C++ Function Manual

Revision: V2.1.7

Date: 2024.01.28

Shanghai JAKA Robotics Ltd
Tel ： +400 006 2665 | Web:www.jaka.com
Building 33-35, 610,Jianchuan Road, Minhang District,Shanghai
Building 10, No.377 South Wuyi Rd, Changzhou, Jiangsu

Property Description

Revision History

Revision No.	Date of Revision	Supported Versions	Description
V1.0.0	2020.05.27	V1.4.10/V2.0.10 and above	Create
V1.0.1	2020.07.17	V1.4.10/V2.0.10 and above	Add 3.20, 3.21, 4.63-4.67
V1.0.2	2020.09.25	V1.4.10/V2.0.10 and above	Add 3.22, 3.23, 4.68--4.75
V1.0.3	2020.10.22	V1.4.10/V2.0.10 and above	Add 3.24, 4.76--4.78
V1.0.4	2020.11.25	V1.4.12/V2.0.12 and above	Fix the move blocking problem and add 4.79-4.83
V1.0.5	2020.12.08	V1.4.12/V2.0.12 and above	Add 3.20-3.22, 4.15, 4.17, 4.86
V1.0.6	2020.01.18	V1.4.24/V2.0.24 and above	Revise 3.21, add 4.36, 4.30, 4.89-4.100
V1.0.7	2021.04.27	V1.4.24/V1.5.12.17/V2.0.24 and above	Add 3.28-3.32, add 4.101-4.109
V1.0.8	2021.08.30	V1.4.24/V1.5.12.17/V2.0.24 and above	Add API use instructions
V2.1.1	2021.12.10	V1.4.24/V1.5.12.17/V2.0.24 and above	Add FTP interface
V2.1.2	2022.7.1	V1.4.24/V1.5.13.08/V2.0.24 add above	Modify the structure of catalog; Add API usage instructions
V2.1.3	2022.11.28	V1.4.24/V1.5.13.08/V2.0.24 add above	Designation the MoveC circles number
V2.1.7	2024.01.28	V1.5.13.08 and above	Add interfaces of getting and setting the robot mounting angle Add interfaces return value -15 Fix the acceleration unit of MoveC

Content

1. Introduction

JAKA API communicates with the robot based on the network communication protocol TCP/IP, with an interface for robotic arm manipulation and supporting four programming languages, python, C, C++, and C#.

JAKA SDK uses C++ for development, and the method of class is the interface for robot arm manipulation.

2. Document Notes

- The unit of length in the interface is unified as millimeter (mm), and the unit of angle is unified as radians (rad).
- How to get version number: Right-click the dll file in windows, select file properties, and Interrogate the version information in the "details" tab. Import the Command in Linux “strings libjakaAPI.so | grep jakaAPI_version” to Interrogate the version number.
- JAKA SDK adopts UTF-8 encoding.
- The window.h in the code example is only used as delay library. Please use other method for the delay in Linux .
- Such commands as joint_move and linear_move can not be used under the Servo mode.

3. Data Structure

3.1 Callback function type

Set the callback function type in case of an robot error

```
/**
* @Brief Robot callback function (int)
*/
typedef void(*CallBackFuncType)(int);
```

3.2 List of return value

```
#define ERR_SUCC                0            //Successful
#define ERR_FUCTION_CALL_ERROR   2            //Abnormal call, abnormal call interface, the controller does not support
#define ERR_INVALID_HANDLER     -1            //Invalid control handle
#define   ERR_INVALID_PARAMETER  -2           //Invalid parameter
#define ERR_COMMUNICATION_ERR   -3            //Communication error
#define ERR_KINE_INVERSE_ERR     -4           //Kine-inverse error
#define ERR_EMERGENCY_PRESSED    -5           //E-stop pressed
#define ERR_NOT_POWERED         -6            //Not power on
#define ERR_NOT_ENABLED         -7            //Not enable
#define ERR_DISABLE_SERVOMODE    -8           //Not in the servo mode
#define ERR_NOT_OFF_ENABLE       -9           //Not turned off
#define ERR_PROGRAM_IS_RUNNING   -10          //No operation is allowed when the program is running
#define ERR_CANNOT_OPEN_FILE     -11          //Unable to open the file, the file does not exist
#define ERR_MOTION_ABNORMAL      -12          //Abnormalities during the running process
#define ERR_FTP_PREFROM         -14           //Abnormal FTP
#define ERR_VALUE_OVERSIZE       -15         //Insufficient reserved memory to store all query information
```

3.3 Return value type

```
typedef int errno_t;      //Interface return value type
```

3.4 Bool type

```
typedef int BOOL;         //Bool type
```

3.5 Cartesian space position data type

```
/**
* @Brief Cartesian space position data type
*/
typedef struct
{
    double x;       ///< x coordinate, unit: mm
    double y;       ///< y coordinate, unit: mm
    double z;       ///< z coordinate, unit: mm
}CartesianTran;
```

3.6 RPY orientation data type

```
/**
* @brief RPY orientation data type
*/
typedef struct
{
  double rx;    ///< Rotation angle around X-axis, unit: rad
  double ry;    ///< Rotation angle around Y-axis, unit: rad
  double rz;    ///< Rotation angle around Z-axis, unit: rad
}Rpy;
```

3.7 Quaternion orientation data type

```
/**
* @brief Quaternion orientation data type
*/
typedef struct
{
  double s;
  double x;
  double y;
  double z;
}Quaternion;
```

3.8 Cartesian space pose type

```
/**
 *@brief Cartesian space pose type
 */
typedef struct
{
  CartesianTran tran;   ///< Cartesian space position
  Rpy rpy;              ///< Cartesian space orientation
}CartesianPose;
```

3.9 Rotation matrix data type

```
/**
* @brief Rotation matrix data type
*/
typedef struct
{
  CartesianTran x;  ///< x-component
  CartesianTran y;  ///< y-component
  CartesianTran z; ///< z-component
}RotMatrix;
```

3.10 Program status enum type

```
/**
* @brief Program status enum type
*/
typedef enum
{
    PROGRAM_IDLE,       ///< The robot stops running
    PROGRAM_RUNNING,    ///< The robot is running
    PROGRAM_PAUSED      ///< The robot is paused
}ProgramStatus;
```

3.11 Coordinate frame selection enum type

```
/**
* @brief Coordinate frame selection enum type
*/
typedef enum
{
```

```
    COORD_BASE,     ///< Base coordinate frame
    COORD_JOINT,    ///< Joint space
    COORD_TOOL      ///< Tool coordinate frame
}CoordType;
```

3.12 Move mode enum type

```
/**
* @brief Movement enum type
*/
typedef enum
{
    ABS = 0,        ///< Absolute move
    INCR            ///< Incremental move
}MoveMode;
```

3.13 System monitor data type

```
/**
* @brief System monitor data type
*/
typedef struct
{
    int scbMajorVersion;            ///<scbMajor version number
    int scbMinorVersion;            ///<scbMinor version number
    int cabTemperature;             ///<Cabinet temperature
    double robotAveragePower;       ///<Average power of control cabinet bus
    double robotAverageCurrent;     ///<Average current of control cabinet bus
    double instCurrent[6];          ///<The instantaneous current of the robot's 6 joint axes
    double instVoltage[6];          ///<The instantaneous voltage of the robot's 6 joint axes
    double instTemperature[6];      ///<The instantaneous temperature of the robot's 6 joint
axes
}SystemMonitorData;
```

3.14 Payload data type

```
/**
* @brief Payload data type
*/
typedef struct
{
    double mass;                ///<Load mass, unit: kg
```

```
    CartesianTran centroid;     ///<Load centroid, unit: mm
}PayLoad;
```

3.15 Joint value data type

```
/**
* @brief Joint value data type
*/
typedef struct
{
    double jVal[6];     ///< 6Joint position value, unit: rad
}JointValue;
```

3.16 I/O type

```
/**
* @brief IOEnum type
*/
typedef enum
{
    IO_CABINET,     ///< Control cabinet panel IO
    IO_TOOL,        ///< Tool IO
    IO_EXTEND       ///< Extend I/O
}IOType;
```

3.17 Robot status data type

```
/**
* @brief Robot status data
*/
typedef struct
{
    BOOL estoped;       ///< Whether to make an emergency stop
    BOOL poweredOn;     ///< Whether to turn on the power supply
    BOOL servoEnabled;  ///< Whether to enable
}RobotStatus;
```

3.18 Torque value type

```
/**
* @brief Torque value type
*/
```

```
typedef struct
{
    double jTorque[6];  ///< Torque value of each joint, unit: N
}TorqueValue;
```

3.19 Joint monitor data type

```
/**
* @brief Robot joint monitor data
*/
typedef struct
{
    double instCurrent;         ///< InstCurrent
    double instVoltage;         ///< instVoltage
    double instTemperature;     ///< InstTemperature
}JointMonitorData;
```

3.20 Robot monitor data type

```
/**
* @brief Robot monitor data type
*/
typedef struct
{
    double scbMajorVersion;                     ///< scbMajor version number
    double scbMinorVersion;                     ///< scbMinor version number
    double cabTemperature;                      ///< Controller temperature, unit: °C
    double robotAveragePower;                   ///< Robot average power, unit: V
    double robotAverageCurrent;                 ///< Robot average current, unit: A
    JointMonitorData jointMonitorData[6];       ///< 6 joints monitor data
}RobotMonitorData;
```

3.21 F/T sensor monitor data type

```
/**
* @brief F/T sensor monitor data type
*/
typedef struct
{
    char ip[20];                        ///< F/T sensor ip address
    int port;                           ///< F/T sensor port number
    PayLoad payLoad;                    ///< Tool payload
```

```
    int status;                         ///< F/T sensor status
    int errcode;                        ///< F/T sensor error code
    double actTorque[6];                ///< F/T sensor actual torque value
    double torque[6];                   ///< F/T sensor torque reading value
    double realTorque[6];               ///< Actual contact force values from the torque sensor (unchanged with initialization options)
}TorqSensorMonitorData;
```

3.22 Robot status monitor data type

```
/**
 * @brief Robot status monitor data, use the get_robot_status function to update the robot status data
 */
typedef struct
{
    int errcode;                                        ///< Error code, 0 means normal operation, others represent abnormal operation
    int inpos;                                          ///< Whether the robot is in position, 0 means robot still not moves to position, 1 means robot has been moved to position
    int powered_on;                                     ///< Whether the robot is powered on, 0 means not powered on, 1 means powered on
    int enabled;                                        ///< Whether the robot is enabled or not, 0 means not enabled, 1 means enabled
    double rapidrate;                                   ///< Robot rapid rate
    int protective_stop;                                ///< Whether it has detected a collision, 0 means no collision detected, 1 means collision detected
    int emergency_stop;                                 ///< Whether the robot has an emergency stop, 0 means no emergency stop, 1 means emergency stop
    int dout[1024];                                     ///< Digital output signal of the robot control cabinet, dout[0] is the number of signals
    int tio_dout[1024];                                 ///< Digital output signal of robot end tool, tio_dout[0] is the number of signals
    int extio[1024];                                    ///< The external application digital output signal of the robot, extio[0] is the number of signals
    int din[1024];                                      ///< Digital input signal of robot control cabinet, din[0] is the number of signals
    int tio_din[1024];                                  ///< Digital input signal of robot end tool, tio_din[0] is the number of signals
    double ain[1024];                                   ///< Robot control cabinet analog input signal, ain[0] is the number of signals
    double tio_ain[1024];                               ///< Robot end tool analog input signal, tio_ain[0] is the number of signals
```

```
    double aout[1024];                                  ///< Robot control cabinet analog output signal, aout[0] is the number of signals
    unsigned int current_tool_id;                       ///< The current tool coordinate frame id
    double cartesiantran_position[6];                   ///< Robot end Cartesian position
    double joint_position[6];                           ///< Robot joint position
    unsigned int on_soft_limit;                         ///< Whether the robot is on limit, 0 means limit protection not triggered, 1 means limit protection triggered
    unsigned int current_user_id;                       ///< The current user coordinate frame  id
    int drag_status;                                    ///< Whether the robot is in drag status, 0 means not in drag status, 1 means in drag status
    RobotMonitorData robot_monitor_data;                ///< Robot status monitor data
    TorqSensorMonitorData torq_sensor_monitor_data;     ///< Robot F/T sensor status monitor data
    int is_socket_connect;                                      ///<whether the connection between SDK and controller is normal, 0 means abnormal connection, 1 means normal connection
}RobotStatus;
```

3.23 Robot error code data type

```
/**
* @brief Robot error code data type
*/
typedef struct
{
    long code;                              ///< Error code:
    char message[120];                      ///< Prompt message corresponding to error code
}ErrorCode;
```

3.24 Trajectory track parameter store data type

```
/**
* @brief Trajectory track parameter store data type
*/
typedef struct
{
    double xyz_interval;                                ///< Space position acquisition accuracy
```

```
    double rpy_interval;                                    ///< Orientation capture accuracy
    double vel;                                             ///< Script running velocity mm/s
    double acc;                                             ///< Script running acceleration mm/s²
}TrajTrackPara;
```

3.25 Multiple string storage data type

```
/**
* @brief Multiple string storage data type
*/
typedef struct
{
    int len;                                                ///< Number of strings
    char name[128][128];                                    ///< Data storage two-dimensional array
}MultStrStorType;
```

3.26 Optional move parameters

```
/**
* @brief Optional parameters
*/
typedef struct
{
    int executingLineId;                                    ///< Control command id
}OptionalCond;
```

3.27 Enum type of robot motion automatic termination due to abnormal network

```
/**
*@brief enum type of robot motion automatic termination due to abnormal network
*/
typedef enum
{
    MOT_KEEP,       ///< Robot keeps original motion when the network is abnormal
    MOT_PAUSE,      ///< Robot pauses motion when the network is abnormal
```

```
    MOT_ABORT       ///< Robot stops moving when the network is abnormal
}ProcessType;
```

3.28 Robot compliance control parameter type

```
/**
* @brief Compliance control parameter type
*/
typedef struct
{
    int opt;      ///< Compliance direction, optional value: 1 2 3 4 5 6, correspond to fx fy fz mx my mz respectively,0 means not checked
    double ft_user; ///< The force of user use to make the robot moves in a certain direction at the maximum speed
    double ft_rebound;   ///< Springback: the force for the robot moves to the initial state
    double ft_constant;  ///< Constant force
    int ft_normal_track; ///< Whether the normal track is turned on, 0 means turn off, 1 means turn on
} AdmitCtrlType;
```

3.29 Robot compliance control parameter type

```
/**
* @brief Compliance control parameter type
*/
typedef struct
{
AdmitCtrlType admit_ctrl[6];
} RobotAdmitCtrl;
```

3.30 Velocity compliance control level and rate level setting

```
/**
* @brief setting the velocity compliance control level and rate level setting
* velocity compliance control has 3 levels, and 1>rate1>rate2>rate3>rate4>0
* When level is 1, can only set rate1 and rate2. The value of rate3 and rate4 is 0
* When level is 2, can only set rate1,rate2 and rate3. The value of rate4 is 0
* When level is 3, can set rate1,rate2, rate3 and rate4
*/
```

```
typedef struct
{
    int vc_level;                    //Velocity compliance control level
    double rate1;                   //Rate1
    double rate2;                                //Rate2
    double rate3;                   //Rate3
    double rate4;                   //Rate4
}VelCom;
```

3.31 Force value and torque value of force sensor

```
/**
* @brief force value and torque value of force sensor
*/
typedef struct
{
    double fx;          // Force value around x-axis, unit: N
    double fy;          // Force value around y-axis, unit: N
    double fz;          // Force value around z-axis, unit: N
    double tx;          // Torque value around x-axis, unit: Nm
    double ty;          // Torque value around y-axis, unit: Nm
    double tz;          // Torque value around z-axis, unit: Nm
}FTxyz;
```

3.32 DH parameters

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

3.33 RS485 Signal Parameter

```
/**
 *  @brief  rs485 Semaphore parameter
 */
```

```
typedef struct
{
    char sig_name[20];// Signal name
    int chn_id;     // RS485 channel ID
    int sig_type;   // Signal type
    int sig_addr;   // Register address
    int value;      // Value, Invalid when setting
    int frequency;  // The refresh frequency of semaphore in the controller is not more than 10
}SignInfo;
```

3.34 RS485 Configuration Parameter

```
/**
 *  @brief rs485RTU  configuration parameter
 */
typedef struct
{
     int chn_id;       // RS485 channelID  chn_id is used as the input parameter when interrogation
    int slaveId;    //  When channel mode is set as Modbus RTU, additional Modbus slave node ID is required, and other modes can be ignored
    int baudrate;   // Baud rate 4800,9600,14400,19200,38400,57600,115200,230400
    int databit;    // Data bit 7, 8
    int stopbit;    // Stop bit 1, 2
    int parity;     // Parity bit 78-> non parity 79->odd parity 69->even parity
}ModRtuComm;
```

4. API

4.1 Basic operation of the robot

4.1.1 Robot control constructor

```
/**
* @brief Robotic arm control constructor
*/
JAKAZuRobot();
```

4.1.2 Robot log in

```
/**
* @brief Connect the robot controller
* @param ip  Controller's ip address
* @return ERR_SUCC Error or Success
*/
errno_t  login_in(const char* ip);
```

4.1.3 Robot log out

```
/**
* @brief Disconnect the controller,After a successful call of this interface, no functions other than login_in can be called.
* @return  ERR_SUCC Error or Success
*/
errno_t  login_out();
```

4.1.4 Power on robot

```
/**
* @brief Power on the robot
* @return  ERR_SUCC Error or Success
*/
errno_t  power_on();
```

4.1.5 Power off robot

```
/**
* @brief Power off the robot
* @return  ERR_SUCC Error or Success
*/
errno_t  power_off();
```

4.1.6 Shutdown robot

```
/**
* @brief Shutdown the control cabinet
* @return  ERR_SUCC Error or Success
```

```
4. */
5. errno_t  shut_down();
```

4.1.7 Enable robot

```
1. /**
2. * @brief Enable the robot
3. * @return  ERR_SUCC Error or Success
4. */
5. errno_t  enable_robot();
```

4.1.8 Disable robot

```
1. /**
2. * @brief Disable the robot
3. * @return  ERR_SUCC Error or Success
4. */
5. errno_t  disable_robot();
```

4.1.9 Enable drag mode

```
1. /**
2. * @brief Enable drag mode
3. * @param enable TRUE means to enter the drag mode, FALSE means to quit the drag mode
4. * @return  ERR_SUCC Error or Success
5. */
6. errno_t  drag_mode_enable(BOOL enable);
```

Sample Code:

```
1. #include <iostream>
2. #include "JAKAZuRobot.h"
3. #include <windows.h>
4. #define PI 3.1415926
5. //drag mode
6. int  example_drag()
7. {
8.     BOOL in_drag;
9.     JAKAZuRobot demo;
10.    demo.login_in("192.168.2.152");
11.    demo.power_on();
12.    demo.enable_robot();
13.    //Confirm the robot whether in drag mode
14.    demo.is_in_drag_mode(&in_drag);
```

```
    std::cout << "in_drag is : " << in_drag << std::endl;
    //Enable the drag mode
    demo.drag_mode_enable(TRUE);
    Sleep(10000);
    demo.is_in_drag_mode(&in_drag);
    std::cout << "in_drag is : " << in_drag << std::endl;
    //Disable the drag mode
    demo.drag_mode_enable(FALSE);
    Sleep(100);
    demo.is_in_drag_mode(&in_drag);
    std::cout << "in_drag is : " << in_drag << std::endl;
    while (1)
    {
        demo.is_in_drag_mode(&in_drag);
        std::cout << "in_drag is : " << in_drag << std::endl;
        Sleep(100);
    }
    return 0;
}
```

4.1.10 Interrogate whether in drag mode

```
/**
* @brief Interrogate whether in drag mode
* @param in_drag Interrogate results
* @return  ERR_SUCC Error or Success
*/
errno_t  is_in_drag_mode(BOOL* in_drag);
```

4.1.11 Get SDK version

```
/**
* @brief Get the controller version number
* @param version SDK version number
* @return  ERR_SUCC Error or Success
*/
errno_t  get_sdk_version(char* version);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Get SDK version
```

```
int example_getsdk_version()
{
    //Instance API object demo
    JAKAZuRobot demo;
    char ver[100];
    //Login the controller, you need to replace 192.168.2.194 with the IP of your own controller
    demo.login_in("192.168.2.194");
    //Get current SDK version
    demo.get_sdk_version(ver);
    std::cout << " SDK version is :" << ver << std::endl;
    return 0;
}
```

4.1.12 Set SDK file path

```
/**
* @brief Set SDK file path
* @param filepath SDK file path
* @return  ERR_SUCC Error or Success
*/
errno_t  set_SDK_filepath(char* filepath);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Set SDK file path
int example_set_SDK_filepath()
{
//Set SDK file path
    char path[20] = "D://";
    int ret;
    JAKAZuRobot demo;
ret = demo.set_SDK_filepath(path);//Set SDK file path
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    std::cout << ret << std::endl;
    return 0;
}
```

4.1.13 Set SDK debug mode

```
/**
* @brief Set whether enter the debug mode. Select TRUE to enter the debug mode. At this time, debugging information will be output in the standard output stream. When selecting FALSE, debugging information will not be output.
* @return  ERR_SUCC Error or Success
*/
errno_t  set_debug_mode(BOOL mode);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Set whether to open SDK debug mode
int example_set_debug_mode()
{
    BOOL mode;
    JAKAZuRobot demo;
    //Set debug mode, which will print debug information on the terminal
    demo.set_debug_mode(TRUE);
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    return 0;
}
```

4.1.14 Get controller IP

```
/**
* @brief Get controller ip
* @param controller_name Controller name
* @param ip_list Controller ip list, when the controller name is a specific value, the corresponding controller IP address will be returned, when the controller name is empty, all controller IP addresses in the network segment class will be returned
* @return  ERR_SUCC Error or Success
* This function is invalid when the app is initiated
*/
errno_t  get_controller_ip(char* controller_name, char* ip_list);
```

Sample Code:

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

```
#define PI 3.1415926
//Get controller IP
int example_get_controller_ip()
{
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    char ip_list[2000] = { 0 };
    char  controller_name1[50] = "";

    //Get controller IP
    ret = demo.get_controller_ip( controller_name1, ip_list);
    std::cout << ret << std::endl;
    std::cout << " ip_list is :\n" << ip_list << std::endl;
    return 0;
}
```

4.2 Robot Move

The motion interfaces in this class, the controllers are involved in motion planning.

4.2.1 Manual mode move

```
/**
* @brief Control the robot's jog move in manual mode
* @param aj_num Represent joint number [0-5] in joint space, and x, y, z, rx, ry, rz-axis in Cartesian space
* @param move_mode Robot move mode, incremental move or absolute move (i.e. continuous jog move) and continuous move, refer to 2.13 to select the right move mode, optional types are INCR ABS
* @param coord_type Robot move coordinate frame, tool coordinate frame, base coordinate frame (current world/user coordinate frame) or joint space, refer to 2.12 to select the right coordinate frame
* @param vel_cmd Command velocity, unit of rotation axis or joint move is rad/s, move axis unit is mm/s
* @param pos_cmd Command position, unit of rotation axis or joint move is rad, move axis unit is mm
* @return  ERR_SUCC Error or Success
*/
errno_t  jog(int aj_num, MoveMode move_mode, CoordType coord_type, double vel_cmd, double pos_cmd);
```

Sample Code:

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

```
#include <windows.h>
#define PI 3.1415926
//Manual mode move
int main()
{
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.152");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
     //Manual motion, where INCR stands for incremental motion, 0.5 means the speed is 0.5rad/s ,30*PI/180 means execute the line command to move 30*PI/180rad.
    demo.jog(1, INCR, COORD_JOINT , 0.5, 30*PI/180);
    Sleep(5000);
//Stop manual mode
    demo.jog_stop(0);
    //Power off the robot
    demo.disable_robot();
    //Disable the robot
    demo.power_off();
    return 0;
}
```

4.2.2 Manual mode stop

```
/**
* @brief Stop the robot in manual mode
* @param num  Robot axis number 0-5, when number is -1, stop all axes
* @return  ERR_SUCC Error or Success
*/
errno_t  jog_stop(int num);
```

4.2.3 Robot joint move

```
/**
* @brief Robot joint move
* @param joint_pos Joint move position
* @param move_mode Specify move mode: incremental move or absolute move
* @param is_block Set whether the interface is a block interface, TRUE represents a block interface and FALSE represents a non-block interface.
```

```
* @param speed Robot joint move speed, unit: rad/s
* @return  ERR_SUCC Error or Success
*/
errno_t  joint_move(const JointValue* joint_pos, MoveMode move_mode,BOOL is_block, double speed);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#define PI 3.1415926
Robot joint mobe , upper limit of joint speed is 180deg/s
int main()
{
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Define and initialize JointValue variables
    JointValue joint_pos = { 45 * PI / 180, 50 * PI / 180, 50 * PI / 180, 0 * PI / 180, 0 * PI / 180, 0 * PI / 180 };
    //Joint space motion, where ABS stands for absolute motion, TRUE means the command is blocked, and 1 stands for a speed of 1 rad/s
    demo.joint_move(&joint_pos, ABS, TRUE, 1);
//Power off the robot
    demo.disable_robot();
    //Disable the robot
    demo.power_off();
    return 0;
}
```

4.2.4 Robot end linear move

```
/**
* @brief Robot end linear move
* @param end_pos Robot end move end position
* @param move_mode Determine move mode: incremental move or absolute move
* @param is_block Set whether the interface is a block interface, TRUE represents a block interface and FALSE represents a non-block interface.
* @param speed Robot linear move speed, unit: mm/s
* @return  ERR_SUCC Error or Success
* Three conditions in which singularities often occur:
```

```
 * The position of the tool's terminal end is on the plane formed by axes Z1 and Z2;
* The axes Z2, Z3 and Z4 are on a same plane;
* The angle of joint 5 is 0 or  180°, that is, the axis Z4 is parallel to axis Z6;
*/
errnot_t  linear_move(const CartesianPose* end_pos, MoveMode move_mode, BOOL is_block, dou
ble speed);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#define PI 3.1415926
//Linear motion at the end of the robot arm, pay attention to avoid singularities
int main()
{
    //Instance API object demo
    JAKAZuRobot demo;
    RobotStatus status;
    //login controller, you need to replace 192.168.2.229 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Define and initialize the CartesianPose variable with the rotation angle in radians.
    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 / 18
0;
    //Cartesian space motion, where ABS stands for absolute motion, TRUE means the command is
blocked, and 10 stands for a speed of 10mm/s
    printf("rx=%f , ry=%f, rz=%f\n", cart.rpy.rx, cart.rpy.ry, cart.rpy.rz);
    demo.linear_move(&cart, ABS, TRUE, 10);
    //Print cartesian space position
    demo.get_robot_status(&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_po
sition[4], status.cartesiantran_position[5]);
    Sleep(1000);
    return 0;
}
```

4.2.5 Robot extension joint move

```
/**
```

```
 * @brief Robot joint move
 * @param joint_pos Joint move position
 * @move_mode Specify move mode: Incremental move (relative move) or absolute move
 * @param is_block Set whether the interface is a block interface, TRUE represents a block interface and FALSE represents a non-block interface.
 * @param speed Robot joint move speed, unit: rad/s
 * @param acc Angular acceleration of robot joint move, unit: rad/s²
 * @param tol The robot joint move end error, this parameter makes a smoother interface between two move segments, and it requires consecutive multiple move segments with non-block interfaces to use this parameter.
 * @param option_cond Optional parameters for robot joints, if not needed, the value can be left unassigned, just fill in a null pointer
* @return  ERR_SUCC Error or Success
*/
errno_t  joint_move(const JointValue* joint_pos, MoveMode move_mode, BOOL is_block, double speed, double acc, double tol, const OptionalCond* option_cond);
```

4.2.6 Robot extension end linear move

```
/**
 * @brief Robot end linear move
 * @param end_pos Robot end move position
 * @move_mode Specify move mode: Incremental move (relative move) or absolute move
 * @param is_block Set whether the interface is a block interface, TRUE represents a block interface and FALSE represents a non-block interface.
 * @param speed Robot linear move speed, unit: mm/s
 * @param acc Robot linear move acceleration, unit: mm/s²
 * @param tol Tolerance of robot joint move end, unit: mm
 * @param option_cond Optional parameters for robot joints, if not needed, the value can be left unassigned, just fill in a null pointer
* @return  ERR_SUCC Error or Success
*/
errno_t linear_move(const CartesianPose* end_pos, MoveMode move_mode, BOOL is_block, double speed, double accel, double tol, const  OptionalCond* option_cond);
```

Sample Code:

```
int example_linear_move()
{
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    RobotStatus status;
    //login controller, you need to replace 192.168.2.229 with the IP of your own controller.
    demo.login_in("192.168.2.160");
    //Power on the robot
```

```
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    ///Define and initialize the CartesianPose variable with the rotation angle in radians.
    CartesianPose cart;
    cart.tran.x = 300; cart.tran.y = 300; cart.tran.z = 100;
    cart.rpy.rx = 180 * PI / 180; cart.rpy.ry = 0 * PI / 180; cart.rpy.rz = 0 * PI / 180;
    //Cartesian space motion, where ABS stands for absolute motion, TRUE means the command is blocked, and 10 stands for a speed of 10mm/s
    printf("rx=%f , ry=%f, rz=%f\n", cart.rpy.rx, cart.rpy.ry, cart.rpy.rz);
    demo.linear_move(&cart, ABS, FALSE, 200, 10 ,1,NULL);
    for (int i = 10; i > 0;i--) {
        cart.tran.x = 150; cart.tran.y = 300;
        //Cartesian space extended motion, where ABS stands for absolute motion, FALSE stands for non-blocking command, 20 stands for maximum velocity of 20mm/s, 10 stands for acceleration of 10mm/s², 5 stands for arc over radius of 5mm
        demo.linear_move(&cart, ABS, FALSE, 20, 10, 5, NULL);
        cart.tran.x = 150; cart.tran.y = 250;
        demo.linear_move(&cart, ABS, FALSE, 20, 10, 5, NULL);
        cart.tran.x = 225; cart.tran.y = 250;
        demo.linear_move(&cart, ABS, FALSE, 20, 10, 5, NULL);
        cart.tran.x = 300; cart.tran.y = 250;
        demo.linear_move(&cart, ABS, FALSE, 20, 10, 5, NULL);
        cart.tran.x = 300; cart.tran.y = 300;
        demo.linear_move(&cart, ABS, FALSE, 20, 10, 5, NULL);
        Sleep(3000);
    }
    demo.login_out();
    return 0;
}
```

4.2.7 Set robot blocking motion timeout

```
/**
* @brief Set robot blocking motion timeout
* @param seconds time parameters，＞0.5/s
* @return ERR_SUCC Error or Success
*/
errno_t set_block_wait_timeout(float seconds);
```

4.2.8 Robot circular move

```
/**
```

```
 * @brief  Arc move at the end of the robot. The interface uses the current point and two points entered to plan a circular trajectory.
 * @param end_pos Robot end move position
 * @param mid_pos The middle point of robot end move
 * @move_mode Specify move mode: Incremental move, absolute move
 * @param is_block Set whether the interface is a block interface, TRUE represents a block interface and FALSE represents a non-block interface.
 * @param speed Robot linear move speed, unit: mm/s
 * @param acc Robot Cartesian space acceleration
 * @param tol End point error of robot Cartesian space motion
 * @param option_cond Optional parameters for robot joints, if not needed, the value can be left unassigned, just fill in a null pointer
 * @return  ERR_SUCC Error or Success
 */
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, int circle_cnt = 0);
```

```
/**
 * @brief  Arc move at the end of the robot
 * @param end_pos Robot end move position
 * @param mid_pos The middle point of robot end move
 * @move_mode Specify move mode: Incremental move (relative move) or absolute move
 * @param is_block Set whether the interface is a block interface, TRUE represents a block interface and FALSE represents a non-block interface.
 * @param speed Robot circular move speed, unit: mm/s
 * @param acc Robot arc move acceleration, unit: mm/s²
 * @param tol Robot joint motion endpoint error, unit: mm
 * @param option_cond Optional parameters for robot joints, if not needed, the value can be left unassigned, just fill in a null pointer
 * @param circle_cnt Specifies the circular move number of the robot. A value of 0 is equivalent to circular_move
 * @return  ERR_SUCC Error or Success
 */
errno_t  circular_move(const JKHD* handle, const CartesianPose* end_pos, const CartesianPose*                                                                mid_pos
MoveMode move_mode, BOOL is_block, double speed, double accel, double tol, const  OptionalCond* option_cond, int circle_cnt = 0);
```

Sample Code:

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

```
#define PI 3.1415926
//The upper limit of joint speed is 180deg/s in circular move
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;

    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Define and initialize JointValue variables
    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 };
    //Joint space motion, where ABS stands for absolute motion, TRUE means the command is blocked, and 1 stands for a speed of 1 rad/s
    demo.joint_move(&joint_pos, ABS, TRUE, 1);
    //Circular motion, where ABS stands for absolute motion, TRUE means the command is blocked, 20 stands for linear speed of 20mm/s, 1 stands for acceleration, 0.1 stands for robot arm endpoint error, and OPT is an optional parameter.
    demo.circular_move(&end_p, &mid_p, ABS, TRUE, 20, 1, 0.1,&opt);
    return 0;
}
```

Sample Code 2:

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

#define PI (3.1415926)
#define PI_2 (1.5707963)

int example_circular_move()
{
 JAKAZuRobot robot;
robot.login_in("192.168.20.138");
robot.power_on();
```

```
robot.enable_robot();

CartesianPose start_pos {-251.054, -48.360, 374.000, PI, 0, PI_2};  // Start point
CartesianPose mid_pos = {-555.050, 116.250, 374.000, PI, 0, PI_2};  // Middle point
CartesianPose end_pos = {-295.050, 267.450, 374.000, PI, 0, PI_2};  // End point

robot.jog_stop(-1); // Stop current joint motion

JointValue ref_jv {0, PI_2, PI_2, PI_2, -PI_2, 0};  // Move to the vicinity of the starting point first.
robot.joint_move(&ref_jv, MoveMode::ABS, true, 20);

JointValue start_jv;
robot.get_joint_position(&ref_jv);  // Get current joint angle
robot.kine_inverse(&ref_jv, &start_pos, &start_jv); //Taking the current joint angle as a reference, calculate the starting joint angle.
robot.joint_move(&start_jv, MoveMode::ABS, true, 80);  // Move to the starting joint angle position

// Specify 3 revolutions
robot.circular_move(&end_pos, &mid_pos, MoveMode::ABS, true, 120, 100, 0.1, &opt, 3);

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

4.2.9 Motion abort

```
/**
* @brief Terminate the current robotic arm move
* @return  ERR_SUCC Error or Success
*/
errno_t  motion_abort();
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Motion abort
int example_motion_abort()
{
    //Instance API object demo
```

```
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Define and initialize JointValue variables
    printf("start_move");
    JointValue joint_pos = { 0 * PI / 180, 0 * PI / 180, 50 * PI / 180, 0 * PI / 180, 0 * PI / 180, 0 * PI / 180 };
    //Joint space motion, where ABS stands for absolute motion, TRUE means the command is blocked, and 1 stands for a speed of 1 rad/s
    demo.joint_move(&joint_pos, ABS, FALSE, 1);
    Sleep(500);
    //Terminate after 0.5s of move
    demo.motion_abort();
    printf("stop_move");
    return 0;
}
```

4.2.10 Interrogate whether in position

```
/**
* @brief Interrogate whether in position
* @param in_pos Interrogate results
* @return  ERR_SUCC Error or Success
*/
errno_t  is_in_pos(BOOL* in_pos);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Check if the robot movement has stopped
int example_is_in_pos()
{
    //Instance API object demo
    JAKAZuRobot demo;
    BOOL in_pos;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.152");
    //Power on the robot
    demo.power_on();
```

```
    //Enable the robot
    demo.enable_robot();
    while (1)
    {
        //Check if the robot movement has stopped
        demo.is_in_pos(&in_pos);
        std::cout << " in_pos is :" << in_pos << std::endl;
        Sleep(200);
    }
    return 0;
}
```

4.3 Set and Obtain Robot Operation Information

4.3.1 Get robot status monitoring data (the only multi-thread safe interface)

```
/**
* @brief Get robot status data, multi-thread safe
* @param status Interrogate result of robot status
* @return  ERR_SUCC Error or Success
*/
errno_t  get_robot_status(RobotStatus* status);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Get robot status monitoring data
int example_get_robot_status()
{
    //Instance API object demo
    JAKAZuRobot demo;
    RobotStatus robstatus;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Get robot status monitoring data
    demo.get_robot_status(&robstatus);
    demo.login_out();
    return 0;
}
```

4.3.2 Set the time interval for automatic update

```
/**
* @brief Set the robot status data auto update interval, set for get_robot_status()
* @param millisecond  Time parameter, unit: ms
* @return ERR_SUCC Error or Success
*/
errno_t set_status_data_update_time_interval(float millisecond);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Set the time interval for automatic update
int example_set_status_data_updata_interval()
{
    float milisec = 100;
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Set condition of compliance torque
    ret = demo.set_status_data_update_time_interval(milisec);
    std::cout << ret << std::endl;
    return 0;
}
```

4.3.3 Get joint angle

```
/**
* @brief Get the current joint angle of the robot arm and save the joint angle matrix in the input parameter joint_position
* @param joint_position Interrogate results of joint angle
* @return  ERR_SUCC Error or Success
*/
errno_t  get_joint_position(JointValue* joint_position);
```

Sample Code:

```cpp
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Get joint angle
int example_get_joint_position()
{
    //Instance API object demo
    JAKAZuRobot demo;
    JointValue jot_pos;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Get joint angle
    demo.get_joint_position(&jot_pos);
    for (int i = 0; i < 6; i++)
    {
        std::cout << "joint [" << i+1 <<"] is :"<< jot_pos.jVal[i] << std::endl;
    }
    return 0;
}
```

4.3.4 Get tcp pose

```cpp
/**
* @brief Get tcp pose
* @param tcp_position Interrogate result of tool end position
* @return  ERR_SUCC Error or Success
*/
errno_t  get_tcp_position(CartesianPose* tcp_position);
```

Sample Code:

```cpp
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Get tcp pose
int example_get_tcp_position()
{
    //Instance API object demo
    JAKAZuRobot demo;
    CartesianPose tcp_pos;
```

```
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Get tcp pose
    demo.get_tcp_position(&tcp_pos);
    std::cout << "tcp_pos is :\n x: " << tcp_pos.tran.x << "  y: " << tcp_pos.tran.y << "  z: " << tcp_pos.tran.z << std::endl;
    std::cout << "rx: " << tcp_pos.rpy.rx << "  ry: " << tcp_pos.rpy.ry << "  rz: " << tcp_pos.rpy.rz << std::endl;
    return 0;
}
```

4.3.5 Set user coordinate frame parameter

```
/**
* @brief Set the parameter of specified user coordinate frame
* @param id The value range of the user coordinate frame number is [1,10]
* @param user_frame Offset value of user coordinate frame
* @param name Alias of user coordinate frame
* @return  ERR_SUCC Error or Success
*/
errno_t  set_user_frame_data(int id, const CartesianPose* user_frame, const char* name);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//View and adjust user coordinate system
int example_user_frame()
{
    int id_ret, id_set;
    id_set = 2;
    CartesianPose tcp_ret, tcp_set;
    char name[50] = "test";
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    //Interrogate the currently used user coordinate system ID
    demo.get_user_frame_id(&id_ret);
    //Get the currently used user coordinate system information
```

```
    demo.get_user_frame_data(id_ret, &tcp_ret);
    printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tran.y, tcp_ret.tran.y);
    printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.rz);
    //Initialize user coordinate system coordinates
    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;
    //Set user coordinate system information
    demo.set_user_frame_data(id_set, &tcp_set, name);
    //Switch coordinats of user coordinate system currently use
    demo.set_user_frame_id(id_set);
    //Interrogate the currently used user coordinate system ID
    demo.get_user_frame_id(&id_ret);
    //Get the set user coordinate system information
    demo.get_user_frame_data(id_ret, &tcp_ret);
    printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tran.y, tcp_ret.tran.y);
    printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.rz);
    return 0;
}
```

4.3.6 Get user coordinate system information

```
/**
* @brief Interrogate user coordinate system information that is currently in use
* @param id user coordinate system ID
* @param tcp Offset value of user coordinate system
* @return ERR_SUCC Error or Success
*/
errno_t  get_user_frame_data(int id, CartesianPose* tcp);
```

4.3.7 Set user coordinate frame ID

```
/**
* @brief Set user coordinate frame ID
* @param id The value range of the user coordinate frame ID is [0,10], where 0 represents the world coordinate frame
* @return  ERR_SUCC Error or Success
*/
errno_t  set_user_frame_id(const int id);
```

4.3.8 Get user coordinate frame ID currently in use

```
/**
* @brief Get user coordinate frame ID currently use
* @param id Result
* @return  ERR_SUCC Error or Success
*/
*/  errno_t  get_user_frame_id(int* id);
```

4.3.9 Set tool data

```
/**
* @brief Set the specified tool
* @param id The range of tool number is [1,10]
* @param tcp Tool coordinate frame is offset relative to flange coordinate frame
* @param name Specify the alias of the tool
* @return  ERR_SUCC Error or Success
*/
errno_t  set_tool_data (int id, const CartesianPose* tcp, const char* name);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Tool coordinate system view and adjustment
int example_tool()
{
    int id_ret,id_set;
    id_set = 2;
    CartesianPose tcp_ret,tcp_set;
    char name[50] = "test";
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    //Interrogate the currently used tool ID
    demo.get_tool_id(&id_ret);
    //Get information about the currently used tool
    demo.get_tool_data(id_ret,&tcp_ret);
    printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tran.y, tcp_ret.tran.y);
    printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.rz);
    //Initialize tool coordinates
    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;
    //Set tool information
    demo.set_tool_data(id_set, &tcp_set, name);
    //Switch the coordinates of the currently used tool
    demo.set_tool_id(id_set);
    //Interrogate the currently used tool ID
    demo.get_tool_id(&id_ret);
    //Get information about the set tools
    demo.get_tool_data(id_ret, &tcp_ret);
    printf("id_using=%d \nx=%f, y=%f, z=%f\n", id_ret, tcp_ret.tran.x, tcp_ret.tran.y, tcp_ret.tran.y);
    printf("rx=%f, ry=%f, rz=%f\n", tcp_ret.rpy.rx, tcp_ret.rpy.ry, tcp_ret.rpy.rz);
    return 0;
}
```

4.3.10 Get tool information

```
/**
* @brief Interrogate the information of the tool currently used
* @param id Interrogate result of tool ID
* @param tcp Tool coordinate system is offset relative to flange coordinate system
* @return ERR_SUCC Error or Success
*/
errno_t  get_tool_data(int* id, CartesianPose* tcp);
```

4.3.11 Get the tool ID currently in use

```
/**
* @brief Get the tool ID currently in use
* @param id Interrogate result of tool ID
* @return  ERR_SUCC Error or Success
*/
errno_t  get_tool_id(int* id);
```

4.3.12 Set tool ID currently in use

```
/**
* @brief Set the ID of the currently used tool. When the network fluctuates, it takes a certain delay to take effect after switching the ID.
* @param id The value range of tool coordinate frame ID is [0,10], 0 means no tools, flange center
```

```
4. * @return  ERR_SUCC Error or Success
5. */
6. errno_t  set_tool_id(const int id);
```

4.3.13 Set digital output variables

```
1. /**
2. * @brief Set DO Value
3. * @param type DO Type
4. * @param index DO Index (starting from 0)
5. * @param value DO Value
6. * @return  ERR_SUCC Error or Success
7. */
8. errno_t  set_digital_output (IO Type, int index, BOOL value);
```

Sample Code:

```
1. #include <iostream>
2. #include "JAKAZuRobot.h"
3. // Set and interrogate digital outputs
4. int main()
5. {
6.     BOOL DO3;
7.     //Instance API object demo
8.     JAKAZuRobot demo;
9.     //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
10.    demo.login_in("192.168.2.152");
11.    //Power on the robot
12.    demo.power_on();
13.    //Get do3 status
14.    demo.get_digital_output(IO_CABINET, 2, &DO3);
15.    printf("D03 = %d\n", DO3);
16.    //io_cabinet is the controller panel IO, 2 represents DO2, and 1 corresponds to the DO value
   to be set.
17.    demo.set_digital_output(IO_CABINET, 2, 1);
18.    Sleep(1000);//Requires window.h delay of 1s
19.    //Get do3 status
20.    demo.get_digital_output(IO_CABINET, 2, &DO3);
21.    printf("D03 = %d\n", DO3);
22.    return 0;
23. }
```

4.3.14 Set analog output variables

```
1. /**
```

```
* @brief Set analog output (AO) value
* @param type AO Type
* @param index AO Index (starting from 0)
* @param value AO Settings
* @return  ERR_SUCC Error or Success
*/
errno_t  set_analog_output (IO Type, int index, float value);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
//Set and interrogate analog output
int main()
{
    JAKAZuRobot demo;
    demo.login_in("192.168.2.152");
    demo.power_on();
    float AO35;
    //Get Ao status
    demo.get_analog_output(IO_CABINET, 34, &AO35);
    printf("A035 = %f\n", AO35);
    /io_cabinet is the controller panel IO, 2 represents DO3, and 1.5 corresponds to the DO value to be set.
    demo.set_analog_output(IO_CABINET, 34, 1.5);
    Sleep(1000);//Requires window.h delay of 1s
    //Get Ao status
    demo.get_analog_output(IO_CABINET, 34, &AO35);
    printf("A035 = %f\n", AO35);
    return 0;
}
```

4.3.15 Get digital input status

```
/**
* @brief Interrogate DI status
* @param type DI Type
* @param index DI Index (starting from 0)
* @param result DI Status Interrogate result
* @return  ERR_SUCC Error or Success
*/
errno_t  get_digital_input (IO Type, int index, BOOL* result);
```

4.3.16 Get digital output status

```
/**
* @brief Interrogate DO status
* @param type DO Type
* @param index DO Index (starting from 0)
* @param result DO Status Interrogate result
* @return  ERR_SUCC Error or Success
*/
errno_t  get_digital_output (IO Type, int index, BOOL* result);
```

4.3.17 Get analog input variables

```
/**
* @brief Get the type of AI value
* @param type AI Type
* @param index AI Index (starting from 0)
* @param result Interrogate result of AI status
* @return  ERR_SUCC Error or Success
*/
errno_t  get_analog_input(IO Type, int index, float* result);
```

4.3.18 Get analog output variables

```
/**
* @brief Get AO value
* @param type AO Type
* @param index AO Index (starting from 0)
* @param result Interrogate result of AO status
* @return  ERR_SUCC Error or Success
*/
errno_t  get_analog_output (IO Type, int index, float* result);
```

4.3.19 Interrogate whether extension IO in running status

```
/**
* @brief Interrogate whether the extension IO module is running
* @param is_running Interrogate results of extension IO module running status
* @return  ERR_SUCC Error or Success
*/
errno_t  is_extio_running (BOOL* is_running);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
//Interrogate status of extension IO
int main()
{
BOOL is_running;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    //Get TIO status
    demo.is_extio_running(&is_running);
    printf("tio = %d\n", is_running);
    return 0;
}
```

4.3.20 Set payload

```
/**
* @brief Set payload
* @param payload Centroid and mass data of payload
* @return  ERR_SUCC Error or Success
*/
errno_t  set_payload(const PayLoad* payload);
```

4.3.21 Get payload data

```
/**
* @brief Get payload data
* @param payload Load Interrogate results
* @return  ERR_SUCC Error or Success
*/
errno_t  get_payload(PayLoad* payload);
```

Sample Code:

```
int example_payload()
{
    //Instance API object demo
    JAKAZuRobot demo;
    PayLoad payloadret;
    PayLoad payload_set;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Get current payload data
```

```
    demo.get_payload(&payloadret);
    std::cout << " payload mass is :" << payloadret.mass << " kg" << std::endl;
    std::cout << " payload center of mass is \nx: " << payloadret.centroid.x<< "y: " << pa
yloadret.centroid.y << "z: " << payloadret.centroid.z << std::endl;
    payload_set.mass = 1.0;
    //unit: mm
    payload_set.centroid.x = 0; payload_set.centroid.y = 0; payload_set.centroid.z = 10;
    // Set current payload data
    demo.set_payload(&payload_set);
    // Get current payload data
    demo.get_payload(&payloadret);
    std::cout << " payload mass is :" << payloadret.mass << " kg" << std::endl;
    std::cout << " payload center of mass is \nx: " << payloadret.centroid.x << "y: " << p
ayloadret.centroid.y << "z: " << payloadret.centroid.z << std::endl;
    return 0;
}
```

4.3.22 Set tioV3 voltage parameters

```
/**
* @brief Set tioV3 voltage parameters
* @param vout_enable Voltage enable, 0:off, 1 on
* @param vout_vol Voltage 0:24v 1:12v
* @return ERR_SUCC Error or Success
*/
errno_t set_tio_vout_param(int vout_enable, int vout_vol);
```

4.3.23 Get tioV3 voltage parameters

```
/**
* @brief Get tioV3 voltage parameters
* @param vout_enable Voltage enable, 0:off, 1 on
* @param vout_vol Voltage 0:24v 1:12v
* @return ERR_SUCC Error or Success
*/
errno_t get_tio_vout_param(int* vout_enable, int* vout_vol);
```

4.3.24 Get robot status

```
/**
* @brief Get robot status
```

```
* @param state Interrogate result of robot status
* @return ERR_SUCC Error or Success
*/
errno_t  get_robot_state(RobotState* state);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Get robot status (emergency stop, power up, servo enable)
int example_get_robstate()
{
    JAKAZuRobot demo;
    //Declare the robot state structure
    RobotState state;
    demo.login_in("192.168.2.152");
    demo.power_on();
    demo.enable_robot();
    //Get the robot status
demo.get_robot_state(&state);
    std::cout << "is e_stoped : " << state.estoped << std::endl;
    std::cout << "is powered : " << state.poweredOn << std::endl;
    std::cout << "is servoEnabled : " << state.servoEnabled << std::endl;
    return 0;
}
```

4.3.25 TIO Add or Modify Semaphore

```
/**
* @brief Add or modify a semaphore
* @param sign_info Semaphore parameters
* @return ERR_SUCC Success, other values indicate failure
*/
errno_t add_tio_rs_signal(SignInfo sign_info);
```

Sample Code:

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

SignInfo sign_info {0};
memcpy(sign_info.sig_name, "sign_tmp", sizeof("sign_tmp"));
```

```
sign_info.chn_id = 0,
sign_info.sig_type = 0,
sign_info.value = 10,
sign_info.frequency = 5;
sign_info.sig_addr = 0x1;

robot.add_tio_rs_signal(sign_info);

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

4.3.26 TIO Delete Semaphore

```
/**
* @brief Delete a semaphore
* @param sig_name Semaphore name
* @return ERR_SUCC Success, other values indicate failure
*/
errno_t del_tio_rs_signal(const char* sig_name);
```

Sample Code:

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

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

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

4.3.27 TIO RS485 Send Command

```
/**
* @brief Send command via RS485
* @param chn_id Channel number
* @param data Data field
```

```
* @return ERR_SUCC Success, other values indicate failure
*/
errno_t send_tio_rs_command(int chn_id, uint8_t* data, int buffsize);
```

Sample Code:

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

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

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

4.3.28 TIO Get Semaphore Information

```
/**
* @brief Get semaphore information
* @param sign_info Pointer to an array of semaphore information
* @return ERR_SUCC Success, other values indicate failure
*/
errno_t get_rs485_signal_info(SignInfo* sign_info);
```

Sample Code:

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

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

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

4.3.29 TIO Set TIO Mode

```
/**
* @brief Set TIO mode
* @param pin_type TIO type (0 for DI Pins, 1 for DO Pins, 2 for AI Pins)
* @param pin_mode TIO mode
   DI Pins: 0:0x00 DI2 is NPN, DI1 is NPN; 1:0x01 DI2 is NPN, DI1 is PNP;
            2:0x10 DI2  is PNP, DI1 is NPN; 3:0x11 DI2 is PNP, DI1 is PNP
DO Pins: The high four bits in the low 8 bits are used to configure DO2, and the low four bits are used to configure DO1. 0x0: DO is NPN output, 0x1: DO is PNP output, 0x2: DO is push-pull output, 0xF: RS485H
AI Pins: 0: Enable analog input, RS485L disabled; 1: RS485L interface enabled, analog input disabled
* @return ERR_SUCC Success, other values indicate failure
*/
errno_t set_tio_pin_mode(int pin_type, int pin_mode);
```

4.3.30 TIO Get TIO Mode

```
/**
* @brief Get TIO mode
* @param pin_type TIO type (0 for DI Pins, 1 for DO Pins, 2 for AI Pins)
* @param pin_mode TIO mode
   DI Pins: 0:0x00 DI2 is NPN, DI1 is NPN; 1:0x01 DI2 is NPN, DI1 is PNP;
            2:0x10 DI2 is PNP, DI1 is NPN; 3:0x11 DI2 is PNP, DI1 is PNP
DO Pins: The high four bits in the low 8 bits are used to configure DO2, and the low four bits are used to configure DO1. 0x0: DO is NPN output, 0x1: DO is PNP output, 0x2: DO is push-pull output, 0xF: RS485H
AI Pins: 0: Analog input enabled, RS485L disabled; 1: RS485L interface enabled, analog input disabled
* @return ERR_SUCC Success, other values indicate failure
*/
errno_t get_tio_pin_mode(int pin_type, int* pin_mode);
```

4.3.31 TIO RS485 Communication Parameter Configuration

```
/**
* @brief Configure RS485 communication parameters
* @param mod_rtu_com When the channel mode is set to Modbus RTU, specify the Modbus slave ID additionally
* @return ERR_SUCC Success, other values indicate failure
*/
```

```
6. errno_t set_rs485_chn_comm(ModRtuComm mod_rtu_com);
```

4.3.32 TIO RS485 Communication Parameter Query

```
1. /**
2. * @brief Query RS485 communication parameters
3. * @param mod_rtu_com When querying, chn_id serves as an input parameter
4. * @return ERR_SUCC Success, other values indicate failure
5. */
6. errno_t get_rs485_chn_comm(ModRtuComm* mod_rtu_com);
```

4.3.33 TIO RS485 Communication Mode Configuration

```
1. /**
2. * @brief Configure RS485 communication mode
3. * @param chn_id 0: RS485H, channel 1; 1: RS485L, channel 2
4. * @param chn_mode 0: Modbus RTU, 1: Raw RS485, 2: Torque Sensor
5. * @return ERR_SUCC Success, other values indicate failure
6. */
7. errno_t set_rs485_chn_mode(int chn_id, int chn_mode);
```

4.3.34 TIO RS485 Communication Mode Query

```
1. /**
2. * @brief Query RS485 communication mode
3. * @param chn_id Input parameter: 0 for RS485H, channel 1; 1 for RS485L, channel 2
4. * @param chn_mode Output parameter: 0 for Modbus RTU, 1 for Raw RS485, 2 for Torque Sensor
5. * @return ERR_SUCC Success, other values indicate failure
6. */
7. errno_t get_rs485_chn_mode(int chn_id, int* chn_mode);
```

4.3.35 Get Robot Installation Angle

```
1. /**
2. * @brief Get the installation angle
3. * @param quat Quaternion representing the installation angle
4. * @param appang Installation angles in Roll-Pitch-Yaw (RPY) format
5. * @return ERR_SUCC Success, other values indicate failure
6. */
7. errno_t get_installation_angle(Quaternion* quat, Rpy* appang);
```

Sample Code:

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

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

 ret = robot.set_installation_angle(180, 0);
 if (ret != ERR_SUCC)
 {
 std::cerr << "set installation angle failed.\n";
 return -1;
 }

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

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

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

 return 0;
 }
```

4.3.36 Set Robot Installation Angle

```
/**
* @brief Set the installation angle
* @param angleX Rotation angle around the X-axis
* @param angleZ Rotation angle around the Z-axis
* @return ERR_SUCC Success, other values indicate failure
*/
errno_t set_installation_angle(double angleX, double angleZ);
```

Sample Code:

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

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

 ret = robot.set_installation_angle(180, 0);
 if (ret != ERR_SUCC)
 {
 std::cerr << "set installation angle failed.\n";
 return -1;
 }

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

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

```

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

return 0;
}
```

4.4 Set and Interrogate Robot Safety Status

4.4.1 Interrogate whether on limit

```
/**
* @brief Interrogate whether on limit
* @param on_limit Interrogate results
* @return  ERR_SUCC Error or Success
*/
errno_t  is_on_limit(BOOL* on_limit);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Interrogate whether on limit
int example_is_on_limit()
{
    JAKAZuRobot demo;
    BOOL on_limit;
    demo.login_in("192.168.2.152");
    demo.power_on();
    demo.enable_robot();
    while (1)
    {
        //Interrogate whether on limit
        demo.is_on_limit(&on_limit);
        std::cout << " on_limit is :" << on_limit << std::endl;
        Sleep(200);
    }
    return 0;
}
```

4.4.2 Interrogate whether in Collision Protection mode

```
/**
* @brief Interrogate whether in Collision Protection mode
* @param in_collision Interrogate results
* @return  ERR_SUCC Error or Success
*/
errno_t  is_in_collision(BOOL* in_collision);
```

4.4.3 Collision recover

```
/**
* @brief Collision recover
* @return  ERR_SUCC Error or Success
*/
errno_t  collision_recover();
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Collision protection status inquiry, recovery
int example_collision_recover()
{
    JAKAZuRobot demo;
    BOOL in_collision;
    demo.login_in("192.168.2.152");
    demo.power_on();
    demo.enable_robot();
    //Interrogate whether in collision protection mode
    demo.is_in_collision(&in_collision);
    std::cout << " in_collision is :" << in_collision << std::endl;
    if (in_collision)
        //Resume from collision protection if in collision protection mode
        {demo.collision_recover();}
    else
        {std::cout << "robot is not collision" << std::endl;}
    return 0;
}
```

4.4.4 Set collision level

```
/**
* @brief Set collision level
*@Param level Collision level, the value range is [0,5], 0: close collision, 1: collision threshold 25N, 2: collision threshold 50N, 3: collision threshold 75N, 4: collision threshold 100N, 5: collision threshold 125N
* @return  ERR_SUCC Error or Success
*/
errno_t set_collision_level(const int level);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//View and set collision level
int example_collision_level()
{
    //Instance API object demo
    JAKAZuRobot demo;
    int level;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.152");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Interrogate current collision level
    demo.get_collision_level(&level);
    std::cout << " collision level is :" << level << std::endl;
    //Set collision level from 0 to 5. 0 is off collision, 1 is collision threshold 25N, 2 is collision threshold 50N, 3 iscollision threshold 75N, 4 is collision threshold 100N, 5 is collision threshold 125N.
    demo.set_collision_level(2);
    //Interrogate current collision level
    demo.get_collision_level(&level);
    std::cout << " collision level is :" << level << std::endl;
    return 0;
}
```

4.4.5 Get collision level

```
/**
```

```
2. * @brief Get the robot collision level
3. * @return  ERR_SUCC Error or Success
4. */
5. errno_t get_collision_level(int* level);
```

4.4.6 Robot terminates automatically due to abnormal network

```
1.         /**
2.         * @brief  Set the control handle because of abnormal network, the robot controller terminates the current motion after a period of time when SDK loses connection with the robot controller
3.         * @param millisecond 3.Time parameter, unit: ms
4.         * @param mnt Robot motion type when the network is abnormal
5.         * @return ERR_SUCC Error or Success*/
6.         errno_t set_network_exception_handle(float millisecond, ProcessType mnt);
```

Sample Code:

```
1. #include <iostream>
2. #include "JAKAZuRobot.h"
3. #include <windows.h>
4. #define PI 3.1415926
5. //Set the automatic termination motion type when the network is abnormal
6. int example_set_network_exception_handle()
7. {
8.     float milisec = 100;
9.     int ret;
10.    JAKAZuRobot demo;
11.    demo.login_in("192.168.2.194");
12.    demo.power_on();
13.    demo.enable_robot();
14.    //Set condition of compliance torque
15.    ret = demo.set_network_exception_handle(milisec, MOT_KEEP);
16.    std::cout << ret << std::endl;
17.    return 0;
18. }
```

4.4.7 Set callback function

```
1. /**
2. * @brief Set callback function in case of a robot error
3. * @param func Function int to user-defined function
4. * @param error_code Robot error code
5. */
```

```
errno_t set_error_handler(CallBackFuncType func);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Register robot error function  Error handling, class interrupt
void user_error_handle(int error_code)
{
    std::cout << error_code << std::endl;
}
int example_set_err_handle()
{
    JAKAZuRobot demo;
    demo.login_in("192.168.2.229");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Set callback function
    demo.set_error_handler(user_error_handle);
    return 0;
}
```

4.4.8 Get the last error code

```
/**
* @brief Get the last error code in the robot running process, when clear_error is called, the last error code will be cleared
* @return  ERR_SUCC Error or Success
*/
errno_t  get_last_error(ErrorCode* code);
```

4.4.9 Set error code file path

```
/**
* @brief Set the error code file path. If you need to use the get_last_error interface, set the error code file path. If no need to use the get_last_error interface, do not set the interface.
* @return  ERR_SUCC Error or Success
*/
errno_t set_errorcode_file_path(char* path);
```

Sample Code:

```
#include <iostream>
```

```
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Error Code Viewing
int example_get_last_errcode()
{
    int ret;
// Initialize error code file storage path
    char path[100] = "E:\\JAKA_ERROR_CODE.csv";
    JAKAZuRobot demo;
    ErrorCode Eret;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    ret = demo.program_load("not_exist999875");//Intentionally load a non-existent program to raise an error
    std::cout << ret << std::endl;
    demo.get_last_error(&Eret);//Interrogate the last error message
    std::cout << " error code is :" << Eret.code << " message: "<< Eret.message<< std::endl;
    demo.set_errorcode_file_path(path);//Set error code description file
    demo.get_last_error(&Eret);//Interrogate the last error message
    std::cout << " error code is :" << Eret.code << " message: " << Eret.message << std::endl;
    return 0;
}
```

4.5 Use the APP Script Program

4.5.1 Run the loaded program

```
/**
* @brief Run the loaded program
* @return  ERR_SUCC Error or Success
*/
errno_t  program_run();
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Script loading, run control, process view
int example_program()
```

```
{
    char name[128];
    int cur_line;
    JAKAZuRobot demo;
    ProgramState pstatus;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    //Load the example script pre-edited by the app
    demo.program_load("example");
    //Get the loaded program name
    demo.get_loaded_program(name);
    std::cout <<"Pro_name is :"<< name << std::endl;
    //Run the loaded program
    demo.program_run();
    Sleep(1000);//Let the program run for 1s first
    //Pause the running program
    demo.program_pause();
    //Get the line number of the currently executing program
    demo.get_current_line(&cur_line);
    std::cout << "cur_line is :" << cur_line << std::endl;
    //Get current program status
    demo.get_program_state(&pstatus);
    std::cout <<  "pro_status is : " << pstatus << std::endl;
    //Continue running the current program
    demo.program_resume();
    Sleep(10000);//Requires window.h delay of 10s
    //Terminate the current program
    demo.program_abort();
    return 0;
}
```

4.5.2 Pause the running program

```
/**
* @brief Pause the running program
* @return  ERR_SUCC Error or Success
*/
errno_t  program_pause();
```

4.5.3 Resume program

```
/**
```

```
* @brief Resume program
* @return  ERR_SUCC Error or Success
*/
errno_t  program_resume();
```

4.5.4 Abort program

```
/**
* @brief Abort program
* @return  ERR_SUCC Error or Success
*/
errno_t  program_abort();
```

4.5.5 Load the specified program

```
/**
* @brief Load the specified program,The name of the program can be the name in the app  (load the track reproduction data, the loading of the track data needs to add track/ before the folder name)
* @param file Program file path
* @return  ERR_SUCC Error or Success
*/
errno_t  program_load (const char* file);
```

4.5.6 Get the loaded program

```
/**
* @brief Get the name of the loaded operating program
* @param file Program file path
* @return  ERR_SUCC Error or Success
*/
errno_t  get_loaded_program (char* file);
```

4.5.7 Get current line

```
/**
* @brief Get current line
* @param curr_line Interrogate result of current line
* @return  ERR_SUCC Error or Success
*/
errno_t  get_current_line (int* curr_line);
```

4.5.8 Get program status

```
/**
* @brief Get the program status
* @param status Interrogate result of program status
* @return  ERR_SUCC Error or Success
*/
errno_t  get_program_status (ProgramStatus* status);
```

4.5.9 Set rapid rate

```
/**
* @brief Set robot rapid rate
* @param rapid_rate The program rapid rate, [0,1]
* @return  ERR_SUCC Error or Success
*/
errno_t  set_rapidrate (double rapid_rate);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//View and adjust robot speed
int example_rapidrate()
{
    double rapid_rate;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.152");
    demo.power_on();
    demo.enable_robot();
    //Get robot motion rate
    demo.get_rapidrate(&rapid_rate);
    std::cout << "rapid_rate is : " << rapid_rate << std::endl;
    //Set robot motion rate
    demo.set_rapidrate(0.4);
    Sleep(100);
    demo.get_rapidrate(&rapid_rate);
    std::cout << "rapid_rate is : " << rapid_rate << std::endl;
    return 0;
}
```

4.5.10 Get rapid rate

```
/**
* @brief Get robot rapid rate
* @param rapid_rate Current control system rate
* @return  ERR_SUCC Error or Success
*/
errno_t  get_rapidrate (double* rapid_rate);
```

4.6 Trajectory Reproduction

4.6.1 Set trajectory track configuration parameters

```
/**
* @brief Set trajectory track configuration parameters
* @param para Track configuration parameters
* @return  ERR_SUCC Error or Success
*/
errno_t  set_traj_config(const TrajTrackPara* para);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//View and set trajectory reproduction parameter
int example_traj_config()
{
    JAKAZuRobot demo;
    TrajTrackPara trajpar_read;
    TrajTrackPara trajpar_set;
    demo.login_in("192.168.2.194");
    //Interrogate current trajectory reproduction parameter
    demo.get_traj_config(&trajpar_read);
    std::cout << " trajTrackPara is :\n xyz interval:" << trajpar_read.xyz_interval << " rpy interval is :" << trajpar_read.rpy_interval << std::endl;
    std::cout << " vel: " << trajpar_read.vel << "  acc: " << trajpar_read.acc << std::endl;
    //Set current trajectory reproduction parameter
    trajpar_set.xyz_interval = 0.01; trajpar_set.rpy_interval = 0.01; trajpar_set.vel = 10; trajpar_set.acc = 2;
    demo.set_traj_config(&trajpar_set);
```

```
    //Interrogate current trajectory reproduction parameter
    demo.get_traj_config(&trajpar_read);
    std::cout << " trajTrackPara is :\n xyz interval:" << trajpar_read.xyz_interval << "
  rpy interval is :" << trajpar_read.rpy_interval << std::endl;
    std::cout << " vel: " << trajpar_read.vel << "  acc: " << trajpar_read.acc << std::end
  l;
    return 0;
}
```

4.6.2 Get trajectory track configuration parameters

```
/**
* @brief Get trajectory track configuration parameters
* @param para Track configuration parameters
* @return  ERR_SUCC Error or Success
*/
errno_t  get_traj_config(TrajTrackPara* para);
```

4.6.3 Set trajectory sample mode

```
/**
* @brief Set trajectory sample mode
* @param mode Select TRUE to start data collection, when selecting FALSE, data collection is
  closed
* @param filename The name of the data file. When filename is a null int, the storage file is
  named after the current date
* @return  ERR_SUCC Error or Success
*/
errno_t  set_traj_sample_mode(const BOOL mode,char* filename);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Track acquisition switch and status interrogation
int example_traj_sample()
{
    BOOL samp_stu;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    char name[20] = "testxx";
```

```
    //Turn on the track recurrence data collection switch
    demo.set_traj_sample_mode(TRUE, name);
    //Get trajectory sample status
    demo.get_traj_sample_status(&samp_stu);
    Sleep(10000);
    demo.set_traj_sample_mode(FALSE, name);
    return 0;
}
```

4.6.4 Get trajectory sample status

```
/**
* @brief Get trajectory sample status
* @param mode  TRUE means the data is being collected, FALSE means the data collection is over, and it is not allowed to turn on the Data Collection switch again during data collection
* @return  ERR_SUCC Error or Success
*/
errno_t  get_traj_sample_status(BOOL* sample_status);
```

4.6.5 Get exist trajectory file name

```
/**
* @brief Get exist trajectory file name
* @param file name The name of trajectory file
* @return  ERR_SUCC Error or Success
*/
errno_t  get_exist_traj_file_name(MultStrStorType* filename);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Get exist trajectory file name in controller
int example_get_traj_existed_filename()
{
    JAKAZuRobot demo;
    MultStrStorType traj_file;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Interrogate current trajectory file name.
    demo.get_exist_traj_file_name(&traj_file);
    std::cout << "file nums :" << traj_file.len << std::endl;
    for(int i=0; i<traj_file.len;i++)
```

```
        std::cout << traj_file.name[i] << std::endl;
    return 0;
}
```

4.6.6 Rename exist trajectory file name

```
/**
* @brief Rename exist trajectory file name
* @param src Original file name
* @param dest The target file name, the length of the file name cannot exceed 100 characters, the file name cannot be empty, the target file name does not support Chinese
* @return  ERR_SUCC Error or Success
*/
errno_t  rename_traj_file_name(const char* src,const char* dest);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Rename exist trajectory file name
int example_rename_traj_file_name()
{
    JAKAZuRobot demo;
    MultStrStorType traj_file;
    char name_new[20] = "555";
    demo.login_in("192.168.2.194");
    //Interrogate current trajectory file name.
    demo.get_exist_traj_file_name(&traj_file);
    std::cout << "file nums :" << traj_file.len << std::endl;
    for (int i = 0; i < traj_file.len; i++)
        std::cout << traj_file.name[i] << std::endl;
    //Rename exist trajectory file name
    demo.rename_traj_file_name(traj_file.name[0], name_new);
    //Interrogate current trajectory file name.
    demo.get_exist_traj_file_name(&traj_file);
    std::cout << "file nums :" << traj_file.len << std::endl;
    for (int i = 0; i < traj_file.len; i++)
        std::cout << traj_file.name[i] << std::endl;
    return 0;
}
```

4.6.7 Remove the trajectory file in the controller

```
/**
* @brief Remove the trajectory file in the controller
* @param file name The file name of the file to be deleted is the name of data file
* @return  ERR_SUCC Error or Success
*/
errno_t  remove_traj_file(const char* filename);
```

4.6.8 Generate the trajectory execution script

```
/**
* @brief Generate the trajectory execution script
* @param filename The file name of the data file is the name of the data file without suffix
* @return  ERR_SUCC Error or Success
*/
errno_t  generate_traj_exe_file(const char* filename);
```

4.7 Robot servo move

4.7.1 Robot servo move control mode

```
/**
* @brief Robot servo move control mode enable
* @param enable  TRUE means to enter the servo move control mode, FALSE means to quit the mode
* @return ERR_SUCC  Error or Success
*/
  PS: In the versions of v19 and before this is a non-blocked interface, and after version V20 this is changed to be a block-interface.
errno_t  servo_move_enable(BOOL enable);
```

4.7.2 Robot joint servo move

```
/**
* @brief Joint move control mode
* @param joint_pos Joint move position
* @param move_mode Specify move mode: incremental move, absolute move
* @return ERR_SUCC Error or Success
*/
errno_t  servo_j(const JointValue* joint_pos, MoveMode move_mode);
```

Sample Code:

```
1. //Robot joint servo move
2. //You need to call servo_move_enable(TRUE) to enable servo mode before use this interface
3. //The sending cycle of the controller is 8ms, so the recommended cycle of user is also 8ms. The network environment can be reduced in the case of poor conditions
4. //Upper limit of joint speed is 180deg/s
5. //There is a big difference between this instruction and joint_move. The interpolation of joint_move is performed by the controller, and servo_j needs to do the trajectory planning in advance.
6. #include <iostream>
7. #include "JAKAZuRobot.h"
8. #include <windows.h>
9. #define PI 3.1415926
10. int main()
11. {
12.     //Instance API object demo
13.     JAKAZuRobot demo;
14.     //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
15.     demo.login_in("192.168.2.152");
16.     //Power on the robot
17.     demo.power_on();
18.     //Enable the robot
19.     demo.enable_robot();
20.     //TRUE means entering servo mode
21.     demo.servo_move_enable(TRUE);
22.     //Define and initialize JointValue variables
23.     JointValue joint_pos = {-0.001, 0* PI / 180, 0* PI / 180, 0* PI / 180, 0* PI / 180, -0.001};
24.     for (int i = 0; i < 100; i++)
25.     {
26.         //Joint servo move, which INCR means incremental move
27.         demo.servo_j(&joint_pos, INCR);
28.         Sleep(2);
29.     }
30.     //FALSE means exiting servo mode
31.     demo.servo_move_enable(FALSE);
32.     return 0;
33. }
```

4.7.3 Robot joint servo move extension

```
1. /**
2. * @brief  The robot joint move control mode increases the cycle adjustability. Cycle can be adjusted to multiples of 8ms
```

```
 * @param joint_pos Joint move target position
 * @move_mode Designated move mode: incremental move, absolute move
 * @step_num  Multiplying period, servo_j move period is step_num*8ms, where step_num>=1
 * @return ERR_SUCC Error or Success
 */
errno_t  servo_j(const JointValue* joint_pos, MoveMode move_mode, unsigned int step_num);
```

4.7.4 Robot Cartesian servo move

```
/**
 * @brief Control mode of robot cartesian space position
 * @param cartesian_pose End position of robot cartesian space motion
 * @param move_mode Specify move mode: ABS stands for absolute move, INCR stands for relative move
 * @return  ERR_SUCC Error or Success
 */
errno_t  servo_p(const CartesianPose* cartesian_pose, MoveMode move_mode);
```

Sample Code:

```
//Robot Cartesian servo move
//You need to call servo_move_enable(TRUE) to enable servo mode before use this interface
//The sending cycle of the controller is 8ms, so the recommended cycle of user is also 8ms. The network environment can be reduced in the case of poor conditions
//Upper limit of joint speed is 3.141592 rad/s. There is no relatively intuitive restriction on Cartesian space, but this joint speed restriction should be satisfied.
//There is a big difference between this instruction and linear_move. The interpolation of linear_move is performed by the controller, and servo_p needs to do the trajectory planning in advance.
#include <iostream>
#include "JAKAZuRobot.h"
#define PI 3.1415926
int main()//Robot Cartesian servo move
{
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.152");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //TRUEmeans enter servo mode
    demo.servo_move_enable(TRUE);
    //Define and initialize CartesianPose variables
    CartesianPose cart;
```

```
    cart.tran.x = 0; cart.tran.y = 1; cart.tran.z = 0;
    cart.rpy.rx = 0; cart.rpy.ry = 0; cart.rpy.rz = 0;
    for (int i = 0; i < 100; i++)
    {
        //Cartesian servo mode, which INCR stands for incremental move
        demo.servo_p(&cart, INCR);
        Sleep(2);
    }
    //FALSE means exiting servo mode
    demo.servo_move_enable(FALSE);
    return 0;
}
```

4.7.5 Robot cartesian servo move extension

```
/**
* @brief Control mode of robot cartesian position
* @param cartesian_pose End position of robot cartesian space motion
* @move_mode Specify move mode: incremental move or absolute move
* @step_num  Multiplying period, servo_p move period is step_num*8ms, where step_num>=1
* @return  ERR_SUCC Error or Success
*/
errno_t  servo_p(const CartesianPose* cartesian_pose, MoveMode move_mode, unsigned int step_num);
```

4.7.6 None filters in SERVO mode

```
/**
* @brief Do not use filters in the SERVO mode, this command cannot be set in the SERVOJ mode, and can be set after quitting the SERVOJ mode
* @return  ERR_SUCC Error or Success
*/
errno_t  servo_move_use_none_filter();
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
//None filters in SERVO mode
int example_servo_use_none_filter()
{
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
```

```
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    ret = demo.servo_move_use_none_filter();
    std::cout << ret << std::endl;
    return 0;
}
```

4.7.7 Use joint first-order low pass filter in SERVO mode

```
/**
* @brief Use joint First-order low-pass filter in SERVO mode, this command cannot be send in SERVOJ mode, and can be set after quitting SERVOJ
* @param cutoffFreq First-order low-pass filter cut-off frequency
* @return  ERR_SUCC Error or Success
*/
errno_t  servo_move_use_joint_LPF(double cutoffFreq);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Use joint first-order low pass filter in SERVO mode
int example_servo_use_joint_LPF()
{
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //First-order low-pass filtering in servo mode in joint, cutoff frequency is 0.5Hz
    ret = demo.servo_move_use_joint_LPF(0.5);
    std::cout << ret << std::endl;
    return 0;
}
```

4.7.8 Use joint nonlinear filter in SERVO mode

```
/**
* @brief Use joint nonlinear filter in SERVO mode, this command cannot be set in SERVOJ mode but can be set after quitting SERVOJ
* @param max_vr The upper limit of Cartesian space orientation change speed (absolute value) °/s
* @param max_ar The upper limit of accelerated speed of Cartesian space orientation change speed (absolute value)°/s^2
* @param max_jr The upper limit value of jerk (absolute value) of Cartesian space orientation change speed °/s^3
* @return  ERR_SUCC Error or Success
*/
errno_t  servo_move_use_joint_NLF(double max_vr, double max_ar, double max_jr);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Joint nonlinear filter in SERVO mode
int example_servo_use_joint_NLF()
{
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Joint nonlinear filter in SERVO mode
    ret = demo.servo_move_use_joint_NLF(2,2,4);
    std::cout << ret << std::endl;
    return 0;
}
```

4.7.9 Use Cartesian nonlinear filter in SERVO mode

```
/**
* @brief Cartesian space nonlinear filter under the mode, this command cannot be set in SERVOJ mode, but it can be set after quitting SERVOJ
```

```
3. * @param max_vp The upper limit (absolute value) of the move command speed in Cartesian space. Unit: mm/s
4. * @param max_ap The upper limit (absolute value) of the move command accelerated speed in Cartesian space. Unit: mm/s^2
5. * @param max_jp The unit of upper limit (absolute value) of the move command jerk in Cartesian space. mm/s^3
6. * @param max_vr The upper limit of Cartesian space orientation change speed (absolute value) °/s
7. * @param max_ar The upper limit of accelerated speed of Cartesian space orientation change speed (absolute value)°/s^2
8. * @param max_jr The upper limit value of jerk (absolute value) of Cartesian space orientation change speed °/s^3
9. * @return  ERR_SUCC Error or Success
10. */
11. errno_t  servo_move_use_carte_NLF(double max_vp, double max_ap, double max_jp, double max_vr, double max_ar, double max_jr);
```

Sample Code:

```
1. #include <iostream>
2. #include "JAKAZuRobot.h"
3. #include <windows.h>
4. #define PI 3.1415926
5. //Cartesian nonlinear filter in SERVO mode
6. int example_servo_use_carte_NLF()
7. {
8.     int ret;
9.     //Instance API object demo
10.    JAKAZuRobot demo;
11.    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
12.    demo.login_in("192.168.2.194");
13.    //Power on the robot
14.    demo.power_on();
15.    //Enable the robot
16.    demo.enable_robot();
17.    //Cartesian nonlinear filter in SERVO mode
18.    ret = demo.servo_move_use_carte_NLF(2, 2, 4, 2, 2, 4);
19.    std::cout << ret << std::endl;
20.    return 0;
21. }
```

4.7.10 Use joint multi-order mean filter in SERVO mode

```
1. /**
2. * @brief  Use joint space multi-order mean filter under the SERVO mode, this command cannot be set in SERVOJ mode but can be set after quitting SERVOJ
```

```
* @param max_buf The size of the mean filter buffer
* @param kp Acceleration filter factor
* @param kv Speed filter factor
* @param ka Position filter factor
* @return  ERR_SUCC Error or Success
*/
errno_t  servo_move_use_joint_MMF(int max_buf, double kp, double kv, double ka);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Joint multi-order mean filter in SERVO mode
int example_servo_use_joint_MMF()
{
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Joint multi-order mean filter in SERVO mode
    ret = demo.servo_move_use_joint_MMF(20, 0.2, 0.4, 0.2);
    std::cout << ret << std::endl;
    return 0;
}
```

4.7.11 Set speed foresight parameter under robot servo mode

```
/**
 *@brief Joint space multi-order mean filter under the SERVO mode, this command cannot be set in SERVO mode but can be set after exiting SERVO
*@param max_buf the buffer size of the mean filter
 *@param kp acceleration filter factor
*@param kv speed filter factor
*@param ka position filter factor
* @return ERR_SUCC Error or Success
*/
errno_t servo_speed_foresight(int max_buf, double kp);
 */
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Set speed foresight parameter under robot servo mode
int example_speed_foresight()
{
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Joint multi-order mean filter in SERVO mode
    ret = demo.servo_speed_foresight(200, 2);
    std::cout << ret << std::endl;
    return 0;
}
```

4.8 Robot Kinematics

4.8.1 Kine inverse

```
/**
* @brief Calculate the kine inverse of the specified pose under the current tool, current installation angle, and current user coordinate frame settings
* @param ref_pos Reference joint position for kine inverse
* @param cartesian_pose Cartesian space pose value
* @param joint_pos Joint space position calculation result when calculation is successful
* @return  ERR_SUCC Error or Success
*/
errno_t kine_inverse(const JointValue* ref_pos, const CartesianPose* cartesian_pose, JointValue* joint_pos);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Kine inverse of robot. Know tcp_pos,find joint_pos
int example_kine_inverse()
```

```
{
    int ret;
    JAKAZuRobot demo;
    //Initialize reference points
    JointValue ref_jpos = { 0.558, 0.872, 0.872 , 0.349, 0.191, 0.191 };
    //Initialize Cartesian space point coordinates
    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;
    //Initialize return value
    JointValue joint_pos = { 0,0,0,0,0,0 }; ;
    demo.login_in("192.168.2.194");
//Kine inverse
    ret = demo.kine_inverse(&ref_jpos, &tcp_pos, &joint_pos);
    std::cout << ret << std::endl;
    for (int i = 0; i < 6; i++)
    {
        std::cout << "joint [" << i + 1 << "] is :" << joint_pos.jVal[i] << std::endl;
    }
    return 0;
}
```

4.8.2 Kine forward

```
/**
* @brief Calculate the pose value of the specified joint position under the current tool, current installation angle and current user coordinate frame settings
* @param joint_pos Joint space position
* @param cartesian_pose Calculation results of Cartesian space pose
* @return  ERR_SUCC Error or Success
*/
errno_t  kine_forward(const JointValue* joint_pos, CartesianPose* cartesian_pose);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
// Kine forward of robot. Know tcp_pos,find joint_pos
int example_kine_forward()
{
    int ret;
    JAKAZuRobot demo;
//Initialize return value
    CartesianPose tcp_pos;
```

```
    demo.login_in("192.168.2.194");
//Initialize joint matrix
    JointValue joint_pos = { 0.558, 0.872, 0.872 , 0.349, 0.191, 0.191 };
//Kine forward
    ret = demo.kine_forward(&joint_pos, &tcp_pos);
    std::cout << ret << std::endl;
    std::cout << "tcp_pos is :\n x: " << tcp_pos.tran.x << "  y: " << tcp_pos.tran.y << "  z: " << tcp_pos.tran.z << std::endl;
    std::cout << "rx: " << tcp_pos.rpy.rx << "  ry: " << tcp_pos.rpy.ry << "  rz: " << tcp_pos.rpy.rz << std::endl;
    return 0;
}
```

4.8.3 Rpy to rot matrix

```
/**
* @brief Rpy to rot matrix
* @param rpy Rpy parameters to be converted
* @param rot_matrix Rot matrix after conversion
* @return  ERR_SUCC Error or Success
*/
errno_t  rpy_to_rot_matrix(const Rpy* rpy, RotMatrix* rot_matrix);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Rpy to rot matrix
int example_rpy_to_rot_matrix()
{
    int ret;
    JAKAZuRobot demo;
    //Initialize rot matrix
    Rpy rpy;
    rpy.rx = -1.81826; rpy.ry = -0.834253;  rpy.rz = -2.30243;
//Initialize return value
    RotMatrix rot_matrix;
    demo.login_in("192.168.2.194");
//Rpy to rot matrix
    ret = demo.rpy_to_rot_matrix(&rpy, &rot_matrix);
    std::cout << ret << "     eul2rotm" << std::endl;
    printf("%f %f %f\n", rot_matrix.x.x, rot_matrix.y.x, rot_matrix.z.x);
    printf("%f %f %f\n", rot_matrix.x.y, rot_matrix.y.y, rot_matrix.z.y);
    printf("%f %f %f\n", rot_matrix.x.z, rot_matrix.y.z, rot_matrix.z.z);
```

```
    return 0;
}
```

4.8.4 Rot matrix to rpy

```
/**
* @brief Rot matrix to rpy
* @param rot_matrix Rot matrix data to be converted
* @param rpy RPY values obtained
* @return  ERR_SUCC Error or Success
*/
errno_t  rot_matrix_to_rpy(const RotMatrix* rot_matrix, Rpy* rpy);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Rot matrix ---> rpy
int example_rot_matrix_to_rpy()
{
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    //Initialize rpy
    Rpy rpy;
    /Initialize rot matrix
    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;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    ret = demo.rot_matrix_to_rpy(&rot_matrix, &rpy);
    std::cout << ret << "     rotm2eul:" << std::endl;
    printf("%f %f %f \n", rpy.rx, rpy.ry, rpy.rz);
    return 0;
}
```

4.8.5 Quaternion to rot matrix

```
/**
* @brief Quaternion to to rot matrix
* @param quaternion Quaternion data to be converted
```

```
 * @param rot_matrix Rot matrix obtained
 * @return  ERR_SUCC Error or Success
 */
errno_t quaternion_to_rot_matrix(const Quaternion* quaternion, RotMatrix* rot_matrix);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Quaternion --> rot matrix
int example_quaternion_to_rot_matrix()
{
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    //Initialize quaternion
    Quaternion quat;
    quat.s = 0.0629;  quat.x = 0.522886; quat.y = -0.5592;  quat.z = 0.6453;
    //Initialize rot matrix
    RotMatrix rot_matrix;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    ret = demo.quaternion_to_rot_matrix(&quat, &rot_matrix);
    std::cout << ret << "     quatl2rotm:" << std::endl;
    printf("%f %f %f\n", rot_matrix.x.x, rot_matrix.y.x, rot_matrix.z.x);
    printf("%f %f %f\n", rot_matrix.x.y, rot_matrix.y.y, rot_matrix.z.y);
    printf("%f %f %f\n", rot_matrix.x.z, rot_matrix.y.z, rot_matrix.z.z);
    return 0;
}
```

4.8.6 Get the DH parameters of the currently connected robot

```
/**
* @brief Get the robot DH parameters
* @param dhParam DH parameters
* @return ERR_SUCC Error or Success
*/
errno_t get_dh_param(const JKHD *handle, DHParam *dhParam);
```

4.8.7 Rot matrix to quaternion

```
/**
* @brief Rot matrix to quaternion
```

```
3. * @param rot_matrix Rot matrix to be converted
4. * @param quaternion Converted quaternion result
5. * @return  ERR_SUCC Error or Success
6. */
7. errno_t  rot_matrix_to_quaternion(const RotMatrix* rot_matrix, Quaternion* quaternion);
```

Sample Code:

```
1. #include <iostream>
2. #include "JAKAZuRobot.h"
3. #include <windows.h>
4. #define PI 3.1415926
5. //Rot matrix ---> quaternion
6. int example_rot_matrix_to_quaternion()
7. {
8.     int ret;
9.     //Instance API object demo
10.    JAKAZuRobot demo;
11.    //Initialize quaternion
12.    Quaternion quat;
13.    //Initialize rot matrix
14.    RotMatrix rot_matrix;
15.    rot_matrix.x.x = -0.4488774, rot_matrix.y.x = -0.499824, rot_matrix.z.x = 0.740795;
16.    rot_matrix.x.y = -0.662098, rot_matrix.y.y = -0.370777, rot_matrix.z.y = -0.651268;
17.    rot_matrix.x.z = 0.600190, rot_matrix.y.z = -0.782751, rot_matrix.z.z = -0.164538;
18.    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
19.    demo.login_in("192.168.2.194");
20.    ret = demo.rot_matrix_to_quaternion(&rot_matrix, &quat);
21.    std::cout << ret << "    rotm2quat:" << std::endl;
22.    printf("%lf %lf %lf %lf \n", quat.s, quat.x, quat.y, quat.z);
23.    return 0;
24. }
```

4.9 Force Control Robot

Requires additional configuration of tool end force sensors

4.9.1 Set sensor brand

```
1. /**
2. * @brief Set sensor brand
3. * @param sensor_brand sensor brands, 1-6 corresponding to different sensor brands, consult
   engineers for details
4. * @return ERR_SUCC Error or Success
5. */
```

```
errno_t set_torsenosr_brand(int sensor_brand);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Set sensor brand
int example_set_torsensor_brand()
{
    int ret;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    //Set sensor brand
    ret = demo.set_torsenosr_brand(2);
    std::cout << ret << std::endl;
    return 0;
}
```

4.9.2 Get sensor brand

```
/**
* @brief Get sensor brand
* @param sensor_brand Sensor brands ,
* @return ERR_SUCC Error or Success
*/
errno_t get_torsenosr_brand(int* sensor_brand);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Get sensor brand
int example_get_torsensor_brand()
{
    int ret,cur_sensor;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    //Get sensor brand
    ret = demo.get_torsenosr_brand(&cur_sensor);
    std::cout << ret << std::endl;
```

```
    return 0;
}
```

4.9.3 Turn on/off force torque sensor

```
/**
* @brief Turn on/off force torque sensor, servo mode needs to be turned on first
* @param sensor_mode 0 means turning off the sensor, 1 means turning on the sensor
* @return ERR_SUCC Error or Success
*/
errno_t set_torque_sensor_mode(int sensor_mode);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Turn on/off force torque sensor
int example_set_torque_sensor_mode()
{
    int ret;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
demo.servo_move_enable(TRUE);
Sleep(200);
    //Set the status of torque sensor, 1 is on, 0 is off
    ret = demo.set_torque_sensor_mode(1);
    std::cout << ret << std::endl;
    return 0;
}
```

4.9.4 Set compliance control parameter

```
/**
* @brief Set compliance control parameter
* @param axis Optional value from 0 to 5 to configure certain axis,  corresponds to fx, fy, fz, mx, my, mz respectively
* @param opt  0 means not checked non-zero values mean checked
* @param ftUser 5.Damping force, The force of user use to make the robot moves in a certain direction at the maximum speed
* @param ftReboundFK Springback force, the force for the robot moves to the initial state
* @param ftConstant 7.Constant force, all set to 0 in manual operation
```

```
 * @param ftNnormalTrack Normal tracking, all set to 0 in manual operation
 * @return ERR_SUCC 9.Error or Success
*/
errno_t set_admit_ctrl_config(int axis, int opt, int ftUser, int ftConstant, int ftNnormal
    Track, int ftReboundFK);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Set compliance control parameters
int example_set_admit_ctrl_config()
{
    int ret;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    //Set compliance control parameters
    ret = demo.set_admit_ctrl_config(1,1,20,5,0,0);
    std::cout << ret << std::endl;
    return 0;
}
```

4.9.5 Set sensor end payload

```
/**
 * @brief Set sensor end payload
 * @param payload End payload
 * @return ERR_SUCC Error or Success
 */
errno_t set_torq_sensor_tool_payload(const PayLoad* payload);
```

4.9.6 Get end payload identification state

```
/**
 * @brief Get end payload identification state
 * @param identify_status 0 means identification completed, 1 means unfinished, 2 means
    failure
 * @return ERR_SUCC Error or Success
 */
errno_t get_torq_sensor_identify_staus(int* identify_status);
```

4.9.7 Identify tool end payload

```
/**
* @brief Start to identify tool end payload
* @param joint_pos The last position when the torque sensor is used for automatic payload
* @return ERR_SUCC Error or Success
*/
errno_t start_torq_sensor_payload_identify(const JointValue* joint_pos);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Identify tool end load and acquire load identification status, set and acquire sensor end load
int example_sensor_payload()
{
    JointValue joint_pos;
    PayLoad pl,pl_ret;
    int ret;
    JAKAZuRobot demo;
    demo.login_in("192.168.2.194");
    demo.power_on();
    demo.enable_robot();
    //Start identifying sensor payloads
    ret = demo.start_torq_sensor_payload_identify(&joint_pos);
    do
    {
        //Interrogate the status of sensor payloads
        demo.get_torq_sensor_identify_staus(&ret);
        std::cout << ret << std::endl;
    } while (1 == ret);
    //Get identifying results
    ret = demo.get_torq_sensor_payload_identify_result(&pl);
    std::cout << ret << std::endl;
    //Set end payloads of sensor
    ret = demo.set_torq_sensor_tool_payload(&pl);
    //Get the currently set sensor end load
    ret = demo.get_torq_sensor_tool_payload(&pl_ret);
    return 0;
}
```

4.9.8 Get end payload identification result

```
/**
* @brief Get end payload identification result
* @param payload End payload
* @return ERR_SUCC Error or Success
*/
errno_t get_torq_sensor_payload_identify_result(PayLoad* payload);
```

4.9.9 Get sensor end payload

```
/**
* @brief Get sensor end payload
* @param payload End payload
* @return ERR_SUCC Error or Success
*/
errno_t get_torq_sensor_tool_payload(PayLoad* payload);
```

4.9.10 set coordinate frame of admittance control

```
/**
*@brief set coordinate frame of admittance control
* @param ftFrame 0 means tools, 1 means world
* @return ERR_SUCC Error or Success
*/
errno_t set_ft_ctrl_frame(const int ftFrame);
```

4.9.11 get coordinate frame of admittance control

```
/**
* @brief get coordinate frame of admittance control
*@param ftFrame 0 means tools 1 means world
* @return ERR_SUCC Error or Success
*/
errno_t get_ft_ctrl_frame(int* ftFrame);
```

4.9.12 Enable force-control admittance control

```
/**
```

```
2. * @brief enable force-control admittance control, the compliance control parameters need to
   be set first, and turn on and initiate the force-control sensor
3. *@param enable_flag 0 means to turn off force-control drag enabling, 1 means to turn on
4. * @return ERR_SUCC Error or Success
5. */
6. errno_t enable_admittance_ctrl(const int enable_flag);
```

Sample Code:

```
1. #include <iostream>
2. #include "JAKAZuRobot.h"
3. #include <windows.h>
4. #define PI 3.1415926
5. //Enable force-control admittance control
6. int example_enable_admittance_ctrl()
7. {
8.     int ret;
9.     //Instance API object demo
10.    JAKAZuRobot demo;
11.    //login controller, you need to replace 192.168.2.105 with the IP of your own controller.
12.    demo.login_in("10.5.5.100");
13.    //Power on the robot
14.    demo.power_on();
15.    //Enable the robot
16.    demo.enable_robot();
17.    //Set sensor brand
18.    demo.set_torsenosr_brand(2);
19.    //Turn on the force sensor
20.    demo.set_torque_sensor_mode(1);
21.    //Initialize the force sensor
22.    demo.set_compliant_type(1, 1);
23.    printf("inint sensor comple\n");
24.    //Set compliance control parameters
25.    ret = demo.set_admit_ctrl_config(0, 0, 20, 5, 0, 0);
26.    ret = demo.set_admit_ctrl_config(1, 0, 20, 5, 0, 0);
27.    ret = demo.set_admit_ctrl_config(2, 2, 20, 5, 0, 0);
28.    ret = demo.set_admit_ctrl_config(3, 0, 20, 5, 0, 0);
29.    ret = demo.set_admit_ctrl_config(4, 0, 20, 5, 0, 0);
30.    ret = demo.set_admit_ctrl_config(5, 0, 20, 5, 0, 0);
31.    //Set force control drag enable, 1 on, 0 off
32.    ret = demo.enable_admittance_ctrl(1);
33.    printf("enable_admittance_ctrl open! \n");
34.    std::cout << ret << std::endl;
35.    printf("input any word to quit:\n");
36.    std::cin >> ret;
37.    ret = demo.enable_admittance_ctrl(0);
```

```
    ret = demo.set_admit_ctrl_config(2, 0, 20, 5, 0, 0);
    demo.set_torque_sensor_mode(0);
    printf("close\n");
    return 0;
}
```

4.9.13 Set force control type and sensor initial state

```
/**
* @brief Set force control type and sensor initial state
* @param sensor_compensation Whether to enable sensor compensation, 1 means to start and initialize, 0 means not to initialize
* @param compliance_type 0 means constant force compliance control, 1 means velocity compliance control,2 Means speed compliance control
* @return ERR_SUCC Error or Success
*/
errno_t set_compliant_type(int sensor_compensation, int compliance_type);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Set force control type and sensor initial state
int example_set_compliant_type()
{
    int ret,sensor_compensation,compliance_type;
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    demo.servo_move_enable(TRUE);
    //Set force control type and sensor initial state
    ret = demo.set_compliant_type(1,0);
    std::cout << ret << std::endl;
    ret = demo.get_compliant_type(&sensor_compensation, &compliance_type);
    std::cout << ret << std::endl;
    return 0;
}
```

4.9.14 Get force control type and sensor initial state

```
/**
 * @brief Get force control type and sensor initial state
 * @param sensor_compensation Whether to enable sensor compensation, 1 means to start and initialize, 0 means not to initialize
 * @param compliance_type 0 means constant force compliance control, 1 means velocity compliance control,2 Means speed compliance control
 * @return ERR_SUCC Error or Success
 */
errno_t get_compliant_type(int* sensor_compensation, int* compliance_type);
```

4.9.15 Get force control compliance parameter

```
/**
 * @brief Get force control compliance parameter
 * @param admit_ctrl_cfg The address storage of force control compliance parameter
 * @return ERR_SUCC Error or Success
 */
errno_t get_admit_ctrl_config(RobotAdmitCtrl *admit_ctrl_cfg);
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Get compliance force control parameters
int example_get_admit_ctrl_config()
{
    RobotAdmitCtrl adm_ctr_cfg;
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Get compliance force control parameters
    ret = demo.get_admit_ctrl_config(&adm_ctr_cfg);
    std::cout << ret << std::endl;
    return 0;
}
```

4.9.16 Set sensor communication parameter

```
/**
* @brief Set force control sensor communication parameter
* @param type3. Communication type, 0 means using tcp/ip protocol, 1 means using RS485 protocol
* @param ip_addr4.Force control sensor address
* @param port Force control sensor port No. When using tcp/ip protocol
* @return ERR_SUCC Error or Success
*/
errno_t set_torque_sensor_comm(const int type, const char* ip_addr, const int port);
```

Sample Code:

```
int example_torque_sensor_comm()
{
    char ip_set[30]="192.168.2.108";
    int ret=2;
    int type_set = 0,port_set = 4008;
    char ip_ret[30]="1";
    int type_ret = 0, port_ret = 0;
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.105 with the IP of your own controller.
    printf("logining!\n");
    demo.login_in("192.168.2.106");
    //Power on the robot
    printf("powering\n");
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Set sensor brand
    ret = demo.set_torsenosr_brand(4);
    //Get force control communication parameters
    ret = demo.get_torque_sensor_comm(&type_ret, ip_ret, &port_ret);
    std::cout << ret << std::endl;
    std::cout << ip_ret << std::endl;
    std::cout << port_ret << std::endl;
    std::cin >> type_ret;
    //Set force control communication parameters
    ret = demo.set_torque_sensor_comm(type_set, ip_set, port_set);
    std::cout << ret << std::endl;
    std::cout << ip_set << std::endl;
    std::cout << port_set << std::endl;
    std::cin >> type_set;
    return 0;
}
```

4.9.17 Get sensor communication parameter

```
/**
 * @brief get force control sensor communication parameter,
 * @param type Communication type, 0 means using tcp/ip protocol, 1 means using RS485 protocol
 *@param ip_addr currently set communication address of the force control sensor. Only communication interface address
 * @param port Force control sensor port No. When using tcp/ip protocol
 * @return ERR_SUCC Error or Success
 */
errno_t get_torque_sensor_comm(int* type, char* ip_addr,int* port);
```

4.9.18 Turn off force control

```
/**
 * @brief Turn off force contro
 * @return ERR_SUCC Error or Success
 */
errno_t disable_force_control();
```

Sample Code:

```
#include <iostream>
#include "JAKAZuRobot.h"
#include <windows.h>
#define PI 3.1415926
//Turn off force control
int example_disable_force_control()
{
    int ret;
    //Instance API object demo
    JAKAZuRobot demo;
    //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
    demo.login_in("192.168.2.194");
    //Power on the robot
    demo.power_on();
    //Enable the robot
    demo.enable_robot();
    //Turn off force control
    ret = demo.disable_force_control();
    std::cout << ret << std::endl;
    return 0;
}
```

4.9.19 Set velocity compliance control parameter

```
1.        /**
2.        * @brief Set velocity compliance control parameter
3.        * @param vel_cfg velocity compliance control parameter
4.        * @return ERR_SUCC Error or Success
5.        */
6.        errno_t set_vel_compliant_ctrl(const VelCom* vel_cfg);
```

4.9.20 Set compliance control torque condition

```
1.        /**
2.        * @brief Set compliance control torque condition
3.        * @param ft Compliance control torque condition, will stop if the torque exceeds this
  condition
4.        * @return ERR_SUCC Error or Success
5.        */
6.        errno_t set_compliance_condition(const FTxyz* ft);
```

Sample Code:

```
1.  #include <iostream>
2.  #include "JAKAZuRobot.h"
3.  #include <windows.h>
4.  #define PI 3.1415926
5.  //Set condition of compliance torque
6.  int example_set_compliance_condition()
7.  {
8.      FTxyz ft;
9.      ft.fx = 10; ft.fy = 10; ft.fz = 10;
10.     ft.tx = 10; ft.ty = 10; ft.tz = 10;
11.     int ret;
12.     //Instance API object demo
13.     JAKAZuRobot demo;
14.     //login controller, you need to replace 192.168.2.194 with the IP of your own controller.
15.     demo.login_in("192.168.2.194");
16.     //Power on the robot
17.     demo.power_on();
18.     //Enable the robot
19.     demo.enable_robot();
20.     //Set condition of compliance torque
21.     ret = demo.set_compliance_condition(&ft);
22.     std::cout << ret << std::endl;
23.     return 0;
24. }
```

4.9.21 Set low-pass filter parameters for force control

```
/**
* @brief Set the value of the low-pass filter for force control
* @param torque_sensor_filter Value of low-pass filter, Unit: Hz
*/
errno_t set_torque_sensor_filter(const float torque_sensor_filter);
```

4.9.22 Obtain low-pass filter parameters for force control

```
/**
* @brief Get the value of the low-pass filter for force control
* @param torque_sensor_filter Value of low-pass filter, unit: Hz
*/
errno_t get_torque_sensor_filter(float *torque_sensor_filter);
```

4.9.23 Set the sensor limit parameter configuration for force sensors

```
/**
* @brief Set the sensor limit parameter configuration for force sensors
* @param torque_sensor_soft_limit Sensor limit parameter for force sensors
*        Force limit fx, fy, fz Unit: N
*        Torque limit tx, ty, tz Unit: N*m
*/
errno_t set_torque_sensor_soft_limit(const FTxyz torque_sensor_soft_limit);
```

4.9.24 Get the sensor limit parameter configuration for force sensors

```
/**
* @brief Get the sensor limit parameter configuration for force sensors
* @param torque_sensor_soft_limit Sensor limit parameter for force sensors
*        Force limit fx, fy, fz Unit: N
*        Torque limit tx, ty, tz Unit: N*m
*/
errno_t get_torque_sensor_soft_limit(FTxyz *torque_sensor_soft_limit);
```

4.10 FTP Service

4.10.1 Initialize FTP client

```
1.        /**
2.        *@brief initialize FTP client, establish connection with control cabinet, capable of exporting program, track
3.        *@return ERR_SUCC Error or Success
4.        */
5.        errno_t init_ftp_client();
```

4.10.2 FTP upload

```
1.        /**
2.        *@brief upload local files with specified type and name to controller
3.        *@param remote upload to the absolute path of the controller internal file name. If it is a folder, the name should be ended with "\" or "/"
4.        *@param local the absolute path of the local file name. If it is a folder, the name should be ended with a "\" or "/"
5.        *@param opt 1 means single file 2 means folder
6.        *@return ERR_SUCC Error or Success
7.        */
8.        errno_t upload_file(char* local, char* remote, int opt);
```

4.10.3 FTP download

```
1.        /**
2.        *@brief download files with specified type and name from controller to local path
3.        *@param remote controller internal file name absolute path, if it is a folder, the name should be ended with "\" or "/"
4.        *@param local download to the absolute path of local file name. If it is a folder, the name should be ended with a "\"  "/"
5.        *@param opt 1 means single file 2 means folder
6.        *@return ERR_SUCC Error or Success
7.        */
8.        errno_t download_file(char* local, char* remote, int opt);
```

4.10.4 Interrogate FTP directory

```
1.        /**
```

```
2.        *@brief Interrogate FTP directory
3.        *@param remote the original file name of the controller internal file, Interrogate track "/track/", Interrogate script program "/program/"
4.        *@param opt 0 means file name and subdirectory name, 1 means file name, and 2 means subdirectory name
5.        *@param ret returned Interrogate result
6.        *@return ERR_SUCC Error or Success
7.        */
8.        errno_t get_ftp_dir(const char* remotedir, int type, char* ret);
```

4.10.5 Delete FTP

```
1.        /**
2.        *@brief delete files with specified type and name from controller
3.        *@param remote controller internal file name
4.        *@param opt 1 means single file 2 means folder
5.        *@return ERR_SUCC Error or Success
6.        */
7.        errno_t del_ftp_file(char* remote, int opt);
```

4.10.6 Rename FTP

```
1.        /**
2.        *@brief rename controller files with specified type and name
3.        *@param remote original file name of controller internal file
4.        *@param des target file name for the renamed file
5.        *@param opt 1 means single file 2 means folder
6.        *@return ERR_SUCC Error or Success
7.        */
8.        errno_t rename_ftp_file(char* remote, char* des, int opt);
```

4.10.7 Close FTP client

```
1.        /**
2.        *@brief disconnect the link with controller FTP
3.        *@return ERR_SUCC Error or Success
4.        */
errno_t close_ftp_client();
```

5. Feedback

For any inaccurate descriptions or errors in the document, we would like to invite the readers to correct and criticize. In case of any questions during your reading or any comments you want to make, please send an email to support@jaka.com, and our colleagues will reply.