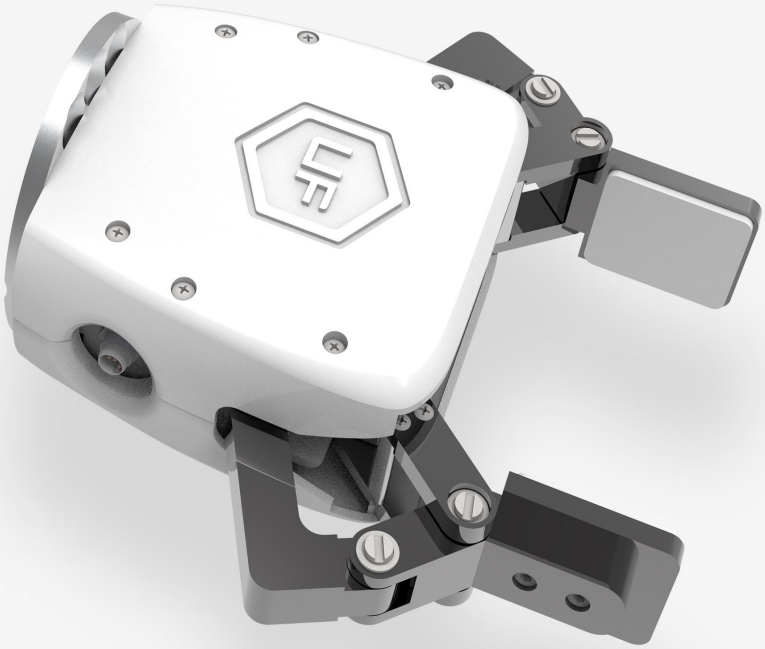
UFACTORY

XARM GRIPPER

USER MANUAL

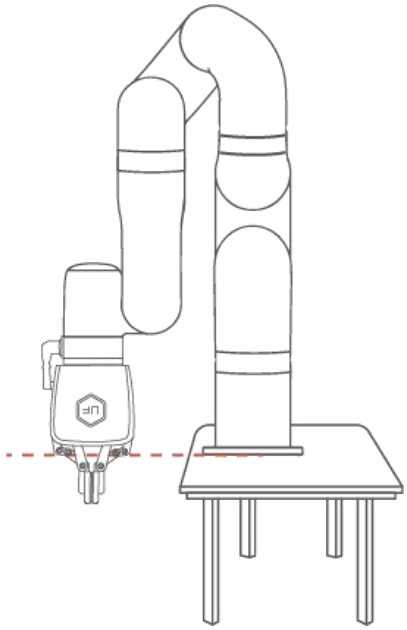


SHENZHEN UFACTORY CO., LTD

V.1.4.0

Table

1. Precautions for Using Gripper



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

1. When the robotic arm is in the zero position, the gripper will exceed the installation surface. Please adjust the robotic arm to a posture suitable for installing the gripper during installation.

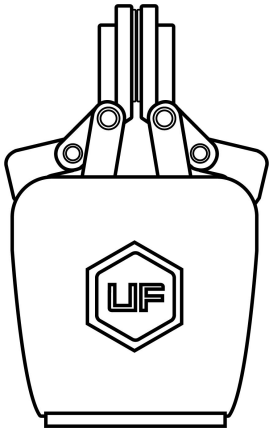
2.When a robotic arm equipped with a gripper is used for trajectory planning, it is necessary to perform a safety assessment on whether to return to the zero point or whether the operation can be performed and to avoid collision.

2. Installation of Gripper

2.1 Gripper Movement Process

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

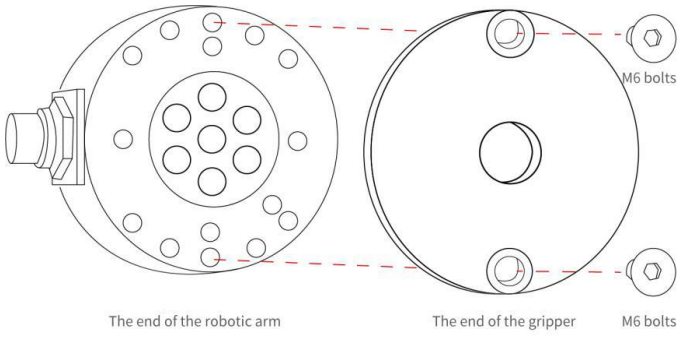
The value range of the gripper opening and closing is: -10 to 850. The larger the value, the greater the stroke of the gripper, meaning the smaller the value, the smaller the stroke of the gripper. If the clamping is not tight, a negative value can be sent until it is tightened.



2.2 Gripper Installation

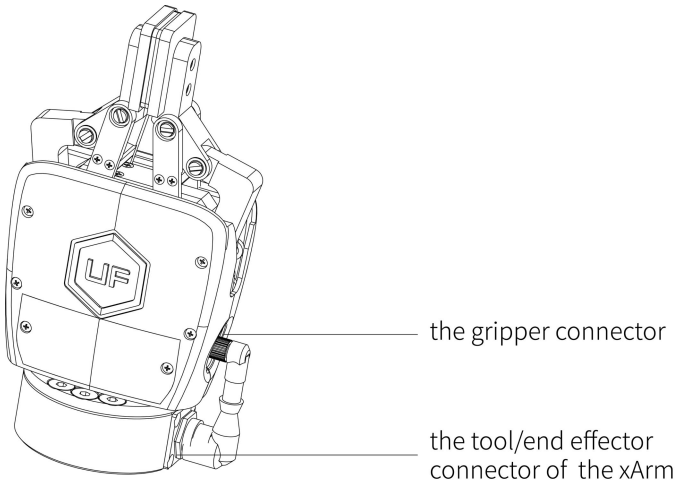
Installation of gripper:

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;



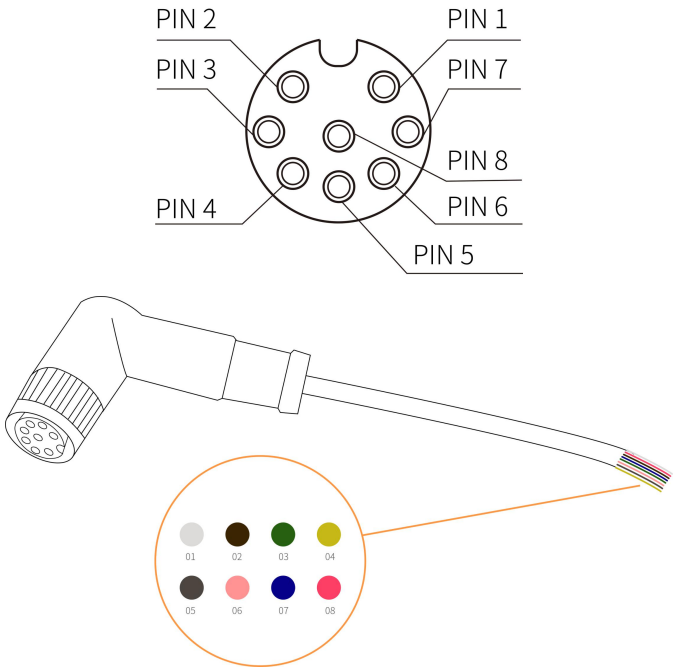
Note:

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.



3. The Gripper Connector

The gripper connector:



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)

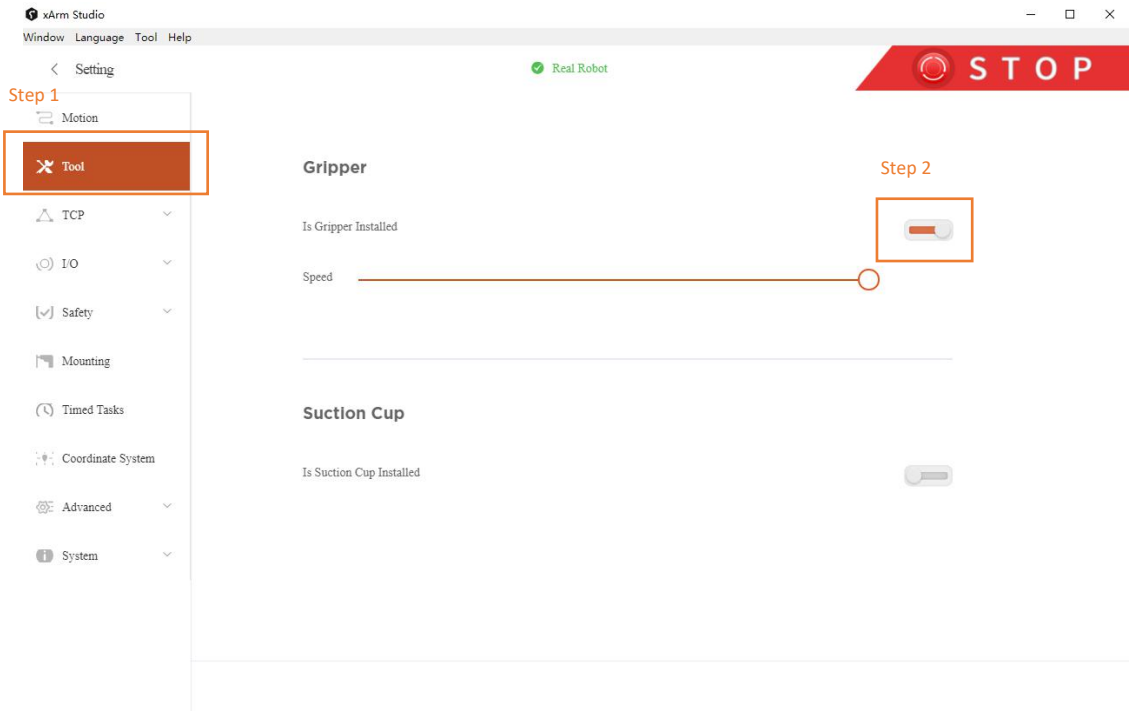
Note:

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

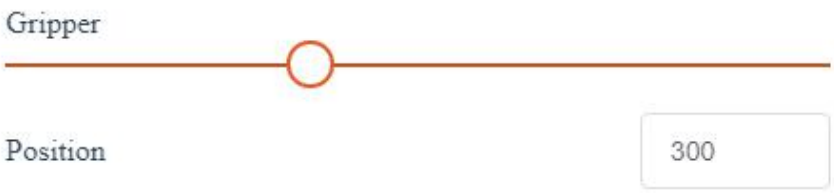
4. Control the Gripper with xArm Studio

4.1 Gripper Operation Steps

1. Open xArm Studio, select [Setting]-[Tool] to install the gripper and set the closing speed of the gripper;

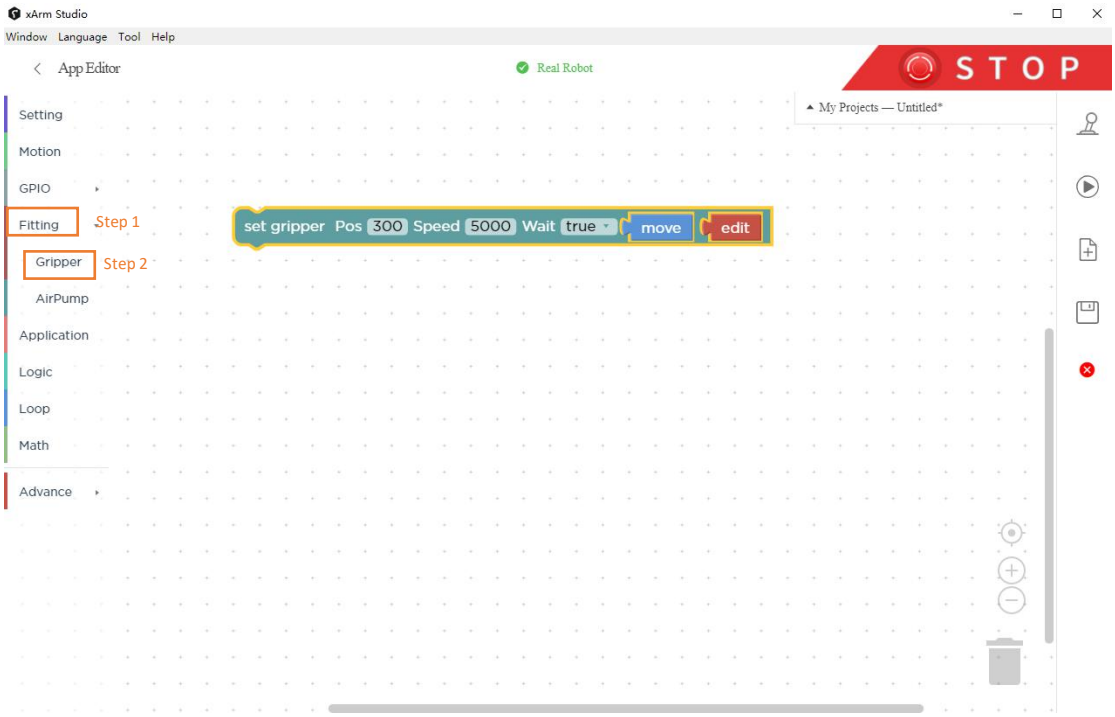


2. Enter [Live Control], you can see the drag bar that controls the stroke of the gripper in the lower right corner. The range of gripper stroke is -10 ~ 850;



4.2 Use of Gripper in Blockly

Enter [Blockly] programming, drag the gripper command to the work area in the code block [Fittings]-[Gripper] to set the position and speed of the gripper and click [move] to expand the gripper.



Typical Example

【sleep ( ) s】: After receiving this command, the robotic arm will stop moving for the set time and then continue to execute the following commands. It is mainly used in motion programs that need require continuous motion and is used to buffer more motion commands before the continuous motion commands.

【move(arc) line X() Y()Z() Roll() Pitch() Yaw() Radius() Wait(true/false) [move] [edit]】:

Indicates the cartesian coordinate value of the linear motion and the TCP rotation angle in mm and °.

【set gripper Pos () Speed () Wait (true / false) [move] [edit] : Set the position and the opening and closing speed of the gripper.

Explanation:

Step 1: Move the robotic arm to position 1: (519, 106.3, 617.2, 177.2, 0.2, 3)

Step 2: After reaching position 1, the gripper moves at a speed of 5000r / min to position of 680

Step 3: Move the robot arm to position 2: (519, 106.3, 454.8, 177.2, 0.2, 3)

Step 4: Robotic arm stays at position 2 for 2 seconds

Step 5: The gripper is in position 2 and moves to the position with displacement of 549 at speed of 5000r / min.

5. Control Gripper with xArm-Python-SDK

5.1 The Flow of Gripper Movement

1. Enable the gripper.

   E. g: arm.set_gripper_enable（True）

2. Set the mode of the gripper.

   E.g: arm.set_gripper_mode（0） 0 - location mode

3. Send position command to the gripper.

   E.g: arm.set_gripper_position（600，wait=True）

If the gripper reports an error, the gripper error message must be cleared before the gripper can work normally again.

5.2 Gripper API

- **set_gripper_enable**

set_gripper_enable	
Description	Enable / Disable the gripper
Parameter	enable： enable=True enable the gripper enable=False disable the gripper
Return Value	normal return value: 0 (successful) if there is an error return value, see the appendix for details: xArm-Python-SDK API code description
Example	arm.set_gripper_enable（True） arm.set_gripper_enable（False）

- **set_gripper_mode**

set_gripper_mode		
Description	set the gripper mode	
Parameter	0	location mode
Return Value	normal return value: 0 (successful) if there is an error return value, see the appendix for details: xArm-Python-SDK API code description	
Example	arm.set_gripper_mode（0）	

- **set_gripper_position**

set_gripper_position		
Description	set the gripper position	
Parameter	pose	the position of the gripper
	speed	set the moving speed of the gripper, unit: r / min
	wait	if wait = True, wait for the current commands to be sended before sending the next commands; if wait = False, send the next commands directly;
	timeout	wait time, default is 10 seconds; unit: s (seconds)
Return Value	normal return value: 0 (successful) if there is an error return value, see the appendix for details: xArm-Python-SDK API code description	
Example	arm.set_gripper_position（600）	

- **set_gripper_speed**

set_gripper_speed		
Description	set the moving speed of the gripper	
Parameter	speed	set the moving speed of the gripper, unit: r / min
Return Value	normal return value: 0 (successful) if there is an error return value, see the appendix for details: xArm-Python-SDK API code	

	description
Example	arm.set_gripper_speed（3000）

- **get_gripper_position**

get_gripper_position	
Description	get the gripper position
Parameter	None
Return Value	normal return value: 0 (successful) if there is an error return value, see the appendix for details: xArm-Python-SDK API code description
Example	arm.get_gripper_position（）

- **get_gripper_err_code**

get_gripper_err_code	
Description	get the gripper error code
Parameter	None
Return Value	normal return value: 0 (successful) if there is an error return value, see the appendix for details: xArm-Python-SDK API code description
Example	arm.get_gripper_err_code（）

- **clean_gripper_error**

clean_gripper_error	
Description	clean the gripper error
Parameter	None
Return Value	normal return value: 0 (successful) if there is an error return value, see the appendix for details: xArmSDK API code description
Example	arm.clean_gripper_error（）

6. Gripper Communication Protocol

6.1 Modbus RTU Communication Format Description

The gripper defaults to the standard Modbus RTU protocol at a default baud rate is 2Mbps and the address is 8. The currently supported function codes are: 0x03 / 0x06 / 0x10.

Protocol format

● **The function code is: 0x03**

Request command format:

Format	Slave address	Function code	Starting address	Number of registers	CRC
Length	1 byte	1 byte	2 bytes	2 bytes	2 bytes
Example (Read alarm code)	0x08	0x03	0x00 0x0F	0x00 0x01	0xB4 0x90

Response command format:

Format	Slave address	Function code	Byte count	Register value	CRC
Length	1 byte	1 byte	1 byte	2 bytes	2 bytes
Example (Read alarm code)	0x08	0x03	0x02	0x00 0x00	0x64 0x45

Note:

N—the value corresponding to the number of registers.

The value of bytes corresponds to 2xN.

Register corresponding length is Nx2.

● **The function code is: 0x06**

Request command format:

Format	Slave address	Function code	Register value	Date	CRC
Length	1 byte	1 byte	2 bytes	2 bytes	2 bytes
Example (Enable the gripper)	0x08	0x06	0x01 0x00	0x00 0x01	0x49 0x6F

Response command format

Format	Slave address	Function code	Register value	Date	CRC
Length	1 byte	1 byte	2 bytes	2 bytes	2 bytes
Example (Enable the gripper)	0x08	0x06	0x01 0x00	0x00 0x01	0x49 0x6F

● **The function code is: 0x10**

Request command format:

Format	Slave address	Function code	Starting address	Number of registers	Byte count	Register value	CRC
Length	1 byte	1 byte	2 bytes	2 bytes	1byte	4bytes	2 bytes
Example (Gripper position command is 100)	0x08	0x10	0x07 0x00	0x00 0x02	0x04	0x00 0x00 0x00 0x64	0xFA 0xF8

Note:

N—the value corresponding to the number of registers.

The value of byte count corresponds to Nx2.

The corresponding length of the data is 2xN.

Response command format:

Format	Slave address	Function code	Starting address	Number of registers	CRC
Length	1 byte	1 byte	2 bytes	2 bytes	2 bytes
Example (Gripper position command is 100)	0x08	0x10	0x07 0x00	0x00 0x02	0x40 0x25

6.2 Register Description

The following register addresses are the addresses written in the gripper's RAM. If these addresses need to be stored in the EEPROM, the register address and 0x1000 must be OR calculated (or operator: |).

6.2.1 Monitoring Group

Register address	Description
0x0000	status
0x0001	speed (r / min)
0x0002	current percentage
0x0003	current (A)
0x0004	command position (p)
0x0006	motor position (p)
0x0008	position error (p)
0x000F	current alarm code
0x0010	current value when alarm occurs
0x0011	speed value when alarm occurs
0x0012	input voltage value when an alarm occurs

6.2.2 Fn1xx Control Parameters

Number	Name	Setting range	Unit	Factory setting	Effective time	Register address
Fn100	enable the gripper	0-1	-	0	immediately	0x0100
Fn101	control mode	0-2 0：location 1：speed	-	0	immediately	0x0101
Fn109	fault reset	0-1	-	0	immediately	0x0109

6.2.3 Fn2xx Gain Parameters

Number	Name	Setting range	Unit	Factory setting	Effective time	Register address
Fn200	position loop gain	10-20000	0.1Hz	200	immediately	0x0200
Fn201	position loop feedforward	0-1000	0.1%	200	immediately	0x0201
Fn202	position loop feedforward filtering time	0-1000	1ms	5	immediately	0x0202
Fn203	speed loop gain	10-20000	0.1	100	immediately	0x0203
Fn204	speed loop integral	10-10000	0.1	300	immediately	0x0204

6.2.4 Fn3xx Position Parameters

Number	Name	Setting range	Unit	Factory setting	Effective time	Register address
Fn300	Position acceleration time	1-2000	ms	100	power on effective	0x0300
Fn301	position deceleration time	1-2000	ms	100	power on effective	0x0301
Fn302	position smoothing time	1-200	ms	10	power on effective	0x0302
Fn303	position speed	1-20000	r/min	1500	immediately	0x0303
Fn308	position error alarm value	0x00000000 -0xFFFFFFFF	Pulse	0x20000	immediately	0x0308
Fn310	position command alarm value	0x00000000 -0xFFFFFFFF	Pulse	0x20000	immediately	0x030A

6.2.5 Fn4xx Speed Parameters

Number	Name	Setting range	Unit	Factory setting	Effective time	Register address
Fn400	speed command	-20000-20000	r/min	100	immediately	0x0400
Fn403	speed limit	0-20000	r/min	5000	immediately	0x0403

6.2.6 Fn5xx Torque Parameters

Number	Name	Setting range	Unit	Factory setting	Effective time	Register address
Fn505	sarting current limit	5-100	-	16	immediately	0x0505
Fn506	hold current limit	1-100	-	10	immediately	0x0506
Fn507	starting current	100-30000			immediately	0x0507

	operation time		ms	1500		

6.2.7 Fn6xx Communication Parameters

Number	Name	Setting range	Unit	Factory setting	Effective time	Register address
Fn600	slave address	1-255	-	8	effective after power on	0x0600
Fn601	bandrate	0:4800 1:9600 2:19200 3:38400 4:57600 5:115200 6:230400 7:460800 8:921600 9:1M 10:1.5M 11:2M	bps	11	effective after power on	0x0601
Fn602	protocol type	0:Modbus RTU 1:customize		0	immediately	0x0602
Fn609	restore factory value	0-1	-	0	effective after power on	0x0609

6.2.8 Fn7xx Position Command

Number	Name	Setting range	Unit	Factory setting	Effective time	Register address
Fn700	position	0x00000000	pulse	0x0000	immediately	0x0700

	command	-0xFFFFFFFF		0000		
Fn702	position feedback	0x00000000 -0xFFFFFFFF	pulse	0x0000 0000	read-only	0x0702
Fn706	electronic gear ratio numerator	1-30000	-	100	power on effective	0x0706
Fn707	electronic gear ratio denominator	1-30000	-	100	power on effective	0x0707

6.2.9 Fn8xx Motor Parameters

Number	Name	Setting range	Unit	Factory setting	Effective time	Register address
Fn800	hardware version	-	-	10	read-only	0x0800
Fn801	software version	-	-		read-only	0x0801
Fn804	motor ID	0-999	-	100	effective after power on	0x0804
Fn805	rated power	1-2000	W	100	effective after power on	0x0805
Fn806	rated voltage	1-6000	0.01V	2400	effective after power on	0x0806
Fn807	rated current	1-2400	0.01A	800	effective after power on	0x0807

Fn808	maximum current	1-5000	0.01A	1650	effective after power on	0x0808
Fn809	rated speed	1-6000	r/min	3000	effective after power on	0x0809
Fn810	maximum speed	1-12000	r/min	6000	effective after power on	0x080A
Fn811	number of pole pairs	0-20	1.1.10. antipole	7	effective after power on	0x080B
Fn823	gripper calibration	0 - 1	-	0	immediately	0x0817

7. Gripper Alarm Code & General Response

The user can restart the robot arm by following these steps:

1. Re-powering the robotic arm via the emergency stop button on the control box.
2. Enable the robotic arm.
   a. xArm Studio enable mode: Click the guide button of the pop-up window or click the button：【Enable Robot】
   b. xArm-Python-SDK enable mode: Refer to the link below
   xArm-Python-SDK alarm processing method
   c. xArm_Ros library: Users can view related documents at
   https://github.com/xArm-Developer/xarm_ros
3. Re-enable the gripper.

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

Alarm Code	Alarm Description	Alarm Response
0x09	Abnormal current detection	Gripper Current Detection Error Please restart the xArm with the Emergency Stop Button on the xArm Controller.
0x0B	Gripper overcurrent	Gripper Current Overlimit Please click “OK” to re-enable the Gripper.
0x0C	Gripper overspeed	Gripper Speed Overlimit Please click “OK” to re-enable the Gripper.
0x0E	Position command is too large	Gripper Position Command Overlimit Please click “OK” to re-enable the Gripper.
0x0F	EEPROM read and write error	Gripper EEPROM Read and Write Error Please click “OK” to re-enable the Gripper.
0x14	Driver IC hardware alarm	Gripper Driver IC Hardware Error Please click “OK” to re-enable the Gripper.

0x15	Driver IC initialization abnormal	Gripper Driver IC Initialization Error Please click “OK” to re-enable the Gripper.
0x17	Position deviation is too large	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.
0x19	Gripper command overrun	Gripper Command Over Software Limit Please check if the gripper command is set beyond the software limit.
0x1A	Gripper position overrun	Gripper Feedback Position Software Limit Please contact technical support.
0x21	Driver overload	Gripper Drive Overloaded Please contact technical support.
0x22	Motor overload	Gripper Motor Overload Please contact technical support.
0x24	Driver type error	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)

8. 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
Communication Mode	RS-485
Communication Protocol	Modbus RTU
Programmable Parameters	Position, Speed
Programmable Gripping Parameters	Position, Speed
Status Indicator	Error status, Power
Feedback	Current, Position
Working Range	86mm
Maximum Clamping Force	30N
Maximum Clamping Weight	3kg
Weight	822g

9. Appendix

- Click the link below to download the gripper 3D file:

  http://download.ufactory.cc/xarm/tool/xArm%20Gripper-17062019.STEP

- xArm-Python-SDK API code description

API Return Value Status Code

API return value status code	
-9: emergency stop	4: tcp reply length error
-8: out of range	5: tcp reply number error
-7: joint angle limit	6: tcp protocol flag error
-6: cartesian pos limit	7: tcp reply command and send command do not match
-5: revesed	8: send command error, may be network exception
-4: command is not exist	9: reversed
-3: revesed	10: reversed
-2: xArm is not ready, may be the motion is not enable or not set state	11: other error
-1: xArm is disconnect or not connect	12: parameter error
- 0: success	31: trajectory read/write failed
1: there are errors that have not been cleared	32: trajectory read/write timeout
2: there are warnings that have not been cleared	33: playback trajectory timeout
3: get response timeout	41: wait to set suction cup timeout

Controller Warning Code

Controller warning code	
11: uxbux que is full	13: the instruction does not exist
12: parameter error	14: command has no solution

Controller Error Code

Controller error code	
10: Servo motor error	24: Speed Exceeds Limit
11: Servo motor 1 error	25: Planning Error
12: Servo motor 2 error	26: Linux RT Error
13: Servo motor 3 error	27: Command Reply Error
14: Servo motor 4 error	28: End Module Communication Error
15: Servo motor 5 error	29: Other Errors
16: Servo motor 6 error	30: Feedback Speed Exceeds limit
17: Servo motor 7 error	31: Collision Caused Abnormal Current
19: Gripper Communication Error	32: Three-point drawing circle calculation error
21: Kinematic Error	33: Controller GPIO error
22: Collision Error	34: Recording Timeout
23: Joints Angle Exceed Limit	35: Safety Boundary Limit

Servo Error Code

Servo Error Code	
10: Current Detection Error	23: Large Motor Position Deviation
11: Joint Current Overlimit	26: Joint N Positive Overrun
12: Joint Speed Overlimit	27: Joint N Negative Overrun
14: Position Command Overlimit	28: Joint Commands Error
15: Joints Overheat	33: Drive Overloaded
16: Encoder Initialization Error	34: Motor Overload
17: Single Ring Encoder Error	35: Motor Type Error
18: Multi-turn Encoder Error	36: Driver Type Error
19: Low Battery Voltage	39: Joint Voltage Overload
20: Driver IC Hardware Error	40: Joint Voltage Insufficient
21: Driver IC Initialization Error	49: EEPROM Read and Write Error
22: Encoder Configuration Error	52: Motor Angle Initialization Error

Gripper Error Code

Gripper Error Code	
9: Gripper Current Detection Error	23: Gripper Large Motor Position Deviation
11: Gripper Current Overlimit	25: Gripper Command Over Software Limit
12: Gripper Speed Overlimit	26: Gripper Feedback Position Software Limit
14: Gripper Position Command Overlimit	33: Gripper Drive Overloaded
15: Gripper EEPROM Read and Write Error	34: Gripper Motor Overload
20: Gripper Driver IC Hardware Error	36: Gripper Driver Type Error
21: Gripper Driver IC Initialization Error	23: Gripper Large Motor Position Deviation