

# C#函数使用手册

版本：V1.0.8

时间：2020.08.23

## 产权说明

上海节卡机器人有限公司 版权所有。

上海节卡机器人有限公司对本文档中介绍的产品所包含的相关技术拥有知识产权。

本文档及相关产品按照限制其使用、复制、分发和反编译的许可证进行分发。未经上海节卡机器人有限公司事先书面授权，不得以任何方式、任何形式复制本产品或本文档的任何部分。

## 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--4.66 部分
V1.0.2	2020.09.27	V1.4.10/V2.0.10 及以上	增加目录 3.19、3.20、4.67--4.74 部分
V1.0.3	2020.10.22	V1.4.10/V2.0.10 及以上	增加目录 3.21、4.75--4.77 部分
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 3.17--3.19、4.14、4.16、4.84
V1.0.6	2020.01.18	V1.4.24/V2.0.24 及以上	修改目录 3.20、增加目录 4.35、4.39、4.84--4.98
V1.0.7	2021.04.28	V1.4.31/V1.5.12.17/V2.0.31 及以上	增加目录 3.25--3.29、4.86--4.87、4.101--4.108
V1.0.8	2021.08.30	V1.4.31/V1.5.12.17/V2.0.31 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.....	- 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 笛卡尔空间位置数据类型.....	- 9 -
3.4 欧拉角姿态数据类型.....	- 9 -
3.5 四元数姿态数据类型.....	- 9 -
3.6 笛卡尔空间位姿类型.....	- 10 -
3.7 旋转矩阵数据类型.....	- 10 -
3.8 程序运行状态枚举类型.....	- 10 -
3.9 坐标系选择枚举类型.....	- 11 -
3.10 运动模式枚举类型.....	- 11 -
3.11 系统检测数据类型.....	- 11 -
3.12 负载数据类型.....	- 12 -
3.13 关节位置数据类型.....	- 12 -
3.14 IO 类型.....	- 12 -
3.15 机器人状态数据类型.....	- 12 -
3.16 机器人力矩数据类型.....	- 13 -
3.17 机器人关节监测数据类型.....	- 13 -
3.18 机器人监测数据类型.....	- 13 -
3.19 力矩传感器监测数据类型.....	- 14 -
3.20 机器人状态监测数据类型.....	- 14 -
3.21 机器人错误码数据类型.....	- 15 -
3.22 轨迹复现配置参数存储数据类型.....	- 16 -
3.23 多个字符串存储数据类型.....	- 16 -
3.24 运动参数可选项.....	- 16 -
3.25 网络异常机器人自动终止类型枚举.....	- 17 -
3.26 柔顺控制参数类型.....	- 17 -
3.27 机器人柔顺控制参数类型.....	- 17 -
3.28 速度柔顺控制等级和比率等级设置.....	- 18 -
3.29 力矩传感器的受力分量和力矩分量.....	- 18 -
4. API.....	- 19 -
4.1 创建机器人控制句柄.....	- 19 -
4.2 销毁机器人控制句柄.....	- 19 -
4.3 机器人上电.....	- 19 -
4.4 机器人下电.....	- 19 -
4.5 机器人关机.....	- 20 -
4.6 机器人上使能.....	- 20 -
4.7 机器人下使能.....	- 20 -
4.8 机器人手动模式下运动.....	- 20 -
4.9 机器人手动模式下运动停止.....	- 21 -
4.10 机器人关节运动.....	- 21 -

4.11 机器人末端直线运动.....	22 -
4.12 机器人伺服位置控制模式使能.....	23 -
4.13 机器人关节空间伺服模式运动.....	23 -
4.14 机器人关节空间伺服模式运动扩展.....	24 -
4.15 机器人笛卡尔空间伺服模式运动.....	24 -
4.16 机器人笛卡尔空间伺服模式运动扩展.....	24 -
4.17 设置数字输出变量.....	25 -
4.18 设置模拟输出变量.....	26 -
4.19 查询数字输入状态.....	26 -
4.20 查询数字输出状态.....	26 -
4.21 获取模拟量输入变量的值.....	26 -
4.22 获取模拟量输出变量的值.....	27 -
4.23 查询扩展 IO 是否运行.....	27 -
4.24 运行当前加载的作业程序.....	27 -
4.25 暂停当前运行的作业程序.....	28 -
4.26 继续运行当前暂停的作业程序.....	29 -
4.27 终止当前执行的作业程序.....	29 -
4.28 加载指定的作业程序.....	29 -
4.29 获取已加载的作业程序的名字.....	29 -
4.30 获取当前机器人作业程序的执行行号.....	30 -
4.31 获取机器人作业程序的执行状态.....	30 -
4.32 设置机器人的运行倍率.....	30 -
4.33 获取机器人的运行倍率.....	31 -
4.34 设置工具信息.....	31 -
4.35 获取工具信息.....	32 -
4.36 设置当前使用的工具 ID.....	33 -
4.37 查询当前使用的工具 ID.....	33 -
4.38 设定用户坐标系信息.....	33 -
4.39 获取用户坐标系信息.....	34 -
4.40 设置当前使用的用户坐标系 ID.....	35 -
4.41 查询当前使用的用户坐标系 ID.....	35 -
4.42 控制机器人进入或退出拖拽模式.....	35 -
4.43 查询机器人是否处于拖拽模式.....	36 -
4.44 获取机器人状态.....	36 -
4.45 获取当前设置下工具末端的位姿.....	37 -
4.46 获取当前机器人关节角度.....	38 -
4.47 查询机器人是否处于碰撞保护模式.....	38 -
4.48 查询机器人是否超出限位.....	39 -
4.49 查询机器人运动是否停止.....	39 -
4.50 碰撞之后从碰撞保护模式恢复.....	40 -
4.51 设置机器人碰撞等级.....	41 -
4.52 获取机器设置的碰撞等级.....	42 -
4.53 机器人求解逆解.....	42 -
4.54 机器人求解正解.....	43 -

4.55 欧拉角到旋转矩阵的转换.....	44 -
4.56 旋转矩阵到欧拉角的转换.....	45 -
4.57 四元数到旋转矩阵的转换.....	46 -
4.58 旋转矩阵到四元数的转换.....	46 -
4.59 注册机器人出错时的回调函数.....	46 -
4.60 机器人负载设置.....	47 -
4.61 获取机器人负载数据.....	48 -
4.62 获取 SDK 版本号.....	48 -
4.63 获取控制器 IP.....	49 -
4.64 获取机器人状态监测数据.....	49 -
4.65 设置 SDK 是否开启调试模式.....	50 -
4.66 机器人运动终止.....	51 -
4.67 设置机器人错误码文件存放路径.....	52 -
4.68 获取机器人目前发生的最后一个错误码.....	52 -
4.69 设置轨迹复现配置参数.....	52 -
4.70 获取轨迹复现配置参数.....	52 -
4.71 采集轨迹复现数据控制开关.....	53 -
4.72 采集轨迹复现数据状态查询.....	54 -
4.73 查询控制器中已经存在的轨迹复现数据的文件名.....	54 -
4.74 重命名轨迹复现数据的文件名.....	54 -
4.75 删除控制器中轨迹复现数据文件.....	54 -
4.76 控制器中轨迹复现数据文件生成控制器执行脚本.....	55 -
4.77 机器人扩展关节运动.....	55 -
4.78 机器人扩展末端直线运动.....	55 -
4.79 机器人圆弧运动.....	56 -
4.80 机器人 SERVO 模式下禁用滤波器.....	57 -
4.81 机器人 SERVO 模式下关节空间一阶低通滤波.....	58 -
4.82 机器人 SERVO 模式下关节空间非线性滤波.....	58 -
4.83 机器人 SERVO 模式下笛卡尔空间非线性滤波.....	59 -
4.84 机器人 SERVO 模式下关节空间多阶均值滤波.....	60 -
4.85 机器人 SERVO 模式速度前瞻参数设置.....	60 -
4.86 设置 SDK 日志路径.....	61 -
4.87 设置传感器品牌.....	61 -
4.88 获取传感器品牌.....	61 -
4.89 开启或关闭力矩传感器.....	62 -
4.90 设置柔顺控制参数.....	62 -
4.91 开始辨识工具末端负载.....	62 -
4.92 获取末端负载辨识状态.....	64 -
4.93 获取末端负载辨识结果.....	64 -
4.94 设置传感器末端负载.....	64 -
4.95 获取传感器末端负载.....	65 -
4.96 力控拖拽使能（导纳控制）.....	65 -
4.97 设置力控类型和传感器初始化状态.....	66 -
4.98 获取力控类型和传感器初始化状态.....	66 -

4.99 获取力控柔顺控制参数.....	- 66 -
4.100 设置力控传感器通信参数.....	- 67 -
4.101 获取力控传感器通信参数.....	- 67 -
4.102 关闭力控.....	- 67 -
4.103 设置速度柔顺控制参数.....	- 68 -
4.104 设置柔顺控制力矩条件.....	- 68 -
4.105 设置网络异常，机器人终止运动等待时间.....	- 68 -
4.106 设置机器人状态数据自动更新时间间隔.....	- 68 -
4.107 初始化 FTP 客户端.....	- 69 -
4.108 FTP 上传.....	- 69 -
4.109 FTP 下载.....	- 69 -
4.110 FTP 目录查询.....	- 70 -
4.111 FTP 删除.....	- 70 -
4.112 FTP 重命名.....	- 70 -
4.113 关闭 FTP 客户端.....	- 71 -
5. 反馈与勘误.....	- 71 -

# 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。
- 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

```
1. /**
2.  * @brief 机器人回调函数指针
3.  */
4. delegate void CallBackFuncType(int error_code);
```

## 3.2 接口调用返回值列表

1. #define ERR_SUCC	0	//调用成功
2. #define ERR_FUCTION_CALL_ERROR	2	//异常调用，调用接口异常，控制器不支持
3. #define ERR_INVALID_HANDLER	-1	//无效的控制句柄



4.	#define ERR_INVALID_PARAMETER	-2	//无效的参数
5.	#define ERR_COMMUNICATION_ERR	-3	//通信连接错误
6.	#define ERR_KINE_INVERSE_ERR	-4	//逆解失败
7.	#define ERR_EMERGENCY_PRESSED	-5	//急停开关被按下
8.	#define ERR_NOT_POWERED	-6	//机器人未上电
9.	#define ERR_NOT_ENABLED	-7	//机器人未使能
10.	#define ERR_DISABLE_SERVOMODE	-8	//机器人没有进入 servo 模式
11.	#define ERR_NOT_OFF_ENABLE	-9	//机器人没有关闭使能
12.	#define ERR_PROGRAM_IS_RUNNING	-10	//程序正在运行，不允许操作
13.	#define ERR_CANNOT_OPEN_FILE	-11	//无法打开文件，文件不存在

### 3.3 笛卡尔空间位置数据类型

```

1.  /**
2.  * @brief 笛卡尔空间位置数据类型
3.  */
4.  public struct CartesianTran
5.  {
6.
7.      public double x;      ///< x 轴坐标，单位 mm
8.      public double y;      ///< y 轴坐标，单位 mm
9.      public double z;      ///< z 轴坐标，单位 mm
10. };

```

### 3.4 欧拉角姿态数据类型

```

1.  /**
2.  * @brief 欧拉角姿态数据类型
3.  */
4.  public struct Rpy
5.  {
6.      public double rx;      ///< 绕固定轴 X 旋转角度，单位：rad
7.      public double ry;      ///< 绕固定轴 Y 旋转角度，单位：rad
8.      public double rz;      ///< 绕固定轴 Z 旋转角度，单位：rad
9.  };

```

### 3.5 四元数姿态数据类型

```

1.  /**
2.  * @brief 四元数姿态数据类型
3.  */
4.  public struct Quaternion

```

```
5. {  
6.     public double s;  
7.     public double x;  
8.     public double y;  
9.     public double z;  
10. };
```

## 3.6 笛卡尔空间位姿类型

```
1. /**  
2.  * @brief 笛卡尔空间位姿类型  
3.  */  
4. public struct CartesianPose  
5. {  
6.     public CartesianTran tran;    ///< 笛卡尔空间位置  
7.     public Rpy rpy;              ///< 笛卡尔空间姿态  
8. };
```

## 3.7 旋转矩阵数据类型

```
1. /**  
2.  * @brief 旋转矩阵数据类型  
3.  */  
4. public struct RotMatrix  
5. {  
6.     public CartesianTran x;    ///< x 轴列分量  
7.     public CartesianTran y;    ///< y 轴列分量  
8.     public CartesianTran z;    ///< z 轴列分量  
9. };
```

## 3.8 程序运行状态枚举类型

```
1. /**  
2.  * @brief 程序运行状态枚举类型  
3.  */  
4. public enum ProgramState  
5. {  
6.     PROGRAM_IDLE,          ///< 机器人停止运行  
7.     PROGRAM_RUNNING,       ///< 机器人正在运行  
8.     PROGRAM_PAUSED         ///< 机器人暂停  
9. };
```

### 3.9 坐标系选择枚举类型

```
1. /**
2.  * @brief 坐标系选择枚举类型
3.  */
4. public enum CoordType
5. {
6.     COORD_BASE,      ///< 基坐标系
7.     COORD_JOINT,      ///< 关节空间
8.     COORD_TOOL        ///< 工具坐标系
9. };
```

### 3.10 运动模式枚举类型

```
1. /**
2.  * @brief jog 运动模式枚举
3.  */
4. public enum MoveMode
5. {
6.     ABS = 0,          ///< 绝对运动
7.     INCR              ///< 增量运动
8. };
```

### 3.11 系统检测数据类型

```
1. /**
2.  * @brief 系统监测数据类型
3.  */
4. public struct SystemMonitorData
5. {
6.     public int scbMajorVersion;      ///< scb 主版本号
7.     public int scbMinorVersion;      ///< scb 次版本号
8.     public int cabTemperature;        ///< 控制柜温度
9.     public double robotAveragePower;  ///< 控制柜总线平均功率
10.    public double robotAverageCurrent; ///< 控制柜总线平均电流
11.    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
12.    public double[] instCurrent;        ///< 机器人 6 个轴的瞬时电流
13.    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
14.    public double[] instVoltage;        ///< 机器人 6 个轴的瞬时电压
15.    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
16.    public double[] instTemperature;    ///< 机器人 6 个轴的瞬时温度
17. };
```

## 3.12 负载数据类型

```
1. /**
2.  * @brief 负载数据类型
3.  */
4. public struct Payload
5. {
6.     public double mass;           ///< 负载质量, 单位: kg
7.     public CartesianTran centroid; ///< 负载质心, 单位: mm
8. };
```

## 3.13 关节位置数据类型

```
1. /**
2.  * @brief 关节位置数据类型
3.  */
4. public struct JointValue
5. {
6.     [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
7.     public double[] jVal ;        ///< 6 关节位置值, 单位: rad
8. };
```

## 3.14 IO 类型

```
1. /**
2.  * @brief IO 类型枚举
3.  */
4. public enum IOType
5. {
6.     IO_CABINET,    ///< 控制柜面板 IO
7.     IO_TOOL,       ///< 工具 IO
8.     IO_EXTEND      ///< 扩展 IO
9. };
```

## 3.15 机器人状态数据类型

```
1. /**
2.  * @brief 机器人状态数据
3.  */
4. public struct RobotState
5. {
```

```

6.     public int estoped;          ///< 是否急停
7.     public int poweredOn;       ///< 是否打开电源
8.     public int servoEnabled;    ///< 是否处于伺服模式
9. };

```

### 3.16 机器人力矩数据类型

```

1.  /**
2.   * @brief 机器人力矩数据
3.   */
4.  public struct TorqueValue
5.  {
6.      [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
7.      public double[] jTorque;    ///< 是否使能
8.  }

```

### 3.17 机器人关节监测数据类型

```

1.  /**
2.   * @brief 机器人关节监测数据类型
3.   */
4.  public struct JointMonitorData
5.  {
6.      double instCurrent;         ///< 瞬时电流
7.      double instVoltage;        ///< 瞬时电压
8.      double instTemperature;    ///< 瞬时温度
9.  }

```

### 3.18 机器人监测数据类型

```

1.  /**
2.   * @brief 机器人监测数据类型
3.   */
4.  public struct RobotMonitorData
5.  {
6.      double scbMajorVersion;    ///< scb 主版本号
7.      double scbMinorVersion;    ///< scb 小版本号
8.      double cabTemperature;     ///< 控制器温度
9.      double robotAveragePower;  ///< 机器人平均电压
10.     double robotAverageCurrent; ///< 机器人平均电流

```

```

11.     [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
12.     public JointMonitorData[] jointMonitorData;          ///< 机器人 6 个关节的监测数据
13. }

```

### 3.19 力矩传感器监测数据类型

```

1.  /**
2.  * @brief 力矩传感器监测数据类型
3.  */
4.  public struct TorqSensorMonitorData
5.  {
6.      [MarshalAs(UnmanagedType.ByValArray, SizeConst = 20)]
7.      public char[] ip;          ///< 力矩传感器 ip 地址
8.      int port;                  ///< 力矩传感器端口号
9.      PayLoad payLoad;          ///< 工具负载
10.     int status;                 ///< 力矩传感器状态
11.     int errcode;                ///< 力矩传感器异常错误码
12.     [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
13.     public double[] actTorque;  ///< 力矩传感器实际接触力值
14.     [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
15.     public double[] torque;     ///< 力矩传感器原始读数
16. }

```

### 3.20 机器人状态监测数据类型

```

1.  /**
2.  * @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
3.  */
4.  typedef struct
5.  {
6.      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
7.      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
8.      int powered_on;            ///< 机器人是否上电标志, 0 为没有
                                   上电, 1 为上电
9.      int enabled;                ///< 机器人是否使能标志, 0 为没有
                                   使能, 1 为使能
10.     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;

```

## 3.21 机器人错误码数据类型

```

1.  /**
2.  * @brief 机器人错误码数据类型
3.  */
4.  public struct ErrorCode
5.  {

```

```

6.     public long code;                                     ///< 错误码编号
7.     [MarshalAs(UnmanagedType.ByValArray, SizeConst = 120)]
8.     public char[] message;                               ///< 错误码对应提示信息
9. }

```

### 3.22 轨迹复现配置参数存储数据类型

```

1.  /**
2.   * @brief 轨迹复现配置参数数据类型
3.   */
4.  public struct TrajTrackPara
5.  {
6.      public double xyz_interval;                         ///< 空间位置采集精度
7.      public double rpy_interval;                         ///< 姿态采集精度
8.      public double vel;                                  ///< 执行脚本运行速度
9.      public double acc;                                  ///< 执行脚本运行加速度
10. }

```

### 3.23 多个字符串存储数据类型

```

1.  /**
2.   * @brief 多个字符串存储数据类型
3.   */
4.  public struct MultStrStorType
5.  {
6.      public int len;                                     ///< 字符串个数
7.      [MarshalAs(UnmanagedType.ByValArray, SizeConst = 128*128)]
8.      public char[] message;                             ///< 错误码对应提示信息
9.  }

```

### 3.24 运动参数可选项

```

1.  /**
2.   * @brief 可选参数项
3.   */
4.  typedef struct
5.  {
6.      int executingLineId;                                ///< 控制命令 id 编号
7.  }OptionalCond;

```



### 3.25 网络异常机器人自动终止类型枚举

```

1.  /**
2.   * @brief 网络异常机器人运动自动终止类型枚举
3.   */
4.  public enum ProcessType
5.  {
6.      MOT_KEEP,    ///< 网络异常时机器人继续保持原来的运动
7.      MOT_PAUSE,   ///< 网络异常时机器人暂停运动
8.      MOT_ABORT    ///< 网络异常时机器人终止运动
9.  }

```

### 3.26 柔顺控制参数类型

```

1.  /**
2.   * @brief 柔顺控制参数类型
3.   */
4.  [StructLayout(LayoutKind.Sequential)]
5.  public struct AdmitCtrlType
6.  {
7.      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
8.      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
9.      double ft_rebound; ///< 回弹力: 机器人回到初始状态的能力
10.     double ft_constant; ///< 恒力
11.     int ft_normal_track; ///< 法向跟踪是否开启, 0 为没有开启, 1 为开启
12. }

```

### 3.27 机器人柔顺控制参数类型

```

1.  /**
2.   * @brief 机器人柔顺控制参数类型
3.   */
4.  [StructLayout(LayoutKind.Sequential)]
5.  public struct RobotAdmitCtrl
6.  {
7.      [MarshalAs(UnmanagedType.ByValArray, SizeConst = 6)]
8.      public AdmitCtrlType[] admit_ctrl;
9.  }

```

### 3.28 速度柔顺控制等级和比率等级设置

```

1.  /**
2.   * @brief 速度柔顺控制等级和比率等级设置
3.   * 速度柔顺控制分三个等级，并且 1>rate1>rate2>rate3>rate4>0
4.   * 等级为 1 时，只能设置 rate1,rate2 两个等级。rate3,rate4 的值为 0
5.   * 等级为 2 时，只能设置 rate1,rate2, rate3 三个等级。rate4 的值为 0
6.   * 等级为 3 时，能设置 rate1,rate2, rate3,rate4 4 个等级
7.   */
8.   [StructLayout(LayoutKind.Sequential)]
9.   public struct VelCom
10.  {
11.      int vc_level;                //速度柔顺控制等级
12.      double rate1;                //比率 1 等级
13.      double rate2;                //比率 2 等级
14.      double rate3;                //比率 3 等级
15.      double rate4;                //比率 4 等级
16.  }

```

### 3.29 力矩传感器的受力分量和力矩分量

```

1.  /**
2.   * @brief 力传感器的受力分量和力矩分量
3.   */
4.   [StructLayout(LayoutKind.Sequential)]
5.   public struct FTxyz
6.  {
7.      double fx;                  // 沿 x 轴受力分量
8.      double fy;                  // 沿 y 轴受力分量
9.      double fz;                  // 沿 z 轴受力分量
10.     double tx;                  // 绕 x 轴力矩分量
11.     double ty;                  // 绕 y 轴力矩分量
12.     double tz;                  // 绕 z 轴力矩分量
13. }

```

## 4.API

### 4.1 创建机器人控制句柄

```
1. /**
2.  * @brief 创建机器人控制句柄
3.  * @param ip 控制器 ip 地址
4.  * @param handle 机器人控制句柄
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7. int create_handler(char[] ip,ref int handle);
```

### 4.2 销毁机器人控制句柄

```
1. /**
2.  * @brief 销毁机器人控制句柄
3.  * @param handle 机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6. int destory_handler(ref int handle);
```

### 4.3 机器人上电

```
1. /**
2.  * @brief 打开机器人电源
3.  * @param handle 机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6. int power_on(ref int handle);
```

### 4.4 机器人下电

```
1. /**
2.  * @brief 关闭机器人电源
3.  * @param handle 机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6. int power_off(ref int handle);
```

## 4.5 机器人关机

```
1. /**
2.  * @brief 机器人控制柜关机
3.  * @param handle 机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6. int shut_down(ref int handle);
```

## 4.6 机器人上使能

```
1. /**
2.  * @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. int enable_robot(ref int 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. int disable_robot(ref int 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，连续运动(即持续 jog 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. int jog(ref int handle, int aj_num, JKTYPE.MoveMode move_mode, JKTYPE.CoordType coord_type,
    double vel_cmd, double pos_cmd);

```

代码示例：

```

1.  /// <summary>
2.  /// jog 运动 vel_cmd 超过 pi/4 会被限幅
3.  /// </summary>
4.  static void example_jog()
5.  {
6.      int handle = 0;
7.      //实例化机器人，将 ip 切换为自己的 ip
8.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
9.      jakaAPI.power_on(ref handle); //机器人上电
10.     jakaAPI.enable_robot(ref handle); //机器人上使能
11.     int aj_num = 0;
12.     //选择关节 1，运动模式：增量运动；关节坐标系；以 3.14/4 的速度沿关节 1 反方向运动 1rad。
13.     jakaAPI.jog(ref handle, aj_num, JKTYPE.MoveMode.INCR, JKTYPE.CoordType.COORD_JOINT, 3.
        14 / 4, -1);
14.     System.Threading.Thread.Sleep(1000);
15.     jakaAPI.jog_stop(ref handle, -1); // -1 代表停止所有轴的运动
16.     jakaAPI.destory_handler(ref handle);
17. }

```

## 4.9 机器人手动模式下运动停止

```

1.  /**
2.  * @brief 控制机器人手动模式下运动停止
3.  * @param handle 机器人控制句柄
4.  * @param num 机械臂轴号 0-5, num 为-1 时，停止所有轴
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int jog_stop(ref int handle, ref int num);

```

## 4.10 机器人关节运动

```

1.  /**
2.  * @brief 机器人关节运动
3.  * @param handle 机器人控制句柄
4.  * @param joint_pos 机器人关节运动目标位置
5.  * @param move_mode 指定运动模式：增量运动或绝对运动
6.  * @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
7.  * @param speed 机器人关节运动速度，单位：rad/s

```

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

```

8.  * @return ERR_SUCC 成功 其他失败
9.  */
10. int joint_move(const JKHD* handle, const JointValue* joint_pos, MoveMode move_mode, BOOL is_block, double speed);

```

代码示例：

```

1.  /// <summary>
2.  /// joint move 关节限速 pi/2 rad/s
3.  /// 支持阻塞和非阻塞，速度单位 rad/s
4.  /// </summary>
5.  static void example_joint_move()
6.  {
7.      int handle = 0;
8.      Stopwatch sw = new Stopwatch();//计时器
9.      //实例化机器人，将 ip 切换为自己的 ip
10.     int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
11.     jakaAPI.power_on(ref handle);
12.     jakaAPI.enable_robot(ref handle);
13.     //定义并初始化 JointValue 变量
14.     JKTYPE.JointValue joint_pos = new JKTYPE.JointValue();
15.     joint_pos.jVal = new double[] { 1, 1, 1, 1, 1, 1 };
16.     Console.WriteLine("joint_move 指令执行开始");
17.     sw.Start();
18.     //阻塞运动到目标点位
19.     jakaAPI.joint_move(ref handle, ref joint_pos, JKTYPE.MoveMode.ABS, true, 1);
20.     sw.Stop();
21.     Console.WriteLine("joint_move 指令执行结束
        {0}ms", sw.ElapsedTicks / (decimal)Stopwatch.Frequency);
22.     jakaAPI.destory_handler(ref handle);
23. }

```

## 4.11 机器人末端直线运动

```

1.  /**
2.  * @brief 机器人末端直线运动
3.  * @param handle 机器人控制句柄
4.  * @param end_pos 机器人末端运动目标位置
5.  * @param move_mode 指定运动模式：增量运动或绝对运动
6.  * @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
7.  * @param speed 机器人直线运动速度，单位：mm/s
8.  * @return ERR_SUCC 成功 其他失败
9.  */
10. int linear_move(ref int handle, ref JKTYPE.CartesianPose end_pos, JKTYPE.MoveMode move_mode,
    bool is_block, double speed);

```

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

## 代码示例：

```

1.  /// <summary>
2.  /// linear move
3.  /// 支持阻塞和非阻塞，速度单位 mm/s
4.  /// 注意避免奇异点，常见奇异点：
5.  /// * 工具末端位置在轴线 Z1 和 Z2 构成的平面上；
6.  /// * 轴线 Z2,Z3,Z4 共面；
7.  /// * 关节 5 角度为 0 或 180°，即 Z4,Z6 平行；
8.  /// </summary>
9.  static void example_linear_move()
10. {
11.     int handle = 0;
12.     Stopwatch sw = new Stopwatch();//计时器
13.     Console.WriteLine(sw.ElapsedTicks / (decimal)Stopwatch.Frequency);
14.     //实例化机器人，将 ip 切换为自己的 ip
15.     int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
16.     jakaAPI.power_on(ref handle);
17.     jakaAPI.enable_robot(ref handle);
18.     Console.WriteLine("linear_move 指令执行开始");
19.     JKTYPE.CartesianPose tcp_pos = new JKTYPE.CartesianPose();
20.     tcp_pos.tran.z = -100;
21.     sw.Start();
22.     jakaAPI.linear_move(ref handle, ref tcp_pos, JKTYPE.MoveMode.INCR, true, 30);
23.     sw.Stop();
24.     Console.WriteLine("linear_move 指令执行结束
        {0}s", sw.ElapsedTicks / (decimal)Stopwatch.Frequency);
25.     jakaAPI.destory_handler(ref handle);
26. }

```

## 4.12 机器人伺服位置控制模式使能

```

1.  /**
2.  * @brief 机器人伺服位置控制模式使能
3.  * @param handle 机器人控制句柄
4.  * @param enable TRUE 为进入伺服位置控制模式，FALSE 表示退出该模式
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int servo_move_enable(ref int handle, bool enable);

```

## 4.13 机器人关节空间伺服模式运动

```

1.  /**
2.  * @brief 机器人关节空间位置控制模式

```

```

3.  * @param handle  机器人控制句柄
4.  * @param joint_pos  机器人关节运动目标位置
5.  * @param move_mode  指定运动模式：绝对运动或相对运动
6.  * @return ERR_SUCC 成功 其他失败
7.  */
8.  int servo_j(ref int handle, ref JKTYPE.JointValue joint_pos, JKTYPE.MoveMode move_mode);

```

## 4.14 机器人关节空间伺服模式运动扩展

```

1.  /**
2.  * @brief 机器人关节空间位置控制模式
3.  * @param handle  机器人控制句柄
4.  * @param joint_pos  机器人关节运动目标位置
5.  * @move_mode  指定运动模式：增量运动或绝对运动
6.  * @step_num  倍分周期，servo_j 运动周期为 step_num*8ms，其中 step_num>=1
7.  * @return ERR_SUCC 成功 其他失败
8.  */
9.  int servo_j_extend(ref int handle, ref JKTYPE.JointValue joint_pos, JKTYPE.MoveMode move_mode, int step_num);

```

## 4.15 机器人笛卡尔空间伺服模式运动

```

1.  /**
2.  * @brief 机器人笛卡尔空间位置控制模式
3.  * @param handle  机器人控制句柄
4.  * @param cartesian_pose  机器人笛卡尔空间运动目标位置
5.  * @param move_mode  指定运动模式：ABS 代表绝对运动，INCR 代表相对运动
6.  * @return ERR_SUCC 成功 其他失败
7.  */
8.  int servo_p(ref int handle, ref JKTYPE.CartesianPose cartesian_pose, JKTYPE.MoveMode move_mode);

```

## 4.16 机器人笛卡尔空间伺服模式运动扩展

```

1.  /**
2.  * @brief 机器人笛卡尔空间位置控制模式
3.  * @param handle  机器人控制句柄
4.  * @param cartesian_pose  机器人笛卡尔空间运动目标位置
5.  * @move_mode  指定运动模式：增量运动或绝对运动

```



```

6.  * @step_num  倍分周期, servo_p 运动周期为 step_num*8ms, 其中 step_num>=1
7.  * @return  ERR_SUCC 成功 其他失败
8.  */
9.  int servo_p_extend(ref int handle, ref JKTYPE.CartesianPose cartesian_pose, JKTYPE.MoveMode move_mode, int step_num);

```

## 4.17 设置数字输出变量

```

1.  /**
2.  * @brief 设置数字输出变量(DO)的值
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 value DO 设置值
7.  * @return  ERR_SUCC 成功 其他失败
8.  */
9.  int set_digital_output(ref int handle, JKTYPE.IOType type, int index, bool value);

```

代码示例:

```

1.  /// <summary>
2.  /// 设置和查询 IO
3.  /// </summary>
4.  static void example_set_digital_output()
5.  {
6.      bool D02 = false;
7.      int handle = 0;
8.      //实例化机器人, 将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     jakaAPI.power_on(ref handle);
11.     jakaAPI.enable_robot(ref handle);
12.     //查询 do2 的状态
13.     jakaAPI.get_digital_output(ref handle, JKTYPE.IOType.IO_CABINET, 2, ref D02);
14.     Console.WriteLine("D03 = {0}\n", D02);
15.     //io_cabinet 是控制柜面板 IO, 2 代表 D02, 1 对应要设置的 DO 值。
16.     jakaAPI.set_digital_output(ref handle, JKTYPE.IOType.IO_CABINET, 2, true);
17.     System.Threading.Thread.Sleep(1000);
18.     //查询 do2 的状态
19.     jakaAPI.get_digital_output(ref handle, JKTYPE.IOType.IO_CABINET, 2, ref D02);
20.     Console.WriteLine("D03 = {0}\n", D02);
21.     jakaAPI.destory_handler(ref handle);
22. }

```

## 4.18 设置模拟输出变量

```
1. /**
2.  * @brief 设置模拟输出变量的值(AO)的值
3.  * @param handle 机器人控制句柄
4.  * @param type AO 类型
5.  * @param index AO 索引 （从 0 开始）
6.  * @param value AO 设置值
7.  * @return ERR_SUCC 成功 其他失败
8.  */
9. int set_analog_output(ref int handle, JKTYPE.IOType type, int index, float value);
```

## 4.19 查询数字输入状态

```
1. /**
2.  * @brief 查询数字输入(DI)状态
3.  * @param handle 机器人控制句柄
4.  * @param type DI 类型
5.  * @param index DI 索引 （从 0 开始）
6.  * @param result DI 状态查询结果
7.  * @return ERR_SUCC 成功 其他失败
8.  */
9. int get_digital_input(ref int handle, JKTYPE.IOType type, int index, ref bool result);
```

## 4.20 查询数字输出状态

```
1. /**
2.  * @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. int get_digital_output(ref int handle, JKTYPE.IOType type, int index, ref bool result);
```

## 4.21 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.  int get_analog_input(ref int handle, JKTYPE.IOType type, int index, ref float result);

```

## 4.22 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.  int get_analog_output(ref int handle, JKTYPE.IOType type, int index, ref float result);

```

## 4.23 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.  int is_extio_running(ref int handle, ref bool is_running);

```

## 4.24 运行当前加载的作业程序

```

1.  /**
2.  * @brief 运行当前加载的作业程序
3.  * @param handle 机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int program_run(ref int handle);

```

代码示例：

```

1.  /// <summary>
2.  /// 加载在 app 上编写好的程序，并进行过程控制
3.  /// </summary>
4.  static void example_program()

```

```

5. {
6.     System.Text.StringBuilder name = new System.Text.StringBuilder();
7.     int handle = 0;
8.     int cur_line = 0;
9.     //实例化机器人，将ip切换为自己的ip
10.    int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
11.    JKTYPE.ProgramState pstatus = new JKTYPE.ProgramState();
12.    //机器人上电
13.    jakaAPI.power_on(ref handle);
14.    //机器人上使能
15.    jakaAPI.enable_robot(ref handle);
16.    //加载预先通过app编辑的名字叫simple的脚本
17.    jakaAPI.program_load(ref handle, "simple".ToCharArray());
18.    //获取已加载的程序名
19.    jakaAPI.get_loaded_program(ref handle, name);
20.    Console.WriteLine("Pro_name is: {0}", name);
21.    //运行当前加载的程序
22.    jakaAPI.program_run(ref handle);
23.    System.Threading.Thread.Sleep(10000);
24.    //暂停当前运行的程序
25.    jakaAPI.program_pause(ref handle);
26.    //获取当前执行程序的行号
27.    jakaAPI.get_current_line(ref handle, ref cur_line);
28.    Console.WriteLine("cur_line is: {0}", cur_line);
29.    //获取当前程序状态
30.    jakaAPI.get_program_state(ref handle, ref pstatus);
31.    Console.WriteLine("pro_status is: {0}", pstatus);
32.    System.Threading.Thread.Sleep(2000);
33.    //继续运行当前程序
34.    jakaAPI.program_resume(ref handle);
35.    System.Threading.Thread.Sleep(10000);
36.    //终止当前程序
37.    jakaAPI.program_abort(ref handle);
38.    jakaAPI.destory_handler(ref handle);
39. }

```

## 4.25 暂停当前运行的作业程序

```

1. /**
2.  * @brief 暂停当前运行的作业程序
3.  * @param handle 机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6. int program_pause(ref int handle);

```

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

## 4.26 继续运行当前暂停的作业程序

```
1. /**
2.  * @brief 继续运行当前暂停的作业程序
3.  * @param handle 机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6. int program_resume(ref int handle);
```

## 4.27 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. int program_abort(ref int handle);
```

## 4.28 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. int program_load(ref int handle, char[] file);
```

## 4.29 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. int get_loaded_program(ref int handle, StringBuilder file);
```

## 4.30 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.  */
7.  int get_current_line(ref int handle, ref int curr_line);

```

## 4.31 获取机器人作业程序的执行状态

```

1.  /**
2.  * @brief 获取机器人作业程序执行状态
3.  * @param handle 机器人控制句柄
4.  * @param status 作业程序执行状态查询结果
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int get_program_state(ref int handle, ref JKTYPE.ProgramState status);

```

## 4.32 设置机器人的运行倍率

```

1.  /**
2.  * @brief 设置机器人运行倍率
3.  * @param handle 机器人控制句柄
4.  * @param rapid_rate 是程序运行倍率，设置范围为[0,1]
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int set_rapidrate(ref int handle, double rapid_rate);

```

代码示例：

```

1.  /// <summary>
2.  /// 获取与设置机器人的运行速度
3.  /// </summary>
4.  static void example_rapidrate()
5.  {
6.      int handle = 0;
7.      double rapid_rate = 0;
8.      //实例化机器人，将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //机器人上电
11.     jakaAPI.power_on(ref handle);
12.     //机器人上使能

```

```
13.   jakaAPI.enable_robot(ref handle);
14.   jakaAPI.get_rapidrate(ref handle, ref rapid_rate);
15.   Console.WriteLine("rapid_rate is : {0}", rapid_rate);
16.   //修改程序运行时机器人运动速度
17.   jakaAPI.set_rapidrate(ref handle, 0.4);
18.   System.Threading.Thread.Sleep(100);
19.   jakaAPI.get_rapidrate(ref handle, ref rapid_rate);
20.   Console.WriteLine("rapid_rate is : {0}", rapid_rate);
21.   jakaAPI.destory_handler(ref handle);
22. }
```

### 4.33 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.  int get_rapidrate(ref int handle, ref double rapid_rate);
```

### 4.34 设置工具信息

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

代码示例:

```
1.  /// <summary>
2.  /// 切换工具坐标系及获取与设置工具坐标系信息
3.  /// </summary>
4.  static void example_tool()
5.  {
6.      int id_ret = 0;
7.      int id_set = 0;
8.      int handle = 0;
9.      id_set = 2;
10.     JKTYPE.CartesianPose tcp_ret = new JKTYPE.CartesianPose();
```

**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

```

11.     JKTYPE.CartesianPose tcp_set = new JKTYPE.CartesianPose();
12.     char[] name = new char[50];
13.     name = "test".ToCharArray();
14.     //实例化机器人, 将 ip 切换为自己的 ip
15.     int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
16.     //机器人上电
17.     jakaAPI.power_on( ref handle);
18.     //查询当前使用的工具 ID
19.     jakaAPI.get_tool_id( ref handle, ref id_ret);
20.     //获取当前使用的工具信息
21.     jakaAPI.get_tool_data( ref handle, id_ret, ref tcp_ret);
22.     Console.WriteLine("id_using={0} \nx={1}, y={2}, z={3}\n", id_ret, tcp_ret.tran.x, tcp_
        ret.tran.y, tcp_ret.tran.y);
23.     Console.WriteLine("rx={0}, ry={1}, rz={2}\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.
        rpy.rz);
24.     //初始化工具坐标
25.     tcp_set.tran.x = 0; tcp_set.tran.y = 0; tcp_set.tran.z = 10;
26.     tcp_set.rpy.rx = 120 * PI / 180; tcp_set.rpy.ry = 90 * PI / 180; tcp_set.rpy.rz = -90
        * PI / 180;
27.     //设置工具信息
28.     jakaAPI.set_tool_data( ref handle, id_set, ref tcp_set, name);
29.     //切换当前使用的工具坐标
30.     jakaAPI.set_tool_id( ref handle, id_set);
31.     System.Threading.Thread.Sleep(1000);
32.     //查询当前使用的工具 ID
33.     jakaAPI.get_tool_id( ref handle, ref id_ret);
34.     //获取设置的工具信息
35.     jakaAPI.get_tool_data(ref handle, id_ret, ref tcp_ret);
36.     Console.WriteLine("id_using={0} \nx={1}, y={2}, z={3}\n", id_ret, tcp_ret.tran.x, tcp_
        ret.tran.y, tcp_ret.tran.y);
37.     Console.WriteLine("rx={0}, ry={1}, rz={2}\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.
        rpy.rz);
38.     jakaAPI.destory_handler(ref handle);
39. }

```

## 4.35 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.  int errno_t get_tool_data(ref int handle, ref int id, ref CartesianPose tcp);

```

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



## 4.36 设置当前使用的工具 ID

```
1. /**
2.  * @brief 设置当前使用的工具 ID
3.  * @param handle 机器人控制句柄
4.  * @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
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7. int set_tool_id(ref int handle, int id);
```

## 4.37 查询当前使用的工具 ID

```
1. /**
2.  * @brief 查询当前使用的工具 ID
3.  * @param handle 机器人控制句柄
4.  * @param id 工具 ID 查询结果
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7. int get_tool_id(ref int handle, ref int id);
```

## 4.38 设定用户坐标系信息

```
1. /**
2.  * @brief 设置指定编号的用户坐标系信息
3.  * @param handle 机器人控制句柄
4.  * @param id 用户坐标系编号 , 取值范围为[1,10]
5.  * @param user_frame 用户坐标系偏置值
6.  * @param name 用户坐标系别名
7.  * @return ERR_SUCC 成功 其他失败
8.  */
9. int set_user_frame_data(ref int handle, int id, ref JKTYPE.CartesianPose user_frame, char[]
   name);
```

代码示例:

```
1. /// <summary>
2. /// 切换用户坐标系及获取与设置用户坐标系信息
3. /// </summary>
4. static void example_user_frame()
5. {
6.     int id_ret = 0;
7.     int id_set = 2;
```

```

8.     int handle = 0;
9.     JKTYPE.CartesianPose tcp_ret = new JKTYPE.CartesianPose();
10.    JKTYPE.CartesianPose tcp_set = new JKTYPE.CartesianPose();
11.    char[] name = new char[50];
12.    name = "test".ToCharArray();
13.    //实例化机器人, 将 ip 切换为自己的 ip
14.    int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
15.    //机器人上电
16.    jakaAPI.power_on(ref handle);
17.    //查询当前使用的用户坐标系 ID
18.    jakaAPI.get_user_frame_id(ref handle, ref id_ret);
19.    //获取当前使用的用户坐标系信息
20.    jakaAPI.get_user_frame_data(ref handle, id_ret, ref tcp_ret);
21.    Console.WriteLine("id_using={0} \nx={1}, y={2}, z={3}\n", id_ret, tcp_ret.tran.x, tcp_
        ret.tran.y, tcp_ret.tran.y);
22.    Console.WriteLine("rx={0}, ry={1}, rz={2}\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.
        rpy.rz);
23.    //初始化用户坐标系坐标
24.    tcp_set.tran.x = 0; tcp_set.tran.y = 0; tcp_set.tran.z = 10;
25.    tcp_set.rpy.rx = 120 * PI / 180; tcp_set.rpy.ry = 90 * PI / 180; tcp_set.rpy.rz = -90
        * PI / 180;
26.    //设置用户坐标系信息
27.    jakaAPI.set_user_frame_data(ref handle, id_set, ref tcp_set, name);
28.    //切换当前使用的用户坐标系坐标
29.    jakaAPI.set_user_frame_id(ref handle, id_set);
30.    System.Threading.Thread.Sleep(1000);
31.    //查询当前使用的用户坐标系 ID
32.    jakaAPI.get_user_frame_id(ref handle, ref id_ret);
33.    //获取当前使用的用户坐标系信息
34.    jakaAPI.get_user_frame_data(ref handle, id_ret, ref tcp_ret);
35.    Console.WriteLine("id_using={0} \nx={1}, y={2}, z={3}\n", id_ret, tcp_ret.tran.x, tcp_
        ret.tran.y, tcp_ret.tran.y);
36.    Console.WriteLine("rx={0}, ry={1}, rz={2}\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.
        rpy.rz);
37.    jakaAPI.destory_handler(ref handle);
38. }

```

## 4.39 获取用户坐标系信息

1. /\*\*
2. \* @brief 查询当前使用的用户坐标系信息
3. \* @param id 用户坐标系 ID 查询结果
4. \* @param tcp 用户坐标系偏置值

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

```

5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int get_user_frame_data(ref int handle handle, ref int id, ref CartesianPose tcp);

```

## 4.40 设置当前使用的用户坐标系 ID

```

1.  /**
2.  * @brief 设置当前使用的用户坐标系 ID
3.  * @param handle 机器人控制句柄
4.  * @param id 用户坐标系 ID，取值范围为[0,10]，其中 0 为世界坐标系
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int set_user_frame_id(ref int handle, int id);

```

## 4.41 查询当前使用的用户坐标系 ID

```

1.  /**
2.  * @brief 查询当前使用的用户坐标系 ID
3.  * @param handle 机器人控制句柄
4.  * @param id 获取的结果
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int get_user_frame_id(ref int handle, ref int id);

```

## 4.42 控制机器人进入或退出拖拽模式

```

1.  /**
2.  * @brief 控制机器人进入或退出拖拽模式
3.  * @param handle 机器人控制句柄
4.  * @param enable TRUE 为进入拖拽模式，FALSE 为退出拖拽模式
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int drag_mode_enable(ref int handle, bool enable);

```

代码示例：

```

1.  /// <summary>
2.  /// 进入拖拽模式，直到获得输入
3.  /// </summary>
4.  static void example_drag()
5.  {
6.      bool in_drag = true;
7.      int handle = 0;

```

```

8.      //实例化机器人, 将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //机器人上电
11.     jakaAPI.power_on( ref handle);
12.     //机器人上使能
13.     jakaAPI.enable_robot( ref handle);
14.     //确认机器人是否在拖拽模式下
15.     jakaAPI.is_in_drag_mode( ref handle, ref in_drag);
16.     Console.WriteLine("in_drag is : {0}", in_drag);
17.     //使能拖拽模式
18.     jakaAPI.drag_mode_enable( ref handle, true);
19.     System.Threading.Thread.Sleep(1000);
20.     jakaAPI.is_in_drag_mode( ref handle, ref in_drag);
21.     Console.WriteLine("in_drag is : {0}", in_drag);
22.     Console.WriteLine("input any word to disable dragmode");
23.     Console.Read();
24.     //去使能拖拽模式
25.     jakaAPI.drag_mode_enable(ref handle, false);
26.     System.Threading.Thread.Sleep(1000);
27.     jakaAPI.is_in_drag_mode(ref handle, ref in_drag);
28.     Console.WriteLine("in_drag is : {0}", in_drag);
29.     jakaAPI.destory_handler(ref handle);
30. }

```

## 4.43 查询机器人是否处于拖拽模式

```

1.  /**
2.   * @brief 查询机器人是否处于拖拽模式
3.   * @param handle 机器人控制句柄
4.   * @param in_drag 查询结果 0 不在拖拽模式, 1 为在拖拽模式下
5.   * @return ERR_SUCC 成功 其他失败
6.   */
7.  int is_in_drag_mode(ref int handle, ref bool in_drag);

```

## 4.44 获取机器人状态

```

1.  /**
2.   * @brief 获取机器人状态
3.   * @param handle 机器人控制句柄
4.   * @param state 机器人状态查询结果
5.   * @return ERR_SUCC 成功 其他失败
6.   */
7.  int get_robot_state(ref int handle, ref JKTYPE.RobotState state);

```

代码示例：

```

1.  /// <summary>
2.  /// 获取机器人状态, (急停 上电 伺服使能)
3.  /// </summary>
4.  static void example_get_robot_state()
5.  {
6.      int handle = 0;
7.      //实例化机器人, 将 ip 切换为自己的 ip
8.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
9.      JKTYPE.RobotState state = new JKTYPE.RobotState();
10.     //机器人上电
11.     jakaAPI.power_on(ref handle);
12.     //机器人上使能
13.     jakaAPI.enable_robot(ref handle);
14.     //获取机器人状态(急停 上电 伺服模式)
15.     jakaAPI.get_robot_state(ref handle, ref state);
16.     Console.WriteLine("is e_stoped :{0} ", state.estoped);
17.     jakaAPI.destory_handler(ref handle);
18. }

```

## 4.45 获取当前设置下工具末端的位姿

```

1.  /**
2.  * @brief 获取当前设置下工具末端的位姿
3.  * @param handle 机器人控制句柄
4.  * @param tcp_position 工具末端位置查询结果
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int get_tcp_position(ref int handle, ref JKTYPE.CartesianPose tcp_position);

```

代码示例：

```

1.  /// <summary>
2.  /// 获取机器人 tcp 空间位姿
3.  /// </summary>
4.  static void example_get_tcp_position()
5.  {
6.      int handle = 0;
7.      JKTYPE.CartesianPose tcp_pos = new JKTYPE.CartesianPose();
8.      //实例化机器人, 将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //机器人上电
11.     jakaAPI.power_on(ref handle);
12.     //机器人上使能
13.     jakaAPI.enable_robot(ref handle);

```

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

```

14.    //获取工具末端位姿
15.    jakaAPI.get_tcp_position(ref handle, ref tcp_pos);
16.    Console.WriteLine("tcp_position is :{0} {1} {2} {3} {4} {5}", tcp_pos.tran.x, tcp_pos.
    tran.y, tcp_pos.tran.z, tcp_pos.rpy.rx, tcp_pos.rpy.ry, tcp_pos.rpy.rz);
17.    jakaAPI.destory_handler(ref handle);
18. }

```

## 4.46 获取当前机器人关节角度

```

1.  /**
2.   * @brief 获取当前机器人关节角度
3.   * @param handle 机器人控制句柄
4.   * @param joint_position 关节角度查询结果
5.   * @return ERR_SUCC 成功 其他失败
6.   */
7.  int get_joint_position(ref int handle, ref JKTYPE.JointValue joint_position);

```

代码示例：

```

1.  /// <summary>
2.  /// 获取机器人关节空间关节角矩阵
3.  /// </summary>
4.  static void example_get_joint_position()
5.  {
6.      int handle = 0;
7.      JKTYPE.JointValue jot_pos = new JKTYPE.JointValue();
8.      //实例化机器人，将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //机器人上电
11.     jakaAPI.power_on(ref handle);
12.     //机器人上使能
13.     jakaAPI.enable_robot(ref handle);
14.     //获取工具末端位姿
15.     jakaAPI.get_joint_position(ref handle, ref jot_pos);
16.     Console.WriteLine("tcp_position is :{0} {1} {2} {3} {4} {5}", jot_pos.jVal[0], jot_pos.
        jVal[1], jot_pos.jVal[2], jot_pos.jVal[3], jot_pos.jVal[4], jot_pos.jVal[5]);
17.     jakaAPI.destory_handler(ref handle);
18. }

```

## 4.47 查询机器人是否处于碰撞保护模式

```

1.  /**
2.   * @brief 查询机器人是否处于碰撞保护模式

```

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

```

3.  * @param handle  机器人控制句柄
4.  * @param in_collision  查询结果 0 为不在碰撞, 1 为出现碰撞
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int is_in_collision(ref int handle, ref bool in_collision);

```

## 4.48 查询机器人是否超出限位

```

1.  /**
2.  * @brief  查询机器人是否超出限位
3.  * @param handle  机器人控制句柄
4.  * @param on_limit  查询结果 0 未超限, 1 超限
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int is_on_limit(ref int handle, ref bool on_limit);

```

## 4.49 查询机器人运动是否停止

```

1.  /**
2.  * @brief  查询机器人运动是否停止
3.  * @param handle  机器人控制句柄
4.  * @param in_pos  查询结果 0 运动进行中, 1 运动完成
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int is_in_pos(ref int handle, ref bool in_pos);

```

代码示例:

```

1.  /// <summary>
2.  ///  查询运动是否到位
3.  /// </summary>
4.  static void example_is_in_pos()
5.  {
6.      bool in_pos = false;
7.      int handle = 0;
8.      //实例化机器人, 将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //机器人上电
11.     jakaAPI.power_on(ref handle);
12.     //机器人上使能
13.     jakaAPI.enable_robot(ref handle);
14.     //定义并初始化关节 JointValue 变量
15.     JKTYPE.JointValue joint_pos = new JKTYPE.JointValue();
16.     joint_pos.jVal = new double[] { 1, 1, 1, 1, 1, 1 };
17.     Console.WriteLine("joint_move 指令执行开始");

```

```

18.    //非阻塞运动到目标点位
19.    jakaAPI.joint_move(ref handle, ref joint_pos, JKTYPE.MoveMode.ABS, false, 1);
20.    while (!in_pos)
21.    {
22.        //查询是否运动停止
23.        jakaAPI.is_in_pos(ref handle, ref in_pos);
24.        Console.WriteLine(" in_pos is :{0}", in_pos);
25.        System.Threading.Thread.Sleep(200);
26.    }
27.    jakaAPI.destory_handler(ref handle);
28. }

```

## 4.50 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.  int collision_recover(ref int handle);

```

代码示例：

```

1.  /// <summary>
2.  /// 碰撞保护状态查询及恢复
3.  /// </summary>
4.  static void example_collision_recover()
5.  {
6.      bool in_collision = false;
7.      int handle = 0;
8.      //实例化机器人，将ip切换为自己的ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //机器人上电
11.     jakaAPI.power_on(ref handle);
12.     //机器人上使能
13.     jakaAPI.enable_robot(ref handle);
14.     //查询是否处于碰撞保护状态
15.     jakaAPI.is_in_collision(ref handle, ref in_collision);
16.     if (in_collision)
17.     //如果处于碰撞保护模式，则从碰撞保护中恢复
18.     {
19.         jakaAPI.collision_recover(ref handle);
20.         Console.WriteLine("robot collision recover");
21.     }
22.     else

```



```

23.    {
24.        Console.WriteLine("robot is not collision");
25.    }
26.    jakaAPI.destory_handler(ref handle);
27. }

```

## 4.51 设置机器人碰撞等级

```

1.  /**
2.  * @brief 设置机器人碰撞等级
3.  * @param handle 机器人控制句柄
4.  * @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
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int set_collision_level(ref int handle,int level);

```

代码示例：

```

1.  /// <summary>
2.  /// 碰撞等级的设置和查询
3.  /// </summary>
4.  static void example_collision_level()
5.  {
6.      int level = 0 ;
7.      int handle = 0;
8.      //实例化机器人，将ip切换为自己的ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //机器人上电
11.     jakaAPI.power_on(ref handle);
12.     //机器人上使能
13.     jakaAPI.enable_robot(ref handle);
14.     //查询当前碰撞等级
15.     jakaAPI.get_collision_level(ref handle, ref level);
16.     Console.WriteLine(" collision level is :{0}", level);
17.     //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，
18.     jakaAPI.set_collision_level(ref handle, 2);
19.     //查询当前碰撞等级
20.     jakaAPI.get_collision_level(ref handle, ref level);
21.     Console.WriteLine(" collision level is :{0}", level);
22.     jakaAPI.destory_handler(ref handle);
23. }

```

## 4.52 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.  * @param handle 机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int get_collision_level(ref int handle, ref int level);

```

## 4.53 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 handle 机器人控制句柄
4.  * @param ref_pos 逆解计算用的参考关节空间位置
5.  * @param cartesian_pose 笛卡尔空间位姿值，欧拉角注意为弧度制
6.  * @param joint_pos 计算成功时关节空间位置计算结果
7.  * @return ERR_SUCC 成功 其他失败
8.  */
9.  int kine_inverse(ref int handle, ref JKTYPE.JointValue ref_pos, ref JKTYPE.CartesianPose
    cartesian_pose, ref JKTYPE.JointValue joint_pos);

```

代码示例：

```

1.  /// <summary>
2.  /// 运动学 逆解 已知 tcp_pos, 求 joint_pos
3.  /// </summary>
4.  static void example_kine_inverse()
5.  {
6.      int handle = 0;
7.      int ret;
8.      //实例化机器人, 将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //初始化参考点
11.     JKTYPE.JointValue ref_jpos = new JKTYPE.JointValue();
12.     ref_jpos.jVal = new double[] { 0.558, 0.872, 0.872, 0.349, 0.191, 0.191 };
13.     //初始化笛卡尔空间点坐标
14.     JKTYPE.CartesianPose tcp_pos = new JKTYPE.CartesianPose();
15.     tcp_pos.tran.x = 243.568; tcp_pos.tran.y = 164.064; tcp_pos.tran.z = 300.002;
16.     tcp_pos.rpy.rx = -1.81826; tcp_pos.rpy.ry = -0.834253; tcp_pos.rpy.rz = -2.30243;
17.     //初始化返回值
18.     JKTYPE.JointValue joint_pos = new JKTYPE.JointValue();
19.     joint_pos.jVal = new double[] { 0, 0, 0, 0, 0, 0 };
20.     //机器人逆解 已知 tcp_pos, 求 joint_pos
21.     ret = jakaAPI.kine_inverse(ref handle, ref ref_jpos, ref tcp_pos, ref joint_pos);

```

```

22.     Console.WriteLine(" {0}   kine_inverse:", ret); //若返回值为-4 则逆解失败, 更改参考点或数学
      上无解
23.     Console.WriteLine("{0} {1} {2} ", joint_pos.jVal[0], joint_pos.jVal[1], joint_pos.jVal
      [2]);
24.     Console.WriteLine("{0} {1} {2} ", joint_pos.jVal[3], joint_pos.jVal[4], joint_pos.jVal
      [5]);
25.     jakaAPI.destory_handler(ref handle);
26. }

```

## 4.54 机器人求解正解

```

1.  /**
2.   * @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
3.   * @param handle 机器人控制句柄
4.   * @param joint_pos 关节空间位置
5.   * @param cartesian_pose 笛卡尔空间位姿计算结果
6.   * @return ERR_SUCC 成功 其他失败
7.   */
8. int kine_forward(ref int handle, ref JKTYPE.JointValue joint_pos, ref JKTYPE.CartesianPose
   cartesian_pose);

```

代码示例:

```

1.  /// <summary>
2.  /// 机器人正解 已知 joint_pos, 求 tcp_pos
3.  /// </summary>
4.  static void example_kine_forward()
5.  {
6.      int handle = 0;
7.      int ret;
8.      //实例化机器人, 将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //初始化笛卡尔空间点坐标
11.     JKTYPE.CartesianPose tcp_pos = new JKTYPE.CartesianPose();
12.     //初始化返回值
13.     JKTYPE.JointValue joint_pos = new JKTYPE.JointValue();
14.     joint_pos.jVal = new double[] { 0.558, 0.872, 0.872, 0.349, 0.191, 0.191 };
15.     ret = jakaAPI.kine_forward(ref handle, ref joint_pos, ref tcp_pos);
16.     Console.WriteLine("tcp_pos is :\n x:{0} y:{1} z:{2} rx:{3} ry:{4} rz:{5}", tcp_pos.
       tran.x, tcp_pos.tran.y, tcp_pos.tran.z, tcp_pos.rpy.rx, tcp_pos.rpy.ry, tcp_pos.rpy.rz);
17.     jakaAPI.destory_handler(ref handle);
18. }

```

## 4.55 欧拉角到旋转矩阵的转换

```

1.  /**
2.  * @brief 欧拉角到旋转矩阵的转换
3.  * @param handle 机器人控制句柄
4.  * @param rpy 待转换的欧拉角数据
5.  * @param rot_matrix 转换后的旋转矩阵
6.  * @return ERR_SUCC 成功 其他失败
7.  */
8.  int rpy_to_rot_matrix(ref int handle, ref JKTYPE.Rpy rpy, ref JKTYPE.RotMatrix rot_matrix);

```

代码示例：

```

1.  /// <summary>
2.  /// 欧拉角->旋转矩阵->四元数
3.  /// </summary>
4.  static void example_rpy2rot_matr2quat()
5.  {
6.      int handle = 0;
7.      int ret;
8.      //实例化机器人, 将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //初始化欧拉角
11.     JKTYPE.Rpy rpy = new JKTYPE.Rpy();
12.     rpy.rx = -1.81826; rpy.ry = -0.834253; rpy.rz = -2.30243;
13.     //初始化旋转矩阵
14.     JKTYPE.RotMatrix rot_matrix = new JKTYPE.RotMatrix();
15.     rot_matrix.x.x = 0; rot_matrix.y.x = 0; rot_matrix.z.x = 0;
16.     rot_matrix.x.y = 0; rot_matrix.y.y = 0; rot_matrix.z.y = 0;
17.     rot_matrix.x.z = 0; rot_matrix.y.z = 0; rot_matrix.z.z = 0;
18.     //初始化四元数
19.     JKTYPE.Quaternion quat = new JKTYPE.Quaternion();
20.     quat.s = 0; quat.x = 0; quat.y = 0; quat.z = 0;
21.     //欧拉角到旋转矩阵
22.     ret = jakaAPI.rpy_to_rot_matrix(ref handle, ref rpy, ref rot_matrix);
23.     Console.WriteLine("    eul2rotm");
24.     Console.WriteLine("{0} {1} {2}", rot_matrix.x.x, rot_matrix.y.x, rot_matrix.z.x);
25.     Console.WriteLine("{0} {1} {2}", rot_matrix.x.y, rot_matrix.y.y, rot_matrix.z.y);
26.     Console.WriteLine("{0} {1} {2}", rot_matrix.x.z, rot_matrix.y.z, rot_matrix.z.z);
27.     //旋转矩阵---->四元数
28.     ret = jakaAPI.rot_matrix_to_quaternion(ref handle, ref rot_matrix, ref quat);
29.     Console.WriteLine(" {0}   rotm2quat:", ret);
30.     Console.WriteLine("{0} {1} {2} {3} ", quat.s, quat.x, quat.y, quat.z);
31.     jakaAPI.destory_handler(ref handle);
32. }

```

## 4.56 旋转矩阵到欧拉角的转换

```

1.  /**
2.  * @brief 旋转矩阵到欧拉角的转换
3.  * @param handle 机器人控制句柄
4.  * @param rot_matrix 待转换的旋转矩阵数据
5.  * @param rpy 转换后的 RPY 欧拉角结果
6.  * @return ERR_SUCC 成功 其他失败
7.  */
8.  int rot_matrix_to_rpy(ref int handle, ref JKTYPE.RotMatrix rot_matrix, ref JKTYPE.Rpy rpy);

```

代码示例：

```

1.  /// <summary>
2.  /// 四元数->旋转矩阵->欧拉角
3.  /// </summary>
4.  static void example_quat2rot_matr2rpy()
5.  {
6.      int handle = 0;
7.      int ret;
8.      //实例化机器人, 将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //初始化欧拉角
11.     JKTYPE.Rpy rpy = new JKTYPE.Rpy();
12.     rpy.rx = 0; rpy.ry = 0; rpy.rz = 0;
13.     //初始化旋转矩阵
14.     JKTYPE.RotMatrix rot_matrix = new JKTYPE.RotMatrix();
15.     rot_matrix.x.x = 0; rot_matrix.y.x = 0; rot_matrix.z.x = 0;
16.     rot_matrix.x.y = 0; rot_matrix.y.y = 0; rot_matrix.z.y = 0;
17.     rot_matrix.x.z = 0; rot_matrix.y.z = 0; rot_matrix.z.z = 0;
18.     //初始化四元数
19.     JKTYPE.Quaternion quat = new JKTYPE.Quaternion();
20.     quat.s = 0.0629; quat.x = 0.522886; quat.y = -0.5592; quat.z = 0.6453;
21.     //四元数-->旋转矩阵
22.     ret = jakaAPI.quaternion_to_rot_matrix(ref handle, ref quat, ref rot_matrix);
23.     Console.WriteLine(" {0}    quat12rotm:", ret);
24.     Console.WriteLine("{0} {1} {2}", rot_matrix.x.x, rot_matrix.y.x, rot_matrix.z.x);
25.     Console.WriteLine("{0} {1} {2}", rot_matrix.x.y, rot_matrix.y.y, rot_matrix.z.y);
26.     Console.WriteLine("{0} {1} {2}", rot_matrix.x.z, rot_matrix.y.z, rot_matrix.z.z);
27.     //旋转矩阵---->欧拉角
28.     ret = jakaAPI.rot_matrix_to_rpy(ref handle, ref rot_matrix, ref rpy);
29.     Console.WriteLine(" {0}    rotm2eul:", ret);
30.     Console.WriteLine("{0} {1} {2} ", rpy.rx, rpy.ry, rpy.rz);
31.     jakaAPI.destory_handler(ref handle);
32. }

```

## 4.57 四元数到旋转矩阵的转换

```

1.  /**
2.   * @brief 四元数到旋转矩阵的转换
3.   * @param handle 机器人控制句柄
4.   * @param quaternion 待转换的四元数数据
5.   * @param rot_matrix 转换后的旋转矩阵结果
6.   * @return ERR_SUCC 成功 其他失败
7.   */
8.  int quaternion_to_rot_matrix(ref int handle, ref JKTYPE.Quaternion quaternion, ref
    JKTYPE.RotMatrix rot_matrix);

```

## 4.58 旋转矩阵到四元数的转换

```

1.  /**
2.   * @brief 旋转矩阵到四元数的转换
3.   * @param handle 机器人控制句柄
4.   * @param rot_matrix 待转换的旋转矩阵
5.   * @param quaternion 转换后的四元数结果
6.   * @return ERR_SUCC 成功 其他失败
7.   */
8.  int rot_matrix_to_quaternion(ref int handle, ref JKTYPE.RotMatrix rot_matrix, ref
    JKTYPE.Quaternion quaternion);

```

## 4.59 注册机器人出错时的回调函数

```

1.  /**
2.   * @brief 注册机器人出现错误时的回调函数
3.   * @param handle 机器人控制句柄
4.   * @param func 指向用户定义的函数的函数指针
5.   * @param error_code 机器人的错误码
6.   */
7.  int set_error_handler(ref int i, CallbackFuncType func);

```

代码示例：

```

1.  //错误处理，多线程方式进行
2.  static void user_error_handle(int error_code)
3.  {
4.      Console.WriteLine("{0}", error_code);
5.      int a = 0;
6.      a = Console.Read();
7.      Console.WriteLine("{0}\n", a);
8.  }

```

```

9.  //注册
10. static void example_set_err_handle()
11. {
12.     int handle = 0;
13.     //实例化机器人，将 ip 切换为自己的 ip
14.     int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
15.     //机器人上电
16.     jakaAPI.power_on(ref handle);
17.     //机器人上使能
18.     jakaAPI.enable_robot(ref handle);
19.     //设置用户异常回调函数
20.     jakaAPI.set_error_handler(ref handle ,user_error_handle);
21.     while (true)
22.     {
23.         System.Threading.Thread.Sleep(1000);
24.         Console.WriteLine("nothing happen");
25.     }
26.     jakaAPI.destory_handler(ref handle);
27. }

```

## 4.60 机器人负载设置

```

1.  /**
2.   * @brief 机器人负载设置
3.   * @param handle 机器人控制句柄
4.   * @param payload 负载质心、质量数据
5.   * @return ERR_SUCC 成功 其他失败
6.   */
7. int set_payload(ref int handle, ref JKTYPE.Payload payload);

```

代码示例：

```

1.  /// <summary>
2.  /// 获取与设置负载
3.  /// </summary>
4.  static void example_payload()
5.  {
6.      int handle = 0;
7.      int ret;
8.      //实例化机器人，将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     JKTYPE.Payload payloadret = new JKTYPE.Payload();
11.     payloadret.mass = 0;
12.     payloadret.centroid.x = 0; payloadret.centroid.y = 0; payloadret.centroid.z = 0;
13.     JKTYPE.Payload payload_set = new JKTYPE.Payload();

```

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

```
14.    //查询当前负载数据
15.    jakaAPI.get_payload(ref handle, ref payloadret);
16.    Console.WriteLine(" payload mass is : {0} kg", payloadret.mass);
17.    Console.WriteLine(" payload center of mass is \nx: {0}  y: {1} z: {2} ", payloadret.ce
        ntroid.x, payloadret.centroid.y, payloadret.centroid.z);
18.    payload_set.mass = 1.0;
19.    //单位 mm
20.    payload_set.centroid.x = 0; payload_set.centroid.y = 0; payload_set.centroid.z = 10;
21.    //设置当前负载数据
22.    jakaAPI.set_payload(ref handle, ref payload_set);
23.    //查询当前负载数据
24.    jakaAPI.get_payload(ref handle, ref payloadret);
25.    Console.WriteLine(" payload mass is : {0} kg", payloadret.mass);
26.    Console.WriteLine(" payload center of mass is \nx: {0}  y: {1} z: {2} ", payloadret.ce
        ntroid.x, payloadret.centroid.y, payloadret.centroid.z);
27.    jakaAPI.destory_handler(ref handle);
28. }
```

## 4.61 获取机器人负载数据

```
1.  /**
2.   * @brief 获取机器人负载数据
3.   * @param handle 机器人控制句柄
4.   * @param payload 负载查询结果
5.   * @return ERR_SUCC 成功 其他失败
6.   */
7.  int get_payload(ref int handle, ref JKTYPE.PayLoad payload);
```

## 4.62 获取 SDK 版本号

```
1.  /**
2.   * @brief 获取机器人控制器版本号
3.   * @param handle 机器人控制句柄
4.   * @param version SDK 版本号
5.   * @return ERR_SUCC 成功 其他失败
6.   */
7.  int get_sdk_version(ref int handle, StringBuilder version);
```



## 4.63 获取控制器 IP

```

1.  /**
2.  * @brief 获取控制器 IP
3.  * @param controller_name 控制器名字
4.  * @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
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int get_controller_ip(char[] controller_name, StringBuilder ip_list);

```

代码示例：

```

1.  /// <summary>
2.  /// 获取局域网内所有机器人的 ip 及名称
3.  /// </summary>
4.  static void example_get_controller_ip()
5.  {
6.      int handle = 0;
7.      int ret;
8.      //实例化机器人，将 ip 切换为自己的 ip
9.      // int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     StringBuilder ip_list = new StringBuilder("", 3000);
11.     char[] controller_name1 = new char[10];
12.     controller_name1[0] = '\0';
13.     Console.WriteLine("start to get ip");
14.     //获取控制器 ip
15.     ret = jakaAPI.get_controller_ip( controller_name1, ip_list);
16.     Console.WriteLine("ip list get :{0}", ret);
17.     Console.WriteLine(" ip_list is :\n {0}", ip_list);
18.     jakaAPI.destory_handler(ref handle);
19. }

```

## 4.64 获取机器人状态监测数据

```

1.  /**
2.  * @brief 获取机器人状态数据
3.  * @param status 机器人状态
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int get_robot_status(ref int handle, ref JKTYPE.RobotStatus status);

```

代码示例：

```

1.  /// <summary>

```

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

```

2.  /// 获取机器人状态信息 24 种
3.  /// </summary>
4.  static void example_get_robot_status()
5.  {
6.      int handle = 0;
7.      int ret = 0 ;
8.      JKTYPE.RobotStatus robstatus = new JKTYPE.RobotStatus();
9.      //实例化机器人, 将 ip 切换为自己的 ip
10.     int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
11.     //机器人上电
12.     jakaAPI.power_on(ref handle);
13.     //机器人上使能
14.     jakaAPI.enable_robot(ref handle);
15.     while (true)
16.     {
17.         ret++;
18.         System.Threading.Thread.Sleep(1000);
19.         //获取机器人状态监测数据
20.         jakaAPI.get_robot_status(ref handle, ref robstatus);
21.         Console.WriteLine("{1}rapidrate:{0}\n", robstatus.rapidrate, ret);
22.     }
23.     jakaAPI.disable_robot(ref handle);
24.     jakaAPI.power_off(ref handle);
25.     jakaAPI.destory_handler(ref handle);
26. }

```

## 4.65 设置 SDK 是否开启调试模式

```

1.  /**
2.   * @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.  int set_debug_mode(ref int handle, bool mode);

```

代码示例:

```

1.  /// <summary>
2.  /// 开启 sdkdebug 模式
3.  /// </summary>
4.  static void example_set_debug_mode()
5.  {
6.      int handle = 0;
7.      int ret = 0;
8.      //实例化机器人, 将 ip 切换为自己的 ip

```

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

```

9.     int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.    //开启调试模式
11.    jakaAPI.set_debug_mode(ref handle, true);
12.    Console.WriteLine("debug :true");
13.    //机器人上电
14.    jakaAPI.power_on(ref handle);
15.    //机器人上使能
16.    jakaAPI.enable_robot(ref handle);
17.    jakaAPI.destory_handler(ref handle);
18. }

```

## 4.66 机器人运动终止

```

1.  /**
2.   * @brief 终止当前机械臂运动
3.   * @return ERR_SUCC 成功 其他失败
4.   */
5.  int motion_abort(ref int handle);

```

代码示例:

```

1.  /// <summary>
2.  /// 运动终止
3.  /// </summary>
4.  static void example_motion_abort()
5.  {
6.      int handle = 0;
7.      int ret = 0;
8.      //实例化机器人, 将ip 切换为自己的ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     JKTYPE.JointValue joint_pos = new JKTYPE.JointValue();
11.     //机器人上电
12.     jakaAPI.power_on(ref handle);
13.     //机器人上使能
14.     jakaAPI.enable_robot(ref handle);
15.     //定义并初始化 JointValue 变量
16.     Console.WriteLine("start_move");
17.     joint_pos.jVal = new double[] { 0 * PI / 180, 0 * PI / 180, 50 * PI / 180, 0 * PI / 180, 0 * PI / 180, 0 * PI / 180 };
18.     //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
19.     jakaAPI.joint_move(ref handle, ref joint_pos, JKTYPE.MoveMode.ABS, false, 1);
20.     System.Threading.Thread.Sleep(500);
21.     //运动 0.5s 后终止
22.     jakaAPI.motion_abort(ref handle);
23.     Console.WriteLine("stop_move");

```

```
24.     jakaAPI.destory_handler(ref handle);  
25. }
```

## 4.67 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.  int set_errorcode_file_path(ref int handle, char[] path);
```

## 4.68 获取机器人目前发生的最后一个错误码

```
1.  /**  
2.  * @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  
3.  * @return ERR_SUCC 成功 其他失败  
4.  */  
5.  int get_last_error(ref int handle, ref JKTYPE.ErrorCode code);
```

## 4.69 设置轨迹复现配置参数

```
1.  /**  
2.  * @brief 设置轨迹复现配置参数  
3.  * @param para 轨迹复现配置参数  
4.  * @return ERR_SUCC 成功 其他失败  
5.  */  
6.  int set_traj_config(ref int handle, ref JKTYPE.TrajTrackPara para);
```

## 4.70 获取轨迹复现配置参数

```
1.  /**  
2.  * @brief 获取轨迹复现配置参数  
3.  * @param para 轨迹复现配置参数  
4.  * @return ERR_SUCC 成功 其他失败  
5.  */  
6.  int get_traj_config(ref int handle, ref JKTYPE.TrajTrackPara para);
```

## 4.71 采集轨迹复现数据控制开关

```

1.  /**
2.  * @brief 采集轨迹复现数据控制开关
3.  * @param mode 选择 TRUE 时，开始数据采集，选择 FALSE 时，结束数据采集
4.  * @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
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int set_traj_sample_mode(ref int handle, ref bool mode, char[] filename);

```

代码示例：

```

1.  /// <summary>
2.  /// 轨迹复现
3.  /// </summary>
4.  static void example_traj_sample()
5.  {
6.      bool samp_stu = false;
7.      int handle = 0;
8.      int ret = 0;
9.      //实例化机器人，将 ip 切换为自己的 ip
10.     int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
11.     jakaAPI.set_debug_mode(ref handle, true);
12.     JKTYPE.JointValue joint_pos = new JKTYPE.JointValue();
13.     joint_pos.jVal = new double[] { 1, 1, 1, 1, 1, 1 };
14.     JKTYPE.JointValue joint_pos2 = new JKTYPE.JointValue();
15.     joint_pos2.jVal = new double[] { -1, 1, 1, 1, 1, 1 };
16.     //机器人上电
17.     jakaAPI.power_on(ref handle);
18.     //机器人上使能
19.     jakaAPI.enable_robot(ref handle);
20.     char[] name = new char[40];
21.     char[] trajname = new char[40];
22.     name = "testxx".ToCharArray();
23.     trajname = "track/testxx".ToCharArray();
24.     jakaAPI.joint_move(ref handle, ref joint_pos2, JKTYPE.MoveMode.ABS, true, 1);
25.     //开启轨迹复现数据采集开关
26.     jakaAPI.set_traj_sample_mode(ref handle, true, name);
27.     //查询轨迹复现采集状态
28.     jakaAPI.get_traj_sample_status(ref handle, ref samp_stu);
29.
30.     //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
31.     jakaAPI.joint_move(ref handle, ref joint_pos, JKTYPE.MoveMode.ABS, true, 1);
32.     jakaAPI.joint_move(ref handle, ref joint_pos2, JKTYPE.MoveMode.ABS, true, 1);
33.     jakaAPI.set_traj_sample_mode(ref handle, false, name);
34.     System.Threading.Thread.Sleep(500);

```

```

35.   jakaAPI.generate_traj_exe_file(ref handle, name);
36.   System.Threading.Thread.Sleep(500);
37.   jakaAPI.program_load(ref handle, trajname);
38.   System.Threading.Thread.Sleep(100);
39.   jakaAPI.program_run(ref handle);
40.   jakaAPI.destory_handler(ref handle);
41. }

```

## 4.72 采集轨迹复现数据状态查询

```

1.  /**
2.  * @brief 采集轨迹复现数据状态查询
3.  * @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
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int  get_traj_sample_status(ref int handle, ref bool sample_status);

```

## 4.73 查询控制器中已经存在的轨迹复现数据的文件名

```

1.  /**
2.  * @brief 查询控制器中已经存在的轨迹复现数据的文件名
3.  * @param filename 控制器中已经存在的轨迹复现数据的文件名
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int  get_exist_traj_file_name(ref int handle, ref JKTYPE.MultStrStorType filename);

```

## 4.74 重命名轨迹复现数据的文件名

```

1.  /**
2.  * @brief 重命名轨迹复现数据的文件名
3.  * @param src 原文件名
4.  * @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
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int  rename_traj_file_name(ref int handle, ref char[] src,ref char[] dest);

```

## 4.75 删除控制器中轨迹复现数据文件

```

1.  /**

```

```

2.  * @brief 删除控制器中轨迹复现数据文件
3.  * @param filename 要删除的文件的文件名，文件名为数据文件名字
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int remove_traj_file(ref int handle, ref char[] filename);

```

## 4.76 控制器中轨迹复现数据文件生成控制器执行脚本

```

1.  /**
2.  * @brief 控制器中轨迹复现数据文件生成控制器执行脚本
3.  * @param filename 数据文件的文件名，文件名为数据文件名字，不带后缀
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int generate_traj_exe_file(ref int handle, ref char[] filename);

```

## 4.77 机器人扩展关节运动

```

1.  /**
2.  * @brief 机器人关节运动
3.  * @param joint_pos 机器人关节运动目标位置
4.  * @move_mode 指定运动模式：增量运动(相对运动)或绝对运动
5.  * @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
6.  * @param speed 机器人关节运动速度，单位：rad/s
7.  * @param acc 机器人关节运动角加速度
8.  * @param tol 机器人关节运动终点误差
9.  * @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
10. * @return ERR_SUCC 成功 其他失败
11. */
12. int joint_move_extend(ref int handle, const JointValue* joint_pos, MoveMode move_mode, BO
    OL is_block, double speed, double acc, double tol, const OptionalCond* option_cond);

```

## 4.78 机器人扩展末端直线运动

```

1.  /**
2.  * @brief 机器人末端直线运动
3.  * @param end_pos 机器人末端运动目标位置
4.  * @move_mode 指定运动模式：增量运动(相对运动)或绝对运动
5.  * @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
6.  * @param speed 机器人直线运动速度，单位：mm/s
7.  * @param acc 机器人关节运动角加速度

```

```

8. * @param tol 机器人关节运动终点误差
9. * @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
10. * @return ERR_SUCC 成功 其他失败
11. */
12. int linear_move_extend(ref int handle, const CartesianPose* end_pos, MoveMode move_mode,
    BOOL is_block, double speed, double accel, double tol, const OptionalCond* option_cond);

```

## 4.79 机器人圆弧运动

```

1. /**
2. * @brief 机器人末端圆弧运动
3. * @param end_pos 机器人末端运动目标位置
4. * @param mid_pos 机器人末端运中间点
5. * @move_mode 指定运动模式: 增量运动(相对运动)或绝对运动
6. * @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
7. * @param speed 机器人直线运动速度, 单位: mm/s
8. * @param acc 机器人关节运动角加速度
9. * @param tol 机器人关节运动终点误差
10. * @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
11. * @return ERR_SUCC 成功 其他失败
12. */
13. int circular_move(ref CartesianPose end_pos, ref CartesianPose mid_pos, int move_mode, BO
    OL is_block, double speed, double accel, double tol, ref OptionalCond option_cond);

```

代码示例:

```

1. /// <summary>
2. /// 圆弧运动
3. /// </summary>
4. static void example_circle_move()
5. {
6.     JKTYPE.OptionalCond opt = new JKTYPE.OptionalCond();
7.     JKTYPE.CartesianPose end_p = new JKTYPE.CartesianPose(), mid_p = new JKTYPE.CartesianP
    ose();
8.     end_p.tran.x = -200; end_p.tran.y = 400; end_p.tran.z = 400;
9.     end_p.rpy.rx = -90 * PI / 180; end_p.rpy.ry = 0 * PI / 180; end_p.rpy.rz = 0 * PI / 18
    0;
10.    mid_p.tran.x = -300; mid_p.tran.y = 400; mid_p.tran.z = 500;
11.    mid_p.rpy.rx = -90 * PI / 180; mid_p.rpy.ry = 0 * PI / 180; mid_p.rpy.rz = 0 * PI / 18
    0;
12.    int handle = 0;
13.    int ret = 0;
14.    //实例化机器人, 将 ip 切换为自己的 ip
15.    int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
16.    //机器人上电

```

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



```

17.   jakaAPI.power_on(ref handle);
18.   //机器人上使能
19.   jakaAPI.enable_robot(ref handle);
20.   //定义并初始化 JointValue 变量
21.   JKTYPE.JointValue joint_pos = new JKTYPE.JointValue();
22.   joint_pos.jVal = new double[] { 85.76 * PI / 180, -6.207 * PI / 180, 111.269 * PI / 180, 74.938 * PI / 180, 94.24 * PI / 180, 0 * PI / 180 };
23.   //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
24.   jakaAPI.joint_move(ref handle, ref joint_pos, JKTYPE.MoveMode.ABS, true, 1);
25.   //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。
26.   jakaAPI.circular_move(ref handle, ref end_p, ref mid_p, JKTYPE.MoveMode.ABS, true, 20, 1, 0.1, ref opt);
27. }

```

## 4.80 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.  int servo_move_use_none_filter(ref int handle);

```

代码示例:

```

1.   /// <summary>
2.   /// 禁用滤波器
3.   /// </summary>
4.   static void example_servo_use_none_filter()
5.   {
6.       int ret;
7.       int handle = 0;
8.       //实例化机器人, 将 ip 切换为自己的 ip
9.       int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.      //机器人上电
11.      jakaAPI.power_on(ref handle);
12.      //机器人上使能
13.      jakaAPI.enable_robot(ref handle);
14.      ret = jakaAPI.servo_move_use_none_filter(ref handle);
15.  }

```

## 4.81 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,默认 0.5hz
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int servo_move_use_joint_LPF(ref int handle, double cutoffFreq);

```

代码示例:

```

1.  /// <summary>
2.  /// 关节一阶低通滤波
3.  /// </summary>
4.  static void example_servo_use_joint_LPF()
5.  {
6.      int ret;
7.      int handle = 0;
8.      //实例化机器人,将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //机器人上电
11.     jakaAPI.power_on(ref handle);
12.     //机器人上使能
13.     jakaAPI.enable_robot(ref handle);
14.     //servo 模式下关节空间一阶低通滤波,截止频率 0.5Hz
15.     ret = jakaAPI.servo_move_use_joint_LPF(ref handle, 0.5);
16. }

```

## 4.82 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.  int servo_move_use_joint_NLF(ref int handle, double max_vr, double max_ar, double max_jr)
9.  ;

```

代码示例:

```

1.  /// <summary>
2.  /// 关节非线性
3.  /// </summary>
4.  static void example_servo_use_joint_NLF()

```

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

```

5. {
6.     int ret;
7.     int handle = 0;
8.     //实例化机器人, 将 ip 切换为自己的 ip
9.     int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.    //机器人上电
11.    jakaAPI.power_on(ref handle);
12.    //机器人上使能
13.    jakaAPI.enable_robot(ref handle);
14.    //servo 模式下笛卡尔空间非线性滤波
15.    ret = jakaAPI.servo_move_use_joint_NLF(ref handle, 2, 2, 4);
16. }

```

## 4.83 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
8.  * @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
9.  * @return ERR_SUCC 成功 其他失败
10. */
11. int servo_move_use_carte_NLF(ref int handle, double max_vp, double max_ap, double max_jp,
    double max_vr, double max_ar, double max_jr);

```

代码示例:

```

1.    /// <summary>
2.    /// 笛卡尔空间非线性滤波
3.    /// </summary>
4.    static void example_servo_use_carte_NLF()
5.    {
6.        int handle = 0;
7.        int ret = 0;
8.        //实例化机器人, 将 ip 切换为自己的 ip
9.        int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.       //机器人上电
11.       jakaAPI.power_on(ref handle);
12.       //机器人上使能
13.       jakaAPI.enable_robot(ref handle);
14.       //servo 模式下笛卡尔空间非线性滤波
15.       ret = jakaAPI.servo_move_use_carte_NLF(ref handle, 2, 2, 4, 2, 2, 4);

```

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

```
16. }
```

## 4.84 机器人 SERVO 模式下关节空间多阶均值滤波

```
1.  /**
2.   * @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
3.   * @param handle 机器人控制句柄
4.   * @param max_buf 均值滤波器缓冲区的大小
5.   * @param kp 加速度滤波系数
6.   * @param kv 速度滤波系数
7.   * @param ka 位置滤波系数
8.   * @return ERR_SUCC 成功 其他失败
9.   */
10. int servo_speed_foresight(ref int i, int max_buf, double kp);
```

代码示例:

```
1.  /// <summary>
2.  /// 关节空间多阶均值滤波
3.  /// </summary>
4.  static void example_servo_use_joint_MMF()
5.  {
6.      int handle = 0;
7.      int ret = 0;
8.      //实例化机器人, 将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //机器人上电
11.     jakaAPI.power_on(ref handle);
12.     //机器人上使能
13.     jakaAPI.enable_robot(ref handle);
14.     //servo 模式下关节空间多阶均值滤波
15.     ret = jakaAPI.servo_move_use_joint_MMF(ref handle, 200, 2, 4, 2);
16. }
```

## 4.85 机器人 SERVO 模式速度前瞻参数设置

```
1.  /**
2.   * @brief SERVO 模式下速度前瞻参数设置
3.   * @param handle 机器人控制句柄
4.   * @param max_buf 缓冲区的大小
5.   * @param kp 加速度滤波系数
6.   * @return ERR_SUCC 成功 其他失败
7.   */
```

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

```
8. int servo_speed_foresight(ref int i, int max_buf, double kp);
```

代码示例：

```
1.  /// <summary>
2.  /// 速度前瞻
3.  /// </summary>
4.  static void example_speed_foresight()
5.  {
6.      int handle = 0;
7.      int ret = 0;
8.      //实例化机器人, 将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("192.168.2.160".ToCharArray(), ref handle);
10.     //机器人上电
11.     jakaAPI.power_on(ref handle);
12.     //机器人上使能
13.     jakaAPI.enable_robot(ref handle);
14.     //servo 模式下速度前瞻
15.     ret = jakaAPI.servo_speed_foresight(ref handle, 200, 2);
16. }
```

## 4.86 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. int set_SDK_filepath(ref int handle, ref char[] filepath);
```

## 4.87 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
5.  */
6. int set_torsenosr_brand(ref int handle, int sensor_brand);
```

## 4.88 获取传感器品牌

```
1. /**
```

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

```

2.  * @brief 获取传感器品牌
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
5.  */
6.  int get_torsenosr_brand(ref int handle, ref int sensor_brand);

```

## 4.89 开启或关闭力矩传感器

```

1.  /**
2.  * @brief 开启或关闭力矩传感器
3.  * @param sensor_mode 0 代表关闭传感器，1 代表开启力矩传感器
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int set_torque_sensor_mode(ref int handle, int sensor_mode);

```

## 4.90 设置柔顺控制参数

```

1.  /**
2.  * @brief 设置柔顺控制参数
3.  * @param axis 代表配置哪一轴，可选值为 0~5
4.  * @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
5.  * @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
6.  * @param ftReboundFK 回弹力，表示机器人回到初始状态的能力
7.  * @param ftConstant 代表恒力，手动操作时全部设置为 0
8.  * @param ftNnormalTrack 法向跟踪，手动操作时全部设置为 0，
9.  * @return ERR_SUCC 成功 其他失败
10. */
11. int set_admit_ctrl_config(ref int handle, int axis, int opt, int ftUser, int ftConstant, int ftNnormalTrack, int ftReboundFK);

```

## 4.91 开始辨识工具末端负载

```

1.  /**
2.  * @brief 开始辨识工具末端负载
3.  * @param joint_pos 使用力矩传感器进行自动负载辨识时的结束位置
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int start_torq_sensor_payload_identify(ref int handle, ref JointValue joint_pos);

```

代码示例：

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

```
1.  /// <summary>
2.  /// 力控传感器（额外配件）辨识负载
3.  /// </summary>
4.  static void example_sensor_payload()
5.  {
6.      Console.WriteLine("sensor in\n");
7.      JKTYPE.JointValue joint_pos= new JKTYPE.JointValue();
8.      JKTYPE.PayLoad pl = new JKTYPE.PayLoad(), pl_ret= new JKTYPE.PayLoad();
9.      int handle = 0;
10.     int ret = 0;
11.     //实例化机器人, 将ip 切换为自己的ip
12.     Console.WriteLine("login be\n");
13.     int result = jakaAPI.create_handler("10.5.5.100".ToCharArray(), ref handle);
14.     Console.WriteLine("login finish\n");
15.     //机器人上电
16.     jakaAPI.power_on(ref handle);
17.     //机器人上使能
18.     jakaAPI.enable_robot(ref handle);
19.     Console.WriteLine("enable finish\n");
20.     ret = jakaAPI.set_torsenosr_brand(ref handle, 2); //设置传感器品牌
21.     ret = jakaAPI.set_torque_sensor_mode(ref handle, 1); //开启力控传感器
22.     jakaAPI.get_joint_position(ref handle, ref joint_pos);
23.     joint_pos.jVal[3] += PI / 4;
24.     if (joint_pos.jVal[3] > 265 * PI / 180)
25.         joint_pos.jVal[3] -= PI / 2;
26.     joint_pos.jVal[4] += PI / 4;
27.     if (joint_pos.jVal[4] > 320 * PI / 180)
28.         joint_pos.jVal[4] -= PI / 2;
29.     joint_pos.jVal[5] += PI / 4;
30.     if (joint_pos.jVal[5] > 265 * PI / 180)
31.         joint_pos.jVal[5] -= PI / 2;
32.     //开始辨识传感器负载
33.     ret = jakaAPI.start_torq_sensor_payload_identify(ref handle, ref joint_pos);
34.     do
35.     {
36.         //查询传感器负载状态
37.         jakaAPI.get_torq_sensor_identify_staus(ref handle, ref ret);
38.         Console.WriteLine("{0}", ret);
39.         System.Threading.Thread.Sleep(1000);
40.     } while (1 == ret);
41.     //获取辨识结果
42.     ret = jakaAPI.get_torq_sensor_payload_identify_result(ref handle, ref pl);
43.     //设置传感器末端负载
44.     Console.WriteLine("payload: {0} {1} {2} {3}", pl.mass, pl.centroid.x, pl.centroid.y, pl.centroid.z);
```

```
45.     ret = jakaAPI.set_torq_sensor_tool_payload(ref handle, ref pl);
46.     System.Threading.Thread.Sleep(10000);
47.     //获取当前设置的传感器末端负载
48.     ret = jakaAPI.get_torq_sensor_tool_payload(ref handle, ref pl_ret);
49.     System.Threading.Thread.Sleep(10000);
50.     jakaAPI.destory_handler(ref handle);
51. }
```

## 4.92 获取末端负载辨识状态

```
1.  /**
2.  * @brief 获取末端负载辨识状态
3.  * @param identify_status 0 代表辨识完成, 1 代表未完成, 2 代表辨识失败
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int get_torq_sensor_identify_staus(ref int handle, ref int identify_status);
```

## 4.93 获取末端负载辨识结果

```
1.  /**
2.  * @brief 获取末端负载辨识结果
3.  * @param payload 末端负载
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int get_torq_sensor_payload_identify_result(ref int handle, ref PayLoad payload);
```

## 4.94 设置传感器末端负载

```
1.  /**
2.  * @brief 设置传感器末端负载
3.  * @param payload 末端负载
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int set_torq_sensor_tool_payload(ref int handle, ref PayLoad payload);
```



## 4.95 获取传感器末端负载

```

1.  /**
2.  * @brief 获取传感器末端负载
3.  * @param payload 末端负载
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int get_torq_sensor_tool_payload(ref int handle, ref PayLoad payload);

```

## 4.96 力控拖拽使能（导纳控制）

```

1.  /**
2.  * @brief 力控拖拽使能
3.  * @param enable_flag 0 为关闭力控拖拽使能，1 为开启
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int enable_admittance_ctrl(ref int handle, int enable_flag);

```

代码示例：

```

1.  /// <summary>
2.  /// 力控传感器（额外配件）导纳控制，z 轴
3.  /// </summary>
4.  static void example_enable_admittance_ctrl()
5.  {
6.      int handle = 0;
7.      int ret;
8.      //实例化机器人，将 ip 切换为自己的 ip
9.      int result = jakaAPI.create_handler("10.5.5.100".ToCharArray(), ref handle);
10.     //机器人上电
11.     jakaAPI.power_on(ref handle);
12.     //机器人上使能
13.     jakaAPI.enable_robot(ref handle);
14.     //品牌,力控传感器品牌询问服务人员
15.     jakaAPI.set_torsenosr_brand(ref handle, 2);
16.     //开启力控传感器
17.     jakaAPI.set_torque_sensor_mode(ref handle, 1);
18.     //初始化传感器
19.     jakaAPI.set_compliant_type(ref handle, 1, 1);
20.     Console.WriteLine("inint sensor comple\n");
21.     //设置柔顺控制参数，z 轴向下 5N 恒力
22.     ret = jakaAPI.set_admit_ctrl_config(ref handle, 0, 0, 20, 5, 0, 0); //x 轴
23.     ret = jakaAPI.set_admit_ctrl_config(ref handle, 1, 0, 20, 5, 0, 0); //y 轴
24.     ret = jakaAPI.set_admit_ctrl_config(ref handle, 2, 3, 20, 5, 0, 0); //z 轴
25.     ret = jakaAPI.set_admit_ctrl_config(ref handle, 3, 0, 20, 5, 0, 0); //rx

```

```

26.     ret = jakaAPI.set_admit_ctrl_config(ref handle, 4, 0, 20, 5, 0, 0); //ry
27.     ret = jakaAPI.set_admit_ctrl_config(ref handle, 5, 0, 20, 5, 0, 0); //rz
28.     //设置力控拖拽使能, 1 打开, 0 关闭
29.     ret = jakaAPI.enable_admittance_ctrl(ref handle, 1);
30.     Console.WriteLine("enable_admittance_ctrl open! \n");
31.     Console.WriteLine("input any word to quit:\n");
32.     ret = Console.Read();
33.     Console.WriteLine("quit!!");
34.     ret = jakaAPI.enable_admittance_ctrl(ref handle, 0);
35.     ret = jakaAPI.set_admit_ctrl_config(ref handle, 2, 0, 20, 5, 0, 0);
36.     jakaAPI.set_torque_sensor_mode(ref handle, 0);
37.     Console.WriteLine("close\n");
38.     jakaAPI.destory_handler(ref handle);
39. }

```

## 4.97 设置力控类型和传感器初始化状态

```

1.  /**
2.  * @brief 设置力控类型和传感器初始化状态
3.  * @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
4.  * @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
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int set_compliant_type(ref int handle, int sensor_compensation, int compliance_type);

```

## 4.98 获取力控类型和传感器初始化状态

```

1.  /**
2.  * @brief 获取力控类型和传感器初始化状态
3.  * @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
4.  * @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
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int get_compliant_type(ref int handle, ref int sensor_compensation, ref int compliance_type);

```

## 4.99 获取力控柔顺控制参数

```

1.  /**
2.  * @brief 获取力控柔顺控制参数
3.  * @param handle 机器人控制句柄

```

```

4.  * @param admit_ctrl_cfg 机器人力控柔顺控制参数存储地址
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int get_admit_ctrl_config(ref int i, ref JKTYPE.RobotAdmitCtrl admit_ctrl_cfg);

```

## 4.100 设置力控传感器通信参数

```

1.  /**
2.  * @brief 设置力控传感器通信参数
3.  * @param handle 机器人控制句柄
4.  * @param type 0 为使用 tcp/ip 协议, 1 为使用 RS485 协议
5.  * @param ip_addr 为力控传感器地址
6.  * @param port 为使用 tcp/ip 协议时力控传感器端口号
7.  * @return ERR_SUCC 成功 其他失败
8.  */
9.  int set_torque_sensor_comm(ref int i, int type, ref char[] ip_addr, int port);

```

## 4.101 获取力控传感器通信参数

```

1.  /**
2.  * @brief 获取力控传感器通信参数
3.  * @param handle 机器人控制句柄
4.  * @param ip_addr 为力控传感器地址
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int get_torque_sensor_comm(ref int i, ref int type, ref byte ip_addr, ref int port);

```

在 c# 中使用时, 需要将 byte 转为 char:

```
string strGet = System.Text.Encoding.Default.GetString(ip_addr, 0, ip_addr.length());
;
```

## 4.102 关闭力控

```

1.  /**
2.  * @brief 关闭力控
3.  * @param handle 机器人控制句柄
4.  * @return ERR_SUCC 成功 其他失败
5.  */
6.  int disable_force_control(ref int i);

```

## 4.103 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 handle 机器人控制句柄
4.  * @param vel_cfg 为速度柔顺控制参数
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int set_vel_compliant_ctrl(ref int i, ref JKTYPE.VelCom vel);
```

## 4.104 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 handle 机器人控制句柄
4.  * @param ft 为柔顺控制力矩条件
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int set_compliance_condition(ref int i, ref JKTYPE.FTxyz ft);
```

## 4.105 设置网络异常，机器人终止运动等待时间

```
1.  /**
2.  * @brief 设置网络异常，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 handle 机器人控制句柄
4.  * @param millisecond 时间参数，单位毫秒
5.  * @param mnt 网络异常时机器人需要进行的动作类型
6.  * @return ERR_SUCC 成功 其他失败
7.  */
8.  int set_network_exception_handle(ref int i, float millisecond, JKTYPE.ProcessType mnt);
```

## 4.106 设置机器人状态数据自动更新时间间隔

```
1.  /**
2.  * @brief 设置机器人状态数据自动更新时间间隔
3.  * @param handle 机器人控制句柄
4.  * @param millisecond 时间参数，单位毫秒
5.  * @return ERR_SUCC 成功 其他失败
6.  */
7.  int set_status_data_update_time_interval(ref int i, float millisecond);
```

代码示例：

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

```
1.  ///
```

## 4.107 初始化 FTP 客户端

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

## 4.108 FTP 上传

```
1.  /**
2.   * @brief 从本地上传指定类型和名称的文件到控制器
3.   * @param remote 上传到控制器内部文件名绝对路径，若为文件夹需要以 “\” 或 “/” 结尾
4.   * @param local 本地文件名绝对路径
5.   * @param opt 1 单个文件 2 文件夹
6.   * @return ERR_SUCC 成功 其他失败
7.   */
8.     errno_t upload_file(ref int i, char[] local, char[] remote, int opt);
```

## 4.109 FTP 下载

```
1.  /**
2.   * @brief 从控制器下载指定类型和名称的文件到本地
```

```
3.      * @param remote 控制器内部文件名绝对路径，若为文件夹需要以 “\” 或 “/” 结尾
4.      * @param local 下载到本地文件名绝对路径
5.      * @param opt 1 单个文件 2 文件夹
6.      * @return ERR_SUCC 成功 其他失败
7.      */
8.      errno_t download_file(ref int i,char[] local, char[] remote, int opt);
```

## 4.110 FTP 目录查询

```
1.      /**
2.      * @brief 查询控制器指定目录下的结构
3.      * @param remote 控制器内部文件名原名称，查询轨迹 “/track/” ,查询脚本程序 “/program/”
4.      * @param type 0 文件名和子目录名，1 文件名，2 子目录名
5.      * @param ret 查询结果
6.      * @return ERR_SUCC 成功 其他失败
7.      */
8.      errno_t get_ftp_dir(ref int i,const char[] remotedir, int type, ref char[] ret);
```

## 4.111 FTP 删除

```
1.      /**
2.      * @brief 从控制器删除指定类型和名称的文件
3.      * @param remote 控制器内部文件名
4.      * @param opt 1 单个文件 2 文件夹
5.      * @return ERR_SUCC 成功 其他失败
6.      */
7.      errno_t del_ftp_file(ref int i,char[] remote, int opt);
```

## 4.112 FTP 重命名

```
1.      /**
2.      * @brief 重命名控制器指定类型和名称的文件
3.      * @param remote 控制器内部文件名原名称
4.      * @param des 重命名的目标名
5.      * @param opt 1 单个文件 2 文件夹
6.      * @return ERR_SUCC 成功 其他失败
7.      */
8.      errno_t rename_ftp_file(ref int i,char[] remote, char[] des, int opt);
```

## 4.113 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(ref int i);
```

## 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](mailto:support@jaka.com)，我们的同事会尽量一一回复。