北京思灵机器人科技有限责任公司

Diana API 说明文档

（C 语言）

此页为空白页

目录

修订历史

版本	修改内容	修订人	修订时间
V1.0	创建	孟庆婷	
V2.0	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
V2.1	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。	孟庆婷	2020-9-9

	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。		
V2.2	1. 新增四个接口：ToAxis，ToRPY，Homogeneous2Pose，Pose2Homogeneous。	孟庆婷	2020-9-12
V2.3	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
V2.4	1. 新增接口： speedJ_ex，speedL_ex，servoJ_ex，servoL_ex	孟庆婷	2020-10-12
V2.5	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
V2.6	1. 修改 save 接口名为 saveEnvironment	孟庆婷	2020-11-25
V2.7	2. 新增 IO 相关接口：71-81	李漠	2020-11-27
V2.8	1. 新增接口 getDH、getOriginalJointTorque、getJacobiM atrix 2. 修改函数 getJointTorque 含义	石国庆	2020-11-27
V2.9	1. 新增 resetDH 2. 更新错误码	孟庆婷	2020-12-10
V3.0	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
V3.1	新增路点变量接口：	李漠	2021-04-25

	setWayPoint、getWayPoint、addWayPoint、deleteWayPoint		
V3.2	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
V3.3	1. API 支持同时控制多台机械臂	石国庆	2021-06-25
V3.4	1. 修正了一些书写错误	石国庆	2021-07-15
V3.5	1. 补充 API 函数及调整 API 顺序	石国庆	2021-09-16

该操作库函数的所有输入输出参数，均采用国际量纲，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。

1 initSrvNetInfo

int initSrvNetInfo(srv_net_st* pinfo)
初始化网络结构体，这会使得全部端口随机分配 **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, 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：void 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），关节角电流当前值数组（jointCurrent），关节角扭矩数组（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)
{
    static int staCnt = 1;
    if((staCnt++ % 1000 == 0) && pinfo)
    {
        for(int i = 0; i < 7; ++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 * 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();
srv_net_st * pinfo1 = new srv_net_st();
memset(pinfo ->SrvIp, 0x00, sizeof(pinfo ->SrvIp));
memset(pinfo1 ->SrvIp, 0x00, sizeof(pinfo1 ->SrvIp));
memcpy(pinfo ->SrvIp, "172.168.10.75", strlen("172.168.10.75"));
memcpy(pinfo1 ->SrvIp, "172.168.10.76", strlen("172.168.10.76"));
pinfo->LocHeartbeatPort = 0;
pinfo->LocRobotStatePort = 0;
pinfo->LocSrvPort = 0;
pinfo1->LocHeartbeatPort = 0;
pinfo1->LocRobotStatePort = 0;
pinfo1->LocSrvPort = 0;
int ret = initSrv(errorControl, logRobotState,  pinfo);
if(ret < 0)
{
      printf("172.168.10.75 initSrv failed! Return value = %d\n", ret);
}
ret = initSrv(errorControl, logRobotState,  pinfo1);
if(ret < 0)
{
      printf("172.168.10.76 initSrv failed! Return value = %d\n", ret);
}
if(pinfo)
{
   delete pinfo;
   pinfo = nullptr;
}
destroySrv();
```

3 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);
}

4 setPushPeriod

int setPushPeriod(intPeriod,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”; setPushPeriod(10,strIpAddress);

5 moveTCP

int moveTCP(d, v, a, active_tcp=nullptr, strIpAddress = ““)
手动移动指定 IP 地址的机械臂末端中心点。该函数会立即返回，停止运动需要调用 stop

函数，现支持在不同工具坐标系下移动，由传入的 active_tcp 决定。

参数：

d：表示移动方向的枚举类型，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：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：

```
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);
}
```

6 rotationTCP

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

使指定的 IP 地址的机械臂绕末端中心点变换位姿。该函数会立即返回，停止运动需要调用 stop 函数。现支持在不同工具坐标系下旋转，由传入的 active_tcp 决定。

参数：

d：表示移动方向的枚举类型，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：rad/s。

a：加速度，单位：rad/s^2。

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：

```
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);
}
```

7 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);
}

8 moveJToTarget

int moveJToTarget(joints, v, a, strIpAddress = ““)
控制指定 IP 地址的机械臂以七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。 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[7] = {0.0,0.0,0.0,0.0,0.0,0.0,0.0};
double vel = 0.2;
double acc = 0.4;
int ret = moveJToTarget(joints, vel, acc, strIpAddress);
if(ret < 0)
{
      printf("moveJToTarget failed! Return value = %d\n", ret);
}
wait_move(strIpAddress);
```

9 moveJToPose

int moveJToPose(pose, v, a, active_tcp=nullptr, strIpAddress = "")

控制指定 IP 地址的机械臂以工具中心点位姿为终点的 moveJ。该函数会立即返回，停止运动需要调用 stop 函数。pose 可以相对于不同工具坐标系，由传入的 active_tcp 决定。

参数：

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

10 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[7] = {0.0,0.0,0.0,0.0,0.0,0.0,0.0};
double vel = 0.2;
double acc = 0.4;

```
const char* strIpAddress = “192.168.10.75”;
int ret = moveJ (joints, vel, acc, strIpAddress);
if(ret < 0)
{
    printf(“moveJ failed! Return value = %d\n”, ret);
}
wait_move(strIpAddress);
```

11 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;
const char* strIpAddress = “192.168.10.75”;
int ret = moveL (poses, vel, acc, nullptr, strIpAddress);
if(ret < 0)
{
printf(“moveL failed! Return value = %d\n”, ret);
}
wait_move(strIpAddress);

12 moveLToTarget

int moveLToTarget(joints, v, a, strIpAddress = ““)
控制指定 IP 地址的机械臂以七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 7。

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：

```
double joints[7] = {0.0,0.0,0.0,0.0,0.0,0.0,0.0};
double vel = 0.2;
double acc = 0.4;
const char* strIpAddress = “192.168.10.75”;
int ret = moveLToTarget(joints, vel, acc);
if(ret < 0)
{
    printf(“moveLToTarget failed! Return value = %d\n”, ret);
}
wait_move(strIpAddress);
```

13 moveLToPose

int moveLToPose(pose, v, a, active_tcp=nullptr, strIpAddress = “”)

控制指定 IP 地址的机械臂以工具中心点位姿为终点的 moveL。该函数会立即返回，停止运动需要调用 stop 函数。pose 可以相对于不同工具坐标系，由传入的 active_tcp 决定

参数：

pose：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：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 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;
const char* strIpAddress = “192.168.10.75”;
int ret = moveLToPose(poses, vel, acc, nullptr, strIpAddress);
if(ret < 0)
{
printf(“moveLToPose failed! Return value = %d\n”, ret);
}
wait_move(strIpAddress);

14 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 为可选项，如果提供了 t 值，控制指定 IP 地址的机械臂将在 t 时间后减速。如果没有提供时间 t 值，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。 **参数：** speed：关节角速度数组首地址，数组长度为 7。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[7] = {0.0,0.0,0.0,0.0,0.0,0.0,0.5};
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);
}
```

15 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，笛卡尔空间下直线运动。时间 t 为可选项，时间 t 是可选项，如果提供了 t 值，机器人将在 t 时间后减速。如果没有提供时间 t 值，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。现支持在不同工具坐标系下移动，由传入的 active_tcp 决定。

参数：

speed：工具空间速度，数组长度为 6,其中前 3 个单位为 m/s，后 3 个单位为 rad/s。

a：加速度，数组长度为 2，其中第 1 个单位为 m/s^2，第 2 个单位为 rad/s^2。

t：时间，单位：s。

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 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);
}
```

16 freeDriving

int freeDriving(enable, 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：** enable：bool 变量，是否进入零力驱动模式，true 表明进入零力驱动，false 为退出零力驱动进入正常模式。只有在正常模式下，才可以控制机器人运动。 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(true, strIpAddress);
if(ret < 0)
{
printf("freeDriving failed! Return value = %d\n", ret);
}

17 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。对应于 UR 的 stopL，stopJ 指令。 **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);
}
```

18 forward

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

正解函数，针对指定 IP 地址机器人，由传入的关节角都计算出的正解 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 7。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[7] = {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);
}
```

19 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] = {0.64221, 0.0, 0.9403, 0.0, 0.0, 0.0};
double joints[7] = {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);
}

20 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 7。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[7] = {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);
}
```

21 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 7。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[7] = {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);
}
```

22 getJointCurrent

int getJointCurrent(joints, strIpAddress = “”)
获取当前关节电流。 **参数：** joints：输出关节角度数组首地址，数组长度为 7。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[7] = {0.0};
strIpAddress=”192.168.10.75”
int ret = getJointCurrent(joints, strIpAddress);
if(ret < 0)
{
printf(“getJointCurrent failed! Return value = %d\n”, ret);
}

23 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：输出关节阻抗数组首地址，数组长度为 7。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[7] = {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);
}
```

24 getTcpPos

int getTcpPos(pose, strIpAddress = "")

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

参数：

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

25 getTcpExternalForce

double getTcpExternalForce(strIpAddress = "")

获取指定 IP 地址机械臂末端实际感受到的力大小，末端可被 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);

26 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);
}

27 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);
}
```

28 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);
}
```

29 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();

30 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(“%s\n”,msg);
```

31 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);

32 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);

33 setDefaultActiveTcp

int setDefaultActiveTcp(default_tcp , strIpAddress = ““)
设置指定 IP 地址机械臂的默认工具坐标系。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”;
double default_tcp[16] = {0.433,0.250,-0.866,0.0,0.500,-0.866,0.0,0.0,-0.750,-0.433,
-0.500,0.0,-0.231,0.155,0.934,1.000};
int ret = setDefaultActiveTcp(default_tcp, strIpAddress);
if(ret < 0)
{
    printf(“setDefaultActiveTcp failed! Return value = %d\n”, ret);
}
```

34 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”);
}
```

35 getTcpForce

int getTcpForce(forces, strIpAddress = ““)

获取指定 IP 地址机械臂的 TCP 末端六维力，末端可被 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”);
}
```

36 getJointForce

int getJointForce(forces, strIpAddress = ““)

获取指定 IP 地址机械臂的轴空间七个关节所受力。

参数：

forces：七关节轴力矩数组的首地址，数组长度为 7。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[7];
const char* strIpAddress = “192.168.10.75”;
int ret = getJointForce (jointForce, strIpAddress);
if(ret == 0)
{
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]);
}
else
{
printf(“get joint force failed \n”);
}

37 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”);
}
```

38 initDHCali

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

根据输入的关节角以及末端位置数组计算指定 IP 地址机械臂的 DH 参数。

参数：

tcpMeas：输入参数。TCP 位置数据数组的首地址，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 7 * nrSets，每组数据为各关节角位置信息[q1~q7]，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：

```
unsigned int rowNo_refData = 32
double jntMeas[224] = {…};
double tcpMeas[96] = {…};
const char* strIpAddress = “192.168.10.75”;
if (initDHCali(tcpMeas, jntMeas, rowNo_refData, strIpAddress) != 0)
{
    printf("DHCaliFcns.InitDHCali falid!\n");
}
```

39 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 28。每七个数为一组，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。

返回值：

0：获取成功。

1：获取结果精度可能较低。

-1：获取失败，发生异常。

调用示例：

```
double rDH[28], wRT[6], tRT[3],confid[2];
const char* strIpAddress = “192.168.10.75”;
if (getDHCaliResult(rDH, wRT, tRT, confid[2],strIpAddress) < 0)
{
    printf("DHCaliFcns.getDHCaliResult falid.\n");
}
```

40 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 7。

alpha：输入参数。各关节的 alpha 参数数组的首地址，数组长度为 7。

d：输入参数。各关节的 d 参数数组的首地址，数组长度为 7。

theta：输入参数。各关节的 theta 参数数组的首地址，数组长度为 7。

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[7] = {…}, alpha[7] = {…}, d[7] = {…}, theta[7] = {…};
const char* strIpAddress = “192.168.10.75”;
if (setDH(a, alpha, d, theta, strIpAddress) < 0)
{
printf("DHCaliFcns.setDH falid.\n");
}

41 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");
}

42 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");
}

43 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。 1：paused。 2：idle。 3：free-driving。

4：zero-space-free-driving。

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 if (3 == state)
{
    printf("\t[robot state]: free-driving\n");
}
else
{
    printf("\t[robot state]: zero-space-free-driving\n");
}
```

44 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");
}

45 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。七关节轴力矩阈值数组的首地址，数组长度为 7。 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};
if (setJointCollision(collision, strIpAddress) < 0)
{
printf("Diana API setJointCollision failed!\n");
}

46 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力矩阈值。 **参数：** collision：输入参数。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");
}
```

47 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：表达力的方向的数组首地址，数组长度为 3。

dblForceValue：力大小。单位：N。推荐取值范围（1N~100N）。

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");
}
```

48 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");
}
```

49 setDefaultActiveTcpPose

int setDefaultActiveTcpPose (arrPose,strIpAddress = “”)
设置指定 IP 地址机械臂默认的工具坐标系位姿。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：** 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");
}

50 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 的合力矩阈值。 **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");
}
```

51 setJointImpedance

int setJointImpedance (arrStiff, arrDamp,strIpAddress = ““)
设置指定 IP 地址机械臂各关节阻抗参数，包含钢度 Stiffiness 和阻尼 Damping 的数据。 **参数：** arrStiff：表示各关节钢度 Stiffiness 的数组的首地址，数组长度为 7。 arrDamp：表示各关节阻尼 Damping 的数组的首地址，数组长度为 7。 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, 1000, 500, 1000, 1000};
double arrDamp[7] = { 50, 40, 15, 30, 9.88, 3.4, 1.0};
const char* strIpAddress = “192.168.10.75”;
if (setJointImpedance (arrStiff, arrDamp, strIpAddress) < 0)
{
printf("Diana API setJointImpedance failed!\n");
}

52 getJointImpedance

int getJointImpedance (arrStiff, arrDamp,strIpAddress = ““)
获取指定 IP 地址机械臂各关节阻抗参数，包含钢度 Stiffiness 和阻尼 Damping 的数据。 **参数：** arrStiff：表示各关节钢度 Stiffiness 的数组的首地址，数组长度为 7，用于接收获取到的值。

arrDamp：表示各关节阻尼 Damping 的数组的首地址，数组长度为 7，用于接收获取到的值。 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 arrStiff [7] = {0};
double arrDamp [7] = {0};
const char* strIpAddress = “192.168.10.75”;
if (getJointImpedance (arrStiff, arrDamp, strIpAddress) < 0)
{
printf("Diana API getJointImpedance failed!\n");
}

53 setCartImpedance

int setCartImpedance (arrStiff, arrDamp, 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，各维度钢度 Stiffiness 的数组的首地址，数组长度为 6。 arrDamp：表示笛卡尔空间，各维度阻尼 Damping 的数组的首地址，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： double arrStiff[6] = { 1000, 1000, 1000, 500, 500, 500}; double arrDamp[6] = { 10, 10, 10, 5, 5, 5}; const char* strIpAddress = “192.168.10.75”; if (setCartImpedance(arrStiff, arrDamp, strIpAddress) < 0)

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

54 getCartImpedance

int getCartImpedance (arrStiff, arrDamp, strIpAddress = “”)

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

参数：

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

arrDamp：表示笛卡尔空间，各维度阻尼 Damping 的数组的首地址，数组长度为 6，用于接收获取到的值。

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 arrStiff [6] = {0};
double arrDamp [6] = {0};
const char* strIpAddress = “192.168.10.75”;
if (getCartImpedance (arrStiff, arrDamp, strIpAddress) < 0)
{
    printf("Diana API getCartImpedance failed!\n");
}
```

55 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
{
zeroSpaceFreeDriving(false, strIpAddress);
}

56 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。

57 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 7。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：

```
unsigned int path_id = 0;
printf("start test moveJ path.\n");
const char* strIpAddress = “192.168.10.75”;
createPath(1, 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);
```

58 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

7。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。每个关节位移的一半为交融半径最大值。 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：
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);

59 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。

60 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。

61 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

62 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

63 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 homogeneous2Pose 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 homogeneous2Pose got: -0.231000, 0.155000, 0.934000, 3.141593, 1.047198, 0.523599

64 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:
0.433013, 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

65 enableTorqueReceiver

int enableTorqueReceiver(bEnable, trIpAddress = ““)
在指定 IP 地址机械臂上，打开或关闭实时扭矩的接受。 **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);
}

66 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 7 的数组。 t:持续时间，单位 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”;
double dblTorque[7] = {0.0};
double t = 1000;
int ret = sendTorque_rt(dblTorque,t,strIpAddress);
if(ret < 0)
{
printf(“sendTorque_rt failed! Return value = %d\n”, ret);

```
}
```

67 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);
}
```

68 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);
}
```

69 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。

参数：

joints：目标关节角位置数组的首地址，数组长度为 7。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;
target1={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();
```

70 servoL

int servoL (pose, time, look_ahead_time, gain, scale, active_tcp=nullptr, strIpAddress = “”)
笛卡尔空间内，伺服指定 IP 地址机械臂到指定位姿。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。现支持在不同工具坐标系下运动，由传入的 active_tcp 决定。 **参数：** pose：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：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：
const double PI=3.141592653;
const char* strIpAddress = "192.168.10.75";
double joints[7]={0, 0, 0, PI/2, 0, -PI/2, 0};
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, nullptr, strIpAddress);
    if(ret < 0)
        break;
    sleep(0.001);
}
stop(strIpAddress);
```

71 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。 **参数：** joints：目标关节角位置数组的首地址，数组长度为 7。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();
```

72 servoL_ex

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

笛卡尔空间内，伺服指定 IP 地址机械臂到指定位姿优化版。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。现支持在不同工具坐标系下运动，由传入的 active_tcp 决定。

参数：

pose：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 servoL；值为 false 则无需反馈直接返回。

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：

```
const double PI=3.141592653;
const char* strIpAddress = "192.168.10.75";
```

```
double joints[7]={0, 0, 0, PI/2, 0, -PI/2, 0};
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_ex(pose, 0.01, 0.1, 300,true,nullptr,strIpAddress);
   if(ret < 0)
      break;
   sleep(0.001);
}
stop(strIpAddress);
```

73 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 为可选项，时间 t 是可选项，如果提供了 t 值，机器人将在 t 时间后减速。如果没有提供时间 t 值，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：关节角速度数组首地址，数组长度为 7。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[7] = {0.0,0.0,0.0,0.0,0.0,0.0,0.5};
```

```
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);
}
```

74 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。时间 t 为可选项，时间 t 是可选项，如果提供了 t 值，机器人将在 t 时间后减速。如果没有提供时间 t 值，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。现支持在不同工具坐标系下移动，由传入的 active_tcp 决定。

参数：

speed：工具空间速度，数组长度为 6,其中前 3 个单位为 m/s，后 3 个单位为 rad/s。

a：加速度，单位：m/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：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 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);
}
```

75 dumpToUDisk

int dumpToUDisk(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 dumpToUDiskEx 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：** 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("192.168.10.75") 4. 拔下 u 盘查看

76 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 7 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。 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[7] = {0.0};
double pose[6] = {0.64221, 0.0, 0.9403, 0.0, 0.0, 0.0};
double joints[7] = {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);
}

77 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 7 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[7] = {0.0};
int ret = getJointLinkPos(joints,strIpAddress);
if(ret < 0)

```
{
    printf("getJointLinkPos failed! Return value = %d\n", ret);
}
```

78 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_PATH，该模式下路径中可以添加 moveL、moveJ、moveC 对应的路径点，路径点用 7 个关节角度确定。 1：表示 MOVEP_JOINT_PATH，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 7 个关节角度确定。 2:表示 NORMAL_POSE_PATH ，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_PATH，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。

79 addMoveLByTarget

int addMoveLByTarget (complex_path_id, target_joints, vel, acc, blendradius, 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 7。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。

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 dblFirstPosition [7] = {……};
double dblSecondPosition [7] = {……};
unsigned int uintComplexPathId = 0;
const char* strIpAddress = “192.168.10.75”;
createComplexPath(NORMAL_JOINT_PATH, uintComplexPathId, strIpAddress);
addMoveLByTarget(uintComplexPathId, dblFirstPosition, 0.2, 0.4, 0, strIpAddress);
addMoveLByTarget(uintComplexPathId, dblSecondPosition, 0.2, 0.2, 0, strIpAddress);
runComplexPath(uintComplexPathId, strIpAddress);
destroyComplexPath(uintComplexPathId, strIpAddress);
wait_move(strIpAddress);
```

80 addMoveLByPose

```
int addMoveLByPose(complex_path_id, target_pose, vel, acc, blendradius, 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。

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] = {……};
double dblSecondPose [6] = {……};
unsigned int uintComplexPathId = 0;
const char* strIpAddress = “192.168.10.75”;
createComplexPath(NORMAL_POSE_PATH, uintComplexPathId, strIpAddress);
addMoveLByPose(uintComplexPathId, dblFirstPose, 0.2, 0.4, 0, strIpAddress);
addMoveLByPose(uintComplexPathId, dblSecondPose, 0.2, 0.2, 0, strIpAddress);
runComplexPath(uintComplexPathId, strIpAddress);
destroyComplexPath(uintComplexPathId, strIpAddress);
wait_move(strIpAddress);
```

81 addMoveJByTarget

```
int addMoveJByTarget (complex_path_id, target_joints, vel_percent, acc_percent, blendradius_percent, 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 7。单位：rad。

vel_percent：moveJ 移动到目标路点的速度百分比，相对于系统软限位中设定最大速度的百分比。

acc_percent：moveJ 移动到目标路点的加速度百分比，相对于系统中设定最大加速度的百分比。

blendradius _percent：交融半径百分比，相对于系统软限位中设定最大交融半径的百分

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：

```
double dblFirstPosition [7] = {……};
double dblSecondPosition [7] = {……};
unsigned int uintComplexPathId = 0;
const char* strIpAddress = “192.168.10.75”;
createComplexPath(NORMAL_JOINT_PATH, uintComplexPathId, strIpAddress);
addMoveJByTarget (uintComplexPathId, dblFirstPosition, 0.2, 0.4, 0, strIpAddress);
addMoveJByTarget(uintComplexPathId, dblSecondPosition, 0.2, 0.2, 0, strIpAddress);
runComplexPath(uintComplexPathId, strIpAddress);
destroyComplexPath(uintComplexPathId, strIpAddress);
wait_move(strIpAddress);
```

82 addMoveJByPose

int addMoveJByPose (complex_path_id, target_pose, vel_percent, acc_percent, blendradius_percent, 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。相对于系统软限位中设定最大速度的百分比。

acc_percent：moveJ 移动到目标路点的加速度百分比。相对于系统软限位中设定最大加速度的百分比。

blendradius _percent：交融半径百分比。相对于系统设定最大交融半径的百分比。

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：
double dblFirstPose [6] = {……};
double dblSecondPose [6] = {……};
unsigned int uintComplexPathId = 0;
const char* strIpAddress = “192.168.10.75”;
createComplexPath(NORMAL_POSE_PATH, uintComplexPathId. strIpAddress);
addMoveJByPose(uintComplexPathId, dblFirstPose, 0.2, 0.4, 0. strIpAddress);
addMoveJByPose(uintComplexPathId, dblSecondPose, 0.2, 0.2, 0. strIpAddress);
runComplexPath(uintComplexPathId. strIpAddress);
destroyComplexPath(uintComplexPathId. strIpAddress);
wait_move(strIpAddress);

83 addMoveCByTarget

int addMoveCByTarget (complex_path_id, pass_joints, target_joints, vel, acc, blendradius, ignore_rotation, 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 7。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 7。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 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 dblFirstPosition [7] = {……};
double dblSecondPosition [7] = {……};
unsigned int uintComplexPathId = 0;
const char* strIpAddress = “192.168.10.75”;
createComplexPath(NORMAL_JOINT_PATH, uintComplexPathId, strIpAddress);
addMoveCByTarget (uintComplexPathId, dblFirstPosition, dblSecondPosition, 0.2, 0.4, 0,true, strIpAddress);
runComplexPath(uintComplexPathId, strIpAddress);
destroyComplexPath(uintComplexPathId, strIpAddress);
wait_move(strIpAddress);
```

84 addMoveCByPose

```
int addMoveCByPose (complex_path_id, pass_pose, target_pose, vel, acc, blendradius, ignore_rotation, 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

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] = {……};
double dblSecondPose [6] = {……};
unsigned int uintComplexPathId = 0;
const char* strIpAddress = “192.168.10.75”;
createComplexPath(NORMAL_POSE_PATH, uintComplexPathId, strIpAddress);
addMoveCByPose(uintComplexPathId, dblFirstPose, dblSecondPose, 0.2, 0.4, 0,true, strIpAddress);
runComplexPath(uintComplexPathId, strIpAddress);
destroyComplexPath(uintComplexPathId, strIpAddress);
wait_move(strIpAddress);

85 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。

86 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。

87 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 参数等所用可通过 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”);
}

88 dumpToUDiskEx

int dumpToUDiskEx (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 dumpToUDiskEx 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：单位为秒，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： 5. 系统开机 6. 插入 u 盘到控制箱 7. 调用 Api 函数 dumpToUDiskEx(-1,’192.168.10.75’) 8. 拔下 u 盘查看

89 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：表达力的方向的数组首地址，数组长度为 3。 dblForceValue：力大小。单位：N。推荐取值范围（1N~100N）。 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 ：可选参数，坐标系对应的位姿向量，大小为 6 的数组，不传则默认为基坐标系 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：
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_ex(dblForceDir, dblForceValue, dblMaxVel, dblMaxOffset,strIpAddress) <
0)
{
    printf("Diana API enterForceMode_ex failed!\n");
}
```

90 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”);
}

91 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”);
}
```

92 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”);
}
```

93 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”);
}

94 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”);
}
```

95 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”);
}
```

96 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”);
}

97 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 7 alphaDH:连杆转角的数组，长度为 7 dDH:连杆偏距的数组，长度为 7

thetaDH:连杆的关节角的数组，长度为 7 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[7],alpha[7],d[7],theta[7];
const char* strIpAddress = “192.168.10.75”;
int ret = getDH(a,alpha,d,theta, strIpAddress);
if(ret < 0)
{
printf(“getDH failed!\n”);
}

98 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 7 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[7];
const char* strIpAddress = “192.168.10.75”;
int ret = getOriginalJointTorque(torques, strIpAddress);
if(ret < 0)
{
printf(“getOriginalJointTorque failed!\n”);

```
}
```

99 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 42

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 [42];
const char* strIpAddress = “192.168.10.75”;
int ret = getJacobiMatrix (matrix_jacobi, strIpAddress);
if(ret < 0)
{
    printf(“getJacobiMatrix failed!\n”);
}
```

100 resetDH

int resetDH(strIpAddress = “”)

重置指定 IP 地址机械臂用户自定义的 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");
}
```

101 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。

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");
}
```

102 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。

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”);
}
```

103 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”);
}
```

104 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”);
}
```

105 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

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”);
}
```

106 pauseProgram

int pauseProgram(strIpAddress = “”)
暂停指定 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 = pauseProgram(strIpAddress);
if(ret == 0)
{
printf(“All program is paused!\n”);
}

107 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”);

```
}
```

108 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”);
}

109 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：成功。 -1：失败。

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");
}
```

110 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");
}
```

111 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，值与含义如下：

0：无碰撞检测

1：关节空间检测

2：笛卡尔空间检测

3： 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(3, strIpAddress);
if(ret < 0)
{
    printf(“setCollision failed!\n”);
}
```

112 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。

113 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。

114 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。

115 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。

116 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：
const char* strIpAddress = “192.168.10.75”;
double double_array[4] = {0.0};
int8_t int8_array[3] = {0};
setPushPeriod(10);
mappingDoubleIO("board", "ai0", 0);
mappingDoubleIO("board", "ai1", 1);
mappingDoubleVariant("test_value_0", 2);
mappingDoubleVariant("test_value_1", 3);
mappingInt8IO("board", "di0", 0);
mappingInt8IO("board", "di1", 1);
mappingInt8IO("board", "di2", 2);
int ret = setMappingAddress(double_array, 4, int8_array, 3);
if(ret < 0)
{
printf(“setMappingAddress failed!\n”);
}

```
//获取 AI 或 DI 值
while (true){
        int c = getchar();
        if (c == 'q')
            break;
        lockMappingAddress();
        printf("di[0-3] = {%d, %d, %d,}\n", int8_array[0], int8_array[1], int8_array[2]);
        printf("ai[2] = {%f, %f}\n", double_array[0], double_array[1]);
        printf("test_value_0=%f, test_value_1=%f\n", double_array[TEST_VALUE_0],
double_array[TEST_VALUE_1]);
        unlockMappingAddress();
    }
```

117 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。

118 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。

119 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：** 非负数：关节数目。 -1：失败。
调用示例： const char* strIpAddress = “192.168.10.75”; int count = getJointCount(strIpAddress); printf(“Joint Count = %d\n”,count);

120 getWayPoint

int getWayPoint(strWayPointName,dblTcppos,dblJoints, 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。 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。

121 setWayPoint

int setWayPoint(strWayPointName,dblTcppos,dblJoints, 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。 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： string strWayPointName = “setName”; const char* strIpAddress = “192.168.10.75”; double dblTcppos[6] = {0.1,0.1,0.1,0, 0, 0}; double dblJoint[7]={0.1,0.1,0.1,0.1,0.1,0.1,0.1}; setWayPoint(strWayPointName .c_str(),dblTcppos,dblJoint, strIpAddress);

122 addWayPoint

int addWayPoint(strWayPointName,dblTcppos,dblJoints, 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。 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。

123 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： string strWayPointName = “deleteName”; const char* strIpAddress = “192.168.10.75”; deleteWayPoint(deleteName.c_str(),strIpAddress);

124 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);
}
```

125 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");
}
```

126 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);
}
```

127 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，大小为 7 的数组，单位 rad/s

dblJointAcc：输入参数，各关节运动的加速度，大小为 7 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：

0：成功。

-1：失败。

调用示例：

```
const char* strIpAddress = "192.168.10.75";
double dblJointVel[7]={6.28,6.28,6.28,6.28,6.28,6.28,6.28};
double dblJointAcc[7]={6.28,6.28,6.28,6.28,6.28,6.28,6.28};
int ret = zeroSpaceManualMove(1,dblJointVel,dblJointAcc,strIpAddress);
if (ret < 0){
    printf("Diana API zeroSpaceManualMove failed!\n");
}
else{
    M_SLEEP(1);
}
```

128 moveTcp_ex

int moveTcp_ex(c,d, v, a, strIpAddress = ““)

手动移动指定 IP 地址的机械臂末端中心点。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);
}
```

129 rotationTCP_ex

int rotationTCP_ex(c, d, v, a, strIpAddress = “”)

使指定的 IP 地址的机械臂绕末端中心点变换位姿。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：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);
}
```

130 setExternalAppendTorCutoffFreq

int setExternalAppendTorCutoffFreq(dblFreq, strIpAddress = “”)
针对指定 IP 地址机械臂，设置其附加力矩的滤波截止频率 **参数：** dblFreq：输入参数，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);
}

131 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}; double dstPose[6] = {0};

```
poseTransform(srcPose,srcMatrixPose,srcMatrixPose,dstPose);
for(int i = 0;i<7;++i)
    printf("%lf ",dstPose[i]);
```

132 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(false,"192.168.10.75");
if(ret < 0){
    printf(“setEndKeyEnableState failed! Return value = %d\n”, ret);
}
```

133 updateForce

int updateForce(dblForceDirection,dblForceValue,strIpAddress = ““)

针对指定 IP 地址机械臂，在力控模式下，实时改变力指令的大小与方向

参数：

dblForceDirection：输入参数，力的方向，大小为三的数组，分别表示下 x,y,z 方向的分量

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};
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(dblForceDirection,dblForceValue,strIpAddress);
        M_SLEEP(1);
    }
    int intExitMode = 0;
    if (leaveForceMode(intExitMode.strIpAddress) < 0)
    {
        printf("Diana API leaveForceMode failed!\n");
    }
}
```

134 updateForce_ex

int updateForce_ex(dblForceDirection,dblForceValue,dblActiveTcp = nullptr,strIpAddress = “”)
针对指定 IP 地址机械臂，在力控模式下，实时改变力指令的大小与方向，相比于 updateForce 增加了对工具坐标系支持

参数：

dblForceDirection：输入参数，力的方向，大小为三的数组,分别表示在 xyz 各方向上的分量

dblForceValue：输入参数，力的大小，单位 N

dblActiveTcp：可选参数，坐标系对应的位姿向量，大小为 6 的数组，不传则默认为基坐标系

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：

```
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};
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_ex(dblForceDirection,dblForceValue,nullptr,strIpAddress);
```

```
            M_SLEEP(1);
        }
        int intExitMode = 0;
        if (leaveForceMode(intExitMode.strIpAddress) < 0)
        {
            printf("Diana API leaveForceMode failed!\n");
        }
}
```

135 enablePassThrough

int

enablePassThrough(bEnable,uintStepCnt,uintMemCnt,uintGetDataPeriod,intSampleRate,strIpAddress="")

针对指定 IP 地址的机械臂，开启或关闭透传功能，即不经过控制器规划，使机械臂直接运动到目的点位

参数：

bEnable：输入参数，使能透传功能

uintStepCnt：输入参数，当经过指定数目的点后机械臂才开始移动

uintMemCnt：输入参数，暂存点位的缓存区的大小，需要大于等于 uintStepCount，透传功能才生效

uintGetDataPeriod：输入参数，设置控制器从缓存区取点的间隔，单位 ms

intSampleRate：输入参数，滤波频率，当该值小于 0 或大于 1000 则不开启滤波

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";
int ret = enablePassThrough(true,5,30,1,500,ipAddress);
if(ret < 0){
    printf(“enablePassThrough failed! Return value = %d\n”, ret);
```

```
}else{
    double joints[7] = {0,0,0,1.5707963267948966192313216916398,0,0,0};
    moveJ(joints,0.1,0.1);
    wait_move();
    int count = 0;
    while(count++<2000){
        joints[3] += 0.0017453292519943295769236907684 89;
        sendPassThroughJoints_rt(joints,ipAddress);
        M_SLEEP(1);
    }
    enablePassThrough(false,0,0,0,0,ipAddress);
}
```

136 sendPassThroughJoints_rt

int sendPassThroughJoints_rt(joints, strIpAddress="")
针对指定 IP 地址的机械臂，实时发送透传需要的关节角度 **参数：** joints：输入参数，想要机械臂运动到的关节角度 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： 见 enablePassThrough 示例

137 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，工业臂只能锁定七轴。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。当 active_tcp 不为空时，pose 描述工具中心点在基座标系下的位姿，否则 active_tcp 描述法兰中心点在基座标系下的位姿。

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 7 的数组。

active_tcp：工具坐标系对应的位姿向量，大小为 6 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：

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 = 7;
double lock_joint_position = 0;
double ref_joints[7] = {0.0};
int id = inverseClosedFull(pose, lock_joint_index, lock_joint_position, ref_joints, 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)", joints[0], joints[1], joints[2],
```

```
joints[3], joints[4], joints[5], joints[6]);
                }
            }
            destoryInverseClosedItems(id);
        }
}
```

138 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。

139 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。

140 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。

141 nullSpaceFreeDriving

int nullSpaceFreeDriving(enable,strIpAddress = “”)
zeroSpaceFreeDriving 的宏，用法参见 zeroSpaceFreeDriving。
调用示例： 见 zeroSpaceFreeDriving 示例。

142 nullSpaceManualMove

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

143 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);
}

144 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);
}

145 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);
}
```

146 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);
}
```

147 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。时间 t 为可选项，时间 t 是可选项，如果提供了 t 值，机器人将在 t 时间后减速。如果没有提供时间 t 值，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。现支持在不同工具坐标系下移动，由传入的 active_tcp 决定。

参数：

speed：工具空间速度，数组长度为 6,其中前 3 个单位为 m/s，后 3 个单位为 rad/s。

a：加速度，数组长度为 2，第 1 个单位：m/s^2，第 2 个单位为 rad/s^2。

t：时间，单位：s。

active_tcp：当前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 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);
}
```

148 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]);
}
```

149 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*7 的矩阵。

dblJointPosition：输入参数，用于计算雅各比矩阵的关节角，大小为 7 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 = 7;
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");
    }
}
```

150 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，现在支持在不同的工具坐标系下求解，由传入的 active_tcp 决定。

参数：

dblJacobiMatrix：输出参数，雅各比矩阵，大小为 6*7 的矩阵。

dblJointPosition：输入参数，用于计算雅各比矩阵的关节角，大小为 7 的数组。

intJointCount：输入参数，关节数量。

active_tcp：当前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：

```
const char *strIpAddress = "192.168.10.75";
const int intJointCount = 7;
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");
    }
}
```

附件 A：

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_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_JOINT_REGIST_ERROR	-2001	硬件错误
ERROR_CODE_COMMUNICATE_ERROR	-2101	底层通信失败（ln）
ERROR_CODE_CALLING_CONFLICT_ERROR	-2201	暂停状态执行操作失败
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_PLAN_ERROR	-2301	路径规划失败
ERROR_CODE_INTERPOLATE_POSITION_ERRO R	-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_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

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。