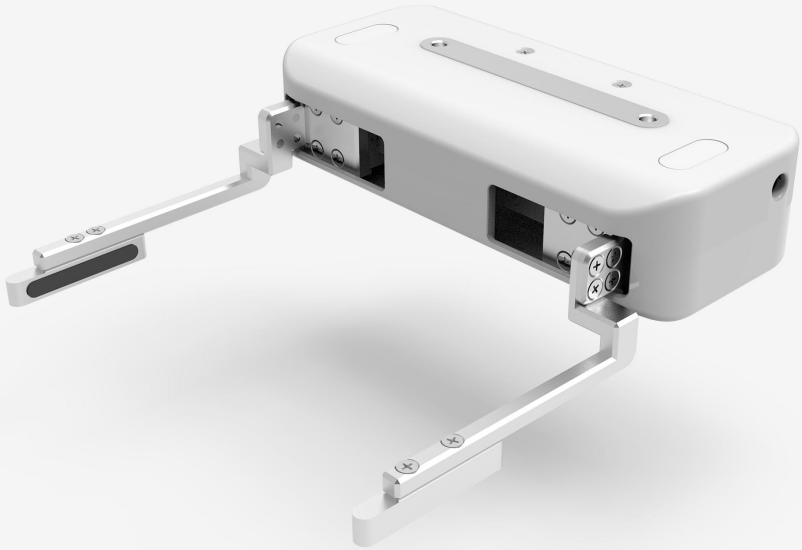
众为创造
UFACTORY

UFACTORY xARM

BIO机械爪用户手册



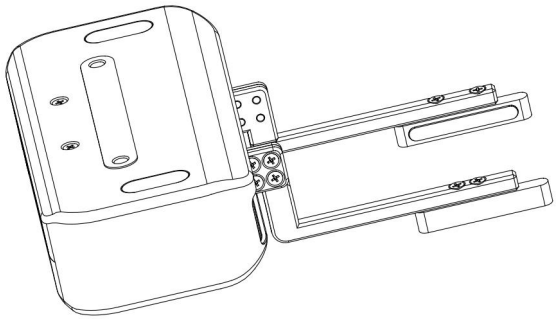
使用前请仔细阅读本手册

V 2.3.0

目录

1.总体介绍

1.1. 机械爪简要介绍



BIO 机械爪

BIO 机械爪是专为处理液体板而设计的平行夹爪，具有快速安装以及简单应用的特点。其手指具有出色的抓握能力以及灵活性。

BIO 机械爪的主要特点：

（1）专为液体处理而设计

官方安装机械爪的最大开合为 150mm，最小开合为 70mm。特殊设计的指尖非常适合用于抓取液体板。

（2）可自定义的手指

机械爪的手指可自由定制，用于处理不同形状的管状和板状的产品。

（3）与 xArm 高度集成

BIO 机械爪是专为 xArm 设计的，可以通过 xArm 工具末端的 IO 端口进行控制，无需使用外部电缆和连接器，可以确保整个集成稳定且安全。

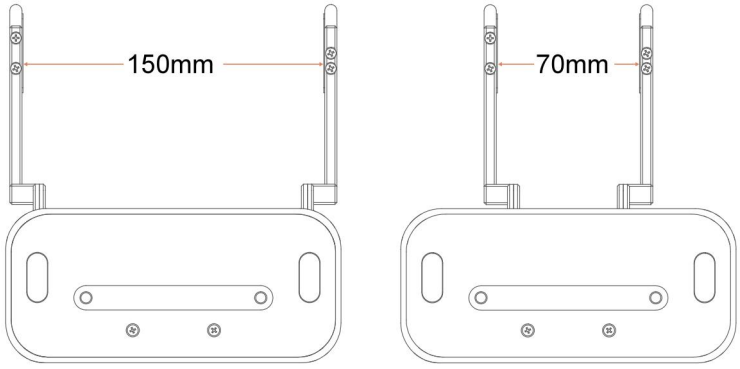
注意： BIO 机械爪没有“位置控制”功能，

1.2. 物体拾取

BIO 机械爪的手指采用平行抓握，如下图所示。

机械爪可抓握物体的宽度：70-150mm。

注意：客户可以根据自己的需求安装自定义的夹具，开合的大小会有变化。



(1)最大行程　　(2）最小行程

1.3. 设置与控制

BIO 机械爪通过一条电缆直接供电和控制，该电缆用于 24V 直流供电和基于 RS-485 的 Modbus RTU 通信。

1.4. 安全

警告

操作员在使用 BIO 机械爪之前必须已阅读并理解手册中的所有说明。

注意

术语“操作员”是指负责在 BIO 机械爪上进行以下任何操作的任何人：

- 安装

- 控制
- 维护
- 检查
- 标定
- 编程
- 退役

本文档说明了 BIO 机械爪从安装到运行再到使用的整个生命周期的一般操作。

本文档中的图形和照片是代表性的示例，它们与交付的产品之间可能存在差异。

1.4.1. 警告

注意

不遵守警告而使用抓取工具，可能导致操作人员受伤或设备损坏。

警告

在操作机器人之前，必须正确固定好夹具。

请勿安装或操作已损坏或缺少零件的机械爪。

切勿为机械爪通交流电。

确保所有接线端子稳定连接在机械臂和机械爪两端。

请始终使用建议的电气连接。

在初始化机械臂程序之前，请确保没有人在机械臂和机械爪路径中。

始终不要超过机械爪的有效载荷。

根据您的应用情况，相应设置机械爪的速度。

接通电源时，手指和衣服应远离机械爪。

请勿在人或动物身上使用机械爪。

1.4.2. 风险评估和最终应用

BIO 机械爪用于工业机器人，最终应用中使用的机器人、抓取器和任何其他设备必须进行风险评估。机器人集成商的责任是确保遵守所有本地安全措施和规定。根据不同的应用，可能存在需要采取额外保护/安全措施的风险，例如，机械爪操作的工件可能对操作员具有固有的危险。

1.4.3. 用途

BIO 机械爪用于抓取并临时固定或保持物体。

警告

机械爪不适用于对物体或表面施加力。

该产品旨在安装在机器人或其他自动化设备上。

信息

始终遵守有关自动化安全和通用机器安全的本地和国家法律，法规和指令。

本设备只能在其技术数据范围内使用。产品的任何其他使用均被视为不当和意外使用。

对于因任何不当使用或不当使用引起的任何损坏，UFACTORY 将不承担任何责任。

2.安装

以下小节将指导您完成 BIO 机械爪的安装和常规设置。

（1）发货清单

（2）机械安装部分

（3）电气设置部分

警告

安装之前：

阅读并理解与 BIO 机械爪有关的安全说明。

根据发货清单和订单验证包裹。

备有需求中列出的所需零件。

安装时：

满足环境条件。

在牢固地固定住机械爪并清除危险区域之前，请勿操作机械爪或打开电源。

机械爪的手指可能会移动并造成伤害或损坏。

2.1. 发货清单

2.1.1. 通用套件

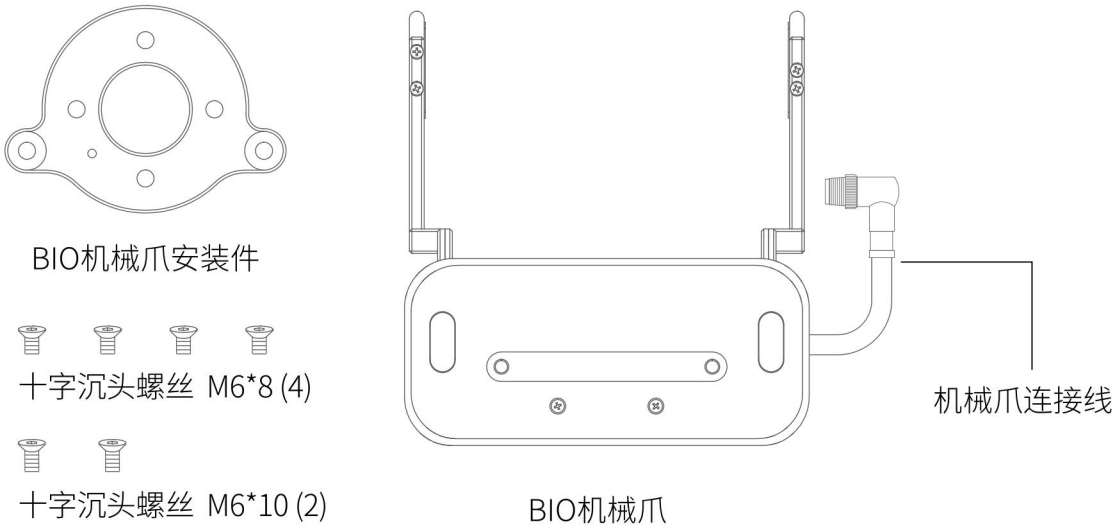
BIO 机械爪套件通常包括以下物品（如下图所示）：

BIO 机械爪

BIO 机械爪安装件

十字沉头螺丝 M6*8(4 个)

十字沉头螺丝 M6*10(2 个)



通用套件图

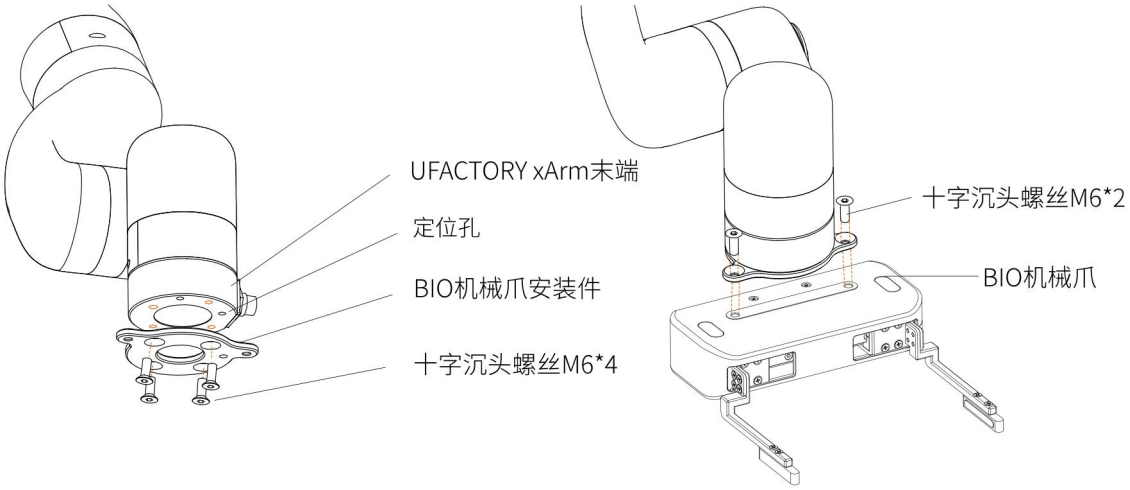
2.2. 机械安装

BIO 机械爪安装步骤（如下图所示）：

（1）将 BIO 机械爪安装板用螺丝固定到机械臂的工具末端，注意将安装板上的定位柱对齐机械臂末端的定位孔。

（2）将 BIO 机械爪用螺丝固定到 BIO 机械爪安装板上。

（3）用机械爪连接线将机械臂与 BIO 机械爪连接。



BIO 机械爪安装步骤图

注意：

1） 接通机械爪连接线时一定要使机械臂断电，急停开关处于按下状态，机械臂电源指示灯 熄灭，避免热插拔引起机械臂故障；

2） 因机械爪连接线长度限制，机械爪接口与末端接口需在相同的方向；

3） 机械爪连接线接通机械爪跟机械臂时注意务必对齐两端接口的定位孔，连接线的公针较为纤细，避免在拆装时使公针弯曲。

2.3. 电气设置

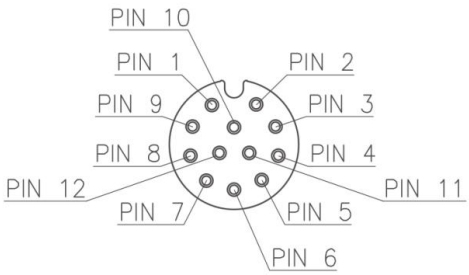
BIO 机械爪通过单根设备电缆与 xArm 机械臂建立电源和通信。设备电缆为机械爪提供 24V 电源，并实现了与机械臂控制器的串行 RS485 通信。

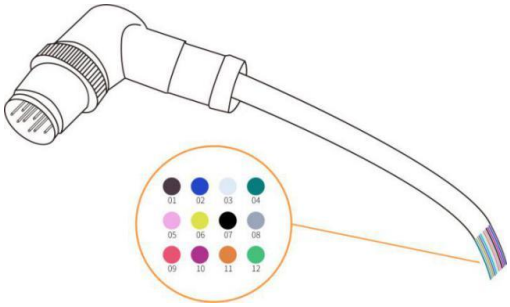
警告：

请断开机械臂电源后，再用机械爪连接线将机械爪和机械臂连接在一起。

2.3.1. 引脚接口

机械爪通过位于其外表面上的 12pin 接头与机械臂工具端连接。





线序	颜色	信号
1	棕	+24V（电源）
2	蓝	+24V（电源）
3	白	0V （GND）
4	绿	0V （GND）
5	粉	用户 485-A
6	黄	用户 485-B
7	黑	工具输出 0 （TO0）
8	灰	工具输出 1 （TO1）
9	红	工具输入 0 （TI0）
10	紫	工具输入 1 （TI1）
11	橙	模拟输入 0 （AI0）
12	浅绿	模拟输入 1 （AI1）

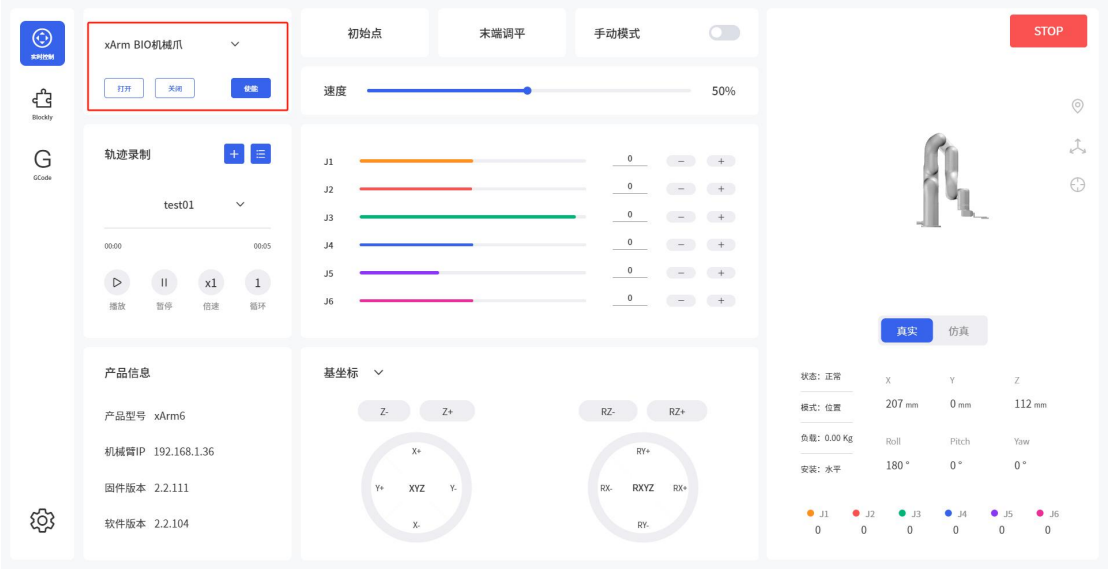
3.BIO 机械爪的控制方式

3.1. 用 xArm Studio 控制 BIO 机械爪

1. 设置 BIO 机械爪

进入【实时控制】-【选择末端工具】

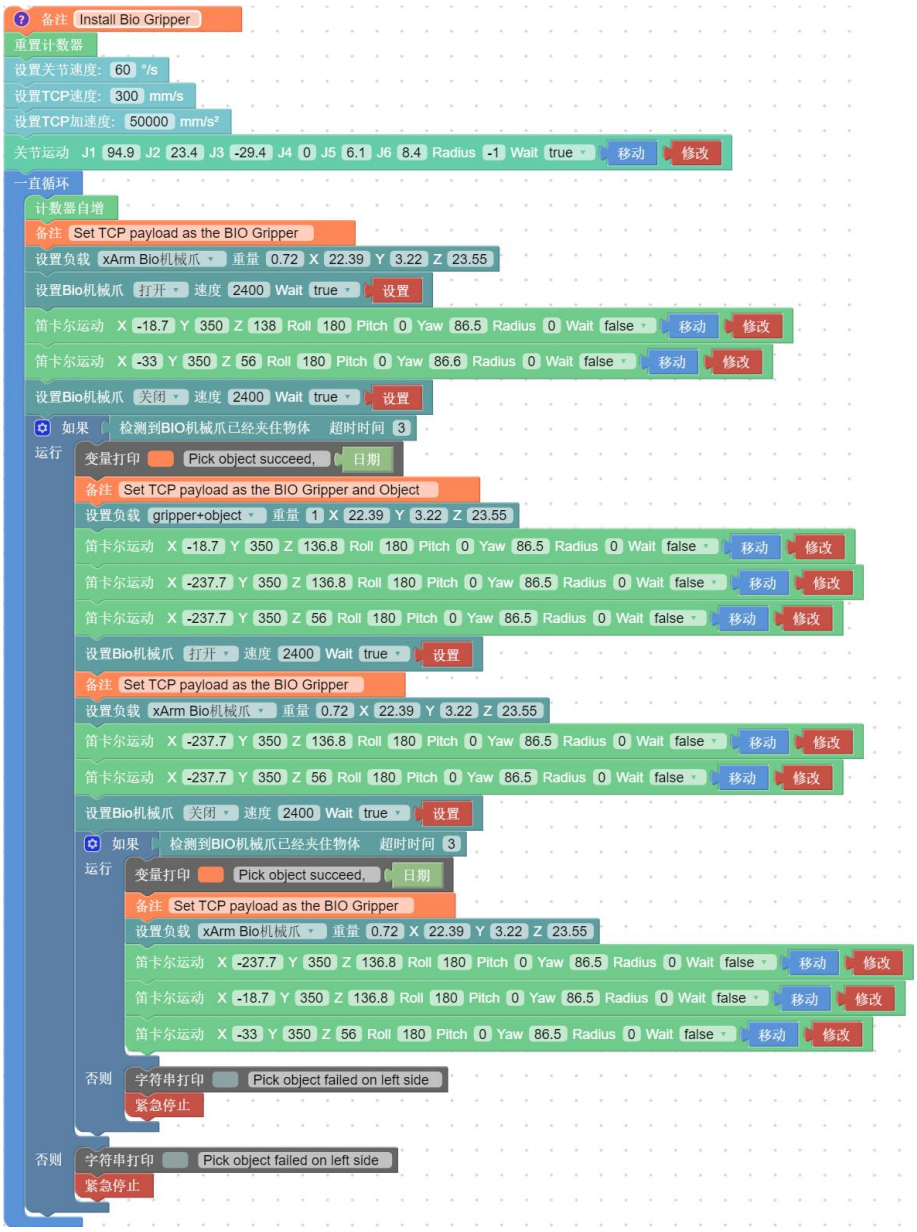
- **选择末端执行器：xArm BIO 机械爪**



点击使能后，通过打开和关闭，来控制 xArm 机械爪。

- 通过 Blockly 编程来控制 BIO 机械爪

BIO_Gripper.Blockly



这段程序的作用：执行此程序，可控制机械爪在指定位置夹取目标物，然后将目标物放到特定的位置。

注意：

1）当机械爪安装到机械臂上时，在 Blockly 程序中应当设置 TCP 负载，当机械爪夹取物体后，其重量发生变化，则需要设置新的 TCP 负载。

3.2. 用 Python-SDK 控制 BIO 机械爪

对于使用 Python-SDK 控制 BIO 机械爪的详细内容请见点击下面的链接查看:

https://github.com/xArm-Developer/xArm-Python-SDK/blob/master/example/wrapper/common/5009-set_bio_gripper.py

3.3. 用 Modbus TCP 通信协议控制 BIO 机械爪

本节主要阐述了如何通过调用 xArm 控制器的 Modbus-TCP 协议来控制 BIO 机械爪。

3.3.1. Modbus TCP 通信协议

Modbus-TCP:

Modbus 协议是一项应用层报文传输协议，有 ASCII、RTU、TCP 三种报文类型。标准 Modbus 协议物理层接口有 RS232、RS422、RS485 和以太网接口，采用 master/slave 方式通信。

Modbus　TCP 通信过程:

（1）建立 TCP 连接

（2）准备 modbus 报文

（3）使用 send 命令发送报文

（4）在同一连接下等待应答

（5）使用 recv 命令读取报文，完成一次数据交换

（6）通信任务结束时，关闭 TCP 连接

参数：

默认 TCP 端口：502　　协议标识：0x00 0x02 控制(当前只有这一个)

关于用户使用通信协议组织数据的大小端问题：

在本章节中，数据解析均为大端解析。

3.3.2. 读取 BIO 机械爪寄存器

3.3.2.1 寄存器功能

读取保持寄存器			
请求指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议标识	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x08
	寄存器	1 Byte	0x7C
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03
	寄存器起始地址	2 Bytes	**Address**
	寄存器数量	2 Bytes	**N***
响应指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	**6+N*x2**
	寄存器	1 Byte	0x7C
	状态	1 Byte	0x00
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03
	字节数	1 Byte	**N*x2**
	寄存器值	**N***x2	**Value**

注： **N*** = 寄存器数量

Address= 寄存器起始地址（见下面列表）

寄存器：

	寄存器起始地址	寄存器值	
获取机械爪状态	0x0000	2 Bytes	**未使能状态：**0x0000 **使能中状态：**0x0004 **使能完成状态：**0x0008 **停止状态：**0x0008 **运动状态：**0x0009 **夹取状态：**0x000A **报错状态：**0x000B
获取机械爪错误	0x000F	2 Bytes	**有错误：**其他返回值都代表有错误（除 0 以外） **无错误：**0x0000

3.3.2.2 示例

1. 获取机械爪状态

获取机械爪状态			
请求指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议标识	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x08
	寄存器	1 Byte	0x7C
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03
	寄存器起始地址	2 Bytes	0x00，0x00
	寄存器数量	2 Bytes	0x00，0x01
响应指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x08
	寄存器	1 Byte	0x7C
	状态	1 Byte	0x00
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03

	字节数	1 Byte	0x02
	寄存器值（机械爪在运动状态）	2 Bytes	0x00，0x09

2. 获取机械爪错误

读取保持寄存器			
请求指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x08
	寄存器	1 Byte	0x7C
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03
	寄存器起始地址	2 Bytes	0x00 0x0F
	寄存器数量	2 Bytes	0x00 0x01
响应指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x08
	寄存器	1 Byte	0x7C
	状态	1 Byte	0x00
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03
	字节数	1 Byte	0x02
	寄存器值（无错误发生）	2 Bytes	0x00，0x00

3.3.3. 写入 BIO 机械爪寄存器

3.3.3.1 寄存器功能

写入寄存器			
请求指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	**9+N*x2**
	寄存器	1 Byte	0x7C
内部使用	主机 ID	1 Byte	0x09

Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	**Address**
	寄存器数量	2 Bytes	**N***
	字节数	1 Byte	**N*x2**
	寄存器	**N***x2 Bytes	**Value**
响应指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x09
	寄存器	1 Byte	0x7C
	状态	1 Byte	0x00
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	**Address**
	寄存器数量	2 Bytes	**N***

注： **N*** = 寄存器数量

Address= 寄存器起始地址（见下面列表）

寄存器：

	寄存器起始地址	**寄存器值**	
使能/关闭机械爪	0x0100	2 Bytes	**使能:** 0x0001 **停用:** 0x0000
设置机械爪位置	0x0700	4 Bytes	**打开机械爪：**0x0000 0x0082 **闭合机械爪：**0x0000 0x0032
设置机械爪速度	0x0303	2 Bytes	0x0000-0x0BB8
清除机械爪错误	0x000F	2 Bytes	0x0000

3.3.3.2 示例

1. 使能机械爪

使能机械爪			
请求指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01

	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x0B
	寄存器	1 Byte	0x7C
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x01，0x00
	寄存器数量	2 Bytes	0x00，0x01
	字节数	1 Byte	0x02
	寄存器	2 Bytes	0x00，0x01
响应指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x09
	寄存器	1 Byte	0x7C
	状态	1 Byte	0x00
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x01，0x00
	寄存器数量	2 Bytes	0x00，0x01

2.设置机械爪速度

设置机械爪速度			
请求指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x0B
	寄存器	1 Byte	0x7C
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x03，0x03
	寄存器数量	2 Bytes	0x00，0x01
	字节数	1 Byte	0x02
	寄存器（设置速度为 1500）	2 Bytes	0x05，0xDC
响应指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02

	长度	2 Bytes	0x00，0x09
	寄存器	1 Byte	0x7C
	状态	1 Byte	0x00
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x03，0x03
	寄存器数量	2 Bytes	0x00，0x01

3. 设置机械爪位置

设置机械爪位置			
请求指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x0D
	寄存器	1 Byte	0x7C
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x07，0x00
	寄存器数量	2 Bytes	0x00，0x02
	字节数	1 Byte	0x04
	寄存器（打开机械爪）	4 Bytes	0x00，0x00，0x00，0x130
响应指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x09
	寄存器	1 Byte	0x7C
	状态	1 Byte	0x00
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x07，0x00
	寄存器数量	2 Bytes	0x00，0x02

4. 清除机械爪错误

清除机械爪错误
请求指令格式

Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x0B
	寄存器	1 Byte	0x7C
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x00 0x0F
	寄存器数量	2 Bytes	0x00 0x01
	字节数	1 Byte	0x02
	寄存器	2 Bytes	0x00 0x01
响应指令格式			
Modbus TCP 包头	事务标识	2 Bytes	0x00，0x01
	协议	2 Bytes	0x00，0x02
	长度	2 Bytes	0x00，0x09
	寄存器	1 Byte	0x7C
	状态	1 Byte	0x00
内部使用	主机 ID	1 Byte	0x09
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x00 0x0F
	寄存器数量	2 Bytes	0x00 0x01

3.3.4. 机械爪控制

控制机械爪运动的完整流程如下：

（1）使能机械爪

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

（2）打开机械爪

0x00，0x01，0x00，0x02，0x00，0x0D，0x7C，0x09，0x08，0x10，0x07，0x00，0x00，0x02，0x04，0x00，0x00，0x00，0x130

（3）闭合机械爪

0x00，0x01，0x00，0x02，0x00，0x0D，0x7C，0x09，0x08，0x10，0x07，0x00，0x00，0x02，0x04，0x00，0x00，0x00，0x50

3.4. 用 Modbus RTU 通信协议控制 BIO 机械爪

3.4.1. Modbus RTU 通信协议

机械爪默认为标准 Modbus RTU 协议，默认波特率 2Mbps，机械爪 ID 为 0x08。目前支持的功能码有：0x03/0x10。

3.4.2. 读取 BIO 机械爪寄存器

读取保持寄存器			
请求指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03
	寄存器起始地址	2 Bytes	**Address**
	寄存器数量	2 Bytes	**N***
	Modbus CRC 16	2 Bytes	**CRC***
响应指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03
	字节数	1 Byte	**N*x2**
	寄存器值	**N***x2 Bytes	**Value**
	Modbus CRC16	2 Bytes	**CRC***

注： **N*** = 寄存器数量

Address= 寄存器起始地址（见下面列表）

CRC* = 循环冗余校验

寄存器：

	寄存器起始地址	寄存器值	
获取机械爪状态	0x0000	2 Bytes	**未使能状态:** 0x0000 **使能中状态:** 0x0004 **使能完成状态:** 0x0008 **停止状态:** 0x0008 **运动状态:** 0x0009 **夹取状态:** 0x000A **报错状态:** 0x000B
获取机械爪错误	0x000F	2 Bytes	**有错误:** 其他返回值都代表有错误（除 0 以外） **无错误:** 0x0000

3.4.3. 写入 BIO 机械爪寄存器

写入寄存器			
请求指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	**Address**
	寄存器数量	2 Bytes	**N***
	字节数	1 Byte	**N*x2**
	寄存器	**N***x2 Bytes	**Value**
	Modbus CRC 16	2 Bytes	**CRC***
响应指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	**Address**
	寄存器数量	2 Bytes	**N***
	Modbus CRC 16	2 Bytes	**CRC***

注： **N*** = 寄存器数量

Address= 寄存器起始地址（见下面列表）

CRC* = 循环冗余校验

寄存器:

	寄存器起始地址	寄存器值	
使能/关闭机械爪	0x0100	2 Bytes	**使能:** 0x0001 **停用:** 0x0000
设置机械爪位置	0x0700	4 Bytes	**打开机械爪:** 0x0000 0x0082

			闭合机械爪：0x0000 0x0032
设置机械爪速度	0x0303	2 Bytes	0x0000-0x0BB8 **单位**：r/min
清除机械爪错误	0x000F	2 Bytes	0x0000

3.4.4. Modbus RTU 示例

1. 使能机械爪

使能机械爪			
请求指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x01，0x00
	寄存器数量	2 Bytes	0x00，0x01
	字节数	1 Byte	0x02
	寄存器	2 Bytes	0x00，0x01
	Modbus CRC16	2 Bytes	0x1D，0x00
响应指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x01，0x00
	寄存器数量	2 Bytes	0x00，0x01
	Modbus CRC16	2 Bytes	0x00，0xAC

1. 设置机械爪速度

设置机械爪速度			
请求指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x03，0x03
	寄存器数量	2 Bytes	0x00，0x01
	字节数	1 Byte	0x02
	寄存器（设置速度为	2 Bytes	0x05，0xDC
	Modbus CRC16	2 Bytes	0xFD，0xFA

响应指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x03，0x03
	寄存器数量	2 Bytes	0x00，0x01
	Modbus CRC16	2 Bytes	0xF1，0x14

2. 打开机械爪

设置机械爪位置			
请求指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x07，0x00
	寄存器数量	2 Bytes	0x00，0x02
	字节数	1 Byte	0x04
	寄存器（打开机械爪）	4 Bytes	0x00，0x00，0x00，0x82
	Modbus CRC16	2 Bytes	0x7B，0x62
响应指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x07，0x00
	寄存器数量	2 Bytes	0x00，0x02
	Modbus CRC16	2 Bytes	0x40，0x25

读取机械爪状态直到它处于停止状态

获取机械爪状态			
请求指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03
	寄存器起始地址	2 Bytes	0x00，0x00
	寄存器数量	2 Bytes	0x00，0x01
	Modbus CRC16	2 Bytes	0x84，0x93
响应指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03
	字节数	1 Byte	0x02
	寄存器值（机械爪在停止状态）	2 Bytes	0x00，0x00
	Modbus CRC16	2 Bytes	0x64，0x45

3. 闭合机械爪

设置机械爪位置			
请求指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x07，0x00
	寄存器数量	2 Bytes	0x00，0x02
	字节数	1 Byte	0x04
	寄存器（打开机械爪）	4 Bytes	0x00，0x00，0x00，0x32
	Modbus CRC16	2 Bytes	0x7A，0xD6
响应指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x10
	寄存器起始地址	2 Bytes	0x07，0x00
	寄存器数量	2 Bytes	0x00，0x02
	Modbus CRC16	2 Bytes	0x40，0x25

读取机械爪状态直到它处于停止状态

获取机械爪状态			
请求指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03
	寄存器起始地址	2 Bytes	0x00，0x00
	寄存器数量	2 Bytes	0x00，0x01
	Modbus CRC16	2 Bytes	0x84，0x93
响应指令格式			
Modbus RTU 数据	机械爪 ID	1 Byte	0x08
	功能码	1 Byte	0x03
	字节数	1 Byte	0x02
	寄存器值（机械爪在运动状态）	2 Bytes	0x00，0x01
	Modbus CRC16	2 Bytes	0x64，0x45

4.报警与处理

处理方式可采用重新上电，步骤如下（重新上电需要走完以下所有步骤）：

1. 通过控制器上的紧急停止按钮重新对机械臂上电

2. 使能机械臂

xArm Studio 使能方式：点击报错弹窗的引导按钮或者首页的使能机械臂按钮。

xArm-Python-SDK 使能方式 ： 参照报警处理方式。

xArm-ROS 库：查看相关文档 ： https://github.com/xArm-Developer/xarm_ros

3. 重新使能机械爪

报警代码	报警说明	报警处理
0x0B	机械爪过流	机械爪电流过大 请点击“确认”重新使能机械爪 或断电重启
0x0C	夹取物脱落	夹取物脱落 请放置好夹取物并清除错误，然后将 0x000F 寄存器置 0，重新进行夹取
若多次重新上电无效后请寻找 UFACTORY 团队支持。		

xArm-Python-SDK 报警处理方式：

在用 Python 库设计机械臂运动规划时，如果机械臂出现故障，需要手动清除错误。清除错误后，仍需重新使能机械臂，并将机械臂设置为运动模式，方可使机械臂正常运动。此时根据上报的错误信息，应重新调整机械臂的路径规划。

Python 库清除错误步骤：

（详细说明请查看 GitHub：

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

5.BIO 机械爪技术规格

BIO 机械爪	
额定电源电压	24V DC
绝对最大电源电压	28V DC
静态功耗（最低功耗）	0.96W
峰值电流	1.5A
质量	760g
最大夹持力度	20N
行程范围	70-150mm
通信方式	RS-485
通信协议	Modbus RTU
可编程参数	速度
状态	工作状态，电源状态
反馈	抓取检测，滑落检测

6.售后服务

1. 售后政策：

对于产品的质量保证以及维修和退换货的详情，见官网的售后政策：

https://www.cn.ufactory.cc/warrenty

2. 售后服务流程：

（1）联系技术支持（support@ufactory.cc），确认产品需要寄回维修，确定需要寄回的部件。

（2）我司根据售后政策，判定产品保修状况，付费或免费维修。

（3）维修、测试完成后，我们会将产品寄回，一般情况下，整个维修流程大约需要 1-2 周。

注意：

1. 当需要将产品寄回我司进行维修时，需要将产品用包装箱打包好，避免在运输过程中发生不必要的碰撞，导致机械爪受损。