DOBOT CR5

User Guide

Issue: V 3.5.3.1

Date: 2020-05-09

Shenzhen 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

Preface

Purpose

This Document describes the functions, technical specifications, installation guide and system commissioning of DOBOT CR5 robot system, 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
2020/01/16	The first release

Symbol Conventions

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

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

Contents

1. 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
⚠WARNNING	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.

⚠DANGER

- 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 controller is degraded. If turning on or off the power continually is required, please keep frequency less than once a minute.

⚠HOT

- The robot and the controller will generate heat during operation. Please do not operate or touch the robot when the robot is working or has just stopped working.
- Turn off the power and wait an hour for the robot to cool down.
- Do not put your fingers where the control cabinet gets hot.

⚠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.
- Handling operations such as lifting rings and driving need to use appropriate and reliable lifting equipment. According to the relevant regulations of various countries, it must be carried out by personnel with operating qualification certificates or personnel authorized by the company.
- Please make sure that there are no obstacles within 2 meters of the robot during transportation, and relevant personnel should stay away from the suspended robot.
- 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.
- Please make sure that there are no obstacles around the packing area, so that the staff can leave in a timely manner.
- 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 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.
- The personnel responsible for installation, operation, and maintenance of the equipment must first undergo strict training, understand various safety precautions, and master the correct operation and maintenance methods before operating and maintaining the equipment.
- When an abnormality occurs in the mechanical arm, it is necessary to ensure that the machine is stopped and then checked.
- After the commissioning of the operator is completed, the test needs to be performed in the Manual mode first, and then it is automatically run after it is confirmed to be correct.

- 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 observe the ESD regulations when disassembling the robot or controller.
- Avoid dismantling the power supply system in the controller. After the controller is turned off, its power supply system may still have high voltage for several hours.
- 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.

⚠WARNING

- 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.
- All the required cables must be connected before powering on the equipment.
- 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 do not plug or unplug the power cables or communication cables when

equipment is normally operated.

- 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 controller.
- Please do not reach out into the workspace of the robot when operating it, otherwise, it will be vulnerable to injury the device or the person.
- Even if the power switch of the controller 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 controller.
- Please do not reach out into the workspace of the robot when operating it, otherwise, it will be vulnerable to injury the device or the person.
- 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

The collaborative robot work system is composed of the collaborative robot body, robot control software, and robot operation software. DOBOT CR5 supports APP wireless direct connection, which is really simple and easy to use. With the self-developed dynamic algorithm, one-handed teach-in and sensor less collision detection are realized to ensure the safety of human and machine working together. The drag trajectory reproduction function is creatively introduced, which completely reproduces the drag trajectory and reduces the threshold for robot use. DOBOT CR5 has a repeat positioning accuracy of ± 0.03mm, a max load of 5kg, and a joint speed of 180 °/s. It is a product with the advantages of both industrial robots and collaborative robots.

Figure 2.1 Robot system

2.1 Technical Specifications

2.1.1 Robot Body Technical Parameters

Table 2.1 Robot body technical parameters

Product	DOBOT CR5
Weight	25kg
Max load(kg)	5kg
Reach(mm)	900mm
Rated current	7.3A

Rated voltage	DC 48V	
Maximum Speed of End-effector	3m/s	
Motion Range (°)	J1	±360°
	J2	±360°
	J3	±160°
	J4	±360°
	J5	±360°
	J6	±360°
Joint Maximum Speed(°/s)	J1	180°/s
	J2	180°/s
	J3	180°/s
	J4	180°/s
	J5	180°/s
	J6	180°/s
End-effector I/O Interface	DI	2
	DO	2
	AI	2
	AO	0
Communication Interface	RS485	
Controller I/O Interface	DI	16
	DO/DI	16
	AI	2
	AO	2
	Incremental Encoder ABZ Input	1
Repeatability	±0.03mm	
Communication	TCP/IP, Modbus, EtherCAT, WiFi	
IP Classification	IP54	
Temperature	0℃~45℃	
Power Consumption	150W	

Material	Aluminum alloy, ABS

2.1.2 Controller Technical Parameters

Table 2.2 Controller technical parameters

Product	DOBOT CC161
Axis control	6 axes + external axes
Input power	1 PHASE AC 100V~240V AC, 47HZ~63HZ
Output power	48V DC, MAX 20A
Communication interface	EtherCAT (used for external axis), Ethernet
I/O interface	• 16 Digital Inputs • 16 Digital Inputs/Outputs (Multiplexing) • 2 Analog Outputs (Voltage: 0V-10V, Current: 4mA-20mA) • 2 Analog Inputs (Voltage: 0V-10V, Current: 4mA-20mA) • 1 Incremental Encoder ABZ Input
Teaching method	APP
Programming Language	Script/Tree Programming/Graphical Programming
Installation Method	Floor mounting
Environment	Temperature: 0℃ - 45℃ Humidity: ≤95%, and no condensation
Protection Grade	IP20
Cooling Method	Forced air cooling
Safety Function	Emergency stop function and reserved external safety interface (can be controlled by I/O interface) Protective stop interface, Automatic Operation Remote Confirmation
Indicator Status	The indicator light will be steady red when the power is on; the indicator light will be off when the power is off
Maintenance	• Diagnostic software • Power-off Zero Save • Reserved remote service

2.2 Robot Workspace

Figure 2.2 shows the workspace of CR5 robot.

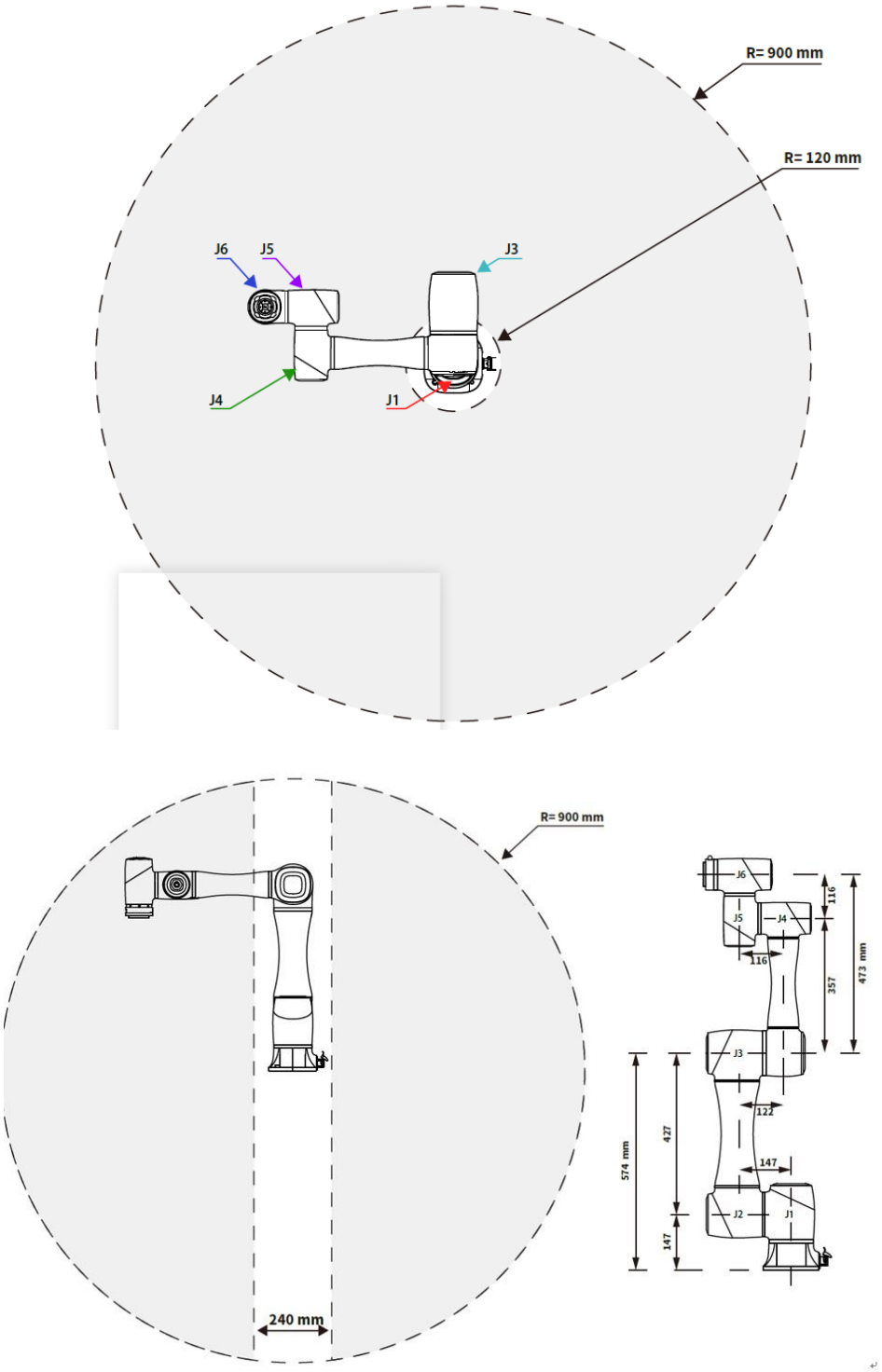


Figure 2.2 CR5 robot workspace

2.3 End-effector Size

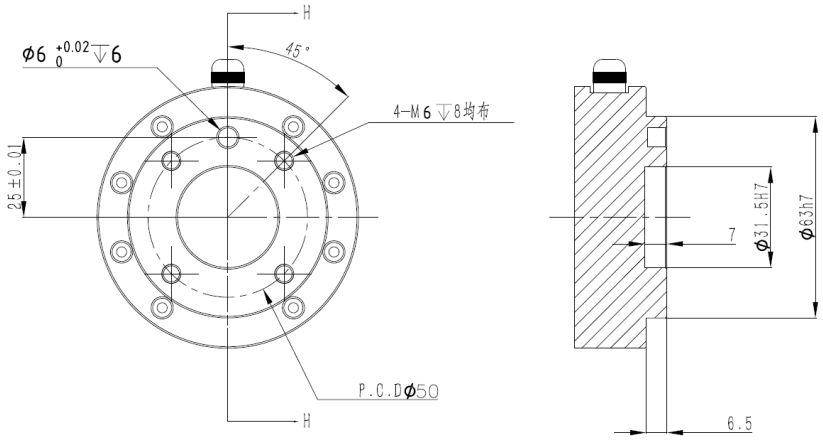


Figure 2.3 End-effector Size

2.4 End-effector Load Description

The robot actuator can bear the load of the cylinder whose center of mass is located at an axial distance of LD80mm from the center of the End -effector and a radial distance of LR60mm and no more than 5kg. As shown in Figure 2.4.

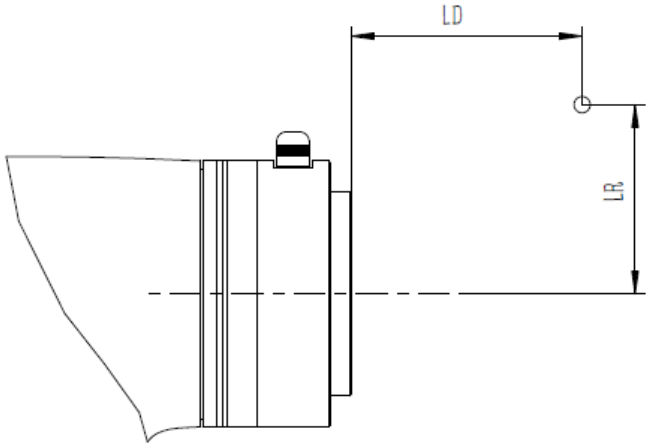


Figure 2.4 End-effector load diagram

2.5 Product Features

2.5.1 Motion Function

The motion trajectory consists of a series of interpolated motions since the interpolated motion is the basic motion type. According to the different trajectories, motion functions are classified as joint interpolated motion, linearly interpolated motion, circular interpolated motion and continuous path. The joint interpolated motion is in the joint space. And the other interpolated motions are in the Cartesian space.

2.5.1.1 Joint Interpolated Motion

Joint interpolated motion includes **Go**, **MoveJ** modes.

- Go/MoveJ: From point A to point B, each joint will run from an initial angle to its target angle, regardless of the trajectory, as shown in Figure 2.5.

Figure 2.5 Go/MoveJ modes

2.5.1.2 Linearly Interpolated Motion

The joints will perform a straight line trajectory from point A to point B, as shown in Figure 2.6.

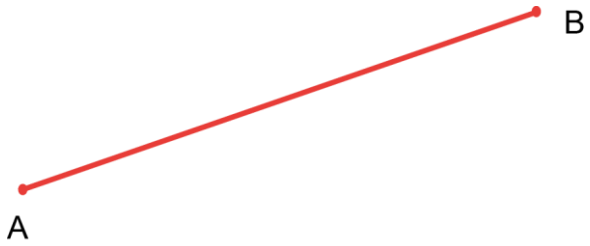


Figure 2.6 Move mode

- Jump: The trajectory looks like a door. From point A to point B, the robot will move in the **Move** mode
    1. Move up to the lifting height (**StartHeight** is a relative height).
    2. Move up to the maximum lifting height (**zLimit**).
    3. Move horizontally to a point that is above point **B**.
    4. Move down to a point where the height is point **B** plus the dropping height (**EndHeight** is a relative height).
    5. Move down to Point **B**.

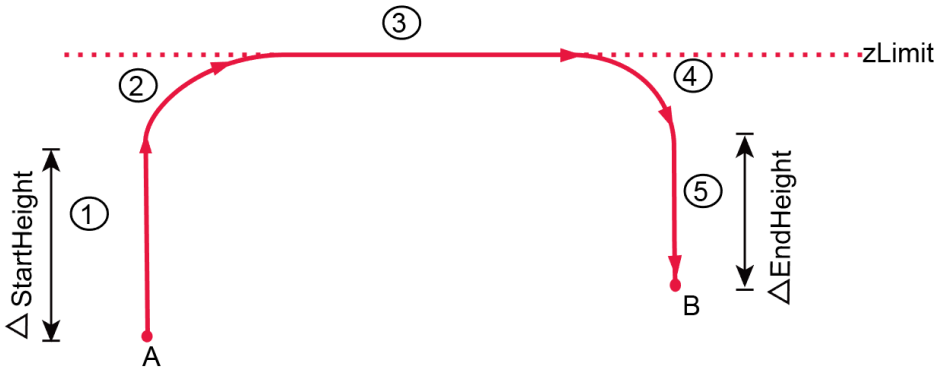


Figure 2.7 Jump mode

NOTICE

- Point **A** and point **B** cannot be higher than **zLimit**. Otherwise, an alarm will be triggered.
- If point **A** plus **StartHeight** or point **B** plus **EndHeight** is higher than **zLimit**, the robot moves up from point **A** to **zLimit** or moves down from **zLimit** to point **B** directly, the trajectory looks like a door without transition, as shown in Figure 2.8.

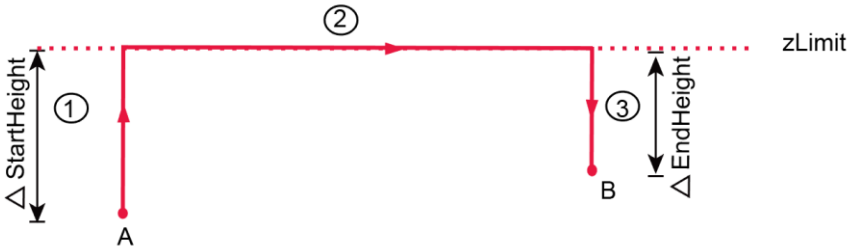


Figure 2.8 Jump mode (1)

- If the heights of point **A** and point **B** are the same with **zLimit**, the trajectory is shown in Figure 2.9.

Figure 2.9 Jump mode (2)

2.5.1.3 ARC (Circular Interpolated Motion)

The trajectory is an arc, which is determined by three points (the current point, any point and the end point on the arc), as shown in Figure 2.10.

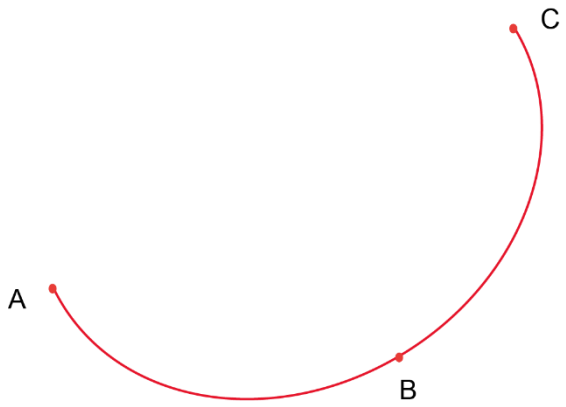


Figure 2.10 Arc trajectory

2.5.1.4 Circle (Circular Interpolated Motion))

The trajectory is a circle, which is determined by three points (the current point, any point and the end point on the arc) as well, as shown in Figure 2.11.

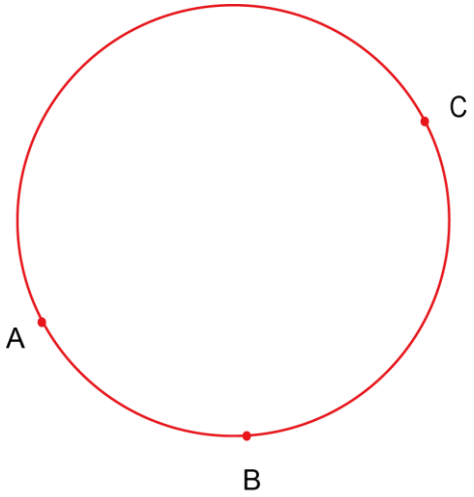


Figure 2.11 Circle trajectory

2.5.2 Coordinate System

This topic describes the coordinate systems for different types of robots, which are divided into Joint coordinate system, Base coordinate system, User coordinate system, and Tool coordinate system. The next three coordinate systems which are based on the right-handed rule are called the Cartesian coordinate system.

2.5.2.1 Joint Coordinate System

The Joint coordinate system is determined by the motion joints.

Figure 2.12 shows the Joint coordinate system of a CR5 robot. All the joints are rotating joints.

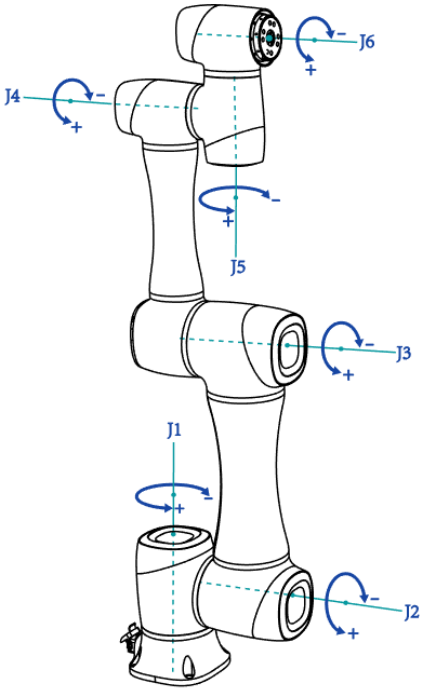


Figure 2.12 Joint coordinate of a CR5 robot

2.5.2.2 Base Coordinate System

The Base coordinate system is determined by the base.

Figure 2.13 shows the Base coordinate system of a CR5 robot. **Rx$_B$**, **Ry$_B$**, **Rz$_B$** are the orientation data, which are designated by rotating the tool center point (TCP) around the X, Y, Z axes under the Base coordinate system.

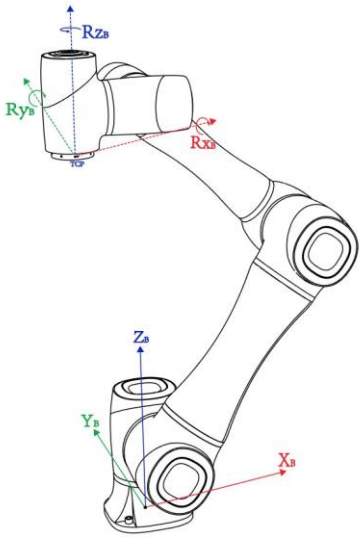


Figure 2.13 Base coordinate system of CR5 robot

2.5.2.3 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 2.14 shows the default Tool coordinate system of a CR5 robot. $\mathbf{R_{XT}}$, $\mathbf{R_{YT}}$, $\mathbf{R_{ZT}}$ are the orientation data, which are designated by rotating the tool center point (TCP) around the X, Y, Z axes under the default Tool coordinate system.

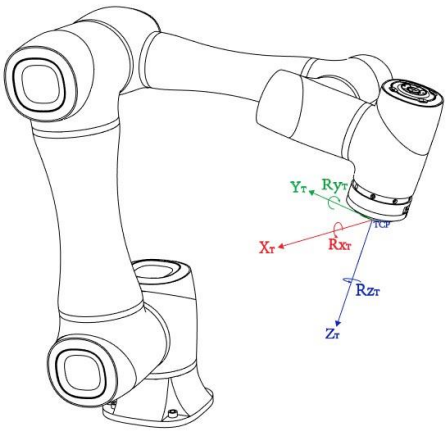


Figure 2.14 The default Tool coordinate system of CR5 robot

2.5.2.4 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.

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.

Figure 2.15 shows the default User coordinate system of a CR5 robot. $\mathbf{R_{XU}}$, $\mathbf{R_{YU}}$, $\mathbf{R_{ZU}}$ are the orientation data, which are designated by rotating the tool center point (TCP) around the X, Y, Z axes under the selected User coordinate system.

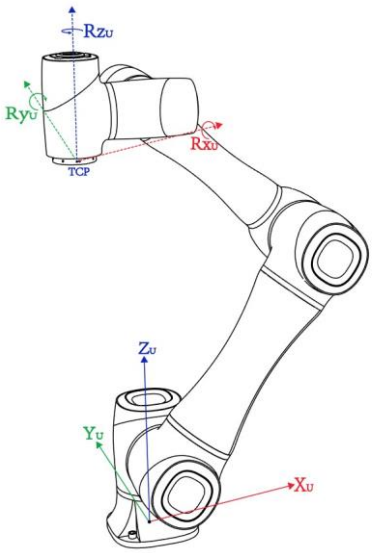


Figure 2.15 The default User coordinate system of CR5 robot

2.5.3 Arm Orientation

Table 2.3 lists the values of the R, D, N arm parameters and their right orientations.

Table 2.3 R, D, N identification

Arm parameter	1	-1
R	Righty hand orientation	Lefty hand orientation
D	Above elbow orientation	Below elbow orientation
N	No-flip wrist orientation	Flip wrist orientation

Besides, in this system, we use **Cfg** to represent the sixth-axis angle, as shown in Table 2.4.

Table 2.4 Cfg identification

Cfg	The sixth-axis angle
…	…
2	[90°,180°)
1	(0°,90°)
-1	[0°,-90°)
-2	[-90°,-180°)
…	…

2.5.4 Singularity Point

When the robot moves under the Cartesian coordinate system, the resultant velocity of the two axes cannot be in any direction if the directions of them are aligned, resulting in that the degrees of freedom of the robot are degraded. Namely, the robot moves to the singularity point and an alarm about singularity point is triggered. However, when the robot moves to the singularity point under the Joint coordinate system, the movement will not be affected. And also, the alarm will not be triggered.

There are three singularity points shown as follows.

- Wrist singularity point: The axes of J4 and J6 are aligned.

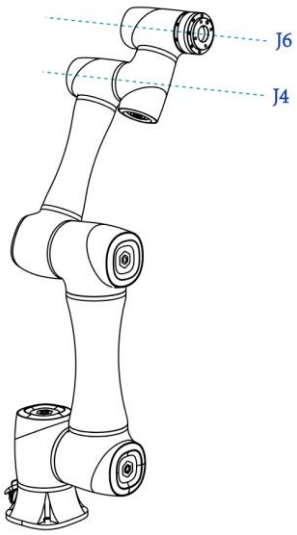


Figure 2.16 Wrist singularity point

- Shoulder singularity point: The TCP is located in the plane performed by Joint1 and Joint2.

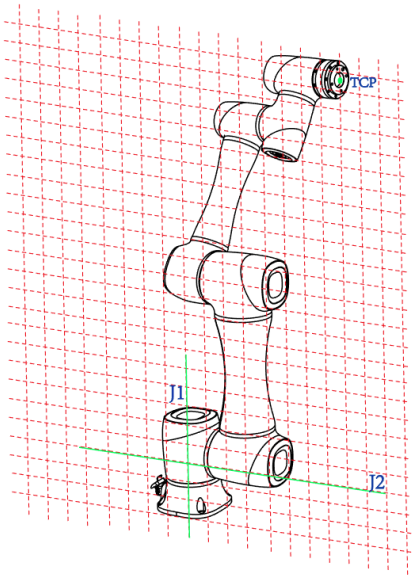


Figure 2.17 Shoulder singularity point

- Elbow singularity point: The Rear arm and Forearm in a straight line.

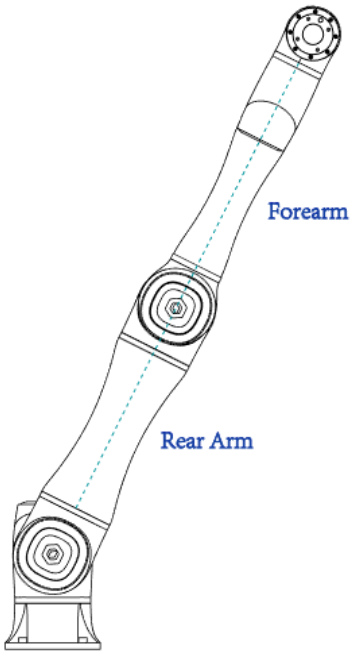


Figure 2.18 Elbow singularity point

2.5.5 Collision Detection

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

automatically when the robot arm hit an obstacle.

2.6 Key Description on Robot

There are function keys and LED indicator on the robot for user to manually control robot.

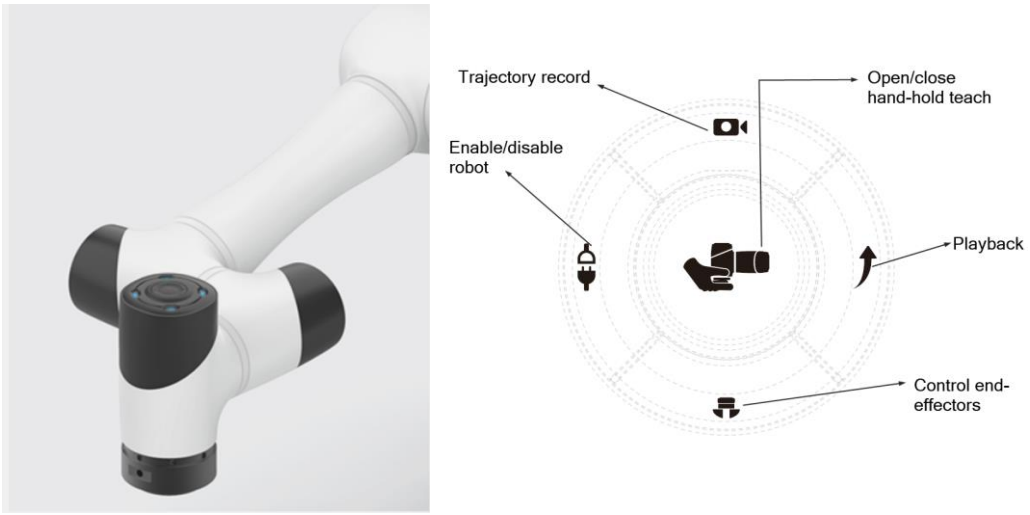


Figure 2.19 Keys on robot

Table 2.5 Key description

Key	Description
Open/close hand-hold teach	• Long press and then release: Open hand-hold teach, and the LED indicator turns blue and blinks • Short press and then release: Close hand-hold teach, and the LED indicator turns green
Trajectory record	• Long press and release: Open trajectory record, and the LED indicator turns yellow • Short press and release: Close trajectory record, and the LED indicator turns green
Playback	• Long press and then release: Open playback, and the LED indicator turns yellow and blinks • Short press and then release: Close playback, and the LED indicator turns green
Control end-effectors	Short press and then release: Open or close end-effectors
Enable/disable robot	Long press 3s and then release: Enable robot., the LED indicator turns green from blue and blinks Long press 6s and then release: Disable robot, the LED indicator turns blue from green and blink.

📖NOTE

- If there is an error on robot, the LED indicator turns red.
- If you switch the robot mode to auto mode on the APP, the LED indicator turns green and blinks.
- The connection between robot and he controller is abnormal, the LED indicator turns blue and blinks.

3. Electrical Specifications

3.1 Controller I/O Interface Description

A robot controller contains I/O interfaces, for connecting to external equipment, such as air pump, PLC, etc. These I/O interfaces provide 32 digital inputs, 16 digital outputs, 2 analog outputs, and 2 analog inputs, as shown in Figure 3.1.

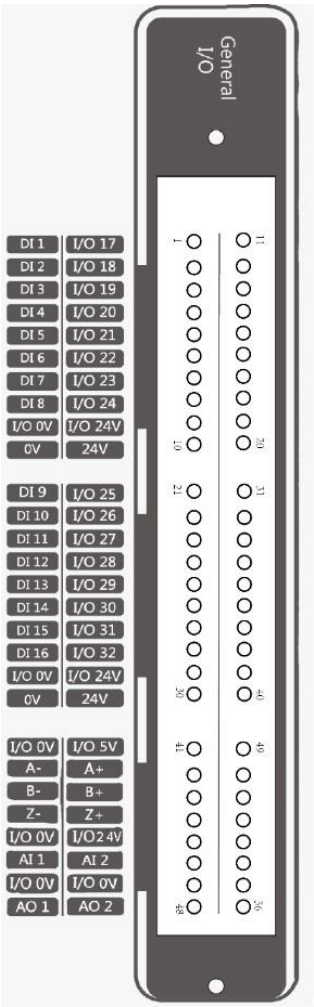


Figure 3.1 I/O interface

Table 3.1 PIN definition of digital output terminal J1

PIN	Name	Definition
1	DI1	Digital input 1
2	DI2	Digital input 2
3	DI3	Digital input 3

PIN	Name	Definition
4	DI4	Digital input 4
5	DI5	Digital input 5
6	DI6	Digital input 6
7	DI7	Digital input 7
8	DI8	Digital input 8
9	I/O 0V	I / O interface reference level
10	0V	Internal reference level
11	I/O17	Digital input 17 /output 1
12	I/O18	Digital input 18 /output 2
13	I/O19	Digital input 19 /output 3
14	I/O20	Digital input 20/output 4
15	I/O21	Digital input 21/output 5
16	I/O22	Digital input 22/output 6
17	I/O23	Digital input 23/output 7
18	I/O24	Digital input 24/output 8
19	I/O 24V	I / O interface 24V power input
20	24V	Internal 24V power output
21	DI9	Digital input 9
22	DI10	Digital input 10
23	DI11	Digital input 11
24	DI12	Digital input 12
25	DI13	Digital input 13
26	DI14	Digital input 14
27	DI15	Digital input 15
28	DI16	Digital input 16
29	I/O 0V	I / O interface reference level
30	0V	Internal reference level
31	I/O25	Digital input 25/output 9
32	I/O26	Digital input 26/output 10
33	I/O27	Digital input 27/output 11

PIN	Name	Definition
34	I/O28	Digital input 28/output 12
35	I/O29	Digital input 29/output 13
36	I/O30	Digital input 30/output 14
37	I/O31	Digital input 31/output 15
38	I/O32	Digital input 32/output 16
39	I/O 24V	I / O interface 24V power input
40	24V	Internal 24V power output
41	I/O 0V	I/O interface reference level
42	A-	Incremental Encoder ABZ Input
43	B-	
44	Z-	
50	A+	
51	B+	
52	Z+	
45	I/O 0V	I / O interface reference level
46	AI 1	Analog input 1
47	I/O 0V	I / O interface reference level
48	AO 1	Analog output 1
49	I/O 5V	Internal 5V power output
53	I/O 24V	I / O interface 24V power input / output (in the case of other I / O 24V input, this interface can be used as an output to power the analog module)
54	AI 2	Analog input 2
55	I/O 0V	I / O interface reference level
56	AO 2	Analog output 2

3.2 Emergency Stop I/O description

3.2.1 User Emergency Stop

User emergency stop I / O is an emergency stop interface provided to users, and users can

connect external emergency stop devices. The interface description is as follows:

Table 3.2 User emergency stop I/O description

Output Interface	Input Interface	Factory connection	Description
I/O28（X18:4）	DI12（X19:4）	Short connection	I/O28 and DI12, I/O29 and DI13 are redundant circuits of the user emergency stop interface. Any one of the disconnections will cause the user emergency stop function to be triggered
I/O29（X18:5）	DI13（X19:5）	Short connection	

3.2.2 Protective Stop

Protective stop I/O is a protective stop interface provided to users, and users can connect external protective stop devices. The interface description is as follows:

Table 3.3 Protective stop I/O description

Output Interface	Input Interface	Factory connection	Description
I/O31（X18:7）	DI15（X18:7）	Short connection	I/O31 and DI15, I/O32 and DI16 are redundant circuits of the protective stop interface，Any one of the disconnections will cause the protective stop function to be triggered
I/O32（X18:8）	DI16（X19:8）	Short connection	

3.2.3 Automatic Operation Remote Confirmation

The automatic operation remote confirmation interface is an interface for users to access the automatic operation remote confirmation button. The main function is that the user can manually confirm the operation at the remote before automatic operation to ensure the safety of personnel. The interface description is as follows:

Table 3.4 Automatic Operation Remote Confirmation I/O description

Output Interface	Input Interface	Factory connection	Description
I/O27（X18:3）	DI11（X19:3）	Short connection	I/O27 and DI11 are one circuits of automatic remote confirmation interface, users can confirm the operation script of the robot arm through APP

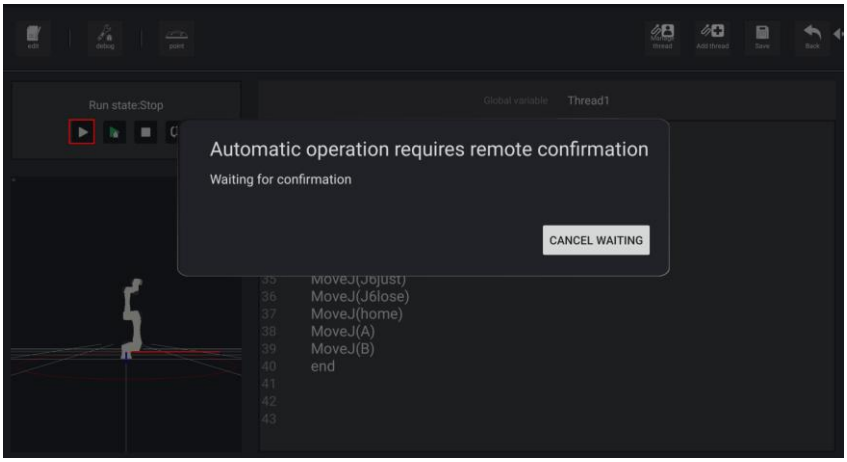


Figure 3.2 Remote confirmation tips

3.3 End I/O Interface Description

The cable used for the end interface is our designated cable，the model is **Lumberg RKMV 8-354**.

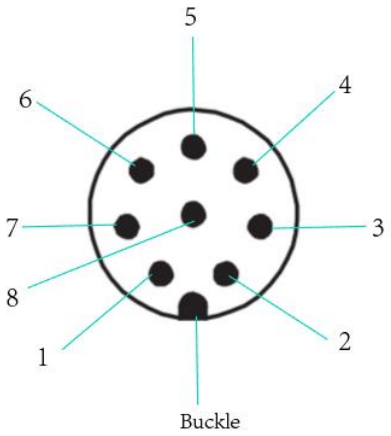


Figure 3.3 End I/O interface

Table 3.5 End I/O description

Pin	name	define
1	AI_1/485A	Analog input 1 / 485A
2	AI_2/485B	Analog input 2 / 485B
3	DI_2	Digital input 2
4	DI_1	Digital input 1
5	24V	24V（Out）
6	DO_2	Digital output 2

7	DO_1	Digital output 1
8	GND	GND

3.4 Interface Board

3.4.1 External Interface Board Description

Figure 3.4 shows the interface board of the CC series controller. Table 3.6 lists the description.

Figure 3.4 Interface board of the controller

Table 3.6 Interface description

No.	Description
1	LAN interface For connecting to external network equipment (Used for debugging). The network segment of the external device should be set to 192.168.5.X, and the default network segment of the machine is network segment 5.
2	Emergency stop interface For connecting to emergency stop switch
3	USB interface For connecting to WiFi module
4	I/O interface. For details, please see *3.1 Controller I/O* Interface Description

No.	Description
5	Power switch of controller For controlling the controller power on and off
6	Power interface For accessing single-phase 1100/220V power supply
7	Heavy-duty connector interface For connecting to robot
8	Power switch of robot For controlling the robot power on and off

3.4.2 Internal Module Description

A controller contains all necessary functions for controlling the robot. Figure 3.5 shows the internal modules.

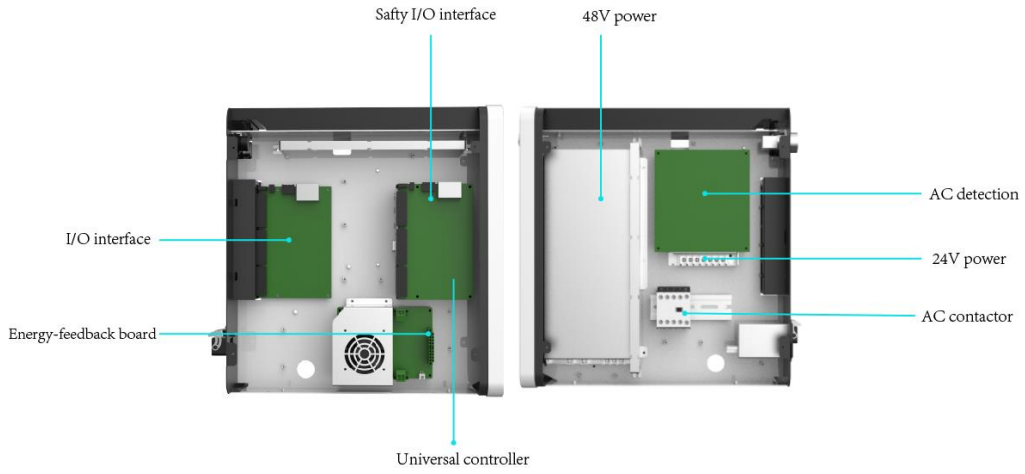


Figure 3.5 Internal module

Table 3.7 lists the internal module description

Table 3.7 Module description

Module	Function
AC contactor	Control the power input of the robot
I/O interface	Connect and communicate with external/internal devices, and support safety logic processing
Energy-feedback Board	Consume the feedback energy generated by the motor when the servo drive is powered off or slows down, to

Module	Function
	protect the servo drive
24V power	Turn single-phase 110V/230V AC into 24V DC for powering controller module, I/O module, and other logic modules
AC detection	Check the AC input and internal DC status of the controller. Under abnormal AC and DC conditions, the AC detection board can provide sufficient DC power to ensure the equipment's emergency stop operation.
48V power	Turn single-phase 110V/220V AC into 48V DC for powering servo and motor
Controller module	Controller module is used to control the logic unit of the robot system, responsible for human-interface interaction, trajectory planning, motion control, status monitoring, and so on
Safety I/O interface	Safety I/O interface is used to control the safety circuit of system, such as emergency stop

3.4.3 Digital Input

Figure 3.6 shows the simple digital input circuit and Table 3.8 lists the technical specifications.

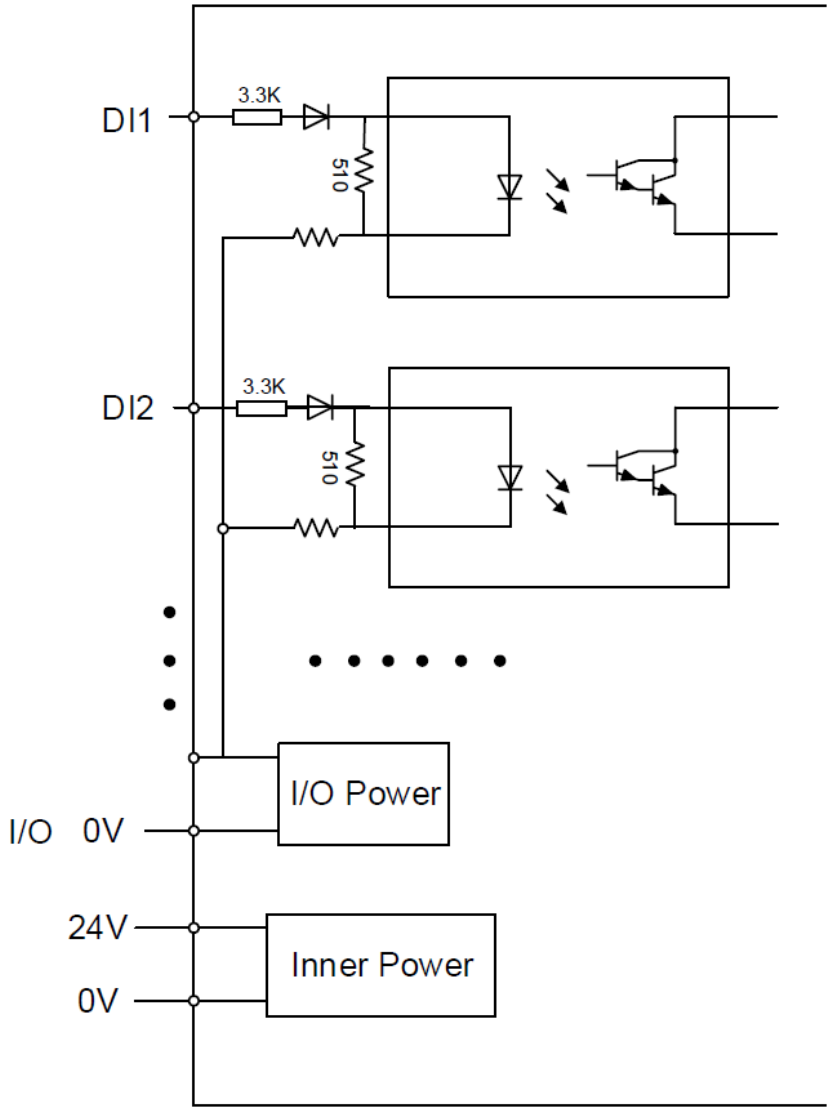


Figure 3.6 Simple digital input circuit

Table 3.8 Technical specifications

Item	Specification
Input channel	16 channels
Connection method	Crimping terminal
Input type	Optical coupling type
Input voltage (DC)	0V or 24V±10%
Isolation method	Optical coupling isolation

3.4.4 Multiplex Digital Input/ Digital Output

3.4.4.1 Introduction

The multiplex digital input/ digital output interface can be powered by the internal or external power supply. Figure 3.7 shows the simple digital output circuit and Table 3.9 lists the technical specifications.

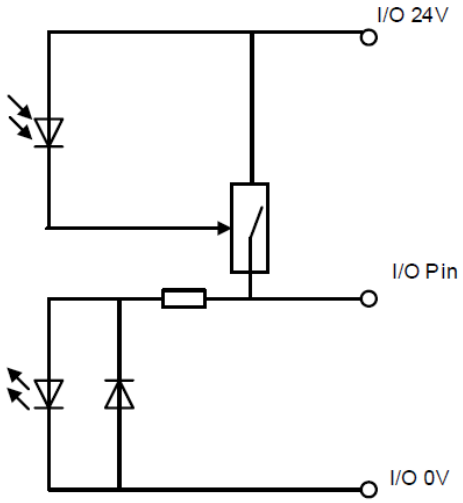


Figure 3.7 Simple digital output circuit

Table 3.9 Technical specifications

Item	Specification
Output channel	16 channels
Connection method	Crimping terminal
Output type	High-side switch
Power supply (DC)	24V±10%
Load current of single channel	500mA
Output current	2A
Isolation method	Digital isolation

3.4.5 Analog Input

3.4.5.1 Introduction

An analog input can be set to a current input or voltage input by the DIP switch for measuring current or voltage. A controller uses 2-channel DIP switches to control 2 channel analog inputs. Each channel can be controlled separately. Figure 3.8 shows the simple analog input circuit. **V** indicates the voltage input and **I** indicates the current input. The default is voltage input. If you need to change, please contact our technical support.

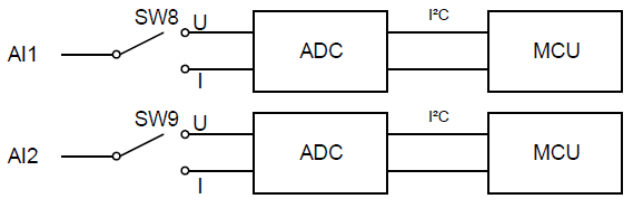


Figure 3.8 Simple analog input circuit

Table 3.10 lists the relation between the DIP switch and analog input.

Table 3.10 The relation between the DIP switch and analog input

Analog input type	Range
Current input	Current : 4mA - 20 mA
Voltage input	Voltage : ±10V

3.4.6 Analog Output

3.4.6.1 Introduction

Much like the analog input, an analog output can be set to a current output or voltage output by the DIP switch. A controller uses 2-channel DIP switches to control 2 channel analog outputs. Each channel can be controlled separately. Figure 3.9 shows the simple analog output circuit. **V** indicates the voltage output and **I** indicates the current output. The default is voltage output. If you need to change, please contact our technical support.

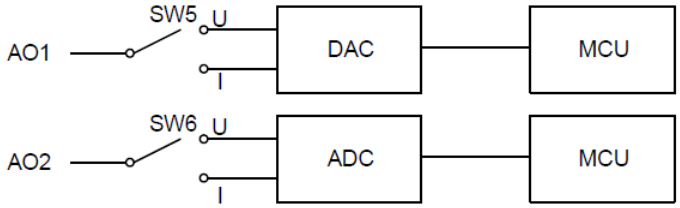


Figure 3.9 Simple analog output circuit

Table 3.11 lists the relation between the DIP switch and analog output.

Table 3.11 The relation between the DIP switch status and analog output

Analog output type	Range
Current output	Current : 4mA - 20mA
Voltage output	Voltage: 0V - 10V

3.4.7 Incremental Encoder ABZ Input

An encoder is a device that converts angular or linear displacement into electrical signals. Namely, it converts displacement into periodic electrical signals and then converts electrical signals into count pluses. So, the displacements can be measured by the number of pluses.

This topic takes OMRON E6B2-CWZ1X as an example to describe how to use.

As the different color cables of the encoder, connect the 5V power line to Pin49, the 0V power line to Pin41, and then connect each coded wiring in turn.

The cable color description is shown as follow.

Table 3.12 Cable color description

Color	Description
Brown	I/O 5V
Blue	I/O 0V
Black	A+
White	B+
Orange	Z+
Black and red	A-
White and red	B-
Orange and red	Z-
Shield	Ground

NOTE

If a ground wire is required, it can be fixed to the control cabinet with screws. Non-special circumstances (strong magnetic interference, etc.)

4. Installation and Commissioning

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.
- Do not install in a closed environment.
- 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. The base of the robot must use four hexagon socket bolts M8 (GB / T 3098.1-82 strength level 12.9) and tighten with 20 N • m torque.
- When the robot is mounted on the wall or upside down, just in case, the anti-falling measures of the robot base must be done.
- 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

4.2.1 Controller Installation Location

Please place the controller on the horizontal surface outside robot's workspace and reserve enough space for connecting cables and operating controller. Figure 4.1 shows the installation space requirement. There is no blockage near the air outlet for sufficient heat dissipation.

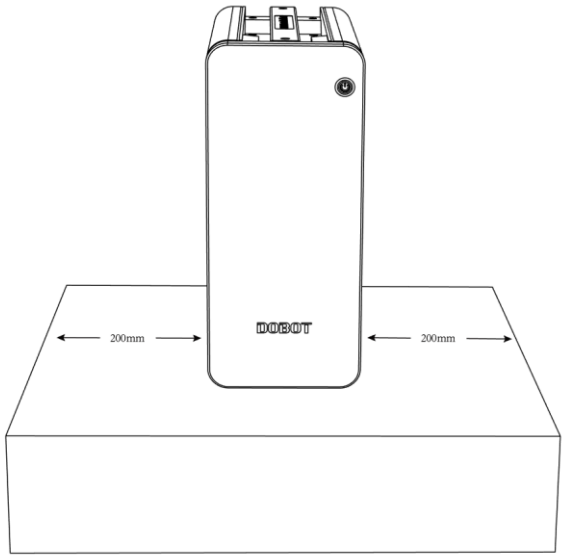


Figure 4.1 Installation space requirement

4.2.2 Robot 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. 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.

4.3 Connecting cables

4.3.1 Precautions

- The specifications and installation of the external wiring should comply with local laws and regulations.
- Do not disassemble the controller by yourself. Otherwise, it may result in electric leakage.
- The equipment must be grounded properly at all times to avoid the risk of electric shock.
- Do not allow unnecessary strain on the cables. Otherwise, damaged cables, disconnection, and contact failure are extremely hazardous and may result in electric shock.
- Before connecting to external equipment, please turn off the controller and related equipment, and then unplug the power. If not, it may result in electric shock or malfunction of the robot system.
- Please make sure that the cables are connected correctly. Otherwise, it may result in a malfunction of internal modules or external devices.
- Please use the matched cables for personal security and equipment protection.
- After the cable connections are complete, please make sure that there are no redundant

screws or exposed cables inside the equipment.

- When the equipment is running, please do not plug or unplug the power and communication cables.
- Please confirm that the device cable is connected correctly, otherwise it may cause the internal module or external device to malfunction.
- Before connecting, check whether the insulation and shield of the external cable are damaged.

4.3.2 Connecting Controller and Robot by Heavy Duty Cables

Connect controller and robot by heavy-duty cables. Figure 4.2 shows the connection of the controller with heavy-duty cables. After plugging the heavy cables into the heavy-duty connector interface of the controller, please fasten the heavy-duty connector.

Figure 4.2 Connect controller and robot by heavy-duty cables

4.3.3 Connecting Emergency Stop Switch

Plug emergency stop switch cable into controller, when connecting them, you need to align the red dot on the connector with the red dot on the interface. As shown in Figure 4.3.

Figure 4.3 Connecting emergency stop switch

⚠NOTICE

- Before running a robot, please make sure that the emergency stop switch has been turned on (the red button has been released). Otherwise, the robot cannot work normally.
- In the emergency situation, press the emergency stop switch to make the robot stop running immediately.
- Rotate the emergency stop switch (Red button) clockwise, If the red button is released, the emergency stop switch is turned on successfully.

4.3.4 Connecting WiFi Module

Plug WiFi module into USB interface.as shown in Figure 4.4. The Overall connection diagram is shown in Figure 4.4, the default IP is 192.168.1.6.

Figure 4.4 Connecting WiFi module

4.3.5 Connecting to Power Supply

Plug power cable into controller. As shown in Figure 4.5.

Figure 4.5 Connecting to Power Supply

4.4 Debugging Power On and Off

Preparations

- Controller have been connected to robot
- Emergency stop switch has been turned on.

Procedure

Step 1 Press the power switch to power on the controller, then press the controller button to

start robot. As shown in Figure 4.6.

Figure 4.6 Start robot

Step 2 Open tablet to search and connect local area network, the prefixes of network is **Dobot_WIFI_XXX**, and XXX is the number of robot which locates in the base of robot. The password is 1234567890. You can also modify it. As shown in Figure 4.7.

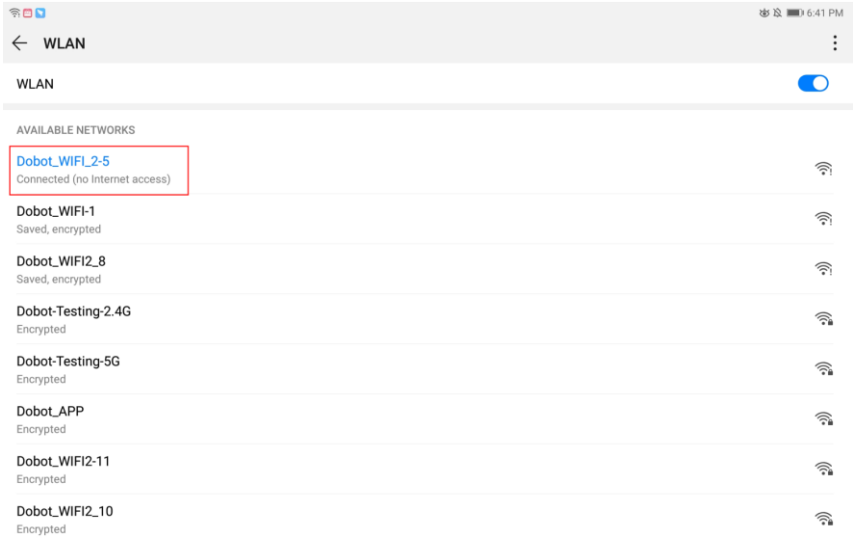


Figure 4.7 connecting local area network

Step 3 Open APP and click **Connect** to connect controller and APP. As shown in Figure 4.8.

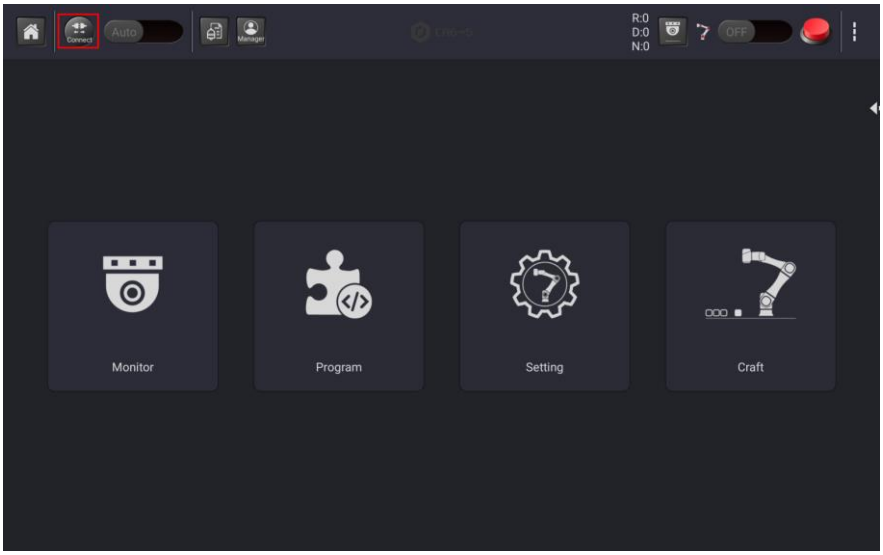


Figure 4.8 Connecting controller and APP

Step 4 Enable robot. Set the button **ON/OFF** to **ON**.

The robot icon  turns green, indicating that robot enables successfully. Now you can jog robot.

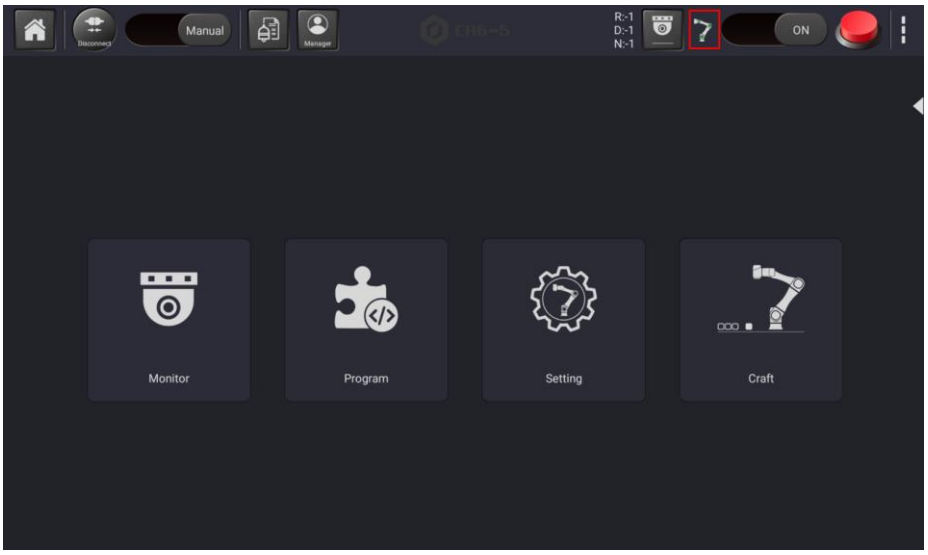


Figure 4.9 Enable robot

⚠NOTICE

The robot arm will move a certain distance during the enabling process. Please ensure that there are no obstacles around the robot.

Step 5 Disable robot. Set the button **ON/OFF** to **OFF**, and the robot icon  turns red, indicating that robot disables successfully.

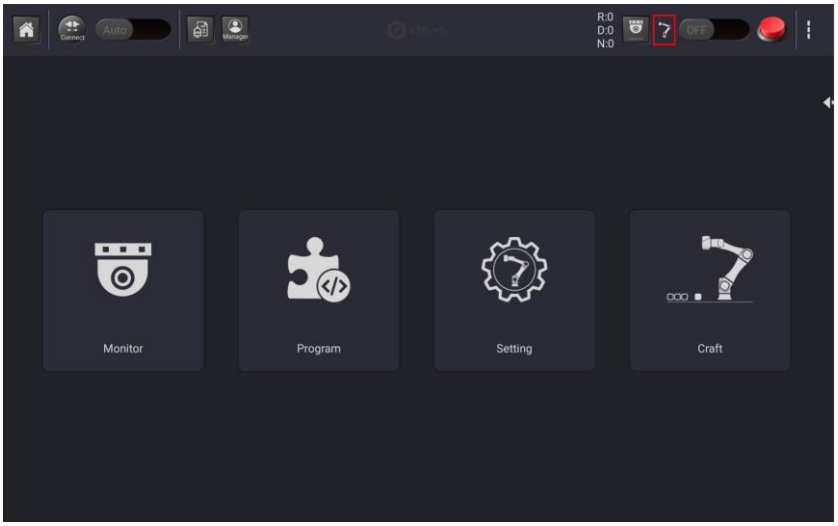


Figure 4.10 Disable robot

Step 6 Click **Disconnect** to disconnect controller and APP.

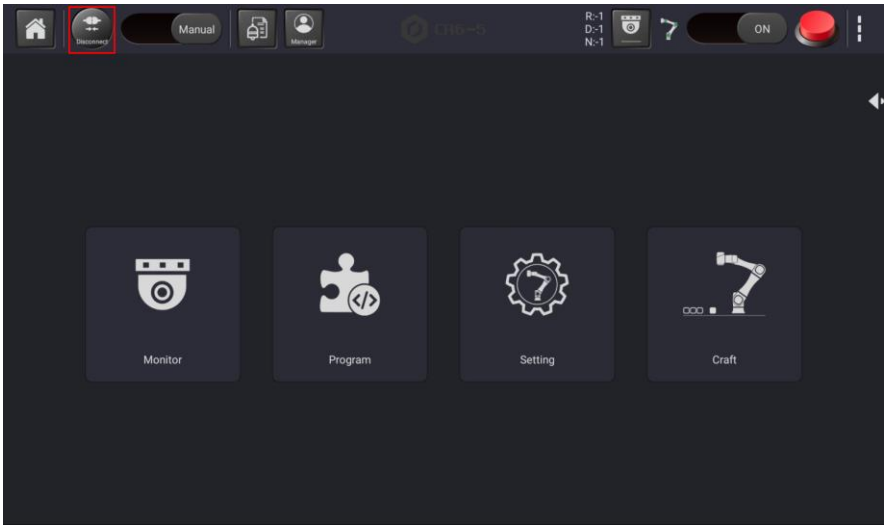


Figure 4.11 Disconnection

5. Function Description of Software

The robot supports Android/iOS tablet operation. Other types of devices are not currently supported. This manual uses Android tablet as an example.

The interface is shown in Figure 5.1, and its detailed description is shown in Table 5.2.

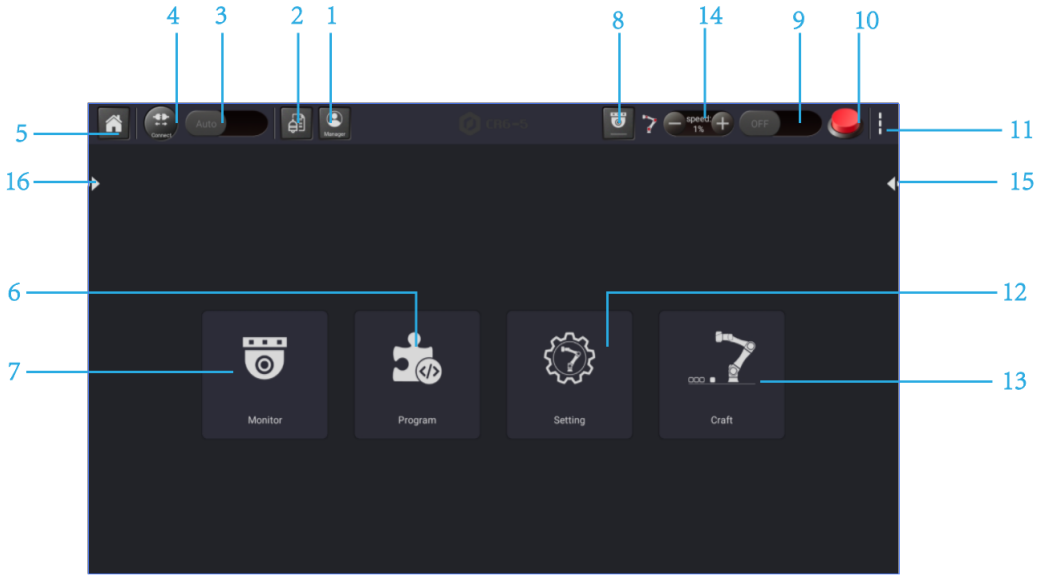


Figure 5.1 Homepage

Figure 5.2 Jogging interface

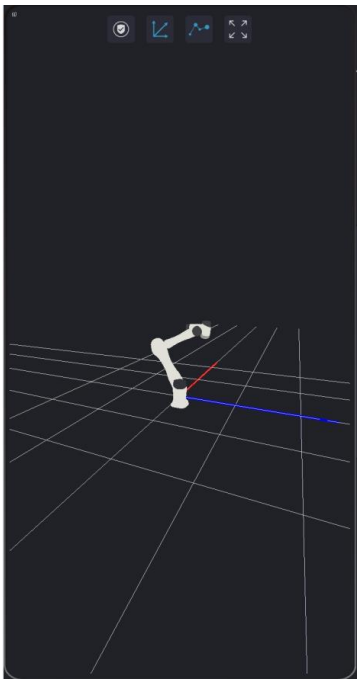


Figure 5.3 3D illustration of a robotic arm

Table 5.1 Cable color description

NO.	Description
1	Manager APP login personnel are divided into observer, operator, programer and manager, different personnel can operate different functions, including the most authority of the management, can operate all functions
2	alarm log You can click it to check alarm log
3	• Manual mode: You can jog robot arm, program or set parameters, etc • Auto mode: You can monitor motion tracks, I/O ports, debug program, etc
4	Connection button When device and robot arm are connected to network, click the button to connect the device to robot arm
5	Click this button to go back to the previous page
6	Programming module Programming modules are mainly used for editing and running

NO.	Description
	programs
7	Monitoring module You can view the status of robot arm, set the output of I/O , set the end parameters of robot arm and other functions
8	Monitoring module shortcuts
9	Robotic arm enable button ON： the icon is green when robot motor is in the enabled status OFF： the icon is red when robot motor is in the disabled status
10	Emergency stop switch Emergency stop switch can be pressed when robot arm is in short of time during operation, to control servo drive power off and robot arm stop urgently but constant power
11	System settings Click this button to expand the page to view help documentation, lock screen, switch skins, etc.
12	Setting Set the related parameters of robot arm, including motion parameters, coordinate system settings, calibration, etc
13	Craft Support Drag teach, Conveyor belt, Palletizing, and Vision, etc
14	Set speed ratio
15	Jogging robotic arm Click the button to expand the jogging page, as shown in Figure 5.2, you can jogging the robot arm. Click the button  to switch the joint coordinate system or the Descartes coordinate system
16	Click the button to expand the 3D illustration of a robotic arm, as shown in Figure 5.3

⚠NOTICE

The software supports the screen lock function. If the software has not been operated for a long time, the screen is automatically locked. The unlock password is 000000 by default.

5.1 Setting

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

5.1.1 Setting Motion Parameter

5.1.1.1 Setting Jog

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**

Set the maximum velocity and acceleration in Joint and Cartesian coordinate system when jogging a robot. As shown in Figure 5.4.

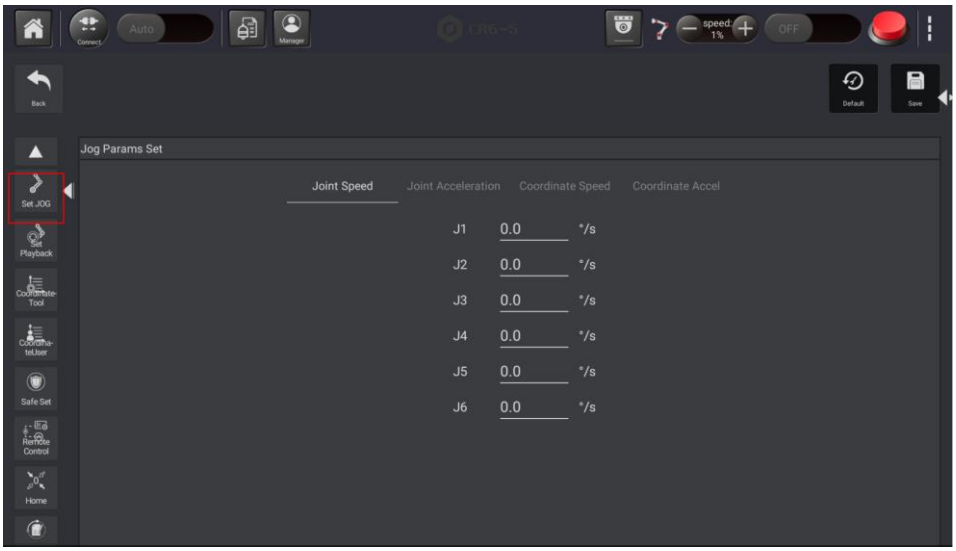


Figure 5.4 Jogging parameters in the Joint coordinate system

5.1.1.2 Setting Playback

Set the maximum velocity, acceleration, and jerk in the Joint and Cartesian coordinate system when running robot programs, as shown in Figure 5.5.

Figure 5.5 Playback parameters

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

NOTE

The rates (velocity rate, acceleration rate, or jerk rate) can be set in the related speed functions. For details, please see *6.9 Motion Parameter Commands.*

If the motion mode is **Jump** when running robot programs, you need to set **StartHeight**, **EndHeight**, and **zLimit**. For details about **Jump**, please see2.5.1.2Linearly Interpolated Motion.

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 5.6. Please refer to Table 6.24 for use

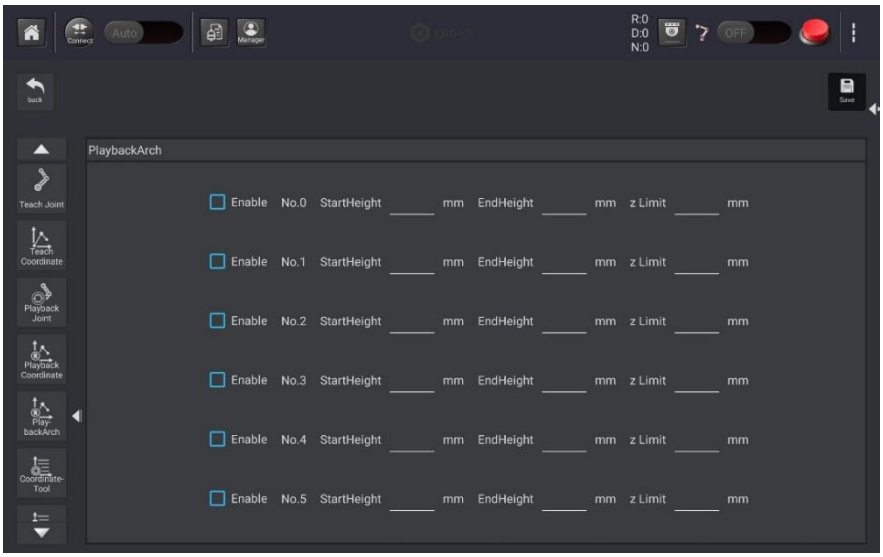


Figure 5.6 Jump parameters

5.1.1.3 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.

- Point: move TCP to any point **A** to create origin, and create user coordinate system according to the default tool coordinate system As shown in Figure 5.7.

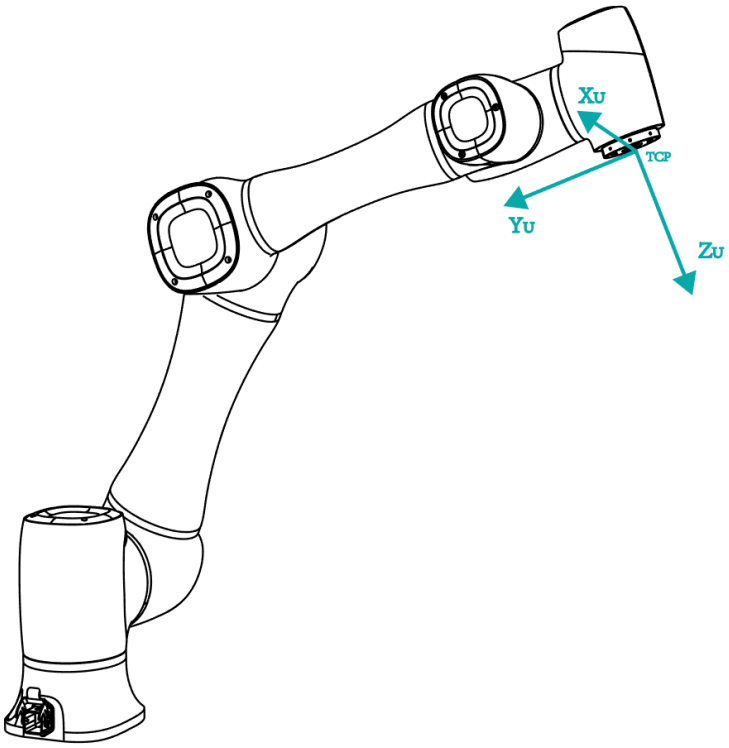


Figure 5.7 Point

- Line: Confirm a straight line by any two points **A** and **B**. The direction from A to B is defined as the positive direction of Y-axis, The Z-axis of Tool coordinate system of which point A is the origin is projected into the vertical plane that confirmed by points A and point B, we can define it as the positive direction of Z-axis. and then the positive direction X axis can be defined based on the right-hand rule. As shown Figure 5.8.

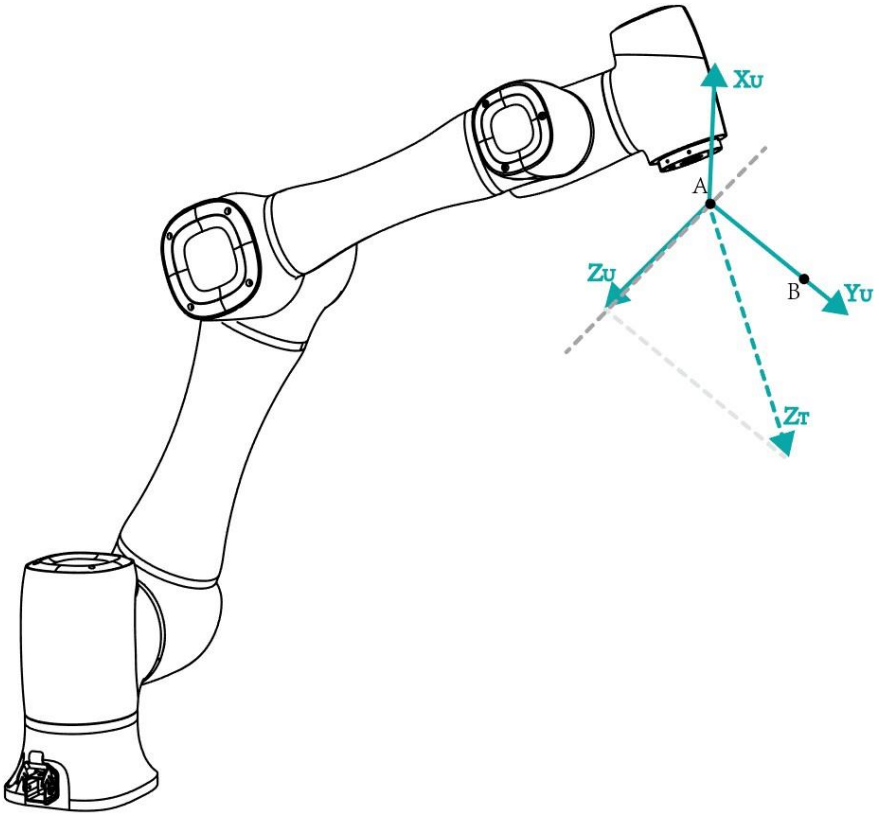


Figure 5.8 Line

- Area: User coordinate system is created by three-point calibration method. Move the robot to three points **A(x1, y1, z1)**, **B(x2, y2, z2)**, and C(**x3**, **y3**, **z3**). Point A is defined as the origin and the line from point A to Point B is defined as the positive direction of X-axis. The line that point C is perpendicular to X-axis is defined as the position direction of Y-axis. And then the Z-axis can be defined based on the right-handed rule, as shown in Figure 5.9.

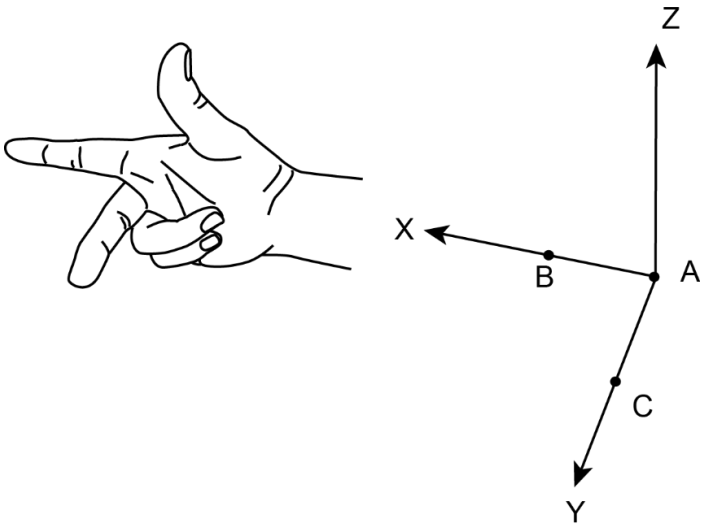


Figure 5.9 Area

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

Prerequisites

- The robot has been powered on.
- The APP has been in the manual mode.
- The robot is enabled.
- The robot is in the Cartesian coordinate system.

Procedures

Step 1 Click **Coordinate User** on the **Setting** page. As shown Figure 5.10.

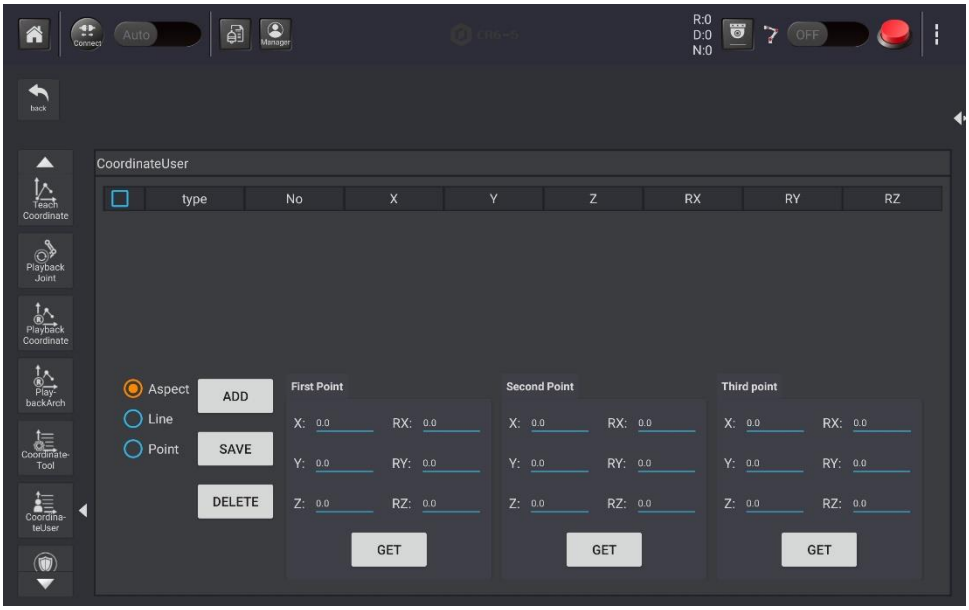


Figure 5.10 User coordinate system

Step 2 Click **Aspect** and jog robot to the first point and click **GET** to get the first point.

Step 3 Jog robot to the second point and click **GET** to get the second point.

Step 4 Jog robot to the third point and click **GET** to get the third point.

Step 5 Click **ADD** and **SAVE** to create User 1coordinate system.

Step 6 Click **Change coordinate** on the APP and select **NO 1**. If the icon **NO.0** turns into **NO.1**, you can use the User 1 coordinate system for teaching and programming.

5.1.1.4 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.

Tool coordinate system is created by three-point calibration method (TCP +ZX): After the end effector is mounted, please adjust the direction of the end effector, to make TCP (Tool Center Point) align with the same point (reference point) in three different directions, for obtaining the position offset of the end effector, and then jog the robot to the other three points (**A**, **B**, **C**) for obtaining the angle offset, as shown in Figure 5.11.

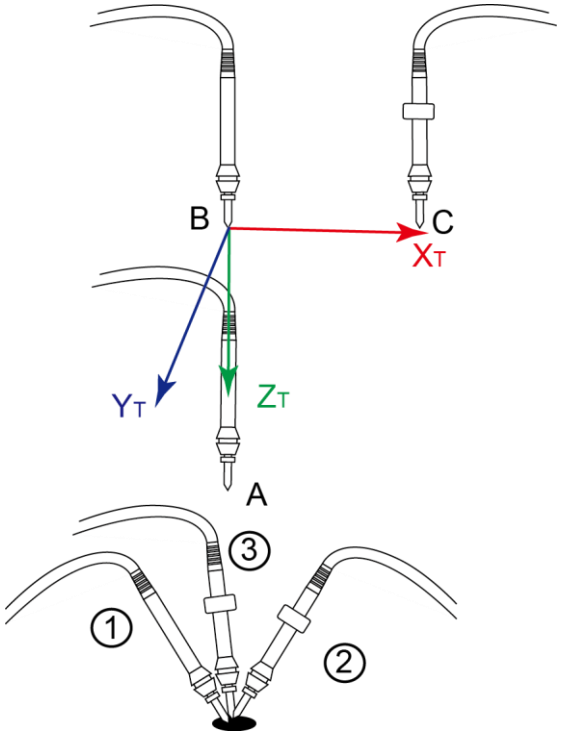


Figure 5.11 Three points calibration method (TCP+ZX)

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

Prerequisites

- The robot has been powered on.
- The APP has been in the manual mode.
- The robot is in the Cartesian coordinate system.

Procedure

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

Step 2 Click **Tool coordinate** on the **Setting** page. As shown Figure 5.12.

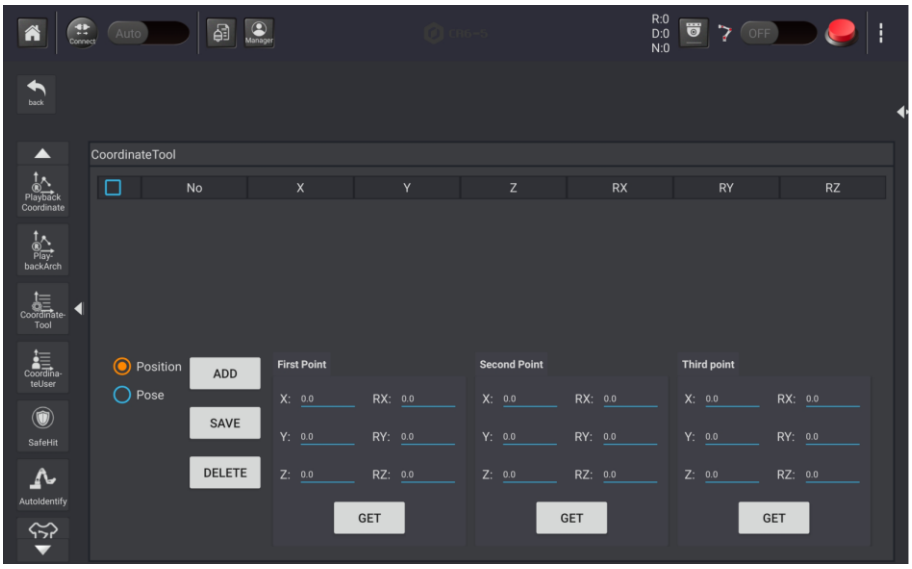


Figure 5.12 Tool Coordinate page

NOTE

Rx, **Ry**, **Rz** are the orientation data, which are designated by rotating the tool center point (TCP) around the X, Y, Z axes under the selected Tool coordinate system.

Step 3 Click **Position**, Jog the robot to the reference point in the first direction, then click **GET** to get the coordinates of the first point.

Step 4 Jog the robot to the reference point in the second direction, then click **GET** to get the coordinates of the second point.

Step 5 Jog the robot to the reference point in the third direction, then click **GET** to get the coordinates of the third point.

Step 6 Jog the robot to the reference point (point **A)** in the vertical direction, then click **GET** to get the fourth point.

Step 7 Jog the Z-axis to a point (point **B**) along the positive direction, then click **GET** to get the fifth point

This step defines the Z-axis.

Step 8 Jog the X-axis to another point (point **C**), then click **GET** to get the sixth point

The three points (A, B, C) cannot lie in the same line.

This step defines the X-axis, and the Y-axis can be defined based on the right-handed rule.

Step 9 Click **ADD** and **SAVE** to create Tool 1coordinate system.

Step 10 Click **Change coordinate** on the APP and select **NO.1**. If the If the icon **NO.0** turns into **NO.1**, you can use the User 1 coordinate system for teaching and programming.

5.1.2 Setting Safety

In the safety setting module, you can set safety parameters such as safety hit, power control, joint brake, etc.

5.1.2.1 Safety Hit

If the collision detection is activated, the robot arm will stop running automatically when the robot arm hits an obstacle.

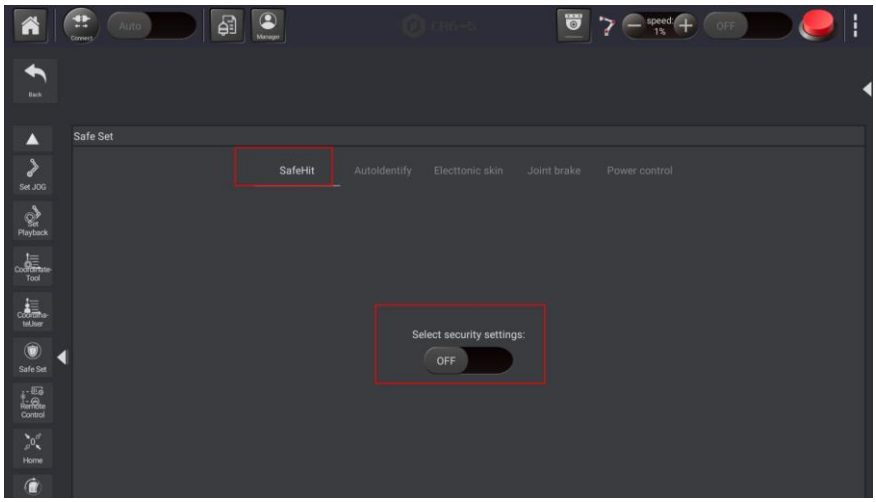


Figure 5.13 Safety Hit

5.1.2.2 Electronic Skin

If the electronic skin function is activated, the robot will evade it when the robot hits an obstacle.

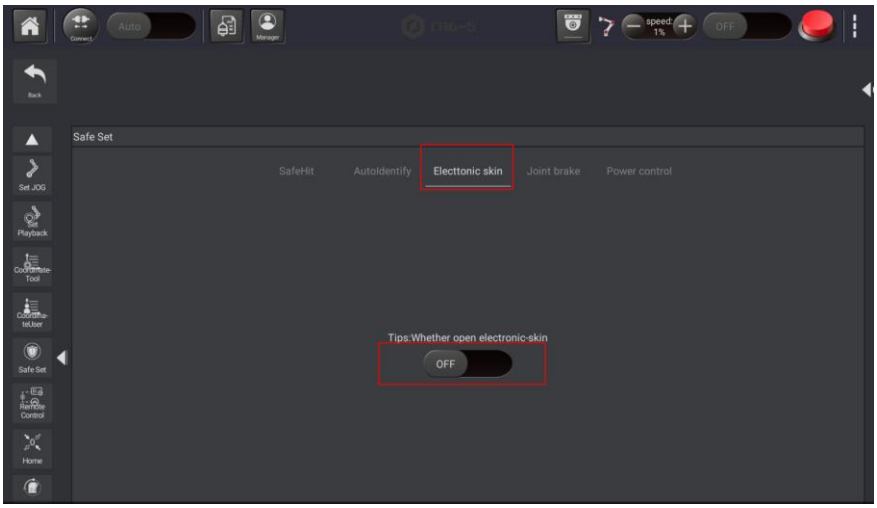


Figure 5.14 Safety Hit

5.1.2.3 Joint Brake

After the robot is disabled, if you want to drag joints, you need to enable **Joint brake** function to drag them.

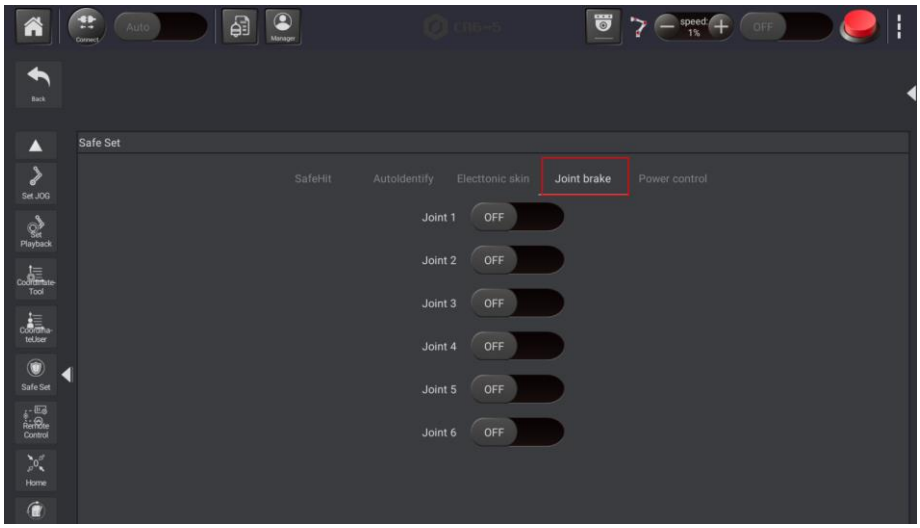


Figure 5.15 Joint brake

5.1.2.4 Power Control

This function is used to control robot power on or power off.

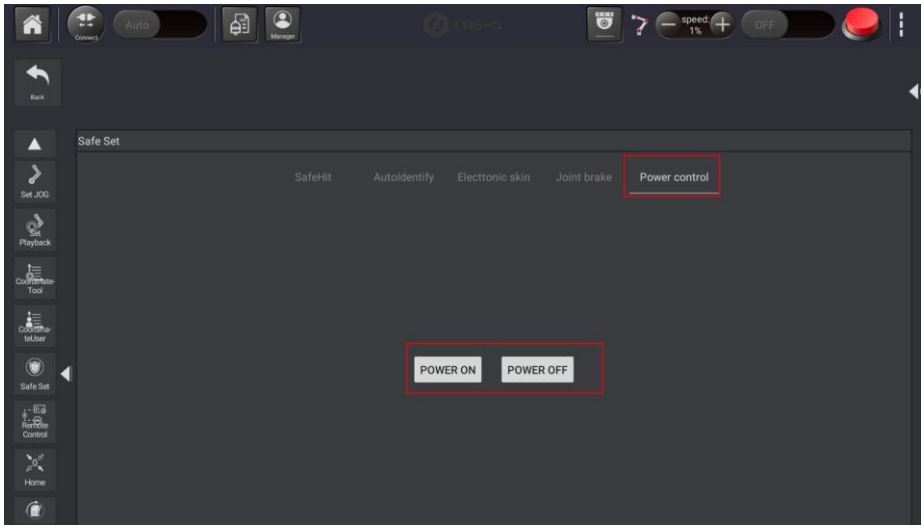


Figure 5.16 Power control

5.1.3 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 APP with the robot motor 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 robot system 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 robot motor 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.

5.1.3.1 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 5.2.

Table 5.2 Specific I/O interface description

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

⚠NOTICE

All input signals are low to high.

Prerequisites

- The project to be running in the remote mode has been prepared.
- The external equipment has been connected to the robot system by the I/O interface. The specific I/O interface description is shown in Table 5.2. The IP address of the Dobot robot control system and the external device must be on the same network segment. The default IP address of the Dobot robot control system is 192.168.5.1 and the port is 8080.

- The robot has been powered on.

  NOTE

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

Procedure

Step 1 Click **RemoteControl** on the **Settings** page.

The remote control page is displayed, as shown in Figure 5.17.

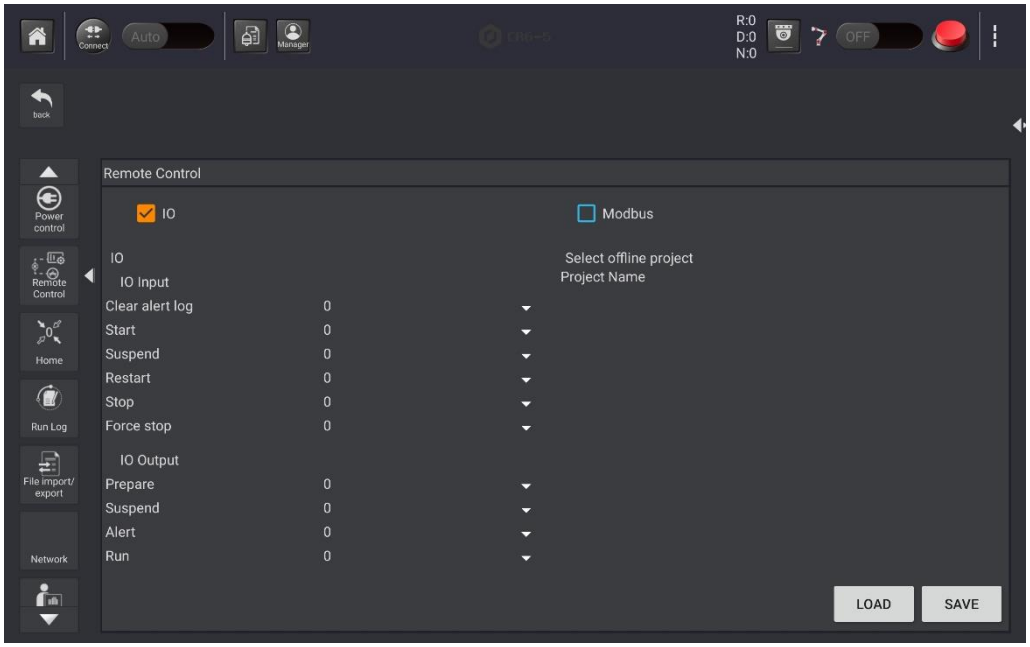


Figure 5.17 Remote control page

Step 2 Select **IO** on the **Control Mode** section and select the offline project on the **Select Offline Project** section. Click **SAVE** and **LOAD.**

Right now, only the emergency stop button and the real-time coordinates displaying section are 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.

5.1.3.2 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 *6.16.1Modbus Register Description.*

Table 5.3 Specific Modbus register description

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

Prerequisites

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

  The IP address of the robot system must be in the same network segment of the external equipment without conflict. The default IP address is 192.168.5.1.
- The robot has been powered on.

  NOTE

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

Procedure

Step 1 Click **RemoteControl** on the **Settings** page.

The remote control page is displayed, as shown in Figure 5.18.

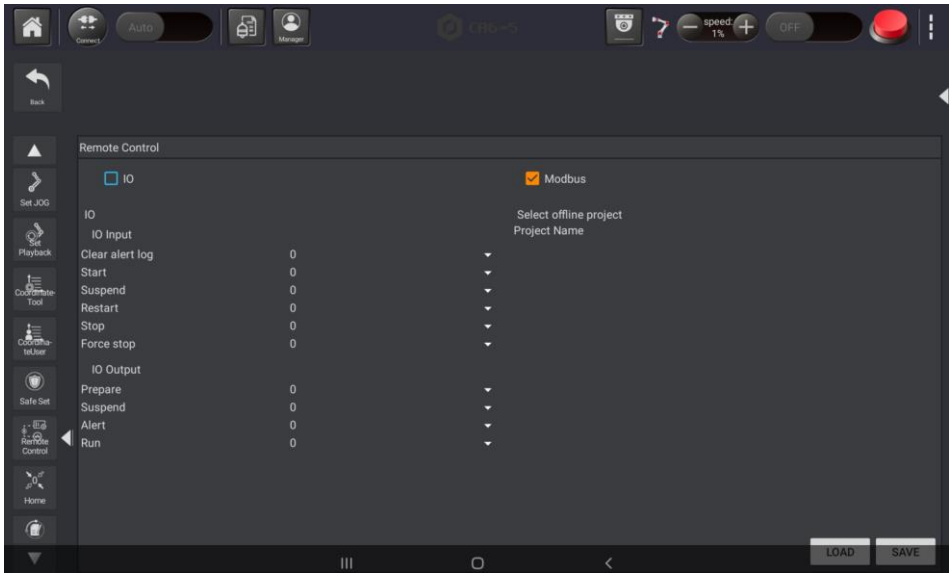


Figure 5.18 Remote control page

Step 2 Select **Modbus** on the **Control Mode** section and select the offline project on the **Select Offline Project** section. Click **SAVE** and **LOAD**.

Right now, only the emergency stop button and the real-time coordinates displaying section are 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 Modbus mode will be invalid.

5.1.4 Homing

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.

Click **Home** to make robot move to the origin position after switching robot to manual mode and enabling robot.

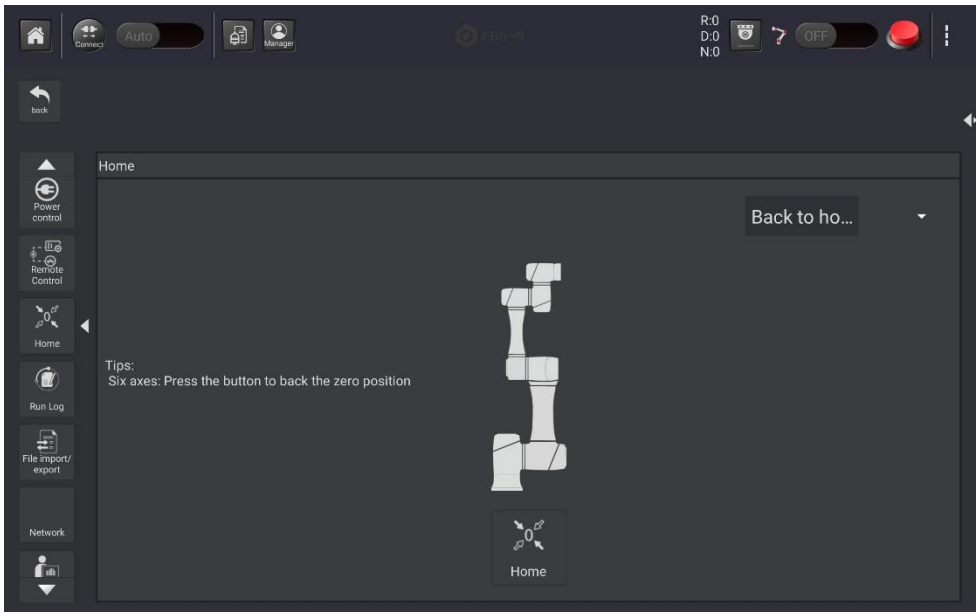


Figure 5.19 Homing page

If the origin of robot has been changed, you need to reset it. **Switch Back to home** to **Set home position**, and put the robot in the original position and click **Home** to set origin of the robot. This feature only supports administrator settings.

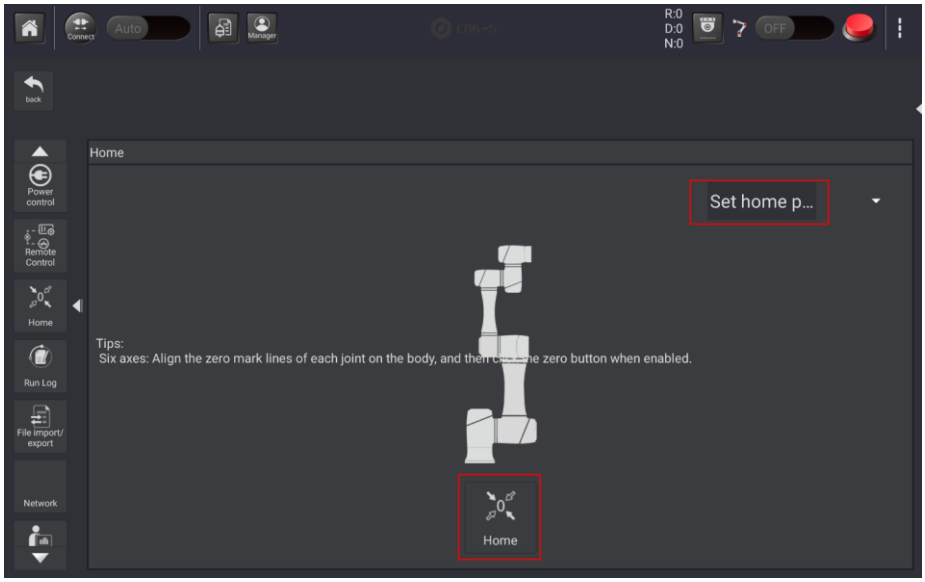


Figure 5.20 Set home position

5.1.5 Running Log

You can check, export or clear running logs on this page.

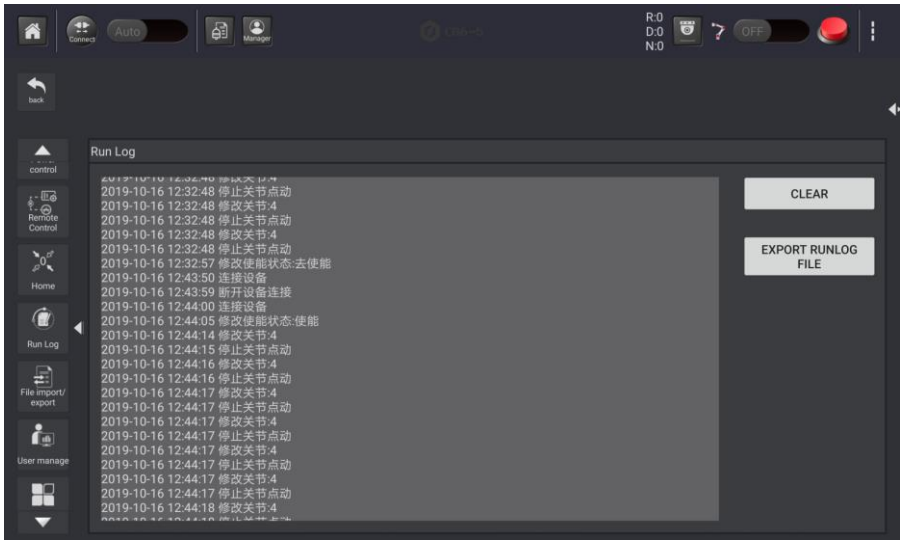


Figure 5.21 Running log

5.1.6 Setting Software

5.1.6.1 Setting APP lock

You can set APP lock screen time as needed. If the software is not operated within this time period, the system will automatically lock the screen. The unlock password defaults to: 000000.

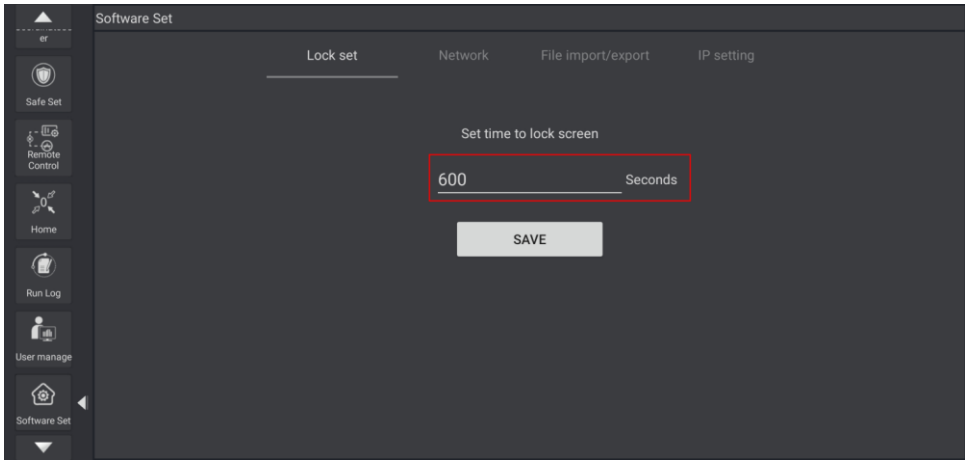


Figure 5.22 Setting lock time

5.1.6.2 Importing/exporting Files

This function is used to import or export file from controller which can improve reusability.

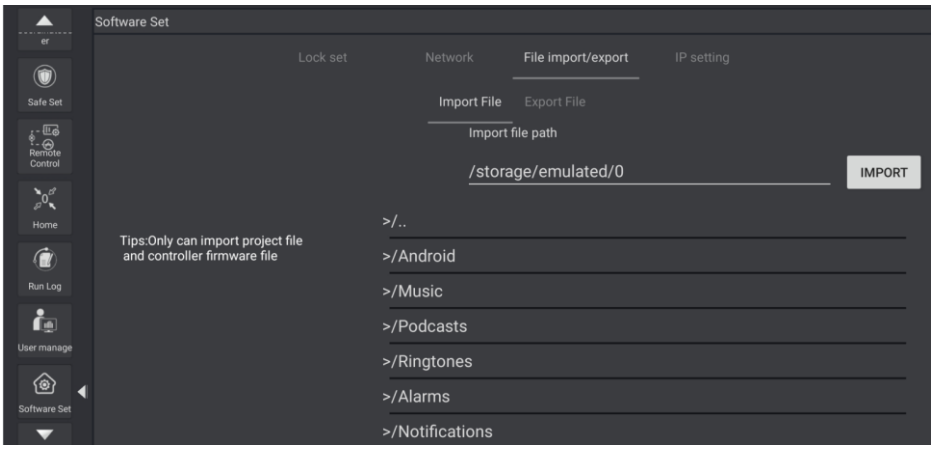


Figure 5.23 Import/export file

5.1.6.3 Setting Network

You can set the IP, port and password of the server (WiFi) according to actual needs. The default IP is 192.168.1.6 and the port is 22000.

Figure 5.24 Set network

5.1.6.4 Setting Controller IP

You can set the controller IP as required. The default controller IP is 192.168.5.1.

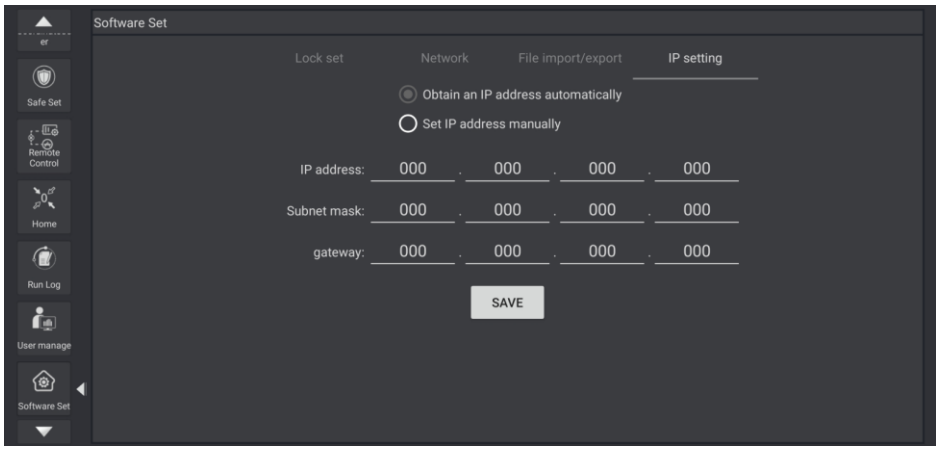


Figure 5.25 Setting controller IP

5.2 Hand-Hold Teach

You can drag robot to teach by APP or function keys on the end of robot. For function key details, please refer to *2.6 Key Description on Robot*.

Click **Craft>Drag teach**, the dragging page is displayed, as shown in Figure 5.26.

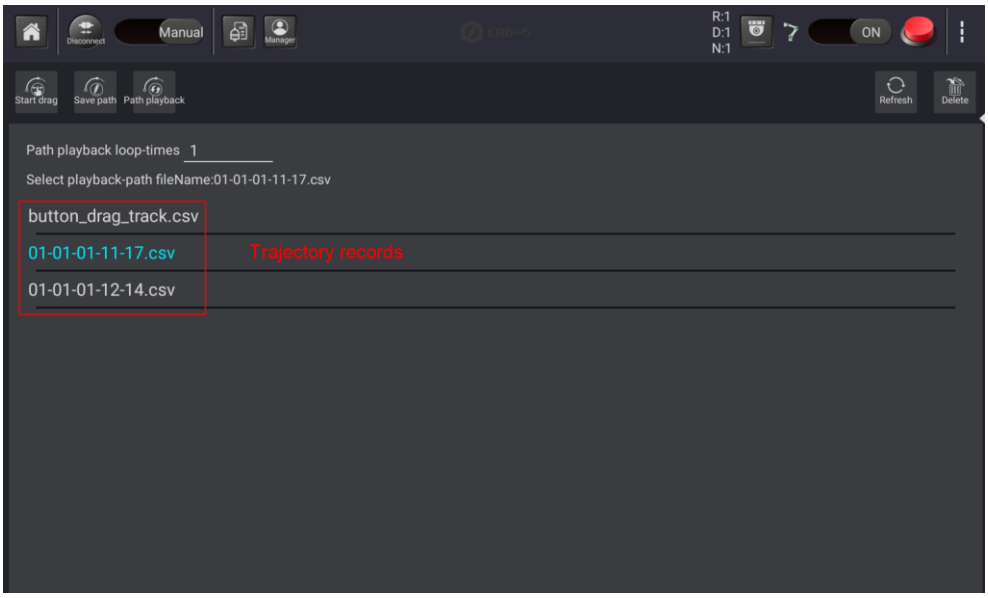
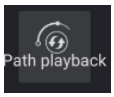


Figure 5.26 Hand-hold teach page

Table 5.4 Button description

Button	Description
	Click this button to open or close drag mode

Button	Description
	Click this button to save trajectory
	Click this button to start or stop playback. Select the recorded trajectory, and click Path playback to start trajectory playback You can set **Path playback loop-time** on the hand-hold teach page, the default value is **1**
	Click this button to refresh trajectory records
	Select a trajectory and click this button to delete it

NOTE

There are two kinds of trajectory names. For example, **01-01-00-20-07.csv** is a trajectory which records by APP. You can record multiple trajectories in this type. **Button_drag_track.csv** is a trajectory which records by clicking function keys on the end of robot. In this type, you can only record one trajectory.

5.3 Monitor

5.3.1 I/O Monitor

You can monitor the status of I/O, as shown in Figure 5.27, you can select checkbox to change the **X:0** from output into input. Click **X:0** to set High(1) or Low(0) level (X indicates the number of I/O).

Figure 5.27 I/O Monitor

5.3.2 Robot Status

You can check temperature, voltage, current of robot and so on.

Figure 5.28 Robot status

5.3.3 Controlling End-effectors

5.3.3.1 Setting Gripper

You can enable and control gripper in this page. Select **Enable** to enable gripper, select **Open** or **Close** to open or close gripper.

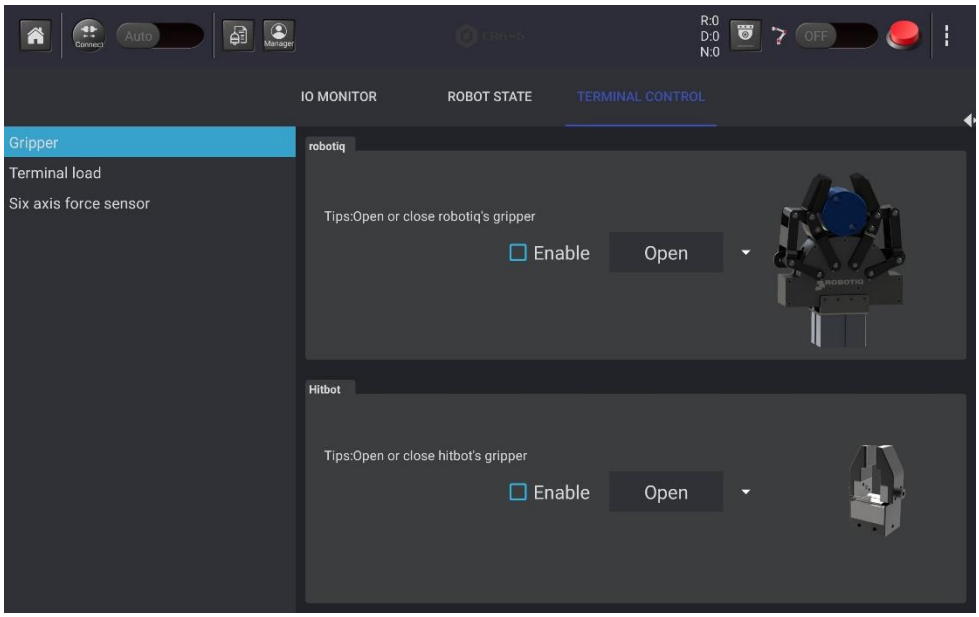


Figure 5.29 Set gripper

5.3.3.2 Setting End-loading of Robot

Set **LoadValue** in this page, the value range is 0 kg to 5 kg, Other parameters do not need to set.

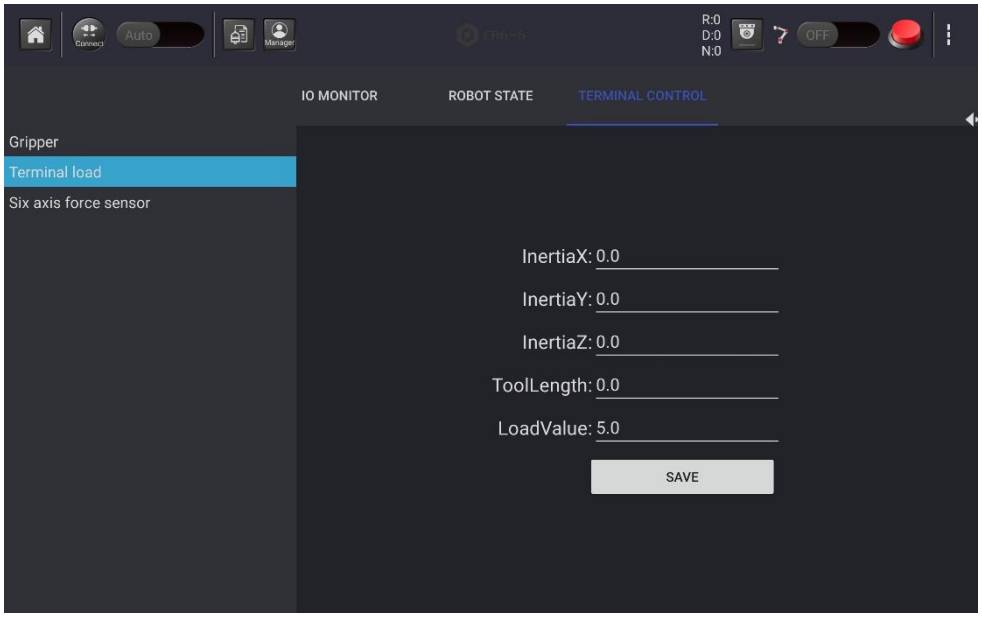


Figure 5.30 set end-loading of robot

5.3.3.3 Setting Six-axis Force Sensor

Set six-axis force sensor to open or close. When opening six-axis force sensor, you can calibrate it or operate homing procedure. The six-axis force sensor supported by the current machine are Robotip brand FT 300 models

- Home

Before using six-axis force sensor, you need to Select **open** and click **ZERO SENSOR** to operate six-axis force sensor homing procedure.

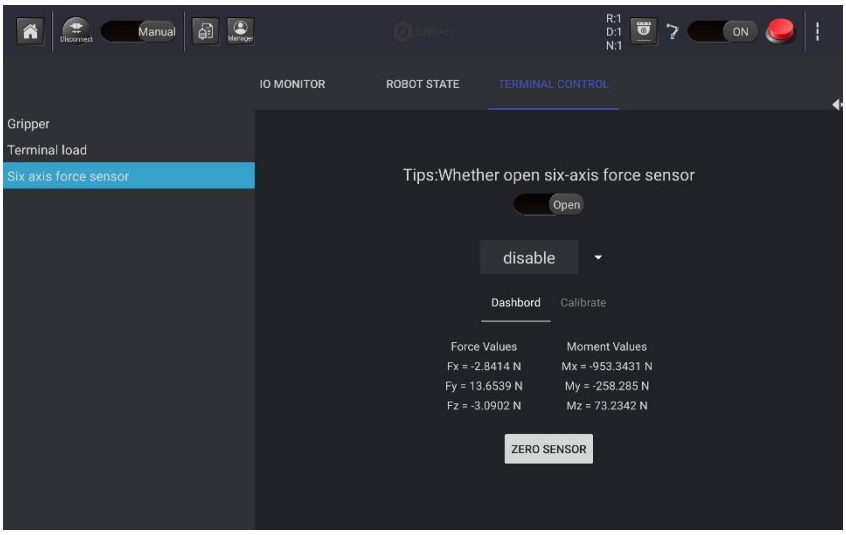


Figure 5.31 Setting Six-axis Force Sensor

- Calibration

Step 1 Drag robot to make the X-axis shown on the six-axis force sensor down according to the tips on the **Calibrate** tab, and click **CONTINUE** to calibrate X-axis of the six-axis force sensor.

Figure 5.32 Calibrate X-axis

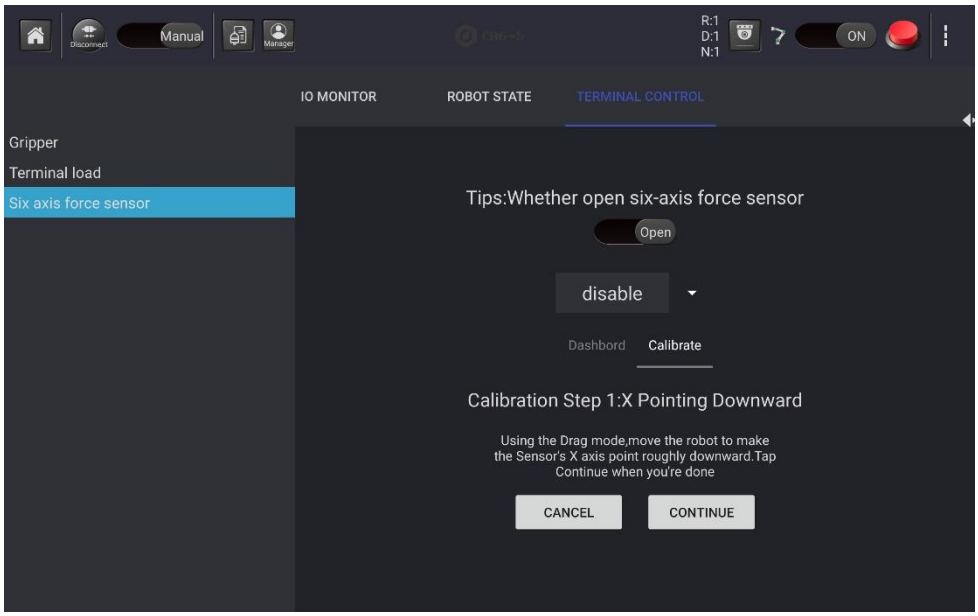


Figure 5.33 Calibrate X-axis

Step 2 Drag robot to make the Y-axis down shown on the six-axis force sensor down according to the tips on the **Calibrate** tab, and click **CONTINUE** to calibrate Y-axis of the six-axis force sensor.

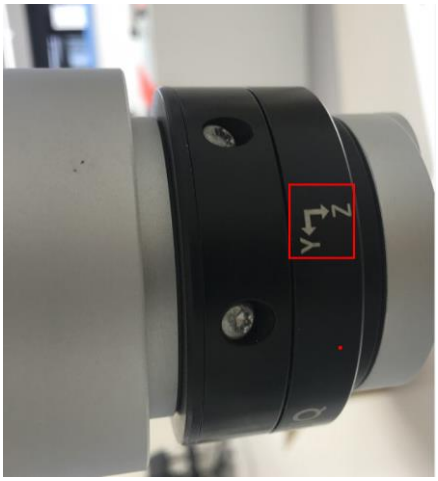


Figure 5.34 Calibrate Y-axis

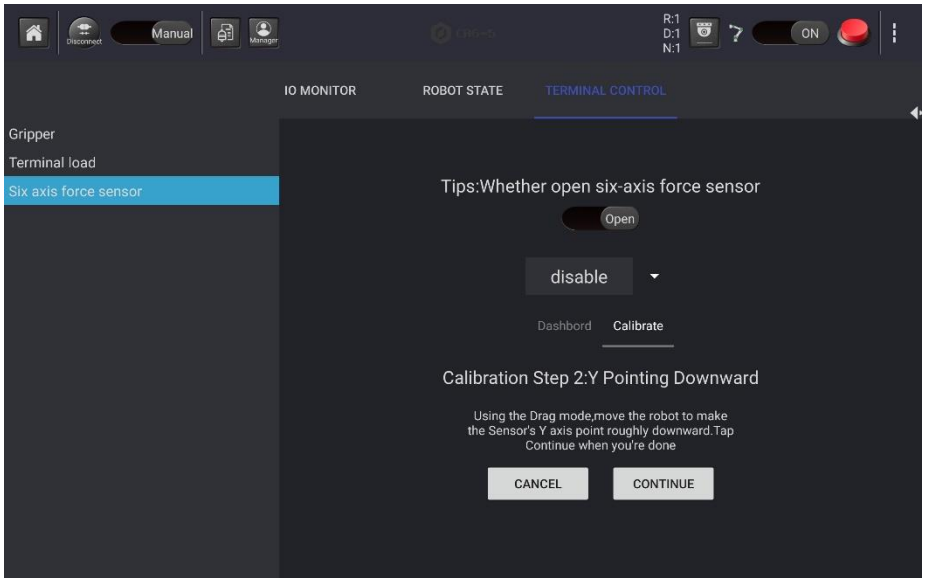


Figure 5.35 Calibrate Y-axis

Step 3 Drag robot to make the Z-axis down shown on the six-axis force sensor down according to the tips on the **Calibrate** tab, and click **CONTINUE** to calibrate Z-axis of the six-axis force sensor.

Figure 5.36 Calibrate Z-axis

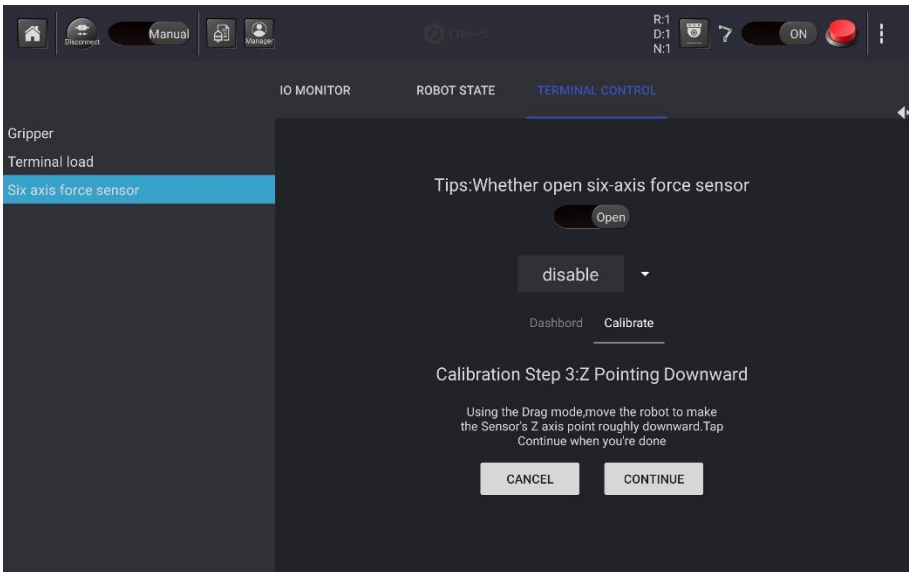


Figure 5.37 Calibrate Z-axis

After the calibration is completed, the programming instruction can be called in the programming module to control the six-dimensional force sensor. For details, please refer to 6.10Six-axis Force Sensor .

5.4 Programming

5.4.1 Program Description

CR5 robot supports script programming, graphics programming, and Blockly programming which use the Lua language as the programming language.

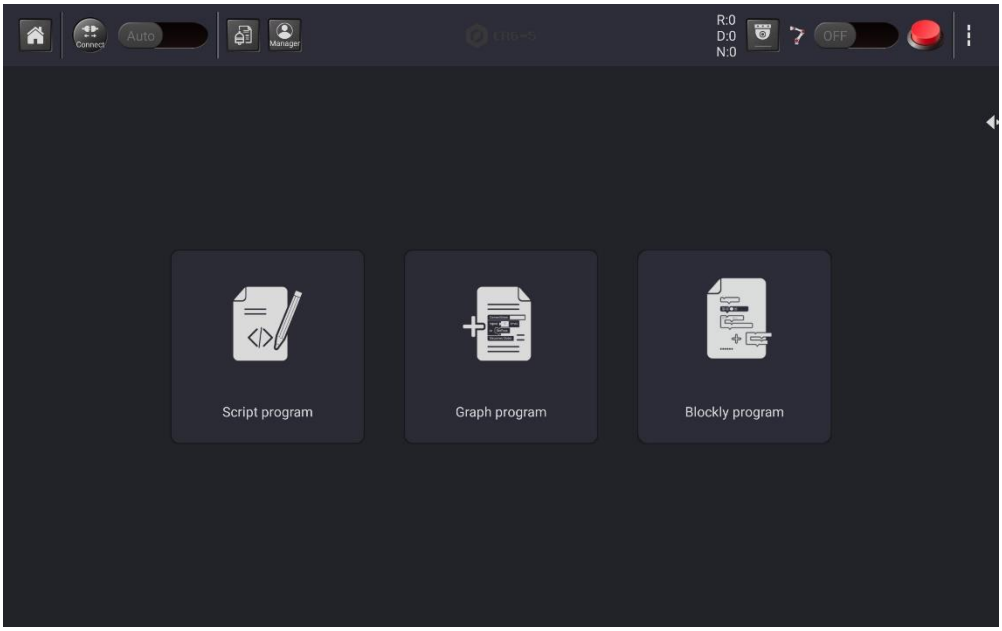


Figure 5.38 Program Description

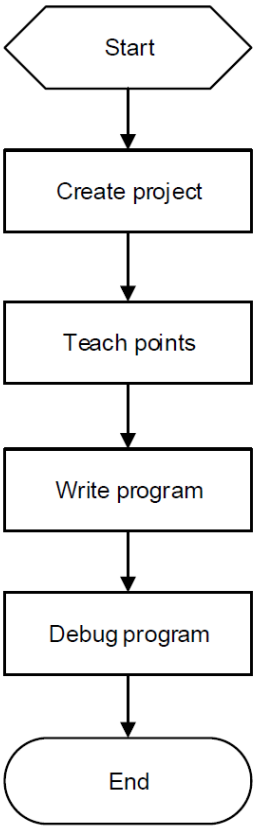


Figure 5.39 Programming process

- **Script Programming**

Script programming uses Lua as the programming language, as shown in Figure 5.40.

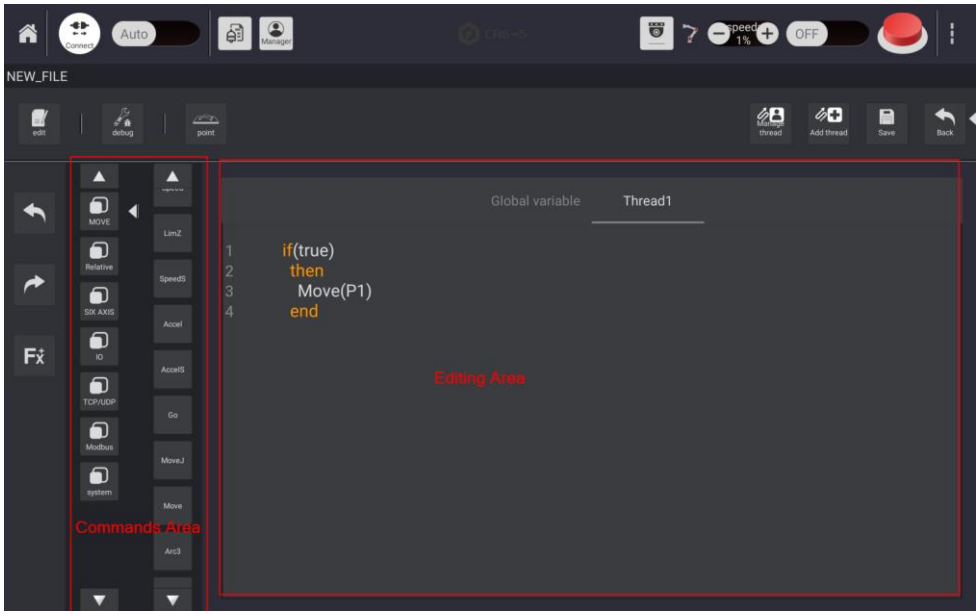
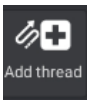


Figure 5.40 Script programming

Table 5.5 Button description

Button	Description
	Click this button to write program
	Click this button to debug program
	Click this button to set teaching points.
	Click this button to delete thread
	Click this button to add thread
	Click this button to save program

- **Tree Programming**

Tree programming is programmed in a way similar to building blocks, making it easier for users to operate and get started.

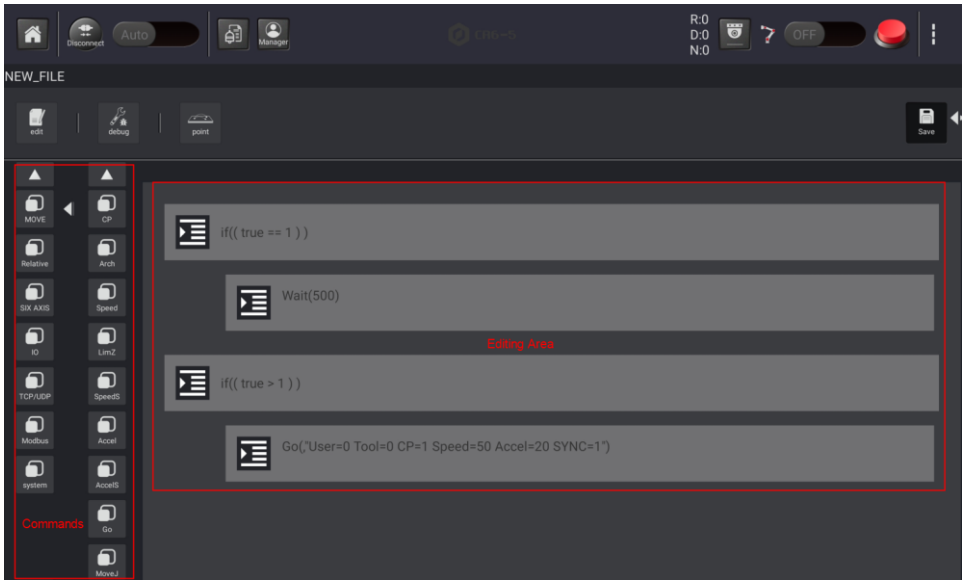


Figure 5.41 Tree programming

- **Blockly Program**

Blockly is a graphical programming platform developed by YueJiang based on Google's open-source platform. The operation of the robot can be programmed through puzzles which is intuitive

and easy to understand.

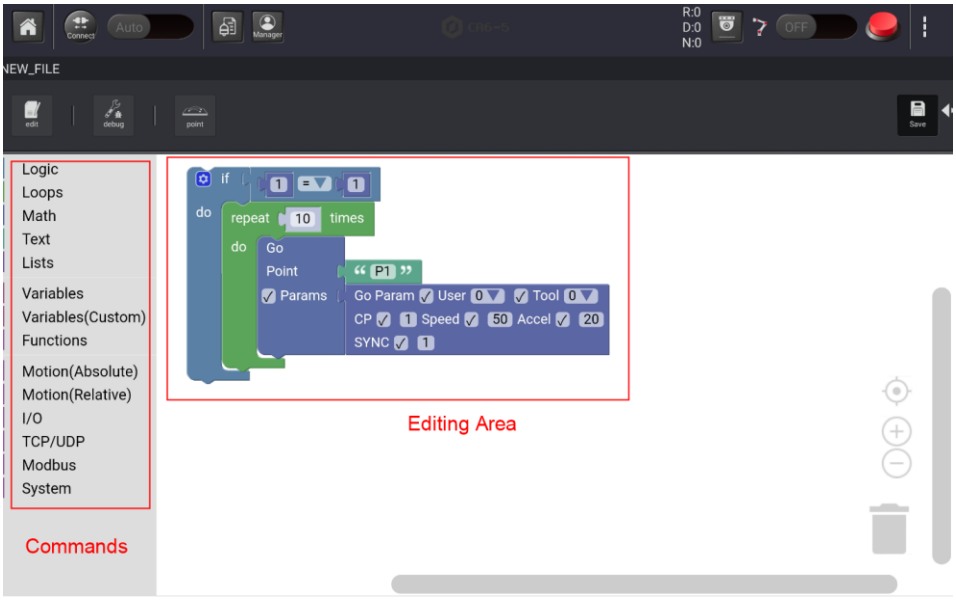


Figure 5.42 Blockly programming

5.4.2 Program Example

This section takes script programming as an example to describe how to program. We call **Go** command to make the robot move between point P1 and point P2 circularly.

Prerequisites

Robot has connected to APP.

Procedure

Step 1 Click **New File** in **Script program** page, as shown in Figure 5.43 .

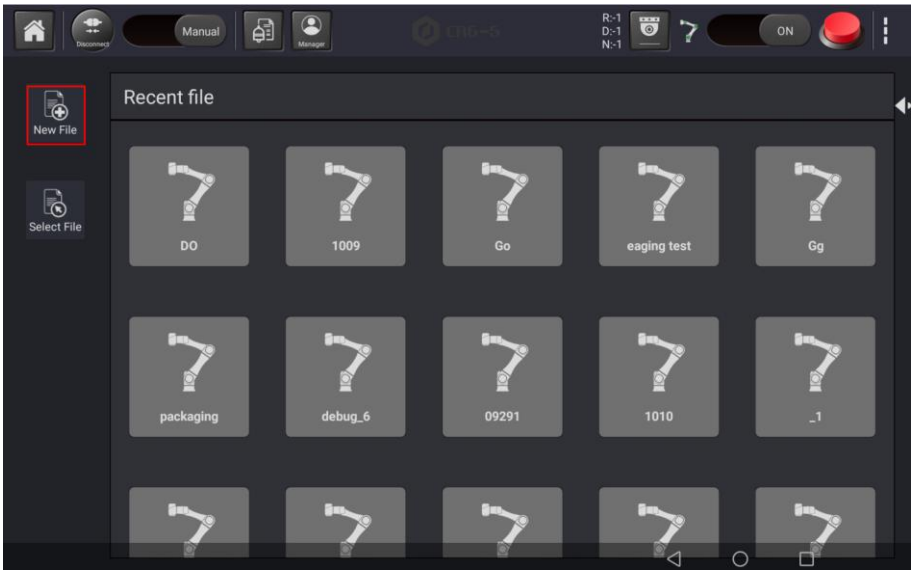


Figure 5.43 Create new file

Step 2 Jog robot to a point and click  to add the point on the **point** page. The saved points are shown in Figure 5.44.

Arm is the arm orientation, **Tool** is tool coordinate system, **User** is User coordinate system.

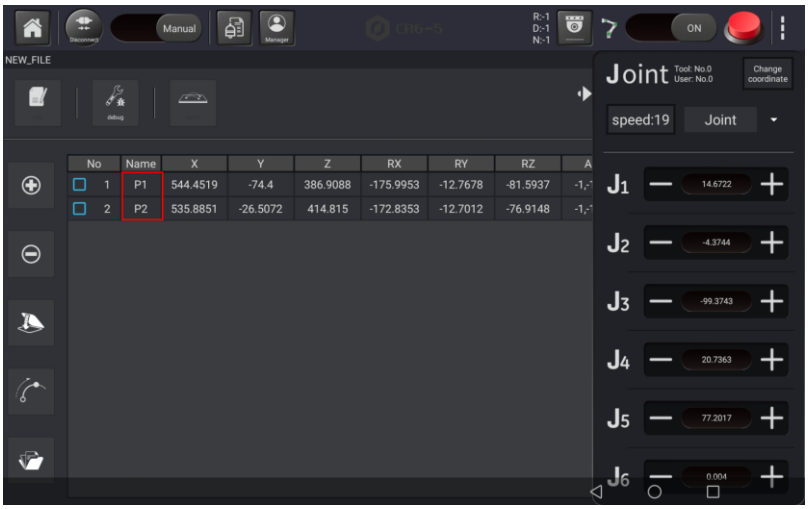


Figure 5.44 Save teaching points

Table 5.6 Button description

Button	Description
	Add a teaching point
	Delete a teaching point
	Cover a point you can select a teaching point and click **Cover** to cover the current teaching point
	Move to a point Select a point, and long press this button to make robot move to this point
	Download teaching points

Step 3 Click **Edit** to edit program, click  to select **Move>Go**, add two Go commands and select Point P1 and P2 in the two **Go** commands respectively, as shown in Figure 5.45.

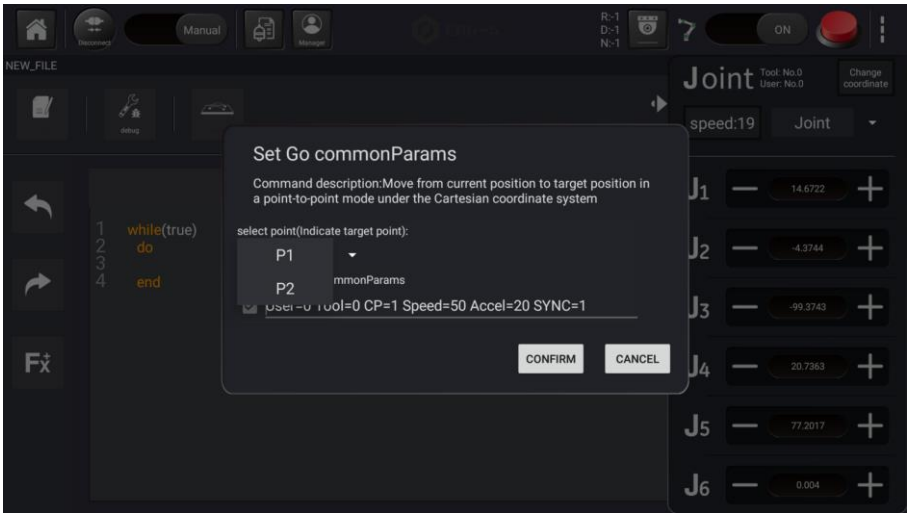


Figure 5.45 Edit program

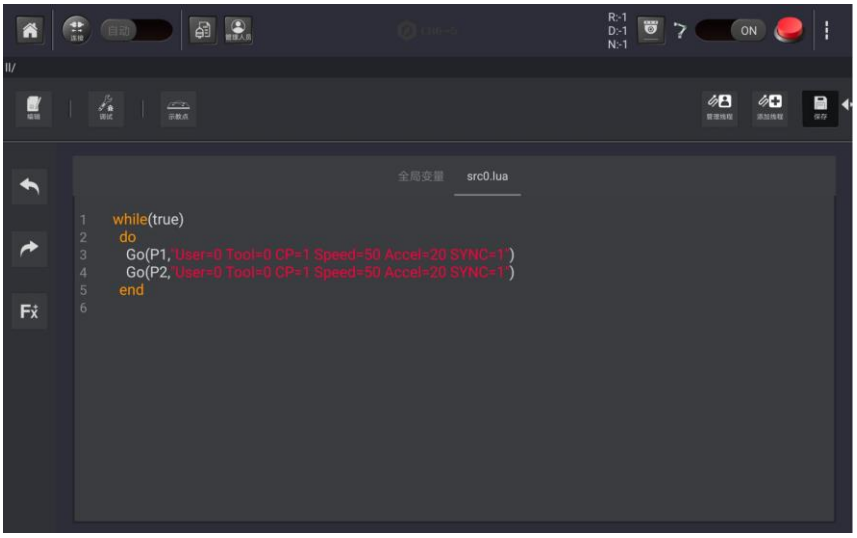


Figure 5.46 Finish writing

Step 4 Click **Save** in manual mode.

Step 5 Click **debug** and set running speed. Click ▶ to run this program, the robot will move between point **P1** and point **P2**. Also, you can click ■ to stop it.

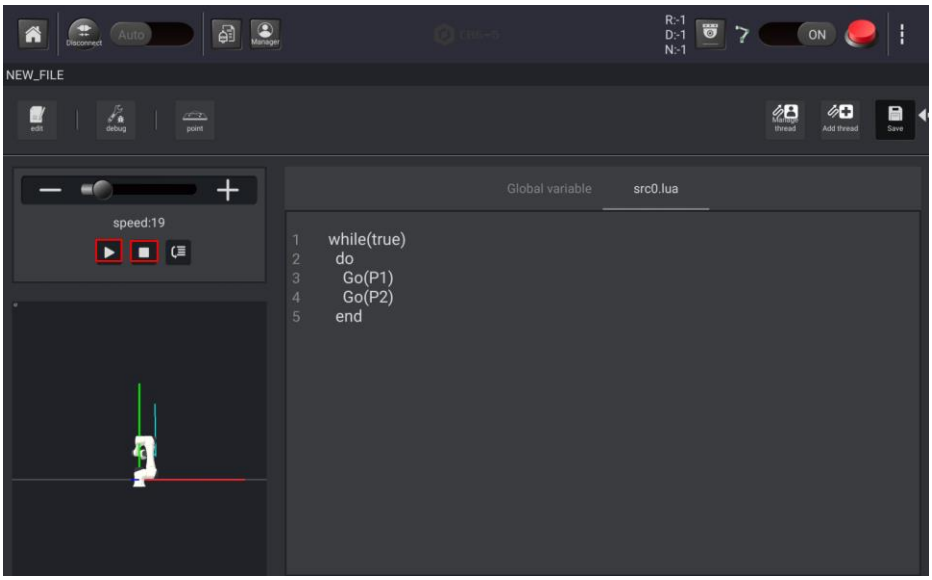


Figure 5.47 Debug

Table 5.7 Button description

Button	Description
	Start button, click it to run program
	Pause button, click it to pause program
	Run a program step by step, you can run a program step by step by setting a breakpoint

6. Program Language

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

6.1 Arithmetic Operators

Table 6.1 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

6.2 Relational Operator

Table 6.2 Relational Operator

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

6.3 Logical Operators

Table 6.3 Logical operator

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

6.4 General Keywords

Table 6.4 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

6.5 General Symbol

Table 6.5 General symbol

Command	Description
#	Get the length of the array **table**

6.6 Processing Control Commands

Table 6.6 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)

6.7 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
```

- Define a joint coordinate point, of which **R** sets to **1**, **D** sets to **-1**, **N** sets to **0**, **Cfg** sets to **1**, the User and Tool coordinate systems are both default coordinate systems.

```
P = {armOrientation = {1, 1, -1, 1}, joint = {20,10,22,2.14,0.87,3.85}, tool = 0, user = 0}
```

- Global variable

```
flag = 0
```

6.8 Motion Commands

Table 6.7 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	Robot moves from the current position to a target position in the **Move** 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 axes offset under the Cartesian coordinate system to return a new Cartesian coordinate point
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

Command	Description
	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 6.8 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	Required parameter: P: Indicate target point, which is user-defined or obtained from the **TeachPoint** page. Only Cartesian coordinate points are supported Optional parameter: • User: Indicate User coordinate system. Value range: 0 - 9 • Tool: Indicate Tool coordinate system. Value range: 0-9 • CP: Whether to set continuous path function. Value range: 0- 100 • Speed: Velocity rate. Value range: 1 - 100 • Accel: Acceleration rate. Value range: 1 -100 • SYNC: Synchronization flag. Value range: 0 or 1. If **SYNC** is **0**, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If **SYNC** is **1**, it indicates synchronous execution. After calling this command, it will not return until it is executed completely
Example	The robot moves to point P1 as the default setting Go(P1)

Table 6.9 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	Required parameter: P: Indicate the joint angle of the target point, which cannot be obtained from the **TeachPoint** page. You need to define the joint coordinate point before calling this command Optional parameter: • CP: Whether to set continuous path function. Value range: 0 - 100 • Speed: Velocity rate. Value range: 1 - 100 • Accel: Acceleration rate. Value range: 1 - 100 • SYNC: Synchronization flag. Value range: 0 or 1. If **SYNC** is **0**, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If **SYNC** is **1**, it indicates synchronous execution. After calling this command, it will not return until it is executed completely
Example	local P = {joint={0,-0.0674194,0,0}} MoveJ(P)

Table 6.10 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	Required parameter: P: Indicate the target point, which is user-defined or obtained from the **TeachPoint** page. Only Cartesian coordinate points are supported Optional parameter: • User: Indicate User coordinate system. Value range: 0 - 9 • Tool: Indicate Tool coordinate system. Value range: 0 - 9 • CP: Whether to set continuous path function. Value range: 0 - 100 • SpeedS: Velocity rate. Value range: 1 - 100 • AccelS: Acceleration rate. Value range: 1 - 100 • SYNC: Synchronization flag. Value range: 0 or 1. If **SYNC** is **0**, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If **SYNC** is **1**, it indicates synchronous execution. After calling this command, it will not return until it is executed completely
Example	The robot moves to point P1 as the default setting Move(P1)

Table 6.11 Arc3 command

Function	Arc3(P1,P2, " User=*1* Tool=*2* CP=*1* SpeedS=*50* AccelS=*20* *SYNC*=1")

Description	Move from the current position to a target position in an arc interpolated mode under the Cartesian coordinate system This command needs to combine with other motion commands, to obtain the starting point of an arc trajectory
Parameter	Required parameter: • P1: Middle point, which is user-defined or obtained from the **TeachPoint** page. Only Cartesian coordinate points are supported • P2: End point, which is user-defined or obtained from the **TeachPoint** page. Only Cartesian coordinate points are supported Optional parameter: • User: Indicate User coordinate system. Value range: 0 - 9 • Tool: Indicate Tool coordinate system. Value range: 0 - 9 • CP: Whether to set continuous path function. Value range: 0 - 100 • SpeedS: Velocity rate. Value range: 1 - 100 • AccelS: Acceleration rate. Value range: 1 – 100 • SYNC: Synchronization flag. Value range: 0 or 1. If **SYNC** is **0**, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If **SYNC** is **1**, it indicates synchronous execution. After calling this command, it will not return until it is executed completely
Example	While true do Go(P1) Arc3(P2,P3) end The robot cycles from point P1 to point P3 in the arc interpolated mode

Table 6.12 Jump command

Function	Jump(P," User=*1* Tool=*2* SpeedS=*50* AccelS=*20* Start=*10* Zlimit=*80* End=*50* *SYNC*=1")
Description	The robot moves from the current position to a target position in the **Move** mode. The trajectory looks like a door
Parameter	Required parameter: P: Indicate the target point, which is user-defined or obtained from the **TeachPoint** page. Only Cartesian coordinate points are supported. Also, the target point cannot be higher than **Zlimit**, to avoid an alarm about JUMP parameter error Optional parameter: • User: Indicate User coordinate system. Value range: 0 - 9 • Tool: Indicate Tool coordinate system. Value range: 0 - 9 • SpeedS: Velocity rate. Value range: 1 - 100

	• AccelS: Acceleration rate. Value range: 1 - 100 • Arch: Arch index. Value range: 0 - 9 • Start: Lifting height • Zlimit: Maximum lifting height • End: Dropping height • SYNC: Synchronization flag. Value range: 0 or 1. If **SYNC** is **0**, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If **SYNC** is **1**, it indicates synchronous execution. After calling this command, it will not return until it is executed completely
Example	Jump(P1) The robot moves to point P1 in the Jump mode

⚠NOTICE

The lifting height and dropping height cannot be higher than Zlimit, to avoid an alarm on JUMP parameter error. For details about JUMP, please see *2.5.1.1 Joint Interpolated Motion*.

Table 6.13 Circle3 command

Function	Circle3(P1,P2,Count, " User=*1* Tool=*2* CP=*1*SpeedS=*50* AccelS=*20*")
Description	Move from the current position to a target position in a circular interpolated mode under the Cartesian coordinate system This command needs to combine with other motion commands, to obtain the starting point of an arc trajectory
Parameter	Required parameter • P1: Middle point, which is user-defined or obtained from the **TeachPoint** page. Only Cartesian coordinate points are supported • P2: End point, which is user-defined or obtained from the **TeachPoint** page. Only Cartesian coordinate points are supported • Count: Number of circles. Value range: 1 - 999 Optional parameter: • User: Indicate User coordinate system. Value range: 0 - 9 • Tool: Indicate Tool coordinate system. Value range: 0 - 9 • CP: Whether to set continuous path function. Value range: 0 - 100 • SpeedS: Velocity rate. Value range: 1 - 100 • AccelS: Acceleration rate. Value range: 1 - 100 • SYNC: Synchronization flag. Value range: 0 or 1. If **SYNC** is **0**, it indicates asynchronous

	execution, this command has a return immediately after calling it, regardless of the command process. If **SYNC** is **1**, it indicates synchronous execution. After calling this command, it will not return until it is executed completely
Example	Go(P1) Circle3(P2,P3,1) Robot cycles from point P1 to point P3 in the circular interpolated mode

Table 6.14 RP command

Function	RP(P1, {OffsetX, OffsetY, OffsetZ})
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	• P1: Indicate the current Cartesian coordinate point, which is user-defined or obtained from the **TeachPoint** page. Only Cartesian coordinate points are supported • OffsetX，OffsetY，OffsetZ: X, Y, Z axes offset in the Cartesian coordinate system Unit: mm
Return	Cartesian coordinate point
Example	P2=RP(P1, {50,10,32}) Move(P2) or Move(RP(P1, {50,10,32}))

Table 6.15 RJ command

Function	RJ(P1, {Offset1, Offset2, Offset3, Offset4, Offset5, Offset6})
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 **MoveJ** command
Parameter	• P1: Indicate the current joint coordinate point, which cannot be obtained from the **TeachPoint** page. You need to define the joint coordinate point before calling this command • Offset1~Offset6: J1 - J6 axes offset. Unit: °
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 6.16 GoR command

Function	GoR({OffsetX, OffsetY, OffsetZ},” 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	Required parameter: OffsetX, OffsetY, OffsetZ: X, Y, Z axes offset in the Cartesian coordinate system Unit: mm Optional parameter: • User: Indicate User coordinate system. Value range: 0 - 9 • Tool: Indicate Tool coordinate system. Value range: 0-9 • CP: Whether to set continuous path function. Value range: 0- 100 • Speed: Velocity rate. Value range: 1 - 100 • Accel: Acceleration rate. Value range: 1 -100 • SYNC: Synchronization flag. Value range: 0 or 1. If **SYNC** is **0**, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If **SYNC** is **1**, it indicates synchronous execution. After calling this command, it will not return until it is executed completely
Example	Go(P1) GoR({10,10,10},"Accel=100 Speed=100 CP=100")

Table 6.17 MoveJR command

Function	MoveJR({Offset1, Offset2, Offset3, Offset4, Offset5, Offset6},” 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	Required parameter: Offset1 - Offset6: J1 - J6 axes offset. Unit: ° Optional parameter: • CP: Whether to set continuous path function. Value range: 0 - 100 • Speed: Velocity rate. Value range: 1 - 100 • Accel: Acceleration rate. Value range: 1 - 100 • SYNC: Synchronization flag. Value range: 0 or 1. If **SYNC** is **0**, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If **SYNC** is **1**, it indicates synchronous execution. After calling this command, it will not return until it is executed completely
Example	Go(P1)

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

Table 6.18 MoveR command

Function	MoveR({OffsetX, OffsetY, OffsetZ},” 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	Required parameter: OffsetX, OffsetY, OffsetZ: X, Y, Z axes offset in the Cartesian coordinate system Unit: mm Optional parameter: • User: Indicate User coordinate system. Value range: 0 - 9 • Tool: Indicate Tool coordinate system. Value range: 0 - 9 • CP: Whether to set continuous path function. Value range: 0 - 100 • SpeedS: Velocity rate. Value range: 1 - 100 • AccelS: Acceleration rate. Value range: 1 - 100 • SYNC: Synchronization flag. Value range: 0 or 1. If **SYNC** is **0**, it indicates asynchronous execution, this command has a return immediately after calling it, regardless of the command process. If **SYNC** is **1**, it indicates synchronous execution. After calling this command, it will not return until it is executed completely
Example	Go(P1) MoveR({20,20,20},"AccelS=100 SpeedS=100 CP=100")

6.9 Motion Parameter Commands

Table 6.19 Motion parameter command

Command	Description
Accel	Set the acceleration rate. This command is valid only when the motion mode is **Go**, **Jump**, or **MoveJ**
AccelS	Set the acceleration rate. This command is valid only when the motion mode is **Move**, **Jump**, **Arc3**, or **Circle3**
Speed	Set the velocity rate. This command is valid only when the motion mode is **Go**, or **MoveJ**
SpeedS	Set the velocity rate. This command is valid only when the motion mode is **Move**, **Jump**, **Arc3**, or **Circle3**

Command	Description
Arch	Set the index of sets of parameters (**StartHeight**, **zLimit**, **EndHeight**) in **Jump** mode
CP	Set the continuous path function
LimZ	Set the maximum lifting height in the **Jump** mode

Table 6.20 Accel command

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

Table 6.21 AccelS command

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

Table 6.22 Speed command

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

Table 6.23 SpeedS command

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

Table 6.24 Arch command

Function	Arch(Index)
Description	Set the index of sets of parameters (**StartHeight**, **zLimit**, **EndHeight**) in the **Jump** mode The sets of parameters need to be set on the **Setting** > **PlaybackArch** of the APP.
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 **Setting** > **PlaybackArch** of the APP
Example	Arch(1) Jump(P1)

Table 6.25 CP command

Function	CP(R)
Description	Set the continuous path rate. This command is valid only when the motion mode is **Go**, **Move**, **Arc3**, **Circle3**, or **MoveJ**
Parameter	R: Continuous path rate. Value range: 0 -100 **0** 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

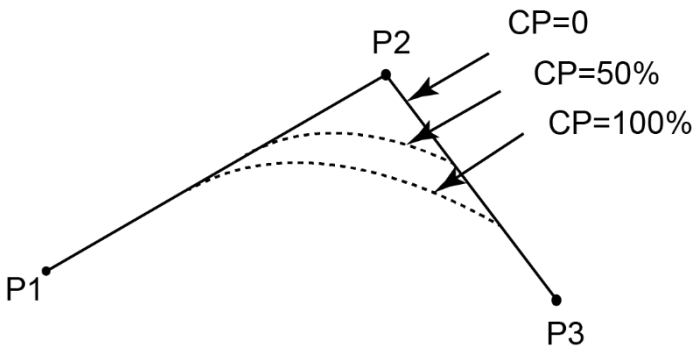


Figure 6.1 Continuous path

Table 6.26 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 ")

6.10 Six-axis Force Sensor Commands

Table 6.27 Six-axis force sensor command

Command	Description
SixForceHome	Six-axis force sensor homing command
Spiral	Six-axis force sensor spiral motion command
Rotation	Six-axis force sensor rotation motion command
Linear	Six-axis force sensor linear motion command

Table 6.28 Six-axis force sensor homing command

Function	SixForceHome
Description	Homing six-axis force sensor
Parameter	None
Example	SixForceHome(): execute the command to home six-axis force sensor

Table 6.29 Six-axis force sensor spiral command

Function	Spiral(P, User, Tool, Direction, SpeedC, Force, Insertion, Perturn, PeckMode, MaxValue)
Description	The robot arm performs a spiral motion between the current position and the specified position to find the hole position. The specified point needs to be closer to the hole position, which is the starting point for hole position exploration.
Parameter	• P: the specified position • User: User coordinate system, value range: 0~9 • Tool: Tool coordinate system, Value range: 0~9 • Direction: Jack direction (0: Forward, 1: Reverse) • SpeedC: Jack speed(mm/s) • Force: Spiral threshold (N) • Insertion: Jack threshold (N) • Perturn: Spiral radius (mm) • PeckMode: Point contact mode (ON/OFF) • MaxValue: Maximum spiral radius (mm)
Example	Spiral(P1,User=1 Tool=2 Dirction=0 SpeedC=5 Force =10 Insertion=3 Perturn=0.7 PeckMode=OFF MaxValue =5") Do a spiral motion between the current position and P1 to find the hole position. When the resistance in the direction of the jack is greater than the Force threshold, the robot performs a spiral motion to explore the hole position. When the resistance in the direction of the jack is less than the Insertion threshold, the robot moves in the direction of the jack to perform the jack work.

Table 6.30 Six-axis force sensor rotation command

Function	Rotation (P, User, Tool, Direction, SpeedC, Force, RotationSpeed, MaxTorque, PeckMode, MaxValue)
Description	The robotic arm rotates between the current position and the specified position to find the hole position. The specified point needs to be close to the hole position, which is the starting point for hole position exploration.

Parameter	• P: the specified position • User: User coordinate system, value range: 0~9 • Tool: Tool coordinate system, Value range: 0~9 • Direction: Jack direction (0: Forward, 1: Reverse) • SpeedC: Jack speed(mm/s) • Force: Rotation threshold (N) • RotationSpeed: Rotation speed (°/s) • MaxTorque: Maximum torque (Nm) • PeckMode: Point contact mode (ON/OFF) • MaxValue: Maximum spiral radius (mm)
Example	Rotation (P1, "User=1 Tool=2 Dirction=0 SpeedC =5 Force =10 RotationSpeed=5 MaxTorque=1 PeckMode=OFF MaxValue =45") Do a rotation motion between the current position and P1 to find the hole position. When the resistance in the direction of the jack is greater than the Force threshold, the robot performs a rotation motion to explore the hole position. When the resistance in the direction of the jack is less than the Force threshold, the robot moves in the direction of the jack to perform the jack work.

Table 6.31 Six-axis force sensor linear command

Function	Linear (User, Tool, Direction, SpeedC, Force, MaxValue)
Description	The robot arm makes a linear jack movement in the direction of the hole
Parameter	• User: User coordinate system, value range: 0~9 • Tool: Tool coordinate system, Value range: 0~9 • Direction: Jack direction (0: Forward, 1: Reverse) • SpeedC: Jack speed(mm/s) • Force: Rotation threshold (N) • MaxValue: Maximum spiral radius (mm)
Example	Linear("User=1 Tool=2 Dirction=0 SpeedC =5 Force=10 MaxValue=45") Do a linear jack movement at the current hole position. When the resistance in the insertion direction is greater than the Force threshold, the insertion is considered complete.

6.11 Input/output Commands

Table 6.32 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 robot system supports two kinds of commands: Immediate command and queue command:

- Immediate command: The robot system 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 robot system receives a command, this command will be pressed into the internal command queue. The robot system will execute commands in the order in which the commands were pressed into the queue.

Table 6.33 Digital input command

Function	DI(*index*)
Description	Get the status of the digital input port
Parameter	index: Digital input index. Value range: 1 - 16
Return	• When an index is set in the DI function, **DI(index)** returns the status (ON/OFF) of this speicified input port • When there is no index in the DI function, **DI()** returns the status of all the input ports, which are saved in a table 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 **di.num** and **di.value[n]**
Example	if (DI(1))==ON then Move(P1) end The robot moves to point P1 when the status of the digital input port **1** is **ON**

Table 6.34 Digital output command (Queue command)

Function	DO(*index, ON | OFF*)

Description	Set the status of digital output port (Queue command)
Parameter	• index: Digital output index. Value range: 1- 24 • ON/OFF: Status of the digital output port. ON: High level; OFF: Low level
Example	DO(1,ON) Set the status of the digital output port 1to **ON**

Table 6.35 Digital output command (Immediate command)

Function	DOExecute(*index, ON | OFF*)
Description	Set the status of digital output port (Immediate command)
Parameter	• index: Digital output index. Value range: 1 - 24 • ON/OFF: Status of the digital output port. ON: High level; OFF: Low level
Example	DOExecute(1,OFF) Set the status of the digital output port 1 to **OFF**

6.12 Program Managing Commands

Table 6.36 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 6.37 Wait command

Function	Wait(*time*)
Description	Set the delay time for robot motion commands
Parameter	time: Delay time. Unit: ms
Example	Go(P1) Wait(1000) Wait for 1000ms after the robot moves to point P1

Table 6.38 Sleep command

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

Table 6.39 Pause command

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

Table 6.40 Star timing command

Function	ResetElapsedTime()
Description	Start timing after all commands before this command are executed completely. Use in conjunction with ElapsedTime() command For example: Get the execution time that a piece of code takes
Parameter	None
Return	None
Example	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)

Table 6.41 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	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)

Table 6.42 Get current time command

Function	Systime()

Description	Get the current time
Parameter	None
Return	Current time
Example	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)

6.13 Pose Getting Command

Table 6.43 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	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} -- Lift a certain height Go(liftPose,"Speed=100 Accel=100") Go(P1)

Table 6.44 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	local armPose local joint = GetAngle() --Get the current pose local liftPose = {armOrientation = armPose , joint = {joint.joint[1], joint.joint[2], joint.joint[3], joint.joint[4]}, tool = 0, user = 0}

6.14 TCP

Table 6.45 Create TCP command

Function	err, socket = TCPCreate(*isServer*, *IP*, *port*)
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, **port** cannot be set to 502 and 8080. Otherwise, it will be in conflict with the Modbus default port or the port used in the conveyor tracking application, causing the creation to fail
Return	err: 0: TCP network is created successfully 1: TCP network is created failed Socket: Socket object
Example	Please refer to Program 6.1 and Program 6.2

Table 6.46 TCP connection command

Function	TCPStart(*socket*, *timeout*)
Description	Connect a client to a server with the TCP protocol
Parameter	socket: Socket object timeout: Wait timeout. Unit: s. If **timeout** is 0, the connection is still waiting. If not, after exceeding the timeout, the connection is exited.

Return	• 0: TCP connection is successful • 1: Input parameters are incorrect • 2: Socket object is not found • 3: Timeout setting is incorrect • 4: If the robot is set as a client, it indicates that the connection is wrong. If the robot is set as a server, it indicates that receiving data is wrong
Example	Please refer to Program 6.1 and Program 6.2

Table 6.47 Receive data command

Function	err, Recbuf = TCPRead(*socket*, *timeout*, *type*)
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 **timeout** is 0 or is not set, this command is a block reading. Namely, the program will not continue to run until receiving data is complete. If not, after exceeding the timeout, the program will continue to run regardless of whether receiving data is complete type: Buffer type. If **type** is not set, the buffer format of **RecBuf** is a table. If **type** is set to **string**, the buffer format is a string
Return	err: 0: Receiving data is successful 1: Receiving data is failed Recbuf: Data buffer
Example	Please refer to Program 6.1 and Program 6.2

Table 6.48 Send data command

Function	TCPWrite(*socket*, *buf*, *timeout*)
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 **timeout** is 0 or not set, this command is a block reading. Namely, the program will not continue to run until sending data is complete. If not, after exceeding the timeout, the program will continue to run regardless of whether sending data is complete

Return	0: Sending data is successful 1: Sending data is failed
Example	Please refer to Program 6.1 and Program 6.2

Table 6.49 Release TCP network command

Function	TCPDestroy(*socket*)
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 6.1 and Program 6.2

⚠ 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.

Program 6.1 TCP server demo

```
local ip="192.168.5.1"                                  // 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(buf)
```

```
                else
                    print("Read error ".. err)
                    break
                end
            end
        else
            print("Create failed ".. err)
        end
        TCPDestroy(socket)
else
        print("Create failed ".. err)
end
```

Program 6.2 TCP client demo

```
local ip="192.168.5.25"                                  // 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(buf)
                else
                    print("Read error ".. err)
                    break
                end
            end
```

```
    else
        print("Create failed ".. err)
    end
    TCPDestroy(socket)
else
    print("Create failed ".. err)
end
```

6.15 UDP

Table 6.50 Create UDP network command

Function	err, socket = UDPCreate(*isServer*, *IP*, *port*)
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, **port** 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 6.3 and Program 6.4

Table 6.51 Receive data command

Function	err, Recbuf = UDPRead(*socket*, *timeout*, *type*)
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 **timeout** 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 **type** is not set, the buffer format of **RecBuf** is a table. If **type** is set to **string**, the buffer format is a string

Return	err: 0: Receiving data is successful 1: Receiving data is failed Recbuf: Data buffer
Example	Please refer to Program 6.3 and Program 6.4

Table 6.52 Send data command

Function	UDPWrite(*socket*, *buf*, *timeout*)
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 **timeout** is 0 or not set, this command is a block reading. Namely, the program will not continue to run until sending data is complete. If not, after exceeding the timeout, the program will continue to run regardless of whether sending data is complete
Return	0: Sending data is successful 1: Sending data is failed
Example	Please refer to Program 6.3 and Program 6.4

⚠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.

Program 6.3 UDP server demo

```
local ip="192.168.5.1"                                  // 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
            Go(P2)
            print(buf)
        else
            print("Read error ".. err)
            break;
        end
    end
else
    print("Create failed ".. err)
end
```

Program 6.4 UDP client demo

```
local ip="192.168.1.25"                                     // 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(buf)
        else
            print("Read error ".. err)
            break
        end
```



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

6.16 Modbus

6.16.1 Modbus Register Description

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

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 robot system. Each register has 4096 addresses. For details, please see as follows.

- Coil register

Table 6.53 Coil register description

Coil register address (e.g.: PLC)	Coil register address (Robot system)	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 6.54 Discrete input register description

Discrete input register address (e.g.: PLC)	Discrete input register address(Robot system)	Data type	Description
10001	0	Bit	Automated exit
10002	1	Bit	Ready state
10003	2	Bit	Paused state

Discrete input register address (e.g.: PLC)	Discrete input register address(Robot system)	Data type	Description
10004	3	Bit	Running state
10005	4	Bit	Alarm state
10006-10999	5-998	Bit	Reserved
11000-14096	999-4095	Bit	User-defined

- Input register

Table 6.55 Input register description

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

- Holding register

Table 6.56 Holding register description

Holding register address (e.g.: PLC)	Holding register address (Robot system)	Data type	Description
40001-41000	0-999	Byte	Reserved
41001-44095	1000-4095	Byte	User-defined

6.16.2 Command Description

Table 6.57 Rea coil register command

Function	GetCoils(*addr*, *count*)
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 Coils = GetCoils(0,5) Return: Coils={1,0,0,0,0} As shown in Table 6.52, it indicates that the robot is in the starting state

Table 6.58 Set coil register command

Function	SetCoils(*addr*, *count*, *table*)
Description	Set the coil register in the Modbus slave This command is not supported when the coil register address is from 0 to 5
Parameter	Addr: Starting address of the coils to set. Value range: 6 - 4095 count: Number of the coils to set. Value range: 0 to 4096-addr table: Coil value, stored in a table
Return	None
Example	Set 5 coils starting at address 1024 local Coils = {0,1,1,1,0} SetCoils(1024, #coils, coils)

Table 6.59 Read discrete input register command

Function	GetInBits(*addr*, *count*)
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 inBits = GetInBits(0,5) Return: inBits = {0,0,0,1,0} As shown in Table 6.53, it indicates the robot is in running state

Table 6.60 Read input register command

Function	GetinRegs(*addr*, *count*, *type*)
Description	Read the input register value with the specified data type from the Modbus slave
Parameter	addr: Starting address of the input registers. Value range: 0 - 4095 count: Number of the input registers to read. Value range: 0 ~ 4096-addr type: Data type • Empty: Read 16-bit unsigned integer (two bytes, occupy one register) • “U16”: Read 16-bit unsigned integer (two bytes, occupy one register) • “U32”: Read 32-bit unsigned integer (four bytes, occupy two registers) • “F32”: Read 32-bit single-precision floating-point number (four bytes, occupy two registers) • “F64”: Read 64-bit double-precision floating-point number (eight bytes, occupy four registers)
Return	Return a table, the first value in the table corresponds to the input register value at the starting address
Example	Example 1: Read a 16-bit unsigned integer starting at address 2048 data = GetInRegs(2048,1) Example 2: Read a 32-bit unsigned integer starting at address 2048 data = GetInRegs(2048, 1, “U32”)

Table 6.61 Read holding register command

Function	GetHoldRegs(*addr*, *count*, *type*)
Description	Read the holding register value from the Modbus slave according to the specified data type
Parameter	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 • Empty: Read 16-bit unsigned integer (two bytes, occupy one register) • “U16”: Read 16-bit unsigned integer (two bytes, occupy one register) • “U32”: Read 32-bit unsigned integer (four bytes, occupy two registers) • “F32”: Read 32-bit single-precision floating-point number (four bytes, occupy two registers) • “F64”: Read 64-bit double-precision floating-point number (eight bytes, occupy four registers)
Return	Return a table, the first value in the table corresponds to the input register value at the starting address

Example	Example 1: Read a 16-bit unsigned integer starting at address 2048 data = GetHoldRegs(2048,1) Example 1: Read a 32-bit unsigned integer starting at address 2048 data = GetInRegs(2048, 1, “U32”)

Table 6.62 Set holding register command

Function	SetHoldRegs(*addr*, *count*, *table*, *type*)
Description	Set the holding register in the Modbus slave
Parameter	addr: Starting address of the holding registers to set. Value range: 0 - 4095 count: Number of the holding registers to set. Value range: 0 to 4096-addr table: Holding register value, stored in a table type: Datatype • Empty: Read 16-bit unsigned integer (two bytes, occupy one register) • “U16”: Set 16-bit unsigned integer (two bytes, occupy one register) • “U32”: Set 32-bit unsigned integer (four bytes, occupy two registers) • “F32”: Set 32-bit single-precision floating-point number (four bytes, occupy two registers) • “F64”: Set 64-bit double-precision floating-point number (eight bytes, occupy four registers)
Return	None
Example	Example1: Set a 16-bit unsigned integer starting at address 2048 local data = {6000} SetHoldRegs(2048, #data, data, “U16”) Example2: Set a 64-bit double-precision floating-point number starting at address 2048 local data = {95.32105} SetHoldRegs(2048, #data, data, “F64”)