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，	孟庆婷	2020-9-9

	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。		
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、getJacobiMatrix 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： setWayPoint、getWayPoint、addWayPoint、	李漠	2021-04-25

	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

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 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();
memset(pinfo ->SrvIp, 0x00, sizeof(pinfo ->SrvIp));
memcpy(pinfo ->SrvIp, "127.0.0.1", strlen("127.0.0.1"));
pinfo->LocHeartbeatPort = 0;
pinfo->LocRobotStatePort = 0;
pinfo->LocSrvPort = 0;
int ret = initSrv(errorControl, logRobotState,  pinfo);
if(ret < 0)
{
   printf("initSrv failed! Return value = %d\n", ret);
}
```

2 destroySrv

int destroySrv()

结束调用 API，用于结束时释放资源。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：

无。

返回值：

0：成功。

-1：失败。

调用示例：

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

3 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();
```

4 formatError

const char *formatError(e)

获取错误码 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。

返回值：

错误描述信息。

调用示例：

```
int e = -1003;
const char* msg = formatError(e);
```

5 getLastError

int getLastError()

返回最近发生的错误码。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：

无。

返回值： 0：没有错误。 其它值：具体错误码。
调用示例： int e = getLastError();

6 setLastError

int setLastError(int e)
重置错误码。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。 **返回值：** 错误码。
调用示例： int e = setLastError(0);

7 setDefaultActiveTcp

int setDefaultActiveTcp(default_tcp)
设置默认的工具坐标系。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。 **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。 **返回值：** 0：成功。 -1：失败。

调用示例：
double
int ret = setDefaultActiveTcp(nullptr);
if(ret < 0)
{
printf(“setDefaultActiveTcp failed! Return value = %d\n”, ret);
}

8 moveTCP

int moveTCP(d, v, a, active_tcp)
手动移动末端中心点。该函数会立即返回，停止运动需要调用 stop 函数。 **参数：** d：表示移动方向的枚举类型。MOVE_X_UP 表示沿 x 轴正向；MOVE_X_DOWN 表示沿 x 轴负向；MOVE_Y_UP 表示沿 y 轴正向；MOVE_Y_DOWN 表示沿 y 轴负向；MOVE_Z_UP 表示沿 z 轴正向；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：当前工具坐标系名称，字符串类型。当前版本功能库暂不支持此功能，该函数调用时传入的 active_tcp 参数将被作为 nullptr 处理。后续升级后可用，为空时，将使用默认的工具坐标系 default_tcp。 **返回值：** 0：成功。 -1：失败。

调用示例：
tcp_direction_e dtype = MOVE_X_UP;
double vel = 0.1;
double acc = 0.2;
int ret = moveTCP(dtype, vel, acc, nullptr);
if(ret < 0)
{
printf(“moveTCP failed! Return value = %d\n”, ret);
}

9 rotationTCP

int rotationTCP(d, v, a, active_tcp)
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：** d：表示移动方向的枚举类型。MOVE_X_UP 表示沿 x 轴正向；MOVE_X_DOWN 表示沿 x 轴负向；MOVE_Y_UP 表示沿 y 轴正向；MOVE_Y_DOWN 表示沿 y 轴负向；MOVE_Z_UP 表示沿 z 轴正向；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：当前工具坐标系名称， 字符串类型。当前版本功能库暂不支持此功能，该函

数调用时传入的 active_tcp 参数将被作为 nullptr 处理。后续升级后可用，为空时，将使用默认的工具坐标系 default_tcp。

返回值：

0：成功。

-1：失败。

调用示例：

```
tcp_direction_e dtype = MOVE_X_UP;
double vel = 0.1;
double acc = 0.2;
int ret = rotationTCP(dtype, vel, acc, nullptr);
if(ret < 0)
{
   printf("rotationTCP failed! Return value = %d\n", ret);
}
```

10 moveJoint

moveJoint(d, i, v, a)

手动控制关节移动。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。MOVE_ UP 表示关节沿正向旋转；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。

返回值：

0：成功。

-1：失败。

调用示例：

```
joint_direction_e dtype = MOVE_UP;
int index = 1;
double vel = 0.8;
double acc = 0.8;
int ret = moveJoint(dtype, index, vel, acc);
if(ret < 0)
{
   printf("moveJoint failed! Return value = %d\n", ret);
```

```
}
```

11 moveJToTarget

int moveJToTarget(joints, v, a)

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。

返回值：

0：成功。

-1：失败。

调用示例：

```
void wait_move()
{
   M_SLEEP(20);
   while (true)
   {
      const char state = getRobotState();
      if (state != 0)
      {
         break;
      }
      else
      {
         M_SLEEP(1);
      }
   }
   stop();
}
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);
if(ret < 0)
{
   printf("moveJToTarget failed! Return value = %d\n", ret);
}
wait_move();
```

12 moveJToPose

int moveJToPose(pose, v, a, active_tcp)

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。

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：当前工具坐标系名称，字符串类型。当前版本功能库暂不支持此功能，该函数调用时传入的 active_tcp 参数将被作为 nullptr 处理。后续升级后可用，为空时，将使用默认的工具坐标系 default_tcp。

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 ret = moveJToPose(poses, vel, acc, nullptr);
if(ret < 0)
{
   printf("moveJToPose failed! Return value = %d\n", ret);
}
wait_move();
```

13 moveLToTarget

int moveLToTarget(joints, v, a)

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。

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;
int ret = moveLToTarget(joints, vel, acc);
if(ret < 0)
{
   printf("moveLToTarget failed! Return value = %d\n", ret);
}
wait_move();
```

14 moveLToPose

int moveLToPose(pose, v, a, active_tcp)

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。

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：目标工具坐标系相对末端位姿，double 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 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 ret = moveLToPose(poses, vel, acc, nullptr);
if(ret < 0)
{
   printf("moveLToPose failed! Return value = %d\n", ret);
}
wait_move();
```

15 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;
int ret = moveJ (joints, vel, acc);
if(ret < 0)
{
   printf("moveJ failed! Return value = %d\n", ret);
}
wait_move();
```

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

17 speedJ

int speedJ(speed, a, t)

速度模式，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。

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;
int ret = speedJ(speeds, acc, 0);
if(ret < 0)
{
   printf("speedJ failed! Return value = %d\n", ret);
}
```

18 speedL

int speedL(speed, a, t, active_tcp)

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：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。

active_tcp：当前工具坐标系名称，字符串类型。当前版本功能库暂不支持此功能，该函数调用时传入的 active_tcp 参数将被作为 nullptr 处理。后续升级后可用，为空时，将使用默认的工具坐标系 default_tcp。

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};
int ret = speedL(speeds, acc, 0, nullptr);
if(ret < 0)
{
   printf("speedL failed! Return value = %d\n", ret);
}
```

19 freeDriving

int freeDriving(enable)

实现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。

返回值：

0：成功。

-1：失败。

调用示例：

```
int ret = freeDriving(true);
if(ret < 0)
{
   printf("freeDriving failed! Return value = %d\n", ret);
}
```

20 stop

int stop()

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：

无。

返回值：

0：成功。

-1：失败。

调用示例：

```
int ret = stop();
if(ret < 0)
{
   printf("stop failed! Return value = %d\n", ret);
```

```
}
```

21 forward

int forward(joints, pose, active_tcp)

正解函数，通过关节角度计算出正解 TCP 位姿。

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：当前工具坐标系名称，字符串类型。当前版本功能库暂不支持此功能，该函数调用时传入的 active_tcp 参数将被作为 nullptr 处理。后续升级后可用，为空时，将使用默认的工具坐标系 default_tcp。

返回值：

0：成功。

-1：失败。

调用示例：

```
double joints[7] = {0.0};
double pose[6] = {0.0};
int ret = forward(joints, pose, nullptr);
if(ret < 0)
{
   printf("forward failed! Return value = %d\n", ret);
}
```

22 inverse

int inverse(pose, joints, active_tcp)

逆解函数，通过 TCP 位姿计算出最佳逆解关节角度。

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：当前工具坐标系名称，字符串类型。当前版本功能库暂不支持此功能，该函数调用时传入的 active_tcp 参数将被作为 nullptr 处理。后续升级后可用，为空时，将使用默认的工具坐标系 default_tcp。

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};
int ret = inverse(pose, joints, nullptr);
if(ret < 0)
{
   printf("inverse failed! Return value = %d\n", ret);
}
```

23 getJointPos

int getJointPos(joints)

获取关节角度位置，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。

返回值：

0：成功。

-1：失败。

调用示例：

```
double joints[7] = {0.0};
int ret = getJointPos(joints);
if(ret < 0)
{
   printf("getJointPos failed! Return value = %d\n", ret);
}
```

24 getJointAngularVel

int getJointAngularVel(vels)

获取当前各关节角速度。

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。

返回值：

0：成功。

-1：失败。

调用示例：

```
double speeds[7] = {0.0};
int ret = getJointAngularVel(speeds);
if(ret < 0)
{
   printf("getJointAngularVel failed! Return value = %d\n", ret);
}
```

25 getJointCurrent

int getJointCurrent(joints)
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。 **返回值：** 0：成功。 -1：失败。

调用示例：
double joints[7] = {0.0};
int ret = getJointCurrent(joints);
if(ret < 0)
{
printf("getJointCurrent failed! Return value = %d\n", ret);
}

26 getJointTorque

int getJointTorque(torques)
获取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。 **返回值：** 0：成功。 -1：失败。

调用示例：
double torques[7] = {0.0};
int ret = getJointTorque(torques);
if(ret < 0)
{
printf("getJointTorque failed! Return value = %d\n", ret);

```
}
```

27 getTcpPos

int getTcpPos(pose)

获取当前 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。

返回值：

0：成功。

-1：失败。

调用示例：

```
double poses[6] = {0.0};
int ret = getTcpPos(poses);
if(ret < 0)
{
   printf("getTcpPos failed! Return value = %d\n", ret);
}
```

28 getTcpExternalForce

double getTcpExternalForce()

获取末端实际感受到的力大小，末端可被 setDefaultActiveTcp 函数改变。

参数：

无。

返回值：

返回力的大小。

调用示例：

```
double force = getTcpExternalForce();
```

29 releaseBrake

int releaseBrake()

打开抱闸，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：

无。

返回值：

0：成功。

-1：失败。

调用示例：

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

30 holdBrake

int holdBrake()
关闭抱闸，停止机械臂。 **参数：** 无。 **返回值：** 0：成功。 -1：失败。

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

31 changeControlMode

int changeControlMode(m)
控制模式切换。 **参数：** m：枚举类型。MODE_INVALID 无意义；T_MODE_POSITION 位置模式；T_MODE_JOINT_IMPEDANCE 关节空间阻抗模式；T_MODE_CART_IMPEDANCE 笛卡尔空间阻抗模式。 **返回值：** 0：成功。 -1：失败。

调用示例：
int ret = changeControlMode(MODE_POSITION);
if(ret < 0)
{

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

32 getLinkState

int getLinkState()

获取链路状态。

参数：

无。

返回值：

0：链路正常。

-1：链路断开。

调用示例：

```
int ret = getLinkState();
if(ret == 0)
{
    printf("network connected \n");
}
else
{
    printf("network disconnected \n");
}
```

33 getTcpForce

int getTcpForce(forces)

获取 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。

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];
int ret = getTcpForce(tcpForce);
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");
}
```

34 getJointForce

int getJointForce(forces)

获取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。

返回值：

0：链路正常。

-1：链路断开。

调用示例：

```
double jointForce[7];
int ret = getJointForce (jointForce);
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");
}
```

35 isCollision

bool isCollision()

从轴空间判断获取机器人是否发生碰撞。

参数：

无。

返回值：

true：机器人发生碰撞。

false：机器人未发生碰撞。

调用示例：

```
if(isCollision())
```

```
{
    printf("Warning: Robot got collision\n");
}
else
{
    printf("Robot is running\n");
}
```

36 initDHCali

Int initDHCali(tcpMeas, jntPosMeas, nrSets)
根据输入的关节角以及末端位置数组计算 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。 **返回值：** 0：正常。 -1：失败。

调用示例：
double jntMeas[224] = {…};
double tcpMeas[96] = {…};
if (initDHCali(tcpMeas, jntMeas, rowNo_refData) != 0)
{
printf("DHCaliFcns.InitDHCali falid!\n");
}

37 getDHCaliResult

int getDHCaliResult(rDH, wRT, tRT, confid)
获取 DH 参数计算结果。 **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。

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[28], wRT[6], tRT[3];
if (getDHCaliResult(rDH, wRT, tRT) < 0)
{
   printf("DHCaliFcns.getDHCaliResult falid.\n");
}
```

38 setDH

int setDH(a, alpha, d, theta)

设置机器人当前 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。

返回值：

0：成功。

-1：失败。

调用示例：

```
double a[7], alpha[7], d[7], theta[7];
if (setDH(a, alpha, d, theta) < 0)
{
      printf("DHCaliFcns.setDH falid.\n");
}
```

39 setWrd2BasRT

int setWrd2BasRT(RTw2b)

初始化世界坐标系到机器人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。

返回值：

0：成功。

-1：失败。

调用示例：

```
double wRT[6] = {…}
if (setWrd2BasRT(wRT) < 0)
{
    printf("DHCaliFcns. setWrd2BasRT falid.\n");
}
```

40 setFLa2TcpRT

int setFla2TcpRT(RTf2t)

初始化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。

返回值：

0：成功。

-1：失败。

调用示例：

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

41 getRobotState

const char getRobotState()
获取机器人当前工作状态。 **参数：** 无。 **返回值：** 0：running。 1：paused。 2：idle。 3：free-driving。 4：zero-space-free-driving。

调用示例：
const char state = getRobotState();
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");
}

42 resume

int resume()
当机器人发生碰撞或其他原因暂停后，恢复运行时使用。 **参数：** 无。 **返回值：**

0：成功。 -1：失败。

调用示例：
if (resume() < 0)
{
printf("Diana API resume failed!\n");
}

43 setJointCollision

int setJointCollision(collision)
设置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。 **返回值：** 0：设置成功。 -1：设置失败。

调用示例：
double collision[7] = {200, 200, 200, 200, 200, 200, 200};
if (setJointCollision(collision) < 0)
{
printf("Diana API setJointCollision failed!\n");
}

44 setCartCollision

int setCartCollision(collision)
设置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。 **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};
if (setCartCollision (collision) < 0)
{
printf("Diana API setCartCollision failed!\n");

```
}
```

45 enterForceMode

int enterForceMode(intFrameType, dblFrameMatrix, dblForceDirection, dblForceValue, dblMaxApproachVelocity, dblMaxAllowTcpOffset)

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）。

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;
if (enterForceMode(iFrameType, dblFrameMatrix, dblForceDir, dblForceValue, dblMaxVel, dblMaxOffset) < 0)
{
   printf("Diana API enterForceMode failed!\n");
}
```

46 leaveForceMode

int leaveForceMode (intExitMode)

退出力控模式,并设置退出后 robot 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。 返回值： 0：成功。 -1：失败。

调用示例：
int intExitMode = 0;
if (leaveForceMode (intExitMode) < 0)
{
printf("Diana API leaveForceMode failed!\n");
}

47 setDefaultActiveTcpPose

int setDefaultActiveTcpPose (arrPose)
设置默认工具坐标系。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：** 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。 **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};
if (setDefaultActiveTcpPose (pose) < 0)
{
printf("Diana API setDefaultActiveTcpPose failed!\n");
}

48 setResultantCollision

int setResultantCollision (force)
设置笛卡尔空间碰撞检测 TCP 的合力矩阈值。 **参数：** force：合力值。 **返回值：** 0：成功。

-1：失败。

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

49 setJointImpedance

int setJointImpedance (arrStiff, arrDamp)
设置 Robot 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。 **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};
if (setJointImpedance (arrStiff, arrDamp) < 0)
{
printf("Diana API setJointImpedance failed!\n");
}

50 getJointImpedance

int getJointImpedance (arrStiff, arrDamp)
获取 Robot 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。 **返回值：** 0：成功。 -1：失败。

调用示例：

```
double arrStiff [7] = { 0};
double arrDamp [7] = { 0};
if (getJointImpedance (arrStiff, arrDamp) < 0)
{
   printf("Diana API getJointImpedance failed!\n");
}
```

51 setCartImpedance

int setCartImpedance (arrStiff, arrDamp)

设置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。

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};
if (setCartImpedance(arrStiff, arrDamp) < 0)
{
   printf("Diana API setCartImpedance failed!\n");
}
```

52 getCartImpedance

int getCartImpedance (arrStiff, arrDamp)

获取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。

返回值：

0：成功。

-1：失败。

调用示例：

```
double arrStiff [6] = {0};
double arrDamp [6] = {0};
if (getCartImpedance (arrStiff, arrDamp) < 0)
{
   printf("Diana API getCartImpedance failed!\n");
}
```

53 zeroSpaceFreeDriving

int zeroSpaceFreeDriving (enable)

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。

返回值：

0：成功。

-1：失败。

调用示例：

```
if (zeroSpaceFreeDriving (true) < 0)
{
   printf("Diana API zeroSpaceFreeDriving failed!\n");
}
else
{
   zeroSpaceFreeDriving(false);
}
```

54 createPath

int createPath (type, id_path)

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。

返回值：

0：成功。

-1：失败。
调用示例： 详见 addMoveJ 或 addMoveL 调用示例。

55 addMoveL

int addMoveL (id_path, joints, vel, acc, blendradius)
向已创建的路段添加 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。 **返回值：** 0：成功。 -1：失败。

调用示例：
unsigned int path_id = 0;
printf("start test moveJ path.\n");
createPath(1, path_id);
addMoveL(path_id, dblFirstMoveLPosition, 0.2, 0.2, 0.3);
addMoveL(path_id, dblSecondMoveLPosition, 0.2, 0.2, 0.3);
addMoveL(path_id, dblThirdMoveLPosition, 0.2, 0.2, 0.3);
runPath(path_id);
destroyPath(path_id);
wait_move();

56 addMoveJ

int addMoveJ (id_path, joints, vel_percent, acc_percent, blendradius_percent)
向已创建的路段添加 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。（拟改为 rad/s） acc_percent：moveJ 移动到目标路点的加速度百分比。（拟改为 rad/s^2）

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。 **返回值：** 0：成功。 -1：失败。

调用示例：
unsigned int path_id = 0;
printf("start test moveJ path.\n");
createPath(1, path_id);
addMoveJ(path_id, dblFirstPosition, 0.2, 0.2, 0.3);
addMoveJ(path_id, dblSecondPosition, 0.2, 0.2, 0.3);
addMoveJ(path_id, dblThirdPosition, 0.2, 0.2, 0.3);
runPath(path_id);
destroyPath(path_id);
wait_move();

57 runPath

int runPath (id_path)
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。 **返回值：** 0：成功。 -1：失败。
调用示例： 详见 addMoveJ 或 addMoveL 调用示例。

58 destroyPath

int destroyPath (id_path)
销毁路段。 **参数：** id_path：输入参数。要销毁的路径 ID。 **返回值：** 0：成功。 -1：失败。
调用示例： 详见 addMoveJ 或 addMoveL 调用示例。

59 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

60 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

61 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：

```
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
```

62 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：

```
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, 3.141593, 1.047198, 0.523599

63 servoJ

int servoJ (joints, time, look_ahead_time, gain)

关节空间内，伺服到指定关节角位置。 Servo 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）。

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};
for(int i = 0; i < 100; ++i)
{
  target1[6] = target1[6]+PI/180;
  ret = servoJ( target1, 0.01, 0.1, 300);
  if(ret < 0)
    break;
  sleep(0.001);
}
stop();
```

64 servoL

int servoL (pose, time, look_ahead_time, gain, scale)

笛卡尔空间内，伺服到指定位姿。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。

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。

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）。

返回值：

0：成功。

-1：失败。

调用示例：

65 speedJ_ex

int speedJ_ex (speed, a, t, realiable)

速度模式优化版，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。

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;
int ret = speedJ_ex(speeds, acc, 0, true);
if(ret < 0)
{
   printf("speedJ_ex failed! Return value = %d\n", ret);
}
```

66 speedL_ex

int speedL_ex(speed, a, t,,realiable, active_tcp)
速度模式优化版，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。 **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。 active_tcp：当前工具坐标系名称，字符串类型。当前版本功能库暂不支持此功能，该函数调用时传入的 active_tcp 参数将被作为 nullptr 处理。后续升级后可用，为空时，将使用默认的工具坐标系 default_tcp。 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。 **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};
int ret = speedL_ex(speeds, acc, 0, true, nullptr);
if(ret < 0)
{
printf("speedL_ex failed! Return value = %d\n", ret);
}

67 servoJ_ex

int servoJ_ex (joints, time, look_ahead_time, gain, realiable)
关节空间内，伺服到指定关节角位置优化版。 Servo 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。

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
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);
    next_time.tv_nsec += 1000000;//计算下次唤醒时间
    //sleep 到 next_time
}
stop();
```

68 servoL_ex

int servoL_ex (pose, time, look_ahead_time, gain, scale, realiable)

笛卡尔空间内，伺服到指定位姿优化版。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。

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。

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。

返回值：

0：成功。

-1：失败。

调用示例：

69 dumpToUDisk

int dumpToUDisk ()
导出日志文件到 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：** 无。 **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（） 4. 拔下 u 盘查看

70 saveEnvironment

int saveEnvironment ()
将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：** 无。 **返回值：** 0：成功。 -1：失败。

调用示例：
int ret = saveEnvironment ();
if(ret < 0)
{
printf(“Save failed!\n”);
}

71 readDI

int readDI(group_name, name, value)
读取一个数字输入的值。 **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。

返回值：

0：成功。

-1：失败。

调用示例：

```
int value;
int ret = readDI(“board”, “di0”, value);
if(ret < 0)
{
   printf(“readDI failed!\n”);
}
```

72 readAI

int readAI (group_name, name, mode, value)

读取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。

返回值：

0：成功。

-1：失败。

调用示例：

```
int mode;
double value;
int ret = readAI(“board”, “ai0”,mode,value);
if(ret < 0)
{
   printf(“ReadAI failed!\n”);
}
```

73 setAIMode

int setAIMode (group_name, name, mode)

设置模拟输入的模式。

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。

返回值：

0：成功。

-1：失败。

调用示例：

```
int mode = 1;
int ret = setAIMode(“board”, “ai0”,mode);
if(ret < 0)
{
   printf(“SetAIMode failed!\n”);
}
```

74 writeDO

int writeDO (group_name, name,value)

设置一个数字输出的值。

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。

返回值：

0：成功。

-1：失败。

调用示例：

```
Int value = 0;
int ret = writeDO (“board”, “do0”,value);
if(ret < 0)
{
   printf(“WriteDO failed!\n”);
}
```

75 writeAO

int writeAO (group_name, name, mode, value)

设置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。

返回值：

0：成功。

-1：失败。

调用示例：

```
int mode = 1;
double value = 8.8;
int ret = writeAO (“board”, “ao0”, mode, value);
if(ret < 0)
{
   printf(“WriteAO failed!\n”);
}
```

76 readBusCurrent

int readBusCurrent(current)

读取总线电流。

参数：

current：总线电流。

返回值：

0：成功。

-1：失败。

调用示例：

```
double current;
int ret = readBusCurrent(current);
if(ret < 0)
{
   printf(“ReadBusCurrent failed!\n”);
}
```

77 readBusVoltage

int readBusVoltage (voltage)
读取总线电压。 **参数：** voltage：总线电压。 **返回值：** 0：成功。 -1：失败。

调用示例：
double voltage;
int ret = readBusVoltage(voltage);
if(ret < 0)
{
printf(“ReadBusVoltage failed!\n”);
}

78 getDH

int getDH(aDH, alphaDH,dDH,thetaDH)
获取 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 **返回值：** 0：成功。 -1：失败。

调用示例：
double a[7],alpha[7],d[7],theta[7];
int ret = getDH(a,alpha,d,theta);
if(ret < 0)
{
printf(“getDH failed!\n”);
}

79 getOriginalJointTorque

int getOriginalJointTorque(torques)
获取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 **返回值：** 0：成功。 -1：失败。

调用示例：
double torques[7];
int ret = getOriginalJointTorque(torques);
if(ret < 0)
{
printf(“getOriginalJointTorque failed!\n”);
}

80 getJacobiMatrix

int getJacobiMatrix(matrix_jacobi)
获取雅各比矩阵。 **参数：** matrix_jacobi：雅各比矩阵的数组，长度为 42 **返回值：** 0：成功。 -1：失败。

调用示例：
double matrix_jacobi [42];
int ret = getJacobiMatrix (matrix_jacobi);
if(ret < 0)
{
printf(“getJacobiMatrix failed!\n”);
}

81 resetDH

int resetDH()
重置用户自定义 DH 参数。 **参数：** 无 **返回值：** 0：成功。 -1：失败。

调用示例：
int ret = resetDH();
if(ret < 0)
{
printf(“resetDH failed!\n”);
}

82 runProgram

int runProgram(programName)
运行程序。 **参数：** programName：程序的名字。 **返回值：** 0：成功。 -1：失败。

调用示例：
int ret=runProgram(“AgileRobots”);
if(ret < 0)
{
printf(“ runProgram failed!\n”);
}

83 stopProgram

int stopProhram(programName)
停止指定程序。 **参数：** programName：程序的名字。 **返回值：** 0：成功。 -1：失败。

调用示例：
int ret=stopProgram(“AgileRobots”);
if(ret < 0)
{
printf(“ runProgram failed!\n”);
}

84 getVariableValue

int getVariableValue(variableName,value)
获取全局变量的值。 **参数：** variableName：全局变量的名字 value: 获取的全局变量的值 **返回值：** 0：成功。

-1：失败。

调用示例：
double value;
int ret = getVariableValue (“GLOBAL”,value);
if(ret < 0)
{
printf(“getVariableValue failed!\n”);
}

85 setVariableValue

int setVariableValue(variableName,value)
设置全局变量的值。 **参数：** variableName：全局变量的名字 value: 设置的全局变量的值 **返回值：** 0：成功。 -1：失败。

调用示例：
double value=2.0;
int ret = setVariableValue (“GLOBAL”,value);
if(ret < 0)
{
printf(“setVariableValue failed!\n”);
}

86 isTaskRunning

int isTaskRunning(programName)
判断指定程序是否在运行。 **参数：** programName：程序名称 **返回值：** 0：运行中。 -1：没有运行中。

调用示例：
int ret = isTaskRunning (“AgileRobots”);
if(ret < 0)
{
printf(“AgileRobots is not running!\n”);
}

87 pauseProgram

int pauseProgram()
暂停所有程序。 **参数：** **返回值：** 0：成功。 -1：失败。

调用示例：
int ret = pauseProgram();
if(ret == 0)
{
printf(“All program is paused!\n”);
}

88 resumeProgram

int resumeProgram()
恢复运行已经暂停的程序。 **参数：** **返回值：** 0：成功。 -1：失败。

调用示例：
int ret = resumeProgram();
if(ret == 0)
{
printf(“All program is resume!\n”);
}

89 stopAllProgram

int stopAllProgram()
停止所有程序。 **参数：** **返回值：** 0：成功。 -1：失败。

调用示例：
int ret = stopAllProgram();
if(ret == 0)
{
printf(“All program is stop!\n”);
}

90 isAnyTaskRunning

int isAnyTaskRunning()
是否有程序在运行。 **参数：** **返回值：** 0：成功。 -1：失败。

调用示例：
int ret = isAnyTaskRunning();
if(ret < 0)
{
printf(“Any program is not running!\n”);
}

91 cleanErrorInfo

int cleanErrorInfo()
清除错误信息。 **参数：** **返回值：** 0：成功。 -1：失败。

调用示例：
int ret = cleanErrorInfo();
if(ret < 0)
{
printf(“cleanErrorInfo failed!\n”);
}

92 setCollisionLevel

int setCollisionLevel(level)
设置碰撞检测类型 **参数：** level：碰撞等级，int 类型 **返回值：** 0：成功。 -1：失败。

调用示例：
int ret = setCollisionLevel(3);
if(ret < 0)
{
printf(“setCollision failed!\n”);
}

93 mappingInt8Variant

int mappingInt8Variant(variableName,index)
映射 int8 型变量。 **参数：** variableName：变量名称 index: 映射变量序号 **返回值：** 0：成功。 -1：失败。
调用示例： 参考 setMappingAddress 示例。

94 mappingDoubleVariant

int mappingDoubleVariant(variableName,index)
映射 double 型变量。 **参数：** variableName：变量名称 index: 映射变量序号 **返回值：** 0：成功。 -1：失败。
调用示例： 参考 setMappingAddress 示例。

95 mappingInt8IO

int mappingInt8IO(groupName,name,index)
映射数字信号 IO。 **参数：** groupName：IO 组名 name：IO 名字 index: 映射序号 **返回值：** 0：成功。 -1：失败。
调用示例：

参考 setMappingAddress 示例。

96 mappingDoubleIO

int mappingDoubleIO(groupName,name,index)

映射模拟信号 IO。

参数：

groupName：IO 组名

name: IO 名称

index: 映射序号

返回值：

0：成功。

-1：失败。

调用示例：

参考 setMappingAddress 示例。

97 setMappingAddress

int setMappingAddress(dblAddress,doubleItemCount,int8Address,int8ItemCount)

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

返回值：

0：成功。

-1：失败。

调用示例：

```
double value;
int ret = getVariableValue (“GLOBAL”,value);
if(ret < 0)
{
   printf(“getVariableValue failed!\n”);
}
```

98 lockMappingAddress

int lockMappingAddress()

访问数据时加锁。

参数：

返回值： 0：成功。 -1：失败。
调用示例： 参考 setMappingAddress 示例。

99 unlockMappingAddress

int unlockMappingAddress()
解锁数据锁。 **参数：** **返回值：** 0：成功。 -1：失败。
调用示例： 参考 setMappingAddress 示例。

100 setWayPoint

int setWayPoint(strWayPointName,dblTcppos,dblJoints)
修改路点变量。 **参数：** strWayPointName：路点变量名称 。 dblTcppos：位姿信息。 dblJoints：关节角信息。 **返回值：** 0：成功。 -1：失败。
调用示例： string strWayPointName = “setName”; 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);

101 getWayPoint

int getWayPoint(strWayPointName,dblTcppos,dblJoints)
获取路点变量信息。 **参数：** strWayPointName：路点变量名称 。 dblTcppos：位姿信息。 dblJoints：关节角信息。 **返回值：** 0：成功。 -1：失败。
调用示例： 参考 setWayPoint 示例。

102 addWayPoint

int addWayPoint(strWayPointName,dblTcppos,dblJoints)
新增路点变量。 **参数：** strWayPointName：路点变量名称 。 dblTcppos：位姿信息。 dblJoints：关节角信息。

返回值： 0：成功。 -1：失败。
调用示例： 参考 setWayPoint 示例。

103 deleteWayPoint

int deleteWayPoint(strWayPointName)
删除路点变量。 **参数：** strWayPointName：路点变量名称。 **返回值：** 0：成功。 -1：失败。
调用示例： String strWayPointName = “deleteName”; deleteWayPoint(deleteName.c_str());

104 createComplexPath

int createComplexPath (complex_path_type, complex_path_id)
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 7 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 7 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。 **返回值：** 0：成功。

-1：失败。
调用示例： 详见 addMoveJByTarget 或 addMoveLByTarget 调用示例。

105 addMoveLByTarget

int addMoveLByTarget (complex_path_id, target_joints, vel, acc, blendradius)
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。 **返回值：** 0：成功。 -1：失败。

调用示例：
double dblFirstPosition [7] = {……};
double dblSecondPosition [7] = {……};
unsigned int uintComplexPathId = 0;
createComplexPath(NORMAL_JOINT_PATH, uintComplexPathId);
addMoveLByTarget(uintComplexPathId, dblFirstPosition, 0.2, 0.4, 0);
addMoveLByTarget(uintComplexPathId, dblSecondPosition, 0.2, 0.2, 0);
runComplexPath(uintComplexPathId);
destroyComplexPath(uintComplexPathId);
wait_move();

106 addMoveLByPose

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

返回值：

0：成功。

-1：失败。

调用示例：

```
double dblFirstPose [6] = {……};
double dblSecondPose [6] = {……};
unsigned int uintComplexPathId = 0;
createComplexPath(NORMAL_POSE_PATH, uintComplexPathId);
addMoveLByPose(uintComplexPathId, dblFirstPose, 0.2, 0.4, 0);
addMoveLByPose(uintComplexPathId, dblSecondPose, 0.2, 0.2, 0);
runComplexPath(uintComplexPathId);
destroyComplexPath(uintComplexPathId);
wait_move();
```

107 addMoveJByTarget

int addMoveJByTarget (complex_path_id, target_joints, vel_percent, acc_percent, blendradius_percent)

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，相对于系统软限位中设定最大交融半径的百分比。

返回值：

0：成功。

-1：失败。

调用示例：

```
double dblFirstPosition [7] = {……};
```

```
double dblSecondPosition [7] = {……};
unsigned int uintComplexPathId = 0;
createComplexPath(NORMAL_JOINT_PATH, uintComplexPathId);
addMoveJByTarget (uintComplexPathId, dblFirstPosition, 0.2, 0.4, 0);
addMoveJByTarget(uintComplexPathId, dblSecondPosition, 0.2, 0.2, 0);
runComplexPath(uintComplexPathId);
destroyComplexPath(uintComplexPathId);
wait_move();
```

108 addMoveJByPose

int addMoveJByPose (complex_path_id, target_pose, vel_percent, acc_percent, blendradius_percent)

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。相对于系统软限位中设定最大交融半径的百分比。

返回值：

0：成功。

-1：失败。

调用示例：

```
double dblFirstPose [6] = {……};
double dblSecondPose [6] = {……};
unsigned int uintComplexPathId = 0;
createComplexPath(NORMAL_POSE_PATH, uintComplexPathId);
addMoveJByPose(uintComplexPathId, dblFirstPose, 0.2, 0.4, 0);
addMoveJByPose(uintComplexPathId, dblSecondPose, 0.2, 0.2, 0);
runComplexPath(uintComplexPathId);
destroyComplexPath(uintComplexPathId);
wait_move();
```

109 addMoveCByTarget

int addMoveCByTarget (complex_path_id, pass_joints, target_joints, vel, acc, blendradius, ignore_rotation)
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 **返回值：** 0：成功。 -1：失败。

调用示例：
double dblFirstPosition [7] = {……};
double dblSecondPosition [7] = {……};
unsigned int uintComplexPathId = 0;
createComplexPath(NORMAL_JOINT_PATH, uintComplexPathId);
addMoveCByTarget (uintComplexPathId, dblFirstPosition, dblSecondPosition, 0.2, 0.4, 0,true);
runComplexPath(uintComplexPathId);
destroyComplexPath(uintComplexPathId);
wait_move();

110 addMoveCByPose

int addMoveCByPose (complex_path_id, pass_pose, target_pose, vel, acc, blendradius, ignore_rotation)
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/s2。

blendradius：交融半径。单位：m。

ignore_rotation：是否忽略姿态变化

返回值：

0：成功。

-1：失败。

调用示例：

```
double dblFirstPose [6] = {……};
double dblSecondPose [6] = {……};
unsigned int uintComplexPathId = 0;
createComplexPath(NORMAL_POSE_PATH, uintComplexPathId);
addMoveCByPose(uintComplexPathId, dblFirstPose, dblSecondPose, 0.2, 0.4, 0,true);
runComplexPath(uintComplexPathId);
destroyComplexPath(uintComplexPathId);
wait_move();
```

111 runComplexPath

int runComplexPath (complex_path_id)

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。

返回值：

0：成功。

-1：失败。

调用示例：

详见 addMoveJByTarget 或 addMoveLByTarget 调用示例。

112 destroyComplexPath

int destroyComplexPath (complex_id_path)

销毁路段。

参数：

complex_id_path：输入参数。要销毁的路径 ID。

返回值：

0：成功。 -1：失败。
调用示例： 详见 addMoveJBytarget 或 addMoveLByTarget 调用示例。

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_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_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_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。