



DOBOT

User Guide

DobotSCStudio

User Guide

(MG400 & M1 Pro)

Issue: V2.1.8

Date: 2021-10-19

Shenzhen Yuejiang Technology Co., Ltd

Copyright © Shenzhen Yuejiang Technology Co., Ltd 2021. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without the prior written consent of Yuejiang Technology Co., Ltd

Disclaimer

To the maximum extent permitted by applicable law, the products described (including its hardware, software, and firmware, etc.) in this document are provided **AS IS**, which may have flaws, errors or faults. Yuejiang makes no warranties of any kind, express or implied, including but not limited to, merchantability, satisfaction of quality, fitness for a particular purpose and non-infringement of third party rights. In no event will Yuejiang be liable for any special, incidental, consequential or indirect damages resulting from the use of our products and documents.

Before using our product, please thoroughly read and understand the contents of this document and related technical documents that are published online, to ensure that the robot is used on the premise of fully understanding the robot and related knowledge. Please use this document with technical guidance from professionals. Even if follow this document or any other related instructions, damages or losses will be happening in the using process. Dobot shall not be considered as a guarantee regarding all security information contained in this document.

The user has the responsibility to make sure following the relevant practical laws and regulations of the country, in order that there is no significant danger in the use of the robot.

Shenzhen Yuejiang Technology Co., Ltd.

Address: Address: Floor 9-10, Building 2, Chongwen Garden, Nanshan iPark, Liuxian Blvd,
Nanshan District, Shenzhen, Guangdong Province, China

Website: www.dobot.cc

Preface

Purpose

This manual introduces the functions and usage of the robot control software DobotSCStudio, which is convenient for users to understand and use MG400.

Intended Audience

This document is intended for:





- Customer
- Sales Engineer
- Installation and Commissioning Engineer
- Technical Support Engineer

Change History

Date	Change Description
2021/10/19	Update the UI interface; add Blockly programming and safesetting functions; update programming commands; delete calibration, drag and brake function
2021/04/29	The first release

Symbol Conventions

The symbols that may be founded in this document are defined as follows.

Symbol	Description
 DANGER	Indicates a hazard with a high level of risk which, if not avoided, could result in death or serious injury
 WARNING	Indicates a hazard with a medium level or low level of risk which, if not avoided, could result in minor or moderate injury, robot damage
 NOTICE	Indicates a potentially hazardous situation which, if not avoided, can result in equipment damage, data loss, or unanticipated result
 NOTE	Provides additional information to emphasize or supplement important points in the main text

Contents

1. Overview.....	5
1.1 Description on Main Interface.....	5
2. Fast Connection.....	7
3. Function Description.....	10
3.1 Enabling.....	10
3.2 Setting Global Velocity Rate.....	10
3.3 Alarm Description.....	11
3.4 Jogging.....	11
3.5 Blockly.....	13
3.6 Programming.....	16
3.6.1 Project Description.....	16
3.6.2 Programming Interface Description.....	16
3.6.3 Programming Description.....	18
3.7 Parameter.....	28
3.7.1 Setting User Coordinate System.....	28
3.7.2 Setting Tool Coordinate System.....	33
3.7.3 I/O Monitor.....	36
3.7.4 Controller Setting.....	37
3.7.5 Remote Control.....	39
3.7.6 RobotParams.....	42
3.7.7 RobotSetting.....	47
3.7.8 SafeSetting.....	48
3.8 ToolConfig.....	50
3.8.1 BasicConfig.....	50
3.8.2 PluginsInfo.....	50
3.8.3 Log.....	50
3.8.4 Network Service.....	51
3.8.5 Tools.....	52
3.8.6 VirtualRobot.....	52
3.8.7 WiFi Setting.....	52
4. Program Language.....	54
4.1 Arithmetic Operators.....	54
4.2 Relational Operator.....	54
4.3 Logical Operators.....	54
4.4 General Keywords.....	55
4.5 General Symbol.....	55
4.6 Processing Control Commands.....	55
4.7 Global Variable.....	55
4.8 Motion Commands.....	56
4.9 Motion Parameter Commands.....	62
4.10 Input/output Commands.....	64
4.11 Program Managing Commands.....	65

4.12 Pose Getting Command.....	67
4.13 TCP.....	69
4.14 UDP.....	71
4.15 Modbus.....	72
4.15.1 Description on Modbus Register.....	72
4.15.2 Command Description.....	74
4.16 Conveyor Tracking.....	76
4.17 Pallet.....	78
Appendix A Servo Alarm Description.....	83
Appendix B Controller Alarm Description.....	88

1. Overview

DobotSCStudio is an industrial robot programming platform launched by Yuejiang, which is suitable for the whole series of industrial robots (MG400/SA/SR/CR/M1 Pro). With friendly interface, it supports secondary development by users. It also provides kinematics algorithm of various mechanical structures and integrated virtual simulation environment to realize rapid deployment of various process applications on site.

1.1 Description on Main Interface

Figure 1.1 shows the main interface of DobotSCStudio. Table 1.1 lists the interface description.

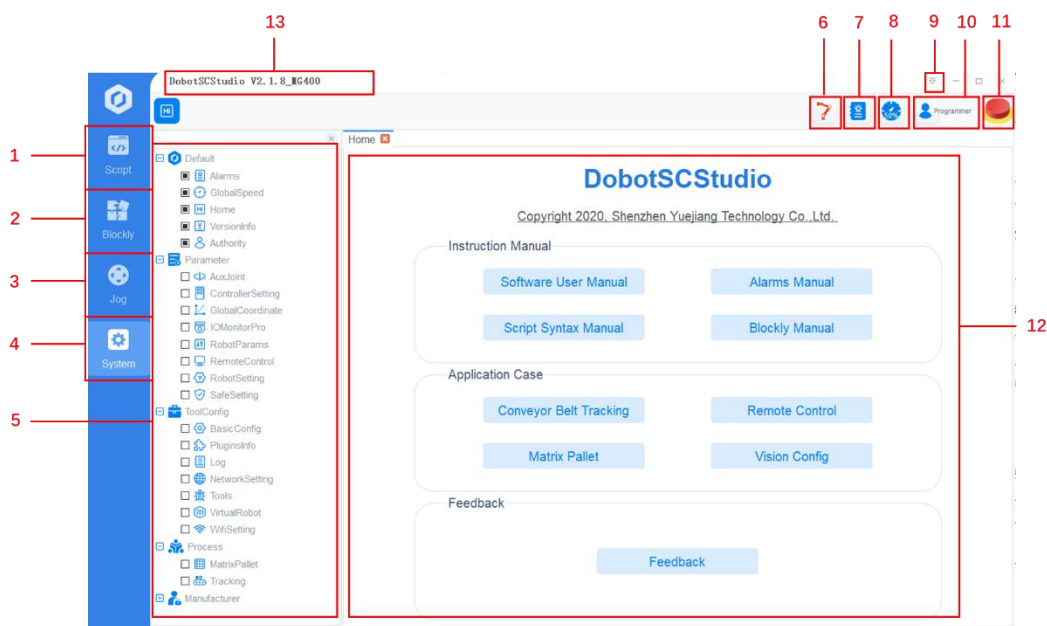


Figure 1.1 Main interface

Table 1.1 Interface description

No.	Description
1	Script You can build or import a project, and debug or run it
2	Blockly You can write programs by graphical language to quickly and conveniently control the robot
3	Jog Jog the robot in different coordinate systems. Jog the robot in the Joint coordinate system: From top to bottom, jog J1, J2, J3 and J4

No.	Description
	Jog the robot in the Cartesian coordinate system: From top to bottom, jog the X, Y, Z, R
4	<p>System</p> <p>You can set system configurations, such as NetworkSetting, RobotParams, Coordinate, Process, etc.</p>
5	System bar
6	Click this button to enable or disable the motor
7	<p>Check robot alarms</p> <p>When an alarm is triggered, this icon will flash red</p> <p>You can check the alarm details on the operation panel and clear it</p>
8	Set global velocity rate
9	<p>IP setting or check update</p> <p>After connecting robot and PC with network cable, you need to select Real on the IP Settings page and select robot's IP address for connecting to DobotSCStudio</p>
10	<p>Select user mode</p> <ul style="list-style-type: none"> • Watcher: Check the system status, I/O status, robot pose, and alarms • Operator: Operate a robot based on the existing scripts without programming • Programmer: On the basis of operator authority, you can program and teach • Manager: On the basis of programmer authority, you can set or modify parameters <p>Please select user mode based on site requirements</p> <p>Default password: admin. You can modify the password on the ToolConfig > BasicConfig > UserMode page in the Manager mode</p>
11	<p>Emergency stop switch</p> <p>Press and hold it in an emergency, and the drive power supply of MG400 will be powered off for emergency braking</p>
12	Interactive window
12	<p>Show the current running mode</p> <p>Running mode: I/O, Modbus, SCStudio (SCStudio mode is not displayed)</p>

2. Fast Connection

DobotSCStudio can communicate with MG400 directly through Ethernet1. At this point, the IP address of MG400 should be in the same network segment as that of the PC. The default IP address of the robot is 192.168.1.6 and cannot be modified. Please modify the IP address of PC to make them in the same network segment.

NOTE

- Minimum computer configuration for installing DobotSCStudio:
System: Windows7 64-bit/Windows10 32/64 bit
Memory: 4GB or above
CPU: Intel i3 or above
- This section uses Windows7 OS as an example to describe how to change the IP address. Please change it based on site requirements.

Procedure

Step 1 Connect power adapter to robot **Power Switch** interface.

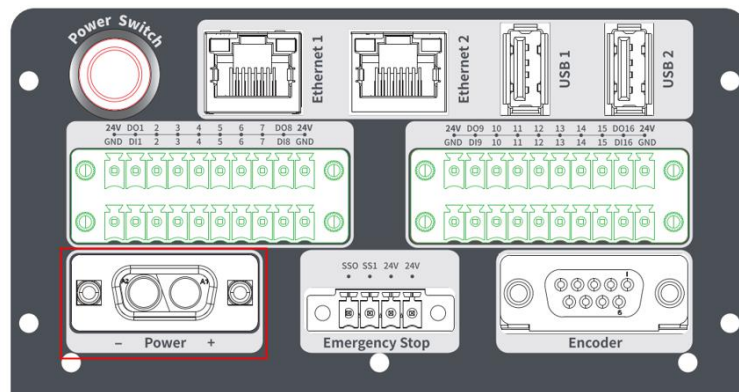


Figure 2.1 Connect to robot Power Switch interface

Step 2 Connect emergency stop switch to **Emergency Stop** interface.

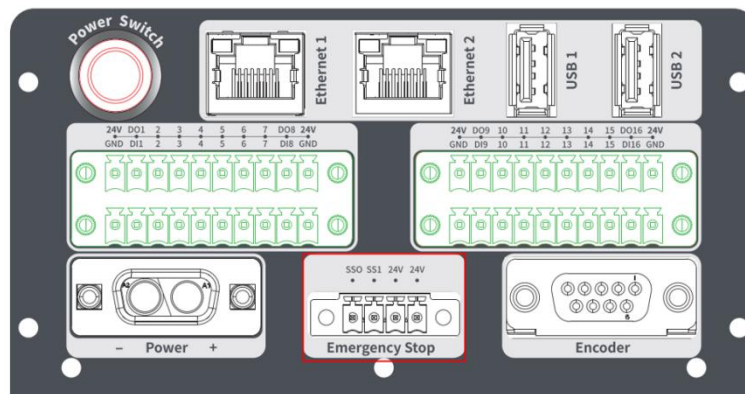


Figure 2.2 Connect to Emergency Stop interface

- Step 3** Connect one end of the network cable to the **Ethernet1** interface on the robot and the other end to the PC.
- Step 4** Click **Start > Control Panel** on the PC and select **Network and Sharing Centre**. The **Network and Sharing Centre** page is displayed.
- Step 5** Click **Local Area Connection** on the **Network and Sharing Center** page.
- Step 6** Click **Properties**.
- Step 7** Double-click Internet Protocol Version 4(TCP/IPv4).
- Step 8** Select **Use the following IP address**, and change the IP address, subnet mask, and gateway of the PC.

You can change the IP address of the PC to make it on the same network segment as that of the robot without conflict. The subnet mask and gateway of the PC must be the same as that of robot. For example, the computer IP address is 192.168.1.40, and the default gateway is 255.255.255.0.

NOTICE

If the PC is connected to robot over a network cable directly, you only need to set the IP address and subnet mask of the PC.

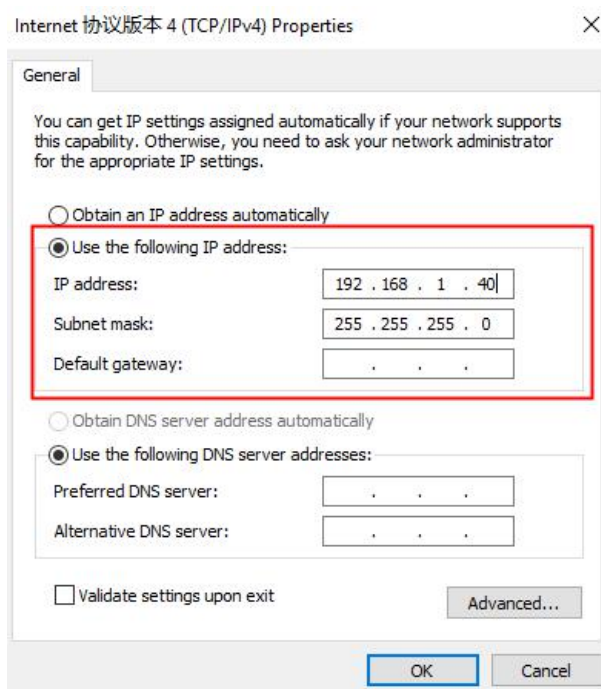



Figure 2.3 IP address modification

- Step 9** Click **OK**.
- Step 10** Click  > **IP settings...** on the upper right pane of the DobotSCStudio page and select the robot's IP address, then click **OK**.

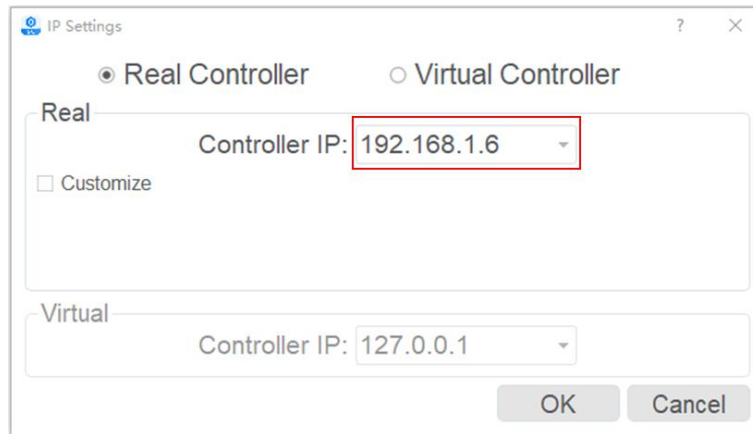


Figure 2.4 IP setting

After the connection is successful, the DobotSCStudio will be shown as below.

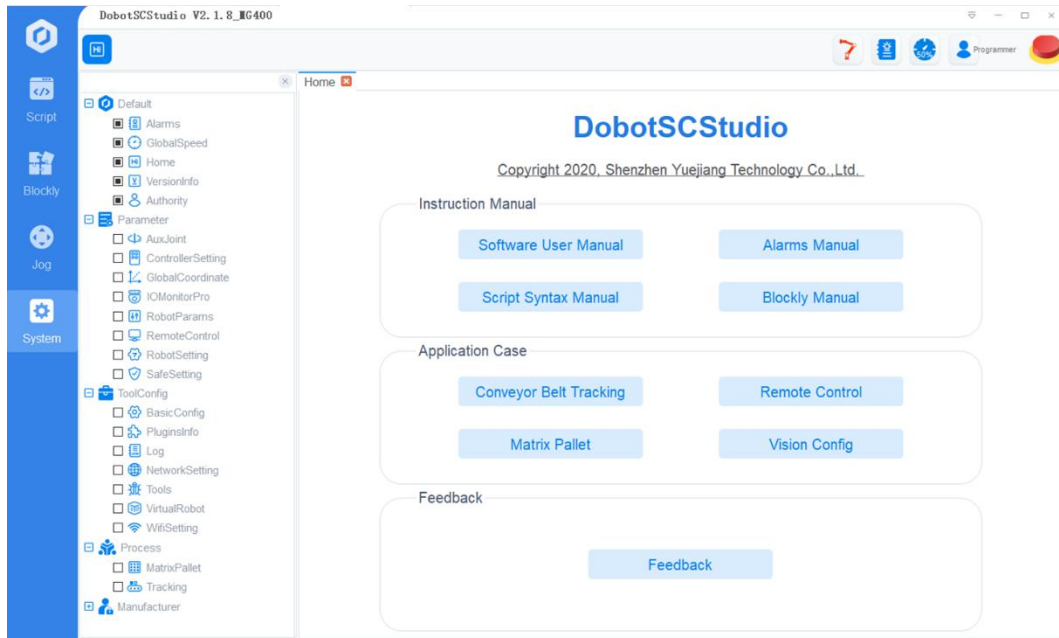







Figure 2.5 Connected successfully

3. Function Description

3.1 Enabling

Click  to enable MG400. When the icon  turns into , MG400 can be controlled by running the program or Jogging.

3.2 Setting Global Velocity Rate

Please click  and then click buttons on the operation panel to increase or decrease the global velocity ratio by 1%, 5%, 10%, 25% and 50%, as shown in Figure 3.1. You can also change the velocity in  > **Default** > **GlobalSpeed**. The Global Velocity Rate is not modified when the program is running. It can only be done when the program is not running or is suspended.

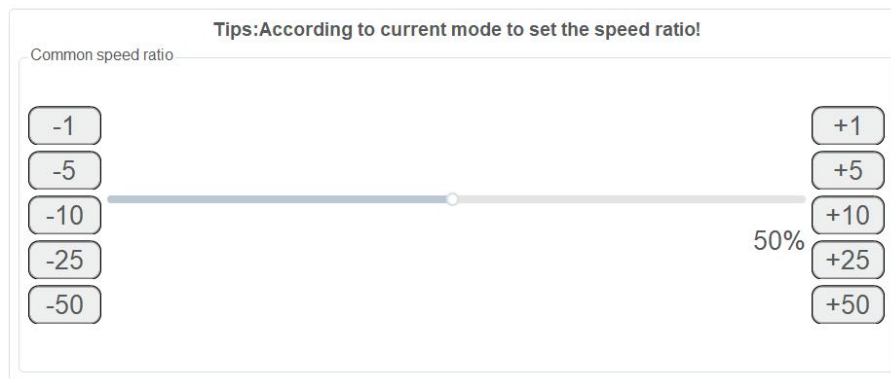


Figure 3.1 Modify the global velocity rate

When doing jogging or playback, the method calculating the velocity and acceleration for each axis (in Joint or Cartesian coordinate system) is shown as follows.

- Actual jogging velocity = the maximum jogging velocity * global velocity rate
- Actual jogging acceleration = the maximum jogging acceleration * global velocity rate
- Actual playback velocity = the maximum playback velocity * global velocity rate * the set velocity rate in the velocity function
- Actual playback acceleration = the maximum playback acceleration * global velocity rate * the set acceleration rate in the acceleration function
- Actual playback jerk = the maximum playback jerk * global velocity rate * the set acceleration rate in the jerk function



NOTE

- The maximum velocity, acceleration, or jerk can be set on the **Settings** page. For details, please see 3.7.6 *RobotParams*.
- The rates (velocity rate, acceleration rate, or jerk rate) can be set in the related


speed functions.

3.3 Alarm Description

If teaching point is incorrect, for example, a robot moves to where a point is at a limited position or a singular point, an alarm will be triggered.

If an alarm is triggered when running MG400, the alarm icon  on the DobotSCStudio turns into . You can check the alarm information on the **Alarm** page, as shown in Figure 3.2.

Please clear the alarm as follows:

- If a limitation alarm is triggered, please jog the limited joint axis towards the opposite direction to clear the alarm.
- If other alarms are triggered, please click  on the alarm page to clear the alarm. If the alarm cannot be cleared, please reboot MG400.

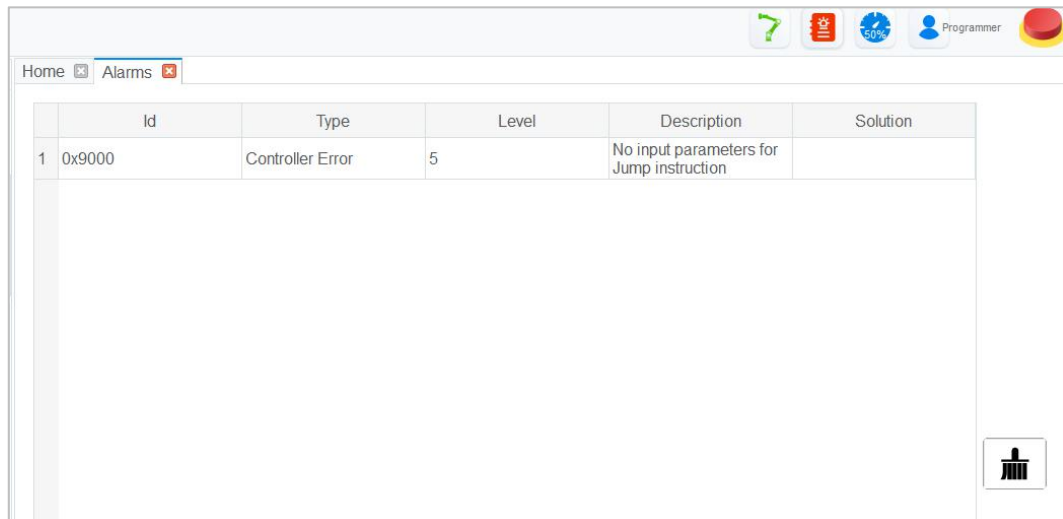


Figure 3.2 Alarm page

3.4 Jogging

You can jog the robot in different coordinate systems, Figure 3.3 shows the jogging panel, and Table 3.1 lists the description on jogging panel.

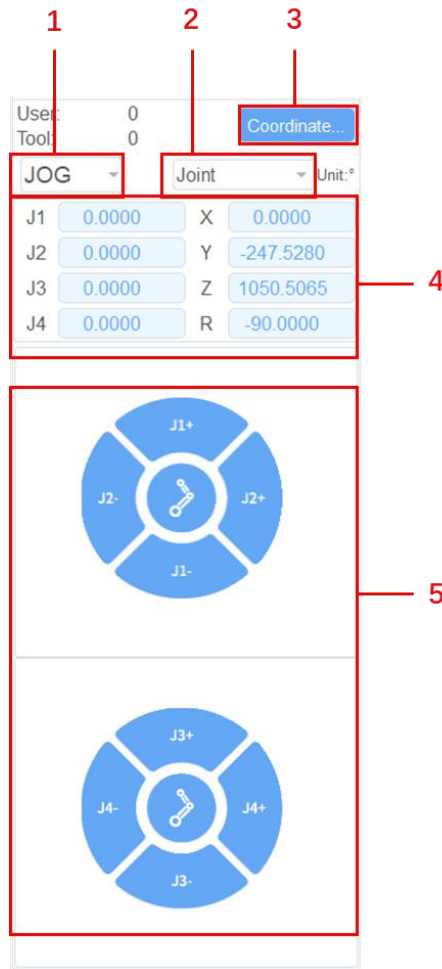


Figure 3.3 Jogging panel

Table 3.1 Description of jogging panel

No.	Description
1	<p>Step mode</p> <p>You can select the right step in the Step mode. The step supports JOG, 0.1, 0.5, 1 and 5.</p> <ul style="list-style-type: none"> JOG: indicates that in continuous jog movement, the speed is the maximum speed * global velocity rate 0.1, represents the displacement of 0.1° (joint coordinate system) or 0.1mm (Cartesian coordinate system) in a single jog movement 0.5, represents the displacement of 0.5° (joint coordinate system) or 0.5mm (Cartesian coordinate system) in a single jog movement. 1.0, represents the displacement of 1° (joint coordinate system) or 1mm (Cartesian coordinate system) in a single jog movement. 5.0, represents the displacement of 5° (joint coordinate system) or 5mm (Cartesian coordinate system) in a single jog movement

No.	Description
2	Jogging type It supports two types: Joint coordinate system, Cartesian coordinate system
3	Coordinate system According to the actual needs, you can select one of the preset user coordinate systems as the current user coordinate system
4	Location data Display the current joint position and tool center position
5	Jogging button Jog the robot in the Joint coordinate system: From top to bottom, jog J1, J2, J3 and J4 Jog the robot in the Cartesian coordinate system: From top to bottom, jog X, Y, Z, and R

3.5 Blockly

Blockly is a kind of block programming. You can write programs by graphical language to quickly and conveniently control the robot. Figure 3.4 shows the blockly panel, and Table 3.2 lists the description on Blockly panel.

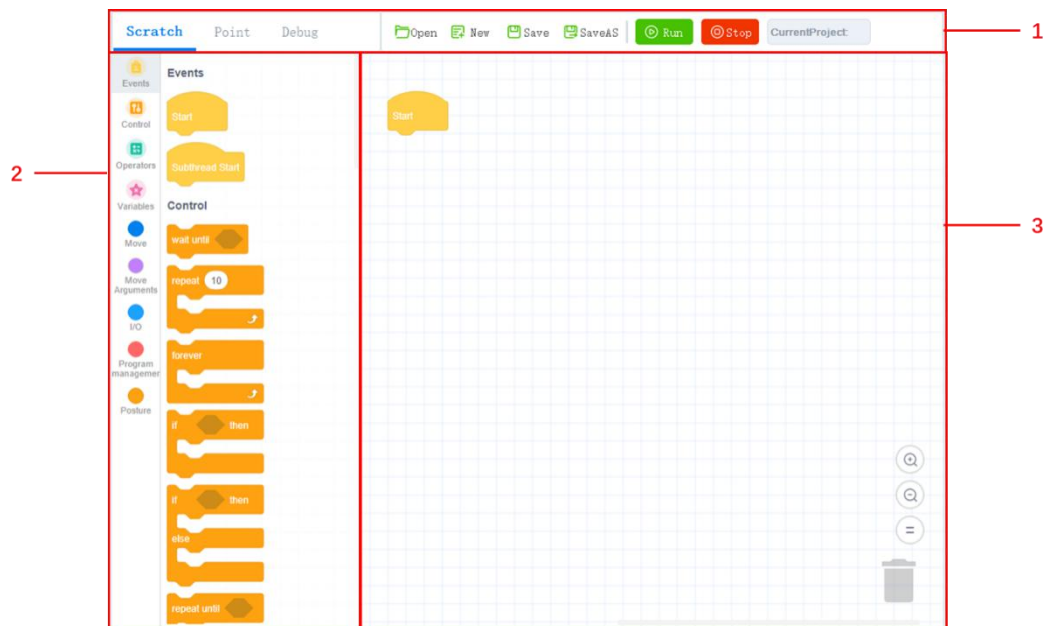













Figure 3.4 Blockly page

Table 3.2 Description on Blockly page


No.	Function	Description
-----	----------	-------------

1	Menu bar	<p> Open : Open a project that you have created</p> <p> New : Create a new project</p> <p> Save : Save the project</p> <p> Save As : Save the current project with a new name</p> <p> Run : Start running the program in the current code area</p> <p> Stop : Stop the running program</p> <p>Point: Save teaching point that can be called when writing a program</p> <p>Debug: Convert blockly into corresponding script. You can copy the script to  to see its running status. Please see <i>3.6 Programming</i> for details</p>
2	Block area	Provide all blocks
3	Code area	<p>Drag block to this page and edit it. Click the icon    in the code area</p> <p>to zoom in, zoom out and restore the blocks,  can be used to delete the selected block</p>

Prerequisites

The robot has been powered on.

Procedure

Step 1 Click  to enter the Blockly page.

The system creates a new project by default and supports multi-thread movement.

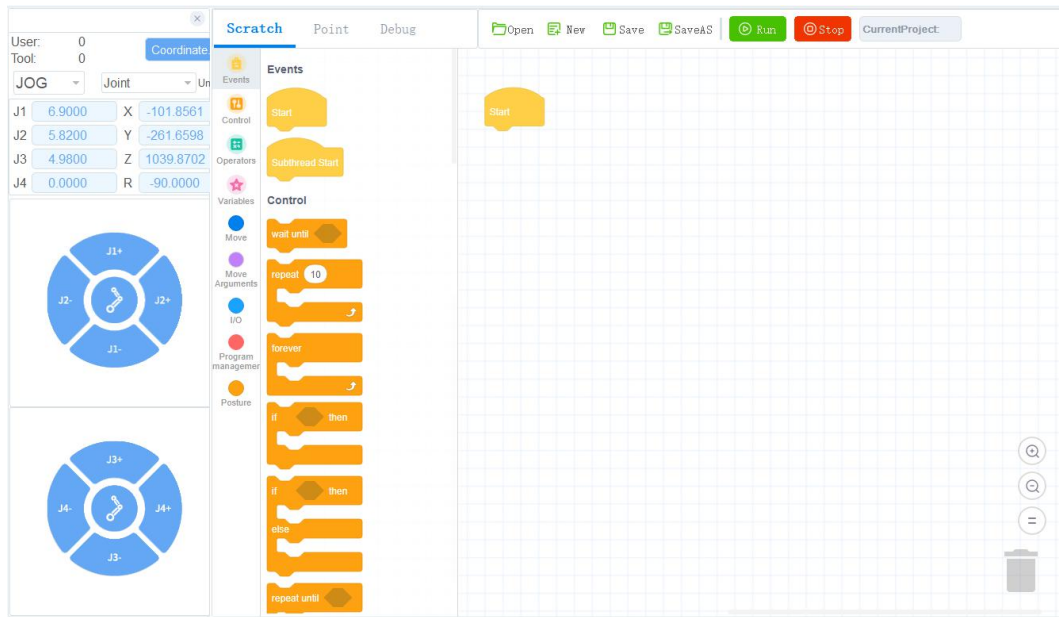


Figure 3.5 Blockly page

Step 2 Drag the blocks to the code area to start programming, as shown in Figure 3.6.

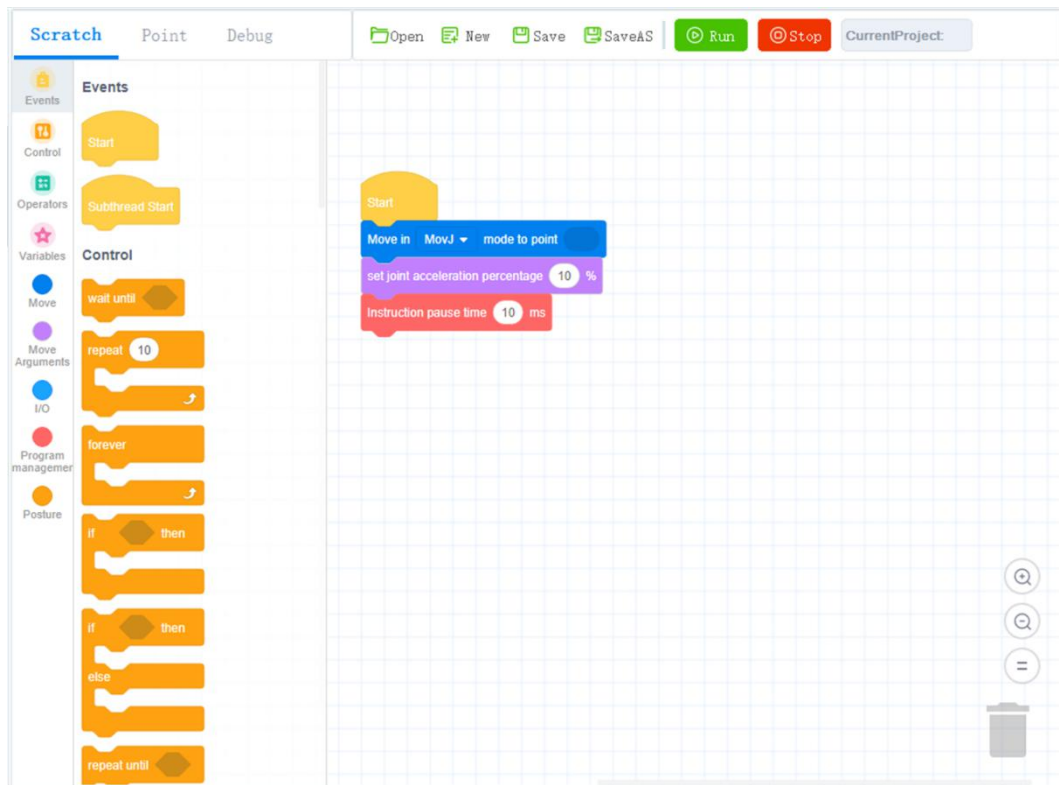




Figure 3.6 Writing a program

Step 3 Click  **Save** to save the current project

If it is the first time to save, you need to enter the project name.

Step 4 Click  to enable the robotic arm.

Step 5 Click  to start and run the program in the current code area.

3.6 Programming

3.6.1 Project Description

The robot program is managed in project form, including teaching points list, global variables, and program files. Figure 3.7 shows the project structure.

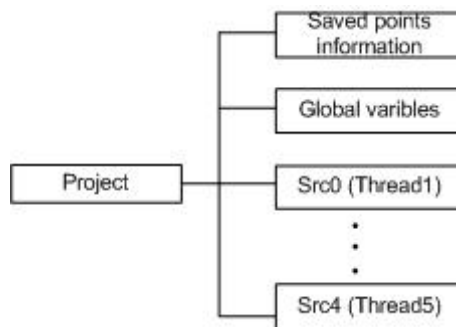


Figure 3.7 Project structure

- In script, no more than five threads can be executed simultaneously. Src0.lua is the main thread, and other threads are sub threads, which run parallel to the main thread.
- In the sub threads, the motion commands cannot be called. Only the main thread supports motion commands.
- Global variable module is only used to define global variables and module functions. The motion commands cannot be called here.

3.6.2 Programming Interface Description

When you write a program, you need to switch to programmer mode or above. Figure 3.8 shows the programming panel and Table 3.3 lists its description.

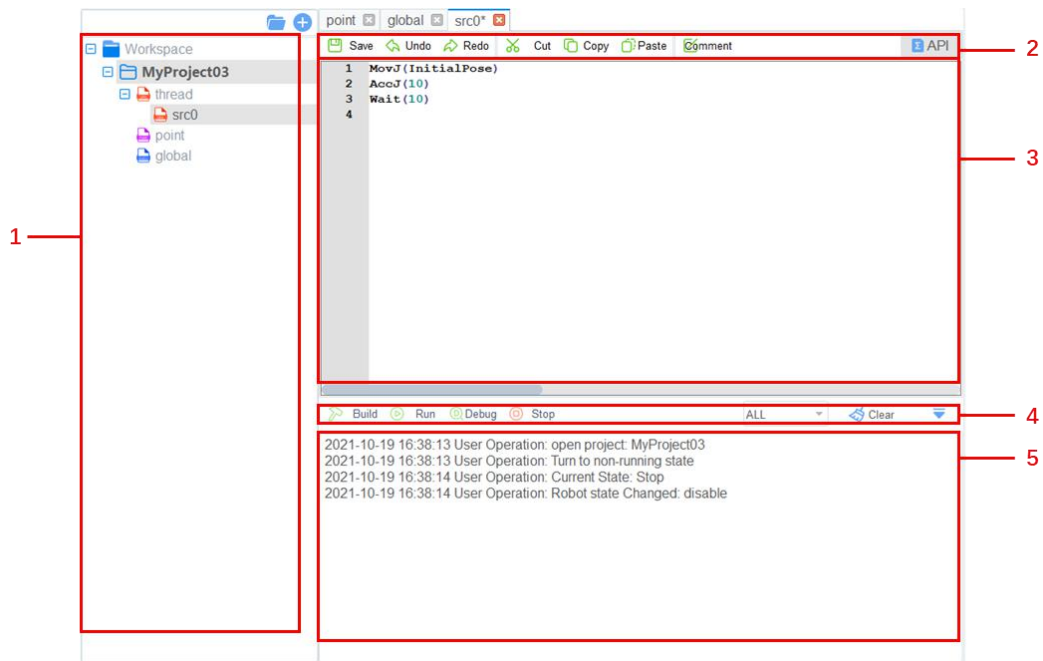




Figure 3.8 Programming panel







Table 3.3 Description on programming panel

No.	Description
1	Project files <ul style="list-style-type: none"> Point: Teach points. For details, please see 3.6.3.3 <i>Teaching points</i>. Global: Define and initialize global variables, points or functions Src0~Src4: Multithreaded files. The number of tasks is related to threads that you set when creating a project. Up to five threads can be executed simultaneously
2	Common buttons. For details, please see Table 3.4
3	Programming area
4	Running button, for details, please see Table 3.6
5	Debug result

Table 3.4 describes the common button.

Table 3.4 Description on common buttons

Icon	Description
 Save	Save the project
 Undo	Cancel

 Redo	Redo
 Copy	Copy the selected codes
 Cut	Cut the selected codes
 Paste	Paste the selected codes
 Comment	Code comment
 API	API libraries, for details, please see <i>4 Program Language</i>

3.6.3 Programming Description

Figure 3.9 shows the programming process.

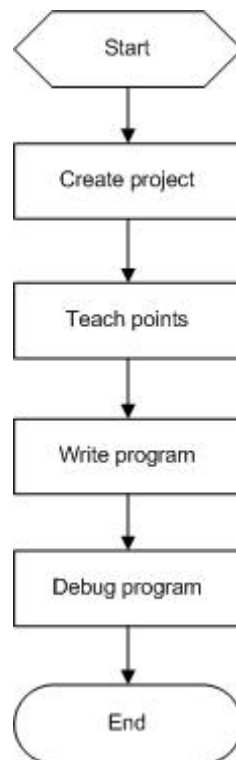


Figure 3.9 Programming process

3.6.3.1 Creating Project

Prerequisites

The robot has been powered on.

Procedure

Step 1 Click .

The programming page is displayed, as shown in Figure 3.10.

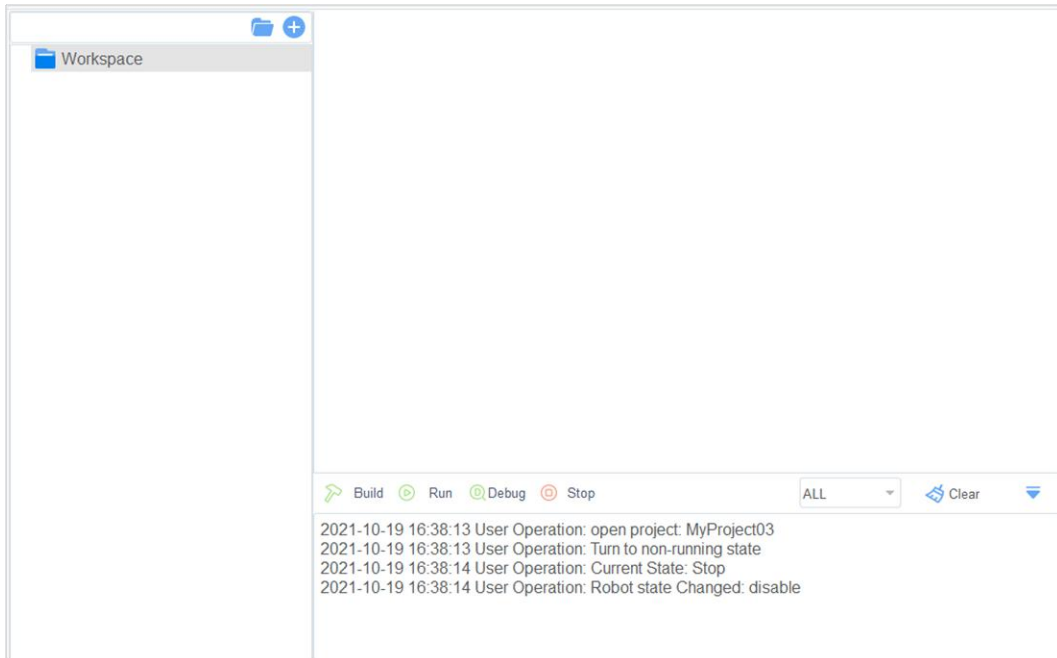



Figure 3.10 Programming page

Step 2 Click  to enter the project creating page. Input the project name, and you can also select a template. Click **OK**.

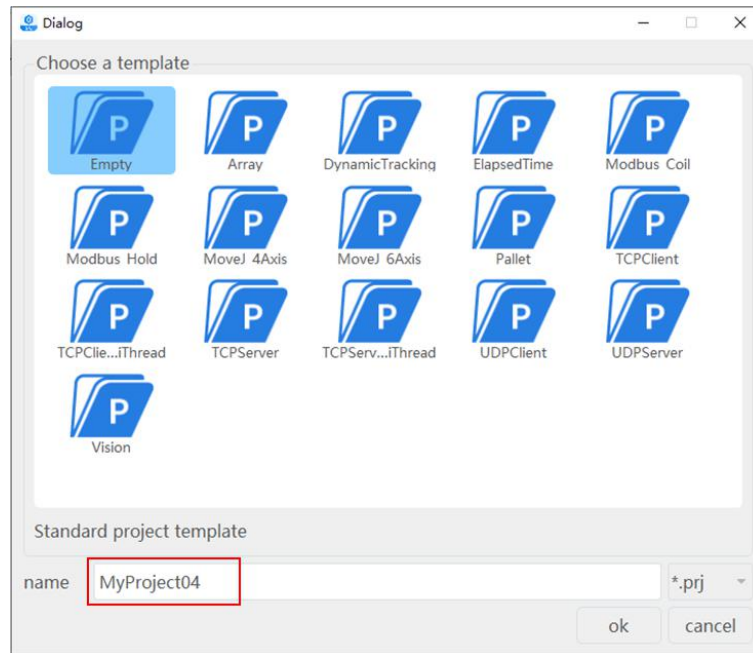


Figure 3.11 Create a project

Step 3 Set the number of threads based on site requirements, as shown in Figure 3.12. Click **thread** and right-click **New thread file**.

The maximum number of threads is 5.

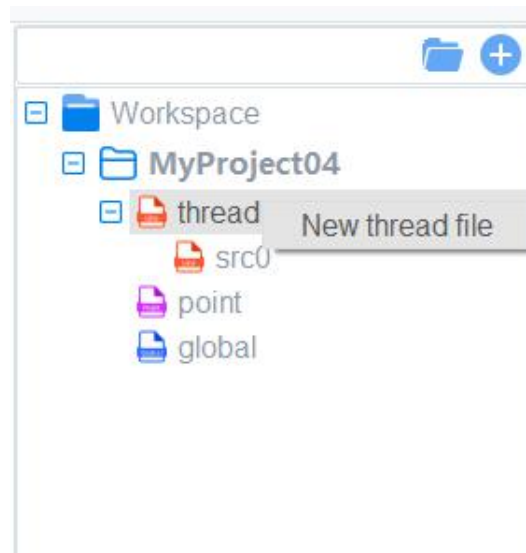


Figure 3.12 Create a project

Step 4 (Optional) Import the existing taught positions list.

If you want to reuse a taught position list from an existing project, please right-click **point** and click **Import points File**, as shown in Figure 3.13.

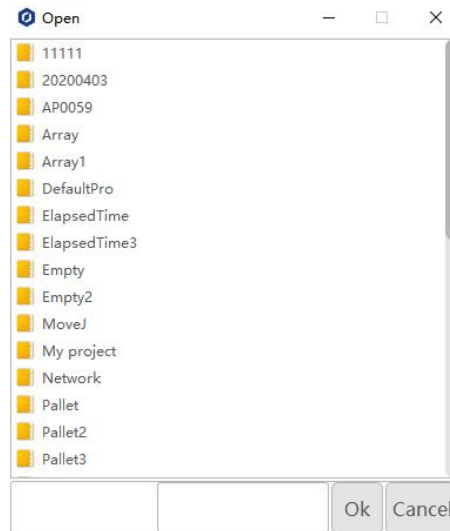


Figure 3.13 Import the existing teaching points list

3.6.3.2 Import Project

If you need to reuse project files of other MG400, you can export project files of other MG400 to local computer and then import them into the current MG400 from local computer.

Prerequisites

The robot has been powered on.

Procedure

Step 1 Click **Workspace** and right-click **Import Project**.

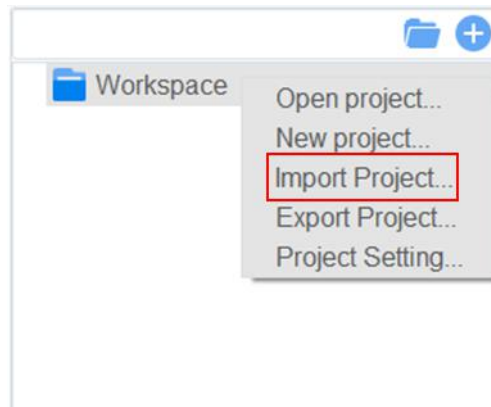


Figure 3.14 Import Project

Step 2 Select a project to be imported.

In the **Import Project** page, there are two files: **prj.json** and **point.json**. Please select the project file **prj.json**.

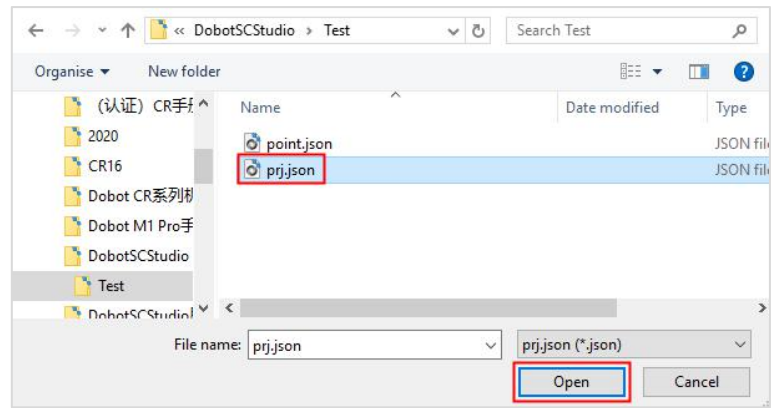


Figure 3.15 Select a project

Step 3 Click Open.

The imported project files are displayed.

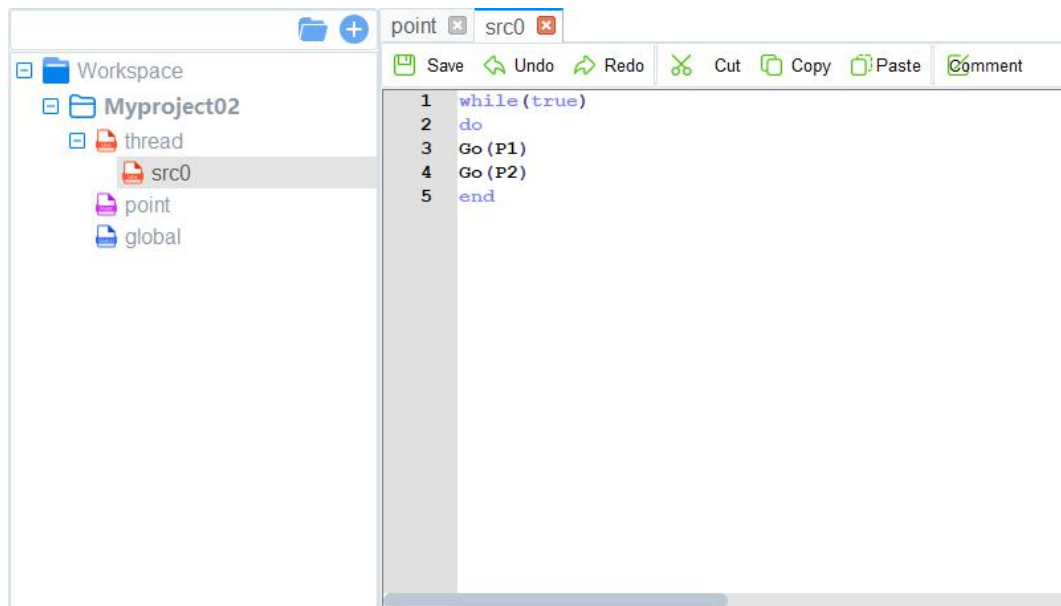


Figure 3.16 Display the project

NOTE

If the project has already been saved or imported into MG400, it is not allowed to import the same project, and a warning window will pop up indicating that you need to re-select a new project file to be imported.

3.6.3.3 Teaching points

Prerequisites

The project has been created or imported.


Procedure


After creating a project, please teach positions on the **point** page for calling commands when programming a robot. If the existing taught positions list has been imported, this operation can be

skipped.

Step 1 Enable MG400.



Step 2 Click  to move the robot to a point.

Step 3 Double click **point** to enter point page and click  **Add** to add a teaching point.








The information of teaching point is displayed on the **point** page, as shown in Figure 3.17.

Tool is the Tool coordinate system, **User** is the User coordinate system.


									
	No.	Alias	X	Y	Z	R	Arm	Tool	User
1	P1		-17.0139	-348.4705	986.5601	-90.0000	Right	No.0	No.0
2	P2		-74.4932	-400.2468	979.8826	-90.0000	Right	No.0	No.0

Figure 3.17 Teaching points list


Table 3.5 Button description

Button	Description
 Add	Add a point
 Delete	Delete a point
 Cover	Cover a point. Select a teaching point, after jogging the robot to a point, click the icon to cover the selected teaching point
 RunTo	Run to a point, select a point, click the button to run the robot to this point
 Save	Save teaching point
 Undo	Cancel
 Redo	Recover

- You can select a teaching position and double-click the parameters to modify the relevant information.

- Also, you can select a teaching position and click  **Cover** to cover the current teaching position.

Step 4 Add points by referring to Step 2 and Step 3.

Step 5 Click  **Save** to save the teaching points.

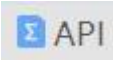
3.6.3.4 Writing a Program

Prerequisites

- The project has been created or imported.
- The points have been taught.

Procedure

In MG400, common commands for programming with Lua language have been encapsulated. Please see 4Program Language.

Step 1 Click , and the command list will display on the right side of the page, as shown in Figure 3.18.

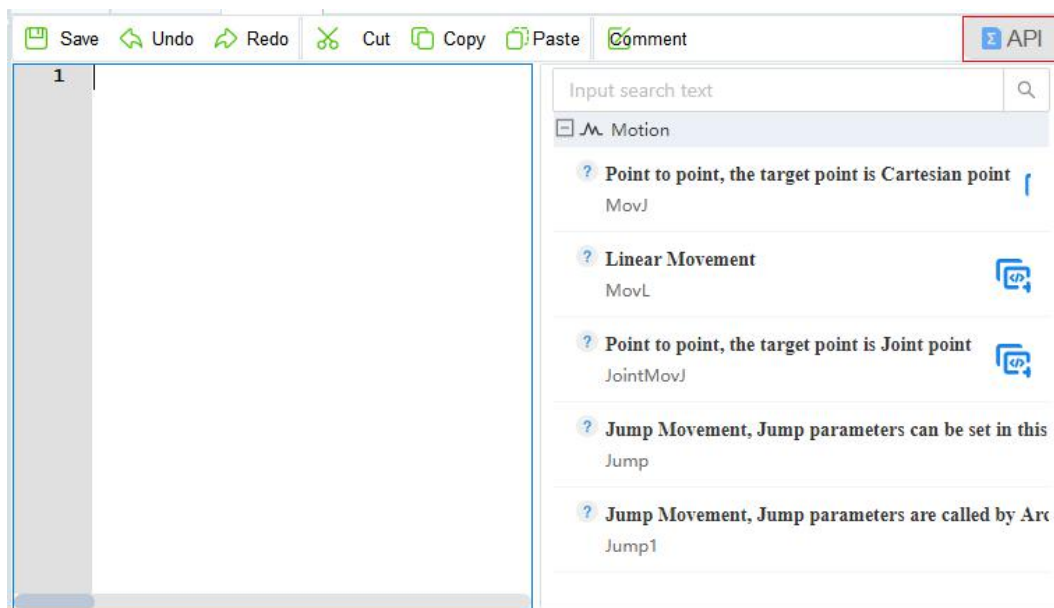



Figure 3.18 Command list

Step 2 Select the commands from the list. Double-click the command and it will display on the script area.

Step 3 Click  **Save** after you finish the programming.

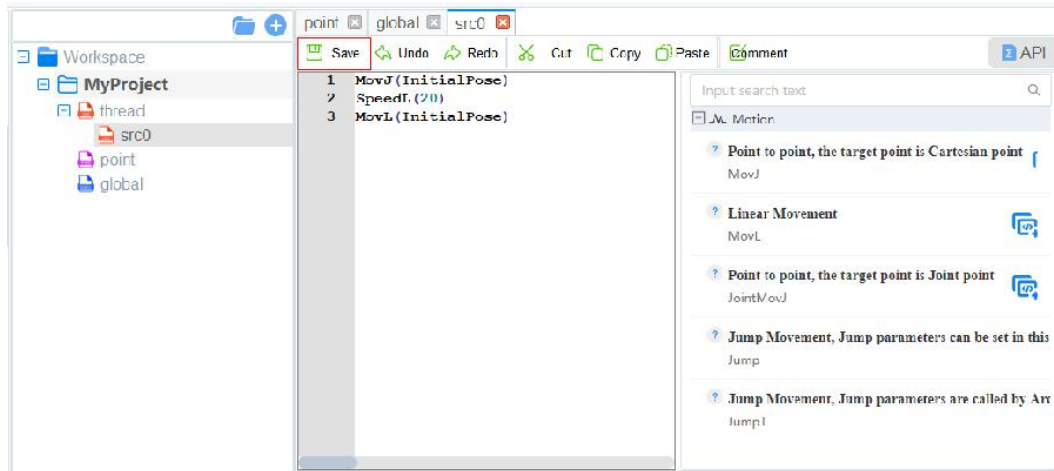


Figure 3.19 Save the project

3.6.3.5 Debugging Program

Step 1 Click  to enable the motor.

Now, the programming page is as shown in Figure 3.20.

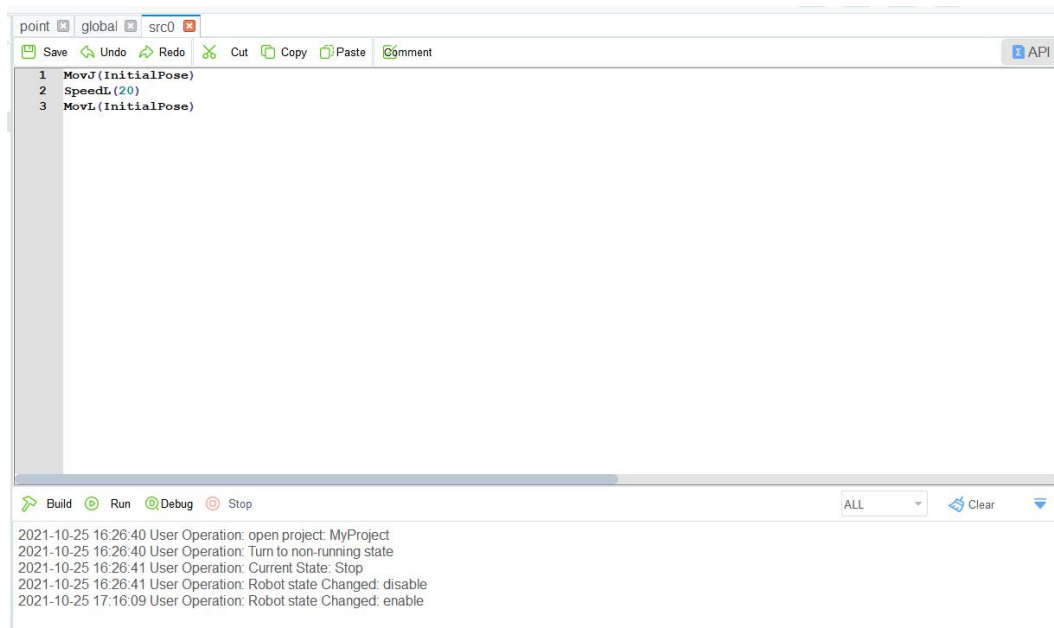



















Figure 3.20 Programming page

Table 3.6 lists the description on the program-running buttons which are shown in Figure 3.20

Table 3.6 Program-running button description

Icon	Description
 Build	Build program Check if the code is correct
 Run	Once-click run After clicking this button,  Run turns into  Pause and the program starts running If you need to pause the running program, please click 
 Debug	Start to run a program Click once: Start to debug a program,  Step appears on the screen. Click  Step : Start to run a program
 Stop	Stop the running program
 Monitor	After you click  Step in debugging the program,  Monitor appears on the screen to monitor the running of program
 Resume	After you click  Pause,  Resume appears on the screen. You can click it to continue running the program.

Step 2 Click  Run to start debugging the program.

- If you want to run a program step by step, click  Debug . Click  Step after it appears on the screen.

3.6.3.6 Export Project

Project is saved in MG400 by default, and you can import the project to a local directory.

Step 1 Click **Workspace** and right-click **Export Project**.

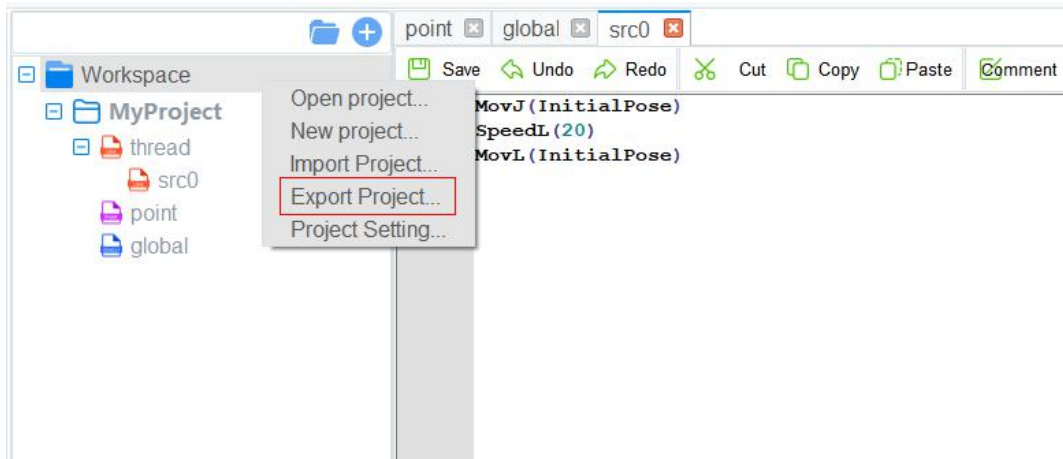


Figure 3.21 Export Project

Step 2 In the project list, click the project to be exported.
For example, click “Factory Test 1”.

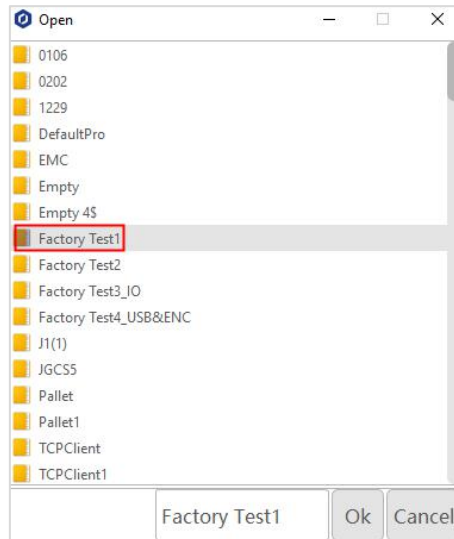


Figure 3.22 Select Project Files

Step 3 Select a right path, then click **Select Folder**.

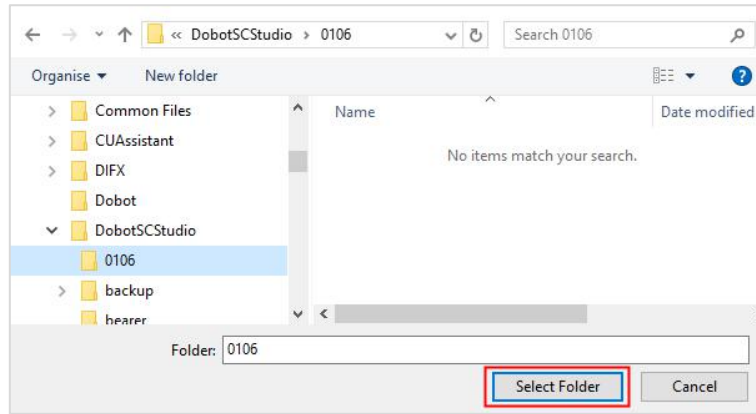


Figure 3.23 Save path

3.7 Parameter

Before teaching or running robot programs, a series of settings are required, including motion parameter setting, language selecting, user mode selecting and process setting, etc.

3.7.1 Setting User Coordinate System

When the position of workpiece is changed or a robot program needs to be reused in multiple processing systems of the same type, you can create coordinate systems on the workpiece to simplify programming. There are totally 10 groups of User coordinate systems, of which the first one is defined as the Base coordinate system by default and cannot be changed. And the others can be customized by users.



NOTICE

When creating a User coordinate system, please make sure that the reference coordinate system is the Base coordinate system. Namely, the User coordinate system

icon should be **User: 0** when creating a User coordinate system.

User coordinate system is created by two-point calibration method. Move the robot to three points **P0(x0, y0, z0)**, **P1(x1, y1, z1)**. Point P0 is defined as the origin and the line from point P0 to Point P1 is defined as the positive direction of X-axis. And then the Y-axis and Z-axis can be defined based on the right-handed rule, as shown in Figure 3.24.

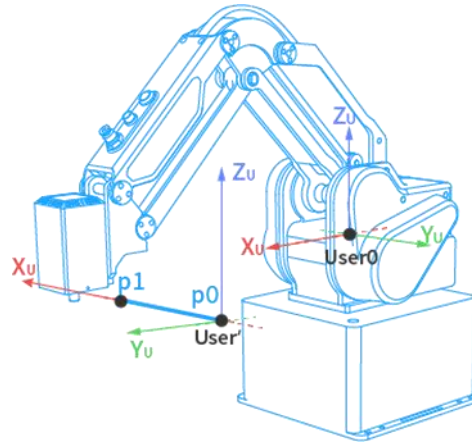



Figure 3.24 Two point calibration

Take the establishment of User 2 coordinate system as an example based on two-point calibration method.

Prerequisites

- The robot has been powered on.
- MG400 has been enabled.
- The robot is in the Cartesian coordinate system.

Procedure

Step 1 Click  > **Parameter** > **GlobalCoordinate** > **Coordinate User**.

The **Coordinate User** page is displayed, as shown in Figure 3.25.

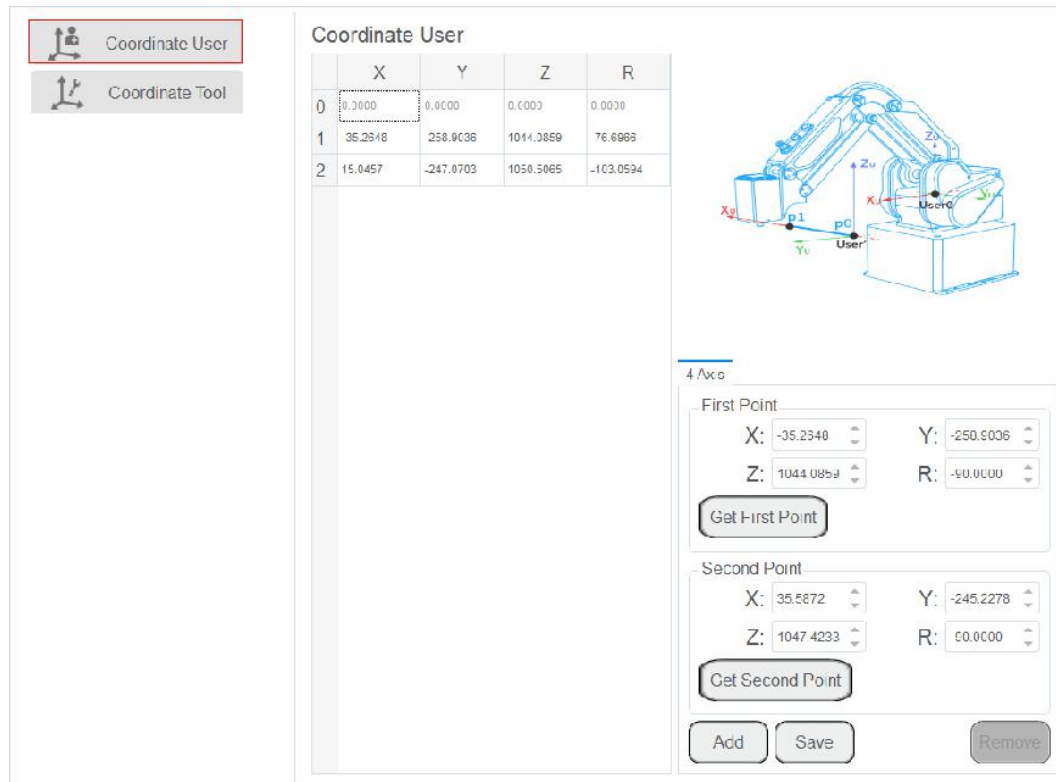


Figure 3.25 User coordinate system page

- Step 2** Jog the robot to the first point, then click **Get First Point** on the **P1** tab to obtain the coordinates of the first point.
- Step 3** Jog the robot to the second point, then click **Get Second Point** on the **P2** tab to obtain the coordinates of the second point.
- Step 4** Click **Cover** and **Save** to generate the User 2 coordinate system, as shown in Figure 3.26. If you do not select an existing coordinate system, you can repeat Step1~Step3, and then click **Add** and **Save**, as shown in Figure 3.27.

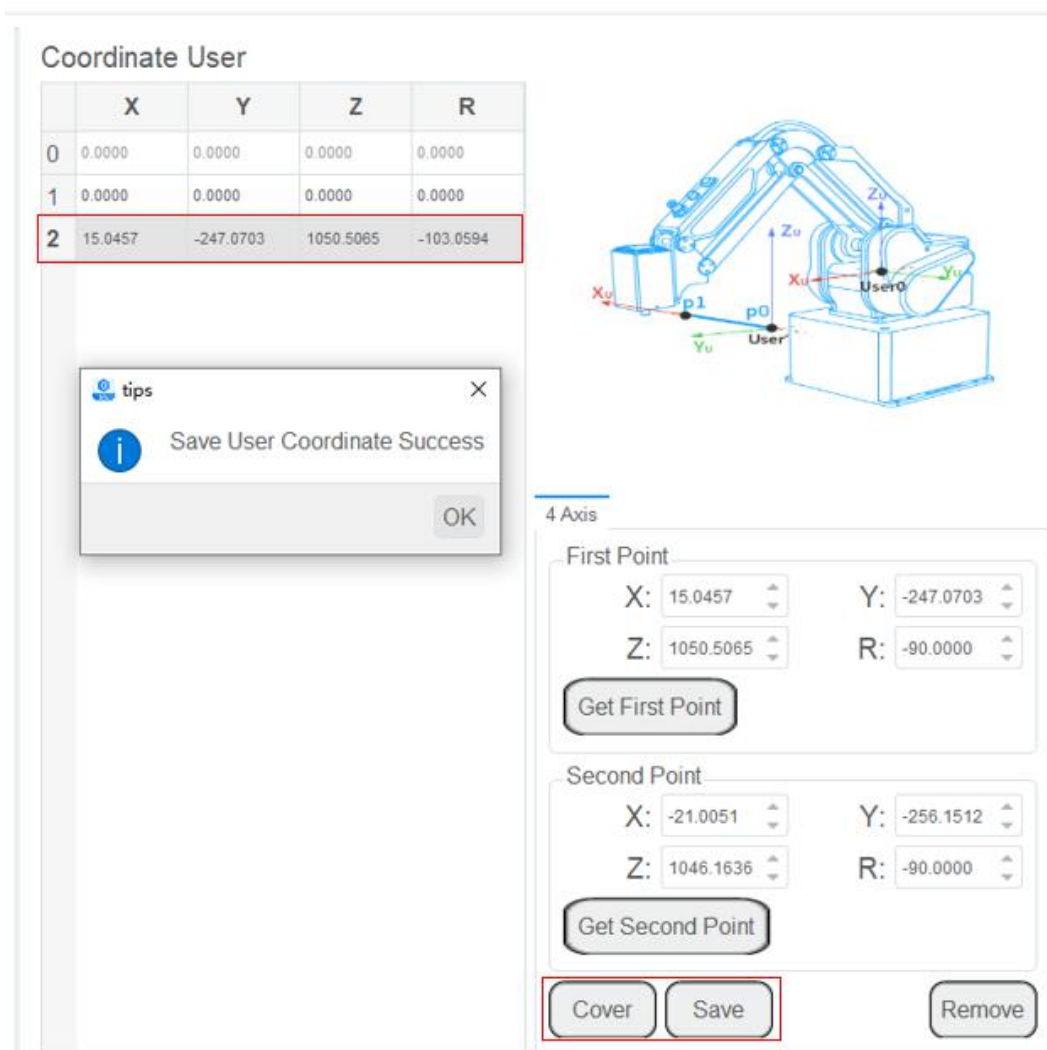


Figure 3.26 Click Cover to generate a coordinate system

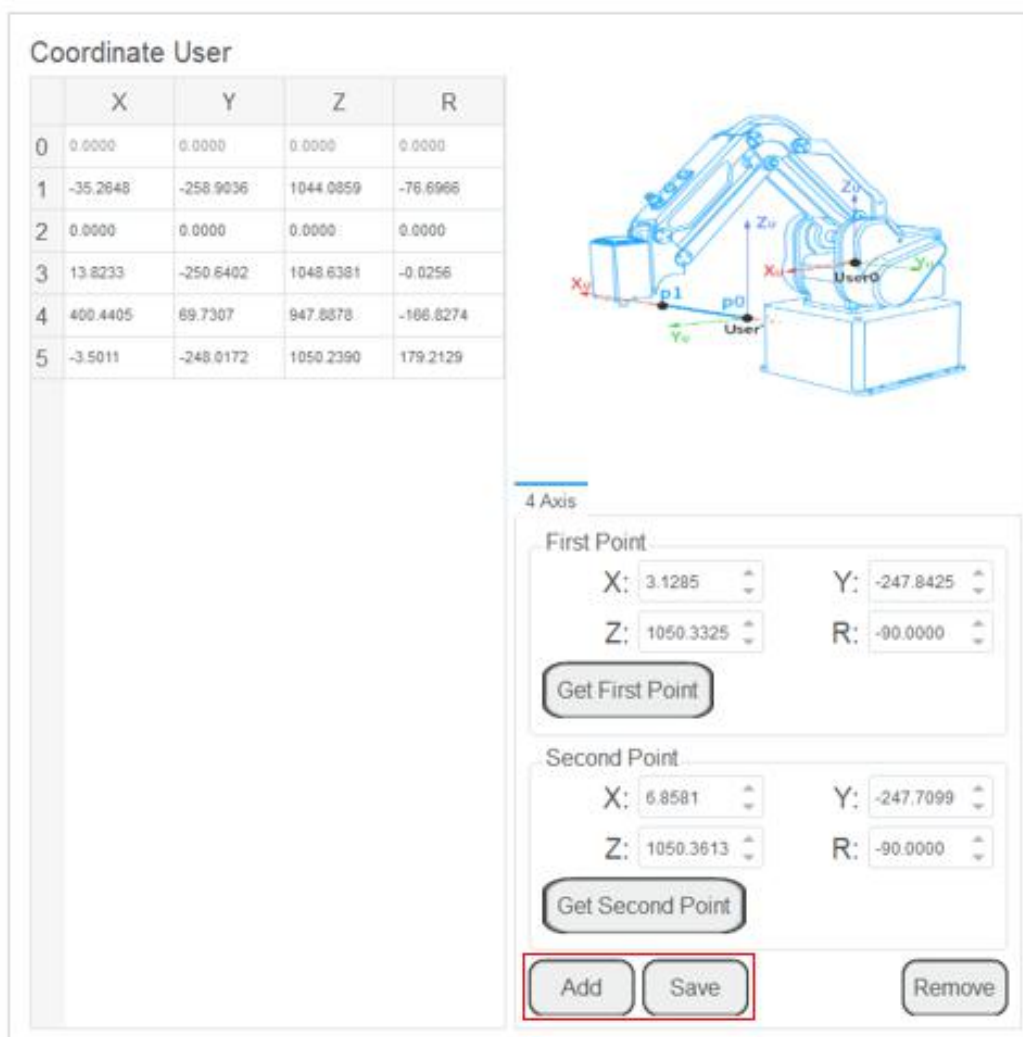


Figure 3.27 Click Add to generate a coordinate system

Step 5 Select **User 2** on Jog interface.

You can use the **User 2** coordinate system for teaching and programming.

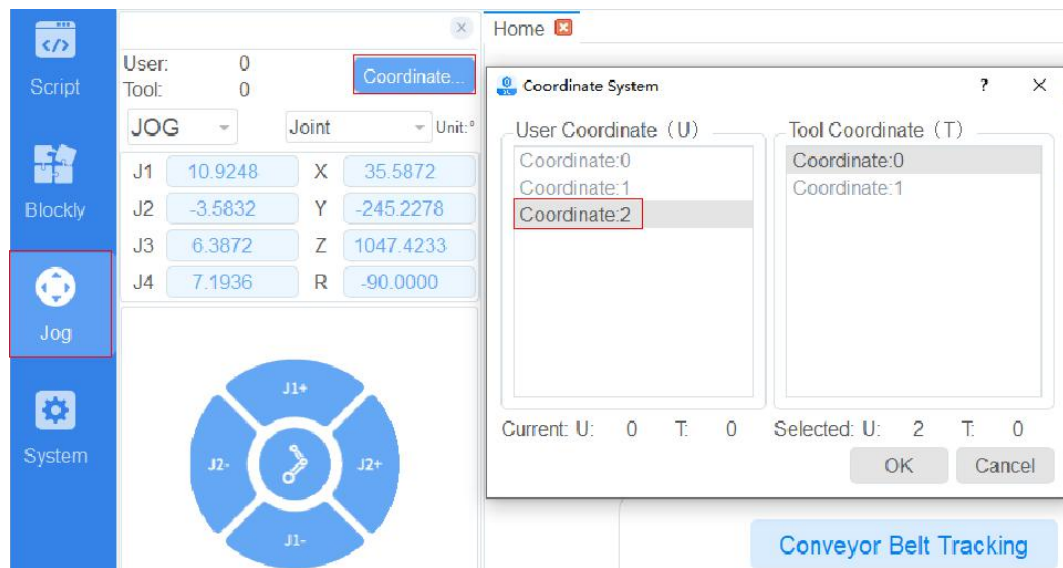


Figure 3.28 Select user coordinate system

3.7.2 Setting Tool Coordinate System

When an end effector such as welding gun, gripper is mounted on the robot, the Tool coordinate system is required for programming and operating a robot. For example, you can use multiple grippers to carry multiple workpieces simultaneously to improve the efficiency by setting each gripper to a Tool coordinate system.

There are totally 10 groups of Tool coordinate systems. Tool 0 coordinate system is the predefined Tool coordinate system which is located at the robot flange and cannot be changed.

NOTICE

When creating a Tool coordinate system, please make sure that the reference coordinate system is the predefined Tool coordinate system. Namely, the Tool coordinate system icon should be **Tool: 0** when creating a Tool coordinate system.

Tool coordinate system of robot is created by two-point calibration method: After an end effector is mounted, please adjust the direction of this end effector to make the TCP (Tool Center Point) align with the same point (reference point) in two different directions, for obtaining the position offset to generate a Tool coordinate system, as shown in Figure 3.29.

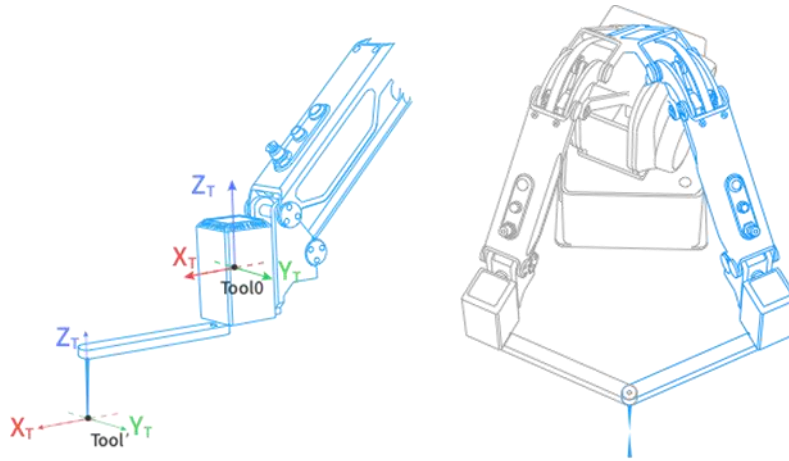


Figure 3.29 Two points calibration method (TCP+ZX)


Take the establishment of Tool 1 coordinate system as an example.

Prerequisites

- The robot has been powered on.
- MG400 has been enabled.
- The robot is in the Cartesian coordinate system.

Procedure

Step 1 Mount an eccentric end effector on the robot. The detailed instructions are not described in this topic.

Step 2 Click  > **Parameter** > **GlobalCoordinate** > **Coordinate Tool**.

The Coordinate Tool page is displayed in Figure 3.30.

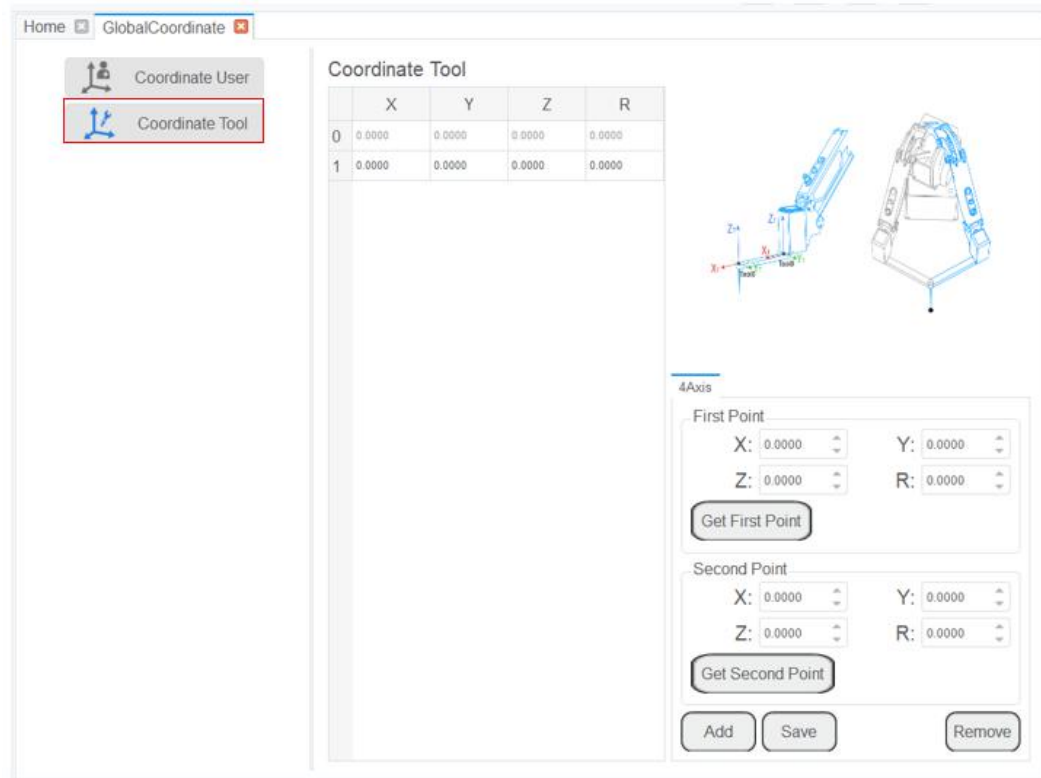


Figure 3.30 Tool Coordinate page

- Step 3** Jog the robot to the reference point in the first direction, then click **Get First Point** to obtain the coordinates of the first point.
- Step 4** Jog the robot to the reference point in the second direction, then click **Get Second Point** to obtain the coordinates of the second point.
- Step 5** Click **Save** to generate the Tool 1 coordinate system.

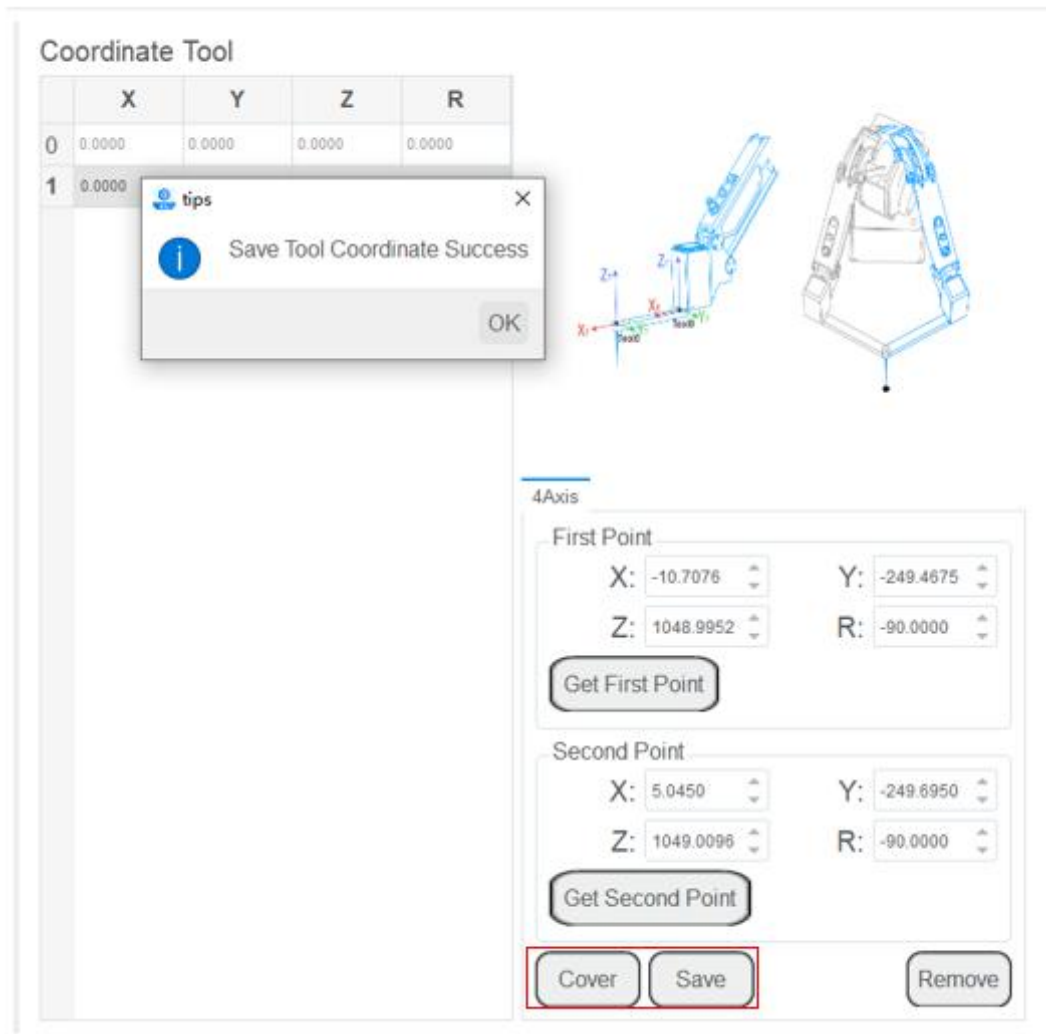


Figure 3.31 Tool 2 coordinate system

3.7.3 I/O Monitor


You can set or monitor the I/O status of the robot and robot end on this page. Click  > **Parameter > IOMonitorPro** to enter the I/O monitor page, as shown in Figure 3.32.



Figure 3.32 I/O monitor page

There are three features: Output, monitor and simulation.

- Output: Set the digital output.
- Monitor: Check the status of the input and output.
- Simulation: Simulate the digital input for debugging and running program, as shown in Figure 3.33.

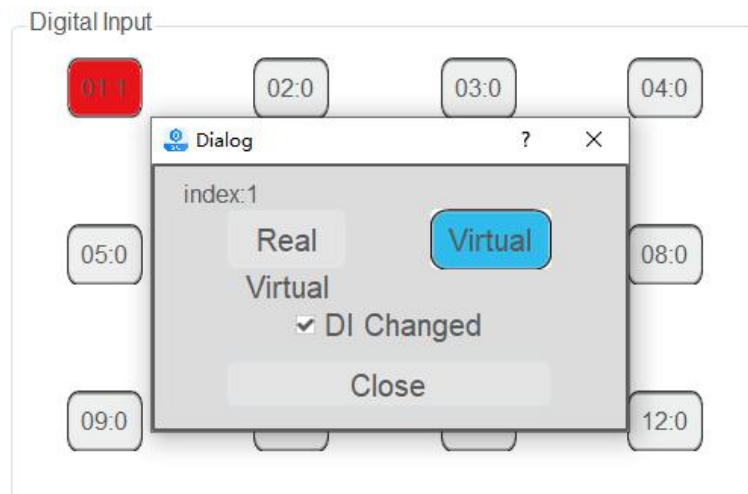



Figure 3.33 Simulation

3.7.4 Controller Setting

3.7.4.1 Reboot

When the controller firmware has been updated or the robot is abnormal, you need to reboot

the robot. Now you can click  to reboot it on the **Parameter > ControllerSetting > Reboot** page.

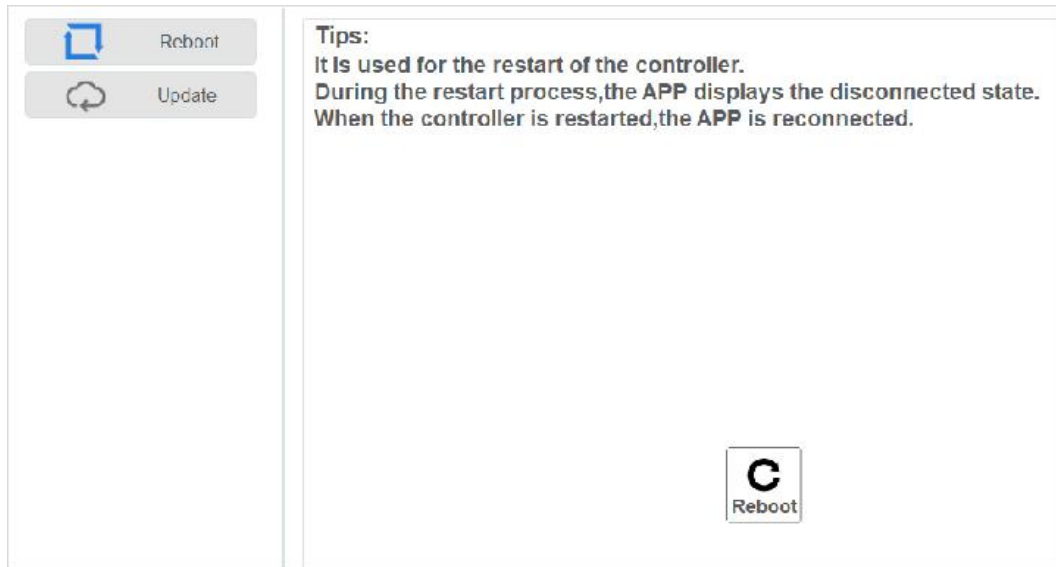


Figure 3.34 Reboot

3.7.4.2 Update

When the controller firmware needs to be updated, you can import the latest firmware on this page. After importing the firmware, please reboot the robot.

Please contact the Dobot support engineer to obtain the latest firmware.

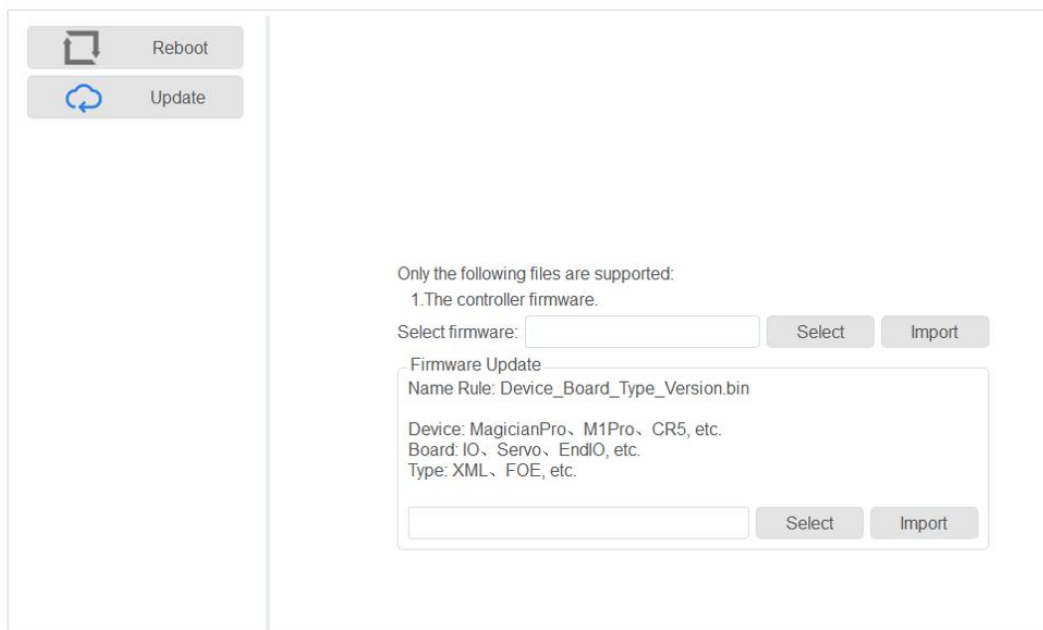


Figure 3.35 Update firmware

3.7.5 Remote Control

External equipment can send commands to a robot by different remote control modes, such as remote I/O mode and remote Modbus mode. The default mode is Teaching mode when the robot is shipped out. When you need to set the remote mode, please set it on the DobotSCStudio with MG400 in the disabled state.

NOTICE

- Robot rebooting is not required when switching the remote mode.
- The emergency stop switch on the hardware is always available no matter what mode MG400 is in.
- When MG400 is running in remote control mode, if the start signal is triggered on the external device, MG400 will be automatically enabled and run according to the selected project file.

3.7.5.1 Remote I/O

When the remote mode is I/O mode, external equipment can control a robot in this mode. The specific description on I/O interface is shown in Table 3.7.

Table 3.7 Description on I/O interface

I/O interface	Description
Input (For external control)	
DI 11	Clear alarm
DI 12	Continue to run
DI 13	Pause running in the I/O mode
DI 14	Stop running and exit the I/O mode
DI 15	Start to run in the I/O mode
DI 16	Emergency stop and exit the I/O mode
Output (For displaying the status)	
DO 13	Ready status
DO 14	Pause status
DO 15	Alarm status
DO 16	Running status

NOTICE

All input signals are rising-edge triggered. A change from low level to high level is effective.

Prerequisites

- The project to be running in the remote mode has been prepared.
- The external equipment has been connected to MG400 by the I/O interface. The specific I/O interface description is shown in Table 3.7.
- The robot has been powered on.

NOTE

The details on how to connect external equipment and use it are not described in this topic.

Procedure

Step 1 Click  > **Parameter** > **RemoteControl**.

The remote control page is displayed, as shown in Figure 3.36.

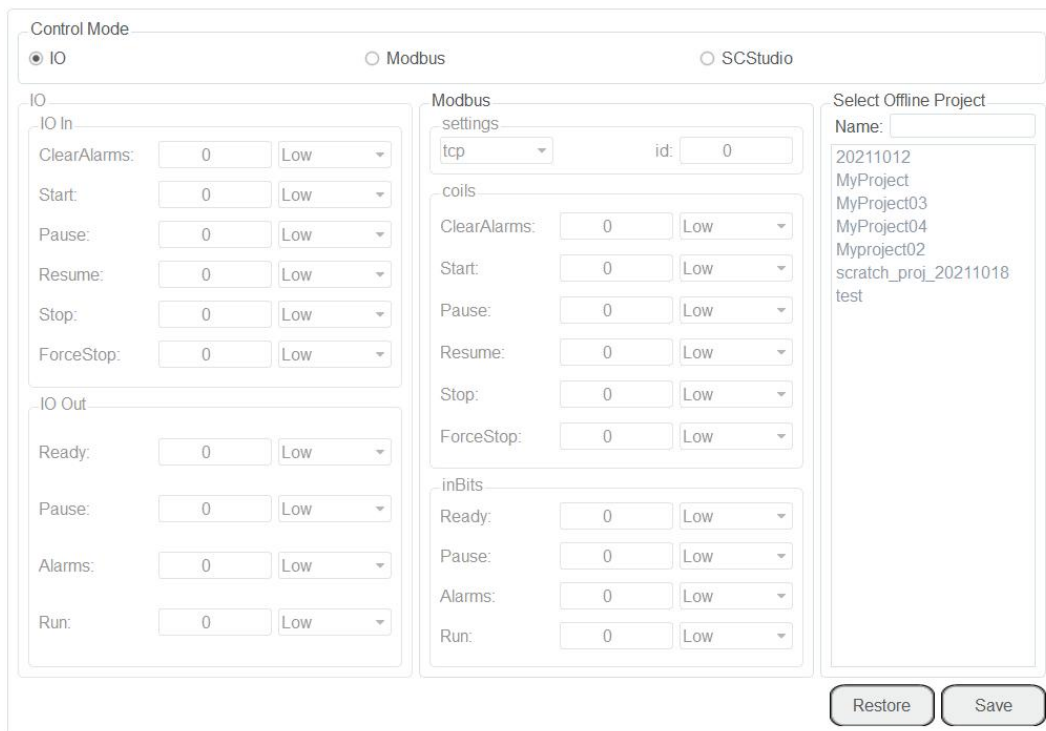


Figure 3.36 Remote control page

Step 2 Select **IO** on the **Control Mode** section and select the offline project on the **Select Offline Project** section, click **Save**.

The **Save success, now remote control mode is IO** page is displayed.

Right now, only the emergency stop button is available.

Step 3 Trigger the starting signal on the external equipment.

The robot will move as the selected offline project. If the stop signal is triggered, the robot will exit remote IO mode.

3.7.5.2 Remote Modbus

When the remote mode is Modbus mode, external equipment can control a robot in this mode. For details about Modbus registers, please see *4.15.1 Description on Modbus Register*. The specific description on Modbus register is shown in Table 3.8.

Table 3.8 Specific Modbus register description

Register address (Take a PLC as an example)	Register address (MG400)	Description
Coil register		
00001	0	Start running in the remote Modbus mode
00002	1	Pause running in the remote Modbus mode
00003	2	Continue to run
00004	3	Stop to run and exit the remote Modbus mode
00005	4	Emergency stop and exit the remote Modbus mode
00006	5	Clear alarm
Discrete input register		
10001	0	Auto-exit
10002	1	Ready status
10003	2	Pause status
10004	3	Running status
10005	4	Alarm status

Prerequisites

- The project to be running in the remote mode has been prepared.
- The robot has been connected to the external equipment with the Ethernet2 interface. The default IP address is **192.168.2.6**. You can connect them directly or via a router, please select based on site requirements.

The IP address of MG400 must be in the same network segment of the external equipment without conflict. You can modify the IP address on the **ToolConfig > NetworkSetting** page; the default port is **502** and cannot be modified.

- The robot has been powered on.

NOTE

The details on how to connect external equipment and use it are not described in this topic.

Procedure

Step 1 Click  > **Parameter** > **RemoteControl**.

The remote control page is displayed, as shown in Figure 3.37.

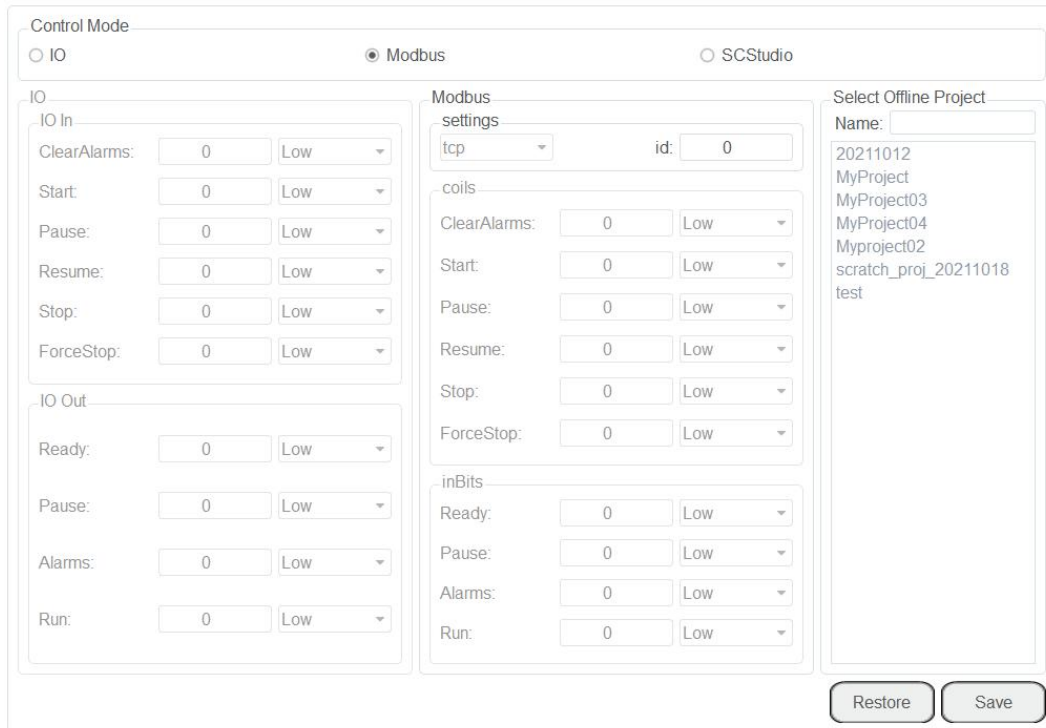


Figure 3.37 Remote control page

Step 2 Select **Modbus** on the **Control Mode** section and select the offline project on the **Select Offline Project** section, and click **Save**.

The **Save success, now remote control mode is Modbus** page is displayed.

Step 3 Trigger the starting signal on the external equipment.

The robot will move as the selected offline project. If the stop signal is triggered, the remote Modbus mode will be invalid.

3.7.6 RobotParams

You can set the velocity, acceleration or other parameters in different coordinate systems when jogging a robot or running robot programs. After setting the parameters, please click **Save**.

Click  > **Parameter** > **RobotParams** to enter **RobotParams** interface.

- Teach Joint Parameter: Set the maximum velocity and acceleration in the Joint coordinate system when jogging a robot. The jogging parameters of a robot in the Joint coordinate system are as shown in Figure 3.38.

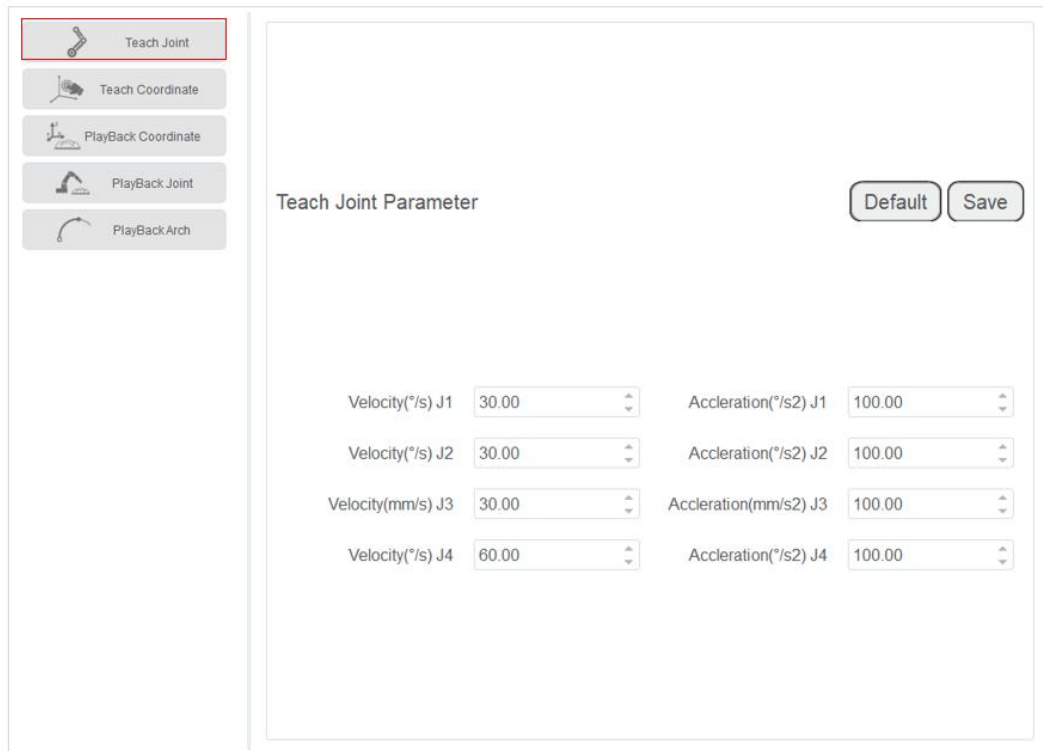


Figure 3.38 Jogging parameters in the Joint coordinate system

The relations between the actual velocity of each joint and the maximum velocity are shown as follows:

- Jog velocity of each joint = maximum velocity of each joint * global velocity rate
- Jog acceleration of each joint = maximum acceleration of each joint * global velocity rate

NOTE

- You can set the global velocity rate on the main page. For details, see 3.2 *Setting Global Velocity Rate*.
 - You can set the percentage of commands by calling speed commands when programming.
- Teach Coordinate Parameter: Set the maximum velocity and acceleration in the Cartesian coordinate system when jogging a robot. The jogging parameters of a robot in the Cartesian coordinate system are as shown in Figure 3.39.

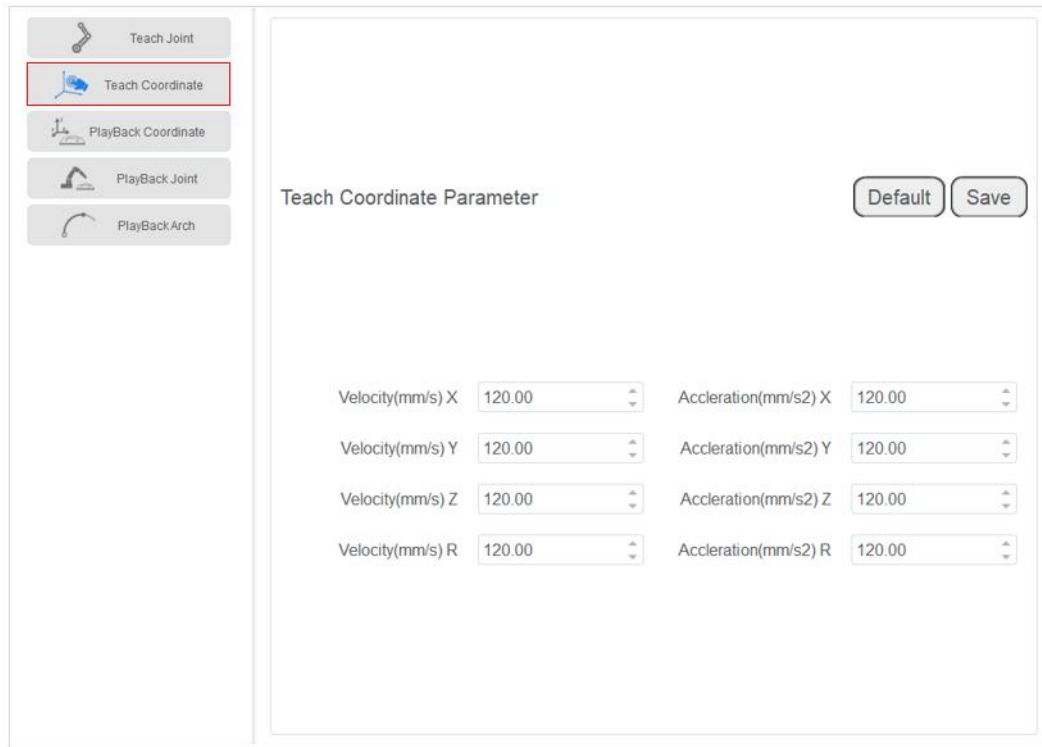


Figure 3.39 Jogging parameters in the Cartesian coordinate system

The relations between the actual velocity of each Cartesian axis and the maximum velocity are shown as follows:

- Jog velocity of each Cartesian axis = maximum velocity of each Cartesian axis * global velocity rate
- Jog acceleration of each Cartesian axis = maximum acceleration of each Cartesian axis * global velocity rate
- Playback Coordinate Parameter: Set the maximum velocity, acceleration and jerk in the Cartesian coordinate system when running robot programs. The playback parameters of a robot in the Cartesian coordinate system are as shown in Figure 3.40.

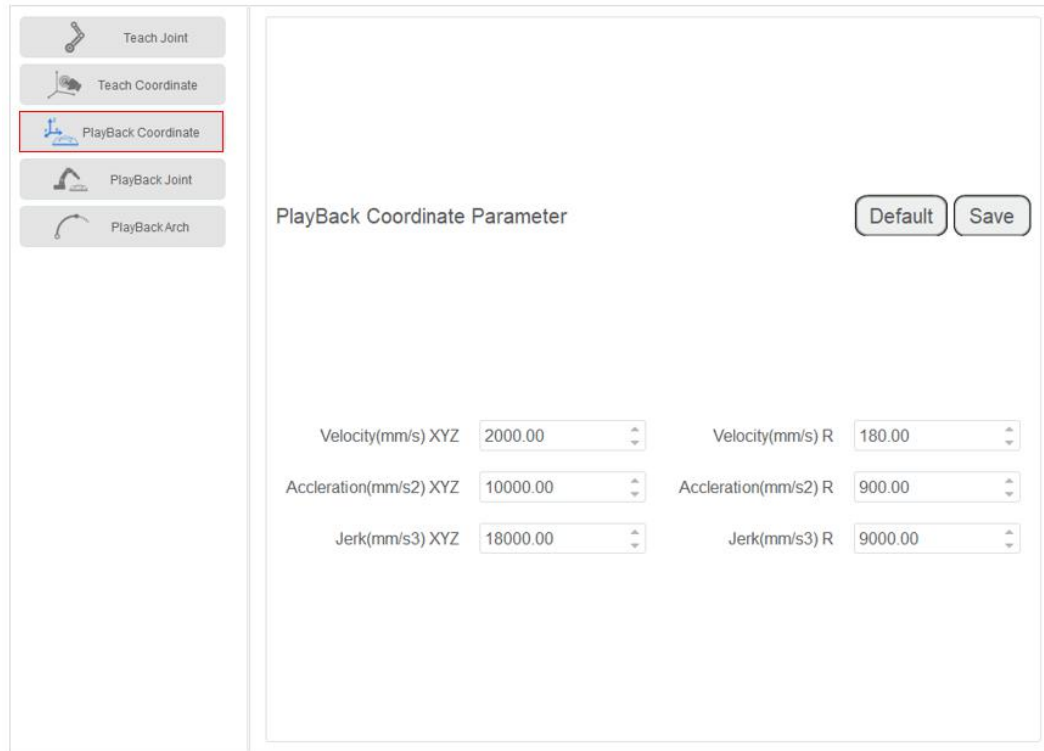


Figure 3.40 Playback parameters in the Cartesian coordinate system

The relations between the actual velocity of each Cartesian axis and the maximum velocity are shown as follows:

- Playback velocity of each Cartesian axis = maximum velocity of each Cartesian axis * global velocity rate * percentage set in commands
- Playback acceleration of each Cartesian axis = maximum acceleration of each Cartesian axis * global velocity rate * percentage set in commands
- Playback jerk of each Cartesian axis = maximum jerk of each Cartesian axis * global velocity rate * percentage set in commands

- Playback Joint Parameter: Set the maximum velocity, acceleration, and jerk in the Joint coordinate system when running robot programs. The playback parameters of a robot in the Joint coordinate system are as shown in Figure 3.41.

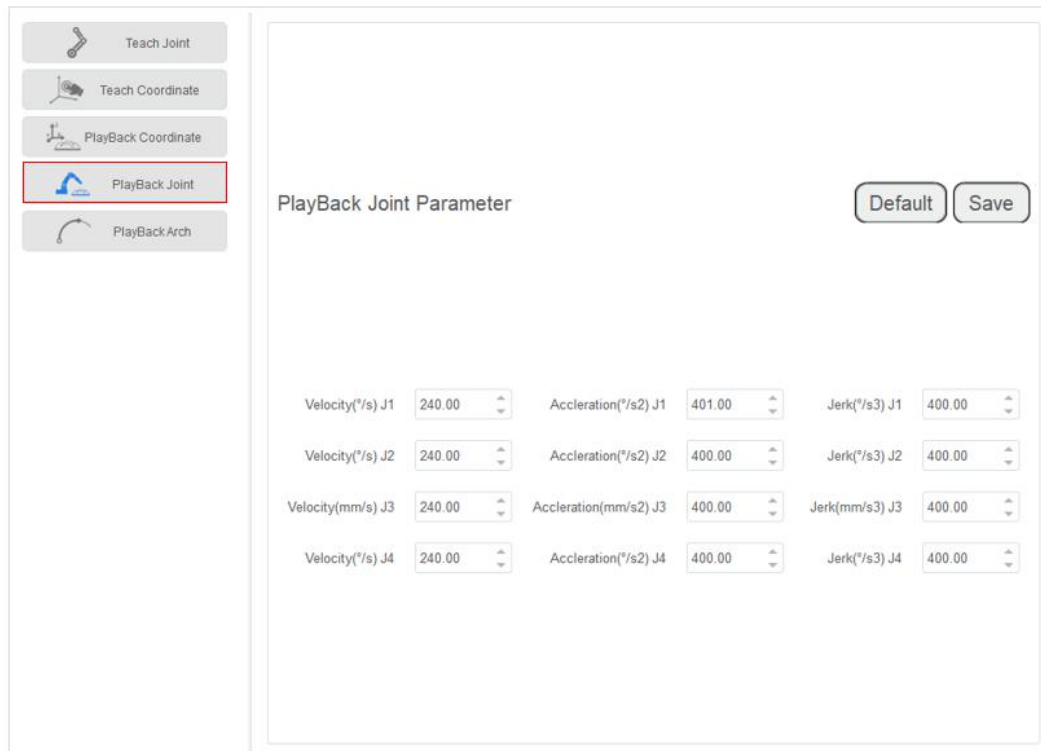


Figure 3.41 Playback parameters in the Joint coordinate system

The relations between the actual velocity of each joint and the maximum velocity are shown as follows:

- Playback velocity of each joint = maximum velocity of each joint * global velocity rate * percentage set in commands
- Playback acceleration of each joint = maximum acceleration of each joint * global velocity rate * percentage set in commands
- Playback jerk of each joint = maximum jerk of each joint * global velocity rate * percentage set in commands

- Playback Arch Parameter: If the motion mode is **Jump** when running robot programs, you need to set **StartHeight**, **EndHeight**, and **zLimit**.

You can set 10 sets of Jump parameters. Please set and select any set of parameters for calling Jump command during programming, as shown in Figure 3.42.

Teach Joint

Teach Coordinate

PlayBack Coordinate

PlayBack Joint

PlayBack Arch

PlayBack Arch Parameter (Unit: mm)

Default

Save

No.0	StartHeight	<input type="text" value="252.25"/>	EndHeight	<input type="text" value="260.00"/>	zLimit	<input type="text" value="280.00"/>
No.1	StartHeight	<input type="text" value="10.00"/>	EndHeight	<input type="text" value="10.00"/>	zLimit	<input type="text" value="20.00"/>
No.2	StartHeight	<input type="text" value="20.00"/>	EndHeight	<input type="text" value="1100.00"/>	zLimit	<input type="text" value="50.00"/>
No.3	StartHeight	<input type="text" value="20.00"/>	EndHeight	<input type="text" value="20.00"/>	zLimit	<input type="text" value="1100.00"/>
No.4	StartHeight	<input type="text" value="20.00"/>	EndHeight	<input type="text" value="20.00"/>	zLimit	<input type="text" value="1300.00"/>
No.5	StartHeight	<input type="text" value="20.00"/>	EndHeight	<input type="text" value="20.00"/>	zLimit	<input type="text" value="50.00"/>
No.6	StartHeight	<input type="text" value="21.00"/>	EndHeight	<input type="text" value="19.00"/>	zLimit	<input type="text" value="50.00"/>
No.7	StartHeight	<input type="text" value="20.00"/>	EndHeight	<input type="text" value="20.00"/>	zLimit	<input type="text" value="50.00"/>
No.8	StartHeight	<input type="text" value="20.00"/>	EndHeight	<input type="text" value="20.00"/>	zLimit	<input type="text" value="50.00"/>
No.9	StartHeight	<input type="text" value="20.00"/>	EndHeight	<input type="text" value="20.00"/>	zLimit	<input type="text" value="50.00"/>

Figure 3.42 Jump parameters

3.7.7 RobotSetting

3.7.7.1 Zero Calibration

After some parts (motors, reduction gear units) of the robot have been replaced or the robot has been hit, the origin of the robot will be changed. You need to reset the origin.

Step 1 Click  > **Parameter** > **RobotSetting** > **Zero Calibration** to enter Zero Calibration interface, as shown in Figure 3.43.

Adjust each axis of M1 to the mechanical homing point according to the prompts.

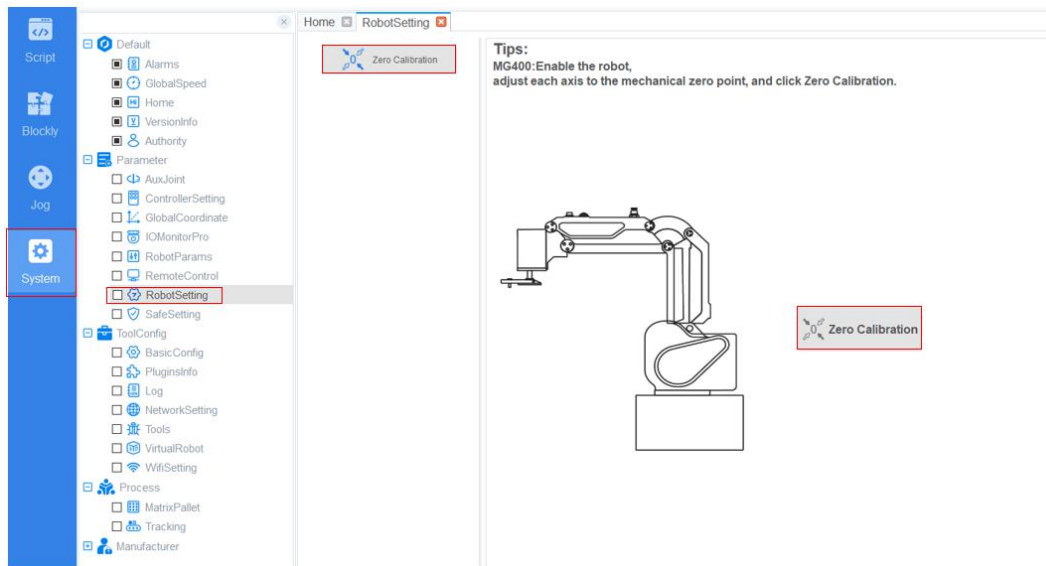


Figure 3.43 Zero Calibration

Step 2 Click the **Zero Calibration**.

Step 3 Click **Yes** in the current prompt window.



Figure 3.44 Confirm the zero calibration

3.7.8 SafeSetting

3.7.8.1 Collision Detection

Collision detection is mainly used for reducing the impact on the robot arm, to avoid damage to the robot arm or external equipment. If the collision detection is activated, the robot arm will stop running automatically when the robot arm hits an obstacle.

You can enable collision detection function on the **Parameter > SafeSetting > Collision Detection** page and set the collision level. Meanwhile, you can select **Automatic restart 5 seconds after collision**, namely, when the robot arm stops for five seconds after hitting an obstacle, you can drag the robot to a safe position.

There are five collision levels to select. The higher the level is, the less force the robot arm needs to stop after collision detection.

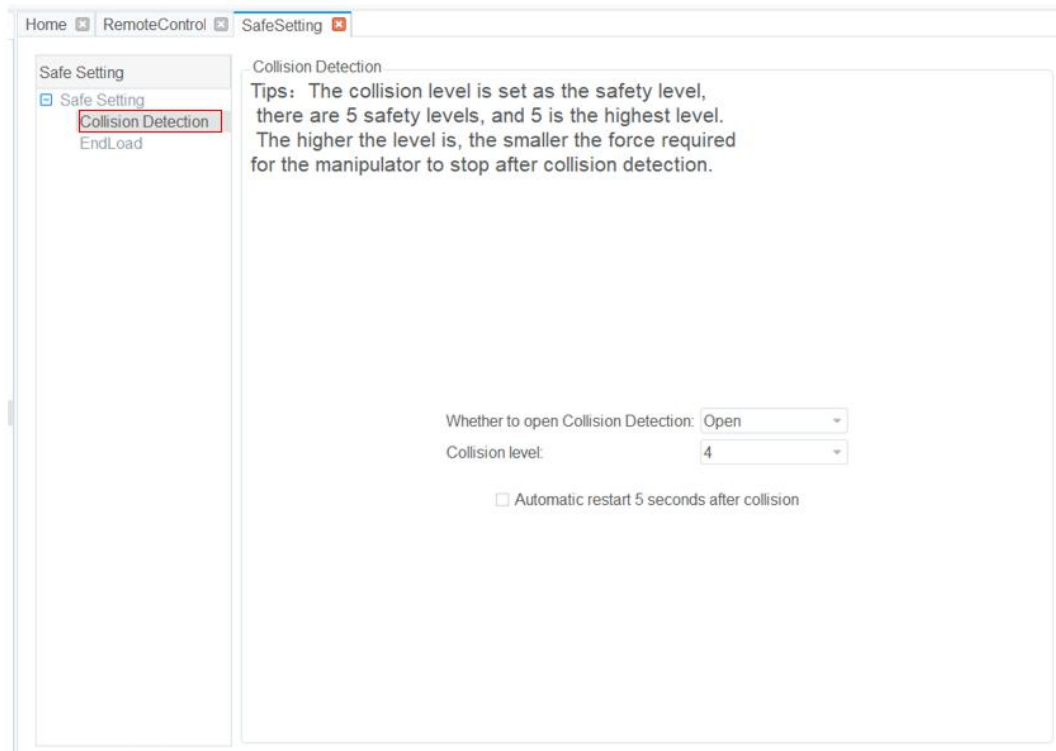


Figure 3.45 Collision detection

3.7.8.2 End load

To ensure optimum robot performance, it is important to make sure the load and inertia of the end effector are within the maximum range for the robot.

The weight of load includes weight of the end effector and work piece. You can set the load in **Parameter > SafeSetting > End load**. Or you can also set it when enabling the robot motor, as shown in Figure 3.46.

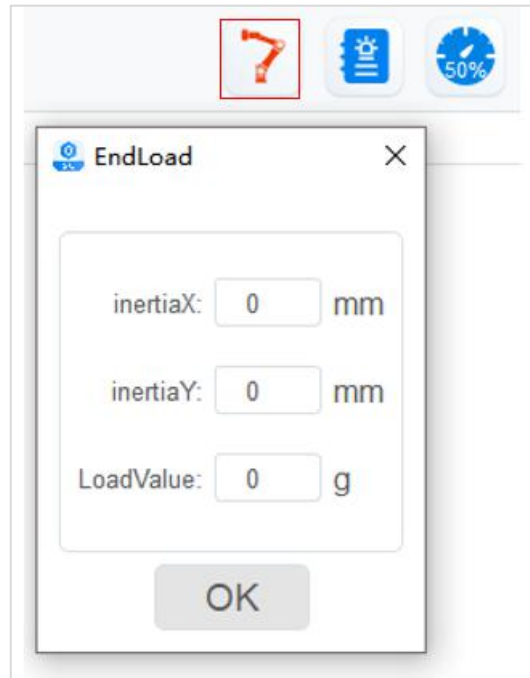


Figure 3.46 End load

3.8 ToolConfig

3.8.1 BasicConfig

You can select languages on the **ToolConfig > BasicConfig > Language** page. Also, you can modify the password on the **ToolConfig > BasicConfig > UserMode** page.

3.8.2 PluginsInfo

User can check the plug information on this page, including author, version, etc. The details will not be described in this topic.

3.8.3 Log

You can understand the historical operation of the robot by viewing the log. The log can be screened according to three types of logs: user operation, control error and servo error. Click **Reset** to clear the log.

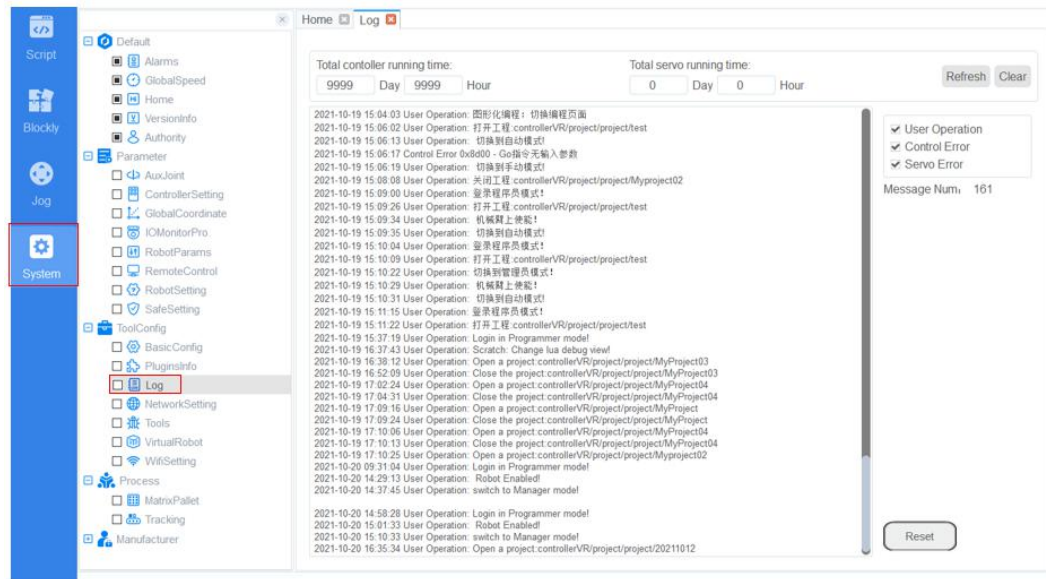


Figure 3.47 Log

3.8.4 Network Service

MG400 can be communicated with external equipment by the **Ethernet2** interface which supports TCP, UDP and Modbus protocols. The default IP address is **192.168.2.6**. In real applications, if the TCP or UDP protocol is used, MG400 can be a client or a server based on site requirements; if the Modbus protocol is used, MG400 only can be the Modbus slave, and the external equipment is the master.


You can modify the IP address on the  > **ToolConfig** > **NetworkSetting** page, as shown in Figure 3.48. The IP address of MG400 must be in the same network segment of the external equipment without conflict.



Figure 3.48 IP address setting

If MG400 connects to the external equipment directly, with a router or with a

switchboard, please select **Manual IP Address** and modify **IP Address**, **subnet mask**, **default gateway**, and then click **Save**.



Please DO NOT insert the network cable into the WAN interface when using a router for the connection.

3.8.5 Tools

DobotSCStudio supports serial port debugging, TCP/UDP debugging and Modbus debugging for user. The details on how to use it will not be described in this topic.

3.8.6 VirtualRobot

When user jogs or runs a robot, the virtual simulation interface can be used to view the robot movement in real time.

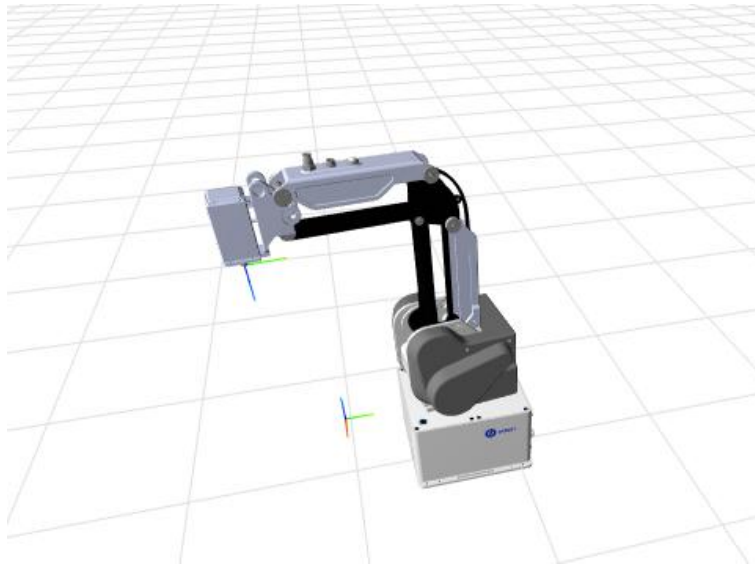



Figure 3.49 Virtual simulation

3.8.7 WiFi Setting

The robot system can be communicated with external equipment by the WiFi module. You can modify the WiFi name and password on the  > **ToolConfig** > **WiFiSetting** page and then restart the controller to make it effective. The default password is **1234567890**.

Wifi

SSID:

Password:

Modify

Figure 3.50 WiFi setting

4. Program Language

SC series controller encapsulates the robot dedicated API commands for programming with Lua language. This section describes commonly used commands for reference.

4.1 Arithmetic Operators

Table 4.1 Arithmetic operator

Command	Description
+	Addition
-	Subtraction
*	Multiplication
/	Floating point division
//	Floor division
%	Remainder
^	Exponentiation
&	And operator
	OR operator
~	XOR operator
<<	Left shift operator
>>	Right shift operator

4.2 Relational Operator

Table 4.2 Relational Operator

Command	Description
==	Equal
~=	Not equal
<=	Equal or less than
>=	Equal or greater than
<	Less than
>	Greater than

4.3 Logical Operators

Table 4.3 Logical operator

Command	Description
or	Logical OR operator
not	Logical NOT operator
and	Logical AND operator

4.4 General Keywords

Table 4.4 General keyword

Command	Description
break	Break out of a loop
local	Define a local variable, which is available in the current script
nil	Null
return	Return a value
enter	Line feed

4.5 General Symbol

Table 4.5 General symbol

Command	Description
#	Get the length of the array table

4.6 Processing Control Commands

Table 4.6 Processing control command

Command	Description
if...then...elseif...then...else...end	Conditional instruction (if)
while...do...end	Loop instruction (while)
for...do...end	Loop instruction (for)
repeat... until()	Loop instruction (repeat)

4.7 Global Variable

The robot global variables can be defined in the **global.lua** file, including global functions, global points, and global variables.

- Global function:

```
function exam()
    print("This is an example")
end
```

- Global point:

Define a Cartesian coordinate point, the User and Tool coordinate systems are both default coordinate systems.

```
P = { armOrientation = "right", coordinate = {10,10,10,0}, tool = 0, user = 0}
```

- Define a joint coordinate point

```
P = {joint = {20,10,22,85}}
```

- Global variable

```
flag = 0
```

4.8 Motion Commands

Table 4.7 Motion commands

Command	Description
MovJ	Point to Point, the target point is Cartesian point
JointMovJ	Point to point, the target point is Joint point
MovL	Linear Movement, the target point is Cartesian point
Jump	Jump Movement. The jump parameters can be set in this command
	Jump Movement. The jump parameters are called by Arch index
RelMovL	Move to the Cartesian offset position in a straight line
RelMovJ	Move to the Cartesian offset position in a point-to-point mode
MovLIO	Linear movement in parallel with output
MovJIO	Point to point movement in parallel with output
Arc	Arc movement
Circle	Circle movement



NOTICE

Optional parameters for each motion command can be set individually

Table 4.8 MovJ command


Function	MovJ(P)
	local Option={CP=1, SpeedJ=50, AccJ=20} MovJ(P, Option)
Description	Point to Point, the target point is Cartesian point
Parameter	<p>Required parameter: P, Indicate target point, which is user-defined or obtained from the points list. Only Cartesian point is supported.</p> <p>Optional parameter: {CP=1, SpeedJ=50, AccJ=20}. You can double-click  to insert the command with optional parameters.</p> <ul style="list-style-type: none"> CP: Continuous path rate. Value range: 0~100 SpeedJ: Velocity rate. Value range: 1~100 AccJ: Acceleration rate. Value range: 1~100

Table 4.9 JointMovJ command


Function	JointMovJ(P)
	local Option={CP=1, SpeedJ=50, AccJ=20} local P={joint={J1,J2,J3,J4}} JointMovJ(P, Option)
Description	Point to point, the target point is Joint point
Parameter	<p>Required parameter: P, Indicate the target point, which is user-defined or obtained from the points list. Only joint point is supported.</p> <p>Optional parameter: {CP=1, SpeedJ=50, AccJ=20}. You can double-click  to insert the command with optional parameters.</p> <ul style="list-style-type: none"> CP: Continuous path rate. Value range: 0~100 SpeedJ: Velocity rate. Value range: 1~100 AccJ: Acceleration rate. Value range: 1~100

Table 4.10 MovL command

Function	MovL(P)
	local Option={CP=1, SpeedL=50, AccL=20} MovL(P, Option)
Description	Linear Movement, the target point is Cartesian point
Parameter	<p>Required parameter: P, Indicate the target point, which is user-defined or obtained from the points list. Only Cartesian point is supported.</p>


	<p>Optional parameter: {CP=1, SpeedL=50, AccL=20}. You can double-click  to insert the command with optional parameters.</p> <ul style="list-style-type: none"> CP: Continuous path rate. Value range: 0~100 SpeedL: Velocity rate. Value range: 1~100 AccL: Acceleration rate. Value range: 1~100
--	--

Table 4.11 Arc command


Function	Arc(P1, P2)
	<p>local Option={CP=1, SpeedL=50, AccL=20}</p> <p>Arc(P1, P2, Option)</p>
Description	Arc movement. This command needs to combine with other motion commands, to obtain the starting point of an arc trajectory
Parameter	<p>Required parameter:</p> <ul style="list-style-type: none"> P1, Middle point, which is user-defined or obtained from the points list. Only Cartesian point is supported. P2, End point, which is user-defined or obtained from the points list. Only Cartesian point is supported. <p>Optional parameter: {CP=1, SpeedL=50, AccL=20}. You can double-click  to insert the command with optional parameters.</p> <ul style="list-style-type: none"> CP: Continuous path rate. Value range: 0~100 SpeedL: Velocity rate. Value range: 1~100 AccL: Acceleration rate. Value range: 1~100

Table 4.12 Jump command

Function	<p>local Option={SpeedL=50, AccL=20, Start=10, ZLimit=100, End=20}</p> <p>Jump(P, Option)</p>
Description	Jump Movement. The jump parameters can be set in this command.
Parameter	<p>Required parameter: P, Indicate the target point, which is user-defined or obtained from the points list. Only Cartesian point is supported.</p> <p>Optional parameter: {SpeedL=50, AccL=20, Start=10, ZLimit=100, End=20}</p> <ul style="list-style-type: none"> SpeedL: Velocity rate. Value range: 1~100 AccL: Acceleration rate. Value range: 1~100 Start: Lifting height(h1). ZLimit: Maximum lifting height(z_limit). The height of the starting point and target point

	cannot exceed ZLimit; otherwise, an error alarm is triggered.
	<ul style="list-style-type: none"> End: Dropping height(h2).

Table 4.13 Jump command

Function	local Option={SpeedL=50, AccL=20, Arch=1} Jump(P, Option)
Description	Jump Movement. The jump parameters are called by Arch index
Parameter	<p>Required parameter: P, Indicate the target point, which is user-defined or obtained from the points list. Only Cartesian point is supported.</p> <p>Optional parameter: {SpeedL=50, AccL=20, Start=10, Arch=1}</p> <ul style="list-style-type: none"> SpeedL: Velocity rate. Value range: 1~100 AccL: Acceleration rate. Value range: 1~100 Arch: Arch index. Value range: 0~9. Please set Jump parameters on the System > Parameters > RobotParams > PlayBackArch page.

Table 4.14 Circle command


Function	Circle(P1, P2, Count)
	<p>local Option={CP=1, SpeedL=50, AccL=20}</p> <p>Circle(P1, P2, Count, Option)</p>
Description	<p>Move from the current position to a target position in a circular interpolated mode under the Cartesian coordinate system</p> <p>This command needs to combine with other motion commands, to obtain the starting point of an arc trajectory</p>
Parameter	<p>Required parameter:</p> <ul style="list-style-type: none"> P1, Middle point, which is user-defined or obtained from the points list. Only Cartesian point is supported. P2, Middle point, which is user-defined or obtained from the points list. Only Cartesian point is supported. Count, Number of circles. <p>Optional parameter: {CP=1, SpeedL=50, AccL=20}. You can double-click  to insert the command with optional parameters.</p> <ul style="list-style-type: none"> CP: Continuous path rate. Value range: 0~100 SpeedL: Velocity rate. Value range: 1~100 AccL: Acceleration rate. Value range: 1~100

Table 4.15 RelMovJ command


Function	local Offset = {OffsetX, OffsetY, OffsetZ, OffsetR} RelMovJ(Offset)
	local Offset = {OffsetX, OffsetY, OffsetZ, OffsetR} local Option={CP=1, SpeedJ=50, AccJ=20} RelMovJ(Offset, Option)
Description	Move to the Cartesian offset position in a point-to-point mode
Parameter	<p>Required parameter: {OffsetX, OffsetY, OffsetZ, OffsetR}, X, Y, Z, R axes offset in the Cartesian coordinate system.</p> <p>Optional parameter: {CP=1, SpeedJ=50, AccJ=20}. You can double-click  to inset the command with optional parameters.</p> <ul style="list-style-type: none"> CP: Continuous path rate. Value range: 0~100 SpeedJ: Velocity rate. Value range: 1~100 AccJ: Acceleration rate. Value range: 1~100

Table 4.16 RelMovL command


Function	local Offset = {OffsetX, OffsetY, OffsetZ, OffsetR} RelMovL(Offset)
	local Offset = {OffsetX, OffsetY, OffsetZ, OffsetR} local Option={CP=1, SpeedL=50, AccL=20} RelMovL(Offset, Option)
Description	Move to the Cartesian offset position in a straight line
Parameter	<p>Required parameter: {OffsetX, OffsetY, OffsetZ, OffsetR}, X, Y, Z, R axes offset in the Cartesian coordinate system.</p> <p>Optional parameter: {CP=1, SpeedL=50, AccL=20}. You can double-click  to inset the command with optional parameters.</p> <ul style="list-style-type: none"> CP: Continuous path rate. Value range: 0~100 SpeedL: Velocity rate. Value range: 1~100 AccL: Acceleration rate. Value range: 1~100

Table 4.17 MovLIO command

Function	local IO={ {Mode, Distance, Index, Status}, {Mode, Distance, Index, Status}, ...} MovLIO(P, IO)
	local IO={ {Mode, Distance, Index, Status}, {Mode, Distance, Index, Status}, ...}



	<p>local Option={CP=1, SpeedL=50, AccL=20}</p> <p>MovLIO(P, IO, Option)</p>
Description	Linear movement in parallel with output . Multiple digital output ports can be set
Parameter	<p>Required parameter:</p> <ul style="list-style-type: none"> P, Indicate the target point, which is user-defined or obtained from the points list. Only Cartesian point is supported. {Mode, Distance, Index, Status}, {Mode, Distance, Index, Status}... Multiple digital output ports can be set. <ul style="list-style-type: none"> Mode: Set Distance mode. 0: Distance is a percentage; 1: Distance from the starting point, or from the target point. Distance: If the Mode is a percentage, it represents the percentage of the distance between the starting point and the target point. If the Mode is a distance, it represents the distance from the starting point, or from the target point. If the distance is set to positive, it indicates the distance from the starting point; if set to negative, it indicates the distance from the target point. Index: Digital output port. Value range: 1~18 Status: Status of the digital output port. Value range: 0 or 1 <p>Optional parameter: {CP=1, SpeedL=50, AccL=20}. You can double-click  to insert the command with optional parameters.</p> <ul style="list-style-type: none"> CP: Continuous path rate. Value range: 0~100 SpeedL: Velocity rate. Value range: 1~100 AccL: Acceleration rate. Value range: 1~100

Table 4.18 MovJIO

Function	<p>local IO={ {Mode, Distance, Index, Status}, {Mode, Distance, Index, Status},...}</p> <p>MovJIO(P, IO)</p>
	<p>local IO={ {Mode, Distance, Index, Status}, {Mode, Distance, Index, Status},...}</p> <p>local Option={CP=1, SpeedJ=50, AccJ=20}</p> <p>MovJIO(P, IO, Option)</p>
Description	Point to point movement in parallel with output. Multiple digital output ports can be set
Parameter	<p>Required parameter:</p> <ul style="list-style-type: none"> P, Indicate the target point, which is user-defined or obtained from the points list. Only Cartesian point is supported. {Mode, Distance, Index, Status}, {Mode, Distance, Index, Status},... Multiple digital output ports can be set. <ul style="list-style-type: none"> Mode: Set Distance mode. 0: Distance is a percentage; 1: Distance from the starting point, or from the target point.

	<ul style="list-style-type: none"> ■ Distance: If the Mode is a percentage, it represents the percentage of the distance between the starting point and the target point. If the Mode is a distance, it represents the distance from the starting point, or from the target point. If the distance is set to positive, it indicates the distance from the starting point; if set to negative, it indicates the distance from the target point. ■ Index: Digital output port. Value range: 1~18 ■ Status: Status of the digital output port. Value range: 0 or 1 <p>Optional parameter: {CP=1, SpeedJ=50, AccJ=20}. You can double-click  to insert the command with optional parameters.</p> <ul style="list-style-type: none"> • CP: Continuous path rate. Value range: 0~100 • SpeedJ: Velocity rate. Value range: 1~100 • AccJ: Acceleration rate. Value range: 1~100
--	---

4.9 Motion Parameter Commands

Table 4.19 Motion parameter commands

Command	Description
AccJ	Set the joint acceleration rate. This command is valid only when the motion mode is MovJ, MovJIO, or JointMovJ
AccL	Set the Cartesian acceleration rate. This command is valid only when the motion mode is MovL, MovLIO, Jump, Arc, Circle
SpeedJ	Set the joint velocity rate. This command is valid only when the motion mode is MovJ, MovJIO, or JointMovJ
SpeedL	Set the Cartesian velocity rate. This command is valid only when the motion mode is MovL, MovLIO, Jump, Arc, Circle
CP	Set the continuous path rate
Sync	Whether to stop at this point
SetPayload	Set payload, X-axis offset, Y-axis offset and servo index

Table 4.20 AccJ command

Function	AccJ(R)
Description	Set the joint acceleration rate. This command is valid only when the motion mode is MovJ, MovJIO, or JointMovJ
Parameter	Required parameter: Acceleration rate. Value range: 1~100

Table 4.21 AccL command

Function	AccL(R)
Description	Set the Cartesian acceleration rate. This command is valid only when the motion mode is MovL, MovLIO, Jump, Arc, Circle
Parameter	Acceleration rate. Value range: 1~100

Table 4.22 SpeedJ command

Function	SpeedJ(R)
Description	Set the joint velocity rate. This command is valid only when the motion mode is MovJ, MovJIO, or JointMovJ
Parameter	Velocity rate. Value range: 1~100

Table 4.23 SpeedL command

Function	SpeedL(R)
Description	Set the Cartesian velocity rate. This command is valid only when the motion mode is MovL, MovLIO, Jump, Arc, Circle
Parameter	Velocity rate. Value range: 1~100

Table 4.24 CP command

Function	CP(R)
Description	Set the continuous path rate. CP means when the robot arm passes through the middle point from the starting point to the end point, whether it transitions through the middle point in a rectangular way or in a curve way. This command is invalid when the motion mode is Jump
Parameter	Continuous path rate. Value range: 0~100

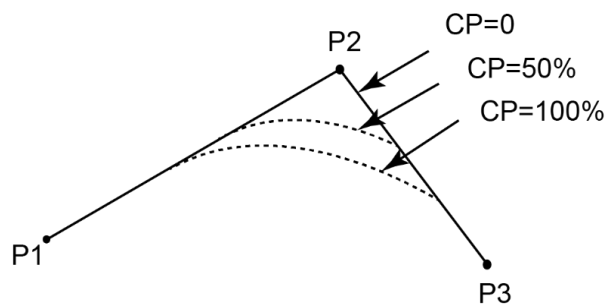


Figure 4.1 Continuous path

Table 4.25 Sync command

Function	Sync()
Description	Whether to stop at this point
Parameter	Null

Table 4.26 SetPayload command

Function	SetPayload(payload, {x, y}, index)
Description	Set payload, X-axis offset, Y-axis offset and servo index
Parameter	<p>Required parameter:</p> <ul style="list-style-type: none"> payload: Payload. Value range: 0~750. Unit: g {x,y}: Offset in X-axis and Y-axis <p>Optional parameter: index, servo parameter index. The default value range is 1~10.</p>

4.10 Input/output Commands

Table 4.27 Input/output command

Command	Description
DI	Get the status of the digital input port
DO	Set the status of the digital output port (Queue command)
DOInstant	Set the status of digital output port (Immediate command)

NOTE

Dobot MG400 supports two kinds of commands: immediate command and queue command:

- Immediate command: MG400 will process the command once the command is received regardless of whether the controller is processing other commands.
- Queue command: When MG400 receives a command, this command will be pressed into the internal command queue. MG400 will execute commands in the order in which the commands were pressed into the queue.

Table 4.28 DI command

Function	DI(index)
Description	Get the status of the digital input port

Parameter	Index: Digital input index. Value range: 1~18
Return	<ul style="list-style-type: none"> When an index is set in the DI function, DI(index) returns the status (ON/OFF) of this specified input port When there is no index in the DI function, DI() returns the status of all the input ports, which are saved in a table <p>For example, local di= DI(), the saving format is {num = 24 value = {0x55, 0xAA, 0x52}}, you can obtain the status of the specified input port with di.num and di.value[n]</p>

Table 4.29 DO command (Queue command)

Function	DO(index, ON OFF)
Description	Set the status of digital output port (Queue command)
Parameter	Index: Digital output port. Value range: 1~18 <ul style="list-style-type: none"> ON/OFF: Status of the digital output port.

Table 4.30 DOInstant command

Function	DOInstant(Index,ON/OFF)
Description	Set the status of digital output port (Immediate command)
Parameter	<ul style="list-style-type: none"> Index: Digital output port. Value range: 1~18 ON/OFF: Status of the digital output port

4.11 Program Managing Commands

Table 4.31 Program managing commands

Command	Description
Wait	Set the delay time for robot motion commands
Sleep	Set the delay time for all commands
Pause	Pause the running program
ResetElapsedTime	Start timing
ElapsedTime	Stop timing
System	Get the current time

Table 4.32 Wait command

Function	Wait(<i>time</i>)
Description	Set the delay time for robot motion commands
Parameter	time: Delay time. Unit: ms

Table 4.33 Sleep command

Function	Sleep(<i>time</i>)
Description	Set the delay time for all commands
Parameter	time: Delay time. Unit: ms

Table 4.34 Pause command

Function	Pause()
Description	Pause the running program. When the program runs to this command, robot pauses running and you need to click Resume on the Software to recover the running.
Parameter	Null

Table 4.35 Start timing command

Function	ResetElapsedTime()
Description	Start timing after all commands before this command are executed completely. Use in conjunction with ElapsedTime() command
Parameter	Null

Table 4.36 Stop timing command

Function	ElapsedTime()
Description	Stop timing and return the time difference. Use in conjunction with ResetElapsedTime() command
Parameter	Null
Return	Time difference. Unit: ms

Table 4.37 Get current time command

Function	Systime()
Description	Get the current time
Parameter	Null

4.12 Pose Getting Command

Table 4.38 Pose commands

Command	Description
GetPose	Get Cartesian coordinates
GetAngle	Get Joint coordinates
RelPoint	Cartesian point offset
RelJoint	Joint point offset
Cartesian Point	Define a Cartesian point
Joint Point	Define a joint point

Table 4.39 Pose command (1)

Function	GetPose()
Description	Get the current pose of the robot under the Cartesian coordinate system If you have set the User or Tool coordinate system, the current pose is under the current User or Tool coordinate system
Parameter	Null
Return	Cartesian coordinate of the current pose

Table 4.40 Pose command (2)

Function	GetAngle()
Description	Get the current pose of the robot under the Joint coordinate system
Parameter	Null
Return	Joint coordinate of the current pose

Table 4.41 RelPoint command

Function	local Offset={OffsetX, OffsetY, OffsetZ, OffsetR} RelPoint(P, Offset)
Description	Set the X, Y, Z, R axes offset under the Cartesian coordinate system to return a new Cartesian coordinate point. The robot can move to this point in all motion commands except JointMovJ
Parameter	<ul style="list-style-type: none"> P, Indicate the current Cartesian point, which is user-defined or obtained from the points list. Only Cartesian point is supported. {OffsetX, OffsetY, OffsetZ, OffsetR}: X, Y, Z, R axes offset in the Cartesian coordinate system.
Return	Cartesian point

Table 4.42 RelJoint command

Function	local Offset={Offset1, Offset2, Offset3, Offset4} RelJoint(P, Offset)
Description	Set the joint offset in the Joint coordinate system to return a new joint point. The robot can move to this point only in JointMovJ command
Parameter	P, Indicate the current joint point, which is user-defined or obtained from the points list. Only joint point is supported. {Offset1, Offset2, Offset3, Offset4}: J1 - J4 axes offset.
Return	Joint point

Table 4.43 local P command

Function	local P={coordinate = {x,y,z,r}, tool = 0, user = 0}
Description	Define a Cartesian point
Parameter	{x,y,z,r}: X, Y, Z, R axes coordinates. tool: Tool coordinate system index. Value range: 0~9 user: User coordinate system index. Value range: 0~9

Table 4.44 local P command

Function	local P={joint= {j1,j2,j3,j4}}
Description	Define a joint point
Parameter	{j1,j2,j3,j4}, J1-J4 axes coordinates

4.13 TCP

Table 4.45 TCP commands

Command	Description
TCPCreate	Create a TCP network
TCPStart	Establish TCP connection
TCPRead	Receive data from a client
TCPWrite	Sends data to a client
TCPDestroy	Release a TCP network

Table 4.46 Create TCP command

Function	Err, Socket = TCPCreate(IsServer, IP, Port)
Description	Create a TCP network Only support a single connection
Parameter	<ul style="list-style-type: none"> IsServer: Whether to create a server. false: Create a client; true: Create a server IP: IP address of the server, which is in the same network segment of the client without conflict Port: Server port. When the robot is set as a server, port cannot be set to 502 and 8080. Otherwise, it will be in conflict with the Modbus default port or the port used in the conveyor tracking application, causing the creation to fail
Return	Err: <ul style="list-style-type: none"> 0: TCP network is created successfully 1: TCP network fails to be created Socket: Socket object

Table 4.47 TCP connection command

Function	TCPStart(Socket, Timeout)
Description	Establish TCP connection
Parameter	Required parameter: <ul style="list-style-type: none"> Socket: Socket object. Timeout: Wait timeout. Unit: s. If Timeout is 0, the connection is still waiting. If not, after exceeding the timeout, the connection is exited.

Return	<ul style="list-style-type: none"> 0: TCP connection is successful 1: Input parameters are incorrect 2: Socket object is not found 3: Timeout setting is incorrect 4: If the robot is set as a client, it indicates that the connection is wrong. If the robot is set as a server, it indicates that receiving data is wrong
--------	---

Table 4.48 Receive data command

Function	Err, RecBuf = TCPRead(Socket, Timeout, Type)
Description	Robot as a client receives data from a server or as a server receives data from a client
Parameter	<ul style="list-style-type: none"> Socket: socket object Timeout: Receiving timeout. Unit: s. If timeout is 0 or is not set, this command is a block reading. Namely, the program will not continue to run until receiving data is complete. If not, after exceeding the timeout, the program will continue to run regardless of whether receiving data is complete Type: Buffer type. If type is not set, the buffer format of RecBuf is a table. If type is set to string, the buffer format is a string
Return	Err: <ul style="list-style-type: none"> 0: Receiving data is successful 1: Data fails to be received Recbuf: Data buffer

Table 4.49 Send data command

Function	TCPWrite(Socket, Buf, Timeout)
Description	Robot as a client sends data to a server or as a server sends data to a client
Parameter	<ul style="list-style-type: none"> Socket: Socket object. Buf: Data sent by the robot. Timeout: Timeout. Unit: s. If Timeout is 0 or not set, this command is a block reading. Namely, the program will not continue to run until sending data is complete. If not, after exceeding the timeout, the program will continue to run regardless of whether sending data is complete.
Return	0: Sending data is successful 1: Data fails to be sent

Table 4.50 Release TCP network command

Function	TCPDestroy(Socket)
Description	Release a TCP network
Parameter	Socket: Socket object
Return	0: Releasing TCP is successful 1: TCP fails to be released

4.14 UDP

Table 4.51 UDP commands

Command	Description
TCPCreate	Create a UDP network
UDPRead	Receive data from a client
UDPWrite	Send data to a client

Table 4.52 Create UDP network command

Function	Err, Socket = UDPCreate(IsServer, IP, Port)
Description	Create a UDP network Only a single connection is supported
Parameter	<ul style="list-style-type: none">IsServer: Whether to create a server. false: Create a client; true: Create a server.IP: IP address of the server, which is in the same network segment of the client without conflict.Port: Server port. When the robot is set as a server, port cannot be set to 502 and 8080. Otherwise, it will be in conflict with the Modbus default port or the port used in the conveyor tracking application, causing the creation to fail.
Return	Err: <ul style="list-style-type: none">0: The UDP network is created successfully1: The UDP network fails to be created Socket: Socket object

Table 4.53 Receive data command

Function	Err, RecBuf = UDPRead(Socket, Timeout, Type)
----------	--

Description	Robot as a client receives data from a server or as a server receives data from a client
Parameter	<ul style="list-style-type: none"> Socket: Socket object. Timeout: Receiving timeout. Unit: s. If Timeout is 0 or is not set, this command is a block reading. Namely, the program will not continue to run until receiving data is complete. If not, after exceeding the timeout, the program will continue to run regardless of whether receiving data is complete. Type: Buffer type. If Type is not set, the buffer format of RecBuf is a table. If Type is set to string, the buffer format is a string
Return	Err: <ul style="list-style-type: none"> 0: Receiving data is successful 1: Data fails to be received Recbuf: Data buffer

Table 4.54 Send data command

Function	UDPWrite(Socket, Buf, Timeout)
Description	Robot as a client sends data to a server or as a server sends data to a client
Parameter	Socket: Socket object Buf: Data sent by the robot Timeout: Timeout. Unit: s. If timeout is 0 or not set, this command is a block reading. Namely, the program will not continue to run until sending data is complete. If not, after exceeding the timeout, the program will continue to run regardless of whether sending data is complete
Return	0: Sending data is successful 1: Data fails to be sent

4.15 Modbus

4.15.1 Description on Modbus Register

Modbus protocol is a serial communication protocol. MG400 can communicate with external equipment by this protocol. Here, External equipment such as a PLC is set as the Modbus master, and MG400 is set as the slave.

Modbus data is most often read and written as registers. Based on our robot memory space, we also define four types of registers: coil, discrete input, input, and holding registers for data interaction between the external equipment and MG400. Each register has 4096 addresses. For details, please see as follows.

- Coil register

Table 4.55 Coil register description

Coil register address (e.g.: PLC)	Coil register address (MG400)	Data type	Description
00001	0	Bit	Start
00002	1	Bit	Pause
00003	2	Bit	Continue
00004	3	Bit	Stop
00005	4	Bit	Emergency stop
00006	5	Bit	Clear alarm
00007~0999	6~998	Bit	Reserved
01001~04096	999~4095	Bit	User-defined

- Discrete input register

Table 4.56 Description on discrete input register

Discrete input register address (e.g. PLC)	Discrete input register address (MG400)	Data type	Description
10001	0	Bit	Automated exit
10002	1	Bit	Ready state
10003	2	Bit	Paused state
10004	3	Bit	Running state
10005	4	Bit	Alarm state
10006~10999	5~998	Bit	Reserved
11000~14096	999~4095	Bit	User-defined

- Input register

Table 4.57 Description on input register

Input register address (e.g. PLC)	Input register address (MG400)	Data type	Description
30001~34096	0-4095	Byte	Reserved

- Holding register

Table 4.58 Description on holding register

Holding register address (e.g.: PLC)	Holding register address (MG400)	Data type	Description
40001~41000	0~999	Byte	Reserved
41001~44096	1000~4095	Byte	User-defined

4.15.2 Command Description

Table 4.59 Modbus commands

Command	Description
GetCoils	Read the value from Modbus slave coil register address
SetCoils	Set the coil register in the Modbus slave
GetInBits	Read the value from the Modbus slave discrete register address
GetInRegs	Read the input register value with the specified data type from the Modbus slave
GetHoldRegs	Read the holding register value from the Modbus slave
SetHoldRegs	Set the holding register in the Modbus slave

Table 4.60 Read coil register command

Function	GetCoils(Addr, Count)
Description	Read the value from Modbus slave coil register address
Parameter	Addr: Starting address of the coils. Value range: 0-4095 Count: Number of the coils to read. Value range: 0 to 4096- Addr
Return	Return a table, each with the value 1 or 0, where the first value in the table corresponds to the coil value at the starting address

Table 4.61 Set coil register command

Function	SetCoils(Addr, Count, Table)
Description	Set the coil register in the Modbus slave This command is not supported when the coil register address is from 0 to 5
Parameter	Addr: Starting address of the coils to set. Value range: 6 - 4095 Count: Number of the coils to set. Value range: 0 to 4096- Addr Table: The values written into the coil register: Data type: bit

Table 4.62 Read discrete input register command

Function	GetInBits(Addr, Count)
Description	Read the value from the Modbus slave discrete register address
Parameter	Addr: Starting address of the discrete inputs to read. Value range: 0-4095 Count: Number of the discrete inputs to read. Value range: 0 to 4096- Addr
Return	Return a table, each with the value 1 or 0, where the first value in the table corresponds to the discrete value at the starting address. Data type: bit

Table 4.63 Read input register command

Function	GetInRegs(Addr, Count, Type)
Description	Read the input register value with the specified data type from the Modbus slave
Parameter	Addr: Starting address of the input registers. Value range: 0 - 4095 Count: Number of the input registers to read. Value range: 0 ~ 4096-addr Type: Data type <ul style="list-style-type: none">• Empty: Read 16-bit unsigned integer (two bytes, occupy one register)• “U16”: Read 16-bit unsigned integer (two bytes, occupy one register)• “U32”: Read 32-bit unsigned integer (four bytes, occupy two registers)• “F32”: Read 32-bit single-precision floating-point number (four bytes, occupy two registers)• “F64”: Read 64-bit double-precision floating-point number (eight bytes, occupy four registers)
Return	Return a table, the first value in the table corresponds to the input register value at the starting address

Table 4.64 Read holding register command

Function	GetHoldRegs(Addr, Count, Type)
Description	Read the holding register value from the Modbus slave according to the specified data type

Parameter	<p>Addr: Starting address of the holding registers. Value range: 0 - 4095</p> <p>Count: Number of the holding registers to read. Value range: 0 to 4096-addr</p> <p>Type: Datatype</p> <ul style="list-style-type: none"> • Empty: Read 16-bit unsigned integer (two bytes, occupy one register) • “U16”: Read 16-bit unsigned integer (two bytes, occupy one register) • “U32”: Read 32-bit unsigned integer (four bytes, occupy two registers) • “F32”: Read 32-bit single-precision floating-point number (four bytes, occupy two registers) • “F64”: Read 64-bit double-precision floating-point number (eight bytes, occupy four registers)
Return	Return a table, the first value in the table corresponds to the input register value at the starting address

Table 4.65 Set holding register command

Function	SetHoldRegs(Addr, Count, Table, Type)
Description	Set the holding register in the Modbus slave
Parameter	<p>Addr: Starting address of the holding registers to set. Value range: 0 - 4095</p> <p>Count: Number of the holding registers to set. Value range: 0 to 4096-addr</p> <p>Table: Holding register value, stored in a table</p> <p>Type: Datatype</p> <ul style="list-style-type: none"> • Empty: Read 16-bit unsigned integer (two bytes, occupy one register) • “U16”: Set 16-bit unsigned integer (two bytes, occupy one register) • “U32”: Set 32-bit unsigned integer (four bytes, occupy two registers) • “F32”: Set 32-bit single-precision floating-point number (four bytes, occupy two registers) • “F64”: Set 64-bit double-precision floating-point number (eight bytes, occupy four registers)

4.16 Conveyor Tracking

Table 4.66 Conveyor commands

Command	Description
CnvVison	Set conveyor number to create a tracing queue
GetCnvObject	Obtain status of the object
SetCnvPointOffset	Set X,Y axes offset under the set User coordinate system

Command	Description
SetCnvTimeCompensation	Set time compensation
SyncCnv	Synchronize the specified conveyor
StopSyncCnv	Stop synchronizing the conveyor

Table 4.67 CnvVison command

Function	CnvVison(CnvID)
Description	Set conveyor number to create a tracing queue
Parameter	CnvID, Conveyor number. Only support single conveyor
Return	0: No error 1: Error

Table 4.68 GetCnvObject command

Function	GetCnvObject(CnvID, ObjID)
Description	Obtain the information of the part on the conveyor to check whether the part is in the pickup area
Parameter	CnvID: Conveyor index. ObjID: Part index.
Return	Part status: Whether there is a part. Value range: true or false Part type Part coordinate (x,y,r)

Table 4.69 SetCnvPointOffset command

Function	SetCnvPointOffset(OffsetX,OffsetY)
Description	Set X,Y axes offset under the set User coordinate system
Parameter	OffsetX: X-axis offset. OffsetY: Y-axis offset.
Return	0: No error 1: Error

Table 4.70 SetCnvTimeCompensation command

Function	SetCnvTimeCompensation(Time)
Description	Set time compensation. This command is used for compensating the pick-up position offset in the moving direction of the conveyor which is caused by taking photos with a time delay
Parameter	Time, time-offset. Unit: ms
Return	0: No error 1: Error

Table 4.71 SyncCnv command

Function	SyncCnv(CnvID)
Description	Synchronize the specified conveyor. The motion commands used between SyncCnv(<i>CnvID</i>) and StopSyncCnv(<i>CnvID</i>) only support MovL command
Parameter	CnvID, Conveyor index
Return	0: No error 1: Error

Table 4.72 StopSyncCnv command

Function	StopSyncCnv(CnvID)
Description	Stop synchronizing the conveyor. The other commands following this command will not be executed until this command running is completed
Parameter	CnvID, Conveyor index
Return	0: No error 1: Error

4.17 Pallet

Table 4.73 Pallet commands

Command	Description
MatrixPallet	Instantiate matrix pallet
TeachPallet	Instantiate teaching pallet
SetPartIndex	Set the next stack index which is to be operated
GetPartIndex	Get the current operated stack index
SetLayerIndex	Set the next pallet layer index which is to be operated

Command	Description
GetLayerIndex	Get the current pallet layer index
Reset(Pallet)	Reset pallet
IsDone(Pallet)	Check whether the stack assembly or dismantling is complete
Release(Pallet)	Release palletizing instance
PalletMoveIn	The robot moves from the current position to the first stack position as the configured stack assembly path
PalletMoveOut	The robot moves from the current position to the transition point as the configured stack dismantling path

Table 4.74 MatrixPallet command


Function	Pallet = MatrixPallet (Index)
	local Option={IsUnstack= true, User= 1}
	Pallet = MatrixPallet (Index,ID, Option)
Description	Instantiate matrix pallet
Parameter	<p>Required parameter:</p> <ul style="list-style-type: none"> Index: Matrix pallet index. <p>Optional parameter: {IsUnstack= true, User= 1}. You can double-click  to insert the command with optional parameters.</p> <ul style="list-style-type: none"> IsUnstack: Stack mode. Value range: true or false. true: Dismantling mode . false: Assembly mode. If not set, the default is assembly mode User: User coordinate system index. If not set, the default is User 0 coordinate system.
Return	Matrix pallet object

Table 4.75 TeachPallet command

Function	Pallet = TeachPallet (Index, Option)
	local Option={IsUnstack= true, User= 1}
	Pallet = TeachPallet (Index,ID, Option)
Description	Instantiate teaching pallet


Parameter	<p>Required parameter:</p> <ul style="list-style-type: none"> Index: Teaching pallet index. <p>Optional parameter: {IsUnstack= true, User= 1}. You can double-click  to insert the command with optional parameters.</p> <ul style="list-style-type: none"> IsUnstack: Stack mode. Value range: true or false. true: Dismantling mode . false: Assembly mode. If not set, the default is assembly mode User: User coordinate system index. If not set, the default is User 0 coordinate system
Return	Teaching pallet object

Table 4.76 SetPartIndex command

Function	SetPartIndex(Pallet, Index)
Description	Set the next stack index which is to be operated
Parameter	<p>Pallet: Pallet object.</p> <p>Index: The next stack index. Initial value: 0</p>

Table 4.77 GetPartIndex command

Function	GetPartIndex(Pallet)
Description	Get the current operated stack index
Parameter	Pallet, Pallet object
Return	The current operated stack index

Table 4.78 SetLayerIndex command

Function	SetLayerIndex(Pallet, Index)
Description	Set the next pallet layer index which is to be operated
Parameter	<p>Pallet: Pallet object.</p> <p>Index: The next pallet layer index. Initial value: 0</p>

Table 4.79 GetLayerIndex command

Function	GetLayerIndex(Pallet)
Description	Get the current pallet layer index
Parameter	Pallet, Pallet object

Return	The current pallet layer index
--------	--------------------------------

Table 4.80 Reset command

Function	Reset(Pallet)
Description	Reset pallet
Parameter	Pallet, Pallet object

Table 4.81 IsDone command

Function	IsDone(Pallet)
Description	Check whether the stack assembly or dismantling is complete
Parameter	Pallet, Pallet object
Return	true: Finished. false: Un-finished.

Table 4.82 Release command

Function	Release(Pallet)
Description	Release palletizing instance
Parameter	Pallet, Pallet object

Table 4.83 PalletMoveIn command

Function	PalletMoveIn(Pallet)
	local Option={SpeedAB=20, SpeedBC=30, AccAB=20, AccBC=10, CP=20} PalletMoveIn(Pallet, Option)
Description	The robot moves from the current position to the first stack position as the configured stack assembly path



Parameter	<p>Required parameter: Pallet, Pallet object.</p> <p>Optional parameter: {SpeedAB=20, SpeedBC=30, AccAB=20, AccBC=10, CP=20}. You can double-click  to insert the command with optional parameters.</p> <ul style="list-style-type: none"> SpeedAB: Velocity rate when the robot moves from the transition point to the preparation point. Value range: 1~100 SpeedBC: Velocity rate when the robot moves from the preparation point to the first stack point. Value range: 1~100 AccAB: Acceleration rate when the robot moves from the transition point to the preparation point. Value range: 1~100 AccBC: Acceleration rate when the robot moves from the preparation point to the first stack point. Value range: 1~100 CP: Continuous path rate. Value range: 0~100
-----------	---

Table 4.84 PalletMoveOut command

Function	PalletMoveOut(Pallet) local Option={SpeedAB=20, SpeedBC=30, AccAB=20, AccBC=10, CP=20} PalletMoveOut(Pallet, Option)
Description	The robot moves from the current position to the transition point as the configured stack dismantling path
Parameter	<p>Required parameter: Pallet, Pallet object.</p> <p>Optional parameter: {SpeedAB=20, SpeedBC=30, AccAB=20, AccBC=10, CP=20}. You can double-click  to insert the command with optional parameters.</p> <ul style="list-style-type: none"> SpeedAB: Velocity rate when the robot moves from the preparation point to the transition point. Value range: 1~100 SpeedBC: Velocity rate when the robot moves from the first stack point to the preparation point. Value range: 1~100 AccAB: Acceleration rate when the robot moves from the preparation point to the transition point. Value range: 1~100 AccBC: Acceleration rate when the robot moves from the first stack point to the preparation point. Value range: 1~100 CP: Continuous path rate. Value range: 0~100

Appendix A Servo Alarm Description

ID	Level	Description	Solution
25376	0	Abnormalities in internal servo parameters	System error, please contact technical support engineer
21120	0	Programmable logic configuration faults	System error, please contact technical support engineer
29953	5	FPGA software version too low	Please contact technical support engineer
29954	5	Programmable logic interrupt fault	If connecting the power for many times, the alarm is still reported, please replace the drive
25377	5	Internal program exceptions	System error, please contact technical support engineer
21808	0	Parameter storage failure	Reset the parameter and power on again, or please contact technical support engineer
28962	0	Product matching faults	1. Check whether the motor parameter matches the motor model in nameplate; 2. Check whether the motor and driver match, otherwise, select the right motor and driver
21574	0	Invalid servo ON command fault	System error, please contact technical support engineer
28964	0	Absolute position mode product matching fault	System error, please contact technical support engineer
25378	0	Repeated assignment of DI functions	1. Check whether the same function is assigned to different DI's 2. Confirm whether the corresponding MCU supports the assigned functionality
25379	0	DO function allocation overrun	Check whether the motor and circuit are working properly, or contact technical support engineer
29488	0	Data in the motor encoder ROM is incorrectly checked or parameters are not stored	System error, please contact technical support engineer
8752	0	Hardware overcurrent	System error, please contact technical support engineer
8977	0	DQ axis current overflow fault	System error, please contact technical support engineer

ID	Level	Description	Solution
65288	0	FPGA system sampling operation timeout	System error, please contact technical support engineer
9024	0	Output shorted to ground	Please contact technical support engineer
13184	0	UVW phase sequence error	System error, please contact technical support engineer
33922	0	Flying Cars	Please contact technical support engineer
12816	0	Electrical over-voltage in the main circuit	System error, please contact technical support engineer
12832	0	Main circuit voltage undervoltage	System error, please contact technical support engineer
12592	0	Main circuit electrical shortage	Check the cable connection of power, otherwise, replace the driver
12576	0	Control of electrical undervoltage	System error, please contact technical support engineer
33920	0	Overspeed	System error, please contact technical support engineer
65296	0	Pulse output overspeed	System error, please contact technical support engineer
65282	0	Failure to identify angles	System error, please contact technical support engineer
9040	0	Drive overload	Replace the driver
29056	0	Motor overload	System error, please contact technical support engineer
28961	0	Overheating protection for blocked motors	Check whether the hardware is working properly, or contact technical support engineer
17168	0	Radiator overheating	Drop the environment temperature, or contact technical support engineer
29571	0	Encoder battery failure	Connect battery, or contact technical support engineer
29490	0	Encoder multi-turn count error	Replace the motor
29491	0	Encoder multi-turn count overflow	System error, please contact technical support engineer
29492	0	Encoder interference	System error, please contact technical support

ID	Level	Description	Solution
			engineer
29493	0	External encoder scale failure	System error, please contact technical support engineer
29494	0	Encoder data abnormalities	System error, please contact technical support engineer
29495	0	Encoder return checksum exception	System error, please contact technical support engineer
29496	0	Loss of encoder Z signal	System error, please contact technical support engineer
34321	0	Excessive position deviation	Check whether the motor is working properly, or contact technical support engineer
34322	0	Position command too large	System error, please contact technical support engineer
34323	0	Excessive deviation from fully closed-loop position	System error, please contact technical support engineer
25380	0	Electronic gear setting overrun	System error, please contact technical support engineer
25381	0	Wrong parameter setting for fully closed loop function	System error, please contact technical support engineer
25382	0	Software position upper and lower limits set incorrectly	System error, please contact technical support engineer
25383	0	Wrong home position offset setting	System error, please contact technical support engineer
30083	0	Loss of synchronization	System error, please contact technical support engineer
30081	0	Unburned XML configuration file	Burn the XML configuration file
65298	0	Network initialization failure	System error, please contact technical support engineer
30082	0	Sync cycle configuration error	System error, please contact technical support engineer
30084	0	Excessive synchronisation period error	System error, please contact technical support engineer
25384	0	Fault in crossover pulse output setting	System error, please contact technical support engineer

ID	Level	Description	Solution
65521	0	Zero return timeout fault	System error, please contact technical support engineer
29570	0	Encoder battery warning	Replace battery
21570	0	DI emergency brake	System error, please contact technical support engineer
12851	0	Motor overload warning	System error, please contact technical support engineer
12817	0	Brake resistor overload alarm	System error, please contact technical support engineer
25385	0	External braking resistor too small	System error, please contact technical support engineer
13105	0	Motor power cable disconnection	System error, please contact technical support engineer
25386	0	Change of parameters requires re-powering to take effect	Clear the alarm and power on again
30208	0	Frequent parameter storage	Check whether the upper computer is working normal, or contact technical support engineer
21571	0	Forward overtravel warning	System error, please contact technical support engineer
21572	0	Reverse overtravel warning	System error, please contact technical support engineer
29569	0	Internal failure of the encoder	System error, please contact technical support engineer
12597	0	Input phase failure warning	System error, please contact technical support engineer
65432	0	Zero return mode setting error	System error, please contact technical support engineer
65344	0	Parameter recognition failure	System error, please contact technical support engineer
21121	0	internal error	System error, please contact technical support engineer
29956	0	FPGA configuration error	System error, please contact technical support

ID	Level	Description	Solution
			engineer
51020	0	Driver board identification error	System error, please contact technical support engineer
29568	0	Encoder connection error	Check the cable connection of encoder, or contact technical support engineer
8992	0	Software overcurrent	System error, please contact technical support engineer
9088	0	Current zero point too large	System error, please contact technical support engineer
30080	0	EtherCAT communication failure	System error, please contact technical support engineer
33921	0	Excessive speed tracking error	System error, please contact technical support engineer
21120	0	STO Warning	System error, please contact technical support engineer
21569	0	Upper and lower board connection failure	System error, please contact technical support engineer
8980	0	Busbar overcurrent	System error, please contact technical support engineer
17169	0	Damaged or uninstalled temperature measuring resistors	System error, please contact technical support engineer
29572	0	Encoder Eeprom reading CRC fault	System error, please contact technical support engineer
12928	0	Servo and motor power matching faults	System error, please contact technical support engineer

Appendix B Controller Alarm Description

ID	Level	Description	Solution
17	5	Inverse kinematics error with no solution	Reselect movement points
18	5	Inverse kinematics error with result out of working area	Reselect movement points
19	5	Duplicated data in JUMP or ARC or Circles instruction	Reselect movement points
20	5	Wrong input parameters for arc	Enter the correct parameters
21	5	The Start and the End is negative or the zLimit is below the start and end points	Enter the correct parameters
22	5	Wrong arm orientation switch	Reselect movement points
23	5	Plan point during linear motion out of working area	Reselect movement points
24	5	Plan point during circular arc motion out of working area	Reselect movement points
25	5	Wrong mode for motion instruction	Internal software error, restart or contact manufacturer
26	5	Wrong input parameters for speed	Input correct parameter
27	5	Wrong trajectory motion plan of continuous path	Input correct parameter
28	0	Wrong input parameters for circle	Input correct parameter
29	5	Plan point during circular circle motion out of working circle	Reselect movement points
30	5	Inching target position inaccessible	Reverse inch out of limit
32	5	Inverse kinematics singularity during moving	Reselect movement points
33	5	Inverse kinematics with no solution during moving	Reselect movement points
34	5	Inverse kinematics with result out of working area	Reselect movement points
48	5	Joint1 overspeed	Reset the speed or re-select the movement point away from the singularity
49	5	Joint2 overspeed	Reset the speed or re-select the movement point away from the singularity

ID	Level	Description	Solution
50	5	Joint3 overspeed	Reset the speed or re-select the movement point away from the singularity
51	5	Joint4 overspeed	Reset the speed or re-select the movement point away from the singularity
52	0	Joint1 position out of range	Internal error, restart or contact manufacturer
53	0	Joint2 position lag error	Internal error, restart or contact manufacturer
54	0	Joint3 position lag error	Internal error, restart or contact manufacturer
55	0	Joint4 position lag error	Internal error, restart or contact manufacturer
64	5	Joint1 exceeds positive limit	Reverse jog out of limit
65	5	Joint1 exceeds negative limit	Reverse jog out of limit
66	5	Joint2 exceeds positive limit	Reverse jog out of limit
67	5	Joint2 exceeds negative limit	Reverse jog out of limit
68	5	Joint3 exceeds positive limit	Reverse jog out of limit
69	5	Joint3 exceeds negative limit	Reverse jog out of limit
70	5	Joint4 exceeds positive limit	Reverse jog out of limit
71	5	Joint4 exceeds negative limit	Reverse jog out of limit
72	5	Parallelogram positive limit	Reverse jog out of limit
73	5	Parallelogram negative limit	Reverse jog out of limit
80	0	Joint1 lose step	Internal error, restart or contact manufacturer
81	0	Joint2 lose step	Internal error, restart or contact manufacturer
82	0	Joint3 lose step	Internal error, restart or contact manufacturer
83	0	Joint4 lose step	Internal error, restart or contact manufacturer
84	0	Algorithm timeout	Internal error, restart or contact manufacturer
85	0	Emergency button pressed	Release the emergency stop button
96	0	Joint1 drive alarm	Check if the communication of joint 1 is normal and then clear the error
97	0	Joint1 Servo power off	Re-enable joint 1
98	0	Joint2 drive alarm	Check if the communication of joint 2 is normal and then clear the error
99	0	Joint2 Servo power off	Re-enable joint 2
100	0	Joint3 drive alarm	Re-enable joint 3

ID	Level	Description	Solution
101	0	Joint3 Servo power off	Re-enable joint 3
102	0	Joint4 drive alarm	Re-enable joint 4
103	0	Joint4 drive power off	Re-enable joint 4
104	0	Robot homing failed	Home again
105	0	Robot Servo on failed	Check whether the hardware is normal and re-enable
106	0	Abnormal conveyor data	Please contact technical support engineer
107	0	Abnormal conveyor synchronization	Please contact technical support engineer
108	0	Conveyor conveyor encoder 1 is disconnected	Please contact technical support engineer
109	0	Conveyor conveyor encoder 2 is disconnected	Please contact technical support engineer
110	0	Encoder position error	Internal error, restart or contact manufacturer
112	0	Collision Detection	Keep away from the work area and continue to run
161	0	Error switching drag and drop mode	Internal error, restart or contact manufacturer
4096	5	Failed to open mechanical file	Check if the file location is correct and restart
8192	5	Failed to open project file	Check if the file location is correct and restart
8193	5	Failed to open program file	Check if the file location is correct and restart
8194	5	Failed to open global variable file	Check if the file location is correct and restart
8195	5	Failed to open teaching point file	Check if the file location is correct and restart
8196	5	Failed to start debugger process	Rerun debugger process
12288	5	Emergency stop detected	Power on again
12289	5	External emergency stop detected	Power on again
12290	0	The servo power board temperature is too high	Turn off the machine and let it cool for a period of time
33024	5	No input parameters for CP instruction	Enter the correct parameters
33025	5	Input parameters of CP instruction out of range	Enter the correct parameters
33280	5	No input parameters for Arch instruction	Please enter parameters
33281	5	Index parameter of Arch instruction out of range	Enter the correct parameters

ID	Level	Description	Solution
33282	5	Index parameter of Arch instruction not configured yet	Please set index parameters
33536	5	No input parameters for LimZ instruction	Please enter parameters
33537	5	Input parameters of LimZ instruction out of range	Enter the correct parameters
33792	5	No input parameters for Speed instruction	Please enter parameters
33793	5	Ratio parameter of Speed instruction out of range [1, 100]	Enter the correct parameters
34048	5	No input parameters for Accel instruction	Please enter parameters
34049	5	Ratio parameter of Accel instruction out of range [1, 100]	Enter the correct parameters
34304	5	No input parameters for Jerk instruction	Please enter parameters
34305	5	Ratio parameter of Jerk instruction out of range [1, 100]	Enter the correct parameters
34560	5	No input parameters for SpeedS instruction	Please enter parameters
34561	5	Ratio parameter of SpeedS instruction out of range [1, 100]	Enter the correct parameters
34816	5	No input parameters for SpeedR instruction	Please enter parameters
34817	5	Ratio parameter of SpeedR instruction out of range [1, 100]	Enter the correct parameters
35072	5	No input parameters for AccelS instruction	Please enter parameters
35073	5	Ratio parameter of AccelS instruction out of range [1, 100]	Please enter parameters
35328	5	No input parameters for AccelR instruction	Enter the correct parameters
35329	5	Ratio parameter of AccelR instruction out of range [1, 100]	Enter the correct parameters
35584	5	No input parameters for JerkS instruction	Please enter parameters
35585	5	Ratio parameter of JerkS instruction out of range [1, 100]	Enter the correct parameters
35840	5	No input parameters for JerkR instruction	Please enter parameters
35841	5	Ratio parameter of JerkR instruction out of range [1, 100]	Enter the correct parameters
36096	5	No input parameters for Go instruction	Please enter parameters

ID	Level	Description	Solution
36097	5	No motion point parameter for Go instruction	Please enter parameters
36098	5	Incorrect motion point for Go instruction	Enter the correct parameters
36099	5	Incorrect control parameter for Go instruction	Enter the correct parameters
36352	5	No input parameters for Move instruction	Please enter parameters
36353	5	No motion point parameter for Move instruction	Please enter parameters
36354	5	Incorrect motion point for Move instruction	Enter the correct parameters
36355	5	Incorrect control parameter for Move instruction	Enter the correct parameters
36608	5	No input parameters for Arch3 instruction	Please enter parameters
36609	5	No motion point parameter for Arch3 instruction	Please enter parameters
36610	5	Incorrect motion point for Arch3 instruction	Enter the correct parameters
36611	5	Incorrect control parameter for Arch3 instruction	Enter the correct parameters
36864	5	No input parameters for Jump instruction	Please enter parameters
36865	5	No motion point parameter for Jump instruction	Please enter parameters
36866	5	Incorrect motion point for Jump instruction	Enter the correct parameters
36867	5	Incorrect control parameter for Jump instruction	Enter the correct parameters
40960	5	No input parameters for Circle3 instruction	Please enter parameters
40961	5	No motion point parameter for Circle3 instruction	Please enter parameters
40962	5	Incorrect motion point for Circle3 instruction	Enter the correct parameters
40963	5	Incorrect control parameter for Circle3 instruction	Enter the correct parameters
45056	5	Circle3 Option Error	Enter the correct parameters
45057	5	Jump Option Error	Enter the correct parameters

ID	Level	Description	Solution
45058	5	Arch Option Error	Enter the correct parameters
45059	5	Arch3 Option Error	Enter the correct parameters
45060	5	Jerk Option Error	Enter the correct parameters
45061	5	JerkR Option Error	Enter the correct parameters
45062	5	JerkS Option Error	Enter the correct parameters
45063	5	Accel Option Error	Enter the correct parameters
45064	5	AccelR Option Error	Enter the correct parameters
45065	5	AccelS Option Error	Enter the correct parameters
45066	5	SpeedFactor Option Error	Enter the correct parameters
45067	5	Speed Option Error	Enter the correct parameters
45068	5	SpeedR Option Error	Enter the correct parameters
45069	5	Limz Option Error	Enter the correct parameters
45070	5	CP Option Error	Enter the correct parameters
45071	5	DO Option Error	Enter the correct parameters
45072	5	Go Option Error	Enter the correct parameters
45073	5	Move Option Error	Enter the correct parameters
45074	5	MoveJ Option Error	Enter the correct parameters
45075	5	Ecp Option Error	Enter the correct parameters
45076	5	EcpSet Option Error	Enter the correct parameters
45077	5	SetExitMode Option Error	Enter the correct parameters
32768	5	No input parameters for speedFactor instruction	Enter the correct parameters
32769	5	Input parameters of speedFactor instruction out of range	Enter the correct parameters
32770	5	DO input parameters Error	Enter the correct parameters
32771	5	DI input parameters Error	Enter the correct parameters
36100	5	No input parameters for movej instruction	Enter the correct parameters
36101	5	No motion point parameter for movej instruction	Enter the correct parameters
36102	5	No motion point parameter for movej instruction	Enter the correct parameters

ID	Level	Description	Solution
36103	5	Incorrect motion point for RP instruction	Enter the correct parameters
36104	5	Incorrect offset for RP instruction	Enter the correct parameters
36105	5	Incorrect motion point for RJ instruction	Enter the correct parameters
36106	5	Incorrect offset for RJ instruction	Enter the correct parameters
36107	5	No input parameters for GoR instruction	Enter the correct parameters
36108	5	Incorrect motion point for GoR instruction	Enter the correct parameters
36109	5	No input parameters for MoveJR instruction	Enter the correct parameters
36110	5	Incorrect motion point for MoveJR instruction	Enter the correct parameters
45079	5	loadSwitch Option Error	Enter the correct parameters
45080	5	loadSet Options Error	Enter the correct parameters
45081	5	CPPParamErrorOption	Enter the correct parameters
45082	5	TOOLParamErrorOption	Enter the correct parameters
45083	5	USERParamErrorOption	Enter the correct parameters
45084	5	SPEEDParamErrorOption	Enter the correct parameters
45085	5	SPEEDSPParamErrorOption	Enter the correct parameters
45086	5	ACCELParamErrorOption	Enter the correct parameters
45087	5	ACCELSParamErrorOption	Enter the correct parameters
45088	5	ARCHParamErrorOption	Enter the correct parameters
45089	5	STARTParamErrorOption	Enter the correct parameters
45090	5	ZLIMITParamErrorOption	Enter the correct parameters
45091	5	ENDParamErrorOption	Enter the correct parameters
45092	5	SYNCaramErrorOption	Enter the correct parameters
45093	5	ARMPParamErrorOption	Enter the correct parameters
45312	5	loadSwitch Option Error	Enter the correct parameters
45313	5	loadSet Options Error	Enter the correct parameters
49152	5	Enable remote control when enabled	Enter the correct parameters
36111	5	No input parameters for GoIO instruction	Enter the correct parameters

ID	Level	Description	Solution
36112	5	Incorrect motion point for GoIO instruction	Enter the correct parameters
36113	5	Incorrect parameters for GoIO instruction	Enter the correct parameters
36114	5	No input parameters for MoveIO instruction	Enter the correct parameters
36115	5	Incorrect motion point for MoveIO instruction	Enter the correct parameters
36116	5	Incorrect parameters for MoveIO instruction	Enter the correct parameters
36117	5	No input parameters for MoveJIO instruction	Enter the correct parameters
36118	5	Incorrect motion point for MoveJIO instruction	Enter the correct parameters
36119	5	No input parameters for MoveJIO instruction	Enter the correct parameters