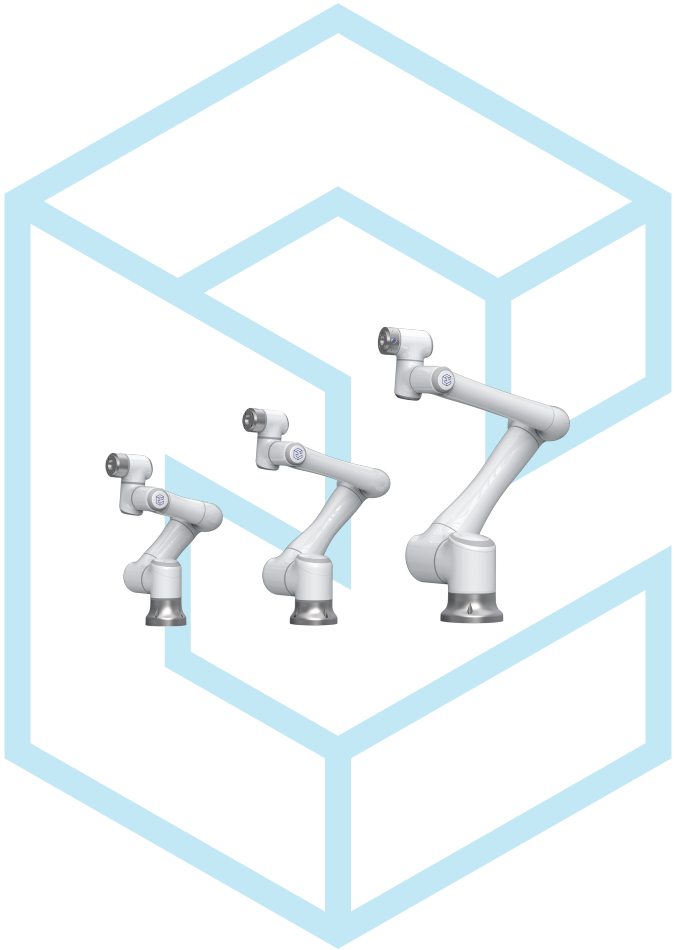
ELITE ROBOT
艾利特机器人

ELITE ROBOT EC系列

编程手册



SDK 手册

苏州艾利特机器人有限公司

2020-12-11

版本：5.0.32

目录

第 1 章 数据结构

```
#define AXIS_COUNT 8
#define ROBOT_POSE_SIZE 6

#define ELT_TRUE 1
#define ELT_FALSE 0

#define ELT_SUCCESS 1
#define ELT_FAILURE 0
#define ELT_ERROR -1

/* 登陆及操作时的上下文 */
typedefstructelt_ctx*ELT_CTX;

typedefstructelt_error_t {
        intcode;
        charerr_msg[256];
} elt_error;

// 机器人状态
enum EltRobotState {
        Stop=0,             // 停止状态
        Pause=1,            // 暂停状态
        EmeStop=2,          // 急停状态
        Running=3,          // 运行状态
        Error=4,            // 错误状态
};

// 机器人模式
enum EltRobotMode {
        Teach=0,             // 示教模式
        Play=1,              // 运行模式
        Remote=2,            // 远程模式
};

// 机器人坐标
enum EltRobotCoord {
```

```
37          ROBOT_COORDINAT_JOINT=0,
38          ROBOT_COORDINAT_CART=1,
39          ROBOT_COORDINAT_TOOL=2,
40          ROBOT_COORDINAT_USER=3,
41          ROBOT_COORDINAT_CYLINDER=4,
42  };
43
44  // 机器人循环模式
45  enum EltCycleMode {
46          CYCLE_MODE_STEP=0,
47          CYCLE_MODE_ONE=1,
48          CYCLE_MODE_SERIES=2,
49  };
50
51  // 机器人运动类型
52  enum EltMoveType \{
53          TRACK_MOVE_JOINT=0,
54          TRACK_MOVE_LINE=1,
55          TRACK_MOVE_ROTATE=2,
56          TRACK_MOVE_CIRCLE=3,
57  };
58
59  // IO状态
60  enum EltIOStatus {
61          IO_OFF=0,          // 无效
62          IO_ON=1,           // 有效
63  };
64
65  // 关节角信息
66  typedef double elt_robot_pos[AXIS_COUNT];
67
68  // 位姿信息(X,Y,Z,Rx,Ry,Rz)(X,Y,Z单位为毫米,Rx,Ry,Rz单位为弧度)
69  typedef double elt_robot_pose[ROBOT_POSE_SIZE];
70
71  // 马达速度
72  typedef double elt_motor_speed[AXIS_COUNT];
73
74  // 机器人编码器
75  typedef double elt_robot_encode[AXIS_COUNT];
```

```
76
77 // 机器人力矩
78 typedef double elt_robot_torques[AXIS_COUNT];
```

第 2 章 注意事项

2.1 内存管理事项

如需获取机器人及控制器的数据，请预先分配用于承载所获取数据的变量内存，以地址指针的形式传入接口函数，函数本身不负责对这些变量的内存管理，例如：

```
intret;
elt_error err;
int mode;
//double pose_array[AXIS_COUNT] = {0.0};
elt_robot_posepose_array;
ret=elt_get_current_encode(ctx, pose_array, &err);
ret=elt_get_robot_mode(ctx, &mode, &err);
```

第 3 章 服务接口

3.1 上下文服务 (ContextService)

3.1.1 初始化上下文

```
ELT_CTX elt_create_ctx(const char *addr, int port)
```

功能： 初始化上下文

参数： addr：目标机械臂控制系统的 IP 地址；
port：服务端口号，目前默认 8055

返回： 成功返回 ctx 上下文的指针, 错误返回 NULL

示例：
```
ctx = elt_creat_ctx( “192.168.1.200” ,8055)
```

3.1.2 销毁上下文

```
int elt_destroy_ctx(ELT_CTX ctx)
```

功能： 退出时，销毁上下文

参数： ctx：上下文的指针

返回： 成功或错误

示例：
```
ret = elt_destroy_ctx(ctx)
```

3.2 登陆服务 (LoginService)

3.2.1 登陆

```
int elt_login(ELT_CTX ctx)
```

功能： 与机器人控制系统建立连接并获得授权

参数： 登陆系统是否成功：true 表示登陆成功，false 表示登陆失败

返回： 成功或错误

示例：
```
ret = elt_login(ctx)
```

3.2.2 退出

```
int elt_logout(ELT_CTX ctx)
```

功能： 与机器人控制系统建立断开连接并取消授权

参数： ctx：登陆上下文

返回： 是否成功登出，成功或者失败

示例：
```
ret = elt_logout(ctx)
```

3.3 伺服服务 (ServoService)

3.3.1 获取机械臂伺服状态

```
int elt_get_servo_status(ELT_CTX ctx, int *status, elt_error *err)
```

功能： 获取机械臂伺服状态

参数： ctx：登录上下文
status：存储获取的机器人状态
ELT_TRUE：启用
ELT_FALSE：未启用
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_servo_status(ctx,&status,&err)
```

3.3.2 获取机械臂上下电状态

```
int elt_get_motor_status(ELT_CTX ctx, int *status, elt_error *err)
```

功能： 获取机械臂上下电状态

参数： ctx：登录上下文
status：存储获取的机器人状态
ELT_TRUE：上电
ELT_FALSE：下电
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_motor_status(ctx,&status,&err)
```

3.3.3 设置机械臂伺服状态

```
int elt_set_servo_status(ELT_CTX ctx, int status, elt_error *err)
```

功能： 设置伺服使能状态

参数： ctx：登录上下文
status：1 on 0 off
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_set_servo_status(ctx,1,&err)
```

注意： 本命令适用于 v2.10.0 及以上版本。

3.3.4 同步伺服编码器数据

```
int elt_sync_motor_status(ELT_CTX ctx, elt_error *err)
```

功能： 同步伺服编码器数据

参数： ctx：登录上下文
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_sync_motor_status(ctx,&err)
```

注意： 本命令适用于 v2.10.0 及以上版本。

3.3.5 清除报警

```
int elt_clear_alarm(ELT_CTX ctx, int force, elt_error *err)
```

功能： 清除报警

参数： ctx：登录上下文
force：普通清除 0，强制清除 1
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_sync_motor_status(ctx,&err)
```

注意： 本命令适用于 v2.10.0 及以上版本。

3.3.6 设置为同步状态

```
int elt_sync_on(ELT_CTX ctx, elt_error *err)
```

功能： 设置为同步状态

参数： ctx：登录上下文
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_sync_on(ctx,&err)
```

注意： 本命令适用于 v2.10.0 及以上版本。

3.3.7 设置为未同步状态

```
int elt_sync_off(ELT_CTX ctx,  elt_error *err)
```

功能： 设置为未同步状态

参数： ctx：登录上下文
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_sync_off(ctx,&err)
```

注意： 本命令适用于 v2.10.0 及以上版本。

3.3.8 获取同步状态

```
int elt_get_sync_state(ELT_CTX ctx, int *state, elt_error *err)
```

功能： 获取同步状态

参数： ctx：登录上下文
state：返回当前同步状态，未同步为 0，同步为 1
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_sync_state(ctx,&state,&err)
```

3.4 参数服务 (ParamService)

3.4.1 获取机器人状态

```
int elt_get_robot_state(ELT_CTX ctx, int *state, elt_error *err)
```

功能： 获取机器人状态

参数： ctx：登陆上下文
state：存储获取的机器人状态 <EltRobotState>
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_robot_state(ctx,&state,&err)
```

3.4.2 获取机器人模式

```
int elt_get_robot_mode(ELT_CTX ctx, int *mode, elt_error *err)
```

功能： 获取机器人模式

参数： ctx：登陆上下文
mode：储获取的机器人模式 <EltRobotMode>
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_robot_mode(ctx,&mode,&err)
```

3.4.3 获取机器人当前位置信息

```
int elt_get_robot_pos(ELT_CTX ctx, elt_robot_pos pos_array, elt_error
    *err)
```

功能： 获取机器人当前位置信息

参数： ctx：登陆上下文
pos_array：存储获取的机器人关节位置
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_robot_pos(ctx,pos_array,&error)
```

3.4.4 获取机器人当前位姿信息

```
int elt_get_robot_pose(ELT_CTX ctx, elt_robot_pose pose_array,
    elt_error *err)
```

功能： 获取机器人当前位姿信息

参数： ctx：登陆上下文
pos_array：存储获取的机器人位姿信息
err：错误信息

返回： 成功或者失败

示例：
```
ret = elite_get_robot_pose(ctx,pos_array,&err)
```

3.4.5 获取机器人指定用户坐标下的位姿信息

```
int elt_get_robot_pose(ELT_CTX ctx, elt_robot_pose pose_array,
    elt_error *err, elt_robot_pose coord_pose)
```

功能： 获取机器人指定用户坐标下的位姿信息

参数： ctx：登陆上下文
pos_array：存储指定用户坐标下的位姿信息
err：错误信息
coord_pose：用户坐标位姿（用户坐标标定后的计算值，存储在/rbctrl/userframe.xml文件中）

返回： 成功或者失败

示例：
```
elt_robot_pose coord_pose = {780.116522,132.508903,
-169.680244,0.000000,-0.000001,-1.570796};
ret = elt_get_robot_pose(ctx,user_pose,&err,coord_pose)
```

注意： 本命令适用于 v2.12.2 及以上版本。

3.4.6 获取机器人马达速度

```
int elt_get_motor_speed(ELT_CTX ctx, elt_motor_speed speed_array,
    elt_error *err)
```

功能： 获取机器人马达速度

参数： ctx：登陆上下文
speed_array：存储获取的机器人马达速度，转/分
err：错误信息

返回： 成功或者失败

示例：
```
elt_get_motor_speed(ctx,speed_array,&error)
```

3.4.7 获取机器人当前坐标

```
int elt_get_current_coord(ELT_CTX ctx, int *coord, elt_error *err)
```

功能： 获取机器人当前坐标

参数： ctx：登陆上下文
coord：存储获取的机器人当前坐标 <EltRobotCoord>
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_current_coord(ctx,&coord,&err)
```

3.4.8 获取机器人循环模式

```
int elt_get_cycle_mode(ELT_CTX ctx, int *mode, elt_error *err)
```

功能： 获取机器人循环模式

参数： ctx：登陆上下文
mode：存储获取的机器人循环模式 <EltCycleMode>
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_cycle_mode(ctx,&mode,&err)
```

3.4.9 获取机器人当前作业运行行号

```
int elt_get_current_job_line(ELT_CTX ctx, int *line_no, elt_error *
    err)
```

功能： 获取机器人当前作业运行行号

参数： ctx：登陆上下文
line_no：存储获取的机器人当前作业运行行号
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_current_job_line(ctx,&line_no,&err)
```

3.4.10 获取机器人当前编码器值列表

```
int elt_get_current_encode(ELT_CTX ctx, elt_robot_encode encode_array
    , elt_error *err)
```

功能： 获取机器人当前编码器值列表

参数： ctx：登陆上下文
encode_array：存储获取的机器人当前编码器值列表
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_current_code(ctx,encode_array,&err)
```

3.4.11 获取机器人当前工具号

```
int elt_get_tool_number(ELT_CTX ctx, int *tool_num, elt_error*err)
```

功能： 获取机器人当前工具号

参数： ctx：登陆上下文
tool_num：存储获取的机器人当前工具号，范围：0~7
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_tool_number(ctx,&tool_number,&err)
```

3.4.12 切换机器人当前工具号

```
int elt_change_tool_number(ELT_CTX ctx, int tool_num, elt_error *err)
```

功能： 切换机器人当前工具号

参数： ctx：登陆上下文
tool_num：工具号，范围：0~7
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_change_tool_number(ctx,1,&err)
```

注意： 本命令适用于 v2.12.0 及以上版本。

3.4.13 获取机器人当前用户工具号

```
int elt_get_user_tool_number(ELT_CTX ctx, int *tool_num, elt_error *
    err)
```

功能： 获取机器人当前用户工具号

参数： ctx：登陆上下文
user_num：存储获取的机器人当前用户工具号，范围：0~7
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_user_tool_number(ctx,&user_number,&err)
```

3.4.14 切换机器人当前用户工具号

```
int elt_change_user_tool_number(ELT_CTX ctx, int user_num, elt_error
    *err)
```

功能： 切换机器人当前用户工具号

参数： ctx：登陆上下文
user_num：用户工具号，范围：0~7
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_change_user_tool_number(ctx,1,&err)
```

注意： 本命令适用于 v2.12.0 及以上版本。

3.4.15 获取机器人当前力矩信息

```
int elt_get_robot_torques(ELT_CTX ctx, elt_robot_torquestorques,
    elt_error *err)
```

功能： 获取机器人当前力矩信息

参数： ctx：登陆上下文
torques：存储获取的机器人当前力矩信息（8 个轴的力矩），单位%(额定扭矩百分比)
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_robot_torques(ctx,torques,&err)
```

3.4.16 获取机器人当前运行点位序号

```
int elt_get_pathpoint_index(ELT_CTX ctx, int *index, elt_error *err)
```

功能： 获取机器人当前运行点位序号

参数： ctx：登陆上下文
index：存储当前运行点位序号，-1 为非路点运动
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_pathpoint_index(ctx,&index,&err)
```

注意： 本命令适用于 v2.12.0 及以上版本。

3.4.17 获取模拟量输入

```
int elt_get_analog_input(ELT_CTX ctx, int addr,
double *value, elt_error *err)
```

功能： 获取模拟量输入

参数： ctx：登陆上下文
addr：模拟量地址，范围：0~1
value：存储获取的模拟量输入，范围：-10~10
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_analog_input(ctx,0,&value,&err)
```

3.4.18 设置模拟量输出

```
int elt_set_analog_output(ELT_CTX ctx, int addr,
double value, elt_error *err)
```

功能： 设置模拟量输出

参数： ctx：登陆上下文
addr 模拟量地址，范围：0~3
value：模拟量值，范围：-10~10
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_set_analog_output(ctx,0,&value,&err)
```

3.4.19 指定坐标系

```
int elt_change_coord_mode(ELT_CTX ctx, int coord_mode,
elt_error *err)
```

功能： 指定坐标系

参数： ctx：登陆上下文
coord_mode：坐标系设置范围：0~4
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_change_coord_mode(ctx,1,&err)
```

注意： 本命令适用于 v2.12.0 及以上版本。

3.4.20 拖动示教开关

```
int elt_drag_teach(ELT_CTX ctx, int onoff, elt_error *err)
```

功能： 拖动使能开关

参数： ctx：登陆上下文
onoff：开关，范围：0~1
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_drag_teach(ctx,1,&err)
```

3.4.21 设置机械臂负载和重心

```
int elt_set_payload(ELT_CTX ctx, int tool_num, double m,
double point[], elt_error *err)
```

功能： 设置机械臂负载和重心

参数： ctx：登陆上下文
param tool_num：工具号，范围：0~7
注：不切换机器人当前工具号
m：负载重量，单位 Kg，范围：0~12
point：重心，x,y,z, 单位毫米，范围：-5000~5000
err：错误信息

返回： 成功或者失败

示例：
```
double point[3] = {20.12,-21.22,30};
ret = elt_set_payload(ctx,0,6,point,&error)
```

注意： 本命令适用于 v2.9.3 及以上版本。

3.4.22 设置机械臂工具中心

```
int elt_set_tcp(ELT_CTX ctx, int tool_num, double point[],
elt_error *err)
```

功能： 设置机械臂工具中心

参数： ctx：登陆上下文
tool_num：工具号，范围：0~7
point：工具中心，前三项单位毫米，范围：-500~500，后三项单位弧度，范围：-3.14~3.14
err：错误信息

返回： 成功或者失败

示例：
```
elt_robot_pos target_pos_array[8] =
   {0.0,-90.0,0.0,-90.0,90.0,0.0,0.0};
ret = elte_joint_move(ctx,target_pos_array,20,&err)
```

注意： 本命令适用于 v2.9.3 及以上版本。

3.5 运动服务 (MovementService)

3.5.1 关节运动

```
int elt_joint_move(ELT_CTX ctx, elt_robot_pos target_pos_array,
    double speed, elt_error *err)
```

功能：关节运动

参数：ctx：登陆上下文

target_pos_array：目标关节点

speed：运行速度百分比，范围：1~100，单位：百分比

返回：成功或者失败

示例：

```
elt_robot_pos target_pos_array[8] =
    {0.0,-90.0,0.0,-90.0,90.0,0.0,0.0};
ret = elte_joint_move(ctx,target_pos_array,20,&err)
```

3.5.2 直线运动

```
int elt_line_move(ELT_CTX ctx, elt_robot_pos target_pos_array,
double speed, elt_error *err)
```

功能：直线运动

参数：ctx：登陆上下文

target_pos_array：目标关节点

speed：运行速度，范围：直线最小速度参数值～直线最大速度参数值，单位：毫米/秒

返回：成功或者失败

示例：

```
elt_robot_pos target_pos_array[8] = {
    -32.35,-95.98,101.19,-107.17,100.89,34.54,0.0,0.0};
ret = elt_line_move(ctx,target_pos_array,100,&err)
```

3.5.3 指定坐标系下直线运动

```
int elt_line_move(ELT_CTX ctx, elt_robot_pose target_pose_array,
    double speed, elt_error *err, elt_robot_pose coord_pose)
```

功能： 指定坐标系下直线运动

参数： ctx：登陆上下文
target_pos_array：指定用户坐标系下的目标位姿
speed：运行速度，范围：直线最小速度参数值 ~ 直线最大速度参数值，单位：毫米/秒
coord_pose：用户坐标位姿

返回： 成功或者失败

示例：
```
elt_robot_pose coord_pose =
   {780.116522,132.508903,-169.680244,
0.000000,-0.000001,-1.570796};
ret = elt_line_move(ctx,target_pose,100&err,coord_pose)
```

3.5.4 圆弧运动

```
int elt_arc_move(ELT_CTX ctx, elt_robot_pos middle_target_pos_array,
   elt_robot_postarget_pos_array, double speed, elt_error *err)
```

功能： 圆弧运动

参数： ctx：登陆上下文
middle_target_pos_array：中间关节点
target_pos_array：目标关节点
speed：运行速度，范围：1~ 直线最大速度参数值，单位：毫米/秒

返回： 成功或者失败

示例：
```
ret = elt_arc_move(ctx,modile_target_pos_array,
   target_pos_array,&err)
```

3.5.5 旋转运动

```
int elt_rotate_move(ELT_CTX ctx, elt_robot_pos target_pos_array,
   double speed, elt_error *err)
```

功能： 旋转运动

参数： ctx：登陆上下文
target_pos_array：目标关节点
speed：运行速度，范围：1~ 旋转角最大速度参数值, 单位：度/秒

返回： 成功或者失败

示例：
```
ret = elt_rotate_move(ctx, target_pos_array,20,&err)
```

3.5.6 设置路点运动时最大关节速度

```
int elt_set_waypoint_max_joint_speed(ELT_CTX ctx,
double speed, elt_error *err)
```

功能： 设置路点运动时最大关节速度

参数： ctx：登陆上下文
speed：关节速度，范围：1~100，单位：百分比
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_set_waypoint_max_joint_speed(ctx,20,&err)
```

3.5.7 设置路点运动时最大直线速度

```
int elt_set_waypoint_max_line_speed(ELT_CTX ctx,
double speed, elt_error *err)
```

功能： 设置路点运动时最大直线速度

参数： ctx：登陆上下文
speed：直线速度，范围：直线最小速度参数值 ~ 直线最大速度参数值, 单位：毫米/秒
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_set_waypoint_max_line_speed(ctx,100 ,&err)
```

3.5.8 设置路点运动时最大旋转速度

```
int elt_set_waypoint_max_rotate_speed(ELT_CTX ctx,
double speed, elt_error *err)
```

功能： 设置路点运动时最大旋转速度

参数： ctx：登陆上下文
speed：运行速度，范围：1~ 旋转角最大速度参数值, 单位：度/秒
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_set_waypoint_max_rotate_speed(ctx,10,&err)
```

3.5.9 添加路点信息 2.0

```
iint elt_add_pathpoint (ELT_CTX ctx, elt_robot_pos pathpoint_array,
int moveType, double speed, int smooth, elt_error *err)
```

功能： 添加路点信息

参数： ctx：登陆上下文
pathpoint_array：目标位置（关节角）
moveType：运动类型 0, 关节运动，1：直线运动，2：绕工具尖端点旋转运动，3：圆弧运动
speed: 运动速度关节运动速度范围：1~100, 直线及圆弧速度范围：1~3000, 旋转运动速度范围：1~300
smooth：平滑度范围：0~7
acc：加速段的加速度百分比范围：1~100
dec：减速段的加速度百分比，范围：1~100
err：错误信息

返回： 成功或者失败

示例：
```
elt_robot_pos C0={-0.0034,-90.0026,
-0.0015,-90.0000,90.0000,0.0000,0.0000,0.0000}
Ret = elt_add_pathpoint (ctx,C0,0,20,7,&err)
```

3.5.10 添加路点信息 2.0（高级）

```
int elt_add_pathpoint_senior(ELT_CTX ctx,
elt_robot_pos pathpoint_array, int moveType, double speed,
int smooth, int acc, int dec, elt_error *err)
```

功能： 添加路点信息

参数： ctx：登陆上下文
pathpoint_array：目标位置（关节角）
moveType：运动类型 0, 关节运动，1：直线运动，2：绕工具尖端点旋转运动，3：圆弧运动
speed: 运动速度关节运动速度范围：1~100, 直线及圆弧速度范围：1~3000, 旋转运动速度范围：1~300
smooth：平滑度范围：0~7
acc：加速段的加速度百分比范围：1~100
dec：减速段的加速度百分比，范围：1~100
err：错误信息

返回： 成功或者失败

示例：
```
elt_robot_pos C0={-0.0034,-90.0026,
-0.0015,-90.0000,90.0000,0.0000,0.0000,0.0000}
ret=elt_add_pathpoint_senior(ctx,C0,0,30,7,50,50,&err)
```

注意： 算法暂不支持该函数使用。
本命令适用于 v2.12.0 及以上版本。

3.5.11 清除路点信息 2.0

```
int elt_clear_pathpoint(ELT_CTX ctx, elt_error *err)
```

功能： 清除路点信息

参数： ctx：登陆上下文
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_clear_pathpoint(ctx,&err)
```

3.5.12 轨迹运动 2.0

```
int elt_path_move(ELT_CTX ctx, elt_error *err)
```

功能： 轨迹运动 2.0

参数： ctx：登陆上下文
err：错误信息

返回： 点位数量或者失败

示例：
```
ret = elt_path_move(ctx,&err)
```

3.5.13 jog 运动

```
int elt_jog(ELT_CTX ctx, int index, elt_error * err)
```

功能： jog 运动

参数： ctx：登陆上下文
index：轴方向或者坐标系方向编号
err：错误信息

返回： 成功或者失败

示例：
```
elt_jog(ctx,0,&err)
```

注意： jog 运动只在示教模式下可以使用，其他模式下返回失败；
jog 运动结束之后需通过调用 elt_stop() 函数来使机器人停止；
本命令适用于 v2.12.0 及以上版本。

3.5.14 停止机器人运行

```
int elt_stop(ELT_CTX ctx, elt_error *err)
```

功能： 停止机器人运行

参数： ctx：登陆上下文
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_stop(ctx,&err)
```

3.5.15 机器人自动运行

```
int elt_run(ELT_CTX ctx, elt_error *err)
```

功能：机器人自动运行

参数：ctx：登陆上下文
err：错误信息

返回：成功或者失败

示例：
```
ret = elt_run(ctx,&err)
```

3.5.16 机器人暂停

```
int elt_pause(ELT_CTX ctx, elt_error *err)
```

功能：机器人暂停

参数：ctx：登陆上下文
err：错误信息

返回：成功或者失败

示例：
```
ret = elt_pause(ctx,&err)
```

3.5.17 检查 jbi 文件是否存在

```
int elt_check_jbi_exist(ELT_CTX ctx,char* filename,
int *state, elt_error *err)
```

功能：检查 jbi 文件是否存在

参数：ctx：登陆上下文
filename：待检查文件名
state：返回 jbi 文件是否存在，0：不存在，1：存在
err：错误信息

返回：成功或者失败

示例：
```
char filename[32] = "test.jbi";
ret = elt_check_jbi_exist(ctx, filename, &state, &err)
```

3.5.18 运行 jbi 文件

```
int elt_run_jbi(ELT_CTX ctx, char* filename, elt_error *err)
```

功能：检查 jbi 文件是否存在

参数：ctx：登陆上下文
filename：待运行文件名
err：错误信息

返回：成功或者失败

示例：
```
    ret = elt_run_jbi(ctx, filename, &err);
int elt_run_jbi(ELT_CTX ctx, char* filename, elt_error *err)
```

3.5.19 获取 jbi 文件运行状态

```
int elt_get_jbi_state(ELT_CTX ctx,char* filename,
int *state, elt_error *err)
```

功能：获取 jbi 文件运行状态

参数：ctx：登陆上下文
filename：返回当前主程序名（客户程序自行申请长度为 32 的 char 数组）
state：返回当前主程序运行状态，参见 EltJbiState 枚举变量说明。
err：错误信息

返回：成功或者失败

示例：
```
    char filename[32];
    ret = elt_get_jbi_state(ctx, filename, &state, &err)
```

3.6 运动学服务 (KinematicsService)

3.6.1 逆解函数

```
int elt_inverse_kinematic(ELT_CTX ctx,
   elt_robot_posetarget_pose_array,
elt_robot_posresponse_pose_array, elt_error *err)
```

功能： 逆解函数，根据位姿信息得到对应的机械臂关节角信息

参数： ctx：登陆上下文
target_pose_array：目标位姿信息
response_pos_array：存储获取的响应关节角信息
err：错误信息

返回： 成功或者失败

示例：
```
elt_robot_pose target_pose_array = {
   263.13,-263.96,454.96,3.09,-0.01,1.99}
ret = elt_inverse_kinematic(ctx, target_pose_array,
   response_pos_array, &err)
```

3.6.2 正解函数

```
int elt_positive_kinematic(ELT_CTX ctx,
   elt_robot_postarget_pose_array,
elt_robot_poseresponse_pose_array, elt_error *err)
```

功能： 正解函数，根据机械臂关节角信息得到对应的位姿信息

参数： ctx：登陆上下文
target_pos_array：目标关节角信息
response_pose_array：存储获取的响应位姿信息
err：错误信息

返回： 成功或者失败

示例：
```
elt_robot_pos target_pos_array
   ={62.1565,-77.00,83.96,-89.99,89.99,0.00,0.00,0.00};
ret = elt_positive_kinematic(ctx, target_pos_array,
   response_pose_array, &err)
```

3.6.3 基坐标到用户坐标位姿转化

```
int elt_base2user(ELT_CTX ctx, elt_robot_pose base_pose_array, int
   user_no, elt_robot_pose user_pose_array, elt_error *err)
```

功能： 基坐标到用户坐标位姿转化函数，当前用户坐标系下，根据基坐标的位姿信息得到对应用户坐标系下的位姿信息

参数： ctx：登陆上下文
base_pose_array：基坐标系下的位姿信息
user_no 用户坐标号，范围：0~7
user_pose_array：存储获取的用户标系下的位姿信息
err：错误信息

返回： 成功或者失败

示例：
```
elt_robot_pose base_pose_array
   ={380.968,198.847,3.0144,0.677648,0.276715};
ret = elt_base2user(ctx, base_pose_array, 0, user_pose_array,
    &err)
```

3.6.4 用户坐标到基坐标位姿转化

```
int elt_user2base(ELT_CTX ctx, elt_robot_pose user_pose_array, int
   user_no, elt_robot_pose base_pose_array, elt_error *err)
```

功能： 用户坐标到基坐标位姿转化，当前用户坐标系下，根据用户坐标的位姿信息得到对应基坐标系下的位姿信息

参数： ctx：登陆上下文
user_pose_array：用户标系下的位姿信息
user_no：用户坐标号，范围：0~7
base_pose_array：存储获取的基坐标系下的位姿信息
err：错误信息

返回： 成功或者失败

示例：
```
elt_robot_pose user_pose_array
   ={-66.6853,-390.273,221.54,0.67764,0.276715,1.5708};
ret = elt_user2base(ctx, user_pose_array, 0, base_pose_array,
    &err)
```

3.6.5 逆解函数 2.0，带参考点位置逆解

```
int elt_inverse_kinematic_ref(ELT_CTX ctx, elt_robot_pose
   target_pose_array, elt_robot_pos reference_pos_array,
   elt_robot_pos response_pos_array, elt_error *err)
```

功能： 逆解函数 2.0，带参考点位置逆解，根据位姿信息得到对应的机械臂关节角信息

参数： ctx：登陆上下文
param target_pose_array：目标位姿信息
param reference_pos_array：逆解参考点关节角信息
param response_pos_array：存储获取的响应关节角信息
err：错误信息

返回： 成功或者失败

示例：
```
elt_robot_pose target_pose_array = {
   263.13,-263.96,454.96,3.09,-0.01,1.99};
elt_robot_pos reference_pos_array =
   {260.0.-260.0,450.0,3.0,0.0,2.0};
ret = elt_inverse_kinematic_ref(ctx, target_pose_array,
   reference_pos_array, response_pos_array, &err)
```

3.6.6 位姿相乘

```
int elt_pose_mul(ELT_CTX ctx, elt_robot_pose, pose_array1,
elt_robot_pose pose_array2, elt_robot_pose response_pose_array,
   elt_error *err)
```

功能： 位姿相乘

参数： ctx：登陆上下文
pose_array1：位姿信息
pose_array2：位姿信息
response_pose_array：位姿相乘结果信息
err：错误信息

返回： 成功或者失败

示例：
```
elt_robot_pose pose1 = {200,-50,300,0.5,1,-0.1};
elt_robot_pose pose2 = {10,0,0,0,0,0};
ret = elt_pose_mul(ctx,pose1,pose2, response_pose_array,&err)
```

注意： 本命令适用于 v2.12.1 及以上版本。

3.6.7 位姿求逆

```
int elt_pose_inv(ELT_CTX ctx, elt_robot_pose pose_array,
elt_robot_pose response_pose_array, elt_error *err)
```

功能： 位姿求逆

参数： ctx：登陆上下文
pose_array：位姿信息
response_pose_array：位姿求逆结果信息
err：错误信息

返回： 成功或者失败

示例：
```
elt_robot_pose pose1 = {200,-50,300,0.5,1,-0.1};
ret = elt_pose_inv( ctx, pose1, response_pose_array, &err)
```

注意： 本命令适用于 v2.12.1 及以上版本。

3.7 IO 服务 (IOService)

3.7.1 获取输入 IO 状态

```
int elt_get_input(ELT_CTXctx, intaddr, int *status, elt_error*err)
```

功能： 获取输入 IO 状态

参数： ctx：登陆上下文
addr：输入 IO 地址，范围：0~127
status：存储获取的输入 IO 状态 <EltIOStatus>
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_input(ctx, 1, &status, &err)
```

3.7.2 获取输出 IO 状态

```
int elt_get_output(ELT_CTX ctx, intaddr, int *status, elt_error *err)
```

功能： 获取输出 IO 状态

参数： ctx：登陆上下文
addr：输出 IO 地址，范围：0~127
status：存储获取的输出 IO 状态 <EltIOStatus>
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_output(ctx, 1, &status, &err)
```

3.7.3 设置输出 IO 状态

```
int elt_set_output(ELT_CTX ctx, int addr, int status, elt_error *err)
```

功能： 设置输出 IO 状态

参数： ctx：登陆上下文
addr：输入 IO 地址，范围：0~127
status：IO 状态 <EltIOStatus>
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_set_output(ctx, 2, 1, &err)
```

3.7.4 获取虚拟输入 IO 状态

```
int elt_get_virtual_input(ELT_CTX ctx, int addr, int *status,
    elt_error *err)
```

功能： 获取虚拟输入 IO 状态

参数： ctx：登陆上下文
addr：虚拟 IO 地址，范围：0~399
status：存储获取的输入 IO 状态 <EltIOStatus>
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_virtual_input(ctx, 0, &status, &err)
```

3.7.5 获取虚拟输出 IO 状态

```
int elt_get_virtual_output(ELT_CTX ctx, int addr, int *status,
    elt_error *err)
```

功能： 获取虚拟输出 IO 状态

参数： ctx：登陆上下文
addr：虚拟 IO 地址，范围：400~1536
status：存储获取的输出 IO 状态 <EltIOStatus>
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_virtual_output(ctx, addr[i], &status, &err)
```

3.7.6 设置虚拟输出 IO 状态

```
int elt_set_virtual_output(ELT_CTX ctx, int addr, int status,
    elt_error *err)
```

功能：设置虚拟输出 IO 状态

参数：ctx：登陆上下文
addr：输出 IO 地址，范围：400~799
status：IO 状态 <EltIOStatus>
err：错误信息

返回：成功或者失败

示例：
```
ret = elt_set_virtual_output(ctx, 528, 1, &err)
```

3.8 变量服务 (VarService)

3.8.1 获取系统 B 变量值

```
int elt_get_sysvar_b(ELT_CTX ctx, int addr, int *value, elt_error *
    err)
```

功能：获取系统 B 变量值

参数：ctx：登陆上下文
addr：变量地址，范围：0~255
value：存储获取的 B 变量值，范围：0~65535
err：错误信息

返回：成功或者失败

示例：
```
ret = elt_get_sysvar_b(ctx, 0, &value, &err)
```

3.8.2 设置系统 B 变量值

```
int elt_set_sysvar_b(ELT_CTX ctx, int addr, int value, elt_error *err
    )
```

功能： 设置系统 B 变量值

参数： ctx：登陆上下文
addr：变量地址，范围：0~255
value：变量值，范围：0~2^{16}-1
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_set_sysvar_b(ctx, 0, 1, &err)
```

3.8.3 获取系统 I 变量值

```
int elt_get_sysvar_i(ELT_CTX ctx, int addr, int *value, elt_error *
    err)
```

功能： 获取系统 I 变量值

参数： ctx：登陆上下文
addr：变量地址，范围：0~255
value：存储获取的 I 变量值
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_sysvar_i(ctx, 0, &value, &err)
```

3.8.4 设置系统 I 变量值

```
int elt_set_sysvar_i(ELT_CTX ctx, int addr, int value, elt_error *err
    )
```

功能： 设置系统 I 变量值

参数： ctx：登陆上下文
addr：变量地址，范围：-2^{31}~2^{31}-1
value：变量值
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_set_sysvar_i(ctx, 0, 1, &err)
```

3.8.5 获取系统 D 变量值

```
int elt_get_sysvar_d(ELT_CTX ctx, int addr, double *value, elt_error
    *err)
```

功能： 获取系统 D 变量值

参数： ctx：登陆上下文
addr：变量地址，范围：0~255
value：存储获取的 D 变量值，范围：-3.4E+38~3.4E+38
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_sysvar_d(ctx, 0, &value, &err)
```

3.8.6 设置系统 D 变量值

```
int elt_set_sysvar_d(ELT_CTX ctx, int addr, double value, elt_error *
    err)
```

功能： 设置系统 D 变量值

参数： ctx：登陆上下文
addr：变量地址，范围：0~255
value：变量值，范围：-3.4E+38~3.4E+38
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_set_sysvar_d(ctx, 0, 1.03, &err)
```

3.8.7 获取系统 P 变量是否启用

```
int elt_get_sysvar_p_state(ELT_CTX ctx, int addr, int *state,
    elt_error *err)
```

功能： 获取系统 P 变量是否启用

参数： param ctx：登陆上下文
param addr：变量地址，范围：0~255
param state：存储 P 变量启用状态，0：未启用，1：已启用
param err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_sysvar_p_state(ctx,0,&state,&err)
```

注意： 本命令适用于 v2.11.0 及以上版本。

3.8.8 获取 P 变量的值

```
ELT_SDK_PUBLIC(int) elt_get_sysvar_p(ELT_CTX ctx, int addr,
    elt_robot_pos pos_array, elt_error *err)
```

功能： 获取系统 P 变量值，若 P 变量未启用会返回错误。

参数： ctx：登陆上下文
addr：变量地址，范围：0~255
pos_array：存储获取的 P 变量值。
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_sysvar_p(ctx,0 ,pos_array,&err)
```

注意： 本命令适用于 v2.11.0 及以上版本。

3.8.9 获取 V 变量的值

```
int elt_get_sysvar_v(ELT_CTX ctx, int addr, elt_robot_pose pose_array
    , elt_error *err)
```

功能： 获取系统 V 变量值

参数： ctx：登陆上下文
addr：变量地址，范围：0~255
pose_array：存储获取的 V 变量值。
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_sysvar_v(ctx,0,pose_array,&err)
```

注意： 本命令适用于 v2.11.0 及以上版本。

3.9 透传服务 (TransparentTransmissionService)

3.9.1 初始化透传服务

```
int elt_transparent_transmission_init(ELT_CTX ctx, double lookahead,
    double t, double smoothness, elt_error *err)
```

功能： 初始化机器人透传服务

参数： ctx：登陆上下文
lookahead：去抖窗口宽度，单位 ms，范围：10~1000，注意：此参数务必要大于 SDK 发送数据给控制器的周期
t：采样时间，单位 ms，范围：2~100，对应上位机通过 SDK 发送数据给控制器的周期
smoothness：增益，当前版本未使用，范围：0~1
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_transparent_transmission_init(ctx, 400, 10, 1, &
    error)
```

注意： 本命令适用于 v2.9.3 及以上版本。

3.9.2 设置当前透传伺服目标关节点

```
int elt_transparent_transmission_set_current_servo_joint(ELT_CTX ctx,
    elt_robot_postarget_pos, elt_error *err)
```

功能： 设置当前透传伺服目标关节点

参数： ctx：登陆上下文
target_pos：目标关节点
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_transparent_transmission_set_current_servo_joint(
    ctx,target_pos,&err)
```

注意： 本命令适用于 v2.9.3 及以上版本。

3.9.3 添加透传伺服目标关节点信息到缓存中

```
int elt_transparent_transmission_put_servo_joint_to_buffer(ELT_CTX
    ctx, unsignedintperiod, elt_robot_postarget_pos, elt_error *err)
```

功能： 添加透传伺服目标关节点信息到缓存中

参数： ctx：登陆上下文
target_pos：目标关节点
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_transparent_transmission_put_servo_joint_to_buffer(
    ctx,target_pos_array, &err)
```

注意： 本命令适用于 v2.9.3 及以上版本。

3.9.4 清空透传缓存

```
int elt_transparent_transmission_clear_servo_joint_buffer(ELT_CTX ctx
    , elt_error *err)
```

功能： 清空透传缓存

参数： ctx：登陆上下文
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_transparent_transmission_clear_servo_joint_buffer(
    ctx, &error)
```

注意： 本命令适用于 v2.9.3 及以上版本。

3.9.5 获取当前机器人是否处于透传状态

```
int elt_get_transparent_transmission_state(ELT_CTX ctx, int *state,
    elt_error *err)
```

功能： 获取当前机器人是否处于透传状态

参数： ctx：登陆上下文
state：当前透传状态。0：非透传状态，1：透传状态
err：错误信息

返回： 成功或者失败

示例：
```
ret = elt_get_transparent_transmission_state(ctx,&state,&err)
```

3.10 版本服务 (VersionService)

3.10.1 获取 SDK 版本号

```
int elt_get_sdk_version(ELT_CTX ctx, char* version, elt_error *err)
```

功能： 获取 SDK 版本号

参数： ctx：登陆上下文
version：版本号
err：错误信息

返回： 成功或者失败

示例：
```
char version[32];
ret = elt_get_sdk_version(ctx,version,&err)
```

注意： 本命令适用于 v2.12.0 及以上版本。

3.10.2 获取控制器软件版本号

```
int elt_get_soft_version(ELT_CTX ctx, char* version, elt_error *err)
```

功能： 获取控制器软件版本号

参数： ctx：登陆上下文
version：版本号
err：错误信息

返回： 成功或者失败

示例：
```
char soft_version[32];
ret = elt_get_soft_version(ctx,soft_version,&err)
```

注意： 本命令适用于 v2.12.0 及以上版本。

第 4 章 Examples

4.1 Example 1

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

ELT_CTX ctx;
int ret;

ctx=elt_create_ctx("192.168.1.188", 8055);
if (ctx==NULL){
        return;
}

printf("ctx created\n");
ret=elt_login(ctx);
printf("login: %d\n", ret);
ret=elt_logout(ctx);
printf("logout: %d\n", ret);
ret=elt_destroy_ctx(ctx);
printf("ctx destroy: %d\n", ret);
printf("Leaving...\n");
```

4.2 Example 2

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

ELT_CTX ctx;
elt_error err;
int ret;
double elt_robot_pose[ROBOT_POSE_SIZE] = {90, 0, 90, 0, 90, 0};
double response[AXIS_COUNT] = {0};

ctx=elt_create_ctx("192.168.1.188", 8055);
if (ctx==NULL){
```

```
12      return;
13  }
14  printf("ctx created\n");
15  ret=elt_login(ctx);
16  printf("login: %d\n", ret);
17  ret=elt_inverse_kinematic(ctx, target, response, &err);
18  if (ret!=ELT_SUCCESS) {
19      printf("err when getting current coord\n");
20      printf("err code: %d\n", err->code);
21      printf("err message: %s\n", err->err_msg);
22  }
23
24  else {
25      for (int i=0; i<AXIS_COUNT, i++){
26          printf("response[%d] %lf\n", i, response[i]);
27      }
28  }
29
30  ret=elt_logout(ctx);
31  printf("logout: %d\n", ret);
32  ret=elt_destroy_ctx(ctx);
33  printf("ctx destroy: %d\n", ret);
34  printf("Leaving...\n");
```

4.3 Example 3

```
1  #include <stdio.h>
2  #include "elt_robot.h"
3
4  ELT_CTX ctx;
5
6  void print_error_info(elt_error*err)
7  {
8      printf("err code: %d\n", err->code);
9      printf("err message: %s\n", err->err_msg);
10 }
11
12 void* get_robot_current_encode()
```

```
13 {
14     int ret=0;
15     elt_error err;
16 
17     for (int i=0; i<10; i++){
18         double pose_array[AXIS_COUNT] = {0.0};
19         ret=elt_get_current_encode(ctx, pose_array, &err);
20         if (ret!=ELT_SUCCESS) {
21             printf("err when getting current coord\n");
22             print_error_info(\&err);
23             break;
24         }
25         printf("robot current coord: ");
26         for (int j=0; j<AXIS_COUNT; j++)
27             printf("%lf ", pose_array[j]);
28         printf("\n");
29     }
30     return NULL;
31 }
32 
33 void* get_robot_mode()
34 {
35     int i=0;
36     int mode=0;
37     elt_error err;
38 
39     for (int i=0; i<10; i++){
40         ret=elt_get_robot_mode(ctx, &mode, &err);
41         if (ret!=ELT_SUCCESS) {
42             printf("err when getting mode\n");
43             print_error_info(\&err);
44             break;
45         }
46         printf("robot mode: %d\n", mode);
47     }
48     return NULL;
49 }
50 
51 
```

```
52 int main(int argc, char* argv[])
53 {
54     int ret;
55     ctx=elt_create_ctx("192.168.1.188", 8055);
56     if (ctx==NULL)
57         return;
58 
59     printf("ctx created\n");
60     ret=elt_login(ctx);
61     printf("login: %d\n", ret);
62     pthread_t thread1, thread2;
63     ret=pthread_create(\&thread1, NULL, (void*)get_robot_current_encode
           , NULL);
64     ret=pthread_create(\&thread2, NULL, (void*)get_robot_mode, NULL);
65     pthread_join(thread2, NULL);
66     pthread_join(thread1, NULL);
67     ret=elt_logout(ctx);
68     printf("logout: %d\n", ret);
69     ret=elt_destroy_ctx(ctx);
70     printf("ctx destroy: %d\n", ret);
71     printf("Leaving...\n");
72 }
```

ELITE ROBOT
艾利特机器人

明天比今天更简单一点

联系我们

商务合作：market@elibot.cn

技术咨询：tech@elibot.cn

苏州总部（生产研发基地）

苏州工业园区和顺路 28 号 C 栋（总部）

+86-400-189-9358

+86-0512-83951898

北京（研发中心）

北京市海淀区西小口东升科技园

上海办事处

上海市诸光路 1588 弄虹桥世界中心

深圳技术服务中心

深圳市宝安区航城街道航空路泰华梧桐岛

关注公众号了解更多