JAKA 节卡

C 语言 SDK 使用手册

版本：V1.0.8

时间：2021.08.30

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	2020.5.27	V1.4.10/V2.0.10 及以上	创建
V1.0.1	2020.7.17	V1.4.10/V2.0.10 及以上	增加目录 3.20、3.21、4.63--4.66
V1.0.2	2020.9.27	V1.4.10/V2.0.10 及以上	增加目录 3.22、3.23、4.67--4.74
V1.0.3	2020.10.22	V1.4.10/V2.0.10 及以上	增加目录 3.24、4.75--4.77
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 4.78-4.82 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、4.14、4.16、4.85
V1.0.6	2020.01.18	V1.4.24/V2.0.24 及以上	修改目录 3.23、增加目录 4.35、4.39、4.88-4.99
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 4.100-4.109
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
V2.0.1	2021.12.10	V1.4.24/V1.5.12.17/V2.0.24 及以上	增加 ftp 接口

目录

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，其中函数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。
- Servo 模式下不能使用 joint_move、linear_move 等指令。

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
```

```
11.     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
12.     int emergency_stop;                                     ///< 机器人是否急停,0 为没有急停，1 为急停
13.     int dout[1024];                                         ///< 机器人控制柜数字输出信号,dout[0]为信号的个数
14.     int tio_dout[1024];                                     ///< 机器人末端工具数字输出信号,tio_dout[0]为信号的个数
15.     int extio[1024];                                        ///< 机器人外部应用数字输出信号,extio[0]为信号的个数
16.     int din[1024];                                          ///< 机器人控制柜数字输入信号,din[0]为信号的个数
17.     int tio_din[1024];                                      ///< 机器人末端工具数字输入信号,tio_din[0]为信号的个数
18.     double ain[1024];                                       ///< 机器人控制柜模拟输入信号,ain[0]为信号的个数
19.     double tio_ain[1024];                                   ///< 机器人末端工具模拟输入信号,tio_ain[0]为信号的个数
20.     double aout[1024];                                      ///< 机器人控制柜模拟输出信号,aout[0]为信号的个数
21.     unsigned int current_tool_id;                           ///< 机器人目前使用的工具坐标系 id
22.     double cartesiantran_position[6];                       ///< 机器人末端所在的笛卡尔空间位置
23.     double joint_position[6];                               ///< 机器人关节空间位置
24.     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
25.     unsigned int current_user_id;                           ///< 机器人目前使用的用户坐标系 id
26.     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
27.     RobotMonitorData robot_monitor_data;                    ///< 机器人状态监测数据
28.     TorqSensorMonitorData torq_sensor_monitor_data;         ///< 机器人力矩传感器状态监测数据
29.     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
30. }RobotStatus;
```

3.24 机器人错误码数据类型

```
1.  /**
2.  * @brief 机器人错误码数据类型
3.  */
```

```
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 创建机器人控制句柄

```
/**
* @brief 创建机器人控制句柄
* @param ip  控制器 ip 地址
* @param handle  机器人控制句柄
* @return ERR_SUCC 成功 其他失败
*/
errno_t  create_handler(const char* ip, JKHD* handle);
```

4.2 销毁机器人控制句柄

```
/**
* @brief 销毁机器人控制句柄
* @param handle  机器人控制句柄
* @return ERR_SUCC 成功 其他失败
*/
errno_t  destory_handler(const JKHD* handle);
```

4.3 机器人上电

```
/**
* @brief 打开机器人电源
* @param handle  机器人控制句柄
* @return ERR_SUCC 成功 其他失败
*/
errno_t  power_on(const JKHD* handle);
```

4.4 机器人下电

```
/**
* @brief 关闭机器人电源
* @param handle  机器人控制句柄
* @return ERR_SUCC 成功 其他失败
*/
errno_t  power_off(const JKHD* handle);
```

4.5 机器人关机

```
/**
* @brief 机器人控制柜关机
* @param handle  机器人控制句柄
* @return ERR_SUCC 成功 其他失败
*/
errno_t  shut_down(const JKHD* handle);
```

4.6 机器人上使能

```
/**
* @brief 控制机器人上使能
```

```
3.  * @param handle  机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  errno_t  enable_robot(const JKHD* handle);
```

4.7 机器人下使能

```
1.  /**
2.  * @brief 控制机器人下使能
3.  * @param handle  机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  errno_t  disable_robot(const JKHD* handle);
```

4.8 机器人手动模式下运动

```
1.  /**
2.  * @brief 控制机器人手动模式下运动
3.  * @param handle  机器人控制句柄
4.  * @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
5.  * @param move_mode 机器人运动模式，增量运动、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
6.  * @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 3.12 选择合适类型
7.  * @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
8.  * @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
9.  * @return ERR_SUCC 成功 其他失败
10. */
11. errno_t  jog(const JKHD* handle, int aj_num, MoveMode move_mode, CoordType coord_type, double vel_cmd, double pos_cmd);
```

代码示例：

```
1.  #include "jakaAPI.h"
2.  #include <stdio.h>
3.  #include <windows.h>
4.  #define PI 3.1415926
5.  char IP[20] = "192.168.1.105";
6.  //4.1~4.9 机器人手动运动
7.  int example_jog()
8.  {
9.      //实例 API 对象 demo
10.     JKHD demo;
11.     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
```

```
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //关节空间运动，其中 INCR 代表增量运动，0.5 代表速度为 0.5rad/s ,pi 代表执行该行命令运动 3.14rad,
    jog(&demo,1, INCR, COORD_JOINT, 0.5, 30 * PI / 180);
    Sleep(10000);
    jog_stop(&demo,1);
    return 0;
}
```

4.9 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 handle  机器人控制句柄
* @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(const JKHD* handle,int num);
```

4.10 机器人关节运动

```
/**
* @brief 机器人关节运动
* @param handle  机器人控制句柄
* @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、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 JKHD* handle, const JointValue* joint_pos, MoveMode move_mode,BOOL is_block, double speed);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
```

```
#define PI 3.1415926
char IP[20] = "192.168.1.105";
int example_joint_move()
{
    //实例 API 对象 demo
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     //定义并初始化关 JointValue 变量
     JointValue joint_pos = { 45 * PI / 180, 50 * PI / 180, 50 * PI / 180, 0 * PI / 180, 0 * PI / 180, 0 * PI / 180 };
     //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 1rad/s
     joint_move(&demo, &joint_pos, ABS, TRUE, 1);
     return 0;
}
```

4.11 机器人末端直线运动

```
/**
* @brief 机器人末端直线运动
* @param handle  机器人控制句柄
* @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、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
* 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 JKHD* handle, const CartesianPose* end_pos, MoveMode move_mode, BOOL is_block, double speed);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
```

```c
//4.11 机器人末端直线运动,注意避免奇异点
int example_linear_move()
{
    //实例 API 对象 demo
    JKHD demo;
    RobotStatus status;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //定义并初始化 CartesianPose 变量，旋转角为弧度。
    CartesianPose cart;
    cart.tran.x = 100; cart.tran.y = 200; cart.tran.z = 300;
    cart.rpy.rx = 120 * PI / 180; cart.rpy.ry = 90 * PI / 180; cart.rpy.rz = -90 * PI / 180;
    //笛卡尔空间运动，其中 ABS 代表绝对运动，TRUE 代表指令是阻塞的，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);
    linear_move(&demo ,&cart, ABS, TRUE, 10);
    get_robot_status(&demo ,&status);
    printf("errcode=%d \nx=%f, y=%f, z=%f\n", status.errcode, status.cartesiantran_position[0], status.cartesiantran_position[1], status.cartesiantran_position[2]);

    printf("rx=%f, ry=%f, rz=%f", status.cartesiantran_position[3], status.cartesiantran_position[4], status.cartesiantran_position[5]);
    Sleep(1000);
    return 0;
}
```

4.12 机器人圆弧运动

```c
/**
* @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)、绝对运动和连续运动
* @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 机器人末端运动加速度
* @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  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);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
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 = -90 * PI / 180; end_p.rpy.ry = 0 * PI / 180; end_p.rpy.rz = 0 * PI / 180;
    mid_p.tran.x = -300; mid_p.tran.y = 400; mid_p.tran.z = 500;
    mid_p.rpy.rx = -90 * PI / 180; mid_p.rpy.ry = 0 * PI / 180; mid_p.rpy.rz = 0 * PI / 180;
    //实例 API 对象 demo
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //定义并初始化关 JointValue 变量
    JointValue joint_pos = { 85.76 * PI / 180, -6.207 * PI / 180, 111.269 * PI / 180, 74.938 * PI / 180, 94.24 * PI / 180, 0 * PI / 180 };
    //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 1rad/s
    joint_move(&demo, &joint_pos, ABS, TRUE, 1);
    printf("circle start");
    //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 代表机器人终点误差，opt 为可选参数。
    circular_move(&demo, &end_p, &mid_p, ABS, TRUE, 20, 1, 0.1, &opt);
    printf("end");
    return 0;
}
```

4.13 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(const JKHD* handle);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//机器人运动终止
int example_motion_abort()
{
    //实例 API 对象 demo
     JKHD demo;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     //定义并初始化关 JointValue 变量
     printf("start_move\n");
     JointValue joint_pos = { 0 * PI / 180, 0 * PI / 180, 50 * PI / 180, 0 * PI / 180, 0 * PI / 180, 0 * PI / 180 };
     //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 1rad/s
     joint_move(&demo, &joint_pos, ABS, FALSE, 1);
     Sleep(500);
     //运动 0.5s 后终止
     motion_abort(&demo);
     printf("stop_move\n");
     return 0;
}
```

4.14 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 handle  机器人控制句柄
 * @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(const JKHD* handle, BOOL enable);
```

4.15 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 handle  机器人控制句柄
* @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、绝对运动
* @return ERR_SUCC 成功 其他失败
*/
errno_t  servo_j(const JKHD* handle, const JointValue* joint_pos, MoveMode move_mode);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//机器人运动终止
int example_motion_abort()
{
    //实例 API 对象 demo
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //定义并初始化关 JointValue 变量
    printf("start_move\n");
    JointValue joint_pos = { 0 * PI / 180, 0 * PI / 180, 50 * PI / 180, 0 * PI / 180, 0 * PI / 180, 0 * PI / 180 };
    //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 1rad/s
    joint_move(&demo, &joint_pos, ABS, FALSE, 1);
    Sleep(500);
    //运动 0.5s 后终止
    motion_abort(&demo);
    printf("stop_move\n");
    return 0;
}
```

4.16 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、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);
```

4.17 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 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
 * @param move_mode 指定运动模式：增量运动、绝对运动
 * @return ERR_SUCC 成功 其他失败
 */
errno_t    servo_p(const JKHD* handle, const CartesianPose* cartesian_pose, MoveMode move_mode);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
int example_servo_p()//机器人伺服笛卡尔空间运动
{
    //实例 API 对象 demo
    JKHD demo;
     //登陆控制器，需要将 192.168.2.105 替换为自己控制器的 IP
     login_in(&demo,"192.168.1.200");
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     //TRUE 为进入伺服模式
     servo_move_enable(&demo,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;
     printf("servo move start");
```

```
    for (int i = 0; i < 400; i++)
    {
        //笛卡尔空间伺服运动，其中 INCR 代表增量运动
        servo_p(&demo,&cart, INCR);
        Sleep(2);
    }
    //FALSE 为退出伺服模式
    servo_move_enable(&demo,FALSE);
    return 0;
}
```

4.18 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);
```

4.19 机器人 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(const JKHD* handle);
```

4.20 机器人 SERVO 模式下关节空间一阶低通滤波

```
/**
* @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
* @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(const JKHD* handle, double cutoffFreq);
```

代码示例：

```c
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 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
     JKHD demo;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     //servo 模式下关节空间一阶低通滤波,截止频率 0.5Hz
     ret = servo_move_use_joint_LPF(&demo, 0.5);
     return 0;
}
```

4.21 机器人 SERVO 模式下关节空间非线性滤波

```c
/**
* @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
* @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(const JKHD* handle, double max_vr, double max_ar, double max_jr);
```

代码示例：

```c
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 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
```

```
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //servo 模式下关节空间非线性滤波
    ret = servo_move_use_joint_NLF(&demo, 2, 2, 4);
    return 0;
}
```

4.22 机器人 SERVO 模式下笛卡尔空间非线性滤波

```
/**
* @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
* @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(const JKHD* handle, double max_vp, double max_ap, double
    max_jp, double max_vr, double max_ar, double max_jr);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//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
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
```

```
    //servo 模式下笛卡尔空间非线性滤波
    ret = servo_move_use_carte_NLF(&demo, 2, 2, 4, 2, 2, 4);
    return 0;
}
```

4.23 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(const JKHD* handle, int max_buf, double kp, double kv, double ka);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//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
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //servo 模式下关节空间多阶均值滤波
    ret = servo_move_use_joint_MMF(&demo, 20, 0.1, 0.2, 0.6);
    return 0;
}
```

4.24 机器人 SERVO 模式下速度前瞻参数设置

```
/**
```

```
 * @brief SERVO 模式下速度前瞻参数设置
 * @param handle  机器人控制句柄
 * @param max_buf 缓冲区的大小
 * @param kp 加速度滤波系数
 * @return ERR_SUCC 成功 其他失败
 */
 errno_t servo_speed_foresight(const JKHD* handle, int max_buf, double kp);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//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
     JKHD demo;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     //servo 模式下关节空间多阶均值滤波
     ret = servo_move_use_joint_MMF(&demo, 20, 0.1, 0.2, 0.6);
     return 0;
}
```

4.25 设置数字输出变量

```
/**
 * @brief 设置数字输出变量(DO)的值
 * @param handle 机器人控制句柄
 * @param type DO 类型
 * @param index DO 索引（从 1 开始）
 * @param value DO 设置值
 * @return ERR_SUCC 成功 其他失败
 */
errno_t  set_digital_output(const JKHD* handle, IOType type, int index, BOOL value)
;
```

代码示例：

```c
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
int example_set_digital_output()
{
    BOOL DO2;
    //实例 API 对象 demo
     JKHD demo;
     RobotStatus status;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     //查询 do2 的状态
     get_digital_output(&demo,IO_CABINET, 2, &DO2);
     printf("DO2 = %d\n", DO2);
     //io_cabinet 是控制柜面板 IO，2 代表 DO2，1 对应要设置的 DO 值。
     set_digital_output(&demo,IO_CABINET, 2, 1);
     Sleep(1000);//需要 window.h 延时 1s
     //查询 do2 的状态
     get_digital_output(&demo,IO_CABINET, 2, &DO3);
     printf("DO2 = %d\n", DO3);
     return 0;
}
```

4.26 设置模拟输出变量

```c
/**
* @brief 设置模拟输出变量的值(AO)的值
* @param handle 机器人控制句柄
* @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(const JKHD* handle, IOType type, int index, float value);
```

代码示例：

```c
#include "jakaAPI.h"
#include <stdio.h>
```

```
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//设置模拟输出
int example_set_analog_output()
{
    //实例 API 对象 demo
    JKHD demo;
    RobotStatus status;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    float AO35;
    //查询 do 的状态
    get_analog_output(&demo,IO_CABINET, 34, &AO35);
    printf("AO35 = %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。
    set_analog_output(&demo,IO_CABINET, 34, 1.5);
    Sleep(1000);//需要 window.h 延时 1s
    //查询 do 的状态
    get_analog_output(&demo,IO_CABINET, 34, &AO35);
    printf("AO35 = %f\n", AO35);
    return 0;
}
```

4.27 查询数字输入状态

```
/**
* @brief 查询数字输入(DI)状态
* @param handle 机器人控制句柄
* @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(const JKHD* handle, IOType type, int index, BOOL* result);
```

4.28 查询数字输出状态

```
/**
* @brief 查询数字输出(DO)状态
```

```
3.  * @param handle 机器人控制句柄
4.  * @param type DO 类型
5.  * @param index DO 索引 （从 0 开始）
6.  * @param result DO 状态查询结果
7.  * @return ERR_SUCC 成功 其他失败
8.  */
9.  errno_t  get_digital_output(const JKHD* handle, IOType type, int index, BOOL* result);
```

4.29 获取模拟量输入变量的值

```
1.  /**
2.  * @brief 获取模拟量输入变量(AI)的值
3.  * @param handle 机器人控制句柄
4.  * @param type AI 的类型
5.  * @param index AI 索引 （从 0 开始）
6.  * @param result 指定 AI 状态查询结果
7.  * @return ERR_SUCC 成功 其他失败
8.  */
9.  errno_t  get_analog_input(const JKHD* handle, IOType type, int index, float* result);
```

4.30 获取模拟量输出变量的值

```
1.  /**
2.  * @brief 获取模拟量输出变量(AO)的值
3.  * @param handle 机器人控制句柄
4.  * @param type AO 的类型
5.  * @param index AO 索引 （从 0 开始）
6.  * @param result 指定 AO 状态查询结果
7.  * @return ERR_SUCC 成功 其他失败
8.  */
9.  errno_t  get_analog_output(const JKHD* handle, IOType type, int index, float* result);
```

4.31 查询扩展 IO 是否运行

```
1.  /**
2.  * @brief 查询扩展 IO 模块是否运行
3.  * @param handle 机器人控制句柄
4.  * @param is_running 扩展 IO 模块运行状态查询结果
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  errno_t  is_extio_running(const JKHD* handle, BOOL* is_running);
```

代码示例：

```c
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//查询扩展 IO 状态
int example_is_extio_running()
{
    //实例 API 对象 demo
     JKHD demo;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     BOOL is_running;
     //查询 TIO 的状态
     is_extio_running(&demo, &is_running);
     printf("tio = %d\n", is_running);
     return 0;
}
```

4.32 运行当前加载的作业程序

```c
/**
* @brief 运行当前加载的作业程序
* @param handle  机器人控制句柄
* @return ERR_SUCC 成功 其他失败
*/
errno_t  program_run(const JKHD* handle);
```

代码示例：

```c
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//脚本加载，运行控制，过程查看
int example_program()
{
    char name[128];
     int cur_line;
     //实例 API 对象 demo
     JKHD demo;
     ProgramState pstatus;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
```

```
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //加载预先通过 app 编辑的 example 脚本
    program_load(&demo, "test");
    //获取已加载的程序名
    get_loaded_program(&demo,name);
    printf("Pro_name is :  %s\n", name);
    //运行当前加载的程序
    program_run(&demo);
    Sleep(1000);//需要 window.h 延时 1s
    //暂停当前运行的程序
    program_pause(&demo);
    //获取当前执行程序的行号
    get_current_line(&demo, &cur_line);
    printf("cur_line is : %d\n", cur_line);
    //获取当前程序状态
    get_program_state(&demo, &pstatus);
    printf("pro_status is : %d\n",pstatus);
    //继续运行当前程序
    program_resume(&demo);
    Sleep(10000);//需要 window.h 延时 10s
    //终止当前程序
    program_abort(&demo);
    return 0;
}
```

4.33 暂停当前运行的作业程序

```
/**
* @brief 暂停当前运行的作业程序
* @param handle  机器人控制句柄
* @return ERR_SUCC 成功 其他失败
*/
errno_t  program_pause(const JKHD* handle);
```

4.34 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
* @param handle  机器人控制句柄
```

```
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  errno_t  program_resume(const JKHD* handle);
```

4.35 终止当前执行的作业程序

```
1.  /**
2.  * @brief 终止当前执行的作业程序
3.  * @param handle  机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  errno_t  program_abort(const JKHD* handle);
```

4.36 加载指定的作业程序

```
1.  /**
2.  * @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/）
3.  * @param handle 机器人控制句柄
4.  * @param file 程序文件路径
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  errno_t  program_load(const JKHD* handle, const char* file);
```

4.37 获取已加载的作业程序的名字

```
1.  /**
2.  * @brief 获取已加载的作业程序名字
3.  * @param handle 机器人控制句柄
4.  * @param file 程序文件路径
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  errno_t  get_loaded_program(const JKHD* handle, char* file);
```

4.38 获取当前机器人作业程序的执行行号

```
1.  /**
2.  * @brief 获取当前机器人作业程序的执行行号
3.  * @param handle  机器人控制句柄
4.  * @param curr_line 当前行号查询结果
5.  * @return ERR_SUCC 成功 其他失败
6.  */
```

```
errno_t  get_current_line(const JKHD* handle, int* curr_line);
```

4.39 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 handle  机器人控制句柄
* @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(const JKHD* handle, ProgramState* status);
```

4.40 设置机器人的运行倍率

```
/**
* @brief 设置机器人运行倍率
* @param handle  机器人控制句柄
* @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(const JKHD* handle, double rapid_rate);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//机器人速度查看及调整
int example_rapidrate()
{
    double rapid_rate;
     //实例 API 对象 demo
     JKHD demo;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     get_rapidrate(&demo, &rapid_rate);
     printf("rapid_rate is : %f",rapid_rate);
     set_rapidrate(&demo, 0.4);
     Sleep(100);
```

```
    get_rapidrate(&demo, &rapid_rate);
    printf("rapid_rate is : %f", rapid_rate );
    return 0;
}
```

4.41 获取机器人的运行倍率

```
/**
* @brief 获取机器人运行倍率
* @param handle 机器人控制句柄
* @param rapid_rate 当前控制系统倍率
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_rapidrate(const JKHD* handle, double* rapid_rate);
```

4.42 设置工具信息

```
/**
* @brief 设置指定编号的工具信息
* @param handle 机器人控制句柄
* @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(const JKHD* handle, int id, const CartesianPose* tcp, const char* name);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";

char IP[20] = "192.168.1.105";
int example_tool()
{
    int id_ret, id_set;
     id_set = 2;
     CartesianPose tcp_ret, tcp_set;
     char name[50] = "test";
     //实例 API 对象 demo
     JKHD demo;
```

```
15.     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
16.     create_handler(IP, &demo);
17.     //机器人上电
18.     power_on(&demo);
19.     //查询当前使用的工具 ID
20.     get_tool_id(&demo, &id_ret);
21.     //获取当前使用的工具信息
22.     get_tool_data(&demo, id_ret, &tcp_ret);
23.     printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tr
    an.y, tcp_ret.tran.y);
24.     printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.r
    z);
25.     //初始化工具坐标
26.     tcp_set.tran.x = 0; tcp_set.tran.y = 0; tcp_set.tran.z = 10;
27.     tcp_set.rpy.rx = 120 * PI / 180; tcp_set.rpy.ry = 90 * PI / 180; tcp_set.rpy.
    rz = -90 * PI / 180;
28.     //设置工具信息
29.     set_tool_data(&demo, id_set, &tcp_set, name);
30.     //切换当前使用的工具坐标
31.     set_tool_id(&demo, id_set);
32.     Sleep(1000);
33.     //查询当前使用的工具 ID
34.     get_tool_id(&demo, &id_ret);
35.     //获取设置的工具信息
36.     get_tool_data(&demo, id_ret, &tcp_ret);
37.     printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tr
    an.y, tcp_ret.tran.y);
38.     printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.r
    z);
39.     return 0;
40. }
```

4.43 获取工具坐标信息

```
1.  /**
2.  * @brief 查询当前使用的工具信息
3.  * @param id 工具 ID
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(const JKHD* handle, int id, CartesianPose* tcp);
```

4.44 设置当前使用的工具 ID

```
/**
* @brief 设置当前使用的工具 ID
* @param handle  机器人控制句柄
* @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 JKHD* handle, const int id);
```

4.45 查询当前使用的工具 ID

```
/**
* @brief 查询当前使用的工具 ID
* @param handle  机器人控制句柄
* @param id 工具 ID 查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_tool_id(const JKHD* handle, int* id);
```

4.46 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 handle 机器人控制句柄
* @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(const JKHD* handle, int id, const CartesianPose* user_frame,
const char* name);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//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";
    //实例 API 对象 demo
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //查询当前使用的用户坐标系 ID
    get_user_frame_id(&demo, &id_ret);
    //获取当前使用的用户坐标系信息
    get_user_frame_data(&demo, id_ret, &tcp_ret);
    printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tr
an.y, tcp_ret.tran.y);
    printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.r
z);
    //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 * PI / 180; tcp_set.rpy.ry = 90 * PI / 180; tcp_set.rpy.
rz = -90 * PI / 180;
    //设置用户坐标系信息
    set_user_frame_data(&demo, id_set, &tcp_set, name);
    //切换当前使用的用户坐标系坐标
    set_user_frame_id(&demo, id_set);
    //查询当前使用的用户坐标系 ID
    get_user_frame_id(&demo, &id_ret);
    //获取设置的用户坐标系信息
    get_user_frame_data(&demo, id_ret, &tcp_ret);
    printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tr
an.y, tcp_ret.tran.y);
    printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.r
z);
    return 0;
}
```

4.47 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(const JKHD* handle, int id, CartesianPose* tcp);
```

4.48 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 handle  机器人控制句柄
* @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 JKHD* handle, const int id);
```

4.49 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 handle  机器人控制句柄
* @param id 获取的结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_user_frame_id(const JKHD* handle, int* id);
```

4.50 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 handle  机器人控制句柄
* @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(const JKHD* handle, BOOL enable);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//拖拽模式
int example_drag()
{
    BOOL in_drag;
```

```
    //实例 API 对象 demo
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //确认机器人是否在拖拽模式下
    is_in_drag_mode(&demo, &in_drag);
    printf("in_drag is : %d\n", in_drag);
    //使能拖拽模式
    drag_mode_enable(&demo, TRUE);
    Sleep(20000);
    is_in_drag_mode(&demo, &in_drag);
    printf("in_drag is : %d\n", in_drag);
    //去使能拖拽模式
    drag_mode_enable(&demo, FALSE);
    Sleep(20000);
    is_in_drag_mode(&demo, &in_drag);
    printf("in_drag is : %d\n",in_drag);
    while (1)
    {
        is_in_drag_mode(&demo, &in_drag);
        printf("in_drag is : %d\n" ,in_drag);
        Sleep(1000);
    }
    return 0;
}
```

4.51 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 handle  机器人控制句柄
* @param in_drag 查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  is_in_drag_mode(const JKHD* handle, BOOL* in_drag);
```

4.52 获取机器人状态

```
/**
```

```
 * @brief 获取机器人状态
 * @param handle  机器人控制句柄
 * @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(const JKHD* handle, RobotState* state);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//获取机器人状态(急停 上电 伺服使能)
int example_get_robstate()
{
    //实例 API 对象 demo
    JKHD demo;
    RobotState state;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    get_robot_state(&demo, &state);
    printf( "is servoEnabled : %d", state.servoEnabled);
    return 0;
}
```

4.53 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 handle  机器人控制句柄
 * @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(const JKHD* handle, CartesianPose* tcp_position);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
```

```
//获取工具末端位姿
int example_get_tcp_position()
{
    //实例 API 对象 demo
     JKHD 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
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     //获取工具末端位姿
     get_tcp_position(&demo ,&tcp_pos);
     return 0;
}
```

4.54 获取当前机器人关节角度

```
/**
* @brief 获取当前机器人关节角度
* @param handle  机器人控制句柄
* @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(const JKHD* handle, JointValue* joint_position);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//获取当前机器人关节角
int example_get_joint_position()
{
    //实例 API 对象 demo
     JKHD 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
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
```

```
    //获取当前关节角
    get_joint_position(&demo, &jot_pos);
    return 0;
}
```

4.55 查询机器人是否超出限位

```
/**
* @brief 查询机器人是否超出限位
* @param handle  机器人控制句柄
* @param on_limit 查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  is_on_limit(const JKHD* handle, BOOL* on_limit);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//查询是否超出限位
int example_is_on_limit()
{
    //实例 API 对象 demo
    JKHD demo;
    BOOL on_limit;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    while (1)
    {
        //查询是否超限
        is_on_limit(&demo, &on_limit);
        printf("on_limit is :%d\s",on_limit );
        Sleep(1000);
    }
    return 0;
}
```

4.56 查询机器人运动是否停止

```
/**
* @brief 查询机器人运动是否停止
* @param handle  机器人控制句柄
* @param in_pos 查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  is_in_pos(const JKHD* handle, BOOL* in_pos);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//4.50 查询机器人运动是否停止
int example_is_in_pos()
{
    //实例 API 对象 demo
    JKHD 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
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    while (1)
    {
        //查询是否运动停止
        is_in_pos(&demo, &in_pos);
        printf(" in_pos is :%d",in_pos);
        Sleep(200);
    }
    return 0;
}
```

4.57 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 handle  机器人控制句柄
* @param in_collision 查询结果
```

```
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  errno_t  is_in_collision(const JKHD* handle, BOOL* in_collision);
```

4.58 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.  * @param handle  机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  errno_t  collision_recover(const JKHD* handle);
```

代码示例：

```
1.  #include "jakaAPI.h"
2.  #include <stdio.h>
3.  #include <windows.h>
4.  #define PI 3.1415926
5.  char IP[20] = "192.168.1.105";
6.  int example_collision_recover()
7.  {
8.      //实例 API 对象 demo
9.      JKHD demo;
10.      BOOL in_collision;
11.      //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
12.      create_handler(IP, &demo);
13.      //机器人上电
14.      power_on(&demo);
15.      //机器人上使能
16.      enable_robot(&demo);
17.      //查询是否处于碰撞保护状态
18.      is_in_collision(&demo, &in_collision);
19.      if (in_collision)
20.      //如果处于碰撞保护模式，则从碰撞保护中恢复
21.      {
22.          printf("robot is in collision");
23.          collision_recover(&demo);
24.      }
25.      else
26.      {
27.          printf("robot is not collision");
28.      }
29.      return 0;
30.  }
```

4.59 设置机器人碰撞等级

```
/**
* @brief 设置机器人碰撞等级
* @param handle  机器人控制句柄
* @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 JKHD* handle, const int level);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//4.52~53 碰撞等级查看，设置
int example_collision_level()
{
    //实例 API 对象 demo
    JKHD 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
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //查询当前碰撞等级
    get_collision_level(&demo, &level);
    printf(" collision level is :%d",level );
    //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，
    set_collision_level(&demo, 1);
    //查询当前碰撞等级
    get_collision_level(&demo, &level);
    printf( " collision level is :%d",level);
    return 0;
}
```

4.60 获取机器设置的碰撞等级

```
/**
```

```
 * @brief 获取机器人设置的碰撞等级
 * @param handle  机器人控制句柄
 * @return ERR_SUCC 成功 其他失败
 */
errno_t get_collision_level(const JKHD* handle, int* level);
```

4.61 错误状态清除

```
/**
 * @brief  错误状态清除
 * @param handle  机器人控制句柄
 * @return ERR_SUCC 成功 其他失败
 */
 DLLEXPORT_API errno_t clear_error(const JKHD *handle);
```

4.62 机器人求解逆解

```
/**
 * @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 handle  机器人控制句柄
 * @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 JKHD* handle, const JointValue* ref_pos, const CartesianPose*
cartesian_pose, JointValue* joint_pos);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 机器人逆解 已知 tcp_pos,求 joint_pos
int example_kine_inverse()
{
    int ret;
     //实例 API 对象 demo
     JKHD demo;
     create_handler(IP, &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 };
    ret = kine_inverse(&demo, &ref_jpos, &tcp_pos, &joint_pos);
    for (int i = 0; i < 6; i++)
    {
        printf("joint_pos[%d] is : %d \n", i, joint_pos.jVal[i]);
    }
    return 0;
}
```

4.63 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 handle  机器人控制句柄
* @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 JKHD* handle, const JointValue* joint_pos, CartesianPose* cartesian_pose);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 机器人正解 已知 joint_pos,求 tcp_pos
int example_kine_forward()
{
    int ret;
    //实例 API 对象 demo
    JKHD demo;
    create_handler(IP, &demo);
    //初始化笛卡尔空间点坐标
    CartesianPose tcp_pos;
    //初始化返回值
    JointValue joint_pos = { 0.558, 0.872, 0.872 , 0.349, 0.191, 0.191 };
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
```

```
    ret = kine_forward(&demo, &joint_pos, &tcp_pos);
    printf("tcp_pos is :\n x:%d  y:%d  z:%d rx:%d  ry:%d  rz:%d", tcp_pos.tran.x, tcp_pos.tran.y, tcp_pos.tran.z, tcp_pos.rpy.rx, tcp_pos.rpy.ry, tcp_pos.rpy.rz);

    return 0;
}
```

4.64 欧拉角到旋转矩阵的转换

```
/**
* @brief 欧拉角到旋转矩阵的转换
* @param handle  机器人控制句柄
* @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 JKHD* handle, const Rpy* rpy, RotMatrix* rot_matrix);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 欧拉角到旋转矩阵
int example_rpy_to_rot_matrix()
{
    int ret;
    //实例 API 对象 demo
    JKHD demo;
    //初始化欧拉角
    Rpy rpy;
    rpy.rx = -1.81826; rpy.ry = -0.834253;  rpy.rz = -2.30243;
    create_handler(IP, &demo);
    //初始化旋转矩阵
    RotMatrix rot_matrix;
    ret = rpy_to_rot_matrix(&demo, &rpy, &rot_matrix);
    printf("     eul2rotm");
    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;
}
```

4.65 旋转矩阵到欧拉角的转换

```
/**
* @brief 旋转矩阵到欧拉角的转换
* @param handle  机器人控制句柄
* @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 JKHD* handle, const RotMatrix* rot_matrix, Rpy* rpy);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 旋转矩阵--->欧拉角
int example_rot_matrix_to_rpy()
{
    int ret;
     //实例 API 对象 demo
     JKHD 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
     create_handler(IP, &demo);
     ret = rot_matrix_to_rpy(&demo, &rot_matrix, &rpy);
     printf(" %d    rotm2eul:",ret);
     printf("%f %f %f \n", rpy.rx, rpy.ry, rpy.rz);
     return 0;
}
```

4.66 四元数到旋转矩阵的转换

```
/**
```

```
 * @brief 四元数到旋转矩阵的转换
 * @param handle  机器人控制句柄
 * @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 JKHD* handle, const Quaternion* quaternion, RotMatrix* rot_matrix);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 四元数-->旋转矩阵
int example_quaternion_to_rot_matrix()
{
    int ret;
    //实例 API 对象 demo
    JKHD 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;
    //quat.s = 0.707;  quat.x = 0; quat.y = 0.707107;  quat.z = 0;
    //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
    create_handler(IP, &demo);
    ret = quaternion_to_rot_matrix(&demo, &quat, &rot_matrix);
    printf(" %d    quatl2rotm:",ret);
    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;
}
```

4.67 旋转矩阵到四元数的转换

```
/**
 * @brief 旋转矩阵到四元数的转换
 * @param handle  机器人控制句柄
 * @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 JKHD* handle, const RotMatrix* rot_matrix, Quaternion* quaternion);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 旋转矩阵--->四元数
int example_rot_matrix_to_quaternion()
{
    int ret;
     //实例 API 对象 demo
     JKHD 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
     create_handler(IP, &demo);
     ret = rot_matrix_to_quaternion(&demo, &rot_matrix, &quat);
     printf(" %d   rotm2quat:",ret);
     printf("%lf %lf %lf %lf \n", quat.s, quat.x, quat.y, quat.z);
     return 0;
}
```

4.68 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 handle  机器人控制句柄
* @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(const JKHD* handle, CallBackFuncType func);
```

代码示例：

```c
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
char IP[20] = "192.168.1.105";
//注册机器人出错函数
//错误处理
void user_error_handle(int error_code)
{
    printf("%d",error_code);
}
//注册
int example_set_err_handle()
{
    //实例 API 对象 demo
    JKHD demo;
    //登陆控制器，需要将 192.168.2.229 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //设置用户异常回调函数
    set_error_handler(&demo, user_error_handle);
    while (1)
    {
    }
    return 0;
}
```

4.69 机器人负载设置

```c
/**
* @brief 机器人负载设置
* @param handle  机器人控制句柄
* @param payload 负载质心、质量数据
* @return ERR_SUCC 成功 其他失败
*/
errno_t  set_payload(const JKHD* handle, const PayLoad* payload);
```

代码示例：

```c
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
```

```
#define PI 3.1415926
char IP[20] = "192.168.1.105";
int example_payload()
{
    //实例 API 对象 demo
    JKHD demo;
     PayLoad payloadret;
     PayLoad payload_set;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //查询当前负载数据
     get_payload(&demo, &payloadret);
     printf(" payload mass is : %f kg\n",payloadret.mass);
     printf(" payload center of mass is \nx: %f  y: %f z: %f \n", payloadret.centroid.x , payloadret.centroid.y ,payloadret.centroid.z);
     payload_set.mass = 1.0;
     //单位 mm
     payload_set.centroid.x = 0; payload_set.centroid.y = 0; payload_set.centroid.z = 10;
     //设置当前负载数据
     set_payload(&demo, &payload_set);
     //查询当前负载数据
     get_payload(&demo, &payloadret);
     printf(" payload mass is : %f kg\n", payloadret.mass);
     printf(" payload center of mass is \nx: %f  y: %f z: %f \n", payloadret.centroid.x, payloadret.centroid.y, payloadret.centroid.z);
     return 0;
}
```

4.70 获取机器人负载数据

```
/**
* @brief 获取机器人负载数据
* @param handle  机器人控制句柄
* @param payload 负载查询结果
* @return ERR_SUCC 成功 其他失败
*/
errno_t  get_payload(const JKHD* handle, PayLoad* payload);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
```

```
//负载查看及设置
int example_payload()
{
    //实例 API 对象 demo
     JKHD demo;
     PayLoad payloadret;
     PayLoad payload_set;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //查询当前负载数据
     get_payload(&demo, &payloadret);
     printf(" payload mass is : %f kg\n",payloadret.mass);
     printf(" payload center of mass is \nx: %f  y: %f z: %f \n", payloadret.centroid.x , payloadret.centroid.y ,payloadret.centroid.z);
     payload_set.mass = 1.0;
     //单位 mm
     payload_set.centroid.x = 0; payload_set.centroid.y = 0; payload_set.centroid.z = 10;
     //设置当前负载数据
     set_payload(&demo, &payload_set);
     //查询当前负载数据
     get_payload(&demo, &payloadret);
     printf(" payload mass is : %f kg\n", payloadret.mass);
     printf(" payload center of mass is \nx: %f  y: %f z: %f \n", payloadret.centroid.x, payloadret.centroid.y, payloadret.centroid.z);
     return 0;
}
```

4.71 获取 SDK 版本号

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

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//获取 sdk 版本号
```

```
int example_getsdk_version()
{
    //实例 API 对象 demo
    JKHD 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
    create_handler(IP, &demo);
    //查询当前 SDK 版本
    get_sdk_version(&demo, ver);
    printf(" SDK version is :%s\n", ver );
    return 0;
}
}
```

4.72 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);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//获取控制器 ip
int example_get_controller_ip()
{
    int ret;
    //实例 API 对象 demo
    JKHD demo;
    char ip_list[2000] = "";
    char  controller_name1[50] = "";
    //获取控制器 ip
    ret = get_controller_ip(controller_name1, ip_list);
    printf(" ip_list is :\n %s" ,ip_list);

    return 0;
```

```
}
```

4.73 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(const JKHD* handle, RobotStatus* status);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//获取机器人状态监测数据
int example_get_robot_status()
{
    //实例 API 对象 demo
    JKHD 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
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //获取机器人状态监测数据
    get_robot_status(&demo, &robstatus);
    disable_robot(&demo);
    power_off(&demo);
    destory_handler(&demo);
    return 0;
}
```

4.74 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
```

```
3.  * @return ERR_SUCC 成功 其他失败
4.  */
5.  errno_t  set_debug_mode(const JKHD* handle, BOOL mode);
```

代码示例：

```
1.  #include "jakaAPI.h"
2.  #include <stdio.h>
3.  #include <windows.h>
4.  #define PI 3.1415926
5.  char IP[20] = "192.168.1.105";
6.  //设置 SDK 是否开启调试模式
7.  int example_set_debug_mode()
8.  {
9.      BOOL mode;
10.     //实例 API 对象 demo
11.     JKHD demo;
12.     printf("debug :true");
13.     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
14.     create_handler(IP, &demo);
15.     //设置调试模式
16.     set_debug_mode(&demo, TRUE);
17.     //机器人上电
18.     power_on(&demo);
19.     //机器人上使能
20.     enable_robot(&demo);
21.     return 0;
22. }
```

4.75 设置机器人错误码文件存放路径

```
1.  /**
2.  * @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
3.  * @return ERR_SUCC 成功 其他失败
4.  */
5.  errno_t set_errorcode_file_path(const JKHD* handle, char* path);
```

代码示例：

```
1.  #include "jakaAPI.h"
2.  #include <stdio.h>
3.  #include <windows.h>
4.  #define PI 3.1415926
5.  char IP[20] = "192.168.1.105";
6.  //错误码查看
7.  int example_get_last_errcode()
8.  {
```

```
    int ret;
    char path[100] = "E:\\JAKA_ERROR_CODE.csv";
    //实例 API 对象 demo
    JKHD demo;
    ErrorCode Eret;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    ret = program_load(&demo, "not_exist999875");//故意加载一个不存在的程序,引发报错。
    get_last_error(&demo, &Eret);//查询最后一个报错信息
    printf(" error code is : %x \nmessage:%s " ,Eret.code, Eret.message);
    ret = set_errorcode_file_path(&demo, path);//设置错误码说明文件
    get_last_error(&demo, &Eret);//查询最后一个报错信息
    printf(" error code is : %x \nmessage:%s ", Eret.code, Eret.message);
    return 0;
}
```

4.76 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(const JKHD* handle, ErrorCode* code);
```

4.77 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 JKHD* handle, const TrajTrackPara* para);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//轨迹复现参数查看及设置
```

```
int example_traj_config()
{
    //实例 API 对象 demo
    JKHD demo;
    TrajTrackPara trajpar_read;
    TrajTrackPara trajpar_set;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //查询当前轨迹复现参数
    get_traj_config(&demo, &trajpar_read);
    printf(" trajTrackPara is :\n xyz interval:%f  rpy interval is :%f  vel:%f  acc:%f ", trajpar_read.xyz_interval, trajpar_read.rpy_interval, trajpar_read.vel, trajpar_read.acc);
    //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;
    set_traj_config(&demo, &trajpar_set);

    //查询当前轨迹复现参数
    get_traj_config(&demo, &trajpar_read);
    printf(" trajTrackPara is :\n xyz interval:%f  rpy interval is :%f  vel:%f  acc:%f ", trajpar_read.xyz_interval, trajpar_read.rpy_interval, trajpar_read.vel, trajpar_read.acc);
    return 0;
}
```

4.78 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(const JKHD* handle, TrajTrackPara* para);
```

4.79 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 JKHD* handle, const BOOL mode,char* filename);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//轨迹采集开关与状态查询
int example_traj_sample()
{
    BOOL samp_stu;
     //实例 API 对象 demo
     JKHD demo;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     char name[20] = "ddd";
     //开启轨迹复现数据采集开关
     set_traj_sample_mode(&demo, TRUE, name);
     printf("start\n");
     drag_mode_enable(&demo, TRUE);
     //查询轨迹复现采集状态
     get_traj_sample_status(&demo, &samp_stu);

     printf("status:%d\n",samp_stu);
     Sleep(10000);
     printf("end\n");
     drag_mode_enable(&demo, FALSE);
     set_traj_sample_mode(&demo, FALSE, name);
     return 0;
}
```

4.80 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(const JKHD* handle, BOOL* sample_status);
```

4.81 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(const JKHD* handle, MultStrStorType* filename);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//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()
{
    //实例 API 对象 demo
    JKHD 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
    create_handler(IP, &demo);

    //查询当前轨迹文件名。
    get_exist_traj_file_name(&demo, &traj_file);
    printf( "file nums :%d" , traj_file.len);
    for (int i = 0; i < traj_file.len; i++)
        printf("%s\n",traj_file.name[i]);
    return 0;
}
```

4.82 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 JKHD* handle, const char* src,const char* dest);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
```

```
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//重命名轨迹复现的数据文件名
int example_rename_traj_file_name()
{
    //实例 API 对象 demo
    JKHD demo;
    MultStrStorType traj_file;
    char name_new[20] = "555";
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //查询当前轨迹文件名。
    get_exist_traj_file_name(&demo, &traj_file);
    printf("file nums :%d", traj_file.len);
    for (int i = 0; i < traj_file.len; i++)
        printf("%s\n", traj_file.name[i]);
    //重命名轨迹复现的数据文件名
    rename_traj_file_name(&demo, traj_file.name[0], name_new);
    //查询当前轨迹文件名。
    get_exist_traj_file_name(&demo, &traj_file);
    printf("file nums :%d", traj_file.len);
    for (int i = 0; i < traj_file.len; i++)
        printf("%s\n", traj_file.name[i]);
    return 0;
}
```

4.83 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 JKHD* handle, const char* filename);
```

4.84 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
*/
```

```
6. errno_t  generate_traj_exe_file(const JKHD* handle, const char* filename);
```

4.85 设置 SDK 日志路径

```
1. /**
2. * @brief 设置 SDK 日志路径
3. * @param filepath SDK 日志路径
4. * @return ERR_SUCC 成功 其他失败
5. */
6. errno_t  set_SDK_filepath(const JKHD* handle, char* filepath);
```

代码示例：

```
1. #include "jakaAPI.h"
2. #include <stdio.h>
3. #include <windows.h>
4. #define PI 3.1415926
5. char IP[20] = "192.168.1.105";
6. //设置 SDK 日志路径
7. int example_set_SDK_filepath()
8. {
9.     char path[20] = "E://";
10.     int ret;
11.     //实例 API 对象 demo
12.     JKHD demo;
13.     printf("login \n");
14.     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
15.     create_handler(IP, &demo);
16.     printf("login complete \n");
17.     ret = set_SDK_filepath(&demo, path);//设置 SDK 文件路径
18.     printf("%d\npower on\n", ret);
19.     //机器人上电
20.     power_on(&demo);
21.     //机器人上使能
22.     enable_robot(&demo);
23.     return 0;
24. }
```

4.86 设置传感器品牌

```
1. /**
2. * @brief 设置传感器品牌
3. * @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
4. * @return ERR_SUCC 成功 其他失败
```

```
*/
errno_t set_torsenosr_brand(const JKHD* handle, int sensor_brand);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 设置传感器品牌
int example_set_torsensor_brand()
{
    int ret;
     //实例 API 对象 demo
     JKHD demo;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     //设置传感器品牌
     ret = set_torsenosr_brand(&demo, 1);
     return 0;
}
```

4.87 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(const JKHD* handle, int* sensor_brand);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 获取传感器品牌
int example_get_torsensor_brand()
{
    int ret, cur_sensor;
```

```
    //实例 API 对象 demo
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo );
    //机器人上使能
    enable_robot(&demo);
    //获取传感器品牌
    ret = get_torsenosr_brand(&demo, &cur_sensor);
    printf("cur_sensor is :%d", cur_sensor);
    return 0;
}
```

4.88 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(const JKHD* handle, int sensor_mode);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 开启或关闭力矩传感器
int example_set_torque_sensor_mode()
{
    int ret;
    //实例 API 对象 demo
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    printf("enable finish");
    //设置力矩传感器状态，1 打开，0 关闭
    ret = set_torque_sensor_mode(&demo, 1);
```

```
    return 0;
}
```

4.89 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(const JKHD* handle, int axis, int opt, double ftUser, int ftConstant, double ftNnormalTrack, double ftReboundFK);
```

4.90 开始辨识工具末端负载

```
/**
* @brief 开始辨识工具末端负载，需要前置开启并初始化力控传感器
* @param joint_pos 使用力矩传感器进行自动负载辨识时的结束位置，使 j4j5j6 三轴旋转 90 度左右
* @return ERR_SUCC 成功 其他失败
*/
errno_t start_torq_sensor_payload_identify(const JKHD* handle, const JointValue* joint_pos);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
//4.94~4.98 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;
     //实例 API 对象 demo
     JKHD demo;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
```

```
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    printf("enable finish\n");
    //开始辨识传感器负载
    ret = start_torq_sensor_payload_identify(&demo, &joint_pos);
    do
    {
        //查询传感器负载状态
        get_torq_sensor_identify_staus(&demo, &ret);
        printf("%d\n",ret);
        Sleep(1000);
    } while (1 == ret);
    //获取辨识结果
    ret = get_torq_sensor_payload_identify_result(&demo, &pl);
    //设置传感器末端负载
    ret = set_torq_sensor_tool_payload(&demo, &pl);
    Sleep(10000);
    //获取当前设置的传感器末端负载
    ret = get_torq_sensor_tool_payload(&demo, &pl_ret);
    Sleep(10000);
    return 0;
}
```

4.91 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(const JKHD* handle, int* identify_status);
```

4.92 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(const JKHD* handle, PayLoad* payload);
```

4.93 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 JKHD* handle, const PayLoad* payload);
```

4.94 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(const JKHD* handle, PayLoad* payload);
```

4.95 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 JKHD* handle, const int ftFrame);
```

4.96 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(const JKHD* handle, int* ftFrame);
```

4.97 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 JKHD* handle, const int enable_flag);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 力控导纳控制使能
int example_enable_admittance_ctrl()
{
    int ret;
     //实例API对象demo
     JKHD demo;
     //登陆控制器，需要将192.168.2.105替换为自己控制器的IP
     login_in(&demo,"10.5.5.100");
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     //品牌
     set_torsenosr_brand(&demo,2);
     //开启力控传感器
     set_torque_sensor_mode(&demo,1);
     //初始化传感器
     set_compliant_type(&demo,1, 1);
     printf("inint sensor comple\n");
     //设置柔顺控制参数
     ret = set_admit_ctrl_config(&demo,0, 0, 20, 5, 0, 0);
     ret = set_admit_ctrl_config(&demo, 1, 0, 20, 5, 0, 0);
     ret = set_admit_ctrl_config(&demo, 2, 2, 20, 5, 0, 0);
     ret = set_admit_ctrl_config(&demo, 3, 0, 20, 5, 0, 0);
     ret = set_admit_ctrl_config(&demo, 4, 0, 20, 5, 0, 0);
     ret = set_admit_ctrl_config(&demo, 5, 0, 20, 5, 0, 0);
     //设置力控拖拽使能，1打开，0关闭
     ret = enable_admittance_ctrl(&demo,1);
     printf("enable_admittance_ctrl open! \n");
     printf("input any word to quit:\n");
     scanf("%d", ret);
     ret = enable_admittance_ctrl(&demo,0);
     ret = set_admit_ctrl_config(&demo,2, 0, 20, 5, 0, 0);
```

```
    set_torque_sensor_mode(&demo,0);
    printf("close\n");
    return 0;
}
```

4.98 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(const JKHD* handle, int sensor_compensation, int compliance_type);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 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
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //设置力控类型和传感器初始化状态
    ret = set_compliant_type(&demo, 1, 1);
    Sleep(1000);
    ret = get_compliant_type(&demo, &sensor_compensation, &compliance_type);
    Sleep(1000);
    return 0;
}
```

4.99 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(const JKHD* handle, int* sensor_compensation, int* compliance_type);
```

4.100 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(const JKHD* handle, RobotAdmitCtrl *admit_ctrl_cfg);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 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
     JKHD demo;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo );
     //机器人上使能
     enable_robot(&demo);
     //获取力控柔顺控制参数
     ret = get_admit_ctrl_config(&demo, &adm_ctr_cfg);
     return 0;
}
```

4.101 设置力控传感器通信参数

```
/**
 * @brief 设置力控传感器 ip 地址
 * @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 JKHD* handle,const int type, const char* ip_addr, const int port);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 设置与获取力控传感器通信参数
int example_torque_sensor_comm()
{
    char ip_set[20] = "192.168.2.14";
    int ret, type_set = 0, port_set = 10000;
    char ip_ret[20];
    int type_ret, port_ret;
    //实例 API 对象 demo
    JKHD demo;
    //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
    create_handler(IP, &demo);
    //机器人上电 f
    power_on(&demo);
    //机器人上使能
    enable_robot(&demo);
    //设置力控通信参数
    ret = set_torque_sensor_comm(&demo, type_set, ip_set, port_set);
    Sleep(1000);
    //获取力控通讯参数
    ret = get_torque_sensor_comm(&demo, &type_ret, ip_ret, &port_ret);
    return 0;
}
```

4.102 获取力控传感器 IP 地址

```
/**
```

```
2.  * @brief 获取力控传感器 ip 地址
3.  * @param type 通信类型，0 为使用 tcp/ip 协议，1 为使用 RS485 协议
4.  * @param ip_addr 为力控传感器地址
5.  * @param port 为使用 tcp/ip 协议时力控传感器端口号
6.  * @return ERR_SUCC 成功 其他失败
7.  */
8.  errno_t get_torque_sensor_comm(const JKHD* handle,int* type, char* ip_addr,int* port);
```

4.103 关闭力矩控制

```
1.  /**
2.  * @brief 关闭力矩控制
3.  * @return ERR_SUCC 成功 其他失败
4.  */
5.  errno_t disable_force_control();
```

4.104 设置速度柔顺控制参数

```
1.  /**
2.  * @brief 设置速度柔顺控制参数
3.  * @param vel_cfg 为速度柔顺控制参数
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  errno_t set_vel_compliant_ctrl(const JKHD* handle,const VelCom* vel_cfg);
```

4.105 设置柔顺控制力矩条件

```
1.  /**
2.  * @brief 设置柔顺控制力矩条件
3.  * @param ft 为柔顺控制力矩条件
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  errno_t set_compliance_condition(const JKHD* handle,const FTxyz* ft);
```

4.106 设置网络异常时机器人自动终止运动类型

```
1.  /**
2.  * @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
3.  * @param millisecond 时间参数，单位：ms
4.  * @param mnt 网络异常时机器人需要进行的动作类型
5.  * @return ERR_SUCC 成功 其他失败
```

```
*/
errno_t set_network_exception_handle(const JKHD* handle,float millisecond, ProcessType mnt);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 设置网络异常时机器人自动终止运动类型
int example_set_network_exception_handle()
{
    float milisec = 100;
     int ret;
     //实例 API 对象 demo
     JKHD demo;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     //设置柔顺力矩条件
     ret = set_network_exception_handle(&demo, milisec, MOT_KEEP);
     return 0;
}
```

4.107 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);
```

代码示例：

```
#include "jakaAPI.h"
#include <stdio.h>
#include <windows.h>
#define PI 3.1415926
char IP[20] = "192.168.1.105";
// 设置状态自动更新时间
```

```
int example_set_status_data_updata_interval()
{
    float milisec = 20.0;
     int ret;
     //实例 API 对象 demo
     JKHD demo;
     //登陆控制器，需要将 192.168.2.194 替换为自己控制器的 IP
     create_handler(IP, &demo);
     //机器人上电
     power_on(&demo);
     //机器人上使能
     enable_robot(&demo);
     //设置柔顺力矩条件
     ret = set_status_data_update_time_interval(&demo, milisec);
     printf("set %f milisec", milisec);
     return 0;
}
```

4.108 初始化 FTP 客户端

```
/**
* @brief 初始化 ftp 客户端，与控制柜建立连接，可导出 program、track
* @return ERR_SUCC 成功 其他失败
*/
errno_t init_ftp_client(const JKHD* handle,);
```

4.109 FTP 上传

```
/**
* @brief 从本地上传指定类型和名称的文件到控制器
* @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
* @param local 本地文件名绝对路径
* @param opt 1 单个文件 2 文件夹
* @return ERR_SUCC 成功 其他失败
*/
errno_t upload_file(const JKHD* handle,char* local, char* remote, int opt);
```

4.110 FTP 下载

```
/**
* @brief 从控制器下载指定类型和名称的文件到本地
```

```
* @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
* @param local 下载到本地文件名绝对路径
* @param opt 1 单个文件 2 文件夹
* @return ERR_SUCC 成功 其他失败
*/
errno_t download_file(const JKHD* handle,char* local, char* remote, int opt);
```

4.111 FTP 目录查询

```
/**
* @brief 重命名控制器指定类型和名称的文件
* @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/”
* @param des 重命名的目标名
* @param opt 1 单个文件 2 文件夹
* @return ERR_SUCC 成功 其他失败
*/
errno_t get_ftp_dir(const JKHD* handle,const char* remotedir, int type, char* ret);
```

4.112 FTP 删除

```
/**
* @brief 从控制器删除指定类型和名称的文件
* @param remote 控制器内部文件名
* @param opt 1 单个文件 2 文件夹
* @return ERR_SUCC 成功 其他失败
*/
errno_t del_ftp_file(const JKHD* handle,char* remote, int opt);
```

4.113 FTP 重命名

```
/**
* @brief 重命名控制器指定类型和名称的文件
* @param remote 控制器内部文件名原名称
* @param des 重命名的目标名
* @param opt 1 单个文件 2 文件夹
* @return ERR_SUCC 成功 其他失败
*/
errno_t rename_ftp_file(const JKHD* handle,char* remote, char* des, int opt);
```

4.114 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.          */
errno_t close_ftp_client(const JKHD* handle,);
```

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。