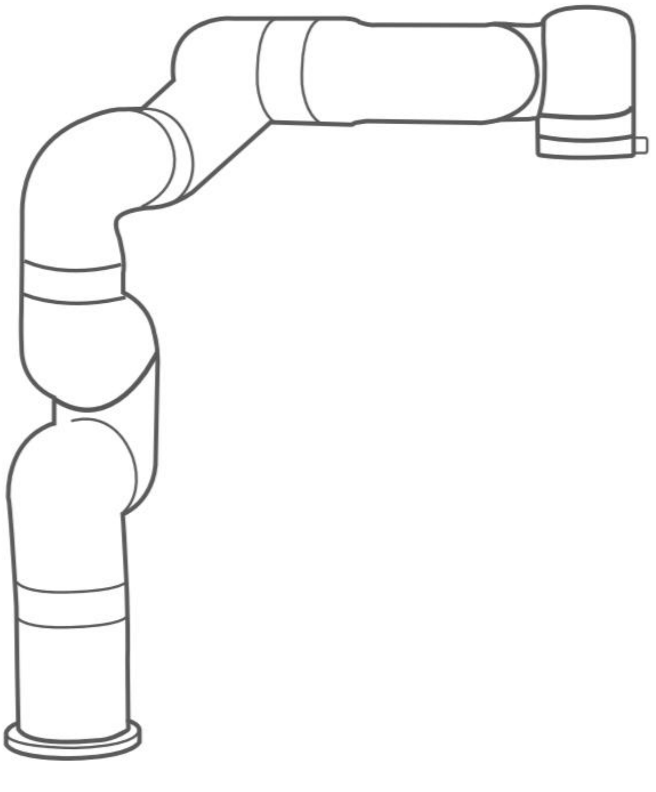
UFACTORY

xARM 7

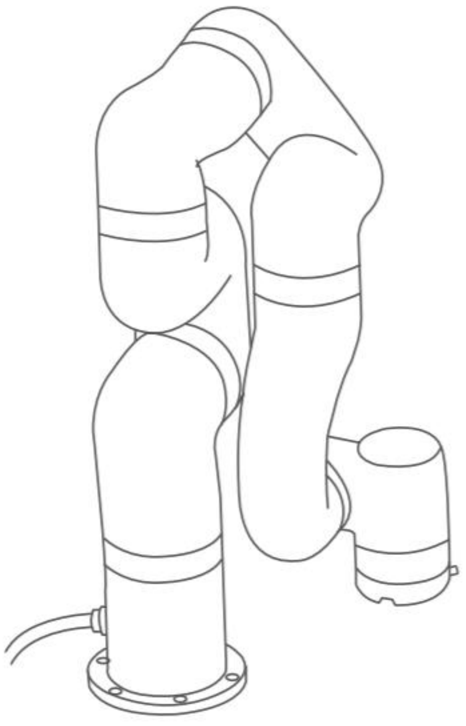
USER MANUAL



SHENZHEN UFACTORY CO.,LTD.

V1.2.2

Thanks for purchasing and using the light-weight multi-DOF(multi-degree-of-freedom) programmable robotic arm developed by UFACTORY



As a cost-effective, lightweight, easy-to-use robotic arm, xArm significantly boosts the operational efficiency with automation technology by employing a handful of SDK including python, ROS C++. xArm features custom APP, an user-friendly software xArm Studio which allows users to teach xArm to perform repetitive movement with ‘hand-guide’ teaching, with a custom APP, allowing you to interact with xArm with ease and complete complex path planning in 10 minutes while maximizing its potential to the utmost level.

Table of Contents

目录

Preface

Product Formation

Package contains:

- Robotic Arm x 1
- controller x 1
- Power cable for the controller x 1
- Power cable for the Robotic arm x 1
- Communication cable for the robotic arm x 1
- Ethernet Cable x1
- Robotic arm end effector adapter cable x1
- This Manual x 1

Additional Information

Please refer to the UFACTORY official website (https://www.ufactory.cc/) for xArm Studio software download, video tutorial, English manual .

1. Important Safety Precautions

Safety Instructions for Users

1.1. Introduction

This chapter contains important safety information, integrators and users of xArm must follow the instructions and pay special attention to the content with warning signsDue to the complexity of the robotic arm system and its degree of danger, please ensure you fully understand the content of this manual and strictly adhere to the instructions. When using SDK (Python/ROS/C++) and graphical interface (xArm Studio), please read the relevant interface instructions given in this operation manual.

UFACTORY is devoted to providing reliable and safety information, but these contents do not constitute warranties by UFACTORY. UFACTORY will not have or accept any liability, obligation or responsibility whatsoever for any loss, destruction or damage however arising from or in respect of any use or misuse of xArm.

1.2. Validity and Responsibility

The information does not cover how to design, install, and operate a complete robotic arm application, nor does it cover all peripheral equipment that can influence the safety of the complete system. The complete system must be designed and installed in accordance with the safety requirements set forth in the standards and regulations of the country where the robotic arm is installed.

The integrators of the xArm are responsible for the compliance of applicable safety laws and regulations in the country, to prevent any hazards in the operating environment.

Safety precautions include but are not limited to：

- Making a risk assessment for the complete system. Make sure to have a safe distance between human and xArm when interacting with the xArm.
- Interfacing other machines and additional safety devices if defined by the risk assessment.
- When using software for programming, please read carefully the port’s statement and set up the appropriate safety settings in the software.
- Specifying instructions for use to prevent unnecessary property damage or personal injury caused by improper operation.

1.3. Limitations on Liability Exceptions

Any information given in this manual regarding safety must not be construed as a warranty by UFACTORY that the industrial manipulator will not cause injury or damage even if all safety instructions are complied with.

1.4. Safety Alarms in this Manual

⚠ DANGER	**DANGER:** This indicates an imminently hazardous electrical situation which, if not avoided, could result in death or serious damage to the device.
⚠ WARNING	**WARNINGR:** This indicates a potential hazardous situation which, if not avoided, could result in death or serious damage to the device.
⚠ HIGH TEMPERATURE	**HIGH TEMPERATURE** This indicates a potential hot surface, if touch, could result in personal injury.
⚠ NOTICE	**NOTICE** Iif not avoided, could result in personal injury or damage to the equipment.

CAUTION

CAUTION:

If not avoided, could result in personal injury or damage to the equipment.

1.5. Safety Precautions

1.5.1. Overview

This section contains some general warnings and cautions on installation and application planning for the robotic arm. To prevent damage to the machine and associated equipment, users need to learn all the relevant descriptions and fully understand the safety precautions. Wedo not control or guarantee the relevance or completeness or usefulness of such information in this manual, users are suggested to conduct self-assessment of their specific problems.

1.5.2. General Alarms and Cautions

DANGER

1. Make sure to use the correct installation settings in this manual for the robotic arm and all the electrical equipment.
2. Please follow the instructions in this manual, installation and commissioning needs to be performed by professionals in accordance with the installation standards.
3. Make sure the robotic arm and tool are properly and securely bolted in place.
4. The integrity of device and system must be checked before each use (e.g. the operational safety and the possible damage of robotic arm and other device systems).
5. Preliminary testing and inspection for both robotic arm and perimeter protection system before production is essential.
6. The operator must be trained to guarantee a correct operation procedure when using SDK(Python/ROS/C++)

	and graphical interface xArm Studio. 7. A complete safety assessment must be recorded each time the robotic arm is re-installed and debugged. 8. When the robotic arm is in an accident or abnormal operation, the emergency stop switch needs to be pressed down to stop the movement, and the robotic arm will stop and hold the position. 9. The xArm joint module has brakes inside, it will remain manipulator's pose when a power outage occurs. 10. When the robotic arm is in operation, make sure no people or other equipment are in the working area. 11. When releasing the brakes of xArm, please take protective measures to prevent the robotic arm or operator from damage or injury. 12. When connecting the xArm with other machinery, it may increase risk and result in dangerous consequences. Make sure a consistent and complete safety assessment is conducted for the installation system.
HIGH TEMPERATURE	1. The robotic arm and controller box generates heat during operation. Do not handle or touch the robotic arm and controller box while in operation or immediately after the operation. 2. Never stick fingers behind the internal cover of the controller box.
CAUTION	1. Make sure the robotic arm's joints and tools are installed properly and safely, check the status for all circuits. 2. Make sure that there is enough space for the manipulator to move freely. 3. Make sure that there is no obstacle in the robotic arm's working space. 4. The controller must be placed outside the working range of the robotic arm to ensure the emergency stop button can be pressed once an emergency occurs. 5. If the robotic arm is in operation and needs an emergency stop, make sure the restart/reset will not touch any obstacle. 6. Do not modify the robotic arm. Any modification may lead

	to unpredictable danger to the integrator. The robotic arms authorize restructuring need in accordance with the latest version of all relevant service manuals. If the robotic arm is modified or altered in any way, UFACTORY (Shenzhen) Technology Co.,Ltd disclaims all liability. 7. Users needs to check the collision and protection measures before transportation.
NOTICE	When the xArm cooperates with other machinery, a comprehensive safety assessment of the entire collaboration system should be performed. It is recommended that the equipment that may cause mechanical damage must be placed outside the working range of xArm during application planning.

1.5.3. Operator Safety

In the operation of the robotic arm system, we must ensure the safety of the operators first. The general precautions are listed in the table below. Please take appropriate measures to ensure the safety of operators.

CAUTION

1. Each operator who uses the robotic arm system should be trained through a training course hosted by UFACTORY (Shenzhen) Technology Co., Ltd. Users should fully understand the standardized operating procedures with the robotic arm, and the solution to the robotic arm running error.
2. When the device is running, even if the robotic arm seems to stop, it is possible that the robotic arm is waiting for the signal and in the upcoming action status. Even in such a state, it should be considered as the robotic arm is in action.
3. A line should be drawn to mark the range of motion of the robotic arm to let the operator acknowledge the robotic arm, including its end tools, (such as gripper and suction cup, etc) operating range.
4. Check the robotic arm regularly to prevent loosening of the bolts that may cause undesirable consequences.
5. Be careful when the robotic arm is running too fast.
6. Be careful about dropping items that can be caused by accidental power off or unstable clamping of the robotic arm.
7. xArm has no collision detection function, please make sure there are no obstacles in the operation of the robotic arm.

2. Precautions for Transportation

Keep the original packaging when transporting. Store the packaging material in a dry place for future repackaging and moving the robotic arm.

When repacking, the robotic arm needs to be returned to the zero point.

When moving the robotic arm from packaging to the installation position, hold the robotic arm in place until all mounting bolts are safely tightened at the base of the robotic arm.

CAUTION

1. When lifting the arm, make sure to hold it steadily, and use the appropriate lifting equipment.
2. All regional and national guidelines should be followed. UFACTORY (Shenzhen) Technology Co., Ltd. is not responsible for any damage caused during the transportation of the equipment.

3. Maintenance and Repair

Maintenance must be performed by authorized integrators or UFACTORY (Shenzhen) Technology Co., Ltd. All parts returned to UFACTORY (Shenzhen) Technology Co., Ltd. shall be returned according to the service manual.

Make sure to reach the safety requirements of maintenance and repair, follow all regional and national guidelines and test whether all safety functions work normally.

The purpose of maintenance and repair is to make sure that system runs normally or help it return to normal condition when a system error occurs, including faults diagnosis and actual maintenance.

Operators must follow the safety procedures and warnings as below:

WARNING	1. Do not change any of the information in the software security configuration. If the safety parameters have been changed, the entire robotic arm system should be considered as a new system, and all the safety audit process (e.g., risk assessment) must be updated. 2. Replace the defective components by new components with the same article number or equivalent components approved by UFACTORY (Shenzhen) Technology Co., Ltd. 3. Record all maintenance operations and save in technical documentation. The system should be considered as a new system, and all the safety audit process (e.g., risk assessment) must be updated.
DANGER	1. Remove the main input cable from the bottom of the controller to ensure that it is completely powered off. Take the necessary precautions to prevent other people from recharging the system during the repair period. After the power is turned off, the system must be re-examined to ensure that it is powered off. 2. Please check the ground connection before turning the system back on. 3. Observe ESD (Electro-Static discharge) regulations when parts of the manipulator or controller are disassembled. 4. Avoid disassembling the power supplies inside the controller. High voltages can be present inside these power supplies for several hours after the controller has been switched off. 5. Prevent water and dust entering the manipulator or controller.

4. Warranties

4.1. Product Warranty

xArm robotic arms have a finite warranty period for 12 months.

In the case of new devices and their components exhibiting defects resulting from manufacturing and/or material faults within 12 months of entry into service (maximum of 15 months includes the shipping time), UFACTORY (Shenzhen) Technology Co., Ltd. shall provide the necessary reserve components to replace or repair the related components.

UFACTORY (Shenzhen) Technology Co., Ltd. has the ownership of the devices or components which have been replaced or returned to UFACTORY (Shenzhen) Technology Co., Ltd.

If the products are no longer under warranty, UFACTORY (Shenzhen) Technology Co., Ltd. reserves the right to charge customers for replacing or repairing the products.

If there are any defects appear in the device outside the warranty period, UFACTORY (Shenzhen) Technology Co., Ltd. is not responsible for any damage or loss caused by the equipment, such as loss of production or damage to other production equipment.

4.2. Disclaimer

This Warranty will be invalid if the equipment defect is caused by improper handling or failure to follow the relevant information described in the user manual.

This warranty does not cover failures or damage caused by the following conditions:

1. Does not meet the requirements of industry standards or not following the user manual to install, connect wires and connect to other control devices.
2. Using products beyond the specifications or standards of the manual.

3. Using products beyond the appointed purposes.
4. Storage method and working environment are beyond the appointed range (e.g. pollution, salt injury and moisture condensation).
5. The product gets damages due to improper transportation.
6. Damage caused by accident or crash.
7. Not installing the original assembled components and accessories.
8. The damage caused by the third-party which is not UFACTORY (Shenzhen) Technology Co., Ltd. or the designated integrator while reconstructing, adjusting or repairing the original components.
9. Any natural disasters including fires, earthquakes, tsunamis, lightning, high winds and flooding.
10. Any malfunction does not caused by UFACTORY (Shenzhen) Technology Co., Ltd.other than the circumstances mentioned above.

The warranty does not apply if the user or the cause of the malfunction falls under any of the following: ：

1. Unable to identify the production date or the warranty start date.
2. Changing the software or internal data.
3. The malfunction cannot reappear or be identified by UFACTORY (Shenzhen) Technology Co., Ltd.
4. Using the products as radioactive equipment, biological test equipment or any other dangerous environment ascertained by UFACTORY (Shenzhen) Technology Co., Ltd.

According to the product warranty, UFACTORY (Shenzhen) Technology Co., Ltd. only provides a warranty to the flaws and defects in the products and components which are sold to the dealers.

UFACTORY (Shenzhen) Technology Co., Ltd. is not responsible for the relevant warranty responsibility to any other express or implied warranty or responsibility, including but not limited to the implied warranties to the merchantability or the specific use. In addition, UFACTORY (Shenzhen) Technology Co., Ltd. is not responsible for any indirect damage and consequences caused by the relevant products.

5. The hardware Composition of xArm

5.1. Hardware Composition

The composition of robotic arm hardware includes:

- Robot （Figure 5-1）
- controller (Figure 5-2)
- Robot signal cable (Figure 5-3)
- Robot power supply cable (Figure 5-4)
- controller power supply cable (Figure 5-5)

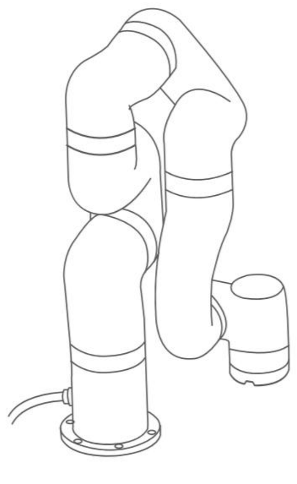


Figure 5-1

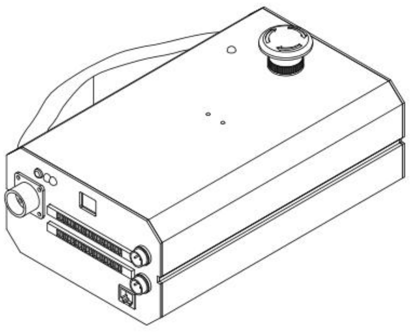


Figure 5-2

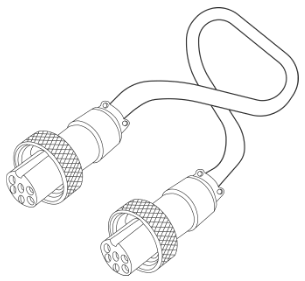


Figure 5-3

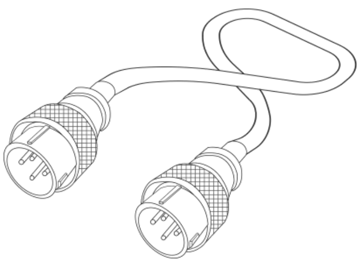


Figure 5-4

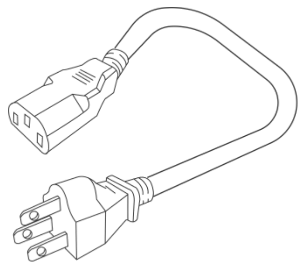


Figure 5-5

The xArm robotic arm system consists of a pedestal and rotary joints, and each joint represents a degree of freedom. From the button to the top, in order, Joint 1, Joint 2, Joint 3, etc. The last joint is known as the tool side and can be used to connect end-effector (e.g. gripper, suction cup, etc).

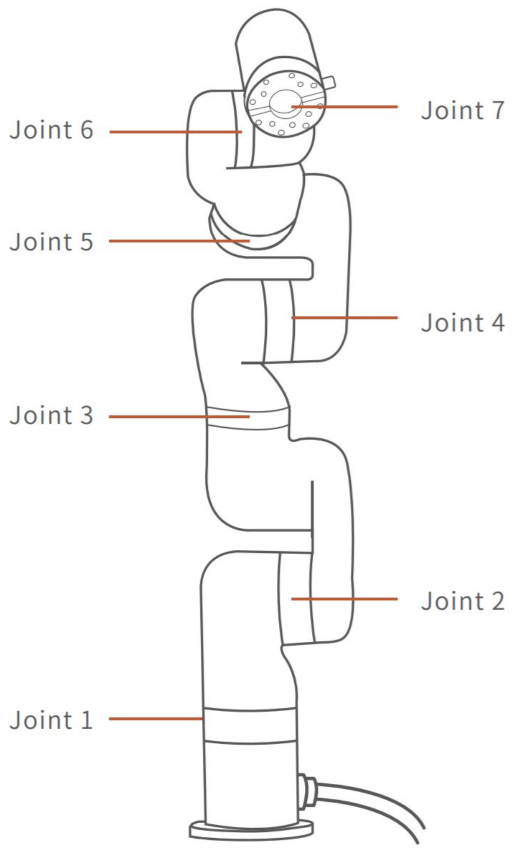
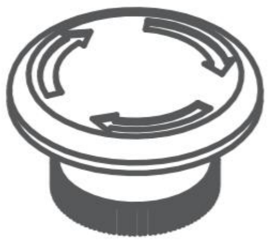
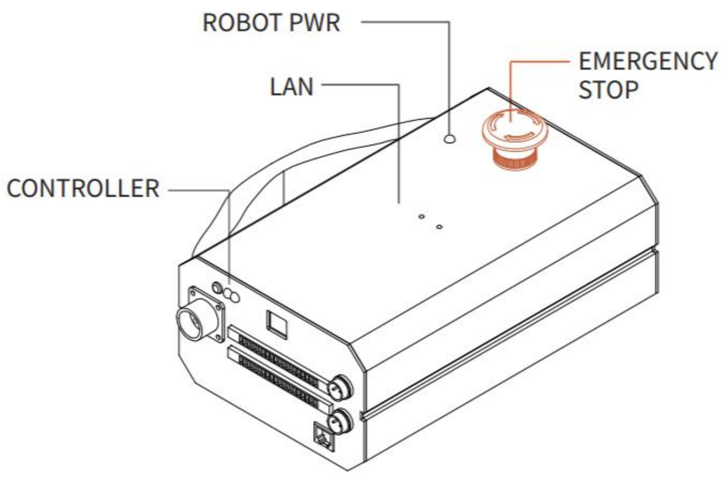


Figure 5-6 Robot Joints

5.2. Emergency Stop Button

By pressing the emergency stop button of the controller, a command will be sent to the controller for software deceleration to stop all activities of the robotic arm, plus clear all the cache commands in the controller; the power supply for the robotic arm will be disconnect within 300ms. Emergency stop should not be used as a risk reduction measure. When an emergency occurs during the operation of the robotic arm, users can press the emergency stop and the posture of the arm will remain at the moment when the button is pressed. The emergency stop button is shown below:





Emergency Stop：press the emergency stop button to power off the xArm, and the power light is extinct.

Power-on: when the button is rotated in the direction indicated by the arrow, the button is pulled up, the xArm power indicator lights up, and the arm is powered.

Note:

After pressing the emergency stop button, the following operations should be performed to re-start the robotic arm:

1. Power up the robotic arm
2. Enable the robotic arm (enable the servo motor)

5.3. Workspace of the Robot

The robotic arm workspace refers to the area within the extension of the joint, and the working range is a sphere with a radius of 700mm. Figure 5-8 shows the dimensions and working range of the robotic arm. When installing the robotic arm, make sure the range of motion of the robotic arm is taken into account, so as not to bump into the surrounding people and equipment (the end-effector is not included in the working range).

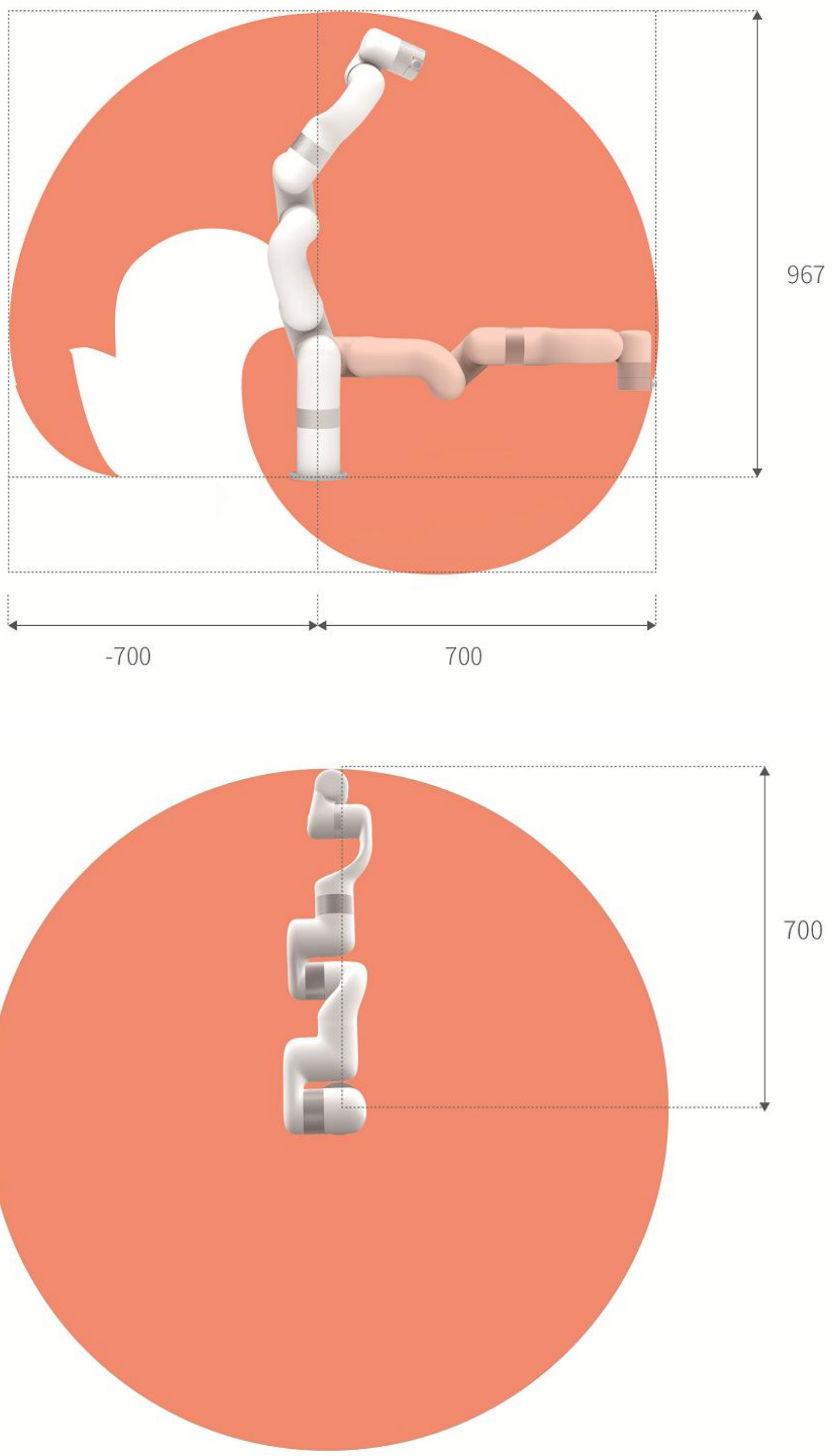


Figure 5-8 xArm workspace illustration (unit: mm)

6. Robot Installation

6.1. Safety Guidelines for the Robot Environment

DANGER	1. Make sure the arm is properly and safely installed in place. The mounting surface must be shockproof and sturdy. 2. To install the arm body, check that the bolts are tight. 3. The robotic arm should be installed on a sturdy surface that is sufficient to withstand at least 10 times the full torsion of the base joint and at least 5 times the weight of the arm.
WARNING	1. The robotic arm and its hardware composition must not be in direct contact with the liquid, and should not be placed in a humid environment for a long time. 2. A safety assessment is required each time it is installed. 3. When connecting or disconnecting the arm cable, make sure that the external AC is disconnected. To avoid any electric shock hazard, do not connect or disconnect the robotic arm cable when the robotic arm is connecting with external AC.

6.2. Robot Installation

1. Brief installation steps:
    a. Define a robotic arm workspace
    b. Fix the robotic arm base
    c. Connect the robotic arm with the controller
    d. Connect the controller with cable.
    e. Install end-effector

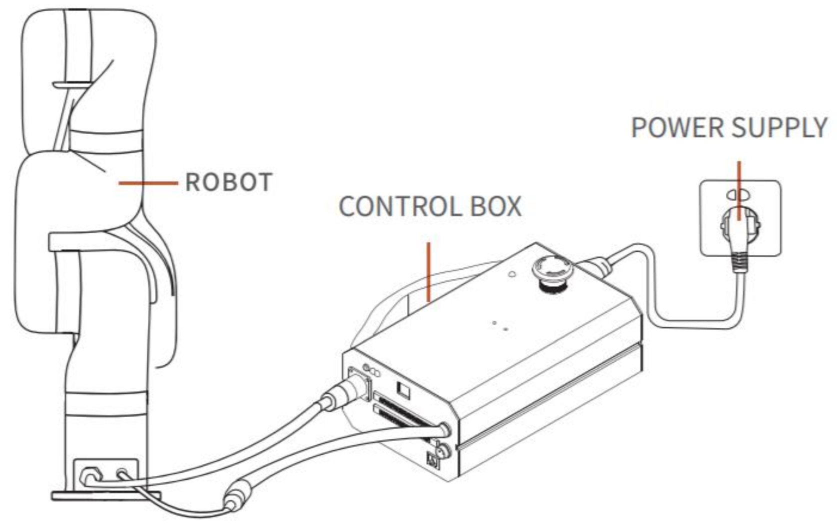


Figure 6-1

6.3. Robot Installation

The robotic arm has six M5 bolts and can be mounted through six ∅ 5.5 holes in the base of the robotic arm. It is recommended to tighten these bolts with a torque of 20N·m.

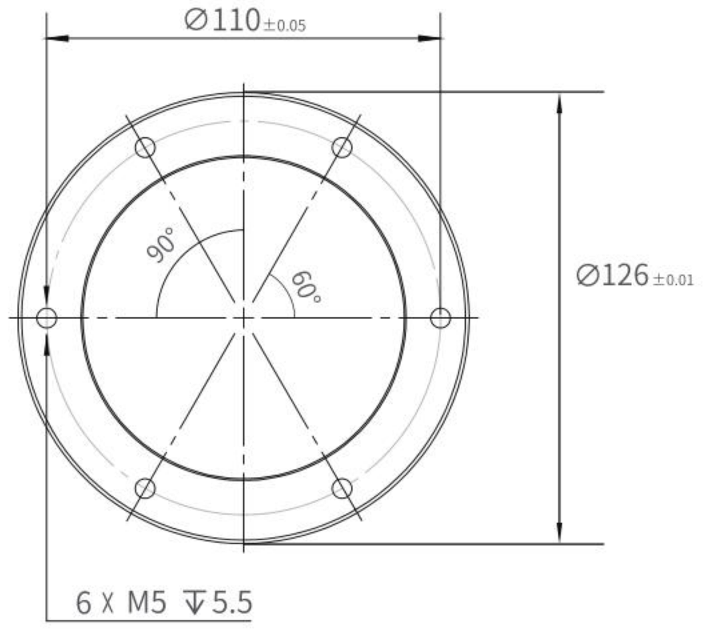


Figure 6-2 Robot Base Mounting (unit: mm)

6.3.1. Robotic Arm Mounting Configuration

The mounting configuration for the robotic arm is designed to inform the controller of the direction of gravity. The controller uses an advanced dynamics model to ensure that the robotic arm moves smoothly and accurately under a free-drive mode without external support. Thereby, it is important to correctly install the robotic arm.

The robotic arm has a 360° mounting position adaptive function, which can be installed on the base, ceiling mount, wall mount or other specific methods. The default method of the robotic arm is being installed on a table or ground.

Horizontal-Mount (0°): The default method is horizontal installation, and the horizontally mounted arm does not need a tilt angle and a rotation angle.

Ceiling-Mount (180°): For ceiling-mount, users simply need to set the mounting method for hoist, and it is not necessary to set the angle of rotation.

Wall-Mount (90°): For wall-mount, users need to set the rotation angle according to the actual situation.

Mount at other angles: For mounting at a certain angle. It is necessary to set the tilt angle and the rotation angle according to the actual situation.

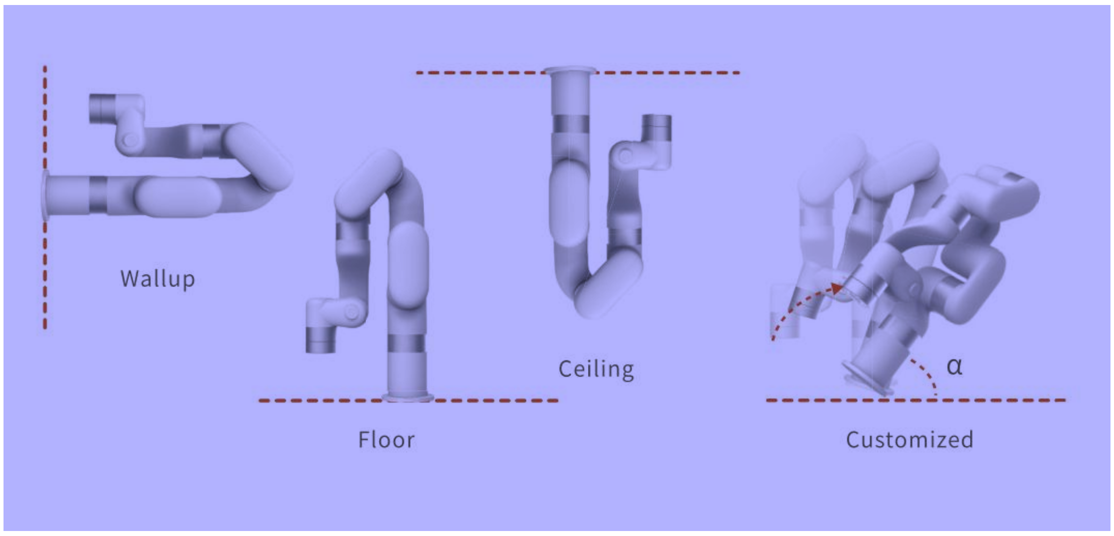
How to determine the tilt angle and rotation angle?

The initial position of the robotic arm: On the horizontal plane, when user is facing the robotic arm, the initial position is on the left-hand side of the user in a downward direction.

Tilt angle: The initial position of the robotic arm and the base of the robotic arm to be mounted should be in a tilt angle, which ranges from 0 to 180°.

Rotation angle: The initial position of the robotic arm and the end direction of the robotic arm to be mounted should be used as the rotation angle.

The method of determining the rotation angle ± direction: Hold it with your right hand and point your thumb in the direction of the robotic arm which is vertically mounted. The direction where your four fingers point is the positive direction and vice versa. The range of rotation angle is ±180°.



危险	1. Make sure the robotic arm is properly placed according to the actual use. 2. Must be mounted on a sturdy, shock-resistant surface to avoid the risk of rollover of the arm
警告	If the robotic arm is not properly installed, this will result in frequent safe stops; or the possibility that the robotic arm will move when in freedrive move.

2.1. End-effector Installation

End-effector flange has fourteen M6 threaded holes and one Φ5 positioning hole, where end-effector of two different sizes can be mounted. If the effector to be mounted does not have a positioning hole, The orientation of the end-effector must be documented in a file format, in order to avoid errors and unexpected result when reinstalling the end-effector. The end-effector flange supports the DIN ISO 9409-1-A50/A40 standard.

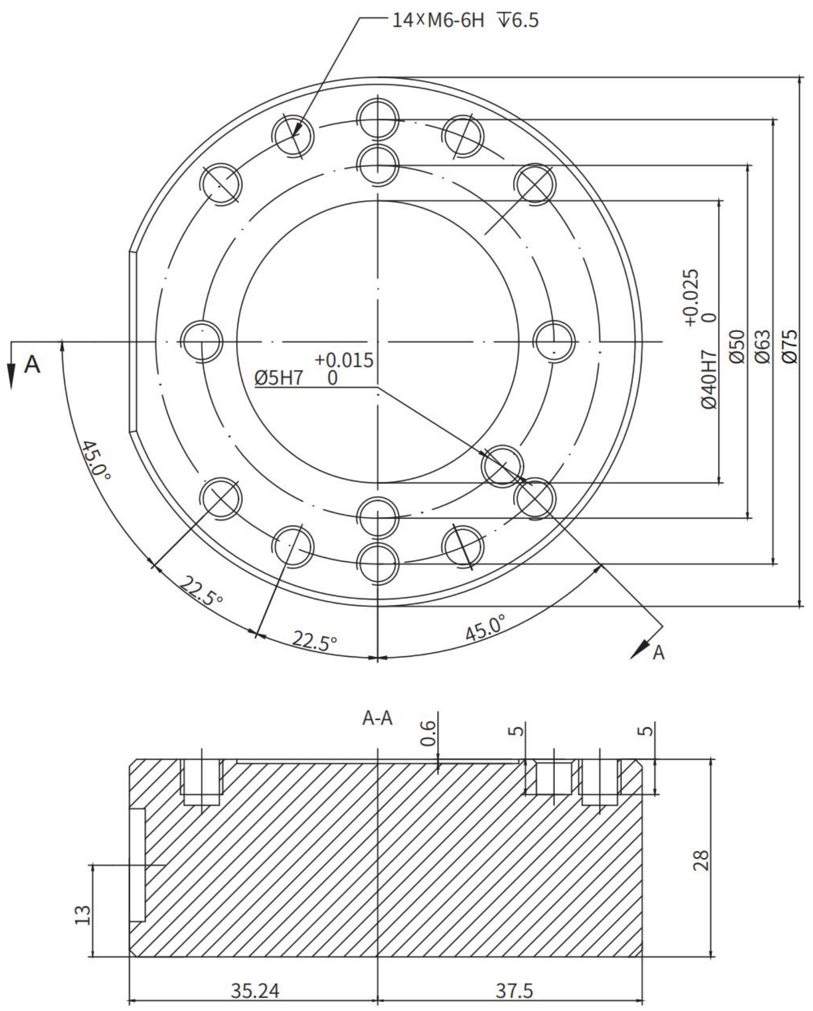


Figure 6-4 Mechanical dimensions of end-effector flange (unit: mm)

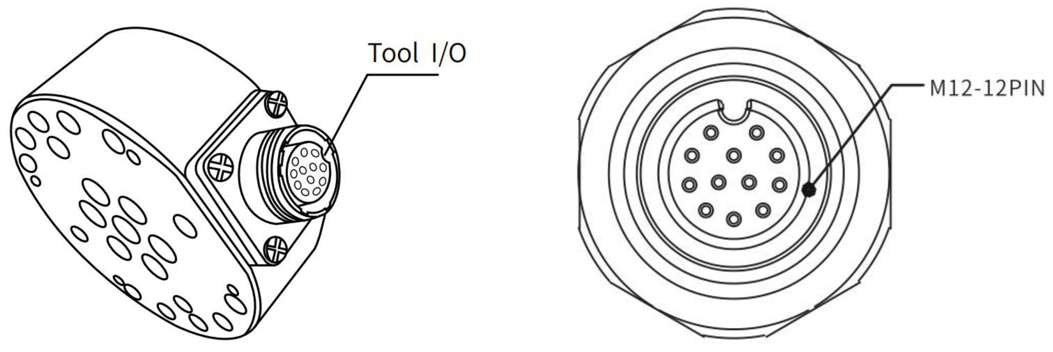


Figure 6-5 Dimension drawing of tool flange aviation joint (unit: mm)

DANGER

3.

1. Make sure the tool is properly and safely bolted in place.
2. If the end-effector does not have a locating hole, the orientation of the end-effector must be archived as a file.
3. Make sure that the tool is constructed such that it cannot create a hazardous situation by dropping a part unexpectedly.
4. Make sure to pay attention to the operation specifications of sharp end-effector tools.
5. If the installed end-effector exceeds the robotic arm mounting surface at the zero position of the robotic arm, a safety assessment is required for the zero return or the zero return operation.

6. Power supply for the Robotic Arm

6.1. Preparation before Connecting to the Power supply

- Make sure the power cable and the communication wire is properly connected between the controller and the robotic arm.
- Make sure the network cable or 485 cable is properly connected.
- Make sure the power cable for the controller is properly connected.
- Make sure the robotic arm will not hit any personnel or equipment within the working range.

6.2. Connect To The Power Supply

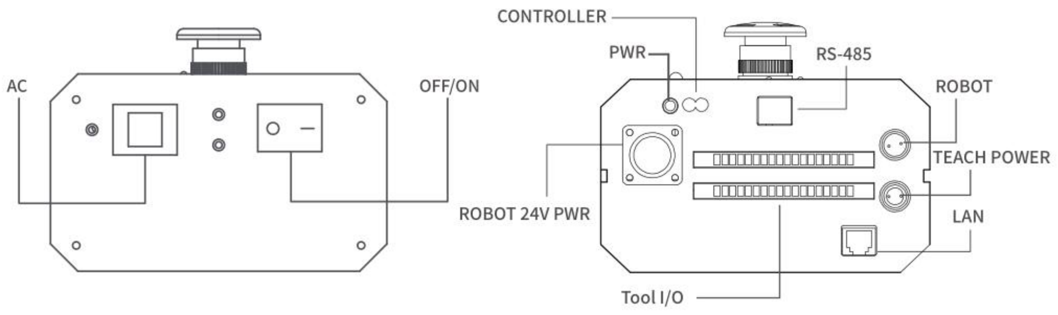


Figure 7-1

1. Turn on the OFF/ON button ,at which OFF/ON indicator lights up.
2. Press the power button, when the status indicator lights up, the controller is turned on.
3. The emergency stop button rotates in the direction indicated by the arrow and is pulled up, at which point the xArm power indicator lights up.
4. Enable the robotic arm (enable the servo motor).

6.3. Connecting to the Power supply

1.Turn on power button of the controller, the ON/OFF indicator lit

2.Check whether the status indicator is lit. If it is lit, it means the controller is turned on. Otherwise, you need to manually press the Power button to turn it on.

3.Rotates the emergency button in the direction indicated by the arrow and pull it up. The power indicator of the robotic arm lit.

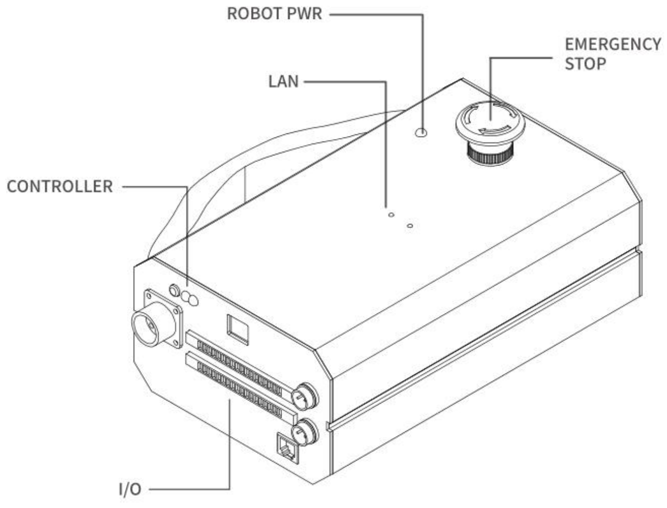
4. Enable the robotic arm (enable servo motor)

6.4. Shutting Down the Robotic Arm System

1. Shutdown the robotic arm in order

   (1) Cut off the power supply arm

   Press the emergency stop button to power off the robotic arm, then the power indicator light will be turned off.



2. Shutdown the controller

   (1) Press the power button of the controller for about 3s to turn off the status light.

   (2) Turn off the power supply of the controller (the power switch takes about 5 seconds to turn off the power of the controller. If users like to restart the power right after turning off the power supply, they need to manually press the Power button.)

WARNING

1.Unplugging the power cord directly from the wall outlet to show down the system may result in damage to the robotic arm file system, which may result in robotic arm malfunction.

7. Robotic Arm Controller

7.1. Connect the controller to the robotic arm

1. The robotic arm power supply cable connects the power port of the robotic arm and the ROBOT power port of the controller.
2. The robotic arm signal cable is connected to the signal interface of the robotic arm and the ROBOT signal interface of the controller.

7.2. Power Connection

There is a standard IEC plug at the end of the controller’s main cable. Connect a local dedicated main outlet or cable to the IEC plug. The controller is powered by 110V-240V AC, the input frequency is 50-60HZ, and its internal switching power supply converts 110V-240V AC into 12V, 24V DC, which supplies power to the load of the controller and the robotic arm.

Therefore, it is necessary to check whether the connection between the robotic arm and the controller is secured before use. The hardware protection and software protection of the controller can ensure the safety of use largely. The emergency stop button of the controller allows the user to cut off the power of the robotic arm in the shortest time and protect the safety of personnel and the equipment.

To power up the robotic arm, the controller must be connected to the power supply. In this process, the corresponding IEC C19 wire must be used.

Connect to the standard IEC C20 plug of the controller to complete the process, see figure 8-1.

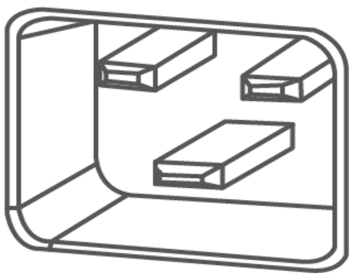


Figure 8-1

7.3. Electrical Alarms and Cautions

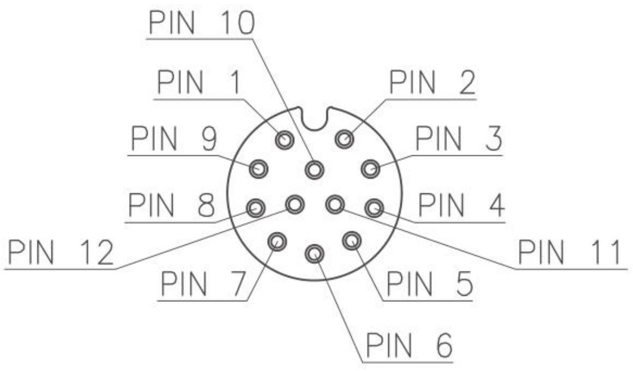
Always follow the warnings and cautions below when designing and installing a robotic arm application. These warnings and cautions are also subject to the implementation of maintenance work.

DANGER	Never connect a safety signal to a non-safety PLC. Failure to follow this warning may result in serious injury or death due to invalid safety stop function.
NOTICE	1. Make sure that all the non-waterproof equipments are kept dry. If water enters the product, turn off the power supply and contact your supplier. 2. Use only the original cable of the robotic arm. Do not use the robotic arm in applications where the cable needs to be bent. If you need a longer cable or flexible cable, please contact your supplier. 3. All GND connectors mentioned in this article are only suitable for powering and transmitting signals. For protective earth (PE), use the screw connector marked in the controller with the grounding mark. The grounding connector should have at least the rated current of the highest current in the system. 4. Be careful when installing the interface cable to the I/O of the robotic arm.
CAUTION	1. Interference with signals above the level specified in the IEC standard will cause abnormal behavior of the robotic arm. Extremely high signal levels or excessive exposure can cause permanent damage to the robotic arm arm. EMC problems usually occur during the soldering process and are usually prompted by an error message in the log. UFACTORY (Shenzhen) Technology Co., Ltd. is not responsible for any lose caused by EMC problems. 2. The length of the I/O cable that used to connect the controller and other mechanical and plant equipment must not exceed 30 meters unless it is feasible after the extension testing.

	When the electrical interface of the controller is wired, the controller must be powered off.

7.4. End-Effector I/O

At the tool side of the robotic arm arm, there is an avionic socket 12-pin female industrial connector. This connector provides power and control signals for the grippers and sensors used on a particular roboticarm tool. Please refer to the figure below:



There are 12 lines inside the cable with different colors, each color represents different functions, please refer to the following table:

Line sequence	Color	Signal
1	Brown	+24V（Power）
2	Blue	+24V（Power）
3	White	0V (GND)
4	Green	0V (GND)
5	Pink	User 485-A
6	Yellow	User 485-A
7	Black	Tool Output 0 （TO0）
8	Gray	Tool Output 1 （TO1）

9	Red	Tool Input 0 （TI0）
10	Purple	Tool Input 1 （TI1）
11	Orange	Analog input 0 (AI0)
12	light green	Analog input 1 (AI1)

The electrical specifications are as follows:

Parameter	Min. value	Typical Value	Max. Value	unit
Supply Voltage in 24V Mode	-	24	30	V
Supply Current *	-	-	2000	mA

Note： * It is strongly recommended to use a protection diode for inductive loads.

危险

Make sure that the connecting tool and the gripper do not cause any danger when the power is interrupted, for example, the workpiece is dropped from the tool.

7.4.1. Digital Output

The digital output is implemented in the form of NPN with an open collector. When the digital output is activated, the corresponding connector will be driven to GND. When the digital output is disabled, the corresponding connector will be open (open collector/open drain). The electrical specifications are as follows:

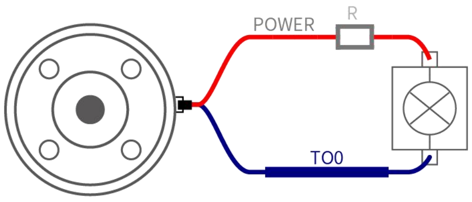
Parameter	Min	Typical	Max	unit
Open-circuit Voltage	-0.5	-	26	V
Voltage when sinking 300mA	-	0.05	0.20	V
Sink Current	0	-	50	mA
Current through GND	0	-	50	mA

小心

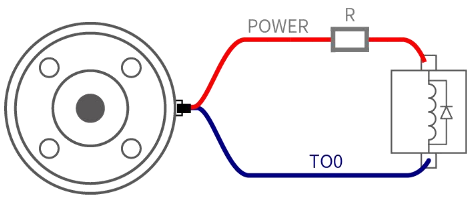
There is no current limit on the digital output of the tool, which can cause permanent damage if the specified data is exceeded.

7.4.1.1. Using Tools for Digital Output

The following example shows how to use the digital output. As the internal output is an open collector, the resistor should be connected to the power supply according to the load. The size and power of the resistor depends on the specific use.



Note: It is highly recommended to use a protection diode for inductive loads as shown below.



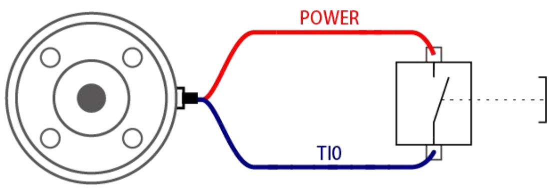
7.4.2. Digital Input

The digital input is already equipped with a pull-down resistor. This means that the reading of the floating input is always low. The electrical specifications are as follows:

Parameter	Min	Typical	Max	unit
Input Voltage	-0.5	-	30	V
Logic Low Voltage	-	-	1.0	V
Logic High Voltage	1.6	-	-	V
Input Resistance	-	47k	-	Ω

7.4.2.1. Using Tool for Digital Input

The following figure shows the connection with the simple buttons.



7.4.3. Tool Analog Input

The tool analog input is a non-differential input. The electrical specifications are as follows:

Parameter	Min	Typical	Max	unit
Input voltage in voltage mode	-0.5	-	3.3	V
Resolution	-	12	-	Bit
Input current in input current mode	-	-	-	mA
Pull-down resistors in the 4mA to 20mA current range	-	-	165	Ω
Resolution	-	12	-	Bit

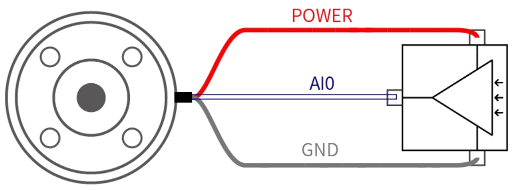
小心

1. In the current-voltage mode, the analog input does not provide overvoltage protection. Exceeding the limits in the electrical code may result in permanent damage to the input.

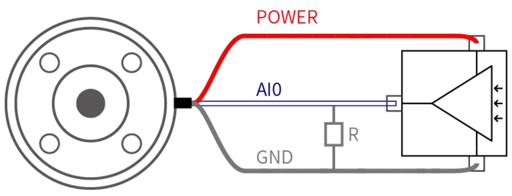
2. In current mode, the pull-down resistor size depends on the range of the input current.

7.4.3.1. Using Tool for Analog input, non-differential

The following figures show how the analog sensor can be connected to a non-differential output.



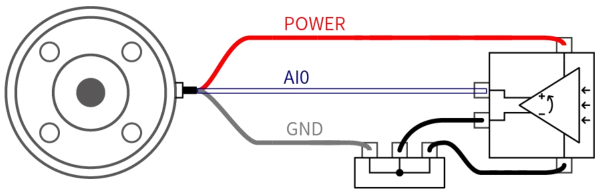
Voltage mode



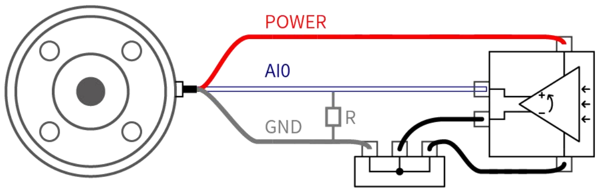
Circuit mode

7.4.3.2. Using Tool for Analog Input, Differential

The following figures show how the analog sensor is connected to the differential output. Connect the negative output to GND (0V), then it can work like a non-differential sensor.



Voltage Mode

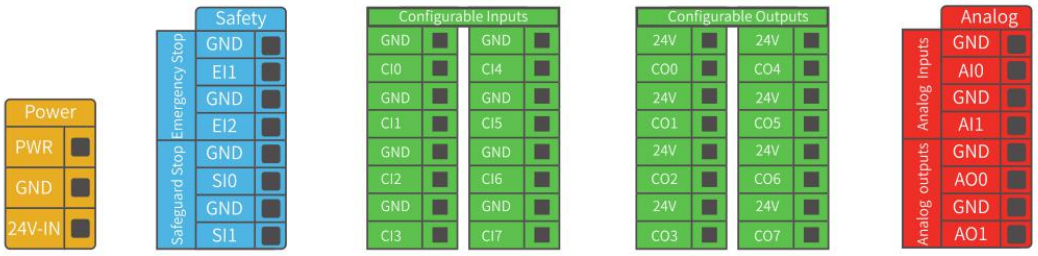


Current Mode

7.5. Controller Electrical IO

This chapter explains how to connect the device to the electrical I/O outside of the controller. This I/O is extremely flexible and can be used in many different devices, including pneumatic relays, PLCs and emergency stop buttons.

The figure below shows the electrical interface layout inside the control box.



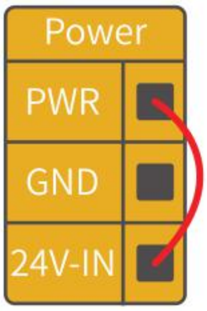
7.5.1. Universal specifications for all Digital I/O

This section describes the electrical specifications for the following 24V digital I/Os for the controller

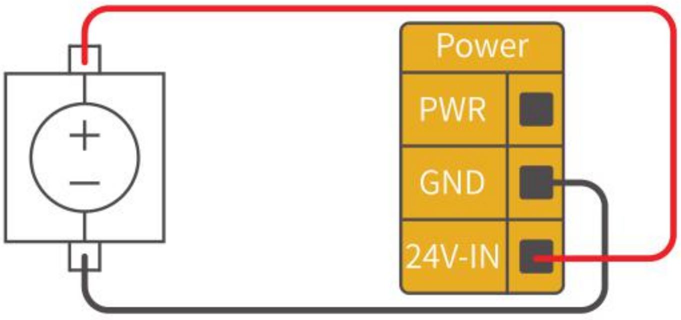
• Dedicated security I/O.

• Configurable common I/O.

It is very important to install xArm according to the electrical specifications, and both above I/O must comply with the specifications. The digital I/O can be powered by a 24V power supply or by an external power supply by configuring the power junction box. PWR is the internal 24V power output. The lower terminal (24V-IN) is the 24V input external power input for I/O. The default configuration is to use internal power, see below.



If larger current is needed, connect the external power supply as shown below.



The electrical specifications for the internal and external power supplies are as follows.

Terminal	Parameter	Min.value	Typical value	Max.value	Unit
Built-in 24V power supply					
[PWR - GND]	Voltage	23	24	26	V
[PWR - GND]	Current	0	-	2	A
External 24V input requirement					
[24V - 0V]	voltage	20	24	29	V
[24V - 0V]	Current	0	-	5	A

The digital I/O electrical specifications are as follows.

Terminal	Parameter	Min.value	Typical value	Max.value	Unit
Digital output					
[COx]	Current*	0	-	50	mA
[COx]	voltage goes down	0	-	0.5	V
[COx]	Open drain Current	0	-	0.1	mA
[COx]	Function	-	NPN （OC）	-	Type
Digital input					
[EIx/SIx/CIx/RIx]	voltage	0	-	30	V
[EIx/SIx/CIx/RIx]	OFF area	0.7	-	30	V
[EIx/SIx/CIx/RIx]	ON area	0	-	0.5	V
[EIx/SIx/CIx/RIx]	Current（0-	0.5	-	0.7	mA

	0.5)				
[EIx/SIx/CIx/RIx]	Function	-	-	-	Type
Note: ** For resistive or inductive loads up to 1H.					

Precaution:
1. There is no current limit on the digital output of the controller. If the specified data is exceeded, permanent damage may result.

7.5.2. Dedicated security I/O

This section describes the dedicated safety inputs and its configuration of the secure I/O. Please follow the universal specifications in Section 5.3.1.

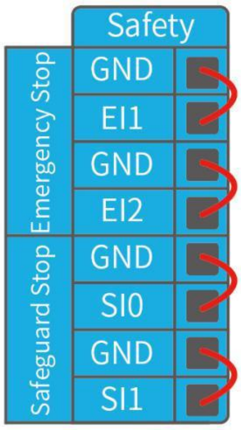
Safety devices and equipment must be installed in comply with the safety instructions and risk assessment (see Chapter 1).

All secure I/Os exist in pairs (redundancy) and must be kept in two separate branches. A single I/O failure should not result in the loss of safety features. There are two fixed safety inputs: Robotic arm emergency stop and protective stop. The robotic arm emergency stop input is only used for the emergency stop of the device. The protective stop input is used for all types of safety protection. The functional differences are as follows.

	Emergency Stop	Protective Stop
Stops the movement of the robotic arm	yes	yes
Program execution	Stop	Suspend
The power supply of the robotic arm	Off	On
Reset	Manual	Auto or manual
Usage frequency	Not frequent	No more than once per run cycle
Need reinitiation	Only releasing the brake	No

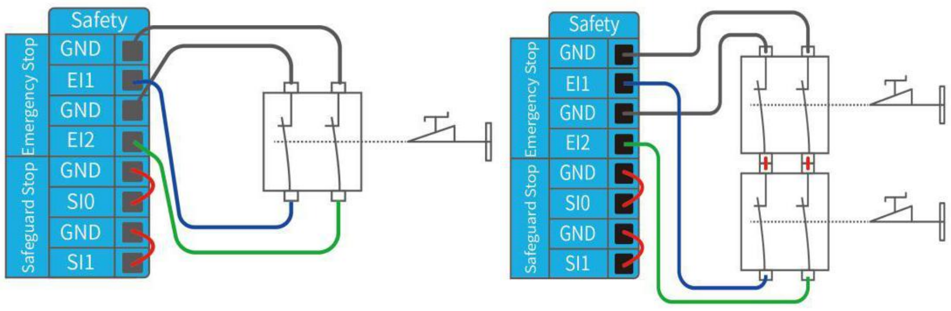
7.5.2.1. Default Security Configuration

The robotic arm has been configured by default and can be operated without any additional safety equipment, see the figure below.



7.5.2.2. Connect To the Emergency Stop button

In most applications, one or more additional emergency stop buttons are required. The figure below shows how to connect one or more emergency stop buttons



7.5.2.3. Share emergency stop with other machines

When a robotic arm is used with other machines, it requires to set up a common emergency stop circuit in most of the time. By setting up a public circuit, the operator does not have to think about which emergency stop button to use.

Since both machines need to wait for each other to jump out of an emergency stop condition, thereby, the standard robotic arm emergency stop input cannot be used for sharing.

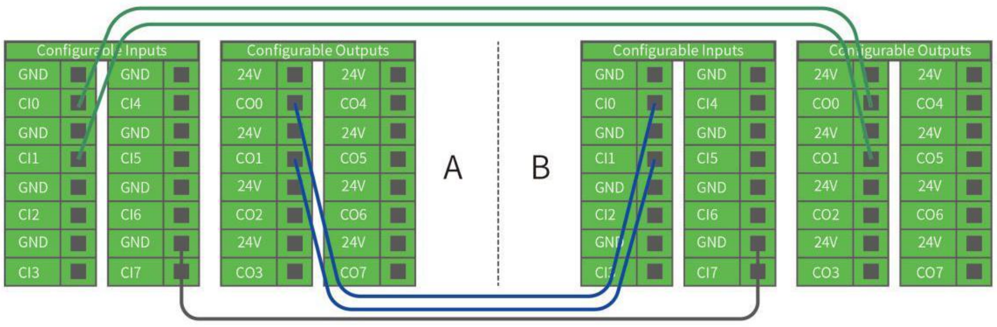
To share emergency stop functionality with other machines, you must configure configurable I/O capabilities through Studio.

• Configurable input pair: External emergency stop.

• Configurable output pair: System emergency stop.

The figure below shows how the xarm robotic arm shares the emergency stop function. In this example, the configuration I/O used is

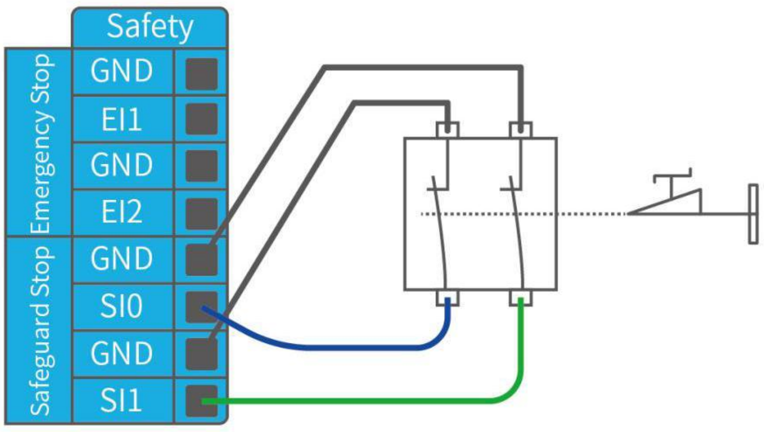
“CI0-CI1” and “CO0-CO1”.



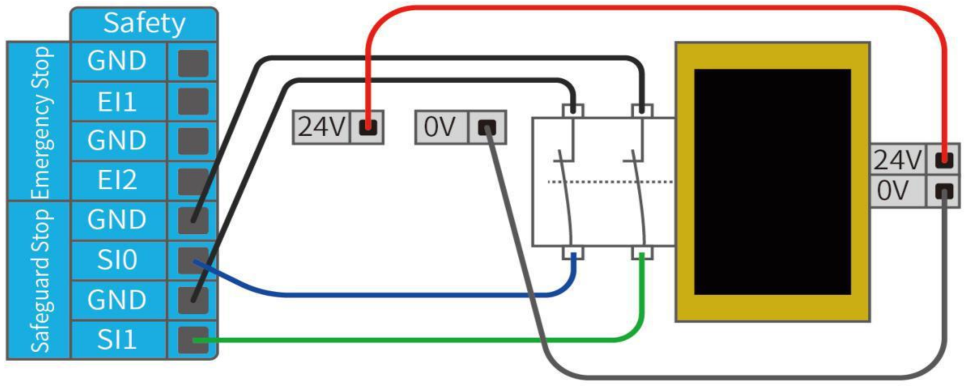
In case you need to connect more than two xArm or other machines, it is required to use a safety PLC to control the emergency stop signal.

8.5.2.4 Automatically recoverable protective stops

The door switch is an example of a basic protective stop device. When the door is open, the robotic arm stops. See the figure below.

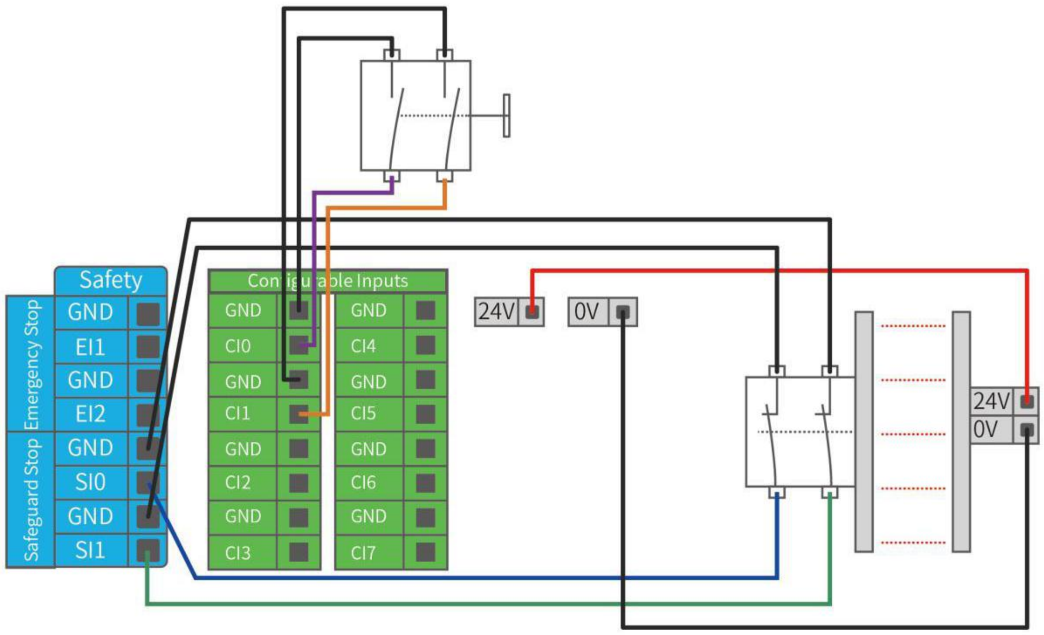


This configuration is only for applications where the operator is unable to close the door behind it. Configurable I/O can be used to set the reset button outside the door, in order to reactivate the movement of the robotic arm. Another example of an automatic recovery is the use of a safety pad or a safety laser scanner, see the figure below.



8.5.2.5 Protective stop with reset button

If you use a protective interface to interact with the light curtain, you need to reset from outside the safety zone. The reset button must be a two-channel button. In this example, the I/O of the reset configuration is “CI0-CI1”, see the figure below.



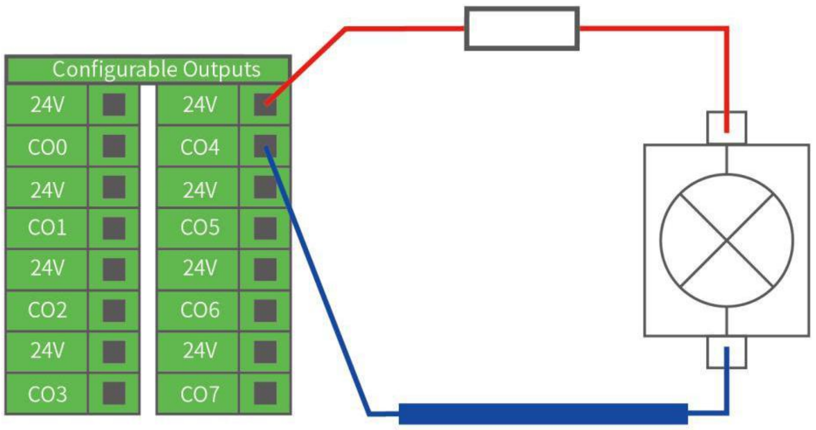
7.5.3. Common Digital I/O Function

7.5.3.1. Configurable digital Output

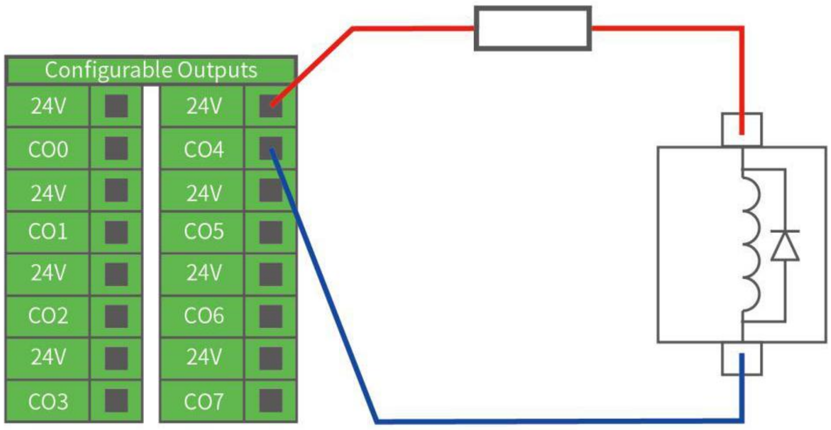
The digital output is implemented in the form of NPN. When the digital output is enabled, the corresponding connector will be driven to GND. When the digital output is disabled, the corresponding connector will be open (open/open).

Users must follow the electrical specifications set in section 8.5.1 'universal specification'.

The following example shows how to use the digital output, as the internal output is an open-drain output, so you need to connect the resistor to the power supply according to the load. The size and power of the resistor depends on the specific use.



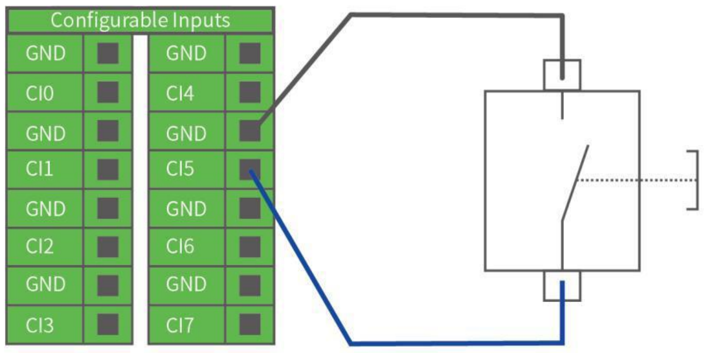
Note: It is highly recommended to use a protection diode for inductive loads as shown below.



7.5.3.2. Configurable digital input

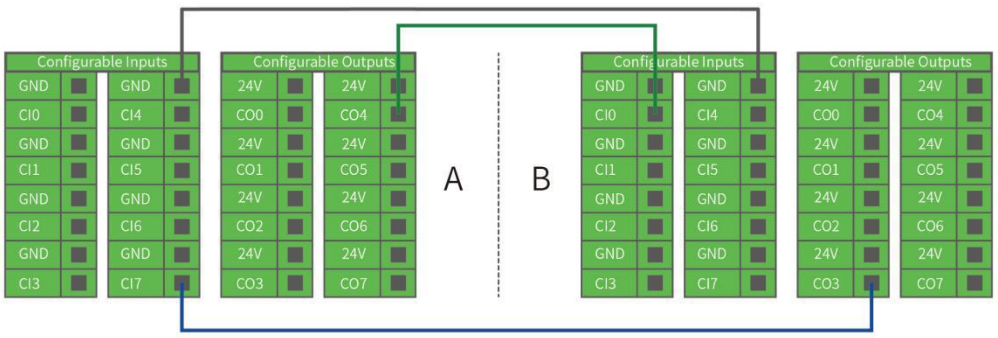
The digital input is implemented in the form of a weak pull-up resistor. This means that the reading of the floating input is always high.

User must follow the electrical specifications set in the 8.5.1 ‘universal specification’. This example shows how a simple button is connected to a digital input.



8.5.3.3 Communicate with other machines or PLCs

If general GND (0V) is established and the machine uses open-drain output technology, digital I/O and other can be used device communication, see the figure below.



7.5.4. Universal analog I/O

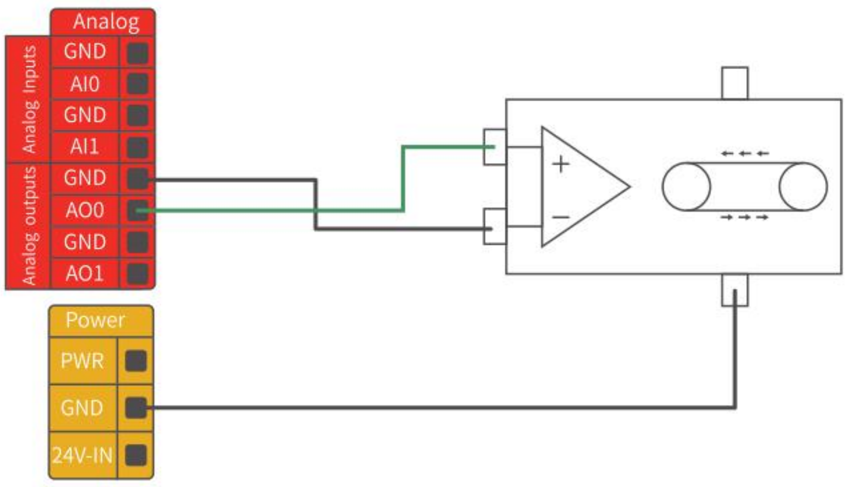
This type of interface can be used to set or measure voltage (0-10V) into and out of other devices.
For the highest accuracy, the following instructions are recommended:
• Use the GND terminal closest to this I/O.
• The device and controller use the same ground (GND). The analog I/O is not isolated from the controller.
• Use shielded cables or twisted pairs. Connect the shield to the “GND” terminal on the “Power” terminal.

Terminal	Parameter	Min. value	Typical value	Max.value	Unit
Analog input under voltage mode					
[AIx - AG]	voltage	0	-	10	V
[AIx - AG]	Resistance	-	10	-	Kohm
[AIx - AG]	Resolution	-	12	12	bit
Analog output under voltage mode					
[AOx - AG]	voltage	0	-	10	V
[AOx - AG]	Current	0	-	20	mA
[AOx - AG]	Resistance	-	100	-	Kohm
[AOx - AG]	Resolution	-	12	-	bit

The following example shows how analog I/O is used.

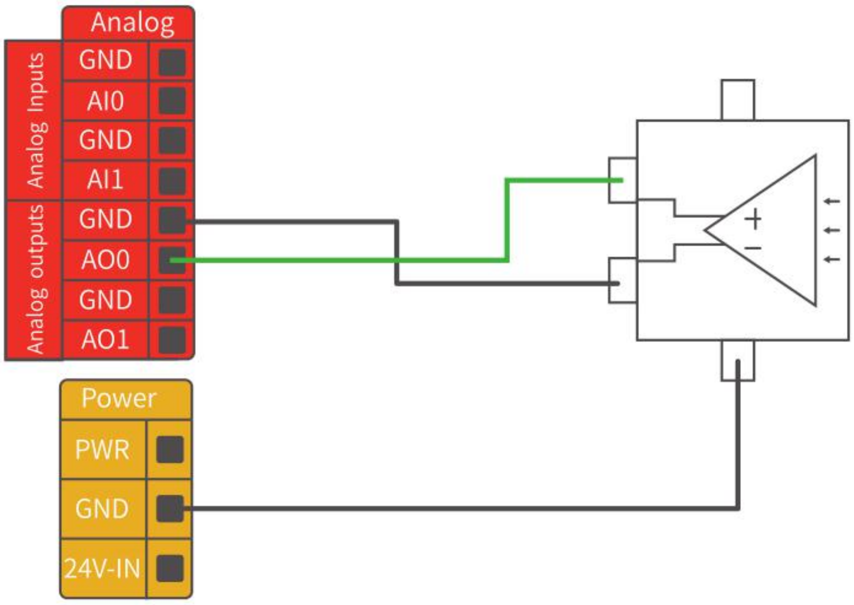
7.5.4.1. Using analog output

The following example shows how to use the analog speed control input to control the conveyor belt.



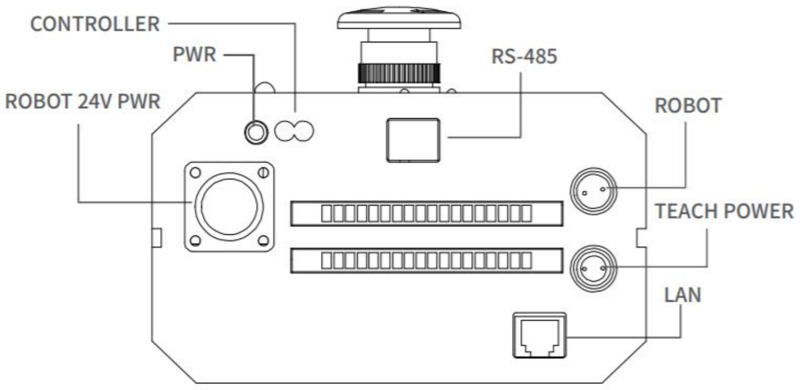
7.5.4.2. Using analog Input

The following example shows how to connect an analog sensor.



7.6. Communication Interface

The controller provides RS-485 interface and Ethernet interface, as shown in the figure below.



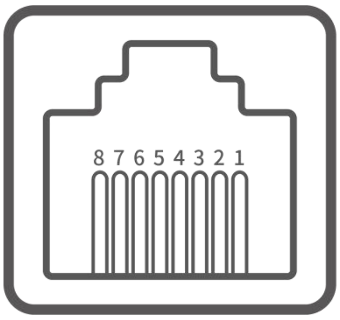
7.6.1. RS-485 Communication

The controller provides an RS-485 interface. (485 communication circuit is not factory standard and requires additional purchase)

The controller and the computer are connected through the 485 interface, one end of the 485 communication circuit is connected with the computer, and the other end is connected with the RS-485 interface of the control box.

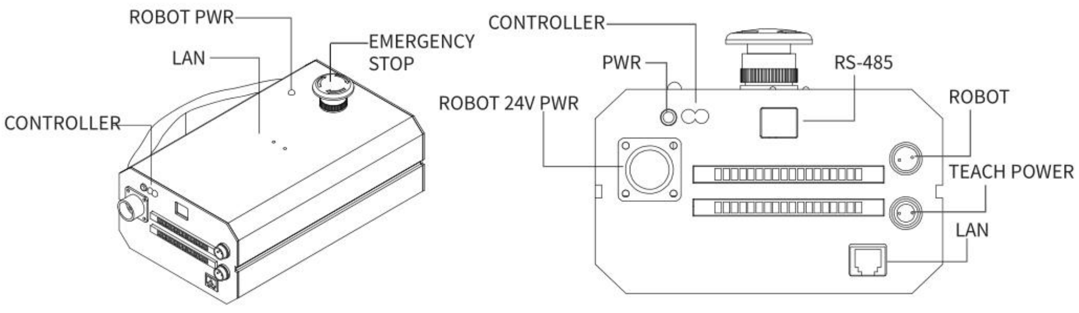
7.6.1.1. RS-485 Circuit Order

Serial number	Description
1	485_A
2	485_B
3	GND
4	Not connect
5	Not connect
6	Not connect
7	Not connect
8	Not connect



7.6.2. Ethernet TCP/IP

The controller provides a gigabit Ethernet interface.



Ethernet connection steps：The controller and the computer are connected via

Ethernet. One end of the network cable is connected to the network interface of the controller, and the other end is connected to the computer or LAN network interface. If the connection is successful, the network port indicator blinks frequently.

The default network segment IP address of the controller is 192.168.1.*(2~254). For specific IP address, please check the controller label. When communicating with the robotic arm, the network IP address of the computer should in the same network segment with the IP address of the controller.

Note: To connect with Ethernet, please check if the computer's IP address is 192.168.1.*, check if the network proxy is enabled, and check if the robotic arm's IP address conflicts with that of other devices in the LAN. If it is not such field, please change the computer IP address and close the computer's network proxy. To test whether the computer can communicate with the robotic arm, open the command terminal and input ping 192.168.1.* (the IP address of the robotic arm). If the ping is working, the communication between the computer and the robotic arm is successful.

7.7. Controller Command Caching Mechanism

The current controller can cache 512 commands. If more than 512 commands need to be sent, the controller will execute the current commands before sending further commands. When awaiting commands of the controller exceed the maximum buffer amount (512), and continues to send the commands, a return warning will be shown. The warning code is decimal 11, and the command will be discarded. The commands issued should not exceed 256 command caches.

7.8. Controller Configuration File

The robotic arm features four underlying motion modes: joint motion, linear motion, linear arc motion, and servoj motion.

The controller configuration file saves jerk, maximum acceleration and tcp offset. To modify or delete the configuration, saved in the file after modification, and the configuration file restarts the robotic arm and the controller. The default configuration is used after the modified configuration is removed.

8. The motion of the robotic arm

The robotic arm arm has four underlying motion modes: joint motion, linear motion, linear circular motion, and servoj motion. The robotic arm moves from point A to point B and then to point C, the four basic motion modes have different joint rotation processes.

The following motion modes are in position mode:

- Joint Motion: to achieve the point-to-point motion of joint space (unit: degree/radian), the speed between each command is discontinuous.
- Linear Motion: to achieve linear motion between Cartesian coordinates (unit: mm), the speed between each instruction is discontinuous.
- Linear Circular Motion: to achieve linear motion between Cartesian coordinates (unit : mm), inserting an arc between two straight lines, and the speed between each command is continuous.

The following motion modes are in servoj mode:

- Servoj motion: move to the given joint position with the fastest speed (180°/s) and acceleration (unit: degree/radian). This command has no buffer, only the latest received target point, and user needs to enter the servoj mode to use the controller maximum receiving frequency of 200Hz. The xArm-Python-SDK interface function we provide also reserves the speed, acceleration and time settings, but it does not work at present. The suggested way of use: If you want to plan your own track, you can use this command to issue a smoothed track point with interpolation at a certain frequency (preferably 100Hz or 200 Hz), similar to the position servo control command. (Note: this step is similar to the highest step response, for safety reasons, do not give a distant target position at once)。 Using this mode requires detailed position planning for each axis and motion estimation it difficult to develop.

CAUTION

Operators who design the robotic arm motion path must be qualified with the following conditions:

1. The operator should have a strong sense of security consciousness, and sufficient knowledge on robotic arm operations.
2. The operator should have in-depth knowledge on robotic arm, and understands the joint motion mode and linear motion mode.
3. The operator should have safety knowledge on emergencies.
4. When planning the path for the robotic arm, risk assessment must be done and the operator should be cautious.

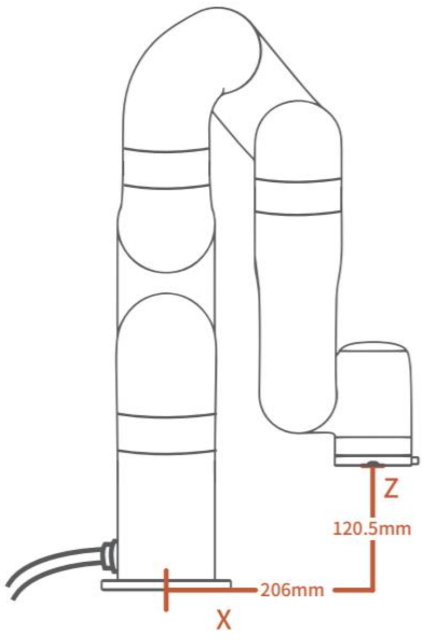
8.1. Zero Pose of the Robot

The zero return command of the robotic arm is the movement of the joint, which will turn off the collision detection of the robotic arm itself.

The zero point [0,0,0,0,0,0] is sent with the joint command, the robotic arm has its own collision detection.

Figure 6.1 shows the zero pose of the robotic arm, and the rotation angle of each joint is 0.

When the x/y/z/Roll/Pitch/Yaw value of the TCP offset setting is 0, the robotic arm zero point Cartesian value would be: x/y/z/Roll/Pitch/Yaw = [206mm, 0, 120.5mm, 180°, 0, 0]



DANGER

If the end-effector is installed in the robotic arm, make sure to assess whether the robotic arm will hit the obstacles or the fixed surface of the robotic arm when it returns to the zero pose.

CAUTION

The robotic arm should be back to the zero pose before packaging.

8.2. Robotic Arm Collision Detection

The xArm7 is equipped with collision detection. When the power on the arm exceeds the default normal force range, it rebounds a certain distance and automatically stops to prevent itself or operator from colliding. The collision sensitivity range is 0~5 levels. When set to 0, it means that collision monitoring is not turned on. The larger the set value, the higher the collision sensitivity level, and the smaller the force required after the arm collision monitoring. In scenes that require faster speeds, the collision sensitivity is as small as possible.

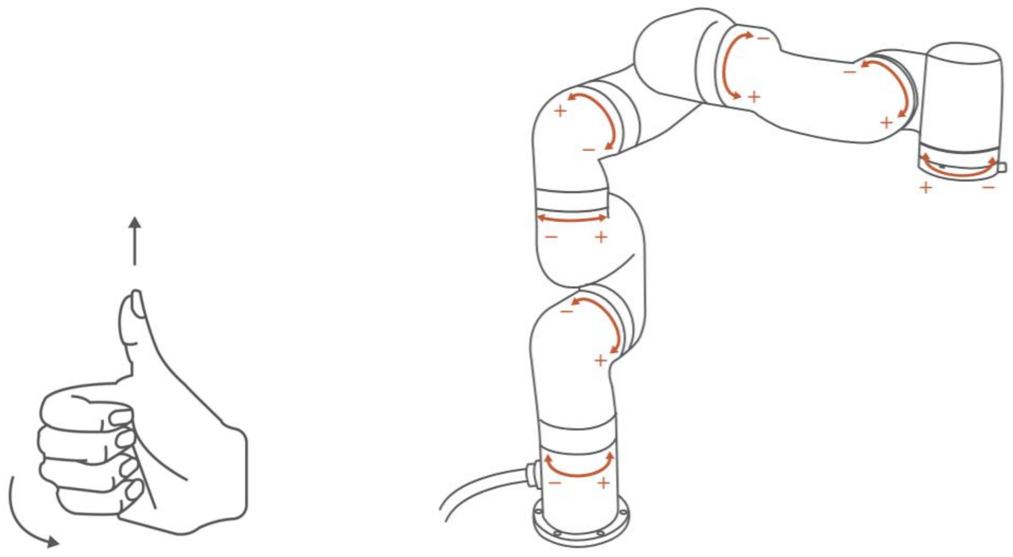
8.3. Joint Motion

The robotic arm consists of joint modules. The position of the end-effector is controlled by coordinating the rotation angle of each joint.

The joint motion reach the target point with the fastest path, the end trajectory is not a straight line, and the speed unit is °/s. After the target point is set, the corresponding poses are unique both in the end path and the trajectory process.

8.3.1. Joint Rotating Direction

To confirm the direction of joint rotation, the following method can be followed: hold the right hand, the thumb points to the next joint direction, the other four fingers point to the positive direction value and vice versa. When setting the joint value, ± indicates the joint rotating direction.



8.3.2. Joint Range

There are two joint rotating unit s: radians (rad) and degree (°). When using the SDK, the unit to be used should be clearly defined. The figure below shown the maximum value of each joint in the rotating direction (±). In fact, the joint rotation will approach the maximum value infinitely. When the stated value is equal to the maximum value, the error reporting mechanism of the robotic arm may be invoked.

Joint	Joint （unit: rad）	Joint （unit: degree）
1	±2PI	±360
2	-2.18 ~ 2.18	-125~ 125
3	±2PI	±360
4	-4.0100 ~ 0.1000	-230 ~ 6
5	±2PI	±360
6	-1.7500 ~ PI	-100 ~ 180

8.3.3. Joint Operating Speed

Joint Speed: refers to the rotating degree per second, the range is 1°/s ~ 180°/s. When the robotic arm is in operation, the maximum speed will be influenced by the payload, speed and the pose, and the maximum speed would not be an absolute value.

Acceleration: the start-stop time can be adjusted, so as the jitter during the start-stop. The range is recommended to be within 20 times the maximum operating speed [20*180°/s].

Note: the speed at which the joint runs between each command is discrete, and the robotic arm will have a brief pause when running the joint command.

NOTICE

1. If the operating speed is too fast (>=150°/s), the robotic arm arm can trigger the error reporting mechanism easily.

2. For the linkage of multiple joints, the speed is preferably set within 100 ° / s.

3. The robotic arm arm cannot reach the stated constant speed in different postures.

4. When the robotic arm is running in a high speed, make sure not to bump into the surrounding people and equipment

5. When changing the speed, it takes a certain amount of time to reach the stated speed, and the acceleration should be set.

8.3.4. Joint Brakes

Each joint of the robotic arm is equipped with a brake. The joint brake can be released when the robotic arm is motion enabled, and the joint that releases the brake can be freely dragged.When the robotic arm is motion enabled, all the joint brakes should be manually turned off to ensure a normal operation on the robotic arm.

危险	When releasing the joint brakes, someone must support the robot's posture to prevent the robotic arm from falling without external force and damage the robotic arm and surrounding equipment.
注意	1. After the release of the joint brake and manually dragging the arm, please always pay attention to the degree of joint rotation to avoid exceeding the rotation range of the robot joint and damage the internal structure of the robotic arm. 2. After all the joints are closed, the robot arm can be driven. 3. Brakes are installed in the joint module, and the robot arm posture is maintained when the power is off.

8.3.5. Free Drive of the Joints

When the hand-guiding teaching mode is turned on, the robot joint can be manually dragged to reach the target position point, which makes the recording movement of the robot more convenient, thereby reducing the development workload. When a hazard occurs, users can also use the teaching mode to manually drag the robot to quickly leave the danger zone.

Using the correct mounting settings ensures that the robotic arm is self-supporting when the teaching mode is turned on with end-loading and non-planar mounting, avoiding changes in the robot arm posture caused by the endless gravity caused by no external force.

警告	1.Make sure to use the correct installation settings (for example, the robot's mounting angle, TCP's center of gravity, TCP offset). 2. Before turning on the free drive, please make sure the TCP settings and robot installation settings are correct. If these settings are not correct, the robot arm may move while the free drive is active.

8.3.6. Recording

The position of the joint is obtained and recorded by 100HZ to record the motion trajectory of the robotic arm in free driving, and the maximum recording time is 5 minutes. The playback speed and acceleration of the trajectory cannot be set for the time being.

8.4. Cartesian Space Motion

8.4.1. Introduction

Users can control the motion status of the robotic arm based on the base coordinate system and TCP coordinate system. The trajectory of the robotic arm terminal of the Cartesian space motion is a straight line. The tool moves linearly between waypoints when it's under the Cartesian motion. This means that each joint performs a more complex movement to keep the tool in a straight path. The end path is unique once the target point is confirmed, and the corresponding posture in the trajectory process is random.

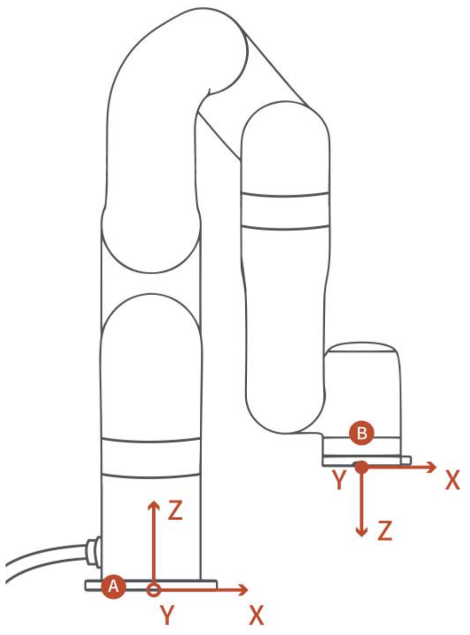
The position of the X, Y, and Z control tools in the base coordinate system, the arrow points to the + direction, and the unit is mm. The Roll/Pitch/Yaw of the TCP coordinate system control the direction of the tools, and the unit is degree.

Linear motion and circular linear motion belong to the Cartesian space trajectory planning, which needs to be solved by inverse kinematics. Therefore, there may be no solution, multiple solutions, and approximation solutions; and due to the nonlinear relationship between the joint space and Cartesian space, the axial motion may exceed its maximum speed and acceleration limits.

8.4.2. TCP Setting

TCP offset: When the x/y/z/Roll/Pitch/Yaw value of the TCP offset setting is 0, TCP coincides with the centre point of the tool output flange.

TCP load and center of gravity: Please set the payload and tool center of gravity offset according to the actual situation of the arm. The maximum payload does not exceed 5KG. As the tool's center of gravity shifts, the actual payload will decrease slightly. If there is no actual load at the end of the arm, the payload must be set to 0.

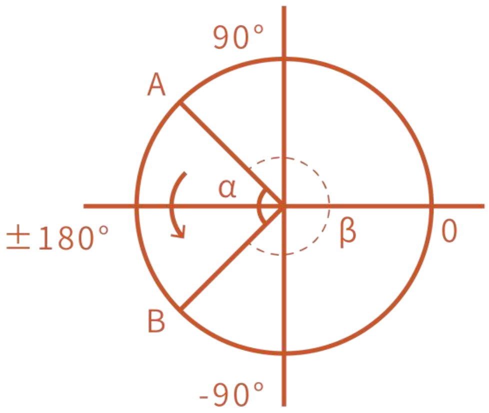


A: base coordinates B:TCP coordinates

8.4.3. TCP Coordinate System

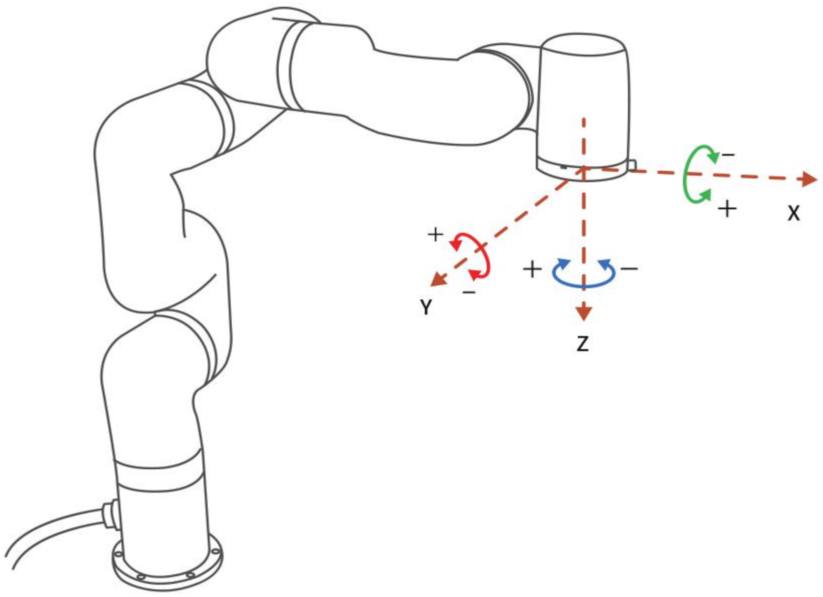
The TCP coordinate system is defined by the centre point (TCP) of the end of the robotic arm, and it is the result of rotating [180°, 0°, 0°] around the x/y/z axis of the base coordinate system in order. The x/y/z spatial orientation of the TCP coordinate system changes according to the changes of the angle of rotation.

Roll /Pitch/Yaw rotates around the X/Y/Z of the TCP coordinate system. The value of ± is the value of the circle in the range of the rotation angle. The direction of rotation will be rotated according to the smaller angle between the two points. In particular, it is important to strictly control the magnitude of the deflection angle between the two points to control the direction of rotation, and if necessary, insert a third point between the two points. As shown in figure 6.4, if a deflection is needed from position point A to point B, the robotic arm moves in the direction of α angle. If the robotic arm needs to be moved in the direction of β angle, a new position between the angles of β should be inserted, and the angle that formed by the inserted point and A should be smaller than α.



- The ±180° points of the Roll/Pitch/Yaw are coinciding in the space, and the range is ±180°, so it is possible to have ±180° when the robotic arm is reporting the position.
- RPY [rad] Roll angle, pitch angle, and yaw angle (RPY). The RPY rotation matrix (X, Y', Z'' rotation) is determined by the following formula:

$R\,rpy(\gamma, \beta, \alpha) = R\,Z(\alpha) \cdot R\,Y(\beta) \cdot R\,X(\gamma)$



DANGER	1. In the case where the length of the end tool is more than the end joint of the robotic arm, the direction of rotation must be clear when setting the degree of rotation. 2. You must check the TCP offset before recording the Cartesian position.
CAUTION	1. The Cartesian motion is a straight line, and the two Cartesian instructions cannot cross the robotic arm base coordinate system itself. 2. The value range of Roll/Pitch/Yaw is ±180°, which is the idealized maximum value. The actual value depends on the specific pose of the robotic arm.

8.4.4. TCP Position Range

To approach the limit value, the joint attitude needs to be adjusted. In the case where the other two axes are 0, the TCP position limit value shown in the table below are theoretically supported.

Axis	x	y	z	roll	pitch	yaw
Range	±700mm	±700mm	400mm~951.5mm	±180°		

8.4.4.1. Safety Boundary and Speed Reduction

Safety Boundary: When this mode is activated, the boundary range of the robotic arm in Cartesian space can be limited. If the tool center point of the robotic arm exceeds the safety boundary, the robotic arm will stop moving. Defining a safety boundary only limits the TCP, has no effect on the range of motion of the joint.

Speed Reduction: When this mode is activated, the maximum motion speed and maximum joint speed of the robotic arm in Cartesian space can be limited.

8.4.5. TCP Operating Speed

The Cartesian speed range is from 1mm/s to 1000mm/s. The maximum speed is also affected by the payload, speed and posture of the robotic arm. If the speed is set very close to the limit speed, then the robotic arm will slow down or trigger the error reporting mechanism.

According to the scene, the start-stop time and jitter at the start-stop time can be adjusted. It is recommended to be within 20 times the [20mm/s^2]. The speed and acceleration limits are different in different poses, theoretically supporting 1000mm/s.

When a command has displacement and rotation, the time required for the displacement motion and the rotational motion is calculated. The speed depends on the one that takes more time, but in principle, it is better to separate the displacement from the rotation command.

8.4.6. Linear Motion

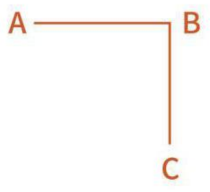
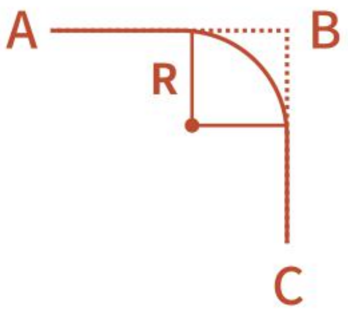


Figure 6.6

The linear motion of the robotic arm is equipped with the following features:

- The end motion trajectory is a straight line.
- In terms of speed, the speed between linear motion commands is 0, and the speed between each command is discontinuous.
- In terms of track accuracy, the linear motion command will strictly reach points B and C from the current point A.

8.4.7. Linear Motion with Circular Arc



- In terms of speed, the speed between arc motion commands is continuous.

- In terms of track accuracy, it will reach point C from the current point A, but will not reach point B in the middle, and will bypass the circular path around point B.
- The arc blending radius is reflected in each command with an arc line, and an arc is made at the middle point. The radius of this circle is the blending radius.
- The circular arc motion is adapted to: a continuous speed is needed between the commands, and the trajectory accuracy of each point is not required.
- When moving in a circular arc motion, the robotic arm needs a cache command to programme the motion.

8.5. Path Planning Guidelines

DANGER	● If an emergency stop occurs, and a zero return operation or other path planning is needed, be sure to perform a safety assessment to prevent collisions and, if necessary, unlock the joint to a safe point and then reset. ● If the robotic arm collides with an obstacle during the movement, the robotic arm will be stopped by external force. At this time, an error will be reported and a mechanism for clearing the error is required.
CAUTION	● Repeat the same set of Cartesian commands. When no joint command is used, the posture of the arm will change slightly. If it continues to accumulate, it may cause an error. In motion planning, a joint command correction is needed. ● When going to the same position, the rotation process of the joint motion, Cartesian motion, and the robotic arm are different. ● In a certain posture of the robotic arm arm, it may not be able to take the Cartesian linear motion. In this case, the rotation angle of the joint needs to be readjusted. ● If the x/y/z cannot move simultaneously with the Cartesian linear motion, please decompose the position. ● Try to avoid displacement and rotation in the same Cartesian command.

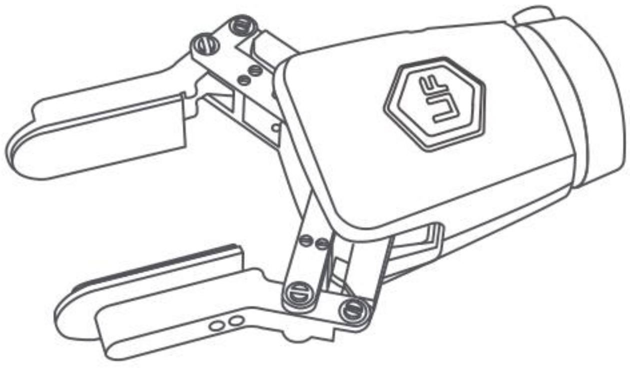
	● When the error mechanism occurs, some need to be cleared manually to keep the robotic arm in normal operation.
WARNING	● Approach the joint speed, joint range, TCP speed, and the TCP position range may cause an error reporting mechanism. ● Exceeding the specified joint speed, joint range, TCP speed, and TCP position will trigger an error reporting mechanism.

9. End-effector

9.1. Gripper

The gripper is a robotic arm end-effector that can dynamically pick up objects.

The position range of the robotic arm: -10 to 850, the larger the value, the larger the open degree. On the contrary, the smaller the value, the smaller the open degree. If the clamping is not tight, a negative value can be sent until it is tightened.



9.1.1. The Flow of Gripper Movement：

1. Enable the gripper
2. Send out a position for clamping
3. The current range of value: -10 ~ 850

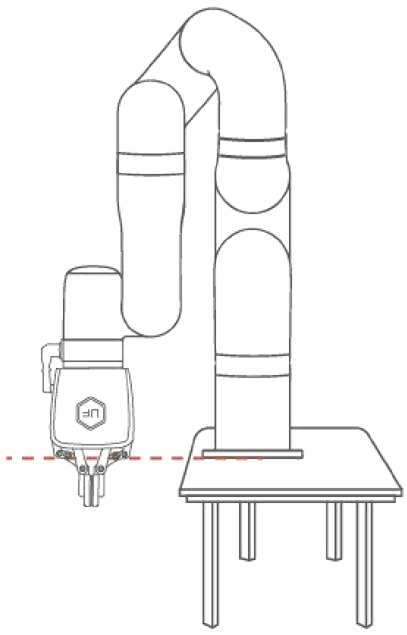
9.1.2. Precautions

DANGER

1. The Robot with end-effector angle equal to zero will exceed the mounting surface, please adjust the robotic arm to a posture suitable for mounting the mechanical claws during installation.

2. When trajectory planning is performed on a robotic arm with a robotic arm claw, safety assessment must be performed between position commands to avoid collision with nearby equipment.

3. When the robotic arm with the robotic arm claw is used for trajectory planning, you must be performed for safety evaluation to see whether the robotic arm can return to the angle is defined to be zero.



The gripper of the robotic arm in the zero position will exceed the mounting surface.

10. xArm-Python-SDK Library

10.1. Introduction

The xArm-Python-SDK is an integrated library for controlling a series of motion planning that is based on controller communication protocol package, and the SDK is for advanced users. When controlling the real machine, make sure the robotic arm will not touch any obstacle. Please read this manual and the Python interface instructions carefully before using the Python library.

Note: The Python chapter of this manual was written in January 2019.

For the latest code and document, please check github：https://github.com/xArm-Developer/xArm-Python-SDK

10.2. Download and Install the Python Library

Before running the Python library, make sure the computer have the required operating environment.

To download Python, please check the official website of Python: https://www.python.org/

Python version： Python 3 and above

System： Win/Mac/Linux

To download the xArm-Pyhton-SDK： https://github.com/xArm-Developer/xArm-Python-SDK

Installation package： git clone git@github.com:xArm-Developer/xArm-Python-SDK.git

Installation： python setup.py install

10.3. Get started

Before controlling the robotic arm motion, make sure the hardware connection is successful, please check the following:

1. The robotic arm is installed properly and will not touch any obstacle.
2. The cable is connected properly.

3. The controller is turned on and the status indicator lights up.
4. The communication between the controller and the computer is successful, via Ethernet TCP/IP (the controller LAN indicator lights up) or RS-485.
5. The xArm power indicator lights up and the robotic arm is powered.

10.4. Typical motion flow of the robotic arm (position mode)

a. Connect the robotic arm

b. Enable the robotic arm (enable the servo motor)

c. Set the robotic arm mode as '0' (position mode)

d. Set the robotic arm status as '0' (motion status)

e. Send location

10.5. Initialize the robotic arm motion mode

Please check below for the Python code and click on the link for more Example s: **https://github.com/xArm-Developer/xArm-Python-SDK/tree/dev/ Example /wrapper**

```
'''
Note:
Initialize the robotic arm
1.Import the XArmAPI interface
'''
import os
import sys
sys.path.append(os.path.join(os.path.dirname(__file__), '../..'))
from xarm.wrapper import XArmAPI
'''
Note:
# 2.Connect the robotic arm controller
# Note: 1: When connecting by IP address, the default network
segment address of the controller is 192.168.1.*(2~254). For
```

specific IP address, please check the controller: connect with xArm, and change the computer IP address to 192.168.1.* (2~254) network segment.

2: Connected via 485 interface, and 485 communication cord is required. e.g.: xarm = XArmAPI('com5')

'''

xarm = XArmAPI('192.168.1.113')

'''

Note:

# 3. Enable xArm robotic arm (enable the servo motor)

'''

xarm.motion_enable(enable=True)

'''

Note:

# 4. Set the position mode of the xArm robotic arm

# parameter 0 indicates that the robotic arm is set as position mode

'''

xarm.set_mode(0)

'''

Note:

# 5. Set the motion status of the xArm robotic arm

# parameter state=0 indicates that the robotic arm enter motion status '''

xarm.set_state(state=0)

'''

Note:

# 6. Set the position of the xArm robotic arm

# angle=[20, 0, 0, 0, 0, 0, 0] indicates that joint 1 of the robotic arms is rotated by 20°, and the remaining joints are not rotated.

speed = 30 indicates that the joint rotation speed is 30°/s

# is_radian = False means the unit is not the radian (rad), but the degree (°)

```
# wait = True means waiting for the robotic arm to complete the command
'''
xarm.set_servo_angle(angle=[20, 0, 0, 0, 0, 0, 0], speed=30,
is_radian=False, wait=True)
```

10.6. Alarm ResponseAlarm Handling Mode

When designing the robotic arm motion path with the Python library, if the robotic arm error mechanism (see Appendix for Alarm information) occurs, then it needs to be cleared manually. After clearing the error, the robotic arm should be motion enabled. Set the motion mode to allow normal operation of the robotic arm. Then the path planning of the robotic arm should be re-adjusted according according to the reported error information.

Python library error clearing steps: (Please check GitHub for details on the following interfaces)

a. error clearing： clean_error()

b. Re-enable the robotic arm： motion_enable(true)

c. Set the motion state： set_state(0)

d. Set the position of xArm: set_position(x=300)

10.7. Parameter and Interface Command

For more information on the Python library and the interface command details, please scan the QR code or click on the link :https://github.com/xArm-Developer/xArm-Python-SDK

11. ROS Library

11.1. Introduction

This code library contains sample development kit for xArm model files and related controls, plans, etc. The environment for development and testing is Ubuntu 16.04 + ROS Kinetic Kame. New function support, sample code, bug fixes, etc. will be updated, and the tutorial is based on the online version.

For the latest code and document, please check our github: https://github.com/xArm-Developer/xarm_ros

11.2. Preparation

1. Install the gazebo_ros interface module

gazebo_ros_pkgs: http://gazebosim.org/tutorials?tut=ros_installing

ros_control: http://wiki.ros.org/ros_control (Choose the ROS version you are using)

2. Complete learning on related official tutorial

ROS Wiki: http://wiki.ros.org/

Gazebo Tutorial: http://gazebosim.org/tutorials

Gazebo ROS Control: http://gazebosim.org/tutorials/?tut=ros_control

3. If using Gazebo: please download the 'table' 3D model in advance

This model will be used in the Gazebo demo. In the Gazebo simulation environment, look for 'table' in the model database list and drag the model into the next 3D environment. With this operation, the 'table' model is automatically downloaded to the local.

11.3. Additional Information

For more information, please click: https://github.com/xArm-Developer/xarm_ros or scan the QR code below

Appendix

1. Technical Specifications

xArm		
Joint Range	1,3,5,7	±360°
	2	±125°
	4	-230°~6°
	6	-100°~180°
Cartesian Range	X	±700mm
	Y	±700mm
	Z	-400mm~951.5mm
	Roll/Yaw/Pitch	± 180°
Maximum joint speed		180°/s
Payload		3.5kg
Reach		700mm
Degrees of Freedom		7
Repeatability		±0.1mm
Max Speed of End-effector		1m/s
Weight(robot arm only)		11.5kg
Weight(controller box only)		3.5kg
Controller Size		260mm*180mm*100mm(L*W*H)
*Ambient Temperature Range		0-50 °C*
Power Consumption		Min 8.4 W, Typical 120 W, Max 240 W
Input Power Supply		24 V DC, 15 A
Power Supply		100-240 V AC, 50-60 Hz
IP Classification		IP54
ISO Class Cleanroom		5

Robotic Arm Mounting	Any
Controller I/O interface	8 digital inputs, 8 digital outputs, 2 analog inputs, 2 analog outputs,
End-effector I/O interface	2 digital inputs, 2 digital outputs, 2 analog inputs, RS485
Programming	Python/C++/ROS underlying interface
Robotic Arm Communication Protocol	Self-defined
Controller Communication Protocol	TCP-IP/modbus RTU
Controller Communication Model	RS-485/Ethernet
End-effector communication protocol	RS-485/modbus RTU
Footprint	Ø 126 mm
Materials	Aluminum、 Carbon Fiber
Tool Connector Type	M5*6
Gripper	
Nominal Supply Voltage	24V DC
Absolute Maximum Supply Voltage	28V DC
Quiescent Power (Minimum Power Consumption)	1.5W
Peak Current	1.5A
Communication Mode	RS-485
Communication Protocol	Modbus RTU
Programmable Gripping Parameters	Position, Speed and Force Control
Status indicator	Error status, power
Feedback	Current, position
Suction Cup	

Payload	4kg
Suction Cups	5
Maximum Vacuum Degree	-90kpa
Notes: *1. The ambient temperature of xArm is 0-50 °C, and reduces when the joint continued for high-speed operation.	

2. Robot Movement & Status Analysis

There are four types of controller motion mode:

0：Position Control mode (joint motion, linear motion, circular, linear motion)

1：Servo motion mode (servo motion)

2：Joint teaching mode (not available)

3：Cartesian teaching mode (not available)

There are three types of controller status setting:

3：Suspend the current movement

4：Stop all the current movements (including command buffer, restart the system)

0：Start the movement

There are four types of controller status acquisition:

1：In movement —— executing the movement instructions

2：Sleeping —— the controller is ready in the movement status, but no movement commands can be executed

3：Suspended

4：Stop —— default start-up status, the robotic arm will not receive and execute all movement-related commands under the default start-up status; but will receive and execute system settings and get information-type commands. The setting needs to be set as [0: start the movement] to ensure normal execution of movement instructions.

3. Controller Communication Protocol

Protocol Format

3.1.【The Meaning of symbols】

【u8】:	1 byte, 8-bit unsigned int
【u16】:	2 bytes, 16-bit unsigned int
【fp32】:	4 bytes, float
【str】:	String

【System Reset 】: After using this command, the command cache will be cleared, and the robotic arm movement will stop. The robotic arm status will jump to stop, and the current position of the robotic arm will be retrieved.

3.2.【UXBUS Communication Format】

Parameters

Default Baud Rate：2M

Default controller ID：0X55（follow ID）

Default Host ID：0XAA（main ID）

controller Broadcast ID：0XFF

Required Length： Register Length(bytes) + parameter Length(bytes) = 1 + n

Response Length： status Length(bytes) + Reply parameters Length(bytes) = 1 + n

required：

Request instruction format

Format	Main ID(u8)	Follow ID(u8)	Length (u8)	Register (u8)	Parameter	Modbus_crc (u16)
Length	1 byte	1 byte	1 byte	1 byte	n byte	2 byte
Example (enable servo)	0xAA	0x55	0x03	0x0B	0x09 0x01	0xA2 0x0B

Response command format

Format	Follow ID(u8)	Main ID(u8)	Length (u8)	Status (u8)	Reply Parameters	Modbus_crc (u16)
Length	1 byte	1 byte	1 byte	1 byte	n byte	2 bytes
Example (enable servo)	0x55	0xAA	0x01	0x00	none	0x30 0x58

3.3.【MODBUS-TCP Communication Format】

Parameters

Default TCP Port：502

protocol: 0x00 0x02 Control (Only this one for now)

Request instruction format

Format	Transaction Identifier (u16)	Protocol (u16)	Length (u16)	Register (u8)	Parameters (refer to the statement of each command)
Length	2 bytes	2 bytes	2 bytes	1 byte	n byte
Example (enable servo)	0x00 0x11	0x00 0x02	0x00 0x03	0x0B	0x09 0x01

Response command format

Format	Transaction Identifier (u16)	Protocol (u16)	Length (u16)	Register+Status (u8)+(u8)	Parameters (refer to the statement of each command)
Length	2 bytes	2 bytes	2 bytes	2 bytes	n byte
Example (enable servo)	0x00 0x11	0x00 0x02	0x00 0x02	0x0B 0x00	none

Status Bit of the Response Format

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
0: normal	1: err 0: normal	1：war 0: normal	0: normal	0: normal	0: normal	0: normal	0: normal

3.4.【Automatic Reporting Format】

REPORT_TCP_DEVELOP

Default Port：30003

Frequency：100Hz

Byte Order Content：

1~4 bytes：number of bytes

5 bytes：bit0~bit3 indicates the motion status, bit4~bit7 indicates the motion mode

6~7 bytes：Number of Instruction Caches (u16)

8~35 bytes：Current angle of the robotic arm (fp32 * 7)

36~59 bytes：Current position of the robotic arm (fp32 * 6)

60~87 bytes：Joint torque (fp32 * 7)

REPORT_TCP_NORMAL

Default port：30001

Frequency：5Hz

Byte Order Content：

1~87 bytes：Same as【the Auto Reporting Format of REPORT_TCP_DEVELOP】

88 byte：servo brake status (u8 bit0 ~ bit correspond to 1~6 joints respectively, 0 not enabled, 1 enabled)

89 byte：servo brake status (u8 bit0 ~ bit correspond to 1~6 joints respectively, 0 not enabled, 1 enabled)

90 byte：Error code (u8)

91 byte：Alarm code (u8)

92~115 bytes：TCP Offset (fp32 * 6)

116~131 bytes：End load parameter (fp32 * 4)

132 bytes：Collision detection sensitivity (u8)

133 byte：Hand-guiding teaching sensitivity (u8)

3.5.【Register】

0~10： Public Port Segment

11~100： Motion Control Port Segment

101~200：Hardware Module Port Segment

201~250：System Advanced Port Segment

0~10 Common Port Section

Get version information(0x01)

Register：1(0x01)

Function：Acquire version information

Required parameters：none

Reply Parameters: str

Remote shutdown (0x10)

Register: 10(0x0A)

Function：Remotely shut down the operating system

Required parameters: 1 byte

Parameter 1(u8): Operation

1: Remotely shut down the operating system temporarily

Reply Parameters: none

11~20 System Status

Register：11(0x0B) —— Enable/Disable servo (System reset)	
Required parameters	Length：2 bytes parameter 1(u8)：axis 1~6：motor joint (1~6) 8：act on all joints parameter 2(u8)： 1：Enable servo 0：Disable servo

Reply parameters	none
Example (MODBUS-TCP)	Enable all servos：00 01 00 02 00 03 0b 09 01 Disable all servos：00 01 00 02 00 03 0b 09 00
xArm-Python-SDK interface	motion_enable()
Register：12(0x0C) —— Motion Status Setting	
Required parameters	Length：1 byte parameter 1(u8)：Motion Status 3：suspend the current motion 4：stop all current movements（restart the system） 0：enter the motion mode
Reply parameters	none
Example (MODBUS-TCP)	00 01 00 02 00 02 0c 00 Set the state to 0
xArm-Python-SDK interface	set_state()
Register：13 (0x0D) —— Acquire the motion status	
Required parameters	none
Reply parameters	state(u8): motion status 1：In motion 2：Sleep 3：Suspend 4：Stop
Attention	Note: The difference between Stop and Sleep: both states are

	stationary, but xArm can receive motion commands and run in Sleep mode, while it can also receive motion commands in Stop mode but will not be able to run. User needs to reset xArm to [On Motion] to run such motion commands, when the system environment is not ready or an error has occurred, the xArm may stay in Stop mode.
xArm-Python-SDK interface	get_state()
Register：14 (0x0E) —— Acquire the number of commands in the command buffer	
Required parameters	none
Reply parameters	Length：2 bytes parameter1(u16): The number of commands in the command buffer
Example 1(MODBUS-TCP)	00 01 00 02 00 01 0e
xArm-Python-SDK interface	get_cmdnum()
Register：15—— Acquire error and warning code	
Required parameters	none
Reply parameters	Length：2 bytes parameter 1(u8): error code parameter 2(u8): warning code
Example 1(MODBUS-TCP)	00 01 00 02 00 01 0e
xArm-Python-SDK interface	get_err_warn_code()
Register：16 (0x10)—— Clear Controller error（System reset）	
Required parameters	none
Reply parameters	none
xArm-Python-SDK interface	clean_error()
Register：17—— Clear Controller warning	

Required parameters	none
Reply parameters	none
xArm-Python-SDK interface	clean_warn()
Register：18 (0x12) —— Setting the brake Switches Separately (System reset)	
Required parameters	Length：2 bytes parameter 1(u8)：Control the brakes 1~6： Select a motor joint separately 8 ： acts on all joints parameter 2(u8)：operation 1：enable the servo 0： release the brake
Reply parameters	none
xArm-Python-SDK interface	enable the servo：set_servo_attach() release the brake：set_servo_detach()
Register：19 (0x13) —— Setting the System Motion Mode (System reset)	
Required parameters	Length：1 byte parameter 1(u8)：Motion Mode 0 ：position control mode 1 ：servo motion mode 2 ：joint teaching mode (not available) 3 ：Cartesian teaching mode (not available)

Reply parameters	none
xArm-Python-SDK interface	set_mode()

21~30 Basic Motion

Register：21 (0x15) —— Cartesian linear motion	
Required parameters	Length：36 bytes parameter1(fp32*6):Cartesian coordinates[X Y Z A B C],unit： mm/radian parameter2(fp32)：motion speed, unit： mm/s,rad/s parameter3(fp32)：acceleration, unit： mm/s^2, rad/s^2 parameter4(fp32)：motion time（no meaning, 0）
Reply parameters	Length：2 bytes parameter1(u16) ： the number of commands in the instruction buffer
xArm-Python-SDK interface	set_position()
Register：22 (0x16) —— Linear motion with circular arc	
Required parameters	Length：40 bytes parameter1(fp32*6):Cartesian coordinates[X Y Z A B C], unit： mm/rad parameter2(fp32)：arc blending radius, unit： mm parameter3(fp32)：motion speed, unit： mm/s, rad/s parameter4(fp32)：acceleration, unit： mm/s^2, rad/s^2 parameter5(fp32)：motion time（no meaning, 0）

Reply parameters	Length：2 bytes parameter1(u16) ： the number of commands in the command buffer
xArm-Python-SDK interface	set_position()
Register：23 (0x17) —— P2P Joint Motion	
Required parameters	Length：40 bytes parameter1(fp32*7)：correspond to axis1~6 respectively,unit: rad parameter2(fp32)：motion speed, unit: rad/s parameter3(fp32)：acceleration, unit: rad/s^2 parameter4(fp32)：motion time（no meaning, 0）
Reply parameters	Length：2 bytes parameter1(u16): the number of commands in the instruction buffer num(u16): the number of commands in the command buffer
Example 1(MODBUS-TCP)	Example ： 00 01 00 02 00 29 17 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000 00 00 00 00 00 00 00 00 c2 b8 32 3e c2 b8 32 3e 00 00 00 00 （Note: Set the joint position to [0 0 0 0 0 0 0] speed is 10*pi/180 radians per second, acceleration is 10*pi/180 radians per square second
xArm-Python-SDK interface	set_servo_angle()
Register：25 (0x19) —— Return to ZeroPosition	
Required parameters	Length：12 bytes parameter1(fp32)：motion speed, unit: rad/s parameter2(fp32)：acceleration, unit: rad/s^2

	parameter3(fp32)：motion time（no meaning, 0）
Reply parameters	Length：2 bytes parameter1(u16): the number of commands in the instruction buffer num(u16) the number of commands in the command buffer
xArm-Python-SDK interface	move_gohome()
Register：26(0x1A) —— Pause Commands, Commands Delay	
Required parameters	Length：4 bytes parameter1(fp32)： pause time, unit: s
Reply parameters	Length：2 bytes parameter1(u16): number of commands in the instruction buffer
xArm-Python-SDK interface	set_pause_time()
Register: 27(0x1B)——Linear Circular Motion The motion calculates the trajectory of the space circle according to the three-point coordinates, and the three-point coordinates are (current starting point, parameter 1, parameter 2)	
Required parameters	Length: 64 bytes parameter1(fp32*6): Cartesian coordinates [X Y Z A B C], unit: mm/radian parameter2(fp32*6): Cartesian coordinates [X Y Z A B C], unit: mm/radian arameter3(fp32): Percentage of the length of arc in motion to circumference, unit: % parameter4(fp32): motion speed, unit: m/s, radian/s parameter5(fp32): acceleration, unit: mm/s^2, radian/s^2 parameter6(fp32): motion time (no meaning for now, 0)
Reply parameters	Length: 2 bytes parameter1(u16): number of commands in the command buffer

xArm-Python-SDK interface	set_position()
Register：29 (0x1D) —— Servoj Motion	
Required parameters	Length：40 bytes parameter1(fp32*7)：correspond to axis 1~6 respectively, unit: rad parameter2(fp32)：motion speed（no meaning for now,0） parameter3(fp32)：acceleration（no meaning for now,0） parameter4(fp32)：motion time（no meaning for now,0）
Reply parameters	none
xArm-Python-SDK interface	set_servo_angle_j()

31~40 Motion Parameter Setting

Register：31 (0x1F) —— Set the jerk of the Cartesian space translation	
Required parameters	Length：4 bytes parameter1(fp32)：jerk, unit: mm/s^3
Reply parameters	Length：2 bytes parameter1(u16): number of commands in the command buffer
xArm-Python-SDK interface	set_tcp_jerk()
Register：32 (0x20) —— Set the maximum acceleration of the Cartesian space translation	
Required parameters	Length：4 bytes

	parameter1(fp32)： max acceleration, unit: mm/s^2
Reply parameters	Length： 2 bytes parameter1(u16): number of commands in the instruction buffer
xArm-Python-SDK interface	set_tcp_maxacc()
Register： 33 (0x21) —— Set joint space jerk	
Required parameters	Length： 4 bytes parameter1(fp32)： jerk, unit: radians/s^3
Reply parameters	Length： 2 bytes parameter1(u16): number of commands in the command buffer
xArm-Python-SDK interface	set_joint_jerk()
Register： 34 (0x22) ——Set joint space max acceleration	
Required parameters	Length： 4 bytes parameter1(fp32)： max acceleration, unit: radians/s^2
Reply parameters	Length： 2 bytes parameter1(u16): number of commands in the instruction buffer
xArm-Python-SDK interface	set_joint_maxacc()
Register： 35 (0x23) ——Set the offset of the robotic arm end-effector(System reset)	
Required parameters	Length： 24 bytes parameter1(fp32*6):TCP offset [X Y Z A B C], unit： mm
Reply parameters	none
xArm-Python-SDK interface	set_tcp_offset()

Register：36 (0x24) ——End load setting (System reset)	
Required parameters	Length：16 bytes parameter1(fp32)：payload, unit：kg parameter2(fp32*3)：The load center of gravity[x y z], unit：mm
Reply parameters	none
xArm-Python-SDK interface	set_tcp_load()
Register：37(0x25) ——Set collision detection sensitivity (System reset)	
Required parameters	Length：1 byte parameter1(u8)：detect sensitivity
Reply parameters	none
xArm-Python-SDK interface	set_collision_sensitivity()
Register：38(0x26) ——Set hand-guiding sensitivity for teaching mode (System reset)	
Required parameters	Length：1 byte parameter1(u8)：teach sensitivity
Reply parameters	none
xArm-Python-SDK interface	set_teach_sensitivity()
Register：39 (0x27) ——Delete the current system configuration parameters	
Required parameters	none
Reply parameters	none
xArm-Python-SDK interface	clean_conf
Register：40 (0x28) ——Save the current system configuration parameters	
Required parameters	none

Reply parameters	none
xArm-Python-SDK interface	save_conf()

41~50 Acquire Motion Information

Register：41 (0x29) ——Acquire the current Cartesian position of the controller	
Required parameters	none
Reply parameters	Length：24 bytes parameter1(fp32*6): Cartesian coordinates, unit: mm, rad
xArm-Python-SDK interface	get_position()
Register：42 (0x2A) ——Acquire the current joint position of the controller	
Required parameters	none
Reply parameters	Length：28 bytes parameter1(fp32*7): the angle of 1~6 joints, unit: rad
xArm-Python-SDK interface	get_servo_angle()
Register：43 (0x2B) ——Acquire the solution of the positive kinematics	
Required parameters	Length：28 bytes parameter1(fp32*7)：the angle of 1~6 joints, unit: rad
Reply parameters	Length：24 bytes parameter1(fp32*6)：Cartesian coordinates, unit: mm, rad
xArm-Python-SDK interface	Forward kinematics：get_forward_kinematics() Inverse Kinematics: get_inverse_kinematics()
Register：45 (0x2D) ——Check the limit of the joint space	

Required parameters	Length：28 bytes parameter1(fp32*7)：the angle of 1~6 joints, unit: rad
Reply parameters	Length：1 byte parameter1(u8)：search result 1：collision occurs 0：no collision
xArm-Python-SDK interface	is_joint_limit()
Register：47 (0x2F) —— Acquire the current joint torque of the servo	
Required parameters	none
Reply parameters	Length：28 bytes parameter1(fp32*7): The moment of joints 1~6, unit：N.m
xArm-Python-SDK interface	

101~115 Servo Module

Register：105 (0x69) ——Set the current robotic arm joint position as '0'	
Required parameters	Length：1 byte parameter1：Selective servo joint 1~6：motor joint (1~6) 8：act on all joints
Reply parameters	none
xArm-Python-SDK interface	move_gohome()
Register：106 (0x6A) ——Acquire the status of the current robotic arm servo	

Required parameters	none
Reply parameters	Length： 17 bytes parameter1(u8)：Instruction execution state 0：Normal 1： The server has an error message 3： Communication fail parameter2(u8*2)：Joint1 [servo status + servo error code] parameter3(u8*2)：Joint2 [servo status + servo error code] parameter4(u8*2)：Joint3 [servo status + servo error code] parameter5(u8*2)：Joint4 [servo status + servo error code] parameter6(u8*2)：Joint5 [servo status + servo error code] parameter7(u8*2)：Joint6 [servo status + servo error code] parameter8(u8*2)：Joint7 [servo status + servo error code] parameter9(u8*2)：gripper [servo status + servo error code]
xArm-Python-SDK interface	get_servo_debug_msg()

124 Gripper Module

Register: 124 (0x7C) ——enable/disable the gripper	
Required parameters	Length: 10 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes

	parameter6: 2*Data Length, 1 byte parameter7: Data, 2 bytes 0: Enable the gripper 1: Disable the gripper
Reply parameters	Length: 7 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes
Example1 (MODBUS-TCP)	Enable: 00 01 00 02 00 0B 7C 09 08 10 01 00 00 01 02 00 01 Disable: 00 01 00 02 00 0B 7C 09 08 10 01 00 00 01 02 00 00
xArm-Python-SDK interface	set_gripper_enable()
Register: 124(0x7C) ——Set the gripper mode	
Required parameters	Length: 10 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes parameter6: 2*Data Length, 1 byte parameter7: Data, 2 bytes 0: Position Mode 1: Speed Mode
Reply parameters	Length: 7 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte

	parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes
Example1 (MODBUS-TCP)	Position Mode: 00 01 00 02 00 0B 7C 09 08 10 01 01 00 01 02 00 00
xArm-Python-SDK interface	set_gripper_mode()
Register: 124 (0x7C) ——Set the zero point of the gripper	
Required parameters	Length: 10 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes parameter6: 2*Data Length, 1 byte parameter7: Data, 2 bytes
Reply parameters	Length: 7 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes
Example1 (MODBUS-TCP)	00 01 00 02 00 0B 7C 09 08 10 08 17 00 01 02 00 01
xArm-Python-SDK interface	set_gripper_position()
Register: 124 (0x7C) ——Acquire the gripper position	
Required parameters	Length: 7 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes

	parameter5: Data Length, 2 bytes
Reply parameters	Length: 8 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Data Length, 1 byte parameter5: Data, 4 bytes, current position.
Example1 (MODBUS-TCP)	00 01 00 02 00 08 7C 09 08 03 07 02 00 02
xArm-Python-SDK interface	get_gripper_position()
Register：124 (0x7C) ——Set the gripper position	
Required parameters	Length: 12 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes parameter6: 2*Data Length, 1 byte parameter7: Data, 4 bytes, position command
Reply parameters	Length: 7 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes
Example1 (MODBUS-TCP)	00 01 00 02 00 0D 7C 09 08 10 07 00 00 02 04 00 00 00 00
xArm-Python-SDK interface	set_gripper_position()
Register: 124 (0x7C) ——Set the gripper position speed	

Required parameters	Length: 10 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes parameter6: 2*Data Length, 1 byte parameter7: Data, 2 bytes. Speed command, unit r/min
Reply parameters	Length: 7 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes
Example1 (MODBUS-TCP)	00 01 00 02 00 0B 7C 09 08 10 03 03 00 01 02 05 DC
xArm-Python-SDK interface	set_gripper_speed()
Register：124 (0x7C) ——Acquire the gripper error	
Required parameters	Length: 7 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes
Reply parameters	Length: 6 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Data Length, 1 byte parameter5: Data, 2 bytes, current non-zero error message

	indicates an error
Example1 (MODBUS-TCP)	00 01 00 02 00 08 7C 09 08 03 00 0F 00 01
xArm-Python-SDK interface	get_gripper_err_code()

Register: 124 (0x7C) ——Clear the gripper error	
Required parameters	Length: 10 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes parameter6: 2*Data Length, 1 byte parameter7: Data, 2 bytes
Reply parameters	Length: 7 bytes parameter1: Host ID, 1 byte parameter2: Gripper ID, 1 byte parameter3: Function Code, 1 byte parameter4: Address, 2 bytes parameter5: Data Length, 2 bytes
Example1 (MODBUS-TCP)	00 02 00 02 00 0B 7C 09 08 10 01 09 00 01 02 00 01
xArm-Python-SDK interface	clean_gripper_error()

131~145 Controller GPIO module

Register: 131 (0x83) - Get configurable digital gpio input	
Send parameter	none
Response parameter	Length：2 bytes

	Parameter 1(u16)： gpio signal, bit0 ~ bit5 correspond to signals of gpio0~gpio5
xArm-Python-SDK interface	get_cgpio_digital()
Register： 132 (0x84) ——Obtain analog input AIN1	
Send parameter	none
Response parameter	Length： 2 bytes parameter1(u16)： analog input1, Range 0~4096, Corresponding to0~10V
xArm-Python-SDK Interface	get_cgpio_analog()
Register： 133 (0x85) ——Obtain analog input AIN2	
Send parameter	none
Response parameter	Length： 2 bytes parameter1(u16)： analog input2, Range 0~4096, Corresponding to0~10V
xArm-Python-SDK Interface	get_cgpio_analog()
Register： 134 (0x86) ——Set configurable digital gpio output	
Send parameter	Length： 2 bytes parameter1(u16)： Gpio signal, the upper 8 bits are the enable bits, and the lower 8 bits are the set bits
Response parameter	none
xArm-Python-SDK Interface	set_cgpio_digital()
Register： 135 (0x87) ——Set the analog outputAIO1	

Send parameter	Length：2 bytes parameter1(u16)：analog output1, Range 0~4096, Corresponding to 0~10V
Response parameter	none
xArm-Python-SDK Interface	set_cgpio_analog()
Register：136 (0x88) ——Set the analog output AIO2	
Send parameter	Length：2 bytes parameter1(u16)：analog output2, Range 0~4096, Corresponding to 0~10V
Response parameter	none
xArm-Python-SDK Interface	set_cgpio_analog()
Register：138 (0x8A) ——Configuring digital output IO Function	
Send parameter	Length：2 bytes parameter1(u8)：gpio serial number,0~5 Corresponding to gpio0 ~ gpio5 parameter2(u8)：function number 0：The system is in 'STOP' status 1：System error 2：xarm is operating
Response parameter	none
xArm-Python-SDK Interface	set_cgpio_digital_output_function()
Register：139 (0x8B) ——Obtain GPIO status	
Send parameter	none
Response parameter	Length：34 bytes

	parameter1(u8)：gpio Module status 0 normal 3 Gripper has error message 6 Communication failure parameter2(u8): gpio module error code 0: normal Not 0：Error code parameter3(u16)：Digital input function io status parameter4(u16)：Digital input configuration io status parameter 5(u16)：Digital output function io status parameter 6(u16)：Digital output configuration io status parameter 7(u16)：Analog input1 parameter 8(u16)：Analog input2 parameter 9(u16)：Analog output1 parameter 10(u16)：Analog output2 parameter 11(u8*8)：Digital input io configuration message parameter 12(u8*8)：Digitial ouptu io configuration message
xArm-Python-SDK Interface	get_cgpio_state()

4. Joints Alarm Message and General Response

Users can power off and on the system as an alarm response, the steps are as follows (re-powering needs to go through all the following steps):

1.Re-powering the robotic arm via the emergency stop button on the controller.

2. Enable robot arm

xArm Studio enable mode: Click the guide button of the error pop-up window or the 'STOP' red button in the upper right corner.

xArm-Python-SDK enable mode: Refer to 11.6 Alarm Handling Mode.

Xarm_ros library: users can view related documents at https://github.com/xArm-Developer/xarm_ros.

If the problem remains unsolved after power on/off for multiple times, please contact UFACTORY team for support.

Alarm Code	Alarm Description	Alarm Response
S10	Abnormal current detection	Power on again
S11	Joint overcurrent	Check if the brake is normally open
S12	Joint overspeed	Reduce the moving speed
S14	Position command is too large	Check if the position setting value is too large
S15	Joint overheat	Check if it is caused by overtime operation
S16	Encoder initialization abnormal	Check if the robotic arm moves during power-on and powers on again.
S17	Single-turn encoder failure	Power on again
S18	Multi-turn encoder failure	Power on again
S19	Warning of low battery voltage	Power on again
S20	Driver IC hardware alarm	Power on again
S21	Driver IC initialization abnormal	Power on again
S22	Encoder configuration error	Power on again
S23	Position deviation is too large	Check if it is blocked and the position setting value is too large

S26	Joint forward overrun	Check if the joint position is overrun
S27	Joint negative overrun	Check if the joint position is overrun
S28	Joint command error	Check if the joint is disabled
S33	Driver overload	Check if the load is too large
S34	Motor overload	Check if the load is too large
S35	Motor type error	Power on again
S36	Driver type error	Power on again
S39	Joint overvoltage	Reduce the acceleration
S40	Joint undervoltage	Reduce the acceleration
S49	EEPROM read and write error	Power on again
S52	Motor angle initialization failed	Power on again
For alarm codes that are not listed in the above table: Power on again. If the problem remains unsolved after power on/off for multiple times, please contact technical support.		

5. Controller Error Code and General Response

5.1 Controller Error Code

If there is an error in the hardware of the robotic arm/the software of the controller/in sending command, an error or warning will be issued. This error/warning signal will be fed back when the user sends any command; that is, the feedback is passive and not actively reported.

After the above error occurs, the robotic arm will stop working immediately and discard the controller cache command. Users need to clear these errors manually to allow normal operation. Please re-adjust the motion planning of the robotic arm according to the reported error message.

Error Code	Description	Alarm Response
C11~C17	11~17 respectively are 1st to 7th joint	Check if the joint range is out of range
C21	Kinematic error	Please re-plan the path.
C23	Joint angle exceeds limit	Adjust the joint to the specific range
C24	Speed exceeds limit	Please reduce the speed and acceleration values
C25	Planning error	Please re-plan the path or reduce the speed of the movement.
C26	Linux rt Error	Please contact technical support.
C27	Reply command error	Power on again
C29	Other errors	Please contact technical support.
C30	Feedback speed exceeds limit	Feedback speed exceeds limit. Please contact technical support.
C31	Collision causes abnormal current	Check if there is a collision, the load setting is correct, and the teaching sensitivity matches the speed.
C32	Three-point arc command calculation error	Please re-plan the path.
C33	Controller gpio module error	Controller GPIO module abnormal. If the error occurs repeatedly, please contact technical support.
C34	Trajectory recording overtime	Trajectory recording overtime. Trajectory recording time exceeds the maximum limit of 5 minutes, please re-record.
C35	Reach the safety boundary	The robotic arm reaches the safety boundary. When the robotic arm reaches the safety boundary, please move the robotic arm into the safety boundary

		after turning on Manual Mode on the interface of Real Time Control.

5.2 Controller Alarm Code

Alarm does not affect the normal operation of the robotic arm, but it may affect the user's program operation. Once the warning occurs, the arm will set the warning flag and return it together in the command reply. Otherwise, no other operations will be performed. The robotic arm will still operate normally.

Alarm code	Description	Alarm Response
11 (0x0B)	Buffer overflow	Controller command cache
12 (0x0C)	Command parameter abnormal	Check sent command
13 (0x0D)	Command doesn’t exist	Check sent command
14 (0x0E)	Unknown Command	Check sent command

6. Gripper Alarm Code & General Response

Users can power off and on the system as an alarm response, the steps are as follows (re-powering needs to go through all the following steps):

1.Re-powering the robotic arm via the emergency stop button on the controller.

2. Enable robot arm

xArm Studio enable mode: Click the guide button of the error pop-up window or the ‘STOP’ red button in the upper right corner.

xArm-Python-SDK enable mode: Refer to 11.6 Alarm Handling Mode.

Xarm_ros library: users can view related documents at https://github.com/xArm-Developer/xarm_ros

1. Re-enable the mechanical claws:

If the problem remains unsolved after power on/off for multiple times, please contact UFACTORY team for support.

Alarm Code	Alarm Description	Alarm Response
G9	Abnormal current detection	Power on the robotic arm again and enable the gripper
G11	Gripper overcurrent	Re-enable the gripper.
G12	Gripper overspeed	Power on again and enable the gripper
G14	Position command is too large	Please check if the position setting value is too large
G15	EEPROM read and write error	Power on again
G20	Driver IC hardware alarm	Power on again
G21	Driver IC initialization abnormal	Power on again
G23	Position deviation is too large	Please check if it is blocked or not, and the position setting value is too large.
G25	Gripper command overrun	Please check the gripper command
G26	Gripper position overrun	Please check the gripper position
G33	Driver overload	Please check if the load is too large
G34	Motor overload	Please check if the load is too large
G36	Driver type error	Power on again
For alarm codes that are not listed in the above table: enable the robotic arm and gripper. If the problem remains unsolved after power on/off for multiple times, please contact technical support.		