智元机器人

产品使用说明书

OmniHand 专业款 2025

版本: 1.3

语言：简体中文

智元创新（上海）科技股份有限公司
www.AGIBOT.com

目录

1. 重要安全须知

本章节介绍了安装和使用 OmniHand 专业款 灵巧手时必须遵守的安全要求，OmniHand 专业款 灵巧手是一款半成品机械，因此，在首次使用之前，必须仔细阅读本说明书，全面了解相关安全功能及潜在危险。相关人员在每次安装 OmniHand 专业款 灵巧手之前，必须进行风险评估，并在使用过程中严格遵守本说明书中所有说明和指导信息。

1.1 使用注意事项

严禁将 OmniHand 专业款 灵巧手置于高温、高湿的环境中。

请勿让 OmniHand 专业款 灵巧手承受过大的负载和冲击。

严禁在指尖施加过大的力。

严禁用力拉扯 OmniHand 专业款 灵巧手手指。

严禁将 OmniHand 专业款 灵巧手靠近火源。

严禁将 OmniHand 专业款 灵巧手置于易燃、易爆环境中。

严禁在强电磁场环境中使用，如高压电线附近、大功率机器附近等。

严禁用 OmniHand 专业款 灵巧手去抓取过重、过热、尖锐、表面粗糙、有腐蚀性的物体。

在没有保护的情况下，严禁让 OmniHand 专业款 灵巧手接触液体（酒、水、饮料等）和沙尘，如不慎接触，请立即关闭，并联系售后维修人员。

严禁私自拆解 OmniHand 专业款 灵巧手，否则将导致《售后服务承诺书》中保修相关条款失效。发生任何故障，请及时联系售后维修人员。

严禁用 OmniHand 专业款 灵巧手操作危险机械。由于此类行为导致的人身伤害及财产损失，我司不承担任何责任。

1.2 风险评估

潜在危害清单

机械危害：因接触 OmniHand 专业款 灵巧手或其夹持的尖锐物体而导致压伤、擦伤等。

电气危害：因接触带电零件而导致电烧伤或触电。

高温危害：因长时间接触 OmniHand 专业款 灵巧手的高温表面而导致皮肤烧伤等。

噪音危害：无。

振动危害：无。

辐射危害：无。

材料/物质危害：无。

人体工程学危害：无。

环境危害：无。

1.3 售后服务承诺书

OmniHand 专业款 灵巧手具有 6 个月有限保修期。若在投入使用后，出现因制造或材料不良所致的缺陷，公司提供必要的备用部件予以更换或维修。若设备缺陷是由处理不当或未遵循用户指南中所述的相关信息或私自拆卸产品所致，则本产品质量保证即告失效。在不违背本产品质量保证的原则下，若产品已经超出保修期，公司保留向客户收取更换或维修费用的权利。

在保修期外，如果设备呈现缺陷，公司不承担由此引起的任何损害或损失，包括但不仅限于生

产损失或对其他生产设备造成的损坏。

1.4 免责声明

公司致力于不断提高产品可靠性和性能，并因此保留升级产品的权利，恕不另行通知。公司力求确保本手册内容的准确性和可靠性，但不对其中的任何错误或遗漏信息负责。以下情况导致的故障不在本保修范围内：

1）未按用户手册要求安装、接线、连接其他控制设备；

2）未经允许，私自拆装灵巧手；

3）使用时超出用户手册所示规格或标准；

4）由于运输不当导致的产品损坏；

5）事故或碰撞导致的损坏；

6）火灾、地震、海啸、雷击、大风和洪水等自然灾害。

本公司对于因客户违反本章节内的免责声明而造成的任何损失或伤害不承担任何责任。请客户在购买和使用本产品前，仔细阅读并同意本免责声明。

2. 产品简介

2.1 产品概述

OmniHand 专业款 2025 是一款高集成度全能作业灵巧手，强感知，更有料。

行业内最全面的高自由度商用灵巧手，兼具尺寸小、重量轻、高负载、高精度、开合快、多维触觉等特性， 强化「作业智能」属性。

专业款灵巧手 主动自由度电机顺序示意图

2.2 主要特性

- **拟人尺寸：**19 自由度，仅重 820g，同配置灵巧手中最小、最轻、最拟人
- **全能作业：**优化构型设计，轻松操作专业工具，单手指尖最大 20N 出力
- **超强感知：**多模态感知（位置、法向力、切向力），最高 0.1N 级灵敏度，搭配智能算法

2.3 产品参数

Omnihand Pro 2025（专业款）		
本体参数	重量	≤820g
	尺寸	207*98*56mm
	主动自由度	12
	总自由度	19
	最小开合时间（典型值）	0.7s

	指尖重复定位精度（典型值）	0.4mm
	负载能力	五指抓握力 ≥10kg
	工作电压	8.3-35.6V
	通讯接口	CAN-FD
	工作温度范围	-20~50°C
	在线升级	支持 OTA 在线升级
触觉传感器	力感知类型	指尖三维力
	阵列分辨率	0.1N
	感知范围	0~50N
	最大接受力(不损坏)	1000N

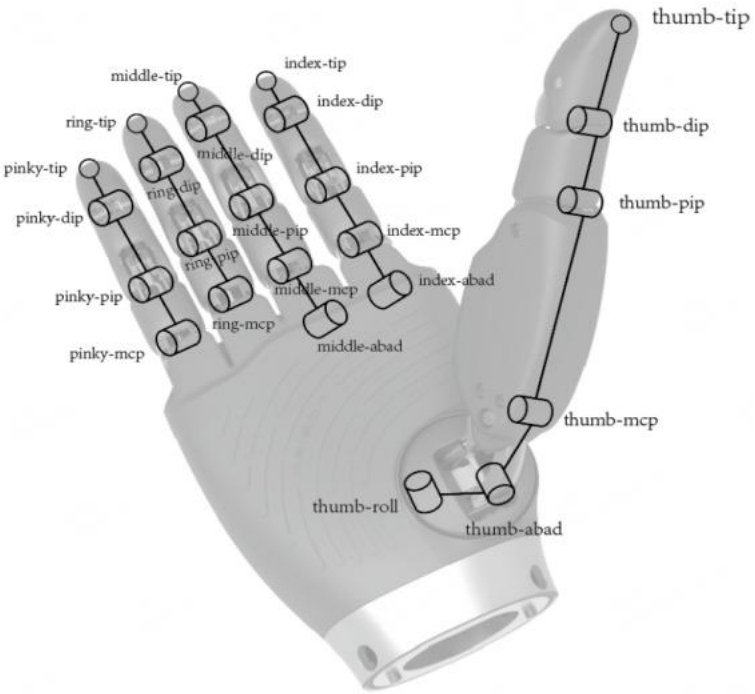
[1] 以上参数，在不同业务场景、不同型号参数配置等情况，在应用中有所差异，请以实际为准。

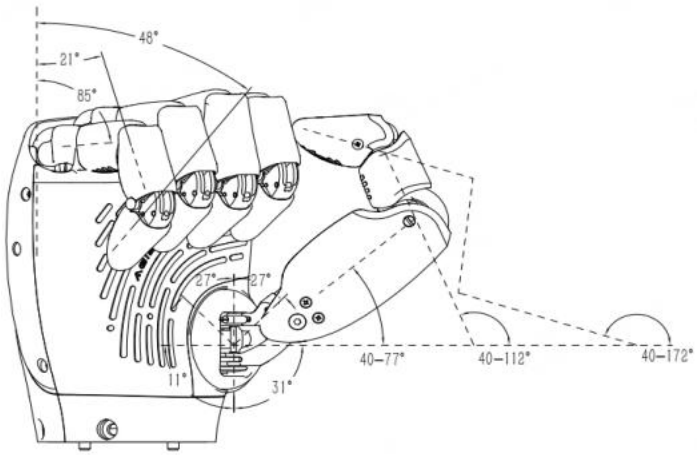
[2] 产品外观后续可能会有升级调整，请以届时实物为准。

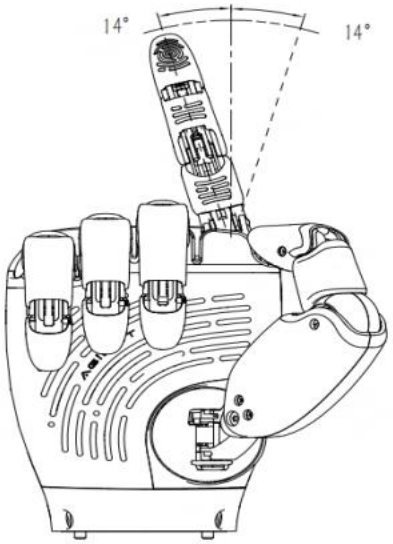
2.4 关节角度范围

手指	关节	运动方向	角度最小值	角度最大值
食指	DIP	屈伸	0°	226°
	PIP	屈伸	0°	157°
	MCP	屈伸	0°	76°

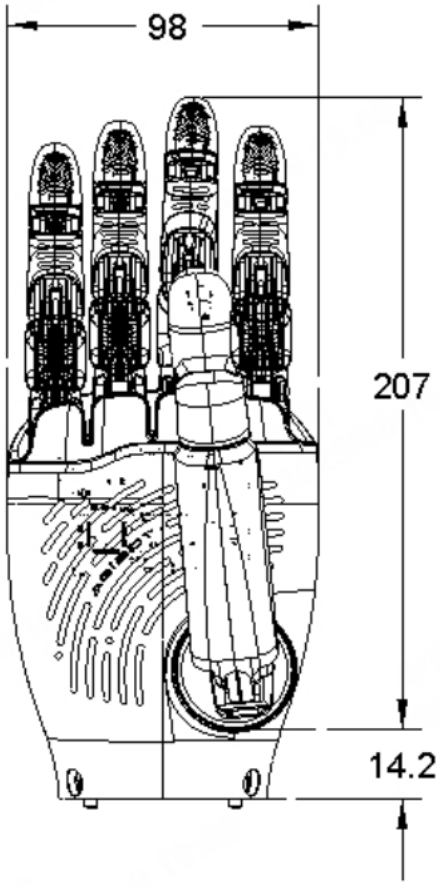
	ABAD	侧摆	0°	14°
中指	DIP	屈伸	0°	227°
	PIP	屈伸	0°	159°
	MCP	屈伸	0°	77°
	ABAD	侧摆	0°	14°
无名指，小指	DIP	屈伸	0°	228°
	PIP	屈伸	0°	159°
	MCP	屈伸	0°	85°
	ABAD	侧摆	0°	0°
大拇指	DIP	屈伸	40°	172°
	PIP	屈伸	40°	112°
	MCP	屈伸	40°	77°
	ABAD	侧摆	0	54°
	ROLL	自旋	0°	42°

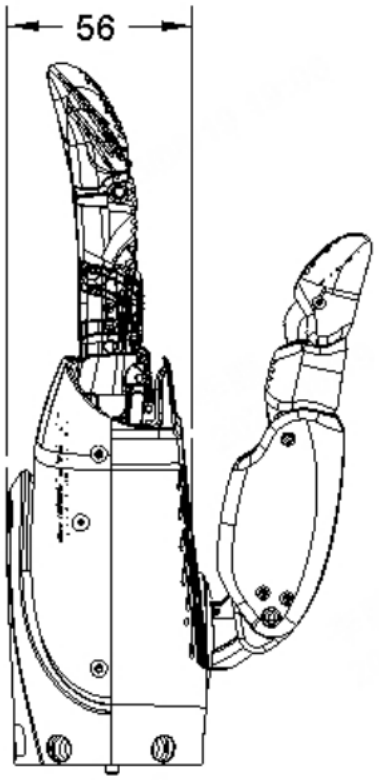

thumb-tip
thumb-dip
thumb-pip
thumb-mcp
thumb-roll
thumb-abad
index-tip
index-dip
index-pip
index-mcp
index-abad
middle-tip
middle-dip
middle-pip
middle-mcp
middle-abad
ring-tip
ring-dip
ring-pip
ring-mcp
pinky-tip
pinky-dip
pinky-pip
pinky-mcp


48°
21°
85°
27°
27°
11°
31°
40–77°
40–112°
40–172°


14°
14°

2.5 产品尺寸



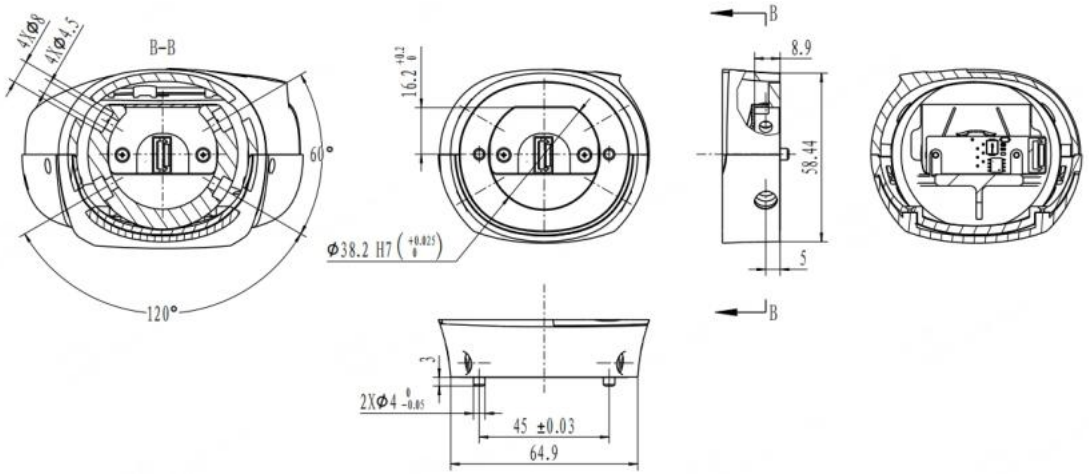


3. 产品安装说明

3.1 装箱清单

- OmniHand 专业款 2025 × 1
- USB 转 CAN-FD 通讯线 × 1
- USB 转 CAN-FD 转接盒子
- 电源适配器 × 1
- 产品合格证 × 1

3.2 机械接口

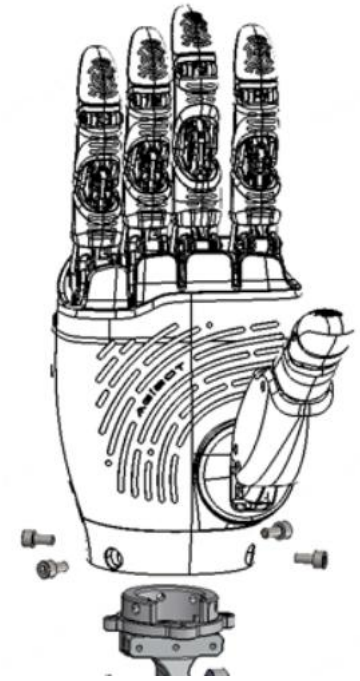


3.3 安装说明

1. 安装之前 需要自行准备 4 颗 M4*8 圆柱头螺钉 以及相应的内六角扳手

2. 连接通讯线束（注意线束卡扣方向）

3. 将机械臂末端法兰插入灵巧手的底部接口中， 然后用 4 颗 M4 的螺钉锁紧，建议锁紧扭矩 M：1.1Nm<M＜1.5Nm；灵巧手适配其他机器人本体的情况，请查看机械接口图示，设计转接法兰



4. 锁紧螺钉建议涂螺纹胶；

3.4 电气和通讯接口

工作电压电流：典型值 24V，2A。峰值电流≤3.5A@24V（<30s）。

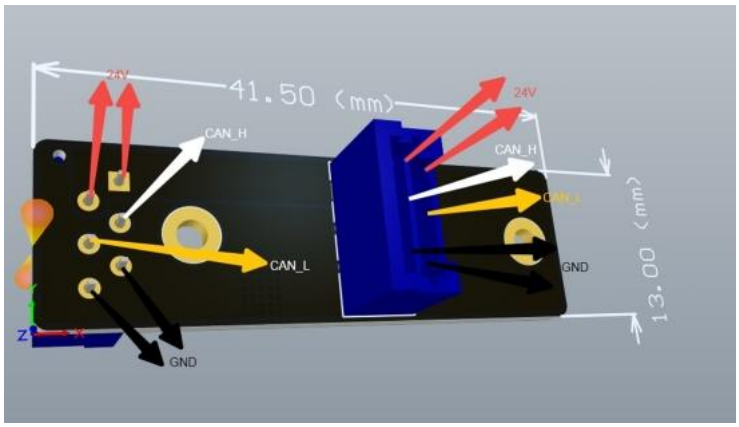
电压输入范围：12~28V，最大耐受电压 45V（<1s），欠压（8V）过压（35V）保护关闭，关闭电流小于 1mA。

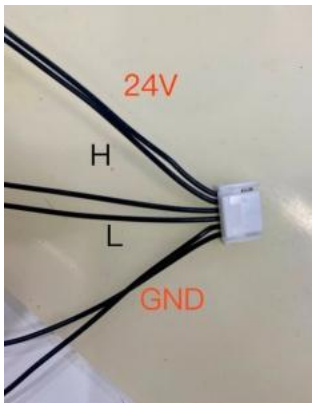
电压报警设置：可配置欠压过压报警阈值，报警不动作。

通信接口：CANFD，有 120 欧姆终端电阻。

电源端子型号：插座：BM06B-ZESS-TBT，插头：ZER-06V-S。

端子信号顺序：1、2--24V；3--CANH；4--CANL；5、6--GND。





3.5 整手不支持热插拔

整手在带电状态下，不支持电源及信号线热插拔功能，请使用者规范使用；如若未按要求进行热插拔出现故障，客户承担相应责任。

4. 通讯协议说明

4.1 通讯接口

OmniHand Pro 2025 支持基于 CAN-FD 的二次开发， 开发文档请查阅 https://github.com/AgibotTech/OmniHand-Pro-2025

- CAN-FD 接口，默认 1Mbps 80% + 5Mbps 80% 扩展标识符（29bit） 数据帧格式。禁用标准标识符和远程帧格式；
- 控制命令采用一问一答的机制，部分状态信息可配为主动上报，应答超时时间 50ms；
  - 发送标识符和应答标识符相同，返回读写数据；
- 扩展帧定义：
  - `Bit0 ~ Bit6`：设备 ID；

- Bit7：读写标志位，0 表示读寄存器操作，1 表示写寄存器操作；
- Bit8 ~ Bit14：产品 ID；
- Bit15：保留；
- Bit16 ~ Bit23：寄存器地址；
- Bit24 ~ Bit28：子寄存器地址；

Byte3							
bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24
x	x	x	子寄存器地址：0x0 ~ 0x1F				

Byte2							
bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16
寄存器地址：0x00 ~ 0xFF							

Byte1							
bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
保留	产品 ID：0x01 ~ 0x7F						

Byte0							
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
R/W	设备 ID：0x01 ~ 0x7F						

- 设备 ID 定义：

    - 广播地址：`0x00`；
    - 默认地址：`0x01`；
    - 设备地址：`0x01 ~ 0x7F`；
- 产品 ID 定义：
    - 根据产品定义，固定且不可修改；
- 数据定义：**小端格式**；

4.2 寄存器地址及其子地址定义

- 读寄存器将读写标志位 `Bit7` 置 `0`，写寄存器将读写标志位 `Bit7` 置 `1`；
- 配置信息寄存器写入会自动保存，重启之后生效；
- 读寄存器发送的长度不做限制，数据不做限制，默认数据长度为 `0`，不发送数据内容；
- 写寄存器必须符合寄存器长度，数据必须合法，否则不会应答；
- 受限于子寄存器长度，最大支持一次性配置自由度为 32 自由度；

4.3 通用寄存器定义

编号	寄存器地址	寄存器名称	子寄存器	子寄存器名称	寄存器内容	寄存器长度	读写权限

Pn 1	0x01	厂家信息	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	48Byte	只读
			0x01	产品型号	ASCII 字符串，剩余的长度补 0x00； 示例：SkillHand S6	16Byte	只读
			0x02	产品序列号	ASCII 字符串，固定长度； SSCMMVVYMXXXXX SS：产品系列，共 2 位，首位采用 26 进制(A~Z)编码，次位采用 10 进行(0~9)编码； C：产品配置，共 1 位，采用 32 进制(0~Z)编码； MM：组件号，共 2 位，采用 10 进制(0~9)编码； VV：BOM 版本，共 2 位，采用 10 进制(0~9)编码； Y：生产年份，共 1 位，采用 26 进制(A~Z)编码； M：生产月份，共 1 位，采用 12 进制(1~C)编码； XXXXX：流水号，共 5 位，采用 10 进制(0~9)编码；	14Byte	只读
			0x03	硬件版本信息	硬件版本信息； Byte 0：Major 主版本号； Byte 1：Minor 次版本号； Byte 2：Patch 修订版本号； Byte 3：Reserve 保留； Hardware Version：Major.Minor.Patch	4Byte	只读

			0x04	软件版本信息	软件版本信息； Byte 0：Major 主版本号； Byte 1：Minor 次版本号； Byte 2：Patch 修订版本号； Byte 3：Reserve 保留； Software Version：Major.Minor.Patch	4Byte	只读
			0x05	供电电压	32 位无符号整型数据；默认值 12500(12.5V)；单位 mV； 有效值 8000，24000 调节单位 500mV；	2Byte	只读
			0x06	主动自由度	示例：0x0C，主动自由度数量；取值范围【1，32】	1Byte	只读
Pn2	0x02	设备信息	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	5Byte	读写
			0x01	设备 ID	默认设备 ID：0x01； 设备 ID 地址：0x01 ~ 0x7F；	1Byte	读写
Pn3	0x03	电流阈值	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	读写
			0x01	#1 关节电机电流阈值	16 位无符号整型数据；默认值：1500；单位：mA； 2Byte 表示一个关节电机；有效值 0，65535	2Byte	读写
			0x02	#2 关节电机电流阈		2Byte	读写

				值			
			…	… …		…	读写
Pn4	0x04	温度阈值	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	读写
			0x01	#1 关节电机回温启动阈值	回温启动阈值 - 8 位无符号整型数据；默认值 60；单位 °C； 1Byte 表示一个关节电机；有效值【50， 过温保护阈值 - 5】	2Byte	读写
				#1 关节电机过温保护阈值	过温保护阈值 - 8 位无符号整型数据；默认值 80；单位 °C； 1Byte 表示一个关节电机；有效值【回温启动阈值 + 5， 80】		
			0x02	#2 关节电机回温启动阈值		2Byte	读写
				#2 关节电机过温保护阈值			
			…	… …		… …	…
				… …			
Pn5	0x05	触觉传感器	0x01	大拇指指尖传感器	小端 byte0：传感器在线状态） byte1-18：各通道值 byte19-24：三维力 （0-3000）单	48byte	预留

					位 0.10N byte25-28：自电容接近 Byte0 为 1 时表示传感器在线 Byte19-20 代表法向力（小端模式） Byte21-22 代表切向力（小端模式） Byte23-24 代表切向力角度（小端模式） 法向力：垂直于手指表面的力 切向力：水平于手指表面的力 切向力角度：指尖位置为向上 0°，顺时针旋转，范围：0°~359°。		
			0x02	食指指尖传感器			
			0x03	中指传感器			
			0x04	无名指指尖传感器			
			0x05	小拇指指尖传感器			
			…	…	… …	… …	
Pn15	0x0F	预留	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	预留
Pn16	0x10	控制模式	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	读写
			0x01	#1 关节电机控制模式	8 位无符号整型数据；根据产品形态定义针对控制模式进行对应的扩展，默认值 0；	1Byte	读写

			0x02	#2 关节电机控制模式	低 3bit 有效 xxxx x111 0 - 位置模式； 1 - 速度模式；	1Byte	读写
			…	… …		… …	…
Pn 19	0x13	位置控制	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	读写
			0x01	#1 关节电机目标位置	16 位有符号整型数据；默认值 0；单位：1/65535 最大位置； 2Byte 表示一个关节电机；有效值根据产品进行定义； 写寄存器，设置目标位置，返回当前的实际位置； 读寄存器，获取当前的实际位置；	2Byte	读写
			0x02	#2 关节电机目标位置		2Byte	读写
			…	… …		… …	…
Pn 23	0x17	预留	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	预留
…	…	…	…	…	… …	… …	…
Pn 31	0x1F	预留	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	预留
Pn 32	0x20	错误上报	0x0	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主	只读

						动自由度	
			0x1	#1 关节电机错误信息	在发生错误时主动上报当前电机的错误信息；16 位无符号整型数据；错误信息定义如下： Bit0：堵转保护； Bit1：过温保护； Bit2：过流保护； Bit3：电机异常； Bit4：通讯异常； Bit5 ~ Bit15：保留； > 根据产品信息定义可以进行补充 对该寄存器写 0x0000 主动清除当前的错误信息；	2Byte	读写
			0x2	#2 关节电机错误信息		2Byte	读写
			...	... ...		... ...	...
Pn 33	0x21	温度上报	0x0	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	只读
			0x1	#1 关节电机温度信息	写该寄存器配置主动上报周期时间，单位：ms； 16 位无符号整型数据；默认值：0，不主动上报； 示例：以 1Hz 频率主动上报关节温度，下发：0x03E8; 读该寄存器主动获取当前温度信息，单位：°C； 16 位有符号整型数据； 示例：当前关节温度为 50°C，上报：0x0032；	2Byte	读写
			0x2	#2 关节电机温度信息		2Byte	读写
			...	... ...		... ...	...

					> 大于 1Byte 按照小端模式发送		
Pn 34	0x22	电流上报	0x0	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	只读
			0x1	#1 关节电机温度信息	写该寄存器配置主动上报周期时间，单位：ms； 16 位无符号整型数据；默认值：0，不主动上报； 示例：以 1Hz 频率主动上报关节电流，下发：0x03E8； 读该寄存器主动获取当前电流，单位：mA； 16 位有符号整型数据； 示例：当前关节电流为 1500mA，上报：0x05DC； > 大于 1Byte 按照小段模式发送	2Byte	读写
			0x2	#2 关节电机温度信息		2Byte	读写
			...	... ...		... ...	...
Pn 35	0x23	预留	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	预留
...	...	...	...	...	... ...	...	...
Pn 47	0x2F	预留	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	预留

4.4 厂家配置寄存器定义

不同的产品特殊配置参数，如传感器校准标定，手指关节 SN 等等信息；

编号	寄存器地址	寄存器名称	子寄存器	子寄存器名称	寄存器内容	寄存器长度	读写权限
Pn48	0x30	预留	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	预留
…	…	…	…	…	… …	…	…
Pn127	0x7F	预留	0x00	子寄存器操作	读写该寄存器下所有的子寄存器；	单个子寄存长度×主动自由度	预留

4.5 固件升级寄存器(预留)

Host：上位机

Device：灵巧手

Byte								**Byte**							
bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24	bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16
x	x	x	R/W	子寄存器地址：0x0 ~ 0xF				寄存器地址：0x00 ~ 0xFF							

Byte								**Byte**							
bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
保留	产品 ID：0x01 ~ 0x7F							保留	设备 ID：0x01 ~ 0x7F						

参数	读写	寄存器地	寄存器名	子寄	子寄存器	寄存器内容	寄存器长	数据

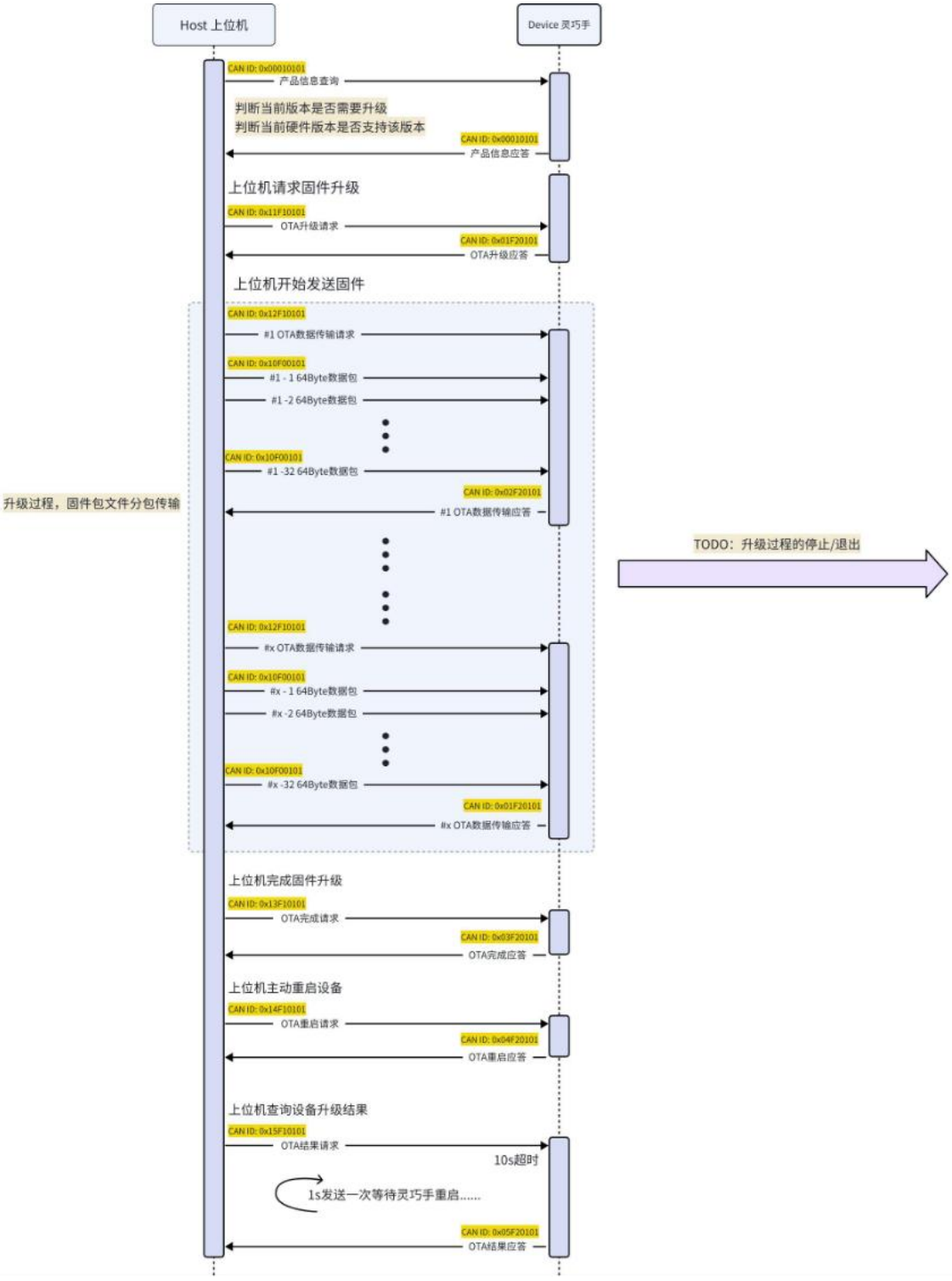
编号	标志位	址	称	存器	名称		度	流
Pn1	1	0xF0	OTA数据包	0x00	数据包	数据包内容	64Byte	Host -> Device
Pn2	1	0xF1	OTA相关请求命令	0x00	N/C	不响应		
				0x01	OTA升级请求	0 - 无符号 32 位整型数据，固件长度； 注：固件长度需要发送 2K 对齐的固件长度； 1 - 无符号 16 位整型数据，固件分包个数；(2K 为一个分包) 2 - 无符号 16 位整型数据，预留默认为 0； 3 - 无符号 32 位整型数据，预留默认为 0； 4- 无符号 32 位整型数据，预留默认为 0；	16Byte	Host -> Device
				0x02	OTA数据传输请求	0- 无符号 16 位整型数据，固件分包索引； 1 - 无符号 16 位整型数据，固件分包 CRC 校验值；(CRC16-Modbus) 注：数据包长度默认 2K，不足 2K 的分包使用 0xFF 补齐；	4Byte	Host -> Device
				0x03	OTA完成	无符号 32 位整型数据；预留默认为 0；	4Byte	Host -> Devi

					请求			ce
				0x04	OTA 重启请求	无符号 32 位整型数据；延迟 x 时间后重启，0 立即重启；	4Byte	Host -> Device
				0x05	OTA 结果请求	无符号 32 位整型数据；预留默认为 0；	4Byte	Host -> Device
				0x06	OTA 退出请求	无符号 32 位整型数据；	4Byte	Host -> Device
Pn3	0	0xF2	OTA 相关应答命令	0x00	N/C	不响应		
				0x01	OTA 升级应答	无符号 32 位整型数据； 0x00000000：正常应答； 0x00000001：Flash 分区表获取失败； 0x00000002：Download 分区擦除失败； 0x00000003：固件长度不合法	4Byte	Device -> Host
				0x02	OTA 数据传输应答	无符号 32 位整型数据； 0x00000000：正常应答； 0x00000001：CRC 校验失败；	4Byte	Device -> Host
				0x03	OTA 完成应答	无符号 32 位整型数据； 0x00000000：正常应答； 0x00000001：没有 OTA 请求；	4Byte	Device -> Host

						0x00000002：固件不完整； 0x00000003：Flash 分区表获取失败； 0x00000004：升级标志位设置失败；		
				0x04	OTA 重启应答	无符号 32 位整型数据； 0x00000000：正常应答；	4Byte	Device -> Host
				0x05	OTA 结果应答	无符号 32 位整型数据； 0x00000000：正常应答； 0x00000001：未知错误； 0x00000002：数据读取错误； 0x00000003：数据写入错误； 0x00000004：固件损坏； 0x00000005：没有足够的更新空间； 0x00000006：不支持的压缩算法； 0x00000007：不支持的加密类型； 0x00000008：不支持的差分算法； 0x00000009：不支持的数字签名； 0x0000000A：内存不足； 0x0000000B：不支持的容器类型； 0x0000000C：目标分区不存在；	4Byte	Device -> Host

						0x0000000D：Flash 擦除失败；		
				0x06	OTA 退出应答	无符号 32 位整型数据； 0x00000000：正常应答；	4Byte	Device -> Host

4.5.1 升级流程



4.5.2 固件数据包 CRC16 校验示例

C++

```
class ModbusCRC16 {
public:
```

```
    static uint16_t Calculate(const std::vector<uint8_t> &buffer) {
        uint16_t crc = 0xFFFF;
        for (auto data : buffer) {
            crc = (crc >> 8) ^ CRC_TABLE[(crc ^ data) & 0xFF];
        }
        return crc;
    }

private:
    static constexpr uint16_t CRC_TABLE[256] = {
        0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241,
        0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0440,
        0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40,
        0x0A00, 0xCAC1, 0xCB81, 0x0B40, 0xC901, 0x09C0, 0x0880, 0xC841,
        0xD801, 0x18C0, 0x1980, 0xD941, 0x1B00, 0xDBC1, 0xDA81, 0x1A40,
        0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01, 0x1DC0, 0x1C80, 0xDC41,
        0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0, 0x1680, 0xD641,
        0xD201, 0x12C0, 0x1380, 0xD341, 0x1100, 0xD1C1, 0xD081, 0x1040,
        0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281, 0x3240,
        0x3600, 0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480, 0xF441,
        0x3C00, 0xFCC1, 0xFD81, 0x3D40, 0xFF01, 0x3FC0, 0x3E80, 0xFE41,
        0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881, 0x3840,
        0x2800, 0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80, 0xEA41,
        0xEE01, 0x2EC0, 0x2F80, 0xEF41, 0x2D00, 0xEDC1, 0xEC81, 0x2C40,
        0xE401, 0x24C0, 0x2580, 0xE541, 0x2700, 0xE7C1, 0xE681, 0x2640,
        0x2200, 0xE2C1, 0xE381, 0x2340, 0xE101, 0x21C0, 0x2080, 0xE041,
        0xA001, 0x60C0, 0x6180, 0xA141, 0x6300, 0xA3C1, 0xA281, 0x6240,
        0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0, 0x6480, 0xA441,
        0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80, 0xAE41,
        0xAA01, 0x6AC0, 0x6B80, 0xAB41, 0x6900, 0xA9C1, 0xA881, 0x6840,
        0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80, 0xBA41,
        0xBE01, 0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81, 0x7C40,
        0xB401, 0x74C0, 0x7580, 0xB541, 0x7700, 0xB7C1, 0xB681, 0x7640,
        0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080, 0xB041,
        0x5000, 0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280, 0x9241,
        0x9601, 0x56C0, 0x5780, 0x9741, 0x5500, 0x95C1, 0x9481, 0x5440,
        0x9C01, 0x5CC0, 0x5D80, 0x9D41, 0x5F00, 0x9FC1, 0x9E81, 0x5E40,
```

```
        0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901, 0x59C0, 0x5880, 0x9841,
        0x8801, 0x48C0, 0x4980, 0x8941, 0x4B00, 0x8BC1, 0x8A81, 0x4A40,
        0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80, 0x8C41,
        0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680, 0x8641,
        0x8201, 0x42C0, 0x4380, 0x8341, 0x4100, 0x81C1, 0x8081, 0x4040
    };
};
```

4.6 通用控制示例

混合控制示例 1:

灵巧手设备 ID 为 0x01，12 个自由度，控制模式为位置力控模式，一次性可以下发 12 个关节的目标信息；

- 请求的每一个关节的 bag 组成为：关节 ID + 控制模式 + 目标位置 + 目标力距共 5Byte；
- 应答的每一个关节的 bag 组成为：关节 ID + 控制模式 + 实际位置 + 实际力距共 5Byte；
    - 请求当前所有的关节控制模式为位置力控模式，目标位置为 1000，目标力距为 1000；
    - 应答当前所有的关节控制模式为位置力控模式，实际位置为 500，实际力距为 500；

	CANFD ID	Length	Data(HEX)

请求	0x00140181	64Byte(60Byte有效)	60 E8 03 E8 03 61 E8 03 E8 03 62 E8 03 E8 03 63 E8 03 E8 03 64 E8 03 E8 03 65 E8 03 E8 03 66 E8 03 E8 03 67 E8 03 E8 03 68 E8 03 E8 03 69 E8 03 E8 03 6A E8 03 E8 03 6B E8 03 E8 03 00 00 00 00
应答	0x00140101	64Byte(60Byte有效)	60 F4 01 F4 01 61 F4 01 F4 01 62 F4 01 F4 01 63 F4 01 F4 01 64 F4 01 F4 01 65 F4 01 F4 01 66 F4 01 F4 01 67 F4 01 F4 01 68 F4 01 F4 01 69 F4 01 F4 01 6A F4 01 F4 01 6B F4 01 F4 01 00 00 00 00

混合控制示例 2:

灵巧手设备 ID 为 0x01，12 个自由度，控制模式为速度力控模式，一次性可以下发 12 个关节的目标信息；

- 请求的每一个关节的 bag 组成为：关节 ID + 控制模式 + 目标速度 + 目标力距共 5Byte；
- 应答的每一个关节的 bag 组成为：关节 ID + 控制模式 + 实际速度 + 实际力距共 5Byte；
    - 请求当前所有的关节控制模式为速度力控模式，目标速度 1000，目标力距 1000；
    - 应答当前所有的关节控制模式为速度力控模式，实际速度 1000，实际力距 500；

	CANFD ID	Length	Data
请求	0x00140181	64Byte(60Byte有效)	80 E8 03 E8 03 81 E8 03 E8 03 82 E8 03 E8 03 83 E8 03 E8 03 84 E8 03 E8 03 85 E8 03 E8 03 86 E8 03 E8 03 87 E8 03 E8 03 88 E8 03 E8 03 89 E8 03 E8 03 8A E8 03 E8 03 8B E8 03 E8 03 00 00 00 00
应答	0x00140101	64Byte(60Byte有效)	80 E8 03 F4 01 81 E8 03 F4 01 82 E8 03 F4 01 83 E8 03 F4 01 84 E8 03 F4 01 85 E8 03 F4 01 86 E8 03 F4

			01 87 E8 03 F4 01 88 E8 03 F4 01 89 E8 03 F4 01 8A E8 03 F4 01 8B E8 03 F4 01 00 00 00 00

混合控制示例 3:

灵巧手设备 ID 为 0x01，12 个自由度，控制模式为位置速度力控模式，最多一次性可以下发 8 个关节的目标信息；

- 请求的每一个关节的 bag 组成为：关节 ID + 控制模式 + 目标位置 + 目标速度 + 目标力距共 7Byte；
- 应答的每一个关节的 bag 组成为：关节 ID + 控制模式 + 实际位置 + 实际速度 + 实际力距共 7Byte；
    - 请求当前所有的关节控制模式为位置速度力控模式，目标位置 1000，目标速度 1000，目标力距 1000；
    - 应答当前所有的关节控制模式为位置速度力控模式，实际位置 500，实际速度 1000，实际力距 500；

	CANFD ID	Length	Data
请求	0x00140181	64Byte(56Byte 有效)	A0 E8 03 E8 03 E8 03 A1 E8 03 E8 03 E8 03 A2 E8 03 E8 03 E8 03 A3 E8 03 E8 03 E8 03 A4 E8 03 E8 03 E8 03 A5 E8 03 E8 03 E8 03 A6 E8 03 E8 03 E8 03 A7 E8 03 E8 03 E8 03 00 00 00 00 00 00 00 00
应答	0x00140101	64Byte(56Byte 有效)	A0 F4 01 E8 03 F4 01 A1 F4 01 E8 03 F4 01 A2 F4 01 E8 03 F4 01 A3 F4 01 E8 03 F4 01 A4 F4 01 E8 03 F4 01 A5 F4 01 E8 03 F4 01 A6 F4 01 E8 03 F4 01 A7 F4 01 E8 03 F4 01 00 00 00 00 00 00 00 00

–

_

手势控制动作编排示例:

灵巧手设备 ID 为 0x01，12 自由度，设定用户自定义手势 1 的动作为所有的关节都运动到 1000 位置；

	CANFD ID	Length	Data(HEX)
请求	0x0A150181	24Byte	E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03
应答	0x0A150101	24Byte	E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03

手势控制动作执行示例:

灵巧手设备 ID 为 0x01，12 自由度，执行用户自定义手势 1 预设动作；

	CANFD ID	Length	Data(HEX)
请求	0x0A150101	0Byte	
应答	0x0A150101	24Byte	E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03 E8 03

循环控制序列编排示例 ：

灵巧手设备 ID 为 0x01，设定用户自定义手势序列为执行数字 1->延迟 1s->执行数字 2->延迟 1s->执行数字 3->延迟 1s；

	CANFD ID	Length	Data(HEX)

请求	0x00160181	6Byte	00 7D 01 7D 02 7D
应答	0x00160101	6Byte	00 7D 01 7D 02 7D

循环控制序列执行示例:

灵巧手设备 ID 为 0x01，执行用户自定义手势序列；

	CANFD ID	Length	Data(HEX)
请求	0x00160101	0Byte	
应答	0x00160101	6Byte	00 7D 01 7D 02 7D

灵巧手设备 ID 为 0x01，设定用户自定义手势序列为 执行数字 1->延迟 1s->执行数字 2->延迟 1s->执行数字 3->延迟 1s；

	CANFD ID	Length	Data(HEX)
请求	0x00160181	6Byte	00 7D 01 7D 02 7D
应答	0x00160101	6Byte	00 7D 01 7D 02 7D

循环控制序列执行示例:

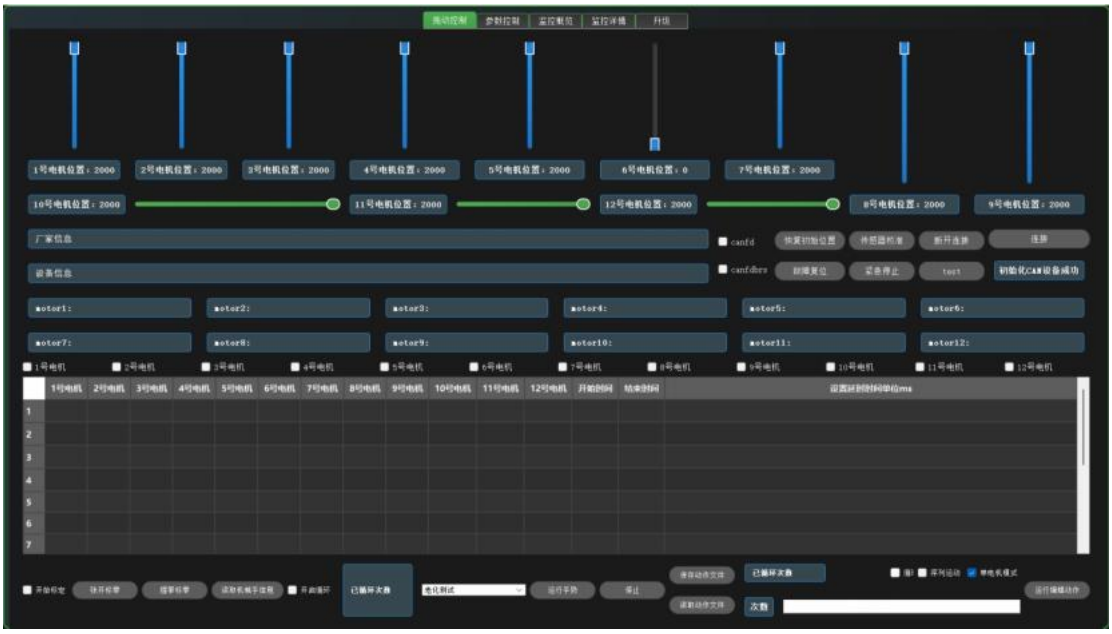
灵巧手设备 ID 为 0x01，执行用户自定义手势序列；

	CANFD ID	Length	Data(HEX)
请求	0x00160101	0Byte	
应答	0x00160101	6Byte	00 7D 01 7D 02 7D

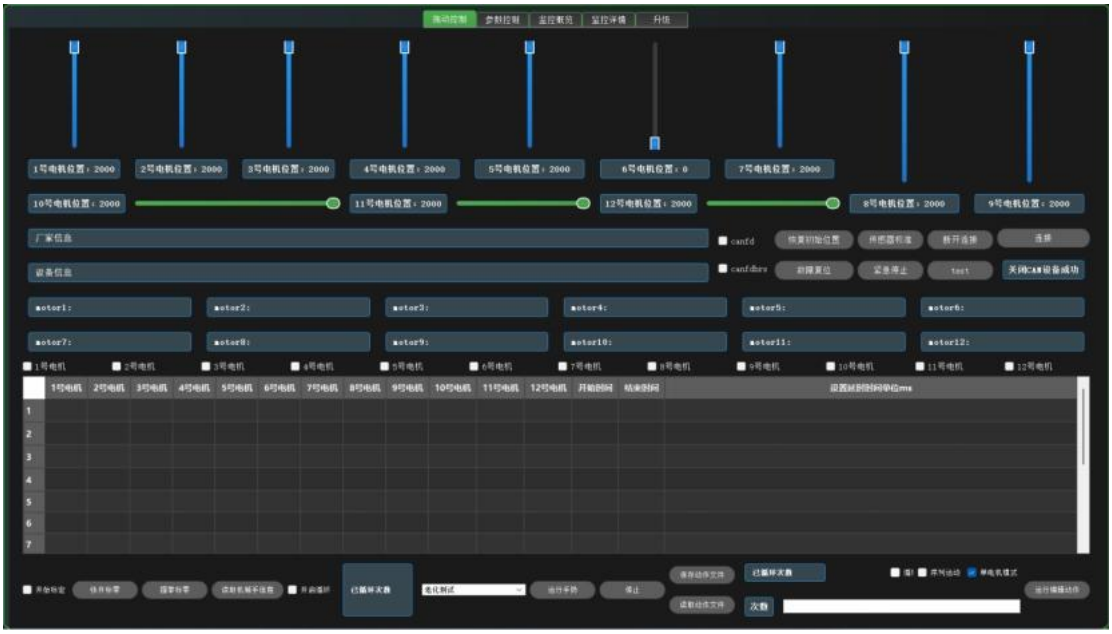
5. 上位机使用说明

5.1 上位机连接

所需电脑系统为 Windows 10 及以上



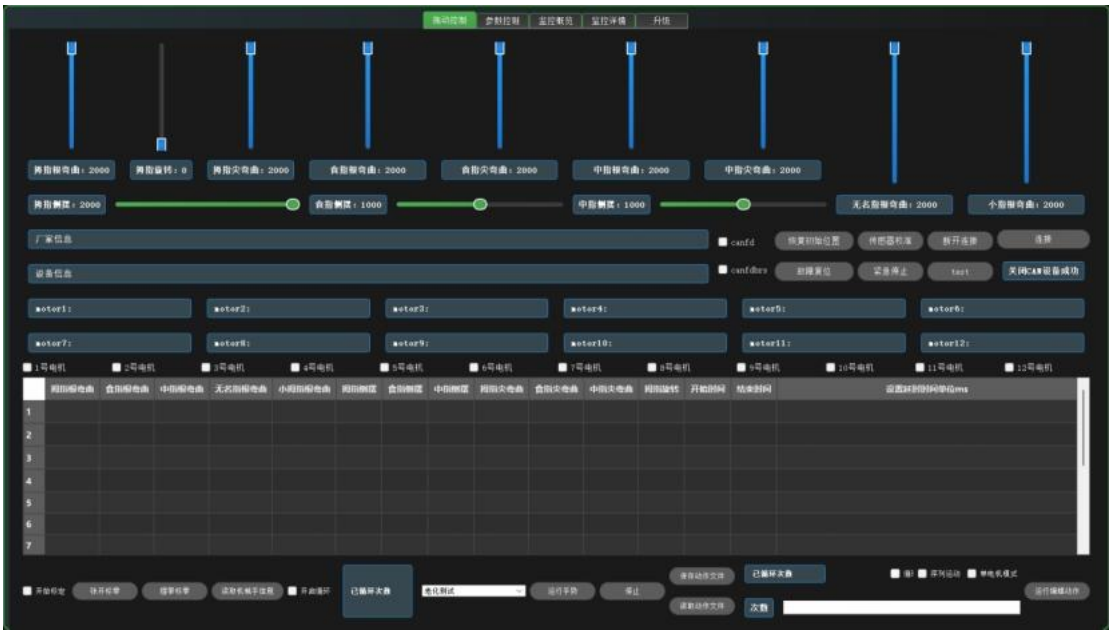
点击**连接**按钮，连接状态栏显示初始化 CAN 设备成功。



点击**断开连接**，连接状态栏显示关闭 CAN 设备成功。

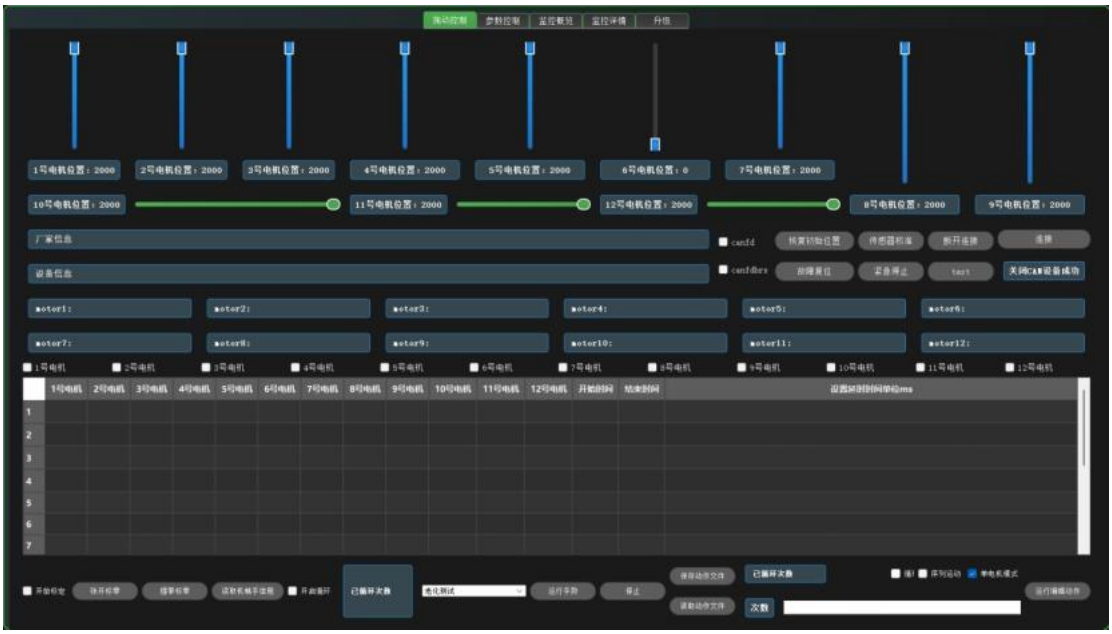
5.2 上位机滑动条控制

5.2.1 关节控制



连接灵巧手后，可以通过滑动条控制对应关节。

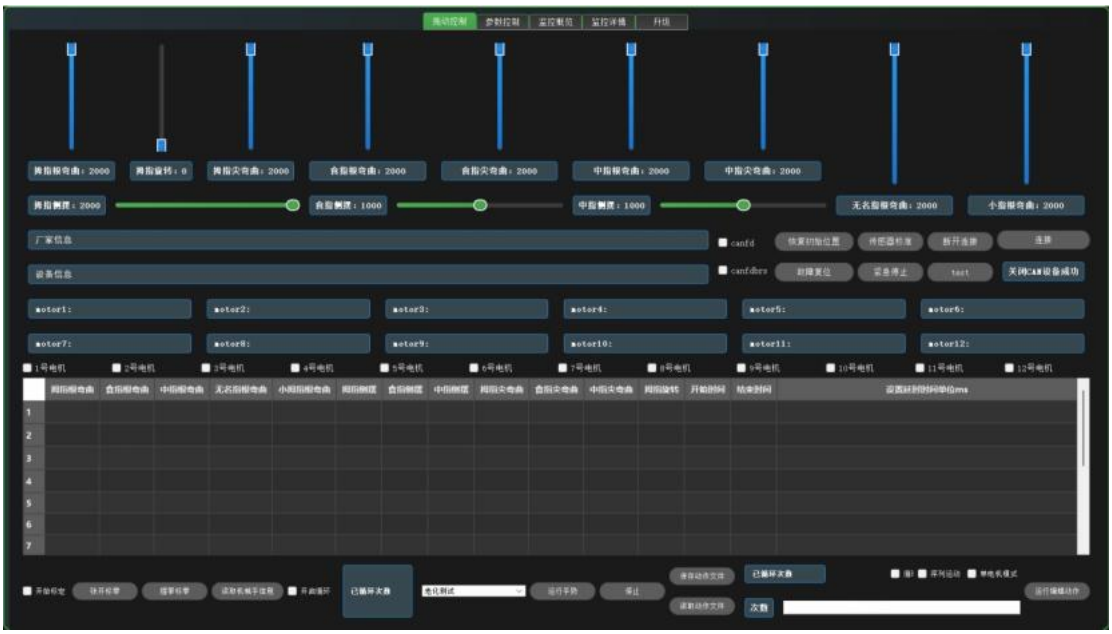
5.2.2 单电机控制

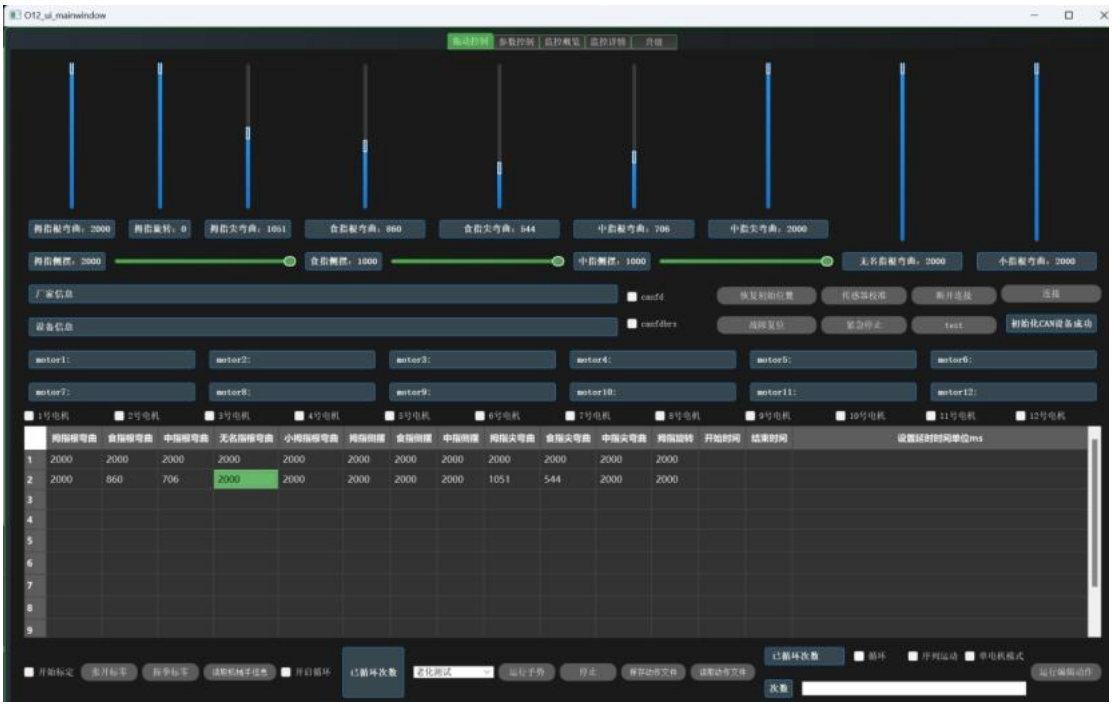


勾选**单电机模式**，可以改成单控个别电机。

5.3 表格控制

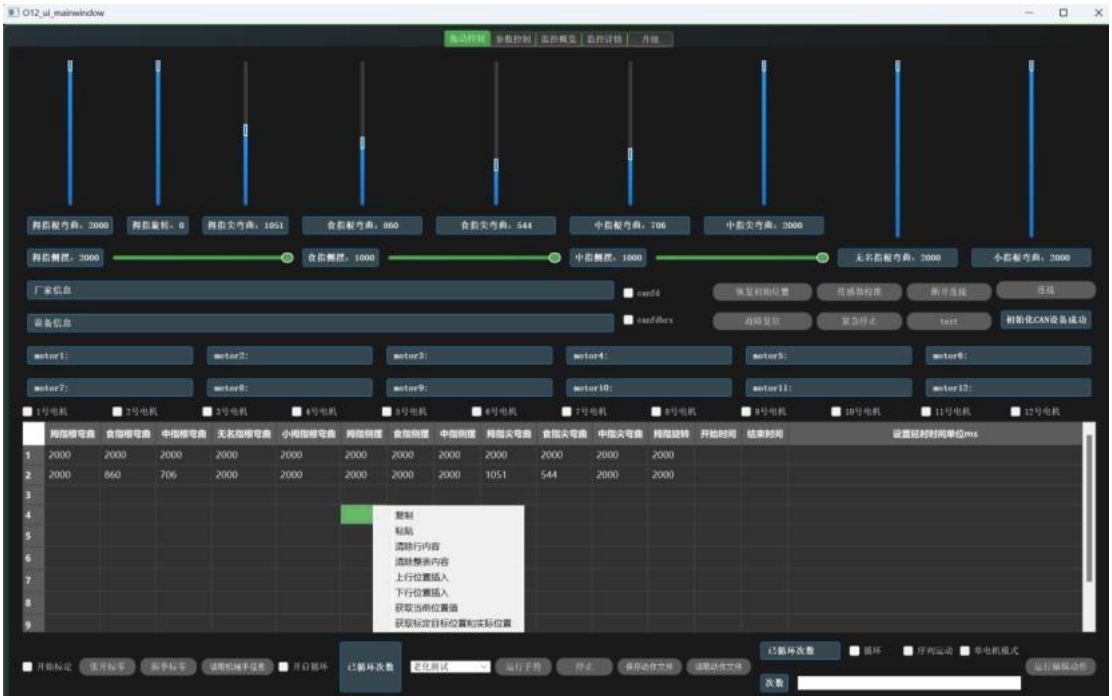
5.3.1 关节控制





表格的每一列对应一个关节，将一行的第一个到第十二个单元格填写上数值，则构成一个完整的手部动作。数值可填写区间为 **0~2000** 。第十三个单元格和第十四个单元格显示的是动作运行的开始和结束时间，第十五个单元格用于设置延时时间，不填写情况下默认 1000 毫秒。

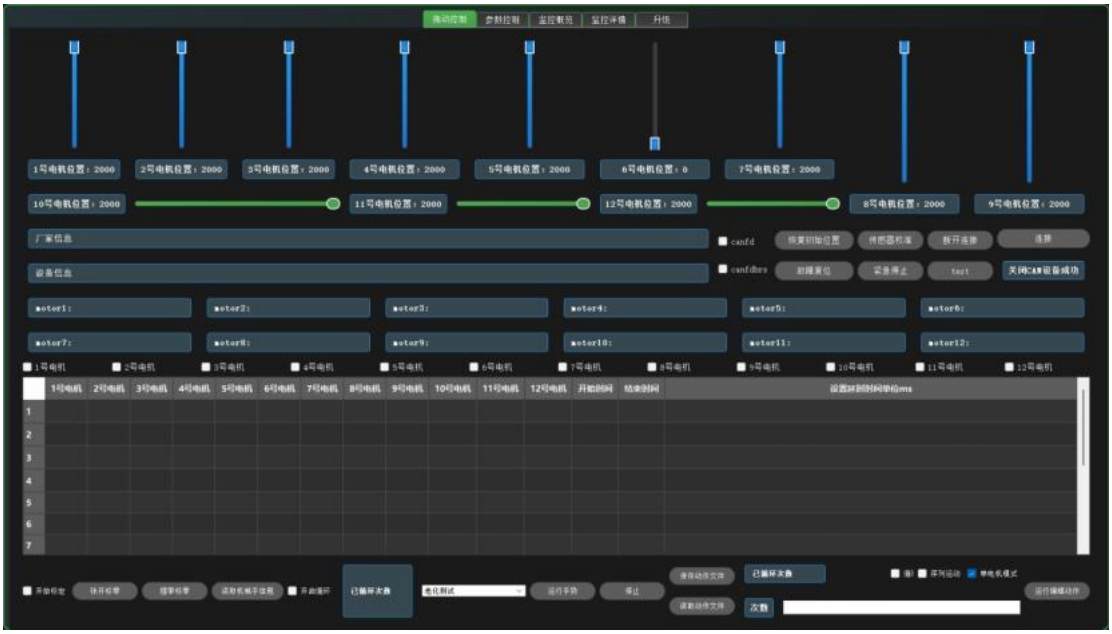
鼠标左键选中要填写的单元格，点击鼠标右键，在出现的菜单中，选择获取当前位置值，即可将该列对应的关节滑动条显示的当前位置值输入到该单元格中。点击**运行编辑动作**按钮，即可执行选中行的动作。



勾选**序列运动**，点击**运行编辑动作**按钮，可以执行第一行到最后一行的所有动作。

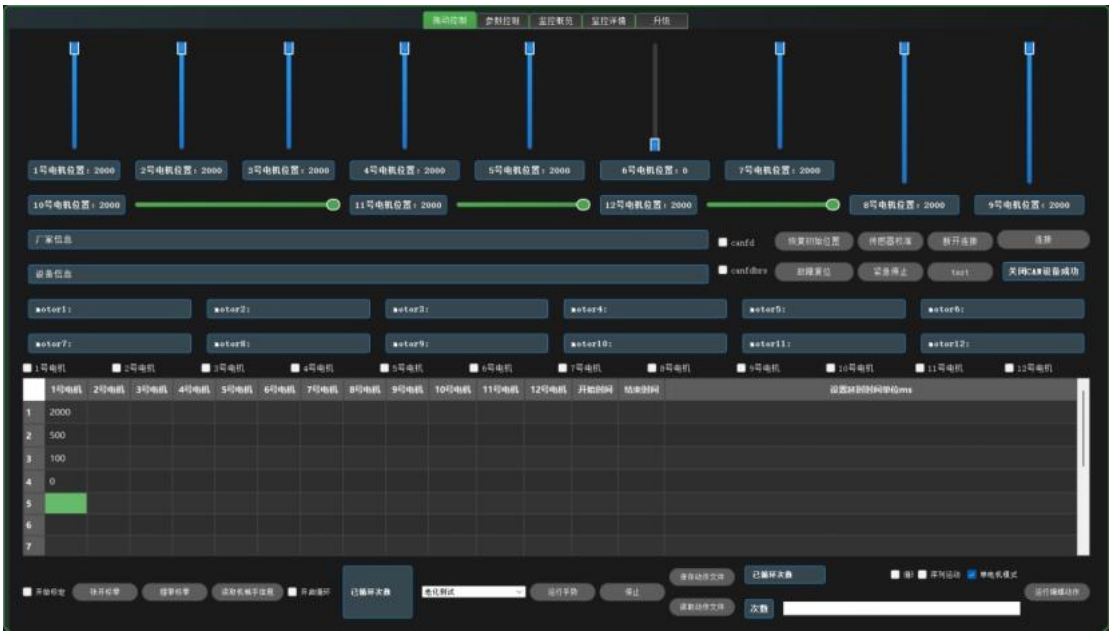
勾选**循环**，并在次数输入栏输入大于 0 的数字，点击**运行编辑动作**按钮，可以按输入次数，循环执行第一行到最后一行的所有动作。

5.3.2 单电机控制



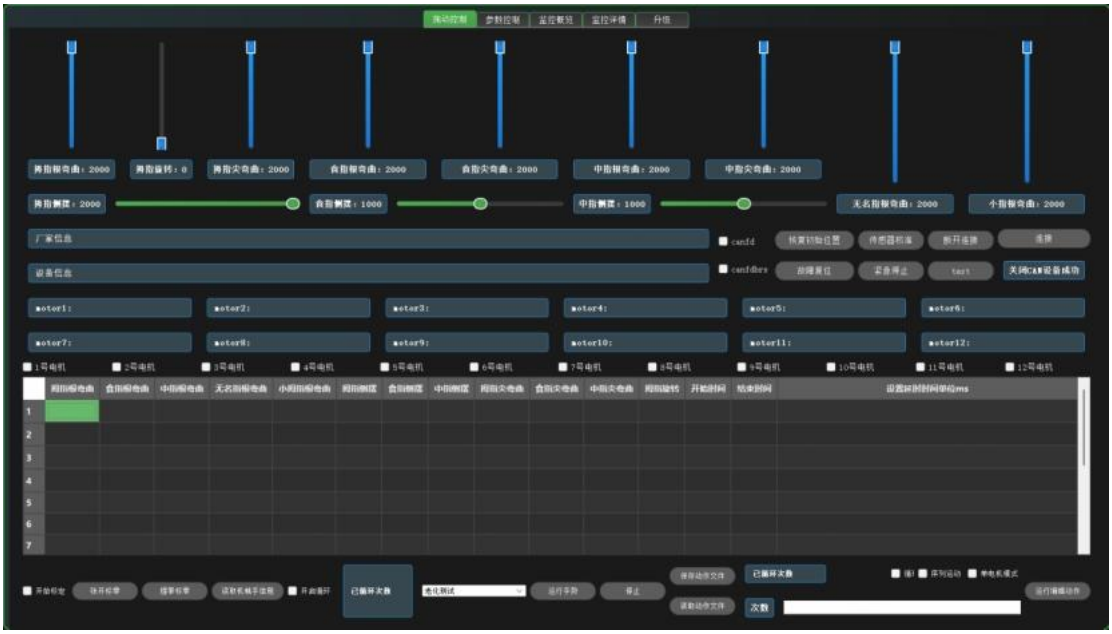
勾选**单电机模式**，表格切换到对应的电机标题。

首先勾选要控制电机的勾选框，比如 1 号电机。在表格对应的 1 号电机列输入要控制的 0~2000 的位置值，选中要运动的位置行，点击**运行编辑动作**按钮，对应电机运动到输入目标位置。比如鼠标左键选择第一行，点击**运行编辑动作**按钮，一号电机运行到 2000 位置。



序列运动，循环同表格关节控制。

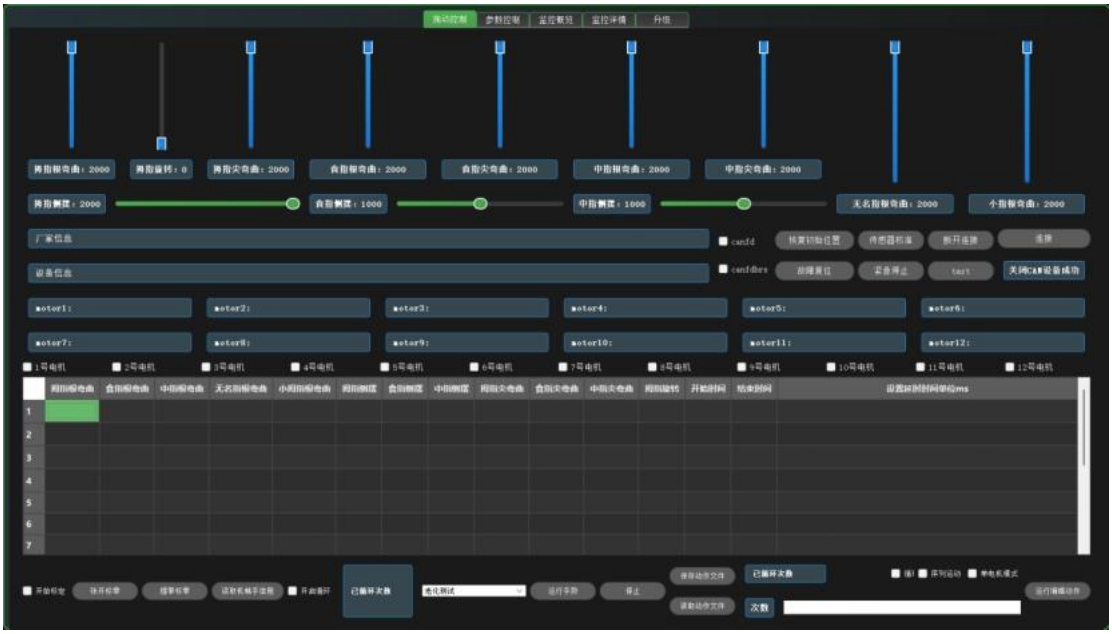
5.3.3 动作记录



点击**保存动作**按钮，可以将表格里的位置数据保存到 .txt 文件里。

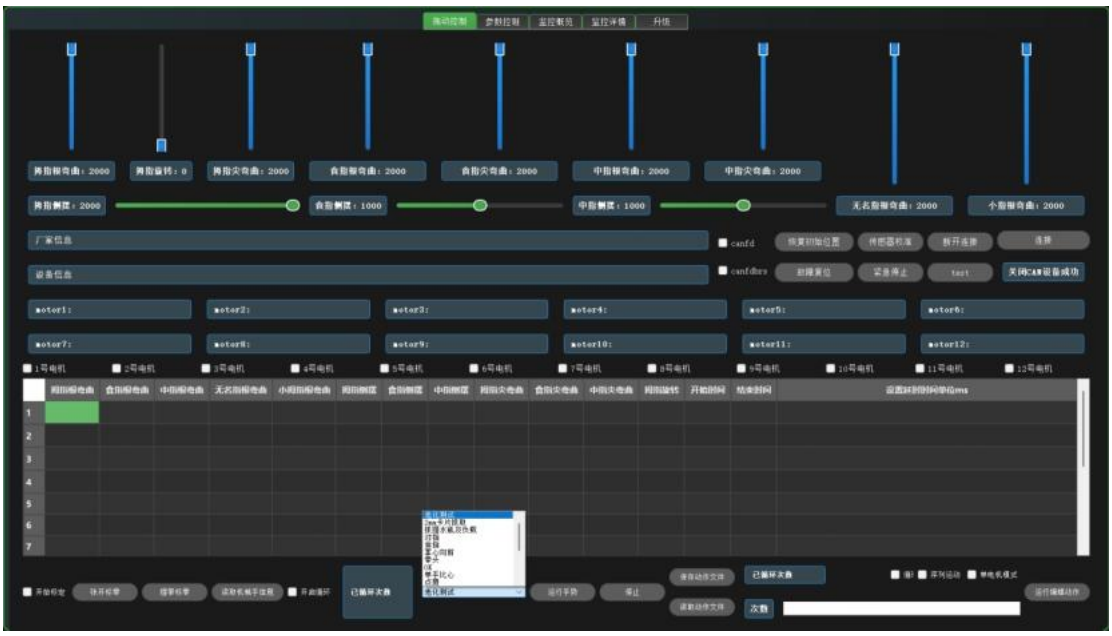
点击**读取动作**按钮，可以将保存的动作文件在表格里显示并运行。

5.3.4 紧急停止



循环过程中，可以点击**紧急停止**按钮停止循环。**恢复初始位置**按钮会让手张开，**故障复位**会发送故障复位命令，并解除紧急停止的停止锁。

5.4 预设动作

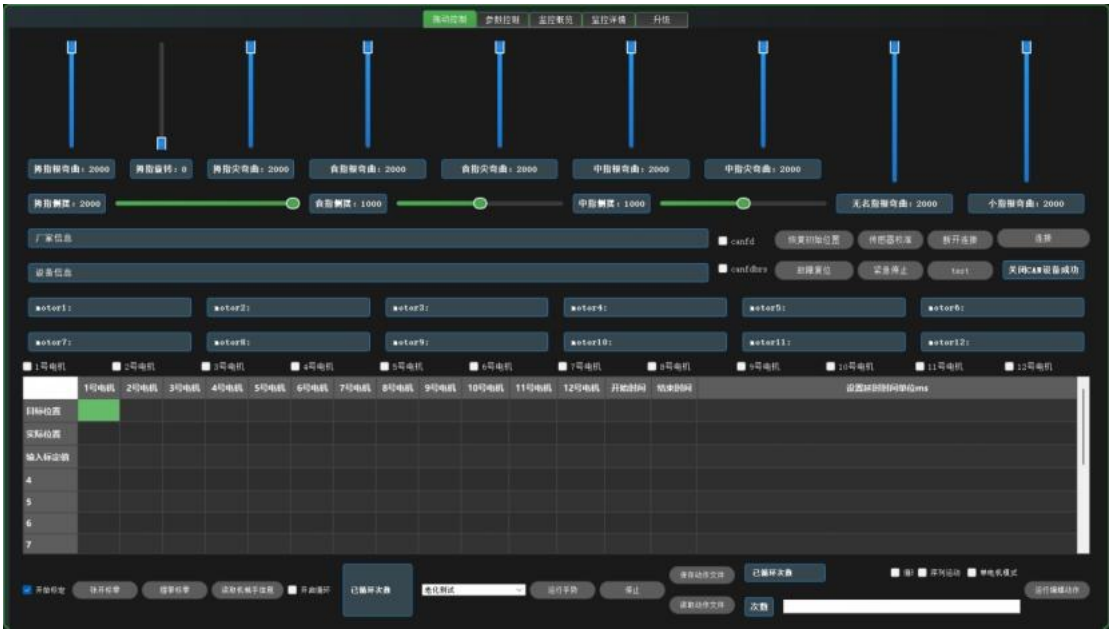


点击**运行手势**按钮，可以运行预设的某一个手势。

勾选**循环**，输入次数，点击**运行手势**按钮，会循环所有预设手势。点击**停止**按钮可以停止

循环。

5.5 标零

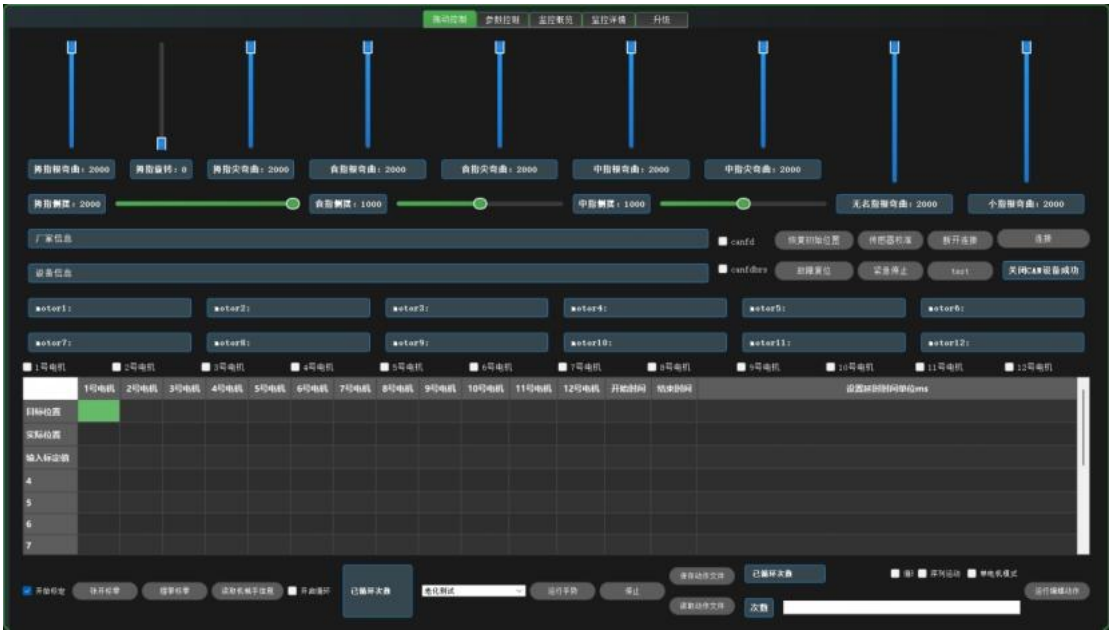


勾选左下角的**开始标定**，鼠标右键点击获取标定目标位置和实际位置，在输入标定值行输入对应电机的标定值，并勾选对应的电机勾选框。

点击**张开标零**，设置张开零点。

点击**握拳标零**，设置握拳零点。

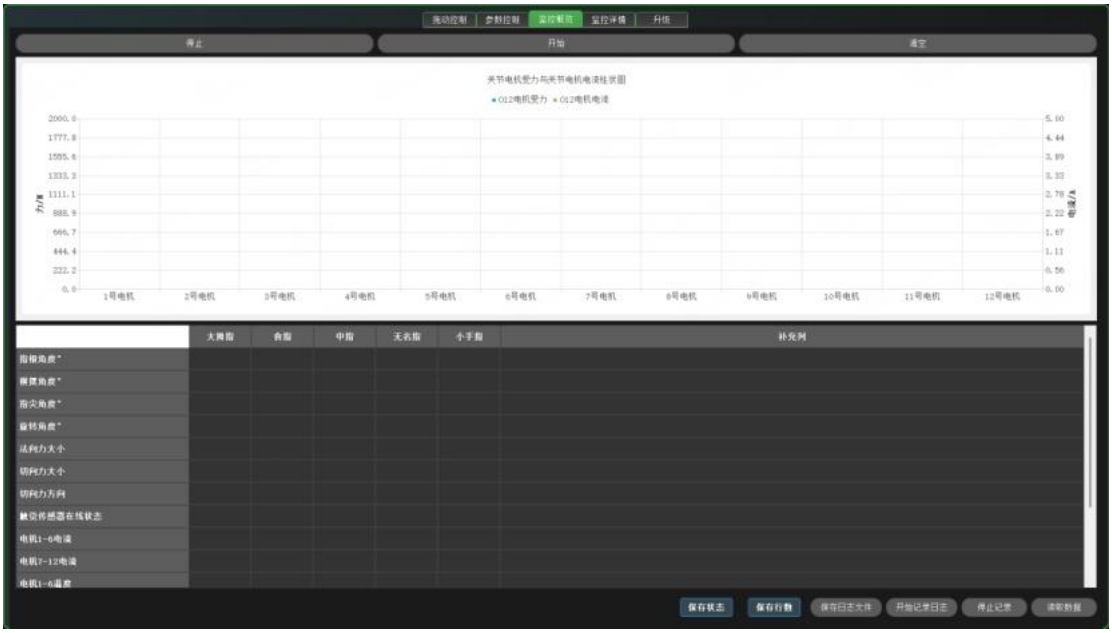
5.6 信息读取和故障上报



点击 **读取机械手信息**，可以在厂家信息栏和设备信息栏显示具体信息。

motor1 显示栏 至 motor12 显示栏 用于显示上位机主动上报的故障码。

5.7 监控概览

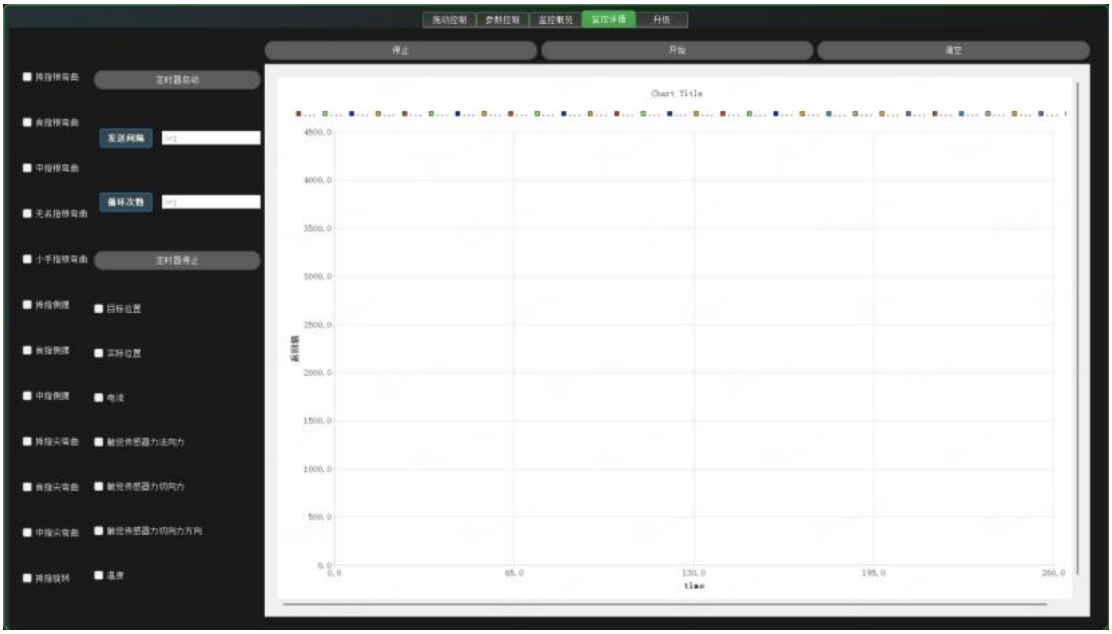


点击**开始**按钮，开始显示力和电流值。

点击**读取数据**按钮，可以在表格显示对应的数据。

点击**开始记录日志**按钮，可以记录温度，电流，错误码，力矩的值，点击保存日志文件将保存记录的数据，如果不点击保存日志文件按钮，每 5 分钟，会自动保存一次，保存的文件会放置在上位机程序文件夹内。

5.8 监控详情

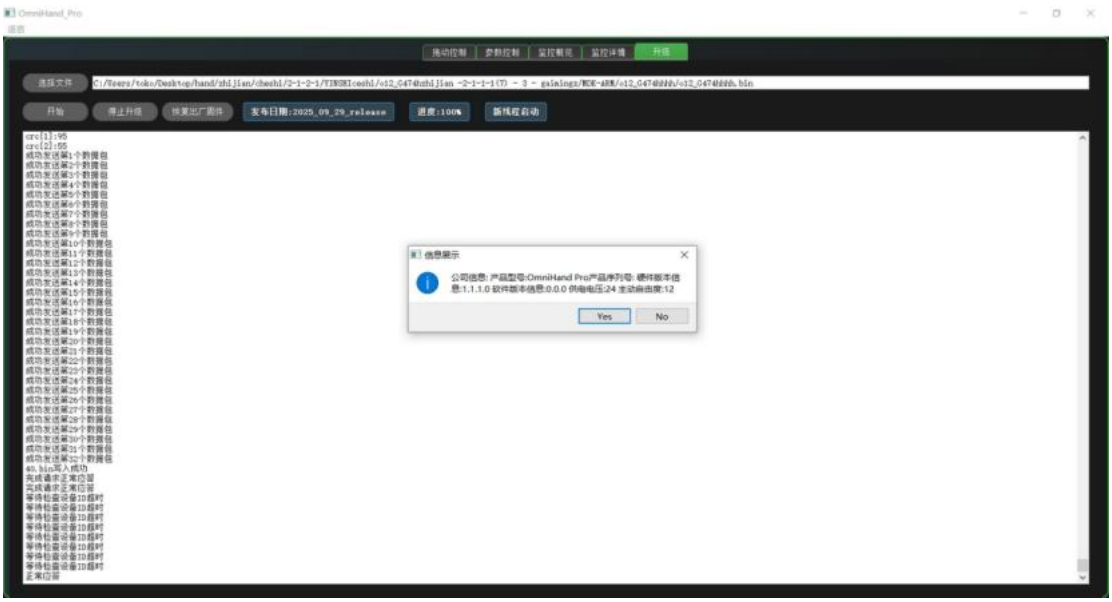


勾选要查看的信息类型，然后勾选对应的手指或电机，点击**开始**，显示曲线。

5.9 升级

点击**选择文件**按钮，选择要传输的 bin 文件。

点击**开始**按钮，会弹出软件和硬件信息。点击 **Yes** 按钮，开始传输固件。点击 **No** 按钮，取消升级。



停止升级和恢复出厂都能在升级过程中，和断电重连的情况下点击。

停止升级的作用是可以在中断升级过程，运行上一个应用程序。

恢复出厂固件是恢复出厂的固件。