Real-time speed	float(32b)	Axis 1 real-time speed R/PM	Read only	24832	0x6100	2
Servo	Maximum	float(32b)	Axis 1 maximum speed r/pm	Read	24834	0x6102	2

monitor	speed			only			
Servo monitor	Real-time load	float(32b)	Axis 1 real-time load %	Read only	24836	0x6104	2
Servo monitor	Max. load	float(32b)	Axis 1 max load %	Read only	24838	0x6106	2
Servo monitor	Real-time current	float(32b)	Axis 1 real time current A	Read only	24840	0x6108	2
Servo monitor	Max. current	float(32b)	Axis 1 max. Current A	Read only	24842	0x610a	2
Servo monitor	Motor real time absolute position	int(32b)	Axis 1 motor real-time	Read only	24844	0x610c	2
Servo monitor	Servo Command position	int(32b)	Axis 1 directive real-time plus	Read only	24846	0x610e	2
Servo monitor	Axis 2 real time data	Same as Axis 1	Same as Axis 1	Read only	24848	0x6110	2
Servo monitor	Axis 3 real time data	Same as Axis 1	Same as Axis 1	Read only	24864	0x6120	2
Servo monitor	Axis 4 real time data	Same as Axis 1	Same as Axis 1	Read only	24880	0x6130	2
Servo monitor	Axis 5 real time data	Same as Axis 1	Same as Axis 1	Read only	24896	0x6140	2
Servo monitor	Axis 6 real time data	Same as Axis 1	Same as Axis 1	Read only	24912	0x6150	2
Teach operation	Target Teach number	unsigned short(16b)	0~999	Read/write	25088	0x6200	1
Teach operation	Teach operation directive	unsigned short(16b)	0x01: record Teach at current coordinate 0x02: Move to Teach Pendant position 0x03: Teach data modify and update 0x04: Save Teach file 0x05: Cancel Teach data	Read/write	25089	0x6201	1
Teach operation	Move to point mode	unsigned short(16b)	0x01: MOVJ;0x02: MOVL;0x03: JUMP;	Read/write	25857	0x6501	1
Teach operation	JUMP movement limits height	float(32b)	Z Limit high data mm	Read/write	25878	0x6516	2

Teach data	Current operation Teach number	unsigned short(16b)	0~999	Read/write	25090	0x6202	1
Teach data	p[n.0]Effective	unsigned short(16b)	1: Valid 0: invalid	Read/write	25091	0x6203	1
Teach data	p[n.0]Enter form	short(16b)	0: left hand 1: right hand -1: N/A	Read/write	25092	0x6204	1
Teach data	p[n.0]Coord type	unsigned short(16b)	0: Joint 1: Cartesian	Read/write	25093	0x6205	1
Teach data	p[n.0] User Coord number	short(16b)	0~15; -1: N/A	Read/write	25094	0x6206	1
Teach data	Preserved	short(16b)			25095	0x6207	1
Teach data	p[n.0] Coord value	float(32b)	X/J1	Read/write	25096	0x6208	2
Teach data	p[n.0] Coord value	float(32b)	Y/J2	Read/write	25098	0x620a	2
Teach data	p[n.0] Coord value	float(32b)	Z/J3	Read/write	25100	0x620c	2
Teach data	p[n.0] coord value	float(32b)	A/J4	Read/write	25102	0x620e	2
Teach data	p[n.0] Coord value	float(32b)	B/J5	Read/write	25104	0x6210	2
Teach data	p[n.0] Coord value	float(32b)	C/J6	Read/write	25106	0x6212	2
coordinate operation	Current coordinate type	unsigned short(16b)	0x00: User coordinate system 0x01: Tool coordinate system	Read/write	25344	0x6300	1
coordinate operation	The current operating coordinate system	unsigned short(16b)	0~15: Corresponding to the user coordinate system 0~8: Corresponding tool coordinate system	Read/write	25345	0x6301	1
coordinate operation	The current operation indicates the point number	unsigned short(16b)	1~6: Corresponding to 1st ~6th point;	Read/write	25346	0x6302	1

coordinate operation	Coordinate system command	unsigned short(16b)	0x01: Empty coordinate 0x02: Mark point entry 0x03: Move to the designated point 0x04: Calibration calculation 0x05: designated point data store 0x06: Coordinate system data storage 0x07: Cancel save 0x08: point data update 0x09: coordinate system data update	Read/write	25347	0x6303	1
coordinate data	Designated point temporary data	float(32b)	X/J1	Read/write	25348	0x6304	2
coordinate data	Designated point temporary data	float(32b)	Y/J2	Read/write	25350	0x6306	2
coordinate data	Designated point temporary data	float(32b)	Z/J3	Read/write	25352	0x6308	2
coordinate data	Designated point temporary data	float(32b)	A/J4	Read/write	25354	0x630a	2
coordinate data	Designated point temporary data	float(32b)	B/J5	Read/write	25356	0x630c	2
coordinate data	Designated point temporary data	float(32b)	C/J6	Read/write	25358	0x630e	2
coordinate data	Coordinate temporary value	float(32b)	X/J1	Read/write	25360	0x6310	2
coordinate data	Coordinate temporary value	float(32b)	Y/J2	Read/write	25362	0x6312	2
coordinate data	Coordinate temporary value	float(32b)	Z/J3	Read/write	25364	0x6314	2
coordinate data	Coordinate temporary value	float(32b)	A/J4	Read/write	25366	0x6316	2
coordinate	Coordinate	float(32b)	B/J5	Read/write	25368	0x6318	2

data	temporary value						
coordinate data	Coordinate temporary value	float(32b)	C/J6	Read/write	25370	0x631a	2
Input point	input coil	bit(coil)	0: OFF; 1: ON;	Read/write	16384	0x4000	1
Output point	Output coil	bit(coil)	0: OFF; 1: ON;	Read/write	16640	0x4100	1
Internal semaphore	**Coil**	bit(coil)		Read/write	16896	0x4200	1
Manual	Forward motion command	bit(coil)	Manual X/J1+Command(1-move;0-N/A)	Read/write	17408	0x4400	1
Manual	Forward motion command	bit(coil)	Manual Y/J2+Command(1-move;0-N/A)	Read/write	17409	0x4401	1
Manual	Forward motion command	bit(coil)	Manual Z/J3+Command(1-move;0-N/A)	Read/write	17410	0x4402	1
Manual	Forward motion command	bit(coil)	Manual A/J4+Command(1-move;0-N/A)	Read/write	17411	0x4403	1
Manual	Forward motion command	bit(coil)	Manual B/J5+Command(1-move;0-N/A)	Read/write	17412	0x4404	1
Manual	Forward motion command	bit(coil)	Manual C/J6+Command(1-move;0-N/A)	Read/write	17413	0x4405	1
Manual	Negative motion command	bit(coil)	Manual X/J1-command(1-move;0-N/A)	Read/write	17424	0x4410	1
Manual	Negative motion command	bit(coil)	Manual Y/J2-command(1-move;0-N/A)	Read/write	17425	0x4411	1
Manual	Negative motion command	bit(coil)	Manual Z/J3-command(1-move;0-N/A)	Read/write	17426	0x4412	1
Manual	Negative motion command	bit(coil)	Manual A/J4-command(1-move;0-N/A)	Read/write	17427	0x4413	1
Manual	Negative motion command	bit(coil)	Manual B/J5-command(1-move;0-N/A)	Read/write	17428	0x4414	1
Manual	Negative motion command	bit(coil)	Manual C/J6-command(1-move;0-N/A)	Read/write	17429	0x4415	1
Custom	Data I0	signed int(32b)	integer	Read/write	40960	0xa000	2

Custom	Data I1	signed int(32b)	integer	Read/write	40962	0xa002	2
Custom	...	signed int(32b)	integer	Read/write		...	
Custom	Data I99	signed int(32b)	integer	Read/write	41158	0xa0c6	2
Custom	Data F0	float(32b)	float	Read/write	41160	0xa0c8	2
Custom	Data F1	float(32b)	float	Read/write	41162	0xa0ca	2
Custom	...	float(32b)	float	Read/write		...	
Custom	Data F99	float(32b)	float	Read/write	41358	0xa18e	2
Custom	Data GI0	signed int(32b)	integer	Read/write	41984	0xa400	2
Custom	Data GI1	signed int(32b)	integer	Read/write	41986	0xa402	2
Custom	...	signed int(32b)	integer	Read/write		...	
Custom	Data GI99	signed int(32b)	integer	Read/write	42082	0xa462	2
Custom	Data GF0	float(32b)	float	Read/write	42240	0xa500	2
Custom	Data GF1	float(32b)	float	Read/write	42242	0xa502	2
Custom	...	float(32b)	float	Read/write		...	
Custom	Data GF99	float(32b)	float	Read/write	42338	0xa562	2

.

Guangdong Tiantai Robot Co., Ltd.

Address: No.23, Dalianghonggang Section, G105 National Road, Shunde District, Foshan City, Guangdong Province, PRC.

Tel：+86 0757-22222777 Website: http://www.gdtiantai.com