AGILE ROBOTS
思灵机器人

Connecting Artificial Intelligence
to the Physical World

PRODUCT MANUAL

思灵机器人产品手册

工业版 API 手册（C&C++版）

软件版本　V2.16.0

文档版本　V2.16.0

www.agile-robots.cn

目录

修订历史

修改内容	修订人	软件版本	修订时间
创建	孟庆婷	-	
1. 更改原 setCollision 接口函数为 setCartCollision 和 setJointCollision 两个接口函数。 2. 新增接口 getTcpForce，getJointForce，isCollision，initDHCali，getDHCaliResult，setDH，setWrd2BasRT，setFla2TcpRT，getRobotState，resume	孟庆婷	-	2020-7-14
1. 修改 moveJToTarget，moveJToPose，moveLToTarget，moveLToPose 接口函数的参数，去掉交融半径；修改 speedJ 接口函数的参数，去掉 active_tcp。 2. 修改 getRobotState，增加 free-driving 和 zero-space-free-driving 两种状态。 3. 新增接口：enterForceMode，leaveForceMode，setDefaultActiveTcpPose，setResultantCollision，setJointImpedance，getJointImpedance，setCartImpedance，getCartImpedance，zeroSpaceFreeDriving。 4. 新增多路点功能相关接口：createPath，addMoveL，addMoveJ，runPath，destroyPath。 5. 新增硬件错误码，及修改 formatError 的说明，并对 initSrv 中回调函数 fnError 的实现提出一些建议。 6. 修改 speedJ 和 speedL 的参数 t 的含义。	孟庆婷	-	2020-9-9
1. 新增四个接口：ToAxis，ToRPY，Homogeneous2Pose，Pose2Homogeneous。	孟庆婷	-	2020-9-12
1. 新增目录 2. 修改 enterForceMode 描述。 3. 新增接口：servoJ，servoL 4. 为接口函数 moveJToTarget, moveJToPose, moveLToTarget, moveLToPose, moveJ, moveL, 添加 wait_move()函数示例。	孟庆婷	-	2020-9-25
1. 新增接口：speedJ_ex，speedL_ex，servoJ_ex，servoL_ex	孟庆婷	-	2020-10-12
1. 新增接口 dumpToUDisk 2. 修改 getDHCaliResult 参数个数，原有 3 个参数，现为 4 个，增加绝对定位精度参考值数组。	孟庆婷	-	2020-10-25

1. 修改 save 接口名为 saveEnvironment	孟庆婷	-	2020-11-25
2. 新增 IO 相关接口：71-81	李漠	-	2020-11-27
1. 新增接口 getDH、getOriginalJointTorque、getJacobiM atrix 2. 修改函数 getJointTorque 含义	石国庆	-	2020-11-27
1. 新增 resetDH 2. 更新错误码	孟庆婷	-	2020-12-10
1. 修改 IO 读写接口 readDI、readAI、writeDO、writeAO、setAIMode 2. 新增接口 runProgram、stopProgram、getVariableValue、setVariableValue、isTaskRunning、pauseProgram、resumeProgram、stopAllProgram、isAnyTaskRunning、cleanErrorInfo、setCollisionLevel、mappingInt8Variant、mappingDoubleVariant、mappingInt8IO、mappingDoubleIO、setMappingAddress、lockMappingAddress、unlockMappingAddress	李漠	-	2021-03-01
新增路点变量接口：setWayPoint、getWayPoint、addWayPoint、deleteWayPoint	李漠	-	2021-04-25
新增创建复杂路径接口：createComplexPath、addMoveLByTarget、addMoveLByPose、addMoveJByTarget、addMoveJByPose、addMoveCByTarget、addMoveCByPose、runComplexPath、destroyComplexPath	孙岩祥	-	2021-04-26
1. API 支持同时控制多台机械臂	石国庆	-	2021-06-25
1. 修正了一些书写错误	石国庆	-	2021-07-15
1. 补充 API 函数及调整 API 顺序	石国庆	-	2021-09-16
1. 文档中关节个数统一用 JOINT_NUM 表示，对于 Diana 机器人 JOINT_NUM=7，对于 Thor 机器人 JOINT_NUM=6	邹艳艳	-	2021-12-03
1. 新增 readDO 函数	杨冬	-	2021-12-11
1. 添加 18 个接口说明：	孟庆婷	v2.7	2022-3-

getMechanicalJointsPositionRange、 getMechanicalMaxJointsVel、 getMechanicalMaxJointsAcc、 getMechanicalMaxCartVelAcc、 getJointsPositionRange、getMaxJointsVel、 getMaxJointsAcc、getMaxCartTranslationVel、 getMaxCartRotationVel、 getMaxCartTranslationAcc、 getMaxCartRotationAcc、 setJointsPositionRange、setMaxJointsVel、 setMaxJointsAcc、setMaxCartTranslationVel、 setMaxCartRotationVel、 setMaxCartTranslationAcc、 setMaxCartRotationAcc			24
更新 freeDriving 函数传参 添加 getJointsSoftPositionRange setJointsSoftPositionRange getFunctionOptI4 setFunctionOptI4 enterSafetyIdle leaveSafetyIdle getCartImpedanceCoordinateType setCartImpedanceCoordinateType setJointLockedInCartImpedanceMode getJointLockedInCartImpedanceMode	杨冬	v2.8	2022-04-25
修复文档中的若干错误，添加 API 线程安全对照表，阻抗参数设置函数调整为阻尼比	杜东方	v2.9	2022-07-26
updateForce 函数去掉力方向参数； 去除 updateForce_ex 函数； 修改 setCollisionLevel 函数支持组合检测级别	杨冬	v2.10	2022-08-16
修改部分专业术语； 细化运动指令中 acvitve_tcp 相关使用说明	杜东方	v2.10	2020-09-30
添加接口 getThresholdTorque,setThresholdTorque	王伊	V2.11	2022-12-19
添加接口： setHeartbeatParam getHeartbeatParam customRobotState getCustomRobotState getTcpPoseByTcpName getTcpPoseByWorkPieceName	田涛	V2.12	2023-02-08

getPayLoadByTcpName setDefaultToolTcpCoordinate setDefaultWorkPieceTcpCoordinate getDefaultTcpCoordinate getDefaultWorkPieceCoordinate 更新以下接口参数： moveJToTarget moveJToPose moveJ moveL moveLToTarget moveLToPose addMoveLSegmentByTarget addMoveLSegmentByPose addMoveJSegmentByTarget addMoveJSegmentByPose addMoveCSegmentByTarget addMoveCSegmentByPose 添加整形功能参数： zv_shaper_order zv_shaper_frequency zv_shaper_damping_ratio	王伊	V2.12	2023-02-10
开放输入整形功能，修改整形参数 zv_shaper_order 的取值范围，影响函数同上 V3.13 更新以下接口函数： setVelocityPercentValue	王伊 王伊	V2.13 V2.14	2023-03-15 2023-06-12
删除 dumpToUDiskEx，保留 dumpToUDisk，原 dumpToUDiskEx 的实现逻辑移到 dumpToUDisk 内	齐洪	V2.14	2023-09-02
添加接口： switchRescueMode 修改接口名： enterSafetyIdle 改为 enterRescueMode leaveSafetyIdle 改为 leaveRescueMode	田涛	V2.16	2023-11-21

概述

该操作库函数的所有输入输出参数，均采用国际单位，即力（N），扭矩（Nm），电流（A），长度（m），线速度（m/s），线加速度（m/s^2），角度（rad），角速度（rad/s），角加速度（rad/s^2），时间（s）。如无特殊说明，所有输入输出参数均为轴角或轴角转换的齐次矩阵。另外，文档中涉及关节个数的位置均用 JOINT_NUM 表示，针对 Diana，JOINT_NUM=7，针对 Thor，JOINT_NUM=6。

兼容性说明

操作系统	支持情况	验证用的编译器
Ubuntu 18.04 x64	推荐☆	gcc 7.5.0
Ubuntu 20.04 x64	支持	gcc 7.5.0
Ubuntu 22.04 x64	支持	gcc 7.5.0
Ubuntu 18.04 arm aarch64	支持	gcc-linaro-7.5.0
Windows 10 x64	推荐☆	MSVC 2017
Windows 11 x64	支持	MSVC 2017

函数说明

1 initSrvNetInfo

void initSrvNetInfo(srv_net_st* pInfo)
初始化机器人的网络结构体，这会使得全部端口随机分配。注：srv_net_st 结构体最好均通过 initSrvNetInfo 来初始化各个端口，然后再手动给 pInfo->SrvIp 赋值，见调用示例。 **参数：** pInfo： 网络结构体，包含 IP 地址以及所需要的全部端口号。 **返回值：** 无

调用示例：
srv_net_st* pInfo = new srv_net_st();
initSrvNetInfo(pInfo);
strcpy(pInfo->SrvIp,"192.168.10.75");
...
delete pInfo;
pInfo = nullptr;

2 initSrv

int initSrv(fnError, fnState, srv_net_st *pinfo)

初始化 API，完成其他功能函数使用前的初始化准备工作。

参数：

fnError：错误处理回调函数。函数声明形式：FNCERRORCALLBACK fnError(int e) 其中 e 为错误码（包含通信错误例如版本不匹配，链路错误例如网络断开，硬件故障例如编码器错误等），可调用 formatError 获取字符串提示信息。fnError 函数会用于多线程中实时反馈，所以尽量不要在函数实现中使用 sleep 函数之类会阻塞线程的操作。

fnState：robot state 回调函数。回调函数参数名为 StrRobotStateInfo 的结构体，包含：关节角度数组（jointPos），关节角速度数组（jointAngularVel），关节电流数组（jointCurrent），关节扭矩数组（jointTorque），TCP 位姿向量（tcpPos），TCP 外部力（tcpExternalForce），是否发生碰撞标志（bCollision），TCP 外部力是否有效标志（bTcpForceValid），TCP 六维力数组（tcpForce）和轴空间外部力数组（jointForce）。

pinfo：srv_net_st 结构体指针，用于配置本地连接服务器、心跳服务和状态反馈服务的端口号信息及服务器 IP。端口号如果传 0 则由系统自动分配。

返回值：

0：成功。

-1：失败。

调用示例：

```
#include "DianaAPIDef.h"
void logRobotState(StrRobotStateInfo *pinfo,const char *strIpAddress)
{
   strIpAddress="192.168.10.75";
   static int staCnt = 1;
   if((staCnt++ % 1000 == 0) && pinfo)
   {
      for(int i = 0; i < JOINT_NUM; ++i)
      {
         printf("jointPos[%d] = %f \n", i, pinfo->jointPos[i]);
         printf("jointCurrent [%d] = %f \n", i, pinfo-> jointCurrent [i]);
         printf("jointTorque [%d] = %f \n", i, pinfo-> jointTorque [i]);
         if(i < 6)
         {
            printf("tcpPos [%d] = %f \n", i, pinfo-> tcpPos [i]);
         }
      }
   }
```

```
}
void errorControl(int e,const char *strIpAddress )
{
  strIpAddress="192.168.10.75";
   const char * strError = formatError(e);  //该函数后面会介绍
   printf("error code (%d):%s\n", e, strError);
}

srv_net_st * pinfo = new srv_net_st();
initSrvNetInfo(pinfo);

memcpy(pinfo ->SrvIp, "192.168.10.75", strlen("192.168.10.75"));



int ret = initSrv(errorControl, logRobotState,  pinfo);
if(ret < 0)
{
   printf("192.168.10.75 initSrv failed! Return value = %d\n", ret);
}
if(pinfo)
{
 delete pinfo;
 pinfo = nullptr;
}
destroySrv(strIpAddress);
```

3 initSrvV2

int initSrvV2(fnError, fnWarning, fnState, srv_net_st *pinfo)
初始化 API，完成其他功能函数使用前的初始化准备工作。比 initSrv 多了一个 fnWarning 函数参数。 **参数：** fnError：同 initSrv。 fnWarning: 警告处理回调函数。函数声明形式 FNCWARNINGCALLBACK fnWarning(int e, const char *strIpAddress) 其中 e 为警告码，strIpAddress 为机械臂的 IP 地址字符串。fnWarning 函数会用于多线程中实时反馈，所以尽量不要在函数实现中使用 sleep 函数之

类会阻塞线程的操作。

fnState：同 initSrv。

返回值：

0：成功。

-1：失败。

调用示例：

```
#include "DianaAPIDef.h"
void logRobotState(StrRobotStateInfo *pinfo,const char *strIpAddress)
{
   strIpAddress="192.168.10.75";
   static int staCnt = 1;
   if((staCnt++ % 1000 == 0) && pinfo)
   {
      for(int i = 0; i < JOINT_NUM; ++i)
      {
         printf("jointPos[%d] = %f \n", i, pinfo->jointPos[i]);
         printf("jointCurrent [%d] = %f \n", i, pinfo-> jointCurrent [i]);
         printf("jointTorque [%d] = %f \n", i, pinfo-> jointTorque [i]);
         if(i < 6)
         {
            printf("tcpPos [%d] = %f \n", i, pinfo-> tcpPos [i]);
         }
      }
   }
}
void errorControl(int e,const char *strIpAddress )
{
  strIpAddress="192.168.10.75";
   const char * strError = formatError(e);  //该函数后面会介绍
   printf("error code (%d):%s\n", e, strError);
}

void warningControl(int w,const char *strIpAddress )
{
  strIpAddress="192.168.10.75";
   printf("error code (%d) \n", e);
}
```

```
srv_net_st * pinfo = new srv_net_st();
initSrvNetInfo(pinfo);

memcpy(pinfo ->SrvIp, "192.168.10.75", strlen("192.168.10.75"));



int ret = initSrvV2(errorControl, warningControl , logRobotState,  pinfo);
if(ret < 0)
{
   printf("192.168.10.75 initSrvV2 failed! Return value = %d\n", ret);
}
if(pinfo)
{
  delete pinfo;
  pinfo = nullptr;
}
destroySrv(strIpAddress);
```

4 destroySrv

```
int destroySrv(const char* strIpAddress = "")
```

结束调用 API，用于结束时释放指定 IP 地址机械臂的资源。如果该函数未被调用就退出系统（例如客户端程序在运行期间崩溃），服务端将因为检测不到心跳而认为客户端异常掉线，直至客户端再次运行，重新连接。除此之外不会引起严重后果。

参数：

strIpAddress:可选参数，需要释放服务资源的机械臂的 IP 地址字符串，如果为空，则会释放全部已经成功 initSrv 的机械臂的资源。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = destroySrv(strIpAddress);
if(ret < 0)
{
   printf("destroySrv failed! Return value = %d\n", ret);
```

```
}
```

5 setPushPeriod

int setPushPeriod(int intPeriod, const char *strIpAddress = "")

设置指定 IP 地址机械臂的数据推送周期

参数：

intPeriod：输入参数。推送周期，单位为 ms。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = setPushPeriod(10,strIpAddress);
if(ret < 0)
{
   printf("setPushPeriod failed! Return value = %d\n", ret);
}
```

6 moveTCP

int moveTCP(d, v, a, active_tcp=nullptr, strIpAddress = "")

手动移动指定 IP 地址的机械臂工具中心点。该函数会立即返回，停止运动需要调用 stop 函数。

参数：

d：表示移动方向的枚举类型，参考坐标系通过 active_tcp 指定，枚举及其含义如下

1. T_MOVE_X_UP 表示沿 x 轴正向；
2. T_MOVE_X_DOWN 表示沿 x 轴负向；
3. T_MOVE_Y_UP 表示沿 y 轴正向；
4. T_MOVE_Y_DOWN 表示沿 y 轴负向；
5. T_MOVE_Z_UP 表示沿 z 轴正向；
6. T_MOVE_Z_DOWN 表示沿 z 轴负向。

v：速度，单位：m/s。

a：加速度，单位：m/s^2。

active_tcp：d 参数的参考坐标系（基于法兰坐标系），大小为 6 的数组（位置和旋转矢量（轴角））；为空时将参考基坐标系。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂

时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
tcp_direction_e dtype = T_MOVE_X_UP;
double vel = 0.1;
double acc = 0.2;
const char* strIpAddress = "192.168.10.75";
int ret = moveTCP(dtype, vel, acc, nullptr, strIpAddress);
if(ret < 0)
{
   printf("moveTCP failed! Return value = %d\n", ret);
}
M_SLEEP(3000);
stop(strIpAddress);
```

7 rotationTCP

int rotationTCP(d, v, a, active_tcp=nullptr, strIpAddress = "")

使指定的 IP 地址的机械臂绕当前工具中心点变换姿态。该函数会立即返回，停止运动需要调用 stop 函数。

参数：

d：表示旋转方向的枚举类型，参考坐标系通过 active_tcp 指定，枚举及其含义如下：

1. T_MOVE_X_UP 表示绕 x 轴正向旋转；
2. T_MOVE_X_DOWN 表示绕 x 轴负向旋转；
3. T_MOVE_Y_UP 表示绕 y 轴正向旋转；
4. T_MOVE_Y_DOWN 表示绕 y 轴负向旋转；
5. T_MOVE_Z_UP 表示绕 z 轴正向旋转；
6. T_MOVE_Z_DOWN 表示绕 z 轴负向旋转。

v：速度，单位：rad/s。

a：加速度，单位：rad/s^2。

active_tcp：d 参数的参考坐标系（基于法兰坐标系），大小为 6 的数组（位置和旋转矢量（轴角）），旋转时仅参考方向，旋转中心是系统当前工具中心点；为空时使用基坐标系，旋转中心仍是系统当前工具中心点。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
tcp_direction_e dtype = T_MOVE_X_UP;
double vel = 0.1;
double acc = 0.2;
const char* strIpAddress = "192.168.10.75";
int ret = rotationTCP(dtype, vel, acc, nullptr, strIpAddress);
if(ret < 0)
{
   printf("rotationTCP failed! Return value = %d\n", ret);
}
M_SLEEP(3000);
stop(strIpAddress);
```

8 moveJoint

```
moveJoint(d, i, v, a, strIpAddress = "")
```

手动控制指定 IP 地址的机械臂关节移动。该函数会立即返回，停止运动需要调用 stop 函数。

参数：

d：表示关节移动方向的枚举类型。T_MOVE_ UP 表示关节沿正向旋转；T_MOVE_DOWN 表示关节沿负向旋转。

i：关节索引号。

v：速度，单位：rad/s。

a：加速度，单位：rad/s^2。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
joint_direction_e dtype = T_MOVE_UP;
int index = 1;
double vel = 0.8;
double acc = 0.8;
const char* strIpAddress = "192.168.10.75";
```

```
int ret = moveJoint(dtype, index, vel, acc, strIpAddress);
if(ret < 0)
{
   printf("moveJoint failed! Return value = %d\n", ret);
}
M_SLEEP(3000);
stop(strIpAddress);
```

9 moveJToTarget

```
int moveJToTarget(joints, v, a, zv_shaper_order=0, zv_shaper_frequency=0,
zv_shaper_damping_ratio=0, strIpAddress = "")
```

控制指定 IP 地址的机械臂以 JOINT_NUM 个关节角度为终点的 moveJ。该函数会立即返回，停止运动需要调用 stop 函数。

参数：

joints：终点关节角度数组首地址。

v：速度，单位：rad/s。

a：加速度，单位：rad/s^2。

zv_shaper_order，zv_shaper_frequency，zv_shaper_damping_ratio 为整形功能相关参数，函数中默认值都为 0，此时使用机械臂默认配置参数。

zv_shaper_order：阶次，整数型参数，范围[-1,2]，其中取值-1 时，关闭整形功能， 当 zv_shaper_order， zv_shaper_frequency， zv_shaper_damping_ratio 任意一值不为 0，则使用函数传入的阶次值。

zv_shaper_frequency：频率，浮点型参数，单位：Hz。支持取值范围[0，1000]，取值为 0 时，使用机械臂默认配置参数。

zv_shaper_damping_ratio：阻尼比，浮点型参数。支持取值范围[0,1]，取值为 0 时，使用机械臂默认配置参数。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
void wait_move(const char* strIpAddress)
{
   M_SLEEP(20);
   while (true)
```

```
    {
        const char state = getRobotState(strIpAddress);
        if (state != 0)
        {
            break;
        }
        else
        {
            M_SLEEP(1);
        }
    }
    stop(strIpAddress);
}
double joints[JOINT_NUM] = {0.0};
double vel = 0.2;
double acc = 0.4;
int zv_shaper_order = 0;
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
int ret = moveJToTarget(joints, vel, acc, zv_shaper_order, zv_shaper_frequency,
zv_shaper_damping_ratio, strIpAddress);
if(ret < 0)
{
    printf("moveJToTarget failed! Return value = %d\n", ret);
}
wait_move(strIpAddress);
```

10 moveJToPose

```
int moveJToPose(pose, v, a, zv_shaper_order=0, zv_shaper_frequency=0,
zv_shaper_damping_ratio=0, active_tcp=nullptr, strIpAddress = "")
```

控制指定 IP 地址的机械臂以 moveJ 的方式移动指定的工具中心点至位姿 pose。该函数会立即返回，停止运动需要调用 stop 函数。

参数：

pose：基坐标系下的终点位姿数组首地址，数组长度为 6。保存 TCP 坐标（x, y, z）和轴角（rx, ry, rz）组合的矢量数据。

v：速度，单位：rad/s。

a：加速度，单位：rad/s^2。

active_tcp：需要移动的工具中心点对应的位姿向量（基于法兰坐标系），大小为 6 的数组（位置和旋转矢量（轴角））；为空时将移动系统当前工具的中心点至 pose。（注意：此处 active_tcp 代表需要移动的工具）

zv_shaper_order，zv_shaper_frequency，zv_shaper_damping_ratio 为整形功能相关参数，函数中默认值都为 0，此时使用机械臂默认配置参数。

zv_shaper_order：阶次，整数型参数，范围[-1,2]，其中取值-1 时，关闭整形功能， 当 zv_shaper_order， zv_shaper_frequency， zv_shaper_damping_ratio 任意一值不为 0，则使用函数传入的阶次值。

zv_shaper_frequency：频率，浮点型参数，单位：Hz。支持取值范围[0，1000]，取值为 0 时，使用机械臂默认配置参数。

zv_shaper_damping_ratio：阻尼比，浮点型参数。支持取值范围[0,1]，取值为 0 时，使用机械臂默认配置参数。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double poses[6] = {0.087,0.0,1.0827,0.0,0.0,0.0};
double vel = 0.2;
double acc = 0.4;
int zv_shaper_order = 0;
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
const char* strIpAddress = "192.168.10.75";
int ret = moveJToPose(poses, vel, acc, nullptr, zv_shaper_order, zv_shaper_frequency,
zv_shaper_damping_ratio, strIpAddress);
if(ret < 0)
{
   printf("moveJToPose failed! Return value = %d\n", ret);
}
wait_move(strIpAddress);
```

11 moveJ

moveJ

宏定义，默认匹配 moveJToTarget。

参数：

同 moveJToTarget。

返回值：

0：成功。

-1：失败。

调用示例：

```
double joints[JOINT_NUM] = {0.0};
double vel = 0.2;
double acc = 0.4;
int zv_shaper_order = 0;
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
const char* strIpAddress = "192.168.10.75";
int ret = moveJ (joints, vel, acc,
zv_shaper_order,zv_shaper_frequency,zv_shaper_damping_ratio,strIpAddress);
if(ret < 0)
{
   printf("moveJ failed! Return value = %d\n", ret);
}
wait_move(strIpAddress);
```

12 moveL

moveL

宏定义，默认匹配 moveLToPose。

参数：

同 moveLToPose。

返回值：

0：成功。

-1：失败。

调用示例：

```
double poses[6] = {0.087,0.0,1.0827,0.0,0.0,0.0};
double vel = 0.2;
double acc = 0.4;
int zv_shaper_order = 0;
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
const char* strIpAddress = "192.168.10.75";
int ret = moveL (poses, vel, acc, nullptr, zv_shaper_order, zv_shaper_frequency,
zv_shaper_damping_ratio, strIpAddress);
```

```
if(ret < 0)
{
   printf("moveL failed! Return value = %d\n", ret);
}
wait_move(strIpAddress);
```

13 moveLToTarget

int moveLToTarget(joints, v, a, zv_shaper_order=0, zv_shaper_frequency=0, zv_shaper_damping_ratio=0, strIpAddress = "")

控制指定 IP 地址的机械臂以 JOINT_NUM 个关节角度为终点的 moveL。该函数会立即返回，停止运动需要调用 stop 函数。

参数：

joints：终点关节角度数组首地址，数组长度为 JOINT_NUM。

v：速度，单位：m/s。

a：加速度，单位：m/s^2。

zv_shaper_order，zv_shaper_frequency，zv_shaper_damping_ratio 为整形功能相关参数，函数中默认值都为 0，此时使用机械臂默认配置参数。

zv_shaper_order：阶次，整数型参数，范围[-1,2]，其中取值-1 时，关闭整形功能， 当 zv_shaper_order， zv_shaper_frequency， zv_shaper_damping_ratio 任意一值不为 0，则使用函数传入的阶次值。

zv_shaper_frequency：频率，浮点型参数，单位：Hz。支持取值范围[0，1000]，取值为 0 时，使用机械臂默认配置参数。

zv_shaper_damping_ratio：阻尼比，浮点型参数。支持取值范围[0,1]，取值为 0 时，使用机械臂默认配置参数。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double joints[JOINT_NUM] = {0.0};
double vel = 0.2;
double acc = 0.4;
int zv_shaper_order = 0;
```

```
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
const char* strIpAddress = "192.168.10.75";
int ret = moveLToTarget(joints, vel, acc, zv_shaper_order, zv_shaper_frequency,
zv_shaper_damping_ratio);
if(ret < 0)
{
   printf("moveLToTarget failed! Return value = %d\n", ret);
}
wait_move(strIpAddress);
```

14 moveLToPose

```
int moveLToPose(pose, v, a, zv_shaper_order=0, zv_shaper_frequency=0,
zv_shaper_damping_ratio=0, active_tcp=nullptr, strIpAddress = "")
```

控制指定 IP 地址的机械臂以 moveL 的方式移动工具中心点至位姿 pose。该函数会立即返回，停止运动需要调用 stop 函数。

参数：

pose：基坐标系下的终点位姿数组首地址，数组长度为 6，保存 TCP 坐标（x, y, z）和轴角（rx, ry, rz）组合数据。

v：速度，单位：m/s。

a：加速度，单位：m/s^2。

active_tcp：需要移动的工具中心点对应的位姿向量（基于法兰坐标系），大小为 6 的数组（位置和旋转矢量（轴角））；为空时将移动系统当前工具的中心点至 pose。（注意：此处 active_tcp 代表需要移动的工具）

zv_shaper_order，zv_shaper_frequency，zv_shaper_damping_ratio 为整形功能相关参数，函数中默认值都为 0，此时使用机械臂默认配置参数。

zv_shaper_order：阶次，整数型参数，范围[-1,2]，其中取值-1 时，关闭整形功能， 当 zv_shaper_order， zv_shaper_frequency， zv_shaper_damping_ratio 任意一值不为 0，则使用函数传入的阶次值。

zv_shaper_frequency：频率，浮点型参数，单位：Hz。支持取值范围[0，1000]，取值为 0 时，使用机械臂默认配置参数。

zv_shaper_damping_ratio：阻尼比，浮点型参数。支持取值范围[0,1]，取值为 0 时，使用机械臂默认配置参数。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂

时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
double poses[6] = {0.087,0.0,1.0827,0.0,0.0,0.0};
double vel = 0.2;
double acc = 0.4;
double radius = 0.0;
int zv_shaper_order = 0;
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
const char* strIpAddress = "192.168.10.75";
int ret = moveLToPose(poses, vel, acc, nullptr, zv_shaper_order, zv_shaper_frequency,
zv_shaper_damping_ratio, strIpAddress);
if(ret < 0)
{
printf("moveLToPose failed! Return value = %d\n", ret);
}
wait_move(strIpAddress);

15 speedJ

int speedJ(speed, a, t, strIpAddress = "")
控制指定 IP 地址的机械臂进入速度模式，关节空间运动。时间 t 为非零时，控制指定 IP 地址的机械臂将在 t 时间后减速。如果 t 为 0，机械臂将在达到目标速度时减速。该函数调用后立即返回。停止运动需要调用 stop 函数。 该函数暂不支持传送带跟踪期间使用，目前没有主动报错，使用时会卡住程序，如果使用了传送带功能请注意规避。 **参数：** speed：关节角速度数组首地址，数组长度为 JOINT_NUM。单位：rad/s。 a：加速度，单位：rad/s^2。 t：时间，单位：s。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：**

0：成功。

-1：失败。

调用示例：

```
double speeds[JOINT_NUM] = {0.0};
speeds[0]=0.2;
double acc = 0.40;



const char* strIpAddress = "192.168.10.75";
int ret = speedJ(speeds, acc, 0, strIpAddress);
if(ret < 0)
{
   printf("speedJ failed! Return value = %d\n", ret);
}
M_SLEEP(5000);
stop(strIpAddress);
```

16 speedL

int speedL(speed, a, t, active_tcp=nullptr, strIpAddress = "")

控制指定 IP 地址的机械臂进入速度模式，进行笛卡尔空间下直线运动，支持同步旋转，但笛卡尔方向必须有速度或者加速度才能旋转。时间 t 为非零时，机器人将在 t 时间后减速。如果 t 为 0，机械臂将在达到目标速度时减速。该函数调用后立即返回。停止运动需要调用 stop 函数。

该函数暂不支持传送带跟踪期间使用，目前没有主动报错，使用时会卡住程序，如果使用了传送带功能请注意规避。

参数：

speed：基坐标系下的工具空间速度（位置和旋转），数组长度为 6,其中前 3 个单位为 m/s，后 3 个单位为 rad/s。位置和旋转均参考基坐标系，旋转时的旋转中心可由 active_tcp 指定。

a：加速度数组，数组长度为 2，单位：m/s^2，rad/s^2。

t：时间，单位：s。

active_tcp：基于法兰坐标系的位姿，指定旋转时的旋转中心，大小为 6 的数组（位置+旋转矢量（轴角））；为空时旋转中心是系统当前工具中心点。（注意：机械臂做直线运动时中心点会随动，所以无旋转运动的情况下，此参数看不出影响）

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂

时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double speeds[6] = {0.1,0.0,0.0,0.0,0.0,0.0};
double acc[2] = {0.30, 0.50};
const char* strIpAddress = "192.168.10.75";
int ret = speedL(speeds, acc, 0, nullptr, strIpAddress);
if(ret < 0)
{
   printf("speedL failed! Return value = %d\n", ret);
}
```

17 freeDriving

```
int freeDriving(mode, strIpAddress = "")
```

实现控制指定 IP 地址的机械臂正常模式与零力驱动模式之间的切换。

参数：

mode：int 变量，描述零力驱动工作模式，0 表明退出零力驱动，1 为进入正常零力驱动模式。2 为进入安全零力驱动模式。安全零力驱动模式进入之前必须先利用 enterRescueMode 进入安全处理，再使用 switchRescueMode(12)激活安全零力功能，见示例。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = freeDriving(1, strIpAddress);
if(ret < 0)
{
   printf("freeDriving failed! Return value = %d\n", ret);
}
freeDriving(0, strIpAddress);
```

```
//安全零力驱动示例
enterRescueMode();
switchRescueMode(12);
releaseBrake();
freeDriving(2);
sleep(10);
freeDriving(0);
leaveRescueMode();
```

18 stop

int stop(strIpAddress = "")

控制指定 IP 地址的机械臂停止当前执行的任务。将会以最大加速度停止。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = stop(strIpAddress);
if(ret < 0)
{
   printf("stop failed! Return value = %d\n", ret);
}
```

19 forward

int forward(joints, pose, active_tcp=nullptr, strIpAddress = "")

正解函数，针对指定 IP 地址机器人，由传入的关节角计算出的正解 TCP 位姿。现支持在不同工具坐标系下求正解，由传入的 active_tcp 决定。

参数：

joints：传入关节角度数组首地址，数组长度为 JOINT_NUM。单位：rad。

pose：输出位姿数组首地址，数组长度为 6。用于传递转化后的结果，数据为包含 active_tcp 坐标（x, y, z）和旋转矢量（轴角坐标）组合。

active_tcp：工具坐标系对应的位姿向量，大小为 6 的数组，为空时，将使用默认的工具

坐标系 default_tcp。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double joints[JOINT_NUM] = {0.0};
double pose[6] = {0.0};
const char* strIpAddress = "192.168.10.75";
int ret = forward(joints, pose, nullptr, strIpAddress);
if(ret < 0)
{
   printf("forward failed! Return value = %d\n", ret);
}
else
{
    printf("forward succeed! Pose: %f, %f, %f, %f, %f, %f\n",pose[0], pose[1], pose[2], pose[3], pose[4], pose[5]);
}
```

20 inverse

int inverse(pose, joints, active_tcp=nullptr, strIpAddress = "")

逆解函数，针对指定 IP 地址机器人，通过 TCP 位姿计算出最佳逆解关节角度。现支持在不同工具坐标系下求逆解，由传入的 active_tcp 决定。

参数：

pose：输入位姿数组首地址，数据为包含 active_tcp 坐标（x, y, z）和旋转矢量（轴角坐标）组合。

joints：输出关节角度数组首地址，用于传递转换的结果。

active_tcp：工具坐标系对应的位姿向量，大小为 6 的数组，为空时，将使用默认的工具坐标系 default_tcp。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double pose[6] = {0.4221, 0.0, 0.9403, 0.0, 0.0, 0.0};
double joints[JOINT_NUM] = {0.0};
const char* strIpAddress = "192.168.10.75";
int ret = inverse(pose, joints, nullptr,strIpAddress);
if(ret < 0)
{
   printf("inverse failed! Return value = %d\n", ret);
}
else
{
    printf("inverse succeed! joints: %f, %f, %f, %f, %f, %f, %f\n",joints[0], joints[1], joints[2], joints[3], joints[4], joints[5], joints[6]);
}
```

21 getJointPos

int getJointPos(joints, strIpAddress = "")
获取指定 IP 地址机械臂各个关节角度的位置，库初始化后，后台会自动同步机器人状态信息，因此所有的监测函数都是从本地缓存取数。 **参数：** joints：输出关节角数组首地址，数组长度为 JOINT_NUM。用于传递获取到的结果。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：

```
double joints[JOINT_NUM] = {0.0};
const char* strIpAddress = "192.168.10.75";
int ret = getJointPos(joints, strIpAddress);
if(ret < 0)
{
   printf("getJointPos failed! Return value = %d\n", ret);
}
else{
   printf("getJointPos: %f, %f, %f, %f, %f, %f, %f\n", joints[0], joints[1],joints[2],joints[3],joints[4],joints[5],joints[6]);
```

```
}
```

22 getJointAngularVel

int getJointAngularVel(vels, strIpAddress = "")

获取指定 IP 地址机械臂当前各关节的角速度。

参数：

vels：输出关节角速度数组首地址，数组长度为 JOINT_NUM。用于传递获取到的结果。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double speeds[JOINT_NUM] = {0.0};
const char* strIpAddress = "192.168.10.75";
int ret = getJointAngularVel(speeds,strIpAddress);
if(ret < 0)
{
   printf("getJointAngularVel failed! Return value = %d\n", ret);
}
else{
    printf("getJointAngularVel: %f, %f, %f, %f, %f, %f, %f\n", speeds[0], speeds[1],speeds[2],speeds[3],speeds[4],speeds[5],speeds[6]);
}
```

23 getJointCurrent

int getJointCurrent(joints, strIpAddress = "")

获取当前关节电流。

参数：

joints：输出关节电流数组首地址，数组长度为 JOINT_NUM。用于传递转获取到的结果。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double joints[JOINT_NUM] = {0.0};
strIpAddress="192.168.10.75"
int ret = getJointCurrent(joints, strIpAddress);
if(ret < 0)
{
   printf("getJointCurrent failed! Return value = %d\n", ret);
}
else
{
   printf("getJointCurrent: %f, %f, %f, %f, %f, %f, %f\n", joints[0], joints[1],joints[2],joints[3],joints[4],joints[5],joints[6]);
}
```

24 getJointTorque

```
int getJointTorque(torques, strIpAddress = "")
```

获取指定 IP 地址机械臂各关节真实扭矩数据，即减去零偏的数据

参数：

torques：输出关节扭矩数组首地址，数组长度为 JOINT_NUM。用于传递获取到的结果。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double torques[JOINT_NUM] = {0.0};
const char* strIpAddress = "192.168.10.75";
int ret = getJointTorque(torques, strIpAddress);
if(ret < 0)
{
   printf("getJointTorque failed! Return value = %d\n", ret);
}
else
{
   printf("getJointTorque: %f, %f, %f, %f, %f, %f, %f\n", torques[0], torques[1],torques[2],torques[3],torques[4],torques[5],torques[6]);
```

```
}
```

25 getTcpPos

int getTcpPos(pose, strIpAddress = "")

获取指定 IP 地址机械臂当前 TCP 位姿数据，TCP 位姿可被 setDefaultActiveTcp 和 setDefaultActiveTcpPose 函数改变。

参数：

pose：输出 TCP 位姿数组首地址，数组长度为 6。用于传递获取到的结果。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double poses[6] = {0.0};
const char* strIpAddress = "192.168.10.75";
int ret = getTcpPos(poses, strIpAddress);
if(ret < 0)
{
   printf("getTcpPos failed! Return value = %d\n", ret);
}
else
{
   printf("getTcpPos: %f, %f, %f, %f, %f, %f\n", poses[0], poses[1],poses[2],poses[3],poses[4],poses[5]);
}
```

26 getTcpExternalForce

double getTcpExternalForce(strIpAddress = "")

获取指定 IP 地址机械臂 TCP 实际感受到的合力大小，TCP 位姿可被 setDefaultActiveTcp 函数改变。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

返回力的大小。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
double force = getTcpExternalForce(strIpAddress);
if(ret < 0)
{
    printf("getTcpExternalForce failed! Return value = %d\n", ret);
}
else
{
    printf("getTcpExternalForce: %f\n", force);
}
```

27 releaseBrake

int releaseBrake(strIpAddress = "")

打开指定 IP 地址机械臂的抱闸，启动机械臂。调用该接口后，需要调用者延时 2s 后再做其他操作。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = releaseBrake (strIpAddress);
if(ret < 0)
{
   printf("releaseBrake failed! Return value = %d\n", ret);
}
M_SLEEP(2000);
```

28 holdBrake

int holdBrake(strIpAddress = "")

关闭指定 IP 地址机械臂的抱闸，停止机械臂。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = holdBrake (strIpAddress);
if(ret < 0)
{
   printf("holdBrake failed! Return value = %d\n", ret);
}
```

29 changeControlMode

int changeControlMode(m, strIpAddress = "")

控制指定 IP 地址机械臂的模式切换。

参数：

m：枚举类型。

1. T_MODE_INVALID 无意义；
2. T_MODE_POSITION 位置模式；
3. T_MODE_JOINT_IMPEDANCE 关节空间阻抗模式；
4. T_MODE_CART_IMPEDANCE 笛卡尔空间阻抗模式。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = changeControlMode(T_MODE_POSITION, strIpAddress);
if(ret < 0)
{
   printf("changeControlMode failed! Return value = %d\n", ret);
}
```

30 getLibraryVersion

unsigned short getLibraryVersion()

获取当前库的版本号。

参数：

无。

返回值：

当前版本号,高 8 位为主版本号，低 8 位为次版本号。

调用示例：

```
unsigned short uVersion = getLibraryVersion();
```

31 formatError

```
const char *formatError(e, const char* strIpAddress = "")
```

获取指定 IP 地址机械臂的错误码 e 的字符串描述，该错误码在初始化指定的回调函数中会作为形参传入，也可以在函数调用失败后查询得到。对于错误码为-2001 的硬件错误，会延时回馈，一般建议对此类错误延时 100 毫秒后调用 formatError 函数获取具体硬件错误提示信息，否则将提示"refresh later ..."而看不到具体内容。

参数：

e：错误码。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

错误描述信息。

调用示例：

```
int e = -1003;
const char* msg = formatError(e,"192.168.10.75");
printf("error [%d] : %s\n",e, msg);
```

32 getLastError

```
int getLastError(const char* strIpAddress = "")
```

返回指定 IP 地址机械臂最近发生的错误码。该错误码会一直保存，确保可以查询得到，直至库卸载，因此，当库函数调用失败后，如果想知道具体的错误原因，应该调用该函数获取错误码。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：没有错误。

其它值：具体错误码。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int e = getLastError(strIpAddress);
printf("getLastError code [%d] \n",e);
```

33 setLastError

```
int setLastError(int e, const char* strIpAddress = "")
```

重置指定 IP 地址机械臂错误码。将系统中记录的错误码重置为 e，通常用于清除错误。

参数：

e：错误码。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

错误码。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int e = setLastError(0, strIpAddress);
printf("setLastError code [%d] \n",e);
```

34 getLastWarning

```
int getLastWarning(const char* strIpAddress = "")
```

返回指定 IP 地址机械臂最近发生的警告码。该警告码会一直保存，确保可以查询得到，直至库卸载，因此，当库函数调用失败后，如果想知道具体的警告原因，应该调用该函数获取警告码。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：没有警告。

其它值：具体警告码。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int w = getLastWarning(strIpAddress);
printf("getLastWarning code [%d] \n",w);
```

35 setLastWarning

int setLastWarning(int w, const char* strIpAddress = "")
重置指定 IP 地址机械臂警告码。将系统中记录的警告码重置为 w，通常用于警告错误。 **参数：** w：警告码。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 警告码。

调用示例：
const char* strIpAddress = "192.168.10.75";
int w = setLastWarning(0, strIpAddress);
printf("setLastWarning code [%d] \n",w);

36 setDefaultActiveTcp

int setDefaultActiveTcp(default_tcp , strIpAddress = "")
设置指定 IP 地址字符串的默认工具坐标系。在没有调用该函数时，默认工具中心点为法兰盘中心，调用该函数后，默认的工具坐标系将被改变。该函数将会改变 moveTCP，rotationTCP，moveJToPos，moveLToPose，speedJ，speedL，forward，inverse，getTcpPos，getTcpExternalForce 的默认行为。 **参数：** default_tcp：输入参数，TCP 相对于末端法兰盘的 4*4 齐次变换矩阵的首地址，数组长度为 16。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
const char* strIpAddress = "192.168.10.75";
int ret = setDefaultActiveTcp(nullptr, strIpAddress);
if(ret < 0)
{
printf("setDefaultActiveTcp failed! Return value = %d\n", ret);
}

37 getLinkState

int getLinkState(strIpAddress = "")
获取与指定 IP 地址机械臂间的链路状态。 **参数：** strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：链路正常。 -1：链路断开。

调用示例：
const char* strIpAddress = "192.168.10.75";
int ret = getLinkState(strIpAddress);
if(ret == 0)
{
printf("network connected \n");
}
else
{
printf("network disconnected \n");
}

38 getTcpForce

int getTcpForce(forces, strIpAddress = "")
获取指定 IP 地址机械臂的 TCP 所受外部六维力，TCP 位姿可被 setDefaultActiveTcp 函数改变。 **参数：** forces：工具中心点处六维力数组的首地址，数组长度为 6。用于传递获取到的结果。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0： 获取到六维力且六维力有效。 -1： 获取不到六维力矩，或六维力数值无效（发生奇异）。

调用示例：
double tcpForce[6];
const char* strIpAddress = "192.168.10.75";
int ret = getTcpForce(tcpForce,strIpAddress);
if(ret == 0)

```
{
printf("get tcp force: %f, %f, %f, %f, %f, %f \n", tcpForce[0] , tcpForce[1] , tcpForce[2] , tcpForce[3] , tcpForce[4] , tcpForce[5] );
}
else
{
printf("get tcp force failed \n");
}
```

39 getJointForce

int getJointForce(forces, strIpAddress = "")

获取指定 IP 地址机械臂的轴空间 JOINT_NUM 个关节所受外力。

参数：

forces：关节轴力矩数组的首地址，数组长度为 JOINT_NUM。用于传递获取到的结果。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：获取成功。

-1：获取失败。

调用示例：

```
double jointForce[JOINT_NUM];
const char* strIpAddress = "192.168.10.75";
int ret = getJointForce (jointForce, strIpAddress);
if(ret < 0)
{
printf("get joint force failed \n");
}
else
{
printf("get joint force: %f, %f, %f, %f, %f, %f, %f \n", jointForce[0] , jointForce[1] , jointForce[2] , jointForce[3] , jointForce[4] , jointForce[5], jointForce[6] );
}
```

40 isCollision

bool isCollision(strIpAddress = "")

从轴空间判断指定 IP 地址机械臂是否发生碰撞。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

true：机器人发生碰撞。

false：机器人未发生碰撞。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
if(isCollision(strIpAddress))
{
    printf("Warning: Robot got collision\n");
}
else
{
    printf("Robot is running\n");
}
```

41 initDHCali

int initDHCali(tcpMeas, jntPosMeas, nrSets, strIpAddress = "")

根据输入的关节角以及 TCP 位置数组计算指定 IP 地址机械臂的 DH 参数。

参数：

tcpMeas：输入参数。TCP 位置数组的首地址，数组长度为 3 * nrSets。每组数据为[x,y,z]，共 nrSets 组。单位：米。

jntPosMeas：输入参数。关节角位置数组的首地址，数组长度为 JOINT_NUM * nrSets，每组数据为各关节角位置信息，共 nrSets 组。单位：弧度。

nrSets：输入参数。测量样本数量，最少 32 组，至少保证大于或等于需要辨识的参数。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：正常。

-1：失败。

调用示例：

```
int rowNo_refData = 32
double jntMeas[JOINT_NUM*32] = {…};
double tcpMeas[96] = {…};
const char* strIpAddress = "192.168.10.75";
if (initDHCali(tcpMeas, jntMeas, rowNo_refData, strIpAddress) != 0)
```

```
{
   printf("DHCaliFcns.InitDHCali falid!\n");
}
```

42 getDHCaliResult

int getDHCaliResult(rDH, wRT, tRT, confid, strIpAddress = "")

获取指定 IP 地址机械臂 DH 参数的计算结果。

参数：

rDH：输出参数。机器人各关节 DH 参数数组的首地址，数组长度为 4*JOINT_NUM。每 JOINT_NUM 个数为一组，共四组数据[a，alpha，d，theta]。单位：rad、m。

wRT：输出参数。机器人基坐标系相对于世界坐标系下的位姿数组的首地址，数组长度为 6。位姿数据[x, y, z, Rx，Ry，Rz]。单位：rad、m。

tRT： 输出参数。靶球在法兰坐标系下的位置描述数组的首地址，数组长度为 3。数组为靶球位置坐标[x,y,z]。单位：m。

confid：输出参数。绝对定位精度参考值数组的首地址，数组长度为 2。其中，第一个值为标定前绝对定位精度，第二个值为标定后绝对定位精度。单位：m。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

该 API 接口一般不能直接调用，需要配合 initDHCali 接口使用，直接调用会失败。

返回值：

0：获取成功。

1：获取结果精度可能够较低。

-1：获取失败，发生异常。

调用示例：

```
double rDH[4*JOINT_NUM], wRT[6], tRT[3], confid[2];
const char* strIpAddress = "192.168.10.75";
if (getDHCaliResult(rDH, wRT, tRT, confid, strIpAddress) < 0)
{
   printf("DHCaliFcns.getDHCaliResult falid.\n");
}
```

43 setDH

int setDH(a, alpha, d, theta, strIpAddress = "")

设置指定 IP 地址机械臂当前 DH 参数。特别注意，**错误的参数设置**可能引起机器人损坏，需谨慎设置！

参数：

a：输入参数。各关节的 a 参数数组的首地址，数组长度为 JOINT_NUM。

alpha：输入参数。各关节的 alpha 参数数组的首地址，数组长度为 JOINT_NUM。

d：输入参数。各关节的 d 参数数组的首地址，数组长度为 JOINT_NUM。

theta：输入参数。各关节的 theta 参数数组的首地址，数组长度为 JOINT_NUM。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double a[JOINT_NUM], alpha[JOINT_NUM], d[JOINT_NUM], theta[JOINT_NUM];
const char* strIpAddress = "192.168.10.75";
if (setDH(a, alpha, d, theta, strIpAddress) < 0)
{
    printf("DHCaliFcns.setDH falid.\n");
}
```

44 setWrd2BasRT

int setWrd2BasRT(RTw2b, strIpAddress = "")

初始化世界坐标系到指定 IP 地址机械臂坐标系的平移和旋转位姿。用于 DH 参数标定前设置，若用户不能提供此参数，DH 参数标定功能依旧可以使用。如果调用此函数则使用用户自定义的位姿。特别注意，此功能每次移动机器人与激光跟踪仪都需要重新计算，使用错误的参数可能引起 DH 参数计算不准确或标定异常。

参数：

RTw2b：输入参数。世界坐标系到机器人坐标系的平移和旋转位姿数组的首地址，数组长度为 6。单位：米和弧度。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double wRT[6]  = {…}
const char* strIpAddress = "192.168.10.75";
if (setWrd2BasRT(wRT, strIpAddress) < 0)
```

```
{
    printf("DHCaliFcns. setWrd2BasRT falid.\n");
}
```

45 setFLa2TcpRT

int setFla2TcpRT(RTf2t, strIpAddress = "")

初始化指定 IP 地址机械臂法兰坐标系到工具坐标系的平移位置。用于 DH 参数标定前设置，若用户不能提供此参数，DH 参数标定功能依旧可以使用。如果调用此函数则使用用户自定义的位姿。特别注意，此功能每次移动机器人与激光跟踪仪都需要重新计算，使用错误的参数可能引起 DH 参数计算不准确或标定异常。

参数：

RTf2t：输入参数。初始化法兰坐标系到工具坐标系的平移位置数组的首地址，数组长度为 3，位置信息数据[x,y,z]。单位：米。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double Fla[3]  = {…}
const char* strIpAddress = "192.168.10.75";
if (setFla2TcpRT (Fla, strIpAddress) < 0)
{
    printf("DHCaliFcns. setFla2TcpRT falid.\n");
}
```

46 getRobotState

const char getRobotState(strIpAddress = "")

获取指定 IP 地址机械臂当前工作状态。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：running（机械臂正在运动或者示教器程序正在运行）

1：paused（暂停正在运动的机械臂或者暂停正在运行的示教器程序）

2：idle（空闲状态）

3：free-driving（进入正常零力驱动模式） 4：zero-space-free-driving（进入零空间零力驱动模式） 5：hold-brake（机械臂关闭抱闸） 6：error（机械臂发生错误） 7：handling-error（机械臂的关节超出极限位置） 8：rescueMode（安全处理模式）

调用示例：
const char* strIpAddress = "192.168.10.75";
const char state = getRobotState(strIpAddress);
if (0 == state)
{
printf("\t[robot state]:running\n");
}
else if (1 == state)
{
printf("\t[robot state]:paused\n");
}
else if (2 == state)
{
printf("\t[robot state]:idle\n");
}
else
{
printf("\t[robot state]: unknown state \n");
}

47 resume

int resume(strIpAddress = "")
当指定 IP 地址机械臂发生碰撞或其他原因暂停后，恢复运行时使用。 **参数：** strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。
调用示例：

```
const char* strIpAddress = "192.168.10.75";
if (resume(strIpAddress) < 0)
{
   printf("Diana API resume failed!\n");
}
```

48 setJointCollision

int setJointCollision(collision,strIpAddress = "")

设置指定 IP 地址机械臂关节空间碰撞检测的力矩阈值。

参数：

collision：输入参数。JOINT_NUM 个关节轴力矩阈值数组的首地址，数组长度为 JOINT_NUM。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：设置成功。

-1：设置失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
double collision[7] = {200, 200, 200, 200, 200, 200, 200};//以 7 轴机器人为例
if (setJointCollision(collision, strIpAddress) < 0)
{
   printf("Diana API setJointCollision failed!\n");
}
```

49 setCartCollision

int setCartCollision(collision,strIpAddress = "")

设置指定 IP 地址机械臂笛卡尔空间碰撞检测的六维力阈值。

参数：

collision：输入参数。在基坐标系下的六维力数组的首地址，数组长度为 6。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：设置成功。

-1：设置失败。

调用示例：

```
double collision[6] = { 200, 200, 200, 200, 200, 200};
const char* strIpAddress = "192.168.10.75";
if (setCartCollision (collision, strIpAddress) < 0)
{
   printf("Diana API setCartCollision failed!\n");
}
```

50 enterForceMode

int enterForceMode(intFrameType, dblFrameMatrix, dblForceDirection, dblForceValue, dblMaxApproachVelocity, dblMaxAllowTcpOffset,strIpAddress = "")

使指定 IP 地址机械臂进入力控模式。

参数：

intFrameType：参考坐标系类型。0：基坐标系；1：工具坐标系；2：自定义坐标系（暂不支持）。

dblFrameMatrix：自定义坐标系矩阵（暂不支持），使用时传单位矩阵即可。

dblForceDirection：力指令的方向的数组首地址，数组长度为 3。

dblForceValue：力指令的大小。单位：N。

dblMaxApproachVelocity：最大接近速度。单位：m/s。推荐取值范围（0.01m/s~0.5m/s）。

dblMaxAllowTcpOffset：允许的最大偏移。单位：m。推荐取值范围（0.01m~1m）。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
int iFrameType = 1;
double dblFrameMatrix[16] = {1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1};
double dblForceDir[3]={0,0,-1};
double dblForceValue = 2.0;
double dblMaxVel = 0.1;
double dblMaxOffset = 0.2;
const char* strIpAddress = "192.168.10.75";
if (enterForceMode(iFrameType, dblFrameMatrix, dblForceDir, dblForceValue, dblMaxVel,
dblMaxOffset,strIpAddress) < 0)
{
```

```
    printf("Diana API enterForceMode failed!\n");
}
```

51 leaveForceMode

int leaveForceMode (intExitMode,strIpAddress = "")

设置指定 IP 地址机械臂退出力控模式,并设置退出后机械臂的工作模式。

参数：

intExitMode：控制模式。

0：代表位置模式；

1：代表关节空间阻抗模式；

2：代表笛卡尔空间阻抗模式。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
int intExitMode = 0;
const char* strIpAddress = "192.168.10.75";
if (leaveForceMode (intExitMode, strIpAddress) < 0)
{
    printf("Diana API leaveForceMode failed!\n");
}
```

52 setDefaultActiveTcpPose

int setDefaultActiveTcpPose (arrPose,strIpAddress = "")

设置指定 IP 地址机械臂默认的工具坐标系。在没有调用该函数时，默认工具中心点为法兰盘中心，调用该函数后，默认的工具坐标系将被改变。该函数将会改变 moveTCP，rotationTCP，moveJToPos，moveLToPose，speedJ，speedL，forward，inverse，getTcpPos，getTcpExternalForce 的默认行为。调用该函数后，客户端生效有一个推送周期的延迟，所以当想要获取到最新工具坐标系下的位姿信息，需要延迟一个推送周期的时间。

参数：

arrPose：输入参数。TCP 相对于末端法兰盘的位姿向量的首地址，数组长度为 6，其中，后三个角度为轴角

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂

时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
double pose[6] = {0.1,0.1,0.1,0, 0, 0};
const char* strIpAddress = "192.168.10.75";
if (setDefaultActiveTcpPose (pose,strIpAddress) < 0)
{
printf("Diana API setDefaultActiveTcpPose failed!\n");
}

53 setResultantCollision

int setResultantCollision (force,strIpAddress = "")
设置指定 IP 地址机械臂笛卡尔空间碰撞检测 TCP 的合力阈值。 **参数：** force：合力值。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
double force = 8.9;
const char* strIpAddress = "192.168.10.75";
if (setResultantCollision (force, strIpAddress) < 0)
{
printf("Diana API setResultantCollision failed!\n");
}

54 setJointImpeda

int setJointImpeda (arrStiff, dblDamp,strIpAddress = "")
设置指定 IP 地址机械臂各关节阻抗参数，包含刚度 Stiffiness 和阻尼比 DampingRatio 的数据。 **参数：** arrStiff：表示各关节刚度 Stiffiness 的数组的首地址，数组长度为 JOINT_NUM。

dblDamp：表示关节阻尼比 DampingRatio。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double arrStiff[7] = { 3000, 3000, 3000, 1000, 500, 1000, 1000};
double dblDamp = 0.7;
const char* strIpAddress = "192.168.10.75";
if (setJointImpeda (arrStiff, dblDamp, strIpAddress) < 0)
{
   printf("Diana API setJointImpeda failed!\n");
}
```

55 getJointImpeda

```
int getJointImpeda(arrStiff, dblDamp,strIpAddress = "")
```

获取指定 IP 地址机械臂各关节阻抗参数，包含刚度 Stiffiness 和阻尼比 DampingRatio 的数据。

参数：

arrStiff：表示各关节刚度 Stiffiness 的数组的首地址，数组长度为 JOINT_NUM，用于接收获取到的值。

dblDamp：表示关节阻尼比 DampingRatio。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double arrStiff [JOINT_NUM] = { 0};
double dblDamp=  0;
const char* strIpAddress = "192.168.10.75";
if (getJointImpeda (arrStiff, &dblDamp, strIpAddress) < 0)
{
   printf("Diana API getJointImpeda failed!\n");
}
```

```
else
{
   printf("getJointImpeda : Stiff=%f, %f, %f, %f, %f, %f, %f\n Damp=%f \n",
   arrStiff[0], arrStiff[1], arrStiff[2], arrStiff[3], arrStiff[4], arrStiff[5], arrStiff[6],
   dblDamp);
}
```

56 setCartImpeda

int setCartImpeda (arrStiff, dblDamp, strIpAddress = "")

设置指定 IP 地址机械臂笛卡尔空间阻抗参数。

参数：

arrStiff：表示笛卡尔空间，各维度刚度 Stiffiness 的数组的首地址，数组长度为 6。

dblDamp：表示笛卡尔空间的阻尼比 DampingRatio。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double arrStiff[6] = { 1000, 1000, 1000, 500, 500, 500};
double dblDamp = 0.7;
const char* strIpAddress = "192.168.10.75";
if (setCartImpeda(arrStiff, dblDamp, strIpAddress) < 0)
{
   printf("Diana API setCartImpeda failed!\n");
}
```

57 getCartImpeda

int getCartImpeda (arrStiff, dblDamp, strIpAddress = "")

获取指定 IP 地址机械臂笛卡尔空间各维度阻抗参数，包含刚度 Stiffiness 和阻尼比 DampingRatio 的数据。

参数：

arrStiff：表示笛卡尔空间，各维度刚度 Stiffiness 的数组的首地址，数组长度为 6，用于接收获取到的值。

dblDamp：表示笛卡尔空间的阻尼比 DampingRatio。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂

时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double arrStiff [6] = {0};
double dblDamp = 0;
const char* strIpAddress = "192.168.10.75";
if (getCartImpeda (arrStiff, &dblDamp, strIpAddress) < 0)
{
   printf("Diana API getCartImpeda failed!\n");
}
else
{
   printf("getCartImpeda: Stiff=%f, %f, %f, %f, %f, %f\n Damp=%f \n",
   arrStiff[0], arrStiff[1], arrStiff[2], arrStiff[3], arrStiff[4], arrStiff[5],
   dblDamp);
}
```

58 zeroSpaceFreeDriving

int zeroSpaceFreeDriving (enable, strIpAddress = "")

控制指定 IP 地址机械臂进入或退出零空间自由驱动模式。

参数：

enable：输入参数。

true 为进入零空间自由驱动模式；

false 为退出零空间自由驱动模式。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
if (zeroSpaceFreeDriving (true, strIpAddress) < 0)
{
   printf("Diana API zeroSpaceFreeDriving failed!\n");
```

```
}
else
{
   printf("Diana API zeroSpaceFreeDriving succeed!\n");
    M_SLEEP(5000);
   zeroSpaceFreeDriving(false, strIpAddress);
}
```

59 createPath

int createPath (type, id_path, strIpAddress = "")

为指定 IP 地址机械臂创建一个路段。

参数：

type：输入参数。1 :表示 moveJ, 2：表示 moveL。

id_path：输出参数。用于保存新创建 Path 的 ID。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

详见 addMoveJ 或 addMoveL 调用示例。

60 addMoveL

int addMoveL (id_path, joints, vel, acc, blendradius, strIpAddress = "")

向指定 IP 地址机械臂已创建的路段添加 MoveL 路点。

参数：

id_path：输入参数。要添加路点的路径 ID。

joints：输入参数。要添加的路点，即该路点的各关节角度数组的首地址，数组长度为 JOINT_NUM。单位：rad。

vel：moveL 移动到目标路点的速度。单位：m/s。

acc：moveL 移动到目标路点的加速度。单位：m/s^2。

blendradius _percent：交融半径。单位：m。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
#define PI 3.14159265358979323846
#define DEGREE_TO_RAD(x) ((x)*PI / 180.0)
#define RAD_TO_DEGREE(x) ((x)*180.0 / PI)

double dblFirstPosition[7] = { DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(90.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(-90.0), DEGREE_TO_RAD(0.0) };
double dblSecondPosition[7] = { DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(120.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(-60.0), DEGREE_TO_RAD(0.0) };
double dblThirdPosition[7] = { DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(150.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(-30.0), DEGREE_TO_RAD(0.0) };

unsigned int path_id = 0;
printf("start test moveJ path.\n");
const char* strIpAddress = "192.168.10.75";
createPath(2, path_id,strIpAddress);
addMoveL(path_id, dblFirstMoveLPosition, 0.2, 0.2, 0.3,strIpAddress);
addMoveL(path_id, dblSecondMoveLPosition, 0.2, 0.2, 0.3,strIpAddress);
addMoveL(path_id, dblThirdMoveLPosition, 0.2, 0.2, 0.3,strIpAddress);
runPath(path_id,strIpAddress);
destroyPath(path_id,strIpAddress);
wait_move(strIpAddress);
```

61 addMoveJ

int addMoveJ (id_path, joints, vel_percent, acc_percent, blendradius_percent, strIpAddress = "")

向指定 IP 地址机械臂已创建的路段添加 MoveJ 路点。

参数：

id_path：输入参数。要添加路点的路径 ID。

joints：输入参数。要添加的路点，即该路点的各关节角度数组的首地址，数组长度为 JOINT_NUM。单位：rad。

vel_percent：moveJ 移动到目标路点的速度百分比。

acc_percent：moveJ 移动到目标路点的加速度百分比。

blendradius _percent：交融半径百分比。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
#define PI 3.14159265358979323846
#define DEGREE_TO_RAD(x) ((x)*PI / 180.0)
#define RAD_TO_DEGREE(x) ((x)*180.0 / PI)

double dblFirstPosition[7] = { DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(90.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(-90.0), DEGREE_TO_RAD(0.0) };
double dblSecondPosition[7] = { DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(120.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(-60.0), DEGREE_TO_RAD(0.0) };
double dblThirdPosition[7] = { DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(150.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(-30.0), DEGREE_TO_RAD(0.0) };

unsigned int path_id = 0;
printf("start test moveJ path.\n");
const char* strIpAddress = "192.168.10.75";
createPath(1, path_id,strIpAddress);
addMoveJ(path_id, dblFirstPosition, 0.2, 0.2, 0.3,strIpAddress);
addMoveJ(path_id, dblSecondPosition, 0.2, 0.2, 0.3,strIpAddress);
addMoveJ(path_id, dblThirdPosition, 0.2, 0.2, 0.3,strIpAddress);
runPath(path_id,strIpAddress);
destroyPath(path_id,strIpAddress);
wait_move(strIpAddress);
```

62 runPath

int runPath (id_path, strIpAddress = "")

为指定 IP 地址机械臂启动运行设置好的路段。

参数：

id_path：输入参数。要运行的路径 ID。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

详见 addMoveJ 或 addMoveL 调用示例。

63 destroyPath

int destroyPath (id_path, strIpAddress = "")

销毁指定 IP 地址机械臂某个路段。

参数：

id_path：输入参数。要销毁的路径 ID。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

详见 addMoveJ 或 addMoveL 调用示例。

64 rpy2Axis

int rpy2Axis (arr)

欧拉角转轴角。

参数：

arr：输入参数。欧拉角数组的首地址，数组长度为 3。

返回值：

0：成功。

-1：失败。

调用示例：

```
const double PI= 3.1415926;
double rpy[3]={PI, PI/3, PI/6};
int ret = rpy2Axis(rpy);
```

printf("Diana API rpy2Axis got: %f, %f, %f\n", rpy[0] , rpy[1] , rpy[2]);

输出结果：2.431323, 0.651471, -1.403725

65 axis2RPY

int axis2RPY (arr)

轴角转欧拉角。

参数：

arr：输入参数。轴角数组的首地址，数组长度为 3。

返回值：

0：成功。

-1：失败。

调用示例：

```
const double PI= 3.1415926;
double rpy[3]={2.431323, 0.651471, -1.403725};
ret = axis2RPY(rpy);
printf("Diana API axis2Rpy got: %f, %f, %f\n", rpy[0] , rpy[1] , rpy[2]);
```

输出结果： -3.141593, 1.047198, 0.523599

66 homogeneous2Pose

int homogeneous2Pose (matrix, pose)

齐次变换矩阵转位姿。

参数：

matrix：齐次变换矩阵数组的首地址，数组长度为 16。

pose：位姿数组的首地址，数组长度为 6。

返回值：

0：成功。

-1：失败。

调用示例：

```
double pose[6]={-0.231, 0.155, 0.934, PI, PI/3, PI/6};
double matrix[16]={ 0.433013, 0.250000, -0.866025, 0.000000, 0.500000, -0.866025, -
0.000000, 0.000000, -0.750000, -0.433013, -0.500000, 0.000000, -0.231000, 0.155000,
0.934000, 1.000000};
int ret = homogeneous2Pose(matrix, pose);
printf("Diana API pose2Homogeneous got: %f, %f, %f, %f, %f, %f\n",
        pose[0] , pose[1] , pose[2] , pose[3], pose[4] , pose[5]);
```

输出结果：

Diana API pose2Homogeneous got: -0.231000, 0.155000, 0.934000, 2.431323, 0.651471, -1.403724

67 pose2Homogeneous

int pose2Homogeneous (pose, matrix)
位姿转齐次变换矩阵。 **参数：** pose：位姿数组的首地址，数组长度为 6。 matrix：齐次变换矩阵数组的首地址，数组长度为 16。 **返回值：** 0：成功。 -1：失败。

调用示例：
const double PI= 3.1415926;
double pose[6]={-0.231, 0.155, 0.934, PI, PI/3, PI/6};
double matrix[16]={0};
ret = pose2Homogeneous(pose, matrix);
printf("Diana API pose2Homogeneous got:
\n%f, %f, %f, %f\n%f, %f, %f, %f\n%f, %f, %f, %f\n%f, %f, %f, %f\n",
matrix[0] , matrix[1] , matrix[2] , matrix[3],
matrix[4] , matrix[5] , matrix[6] , matrix[7],
matrix[8] , matrix[9] , matrix[10] , matrix[11],
matrix[12] , matrix[13] , matrix[14] , matrix[15]);
输出结果：
Diana API pose2Homogeneous got:
0.758804, 0.546150, 0.354875, 0.000000
0.611590, -0.784849, -0.099843, 0.000000
0.223995, 0.292800, -0.929567, 0.000000
-0.231000, 0.155000, 0.934000, 1.000000

68 enableTorqueReceiver

int enableTorqueReceiver(bEnable, strIpAddress = "")
在指定 IP 地址机械臂上，打开或关闭实时扭矩的接收。 **参数：** bEnable：输入参数，是否开启实时扭矩的接收。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值： 0：成功。 -1：失败。

调用示例：
const char* strIpAddress = "192.168.10.75";
bool bEnable = true;
int ret = enableTorqueReceiver (bEnable,strIpAddress);
if(ret < 0)
{
printf("enableTorqueReceiver failed! Return value = %d\n", ret);
}

69 sendTorque_rt

int sendTorque_rt(torque,t,strIpAddress = "")
对指定 IP 地址机械臂，用户发送实时扭矩 **参数：** torque：输入参数，用户传入的扭矩值，大小为 JOINT_NUM 的数组。 t:持续时间，单位 s strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
const char* strIpAddress = "192.168.10.75";
double dblTorque[JOINT_NUM] = {0.0};
double t = 1000;
int ret = sendTorque_rt(dblTorque,t,strIpAddress);
if(ret < 0)
{
printf("sendTorque_rt failed! Return value = %d\n", ret);
}

70 enableCollisionDetection

int enableCollisionDetection(bEnable,strIpAddress = "")
开启指定 IP 地址机械臂碰撞检测

参数：

bEnable：输入参数，是否开启碰撞检测模式。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = enableCollisionDetection(true,strIpAddress);
if(ret < 0)
{
   printf("enableCollisionDetection failed! Return value = %d\n", ret);
}
```

71 setActiveTcpPayload

```
int setActiveTcpPayload(payload,strIpAddress = "")
```

设置指定 IP 地址机械臂的负载信息

参数：

payload：负载信息，第 1 位为质量，2~4 位为质心，5~10 位为张量，大小为 10 的数组

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
double dblPayload[10] = {0.0};
int ret = setActiveTcpPayload (dblPayload,strIpAddress);
if(ret < 0)
{
   printf("setActiveTcpPayload failed! Return value = %d\n", ret);
}
```

72 servoJ

```
int servoJ (joints, time, look_ahead_time, gain, strIpAddress = "")
```

关节空间内，伺服指定 IP 地址机械臂到指定关节角位置。 servoJ 函数用于在线控制机械臂，lookahead 时间和 gain 能够调整轨迹是否平滑或尖锐。 注意：太高的 gain 或太短的 lookahead 时间可能会导致不稳定。由于该函数主要用于以较短位移为目标点的多次频繁调用，建议在实时系统环境下使用。

该函数暂不支持传送带跟踪期间使用，目前没有主动报错，使用时会卡住程序，如果使用了传送带功能请注意规避。

参数：

joints：目标关节角位置数组的首地址，数组长度为 JOINT_NUM。单位：rad。

time：运动时间。单位：s。

look_ahead_time：时间（S），范围（0.03-0.2）用这个参数使轨迹更平滑。

gain：目标位置的比例放大器，范围（100,2000）。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const double PI=3.1415926;
//以 7 轴机器人为例
double target1[7]={0, PI/6, 0, PI/2, 0, -PI/2, 0};
const char* strIpAddress = "192.168.10.75";
for(int i = 0; i < 100; ++i)
{
 target1[6] = target1[6]+PI/180;
 ret = servoJ( target1, 0.01, 0.1, 300, strIpAddress);
 if(ret < 0)
   break;
 sleep(0.001);
}
stop(strIpAddress);
```

73 servoL

int servoL (pose, time, look_ahead_time, gain, scale, active_tcp=nullptr, strIpAddress = "")

笛卡尔空间内，控制指定 IP 地址机械臂工具中心点到指定位姿。由于该函数主要用于以较短位移为目标点的多次频繁调用，建议在实时系统环境下使用。

该函数暂不支持传送带跟踪期间使用，目前没有主动报错，使用时会卡住程序，如果使用了传送带功能请注意规避。

参数：

pose：基坐标系下的目标位姿数组的首地址，数组长度为 6。前三个元素单位：m；后三个元素单位：rad，注意，后三个角度需要是轴角。

time：运动时间。单位：s。

look_ahead_time：时间（S），范围（0.03-0.2）用这个参数使轨迹更平滑。

gain：目标位置的比例放大器，范围（100,2000）。

scale：平滑比例系数。范围（0.0~1.0）。

active_tcp：需要移动的工具中心点对应的位姿向量（基于法兰坐标系），大小为 6 的数组，为空时将移动系统当前工具的中心点至 pose。（注意：此处 active_tcp 作为工具）

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const double PI=3.141592653;
const char* strIpAddress = "192.168.10.75";
double joints[7]={0, 0, 0, PI/2, 0, -PI/2, 0};//以 7 轴机器人为例
moveJ(joints,0.1,0.1,strIpAddress);
wait_move(strIpAddress);
double pose[6]={0};
getTcpPos(pose,strIpAddress);
for(int i = 0; i < 1000; ++i){
  pose[2] = pose[2] + 0.001;
  ret = servoL(pose, 0.01, 0.1, 300, 1.0, nullptr, strIpAddress);
  if(ret < 0)
    break;
  sleep(0.001);
}
stop(strIpAddress);
```

74 servoJ_ex

int servoJ_ex (joints, time, look_ahead_time, gain, realiable, strIpAddress = "")

关节空间内，伺服指定 IP 地址机械臂到指定关节角位置优化版。 servoJ_ex 函数用于在

线控制机器人，lookahead时间和 gain能够调整轨迹是否平滑或尖锐。 注意：太高的 gain或太短的 lookahead时间可能会导致不稳定。由于该函数主要用于以较短位移为目标点的多次频繁调用，建议在实时系统环境下使用。

该函数暂不支持传送带跟踪期间使用，目前没有主动报错，使用时会卡住程序，如果使用了传送带功能请注意规避。

参数：

joints：目标关节角位置数组的首地址，数组长度为 JOINT_NUM。单位：rad。

time：运动时间。单位：s。

look_ahead_time：时间（S），范围（0.03-0.2）用这个参数使轨迹更平滑。

gain：目标位置的比例放大器，范围（100,2000）。

realiable：bool 型变量，值为 true 需要 socket 反馈通信状态，行为等同 servoJ；值为 false 则无需反馈直接返回。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
timespec next_time;
//获取系统时间存入 next_time
const char* strIpAddress = "192.168.10.75";
for (int i = 0; i < 50000; i++)
{
    target_point[0] = (cos(2 * PI / 5 * i * 0.001) - 1) * PI / 3;
    servoJ_ex(target_point, 0.001, 0.1, 500, false, strIpAddress);
    next_time.tv_nsec += 1000000;//计算下次唤醒时间
    //sleep 到 next_time
}
stop(strIpAddress);
```

75 servoL_ex

int servoL_ex (pose, time, look_ahead_time, gain, scale, realiable,active_tcp=nullptr,strIpAddress = "")

笛卡尔空间内，控制指定 IP 地址机械臂工具中心点到指定位姿优化版。由于该函数主要用于以较短位移为目标点的多次频繁调用，建议在实时系统环境下使用。

该函数暂不支持传送带跟踪期间使用，目前没有主动报错，使用时会卡住程序，如果使用了传送带功能请注意规避。

参数：

pose：基坐标系下的目标位姿数组的首地址，数组长度为 6。前三个元素单位：m；后三个元素单位：rad。注意，后三个角度需要是轴角。

time：运动时间。单位：s。

look_ahead_time：时间（S），范围（0.03-0.2）用这个参数使轨迹更平滑。

gain：目标位置的比例放大器，范围（100,2000）。

scale：平滑比例系数。范围（0.0~1.0）。

realiable：bool 型变量，值为 true 需要 socket 反馈通信状态，行为等同 servoJ；值为 false 则无需反馈直接返回。

active_tcp：需要移动的工具中心点对应的位姿向量（基于法兰坐标系），大小为 6 的数组，为空时将移动系统当前工具的中心点至 pose。（注意：此处 active_tcp 作为工具）

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const double PI=3.141592653;
const char* strIpAddress = "192.168.10.75";
double joints[7]={0, 0, 0, PI/2, 0, -PI/2, 0};//以 7 轴机器人为例
moveJ(joints,0.1,0.1);
wait_move(strIpAddress);
double pose[6]={0};
getTcpPos(pose,strIpAddress);
for(int i = 0; i < 1000; ++i){
  pose[2] = pose[2] + 0.001;
  ret = servoL_ex(pose, 0.01, 0.1, 300, 1.0,true,nullptr, strIpAddress);
  if(ret < 0)
    break;
  sleep(0.001);
}
stop(strIpAddress);
```

76 speedJ_ex

int speedJ_ex (speed, a, t, realiable, strIpAddress = "")
速度模式优化版，使指定 IP 地址机械臂进行关节空间运动。时间 t 为非零时，机器人将在 t 时间后减速。如果 t 为 0，机器人将在达到目标速度时减速。该函数调用后立即返回。停止运动需要调用 stop 函数。 该函数暂不支持传送带跟踪期间使用，目前没有主动报错，使用时会卡住程序，如果使用了传送带功能请注意规避。 **参数：** speed：关节角速度数组首地址，数组长度为 JOINT_NUM。单位：rad/s。 a：加速度，单位：rad/s^2。 t：时间，单位：s。 realiable：bool 型变量，值为 true 需要 socket 反馈通信状态，行为等同 speedJ；值为 false 则无需反馈直接返回。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
double speeds[JOINT_NUM] = {0.0};
speeds[1]=0.2;
double acc = 0.40;
const char* strIpAddress = "192.168.10.75";
int ret = speedJ_ex(speeds, acc, 0, true, strIpAddress);
if(ret < 0)
{
printf("speedJ_ex failed! Return value = %d\n", ret);
}

77 speedL_ex

int speedL_ex(speed, a, t,,realiable, active_tcp=nullptr, strIpAddress = "")
速度模式优化版，使指定 IP 地址机械臂笛卡尔空间下直线运动，支持同步旋转，但笛卡尔方向必须有速度或者加速度才能旋转。时间 t 为非 0 时，机器人将在 t 时间后减速。如果 t 为 0，机器人将在达到目标速度时减速。该函数调用后立即返回。停止运动需要调用 stop 函数。 该函数暂不支持传送带跟踪期间使用，目前没有主动报错，使用时会卡住程序，如果使

用了传送带功能请注意规避。

参数：

speed：基坐标系下的工具空间速度，数组长度为 6（位置和旋转），其中前 3 个单位为 m/s，后3个单位为rad/s。位置和旋转均参考基坐标系，旋转时的旋转中心可由active_tcp 指定。

a：加速度数组，数组长度为 2，单位：m/s^2 ，rad/s^2。

t：时间，单位：s。

realiable：bool 型变量，值为 true 需要 socket 反馈通信状态，行为等同 speedL；值为 false 则无需反馈直接返回。

active_tcp：基于法兰坐标系的位姿，指定旋转时的旋转中心，大小为 6 的数组（位置+旋转矢量（轴角））；为空时旋转中心是系统当前工具中心点。（注意：机械臂做直线运动时中心点会随动，所以无旋转运动的情况下，此参数看不出影响）

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double speeds[6] = {0.1,0.0,0.0,0.0,0.0,0.0};
double acc[2] = {0.30, 0.50};
const char* strIpAddress = "192.168.10.75";
int ret = speedL_ex(speeds, acc, 0,  true, nullptr, strIpAddress);
if(ret < 0)
{
   printf("speedL_ex failed! Return value = %d\n", ret);
}
```

78 dumpToUDisk

int dumpToUDisk (time_out, strIpAddress = "")

导出指定 IP 地址机械臂的日志文件到 u 盘。控制箱中的系统日志文件（主要包含 ControllerLog.txt 和 DianaServerLog.txt）会自动复制到 u 盘。需要注意的是目前控制箱仅支持 FAT32 格式 u 盘，调用 dumpToUDisk 函数前需先插好 u 盘，如果系统日志拷贝失败将不会提示。

参数：

time_out：单位 秒，设置超时时间，一般需要大于 3 秒,-1 表示设置超时时间无穷大。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

1. 系统开机
2. 插入 u 盘到控制箱
3. 调用 Api 函数 dumpToUDisk(-1,"192.168.10.75")
4. 拔下 u 盘查看

79 inverse_ext

int inverse_ext(ref_joints, pose,joints,active_tcp=nullptr, strIpAddress = "")

针对指定 IP 地址机器人，逆解函数，给定一个参考关节角，算出欧式距离最近的逆解。现支持在不同工具坐标系下求逆解，由传入的 active_tcp 决定。

参数：

ref_joints：参考的关节角，大小为 JOINT_NUM 的数组。

pose：输入位姿数组首地址，数据为包含 active_tcp 坐标（x, y, z）和旋转矢量（轴角坐标）组合。

joints：输出关节角度数组首地址，用于传递转换的结果，大小为 JOINT_NUM 的数组。

active_tcp：工具坐标系对应的位姿向量，大小为 6 的数组，为空时，将使用默认的工具坐标系 default_tcp。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double ref_joints[JOINT_NUM] = {0.0};
double pose[6] = {0.4221, 0.0, 0.9403, 0.0, 0.0, 0.0};
double joints[JOINT_NUM] = {0.0};
const char* strIpAddress = "192.168.10.75";
int ret = inverse_ext(ref_joints,pose, joints, nullptr,strIpAddress);
if(ret < 0)
{
   printf("inverse_ext failed! Return value = %d\n", ret);
```

```
}
else
{
   printf("inverse_ext :%f, %f, %f, %f, %f, %f, %f\n", joints[0], joints[1], joints[2], joints[3],
joints[4], joints[5], joints[6]);
}
```

80 getJointLinkPos

int getJointLinkPos(joints,strIpAddress = "")

获取指定 IP 地址机械臂当前低速侧关节角

参数：

joints：输出参数。低速侧关节角,大小为 JOINT_NUM 的数组。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
double joints[JOINT_NUM] = {0.0};
int ret = getJointLinkPos(joints,strIpAddress);
if(ret < 0)
{
   printf("getJointLinkPos failed! Return value = %d\n", ret);
}
else
{
   printf("getJointLinkPos :%f, %f, %f, %f, %f, %f, %f\n", joints[0], joints[1], joints[2],
joints[3], joints[4], joints[5], joints[6]);
}
```

81 createComplexPath

int createComplexPath (complex_path_type, complex_path_id, strIpAddress = "")

在指定 IP 地址机械臂上创建一个复杂路段。

参数：

complex_path_type：输入参数。

0 :表示 NORMAL_JOINT，该模式下路径中可以添加 moveL、moveJ、moveC 对应的路径

点，路径点用 JOINT_NUM 个关节角度确定。

1：表示 MOVEP_JOINT，该模式下机械臂关节做点对点的匀速运动，路径中可以添加 moveL、moveC 对应的路径点，不可以添加 moveJ 对应的路径点，路径点用 JOINT_NUM 个关节角度确定。

2:表示 NORMAL_POSE ，该模式下路径中可以添加 moveL、moveJ、moveC 对应的路径点，路径点用末端 TCP 位姿数据确定。

3：表示 MOVEP_POSE，该模式下机械臂末端做点对点的匀速运动，路径中可以添加 moveL、moveC 对应的路径点，不可以添加 moveJ 对应的路径点，路径点用末端 TCP 位姿数据确定。

complex_path_id：输出参数。用于保存新创建 complexPath 的 ID。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

详见 addMoveJByTarget 或 addMoveLByTarget 调用示例。

82 addMoveLByTarget

int addMoveLByTarget (complex_path_id, target_joints, vel, acc, blendradius, zv_shaper_order=0, zv_shaper_frequency=0, zv_shaper_damping_ratio=0, strIpAddress = "")

在指定 IP 地址机械臂上，向已创建的路段添加 MoveL 路点。

参数：

complex_path_id：输入参数。要添加路点的路径 ID。

target_joints：输入参数。要添加的路点，即该路点的各关节角度数组的首地址，数组长度为 JOINT_NUM。单位：rad。

vel：moveL 移动到目标路点的速度。单位：m/s。

acc：moveL 移动到目标路点的加速度。单位：m/s2。

blendradius：交融半径。单位：m。

zv_shaper_order，zv_shaper_frequency，zv_shaper_damping_ratio 为整形功能相关参数，函数中默认值都为 0，此时使用机械臂默认配置参数。

zv_shaper_order：阶次，整数型参数，范围[-1,2]，其中取值-1 时，关闭整形功能， 当 zv_shaper_order， zv_shaper_frequency， zv_shaper_damping_ratio 任意一值不为 0，则使用函数传入的阶次值。

zv_shaper_frequency：频率，浮点型参数，单位：Hz。支持取值范围[0，1000]，取值为 0 时，使用机械臂默认配置参数。

zv_shaper_damping_ratio：阻尼比，浮点型参数。支持取值范围[0,1]，取值为 0 时，使用机械臂默认配置参数。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
#define PI 3.14159265358979323846
#define DEGREE_TO_RAD(x) ((x)*PI / 180.0)
#define RAD_TO_DEGREE(x) ((x)*180.0 / PI)

double dblFirstPosition [JOINT_NUM] = {DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(90.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(-90.0), DEGREE_TO_RAD(0.0)};
double dblSecondPosition [JOINT_NUM] = {DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(150.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(-30.0), DEGREE_TO_RAD(0.0) };

unsigned int uintComplexPathId = 0;
int zv_shaper_order = 0;
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
const char* strIpAddress = "192.168.10.75";
createComplexPath(NORMAL_JOINT_PATH, uintComplexPathId, strIpAddress);
addMoveLByTarget(uintComplexPathId, dblFirstPosition, 0.2, 0.4, 0,  zv_shaper_order,
zv_shaper_frequency, zv_shaper_damping_ratio, strIpAddress);
addMoveLByTarget(uintComplexPathId, dblSecondPosition, 0.2, 0.2, 0,  zv_shaper_order,
zv_shaper_frequency, zv_shaper_damping_ratio, strIpAddress);
runComplexPath(uintComplexPathId, strIpAddress);
destroyComplexPath(uintComplexPathId, strIpAddress);
wait_move(strIpAddress);
```

83 addMoveLByPose

int addMoveLByPose(complex_path_id, target_pose, vel, acc, blendradius, zv_shaper_order=0, zv_shaper_frequency=0, zv_shaper_damping_ratio=0, strIpAddress = "")
在指定 IP 地址机械臂上，向已创建的路段添加 MoveL 路点。 **参数：** complex_path_id,：输入参数。要添加路点的路径 ID。 target_pose：输入参数。路径点位姿数组首地址，数组长度为 6，保存 TCP 坐标（x, y, z）和轴角（rx, ry, rz）组合数据。 vel：moveL 移动到目标路点的速度。单位：m/s。 acc：moveL 移动到目标路点的加速度。单位：m/s2。 blendradius：交融半径。单位：m。 zv_shaper_order，zv_shaper_frequency，zv_shaper_damping_ratio 为整形功能相关参数，函数中默认值都为 0，此时使用机械臂默认配置参数。 zv_shaper_order：阶次，整数型参数，范围[-1,2]，其中取值-1 时，关闭整形功能， 当 zv_shaper_order， zv_shaper_frequency， zv_shaper_damping_ratio 任意一值不为 0，则使用函数传入的阶次值。 zv_shaper_frequency：频率，浮点型参数，单位：Hz。支持取值范围[0，1000]，取值为 0 时，使用机械臂默认配置参数。 zv_shaper_damping_ratio：阻尼比，浮点型参数。支持取值范围[0,1]，取值为 0 时，使用机械臂默认配置参数。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
double dblFirstPose [6] = { 0.4000, 0.0, 0.3000, 0.0, 0.0, 0.0 };
double dblSecondPose [6] = { 0.1000, 0.0, 0.8000, 0.0, 0.0, 0.0 };
unsigned int uintComplexPathId = 0;
int zv_shaper_order = 0;
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
const char* strIpAddress = "192.168.10.75";

```
createComplexPath(NORMAL_POSE_PATH, uintComplexPathId, strIpAddress);
addMoveLByPose(uintComplexPathId, dblFirstPose, 0.2, 0.4, 0,  zv_shaper_order,
zv_shaper_frequency, zv_shaper_damping_ratio, strIpAddress);
addMoveLByPose(uintComplexPathId, dblSecondPose, 0.2, 0.2, 0,  zv_shaper_order,
zv_shaper_frequency, zv_shaper_damping_ratio, strIpAddress);
runComplexPath(uintComplexPathId, strIpAddress);
destroyComplexPath(uintComplexPathId, strIpAddress);
wait_move(strIpAddress);
```

84 addMoveJByTarget

```
int addMoveJByTarget (complex_path_id, target_joints, vel_percent, acc_percent,
blendradius_percent, zv_shaper_order=0, zv_shaper_frequency=0,
zv_shaper_damping_ratio=0, strIpAddress = "")
```

在指定 IP 地址机械臂上，向已创建的路段添加 MoveJ 路点。

参数：

complex_path_id：输入参数。要添加路点的路径 ID。

target_joints：输入参数。要添加的路点，即该路点的各关节角度数组的首地址，数组长度为 JOINT_NUM。单位：rad。

vel_percent：moveJ 移动到目标路点的速度百分比，相对于系统安全中设定关节最大速度的百分比。

acc_percent：moveJ 移动到目标路点的加速度百分比，相对于系统安全中设定关节最大加速度的百分比。

blendradius _percent：交融半径百分比，相对于该轨迹最大交融半径的百分比。

zv_shaper_order，zv_shaper_frequency，zv_shaper_damping_ratio 为整形功能相关参数，函数中默认值都为 0，此时使用机械臂默认配置参数。

zv_shaper_order：阶次，整数型参数，范围[-1,2]，其中取值-1 时，关闭整形功能， 当 zv_shaper_order， zv_shaper_frequency， zv_shaper_damping_ratio 任意一值不为 0，则使用函数传入的阶次值。

zv_shaper_frequency：频率，浮点型参数，单位：Hz。支持取值范围[0，1000]，取值为 0 时，使用机械臂默认配置参数。

zv_shaper_damping_ratio：阻尼比，浮点型参数。支持取值范围[0,1]，取值为 0 时，使用机械臂默认配置参数。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
#define PI 3.14159265358979323846
#define DEGREE_TO_RAD(x) ((x)*PI / 180.0)
#define RAD_TO_DEGREE(x) ((x)*180.0 / PI)

double dblFirstPosition [JOINT_NUM] = {DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(90.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(-90.0), DEGREE_TO_RAD(0.0)};
double dblSecondPosition [JOINT_NUM] = {DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(150.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(-30.0), DEGREE_TO_RAD(0.0) };

unsigned int uintComplexPathId = 0;
int zv_shaper_order = 0;
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
const char* strIpAddress = "192.168.10.75";
createComplexPath(NORMAL_JOINT_PATH, uintComplexPathId, strIpAddress);
addMoveJByTarget (uintComplexPathId, dblFirstPosition, 0.2, 0.4, 0,  zv_shaper_order,
zv_shaper_frequency, zv_shaper_damping_ratio, strIpAddress);
addMoveJByTarget(uintComplexPathId, dblSecondPosition, 0.2, 0.2, 0,  zv_shaper_order,
zv_shaper_frequency, zv_shaper_damping_ratio, strIpAddress);
runComplexPath(uintComplexPathId, strIpAddress);
destroyComplexPath(uintComplexPathId, strIpAddress);
wait_move(strIpAddress);
```

85 addMoveJByPose

```
int addMoveJByPose (complex_path_id, target_pose, vel_percent, acc_percent,
blendradius_percent, zv_shaper_order=0, zv_shaper_frequency=0,
zv_shaper_damping_ratio=0, strIpAddress = "")
```

在指定 IP 地址机械臂上，向已创建的路段添加 MoveJ 路点。

参数：

complex_path_id：输入参数。要添加路点的路径 ID。

target_pose：输入参数。要添加的路点，路径点位姿数组首地址，数组长度为 6，保存

TCP 坐标（x, y, z）和轴角（rx, ry, rz）组合数据。

vel_percent：moveJ 移动到目标路点的速度百分比。相对于系统安全中设定关节最大速度的百分比。

acc_percent：moveJ 移动到目标路点的加速度百分比。相对于系统安全中设定关节最大加速度的百分比。

blendradius _percent：交融半径百分比。相对于该轨迹最大交融半径的百分比。

zv_shaper_order，zv_shaper_frequency，zv_shaper_damping_ratio 为整形功能相关参数，函数中默认值都为 0，此时使用机械臂默认配置参数。

zv_shaper_order：阶次，整数型参数，范围[-1,2]，其中取值-1 时，关闭整形功能， 当 zv_shaper_order， zv_shaper_frequency， zv_shaper_damping_ratio 任意一值不为 0，则使用函数传入的阶次值。

zv_shaper_frequency：频率，浮点型参数，单位：Hz。支持取值范围[0，1000]，取值为 0 时，使用机械臂默认配置参数。

zv_shaper_damping_ratio：阻尼比，浮点型参数。支持取值范围[0,1]，取值为 0 时，使用机械臂默认配置参数。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double dblFirstPose [6] = { 0.4000, 0.0, 0.3000, 0.0, 0.0, 0.0 };
double dblSecondPose [6] = { 0.1000, 0.0, 0.8000, 0.0, 0.0, 0.0 };
unsigned int uintComplexPathId = 0;
int zv_shaper_order = 0;
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
const char* strIpAddress = "192.168.10.75";
createComplexPath(NORMAL_POSE_PATH, uintComplexPathId, strIpAddress);
addMoveJByPose(uintComplexPathId, dblFirstPose, 0.2, 0.4, 0, zv_shaper_order, zv_shaper_frequency, zv_shaper_damping_ratio, strIpAddress);
addMoveJByPose(uintComplexPathId, dblSecondPose, 0.2, 0.2, 0, zv_shaper_order, zv_shaper_frequency, zv_shaper_damping_ratio, strIpAddress);
runComplexPath(uintComplexPathId, strIpAddress);
```

```
destroyComplexPath(uintComplexPathId, strIpAddress);
wait_move(strIpAddress);
```

86 addMoveCByTarget

int addMoveCByTarget (complex_path_id, pass_joints, target_joints, vel, acc, blendradius, ignore_rotation, zv_shaper_order=0, zv_shaper_frequency=0, zv_shaper_damping_ratio=0, strIpAddress = "")

在指定 IP 地址机械臂上，向已创建的路段添加 MoveC 路点。

参数：

complex_path_id：输入参数。要添加路点的路径 ID。

pass_joints：输入参数。要添加的 moveC 中间路点，即该路点的各关节角度数组的首地址，数组长度为 JOINT_NUM。单位：rad。

target_joints：输入参数。要添加的 moveC 目标路点，即该路点的各关节角度数组的首地址，数组长度为 JOINT_NUM。单位：rad。

vel：moveC 移动到目标路点的速度。单位：m/s。

acc：moveC 移动到目标路点的加速度。单位：m/s2。

blendradius：交融半径。单位：m。

ignore_rotation：是否忽略姿态变化

zv_shaper_order，zv_shaper_frequency，zv_shaper_damping_ratio 为整形功能相关参数，函数中默认值都为 0，此时使用机械臂默认配置参数。

zv_shaper_order：阶次，整数型参数，范围[-1,2]，其中取值-1 时，关闭整形功能， 当 zv_shaper_order， zv_shaper_frequency， zv_shaper_damping_ratio 任意一值不为 0，则使用函数传入的阶次值。

zv_shaper_frequency：频率，浮点型参数，单位：Hz。支持取值范围[0，1000]，取值为 0 时，使用机械臂默认配置参数。

zv_shaper_damping_ratio：阻尼比，浮点型参数。支持取值范围[0,1]，取值为 0 时，使用机械臂默认配置参数。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
#define PI 3.14159265358979323846
```

```
#define DEGREE_TO_RAD(x) ((x)*PI / 180.0)
#define RAD_TO_DEGREE(x) ((x)*180.0 / PI)

double dblFirstPosition [JOINT_NUM] = { DEGREE_TO_RAD(20.0), DEGREE_TO_RAD(-30.0), DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(120.0), DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(-90.0), DEGREE_TO_RAD(0.0) };
double dblSecondPosition [JOINT_NUM] = { DEGREE_TO_RAD(50.0), DEGREE_TO_RAD(-30.0), DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(120.0), DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(-90.0), DEGREE_TO_RAD(0.0) };

unsigned int uintComplexPathId = 0;
int zv_shaper_order = 0;
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
const char* strIpAddress = "192.168.10.75";
createComplexPath(NORMAL_JOINT_PATH, uintComplexPathId, strIpAddress);
addMoveCByTarget (uintComplexPathId, dblFirstPosition, dblSecondPosition, 0.2, 0.4, 0,true, zv_shaper_order, zv_shaper_frequency, zv_shaper_damping_ratio, strIpAddress);
runComplexPath(uintComplexPathId, strIpAddress);
destroyComplexPath(uintComplexPathId, strIpAddress);
wait_move(strIpAddress);
```

87 addMoveCByPose

```
int addMoveCByPose (complex_path_id, pass_pose, target_pose, vel, acc, blendradius, ignore_rotation, zv_shaper_order=0, zv_shaper_frequency=0, zv_shaper_damping_ratio=0, strIpAddress = "")
```

在指定 IP 地址机械臂上，向已创建的路段添加 MoveC 路点。

参数：

complex_path_id：输入参数。要添加路点的路径 ID。

pass _pose：输入参数。要添加的 moveC 中间路点，即路径点位姿数组首地址，数组长度为 6，保存 TCP 坐标（x, y, z）和轴角（rx, ry, rz）组合数据。

target_pose：输入参数。要添加的 moveC 目标路点，即该路径点位姿数组首地址，数组长度为 6，保存 TCP 坐标（x, y, z）和轴角（rx, ry, rz）组合数据。

vel：moveL 移动到目标路点的速度。单位：m/s。

acc：moveL 移动到目标路点的加速度。单位：m/s^2。

blendradius：交融半径。单位：m。

ignore_rotation：是否忽略姿态变化

zv_shaper_order，zv_shaper_frequency，zv_shaper_damping_ratio 为整形功能相关参数，函数中默认值都为 0，此时使用机械臂默认配置参数。

zv_shaper_order：阶次，整数型参数，范围[-1,2]，其中取值-1 时，关闭整形功能， 当 zv_shaper_order， zv_shaper_frequency， zv_shaper_damping_ratio 任意一值不为 0，则使用函数传入的阶次值。

zv_shaper_frequency：频率，浮点型参数，单位：Hz。支持取值范围[0，1000]，取值为 0 时，使用机械臂默认配置参数。

zv_shaper_damping_ratio：阻尼比，浮点型参数。支持取值范围[0,1]，取值为 0 时，使用机械臂默认配置参数。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double dblFirstPose [6] = { 0.269391, 0.1000, 0.663360, 0.0, 0.0, 0.0 };
double dblSecondPose [6] = { 0.269391, 0.1000, 0.863360, 0.0, 0.0, 0.0 };
unsigned int uintComplexPathId = 0;
int zv_shaper_order = 0;
double zv_shaper_frequency = 0;
double zv_shaper_damping_ratio = 0;
const char* strIpAddress = "192.168.10.75";
createComplexPath(NORMAL_POSE_PATH, uintComplexPathId, strIpAddress);
addMoveCByPose(uintComplexPathId, dblFirstPose, dblSecondPose, 0.2, 0.4, 0,true, zv_shaper_order, zv_shaper_frequency, zv_shaper_damping_ratio, strIpAddress);
runComplexPath(uintComplexPathId, strIpAddress);
destroyComplexPath(uintComplexPathId, strIpAddress);
wait_move(strIpAddress);
```

88 runComplexPath

int runComplexPath (complex_path_id, strIpAddress = "")

在指定 IP 地址机械臂上，启动运行设置好的路段。

参数：

complex_path_id：输入参数。要运行的路径 ID。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂

时生效。 **返回值：** 0：成功。 -1：失败。
调用示例： 详见 addMoveJByTarget 或 addMoveLByTarget 调用示例。

89 destroyComplexPath

int destroyComplexPath (complex_id_path, strIpAddress = "")
在指定 IP 地址机械臂上，销毁某个路段。 **参数：** complex_id_path：输入参数。要销毁的路径 ID。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。
调用示例： 详见 addMoveJBytarget 或 addMoveLByTarget 调用示例。

90 saveEnvironment

int saveEnvironment (strIpAddress = "")
将指定 IP 地址机械臂的控制器当前参数数据写入配置文件，用于重启机器人时初始化设置各参数，包括碰撞检测阈值、阻抗参数、DH 参数等所有可通过 API 设置的参数数据。 **参数：** strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
const char* strIpAddress = "192.168.10.75";
int ret = saveEnvironment (strIpAddress);
if(ret < 0)
{

```
    printf("Save failed!\n");
}
```

91 enterForceMode_ex

int enterForceMode_ex(dblForceDirection,dblForceValue,dblMaxApproachVelocity, dblMaxAllowTcpOffset, double *dblActiveTcp = nullptr,strIpAddress = "")

使指定 IP 地址机械臂进入力控模式，相较于 enterForceMode 增加不同坐标系的支持。

参数：

dblForceDirection：力指令的方向的数组首地址，数组长度为 3。

dblForceValue：力指令的大小。单位：N。

dblMaxApproachVelocity：最大接近速度。单位：m/s。推荐取值范围（0.01m/s~0.5m/s）。

dblMaxAllowTcpOffset：允许的最大偏移。单位：m。推荐取值范围（0.01m~1m）。

dblActiveTcp：支持的坐标系对应的矩阵，大小为 16 的数组

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double dblForceDir[3]={0,0,-1};
double dblForceValue = 2.0;
double dblMaxVel = 0.1;
double dblMaxOffset = 0.2;
double dblActiveTcp[16] = {1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1};
const char* strIpAddress = "192.168.10.75";
if (enterForceMode_ex(dblForceDir, dblForceValue, dblMaxVel, dblMaxOffset,
dblActiveTcp,strIpAddress) < 0)
{
   printf("Diana API enterForceMode_ex failed!\n");
}
```

92 readDI

int readDI(group_name, name, value, strIpAddress = "")

读取指定 IP 地址机械臂一个数字输入的值。

参数：

group_name：数字输入的分组，例如，’board’,’plc’；

name：数字输入的信号名，例如，’di0’；

value：读取返回的值。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
int value;
const char* strIpAddress = "192.168.10.75";
int ret = readDI("board", "di0", value, strIpAddress);
if(ret < 0)
{
    printf("readDI failed!\n");
}
else
{
   printf("di0:%d\n", value);
}
```

93 readDO

int readDO(group_name, name, value, strIpAddress = "")

读取指定 IP 地址机械臂一个数字输出的值。

参数：

group_name：数字输出的分组，例如，’board’,’plc’；

name：数字输出的信号名，例如，’do0’；

value：读取返回的值。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
int value;
const char* strIpAddress = "192.168.10.75";
```

```
int ret = readDO("board", "do0", value, strIpAddress);
if(ret < 0)
{
    printf("readDO failed!\n");
}
else
{
   printf("do0: %d\n", value);
}
```

94 readAI

int readAI (group_name, name, mode, value, strIpAddress = "")

读取指定 IP 地址机械臂一个模拟输入的值和模式。

参数：

group_name：模拟输入的分组，例如，’board’,’plc’；

name：模拟输入的信号名，例如，’ai0’；

mode：当前模拟输入模式；

value：读取返回的值。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
int mode;
double value;
const char* strIpAddress = "192.168.10.75";
int ret = readAI("board", "ai0",mode,value, strIpAddress);
if(ret < 0)
{
   printf("ReadAI failed!\n");
}
else
{
   printf("ai0 mode= %d, value=%f\n", mode, value);
}
```

95 readAO

int readAO (group_name, name, mode, value, strIpAddress = "")
读取指定 IP 地址机械臂一个模拟输出的值和模式。 **参数：** group_name：模拟输出的分组，例如，’board’,’plc’； name：模拟输出的信号名，例如，’ao0’； mode：当前模拟输出模式； value：读取返回的值。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
int mode;
double value;
const char* strIpAddress = "192.168.10.75";
int ret = readAO("board", "ao0",mode,value, strIpAddress);
if(ret < 0)
{
printf("ReadAO failed!\n");
}
else
{
printf("ao0 mode= %d, value=%f\n", mode, value);
}

96 setAIMode

int setAIMode (group_name, name, mode, strIpAddress = "")
设置指定 IP 地址机械臂模拟输入的模式。 **参数：** group_name：模拟输入的分组，例如，’board’,’plc’； name：模拟输入的信号名，例如，’ai0’； mode: 模拟输入模式，1 代表电流，2 代表电压。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂

时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
int mode = 1;
const char* strIpAddress = "192.168.10.75";
int ret = setAIMode("board", "ai0",mode, strIpAddress);
if(ret < 0)
{
printf("SetAIMode failed!\n");
}

97 writeDO

int writeDO (group_name, name,value, strIpAddress = "")
设置指定 IP 地址机械臂一个数字输出的值。 可用于改变模拟信号的模式，将信号名替换为"aimode"或"aomode"。 **参数：** group_name：数字输出的分组，例如，’board’,’plc’ name：数字输出的信号名，例如，’do0’ value：设置的值 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效 **返回值：** 0：成功。 -1：失败。

调用示例：
int value = 0;
const char* strIpAddress = "192.168.10.75";
int ret = writeDO ("board", "do0",value, strIpAddress);
if(ret < 0)
{
printf("WriteDO failed!\n");
}

98 writeAO

int writeAO (group_name, name, mode, value, strIpAddress = "")
设置指定 IP 地址机械臂一个模拟输出的值和模式。 **参数：** group_name：模拟输出的分组，例如，’board’,’plc’； name：模拟输出的信号名，例如，’ao0’； mode：当前模拟输出模式； value：设置输出的值。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
int mode = 1;
double value = 8.8;
const char* strIpAddress = "192.168.10.75";
int ret = writeAO ("board", "ao0", mode, value, strIpAddress);
if(ret < 0)
{
printf("WriteAO failed!\n");
}

99 readBusCurrent

int readBusCurrent(current, strIpAddress = "")
读取指定 IP 地址机械臂总线电流。 **参数：** current：总线电流。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。
调用示例： double current; const char* strIpAddress = "192.168.10.75";

```
int ret = readBusCurrent(current, strIpAddress);
if(ret < 0)
{
   printf("ReadBusCurrent failed!\n");
}
else
{
   printf("ReadBusCurrent :%f\n", current);
}
```

100 readBusVoltage

int readBusVoltage (voltage, strIpAddress = "")

读取指定 IP 地址机械臂总线电压。

参数：

voltage：总线电压。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double voltage;
const char* strIpAddress = "192.168.10.75";
int ret = readBusVoltage(voltage, strIpAddress);
if(ret < 0)
{
   printf("ReadBusVoltage failed!\n");
}
else
{
   printf("readBusVoltage :%f\n", voltage);
}
```

101 getDH

int getDH(aDH, alphaDH,dDH,thetaDH, strIpAddress = "")

获取指定 IP 地址机械臂的 DH 参数。

参数：

aDH：连杆长度的数组，长度为 JOINT_NUM

alphaDH:连杆转角的数组，长度为 JOINT_NUM

dDH:连杆偏距的数组，长度为 JOINT_NUM

thetaDH:连杆的关节角的数组，长度为 JOINT_NUM

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double a[JOINT_NUM],alpha[JOINT_NUM],d[JOINT_NUM],theta[JOINT_NUM];
const char* strIpAddress = "192.168.10.75";
int ret = getDH(a,alpha,d,theta, strIpAddress);
if(ret < 0)
{
   printf("getDH failed!\n");
}
```

102 getOriginalJointTorque

```
int getOriginalJointTorque(torques, strIpAddress = "")
```

获取指定 IP 地址机械臂传感器反馈的扭矩值，未减去零偏。

参数：

torques：反馈扭矩的数组，长度为 JOINT_NUM

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double torques[JOINT_NUM];
const char* strIpAddress = "192.168.10.75";
int ret = getOriginalJointTorque(torques, strIpAddress);
if(ret < 0)
{
   printf("getOriginalJointTorque failed!\n");
}
```

103 getJacobiMatrix

int getJacobiMatrix(matrix_jacobi, strIpAddress = "")
获取指定 IP 地址机械臂的雅各比矩阵。 **参数：** matrix_jacobi：雅各比矩阵的数组，长度为 6*JOINT_NUM strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
double matrix_jacobi [6*JOINT_NUM]={0.0};
const char* strIpAddress = "192.168.10.75";
int ret = getJacobiMatrix (matrix_jacobi, strIpAddress);
if(ret < 0)
{
printf("getJacobiMatrix failed!\n");
}

104 resetDH

int resetDH(strIpAddress = "")
重置指定 IP 地址机械臂用户自定义 DH 参数。 **参数：** strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
const char* strIpAddress = "192.168.10.75";
int ret = resetDH(strIpAddress);
if(ret < 0)
{
printf("resetDH failed!\n");
}

105 runProgram

int runProgram(programName, strIpAddress = "")
运行指定 IP 地址机械臂某个程序。 **参数：** programName：程序的名字，最长 127 个英文字符。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
const char* strIpAddress = "192.168.10.75";
int ret=runProgram("AgileRobots", strIpAddress);
if(ret < 0)
{
printf(" runProgram failed!\n");
}

106 stopProgram

int stopProhram(programName, strIpAddress = "")
停止指定 IP 地址机械臂某个程序。 **参数：** programName：程序的名字，最长 127 个英文字符。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
const char* strIpAddress = "192.168.10.75";
int ret=stopProgram("AgileRobots", strIpAddress);
if(ret < 0)
{
printf(" stopProgram failed!\n");
}

107 getVariableValue

int getVariableValue(variableName,value, strIpAddress = "")
获取指定 IP 地址机械臂某个全局变量的值。 **参数：** variableName：全局变量的名字 value: 获取的全局变量的值 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
double value;
const char* strIpAddress = "192.168.10.75";
int ret = getVariableValue ("GLOBAL", value, strIpAddress);
if(ret < 0)
{
printf("getVariableValue failed!\n");
}

108 setVariableValue

int setVariableValue(variableName,value, strIpAddress = "")
设置指定 IP 地址机械臂某个全局变量的值。 **参数：** variableName：全局变量的名字 value: 设置的全局变量的值 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
double value=2.0;
const char* strIpAddress = "192.168.10.75";
int ret = setVariableValue ("GLOBAL",value, strIpAddress);

```
if(ret < 0)
{
   printf("setVariableValue failed!\n");
}
```

109 isTaskRunning

int isTaskRunning(programName, strIpAddress = "")

判断指定 IP 地址机械臂某个程序是否在运行。

参数：

programName：程序名称，最长 127 个英文字符。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：运行中。

-1：没有运行中。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = isTaskRunning ("AgileRobots", strIpAddress);
if(ret < 0)
{
   printf("AgileRobots is not running!\n");
}
```

110 pauseProgram

int pauseProgram(strIpAddress = "")

暂停指定 IP 地址机械臂所有程序。

注意：该指令会暂停所有程序，且内部会保留暂停标记，调用过此指令后必须调用 resumeProgram 或者 stopAllProgram 清除暂停标记，否则下次运行程序会直接进入暂停态。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = pauseProgram(strIpAddress);
if(ret  == 0)
{
   printf("All program is paused!\n");
}
```

111 resumeProgram

int resumeProgram(strIpAddress = "")

恢复运行指定 IP 地址机械臂已经暂停的程序。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = resumeProgram(strIpAddress);
if(ret == 0)
{
   printf("All program is resume!\n");
}
```

112 stopAllProgram

int stopAllProgram(strIpAddress = "")

停止指定 IP 地址机械臂所有程序。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = stopAllProgram(strIpAddress);
```

```
if(ret == 0)
{
   printf("All program is stop!\n");
}
```

113 isAnyTaskRunning

int isAnyTaskRunning(strIpAddress = "")

判断指定 IP 地址机械臂是否有程序在运行。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：有任务运行。

-1：无任务运行。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = isAnyTaskRunning(strIpAddress);
if(ret < 0)
{
   printf("Any program is not running!\n");
}
```

114 cleanErrorInfo

int cleanErrorInfo(strIpAddress = "")

清除指定 IP 地址机械臂的错误信息。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = cleanErrorInfo(strIpAddress);
if(ret < 0)
{
```

```
    printf("cleanErrorInfo failed!\n");
}
```

115 setCollisionLevel

int setCollisionLevel(level, strIpAddress = "")

设置指定 IP 地址机械臂的碰撞检测类型

参数：

level：碰撞等级，int 类型，值为 0 或者下面几种值的组合（0 表示只考虑后台最大保护阈值）：

0x01：关节空间检测

0x02：笛卡尔空间检测

0x04： TCP 合力检测

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = setCollisionLevel(0x01|0x02, strIpAddress);
if(ret < 0)
{
    printf("setCollision failed!\n");
}
```

116 mappingInt8Variant

int mappingInt8Variant(variableName,index, strIpAddress = "")

在指定 IP 地址机械臂上映射 int8 型变量。

参数：

variableName：变量名称

index: 映射变量序号

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。
调用示例： 参考 setMappingAddress 示例。

117 mappingDoubleVariant

int mappingDoubleVariant(variableName,index, strIpAddress = "")
在指定 IP 地址机械臂上映射 double 型变量。 **参数：** variableName：变量名称 index: 映射变量序号 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。
调用示例： 参考 setMappingAddress 示例。

118 mappingInt8IO

int mappingInt8IO(groupName,name,index, strIpAddress = "")
在指定 IP 地址机械臂上映射数字信号 IO。 **参数：** groupName：IO 组名 name：IO 名字 index: 映射序号 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。
调用示例： 参考 setMappingAddress 示例。

119 mappingDoubleIO

int mappingDoubleIO(groupName,name,index, strIpAddress = "")

在指定 IP 地址机械臂上映射模拟信号 IO。 **参数：** groupName：IO 组名 name: IO 名称 index: 映射序号 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。
调用示例： 参考 setMappingAddress 示例。

120 setMappingAddress

int setMappingAddress(dblAddress,doubleItemCount,int8Address,int8ItemCount, strIpAddress = "")
在指定 IP 地址机械臂上设置地址映射。 **参数：** dblAddress：double 型数据的映射地址，double 数组 doubleItemCount：double 型数据的映射数量，int 型，最大支持 40 个 int8Address：int8 型数据的映射地址，int8 数组 int8ItemCount: int8 型数据的映射数量，int 型，最大支持 160 个 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。
调用示例： mapping 的使用说明： mapping 的作用是将机械臂上 double 型变量，int8 型变量（程序变量表中），数字信号 IO，模拟信号 IO 通过索引与自定义的内存地址（数组）建立映射连接，通过自定义的内存地址访问上述变量 1、步骤 1：首先定义一些 INT8 型和 DOUBLE 型的索引值，通过枚举方式：

```
enum INT8_VALUE_INDEX
{
   DI0 = 0,
   DI1,
   DI2,
   DI3,
   DI4,
   DI5,
};

enum DOUBLE_VALUE_INDEX
{
   AI0 = 0,
   AI1,
   TEST_VALUE_0,
   TEST_VALUE_1,
};
```

2、步骤 2：将机械臂上 double 型变量，int8 型变量（程序变量表中），数字信号 IO，模拟信号 IO 与索引值建立映射：

```
//mapping double value
   mappingDoubleIO("board", "ai0", AI0);
   mappingDoubleIO("board", "ai1", AI1);
   mappingDoubleVariant("test_value_0", TEST_VALUE_0);
   mappingDoubleVariant("test_value_1", TEST_VALUE_1);

//mapping int8 value
   mappingInt8IO("board", "di0", DI0);
   mappingInt8IO("board", "di1", DI1);
   mappingInt8IO("board", "di2", DI2);
   mappingInt8IO("board", "di3", DI3);
   mappingInt8IO("board", "di4", DI4);
   mappingInt8IO("board", "di5", DI5);
```

3、步骤 3：设置映射地址（自定义的内存地址（数组））

```
   int8_t int8_array[6] = {0};
```

```
double double_array[4] = {0.0};
setMappingAddress(double_array, 4, int8_array, 6);
```

4、步骤 4：使用自定义的 double_array 和 int8_array 地址名通过索引值访问机械臂上 double 型变量，int8 型变量（程序变量表中），数字信号 IO，模拟信号 IO 的值

```
lockMappingAddress();//给数据加锁
printf("数字信号 DI[0-7] = {%d, %d, %d, %d, %d, %d, %d, %d}\n", int8_array[DI0], int8_array[DI1], int8_array[DI2], int8_array[DI3], int8_array[DI4], int8_array[DI5]);
printf("模拟信号 AI[2] = {%f, %f}\n", double_array[AI0], double_array[AI1]);
printf("double 型变量 test_value_0=%f, test_value_1=%f\n", double_array[TEST_VALUE_0], double_array[TEST_VALUE_1]);
unlockMappingAddress();//给数据解锁
```

121 lockMappingAddress

int lockMappingAddress(strIpAddress = "")

在指定 IP 地址机械臂上访问数据时加锁。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

参考 setMappingAddress 示例。

122 unlockMappingAddress

int unlockMappingAddress(strIpAddress = "")

在指定 IP 地址机械臂上解锁数据锁。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例： 参考 setMappingAddress 示例。

123 getJointCount

int getJointCount(strIpAddress = "")
在指定 IP 地址机械臂上，获取其机械臂的关节数目 **参数：** strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 关节数量。

调用示例：
const char* strIpAddress = "192.168.10.75";
int count = getJointCount(strIpAddress);
printf("Joint Count = %d\n",count);

124 getWayPoint

int getWayPoint(strWayPointName, dblTcppos, dblJoints, strToolName, strWorkpieceName, strIpAddress = "")
获取指定 IP 地址机械臂上路点变量信息。 **参数：** strWayPointName：路点变量名称 。 dblTcppos：位姿信息，大小为 6 的数组，其中，后三个角度为轴角 dblJoints：关节角信息。 strToolName: 路点关联工具坐标系名称； strWorkpieceName：路点关联工件坐标系名称； strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。
调用示例： 参考 setWayPoint 示例。

125 setWayPoint

int setWayPoint(strWayPointName ,dblTcppos, dblJoints, strToolName, strWorkpieceName,

strIpAddress = "")

修改指定 IP 地址机械臂上路点变量的值

参数：

strWayPointName：路点变量名称 。

dblTcppos：位姿信息，大小为 6 的数组，其中，后三个角度为轴角

dblJoints：关节角信息。

strToolName: 路点关联工具坐标系名称；

strWorkpieceName：路点关联工件坐标系名称；

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

<0：失败。失败原因详见错误码。

调用示例：

```
    double Target[7] = { DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(90.0), DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(-90.0), DEGREE_TO_RAD(0.0) };
    moveJ(Target, 0.8, 0.4);
    wait_move();

    const char *strWayPointName = "api_01";
    const char *strToolName = "tool1";
    const char *strWorkpieceName = "work1";
    char strToolTmp[64];
    char strWorkpieceTmp[64];
    double add_dblTcppos[6] = { 0.0 };
    getTcpPos(add_dblTcppos);
    double add_dblJoints[7] = { 0.0 };
    getJointPos(add_dblJoints);

    int ret = addWayPoint(strWayPointName, add_dblTcppos, add_dblJoints, strToolName, strWorkpieceName);
    if (ret < 0)
    {
        printf( "addWayPoint failed!\n");
```

```
    }
    else
    {
        printf( "addWayPoint succeed!\n");
    }

    double get_dblTcppos[6] = { 0.0 };
    double get_dblJoints[7] = { 0.0 };
    ret = getWayPoint(strWayPointName, get_dblTcppos, get_dblJoints, strToolTmp,
strWorkpieceTmp);
    if (ret < 0)
    {
        printf( "getWayPoint failed!\n");
    }
    else
    {
        printf( "getWayPoint succeed!\n");
        printf( "tool name : %s\n", strToolTmp);
        printf( "workpiece name : %s\n", strWorkpieceTmp);
        printf( "addWayPoint-Tcppos: %f, %f, %f, %f, %f, %f\n",
            add_dblTcppos[0], add_dblTcppos[1], add_dblTcppos[2], add_dblTcppos[3],
add_dblTcppos[4], add_dblTcppos[5]);
        printf( "getWayPoint-Tcppos: %f, %f, %f, %f, %f, %f\n\n\n",
            get_dblTcppos[0], get_dblTcppos[1], get_dblTcppos[2], get_dblTcppos[3],
get_dblTcppos[4], get_dblTcppos[5]);

        printf( "addWayPoint-Joints: %f, %f, %f, %f, %f, %f, %f\n",
            add_dblJoints[0], add_dblJoints[1], add_dblJoints[2], add_dblJoints[3],
add_dblJoints[4], add_dblJoints[5], add_dblJoints[6]);
        printf( "getWayPoint-Joints: %f, %f, %f, %f, %f, %f, %f\n\n\n",
            get_dblJoints[0], get_dblJoints[1], get_dblJoints[2], get_dblJoints[3],
get_dblJoints[4], get_dblJoints[5], get_dblJoints[6]);
    }

    double Target1[7] = { DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(-60.0),
DEGREE_TO_RAD(0.0), DEGREE_TO_RAD(150.0), DEGREE_TO_RAD(0.0),
DEGREE_TO_RAD(-90.0), DEGREE_TO_RAD(0.0) };
    moveJ(Target1, 0.8, 0.4);
```

```
    wait_move();

    double set_dblTcppos[6] = { 0.0 };
    getTcpPos(set_dblTcppos);
    double set_dblJoints[7] = { 0.0 };
    getJointPos(set_dblJoints);
    ret = setWayPoint(strWayPointName, set_dblTcppos, set_dblJoints, strToolName,
strWorkpieceName);
    if (ret < 0)
    {
        printf( "setWayPoint failed!\n");
    }
    else
    {
        printf( "setWayPoint succeed!\n");
    }

    ret = getWayPoint(strWayPointName, get_dblTcppos, get_dblJoints, strToolName,
strWorkpieceName);
    if (ret < 0)
    {
        printf( "getWayPoint failed!\n");
    }
    else
    {
        printf( "getWayPoint succeed!\n");
        printf( "setWayPoint-Tcppos: %f, %f, %f, %f, %f, %f\n",
            set_dblTcppos[0], set_dblTcppos[1], set_dblTcppos[2], set_dblTcppos[3],
set_dblTcppos[4], set_dblTcppos[5]);
        printf( "getWayPoint-Tcppos: %f, %f, %f, %f, %f, %f\n\n\n",
            get_dblTcppos[0], get_dblTcppos[1], get_dblTcppos[2], get_dblTcppos[3],
get_dblTcppos[4], get_dblTcppos[5]);

        printf( "setWayPoint-Joints: %f, %f, %f, %f, %f, %f, %f\n",
            set_dblJoints[0], set_dblJoints[1], set_dblJoints[2], set_dblJoints[3],
set_dblJoints[4], set_dblJoints[5], set_dblJoints[6]);
        printf( "getWayPoint-Joints: %f, %f, %f, %f, %f, %f, %f\n\n\n",
            get_dblJoints[0], get_dblJoints[1], get_dblJoints[2], get_dblJoints[3],
```

```
get_dblJoints[4], get_dblJoints[5], get_dblJoints[6]);
    }

    ret = deleteWayPoint(strWayPointName);
    if (ret < 0)
    {
        printf( "deleteWayPoint failed!\n");
    }
    else
    {
        ret = getWayPoint(strWayPointName, get_dblTcppos, get_dblJoints ,strToolTmp,
strWorkpieceTmp);
        if (ret < 0)
        {
            printf( "deleteWayPoint succeed!\n");
        }
        else
        {
            printf( "deleteWayPoint failed!\n");
        }
    }
```

126 addWayPoint

```
int addWayPoint(strWayPointName, dblTcppos, dblJoints, strToolName, strWorkpieceName,
strIpAddress = "")
```

在指定 IP 地址机械臂上新增路点变量。

参数：

strWayPointName：路点变量名称 。

dblTcppos：位姿信息，大小为 6 的数组，其中，后三个角度为轴角

dblJoints：关节角信息。

strToolName: 路点关联工具坐标系名称；

strWorkpieceName：路点关联工件坐标系名称；

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

<0：失败。失败原因详见错误码。
调用示例： 参考 setWayPoint 示例。

127 deleteWayPoint

int deleteWayPoint(strWayPointName, strIpAddress = "")
在指定 IP 地址机械臂上删除路点变量。 **参数：** strWayPointName：路点变量名称。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。
调用示例： 参考 setWayPoint 示例。

128 getDefaultActiveTcp

int getDefaultActiveTcp(default_tcp , strIpAddress = "")
获取指定 IP 地址机械臂的默认工具坐标系。 **参数：** default_tcp：输出参数，Tcp 相对于末端法兰盘的 4*4 齐次变换矩阵的首地址，数组长度为 16。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
const char* strIpAddress = "192.168.10.75";
double default_tcp[16] = {0.0};
int ret = getDefaultActiveTcp(default_tcp, strIpAddress);
if(ret < 0)
{
  printf("getDefaultActiveTcp failed! Return value = %d\n", ret);

```
}
```

129 getDefaultActiveTcpPose

int getDefaultActiveTcpPose (arrPose,strIpAddress = "")

获取指定 IP 地址机械臂默认的工具坐标系的位姿。

参数：

arrPose：输出参数。TCP 相对于末端法兰盘的位姿向量的首地址，数组长度为 6

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double pose[6] = {0.0};
const char* strIpAddress = "192.168.10.75";
if (getDefaultActiveTcpPose (pose,strIpAddress) < 0)
{
   printf("Diana API getDefaultActiveTcpPose failed!\n");
}
else
{
   printf("getDefaultActiveTcpPose: %f, %f, %f, %f, %f, %f\n", pose[0], pose[1], pose[2], pose[3], pose[4], pose[5]);
}
```

130 getActiveTcpPayload

int getActiveTcpPayload(payload,strIpAddress = "")

获取指定 IP 地址机械臂的负载信息

参数：

payload：负载信息，第 1 位为质量，2~4 位为质心，5~10 位为张量，大小为 10 的数组

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
double dblPayload[10] = {0.0};
int ret = getActiveTcpPayload (dblPayload,strIpAddress);
if(ret < 0)
{
   printf("getActiveTcpPayload failed! Return value = %d\n", ret);
}
else
{
   printf("getActiveTcpPayload: %f, %f, %f, %f, %f, %f, %f, %f, %f, %f\n", dblPayload[0],
dblPayload[1], dblPayload[2], dblPayload[3], dblPayload[4], dblPayload[5],
   dblPayload[6], dblPayload[7], dblPayload[8], dblPayload[9]);
}
```

131 zeroSpaceManualMove

int zeroSpaceManualMove (direction,dblJointVel,dblJointAcc,strIpAddress = "")

控制指定 IP 地址机械臂进入或退出零空间手动移动模式。

参数：

direction：输入参数，控制零空间手动运动的方向，1 表示向前，-1 表示向后运动

dblJointVel：输入参数，各关节运动的速度，单位 rad/s

dblJointAcc：输入参数，各关节运动的加速度，单位 rad/s^2

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
//以 7 轴机器人为例
double dblJointVel[7]={0.1,0.1,0.1,0.1,0.1,0.1,0.1};
double dblJointAcc[7]={0.1,0.1,0.1,0.1,0.1,0.1,0.1};
int ret = zeroSpaceManualMove(1,dblJointVel,dblJointAcc,strIpAddress);
if (ret < 0){
   printf("Diana API zeroSpaceManualMove failed!\n");
}
else{
   M_SLEEP(5000);
```

```
}
```

132 moveTcp_ex

```
int moveTcp_ex(c,d, v, a, strIpAddress = "")
```

手动移动指定 IP 地址的机械臂 TCP。该函数会立即返回，停止运动需要调用 stop 函数。

参数：

c：坐标系类型，枚举及其含义如下：

- E_BASE_COORDINATE：基坐标系
- E_TOOL_COORDINATE：工具坐标系
- E_WORK_PIECE_COORDINATE：工件坐标系
- E_VIEW_COORDINATE：视角坐标系

d：表示移动方向的枚举类型，枚举及其含义如下

- T_MOVE_X_UP 表示沿 x 轴正向
- T_MOVE_X_DOWN 表示沿 x 轴负向
- T_MOVE_Y_UP 表示沿 y 轴正向
- T_MOVE_Y_DOWN 表示沿 y 轴负向
- T_MOVE_Z_UP 表示沿 z 轴正向
- T_MOVE_Z_DOWN 表示沿 z 轴负向

v：速度，单位：m/s。

a：加速度，单位：m/s^2。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
coordinate_e e = E_BASE_COORDINATE;
tcp_direction_e dtype = T_MOVE_X_UP;
double vel = 0.1;
double acc = 0.2;
const char* strIpAddress = "192.168.10.75";
int ret = moveTcp_ex(e, dtype, vel, acc, strIpAddress);
if(ret < 0)
{
   printf("moveTcp_ex failed! Return value = %d\n", ret);
}
```

133 rotationTCP_ex

```
int rotationTCP_ex(c, d, v, a, strIpAddress = "")
```

使指定的 IP 地址的机械臂绕 TCP 变换位姿。该函数会立即返回，停止运动需要调用 stop 函数。

参数：

c：坐标系类型，枚举及其含义如下：

- E_BASE_COORDINATE：基坐标系
- E_TOOL_COORDINATE：工具坐标系
- E_WORK_PIECE_COORDINATE：工件坐标系
- E_VIEW_COORDINATE：视角坐标系

d：表示移动方向的枚举类型，枚举及其含义如下：

- T_MOVE_X_UP 表示绕 x 轴正向旋转
- T_MOVE_X_DOWN 表示绕 x 轴负向旋转
- T_MOVE_Y_UP 表示绕 y 轴正向旋转
- T_MOVE_Y_DOWN 表示绕 y 轴负向旋转
- T_MOVE_Z_UP 表示绕 z 轴正向旋转
- T_MOVE_Z_DOWN 表示绕 z 轴负向旋转

v：速度，单位：rad/s。

a：加速度，单位：rad/s^2。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效

返回值：

0：成功。

-1：失败。

调用示例：

```
coordinate_e e = E_BASE_COORDINATE;
tcp_direction_e dtype = T_MOVE_X_UP;
double vel = 0.1;
double acc = 0.2;
const char* strIpAddress = "192.168.10.75";
int ret = rotationTCP_ex(e,dtype, vel, acc, strIpAddress);
if(ret < 0)
{
   printf("rotationTCP failed! Return value = %d\n", ret);
}
else
{
   M_SLEEP(3000);
}
```

134 setExternalAppendTorCutoffFreq

int setExternalAppendTorCutoffFreq(dblFreq, strIpAddress = "")
针对指定 IP 地址机械臂，设置各关节附加力矩的滤波截止频率 **参数：** dblFreq：输入参数，各关节附加力矩的滤波截止频率，需要提供一个正值。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
int ret = setExternalAppendTorCutoffFreq(1.0,"192.168.10.75");
if(ret < 0){
printf("setExternalAppendTorCutoffFreq failed! Return value = %d\n", ret);
}

135 poseTransform

int poseTransform(srcPose, srcMatrixPose,dstMatrixPose,dstPose)
把指定坐标系下的位姿转换到其他坐标系下 **参数：** srcPose：输入参数，源坐标系下的位姿，大小为 6 的数组 srcMatrixPose：输入参数，源坐标系对应的位姿向量，大小为 6 的数组 dstMatrixPose：输入参数，目标坐标系对应的位姿向量，大小为 6 的数组 dstPose：输出参数，转换到目标坐标系下的位姿，大小为 6 的数组 **返回值：** 0：成功。 -1：失败。

调用示例：
double srcPose[6] = {0.087,0.0,1.0827,0.0,0.0,0.0};
double srcMatrixPose[6] = {0};
double dstMatrixPose[6] = {0.1, 0.2, 0.3, 0.0, 0.0, 0.0};
double dstPose[6] = {0};
poseTransform(srcPose,srcMatrixPose,dstMatrixPose,dstPose);
for(int i = 0;i<6;++i)
{
printf("%lf ",dstPose[i]);
}

136 setEndKeyEnableState

int setEndKeyEnableState(bEnable, strIpAddress = "")
针对指定 IP 地址机械臂，使能其末端零力驱动按钮 **参数：** bEnable：输入参数，使能末端零力驱动按钮 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
int ret = setEndKeyEnableState(true,"192.168.10.75");
if(ret < 0){
printf("setEndKeyEnableState failed! Return value = %d\n", ret);
}

137 updateForce

int updateForce(dblForceValue,strIpAddress = "")
针对指定 IP 地址机械臂上，在力控模式下，实时改变力指令的大小 **参数：** dblForceValue：输入参数，力的大小，需要是一个大于 0 的数 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
int iFrameType = 1;
double dblFrameMatrix[16] = {1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1};
double dblForceDirection[3]={0,0,-1};
double dblForceValue = 1.0;
double dblMaxVel = 0.1;
double dblMaxOffset = 0.2;
const char* strIpAddress = "192.168.10.75";
double joints[7] = {0,0,0,3.141592653/2,0,0,0};//以 7 轴机器人为例

```
moveJ(joints,0.2,0.2,strIpAddress);
wait_move(strIpAddress);
if (enterForceMode(iFrameType, dblFrameMatrix, dblForceDirection, dblForceValue, dblMaxVel, dblMaxOffset,strIpAddress) < 0)
{
   printf("Diana API enterForceMode failed!\n");
}else{
   int count = 0;
   while(count++ < 2000){
      dblForceValue -=0.001;
      updateForce(dblForceValue,strIpAddress);
      M_SLEEP(1);
   }
   int intExitMode = 0;
   if (leaveForceMode(intExitMode, strIpAddress) < 0)
   {
      printf("Diana API leaveForceMode failed!\n");
   }
}
```

138 inverseClosedFull

int inverseClosedFull(pose,lock_joint_index, lock_joint_position, ref_joints, active_tcp=nullptr,strIpAddress = "")
在指定 IP 地址机械臂上，基于工具坐标系，给定一个参考关节角，约束单轴求逆解，注意，工业版 Diana 只能锁定第七轴。现支持在不同工具坐标系下求逆解，由传入的 active_tcp 决定。 **参数：** pose：输入位姿数组首地址，数据为包含 active_tcp 坐标（x, y, z）和旋转矢量（轴角坐标）组合。 lock_joint_index：输入参数，被约束的关节号 lock_joint_position：输入参数，被约束关节的角度，单位为弧度 ref_joints：参考的关节角，大小为 JOINT_NUM 的数组。 active_tcp：工具坐标系对应的位姿向量，大小为 6 的数组，为空时，将使用默认的工具坐标系 default_tcp。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

非负数：生成逆解对应的 ID

-1：失败。

调用示例：

```
const char *strIpAddress = "192.168.10.75";
double pose[6] = {0.087, 0.0, 1.0827, 0.0, 0.0, 0.0};
int lock_joint_index = JOINT_NUM;
double lock_joint_position = 0;
double ref_joints[ JOINT_NUM] = {0.0};
double joints[JOINT_NUM]={0.0};
int id = inverseClosedFull(pose, lock_joint_index, lock_joint_position, ref_joints, nullptr, strIpAddress);
if (id == -1){
   printf("inverseClosedFull failed! Return value = %d\n", id);
}
else{
   int size = getInverseClosedResultSize(id);
   int ret = -1;
   if (size > 0){
      for (int i = 0; i < size; ++i){
         if (getInverseClosedJoints(id, i, joints, strIpAddress) == 0){
            printf("(%lf,%lf,%lf,%lf,%lf,%lf,%lf)", joints[0], joints[1], joints[2], joints[3], joints[4], joints[5], joints[6]);
         }
      }
      destoryInverseClosedItems(id);
   }
}
```

139 getInverseClosedResultSize

int getInverseClosedResultSize(id,strIpAddress = "")

在指定 IP 地址的机械臂上，根据 ID 获取约束单轴求逆解结果的组数

参数：

id：输入参数，所求逆解对应的 ID

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

非负数：ID 对应逆解的组数 -1：失败。
调用示例： 见 inverseClosedFull 示例

140 getInverseClosedJoints

int getInverseClosedJoints(id,index,joints,strIpAddress = "")
在指定 IP 地址机械臂上，根据 ID 按索引获取对应关节角 **参数：** id：输入参数，所求逆解对应的 ID index：输入参数，对应的多组逆解中的编号 joints：输出参数，需要求的多组逆解中编号对应的逆解值 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功； -1：失败。
调用示例： 见 inverseClosedFull 示例

141 destoryInverseClosedItems

int destoryInverseClosedItems(id,strIpAddress = "")
在指定 IP 地址机械臂上，根据 ID 删除约束单轴求逆解的结果数据集 **参数：** id：输入参数，逆解对应的 ID strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。
调用示例： 见 inverseClosedFull 示例

142 nullSpaceFreeDriving

int nullSpaceFreeDriving(enable,strIpAddress = "")

zeroSpaceFreeDriving 的宏，用法参见 zeroSpaceFreeDriving
调用示例： 见 zeroSpaceFreeDriving 示例

143 nullSpaceManualMove

int nullSpaceFreeDriving(enable,strIpAddress = "")
zeroSpaceManualMove 的宏，用法参见 zeroSpaceManualMove
调用示例： 见 zeroSpaceManualMove 示例

144 getGravInfo

int getGravInfo(grav,strIpAddress = "")
针对指定 IP 地址的机械臂，获取其安装信息的重力矢量 参数： grav：重力矢量，大小为 3 的数组 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
const char* ipAddress = "192.168.10.75";
double grav[3] = {0.0};
int ret = getGravInfo(grav,ipAddress);
if(ret < 0){
printf("getGravInfo failed! Return value = %d\n", ret);
}
else{
printf("getGravInfo :%f, %f, %f\n", grav[0], grav[1], grav[2]);
}

145 setGravInfo

int setGravInfo(grav,strIpAddress = "")
针对指定 IP 地址的机械臂，设置其重力矢量 参数： grav：重力矢量，大小为 3 的数组

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* ipAddress = "192.168.10.75";
double grav[3] = {0.0};
int ret = setGravInfo(grav,ipAddress);
if(ret < 0){
   printf("setGravInfo failed! Return value = %d\n", ret);
}
```

146 getGravAxis

```
int getGravAxis(grav_axis,strIpAddress = "")
```

针对指定 IP 地址的机械臂，获取其安装信息的轴角

参数：

grav_axis：安装时的轴角，单位为 rad

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* ipAddress = "192.168.10.75";
double grav_axis[3] = {0.0};
int ret = getGravAxis (grav_axis,ipAddress);
if(ret < 0){
   printf("getGravAxis failed! Return value = %d\n", ret);
}
else{
   printf("getGravAxis :%f, %f, %f\n", grav_axis[0], grav_axis[1], grav_axis[2]);
}
```

147 setGravAxis

```
int setGravAxis(grav_axis,strIpAddress = "")
```

针对指定 IP 地址的机械臂，设置其安装信息的轴角 参数： grav_axis：安装轴角，单位 rad strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
const char* ipAddress = "192.168.10.75";
double grav_axis [3] = {0.0};
int ret = setGravAxis(grav_axis,ipAddress);
if(ret < 0){
printf("setGravAxis failed! Return value = %d\n", ret);
}

148 speedLOnTcp

int speedLOnTcp(speed, a, t, active_tcp=nullptr, strIpAddress = "")
速度模式优化版，使指定 IP 地址机械臂笛卡尔空间下直线运动，支持同步旋转，但笛卡尔方向必须有速度或加速度才能旋转。时间 t 为非零时，机器人将在 t 时间后减速。如果 t 为 0，机器人将在达到目标速度时减速。该函数调用后立即返回。停止运动需要调用 stop 函数。 **参数：** speed：指定系统当前工具中心点的空间速度，数组长度为 6（位置和旋转），其中前 3 个单位为 m/s，后 3 个单位为 rad/s。平移速度和旋转速度的参考坐标系由 active_tcp 指定。 a：加速度数组，数组长度为 2，前一个代表平移加速度，单位：m/s^2；后一个代表旋转加速度，单位：rad/s^2 t：时间，单位：s。 active_tcp：基于法兰中心的位姿，大小为 6 的数组（位置和旋转矢量（轴角）），平移和旋转方向参考此位姿，旋转中心是系统当前工具中心点；为空时平移和旋转方向参考系统当前工具坐标系，旋转中心是系统当前工具中心点。strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。

-1：失败。

调用示例：

```
double speeds[6] = {0.1,0.0,0.0,0.0,0.0,0.0};
double acc[2] = {0.30, 0.50};
const char* strIpAddress = "192.168.10.75";
int ret = speedLOnTcp(speeds, acc, 0,  nullptr, strIpAddress);
if(ret < 0)
{
   printf("speedLOnTcp failed! Return value = %d\n", ret);
}
M_SLEEP(2000);
```

149 getTcpForceInToolCoordinate

int getTcpForceInToolCoordinate(forces,strIpAddress = "")

在指定 IP 地址机械臂上，获取工具坐标系的 Tcp 外力值

参数：

forces：输出参数，Tcp 外力，大小为 6

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char *strIpAddress = "192.168.10.75";
double forces[6] = {0.0};
int ret = getTcpForceInToolCoordinate(forces,strIpAddress);
for(int j=0;j<6;++j){
   printf("%lf,",forces[j]);
}
```

150 calculateJacobi

int calculateJacobi(dblJacobiMatrix,dblJointPosition,intJointCount,strIpAddress = "")

在指定 IP 地址机械臂上，求解末端法兰中心点坐标系相对于基坐标系的雅各比矩阵

参数：

dblJacobiMatrix：输出参数，雅各比矩阵，大小为 6* JOINT_NUM 的矩阵

dblJointPosition：输入参数，用于计算雅各比矩阵的关节角（与当前机械臂反馈关节角

无关），大小为 JOINT_NUM 的数组

intJointCount：输入参数，关节数量

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char *strIpAddress = "192.168.10.75";
const int intJointCount =  JOINT_NUM;
const int intTcpCount = 6;
double dblJacobiMatrix[intTcpCount*intJointCount] = {0};
double dblJointPosition[intJointCount] = {0};
int ret = calculateJacobi(dblJacobiMatrix,dblJointPosition,intJointCount,strIpAddress);
if(ret == - 1){
   printf("cannot get jacobi matrix");
}else{
   for(int i = 0;i< intJointCount;++i){
      for(int j=0;j< intTcpCount;++j){
         printf("%lf,",dblJacobiMatrix[i* intTcpCount + j]);
      }
      printf("\n");
   }
}
```

151 calculateJacobiTF

```
int
calculateJacobiTF(dblJacobiMatrix,dblJointPosition,intJointCount,active_tcp=nullptr,strIpAddr
ess = "")
```

在指定 IP 地址机械臂上，求解工具中心点坐标系相对于基坐标系的雅各比矩阵，现在支持求解不同的工具，由传入的 active_tcp 决定

参数：

dblJacobiMatrix：输出参数，雅各比矩阵，大小为 6* JOINT_NUM 的矩阵

dblJointPosition：输入参数，用于计算雅各比矩阵的关节角（与当前机械臂反馈关节角无关），大小为 JOINT_NUM 的数组

intJointCount：输入参数，关节数量

active_tcp：基于法兰中心的工具位姿，大小为 6 的数组，根据关节角 dblJointPosition 求解得到此工具中心点相对于基坐标系的雅各比矩阵；为空时，视为无工具，求解得到法兰中心相对基坐标系的雅各比矩阵。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char *strIpAddress = "192.168.10.75";
const int intJointCount =  JOINT_NUM;
const int intTcpCount = 6;
double dblJacobiMatrix[intTcpCount*intJointCount] = {0};
double dblJointPosition[intJointCount] = {0};
int ret = calculateJacobiTF(dblJacobiMatrix,dblJointPosition,intJointCount, nullptr ,strIpAddress);
if(ret == - 1){
   printf("cannot get jacobi matrix");
}else{
   for(int i = 0;i<intTcpCount;++i){
      for(int j=0;j<intJointCount;++j){
         printf("%lf,",dblJacobiMatrix[i* intTcpCount + j]);
      }
      printf("\n");
   }
}
```

152 getMechanicalJointsPositionRange

int getMechanicalJointsPositionRange (dblMinPos, dblMaxPos, strIpAddress = " ")

获取指定 IP 地址机械臂各关节角的机械限位。

参数：

dblMinPos：输出参数，用于存放各关节角的最小机械限位，大小为 JOINT_NUM 的数组，单位：rad。

dblMaxPos：输出参数，用于存放各关节角的最大机械限位，大小为 JOINT_NUM 的数组，单位：rad。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂

时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblMinPos[7] = {0}, dblMaxPos[7]={0};
int ret = getMechanicalJointsPositionRange(dblMinPos, dblMaxPos, strIpAddress);
printf("getMechanicalJointsPositionRange ret = %d dblMinPos = {%f,%f,%f,%f,%f,%f,%f}
dblMaxPos = {%f,%f,%f,%f,%f,%f,%f}\n"
        , ret
        , dblMinPos[0]
        , dblMinPos[1]
        , dblMinPos[2]
        , dblMinPos[3]
        , dblMinPos[4]
        , dblMinPos[5]
        , dblMinPos[6]
        , dblMaxPos[0]
        , dblMaxPos[1]
        , dblMaxPos[2]
        , dblMaxPos[3]
        , dblMaxPos[4]
        , dblMaxPos[5]
        , dblMaxPos[6]);
```

153 getMechanicalMaxJointsVel

int getMechanicalMaxJointsVel (dblVel, strIpAddress = " ")

获取指定 IP 地址机械臂各关节最大机械速度。

参数：

dblVel：输出参数，用于存放各关节的最大机械速度，大小为 JOINT_NUM 的数组，单位：rad/s。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblMaxVel[7]={0};
int ret = getMechanicalMaxJointsVel(dblMaxVel, strIpAddress);
printf("getMechanicalJointsMaxVel ret = %d dblMaxVel = {%f,%f,%f,%f,%f,%f,%f}\n"
, ret
, dblMaxVel[0]
, dblMaxVel[1]
, dblMaxVel[2]
```

```
, dblMaxVel[3]
, dblMaxVel[4]
, dblMaxVel[5]
, dblMaxVel[6]);
```

154 getMechanicalMaxJointsAcc

int getMechanicalMaxJointsAcc (dblAcc, strIpAddress = " ")

获取指定 IP 地址机械臂各关节最大机械加速度。

参数：

dblAcc：输出参数，用于存放各关节的最大机械加速度，大小为 JOINT_NUM 的数组，单位：rad/s^2。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblMaxAcc[7]={0};
int ret = getMechanicalMaxJointsAcc(dblMaxAcc, strIpAddress);
printf("getMechanicalJointsMaxAcc ret = %d dblMaxAcc = {%f,%f,%f,%f,%f,%f,%f}\n"
, ret
, dblMaxAcc[0]
, dblMaxAcc[1]
, dblMaxAcc[2]
, dblMaxAcc[3]
, dblMaxAcc[4]
, dblMaxAcc[5]
, dblMaxAcc[6]);
```

155 getMechanicalMaxCartVelAcc

int getMechanicalMaxCartVelAcc (dblMaxCartTranslationVel, dblMaxCartRotationVel, dblMaxCartTranslationAcc, dblMaxCartRotationAcc, strIpAddress = " ")

获取指定 IP 地址笛卡尔空间最大机械平移速度、最大机械旋转速度、最大机械平移加速度和最大机械旋转加速度。

参数：

dblMaxCartTranslationVel：输出参数，用于存放笛卡尔空间最大机械平移速度，double 型变量，单位：m/s。

dblMaxCartRotationVel：输出参数，用于存放笛卡尔空间最大机械旋转速度，double 型变量，单位：rad/s。

dblMaxCartTranslationAcc：输出参数，用于存放笛卡尔空间最大机械平移加速度，

double 型变量，单位：m/s^2。

dblMaxCartRotationAcc：输出参数，用于存放笛卡尔空间最大机械旋转加速度，double 型变量，单位：rad/s^2。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblLinearVel = 0, dblRotationVel = 0, dblLinearAcc = 0, dblRotationAcc = 0;
int ret = getMechanicalMaxCartVelAcc(dblLinearVel, dblRotationVel, dblLinearAcc,
dblRotationAcc, strIpAddress);
printf("getMechanicalCartMaxVelAcc ret = %d
{dblLinearVel(%f),dblRotationVel(%f),dblLinearAcc(%f),dblRotationAcc(%f)}\n"
, ret
, dblLinearVel
, dblRotationVel
, dblLinearAcc
, dblRotationAcc);
```

156 getJointsPositionRange

int getJointsPositionRange (dblMinPos, dblMaxPos, strIpAddress = " ")

获取指定 IP 地址机械臂各关节的极限位。

参数：

dblMinPos：输出参数，用于存放关节角的最小极限位，大小为 JOINT_NUM 的数组，单位：rad。

dblMaxPos：输出参数，用于存放关节角的最大极限位，大小为 JOINT_NUM 的数组，单位：rad。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblMinPos[7] = {0}, dblMaxPos[7]={0};
int ret = getJointsPositionRange(dblMinPos, dblMaxPos, strIpAddress);
printf("getJointsPositionRange ret = %d dblMinPos = {%f,%f,%f,%f,%f,%f,%f} dblMaxPos =
{%f,%f,%f,%f,%f,%f,%f}\n"
, ret
```

```
, dblMinPos[0]
, dblMinPos[1]
, dblMinPos[2]
, dblMinPos[3]
, dblMinPos[4]
, dblMinPos[5]
, dblMinPos[6]
, dblMaxPos[0]
, dblMaxPos[1]
, dblMaxPos[2]
, dblMaxPos[3]
, dblMaxPos[4]
, dblMaxPos[5]
, dblMaxPos[6]);
```

157 getMaxJointsVel

int getMaxJointsVel (dblVel, strIpAddress = " ")

获取指定 IP 地址机械臂各关节最大软速度。

参数：

dblVel：输出参数，用于存放各关节的最大软速度，大小为JOINT_NUM的数组，单位：rad/s。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblMaxVel[7]={0};
int ret = getMaxJointsVel(dblMaxVel, strIpAddress);
printf("getMaxJointsVel ret = %d dblMaxVel = {%f,%f,%f,%f,%f,%f,%f}\n"
, ret
, dblMaxVel[0]
, dblMaxVel[1]
, dblMaxVel[2]
, dblMaxVel[3]
, dblMaxVel[4]
, dblMaxVel[5]
, dblMaxVel[6]);
```

158 getMaxJointsAcc

int getMaxJointsAcc (dblAcc, strIpAddress = " ")

获取指定 IP 地址机械臂各关节最大软加速度。

参数：

dblAcc：输出参数，用于存放各关节的最大软加速度，大小为 JOINT_NUM 的数组，单

位：rad/s^2。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblMaxAcc[7]={0};
int ret = getMaxJointsAcc(dblMaxAcc, strIpAddress);
printf("getMaxJointsAcc ret = %d dblMaxAcc = {%f,%f,%f,%f,%f,%f,%f}\n"
, ret
, dblMaxAcc[0]
, dblMaxAcc[1]
, dblMaxAcc[2]
, dblMaxAcc[3]
, dblMaxAcc[4]
, dblMaxAcc[5]
, dblMaxAcc[6]);
```

159 getMaxCartTranslationVel

int getMaxCartTranslationVel (dblMaxCartTranslationVel, strIpAddress = " ")

获取指定 IP 地址机械臂笛卡尔空间最大软平移速度。

参数：

dblMaxCartTranslationVel：输出参数，用于存放笛卡尔空间最大软平移速度，double 型变量，单位：m/s。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblLinearVel = 0;
int ret = getMaxCartTranslationVel(dblLinearVel, strIpAddress);
printf("getMaxCartTranslationVel ret = %d dblLinearVel=%f\n"
, ret
, dblLinearVel
);
```

160 getMaxCartRotationVel

int getMaxCartRotationVel (dblMaxCartRotationVel, strIpAddress = " ")

获取指定 IP 地址机械臂笛卡尔空间最大软旋转速度。

参数：

dblMaxCartRotationVel：输出参数，用于存放笛卡尔空间最大软旋转速度，double 型变量，单位：rad/s。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblRotationVel = 0;
int ret = getMaxCartRotationVel(dblRotationVel, strIpAddress);
printf("getMaxCartRotationVel ret = %d dblRotationVel=%f\n"
, ret
, dblRotationVel);
```

161 getMaxCartTranslationAcc

int getMaxCartTranslationAcc (dblMaxCartTranslationAcc, strIpAddress = " ")

获取指定 IP 地址机械臂笛卡尔空间最大软平移加速度。

参数：

dblMaxCartTranslationAcc：输出参数，用于存放笛卡尔空间最大软平移加速度，double 型变量，单位：m/s^2。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblLinearAcc = 0;
int ret = getMaxCartTranslationAcc(dblLinearAcc, strIpAddress);
printf("getMaxCartTranslationAcc ret = %d dblLinearAcc=%f\n"
, ret
, dblLinearAcc);
```

162 getMaxCartRotationAcc

int getMaxCartRotationAcc (dblMaxCartRotationAcc, strIpAddress = " ")

在指定 IP 地址机械臂笛卡尔空间最大软旋转加速度。

参数：

dblMaxCartRotationAcc：输出参数，用于存放笛卡尔空间最大软旋转加速度，double 型变量，单位：rad/s^2。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblRotationAcc = 0;
int ret = getMaxCartRotationAcc(dblRotationAcc, strIpAddress);
printf("getMaxCartRotationAcc ret = %d dblRotationAcc=%f\n"
, ret
, dblRotationAcc);
```

163 setJointsPositionRange

int setJointsPositionRange (dblMinPos, dblMaxPos, strIpAddress = " ")

设置指定 IP 地址机械臂各关节极限位。

参数：

dblMinPos：输入参数，关节角的最小极限限位，大小为 JOINT_NUM 的数组，单位：rad。

dblMaxPos：输入参数，关节角的最大极限限位，大小为 JOINT_NUM 的数组，单位：rad。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

<0：失败。失败原因详见错误码。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblMinPos[7] = {0}, dblMaxPos[7]={0};
int ret = getJointsPositionRange(dblMinPos, dblMaxPos, strIpAddress);
for (int i = 0; i < 7; ++i) {
	dblMinPos[i]++;
	dblMaxPos[i]--;
}
ret = setJointsPositionRange(dblMinPos,dblMaxPos,strIpAddress);
printf("setJointsPositionRange return %d\n",ret);
ret = saveEnvironment(strIpAddress);
printf("saveEnvironment return %d\n",ret);
```

164 setMaxJointsVel

int setMaxJointsVel (dblVel, strIpAddress = " ")
设置指定 IP 地址机械臂各关节的最大软速度。 **参数：** dblVel：输入参数，各关节的最大软速度，大小为 JOINT_NUM 的数组，单位：rad/s。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** <0：失败。失败原因详见错误码。 0：成功。

调用示例：
const char* strIpAddress= "192.168.10.75";
double dblMaxVel[7]={0};
int ret = getMaxJointsVel(dblMaxVel, strIpAddress);
for (int i = 0; i < 7; ++i) {
dblMaxVel[i]--;
}
ret = setMaxJointsVel(dblMaxVel, strIpAddress);
printf("setMaxJointsVel return %d\n",ret);
ret = saveEnvironment(strIpAddress);
printf("saveEnvironment return %d\n",ret);

165 setMaxJointsAcc

int setMaxJointsAcc (dblAcc, strIpAddress = " ")
设置指定 IP 地址机械臂各关节的最大软加速度。 **参数：** dblAcc：输入参数，用于存放各关节的最大软加速度，大小为 JOINT_NUM 的数组，单位：rad/s^2。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** -1：失败。 0：成功。

调用示例：
const char* strIpAddress= "192.168.10.75";
double dblMaxAcc[7]={0};
int ret = getMaxJointsAcc(dblMaxAcc, strIpAddress);
for (int i = 0; i < 7; ++i) {
dblMaxAcc[i]--;
}
ret = setMaxJointsAcc(dblMaxAcc, strIpAddress);
ret = saveEnvironment(strIpAddress);
printf("saveEnvironment return %d\n",ret);

166 setMaxCartTranslationVel

int setMaxCartTranslationVel (dblMaxCartTranslationVel, strIpAddress = " ")
设置指定 IP 地址机械臂笛卡尔空间最大软平移速度。 **参数：** dblMaxCartTranslationVel：输入参数，用于存放笛卡尔空间最大软平移速度，double 型变量，单位：m/s。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** -1：失败。 0：成功。

调用示例：
const char* strIpAddress= "192.168.10.75";
double dblLinearVel = 0.5;
int ret = setMaxCartTranslationVel(dblLinearVel, strIpAddress);
printf("setMaxCartTranslationVel return = %d\n",ret);
ret = saveEnvironment(strIpAddress);
printf("saveEnvironment return %d\n",ret);

167 setMaxCartRotationVel

int setMaxCartRotationVel (dblMaxCartRotationVel, strIpAddress = " ")
设置指定 IP 地址机械臂笛卡尔空间最大软旋转速度。 **参数：** dblMaxCartRotationVel：输入参数，迪卡尔空间最大软旋转速度，double 型变量，单位：rad/s。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** -1：失败。 0：成功。

调用示例：
const char* strIpAddress= "192.168.10.75";
double dblRotationVel = 1;
int ret = setMaxCartRotationVel(dblRotationVel, strIpAddress);
printf("setMaxCartRotationVel return = %d\n",ret);
ret = saveEnvironment(strIpAddress);
printf("saveEnvironment return %d\n",ret);

168 setMaxCartTranslationAcc

int setMaxCartTranslationAcc (dblMaxCartTranslationAcc, strIpAddress = " ")
设置指定 IP 地址机械臂笛卡尔空间最大软平移加速度。 **参数：** dblMaxCartTranslationAcc：输入参数，笛卡尔空间最大软平移加速度，double 型变量，单位：m/s^2。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** -1：失败。 0：成功。

调用示例：
const char* strIpAddress= "192.168.10.75";
double dblLinearAcc = 1;
int ret = setMaxCartTranslationAcc(dblLinearAcc, strIpAddress);
printf("setMaxCartTranslationAcc return = %d\n",ret);
ret = saveEnvironment(strIpAddress);
printf("saveEnvironment return %d\n",ret);

169 setMaxCartRotationAcc

int setMaxCartRotationAcc (dblMaxCartRotationAcc, strIpAddress = " ")
设置指定 IP 地址机械臂笛卡尔空间最大软旋转加速度。 **参数：** dblMaxCartRotationAcc：输入参数，笛卡尔空间最大软旋转加速度，double 型变量，单位：rad/s^2。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** -1：失败。 0：成功。

调用示例：
const char* strIpAddress= "192.168.10.75";
double dblRotationAcc = 1;
int ret = setMaxCartRotationAcc(dblRotationAcc, strIpAddress);
printf("setMaxCartRotationAcc return = %d\n",ret);
ret = saveEnvironment(strIpAddress);
printf("saveEnvironment return %d\n",ret);

170 requireHandlingError

int requireHandlingError (int *error, strIpAddress = " ")
获取已确认且待处理的错误码。 **参数：** error：用于存放错误码的值 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** -1：失败。 0：成功。
调用示例： **int errorCode = 0, ret = requireHandlingError(&errorCode);**

171 getJointsSoftLimitRange

int getJointsSoftLimitRange (double* dblMinPos, double* dblMaxPos, strIpAddress = " ")
获取指定 IP 地址机械臂各关节软限位。 **参数：** dblMinPos：输出参数，关节角的最小软限位，大小为 JOINT_NUM 的数组，单位：rad。 dblMaxPos：输出参数，关节角的最大软限位，大小为 JOINT_NUM 的数组，单位：rad。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** <0：失败。失败原因详见错误码。 0：成功。

调用示例：
const char* strIpAddress= "192.168.10.75";
double dblMinPos[7] = {0}, dblMaxPos[7]={0};
int ret = getJointsSoftLimitRange(dblMinPos, dblMaxPos, strIpAddress);
printf("getJointsSoftLimitRange return %d\n",ret);
for (int i = 0; i < 7; ++i) {
printf("Joint %d min:%f\n", i + 1, dblMinPos[i]);
printf("Joint %d max:%f\n", i + 1, dblMaxPos[i]);
 	
}

172 setJointsSoftLimitRange

int setJointsSoftLimitRange (dblMinPos, dblMaxPos, strIpAddress = " ")

设置指定 IP 地址机械臂各关节软限位。

参数：

dblMinPos：输入参数，关节角的最小软限位，大小为 JOINT_NUM 的数组，单位：rad。

dblMaxPos：输入参数，关节角的最大软限位，大小为 JOINT_NUM 的数组，单位：rad。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
const char* strIpAddress=  "192.168.10.75";
double dblMinPos[7] = {0}, dblMaxPos[7]={0};
int ret = getJointsSoftLimitRange(dblMinPos, dblMaxPos, strIpAddress);
for (int i = 0; i < 7; ++i) {
      dblMinPos[i]++;
      dblMaxPos[i]--;
}
setJointsSoftLimitRange(dblMinPos, dblMaxPos, strIpAddress);
```

173 getFunctionOptI4

int getFunctionOptI4 (function_id, opt_name, int *opt_value, strIpAddress = " ")

获取指定功能的可选参数。

参数：

function_id：指定功能 Id, 0 为自由驱动（暂不支持），1 为笛卡尔阻抗。

opt_name：功能的可选参数，function_id 为 1 时，参数如下：

a) 0x10100：获取笛卡尔阻抗的参考坐标系；
b) 0x10101：获取笛卡尔阻抗是否锁轴。

opt_value：用于存放参数的值，随 opt_name 的值变化：

a) 0x10100：0 代表基坐标系，1 代表工具坐标系；
b) 0x10101：1 代表锁轴。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂

时生效。 **返回值：** -1：失败。 0：成功。

调用示例：
int opt_value = 0;
int ret = getFunctionOptI4 (1, 0x10100, &opt_value);

174 setFunctionOptI4

int setFunctionOptI4 (function_id, opt_name, int opt_value, strIpAddress = " ")
设置指定功能的可选参数。 **参数：** function_id：指定功能 Id, 0 为自由驱动（暂不支持），1 为笛卡尔阻抗。 opt_name：功能的可选参数，function_id 为 1 时，参数如下： a) 0x10100：表示笛卡尔阻抗的参考坐标系； b) 0x10102：笛卡尔阻抗锁轴。 opt_value：用于设置参数的值，随 opt_name 的值变化： a) 0x10100：0 代表基坐标系，1 代表工具坐标系； b) 0x10102：当前版本仅支持锁第 3 轴（即取值 2），-1 代表解锁。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** -1：失败。 0：成功。
调用示例： int ret = setFunctionOptI4 (1, 0x10100, 1);

175 enterRescueMode

int enterRescueMode(strIpAddress = "")
进入安全处理状态，安全处理状态有三种模式，安全零力模式，关节驱动模式和笛卡尔驱动模式，默认进入关节驱动模式，进入安全处理状态后可通过 switchRescueMode 来切换安全处理模式，只有控制器处于 Idle 状态时可以进行该操作。原函数名为 enterSafetyIdle。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = enterRescueMode(strIpAddress);
```

176 leaveRescueMode

int leaveRescueMode (strIpAddress = "")

退出安全处理模式。原函数名为 leaveSafetyIdle。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = leaveRescueMode(strIpAddress);
```

177 getCartImpedanceCoordinateType

int getCartImpedanceCoordinateType(strIpAddress = " ")

在指定 IP 地址机械臂上，获取设置笛卡尔阻抗模式时坐标系种类。

参数：

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

坐标系的类型，0：基坐标系， 1：工具坐标系。

调用示例：
const char* strIpAddress = "192.168.10.75 ";
int ret = getCartImpedanceCoordinateType(strIpAddress);
if(ret == 0){
printf("it's base coordinate\n ");
}
else{
printf("it's tool coordinate\n ");
}

178 setCartImpedanceCoordinateType

int setCartImpedanceCoordinateType(intCoordinateType, strIpAddress = " ")
在指定 IP 地址机械臂上，设置笛卡尔阻抗模式时坐标系种类。 **参数：** intCoordinateType：坐标系的类型，0：基坐标系， 1：工具坐标系。 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
const char* strIpAddress = "192.168.10.75 ";
int ret = setCartImpedanceCoordinateType(0,strIpAddress);
if(ret < 0){
printf("setCartImpedanceCoordinateType failed! Return value = %d\n ", ret);
}

179 setJointLockedInCartImpedanceMode

int setJointLockedInCartImpedanceMode(const bool bLock, const int intLockedJointIndex, strIpAddress = "")
设置指定 IP 地址机械臂在笛卡尔阻抗和/或力控模式下锁定/解锁某轴（当前版本仅支持第 3 轴）。该设置将在机械臂进入笛卡尔阻抗或者力控模式时正式生效。（注：该 API 是一个设置选项，默认情况下不锁轴，如果需要在笛卡尔阻抗模式或力控模式下锁定某

轴，需要先设置该选项再进入笛卡尔阻抗模式或力控模式）

参数：

bLock：输入参数，如果该值为 true，表示机械臂在笛卡尔阻抗和/或力控模式下将锁定某轴（当前版本仅支持锁定第 3 轴，即 intLockedJointIdex 必须为 2，否则锁定无效）；如果该值为 false，则不论 intLockedJointIdex 取值多少，均表示机械臂在笛卡尔阻抗和/或力控模式下解锁某轴（当前版本仅支持解锁第 3 轴）。

intLockedJointIdex：输入可选参数，表示轴的索引值（索引从 0 开始），缺省值为 2（即第 3 轴）。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。

0：成功。

调用示例：

```
// 设置锁定 3 轴
const char* robot0 =  "192.168.10.75";
const bool bLock = true;
int intLockedJointIndex = 2;
int ret = setJointLockedInCartImpedanceMode(bLock, intLockedJointIndex, robot0);
printf("setJointLockedInCartImpedanceMode return = %d\n",ret);

// 设置解锁 3 轴
const char* robot0 =  "192.168.10.75";
const bool bLock = false;
int intLockedJointIndex = -1;
int ret = setJointLockedInCartImpedanceMode(bLock, intLockedJointIndex, robot0);
printf("setJointLockedInCartImpedanceMode return = %d\n",ret);
```

180 getJointLockedInCartImpedanceMode

```
int getJointLockedInCartImpedanceMode(bool &isLocked, strIpAddress =  "")
```

查询指定 IP 地址机械臂在笛卡尔阻抗和/或力控模式下某轴（当前版本仅支持第 3 轴）是否属于锁定/解锁状态。

参数：

isLocked：输出参数，如果该值为 true，表示机械臂在笛卡尔阻抗和/或力控模式下某轴（当前版本仅支持第 3 轴）处于锁定状态；如果该值为 false，则表示机械臂在笛卡尔阻抗和/或力控模式下某轴（当前版本仅支持第 3 轴）处于解锁状态。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

-1：失败。 0：成功。

调用示例：
const char* robot0 = "192.168.10.75";
bool bIsLocked = false;
int ret = getJointLockedInCartImpedanceMode(bIsLocked, robot0);
printf("getJointLockedInCartImpedanceMode return = %d\n",ret);

181 setThresholdTorque

int setThresholdTorque (arrThreshold,strIpAddress = "")
设置指定 IP 地址机械臂各关节传感器检测阈值，此阈值在切换工作模式时用于检测是否允许切换工作模式。 **参数：** arrThreshold：表示各关节阈值 arrThreshold 的数组的首地址，数组长度为 JOINT_NUM, 单位（N.m) strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
double arrThreshold[7] = { 6, 6, 5, 5, 2, 2, 2};
const char* strIpAddress = "192.168.10.75";
if (setThresholdTorque (arrThreshold, strIpAddress) < 0)
{
  printf("Diana API setThresholdTorque failed!\n");
}

182 getThresholdTorque

int getThresholdTorque(arrThreshold,strIpAddress = "")
获取指定 IP 地址机械臂各关节传感器检测阈值，此阈值在切换工作模式时用于检测是否允许切换工作模式。 **参数：** arrThreshold：表示各关节阈值 arrThreshold 的数组的首地址，数组长度为 JOINT_NUM, 单位（N.m) strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
double arrThreshold [JOINT_NUM] = { 0};
const char* strIpAddress = "192.168.10.75";
if ( getThresholdTorque (arrThreshold, strIpAddress) < 0)
{
   printf("Diana API getThresholdTorque failed!\n");
}
else
{
   printf(" getThresholdTorque : arrThreshold=%f, %f, %f, %f, %f, %f, %f\n",
   arrThreshold[0], arrThreshold[1], arrThreshold[2], arrThreshold[3], arrThreshold[4],
arrThreshold[5], arrThreshold[6]);
}
```

183 setHeartbeatParam

int setHeartbeatParam(int disconnectTimeout, int stopRobotTimeout, strIpAddress = "")

设置 API 心跳相关的超时时间。

参数：

disconnectTimeout：API 断连超时时间,单位(毫秒 ms)(-1 时相当于永不断连)

stopRobotTimeout：停止机器人超时时间,单位(毫秒 ms)(-1 时相当于永不停止)

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = setHeartbeatParam(-1, -1, strIpAddress);
if (ret == -1) {
   printf("setHeartbeatParam fail\n");
}
```

184 getHeartbeatParam

int getHeartbeatParam(int *disconnectTimeout, int *stopRobotTimeout, strIpAddress = "")

获取 API 心跳相关的超时时间。

参数：

disconnectTimeout：API 断连超时时间,单位(毫秒 ms)(-1 时相当于永不断连)

stopRobotTimeout：停止机器人超时时间,单位(毫秒 ms)(-1 时相当于永不停止)

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int disconnectTimeout;
int stopRobotTimeout;
int ret = getHeartbeatParam(&disconnectTimeout, &stopRobotTimeout, strIpAddress);
if (ret == 0) {
   printf("timeout:%d:%d\n", disconnectTimeout, stopRobotTimeout);
}
```

185 customRobotState

int customRobotState(int action, unsigned long long customBits, strIpAddress = "")

定制机器人状态推送信息。

参数：

action：_CustmRoobotStateAction 枚举类型，分别定义为：

API_CUSTOM_ADD：增加 customBits 指示的推送信息。

API_CUSTOM_DEL：减少 customBits 指示的推送信息。

API_CUSTOM_RSEET：重置为 customBits 指示的推送信息。

CustomBits：定制推送信息比特位。可定制以下信息：

```
#define  ROBOTSTATE_CUSTOM_BASIC          (0x00000001)  //基本状态信息，必须推送，不可定制，包括 BasicRobotState_1 中 stateBits, trajectoryState 及 BasicRobotState2 中所有数据
#define  ROBOTSTATE_CUSTOM_JOINTPOS       (0x00000002)  //关节反馈位置
#define  ROBOTSTATE_CUSTOM_LINKJOINTPOS   (0x00000004)  //低速侧关节反馈位置
#define  ROBOTSTATE_CUSTOM_JOINTANGULARVEL  (0x00000008)  // 关节反馈
```

速度

#define ROBOTSTATE_CUSTOM_JOINTCURRENT (0x00000010) // 关节反馈电流

#define ROBOTSTATE_CUSTOM_ORIGINJOINTTORQUE (0x00000020) // 关节反馈力矩(含零偏)

#define ROBOTSTATE_CUSTOM_JOINTTORQUEOFFSET (0x00000040) //关节扭矩传感器零偏

#define ROBOTSTATE_CUSTOM_JOINTFORCE (0x00000080) // 关节外力

#define ROBOTSTATE_CUSTOM_TCPFORCE (0x00000100) // 末端受到的外力

#define ROBOTSTATE_CUSTOM_ALL (0xffffffff) //定制所有

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = customRobotState(API_CUSTOM_RESET, ROBOTSTATE_CUSTOM_TCPFORCE | ROBOTSTATE_CUSTOM_JOINTFORCE | ROBOTSTATE_CUSTOM_JOINTANGULARVEL, strIpAddress);
if (ret == -1) {
    printf("customRobotState failed\n");
}
```

186 getCustomRobotState

int getCustomRobotState(unsigned long long *customBits, strIpAddress = "")

获取当前定制的推送信息。

参数：

customBits：定制的推送信息比特位。参见 customRobotState

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
unsigned long long stateFlags = 0;
ret = getCustomRobotState(&stateFlags, strIpAddress);
if (ret == -1) {
   printf("getCustomRobotState failed\n");
  } else {
   if (stateFlags & ROBOTSTATE_CUSTOM_JOINTPOS == 0) {
      printf("关节反馈位置推送信息未定制");
   } else {
      printf("关节反馈位置推送信息已定制");
   }
}
```

187 getTcpPoseByTcpName

int getTcpPoseByTcpName(const char *tcpName, double *coordinate,strIpAddress = "")

根据工具坐标系名称获取工具坐标系位姿。

参数：

tcpName：工具坐标系名称

cooridnate：位姿数组的首地址，数组长度为 6

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
const char *tcpName = "coor1";
double coorTcpPose[6];
int ret = getTcpPoseByTcpName(tcpName, coorTcpPose, strIpAddress);
if (ret == 0) {
   for (int i = 0; i < 6; i++) {
      printf("%f, ", coorTcpPose[i]);
   }
}
```

188 getTcpPoseByWorkPieceName

int getTcpPoseByWorkPieceName(const char *workPieceName, double *coordinate,

strIpAddress = "")
根据工件坐标系名称获取工件坐标位姿。 **参数：** workPieceName：工件坐标系名称 cooridnate：位姿数组的首地址，数组长度为 6 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。

调用示例：
const char* strIpAddress = "192.168.10.75";
const char *tcpName = "wpcoor1";
double coorTcpPose[6];
int ret = getTcpPoseByWorkPieceName(tcpName, coorTcpPose, strIpAddress);
if (ret == 0) {
for (int i = 0; i < 6; i++) {
printf("%f, ", coorTcpPose[i]);
}
}

189 getPayLoadByTcpName

int getPayLoadByTcpName(const char *tcpName, double *payload, strIpAddress = "")
根据工具坐标系名称获取工具坐标系负载信息。 **参数：** tcpName：工具坐标系名称 payload：负载数组首地址，数组长度为 10 strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。 **返回值：** 0：成功。 -1：失败。
调用示例： const char* strIpAddress = "192.168.10.75"; double payLoad[10]; int ret = getPayLoadByTcpName(tcpName, payLoad, strIpAddress);

```
if (ret == 0) {
    for (int i = 0; i < 10; i++) {
        printf("%f, ", payLoad[i]);
    }
}
```

190 setDefaultToolTcpCoordinate

int setDefaultToolTcpCoordinate(const char *tcpName, strIpAddress = "")

设置默认工具坐标系。

参数：

tcpName：工具坐标系名称

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
const char *tcpName = "coor1";
int ret = setDefaultToolTcpCoordinate(tcpName, strIpAddress);
if (ret == -1) {
    printf("setDefaultToolTcpCoordinate fail \n");
}
```

191 setDefaultWorkPieceTcpCoordinate

int setDefaultWorkPieceTcpCoordinate(const char *workPieceName, strIpAddress = "")

设置默认工件坐标系。

参数：

workPieceName：工件坐标系名称

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
```

```
const char *setWpName = "wpcoor1";
int ret = setDefaultWorkPieceTcpCoordinate(setWpName, strIpAddress);
if (ret == -1) {
   printf(" setDefaultWorkPieceTcpCoordinate fail \n");
}
```

192 getDefaultTcpCoordinate

```
int getDefaultTcpCoordinate(char *tcpName, strIpAddress = "")
```

获取默认的工具坐标系名称。

参数：

tcpName：工具坐标系名称数组首地址，数组长度不大于 256

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
char defaultTcpName[256];
int ret = getDefaultTcpCoordinate(defaultTcpName, strIpAddress);
if (ret == 0) {
   printf("default Tcp Name:%s\n", defaultTcpName);
}
```

193 getDefaultWorkPieceCoordinate

```
int getDefaultWorkPieceCoordinate(const char *workpieceName, strIpAddress = "")
```

获取默认的工件坐标系名称。

参数：

workpieceName：工件坐标系名称数组首地址，数组长度不少于 256

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
```

```
char defaultWpName[256];
int ret = getDefaultWorkPieceCoordinate(defaultWpName, strIpAddress);
if (ret == 0) {
   printf("defaultWpName:%s\n", defaultWpName);
}
```

194 setVelocityPercentValue

int setVelocityPercentValue(int value, strIpAddress = "")

设置速度百分比。

参数：

value：整形，百分比的值。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int value = 20;
int ret = setVelocityPercentValue(value, strIpAddress);
if (ret == 0) {
   printf("setVelocityPercentValue:%s\n", value);
}
```

195 switchRescueMode

int switchRescueMode (int scuWorkMode, strIpAddress = "")

切换安全处理模式，只有在安全处理模式下才能进行切换。

参数：

scuWorkMode：安全处理模式，取值为：12-安全零力模式，13-关节驱动模式，14-笛卡尔驱动模式。

strIpAddress：可选参数，需要控制机械臂的 IP 地址字符串，不填仅当只连接一台机械臂时生效。

返回值：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
int ret = enterRescueMode(strIpAddress);
if (ret == 0) {
   ret = switchRescueMode(12, strIpAddress);
}
```

附件 A：DianaApi 接口错误码

表 1：Diana API 接口错误码表

系统错误宏定义	错误码	说明
ERROR_CODE_WSASTART_FAIL	-1001	加载 windows 系统 socket 库失败
ERROR_CODE_CREATE_SOCKET_FAIL	-1002	创建 socket 对象失败
ERROR_CODE_BIND_PORT_FAIL	-1003	socket 绑定端口失败
ERROR_CODE_SOCKET_READ_FAIL	-1004	socket 的 select 调用失败
ERROR_CODE_SOCKET_TIMEOUT	-1005	socket 的 select 调用超时
ERROR_CODE_RECVFROM_FAIL	-1006	socket 接收数据失败
ERROR_CODE_SENDTO_FAIL	-1007	socket 发送数

		据失败
ERROR_CODE_LOST_HEARTBEAT	-1008	服务端的心跳连接丢失
ERROR_CODE_LOST_ROBOTSTATE	-1009	服务端信息反馈丢失
ERROR_CODE_GET_DH_FAILED	-1010	获取 DH 信息失败
ERROR_CODE_RELEASE_BRAKE_FAILED	-1011	打开抱闸失败
ERROR_CODE_HOLD_BRAKE_FAILED	-1012	关闭抱闸失败
ERROR_CODE_IP_ADDRESS_NOT_REGISTER	-1013	该 IP 机械臂尚未 initSrv
ERROR_CODE_ROBOTARM_OVERNUMBER	-1014	超过最大支持机械臂数
ERROR_CODE_SOCKET_OTHER_ERROR	-1015	其他 socket 连接错误
ERROR_CODE_JOINT_REGIST_ERROR	-2001	硬件错误
ERROR_CODE_EEPROM_READ	-2010	关节读 EEPROM 参数错误
ERROR_CODE_EEPROM_WRITE	-2011	关节写

		EEPROM参数错误
ERROR_CODE_LS_ENCODER_OVERSPEED	-2012	低速侧编码器反馈位置超速
ERROR_CODE_LS_ENCODER_FB_ERROR	-2013	低速侧编码器反馈数据错误
ERROR_CODE_MS_SINGAL_Z_ERROR	-2014	高速侧编码器Z信号异常
ERROR_CODE_THREE_PHASE_CURRENT	-2015	电机三相电流瞬时过流
ERROR_CODE_TORQUE_SENSOR_READ_ERROR	-2016	扭矩传感器读取故障
ERROR_CODE_COMMUNICATE_ERROR	-2101	底层通信失败
ERROR_CODE_LOST_HEART_WITH_DIANAROBOT_ERROR	-2102	与后台服务心跳断开
ERROR_CODE_CALLING_CONFLICT_ERROR	-2201	调用冲突
ERROR_CODE_COLLISION_ERROR	-2202	发生碰撞
ERROR_CODE_NOT_FOLLOW_POSITION_CMD	-2203	力控模式关节

		位置与指令发生滞后
ERROR_CODE_NOT_FOLLOW_TCP_CMD	-2204	力控模式 TCP 位置与指令发生滞后
ERROR_CODE_NOT_ALL_AT_OP_STATE	-2205	有关节未进入正常状态
ECODE_OUT_RANGE_FEEDBACK	-2206	关节角反馈超软限位
ECODE_EMERGENCY_STOP	-2207	急停已拍下
ECODE_NO_INIT_PARAMETER	-2208	找不到关节初始参数
ECODE_NOT_MATCH_LOAD	-2209	负载与理论值不匹配
ERROR_CODE_CANNOT_MOVE_WHILE_FREE_DRIVING	-2210	自由驱动模式不能执行其他运动
ERROR_CODE_CANNOT_MOVE_WHILE_ZERO_SPACE_FREE_DRIVING	-2211	零空间自由驱动模式下不能执行其

		他运动
ERROR_CODE_ROBOT_IN_VIRTUAL_WALL	-2214	有关节在虚拟墙内
ERROR_CODE_CONFLICT_TASK_RUNNING	-2215	运动任务冲突
ERROR_CODE_OUT_OF_PHYSICAL_RANGE_FEEDBACK	-2216	超出物理限位
ERROR_CODE_OUT_SOFT_RANGE_FEEDBACK	-2217	超出软限位
ERROR_CODE_CONVEYOR_NOT_ONLINE	-2218	传送带编码器不在线
ERROR_CODE_CONVEYOR_IS_TRACKED	-2219	传送带正在被跟踪，不能moveJ
ERROR_CODE_CONVEYOR_CANNOT_TRACK	-2220	开启跟踪传送带失败
ERROR_CODE_INPUT_OUT_OF_EXTREME_POSITION_RANGE	-2221	超出关节极限位置
ERROR_CODE_SLOPOVER_VIRTUAWALL	-2222	关节越过虚拟墙
ERROR_CODE_SLOPOVER_REDUCE_VIRTUAWALL	-2223	关节越过减速墙
ERROR_CODE_PLAN_ERROR	-2301	路径规划失败
ERROR_CODE_INTERPOLATE_POSITION_ERROR	-2302	位置模

		式插补失败
ERROR_CODE_INTERPOLATE_TORQUE_ERROR	-2303	阻抗模式插补失败
ERROR_CODE_SINGULAR_VALUE_ERROR	-2304	奇异位置
ERROR_CODE_ PLANNER_ERROR	-2305	规划失败
ERROR_CODE_ HOME_POSITION_ERROR	-2306	需要寻零
ERROR_CODE_ FATAL	-2307	严重错误(关节位置超出物理极限)
ERROR_CODE_ POS_LIMIT	-2308	位置超出限制
ERROR_CODE_ FORCE_LIMIT	-2309	关节力矩超出限制
ERROR_CODE_ SPEED_LIMIT	-2310	速度超出限制
ERROR_CODE_ ACC_LIMIT	-2311	加速度超出限制
ERROR_CODE_ JERK_LIMIT	-2312	加加速度超出限制
ERROR_CODE_ MOTION_LIMIT	-2313	位置超出限制
ERROR_CODE_ IK_TRACK	-2314	轨迹跟踪过程

		逆解求解失败
ERROR_CODE_ IK_GENERAL	-2315	通用位置逆解求解失败
ERROR_CODE_ PLAN_INPUT	-2316	轨迹规划输入错误
ERROR_CODE_ PLAN_MOVJ	-2317	关节空间轨迹规划失败
ERROR_CODE_ PLAN_MOVL	-2318	直线轨迹规划失败
ERROR_CODE_ PLAN_MOVC	-2319	圆弧轨迹规划失败
ERROR_CODE_ PLAN_BLEND	-2320	过渡轨迹规划失败
ERROR_CODE_ PLAN_SPDJ	-2321	SpeedJ 轨迹规划失败
ERROR_CODE_ PLAN_SPDL	-2322	SpeedL 轨迹规划失败
ERROR_CODE_ PLAN_SRVJ	-2323	ServoJ 轨迹规划失败
ERROR_CODE_ PLAN_SRVL	-2324	ServoL 轨迹规划失败

ERROR_CODE_ MOVE_UNKNOWN	-2325	未知运动类型或运动类型不匹配
ERROR_CODE_ MOVE_UNPLAN	-2326	轨迹未规划
ERROR_CODE_ MOVE_INPUT	-2327	轨迹插补输入错误
ERROR_CODE_ MOVE_INTERP	-2328	轨迹插补失败
ERROR_CODE_ PLAN_TRANSLATION	-2329	移动规划失败
ERROR_CODE_ PLAN_ROTATION	-2330	旋转规划失败
ERROR_CODE_ PLAN_JOINTS	-2331	关节规划失败
ERROR_CODE_ UNMATCHED_JOINTS_NUMBER	-2332	零空间自由驱动关节数不匹配
ERROR_CODE_ TCPCALI_FUTILE_WPS	-2333	示教点不合理
ERROR_CODE_ TCPCALI_FIT_FAIL	-2334	拟合 TCP 失败
ERROR_CODE_ DHCALI_FIT_WF_FAIL	-2335	DH 参数初始化世界坐标系失败

ERROR_CODE_ DHCALI_FIT_TF_FAIL	-2336	DH 参数初始化工具坐标系失败
ERROR_CODE_DHCALI_FIT_DH_FAIL	-2337	DH 参数拟合失败
ERROR_CODE_ DHCALI_INIT_FAIL	-2338	DH 参数初始化失败
ERROR_CODE_ SLFMOV_SINGULAR	-2339	零空间运动至奇异位置
ERROR_CODE_ SLFMOV_FUTILE	-2340	零空间运动在笛卡尔空间内无效
ERROR_CODE_SLFMOV_JNTLIM	-2341	零空间运动至关节限位
ERROR_CODE_ SLFMOV_SPDLIM	-2342	零空间运动至关节限位
ERROR_CODE_ SLFMOV_FAIL	-2343	零空间运动插补失败
ERROR_CODE_ SLFMOV_FFC_FAIL	-2344	零空间运动前

		馈补偿错误
ERROR_CODE_ LOADIDENT_INIT_FAIL	-2345	负载辨识初始化失败
ERROR_CODE_ LOADIDENT_UFB_FAIL	-2346	负载辨识更新反馈数据错误
ERROR_CODE_ LOADIDENT_FIT_FAIL	-2347	负载辨识失败
ERROR_CODE_ LOADIDENT_NONLOADED	-2348	未检测到有效负载
ERROR_CODE_PARAMETER_POINTER_EQUALS_NULLPTR	-2901	输入参数为空
ERROR_CODE_PARAMETER_POINTER_EQUALS_NAN_OR_INF	-2902	输入参数存在 nan 或者 inf
ERROR_CODE_ENTER_FORCE_MODE_ERROR	-2903	进入力控模式失败
ERROR_CODE_CANNOT_SET_VELOCITY_PERCENT_VALUE	-2904	设置速度百分比失败
ERROR_CODE_INPUT_OUT_OF_PHYSICAL_POSITION_RANGE	-2905	输入参数超出物理极限位置
ERROR_CODE_RESOURCE_UNAVAILABLE	-3001	参数错误
ERROR_CODE_DUMP_LOG_TIMEOUT	-3002	导出

		Log 文件超时
ERROR_CODE_DUMP_LOG_FAILED	-3003	导出 Log 文件失败
RESET_DH_FAILED	-3004	重置 DH 参数失败
ILLEGAL_PARAMETER	-3006	接口函数传入非法参数

注：表 1 中 ERROR_CODE_JOINT_REGIST_ERROR（-2001） 硬件错误和 ERROR_CODE_NOT_ALL_AT_OP_STATE(-2205)的 OP 状态错误需要通过调用 holdBrake() 合抱闸函数或重启硬件来清除错误。

附录 B：如何确保运动学逆解唯一

Diana 机械臂为七自由度机械臂，由于多了一个冗余自由度，理论上存在无数多组逆解(根据末端位姿求解关节角)，在实际应用中，当执行逆解运算或者进行笛卡尔空间运动时，有可能出现逆解不唯一的情况。为了确保逆解的唯一性，可采取如下解决方案。

第一种情况：已知其中某个路点对应的关节角

例：已知 A 点关节角 Joints_A 和 B 点位姿 Pose_B，机械臂在两点之间进行往复运动。

解决方案：向目标点 A 运动时，调用 moveJToTarget 或 moveLToTarget 函数。

```
const char* strIpAddress = "192.168.10.75";
double Joints_A[7] = {0.000000, 0.523599, 0.000000, 1.570796, 0.000000, -0.872665, 0.000000};    // A 点关节角
double pose_B[6] = {0.5, 0.5, 0.5, 0, 0, 0};  // B 点位姿
double velL = 0.2, accL = 0.8;      // 直线运动的速度与加速度
moveJToTarget(Joints_A, velL, accL,strIpAddress);
wait_move(strIpAddress);
int count = 10;
for (int i = 0; i < count; i++)
{
    // 调用 moveJToTarget 或 moveLToTarget 函数移动至目标点 A
    moveLToTarget(Joints_A, velL, accL, strIpAddress);
    wait_move(strIpAddress);
    // 调用 moveJToPose 或 moveLToPose 函数移动至目标点 B
    moveLToPose(pose_B, velL, accL, nullptr, strIpAddress);
    wait_move(strIpAddress);
}
```

第二种情况：所有路点对应的关节角均未知

例：已知 A 点位姿 Pose_A 和 B 点位姿 Pose_B，机械臂在两点之间进行往复运动。

解决方案：首先在 A 点和 B 点附近分别示教一个参考点，并记录下参考点位下的机械臂关节角 q_ref_A 和 q_ref_B，然后利用 inverse_ext 函数，求解目标位姿下相对于参考点关节角距离最近的逆解，最后调用 moveJToTarget 或 moveLToTarget 函数进行运动。

```
const char* strIpAddress = "192.168.10.75";
double Pose_A[6] = {0.5, 0.5, 0.5, 0, 0, 0};
double Pose_B[6] = {0.4, 0.6, 0.2, 0, 0, 0};
// 示教两个参考点位并记录下关节角 q_ref_A, q_ref_B
double q_ref_A [7] = {-0.645772, 0.261799, -0.157080, 1.675516, 0.052360, -1.186824, -0.802851};
double q_ref_B[7] = {-0.645772, 0.261799, -0.157080, 1.675516, 0.052360, -1.186824, -0.802851};
double velJ = 0.25, accJ = 1.0;      // 关节空间运动的速度与加速度
double velL = 0.1, accL = 0.4;       // 直线运动的速度与加速度
int count = 10;
for ( int i = 0; i < count; i++)
{
    // 调用 inverse_ext 函数, 根据 q_ref_A 计算 Pose_A 所对应的关节角 Joints_A
    double Joints_A [7] = {0.0};
    inverse_ext(q_ref_A, Pose_A, Joints_A);
    // 调用 moveJToTarget 或 moveLToTarget 函数移动到目标点 A
    moveJToTarget(Joints_A, velJ, accJ);
    wait_move(strIpAddress);
    // 调用 inverse_ext 函数, 根据 q_ref_B 计算 Pose_B 所对应的关节角 Joints_B
    double Joints_B [7] = {0.0};
    inverse_ext(q_ref_B, Pose_B, Joints_B);
    // 调用 moveJToTarget 或 moveLToTarget 函数移动到目标点 B
    moveLToTarget(Joints_B, velL, accL);
    wait_move(strIpAddress);
}
```