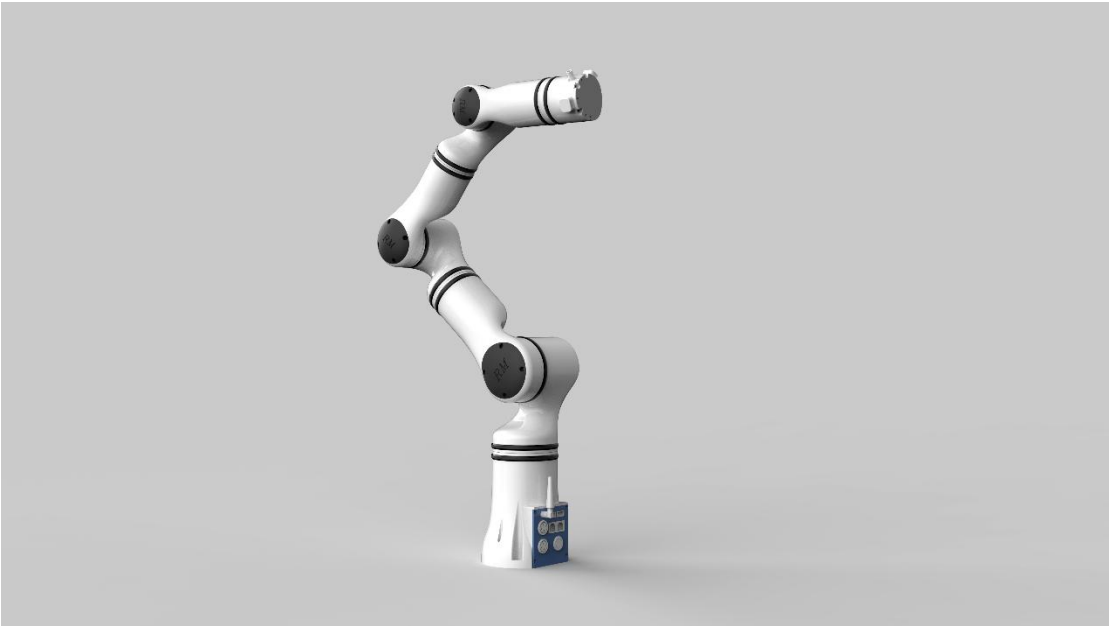
睿尔曼机械臂 SDK(C)学习资料

（内部学习）



睿尔曼智能科技（北京）有限公司

目录

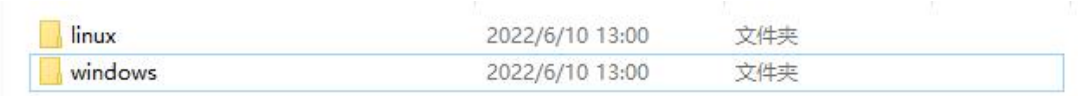
1.SDK 包简介

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。



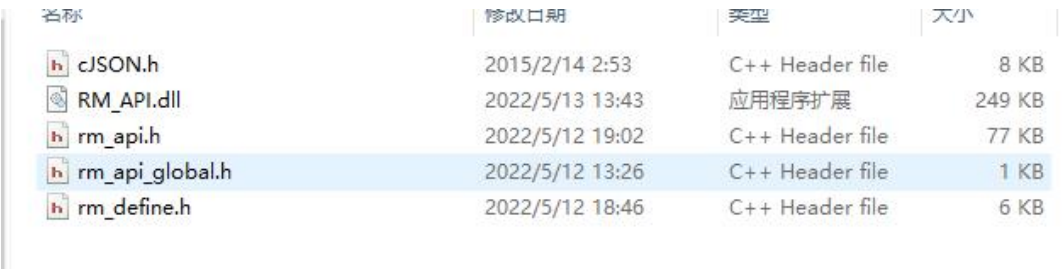
API 组成如下图所示:

（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。

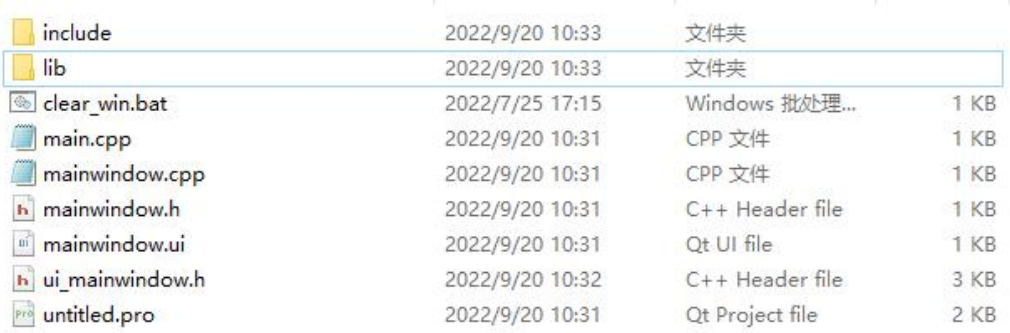


2.使用说明

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;
```

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 RM_API_Init()函数初始化 API。设置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。

3.接口函数示例

3.1 连接相关函数

1、API 初始化 RM_API_InitI

int RM_API_Init(int devMode, RM_Callback pCallback);	
函数功能	该函数用于 API 初始化
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。

示例	//初始化 API
	RM_API_Init(65, NULL);

2、API 反初始化 RM_API_UnInit

int RM_API_UnInit();	
函数功能	该函数用于 API 初始化

示例	//API 反初始化
	RM_API_UnInit();

3、查询 API 版本信息 RM_API

char * API_Version();	
函数功能	该函数用于查询 API 版本信息
返回值	API 版本号

示例	//返回 API 版本
	API_Version();

4、连接机械臂 Arm_Socket_Start

SOCKHANDLE Arm_Socket_Start(char* arm_ip,int arm_port, int recv_timeout);	
函数功能	该函数用于连接机械臂
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_port** 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:返回句柄，失败: < 0

示例	// 连接服务器 返回全局句柄
	m_sockhand = Arm_Socket_Start("192.168.1.18",8080,50000);

5、查询连接状态 Arm_Sockrt_State

int Arm_Sockrt_State(SOCKHANDLE ArmSocket,);	
函数功能	该函数用于查询机械臂连接状态
参数	① **ArmSocket** socket 句柄
返回值	正常返回：SYS_NORMAL 异常:错误码, define.h 查询

示例	// 网络连接状态
	ret = Arm_Sockrt_State(m_sockhand);

6、关闭连接 Arm_Socket_Close

void Arm_Socket_Close(SOCKHANDLE ArmSocket,);	
函数功能	该函数用于关闭与机械臂的 socket 连接。
参数	① **ArmSocket** socket 句柄
示例	Arm_Socket_Close(m_sockhand); //关闭 socket 连接

3.2 关节配置函数

对机械臂的关节参数进行设置，如果关节发生错误，则无法修改关节参数，必须先清除关节错误代码。另外设置关节另外之前，必须先将关节掉使能，否则会设置不成功。

注意：关节所有参数在修改完成后，会自动保存到关节 Flash，立即生效，之后关节处于掉使能状态，修改完参数后必须发送指令控制关节上使能。

注意：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。

7、设置关节最大速度 Set_Joint_Speed

int Set_Joint_Speed(SOCKHANDLE ArmSocket, byte joint_num, float speed, bool block);	
函数功能	该函数用于设置关节最大速度，单位：RPM。
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。

示例	//设置关节 2，最大转速 30RPM
	byte joint_num = 2;
	float speed = 30;
	ret = Set_Joint_Speed(m_sockhand , joint_num , speed, 1);

8、设置关节最大加速度 Set_Joint_Acc

int Set_Joint_Acc(SOCKHANDLE ArmSocket, byte joint_num, float acc, bool block);	
函数功能	该函数用于设置关节最大加速度，单位：RPM/s。
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。

示例	//设置关节 2 最大加速度 500RPM/s
	byte joint_num = 2;
	float acc = 500;
	ret = Set_Joint_Acc(m_sockhand ,joint_num, acc,1);

9、设置关节最小限位 Set_Joint_Min_Pos

int Set_Joint_Min_Pos(SOCKHANDLE ArmSocket, byte joint_num, float joint, bool block);	
函数功能	该函数用于设置关节所能到达的最小位置，单位：度。
参数	① **ArmSocket** socket 句柄 ② **joint_num** 关节序号，1~6 ③ **joint** 关节最小位置，单位：° 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。

示例	//设置关节 6，最小限位度数-360°
	byte joint_num = 6;
	float joint = -360;
	ret = Set_Joint_Min_Pos(m_sockhand,joint_num,joint,1);

10、设置关节最大位置 Set_Joint_Max_Pos

int Set_Joint_Max_Pos(SOCKHANDLE ArmSocket, byte joint_num, float joint, bool block);	
函数功能	该函数用于设置关节所能到达的最大位置，单位：度。
参数	① **ArmSocket** socket 句柄 ② **joint_num** 关节序号，1~6 ③ **joint** 关节最大位置，单位：° 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。

示例	//设置关节 6，最大限位度数 360°
	byte joint_num = 6;
	float joint = 360;
	ret = Set_Joint_Max_Pos(m_sockhand,joint_num,joint,1);

11、设置关节使能 Set_Joint_EN_State

int Set_Joint_EN_State(SOCKHANDLE ArmSocket, byte joint_num, bool state, bool block);	
函数功能	该函数用于设置关节使能状态。
参数	① **ArmSocket** socket 句柄 ② **joint_num** 关节序号，1~6 ③ **state**

	true-上使能，false-掉使能 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。

示例	//设置关节 1 上使能
	byte joint_num = 1;
	ret = Set_Joint_EN_State(m_sockhand,joint_num ,true,1);

12、设置关节零位 Set_Joint_Zero_Pos

int Set_Joint_Zero_Pos(SOCKHANDLE ArmSocket, byte joint_num, bool block);	
函数功能	该函数用于将当前位置设置为关节零位。
参数	① **ArmSocket** socket 句柄 ② **joint_num** 关节序号，1~6 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。

示例	//设置关节 3 位置为零位
	byte joint_num = 3;
	ret = Set_Joint_Zero_Pos(m_sockhand,joint_num,1);

13、清除关节错误代码 Set_Joint_Err_Clear

int Set_Joint_Err_Clear(SOCKHANDLE ArmSocket, byte joint_num, bool block);	
函数功能	该函数用于清除关节错误代码。
参数	① **ArmSocket** socket 句柄

	② **joint_num** 关节序号，1~6 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。

示例	//清除关节 2 错误代码
	byte joint_num = 2;
	ret = Set_Joint_Err_Clear(m_sockhand,joint_num,1);

14、开始关节标定 Start_Calibrate

int Start_Calibrate(SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于开始关节标定。
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。

示例	//开始关节标定
	ret = Start_Calibrate(m_sockhand,1);

15、结束关节标定 Stop_Calibrate

int Stop_Calibrate(SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于结束关节标定。
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。

示例	//结束关节标定
	ret = Stop_Calibrate(m_sockhand,1);

16、查询关节标定状态 Get_Calibrate_State

int Get_Calibrate_State(SOCKHANDLE ArmSocket, float *joint_state, uint16_t *state,int* time);	
函数功能	该函数用于查询关节标定状态。
参数	① **ArmSocket** socket 句柄 ② **joint_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。 ③ **err** 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。
示例	//查询关节标定状态 float joint_state[6] = {0}; uint16_t state[6] = {0};

	int *time = 0; ret = Get_Calibrate_State(m_sockhand,joint_state,state,time);

3.3 关节参数查询函数

17、获取关节最大速度 Get_Joint_Speed

int Get_Joint_Speed(SOCKHANDLE ArmSocket, float *speed);	
函数功能	该函数用于查询关节最大速度。
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。

示例	//查询关节最大速度
	float speed[6] = {0,1,2,3,4,5};
	ret = Get_Joint_Speed(m_sockhand,speed);

18、获取关节最大加速度 Get_Joint_Acc

int Get_Joint_Acc(SOCKHANDLE ArmSocket, float *acc);	
函数功能	该函数用于获取关节最大加速度。
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。

示例	//查询关节最大加速度
	float acc[6] = {0,1,2,3,4,5};
	ret = Get_Joint_Acc(m_sockhand,acc);

19、获取关节最小位置 Get_Joint_Min_Pos

int Get_Joint_Min_Pos(SOCKHANDLE ArmSocket, float *min_joint);	
函数功能	该函数用于获取关节最小位置。
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。

示例	//查询关节最小位置
	float min_joint[6] = {0,1,2,3,4,5};
	ret = api -> Get_Joint_Min_Pos(min_joint);

20、获取关节最大位置 Get_Joint_Max_Pos

int Get_Joint_Max_Pos(SOCKHANDLE ArmSocket, float *max_joint);	
函数功能	该函数用于获取关节最大位置。
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。

示例	//查询关节最大位置
	float max_joint[6] = {0,1,2,3,4,5};
	ret = Get_Joint_Max_Pos(max_joint);

21、获取关节使能状态 Get_Joint_EN_State

int Get_Joint_EN_State(SOCKHANDLE ArmSocket, byte *state);	
函数功能	该函数用于获取关节使能状态。
参数	① **ArmSocket** socket 句柄 ② **state**

	存放关节 1~6 使能状态数组地址，1-上使能状态，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。

示例	//查询关节使能状态
	byte sta[6] = {0,1,2,3,4,5};
	ret = Get_Joint_EN_State(m_sockhand,sta);

22、获取关节错误代码 Get_Joint_Err_Flag

int Get_Joint_EN_State(SOCKHANDLE ArmSocket, byte *state);	
函数功能	该函数用于获取关节使能状态。
参数	① **ArmSocket** socket 句柄 ② **state** 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。

示例	//获取关节错误代码
	uint16_t sta[6] = {0,1,2,3,4,5};
	ret = api -> Get_Joint_Err_Flag(sta);

23、查询关节软件版本号 Get_Joint_Software_Version

int Get_Joint_Software_Version(SOCKHANDLE ArmSocket, float* version);	
函数功能	该函数用于查询关节软件版本号。
参数	① **ArmSocket** socket 句柄 ② **version** 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。

示例	//获取关节软件版本号
	float ver[6] = {0};
	ret = Get_Joint_Software_Version(m_sockhand,ver);

3.4 机械臂末端运动参数配置

24、设置末端最大线速度 Set_Arm_Line_Speed

int Set_Arm_Line_Speed(SOCKHANDLE ArmSocket, float speed, bool block);	
函数功能	该函数用于设置机械臂末端最大线速度。
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。

示例	//设置机械臂末端最大线速度 0.1m/s
	float speed = 0.1;
	ret = Set_Arm_Line_Speed(m_sockhand,speed,1);

25、设置末端最大线加速度 Set_Arm_Line_Acc

int Set_Arm_Line_Acc(SOCKHANDLE ArmSocket, float acc, bool block);	
函数功能	该函数用于设置机械臂末端最大线加速度。
参数	④ **ArmSocket** socket 句柄 ⑤ **acc** 末端最大线加速度，单位 m/s^2 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。
示例	//设置机械臂末端最大线加速度 2m/s² float acc = 2;

	ret = Set_Arm_Line_Acc(m_sockhand,acc, 1);

26、设置末端最大角速度 Set_Arm_Angular_Speed

int Set_Arm_Angular_Speed(SOCKHANDLE ArmSocket, float speed, bool block);	
函数功能	该函数用于设置机械臂末端最大角速度。
参数	① **ArmSocket** socket 句柄 ② **speed** 机械臂末端最大角速度，单位 rad/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。

示例	//设置机械臂末端最大角速度 0.2rad/s
	float speed = 0.2;
	ret = Set_Arm_Angular_Speed(m_sockhand,speed, 1);

27、设置末端最大角加速度 Set_Arm_Angular_Acc

int Set_Arm_Angular_Acc(SOCKHANDLE ArmSocket, float acc, bool block);	
函数功能	该函数用于设置机械臂末端最大角加速度。
参数	① **ArmSocket** socket 句柄 ② **acc** 末端最大角加速度，单位 rad/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。
示例	//设置机械臂末端最大角加速度 4rad/s²

	float acc = 4; ret = Set_Arm_Angular_Acc(m_sockhand,acc, 1);

28、获取末端线速度 Get_Arm_Line_Speedd

int Get_Arm_Line_Speed(SOCKHANDLE ArmSocket, float *speed);	
函数功能	该函数用于获取机械臂末端线速度。
参数	① **ArmSocket** socket 句柄 ② **speed** 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。

示例	//获取机械臂末端线速度
	float speed = 0;
	ret = Get_Arm_Line_Speed(m_sockhand,&speed);

29、获取末端线加速度 Get_Arm_Line_Acc

int Get_Arm_Line_Acc(SOCKHANDLE ArmSocket, float *acc);	
函数功能	该函数用于获取机械臂末端线加速度。
参数	① **ArmSocket** socket 句柄 ② **acc** 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。

示例	//获取机械臂末端线加速度
	float acc = 0;
	ret = Get_Arm_Line_Acc(m_sockhand,&acc);

30、获取末端角速度 Get_Arm_Angular_Speed

int Get_Arm_Angular_Speed(SOCKHANDLE ArmSocket, float *speed);

函数功能	该函数用于获取机械臂末端角速度。
参数	① **ArmSocket** socket 句柄 ② **speed** 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。

示例	//获取机械臂末端角速度
	float speed = 0;
	ret = Get_Arm_Angular_Speed(m_sockhand,&speed);

31、获取末端角加速度 Get_Arm_Angular_Acc

int Get_Arm_Angular_Acc(SOCKHANDLE ArmSocket, float *acc);	
函数功能	该函数用于获取机械臂末端角加速度。
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。

示例	//获取末端角加速度
	float acc = 0;
	ret = Get_Arm_Angular_Acc(m_sockhand,&acc);

32、设置末端参数为初始值 Set_Arm_Tip_Init

int Set_Arm_Tip_Init(SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于设置机械臂末端参数为初始值。
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。

示例	//初始化机械臂参数，机械臂的末端参数回复到默认值。其中
	末端线速度：0.1m/s 末端线加速度：0.5m/s²
	末端角速度：0.2rad/s 末端角加速度：1rad/s²
	ret = Set_Arm_Tip_Init(m_sockhand,1);

33、设置碰撞等级 Set_Collision_Stage

int Set_Collision_Stage (SOCKHANDLE ArmSocket, int stage, bool block);	
函数功能	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。
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。

示例	//设置机械臂碰撞防护等级为 1
	int stage = 1;
	ret = Set_Collision_Stage (m_sockhand,stage,1);

34、查询碰撞等级 Get_Collision_Stage

int Get_Collision_Stage (SOCKHANDLE ArmSocket, int* stage);	
函数功能	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。
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。

示例	//查询机械臂动力学碰撞等级
	int stage = -1;
	ret = Get_Collision_Stage (m_sockhand,&stage);

35、设置 DH 参数 Set_DH_Data

int Set_DH_Data (SOCKHANDLE ArmSocket, float lsb, float lse, float lew, float lwt, float d3, bool block);	
函数功能	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。
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。

示例	//设置机械臂 DH 参数
	float lsb = 240.5;
	float lse = 256;
	float lew = 210;
	float lwt = 143.9;
	float d3 = 0;
	ret = Set_DH_Data(m_sockhand,lsb,lse,lew,lwt,d3,1);

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！

36、获取 DH 参数 Get_DH_Data

int Get_DH_Data (SOCKHANDLE ArmSocket, float* lsb, float* lse, float* lew, float* lwt, float* d3);	
函数功能	该函数用于获取机械臂 DH 参数。
参数	① **ArmSocket** socket 句柄 ② **lsb, 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。

示例	//获取机械臂 DH 参数
	float lsb = 0;
	float lse = 0;
	float lew = 0;
	float lwt = 0;
	float d3 = 0;
	ret = Get_DH_Data(m_sockhand,&lsb,&lse,&lew,&lwt,&d3);

37、设置关节零位补偿角度 Set_Joint_Zero_Offset

int Set_Joint_Zero_Offset (SOCKHANDLE ArmSocket, float *offset, bool block);	
函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **offset** 关节 1~6 零位补偿角度数组，角度单位：度 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。

示例	//设置关节零位偏移
	//关节 1~6 的零位补偿角度：1°，-2°，3°，-4°，5°，-6°
	float offset[7] = {1,-2,3,-4,5,-6} ;
	ret = Set_Joint_Zero_Offset (m_sockhand,offset, 1);

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！

38、设置动力学参数 Set_Arm_Dynamic_Parm

int Set_Arm_Dynamic_Parm (SOCKHANDLE ArmSocket, const float* parm, bool block);	
函数功能	该函数用于设置机械臂动力学参数。
参数	① **ArmSocket** socket 句柄 ② **offset** 关节 1~6 动力学参数 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。

示例	//重新设置机械臂动力学参数。
	float parm[7] = {0,1,2,3,4,5,6} ;
	ret = Set_Arm_Dynamic_Parm(m_sockhand,parm, 1);

3.5 机械臂末端接口板

39、查询末端接口板软件版本号 Get_Tool_Software_Version

int Get_Tool_Software_Version(SOCKHANDLE ArmSocket, int *version);	
函数功能	该函数用于查询末端接口板软件版本号
参数	① **ArmSocket**

	socket 句柄 ② **version** 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。

示例	//查询末端接口板软件版本号
	int ver = 0;
	ret = Get_Tool_Software_Version(m_sockhand,&ver);

3.6 机械臂状态伺服配置

40、设置伺服状态查询 Set_Arm_Servo_State

int Set_Arm_Servo_State(SOCKHANDLE ArmSocket, bool cmd, bool block);	
函数功能	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。控制器上电默认周期查询机械臂状态，为减小 CANFD 总线负载，需要关闭伺服查询时，操作该指令。
参数	① **ArmSocket** socket 句柄 ② **cmd:** true-打开，　false-关闭 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。

示例	//打开控制器对机械臂伺服状态查询
	bool cmd = true;
	ret = Set_Arm_Servo_State(m_sockhand,cmd, 1);

41、系统状态自动回传 Set_System_State_Servo

int Set_System_State_Servo(SOCKHANDLE ArmSocket, bool cmd, bool block);	
函数功能	该函数用于打开或者关闭系统状态自动回传，自动回传打开后，以

	50Hz 的频率周期回传系统状态。
参数	④ **ArmSocket** socket 句柄 ⑤ **cmd:** true-打开， false-关闭 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。

示例	//打开控制器对机械臂伺服状态查询
	bool cmd = true;
	ret = Set_Arm_Servo_State(m_sockhand,cmd, 1);

3.7 工具坐标系设置

42、标定点位 Auto_Set_Tool_Frame

int Auto_Set_Tool_Frame(SOCKHANDLE ArmSocket, byte point_num, bool block);	
函数功能	该函数用于六点法自动设置工具坐标系（标记点位）。
参数	① **ArmSocket** socket 句柄 ② **point_num** 1~6 代表 6 个标定点。 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。

示例	//自动计算工具坐标系标记点位，标定当前位置为参考点 6
	byte point_num = 6;
	ret = Auto_Set_Tool_Frame(m_sockhand,point_num,1);

备注	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。

43、生成工具坐标系 Generate_Auto_Tool_Frame

int Generate_Auto_Tool_Frame(SOCKHANDLE ArmSocket, const char *name,float payload,float x,float y,float z, bool block);	
函数功能	该函数用于六点法自动设置工具坐标系(生成坐标系)。
参数	① **ArmSocket** socket 句柄 ② **name** 工具坐标系名称，不能超过十个字节。 ③ **payload** 新工具执行末端负载重量 单位 g ④ **x,y,z** 新工具执行末端负载的位置 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。

示例	//自动计算工具坐标系生成坐标系，名称 t1，末端负载 5000g，质心位置：x-1mm,y-2mm,z-3mm
	const char *name = "t1";
	float payload = 5000;
	float x = 1;
	float y = 2;
	float z = 3;
	ret = Generate_Auto_Tool_Frame(m_sockhand,name,payload,x,y,z,1);

备注	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。

44、手动设置工具坐标系 Manual_Set_Tool_Frame

int Manual_Set_Tool_Frame(SOCKHANDLE ArmSocket, char *name, POSE pose,float payload,float x,float y,float z, bool block);	
函数功能	该函数用于手动设置工具坐标系。
参数	① **ArmSocket** socket 句柄 ② **name** 工具坐标系名称，不能超过十个字节。 ③ **pose** 新工具执行末端相对于机械臂法兰中心的位姿 ④ **payload** 新工具执行末端负载重量 单位 g ⑤ **x,y,z** 新工具执行末端负载的位置 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。

示例	//手动输入工具坐标系，名称 t2
	//工具位置：x：331.78mm，y:49.12mm，z：98.98mm
	//工具姿态：rx：3.11249rad，ry：0.0rad，rz：-3.00656rad
	//末端负载 5000g，质心位置：x-1mm,y-2mm,z-3mm
	const char *name1 = "t2";
	POSE pose;
	pose.px=331.78;
	pose.py=49.12;
	pose.pz=98.98;

	pose.rx=3.11249;
	pose.ry=0.0;
	pose.rz=-3.00656;
	float payload = 5000;
	float x = 1;
	float y = 2;
	float z = 3;
	ret = Manual_Set_Tool_Frame(m_sockhand,name1,pose,payload,x,y,z,1);

备注	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。

45、切换当前工具坐标系 Change_Tool_Frame

int Change_Tool_Frame(SOCKHANDLE ArmSocket, char *name, bool block);	
函数功能	该函数用于切换当前工具坐标系
参数	① **ArmSocket** socket 句柄 ② **name** 目标工具坐标系名称 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。

示例	//切换为 t1 工具坐标系
	const char *name = "t1";
	ret = Change_Tool_Frame(m_sockhand,name,1);

46、删除指定工具坐标系 Delete_Tool_Frame

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

函数功能	该函数用于删除指定工具坐标系
参数	① **ArmSocket** socket 句柄 ② **name** 要删除的工具坐标系名称 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。

示例	//删除 t1 工具坐标系
	const char *name = "t1";
	ret = Delete_Tool_Frame(m_sockhand,name,1);

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。

47、设置末端负载质量和质心 Set_Payload

int Set_Payload(SOCKHANDLE ArmSocket, int payload, float cx, float cy, float cz, bool block);	
函数功能	该函数用于设置末端负载质量和质心
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 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。
示例	//设置末端负载质量和质心

	float payload = 5000;
	float cx = 1;
	float cy = 2;
	float cz = 3;
	ret = Set_Payload(m_sockhand,payload, cx, cy, cz, 1);

48、取消末端负载 Set_None_Payload

int Set_None_Payload(SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于取消末端负载
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。

示例	//取消末端负载
	ret = Set_None_Payload(m_sockhand, 1);

3.8 工具坐标系查询

49、获取当前工具坐标系 Get_Current_Tool_Frame

int Get_Current_Tool_Frame(SOCKHANDLE ArmSocket, FRAME *tool);	
函数功能	该函数用于获取当前工具坐标系
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。
示例	//获取当前工具坐标系

	FRAME tool; ret = Get_Current_Tool_Frame(m_sockhand,&tool);

50、获取指定工具坐标系 Get_Given_Tool_Frame

int Get_Given_Tool_Frame(SOCKHANDLE ArmSocket, char *name, FRAME *tool);	
函数功能	该函数用于获取指定工具坐标系
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。

示例	//获取 t2 工具坐标系参数
	const char *name = "t2";
	FRAME tool;
	ret = Get_Given_Tool_Frame(m_sockhand,name,&tool);

51、获取所有工具坐标系名称 Get_All_Tool_Frame

int Get_All_Tool_Frame(SOCKHANDLE ArmSocket, FRAME_NAME *names);	
函数功能	该函数用于获取所有工具坐标系名称
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。
示例	//获取所有工具坐标系名称

	FRAME_NAME names[10]={0}; ret = Get_All_Tool_Frame(m_sockhand,names);

3.9 工作坐标系设置

52、自动设置工作坐标系 Auto_Set_Work_Frame

int Auto_Set_Work_Frame(SOCKHANDLE ArmSocket, char *name, byte point_num, bool block);	
函数功能	该函数用于三点法自动设置工作坐标系。
参数	① **ArmSocket** socket 句柄 ② **name** 工作坐标系名称，不能超过十个字节。 ③ **point_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。 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。

示例	//自动设置工作坐标系，名称 w1，标记当前位置为点 3
	const char *name3 = "w1";
	byte point_num=3;

备注	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。

53、手动设置工作坐标系 Manual_Set_Work_Frame

int Manual_Set_Work_Frame(SOCKHANDLE ArmSocket, char *name, POSE pose,

bool block);	
函数功能	该函数用于手动设置工作坐标系。
参数	① **ArmSocket** socket 句柄 ② **name** 工作坐标系名称，不能超过十个字节。 ③ **pose** 新工作坐标系相对于基坐标系的位姿 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。

示例	// 手动输入工作坐标系，名称 w2
	坐标系位置：x：0.1m，y:0.2m，z：0.03m
	坐标系姿态：rx：0.4rad，ry：0.5rad，rz：0.6rad
	const char *name = "w2";
	POSE pose;
	pose.px = 100;
	pose.py = 200;
	pose.pz = 30;
	pose.rx = 0.4;
	pose.ry = 0.5;
	pose.rz = 0.6;
	ret = Manual_Set_Work_Frame(m_sockhand,name,pose,1);

备注	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。

54、切换当前工作坐标系 Change_Work_Frame

int Change_Work_Frame(SOCKHANDLE ArmSocket, char *name, bool block);	
函数功能	该函数用于切换当前工作坐标系
参数	① **ArmSocket** socket 句柄

	② **name** 目标工作坐标系名称 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。

示例	//切换为 w1 工作坐标系
	const char *name = "w1";
	ret = Change_Work_Frame(m_sockhand,name,1);

55、删除指定工作坐标系 Delete_Work_Frame

int Delete_Work_Frame(SOCKHANDLE ArmSocket, const char *name, bool block);	
函数功能	该函数用于删除指定工作坐标系。
参数	① **ArmSocket** socket 句柄 ② **name** 要删除的工具坐标系名称 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。

示例	//删除 w1 工作坐标系
	const char *name = "w1";
	ret = Delete_Work_Frame(m_sockhand,name,1);

备注	删除坐标系后，机械臂将切换到机械臂基坐标系

3.10 工作坐标系查询

56、获取当前工作坐标系 Get_Current_Work_Frame

int Get_Current_Work_Frame(SOCKHANDLE ArmSocket, FRAME *frame);	
函数功能	该函数用于获取当前工作坐标系。
参数	① **ArmSocket** socket 句柄 ② **frame** 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。

示例	//获取当前工作坐标系
	FRAME work;
	ret = Get_Current_Work_Frame(m_sockhand,&work);

57、获取指定工作坐标系 Get_Given_Work_Frame

int Get_Given_Work_Frame(SOCKHANDLE ArmSocket, char *name, POSE *pose);	
函数功能	该函数用于获取指定工作坐标系
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。

示例	//获取 w2 工作坐标系
	const char *name = "w2";
	POSE pose = {0};
	ret = Get_Given_Work_Frame(m_sockhand,name,&pose);

58、获取所有工作坐标系名称 Get_All_Work_Frame

int Get_All_Work_Frame(SOCKHANDLE ArmSocket, FRAME_NAME *names);	
函数功能	该函数用于获取所有工作坐标系名称。
参数	① **ArmSocket** socket 句柄 ② **names** 返回的工作名称数组 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。

示例	//获取所有工作坐标系名称
	FRAME_NAME names[10]={0};
	ret = Get_All_Work_Frame(m_sockhand,names);

3.11 机械臂状态查询

59、获取机械臂当前状态 Get_Current_Arm_State

int Get_Current_Arm_State(SOCKHANDLE ArmSocket, float*joint, POSE *pose, uint16_t *Arm_Err, uint16_t *Sys_Err);	
函数功能	该函数用于获取机械臂当前状态
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。
示例	//获取机械臂当前状态 float joint[6] ; POSE po1; uint16_t Arm_Err; uint16_t Sys_Err; ret = Get_Current_Arm_State(m_sockhand,joint,&po1,&Arm_Err,&Sys_Err);

60、获取关节温度 Get_Joint_Temperature

int Get_Joint_Temperature(SOCKHANDLE ArmSocket, float *temperature);	
函数功能	该函数用于获取关节当前温度。
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。

示例	//获取关节当前温度
	float temperature[6] = {0,1,2,3,4,5};
	ret = Get_Joint_Temperature(m_sockhand,temperature);

61、获取关节电流 Get_Joint_Current

int Get_Joint_Current(SOCKHANDLE ArmSocket, float *current);	
函数功能	该函数用于获取关节当前电流。
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。
示例	//获取关节当前电流

	float current[6] = {0,1,2,3,4,5}; ret = Get_Joint_Current(m_sockhand,current);

62、获取关节电压 Get_Joint_Voltage

int Get_Joint_Voltage(SOCKHANDLE ArmSocket, float *voltage);	
函数功能	该函数用于获取关节当前电压。
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。

示例	//获取关节当前电压
	float voltage[6] = {0,1,2,3,4,5};
	ret = Get_Joint_Voltage(m_sockhand,voltage);

63、获取关节当前角度 Get_Joint_Degree

int Get_Joint_Degree (SOCKHANDLE ArmSocket, float *joint)；	
函数功能	该函数用于获取机械臂各关节的当前角度
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。

示例	//获取机械臂各关节的当前角度
	float joint[6] = {0,1,2,3,4,5};
	ret = Get_Joint_Degree (m_sockhand,joint);

64、获取所有状态 Get_Arm_All_State

int Get_Arm_All_State (SOCKHANDLE ArmSocket, JOINT_STATE *joint_state)；	
函数功能	该函数用于获取机械臂所有状态
参数	① **ArmSocket**

	socket 句柄 ② **joint_state** 存储机械臂信息的结构体；
返回值	成功返回：0。失败返回:错误码, define.h 查询。

示例	//获取机械臂所有状态
	JOINT_STATE joint_state = {0};
	ret = Get_Arm_All_State (m_sockhand,&joint_state);

65、获取轨迹规划计数 Get_Arm_Plan_Nume

int Get_Arm_Plan_Num (SOCKHANDLE ArmSocket, int* plan);	
函数功能	该函数用于获取机械臂轨迹规划计数
参数	① **ArmSocket** socket 句柄 ② **plan** 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。

示例	//获取机械臂轨迹规划计数
	int plan;
	ret = Get_Arm_Plan_Num (m_sockhand,&plan);

3.12 机械臂初始位姿

66、设置初始位置角度 Set_Arm_Init_Pose

int Set_Arm_Init_Pose(SOCKHANDLE ArmSocket, float *target, bool block);	
函数功能	该函数用于设置机械臂的初始位置角度。
参数	① **ArmSocket** socket 句柄 ② **target** 机械臂初始位置关节角度数组

	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。

示例	//设置初始位置关节角度 0°,0°,0°,0°,0°,0°,0°
	const float target[7] = {0,0,0,0,0,0,0};
	ret = Set_Arm_Init_Pose(m_sockhand,target, 1);

67、获取初始位置角度 Get_Arm_Init_Pose

int Get_Arm_Init_Pose(SOCKHANDLE ArmSocket, float *joint);	
函数功能	该函数用于获取机械臂初始位置角度。
参数	① **ArmSocket** socket 句柄 ② **joint** 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。

示例	//获取初始位置角度
	float joint[7] = {0,1,2,3,4,5,6};
	ret = Get_Arm_Init_Pose(m_sockhand,joint);

68、设置安装角度 Set_Install_Pose

int Set_Install_Pose(SOCKHANDLE ArmSocket, float x,float y,bool block);	
函数功能	该函数用于设置机械臂安装方式。
参数	① **ArmSocket** socket 句柄 ② **x** 旋转角 ③ **y** 俯仰角

	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。

示例	//设置安装角度旋转角 0°，俯仰角 0°
	float x = 0;
	float y = 0;
	ret = Set_Install_Pose(m_sockhand,x,y,1);

3.13 机械臂运动规划

69、关节空间运动 Movej_Cmd

int Movej_Cmd(SOCKHANDLE ArmSocket, float *joint, byte v, float r, bool block);	
函数功能	该函数用于关节空间运动。
参数	① **ArmSocket** socket 句柄 ② **joint** 目标关节 1~6 角度数组 ③ **v** 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。 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。
示例	//关节运动，关节角度[0°,0°,0°,0°,90°,0°,0°],速度系数 20%，交融半径：0

	float joint[7] = {0,0,0,0,90,0,0}; ret = Movej_Cmd(m_sockhand,joint,20,0,1);

70、笛卡尔空间直线运动 Movel_Cmd

int Movel_Cmd(SOCKHANDLE ArmSocket, POSE pose, byte v, float r, bool block);	
函数功能	该函数用于笛卡尔空间直线运动。
参数	① **ArmSocket** socket 句柄 ② **pose** 目标位姿，位置单位：米，姿态单位：弧度 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。

示例	//直线运动，目标位置：x：-0.3m，y:-0.03m，z：0.215m；目标姿态：rx:3rad，ry:0.1rad，rz:0.1rad；速度系数 20%，不交融
	POSE pose;
	pose.px=-0.300;
	pose.py=-0.030;
	pose.pz=0.215;
	pose.rx=3.0;
	pose.ry=0.1;
	pose.rz=0.1;
	ret = Movel_Cmd(m_sockhand,pose, 20,0,1);

71、笛卡尔空间圆弧运动 Movec_Cmd

int Movec_Cmd(SOCKHANDLE ArmSocket, POSE pose_via, POSE pose_to, byte v,

float r, byte loop, bool block);	
函数功能	该函数用于笛卡尔空间圆弧运动
参数	① **ArmSocket** socket 句柄 ② **pose_vai** 中间点位姿，位置单位：米，姿态单位：弧度 ③ **pose_to** 终点位姿，位置单位：米，姿态单位：弧度 ④ **v** 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 ⑤ **r** 轨迹交融半径，目前默认 0。 ⑥ **loop** 规划圈数，目前默认 0. ⑦ **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。

示例	/*圆弧运动:中间点位置：x：-0.3m，y:-0.03m，z：0.215m;中间点姿态：rx:3rad，ry:0.1rad，rz:0.1rad;终点位置：x：0.4m，y:-0.03m，z：0.215m;终点姿态：rx:3rad，ry:0.1rad，rz:0.1rad;速度系数 20%，;不交融;不循环*/
	POSE povia;
	povia.px=-0.300;
	povia.py=-0.03;
	povia.pz=0.215;
	povia.rx=3.0;
	povia.ry=0.1;
	povia.rz=0.1;
	POSE poto;
	poto.px=-0.4;
	poto.py=-0.030;
	poto.pz=0.215;
	poto.rx=3.0;

	poto.ry=0.1;
	poto.rz=0.1;
	ret = Movec_Cmd(m_sockhand,povia,poto,20,0,0,1);

72、关节角度 CANFD 透传 Movej_CANFD

int Movej_CANFD(SOCKHANDLE ArmSocket, float *joint, bool block);	
函数功能	该函数用于角度不经规划，直接通过 CANFD 透传给机械臂，使用透传接口时，请勿使用其他运动接口。
参数	① **ArmSocket** socket 句柄 ② **joint** 关节 1~6 目标角度数组
返回值	成功返回：0。失败返回:错误码, define.h 查询。
示例	//角度透传到 CANFD，目标关节角度：[1°,0°,20°,30°,0°,20°] float fl1[6] = { 1, 0, 20, 30, 0, 20}; Movej_CANFD(m_sockhand,fl1);
备注	用户使用该指令时请做好轨迹规划，轨迹规划的平滑程度决定了机械臂的运行状态，帧与帧之间关节的角度差不能超过 10°，并保证关节规划的速度不要超过 180°/s，否则关节会不响应。

73、位姿 CANFD 透传 Movep_CANFD

int Movep_CANFD(SOCKHANDLE ArmSocket, POSE pose, bool block);	
函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **pose** 位姿
返回值	成功返回：0。失败返回:错误码, define.h 查询。
示例	/*pose：目标位姿，位置精度：0.001mm，姿态精度：0.001rad 目标位置：x：0m，y:0m，z：0.85049m

	目标姿态：rx:0rad，ry:0rad，rz:3.142rad
	目标位姿为当前工具在当前工作坐标系下的数值。*/
	POSE pose;
	pose.px=0;
	pose.py=0;
	pose.pz=0.85049;
	pose.rx=0;
	pose.ry=0;
	pose.rz=3.142;
	Movep_CANFD(m_sockhand,pose);

74、快速急停 Move_Stop_Cmd

int Move_Stop_Cmd(SOCKHANDLE ArmSocket, bool block);	
函数功能	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。
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。

示例	//轨迹急停
	ret = Move_Stop_Cmd(m_sockhand,1);

75、暂停当前规划 Move_Pause_Cmd

int Move_Pause_Cmd(SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于轨迹暂停，暂停在规划轨迹上，轨迹可恢复。
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。

示例	//轨迹暂停
	ret = Move_Pause_Cmd(m_sockhand,1);

76、继续当前轨迹 Move_Continue_Cmd

int Move_Continue_Cmd(SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于轨迹暂停后，继续当前轨迹运动
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。

示例	//轨迹暂停后恢复
	ret = Move_Continue_Cmd(m_sockhand,1);

77、清除当前轨迹 Clear_Current_Trajectory

int Clear_Current_Trajectory(SOCKHANDLE ArmSocket, bool block);	
函数功能	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！！！！
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。

示例	//清除当前轨迹
	ret = Clear_Current_Trajectory(m_sockhand,1);

78、清除所有轨迹 Clear_All_Trajectory

int Clear_All_Trajectory(SOCKHANDLE ArmSocket, bool block);	
函数功能	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！
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。

示例	//清除所有轨迹
	ret = Clear_All_Trajectory(m_sockhand,1);

79、获取当前规划轨迹 Get_Current_Trajectory

int Get_Current_Trajectory(SOCKHANDLE ArmSocket, ARM_CTRL_MODES *type, float *data);	
函数功能	该函数用于获取当前正在规划的轨迹信息
参数	① **ArmSocket** socket 句柄 ② **type** 返回的规划类型 ③ **data** 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。
示例	//获取当前规划轨迹 ARM_CTRL_MODES type; float data[7] = {0,0,0,0,0,0,0};

	ret = Get_Current_Trajectory(m_sockhand,&type,data);

80、关节空间运动 Movej_P_Cmd

int Movej_P_Cmd(SOCKHANDLE ArmSocket, POSE pose, byte v, float r, bool block);	
函数功能	该函数用于关节空间运动到目标位姿
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。

示例	//关节空间运动，目标位置：x：0.1m，y:0.2m，z：0.03m；目标姿态：rx:0.4rad，ry:0.5rad，rz:0.6rad；速度系数 20%，不交融
	POSE pose;
	pose.px=-0.3;
	pose.py=-0.03;
	pose.pz=0.215;
	pose.rx=3.0;
	pose.ry=0.1;
	pose.rz=0.1;

	ret = Movej_P_Cmd(m_sockhand,pose, 20,0,1);

3.14 机械臂示教

81、关节示教 Joint_Teach_Cmd

int Joint_Teach_Cmd(SOCKHANDLE ArmSocket, byte num, byte direction, byte v, bool block);	
函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **num** 示教关节的序号，1~6 ③ **direction** 示教方向，0-负方向，1-正方向 ④ **v** 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 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。

示例	//关节 6 示教，正方向，速度 20%
	byte num = 6;
	byte direction = 1;
	byte v = 20;
	ret = Joint_Teach_Cmd(m_sockhand,num ,direction ,v,1);

82、位置示教 Pos_Teach_Cmd

int Pos_Teach_Cmd(SOCKHANDLE ArmSocket, POS_TEACH_MODES type, byte direction, byte v, bool block);	
函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **type** 示教类型 ③ **direction** 示教方向，0-负方向，1-正方向 ④ **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 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。

示例	//位置示教，x 轴负方向，速度 20%
	POS_TEACH_MODES type = X_Dir;
	byte direction = 1;
	byte v = 20;
	ret = Pos_Teach_Cmd(m_sockhand,type, direction, v, 1);

83、姿态示教 Ort_Teach_Cmd

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

函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **type** 示教类型 ③ **direction** 示教方向，0-负方向，1-正方向 ④ **v** 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 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。

示例	//姿态示教，rx 轴负方向，速度 20%
	ORT_TEACH_MODES type = RX_Rotate;
	byte direction = 1;
	byte v = 20;
	ret = Ort_Teach_Cmd(m_sockhand,type, direction, v, 1);

84、示教停止 Teach_Stop_Cmd

int Teach_Stop_Cmd(SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于示教停止。
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。

示例	//示教停止
	ret = Teach_Stop_Cmd(m_sockhand,1);

3.15 机械臂步进

85、关节步进 Joint_Step_Cmd

int Joint_Step_Cmd(SOCKHANDLE ArmSocket, byte num, float step, byte v, bool block);	
函数功能	该函数用于关节步进。关节在当前位置下步进指定角度。
参数	① **ArmSocket** socket 句柄 ② **num** 关节序号，1~6 ③ **step** 步进的角度 ④ **v** 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 ⑤ **block** 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。

示例	//关节步进，关节 1 反方向步进 10 度，速度系数 30%
	byte num = 1;
	float step = -10;
	byte v = 30;
	ret = Joint_Step_Cmd(m_sockhand,num, step, v,1);

86、位置步进 Pos_Step_Cmd

int Pos_Step_Cmd(SOCKHANDLE ArmSocket, POS_TEACH_MODES type, float step, byte v, bool block);	
函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **type** 示教类型 ③ **step** 步进的距离，单位 m ④ **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 ⑤ **block** 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。

示例	//位置步进，y 轴负方向步进 0.1m，速度 30%
	POS_TEACH_MODES type = Y_Dir;
	float step = -0.1;
	byte v = 30;
	ret = Pos_Step_Cmd(m_sockhand,type, step, v,1);

87、姿态步进 Ort_Step_Cmd

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

函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **type** 示教类型 ③ **step** 步进的弧度，单位 rad，精确到 0.001rad ④ **v** 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 ⑤ **block** 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。

示例	//姿态步进，y 轴负方向旋转 0.5rad，速度 30%
	ORT_TEACH_MODES type = RY_Rotate;
	float step = -0.5;
	byte v = 30;
	ret = Ort_Step_Cmd(m_sockhand, type, step, v, 1);

3.16 控制器配置

88、获取控制器状态 Get_Controller_State

int Get_Controller_State(SOCKHANDLE ArmSocket, float *voltage, float *current, float *temperature, uint16_t *sys_err);	
函数功能	该函数用于获取控制器状态。
参数	① **ArmSocket**

	socket 句柄 ② **voltage** 返回的电压 ③ **current** 返回的电流 ④ **temperature** 返回的温度 ⑤ **sys_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。

示例	//获取控制器状态
	float voltage = 0;
	float current = 0;
	float temperature = 0;
	uint16_t sys_err = 0;
	ret = Get_Controller_State(m_sockhand, &voltage, &current, &temperature, &sys_err);

89、设置 WiFi AP 模式设置 Set_WiFi_AP_Data

int Set_WiFi_AP_Data(SOCKHANDLE ArmSocket, char *wifi_name, char* password);	
函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **wifi_name** 控制器 wifi 名称 ③ **password**

	wifi 密码
返回值	成功返回：0。失败返回:错误码, define.h 查询。

示例	//配置 wifiAP 内容，wifi 名称：robot，连接密码：12345678
	char *wifi_name = (char*)"robot";
	char* password = (char*)"12345678";
	ret = Set_WiFi_AP_Data(m_sockhand,wifi_name,password);

90、设置 WiFi STA 模式设置 Set_WiFI_STA_Data

int Set_WiFI_STA_Data(SOCKHANDLE ArmSocket, char *router_name, char* password);	
函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **router_name** 路由器名称 ③ **password** 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。

示例	//配置 wifiSTA 内容，目标路由器名称：robot，路由器密码：12345678
	char *router_name = (char*)"robot";
	char* password = (char*)"12345678";
	ret = Set_WiFI_STA_Data(m_sockhand,router_name,password);

91、设置 UART_USB 接口波特率 Set_USB_Data

int Set_USB_Data(SOCKHANDLE ArmSocket, int baudrate);	
函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **baudrate** 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。

示例	//配置 USB 波特率为 460800
	int baudrate = 460800;
	ret = Set_USB_Data(m_sockhand,baudrate);

92、设置 RS485 配置 Set_RS485

int Set_RS485(SOCKHANDLE ArmSocket, int baudrate);	
函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **baudrate** 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。

示例	//配置 RS485 波特率为 460800
	int baudrate = 460800;
	ret = Set_RS485(m_sockhand,baudrate);

93、切换以太网口通信 Set_Ethernet

int Set_Ethernet(SOCKHANDLE ArmSocket,);

函数功能	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。
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。

示例	//切换到以太网口通信模式
	ret = Set_Ethernet(m_sockhand);

94、设置机械臂电源 Set_Arm_Power

int Set_Arm_Power (SOCKHANDLE ArmSocket, bool cmd, bool block);	
函数功能	该函数用于设置机械臂电源。
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。

示例	//控制机械臂上电
	ret = Set_Arm_Power (m_sockhand,true, 1);

95、获取机械臂电源 Get_Arm_Power_State

int Get_Arm_Power_State (SOCKHANDLE ArmSocket, int* power);	
函数功能	该函数用于获取机械臂电源
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。

示例	//读取机械臂电源状态
	int power;
	ret = Get_Arm_Power_State (m_sockhand,&power);

96、读取机械臂软件版本 Get_Arm_Software_Version

int Get_Arm_Software_Version(SOCKHANDLE ArmSocket, char* plan_version, char* ctrl_version, char *kernal1, char *kernal2);	
函数功能	该函数用于读取机械臂软件版本
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。
示例	//读取机械臂软件版本 char plan_version[8]; char ctrl_version[8]; ret =Get_Arm_Software_Version(m_sockhand,plan_version,ctrl_version);

97、获取控制器的累计运行时间 Get_System_Runtime

Get_System_Runtime (SOCKHANDLE ArmSocket, string *state,int *day, int *hour, int *min, int *sec);	
函数功能	读取控制器的累计运行时间。
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。

示例	//位置示教，x 轴负方向，速度 20%
	char state = 0;
	int day;
	int hour;
	int min;
	int sec;
	ret = Get_System_Runtime (m_sockhand,&state, &day, &hour, &min, &sec);

98、清空控制器累计运行时间 Clear_System_Runtime

int Clear_System_Runtime(SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于清空控制器累计运行时间。
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。

示例	//清空控制器累计运行时间
	ret = Clear_System_Runtime(m_sockhand,1);

99、获取关节累计转动角度 Get_Joint_Odom

int Get_Joint_Odom(SOCKHANDLE ArmSocket, char* state,float* odom);	
函数功能	该函数用于读取关节的累计转动角度。
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。

示例	//获取关节累计转动角度
	char state;
	float odom[7];
	ret = Get_Joint_Odom(m_sockhand,&state,odom);

100、清除关节累计转动角度 Clear_Joint_Odom

int Clear_Joint_Odom(SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于清空关节累计转动角度
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。

示例	//清除关节累计转动角度
	ret = Clear_Joint_Odom(m_sockhand,1);

101、配置高速网口 Set_High_Speed_Eth

int Set_High_Speed_Eth (SOCKHANDLE ArmSocket, byte num, bool block);	
函数功能	该函数用于配置高速网口。
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。

示例	//关闭高速网口
	byte num = 0;
	ret = Set_High_Speed_Eth (m_sockhand,num, 1);

3.17 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

102、设置 IO 状态 Set_IO_State

int Set_IO_State(SOCKHANDLE ArmSocket, int IO,byte num, bool state, bool block);	
函数功能	该函数用于配置指定 IO 输出状态。
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 ③ **num** 指定 IO 输出通道，范围 1~4 ④ **state** true-输出高电平，false-输出低电平 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。

示例	//设置数字 IO 1 号通道输出高电平
	int IO = 0;
	byte num = 1;
	bool state = true;
	ret = Set_IO_State (m_sockhand,IO,num,state,1);

103、查询指定 IO 状态 Get_IO_State

int Get_IO_State(SOCKHANDLE ArmSocket, int IO,byte num, byte *state);	
函数功能	该函数用于查询指定 IO 状态。
参数	① **ArmSocket** socket 句柄 ② **IO** 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。 ③ **num** 指定数字 IO 输出通道，范围 1~4 ④ **state** 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。

示例	//查询数字 IO 输出 1 号通道状态
	int IO = 0;
	byte num = 1;
	byte state;
	ret = Get_IO_State(m_sockhand,IO,num,&state);

104、查询所有 IO 输入状态 Get_IO_Input

int Get_IO_Input(SOCKHANDLE ArmSocket, byte *DI_state, float *AI_voltage);	
函数功能	该函数用于查询所有 IO 输入状态。
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。

示例	//查询所有 IO 输入状态
	byte DI_state[3] = {0};
	float AI_voltage[4] = {0};
	ret = Get_IO_Input(m_sockhand,DI_state,AI_voltage);

105、查询所有 IO 输出状态 Get_IO_Output

int Get_IO_Output(SOCKHANDLE ArmSocket, byte *DO_state, float *AO_voltage);	
函数功能	该函数用于查询所有 IO 输出状态。
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。

示例	//查询所有输出状态
	byte DO_state[4] = {0};
	float AO_voltage[4] = {0};
	ret = Get_IO_Output(m_sockhand,DO_state,AO_voltage);

3.18 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

106、设置工具端数字 IO 输出状态 Set_Tool_DO_State

int Set_Tool_DO_State(SOCKHANDLE ArmSocket, byte num, bool state, bool block);	
函数功能	该函数用于配置工具端指定数字 IO 输出状态。
参数	① **ArmSocket** socket 句柄 ② **num** 指定数字 IO 输出通道，范围 1~2 ③ **state** true-输出高电平，false-输出低电平

	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。

示例	//设置工具端 1 号通道输出高电平
	byte num = 1;
	bool state = true;
	ret = Set_Tool_DO_State(m_sockhand,num,state,1);

107、设置工具端 IO 模式 Set_Tool_IO_Mode

int Set_Tool_IO_Mode(SOCKHANDLE ArmSocket, byte num, bool state,bool block);	
函数功能	该函数用于设置 IO 模式。
参数	① **ArmSocket** socket 句柄 ② **num** 指定数字通道，范围 1~2 ③ **state** 指定数字 IO 通道当前输出状态，0x01-输出，0x00-输入 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。

示例	//设置工具端 IO2 号数字通道为输出模式
	byte num = 2;
	bool state = 1;
	ret = Set_Tool_IO_Mode(m_sockhand,num,state,1);

108、查询工具端数字 IO 状态 Get_Tool_IO_State

int Get_Tool_IO_State(SOCKHANDLE ArmSocket, float* IO_Mode, float *IO_state);	
函数功能	该函数用于查询工具端数字 IO 状态。
参数	① **ArmSocket** socket 句柄 ② **IO_Mode** 指定数字 IO 通道模式(范围 1~2), 0-输入模式，1-输出模式 ③ **IO_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。
示例	//查询工具端数字 IO 状态 float IO_Mode[2]; float IO_state[2]; ret = Get_Tool_IO_State(IO_Mode,IO_state)

109、设置工具端电源输出 Set_Tool_Voltage

int Set_Tool_Voltage(SOCKHANDLE ArmSocket, byte type, bool block);	
函数功能	该函数用于设置工具端电源输出。
参数	① **ArmSocket** socket 句柄 ② **type** 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。

示例	//设置工具端电源输出类型 5V
	byte type = 1;
	ret = Set_Tool_Voltage(m_sockhand, type, 1);

110、获取工具端电源输出 Get_Tool_Voltage

int Get_Tool_Voltage(SOCKHANDLE ArmSocket, byte *voltage);	
函数功能	该函数用于获取工具端电源输出。
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。

示例	//位置示教，x 轴负方向，速度 20%
	byte voltage;
	ret = Get_Tool_Voltage(m_sockhand,&voltage);

3.19 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。

111、配置手爪的开口度 Set_Gripper_Route

int Set_Gripper_Route(SOCKHANDLE ArmSocket, int min_limit, int max_limit, bool block);	
函数功能	该函数用于配置手爪的开口度。
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。

示例	//设置手爪开口最小值 70，最大值 500
	int min_limit = 70;
	int max_limit = 500;
	ret = Set_Gripper_Route(m_sockhand,min_limit, max_limit,1);

112、设置夹爪松开到最大位置 Set_Gripper_Release

int Set_Gripper_Release (SOCKHANDLE ArmSocket, int speed, bool block);	
函数功能	该函数用于控制手爪以指定速度张开到最大开口处
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。

示例	//手爪以 500 的速度松开
	int speed =500;
	ret = Set_Gripper_Release (m_sockhand, speed, 1);

113、设置夹爪夹取 Set_Gripper_Pick

int Set_Gripper_Pick(SOCKHANDLE ArmSocket, int speed, int force, bool block);	
函数功能	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。

参数	① **ArmSocket** socket 句柄 ② **speed** 手爪夹取速度 ，范围：1~1000，无单位量纲 ③ **force** 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。

示例	//设置夹取速度为 500，力矩阈值 200
	int speed = 500;
	int force = 200;
	ret = Set_Gripper_Pick(m_sockhand,speed,force,1);

114、设置夹爪持续夹取 Set_Gripper_Pick_On

int Set_Gripper_Pick_On(SOCKHANDLE ArmSocket, int speed, int force, bool block);	
函数功能	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。
参数	① **ArmSocket** socket 句柄 ② **speed** 手爪夹取速度 ，范围：1~1000，无单位量纲 ③ **force** 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。

示例	//设置夹取速度 500，夹取力矩阈值 200。
	int speed = 500;
	int force = 200;
	ret = Set_Gripper_Pick_On(m_sockhand,speed,force,1);

115、设置夹爪到指定开口位置 Set_Gripper_Position

int Set_Gripper_Position (SOCKHANDLE ArmSocket, int position, bool block);	
函数功能	该函数用于控制手爪到达指定开口度位置。
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。

示例	//控制手爪到达 500 开口度
	int position = 500;
	ret = Set_Gripper_Position(m_sockhand,position,1);

3.20 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。

116、进入拖动示教模式 Start_Drag_Teach

int Start_Drag_Teach (SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于控制机械臂进入拖动示教模式
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。

示例	//控制机械臂进入拖动示教模式
	ret = Start_Drag_Teach(m_sockhand,1);

117、退出拖动示教模式 Stop_Drag_Teach

int Stop_Drag_Teach (SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于控制机械臂退出拖动示教模式
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。

示例	//退出拖动示教模式
	ret = Stop_Drag_Teach(m_sockhand,1);

118、拖动示教轨迹复现 Run_Drag_Trajectory

int Run_Drag_Trajectory (SOCKHANDLE ArmSocket, bool block);	
函数功能	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 120 函数，否则会返回报错信息。
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。

示例	//复现拖动示教轨迹
	ret = Run_Drag_Trajectory (m_sockhand,1);

119、拖动示教轨迹复现暂停 Pause_Drag_Trajectory

int Pause_Drag_Trajectory (bool block);	
函数功能	该函数用于控制机械臂在轨迹复现过程中的暂停。
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。

示例	//轨迹复现暂停
	ret = Pause_Drag_Trajectory (m_sockhand,1);

120、拖动示教轨迹复现继续 Continue_Drag_Trajectory

int Continue_Drag_Trajectory (SOCKHANDLE ArmSocket, bool block);	
函数功能	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。
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。

示例	//轨迹复现继续
	ret = Continue_Drag_Trajectory (m_sockhand,1);

121、拖动示教轨迹复现停止 Stop_Drag_Trajectory

int Stop_Drag_Trajectory (SOCKHANDLE ArmSocket, bool block);	
函数功能	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。
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。

示例	//轨迹复现停止
	ret = Stop_Drag_Trajectory (m_sockhand,1);

122、运动到轨迹起点 Drag_Trajectory_Origin

int Drag_Trajectory_Origin (SOCKHANDLE ArmSocket, bool block);	
函数功能	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。
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。

示例	//运动到轨迹起点
	ret =Drag_Trajectory_Origin (m_sockhand,1);

123、复合模式拖动示教 Start_Multi_Drag_Teach

int Start_Multi_Drag_Teach(SOCKHANDLE ArmSocket, int mode,bool block);	
函数功能	开始复合模式拖动示。
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。

示例	//使用末端六维力，只动位置
	int mode = 1;
	ret = Start_Multi_Drag_Teach(m_sockhand,mode,1);

124、设置力位混合控制 Set_Force_Postion

int Set_Force_Postion(SOCKHANDLE ArmSocket, int sensor, int mode, int direction, int N, int block);	
函数功能	设置力位混合控制
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。

示例	//设置六维力基坐标系 Z 轴力控，1N 力大小
	ret = Set_Force_Postion (m_sockhand, 1, 2, 10, 1);

125、结束力位混合控制 Stop_Force_Postion

int Stop_Force_Postion (SOCKHANDLE ArmSocket, bool block);	
函数功能	结束力位混合控制.
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。

示例	//结束力位混合控制
	ret = Stop_Force_Postion_Move(m_sockhand,1);

3.21 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。

126、获取六维力数据 Get_Force_Data

int Get_Force_Data(SOCKHANDLE ArmSocket, float *Force);	
函数功能	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。
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。

示例	//获取六维力数据
	float Force[6];
	ret = Get_Force_Data(m_sockhand,Force);

127、清空六维力数据 Clear_Force_Data

int Clear_Force_Data(SOCKHANDLE ArmSocket, bool block);	
函数功能	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。
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。

示例	//清空六维力数据
	float Force[6];
	ret = Clear_Force_Data(m_sockhand,1);

128、设置六维力重心参数 Set_Force_Sensor

int Set_Force_Sensor (SOCKHANDLE ArmSocket);	
函数功能	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。
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。

示例	//设置六维力重心参数
	ret = Set_Force_Sensor (m_sockhand);

129、手动标定六维力数据 Manual_Set_Force

int Manual_Set_Force (int type,float *joint);	
函数功能	手动标定六维力数据；共四个点位，需要调用函数四次
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。

示例	//手动标定六维力数据，即第一个位置关节 1~7 的目标角度依次为 0°，1°，2°，3°，4°，5°，6°
	float joint1[6] = {0,1,2,3,4,5};
	ret = Manual_Set_Force (m_sockhand,1,joint1);

130、退出标定流程 Stop_Set_Force_Sensor

int Stop_Set_Force_Sensor (SOCKHANDLE ArmSocket, bool block);	
函数功能	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。
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。

示例	//退出标定流程
	ret = Stop_Set_Force_Sensor (m_sockhand,1);

3.21 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。

131、设置灵巧手手势序号 Set_Hand_Posture

int Set_Hand_Posture (SOCKHANDLE ArmSocket, int posture_num, bool block);	
函数功能	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。
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。

示例	//设置灵巧手执行 1 号手势
	int posture_num = 1;
	ret = Set_Hand_Posture (m_sockhand,posture_num, 1);

132、设置灵巧手动作序列序号 Set_Hand_Seq

int Set_Hand_Seq (SOCKHANDLE ArmSocket, int seq_num, bool block);	
函数功能	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。
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。

示例	//设置灵巧手执行 1 号动作序列
	int seq_num = 1;
	ret = Set_Hand_Seq (m_sockhand,seq_num, 1);

133、设置灵巧手角度 Set_Hand_Angle

int Set_Hand_Angle (SOCKHANDLE ArmSocket, int *angle, bool block);	
函数功能	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。
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。

示例	//设置灵巧手各手指动作
	const int angle[6]= {-1,100,200,300,400,500};
	ret = Set_Hand_Angle(m_sockhand,angle,1);

134、设置灵巧手各关节速度 Set_Hand_Speed

int Set_Hand_Speed (SOCKHANDLE ArmSocket, int speed, bool block);	
函数功能	设置灵巧手各关节速度
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。

示例	//设置灵巧手各手指速度
	int speed = 500;
	ret = Set_Hand_Speed (m_sockhand,speed, 1);

135、设置灵巧手各关节力阈值 Set_Hand_Force

int Set_Hand_Force (SOCKHANDLE ArmSocket, int force, bool block);	
函数功能	设置灵巧手各关节力阈值。
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。

示例	//设置灵巧手力阈值 500
	int force = 500;
	ret = Set_Hand_Force (m_sockhand,force, 1);

3.22 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。

136、设置末端 PWM 输出 Set_PWM

int Set_PWM (SOCKHANDLE ArmSocket, int Frq, int Dpulse, bool block);	
函数功能	该函数用于设置末端 PWM 输出。
参数	① **ArmSocket** socket 句柄 ② **Frq** PWM 输出频率，范围：100~10000Hz ③ **Dpulse** PWM 输出占空比，范围：0~100，精度不超过 1% 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。

示例	//设置末端 PWM 输出频率 200Hz,输出占空比 20
	int Frq = 200;
	int Dpulse = 20;
	ret = Set_PWM (m_sockhand,Frq,Dpulse,1);

137、停止末端 PWM 输出 Stop_PWM

int Stop_PWM (SOCKHANDLE ArmSocket, bool block);	
函数功能	该函数用于停止末端 PWM 输出。
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。

示例	//停止末端 PWM 输出
	ret = Stop_PWM (m_sockhand,1);

3.23 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。

138、查询一维力数据 Get_Fz

int Get_Fz (SOCKHANDLE ArmSocket, float *data);	
函数功能	该函数用于查询末端一维力数据。
参数	① **ArmSocket** socket 句柄 ② **data** 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。

示例	//查询末端一维力数据
	float data;
	ret = Get_Fz (m_sockhand,&data);

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。

139、清零一维力数据 Clear_Fz

int Clear_Fz (SOCKHANDLE ArmSocket, bool block);	
函数功能	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。
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。

示例	//清空末端一维力数据
	ret = Clear_Fz (m_sockhand,1);

140、自动标定末端一维力数据 Auto_Set_Fz

int Auto_Set_Fz (SOCKHANDLE ArmSocket);	
函数功能	该函数用于自动标定末端一维力数据。
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。

示例	//自动标定末端一维力
	ret = Auto_Set_Fz (m_sockhand);

141、手动标定末端一维力数据 Manual_Set_Fz

int Manual_Set_Fz (SOCKHANDLE ArmSocket, float* joint,float* joint2);	
函数功能	该函数用于手动标定末端一维力数据。
参数	① **ArmSocket** socket 句柄 ② **joint** 点位 1 关节角度 ③ **joint2** 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。

示例	//手动标定末端一维力数据
	float joint[7]={0,0,0,0,0,0,0};
	float joint1[7]={0,0,90,0,0,90,0};
	ret = Manual_Set_Fz (m_sockhand,joint,joint1);

3.25Modbus 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。

142、设置通讯端口 Modbus RTU 模式 Set_Modbus_Mode

Set_Modbus_Mode (SOCKHANDLE ArmSocket, int port,int baudrate,int timeout,bool block);	
函数功能	配置通讯端口 Modbus RTU 模式。
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。

示例	//配置通讯端口为末端接口板 RS485 端口，波特率配置 115200，超时时间 2 百毫秒
	int port = 1;
	int baudrate = 115200;
	int timeout = 2;
	ret = Set_Modbus_Mode (m_sockhand,port,baudrate,timeout,1);

143、关闭通讯端口 Modbus RTU 模式 Close_Modbus_Mode

Close_Modbus_Mode (SOCKHANDLE ArmSocket, int port , bool block);	
函数功能	关闭通讯端口 Modbus RTU 模式。
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。
示例	//关闭控制器 RS485 通讯端口 int port = 0;

	ret = Close_Modbus_Mode (m_sockhand,port,1);

144、读线圈 Get_Read_Coils

Get_Read_Coils (SOCKHANDLE ArmSocket, int port, int address, int num, int device, int* coils_data);	
函数功能	读线圈。
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。

示例	//通讯端口为末端接口板 RS485 端口，读线圈
	int port = 1;
	int address = 10;
	int num = 1;
	int device = 2;
	int coils_data;
	ret = Get_Read_Coils (m_sockhand,port, address, num, device, &coils_data);

145、读离散输入量 Get_Read_Input_Status

Get_Read_Input_Status (SOCKHANDLE ArmSocket, int port, int address, int num, int device, int* coils_data);	
函数功能	读离散输入量。
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。

示例	//通讯端口为末端接口板 RS485 端口，读离散输入量
	int port = 1;
	int address = 10;
	int num = 2;
	int device = 2;
	int coils_data;
	ret = Get_Read_Input_Status (m_sockhand, port, address, num, device, &coils_data);

146、读保持寄存器 Get_Read_Holding_Registers

Get_Read_Holding_Registers (SOCKHANDLE ArmSocket, int port, int address, int device, int *coils_data);	
函数功能	读保持寄存器。
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。

示例	//通讯端口为末端接口板 RS485 端口，读保持寄存器
	int port = 1;
	int address = 10;
	int device = 2;
	int coils_data;
	ret = Get_Read_Holding_Registers (m_sockhand, port, address, device, &coils_data);

147、读输入寄存器 Get_Read_Input_Registers

Get_Read_Input_Registers (SOCKHANDLE ArmSocket, int port, int address, int device, int coils_data);	
函数功能	读输入寄存器。

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。

示例	//通讯端口为末端接口板 RS485 端口，读输入寄存器
	int port = 1;
	int address = 10;
	int device = 2;
	int coils_data;
	ret = Get_Read_Input_Registers (m_sockhand, port, address, device, &coils_data);

148、写单圈数据 Write_Single_Coil

Write_Single_Coil (SOCKHANDLE ArmSocket, int port, int address, int data, int device, bool block);	
函数功能	写单圈数据。
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。

示例	//通讯端口为末端接口板 RS485 端口，写单圈数据
	int port = 1;
	int address = 10;
	int data = 1000;
	int device = 2;
	ret = Write_Single_Coil (m_sockhand, port, address, data, device, 1);

149、写单个寄存器 Write_Single_Register

Write_Single_Register (SOCKHANDLE ArmSocket, int port, int address, int data, int device, bool block);	
函数功能	写单个寄存器。
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。

示例	//通讯端口为末端接口板 RS485 端口，写单个寄存器
	int port = 1;
	int address = 10;
	int data = 1000;
	int device = 2;
	int coils_data;
	ret = Write_Single_Register (m_sockhand, port, address, data, device, 1);

3.26 升降机构

睿尔曼机械臂可集成自主研发升降机构。

150、移动平台运动速度 Set_Lift_Speed

Set_Lift_Speed (SOCKHANDLE ArmSocket, int speed,bool block);	
函数功能	设置移动平台运动速度。
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。

示例	//设置移动平台移动速度 50%，向下运动
	int speed = -50;
	ret = Set_Lift_Speed (m_sockhand,speed,1);

151、设置升降机构位置 Set_Lift_Height

Set_Lift_Height (SOCKHANDLE ArmSocket, int height,int speed,bool block);	
函数功能	升降机构位置闭环控制。
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。

示例	//设置目标高度 100mm，升降速度 50%
	int height = 100;
	int speed = 50;
	ret = Set_Lift_Height (m_sockhand,height,speed,1);

152、获取升降机构状态 Get_Lift_State

Get_Lift_Height (SOCKHANDLE ArmSocket, int *height,int *current,int* err);	
函数功能	升降机构状态。
参数	① **ArmSocket** socket 句柄 ② **height** 目标高度　　单位：mm　精度：1mm　范围：0~2300

	③ **urrent** 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	成功返回：0。失败返回:错误码, define.h 查询。

示例	//获取升降机构状态
	int height;
	int current;
	int err_flag;
	ret = Get_Lift_State(m_sockhand, &height, &current, &err_flag);

3.27 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。

153、开启透传力位混合控制补偿模式 Start_Force_Position_Move

Start_Force_Position_Move (SOCKHANDLE ArmSocket, bool block);	
函数功能	开启透传力位混合控制补偿模式。
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。

示例	//开启透传力位混合控制补偿模式
	ret = Start_Force_Position_Move(m_sockhand,1);

154、力位混合控制补偿透传模式（关节角）Force_Position_Move

Force_Position_Move_Joint(SOCKHANDLE ArmSocket, const float *joint,byte sensor,byte mode,int dir,float force,bool block);	
函数功能	力位混合控制补偿数据。
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。

示例	//透传力位混合补偿
	const float joint[7] = {1,2,3,4,5,6,7};
	byte sensor = 0;
	byte mode = 0;
	int dir = 0;
	float force = 15;
	ret = Force_Position_Move_Joint (m_sockhand,joint,sensor,mode,dir,force);

155、力位混合控制补偿透传模式（位姿）Force_Position_Move

Force_Position_Move_POSE(SOCKHANDLE ArmSocket, POSE pose,byte sensor,byte mode,int dir,float force,bool block);	
函数功能	力位混合控制补偿数据。
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。

示例	//力位混合补偿数据
	POSE pose;
	pose.px=-0.300;
	pose.py=-0.030;
	pose.pz=0.215;
	pose.rx=3.0;
	pose.ry=0.1;

	pose.rz=0.1;
	byte sensor = 0;
	byte mode = 0;
	int dir = 0;
	float force = 15;
	ret =
	Force_Position_Move_POSE(m_sockhand,pose,sensor,mode,dir,force,1);

156、关闭透传力位混合控制补偿模式 Stop_Force_Position_Move

Stop_Force_Position_Move (SOCKHANDLE ArmSocket, bool block);	
函数功能	关闭透传力位混合控制补偿模式。
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。

示例	//关闭透传力位混合控制补偿模式
	ret = Stop_Force_Position_Move(m_sockhand,1);

3.27 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。

157、设置算法的安装角度 setAngle

setAngle(float x,float y,float z);	
函数功能	设置算法的安装角度参数。
参数	Φ x x 轴安装角度。

	② y Y 轴安装角度。 ③ z Z 轴安装角度。

示例	//设置算法的安装角度
	float x = 0;
	float y = 0;
	float z = 0;
	setAngle(x, y, z);

158、设置算法接口的末端 DH 参数 setLwt

setLwt(int type);	
函数功能	设置算法接口的末端 DH 参数。
参数	① **type** **0**-标准版，1-一维力，2-六维力

示例	//设置算法接口的末端 DH 参数为一维力
	int type = 1;
	setLwt(type);

159、正解 Forward_Kinematics

Pose Forward_Kinematics(const float * joint);	
函数功能	用于睿尔曼机械臂正解计算。
参数	① **joint** 各个关节的关节角度
返回值	正解结果
示例	//机械臂正解计算 const float joint[6] = {0,0,0,0,0,0};

	Pose pose1 = Forward_Kinematics(joint);

160、逆解 inverse_Kinematics

int inverse_kinematics(const float *q_in, const Pose *q_pose, float *q_out, const uint8_t flag);	
函数功能	用于睿尔曼机械臂逆解计算。
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；

示例	//逆解计算
	Pose tar;
	tar.position.x=0;
	tar.position.y=0;
	tar.position.z=0;
	tar.euler.Phi=0;
	tar.euler.Psi=0;
	tar.euler.Theta=0;
	float q_in[6]={0},q_out[6]={0};
	int ret = inverse_kinematics(q_in, &tar, q_out,1);