UFACTORY

UFACTORY XARM GRIPPER

USER MANUAL

SHENZHEN UFACTORY CO., LTD

V.1.6.1

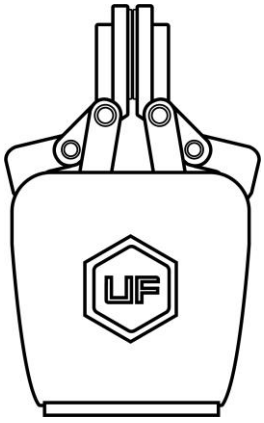
Table

1. General Presentation

1.1. Gripper Introduction

The gripper is the end-effector of the robotic arm, which can grasp objects dynamically.

The value range of the gripper opening and closing is: -10 to 850. The larger the value, the greater the stroke of the gripper, meaning the smaller the value, the smaller the stroke of the gripper. If the clamping is not tight, a negative value can be set until it is tightened.



1.2. Setup and Control

The gripper is powered and controlled directly via a single gripper connection cable that carries a 24V DC supply and Modbus RTU communication over RS-485.

1.3.Safety

Warning

The operator must have read and understood all of the instructions in the following manual before handling the xArm Gripper.

Caution

The term "operator" refers to anyone responsible for any of the following operations on the xArm Gripper:

- Installation
- Control
- Maintenance
- Inspection
- Calibration
- Programming
- Decommissioning

This documentation explains the various components of the xArm Gripper and general operations regarding the whole life-cycle of the product from installation to operation and decommissioning.

The drawings and photos in this documentation are representative examples and differences may exist between them and the delivered product.

1.3.1. Warning

Caution

Any use of the Gripper in noncompliance of these warnings is inappropriate and may cause injury or damage.

Warning

- The Gripper needs to be properly secured before operating the robot.
- Do not install or operate a Gripper that is damaged or lacking parts.
- Never supply the Gripper with an alternative current (AC) source.
- Make sure all cord sets are always secured at both ends,Gripper end & Robot end
- Always satisfy the recommended keying for electrical connections.
- Be sure no one is in the robot and/or gripper path before initializing the robot's routine.
- Always satisfy the gripper payload.
- Set the gripper speed accordingly, based on your application.
- Keep fingers and clothes away from the gripper while the power is on.
- Do not use the gripper on people or animals.

1.3.2. Risk Assessment and Final Application

The xArm Gripper is meant to be used on an industrial robot. The robot, gripper and any other equipment used in the final application must be

evaluated with a risk assessment. The robot integrator must ensure that all local safety measures and regulations are respected. Depending on the application, there may be risks that need additional protection/safety measures, for example, the work-piece the gripper is manipulating may be inherently dangerous to the operator.

1.3.3. Intended Use

The gripper is designed for grasping and temporarily securing or holding objects.

Caution

The Gripper is NOT intended for applying force against objects or surfaces.

The product is intended for installation on a robot or other automated machinery and equipment.

Info

Always comply with local and/or national laws, regulations and directives on automation safety and general machine safety.

The unit may be used only within the range of its technical data. Any other use of the product is deemed improper and unintended use.

UFACTORY will not be liable for any damages resulting from any improper or unintended use.

2. Installation

The following subsections will guide you through the installation and general setup of xArm Gripper.

(1) The Scope of Delivery section

(2) The Mechanical Installation section

(3) The Electrical Setup section

Warning

Before installing:

Read and understand the safety instructions related to the xArm Gripper.

Verify your package according to the Scope of delivery and your order info.

Have the required parts, equipment and tools listed in the requirements readily available.

Installing:

Satisfy the environmental conditions.

Do not operate the Gripper, or even turn on the power supply, before it is firmly anchored and the danger zone is cleared.

Caution the fingers of the gripper which may move and cause injury or damage.

2.1. Scope of Delivery

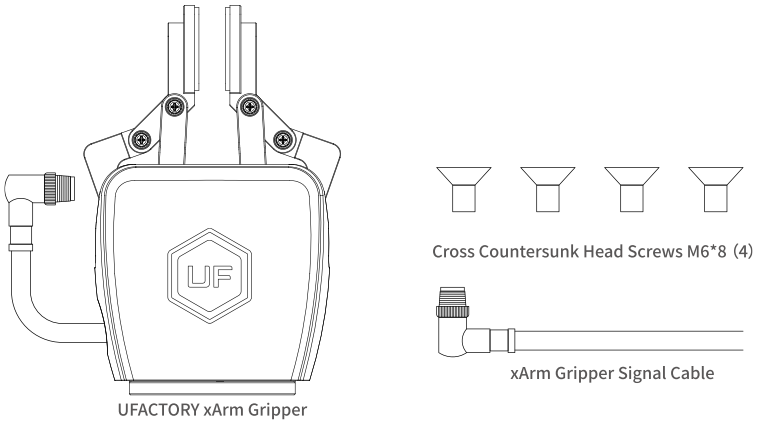
2.1.1. General Kit

A Gripper Kit generally includes these items:

xArm Gripper

xArm Gripper signal cable

Cross countersunk head screws M6*8 (4)



2.2. Mechanical Installation

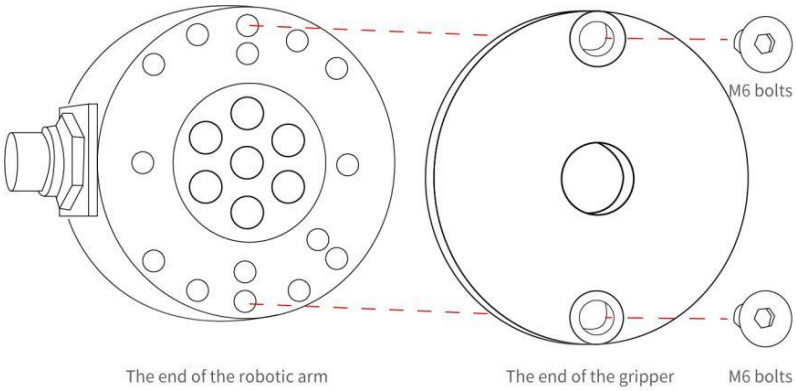
xArm Gripper installation steps (as shown below):

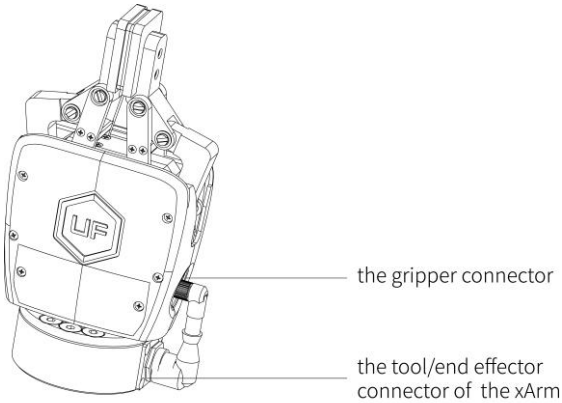
1. Move the robotic arm to a safe position. Avoid touching the robotic arm mounting surface or other equipment;

2. Power off the robotic arm by pressing the emergency stop button on the control box;

3. Fix the gripper on the end of the robotic arm with 2 M6 bolts;

4. Connect the robotic arm and the gripper with the gripper connection cable;





Note:

1. When wiring the gripper connection cable, be sure to power off the robotic arm, the emergency stop button is in the pressed state and the power indicator of the robotic arm is off, so as to avoid robotic arm failure caused by hot plugging;

2. Due to the limitation of the length of the gripper connection cable, the gripper connector and the tool/end effector connector must be on the same side;

3. When connecting the gripper and the robotic arm, be sure to align the positioning holes at the ends of the gripper and the robotic arm. Since the male

pins of the gripper connection cable are relatively thin, avoid bending the male pins during disassembly.

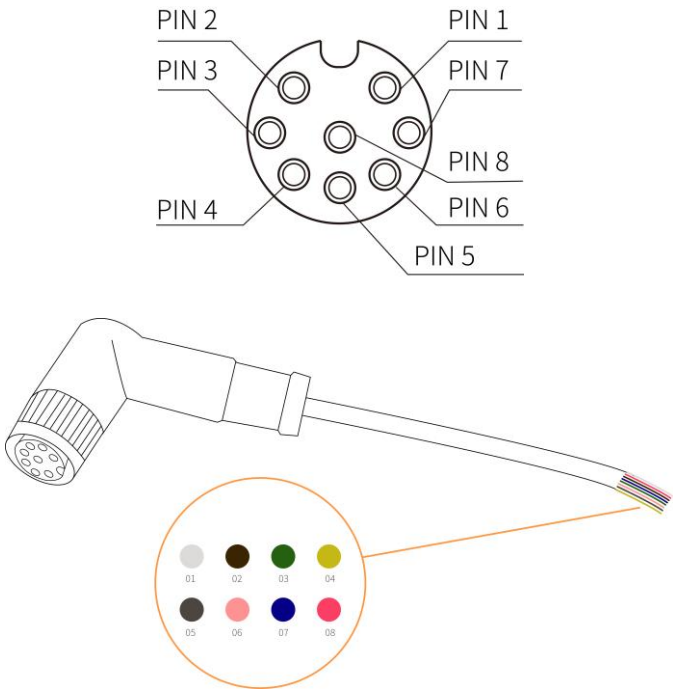
2.3. Electrical Setup

Power and communication are established with the xArm Gripper via a single gripper connection cable. The gripper connection cable provides a 24V power supply to the Gripper and enables serial RS485 communication to the control box.

Warning

Power must be off before connecting the Gripper and the robotic arm via the gripper connection cable.

2.3.1. Pinout Interface



There are 8 pins inside the cable with different colors, each color represents different functions, please refer to the following table:

Line sequence	Color	Signal
1	White	24V
2	Brown	24V
3	Green	GND
4	Yellow	GND
5	Gray	485-A
6	Powder	485-B
7	Blue	IN0(Digital input)
8	Red	IN1(Digital input)

Note:

1. For details of the tool/end effector connector of the robotic arm, please refer to the xArm user manual: https://www.ufactory.cc/pages/download-xarm

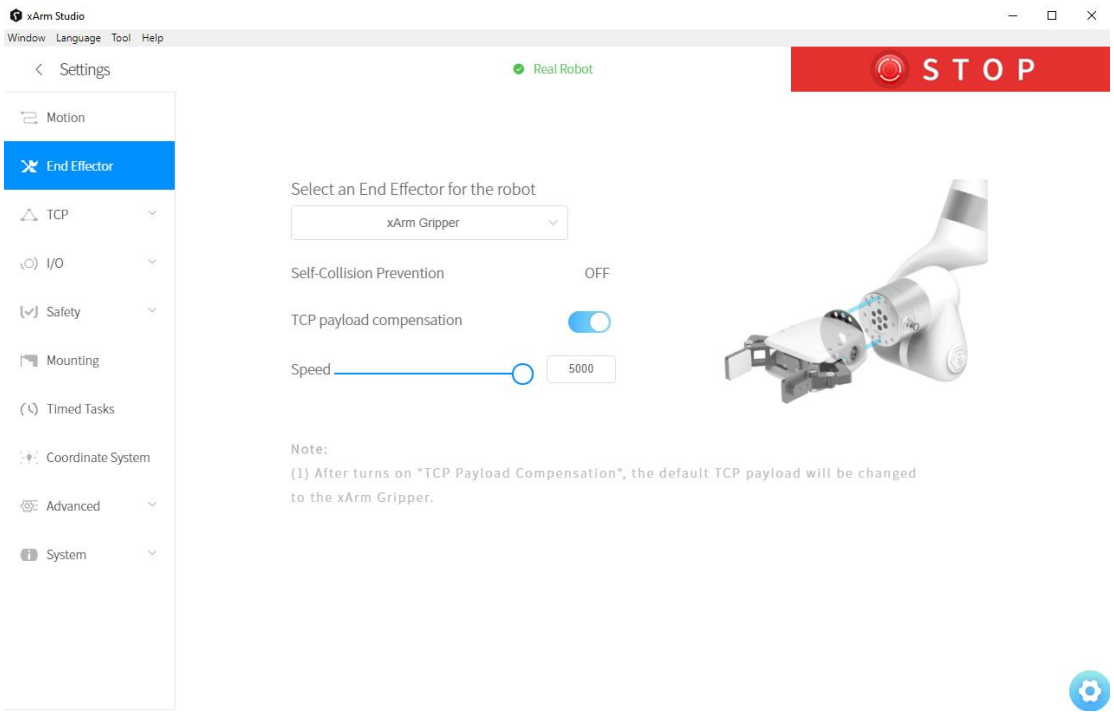
3. Control

3.1. Use xArm Studio to Control xArm Gripper

1. Set up xArm Gripper

- Enter [Settings]-[End Effector]

Select the end effector: xArm Gripper



1. The opening and closing speed of the gripper can be adjusted.

2. The self-collision prevention model of the gripper can be turned on by clicking the button.

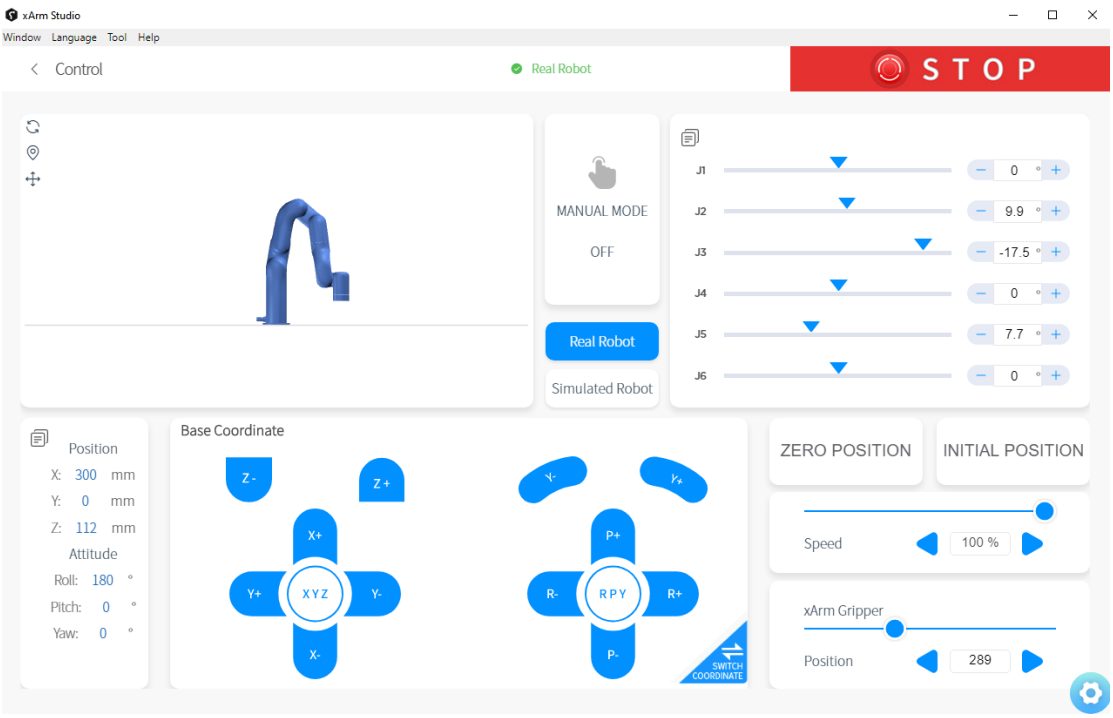
3. When "TCP payload compensation" is turned on, the default TCP payload will be changed to the TCP payload parameter of the gripper.

2. Control xArm Gripper

- Control the xArm gripper in the live control

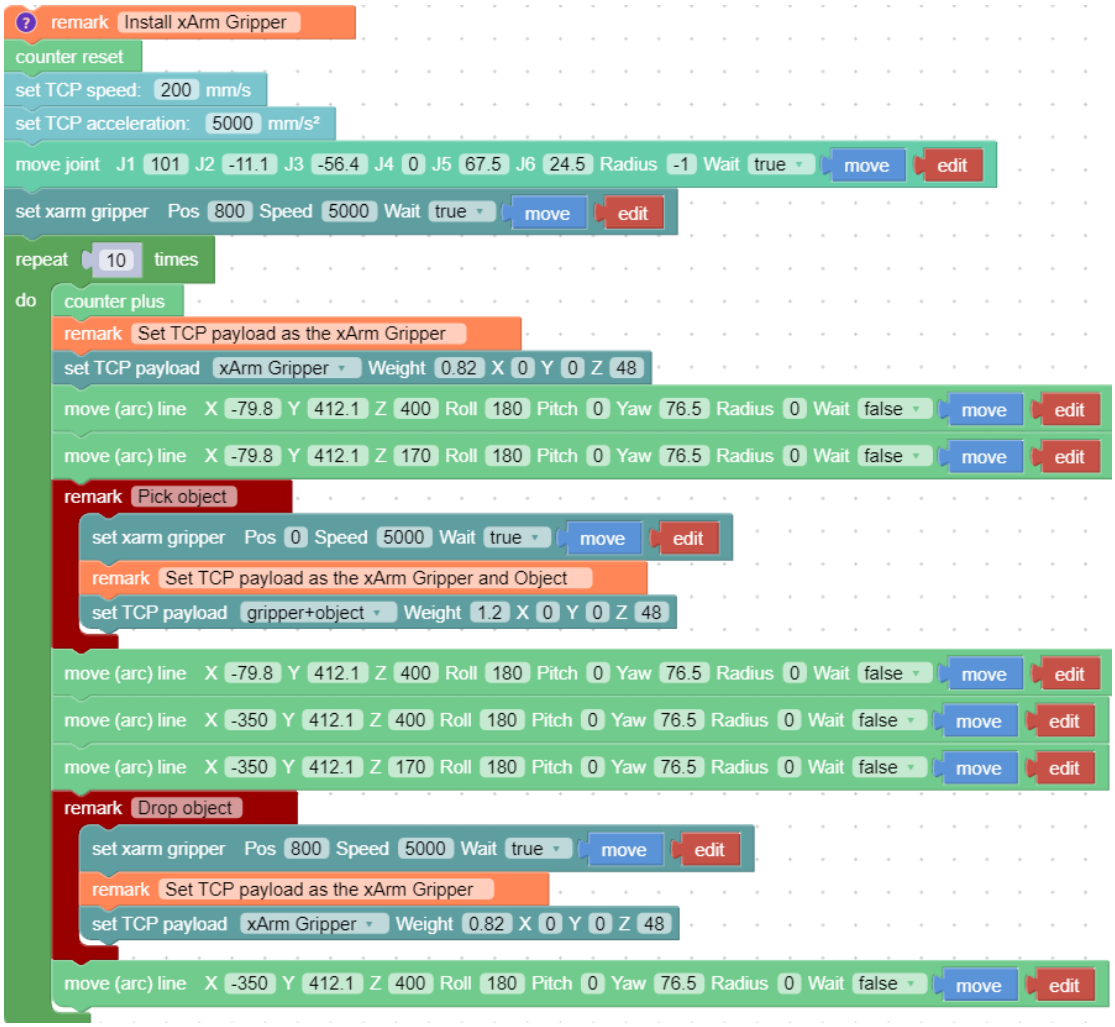
Control Method:

1) By dragging this progress bar, you can control the opening and closing stroke of the gripper.



- Control the xArm gripper through Blockly

xArm Gripper.Blockly



The role of this program: execute this program to control the gripper to pick the target object at the specified position, and then place the target object at the target position.

Note:

1) When the gripper is installed on the robotic arm, the TCP Payload of the gripper should be set in the Blockly program. When the total weight

of the gripper changes after the object is picked, a new TCP Payload needs to be set.

3.2. Use Python-SDK to Control xArm Gripper

For details on controlling Gripper with python-SDK, please refer to the link below:

https://github.com/xArm-Developer/xArm-Python-SDK/blob/master/example/wrapper/common/5004-set_gripper.py

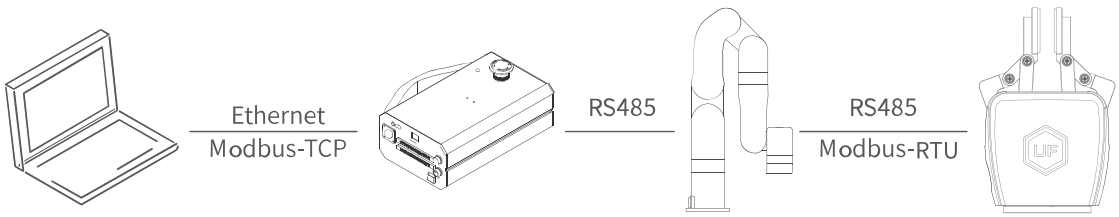
3.3. Use ROS-SDK to Control xArm Gripper

Please refer to Section 5.7.7 in the ReadMe file attached to the ROS package to control the gripper.

xArm ROS-SDK link :

https://github.com/xArm-Developer/xarm_ros

3.4. Use Modbus-TCP Communication Protocol to Control xArm Gripper



This section mainly explains how to control the xArm Gripper by using the Modbus-TCP protocol through xArm control box.

3.4.1. Modbus-TCP Communication Format

Modbus-TCP:

Modbus protocol is an application layer message transmission protocol, including three message types: ASCII, RTU, and TCP. The standard Modbus protocol physical layer interface includes RS232, RS422, RS485 and Ethernet interfaces, and adopts master / slave communication.

Modbus TCP Communication Process:

1. Establish a TCP connection

2. Prepare Modbus messages

3. Use the send command to send a message

4. Waiting for a response under the same connection

5. Use the recv command to read the message and complete a data exchange

6. When the communication task ends, close the TCP connection

Parameter:

Default TCP Port: 502 Protocol: 0x00 0x02

On the problem of users using communication protocols to organize data in big endian and little endian:

In this article, data analysis is big-endian analysis.

3.4.2. Read xArm Gripper Register

3.4.2.1. Register Function

Read Register			
Request			
MBTP Header	Transaction Identifier	2 Bytes	0x0001
	Protocol Identifier	2 Bytes	0x0002
	Length	2 Bytes	6+**N***x2
	Unit Identifier	1 Byte	0x7C
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x03
	Register Starting Address	2 Bytes	**Address**
	Quantity of Registers	**N***x2 Bytes	**N***
Response			
MBTP Header	Transaction Identifier	2 Bytes	0x0001
	Protocol Identifier	2 Bytes	0x0002
	Length	2 Bytes	6+**N***x2
	Unit Identifier	1 Byte	0x7C
	Status Value	1 Byte	0x00
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x03
	Byte Count	1 Byte	**N*x2**
	Registers Value	**N***x2 Bytes	**Value**

N* = Quantity of Registers

Address = Register Starting Address

Resgister:

	Resgister Starting Address	Registers Value	
Get Gripper status Register	0x0000	2 Bytes	**Stop status**: 0x0000 **Motion status:** 0x0001 **Clipping status:** 0x0010
Get Gripper position Register	0x0702	4bytes	0xFFFFFFFB-0x00000320
Get Gripper Error Register	0x000F	2 Bytes	**An error occurs:** all other return values indicate an error(except 0) **No error occurred:** 0x0000

3.4.2.2. Example

1. Get the xArm Gripper status

Get the xArm Gripper status			
Request			
MBTP Header	Transaction Identifier	2 Bytes	0x00，0x01
	Protocol Identifier	2 Bytes	0x00，0x02
	Length	2 Bytes	0x00，0x08
	Unit Identifier	1 Byte	0x7C
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x03
	Register Starting Address	2 Bytes	0x00，0x00
	Quantity of Registers	2 Bytes	0x00，0x01
Response			
MBTP Header	Transaction Identifier	2 Bytes	0x00，0x01
	Protocol Identifier	2 Bytes	0x00，0x02
	Length	2 Bytes	0x00，0x08
	Unit Identifier	1 Byte	0x7C

	Status Value	1 Byte	0x00
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x03
	Byte Count	1 Byte	0x02
	Registers Value (Robotic arm is in motion status)	2 Bytes	0x00，0x01

2. Get the xArm Gripper position

Get the xArm Gripper position			
Request			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x08
	Unit Identifier	1 Byte	0x7C
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x03
	Register Starting Address	2 Bytes	0x07,0x02
	Quantity of Registers	2 Bytes	0x00,0x02
Response			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x0A
	Unit Identifier	1 Byte	0x7C
	Status Value	1 Byte	0x00
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x03
	Byte Count	1 Byte	0x04
	Registers Value (position: 400)	2 Bytes	0x00,0x00,0xC8,0x43

3. Get the xArm Gripper Error

Get the xArm Gripper Error			
Request			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x08
	Unit Identifier	1 Byte	0x7C
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x03
	Register Starting Address	2 Bytes	0x00,0x0F
	Quantity of Registers	2 Bytes	0x00,0x01
Response			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x08
	Unit Identifier	1 Byte	0x7C
	Status Value	1 Byte	0x00
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x03
	Byte Count	1 Byte	0x02
	Registers Value (No error occurred in the Gripper)	2 Bytes	0x00,0x00

3.4.3. Write xArm Gripper Register

3.4.3.1. Register Function

Write Register			
Request			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	**9+N*x2**
	Unit Identifier	1 Byte	0x7C

Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	**Address**
	Quantity of Registers	2 Bytes	**N***
	Byte Count	1 Byte	**N*x2**
	Registers Value	**N***x2 Bytes	**Value**
Response			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x09
	Unit Identifier	1 Byte	0x7C
	Status Value	1 Byte	0x00
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	**Address**
	Quantity of Registers	2 Bytes	**N***

N* = Quantity of Registers

Address = Register Starting Address

Resgister:

	Resgister Starting Address	**Registers Value**	
Set Gripper Mode Register	0x0101	2bytes	**Position mode:** 0x0000
Enable/Disable Gripper Register	0x0100	2 Bytes	**Enable :** 0x0001 **Disable :** 0x0000
Set Gripper Position Register	0x0700	4 Bytes	**Open the Gripper :** 0x0000 0x0082 **Close the Gripper :** 0x0000 0x0032
Set Position Speed Register	0x0303	2 Bytes	0x0100-0x0400 **Unit :** r/min

3.4.3.2. Example

1. Set xArm Gripper Mode

Set xArm Gripper Mode			
Request			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x0B
	Unit Identifier	1 Byte	0x7C
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01,0x01
	Quantity of Registers	2 Bytes	0x00,0x01
	Byte Count	1 Byte	0x02
	Registers Value (Position mode)	2 Bytes	0x00,0x00
Response			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x09
	Unit Identifier	1 Byte	0x7C
	Status Value	1 Byte	0x00
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01,0x01
	Quantity of Registers	2 Bytes	0x00,0x01

2. Enable/Disable xArm Gripper

Enable/Disable xArm Gripper			
Request			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x0B

	Unit Identifier	1 Byte	0x7C
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01,0x00
	Quantity of Registers	2 Bytes	0x00,0x01
	Byte Count	1 Byte	0x02
	Registers Value	2 Bytes	0x00,0x01
Response			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x09
	Unit Identifier	1 Byte	0x7C
	Status Value	1 Byte	0x00
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01,0x00
	Quantity of Registers	2 Bytes	0x00,0x01

3. Set xArm Gripper Speed

Set xArm Gripper Speed			
Request			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x0B
	Unit Identifier	1 Byte	0x7C
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x03,0x03
	Quantity of Registers	2 Bytes	0x00,0x01
	Byte Count	1 Byte	0x02
	Registers Value(1500r/min)	2 Bytes	0x05,0xDC
Response			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01

	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x09
	Unit Identifier	1 Byte	0x7C
	Status Value	1 Byte	0x00
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x03,0x03
	Quantity of Registers	2 Bytes	0x00,0x01

4. Set xArm Gripper Position

Set xArm Gripper Position			
Request			
MBTP Header	Transaction Identifier	2 Bytes	0x00，0x01
	Protocol Identifier	2 Bytes	0x00，0x02
	Length	2 Bytes	0x00，0x0D
	Unit Identifier	1 Byte	0x7C
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x07,0x00
	Quantity of Registers	2 Bytes	0x00,0x02
	Byte Count	1 Byte	0x04
	Registers Value (Open the BIO Gripper)	4 Bytes	0x00,0x00,0xC8,0x43
Response			
MBTP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol Identifier	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x09
	Unit Identifier	1 Byte	0x7C
	Status Value	1 Byte	0x00
Internal Use	Internal Use	1 Byte	0x09
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x07,0x00

	Quantity of Registers	2 Bytes	0x00,0x02

3.4.4. xArm Gripper Control Process

The complete process of controlling the motion of the xArm Gripper is as follows:

(1) Enable the Gripper

0x00, 0x01, 0x00, 0x02, 0x00, 0x0B, 0x7C, 0x09, 0x08, 0x10, 0x01, 0x00, 0x00, 0x01, 0x02, 0x00, 0x01

(2) Open the Gripper

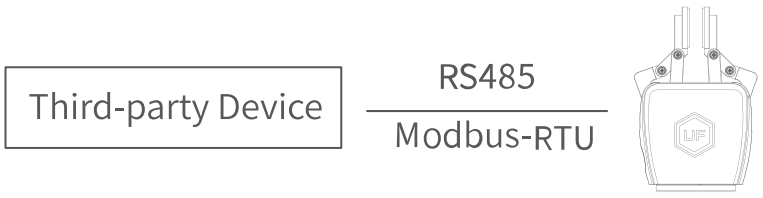
0x00, 0x01, 0x00, 0x02, 0x00, 0x0D, 0x7C, 0x09, 0x08, 0x10, 0x07, 0x00, 0x00, 0x02, 0x04, 0x00, 0x00, 0x00, 0x82

(3) Close the Gripper

0x00, 0x01, 0x00, 0x02, 0x00, 0x0D, 0x7C, 0x09, 0x08, 0x10, 0x07, 0x00, 0x00, 0x02, 0x04, 0x00, 0x00, 0x00, 0x32

3.5. Use Modbus-RTU Communication Protocol to Control xArm Gripper

3.5.1. Modbus RTU Communication Format



The gripper defaults to the standard Modbus RTU protocol at a default

baud rate is 2Mbps and the slave ID is 0x08. The currently supported function codes are: 0x03 / 0x10. In this article, data analysis is big-endian analysis.

Read Register			
Request			
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x03
	Register Starting Address	2 Bytes	**Address**
	Quantity of Register	2 Bytes	**N***
	Modbus CRC16	2 Bytes	**CRC***
Response			
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x03
	Byte Count	1 Byte	**N*x2**
	Registers Value	**N***x2 Bytes	**Value**
	Modbus CRC16	2 Bytes	**CRC***

3.5.2. Read xArm Gripper Register

N* = Quantity of Registers

Address = Register Starting Address

CRC* = Cyclic Redundancy Check

Resgister:

	Resgister Starting Address	Register Value	
Get Gripper status Register	0x0000	2 Bytes	**Stop status**: 0x0000 **Motion status:** 0x0001 **Clipping status:** 0x0010

Get Gripper position Register	0x0702	4bytes	0xFFFFFFFB-0x00000320
Get Gripper Error Register	0x000F	2 Bytes	**An error occurs:** all other return values indicate an error(except 0) **No error occurred:** 0x0000

3.5.3. Write xArm Gripper Register

Write Register			
Request			
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	**Address**
	Quantity of Register	2 Bytes	**N***
	Byte Count	1 Byte	**N*x2**
	Registers Value	**N***x2 Bytes	**Value**
	Modbus CRC16	2 Bytes	**CRC***
Response			
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	**Address**
	Quantity of Registers	2 Bytes	**N***
	Modbus CRC16	2 Bytes	**CRC***

N* = Quantity of Registers

Address = Register Starting Address

CRC* = Cyclic Redundancy Check

Resgister:

	Resgister Starting Address	**Register Value**	
Enable/Disable Gripper	0x0100	2 Bytes	**Enable :** 0x0001 **Disable :** 0x0000
Set Gripper Position Register	0x0700	4 Bytes	**Open the Gripper :** 0x0000 0x0082

			Close the Gripper : 0x0000 0x0032
Set Position Speed Register	0x0303	2 Bytes	0x0100-0x0400 **Unit :** r/min
Set Gripper Mode Register	0x0101	2bytes	**Position mode:** 0x0000

3.5.4. Modbus RTU Example

This section demonstrates the example given in the Control Logic section when programmed using the Modbus RTU protocol.

Step1：Set xArm Gripper Mode

Set xArm Gripper Mode			
Request			
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01,0x01
	Quantity of Registers	2 Bytes	0x00,0x01
	Byte Count	1 Byte	0x02
	Registers Value	2 Bytes	0x00,0x00
	Modbus CRC16	2 Bytes	0xDD,0x11
Response			
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01,0x01
	Quantity of Registers	2 Bytes	0x00,0x01
	Modbus CRC16	2 Bytes	0x51,0x6C

Step2：Enable xArm Gripper

Enable xArm Gripper
Request

Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01,0x00
	Quantity of Registers	2 Bytes	0x00,0x01
	Byte Count	1 Byte	0x02
	Registers Value	2 Bytes	0x00,0x01
	Modbus CRC16	2 Bytes	0x1D,0x00
Response			
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01,0x00
	Quantity of Registers	2 Bytes	0x00,0x01
	Modbus CRC16	2 Bytes	0x00,0xAC

Step3：Set xArm Gripper Speed

Set xArm Gripper Speed			
Request			
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x03,0x03
	Quantity of Registers	2 Bytes	0x00,0x01
	Byte Count	1 Byte	0x02
	Registers Value(1500r/min)	2 Bytes	0x05,0xDC
	Modbus CRC16	2 Bytes	0xFD,0xFA
Response			
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x03,0x03
	Quantity of Registers	2 Bytes	0x00,0x01
	Modbus CRC16	2 Bytes	0xF1,0x14

Step4：Set xArm Gripper Position

Set xArm Gripper Position			
Request			
Modbus RTU Data	Slave ID (Gripper)	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x07,0x00

	Quantity of Registers	2 Bytes	0x00,0x02
	Byte Count	1 Byte	0x04
	Registers Value (position：400)	4 Bytes	0x00,0x00,0x00,0x82
	Modbus CRC16	2 Bytes	0x7B,0x62
Response			
Modbus RTU Data	Slave ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x07,0x00
	Quantity of Registers	2 Bytes	0x00,0x02
	Modbus CRC16	2 Bytes	0x40,0x25

4. Gripper Alarm Code & General Response

The user can re-power on the robotic arm as an error handling, the steps are as follows (all the following steps are needed):

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

2. Enable the robotic arm.

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

b. xArm-Python-SDK enable method: Refer to Error Handling Method.

c.xArm_ros: users can view related documents at:

https://github.com/xArm-Developer/xarm_ros

3. Re-enable the gripper.

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

Software Error	Error Handling
G9	Gripper Current Detection Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
G11	Gripper Current Overlimit Please click "OK" to re-enable the Gripper.
G12	Gripper Speed Overlimit Please click "OK" to re-enable the Gripper.
G14	Gripper Position Command Overlimit Please click "OK" to re-enable the Gripper.
G15	Gripper EEPROM Read and Write Error Please click "OK" to re-enable the Gripper.
G20	Gripper Driver IC Hardware Error Please click "OK" to re-enable the Gripper.
G21	Gripper Driver IC Initialization Error Please click "OK" to re-enable the Gripper.
G23	Gripper Large Motor Position Deviation Please check if the movement of the Gripper is blocked, if not, please click "OK" to re-enable the Gripper.
G25	Gripper Command Over Software Limit Please check if the gripper command is set beyond the software limit.
G26	Gripper Feedback Position Software Limit Please contact technical support.
G33	Gripper Drive Overloaded Please contact technical support.
G34	Gripper Motor Overload Please contact technical support.
G36	Gripper Driver Type Error Please click "OK" to re-enable the Gripper.
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.

Appendix:

xArm-Python-SDK alarm processing method:

When designing the robotic arm motion path with the Python library, if the robot fails, you need to manually clear the errors. After clearing the error, you still need to re-enable the robotic arm and set the robot to motion mode for the robot to move normally. Then the path planning of the robotic arm should be re-adjusted 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)

5. xArm Gripper Technical Specifications

Gripper	
Nominal Supply Voltage	24V DC
Absolute Maximum Supply Voltage	28V DC
Quiescent Power (Minimum Power Consumption)	1.5W
Peak Current	1.5A
Working Range	0- 84mm
Maximum Clamping Force	30N
Weight	802g
Communication Mode	RS-485
Communication Protocol	Modbus RTU
Programmable Gripping Specification	Position, Speed
Feedback	Position

6. After-sales Service

1. After-sales policy:

For the detailed after-sales policy of the product, see the official website:

https://store-ufactory-cc.myshopify.com/pages/warranty-returns

2. The general process of after-sales service is:

(1) Contact UFACTORY technical support (support@ufactory.cc) to confirm whether the product needs to repair and which part should be sent back to UFACTORY.

(2) After the bill of lading on UPS, we will send the invoice and label to you by mail. You need to make an appointment with the local UPS and then send the product to us.

(3) UFACTORY will check the product warranty status according to the after-sales policy.

(4) Generally, the process takes around 1-2 weeks except for shipment.

Note:

1. Please keep the original packaging materials of the product. When you need to send the product back to get repaired, please pack the product with the original box to protect the product during the transportation.