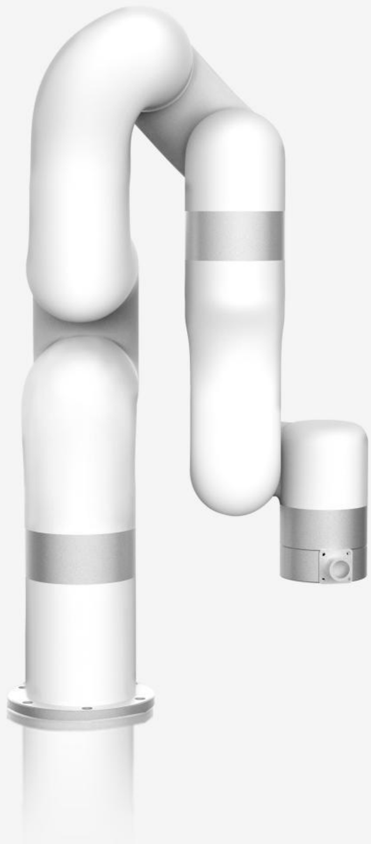
UFACTORY

xARM

DEVELOPER MANUAL



SHENZHEN UFACTORY CO., LTD

V 1.6.0

Contents

1. Introduction

1.1. Notice

(1) This manual is dedicated for developers who develop the applications base on the xArm Modbus-TCP communication protocol. For xArm Studio application development, please refer to "xArm User Manual". For Python (C++ or ROS) application development, please refer to "1.6 Further Developer Resources".

(2) Considering the potential risks of using xArm Modbus-TCP communication protocol for application development, operators need to read and understand all the contents of "xArm User Manual", familiar with xArm risk assessment and robot motion planning, and proficient in robot parameter setting and program creating in "xArm Studio" before Modbus-TCP end developing.

Before meeting the above conditions, we strongly recommend operators should refer to 'xArm User Manual' and program xArm robot by xArm Studio. Until then, operators could start xArm Modbus-TCP application development based on the communication protocol xArm provided.

It will reduce the potential risks as well as increase the efficiency of your application development based on xArm Modbus-TCP.

1.2. Main Contents of the Manual

(1) xArm motion characteristics

(2) xArm error reporting and handling

(3) xArm technical specifications

1.3. xArm Motion Parameters

The parameters of the robotic arm are shown in Table 1.1 and Table 1.2.

Table 1.1 working range of each joint of the robotic arm

	Robotic Arm	xArm 5	xArm 6	xArm 7
Maximum Speed		180°/s	180°/s	180°/s
Working Range	1st Axis	±360°	±360°	±360°
	2st Axis	-118°～120°	-118°～120°	-118°～120°
	3st Axis	-225°～11°	-225°～11°	±360°
	4st Axis	±360°	±360°	-11°～225°
	5st Axis	-97°～180°	-97°～180°	±360°
	6st Axis	None	±360°	-97°～180°
	7st Axis	None	None	±360°

Table 1.2 range of various motion parameters of the robotic arm

	TCP Motion	Joint Motion
Speed	0～1000mm/s	0～180°/s
Acceleration	0～50000mm/s^2	0～1145$°/s^2$
Jerk	0～10000mm/s^3	0～28647$°/s^3$

Note:

1. In the TCP motion (Cartesian space motion) commands (set_position () function of the SDK), If a motion command involves both position transformation and attitude transformation, the attitude rotation speed is generally calculated automatically by the system. In this situation, the specified speed parameter is the maximum linear speed, range from: 0 ～ 1000mm / s.
2. When the expected TCP motion only changes the attitude (roll, pitch, yaw), with position (x, y, z) remains unchanged, the specified speed is the attitude rotation speed, so the range 0 to 1000 corresponds to 0 to 180 ° / s.

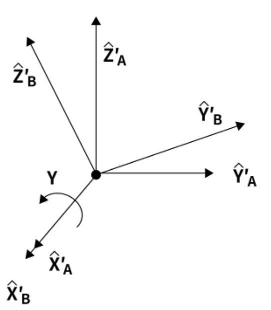
1.4. Unit Definition

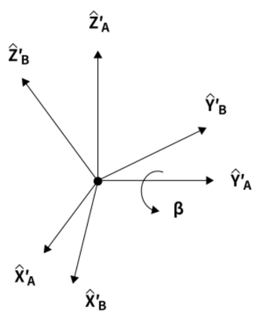
The Python / Blockly examples and the units standard in the communication protocol are shown in Table 1.3.

Table 1.3. Default units in Python / Blockly example and Communication Protocol

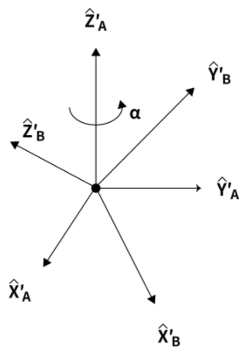
Parameter	Python-SDK	Blockly	Communication Protocol
X（Y/Z）	millimeter（mm）	millimeter（mm）	millimeter（mm）
Roll（Pitch/Yaw）	degree（°）	degree（°）	radian（rad）
J_1（J_2/J_3/J_4/J_5/J_6/J_7）	degree（°）	degree（°）	radian（rad）
TCP Speed	mm/s	mm/s	mm/s
TCP Acceleration	mm/s^2	mm/s^2	mm/s^2
TCP Jerk	mm/s^3	mm/s^3	mm/s^3
Joint Speed	°/s	°/s	rad/s
Joint Acceleration	$°/s^2$	$°/s^2$	rad/s^2
Joint Jerk	$°/s^3$	$°/s^3$	rad/s^3

1.5. Terms and Definitions

<table>
<tr><td>Control Box</td><td>The control box, core part of the robotic arm, is the integration of the robotic arm control system.</td></tr>
<tr><td>End Effector</td><td>The end effector, installed on the front end of the wrist of the robotic arm, is used to install special tools (such as grippers, vacuum gripper, etc.), which can directly perform work tasks.</td></tr>
<tr><td>Enable Robotic Arm</td><td>Power on the robotic arm and turn on the motor of the robotic arm. After the robotic arm is enabled, it can start to move normally.</td></tr>
<tr><td>TCP</td><td>Tool center point.</td></tr>
<tr><td>TCP Motion</td><td>TCP motion is the Cartesian space motion, with target position in Cartesian space coordinate and the end follows the specified trajectory(arc, line, etc.).</td></tr>
<tr><td>TCP Payload
（End Payload）</td><td>The payload weight refers to the actual (end tool +other object) weight in Kg; the X / Y / Z-axis indicates the position of the center of mass of the TCP relative to the default tool coordinate system,with unit of mm.</td></tr>
<tr><td>TCP Offset
(Tool Center Point Offset)</td><td>Set the relative offset between the default tool coordinate system at flange center and the actual tool coordinate system, with distance unit of mm.</td></tr>
<tr><td>Roll/Pitch/Yaw</td><td>Roll / Pitch / Yaw sequentially rotates around the X / Y / Z of the selected coordinate system (base coordinate system).
The following describes the roll/pitch/yaw orientation representation of {B} relative to {A}:
For example, the coordinate system {B} and a known reference coordinate system {A} are first superposed. First rotate {B} around $\hat{X}_A$ by γ, then around $\hat{Y}_A$ by β, and finally around $\hat{Z}_A$ by α.
Each rotation is around a fixed axis of the reference coordinate system {A}. This method is called the XYZ fixed angle coordinate system, and sometimes they are defined as the roll angle, pitch angle, and yaw angle.
The above description is shown in the following figure:




</td></tr>
</table>

	 The equivalent rotation matrix is: ${}^{A}_{B}R_{XYZ}(\gamma,\beta,\alpha)=R_Z(\alpha)R_Y(\beta)R_X(\gamma)$ Note: γ corresponds to roll; β corresponds to pitch; α corresponds to yaw.
Axis-Angle	Rx / Ry / Rz representation also, using 3 values to represent the pose (but not three rotation angles), which is the product of a three-dimensional rotation vector [x, y, z] and a rotation angle[phi (scalar)]. The characteristics of the axis angle: Assume the rotation axis is [x , y, z], and the rotation angle is phi. Then the representation of the axial angle: [Rx, Ry, Rz] = [x * phi, y * phi, z * phi] Note: 1. [x, y, z] is a unit vector, and phi is a non-negative value. 2. The vector length (modulus) of [Rx, Ry, Rz] can be used to estimate the rotation angle, and the vector direction is the rotation direction. 3. If you want to express reverse rotation, invert the rotation axis vector [x, y, z], and the value of phi remains unchanged. 4. Using phi and [x, y, z] can also derive the attitude representation as unit quaternion q = [cos (phi / 2), sin (phi / 2) * x, sin (phi / 2) * y, sin (phi / 2) * z]. For example: The vector of the rotation axis represented by the base coordinate system is [1, 0, 0], and the rotation angle is 180 degrees (π), then the axis angle representation of this pose is [π, 0, 0]. The rotation axis is [0.707, 0.707, 0] and the rotation angle is 90 degrees (π / 2), then the axis angle posture is [0.707 * (π / 2), 0.707 * (π / 2), 0].
The Base Coordinate System (please refer to the figure 1)	The base coordinate system is a Cartesian coordinate system based on the mounting base of the robotic arm and used to describe the motion of the robotic arm. (front and back: X axis, left and right: Y axis, up and down: Z axis)
Tool Coordinate System (please refer to the figure 1)	Consists of tool center point and coordinate orientation. If the TCP offset is not set, the default tool coordinate system is located at flange center. For tool coordinate system based motion: The tool center point is taken as the zero point, and the trajectory of the robotic arm refers to the tool coordinate system.
User Coordinate System	The user coordinate system can be defined as any other reference coordinate

(please refer to the figure 1)	system rather than the robot base.
Manual Mode	In this mode, the robotic arm will enter the 'zero gravity' mode, since the gravity is compensated, the user can guide the robotic arm position directly by hand.
Teach Sensitivity	Teach sensitivity range is from 1 to 5 level. The larger the set value, the higher the teach sensitivity level, and the less the force required to drag the joint in the manual mode.
Collision Sensitivity	The collision sensitivity range is from 0 to 5 level. When it is set to 0, it means that collision detection is not enabled. The larger the set value, the higher the collision sensitivity level, and the smaller the force required to trigger the collision protection response of the robotic arm.
GPIO	General-purpose input and output. For the input, you can check the potential of the pin by reading a register; For the output, you can write a certain register to make this pin output high or low potential;
Safety Boundary	When this mode is activated, the boundary range of the cartesian space of the robotic arm can be limited. If the tool center point (TCP) exceeds the set safety boundary, the robotic arm will stop moving.
Reduced Mode	When this mode is activated, the maximum linear velocity of the Cartesian motion of the robotic arm, the maximum joint speed, and the range of the joint motion will be limited.

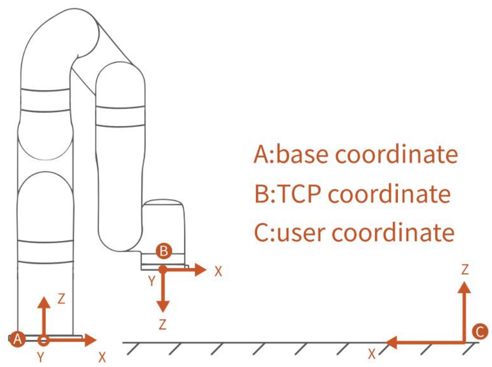


Figure 1

1.6. Further Developer Resources

ROS Library & Github: https://github.com/xArm-Developer/xarm_ros

xArm Python SDK Library: https://github.com/xArm-Developer/xArm-Python-SDK

xArm CPLUS SDK Library: https://github.com/xArm-Developer/xArm-CPLUS-SDK

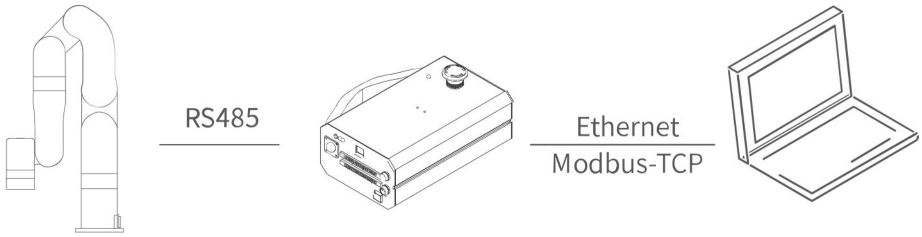
Note: For the above three developer resources, we have detailed installation steps and commands on github. Please download the installation package for further development.

1.7. More Information

- ● More product information: https://www.ufactory.cc/#/en/
- ● For technical support, please email to: support@ufactory.cc
- ● For sales support, please email to: sales@ufactory.cc

2. xArm Communication Protocol

2.1. Control Box Communication Protocol



Note: The current protocol has some format changes for xArm. Please use this manual as the main protocol when running the robotic arm.

The main content of this chapter has two parts:

(1) Control the motion of the robotic arm by Modbus TCP through AC/DC Control Box.

(2) Control the IO device of the control box and the IO device at the end of the robotic arm by Modbus TCP through AC/DC Control Box.

2.1.1. Unit Definition

The following explains some of the symbols used in the examples and tables:

【u8】: 1 Byte, 8-bit unsigned int

【u16】: 2 Bytes, 16-bit unsigned int

【fp32】: 4 Bytes, float

【str】: string

【System reset】: The user just enters the state after the mode switch or changes some settings (such as TCP offset, sensitivity, etc.). The above operations will terminate the ongoing movement of the robotic arm and clear the cache commands, which is the same as the STOP state.

2.1.2. 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 Control (Only this one for now)

Request Commands Format

Format	Transaction Identifier (u16)	Protocol (u16)	Length (u16)	Register (u8)	Parameters (Refer to the statement of each commands
Length	2 Bytes	2 Bytes	2 Bytes	1 Byte	n Bytes
Example (Enable the robotic arm)	0x00 0x01	0x00 0x02	0x00 0x03	0x0B	0x08 0x01

Response command format

Format	Transaction Identifier (u16)	Protocol (u16)	Length (u16)	Register (u8)	Status (u8)	Parameters (Refer to the statement of each commands
Length	2 Bytes	2 Bytes	2 Bytes	1 Byte	1 Byte	n Bytes
Example (Enable the robotic arm)	0x00 0x01	0x00 0x02	0x00 0x02	0x0B	0x00	none

Status Bit of the Response Format

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0: normal	1: error 0: normal	1: warning 0: normal	1: cannot perform motion 0: normal	0: normal	0: normal	0: normal	0: normal

General notes:

●**Transaction Identifier:** Generally, 1 is added after each communication to distinguish different communication data packets.

● **Protocol :** 0x00 0x02 means ModbusTCP protocol.

● **Length:** Indicates the next data length in bytes.

● **Register:** Device address.

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

Modbus-TCP control protocol:

1. The transaction identifier (u16) are analyzed in big endian order.
2. protocol identifier (u16) and are analyzed in big endian order.
3. length (u16) of the message head are analyzed in big endian order.
4. The 32-bit data (fp32, int32) in the parameter are analyzed in little endian order.
5. Integer data(u16) involving GPIO operation are analyzed in big endian order.

Automatic reporting data analysis:

1. Integer data (16/32 bits) are analyzed in big endian order.
2. Floating-point (fp32) data is analyzed in little endian order.

Example:

Assume that the type of the variable x is int, located at address 0x100, there is a hexadecimal number 0x12345678 (high order is 0x12, low order is 0x78), and the byte order of the address range 0x100-0x103 depends on the type of machine:

Big-endian method:

	0x100	0x101	0x102	0x103	
...	0x12	0x34	0x56	0x78	...

Little-endian method:

	0x100	0x101	0x102	0x103	
...	0x78	0x56	0x34	0x12	...

2.1.3. Register (Robotic Arm Control)

2.1.3.1 Register (General)

The following is an example of joint motion, axis angular motion, setting parameters, getting parameters, and special IO commands

Function	Joint motion	Set the maximum acceleration of TCP motion	Get cartesian position	Linear motion of the target in the axial angle posture	The operation triggered by the position of the general digital IO of the control box

Joint motion (P2P motion)				
Register23（0x17）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x29
	Register	1 Byte	u8	0x17
Parameters	Joint1（J1=π/3）	4 Bytes	fp32	0x92,0x0A,0x86,0x3F
	Joint2（J2=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint3（J3=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint4（J4=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint5（J5=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint6（J6=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint7（J7=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter8(speed=20*π/180rad/s)	4 Bytes	fp32	0xC2,0xB8,0xB2,0x3E
	Parameter9(acceleration=500*π/180rad/s²）	4 Bytes	fp32	0x58,0xA0,0x0B,0x41
	Parameter10(motion time=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x17
Parameters	State	1 Byte	u8	0x00
	Parameter	2 Bytes	u16	0x00,0x01

Set the maximum acceleration of TCP motion				
Register32（0x20）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x05
	Register	1 Byte	u8	0x20
Parameters	Parameter1 (maxacc=1000mm/s²)	4 Bytes	fp32	0x00,0x00,0x7A,0x44
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02

	Length	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x20
Parameters	State	1 Byte	u8	0x00
	Parameter	2 Bytes	u16	0x00,0x01

Get Cartesian position				
Register41（0x29）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x29
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x0,0x1A
	Register	1 Byte	u8	0x29
Parameters	State	1 Byte	u8	0x00
	Parameter1(x=207mm)	4 Bytes	fp32	0x00,0x00,0x4F,0x43
	Parameter2(y=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3(z=112mm)	4 Bytes	fp32	0x00,0x00,0xE0,0x42
	Parameter4(roll=π)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5(pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6(yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00

Linear motion of the target in the axis angle posture				
Register92（0x5C）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)（ParameterLength	2 Bytes	u16	0x00,0x27
	Register	1 Byte	u8	0x5C
Parameters	Parameter1(X=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter2(Y=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3(Z=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter4(Rx=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter5(Ry=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6(Rz=2π)	4 Bytes	fp32	0xDB,0x0F,0xC9,0x40
	Parameter7(speed=100mm/s)	4 Bytes	fp32	0x00,0x00,0xC8,0x42
	Parameter8(acceleration=2000mm/s^2))	4 Bytes	fp32	0x00,0x00,0xFA,0x44

	Parameter9(motion time=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter10（Motion coordinate system) 0 represents base coordinate system motion 1 represents tool coordinate system motion	1 Byte	u8	0x00
	Parameter11(absolute pose) If the motion coordinate system is the base coordinate system 0 represents the given pose is an absolute pose 1 represents the given pose is a relative pose (the given parameters 1-6 coordinates are based on the current an offset of position)	1 Byte	u8	0x01
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x5C
Parameters	State	1 Byte	u8	0x00
	Parameter	2 Bytes	u16	0x00,0x01

The operation triggered by the position of the general digital IO of the control box				
Register145（0x91）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x13
	Register	1 Byte	u8	0x91
Parameters	Parameter1(iomum=0)	1 Byte	u8	0x00
	Parameter2(on-off：on(1))	1 Byte	u8	0x01
	Parameter3（x=300）	4 Bytes	fp32	0x00,0x00,0x96,0x43
	Parameter4（y=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter5（z=300）	4 Bytes	fp32	0x00,0x00,0x96,0x43
	Parameter6（Tolerance radius（tol_r）=3）	4 Bytes	fp32	0x00,0x00,0x40,0x40
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x91
Parameters	State	1 Byte	u8	0x00

2.1.3.2 Register (Robotic Arm Control)

0~10: Public Port Section

Get version information（0x01）

Get the value of joint torque or actual current (0x05)

Remotely shut down the operating system（0x0A）

11~20: System State

Enable/Disable servo(System reset)（0x0B）

Motion state setting（0x0C）

Get the motion state（0x0D）

Get the number of commands in the command buffer（0x0E）

Get error and warning code（0x0F）

Clear control box error(System reset)（0x10）

Clear control box warning（0x11）

Setting the brake switches separately（0x12）

Setting the system motion mode（0x13）

20~30: Basic Motion

Cartesian linear motion（0x15）

Linear motion with circular arc（0x16）

P2P joint motion（0x17）

Set joint torque or motor current report（0x18）

Return to zero position（0x19）

Pause commands, Commands delay（0x1A）

Linear circular motion（0x1B）

Linear motion in tool coordinate system（0x1C）

Servoj motion（0x1D）

Servo_cartesian motion（0x1E）

31~40: System Parameter Setting

Set the jerk of the cartesian space translation（0x1F）

Set the maximum acceleration of the cartesian space translation（0x20）

0~10 Common Port Section

Get version information				
Register：1(0x01)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x01
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x01
Parameter	State	1 Byte	u8	0x00

Get the value of Joint torque or actual current				
Register: 5(0x05)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x05
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x03
	Register	1 Byte	u8	0x05
Parameters	State	1 Byte	u8	0x00
	Parameter 1 (Value of theoretical joint torque) 0: Value of theoretical joint torque 1: Value of actual current of servo	1 Byte	u8	0x00

Remote shut down the operating system				
Register10（0x0A）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02

	Register	1 Byte	u8	0x0A
Parameters	Parameter1 (Operation: remote shut down the operating system temporarily)	1 Byte	u8	0x01
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x0A
Parameters	State	1 Byte	u8	0x00
	Parameter	2 Bytes	u16	0x00,0x01

11~20 System State

Enable/Disable servo (System reset) **Note: The above operations will terminate the ongoing movement of the robotic arm and clear the cache commands, which is the same as the STOP state.**				
Register：11(0x0B)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x03
	Register	1 Byte	u8	0x0B
Parameters	Joint Number(Select all joints) 1-7：Motor joint(1-7) 8：Select all joints	1 Byte	u8	0x08
	Whether to enable the servo 1：Enable servo 0：Disable servo	1 Byte	u8	Enable: 0x01 Disable: 0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x0B
Parameters	State	1 Byte	u8	0x00

Motion state setting				
Register：12(0x0C)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02

	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x0C
Parameters	Parameter1: Motion Sate 3: Suspend the current motion 4: Stop all current motion (restart the system) 0: Enter the motion mode	1 Byte	u8	0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x0C
Parameters	State	1 Byte	u8	0x00

Get the motion state				
Register：13 (0x0D)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x0D
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x03
	Register	1 Byte	u8	0x0D
Parameters	State	1 Byte	u8	0x00
	Parameter1 Motion state： 1：In motion 2：Sleep 3：Suspend 4：Stop 5：System reset The user just enters the state after the mode switch or changes some settings (such as TCP offset, sensitivity, etc.). The above operations will terminate the ongoing movement of the robotic arm and clear the cache commands, which is the same as the STOP state.	1 Byte	u8	0x01

Get the number of commands in the command buffer				
Register：14 (0x0E)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x0E
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x0E
Parameters	State	1 Byte	u8	0x00
	Parameter1 (The number of commands in the buffer)	2 Bytes	u16	0x00,0x01

Get error and warning code				
Register：15 (0x0F)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x0F
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x0F
Parameters	State	1 Byte	u8	0x00
	Parameter1（Error code）	1 Byte	u8	0x00
	Parameter2（Warning code）	1 Byte	u8	0x00

Clear control box error (System reset) Note: The above operations will terminate the ongoing movement of the robotic arm and clear the cache commands, which is the same as the STOP state.				
Register：16 (0x10)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x10
Response				

Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x10
Parameters	State	1 Byte	u8	0x00

Clear control box warning				
Register：17 (0x11)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x11
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x11
Parameters	State	1 Byte	u8	0x00

Setting the brake switches separately (System reset) **Note: The above operations will terminate the ongoing movement of the robotic arm and clear the cache commands, which is the same as the STOP state.**				
Register：18 (0x12)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x03
	Register	1 Byte	u8	0x12
Parameters	Parameter1(Select all joints) Control the brakes： 1~6: Select motor joint separately 8: Select all joints	1 Byte	u8	0x08
	Parameter2 (Enable the brake) Operation： 1: Enable the brake 0: Release the brake	1 Byte	u8	0x01
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x12

Parameters	State	1 Byte	u8	0x00

Setting the system motion mode (System reset) Note: The above operations will terminate the ongoing movement of the robotic arm and clear the cache commands, which is the same as the STOP state.				
Register：19　(0x13)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x13
Parameters	Parameter1(Position control mode) Motion mode： 0: Position control mode 1: servo motion mode 2: Joint teaching mode 3: Cartesian teaching mode (not yet available)	1 Byte	u8	0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x13
Parameters	State	1 Byte	u8	0x00

21~30 Basic Motion

Cartesian linear motion				
Register21（0x15）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x25
	Register	1 Byte	u8	0x15
Parameters	Parameter1(x=400mm)	4 Bytes	fp32	0x00,0x00,0xC8,0x43
	Parameter2(y=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3(z=200mm)	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameter4(roll=π)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5(pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6(yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter8(speed=100mm/s)	4 Bytes	fp32	0x00,0x00,0xC8,0x42
	Parameter9(acceleration=2000mm/s	4 Bytes	fp32	0x00,0x00,0xFA,0x44
	Parameter10(motion time=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00

Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x15
Parameters	State	1 Byte	u8	0x00
	Parameter	2 Bytes	u16	0x00,0x01

Linear motion with circular arc				
Register：22 (0x16)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x29
	Register	1 Byte	u8	0x16
Parameters	Parameter1(x=400mm)	4 Bytes	fp32	0x00,0x00,0xC8,0x43
	Parameter2(y=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3(z=200mm)	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameter4(roll=π)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5(pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6(yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter7 (Arc blending radius=50 mm)	4 Bytes	fp32	0x00,0x00,0x48,0x42
	Parameter8 (motion speed=100 mm/s)	4 Bytes	fp32	0x00,0x00,0xC8,0x42
	Parameter9 (acceleration=2000mm/s^2)	4 Bytes	fp32	0x00,0x00,0xFA,0x44
	Parameter10 (motion time (0))	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x16
Parameters	State	1 Byte	u8	0x00
	Parameter1 (Number of commands in the buffer)	2 Bytes	u16	0x00,0x01

P2P joint motion				
Register：23 (0x17)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x29
	Register	1 Byte	u8	0x17
Parameters	Joint1（J1= π/3）	4 Bytes	fp32	0x92,0x0A,0x86,0x3F

	Joint2（J2=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint3（J3=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint4（J4=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint5（J5=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint6（J6=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint7（J7=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter8(speed=20*π/180rad/s)	4 Bytes	fp32	0xC2,0xB8,0xB2,0x3E
	Parameter9(acceleration500*π/180rad/s^2）	4 Bytes	fp32	0x58,0xA0,0x0B,0x41
	Parameter10(motion time=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x17
Parameters	State	1 Byte	u8	0x00
	Parameter1 (Number of commands in the buffer)	2 Bytes	u16	0x00,0x01

Set joint torque or motor current report				
Register：24 (0x18)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x18
Parameters	Parameter1 0: Report the estimated value of joint torque, Unit:Nm 1: Report the current value read by the motor, Unit:A	1 Byte	u8	0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x18
Parameters	State	1 Byte	u8	0x00

Return to zero position				
Register：25 (0x19)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x0D

	Register	1 Byte	u8	0x19
Parameters	Parameter 1（speed=50rad/s）	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter2（acceleration=600rad/s^2）	4 Bytes	fp32	0xF3,0x66,0xDF,0x40
	Parameter3（motion time=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x19
Parameters	State	1 Byte	u8	0x00
	Parameter1 (Number of commands in the buffer)	2 Bytes	u16	0x00,0x01

Pause commands, Command delay				
Register：26(0x1A)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x05
	Register	1 Byte	u8	0x1A
Parameters	Parameter1 (Pause time=3s)	4 Bytes	fp32	0x00,0x00,0x40,0x40
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x1A
Parameters	State	1 Byte	u8	0x00
	Parameter1 (Number of commands in the buffer)	2 Bytes	u16	0x00,0x01

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)**				
Register：27 (0x1B)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x41
	Register	1 Byte	u8	0x1B
Parameters	Parameter1(x=400mm)	4 Bytes	fp32	0x00,0x00,0xC8,0x43
	Parameter2(y=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00

	Parameter3(z=200mm)	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameter4(roll=π)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5(pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6(yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter7(x=400mm)	4 Bytes	fp32	0x00,0x00,0xC8,0x43
	Parameter8(y=0mm)	4 Bytes	fp32	0x00, 0x00, 0xC8, 0x42
	Parameter9(z=200mm)	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameter10(roll=π)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter11(pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter12(yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter13（Percentage of the length of arc in motion to circumference=50%）	4 Bytes	fp32	0x00,0x00,0x48,0x42
	Parameter14(speed=20*π/180rad/s)	4 Bytes	fp32	0x00,0x00,0xC8,0x42
	Parameter15(acceleration500*π/180rad/s^2)	4 Bytes	fp32	0x00,0x00,0xFA,0x44
	Parameter16(motion time=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x1B
Parameters	State	1 Byte	u8	0x00
	Parameter1 (Number of commands in the buffer)	2 Bytes	u16	0x00,0x01

Linear motion in tool coordinate system **Move in Cartesian linear relative motion based on the current tool coordinate system.**				
Register：28 (0x1C)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x25
	Register	1 Byte	u8	0x1C
Parameters	Parameter1(x=400mm)	4 Bytes	fp32	0x00,0x00,0xC8,0x43
	Parameter2(y=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3(z=200mm)	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameter4(roll=π)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5(pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6(yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter7(speed=20mm/s)	4 Bytes	fp32	0xC2,0xB8,0xB2,0x3E
	Parameter8(acceleration=2000mm/s^2）	4 Bytes	fp32	0x00,0x00,0xFA,0x44
	Parameter9(motion time=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00

Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x1C
Parameters	State	1 Byte	u8	0x00
	Parameter1 (Number of commands in the buffer)	2 Bytes	u16	0x00,0x01

Servoj motion				
Register：29 (0x1D)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x29
	Register	1 Byte	u8	0x1D
Parameters	Joint1（J1= π/3）	4 Bytes	fp32	0x92,0x0A,0x86,0x3F
	Joint2（J2=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint3（J3=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint4（J4=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint5（J5=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint6（J6=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Joint7（J7=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter8 (speed, meaningless, 0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter9 (acceleration, meaningless, 0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter10 (motion time, meaningless, 0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x1D
Parameters	State	1 Byte	u8	0x00

Servo_cartesian motion **Interface for receiving high-frequency continuous cartesian trajectory motion.**				
Register：30 (0x1E)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x25

	Register	1 Byte	u8	0x1E
Parameters	Parameter1(x=400mm)	4 Bytes	fp32	0x00,0x00,0xC8,0x43
	Parameter2(y=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3(z=200mm)	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameter4(roll=π)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5(pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6(yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter8 (speed, meaningless, 0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter9 (acceleration, meaningless, 0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter10 Motion coordinate system： 0 ：the base coordinate system 1 ：the tool coordinate system	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x1E
Parameters	State	1 Byte	u8	0x00

31~40 Motion Parameter Setting

Set the jerk of the Cartesian space translation				
Register：31 (0x1F)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x05
	Register	1 Byte	u8	0x1F
Parameters	Parameter1（Jerk=2000 mm/s^3）	4 Bytes	fp32	0x00,0x00,0xFA,0x44
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x1F
Parameters	State	1 Byte	u8	0x00
	Parameter1 (Number of commands in the buffer)	2 Bytes	u16	0x00,0x01

Set the maximum acceleration of the Cartesian space translation
Register：32 (0x20)

Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x05
	Register	1 Byte	u8	0x20
Parameters	Parameter1 (Maximum acceleration=6000mm/s^2)	4 Bytes	fp32	0x00,0x80,0xbb,0x45
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x20
Parameters	State	1 Byte	u8	0x00
	Parameter1 (Number of commands in the buffer)	2 Bytes	u16	0x00,0x01

Set the joint space jerk				
Register：33 (0x21)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x05
	Register	1 Byte	u8	0x21
Parameters	Parameter1（Jerk=10000rad/s^3）	4 Bytes	fp32	0x00,0x40,0x1C,0x46
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x21
Parameters	State	1 Byte	u8	0x00
	Parameter1 (Number of commands in the buffer)	2 Bytes	u16	0x00,0x01

Set joint space max acceleration				
Register：34 (0x22)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x05

	Register	1 Byte	u8	0x22
Parameters	Parameter（Max acceleration=400rad/s^2）	4 Bytes	fp32	0x00,0x00,0xC8,0x43
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x22
Parameters	State	1 Byte	u8	0x00
	Parameter1 (Number of commands in the buffer)	2 Bytes	u16	0x00,0x01

Set the offset of the robotic arm end-effector (System reset) **Note: The above operations will terminate the ongoing movement of the robotic arm and clear the cache commands, which is the same as the STOP state.**				
Register：35 (0x23)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x19
	Register	1 Byte	u8	0x23
Parameters	Parameter1(x=400mm)	4 Bytes	fp32	0x00,0x00,0xC8,0x43
	Parameter2(y=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3(z=200mm)	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameter4(roll=π)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5(pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6(yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x23
Parameters	State	1 Byte	u8	0x00

End payload setting				
Register：36 (0x24)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x11
	Register	1 Byte	u8	0x24
Parameters	Parameter1（Payload=1kg）	4 Bytes	fp32	0x00,0x00,0x80,0x3F

	Parameter2(Payload center of mass X=400mm）	4 Bytes	fp32	0x00,0x00,0xC8,0x43
	Parameter3(Payload center of mass Y=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter4(Payload center of mass Z=200mm）	4 Bytes	fp32	0x00,0x00,0x48,0x43
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x24
Parameters	State	1 Byte	u8	0x00

Set collision detection sensitivity (System reset) **Note: The above operations will terminate the ongoing movement of the robotic arm and clear the cache commands, which is the same as the STOP state.**				
Register：37(0x25)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x25
Parameters	Parameter1 (Detect sensitivity=4)	1 Byte	u8	0x04
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x25
Parameters	State	1 Byte	u8	0x00

Set teaching sensitivity for teaching mode (System reset) **Note: The above operations will terminate the ongoing movement of the robotic arm and clear the cache commands, which is the same as the STOP state.**				
Register：38(0x26)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x26
Parameters	Parameter1 (Teach sensitivity=4)	1 Byte	u8	0x04
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x26

Parameters	State	1 Byte	u8	0x00

Delete the current system configuration parameters				
Register：39 (0x27)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x27
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x27
Parameters	State	1 Byte	u8	0x00

Save the current system configuration parameters				
Register：40 (0x28)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x28
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x28
Parameters	State	1 Byte	u8	0x00

41~50 Get Motion Information

Get the current Cartesian position of the robotic arm				
Register41（0x29）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x29
Response				

Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x0,0x1A
	Register	1 Byte	u8	0x29
Parameters	State	1 Byte	u8	0x00
	Parameter1(x=207mm)	4 Bytes	fp32	0x00,0x00,0x4F,0x43
	Parameter2(y=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3(z=112mm)	4 Bytes	fp32	0x00,0x00,0xE0,0x42
	Parameter4(roll=π)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5(pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6(yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00

Get the current joint position of the robotic arm				
Register：42 (0x2A)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x2A
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x1E
	Register	1 Byte	u8	0x2A
Parameters	State	1 Byte	u8	0x00
	joint1（J1=π/3）	4 Bytes	fp32	0x92,0x0A,0x86,0x3F
	joint2（J2=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint3（J3=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint4（J4=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint5（J5=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint6（J6=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint7（J7=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00

Get the solution of the inverse kinematics				
Register：43 (0x2B)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x19
	Register	1 Byte	u8	0x2B
Parameters	Parameter1(x=400mm)	4 Bytes	fp32	0x00,0x00,0xC8,0x43
	Parameter2(y=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00

	Parameter3(z=200mm)	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameter4(roll=π)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5(pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6(yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x1E
	Register	1 Byte	u8	0x2B
Parameters	State	1 Byte	u8	0x00
	joint1（J_1= 0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint2（J_2=0.081803）	4 Bytes	fp32	0x38,0x88,0xA7,0x3D
	joint3（J_3=-0.641152）	4 Bytes	fp32	0x88,0x22,0x24,0xBF
	joint4（J_4=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint5（J_5=0.559349）	4 Bytes	fp32	0x81,0x31,0x0F,0x3F
	joint6（J_6=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint7（J_7=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00

Get the solution of the forward kinematics				
Register：44 (0x2C)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x1D
	Register	1 Byte	u8	0x2C
Parameters	joint1（J1= π/3）	4 Bytes	fp32	0x92,0x0A,0x86,0x3F
	joint2（J2=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint3（J3=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint4（J4=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint5（J5=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint6（J6=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint7（J7=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x1A
	Register	1 Byte	u8	0x2C
Parameters	State	1 Byte	u8	0x00
	Parameter1(x=103.5mm)	4 Bytes	fp32	0x18,0x00,0xCF,0x42
	Parameter2(y=179.27mm)	4 Bytes	fp32	0x80,0x44,0x33,0x43
	Parameter3(z=112mm)	4 Bytes	fp32	0x08,0x01,0xA0,0x42
	Parameter4(roll=-π)	4 Bytes	fp32	0xDB,0x0F,0x49,0xC0

	Parameter5(pitch=-0)	4 Bytes	fp32	0x00,0x00,0x00,0x80
	Parameter6(yaw=-π/3)	4 Bytes	fp32	0x92,0x0A,0x86,0x3F

Check the limit of joint space				
Register：45 (0x2D)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x1D
	Register	1 Byte	u8	0x2D
Parameters	joint1（J1=π/3）	4 Bytes	fp32	0x92,0x0A,0x86,0x3F
	joint2（J2=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint3（J3=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint4（J4=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint5（J5=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint6（J6=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	joint7（J7=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x03
	Register	1 Byte	u8	0x2D
Parameters	State	1 Byte	u8	0x00
	Parameter1 Search result: 1 ： Collision occurs 0 ： No collision occurs	1 Byte	u8	0x00

51~100 Other Robotic Arm Function

Set the gravity direction **Set the gravity direction for correct torque compensation and collision detection. After modification, it shall call the save_conf () function or refer to Register: 40(0x28) to save the setting, otherwise it will be invalid after the next restart.**				
Register：51 (0x33)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x0D
	Register	1 Byte	u8	0x33

Parameters	Parameter1 Gravity direction vector X=0 （base coordinate system）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter2 Gravity direction vector Y=0 （base coordinate system）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3 Gravity direction vector Z=-1 （base coordinate system）	4 Bytes	fp32	0x00,0x00,0x80,0xBF
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x33
Parameters	State	1 Byte	u8	0x00

Set the safe boundary range **C35 Set the boundary range of the safety fence in the three-dimensional space. If TCP of the robotic arm exceeds this boundary, error C35of the Control Box will be triggered.**				
Register：52 (0x34)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x19
	Register	1 Byte	u8	0x34
Parameters	Parameter1 Cartesian boundary value x+=600mm	4 Bytes	int32	0x58,0x02,0x00,0x00
	Parameter2 Cartesian boundary value x-=200mm	4 Bytes	int32	0xC8,0x00,0x00,0x00
	Parameter3 Cartesian boundary value y+ =500mm	4 Bytes	int32	0xF4,0x01,0x00,0x00
	Parameter4 Cartesian boundary value y- =100mm	4 Bytes	int32	0x64,0x00,0x00,0x00
	Parameter5 Cartesian boundary value z+=600mm	4 Bytes	int32	0x58,0x02,0x00,0x00
	Parameter6 Cartesian boundary value z-=200mm	4 Bytes	int32	0xC8,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02

	Register	1 Byte	u8	0x34
Parameters	State	1 Byte	u8	0x00

Get current joint torque of the servo **Estimate the joint torque based on current and theoretical model, which is for reference only.**				
Register：55 (0x37)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x37
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x1E
	Register	1 Byte	u8	0x37
Parameters	State	1 Byte	u8	0x00
	Parameter1 (Theoretical torque of joint1=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter2 (Theoretical torque of joint2= -13.7 N.m)	4 Bytes	fp32	0x2A,0xC5,0x5B,0xC1
	Parameter3 (Theoretical torque of joint3= -6.17 N.m)	4 Bytes	fp32	0x79,0xA4,0xC5,0xC0
	Parameter4 (Theoretical torque of joint4=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter5 (Theoretical torque of joint5=-1.83N.m)	4 Bytes	fp32	0x87,0xA3,0xE9,0xBF
	Parameter6 (Theoretical torque of joint6=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter7 (Theoretical torque of joint7=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00

Safety boundary start switch **Set the safety fence boundary validation switch in three-dimensional space. If the TCP of the robotic arm exceeds this boundary after validation, error C35 of the Control Box will be triggered.**				
Register: 59 (0x3B)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01

	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x3B
Parameters	Parameter1 Validation switch 0: Turn off safety boundary detection 1: Turn on safety boundary detection	1 Byte	u8	0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x3B
Parameters	State	1 Byte	u8	0x00

Set the joint torque (theoretical) and current of servo correspond to the contents of reporting port 60~87 Bytes				
Register: 70 (0x46)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x46
Parameters	Parameter1 (value of theoretical joint torque) 0: value of theoretical joint torque, unit : Nm 1: value of actual current of servo, unit : A	1 Byte	u8	0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x46
Parameters	State	1 Byte	u8	0x00

Sets the offset of the user coordinate system and the base coordinate system **Sets the offset of the user coordinate system and the base coordinate system, specifically the offset described by the base coordinate system of the robotic arm under the user-defined coordinate system**
Register: 73 (0x49)
Request

Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x19
	Register	1 Byte	u8	0x49
Parameters	Parameter1 (Cartesian offset X=400mm)	4 Bytes	fp32	0x00,0x00,0xC8,0x43
	Parameter2 (Cartesian offset Y=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3 (Cartesian offset Z=200mm)	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameter4 (Cartesian offset Roll=πrad)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5 (Cartesian offset Pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6 (Cartesian offset Yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x49
Parameters	State	1 Byte	u8	0x00

Calculate the attitude offset of two given points **Given two coordinate points of the robotic arm, the offset coordinate between them can be calculated.**					
Register: 76 (0x4C)					
Request					
Modbus TCP Header	Transaction ID		2 Bytes	u16	0x00,0x01
	Protocol		2 Bytes	u16	0x00,0x02
	Length (parameter length+1)		2 Bytes	u16	0x00,0x33
	Register		1 Byte	u8	0x4C
Parameters	Point1	Parameter1（X=400）	4 Bytes*6	fp32*6	0x00,0x00,0xC8,0x43
		Parameter2（Y=0）			0x00,0x00,0x00,0x00
		Parameter3（Z=200）			0x00,0x00,0x48,0x43
		Parameter4（Roll=π）			0xDB,0x0F,0x49,0x40
		Parameter5（Pitch=0）			0x00,0x00,0x00,0x00
		Parameter6（Yaw=0）			0x00,0x00,0x00,0x00
	Point2	Parameter7（X=400）	4 Bytes*6	fp32*6	0x00,0x00,0xC8,0x43
		Parameter8（Y=0）			0x00,0x00,0x00,0x00
		Parameter9（Z=100）			0x00,0x00,0xC8,0x42

		Parameter10（Roll=π）			0xDB,0x0F,0x49,0x40
		Parameter11（Pitch=0）			0x00,0x00,0x00,0x00
		Parameter12（Yaw=0）			0x00,0x00,0x00,0x00
	Parameter13（RPY） Representation of input pose: 0 : RPY（Roll,Pitch,Yaw） 1 : axial angle（Rx,Ry,Rz）		1 Byte	u8	0x00
	Parameter14（RPY） Representation of output pose: 0 : RPY（Roll,Pitch,Yaw） 1 : axial angle（Rx,Ry,Rz）		1 Byte	u8	0x00
Response					
Modbus TCP Header	Transaction ID		2 Bytes	u16	0x00,0x01
	Protocol		2 Bytes	u16	0x00,0x02
	Length (parameter length+1)		2 Bytes	u16	0x00,0x1A
	Register		1 Byte	u8	0x4C
Parameters	State		1 Byte	u8	0x00
	Parameter1 (Cartesian offset X=0)		4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter1 (Cartesian offset Y=0)		4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter1 (Cartesian offset Z=-100mm)		4 Bytes	fp32	0x00, 0x00, 0xC8, 0xC2
	Parameter1 (Cartesian offset Roll=-0)		4 Bytes	fp32	0x00, 0x00, 0x80, 0x99
	Parameter1 (Cartesian offset Pitch=-0)		4 Bytes	fp32	0x00, 0x00, 0x00, 0x80
	Parameter1 (Cartesian offset Yaw=0)		4 Bytes	fp32	0x00,0x00,0x00,0x00

Set the self-collision detection function of the robotic arm (/the end tools)				
Register: 77 (0x4D)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x4D

Parameters	Parameter 1 (turn on self-collision detection) 0: turn off self-collision detection 1: turn on self-collision detection	1 Byte	u8	0x01
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x4D
Parameters	State	1 Byte	u8	0x00

The geometric model of the end tool added when setting the self-collision detection				
Register: 78 (0x4E)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x0E **(2+x*4)**
	Register	1 Byte	u8	0x4E
Parameters	Parameter 1 (The end tool is a cuboid) x=20,y=30,z=50 Additional definition parameter area: x maximum is 6, the actual length depends on the number of parameters required by the tool type definition. If there is no parameter, there is no data here. **End tool type:** 1) Custom detection model (additional parameters are required): *Cylinder: Additional definition parameters are: radius (mm), height (mm) *Cuboid: Additional definition parameters are: length[x(mm)] and width[y(mm)], height[z(mm)]consistent with the direction of the default TCP coordinate system 2) Supported detection models (no need to define additional parameters):	**12Bytes (x*4 Byte)**	3*fp32 (x*fp32)	0x00,0x00,0xA0,0x41 0x00,0x00,0xF0,0x41 0x00,0x00,0x48,0x42

	No end tool, xArm gripper, xArm vacuum gripper, xArm BIO gripper, Robotiq 2F-85 gripper, Robotiq 2F-140 gripper			
	Parameter 2 (end tool type number = 22) **End tool type number:** 1) Custom detection models (additional parameters are required): Cylinder: 21 Cuboid: 22 2) Supported detection models (no need to define additional parameters): No end tools: 0 xArm gripper: 1 xArm vacuum gripper: 2 xArm BIO gripper: 3 Robotiq 2F-85 gripper: 4 Robotiq 2F-140 gripper: 5	1 Byte	u8	0x16
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x4E
Parameters	State	1 Byte	u8	0x00

Get the attitude represented by the axis angle attitude **Get the current TCP pose, and use the axial angle to represent the pose of the robotic arm.**				
Register：91 (0x5B)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x5B
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x1A
	Register	1 Byte	u8	0x5B

Parameters	State	1 Byte	u8	0x00
	Parameter1 (Current Cartesian coordinate X=300mm）	4 Bytes	fp32	0x00,0x00,0x96,0x43
	Parameter2 (Current Cartesian coordinate Y=0）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3 (Current Cartesian coordinate Z=150mm）	4 Bytes	fp32	0x00,0x00,0x16,0x43
	Parameter4 (Current Cartesian coordinate Rx=π rad)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5 (Current Cartesian coordinate Ry=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6 (Current Cartesian coordinate Rz=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00

Linear motion with axis angle attitude as target **When planning a linear motion, the target pose is expressed in terms of axial angles, which supports the absolute target pose/relative target pose, as well as the motion options of the base coordinate system/tool coordinate system.**				
Register：92 (0x5C)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x27
	Register	1 Byte	u8	0x5C
Parameters	Parameter1（X=300mm）	4 Bytes	fp32	0x00, 0x00, 0x96, 0x43
	Parameter2（Y=0）	4 Bytes	fp32	0x00, 0x00, 0x00, 0x00
	Parameter3（Z=150mm）	4 Bytes	fp32	0x00, 0x00, 0x16, 0x43
	Parameter4（Rx=π rad）	4 Bytes	fp32	0xDB, 0x0F, 0x49, 0x40
	Parameter5（Ry=0）	4 Bytes	fp32	0x00, 0x00, 0x00, 0x00
	Parameter6（Rz=0）	4 Bytes	fp32	0x00, 0x00, 0x00, 0x00
	Parameter7（motion speed=200 mm/s）	4 Bytes	fp32	0x00, 0x00, 0x48, 0x43
	Parameter8（acceleration=2000mm/s^2）	4 Bytes	fp32	0x00, 0x00, 0xFA, 0x44
	Parameter9（motion time, 0）	4 Bytes	fp32	0x00, 0x00, 0x00, 0x00
	Parameter10（base coordinate system motion） Motion coordinate system： 0: the base coordinate system motion 1: the tool coordinate system motion	1 Byte	u8	0x00

	Parameter11（absolute pose） If the motion coordinate system is the base coordinate system. 0 represents the given pose is an absolute pose 1 represents the given pose is a relative pose (the given parameters 1-6 coordinates are based on the current an offset of position)	1 Byte	u8	0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x5C
Parameters	Parameter1 （Number of commands in the buffer）	2 Bytes	u16	0x00, 0x01

Servo_cartesian motion (axis angle) **An interface for receiving high-frequency continuous Cartesian trajectory motion, and the posture is represented by the axis angle.**				
Register：93 (0x5D)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x26
	Register	1 Byte	u8	0x5D
Parameters	Parameter1（X=300mm）	4 Bytes	fp32	0x00, 0x00, 0x96, 0x43
	Parameter2（Y=0）	4 Bytes	fp32	0x00, 0x00, 0x00, 0x00
	Parameter3（Z=150mm）	4 Bytes	fp32	0x00, 0x00, 0x16, 0x43
	Parameter4（Rx=πrad）	4 Bytes	fp32	0xdb, 0x0f, 0x49, 0x40
	Parameter5（Ry=0）	4 Bytes	fp32	0x00, 0x00, 0x00, 0x00
	Parameter6（Rz=0）	4 Bytes	fp32	0x00, 0x00, 0x00, 0x00
	Parameter7（motion speed=200mm/s）	4 Bytes	fp32	0x00, 0x00, 0x48, 0x43
	Parameter8（acceleration=2000mm/s^2）	4 Bytes	fp32	0x00, 0x00, 0xFA, 0x44
	Parameter9（base coordinate system motion） Motion coordinate system： 0: the base coordinate system motion 1: the tool coordinate system motion	4 Bytes	fp32	0x00, 0x00, 0x00, 0x00

	Parameter10（absolute pose） If the motion coordinate system is the base coordinate system. 0 represents the given pose is an absolute pose 1 represents the given pose is a relative pose (the given parameters 1-6 coordinates are based on the current an offset of position)	1 Byte	u8	0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x5D
Parameters	State	1 Byte	u8	0x00

101~115 Servo Module

Get the state of the current robotic arm servo				
Register：106 (0x6A)				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x6A
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x13
	Register	1 Byte	u8	0x6A
Parameters	Parameter1（Normal） Commands execution state: 0: Normal 1: The server has error message 3: Communication fail	1 Byte	u8	0x00
	Parameter2 (Joint1 servo state)	1 Byte	u8	0x00
	Parameter3 (Joint1 servo error code=Normal)	1 Byte	u8	0x00
	Parameter4 (Joint2 servo state=Normal)	1 Byte	u8	0x00
	Parameter5(Joint2 servo error code=Normal)	1 Byte	u8	0x00
	Parameter6 (Joint3 servo state=Normal)	1 Byte	u8	0x00
	Parameter7 (Joint3 servo error code=Normal)	1 Byte	u8	0x00
	Parameter8 (Joint4 servo state=Normal)	1 Byte	u8	0x00
	Parameter9 (Joint4 servo error code=Normal)	1 Byte	u8	0x00

	Parameter10 (Joint5 servo state=Normal)	1 Byte	u8	0x00
	Parameter11 (Joint5 servo error code=Normal)	1 Byte	u8	0x00
	Parameter12 (Joint6 servo state=Normal)	1 Byte	u8	0x00
	Parameter13(Joint6 servo error code=Normal)	1 Byte	u8	0x00
	Parameter14 (Joint7 servo state=Normal)	1 Byte	u8	0x00
	Parameter15 (Joint7 servo error code=Normal)	1 Byte	u8	0x00
	Parameter16 (Gripper servo state=Normal)	1 Byte	u8	0x00
	Parameter17 (Gripper servo error code=Normal)	1 Byte	u8	0x00

2.1.4. Register (Peripherals Control through Robot IOs)

124: Gripper Module

Enable/Disable the gripper（0x7C）

Set the gripper mode（0x7C）

Set the gripper speed (0x7C)

Set the gripper position（0x7C）

Get the gripper position（0x7C）

Get the gripper error（0x7C）

Clear the gripper error（0x7C）

124~127: RS485 Control on the End-effector

Set the end RS485 baud rate(0x7F)

127~128: IO Control on the End-effector

IO control on the End-effector (0x7F)

Get the input of the end digital quantity（0x80）

Get the input of the end analog（0x80）

130~141: IO Control on the Control Box

Get configurable digital gpio input（0x83）

Get analog input AI1（0x84）

Get analog input AI2（0x85）

Set configurable digital gpio output（0x86）

Set the analog output AO1（0x87）

Set the analog output AO2（0x88）

142~146: Special IO Commands

124 Gripper Module

xArm Gripper fixed parameter explanation:

Parameter	Host ID	Gripper ID	Function Code
Length	1Byte	1Byte	1Byte
Fixed Value	0x09	0x08	0x10

Note:

1. If it is a third-party gripper, the gripper ID and function code are different from the fixed values above.
2. Gripper control is based on RS485 port on the end-effector.

Enable/ Disable the gripper			
Register：124 (0x7C)			
Request			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x0B
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	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
	Register（Enable gripper）	2 Bytes	0x00,0x01
Response			

Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x08
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01,0x00
	Quantity of Registers	2 Bytes	0x00,0x01

Set the gripper mode			
Register：124 (0x7C)			
Request			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x0B
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	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
	Data 0：Position mode 1：Speed mode	2 Bytes	0x00,0x00
Response			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x08
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01,0x00
	Quantity of Registers	2 Bytes	0x00,0x01

Set the gripper speed			
Register：124 (0x7C)			
Request			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02

	Length	2 Bytes	0x00,0x0B
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	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
	Register （Setting the speed to 1500r/min）	2 Bytes	0x05,0xDC
Response			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x08
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x03,0x03
	Quantity of Registers	2 Bytes	0x00,0x01

Set the gripper position			
Register：124 (0x7C)			
Request			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x0D
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	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
	Register（Gripper position=400）	4 Bytes	0x00,0x00,0xC8,0x43
Response			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x08
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09

Modbus RTU Data	Gripper ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x07,0x00
	Quantity of Registers	2 Bytes	0x00,0x02

Get the gripper position			
Register：124 (0x7C)			
Request			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x08
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x07,0x02
	Quantity of Registers	2 Bytes	0x00,0x02
Response			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x08
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x07,0x02
	Quantity of Registers	2 Bytes	0x00,0x02

Get the gripper error			
Register：124 (0x7C)			
Request			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x08
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	1 Byte	0x08
	Function Code	1 Byte	0x03
	Register Starting Address	2 Bytes	0x00,0x0F
	Quantity of Registers	2 Bytes	0x00,0x01
Response			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01

	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x07
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	1 Byte	0x08
	Function Code	1 Byte	0x03
	Byte Count	1 Byte	0x02
	Register Data (No Error)	2 Bytes	0x00,0x00

Clear the gripper error			
Register：124 (0x7C)			
Request			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x0B
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01 0x09
	Quantity of Registers	2 Bytes	0x00 0x01
	Byte Count	1 Byte	0x02
	Register	2 Bytes	0x00 0x01
Response			
Modbus TCP Header	Transaction Identifier	2 Bytes	0x00,0x01
	Protocol	2 Bytes	0x00,0x02
	Length	2 Bytes	0x00,0x08
	Register	1 Byte	0x7C
Internal Use	Host ID	1 Byte	0x09
Modbus RTU Data	Gripper ID	1 Byte	0x08
	Function Code	1 Byte	0x10
	Register Starting Address	2 Bytes	0x01,0x09
	Quantity of Registers	2 Bytes	0x00,0x01

124~127: RS485 Control on the End-effector

Set the end RS485 band rate				
Register: 127 (0x7F)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02

	Length	2 Bytes	u16	0x00,0x08
	Register	1 Byte	u8	0x7F
Parameters	Host ID	1 Byte	u8	0x09
	Address	2 Bytes	u16	0x1A,0x0B
	Parameter1 (2000000bps) 0:4800 bps; 1:9600bps; 2:19200bps; 3:38400bps; 4:57600bps; 5:115200bps 6:230400bps; 7: 460800bps; 8:921600bps; 9: 1000000bps; 10:1500000bps; 11:2000000bps; 12:2500000bps;	4 Bytes	fp32	0x00,0x00,0x30,0x41
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x7F

127~128: IO Control on the End-effector

IO control on the End-effector				
Register：127 (0x7F)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x08
	Register	1 Byte	u8	0x7F
Parameters	Host ID	1 Byte	u8	0x09
	Address	2 Bytes	u16	0x0A,0x15
	Parameters1(Open 0) Data: 256.0: Close 0 257.0: Open 512.0: Close 1 514: Open 1	4 Bytes	fp32	0x00,0x80,0x80,0x43
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x7F
Parameters	State	1 Byte	u8	0x00

Get the input of the end digital quantity				
Register：128 (0x80)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x80
Parameters	Host ID	1 Byte	u8	0x09
	Address	2 Bytes	u16	0x0A, 0x14
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x06
	Register	1 Byte	u8	0x80
Parameters	State	1 Byte	u8	0x00
	Parameters1（0） The end byte indicates the input status. The digit of 0 corresponds to input 0 and the digit of 1 corresponds to input 1.	4 Bytes	u8*4	0x00,0x00,0x00,0x00

Get the input of the end analog				
Register：128 (0x80)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x80
Parameters	Host ID	1 Byte	u8	0x09
	Address(input 0) Address 0a 16 ： input 0 Address 0a 17 ： input 1	2 Bytes	u16	0x0A,0x16
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x06
	Register	1 Byte	u8	0x80
Parameters	State	1 Byte	u8	0x00

	Parameter1（input1） analog input, range 0~4095, corresponding to 0~3.3V	4 Bytes	u32	0x00, 0x00, 0x07, 0x0d

131~140 IO Control on the Control Box

Get configurable digital GPIO input				
Register：131 (0x83)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x83
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x83
Parameters	State	1 Byte	u8	0x00
	Parameters1（The signal of GPIO1 is low） GPIO signal: Bit0 ~ Bit7 Correspond to signals of GPIO0~GPIO7	2 Bytes	u16	0xFF,0xFD

Get analog input AI1				
Register：132 (0x84)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x84
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x84
Parameters	State	1 Byte	u8	0x00

	Parameters1（Analog input0） Analog input0, Range 0~4095 Corresponding to0~10V	2 Bytes	u16	0x00,0x12

Get analog input AI2				
Register：132 (0x85)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x85
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x85
Parameters	State	1 Byte	u8	0x00
	Parameters1（Analog input1） Analog input1, Range 0~4095 Corresponding to0~10V	2 Bytes	u16	0x00,0x15

Set configurable digital GPIO output				
Register：134 (0x86)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x03
	Register	1 Byte	u8	0x86
Parameters	Parameters1(The signal of GPIO7 is low) GPIO signal: the upper 8 bits are the enable bits, and the lower 8 bits are the set bits	2 Bytes	u16	0x80,0x00
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x86
Parameters	State	1 Byte	u8	0x00

Set the analog output AO1				
Register：135 (0x87)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x03
	Register	1 Byte	u8	0x87
Parameters	Parameters1(Analog output 0 is 0) Analog output0, Range 0~4095 Corresponding to 0~10V	2 Bytes	u16	0x00,0x00
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x87
Parameters	State	1 Byte	u8	0x00

Set the analog output AO2				
Register：136 (0x88)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x03
	Register	1 Byte	u8	0x88
Parameters	Parameters1(Analog output 1 is 0) Analog output 1, Range 0~4095 Corresponding to 0~10V	2 Bytes	u16	0x00,0x00
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x88
Parameters	State	1 Byte	u8	0x00

Configure digital output IO function				
Register：138 (0x8A)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x03
	Register	1 Byte	u8	0x8A

	Parameters1（GPIO7） GPIO serial number,0~7 Corresponding to GPIO0 ~ GPIO7	1 Byte	u8	0x07
	Parameters2（‘STOP’ state） Function number 0: System in ‘STOP’ state 1: System error 2: xArm is operating	1 Byte	u8	0x00
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x8A
Parameters	State	1 Byte	u8	0x00

Get GPIO state				
Register：139 (0x8B)				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x01
	Register	1 Byte	u8	0x8B
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x24
	Register	1 Byte	u8	0x8B
Parameters	State	1 Byte	u8	0x00
	GPIO Module status 0: Normal 3: Gripper has error message 6: Communication failure	1 Byte	u8	0x00
	GPIO module error code 0: Normal Not 0: Error code	1 Byte	u8	0x00
	Digital input function IO status	2 Bytes	u16	0x01,0x00
	Digital input configuration IO status	2 Bytes	u16	0xFF,0xFD
	Digital output function IO status	2 Bytes	u16	0x00,0x00

	Digital output configuration IO status	2 Bytes	u16	0xFF,0x00
	Analog input 1	2 Bytes	u16	0x00,0x11
	Analog input 2	2 Bytes	u16	0x00,0x15
	Analog output 1	2 Bytes	u16	0x00,0x00
	Analog output 2	2 Bytes	u16	0x00,0x00
	Digital input IO configuration message	1 Byte*8	u8*8	0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00
	Digital output IO configuration message	1 Byte*8	u8*8	0x00,0x00,0x00,0x00,0x00, 0x00,0x00,0x00

142~147: Special IO commands

Operation of general digital IO delay output of control box **Starting from the moment when the command is issued, the digital output switch of the control box is triggered after a period of time.**				
Register142（0x8E）				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x07
	Register	1 Byte	u8	0x8E
	Parameters1(0) Digital IO port number of control box (0-7)	1 Byte	u8	0x00
	Parameters2(on) Switch value (0 is off, 1 is on)	1 Byte	u8	0x01
	Parameters3 (The time when the delay takes effect from the current time=3s)	4 Bytes	fp32	0x00,0x00,0x40,0x40
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x8E
Parameters	State	1 Byte	u8	0x00

Operation of the end general digital IO delay output **Starting from the moment when the command is issued, the end digital output switch is triggered after a period of time.**				
Register143（0x8F）				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01

	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x07
	Register	1 Byte	u8	0x8F
	Parameters1(0) The end digital IO port number of control box (0/1)	1 Byte	u8	0x00
	Parameters2(on) Switch value (0 is off, 1 is on)	1 Byte	u8	0x01
	Parameters3 (The time when the delay takes effect from the current time=3s)	4 Bytes	fp32	0x00,0x00,0x40,0x40
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x8F
Parameters	State	1 Byte	u8	0x00

Operation triggered by the position of the general digital IO of the control box **Starting from the moment when the instruction is issued, the TCP triggers the digital output switch of the control box after it reaches the specified position area, which is valid for a single time.**				
Register144（0x90）				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x13
	Register	1 Byte	u8	0x90
	Parameters1(0) IO port number of the control box: 0-7	1 Byte	u8	0x00
	Parameters2(on) Switch value (on_off): 0 is off, 1 is on	1 Byte	u8	0x01
	Parameters3（x=400mm）	4 Bytes	fp32	0x00,0x00,0xc8,0x43
	Parameters4（y=0mm）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameters5（z=200mm）	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameters6 Tolerance radius (tol_r=50mm), when the robotic arm reaches the specified position (the area of the sphere specified by the trigger position point (x, y, z) as the center (the radius of the sphere is the	4 Bytes	fp32	0x00,0x00,0x48,0x42

	tolerance radius)), trigger IO . If the tolerance radius is not set, when the robotic arm passes the specified point at a speed other than 0, it may cause a missed			
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x90
Parameters	State	1 Byte	u8	0x00

Operation triggered by the position of the end general digital IO **Starting from the moment when the instruction is issued, the TCP triggers the end digital output switch after it reaches the specified position area, which is valid for a single time.**				
Register145（0x91）				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x13
	Register	1 Byte	u8	0x91
	Parameters1(0) IO port number of the end: 0/1	1 Byte	u8	0x00
	Parameters2(on) Switch value (on_off): 0 is off, 1 is on	1 Byte	u8	0x01
	Parameters3（x=400mm）	4 Bytes	fp32	0x00,0x00,0xc8,0x43
	Parameters4（y=0mm）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameters5（z=200mm）	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameters6 Tolerance radius (tol_r=50mm) when the robotic arm reaches the specified position (the area of the sphere specified by the trigger position point (x, y, z) as the center (the radius of the sphere is the tolerance radius)), trigger IO . If the tolerance radius is not set, when the robotic arm passes the specified point at a speed other than 0, it may cause a missed trigger because it cannot be accurately detected.	4 Bytes	fp32	0x00,0x00,0x48,0x42
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01

	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x91
Parameters	State	1 Byte	u8	0x00

Whether the control box and terminal IO are automatically cleared in the STOP state				
Register146（0x92）				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x03
	Register	1 Byte	u8	0x92
	Parameters1(the control box IO) IO type 0 represents the control box IO 1 represents the end IO	1 Byte	u8	0x00
	Parameters2(on) Switch value 0 is off, the STOP status is not cleared. 1 is on, and the STOP status is cleared.	1 Byte	u8	0x01
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x92
Parameters	State	1 Byte	u8	0x00
	Parameter1	2 Bytes	u16	0x00,0x01

Operation triggered by the position of the general Analog IO of the control box **Starting from the moment when the command is issued, the TCP triggers the analog output switch of the control box after it reaches the specified position area, which is valid for a single time.**				
Register147（0x93）				
Request				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x14
	Register	1 Byte	u8	0x93

	Parameters1(0) IO port number of the control box: 0/1	1 Byte	u8	0x00
	Parameters2(on) Parameters1(Analog output 0 is 0) Analog output 0, Range 0~4095 Corresponding to 0~10V	2 Byte	u16	0x00,0x00
	Parameters3（x=400mm）	4 Bytes	fp32	0x00,0x00,0xc8,0x43
	Parameters4（y=0mm）	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameters5（z=200mm）	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameters6 Tolerance radius (tol_r=50mm), when the robotic arm reaches the specified position (the area of the sphere specified by the trigger position point (x, y, z) as the center (the radius of the sphere is the tolerance radius)), trigger IO . If the tolerance radius is not set, when the robotic arm passes the specified point at a speed other than 0, it may cause a missed	4 Bytes	fp32	0x00,0x00,0x48,0x42
Response				
Modbus TCP Header	Transaction Identifier	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x93
Parameters	State	1 Byte	u8	0x00

2.1.5. Modbus TCP Example

If you want the robotic arm to perform a basic motion, please send the commands as follows:

(1) Enable the robotic arm.

(2) Set the motion mode of the robotic arm.

(3) Set the motion state of the robotic arm.

(4) Send motion commands.

The following will give an example according to the above steps:

Function	Enable the robotic arm	Setting mode	Setting state	Cartesian linear motion

Note:

(1) 3.2.4 has a detailed description of the register list.

(2) Please refer to P31-P32 for the format of the request and response command parameters in the following examples.

(3) The following explains some of the symbols used in the examples and tables:

u8 (1 Byte, 8-bit unsigned int)

u16 (2 Bytes, 16-bit unsigned int, big-endian analysis)

fp32 (4 Bytes, float, little-endian analysis)

str (string)

Enable the robotic arm				
Register11（0x0B）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x03
	Register	1 Byte	u8	0x0B
	Parameter1(servo_id)	1 Byte	u8	0x08
	Parameter2(enable)	1 Byte	u8	0x01
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x0B
Parameters	State	1 Byte	u8	0x00

Setting mode				
Register19（0x13）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x13
	Parameter1(Motion mode)	1 Byte	u8	0x00
Response				

Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x13
Parameters	State	1 Byte	u8	0x00

Setting state				
Register12（0x0C）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x0C
	Parameter1(Motion state)	1 Byte	u8	0x00
Response				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x02
	Register	1 Byte	u8	0x0C
Parameters	State	1 Byte	u8	0x00

Cartesian linear motion				
Register21（0x15）				
Request				
Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x25
	Register	1 Byte	u8	0x15
Parameters	Parameter1(x=400mm)	4 Bytes	fp32	0x00,0x00,0xC8,0x43
	Parameter2(y=0mm)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter3(z=200mm)	4 Bytes	fp32	0x00,0x00,0x48,0x43
	Parameter4(roll=π)	4 Bytes	fp32	0xDB,0x0F,0x49,0x40
	Parameter5(pitch=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter6(yaw=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
	Parameter8(speed=100mm/s)	4 Bytes	fp32	0x00,0x00,0xC8,0x42
	Parameter9(acceleration=2000mm/s	4 Bytes	fp32	0x00,0x00,0xFA,0x44
	Parameter10(motion time=0)	4 Bytes	fp32	0x00,0x00,0x00,0x00
Response				

Modbus TCP Header	Transaction ID	2 Bytes	u16	0x00,0x01
	Protocol	2 Bytes	u16	0x00,0x02
	Length (parameter length+1)	2 Bytes	u16	0x00,0x04
	Register	1 Byte	u8	0x15
Parameters	State	1 Byte	u8	0x00
	Parameter1	2 Bytes	u16	0x00,0x01

2.1.6. Automatic Reporting Format

REPORT_TCP_DEVELOP:

REPORT_TCP_DEVELOP			
Default Port	30003		
Frequency	100Hz		
Byte Order Content	1~4 Bytes		Number of Bytes
	5 Byte	u8	Bit0-Bit3 indicates the motion status, Bit4-Bit7indicates the motion mode.
	6~7 Bytes	u16	Number of commands Caches, big-endian byte order
	8~35 Bytes	fp32	Current angle of the robotic arm
	36~59 Bytes	fp32	Current position of the robotic arm
	60~87 Bytes	fp32	Joint torque
Example			
Assumption: Get 36-50 Bytes of data	0x18,0x00,0x4F,0x43,0x24,0xFC,0x8A,0x28,0x08,0x01,0xE0,0x42 0xDB,0x0F,0x49,0xC0,0x00,0x00,0x00,0x24,0x00,0x00,0x00,0x00,		
Analysis Results	0x18,0x00,0x4F,0x43	207.0003662109375	
	0x24,0xFC,0x8A,0x28	1.54304263051859e-14	
	0x08,0x01,0xE0,0x42	112.00201416015625	
	0xDB,0x0F,0x49,0xC0	3.1415927410125732	
	0x00,0x00,0x00,0x24	2.7755575615628914e-17	
	0x00,0x00,0x00,0x00	0.0	

REPORT_TCP_NORMAL:

REPORT_TCP_NORMAL			
Default Port	30001		
Frequency	5Hz		
Byte Order Content	1~87Bytes		The same as [the Auto Reporting Format of REPORT_TCP_DEVELOP]
	88 Bytes	u8	Servo brake status (u8 Bit0 ~ Bit correspond to 1~6 joints respectively, 0 not enabled, 1 enabled)
	89 Bytes	u8	Servo brake status (u8 Bit0 ~ Bit correspond to 1~6 joints respectively, 0 not enabled, 1 enabled)
	90 Bytes	u8	Error code

	91 Bytes	u8	Warning code
	92~115 Bytes	fp32 *6	TCP offset, little-endian byte order
	116~131Bytes	fp32 *4	End load Parameter
	132 Bytes	u8	Collision detection sensitivity
	133 Bytes	u8	Teaching sensitivity
	134~145 Bytes	fp32 *3	Vectors (x, y, z) indicating the direction of gravity, relative to the base coordinate system.
Example			
The same as [REPORT_TCP_DEVELOP]			

REPORT_TCP_RICH:

REPORT_TCP_RICH			
Default Port	30002		
Frequency	5Hz		
Byte Order Content	1~145 Bytes		The same as [the Auto Reporting Format of REPORT_TCP_DEVELOP]
	146 Bytes	u8	Robotic arm type number（5/6/7)
	147 Bytes	u8	Robotic arm joint number（5/6/7)
	148 Bytes	u8	MASTER ID Communication (0xAA fixed）
	149 Bytes	u8	SLAVE ID Communication （0x55 fixed）
	150 Bytes	0	Reserved
	151 Bytes	0	Reserved
	152~181Bytes bytes		Firmware version string (30 Bytes）
	182~201 Bytes	fp32 *5	[current cartesian jerk (mm / s^3), (configurable)minimum cartesian acceleration (mm / s^2), (configurable)maximum cartesian acceleration (mm / s2), (configurable)minimum cartesian speed (mm / s), (configurable)maximum cartesian speed (mm / s)]
	202~221 Bytes	fp32 *5	[current joint jerk (radian / s^3), (configurable)minimum joint acceleration (radian / s^2), (configurable)maximum joint acceleration (radian / s^2), (configurable)minimum joint speed (radian / s), (configurable)maximum joint speed (radian / s)]
	222~229 Bytes	fp32 *2	[Attitude rotation jerk (radian / s^3), maximum attitude rotation acceleration(radian / s^2)] Note: Users cannot set the above two parameter values by yourselves
	230~243 Bytes	u8	[Joint servo error type, joint servo error code]
	244~245 Bytes	u8	[End IO error type, end IO error code]
	246~252 Bytes	u8	[Joint Celsius]
	253~256 Bytes	fp32	TCP line speed (mm / s) of the current cartesian planning commands
	257~284 Bytes	fp32 * 7	The angular velocity of each joint of the current joint

	285~288 Bytes	u32	The value of the current commands counter
	289~312 Bytes	fp32 * 6	User coordinate system offset [x (mm), y (mm), z (mm), roll (radian), pitch (radian), yaw (radian)]
	313 Bytes	u8	The switch value of the control box IO stop state
	314 Bytes	u8	The switch value of the end IO stop state clearing
	315 Bytes	u8	Virtual control switch
	316 Bytes	u8	Self-collision detection switch
	317 Bytes	u8	Self-collision detection end tool type number
	318~341Bytes	fp32 * 6	Self-collision detection end tool model parameters, unit: mm, little-endian byte order
	342~355Bytes	u16*7	Robotic arm joint voltage (value has been processed by X100)
	356~383 Bytes	fp32 * 7	Joint current, unit: A
	384Bytes	u8	GPIO module status (refer to Register 139) 0: normal 3: The paw has an error message 6: Communication failed
	385 Bytes	u8	Error code of GPIO module (refer to Register 139) 0: normal Non-zero: error code
	386~387 Bytes	u16	Digital input function IO status (refer to Register 139)
	388~389 Bytes	u16	Digital input configuration IO status (refer to Register 139)
	390~391 Bytes	u16	Digital output function IO status (refer to Register 139)
	392~393 Bytes	u16	Digital output configuration IO status (refer to Register 139)
	394~395 Bytes	u16	Analog input 1 (refer to Register 139)
	396~397 Bytes	u16	Analog input 2 (refer to Register 139)
	398~399 Bytes	u16	Analog output 1 (refer to Register 139)
	400~401Bytes	u16	Analog output 2 (refer to Register 139)
	402~409 Bytes	u8*8	Digital input IO configuration information (refer to Register 139)
	410~417 Bytes	u8*8	Digital output IO configuration information (refer to Register 139)
Example			
The same as [REPORT_TCP_DEVELOP]			

3. Error Reporting and Handling

3.1. Joints Error Message and Error Handling

- Error processing method: Re-power on, the steps are as follows:
    1. Turn the emergency stop button on the control box
    2. Enable robotic arm
- xArm Studio enable mode: Click the guide button in the error pop-up window or the [Enable Robot] button on the homepage.
- xArm-Python-SDK enable mode: Error Handling Mode.
- xArm-library: operators 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.

Software Error Code	Error Code	Error Handling
S10	0x0A	Current Detection Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S11	0x0B	Joint Overcurrent Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S12	0x0C	Joint Overspeed Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S14	0x0E	Position Command Overlimit Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S15	0x0F	Joints Overheat If the robotic arm is running for a long time, please stop running and restart the xArm after it's cool down.
S16	0x10	Encoder Initialization Error Please ensure that there is no external force to push the robotic arm when the it's energized. Please restart the xArm with the Emergency Stop Button on the xArm

		Control Box.
S17	0x11	Single-turn Encoder Error Please re-enable the robot
S18	0x12	Multi-turn Encoder Error Please contact technical support.
S19	0x13	Low Battery Voltage Please contact technical support.
S20	0x14	Driver IC Hardware Error Please re-enable the robot.
S21	0x15	Driver IC Initialization Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S22	0x16	Encoder Configuration Error Please contact technical support.
S23	0x17	Large Motor Position Deviation Please check whether the xArm movement is blocked, whether the payload exceeds the rated payload of xArm, and whether the acceleration value is too large.
S26	0x1A	Joint N Positive Overrun Please check if angle value of the joint N is too large.
S27	0x1B	Joint N Negative Overrun Please check if the angle value of joint N is too large, if so, please click Clear Error and manually unlock the joint and rotate the joint to the allowed range of motion.
S28	0x1C	Joint Commands Error The xArm is not enabled, please click Enable Robot.
S33	0x21	Drive Overloaded Please make sure the payload is within the rated load.
S34	0x22	Motor Overload Please make sure the payload is within the rated load.
S35	0x23	Motor Type Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S36	0x24	Driver Type Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S39	0x27	Joint Overvoltage Please reduce the acceleration value in the Motion Settings.
S40	0x28	Joint Undervoltage Please reduce the acceleration value in the Motion Settings.

		Please check if the control box emergency stop switch is released.
S49	0x31	EEPROM Read and Write Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
S52	0x34	Initialization of Motor Angle Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.

3.2. Control Box Error Code and Error Handling

3.2.1. Control Box Error Code

If there is any error in the hardware of the robotic arm in the software of the Control Box/in sending command, an error or warning will be issued. This error/warning signal will be fed back when the operators send any command; In other words, the feedback is passive and not actively reported.

After the above error occurs, the robotic arm will stop working immediately and discard the Control Box 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.

Software Error Code	Error Code	Error Handling
C1	0x01	The Emergency Stop Button on the Control Box is Pushed in to Stop Please release the Emergency Stop Button, and then click "Enable Robot"
C11-C17	0x0B-0x11	Power on again.
C19	0x13	Gripper Communication Error Please check if the Gripper is installed or the Gripper is installed correctly, or restart the xArm with the Emergency Stop Button on the xArm Control Box.
C21	0x15	Kinematic Error Please re-plan the path.

C22	0x16	Self-collision Error, Please Re-plan the Path. If the robotic arm continues to report self-collision errors, please go to the "live control" interface to turn on the "manual mode" and drag the robotic arm back to the normal position.
C23	0x17	Joints Angle Exceed Limit Please click the "ZERO" button to return to the zero pozition.
C24	0x18	Speed Exceeds Limit Please check if the xArm is at singularity point, or reduce the speed and acceleration values.
C25	0x19	Planning Error Please re-plan the path or reduce the speed.
C26	0x1A	Linux RT Error Please contact technical support.
C27	0x1B	Command Reply Error Pleas retry, or restart the xArm with the Emergency Stop Button on the xArm Control Box.
C28	0x1C	End Module Communication Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
C29	0x1D	Other Errors Please contact technical support.
C30	0x1E	Feedback Speed Exceeds limit Please contact technical support.
C31	0x1F	Collision Caused Abnormal Current Please check for collisions, check that the payload settings are correct, and that the collision sensitivity matches the speed.
C32	0x20	Three-point drawing circle calculation error Please reset the arc command.
C33	0x21	Control Box GPIO Error If the error occurs repeatedly, please contact technical support.
C34	0x22	Recording Timeout The track recording duration exceeds the maximum duration limit of 5 minutes. It is recommended to re-record.
C35	0x23	Safety Boundary Limit The xArm reaches the safety boundary. Please let the xArm work within the safety boundary.
C36	0x24	The number of delay commands exceeds the limit The number of delay IO commands or position

		detection IO commands to be executed cannot exceed 36, please check whether there are too many delay commands or position detection IO commands in the code.
C37	0x25	Abnormal Motion in Manual Mode Please check whether the TCP payload setting of the robotic arm and the installation method of the robotic arm match the actual settings.
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.		

3.2.2. Control Box Error Code

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

Error code	Description	Error Handling
11 (0x0B)	Buffer overflow	Control the volume of command
12 (0x0C)	Command parameter abnormal	Check sent command
13 (0x0D)	Unknown Command	Check sent command
14 (0x0E)	Command no solution	Check sent command

3.3. Gripper Error Code & Error Handling

Operators can power off and on the system as an error handling, 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 control box.
2. Enable robotic arm.

a. xArm Studio enable mode:Click the guide button in the error pop-up window or the [Enable Robot] button on the homepage.

b. xArm-Python-SDK enable mode: xArm-Python-SDK Error Handling.

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 power on/off for multiple times, please contact UFACTORY team for support.

Software Error Code	Error Code	Error Handling
G9	0x09	Gripper Current Detection Error Please restart the xArm with the Emergency Stop Button on the xArm Control Box.
G11	0x0B	Gripper Current Overlimit Please click "OK" to re-enable the Gripper.
G12	0x0C	Gripper Speed Overlimit Please click "OK" to re-enable the Gripper.
G14	0x0E	Gripper Position Command Overlimit Please click "OK" to re-enable the Gripper.
G15	0x0F	Gripper EEPROM Read and Write Error Please click "OK" to re-enable the Gripper.
G20	0x14	Gripper Driver IC Hardware Error Please click "OK" to re-enable the Gripper.
G21	0x15	Gripper Driver IC Initialization Error Please click "OK" to re-enable the Gripper.
G23	0x17	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	0x19	Gripper Command Over Software Limit Please check if the gripper command is set beyond the software limit.
G26	0x1A	Gripper Feedback Position Software Limit Please contact technical support.
G33	0x21	Gripper Drive Overloaded Please contact technical support.
G34	0x22	Gripper Motor Overload Please contact technical support.
G36	0x24	Gripper Driver Type Error Please click "OK" to re-enable the Gripper.

xArm-Python-SDK Error Handling:

When designing the robotic arm motion path with the Python library, if the robotic arm error (see Appendix for Alarm information) occurs, it needs to be cleared manually. After clearing the error, the robotic arm should be motion enabled.

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)

4. Technical Specifications

4.1. xArm5/6/7 Common Specifications

xArm		
Cartesian Range	X	±700mm
	Y	±700mm
	Z	-400mm~951.5mm
	Roll/Yaw/Pitch	± 180°
Maximum Joint Speed		180°/s
Reach		700mm
Repeatability		±0.1mm
Max Speed of End-effector		1m/s
*Ambient Temperature Range		0-50 °C*
Power Consumption		Min 8.4 W, Typical 200W, Max 400W
Input Power Supply		24 V DC, 16.5 A
ISO Class Cleanroom		5
Robotic Arm Mounting		Any
Programming		xArm Studio/Python/C++/ROS
Robotic Arm Communication Protocol		Modbus-TCP
End-effector I/O Interface		2 Digital inputs, 2 Digital outputs, 2 Analog inputs
End-effector Communication Protocol		Modbus-RTU
Footprint		Ø 126 mm
Materials		Aluminium, Carbon Fiber
End Tool Flange		DIN ISO 9409-1-A50/63（M5*6）

Control Box		
	AC Control Box	DC Control Box
Input	100-240VAC 50/60Hz	24VDC
Output	24VDC 16.5A	
Control Box Communication Protocol	Modbus TCP	
Control Box Communication Model	Ethernet	
Control Box I/O Interface	8*CI(Digital In) 8*CO(Digital Out) 2*AI(Analog In) 2*AO(Analog Out)	
Weight	3.8kg	1.6kg
Dimension(L*W*H)	280*200*116mm	180*145*68mm

xArm accessories parameters：

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	86mm
Maximum Clamping Force	30N

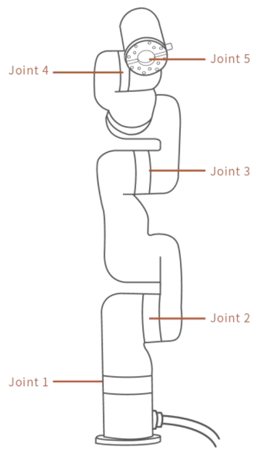
Weight (g)	822g
Communication Mode	RS-485
Communication Protocol	Modbus RTU
Programmable Gripping Parameters	Position, Speed
Feedback	Position
Vacuum Gripper	
Rated Supply Voltage	24V DC
Absolute Maximum Supply Voltage	28V DC
Quiescent Current(mA)	30mA
Peak Current(mA)	400mA
Vacuum	78%
Vacuum Flow (L/min)	> 5.6L/min
Weight (g)	610 g
Dimensions (L*W*H)	122.5 * 91.6 * 75mm
Payload (kg)	≤5kg
Noise Level (30cm away)	< 60dB
Communication Mode	Digital IO
State Indicator	Power, Working State
Feedback	Air Pressure (Low or Normal)

Notes:

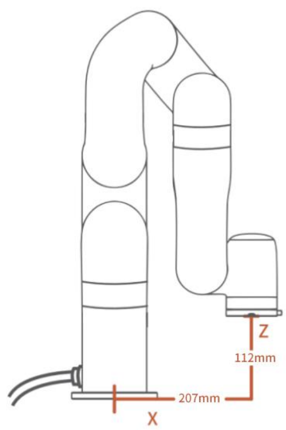
1. The ambient temperature of xArm is 0-50 °C, please reduce the temperature if continuous high-speed operation is needed.

4.2. xArm 5 Specifications

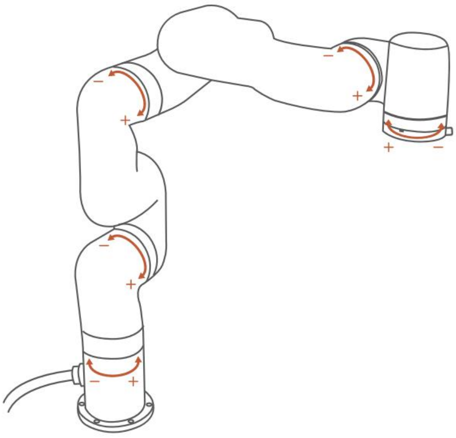
Joint Range	1,5	±360°
	2	-118°~120°
	3	-225°~11°
	4	-97°~180°
Payload		3kg
Degrees of Freedom		5
Weight(robotic arm only)		11.2kg



Robot Joints



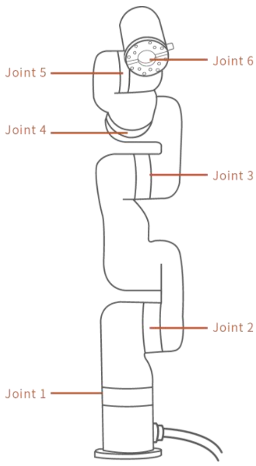
Robot Zero Attitude



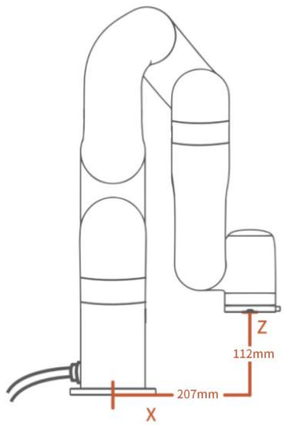
Joint Rotating Direction

4.3. xArm 6 Specifications

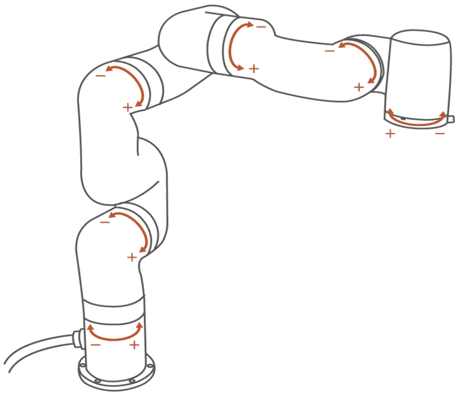
Joint Range	1,4,6	±360°
	2	-118°~120°
	3	-225°~11°
	5	-97°~180°
Payload		5kg
Degrees of Freedom		6
Repeatability		±0.1mm
Weight(robotic arm only)		12.2kg



Robot Joints

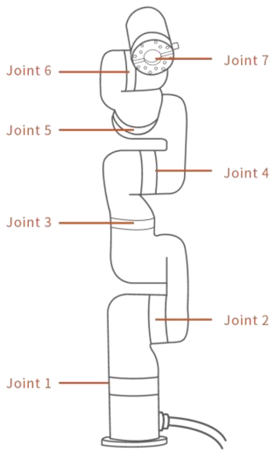
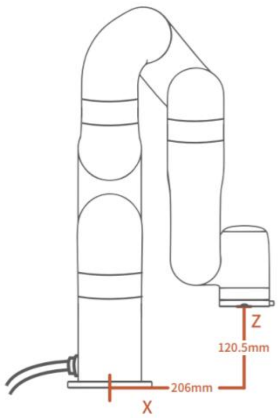


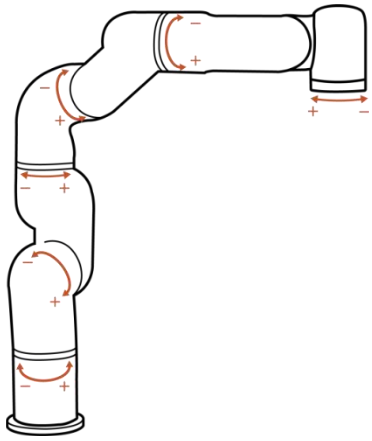
Robot Zero Attitude



Joint Rotating Direction

4.4. xArm 7 Specifications

Joint Range	1,3,5,7	±360°
	2	-118°~120°
	4	-11°~225°
	6	-97°~180°
Payload		3.5kg
Degrees of Freedom		7
Weight(robotic arm only)		13.7kg
 Robot Joints		 Robot Zero Attitude



Joint Rotating Direction