JAKA

# 节卡机器人

## SDK 手册

## 【C++】

文档版本：V2.1.14

SDK 版本：V2.1.14

# 产权说明

## 【版本记录】

| 版本编号 | 版本日期 | 支持版本 | 变更概述 |
|---|---|---|---|
| V1.0.0 | 2020.05.27 | V1.4.10/V2.0.10 及以上 | 文档初始化 |
| V1.0.1 | 2020.07.17 | V1.4.10/V2.0.10 及以上 | 增加目录 3.20、3.21、4.63--4.67 |
| V1.0.2 | 2020.09.25 | V1.4.10/V2.0.10 及以上 | 增加目录 3.22、3.23、4.68--4.75 |
| V1.0.3 | 2020.10.22 | V1.4.10/V2.0.10 及以上 | 增加目录 3.24、4.76--4.78 |
| V1.0.4 | 2020.11.25 | V1.4.12/V2.0.12 及以上 | 修复阻塞运动卡顿问题，增加目录 4.79-4.83 部分 |
| V1.0.5 | 2020.12.08 | V1.4.12/V2.0.12 及以上 | 增加目录 3.20-3.22、4.15、4.17、4.86 |
| V1.0.6 | 2020.01.18 | V1.4.24/V2.0.24 及以上 | 1）修改目录 3.21<br>2）增加目录 4.36、4.30、4.89-4.100 |
| V1.0.7 | 2021.04.27 | V1.4.24/V1.5.12.17/V2.0.24 及以上 | 增加目录 3.28-3.32、增加目录 4.101-4.109 |
| V1.0.8 | 2021.08.30 | V1.4.24/V1.5.12.17/V2.0.24 及以上 | 增加 API 使用说明 |
| V2.1.1 | 2021.12.10 | V1.4.24/V1.5.12.17/V2.0.24 及以上 | 增加 ftp 接口 |
| V2.1.2 | 2022.7.1 | V1.4.24/V1.5.13.08/V2.0.24 及以上 | 1）修改目录结构<br>2）增加 API 使用说明 |
| V2.1.3 | 2022.11.28 | V1.4.24/V1.5.13.08/V2.0.24 及以上 | 增加圆弧运动圈数指定 |
| V2.1.4 | 2023.06.14 | V1.5.13.08/V2.0.24 及以上 | 1）增加 FTP 接口参数说明<br>2）修正设置 SDK 日志路径文档<br>3）扩展力矩传感器监测数据类型 |
| V2.1.7 | 2024.1.28 | V1.5.13.08 及以上 | 1）ADD:增加机器人安装角度的设置与获取接口<br>2）ADD:添加接口返回值-15<br>3）FIX:修改圆弧运动加速度单位 |
| v2.1.8 | 2024.3.21 | V1.5.13.08 及以上 | 增加获取 SDK 日志路径的接口说明 |
| V2.1.11 | 2024.4.30 | V1.5.13.08 及以上 | 1）FIX: 完善 10004 通信端口<br>2）FIX: 修复 set_ft_ctrl_frame 接口无法使用的问题<br>3）FIX: 修复 10001 端口指令错误重发的情况 |

| | | | |
|---|---|---|---|
| V2.1.14 | 2024.09.30 | V1.5.13.08 及以上 | Fix：<br>1、 重新定义 10004 端口机制，解决之前 10004 导致的 SDK 崩溃问题<br>2、 重新定义 motion block 机制，解决之前运动 block 不住的问题<br>3、 解决 inpos 到位不准问题<br>Add:<br>1、 增加设置 和 获取系统变量接口 set_user_var()、 get_user_var()<br>2、 增加 get_motion_status 接口：获取运动相关状态信息<br>3、 增加部分内嵌 S 相关接口<br>头文件：<br>1、 头文件注释全部更新为英文<br>2、 头文件前增加版权信息及版本信息 |

目录

# 1. 简介

JAKA SDK（软件开发工具包）定义了一套全面的 API 以实现对 JAKA 机器人的交互和控制，它为客户开发自己的应用程序来连接 JAKA 机器人控制器提供了一种方式。

JAKA API 基于网络通信协议 TCP/IP 与机器人通信，目前支持 C/ C++、C#和 Python 语言。

本文档将介绍 JAKA SDK（C++）中定义的数据类型和 API，主要适用于使用 C/ C++ 创建与虚拟或真实控制器通信的机器人应用程序的软件开发人员。对于需要了解 JAKA 机器人控制器应用程序的用户也会有一定帮助。

# 2. 文档须知

在编写本文档时，我们预期读者应具有以下基础：

- 熟悉 JAKA 机器人，对机器人控制有基本的了解；
- 能够在目标操作系统(Linux, Windows 等)上使用 C/ C++编程语言。

此外，以下为一些用户应注意的事项：

- 界面中的长度单位统一为毫米(mm)，角度单位统一为弧度(rad)；
- 版本号查询方法:在 windows 中右键单击 dll 文件，选择属性，在“详细信息”选项卡中可以看到版本信息。输入 Linux 下的命令" strings libjakaAPI。so | grep jakaAPI version "查询版本号；
- JAKA 使用的 SDK 编码方式为 UTF-8 编码；
- 在伺服模式下不能使用 joint_move, linear_move。

# 3. 数据结构

## 3.1 回调函数类型

用户注册机械臂发生异常时的回调函数类型

```
/**
* @brief 机械臂回调函数指针
*/
typedef void(*CallBackFuncType)(int);
```

## 3.2 接口调用返回值列表

```
#define ERR_SUCC                0             //success
#define ERR_FUCTION_CALL_ERROR  2             //interface error or controller not support
#define ERR_INVALID_HANDLER        -1         //invalid handler
#define   ERR_INVALID_PARAMETER   -2          //invalid parameter
#define ERR_COMMUNICATION_ERR   -3            //fail to connect
#define ERR_KINE_INVERSE_ERR    -4            //kine_inverse error
#define ERR_EMERGENCY_PRESSED   -5            //e-stop
#define ERR_NOT_POWERED         -6            //not power on
#define ERR_NOT_ENABLED         -7            //not enable
#define ERR_DISABLE_SERVOMODE    -8            //not in servo mode
#define ERR_NOT_OFF_ENABLE       -9            //must turn off enable before power off
#define ERR_PROGRAM_IS_RUNNING   -10           //cannot operate, program is running
#define ERR_CANNOT_OPEN_FILE     -11           //cannot open file, or file doesn't exist
#define ERR_MOTION_ABNORMAL      -12           //motion abnormal
#define ERR_FTP_PERFORM           -14          //ftp error
#define ERR_VALUE_OVERSIZE       -15           //socket msg or value oversize
#define ERR_KINE_FORWARD         -16           //kine_forward error
#define ERR_EMPTY_FOLDER         -17           //not support empty folder
#define ERR_PROTECTIVE_STOP      -20           // protective stop
#define ERR_EMERGENCY_STOP      -21            // protective stop
#define ERR_SOFT_LIMIT          -22            // on soft limit
#define ERR_CMD_ENCODE          -30            // fail to encode cmd string
#define ERR_CMD_DECODE          -31            // fail to decode cmd string
#define ERR_UNCOMPRESS          -32            // fail to uncompress port 10004 string
#define ERR_MOVEL               -40            // move linear error
#define ERR_MOVEJ               -41            // move joint error
#define ERR_MOVEC               -42            // move circular error
#define ERR_MOTION_TIMEOUT      -50            // block_wait timeout
```

```
#define ERR_POWERON_TIMEOUT      -51            // power on timeout
#define ERR_POWEROFF_TIMEOUT      -52           // power off timeout
#define ERR_ENABLE_TIMEOUT      -53             // enable timeoff
#define ERR_DISABLE_TIMEOUT      -54            // disable timeout
#define ERR_USERFRAME_SET_TIMEOUT      -55      // set userframe timeout
#define ERR_TOOL_SET_TIMEOUT      -56           // set tool timeout
#define ERR_IO_SET_TIMEOUT      -60             // set io timeout
```

## 3.3 接口调用返回值类型

```
typedef int errno_t;        //接口返回值类型
```

## 3.4 布尔类型

```
typedef int BOOL;           //布尔类型
```

## 3.5 笛卡尔空间位置数据类型

```
/**
* @brief 笛卡尔空间位置数据类型
*/
typedef struct
{
    double x;       ///< x 轴坐标，单位 mm
    double y;       ///< y 轴坐标，单位 mm
    double z;       ///< z 轴坐标，单位 mm
}CartesianTran;
```

## 3.6 欧拉角姿态数据类型

```
/**
* @brief 欧拉角姿态数据类型
*/
typedef struct
{
  double rx;    ///< 绕固定轴 X 旋转角度，单位：deg
  double ry;    ///< 绕固定轴 Y 旋转角度，单位：deg
  double rz;    ///< 绕固定轴 Z 旋转角度，单位：deg
}Rpy;
```

## 3.7 四元数姿态数据类型

```
/**
* @brief 四元数姿态数据类型
*/
typedef struct
{
  double s;
  double x;
  double y;
  double z;
}Quaternion;
```

## 3.8 笛卡尔空间位姿类型

```
/**
 *@brief 笛卡尔空间位姿类型
 */
typedef struct
{
  CartesianTran tran;   ///< 笛卡尔空间位置
  Rpy rpy;              ///< 笛卡尔空间姿态
}CartesianPose;
```

## 3.9 旋转矩阵数据类型

```
/**
* @brief 旋转矩阵数据类型
*/
typedef struct
{
  CartesianTran x;  ///< x 轴列分量
  CartesianTran y;  ///< y 轴列分量
  CartesianTran z;  ///< z 轴列分量
}RotMatrix;
```

## 3.10 程序运行状态枚举类型

```
/**
* @brief 程序运行状态枚举类型
*/
```

```
typedef enum
{
    PROGRAM_IDLE,       ///< 机械臂停止运行
    PROGRAM_RUNNING,    ///< 机械臂正在运行
    PROGRAM_PAUSED      ///< 机械臂暂停
}ProgramState;
```

## 3.11 坐标系选择枚举类型

```
/**
* @brief 坐标系选择枚举类型
*/
typedef enum
{
    COORD_BASE,     ///< 基坐标系
    COORD_JOINT,    ///< 关节空间
    COORD_TOOL      ///< 工具坐标系
}CoordType;
```

## 3.12 运动模式枚举类型

```
/**
* @brief 运动模式枚举
*/
typedef enum
{
    ABS = 0,          ///< 绝对运动
    INCR =1           ///< 增量运动
}MoveMode;
```

## 3.13 系统检测数据类型

```
/**
* @brief 系统监测数据类型
*/
typedef struct
{
    int scbMajorVersion;            ///<scb 主版本号
    int scbMinorVersion;            ///<scb 次版本号
    int cabTemperature;             ///<控制柜温度
    double robotAveragePower;       ///<控制柜总线平均功率
    double robotAverageCurrent;     ///<控制柜总线平均电流
```

```
    double instCurrent[6];          ///<机械臂 6 个关节轴的瞬时电流
    double instVoltage[6];          ///<机械臂 6 个关节轴的瞬时电压
    double instTemperature[6];      ///<机械臂 6 个关节轴的瞬时温度
}SystemMonitorData;
```

## 3.14 负载数据类型

```
/**
* @brief 负载数据类型
*/
typedef struct
{
    double mass;                ///<负载质量，单位：kg
    CartesianTran centroid;     ///<负载质心，单位：mm
}PayLoad;
```

## 3.15 关节位置数据类型

```
/**
* @brief 关节位置数据类型
*/
typedef struct
{
    double jVal[6];     ///< 6 关节位置值，单位：deg
}JointValue;
```

## 3.16 IO 类型

```
/**
* @brief IO 类型枚举
*/
typedef enum
{
    IO_CABINET,     ///< 控制柜面板 IO
    IO_TOOL,        ///< 工具 IO
    IO_EXTEND       ///< 扩展 IO
    IO_REALY,       ///< 继电器 IO，目前仅 CAB V3 支持 DO
    IO_MODBUS_SLAVE, ///< Modbus 从站 IO,从 0 索引
    IO_PROFINET_SLAVE, ///< Profinet 从站 IO,从 0 索引
    IO_EIP_SLAVE      ///< ETHRENET/IP 从站 IO,从 0 索引
}IOType;
```

## 3.17 机械臂状态数据类型

```
/**
* @brief 机械臂状态数据
*/
typedef struct
{
    BOOL estoped;       ///< 是否急停
    BOOL poweredOn;     ///< 是否打开电源
    BOOL servoEnabled;  ///< 是否使能
}RobotState;
```

## 3.18 机械臂力矩数据类型

```
/**
* @brief 机械臂力矩数据类型
*/
typedef struct
{
    double jTorque[6];  ///< 各关节力矩值，单位：N
}TorqueValue;
```

## 3.19 机械臂关节监测数据类型

```
/**
* @brief 机器人关节监测数据
*/
typedef struct
{
    double instCurrent;     ///< 瞬时电流
    double instVoltage;     ///< 瞬时电压
    double instTemperature; ///< 瞬时温度
    double instVel;         ///< 瞬时速度 控制器1.7.0.20及以上
    double instTorq;        ///< 瞬时力矩
} JointMonitorData;
```

## 3.20 机械臂监测数据类型

```
/**
* @brief 机械臂监测数据类型
*/
typedef struct
```

```
{
    double scbMajorVersion;                 ///< scb 主版本号
    double scbMinorVersion;                 ///< scb 小版本号
    double cabTemperature;                  ///< 控制器温度
    double robotAveragePower;               ///< 机械臂平均电压
    double robotAverageCurrent;             ///< 机械臂平均电流
    JointMonitorData jointMonitorData[6];   ///< 机械臂 6 个关节的监测数据
}RobotMonitorData;
```

## 3.21 力矩传感器监测数据类型

```
/**
* @brief 力矩传感器监测数据类型
*/
typedef struct
{
    char ip[20];                    ///< 力矩传感器 ip 地址
    int port;                       ///< 力矩传感器端口号
    PayLoad payLoad;                ///< 工具负载
    int status;                     ///< 力矩传感器状态
    int errcode;                    ///< 力矩传感器异常错误码
    double actTorque[6];            ///< 力矩传感器实际接触力值
    double torque[6];               ///< 力矩传感器原始读数值
    double realTorque[6];           ///< 力矩传感器实际接触力值（不随初始化选项变化）
}TorqSensorMonitorData;
```

## 3.22 机械臂状态监测数据类型（接口返回数据废弃，接口可用）

```
/**
* @brief 机器人状态监测数据,使用 get_robot_status 函数更新机器人状态数据
*/
typedef struct
{
    int errcode;                    ///< 机器人运行出错时错误编号，0 为运行正常，其它为运行异常
    int inpos;                      ///< 机器人运动是否到位标志，0 为没有到位，1 为运动到位
    int powered_on;                 ///< 机器人是否上电标志，0 为没有上电，1 为上电
    int enabled;                    ///< 机器人是否使能标志，0 为没有使能，1 为使能
    double rapidrate;               ///< 机器人运动倍率
```

```
    int protective_stop;                                ///< 机器人是否检测到碰撞，0 为没有检测到碰撞，1 为检测到碰撞
    int emergency_stop;                                 ///< 机器人是否急停,0为没有急停,1为急停
    int dout[256];                                      ///< 机器人控制柜数字输出信号,dout[0]为信号的个数
    int din[256];                                       ///< 机器人控制柜数字输入信号,din[0]为信号的个数
    double ain[256];                                    ///< 机器人控制柜模拟输入信号,ain[0]为信号的个数
    double aout[256];                                   ///< 机器人控制柜模拟输出信号,aout[0]为信号的个数
    int tio_dout[16];                                    ///< 机器人末端工具数字输出信号, tio_dout[0]为信号的个数
    int tio_din[16];                                    ///< 机器人末端工具数字输入信号, tio_din[0]为信号的个数
    double tio_ain[16];                                 ///< 机器人末端工具模拟输入信号, tio_ain[0]为信号的个数
    int tio_key[3];                                     ///< 机器人末端工具按钮 [0]free; [1]point; [2]pause_resume;
    Io_group extio;                                     ///< 机器人外部应用 IO
    Io_group modbus_slave;                              ///< 机器人 Modbus 从站
    Io_group profinet_slave;                            ///< 机器人 Profinet 从站
    Io_group eip_slave;                                 ///< 机器人 Ethernet/IP 从站
    unsigned int current_tool_id;                       ///< 机器人目前使用的工具坐标系 id
    double cartesiantran_position[6];                   ///< 机器人末端所在的笛卡尔空间位置
    double joint_position[6];                           ///< 机器人关节空间位置
    unsigned int on_soft_limit;                         ///< 机器人是否处于限位，0 为没有触发限位保护，1 为触发限位保护
    unsigned int current_user_id;                       ///< 机器人目前使用的用户坐标系 id
    int drag_status;                                    ///< 机器人是否处于拖拽状态，0 为没有处于拖拽状态，1 为处于拖拽状态
    RobotMonitorData robot_monitor_data;                ///< 机器人状态监测数据
    TorqSensorMonitorData torq_sensor_monitor_data;     ///< 机器人力矩传感器状态监测数据
    int is_socket_connect;                              ///< sdk 与控制器连接通道是否正常，0 为连接通道异常，1 为连接通道正常
} RobotStatus;
```

## 3.23 机械臂错误码数据类型

```
/**
* @brief 机械臂错误码数据类型
*/
typedef struct
{
```

```
    long code;                                  ///< 错误码编号
    char message[120];                          ///< 错误码对应提示信息
}ErrorCode;
```

## 3.24 轨迹复现配置参数存储数据类型

```
/**
* @brief 轨迹复现配置参数存储数据类型
*/
typedef struct
{
    double xyz_interval;                        ///< 空间位置采集精度
    double rpy_interval;                        ///< 姿态采集精度
    double vel;                                 ///< 执行脚本运行速度
    double acc;                                 ///< 执行脚本运行加速度
}TrajTrackPara;
```

## 3.25 多个字符串存储数据类型

```
/**
* @brief 多个字符串存储数据类型
*/
typedef struct
{
    int len;                                    ///< 字符串个数
    char name[128][128];                        ///< 数据存储二维数组
}MultStrStorType;
```

## 3.26 运动参数可选项

```
/**
* @brief 可选参数项
*/
typedef struct
{
    int executingLineId;                        ///< 控制命令 id 编号
}OptionalCond;
```

## 3.27 网络异常机械臂运动自动终止类型枚举

```
/**
```

```
 * @brief 网络异常机械臂运动自动终止类型枚举
 */
typedef enum
{
    MOT_KEEP,       ///< 网络异常时机械臂继续保持原来的运动
    MOT_PAUSE,      ///< 网络异常时机械臂暂停运动
    MOT_ABORT       ///< 网络异常时机械臂终止运动
}ProcessType;
```

## 3.28 机械臂柔顺控制参数类型

```
/**
* @brief 柔顺控制参数类型
*/
typedef struct
{
    int opt;      ///< 柔顺方向，可选值为 1 2 3 4 5 6 分别对应 fx fy fz mx my mz,0 代表没有勾选
    double ft_user; ///< 用户用多大的力才能让机械臂的沿着某个方向以最大速度进行运动
    double ft_rebound;   ///< 回弹力:机械臂回到初始状态的能力
    double ft_constant;  ///< 恒力
    int ft_normal_track; ///< 法向跟踪是否开启，0 为没有开启，1 为开启
} AdmitCtrlType;
```

## 3.29 机械臂柔顺控制参数类型

```
/**
* @brief 机械臂柔顺控制参数类型
*/
typedef struct
{
AdmitCtrlType admit_ctrl[6];
} RobotAdmitCtrl;
```

## 3.30 速度柔顺控制等级和比率等级设置

```
/**
* @brief 速度柔顺控制等级和比率等级设置
* 速度柔顺控制分三个等级，并且  1>rate1>rate2>rate3>rate4>0
* 等级为 1 时，只能设置 rate1,rate2 两个等级。rate3,rate4 的值为 0
* 等级为 2 时，只能设置 rate1,rate2，rate3 三个等级。rate4 的值为 0
* 等级为 3 时，能设置 rate1,rate2，rate3,rate4 4 个等级
```

```
*/
typedef struct
{
    int vc_level;                   //速度柔顺控制等级
    double rate1;                   //比率等级 1
    double rate2;                   //比率等级 2
    double rate3;                   //比率等级 3
    double rate4;                   //比率等级 4
}VelCom;
```

## 3.31 力传感器的受力分量和力矩分量

```
/**
* @brief 力传感器的受力分量和力矩分量
*/
typedef struct
{
    double fx;          // 沿 x 轴受力分量，单位：N
    double fy;          // 沿 y 轴受力分量，单位：N
    double fz;          // 沿 z 轴受力分量，单位：N
    double tx;          // 绕 x 轴力矩分量，单位：Nm
    double ty;          // 绕 y 轴力矩分量，单位：Nm
    double tz;          // 绕 z 轴力矩分量，单位：Nm
}FTxyz;
```

## 3.32 DH 参数

```
/**
 *  @brief DH 参数
 */
typedef struct
{
    double alpha[6];
    double a[6];
    double d[6];
    double joint_homeoff[6];
} DHParam;
```

## 3.33 RS485 信号量参数

```
/**
 *  @brief rs485 信号量参数
```

```
 */
typedef struct
{
    char sig_name[20];//标识名
    int chn_id;     //RS485 通道 ID
    int sig_type;   //信号量类型
    int sig_addr;   //寄存器地址
    int value;      //值   设置时无效
    int frequency;  //信号量在控制器内部刷新频率不大于 10
}SignInfo;
```

## 3.34 RS485 RTU 配置参数

```
/**
 *  @brief rs485RTU 配置参数
 */
typedef struct
{
    int chn_id;     //RS485 通道 ID   查询时 chn_id 作为输入参数
    int slaveId;    //当通道模式设置为 Modbus RTU 时，需额外指定 Modbus 从站节点 ID，其余模式可忽略
    int baudrate;   //波特率 4800,9600,14400,19200,38400,57600,115200,230400
    int databit;    //数据位 7，8
    int stopbit;    //停止位 1，2
    int parity;     //校验位 78-> 无校验 79->奇校验 69->偶校验
}ModRtuComm;
```

## 3.35 系统变量 Var

```
/**
 *@brief 控制器系统变量
 *@param id controller inner usage
 *@param value value type always double
 *@param alias variable alias which is less than 100 bytes
 */
typedef struct {
      int id;
      double value;
          char alias[100];
} UserVariable;

/**
 * @brief 系统变量最大数量为 100
```

```
*/
typedef struct{
        UserVariable v[100];
} UserVariableList;

```

## 3.36 机器人状态 simple

```
/**
* @brief 机器人状态（simple）
*/
typedef struct
{
    int errcode;      ///< 0: 正常，其他：错误码
    char errmsg[200]  ///< 控制器错误信息
    int powered_on;   ///< 0: 下电，1: 上电
    int enabled;      ///< 0: 下使能，1: 使能
} RobotStatus_simple;
```

## 3.37 运动状态

```
/**
 *@brief 运动状态
 */
typedef struct
{
    int motion_line;          ///< 运动指令编号，当程序执行时为该运动对应的行号
    int motion_line_sdk;      ///< 保留
    BOOL inpos;               ///< 运动段是否运动完成状态标志
    BOOL err_add_line;        ///< 添加运动段失败状态标志
    int queue;                ///< 缓冲区队列中的运动指令数
    int active_queue;         ///< 转接缓冲区中的运动指令数
    BOOL queue_full;          ///< 缓冲区队列满标志，该标志为 TURE 时用户下发运动指令将丢弃
    BOOL paused;              ///< 运动指令暂停标志
} MotionStatus;
```

# 4. API

## 4.1 机械臂基础

### 4.1.1 机械臂控制类构造函数

```
/**
* @brief 机械臂控制类构造函数，所有的接口都在该类中。
*/
JAKAZuRobot();
```

### 4.1.2 机械臂登录

```
/**
* @brief 连接机械臂控制器，该接口启动成功，是其他接口调用的基础。
* @param ip  控制器 ip 地址
* @return ERR_SUCC 成功 其他失败
*/
errno_t  login_in(const char* ip);
```

### 4.1.3 机械臂注销

```
/**
* @brief 断开控制器连接，该接口调用成功后，除了 login_in 之外的函数均无法调用。
* @return ERR_SUCC 成功 其他失败
*/
errno_t  login_out();
```

### 4.1.4 机械臂上电

```
/**
* @brief 打开机械臂电源
* @return ERR_SUCC 成功 其他失败
*/
errno_t  power_on();
```

### 4.1.5 机械臂下电

```
/**
* @brief 关闭机械臂电源
* @return ERR_SUCC 成功 其他失败
*/
errno_t  power_off();
```

### 4.1.6 机械臂关机

```
/**
* @brief 机械臂控制柜关机
* @return ERR_SUCC 成功 其他失败
*/
errno_t  shut_down();
```

### 4.1.7 机械臂上使能

```
/**
* @brief 控制机械臂上使能
* @return ERR_SUCC 成功 其他失败
*/
errno_t  enable_robot();
```

### 4.1.8 机械臂下使能

```
/**
* @brief 控制机械臂下使能
* @return ERR_SUCC 成功 其他失败
*/
errno_t  disable_robot();
```

### 4.1.9 控制机械臂进入或退出拖拽模式

```
/**
* @brief 控制机械臂进入或退出拖拽模式
* @param enable  TRUE 为进入拖拽模式，FALSE 为退出拖拽模式
* @return ERR_SUCC 成功 其他失败
*/
errno_t  drag_mode_enable(BOOL enable);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
//拖拽模式
int  example_drag()
{
    BOOL in_drag;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.152");
    demo.power_on();
    demo.enable_robot();
    //确认机械臂是否在拖拽模式下
    demo.is_in_drag_mode(&in_drag);
    std::cout << "in_drag is : " << in_drag << std::endl;
    //使能拖拽模式
    demo.drag_mode_enable(TRUE);
    Sleep(10000);
    demo.is_in_drag_mode(&in_drag);
    std::cout << "in_drag is : " << in_drag << std::endl;
    //去使能拖拽模式
    demo.drag_mode_enable(FALSE);
    Sleep(100);
    demo.is_in_drag_mode(&in_drag);
    std::cout << "in_drag is : " << in_drag << std::endl;
    while (1)
    {
        demo.is_in_drag_mode(&in_drag);
        std::cout << "in_drag is : " << in_drag << std::endl;
        Sleep(100);
    }
    return 0;
}
```

## 4.1.10 查询机械臂是否处于拖拽模式

```
/**
* @brief 查询机械臂是否处于拖拽模式
* @param in_drag 查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  is_in_drag_mode(BOOL* in_drag);
```

### 4.1.11 获取 SDK 版本号

```
/**
* @brief 获取机械臂控制器版本号
* @param version SDK 版本号
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_sdk_version(char* version);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//4.63 获取 sdk 版本号
int example_getsdk_version()
{
    //实例 API 对象 demo
    JAKAZuRobot demo;
    char ver[100];
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //查询当前 SDK 版本
    demo.get_sdk_version(ver);
    std::cout << " SDK version is :" << ver << std::endl;
    return 0;
}
```

### 4.1.12 获取 SDK 日志路径（静态方法）

```
/**
* @brief 获取 SDK 日志路径， 用于 SDK 对象创建前
* @param filepath SDK 日志路径
* @return ERR_SUCC 成功 其他失败
*/
errno_t  static_get_SDK_filepath(char* filepath);
```

### 4.1.13 获取 SDK 日志路径

```
/**
* @brief 获取 SDK 日志路径
* @param filepath SDK 日志路径
* @return ERR_SUCC 成功 其他失败
```

```
*/
errno_t  get_SDK_filepath(char* filepath);
```

## 4.1.14 设置 SDK 日志路径（静态方法）

```
/**
* @brief 设置 SDK 日志路径， 用于 SDK 对象创建前
* @param filepath SDK 日志路径
* @return ERR_SUCC 成功 其他失败
*/
errno_t  static_set_SDK_filepath(char* filepath);
```

## 4.1.15 设置 SDK 日志路径

```
/**
* @brief 设置 SDK 日志路径
* @param filepath SDK 日志路径
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_SDK_filepath(char* filepath);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//设置 SDK 日志路径
int example_set_SDK_filepath()
{
//设置 SDK 日志路径
    char path[20] = "D://test.log";
    int ret;
    JAKAZuRobot demo;
ret = demo.set_SDK_filepath(path);//设置 SDK 文件路径
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    std::cout << ret << std::endl;
    return 0;
}
```

## 4.1.16 设置 SDK 是否开启调试模式

```
/**
* @brief 设置是否开启调试模式，选择 TRUE 时，开始调试模式，此时会在标准输出流中输出调试信息，选择 FALSE 时，不输出调试信息
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_debug_mode(BOOL mode);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//设置 SDK 是否开启调试模式
int example_set_debug_mode()
{
    BOOL mode;
    JAKAZuRobot demo;
    //设置调试模式，将在终端上打印调试信息
    demo.set_debug_mode(TRUE);
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    return 0;
}
```

## 4.1.17 获取控制器 IP

```
/**
* @brief 获取控制器 IP
* @param controller_name 控制器名字
* @param ip_list 控制器 ip 列表，控制器名字为具体值时返回该名字所对应的控制器 IP 地址，控制器名字为空时，返回网段类内的所有控制器 IP 地址
* @return ERR_SUCC 成功 其他失败
* 当app启动时，该函数失效
*/
errno_t  get_controller_ip(char* controller_name, char* ip_list);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//获取控制器 ip
```

```
int example_get_controller_ip()
{
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    char ip_list[2000] = { 0 };
    char  controller_name1[50] = "";

    //获取控制器 ip
    ret = demo.get_controller_ip( controller_name1, ip_list);
    std::cout << ret << std::endl;
    std::cout << " ip_list is :\n" << ip_list << std::endl;
    return 0;
}
```

# 4.2 机械臂运动

## 4.2.1 机械臂手动模式下运动

```
/**
* @brief 控制机械臂手动模式下 jog 运动
* @param aj_num 关节空间下代表关节号[0-5]，笛卡尔下依次为轴 x，y，z，rx，ry，rz
* @param move_mode 机械臂运动模式，增量运动、者绝对运动(即持续 jog 运动)和连续运动,参考 2.13 选择合适类型 ，可选类型有 INCR ABS
* @param coord_type 机械臂运动坐标系，工具坐标系，基坐标系（当前的世界/用户坐标系）或关节空间，参考 2.12 选择合适类型 可选类型有
* @param vel_cmd 指令速度，旋转轴或关节运动单位为 deg/s，移动轴单位为 mm/s
* @param pos_cmd 指令位置，旋转轴或关节运动单位为 deg，移动轴单位为 mm 当 move_mode 是连续时，参数可忽略
* @return ERR_SUCC 成功 其他失败
*/
errno_t  jog(int aj_num, MoveMode move_mode, CoordType coord_type, double vel_cmd, double pos_cmd);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//机械臂手动运动
int main()
{
    //实例 API 对象 demo
```

```
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.152");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //手动运动，其中 INCR 代表增量运动，0.5 代表速度为 0.5deg/s
    demo.jog(1, INCR, COORD_JOINT , 0.5, 30);
    Sleep(5000);
//手动运动停止
    demo.jog_stop(0);
    //机械臂去使能
    demo.disable_robot();
    //机械臂下电
    demo.power_off();
    return 0;
}
```

### 4.2.2 机械臂手动模式下运动停止

```
/**
* @brief 控制机械臂手动模式下运动停止
* @param num  机械臂轴号 0-5，num 为-1 时，停止所有轴
* @return ERR_SUCC 成功 其他失败
*/
errno_t  jog_stop();
```

### 4.2.3 机械臂关节运动

```
/**
* @brief 机械臂关节运动
* @param joint_pos 机械臂关节运动目标位置
* @move_mode 指定运动模式：增量运动、绝对运动
* @param is_block 设置接口是否为阻塞接口，TRUE 为阻塞接口 FALSE 为非阻塞接口 ，阻塞时默认 30s 超时
* @param speed 机械臂关节运动速度，单位：deg/s
* @param acc 机械臂关节运动角加速度
* @param tol 机械臂关节运动终点误差，该参数使得两段运动之间更平滑，使用该参数时需要连续多段运动，且为非阻塞状态。
* @param option_cond 机械臂关节可选参数，如果不需要，该值可不赋值,填入空指针就可
* @return ERR_SUCC 成功 其他失败
*/
```

```
errno_t  joint_move(const JointValue* joint_pos, MoveMode move_mode, BOOL is_block, double speed, double acc, double tol, const OptionalCond* option_cond);
```

### 4.2.4 机械臂末端直线运动

```
/**
* @brief 机械臂末端直线运动
* @param end_pos 机械臂末端运动目标位置
* @move_mode 指定运动模式：增量运动、绝对运动
* @param is_block 设置接口是否为阻塞接口，TRUE 为阻塞接口 FALSE 为非阻塞接口，阻塞时默认 30s 超时
* @param speed 机械臂直线运动速度，单位：mm/s
* @param acc 机械臂直线运动线加速度
* @param tol 机械臂直线运动终点误差.该参数使得两段运动之间更平滑，使用该参数时需要连续多段运动，且为非阻塞状态。
* @param option_cond 机械臂直线运动可选参数，如果不需要，该值可不赋值，填入空指针就可
* @param ori_vel 姿态速度，单位 deg/s
* @param ori_acc 姿态加速度，单位 deg/s^2
* @return ERR_SUCC 成功 其他失败
*/
* 奇异点常出现的三种情况：
* 工具末端位置在轴线 Z1 和 Z2 构成的平面上；
* 轴线 Z2,Z3,Z4 共面；
* 关节 5 角度为 0 或 180°，即 Z4,Z6 平行；

errno_t  linear_move(const CartesianPose* end_pos, MoveMode move_mode, BOOL is_block, double speed, double accel = 50, double tol = 0, const  OptionalCond* option_cond = nullptr, double ori_vel = 3.14, double ori_acc = 12.56);
```

代码示例：

```
int example_linear_move()
{
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    RobotStatus status;
    //登陆控制器，需要将 192.168.2.229 替换为自己控制器的 IP
    demo.login_in("192.168.2.160");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //定义并初始化 CartesianPose 变量，旋转角为弧度。
    CartesianPose cart;
    cart.tran.x = 300; cart.tran.y = 300; cart.tran.z = 100;
```

```
    cart.rpy.rx = 3.14; cart.rpy.ry = 0 ; cart.rpy.rz = 0;
    //笛卡尔空间运动，其中 ABS 代表绝对运动，FALSE 代表指令是非阻塞的，10 代表速度为 10mm/s
    printf("rx=%f , ry=%f, rz=%f\n", cart.rpy.rx, cart.rpy.ry, cart.rpy.rz);
    demo.linear_move(&cart, ABS, FALSE, 200, 10 ,1,NULL);
    for (int i = 10; i > 0;i--) {
        cart.tran.x = 150; cart.tran.y = 300;
        //笛卡尔空间扩展运动，其中 ABS 代表绝对运动，FALSE 代表指令是非阻塞的，20 代表最大速度为 20mm/s ,10 代表加速度为 10mm/s2,5 代表圆弧过度半径为 5mm
        demo.linear_move(&cart, ABS, FALSE, 20, 10, 5, NULL);
        cart.tran.x = 150; cart.tran.y = 250;
        demo.linear_move(&cart, ABS, FALSE, 20, 10, 5, NULL);
        cart.tran.x = 225; cart.tran.y = 250;
        demo.linear_move(&cart, ABS, FALSE, 20, 10, 5, NULL);
        cart.tran.x = 300; cart.tran.y = 250;
        demo.linear_move(&cart, ABS, FALSE, 20, 10, 5, NULL);
        cart.tran.x = 300; cart.tran.y = 300;
        demo.linear_move(&cart, ABS, FALSE, 20, 10, 5, NULL);
        Sleep(3000);
    }
    demo.login_out();
    return 0;
}
```

### 4.2.5 机械臂圆弧运动

```
/**
* @brief 机械臂末端圆弧运动，该接口使用当前点和输入的两个点规划圆形轨迹。
* @param end_pos 机械臂末端运动目标位置
* @param mid_pos 机械臂末端运中间点
* @move_mode 指定运动模式：增量运动、绝对运动
* @param is_block 设置接口是否为阻塞接口，TRUE 为阻塞接口 FALSE 为非阻塞接口
* @param speed 机械臂直线运动速度，单位：mm/s
* @param acc 机械臂笛卡尔空间加速度，单位：mm/s^2
* @param tol 机械臂笛卡尔空间运动终点误差
* @param option_cond 机械臂关节可选参数，如果不需要，该值可不赋值,填入空指针就可
* @param circle_cnt 指定机器人圆弧运动圈数。为 0 时等价于 circular_move
* @param circle_mode 指定机器人圆弧运动模式，参数解释如下：
- 0：固定采用起始姿态到终止姿态旋转角度小于 180°的轴角进行姿态变化；(当前方案)
- 1：固定采用起始姿态到终止姿态旋转角度大于 180°的轴角进行姿态变化；
- 2：根据中间点姿态自动选择选择角度小于 180°还是大于 180°；
- 3： 姿态夹角与圆弧轴线始终保持一致。(当前整圆运动)
* @return ERR_SUCC 成功 其他失败
*/
```

19. errno_t circular_move(const CartesianPose *end_pos, const CartesianPose *mid_pos, MoveMode move_mode, BOOL is_block, double speed, double accel, double tol, const OptionalCond *option_cond = nullptr, int circle_cnt = 0, int circle_mode = 0);

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
define PI = 3.14
//机械臂圆弧运动 ， 关节速度上限 3.14deg/s
int example_circle_move()
{
    OptionalCond opt;
    CartesianPose end_p,mid_p;
    end_p.tran.x = -200; end_p.tran.y = 400; end_p.tran.z = 400;
    end_p.rpy.rx = -PI/2 ; end_p.rpy.ry = 0 ; end_p.rpy.rz =0 ;
    mid_p.tran.x = -300; mid_p.tran.y = 400; mid_p.tran.z = 500;
    mid_p.rpy.rx = -PI/2 ; mid_p.rpy.ry = 0 ; mid_p.rpy.rz =0 ;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //定义并初始化关 JointValue 变量
    JointValue joint_pos = { 85.76 , -6.207 , 111.269 , 74.938 , 94.24 , 0  };
    //关节空间运动，其中 ABS 代表绝对运动，TRUE 代表指令是阻塞的，1 代表速度为 1deg/s
    demo.joint_move(&joint_pos, ABS, TRUE, 1);
    //圆弧运动，其中 ABS 代表绝对运动，TRUE 代表指令是阻塞的，20 代表直线速度为 20mm/s，1 代表加速度，0.1 代表机械臂终点误差，opt 为可选参数，3 代表圈数为 3。
    demo.circular_move(&end_p, &mid_p, ABS, TRUE, 20, 1, 0.1,&opt);
    return 0;
}
```

代码示例2：

```
#include "jktypes.h"
#include <JAKAZuRobot.h>

#define PI (3.1415926)
#define PI_2 (1.5707963)
int example_circular_move()
{
    JAKAZuRobot robot;
```

```
    robot.login_in("192.168.20.138");
    robot.power_on();
    robot.enable_robot();

    CartesianPose start_pos {-251.054, -48.360, 374.000, 3.14, 0, 1.57};  // 开始点
    CartesianPose mid_pos = {-555.050, 116.250, 374.000, 3.14, 0, 1.57};  // 中间点
    CartesianPose end_pos = {-295.050, 267.450, 374.000, 3.14, 0, 1.57};  // 结束点

    robot.jog_stop(-1); // 停止当前关节运动

     JointValue ref_jv {0, 90, 90, 90, -90, 0};  // 先移动到到开始点附近
    robot.joint_move(&ref_jv, MoveMode::ABS, true, 20);

    JointValue start_jv;
     robot.get_joint_position(&ref_jv);  // 获取当前关节角
     robot.kine_inverse(&ref_jv, &start_pos, &start_jv); // 以当前关节角为参考,计算开始关节角
    robot.joint_move(&start_jv, MoveMode::ABS, true, 80);  // 移动到开始关节角位置

    // 指定旋转 3 圈
    robot.circular_move(&end_pos, &mid_pos, MoveMode::ABS, true, 120, 100, 0.1, &opt,
3);

    robot.disable_robot();
    robot.power_off();
    robot.login_out();
}
```

## 4.2.6 机械臂运动终止

```
/**
* @brief 终止当前机械臂运动，机械臂脚本程序也会停止
* @return ERR_SUCC 成功 其他失败
*/
errno_t  motion_abort();
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//机械臂运动终止
int example_motion_abort()
{
    //实例 API 对象 demo
```

```
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //定义并初始化关 JointValue 变量
    printf("start_move");
    JointValue joint_pos = { 0 , 0 , 50 , 0 , 0 , 0  };
    //关节空间运动，其中 ABS 代表绝对运动，TRUE 代表指令是阻塞的，1 代表速度为 1deg/s
    demo.joint_move(&joint_pos, ABS, FALSE, 1);
    Sleep(500);
    //运动 0.5s 后终止
    demo.motion_abort();
    printf("stop_move");
    return 0;
}
```

## 4.2.7 查询机械臂运动是否停止

```
/**
* @brief 查询机械臂运动是否停止
* @param in_pos 查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  is_in_pos(BOOL* in_pos);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
 #define PI 3.1415926
//查询机械臂运动是否停止
int example_is_in_pos()
{
    //实例 API 对象 demo
    JAKAZuRobot demo;
    BOOL in_pos;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.152");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
```

```
    demo.enable_robot();
    while (1)
    {
        //查询是否运动停止
        demo.is_in_pos(&in_pos);
        std::cout << " in_pos is :" << in_pos << std::endl;
        Sleep(200);
    }
    return 0;
}
```

### 4.2.8 查询机械臂运动状态

```
/**
 *@brief get motion status
 *@param status struct of motion status
 */
errno_t get_motion_status(MotionStatus *status);
```

### 4.2.9 机械臂设置阻塞运动超时时间

```
/**
* @brief 设置机器人阻塞等待超时时间
* @param seconds 时间参数，单位秒 大于0.5
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_block_wait_timeout(float seconds);
```

## 4.3 机械臂操作信息设置与获取

### 4.3.1 获取机械臂状态监测数据（唯一多线程安全接口）

```
/**
* @brief 获取机械臂状态数据，多线程安全
* @param status 机械臂状态
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_robot_status(RobotStatus* status);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
```

```
#include <windows.h>

//获取机械臂状态监测数据
int example_get_robot_status()
{
    //实例 API 对象 demo
    JAKAZuRobot demo;
    RobotStatus robstatus;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //获取机械臂状态监测数据
    demo.get_robot_status(&robstatus);
    demo.login_out();
    return 0;
}
```

### 4.3.2 获取当前机械臂关节角度（simple）

```
/**
* @brief 获取当前机械臂关节角度，将关节角矩阵保存在输入参数joint_position中
* @param joint_position 关节角度查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_joint_position(JointValue* joint_position);
```

### 4.3.3 获取当前机械臂关节角度

```
/**
* @brief 获取当前机械臂关节角度，将关节角矩阵保存在输入参数 joint_position 中
* @param joint_position 关节角度查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_joint_position(JointValue* joint_position);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//获取当前机械臂关节角
int example_get_joint_position()
{
    //实例 API 对象 demo
    JAKAZuRobot demo;
```

```
    JointValue jot_pos;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //获取当前关节角
    demo.get_joint_position(&jot_pos);
    for (int i = 0; i < 6; i++)
    {
        std::cout << "joint [" << i+1 <<"] is :"<< jot_pos.jVal[i] << std::endl;
    }
    return 0;
}
```

### 4.3.4 设置机械臂状态数据自动更新时间间隔

```
/**
* @brief 设置机械臂状态数据自动更新时间间隔,特指 get_robot_status()
* @param millisecond 时间参数，单位：ms
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_status_data_update_time_interval(float millisecond);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//设置状态自动更新时间
int example_set_status_data_updata_interval()
{
    float milisec = 100;
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //设置柔顺力矩条件
```

```
    ret = demo.set_status_data_update_time_interval(milisec);
    std::cout << ret << std::endl;
    return 0;
}
```

### 4.3.5 获取当前设置下工具末端的位姿

```
/**
* @brief 获取当前设置下工具末端的位姿，将位姿保存在输入参数 tcp_position 中
* @param tcp_position 工具末端位置查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_tcp_position(CartesianPose* tcp_position);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//获取工具末端位姿
int example_get_tcp_position()
{
    //实例 API 对象 demo
    JAKAZuRobot demo;
    CartesianPose tcp_pos;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //获取工具末端位姿
    demo.get_tcp_position(&tcp_pos);
    std::cout << "tcp_pos is :\n x: " << tcp_pos.tran.x << "  y: " << tcp_pos.tran.y << "  z: " << tcp_pos.tran.z << std::endl;
    std::cout << "rx: " << tcp_pos.rpy.rx << "  ry: " << tcp_pos.rpy.ry << "  rz: " << tcp_pos.rpy.rz << std::endl;
    return 0;
}
```

### 4.3.6 设定用户坐标系信息

```
/**
* @brief 设置指定编号的用户坐标系信息
```

```
 * @param id 用户坐标系编号 ，取值范围为[1,10]
 * @param user_frame 用户坐标系偏置值
 * @param name 用户坐标系别名
 * @return ERR_SUCC 成功 其他失败
 */
errno_t  set_user_frame_data(int id, const CartesianPose* user_frame, const char* name);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//用户坐标系查看及调整
int example_user_frame()
{
    int id_ret, id_set;
    id_set = 2;
    CartesianPose tcp_ret, tcp_set;
    char name[50] = "test";
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    //查询当前使用的用户坐标系 ID
    demo.get_user_frame_id(&id_ret);
    //获取当前使用的用户坐标系信息
    demo.get_user_frame_data(id_ret, &tcp_ret);
    printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tran.y, tcp_ret.tran.y);
    printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.rz);
    //初始化用户坐标系坐标
    tcp_set.tran.x = 0; tcp_set.tran.y = 0; tcp_set.tran.z = 10;
    tcp_set.rpy.rx = 120 ; tcp_set.rpy.ry = 90 ; tcp_set.rpy.rz = -90 ;
    //设置用户坐标系信息
    demo.set_user_frame_data(id_set, &tcp_set, name);
    //切换当前使用的用户坐标系坐标
    demo.set_user_frame_id(id_set);
    //查询当前使用的用户坐标系 ID
    demo.get_user_frame_id(&id_ret);
    //获取设置的用户坐标系信息
    demo.get_user_frame_data(id_ret, &tcp_ret);
    printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tran.y, tcp_ret.tran.y);
    printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.rz);
    return 0;
}
```

### 4.3.7 获取用户坐标系信息

```
/**
* @brief 查询当前使用的用户坐标系信息
* @param id 用户坐标系 ID
* @param tcp 用户坐标系偏置值
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_user_frame_data(int id, CartesianPose* tcp);
```

### 4.3.8 设置当前使用的用户坐标系 ID

```
/**
* @brief 设置当前使用的用户坐标系 ID
* @param id 用户坐标系 ID ，取值范围为[0,10]，其中 0 为世界坐标系
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_user_frame_id(const int id);
```

### 4.3.9 查询当前使用的用户坐标系 ID

```
/**
* @brief 查询当前使用的用户坐标系 ID
* @param id 获取的结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_user_frame_id(int* id);
```

### 4.3.10 配置工具信息

```
/**
* @brief 配置指定编号的工具信息
* @param id 工具编号,取值范围为[1,10]
* @param tcp 工具坐标系相对法兰坐标系偏置
* @param name 指定工具的别名
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_tool_data(int id, const CartesianPose* tcp, const char* name);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
```

```
#include <windows.h>

//工具坐标系查看及调整
int example_tool()
{
    int id_ret,id_set;
    id_set = 2;
    CartesianPose tcp_ret,tcp_set;
    char name[50] = "test";
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    //查询当前使用的工具 ID
    demo.get_tool_id(&id_ret);
    //获取当前使用的工具信息
    demo.get_tool_data(id_ret,&tcp_ret);
    printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tran.y, tcp_ret.tran.y);
    printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.rz);
    //初始化工具坐标
    tcp_set.tran.x = 0; tcp_set.tran.y = 0; tcp_set.tran.z = 10;
    tcp_set.rpy.rx = 120 ; tcp_set.rpy.ry = 90 ; tcp_set.rpy.rz = -90 ;
    //设置工具信息
    demo.set_tool_data(id_set, &tcp_set, name);
    //切换当前使用的工具坐标
    demo.set_tool_id(id_set);
    //查询当前使用的工具 ID
    demo.get_tool_id(&id_ret);
    //获取设置的工具信息
    demo.get_tool_data(id_ret, &tcp_ret);
    printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tran.y, tcp_ret.tran.y);
    printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.rz);
    return 0;
}
```

## 4.3.11 获取工具信息

```
/**
* @brief 查询当前使用的工具信息
* @param id 工具 ID 查询结果
* @param tcp 工具坐标系相对法兰坐标系偏置
* @return ERR_SUCC 成功 其他失败
*/
```

```
errno_t  get_tool_data(int* id, CartesianPose* tcp);
```

### 4.3.12 查询当前使用的工具 ID

```
/**
* @brief 查询当前使用的工具 ID
* @param id 工具 ID 查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_tool_id(int* id);
```

### 4.3.13 设置当前使用的工具 ID

```
/**
* @brief 设置当前使用的工具 ID，在网络波动时，切换 ID 后需要一定延时才能生效。
* @param id 工具坐标系 ID ,取值范围为[0,10]，0 为不使用工具即法兰中心
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_tool_id(const int id);
```

### 4.3.14 设置数字输出变量

```
/**
* @brief 设置数字输出变量(DO)的值
* @param type DO 类型
* @param index DO 索引（从 0 开始）
* @param value DO 设置值
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_digital_output(IOType type, int index, BOOL value);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
//设置与查询数字输出
int main()
{
    BOOL DO3;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.152");
```

```
    //机械臂上电
    demo.power_on();
    //查询 do3 的状态
    demo.get_digital_output(IO_CABINET, 2, &DO3);
    printf("D03 = %d\n", DO3);
    //io_cabinet 是控制柜面板 IO，2 代表 DO3，1 对应要设置的 DO 值。
    demo.set_digital_output(IO_CABINET, 2, 1);
    Sleep(1000);//需要 window.h 延时 1s
    //查询 do3 的状态
    demo.get_digital_output(IO_CABINET, 2, &DO3);
    printf("D03 = %d\n", DO3);
    return 0;
}
```

## 4.3.15 设置模拟输出变量

```
/**
* @brief 设置模拟输出变量的值(AO)的值
* @param type AO 类型
* @param index AO 索引 （从 0 开始）
* @param value AO 设置值
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_analog_output(IOType type, int index, float value);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
//设置与查询模拟输出
int main()
{
    JAKAZuRobot demo;
    demo.login_in("192.168.2.152");
    demo.power_on();
    float AO35;
    //查询 Ao 的状态
    demo.get_analog_output(IO_CABINET, 34, &AO35);
    printf("A035 = %f\n", AO35);
    //io_cabinet 是控制柜面板 IO，2 代表 DO3，1.5 对应要设置的 DO 值。
    demo.set_analog_output(IO_CABINET, 34, 1.5);
    Sleep(1000);//需要 window.h 延时 1s
    //查询 Ao 的状态
    demo.get_analog_output(IO_CABINET, 34, &AO35);
    printf("A035 = %f\n", AO35);
    return 0;
```

```
}
```

### 4.3.16 查询数字输入状态

```
/**
* @brief 查询数字输入(DI)状态
* @param type DI 类型
* @param index DI 索引 （从 0 开始）
* @param result DI 状态查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_digital_input(IOType type, int index, BOOL* result);
```

### 4.3.17 查询数字输出状态

```
/**
* @brief 查询数字输出(DO)状态
* @param type DO 类型
* @param index DO 索引 （从 0 开始）
* @param result DO 状态查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_digital_output(IOType type, int index, BOOL* result);
```

### 4.3.18 获取模拟量输入变量的值

```
/**
* @brief 获取模拟量输入变量(AI)的值
* @param type AI 的类型
* @param index AI 索引 （从 0 开始）
* @param result 指定 AI 状态查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_analog_input(IOType type, int index, float* result);
```

### 4.3.19 获取模拟量输出变量的值

```
/**
* @brief 获取模拟量输出变量(AO)的值
* @param type AO 的类型
* @param index AO 索引 （从 0 开始）
* @param result 指定 AO 状态查询结果
```

```
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_analog_output(IOType type, int index, float* result);
```

## 4.3.20 查询扩展 IO 是否运行

```
/**
* @brief 查询扩展 IO 模块是否运行
* @param is_running 扩展 IO 模块运行状态查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  is_extio_running(BOOL* is_running);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
//查询扩展 IO 状态
int main()
{
BOOL is_running;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    //查询 TIO 的状态
    demo.is_extio_running(&is_running);
    printf("tio = %d\n", is_running);
    return 0;
}
```

## 4.3.21 机械臂负载设置

```
/**
* @brief 机械臂负载设置
* @param payload 负载质心、质量数据
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_payload(const PayLoad* payload);
```

## 4.3.22 获取机械臂负载数据

```
/**
* @brief 获取机械臂负载数据
* @param payload 负载查询结果
```

```
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  errno_t  get_payload(PayLoad* payload);
```

代码示例：

```
1.  int example_payload()
2.  {
3.      //实例 API 对象 demo
4.      JAKAZuRobot demo;
5.      PayLoad payloadret;
6.      PayLoad payload_set;
7.      //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
8.      demo.login_in("192.168.2.194");
9.      //查询当前负载数据
10.     demo.get_payload(&payloadret);
11.     std::cout << " payload mass is :" << payloadret.mass << " kg" << std::endl;
12.     std::cout << " payload center of mass is \nx: " << payloadret.centroid.x<< "y: " << pay
    loadret.centroid.y << "z: " << payloadret.centroid.z << std::endl;
13.     payload_set.mass = 1.0;
14.     //单位 mm
15.     payload_set.centroid.x = 0; payload_set.centroid.y = 0; payload_set.centroid.z = 10;
16.     //设置当前负载数据
17.     demo.set_payload(&payload_set);
18.     //查询当前负载数据
19.     demo.get_payload(&payloadret);
20.     std::cout << " payload mass is :" << payloadret.mass << " kg" << std::endl;
21.     std::cout << " payload center of mass is \nx: " << payloadret.centroid.x << "y: " << pa
    yloadret.centroid.y << "z: " << payloadret.centroid.z << std::endl;
22.     return 0;
23. }
```

## 4.3.23 设置 TIOV3 电压参数

```
1.  /**
2.  * @brief 设置 tioV3 电压参数
3.  * @param vout_enable 电压使能，0:关，1 开
4.  * @param vout_vol 电压大小 0:24v 1:12v
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  errno_t set_tio_vout_param(int vout_enable, int vout_vol);
```

## 4.3.24 获取 TIOV3 电压参数

```
1.  /**
```

```
 * @brief 获取 tioV3 电压参数
 * @param vout_enable 电压使能，0:关，1 开
 * @param vout_vol 电压大小 0:24v 1:12v
 * @return ERR_SUCC 成功 其他失败
 */
errno_t get_tio_vout_param(int* vout_enable, int* vout_vol);
```

## 4.3.25 TIO 添加或修改信号量

```
/**
* @brief 添加或修改信号量
* @param sign_info 信号量参数
* @return ERR_SUCC 成功 其他失败
*/
errno_t add_tio_rs_signal(SignInfo sign_info);
```

代码示例：

```
void example_add_tio_rs_signal()
{
    JAKAZuRobot robot;
    robot.login_in("192.168.20.138");
    robot.power_on();
    robot.enable_robot();

    SignInfo sign_info {0};
    memcpy(sign_info.sig_name, "sign_tmp", sizeof("sign_tmp"));
    sign_info.chn_id = 0,
    sign_info.sig_type = 0,
    sign_info.value = 10,
    sign_info.frequency = 5;
    sign_info.sig_addr = 0x1;

    robot.add_tio_rs_signal(sign_info);

    robot.disable_robot();
    robot.power_off();
    robot.login_out();
}
```

## 4.3.26 TIO 删除信号量

```
/**
* @brief 删除信号量
```

```
* @param sig_name 信号量名称
* @return ERR_SUCC 成功 其他失败
*/
errno_t del_tio_rs_signal(const char* sig_name);
```

代码示例：

```
void example_del_tio_rs_signal()
{
    JAKAZuRobot robot;
    robot.login_in("192.168.20.138");
    robot.power_on();
    robot.enable_robot();

    const char* name = "sign_tmp";
    robot.del_tio_rs_signal(name);

    robot.disable_robot();
    robot.power_off();
    robot.login_out();
}
```

## 4.3.27 TIO RS485 发送指令

```
/**
* @brief RS485 发送命令
* @param chn_id 通道号
* @param data 数据字段
* @return ERR_SUCC 成功 其他失败
*/
errno_t send_tio_rs_command(int chn_id, uint8_t* data,int buffsize);
```

代码示例：

```
void example_send_tio_rs_command()
{
    JAKAZuRobot robot;
    robot.login_in("192.168.20.138");
    robot.power_on();
    robot.enable_robot();

    uint8_t cmd[] = {'t', 'e', 's', 't', 'c', 'm', 'd'};
    robot.send_tio_rs_command(0, cmd, sizeof(cmd));

    robot.disable_robot();
    robot.power_off();
    robot.login_out();
}
```

## 4.3.28 TIO 获取信号量信息

```
/**
* @brief 获取信号量信息
* @param SignInfo* 信号量信息数组
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_rs485_signal_info(SignInfo* sign_info);
```

代码示例：

```
void example_get_rs485_signal_info()
{
    JAKAZuRobot robot;
    robot.login_in("192.168.20.138");
    robot.power_on();
    robot.enable_robot();

    SignInfo sign_info[256];
    robot.get_rs485_signal_info(sign_info);

    robot.disable_robot();
    robot.power_off();
    robot.login_out();
}
```

## 4.3.29 TIO 设置 TIO 模式

```
/**
* @brief 设置 tio 模式
* @param pin_type tio 类型 0 for DI Pins, 1 for DO Pins, 2 for AI Pins
* @param pin_type tio 模式 DI Pins: 0:0x00 DI2 为 NPN,DI1 为 NPN,1:0x01 DI2 为 NPN,DI1 为 PNP, 2:0x10 DI2 为 PNP,DI1 为 NPN,3:0x11 DI2 为 PNP,DI1 为 PNP
                        DO Pins: 低 8 位数据高 4 位为 DO2 配置，低四位为 DO1 配置,0x0 DO 为 NPN 输出，0x1 DO 为 PNP 输出，0x2 DO 为推挽输出，0xF RS485H 接口
                        AI Pins: 0:模拟输入功能使能，RS485L 禁止，1:RS485L 接口使能，模拟输入功能禁止
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_tio_pin_mode(int pin_type, int pin_mode);
```

### 4.3.30 TIO 获取 TIO 模式

```
/**
* @brief 获取 tio 模式
* @param pin_type tio 类型 0 for DI Pins, 1 for DO Pins, 2 for AI Pins
* @param pin_type tio 模式 DI Pins: 0:0x00 DI2 为 NPN,DI1 为 NPN,1:0x01 DI2 为 NPN,DI1 为 PNP, 2:0x10 DI2 为 PNP,DI1 为 NPN,3:0x11 DI2 为 PNP,DI1 为 PNP
                        DO Pins: 低 8 位数据高 4 位为 DO2 配置，低四位为 DO1 配置,0x0 DO 为 NPN 输出，0x1 DO 为 PNP 输出，0x2 DO 为推挽输出，0xF RS485H 接口
                        AI Pins: 0:模拟输入功能使能，RS485L 禁止，1:RS485L 接口使能，模拟输入功能禁止
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_tio_pin_mode(int pin_type, int* pin_mode);
```

### 4.3.31 TIO RS485 通讯参数配置

```
/**
* @brief RS485 通讯参数配置
* @param ModRtuComm 当通道模式设置为 Modbus RTU 时，需额外指定 Modbus 从站节点 ID
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_rs485_chn_comm(ModRtuComm mod_rtu_com);
```

### 4.3.32 TIO RS485 通讯参数查询

```
/**
* @brief RS485 通讯参数查询
* @param ModRtuComm 查询时 chn_id 作为输入参数
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_rs485_chn_comm(ModRtuComm* mod_rtu_com);
```

### 4.3.33 TIO RS485 通讯模式配置

```
/**
* @brief RS485 通讯模式配置
* @param chn_id 0: RS485H, channel 1; 1: RS485L, channel 2
* @param chn_mode 0: Modbus RTU, 1: Raw RS485, 2, torque sensor
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_rs485_chn_mode(int chn_id, int chn_mode);
```

## 4.3.34 TIO RS485 通讯模式查询

```
/**
* @brief RS485 通讯模式查询
* @param chn_id 输入参数 0: RS485H, channel 1; 1: RS485L, channel 2
* @param chn_mode 输出参数  0: Modbus RTU, 1: Raw RS485, 2, torque sensor
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_rs485_chn_mode(int chn_id, int* chn_mode);
```

## 4.3.35 获取机器人安装角度

```
/**
* @brief 获取安装角度
* @param quat 安装角度四元数
* @param appang 安装角度 RPY 角
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_installation_angle(Quaternion* quat, Rpy* appang);
```

代码示例：

```
#include <JAKAZuRobot.h>
#include <iostream>

int main()
{
    JAKAZuRobot robot;
    errno_t ret = robot.login_in("192.168.137.152");
    if (ret != ERR_SUCC)
    {
        std::cerr << "login failed.\n";
        return -1;
    }

    ret = robot.set_installation_angle(3.14, 0);
    if (ret != ERR_SUCC)
    {
        std::cerr << "set installation angle failed.\n";
        return -1;
    }

    Quaternion quat;
    Rpy rpy;
    ret = robot.get_installation_angle(&quat, &rpy);
```

```
    if (ret != ERR_SUCC)
    {
        std::cerr << "get installation angle failed.\n";
        return -1;
    }

    std::cout << "quat: [" << quat.x << ", " << quat.y << ", " << quat.z << ", " << quat.s << "]\n";
    std::cout << "anglex: " << rpy.rx << ", anglez: " << rpy.rz << "\n";

    ret = robot.login_out();
    if (ret != ERR_SUCC)
    {
        std::cerr << "login out failed.\n";
        return -1;
    }

    return 0;
}
```

## 4.3.36 设置机器人安装角度

```
/**
* @brief 设置安装角度
* @param angleX 绕 X 轴旋转角度
* @param angleZ 绕 Z 轴旋转角度
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_installation_angle(double angleX, double angleZ);
```

代码示例：

```
#include <JAKAZuRobot.h>
#include <iostream>

int main()
{
    JAKAZuRobot robot;
    errno_t ret = robot.login_in("192.168.137.152");
    if (ret != ERR_SUCC)
    {
        std::cerr << "login failed.\n";
        return -1;
    }

    ret = robot.set_installation_angle(3.14, 0);
```

```
    if (ret != ERR_SUCC)
    {
        std::cerr << "set installation angle failed.\n";
        return -1;
    }

    Quaternion quat;
    Rpy rpy;
    ret = robot.get_installation_angle(&quat, &rpy);
    if (ret != ERR_SUCC)
    {
        std::cerr << "get installation angle failed.\n";
        return -1;
    }

    std::cout << "quat: [" << quat.x << ", " << quat.y << ", " << quat.z << ", " << quat.s << "]\n";
    std::cout << "anglex: " << rpy.rx << ", anglez: " << rpy.rz << "\n";

    ret = robot.login_out();
    if (ret != ERR_SUCC)
    {
        std::cerr << "login out failed.\n";
        return -1;
    }

    return 0;
}
```

## 4.3.37 设置系统变量

```
/**
 *@brief set user var
 */
errno_t set_user_var(UserVariable v);
```

## 4.3.38 获取机械臂状态

```
/**
* @brief 获取机械臂状态
* @param state 机械臂状态查询结果
* @return ERR_SUCC 成功 其他失败
```

```
 */
errno_t  get_robot_state(RobotState* state);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//获取机械臂状态(急停 上电 伺服使能)
int example_get_robstate()
{
    JAKAZuRobot demo;
    //声明机械臂状态结构体
    RobotState state;
    demo.login_in("192.168.2.152");
    demo.power_on();
    demo.enable_robot();
    //查询机械臂状态
demo.get_robot_state(&state);
    std::cout << "is e_stoped : " << state.estoped << std::endl;
    std::cout << "is powered : " << state.poweredOn << std::endl;
    std::cout << "is servoEnabled : " << state.servoEnabled << std::endl;
    return 0;
}
```

### 4.3.39 获取系统变量

```
/**
 *@brief get user_var list
 */
errno_t get_user_var(UserVariableList* vlist);
```

## 4.4 机械臂安全状态设置与查询

### 4.4.1 查询机械臂是否超出限位

```
/**
* @brief 查询机械臂是否超出软限位
* @param on_limit 查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  is_on_limit(BOOL* on_limit);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//查询是否超出限位
int example_is_on_limit()
{
    JAKAZuRobot demo;
    BOOL on_limit;
    demo.login_in("192.168.2.152");
    demo.power_on();
    demo.enable_robot();
    while (1)
    {
        //查询是否超限
        demo.is_on_limit(&on_limit);
        std::cout << " on_limit is :" << on_limit << std::endl;
        Sleep(200);
    }
    return 0;
}
```

### 4.4.2 查询机械臂是否处于碰撞保护模式

```
/**
* @brief 查询机械臂是否处于碰撞保护模式
* @param in_collision 查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  is_in_collision(BOOL* in_collision);
```

### 4.4.3 碰撞之后从碰撞保护模式恢复

```
/**
* @brief 碰撞之后从碰撞保护模式恢复
* @return ERR_SUCC 成功 其他失败
*/
errno_t  collision_recover();
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
```

```
#include <windows.h>

//碰撞保护状态查询，恢复
int example_collision_recover()
{
    JAKAZuRobot demo;
    BOOL in_collision;
    demo.login_in("192.168.2.152");
    demo.power_on();
    demo.enable_robot();
    //查询是否处于碰撞保护状态
    demo.is_in_collision(&in_collision);
    std::cout << " in_collision is :" << in_collision << std::endl;
    if (in_collision)
        //如果处于碰撞保护模式，则从碰撞保护中恢复
        {demo.collision_recover();}
    else
        {std::cout << "robot is not collision" << std::endl;}
    return 0;
}
```

### 4.4.4 设置机械臂碰撞等级

```
/**
* @brief 设置机械臂碰撞等级
* @param level  碰撞等级，取值范围[0,5]，其中 0 为关闭碰撞，1 为碰撞阈值 25N，2 为碰撞阈值 50N，3 为碰撞阈值 75N，4 为碰撞阈值 100N，5 为碰撞阈值 125N
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_collision_level(const int level);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//碰撞等级查看，设置
int example_collision_level()
{
    //实例 API 对象 demo
    JAKAZuRobot demo;
    int level;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.152");
    //机械臂上电
```

```
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //查询当前碰撞等级
    demo.get_collision_level(&level);
    std::cout << " collision level is :" << level << std::endl;
    //设置碰撞等级，[0,5]，0 为关闭碰撞，1 为碰撞阈值 25N，2 为碰撞阈值 50N，3 为碰撞阈值 75N，4 为碰撞阈值 100N，5 为碰撞阈值 125N，
    demo.set_collision_level(2);
    //查询当前碰撞等级
    demo.get_collision_level(&level);
    std::cout << " collision level is :" << level << std::endl;
    return 0;
}
```

### 4.4.5 获取机器设置的碰撞等级

```
/**
* @brief 获取机械臂设置的碰撞等级
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_collision_level(int* level);
```

### 4.4.6 设置网络异常时机械臂自动终止运动类型

```
/**
* @brief 设置网络异常控制句柄，SDK 与机械臂控制器失去连接后多长时间机械臂控制器终止机械臂当前运动
* @param millisecond 时间参数，单位：ms
* @param mnt 网络异常时机械臂需要进行的动作类型
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_network_exception_handle(float millisecond, ProcessType mnt);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//设置网络异常时机械臂自动终止运动类型
int example_set_network_exception_handle()
{
    float milisec = 100;
    int ret;
```

```
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    //设置柔顺力矩条件
    ret = demo.set_network_exception_handle(milisec, MOT_KEEP);
    std::cout << ret << std::endl;
    return 0;
}
```

### 4.4.7 获取机械臂目前发生的最后一个错误码

```
/**
* @brief 获取机械臂运行过程中最后一个错误码,当调用 clear_error 时，最后一个错误码会清零
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_last_error(ErrorCode* code);
```

### 4.4.8 注册机械臂出错时的回调函数

```
/**
* @brief 注册机械臂出现错误时的回调函数
* @param func 指向用户定义的函数的函数指针
* @param error_code 机械臂的错误码
*/
errno_t set_error_handler(CallBackFuncType func);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//注册机械臂出错函数 错误处理，类中断
void user_error_handle(int error_code)
{
    std::cout << error_code << std::endl;
}
int example_set_err_handle()
{
    JAKAZuRobot demo;
    demo.login_in("192.168.2.229");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
```

```
    demo.enable_robot();
    //设置用户异常回调函数
    demo.set_error_handler(user_error_handle);
    return 0;
}
```

## 4.4.9 设置机械臂错误码文件存放路径

```
/**
* @brief 设置错误码文件路径，需要使用 get_last_error 接口时需要设置错误码文件路径，如果不使用 get_last_error 接口，则不需要设置该接口
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_errorcode_file_path(char* path);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//错误码查看
int example_get_last_errcode()
{
    int ret;
//初始化错误码文件存放路径
    char path[100] = "E:\\JAKA_ERROR_CODE.csv";
    JAKAZuRobot demo;
    ErrorCode Eret;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    ret = demo.program_load("not_exist999875");//故意加载一个不存在的程序，引发报错。
    std::cout << ret << std::endl;
    demo.get_last_error(&Eret);//查询最后一个报错信息
    std::cout << " error code is :" << Eret.code << " message: "<< Eret.message<< std::endl;

    demo.set_errorcode_file_path(path);//设置错误码说明文件
    demo.get_last_error(&Eret);//查询最后一个报错信息
    std::cout << " error code is :" << Eret.code << " message: " << Eret.message << std::endl;
    return 0;
}
```

# 4.5 使用 APP 脚本程序

## 4.5.1 运行当前加载的作业程序

```
/**
* @brief 运行当前加载的作业程序
* @return ERR_SUCC 成功 其他失败
*/
errno_t  program_run();
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//脚本加载，脚本运行控制，脚本过程查看
int example_program()
{
    char name[128];
    int cur_line;
    JAKAZuRobot demo;
    ProgramState pstatus;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    //加载预先通过 app 编辑完成的 example 脚本
    demo.program_load("example");
    //获取已加载的程序名
    demo.get_loaded_program(name);
    std::cout <<"Pro_name is :"<< name << std::endl;
    //运行当前加载的程序
    demo.program_run();
    Sleep(1000);//让程序先运行 1s
    //暂停当前运行的程序
    demo.program_pause();
    //获取当前执行程序的行号
    demo.get_current_line(&cur_line);
    std::cout << "cur_line is :" << cur_line << std::endl;
    //获取当前程序状态
    demo.get_program_state(&pstatus);
    std::cout <<  "pro_status is : " << pstatus << std::endl;
    //继续运行当前程序
    demo.program_resume();
    Sleep(10000);//需要 window.h 延时 10s
```

```
    //终止当前程序
    demo.program_abort();
    return 0;
}
```

### 4.5.2 暂停当前运行的作业程序

```
/**
* @brief 暂停当前运行的作业程序
* @return ERR_SUCC 成功 其他失败
*/
errno_t  program_pause();
```

### 4.5.3 继续运行当前暂停的作业程序

```
/**
* @brief 继续运行当前暂停的作业程序
* @return ERR_SUCC 成功 其他失败
*/
errno_t  program_resume();
```

### 4.5.4 终止当前执行的作业程序

```
/**
* @brief 终止当前执行的作业程序
* @return ERR_SUCC 成功 其他失败
*/
errno_t  program_abort();
```

### 4.5.5 加载指定的作业程序

```
/**
* @brief 加载指定的作业程序 程序名为 app 中的名称即可（加载轨迹复现数据，轨迹复现数据的加载需要在文件夹名字前加上 track/）
* @param file 程序文件路径
* @return ERR_SUCC 成功 其他失败
*/
errno_t  program_load(const char* file);
```

### 4.5.6 获取已加载的作业程序的名字

```
/**
* @brief 获取已加载的作业程序名字
* @param file 程序文件路径
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_loaded_program(char* file);
```

### 4.5.7 获取当前机械臂作业程序的执行行号

```
/**
* @brief 获取当前机械臂作业程序的执行行号
* @param curr_line 当前行号查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_current_line(int* curr_line);
```

### 4.5.8 获取机械臂作业程序的执行状态

```
/**
* @brief 获取机械臂作业程序执行状态
* @param status 作业程序执行状态查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_program_state(ProgramState* status);
```

### 4.5.9 设置机械臂的运行倍率

```
/**
* @brief 设置机械臂运行倍率
* @param rapid_rate 是程序运行倍率，设置范围为[0,1]
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_rapidrate(double rapid_rate);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//机械臂速度查看及调整
int example_rapidrate()
```

```
{
    double rapid_rate;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.152");
    demo.power_on();
    demo.enable_robot();
    //获取机械臂运动速率
    demo.get_rapidrate(&rapid_rate);
    std::cout << "rapid_rate is : " << rapid_rate << std::endl;
    //设置机械臂运动速率
    demo.set_rapidrate(0.4);
    Sleep(100);
    demo.get_rapidrate(&rapid_rate);
    std::cout << "rapid_rate is : " << rapid_rate << std::endl;
    return 0;
}
```

### 4.5.10 获取机械臂的运行倍率

```
/**
* @brief 获取机械臂运行倍率
* @param rapid_rate 当前控制系统倍率
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_rapidrate(double* rapid_rate);
```

## 4.6 机械臂轨迹复现

### 4.6.1 设置轨迹复现配置参数

```
/**
* @brief 设置轨迹复现配置参数
* @param para 轨迹复现配置参数
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_traj_config(const TrajTrackPara* para);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//轨迹复现参数查看及设置
```

```
int example_traj_config()
{
    JAKAZuRobot demo;
    TrajTrackPara trajpar_read;
    TrajTrackPara trajpar_set;
    demo.login_in("192.168.2.194");
    //查询当前轨迹复现参数
    demo.get_traj_config(&trajpar_read);
    std::cout << " trajTrackPara is :\n xyz interval:" << trajpar_read.xyz_interval << "  rpy interval is :" << trajpar_read.rpy_interval << std::endl;
    std::cout << " vel: " << trajpar_read.vel << "  acc: " << trajpar_read.acc << std::endl;
    //设置当前轨迹复现参数
    trajpar_set.xyz_interval = 0.01; trajpar_set.rpy_interval = 0.01; trajpar_set.vel = 10; trajpar_set.acc = 2;
    demo.set_traj_config(&trajpar_set);
    //查询当前轨迹复现参数
    demo.get_traj_config(&trajpar_read);
    std::cout << " trajTrackPara is :\n xyz interval:" << trajpar_read.xyz_interval << "  rpy interval is :" << trajpar_read.rpy_interval << std::endl;
    std::cout << " vel: " << trajpar_read.vel << "  acc: " << trajpar_read.acc << std::endl;
    return 0;
}
```

### 4.6.2 获取轨迹复现配置参数

```
/**
* @brief 获取轨迹复现配置参数
* @param para 轨迹复现配置参数
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_traj_config(TrajTrackPara* para);
```

### 4.6.3 采集轨迹复现数据控制开关

```
/**
* @brief 采集轨迹复现数据控制开关，该开关开启后需要启动 拖拽 才有轨迹文件生成。
* @param mode 选择 TRUE 时，开始数据采集，选择 FALSE 时，结束数据采集
* @param filename 采集数据的存储文件名，当 filename 为空指针时，存储文件以当前日期命名
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_traj_sample_mode(const BOOL mode,char* filename);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//轨迹采集开关与状态查询
int example_traj_sample()
{
    BOOL samp_stu;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    char name[20] = "testxx";
    //开启轨迹复现数据采集开关
    demo.set_traj_sample_mode(TRUE, name);
    //查询轨迹复现采集状态
    demo.get_traj_sample_status(&samp_stu);
    Sleep(10000);
    demo.set_traj_sample_mode(FALSE, name);
    return 0;
}
```

### 4.6.4 采集轨迹复现数据状态查询

```
/**
* @brief 采集轨迹复现数据状态查询
* @param mode 为 TRUE 时，数据正在采集，为 FALSE 时，数据采集结束，在数据采集状态时不允许再次开启数据采集开关
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_traj_sample_status(BOOL* sample_status);
```

### 4.6.5 查询控制器中已经存在的轨迹复现数据的文件名

```
/**
* @brief 查询控制器中已经存在的轨迹复现数据的文件名
* @param filename 控制器中已经存在的轨迹复现数据的文件名
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_exist_traj_file_name(MultStrStorType* filename);
```

代码示例：

```
#include <iostream>
```

```
#include "JAKAZuRobot.h"
#include <windows.h>

//查询控制器中已经存在的轨迹复现数据的文件名
int example_get_traj_existed_filename()
{
    JAKAZuRobot demo;
    MultStrStorType traj_file;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //查询当前轨迹文件名。
    demo.get_exist_traj_file_name(&traj_file);
    std::cout << "file nums :" << traj_file.len << std::endl;
    for(int i=0; i<traj_file.len;i++)
        std::cout << traj_file.name[i] << std::endl;
    return 0;
}
```

## 4.6.6 重命名轨迹复现数据的文件名

```
/**
* @brief 重命名轨迹复现数据的文件名
* @param src 原文件名
* @param dest 目标文件名，文件名长度不能超过 100 个字符，文件名不能为空，目标文件名不支持中文
* @return ERR_SUCC 成功 其他失败
*/
errno_t  rename_traj_file_name(const char* src,const char* dest);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//重命名轨迹复现的数据文件名
int example_rename_traj_file_name()
{
    JAKAZuRobot demo;
    MultStrStorType traj_file;
    char name_new[20] = "555";
    demo.login_in("192.168.2.194");
    //查询当前轨迹文件名。
    demo.get_exist_traj_file_name(&traj_file);
    std::cout << "file nums :" << traj_file.len << std::endl;
    for (int i = 0; i < traj_file.len; i++)
        std::cout << traj_file.name[i] << std::endl;
```

```
    //重命名轨迹复现的数据文件名
    demo.rename_traj_file_name(traj_file.name[0], name_new);
    //查询当前轨迹文件名。
    demo.get_exist_traj_file_name(&traj_file);
    std::cout << "file nums :" << traj_file.len << std::endl;
    for (int i = 0; i < traj_file.len; i++)
        std::cout << traj_file.name[i] << std::endl;
    return 0;
}
```

### 4.6.7 删除控制器中轨迹复现数据文件

```
/**
* @brief 删除控制器中轨迹复现数据文件
* @param filename 要删除的文件的文件名，文件名为数据文件名字
* @return ERR_SUCC 成功 其他失败
*/
errno_t  remove_traj_file(const char* filename);
```

### 4.6.8 控制器中轨迹复现数据文件生成控制器执行脚本

```
/**
* @brief 控制器中轨迹复现数据文件生成控制器执行脚本
* @param filename 数据文件的文件名，文件名为数据文件名字，不带后缀
* @return ERR_SUCC 成功 其他失败
*/
errno_t  generate_traj_exe_file(const char* filename);
```

## 4.7 机械臂伺服运动模式

### 4.7.1 机械臂伺服位置控制模式使能

```
/**
* @brief 机械臂伺服位置控制模式使能。
* @param enable  TRUE 为进入伺服位置控制模式，FALSE 表示退出该模式
* @return ERR_SUCC 成功 其他失败
*/
 Ps:v19 及之前的版本该接口为非阻塞指令，V20 之后该指令变为为阻塞指令。
errno_t  servo_move_enable(BOOL enable);
```

## 4.7.2 机械臂关节空间伺服模式运动

```
/**
* @brief 机械臂关节空间位置控制模式
* @param joint_pos 机械臂关节运动目标位置
* @param move_mode 指定运动模式：增量运动、绝对运动和连续运动
* @return ERR_SUCC 成功 其他失败
*/
errno_t  servo_j(const JointValue* joint_pos, MoveMode move_mode);
```

代码示例：

```
//机械臂伺服关节运动
//使用该接口时需要先调用 servo_move_enable(TRUE)开启伺服控制
//控制器的发送周期为 8ms，建议用户发送周期也为 8ms，网络环境较差的情况可以适当减小周期
//关节速度上限 180deg/s
//本指令与 joint_move 有较大差别，joint_move 的插补由控制器进行，servo_j 需要预先进行轨迹规划。
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

int main()
{
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.152");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //TRUE 为进入伺服模式
    demo.servo_move_enable(TRUE);
    //定义并初始化关 JointValue 变量
    JointValue joint_pos = {-0.001, 0, 0, 0, 0, -0.001};
    for (int i = 0; i < 100; i++)
    {
        //关节空间伺服运动，其中 INCR 代表增量运动
        demo.servo_j(&joint_pos, INCR);
    }
    //FALSE 为退出伺服模式
    demo.servo_move_enable(FALSE);
    return 0;
}
```

### 4.7.3 机械臂关节空间伺服模式运动扩展

```
/**
* @brief 机械臂关节空间位置控制模式，增加了周期的调节性可以将周期调整为 8ms 的倍数
* @param joint_pos 机械臂关节运动目标位置
* @move_mode 指定运动模式：增量运动、绝对运动
* @step_num  倍分周期，servo_j 运动周期为 step_num*8ms，其中 step_num>=1
* @return ERR_SUCC 成功 其他失败
*/
errno_t  servo_j(const JointValue* joint_pos, MoveMode move_mode, unsigned int step_num);
```

### 4.7.4 机械臂笛卡尔空间伺服模式运动

```
/**
* @brief 机械臂笛卡尔空间位置控制模式
* @param cartesian_pose 机械臂笛卡尔空间运动目标位置
* @param move_mode 指定运动模式：增量运动、绝对运动和连续运动
* @return ERR_SUCC 成功 其他失败
*/
errno_t  servo_p(const CartesianPose* cartesian_pose, MoveMode move_mode);
```

代码示例：

```
//机械臂伺服笛卡尔空间运动
//使用该接口时需要先调用 servo_move_enable(TRUE)开启伺服控制
//控制器的发送周期为 8ms，建议用户发送周期也为 8ms，网络环境较差的情况可以适当减小周期
//关节速度上限 180 deg/s，笛卡尔空间没有相对直观的限制，但应满足该关节速度限制
//本指令与 linear_move 有较大差别，linear_move 的插补由控制器进行，servo_p 需要预先进行轨迹规划。
#include <iostream>
#include "JAKAZuRobot.h"

int main()//机械臂伺服笛卡尔空间运动
{
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.152");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //TRUE 为进入伺服模式
    demo.servo_move_enable(TRUE);
```

```
    //定义并初始化关 CartesianPose 变量
    CartesianPose cart;
    cart.tran.x = 0; cart.tran.y = 1; cart.tran.z = 0;
    cart.rpy.rx = 0; cart.rpy.ry = 0; cart.rpy.rz = 0;
    for (int i = 0; i < 100; i++)
    {
        //笛卡尔空间伺服运动，其中 INCR 代表增量运动
        demo.servo_p(&cart, INCR);
    }
    //FALSE 为退出伺服模式
    demo.servo_move_enable(FALSE);
    return 0;
}
```

## 4.7.5 机械臂笛卡尔空间伺服模式运动扩展

```
/**
* @brief 机械臂笛卡尔空间位置控制模式，增加了周期的调节性可以将周期调整为 8ms 的倍数
* @param cartesian_pose 机械臂笛卡尔空间运动目标位置
* @move_mode 指定运动模式：增量运动、绝对运动
* @step_num  倍分周期，servo_p 运动周期为 step_num*8ms，其中 step_num>=1
* @return ERR_SUCC 成功 其他失败
*/
errno_t  servo_p(const CartesianPose* cartesian_pose, MoveMode move_mode, unsigned int step_num);
```

## 4.7.6 机械臂 SERVO 模式下禁用滤波器

```
/**
* @brief SERVO 模式下不使用滤波器,该指令在 SERVO 模式下不可设置，退出 SERVO 后可设置
* @return ERR_SUCC 成功 其他失败
*/
errno_t  servo_move_use_none_filter();
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
//servo 模式禁用滤波器
int example_servo_use_none_filter()
{
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
```

```
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    ret = demo.servo_move_use_none_filter();
    std::cout << ret << std::endl;
    return 0;
}
```

## 4.7.7 机械臂 SERVO 模式下关节空间一阶低通滤波

```
/**
* @brief SERVO 模式下切换到关节空间一阶低通滤波,该指令在 SERVO 模式下不可设置，退出 SERVO 后可设置
* @param cutoffFreq 一阶低通滤波器截止频率
* @return ERR_SUCC 成功 其他失败
*/
errno_t  servo_move_use_joint_LPF(double cutoffFreq);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//servo 模式下关节空间一阶低通滤波
int example_servo_use_joint_LPF()
{
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //servo 模式下关节空间一阶低通滤波,截止频率 0.5Hz
    ret = demo.servo_move_use_joint_LPF(0.5);
    std::cout << ret << std::endl;
    return 0;
}
```

## 4.7.8 机械臂 SERVO 模式下关节空间非线性滤波

```
/**
* @brief SERVO 模式下切换到关节空间非线性滤波,该指令在 SERVO 模式下不可设置，退出 SERVO 后可设置
* @param max_vr 笛卡尔空间姿态变化速度的速度上限值（绝对值）°/s
* @param max_ar 笛卡尔空间姿态变化速度的加速度上限值（绝对值）°/s^2
* @param max_jr 笛卡尔空间姿态变化速度的加加速度上限值（绝对值）°/s^3
* @return ERR_SUCC 成功 其他失败
*/
errno_t  servo_move_use_joint_NLF(double max_vr, double max_ar, double max_jr);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//servo 模式下关节空间非线性滤波
int example_servo_use_joint_NLF()
{
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //servo 模式下关节空间非线性滤波
    ret = demo.servo_move_use_joint_NLF(2,2,4);
    std::cout << ret << std::endl;
    return 0;
}
```

## 4.7.9 机械臂 SERVO 模式下笛卡尔空间非线性滤波

```
/**
* @brief SERVO 模式下切换到笛卡尔空间非线性滤波,该指令在 SERVO 模式下不可设置，退出 SERVO 后可设置
* @param max_vp 笛卡尔空间下移动指令速度的上限值（绝对值）。单位：mm/s
* @param max_ap 笛卡尔空间下移动指令加速度的上限值（绝对值）。单位：mm/s^2
* @param max_jp 笛卡尔空间下移动指令加加速度的上限值（绝对值）单位：mm/s^3
* @param max_vr 笛卡尔空间姿态变化速度的速度上限值（绝对值）°/s
* @param max_ar 笛卡尔空间姿态变化速度的加速度上限值（绝对值）°/s^2
* @param max_jr 笛卡尔空间姿态变化速度的加加速度上限值（绝对值）°/s^3
```

```
 * @return ERR_SUCC 成功 其他失败
*/
errno_t  servo_move_use_carte_NLF(double max_vp, double max_ap, double max_jp, double max_vr, double max_ar, double max_jr);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//servo 模式下笛卡尔空间非线性滤波
int example_servo_use_carte_NLF()
{
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //servo 模式下笛卡尔空间非线性滤波
    ret = demo.servo_move_use_carte_NLF(2, 2, 4, 2, 2, 4);
    std::cout << ret << std::endl;
    return 0;
}
```

## 4.7.10 机械臂 SERVO 模式下关节空间多阶均值滤波

```
/**
 * @brief SERVO 模式下切换到关节空间多阶均值滤波器,该指令在 SERVO 模式下不可设置，退出 SERVO 后可设置
 * @param max_buf 均值滤波器缓冲区的大小
 * @param kp 加速度滤波系数
 * @param kv 速度滤波系数
 * @param ka 位置滤波系数
 * @return ERR_SUCC 成功 其他失败
 */
errno_t  servo_move_use_joint_MMF(int max_buf, double kp, double kv, double ka);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
```

```
//servo 模式下关节空间多阶均值滤波
int example_servo_use_joint_MMF()
{
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //servo 模式下关节空间多阶均值滤波
    ret = demo.servo_move_use_joint_MMF(20, 0.2, 0.4, 0.2);
    std::cout << ret << std::endl;
    return 0;
}
```

## 4.7.11 机械臂 SERVO 模式下速度前瞻参数设置

```
/**
* @brief SERVO 模式下关节空间多阶均值滤波器,该指令在 SERVO 模式下不可设置，退出 SERVO 后可设置
* @param max_buf 均值滤波器缓冲区的大小
* @param kp 加速度滤波系数
* @param kv 速度滤波系数
* @param ka 位置滤波系数
* @return ERR_SUCC 成功 其他失败
*/
errno_t servo_speed_foresight(int max_buf, double kp);
*/
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//servo 模式下速度前瞻参数设置
int example_speed_foresight()
{
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
```

```
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //servo 模式下关节空间多阶均值滤波
    ret = demo.servo_speed_foresight(200, 2);
    std::cout << ret << std::endl;
    return 0;
}
```

# 4.8 机械臂运动学

## 4.8.1 机械臂求解逆解

```
/**
* @brief 计算指定位姿在当前工具、当前安装角度以及当前用户坐标系设置下的逆解
* @param ref_pos 逆解计算用的参考关节空间位置
* @param cartesian_pose 笛卡尔空间位姿值
* @param joint_pos 计算成功时关节空间位置计算结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  kine_inverse(const JointValue* ref_pos, const CartesianPose* cartesian_pose, JointValue* joint_pos);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

// 机械臂逆解 已知 tcp_pos,求 joint_pos
int example_kine_inverse()
{
    int ret;
    JAKAZuRobot demo;
    //初始化参考点
    JointValue ref_jpos = { 0.558, 0.872, 0.872 , 0.349, 0.191, 0.191 };
    //初始化笛卡尔空间点坐标
    CartesianPose tcp_pos;
    tcp_pos.tran.x = 243.568; tcp_pos.tran.y = 164.064; tcp_pos.tran.z = 742.002;
    tcp_pos.rpy.rx = -1.81826; tcp_pos.rpy.ry = -0.834253;  tcp_pos.rpy.rz = -2.30243;
    //初始化返回值
    JointValue joint_pos = { 0,0,0,0,0,0 }; ;
    demo.login_in("192.168.2.194");
//求逆解
    ret = demo.kine_inverse(&ref_jpos, &tcp_pos, &joint_pos);
```

```
    std::cout << ret << std::endl;
    for (int i = 0; i < 6; i++)
    {
        std::cout << "joint [" << i + 1 << "] is :" << joint_pos.jVal[i] << std::endl;
    }
    return 0;
}
```

## 4.8.2 机械臂求解正解

```
/**
* @brief 计算指定关节位置在当前工具、当前安装角度以及当前用户坐标系设置下的位姿值
* @param joint_pos 关节空间位置
* @param cartesian_pose 笛卡尔空间位姿计算结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  kine_forward(const JointValue* joint_pos, CartesianPose* cartesian_pose);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

// 4.55 机械臂正解 已知 joint_pos,求 tcp_pos
int example_kine_forward()
{
    int ret;
    JAKAZuRobot demo;
//初始化返回值
    CartesianPose tcp_pos;
    demo.login_in("192.168.2.194");
//初始化关节矩阵
    JointValue joint_pos = { 0.558, 0.872, 0.872 , 0.349, 0.191, 0.191 };
//求正解
    ret = demo.kine_forward(&joint_pos, &tcp_pos);
    std::cout << ret << std::endl;
    std::cout << "tcp_pos is :\n x: " << tcp_pos.tran.x << "  y: " << tcp_pos.tran.y << "  z: " << tcp_pos.tran.z << std::endl;
    std::cout << "rx: " << tcp_pos.rpy.rx << "  ry: " << tcp_pos.rpy.ry << "  rz: " << tcp_pos.rpy.rz << std::endl;
    return 0;
}
```

### 4.8.3 欧拉角到旋转矩阵的转换

```
/**
* @brief 欧拉角到旋转矩阵的转换
* @param rpy 待转换的欧拉角数据
* @param rot_matrix 转换后的旋转矩阵
* @return ERR_SUCC 成功 其他失败
*/
errno_t  rpy_to_rot_matrix(const Rpy* rpy, RotMatrix* rot_matrix);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

// 4.56 欧拉角到旋转矩阵
int example_rpy_to_rot_matrix()
{
    int ret;
    JAKAZuRobot demo;
    //初始化旋转矩阵
    Rpy rpy;
    rpy.rx = -1.81826; rpy.ry = -0.834253;  rpy.rz = -2.30243;
//初始化返回值
    RotMatrix rot_matrix;
    demo.login_in("192.168.2.194");
//欧拉角变换成旋转矩阵
    ret = demo.rpy_to_rot_matrix(&rpy, &rot_matrix);
    std::cout << ret << "     eul2rotm" << std::endl;
    printf("%f %f %f\n", rot_matrix.x.x, rot_matrix.y.x, rot_matrix.z.x);
    printf("%f %f %f\n", rot_matrix.x.y, rot_matrix.y.y, rot_matrix.z.y);
    printf("%f %f %f\n", rot_matrix.x.z, rot_matrix.y.z, rot_matrix.z.z);
    return 0;
}
```

### 4.8.4 旋转矩阵到欧拉角的转换

```
/**
* @brief 旋转矩阵到欧拉角的转换
* @param rot_matrix 待转换的旋转矩阵数据
* @param rpy 转换后的 RPY 欧拉角结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  rot_matrix_to_rpy(const RotMatrix* rot_matrix, Rpy* rpy);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

// 4.57 旋转矩阵--->欧拉角
int example_rot_matrix_to_rpy()
{
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //初始化欧拉角
    Rpy rpy;
    //初始化旋转矩阵
    RotMatrix rot_matrix;
    rot_matrix.x.x = -0.4488, rot_matrix.y.x = -0.4998, rot_matrix.z.x = 0.7408;
    rot_matrix.x.y = -0.6621, rot_matrix.y.y = -0.3708, rot_matrix.z.y = -0.6513;
    rot_matrix.x.z = 0.6002, rot_matrix.y.z = -0.7828, rot_matrix.z.z = -0.1645;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    ret = demo.rot_matrix_to_rpy(&rot_matrix, &rpy);
    std::cout << ret << "     rotm2eul:" << std::endl;
    printf("%f %f %f \n", rpy.rx, rpy.ry, rpy.rz);
    return 0;
}
```

## 4.8.5 四元数到旋转矩阵的转换

```
/**
* @brief 四元数到旋转矩阵的转换
* @param quaternion 待转换的四元数数据
* @param rot_matrix 转换后的旋转矩阵结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t quaternion_to_rot_matrix(const Quaternion* quaternion, RotMatrix* rot_matrix);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

// 4.58 四元数-->旋转矩阵
int example_quaternion_to_rot_matrix()
{
    int ret;
```

```
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //初始化四元数
    Quaternion quat;
    quat.s = 0.0629;  quat.x = 0.522886; quat.y = -0.5592;  quat.z = 0.6453;
    //初始化旋转矩阵
    RotMatrix rot_matrix;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    ret = demo.quaternion_to_rot_matrix(&quat, &rot_matrix);
    std::cout << ret << "     quatl2rotm:" << std::endl;
    printf("%f %f %f\n", rot_matrix.x.x, rot_matrix.y.x, rot_matrix.z.x);
    printf("%f %f %f\n", rot_matrix.x.y, rot_matrix.y.y, rot_matrix.z.y);
    printf("%f %f %f\n", rot_matrix.x.z, rot_matrix.y.z, rot_matrix.z.z);
    return 0;
}
```

### 4.8.6 获取当前连接机械臂的 DH 参数

```
/**
* @brief 获取机器人 DH 参数
* @param dhParam DH 参数
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_dh_param(const JKHD *handle, DHParam *dhParam);
```

### 4.8.7 旋转矩阵到四元数的转换

```
/**
* @brief 旋转矩阵到四元数的转换
* @param rot_matrix 待转换的旋转矩阵
* @param quaternion 转换后的四元数结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  rot_matrix_to_quaternion(const RotMatrix* rot_matrix, Quaternion* quaternion);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

// 4.59 旋转矩阵--->四元数
int example_rot_matrix_to_quaternion()
{
```

```
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //初始化四元数
    Quaternion quat;
    //初始化旋转矩阵
    RotMatrix rot_matrix;
    rot_matrix.x.x = -0.4488774, rot_matrix.y.x = -0.499824, rot_matrix.z.x = 0.740795;
    rot_matrix.x.y = -0.662098, rot_matrix.y.y = -0.370777, rot_matrix.z.y = -0.651268;
    rot_matrix.x.z = 0.600190, rot_matrix.y.z = -0.782751, rot_matrix.z.z = -0.164538;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    ret = demo.rot_matrix_to_quaternion(&rot_matrix, &quat);
    std::cout << ret << "    rotm2quat:" << std::endl;
    printf("%lf %lf %lf %lf \n", quat.s, quat.x, quat.y, quat.z);
    return 0;
}
```

# 4.9 机器人力控接口

JAKA 机器人提供了一套基于力矩传感器的力控接口，用户可以基于这些接口完成高级的力控功能如导纳控制、柔顺控制等，进而实现一些较复杂的应用场景，如笛卡尔空间指定方向拖拽、力控装配等。但需要注意的是，这些接口依赖于额外配置的工具末端力传感器。

## 4.9.1 设置传感器品牌

```
/**
 * @brief 设置传感器品牌
 * @param sensor_brand 传感器品牌，1~6 对应不同传感器，具体咨询工程师
 * @return ERR_SUCC 成功 其他失败
 */
errno_t set_torsenosr_brand(int sensor_brand);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//设置传感器品牌
int example_set_torsensor_brand()
{
    int ret;
```

```
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    //设置传感器品牌
    ret = demo.set_torsenosr_brand(2);
    std::cout << ret << std::endl;
    return 0;
}
```

### 4.9.2 获取传感器品牌

```
/**
* @brief 获取传感器品牌
* @param sensor_brand 传感器品牌,
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_torsenosr_brand(int* sensor_brand);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//获取传感器品牌
int example_get_torsensor_brand()
{
    int ret,cur_sensor;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    //获取传感器品牌
    ret = demo.get_torsenosr_brand(&cur_sensor);
    std::cout << ret << std::endl;
    return 0;
}
```

### 4.9.3 开启或关闭力矩传感器

```
/**
* @brief 开启或关闭力矩传感器，需要先开启伺服模式
* @param sensor_mode 0 代表关闭传感器，1 代表开启力矩传感器
* @return ERR_SUCC 成功 其他失败
```

```
*/
errno_t set_torque_sensor_mode(int sensor_mode);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//开启或关闭力矩传感器
int example_set_torque_sensor_mode()
{
    int ret;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
demo.servo_move_enable(TRUE);
Sleep(200);
    //设置力矩传感器状态，1 打开，0 关闭
    ret = demo.set_torque_sensor_mode(1);
    std::cout << ret << std::endl;
    return 0;
}
```

### 4.9.4 设置柔顺控制参数

```
/**
* @brief 设置柔顺控制参数
* @param axis 代表配置哪一轴，可选值为 0~5 柔顺方向，分别对应 fx fy fz mx my mz
* @param opt  0 代表没有勾选，非 0 值代表勾选
* @param ftUser 阻尼力，表示用户用多大的力才能让机械臂的沿着某个方向以最大速度进行运动
* @param ftConstant 代表恒力，手动操作时全部设置为 0
* @param ftNnormalTrack 法向跟踪，手动操作时全部设置为 0，
* @param ftReboundFK 回弹力，表示机械臂回到初始状态的能力
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_admit_ctrl_config(int axis, int opt, double ftUser, double ftConstant,
    int ftNnormalTrack, double ftReboundFK);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//设置柔顺控制参数
int example_set_admit_ctrl_config()
```

```
{
    int ret;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    //设置柔顺控制参数
    ret = demo.set_admit_ctrl_config(1,1,20,5,0,0);
    std::cout << ret << std::endl;
    return 0;
}
```

### 4.9.5 设置传感器末端负载

```
/**
* @brief 设置传感器末端负载
* @param payload 末端负载
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_torq_sensor_tool_payload(const PayLoad* payload);
```

### 4.9.6 获取末端负载辨识状态

```
/**
* @brief 获取末端负载辨识状态
* @param identify_status 0 代表辨识完成，1 代表未完成，2 代表辨识失败
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_torq_sensor_identify_staus(int* identify_status);
```

### 4.9.7 开始辨识工具末端负载

```
/**
* @brief 开始辨识工具末端负载
* @param joint_pos 使用力矩传感器进行自动负载辨识时的结束位置
* @return ERR_SUCC 成功 其他失败
*/
errno_t start_torq_sensor_payload_identify(const JointValue* joint_pos);
```

代码示例：

```
#include <iostream>
```

```
#include "JAKAZuRobot.h"
#include <windows.h>

//辨识工具末端负载和获取负载辨识状态，设置与获取传感器末端负载
int example_sensor_payload()
{
    JointValue joint_pos;
    PayLoad pl,pl_ret;
    int ret;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    //开始辨识传感器负载
    ret = demo.start_torq_sensor_payload_identify(&joint_pos);
    do
    {
        //查询传感器负载状态
        demo.get_torq_sensor_identify_staus(&ret);
        std::cout << ret << std::endl;
    } while (1 == ret);
    //获取辨识结果
    ret = demo.get_torq_sensor_payload_identify_result(&pl);
    std::cout << ret << std::endl;
    //设置传感器末端负载
    ret = demo.set_torq_sensor_tool_payload(&pl);
    //获取当前设置的传感器末端负载
    ret = demo.get_torq_sensor_tool_payload(&pl_ret);
    return 0;
}
```

### 4.9.8 获取末端负载辨识结果

```
/**
* @brief 获取末端负载辨识结果
* @param payload 末端负载
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_torq_sensor_payload_identify_result(PayLoad* payload);
```

### 4.9.9 获取传感器末端负载

```
/**
```

```
 * @brief 获取传感器末端负载
 * @param payload 末端负载
 * @return ERR_SUCC 成功 其他失败
 */
errno_t get_torq_sensor_tool_payload(PayLoad* payload);
```

## 4.9.10 设置导纳控制运动坐标系

```
/**
 * @brief 设置导纳控制运动坐标系
 * @param ftFrame 0工具 1世界
 * @return ERR_SUCC 成功 其他失败
 */
errno_t set_ft_ctrl_frame(const int ftFrame);
```

## 4.9.11 获取导纳控制运动坐标系

```
/**
 * @brief 获取导纳控制运动坐标系
 * @param ftFrame 0工具 1世界
 * @return ERR_SUCC 成功 其他失败
 */
errno_t get_ft_ctrl_frame(int* ftFrame);
```

## 4.9.12 力控导纳使能

```
/**
 * @brief 力控导纳使能，需要先行设置柔顺控制参数，并开启和初始化力控传感器
 * @param enable_flag 0 为关闭力控拖拽使能，1 为开启
 * @return ERR_SUCC 成功 其他失败
 */
errno_t enable_admittance_ctrl(const int enable_flag);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//4.99 力控导纳控制使能
int example_enable_admittance_ctrl()
{
    int ret;
    //实例 API 对象 demo
```

```
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.105 替换为自己控制器的 IP
    demo.login_in("10.5.5.100");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //设置力控传感器品牌
    demo.set_torsenosr_brand(2);
    //开启力控传感器
    demo.set_torque_sensor_mode(1);
    //初始化传感器
    demo.set_compliant_type(1, 1);
    printf("inint sensor comple\n");
    //设置柔顺控制参数
    ret = demo.set_admit_ctrl_config(0, 0, 20, 5, 0, 0);
    ret = demo.set_admit_ctrl_config(1, 0, 20, 5, 0, 0);
    ret = demo.set_admit_ctrl_config(2, 2, 20, 5, 0, 0);
    ret = demo.set_admit_ctrl_config(3, 0, 20, 5, 0, 0);
    ret = demo.set_admit_ctrl_config(4, 0, 20, 5, 0, 0);
    ret = demo.set_admit_ctrl_config(5, 0, 20, 5, 0, 0);
    //设置力控拖拽使能，1 打开，0 关闭
    ret = demo.enable_admittance_ctrl(1);
    printf("enable_admittance_ctrl open! \n");
    std::cout << ret << std::endl;
    printf("input any word to quit:\n");
    std::cin >> ret;
    ret = demo.enable_admittance_ctrl(0);
    ret = demo.set_admit_ctrl_config(2, 0, 20, 5, 0, 0);
    demo.set_torque_sensor_mode(0);
    printf("close\n");
    return 0;
}
```

## 4.9.13 设置力控类型和传感器初始化状态

```
/**
* @brief 设置力控类型和传感器初始化状态
* @param sensor_compensation 是否开启传感器补偿,1 代表开启即初始化,0 代表不初始化
* @param compliance_type 0 代表不使用任何一种柔顺控制方法 1 代表恒力柔顺控制,2 代表速度柔顺控制
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_compliant_type(int sensor_compensation, int compliance_type);
```

代码示例：

```cpp
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//设置与获取力控类型和传感器初始化状态
int example_set_compliant_type()
{
    int ret,sensor_compensation,compliance_type;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    demo.servo_move_enable(TRUE);
    //设置力控类型和传感器初始化状态
    ret = demo.set_compliant_type(1,0);
    std::cout << ret << std::endl;
    ret = demo.get_compliant_type(&sensor_compensation, &compliance_type);
    std::cout << ret << std::endl;
    return 0;
}
```

## 4.9.14 获取力控类型和传感器初始化状态

```cpp
/**
* @brief 获取力控类型和传感器初始化状态
* @param sensor_compensation 是否开启传感器补偿,1 代表开启即初始化,0 代表不初始化
* @param compliance_type 0 代表不使用任何一种柔顺控制方法 1 代表恒力柔顺控制,2 代表速度柔顺控制
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_compliant_type(int* sensor_compensation, int* compliance_type);
```

## 4.9.15 获取力控柔顺控制参数

```cpp
/**
* @brief 获取力控柔顺控制参数
* @param admit_ctrl_cfg 机械臂力控柔顺控制参数存储地址
* @return ERR_SUCC 成功 其他失败
*/
 errno_t get_admit_ctrl_config(RobotAdmitCtrl *admit_ctrl_cfg);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//获取力控柔顺控制参数
int example_get_admit_ctrl_config()
{
    RobotAdmitCtrl adm_ctr_cfg;
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //获取力控柔顺控制参数
    ret = demo.get_admit_ctrl_config(&adm_ctr_cfg);
    std::cout << ret << std::endl;
    return 0;
}
```

## 4.9.16 设置力控传感器通信参数

```
/**
* @brief 设置力控传感器通信参数
* @param type 通信类型，0 为使用 tcp/ip 协议，1 为使用 RS485 协议
* @param ip_addr 为力控传感器地址
* @param port 为使用 tcp/ip 协议时力控传感器端口号
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_torque_sensor_comm(const int type, const char* ip_addr, const int port);
```

代码示例：

```
int example_torque_sensor_comm()
{
    char ip_set[30]="192.168.2.108";
    int ret=2;
    int type_set = 0,port_set = 4008;
    char ip_ret[30]="1";
    int type_ret = 0, port_ret = 0;
    //实例 API 对象 demo
    JAKAZuRobot demo;
```

```
    //登陆控制器，需要将 192.168.2.105 替换为自己控制器的 IP
    printf("logining!\n");
    demo.login_in("192.168.2.106");
    //机械臂上电
    printf("powering\n");
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //设置传感器品牌
    ret = demo.set_torsenosr_brand(4);
    //获取力控通讯参数
    ret = demo.get_torque_sensor_comm(&type_ret, ip_ret, &port_ret);
    std::cout << ret << std::endl;
    std::cout << ip_ret << std::endl;
    std::cout << port_ret << std::endl;
    std::cin >> type_ret;
    //设置力控通信参数
    ret = demo.set_torque_sensor_comm(type_set, ip_set, port_set);
    std::cout << ret << std::endl;
    std::cout << ip_set << std::endl;
    std::cout << port_set << std::endl;
    std::cin >> type_set;
    return 0;
}
```

## 4.9.17 获取力控传感器通信参数

```
/**
* @brief 获取力控传感器通信参数，
* @param type 通信类型， 0 为使用 tcp/ip 协议，1 为使用 RS485 协议
* @param ip_addr 为当前设置的力控传感器通信地址。仅为通讯接口地址
* @param port 为使用 tcp/ip 协议时力控传感器端口号
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_torque_sensor_comm(int* type, char* ip_addr,int* port);
```

## 4.9.18 关闭力矩控制

```
/**
* @brief 关闭力矩控制
* @return ERR_SUCC 成功 其他失败
*/
errno_t disable_force_control();
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>

//关闭力矩控制
int example_disable_force_control()
{
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //关闭力矩控制
    ret = demo.disable_force_control();
    std::cout << ret << std::endl;
    return 0;
}
```

## 4.9.19 设置速度柔顺控制参数

```
/**
* @brief 设置速度柔顺控制参数
* @param vel_cfg 为速度柔顺控制参数
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_vel_compliant_ctrl(const VelCom* vel_cfg);
```

## 4.9.20 设置柔顺控制力矩条件

```
/**
* @brief 设置柔顺控制力矩条件
* @param ft 为柔顺控制力矩限制，超过限制值会停止
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_compliance_condition(const FTxyz* ft);
```

代码示例：

```
#include <iostream>
#include "JAKAZuRobot.h"
```

```
#include <windows.h>

//设置柔顺力矩条件
int example_set_compliance_condition()
{
    FTxyz ft;
    ft.fx = 10; ft.fy = 10; ft.fz = 10;
    ft.tx = 10; ft.ty = 10; ft.tz = 10;
    int ret;
    //实例 API 对象 demo
    JAKAZuRobot demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    demo.login_in("192.168.2.194");
    //机械臂上电
    demo.power_on();
    //机械臂上使能
    demo.enable_robot();
    //设置柔顺力矩条件
    ret = demo.set_compliance_condition(&ft);
    std::cout << ret << std::endl;
    return 0;
}
```

## 4.9.21 设置力控的低通滤波器参数

```
/**
* @brief 设置力控的低通滤波器的值
* @param torque_sensor_filter 低通滤波器的值,单位：Hz
*/
errno_t set_torque_sensor_filter(const float torque_sensor_filter);
```

## 4.9.22 获取力控的低通滤波器参数

```
/**
* @brief 获取力控的低通滤波器的值
* @param torque_sensor_filter 低通滤波器的值,单位：Hz
*/
errno_t get_torque_sensor_filter(float *torque_sensor_filter);
```

## 4.9.23 设置力传感器的传感器限位参数配置

```
/**
```

```
2.  * @brief 设置力传感器的传感器限位参数配置
3.  * @param torque_sensor_soft_limit 力传感器的传感器限位参数
4.  *        力限位 fx, fy, fz 单位：N
5.  *        力矩限位 tx, ty, tz 单位：N*m
6.  */
7.  errno_t set_torque_sensor_soft_limit(const FTxyz torque_sensor_soft_limit);
```

## 4.9.24 获取力传感器的传感器限位参数配置

```
1.  /**
2.  * @brief 获取力传感器的传感器限位参数配置
3.  * @param torque_sensor_soft_limit 力传感器的传感器限位参数
4.  *        力限位 fx, fy, fz 单位：N
5.  *        力矩限位 tx, ty, tz 单位：N*m
6.  */
7.  errno_t get_torque_sensor_soft_limit(FTxyz *torque_sensor_soft_limit);
```

以下为内嵌 S 适配接口

## 4.9.25 传感器校零

```
1.   /**
2.    * @brief 触发传感器校零，并且阻塞 0.5 秒。注：该接口仅用于 JAKA S 系列机器人。
3.    */
4.   errno_t zero_end_sensor();
```

## 4.9.26 获取力控工具拖拽开启状态

```
1.   /**
2.   * @brief 获取力控工具拖拽开启状态，即控制器内部是否处于 admitting 模式。注：该接口仅用于 JAKA S 系列机器人。
3.
4.   * @param enable_flag: 0 为关闭力控拖拽使能，1 为开启，drive_stat 是拖拽的当前状态是否触发奇异点、速度、关节限位预警
5.   * @return ERR_SUCC ERR_SUCC 成功 其他失败
6.
7.   */
8.   errno_t get_tool_drive_state(int* enable, int *state);
```

## 4.9.27 获取工具拖拽坐标系

```
1.   /**
```

```
2.  * @brief 获取工具拖拽坐标系。注：该接口仅用于 JAKA S 系列机器人。
3.  * @param toolDragFrame 0 工具 1 世界
4.  * @return ERR_SUCC Success Other failures
5.  */
6.  errno_t get_tool_drive_frame(FTFrameType *ftFrame);
```

## 4.9.28 设置工具拖拽坐标系

```
1.  /**
2.  * @brief 设置工具拖拽坐标系。注：该接口仅用于 JAKA S 系列机器人。
3.
4.  * @param toolDragFrame  0 Tool  1 World
5.  * @return ERR_SUCC 0 工具 1 世界
6.  */
7.  errno_t set_tool_drive_frame(FTFrameType ftFrame);
```

## 4.9.29 获取融合拖拽灵敏度

```
1.  /**
2.  * @brief 获取融合拖拽灵敏度。注：该接口仅用于 JAKA S 系列机器人。
3.  */
4.  errno_t get_fusion_drive_sensitivity_level(int *level);
```

## 4.9.30 设置融合拖拽灵敏度

```
1.  /**
2.  * @brief 设置融合拖拽灵敏度。注：该接口仅用于 JAKA S 系列机器人。
3.  * @param sensitivity_level 灵敏度等级，0-5，0 代表关闭
4.  */
5.  errno_t set_fusion_drive_sensitivity_level(int level);
```

## 4.9.31 获取运动限制（奇异点和关节限位）预警范围

```
1.  /**
2.  * @brief 获取 运动限制（奇异点和关节限位）预警范围 */。注：该接口仅用于 JAKA S 系列机器人。
3.  errno_t get_motion_limit_warning_range(int *warningRange);
```

## 4.9.32 设置运动限制（奇异点和关节限位）预警范围

```
1.  /**
```

```
2.  * @brief 设置运动限制（奇异点和关节限位）预警范围。注：该接口仅用于 JAKA S 系列机器人。
3.  * @param range_level 范围等级，1-5
4.  */
5.  errno_t set_motion_limit_warning_range(int warningRange);
```

### 4.9.33 获取力控限速

```
1.  /**
2.  * @brief 获取力控限速。注：该接口仅用于 JAKA S 系列机器人。
3.  */
4.  errno_t get_compliant_speed_limit(double* vel, double* angularVel);
```

### 4.9.34 设置力控限速

```
1.  /**
2.  * @brief 设置力控限速。注：该接口仅用于 JAKA S 系列机器人。
3.  * @param speed_limit 线速度限制，mm/s
4.  * @param angular_speed_limit 角速度限制，deg/s
5.  */
6.  errno_t set_compliant_speed_limit(double vel, double angularVel);
```

### 4.9.35 获取力矩参考中心

```
1.  /**
2.  * @brief 获取力矩参考中心。注：该接口仅用于 JAKA S 系列机器人。
3.  */
4.  errno_t get_torque_ref_point(int *refPoint);
```

### 4.9.36 设置力矩参考中心

```
1.  /**
2.  * @brief  设置力矩参考中心。注：该接口仅用于 JAKA S 系列机器人。
3.  * @param ref_point 0 代表传感器中心，1 代表 TCP
4.  */
5.  errno_t set_torque_ref_point(int refPoint);
```

### 4.9.37 获取传感器灵敏度

```
1.  /**
2.  * @brief 获取传感器灵敏度。注：该接口仅用于 JAKA S 系列机器人。
3.  */
```

```
errno_t get_end_sensor_sensitivity_threshold(int *refPoint);
```

## 4.9.38 设置传感器灵敏度

```
/**
* @brief 设置传感器灵敏度。注：该接口仅用于 JAKA S 系列机器人。

* @param threshold_percent 6 维数组，0~1，越大传感器越不灵敏
*/
errno_t set_end_sensor_sensitivity_threshold(FTxyz threshold);
```

上述内嵌 S 接口的示例：

```
int main()
{
std::cout << "Hello World!\n";

JAKAZuRobot demo;
demo.login_in("10.5.5.100");

cout << "power on: " << demo.power_on() << endl;
cout << "enable: " << demo.enable_robot() << endl;

cout << "enable tool drive: " << demo.enable_tool_drive(1) << endl;

//set tool_drive_frame
FTFrameType tool_drive_frame;
demo.set_tool_drive_frame(FTFrameType::FTFrame_Tool);
demo.get_tool_drive_frame(&tool_drive_frame);
std::cout << tool_drive_frame << std::endl;

//set torque_ref_point
int ref_point;
demo.set_torque_ref_point(1); //set to follow the TCP
demo.get_torque_ref_point(&ref_point);
std::cout << ref_point << std::endl;

//compliant_speed_limit
double vel, angvel;
demo.set_compliant_speed_limit(500, 60);
demo.get_compliant_speed_limit(&vel, &angvel);
std::cout << vel << ", " << angvel << std::endl; //should be: 500, 60

//torque_sensor_sensitivity_threshold
```

```
FTxyz ft;
ft.fx = 0.3;
ft.fy = 0.3;
ft.fz = 0.3;
ft.tx = 0.3;
ft.ty = 0.3;
ft.tz = 0.3;
demo.set_end_sensor_sensitivity_threshold(ft);
demo.get_end_sensor_sensitivity_threshold(&ft);
std::cout << ft.fx << ", " << ft.fy << ", " << ft.fz << ", " << ft.tx << ", " << ft.ty << ", " << ft.tz << ", " << std::endl; //should be: 0.3, 0.3, 0.3, 0.3, 0.3, 0.3
//lock all other directions, only open z direction
ToolDriveConfig toolDriveCfg{};
RobotToolDriveCtrl robToolCfg;
for (int i = 0; i < 6; i++)
{
toolDriveCfg.axis = i;
if(i == 2)
toolDriveCfg.opt = 1;
Else
toolDriveCfg.opt = 0;
toolDriveCfg.rigidity = 0.3;
toolDriveCfg.rebound = 0;
demo.set_tool_drive_config(toolDriveCfg);
}

demo.get_tool_drive_config(&robToolCfg);

//zero_end_sensor
cout << "zero: " << demo.zero_end_sensor() << endl;

//open tool drive
demo.enable_tool_drive(1);
int enable, state;
demo.get_tool_drive_state(&enable, &state);
Sleep(2000);
demo.enable_tool_drive(0);

//退出登录
demo.login_out();

cout << "end" << endl;
return 0;


```

```
76. }
```

# 4.10 FTP 服务

## 4.10.1 初始化 FTP 客户端

```
1. /**
2. * @brief 初始化 ftp 客户端，与控制柜建立连接，可导出 program、track
3. * @return ERR_SUCC 成功 其他失败
4. */
5. errno_t init_ftp_client();
```

## 4.10.2 FTP 上传

```
1. /**
2. * @brief 从本地上传指定类型和名称的文件到控制器
3. * @param remote 上传到控制器内部文件名绝对路径，若为文件夹需要以“\”或“/”结尾
4. * @param local 本地文件名绝对路径，若为文件夹需要以“\”或“/”结尾
5. * @param opt 1 单个文件 2 文件夹
6. * @return ERR_SUCC 成功 其他失败
7. * @note Windows C++/ Windows C#，传入参数编码为 GBK；Linux 为 UTF-8
8. */
9. errno_t upload_file(char* local, char* remote, int opt);
```

## 4.10.3 FTP 下载

```
1. /**
2. * @brief 从控制器下载指定类型和名称的文件到本地
3. * @param remote 控制器内部文件名绝对路径，若为文件夹需要以“\”或“/”结尾
4. * @param local 下载到本地文件名绝对路径，若为文件夹需要以“\”或“/”结尾
5. * @param opt 1 单个文件 2 文件夹
6. * @return ERR_SUCC 成功 其他失败
7. * @note Windows C++/ Windows C#，传入参数编码为 GBK；Linux 为 UTF-8
8. */
9. errno_t download_file(char* local, char* remote, int opt);
```

## 4.10.4 FTP 目录查询

```
1. /**
2. * @brief 查询 FTP 目录
3. * @param remote 控制器内部文件名原名称，查询轨迹“/track/”,查询脚本程序“/program/”
```

```
4. * @param opt 0 文件名和子目录名 1 文件名 2 子目录名
5. * @param ret 返回的查询结果
6. * @return ERR_SUCC 成功 其他失败
7. * @note Windows C++/ Windows C#，传入参数编码为 GBK；Linux 为 UTF-8
8. */
9. errno_t get_ftp_dir(const char* remotedir, int type, char* ret);
```

### 4.10.5 FTP 删除

```
1. /**
2. * @brief 从控制器删除指定类型和名称的文件
3. * @param remote 控制器内部文件名
4. * @param opt 1 单个文件 2 文件夹
5. * @return ERR_SUCC 成功 其他失败
6. * @note Windows C++/ Windows C#，传入参数编码为 GBK；Linux 为 UTF-8
7. */
8. errno_t del_ftp_file(char* remote, int opt);
```

### 4.10.6 FTP 重命名

```
1. /**
2. * @brief 重命名控制器指定类型和名称的文件
3. * @param remote 控制器内部文件名原名称
4. * @param des 重命名的目标名
5. * @param opt 1 单个文件 2 文件夹
6. * @return ERR_SUCC 成功 其他失败
7. * @note Windows C++/ Windows C#，传入参数编码为 GBK；Linux 为 UTF-8
8. */
9. errno_t rename_ftp_file(char* remote, char* des, int opt);
```

### 4.10.7 关闭 FTP 客户端

```
1. /**
2. * @brief 断开与控制器 ftp 链接
3. * @return ERR_SUCC 成功 其他失败
4. */
5. errno_t close_ftp_client();
```

# 5. 反馈与勘误

文档中若出现不准确的描述或者错误，恳请读者指正批评。如果您在阅读过程中发现任何问题或者有想提出的意见，可以发送邮件到 support@jaka.com ，我们的同事会尽量一一回复。