JAKA 节卡

C++函数使用手册

版本：V1.0.1

时间：2020.05.27

上海节卡机器人科技有限公司　Shanghai JAKA Robotics Ltd
电话 Tel ：+400 006 2665 | 网站 Web:www.jaka.com
上海：上海市闵行区剑川路 610 号 33-35 幢 | Building 33-35, No.610 Jianchuan Rd, Minhang District, Shanghai
常州：江苏省常州市武进国家高新区武宜南路 377 号 10 号楼 | Building 10, No.377 South Wuyi Rd, Changzhou, Jiangsu

产权说明

版本记录

版本编号	版本日期	支持版本	说明
V1.0.0	2020.05.27	V1.4.10/V2.0.10 及以上	创建
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	修改目录 3.21、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 4.101-4.107

目录

1. 简介 - 8 -
2. 文档须知 - 8 -
3. 数据结构 - 8 -
3.1 回调函数类型 - 8 -
3.2 接口调用返回值列表 - 8 -
3.3 接口调用返回值类型 - 9 -
3.4 机器人控制句柄类型 - 9 -
3.5 布尔类型 - 9 -
3.6 笛卡尔空间位置数据类型 - 9 -
3.7 欧拉角姿态数据类型 - 9 -
3.8 四元数姿态数据类型 - 10 -
3.9 笛卡尔空间位姿类型 - 10 -
3.10 旋转矩阵数据类型 - 10 -
3.11 程序运行状态枚举类型 - 10 -
3.12 坐标系选择枚举类型 - 11 -
3.13 运动模式枚举类型 - 11 -
3.14 系统检测数据类型 - 11 -
3.15 负载数据类型 - 12 -
3.16 关节位置数据类型 - 12 -
3.17 IO 类型 - 12 -
3.18 机器人状态数据类型 - 13 -
3.19 机器人力矩数据类型 - 13 -
3.20 机器人关节监测数据类型 - 13 -
3.21 机器人监测数据类型 - 13 -
3.22 力矩传感器监测数据类型 - 14 -
3.23 机器人状态监测数据类型 - 14 -
3.24 机器人错误码数据类型 - 15 -
3.25 轨迹复现配置参数存储数据类型 - 16 -
3.26 多个字符串存储数据类型 - 16 -
3.27 运动参数可选项 - 16 -
3.28 网络异常机器人运动自动终止类型枚举 - 17 -
3.29 机器人柔顺控制参数类型 - 17 -
3.30 机器人柔顺控制参数类型 - 17 -
3.31 速度柔顺控制等级和比率等级设置 - 18 -
3.32 力传感器的受力分量和力矩分量 - 18 -
4. API - 19 -
4.1 机械臂控制类构造函数 - 19 -
4.2 机器人登陆 - 19 -
4.3 机器人注销 - 19 -
4.4 机器人上电 - 19 -
4.5 机器人下电 - 19 -
4.6 机器人关机 - 20 -
4.7 机器人上使能 - 20 -

4.8 机器人下使能 - 20 -
4.9 机器人手动模式下运动 - 20 -
4.10 机器人手动模式下运动停止 - 21 -
4.11 机器人关节运动 - 21 -
4.12 机器人末端直线运动 - 21 -
4.13 机器人伺服位置控制模式使能 - 21 -
4.14 机器人关节空间伺服模式运动 - 22 -
4.15 机器人关节空间伺服模式运动扩展 - 22 -
4.16 机器人笛卡尔空间伺服模式运动 - 22 -
4.17 机器人笛卡尔空间伺服模式运动扩展 - 23 -
4.18 设置数字输出变量 - 23 -
4.19 设置模拟输出变量 - 23 -
4.20 查询数字输入状态 - 23 -
4.21 查询数字输出状态 - 24 -
4.22 获取模拟量输入变量的值 - 24 -
4.23 获取模拟量输出变量的值 - 24 -
4.24 查询扩展 IO 是否运行 - 25 -
4.25 运行当前加载的作业程序 - 25 -
4.26 暂停当前运行的作业程序 - 25 -
4.27 继续运行当前暂停的作业程序 - 25 -
4.28 终止当前执行的作业程序 - 25 -
4.29 加载指定的作业程序 - 26 -
4.30 获取已加载的作业程序的名字 - 26 -
4.31 获取当前机器人作业程序的执行行号 - 26 -
4.32 获取机器人作业程序的执行状态 - 26 -
4.33 设置机器人的运行倍率 - 26 -
4.34 获取机器人的运行倍率 - 27 -
4.35 设置工具信息 - 27 -
4.36 获取工具信息 - 27 -
4.37 设置当前使用的工具 ID - 27 -
4.38 查询当前使用的工具 ID - 28 -
4.39 设定用户坐标系信息 - 28 -
4.40 获取用户坐标系信息 - 28 -
4.41 设置当前使用的用户坐标系 ID - 28 -
4.42 查询当前使用的用户坐标系 ID - 29 -
4.43 控制机器人进入或退出拖拽模式 - 29 -
4.44 查询机器人是否处于拖拽模式 - 29 -
4.45 获取机器人状态 - 29 -
4.46 获取当前设置下工具末端的位姿 - 30 -
4.47 获取当前机器人关节角度 - 30 -
4.48 查询机器人是否处于碰撞保护模式 - 30 -
4.49 查询机器人是否超出限位 - 30 -
4.50 查询机器人运动是否停止 - 30 -
4.51 碰撞之后从碰撞保护模式恢复 - 31 -

4.52 设置机器人碰撞等级 - 31 -
4.53 获取机器设置的碰撞等级 - 31 -
4.54 机器人求解逆解 - 31 -
4.55 机器人求解正解 - 32 -
4.56 欧拉角到旋转矩阵的转换 - 32 -
4.57 旋转矩阵到欧拉角的转换 - 32 -
4.58 四元数到旋转矩阵的转换 - 32 -
4.59 旋转矩阵到四元数的转换 - 33 -
4.60 注册机器人出错时的回调函数 - 33 -
4.61 机器人力矩控制模式使能 - 33 -
4.62 机器人力矩前馈功能 - 33 -
4.63 机器人负载设置 - 34 -
4.64 获取机器人负载数据 - 34 -
4.65 获取 SDK 版本号 - 34 -
4.66 获取控制器 IP - 34 -
4.67 获取机器人状态监测数据 - 35 -
4.68 设置 SDK 是否开启调试模式 - 35 -
4.69 机器人运动终止 - 35 -
4.70 设置机器人错误码文件存放路径 - 35 -
4.71 获取机器人目前发生的最后一个错误码 - 35 -
4.72 设置轨迹复现配置参数 - 36 -
4.73 获取轨迹复现配置参数 - 36 -
4.74 采集轨迹复现数据控制开关 - 36 -
4.75 采集轨迹复现数据状态查询 - 36 -
4.76 查询控制器中已经存在的轨迹复现数据的文件名 - 37 -
4.77 重命名轨迹复现数据的文件名 - 37 -
4.78 删除控制器中轨迹复现数据文件 - 37 -
4.79 控制器中轨迹复现数据文件生成控制器执行脚本 - 37 -
4.80 机器人扩展关节运动 - 37 -
4.81 机器人扩展末端直线运动 - 38 -
4.82 机器人圆弧运动 - 38 -
4.83 机器人 SERVO 模式下禁用滤波器 - 39 -
4.84 机器人 SERVO 模式下关节空间一阶低通滤波 - 39 -
4.85 机器人 SERVO 模式下关节空间非线性滤波 - 39 -
4.86 机器人 SERVO 模式下笛卡尔空间非线性滤波 - 39 -
4.87 机器人 SERVO 模式下关节空间多阶均值滤波 - 40 -
4.88 机器人 SERVO 模式下速度前瞻参数设置 - 40 -
4.89 设置 SDK 日志路径 - 41 -
4.90 设置传感器品牌 - 41 -
4.91 获取传感器品牌 - 41 -
4.92 开启或关闭力矩传感器 - 41 -
4.93 设置柔顺控制参数 - 41 -
4.94 开始辨识工具末端负载 - 42 -
4.95 获取末端负载辨识状态 - 42 -

4.96 获取末端负载辨识结果 - 42 -
4.97 设置传感器末端负载 - 43 -
4.98 获取传感器末端负载 - 43 -
4.99 力控拖拽使能 - 43 -
4.100 设置力控类型和传感器初始化状态 - 43 -
4.101 获取力控类型和传感器初始化状态 - 43 -
4.102 获取力控柔顺控制参数 - 44 -
4. 103 设置力控传感器 IP 地址 - 44 -
4.104 获取力控传感器 IP 地址 - 44 -
4.105 关闭力矩控制 - 45 -
4.106 设置速度柔顺控制参数 - 45 -
4.107 设置柔顺控制力矩条件 - 45 -
4.108 设置网络异常时机器人自动终止运动类型 - 45 -
4.109 设置机器人状态数据自动更新时间间隔 - 46 -
5. 接口使用示例 - 46 -
5.1 机器人设置与启动 - 46 -
5.2 异常回调函数的注册 - 47 -
5.3 机器人关节与笛卡尔空间运动 - 48 -
5.4 控制多台机器人 - 48 -
6. 反馈与勘误 - 49 -

1.简介

jakaAPI 是基于网络通信协议 TCP/IP 实现的，提供了用于操作机械臂的接口，提供 python，C/C++，C#四种语言的支持。

本文是基于 C++开发的，其中类的方法是操作机械臂的接口。

2.文档须知

- 接口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：windows 中右击 dll 文件，选择属性，在“详细信息”选项卡中可以看到版本信息。linux 中输入命令 strings libjakaAPI.so | grep jakaAPI_version 查询版本信息。
- 节卡 SDK 使用的编码方式为 UTF-8 编码。

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 机器人回调函数指针
*/
typedef void(*CallBackFuncType)(int);
```

3.2 接口调用返回值列表

```
#define ERR_SUCC                    0           //调用成功
#define ERR_INVALID_HANDLER         -1          //无效的控制句柄
#define ERR_INVALID_PARAMETER       -2          //无效的传递参数
#define ERR_COMMUNICATION_ERR       -3          //通信错误
#define ERR_KINE_INVERSE_ERR        -4          //逆解失败
```

3.3 接口调用返回值类型

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

3.4 机器人控制句柄类型

```
typedef int JKHD;           //机械臂控制句柄类型
```

3.5 布尔类型

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

3.6 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;
```

3.7 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：rad
  double ry;    ///< 绕固定轴 Y 旋转角度，单位：rad
  double rz;    ///< 绕固定轴 Z 旋转角度，单位：rad
}Rpy;
```

3.8 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;
```

3.9 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;
```

3.10 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;
```

3.11 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,       ///< 机器人停止运行
    PROGRAM_RUNNING,    ///< 机器人正在运行
    PROGRAM_PAUSED      ///< 机器人暂停
}ProgramState;
```

3.12 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;
```

3.13 运动模式枚举类型

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

3.14 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];          ///<机器人 6 个关节轴的瞬时电流
    double instVoltage[6];          ///<机器人 6 个关节轴的瞬时电压
    double instTemperature[6];      ///<机器人 6 个关节轴的瞬时温度
}SystemMonitorData;
```

3.15 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;
```

3.16 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：rad
}JointValue;
```

3.17 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
}IOType;
```

3.18 机器人状态数据类型

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

3.19 机器人力矩数据类型

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

3.20 机器人关节监测数据类型

```
/**
* @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
}JointMonitorData;
```

3.21 机器人监测数据类型

```
/**
* @brief 机器人监测数据类型
*/
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;               ///< 机器人平均电压
    double robotAverageCurrent;             ///< 机器人平均电流
    JointMonitorData jointMonitorData[6];   ///< 机器人 6 个关节的监测数据
}RobotMonitorData;
```

3.22 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
}TorqSensorMonitorData;
```

3.23 机器人状态监测数据类型

```
/**
* @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[65];                                        ///< 机器人控制柜数字输出信号,dout[0]为信号的个数
    int tio_dout[10];                                    ///< 机器人末端工具数字输出信号,tio_dout[0]为信号的个数
    int extio[65];                                       ///< 机器人外部应用数字输出信号,extio[0]为信号的个数
    int din[65];                                         ///< 机器人控制柜数字输入信号,din[0]为信号的个数
    int tio_din[10];                                     ///< 机器人末端工具数字输入信号,tio_din[0]为信号的个数
    double ain[65];                                      ///< 机器人控制柜模拟输入信号,ain[0]为信号的个数
    double tio_ain[10];                                  ///< 机器人末端工具模拟输入信号,tio_ain[0]为信号的个数
    double aout[65];                                     ///< 机器人控制柜模拟输出信号,aout[0]为信号的个数
    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;
```

3.24 机器人错误码数据类型

```
/**
* @brief 机器人错误码数据类型
*/
typedef struct
{
    long code;                                           ///< 错误码编号
```

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

3.25 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;
```

3.26 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;
```

3.27 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;
```

3.28 网络异常机器人运动自动终止类型枚举

```
/**
* @brief 网络异常机器人运动自动终止类型枚举
*/
typedef enum
{
    MOT_KEEP,       ///< 网络异常时机器人继续保持原来的运动
    MOT_PAUSE,      ///< 网络异常时机器人暂停运动
    MOT_ABORT       ///< 网络异常时机器人终止运动
}ProcessType;
```

3.29 机器人柔顺控制参数类型

```
/**
* @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器人的沿着某个方向以最大速度进行运动
    double ft_rebound;    ///< 回弹力:机器人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;
```

3.30 机器人柔顺控制参数类型

```
/**
* @brief 机器人柔顺控制参数类型
*/
typedef struct
{
    AdmitCtrlType admit_ctrl[6];
} RobotAdmitCtrl;
```

3.31 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;
```

3.32 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;
```

4.API

4.1 机械臂控制类构造函数

```
/**
* @brief 机械臂控制类构造函数
*/
JAKAZuRobot();
```

4.2 机器人登陆

```
/**
* @brief 连接机器人控制器
* @param ip  控制器 ip 地址
* @return ERR_SUCC 成功 其他失败
*/
errno_t  login_in(const char* ip);
```

4.3 机器人注销

```
/**
* @brief 断开控制器连接
* @return ERR_SUCC 成功 其他失败
*/
errno_t  login_out();
```

4.4 机器人上电

```
/**
* @brief 打开机器人电源
* @return ERR_SUCC 成功 其他失败
*/
errno_t  power_on();
```

4.5 机器人下电

```
/**
* @brief 关闭机器人电源
```

```
3.  * @return ERR_SUCC 成功 其他失败
4.  */
5.  errno_t  power_off();
```

4.6 机器人关机

```
1.  /**
2.  * @brief 机器人控制柜关机
3.  * @return ERR_SUCC 成功 其他失败
4.  */
5.  errno_t  shut_down();
```

4.7 机器人上使能

```
1.  /**
2.  * @brief 控制机器人上使能
3.  * @return ERR_SUCC 成功 其他失败
4.  */
5.  errno_t  enable_robot();
```

4.8 机器人下使能

```
1.  /**
2.  * @brief 控制机器人下使能
3.  * @return ERR_SUCC 成功 其他失败
4.  */
5.  errno_t  disable_robot();
```

4.9 机器人手动模式下运动

```
1.  /**
2.  * @brief 控制机器人手动模式下 jog 运动
3.  * @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
4.  * @param move_mode 机器人运动模式，增量运动或者绝对运动(即持续 jog 运动),参考 2.13 选择合适类型
5.  * @param coord_type 机器人运动坐标系，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
6.  * @param vel_cmd 指令速度，旋转轴或关节运动单位为 rad/s，移动轴单位为 mm/s
7.  * @param pos_cmd 指令位置，旋转轴或关节运动单位为 rad，移动轴单位为 mm
8.  * @return ERR_SUCC 成功 其他失败
9.  */
```

```
errno_t  jog(int aj_num, MoveMode move_mode, CoordType coord_type, double vel_cmd, double pos_cmd);
```

4.10 机器人手动模式下运动停止

```
/**
* @brief 控制机器人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();
```

4.11 机器人关节运动

```
/**
* @brief 机器人关节运动
* @param joint_pos 机器人关节运动目标位置
* @param move_mode 指定运动模式：增量运动或绝对运动
* @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 机器人关节运动速度，单位：rad/s
* @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);
```

4.12 机器人末端直线运动

```
/**
* @brief 机器人末端直线运动
* @param end_pos 机器人末端运动目标位置
* @param move_mode 指定运动模式：增量运动或绝对运动
* @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 机器人直线运动速度，单位：mm/s
* @return ERR_SUCC 成功 其他失败
*/
errno_t  linear_move(const CartesianPose* end_pos, MoveMode move_mode, BOOL is_block, double speed);
```

4.13 机器人伺服位置控制模式使能

```
/**
```

```
 * @brief 机器人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
 */
errno_t  servo_move_enable(BOOL enable);
```

4.14 机器人关节空间伺服模式运动

```
/**
 * @brief 机器人关节空间位置控制模式
 * @param joint_pos 机器人关节运动目标位置
 * @param move_mode 指定运动模式：绝对运动或相对运动
 * @return ERR_SUCC 成功 其他失败
 */
errno_t  servo_j(const JointValue* joint_pos, MoveMode move_mode);
```

4.15 机器人关节空间伺服模式运动扩展

```
/**
 * @brief 机器人关节空间位置控制模式
 * @param handle  机器人控制句柄
 * @param joint_pos 机器人关节运动目标位置
 * @move_mode 指定运动模式：增量运动或绝对运动
 * @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_extend(const JKHD* handle, const JointValue* joint_pos, MoveMode move_mode, unsigned int step_num);
```

4.16 机器人笛卡尔空间伺服模式运动

```
/**
 * @brief 机器人笛卡尔空间位置控制模式
 * @param cartesian_pose 机器人笛卡尔空间运动目标位置
 * @param move_mode 指定运动模式：ABS 代表绝对运动，INCR 代表相对运动
 * @return ERR_SUCC 成功 其他失败
 */
errno_t  servo_p(const CartesianPose* cartesian_pose, MoveMode move_mode);
```

4.17 机器人笛卡尔空间伺服模式运动扩展

```
/**
* @brief 机器人笛卡尔空间位置控制模式
* @param handle  机器人控制句柄
* @param cartesian_pose 机器人笛卡尔空间运动目标位置
* @move_mode 指定运动模式：增量运动或绝对运动
* @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_extend(const JKHD* handle, const CartesianPose* cartesian_pose, MoveMode move_mode, unsigned int step_num);
```

4.18 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);
```

4.19 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);
```

4.20 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);
```

4.21 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);
```

4.22 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);
```

4.23 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);
```

4.24 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);
```

4.25 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();
```

4.26 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();
```

4.27 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();
```

4.28 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();
```

4.29 加载指定的作业程序

```
/**
* @brief 加载指定的作业程序  （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);
```

4.30 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);
```

4.31 获取当前机器人作业程序的执行行号

```
/**
* @brief 获取当前机器人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);
```

4.32 获取机器人作业程序的执行状态

```
/**
* @brief 获取机器人作业程序执行状态
* @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);
```

4.33 设置机器人的运行倍率

```
/**
* @brief 设置机器人运行倍率
```

```
3.  * @param rapid_rate 是程序运行倍率，设置范围为[0,1]
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  errno_t  set_rapidrate(double rapid_rate);
```

4.34 获取机器人的运行倍率

```
1.  /**
2.  * @brief 获取机器人运行倍率
3.  * @param handle 机器人控制句柄
4.  * @param rapid_rate 当前控制系统倍率
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  errno_t  get_rapidrate(double* rapid_rate);
```

4.35 设置工具信息

```
1.  /**
2.  * @brief 设置指定编号的工具信息
3.  * @param id 工具编号,取值范围为[1,10]
4.  * @param tcp 工具坐标系相对法兰坐标系偏置
5.  * @param name 指定工具的别名
6.  * @return ERR_SUCC 成功 其他失败
7.  */
8.  errno_t  set_tool_data(int id, const CartesianPose* tcp, const char* name);
```

4.36 获取工具信息

```
1.  /**
2.  * @brief 查询当前使用的工具信息
3.  * @param id 工具 ID 查询结果
4.  * @param tcp 工具坐标系相对法兰坐标系偏置
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  errno_t  get_tool_data(int* id, CartesianPose* tcp);
```

4.37 设置当前使用的工具 ID

```
1.  /**
2.  * @brief 设置当前使用的工具 ID
```

```
 * @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);
```

4.38 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);
```

4.39 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);
```

4.40 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);
```

4.41 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);
```

4.42 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);
```

4.43 控制机器人进入或退出拖拽模式

```
/**
 * @brief 控制机器人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);
```

4.44 查询机器人是否处于拖拽模式

```
/**
 * @brief 查询机器人是否处于拖拽模式
 * @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);
```

4.45 获取机器人状态

```
/**
 * @brief 获取机器人状态
 * @param state 机器人状态查询结果
 * @return ERR_SUCC 成功 其他失败
 */
errno_t  get_robot_state(RobotState* state);
```

4.46 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
* @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);
```

4.47 获取当前机器人关节角度

```
/**
* @brief 获取当前机器人关节角度
* @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);
```

4.48 查询机器人是否处于碰撞保护模式

```
/**
* @brief 查询机器人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);
```

4.49 查询机器人是否超出限位

```
/**
* @brief 查询机器人是否超出限位
* @param on_limit 查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  is_on_limit(BOOL* on_limit);
```

4.50 查询机器人运动是否停止

```
/**
* @brief 查询机器人运动是否停止
```

```
3.  * @param in_pos 查询结果
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  errno_t  is_in_pos(BOOL* in_pos);
```

4.51 碰撞之后从碰撞保护模式恢复

```
1.  /**
2.  * @brief 碰撞之后从碰撞保护模式恢复
3.  * @return ERR_SUCC 成功 其他失败
4.  */
5.  errno_t  collision_recover();
```

4.52 设置机器人碰撞等级

```
1.  /**
2.  * @brief 设置机器人碰撞等级
3.  * @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
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  errno_t set_collision_level(const int level);
```

4.53 获取机器设置的碰撞等级

```
1.  /**
2.  * @brief 获取机器人设置的碰撞等级
3.  * @return ERR_SUCC 成功 其他失败
4.  */
5.  errno_t get_collision_level(int* level);
```

4.54 机器人求解逆解

```
1.  /**
2.  * @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
3.  * @param ref_pos 逆解计算用的参考关节空间位置
4.  * @param cartesian_pose 笛卡尔空间位姿值
5.  * @param joint_pos 计算成功时关节空间位置计算结果
6.  * @return ERR_SUCC 成功 其他失败
7.  */
```

```
errno_t  kine_inverse(const JointValue* ref_pos, const CartesianPose* cartesian_pose, JointValue* joint_pos);
```

4.55 机器人求解正解

```
/**
* @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);
```

4.56 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);
```

4.57 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);
```

4.58 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);
```

4.59 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);
```

4.60 注册机器人出错时的回调函数

```
/**
* @brief 注册机器人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 机器人的错误码
*/
errno_t set_error_handler(CallBackFuncType func);
```

4.61 机器人力矩控制模式使能

```
/**
* @brief 机器人力矩控制模式使能
* @param enable  TRUE 为进入力矩控制模式，FALSE 表示退出该模式
* @return ERR_SUCC 成功 其他失败
*/
errno_t torque_control_enable(BOOL enable);
```

4.62 机器人力矩前馈功能

```
/**
* @brief 机器人力矩前馈功能
* @param tor_val  用于力矩前馈的各个关节的力矩值
* @param grv_flag  0 代表选用控制器自带的力矩前馈算法，1 代表设置关节力矩控制的偏置，2 代表关节控制的力矩完全由用户控制
* @return ERR_SUCC 成功 其他失败
*/
errno_t torque_feedforward(TorqueValue tor_val, int grv_flag);
```

4.63 机器人负载设置

```
/**
* @brief 机器人负载设置
* @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);
```

4.64 获取机器人负载数据

```
/**
* @brief 获取机器人负载数据
* @param payload 负载查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_payload(PayLoad* payload);
```

4.65 获取 SDK 版本号

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

4.66 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
*/
errno_t  get_controller_ip(char* controller_name, char* ip_list);
```

4.67 获取机器人状态监测数据

```
/**
* @brief 获取机器人状态数据
* @param status 机器人状态
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_robot_status(RobotStatus* status);
```

4.68 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);
```

4.69 机器人运动终止

```
/**
* @brief 终止当前机械臂运动
* @return ERR_SUCC 成功 其他失败
*/
errno_t  motion_abort();
```

4.70 设置机器人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);
```

4.71 获取机器人目前发生的最后一个错误码

```
/**
* @brief 获取机器人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);
```

4.72 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);
```

4.73 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);
```

4.74 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
* @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);
```

4.75 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);
```

4.76 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);
```

4.77 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);
```

4.78 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);
```

4.79 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);
```

4.80 机器人扩展关节运动

```
/**
```

```
 * @brief 机器人关节运动
 * @param joint_pos 机器人关节运动目标位置
 * @move_mode 指定运动模式：增量运动(相对运动)或绝对运动
 * @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 机器人关节运动速度，单位：rad/s
 * @param acc 机器人关节运动角加速度
 * @param tol 机器人关节运动终点误差
 * @param option_cond 机器人关节可选参数，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);
```

4.81 机器人扩展末端直线运动

```
/**
 * @brief 机器人末端直线运动
 * @param end_pos 机器人末端运动目标位置
 * @move_mode 指定运动模式：增量运动(相对运动)或绝对运动
 * @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 机器人直线运动速度，单位：mm/s
 * @param acc 机器人关节运动角加速度
 * @param tol 机器人关节运动终点误差
 * @param option_cond 机器人关节可选参数，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  linear_move(const CartesianPose* end_pos, MoveMode move_mode, BOOL is_block, doub
le speed, double accel, double tol, const  OptionalCond* option_cond);
```

4.82 机器人圆弧运动

```
/**
 * @brief 机器人末端圆弧运动
 * @param end_pos 机器人末端运动目标位置
 * @param mid_pos 机器人末端运中间点
 * @move_mode 指定运动模式：增量运动(相对运动)或绝对运动
 * @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 机器人直线运动速度，单位：mm/s
 * @param acc 机器人关节运动角加速度
 * @param tol 机器人关节运动终点误差
 * @param option_cond 机器人关节可选参数，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
 */
```

```
13. 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);
```

4.83 机器人 SERVO 模式下禁用滤波器

```
1. /**
2. * @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
3. * @return ERR_SUCC 成功 其他失败
4. */
5. errno_t  servo_move_use_none_filter();
```

4.84 机器人 SERVO 模式下关节空间一阶低通滤波

```
1. /**
2. * @brief SERVO 模式下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
3. * @param cutoffFreq 一阶低通滤波器截止频率
4. * @return ERR_SUCC 成功 其他失败
5. */
6. errno_t  servo_move_use_joint_LPF(double cutoffFreq);
```

4.85 机器人 SERVO 模式下关节空间非线性滤波

```
1. /**
2. * @brief SERVO 模式下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
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
4. * @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
5. * @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
6. * @return ERR_SUCC 成功 其他失败
7. */
8. errno_t  servo_move_use_joint_NLF(double max_vr, double max_ar, double max_jr);
```

4.86 机器人 SERVO 模式下笛卡尔空间非线性滤波

```
1. /**
2. * @brief SERVO 模式下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
3. * @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
4. * @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
5. * @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
6. * @param max_vr 笛卡尔空间姿态变化速度的速度上限值（绝对值）°/s
7. * @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);
```

4.87 机器人 SERVO 模式下关节空间多阶均值滤波

```
/**
 * @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_move_use_joint_MMF(int max_buf, double kp, double kv, double ka);
```

4.88 机器人 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_move_use_joint_MMF(int max_buf, double kp, double kv, double ka);
```

4.89 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);
```

4.90 设置传感器品牌

```
/**
* @brief 设置传感器品牌
* @param sensor_brand 传感器品牌，可选值为 1,2,3 分别代表不同品牌力矩传感器
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_torsenosr_brand(int sensor_brand);
```

4.91 获取传感器品牌

```
/**
* @brief 获取传感器品牌
* @param sensor_brand 传感器品牌，可选值为 1,2,3 分别代表不同品牌力矩传感器
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_torsenosr_brand(int* sensor_brand);
```

4.92 开启或关闭力矩传感器

```
/**
* @brief 开启或关闭力矩传感器
* @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);
```

4.93 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
 * @param 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
 * @param ftUser 阻尼力，表示用户用多大的力才能让机器人的沿着某个方向以最大速度进行运动
 * @param ftReboundFK 回弹力，表示机器人回到初始状态的能力
 * @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，
 * @return ERR_SUCC 成功 其他失败
 */
errno_t set_admit_ctrl_config(int axis, int opt, int ftUser, int ftConstant, int ftNnormal
Track, int ftReboundFK);
```

4.94 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);
```

4.95 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);
```

4.96 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);
```

4.97 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);
```

4.98 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);
```

4.99 力控拖拽使能

```
/**
* @brief 力控拖拽使能
* @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);
```

4.100 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 代表恒力柔顺控制,1 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);
```

4.101 获取力控类型和传感器初始化状态

```
/**
```

```
 * @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 代表恒力柔顺控制,1 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);
```

4.102 获取力控柔顺控制参数

```
/**
 * @brief 获取力控柔顺控制参数
 * @param admit_ctrl_cfg 机器人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);
```

4.103 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);
```

4.104 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 get_torque_sensor_comm(int* type, char* ip_addr,int* port);
```

4.105 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();
```

4.106 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);
```

4.107 设置柔顺控制力矩条件

```
/**
* @brief 设置柔顺控制力矩条件
* @param ft 为柔顺控制力矩条件
* @return ERR_SUCC 成功 其他失败
*/
errno_t set_compliance_condition(const FTxyz* ft);
```

4.108 设置网络异常时机器人自动终止运动类型

```
/**
* @brief 设置网络异常控制句柄，SDK 与机器人控制器失去连接后多长时间机器人控制器终止机械臂当前运动
```

```
* @param millisecond 时间参数，单位：ms
* @param mnt 网络异常时机器人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);
```

4.109 设置机器人状态数据自动更新时间间隔

```
/**
* @brief 设置机器人状态数据自动更新时间间隔
* @param handle  机器人控制句柄
* @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(const JKHD* handle, float millisecond);
```

5.接口使用示例

5.1 机器人设置与启动

```
1. #include "JAKAZuRobot.h"
2.
3. int main()
4. {
```

```
    //实例 API 对象 test
    JAKAZuRobot test;
    //登陆控制器，需要将 192.168.2.138 替换为自己控制器的 IP
    test.login_in("192.168.2.138");
    //机器人上电
    test.power_on();
    //机器人上使能
    test.enable_robot();
    //设置机器人运行倍率
    test.set_rapidrate(0.5);
    return 0;
}
```

5.2 异常回调函数的注册

```
#include "JAKAZuRobot.h"

void user_error_handle(int error_code)
{
    switch (error_code)
    {
    case 1:
    {
        std::cout << "case 1" << std::endl;
        break;
    }
    case 2:
    {
        std::cout << "case 2" << std::endl;
        break;
    }
    default:
    {
        std::cout << "case default" << std::endl;
        break;
    }
    }
}
int main()
{
    //实例 API 对象 test
    JAKAZuRobot test;
    //登陆控制器，需要将 192.168.2.138 替换为自己控制器的 IP
    test.login_in("192.168.2.138");
```

```
    //用户设置发生异常时的回调函数
    test.set_error_handler(user_error_handle);
    return 0;
}
```

5.3 机器人关节与笛卡尔空间运动

```
#include "JAKAZuRobot.h"

#define PI 3.1415926

int main()
{
    //实例 API 对象 test
    JAKAZuRobot test;
    //登陆控制器，需要将 192.168.2.138 替换为自己控制器的 IP
    test.login_in("192.168.2.138");
    //机器人上电
    demo.power_on();
    //机器人上使能
    demo.enable_robot();
    //定义并初始化关 JointValue 变量
     JointValue joint_pos = {90*PI/180,50*PI/180,-70*PI/180,20*PI/180,70*PI/180,11*PI/180};

    //关节空间运动，其中其中 ABS 代表绝对运动，TRUE 代表指令是阻塞的，1 代表速度为 1rad/s
    test.joint_move(&joint_pos, ABS, TRUE, 1);
    //定义并初始化 CartesianPose 变量
    CartesianPose cart;
    cart.tran.x = -115; cart.tran.y = 796; cart.tran.z = 358;
    cart.rpy.rx = 84; cart.rpy.ry = -12; cart.rpy.rz = 166;
    //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
    test.linear_move(&cart, ABS, FALSE, 10);
    return 0;
}
```

5.4 控制多台机器人

```
#include "JAKAZuRobot.h"

int main()
{
    //实例 API 对象 test
    JAKAZuRobot test[2];
```

```
7.      //登陆控制器，需要将 192.168.2.138 替换为自己控制器的 IP
8.      test[0].login_in("192.168.2.138");
9.      //登陆控制器，需要将 192.168.2.64 替换为自己控制器的 IP
10.     test[1].login_in("192.168.2.64");
11.     //机器人上电
12.     test[0].power_on();
13.     test[1].power_on();
14.     //机器人上使能
15.     test[0].enable_robot();
16.     test[1].enable_robot();
17.     return 0;
18. }
```

6.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。