

# **M1 Pro Hardware User Guide**

---

Issue: V1.1

Date: 2021-07-08

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

Website: [www.dobot.cc](http://www.dobot.cc)

## Preface

### Purpose

This Document describes the functions, technical specifications, installation guide of Dobot M1 Pro robot, making it easy for users to fully understand and use it.

### Intended Audience

This document is intended for:





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

### Change History

Date	Change Description
2021/04/23	The first releases.
2021/07/08	Add the general security.

### 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

<b>1. Security Precautions .....</b>	<b>1</b>
1.1 Security Warning Sign .....	1
1.2 General Security.....	1
1.3 Personal Security .....	4
<b>2. Overview .....</b>	<b>6</b>
2.1 Technical Specifications .....	7
2.2 Robot Dimension .....	8
2.2.1 M1 Pro Dimension .....	8
2.2.2 End-effector Dimension .....	8
2.3 Robot Workspace .....	9
2.4 End Flange Size .....	10
2.5 Stop Time and Angle.....	10
2.6 Zero Calibration Description .....	10
2.7 Factory point .....	10
2.8 Product Features .....	11
2.8.1 Arm Orientation.....	11
2.8.2 Coordinate System.....	12
<b>3. Electrical Specifications.....</b>	<b>16</b>
3.1 Interface of Base .....	16
3.1.1 External Interface Board Description .....	16
3.1.2 Digital Input .....	17
3.1.3 Digital Output.....	18
3.2 End-effector I/O Interface Description .....	19
<b>4. Installation .....</b>	<b>20</b>
4.1 Installation Environment.....	20
4.2 Installation Location .....	20
<b>5. Maintenance and Repair .....</b>	<b>21</b>
5.1 Safety Instructions .....	21
5.2 Body Maintenance .....	21









## 1. Security Precautions

This topic describes the security precautions that should be noticed when using this product. Please read this document carefully before using the robot for the first time. This product needs to be carried out in an environment meeting design specification. You cannot remold the product without authorization, otherwise, it could lead to product failure, and even personal injury, electric shock, fire, etc. People who use this product for system design and manufacture must be trained by our company, relevant institution, or must have the same professional skills. The installation personnel, operators, teaching personnel, programmers and system developers of the robot must read this document carefully and use the robot strictly according to the regulations of this document strictly.

### 1.1 Security Warning Sign

The following safety warning signs may appear in this manual, and their meanings are as follows.

Sign	Description
 DANGER	Indicates a high degree of potential danger, which, if unavoidable, will result in death or serious injury
 ELECTRICITY	Dangerous power consumption will soon be caused. If it cannot be avoided, it will cause personal injury or serious injury to the equipment.
 HOT	May cause dangerous hot surfaces, if touched, may cause personal injury
 WARNING	Indicates that there is a moderate or low potential hazard. If it cannot be avoided, it may cause minor injuries to the equipment and damage to the equipment.
 ATTENTION	Indicates a potential risk, and ignoring these texts may result in damage to the robotic arm, loss of data, or unpredictable results
 NOTICE	A situation that, if unavoidable, can cause personal injury or equipment damage  For items marked with such symbols, depending on the specific situation, there is sometimes the possibility of significant consequences

### 1.2 General Security

The following security rules should be followed when using the robot for industrial design and manufacture.



- Robot is electrical equipment. Non-professional technicians cannot modify the

circuit, otherwise, it is vulnerable to injury the device or the person.

- You should comply with the local laws and regulations when operating the robot. The security precautions in this document are only supplemental to the local laws and regulations.
- Please use the robot in the specified environment scope. If not, exceeding the specifications or load conditions will shorten the service life of the robot, even damage it.
- Please ensure that the robot is operated under the security conditions and there is no harmful object around the robot.
- Turning on or off the power continually may result in that the performance of the main circuit components inside the robot is degraded. If turning on or off the power continually is required, please keep frequency less than once a minute.

### NOTICE

- The personnel responsible for installation, operation and maintenance of equipment must first undergo rigorous training, understand various safety precautions, and master the correct operation and maintenance methods before they can operate and maintain equipment.
- Personnel without professional training shall not disassemble and repair the equipment without authorization. If the device fails, please contact Shenzhen Yuejiang Technology Co., Ltd technical support engineer in time.
- Be sure to carry out daily inspections and regular maintenance, and replace faulty components in time to ensure the safe operation of the equipment.
- If the equipment is scrapped, please comply with relevant laws to properly handle industrial waste and protect the environment.
- In order to prevent personnel from accidentally entering the working space of the robotic arm, be sure to set up safety fence to prevent personnel from entering the hazardous area.
- Before operating the robot, make sure that no one is inside the safety fence. When operating the robot, be sure to operate outside the safety fence.
- Do not expose the robot to permanent magnetic fields all the time. Strong magnetic fields can cause damage to the robot.
- Shenzhen Yuejiang Technology Co., Ltd. assumes no responsibility for robot damage or personal injury caused by failure to follow product instructions or other improper operations.
- Shenzhen Yuejiang Technology Co., Ltd. is not responsible for the damage caused during the transportation and handling of equipment.
- Please make sure that the robot is in the packing posture before packaging, and the brakes on each axis are normal.

- When the robot is transported, the packaging needs to be fixed to ensure that the robot is stable.
- After removing the outer packaging, please make sure that the robot maintains the original packing posture and the brakes on each axis are normal.
- During the commissioning process, it is necessary to confirm that no relevant personnel and equipment (include computer used for debugging) stay in the dangerous area of the machine.
- If necessary, wear corresponding safety protective equipment, such as safety helmets, safety shoes (with non-slip soles), face shields, protective glasses and gloves. Inappropriate clothing may cause personal injury.
- In order to prevent personnel from entering the working space of the robot arm by mistake, please set up safety barriers to prevent personnel from entering the hazardous area.
- Do not enter the working space of the manipulator at will during operating the robot, otherwise cause injury to the robot or yourself.
- When an abnormality occurs in the mechanical arm, it is necessary to ensure that the machine is stopped and then checked.
- If the controller needs to be restarted due to power failure, when restarting, the robot must be manually returned to the initial position of the automatic operation program before restarting the automatic operation.
- Before maintenance and wiring work, the power supply must be cut off, and the sign **No power supply** must be put on. Otherwise, electric shock and personal injury may result.
- Please contact our technical support staff for the disassembly and repair of the robot.
- Maintenance and repair work must be carried out by designated personnel, otherwise electric shock and personal injury may result.
- If the brake is manually released, the robot may move because of the action of gravity. So, when manually releasing the brake, please ensure that the robot body and the tools or workpieces installed on the robot are effectively supported.
- In order to prevent electric shock, when replacing parts, please turn off the circuit breaker in advance and cut off the main power before proceeding.
- Turn off the main power supply for 5 minutes before replacing parts.
- The replacement operation must be performed by the specified operator.
- The robot is designed and tested according to the group I class A engineering medical robot standard. In order to reduce the radio interference **in in light** industry or family environment, please take protective measures.
- It is prohibited to operate the robot in strong radiation environment, for example,

RF source without shielding, otherwise, it could lead to robot abnormally.

- When the operator commissioning or operates the equipment, it shall be done in the safe space of the equipment.

 **WARNING**

- In order to protect the equipment and personal safety, when turning off the power, please press the switch, then unplug the AC power cable. The power is disconnected by plug and socket.
- Before the operation, please wear protective clothing, such as antistatic uniform, protective gloves, and protective shoes.
- It is prohibited to modify or remove the nameplates, instructions, icons, and marks on the robot and the related equipment.
- Before operating and maintaining the robot, the personnel responsible for the installation, operation and maintenance must be trained to understand the various security precautions and to master the correct methods of operation, and maintenance.
- Be careful during the robot **carrying or installing**. Please follow the instructions on the packing box to put down the robot gently and place it correctly in direction of the arrow.
- Please use the matched cables when connecting a robot to internal or external equipment for personal security and equipment protection.
- Please ensure that robot and tools are installed correctly.
- Please ensure that the robot has enough space to move freely.
- If the robot is damaged, please do not continue to use it.
- Any impact will release a lot of kinetic energy, which is much higher than that under high speed and high load.

### 1.3 Personal Security

When operating the robot system, it is necessary to ensure the personal safety of the operator. The general precautions are listed below, please strictly follow.

 **WARNING**

- To reduce the risk of personal injury, please comply with local regulations with regard to the maximum weight one person is permitted to carry.
- Do not touch the terminal blocks or disassemble the equipment with the power **ON**. Otherwise, it may result in an electric shock
- Please confirm that the equipment is well grounded, otherwise it will endanger personal safety.

- Do not touch the terminal blocks or remove the interval circuit components in 10 minutes after the power is shut off, to avoid an electric shock since there is residual capacitance inside the robot.
- Even if the power switch of the robot is already in the **OFF** status, touching the terminal blocks or removing the interval circuit components is not allowed, to avoid an electric shock since there is residual capacitance inside the robot.
- When working with robots, please do not wear loose clothing or jewelry. When operating the robot, make sure that the long hair bundle is behind your head.
- If the robot appears to have stopped during the operation of the equipment, it may be because the robot is waiting for the start signal and is in the state of being about to move. In this case, the robot should also be considered to be in motion, please do not approach the robot.
- Please ensure that the robot establishes safety measures near the operation area, such as guardrails, to protect the operator and surrounding people.

## 2. Overview

Dobot Master 2<sup>nd</sup> generation robotic arm (Dobot M1 Pro for short) focuses on the light industrial market with great potential, and supports teaching, playback, script control, blockly, vision identity and other functions, which is flexibly used in intelligent sorting, circuit board soldering and other automatic production lines, so that it can become the sword to solve practical problems for light industrial users, and can also become the platform to carry the imagination of the maker. Dobot M1 Pro has the following characteristics.

- The integrated design of the driver and controller without external controller simplifies the process of the initial installation and deployment.
- Built-in calibrated servo, harmonic reducer, combined with kinematic algorithm, can make M1 pro **to bring** out the best of strength and speed.
- The rated load is 1.5kg, and the repeatability is  $\pm 0.02\text{mm}$ .
- Various I/O and communication interfaces can meet the use of users in different scenarios.



Figure 2.1 DOBOT M1 Pro

## 2.1 Technical Specifications

Table 2.1 M1 Pro technical parameters

Product	DOBOT M1 Pro	
Model	DT-M1-P4R15-01I	
Weight	15.7kg	
Max load	1.5kg	
Reach	400mm	
Power supply	100~240 VAC, 50/60Hz	
Rated voltage	DC48V	
Installation	Table installation, indoor	
Rated power	192W	
Repeatability	±0.02mm	
Base size	230mm*175mm	
Operation software	DobotSCStudio, DobotStudio2020	
Motion range	J1	±85°
	J2	±135°
	J3	5mm~245mm
	J4	±360°
Joint maximum speed	J1	180 %s
	J2	180 %s
	J3	1000mm/s
	J4	1000 %s
End-effector interface	DI	4
	DO	4
	RS485 (Modbus_RTU)	1
Base interface	DI	16
	DO	16
	ABZ incremental encoder (differential)	1
	Ethernet	2

	USB 2.0	2
Temperature range	Storage temperature: -25°C~55°C Working temperature: 0°C~40°C	
Operating altitude range	≤ 1000 m	
Safety Standard	EN ISO 10218-1:2011 Steel wire and wire products. General. Test methods EN 60204-1:2018 Safety of machinery. Electrical equipment of machines. General requirements EN ISO 12100:2010 Safety of machinery. General principles for design. Risk assessment and risk reduction	
EMC Standard	EN 61000-6-2:2019 Electromagnetic compatibility (EMC). Generic standards. Immunity standard for industrial environments EN 61000-6-4:2019 Electromagnetic compatibility (EMC). Generic standards. Emission standard for industrial environments	

## 2.2 Robot Dimension

### 2.2.1 M1 Pro Dimension

Figure 2.2 shows the dimension of M1 Pro robot.

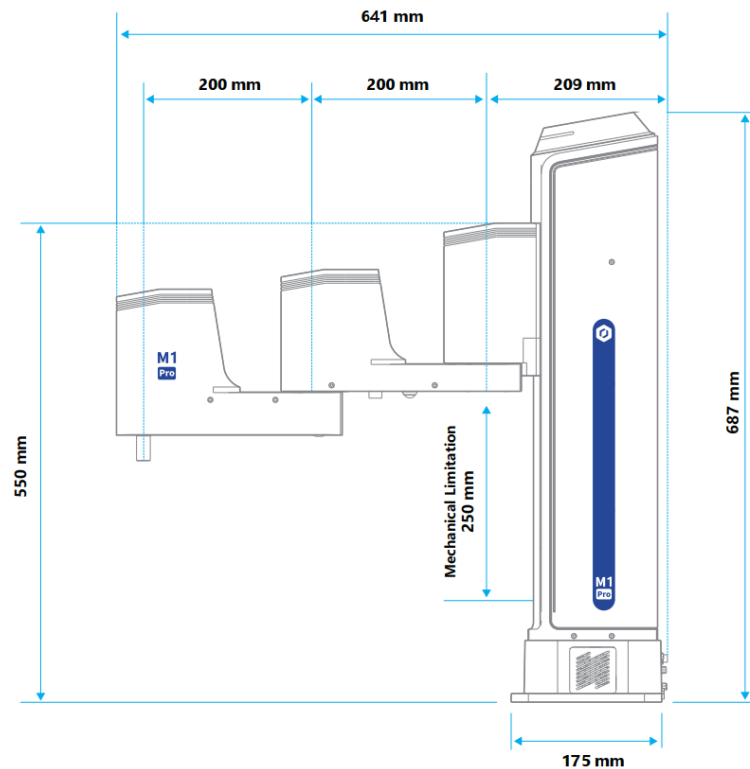


Figure 2.2 M1 Pro dimension

### 2.2.2 End-effector Dimension



You can install the matching gripper, suction cup on the end of the M1 Pro for transporting and intelligent sorting. Figure 2.3 shows the dimension of end-effector.

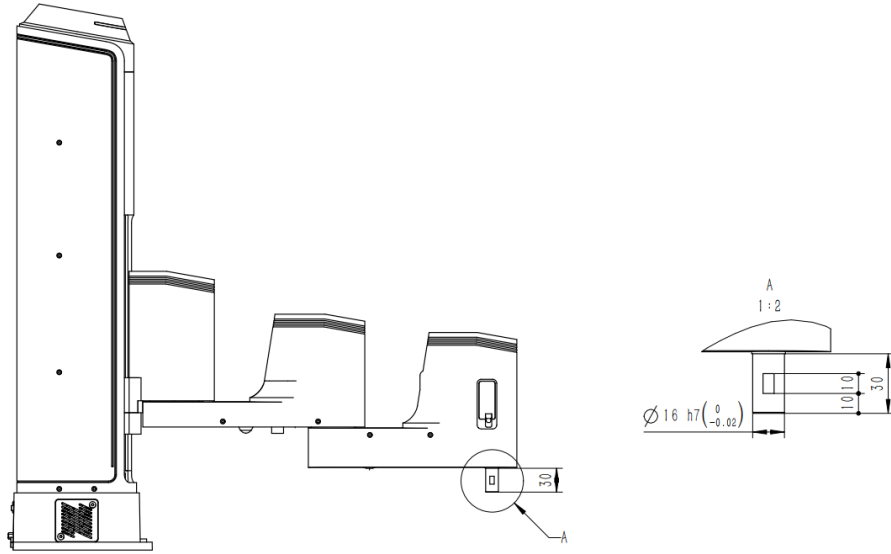


Figure 2.3 End-effector dimension

### 2.3 Robot Workspace

Figure 2.4 shows the workspace of M1 Pro robot.

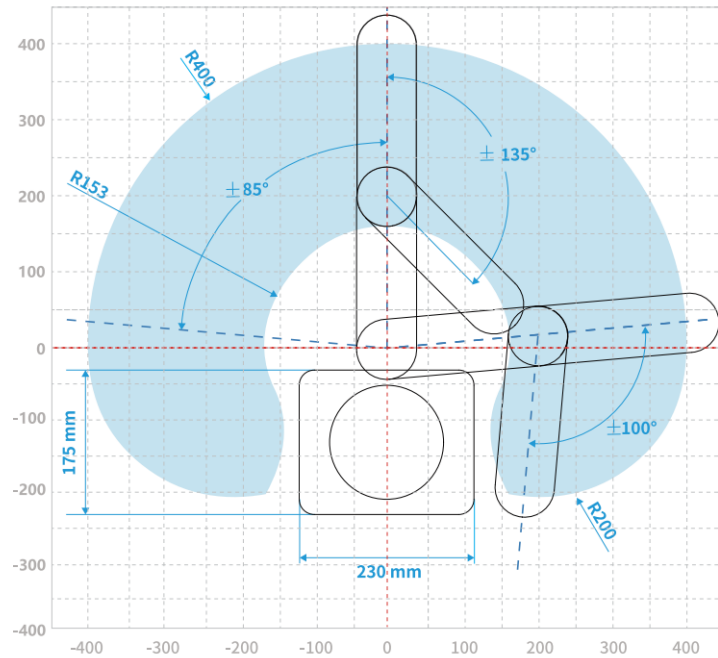


Figure 2.4 M1 Pro robot workspace



When operating the robot, be sure to operate inside the workspace.

## 2.4 End Flange Size

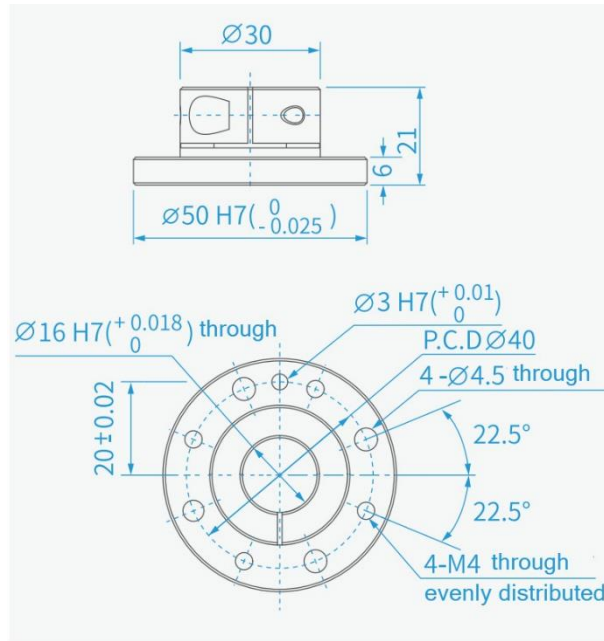


Figure 2.5 End flange size

## 2.5 Stop Time and Angle

The Max. stop time and angle of axis J1, J2, J3 and J4 at the max speed, load and arm stretch are shown below.

Table 2.2 Stop time and angle

Axis	Max. stop angle (°)	Max. stop time (ms)
J1	9.6	103
J2	9.4	102
J4	36.5	100
Axis	Max. stop distance (mm)	Max. stop time (ms)
J3	43	104

## 2.6 Zero Calibration Description

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. For details, please see *DobotSCStudio User Guide (M1 Pro Robot)*.

## 2.7 Factory point

When the robot leaves the factory, moving robot to the factory point can reduce the robot space, easy to pack and transport. Figure 2.6 shows the factory point. The robot has 4 joints, respectively J1, J2, J3 and J4, please see 2.8.2.1 Joint Coordinate System for explanation of joints. The joint angles corresponding to the factory point are:  $J1=0^\circ$ ,  $J2=0^\circ$ ,  $J3=0\text{mm}$ , and  $J4=0^\circ$ . Adjust joint Angles by jog or programming. For details, please see *DobotSCStudio User Guide (M1 Pro Robot)*.

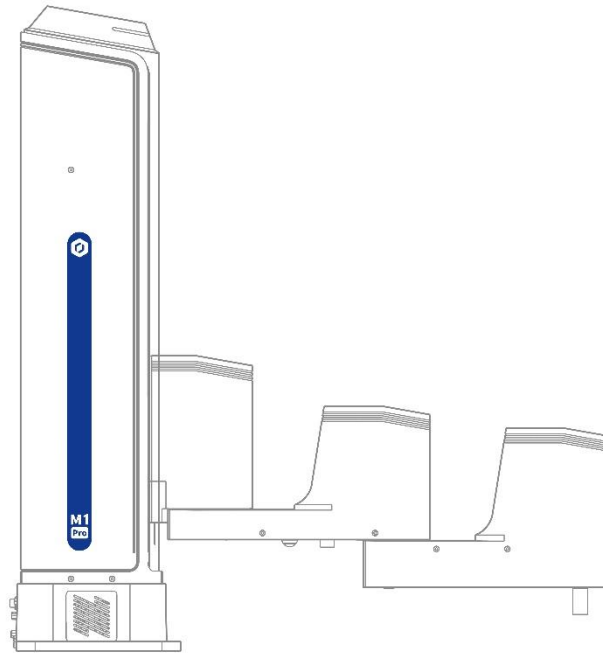


Figure 2.6 Factory point

## 2.8 Product Features

### 2.8.1 Arm Orientation

With two types of the arm orientation (left hand orientation and right hand orientation), M1 Pro can move to nearly any position and orientation within a given work envelope. You need to specify the arm orientation when M1 Pro is moving. If you fail to do so, M1 Pro may move following an unexpected path, resulting in interference with peripheral equipment. The arm orientations are shown in Figure 2.7 and Figure 2.8.

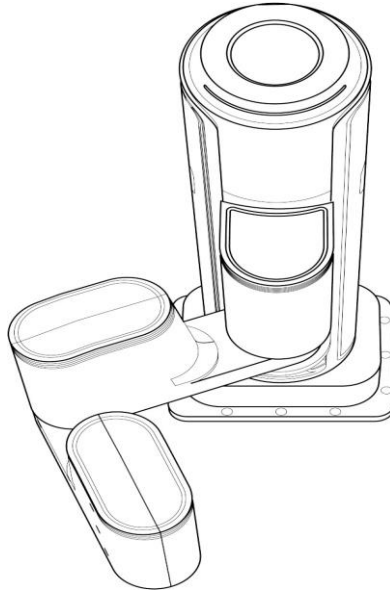


Figure 2.7 Righty hand orientation

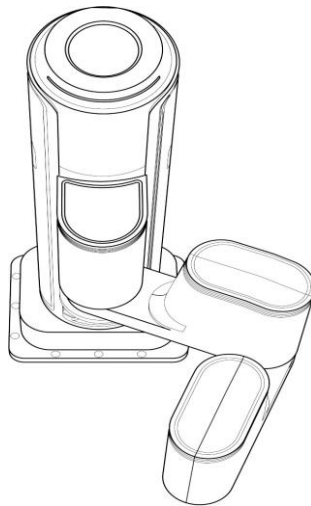


Figure 2.8 Lefty hand orientation

## 2.8.2 Coordinate System

### 2.8.2.1 Joint Coordinate System

The Joint coordinate system is determined by the motion joints.

Figure 2.9 shows the Joint coordinate system of a M1 Pro robot. All the joints are rotating joints.

M1 Pro contains four joints.

- J1, J2, and J4 are the rotating joints, which are located and oriented in the horizontal plane. And their axes are parallel to each other. The positive direction of these joints is counter-clockwise.

- J3 is the moving joint, which is used for the movement of the end effector in the perpendicular plane. The positive direction of J3 is vertical upward.

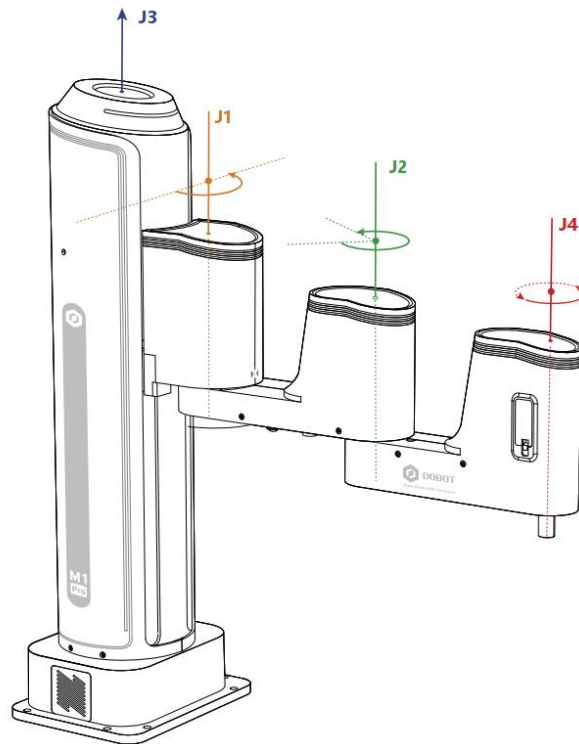


Figure 2.9 Joint coordinate of a M1 Pro robot

### 2.8.2.2 Base Coordinate System

The Base coordinate system is determined by the base. Figure 2.10 shows the Base coordinate system of M1 Pro robot.

- The origin is the axes center of the motor of Rear Arm where Rear Arm is dropped to the bottom of the Z-axis screw.
- The direction of X-axis is perpendicular to the base forward.
- The direction of Y-axis is perpendicular to the base leftward.
- The direction of Z-axis is vertical upward, which is based on the right-hand rule.

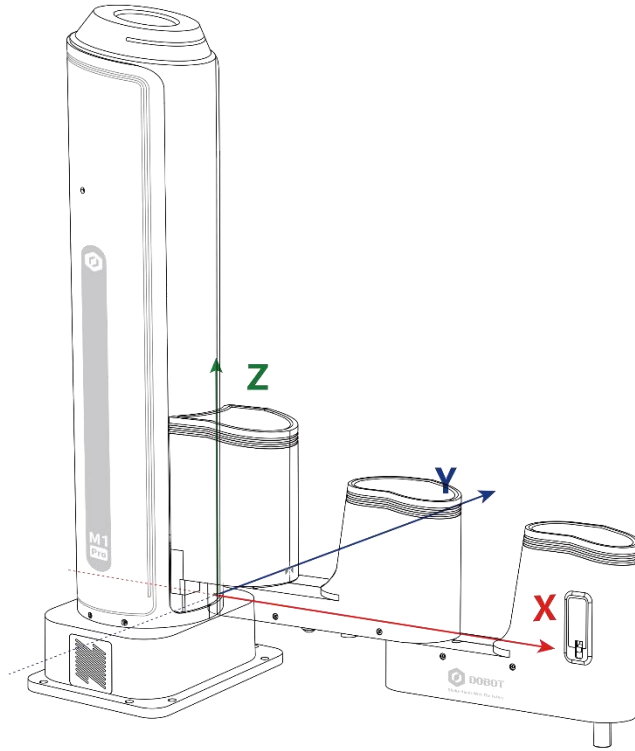


Figure 2.10 Base coordinate system of M1 Pro robot

### 2.8.2.3 Tool Coordinate System

Tool coordinate system is the coordinate system that defines the distance and rotation angle of the offset, of which the origin and orientations vary with the position and attitude of the workpiece located at the robot flange. The 10 types of tool coordinate systems can be defined. Tool 0 coordinate system is the predefined Tool coordinate system which is located at the robot flange without end effector and cannot be changed. And the others can be customized by users. Figure 2.11 shows the default Tool coordinate system of a M1 Pro robot.

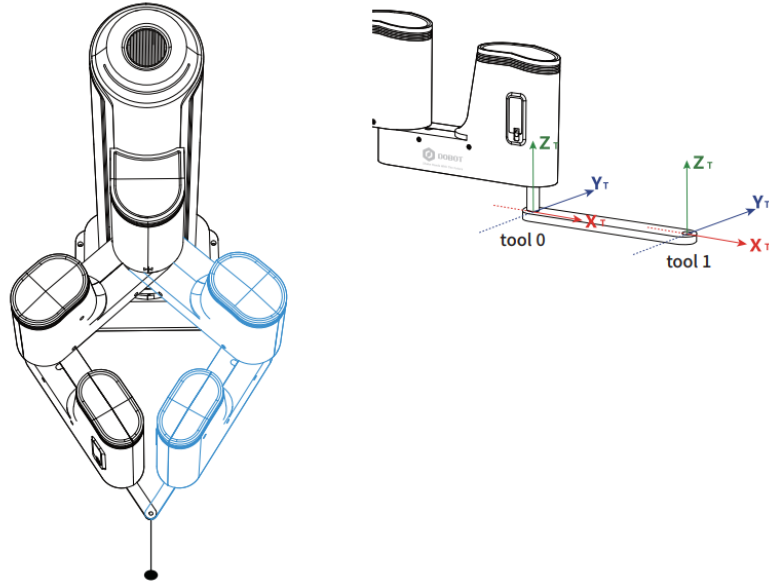


Figure 2.11 The default Tool coordinate system of M1 Pro robot

#### 2.8.2.4 User Coordinate System

The User coordinate system is a movable coordinate system which is used for representing equipment like fixtures, workbenches. The origin and the orientations of axes can be defined based on site requirements, to measure point data within the workspace and arrange tasks conveniently.

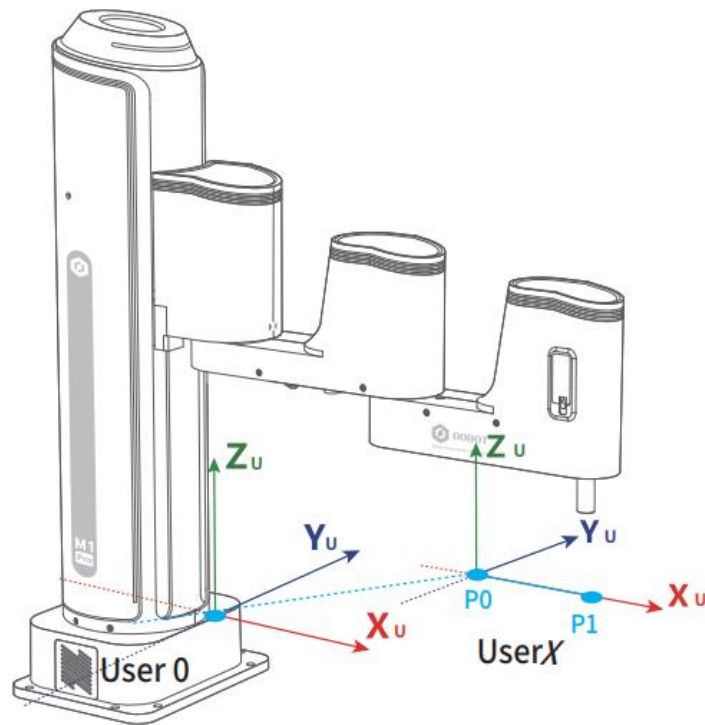


Figure 2.12 The default User coordinate system of M1 PRO robot

### 3. Electrical Specifications

#### 3.1 Interface of Base

##### 3.1.1 External Interface Board Description

Figure 3.1 shows the interface board of the Base. Table 3.1 lists the description.

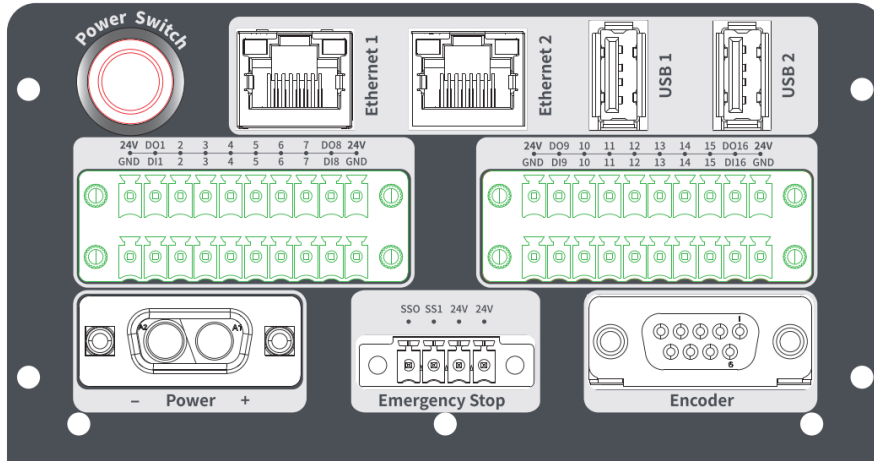


Figure 3.1 Interface board of the base

Table 3.1 Interface description

screen printing	Description
Ethernet1	Ethernet interface The default IP address is 192.168.1.6, which cannot be modified. It can be used for software debugging on the upper computer
Ethernet2	Ethernet interface For connecting to external equipment. The default IP address is 192.168.2.6, which can be modified.
USB1	USB interface For connecting WiFi module, updating firmware, etc
USB2	USB interface For connecting WiFi module, updating firmware, etc
Encoder	Encoder interface For connecting to the conveyor belt for dynamic tracking
Power Switch	For controlling the robot power on and off
Emergency Stop	Emergency stop interface
Power	Power interface For connecting to DC 48V power supply



screen printing	Description
I/O	I/O interface

The Encoder interface of the M1 Pro is shown in Figure 3.2, Table 3.2 lists the description of Encoder interface.

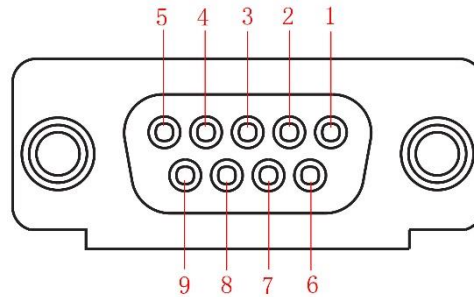


Figure 3.2 Encoder interface

Table 3.2 Encoder Interface description

No.	1	2	3	4	5	6	7	8	9
Description	ABZ_A+	ABZ_A-	ABZ_B+	ABZ_B-	ABZ_Z+	ABZ_Z-	5V	0V	unused

A robot controller contains I/O interfaces, for connecting to external equipment, such as PLC, etc. These I/O interfaces provide 16 digital inputs, 16 digital outputs, as shown in Figure 3.3.

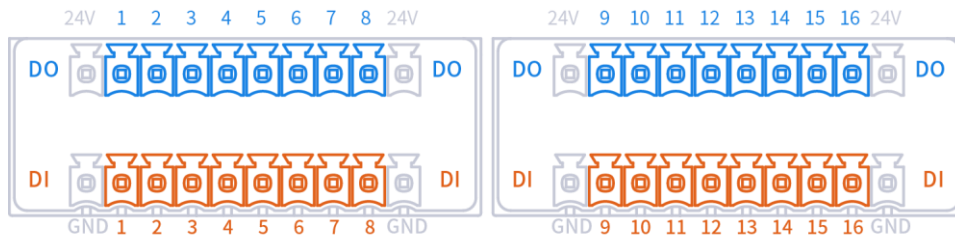


Figure 3.3 I/O interface

#### NOTE

- The output current of every DO can't exceed 500mA.
- The total current can't exceed 2A.

### 3.1.2 Digital Input

Figure 3.4 shows the simple digital input circuit and Table 3.3 lists the technical specifications.

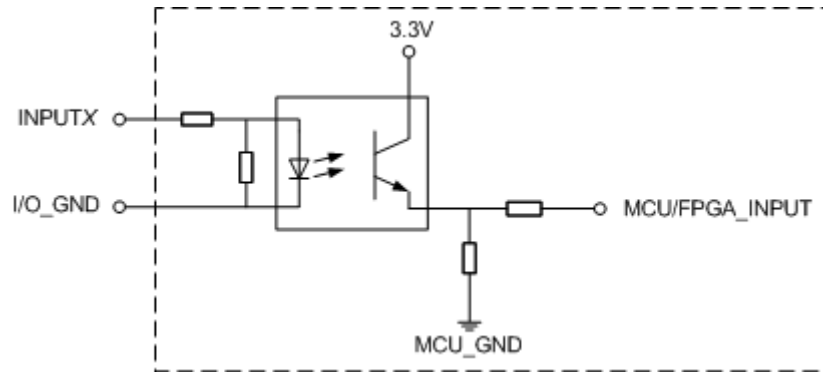


Figure 3.4 Simple digital input circuit

Table 3.3 Technical specifications

Item	Specification
Input channel	16 channels
Connection method	Europe type terminal
Input type	PNP
Input voltage (DC)	24V ±10%
Isolation method	Optical coupling isolation

### 3.1.3 Digital Output

Figure 3.5 shows the simple digital output circuit and Table 3.4 lists the technical specifications.

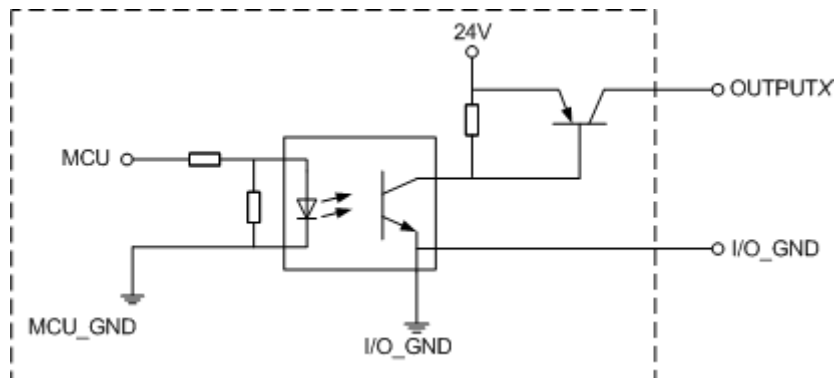


Figure 3.5 Simple digital output circuit

Table 3.4 Technical specifications

Item	Specification
Output channel	16 channels
Connection method	Europe type terminal
Output type	PNP
Power supply (DC)	24V ±10%
Load current of single channel	500mA
Output current	2A
Isolation method	Magnetic isolation

### 3.2 End-effector I/O Interface Description

The end-effector I/O interfaces of the M1 Pro include a RS485 interface, 4 digital inputs and 4 digital outputs, as shown in Figure 3.6, Table 3.5 lists the description of End-effector I/O interface.

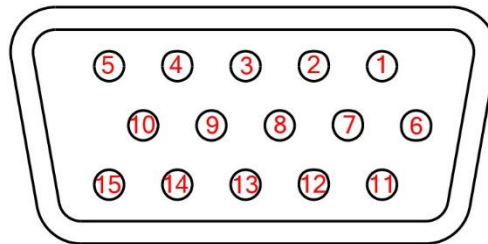


Figure 3.6 End-effector I/O interface

Table 3.5 End-effector I/O interface description

No.	1	2	3	4	5	6	7	8
Description	24V	DO17	DO18	DO19	DO20	unused	unused	unused
No.	9	10	11	12	13	14	15	
Description	RS485 A	RS485 B	DI17	DI18	DI19	DI20	0V	

## 4. Installation

### 4.1 Installation Environment

To maintain the controller and robot performance and to ensure the safety, please place them in an environment with the following conditions.

- Install indoors with good ventilation.
- Keep away from excessive and shock.
- Keep away from direct sunlight.
- Keep away from dust, oily smoke, salinity, metal powder, corrosive gases, and other contaminants.
- Keep away from flammable.
- Keep away from cutting and grinding fluids
- Keep away from sources of electromagnetic interference.
- When the robot is installed, corresponding measures should be taken for positioning. You must use six hexagon socket bolts M6 (GB/T 3098.1-2010) and tighten the base of the robot with 17 N • m torque.
- When the robot is installed, the robot must be fixed on a sufficiently strong base. The base must be able to withstand the reaction force of the robot during acceleration and deceleration and the static weight of the robot and the workpiece.

### 4.2 Installation Location

The stability of a robot depends on the installation. You can design the platform according to the size of the hole of the base and the real environment for mounting a robot. And the installation height of the robot should be above 0.6 meters. The platform must not only bear the robot but also bear the dynamic force by the maximum acceleration. Note the following before mounting the robot.

- Design the platform according to the robot's workspace, and ensure that the robot moves without interference.
- Keep the platform level which is used to mount a robot.

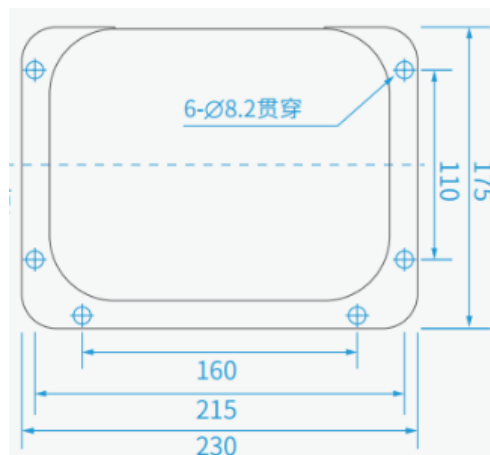


Figure 4.1 Robot base size

## 5. Maintenance and Repair

Maintenance and repairing must be performed in compliance with all safety instructions in this manual.

The purpose of maintenance and repairing is to ensure that the system is kept operational, or to return the system to an operational state in the event of a fault. Repairing includes troubleshooting in addition to the actual repair itself.

Repairing must be performed by an authorized system integrator or Dobot staff.

Robots or parts returned to Dobot should be as the following instructions.

- Remove all parts that do not belong to Dobot.
- Before returning to Dobot, please make a backup copy of the files. Dobot will not be responsible for the loss of programs, data or files stored in robot.
- The robot should move to the package point before returning to Dobot. For details, please see 2.7 Factory point.

### 5.1 Safety Instructions

The following safety procedures and warnings must be observed during the operation of the robot or controller:

- Replace faulty components **using** new components with the same article number or equivalent components approved by Dobot.
- Reactivate any deactivated safety measures immediately after the repairing is completed.
- Record all repairs and save them in the technical document with the robot system.
- Remove the main input cables from the back of the robot to ensure that it is completely unpowered. Take necessary precautions to prevent other **persons** from powering on the system during the repair period.
- Observe ESD regulations during the disassembly of the parts of the robot.
- Prevent water and dust from entering the robot.

### 5.2 Body Maintenance

In order for the robot to maintain high performance for a long time, a maintenance check must be carried out. The person in charge of overhaul must prepare an overhaul plan and carry out an inspection. The overhaul items are shown below.

Table 5.1 Overhaul item

Cycle			Overhaul Item	Overhaul essential
Daily	3 month	6 month		
√			Robot clean	Wipe off dirt, dust, cutting residue on the body

				with water or 10% alcohol
√			Cable, cable protective cover	Observe the moving part of the cable, check whether the cable is damaged, whether there is local bending or distortion; Check whether the cable protective cover is damaged, etc.
	√		Joint bolts	Check the torque based on the specified tightening torque table (Push aside the rubber to check)
√			Tool mounting bolts	Check the torque based on the specified tightening torque table
√			Motor	Abnormal heating or sound confirmation
√			Brake	Check whether the robot arm or end-effector will fall when the servo is powered off
	√		Synchronous belt	Check whether the synchronous belt is worn out, elongated, broken, etc.

Table 5.2 lists the bolted tightening torque table.

Table 5.2 Bolted tightening torque table

Bolt size	Hexagon sock-et head cap scre	Hexagon socket countersunk flat cap head screw	Hexagon screw (rubber part)
2.5 mm	0.7 N•m		-
3 mm	2 N•m	-	1.2 N•m
4 mm	4 N•m	2 N•m	-
5 mm	9 N•m	-	-

The tightening torques will vary depending on the type of base metal or bolt. When not specified, please contact Dobot technical engineer.

In addition, overhauls are required every 10,000 hours of operation time or every 3 years. If you are not clear about the maintenance processes, please contact Dobot technical engineer.



**DOBOT**

User Guide

---

# **DobotSCStudio User Guide (M1 Pro)**

---

Issue: V1.4.8

Date: 2021-04-29

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](http://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 M1 Pro.

### Intended Audience

This document is intended for:





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

### Change History

Date	Change Description
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

<b>1. Overview .....</b>	<b>1</b>
1.1 Main Interface Description .....	1
<b>2. Fast Connection.....</b>	<b>3</b>
<b>3. Function Description .....</b>	<b>6</b>
3.1 Enabling .....	6
3.2 Setting Global Velocity Rate.....	6
3.3 Alarm Description.....	6
3.4 Jogging.....	7
3.5 Programming .....	9
3.5.1 Project Description .....	9
3.5.2 Programming Interface Description .....	9
3.5.3 Programming Description.....	11
3.6 Parameter .....	22
3.6.1 Setting User Coordinate System.....	22
3.6.2 Setting Tool Coordinate System .....	25
3.6.3 I/O Monitor.....	28
3.6.4 Controller Setting .....	29
3.6.5 Remote Control .....	30
3.6.6 RobotParams .....	34
3.6.7 RobotSetting .....	37
3.7 ToolConfig.....	40
3.7.1 BasicConfig .....	40
3.7.2 PlugInfo .....	40
3.7.3 Log.....	40
3.7.4 Network Service .....	41
3.7.5 Tools .....	42
3.7.6 VirtualRobot .....	42
<b>4. Program Language .....</b>	<b>43</b>
4.1 Arithmetic Operators .....	43
4.2 Relational Operator.....	43
4.3 Logical Operators.....	43
4.4 General Keywords .....	44
4.5 General Symbol .....	44
4.6 Processing Control Commands.....	44
4.7 Global Variable .....	44
4.8 Motion Commands.....	45
4.9 Motion Parameter Commands .....	52
4.10 Input/output Commands.....	55
4.11 Program Managing Commands .....	57
4.12 Pose Getting Command .....	59
4.13 TCP.....	60
4.14 UDP .....	64

4.15 Modbus .....	68
4.15.1 Modbus Register Description .....	68
4.15.2 Command Description .....	69
<b>5. Process.....</b>	<b>73</b>
5.1 Conveyor Tracking.....	73
5.1.1 Overview .....	73
5.1.2 Building Environment .....	73
5.1.3 Calibrating Conveyor .....	75
5.1.4 Configuring Conveyor.....	78
5.1.5 Example.....	92
5.2 Palletizing .....	96
5.2.1 Overview .....	96
5.2.2 Setting Pallet.....	96
5.2.3 Example.....	100
<b>Appendix A Servo Alarm Description .....</b>	<b>102</b>
<b>Appendix B Controller Alarm Description .....</b>	<b>107</b>

# 1. Overview

DobotSCStudio is an industrial robot programming platform launched by Yuejiang, which is suitable for the whole series of industrial robots (SA / SR / CR / M1 Pro) interface, innovative interactive programming, and supporting user secondary development. 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 Main Interface Description

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

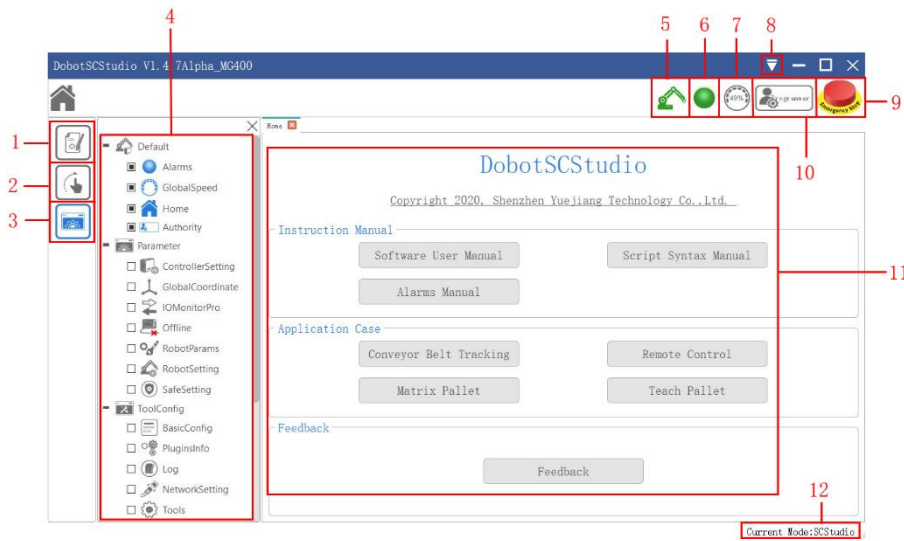


Figure 1.1 Main interface

Table 1.1 Interface description

No.	Description
1	Project You can build or import a project, and debug or run it
2	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 Jog the robot in the Cartesian coordinate system: From top to bottom, jog the X, Y, Z, R
3	System You can set system configurations. Such as Network Setting, RobotParams, Coordinate, Process, etc.
4	System bar

No.	Description
5	Click this button to enable or disable the motor
6	Check robot alarm When an alarm is triggered, this icon will turn red You can check the alarm details on the operation panel and clear it
7	Set global velocity rate
8	IP setting or check update 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
9	Emergency stop switch Press and hold it in an emergency, the drive power supply of M1 Pro will be powered off for emergency braking
10	Select user mode <ul style="list-style-type: none"><li>• Watcher: check the system status, I/O status, robot pose, and alarms</li><li>• Operator: Operate a robot based on the existing scripts without programming</li><li>• Programmer: On the basis of operator authority, you can program and teach</li><li>• Manager: On the basis of programmer authority, you can set or modify parameters</li></ul> Please select user mode based on site requirements Default password: admin. You can modify the password on the ToolConfig > BasicConfig > UserMode page in the Manager mode
11	Interactive window
12	Show the current running mode Running mode: I/O, Modbus, SCStudio

## 2. Fast Connection

DobotSCStudio can communicate with the M1 Pro directly through Ethernet1. At this point, the IP address of the M1 Pro 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.

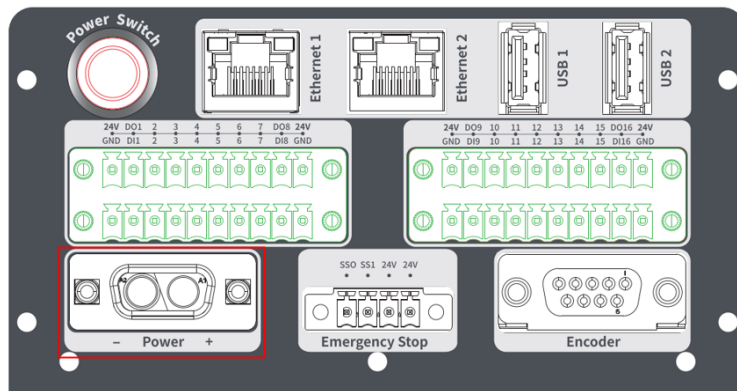


Figure 2.1 Connect to robot Power Switch interface

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

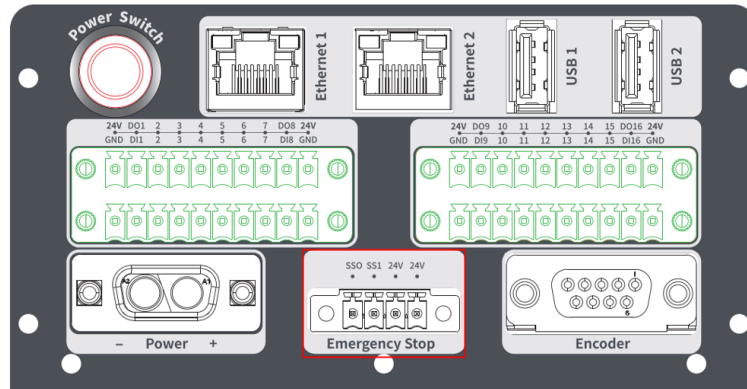


Figure 2.2 Connect to Emergency Stop interface

- 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**.  
Double-click **Internet Protocol Version 4(TCP/IPv4)**.
- Step 7** 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.

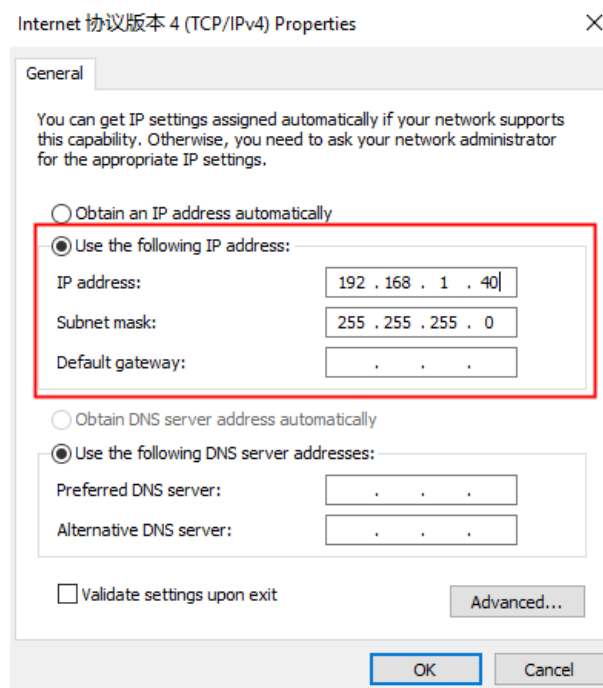



Figure 2.3 IP address modification

- Step 8** Click **OK**.
- Step 9** Click  > **IP settings...** on the upper right pane of the DobotSCStudio page and select the robot's IP address, then click **OK**.

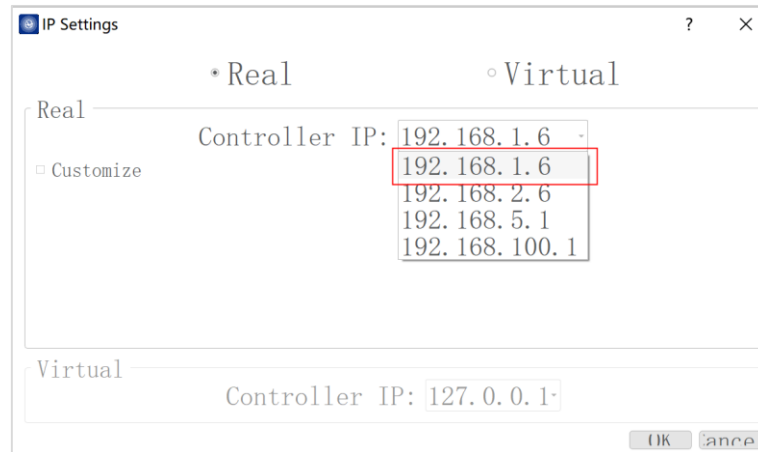


Figure 2.4 IP setting

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

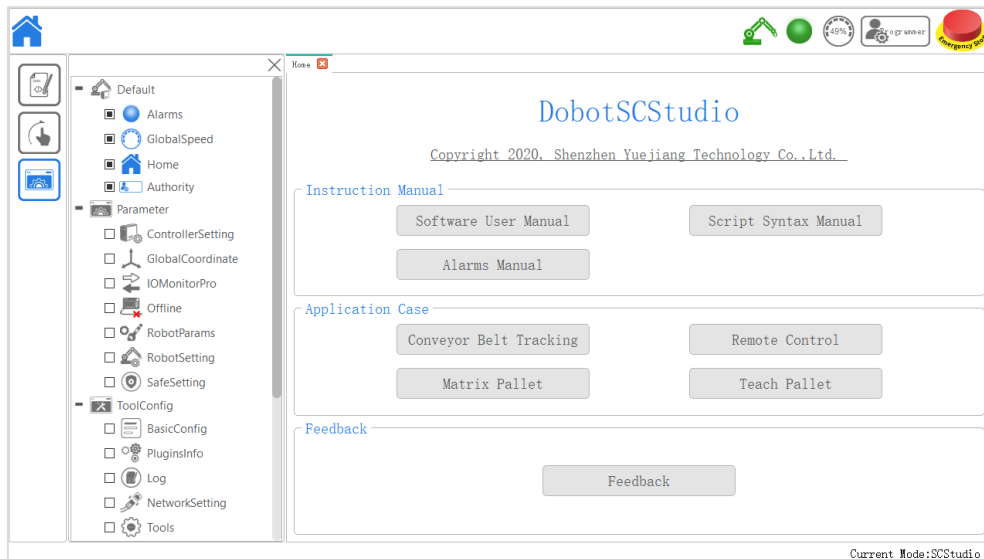





Figure 2.5 Connecting successful




### 3. Function Description

#### 3.1 Enabling

Click  to enable M1 Pro. When the icon  turns into , the M1 Pro can be controlled by running the program or Jogging.

#### 3.2 Setting Global Velocity Rate

Please click  and then click buttons to increase or decrease the global velocity ratio by 1%, 5%, 10%, 25% and 50% on the operation panel, as shown in Figure 3.1. 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.

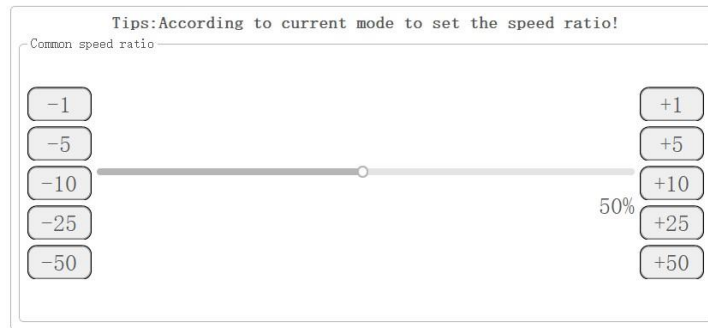


Figure 3.1 Modify the global velocity rate

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 3.6.6 *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 M1 Pro, 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 M1 Pro.

	Id	Type	Level	Cause	Solution
1	0x45	Controller Error	0	Joint3 exceeds negative limit	




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 of jogging panel.

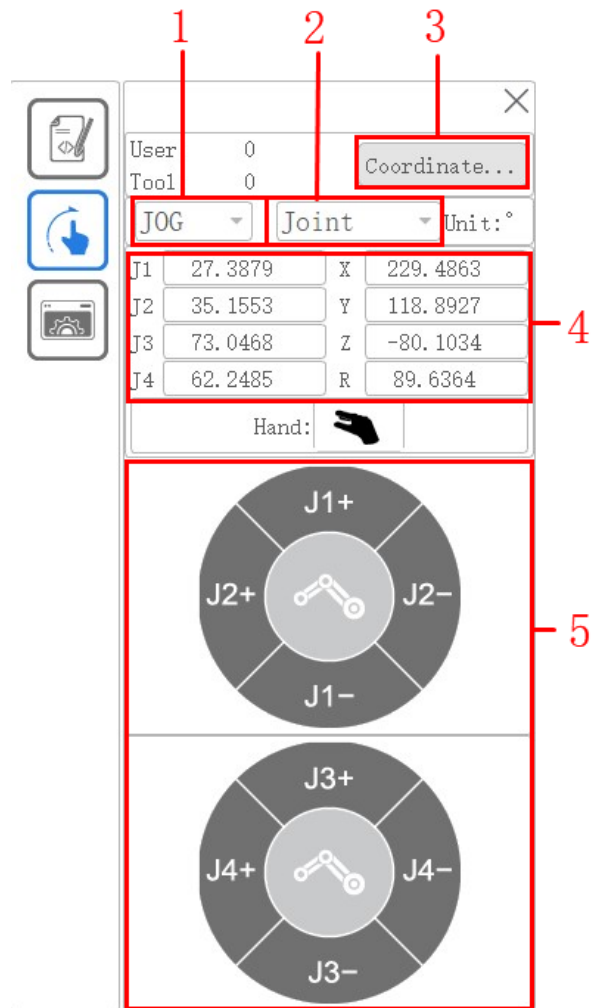


Figure 3.3 Jogging panel

Table 3.1 Description of jogging panel

No.	Description
1	Step mode You can select the right step in the Step mode. The step supports 0.1, 1, 5, and 10. In Cartesian coordinate system, the step unit is mm, and in Joint coordinate system, the step unit is °.
2	Jogging type It supports two types: Joint coordinate system, user coordinate system
3	Coordinate system According to the actual needs, the user 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

No.	Description
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 the X, Y, Z, and R

## 3.5 Programming

### 3.5.1 Project Description

The robot program is managed in project form, including teaching points list, global variables, and program files. Figure 3.4 shows the project structure.

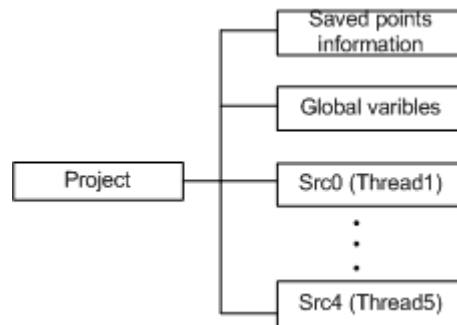


Figure 3.4 Project structure

- In script, multiple threads are supported. Up to 5 threads can be executed simultaneously. Src0.lua is the main thread, other threads are sub threads, which run program parallel to the main thread.
- In the sub threads, the motion commands cannot be called. Only the main thread supports motion commands.
- In addition, global variable module is only used to define global variables and module functions. The motion commands cannot be called here.

### 3.5.2 Programming Interface Description

Figure 3.5 shows the programming panel and Table 3.2 lists its description.

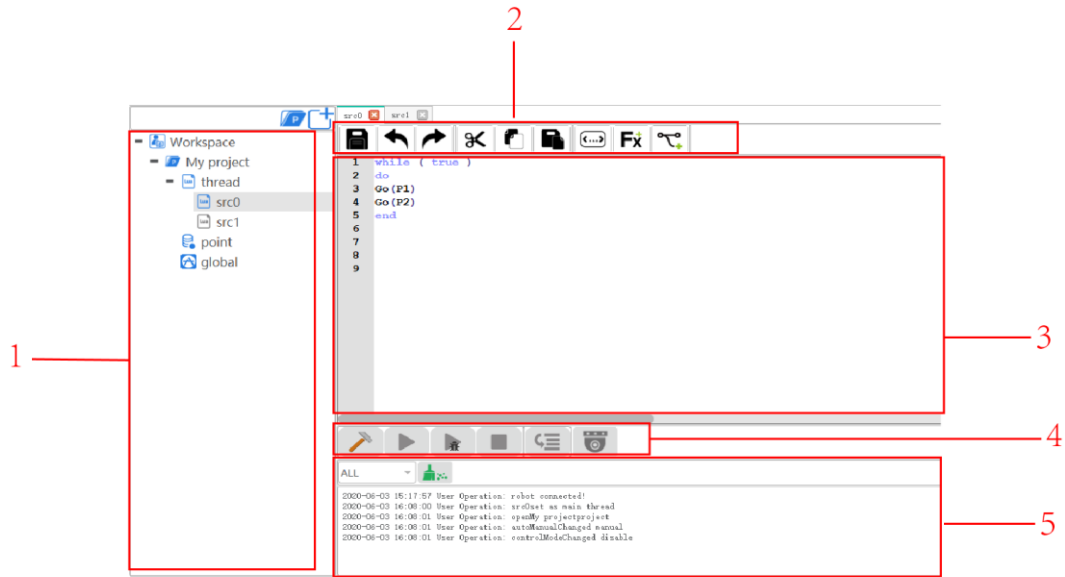


Figure 3.5 Programming panel








Table 3.2 Programming panel description

No.	Description
1	Project files <ul style="list-style-type: none"> <li>Point: Teach points. For details, please see 3.5.3.3 <i>Teaching points</i>.</li> <li>Global: Define and initialize global variables or functions</li> <li>Src0~Src4: Multithreaded files. The number of threads is related to CPU that is set when creating a project. Up to 5 threads can be executed simultaneously</li> </ul>
2	Common buttons. For details, please see Table 3.3
3	Programming area
4	Running button, for details, please see Table 3.5
5	Debug result

Table 3.3 lists the common button description.

Table 3.3 Common button description

Icon	Description
	Save the project
	Cancel

Icon	Description
	Redo
	Copy the selected codes
	Cut the selected codes
	Paste the selected codes
	Motion command libraries, for details, please see 4 Program Language
	Code comment
	Common operation instructions and process control instructions

### 3.5.3 Programming Description

Figure 3.6 shows the programming process.

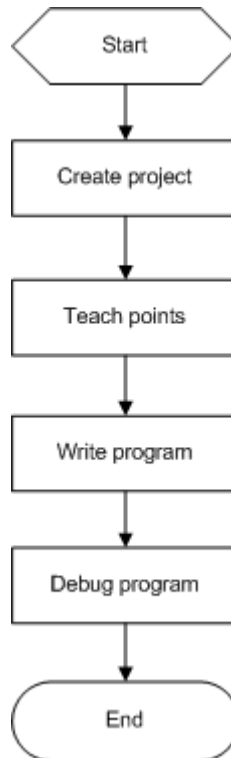


Figure 3.6 Programming process

### 3.5.3.1 Creating Project

#### Prerequisites

The robot has been powered on.

#### Procedure

**Step 1** Click .

The programming page is displayed, as shown in Figure 3.7.

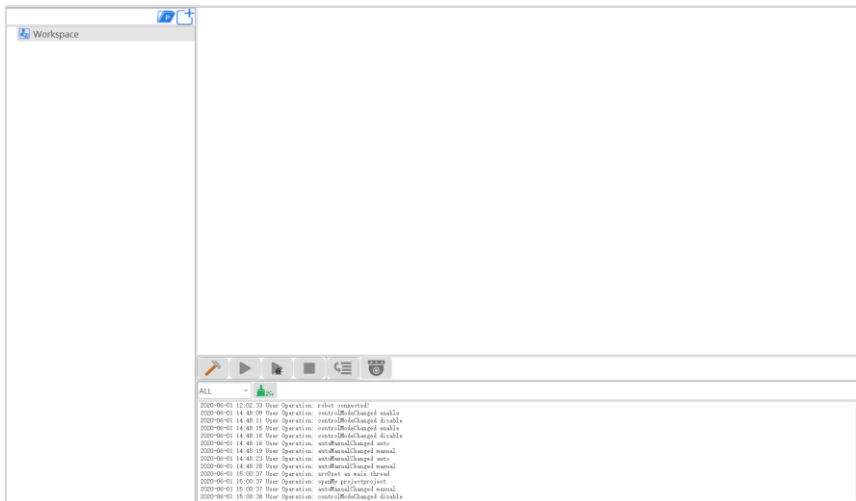



Figure 3.7 Programming page

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

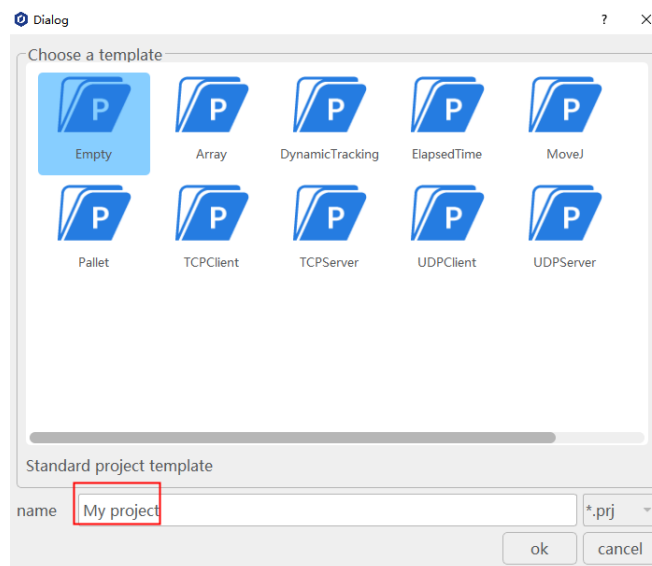


Figure 3.8 Create a project

**Step 3** Set the number of threads based on site requirements, as shown in Figure 3.9. Click **thread** and right-click **New thread file**.

The maximum number of threads is **5**.

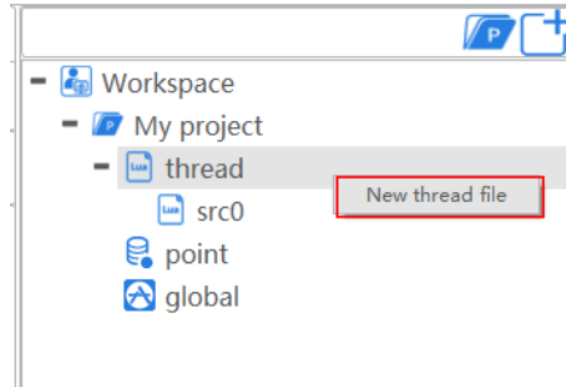


Figure 3.9 Create a project

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

If you want to **reuse a taught positions** list from an existing project, please right-click **Point** and click **import points file**, as shown in Figure 3.10.

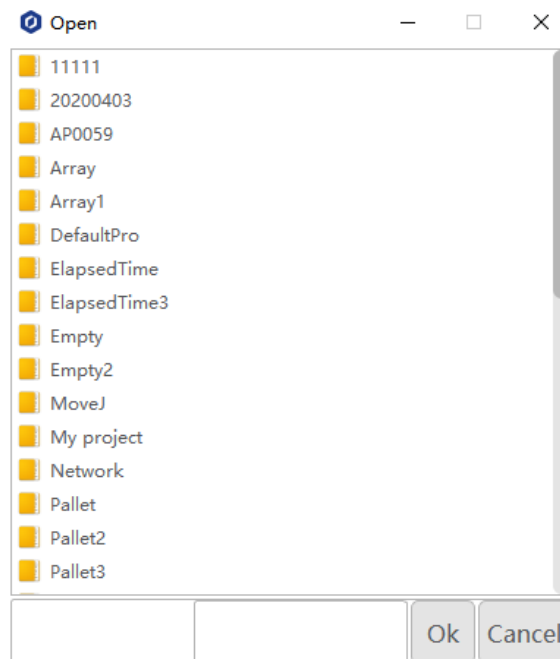


Figure 3.10 Import the existing teaching points list

### 3.5.3.2 Import Project

If you need to reuse project files of other M1 Pro, you can export project files of other M1 Pro to local computer and then import them into the current M1 Pro from local computer.



### Prerequisites

The robot has been powered on.

### Procedure

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

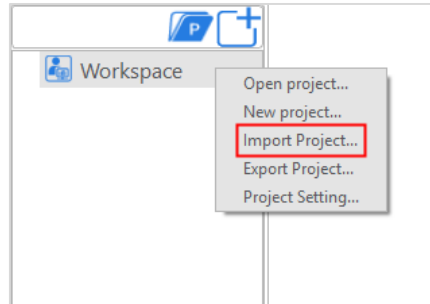


Figure 3.11 Import Project

**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**.

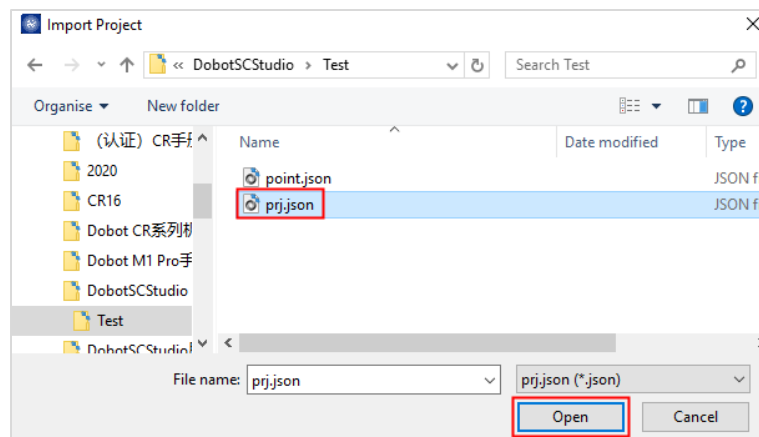


Figure 3.12 Select a project

**Step 3** Click **Open**.

The imported project files are displayed.

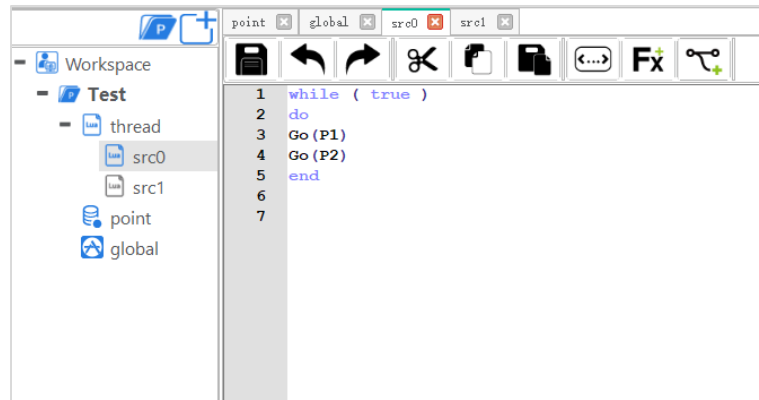
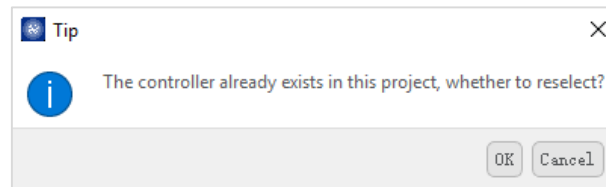


Figure 3.13 Display the project

### NOTE

If the project has already been saved or imported into M1 Pro, 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.




### 3.5.3.3 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 the M1 Pro.
- Step 2** Click **Jog** buttons to move the robot to a point.
- Step 3** Double click **Point** to enter point page and click  to add a teaching point.

The teaching point information is displayed on the **point** page, as shown in Figure 3.14.

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

No.	Alias	X	Y	Z	R	Arm	Tool	User	Load
1 P0		350.0000	50.0000	0.0000	0.0000	Right	- No.0	- No.0	- No.0
2 P1		328.1733	-23.3610	62.0629	2.8213	Right	- No.0	- No.0	- No.0
3 P2		306.5752	119.3947	62.0629	28.1713	Right	- No.0	- No.0	- No.0
4 P3		344.8234	23.5761	33.4501	2.8210	Right	- No.0	- No.0	- No.0
5 P4		231.4439	256.6957	33.4501	46.8710	Right	- No.0	- No.0	- No.0
6 P5		363.5784	-4.1726	-41.4634	44.8423	Right	- No.0	- No.0	- No.0
7 P6		332.1201	-4.3063	-40.7471	44.5819	Right	- No.0	- No.0	- No.0

Figure 3.14 Teaching points list

Table 3.4 Button description

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

- You can select a taught position and double-click the parameters on the line to modify the relevant information.
- Also, you can select a taught position and click to cover the current taught position.

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

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

### 3.5.3.4 Writing a Program

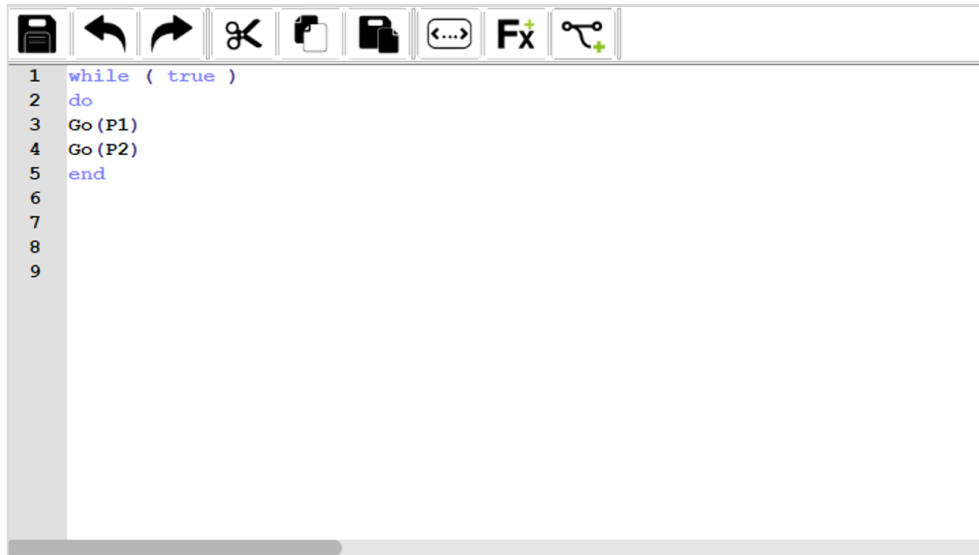
#### Prerequisites

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

### Procedure

In the M1 Pro, we have encapsulated common commands for programming with Lua language.


Supposing that the **P1** and **P2** points have been taught on the **point** page. We call **Go** command on the Src0 Page, to make the robot move between point P1 and point P2 circularly, as shown in Figure 3.15.



```

1  while ( true )
2  do
3  Go (P1)
4  Go (P2)
5  end
6
7
8
9
    
```

Figure 3.15 Lua program

**Step 1** Click  >Syntax, double click `while` to call the loop command, and set the loop condition to **True**.

**Step 2** Add the motion commands between **do** and **end**.

1. Click **Fx** > **Move**.

The motion commands list is displayed, as shown in Figure 3.16.

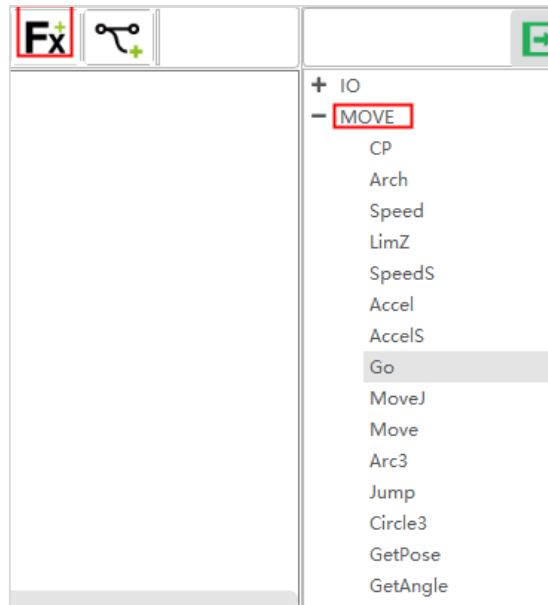


Figure 3.16 Motion commands list

2. Select a command from the motion commands list and click it on the edit window of the **Src0** page.

The parameter setting page of this command is displayed. Take the **Go** command as an example. You can set the point where the robot will move to in the **Go** mode.

3. Select **P1** on the **First Parameter** section of the Go command setting page, and then click **Insert**. Namely, the robot moves to **P1** point in the **Go** mode.

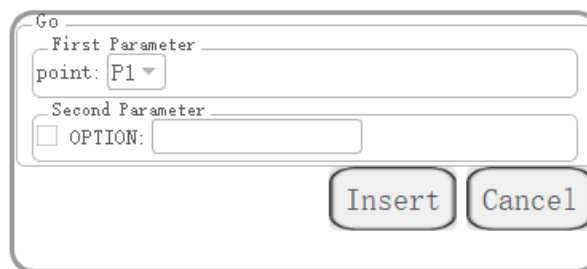


Figure 3.17 Call the **Go** command

If you want to set the motion speed, arm orientation, you can set them on the **Second Parameter** section, as shown in Figure 3.18.

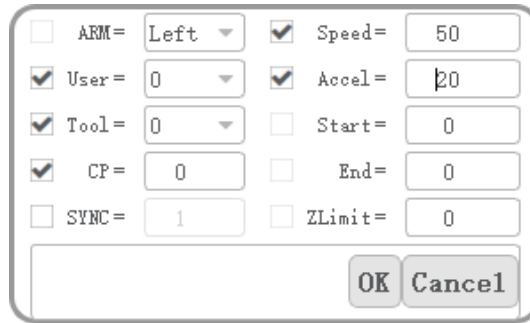


Figure 3.18 Set the optional parameters

4. Select **Go** command from the motion commands list.
5. Select **P2** on the **First Parameter** section of the **Go** command setting page, and then click **Insert**. Namely, the robot moves to **P2** point in the **Go** mode.

#### NOTE

If you want to debug a robot program step by step, please set the breakpoint when writing the program. Click the right line to set, as shown in Figure 3.19.



Figure 3.19 Set breakpoint

**Step 3** Click .

Now, a simple program has been written.

### 3.5.3.5 Debugging Program

**Step 1** Click  to enable the motor.

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

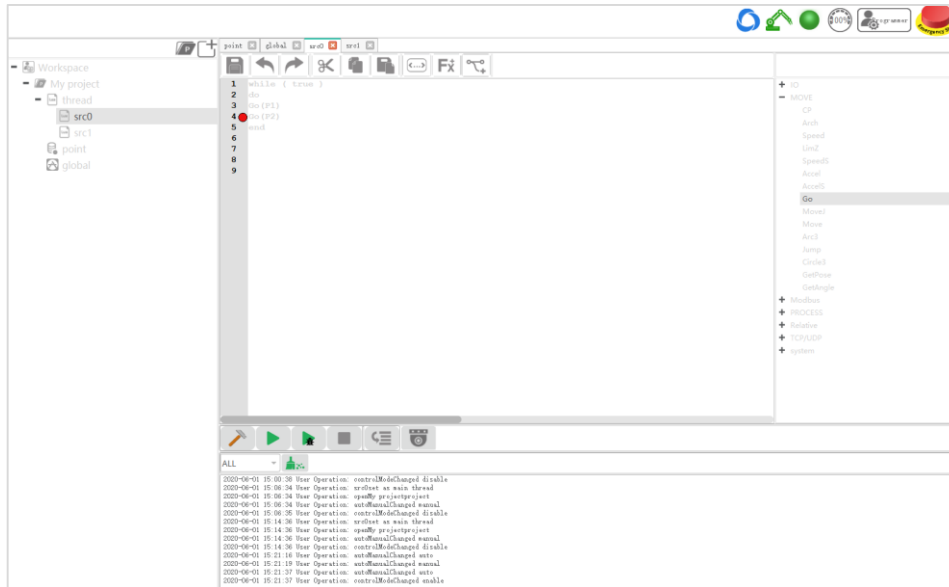






Figure 3.20 Programming page

Table 3.5 lists the description of the program-running buttons which are shown in Figure 3.20



Table 3.5 Program-running button description



Icon	Description
	Build program Check if the code is correct
	Once-click run After clicking this button,  turns into  and the program starts running If you need to pause the running program, please click
	Start to run a program Click once: Start to debug a program,  turns into Click twice: Start to run a program,  turns into If you need to pause the running program, please click
	Stop the running program
	Step into

Icon	Description
	This button is valid only if  turns into 
	Monitor The debugging process can be monitored in real time while debugging the program

**Step 2** Click  to start debugging the program.

- If you have set a breakpoint, the program will be run to the previous line of the breakpoint and then be stopped. If the program needs to be run again, please

click  and then click .

- If you want to run a program step by step, please click . After 

turns into , please click .

### 3.5.3.6 Export Project

Project is saved in M1 Pro by default, and you can import the project into local storage.

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

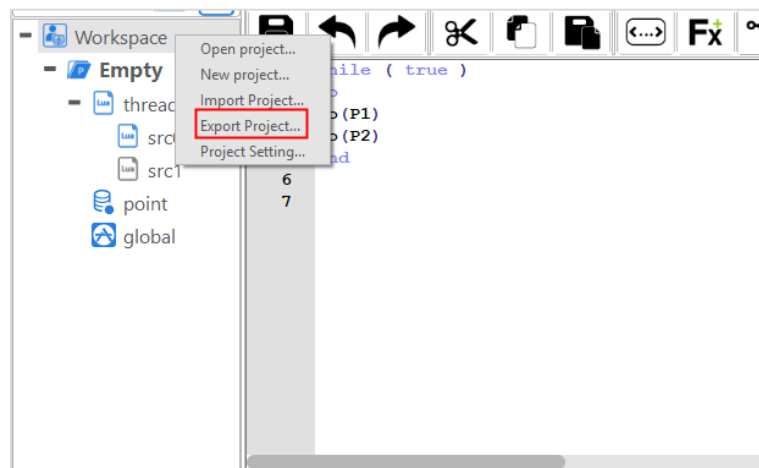


Figure 3.21 Export Project

**Step 2** In the project list, click the project to be exported.

For example, click "Factory Test 1".



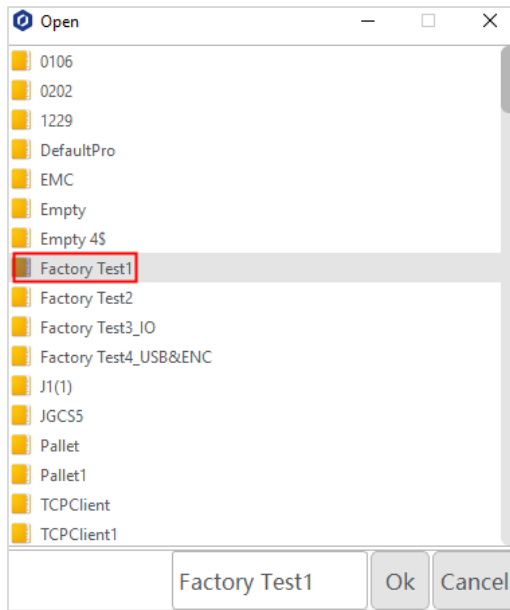


Figure 3.22 Select Project Files

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

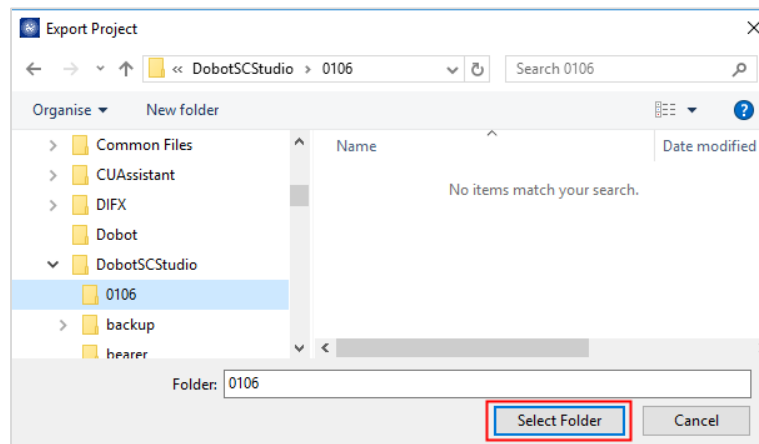


Figure 3.23 Save path

## 3.6 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.

### 3.6.1 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  $P_0(x_0, y_0, z_0)$ ,  $P_1(x_1, y_1, z_1)$ . Point  $P_0$  is defined as the origin and the line from point  $P_0$  to Point  $P_1$  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.

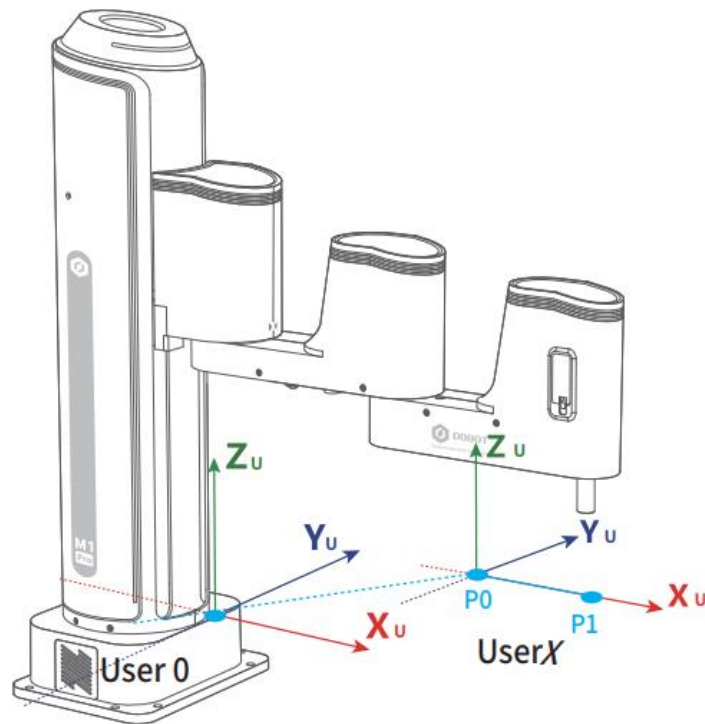


Figure 3.24 Two point calibration

Take the establishment of User 4 coordinate system as an example based on two-point calibration method.

### Prerequisites

- The robot has been powered on.
- The M1 Pro 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.

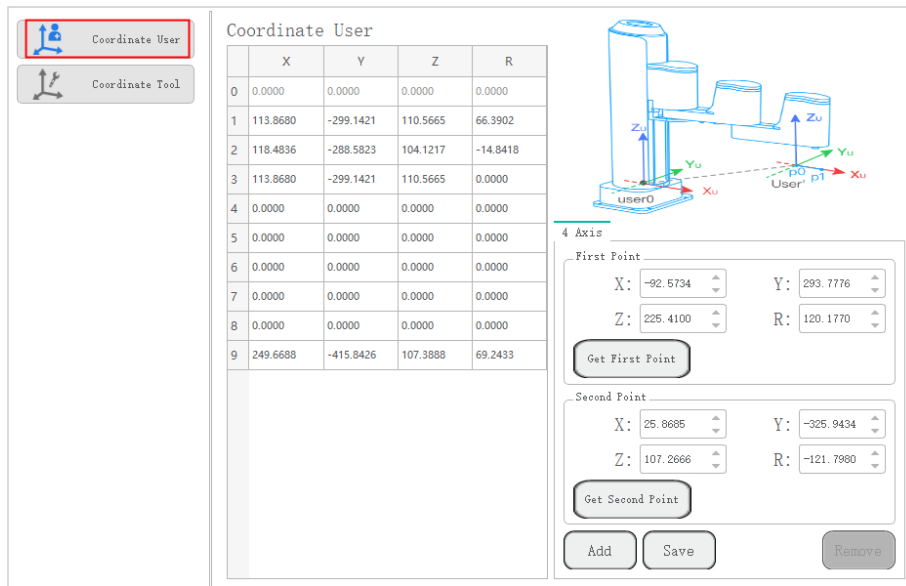


Figure 3.25 User coordinate system page

- 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 4 coordinate system.

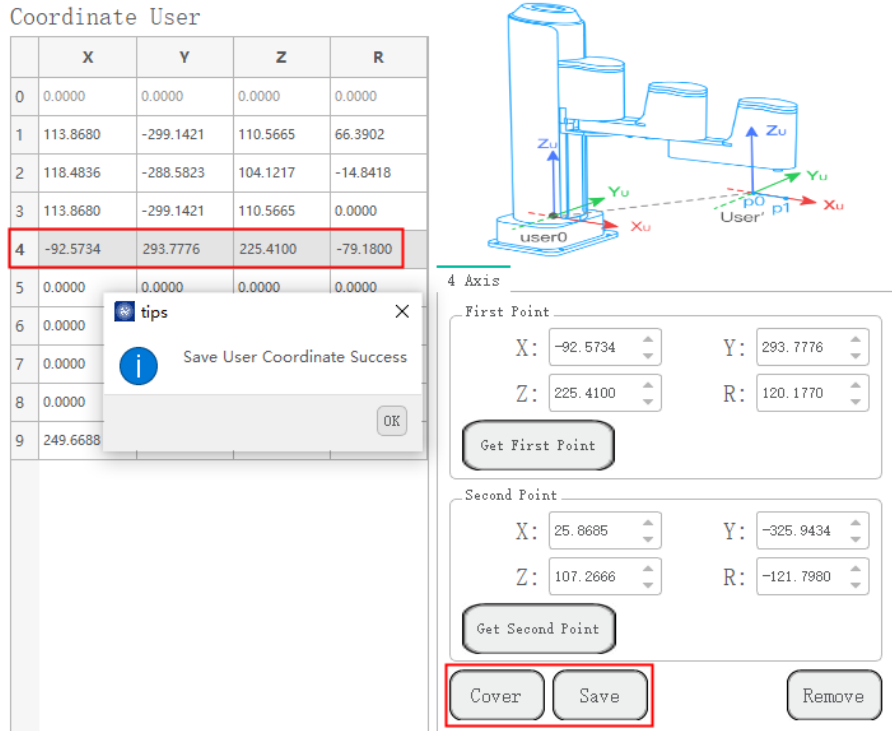


Figure 3.26 User 4 coordinate system

**Step 5** Select **User 4** on Jog interface.

You can use the **User 4** coordinate system for teaching and programming.

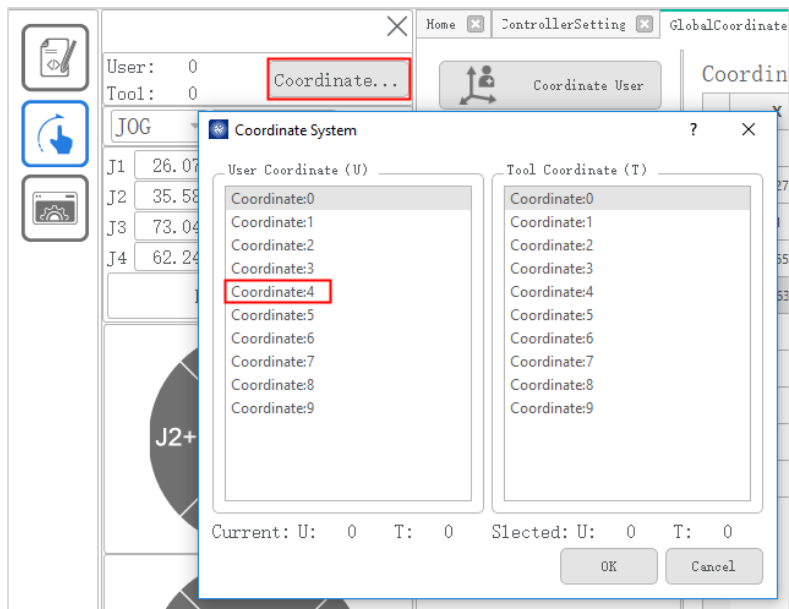


Figure 3.27 Select user coordinate system

### 3.6.2 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 3.28.

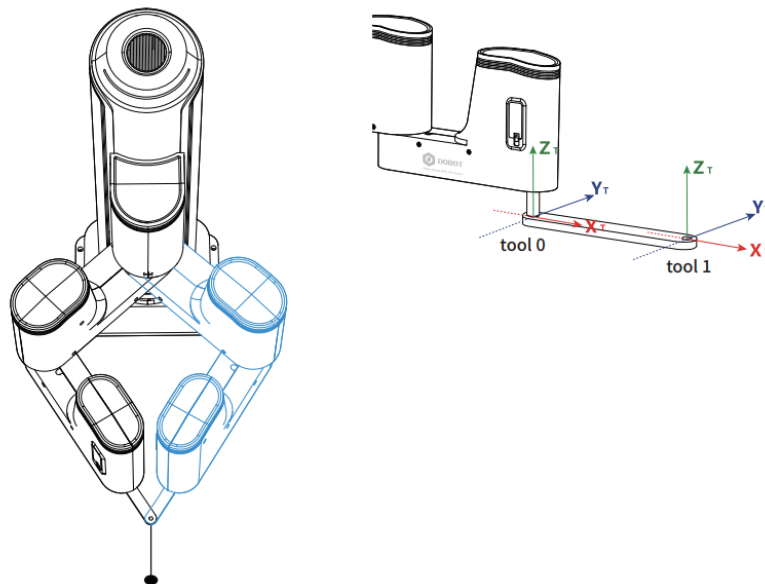


Figure 3.28 Three points calibration method (TCP+ZX)

Take the establishment of Tool 2 coordinate system as an example.

#### Prerequisites

- The robot has been powered on.
- The M1 Pro 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.

The end effector must be eccentrically. Otherwise, the tool coordinate system cannot be successful.

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

The Coordinate Tool page is displayed, as shown in Figure 3.29.

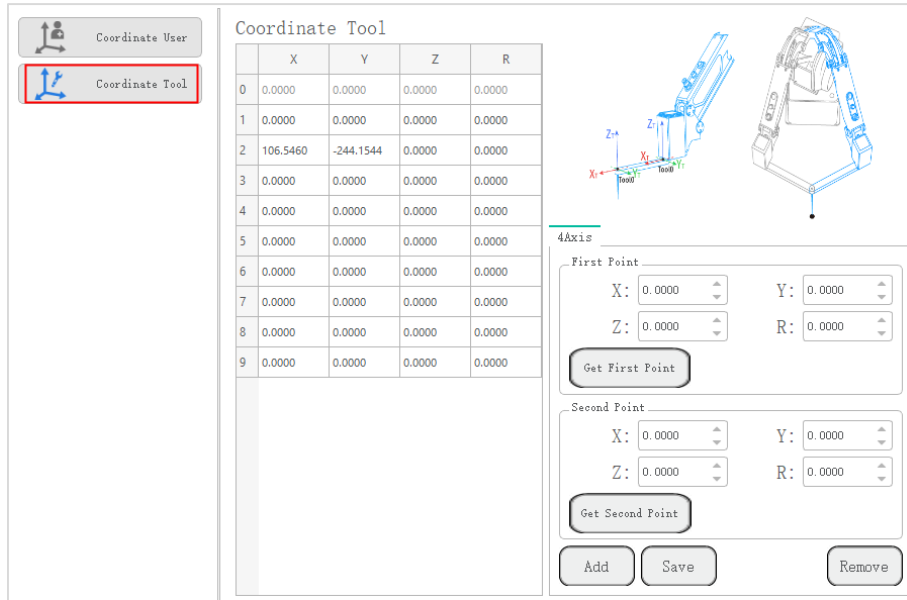


Figure 3.29 Tool Coordinate page

**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 **Cover** and **Save** to generate the Tool 2 coordinate system.

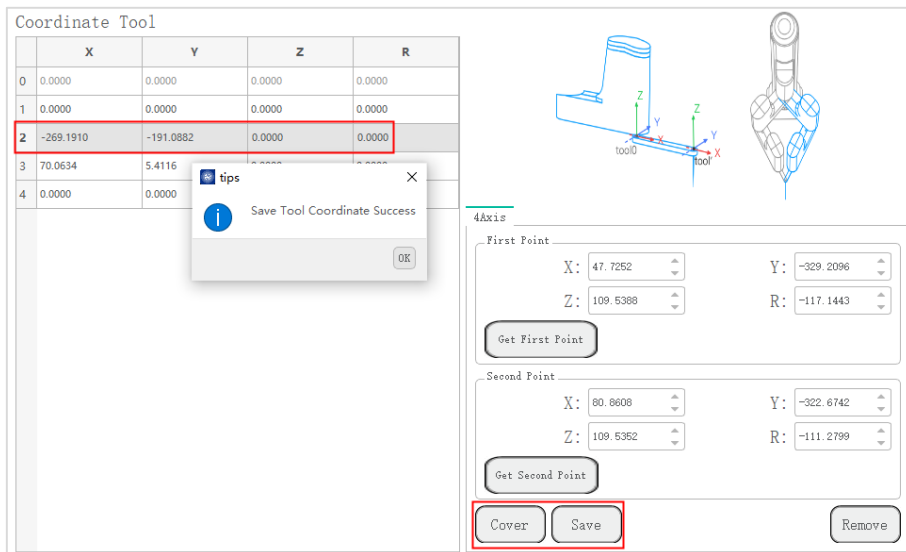



Figure 3.30 Tool 2 coordinate system

### 3.6.3 I/O Monitor

You can set or monitor the I/O status of the robot and robot on this page. Click  > **Parameter > IOMonitor** to enter the I/O monitor page, as shown in Figure 3.31.

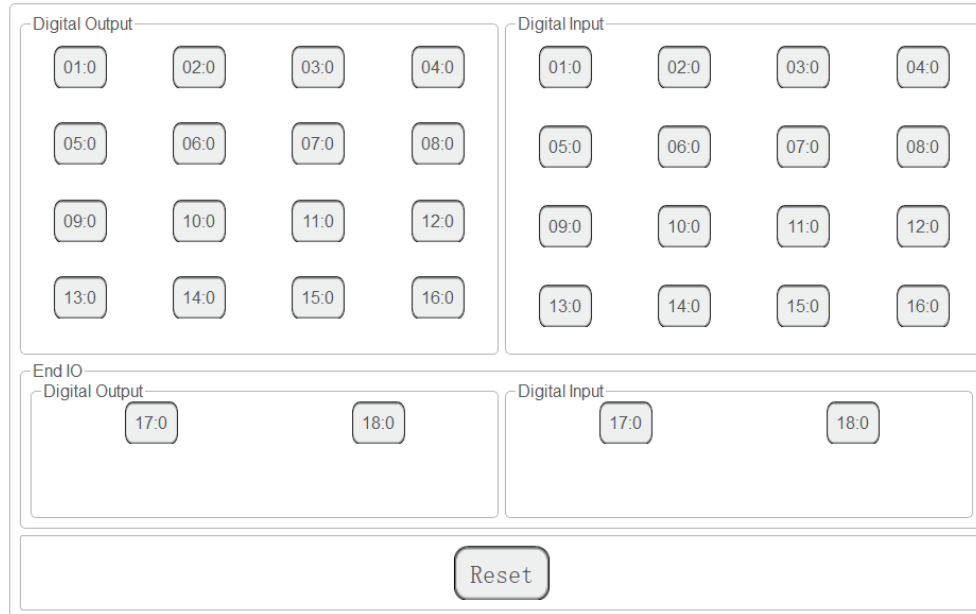


Figure 3.31 I/O monitor page

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 3.32.

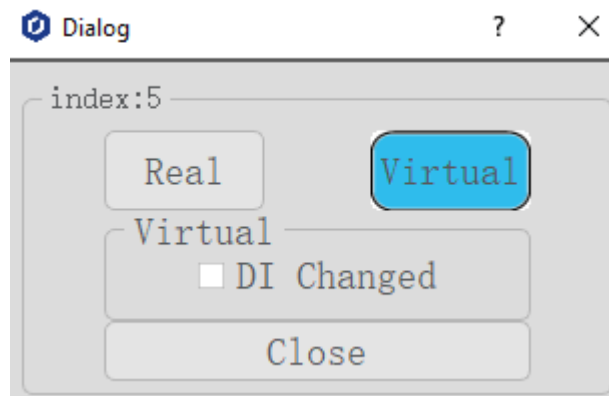



Figure 3.32 Simulation

For details about I/O interface, please see *Dobot M1 Pro Hardware User Guide*.

### 3.6.4 Controller Setting

#### 3.6.4.1 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.

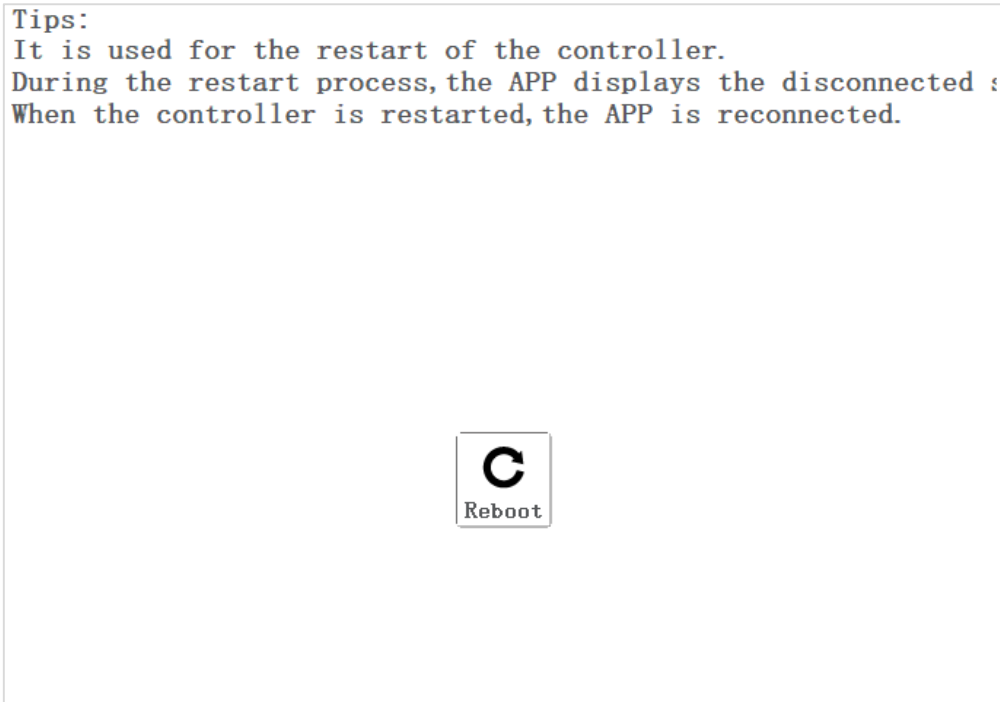


Figure 3.33 Reboot

#### 3.6.4.2 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.



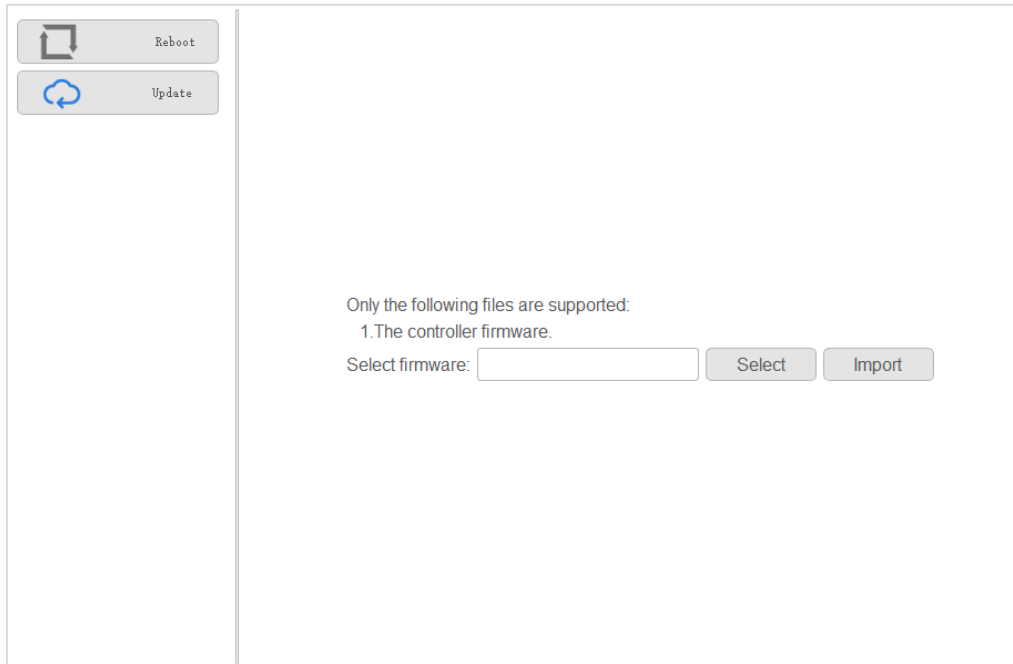


Figure 3.34 Update firmware

### 3.6.5 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 the M1 Pro 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 the M1 Pro is in.
- Please **DO NOT** switch the remote mode when the robot is running in the current remote mode. You need to exit the current mode and then switch to the other remote mode. Namely, please stop the robot running and then switch the mode.
- If the M1 Pro is in the enabled status, the remote control cannot be used. Otherwise, an alarm will be triggered. Please activate the remote control in the disabled status.

#### 3.6.5.1 Remote I/O

When the remote mode is I/O mode, external equipment can control a robot in this mode. The specific I/O interface descriptions are shown in Table 3.6.

Table 3.6 Specific I/O interface description

I/O interface	Description
Input (For external control)	

I/O interface	Description
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.


### Prerequisites

- The project to be running in the remote mode has been prepared.
- The external equipment has been connected to the M1 Pro by the I/O interface. The specific I/O interface description is shown in Table 3.6.
- The robot has been powered on and in the disabled status.

### NOTE

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

### Procedure

**Step 1** Click  > **Parameter > Offline.**

The remote control page is displayed, as shown in Figure 3.35.

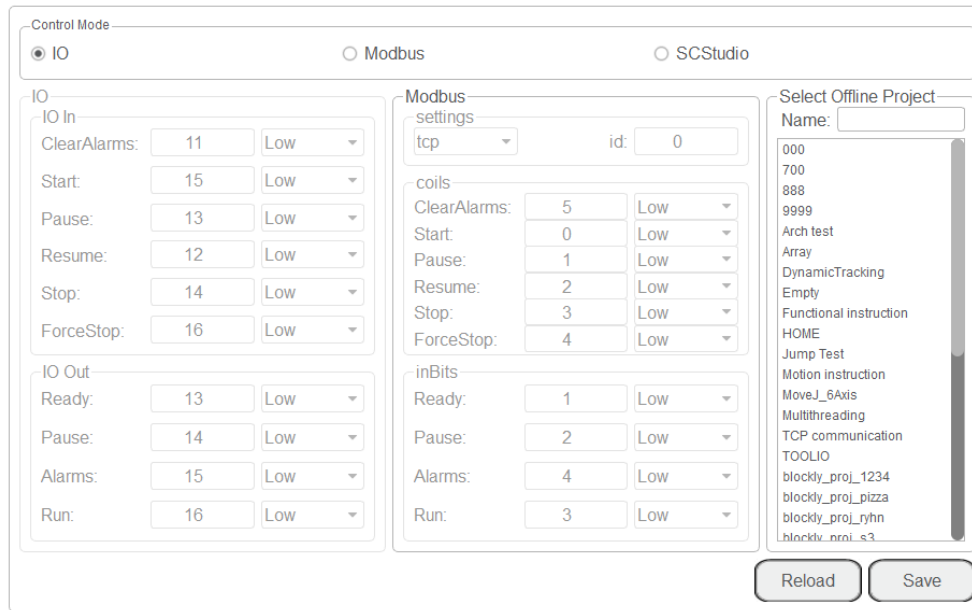


Figure 3.35 Remote control page

**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 remote I/O mode will be invalid.

### 3.6.5.2 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 Modbus Register Description*. The specific Modbus register descriptions are shown in Table 3.7.

Table 3.7 Specific Modbus register description

Register address (Take a PLC as an example)	Register address (M1 Pro)	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

Register address (Take a PLC as an example)	Register address (M1 Pro)	Description
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 the M1 Pro 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 and in the disabled status.

#### NOTE

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

### Procedure

**Step 1** Click  > **Parameter > Offline**.

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

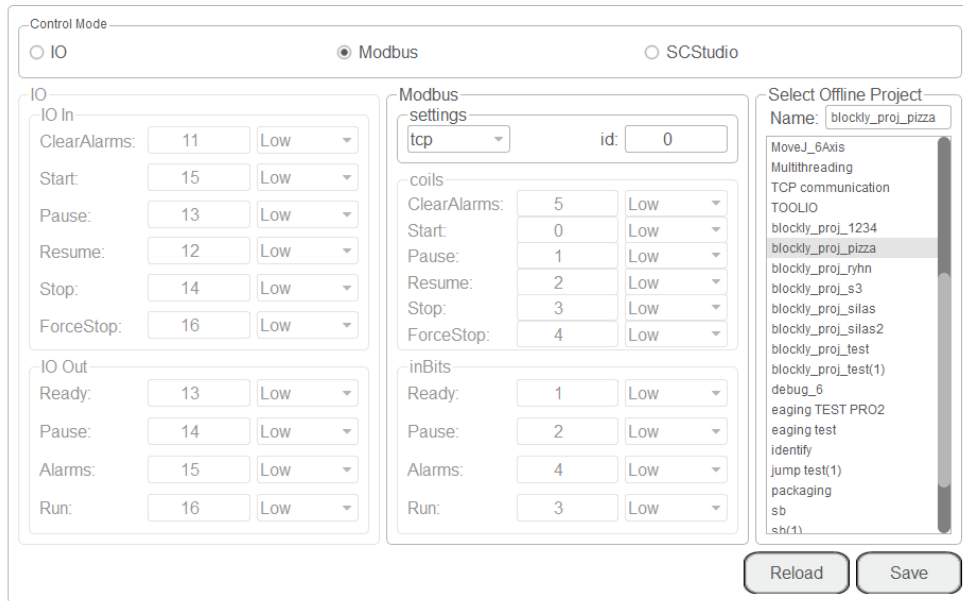


Figure 3.36 Remote control page

**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.

### 3.6.6 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 3.37.

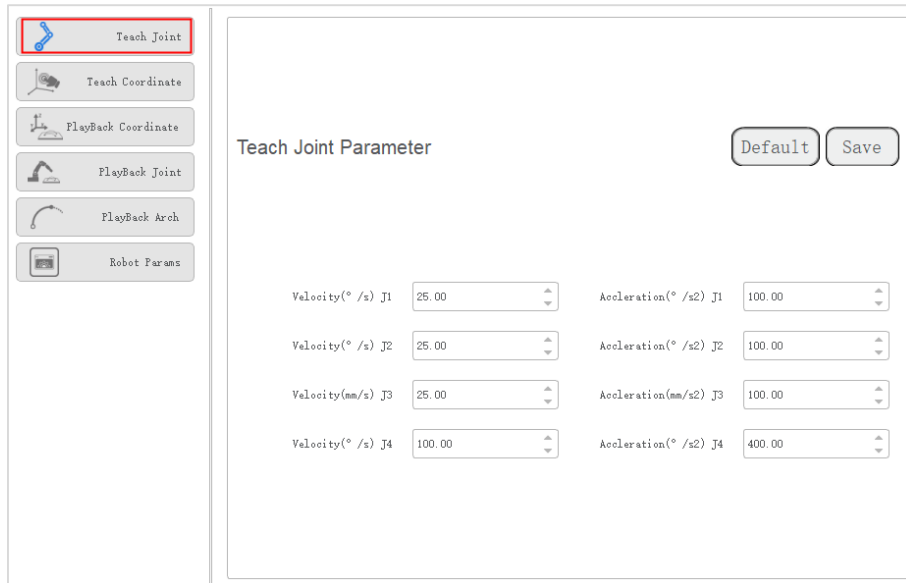


Figure 3.37 Jogging parameters in the Joint coordinate system

- 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 3.38.

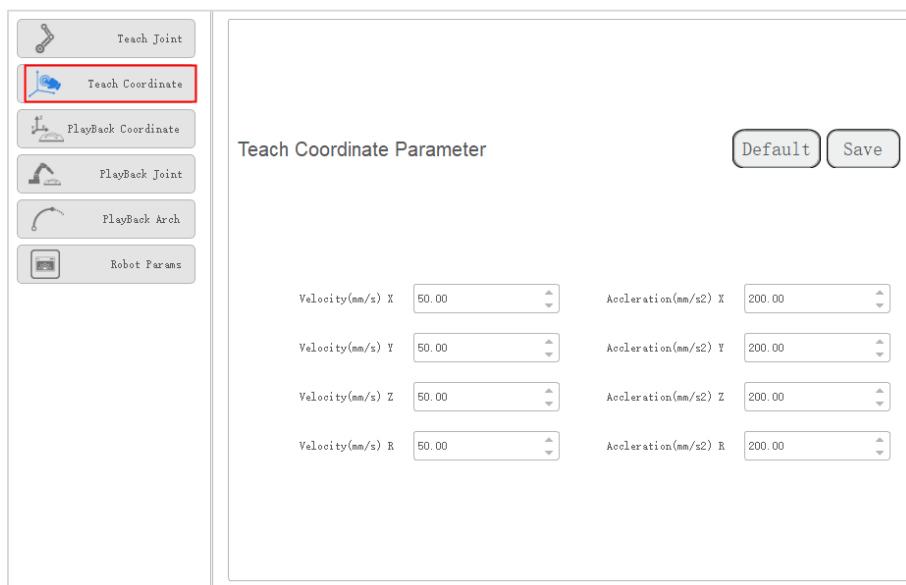


Figure 3.38 Jogging parameters in the Cartesian coordinate system

- **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 3.39.

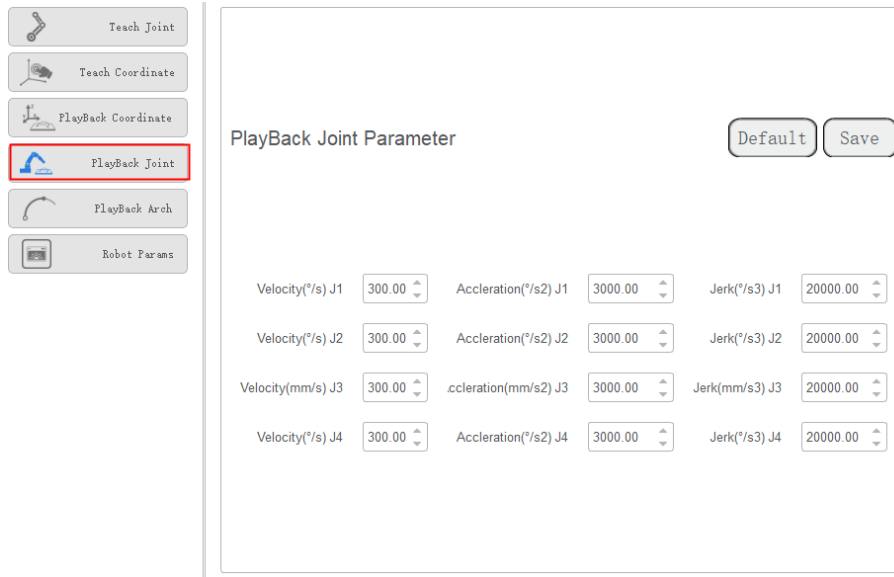


Figure 3.39 Playback parameters in the Joint coordinate system

- 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.

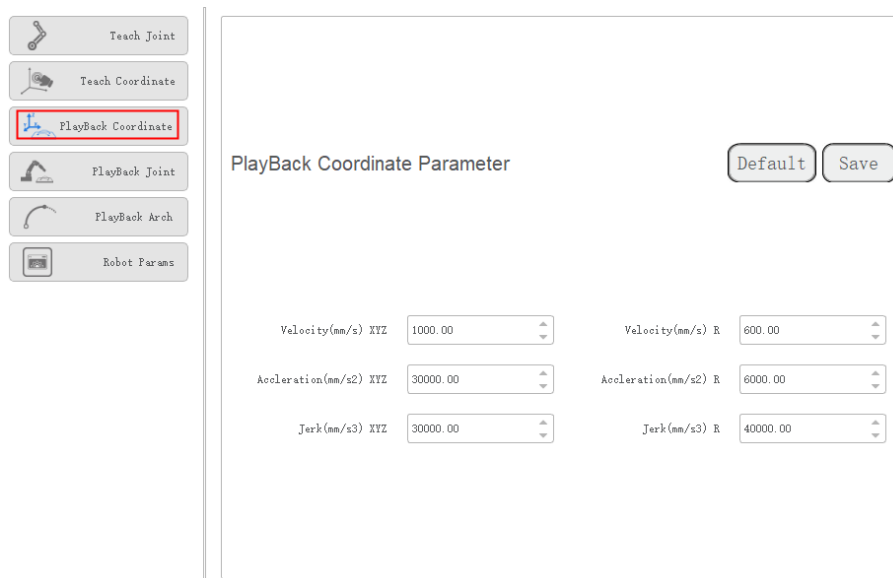


Figure 3.40 Playback parameters in the Cartesian coordinate system

- 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 3.41.

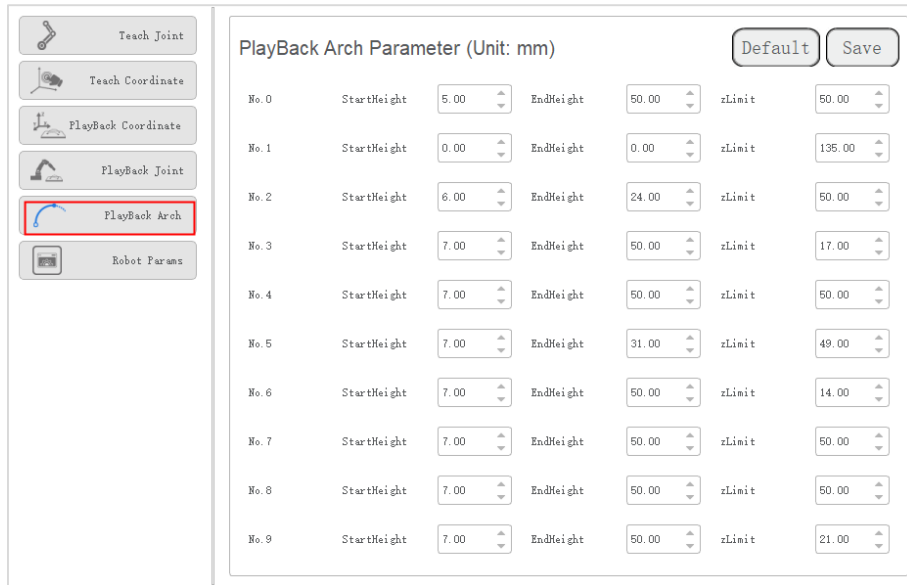


Figure 3.41 Jump parameters

- **Robot Params:** This function is only used for Dobot support engineer.

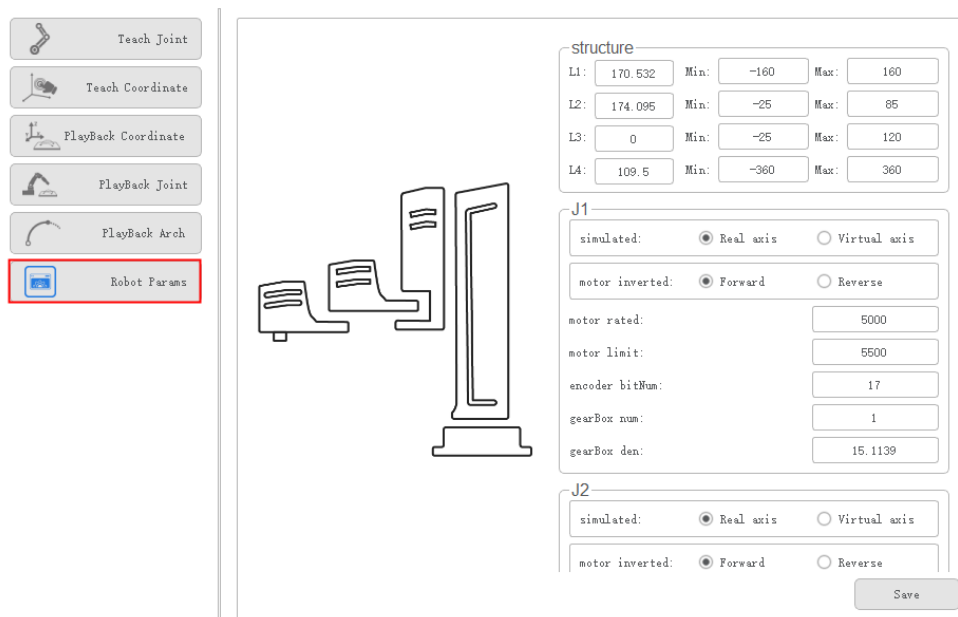


Figure 3.42 Robot Params

### 3.6.7 RobotSetting

#### 3.6.7.1 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 the position of the M1 Pro according to the prompts, adjust each axis to the mechanical Homing point.

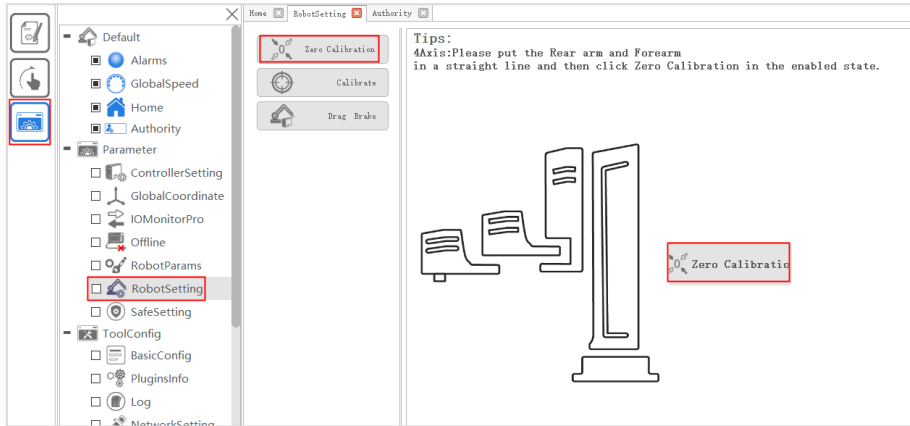
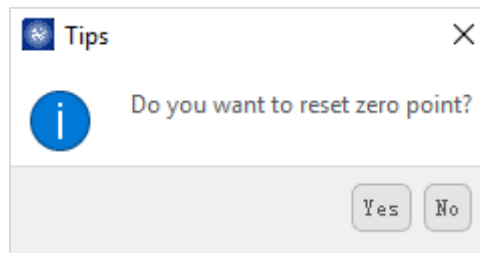


Figure 3.43 Zero Calibration

- Step 2** Click the **Zero Calibration**.
- Step 3** Click **Yes** in the current prompt window.



### 3.6.7.2 Calibrate

In real applications, the high precision of J2 is required. Therefore, after resetting the origin, you need to calibrate J2 to improve the absolute precision. Generally, before and after switching the arm orientation at the same point, the J2 coordinates are axisymmetric.

- Step 1** Click > **Parameter** > **RobotSetting** > **Calibrate** to enter **Calibrate** interface, as shown in Figure 3.44.

Select a reference point in the workspace of the robot.

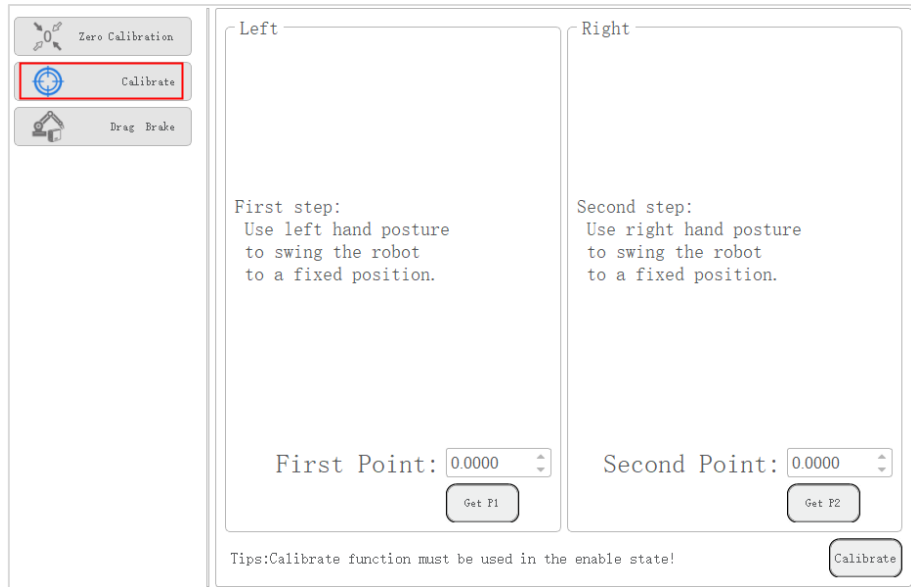



Figure 3.44 Calibrate

- Step 2** Jog the robot to the reference point in the lefty hand orientation, then click **Get P1** to obtain the angle of the J2.
- Step 3** Jog the robot to the reference point in the righty hand orientation, then click **Get P2** to obtain the angle of the J2.
- Step 4** Click **Calibrate**.

### 3.6.7.3 Drag and Brake

M1 Pro J1~J3 axis support manual drag, click  > **Parameter** > **RobotSetting** > **Calibrate** to enter **Drag and Brake** interface, as shown in Figure 3.45.

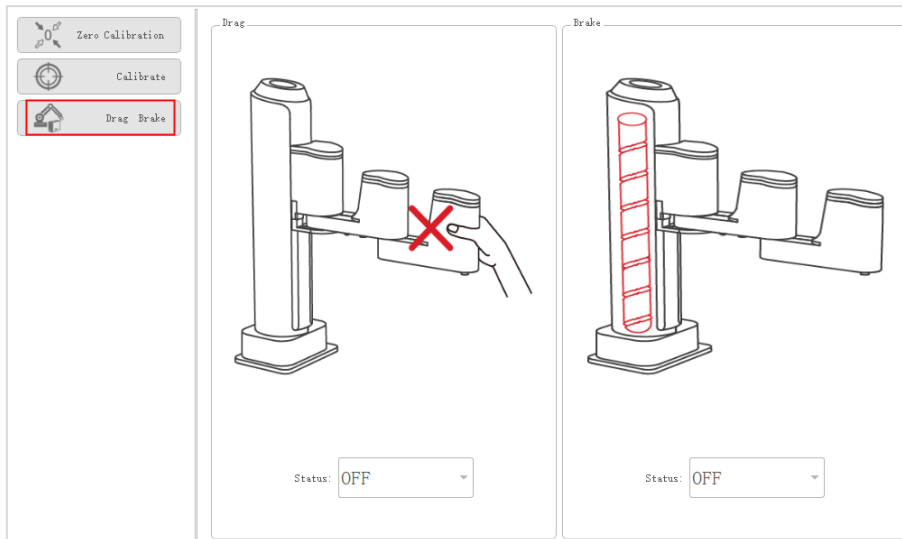


Figure 3.45 Drag and brake

- Drag: Set the **States** of the **Drag** page to **ON**, then J1~J2 axis can be moved left and right manually.
- Brake: the **States** of the **Brake** page to **ON**, then J3 axis can be moved vertically.

## 3.7 ToolConfig

### 3.7.1 BasicConfig

User can view versions of software, controller, algorithm on the **ToolConfig > BasicConfig > Version** page and Select language on the **ToolConfig > BasicConfig > Language** page. Also, you can modify the password on the **ToolConfig > BasicConfig > UserMode** page.

### 3.7.2 PlugInfo

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

### 3.7.3 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.

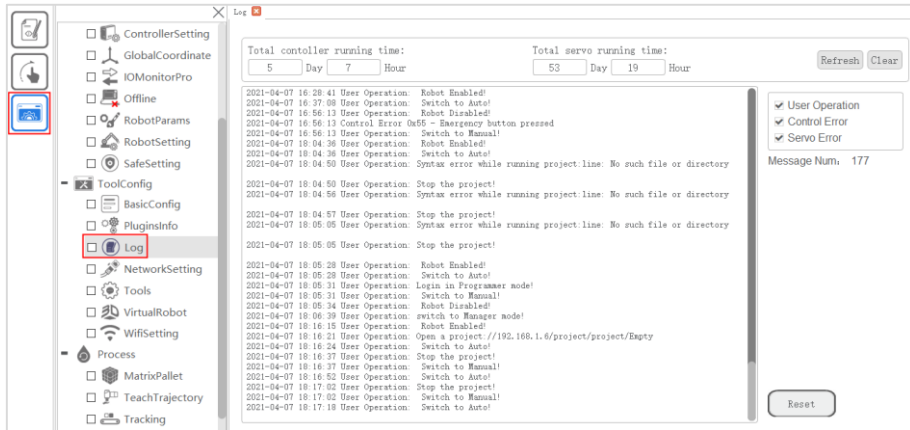


Figure 3.46 Log

### 3.7.4 Network Service

The M1 Pro 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, the M1 Pro can be a client or a server based on site requirements; if the Modbus protocol is used, the M1 Pro 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 3.47. The IP address of the M1 Pro must be in the same network segment of the external equipment without conflict.

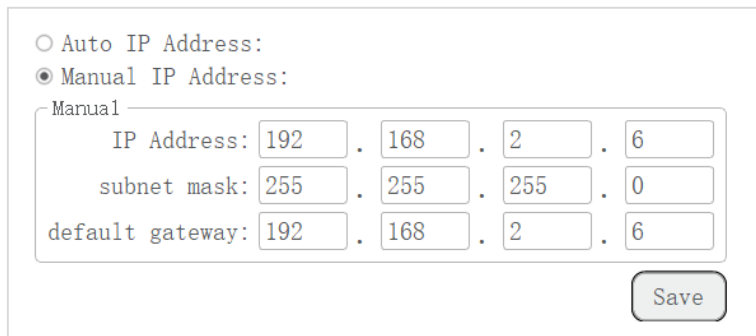


Figure 3.47 IP address setting

If the M1 Pro 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.

### 3.7.5 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.

### 3.7.6 VirtualRobot

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

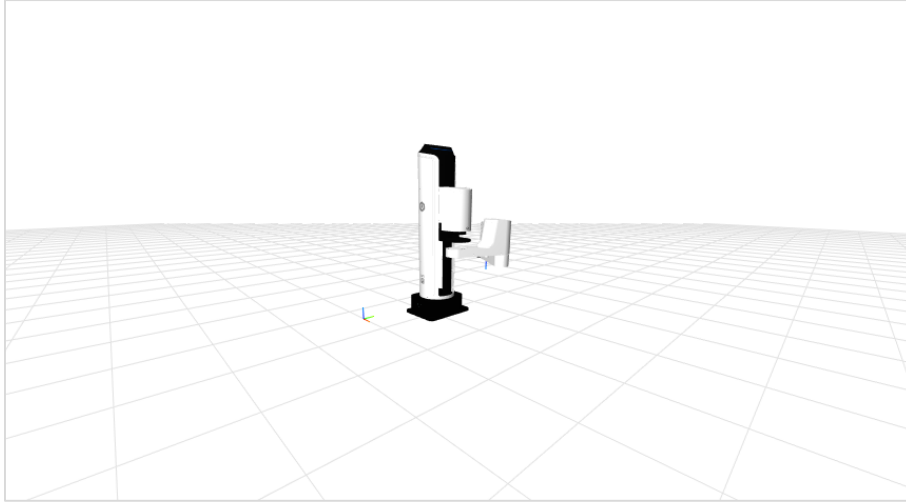


Figure 3.48 Virtual simulation

## 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 <b>table</b>

## 4.6 Processing Control Commands

Table 4.6 Processing control command

Command	Description
if...then...else...elseif...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 command

Command	Description
Go	Move from the current position to a target position in a point-to-point mode under the Cartesian coordinate system
MoveJ	Move from the current position to a target position in a point-to-point motion under the Joint coordinate system
Move	Move from the current position to a target position in a straight line under the Cartesian coordinate system
Arc3	Move from the current position to a target position in an arc interpolated mode under the Cartesian coordinate system
Jump	The robot moves from the current position to a target position in the <b>Move</b> mode. The trajectory looks like a door
Circle3	Move from the current position to a target position in a circular interpolated mode under the Cartesian coordinate system
RP	Set the X, Y, Z, R axes offset under the Cartesian coordinate system to return a new Cartesian coordinate point



Command	Description
RJ	Set the joint offset under the Joint coordinate system to return a new joint coordinate point
MoveR	Move from the current position to the offset position in a straight line under the Cartesian coordinate system
GoR	Move from the current position to the offset position in a point-to-point mode under the Cartesian coordinate system
MoveJR	Move from the current position to the offset position in a point-to-point motion under the Joint coordinate system

 NOTICE

Optional parameters for each motion command can be set individually

Table 4.8 Go command

Function	Go(P,"User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1")
Description	Move from the current position to a target position in a point-to-point mode under the Cartesian coordinate system
Parameter	<p>Required parameter: P: Indicate target point, which is user-defined or obtained from the <b>point</b> page. Only Cartesian coordinate points are supported</p> <p>Optional parameter:</p> <ul style="list-style-type: none"> <li>• User: Indicate User coordinate system. Value range: 0 - 9</li> <li>• Tool: Indicate Tool coordinate system. Value range: 0-9</li> <li>• CP: Whether to set continuous path function. Value range: 0- 100</li> <li>• Speed: Velocity rate. Value range: 1 - 100</li> <li>• Accel: Acceleration rate. Value range: 1 -100</li> <li>• SYNC: Synchronization flag. Value range: 0 or 1. If <b>SYNC</b> is <b>0</b>, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If <b>SYNC</b> is <b>1</b>, it indicates synchronous execution. After calling this command, it will not return until it is executed completely</li> </ul>
Example	<p>The robot moves to point P1 as the default setting</p> <pre>Go(P1)</pre>

Table 4.9 MoveJ command

Function	MoveJ(P," CP=1 Speed=50 Accel=20 SYNC=1")
Description	Move from the current position to a target position in a point-to-point motion under the Joint coordinate system
Parameter	<p>Required parameter: P: Indicate the joint angle of the target point, which cannot be obtained from the <b>point</b> page. You need to define the joint coordinate point before calling this command</p> <p>Optional parameter:</p> <ul style="list-style-type: none"> <li>• CP: Whether to set continuous path function. Value range: 0 - 100</li> <li>• Speed: Velocity rate. Value range: 1 - 100</li> <li>• Accel: Acceleration rate. Value range: 1 - 100</li> <li>• SYNC: Synchronization flag. Value range: 0 or 1. If <b>SYNC</b> is <b>0</b>, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If <b>SYNC</b> is <b>1</b>, it indicates synchronous execution. After calling this command, it will not return until it is executed completely</li> </ul>
Example	<pre>local P = {joint={0,-0.0674194,0,0}} MoveJ(P)</pre>

Table 4.10 Move command

Function	Move(P,"User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
Description	Move from the current position to a target position in a straight line under the Cartesian coordinate system
Parameter	<p>Required parameter: P: Indicate the target point, which is user-defined or obtained from the <b>point</b> page. Only Cartesian coordinate points are supported</p> <p>Optional parameter:</p> <ul style="list-style-type: none"> <li>• User: Indicate User coordinate system. Value range: 0 - 9</li> <li>• Tool: Indicate Tool coordinate system. Value range: 0 - 9</li> <li>• CP: Whether to set continuous path function. Value range: 0 - 100</li> <li>• SpeedS: Velocity rate. Value range: 1 - 100</li> <li>• AccelS: Acceleration rate. Value range: 1 - 100</li> <li>• SYNC: Synchronization flag. Value range: 0 or 1. If <b>SYNC</b> is <b>0</b>, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If <b>SYNC</b> is <b>1</b>, it indicates synchronous execution. After calling this command, it will not return until it is executed completely</li> </ul>
Example	<p>The robot moves to point P1 as the default setting</p> <pre>Move(P1)</pre>

Table 4.11 Arc3 command

Function	Arc3(P1,P2, "User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
Description	<p>Move from the current position to a target position in an arc 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"> <li>• P1: Middle point, which is user-defined or obtained from the <b>point</b> page. Only Cartesian coordinate points are supported</li> <li>• P2: End point, which is user-defined or obtained from the <b>point</b> page. Only Cartesian coordinate points are supported</li> </ul> <p>Optional parameter:</p> <ul style="list-style-type: none"> <li>• User: Indicate User coordinate system. Value range: 0 - 9</li> <li>• Tool: Indicate Tool coordinate system. Value range: 0 - 9</li> <li>• CP: Whether to set continuous path function. Value range: 0 - 100</li> <li>• SpeedS: Velocity rate. Value range: 1 - 100</li> <li>• AccelS: Acceleration rate. Value range: 1 - 100</li> <li>• SYNC: Synchronization flag. Value range: 0 or 1. If <b>SYNC</b> is <b>0</b>, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If <b>SYNC</b> is <b>1</b>, it indicates synchronous execution. After calling this command, it will not return until it is executed completely</li> </ul>
Example	<pre>While true do     Go(P1)     Arc3(P2,P3) end</pre> <p>The robot cycles from point P1 to point P3 in the arc interpolated mode</p>

Table 4.12 Jump command

Function	Jump(P,"User=1 Tool=2 Speed=50 Accel=20 Start=10 ZLimit=80 End=50 SYNC=1")
Description	The robot moves from the current position to a target position in the <b>Move</b> mode. The trajectory looks like a door.
Parameter	<p>Required parameter: P: Indicate the target point, which is user-defined or obtained from the <b>point</b> page. Only Cartesian coordinate points are supported. Also, the target point cannot be higher than <b>ZLimit</b>, to avoid an alarm about JUMP parameter error</p> <p>Optional parameter:</p>

	<ul style="list-style-type: none"> <li>• User: Indicate User coordinate system. Value range: 0 - 9</li> <li>• Tool: Indicate Tool coordinate system. Value range: 0 - 9</li> <li>• Speed: Velocity rate. Value range: 1 - 100</li> <li>• Accel: Acceleration rate. Value range: 1 - 100</li> <li>• Arch: Arch index. Value range: 0 - 9</li> <li>• Start: Lifting height</li> <li>• ZLimit: Maximum lifting height</li> <li>• End: Dropping height</li> <li>• SYNC: Synchronization flag. Value range: 0 or 1. If <b>SYNC</b> is <b>0</b>, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If <b>SYNC</b> is <b>1</b>, it indicates synchronous execution. After calling this command, it will not return until it is executed completely</li> </ul>
Example	<p>Jump(P1)</p> <p>The robot moves to point P1 in the Jump mode</p>

Table 4.13 Circle3 command

Function	Circle3(P1,P2,Count, "User=1 Tool=2 CP=1SpeedS=50 AccelS=20")
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"> <li>• P1: Middle point, which is user-defined or obtained from the <b>point</b> page. Only Cartesian coordinate points are supported</li> <li>• P2: End point, which is user-defined or obtained from the <b>point</b> page. Only Cartesian coordinate points are supported</li> <li>• Count: Number of circles. Value range: 1 - 999</li> </ul> <p>Optional parameter:</p> <ul style="list-style-type: none"> <li>• User: Indicate User coordinate system. Value range: 0 - 9</li> <li>• Tool: Indicate Tool coordinate system. Value range: 0 - 9</li> <li>• CP: Whether to set continuous path function. Value range: 0 - 100</li> <li>• SpeedS: Velocity rate. Value range: 1 - 100</li> <li>• AccelS: Acceleration rate. Value range: 1 - 100</li> <li>• SYNC: Synchronization flag. Value range: 0 or 1. If <b>SYNC</b> is <b>0</b>, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If <b>SYNC</b> is <b>1</b>, it indicates synchronous execution. After calling this command, it will not return until it is executed completely</li> </ul>

Example	Go(P1) Circle3(P2,P3,1) Robot cycles from point P1 to point P3 in the circular interpolated mode
---------	--

Table 4.14 RP command

Function	RP(P1, {OffsetX, OffsetY, OffsetZ, OffsetR})
Description	Set the X, Y, Z 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 MoveJ
Parameter	<ul style="list-style-type: none"> <li>P1: Indicate the current Cartesian coordinate point, which is user-defined or obtained from the <b>point</b> page. Only Cartesian coordinate points are supported</li> <li>OffsetX, OffsetY, OffsetZ, OffsetR: X, Y, Z, R axes offset in the Cartesian coordinate system Unit: mm</li> </ul>
Return	Cartesian coordinate point
Example	P2=RP(P1, {50,10,32,30}) Move(P2) or Move(RP(P1, {50,10,32,30}))

Table 4.15 RJ command

Function	RJ(P1, {Offset1, Offset2, Offset3, Offset4 })
Description	Set the joint offset in the Joint coordinate system to return a new joint coordinate point The robot can move to this point only in <b>MoveJ</b> command
Parameter	<ul style="list-style-type: none"> <li>P1: Indicate the current joint coordinate point, which cannot be obtained from the <b>point</b> page. You need to define the joint coordinate point before calling this command</li> <li>Offset1~Offset4: J1 – J4 axes offset. Unit: °</li> </ul>
Return	Joint coordinate point
Example	local P1 = {joint={0,-0.0674194,0,0}} P2=RJ(P1, {60,50,32,30}) MoveJ(P2) or MoveJ(RJ(P1, {60,50,32,30}))

Table 4.16 GoR command

Function	GoR({OffsetX, OffsetY, OffsetZ, OffsetR }, "User=1 Tool=2 CP=1 Speed=50 Accel=20 SYNC=1 ")
Description	Move from the current position to the offset position in a point-to-point mode under the Cartesian coordinate system
Parameter	<p>Required parameter: OffsetX, OffsetY, OffsetZ, OffsetR: X, Y, Z, R axes offset in the Cartesian coordinate system</p> <p>Unit: mm</p> <p>Optional parameter:</p> <ul style="list-style-type: none"> <li>• User: Indicate User coordinate system. Value range: 0 - 9</li> <li>• Tool: Indicate Tool coordinate system. Value range: 0-9</li> <li>• CP: Whether to set continuous path function. Value range: 0- 100</li> <li>• Speed: Velocity rate. Value range: 1 - 100</li> <li>• Accel: Acceleration rate. Value range: 1 -100</li> <li>• SYNC: Synchronization flag. Value range: 0 or 1. If <b>SYNC</b> is <b>0</b>, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If <b>SYNC</b> is <b>1</b>, it indicates synchronous execution. After calling this command, it will not return until it is executed completely</li> </ul>
Example	<p>Go(P1)</p> <p>GoR({10,10,10,30}, "Accel=100 Speed=100 CP=100")</p>

Table 4.17 MoveJR command

Function	MoveJR({Offset1, Offset2, Offset3, Offset4 }, " CP=1 Speed=50 Accel=20 SYNC=1")
Description	Move from the current position to the offset position in a point-to-point motion under the Joint coordinate system
Parameter	<p>Required parameter: Offset1 – Offset4: J1 – J4 axes offset.</p> <p>Unit: °</p> <p>Optional parameter:</p> <ul style="list-style-type: none"> <li>• CP: Whether to set continuous path function. Value range: 0 - 100</li> <li>• Speed: Velocity rate. Value range: 1 - 100</li> <li>• Accel: Acceleration rate. Value range: 1 - 100</li> <li>• SYNC: Synchronization flag. Value range: 0 or 1. If <b>SYNC</b> is <b>0</b>, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If <b>SYNC</b> is <b>1</b>, it indicates synchronous execution. After calling this command, it will not return until it is executed completely</li> </ul>
Example	Go(P1)

	MoveJR({20,20,10,0},"SYNC=1")
--	-------------------------------

Table 4.18 MoveR command

Function	MoveR({OffsetX, OffsetY, OffsetZ, OffsetR }, "User=1 Tool=2 CP=1 SpeedS=50 AccelS=20 SYNC=1")
Description	Move from the current position to the offset position in a straight line under the Cartesian coordinate system
Parameter	<p>Required parameter: OffsetX, OffsetY, OffsetZ, OffsetR: X, Y, Z, R axes offset in the Cartesian coordinate system</p> <p>Unit: mm</p> <p>Optional parameter:</p> <ul style="list-style-type: none"> <li>• User: Indicate User coordinate system. Value range: 0 - 9</li> <li>• Tool: Indicate Tool coordinate system. Value range: 0 - 9</li> <li>• CP: Whether to set continuous path function. Value range: 0 - 100</li> <li>• SpeedS: Velocity rate. Value range: 1 - 100</li> <li>• AccelS: Acceleration rate. Value range: 1 - 100</li> <li>• SYNC: Synchronization flag. Value range: 0 or 1. If <b>SYNC</b> is <b>0</b>, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If <b>SYNC</b> is <b>1</b>, it indicates synchronous execution. After calling this command, it will not return until it is executed completely</li> </ul>
Example	Go(P1) MoveR({20,20,10,0},"AccelS=100 SpeedS=100 CP=100")

## 4.9 Motion Parameter Commands

Table 4.19 Motion parameter command

Command	Description
Accel	Set the acceleration rate. This command is valid only when the motion mode is <b>Go</b> , <b>Jump</b> , or <b>MoveJ</b>
AccelS	Set the acceleration rate. This command is valid only when the motion mode is <b>Move</b> , <b>Arc3</b> , or <b>Circle3</b>
Speed	Set the velocity rate. This command is valid only when the motion mode is <b>Go</b> , <b>Jump</b> , or <b>MoveJ</b>
SpeedS	Set the velocity rate. This command is valid only when the motion mode is <b>Move</b> , <b>Arc3</b> , or <b>Circle3</b>
Arch	Set the index of sets of parameters ( <b>StartHeight</b> ,

Command	Description
	<b>zLimit, EndHeight</b> ) in <b>Jump</b> mode
CP	Set the continuous path function
LimZ	Set the maximum lifting height in the <b>Jump</b> mode

Table 4.20 Accel command

Function	Accel(R)
Description	Set the acceleration rate. This command is valid only when the motion mode is <b>Go, Jump, or MoveJ</b>
Parameter	R: Percentage. Value range: 1 - 100
Example	Accel(50) Go(P1) The robot moves to point P1 with 50% acceleration rate

Table 4.21 AccelS command

Function	AccelS(R)
Description	Set the acceleration rate. This command is valid only when the motion mode is <b>Move, Arc3, or Circle3</b>
Parameter	R: Percentage. Value range: 1 - 100
Example	AccelS(20) Move(P1) The robot moves to point P1 with 20% acceleration rate

Table 4.22 Speed command

Function	Speed(R)
Description	Set the velocity rate. This command is valid only when the motion mode is <b>Go, Jump, or MoveJ</b>
Parameter	R: Percentage. Value range: 1 - 100
Example	Speed(20) Go(P1) The robot moves to point P1 with 20% velocity rate



Table 4.23 SpeedS command

Function	SpeedS(R)
Description	Set the acceleration rate. This command is valid only when the motion mode is <b>Move</b> , <b>Arc3</b> , or <b>Circle3</b>
Parameter	R: Percentage. Value range: 1 - 100
Example	SpeedS(20) Move(P1) The robot moves to point P1 with 20% velocity rate

Table 4.24 Arch command

Function	Arch(Index)
Description	Set the index of sets of parameters ( <b>StartHeight</b> , <b>zLimit</b> , <b>EndHeight</b> ) in the <b>Jump</b> mode The sets of parameters need to be set on the <b>Parameter &gt; RobotParams&gt;PlayBack Arch</b> of DobotSCStudio
Parameter	Index: Index of the sets parameters. Value range: 0 - 9 This parameter is valid only when the right index has been selected from the <b>Parameter &gt; RobotParams &gt; PlayBack Arch</b> of DobotSCStudio
Example	Arch(1) Jump(P1)

Table 4.25 CP command

Function	CP(R)
Description	Set the continuous path rate. This command is valid only when the motion mode is <b>Go</b> , <b>Move</b> , <b>Arc3</b> , <b>Circle3</b> , or <b>MoveJ</b>
Parameter	R: Continuous path rate. Value range: 0 -100 <b>0</b> indicates that the Continuous path function is disabled
Example	CP(50) Move(P1) Move(P2) The robot moves from point P1 to point P2 with 50% Continuous path ratio

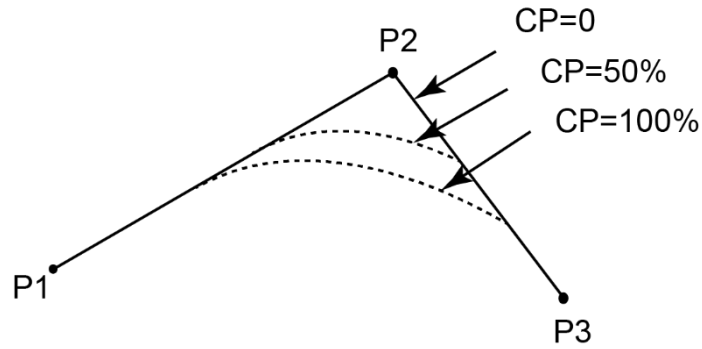


Figure 4.1 Continuous path

Table 4.26 LimZ command

Function	LimZ(zValue)
Description	Set the maximum lifting height in Jump mode
Parameter	zValue: The maximum lifting height which cannot exceed the Z-axis limiting position of the robot
Example	LimZ(80) Jump(P," Start=10 ZLimit=LimZ End=50 ")

## 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)
DOExecute	Set the status of the digital output port (Immediate command)

### NOTE

Dobot M1 Pro supports two kinds of commands: Immediate command and queue command:

- Immediate command: The M1 Pro will process the command once received regardless of whether there is the rest commands processing or not in the current controller;
- Queue command: When the M1 Pro receives a command, this command will be pressed into the internal command queue. The M1 Pro will execute commands in

the order in which the commands were pressed into the queue.

Table 4.28 Digital input command

Function	<i>DI(index)</i>
Description	Get the status of the digital input port
Parameter	index: Digital input index. Value range: 1 - 16
Return	<ul style="list-style-type: none"> <li>When an index is set in the DI function, <b>DI(index)</b> returns the status (ON/OFF) of this specified input port</li> <li>When there is no index in the DI function, <b>DI()</b> returns the status of all the input ports, which are saved in a table</li> </ul> <p>For example, local di=(), the saving format is <b>{num = 24 value = {0x55, 0xAA, 0x52}}</b>, you can obtain the status of the specified input port with <b>di.num</b> and <b>di.value[n]</b></p>
Example	<pre>if (DI(1))==ON then Move(P1) end</pre> <p>The robot moves to point P1 when the status of the digital input port <b>1</b> is <b>ON</b></p>

Table 4.29 Digital output command (Queue command)

Function	<i>DO(index, ON   OFF)</i>
Description	Set the status of digital output port (Queue command)
Parameter	<ul style="list-style-type: none"> <li>index: Digital output index. Value range: 1- 24</li> <li>ON/OFF: Status of the digital output port. ON: High level; OFF: Low level</li> </ul>
Example	<pre>DO(1,ON)</pre> <p>Set the status of the digital output port 1 to <b>ON</b></p>

Table 4.30 Digital output command (Immediate command)

Function	<i>DOExecute(index, ON   OFF)</i>
Description	Set the status of digital output port (Immediate command)
Parameter	<ul style="list-style-type: none"> <li>index: Digital output index. Value range: 1 - 24</li> <li>ON/OFF: Status of the digital output port. ON: High level; OFF: Low level</li> </ul>
Example	<pre>DOExecute(1,OFF)</pre> <p>Set the status of the digital output port 1 to <b>OFF</b></p>

## 4.11 Program Managing Commands

Table 4.31 Program managing command

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	<code>Wait(<i>time</i>)</code>
Description	Set the delay time for robot motion commands
Parameter	time: Delay time. Unit: ms
Example	<pre>Go(P1) Wait(1000) Wait for 1000ms after the robot moves to point P1</pre>

Table 4.33 Sleep command

Function	<code>Sleep(<i>time</i>)</code>
Description	Set the delay time for all commands
Parameter	time: Delay time. Unit: ms
Example	<pre>while true do Speed(100) Go(P1) sleep(3) Speed(100) Accel(40) Go(P2) sleep(3) end</pre>

Table 4.34 Pause command




Function	Pause()
Description	<p>Pause the running program</p> <p>When the program runs to this command, robot pauses running and the button  on DobotSCStudio turns into . If the robot continues to run, please click .</p>
Parameter	None
Example	<pre>while true do Go(P1) Go(P2) Pause() Go(P3) Go(P4) end</pre> <p>The robot moves to point P2 and then pauses running</p>

Table 4.35 Star timing command

Function	ResetElapsedTime()
Description	<p>Start timing after all commands before this command are executed completely. Use in conjunction with ElapsedTime() command</p> <p>For example: Get the execution time that a piece of code takes</p>
Parameter	None
Return	None
Example	<pre>Go(P2, " Speed=100 Accel=100") ResetElapsedTime() for i=1,10 do Jump(P1, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185") Jump(P2, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185") end print (ElapsedTime()) Sleep(1000)</pre>

Table 4.36 Stop timing command

Function	ElapsedTime()
Description	Stop timing and return the time difference. Use in conjunction with ResetElapsedTime() command
Parameter	None
Return	Time difference. Unit: ms
Example	<pre> Go(P2, " Speed=100 Accel=100") ResetElapsedTime() for i=1,10 do Jump(P1, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185") Jump(P2, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185") end print (ElapsedTime()) Sleep(1000)                     </pre>

Table 4.37 Get current time command

Function	Systime()
Description	Get the current time
Parameter	None
Return	Current time
Example	<pre> Go(P2, " Speed=100 Accel=100") local time1=Systime() for i=1,10 do Jump(P1, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185") Jump(P2, " Speed=100 Accel=100 Start=0 End=0 ZLimit=185") end local time2=Systime() local time = time2 - time1 Sleep(1000)                     </pre>

## 4.12 Pose Getting Command

Table 4.38 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	None
Return	Cartesian coordinate of the current pose
Example	<pre> local currentPose = GetPose() --Get the current pose local liftPose = { coordinate = {currentPose.coordinate[1], currentPose.coordinate[2], currentPose.coordinate[3],currentPose.coordinate[4] }, tool = currentPose.tool, user = currentPose.user}                     </pre>

Table 4.39 Pose command (2)

Function	GetAngle()
Description	Get the current pose of the robot under the Joint coordinate system
Parameter	None
Return	Joint coordinate of the current pose
Example	<pre> local armPose local joint = GetAngle() --Get the current pose local liftPose = { joint = {joint.joint[1], joint.joint[2], joint.joint[3], joint.joint[4]}, tool = 0, user = 0}                     </pre>

### 4.13 TCP

Table 4.40 Create TCP command

Function	<code>err, socket = TCPCreate(isServer, IP, port)</code>
Description	Create a TCP network Only support a single connection

Parameter	isServer: Whether to create a server. 0: Create a client; 1: 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, <b>port</b> 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: 0: TCP network is created successfully 1: TCP network is created failed Socket: Socket object
Example	Please refer to Program 4.1 and Program 4.2

Table 4.41 TCP connection command

Function	TCPStart( <i>socket, timeout</i> )
Description	Connect a client to a server with the TCP protocol
Parameter	socket: Socket object timeout: Wait timeout. Unit: s. If <b>timeout</b> is 0, the connection is still waiting. If not, after exceeding the timeout, the connection is exited.
Return	<ul style="list-style-type: none"> <li>• 0: TCP connection is successful</li> <li>• 1: Input parameters are incorrect</li> <li>• 2: Socket object is not found</li> <li>• 3: Timeout setting is incorrect</li> <li>• 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</li> </ul>
Example	Please refer to Program 4.1 and Program 4.2

Table 4.42 Receive data command

Function	err, Recbuf = TCPRead( <i>socket, timeout, type</i> )
Description	Robot as a client receives data from a server Robot as a server receives data from a client



Parameter	socket: socket object  timeout: Receiving timeout. Unit: s. If <b>timeout</b> 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 <b>type</b> is not set, the buffer format of <b>RecBuf</b> is a table. If <b>type</b> is set to <b>string</b> , the buffer format is a string
Return	err:  0: Receiving data is successful  1: Receiving data is failed  Recbuf: Data buffer
Example	Please refer to Program 4.1 and Program 4.2

Table 4.43 Send data command

Function	TCPWrite( <i>socket, buf, timeout</i> )
Description	The robot as a client sends data to a server  The robot as a server sends data to a client
Parameter	socket: Socket object  buf: Data sent by the robot  timeout: Timeout. Unit: s. If <b>timeout</b> 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: Sending data is failed
Example	Please refer to Program 4.1 and Program 4.2

Table 4.44 Release TCP network command

Function	TCPDestroy( <i>socket</i> )
Description	Release a TCP network
Parameter	socket: Socket object
Return	0: Releasing TCP is successful  1: Releasing TCP is failed
Example	Please refer to Program 4.1 and Program 4.2

 NOTICE

- Only a single TCP connection is supported. Please start the server before connecting a client. Please shut down the client before disconnection, to avoid re-connection failure since the server port is not released in time.
- When the robot is set as a server, the IP address of the robot can be checked and modified on the **Config> NetworkSetting** page of DobotSCStudio. Also, the 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.

## Program 4.1 TCP server demo

```
local ip="192.168.2.6" // IP address of the robot as a server
local port=6001 // Server port
local err=0
local socket=0
err, socket = TCPCreate(true, ip, port)
if err == 0 then
    err = TCPStart(socket, 0)
    if err == 0 then
        local RecBuf
        while true do
            TCPWrite(socket, "tcp server test") // Server sends data to client
            err, RecBuf = TCPRead(socket,0,"string") // Server receives the data from client
            if err == 0 then
                Go(P1) //Start to run motion commands after the server receives data
                Go(P2)
                print(Recbuf)
            else
                print("Read error ".. err)
                break
            end
        end
    end
else
    print("Create failed ".. err)
end
TCPDestroy(socket)
else
```

```
print("Create failed ".. err)
end
```

#### Program 4.2 TCP client demo

```
local ip="192.168.2.25"           // External equipment such as a camera is set as the server
local port=6001                 // Server port
local err=0
local socket=0
err, socket = TCPCreate(false, ip, port)
if err == 0 then
    err = TCPStart(socket, 0)
    if err == 0 then
        local RecBuf
        while true do
            TCPWrite(socket, "tcp client test")           // Client sends data to server
            TCPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
            err, RecBuf = TCPRead(socket, 0)              // Client receives data from server
            if err == 0 then
                Go(P1)           // Start to run motion commands after the client receives the data
                Go(P2)
                print(Recbuf)
            else
                print("Read error ".. err)
                break
            end
        end
    end
    else
        print("Create failed ".. err)
    end
    TCPDestroy(socket)
else
    print("Create failed ".. err)
end
```

### 4.14 UDP

Table 4.45 Create UDP network command

Function	err, socket = UDPCreate( <i>isServer, IP, port</i> )
Description	Create a UDP network Only a single connection is supported
Parameter	isServer: Whether to create a server. 0: Create a client; 1: 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, <b>port</b> cannot be set to 502 or 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: 0: The UDP network is created successfully 1: The UDP network is created failed socket: Socket object
Example	Please refer to Program 4.3 and Program 4.4

Table 4.46 Receive data command

Function	err, Recbuf = UDPRead( <i>socket, timeout, type</i> )
Description	The robot as a client receives data from a server The robot as a server receives data from a client
Parameter	socket: socket object timeout: Receiving timeout. Unit: s. If <b>timeout</b> is 0 or 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 <b>type</b> is not set, the buffer format of <b>RecBuf</b> is a table. If <b>type</b> is set to <b>string</b> , the buffer format is a string
Return	err: 0: Receiving data is successful 1: Receiving data is failed Recbuf: Data buffer
Example	Please refer to Program 4.3 and Program 4.4

Table 4.47 Send data command

Function	UDPWrite( <i>socket, buf, timeout</i> )
Description	The robot as a client sends data to a server The robot as a server sends data to a client
Parameter	socket: Socket object buf: Data sent by the robot timeout: Timeout. Unit: s. If <b>timeout</b> 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: Sending data is failed
Example	Please refer to Program 4.3 and Program 4.4


**NOTICE**

- Only a single UDP connection is supported. Please start the server before connecting a client. Please shut down the client before disconnection, to avoid re-connection failure since the server port is not released in time.
- When the robot is set as a server, the IP address of the robot can be checked and modified on the **Config > NetworkSetting** page of DobotSCStudio. Also, the 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.

Program 4.3 UDP server demo

```

local ip="192.168.2.6" // IP address of the robot as a server
local port=6201 // Server port
local err=0
local socket=0
err, socket = UDPCreate(true, ip, port)
if err == 0 then
    local RecBuf
    while true do
        UDPWrite(socket, "udp server test") // Server sends data to client
        err, RecBuf = UDPRead(socket, 0) //Server receives the data from client
        if err == 0 then
            Go(P1) // Start to run motion commands after the server receives the data
        end
    end
end
    
```

```
        Go(P2)
        print(Recbuf)
    else
        print("Read error ".. err)
        break;
    end
end
end
else
    print("Create failed ".. err)
end
```

#### Program 4.4 UDP client demo

```
local ip="192.168.2.25" // IP address of the external equipment
as a server
local port=6200 // server port
local err=0
local socket=0

err, socket = UDPCreate(false, ip, port)
if err == 0 then
    local RecBuf
    while true do
        UDPWrite(socket, "udp client test") // Client sends data to server
        UDPWrite(socket, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07})
        err, RecBuf = UDPRead(socket, 0) // Client receives the data from server
        if err == 0 then
            Go(P1) // Start to run motion commands after the client receives the data
            Go(P2)
            print(Recbuf)
        else
            print("Read error ".. err)
            break
        end
    end
end
else
    print("Create failed ".. err)
```

end

## 4.15 Modbus

### 4.15.1 Modbus Register Description

Modbus protocol is a serial communication protocol. The M1 Pro can communicate with external equipment by this protocol. Here, External equipment such as a PLC is set as the Modbus master, and the M1 Pro 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 M1 Pro. Each register has 4096 addresses. For details, please see as follows.

- Coil register

Table 4.48 Coil register description

Coil register address (e.g.: PLC)	Coil register address (M1 Pro)	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 4.49 Discrete input register description

Discrete input register address (e.g.: PLC)	Discrete input register address(M1 Pro)	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

Discrete input register address (e.g.: PLC)	Discrete input register address(M1 Pro)	Data type	Description
11000-14096	999-4095	Bit	User-defined

- Input register

Table 4.50 Input register description

Input register address (e.g.: PLC)	Input register address (M1 Pro)	Data type	Description
30001-34096	0-4095	Byte	Reserved

- Holding register

Table 4.51 Holding register description

Holding register address (e.g.: PLC)	Holding register address (M1 Pro)	Data type	Description
40001-41000	0-999	Byte	Reserved
41001-44096	1000-4095	Byte	User-defined

#### 4.15.2 Command Description

Table 4.52 Rea coil register command

Function	<code>GetCoils(addr, count)</code>
Description	Read the coil value from the Modbus slave
Parameter	addr: Starting address of the coils to read. 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
Example	Read 5 coils starting at address 0 <code>Coils = GetCoils(0,5)</code> Return: <code>Coils={ 1,0,0,0,0}</code> As shown in Table 4.47, it indicates that the robot is in the starting state



Table 4.53 Set coil register command

Function	<code>SetCoils(addr, count, table)</code>
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: Coil value, stored in a table
Return	None
Example	Set 5 coils starting at address 1024 local Coils = {0,1,1,1,0} <code>SetCoils(1024, #coils, coils)</code>

Table 4.54 Read discrete input register command

Function	<code>GetInBits(addr, count)</code>
Description	Read the discrete input value from Modbus slave
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
Example	Read 5 discrete inputs starting at address 0 <code>inBits = GetInBits(0,5)</code> Return: <code>inBits = {0,0,0,1,0}</code> As shown in Table 4.48, it indicates the robot is in running state

Table 4.55 Read input register command

Function	<code>GetInRegs(addr, count, type)</code>
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"> <li>• Empty: Read 16-bit unsigned integer ( two bytes, occupy one register)</li> <li>• “U16”: Read 16-bit unsigned integer ( two bytes, occupy one register)</li> <li>• “U32”: Read 32-bit unsigned integer (four bytes, occupy two registers)</li> <li>• “F32”: Read 32-bit single-precision floating-point number (four bytes, occupy two registers)</li> <li>• “F64”: Read 64-bit double-precision floating-point number (eight bytes, occupy four registers)</li> </ul>
Return	Return a table, the first value in the table corresponds to the input register value at the starting address
Example	Example 1: Read a 16-bit unsigned integer starting at address 2048 <pre style="background-color: #f0f0f0;">data = GetInRegs(2048,1)</pre> Example 2: Read a 32-bit unsigned integer starting at address 2048 <pre style="background-color: #f0f0f0;">data = GetInRegs(2048, 1, “U32”)</pre>

Table 4.56 Read holding register command

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

Example	Example 1: Read a 16-bit unsigned integer starting at address 2048
	<code>data = GetHoldRegs(2048,1)</code>
	Example 1: Read a 32-bit unsigned integer starting at address 2048
	<code>data = GetInRegs(2048, 1, "U32")</code>

Table 4.57 Set holding register command

Function	<code>SetHoldRegs(addr, count, table, type)</code>
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"> <li>• Empty: Read 16-bit unsigned integer ( two bytes, occupy one register)</li> <li>• "U16": Set 16-bit unsigned integer ( two bytes, occupy one register)</li> <li>• "U32": Set 32-bit unsigned integer (four bytes, occupy two registers)</li> <li>• "F32": Set 32-bit single-precision floating-point number (four bytes, occupy two registers)</li> <li>• "F64": Set 64-bit double-precision floating-point number (eight bytes, occupy four registers)</li> </ul>
Return	None
Example	<p>Example1: Set a 16-bit unsigned integer starting at address 2048</p> <pre>local data = {6000} SetHoldRegs(2048, #data, data, "U16")</pre> <p>Example2: Set a 64-bit double-precision floating-point number starting at address 2048</p> <pre>local data = {95.32105} SetHoldRegs(2048, #data, data, "F64")</pre>

## 5. Process

### 5.1 Conveyor Tracking

#### 5.1.1 Overview

Conveyor tracking is **that the vision** system or sensor system finds the parts on the conveyor when conveyor moves constantly and the robot picks them up as they move.

#### 5.1.2 Building Environment

Figure 5.1 shows the communication process of conveyor tracking. Vision system or photoelectric sensor for detection is selected based on site requirements. If the photoelectric sensor is used, the part is detected by a change in the digital input data. If the vision system is used, the part is detected by the camera and this is triggered by the rising edge of digital output signal.

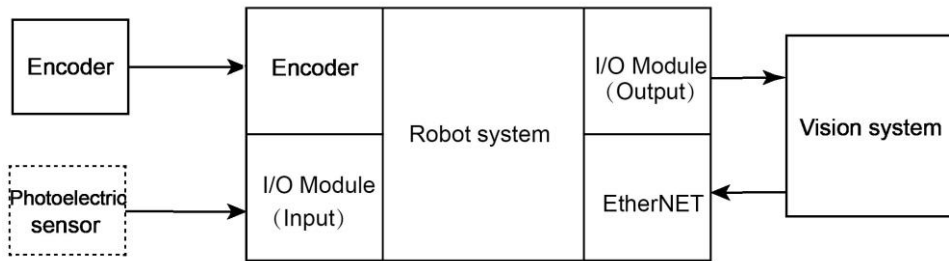


Figure 5.1 Communication process of the conveyor tracking

Figure 5.2 shows the full process environment of conveyor tracking. Please select a vision system or photoelectric sensor for detection based on site requirements.

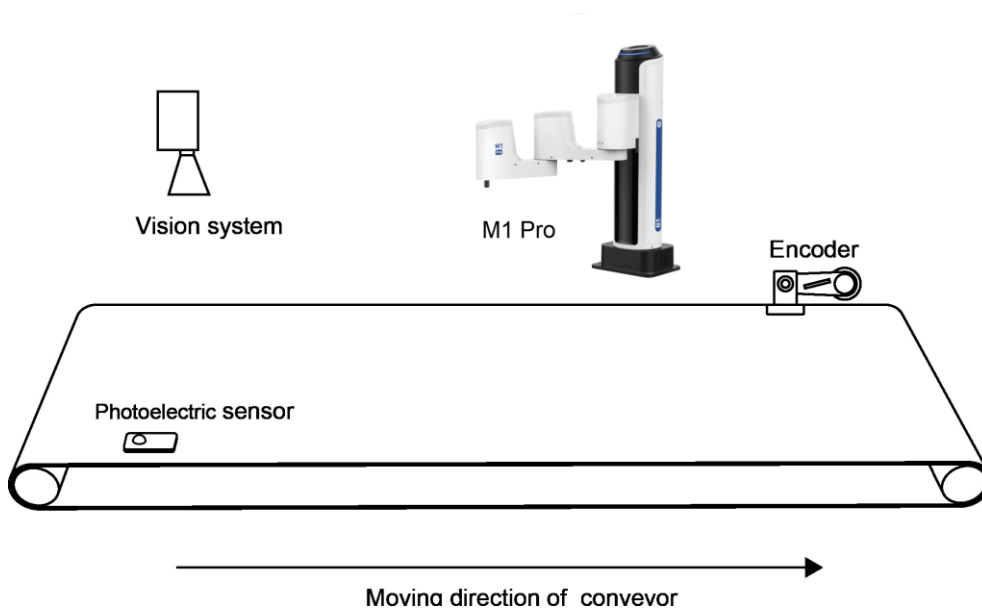


Figure 5.2 Process environment of the conveyor tracking

### 5.1.2.1 Encoder

An Encoder is used for recording the conveyor moving distance and the part position and reporting them to the M1 Pro by a counter. Please connect the Encoder to the **ENC** interface on the M1 Pro.

The ENC interface of the M1 Pro is shown in Figure 5.3, Table 5.1 lists the description of ENC interface. The recommended encoder model is E6B2-CWZ1X (1000P/R).

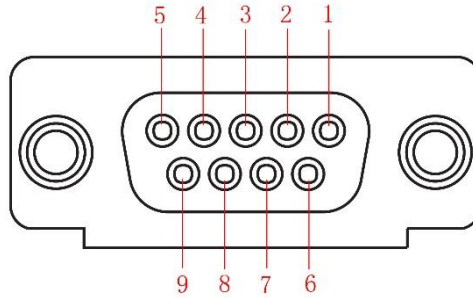


Figure 5.3 ENC interface

Table 5.1 ENC Interface description

No.	1	2	3	4	5	6	7	8	9
Description	ABZ_A+	ABZ_A-	ABZ_B+	ABZ_B-	ABZ_Z+	ABZ_Z-	5V	0V	unused

### 5.1.2.2 Photoelectric Sensor

The photoelectric sensor outputs different level signals according to whether the part is detected or not. When you connect the photoelectric sensor to a DI interface on the M1 Pro, the photoelectric sensor can detect parts by a change in this DI interface.

### 5.1.2.3 Vision System

A vision system is communicated with the M1 Pro by the TCP/IP protocol and is triggered by the DO interface on the M1 Pro to detect the part.

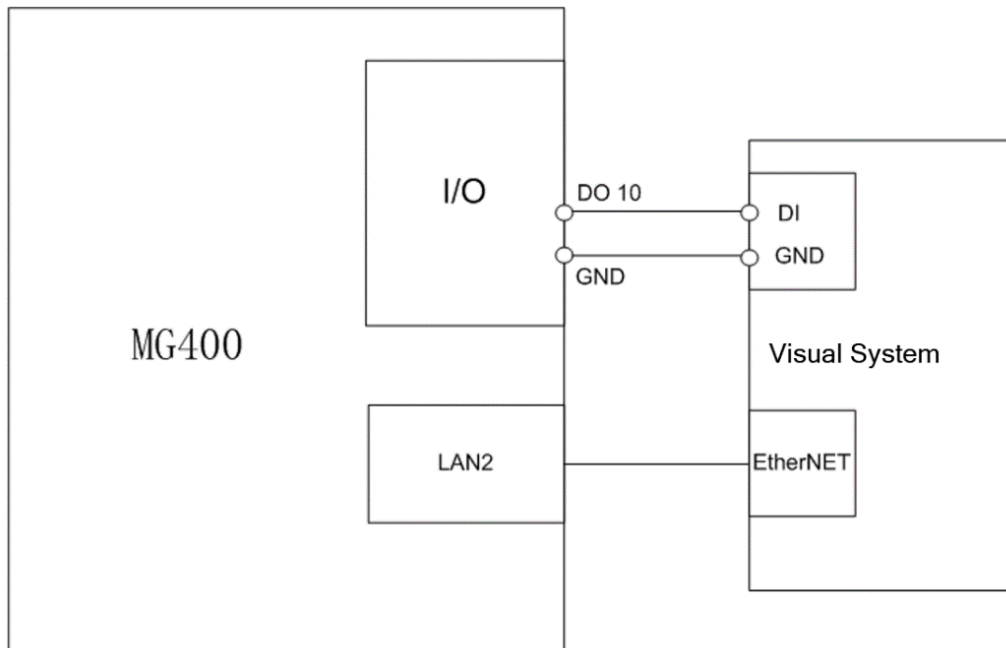


Figure 5.4 Vision system connection

### 5.1.3 Calibrating Conveyor

Before tracking parts, please **calibrating** the conveyor, for obtaining the positional relationship between the conveyor and the robot. In the following steps, we assume that the Y-axis positive direction under the base coordinate system is coincident with the moving direction of the conveyor.

#### Prerequisites

- The robot has been powered on.
- The calibration kit has been installed at the end of the robot.

#### Procedure

**Step 1** Put down a label on the conveyor, as shown in Figure 5.5.

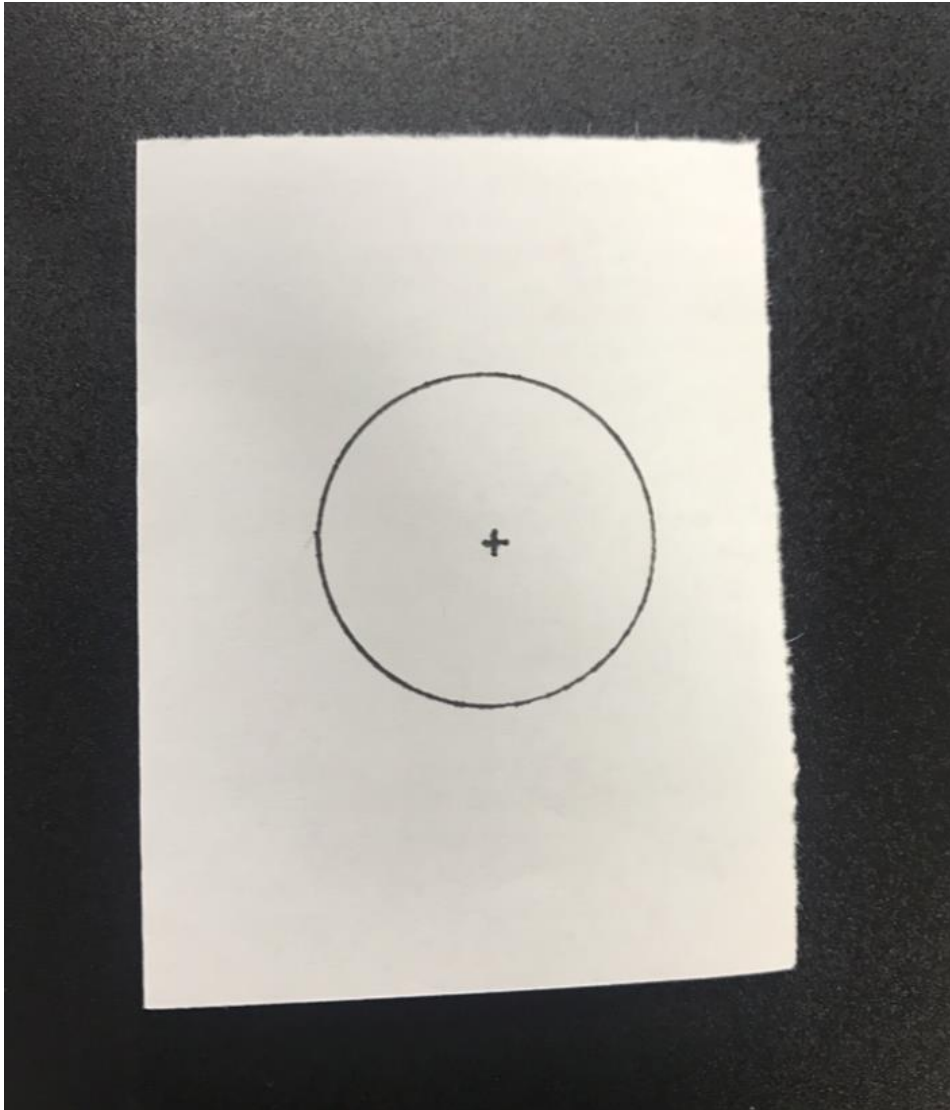


Figure 5.5 Put down a label on the conveyor

**Step 2** Click  > **GlobalCoordinate** > **Coordinate User**.

The **Coordinate User** page is displayed, as shown in Figure 5.6.

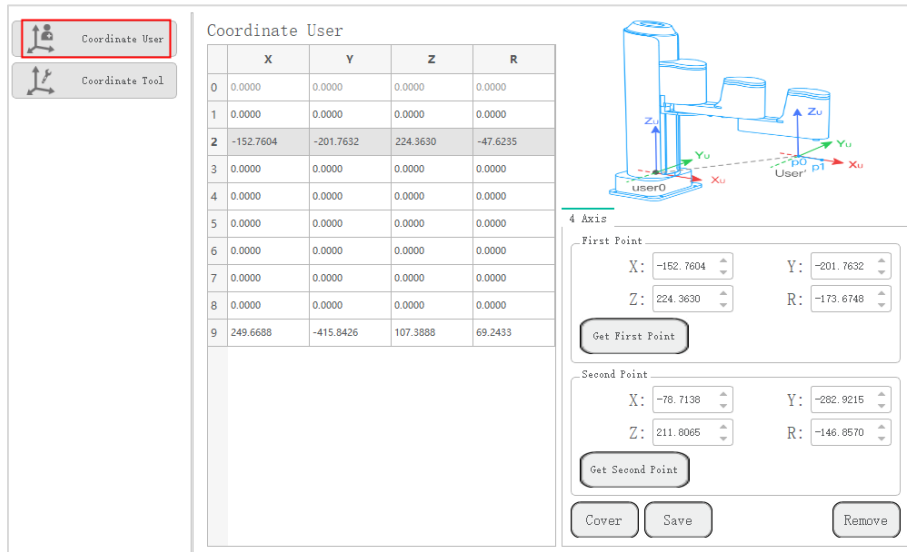


Figure 5.6 User coordinate system page

- Step 3** Click to enable the motor and jog the robot to the label position on the conveyor, and click **Get First Point** on the **First Point** section, as shown in Figure 5.7. We called this point as point A, which is the origin of the User coordinate system.

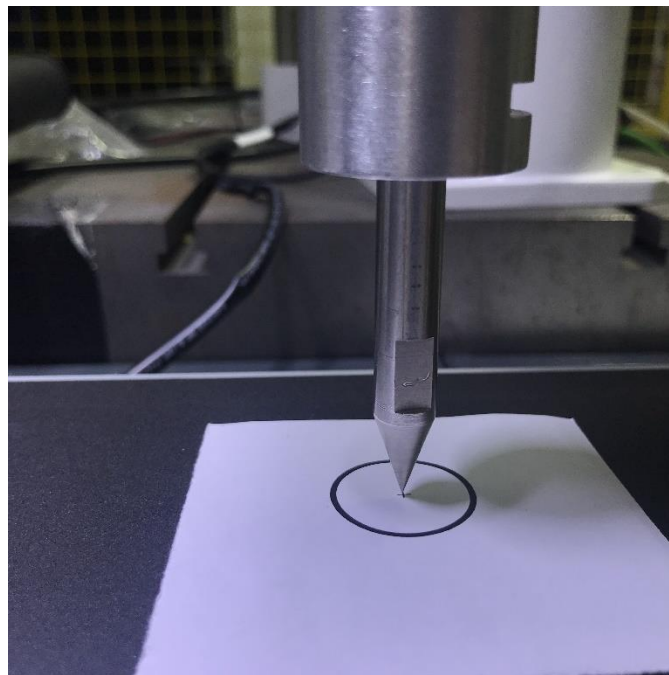


Figure 5.7 Conveyor calibration

- Step 4** Control the conveyor move a specified distance.
- Step 5** Jog the robot to the label position on the conveyor, and click **Get Second Point** on



the **Second Point** section. We called this point as point B. The line from point A to point B 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 5.8.

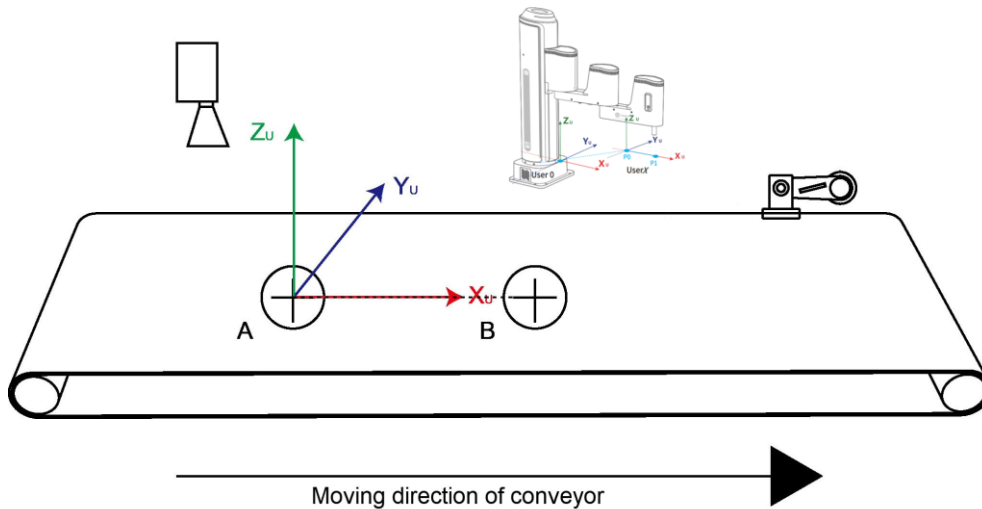


Figure 5.8 User coordinate system calibration

**Step 6** Click **Add** and **Save** to generate the User coordinate system.

### NOTICE

The R-axis coordinate after calibration depends on the positional relationship between robot and conveyor.

- If the moving direction of the conveyor is the same with the X-axis positive direction under the base coordinate system, the R-axis coordinate after calibration is  $0^\circ$ .
- If the moving direction of the conveyor is the same with the X-axis negative direction under the base coordinate system, the R-axis coordinate after calibration is  $180^\circ$  or  $-180^\circ$ .
- If the moving direction of the conveyor is the same with the Y-axis positive direction under the base coordinate system, the R-axis coordinate after calibration is  $90^\circ$ .
- If the moving direction of the conveyor is the same with the Y-axis negative direction under the base coordinate system, the R-axis coordinate after calibration is  $-90^\circ$ .

## 5.1.4 Configuring Conveyor

### Prerequisites

- The robot has been powered on.

- The calibration kit has been installed at the end of the robot.

### NOTICE

- The full process should be operated under the base coordinate system and the matched calibration kit is required.
- Please be sure to follow the steps to operate, otherwise, the parameter setting will fail.

## Procedure

**Step 1** Click  > **Process > Tracking.**

The conveyor tracking page is displayed, as shown in Figure 5.9.

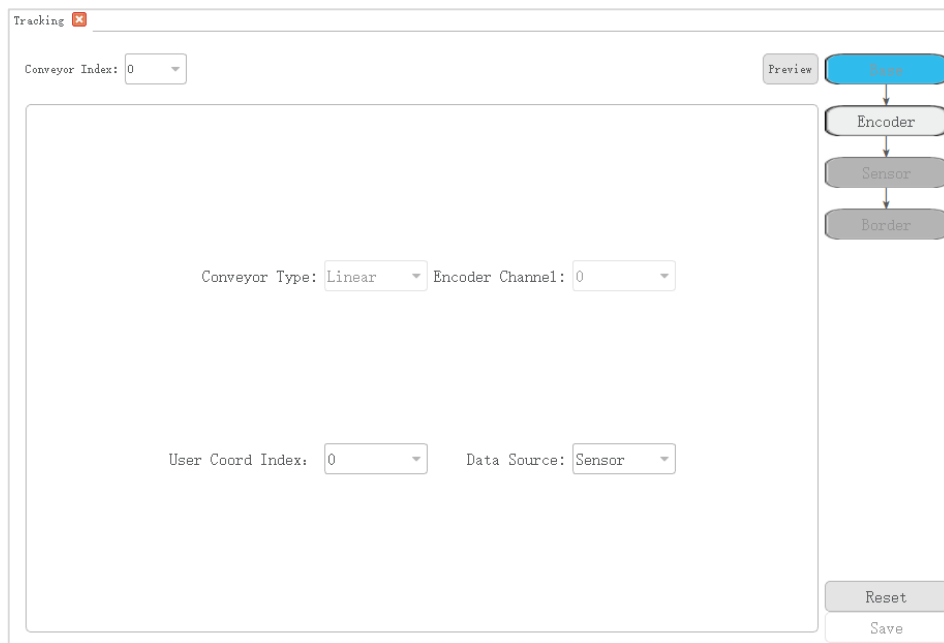


Figure 5.9 Conveyor tracking page

**Step 2** Set the basic parameters.

Table 5.2 shows the basic parameter description.

Table 5.2 Basic parameter description

Parameter	Description
Conveyor index	Conveyor index: 0 or 1 Please select the right index according to the actual situation
Conveyor Type	Conveyor type

Parameter	Description
	<ul style="list-style-type: none"> <li>• Linear</li> <li>• Circular</li> </ul> <p>Only linear type is supported</p>
Encoder Channel	<p>Encoder channel</p> <p>This parameter cannot be set</p>
User Coord Index	<p>User coordinate system index</p> <p>Please select the right index according to the user coordinate system set in <i>5.1.3 Calibrating Conveyor</i></p>
Data Source	<p>Detection Mode</p> <ul style="list-style-type: none"> <li>• Sensor: Use a sensor to detect parts</li> <li>• Camera: Use a vision system to detect parts</li> </ul> <p>Please select the mode based on site requirements</p>

**Step 3** Click **Encoder** to calibrate the Encoder.

1. Put down a label on the conveyor, as shown in Figure 5.10.

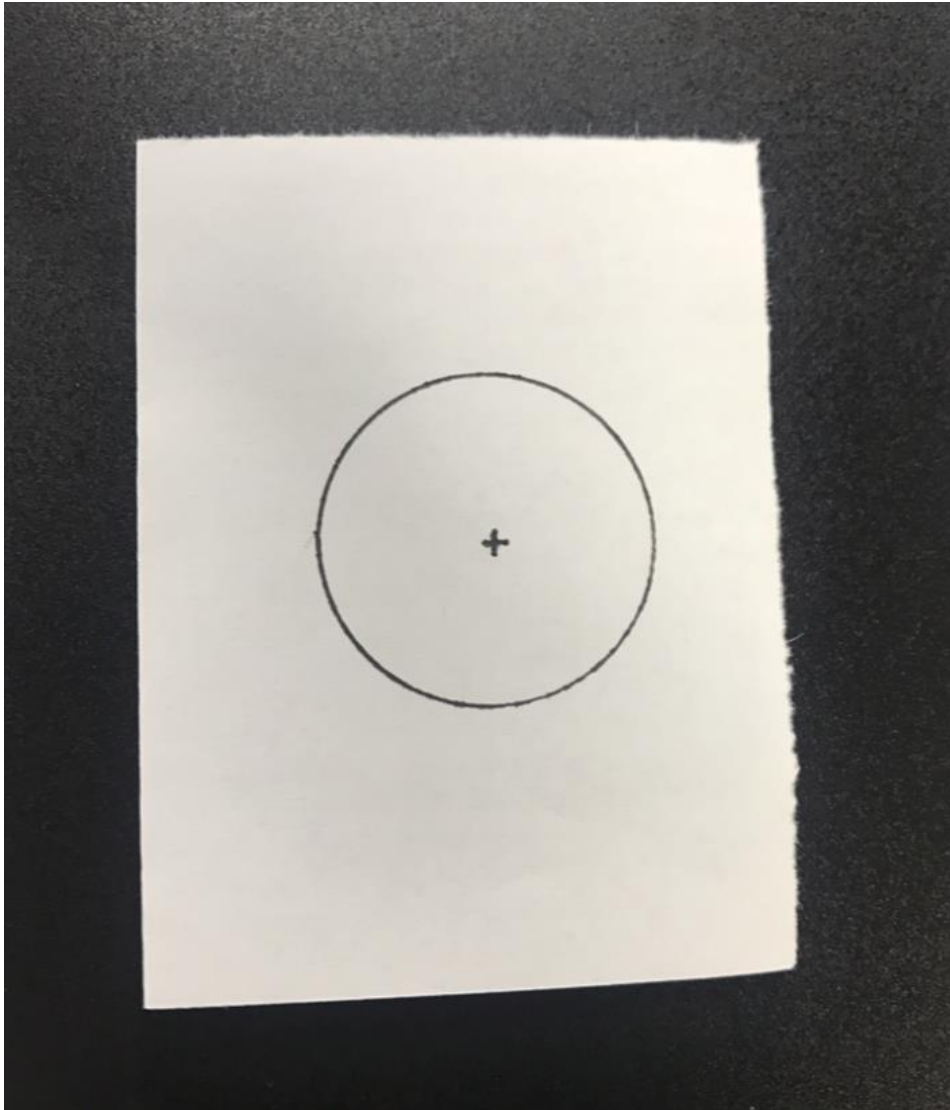


Figure 5.10 Put down a label on the conveyor

2. Jog the robot to the label position on the conveyor, then click **1**, as shown in Figure 5.11 and Figure 5.12.

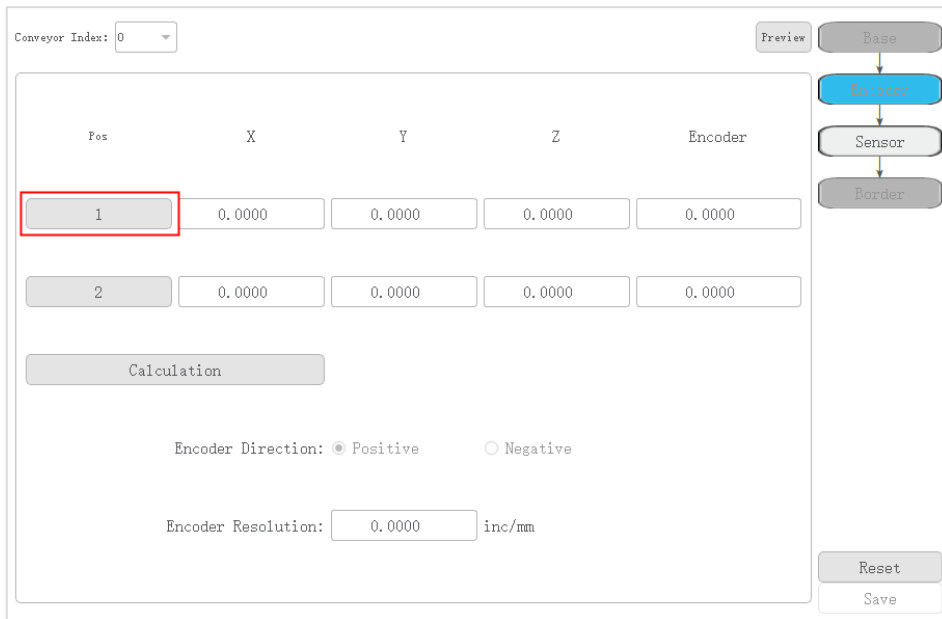


Figure 5.11 Encoder calibration page

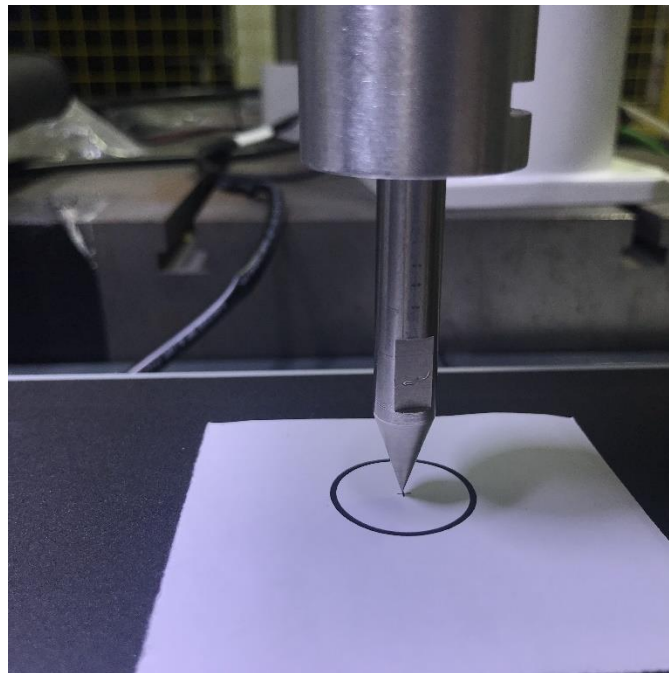


Figure 5.12 Encoder calibration

3. Control the conveyor to move a specified distance.
4. Jog the robot to the label position on the conveyor, then click **2**.
5. Click **Calculation** to obtain the Encoder resolution.

Encoder resolution is the pulse increment of the Encoder per unit length that conveyor moves

If Data Source is sensor, please execute **Step 4**. If not, please execute **Step 5**.

**Step 4** (Optional) Click **Sensor** to calibrate the sensor, as shown in Figure 5.13.

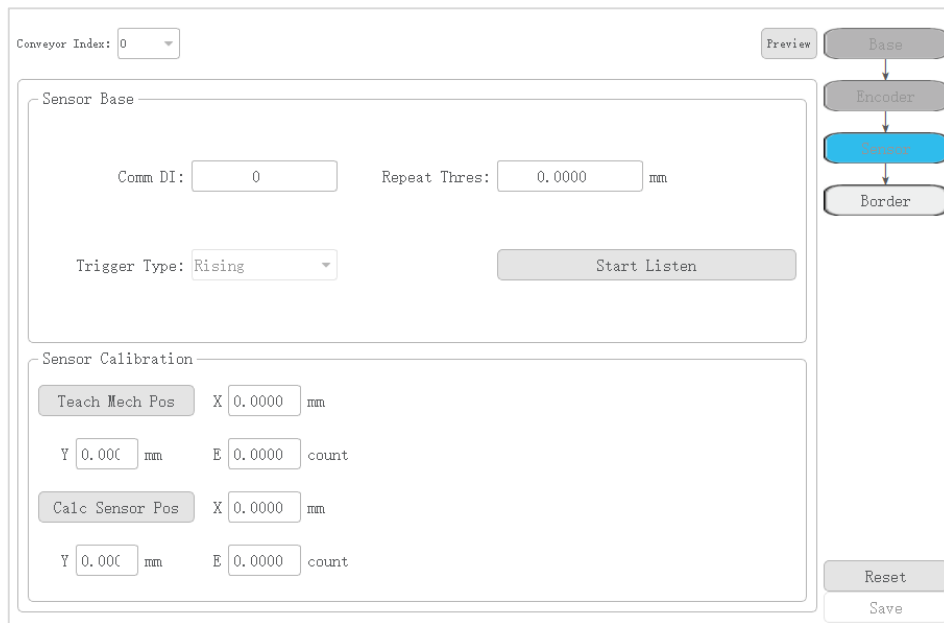


Figure 5.13 Sensor calibration page

Sensor calibration is to obtain the position where the sensor finds the part so that the position of the part under the User coordinate system at every moment can be calculated based on the coordinate offset when the part moves along with the conveyor.

Control the conveyor move to a position where the part on the conveyor is within the workspace of the robot and has been passed the sensor, and jog the robot to the part center for obtaining the current taught position. At the same time, the robot records the moving distance of the conveyor after the part is passed the sensor. According to the current taught position and the moving distance of the conveyor, we can get the part position when the sensor locates the part. For details, please see as follows.

1. Set the related parameters on the sensor setting page.

Table 5.3 lists the sensor parameter description.

Table 5.3 Sensor parameter description

Parameter	Description
Conveyor index	Conveyor index: 0 or 1 Please select the right index according to the actual

Parameter	Description
	situation
Comm DI	Please set the right DI according to the actual situation, the M1 Pro has 18 DI interfaces, usually select the DI interface on the Base
Trigger Type	Signal outputted when the sensor finds the part <ul style="list-style-type: none"> <li>• Raising edge</li> <li>• Falling edge</li> </ul> This parameter cannot be set
Repeat Thres	Reject distance. This parameter is set based on site requirements  This is used to prevent the registration of the duplicate parts

2. Click **Start Listen**.
3. Put a part on the upstream area of the conveyor and Control the conveyor move. After the sensor finds the part and meanwhile the part is in the workspace of the robot, please control the conveyor stop.
4. Jog the robot to the part center, then click **Teach Mech Pose** to obtain the current position.
5. Click **Calc Sensor Pos** to obtain the position where the sensor finds the part

**Step 5** (Optional) Click **Camera** to calibrate the vision system, as shown in Figure 5.14.

Before calibrating the vision system, you need to set the robot IP address and port on the vision software for communication between the robot and vision system with the TCP/IP protocol. The data transmitted between the robot and vision system is visual data, and visual data is pixel coordinate and angle of the part. The robot IP address is 192.168.2.6, and the port is 8080. The data format is as follows.

- The parts found in the same frame are sent together.
- The data format of a part is (conveyor Index;x,y,r,classID). classID is the part type.
- Different part data is separated by a semicolon.
- A frame is terminated by & character.

e.g.: 0;x1,y1,r1,1;x2,y2,r2,4;x3,y3,r3,2&

### NOTICE

Vision software depends on the vision brand. So, the settings about the vision system are different. The details on how to use the vision software are not described in this topic.

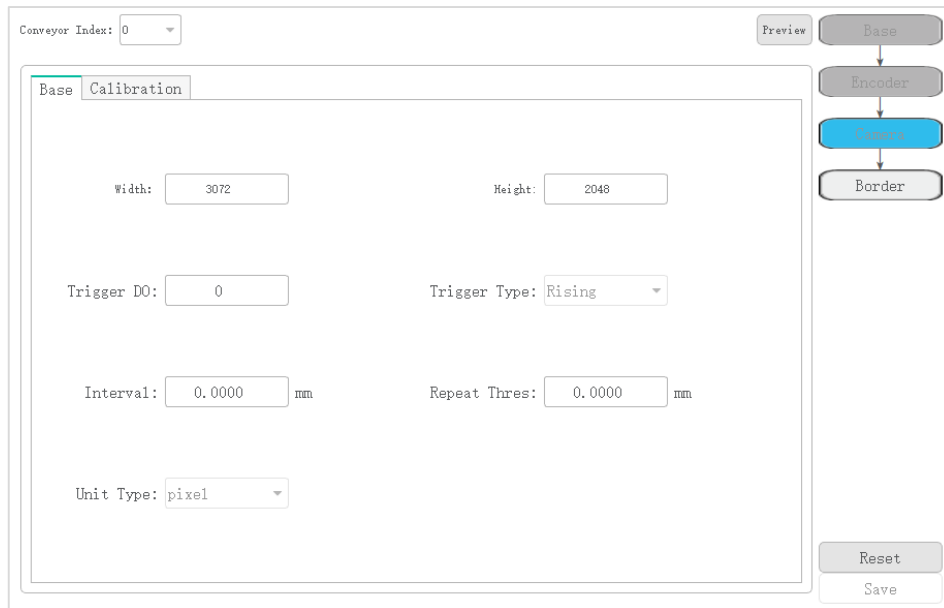


Figure 5.14 Vision system calibration page

The vision system is to obtain the part coordinate under the User coordinate system where the camera finds it from the conveyor. So that the position of the part under the User coordinate system at every moment can be calculated based on the coordinate offset when the part moves along with the conveyor.

Place the calibration board on the conveyor which is in the vision search area and obtain the image coordinates of the nine points on the calibration board and record the current position of the conveyor. Move the conveyor to a position where the calibration board is in the robot workspace and jog the robot to the nine points on the calibration board and obtain their Cartesian coordinates respectively. At the same time, record the current position of the conveyor. Based on the Cartesian coordinates and image coordinates of the nine points and the moving distance of the conveyor, we can calculate the Cartesian coordinates where camera finds the calibration board and obtain the relationship between the image coordinate and the Cartesian coordinate. For details, please see as follows.

1. Set the basic parameters of the vision system on the **Base** tab.

Table 5.4 lists the basic parameter description.

Table 5.4 Basic parameter description

Parameter	Description
Conveyor Index	Conveyor index: 0 or 1 Please select the right index according to the actual situation



Parameter	Description
Width	Pixels in the X direction of the camera
Height	<p>Pixels in the Y direction of the camera</p> <p>For example, for a 6 megapixel camera, its resolution is 3072*2048, the <b>Width</b> is set to 3072 and the <b>Height</b> is set to 2048</p>
Trigger DO	Please set the right DO according to the actual situation, the M1 Pro has 18 DO interfaces, usually select the DO interface on the Base
Trigger Type	This parameter cannot be set
Interval	<p>Interval photography</p> <p>Namely, the robot gives a DO signal to trigger photography each time conveyor moves a given distance</p> <p>The recommended value range is <math>(1-d)/2 &lt; \text{Interval} &lt; (1-d)</math></p> <p><b>l</b> indicates the width of the vision search area in the conveyor moving direction and <b>d</b> indicates the maximum width of the part</p>
Repeat Thres	<p>Reject distance. This parameter is set based on site requirements</p> <p>This is used to prevent the registration of the duplicate parts</p>
Unit Type	This parameter cannot be set

2. Click **Calibration**.

The calibration page is displayed.

3. Place the calibration board on the conveyor which is in the vision search area and click **Conveyor pose** to record the current Encoder value. The calibration board is shown in Figure 5.15.

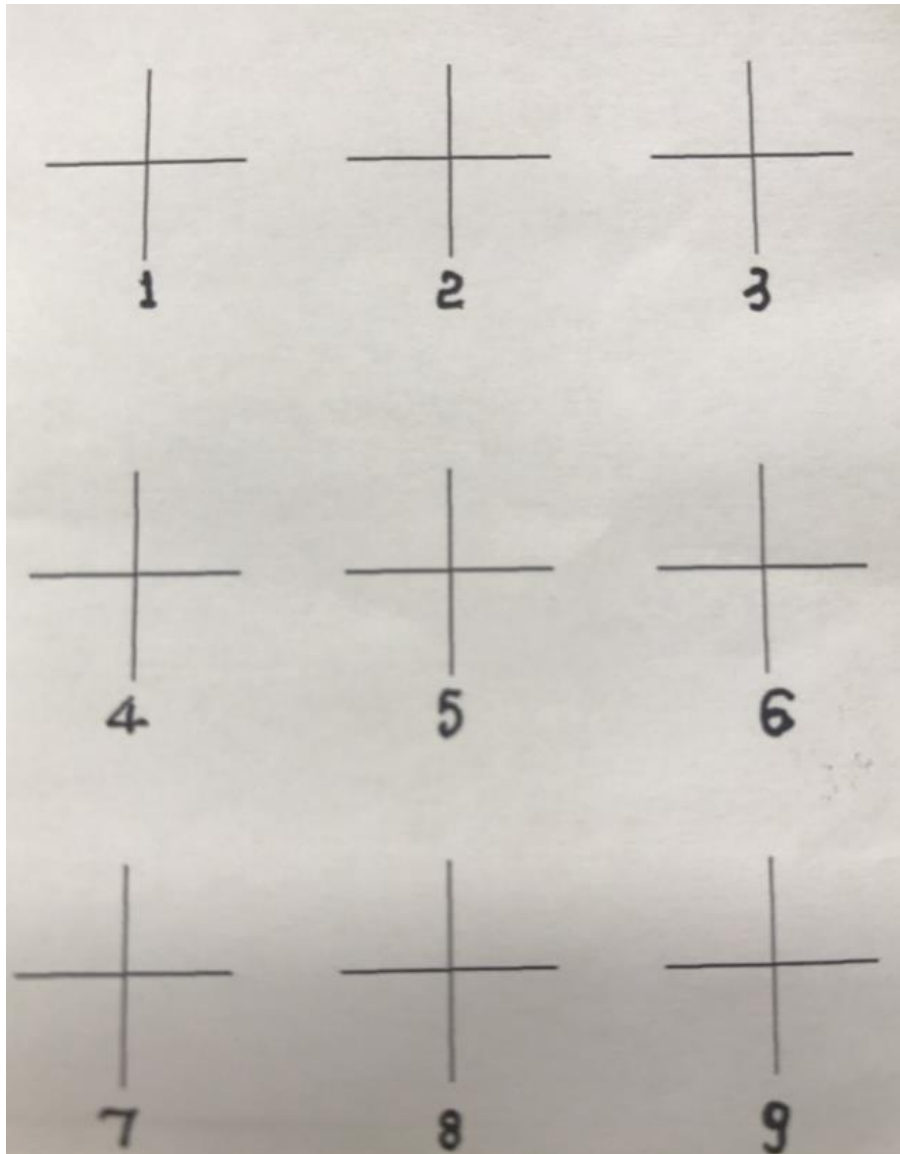


Figure 5.15 Calibration board

4. Get the image coordinates of the nine points on the calibration board from the vision software after the vision system finds the calibration board and input them to the corresponding positions on the calibration page, as shown in Figure 5.16.

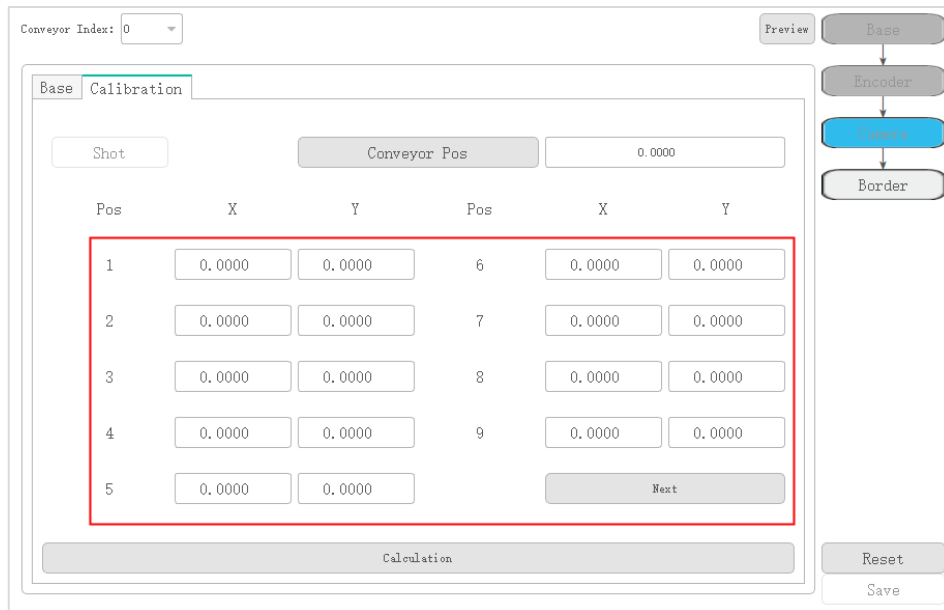


Figure 5.16 Get the image coordinates

5. Click **Next**.
6. Move the conveyor to the position where the calibration board is in the robot workspace and stop, then click **Conveyor Pos** to obtain the current Encoder value.
7. Jog the robot to the corresponding nine points on the calibration board and click the right index on the calibration page respectively as shown in Figure 5.17 and Figure 5.18.

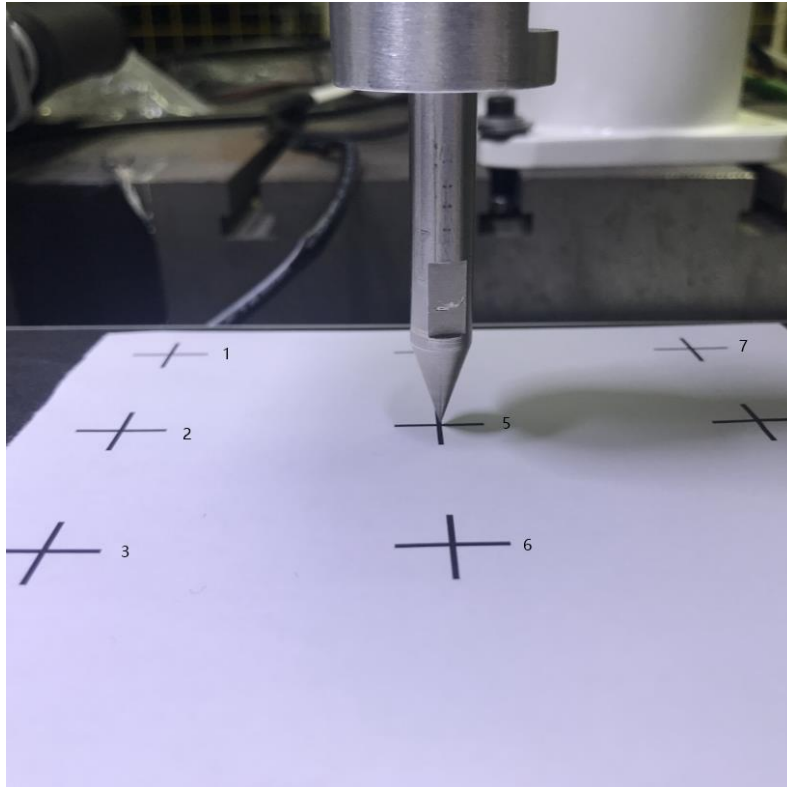


Figure 5.17 Vision calibration

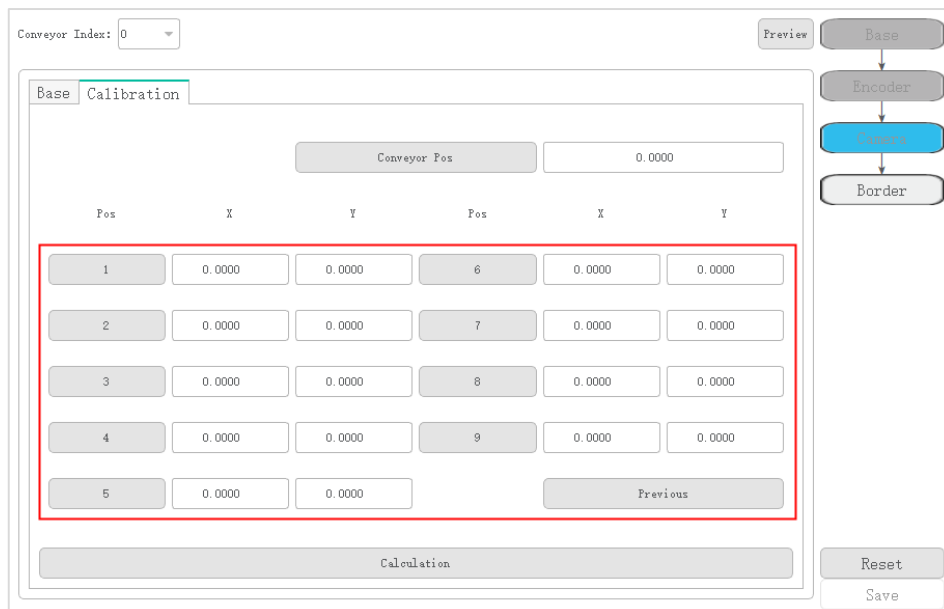


Figure 5.18 Get the Cartesian coordinates of the nine points on the calibration board

NOTICE

Please be sure to obtain the Cartesian coordinates in the order of the image coordinates of the nine points to avoid vision calibration errors.

8. Click **Calculation** to calculate the Cartesian coordinates of the nine points as the vision system finds them and obtain the relationship between the image coordinate and the Cartesian coordinate based on the image coordinates and the Cartesian coordinates of the nine points and the moving distance of the conveyor.

**Step 6** Click **Border** to calibrate the border.

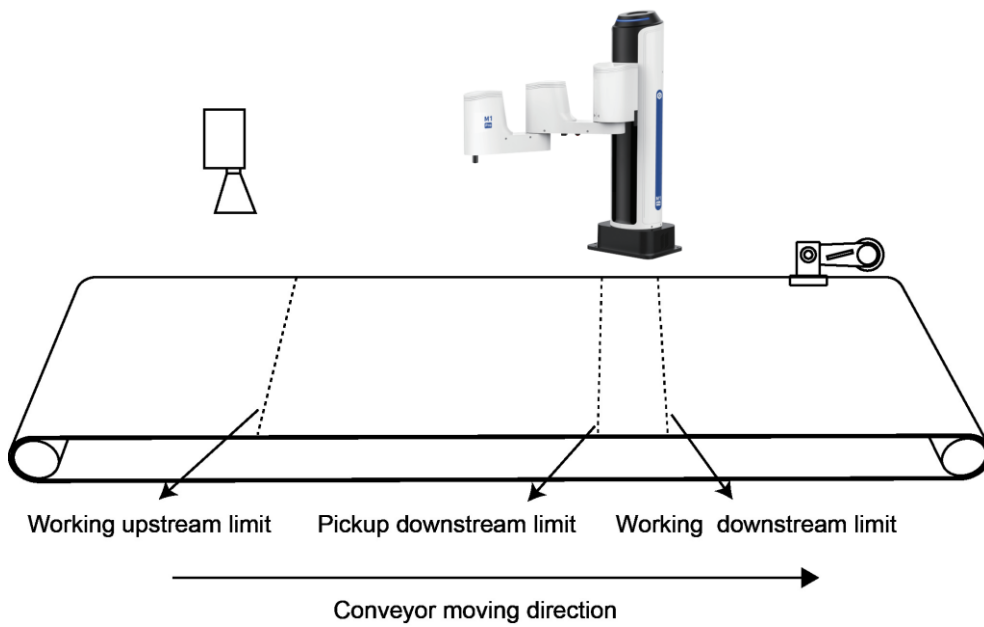


Figure 5.19 Border description

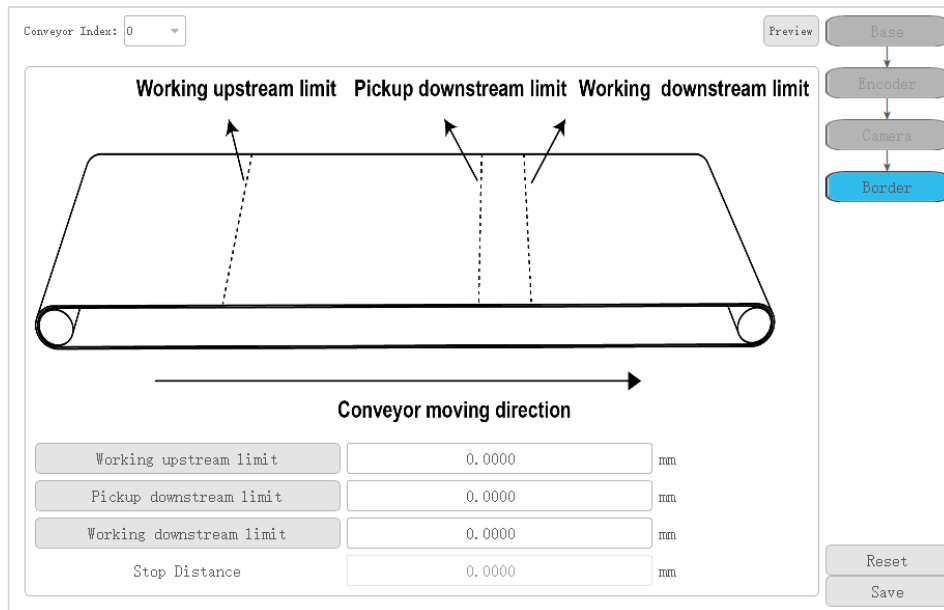


Figure 5.20 Border calibration page

Table 5.5 lists the border parameter description

Table 5.5 Border parameter description

Parameter	Description
Conveyor Index	Conveyor index: 0 or 1 Please select the right index according to the actual situation
Working upstream limit	Working upstream limit means that M1 Pro starts tracking at this position
Pickup downstream limit	When M1 Pro picks up the part, if the part moves out of the Pickup downstream limit, M1 Pro will not pick up it
Working downstream limit	If M1 Pro still tracks the parts at this position, an alarm will be triggered.
Stop distance	This parameter cannot be set

1. Jog the robot to the starting tracking position, then click **Working upstream limit** to obtain the working upstream limit.
2. Jog the robot to the latest starting tracking position that is expected to complete the process, then click **Working upstream limit** to obtain the working upstream limit.

Please set this parameter according to the conveyor speed and practical experience.

3. Jog the robot to the end tracking position, then click **Working downstream limit** to obtain the working downstream limit.

**Step 7** Click **Save**.

### 5.1.5 Example

#### 5.1.5.1 Pickup Example Using Vision Conveyor Tracking

In this application, we need to teach six points.

- Waiting point: P1
- Tracking point: P2
- Pickup point: P3
- Lifting point: P4
- Point above the placing point: P5
- Placing point: P6

If the pickup angle is required, you need to set this angle.

This topic takes the pickup application without angle requirement as an example, Figure 5.21 shows the taught positions.

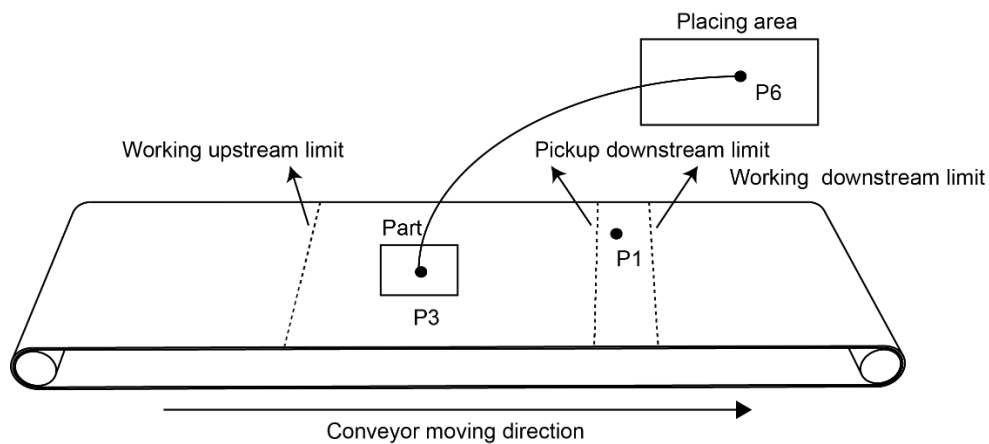


Figure 5.21 Taught position


#### Prerequisites

- The vision conveyor tracking project has been created on the DobotSCStudio. For details, please see *3.5.3.1 Creating Project*.
- The robot has been connected to the air pump.
- The end effector has been mounted on the end of the robot.
- The vision software has been installed.
- (Optional) If an eccentric end effector has been mounted, please set the Tool coordinate system. For details, please see *3.6.2 Setting Tool Coordinate System*.



- Vision software depends on the vision brand. The details on how to set and create a template are not described in this topic.
- You need to teach P2, P3 and P4 points under the set User coordinate system. If an eccentric end effector is used, you need to set the Tool coordinate system and then teach P2, P3 P4 points under the set Tool coordinate system.
- The J3 angle of P1 point is recommended to set to 120°, and the J4 angle is recommended to set to 0°.

## Procedure

**Step 1** Jog the robot to the point P1 under the basic coordinate system and click  on the **TeachPoint** page to add the saved point information.

### NOTICE

If the eccentric end effector is mounted, please execute **Step 2 - Step 3** under the set Tool coordinate system.

**Step 2** Jog the robot to a right height (which is higher than the part) under the set User coordinate system and record the height of the point P2 which is called **z1**.

**Step 3** Jog the robot to the center of the part under the set User coordinate system and record the height of the point P3 which is called **z0**.

**Step 4** Add P2, P3 and P4 points on the **TeachPoint** page under the basic coordinate system.  
 $P2=(0,0,z1,0)$ ,  $P3=(0,0,z0,0)$ ,  $P4=(0,0,z1,0)$

**Step 5** Put a part on the conveyor which is in the vision search area and create the template on the vision software to extract the part features.

Please set the current template angle to 0° .

### NOTE

If you use the sensor to detect parts, please place the parts in the correct direction (The parts detected by the sensor on the conveyor have a fixed direction)

**Step 6** (Optional) This step is executed only when the pickup angle is required. If there is no requirement, please skip this step.


1. Move the conveyor to a position where the part is in the pickup area, jog the robot to the center of this part, and set the R-axis to 0° under the basic coordinate system.
2. Switch to the set User coordinate system and record the R-axis value which is called **r1**.
3. Adjust the R-axis to a right pose to pick up the part, and record the R-axis value which is called **r2**.


Therefore, the R-axis value **r** of P2, P3 and P4 points is **r2 – r1**.

- $r > 180^\circ$ ;  $r = r - 360^\circ$
- $r < -180^\circ$ ;  $r = r + 360^\circ$



- $-180 \leq r \leq 180^\circ$ ,  $r$  remains unchanged
4. Modify the P2, P3 and P4 points on the **TeachPoint** page.  $P2=(0,0,z1,r)$ ,  $P3=(0,0,z0,r)$ ,  $P4=(0,0,z1,r)$

**Step 7** Jog the robot to point P6 under the basic coordinate system, and click  on the **TeachPoint** page to add the saved point information.

**Step 8** Jog the robot to point P5 under the basic coordinate system, and click  on the **TeachPoint** page to add the saved point information.

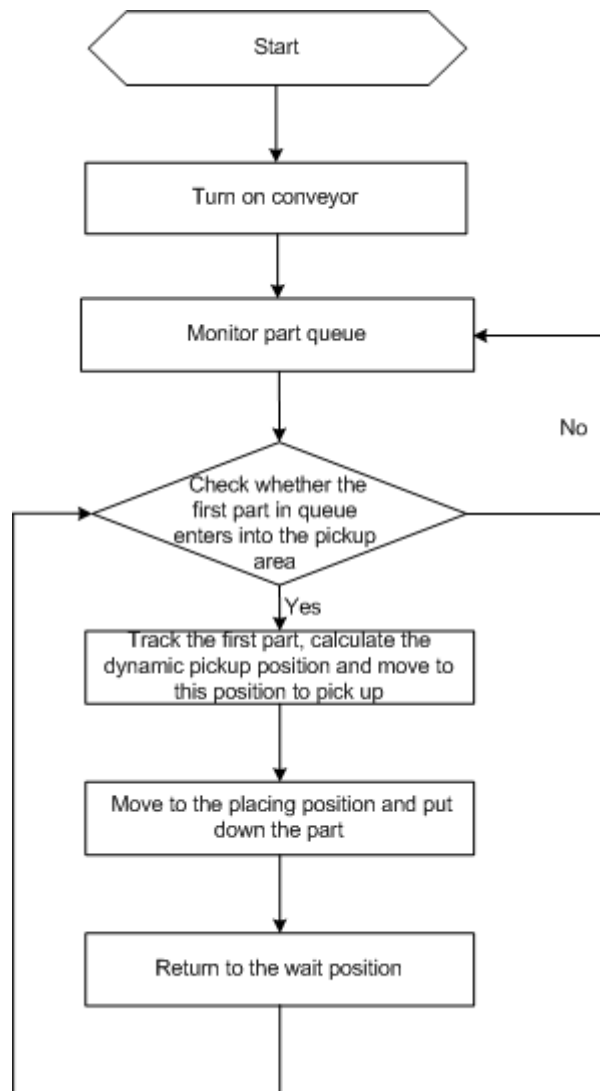


Figure 5.22 Conveyor tracking process



- The motion commands used between SyncCnv(0) and StopSyncCnv() only support Move command
- The wait, DI and DO commands are supported between SyncCnv(0) and StopSyncCnv().

#### Program 5.1 Conveyor tracking program

```
CnvVison(0) // Activate conveyor
DO(1,0) // Control the air pump status by DO1 and DO2
DO(2,0)
local flag //Part flag
local typeObject /// Part type
local point = {0,0,0} //Part coordinate (X,Y,R)
while true do
  Go(P1,"Speed=100 Accel=100 SYNC=1") // Wait point
  print("Test")
  while true do
    flag,typeObject,point = GetCnvObject(0,0) //Check whether there is a part. If there is a part, exit this
loop
  if flag == true then
    break
  end
  Sleep(20)
end
SyncCnv(0) //Synchronize the conveyor and start to track
Move(P2,"SpeedS=100 AccelS=100") // Tracking point
Move(P3,"SpeedS=100 AccelS=100") // Pickup point
Wait(100)
DO(1,1) // Active air pump and pick up part
DO(2,1)
Wait(100)
Move(P4,"SpeedS=100 AccelS=100 SYNC=1") // Lifting point
StopSyncCnv() // Stop conveyor tracking
Sleep(20)
Go(P5,"Speed=100 Accel=100") // Point above the placing point
Go(P6,"Speed=100 Accel=100 SYNC=1") // Placing point
Sleep(1000)
DO(1,0) // Close air pump
```

```

DO(2,0,"SYNC=1")
Sleep(1000)
Go(P5,"SpeedS=100 AccelS=100")
end
    
```

## 5.2 Palletizing

### 5.2.1 Overview

In carrying applications, some parts are regularly arranged with uniform spacing and teaching part positions one by one results in a high error and poor efficiency. Palletizing process can resolve these problems.

A full palletizing process includes pallet parameters setting and pallet programming. After you set the pallet parameters on DobotSCStudio, the generated configuration file will be imported to the M1 Pro automatically, then you can write a pallet program by calling pallet API based on site requirements.

### 5.2.2 Setting Pallet

Pallet parameter settings include basic parameter setting and path point setting. Basic parameter setting is to set pallet name, stack number, palletizing direction and stack spacing. Path points are the configured points on the assembly path or dismantling path.

- Transition point (point A): A point the robot must move to when assembling or dismantling stacks, which is fixed or varies with the pallet layer.
- Preparation point (point B): A point calculated by the target point and the set offset.
- Target point (point C): The first stack point.

Figure 5.23 and Figure 5.24 show the assembly path and dismantling path.

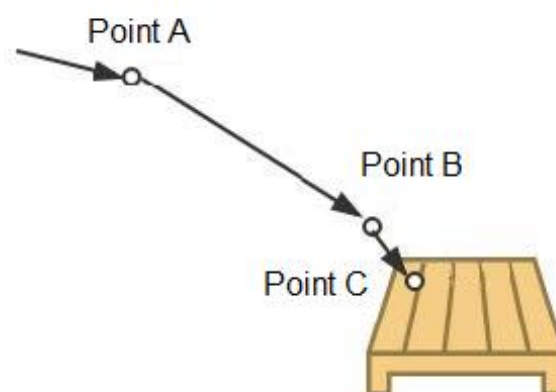


Figure 5.23 Assembly path

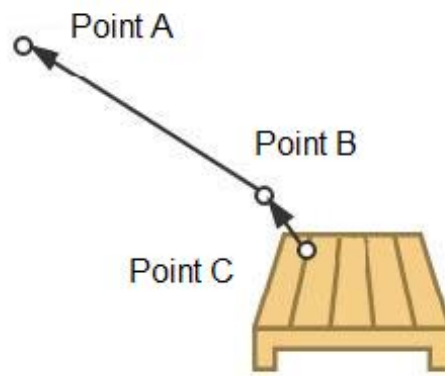


Figure 5.24 Dismantling path

Stack indicates parts or products to be carried. Pallet indicates an object which places the stacks. Assembling stack indicates that the robot places stacks to the pallet as the configured pallet type. Dismantling stack indicates that the robot takes out stacks from the pallet as the configured pallet type. Pallet type indicates the layout of all stacks placed on the pallet. In our M1 Pro, only the matrix pallet is supported, on which the stacks are placed in regular intervals, as shown in Figure 5.25.

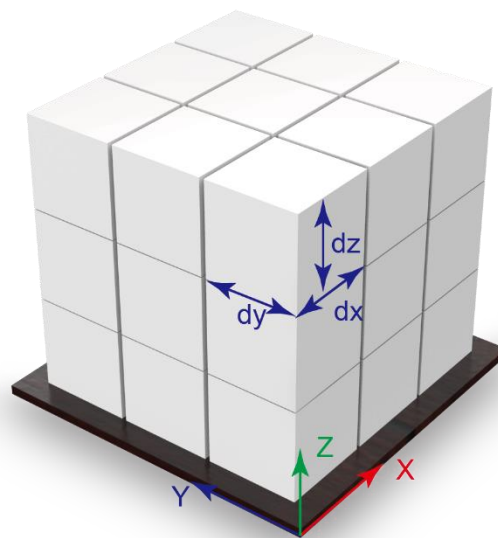


Figure 5.25 Matrix pallet

In this topic, we describe how to set pallet parameters. The 10 types of pallets can be defined.

### Prerequisites

- The robot has been powered on.

- The suction cup or gripper kit has been mounted on the robot
- (Optional) The User coordinate system has been set on the pallet. When teaching positions, you can select the set User coordinate system based on site requirements.
- The M1 Pro has been enabled.
- The user's authority is programmer authority or higher authority.

### Procedure

#### Step 1 Click **Process** > **MatrixPallet**.

The pallet page is displayed, as shown in Figure 5.26.



Figure 5.26 Pallet page

#### Step 2 Set the basic pallet parameters on the **Base** tab.

Table 5.6 shows the basic pallet parameter description.

Table 5.6 Basic pallet parameter description

Parameter	Description
Name	Pallet name
Direction	Palletizing direction Value: X->Y->Z or Y->X->Z In this topic, we select <b>X-&gt;Y-&gt;Z</b>
Count	Number of stacks in X, Y, Z direction respectively
Distance	Stack interval in X, Y, Z direction respectively

**Step 3** Jog the robot to the first stack position and click **Get Pose** on the **1<sup>st</sup> Pallet** tab, as shown in Figure 5.27.

**UserCoord** is the User coordinate system index, which needs to be consistent with the User coordinate system selected during teaching.

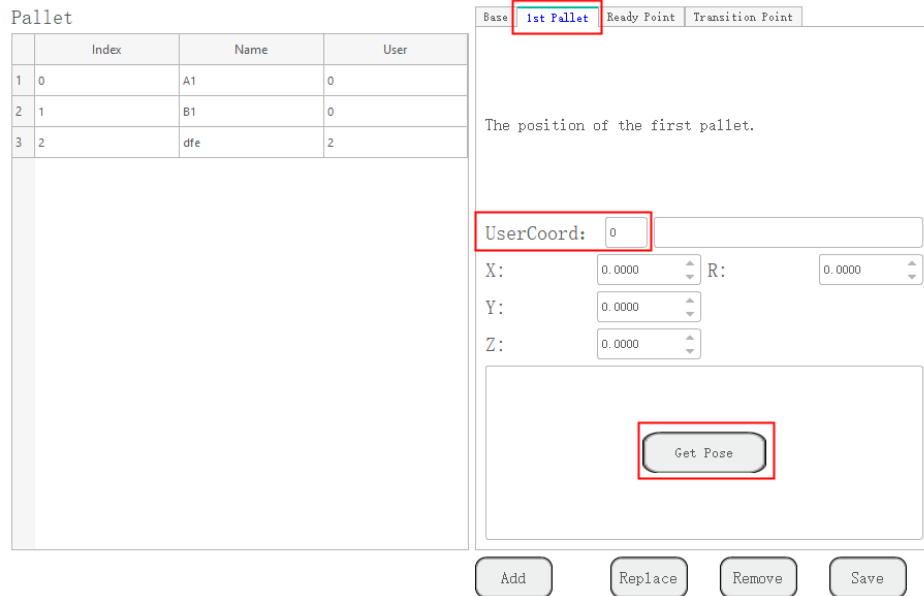


Figure 5.27 Teach the first stack position

**Step 4** Jog the robot to the transition point and click **Get Pose** on the **Transition Point** tab, as shown in Figure 5.28.

**UserCoord** is the User coordinate system index, which needs to be consistent with the User coordinate system selected during teaching.

If **Variation with layer height** is selected, the transition point is varied with the pallet layer. If not, it is the fixed point.

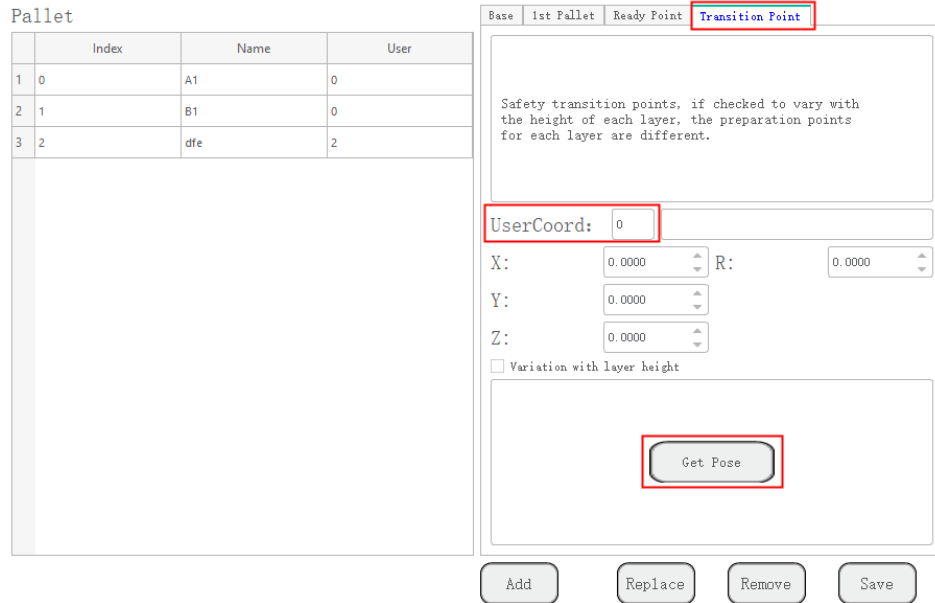


Figure 5.28 Teach the transition point

**Step 5** Jog the robot to the position where is above the first stack, and click **Get Pose** on the **Ready Point** tab.

**UserCoord** is the User coordinate system index, which needs to be consistent with the User coordinate system selected during teaching.

**Step 6** Click **Add** to generate the configuration file and import to the M1 Pro automatically.

### 5.2.3 Example

After setting the pallet parameters, you can call pallet API for programming. This topic takes stack assembly as an example to describe.

Program 5.2 Stack assembly demo

```

local MPpick = MatrixPallet(0,1, "IsUnstack=true Userframe=8")           // Define the pallet instance
Reset(MPpick)                                                            // Initial the pallet instance
while true do
    MoveIn(MPpick,"velAB=90 velBC=50")                                   // Start to assemble
    MoveOut(MPpick)
    result=IsDone(MPpick)
    if (result == true)                                                 // Check whether stack assembly is
complete
    then
        print("EXIT ...")
        break
    end
end
    
```

```
end
```

```
Release(MPpick)
```



## 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	<ol style="list-style-type: none"> <li>1. Check whether the motor parameter matches the motor model in nameplate;</li> <li>2. Check whether the motor and driver match, otherwise, select the right motor and driver</li> </ol>
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	<ol style="list-style-type: none"> <li>1. Check whether the same function is assigned to different DI's</li> <li>2. Confirm whether the corresponding MCU supports the assigned functionality</li> </ol>
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

ID	Level	Description	Solution
			support 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

ID	Level	Description	Solution
			support 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

ID	Level	Description	Solution
36096	5	No input parameters for Go instruction	Please enter parameters
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

ID	Level	Description	Solution
45056	5	Circle3 Option Error	Enter the correct parameters
45057	5	Jump Option Error	Enter the correct parameters
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

ID	Level	Description	Solution
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
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

ID	Level	Description	Solution
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
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



# M1Pro gripper user manual

**(This document works with PGC series  
collaborative parallel grippers, only use for M1pro)**

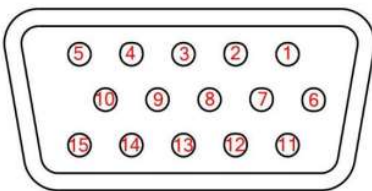
**Ver1.0\_20210818**

# Introduction

# M1Pro and DB15 Introduction



DB15 interface description



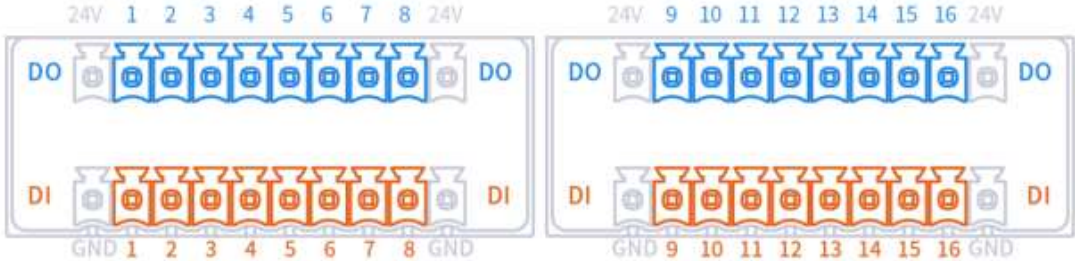
No.	1	2	3	4	5	6	7	8
Description	24V	DO17	DO18	DO19	DO20	unused	unused	unused
No.	9	10	11	12	13	14	15	
Description	RS485 A	RS485 B	DI17	DI18	DI19	DI20	0V	

DOBOT M1 Pro



# M1Pro Digital Output Introduction

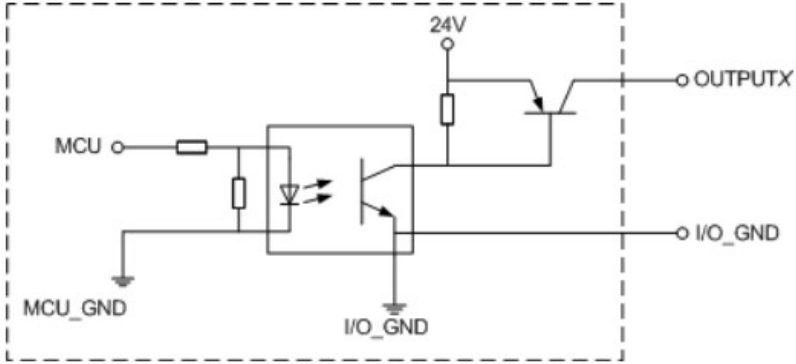
I/O interface



Technical specifications

Item	Specification
Output channel	16 channels
Connection method	Europe type terminal
Output type	<b>PNP</b>
Power supply (DC)	24V ±10%
Load current of single channel	500mA
Output current	2A
Isolation method	Magnetic isolation

Simple digital output circuit



# Input/output Commands of M1Pro Introduction

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

Function	<i>DI(index)</i>
Description	Get the status of the digital input port
Parameter	index: Digital input index. Value range: 1 - 16
Return	<ul style="list-style-type: none"> <li>When an index is set in the DI function, <b>DI(index)</b> returns the status (ON/OFF) of this specified input port</li> <li>When there is no index in the DI function, <b>DI()</b> returns the status of all the input ports, which are saved in a table</li> </ul> <p>For example, local di=(), the saving format is {num = 24 value = {0x55, 0xAA, 0x52}}, you can obtain the status of the specified input port with <b>di.num</b> and <b>di.value[n]</b></p>
Example	<pre>if(DI(1))=ON then Move(P1) end</pre> <p>The robot moves to point P1 when the status of the digital input port <b>1</b> is <b>ON</b></p>

Function	<i>DOExecute(index, ON   OFF)</i>
Description	Set the status of digital output port (Immediate command)
Parameter	<ul style="list-style-type: none"> <li>index: Digital output index. Value range: 1 - 24</li> <li>ON/OFF: Status of the digital output port. ON: High level; OFF: Low level</li> </ul>
Example	<pre>DOExecute(1,OFF)</pre> <p>Set the status of the digital output port 1 to <b>OFF</b></p>

Function	<i>DO(index, ON   OFF)</i>
Description	Set the status of digital output port (Queue command)
Parameter	<ul style="list-style-type: none"> <li>index: Digital output index. Value range: 1 - 24</li> <li>ON/OFF: Status of the digital output port. ON: High level; OFF: Low level</li> </ul>
Example	<pre>DO(1,ON)</pre> <p>Set the status of the digital output port 1 to <b>ON</b></p>

# PGC-140 Introduction

## Indicator color description of PGC-140:

- (1)Uninitialized state: the red light is flashing, and the other lights are off.
- (2)Initialization completed state: the blue light is always on, indicating that it is in an operable state.
- (3)Receive command status: the red light flashes once quickly (because the blue light is always on at this time, the gripper indicator light will appear purple)
- (4)Clamped object status: the green light is always on, and the other lights are off.
- (5)Object falling status: the green light flashes.

# PGC-140

Collaborative parallel grippers



Allowable vertical load (static)	
Fz:	300 N
Allowable moment (static)	
Mx:	7 N·m
My:	7 N·m
Mz:	7 N·m

## DH-ROBOTICS

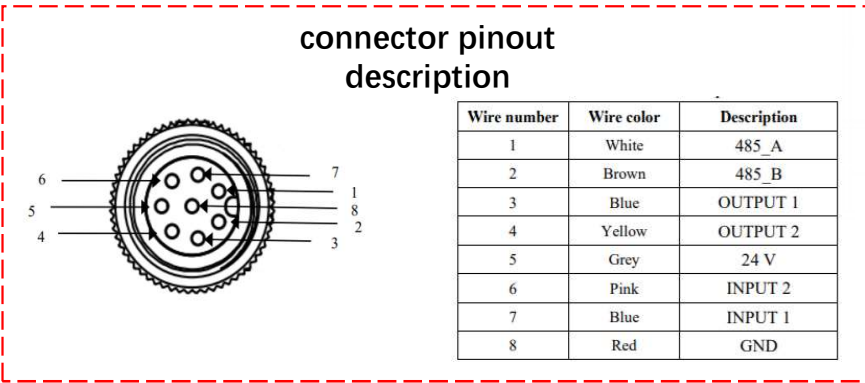
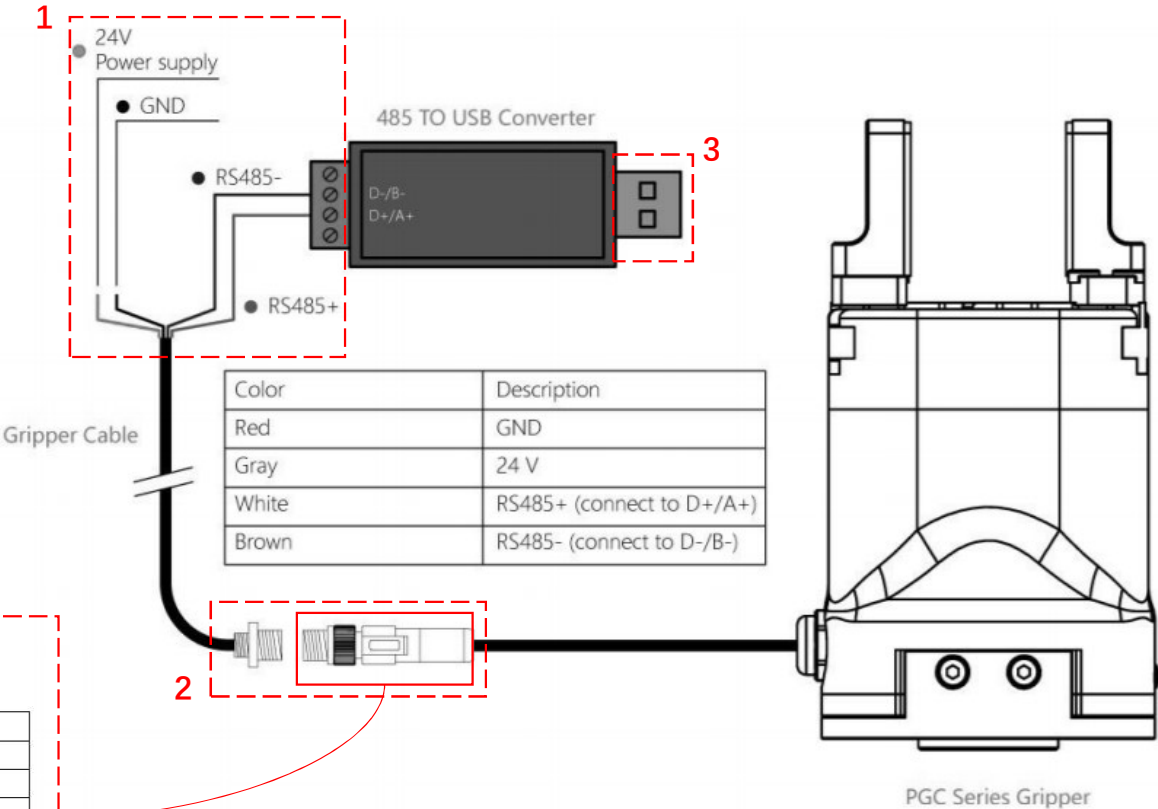
Gripping force(per jaw)	Opening/closing stroke(both sides)
<b>40~140 N</b>	<b>50 mm</b>

Mechanical specifications	
Position repeatability (both sides)	± 0.03 mm
Finger backlash	Unilateral 0.2mm or less
Opening/closing time	0.6 s/0.6 s
Driving method	Rack and pinion + T-slot guidance
	1 kg
Noise emission	< 50 dB

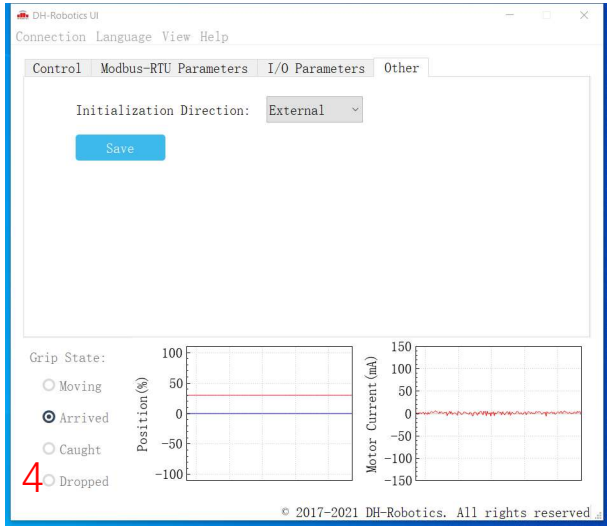
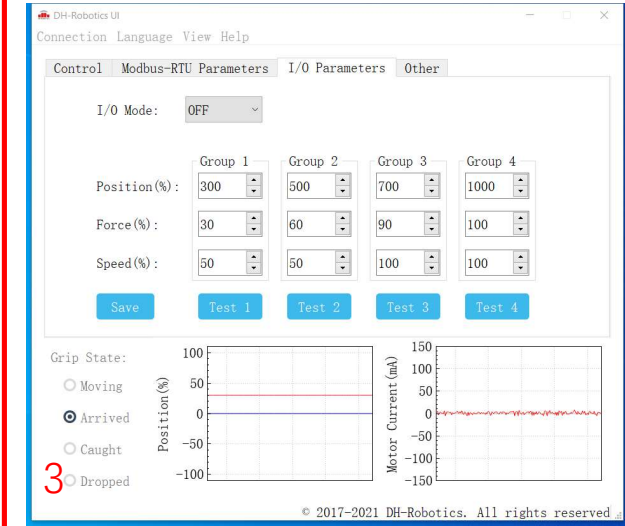
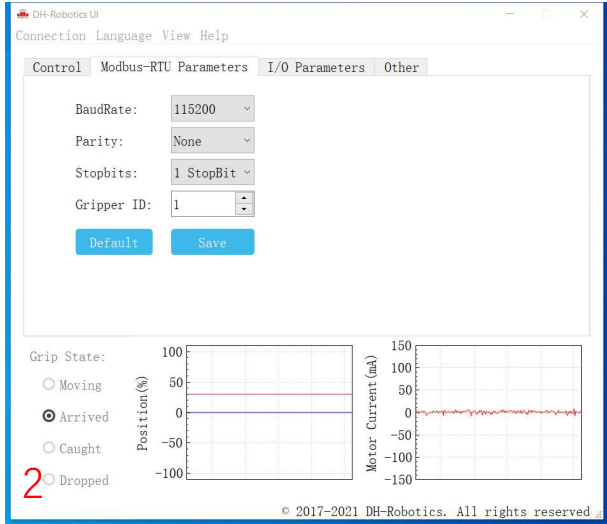
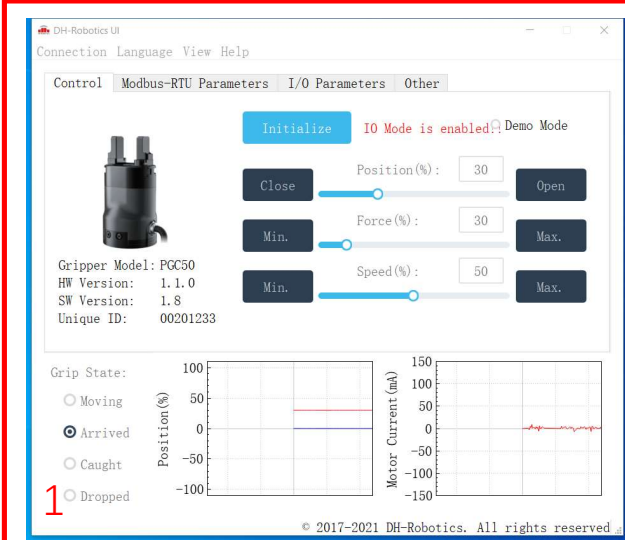
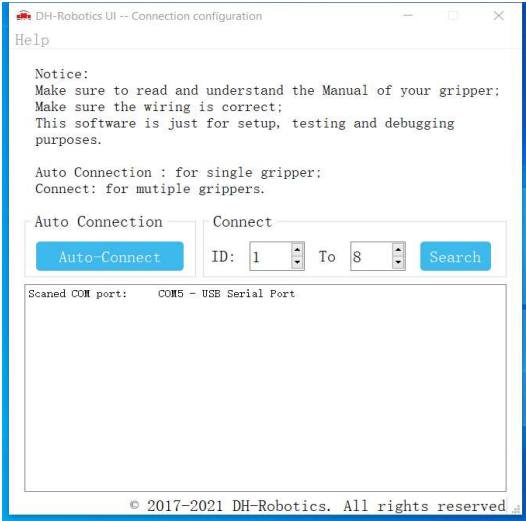
Electrical specifications	
Communication interface	Standard: Modbus RTU (RS485), Digital I/O Optional: TCP/IP, USB2.0, CAN2.0A, PROFINET, EtherCAT
Nominal voltage	24 V DC ± 10%
Nominal current	0.4 A
Peak current	1 A
Ingress protection rating	IP 67
Recommended operating environment	0~40°C, 85% RH under environment

Settings

1. Wire the Gripper cable according to the color( 24V, GND, RS485+, RS485-),see figure1.
2. Use the gripper cable connect to gripper connector, see figure1.
3. Connect the RS-485 to USB converter usb port to the computer, ,see figure3.



4. Open the DH configuration software as below, click Auto-Connect, you will enter the setting page(1,2,3,4).



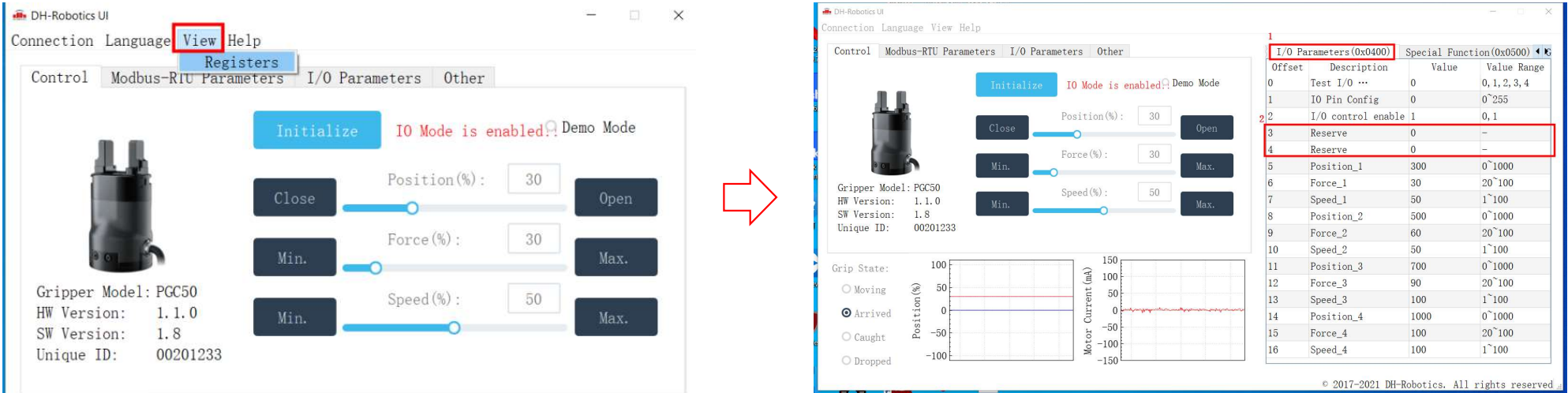


5. Change the IO mode from OFF to ON, so that you can use the IO signal to control the gripper.



**Note:**  
 Position value of 300, 500, 700, 1000 corresponding to 30%, 50%, 70%, 100%.  
 Please pay attention to the corresponding value.

6. Open the View menu, click Registers, choose the I/O Parameters(0x0400), the No.3 and No.4 use for switch the signal between NPN and PNP,

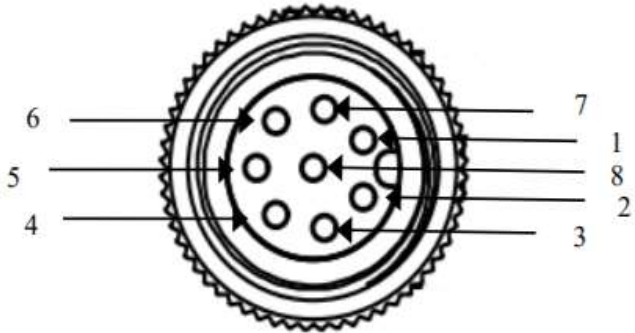


Application



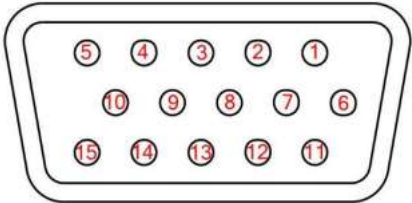
Pls refer to the following interface description, weld the gripper extending cable to the DB15,

PGC-140 connector pinout description



Wire number	Wire color	Description
1	White	485_A
2	Brown	485_B
3	Blue	OUTPUT 1
4	Yellow	OUTPUT 2
5	Grey	24 V
6	Pink	INPUT 2
7	Blue	INPUT 1
8	Red	GND

DB15 interface description



No.	1	2	3	4	5	6	7	8
Description	24V	DO17	DO18	DO19	DO20	unused	unused	unused
No.	9	10	11	12	13	14	15	
Description	RS485 A	RS485 B	DI17	DI18	DI19	DI20	0V	

I/O Control The I/O mode is a common control method in industry. The grippers will monitor the pin states of Input 1 and Input 2 (0V and high resistance states). For these two pins, there will be four logic states:00,01,10,11. You can control this gripper through changing the states of Input 1 and Input 2.

INPUT 1	INPUT 2	Pin state	I/O state	Perform action
High resistance	High resistance	0 0	Group 1	Target position 1,target force 1,target speed 1
0V	High resistance	1 0	Group 2	Target position 2,Target Force 2,Target Speed 2
High Resistance	0V	0 1	Group 3	Target position 3,Target Force 3,Target Speed 3
0V	0V	1 1	Group 4	Target position 4,Target force 4,target speed 4

You can also get the gripper state by detecting the states of Output1 and Output 2(0V and high resistance states).

I/O State (OUT1 OUT2)	State description
0 0	Fingers are in motion
1 0	Fingers are at reference position, No object detected or object has been dropped
0 1	Fingers have stopped due to an object detection

NOTE: Please make sure that the I/O hardware type of the gripper is compatible with your controller's. M1Pro output type is PNP, So your gripper need to support the PNP input signal.