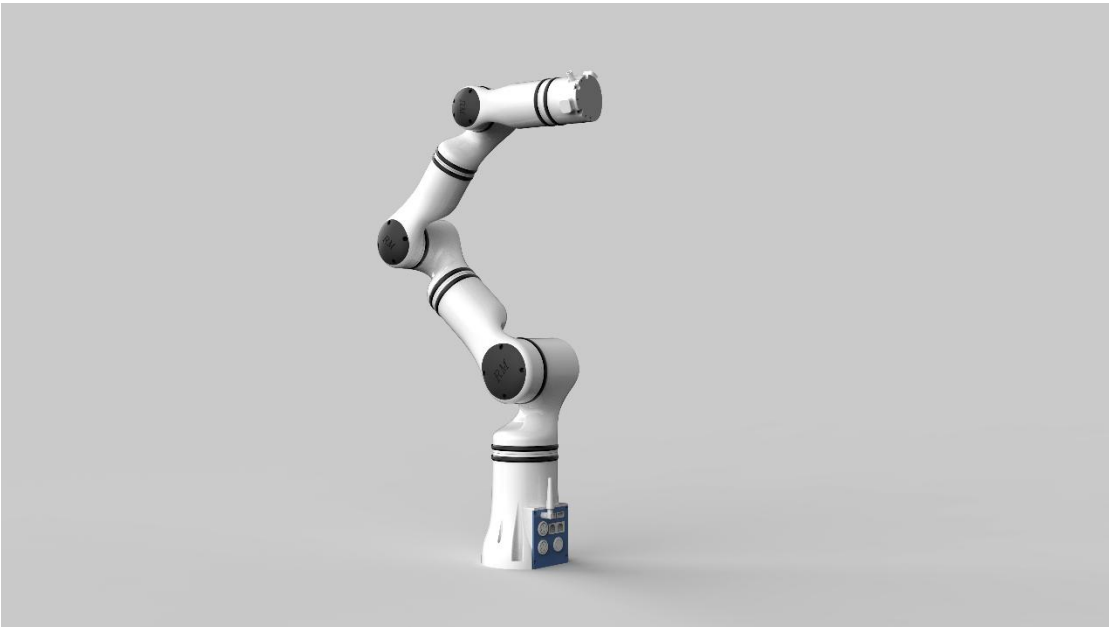
# 睿尔曼机械臂接口函数说明（C++）V4.0.6



睿尔曼智能科技（北京）有限公司

# 文件修订记录：

| 版本号 | 时间 | 备注 |
|---|---|---|
| V1.0 | 2020-05-01 | 拟制 |
| V1.1 | 2020-05-10 | 修订 |
| V1.2 | 2020-05-15 | 修订（通用化修订） |
| V1.3 | 2020-05-17 | 格式及说明整理 |
| V1.4 | 2020-05-25 | 格式整理 |
| V1.5 | 2020-06-05 | 修改 WIFI 配置流程 |
| V1.6 | 2020-06-30 | 加入 IO 部分 |
| V1.7 | 2020-07-03 | 修改部分协议名称 |
| V1.8 | 2020-08-04 | 加入末端接口板部分协议 |
| V1.9 | 2021-03-12 | 加入路径点控制等相关函数 |
| V2.0 | 2021-05-20 | 加入 Movej_P;末端 PWM 设置；末端一维力设置 |
| V2.1 | 2021-09-17 | 增加 Modbus 协议内容 |
| V2.2 | 2021-09-27 | 增加防碰撞等级设置；移动平台及升降机构内容 |
| V2.3 | 2021-10-19 | 勘误 |
| V2.4 | 2022-01-28 | 更新六维力一维力函数代码 |
| V2.5 | 2022-04-12 | 勘误 |
| V2.6 | 2022-04-25 | 新增位姿透传等 |
| V3.0 | 2022-05-13 | 合并 RM65 与 RM75 接口；新增正逆解接口； 优化部分接口 |
| V3.1 | 2022-06-09 | 勘误 |
| V3.2 | 2022-07-12 | Modbus 协议部分增加多圈和多寄存器操作 |
| V4.0 | 2022-09-23 | 重构 API 架构 |
| V4.0.2 | 2022-10-31 | 增加算法工具接口 |

| | | |
|---|---|---|
| V4.0.4 | 2022-11-31 | 修改初始化 API 设置型号，优化部分接口 |
| V4.0.6 | 2022-02-08 | 优化夹爪相关接口阻塞模式，优化获取机械臂全部状态接口 |

# 目录

## 1. 简介

为了方便用户通过上位机自主开发程序控制机械臂，睿尔曼提供了基于TCP/IP socket 接口函数，用户可通过 WIFI（AP 模式或者 STA 模式）、以太网口与机械臂建立通信，并控制机械臂。

## 2. 功能介绍

接口函数分为 Windows 版本和 Linux 版本，可直接加载到 C、C++、C#或者 Python 工程中使用，接口函数将用户指令封装成标准的 JSON 格式下发给机械臂，并解析机械臂回传的数据提供给用户。

接口函数基于 TCP/IP 协议编写，其中：

机械臂 IP 地址：192.168.1.18，端口号：8080

无论是 WIFI 模式还是以太网口模式，机械臂均以该 IP 和端口号对外进行 socket 通信，机械臂为 Server 模式，用户为 Client 模式。另外，为了方便用户调试，机械臂还可切换到 USB 模式，机械臂通过 UART-USB 接口对外通信，用户可通过 USB 线直连电脑，通过上位机串口调试助手根据《JSON 控制协议》直接发送字符串控制机械臂，不过在 USB 模式下不可使用该接口函数。

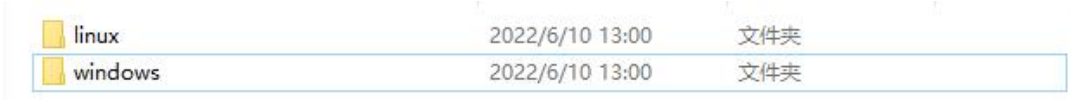
## 3. 使用说明

接口函数包内包括两个文件夹：(两个常用版本使用说明)

（1）linux：Linux 操作系统接口函数；

（2）windows：Windows 操作系统接口函数。

两部分除了调用系统 Socket 库稍有区别之外，其他部分完全相同。



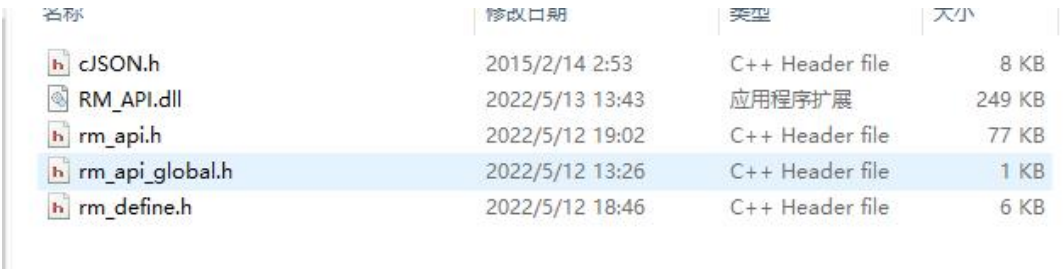

API 组成如下图所示:

（1）cJSON.h：cJSON 库的头文件，定义了 cJSON 库的结构体、数据类型和函数。

（2）rm_define.h：机械臂自定义头文件，包含了定义的数据类型、结构体和错误代码。

（3）rm_api.h ; rm_api_global.h：接口函数定义。

（4）rm_API.dll：接口函数实现。

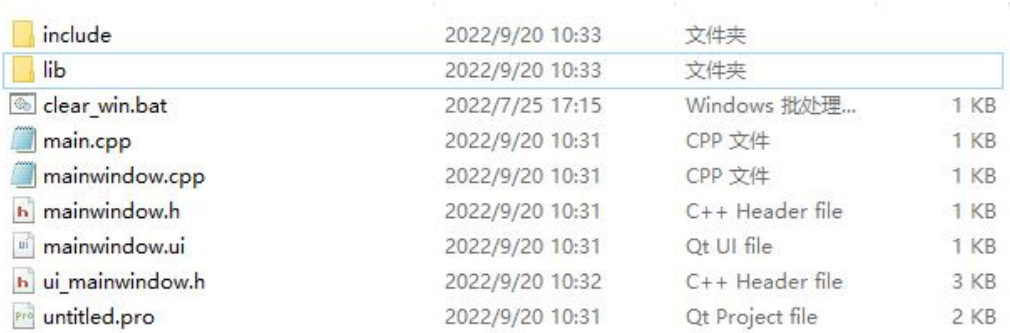


| 名称 | 修改日期 | 类型 | 大小 |
|---|---|---|---|
| cJSON.h | 2015/2/14 2:53 | C++ Header file | 8 KB |
| RM_API.dll | 2022/5/13 13:43 | 应用程序扩展 | 249 KB |
| rm_api.h | 2022/5/12 19:02 | C++ Header file | 77 KB |
| rm_api_global.h | 2022/5/12 13:26 | C++ Header file | 1 KB |
| rm_define.h | 2022/5/12 18:46 | C++ Header file | 6 KB |

API 的头文件中已做了处理，可直接加载到工程中使用。

使用步骤：

步骤一：将我们所提供的 include 以及 lib 文件夹拷贝到工程目录下。



| | | | |
|---|---|---|---|
| include | 2022/9/20 10:33 | 文件夹 | |
| lib | 2022/9/20 10:33 | 文件夹 | |
| clear_win.bat | 2022/7/25 17:15 | Windows 批处理... | 1 KB |
| main.cpp | 2022/9/20 10:31 | CPP 文件 | 1 KB |
| mainwindow.cpp | 2022/9/20 10:31 | CPP 文件 | 1 KB |
| mainwindow.h | 2022/9/20 10:31 | C++ Header file | 1 KB |
| mainwindow.ui | 2022/9/20 10:31 | Qt UI file | 1 KB |
| ui_mainwindow.h | 2022/9/20 10:32 | C++ Header file | 3 KB |
| untitled.pro | 2022/9/20 10:31 | Qt Project file | 2 KB |

步骤二：在 pro 文件中写入头文件以及 dll 文件路径。

```
31 INCLUDEPATH += $$PWD/include
32 LIBS += -L$$PWD/lib -lRM_Base
```

步骤三：添加完成后在文件中引入相应头文件即可使用。

```
#include "rm_base.h"
```

代码使用示例：

```
    // 初始化API
    RM_API_Init( 0);

    // 连接服务器
    m_sockhand =  Arm_Socket_Start("192.168.1.18", 8080, 65, 5000);

    int ret = -1;
    // 网络连接状态
    ret = Arm_Sockrt_State(m_sockhand);
    std::cout<<m_sockhand<<std::endl;

    // 关节空间运动
    float joint[6] = {0,0,0,0,90,0};
    ret = Movej_Cmd(m_sockhand,joint,20,0,1);

    //关闭连接
    Arm_Socket_Close(m_sockhand);
    m_sockhand = -1;
```

```
4.0
0
```

使用流程如下所示：

（1）通过 WIFI 或者以太网口与机械臂连接，保证上位机与机械臂控制器在同一网段内；

（2）调用 Service_RM_API_Init 函数初始化 API。设置机械臂型号以及接收透传接口回调函数，不需要可以传入 NULL。

（3）调用 Arm_Socket_Start()函数，与机械臂进行 socket 连接。注意，经测试在 Windows 操作系统下，可能需要 2~3 次才能建立连接，因此根据该函数的返回值判断是否需要再次调用来保证 socket 连接成功。

（3）连接成功后，用户根据需要调用接口函数。

（4）使用结束后，调用 Arm_Socket_Close()函数关闭 socket 连接，释放系统资源。

# 4. 宏定义

## 4.1 控制器错误类型

| 序号 | 错误代码（16 进制） | 错误内容 |
|---|---|---|
| 1 | 0x0000 | 系统正常 |
| 2 | 0x1001 | 关节通信异常 |
| 3 | 0x1002 | 目标角度超过限位 |
| 4 | 0x1003 | 该处不可达，为奇异点 |
| 5 | 0x1004 | 实时内核通信错误 |

| | | |
|---|---|---|
| 6 | 0x1005 | 关节通信总线错误 |
| 7 | 0x1006 | 规划层内核错误 |
| 8 | 0x1007 | 关节超速 |
| 9 | 0x1008 | 末端接口板无法连接 |
| 10 | 0x1009 | 保留，未定义 |
| 11 | 0x100A | 保留，未定义 |
| 12 | 0x100B | 关节抱闸未打开 |
| 13 | 0x100C | 拖动示教时超速 |
| 14 | 0x100D | 机械臂发生碰撞 |
| 15 | 0x100E | 无该工作坐标系 |
| 16 | 0x100F | 无该工具坐标系 |
| 17 | 0x1010 | 关节发生掉使能错误 |

## 4.2 关节错误类型

| 序号 | 错误代码（16 进制） | 错误内容 |
|---|---|---|
| 1 | 0x0000 | 关节正常 |
| 2 | 0x0001 | FOC 错误 |
| 3 | 0x0002 | 过压 |
| 4 | 0x0004 | 欠压 |
| 5 | 0x0008 | 过温 |
| 6 | 0x0010 | 启动失败 |
| 7 | 0x0020 | 初始化定位失败 |
| 8 | 0x0040 | 过流 |
| 9 | 0x0080 | 软件错误 |
| 10 | 0x0100 | 温度传感器错误 |
| 11 | 0x0200 | 位置传感器错误 |
| 12 | 0x0400 | 驱动芯片错误 |
| 13 | 0x0800 | 位置跟踪错误 |
| 14 | 0x1000 | 电流检测错误 |
| 15 | 0x2000 | 抱闸打开失败 |
| 16 | 0x8000 | 温升异常 |
| 17 | 0xF000 | 通信丢帧 |

## 4.4 数据结构定义

### 4.4.1 位姿结构体 POSE

```
typedef struct
{
```

```
    //位置
    float px;
    float py;
    float pz;
    //欧拉角
    float rx;
    float ry;
    float rz;
}POSE;
```

**结构体成员**

**px,py,pz**

位置坐标，float 类型，单位：m

**rx,ry,rz**

姿态角，float 类型，单位：rad

### 4.4.2 坐标系结构体 FRAME

```
typedef struct
{
char *frame_name;
POSE pose;
float   payload;
float   x;
float   y;
float   z
}FRAME;
```

**结构体成员**

**frame_name**

坐标系名称，字符串

**Pose**

坐标系位姿，参考 1.5.1

**payload**

末端负载重量　单位 g

**x,y,z**

末端负载位置

### 4.4.3 关节状态结构体 JOINT_STATE

```
typedef struct
{
    float joint[6];
    float temperature[6];
    float voltage[6];
    float current[6];
    byte en_state[6];
    uint16_t err_flag[6];
    uint16_t sys_err;
}JOINT_STATE;
```

**结构体成员**

**joint**

关节角度，每个关节角度，float 类型，单位：度

**temperature**

关节温度，float 类型，单位：摄氏度

**voltage**

关节电压，float 类型，单位：V

**current**

关节电流，float 类型，单位：mA

**en_state**

使能状态

**err_flag**

关节错误代码，unsigned int 类型

**sys_err**

机械臂系统错误代码，unsigned int 类型

### 4.4.4 机械臂控制模式枚举 ARM_CTRL_MODES

```
typedef enum
```

```
{
    None_Mode = 0,
    Joint_Mode = 1,
    Line_Mode = 2,
    Circle_Mode = 3,
}ARM_CTRL_MODES;
```

**枚举成员**

**None_Mode**

无规划模式

**Joint_Mode**

关节空间规划模式

**Line_Mode**

笛卡尔空间直线规划模式

**Circle_Mode**

笛卡尔空间圆弧规划模式

### 4.4.5 机械臂位置示教模式 POS_TEACH_MODES

```
typedef enum
{
    X_Dir = 0,
    Y_Dir = 1,
    Z_Dir = 2,
}POS_TEACH_MODES;
```

**枚举成员**

**X_Dir**

X 轴方向，默认 0

**Y_Dir**

Y 轴方向，默认 1

**Z_Dir**

Z 轴方向，默认 2

### 4.4.6 机械臂姿态示教模式 ORT_TEACH_MODES

```
typedef enum
{
    RX_Rotate = 0,
    RY_Rotate = 1,
    RZ_Rotate = 2,
}ORT_TEACH_MODES;
```

**枚举成员**

**RX_Rotate**

X 轴方向，默认 0

**RY_Rotate**

Y 轴方向，默认 1

**RZ_Rotate**

Z 轴方向，默认 2

### 4.4.7 控制器通讯方式选择 ARM_COMM_TYPE

控制器通讯方式选择

```
typedef enum
{
    WIFI_AP = 0,
    WIFI_STA = 1,
    BlueTeeth = 2,
    USB          = 3,
    Ethernet    =4
}ARM_COMM_TYPE;
```

**枚举成员**

**WIFI_AP**

WIFI 工作在 AP 模式

**WIFI_STA**

WIFI 工作在 STA 模式

**BlueTeeth**

蓝牙模式

**USB**

通过控制器 UART-USB 接口通信

**Ethernet**

以太网口

# 5. 接口库函数说明

该部分函数用来控制 socket 连接的打开与关闭。

## 5.1 连接相关函数

### 5.1.1 API 初始化 Service_RM_API_Init

该函数用于初始化 API

int Service_RM_API_Init(int devMode, RM_Callback pCallback);

**参数**

**①dev_mode**

目标设备型号 ARM_65/ARM_75/ARM_63_1/ARM_63_2。

**②pCallback**

用于接收透传接口回调函数，不需要可以传入 NULL。

### 5.1.2 API 反初始化 Service_RM_API_UnInit

该函数用于 API 反初始化

int Service_RM_API_UnInit();

### 5.1.3 查询 API 版本信息 Service_API_Version

该函数用于查询 API 当前版本

char *Service_ API_Version();

**返回值**

API 版本号。

### 5.1.4 连接机械臂 Service_Arm_Socket_Start

该函数用于连接机械臂

SOCKHANDLE Service_Arm_Socket_Start(char* arm_ip,int arm_port, int recv_timeout);

**参数**

**①arm_ip**

用于传入要连接机械臂的 IP 地址，机械臂 IP 地址默认为 192.168.1.18。

**②arm_prot**

用于传入要连接机械臂的端口，机械臂端口默认为 8080。

**③recv_timeout**

Socket 连接超时时间。

**返回值**

成功返回：Socket 句柄。失败返回:错误码, define.h 查询。

### 5.1.5 查询连接状态 Service_Arm_Sockrt_State

用于查询连接状态。

int Service_Arm_Sockrt_State(SOCKHANDLE ArmSocket);

**① ArmSocket**

Socket 句柄。

**返回值**

正常返回：SYS_NORMAL 异常:错误码, define.h 查询

### 5.1.6 关闭连接 Service_Arm_Socket_Close

该函数用于关闭与机械臂的 socket 连接。

void Service_Arm_Socket_Close(SOCKHANDLE ArmSocket);

**① ArmSocket**

Socket 句柄。

## 5.2 关节配置函数

睿尔曼机械臂在出厂前所有参数都已经配置到最佳状态，一般不建议用户修改关节的底层参数。若用户确需修改，首先应使机械臂处于非使能状态，然后再发送修改参数指令，参数设置成功后，发送关节恢复使能指令。需要注意的是，关节恢复使能时，用户需要保证关节处于静止状态，以免上使能过程中关节发生定位报错。关节正常上使能后，用户方可控制关节运动。

### 5.2.1 设置关节最大速度 Service_Set_Joint_Speed

该函数用于设置关节最大速度，单位：RPM。

int Service_Set_Joint_Speed(SOCKHANDLE ArmSocket, byte joint_num, float speed, bool block);

**参数**

① **ArmSocket**

Socket 句柄。

② **joint_num**

关节序号

③**speed**

关节转速，单位：RPM

④**block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.2.2 设置关节最大加速度 Service_Set_Joint_Acc

该函数用于设置关节最大加速度，单位：RPM/s。

int Service_Set_Joint_Acc(SOCKHANDLE ArmSocket, byte joint_num, float acc, bool block);

**参数**

① **ArmSocket**

Socket 句柄。

② **joint_num**

关节序号，1~6

③**acc**

关节转速，单位：RPM/s

④**block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.2.3 设置关节最小位置 Service_Set_Joint_Min_Pos

该函数用于设置关节所能到达的最小位置，单位：度。

int Service_Set_Joint_Min_Pos(SOCKHANDLE ArmSocket, byte joint_num, float joint, bool block);

**参数**

**① ArmSocket**

Socket 句柄。

**②joint_num**

关节序号，1~6

**③joint**

关节最小位置，单位：°

**④block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.2.4 设置关节最大位置 Service_Set_Joint_Max_Pos

该函数用于设置关节最大位置

int Service_Set_Joint_Max_Pos(SOCKHANDLE ArmSocket, byte joint_num, float joint, bool block);

**参数**

**① ArmSocket**

Socket 句柄。

**②joint_num**

关节序号，1~6

**③joint**

关节最大位置，单位：°

**④block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.2.5 设置关节使能 Service_Set_Joint_EN_State

该函数用于设置关节使能状态。

int Service_Set_Joint_EN_State(SOCKHANDLE ArmSocket, byte joint_num, bool state, bool block);

**参数**

**① ArmSocket**

Socket 句柄。

**②joint_num**

关节序号，1~6

**③state**

true-上使能，false-掉使能

**④block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.2.6 设置关节零位 Service_Set_Joint_Zero_Pos

该函数用于将当前位置设置为关节零位。

int Service_Set_Joint_Zero_Pos(SOCKHANDLE ArmSocket, byte joint_num, bool block);

**参数**

**① ArmSocket**

Socket 句柄。

**②joint_num**

关节序号，1~6

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令。

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.2.7 清除关节错误代码 Service_Set_Joint_Err_Clear

该函数用于清除关节错误代码。

int Service_Set_Joint_Err_Clear(SOCKHANDLE ArmSocket, byte joint_num, bool block);

**参数:**

**① ArmSocket**

Socket 句柄。

**②joint_num**

关节序号，1~6

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令。

**返回值:**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.2.8 开始关节标定 Service_Start_Calibrate

该函数用于开始关节标定。

int Service_Start_Calibrate(SOCKHANDLE ArmSocket, bool block);

**参数:**

**① ArmSocket**

Socket 句柄。

**②block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令。

**返回值:**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.2.9 结束关节标定 Service_Stop_Calibrate

该函数用于结束关节标定。

int Service_Stop_Calibrate(SOCKHANDLE ArmSocket, bool block);

**参数:**

**① ArmSocket**

Socket 句柄

**②block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令。

**返回值:**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.2.10 查询关节标定状态 Service_Get_Calibrate_State

该函数用于查询关节标定状态。

int Service_Get_Calibrate_State(SOCKHANDLE ArmSocket, float *joint_state, uint16_t *state,int* time);

**参数:**

**① ArmSocket**

Socket 句柄

**②joint_state**

各个关节状态 0-未标定 1-标定中 2-已标定 3-关节卡死 4-标定超时 5-报错。

**③err**

各个关节报错信息

0x0001FOC 频率过高
0x0010 启动过程失败
0x0100 温度传感器出错
0x0002 过压
0x0020 码盘错误
0x0200 位置超限错误
0x0004 欠压
0x0040 过流
0x0400 DRV8320 错误
0x0008 过温
0x0080 软件错误
0x0800 位置跟踪误差超限
0x1000 电流检测错误

④ **time**

标定时长(s)。

**返回值:**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.3 关节参数查询函数

### 5.3.1 查询关节最大速度 Service_Get_Joint_Speed

该函数用于查询关节最大速度。

int Service_Get_Joint_Speed(SOCKHANDLE ArmSocket, float *speed);

**参数**

① **ArmSocket**

Socket 句柄

② **speed**

存放关节 1~6 转速数组地址，单位：RPM

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.3.2 查询关节最大加速度 Service_Get_Joint_Acc

该函数用于查询关节最大加速度。

int Service_Get_Joint_Acc(SOCKHANDLE ArmSocket, float *acc);

**参数**

① **ArmSocket**

Socket 句柄

② **acc**

存放关节 1~6 加速度数组地址，单位：RPM/s

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.3.3 获取关节最小位置 Service_Get_Joint_Min_Pos

该函数用于获取关节最小位置。

int Service_Get_Joint_Min_Pos(SOCKHANDLE ArmSocket, float *min_joint);

**参数**

① **ArmSocket**

Socket 句柄

②**min_joint**

存放关节 1~6 最小位置数组地址，单位：°

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.3.4 获取关节最大位置 Service_Get_Joint_Max_Pos

该函数用于获取关节最大位置。

int Service_Get_Joint_Max_Pos(SOCKHANDLE ArmSocket, float *max_joint);

**参数**

① **ArmSocket**

Socket 句柄

②**max_joint**

存放关节 1~6 最大位置数组地址，单位：°

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.3.5 获取关节使能状态 Service_Get_Joint_EN_State

该函数用于获取关节使能状态。

int Service_Get_Joint_EN_State(SOCKHANDLE ArmSocket, byte *state);

**参数**

① **ArmSocket**

Socket 句柄

②**state**

存放关节 1~6 使能状态数组地址，1-使能状态，0-掉使能状态

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.3.6 获取关节错误代码 Service_Get_Joint_Err_Flag

该函数用于获取关节错误代码。

int Service_Get_Joint_Err_Flag(SOCKHANDLE ArmSocket, uint16_t *state);

**参数**

**① ArmSocket**

Socket 句柄

**②state**

存放关节错误代码

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.3.7 查询关节软件版本号 Service_Get_Joint_Software_Version

该函数用于查询关节软件版本号。

int Service_Get_Joint_Software_Version(SOCKHANDLE ArmSocket, float* version);

**参数**

**① ArmSocket**

Socket 句柄

**②version**

存放关节 1~6 软件版本号

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.4 机械臂末端运动参数配置

### 5.4.1 设置末端最大线速度 Service_Set_Arm_Line_Speed

该函数用于设置机械臂末端最大线速度。

int Service_Set_Arm_Line_Speed(SOCKHANDLE ArmSocket, float speed, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**speed**

末端最大线速度，单位 m/s

③**block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.2 设置末端最大线加速度 Service_Set_Arm_Line_Acc

该函数用于设置机械臂末端最大线加速度。

int Service_Set_Arm_Line_Acc(SOCKHANDLE ArmSocket, float acc, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**acc**

末端最大线加速度，单位 m/s^2

③**block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.3 设置末端最大角速度 Service_Set_Arm_Angular_Speed

该函数用于设置机械臂末端最大角速度。

int Service_Set_Arm_Angular_Speed(SOCKHANDLE ArmSocket, float speed, bool block);

**参数**

① **ArmSocket**

Socket 句柄

**②speed**

机械臂末端最大角速度，单位 rad/s

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.4 设置末端最大角加速度 Service_Set_Arm_Angular_Acc

该函数用于设置机械臂末端最大角加速度。

int Service_Set_Arm_Angular_Acc(SOCKHANDLE ArmSocket, float acc, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②acc**

末端最大角加速度，单位 rad/s^2

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.5 获取末端线速度 Service_Get_Arm_Line_Speed

该函数用于获取机械臂末端线速度。

int Service_Get_Arm_Line_Speed(SOCKHANDLE ArmSocket, float *speed);

**参数**

**① ArmSocket**

Socket 句柄

**②speed**

各个关节末端线速度

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.6 获取末端线加速度 Service_Get_Arm_Line_Acc

该函数用于获取机械臂末端线加速度。

int Service_Get_Arm_Line_Acc(SOCKHANDLE ArmSocket, float *acc);

**参数**

**① ArmSocket**

Socket 句柄

**②acc**

各个关节末端线加速度

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.7 获取末端角速度 Service_Get_Arm_Angular_Speed

该函数用于获取机械臂末端角速度。

int Service_Get_Arm_Angular_Speed(SOCKHANDLE ArmSocket, float *speed);

**参数**

**① ArmSocket**

Socket 句柄

**②speed**

各个关节末端角速度

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.8 获取末端角加速度 Service_Get_Arm_Angular_Acc

该函数用于获取机械臂末端角加速度。

int Service_Get_Arm_Angular_Acc(SOCKHANDLE ArmSocket, float *acc);

**参数**

① **ArmSocket**

Socket 句柄

②**acc**

各个关节末端角加速度

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.9 设置末端参数为初始值 Service_Set_Arm_Tip_Init

该函数用于设置机械臂末端参数为初始值。

int Service_Set_Arm_Tip_Init(SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**block:**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.10 设置碰撞等级 Service_Set_Collision_Stage

该函数用于设置机械臂动力学碰撞等级，等级 0~8，数值越高，碰撞检测越灵敏，同时也更易发生误碰撞检测。机械臂上电后默认碰撞等级为 0，即不检测碰撞。

int Service_Set_Collision_Stage (SOCKHANDLE ArmSocket, int stage, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**stage**

碰撞检测等级，等级：0~8

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.11 查询碰撞等级 Service_Get_Collision_Stage

该函数用于查询机械臂动力学碰撞等级，等级 0~8，数值越高，碰撞检测越灵敏，同时也更易发生误碰撞检测。机械臂上电后默认碰撞等级为 0，即不检测碰撞。

int Service_Get_Collision_Stage (SOCKHANDLE ArmSocket, int* stage);

**参数**

① **ArmSocket**

Socket 句柄

② **stage**

碰撞检测等级，等级：0~8

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.12 设置 DH 参数 Service_Set_DH_Data

该函数用于设置机械臂 DH 参数，一般用于对机械臂参数进行标定后使用。

int Service_Set_DH_Data (SOCKHANDLE ArmSocket, float lsb, float lse, float lew, float lwt, float d3, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **lsb, lse, lew, lwt, d3**

DH 参数，单位：mm

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

**备注：**该指令用户不可自行使用，必须配合测量设备进行绝对精度补偿时方可使用，否则会导致机械臂参数错误！

### 5.4.13 获取 DH 参数 Service_Get_DH_Data

该函数用于获取机械臂 DH 参数。

int Service_Get_DH_Data (SOCKHANDLE ArmSocket, float* lsb, float* lse, float* lew, float* lwt, float* d3);

**参数**

**① ArmSocket**

Socket 句柄

**②lsb, lse, lew, lwt, d3**

DH 参数，单位：mm

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.4.14 设置关节零位补偿角度 Service_Set_Joint_Zero_Offset

该函数用于设置机械臂各关节零位补偿角度，一般在对机械臂零位进行标定后调用该函数。

int Service_Set_Joint_Zero_Offset (SOCKHANDLE ArmSocket, float *offset, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②offset**

关节 1~6 零位补偿角度数组，角度单位：度

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

**备注：**该指令用户不可自行使用，必须配合测量设备进行绝对精度补偿时方可使用，否则会导致机械臂参数错误！

#### 5.4.15 设置动力学参数 Service_Set_Arm_Dynamic_Parm

该函数用于设置机械臂动力学参数。

int Service_Set_Arm_Dynamic_Parm (SOCKHANDLE ArmSocket, float *offset, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②offset**

关节 1~6 动力学参数

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.5 机械臂末端接口板

#### 5.5.1 查询末端接口板软件版本号 Service_Get_Tool_Software_Version

该函数用于查询末端接口板软件版本号

int Service_Get_Tool_Software_Version(SOCKHANDLE ArmSocket, int *version);

**参数**

**① ArmSocket**

Socket 句柄

**②version**

末端接口板软件版本号

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.6 机械臂状态伺服配置

### 5.6.1 设置伺服状态查询 Service_Set_Arm_Servo_State

该函数用于打开或者关闭控制器对机械臂伺服状态查询，控制 CANFD 总线的数据负载率。

int Service_Set_Arm_Servo_State(SOCKHANDLE ArmSocket, bool cmd, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②cmd:**

true-打开， false-关闭

**③block：**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.7 工具坐标系设置

### 5.7.1 标定点位 Service_Auto_Set_Tool_Frame

该函数用于六点法自动设置工具坐标系（标记点位），机械臂控制器最多只能存储 10 个工具信息，在建立新的工具之前，请确认工具数量没有超过限制，否则建立新工具无法成功。

int Service_Auto_Set_Tool_Frame(SOCKHANDLE ArmSocket, byte point_num, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②point_num**

1~6 代表 6 个标定点。

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

说明：控制器只能存储十个工具，超过十个控制器不予响应，请在标定前查询已有工具

### 5.7.2 生成工具坐标系 Service_Generate_Auto_Tool_Frame

该函数用于六点法自动设置工具坐标系(生成坐标系)，机械臂控制器最多只能存储 10 个工具信息，在建立新的工具之前，请确认工具数量没有超过限制，否则建立新工具无法成功。

int Service_Generate_Auto_Tool_Frame(SOCKHANDLE ArmSocket, char *name, float payload,float x,float y,float z, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②name**

工具坐标系名称，不能超过十个字节。

**③payload**

新工具执行末端负载重量 单位 g

**④x,y,z**

新工具执行末端负载的位置

**⑤block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

说明：控制器只能存储十个工具，超过十个控制器不予响应，请在标定前查询已

有工具

### 5.7.3 手动设置工具坐标系 Service_Manual_Set_Tool_Frame

该函数用于手动设置工具坐标系，机械臂控制器最多只能存储 10 个工具信息，在建立新的工具之前，请确认工具数量没有超过限制，否则建立新工具无法成功。

int Service_Manual_Set_Tool_Frame(SOCKHANDLE ArmSocket, char *name, POSE pose,float payload,float x,float y,float z, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②name**

工具坐标系名称，不能超过十个字节。

**③pose**

新工具执行末端相对于机械臂法兰中心的位姿

**④payload**

新工具执行末端负载重量 单位 g

**⑤x,y,z**

新工具执行末端负载的位置

**⑥block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

说明：控制器只能存储十个工具，超过十个控制器不予响应，请在标定前查询已有工具

### 5.7.4 切换当前工具坐标系 Service_Change_Tool_Frame

该函数用于切换当前工具坐标系

int Service_Change_Tool_Frame(SOCKHANDLE ArmSocket, char *name, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②name**

目标工具坐标系名称

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.7.5 删除指定工具坐标系 Service_Delete_Tool_Frame

该函数用于删除指定工具坐标系

int Service_Delete_Tool_Frame(SOCKHANDLE ArmSocket, char *name, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②name**

要删除的工具坐标系名称

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

备注：删除坐标系后，机械臂将切换到机械臂法兰末端工具坐标系。

### 5.7.6 设置末端负载质量和质心 Service_Set_Payload

该函数用于设置末端负载质量和质心

int Service_Set_Payload(SOCKHANDLE ArmSocket, int payload, float cx, float cy, float cz, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **payload**

末端负载质量，单位：g

③ **cx, cy, cz**

末端负载质心位置，单位：mm

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

#### 5.7.7 取消末端负载 Service_Set_None_Payload

该函数用于取消末端负载

int Service_Set_None_Payload(SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.8 工具坐标系查询

#### 5.8.1 获取当前工具坐标系 Service_Get_Current_Tool_Frame

该函数用于获取当前工具坐标系。

int Service_Get_Current_Tool_Frame(SOCKHANDLE ArmSocket, FRAME *tool);

**参数**

① **ArmSocket**

Socket 句柄

②**tool**

返回的坐标系

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.8.2 获取指定工具坐标系 Service_Get_Given_Tool_Frame

该函数用于获取指定工具坐标系

int Service_Get_Given_Tool_Frame(SOCKHANDLE ArmSocket, char *name, FRAME *tool);

**参数**

① **ArmSocket**

Socket 句柄

②**name**

指定的工具名称

③**tool**

返回的工具参数

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.8.3 获取所有工具坐标系名称 Service_Get_All_Tool_Frame

该函数用于获取所有工具坐标系名称

int Service_Get_All_Tool_Frame(SOCKHANDLE ArmSocket, FRAME_NAME *names);

**参数**

① **ArmSocket**

Socket 句柄

②**names**

返回的工具名称数组

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.9 工作坐标系设置

### 5.9.1 自动设置工作坐标系 Service_Auto_Set_Work_Frame

该函数用于三点法自动设置工作坐标系，机械臂控制器最多只能存储 10 个工作坐标系信息，在建立新的工作坐标系之前，请确认工作坐标系数量没有超过限制，否则建立新工作坐标系无法成功。

int Service_Auto_Set_Work_Frame(SOCKHANDLE ArmSocket, char *name, byte point_num, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**③name**

工作坐标系名称，不能超过十个字节。

**②point_num**

1~3 代表 3 个标定点，依次为原点、X 轴上任意点、Y 轴上任意点，4 代表生成成坐标系。

**④block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

**说明：**控制器只能存储十个工作坐标系，超过十个控制器不予响应，请在标定前查询已有工作坐标系

### 5.9.2 手动设置工作坐标系 Service_Manual_Set_Work_Frame

该函数用于手动设置工作坐标系。

int Service_Manual_Set_Work_Frame(SOCKHANDLE ArmSocket, char *name, POSE pose, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②name**

工作坐标系名称，不能超过十个字节。

**③pose**

新工作坐标系相对于基坐标系的位姿

**④block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

**说明**：控制器只能存储十个工作坐标系，超过十个控制器不予响应，请在标定前查询已有工作坐标系

### 5.9.3 切换当前工作坐标系 Service_Change_Work_Frame

该函数用于切换当前工作坐标系

int Service_Change_Work_Frame(SOCKHANDLE ArmSocket, char *name, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②name**

目标工作坐标系名称

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.9.4 删除指定工作坐标系 Service_Delete_Work_Frame

该函数用于删除指定工作坐标系。

int Service_Delete_Work_Frame(SOCKHANDLE ArmSocket, char *name, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②name**

要删除的工具坐标系名称

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

备注：删除坐标系后，机械臂将切换到机械臂基坐标系

## 5.10 工作坐标系查询

### 5.10.1 获取当前工作坐标系 Service_Get_Current_Work_Frame

该函数用于获取当前工作坐标系。

int Service_Get_Current_Work_Frame(SOCKHANDLE ArmSocket, FRAME *frame);

**参数**

**① ArmSocket**

Socket 句柄

**②frame**

返回的坐标系

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.10.2 获取指定工作坐标系 Service_Get_Given_Work_Frame

该函数用于获取指定工作坐标系

int Service_Get_Given_Work_Frame(SOCKHANDLE ArmSocket,char, char

*name, POSE *pose);

**参数**

① **ArmSocket**

Socket 句柄

②**name**

指定的工作坐标系名称

③**pose**

返回的工作坐标系位姿参数

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.10.3 获取所有工作坐标系名称 Service_Get_All_Work_Frame

该函数用于获取所有工作坐标系名称

int Service_Get_All_Work_Frame(SOCKHANDLE ArmSocket,char, FRAME_NAME *names);

**参数**

① **ArmSocket**

Socket 句柄

②**names**

返回的工作坐标系名称数组

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.11 机械臂状态查询

### 5.11.1 获取机械臂当前状态 Service_Get_Current_Arm_State

该函数用于获取机械臂当前状态

int Service_Get_Current_Arm_State(SOCKHANDLE ArmSocket,char, float*joint, POSE *pose, uint16_t *Arm_Err, uint16_t *Sys_Err);

**参数**

① **ArmSocket**

Socket 句柄

②**joint**

关节 1~6 角度数组

③**pose**

机械臂当前位姿

④**Arm_Err**

机械臂运行错误代码

⑤**Sys_Err**

控制器错误代码

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.11.2 获取关节温度 Service_Get_Joint_Temperature

该函数用于获取关节当前温度。

int Service_Get_Joint_Temperature(SOCKHANDLE ArmSocket,char, float *temperature);

**参数**

① **ArmSocket**

Socket 句柄

②**temperature**

关节 1~6 温度数组

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.11.3 获取关节电流 Service_Get_Joint_Current

该函数用于获取关节当前电流。

int Service_Get_Joint_Current(SOCKHANDLE ArmSocket,float *current);

**参数**

① **ArmSocket**

Socket 句柄

② **current**

关节 1~6 电流数组

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.11.4 获取关节电压 Service_Get_Joint_Voltage

该函数用于获取关节当前电压。

int Service_Get_Joint_Voltage(SOCKHANDLE ArmSocket, float *voltage);

**参数**

① **ArmSocket**

Socket 句柄

②**voltage**

关节 1~6 电压数组

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.11.5 获取关节当前角度 Service_Get_Joint_Degree

该函数用于获取机械臂各关节的当前角度

int Service_Get_Joint_Degree (SOCKHANDLE ArmSocket, float *joint)；

**参数**

① **ArmSocket**

Socket 句柄

②**joint**

关节 1~6 角度存放数组地址；

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.11.6 获取所有状态 Service_Get_Arm_All_State

该函数用于获取机械臂所有状态

int Service_Get_Arm_All_State (SOCKHANDLE ArmSocket, JOINT_STATE *joint_state);

**参数**

**① ArmSocket**

Socket 句柄

**②joint_state**

机械臂所有状态；

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.11.7 获取轨迹规划计数 Service_Get_Arm_Plan_Num

该函数用于获取机械臂轨迹规划计数

int Service_Get_Arm_Plan_Num (SOCKHANDLE ArmSocket, int* plan);

**参数**

**① ArmSocket**

Socket 句柄

**②plan**

轨迹规划计数

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.12 机械臂初始位姿

### 5.12.1 设置初始位置角度 Service_Set_Arm_Init_Pose

该函数用于设置机械臂的初始位置角度。

int Service_Set_Arm_Init_Pose(SOCKHANDLE ArmSocket, float *target, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②target**

机械臂初始位置关节角度数组

**③block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.12.2 获取初始位置角度 Service_Get_Arm_Init_Pose

该函数用于获取机械臂初始位置角度。

int Service_Get_Arm_Init_Pose(SOCKHANDLE ArmSocket, float *joint);

**参数**

**① ArmSocket**

Socket 句柄

**②joint**

机械臂初始位置关节角度数组

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.12.3 设置安装角度 Service_Set_Install_Pose

该函数用于设置机械臂安装方式。

int Service_Set_Install_Pose(SOCKHANDLE ArmSocket, float x,float y,bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②x**

旋转角

**③y**

俯仰角

**④block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.13 机械臂运动规划

### 5.13.1 关节空间运动 Service_Movej_Cmd

该函数用于关节空间运动。

int Service_Movej_Cmd(SOCKHANDLE ArmSocket, float *joint, byte v, float r, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②joint**

目标关节 1~6 角度数组

**③v**

速度比例 1~100，即规划速度和加速度占关节最大线转速和加速度的比例。

**④r**

轨迹交融半径，目前默认 0。

**⑤block**

0-非阻塞，发送后立即返回；1-阻塞，等待机械臂到达位置或者规划失败

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.13.2 笛卡尔空间直线运动 Service_Movel_Cmd

该函数用于笛卡尔空间直线运动。

int Service_Movel_Cmd(SOCKHANDLE ArmSocket, POSE pose, byte v, float r, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②pose**

目标位姿，位置单位：米，姿态单位：弧度

**③v**

速度比例 1~100，即规划速度和加速度占机械臂末端最大线速度和线加速度的百分比

**④r**

轨迹交融半径，目前默认 0。

**⑤block**

0-非阻塞，发送后立即返回；1-阻塞，等待机械臂到达位置或者规划失败

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.13.3 笛卡尔空间圆弧运动 Service_Movec_Cmd

该函数用于笛卡尔空间圆弧运动

int Service_Movec_Cmd(SOCKHANDLE ArmSocket, POSE pose_via, POSE pose_to, byte v, float r, byte loop, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②pose_vai**

中间点位姿，位置单位：米，姿态单位：弧度

**③pose_to**

终点位姿，位置单位：米，姿态单位：弧度

**④v**

速度比例 1~100，即规划速度和加速度占机械臂末端最大角速度和角加速度的百分比

**⑤r**

轨迹交融半径，目前默认 0。

**⑥loop**

规划圈数，目前默认 0.

**⑦block**

0-非阻塞，发送后立即返回；1-阻塞，等待机械臂到达位置或者规划失败

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.13.4 关节角度 CANFD 透传 Service_Movej_CANFD

该函数用于角度不经规划，直接通过 CANFD 透传给机械臂，使用透传接口是，请勿使用其他运动接口。

int Service_Movej_CANFD(SOCKHANDLE ArmSocket, float *joint);

**参数**

**① ArmSocket**

Socket 句柄

**② joint**

关节 1~6 目标角度数组

由于该模式直接下发给机械臂，不经控制器规划，因此只要控制器运行正常并且目标角度在可达范围内，机械臂立即返回成功指令，此时机械臂可能仍在运行；若有错误，立即返回失败指令。

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.13.5 位姿 CANFD 透传 Service_Movep_CANFD

该函数用于位姿不经规划，直接通过 CANFD 透传给机械臂，使用透传接口是，请勿使用其他运动接口。

int Service_Movep_CANFD(SOCKHANDLE ArmSocket, POSE pose);

**参数**

① **ArmSocket**

Socket 句柄

② **pose**

位姿

由于该模式直接下发给机械臂，不经控制器规划，因此只要控制器运行正常并且目标角度在可达范围内，机械臂立即返回成功指令，此时机械臂可能仍在运行；若有错误，立即返回失败指令。

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.13.6 快速急停 Service_Move_Stop_Cmd

该函数用于突发状况，机械臂以最快速度急停，轨迹不可恢复。

int Service_Move_Stop_Cmd(SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.13.7 暂停当前规划 Service_Move_Pause_Cmd

该函数用于轨迹暂停，暂停在规划轨迹上，轨迹可恢复。

int Service_Move_Pause_Cmd(SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.13.8 继续当前轨迹 Service_Move_Continue_Cmd

该函数用于轨迹暂停后，继续当前轨迹运动

int Service_Move_Continue_Cmd(SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.13.9 清除当前轨迹 Service_Clear_Current_Trajectory

该函数用于清除当前轨迹，必须在暂停后使用，否则机械臂会发生意外！！!!

int Service_Clear_Current_Trajectory(SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.13.10 清除所有轨迹 Service_Clear_All_Trajectory

该函数用于清除所有轨迹，必须在暂停后使用，否则机械臂会发生意外！！！！

int Service_Clear_All_Trajectory(SOCKHANDLE ArmSocket, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.13.11 获取当前规划轨迹 Service_Get_Current_Trajectory

该函数用于获取当前正在规划的轨迹信息

int Service_Get_Current_Trajectory(SOCKHANDLE ArmSocket, ARM_CTRL_MODES *type, float *data);

**参数**

**① ArmSocket**

Socket 句柄

**②type**

返回的规划类型

**③data**

无规划和关节空间规划为当前关节 1~6 角度数组；笛卡尔空间规划则为当前末端位姿。

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.13.12 关节空间运动 Service_Movej_P_Cmd

该函数用于关节空间运动到目标位姿

int Service_Movej_P_Cmd(SOCKHANDLE ArmSocket, POSE pose, byte v, float r, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**pose**

目标位姿，位置单位：米，姿态单位：弧度。

**注意**：该目标位姿必须是机械臂末端末端法兰中心基于基坐标系的位姿！！

③**v**

速度比例 1~100，即规划速度和加速度占机械臂末端最大线速度和线加速度的百分比

④**r**

轨迹交融半径，目前默认 0。

⑤**block**

0-非阻塞，发送后立即返回；1-阻塞，等待机械臂到达位置或者规划失败

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.14 机械臂示教

### 5.14.1 关节示教 Service_Joint_Teach_Cmd

该函数用于关节示教，关节从当前位置开始按照指定方向转动，接收到停止指令或者到达关节限位后停止。

int Service_Joint_Teach_Cmd(SOCKHANDLE ArmSocket, byte num, byte direction, byte v, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**num**

示教关节的序号，1~6

**③direction**

示教方向，0-负方向，1-正方向

**④v**

速度比例 1~100，即规划速度和加速度占关节最大线转速和加速度的百分比

**⑤block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.14.2 位置示教 Service_Pos_Teach_Cmd

该函数用于当前工作坐标系下，笛卡尔空间位置示教。机械臂在当前工作坐标系下，按照指定坐标轴方向开始直线运动，接收到停止指令或者该处无逆解时停止。

int Service_Pos_Teach_Cmd(SOCKHANDLE ArmSocket, POS_TEACH_MODES type, byte direction, byte v, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②type**

示教类型

**③direction**

示教方向，0-负方向，1-正方向

**④v**

速度比例 1~100，即规划速度和加速度占机械臂末端最大线速度和线加速度的百分比

**⑤block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.14.3 姿态示教 Service_Ort_Teach_Cmd

该函数用于当前工作坐标系下，笛卡尔空间末端姿态示教。机械臂在当前工作坐标系下，绕指定坐标轴旋转，接收到停止指令或者该处无逆解时停止。

int Service_Ort_Teach_Cmd(SOCKHANDLE ArmSocket, ORT_TEACH_MODES type, byte direction, byte v, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②type**

示教类型

**③direction**

示教方向，0-负方向，1-正方向

**④v**

速度比例 1~100，即规划速度和加速度占机械臂末端最大角速度和角加速度的百分比

**⑤block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.14.4 示教停止 Service_Teach_Stop_Cmd

该函数用于示教停止。

int Service_Teach_Stop_Cmd(SOCKHANDLE ArmSocket, bool block);

**参数**

**① ArmSocket**

Socket 句柄

②**block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.14.5 切换示教运动坐标系 Set_Teach_Frame

该函数用于切换示教运动坐标系。

int Set_Teach_Frame(SOCKHANDLE ArmSocket, int type, int block);

**参数**

① **ArmSocket**

Socket 句柄

②**type**

0: 基座标运动, 2: 工具坐标系运动

③**block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.15 机械臂步进

### 5.15.1 关节步进 Service_Joint_Step_Cmd

该函数用于关节步进。关节在当前位置下步进指定角度。

int Service_Joint_Step_Cmd(SOCKHANDLE ArmSocket, byte num, float step, byte v, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**num**

关节序号，1~6

③**step**

步进的角度

**④v**

速度比例 1~100，即规划速度和加速度占指定关节最大关节转速和关节加速度的百分比

**⑤block**

0-非阻塞，发送后立即返回；1-阻塞，等待机械臂返回失败或者到达位置指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.15.2 位置步进 Service_Pos_Step_Cmd

该函数用于当前工作坐标系下，位置步进。机械臂末端在当前工作坐标系下，朝指定坐标轴方向步进指定距离，到达位置返回成功指令，规划错误返回失败指令。

int Service_Pos_Step_Cmd(SOCKHANDLE ArmSocket, POS_TEACH_MODES type, float step, byte v, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②type**

示教类型

**③step**

步进的距离，单位 m

**④v**

速度比例 1~100，即规划速度和加速度占机械臂末端最大线速度和线加速度的百分比

**⑤block**

0-非阻塞，发送后立即返回；1-阻塞，等待机械臂返回失败或者到达位

置指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.15.3 姿态步进 Service_Ort_Step_Cmd

该函数用于当前工作坐标系下，姿态步进。机械臂末端在当前工作坐标系下，绕指定坐标轴方向步进指定弧度，到达位置返回成功指令，规划错误返回失败指令。

int Service_Ort_Step_Cmd(SOCKHANDLE ArmSocket, ORT_TEACH_MODES type, float step, byte v, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②type**

示教类型

**③step**

步进的弧度，单位 rad，精确到 0.001rad

**④v**

速度比例 1~100，即规划速度和加速度占机械臂末端最大角速度和角加速度的百分比

**⑤block**

0-非阻塞，发送后立即返回；1-阻塞，等待机械臂返回失败或者到达位置指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.16 控制器配置

### 5.16.1 获取控制器状态 Service_Get_Controller_State

该函数用于获取控制器状态。

int Service_Get_Controller_State(SOCKHANDLE ArmSocket, float *voltage,

float *current, float *temperature, uint16_t *sys_err);

**参数**

**① ArmSocket**

Socket 句柄

**②voltage**

返回的电压

**③current**

返回的电流

**④temperature**

返回的温度

**⑤sys_err**

控制器运行错误代码

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.2 设置 WiFi AP 模式设置 Service_Set_WiFi_AP_Data

该函数用于控制器 WiFi AP 模式设置，非阻塞模式，机械臂收到后更改参数，蜂鸣器响后代表更改成功，控制器重启，以 WIFI AP 模式通信。

int Service_Set_WiFi_AP_Data(SOCKHANDLE ArmSocket, char *wifi_name, char* password);

**参数**

**① ArmSocket**

Socket 句柄

**②wifi_name**

控制器 wifi 名称

**③password**

wifi 密码

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.3 设置 WiFi STA 模式设置 Service_Set_WiFI_STA_Data

该函数用于控制器 WiFi STA 模式设置，非阻塞模式，机械臂收到后更改参数，蜂鸣器响后代表更改成功，控制器重启，以 WIFI STA 模式通信。

int Service_Set_WiFI_STA_Data(SOCKHANDLE ArmSocket, char *router_name, char* password);

**参数**

**① ArmSocket**

Socket 句柄

**②router_name**

路由器名称

**③password**

路由器 Wifi 密码

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.4 设置 UART_USB 接口波特率 Service_Set_USB_Data

该函数用于控制器 UART_USB 接口波特率设置非阻塞模式，机械臂收到后更改参数，然后立即通过 UART-USB 接口与外界通信。

该指令下发后控制器会记录当前波特率，断电重启后仍会使用该波特率对外通信。

int Service_Set_USB_Data(SOCKHANDLE ArmSocket, int baudrate);

**参数**

**① ArmSocket**

Socket 句柄

**②baudrate**

波特率 波特率可选范围：9600,38400,115200 和 460800，若用户设置其他数据，控制器会默认按照 460800 处理。

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.5 设置 RS485 配置 Service_Set_RS485

该函数用于控制器设置 RS485 配置。

该指令下发后，若 Modbus 模式为打开状态，则会自动关闭，同时控制器会记录当前波特率，断电重启后仍会使用该波特率对外通信。

int Service_Set_RS485(SOCKHANDLE ArmSocket, int baudrate);

**参数**

**① ArmSocket**

Socket 句柄

**②baudrate**

波特率 波特率可选范围：9600,38400,115200 和 460800，若用户设置其他数据，控制器会默认按照 460800 处理。

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.6 切换以太网口通信 Service_Set_Ethernet

该函数用于控制器切换到以太网口通信模式，非阻塞模式，机械臂收到后立即通过以太网口接口与外界通信。

int Service_Set_Ethernet(SOCKHANDLE ArmSocket);

**① ArmSocket**

Socket 句柄

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.7 设置机械臂电源 Service_Set_Arm_Power

该函数用于设置机械臂电源

int Service_Set_Arm_Power (SOCKHANDLE ArmSocket, bool cmd, bool block);

**参数**

**① ArmSocket**

Socket 句柄

②**cmd**

true-上电，　　false-断电

③**block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.8 获取机械臂电源 Service_Get_Arm_Power_State

该函数用于获取机械臂电源

int Service_Get_Arm_Power_State (SOCKHANDLE ArmSocket, int* power);

**参数**

① **ArmSocket**

Socket 句柄

②**power**

0-上电，　　1-断电

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.9 读取机械臂软件版本 Service_Get_Arm_Software_Version

该函数用于读取机械臂软件版本

int Service_Get_Arm_Software_Version(SOCKHANDLE ArmSocket, char *plan_version,string* ctrl_version);

**参数**

① **ArmSocket**

Socket 句柄

② **plan_version**

读取到的用户接口内核版本号

③ **ctrl_version**

实时内核版本号

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.10 获取控制器的累计运行时间 Service_Get_System_Runtime

读取控制器的累计运行时间。

int Service_Get_System_Runtime (SOCKHANDLE ArmSocket, char *state,int *day, int *hour, int *min, int *sec);

**参数**

① **ArmSocket**

Socket 句柄

② **state**

错误提示，若无结果则系统正常

③ **day**

天

④ **hour**

小时

⑤ **min**

分

⑥ **sec**

秒

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.11 清空控制器累计运行时间 Service_Clear_System_Runtime

该函数用于清空控制器累计运行时间

int Service_Clear_System_Runtime(SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.12 获取关节累计转动角度 Service_Get_Joint_Odom

该函数用于读取关节的累计转动角度

int Service_Get_Joint_Odom(SOCKHANDLE ArmSocket, char* state,float* odom);

**参数**

① **ArmSocket**

Socket 句柄

② **state**

错误提示，若无结果则系统正常

③ **odom**

各关节累计的转动角度

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.13 清除关节累计转动角度 Service_Clear_Joint_Odom

该函数用于清空关节累计转动角度

int Service_Clear_Joint_Odom(SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.16.14 配置高速网口 Service_Set_High_Speed_Eth

配置高速网口

int Service_Set_High_Speed_Eth (SOCKHANDLE ArmSocket, byte num, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **num**

0-关闭；1-打开

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.17 IO 配置

机械臂控制器 IO 配置如下所示。

| | |
|---|---|
| 数字输出：DO | 4 路，可配置为 0~12V |
| 数字输入：DI | 3 路，可配置为 0~12V |
| 模拟输出：AO | 4 路，输出电压 0~10V |
| 模拟输入：AI | 4 路，输入电压 0~10V |

### 5.17.1 设置 IO 状态 Service_Set_IO_State

该函数用于配置指定 IO 输出状态。

int Service_Set_IO_State(SOCKHANDLE ArmSocket, int IO,byte num, bool state, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **IO**

指定设置数字 IO 为 0，指定模拟 IO 为 1

**③num**

指定 IO 输出通道，范围 1~4

**④state**

true-输出高电平，false-输出低电平

**⑤block**

0- 非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.17.2 查询指定 IO 状态 Service_Get_IO_State

该函数用于查询指定 IO 状态。

int Service_Get_IO_State(SOCKHANDLE ArmSocket, int IO,byte num, byte *state);

**参数**

**① ArmSocket**

Socket 句柄

**②IO**

指定 IO 类型，数字 IO 输出状态为 0，数字 IO 输入状态为 1， 模拟 IO 输出状态为 2，模拟 IO 输入状态为 3。

**③num**

指定数字 IO 输出通道，范围 1~4

**④state**

指定数字 IO 通道当前输出状态，0x01-高电平，0x00-低电平

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.17.3 查询所有 IO 输入状态 Service_Get_IO_Input

该函数用于查询所有 IO 输入状态。

int Service_Get_IO_Input(SOCKHANDLE ArmSocket, byte *DI_state, float

*AI_voltage);

**参数**

① **ArmSocket**

Socket 句柄

②**DI_state**

数字 IO 输入状态数组地址，0x01-输入高电平，0x00-输入低电平

③**AI_voltage**

模拟 IO 输入电压数组地址，范围：0~10v

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.17.4 查询所有 IO 输出状态 Service_Get_IO_Output

该函数用于查询所有 IO 输出状态。

int Service_Get_IO_Output(SOCKHANDLE ArmSocket, byte *DO_state, float *AO_voltage);

**参数**

① **ArmSocket**

Socket 句柄

②**DO_state**

数字 IO 输出状态数组地址，0x01-输入高电平，0x00-输入低电平

③**AO_voltage**

模拟 IO 输出电压数组地址，范围：0~10v

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.18 末端工具 IO 配置

机械臂末端工具端具有 IO 端口，数量和分类如下所示：

| | |
|---|---|
| 电源输出 | 1 路，可配置为 0V/5V/12V/24V |
| 数字输出：DO | 2 路，参考电平与电源输出一致 |
| 数字输入：DI | 2 路，参考电平与电源输出一致 |

| | |
|---|---|
| 模拟输出：AO | 1 路，输出电压 0~10V |
| 模拟输入：AI | 1 路，输入电压 0~10V |
| 通讯接口 | 1 路，可配置为 RS485/RS232/CAN |

### 5.18.1 设置工具端数字 IO 输出状态 Service_Set_Tool_DO_State

该函数用于配置工具端指定数字 IO 输出状态。

int Service_Set_Tool_DO_State(SOCKHANDLE ArmSocket, byte num, bool state, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②num**

指定数字 IO 输出通道，范围 1~2

**③state**

true-输出高电平，false-输出低电平

**④block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.18.2 设置工具端 IO 模式 Service_Set_Tool_IO_Mode

该函数用于设置 IO 模式。

int Service_Set_Tool_IO_Mode(SOCKHANDLE ArmSocket, byte num, bool state,bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②num**

指定数字通道，范围 1~2

**③state**

指定数字 IO 通道当前输出状态，0x01-输出，0x00-输入

**④block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.18.3 查询工具端数字 IO 状态 Service_Get_Tool_IO_State

该函数用于查询工具端数字 IO 状态。

int Service_Get_Tool_IO_State(SOCKHANDLE ArmSocket, float* IO_Mode, float *IO_state);

**参数**

**① ArmSocket**

Socket 句柄

**②IO_Mode**

指定数字 IO 通道模式(范围 1~2), 0-输入模式，1-输出模式

**③IO_state**

指定数字 IO 通道当前输入状态(范围 1~2)，0x01-高电平，0x00-低电平

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.18.4 设置工具端电源输出 Service_Set_Tool_Voltage

该函数用于设置工具端电源输出。

int Service_Set_Tool_Voltage(SOCKHANDLE ArmSocket, byte type, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②type**

电源输出类型：0-0V，1-5V，2-12V，3-24V

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

#### 5.18.5 获取工具端电源输出 Service_Get_Tool_Voltage

该函数用于获取工具端电源输出。

int Service_Get_Tool_Voltage(SOCKHANDLE ArmSocket, byte *voltage);

**参数**

① **ArmSocket**

Socket 句柄

②**voltage**

读取回来的电源输出类型：0-0V，1-5V，2-12V，3-24V

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.19 末端手爪控制（选配）

睿尔曼机械臂末端配备了因时机器人公司的 EG2-4B1 手爪，为了便于用户操作手爪，机械臂控制器对用户开放了手爪的 API 函数。

#### 5.19.1 配置手爪的开口度 Service_Set_Gripper_Route

该函数用于配置手爪的开口度。

int Service_Set_Gripper_Route(SOCKHANDLE ArmSocket, int min_limit, int max_limit, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**min**

手爪开口最小值，范围：0~1000，无单位量纲

③**max**

手爪开口最大值，范围：0~1000，无单位量纲

④ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.19.2 设置夹爪松开到最大位置 Service_Set_Gripper_Release

该函数用于控制手爪以指定速度张开到最大开口处

int Service_Set_Gripper_Release (SOCKHANDLE ArmSocket, int speed, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**speed**

手爪松开速度 ，范围 1~1000，无单位量纲

③**block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.19.3 设置夹爪夹取 Service_Set_Gripper_Pick

该函数用于控制手爪以设定的速度去夹取，当手爪所受力矩大于设定的力矩阈值时，停止运动。

int Service_Set_Gripper_Pick(SOCKHANDLE ArmSocket, int speed, int force, bool block);

**参数**

① **ArmSocket**

Socket 句柄

②**speed**

手爪夹取速度 ，范围：1~1000，无单位量纲

**③force**

手爪夹取力矩阈值，范围 ：50~1000，无单位量纲

**④ block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.19.4 设置夹爪持续夹取 Service_Set_Gripper_Pick_On

该函数用于控制手爪以设定的速度去持续夹取，当手爪所受力矩大于设定的力矩阈值时，停止运动。之后当手爪所受力矩小于设定力矩后，手爪继续持续夹取，直到再次手爪所受力矩大于设定的力矩阈值时，停止运动。

int Service_Set_Gripper_Pick_On(SOCKHANDLE ArmSocket, int speed, int force, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②speed**

手爪夹取速度 ，范围：1~1000，无单位量纲

**③force**

手爪夹取力矩阈值，范围 ：50~1000，无单位量纲

**④block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.19.5 设置夹爪到指定开口位置 Service_Set_Gripper_Position

该函数用于控制手爪到达指定开口度位置

int Service_Set_Gripper_Position (SOCKHANDLE ArmSocket, int position, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **position**

手爪指定开口度，范围 ：1~1000，无单位量纲

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.20 拖动示教及轨迹复现

睿尔曼机械臂采用关节电流环实现拖动示教，拖动示教及轨迹复现的配置函数如下所示。

### 5.20.1 进入拖动示教模式 Service_Start_Drag_Teach

该函数用于控制机械臂进入拖动示教模式

int Service_Start_Drag_Teach (SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.20.2 退出拖动示教模式 Service_Stop_Drag_Teach

该函数用于控制机械臂退出拖动示教模式

int Service_Stop_Drag_Teach (SOCKHANDLE ArmSocket,bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.20.3 拖动示教轨迹复现 Service_Run_Drag_Trajectory

该函数用于控制机械臂复现拖动示教的轨迹，必须在拖动示教结束后才能使用，同时保证机械臂位于拖动示教的起点位置。若当前位置没有位于轨迹复现起点，请先调用 5.20.7，否则会返回报错信息。

int Service_Run_Drag_Trajectory (SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.20.4 拖动示教轨迹复现暂停 Service_Pause_Drag_Trajectory

该函数用于控制机械臂在轨迹复现过程中的暂停。

int Service_Pause_Drag_Trajectory (SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.20.5 拖动示教轨迹复现继续 Service_Continue_Drag_Trajectory

该函数用于控制机械臂在轨迹复现过程中暂停之后的继续，轨迹继续时，必须保证机械臂位于暂停时的位置，否则会报错，用户只能从开始位置重新复现轨迹。

int Service_Continue_Drag_Trajectory (SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.20.6 拖动示教轨迹复现停止 Service_Stop_Drag_Trajectory

该函数用于控制机械臂在轨迹复现过程中停止，停止后，不可继续。若要再次轨迹复现，只能从第一个轨迹点开始。

int Service_Stop_Drag_Trajectory (SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.20.7 运动到轨迹起点 Service_Drag_Trajectory_Origin

轨迹复现前，必须控制机械臂运动到轨迹起点，如果设置正确，机械臂将以20%的速度运动到轨迹起点

int Service_Drag_Trajectory_Origin (SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.20.8 复合模式拖动示教 Service_Start_Multi_Drag_Teach

开始复合模式拖动示教

int Service_Start_Multi_Drag_Teach(SOCKHANDLE ArmSocket, int mode,bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **mode**

拖动示教模式

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.20.9 设置力位混合控制 Service_Set_Force_Postion

力位混合控制

int Service_Set_Force_Postion (SOCKHANDLE ArmSocket, int mode,int force,bool block);

**参数**

① **ArmSocket**

socket 句柄

② **sensor**

0-一维力；1-六维力

③ **mode**

0-基坐标系力控；1-工具坐标系力控

④ **direction**

力控方向；0-沿 X 轴；1-沿 Y 轴；2-沿 Z 轴；3-沿 RX 姿态方向；4-沿 RY 姿态方向；5-沿 RZ 姿态方向

⑤ **N**

力的大小，单位 0.1N

⑥ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.20.10 结束力位混合控制 Service_Stop_Force_Postion

结束力位混合控制

int Service_Stop_Force_Postion (SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.21 末端六维力传感器的使用（选配）

睿尔曼 RM-65F 机械臂末端配备集成式六维力传感器，无需外部走线，用户可直接通过协议对六维力进行操作，获取六维力数据。

### 5.21.1 获取六维力数据 Service_Get_Force_Data

查询当前六维力传感器得到的力和力矩信息，若要周期获取力数据，周期不能小于 50ms。

int Service_Get_Force_Data(SOCKHANDLE ArmSocket, float *Force);

**参数**

① **ArmSocket**

Socket 句柄

② **Force**

返回的力和力矩数组地址，数组 6 个元素，依次为 Fx，Fy，Fz, Mx, My, Mz。其中，力的单位为 N；力矩单位为 Nm。

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.21.2 清空六维力数据 Service_Clear_Force_Data

将六维力数据清零，即后续获得的所有数据都是基于当前数据的偏移量。

int Service_Clear_Force_Data(SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.21.3 设置六维力重心参数 Service_Set_Force_Sensor

设置六维力重心参数，六维力重新安装后，必须重新计算六维力所收到的初始力和重心。分别在不同姿态下，获取六维力的数据，用于计算重心位置。该指令下发后，机械臂以 20%的速度运动到各标定点，该过程不可中断，中断后必须重新标定。

**重要说明：**必须保证在机械臂静止状态下标定。

int Service_Set_Force_Sensor (SOCKHANDLE ArmSocket);

**参数**

① **ArmSocket**

Socket 句柄

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.21.4 手动标定六维力数据 Service_Manual_Set_Force

手动标定六维力数据；共四个点位，需要调用函数四次

int Service_Manual_Set_Force (SOCKHANDLE ArmSocket, int type,float *joint);

**参数**

① **ArmSocket**

Socket 句柄

② **type**

点位，依次调用四次发送 1~4；

③ **joint**

关节角度

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.21.5 退出标定流程 Service_Stop_Set_Force_Sensor

在标定六/一维力过程中，如果发生意外，发送该指令，停止机械臂运动，退出标定流程

int Service_Stop_Set_Force_Sensor (SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.22 末端五指灵巧手控制（选配）

睿尔曼 RM-65 机械臂末端配备了五指灵巧手，可通过协议对灵巧手进行设置。

### 5.22.1 设置灵巧手手势序号 Service_Set_Hand_Posture

设置灵巧手手势序号，设置成功后，灵巧手按照预先保存在 Flash 中的手势运动。

int Service_Set_Hand_Posture (SOCKHANDLE ArmSocket, int posture_num, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **posture_num**

预先保存在灵巧手内的手势序号，范围：1~40

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.22.2 设置灵巧手动作序列序号 Service_Set_Hand_Seq

设置灵巧手动作序列序号，设置成功后，灵巧手按照预先保存在 Flash 中的动作序列运动。

int Service_Set_Hand_Seq (SOCKHANDLE ArmSocket, int seq_num, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **seq_num**

预先保存在灵巧手内的动作序列序号，范围：1~40

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.22.3 设置灵巧手角度 Service_Set_Hand_Angle

设置灵巧手角度，灵巧手有 6 个自由度，从 1~6 分别为小拇指，无名指，中指，食指，大拇指弯曲，大拇指旋转。

int Service_Set_Hand_Angle (SOCKHANDLE ArmSocket, int *angle, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **angle**

手指角度数组，6 个元素分别代表 6 个自由度的角度。范围：0~1000.另外，-1 代表该自由度不执行任何操作，保持当前状态

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.22.4 设置灵巧手各关节速度 Service_Set_Hand_Speed

设置灵巧手各关节速度

int Service_Set_Hand_Speed (SOCKHANDLE ArmSocket, int speed, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **speed**

灵巧手各关节速度设置，范围：1~1000

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

#### 5.22.5 设置灵巧手各关节力阈值 Service_Set_Hand_Force

设置灵巧手各关节力阈值

int Service_Set_Hand_Force (SOCKHANDLE ArmSocket, int force, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **force**

灵巧手各关节力阈值设置，范围：1~1000，代表各关节的力矩阈值（四指握力 0~10N，拇指握力 0~15N）。

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.23 末端 PWM（选配）

睿尔曼机械臂末端接口板的数字输出 2 通道可复用为 PWM 输出，输出高电平与当前末端输出电压一致。

#### 5.23.1 设置末端 PWM 输出 Service_Set_PWM

该函数用于设置末端 PWM 输出。

int Service_Set_PWM (SOCKHANDLE ArmSocket, int Frq, int Dpulse, bool block);

**参数**

① **ArmSocket**

Socket 句柄

**②Frq**

PWM 输出频率，范围：100~10000Hz

**③Dpulse**

PWM 输出占空比，范围：0~100，精度不超过 1%

**④block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.23.2 停止末端 PWM 输出 Service_Stop_PWM

该函数用于停止末端 PWM 输出。

int Service_Stop_PWM (SOCKHANDLE ArmSocket, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.24 末端传感器-一维力（选配）

睿尔曼机械臂末端接口板集成了一维力传感器，可获取 Z 方向的力，量程 200N，准度 0.5%FS。


### 5.24.1 查询一维力数据 Service_Get_Fz

该函数用于查询末端一维力数据。

int Service_Get_Fz (SOCKHANDLE ArmSocket, float *data);

**参数**

**① ArmSocket**

Socket 句柄

**②data**

反馈的一维力数据

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

**备注**

第一帧指令下发后，开始更新一维力数据，此时返回的数据有滞后性；请从第二帧的数据开始使用。若周期查询 Fz 数据，频率不能高于 40Hz。

### 5.24.2 清零一维力数据 Service_Clear_Fz

该函数用于清零末端一维力数据。清空一维力数据后，后续所有获取到的数据都是基于当前的偏置。

int Service_Clear_Fz (SOCKHANDLE ArmSocket, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**②block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.24.3 自动标定末端一维力数据 Service_Auto_Set_Fz

该函数用于自动标定末端一维力数据。

int Service_Auto_Set_Fz (SOCKHANDLE ArmSocket);

**参数**

**① ArmSocket**

Socket 句柄

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.24.4 手动标定末端一维力数据 Service_Manual_Set_Fz

该函数用于手动标定末端一维力数据。

int Service_Manual_Set_Fz (SOCKHANDLE ArmSocket, float* joint,float* joint2);

**参数**

**① ArmSocket**

Socket 句柄

**②joint**

点位 1 关节角度

**③joint2**

点位 2 关节角度

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.25 Modbus RTU 配置

睿尔曼机械臂在控制器的 26 芯航插和末端接口板 9 芯航插处，各有 1 路

RS485 通讯接口，这两个 RS485 端口可通过 JSON 协议配置为标准的 Modbus RTU 模式。

### 5.25.1 设置通讯端口 Modbus RTU 模式 Service_Set_Modbus_Mode

配置通讯端口 Modbus RTU 模式。

Service_Set_Modbus_Mode (SOCKHANDLE ArmSocket, int port,int baudrate,int timeout,bool block);

**参数**

**① ArmSocket**

Socket 句柄

**② port**

通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口

**③ baudrate**

波特率，支持 9600,115200,460800 三种常见波特率

**④ timeout**

超时时间，单位百毫秒

**⑤ block**

0-非阻塞，发送后立即返回; 1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.25.2 关闭通讯端口 Modbus RTU 模式 Service_Close_Modbus_Mode

关闭通讯端口 Modbus RTU 模式。

Service_Close_Modbus_Mode (SOCKHANDLE ArmSocket,int port , bool block);

**参数**

**① ArmSocket**

Socket 句柄

② **port**

通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.25.3 读线圈 Service_Get_Read_Coils

读线圈。

Service_Get_Read_Coils (SOCKHANDLE ArmSocket,int, port, int address, int num, int device, int* coils_data);

**参数**

① **ArmSocket**

Socket 句柄

② **port**

通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口

③ **address**

线圈起始地址

④ **num**

要读的线圈的数量，该指令最多一次性支持读 8 个线圈数据，即返回的数据不会超过一个字节

⑤ **device**

外设设备地址

⑥ **coils_data**

返回线圈状态

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.25.4 读离散输入量 Service_Get_Read_Input_Status

读离散输入量。

Service_Get_Read_Input_Status (SOCKHANDLE ArmSocket,int, int port, int address, int num, int device, int* coils_data);

**参数**

① **ArmSocket**

Socket 句柄

② **port**

通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口

③ **address**

线圈起始地址

④ **num**

要读的线圈的数量，该指令最多一次性支持读 8 个线圈数据，即返回的数据不会超过一个字节

⑤ **device**

外设设备地址

⑥ **coils_data**

返回离散量

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.25.5 读保持寄存器 Service_Get_Read_Holding_Registers

读保持寄存器。

Service_Get_Read_Holding_Rsgisters (SOCKHANDLE ArmSocket,int, int port, int address, int device, int *coils_data);

**参数**

① **ArmSocket**

Socket 句柄

② **port**

通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口

③ **address**

线圈起始地址

④ **device**

外设设备地址

⑤ **coils_data**

返回离散量

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.25.6 读输入寄存器 Service_Get_Read_Input_Registers

读输入寄存器。

Service_Get_Read_Input_Registers (SOCKHANDLE ArmSocket,int, int port, int address, int device, int coils_data);

**参数**

① **ArmSocket**

Socket 句柄

② **port**

通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口

③ **address**

线圈起始地址

④ **device**

外设设备地址

⑤ **coils_data**

返回离散量

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.25.7 写单圈数据 Service_Write_Single_Coil

写单圈数据。

Service_Write_Single_Coil (SOCKHANDLE ArmSocket,int, int port, int address, int data, int device, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**② port**

通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口

**③ address**

线圈起始地址

**④ data**

要写入线圈的数据

**⑤ device**

外设设备地址

**⑥ block**

0-非阻塞，发送后立即返回；1-阻塞

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.25.8 写单个寄存器 Service_Write_Single_Register

写单个寄存器。

Service_Write_Single_Register (SOCKHANDLE ArmSocket,int, int port, int address, int data, int device, bool block);

**参数**

**① ArmSocket**

Socket 句柄

**② port**

通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口。

③ **address**

线圈起始地址。

④ **data**

要写入寄存器的数据。

⑤ **device**

外设设备地址。

⑥ **block**

0-非阻塞，发送后立即返回；1-阻塞。

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.25.9 写多个寄存器 Service_Write_Registers

写多个寄存器。

Service_Write_Registers(SOCKHANDLE ArmSocket,int, int port,int address,int num, byte *single_data, int device, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **port**

通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口

③ **address**

寄存器起始地址

④ **num**

写寄存器个数，寄存器每次写的数量不超过 10 个

⑤ **single_data**

要写入寄存器的数据数组，类型：byte*;

⑥ **device**

外设设备地址。

⑦ **block**

0-非阻塞，发送后立即返回；1-阻塞。

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.25.10 写多圈数据 Service_Write_Coils

写多圈数据

Service_Write_Coils(SOCKHANDLE ArmSocket,int, int port,int address,int num, byte *coils_data, int device, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **port**

通讯端口，0-控制器 RS485 端口，1-末端接口板 RS485 接口。

③ **address**

线圈起始地址。

④ **num**

写线圈个数，每次写的数量不超过 160 个。

⑤ **coils_data**

要写入线圈的数据数组，类型：byte。若线圈个数不大于 8，则写入的数据为 1 个字节；否则，则为多个数据的数组。

⑥ **device**

外设设备地址。

⑦ **block**

0-非阻塞，发送后立即返回；1-阻塞。

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.26 升降机构

睿尔曼机械臂可集成自主研发升降机构。

### 5.26.1 移动平台运动速度 Service_Set_Lift_Speed

设置移动平台运动速度。

Service_Set_Lift_Speed (SOCKHANDLE ArmSocket,int, int speed,bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **speed**

升降速度百分比

③ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.26.2 设置升降机构位置 Service_Set_Lift_Height

升降机构位置闭环控制。

Service_Set_Lift_Height (SOCKHANDLE ArmSocket,int, int height,int speed,bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **height**

目标高度

③ **speed**

升降速度百分比

④ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.26.3 获取升降机构状态 Service_Get_Lift_State

升降机构状态。

Service_Get_Lift_State (SOCKHANDLE ArmSocket,int, int *height,int *current,int* err);

**参数**

① **ArmSocket**

Socket 句柄

② **height**

目标高度 单位：mm 精度：1mm 范围：0~2300

③ **current**

当前升降驱动电流 单位：mA 精度：1mA

④ **err**

升降驱动错误代码

**返回值**

0- 成功返回：0。失败返回:错误码, define.h 查询。

## 5.27 透传力位混合控制补偿

针对睿尔曼带一维力和六维力版本的机械臂，用户除了可直接使用示教器调用底层的力位混合控制模块外，还可以将自定义的轨迹以周期性透传的形式结合底层的力位混合控制算法进行补偿。

### 5.27.1 开启透传力位混合控制补偿模式 Service_Start_Force_Position_Move

开启透传力位混合控制补偿模式。

Service_Start_Force_Position_Move (SOCKHANDLE ArmSocket,int, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.27.2 力位混合控制补偿透传模式（关节角）Service_Force_Position_Move

力位混合控制补偿数据。

Service_Force_Position_Move_Joint(SOCKHANDLE ArmSocket, const float *joint,byte sensor,byte mode,int dir,float force,bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **joint**

关节角度

③ **sensor**

所使用传感器类型，0-一维力，1-六维力

④ **mode**

模式，0-沿基坐标系，1-沿工具端坐标系

⑤ **dir**

力控方向，0~5 分别代表 X/Y/Z/Rx/Ry/Rz，其中一维力类型时默认方向为 Z 方向

⑥ **force**

力的大小 单位 N

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.27.3 力位混合控制补偿透传模式（位姿）Service_Force_Position_Move

力位混合控制补偿数据。

Service_Force_Position_Move_POSE(SOCKHANDLE ArmSocket, POSE pose,byte sensor,byte mode,int dir,float force,bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **pose**

当前坐标系下的位姿

③ **sensor**

所使用传感器类型，0-一维力，1-六维力

④ **mode**

模式，0-沿基坐标系，1-沿工具端坐标系

⑤ **dir**

力控方向，0~5 分别代表 X/Y/Z/Rx/Ry/Rz，其中一维力类型时默认方向为 Z 方向

⑥ **force**

力的大小 单位 N

⑦ **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.27.4 关闭透传力位混合控制补偿模式 Service_Stop_Force_Position_Move

关闭透传力位混合控制补偿模式。

Service_Stop_Force_Position_Move (SOCKHANDLE ArmSocket, bool block);

**参数**

① **ArmSocket**

Socket 句柄

② **block**

0-非阻塞，发送后立即返回；1-阻塞，等待控制器返回设置成功指令

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

## 5.28 算法工具接口

针对睿尔曼机械臂，提供正解、逆解等工具接口。

### 5.28.1 设置算法的安装角度 Service_SetAngle

设置算法的安装角度参数。

Service_setAngle(float x, float y, float z);

**参数**

① **x**

X 轴安装角度。

② **y**

Y 轴安装角度。

③ **z**

Z 轴安装角度。

### 5.28.2 设置算法接口的末端 DH 参数 Service_SetLwt

设置算法接口的末端 DH 参数。

Service_setLwt(int type);

**参数**

① **type**

0-标准版，1-一维力，2-六维力

### 5.28.3 正解 Service_Forward_Kinematics

用于睿尔曼机械臂正解计算。

Pose Service_Forward_Kinematics(const float * joint);

**参数**

① **joint**

关节 1 到关节 6 角度

**返回值**

正解结果

### 5.28.4 逆解 Service_Inverse_kinematics

用于睿尔曼机械臂逆解计算。

int Service_inverse_kinematics(const float *q_in, const Pose *q_pose, float *q_out, const uint8_t flag);

**参数**

① **q_in**

上一时刻关节角。

② **q_pose**

目标位姿。

③ **q_out**

输出的关节角度

④ **flag**

姿态参数类别：0-四元数；1-欧拉角

**返回值**

1：正常运行，-9 不可达；-3 关节 1 超限位 ~~ -8 关节 6 超限位；

### 5.28.5 计算环绕运动位姿 Service_RotateMove

用于计算环绕运动位姿。

Service_Pose RotateMove( float *Joint_Cur,int rotateAxis, float rotateAngle, Pose choose_axis);

**参数**

① **Joint_Cur**

当前关节角度

② **rotateAxis**

旋转轴: 1:x 轴, 2:y 轴, 3:z 轴

③ **rotateAngle**

旋转角度：旋转角度，单位(度)

④ **co_tool**

工具坐标系位姿，位置和欧拉角组成，可在示教器读出.不输入时,默认为都是零

⑤ **co_work**

工作坐标系位姿，位置和欧拉角组成，可在示教器读出.不输入时,默认为都是零

⑥ **choose_axis**

**返回值**

成功返回：0。失败返回:错误码, define.h 查询。

### 5.28.6 末端位姿转成工具位姿 Service_end2tool

末端位姿转成工具位姿 计算沿工具坐标系运动之后的终点。

Service_Pose end2tool(Pose eu_end, Pose co_tool, Pose co_work);

**参数**

① **eu_end**

末端位姿

② **co_tool**

工具坐标系位姿

③ **co_work**

工作坐标系位姿

**返回值**

工具位姿。

### 5.28.7 工具位姿转末端位姿 Service_tool2end

工具位姿转末端位姿

Service_Pose tool2end(Pose eu_tool, Pose co_tool, Pose co_work);

**参数**

① **eu_end**

末端位姿

② **co_tool**

工具坐标系位姿

③ **co_work**

工作坐标系位姿

**返回值**

末端位姿。

### 5.28.8 四元数转欧拉角 Service_quaternion2euler

四元数转欧拉角 。

Service_eul quaternion2euler(ort qua);

**参数**

① **qua**

四元数

**返回值**

欧拉角。

### 5.28.9 欧拉角转四元数 Service_euler2quaternion

欧拉角转四元数。

Service_ort euler2quaternion(eul eu);

**参数**

① **qua**

欧拉角

**返回值**

四元数。

### 5.28.10 欧拉角转旋转矩阵 Service_euler2matrix

欧拉角转旋转矩阵。

Service_Matrix euler2matrix(eul state);

**参数**

① **qua**

欧拉角

**返回值**

旋转矩阵。

### 5.28.11 位姿转旋转矩阵 Service_pos2matrix

位姿转旋转矩阵。

Service_Matrix pos2matrix(Pose state);

**参数**

① **qua**

位姿

**返回值**

旋转矩阵。

### 5.28.12 旋转矩阵转位姿 Service_matrix2pos

旋转矩阵转位姿。

Service_Pose matrix2pos(Matrix Mstate);

**参数**

① **qua**

旋转矩阵

**返回值**

位姿。

### 5.28.13 工作坐标系转基座标系 Service_WorkFrame_To_Base

基座标系转工作坐标系

Service_Pose WorkFrame_To_Base(Matrix matrix, Pose state);

**参数**

① **matrix**

工作坐标系在基座标系下的矩阵

② **state**

工具端坐标在工作座标系下位姿

**返回值**

工作坐标系下的位姿。

#### 5.28.14 计算沿工具坐标系运动位姿 Service_cartesian_tool

计算沿工具坐标系运动位姿。

Service_Pose Cartesian_Tool(float* Joint_Cur, int moveAxis, float movelength, int dev_mode);

**参数**

① **Joint_Cur**

当前关节角度

② **moveAxis**

移动轴 moveAxis=1 x 轴，moveAxis = 2 y 轴，moveAxis = 3 z 轴

③ **movelength**

移动长度，米为单位

④ **m_dev**

机械臂型号

**返回值**

工作坐标系下的位姿。

### 5.29 在线编程文件下发 •

#### 5.29.1 文件下发 Send_TrajectoryFile

在线编程文件下发。

**int Set_Teach_Frame(SOCKHANDLE ArmSocket, int type, int block);**

**参数**

① **ArmSocket**

socket 句柄

② **file_name**

轨迹文件完整路径 例: c:/rm_file.txt

③ **file_name_len**

file_name 字段的长度

④ **plan_speed**

规划速度比例

⑤ **block**

RM_NONBLOCK-非阻塞，发送后立即返回；RM_BLOCK-阻塞，等待控制器返回设置成功指