CHAPTER

# TWO

# SDK MANUAL

## 2.1 C++

This manual is the secondary development interface document of C++.

**Important:** Robot parameter unit description: The robot position unit is millimeter (mm), and the attitude unit is degree (°).

**Important:**

1) In code examples that are not specifically stated, the robot has been powered on and enabled by default;
2) All code examples in the documentation default to no interference within the robot's workspace;
3) Please use the data of the on-site robot in the actual use test.

### 2.1.1 Data structure specification

#### 2.1.1.1 Interface call return value type

```
typedef  int errno_t;
```

#### 2.1.1.2 Joint position data type

```
/**
* @brief Joint position data type
*/
typedef  struct
{
    double jPos[6];   /* Six joint positions, unit: deg */
}JointPos;
```

#### 2.1.1.3 Cartesian spatial location data type

```c
/**
* @brief Cartesian spatial location data type
*/
typedef struct
{
    double x;    /* X-axis coordinate, unit: mm  */
    double y;    /* Y-axis coordinate, unit: mm  */
    double z;    /* Z-axis coordinate, unit: mm  */
} DescTran;
```

#### 2.1.1.4 Euler Angle attitude data type

```c
/**
* @brief Euler Angle attitude data type
*/
typedef struct
{
    double rx;   /* Rotation Angle about fixed axis X, unit: deg  */
    double ry;   /* Rotation Angle about fixed axis y, unit: deg  */
    double rz;   /* Rotation Angle about fixed axis Z, unit: deg  */
} Rpy;
```

#### 2.1.1.5 Cartesian space pose data type

```c
/**
*@brief Cartesian space pose type
*/
typedef struct
{
    DescTran tran;      /* Cartesian position  */
    Rpy rpy;            /* Cartesian space attitude  */
} DescPose;
```

#### 2.1.1.6 Extension axis position data type

```c
/**
* @brief Extension axis position data type
*/
typedef  struct
{
    double ePos[4];   /* Position of four expansion shafts, unit: mm */
}ExaxisPos;
```

#### 2.1.1.7 Torque sensor data type

```
/**
* @brief The force component and torque component of the force sensor
*/
typedef struct
{
    double fx;  /* Component of force along the x axis, unit: N  */
    double fy;  /* Component of force along the y axis, unit: N  */
    double fz;  /* Component of force along the z axis, unit: N  */
    double tx;  /* Component of torque about the X-axis, unit: Nm */
    double ty;  /* Component of torque about the Y-axis, unit: Nm */
    double tz;  /* Component of torque about the Z-axis, unit: Nm */
} ForceTorque;
```

#### 2.1.1.8 Spiral parameter data type

```
/**
* @brief  Spiral parameter data type
*/
typedef  struct
{
    int    circle_num;           /* Coil number  */
    float  circle_angle;         /* Spiral Angle  */
    float  rad_init;             /* Initial radius of spiral, unit: mm  */
    float  rad_add;              /* Radius increment  */
    float  rotaxis_add;          /* Increment in the direction of the axis of rotation ␣
↪*/
    unsigned int rot_direction;  /* Rotation direction, 0- clockwise, 1-␣
↪counterclockwise  */
}SpiralParam;
```

### 2.1.2 Basics

#### 2.1.2.1 Instantiate the robot

```
/**
* @brief  Robot interface class constructor
*/
ERARobot();
```

### 2.1.2.2 Establishes communication with the controller

```
/**
* @brief  Establish communication with the robot controller
* @param  [in] ip  Controller IP address. The default value is 192.168.58.2
* @return Error code
*/
errno_t  RPC(const char *ip);
```

### 2.1.2.3 Query the SDK version number

```
/**
* @brief  Query the SDK version number
* @param  [out] version  SDK version
* @return  Error code
*/
errno_t  GetSDKVersion(char *version);
```

### 2.1.2.4 Obtain Controller IP address

```
/**
* @brief  Obtain Controller IP address
* @param  [out] ip  Controller IP
* @return  Error code
*/
errno_t  GetControllerIP(char *ip);
```

### 2.1.2.5 Control the robot to enter or exit the drag teaching mode

```
/**
* @brief  Control the robot to enter or exit the drag teaching mode
* @param  [in] state 0-exit drag mode1-enter the drag mode
* @return  Error code
*/
errno_t  DragTeachSwitch(uint8_t state);
```

### 2.1.2.6 Queries whether the robot is in drag mode

```
/**
* @brief  Check whether the robot is in drag mode
* @param  [out] state 0-non-drag teaching mode1-drag the teaching mode
* @return  Error code
*/
errno_t  IsInDragTeach(uint8_t *state);
```

### 2.1.2.7 Control up enable and down enable

```
/**
* @brief  Enable or disable the function on or off the robot. By default, the function is enabled automatically after the robot is powered on
* @param  [in] state  0-down-enable1-upper enable
* @return  Error code
*/
errno_t  RobotEnable(uint8_t state);
```

### 2.1.2.8 Control robot hand/automatic mode

```
/**
* @brief Control robot hand/automatic mode
* @param [in] mode 0-automatic mode1-manual mode
* @return Error code
*/
errno_t  Mode(int mode);
```

### 2.1.2.9 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                //Instantiate the robot object
    robot.RPC("192.168.58.2");    //Establish a communication connection with the robot controller

    char ip[64]="";
    char version[64] = "";
    uint8_t state;

    robot.GetSDKVersion(version);
    printf("SDK version:%s\n", version);
    robot.GetControllerIP(ip);
    printf("controller ip:%s\n", ip);

    robot.Mode(1);
    sleep(1);
    robot.DragTeachSwitch(1);
    robot.IsInDragTeach(&state);
```

(continues on next page)

(continued from previous page)

```
    printf("drag state :%u\n", state);
    sleep(3);
    robot.DragTeachSwitch(0);
    sleep(1);
    robot.IsInDragTeach(&state);
    printf("drag state :%u\n", state);
    sleep(3);

    robot.RobotEnable(0);
    sleep(3);
    robot.RobotEnable(1);

    robot.Mode(0);
    sleep(1);
    robot.Mode(1);

    return 0;
}
```

## 2.1.3 Movement

### 2.1.3.1 Jog point movement

```
/**
* @brief  Jog point movement
* @param  [in]  ref 0- node movement, 2- base coordinate system, 4- tool coordinate system, 8- workpiece coordinate system
* @param  [in]  nb 1-joint 1(or axis x), 2-joint 2(or axis y), 3-joint 3(or axis z), 4-joint 4(or rotation about axis x), 5-joint 5(or rotation about axis y), 6-joint 6(or rotation about axis z)
* @param  [in]  dir 0-negative correlation, 1-positive correlation
* @param  [in]  vel The percentage of velocity,[0~100]
* @param  [in]  acc The percentage of acceleration, [0~100]
* @param  [in]  max_dis Maximum Angle of single click, unit: [°] or distance, unit: [mm]
* @return  Error code
*/
errno_t  StartJOG(uint8_t ref, uint8_t nb, uint8_t dir, float vel, float acc, float max_dis);
```

### 2.1.3.2 Jog point dynamic deceleration stop

```
/**
* @brief  Jog point dynamic deceleration stop
* @param  [in]  ref  1- point stop, 3- point stop in base coordinate system, 5- point stop in tool coordinate system, 9- point stop in workpiece coordinate system
* @return  Error code
*/
errno_t  StopJOG(uint8_t ref);
```

### 2.1.3.3 The jog stops immediately

```cpp
/**
* @brief The jog stops immediately
* @return  Error code
*/
errno_t  ImmStopJOG();
```

### 2.1.3.4 Code example

```cpp
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                //Instantiate the robot object
    robot.RPC("192.168.58.2");     //Establish a communication connection with the robot controller

    robot.StartJOG(0,1,0,20.0,20.0,30.0);   //For single-joint motion, StartJOG is a non-blocking command. Receiving other motion commands (including StartJOG) while in motion is discarded
    sleep(1);
    //robot.StopJOG(1)  //Robot single axis point deceleration stop
    robot.ImmStopJOG();  //The single axis of the robot stops immediately
    robot.StartJOG(0,2,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(0,3,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(0,4,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(0,5,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(0,6,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();

    robot.StartJOG(2,1,0,20.0,20.0,30.0);   //Point in the base coordinate system
    sleep(1);
    //robot.StopJOG(3)  //Robot single axis point deceleration stop
```

(continues on next page)

(continued from previous page)

```
    robot.ImmStopJOG();  //The single axis of the robot stops immediately
    robot.StartJOG(2,2,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(2,3,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(2,4,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(2,5,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(2,6,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();

    robot.StartJOG(4,1,0,20.0,20.0,30.0);   //Point in the tool coordinate system
    sleep(1);
    //robot.StopJOG(5)  //Robot single axis point deceleration stop
    robot.ImmStopJOG();  //The single axis of the robot stops immediately
    robot.StartJOG(4,2,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(4,3,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(4,4,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(4,5,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(4,6,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();

    robot.StartJOG(8,1,0,20.0,20.0,30.0);   //Point in the workpiece coordinate system
    sleep(1);
    //robot.StopJOG(9)  //Robot single axis point deceleration stop
    robot.ImmStopJOG();  //The single axis of the robot stops immediately
    robot.StartJOG(8,2,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(8,3,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(8,4,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();
    robot.StartJOG(8,5,1,20.0,20.0,30.0);
    sleep(1);
```

(continues on next page)

(continued from previous page)

```
    robot.ImmStopJOG();
    robot.StartJOG(8,6,1,20.0,20.0,30.0);
    sleep(1);
    robot.ImmStopJOG();

    return 0;
}
```

### 2.1.3.5 Joint space motion

```
/**
* @brief  Joint space motion
* @param  [in] joint_pos  Target joint location, unit: deg
* @param  [in] desc_pos   Target Cartesian position
* @param  [in] tool  Tool coordinate number, range [1~15]
* @param  [in] user  Workpiece coordinate number, range [1~15]
* @param  [in] vel  Percentage of speed, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not open for now
* @param  [in] ovl  Velocity scaling factor, range[0~100]
* @param  [in] epos  Position of expansion shaft, unit: mm
* @param  [in] blendT [-1.0]- movement in place (blocking), [0~500.0]- smoothing time␣
↪(non-blocking), in ms
* @param  [in] offset_flag  0- no offset, 1- offset in base/job coordinate system, 2-␣
↪offset in tool coordinate system
* @param  [in] offset_pos  The pose offset
* @return  Error code
*/
errno_t  MoveJ(JointPos *joint_pos, DescPose *desc_pos, int tool, int user, float vel,␣
↪float acc, float ovl, ExaxisPos *epos, float blendT, uint8_t offset_flag, DescPose␣
↪*offset_pos);
```

### 2.1.3.6 Rectilinear motion in Cartesian space

```
/**
* @brief  Rectilinear motion in Cartesian space
* @param  [in] joint_pos  Target joint location, unit: deg
* @param  [in] desc_pos   Target Cartesian position
* @param  [in] tool  Tool coordinate number, range [1~15]
* @param  [in] user  Workpiece coordinate number, range [1~15]
* @param  [in] vel  Percentage of speed, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not open for now
* @param  [in] ovl  Velocity scaling factor, range[0~100]
* @param  [in] blendR [-1.0]- movement in place (blocking), [0~1000.0]- Smoothing radius␣
↪(non-blocking), unit: mm
* @param  [in] epos  Position of expansion shaft, unit: mm
* @param  [in] search  0- no wire seeking, 1- wire seeking
* @param  [in] offset_flag  0- no offset, 1- offset in base/job coordinate system, 2-␣
↪offset in tool coordinate system
* @param  [in] offset_pos  The pose offset
```

(continues on next page)

(continued from previous page)

```
* @return  Error code
*/
errno_t  MoveL(JointPos *joint_pos, DescPose *desc_pos, int tool, int user, float vel, float acc, float ovl, float blendR, ExaxisPos *epos, uint8_t search, uint8_t offset_flag, DescPose *offset_pos);
```

#### 2.1.3.7 Circular arc motion in Cartesian space

```
/**
* @brief  Circular arc motion in Cartesian space
* @param  [in] joint_pos_p  Waypoint joint position, unit: deg
* @param  [in] desc_pos_p   Waypoint Cartesian position
* @param  [in] ptool  Tool coordinate number, range [1~15]
* @param  [in] puser  Workpiece coordinate number, range [1~15]
* @param  [in] pvel  Percentage of speed, range [0~100]
* @param  [in] pacc  Acceleration percentage, range [0~100], not open for now
* @param  [in] epos_p  Position of expansion shaft, unit: mm
* @param  [in] poffset_flag  0- no offset, 1- offset in base/job coordinate system, 2- offset in tool coordinate system
* @param  [in] offset_pos_p  The pose offset
* @param  [in] joint_pos_t  Target joint position, unit: deg
* @param  [in] desc_pos_t   Target point Cartesian position
* @param  [in] ttool  Tool coordinate number, range [1~15]
* @param  [in] tuser  Workpiece coordinate number, range [1~15]
* @param  [in] tvel  Percentage of speed, range [0~100]
* @param  [in] tacc  Acceleration percentage, range [0~100], not open for now
* @param  [in] epos_t  Position of expansion shaft, unit: mm
* @param  [in] toffset_flag  0- no offset, 1- offset in base/job coordinate system, 2- offset in tool coordinate system
* @param  [in] offset_pos_t  The pose offset
* @param  [in] ovl  Velocity scaling factor, range[0~100]
* @param  [in] blendR [-1.0]- movement in place (blocking), [0~1000.0]- Smoothing radius (non-blocking), unit: mm
* @return  Error code
*/
errno_t  MoveC(JointPos *joint_pos_p, DescPose *desc_pos_p, int ptool, int puser, float pvel, float pacc, ExaxisPos *epos_p, uint8_t poffset_flag, DescPose *offset_pos_p, JointPos *joint_pos_t, DescPose *desc_pos_t, int ttool, int tuser, float tvel, float tacc, ExaxisPos *epos_t, uint8_t toffset_flag, DescPose *offset_pos_t,float ovl, float blendR);
```

#### 2.1.3.8 Circular motion in Cartesian space

```
/**
* @brief  Circular motion in Cartesian space
* @param  [in] joint_pos_p  Path point 1 joint position, unit: deg
* @param  [in] desc_pos_p   Waypoint 1 Cartesian position
* @param  [in] ptool  Tool coordinate number, range [1~15]
* @param  [in] puser  Workpiece coordinate number, range [1~15]
* @param  [in] pvel  Percentage of speed, range [0~100]
* @param  [in] pacc  Acceleration percentage, range [0~100], not open for now
* @param  [in] epos_p  Position of expansion shaft, unit: mm
* @param  [in] joint_pos_t  Joint position at waypoint 2, unit: deg
* @param  [in] desc_pos_t   Waypoint 2 Cartesian position
* @param  [in] ttool  Tool coordinate number, range [1~15]
* @param  [in] tuser  Workpiece coordinate number, range [1~15]
* @param  [in] tvel  Percentage of speed, range [0~100]
* @param  [in] tacc  Acceleration percentage, range [0~100], not open for now
* @param  [in] epos_t  Position of expansion shaft, unit: mm
* @param  [in] ovl  Velocity scaling factor, range[0~100]
* @param  [in] offset_flag  0- no offset, 1- offset in base/job coordinate system, 2- offset in tool coordinate system
* @param  [in] offset_pos  The pose offset
* @return  Error code
*/
errno_t  Circle(JointPos *joint_pos_p, DescPose *desc_pos_p, int ptool, int puser, float pvel, float pacc, ExaxisPos *epos_p, JointPos *joint_pos_t, DescPose *desc_pos_t, int ttool, int tuser, float tvel, float tacc, ExaxisPos *epos_t, float ovl, uint8_t offset_flag, DescPose *offset_pos);
```

#### 2.1.3.9 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    JointPos j1,j2,j3,j4;
    DescPose desc_pos1,desc_pos2,desc_pos3,desc_pos4,offset_pos;
    ExaxisPos  epos;

```

(continues on next page)

(continued from previous page)

```
    memset(&j1, 0, sizeof(JointPos));
    memset(&j2, 0, sizeof(JointPos));
    memset(&j3, 0, sizeof(JointPos));
    memset(&j4, 0, sizeof(JointPos));
    memset(&desc_pos1, 0, sizeof(DescPose));
    memset(&desc_pos2, 0, sizeof(DescPose));
    memset(&desc_pos3, 0, sizeof(DescPose));
    memset(&desc_pos4, 0, sizeof(DescPose));
    memset(&offset_pos, 0, sizeof(DescPose));
    memset(&epos, 0, sizeof(ExaxisPos));

    j1 = {114.578,-117.798,-97.745,-54.436,90.053,-45.216};
    desc_pos1.tran.x = -140.418;
    desc_pos1.tran.y = 619.351;
    desc_pos1.tran.z = 198.369;
    desc_pos1.rpy.rx = -179.948;
    desc_pos1.rpy.ry = 0.023;
    desc_pos1.rpy.rz = 69.793;

    j2 = {121.381,-97.108,-123.768,-45.824,89.877,-47.296};
    desc_pos2.tran.x = -127.772;
    desc_pos2.tran.y = 459.534;
    desc_pos2.tran.z = 221.274;
    desc_pos2.rpy.rx = -177.850;
    desc_pos2.rpy.ry = -2.507;
    desc_pos2.rpy.rz = 78.627;

    j3 = {138.884,-114.522,-103.933,-49.694,90.688,-47.291};
    desc_pos3.tran.x = -360.468;
    desc_pos3.tran.y = 485.600;
    desc_pos3.tran.z = 196.363;
    desc_pos3.rpy.rx = -178.239;
    desc_pos3.rpy.ry = -0.893;
    desc_pos3.rpy.rz = 96.172;

    j4 = {159.164,-96.105,-128.653,-41.170,90.704,-47.290};
    desc_pos4.tran.x = -360.303;
    desc_pos4.tran.y = 274.911;
    desc_pos4.tran.z = 203.968;
    desc_pos4.rpy.rx = -176.720;
    desc_pos4.rpy.ry = -2.514;
    desc_pos4.rpy.rz = 116.407;

    int tool = 0;
    int user = 0;
    float vel = 100.0;
    float acc = 100.0;
    float ovl = 100.0;
    float blendT = 0.0;
    float blendR = 0.0;
    uint8_t flag = 0;
    uint8_t search = 0;
```

(continues on next page)

(continued from previous page)

```
    robot.SetSpeed(20);

    int err1 = robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT,flag, &offset_pos);
    printf("movej errcode:%d\n", err1);

    int err2 = robot.MoveL(&j2, &desc_pos2, tool, user, vel, acc, ovl, blendR, &epos,search,flag, &offset_pos);
    printf("movel errcode:%d\n", err2);

    int err3 = robot.MoveC(&j3,&desc_pos3,tool,user,vel,acc,&epos,flag,&offset_pos,&j4,&desc_pos4,tool,user,vel,acc,&epos,flag,&offset_pos,ovl,blendR);
    printf("movec errcode:%d\n", err3);

    int err4 = robot.MoveJ(&j2, &desc_pos2, tool, user, vel, acc, ovl, &epos, blendT,flag, &offset_pos);
    printf("movej errcode:%d\n", err4);

    int err5 = robot.Circle(&j3,&desc_pos3,tool,user,vel,acc,&epos,&j4,&desc_pos4,tool,user,vel,acc,&epos,ovl,flag,&offset_pos);
    printf("circle errcode:%d\n", err5);

    return 0;
}
```

#### 2.1.3.10 Spiral motion in Cartesian space

```
/**
* @brief  Spiral motion in Cartesian space
* @param  [in] joint_pos  Target joint location, unit: deg
* @param  [in] desc_pos   Target Cartesian position
* @param  [in] tool  Tool coordinate number, range [1~15]
* @param  [in] user  Workpiece coordinate number, range [1~15]
* @param  [in] vel  Percentage of speed, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not open for now
* @param  [in] epos  Position of expansion shaft, unit: mm
* @param  [in] ovl  Velocity scaling factor, range[0~100]
* @param  [in] offset_flag  0- no offset, 1- offset in base/job coordinate system, 2- offset in tool coordinate system
* @param  [in] offset_pos  The pose offset
* @param  [in] spiral_param  Spiral parameter
* @return  Error code
*/
errno_t  NewSpiral(JointPos *joint_pos, DescPose *desc_pos, int tool, int user, float vel, float acc, ExaxisPos *epos, float ovl, uint8_t offset_flag, DescPose *offset_pos, SpiralParam spiral_param);
```

### 2.1.3.11 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    JointPos j;
    DescPose desc_pos, offset_pos1, offset_pos2;
    ExaxisPos  epos;
    SpiralParam sp;

    memset(&j, 0, sizeof(JointPos));
    memset(&desc_pos, 0, sizeof(DescPose));
    memset(&offset_pos1, 0, sizeof(DescPose));
    memset(&offset_pos2, 0, sizeof(DescPose));
    memset(&epos, 0, sizeof(ExaxisPos));
    memset(&sp, 0, sizeof(SpiralParam));

    j = {127.888,-101.535,-94.860,17.836,96.931,-61.325};
    offset_pos1.tran.x = 50.0;
    offset_pos1.rpy.rx = -30.0;
    offset_pos2.tran.x = 50.0;
    offset_pos2.rpy.rx = -5.0;

    sp.circle_num = 5;
    sp.circle_angle = 5.0;
    sp.rad_init = 50.0;
    sp.rad_add = 10.0;
    sp.rotaxis_add = 10.0;
    sp.rot_direction = 0;

    int tool = 0;
    int user = 0;
    float vel = 100.0;
    float acc = 100.0;
    float ovl = 100.0;
    float blendT = 0.0;
    uint8_t flag = 2;

    robot.SetSpeed(20);

```

(continues on next page)

(continued from previous page)

```
    int ret = robot.GetForwardKin(&j, &desc_pos);  //The forward kinematic interface can be used to solve Cartesian space coordinates with only joint positions

    if(ret == 0)
    {
        int err1 = robot.MoveJ(&j, &desc_pos, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_pos1);
        printf("movej errcode:%d\n", err1);

        int err2 = robot.NewSpiral(&j, &desc_pos, tool, user, vel, acc, &epos, ovl, flag, &offset_pos2, sp);
        printf("newspiral errcode:%d\n", err2);
    }
    else
    {
        printf("GetForwardKin errcode:%d\n", ret);
    }

    return 0;
}
```

#### 2.1.3.12 Joint space servo mode motion

```
/**
* @brief  Joint space servo mode motion
* @param  [in] joint_pos  Target joint location, unit: deg
* @param  [in] acc  Acceleration percentage range[0~100], not open yet, default: 0
* @param  [in] vel  The value ranges from 0 to 100. The value is not available. The default value is 0
* @param  [in] cmdT Instruction delivery period, unit: s, recommended range [0.001~0.0016]
* @param  [in] filterT Filtering time (unit: s), temporarily disabled. The default value is 0
* @param  [in] gain  The proportional amplifier at the target position, not yet open, defaults to 0
* @return  Error code
*/
errno_t  ServoJ(JointPos *joint_pos, float acc, float vel, float cmdT, float filterT, float gain);
```

#### 2.1.3.13 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"
```

(continues on next page)

(continued from previous page)

```

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    JointPos j;

    memset(&j, 0, sizeof(JointPos));

    float vel = 0.0;
    float acc = 0.0;
    float cmdT = 0.008;
    float filterT = 0.0;
    float gain = 0.0;
    uint8_t flag = 0;
    int count = 500;
    float dt = 0.1;

    int ret = robot.GetActualJointPosDegree(flag, &j);
    if(ret == 0)
    {
        while (count)
        {
            robot.ServoJ(&j, acc, vel, cmdT, filterT, gain);
            j.jPos[0] += dt;
            count -= 1;
            robot.WaitMs(cmdT*1000);
        }
    }
    else
    {
        printf("GetActualJointPosDegree errcode:%d\n", ret);
    }

    return 0;
}
```

#### 2.1.3.14 Cartesian space servo mode motion

```
/**
* @brief  Cartesian space servo mode motion
* @param  [in]  mode  0- absolute motion (base coordinates), 1- incremental motion (base coordinates), 2- incremental motion (tool coordinates)
* @param  [in]  desc_pos  Target Cartesian pose or pose increment
* @param  [in]  pos_gain  Proportional coefficient of pose increment, effective only for incremental motion, range [0~1]
```

(continues on next page)

(continued from previous page)

```
* @param  [in] acc  Acceleration percentage range[0~100], not open yet, default: 0
* @param  [in] vel  The value ranges from 0 to 100. The value is not available. The␣
↪default value is 0
* @param  [in] cmdT Instruction delivery period, unit: s, recommended range [0.001~0.
↪0016]
* @param  [in] filterT Filtering time (unit: s), temporarily disabled. The default value␣
↪is 0
* @param  [in] gain  The proportional amplifier at the target position, not yet open,␣
↪defaults to 0
* @return  Error code
*/
errno_t  ServoCart(int mode, DescPose *desc_pose, float pos_gain[6], float acc, float␣
↪vel, float cmdT, float filterT, float gain);
```

#### 2.1.3.15 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot␣
↪controller

    DescPose desc_pos_dt;
    memset(&desc_pos_dt, 0, sizeof(DescPose));

    desc_pos_dt.tran.z = -0.5;
    float pos_gain[6] = {0.0,0.0,1.0,0.0,0.0,0.0};
    int mode = 2;
    float vel = 0.0;
    float acc = 0.0;
    float cmdT = 0.008;
    float filterT = 0.0;
    float gain = 0.0;
    uint8_t flag = 0;
    int count = 100;

    robot.SetSpeed(20);

    while (count)
    {
```

(continues on next page)

(continued from previous page)

```
        robot.ServoCart(mode, &desc_pos_dt, pos_gain, acc, vel, cmdT, filterT, gain);
        count -= 1;
        robot.WaitMs(cmdT*1000);
    }

    return 0;
}
```

#### 2.1.3.16 Point to point motion in Cartesian space

```
/**
* @brief  Point to point motion in Cartesian space
* @param  [in]  desc_pos  Target Cartesian pose or pose increment
* @param  [in] tool  Tool coordinate number, range [1~15]
* @param  [in] user  Workpiece coordinate number, range [1~15]
* @param  [in] vel  Percentage of speed, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not open for now
* @param  [in] ovl  Velocity scaling factor, range[0~100]
* @param  [in] blendT [-1.0]- movement in place (blocking), [0~500.0]- smoothing time (non-blocking), in ms
* @param  [in] config  Joint space configuration, [-1]- refer to the current joint position, [0~7]- refer to the specific joint space configuration, the default is -1
* @return  Error code
*/
errno_t  MoveCart(DescPose *desc_pos, int tool, int user, float vel, float acc, float ovl, float blendT, int config);
```

#### 2.1.3.17 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    DescPose desc_pos1, desc_pos2, desc_pos3;
    memset(&desc_pos1, 0, sizeof(DescPose));
    memset(&desc_pos2, 0, sizeof(DescPose));
    memset(&desc_pos3, 0, sizeof(DescPose));
```

(continues on next page)

(continued from previous page)

```

    desc_pos1.tran.x = 75.414;
    desc_pos1.tran.y = 568.526;
    desc_pos1.tran.z = 338.135;
    desc_pos1.rpy.rx = -178.348;
    desc_pos1.rpy.ry = -0.930;
    desc_pos1.rpy.rz = 52.611;

    desc_pos2.tran.x = -273.856;
    desc_pos2.tran.y = 643.260;
    desc_pos2.tran.z = 259.235;
    desc_pos2.rpy.rx = -177.972;
    desc_pos2.rpy.ry = -1.494;
    desc_pos2.rpy.rz = 80.866;

    desc_pos3.tran.x = -423.044;
    desc_pos3.tran.y = 229.703;
    desc_pos3.tran.z = 241.080;
    desc_pos3.rpy.rx = -173.990;
    desc_pos3.rpy.ry = -5.772;
    desc_pos3.rpy.rz = 123.971;

    int tool = 0;
    int user = 0;
    float vel = 100.0;
    float acc = 100.0;
    float ovl = 100.0;
    float blendT = -1.0;
    float blendT1 = 0.0;
    int config = -1;

    robot.SetSpeed(20);
    robot.MoveCart(&desc_pos1, tool, user, vel, acc, ovl, blendT, config);
    robot.MoveCart(&desc_pos2, tool, user, vel, acc, ovl, blendT, config);
    robot.MoveCart(&desc_pos3, tool, user, vel, acc, ovl, blendT1, config);

    return 0;
}
```

#### 2.1.3.18 The spline motion begins

```
/**
* @brief  The spline motion begins
* @return  Error code
*/
errno_t  SplineStart();
```

#### 2.1.3.19 Spline motion PTP

```
/**
* @brief  Joint space spline movement
* @param  [in] joint_pos  Target joint location, unit: deg
* @param  [in] desc_pos   Target Cartesian position
* @param  [in] tool  Tool coordinate number, range [1~15]
* @param  [in] user  Workpiece coordinate number, range [1~15]
* @param  [in] vel  Percentage of speed, range [0~100]
* @param  [in] acc  Acceleration percentage, range [0~100], not open for now
* @param  [in] ovl  Velocity scaling factor, range[0~100]
* @return  Error code
*/
errno_t  SplinePTP(JointPos *joint_pos, DescPose *desc_pos, int tool, int user, float vel, float acc, float ovl);
```

#### 2.1.3.20 The spline movement ends

```
/**
* @brief  The spline movement is complete
* @return  Error code
*/
errno_t  SplineEnd();
```

#### 2.1.3.21 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                //Instantiate the robot object
    robot.RPC("192.168.58.2");     //Establish a communication connection with the robot controller

    JointPos j1,j2,j3,j4;
    DescPose desc_pos1,desc_pos2,desc_pos3,desc_pos4,offset_pos;
    ExaxisPos  epos;

    memset(&j1, 0, sizeof(JointPos));
    memset(&j2, 0, sizeof(JointPos));
    memset(&j3, 0, sizeof(JointPos));
    memset(&j4, 0, sizeof(JointPos));
```

(continues on next page)

(continued from previous page)

```
    memset(&desc_pos1, 0, sizeof(DescPose));
    memset(&desc_pos2, 0, sizeof(DescPose));
    memset(&desc_pos3, 0, sizeof(DescPose));
    memset(&desc_pos4, 0, sizeof(DescPose));
    memset(&offset_pos, 0, sizeof(DescPose));
    memset(&epos, 0, sizeof(ExaxisPos));

    j1 = {114.578,-117.798,-97.745,-54.436,90.053,-45.216};
    desc_pos1.tran.x = -140.418;
    desc_pos1.tran.y = 619.351;
    desc_pos1.tran.z = 198.369;
    desc_pos1.rpy.rx = -179.948;
    desc_pos1.rpy.ry = 0.023;
    desc_pos1.rpy.rz = 69.793;

    j2 = {115.401,-105.206,-117.959,-49.727,90.054,-45.222};
    desc_pos2.tran.x = -95.586;
    desc_pos2.tran.y = 504.143;
    desc_pos2.tran.z = 186.880;
    desc_pos2.rpy.rx = 178.001;
    desc_pos2.rpy.ry = 2.091;
    desc_pos2.rpy.rz = 70.585;

    j3 = {135.609,-103.249,-120.211,-49.715,90.058,-45.219};
    desc_pos3.tran.x = -252.429;
    desc_pos3.tran.y = 428.903;
    desc_pos3.tran.z = 188.492;
    desc_pos3.rpy.rx = 177.804;
    desc_pos3.rpy.ry = 2.294;
    desc_pos3.rpy.rz = 90.782;

    j4 = {154.766,-87.036,-135.672,-49.045,90.739,-45.223};
    desc_pos4.tran.x = -277.255;
    desc_pos4.tran.y = 272.958;
    desc_pos4.tran.z = 205.452;
    desc_pos4.rpy.rx = 179.289;
    desc_pos4.rpy.ry = 1.765;
    desc_pos4.rpy.rz = 109.966;

    int tool = 0;
    int user = 0;
    float vel = 100.0;
    float acc = 100.0;
    float ovl = 100.0;
    float blendT = -1.0;
    uint8_t flag = 0;

    robot.SetSpeed(20);

    int err1 = robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT, flag, &offset_pos);
    printf("movej errcode:%d\n", err1);
```

(continues on next page)

(continued from previous page)

```
    robot.SplineStart();
    robot.SplinePTP(&j1, &desc_pos1, tool, user, vel, acc, ovl);
    robot.SplinePTP(&j2, &desc_pos2, tool, user, vel, acc, ovl);
    robot.SplinePTP(&j3, &desc_pos3, tool, user, vel, acc, ovl);
    robot.SplinePTP(&j4, &desc_pos4, tool, user, vel, acc, ovl);
    robot.SplineEnd();

    return 0;
}
```

#### 2.1.3.22 Termination motion

```
/**
* @brief Termination motion
* @return  Error code
*/
errno_t  StopMotion();
```

#### 2.1.3.23 The whole point shift begins

```
/**
* @brief  The whole point shift begins
* @param  [in]  flag 0- offset in base coordinate system/workpiece coordinate system, 2- offset in tool coordinate system
* @param  [in] offset_pos  The pose offset
* @return  Error code
*/
errno_t  PointsOffsetEnable(int flag, DescPose *offset_pos);
```

#### 2.1.3.24 The whole point shift ends

```
/**
* @brief  The whole point shift ends
* @return  Error code
*/
errno_t  PointsOffsetDisable();
```

#### 2.1.3.25 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"
```

(continues on next page)

(continued from previous page)

```cpp

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    JointPos j1,j2;
    DescPose desc_pos1,desc_pos2,offset_pos,offset_pos1;
    ExaxisPos  epos;

    memset(&j1, 0, sizeof(JointPos));
    memset(&j2, 0, sizeof(JointPos));
    memset(&desc_pos1, 0, sizeof(DescPose));
    memset(&desc_pos2, 0, sizeof(DescPose));
    memset(&offset_pos, 0, sizeof(DescPose));
    memset(&offset_pos1, 0, sizeof(DescPose));
    memset(&epos, 0, sizeof(ExaxisPos));

    j1 = {114.578,-117.798,-97.745,-54.436,90.053,-45.216};
    desc_pos1.tran.x = -140.418;
    desc_pos1.tran.y = 619.351;
    desc_pos1.tran.z = 198.369;
    desc_pos1.rpy.rx = -179.948;
    desc_pos1.rpy.ry = 0.023;
    desc_pos1.rpy.rz = 69.793;

    j2 = {115.401,-105.206,-117.959,-49.727,90.054,-45.222};
    desc_pos2.tran.x = -95.586;
    desc_pos2.tran.y = 504.143;
    desc_pos2.tran.z = 186.880;
    desc_pos2.rpy.rx = 178.001;
    desc_pos2.rpy.ry = 2.091;
    desc_pos2.rpy.rz = 70.585;

    offset_pos1.tran.x = 100.0;
    offset_pos1.tran.y = 100.0;
    offset_pos1.tran.z = 100.0;
    offset_pos1.rpy.rx = 5.0;
    offset_pos1.rpy.ry = 5.0;
    offset_pos1.rpy.rz = 5.0;

    int tool = 0;
    int user = 0;
    float vel = 100.0;
    float acc = 100.0;
    float ovl = 100.0;
    float blendT = -1.0;
    float blendR = 0.0;
    uint8_t flag = 0;
```

(continues on next page)

(continued from previous page)

```
    int type = 0;

    robot.SetSpeed(20);

    robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT,flag, &offset_pos);
    robot.MoveJ(&j2, &desc_pos2, tool, user, vel, acc, ovl, &epos, blendT,flag, &offset_pos);
    sleep(2);
    robot.PointsOffsetEnable(type, &offset_pos1);
    robot.MoveJ(&j1, &desc_pos1, tool, user, vel, acc, ovl, &epos, blendT,flag, &offset_pos);
    robot.MoveJ(&j2, &desc_pos2, tool, user, vel, acc, ovl, &epos, blendT,flag, &offset_pos);
    robot.PointsOffsetDisable();

    return 0;
}
```

## 2.1.4 IO

### 2.1.4.1 Set the control box digital output

```
/**
* @brief  Set the control box digital output
* @param  [in] id  I/O number and range[0~15]
* @param  [in] status 0- off, 1- on
* @param  [in] smooth 0- Not smooth, 1- smooth
* @param  [in] block  0- blocking, 1- non-blocking
* @return  Error code
*/
errno_t  SetDO(int id, uint8_t status, uint8_t smooth, uint8_t block);
```

### 2.1.4.2 Set tool digital output

```
/**
* @brief  Set tool digital output
* @param  [in] id  I/O number and range[0~1]
* @param  [in] status 0- off, 1- on
* @param  [in] smooth 0- not smooth, 1- smooth
* @param  [in] block  0- blocking, 1- non-blocking
* @return  Error code
*/
errno_t  SetToolDO(int id, uint8_t status, uint8_t smooth, uint8_t block);
```

#### 2.1.4.3 Set control box analog output

```
/**
* @brief  Set control box analog output
* @param  [in] id  I/O number and range[0~1]
* @param  [in] value Percentage of current or voltage value, range [0~100] corresponding to current value [0~20mA] or voltage [0~10V]
* @param  [in] block  0- blocking, 1- non-blocking
* @return  Error code
*/
errno_t  SetAO(int id, float value, uint8_t block);
```

#### 2.1.4.4 Set tool analog output

```
/**
* @brief  Set tool analog output
* @param  [in] id  I/O number, range [0]
* @param  [in] value Percentage of current or voltage value, range [0~100] corresponding to current value [0~20mA] or voltage [0~10V]
* @param  [in] block  0- blocking, 1- non-blocking
* @return  Error code
*/
errno_t  SetToolAO(int id, float value, uint8_t block);
```

#### 2.1.4.5 Get the control box digital input

```
/**
* @brief  Get the control box digital input
* @param  [in] id  I/O number range[0~15]
* @param  [in] block  0- blocking, 1- non-blocking
* @param  [out] result  0- low, 1- high
* @return  Error code
*/
errno_t  GetDI(int id, uint8_t block, uint8_t *result);
```

#### 2.1.4.6 Get tool numeric input

```
/**
* @brief  Get tool numeric input
* @param  [in] id  I/O number, range[0~1]
* @param  [in] block  0- blocking, 1- non-blocking
* @param  [out] result  0- low, 1- high
* @return  Error code
*/
errno_t  GetToolDI(int id, uint8_t block, uint8_t *result);
```

### 2.1.4.7 Wait for the control box digital input

```
/**
* @brief Wait for the control box digital input
* @param  [in] id  I/O numberrange[0~15]
* @param  [in]  status 0- off, 1- on
* @param  [in]  max_time  Maximum waiting time, expressed in ms
* @param  [in]  opt  After timeout policy, 0- program stops and prompts timeout, 1- ignores timeout prompts and continues execution, 2- waits
* @return  Error code
*/
errno_t  WaitDI(int id, uint8_t status, int max_time, int opt);
```

### 2.1.4.8 Wait for control box multiplex digital input

```
/**
* @brief Wait for control box multiplex digital input
* @param  [in] mode 0- multiplexed and, 1- multiplexed or
* @param  [in] id  I/O numbers. bit0 to bit7 corresponds to DI0 to DI7, and bit8 to bit15 corresponds to CI0 to CI7
* @param  [in]  status 0- off, 1- on
* @param  [in]  max_time  Maximum waiting time, expressed in ms
* @param  [in]  opt  After timeout policy, 0- program stops and prompts timeout, 1- ignores timeout prompts and continues execution, 2- waits
* @return  Error code
*/
errno_t  WaitMultiDI(int mode, int id, uint8_t status, int max_time, int opt);
```

### 2.1.4.9 Wait for the tool number to enter

```
/**
* @brief Wait for the tool number to enter
* @param  [in] id  I/O numbersrange[0~1]
* @param  [in]  status 0- off, 1- on
* @param  [in]  max_time  Maximum waiting time, expressed in ms
* @param  [in]  opt  After timeout policy, 0- program stops and prompts timeout, 1- ignores timeout prompts and continues execution, 2- waits
* @return  Error code
*/
errno_t  WaitToolDI(int id, uint8_t status, int max_time, int opt);
```

### 2.1.4.10 Get control box analog input

```
/**
* @brief  Get control box analog input
* @param  [in] id  I/O numbersrange[0~1]
* @param  [in] block  0- blocking, 1- non-blocking
* @param  [out] result  Percentage of input current or voltage value, range [0-100] corresponding to current value [0-20ms] or voltage [0-10V]
* @return  Error code
*/
errno_t  GetAI(int id, uint8_t block, float *result);
```

### 2.1.4.11 Get the tool analog input

```
/**
* @brief  Get the tool analog input
* @param  [in] id  I/O numbersrange[0~1]
* @param  [in] block  0- blocking, 1- non-blocking
* @param  [out] result  Percentage of input current or voltage value, range [0-100] corresponding to current value [0-20ms] or voltage [0-10V]
* @return  Error code
*/
errno_t  GetToolAI(int id, uint8_t block, float *result);
```

### 2.1.4.12 Wait for control box analog input

```
/**
* @brief Wait for control box analog input
* @param  [in] id  I/O numbersrange[0~1]
* @param  [in]  sign 0-greater than1-less than
* @param  [in]  value Percentage of input current or voltage value, range [0-100] corresponding to current value [0-20ms] or voltage [0-10V]
* @param  [in]  max_time Maximum waiting time, expressed in ms
* @param  [in]  opt  After timeout policy, 0- program stops and prompts timeout, 1- ignores timeout prompts and continues execution, 2- waits
* @return  Error code
*/
errno_t  WaitAI(int id, int sign, float value, int max_time, int opt);
```

### 2.1.4.13 Wait for tool analog input

```
/**
* @brief Wait for tool analog input
* @param  [in] id  I/O numbersrange[0~1]
* @param  [in]  sign 0-greater than1-less than
* @param  [in]  value Percentage of input current or voltage value, range [0-100] corresponding to current value [0-20ms] or voltage [0-10V]
* @param  [in]  max_time  Maximum waiting time, expressed in ms
```

(continues on next page)

(continued from previous page)

```
* @param  [in]  opt  After timeout policy, 0- program stops and prompts timeout, 1- ignores timeout prompts and continues execution, 2- waits
* @return  Error code
*/
errno_t  WaitToolAI(int id, int sign, float value, int max_time, int opt);
```

#### 2.1.4.14 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    uint8_t status = 1;
    uint8_t smooth = 0;
    uint8_t block  = 0;
    uint8_t di = 0, tool_di = 0;
    float ai = 0.0, tool_ai = 0.0;
    float value = 0.0;
    int i;

    for(i = 0; i < 16; i++)
    {
        robot.SetDO(i, status, smooth, block);
        robot.WaitMs(1000);
    }

    status = 0;

    for(i = 0; i < 16; i++)
    {
        robot.SetDO(i, status, smooth, block);
        robot.WaitMs(1000);
    }

    status = 1;

    for(i = 0; i < 2; i++)
    {
```

(continues on next page)

(continued from previous page)

```
        robot.SetToolDO(i, status, smooth, block);
        robot.WaitMs(1000);
    }

    status = 0;

    for(i = 0; i < 2; i++)
    {
        robot.SetToolDO(i, status, smooth, block);
        robot.WaitMs(1000);
    }

    value = 50.0;
    robot.SetAO(0, value, block);
    value = 100.0;
    robot.SetAO(1, value, block);
    robot.WaitMs(1000);
    value = 0.0;
    robot.SetAO(0, value, block);
    value = 0.0;
    robot.SetAO(1, value, block);

    value = 100.0;
    robot.SetToolAO(0, value, block);
    robot.WaitMs(1000);
    value = 0.0;
    robot.SetToolAO(0, value, block);

    robot.GetDI(0, block, &di);
    printf("di0:%u\n", di);
    robot.WaitDI(0,1,0,2);              //Have been waiting
    robot.WaitMultiDI(1,3,3,10000,2);   //Have been waiting
    tool_di = robot.GetToolDI(1, block, &tool_di);
    printf("tool_di1:%u\n", tool_di);
    robot.WaitToolDI(1,1,0,2);          //Have been waiting

    robot.GetAI(0,block, &ai);
    printf("ai0:%f\n", ai);
    robot.WaitAI(0,0,50,0,2);           //Have been waiting
    robot.WaitToolAI(0,0,50,0,2);       //Have been waiting
    tool_ai = robot.GetToolAI(0,block, &tool_ai);
    printf("tool_ai0:%f\n", tool_ai);

    return 0;
}
```

### 2.1.5 Common Settings

#### 2.1.5.1 Set global speed

```
/**
* @brief  Set global speed
* @param  [in]  vel  Percentage of velocity, range[0~100]
* @return  Error code
*/
errno_t  SetSpeed(int vel);
```

#### 2.1.5.2 Set the value of a system variable

```
/**
* @brief  Set the value of a system variable
* @param  [in]  id  Variable number, range[1~20]
* @param  [in]  value Variable value
* @return  Error code
*/
errno_t  SetSysVarValue(int id, float value);
```

#### 2.1.5.3 Set tool coordinate system

```
/**
* @brief  Set tool coordinate system
* @param  [in] id Frame number, range[1~15]
* @param  [in] coord  Tool center position relative to end flange center position
* @param  [in] type  0- tool coordinates, 1- sensor coordinates
* @param  [in] install Installation position, 0- robot end, 1- robot outside
* @return  Error code
*/
errno_t  SetToolCoord(int id, DescPose *coord, int type, int install);
```

#### 2.1.5.4 Set the tool coordinate list

```
/**
* @brief  Set the tool coordinate list
* @param  [in] id Frame number, range[1~15]
* @param  [in] coord  Tool center position relative to end flange center position
* @param  [in] type  0- tool coordinates, 1- sensor coordinates
* @param  [in] install Installation position, 0- robot end, 1- robot outside
* @return  Error code
*/
errno_t  SetToolList(int id, DescPose *coord, int type, int install);
```

#### 2.1.5.5 Set the external tool coordinate system

```
/**
* @brief  Set the external tool coordinate system
* @param  [in] id Frame number, range[1~15]
* @param  [in] etcp  Tool center position relative to end flange center position
* @param  [in] etool  To be determined
* @return  Error code
*/
errno_t  SetExToolCoord(int id, DescPose *etcp, DescPose *etool);
```

#### 2.1.5.6 Set the list of external tool coordinate systems

```
/**
* @brief  Set the list of external tool coordinate systems
* @param  [in] id Frame number, range[1~15]
* @param  [in] etcp  Tool center position relative to end flange center position
* @param  [in] etool  To be determined
* @return  Error code
*/
errno_t  SetExToolList(int id, DescPose *etcp, DescPose *etool);
```

#### 2.1.5.7 Set the workpiece coordinate system

```
/**
* @brief  Set the workpiece coordinate system
* @param  [in] id Frame number, range[1~15]
* @param  [in] coord  Tool center position relative to end flange center position
* @return  Error code
*/
errno_t  SetWObjCoord(int id, DescPose *coord);
```

#### 2.1.5.8 Set the list of work coordinate systems

```
/**
* @brief  Set the list of work coordinate systems
* @param  [in] id Frame number, range[1~15]
* @param  [in] coord  Tool center position relative to end flange center position
* @return  Error code
*/
errno_t  SetWObjList(int id, DescPose *coord);
```

#### 2.1.5.9 Set the end load weight

```
/**
* @brief  Set the end load weight
* @param  [in] weight  Load weight, unit: kg
* @return  Error code
*/
errno_t  SetLoadWeight(float weight);
```

#### 2.1.5.10 Set the end-load centroid coordinates

```
/**
* @brief  Set the end-load centroid coordinates
* @param  [in] coord Centroid coordinates, unit: mm
* @return  Error code
*/
errno_t  SetLoadCoord(DescTran *coord);
```

#### 2.1.5.11 Set the robot installation mode

```
/**
* @brief  Set the robot installation mode
* @param  [in] install  Installation mode: 0- formal installation, 1- side installation,␣
↪2- inverted installation
* @return  Error code
*/
errno_t  SetRobotInstallPos(uint8_t install);
```

#### 2.1.5.12 Set the robot installation Angle

```
/**
* @brief  Set the robot installation Angle, free installation
* @param  [in] yangle  Angle of inclination
* @param  [in] zangle  Angle of rotation
* @return  Error code
*/
errno_t  SetRobotInstallAngle(double yangle, double zangle);
```

#### 2.1.5.13 Wait for the specified time

```
/**
* @brief  Wait for the specified time
* @param  [in]  t_ms  unit: ms
* @return  Error code
*/
errno_t  WaitMs(int t_ms);
```

### 2.1.5.14 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    int i;
    float value;
    int id;
    int type;
    int install;

    DescTran coord;
    DescPose t_coord, etcp, etool, w_coord;
    memset(&coord, 0, sizeof(DescTran));
    memset(&t_coord, 0, sizeof(DescPose));
    memset(&etcp, 0, sizeof(DescPose));
    memset(&etool, 0, sizeof(DescPose));
    memset(&w_coord, 0, sizeof(DescPose));

    robot.SetSpeed(20);

    for(i = 1; i < 21; i++)
    {
        robot.SetSysVarValue(i, i+0.5);
        robot.WaitMs(1000);
    }

    for(i = 1; i < 21; i++)
    {
        robot.GetSysVarValue(i, &value);
        printf("sys value:%f\n", value);
    }

    robot.SetLoadWeight(2.5);

    coord.x = 3.0;
    coord.y = 4.0;
    coord.z = 5.0;

    robot.SetLoadCoord(&coord);
```

(continues on next page)

(continued from previous page)

```

    id = 10;
    t_coord.tran.x = 1.0;
    t_coord.tran.y = 2.0;
    t_coord.tran.z = 3.0;
    t_coord.rpy.rx = 4.0;
    t_coord.rpy.ry = 5.0;
    t_coord.rpy.rz = 6.0;
    type = 0;
    install = 0;
    robot.SetToolCoord(id, &t_coord, type, install);
    robot.SetToolList(id, &t_coord, type, install);

    etcp.tran.x = 1.0;
    etcp.tran.y = 2.0;
    etcp.tran.z = 3.0;
    etcp.rpy.rx = 4.0;
    etcp.rpy.ry = 5.0;
    etcp.rpy.rz = 6.0;
    etool.tran.x = 11.0;
    etool.tran.y = 22.0;
    etool.tran.z = 33.0;
    etool.rpy.rx = 44.0;
    etool.rpy.ry = 55.0;
    etool.rpy.rz = 66.0;
    id = 11;
    robot.SetExToolCoord(id, &etcp, &etool);
    robot.SetExToolList(id, &etcp, &etool);

    w_coord.tran.x = 11.0;
    w_coord.tran.y = 12.0;
    w_coord.tran.z = 13.0;
    w_coord.rpy.rx = 14.0;
    w_coord.rpy.ry = 15.0;
    w_coord.rpy.rz = 16.0;
    id = 12;
    robot.SetWObjCoord(id, &w_coord);
    robot.SetWObjList(id, &w_coord);

    robot.SetRobotInstallPos(0);
    robot.SetRobotInstallAngle(15.0,25.0);

    return 0;
}
```

## 2.1.6 Security settings

### 2.1.6.1 Set collision level

```
/**
* @brief Set collision level
* @param  [in]  mode  0- grade, 1- percentage
* @param  [in]  level Collision threshold, grade range [], percentage range [0~1]
* @param  [in]  config 0- Do not update the configuration file. 1- Update the configuration file
* @return  Error code
*/
errno_t  SetAnticollision(int mode, float level[6], int config);
```

### 2.1.6.2 Set the post-collision policy

```
/**
* @brief  Set the post-collision policy
* @param  [in] strategy  0- Error stop, 1- Continue running
* @return  Error code
*/
errno_t  SetCollisionStrategy(int strategy);
```

### 2.1.6.3 Set the positive limit

```
/**
* @brief  Set the positive limit
* @param  [in] limit Six joint positions, unit: deg
* @return  Error code
*/
errno_t  SetLimitPositive(float limit[6]);
```

### 2.1.6.4 Set the negative limit

```
/**
* @brief  Set the negative limit
* @param  [in] limit Six joint positions, unit: deg
* @return  Error code
*/
errno_t  SetLimitNegative(float limit[6]);
```

### 2.1.6.5 Error status clearing

```
/**
* @brief  Error status clearing
* @return  Error code
*/
errno_t  ResetAllError();
```

### 2.1.6.6 Joint friction compensation switch

```
/**
* @brief  Joint friction compensation switch
* @param  [in]  state  0- off, 1- on
* @return  Error code
*/
errno_t  FrictionCompensationOnOff(uint8_t state);
```

### 2.1.6.7 Set joint friction compensation coefficient - formal

```
/**
* @brief  Set joint friction compensation coefficient - formal
* @param  [in]  coeff Six joint compensation coefficients, range [0~1]
* @return  Error code
*/
errno_t  SetFrictionValue_level(float coeff[6]);
```

### 2.1.6.8 Set joint friction compensation coefficient - side mount

```
/**
* @brief  Set joint friction compensation coefficient - side mount
* @param  [in]  coeff Six joint compensation coefficients, range [0~1]
* @return  Error code
*/
errno_t  SetFrictionValue_wall(float coeff[6]);
```

### 2.1.6.9 Set joint friction compensation coefficient - reverse mount

```
/**
* @brief  Set joint friction compensation coefficient - inverted
* @param  [in]  coeff Six joint compensation coefficients, range [0~1]
* @return  Error code
*/
errno_t  SetFrictionValue_ceiling(float coeff[6]);
```

### 2.1.6.10 Set joint friction compensation coefficient - free mount

```
/**
* @brief  Set joint friction compensation coefficient - free mount
* @param  [in]  coeff Six joint compensation coefficients, range [0~1]
* @return  Error code
*/
errno_t  SetFrictionValue_freedom(float coeff[6]);
```

### 2.1.6.11 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    int mode = 0;
    int config = 1;
    float level1[6] = {1.0,2.0,3.0,4.0,5.0,6.0};
    float level2[6] = {50.0,20.0,30.0,40.0,50.0,60.0};

    robot.SetAnticollision(mode, level1, config);
    mode = 1;
    robot.SetAnticollision(mode, level2, config);
    robot.SetCollisionStrategy(1);

    float plimit[6] = {170.0,80.0,150.0,80.0,170.0,160.0};
    robot.SetLimitPositive(plimit);
    float nlimit[6] = {-170.0,-260.0,-150.0,-260.0,-170.0,-160.0};
    robot.SetLimitNegative(nlimit);

    robot.ResetAllError();

    float lcoeff[6] = {0.9,0.9,0.9,0.9,0.9,0.9};
    float wcoeff[6] = {0.4,0.4,0.4,0.4,0.4,0.4};
    float ccoeff[6] = {0.6,0.6,0.6,0.6,0.6,0.6};
    float fcoeff[6] = {0.5,0.5,0.5,0.5,0.5,0.5};
    robot.FrictionCompensationOnOff(1);
    robot.SetFrictionValue_level(lcoeff);
    robot.SetFrictionValue_wall(wcoeff);
```

(continues on next page)

(continued from previous page)

```
    robot.SetFrictionValue_ceiling(ccoeff);
    robot.SetFrictionValue_freedom(fcoeff);

    return 0;
}
```

## 2.1.7 Status query

### 2.1.7.1 Obtain robot mounting Angle

```
/**
* @brief  Obtain robot mounting Angle
* @param  [out] yangle Angle of inclination
* @param  [out] zangle Angle of rotation
* @return  Error code
*/
errno_t  GetRobotInstallAngle(float *yangle, float *zangle);
```

### 2.1.7.2 Get the system variable value

```
/**
* @brief  Get the system variable value
* @param  [in] id System variable number, range[1~20]
* @param  [out] value  System variable value
* @return  Error code
*/
errno_t  GetSysVarValue(int id, float *value);
```

### 2.1.7.3 Get the current joint position (Angle)

```
/**
* @brief  Get the current joint position (Angle)
* @param  [in] flag 0- blocking, 1- non-blocking
* @param  [out] jPos Six joint positions, unit: deg
* @return  Error code
*/
errno_t  GetActualJointPosDegree(uint8_t flag, JointPos *jPos);
```

### 2.1.7.4 Get the current joint position (radians)

```
/**
* @brief  Get the current joint position (radians)
* @param  [in] flag 0- blocking, 1- non-blocking
* @param  [out] jPos Six joint positions, unit: rad
* @return  Error code
*/
errno_t  GetActualJointPosRadian(uint8_t flag, JointPos *jPos);
```

### 2.1.7.5 Get the current tool pose

```
/**
* @brief  Get the current tool pose
* @param  [in] flag  0- blocking, 1- non-blocking
* @param  [out] desc_pos  Tool position
* @return  Error code
*/
errno_t  GetActualTCPPose(uint8_t flag, DescPose *desc_pos);
```

### 2.1.7.6 Get the current tool coordinate system number

```
/**
* @brief  Get the current tool coordinate system number
* @param  [in] flag  0- blocking, 1- non-blocking
* @param  [out] id  Tool coordinate system number
* @return  Error code
*/
errno_t  GetActualTCPNum(uint8_t flag, int *id);
```

### 2.1.7.7 Get the current workpiece coordinate system number

```
/**
* @brief  Get the current workpiece coordinate system number
* @param  [in] flag  0- blocking, 1- non-blocking
* @param  [out] id  Job coordinate system number
* @return  Error code
*/
errno_t  GetActualWObjNum(uint8_t flag, int *id);
```

#### 2.1.7.8 Get the current end flange pose

```
/**
* @brief  Get the current end flange pose
* @param  [in] flag  0- blocking, 1- non-blocking
* @param  [out] desc_pos  Flange pose
* @return  Error code
*/
errno_t  GetActualToolFlangePose(uint8_t flag, DescPose *desc_pos);
```

#### 2.1.7.9 Inverse kinematics solution

```
/**
* @brief  Inverse kinematics solution
* @param  [in] type 0- absolute pose (base frame), 1- incremental pose (base frame), 2-␣
↪incremental pose (tool frame)
* @param  [in] desc_pos Cartesian pose
* @param  [in] config Joint space configuration, [-1]- based on the current joint␣
↪position, [0~7]- based on the specific joint space configuration
* @param  [out] joint_pos Joint position
* @return  Error code
*/
errno_t  GetInverseKin(int type, DescPose *desc_pos, int config, JointPos *joint_pos);
```

#### 2.1.7.10 Inverse kinematics solution

```
/**
* @brief  Inverse kinematics is solved by referring to the specified joint position
* @param  [in] type 0- absolute pose (base frame), 1- incremental pose (base frame), 2-␣
↪incremental pose (tool frame)
* @param  [in] desc_pos Cartesian pose
* @param  [in] joint_pos_ref Reference joint position
* @param  [out] joint_pos Joint position
* @return  Error code
*/
errno_t  GetInverseKinRef(int type, DescPose *desc_pos, JointPos *joint_pos_ref,␣
↪JointPos *joint_pos);
```

#### 2.1.7.11 Inverse kinematics solution

```
/**
* @brief  To solve the inverse kinematics, refer to the specified joint position to␣
↪determine whether there is a solution
* @param  [in] type 0- absolute pose (base frame), 1- incremental pose (base frame), 2-␣
↪incremental pose (tool frame)
* @param  [in] desc_pos Cartesian pose
* @param  [in] joint_pos_ref Reference joint position
* @param  [out] result 0- no solution, 1-solution
```

(continues on next page)

(continued from previous page)

```
* @return  Error code
*/
errno_t  GetInverseKinHasSolution(int type, DescPose *desc_pos, JointPos *joint_pos_ref,␣
↪uint8_t *result);
```

### 2.1.7.12 Forward kinematics solution

```
/**
* @brief  Forward kinematics solution
* @param  [in] joint_pos Joint position
* @param  [out] desc_pos Cartesian pose
* @return  Error code
*/
errno_t  GetForwardKin(JointPos *joint_pos, DescPose *desc_pos);
```

### 2.1.7.13 Obtain the current joint torque

```
/**
* @brief Obtain the current joint torque
* @param  [in] flag 0- blocking, 1- non-blocking
* @param  [out] torques Joint torque
* @return  Error code
*/
errno_t  GetJointTorques(uint8_t flag, float torques[6]);
```

### 2.1.7.14 Get the weight of the current load

```
/**
* @brief  Gets the weight of the current load
* @param  [in] flag 0- blocking, 1- non-blocking
* @param  [out] weight Load weight, unit: kg
* @return  Error code
*/
errno_t  GetTargetPayload(uint8_t flag, float *weight);
```

### 2.1.7.15 Get the center of mass of the current load

```
/**
* @brief  Get the center of mass of the current load
* @param  [in] flag 0- blocking, 1- non-blocking
* @param  [out] cog Load center of mass, unit: mm
* @return  Error code
*/
errno_t  GetTargetPayloadCog(uint8_t flag, DescTran *cog);
```

### 2.1.7.16 Get the current tool coordinate system

```
/**
* @brief  Get the current tool coordinate system
* @param  [in] flag 0- blocking, 1- non-blocking
* @param  [out] desc_pos Tool coordinate position
* @return  Error code
*/
errno_t  GetTCPOffset(uint8_t flag, DescPose *desc_pos);
```

### 2.1.7.17 Get the current work frame

```
/**
* @brief  Get the current work frame
* @param  [in] flag 0- blocking, 1- non-blocking
* @param  [out] desc_pos Position of workpiece coordinate system
* @return  Error code
*/
errno_t  GetWObjOffset(uint8_t flag, DescPose *desc_pos);
```

### 2.1.7.18 Obtain joint soft limit Angle

```
/**
* @brief  Obtain joint soft limit Angle
* @param  [in] flag 0- blocking, 1- non-blocking
* @param  [out] negative  Negative limit Angle, unit: deg
* @param  [out] positive  Positive limit Angle, unit: deg
* @return  Error code
*/
errno_t  GetJointSoftLimitDeg(uint8_t flag, float negative[6], float positive[6]);
```

### 2.1.7.19 Get system time

```
/**
* @brief  Get system time
* @param  [out] t_ms unit: ms
* @return  Error code
*/
errno_t  GetSystemClock(float *t_ms);
```

### 2.1.7.20 Get the current joint configuration of the robot

```
/**
* @brief  Get the current joint configuration of the robot
* @param  [out]  config  Joint space configuration, range [0~7]
* @return  Error code
*/
errno_t  GetRobotCurJointsConfig(int *config);
```

### 2.1.7.21 Get current speed

```
/**
* @brief  Get the robot's current speed
* @param  [out]  vel  The unit is mm/s
* @return  Error code
*/
errno_t  GetDefaultTransVel(float *vel);
```

### 2.1.7.22 Query whether the robot movement is complete

```
/**
* @brief  Query whether the robot movement is complete
* @param  [out]  state  0- Incomplete, 1- completed
* @return  Error code
*/
errno_t  GetRobotMotionDone(uint8_t *state);
```

### 2.1.7.23 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    float yangle, zangle;
    int flag = 0;
    JointPos j_deg, j_rad;
```

(continues on next page)

(continued from previous page)

```
    DescPose tcp, flange, tcp_offset, wobj_offset;
    DescTran cog;
    int id;
    float torques[6] = {0.0};
    float weight;
    float neg_deg[6]={0.0},pos_deg[6]={0.0};
    float t_ms;
    int config;
    float vel;

    memset(&j_deg, 0, sizeof(JointPos));
    memset(&j_rad, 0, sizeof(JointPos));
    memset(&tcp, 0, sizeof(DescPose));
    memset(&flange, 0, sizeof(DescPose));
    memset(&tcp_offset, 0, sizeof(DescPose));
    memset(&wobj_offset, 0, sizeof(DescPose));
    memset(&cog, 0, sizeof(DescTran));

    robot.GetRobotInstallAngle(&yangle, &zangle);
    printf("yangle:%f,zangle:%f\n", yangle, zangle);

    robot.GetActualJointPosDegree(flag, &j_deg);
    printf("joint pos deg:%f,%f,%f,%f,%f,%f\n", j_deg.jPos[0],j_deg.jPos[1],j_deg.jPos[2],j_deg.jPos[3],j_deg.jPos[4],j_deg.jPos[5]);

    robot.GetActualJointPosRadian(flag, &j_rad);
    printf("joint pos rad:%f,%f,%f,%f,%f,%f\n", j_rad.jPos[0],j_rad.jPos[1],j_rad.jPos[2],j_rad.jPos[3],j_rad.jPos[4],j_rad.jPos[5]);

    robot.GetActualTCPPose(flag, &tcp);
    printf("tcp pose:%f,%f,%f,%f,%f,%f\n", tcp.tran.x, tcp.tran.y, tcp.tran.z, tcp.rpy.rx, tcp.rpy.ry, tcp.rpy.rz);

    robot.GetActualToolFlangePose(flag, &flange);
    printf("flange pose:%f,%f,%f,%f,%f,%f\n", flange.tran.x, flange.tran.y, flange.tran.z, flange.rpy.rx, flange.rpy.ry, flange.rpy.rz);

    robot.GetActualTCPNum(flag, &id);
    printf("tcp num:%d\n", id);

    robot.GetActualWObjNum(flag, &id);
    printf("wobj num:%d\n", id);

    robot.GetJointTorques(flag, torques);
    printf("torques:%f,%f,%f,%f,%f,%f\n", torques[0],torques[1],torques[2],torques[3],torques[4],torques[5]);

    robot.GetTargetPayload(flag, &weight);
    printf("payload weight:%f\n", weight);

    robot.GetTargetPayloadCog(flag, &cog);
    printf("payload cog:%f,%f,%f\n",cog.x, cog.y, cog.z);
```

(continues on next page)

(continued from previous page)

```
    robot.GetTCPOffset(flag, &tcp_offset);
    printf("tcp offset:%f,%f,%f,%f,%f,%f\n", tcp_offset.tran.x,tcp_offset.tran.y,tcp_offset.tran.z,tcp_offset.rpy.rx,tcp_offset.rpy.ry,tcp_offset.rpy.rz);

    robot.GetWObjOffset(flag, &wobj_offset);
    printf("wobj offset:%f,%f,%f,%f,%f,%f\n", wobj_offset.tran.x,wobj_offset.tran.y,wobj_offset.tran.z,wobj_offset.rpy.rx,wobj_offset.rpy.ry,wobj_offset.rpy.rz);

    robot.GetJointSoftLimitDeg(flag, neg_deg, pos_deg);
    printf("neg limit deg:%f,%f,%f,%f,%f,%f\n",neg_deg[0],neg_deg[1],neg_deg[2],neg_deg[3],neg_deg[4],neg_deg[5]);
    printf("pos limit deg:%f,%f,%f,%f,%f,%f\n",pos_deg[0],pos_deg[1],pos_deg[2],pos_deg[3],pos_deg[4],pos_deg[5]);

    robot.GetSystemClock(&t_ms);
    printf("system clock:%f\n", t_ms);

    robot.GetRobotCurJointsConfig(&config);
    printf("joint config:%d\n", config);

    robot.GetDefaultTransVel(&vel);
    printf("trans vel:%f\n", vel);

    return 0;
}
```

### 2.1.8 Trajectory recurrence

#### 2.1.8.1 Set track recording parameters

```
/**
* @brief  Set track recording parameters
* @param  [in] type  Record data type, 1- joint position
* @param  [in] name  Track file name
* @param  [in] period_ms  Data sampling period, fixed value 2ms or 4ms or 8ms
* @param  [in] di_choose  DI Select,bit0 to bit7 corresponds to control box DI0 to DI7, bit8 to bit9 corresponds to end DI0 to DI1, 0- do not select, 1- select
* @param  [in] do_choose  DO select,bit0~bit7 corresponds to control box DO0~DO7, bit8~bit9 corresponds to end DO0~DO1, 0- do not select, 1- select
* @return  Error code
*/
errno_t  SetTPDParam(int type, char name[30], int period_ms, uint16_t di_choose, uint16_t do_choose);
```

### 2.1.8.2 Start track recording

```
/**
* @brief  Start track recording
* @param  [in] type  Record data type, 1- joint position
* @param  [in] name  Track file name
* @param  [in] period_ms  Data sampling period, fixed value 2ms or 4ms or 8ms
* @param  [in] di_choose  DI Select,bit0 to bit7 corresponds to control box DI0 to DI7, bit8 to bit9 corresponds to end DI0 to DI1, 0- do not select, 1- select
* @param  [in] do_choose  DO select,bit0~bit7 corresponds to control box DO0~DO7, bit8~bit9 corresponds to end DO0~DO1, 0- do not select, 1- select
* @return  Error code
*/
errno_t  SetTPDStart(int type, char name[30], int period_ms, uint16_t di_choose, uint16_t do_choose);
```

### 2.1.8.3 Stop track recording

```
/**
* @brief  Stop track recording
* @return  Error code
*/
errno_t  SetWebTPDStop();
```

### 2.1.8.4 Delete track record

```
/**
* @brief  Delete track record
* @param  [in] name  Track file name
* @return  Error code
*/
errno_t  SetTPDDelete(char name[30]);
```

### 2.1.8.5 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                //Instantiate the robot object
```

(continues on next page)

(continued from previous page)

```
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot␣
↪controller

    int type = 1;
    char name[30] = "tpd2023";
    int period_ms = 4;
    uint16_t di_choose = 0;
    uint16_t do_choose = 0;

    robot.SetTPDParam(type, name, period_ms, di_choose, do_choose);

    robot.Mode(1);
    sleep(1);
    robot.DragTeachSwitch(1);
    robot.SetTPDStart(type, name, period_ms, di_choose, do_choose);
    sleep(30);
    robot.SetWebTPDStop();
    robot.DragTeachSwitch(0);

    //robot.SetTPDDelete(name);

    return 0;
}
```

### 2.1.8.6 Trajectory preloading

```
/**
* @brief  Trajectory preloading
* @param  [in] name  Track file name
* @return  Error code
*/
errno_t  LoadTPD(char name[30]);
```

### 2.1.8.7 Trajectory recurrence

```
/**
* @brief  Trajectory recurrence
* @param  [in] name  Track file name
* @param  [in] blend 0- not smooth, 1- smooth
* @param  [in] ovl  Speed scaling percentage, range [0~100]
* @return  Error code
*/
errno_t  MoveTPD(char name[30], uint8_t blend, float ovl);
```

#### 2.1.8.8 Code example

```cpp
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    char name[30] = "tpd2023";
    int tool = 1;
    int user = 0;
    float vel = 100.0;
    float acc = 100.0;
    float ovl = 100.0;
    float blendT = -1.0;
    int config = -1;
    uint8_t blend = 1;

    DescPose desc_pose;
    memset(&desc_pose, 0, sizeof(DescPose));

    desc_pose.tran.x = -378.9;
    desc_pose.tran.y = -340.3;
    desc_pose.tran.z = 107.2;
    desc_pose.rpy.rx = 179.4;
    desc_pose.rpy.ry = -1.3;
    desc_pose.rpy.rz = 125.0;

    robot.LoadTPD(name);
    robot.MoveCart(&desc_pose, tool, user, vel, acc, ovl, blendT, config);
    robot.MoveTPD(name, blend, ovl);

    return 0;
}
```

### 2.1.9 WebAPP program use

#### 2.1.9.1 Set the default job program to be automatically loaded upon startup

```
/**
* @brief  Set the default job program to be automatically loaded upon startup
* @param  [in] flag  0- boot does not automatically load the default program, 1- boot automatically load the default program
* @param  [in] program_name Job program name and path, for example, /erauser/movej.lua, where /erauser/ is a fixed path* @return  Error code
*/

errno_t  LoadDefaultProgConfig(uint8_t flag, char program_name[64]);
```

#### 2.1.9.2 Load the specified job program

```
/**
* @brief  Load the specified job program
* @param  [in] program_name Job program name and path, for example, /erauser/movej.lua, where /erauser/ is a fixed path* @return  Error code
*/

errno_t  ProgramLoad(char program_name[64]);
```

#### 2.1.9.3 Get the loaded job program name

```
/**
* @brief  Get the loaded job program name
* @param  [out] program_name Job program name and path, for example, /erauser/movej.lua, where /erauser/ is a fixed path* @return  Error code
*/

errno_t  GetLoadedProgram(char program_name[64]);
```

#### 2.1.9.4 Get the line number of the current robot job program

```
/**
* @brief  Get the line number of the current robot job program
* @param  [out] line  line number
* @return  Error code
*/
errno_t  GetCurrentLine(int *line);
```

#### 2.1.9.5 Run the currently loaded job program

```
/**
* @brief  Run the currently loaded job program
* @return  Error code
*/
errno_t  ProgramRun();
```

#### 2.1.9.6 Pause the current running job program

```
/**
* @brief  Pause the current running job program
* @return  Error code
*/
errno_t  ProgramPause();
```

#### 2.1.9.7 Resume the currently suspended job program

```
/**
* @brief  Resume the currently suspended job program
* @return  Error code
*/
errno_t  ProgramResume();
```

#### 2.1.9.8 Terminates the currently running job program

```
/**
* @brief  Terminates the currently running job program
* @return  Error code
*/
errno_t  ProgramStop();
```

#### 2.1.9.9 Get the robot job program execution state

```
/**
* @brief  Get the robot job program execution state
* @param  [out]  state 1- program stop or no program running, 2- program running, 3- program pause
* @return  Error code
*/
errno_t  GetProgramState(uint8_t *state);
```

### 2.1.9.10 Code example

```cpp
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    char program_name[64] = "/erauser/ptps.lua"
    ;char loaded_name[64] = "";
    uint8_t state;
    int line;

    robot.Mode(0);
    robot.ProgramLoad(program_name);
    robot.ProgramRun();
    sleep(5);
    robot.ProgramPause();
    robot.GetProgramState(&state);
    printf("program state:%u\n", state);
    robot.GetCurrentLine(&line);
    printf("current line:%d\n", line);
    robot.GetLoadedProgram(loaded_name);
    printf("program name:%s\n", loaded_name);
    sleep(5);
    robot.ProgramResume();
    sleep(5);
    robot.ProgramStop();
    sleep(2);

    return 0;
}
```

### 2.1.10 Peripheral

#### 2.1.10.1 Configure the gripper

```
/**
* @brief  Configure the gripper
* @param  [in] company  Claw manufacturer, to be determined
* @param  [in] device  Device number, not used yet. The default value is 0
* @param  [in] softvesion  Software version. The value is not used. The default value is 0
* @param  [in] bus The device is attached to the terminal bus and is not in use. The default value is 0
* @return  Error code
*/
errno_t  SetGripperConfig(int company, int device, int softvesion, int bus);
```

#### 2.1.10.2 Obtain the gripper configuration

```
/**
* @brief  Obtain the gripper configuration
* @param  [in] company  Claw manufacturer, to be determined
* @param  [in] device  Device number, not used yet. The default value is 0
* @param  [in] softvesion  Software version. The value is not used. The default value is 0
* @param  [in] bus The device is attached to the terminal bus and is not in use. The default value is 0
* @return  Error code
*/
errno_t  GetGripperConfig(int *company, int *device, int *softvesion, int *bus);
```

#### 2.1.10.3 Activate gripper

```
/**
* @brief  Activate Activate gripper
* @param  [in] index  gripper gripper
* @param  [in] act  0- reset, 1- activate
* @return  Error code
*/
errno_t  ActGripper(int index, uint8_t act);
```

### 2.1.10.4 Control gripper

```
/**
* @brief  Control gripper
* @param  [in] index  gripper number
* @param  [in] pos  Percentage of position, range[0~100]
* @param  [in] vel  Percentage of velocity, range[0~100]
* @param  [in] force  Percentage of torque, range[0~100]
* @param  [in] max_time  Maximum wait time, range[0~30000], unit: ms
* @param  [in] block  0- blocking, 1- non-blocking
* @return  Error code
*/
errno_t  MoveGripper(int index, int pos, int vel, int force, int max_time, uint8_t block);
```

### 2.1.10.5 Obtain the gripper motion state

```
/**
* @brief  Obtain the gripper motion state
* @param  [out] fault  0- no error, 1- error
* @param  [out] staus  0- motion incomplete, 1- motion complete
* @return  Error code
*/
errno_t  GetGripperMotionDone(uint8_t *fault, uint8_t *status);
```

### 2.1.10.6 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    int company = 4;
    int device = 0;
    int softversion = 0;
    int bus = 1;
    int index = 1;
    int act = 0;
    int max_time = 30000;
```

(continues on next page)

(continued from previous page)

```
    uint8_t block = 0;
    uint8_t status, fault;

    robot.SetGripperConfig(company, device, softversion, bus);
    sleep(1);
    robot.GetGripperConfig(&company, &device, &softversion, &bus);
    printf("gripper config:%d,%d,%d,%d\n", company, device, softversion, bus);

    robot.ActGripper(index, act);
    sleep(1);
    act = 1;
    robot.ActGripper(index, act);
    sleep(2);

    robot.MoveGripper(index, 100, 50, 50, max_time, block);
    sleep(3);
    robot.MoveGripper(index, 0, 50, 0, max_time, block);

    robot.GetGripperMotionDone(&fault, &status);
    printf("motion status:%u,%u\n", fault, status);

    return 0;
}
```

### 2.1.11 Force control

#### 2.1.11.1 Force sensor configuration

```
/**
* @brief  Configured force sensor
* @param  [in] company  Manufacturer of force sensors, 17-Kunwei Technology
* @param  [in] device  Device number, not used yet. The default value is 0
* @param  [in] softvesion  Software version. The value is not used. The default value is␣
↪0
* @param  [in] bus The device is attached to the terminal bus and is not in use. The␣
↪default value is 0
* @return  Error code
*/
errno_t  FT_SetConfig(int company, int device, int softvesion, int bus);
```

#### 2.1.11.2 Get the force sensor configuration

```
/**
* @brief  Get the force sensor configuration
* @param  [in] company  Force sensor manufacturer, to be determined
* @param  [in] device  Device number, not used yet. The default value is 0
* @param  [in] softvesion  Software version. The value is not used. The default value is␣
↪0
* @param  [in] bus The device is attached to the terminal bus and is not in use. The␣
```

(continues on next page)

(continued from previous page)

```
↪default value is 0
* @return  Error code
*/
errno_t  FT_GetConfig(int *company, int *device, int *softvesion, int *bus);
```

### 2.1.11.3 Force sensor activation

```
/**
* @brief  Force sensor activation
* @param  [in] act  0- reset, 1- activate
* @return  Error code
*/
errno_t  FT_Activate(uint8_t act);
```

### 2.1.11.4 Force sensor calibration

```
/**
* @brief  Force sensor calibration
* @param  [in] act  0- zero removal, 1- zero correction
* @return  Error code
*/
errno_t  FT_SetZero(uint8_t act);
```

### 2.1.11.5 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>

#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot␣
↪controller

    int company = 17;
    int device = 0;
    int softversion = 0;
    int bus = 1;
    int index = 1;
```

(continues on next page)

(continued from previous page)

```
    int act = 0;

    robot.FT_SetConfig(company, device, softversion, bus);
    sleep(1);
    robot.FT_GetConfig(&company, &device, &softversion, &bus);
    printf("FT config:%d,%d,%d,%d\n", company, device, softversion, bus);
    sleep(1);

    robot.FT_Activate(act);
    sleep(1);
    act = 1;
    robot.FT_Activate(act);
    sleep(1);

    robot.SetLoadWeight(0.0);
    sleep(1);
    DescTran coord;
    memset(&coord, 0, sizeof(DescTran));
    robot.SetLoadCoord(&coord);
    sleep(1);
    robot.FT_SetZero(0);
    sleep(1);

    ForceTorque ft;
    memset(&ft, 0, sizeof(ForceTorque));
    robot.FT_GetForceTorqueOrigin(&ft);
    printf("ft origin:%f,%f,%f,%f,%f,%f\n", ft.fx,ft.fy,ft.fz,ft.tx,ft.ty,ft.tz);
    robot.FT_SetZero(1);
    sleep(1);
    memset(&ft, 0, sizeof(ForceTorque));
    printf("ft rcs:%f,%f,%f,%f,%f,%f\n",ft.fx,ft.fy,ft.fz,ft.tx,ft.ty,ft.tz);

    return 0;
}
```

#### 2.1.11.6 Set the reference coordinate system of the force sensor

```
/**
* @brief  Set the reference coordinate system of the force sensor
* @param  [in] ref  0- tool frame, 1- base frame
* @return  Error code
*/
errno_t  FT_SetRCS(uint8_t ref);
```

#### 2.1.11.7 Load weight identification record

```
/**
* @brief  Load weight identification record
* @param  [in] id  Sensor coordinate system number, range [1~14]
* @return  Error code
*/
errno_t  FT_PdIdenRecord(int id);
```

#### 2.1.11.8 Load weight identification calculation

```
/**
* @brief  Load weight identification calculation
* @param  [out] weight  Load weight, unit: kg
* @return  Error code
*/
errno_t  FT_PdIdenCompute(float *weight);
```

#### 2.1.11.9 Load centroid identification record

```
/**
* @brief  Load centroid identification record
* @param  [in] id  Sensor coordinate system number, range [1~14]
* @param  [in] index Point number, range [1~3]
* @return  Error code
*/
errno_t  FT_PdCogIdenRecord(int id, int index);
```

#### 2.1.11.10 Load centroid identification calculation

```
/**
* @brief  Load centroid identification calculation
* @param  [out] cog  Load center of mass, unit: mm
* @return  Error code
*/
errno_t  FT_PdCogIdenCompute(DescTran *cog);
```

#### 2.1.11.11 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

```

(continues on next page)

(continued from previous page)

```cpp
using namespace std;

int main(void)
{
    ERARobot robot;                //Instantiate the robot object
    robot.RPC("192.168.58.2");     //Establish a communication connection with the robot controller

    float weight;

    DescPose tcoord, desc_p1, desc_p2, desc_p3;
    memset(&tcoord, 0, sizeof(DescPose));
    memset(&desc_p1, 0, sizeof(DescPose));
    memset(&desc_p2, 0, sizeof(DescPose));
    memset(&desc_p3, 0, sizeof(DescPose));

    robot.FT_SetRCS(0);
    sleep(1);

    tcoord.tran.z = 35.0;
    robot.SetToolCoord(10, &tcoord, 1, 0);
    sleep(1);
    robot.FT_PdIdenRecord(10);
    sleep(1);
    robot.FT_PdIdenCompute(&weight);
    printf("payload weight:%f\n", weight);

    desc_p1.tran.x = -160.619;
    desc_p1.tran.y = -586.138;
    desc_p1.tran.z = 384.988;
    desc_p1.rpy.rx = -170.166;
    desc_p1.rpy.ry = -44.782;
    desc_p1.rpy.rz = 169.295;

    desc_p2.tran.x = -87.615;
    desc_p2.tran.y = -606.209;
    desc_p2.tran.z = 556.119;
    desc_p2.rpy.rx = -102.495;
    desc_p2.rpy.ry = 10.118;
    desc_p2.rpy.rz = 178.985;

    desc_p3.tran.x = 41.479;
    desc_p3.tran.y = -557.243;
    desc_p3.tran.z = 484.407;
    desc_p3.rpy.rx = -125.174;
    desc_p3.rpy.ry = 46.995;
    desc_p3.rpy.rz = -132.165;

    robot.MoveCart(&desc_p1, 9, 0, 100.0, 100.0, 100.0, -1.0, -1);
    sleep(1);
    robot.FT_PdCogIdenRecord(10, 1);
    robot.MoveCart(&desc_p2, 9, 0, 100.0, 100.0, 100.0, -1.0, -1);
```

(continues on next page)

(continued from previous page)

```
    sleep(1);
    robot.FT_PdCogIdenRecord(10, 2);
    robot.MoveCart(&desc_p3, 9, 0, 100.0, 100.0, 100.0, -1.0, -1);
    sleep(1);
    robot.FT_PdCogIdenRecord(10, 3);
    sleep(1);
    DescTran cog;
    memset(&cog, 0, sizeof(DescTran));
    robot.FT_PdCogIdenCompute(&cog);
    printf("cog:%f,%f,%f\n",cog.x, cog.y, cog.z);

    return 0;
}
```

#### 2.1.11.12 Obtain force/torque data in the reference coordinate system

```
/**
* @brief  Obtain force/torque data in the reference coordinate system
* @param  [out] ft  Force/torquefx,fy,fz,tx,ty,tz
* @return  Error code
*/
errno_t  FT_GetForceTorqueRCS(ForceTorque *ft);
```

#### 2.1.11.13 Obtain the raw force/torque data of the force sensor

```
/**
* @brief  Obtain the raw force/torque data of the force sensor
* @param  [out] ft  Force/torquefx,fy,fz,tx,ty,tz
* @return  Error code
*/
errno_t  FT_GetForceTorqueOrigin(ForceTorque *ft);
```

#### 2.1.11.14 Collision guard

```
/**
* @brief  Collision guard
* @param  [in] flag 0- Disable collision guard. 1- Enable collision guard
* @param  [in] sensor_id Force sensor number
* @param  [in] select  Select the six degrees of freedom whether to detect collision, 0- no detection, 1- detection
* @param  [in] ft  Impact force/torquefx,fy,fz,tx,ty,tz
* @param  [in] max_threshold Maximum threshold
* @param  [in] min_threshold Minimum threshold
* @note   Force/torque detection range(ft-min_threshold, ft+max_threshold)
* @return  Error code
*/
errno_t  FT_Guard(uint8_t flag, int sensor_id, uint8_t select[6], ForceTorque *ft, float max_threshold[6], float min_threshold[6]);
```

### 2.1.11.15 Code Example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    uint8_t flag = 1;
    uint8_t sensor_id = 1;
    uint8_t select[6] = {1,1,1,1,1,1};
    float max_threshold[6] = {10.0,10.0,10.0,10.0,10.0,10.0};
    float min_threshold[6] = {5.0,5.0,5.0,5.0,5.0,5.0};

    ForceTorque ft;
    DescPose desc_p1, desc_p2, desc_p3;
    memset(&ft, 0, sizeof(ForceTorque));
    memset(&desc_p1, 0, sizeof(DescPose));
    memset(&desc_p2, 0, sizeof(DescPose));
    memset(&desc_p3, 0, sizeof(DescPose));

    desc_p1.tran.x = -160.619;
    desc_p1.tran.y = -586.138;
    desc_p1.tran.z = 384.988;
    desc_p1.rpy.rx = -170.166;
    desc_p1.rpy.ry = -44.782;
    desc_p1.rpy.rz = 169.295;

    desc_p2.tran.x = -87.615;
    desc_p2.tran.y = -606.209;
    desc_p2.tran.z = 556.119;
    desc_p2.rpy.rx = -102.495;
    desc_p2.rpy.ry = 10.118;
    desc_p2.rpy.rz = 178.985;

    desc_p3.tran.x = 41.479;
    desc_p3.tran.y = -557.243;
    desc_p3.tran.z = 484.407;
    desc_p3.rpy.rx = -125.174;
    desc_p3.rpy.ry = 46.995;
    desc_p3.rpy.rz = -132.165;

    robot.FT_Guard(flag, sensor_id, select, &ft, max_threshold, min_threshold);
```

(continues on next page)

(continued from previous page)

```
    robot.MoveCart(&desc_p1,9,0,100.0,100.0,100.0,-1.0,-1);
    robot.MoveCart(&desc_p2,9,0,100.0,100.0,100.0,-1.0,-1);
    robot.MoveCart(&desc_p3,9,0,100.0,100.0,100.0,-1.0,-1);
    flag = 0;
    robot.FT_Guard(flag, sensor_id, select, &ft, max_threshold, min_threshold);

    return 0;
}
```

#### 2.1.11.16 Constant force control

```
/**
* @brief  Constant force control
* @param  [in] flag 0- turn off constant force control, 1- turn on constant force control
* @param  [in] sensor_id Force sensor number
* @param  [in] select  Select the six degrees of freedom whether to detect collision, 0- no detection, 1- detection
* @param  [in] ft  Impact force/torquefx,fy,fz,tx,ty,tz
* @param  [in] ft_pid Force pid parameter, torque pid parameter
* @param  [in] adj_sign Adaptive start-stop control, 0- off, 1- on
* @param  [in] ILC_sign ILC start stop control, 0- stop, 1- training, 2- operation
* @param  [in] Maximum Adjustment distance, unit: mm
* @param  [in] Maximum Adjustment Angle, unit: deg
* @return  Error code
*/
errno_t  FT_Control(uint8_t flag, int sensor_id, uint8_t select[6], ForceTorque *ft, float ft_pid[6], uint8_t adj_sign, uint8_t ILC_sign, float max_dis, float max_ang);
```

#### 2.1.11.17 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    uint8_t flag = 1;
    uint8_t sensor_id = 1;
    uint8_t select[6] = {0,0,1,0,0,0};
```

(continues on next page)

(continued from previous page)

```
    float ft_pid[6] = {0.0005,0.0,0.0,0.0,0.0,0.0};
    uint8_t adj_sign = 0;
    uint8_t ILC_sign = 0;
    float max_dis = 100.0;
    float max_ang = 0.0;

    ForceTorque ft;
    DescPose desc_p1, desc_p2, offset_pos;
    JointPos j1,j2;
    ExaxisPos epos;
    memset(&ft, 0, sizeof(ForceTorque));
    memset(&desc_p1, 0, sizeof(DescPose));
    memset(&desc_p2, 0, sizeof(DescPose));
    memset(&offset_pos, 0, sizeof(DescPose));
    memset(&epos, 0, sizeof(ExaxisPos));
    memset(&j1, 0, sizeof(JointPos));
    memset(&j2, 0, sizeof(JointPos));

    j1 = {-68.987,-96.414,-111.45,-61.105,92.884,11.089};
    j2 = {-107.596,-109.154,-104.735,-56.176,90.739,11.091};

    desc_p1.tran.x = 62.795;
    desc_p1.tran.y = -511.979;
    desc_p1.tran.z = 291.697;
    desc_p1.rpy.rx = -179.545;
    desc_p1.rpy.ry = 3.027;
    desc_p1.rpy.rz = -170.039;

    desc_p2.tran.x = -294.768;
    desc_p2.tran.y = -503.708;
    desc_p2.tran.z = 233.158;
    desc_p2.rpy.rx = 179.799;
    desc_p2.rpy.ry = 0.713;
    desc_p2.rpy.rz = 151.309;

    ft.fz = -10.0;

    robot.MoveJ(&j1,&desc_p1,9,0,100.0,180.0,100.0,&epos,-1.0,0,&offset_pos);
    robot.FT_Control(flag, sensor_id, select, &ft, ft_pid, adj_sign, ILC_sign, max_dis,␣
↪max_ang);
    robot.MoveL(&j2,&desc_p2,9,0,100.0,180.0,20.0,-1.0,&epos,0,0,&offset_pos);
    flag = 0;
    robot.FT_Control(flag, sensor_id, select, &ft, ft_pid, adj_sign, ILC_sign, max_dis,␣
↪max_ang);

    return 0;
}
```

### 2.1.11.18 Spiral exploration

```cpp
/**
* @brief  Spiral exploration
* @param  [in] rcs Reference frame, 0- tool frame, 1- base frame
* @param  [in] dr Feed per circle radius
* @param  [in] ft Force/torque thresholdfx,fy,fz,tx,ty,tzrange[0~100]
* @param  [in] max_t_ms Maximum exploration time, unit: ms
* @param  [in] max_vel Maximum linear velocity, unit: mm/s
* @return  Error code
*/
errno_t  FT_SpiralSearch(int rcs, float dr, float ft, float max_t_ms, float max_vel);
```

### 2.1.11.19 Rotary insertion

```cpp
/**
* @brief  Rotary insertion
* @param  [in] rcs Reference frame, 0- tool frame, 1- base frame
* @param  [in] angVelRot Angular velocity of rotation, unit: deg/s
* @param  [in] ft  Force/torque thresholdfx,fy,fz,tx,ty,tzrange[0~100]
* @param  [in] max_angle Maximum rotation Angle, unit: deg
* @param  [in] orn Force/torque direction, 1- along the z axis, 2- around the z axis
* @param  [in] max_angAcc Maximum rotational acceleration, in deg/s^2, not used yet, default is 0
* @param  [in] rotorn  Rotation direction, 1- clockwise, 2- counterclockwise
* @return  Error code
*/
errno_t  FT_RotInsertion(int rcs, float angVelRot, float ft, float max_angle, uint8_t orn, float max_angAcc, uint8_t rotorn);
```

### 2.1.11.20 Linear insertion

```cpp
/**
* @brief  Linear insertion
* @param  [in] rcs Reference frame, 0- tool frame, 1- base frame
* @param  [in] ft  Force/torque thresholdfx,fy,fz,tx,ty,tzrange[0~100]
* @param  [in] lin_v Linear velocity, unit: mm/s
* @param  [in] lin_a Linear acceleration, unit: mm/s^2, not used yet
* @param  [in] max_dis Maximum insertion distance, unit: mm
* @param  [in] linorn  Insert direction, 0- negative, 1- positive
* @return  Error code
*/
errno_t  FT_LinInsertion(int rcs, float ft, float lin_v, float lin_a, float max_dis, uint8_t linorn);
```

#### 2.1.11.21 Code example

```cpp
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot controller

    //Constant force parameter
    uint8_t status = 1;  //Constant force control open sign, 0- off, 1- on
    int sensor_num = 1; //Force sensor number
    float gain[6] = {0.0001,0.0,0.0,0.0,0.0,0.0};  //Maximum threshold
    uint8_t adj_sign = 0;  //Adaptive start-stop state, 0- off, 1- on
    uint8_t ILC_sign = 0;  //ILC control start stop state, 0- stop, 1- training, 2- real operation
    float max_dis = 100.0;  //Maximum adjustment distance
    float max_ang = 5.0;  //Maximum adjustment Angle

    ForceTorque ft;
    memset(&ft, 0, sizeof(ForceTorque));

    //Helix explore parameters
    int rcs = 0;  //Reference frame, 0- tool frame, 1- base frame
    float dr = 0.7;  //Radius feed per turn, unit: mm
    float fFinish = 1.0; //Force or torque threshold (0 to 100), unit: N or Nm
    float t = 60000.0; //Maximum exploration time, unit: ms
    float vmax = 3.0; //The maximum linear velocity, unit: mm/s

    //Linear insertion parameter
    float force_goal = 20.0;  //Force or torque threshold (0 to 100), unit: N or Nm
    float lin_v = 0.0; //Linear velocity, unit: mm/s
    float lin_a = 0.0; //Linear acceleration, unit: mm/s^2, not used yet
    float disMax = 100.0; //Maximum insertion distance, in mm
    uint8_t linorn = 1; //Insert direction, 1- positive, 2- negative

    //Rotational insertion parameter
    float angVelRot = 2.0;  //Angular velocity of rotation, in °/s
    float forceInsertion = 1.0; //Force or torque threshold (0 to 100), in N or Nm
    int angleMax= 45; //Maximum rotation Angle, unit: °
    uint8_t orn = 1; //Direction of force1-fz,2-mz
    float angAccmax = 0.0; //Maximum angular acceleration of rotation, unit: °/s^2, not in use
    uint8_t rotorn = 1; //Rotation direction, 1- clockwise, 2- counterclockwise
```

(continues on next page)

(continued from previous page)

```

    uint8_t select1[6] = {0,0,1,1,1,0}; //Six degrees of freedom options [fx,fy,fz,mx,my,mz], 0- does not work, 1- works
    ft.fz = -10.0;
    robot.FT_Control(status,sensor_num,select1,&ft,gain,adj_sign,ILC_sign,max_dis,max_ang);
    robot.FT_SpiralSearch(rcs,dr,fFinish,t,vmax);
    status = 0;
    robot.FT_Control(status,sensor_num,select1,&ft,gain,adj_sign,ILC_sign,max_dis,max_ang);

    uint8_t select2[6] = {1,1,1,0,0,0};  //Six degrees of freedom options [fx,fy,fz,mx,my,mz], 0- does not work, 1- works
    gain[0] = 0.00005;
    ft.fz = -30.0;
    status = 1;
    robot.FT_Control(status,sensor_num,select2,&ft,gain,adj_sign,ILC_sign,max_dis,max_ang);
    robot.FT_LinInsertion(rcs,force_goal,lin_v,lin_a,disMax,linorn);
    status = 0;
    robot.FT_Control(status,sensor_num,select2,&ft,gain,adj_sign,ILC_sign,max_dis,max_ang);

    uint8_t select3[6] = {0,0,1,1,1,0};  //Six degrees of freedom options [fx,fy,fz,mx,my,mz], 0- does not work, 1- works
    ft.fz = -10.0;
    gain[0] = 0.0001;
    status = 1;
    robot.FT_Control(status,sensor_num,select3,&ft,gain,adj_sign,ILC_sign,max_dis,max_ang);
    robot.FT_RotInsertion(rcs,angVelRot,forceInsertion,angleMax,orn,angAccmax,rotorn);
    status = 0;
    robot.FT_Control(status,sensor_num,select3,&ft,gain,adj_sign,ILC_sign,max_dis,max_ang);

    uint8_t select4[6] = {1,1,1,0,0,0};  //Six degrees of freedom options [fx,fy,fz,mx,my,mz], 0- does not work, 1- works
    ft.fz = -30.0;
    status = 1;
    robot.FT_Control(status,sensor_num,select4,&ft,gain,adj_sign,ILC_sign,max_dis,max_ang);
    robot.FT_LinInsertion(rcs,force_goal,lin_v,lin_a,disMax,linorn);
    status = 0;
    robot.FT_Control(status,sensor_num,select4,&ft,gain,adj_sign,ILC_sign,max_dis,max_ang);

    return 0;
}
```

### 2.1.11.22 Surface positioning

```
/**
* @brief  Surface positioning
* @param  [in] rcs Reference frame, 0- tool frame, 1- base frame
* @param  [in] dir  The direction of travel, 1- positive, 2- negative
* @param  [in] axis Axis of movement, 1-x axis, 2-y axis, 3-z axis
* @param  [in] lin_v Explore the linear velocity in mm/s
* @param  [in] lin_a Explore linear acceleration, in mm/s^2, not used yet, default to 0
* @param  [in] max_dis Maximum exploration distance, in mm
* @param  [in] ft  Action termination force/torque thresholdfx,fy,fz,tx,ty,tz
* @return  Error code
*/
errno_t  FT_FindSurface(int rcs, uint8_t dir, uint8_t axis, float lin_v, float lin_a,␣
↪float max_dis, float ft);
```

### 2.1.11.23 Calculation of midplane position starts

```
/**
* @brief  Calculation of midplane position starts
* @return  Error code
*/
errno_t  FT_CalCenterStart();
```

### 2.1.11.24 Calculation of midplane position ends

```
/**
* @brief  Calculation of midplane position ends
* @param  [out] pos Intermediate plane position
* @return  Error code
*/
errno_t  FT_CalCenterEnd(DescPose *pos);
```

### 2.1.11.25 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                //Instantiate the robot object
```

(continues on next page)

(continued from previous page)

```
    robot.RPC("192.168.58.2");     //Establish a communication connection with the robot controller

    int rcs = 0;
    uint8_t dir = 1;
    uint8_t axis = 1;
    float lin_v = 3.0;
    float lin_a = 0.0;
    float maxdis = 50.0;
    float ft_goal = 2.0;

    DescPose desc_pos, xcenter, ycenter;
    ForceTorque ft;
    memset(&desc_pos, 0, sizeof(DescPose));
    memset(&xcenter, 0, sizeof(DescPose));
    memset(&ycenter, 0, sizeof(DescPose));
    memset(&ft, 0, sizeof(ForceTorque));

    desc_pos.tran.x = -230.959;
    desc_pos.tran.y = -364.017;
    desc_pos.tran.z = 217.5;
    desc_pos.rpy.rx = -179.004;
    desc_pos.rpy.ry = 0.002;
    desc_pos.rpy.rz = 89.999;

    ft.fx = -2.0;

    robot.MoveCart(&desc_pos, 9,0,100.0,100.0,100.0,-1.0,-1);

    robot.FT_CalCenterStart();
    robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal);
    robot.MoveCart(&desc_pos, 9,0,100.0,100.0,100.0,-1.0,-1);
    robot.WaitMs(1000);

    dir = 2;
    robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal);
    robot.FT_CalCenterEnd(&xcenter);
    printf("xcenter:%f,%f,%f,%f,%f,%f\n",xcenter.tran.x,xcenter.tran.y,xcenter.tran.z,xcenter.rpy.rx,xcenter.rpy.ry,xcenter.rpy.rz);
    robot.MoveCart(&xcenter, 9,0,60.0,50.0,50.0,-1.0,-1);

    robot.FT_CalCenterStart();
    dir = 1;
    axis = 2;
    lin_v = 6.0;
    maxdis = 150.0;
    robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal);
    robot.MoveCart(&desc_pos, 9,0,100.0,100.0,100.0,-1.0,-1);
    robot.WaitMs(1000);

    dir = 2;
    robot.FT_FindSurface(rcs, dir, axis, lin_v, lin_a, maxdis, ft_goal);
```

(continues on next page)

(continued from previous page)

```
    robot.FT_CalCenterEnd(&ycenter);
    printf("ycenter:%f,%f,%f,%f,%f,%f\n",ycenter.tran.x,ycenter.tran.y,ycenter.tran.z,
↪ycenter.rpy.rx,ycenter.rpy.ry,ycenter.rpy.rz);
    robot.MoveCart(&ycenter, 9,0,60.0,50.0,50.0,0.0,-1);

    return 0;
}
```

### 2.1.11.26 Compliant control on

```
/**
* @brief  Compliant control on
* @param  [in] p Coefficient of position adjustment or compliance
* @param  [in] force Compliant opening force threshold, unit: N
* @return  Error code
*/
errno_t  FT_ComplianceStart(float p, float force);
```

### 2.1.11.27 Compliant control off

```
/**
* @brief  Compliant control off
* @return  Error code
*/
errno_t  FT_ComplianceStop();
```

### 2.1.11.28 Code example

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cstring>
#include <unistd.h>
#include "ERARobot.h"
#include "RobotTypes.h"

using namespace std;

int main(void)
{
    ERARobot robot;                 //Instantiate the robot object
    robot.RPC("192.168.58.2");      //Establish a communication connection with the robot␣
↪controller

    uint8_t flag = 1;
    int sensor_id = 1;
    uint8_t select[6] = {1,1,1,0,0,0};
```

(continues on next page)

(continued from previous page)

```
    float ft_pid[6] = {0.0005,0.0,0.0,0.0,0.0,0.0};
    uint8_t adj_sign = 0;
    uint8_t ILC_sign = 0;
    float max_dis = 100.0;
    float max_ang = 0.0;

    ForceTorque ft;
    DescPose desc_p1, desc_p2, offset_pos;
    ExaxisPos epos;
    JointPos j1, j2;
    memset(&ft, 0, sizeof(ForceTorque));
    memset(&desc_p1, 0, sizeof(DescPose));
    memset(&desc_p2, 0, sizeof(DescPose));
    memset(&offset_pos, 0, sizeof(DescPose));
    memset(&j1, 0, sizeof(JointPos));
    memset(&j2, 0, sizeof(JointPos));
    memset(&epos, 0, sizeof(ExaxisPos));

    j1 = {-105.3,-68.0,-127.9,-75.5,90.8,77.8};
    j2 = {-105.3,-97.9,-101.5,-70.3,90.8,77.8};

    desc_p1.tran.x = -208.9;
    desc_p1.tran.y = -274.5;
    desc_p1.tran.z = 334.6;
    desc_p1.rpy.rx = 178.8;
    desc_p1.rpy.ry = -1.3;
    desc_p1.rpy.rz = 86.7;

    desc_p2.tran.x = -264.8;
    desc_p2.tran.y = -480.5;
    desc_p2.tran.z = 341.8;
    desc_p2.rpy.rx = 179.2;
    desc_p2.rpy.ry = 0.3;
    desc_p2.rpy.rz = 86.7;

    ft.fx = -10.0;
    ft.fy = -10.0;
    ft.fz = -10.0;
    robot.FT_Control(flag, sensor_id, select, &ft, ft_pid, adj_sign, ILC_sign, max_dis, max_ang);
    float p = 0.00005;
    float force = 30.0;
    robot.FT_ComplianceStart(p, force);
    int count = 15;
    while (count)
    {
        robot.MoveL(&j1,&desc_p1,9,0,100.0,180.0,100.0,-1.0,&epos,0,1,&offset_pos);
        robot.MoveL(&j2,&desc_p2,9,0,100.0,180.0,100.0,-1.0,&epos,0,0,&offset_pos);
        count -= 1;
    }
    robot.FT_ComplianceStop();
    flag = 0;
```

(continues on next page)

(continued from previous page)

```
    robot.FT_Control(flag, sensor_id, select, &ft, ft_pid, adj_sign, ILC_sign, max_dis, max_ang);

    return 0;
}
```

## 2.2 Python

This manual is the secondary development interface document of Python.

**Important:** Robot parameter unit description: The robot position unit is millimeter (mm), and the attitude unit is degree (°).

**Important:**

1) In code examples that are not specifically stated, the robot has been powered on and enabled by default;
2) All code examples in the documentation default to no interference within the robot's workspace;
3) Please use the data of the on-site robot in the actual use test.

### 2.2.1 Basic

#### 2.2.1.1 Instantiating robots

| | |
|---|---|
| Prototype | `RPC(ip)` |
| Description | Instantiating a robot object |
| Parameter | • `ip`:The IP address of the robot, with a default factory IP of "192.168.58.2" |
| Return value | • Success: Returns a robot object<br>• Failed: The created object will be destroyed |

##### 2.2.1.1.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
```

### 2.2.1.2 Query SDK version number

| | |
|---|---|
| Prototype | `GetSDKVersion()` |
| Description | Query SDK version number |
| Parameter | Nothing |
| Return value | • Success:[0,version]<br>• Failed:[errcode,] |

#### 2.2.1.2.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetSDKVersion()     # Query SDK version number
if ret[0] == 0:
    # 0-No fault, return format:[errcode,data],errcode-Fault code,data-Data
    print("SDK version is:",ret[1])
else:
    print("the errcode is: ", ret[0])
```

### 2.2.1.3 Obtain Controller IP

| | |
|---|---|
| Prototype | `GetControllerIP()` |
| Description | Obtain Controller IP |
| Parameter | Nothing |
| Return value | • Success:[0,IP]<br>• Failed:[errcode,] |

#### 2.2.1.3.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetControllerIP()     #Obtain Controller IP
if ret[0] == 0:
    print("controller ip is:",ret[1])
else:
    print("the errcode is: ", ret[0])
```

### 2.2.1.4 Control robot manual/automatic mode switch

| | |
|---|---|
| Prototype | `Mode(state)` |
| Description | Control robot manual/automatic mode switch |
| Parameter | • `state`:1-Manual mode,0-Automatic mode |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.1.4.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.Mode(0)   #The robot goes into automatic operation mode
time.sleep(1)
robot.Mode(1)   #The robot goes into manual mode
```

### 2.2.1.5 Robot drag mode

#### 2.2.1.5.1 Control the robot to enter or exit the drag teaching mode

| | |
|---|---|
| Prototype | `DragTeachSwitch(state)` |
| Description | Control the robot to enter or exit the drag teaching mode |
| Parameter | • `state`:1-Enter drag teaching mode,0-Exit drag teaching mode |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.1.5.2 Check if the robot is in drag mode

| | |
|---|---|
| Prototype | `IsInDragTeach()` |
| Description | Check if the robot is in drag mode |
| Parameter | Nothing |
| Return value | • Success:[0,state],state:0-Non drag teaching mode,1-Drag teaching mode<br>• Failed:[errcode] |

#### 2.2.1.5.2.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.Mode(1) #The robot goes into manual mode
time.sleep(1)
robot.DragTeachSwitch(1)  #When the robot enters the drag teaching mode, it can only enter the drag teaching mode in manual mode
time.sleep(1)
ret = robot.IsInDragTeach()    #Check whether the user is in drag mode, 1-Drag mode, 0-No drag mode
if ret[0] == 0:
    print("drag state is:",ret[1])
else:
    print("the errcode is: ", ret[0])
time.sleep(3)
robot.DragTeachSwitch(0)  #When the robot enters the non-drag teaching mode, it can only enter the non-drag teaching mode in manual mode
time.sleep(1)
ret = robot.IsInDragTeach()    #Check whether the user is in drag mode, 1-Drag mode, 0-No drag mode)
if ret[0] == 0:
    print("drag state is:",ret[1])
else:
    print("the errcode is: ", ret[0])
```

### 2.2.1.6 Control the robot to enable or lower enable

| | |
|---|---|
| Prototype | `RobotEnable(state)` |
| Description | Control the robot to enable or lower enable |
| Parameter | • `state`:1-Upper enable,0-Lower enable |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.1.6.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.RobotEnable(0)   #Enable the robot
```

(continues on next page)

(continued from previous page)

```python
time.sleep(3)
robot.RobotEnable(1)   #This function is enabled on the robot. After the robot is powered on, it is automatically enabled by default
```

## 2.2.2 Movement

### 2.2.2.1 Robot Jog

#### 2.2.2.1.1 jog Jog

| | |
|---|---|
| Prototype | `StartJOG(ref,nb,dir,vel,acc,max_dis)` |
| Description | jog Jog |
| Parameter | - `ref`:0-joint jogging, 2-base coordinate system jogging, 4-tool coordinate system jogging, 8-workpiece coordinate system jogging;<br>- `nb`:1-1joint(x-axis), 2-2joint(y-axis), 3-3join(z-axis), 4-4joint(rx), 5-5joint (ry), 6-6joint(rz);<br>- `dir`:0-negative direction, 1-positive direction;<br>- `vel`:Speed percentage,[0~100];<br>- `acc`:Acceleration percentage,[0~100];<br>- `max_dis`:Maximum angle/distance for a single jog,unit[° or mm] |
| Return value | - Success:[0]<br>- Failed:[errcode] |

#### 2.2.2.1.2 jog jog deceleration stops

| | |
|---|---|
| Prototype | `StopJOG(ref)` |
| Description | jog jog deceleration stops |
| Parameter | - `ref`:1-joint jog stop, 3-base coordinate system jog stop, 5-tool coordinate system jog stop, 9-workpiece coordinate system jog stop |
| Return value | - Success:[0]<br>- Failed:[errcode] |

### 2.2.2.1.3 jog jog immediately stops

| | |
|---|---|
| Prototype | `ImmStopJOG()` |
| Description | jog jog immediately stops |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.2.1.3.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
# Robot single axis point
robot.StartJOG(0,1,0,20.0,20.0,30.0)     # Single joint motion, StartJOG is a non blocking command, and other motion commands (including StartJOG) received during motion will be discarded
time.sleep(1)
#Robot single axis jog deceleration stop
# robot.StopJOG(1)
#Immediate stop of robot single axis jog
robot.ImmStopJOG()
robot.StartJOG(0,2,1,20.0,20.0,30.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(0,3,1,20.0,20.0,30.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(0,4,1,20.0,20.0,30.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(0,5,1,20.0,20.0,30.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(0,6,1,20.0,20.0,30.0)
time.sleep(1)
robot.ImmStopJOG()
# Base coordinate
robot.StartJOG(2,1,0,20.0,20.0,100.0)  #Jogging in the base coordinate system
time.sleep(1)
#Robot single axis jog deceleration stop
# robot.StopJOG(2)
# #Immediate stop of robot single axis jog
robot.ImmStopJOG()
robot.StartJOG(2,1,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
```

(continues on next page)

(continued from previous page)

```
robot.StartJOG(2,2,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,3,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,4,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,5,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(2,6,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
# Tool coordinate
robot.StartJOG(4,1,0,20.0,20.0,100.0)  #Point in the tool coordinate system
time.sleep(1)
#Robot single axis jog deceleration stop
# robot.StopJOG(5)
# #Immediate stop of robot single axis jog
robot.ImmStopJOG()
robot.StartJOG(4,1,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,2,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,3,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,4,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,5,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(4,6,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
# Job coordinate
robot.StartJOG(8,1,0,20.0,20.0,100.0)  #Point in the workpiece coordinate system
time.sleep(1)
#Robot single axis jog deceleration stop
# robot.StopJOG(9)
# #Immediate stop of robot single axis jog
robot.ImmStopJOG()
robot.StartJOG(8,1,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,2,1,20.0,20.0,100.0)
time.sleep(1)
```

(continues on next page)

(continued from previous page)

```python
robot.ImmStopJOG()
robot.StartJOG(8,3,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,4,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,5,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
robot.StartJOG(8,6,1,20.0,20.0,100.0)
time.sleep(1)
robot.ImmStopJOG()
```

### 2.2.2.2 Joint space motion

| | |
|---|---|
| Prototype | `MoveJ(joint_pos,desc_pos,tool,user,vel,acc,ovl,exaxis_pos,blendT,offset_flag,offset_pos)` |
| Description | Joint space motion |
| Parameter | • `joint_pos`:Target joint position, unit[°];<br>• `desc_pos`:Target Cartesian pose,unit[mm][°];<br>• `tool`:Tool number,[0~14];<br>• `user`:Workpiece number,[0~14];<br>• `vel`:Speed percentage,[0~100];<br>• `acc`:Acceleration percentage,[0~100],temporarily closed;<br>• `ovl`:Speed scaling factor,[0~100];<br>• `exaxis_pos`:External axis 1 position to external axis 4 position;<br>• `blendT`:[-1.0]-Motion in place (blocked), [0-500]-Smoothing time (non blocked),unit[ms];<br>• `offset_flag`:[0]-no offset, [1]-offset under workpiece/base coordinate system, [2]-offset under tool coordinate system;<br>• `offset_pos`:Pose offset,unit[mm][°] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.2.2.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
J1=[-168.847,-93.977,-93.118,-80.262,88.985,11.831]
P1=[-558.082,27.343,208.135,-177.205,-0.450,89.288]
eP1=[0.000,0.000,0.000,0.000]
```

(continues on next page)

(continued from previous page)

```python
dP1=[1.000,1.000,1.000,1.000,1.000,1.000]
J2=[168.968,-93.977,-93.118,-80.262,88.986,11.831]
P2=[-506.436,236.053,208.133,-177.206,-0.450,67.102]
eP2=[0.000,0.000,0.000,0.000]
dP2=[1.000,1.000,1.000,1.000,1.000,1.000]
robot.MoveJ(J1,P1,1,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)    #Joint space motionPTP,Tool number1,the actual test is based on field data and Tool number
robot.MoveJ(J2,P2,1,0,100.0,180.0,100.0,eP2,-1.0,0,dP2)
time.sleep(2)
j1 = robot.GetInverseKin(0,P1,-1)       #In the case of Cartesian space coordinates only, the inverse kinematic interface can be used to solve the joint position
print(j1)
j1 = [j1[1],j1[2],j1[3],j1[4],j1[5],j1[6]]
robot.MoveJ(j1,P1,1,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
j2 = robot.GetInverseKin(0,P2,-1)
print(j2)
j2 = [j2[1],j2[2],j2[3],j2[4],j2[5],j2[6]]
robot.MoveJ(j2,P2,1,0,100.0,180.0,100.0,eP2,-1.0,0,dP2)
time.sleep(2)
p1 = robot.GetForwardKin(J1)       #The forward kinematic interface can be used to solve Cartesian space coordinates with only joint positions
print(p1)
p1 = [p1[1],p1[2],p1[3],p1[4],p1[5],p1[6]]
robot.MoveJ(J1,p1,1,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
p2 = robot.GetForwardKin(J2)
print(p2)
p2 = [p2[1],p2[2],p2[3],p2[4],p2[5],p2[6]]
robot.MoveJ(J2,p2,1,0,100.0,180.0,100.0,eP2,-1.0,0,dP2)
```

#### 2.2.2.3 Linear motion in Cartesian space

| | |
|---|---|
| Prototype | `MoveL(joint_pos,desc_pos,tool,user,vel,acc,ovl,blendR, exaxis_pos,search,offset_flag,offset_pos)` |
| Description | Linear motion in Cartesian space |
| Parameter | • `joint_pos`:Target joint position, unit[°];<br>• `desc_pos`:Target Cartesian pose,unit[mm][°];<br>• `tool`:Tool number,[0~14];<br>• `user`:Workpiece number,[0~14];<br>• `vel`:Speed percentage,[0~100];<br>• `acc`:Acceleration percentage,[0~100],temporarily closed;<br>• `ovl`:Speed scaling factor,[0~100];<br>• `blendR`:[-1.0]-motion in place (blocked), [0-1000]-smooth radius(non blocked),unit[mm];<br>• `exaxis_pos`:Position of external axis 1~position of external axis 4;<br>• `search`:[0]-non welding wire positioning, [1]-welding wire positioning;<br>• `offset_flag`:[0]-no offset, [1]-offset under workpiece/base coordinate system, [2]-offset under tool coordinate system;<br>• `offset_pos`:Pose offset,unit[mm][°] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.2.3.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
J1=[95.442,-101.149,-98.699,-68.347,90.580,-47.174]
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
eP1=[0.000,0.000,0.000,0.000]
dP1=[10.000,10.000,10.000,0.000,0.000,0.000]
J2=[123.709,-121.190,-82.838,-63.499,90.471,-47.174]
P2=[-273.856,643.260,259.235,-177.972,-1.494,80.866]
eP2=[0.000,0.000,0.000,0.000]
dP2=[0.000,0.000,0.000,0.000,0.000,0.000]
J3=[167.066,-95.700,-123.494,-42.493,90.466,-47.174]
P3=[-423.044,229.703,241.080,-173.990,-5.772,123.971]
eP3=[0.000,0.000,0.000,0.000]
dP3=[0.000,0.000,0.000,0.000,0.000,0.000]
robot.MoveL(J1,P1,0,0,100.0,180.0,100.0,-1.0,eP1,0,1 ,dP1)   #Rectilinear motion in Cartesian space
robot.MoveL(J2,P2,0,0,100.0,180.0,100.0,-1.0,eP2,0,0,dP2)
robot.MoveL(J3,P3,0,0,100.0,180.0,100.0,-1.0,eP3,0,0,dP3)
```

#### 2.2.2.4 Circular arc motion in Cartesian space

| | |
|---|---|
| Prototype | `MoveC(joint_pos_p,desc_pos_p,ptool,puser,pvel,pacc,exaxis_pos_p, poffset_flag,offset_pos_p,joint_pos_t,desc_pos_t,ttool,tuser, tvel,tacc,exaxis_pos_t ,toffset_flag,offset_pos_t,ovl,blendR)` |
| Description | Circular arc motion in Cartesian space |
| Parameter | • `joint_pos_p`:Path point joint position,unit[°];<br>• `desc_pos_p`:Path point Cartesian pose,unit[mm][°];<br>• `ptool`:Tool number,[0~14];<br>• `puser`:Workpiece number,[0~14];<br>• `pvel`:Speed percentage,[0~100];<br>• `pacc`:Acceleration percentage,[0~100],temporarily closed;<br>• `exaxis_pos_p`:Position of external axis 1~position of external axis 4;<br>• `poffset_flag`:[0]-no offset, [1]-offset under workpiece/base coordinate system, [2]-offset under tool coordinate system;<br>• `offset_pos_p`:Offset,unit[mm][°];<br>• `joint_pos_t`:Target point joint position,unit[°];<br>• `desc_pos_t`:Cartesian pose of the target point,unit[mm][°];<br>• `ttool`:Tool number,[0~14];<br>• `tuser`:Workpiece number,[0~14];<br>• `tvel`:Speed percentage,[0~100];<br>• `tacc`:Acceleration percentage,[0~100],temporarily closed;<br>• `exaxis_pos_t`:Position of external axis 1~position of external axis 4;<br>• `toffset_flag`:[0]-no offset, [1]-offset under workpiece/base coordinate system, [2]-offset under tool coordinate system;<br>• `offset_pos_t`:Offset,unit[mm][°]<br>• `ovl`:Speed scaling factor,[0~100];<br>• `blendR`:[-1.0]-motion in place (blocked), [0-1000]-smooth radius(non blocked),unit[mm] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.2.4.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
J1=[121.381,-97.108,-123.768,-45.824,89.877,-47.296]
P1=[-127.772,459.534,221.274,-177.850,-2.507,78.627]
eP1=[0.000,0.000,0.000,0.000]
dP1=[10.000,10.000,10.000,10.000,10.000,10.000]
J2=[138.884,-114.522,-103.933,-49.694,90.688,-47.291]
P2=[-360.468,485.600,196.363,-178.239,-0.893,96.172]
eP2=[0.000,0.000,0.000,0.000]
dP2=[10.000,10.000,10.000,10.000,10.000,10.000]
pa2=[0.0,0.0,100.0,180.0]
```

(continues on next page)

(continued from previous page)

```
J3=[159.164,-96.105,-128.653,-41.170,90.704,-47.290]
P3=[-360.303,274.911,203.968,-176.720,-2.514,116.407]
eP3=[0.000,0.000,0.000,0.000]
dP3=[10.000,10.000,10.000,10.000,10.000,10.000]
pa3=[0.0,0.0,100.0,180.0]
dP=[10.000,10.000,10.000,10.000,10.000,10.000]
robot.MoveJ(J1,P1,0,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)       #Joint space motionPTP
robot.MoveC(J2,P2,pa2,eP2,0,dP2,J3,P3,pa3,eP3,0,dP3,100.0,-1.0)    #Circular motion in Cartesian space
```

#### 2.2.2.5 Circular motion in Cartesian space

| | |
|---|---|
| Prototype | `Circle(joint_pos_p,desc_pos_p,ptool,puser,pvel,pacc,exaxis_pos_p,joint_pos_t,desc_pos_t,ttool,tuser,tvel,tacc,exaxis_pos_t,ovl,offset_flag,offset_pos)` |
| Description | Circular motion in Cartesian space |
| Parameter | • `joint_pos_p`:Path point joint position,unit[°];<br>• `desc_pos_p`:Path point Cartesian pose,unit[mm][°];<br>• `ptool`:Tool number,[0~14];<br>• `puser`:Workpiece number,[0~14];<br>• `pvel`:Speed percentage,[0~100];<br>• `pacc`:Acceleration percentage,[0~100],temporarily closed;<br>• `exaxis_pos_p`:Position of external axis 1~position of external axis 4;<br>• `joint_pos_t`:Target point joint position,unit[°];<br>• `desc_pos_t`:Cartesian pose of the target point,unit[mm][°];<br>• `ttool`:Tool number,[0~14];<br>• `tuser`:Workpiece number,[0~14];<br>• `tvel`:Speed percentage,[0~100];<br>• `tacc`:Acceleration percentage,[0~100],temporarily closed;<br>• `exaxis_pos_t`:Position of external axis 1~position of external axis 4;<br>• `ovl`:Speed scaling factor,[0~100%];<br>• `offset_flag`:[0]-no offset, [1]-offset under workpiece/base coordinate system, [2]-offset under tool coordinate system;<br>• `offset_pos`:Offset,unit[mm][°] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.2.5.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
J1=[121.381,-97.108,-123.768,-45.824,89.877,-47.296]
P1=[-127.772,459.534,221.274,-177.850,-2.507,78.627]
eP1=[0.000,0.000,0.000,0.000]
dP1=[10.000,10.000,10.000,10.000,10.000,10.000]
J2=[138.884,-114.522,-103.933,-49.694,90.688,-47.291]
P2=[-360.468,485.600,196.363,-178.239,-0.893,96.172]
eP2=[0.000,0.000,0.000,0.000]
dP2=[10.000,10.000,10.000,10.000,10.000,10.000]
pa2=[0.0,0.0,100.0,180.0]
J3=[159.164,-96.105,-128.653,-41.170,90.704,-47.290]
P3=[-360.303,274.911,203.968,-176.720,-2.514,116.407]
eP3=[0.000,0.000,0.000,0.000]
dP3=[10.000,10.000,10.000,10.000,10.000,10.000]
pa3=[0.0,0.0,100.0,180.0]
dP=[10.000,10.000,10.000,10.000,10.000,10.000]
robot.MoveJ(J1,P1,0,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)    #Joint space motionPTP
robot.Circle(J2,P2,pa2,eP2,J3,P3,pa3,eP3,100.0,0,dP)    #Circular motion in Cartesian space
```

### 2.2.2.6 Spiral motion in Cartesian space

| | |
|---|---|
| Prototype | `NewSpiral(joint_pos,desc_pos,tool,user,vel,acc,exaxis_pos,ovl,offset_flag,offset_pos,param)` |
| Description | Spiral motion in Cartesian space |
| Parameter | • `joint_pos`:Target joint position, unit[°];<br>• `desc_pos`:Target Cartesian pose,unit[mm][°];<br>• `tool`:Tool number,[0~14];<br>• `user`:Workpiece number,[0~14];<br>• `vel`:Speed percentage,[0~100];<br>• `acc`:Acceleration percentage,[0~100],temporarily closed;<br>• `exaxis_pos`:Position of external axis 1~position of external axis 4;<br>• `ovl`:Speed scaling factor,[0~100];<br>• `offset_flag`:[0]-no offset, [1]-offset under workpiece/base coordinate system, [2]-offset under tool coordinate system;<br>• `offset_pos`:Pose offset,unit[mm][°]<br>• `param`:[circle_num,circle_angle,rad_init,rad_add,rotaxis_add,rot_direction],circle_num: number of coils, circle_angle: helix angle, rad_init: initial radius of the helix, rad_add: radius increment, rotaxis_add: axis direction increment, rot_direction: rotation direction, 0-clockwise, 1-counterclockwise |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.2.6.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
J1=[127.888,-101.535,-94.860,17.836,96.931,-61.325]
eP1=[0.000,0.000,0.000,0.000]
dP1=[50.0,0.0,0.0,-30.0,0.0,0.0]
J2=[127.888,-101.535,-94.860,17.836,96.931,-61.325]
eP2=[0.000,0.000,0.000,0.000]
dP2=[50.0,0.0,0.0,-5.0,0.0,0.0]
Pa = [5.0,5.0,50.0,10.0,10.0,0.0]
P1 = robot.GetForwardKin(J1)       #The forward kinematic interface can be used to solve Cartesian space coordinates with only joint positions
print(P1)
P1 = [P1[1],P1[2],P1[3],P1[4],P1[5],P1[6]]
robot.MoveJ(J1,P1,0,0,100.0,180.0,100.0,eP1,0.0,2,dP1)
P2 = robot.GetForwardKin(J2)       #The forward kinematic interface can be used to solve Cartesian space coordinates with only joint positions
print(P2)
P2 = [P2[1],P2[2],P2[3],P2[4],P2[5],P2[6]]
robot.NewSpiral(J2,P2,0,0,100.0,180.0,eP2,100.0,2,dP2,Pa)   #Helical motion
```

### 2.2.2.7 Joint space servo mode motion

| | |
|---|---|
| Prototype | `ServoJ(joint_pos,acc,vel,cmdT,filterT,gain)` |
| Description | Joint space servo mode motion |
| Parameter | - `joint_pos`:Target joint position, unit[°];<br>- `acc`:Acceleration, range[0~100],temporarily closed,default to 0;<br>- `vel`: Speed, range[0~100],temporarily closed,default to 0;<br>- `cmdT`:Instruction Cycle,unit[s],[0.001~0.016];<br>- `filterT`:Filtering time,unit[s],temporarily closed;<br>- `gain`:Proportional amplifier for target position,temporarily closed |
| Return value | - Success:[0]<br>- Failed:[errcode] |

#### 2.2.2.7.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
joint_pos = robot.GetActualJointPosDegree(0)
```

(continues on next page)

(continued from previous page)

```python
print(joint_pos)
joint_pos = [joint_pos[1],joint_pos[2],joint_pos[3],joint_pos[4],joint_pos[5],joint_pos[6]]
acc = 0.0
vel = 0.0
t = 0.008
lookahead_time = 0.0
P = 0.0
count = 100
while(count):
    robot.ServoJ(joint_pos, acc, vel, t, lookahead_time, P)
    joint_pos[0] = joint_pos[0] + 0.1
    count = count - 1
    time.sleep(0.008)
```

#### 2.2.2.8 Cartesian space servo mode motion

| | |
|---|---|
| Prototype | `ServoCart(mode,desc_pos,pos_gain,acc,vel,cmdT,filterT,gain)` |
| Description | Cartesian space servo mode motion |
| Parameter | • `mode`:[0]-absolute motion (base coordinate system), [1]-incremental motion (base coordinate system), [2]-incremental motion (tool coordinate system);<br>• `desc_pos`:Target Cartesian Position/Target Cartesian Position Increment;<br>• `pos_gain`:Pose increment ratio coefficient, only effective in incremental motion, range[0~1];<br>• `acc`:Acceleration, range[0~100],temporarily closed,default to 0;<br>• `vel`: Speed, range[0~100],temporarily closed,default to 0;<br>• `cmdT`:Instruction Cycle,unit[s],[0.001~0.016];<br>• `filterT`:Filtering time,unit[s],temporarily closed;<br>• `gain`:Proportional amplifier for target position,temporarily closed |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.2.8.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
mode = 2  #Tool coordinate system incremental motion
n_pos = [0.0,0.0,0.5,0.0,0.0,0.0]   #Incremental pose in Cartesian space
gain = [0.0,0.0,1.0,0.0,0.0,0.0]
acc = 0.0
vel = 0.0
t = 0.008
```

(continues on next page)

(continued from previous page)

```python
lookahead_time = 0.0
P = 0.0
count = 100
while(count):
    robot.ServoCart(mode, n_pos, gain, acc, vel, t, lookahead_time, P)
    count = count - 1
    time.sleep(0.008)
```

### 2.2.2.9 Point-to-point motion in Cartesian space

| | |
|---|---|
| Prototype | `MoveCart(desc_pos,tool,user,vel,acc,ovl,blendT,config)` |
| Description | Point-to-point motion in Cartesian space |
| Parameter | • `desc_pos`:Target Cartesian position;<br>• `tool`:Tool number,[0~14];<br>• `user`:Workpiece number,[0~14];<br>• `vel`: Speed, range[0~100],temporarily closed,default to 0;<br>• `acc`:Acceleration, range[0~100],temporarily closed,default to 0;<br>• `ovl`:Speed scaling factor,[0~100];<br>• `blendT`:[-1.0]-Motion in place (blocked), [0-500]-Smoothing time (non blocked),unit[ms];<br>• `config`:Joint configuration, [-1]-refer to the current joint position for solution, [0-7]-solve based on joint configuration |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.2.9.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
P2=[-273.856,643.260,259.235,-177.972,-1.494,80.866]
P3=[-423.044,229.703,241.080,-173.990,-5.772,123.971]
robot.MoveCart(P1,0,0,100.0,100.0,100.0,-1.0,-1)       #Point-to-point motion in Cartesian space
robot.MoveCart(P2,0,0,100.0,100.0,100.0,-1.0,-1)
robot.MoveCart(P3,0,0,100.0,100.0,100.0,0.0,-1)
time.sleep(1)
robot.StopMotion()    #Stop moving
```

#### 2.2.2.10 Robot spline motion

##### 2.2.2.10.1 Spline motion start

| | |
|---|---|
| Prototype | `SplineStart()` |
| Description | Spline motion start |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.2.10.2 Spline motion PTP

| | |
|---|---|
| Prototype | `SplinePTP(joint_pos,desc_pos,tool,user,vel,acc,ovl)` |
| Description | Spline motion PTP |
| Parameter | • `joint_pos`:Target joint position, unit[°];<br>• `desc_pos`:Target Cartesian pose,unit[mm][°];<br>• `tool`:Tool number,[0~14];<br>• `user`:Workpiece number,[0~14];<br>• `vel`: Speed, range[0~100],temporarily closed,default to 0;<br>• `acc`:Acceleration, range[0~100],temporarily closed,default to 0;<br>• `ovl`:Speed scaling factor,[0~100]; |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.2.10.3 Spline motion end

| | |
|---|---|
| Prototype | `SplineEnd()` |
| Description | Spline motion end |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.2.10.3.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
J1 = [114.578,-117.798,-97.745,-54.436,90.053,-45.216]
P1 = [-140.418,619.351,198.369,-179.948,0.023,69.793]
eP1 = [0.000,0.000,0.000,0.000]
dP1 = [0.000,0.000,0.000,0.000,0.000,0.000]
J2 = [115.401,-105.206,-117.959,-49.727,90.054,-45.222]
P2 = [-95.586,504.143,186.880,178.001,2.091,70.585]
J3 = [135.609,-103.249,-120.211,-49.715,90.058,-45.219]
P3 = [-252.429,428.903,188.492,177.804,2.294,90.782]
J4 = [154.766,-87.036,-135.672,-49.045,90.739,-45.223]
P4 = [-277.255,272.958,205.452,179.289,1.765,109.966]
robot.MoveJ(J1,P1,0,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
robot.SplineStart()    #Spline motion start
robot.SplinePTP(J1,P1,0,0,100.0,180.0,100.0)    #Spline motion PTP
robot.SplinePTP(J2,P2,0,0,100.0,180.0,100.0)
robot.SplinePTP(J3,P3,0,0,100.0,180.0,100.0)
robot.SplinePTP(J4,P4,0,0,100.0,180.0,100.0)
robot.SplineEnd()     #Spline motion end
```

### 2.2.2.11 Robot New Spline Motion

#### 2.2.2.11.1 New spline motion start

| | |
|---|---|
| Prototype | `NewSplineStart(type)` |
| Description | New spline motion start |
| Parameter | • `type`:0-arc transition, 1-given point position path point |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.2.11.2 New spline motion end

| | |
|---|---|
| Prototype | `NewSplineEnd()` |
| Description | New spline motion end |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.2.11.3 New Spline Instruction Points

| | |
|---|---|
| Prototype | `NewSplinePoint(joint_pos,desc_pos,tool,user,vel,acc,ovl,blendR, lastFlag)` |
| Description | New Spline Instruction Points |
| Parameter | • `joint_pos`:Target joint position, unit[°];<br>• `desc_pos`:Target Cartesian pose,unit[mm][°];<br>• `tool`:Tool number,[0~14];<br>• `user`:Workpiece number,[0~14];<br>• `vel`: Speed, range[0~100],temporarily closed,default to 0;<br>• `acc`:Acceleration, range[0~100],temporarily closed,default to 0;<br>• `ovl`:Speed scaling factor,[0~100];<br>• `blendR`: [0-1000]-smooth radius,unit[mm];<br>• `lastFlag`:Is it the last point, 0-No, 1-Yes |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.2.11.3.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
J1 = [114.578,-117.798,-97.745,-54.436,90.053,-45.216]
P1 = [-140.418,619.351,198.369,-179.948,0.023,69.793]
eP1 = [0.000,0.000,0.000,0.000]
dP1 = [0.000,0.000,0.000,0.000,0.000,0.000]
J2 = [115.401,-105.206,-117.959,-49.727,90.054,-45.222]
P2 = [-95.586,504.143,186.880,178.001,2.091,70.585]
J3 = [135.609,-103.249,-120.211,-49.715,90.058,-45.219]
P3 = [-252.429,428.903,188.492,177.804,2.294,90.782]
J4 = [154.766,-87.036,-135.672,-49.045,90.739,-45.223]
P4 = [-277.255,272.958,205.452,179.289,1.765,109.966]
robot.MoveJ(J1,P1,0,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
robot.NewSplineStart(1)    #The spline motion begins
robot.NewSplinePoint(J1,P1,0,0,50.0,50.0,50.0,0.0,0)    #Spline control point
robot.NewSplinePoint(J2,P2,0,0,50.0,50.0,50.0,0.0,0)
robot.NewSplinePoint(J3,P3,0,0,50.0,50.0,50.0,0.0,0)
robot.NewSplinePoint(J4,P4,0,0,50.0,50.0,50.0,0.0,1)
robot.NewSplineEnd()
```

#### 2.2.2.12 Robot terminates motion

| | |
|---|---|
| Prototype | `StopMotion()` |
| Description | To terminate motion, use the termination motion instructions as non-blocking state |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.2.12.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
P2=[-273.856,643.260,259.235,-177.972,-1.494,80.866]
P3=[-423.044,229.703,241.080,-173.990,-5.772,123.971]
robot.MoveCart(P1,0,0,100.0,100.0,100.0,-1.0,-1)       #Point to point motion in joint space
robot.MoveCart(P2,0,0,100.0,100.0,100.0,-1.0,-1)
robot.MoveCart(P3,0,0,100.0,100.0,100.0,0.0,-1)    #This motion instruction is in a non-blocking state
time.sleep(1)
robot.StopMotion()    #Stop motion
```

#### 2.2.2.13 Overall displacement of robot points

##### 2.2.2.13.1 Starting point overall offset

| | |
|---|---|
| Prototype | `PointsOffsetEnable(flag,offset_pos)` |
| Description | Starting point overall offset |
| Parameter | • `flag`:0-offset under base coordinate or workpiece coordinate system, 2-offset under tool coordinate system;<br>• `offset_pos`:Offset,unit[mm][°] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

### 2.2.2.13.2 The overall offset of the point ends

| | |
|---|---|
| Prototype | `PointsOffsetDisable()` |
| Description | The overall offset of the point ends |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.2.13.2.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
#Overall shift of robot point position
J1=[-168.847,-93.977,-93.118,-80.262,88.985,11.831]
P1=[-558.082,27.343,208.135,-177.205,-0.450,89.288]
eP1=[0.000,0.000,0.000,0.000]
dP1=[10.000,10.000,10.000,0.000,0.000,0.000]
J2=[168.968,-93.977,-93.118,-80.262,88.986,11.831]
P2=[-506.436,236.053,208.133,-177.206,-0.450,67.102]
eP2=[0.000,0.000,0.000,0.000]
dP2=[0.000,0.000,0.000,0.000,0.000,0.000]
robot.MoveJ(J1,P1,1,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
robot.MoveJ(J2,P2,1,0,100.0,180.0,100.0,eP2,-1.0,0,dP2)
time.sleep(2)
flag = 0
offset = [100.0,5.0,6.0,0.0,0.0,0.0]   #Pose offset
robot.PointsOffsetEnable(flag, offset)   #Global offset start
robot.MoveJ(J1,P1,1,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)
robot.MoveJ(J2,P2,1,0,100.0,180.0,100.0,eP2,-1.0,0,dP2)
robot.PointsOffsetDisable()  #End of global shift
```

## 2.2.3 IO

### 2.2.3.1 Set the digital output of the control box

| | |
|---|---|
| Prototype | `SetDO(id,status,smooth,block)` |
| Description | Set the digital output of the control box |
| Parameter | • `id`:IO number,range[0~15];<br>• `status`:0-off, 1-on;<br>• `smooth`:0-unsmooth, 1-smooth;<br>• `block`:0-blocking, 1-non blocking. |
| Return value | • success:[0]<br>• Failed:[errcode] |

#### 2.2.3.1.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
for i in range(0,16):
    robot.SetDO(i,1,0,0)   #Open the control box DO
robot.WaitMs(1000)
for i in range(0,16):
    robot.SetDO(i,0,0,0)   #Close the control box DO
robot.WaitMs(1000)
```

### 2.2.3.2 Set tool digital output

| | |
|---|---|
| Prototype | `SetToolDO(id,status,smooth,block)` |
| Description | Set tool digital output |
| Parameter | • `id`:IO number,range[0~15];<br>• `status`:0-off, 1-on;<br>• `smooth`:0-unsmooth, 1-smooth;<br>• `block`:0-blocking, 1-non blocking. |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.3.2.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
for i in range(0,2):
    robot.SetToolDO(i,1,0,0)    #Open the control box DO
robot.WaitMs(1000)
robot.WaitMs(1000)
for i in range(0,2):
    robot.SetToolDO(i,0,0,0)    #Close the control box DO
```

### 2.2.3.3 Set the analog output of the control box

| | |
|---|---|
| Prototype | `SetAO(id,value,block)` |
| Description | Set the analog output of the control box |
| Parameter | • `id`:IO number,range[0~1];<br>• `value`:electricity or voltage value percentage, range [0-100%] corresponds to electricity value [0-20mA] or voltage [0-10V];<br>• `block`:[0]-blocking, [1]-non blocking |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.3.3.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.SetAO(0,0.0,0)    # Set control box analog output
robot.WaitMs(1000)
robot.SetAO(1,100.0,0)
```

#### 2.2.3.4 Set tool analog output

| | |
|---|---|
| Prototype | `SetToolAO(id,value,block)` |
| Description | Set tool analog output |
| Parameter | • `id`:IO number,range[0];<br>• `value`:electricity or voltage value percentage, range [0-100%] corresponds to electricity value [0-20mA] or voltage [0-10V];<br>• `block`:[0]-blocking, [1]-non blocking |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.3.4.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.SetToolAO(0,100.0,0)   # Set tool analog output
robot.WaitMs(1000)
robot.SetToolAO(0,0.0,0)
```

#### 2.2.3.5 Obtain the digital input of the control box

| | |
|---|---|
| Prototype | `GetDI(id,block)` |
| Description | Obtain the digital input of the control box |
| Parameter | • `id`:IO number,range[0~15];<br>• `block`:[0]-blocking, [1]-non blocking |
| Return value | • Success:[0,di],di: 0-Low level,1-High level<br>• Failed:[errcode,] |

##### 2.2.3.5.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
di = robot.GetDI(0,0)   # Obtain the digital input of the control box
print(di)
```

#### 2.2.3.6 Obtain tool digital input

| | |
|---|---|
| Prototype | `GetToolDI(id,block)` |
| Description | Obtain tool digital input |
| Parameter | • `id`:IO number,range[0~1];<br>• `block`:[0]-blocking, [1]-non blocking |
| Return value | • Success:[0,di],di: 0-Low level,1-High level<br>• Failed:[errcode,] |

##### 2.2.3.6.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
tool_di = robot.GetToolDI(1,0)   # Obtain tool digital input
print(tool_di)
```

#### 2.2.3.7 Waiting for digital input from the control box

| | |
|---|---|
| Prototype | `WaitDI(id,status,maxtime,opt)` |
| Description | Waiting for digital input from the control box |
| Parameter | • `id`:IO number,range[0~15];<br>• `status`:0-off,1-on;<br>• `maxtime`:Maximum waiting time, unit[ms];<br>• `opt`:After timeout strategy, 0-program stops and prompts for timeout, 1-ignore timeout prompt to continue executing the program, 2-keep waiting |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.3.7.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.WaitDI(0,1,0,2)    # Waiting for the control box digital input
```

#### 2.2.3.8 Waiting for multiple digital inputs from the control box

| | |
|---|---|
| Prototype | `WaitMultiDI(mode,id,status,maxtime,opt)` |
| Description | Waiting for multiple digital inputs from the control box |
| Parameter | • `mode`:[0]-Multiplex AND, [1]-Multiplex OR;<br>• `id`:IO number, bit0~bit7 corresponds to DI0~DI7, bit8~bit15 corresponds to CI0~CI7;<br>• `status(uint16_t)`:bit0~bit7 corresponds to DI0~DI7 status, bit8~bit15 corresponds to the states of the CI0~CI7 status bits [0]-off, [1]-on;<br>• `maxtime`:Maximum waiting time, unit[ms];<br>• `opt`:After timeout strategy, 0-program stops and prompts for timeout, 1-ignore timeout prompt to continue executing the program, 2-keep waiting |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.3.8.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.WaitMultiDI(1,3,3,10000,2)   #  Waiting for control box multiplex digital input
```

#### 2.2.3.9 Waiting for tool digital input

| | |
|---|---|
| Prototype | `WaitToolDI(id,status,maxtime,opt)` |
| Description | Waiting for the end digital input |
| Parameter | • `id`:IO number,range[0~1];<br>• `status`:0-off,1-on;<br>• `maxtime`:Maximum waiting time, unit[ms];<br>• `opt`:after timeout strategy, 0-program stops and prompts for timeout, 1-ignore timeout prompt to continue executing the program, 2-keep waiting |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.3.9.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.WaitToolDI(1,1,0,2)    #  Wait for the tool number to enter
```

### 2.2.3.10 Waiting for terminal digital input

| | |
|---|---|
| Prototype | `GetAI(id,block)` |
| Description | Waiting for terminal digital input |
| Parameter | • `id`:IO number,range[0~1];<br>• `block`:[0]-blocking, [1]-non blocking |
| Return value | • Success:[0,value], value:Input current or voltage value percentage, range[0-100] corresponds to current value[0-20mA] or voltage[0-10V];<br>• Failed:[errcode,] |

#### 2.2.3.10.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ai = robot.GetAI(0,1)   #  Obtain control box analog input
print(ai)
```

### 2.2.3.11 Obtain tool analog input

| | |
|---|---|
| Prototype | `GetToolAI(id,block)` |
| Description | Obtain terminal analog input |
| Parameter | • `id`:IO number,range[0];<br>• `block`:[0]-blocking, [1]-non blocking |
| Return value | • Success:[0,value], value:Input current or voltage value percentage, range[0-100] corresponds to current value[0-20mA] or voltage[0-10V];<br>• Failed:[errcode,] |

#### 2.2.3.11.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
tool_ai = robot.GetToolAI(0,1)    #   Obtain tool analog input
print(tool_ai)
```

### 2.2.3.12 Waiting for the control box simulation input

| | |
|---|---|
| Prototype | `WaitAI(id,sign,value,maxtime,opt)` |
| Description | Waiting for the control box simulation input |
| Parameter | • `id`:IO number,range[0~1];<br>• `sign`:0-Greater than,1-Less than<br>• `value`:Input current or voltage value percentage, range[0-100] corresponds to current value[0-20mA] or voltage[0-10V];<br>• `maxtime`:Maximum waiting time, unit[ms];<br>• `opt`:After timeout strategy, 0-program stops and prompts for timeout, 1-ignore timeout prompt to continue executing the program, 2-keep waiting |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.3.12.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.WaitAI(0,0,50,0,2)   # Always waiting for tool analog input
```

#### 2.2.3.13 Waiting for tool analog input

| | |
|---|---|
| Prototype | `WaitToolAI(id,sign,value,maxtime,opt)` |
| Description | Waiting for the end analog input |
| Parameter | • `id`:IO number,range[0];<br>• `sign`:0-Greater than,1-Less than<br>• `value`: Input current or voltage value percentage, range[0-100] corresponds to current value[0-20mA] or voltage[0-10V];<br>• `maxtime`:Maximum waiting time, unit[ms];<br>• `opt`:After timeout strategy, 0-program stops and prompts for timeout, 1-ignore timeout prompt to continue executing the program, 2-keep waiting |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.3.13.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.WaitToolAI(0,0,50,0,2)   #  Always waiting for tool analog input
```

### 2.2.4 Common settings

#### 2.2.4.1 Set global speed

| | |
|---|---|
| Prototype | `SetSpeed(vel)` |
| Description | Set global speed |
| Parameter | • `vel`:Speed percentage, range[0~100] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.1.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.SetSpeed(20)   # Set the global speed. Manual mode and automatic mode are set independently
```

### 2.2.4.2 Setting System Variable Values

| | |
|---|---|
| Prototype | `SetSysVarValue(id,value)` |
| Description | Setting System Variable Values |
| Parameter | • `id`:Variable number, range[1~20];<br>• `value`:Variable value |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.2.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
for i in range(1,21):
    robot.SetSysVarValue(i,i+0.5)    #  Setting System Variable Values
robot.WaitMs(1000)
for i in range(1,21):
    sys_var = robot.GetSysVarValue(i)  #  Example Query the values of system variables
    print(sys_var)
```

### 2.2.4.3 Set Tool Coordinate System

| | |
|---|---|
| Prototype | `SetToolCoord(id,t_coord,type,install)` |
| Description | Set Tool Coordinate System |
| Parameter | • `id`:Coordinate system number, range[0~14];<br>• `t_coord`:Position of tool center point relative to end flange center, unit[mm][°];<br>• `type`:0-Tool coordinate system,1-Sensor coordinate system;<br>• `install`:Installation position,0-Robot end,1-Robot external |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.3.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
t_coord = [1.0,2.0,3.0,4.0,5.0,6.0]
robot.SetToolCoord(10,t_coord,0,0)  #  Set tool coordinate system
```

### 2.2.4.4 Set Tool Coordinate Series Table

| | |
|---|---|
| Prototype | `SetToolList(id,t_coord ,type,install)` |
| Description | Set Tool Coordinate Series Table |
| Parameter | • `id`:Coordinate system number, range[0~14];<br>• `t_coord`:Position of tool center point relative to end flange center, unit[mm][°];<br>• `type`:0-Tool coordinate system,1-Sensor coordinate system;<br>• `install`:Installation position,0-Robot end,1-Robot external |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.4.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
t_coord = [1.0,2.0,3.0,4.0,5.0,6.0]
robot.SetToolList(10,t_coord,0,0)  #  Set tool coordinate system
```

### 2.2.4.5 Set the external tool coordinate system

| | |
|---|---|
| Prototype | `SetExToolCoord(id,etcp ,etool)` |
| Description | Set the external tool coordinate system |
| Parameter | • `id`:Coordinate system number, range[0~14];<br>• `etcp`:External tool coordinate system, unit[mm][°];<br>• `etool`:End tool coordinate system, unit[mm][°]; |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.5.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
etcp = [1.0,2.0,3.0,4.0,5.0,6.0]
etool = [21.0,22.0,23.0,24.0,25.0,26.0]
robot.SetExToolCoord(10,etcp,etool)
```

### 2.2.4.6 Set external tool coordinate series table

| | |
|---|---|
| Prototype | `SetExToolList(id,etcp ,etool)` |
| Description | Set external tool coordinate series table |
| Parameter | • `id`:Coordinate system number, range[0~14];<br>• `etcp`:External tool coordinate system, unit[mm][°];<br>• `etool`:End tool coordinate system, unit[mm][°]; |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.6.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
etcp = [1.0,2.0,3.0,4.0,5.0,6.0]
etool = [21.0,22.0,23.0,24.0,25.0,26.0]
robot.SetExToolList(10,etcp,etool)
```

### 2.2.4.7 Set the workpiece coordinate system

| | |
|---|---|
| Prototype | `SetWObjCoord(id,w_coord)` |
| Description | Set the workpiece coordinate system |
| Parameter | • `id`:Coordinate system number, range[0~14];<br>• `w_coord`:Relative pose of coordinate system, unit[mm][°]; |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.7.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
w_coord = [11.0,12.0,13.0,14.0,15.0,16.0]
robot.SetWObjCoord(11,w_coord)
```

### 2.2.4.8 Set the workpiece coordinate series table

| | |
|---|---|
| Prototype | `SetWObjList(id,w_coord)` |
| Description | Set the workpiece coordinate series table |
| Parameter | • `id`:Coordinate system number, range[0~14];<br>• `w_coord`:Relative pose of coordinate system, unit[mm][°]; |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.8.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
w_coord = [11.0,12.0,13.0,14.0,15.0,16.0]
robot.SetWObjList(11,w_coord)
```

### 2.2.4.9 Set end load weight

| | |
|---|---|
| Prototype | `SetLoadWeight(weight)` |
| Description | Set end load weight |
| Parameter | • `weight`:unit[kg] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.9.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.SetLoadWeight(3.0)   # Set load weight
```

### 2.2.4.10 Set the robot installation method - fixed installation

| | |
|---|---|
| Prototype | `SetRobotInstallPos(method)` |
| Description | Set the robot installation method - fixed installation |
| Parameter | • `method`:0-Flat installation, 1-Side installation, 2-Hanging installation |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.10.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2'
robot.SetRobotInstallPos(0)    #   Set the robot installation mode
```

### 2.2.4.11 Set robot installation angle - free installation

| | |
|---|---|
| Prototype | `SetRobotInstallAngle(yangle,zangle)` |
| Description | Set robot installation angle - free installation |
| Parameter | • `yangle`:Angle of roll<br>• `zangle`:Rotation angle |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.11.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.SetRobotInstallAngle(0.0,0.0)    #   Set the robot installation Angle
```

### 2.2.4.12 Set the centroid coordinates of the end load

| | |
|---|---|
| Prototype | `SetLoadCoord(x,y,z)` |
| Description | Set the centroid coordinates of the end load |
| Parameter | • `x, y, z`: Barycentric coordinate,unit[mm] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.12.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.SetLoadCoord(3.0,4.0,5.0)    #   Set the load centroid coordinates
```

### 2.2.4.13 Waiting for specified time

| | |
|---|---|
| Prototype | `WaitMs(t_ms)` |
| Description | waiting for specified time |
| Parameter | • `t_ms`:unit[ms] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.4.13.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.WaitMs(1000)    #  Wait 1000ms
```

## 2.2.5 Security settings

### 2.2.5.1 Set collision level

| | |
|---|---|
| Prototype | `SetAnticollision (mode,level,config)` |
| Description | Set collision level |
| Parameter | • `mode`:0-level, 1-percentage;;<br>• `level=[j1,j2,j3,j4,j5,j6]`:collision threshold;<br>• `config`:0-do not update configuration file, 1-update configuration file |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.5.1.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
level = [1.0,2.0,3.0,4.0,5.0,6.0]
robot.SetAnticollision(0,level,1)     #  Set collision level
level = [50.0,20.0,30.0,40.0,50.0,60.0]
robot.SetAnticollision(1,level,1)     #  Set collision percentage
```

### 2.2.5.2 Set the strategy after collision

| | |
|---|---|
| Prototype | `SetCollisionStrategy (strategy)` |
| Description | Set the strategy after collision |
| Parameter | • `strategy`:0-Error Pause, 1-Continue Running |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.5.2.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.SetCollisionStrategy(1)    # Set post collision strategy,1-Continue Running
```

### 2.2.5.3 Set positive limit

| | |
|---|---|
| Prototype | `SetLimitPositive(p_limit)` |
| Description | Set positive limit |
| Parameter | • `p_limit=[j1,j2,j3,j4,j5,j6]`:six joint positions |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.5.3.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
p_limit = [170.0,80.0,150.0,80.0,170.0,160.0]
robot.SetLimitPositive(p_limit)   #  Set positive limit
```

### 2.2.5.4 Set negative limit

| | |
|---|---|
| Prototype | `SetLimitNegative(n_limit)` |
| Description | Set negative limit |
| Parameter | • `n_limit=[j1,j2,j3,j4,j5,j6]`:six joint positions |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.5.4.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
n_limit = [-170.0,-260.0,-150.0,-260.0,-170.0,-160.0]
robot.SetLimitNegative(n_limit)    # Set negative limit
```

### 2.2.5.5 Error status cleared

| | |
|---|---|
| Prototype | `ResetAllError()` |
| Description | Error status cleared,only resettable errors can be cleared |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.5.5.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.ResetAllError()     #  Error status cleared
```

### 2.2.5.6 Joint friction compensation switch

| | |
|---|---|
| Prototype | `FrictionCompensationOnOff(state)` |
| Description | Joint friction compensation switch |
| Parameter | • `state`:0-off,1-on |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.5.6.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.FrictionCompensationOnOff(1)   #  Joint friction compensation open
```

### 2.2.5.7 Set joint friction compensation coefficient formal installation

| | |
|---|---|
| Prototype | `SetFrictionValue_level(coeff)` |
| Description | Set joint friction compensation coefficient - formal installation |
| Parameter | • `coeff=[j1,j2,j3,j4,j5,j6]`:six joint compensation coefficients |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.5.7.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.FrictionCompensationOnOff(1)   #  Joint friction compensation open
lcoeff = [0.9,0.9,0.9,0.9,0.9,0.9]
robot.SetFrictionValue_level(lcoeff)   #  Set joint friction compensation coefficient
```

### 2.2.5.8 Set joint friction compensation coefficient - Side Mount

| | |
|---|---|
| Prototype | `SetFrictionValue_wall(coeff)` |
| Description | Set joint friction compensation coefficient - Side Mount |
| Parameter | • `coeff=[j1,j2,j3,j4,j5,j6]`:six joint compensation coefficients |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.5.8.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.FrictionCompensationOnOff(1)   #  Joint friction compensation open
wcoeff = [0.4,0.4,0.4,0.4,0.4,0.4]
robot.SetFrictionValue_wall(wcoeff)  #  Set joint friction compensation coefficient
```

### 2.2.5.9 Set joint friction compensation coefficient-Inverted

| | |
|---|---|
| Prototype | `SetFrictionValue_ceiling(coeff)` |
| Description | Set joint friction compensation coefficient-Inverted |
| Parameter | • `coeff=[j1,j2,j3,j4,j5,j6]`:six joint compensation coefficients |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.5.9.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.FrictionCompensationOnOff(1)   #  Joint friction compensation open
ccoeff = [0.6,0.6,0.6,0.6,0.6,0.6]
robot.SetFrictionValue_ceiling(ccoeff)  #  Set joint friction compensation coefficient
```

### 2.2.5.10 Set joint friction compensation coefficient-free installation

| | |
|---|---|
| Prototype | `SetFrictionValue_freedom(coeff)` |
| Description | Set joint friction compensation coefficient-free installation |
| Parameter | • `coeff=[j1,j2,j3,j4,j5,j6]`:six joint compensation coefficients |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.5.10.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.FrictionCompensationOnOff(1)   #  Joint friction compensation open
fcoeff = [0.5,0.5,0.5,0.5,0.5,0.5]
robot.SetFrictionValue_freedom(fcoeff)   #  Set joint friction compensation coefficient
```

## 2.2.6 Status query

### 2.2.6.1 Obtain robot installation angle

| | |
|---|---|
| Prototype | `GetRobotInstallAngle()` |
| Description | Obtain robot installation angle |
| Parameter | Nothing |
| Return value | • Success:[0,yangle,zangle],yangle-angle of roll,zangle-rotation angle<br>• Failed:[errcode,] |

#### 2.2.6.1.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2'
ret = robot.GetRobotInstallAngle()  # Obtain robot installation angle
print(ret)
```

### 2.2.6.2 Obtain system variable values

| | |
|---|---|
| Prototype | `GetSysVarValue(id)` |
| Description | Obtain system variable values |
| Parameter | • `id`:System variable number, range[1~20] |
| Return value | • Success:[0,var_value]<br>• Failed:[errcode,] |

#### 2.2.6.2.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2'
)for i in range(1,21):
    robot.SetSysVarValue(i,i+0.5)    #  Setting System Variable Values
robot.WaitMs(1000)
for i in range(1,21):
    sys_var = robot.GetSysVarValue(i)  #  Query system variable values
    print(sys_var)
```

### 2.2.6.3 Obtain the current joint position (angle)

| | |
|---|---|
| Prototype | `GetActualJointPosDegree(flag)` |
| Description | Obtain the current joint position (angle)) |
| Parameter | • `flag`:0-blocking, 1-non blocking |
| Return value | • Success:[0,joint_pos],joint_pos=[j1,j2,j3,j4,j5,j6]<br>• Failed:[errcode,] |

#### 2.2.6.3.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2'
ret = robot.GetActualJointPosDegree(0)  # Obtain the current joint position of the robot
print(ret)
```

### 2.2.6.4 Obtain the current joint position(radian)

| | |
|---|---|
| Prototype | `GetActualJointPosRadian(flag)` |
| Description | Obtain the current joint position(radian) |
| Parameter | • `flag`:0-blocking, 1-non blocking |
| Return value | • Success:[0,joint_pos],joint_pos=[j1,j2,j3,j4,j5,j6]<br>• Failed:[errcode,] |

#### 2.2.6.4.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetActualJointPosRadian(0)  # Obtain the current joint position of the robot
print(ret)
```

### 2.2.6.5 Obtain the current tool pose

| | |
|---|---|
| Prototype | `GetActualTCPPose(flag)` |
| Description | Obtain the current tool pose |
| Parameter | - `flag`:0-blocking, 1-non blocking |
| Return value | - Success:[0,tcp_pose],tcp_pose=[x,y,z,rx,ry,rz]<br>- Failed:[errcode,] |

#### 2.2.6.5.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetActualTCPPose(0)  # Obtain the current tool pose of the robot
print(ret)
```

### 2.2.6.6 Obtain the current tool coordinate system number

| | |
|---|---|
| Prototype | `GetActualTCPNum(flag)` |
| Description | Obtain the current tool coordinate system number |
| Parameter | - `flag`:0-blocking, 1-non blocking |
| Return value | - Success:[0,tool_id]<br>- Failed:[errcode,] |

#### 2.2.6.6.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetActualTCPNum(0)  # Obtain the current tool coordinate system number
print(ret)
```

### 2.2.6.7 Obtain the current workpiece coordinate system number

| | |
|---|---|
| Prototype | `GetActualWObjNum(flag)` |
| Description | Obtain the current workpiece coordinate system number |
| Parameter | • `flag`:0-blocking, 1-non blocking |
| Return value | • Success:[0,wobj_id]<br>• Failed:[errcode,] |

#### 2.2.6.7.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetActualWObjNum(0)  # Obtain the current workpiece coordinate system number
print(ret)
```

### 2.2.6.8 Obtain the current end flange pose

| | |
|---|---|
| Prototype | `GetActualToolFlangePose(flag)` |
| Description | Obtain the current end flange pose |
| Parameter | • `flag`:0-blocking, 1-non blocking |
| Return value | • Success:[0,flange_pose],flange_pose=[x,y,z,rx,ry,rz]<br>• Failed:[errcode,] |

#### 2.2.6.8.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetActualToolFlangePose(0)  # Obtain the current end flange pose
print(ret)
```

### 2.2.6.9 Inverse kinematics solution

| | |
|---|---|
| Prototype | `GetInverseKin(type,desc_pos,config)` |
| Description | Inverse kinematics, Cartesian pose to solve joint position |
| Parameter | • `type`:0-absolute pose (base coordinate system), 1-relative pose (base coordinate system), 2-relative pose (tool coordinate system)<br>• `desc_pose`:[x,y,z,rx,ry,rz],tool posture,unit[mm][°]<br>• `config`:Joint configuration, [-1]-refer to the current joint position for solution, [0-7]-solve based on joint configuration |
| Return value | • Success:[0,joint_pos],joint_pos=[j1,j2,j3,j4,j5,j6]<br>• Failed:[errcode,] |

#### 2.2.6.9.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
ret = robot.GetInverseKin(0,P1,-1)
print(ret)
```

### 2.2.6.10 Inverse kinematics solution - Specify reference location

| | |
|---|---|
| Prototype | `GetInverseKinRef(type,desc_pos,joint_pos_ref)` |
| Description | Inverse kinematics solve inverse kinematics, tool pose solve joint position, and refer to specified joint position to solve |
| Parameter | • `type`:0-absolute pose (base coordinate system), 1-relative pose (base coordinate system), 2-relative pose (tool coordinate system)<br>• `desc_pos`:[x,y,z,rx,ry,rz]tool posture,unit[mm][°]<br>• `joint_pos_ref`:[j1,j2,j3,j4,j5,j6], joint reference position,unit[°] |
| Return value | • Success:[0,joint_pos],joint_pos=[j1,j2,j3,j4,j5,j6]<br>• Failed:[errcode,] |

#### 2.2.6.10.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
J1=[95.442,-101.149,-98.699,-68.347,90.580,-47.174]
ret = robot.GetInverseKinRef(0,P1,J1)
print(ret)
```

### 2.2.6.11 Inverse kinematics solution - whether there is a solution

| | |
|---|---|
| Prototype | `GetInverseKinHasSolution(type,desc_pos,joint_pos_ref)` |
| Description | Inverse kinematics, tool pose solution, whether joint position is solved |
| Parameter | • `type`:0-Absolute pose (base coordinate system), 1-Relative pose (base coordinate system), 2-Relative pose (tool coordinate system)<br>• `desc_pos`:[x,y,z,rx,ry,rz]tool posture, unit[mm][°]<br>• `joint_pos_ref`:[j1,j2,j3,j4,j5,j6],joint reference position, unit[°] |
| Return value | • Success:[0,result],“True”-with solution,“False”-without solution<br>• Failed:[errcode,] |

#### 2.2.6.11.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
P1=[75.414,568.526,338.135,-178.348,-0.930,52.611]
J1=[95.442,-101.149,-98.699,-68.347,90.580,-47.174]
ret = robot.GetInverseKinHasSolution(0,P1,J1)
print(ret)
```

### 2.2.6.12 Forward kinematics solution

| | |
|---|---|
| Prototype | `GetForwardKin(joint_pos)` |
| Description | Forward kinematics, joint position solving tool pose |
| Parameter | - `joint_pos`:[j1,j2,j3,j4,j5,j6]:joint Position,unit[°] |
| Return value | - Success:[0,desc_pos],desc_pos=[x,y,z,rx,ry,rz]:tool posture,unit[mm][°]<br>- Failed:[errcode,] |

#### 2.2.6.12.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
J1=[95.442,-101.149,-98.699,-68.347,90.580,-47.174]
ret = robot.GetForwardKin(J1)
print(ret)
```

### 2.2.6.13 Obtain the current joint torque

| | |
|---|---|
| Prototype | `GetJointTorques(flag)` |
| Description | Obtain the current joint torque |
| Parameter | - `flag`:0-blocking, 1-non blocking |
| Return value | - Success:[0,torques],torques=[j1,j2,j3,j4,j5,j6]<br>- Failed:[errcode,] |

#### 2.2.6.13.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2'
ret = robot.GetJointTorques(0)  # Obtain the current joint torque
print(ret)
```

### 2.2.6.14 Obtain the weight of the current load

| | |
|---|---|
| Prototype | `GetTargetPayload(flag)` |
| Description | Obtain the weight of the current load |
| Parameter | • `flag`:0-blocking, 1-non blocking |
| Return value | • Success:[0,weight],unit[kg]<br>• Failed:[errcode,] |

#### 2.2.6.14.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetTargetPayload(0)  # Obtain the weight of the current load
print(ret)
```

### 2.2.6.15 Obtain the centroid of the current load

| | |
|---|---|
| Prototype | `GetTargetPayloadCog(flag)` |
| Description | Obtain the centroid of the current load |
| Parameter | • `flag`:0-blocking, 1-non blocking |
| Return value | • Success:[0,cog], cog=[x,y,z]:barycentric coordinate,unit[mm]<br>• Failed:[errcode,] |

#### 2.2.6.15.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetTargetPayloadCog(0)  # Obtain the centroid of the current load
print(ret)
```

### 2.2.6.16 Obtain the current tool coordinate system

| | |
|---|---|
| Prototype | `GetTCPOffset(flag)` |
| Description | Obtain the current tool coordinate system |
| Parameter | • `flag`:0-blocking, 1-non blocking |
| Return value | • Success:[0,tcp_offset], tcp_offset=[x,y,z,rx,ry,rz]:,unit[mm][°]<br>• Failed:[errcode,] |

#### 2.2.6.16.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetTCPOffset(0)  # Obtain the current tool coordinate system
print(ret)
```

### 2.2.6.17 Obtain the current workpiece coordinate system

| | |
|---|---|
| Prototype | `GetWObjOffset(flag)` |
| Description | Obtain the current workpiece coordinate system |
| Parameter | • `flag`:0-blocking, 1-non blocking |
| Return value | • Success:[0,wobj_offset], wobj _offset=[x,y,z,rx,ry,rz]:relative pose,unit[mm][°]<br>• Failed:[errcode,] |

#### 2.2.6.17.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetWObjOffset(0)  # Obtain the current workpiece coordinate system
print(ret)
```

### 2.2.6.18 Obtain joint soft limit angle

| | |
|---|---|
| Prototype | `GetJointSoftLimitDeg(flag)` |
| Description | Obtain joint soft limit angle |
| Parameter | • `flag`:0-blocking, 1-non blocking |
| Return value | • Success:[0, j1min,j1max,j2min,j2max,j3min,j3max,j4min,j4max,j5min,j5max,j6min,j6max] :axis 1 to axis 6 joint negative limit and positive limit,unit[mm]<br>• Failed:[errcode,] |

#### 2.2.6.18.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetJointSoftLimitDeg(0)  # btain joint soft limit angle
print(ret)
```

### 2.2.6.19 Get system time

| | |
|---|---|
| Prototype | `GetSystemClock()` |
| Description | Get system time |
| Parameter | Nothing |
| Return value | • Success:[0,t_ms]:unit[ms]<br>• Failed:[errcode,] |

#### 2.2.6.19.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetSystemClock()  # Obtain the system time of the controller
print(ret)
```

### 2.2.6.20 Obtain the current joint configuration of the robot

| | |
|---|---|
| Prototype | `GetRobotCurJointsConfig()` |
| Description | Obtain the current joint configuration of the robot |
| Parameter | Nothing |
| Return value | • Success:[0,config]:range[0~7]<br>• Failed:[errcode,] |

#### 2.2.6.20.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetRobotCurJointsConfig()  # Obtain the current joint configuration of the robot
print(ret)
```

### 2.2.6.21 Get default speed

| | |
|---|---|
| Prototype | `GetDefaultTransVel()` |
| Description | Get default speed |
| Parameter | Nothing |
| Return value | • Success:[0,vel]:unit[mm/s]<br>• Failed:[errcode,] |

#### 2.2.6.21.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetDefaultTransVel()  # Gets the robot's current speed
print(ret)
```

### 2.2.6.22 Check if the robot motion is complete

| | |
|---|---|
| Prototype | `GetRobotMotionDone()` |
| Description | Check if the robot motion is complete |
| Parameter | Nothing |
| Return value | • Success:[0,state],state:0-incomplete,1-complete<br>• Failed:[errcode,] |

#### 2.2.6.22.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
ret = robot.GetRobotMotionDone()    #Query the motion completion status of the robot
if ret[0] == 0:
    print(ret[1])
else:
    print("the errcode is: ", ret[0])
```

### 2.2.7 Trajectory recurrence

#### 2.2.7.1 Set trajectory recording parameters

| | |
|---|---|
| Prototype | `SetTPDParam(type,name,period_ms,di_choose,do_choose)` |
| Description | Set trajectory recording parameters |
| Parameter | • `type`:Data type, 1-joint position;<br>• `name`:Track name;<br>• `period_ms`:Sampling period, fixed value, 2ms or 4ms or 8ms;<br>• `di_choose`:DI selection, bit0~bit7 corresponds to control boxes DI0~DI7, bit8~bit9 corresponds to terminal DI0~DI1, 0-not selected, 1-selected<br>• `do_choose`:DO selection, bit0~bit7 corresponds to control boxes DO0~DO7, bit8~bit9 corresponds to terminal DO0~DO1, 0-not selected, 1-selected |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.7.1.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
type = 1  # Data type, 1-joint position
name = 'tpd2023'  # Track name
period = 4  #Sampling period, fixed value, 2ms or 4ms or 8ms
di_choose = 0 # di input configuration
do_choose = 0 # do output configuration
robot.SetTPDParam(type, name, period, di_choose, do_choose)    #Configure TPD Parameter
```

#### 2.2.7.2 Start trajectory recording

| | |
|---|---|
| Prototype | `SetTPDStart(type,name,period_ms,di_choose,do_choose)` |
| Description | Start trajectory recording |
| Parameter | • `type`:Data type, 1-joint position;<br>• `name`:Track name;<br>• `period_ms`:Sampling period, fixed value, 2ms or 4ms or 8ms;<br>• `di_choose`:DI selection, bit0~bit7 corresponds to control boxes DI0~DI7, bit8~bit9 corresponds to terminal DI0~DI1, 0-not selected, 1-selected<br>• `do_choose`:DO selection, bit0~bit7 corresponds to control boxes DO0~DO7, bit8~bit9 corresponds to terminal DO0~DO1, 0-not selected, 1-selected |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.7.3 Stop trajectory recording

| | |
|---|---|
| Prototype | `SetWebTPDStop()` |
| Description | Stop trajectory recording |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.7.3.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
type = 1  # Data type, 1-joint position
name = 'tpd2023'  # Track name
period = 4  #Sampling period, fixed value, 2ms or 4ms or 8ms
di_choose = 0 # di input configuration
do_choose = 0 # do output configuration
robot.SetTPDParam(type, name, period, di_choose, do_choose)    #Configure TPD Parameter
robot.Mode(1)  # The robot goes into manual mode
time.sleep(1)
robot.DragTeachSwitch(1)  #The robot goes into drag teaching mode
robot.SetTPDStart(type, name, period, di_choose, do_choose)   # Start recording the teaching track
time.sleep(30)
robot.SetWebTPDStop()  # Stop recording instructional tracks
robot.DragTeachSwitch(0)  #The robot enters the non-drag teaching mode
```

#### 2.2.7.4 Delete trajectory record

| | |
|---|---|
| Prototype | `SetTPDDelete(name)` |
| Description | Delete trajectory record |
| Parameter | • `name`:Track name |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.7.4.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.SetTPDDelete('tpd2023')    # Delete the TPD trace
```

### 2.2.7.5 Trajectory preloading

| | |
|---|---|
| Prototype | `LoadTPD(name)` |
| Description | Trajectory preloading |
| Parameter | • `name`:Track name |
| Return value | • Success:[0]<br>• Failed:[errcode] |

### 2.2.7.6 Trajectory reproduction

| | |
|---|---|
| Prototype | `MoveTPD(name,blend,ovl)` |
| Description | Trajectory reproduction |
| Parameter | • `name`:Track name<br>• `blend`:Is it smooth, 0-not smooth, 1-smooth<br>• `ovl`:Speed scaling factor, range[0~100] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.7.6.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
P1=[-378.9,-340.3,107.2,179.4,-1.3,125.0]
name = 'tpd2023'   #Track name
blend = 1   #Is it smooth, 0-not smooth, 1-smooth
ovl = 100.0   #Speed scaling
robot.LoadTPD(name)  #Trajectory preloading
robot.MoveCart(P1,1,0,100.0,100.0,100.0,-1.0,-1)       #Let's go to the starting point
robot.MoveTPD(name, blend, ovl)  #Trajectory reproduction
```

## 2.2.8 WebAPP program use

### 2.2.8.1 Set up and automatically load the default operating program

| | |
|---|---|
| Prototype | `LoadDefaultProgConfig(flag,program_name)` |
| Description | Set up and automatically load the default operating program |
| Parameter | • `flag`:1-automatically load the default program upon startup, 0-do not automatically load the default program<br>• `program_name`:The name and path of the homework program, such as "/fraser/movej.lua", where "/fraser/" is a fixed path |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.8.1.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.LoadDefaultProgConfig(1, "/fruser/splineptp.lua")  # Set the default job program to automatically load upon start up
```

### 2.2.8.2 Load the specified job program

| | |
|---|---|
| Prototype | `ProgramLoad(program_name)` |
| Description | Load the specified job program |
| Parameter | • `program_name`:The name and path of the homework program, such as "/fraser/movej.lua", where "/fraser/" is a fixed path |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.8.2.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
#The robot webapp program uses the interface
robot.Mode(0)   #The robot goes into automatic operation mode
```

(continues on next page)

(continued from previous page)

```
robot.ProgramLoad('/erauser/testPTP.lua')  #To load the robot program to execute, the␣
↪testPTP.lua program needs to be written on webapp first
```

### 2.2.8.3 Obtain the execution line number of the current robot job program

| | |
|---|---|
| Prototype | `GetCurrentLine()` |
| Description | Obtain the execution line number of the current robot job program |
| Parameter | Nothing |
| Return value | • Success:[0,line_num]<br>• Failed:[errcode] |

### 2.2.8.4 Run the currently loaded job program

| | |
|---|---|
| Prototype | `ProgramRun()` |
| Description | Run the currently loaded job program |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

### 2.2.8.5 Pause the currently running job program

| | |
|---|---|
| Prototype | `ProgramPause()` |
| Description | Pause the currently running job program |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

### 2.2.8.6 Resume the currently paused job program

| | |
|---|---|
| Prototype | `ProgramResume()` |
| Description | Resume the currently paused job program |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

### 2.2.8.7 Terminate the currently running job program

| | |
|---|---|
| Prototype | `ProgramStop()` |
| Description | Terminate the currently running job program |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

### 2.2.8.8 Obtain the execution status of robot job programs

| | |
|---|---|
| Prototype | `GetProgramState()` |
| Description | Obtain the execution status of robot job programs |
| Parameter | Nothing |
| Return value | • Success:[0,state],state:1-program stopped or no program running, 2-program running, 3-program paused<br>• Failed:[errcode] |

### 2.2.8.9 Obtain the name of the loaded job program

| | |
|---|---|
| Prototype | `GetLoadedProgram()` |
| Description | Obtain the name of the loaded job program |
| Parameter | Nothing |
| Return value | • Success:[0,program_name]<br>• Failed:[errcode] |

#### 2.2.8.9.1 Code example

```python
import erarpc
import time
import _thread
def print_program_state(name,rb):
    while(1):
        pstate = robot.GetProgramState()    #Query program running status,1-program stopped or Nothing program running, 2-program running, 3-program suspended
        linenum = robot.GetCurrentLine()    #Query the line number of the current job program
        name = robot.GetLoadedProgram()     #Queries the name of the loaded job program
        print("the robot program state is:",pstate[1])
        print("the robot program line number is:",linenum[1])
        print("the robot program name is:",name[1])
        time.sleep(1)
```

(continues on next page)

(continued from previous page)

```
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
#The robot webapp program uses the interface
robot.Mode(0)   #The robot entered automatic operation mode
robot.ProgramLoad('/erauser/testPTP.lua')  #To load the robot program to execute, the testPTP.lua program needs to be written on webapp first
robot.ProgramRun()     #Executive robot program
_thread.start_new_thread(print_program_state,("print_state",robot))
time.sleep(5)         #10s rest
robot.ProgramPause()   #Pause the robot program in progress
time.sleep(5)
robot.ProgramResume()  #Resume the suspended robot program
time.sleep(5)
robot.ProgramStop()    #Stop the robot program in progress
time.sleep(2)
```

## 2.2.9 Peripheral

### 2.2.9.1 Obtain gripper configuration

| | |
|---|---|
| Prototype | `GetGripperConfig()` |
| Description | Obtain gripper configuration |
| Parameter | Nothing |
| Return value | • Success:[0, company,device,softversion,bus],company:<br>• Failed:[errcode] |

### 2.2.9.2 Activate gripper

| | |
|---|---|
| Prototype | `ActGripper(index,action)` |
| Description | Activate gripper |
| Parameter | • `index`:Claw number;<br>• `action`: 0-reset, 1-activate |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.9.3 Control gripper

| | |
|---|---|
| Prototype | `MoveGripper(index,pos,speed,force,maxtime,block)` |
| Description | Control gripper |
| Parameter | • `index`:Claw number;<br>• `pos`:Position percentage, range[0~100];<br>• `speed`:Speed percentage, range[0~100];<br>• `force`:Moment percentage, range[0~100];<br>• `maxtime`:Maximum waiting time, range[0~30000],unit[ms];<br>• `block`:0-blocking, 1-non blocking |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.9.4 Obtain gripper movement status

| | |
|---|---|
| Prototype | `GetGripperMotionDone()` |
| Description | Obtain gripper movement status |
| Parameter | Nothing |
| Return value | • Success:[0,status], status:0-incomplete movement,1-exercise completion<br>• Failed:[errcode] |

#### 2.2.9.5 Configure gripper

| | |
|---|---|
| Prototype | `SetGripperConfig(company,device,softversion,bus)` |
| Description | Configure gripper |
| Parameter | • `company`:Claw manufacturers, 1-Robotiq, 2-Huiling, 3-Tianji, 4-Dahuan, 5-Zhixing;<br>• `device`:Equipment number: Robotiq(0-2F-85 series), Huiling(0-NK series, 1-Z-EFG-100), Tianji(0-TEG-110), Dahuan(0-PGI-140), Zhixing(0-CTPM2F20)<br>• `softversion`:Software version number, temporarily not used, defaults to 0;<br>• `bus`:Device mounted terminal bus position, temporarily not used, defaults to 0; |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.9.5.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.SetGripperConfig(4,0,0,1)  # Configuring Clamping Claws
time.sleep(1)
config = robot.GetGripperConfig()  # obtain gripper configuration
print(config)
robot.ActGripper(1,0)   # Claw reset
time.sleep(1)
robot.ActGripper(1,1)   # Claw activation
time.sleep(2)
robot.MoveGripper(1,100,48,46,30000,0)   # Claw movement
time.sleep(3)
robot.MoveGripper(1,0,50,0,30000,0)
ret = robot.GetGripperMotionDone()    # Example Query the status of the claw movement
print(ret)
```

## 2.2.10 Force control

### 2.2.10.1 Obtain force sensor configuration

| | |
|---|---|
| Prototype | `FT_GetConfig()` |
| Description | Obtain force sensor configuration |
| Parameter | Nothing |
| Return value | • Success:[0, company,device,softversion,bus],company:Sensor manufacturer<br>• Failed:[errcode] |

### 2.2.10.2 Force sensor configuration

| | |
|---|---|
| Prototype | `FT_SetConfig(company,device,softversion,bus)` |
| Description | Force sensor configuration |
| Parameter | • `company`:Sensor manufacturer,17-Kunwei Technology,19-Aerospace 11th Institute,20-ATI sensors, 21-Zhongke Mi Dian, 22-Weihang Sensitive Core;<br>• `device`:equipment number: Kunwei (0-KWR75B), Aerospace 11th Institute (0-MCS6A-200-4), ATI (0-AXIA80-M8), Zhongkomi Point (0-MST2010), Weihang Minxin (0-WHC6L-YB-10A);<br>• `softversion`:software version number, temporarily not used, defaults to 0;;<br>• `bus`:device mounted terminal bus position, temporarily not used, defaults to 0; |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.10.2.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
company = 17    #Sensor manufacturer,17-Kunwei Technology,
device = 0      #Sensor equipment number
softversion = 0 #software version number
bus = 1         #End bus position
robot.FT_SetConfig(company, device, softversion, bus)   #Configured force sensor
config = robot.FT_GetConfig() #Obtain the configuration information of the force sensor.The manufacturer number is one larger than the feedback
print(config)
```

### 2.2.10.3 Force sensor activation

| | |
|---|---|
| Prototype | `FT_Activate(state)` |
| Description | Force sensor activation |
| Parameter | • `state`:0-Reset,1-Activate |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.10.3.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.FT_Activate(0)  #Sensor reset
time.sleep(1)
robot.FT_Activate(1)  #Sensor activation
time.sleep(1)
```

### 2.2.10.4 Zero calibration of force sensor

| | |
|---|---|
| Prototype | `FT_SetZero(state)` |
| Description | Zero calibration of force sensor |
| Parameter | • `state`:0-Remove zero,1-Zero correction |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.10.4.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.FT_SetZero(0)   #Sensor zero removal
time.sleep(1)
robot.FT_SetZero(1)   #The zero point of the sensor should be corrected. Please note that no tool can be installed at the end of the sensor.
time.sleep(1)
```

### 2.2.10.5 Set the force sensor reference coordinate system

| | |
|---|---|
| Prototype | `FT_SetRCS(ref)` |
| Description | Set the force sensor reference coordinate system |
| Parameter | - `ref`:0-Tool coordinate system,1-Base coordinate system |
| Return value | - Success:[0]<br>- Failed:[errcode] |

#### 2.2.10.5.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
robot.FT_SetRCS(0)    #Set reference coordinate system to tool coordinate system, 0- tool coordinate system, 1- base coordinate system
time.sleep(1)
```

### 2.2.10.6 Load weight identification calculation

| | |
|---|---|
| Prototype | `FT_PdIdenCompute()` |
| Description | Load weight identification calculation |
| Parameter | Nothing |
| Return value | - Success:[0,weight] ,weight-Load weight,unit[kg]<br>- Failed:[errcode] |

#### 2.2.10.7 Load weight identification record

| | |
|---|---|
| Prototype | `FT_PdIdenRecord(tool_id)` |
| Description | Load weight identification record |
| Parameter | • `tool_id`:Sensor coordinate number,range[0~14] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.10.7.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
#Load identification. At this time, the tool to be identified is installed at the end. The tool is installed under the force sensor, and the end is vertical down
robot.FT_SetRCS(0)    #Set reference coordinate system to tool coordinate system, 0- tool coordinate system, 1- base coordinate system
time.sleep(1)
tool_id = 10  #Sensor coordinate number
tool_coord = [0.0,0.0,35.0,0.0,0.0,0.0]   # Position of sensor relative to end flange
tool_type = 1  # 0-Tool, 1-Sensor
tool_install = 0 # 0-Mount end, 1-Outside of robot
robot.SetToolCoord(tool_id,tool_coord,tool_type,tool_install)     #Set sensor coordinate system, sensor relative end flange position
time.sleep(1)
robot.FT_PdIdenRecord(tool_id)   #Record identification data
time.sleep(1)
weight = robot.FT_PdIdenCompute()  #Calculated load weight,unit[kg]
print(weight)
```

#### 2.2.10.8 Load centroid identification calculation

| | |
|---|---|
| Prototype | `FT_PdCogIdenCompute()` |
| Description | Load centroid identification calculation |
| Parameter | Nothing |
| Return value | • Success:[0,cog],cog=[cogx,cogy,cogz] ,Load centroid,unit[mm]<br>• Failed:[errcode] |

#### 2.2.10.9 Load centroid identification record

| | |
|---|---|
| Prototype | `FT_PdCogIdenRecord(tool_id)` |
| Description | Load centroid identification record |
| Parameter | • `tool_id`:Sensor coordinate number,range[0~14] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.10.9.1 Code example

```python
import erarpc
import time
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
#For load centroid identification, the robot needs to teach three different poses, then record the identification data, and finally calculate the load centroid
P1=[-160.619,-586.138,384.988,-170.166,-44.782,169.295]
robot.MoveCart(P1,9,0,100.0,100.0,100.0,-1.0,-1)         #Point to point motion in joint space
time.sleep(1)
robot.FT_PdCogIdenRecord(tool_id,1)                               #Record identification data
time.sleep(1)
P2=[-87.615,-606.209,556.119,-102.495,10.118,178.985]
robot.MoveCart(P2,9,0,100.0,100.0,100.0,-1.0,-1)
time.sleep(1)
robot.FT_PdCogIdenRecord(tool_id,2)
time.sleep(1)
P3=[41.479,-557.243,484.407,-125.174,46.995,-132.165]
robot.MoveCart(P3,9,0,100.0,100.0,100.0,-1.0,-1)
time.sleep(1)
robot.FT_PdCogIdenRecord(tool_id,3)
time.sleep(1)
cog = robot.FT_PdCogIdenCompute()   # Calculated and identified load centroid
print(cog)
```

#### 2.2.10.10 Obtain force/torque data in the reference coordinate system

| | |
|---|---|
| Prototype | `FT_GetForceTorqueRCS()` |
| Description | Obtain force/torque data in the reference coordinate system |
| Parameter | Nothing |
| Return value | • Success:[0,data] ,data=[fx,fy,fz,mx,my,mz]<br>• Failed:[errcode] |

#### 2.2.10.10.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
rcs = robot.FT_GetForceTorqueRCS()  #Query data in the sensor coordinate system
print(rcs)
```

### 2.2.10.11 Obtain raw force/torque data from the force sensor

| | |
|---|---|
| Prototype | `FT_GetForceTorqueOrigin()` |
| Description | Obtain raw force/torque data from the force sensor |
| Parameter | Nothing |
| Return value | • Success:[0,data] ,data=[fx,fy,fz,mx,my,mz]<br>• Failed:[errcode] |

#### 2.2.10.11.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
origin = robot.FT_GetForceTorqueOrigin()   #Example Query the original sensor data
print(origin)
```

### 2.2.10.12 Collision protection

| | |
|---|---|
| Prototype | `FT_Guard(flag,sensor_num,select,force_torque,max_threshold, min_threshold)` |
| Description | Collision protection |
| Parameter | • `flag`:0-Turn off collision protection, 1-Turn on collision protection;<br>• `sensor_num`:Force sensor number;<br>• `select`:Whether the six degrees of freedom detect the collision[fx,fy,fz,mx,my,mz],0-ineffective, 1-effective;<br>• `force_torque`:Collision detection force/moment,unit[N or Nm];<br>• `max_threshold`:Maximum threshold;<br>• `min_threshold`:Minimum Threshold;<br>• Force/torque detection range:(force_torque-min_threshold,force_torque+max_threshold) |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.10.12.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
actFlag = 1   #Enable flag, 0-Disable collision guard, 1-Enable collision guard
sensor_num = 1  #Force sensor number
is_select = [1,1,1,1,1,1]  #Whether the six degrees of freedom detect the collision[fx,fy,fz,mx,my,mz],0-Ineffective, 1-Effective
force_torque = [0.0,0.0,0.0,0.0,0.0,0.0]  #Collision detection force/moment,detection rangeforce_torque-min_threshold,force_torque+max_threshold
max_threshold = [10.0,10.0,10.0,10.0,10.0,10.0]  #Maximum threshold
min_threshold = [5.0,5.0,5.0,5.0,5.0,5.0]   #Minimum Threshold
P1=[-160.619,-586.138,384.988,-170.166,-44.782,169.295]
P2=[-87.615,-606.209,556.119,-102.495,10.118,178.985]
P3=[41.479,-557.243,484.407,-125.174,46.995,-132.165]
robot.FT_Guard(actFlag, sensor_num, is_select, force_torque, max_threshold, min_threshold)    #Enable collision guard
robot.MoveCart(P1,9,0,100.0,100.0,100.0,-1.0,-1)        #Point to point motion in joint space
robot.MoveCart(P2,9,0,100.0,100.0,100.0,-1.0,-1)
robot.MoveCart(P3,9,0,100.0,100.0,100.0,-1.0,-1)
actFlag = 0
robot.FT_Guard(actFlag, sensor_num, is_select, force_torque, max_threshold, min_threshold)    #Disable collision guard
```

### 2.2.10.13 Constant force control

| | |
|---|---|
| Prototype | `FT_Control(flag,sensor_num,select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)` |
| Description | Constant force control |
| Parameter | • `flag`:Constant force control open flag, 0-off, 1-on;<br>• `sensor_num`:Force sensor number;<br>• `select`:Are the six degrees of freedom detected [fx,fy,fz,mx,my,mz],0-ineffective, 1-effective;<br>• `force_torque`:Detection force/torque, unit[N or Nm];<br>• `gain`:[f_p,f_i,f_d,m_p,m_i,m_d],Force PID parameters, Torque PID parameters;<br>• `adj_sign`:Adaptive start stop status, 0-off, 1-on;<br>• `ILC_sign`: ILC control start stop status, 0-stop, 1-training, 2-practical operation;<br>• `max_dis`:Maximum adjustment distance;<br>• `max_ang`:Maximum adjustment angle; |
| Return value | • Success:[0]<br>• Failed:[errcode] |

### 2.2.10.13.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
status = 1  #Constant force control open flag, 0-off, 1-on
sensor_num = 1 #Force sensor number
is_select = [0,0,1,0,0,0]  #Six degrees of freedom choice[fx,fy,fz,mx,my,mz],0-ineffective, 1-effective
force_torque = [0.0,0.0,-10.0,0.0,0.0,0.0]  #Collision detection force and torque, detection rangeforce_torque-min_threshold,force_torque+max_threshold
gain = [0.0005,0.0,0.0,0.0,0.0,0.0]  #Maximum threshold
adj_sign = 0  #Adaptive start stop status, 0-off, 1-on
ILC_sign = 0  #ILC control start stop status, 0-stop, 1-training, 2-practical operation
max_dis = 100.0  #Maximum adjustment distance
max_ang = 0.0  #Maximum adjustment angle
J1=[-68.987,-96.414,-111.45,-61.105,92.884,11.089]
P1=[62.795,-511.979,291.697,-179.545,3.027,-170.039]
eP1=[0.000,0.000,0.000,0.000]
dP1=[0.000,0.000,0.000,0.000,0.000,0.000]
J2=[-107.596,-109.154,-104.735,-56.176,90.739,11.091]
P2=[-294.768,-503.708,233.158,179.799,0.713,151.309]
eP2=[0.000,0.000,0.000,0.000]
dP2=[0.000,0.000,0.000,0.000,0.000,0.000]
robot.MoveJ(J1,P1,9,0,100.0,180.0,100.0,eP1,-1.0,0,dP1)    #Joint space movement PTP, tool number 9, actual test was used according to field data and tool number
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)   #Constant force control
robot.MoveL(J2,P2,9,0,100.0,180.0,20.0,-1.0,eP2,0,0,dP2)   #Rectilinear motion in Cartesian space
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
```

### 2.2.10.14 Spiral line exploration

| | |
|---|---|
| Prototype | `FT_SpiralSearch(rcs,dr,fFinsih,t,vmax)` |
| Description | Spiral line exploration |
| Parameter | • `rcs`:Reference coordinate system, 0-tool coordinate system, 1-base coordinate system<br>• `dr`:Feed rate per circle radius, unit[mm];<br>• `fFinish`:Force or torque threshold (0-100), unit[N/Nm];<br>• `t`:Maximum exploration time,unit[ms];<br>• `vmax`:Maximum linear speed,unit[mm/s] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.10.14.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
#Constant force parameter
status = 1  #Constant force control open flag, 0-off, 1-on
sensor_num = 1 #Force sensor number
is_select = [0,0,1,0,0,0]  #Six degrees of freedom choice[fx,fy,fz,mx,my,mz],0-ineffective, 1-effective
force_torque = [0.0,0.0,-10.0,0.0,0.0,0.0]  #Collision detection force and torque, detection rangeforce_torque-min_threshold,force_torque+max_threshold
gain = [0.0001,0.0,0.0,0.0,0.0,0.0]  #Maximum threshold
adj_sign = 0  #Adaptive start stop status, 0-off, 1-on
ILC_sign = 0  #ILC control start stop status, 0-stop, 1-training, 2-practical operation
max_dis = 100.0  #Maximum adjustment distance
max_ang = 5.0  #Maximum adjustment angle
#Helix explore parameters
rcs = 0  #Reference frame, 0-Tool frame, 1-Base frame
dr = 0.7  #Feed per circle radius,unit[mm]
fFinish = 1.0 #Force or moment threshold0~100,unit[N or Nm]
t = 60000.0 #Maximum exploration time,unit[ms]
vmax = 3.0 #The maximum linear velocity, unit[mm/s]
is_select = [0,0,1,1,1,0]  #Six degrees of freedom choice[fx,fy,fz,mx,my,mz],0-ineffective, 1-effective
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
robot.FT_SpiralSearch(rcs,dr,fFinish,t,vmax)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
```

### 2.2.10.15 Rotate Insert

| | |
|---|---|
| Prototype | `FT_RotInsertion(rcs,angVelRot,forceInsertion,angleMax,orn, angAccmax,rotorn)` |
| Description | Rotate Insert |
| Parameter | • `rcs`:Reference coordinate system, 0-tool coordinate system, 1-base coordinate system;<br>• `angVelRot`:Rotational angular velocity: uni[t°/s];<br>• `forceInsertion`:Force or torque threshold(0~100),unit[N or Nm];<br>• `angleMax`:maximum rotation angle, unit[°];<br>• `orn`:Direction of force, 1-fz,2-mz;<br>• `angAccmax`:Maximum rotational acceleration, unit[°/s^2],not used temporarily<br>• `rotorn`:Rotation direction, 1-clockwise, 2-counterclockwise |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.10.15.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
#Constant force parameter
status = 1  #Constant force control open flag, 0-off, 1-on
sensor_num = 1 #Force sensor number
is_select = [0,0,1,0,0,0]  #Six degrees of freedom choice[fx,fy,fz,mx,my,mz],0-ineffective, 1-effective
force_torque = [0.0,0.0,-10.0,0.0,0.0,0.0]  #Collision detection force and torque, detection rangeforce_torque-min_threshold,force_torque+max_threshold
gain = [0.0001,0.0,0.0,0.0,0.0,0.0]  #Maximum threshold
adj_sign = 0  #Adaptive start stop status, 0-off, 1-on
ILC_sign = 0  #ILC control start stop status, 0-stop, 1-training, 2-practical operation
max_dis = 100.0  #Maximum adjustment distance
max_ang = 5.0  #Maximum adjustment angle
#Rotational insertion parameter
rcs = 0  #Reference frame, 0-Tool frame, 1-Base frame
angVelRot = 2.0  #Rotational angular velocity,unit[°/s]
forceInsertion = 1.0 #Force or moment threshold0~100,unit[N or Nm]
angleMax= 45 #Maximum rotation Angle,unit[°]
orn = 1 #Direction of force,1-fz,2-mz
angAccmax = 0.0 #Maximum rotational acceleration, unit[°/s^2],not used temporarily
rotorn = 1 #Rotation direction, 1-clockwise, 2-counterclockwise
s_select = [0,0,1,1,1,0]  #Six degrees of freedom choice[fx,fy,fz,mx,my,mz],0-ineffective, 1-effective
force_torque = [0.0,0.0,-10.0,0.0,0.0,0.0]  #Collision detection force and torque, detection rangeforce_torque-min_threshold,force_torque+max_threshold
gain = [0.0001,0.0,0.0,0.0,0.0,0.0]  #Maximum threshold
status = 1
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
robot.FT_RotInsertion(rcs,angVelRot,forceInsertion,angleMax,orn,angAccmax,rotorn)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
```

#### 2.2.10.16 Linear insertion

| | |
|---|---|
| Prototype | `FT_LinInsertion(rcs,force_goal,lin_v,lin_a,disMax,linorn)` |
| Description | Linear insertion |
| Parameter | • `rcs`:Reference frame, 0-Tool frame, 1-Base frame;<br>• `force_goal`:Force or torque threshold, unit[N or Nm];<br>• `lin_v`:Linear velocity, unit[mm/s];<br>• `lin_a`:Linear acceleration, unit[mm/s^2],not used temporarily;<br>• `disMax`:Maximum insertion distance,unit[mm];<br>• `linorn`:Insertion direction, 1-positive direction, 2-negative direction; |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.10.16.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
#Constant force parameter
status = 1  #Constant force control open flag, 0-off, 1-on
sensor_num = 1 #Force sensor number
is_select = [0,0,1,0,0,0]  #Six degrees of freedom choice[fx,fy,fz,mx,my,mz],0-ineffective, 1-effective
force_torque = [0.0,0.0,-10.0,0.0,0.0,0.0]  #Collision detection force and torque, detection rangeforce_torque-min_threshold,force_torque+max_threshold
gain = [0.0001,0.0,0.0,0.0,0.0,0.0]  #Maximum threshold
adj_sign = 0  #Adaptive start stop status, 0-off, 1-on
ILC_sign = 0  #ILC control start stop status, 0-stop, 1-training, 2-practical operation
max_dis = 100.0  #Maximum adjustment distance
max_ang = 5.0  #Maximum adjustment angle
#Linear insertion parameter
rcs = 0  #Reference frame, 0-Tool frame, 1-Base frame
force_goal = 20.0  #Force or moment threshold0~100,unit[N or Nm]
lin_v = 0.0 #Linear velocity,unit[mm/s]
lin_a = 0.0 #Linear acceleration, unit[mm/s^2],not used temporarily
disMax = 100.0 #Maximum insertion distance,unit[mm]
linorn = 1 #Insertion direction, 1-positive direction, 2-negative direction
is_select = [1,1,1,0,0,0]  #Six degrees of freedom choice[fx,fy,fz,mx,my,mz],0-ineffective, 1-effective
gain = [0.00005,0.0,0.0,0.0,0.0,0.0]  #Maximum threshold
force_torque = [0.0,0.0,-30.0,0.0,0.0,0.0]  #Collision detection force and torque, detection rangeforce_torque-min_threshold,force_torque+max_threshold
status = 1
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
robot.FT_LinInsertion(rcs,force_goal,lin_v,lin_a,disMax,linorn)
```

(continues on next page)

(continued from previous page)

```
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,
↪max_ang)
```

#### 2.2.10.17 Calculate the middle plane position to start

| | |
|---|---|
| Prototype | `FT_CalCenterStart()` |
| Description | Calculate the middle plane position to start |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.10.18 Calculate the middle plane position to end

| | |
|---|---|
| Prototype | `FT_CalCenterEnd()` |
| Description | Calculate the middle plane position to end |
| Parameter | Nothing |
| Return value | • Success:[0,pos] ,pos=[x,y,z,rx,ry,rz]<br>• Failed:[errcode] |

#### 2.2.10.19 Surface positioning

| | |
|---|---|
| Prototype | `FT_FindSurface (rcs,dir,axis,lin_v,lin_a,disMax,force_goal)` |
| Description | Surface positioning |
| Parameter | • `rcs`: Reference frame, 0-Tool frame, 1-Base frame;<br>• `dir`:Direction of movement, 1-positive, 2-negative;<br>• `axis`:Move Axis,1-x,2-y,3-z;<br>• `lin_v`:Exploring Linear Speed,unit[mm/s];<br>• `lin_a`:Exploring Linear Acceleration,unit[mm/s^2];<br>• `disMax`:Maximum exploration distance,unit[mm]<br>• `force_goal`:Action termination force threshold,unit[N]; |
| Return value | • Success:[0]<br>• Failed:[errcode] |

### 2.2.10.19.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
#Constant force parameter
status = 1  #Constant force control open flag, 0-off, 1-on
sensor_num = 1 #Force sensor number
is_select = [1,0,0,0,0,0]  #Six degrees of freedom choice[fx,fy,fz,mx,my,mz],0-ineffective, 1-effective
force_torque = [-2.0,0.0,0.0,0.0,0.0,0.0]  #Collision detection force and torque, detection rangeforce_torque-min_threshold,force_torque+max_threshold
gain = [0.0002,0.0,0.0,0.0,0.0,0.0]  #Maximum threshold
adj_sign = 0  #Adaptive start stop status, 0-off, 1-on
ILC_sign = 0  #ILC control start stop status, 0-stop, 1-training, 2-practical operation
max_dis = 100.0  #Maximum adjustment distance
max_ang = 5.0  #Maximum adjustment angle
#Surface positioning parameter
rcs = 0 #Reference frame, 0-Tool frame, 1-Base frame
direction = 1 #Direction of movement,1-positive direction, 2-negative direction
axis = 1 #Axis of movement,1-X,2-Y,3-Z
lin_v = 3.0  #Exploring straight-line velocity,unit[mm/s]
lin_a = 0.0  #Exploration linear acceleration,unit[mm/s^2]
disMax = 50.0 #Maximum exploration distance,unit[mm]
force_goal = 2.0 #Action termination force threshold,unit[N]
P1=[-230.959,-364.017,226.179,-179.004,0.002,89.999]
robot.MoveCart(P1,9,0,100.0,100.0,100.0,-1.0,-1)       #Point to point motion in joint space
#Look for the center in the x direction
#The first surface
robot.FT_CalCenterStart()
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
robot.FT_FindSurface(rcs,direction,axis,lin_v,lin_a,disMax,force_goal)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
robot.MoveCart(P1,9,0,100.0,100.0,100.0,-1.0,-1)       #Point to point motion in joint space
robot.WaitMs(1000)
#The second surface
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
direction = 2 #Direction of movement,1-positive direction, 2-negative direction
robot.FT_FindSurface(rcs,direction,axis,lin_v,lin_a,disMax,force_goal)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
#Calculate the x-direction center position
xcenter= robot.FT_CalCenterEnd()
print(xcenter)
xcenter = [xcenter[1],xcenter[2],xcenter[3],xcenter[4],xcenter[5],xcenter[6]]
```

(continues on next page)

(continued from previous page)

```python
robot.MoveCart(xcenter,9,0,60.0,50.0,50.0,0.0,-1)
#Look for the center in the y direction
#The first surface
robot.FT_CalCenterStart()
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
direction = 1 #Direction of movement,1-positive direction, 2-negative direction
axis = 2 #Axis of movement,1-X,2-Y,3-Z
disMax = 150.0 #Maximum exploration distance,unit[mm]
lin_v = 6.0  #Exploring straight-line velocity,unit[mm/s]
robot.FT_FindSurface(rcs,direction,axis,lin_v,lin_a,disMax,force_goal)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
robot.MoveCart(P1,9,0,100.0,100.0,100.0,-1.0,-1)       #Point to point motion in joint space
robot.WaitMs(1000)
#The second surface
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
direction = 2 #Direction of movement,1-positive direction, 2-negative direction
robot.FT_FindSurface(rcs,direction,axis,lin_v,lin_a,disMax,force_goal)
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
#Calculate the y center position
ycenter=robot.FT_CalCenterEnd()
print(ycenter)
ycenter = [ycenter[1],ycenter[2],ycenter[3],ycenter[4],ycenter[5],ycenter[6]]
robot.MoveCart(ycenter,9,0,60.0,50.0,50.0,-1.0,-1)
```

### 2.2.10.20 Flexibility control off

| | |
|---|---|
| Prototype | `FT_ComplianceStop()` |
| Description | Flexibility control off |
| Parameter | Nothing |
| Return value | • Success:[0]<br>• Failed:[errcode] |

#### 2.2.10.21 Flexibility control on

| | |
|---|---|
| Prototype | `FT_ComplianceStart(p,force)` |
| Description | Flexibility control on |
| Parameter | • `p`: Position adjustment coefficient or compliance coefficient<br>• `force`:flexibility opening force threshold, unit[N] |
| Return value | • Success:[0]<br>• Failed:[errcode] |

##### 2.2.10.21.1 Code example

```python
import erarpc
# A connection is established with the robot controller. A successful connection returns a robot object
robot = erarpc.RPC('192.168.58.2')
J1=[-105.3,-68.0,-127.9,-75.5,90.8,77.8]
P1=[-208.9,-274.5,334.6,178.8,-1.3,86.7]
eP1=[0.000,0.000,0.000,0.000]
dP1=[0.000,0.000,0.000,0.000,0.000,0.000]
J2=[-105.3,-97.9,-101.5,-70.3,90.8,77.8]
P2=[-264.8,-480.5,341.8,179.2,0.3,86.7]
eP2=[0.000,0.000,0.000,0.000]
dP2=[0.000,0.000,0.000,0.000,0.000,0.000]
#Constant force parameter
status = 1  #Constant force control open flag, 0-off, 1-on
sensor_num = 1 #Force sensor number
is_select = [1,0,0,0,0,0]  #Six degrees of freedom choice[fx,fy,fz,mx,my,mz],0-ineffective, 1-effective
force_torque = [-2.0,0.0,0.0,0.0,0.0,0.0]  #Collision detection force and torque, detection rangeforce_torque-min_threshold,force_torque+max_threshold
gain = [0.0002,0.0,0.0,0.0,0.0,0.0]  #Maximum threshold
adj_sign = 0  #Adaptive start stop status, 0-off, 1-on
ILC_sign = 0  #ILC control start stop status, 0-stop, 1-training, 2-practical operation
max_dis = 100.0  #Maximum adjustment distance
max_ang = 5.0  #Maximum adjustment angle
#Compliance control
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
p = 0.00005  #Coefficient of position adjustment or compliance
force = 30.0 #Compliant opening force threshold,unit[N]
robot.FT_ComplianceStart(p,force)
count = 15  #Number of cycles
while(count):
    robot.MoveL(J1,P1,9,0,100.0,180.0,100.0,-1.0,eP1,0,1,dP1)   #Rectilinear motion in Cartesian space
    robot.MoveL(J2,P2,9,0,100.0,180.0,100.0,-1.0,eP2,0,0,dP2)
    count = count - 1
```

(continues on next page)

(continued from previous page)

```
robot.FT_ComplianceStop()
status = 0
robot.FT_Control(status,sensor_num,is_select,force_torque,gain,adj_sign,ILC_sign,max_dis,max_ang)
```

## 2.3 Error Code Comparison Table

| Errcode | Describe | Processing method |
|---|---|---|
| -1 | Other errors | Contact the after-sales engineer to view the controller log |
| 0 | Successful call | / |
| 3 | The number of interface parameters is inconsistent | Check the number of interface parameters |
| 4 | Interface parameter value exception | Check parameter value type or range |
| 8 | Failed to open track file | Check if the TPD track file exists or the track name is correct |
| 14 | Interface execution failed | Check whether the web interface reports a fault or status feedback reports a fault |
| 18 | The robot program is running, please stop it first | Stop the program before performing other operations |
| 25 | Data exception, calculation failed | Re-calibration or identification |
| 28 | Inverse kinematics calculation results are abnormal | Check if the pose is reasonable |
| 29 | ServoJ joint overrun | Check whether the joint data is within a reasonable range |
| 30 | Non-resettable fault, please power off and restart the control box | Please power off and restart the control box |
| 34 | Wrong workpiece number | Please check that the workpiece number is reasonable |
| 36 | Filename too long | Please shorten the filename length |
| 38 | Singular pose, calculation failed | Please change pose |
| 64 | Not added to the instruction queue | Contact the after-sales engineer to view the controller log |
| 66 | The middle point 1 of the full circle/helix command is wrong | Check whether the middle point 1 data is correct |
| 67 | The middle point 2 of the full circle/helix command is wrong | Check whether the middle point 2 data is correct |
| 68 | The middle point 3 of the full circle/helix command is wrong | Check whether the middle point 3 data is correct |
| 69 | The middle point of the arc command is wrong | Check if the intermediate point data is correct |
| 70 | Arc instruction target point error | Check if the target point data is correct |
| 73 | Gripper movement error | Check whether the communication status of the gripper is normal |
| 74 | Line instruction point error | Check whether the point data is correct |
| 75 | Channel error | Check if IO number is in range |
| 76 | Wait timeout | Check whether the IO signal is input or the wiring is correct |
| 82 | TPD instruction point error | Re-record the teaching track |
| 83 | TPD instruction tool does not match current tool | Change the tool coordinate system used when teaching to TPD |
| 94 | Spline cue point error | Check whether the point data is correct |
| 108 | Wrong starting point for helix command | Check whether the starting point data is correct |
| 112 | The given pose cannot be reached | Check if the target pose is reasonable |

CHAPTER

THREE

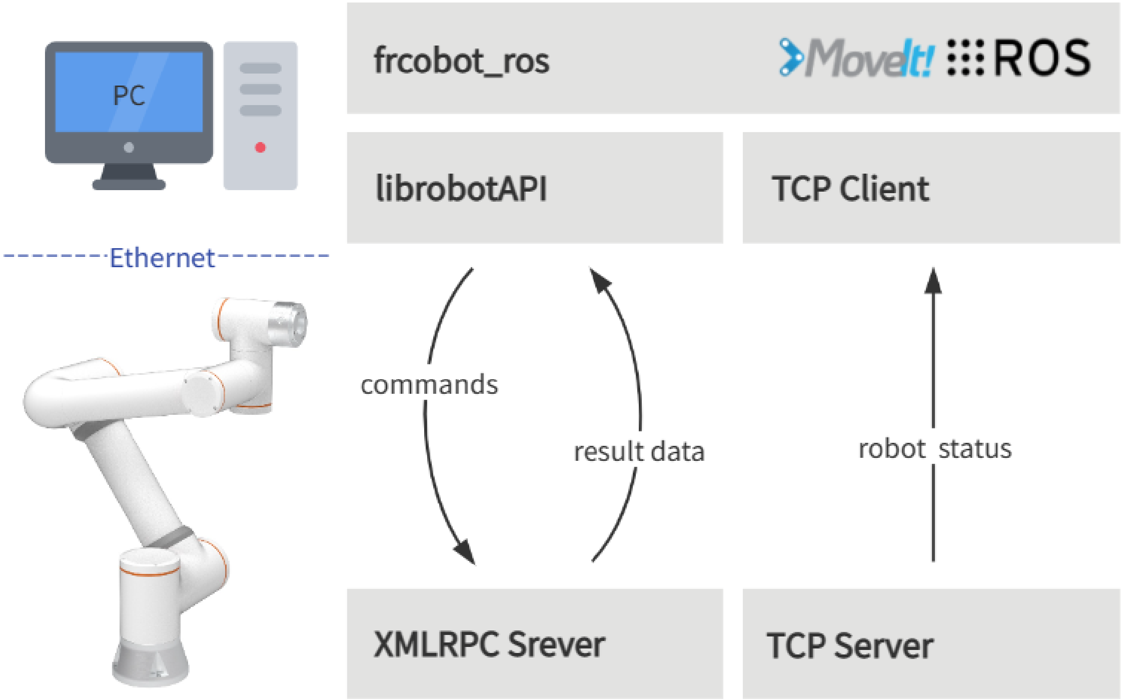
# ERACOBOT_ROS

## 3.1 Overview

The brief architecture of eracobot_ros is shown in the figure below. The collaborative robot side provides an XMLRP C server and a TCP server.

- The XMLRPC server mainly provides the robot command API to complete the robot movement and state value acquisition function, which is mainly based on the C++ SDK.
- The TCP server of the state feedback provides real-time feedback of the state of the robot, and the feedback period is 8ms.

ROS and Moveit! have been installed on the user's PC, and eracobot_ros has been compiled. Each function package in eracobot_ros includes the lib library of the robot API, and establishes a TCP client in eracobot_hw to communicate with the robot status feedback server to obtain robot status feedback data.



## 3.2 Install

This chapter introduces how to build eracobot_ros and the required installation environment.

### 3.2.1 Environmental requirements

The recommended environment for eracobot_ros is as follows:

**Note:**

- Ubuntu 18.04 LTS Bionic Beaver & ROS Melodic Morenia
- Ubuntu 20.04 LTS Focal Fossa & ROS Noetic Ninjemys

The instructions below are for Ubuntu 20.04 LTS systems and ROS Noetic Ninjemys. If you are using Melodic, replace `noetic` in the command line with `melodic`.

### 3.2.2 ROS installation & requirements

After installing the Ubuntu system, Install and configure the ROS Noetic environment.

After configuring ROS Noetic, install the required environment as follows:

```
echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
sudo apt-get install -y \
    ros-noetic-rosparam-shortcuts \
    ros-noetic-ros-control \
    ros-noetic-ros-controllers \
    ros-noetic-moveit
```

### 3.2.3 Compile

After ROS Noetic is properly installed and configured, create a Catkin workspace in a directory of your choice.

```
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws
catkin_init_workspace src
```

Then clone the eracobot_ros library from Github.

```
cd src
git clone https://github.com/ERA Automation/eracobot_ros.git
```

Build the eracobot_ros package

```
cd ~/catkin_ws
catkin_make
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

If an error occurs, please check whether the packages in the ROS installation requirements have been installed successfully. After the compilation is complete, copy the lib library to the ROS lib environment (the path is: /opt/ros/noetic/lib), so that the program can run normally .

```
# The default path of catkin_ws here is "~", if it is different, just change "~" to the actual path
sudo cp ~/catkin_ws/src/eracobot_ros/eracobot_hw/lib/* /opt/ros/noetic/lib
```

## 3.3 Quick start

### 3.3.1 eracobot_hw

`eracobot_hw` mainly provides basic functions for communicating with collaborative robots.

**Note:**

- Contains the collaborative robot status feedback msg
- Provide command demos for controlling collaborative robots
- Provide collaborative robot status feedback nodes and topics
- The status node and command demo can be quickly started through the launch file

The content of `eracobot_hw.launch` is as follows:

```
<launch>

    <!-- params -->
    <param name="robot_ip" type="string" value="192.168.58.2"/>
    <param name="robot_port" type="int" value="8083"/>

    <!-- eracobot status node -->
    <node pkg="eracobot_hw" type="eracobot_status_node" name="eracobot_status_node" output="screen" />

    <!-- eracobot control demo -->
    <node pkg="eracobot_hw" type="eracobot_cmd_demo" name="eracobot_cmd_demo" output="screen" />

</launch>
```

**Important:**

- `robot_ip` and `robot_port` need to be consistent with the IP and port of the controlled collaborative robot
- The default IP of the factory robot is 192.168.58.2, and the user status feedback port is 8083

Use the following commands to quickly start the robot status feedback node and command demo functions.

```
roslaunch eracobot_hw eracobot_hw.launch
```

Open a new terminal, and use the following commands to print and view real-time status feedback data.

```
rostopic ehco /eracobot_status
```

CHAPTER

FOUR

# ERACOBOT_ROS2

## 4.1 Overview

eracobot_ros2 is an API interface developed by ERA collaborative robot based on ROS2, aiming to use ERA SDK more conveniently for entry-level users. The configuration of the default parameters through the parameter configuration file can adapt to different customer requirements.

## 4.2 era_ros2

This chapter describes how to configure the APP running environment.

### 4.2.1 Basic environment installation

It is recommended to use it on Ubuntu22.04LTS (Jammy). After the system is installed, you can install ROS2. It is recommended to use ros2-humble. For the installation of ROS2, please refer to the tutorial: https://docs.ros.org/en/humble/index.html.

### 4.2.2 Compile and build

1. Create colcon workspace era_ros2 consists of two function packages, one is the function package erahal_msgs of the custom data structure, and the other is the program main body era_ros2 function package. After installing the basic environment, first create a colcon workspace, such as:

```
cd ~/
mkdir -p ros2_ws/src
```

2. Compile feature pack Copy the code of the installation package to the ros2_ws/src directory, and run the following command in the ros2_ws directory:

```
colcon build --packages-select erahal_msgs
```

After waiting for the previous command to finish compiling, enter:

```
colcon build --packages-select era_ros2
```

## 4.3 Quick start

### 4.3.1 Start

Open the command line under Ubuntu and enter:

```
cd ros2_ws
source install/setup.bash
ros2 run era_ros2 ros2_cmd_server
```

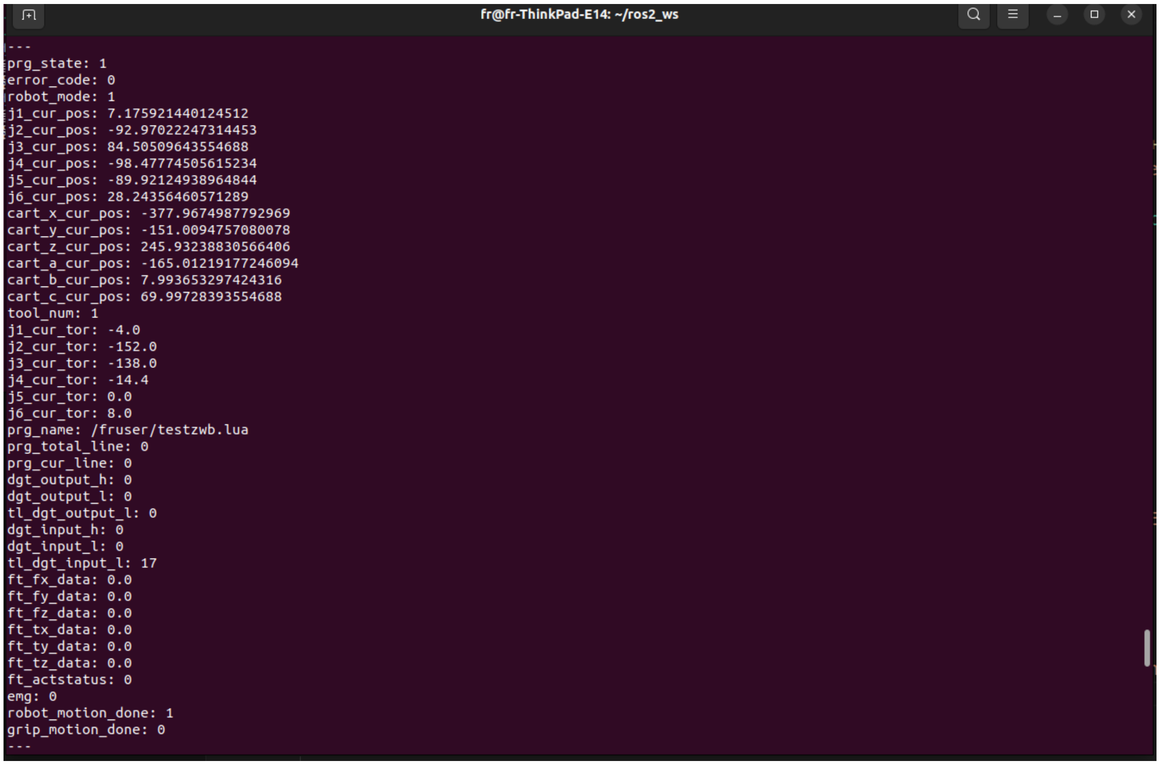
### 4.3.2 View the robotic arm status feedback

The status feedback of the robotic arm is released through the topic. Users can observe the status data refresh through the ros2 built-in command, or write a program to obtain the data. The following shows how to observe the status data of the robotic arm through the ros2 command.

Open the command line under Ubuntu and enter:

```
cd ros2_ws
source install/setup.bash
ros2 topic echo /nonrt_state_data
```

You can see the status data constantly refreshed in the command line window, as shown in the figure below.



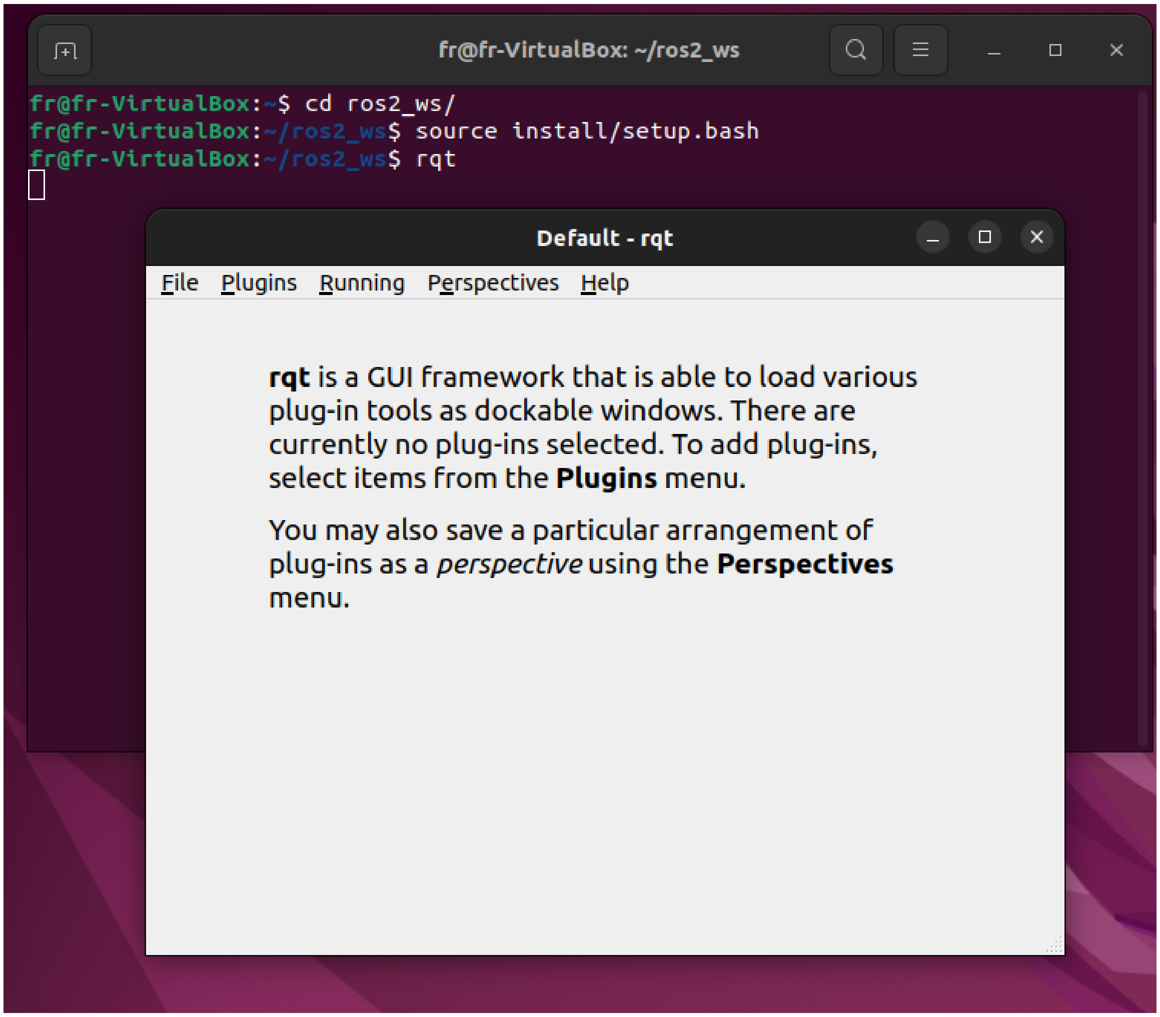
```
---
prg_state: 1
error_code: 0
robot_mode: 1
j1_cur_pos: 7.175921440124512
j2_cur_pos: -92.97022247314453
j3_cur_pos: 84.50509643554688
j4_cur_pos: -98.47774505615234
j5_cur_pos: -89.92124938964844
j6_cur_pos: 28.24356460571289
cart_x_cur_pos: -377.9674987792969
cart_y_cur_pos: -151.0094757080078
cart_z_cur_pos: 245.93238830566406
cart_a_cur_pos: -165.01219177246094
cart_b_cur_pos: 7.993653297424316
cart_c_cur_pos: 69.99728393554688
tool_num: 1
j1_cur_tor: -4.0
j2_cur_tor: -152.0
j3_cur_tor: -138.0
j4_cur_tor: -14.4
j5_cur_tor: 0.0
j6_cur_tor: 8.0
prg_name: /fruser/testzwb.lua
prg_total_line: 0
prg_cur_line: 0
dgt_output_h: 0
dgt_output_l: 0
tl_dgt_output_l: 0
dgt_input_h: 0
dgt_input_l: 0
tl_dgt_input_l: 17
ft_fx_data: 0.0
ft_fy_data: 0.0
ft_fz_data: 0.0
ft_tx_data: 0.0
ft_ty_data: 0.0
ft_tz_data: 0.0
ft_actstatus: 0
emg: 0
robot_motion_done: 1
grip_motion_done: 0
---
```

### 4.3.3 Issue order

Open the command line under Ubuntu and enter:

```
cd ros2_ws
source install/setup.bash
rqt
```

After the above command is executed, a rqt GUI interface will be called out, as shown in the figure below.

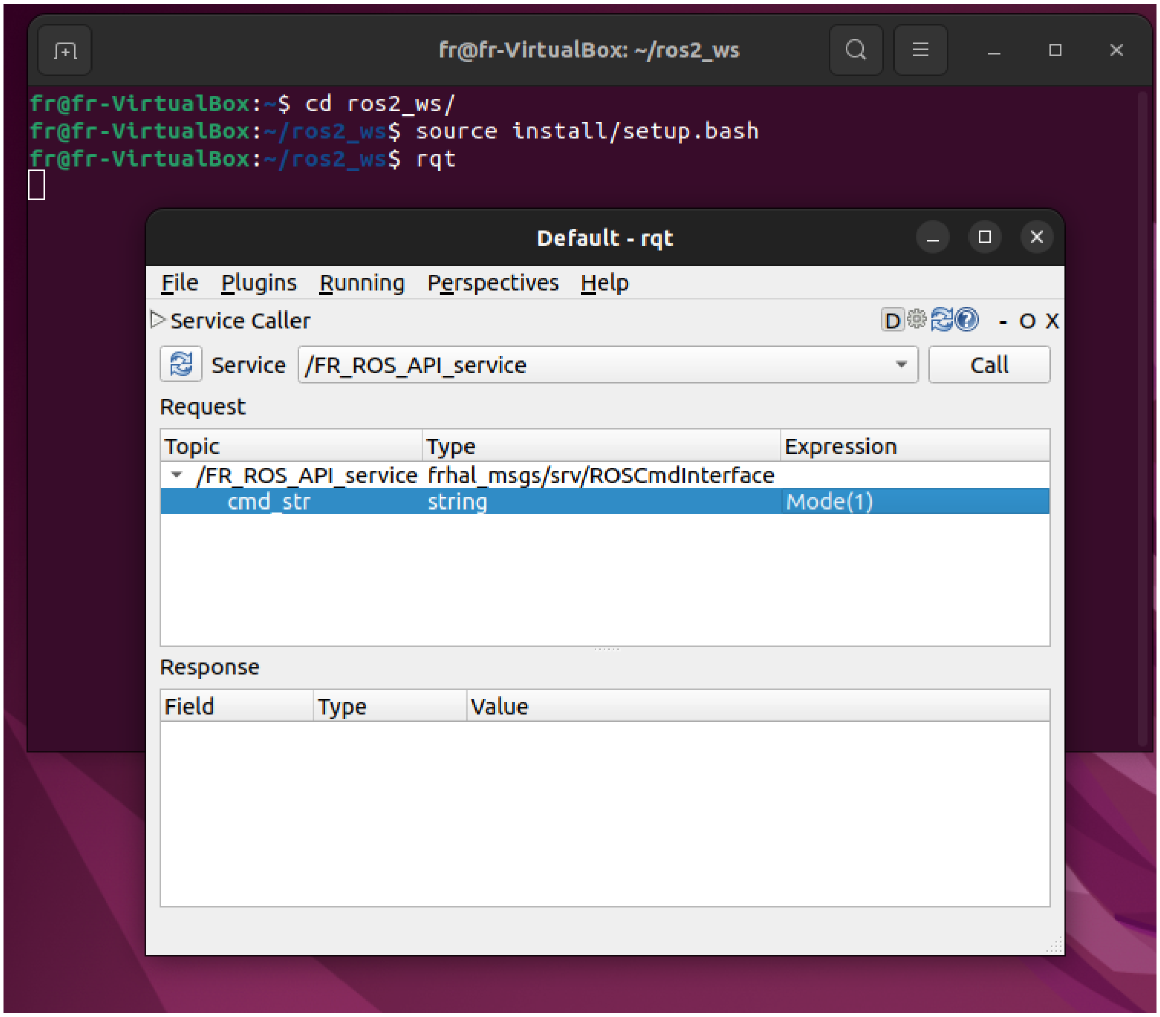


```
fr@fr-VirtualBox:~$ cd ros2_ws/
fr@fr-VirtualBox:~/ros2_ws$ source install/setup.bash
fr@fr-VirtualBox:~/ros2_ws$ rqt
```

**rqt** is a GUI framework that is able to load various plug-in tools as dockable windows. There are currently no plug-ins selected. To add plug-ins, select items from the **Plugins** menu.

You may also save a particular arrangement of plug-ins as a *perspective* using the **Perspectives** menu.

Select plugins->serivce->serivce caller in the GUI interface, call up the following interface, select /ERA_ROS_API_service,
enter the command string in the interface expression and click call to see the reply
message in the dialog box below.



```
fr@fr-VirtualBox:~$ cd ros2_ws/
fr@fr-VirtualBox:~/ros2_ws$ source install/setup.bash
fr@fr-VirtualBox:~/ros2_ws$ rqt
```

Service: /FR_ROS_API_service

| Topic | Type | Expression |
|---|---|---|
| /FR_ROS_API_service | frhal_msgs/srv/ROSCmdInterface | |
| cmd_str | string | Mode(1) |

| Field | Type | Value |
|---|---|---|

**Important:**

- Enter a string rule description:

The program internally screens the input string format. The format of the function input must be in the form of [function name](), and the parameter string in parentheses must be composed of letters, numbers, commas and minus signs. Other characters or spaces will report an error.

- Command feedback value description:

Except for the GET command which will feedback a string of strings, the feedback values of the rest of the functions are all int types. Generally, 0 means that an error occurred, and 1 means that it was executed correctly. If there are other values, please refer to the error code corresponding to the error code defined in the xmlrpc SDK.

### 4.3.4 Modify parameter

Since the simplified SDK is an improvement from the original SDK interface, it can be simplified because some parameters are given default values, and in the actual use process, there will be situations where the default parameters cannot meet the requirements. At this time, you can modify the values of the corresponding default parameters. , and then loaded into the node.

There is a era_ros2_para.yaml parameter file in the source code file. The parameters in the file are preset default param-eters, which are used to simplify the command input parameters. You can modify the parameters according to your specific needs, and then use the command to dynamically modify the parameters: ros2 param load ERA_ROS_API_nod ~/ros2_ws/src/era_ros2/era_ros2_para.yaml.

## 4.4 API Description

```
/*
function bref:store a joint space point
id - the index of point, start from 1,be aware that this id is idependant from the point id of CARTPoint function
double j1-j6 - position of 6 axies, unit is deg
*/
int JNTPoint(int id, double j1, double j2, double j3, double j4, double j5, double j6)
// example
JNTPoint(1,10,11,12,13,14,15)

/*
function bref:store a cartesian space point
id - the index of point, start from 1, be aware that this id is idependant from the point id of JNTPoint function
double x,y,z,rx,ry,yz - cartesian position,unit of distance is mm, angle unit is deg
*/
int CARTPoint(int id, double x,y,z,rx,ry,rz)
// example
CARTPoint(1,100,110,200,0,0,0)

/*
function bref:get the specific id point data of joint space or cartesian space
string name - input 'JNT' or 'CART',JNT means joint space point, 'CART' means cartesian space point
int id - point id, starts from 1
*/
string GET(string name, int id)
// example
GET(JNT,1)

/*
function bref:free drive mode switch
uint8_t state - 1-open free drive mode,0-close free drvie mode
*/
int DragTeachSwitch(uint8_t state)
// example
DragTeachSwitch(0)

```

(continues on next page)

(continued from previous page)

```
/*
function bref:robot servo on switch
uint8_t state - 1-servo on,0-servo off
*/
int RobotEnable(uint8_t state)
// example
RobotEnable(1)

/*
function bref:robot operation mode switch
uint8_t state - 1-manual mode,0-auto mode
*/
int Mode(uint8_t state)
// example
Mode(1)

/*
function bref:set robot speed on current operation mode
float vel - percentage of speed,from 1 to 100
*/
int SetSpeed(float vel)
// example
SetSpeed(10)

/*
function bref:set and load specific index tool coordinate
int id - the index of tool coordinate, from 1 to 15
float x,y,z,rx,ry,rz - transformation of tool coordinate
*/
int SetToolCoord(int id, float x,float y, float z,float rx,float ry,float rz)
// example
SetToolCoord(1,0,0,0,0,0,0)

/*
function bref:set tool coordinate list
int id - the index of tool coordinate list, from 1 to 15
float x,y,z,rx,ry,rz - transformation of tool coordinate
*/
int SetToolList(int id, float x,float y, float z,float rx,float ry,float rz );
// example
SetToolList(1,0,0,0,0,0,0)

/*
function bref:set and load specific index external tool coordinate
int id - the index of external tool coordinate, from 1 to 15
float x,y,z,rx,ry,rz - transformation of external tool coordinate
*/
int SetExToolCoord(int id, float x,float y, float z,float rx,float ry,float rz);
// example
SetExToolCoord(1,0,0,0,0,0,0)

/*
```

(continues on next page)

(continued from previous page)

```
function bref:set external tool coordinate list
int id - the index of external tool coordinate, from 1 to 15
float x,y,z,rx,ry,rz - transformation of external tool coordinate
*/
int SetExToolList(int id, float x,float y, float z,float rx,float ry,float rz);
// example
SetExToolList(1,0,0,0,0,0,0)

/*
function bref:set object coordinate
int id - the index of object coordinate,from 1 to 15
float x,y,z,rx,ry,rz - transformation of object coordinate
*/
int SetWObjCoord(int id, float x,float y, float z,float rx,float ry,float rz);
// example
SetWObjCoord(1,0,0,0,0,0,0)

/*
function bref:set object coordinate list
int id - the index of object coordinate,from 1 to 15
float x,y,z,rx,ry,rz - transformation of object coordinate
*/
int SetWObjList(int id, float x,float y, float z,float rx,float ry,float rz);
// example
SetWObjList(1,0,0,0,0,0,0)

/*
function bref:set TCP load weight
float weight - load weight, unit is kg
*/
int SetLoadWeight(float weight);
// example
SetLoadWeight(3.5)

/*
function bref:set gravity center of load weight
float x,y,z - location os gravity center,uint is mm
*/
int SetLoadCoord(float x,float y,float z);
// example
SetLoadCoord(10,20,30)

/*
function bref:set robot install direction
uint8_t install - 0-floor,1-wall,2-ceiling
*/
int SetRobotInstallPos(uint8_t install);
// example
SetRobotInstallPos(0)

/*
function bref:set robot installation dirction in free install case
```

(continues on next page)

(continued from previous page)

```
double yangle - dip angle
double zangle - rotation angle
*/
int SetRobotInstallAngle(double yangle,double zangle);
// example
SetRobotInstallAngle(90,0)

/*
function bref:set axies collision levels
float level1-level6 - collision level of each axis, from 1 to 10
*/
int SetAnticollision(float level1, float level2, float level3, float level4, float␣
↪level5, folat level6);
// example
SetAnticollision(1,1,1,1,1,1)

/*
function bref:set strategy after collision
int strategy - 0-stop motion and throw error,1-keep running
*/
int SetCollisionStrategy(int strategy);
// example
SetCollisionStrategy(1)

/*
function bref:set positive limit of each axis
float limit1-limit6 - value of limit of each axis
*/
int SetLimitPositive(float limit1, float limit2, float limit3, float limit4, float␣
↪limit5, float limit6);
// example
SetLimitPositve(100,90,90,90,90,90)

/*
function bref:set negetive limit of each axis
float limit1-limit6 - value of limit of each axis
*/
int SetLimitNegative(float limit1, float limit2, float limit3, float limit4, float␣
↪limit5, float limit6);
// example
SetLimitNegative(-100,-90,-90,-90,-90,-90)

/*
function bref:error state clear
*/
int ResetAllError();

/*
function bref:joint friction compensation switch
uint8_t state - 0-off, 1-on
*/
int FrictionCompensationOnOff(uint8_t state);
```

(continues on next page)

(continued from previous page)

```
// example
FrictionCompensationOnOff(1)

/*
function bref:set coefficient of each joint in floor installtion case
float coeff1-coeff6 - coefficient of each joint, from 0 to 1
*/
int SetFrictionValue_level(float coeff1,float coeff1,float coeff3,float coeff4,float coeff5,float coeff6);
// example
SetFrictionValue_level(1,1,1,1,1,1)

/*
function bref:set coefficient of each joint in wall installtion case
float coeff1-coeff6 - coefficient of each joint, from 0 to 1
*/
int SetFrictionValue_wall(float coeff1,float coeff1,float coeff3,float coeff4,float coeff5,float coeff6);
// example
SetFrictionValue_wall(0.5,0.5,0.5,0.5,0.5,0.5)

/*
function bref:set coefficient of each joint in ceiling installtion case
float coeff1-coeff6 - coefficient of each joint, from 0 to 1
*/
int SetFrictionValue_ceiling(float coeff1,float coeff1,float coeff3,float coeff4,float coeff5,float coeff6);
// example
SetFrictionValue_ceiling(0.5,0.5,0.5,0.5,0.5,0.5)

/*
function bref:active gripper
int index - index of gripper
uint8_t act - 0-reset, 1-active
*/
int ActGripper(int index,uint8_t act);
// example
ActGripper(1,1)

/*
function bref:control motion of gripper
int index - index of gripper
int pos - persentage of gripper position, from 0 to 100
*/
int MoveGripper(int index,int pos);
// example
MoveGripper(1,10)

/*
function bref:set digital output of control box
int id - index of IO, from 0 to 15
uint_t status - 0-off, 1-on
```

(continues on next page)

(continued from previous page)

```
*/
int SetDO(int id,uint8_t status);
// example
SetDO(1,1)

/*
function bref:set digitial output of tool
int id - index of IO, from 0 to 1
uint_t status - 0-off, 1-on
*/
int SetToolDO(int id,uint8_t status);
// example
SetToolDO(0,1)

/*
function bref:set analog output of control box
int id - index of IO, from 0 to 1
float vlaue - current of voltage persentage,from 0 to 100
*/
int SetAO(int id,float value);
// example
SetAO(1,100)

/*
function bref:set analog output of tool
int id - index of IO, from 0 to 0
float vlaue - current of voltage persentage,from 0 to 100
*/
int SetToolAO(int id,float value);
// example
SetToolAO(0,100)

/*
function bref:JOG
uint8_t ref - 0-joint coordinate jog, 2-base coordinate jog, 4-tool coordinate jog, 8-object coordinate jog
uint8_t nb - 1-axis1(x axis),2-axis2(y axis),3-axis3(z axis),4-axis4(rx),5-axis5(ry),6-axis6(rz)
uint8_t dir - 0-negetive direction, 1-positive direction
float vel - speed persentage, from 0 to 100
*/
int StartJOG(uint8_t ref, uin8_t nb, uint8_t dir, float vel);
// example
StartJOG(1,1,1,10)

/*
function bref:JOG stop
uint8_t ref - 0-joint coordinate jog stop, 2-base coordinate jog stop, 4-tool coordinate jog stop, 8-object coordinate jog stop
*/
int StopJOG(uint8_t ref);
// example
```

(continues on next page)

(continued from previous page)

```
StopJOG(1)

/*
function bref:JOG immediately stop
*/
int ImmStopJOG();

/*
function bref:point to point motion in joint space
string point_name - name of prestored point,like JNT1 means the first point of joint prestored point,CART means the first point fo cartiean prestored point
float vel - speed persentage, from 0 to 100
*/
int MoveJ(string point_name, float vel);
// example
MoveJ(JNT1,10)

/*
function bref:linear motion in cartesian space
string point_name - name of prestored point,like JNT1 means the first point of joint prestored point,CART means the first point fo cartiean prestored point
float vel - speed persentage, from 0 to 100
*/
int MoveL(string point_name,float vel);
// example
MoveL(CART1,10)

/*
function bref:arc motion in cartesian space
string point1_name point2_name - name of prestored point,like JNT1 means the first point of joint prestored point,CART means the first point fo cartiean prestored point, be aware that the two points must be the same type, which means user must input two JNT points or two CART points
float vel - speed persentage, from 0 to 100
*/
int MoveC(string point1_name,string point2_name, float vel);
// example
MoveC(JNT1,JNT2,10)

/*
function bref:joint space spline motion start
*/
int SplineStart();

/*
function bref:Spline motion in joint space, only JNT point supported, an error will be thrown if input a CART point
string point_name - name of prestored point,like JNT1 means the first point of joint prestored point
float vel - speed persentage, from 0 to 100
*/
int SplinePTP(string point_name, float vel);
```

(continues on next page)

(continued from previous page)

```
// example
SplinePTP(JNT2,10)

/*
function bref:joint space spline motion end
*/
int SplineEnd();

/*
function bref:cartesian space spline motion start
uint8_t ctlpoint - 0-trajectory through the control point, 1-trajectory will no reach the control point
*/
int NewSplineStart(uint8_t ctlpoint);
// example
NewSplineStrart(1)

/*
function bref:Spline motion in cartesian space, only CART point supported, an error will be thrown if input a JNT point
string point_name - name of prestored point,like CART1 means the first point of cartesian prestored point
float vel - speed persentage, from 0 to 100
int lastflag - 0-not last point, 1-last point
*/
int NewSplinePoint(string point_name, float vel, int lastflag);
// example
NewSplinePoint(JNT2,20,0)

/*
function bref:cartesian space spline motion end
*/
int NewSplineEnd();

/*
function bref:stop robot motion
*/
int StopMotion();

/*
function bref:points shift start
int flag - 0-shift on base/object coordinate, 2-shift on tool coordinate
double x,y,z,rx,ry,rz - transformation of shift
*/
int PointsOffsetEnable(int flag,double x,double y,double z,double rx,double ry,double rz);
// example
PointsOffsetEnable(1,10,10,10,0,0,0)

/*
function bref:points shift end
*/
```

(continues on next page)

(continued from previous page)

```
int PointsOffsetDisable();
```