



C# 函数使用手册

Version 1.0.1

遨博（北京）智能科技有限公司

使用手册会定期进行检查和修正，更新后的内容将出现在新版本中。本手册中的内容或信息如有变更，恕不另行通知。

对本手册中可能出现的任何错误或遗漏，或因使用本手册及其中所述产品而引起的意外或间接伤害，遨博（北京）智能科技有限公司概不负责。

安装、使用产品前，请阅读本手册。

请保管好本手册，以便可以随时阅读和参考。

本手册为遨博（北京）智能科技有限公司专有财产，非经遨博（北京）智能科技有限公司书面许可，不得复印、全部或部分复制或转变为任何其他形式使用。

Copyright © 2015-2022 AUBO 保留所有权利。

目录

| | |
|--|----|
| 目录 | 3 |
| 1 数据类型..... | 8 |
| 1.1 机械臂运动..... | 8 |
| 1.1.1 Pos 路点的位置信息 | 8 |
| 1.1.2 cartesianPos_U 路点的位置信息 | 8 |
| 1.1.3 Ori 姿态的四元素表示方法..... | 8 |
| 1.1.4 Rpy 姿态的欧拉角表示方法 | 9 |
| 1.1.5 wayPoint_S 机械臂的路点信息..... | 9 |
| 1.1.6 JointVelcAccParam 机械臂关节速度加速度信息..... | 9 |
| 1.1.7 JointRadian 机械臂关节角度..... | 9 |
| 1.1.8 MoveRelative 描述运动属性中的偏移属性 | 10 |
| 1.2 坐标系..... | 10 |
| 1.2.1 CoordCalibrate 坐标系结构体 | 10 |
| 1.2.2 MoveRotateAxis 转轴定义..... | 11 |
| 1.3 工具..... | 12 |
| 1.3.1 ToolInEndDesc 机械臂工具端参数..... | 12 |
| 1.3.2 ToolCalibrate 工具标定结构体 | 12 |
| 1.3.3 ToolInertia 该结构体描述工具惯量 | 13 |
| 1.3.4 ToolDynamicsParam 动力学参数 | 13 |
| 1.4 机械臂相关信息..... | 13 |
| 1.4.1 RobotEventInfo 机械臂事件 | 13 |
| 1.4.2 RobotDiagnosis 机械臂诊断信息 | 14 |
| 1.4.3 JointStatus 关节状态信息 | 15 |
| 1.4.4 JointVersion 关节版本信息 | 15 |
| 1.4.5 JointProductID 机械臂 ID 信息 | 15 |
| 1.4.6 RobotDevInfo 设备信息..... | 15 |
| 2 接口函数..... | 17 |
| 2.1 机械臂系统接口..... | 17 |
| 2.1.1 rs_login 机械臂登录..... | 17 |
| 2.1.2 rs_logout 退出登录..... | 17 |
| 2.1.3 rs_robot_startup 启动机械臂..... | 17 |
| 2.1.4 rs_robot_shutdown 关机..... | 18 |
| 2.1.5 rs_initialize 初始化机械臂控制库 | 18 |
| 2.1.6 rs_uninitialize 反初始化机械臂控制 | 18 |
| 2.1.7 rs_create_context 创建机械臂控制上下文句柄..... | 18 |
| 2.1.8 rs_destory_context 注销机械臂控制上下文句柄..... | 19 |
| 2.2 状态推送..... | 19 |
| 2.2.1 rs_enable_push_realtime_roadpoint 设置是否允许实时路点信息推送 19 | |
| 2.2.2 rs_setcallback_realtime_roadpoint 获取实时路点信息的回调函数 | 19 |
| 2.2.3 REALTIME_ROADPOINT_CALLBACK 实时路点的回调函数类型 | |

| | |
|--------|--|
| | 20 |
| 2.2.4 | CurrentPositionCallback 路点回调20 |
| 2.2.5 | PrintWaypoint 打印路点信息.....20 |
| 2.2.6 | rs_setcallback_realtime_end_speed 获取实时末端速度信息.....20 |
| 2.2.7 | REALTIME_ENDSPEED_CALLBACK 实时末端速度的回调函数类型 21 |
| 2.2.8 | CurrentEndSpeedCallback 速度回调21 |
| 2.2.9 | rs_setcallback_robot_event 获取实时事件信息的回调函数21 |
| 2.2.10 | ROBOT_EVENT_CALLBACK 机械臂事件的回调函数类型21 |
| 2.2.11 | RobotEventCallback 事件信息回调.....22 |
| 2.2.12 | rs_setcallback_realtime_joint_status 获取实时关节状态.....22 |
| 2.2.13 | ROBOT_JOINT_STATUS_CALLBACK 机械臂关节状态的回调函数 类型 22 |
| 2.2.14 | CurrentJointStatusCallback 关节状态回调23 |
| 2.3 | 机械臂运动相关的接口23 |
| 2.3.1 | rs_init_global_move_profile 初始化全局的运动属性23 |
| 2.3.2 | rs_set_global_joint_maxacc 设置关节型运动的最大加速度23 |
| 2.3.3 | rs_set_global_joint_maxvelc 设置关节型运动的最大速度23 |
| 2.3.4 | rs_get_global_joint_maxacc 获取关节型运动的最大加速度.....24 |
| 2.3.5 | rs_get_global_joint_maxvelc 获取关节型运动的最大速度.....24 |
| 2.3.6 | rs_move_joint 关节运动.....24 |
| 2.3.7 | rs_move_joint_to 保持当前位姿通过关节运动的方式运动到目标位置 24 |
| 2.3.8 | rs_set_no_arrival_ahead 取消提前到位设置.....25 |
| 2.3.9 | rs_set_arrival_ahead_distance 设置提前到位距离模式.....25 |
| 2.3.10 | rs_set_arrival_ahead_time 设置提前到位时间模式.....25 |
| 2.3.11 | rs_set_global_end_max_line_acc 设置末端型运动的最大线加速度..25 |
| 2.3.12 | rs_set_global_end_max_line_velc 设置末端型运动的最大线速度....26 |
| 2.3.13 | rs_get_global_end_max_line_acc 获取末端型运动的最大线加速度 .26 |
| 2.3.14 | rs_get_global_end_max_line_velc 获取末端型运动的最大线速度26 |
| 2.3.15 | rs_set_global_end_max_angle_acc 设置末端型运动的最大角加速度26 |
| 2.3.16 | rs_set_global_end_max_angle_velc 设置末端型运动的最大角速度..27 |
| 2.3.17 | rs_get_global_end_max_angle_acc 获取末端型运动的最大角加速度 27 |
| 2.3.18 | rs_get_global_end_max_angle_velc 获取末端型运动的最大角速度..27 |
| 2.3.19 | rs_move_line 直线运动27 |
| 2.3.20 | rs_move_line_to 保持当前位姿通过直线运动的方式运动到目标位置 28 |
| 2.3.21 | rs_move_rotate 保持当前位置变换姿态做旋转运动28 |
| 2.3.22 | rs_remove_all_waypoint 清除路点容器28 |
| 2.3.23 | rs_add_waypoint 添加路点28 |
| 2.3.24 | rs_set_blend_radius 设置交融半径.....29 |
| 2.3.25 | rs_set_circular_loop_times 设置圆轨迹时圆的圈数.....29 |
| 2.3.26 | rs_move_track 轨迹运动29 |

| | | | |
|--------|-------------------------------------|------------------------|----|
| 2.3.27 | rs_set_relative_offset_on_base | 设置基于基坐标系运动偏移量 | 29 |
| 2.3.28 | rs_set_relative_offset_on_user | 设置基于用户标系运动偏移量 | 30 |
| 2.4 | 运动学相关的接口 | | 30 |
| 2.4.1 | rs_forward_kin | 正解 | 30 |
| 2.4.2 | rs_inverse_kin | 逆解 | 30 |
| 2.4.3 | rs_set_base_coord | 设置基坐标系 | 31 |
| 2.4.4 | rs_set_user_coord | 设置用户坐标系 | 31 |
| 2.4.5 | rs_check_user_coord | 检查用户坐标系参数设置是否合理 | 31 |
| 2.4.6 | rs_base_to_user | 基坐标系转用户坐标系 | 31 |
| 2.4.7 | rs_base_to_base_additional_tool | 基坐标系转基坐标得到工具末端点的位置和姿态 | 32 |
| 2.4.8 | rs_user_to_base | 用户坐标系转基坐标系 | 32 |
| 2.4.9 | rs_rpy_to_quaternion | 欧拉角转四元数 | 33 |
| 2.4.10 | rs_quaternion_to_rpy | 四元数转欧拉角 | 33 |
| 2.5 | 机械臂控制接口 | | 33 |
| 2.5.1 | rs_move_stop | 机械臂运动停止 | 33 |
| 2.5.2 | rs_move_fast_stop | 快速停止机械臂运动 | 34 |
| 2.5.3 | rs_move_pause | 暂停机械臂运动 | 34 |
| 2.5.4 | rs_move_continue | 暂停后恢复机械臂运动 | 34 |
| 2.5.5 | rs_collision_recover | 碰撞后恢复 | 34 |
| 2.5.6 | rs_get_robot_state | 获取机械臂当前运行状态 | 35 |
| 2.6 | 末端工具接口 | | 35 |
| 2.6.1 | rs_set_none_tool_dynamics_param | 设置无工具的动力学参数 | 35 |
| 2.6.2 | rs_set_tool_dynamics_param | 设置工具的动力学参数 | 35 |
| 2.6.3 | rs_get_tool_dynamics_param | 获取工具的动力学参数 | 35 |
| 2.6.4 | rs_set_tool_end_param | 设置末端工具的运动学参数 | 36 |
| 2.6.5 | rs_set_none_tool_kinematics_param | 设置无工具的运动学参数 | 36 |
| 2.6.6 | rs_set_tool_kinematics_param | 设置工具的运动学参数 | 36 |
| 2.6.7 | rs_get_tool_kinematics_param | 获取工具的运动学参数 | 36 |
| 2.7 | 设置和获取机械臂相关参数接口 | | 37 |
| 2.7.1 | rs_set_work_mode | 设置当前机械臂模式：仿真或真实 | 37 |
| 2.7.2 | rs_get_work_mode | 获取当前机械臂模式：仿真或真实 | 37 |
| 2.7.3 | rs_set_collision_class | 设置碰撞等级 | 37 |
| 2.7.4 | rs_get_collision_class | 获取当前碰撞等级 | 37 |
| 2.7.5 | rs_get_joint_status | 获取机械臂关节状态 | 38 |
| 2.7.6 | rs_get_current_waypoint | 获取机械臂当前路点信息 | 38 |
| 2.7.7 | rs_get_socket_status | 获取 socket 链接状态 | 38 |
| 2.7.8 | rs_get_diagnosis_info | 获取机械臂诊断信息 | 38 |
| 2.7.9 | rs_get_device_info | 获取机械臂设备信息 | 39 |
| 2.7.10 | rs_get_error_information_by_errcode | 根据错误号返回错误信息 | 39 |
| 2.8 | 接口板 IO 相关的接口 | | 39 |
| 2.8.1 | rs_set_board_io_status_by_addr | 根据接口板 IO 类型和地址设置 IO 状态 | 39 |
| 2.8.2 | rs_get_board_io_status_by_addr | 根据接口板 IO 类型和地址获取 IO 状 | |

| | |
|---|----|
| 态 | 40 |
| 2.9 工具 IO 相关的接口 | 40 |
| 2.9.1 rs_set_tool_power_type 设置工具端电源电压类型 | 40 |
| 2.9.2 rs_get_tool_power_type 获取工具端电源电压类型 | 40 |
| 2.9.3 rs_set_tool_io_type 设置工具端数字量 IO 的类型：输入或者输出 | 41 |
| 2.9.4 rs_get_tool_io_status 获取工具端 IO 的状态 | 41 |
| 2.9.5 rs_set_tool_do_status 设置工具端 IO 的状态 | 41 |
| 3 错误码 | 42 |
| 3.1 接口函数错误码定义 | 42 |
| 3.2 由于控制器异常事件导致的错误码 | 43 |
| 3.3 由于硬件层异常事件导致的错误码 | 44 |
| 4 接口函数示例 | 48 |
| 4.1 使用 SDK 构建一个最简单的机械臂的控制工程 | 48 |
| 4.2 用回调函数的方式来获取实时信息 | 50 |
| 4.2.1 获取实时路点信息 | 50 |
| 4.2.2 获取实时末端速度信息 | 50 |
| 4.2.3 获取机械臂的事件信息 | 51 |
| 4.2.4 获取关节状态信息 | 51 |
| 4.3 正逆解 | 52 |
| 4.3.1 正解 | 52 |
| 4.3.2 逆解 | 52 |
| 4.4 坐标系转换 | 53 |
| 4.4.1 基坐标系转用户坐标系 rs_base_to_user() | 53 |
| rs_base_to_user 函数示例 1 | 53 |
| rs_base_to_user 函数示例 2 | 56 |
| rs_base_to_user 函数示例 3 | 59 |
| 4.4.2 基坐标系转基坐标系 rs_base_to_base_additional_tool() | 60 |
| rs_base_to_base_additional_tool 函数示例 | 60 |
| 4.4.3 用户坐标系转基坐标系 rs_user_to_base() | 62 |
| rs_user_to_base 函数示例 1 | 62 |
| rs_user_to_base 函数示例 2 | 64 |
| rs_user_to_base 函数示例 3 | 67 |
| 4.5 设置和获取机械臂相关参数 | 68 |
| 4.5.1 获取当前路点信息 | 68 |
| 4.5.2 获取关节状态 | 69 |
| 4.5.3 获取诊断信息 | 70 |
| 4.5.4 获取设备信息 | 72 |
| 4.5.5 设置和获取工作模式：仿真或真实 | 74 |
| 4.5.6 设置和获取碰撞等级 | 74 |
| 4.5.7 根据错误码获取错误信息 | 75 |
| 4.5.8 获取 socket 的连接状态 | 75 |
| 4.6 IO | 76 |
| 4.6.1 工具 IO | 76 |
| 4.6.2 用户 IO | 80 |

| | | |
|--------|--|-----|
| 4.7 | 关节运动..... | 82 |
| 4.7.1 | rs_move_joint 函数..... | 82 |
| 4.7.2 | rs_move_joint_to 函数..... | 84 |
| | rs_move_joint_to 函数示例 1..... | 84 |
| | rs_move_joint_to 函数示例 2..... | 85 |
| | rs_move_joint_to 函数示例 3..... | 86 |
| | rs_move_joint_to 函数示例 4..... | 89 |
| 4.8 | 跟随模式..... | 91 |
| 4.8.1 | 跟随模式之提前到位..... | 91 |
| 4.9 | 直线运动..... | 93 |
| 4.9.1 | rs_move_line 函数..... | 93 |
| 4.9.2 | rs_move_line_to 函数..... | 96 |
| | rs_move_line_to 函数示例 1..... | 96 |
| | rs_move_line_to 函数示例 2..... | 98 |
| | rs_move_line_to 函数示例 3..... | 100 |
| | rs_move_line_to 函数示例 4..... | 103 |
| 4.10 | 偏移运动..... | 106 |
| 4.10.1 | rs_set_relative_offset_on_base 函数..... | 106 |
| | 示例 1: 法兰盘中心在基坐标系下..... | 106 |
| | 示例 2: 工具末端在基坐标系下..... | 108 |
| 4.10.2 | rs_set_relative_offset_on_user 函数..... | 110 |
| | 示例 1: 法兰盘中心在用户坐标系下..... | 110 |
| | 示例 2: 工具末端在用户坐标系下..... | 113 |
| | 示例 3: 工具末端在工具坐标系下..... | 116 |
| 4.11 | 旋转运动..... | 119 |
| 4.11.1 | rs_move_rotate 函数..... | 119 |
| | 示例 1: 法兰盘中心在基坐标系下..... | 119 |
| | 示例 2: 末端工具在基坐标系下旋转..... | 121 |
| | 示例 3: 法兰盘中心在用户坐标系下旋转..... | 123 |
| | 示例 4: 末端工具在用户坐标系下旋转..... | 126 |
| | 示例 5: 在工具坐标系下旋转..... | 129 |
| 4.12 | 轨迹运动..... | 131 |
| 4.12.1 | rs_move_track 函数..... | 131 |
| | 示例 1: 圆运动..... | 131 |
| | 示例 2: 圆弧运动..... | 133 |
| | 示例 3: MOVEP..... | 136 |
| 5 | 环境配置说明..... | 139 |
| 5.1.1 | 新建工程..... | 139 |
| 5.1.2 | 配置 DLL..... | 143 |
| 5.1.3 | 编写并运行工程..... | 143 |

1 数据类型

1.1 机械臂运动

1.1.1 Pos 路点的位置信息

```
[StructLayout(LayoutKind.Sequential)]
public struct Pos
{
    public double x;
    public double y;
    public double z;
}
```

1.1.2 cartesianPos_U 路点的位置信息

```
[StructLayout(LayoutKind.Sequential)]
public struct cartesianPos_U
{
    // 指定数组尺寸
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
    public double[] positionVector;
};
```

1.1.3 Ori 姿态的四元素表示方法

```
[StructLayout(LayoutKind.Sequential)]
public struct Ori
{
    public double w;
    public double x;
    public double y;
    public double z;
};
```


1.1.4 Rpy 姿态的欧拉角表示方法

```
[StructLayout(LayoutKind.Sequential)]
public struct Rpy
{
    public double rx;
    public double ry;
    public double rz;
};
```

1.1.5 wayPoint_S 机械臂的路点信息

```
[StructLayout(LayoutKind.Sequential)]
public struct wayPoint_S
{
    //机械臂的位置信息
    public Pos cartPos;
    //机械臂的姿态信息
    public Ori orientation;
    //机械臂关节角信息
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = ARM_DOF)]
    public double[] jointpos;
};
```

1.1.6 JointVelcAccParam 机械臂关节速度加速度信息

```
[StructLayout(LayoutKind.Sequential)]
public struct JointVelcAccParam
{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = ARM_DOF)]
    public double[] jointPara;
};
```

1.1.7 JointRadian 机械臂关节角度

```
[StructLayout(LayoutKind.Sequential)]
public struct JointRadian
{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = ARM_DOF)]
    public double[] jointRadian;
};
```

1.1.8 MoveRelative 描述运动属性中的偏移属性

```
[StructLayout(LayoutKind.Sequential)]
public struct MoveRelative
{
    //是否使能偏移
    public byte enable;
    //偏移量 x,y,z
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
    public float[] pos;
    //相对姿态偏移量
    public Ori orientation;
};
```

1.2 坐标系

1.2.1 CoordCalibrate 坐标系结构体

```
[StructLayout(LayoutKind.Sequential)]
public struct CoordCalibrate
{
    //坐标系类型：当 coordType==BaseCoordinate 或者 coordType==EndCoordinate 时，下面 3 个参数不做处理
    public int coordType;
    //坐标系标定方法
    public int methods;
    //用于标定坐标系的 3 个点（关节角），对应于机械臂法兰盘中心点基于基坐标系
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
    public JointRadian[] jointPara;
    //标定的时候使用的工具描述
    public ToolInEndDesc toolDesc;
};
```

坐标系类型（参数 `coordType`）枚举如下：

```
const int BaseCoordinate = 0;
const int EndCoordinate = 1;
const int WorldCoordinate = 2;
```

坐标系标定方法（参数 `methods`）枚举如下：

```
// 原点、x 轴正半轴、y 轴正半轴
const int Origin_AnyPointOnPositiveXAxis_AnyPointOnPositiveYAxis = 0;
// 原点、y 轴正半轴、z 轴正半轴
const int Origin_AnyPointOnPositiveYAxis_AnyPointOnPositiveZAxis = 1;
// 原点、z 轴正半轴、x 轴正半轴
const int Origin_AnyPointOnPositiveZAxis_AnyPointOnPositiveXAxis = 2;
// 原点、x 轴正半轴、x、y 轴平面的第一象限上任意一点
const int Origin_AnyPointOnPositiveXAxis_AnyPointOnFirstQuadrantOfXOYPlane
= 3;
// 原点、x 轴正半轴、x、z 轴平面的第一象限上任意一点
const int Origin_AnyPointOnPositiveXAxis_AnyPointOnFirstQuadrantOfXOZPlane
= 4;
// 原点、y 轴正半轴、y、z 轴平面的第一象限上任意一点
const int Origin_AnyPointOnPositiveYAxis_AnyPointOnFirstQuadrantOfYOZPlane
= 5;
// 原点、y 轴正半轴、y、x 轴平面的第一象限上任意一点
const int Origin_AnyPointOnPositiveYAxis_AnyPointOnFirstQuadrantOfYOXPlane
= 6;
// 原点、z 轴正半轴、z、x 轴平面的第一象限上任意一点
const int Origin_AnyPointOnPositiveZAxis_AnyPointOnFirstQuadrantOfZOXPlane
= 7;
// 原点、z 轴正半轴、z、y 轴平面的第一象限上任意一点
const int Origin_AnyPointOnPositiveZAxis_AnyPointOnFirstQuadrantOfZOYPlane
= 8;
```

1.2.2 MoveRotateAxis 转轴定义

```
[StructLayout(LayoutKind.Sequential)]
public struct MoveRotateAxis
{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
    public double[] rotateAxis;
};
```

1.3 工具

1.3.1 ToolInEndDesc 机械臂工具端参数

```
[StructLayout(LayoutKind.Sequential)]
public struct ToolInEndDesc
{
    //工具相对于末端坐标系的位置
    public Pos cartPos;
    //工具相对于末端坐标系的姿态
    public Ori orientation;
}
```

1.3.2 ToolCalibrate 工具标定结构体

```
[StructLayout(LayoutKind.Sequential)]
public struct ToolCalibrate
{
    //用于位置标定点的数量
    public int posCalibrateNum;
    //位置标定点
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 4)]
    public wayPoint_S[] posCalibrateWaypoint;
    //用于姿态标定点的数量
    public int oriCalibrateNum;
    //姿态标定点
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 3)]
    public wayPoint_S[] oriCalibrateWaypoint;
    public int CalibMethod;
};
```

工具标定方法（参数 `CalibMethod`）枚举如下：

```
// 原点、x 轴正半轴、x、y 轴平面的第一象限上任意一点
const int ToolKinematicsOriCalibrateMathod_xOxy = 0;
// 原点、y 轴正半轴、y、z 轴平面的第一象限上任意一点
const int ToolKinematicsOriCalibrateMathod_yOyz = 1;
// 原点、z 轴正半轴、z、x 轴平面的第一象限上任意一点
const int ToolKinematicsOriCalibrateMathod_zOzx = 2;
// 工具 x 轴平行反向于基坐标系 z 轴；工具 xOy 平面平行于基坐标系 z 轴、工具 y 轴与基坐标系负 z 轴夹角为锐角
const int ToolKinematicsOriCalibrateMathod_TxRBz_TxyPBzAndTyABnz = 3;
// 工具 y 轴平行反向于基坐标系 z 轴；工具 yOz 平面平行于基坐标系 z 轴、工具 z 轴与基坐标系负 z 轴夹角为锐角
```

```
const int ToolKinematicsOriCalibrateMethod_TyRBz_TyzPBzAndTzABnz = 4;
// 工具 z 轴平行反向于基坐标系 z 轴; 工具 zOx 平面平行于基坐标系 z 轴、工具 x
// 轴与基坐标系负 z 轴夹角为锐角
const int ToolKinematicsOriCalibrateMethod_TzRBz_TzxPBzAndTxABnz = 5;
```

1.3.3 ToolInertia 该结构体描述工具惯量

```
[StructLayout(LayoutKind.Sequential)]
public struct ToolInertia
{
    public double xx;
    public double xy;
    public double xz;
    public double yy;
    public double yz;
    public double zz;
};
```

1.3.4 ToolDynamicsParam 动力学参数

```
[StructLayout(LayoutKind.Sequential)]
public struct ToolDynamicsParam
{
    public double positionX; //工具重心的 X 坐标
    public double positionY; //工具重心的 Y 坐标
    public double positionZ; //工具重心的 Z 坐标
    public double payload; //工具重量
    public ToolInertia toolInertia; //工具惯量
};
```

1.4 机械臂相关信息

1.4.1 RobotEventInfo 机械臂事件

```
[StructLayout(LayoutKind.Sequential)]
public struct RobotEventInfo
{
    public int eventType; //事件类型号
    public int eventCode; //事件代码
    public IntPtr eventContent; //事件内容(std::string)
};
```

1.4.2 RobotDiagnosis 机械臂诊断信息

```
[StructLayout(LayoutKind.Sequential)]
public struct RobotDiagnosis
{
    public Byte armCanbusStatus;           // CAN 通信状态:0x01~0x80: 关节
    CAN 通信错误 (每个关节占用 1bit) 0x00: 无错误
    public float armPowerCurrent;          // 机械臂 48V 电源当前电流
    public float armPowerVoltage;          // 机械臂 48V 电源当前电压
    public Byte armPowerStatus;            // 机械臂 48V 电源状态 (开、关)
    public Byte contorllerTemp;            // 控制箱温度
    public Byte contorllerHumidity;        // 控制箱湿度
    public Byte remoteHalt;                // 远程关机信号
    public Byte softEmergency;             // 机械臂软急停
    public Byte remoteEmergency;           // 远程急停信号
    public Byte robotCollision;            // 碰撞检测位
    public Byte forceControlMode;          // 机械臂进入力控模式标志位
    public Byte brakeStuats;               // 刹车状态
    public float robotEndSpeed;            // 末端速度
    public int robotMaxAcc;                 // 最大加速度
    public Byte orpeStatus;                // 上位机软件状态位
    public Byte enableReadPose;           // 位姿读取使能位
    public Byte robotMountingPoseChanged; // 安装位置状态
    public Byte encoderErrorStatus;        // 磁编码器错误状态
    public Byte staticCollisionDetect;     // 静止碰撞检测开关
    public Byte jointCollisionDetect;      // 关节碰撞检测 每个关节占用 1bi
    t 0-无碰撞 1-存在碰撞
    public Byte encoderLinesError;         // 光电编码器不一致错误 0-无错
    误 1-有错误
    public Byte jointErrorStatus;          // 关节错误状态
    public Byte singularityOverSpeedAlarm; // 机械臂奇异点过速警告
    public Byte robotCurrentAlarm;        // 机械臂电流错误警告
    public Byte toolIoError;              // 工具错误
    public Byte robotMountingPoseWarning; // 机械臂安装位置错位 (只在力控
    模式下起作用)
    public ushort macTargetPosBufferSize; // mac 缓冲器长度, 预留
    public ushort macTargetPosDataSize;   // mac 缓冲器有效数据长度, 预留
    public Byte macDataInterruptWarning;  // mac 数据中断, 预留
    public Byte controlBoardAbnormalStateFlag; // 主控板(接口板)异常状态标志
};
```

1.4.3 JointStatus 关节状态信息

```
[StructLayout(LayoutKind.Sequential)]
public struct JointStatus
{
    public int jointCurrentI;      // 关节电流
    public int jointSpeedMoto;     // 关节速度
    public float jointPosJ;        // 关节角
    public float jointCurVol;     // 关节电压 单位: mV
    public float jointCurTemp;    // 当前温度
    public int jointTagCurrentI;   // 电机目标电流
    public float jointTagSpeedMoto; // 电机目标速度
    public float jointTagPosJ;     // 目标关节角 单位: 弧度
    public short jointErrorNum;    // 关节错误码
};
```

1.4.4 JointVersion 关节版本信息

```
[StructLayout(LayoutKind.Sequential)]
public struct JointVersion
{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
    public char[] hw_version; // 硬件版本信息
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
    public char[] sw_version; // 固件版本信息
};
```

1.4.5 JointProductID 机械臂 ID 信息

```
[StructLayout(LayoutKind.Sequential)]
public struct JointProductID
{
    [MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
    public char[] productID;
};
```

1.4.6 RobotDevInfo 设备信息

```
[StructLayout(LayoutKind.Sequential)]
public struct RobotDevInfo
{
    
```

```
public Byte type; // 设备型号、芯片型号：上位机主站：0x01 接口板 0x02
[MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
public char[] revision; // 设备版本号，eg:V1.0
[MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
public char[] manu_id; // 厂家ID，"OUR "的ASCII 码 0x4F 55 52 00
[MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
public char[] joint_type; // 机械臂类型
[MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
public JointVersion[] joint_ver; // 机械臂关节及工具端信息
[MarshalAs(UnmanagedType.ByValArray, SizeConst = 64)]
public char[] desc; // 设备描述字符串以 0x00 结束
[MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]
public JointProductID[] jointProductID; // 关节ID 信息
[MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
public char[] slave_version; // 从设备版本号 - 字符串表示，如“V1.0.0”
[MarshalAs(UnmanagedType.ByValArray, SizeConst = 16)]
public char[] extio_version; // IO 扩展板版本号 -字符串标志，如“V1.0.0”

};
```


2 接口函数

2.1 机械臂系统接口

2.1.1 rs_login 机械臂登录

| rs_login(UInt16 rshd, [MarshalAs(UnmanagedType.LPStr)] string addr, int port) | |
|---|---|
| 功能描述: | 登录，与机械臂服务器建立网络连接。该接口的成功是调用其他接口的前提，只有在该接口正确返回的情况下，才能使用其他接口。详细用法请见 4.1 章 。 |
| 参数说明: | <ol style="list-style-type: none"> 1. rshd: 控制上下文句柄。 2. addr: 机械臂服务器的 IP 地址。 3. port: 机械臂服务器的端口号，默认为 8899。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.1.2 rs_logout 退出登录

| rs_logout(UInt16 rshd) | |
|------------------------|--------------------|
| 功能描述: | 退出登录，断开与机械臂服务器的连接。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.1.3 rs_robot_startup 启动机械臂

| rs_robot_startup(UInt16 rshd, ref ToolDynamicsParam tool, byte colli_class, bool read_pos, bool static_colli_detect, int board_maxacc, ref int state) | |
|---|---|
| 功能描述: | 启动机械臂，即初始化-----该操作会完成机械臂的上电，松刹车，设置碰撞等级，设置动力学参数等功能。详细用法请见 4.1 章 。 |
| 参数说明: | <ol style="list-style-type: none"> 1. rshd: 控制上下文句柄。 2. tool: 动力学参数。如果末端夹持工具，此参数应该根据具体的来设定；如果末端没有夹持工具，将此参数的各项设置为 0。 |

| | |
|------|--|
| | 3. colli_class: 碰撞等级。 4. read_pos: 是否允许读取位置，默认是 true。 5. static_colli_detect: 是否允许侦测静态碰撞，默认为 true。 6. board_maxacc: 接口板允许的最大加速度，默认为 1000。 7. state: 机械臂启动状态。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.1.4 rs_robot_shutdown 关机

| rs_robot_shutdown(UInt16 rshd) | |
|--------------------------------|----------------|
| 功能描述: | 机械臂断电。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.1.5 rs_initialize 初始化机械臂控制库

| rs_initialize() | |
|-----------------|--|
| 功能描述: | 初始化机械臂控制库。 详细用法请见 4.1 章 。 |
| 参数说明: | 无。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.1.6 rs_uninitialize 反初始化机械臂控制

| rs_uninitialize() | |
|-------------------|-------------|
| 功能描述: | 反初始化机械臂控制库。 |
| 参数说明: | 无。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.1.7 rs_create_context 创建机械臂控制上下文句柄

| rs_create_context(ref UInt16 rshd) | |
|------------------------------------|---|
| 功能描述: | 创建机械臂控制上下文句柄。 详细用法请见 4.1 章 。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.1.8 rs_destory_context 注销机械臂控制上下文句柄

| rs_destory_context(UInt16 rshd) | |
|---------------------------------|----------------|
| 功能描述: | 注销机械臂控制上下文句柄。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.2 状态推送

2.2.1 rs_enable_push_realtime_roadpoint 设置是否允许实时路点信息推送

| rs_enable_push_realtime_roadpoint(UInt16 rshd, bool enable) | |
|---|---|
| 功能描述: | 设置是否允许实时路点信息推送。 详细用法请见 4.2.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. enable: 是否允许, true 表示允许, false 表示不允许。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.2.2 rs_setcallback_realtime_roadpoint 获取实时路点信息的回调函数

| rs_setcallback_realtime_roadpoint(UInt16 rshd, [MarshalAs(UnmanagedType.FunctionPtr)] REALTIME_ROADPOINT_CALLBACK CurrentPositionCallback, IntPtr arg) | |
|--|---|
| 功能描述: | 获取实时路点信息的回调函数。 详细用法请见 4.2.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. CurrentPositionCallback: 实时路点回调函数。 3. arg: 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.2.3 REALTIME_ROADPOINT_CALLBACK 实时路点的回调

函数类型

| REALTIME_ROADPOINT_CALLBACK(ref wayPoint_S waypoint, IntPtr arg) | |
|--|---|
| 功能描述: | 实时路点的回调函数类型。 详细用法请见 4.2.1 章 。 |
| 参数说明: | 1. waypoint : 路点信息。 2. arg : 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.2.4 CurrentPositionCallback 路点回调

| CurrentPositionCallback(ref wayPoint_S waypoint, IntPtr arg) | |
|--|---|
| 功能描述: | 路点回调。 |
| 参数说明: | 1. waypoint : 路点信息。 2. arg : 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.2.5 PrintWaypoint 打印路点信息

| PrintWaypoint(wayPoint_S point) | |
|---------------------------------|----------------------|
| 功能描述: | 打印路点信息。 |
| 参数说明: | point : 路点信息。 |
| 返回值: | 无。 |

2.2.6 rs_setcallback_realtime_end_speed 获取实时末端速度信息

| rs_setcallback_realtime_end_speed(UInt16 rshd, [MarshalAs(UnmanagedType.FunctionPtr)] REALTIME_ENDSPEED_CALLBACK CurrentEndSpeedCallback, IntPtr arg) | |
|---|---|
| 功能描述: | 获取实时末端速度信息的回调函数。 详细用法请见 4.2.2 章 。 |
| 参数说明: | 1. rshd : 控制上下文句柄。 2. CurrentEndSpeedCallback : 实时末端速度回调函数。 3. arg : 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.2.7 REALTIME_ENDSPEED_CALLBACK 实时末端速度的回调函数类型

| REALTIME_ENDSPEED_CALLBACK(double speed, IntPtr arg) | |
|--|--|
| 功能描述: | 实时末端速度的回调函数类型。 |
| 参数说明: | <ol style="list-style-type: none"> 1. speed: 末端速度。 2. arg: 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.2.8 CurrentEndSpeedCallback 速度回调

| CurrentEndSpeedCallback(double speed, IntPtr arg) | |
|---|--|
| 功能描述: | 速度回调。 |
| 参数说明: | <ol style="list-style-type: none"> 1. speed: 末端速度。 2. arg: 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.2.9 rs_setcallback_robot_event 获取实时事件信息的回调函数

| rs_setcallback_robot_event(UInt16 rshd, [MarshalAs(UnmanagedType.FunctionPtr)] ROBOT_EVENT_CALLBACK RobotEventCallback, IntPtr arg) | |
|--|---|
| 功能描述: | 获取实时事件信息的回调函数。 详细用法请见 4.2.3 章 。 |
| 参数说明: | <ol style="list-style-type: none"> 1. rshd: 控制上下文句柄。 2. RobotEventCallback: 实时事件回调函数。 3. arg: 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.2.10 ROBOT_EVENT_CALLBACK 机械臂事件的回调函数类型

| ROBOT_EVENT_CALLBACK(ref RobotEventInfo rs_event, IntPtr arg) | |
|---|---|
| 功能描述: | 机械臂事件的回调函数类型。 |
| 参数说明: | <ol style="list-style-type: none"> 1. rs_event: 事件信息。 2. arg: 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.2.11 RobotEventCallback 事件信息回调

| RobotEventCallback(ref RobotEventInfo rs_event, IntPtr arg) | |
|---|---|
| 功能描述: | 事件信息回调。 |
| 参数说明: | 1. rs_event : 事件信息。 2. arg : 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.2.12 rs_setcallback_realtime_joint_status 获取实时关节状态

| rs_setcallback_realtime_joint_status(UInt16 rshd, [MarshalAs(UnmanagedType.FunctionPtr)] ROBOT_JOINT_STATUS_CALLBACK RobotJointStatusCallback, IntPtr arg) | |
|--|--|
| 功能描述: | 获取实时关节状态的回调函数。 详细用法请见 4.2.4 章 。 |
| 参数说明: | 1. rshd : 控制上下文句柄。 2. RobotJointStatusCallback : 实时关节状态回调函数。 3. arg : 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.2.13 ROBOT_JOINT_STATUS_CALLBACK 机械臂关节状态的回调函数类型

| ROBOT_JOINT_STATUS_CALLBACK(IntPtr pBuff, int size, IntPtr arg) | |
|---|--|
| 功能描述: | 机械臂关节状态的回调函数类型。 |
| 参数说明: | 1. pBuff : 关节状态。 2. size : 关节数量。 3. arg : 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.2.14 CurrentJointStatusCallback 关节状态回调

| CurrentJointStatusCallback(IntPtr pBuff, int size, IntPtr arg) | |
|--|--|
| 功能描述: | 关节状态回调。 |
| 参数说明: | 1. pBuff : 关节状态。 2. size : 关节数量。 3. arg : 这个参数系统不做任何处理, 只是进行缓存, 当系统调用已注册回调函数时, 该参数会通过回调函数的参数传回。 |
| 返回值: | 无。 |

2.3 机械臂运动相关的接口

2.3.1 rs_init_global_move_profile 初始化全局的运动属性

| rs_init_global_move_profile(UInt16 rshd) | |
|--|---|
| 功能描述: | 初始化全局的运动属性。 注: 调用成功后, 系统会自动清理掉之前设置的用户坐标系, 速度, 加速度等等属性。 |
| 参数说明: | rshd : 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.2 rs_set_global_joint_maxacc 设置关节型运动的最大加速度

| rs_set_global_joint_maxacc(UInt16 rshd, double[] max_acc) | |
|---|---|
| 功能描述: | 设置关节型运动的最大加速度。 注意如果没有特殊需求, 6 个关节尽量配置成一样。 |
| 参数说明: | 1. rshd : 控制上下文句柄。 2. max_acc : 六个关节的最大加速度, 单位 rad/s^2 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.3 rs_set_global_joint_maxvelc 设置关节型运动的最大速度

| rs_set_global_joint_maxvelc(UInt16 rshd, double[] max_velc) | |
|---|---|
| 功能描述: | 设置关节型运动的最大速度, 最大为 180 度/秒。 注意如果没有特殊需求, 6 个关节尽量配置成一样。 |
| 参数说明: | 1. rshd : 控制上下文句柄。 2. max_velc : 六个关节的最大速度, 单位 rad/s 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.4 rs_get_global_joint_maxacc 获取关节型运动的最大加速度

| rs_get_global_joint_maxacc(UInt16 rshd, ref JointVelcAccParam max_acc) | |
|--|---|
| 功能描述: | 获取关节型运动的最大加速度。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. max_acc: 六个关节的最大加速度, 单位 rad/s^2 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.5 rs_get_global_joint_maxvelc 获取关节型运动的最大速度

| rs_get_global_joint_maxvelc(UInt16 rshd, ref JointVelcAccParam max_velc) | |
|--|---|
| 功能描述: | 获取关节型运动的最大速度。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. max_velc: 六个关节的最大速度, 单位 rad/s 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.6 rs_move_joint 关节运动

| rs_move_joint(UInt16 rshd, double[] joint_radia, bool isblock) | |
|--|---|
| 功能描述: | 运动接口之关节运动。 详细用法请见 4.9.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. joint_radian: 六个关节的关节角, 单位 rad 。 3. isblock: 是否阻塞。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.7 rs_move_joint_to 保持当前位姿通过关节运动的方式运动到目标位置

| rs_move_joint_to(UInt16 rshd, ref Pos target, ref ToolInEndDesc tool, bool isblock) | |
|---|--|
| 功能描述: | 机械臂轴动到目标位置。 详细用法请见 4.9.2 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. target: 位置坐标(x, y, z), 单位 m 。 3. tool: 工具描述。 4. isblock: 是否阻塞。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.8 rs_set_no_arrival_ahead 取消提前到位设置

| rs_set_no_arrival_ahead(UInt16 rshd) | |
|--------------------------------------|---|
| 功能描述: | 取消提前到位设置。 详细用法请见 4.8.1 章 。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.9 rs_set_arrival_ahead_distance 设置提前到位距离模式

| rs_set_arrival_ahead_distance(UInt16 rshd, double distance) | |
|---|---|
| 功能描述: | 设置距离模式下的提前到位距离。 详细用法请见 4.8.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. distance: 提前到位距离, 单位 m 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.10 rs_set_arrival_ahead_time 设置提前到位时间模式

| rs_set_arrival_ahead_time(UInt16 rshd, double sec) | |
|--|---|
| 功能描述: | 设置时间模式下的提前到位时间。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. sec: 提前到位时间, 单位 s 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.11 rs_set_global_end_max_line_acc 设置末端型运动的最大线加速度

| rs_set_global_end_max_line_acc(UInt16 rshd, double max_acc) | |
|---|---|
| 功能描述: | 设置末端型运动的最大线加速度。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. max_acc: 末端最大线加速度, 单位 m/s^2 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.12 rs_set_global_end_max_line_velc 设置末端型运动的最大线速度

| rs_set_global_end_max_line_velc(UInt16 rshd, double max_velc) | |
|---|---|
| 功能描述: | 设置末端型运动的最大线速度。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. max_velc: 末端最大线速度, 单位 m/s 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.13 rs_get_global_end_max_line_acc 获取末端型运动的最大线加速度

| rs_get_global_end_max_line_acc(UInt16 rshd, ref double max_acc) | |
|---|--|
| 功能描述: | 获取末端型运动最大线加速度。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. max_acc: 机械臂末端最大线加速度, 单位 m/s^2 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.14 rs_get_global_end_max_line_velc 获取末端型运动的最大线速度

| rs_get_global_end_max_line_velc(UInt16 rshd, ref double max_velc) | |
|---|--|
| 功能描述: | 获取末端型运动的最大线速度。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. max_velc: 机械臂末端最大线速度, 单位 m/s 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.15 rs_set_global_end_max_angle_acc 设置末端型运动的最大角加速度

| rs_set_global_end_max_angle_acc(UInt16 rshd, double max_acc) | |
|--|---|
| 功能描述: | 设置末端型运动的最大角加速度。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. max_acc: 末端最大角加速度, 单位 rad/s^2 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.16 rs_set_global_end_max_angle_velc 设置末端型运动的最大角速度

| rs_set_global_end_max_angle_velc(UInt16 rshd, double max_velc) | |
|--|--|
| 功能描述: | 设置末端型运动的最大角速度。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. max_velc: 末端最大速度, 单位 rad/s 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.17 rs_get_global_end_max_angle_acc 获取末端型运动的最大角加速度

| rs_get_global_end_max_angle_acc(UInt16 rshd, ref double max_acc) | |
|--|---|
| 功能描述: | 获取末端型运动的最大角加速度。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. max_acc: 末端最大角加速度, 单位 rad/s^2 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.18 rs_get_global_end_max_angle_velc 获取末端型运动的最大角速度

| rs_get_global_end_max_angle_velc(UInt16 rshd, ref double max_velc) | |
|--|---|
| 功能描述: | 获取末端型运动的最大角速度。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. max_velc: 机械臂末端最大速度, 单位 rad/s |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.19 rs_move_line 直线运动

| rs_move_line(UInt16 rshd, double[] joint_radia, bool isblock) | |
|---|---|
| 功能描述: | 机械臂保持当前姿态做直线运动。 详细用法请见 4.9.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. joint_radian: 六个关节的关节角, 单位 rad 。 3. isblock: 是否阻塞。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.20 rs_move_line_to 保持当前位姿通过直线运动的方式运动到目标位置

| rs_move_line_to(UInt16 rshd, ref Pos target, ref ToolInEndDesc tool, bool isblock) | |
|--|---|
| 功能描述: | 机械臂轴动到目标位置。 详细用法请见 4.9.2 章 。 |
| 参数说明: | 5. rshd: 控制上下文句柄。 6. target: 位置坐标(x, y, z), 单位 <i>m</i> 。 7. tool: 工具描述。 8. isblock: 是否阻塞。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.21 rs_move_rotate 保持当前位置变换姿态做旋转运动

| rs_move_rotate(UInt16 rshd, ref CoordCalibrate user_coord, ref MoveRotateAxis rotate_axis, double rotate_angle, bool isblock) | |
|---|---|
| 功能描述: | 保持当前位置变换姿态做旋转运动。 详细用法请见 4.11.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. user_coord: 用户坐标系 3. rotate_axis: 转轴(x,y,z)。例如: (1,0,0)表示沿 Y 轴转动。 4. rotate_angle: 旋转角度, 单位 <i>rad</i> 。 5. isblock: 是否阻塞。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.22 rs_remove_all_waypoint 清除路点容器

| rs_remove_all_waypoint(UInt16 rshd) | |
|-------------------------------------|----------------|
| 功能描述: | 清除所有已经设置的全局路点。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.23 rs_add_waypoint 添加路点

| rs_add_waypoint(UInt16 rshd, double[] joint_radia) | |
|--|--|
| 功能描述: | 清除所有已经设置的全局路点 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. joint_radia: 六个关节的关节角, 单位 <i>rad</i> 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.24 rs_set_blend_radius 设置交融半径

| rs_set_blend_radius(UInt16 rshd, double radius) | |
|---|---|
| 功能描述: | 设置交融半径。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. blend_radius: 交融半径, 单位 <i>m</i> 。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.25 rs_set_circular_loop_times 设置圆轨迹时圆的圈数

| rs_set_circular_loop_times(UInt16 rshd, int times) | |
|--|--|
| 功能描述: | 设置圆运动圈数。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. times: 圆的运动圈数。 当 times 大于 0 时, 机械臂进行圆运动 times 次; 当 times 等于 0 时, 机械臂进行圆弧轨迹运动。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.26 rs_move_track 轨迹运动

| rs_move_track(UInt16 rshd, int sub_move_mode, bool isblock) | |
|---|---|
| 功能描述: | 轨迹运动。 详细用法请见 4.12 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. sub_move_mode: 轨迹类型。 3. isblock: 是否阻塞。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.27 rs_set_relative_offset_on_base 设置基于基坐标系运动偏移量

| rs_set_relative_offset_on_base(UInt16 rshd, ref MoveRelative relative) | |
|--|--|
| 功能描述: | 设置基于基坐标系下的相对偏移属性。 详细用法请见 4.10.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. relative: 相对偏移。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.3.28 rs_set_relative_offset_on_user 设置基于用户标系运动偏移量

| | |
|---|---|
| rs_set_relative_offset_on_user(UInt16 rshd, ref MoveRelative relative, ref CoordCalibrate user_coord) | |
| 功能描述: | 设置基于用户标系运动偏移量。 详细用法请见 4.10.2 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. relative: 偏移量。 3. user_coord: 用户坐标系。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.4 运动学相关的接口

2.4.1 rs_forward_kin 正解

| | |
|--|---|
| rs_forward_kin(UInt16 rshd, double[] joint_radia, ref waypoint_S waypoint) | |
| 功能描述: | 正解，此函数为正解函数，已知关节角求对应位置的位置和姿态。 详细用法请见 4.3.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. joint_radia: 六个关节的关节角，单位 <i>rad</i> 。 3. waypoint: 路点信息。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.4.2 rs_inverse_kin 逆解

| | |
|--|--|
| rs_inverse_kin(UInt16 rshd, double[] joint_radia, ref Pos pos, ref Ori ori, ref waypoint_S waypoint) | |
| 功能描述: | 逆解。 详细用法请见 4.3.2 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. joint_radia: 起始点六个关节的关节角，单位 <i>rad</i> 。 3. pos: 位置(x, y, z)，单位 <i>m</i> 。 4. ori: 姿态(w, x, y, z)。 5. waypoint: 路点信息。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.4.3 rs_set_base_coord 设置基坐标系

| | |
|--------------------------------|----------------|
| rs_set_base_coord(UInt16 rshd) | |
| 功能描述: | 设置基坐标系。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.4.4 rs_set_user_coord 设置用户坐标系

| | |
|---|--|
| rs_set_user_coord(UInt16 rshd, ref CoordCalibrate user_coord) | |
| 功能描述: | 设置用户坐标系。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. user_coord: 用户坐标系。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.4.5 rs_check_user_coord 检查用户坐标系参数设置是否合理

| | |
|---|--|
| rs_check_user_coord(UInt16 rshd, ref CoordCalibrate user_coord) | |
| 功能描述: | 检查用户坐标系参数设置是否合理。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. user_coord: 用户坐标系。 |
| 返回值: | 成功: 返回 0。 |
| | 不成功: 返回错误号。 |

2.4.6 rs_base_to_user 基坐标系转用户坐标系

| | |
|---|--|
| rs_base_to_user(UInt16 rshd, ref Pos pos_onbase, ref Ori ori_onbase, ref CoordCalibrate user_coord, ref ToolInEndDesc tool_pos, ref Pos pos_onuser, ref Ori ori_onuser) | |
| 功能描述: | 将法兰盘中心基于基坐标系下的位置和姿态转成工具末端基于用户坐标系下的位置和姿态。 详细用法请见 4.4.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 |

| | |
|------|---|
| | <ol style="list-style-type: none"> 2. pos_onbase: 基坐标系下的位置(x, y, z), 单位 <i>m</i>。 3. ori_onbase: 基坐标系下的姿态(w, x, y, z)。 4. user_coord: 用户坐标系。 5. tool_pos: 用户工具描述。 6. pos_onuser: 用户坐标系下的位置(x, y, z), 单位 <i>m</i>。 7. ori_onuser: 用户坐标系下的姿态(w, x, y, z)。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.4.7 rs_base_to_base_additional_tool 基坐标系转基坐标得到工具末端点的位置和姿态

| | |
|---|---|
| rs_base_to_base_additional_tool(UInt16 rshd, ref Pos flange_center_pos_onbase, ref Ori flange_center_ori_onbase, ref ToolInEndDesc tool_pos, ref Pos tool_end_pos_onbase, ref Ori tool_end_ori_onbase) | |
| 功能描述: | 法兰盘中心在基坐标系下的位置和姿态转工具末端在基坐标系下的位置和姿态。 详细用法请见 4.4.2 章 。 |
| 参数说明: | <ol style="list-style-type: none"> 1. rshd: 控制上下文句柄。 2. flange_center_pos_onbase: 法兰盘中心在基坐标系下的位置(x, y, z), 单位 <i>m</i>。 3. flange_center_ori_onbase: 法兰盘中心在基坐标系下的姿态(w, x, y, z)。 4. tool_pos: 用户工具描述。 5. tool_end_pos_onbase: 工具末端在基坐标系下的位置(x, y, z), 单位 <i>m</i>。 6. tool_end_ori_onbase: 工具末端在基坐标系下的姿态(w, x, y, z)。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.4.8 rs_user_to_base 用户坐标系转基坐标系

| | |
|---|--|
| rs_user_to_base(UInt16 rshd, ref Pos pos_onuser, ref Ori ori_onuser, ref CoordCalibrate user_coord, ref ToolInEndDesc tool_pos, | |
|---|--|

| | |
|--|--|
| ref Pos pos_onbase, ref Ori ori_onbase) | |
| 功能描述: | 将工具末端基于用户坐标系下的位置和姿态转成法兰盘中心基于基坐标系下的位置和姿态。 详细用法请见 4.4.3 章 。 |
| 参数说明: | <ol style="list-style-type: none"> 1. rshd: 控制上下文句柄。 2. pos_onuser: 用户坐标系下的位置(x, y, z), 单位 <i>m</i>。 3. ori_onuser: 用户坐标系下的姿态(w, x, y, z)。 4. user_coord: 用户坐标系。 5. tool_pos: 用户工具描述。 6. pos_onbase: 基坐标系下的位置(x, y, z), 单位 <i>m</i>。 7. ori_onbase: 基坐标系下的姿态(w, x, y, z)。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.4.9 rs_rpy_to_quaternion 欧拉角转四元数

| | |
|---|--|
| rs_rpy_to_quaternion(UInt16 rshd, ref Rpy rpy, ref Ori ori) | |
| 功能描述: | 欧拉角转四元数。 |
| 参数说明: | <ol style="list-style-type: none"> 1. rshd: 控制上下文句柄。 2. rpy: 欧拉角(rx, ry, rz), 单位 <i>rad</i>。 3. ori: 四元数(w, x, y, z)。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.4.10 rs_quaternion_to_rpy 四元数转欧拉角

| | |
|---|--|
| rs_quaternion_to_rpy(UInt16 rshd, ref Ori ori, ref Rpy rpy) | |
| 功能描述: | 四元数转欧拉角。 |
| 参数说明: | <ol style="list-style-type: none"> 1. rshd: 控制上下文句柄。 2. ori: 四元数(w, x, y, z)。 3. rpy: 欧拉角(rx, ry, rz), 单位 <i>rad</i>。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.5 机械臂控制接口

2.5.1 rs_move_stop 机械臂运动停止

| | |
|---------------------------|---|
| rs_move_stop(UInt16 rshd) | |
| 功能描述: | 停止机械臂运动。 注: 需要在与 move 不同的线程中调用。 |

| | |
|-------|----------------|
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.5.2 rs_move_fast_stop 快速停止机械臂运动

| | |
|--------------------------------|--------------------------------------|
| rs_move_fast_stop(UInt16 rshd) | |
| 功能描述: | 快速停止机械臂运动。 注: 需要在与 move 不同的线程中调用。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.5.3 rs_move_pause 暂停机械臂运动

| | |
|----------------------------|------------------------------------|
| rs_move_pause(UInt16 rshd) | |
| 功能描述: | 暂停机械臂运动。 注: 需要在与 move 不同的线程中调用。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.5.4 rs_move_continue 暂停后恢复机械臂运动

| | |
|-------------------------------|---------------------------------------|
| rs_move_continue(UInt16 rshd) | |
| 功能描述: | 暂停后恢复机械臂运动。 注: 需要在与 move 不同的线程中调用。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.5.5 rs_collision_recover 碰撞后恢复

| | |
|-----------------------------------|----------------|
| rs_collision_recover(UInt16 rshd) | |
| 功能描述: | 机械臂碰撞后恢复。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.5.6 rs_get_robot_state 获取机械臂当前运行状态

| rs_get_robot_state(UInt16 rshd, ref int state) | |
|--|--|
| 功能描述: | 获取机械臂当前运行状态。 注意：需要与 move 在不同线程里。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. state: 运行状态。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.6 末端工具接口

2.6.1 rs_set_none_tool_dynamics_param 设置无工具的动力学参数

| rs_set_none_tool_dynamics_param(UInt16 rshd) | |
|--|----------------|
| 功能描述: | 设置无工具的动力学参数。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.6.2 rs_set_tool_dynamics_param 设置工具的动力学参数

| rs_set_tool_dynamics_param(UInt16 rshd, ref ToolDynamicsParam tool) | |
|---|--------------------------------------|
| 功能描述: | 设置工具的动力学参数。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. tool: 动力学参数。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.6.3 rs_get_tool_dynamics_param 获取工具的动力学参数

| rs_get_tool_dynamics_param(UInt16 rshd, ref ToolDynamicsParam tool) | |
|---|--------------------------------------|
| 功能描述: | 获取工具的动力学参数。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. tool: 动力学参数。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.6.4 rs_set_tool_end_param 设置末端工具的运动学参数

| rs_set_tool_end_param(UInt16 rshd, ref ToolInEndDesc tool) | |
|--|---------------------------------------|
| 功能描述: | 设置末端工具的运动学参数。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. tool: 末端工具参数。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.6.5 rs_set_none_tool_kinematics_param 设置无工具的运动学参数

| rs_set_none_tool_kinematics_param(UInt16 rshd) | |
|--|----------------|
| 功能描述: | 设置无工具运动学参数。 |
| 参数说明: | rshd: 控制上下文句柄。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.6.6 rs_set_tool_kinematics_param 设置工具的运动学参数

| rs_set_tool_kinematics_param(UInt16 rshd, ref ToolInEndDesc tool) | |
|---|---|
| 功能描述: | 设置工具的运动学参数。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. tool: 末端工具的运动学参数。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.6.7 rs_get_tool_kinematics_param 获取工具的运动学参数

| rs_get_tool_kinematics_param(UInt16 rshd, ref ToolInEndDesc tool) | |
|---|---|
| 功能描述: | 获取工具的运动学参数。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. tool: 工具的运动学参数。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.7 设置和获取机械臂相关参数接口

2.7.1 rs_set_work_mode 设置当前机械臂模式：仿真或真实

| rs_set_work_mode(UInt16 rshd, int state) | |
|--|--|
| 功能描述: | 设置当前机械臂模式：仿真或真实。 详细用法请见 4.5.5 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. state: 服务器工作模式。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.7.2 rs_get_work_mode 获取当前机械臂模式：仿真或真实

| rs_get_work_mode(UInt16 rshd, ref int state) | |
|--|--|
| 功能描述: | 获取机械臂当前工作模式。 详细用法请见 4.5.5 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. state: 服务器工作模式。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.7.3 rs_set_collision_class 设置碰撞等级

| rs_set_collision_class(UInt16 rshd, int grade) | |
|--|---|
| 功能描述: | 设置碰撞等级。 详细用法请见 4.5.6 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. grade: 碰撞等级: 0~10。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.7.4 rs_get_collision_class 获取当前碰撞等级

| rs_get_collision_class(UInt16 rshd, ref int grade) | |
|--|---|
| 功能描述: | 获取当前碰撞等级。 详细用法请见 4.5.6 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. grade: 碰撞等级: 0~10。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.7.5 rs_get_joint_status 获取机械臂关节状态

| rs_get_joint_status(UInt16 rshd, IntPtr pBuff) | |
|--|--|
| 功能描述: | 获取机械臂关节状态。 详细用法请见 4.5.2 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. pBuff: 关节状态 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.7.6 rs_get_current_waypoint 获取机械臂当前路点信息

| rs_get_current_waypoint(UInt16 rshd, ref wayPoint_S waypoint) | |
|---|--|
| 功能描述: | 获取机械臂当前路点信息。 详细用法请见 4.5.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. waypoint: 路点信息。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.7.7 rs_get_socket_status 获取 socket 链接状态

| rs_get_socket_status(UInt16 rshd, ref byte connected) | |
|---|---|
| 功能描述: | 获取 socket 链接状态。 详细用法请见 4.5.8 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. connected: socket 链接状态。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.7.8 rs_get_diagnosis_info 获取机械臂诊断信息

| rs_get_diagnosis_info(UInt16 rshd, ref RobotDiagnosis robotDiagnosis) | |
|---|--|
| 功能描述: | 获取机械臂诊断信息。 详细用法请见 4.5.3 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. robotDiagnosis: 诊断信息。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.7.9 rs_get_device_info 获取机械臂设备信息

| rs_get_device_info(UInt16 rshd, ref RobotDevInfo dev) | |
|---|--|
| 功能描述: | 获取机械臂设备信息。 详细用法请见 4.5.4 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. dev: 设备信息。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.7.10 rs_get_error_information_by_errcode 根据错误号返回错误信息

| rs_get_error_information_by_errcode(UInt16 rshd, int err_code) | |
|--|--|
| 功能描述: | 根据错误号返回错误信息。 详细用法请见 4.5.7 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. err_code: 错误号。 |
| 返回值: | 返回错误信息。 |

2.8 接口板 IO 相关的接口

2.8.1 rs_set_board_io_status_by_addr 根据接口板 IO 类型和地址设置 IO 状态

| rs_set_board_io_status_by_addr(UInt16 rshd, int io_type, int addr, double val) | |
|--|--|
| 功能描述: | 根据接口板 IO 类型和名称设置 IO 状态。 详细用法请见 4.6.2 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. io_type: IO 类型。 3. addr: IO 地址。 4. val: IO 状态数值。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.8.2 rs_get_board_io_status_by_addr 根据接口板 IO 类型和地址获取 IO 状态

| rs_get_board_io_status_by_addr(UInt16 rshd, int io_type, int addr, ref double val) | |
|--|--|
| 功能描述: | 根据接口板 IO 类型和名称获取 IO 状态。 详细用法请见 4.6.2 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. io_type: IO 类型。 3. addr: IO 地址。 4. val: IO 状态数值。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.9 工具 IO 相关的接口

2.9.1 rs_set_tool_power_type 设置工具端电源电压类型

| rs_set_tool_power_type(UInt16 rshd, int type) | |
|---|--|
| 功能描述: | 设置工具端电源电压类型。 详细用法请见 4.6.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. type: 电源电压类型。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.9.2 rs_get_tool_power_type 获取工具端电源电压类型

| rs_get_tool_power_type(UInt16 rshd, ref int type) | |
|---|--|
| 功能描述: | 获取工具端电源电压类型。 详细用法请见 4.6.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. type: 电源电压类型。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.9.3 rs_set_tool_io_type 设置工具端数字量 IO 的类型：输入或者输出

| rs_set_tool_io_type(UInt16 rshd, int addr, int type) | |
|--|---|
| 功能描述: | 设置工具端数字量 IO 的类型：输入或者输出。 详细用法请见 4.6.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. addr: 工具端 IO 地址。 3. type: 工具端 IO 类型。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.9.4 rs_get_tool_io_status 获取工具端 IO 的状态

| rs_get_tool_io_status(UInt16 rshd, string name, ref double val) | |
|---|---|
| 功能描述: | 根据名称获取工具端 IO 的状态。 详细用法请见 4.6.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. name: IO 名称。 3. val: IO 状态值。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

2.9.5 rs_set_tool_do_status 设置工具端 IO 的状态

| rs_set_tool_do_status(UInt16 rshd, string name, int val) | |
|--|---|
| 功能描述: | 根据名称设置工具端 IO 的状态。 详细用法请见 4.6.1 章 。 |
| 参数说明: | 1. rshd: 控制上下文句柄。 2. name: 工具端 IO 名称。 3. val: 工具端 IO 状态: 0 或 1。 |
| 返回值: | 成功: 返回 0。 |
| | 失败: 返回错误号。 |

3 错误码

3.1 接口函数错误码定义

| 错误号 | 错误代码 | 错误信息 |
|-------|--|-------------------|
| 0 | InterfaceCallSuccCode | 成功 |
| 10000 | ErrCode_Base | |
| 10001 | ErrCode_Failed | 通用失败 |
| 10002 | ErrCode_ParamError | 参数错误 |
| 10003 | ErrCode_ConnectSocketFailed | Socket 连接失败 |
| 10004 | ErrCode_SocketDisconnect | Socket 断开连接 |
| 10005 | ErrCode_CreateRequestFailed | 创建请求失败 |
| 10006 | ErrCode_RequestRelatedVariableError | 请求相关的内部变量出错 |
| 10007 | ErrCode_RequestTimeout | 请求超时 |
| 10008 | ErrCode_SendRequestFailed | 发送请求信息失败 |
| 10009 | ErrCode_ResponseInfoIsNULL | 响应信息为空 |
| 10010 | ErrCode_ResolveResponseFailed | 解析响应失败 |
| 10011 | ErrCode_FkFailed | 正解出错 |
| 10012 | ErrCode_IkFailed | 逆解出错 |
| 10013 | ErrCode_ToolCalibrateError | 工具标定参数有错 |
| 10014 | ErrCode_ToolCalibrateParamError | 工具标定参数有错 |
| 10015 | ErrCode_CoordinateSystemCalibrateError | 坐标系标定失败 |
| 10016 | ErrCode_BaseToUserConvertFailed | 基坐标系转用户坐标系失败 |
| 10017 | ErrCode_UserToBaseConvertFailed | 用户坐标系转基坐标系失败 |
| 10018 | ErrCode_MotionRelatedVariableError | 运动相关的内部变量出错 |
| 10019 | ErrCode_MotionRequestFailed | 运动请求失败 |
| 10020 | ErrCode_CreateMotionRequestFailed | 生成运动请求失败 |
| 10021 | ErrCode_MotionInterruptedByEvent | 运动被事件中断 |
| 10022 | ErrCode_MotionWaypointVetorSizeError | 运动相关的路点容器的长度不符合规定 |
| 10023 | ErrCode_ResponseReturnError | 服务器响应返回错误 |

| | | |
|-------|-----------------------------------|-------------------------------------|
| 10024 | ErrCode_RealRobotNoExist | 真实机械臂不存在，因为有些接口只有在真是机械臂存在的情况下才可以被调用 |
| 10025 | ErrCode_moveControlSlowStopFailed | 调用缓停接口失败 |
| 10026 | ErrCode_moveControlFastStopFailed | 调用急停接口失败 |
| 10027 | ErrCode_moveControlPauseFailed | 调用暂停接口失败 |
| 10028 | ErrCode_moveControlContinueFailed | 调用继续接口失败 |

3.2 由于控制器异常事件导致的错误码

| 错误号 | 错误代码 | 错误信息 |
|-------|---|------------------------|
| 21001 | ErrCodeMoveJConfigError | 关节运动属性配置错误 |
| 21002 | ErrCodeMoveLConfigError | 直线运动属性配置错误 |
| 21003 | ErrCodeMovePConfigError | 轨迹运动属性配置错误 |
| 21004 | ErrCodeInvailConfigError | 无效的运动属性配置 |
| 21005 | ErrCodeWaitRobotStopped | 等待机器人停止 |
| 21006 | ErrCodeJointOutOfRange | 超出关节运动范围 |
| 21007 | ErrCodeFirstWaypointSetError | 请正确设置 MOVEP 第一个路点 |
| 21008 | ErrCodeConveyorTrackConfigError | 传送带跟踪配置错误 |
| 21009 | ErrCodeConveyorTrackTrajectoryTypeError | 传送带轨迹类型错误 |
| 21010 | ErrCodeRelativeTransformIKFailed | 相对坐标变换逆解失败 |
| 21011 | ErrCodeTeachModeCollision | 示教模式发生碰撞 |
| 21012 | ErrCodeexternalToolConfigError | 运动属性配置错误,外部工具或手持工件配置错误 |
| 21101 | ErrCodeTrajectoryAbnormal | 轨迹异常 |

| | | |
|-------|-------------------------------------|---------------|
| 21102 | ErrCodeOnlineTrajectoryPlanError | 轨迹规划错误 |
| 21103 | ErrCodeOnlineTrajectoryTypeIIError | 二型在线轨迹规划失败 |
| 21104 | ErrCodeIKFailed | 逆解失败 |
| 21105 | ErrCodeAbnormalLimitProtect | 动力学限制保护 |
| 21106 | ErrCodeConveyorTrackingFailed | 传送带跟踪失败 |
| 21107 | ErrCodeConveyorOutWorkingRange | 超出传送带工作范围 |
| 21108 | ErrCodeTrajectoryJointOutOfRange | 关节超出范围 |
| 21109 | ErrCodeTrajectoryJointOverspeed | 关节超速 |
| 21110 | ErrCodeOfflineTrajectoryPlanFailed | 离线轨迹规划失败 |
| 21111 | ErrCodeTrajectoryJointAccOutOfRange | 轨迹异常,关节加速度超限 |
| 21120 | ErrCodeForceModeException | 力控模式异常 |
| 21121 | ErrCodeForceModeIKFailed | 轨迹异常, 力控模式下失败 |
| 21122 | ErrCodeForceModeTrackJointverspeed | 关节超速 |
| 21200 | ErrCodeControllerIKFailed | 控制器异常, 逆解失败 |
| 21201 | ErrCodeControllerStatusException | 控制器异常, 状态异常 |
| 21202 | ErrCodeControllerTrackingLost | 关节跟踪误差过大 |
| 21203 | ErrCodeMonitorErrTrackingLost | 关节跟踪误差过大 |
| 21204 | ErrCodeMonitorErrNoArrivalInTime | 预留 |
| 21205 | ErrCodeMonitorErrCurrentOverload | 预留 |
| 21206 | ErrCodeMonitorErrJointOutOfRange | 机械臂关节超出限制范围 |
| 21207 | ErrCodeFifoDataTimeNotRead | 缓存区超时未更新 |
| 21300 | ErrCodeMoveEnterStopState | 运动进入到 stop 阶段 |
| 21301 | ErrCodeMoveInterruptedByEvent | 运动被未知事件中断 |

3.3 由于硬件层异常事件导致的错误码

| 错误号 | 错误代码 | 错误信息 |
|-------|----------------------------|----------------|
| 22001 | ErrCodeHardwareErrorNotify | 机械臂硬件错误不能区分是哪种 |

| | | |
|-------|--------------------------------------|-------------------------|
| | | 硬件异常才会返回该错误 |
| 22101 | ErrCodeJointError | 机械臂关节错误 |
| 22102 | ErrCodeJointOverCurrent | 机械臂关节过流 |
| 22103 | ErrCodeJointOverVoltage | 机械臂关节过压 |
| 22104 | ErrCodeJointLowVoltage | 机械臂关节欠压 |
| 22105 | ErrCodeJointOverTemperature | 机械臂关节过温 |
| 22106 | ErrCodeJointHallError | 机械臂关节霍尔错误 |
| 22107 | ErrCodeJointEncoderError | 机械臂关节编码器错误 |
| 22108 | ErrCodeJointAbsoluteEncoderError | 机械臂关节绝对编码器错误 |
| 22109 | ErrCodeJointCurrentDetectError | 机械臂关节当前位置错误 |
| 22110 | ErrCodeJointEncoderPollution | 机械臂关节编码器污染。建议采取措施:警告性通知 |
| 22111 | ErrCodeJointEncoderZSignalError | 机械臂关节编码器 Z 信号错误 |
| 22112 | ErrCodeJointEncoderCalibrateInvalid | 机械臂关节编码器校准失效 |
| 22113 | ErrCodeJoint_IMU_SensorInvalid | 机械臂关节 IMU 传感器失效 |
| 22114 | ErrCodeJointTemperatureSensorError | 机械臂关节温度传感器出错 |
| 22115 | ErrCodeJointCanBusError | 机械臂关节 CAN 总线出错 |
| 22116 | ErrCodeJointCurrentError | 机械臂关节当前电流错误 |
| 22117 | ErrCodeJointCurrentPositionError | 机械臂关节当前位置错误 |
| 22118 | ErrCodeJointOverSpeed | 机械臂关节超速 |
| 22119 | ErrCodeJointOverAccelerate | 机械臂关节加速度过大错误 |
| 22120 | ErrCodeJointTraceAccuracy | 机械臂关节跟踪精度错误 |
| 22121 | ErrCodeJointTargetPositionOutOfRange | 机械臂关节目标位置超范围 |
| 22122 | ErrCodeJointTargetSpeedOutOfRange | 机械臂关节目标速度超范围 |
| 22123 | ErrCodeJointCollision | 建议采取措施:暂 |

| | | |
|-------|--------------------------------------|------------------------|
| | | 停当前运动 |
| 22200 | ErrCodeDataAbnormal | 机械臂信息异常 |
| 22201 | ErrCodeRobotTypeError | 机械臂类型错误 |
| 22202 | ErrCodeAccelerationSensorError | 机械臂加速度计芯片错误 |
| 22203 | ErrCodeEncoderLineError | 机械臂编码器线数错误 |
| 22204 | ErrCodeEnterDragAndTeachModeError | 机械臂进入拖动示教模式错误 |
| 22205 | ErrCodeExitDragAndTeachModeError | 机械臂退出拖动示教模式错误 |
| 22206 | ErrCodeMACDataInterruptionError | 机械臂 MAC 数据中断错误 |
| 22207 | ErrCodeDriveVersionError | 驱动器版本错误 (关节固件版本不一致) |
| 22300 | ErrCodeInitAbnormal | 机械臂初始化异常 |
| 22301 | ErrCodeDriverEnableFailed | 机械臂驱动器使能失败 |
| 22302 | ErrCodeDriverEnableAutoBackFailed | 机械臂驱动器使能自动回应失败 |
| 22303 | ErrCodeDriverEnableCurrentLoopFailed | 机械臂驱动器使能电流环失败 |
| 22304 | ErrCodeDriverSetTargetCurrentFailed | 机械臂驱动器设置目标电流失败 |
| 22305 | ErrCodeDriverReleaseBrakeFailed | 机械臂释放刹车失败 |
| 22306 | ErrCodeDriverEnablePostionLoopFailed | 机械臂使能位置环失败 |
| 22307 | ErrCodeSetMaxAccelerateFailed | 设置最大加速度失败 |
| 22400 | ErrCodeSafetyError | 机械臂安全出错 |
| 22401 | ErrCodeExternEmergencyStop | 机械臂外部紧急停止 |
| 22402 | ErrCodeSystemEmergencyStop | 机械臂系统紧急停止 |
| 22403 | ErrCodeTeachpendantEmergencyStop | 机械臂示教器紧急停止 |
| 22404 | ErrCodeControlCabinetEmergencyStop | 机械臂控制柜紧急停止 |
| 22405 | ErrCodeProtectionStopTimeout | 机械臂保护停止超时 |

| | | |
|-------|-----------------------------------|-----------------------------|
| 22406 | ErrCodeEeducedModeTimeout | 机械臂缩减模式 超时 |
| 22500 | ErrCodeSystemAbnormal | 机械臂系统异常 |
| 22501 | ErrCode_MCU_CommunicationAbnormal | 机械臂 mcu 通信 异常 |
| 22502 | ErrCode485CommunicationAbnormal | 机械臂 485 通信 异常 |
| 22550 | ErrCodeSoftEmergency | 软急停 |
| 22600 | ErrCodeArmPowerOff | 控制柜接触器断 开导致机械臂 48V 断电 |

4 接口函数示例

4.1 使用 SDK 构建一个最简单的机械臂的控制工程

本案例是使用 SDK 来构建一个最简单的机械臂的控制工程，包括初始化、创建上下文句柄、登录、上电。

代码如下：

```
static void template(UInt16 rshd)
{
    int result = 0xffff;

    //初始化机械臂控制库
    result = AuboRobot.rs_initialize();
    if (result == RS_SUCC)
    {
        Console.Out.WriteLine("机械臂初始化成功！");

        //创建机械臂控制上下文句柄
        result = AuboRobot.rs_create_context(ref rshd);
        if (result == RS_SUCC)
        {
            Console.Out.WriteLine("创建上下文句柄成功！");

            string ip = "192.168.219.132";
            int port = 8899;

            //机械臂登录
            result = AuboRobot.rs_login(rshd, ip, port);
            if (result == RS_SUCC)
            {
                Console.Out.WriteLine("登录成功！");
            }
            else
            {
                Console.Error.WriteLine("登录失败！ 错误码： {0}", result);
            }

            //机械臂上电
            AuboRobot.ToolDynamicsParam tool = new AuboRobot.ToolDynamicsParam();
            csParam();

            int state = 0;
        }
    }
}
```



```
        result = AuboRobot.rs_robot_startup(rshd, ref tool, 6, true, true, 100
0, ref state);
        if (result == RS_SUCC)
        {
            Console.Out.WriteLine("上电成功！状态：{0}", state);
        }
        else
        {
            Console.Error.WriteLine("上电失败！错误码： {0}", result);
        }
    }
    else
    {
        Console.Error.WriteLine("创建上下文句柄失败！错误码： {0}", resul
t);
    }
}
else
{
    Console.Error.WriteLine("机械臂初始化失败！错误码： {0}", result);
}
}
```

4.2 用回调函数的方式来获取实时信息

4.2.1 获取实时路点信息

本案例是用回调函数的方式来获取实时路点信息。

代码如下：

```
static void realtimeRoadPointExample(UInt16 rshd)
{
    int ret = AuboRobot.rs_enable_push_realtime_roadpoint(rshd, true);
    if (ret == RS_SUCC)
    {
        AuboRobot.rs_setcallback_realtime_roadpoint(rshd, AuboRobot.CurrentPositionCallback, IntPtr.Zero);
        Thread.Sleep(5*1000);
    }
    else
    {
        Console.Error.WriteLine("调用 rs_enable_push_realtime_roadpoint 函数失败！错误码：{0}", ret);
    }
}
```

4.2.2 获取实时末端速度信息

本案例是用回调函数的方式来获取实时末端速度信息。

代码如下：

```
static void realtimeEndSpeedExample(UInt16 rshd)
{
    AuboRobot.rs_setcallback_realtime_end_speed(rshd, AuboRobot.CurrentEndSpeedCallback, IntPtr.Zero);
    Thread.Sleep(5*1000);
}
```

4.2.3 获取机械臂的事件信息

本案例是用回调函数的方式来获取机械臂的事件信息。

代码如下：

```
static void realtimeEventInfoExample(UInt16 rshd)
{
    AuboRobot.rs_setcallback_robot_event(rshd, AuboRobot.RobotEventCallback, IntPtr.Zero);
    Thread.Sleep(10*1000);
}
```

4.2.4 获取关节状态信息

本案例是用回调函数的方式来获取机械臂的关节状态信息。

代码如下：

```
static void realtimeJointStatusExample(UInt16 rshd)
{
    AuboRobot.rs_setcallback_realtime_joint_status(rshd, AuboRobot.CurrentJointStatusCallback, IntPtr.Zero);
    Thread.Sleep(5*1000);
}
```

4.3 正逆解

4.3.1 正解

本案例是用 SDK 来实现机器人运动学正解的功能。

代码如下：

```
static void forwardExample(UInt16 rshd)
{
    AuboRobot.wayPoint_S waypoint = new AuboRobot.wayPoint_S();

    //关节角
    double[] joint =
    {
        3.028726 / 180 * M_PI,
        -5.050340 / 180 * M_PI,
        -52.054678 / 180 * M_PI,
        14.933415 / 180 * M_PI,
        -61.892176 / 180 * M_PI,
        5.135716 / 180 * M_PI
    };

    //正解
    AuboRobot.rs_forward_kin(rshd, joint, ref waypoint);

    //打印目标路点信息
    AuboRobot.PrintWaypoint(waypoint);
}
```

4.3.2 逆解

本案例是用 SDK 来实现机器人运动学逆解的功能。

代码如下：

```
static void inverseExample(UInt16 rshd)
{
    AuboRobot.wayPoint_S waypoint = new AuboRobot.wayPoint_S();
    AuboRobot.Pos pos = new AuboRobot.Pos();
    AuboRobot.Rpy rpy = new AuboRobot.Rpy();
    AuboRobot.Ori ori = new AuboRobot.Ori();
}
```

```

//位置
pos.x = -0.359272;
pos.y = -0.185028;
pos.z = 0.736379;

//姿态（欧拉角）
rpy.rx = 151.007202 * M_PI / 180;
rpy.ry = -27.141245 * M_PI / 180;
rpy.rz = -77.805367 * M_PI / 180;

//欧拉角转四元数
AuboRobot.rs_rpy_to_quaternion(rshd, ref rpy, ref ori);

//参考关节角
double[] joint = { };

//逆解
AuboRobot.rs_inverse_kin(rshd, joint, ref pos, ref ori, ref waypoint);

//打印目标路点信息
AuboRobot.PrintWaypoint(waypoint);
}

```

4.4 坐标系转换

4.4.1 基坐标系转用户坐标系 rs_base_to_user()

rs_base_to_user 函数示例 1

本示例是将法兰盘中心在基坐标系下的位置和姿态转成工具在用户坐标系下的位置和姿态。

代码如下：

```

static void base2UserExample1(UInt16 rshd)
{
    //法兰盘中心在基坐标系下的位置
    AuboRobot.Pos pos_onbase = new AuboRobot.Pos();
    pos_onbase.x = -0.161831;
    pos_onbase.y = -0.209027;
    pos_onbase.z = 0.950235;
}

```

```

//法兰盘中心在基坐标系下的姿态（欧拉角）
AuboRobot.Rpy rpy_onbase = new AuboRobot.Rpy();
rpy_onbase.rx = 98.142464 / 180 * M_PI;
rpy_onbase.ry = -19.882261 / 180 * M_PI;
rpy_onbase.rz = -22.816816 / 180 * M_PI;

//法兰盘中心在基坐标系下的姿态（四元数）
AuboRobot.Ori ori_onbase = new AuboRobot.Ori();
AuboRobot.rs_rpy_to_quaternion(rshd, ref rpy_onbase, ref ori_onbase);

//用户坐标系
AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord,
    typeof(AuboRobot.CoordCalibrate));

user_coord.coordType = 2;
user_coord.methods = 0;
user_coord.jointPara[0].jointRadian[0] = -14.717415 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[1] = -9.423585 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[2] = -74.757117 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[3] = 22.657165 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[4] = -84.449238 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[5] = -14.946778 * M_PI / 180.0;

user_coord.jointPara[1].jointRadian[0] = 24.363602 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[1] = 7.233866 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[2] = -55.242389 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[3] = 22.464115 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[4] = -86.957003 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[5] = 24.171302 * M_PI / 180.0;

user_coord.jointPara[2].jointRadian[0] = -12.492231 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[1] = 0.324244 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[2] = -65.331736 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[3] = 22.120198 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[4] = -84.531432 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[5] = -12.712789 * M_PI / 180.0;

//坐标系的工具参数
AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
    );

```

```

tooluser_coord.cartPos.x = 0.1;
tooluser_coord.cartPos.y = 0.1;
tooluser_coord.cartPos.z = 0.2;
tooluser_coord.orientation.w = 1;
tooluser_coord.orientation.x = 0;
tooluser_coord.orientation.y = 0;
tooluser_coord.orientation.z = 0;

user_coord.toolDesc = tooluser_coord;

//工具描述
AuboRobot.ToolInEndDesc tool_desc = new AuboRobot.ToolInEndDesc();
tool_desc.cartPos.x = 0.2;
tool_desc.cartPos.y = 0.1;
tool_desc.cartPos.z = 0.1;
tool_desc.orientation.w = 1;
tool_desc.orientation.x = 0;
tool_desc.orientation.y = 0;
tool_desc.orientation.z = 0;

//工具在用户坐标系下的位置
AuboRobot.Pos pos_onuser = new AuboRobot.Pos();

//工具在用户坐标系下的姿态
AuboRobot.Ori ori_onuser = new AuboRobot.Ori();

//坐标系转换
AuboRobot.rs_base_to_user(rshd, ref pos_onbase, ref ori_onbase, ref user_coord, ref tool_desc, ref pos_onuser, ref ori_onuser);

Console.Out.WriteLine("工具在用户坐标系下的位置：（{0}，{1}，{2}）", pos_onuser.x, pos_onuser.y, pos_onuser.z);
Console.Out.WriteLine("工具在用户坐标系下的姿态（四元数）：（{0}，{1}，{2}，{3}）", ori_onuser.w, ori_onuser.x, ori_onuser.y, ori_onuser.z);
AuboRobot.Rpy rpy_onuser = new AuboRobot.Rpy();
AuboRobot.rs_quaternion_to_rpy(rshd, ref ori_onuser, ref rpy_onuser);
Console.Out.WriteLine("工具在用户坐标系下的姿态（欧拉角）：（{0}，{1}，{2}）", rpy_onuser.rx * 180 / M_PI, rpy_onuser.ry * 180 / M_PI, rpy_onuser.rz * 180 / M_PI);
}

```

rs_base_to_user 函数示例 2

本示例是将法兰盘中心在基坐标系下的位置和姿态转成法兰盘中心在用户坐标系下的位置和姿态。法兰盘中心可看成是一个位置(0,0,0)姿态(1,0,0,0)的特殊工具。

代码如下：

```
static void base2UserExample2(UInt16 rshd)
{
    //法兰盘中心在基坐标系下的位置
    AuboRobot.Pos pos_onbase = new AuboRobot.Pos();
    pos_onbase.x = -0.161831;
    pos_onbase.y = -0.209027;
    pos_onbase.z = 0.950235;

    //法兰盘中心在基坐标系下的姿态（欧拉角）
    AuboRobot.Rpy rpy_onbase = new AuboRobot.Rpy();
    rpy_onbase.rx = 98.142464 / 180 * M_PI;
    rpy_onbase.ry = -19.882261 / 180 * M_PI;
    rpy_onbase.rz = -22.816816 / 180 * M_PI;

    //法兰盘中心在基坐标系下的姿态（四元数）
    AuboRobot.Ori ori_onbase = new AuboRobot.Ori();
    AuboRobot.rs_rpy_to_quaternion(rshd, ref rpy_onbase, ref ori_onbase);

    //用户坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 2;
    user_coord.methods = 0;
    user_coord.jointPara[0].jointRadian[0] = -14.717415 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[1] = -9.423585 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[2] = -74.757117 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[3] = 22.657165 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[4] = -84.449238 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[5] = -14.946778 * M_PI / 180.0;

    user_coord.jointPara[1].jointRadian[0] = 24.363602 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[1] = 7.233866 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[2] = -55.242389 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[3] = 22.464115 * M_PI / 180.0;
```



```

user_coord.jointPara[1].jointRadian[4] = -86.957003 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[5] = 24.171302 * M_PI / 180.0;

user_coord.jointPara[2].jointRadian[0] = -12.492231 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[1] = 0.324244 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[2] = -65.331736 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[3] = 22.120198 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[4] = -84.531432 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[5] = -12.712789 * M_PI / 180.0;

//坐标系的工具参数
AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
());
tooluser_coord.cartPos.x = 0.1;
tooluser_coord.cartPos.y = 0.1;
tooluser_coord.cartPos.z = 0.2;
tooluser_coord.orientation.w = 1;
tooluser_coord.orientation.x = 0;
tooluser_coord.orientation.y = 0;
tooluser_coord.orientation.z = 0;

user_coord.toolDesc = tooluser_coord;

//工具描述
AuboRobot.ToolInEndDesc tool_desc = new AuboRobot.ToolInEndDesc();
tool_desc.cartPos.x = 0;
tool_desc.cartPos.y = 0;
tool_desc.cartPos.z = 0;
tool_desc.orientation.w = 1;
tool_desc.orientation.x = 0;
tool_desc.orientation.y = 0;
tool_desc.orientation.z = 0;

//法兰盘中心在用户坐标系下的位置
AuboRobot.Pos pos_onuser = new AuboRobot.Pos();

//法兰盘中心在用户坐标系下的姿态
AuboRobot.Ori ori_onuser = new AuboRobot.Ori();

//坐标系转换
AuboRobot.rs_base_to_user(rshd, ref pos_onbase, ref ori_onbase, ref user_coord, ref tool_desc, ref pos_onuser, ref ori_onuser);

Console.Out.WriteLine("法兰盘中心在用户坐标系下的位置:  {0}, {1},

```

```
{2}) ", pos_onuser.x, pos_onuser.y, pos_onuser.z);  
    Console.Out.WriteLine("法兰盘中心在用户坐标系下的姿态（四元数）：({0},  
{1}, {2}, {3})", ori_onuser.w, ori_onuser.x, ori_onuser.y, ori_onuser.z);  
    AuboRobot.Rpy rpy_onuser = new AuboRobot.Rpy();  
    AuboRobot.rs_quaternion_to_rpy(rshd, ref ori_onuser, ref rpy_onuser);  
    Console.Out.WriteLine("法兰盘中心在用户坐标系下的姿态（欧拉角）：  
( {0}, {1}, {2} ) ", rpy_onuser.rx * 180 / M_PI, rpy_onuser.ry * 180 / M_PI,  
rpy_onuser.rz * 180 / M_PI);  
}
```

rs_base_to_user 函数示例 3

本示例是将法兰盘中心在基坐标系下的位置和姿态转成工具在基坐标系下的位置和姿态。

代码如下：

```
static void base2UserExample3(UInt16 rshd)
{
    //法兰盘中心在基坐标系下的位置
    AuboRobot.Pos pos_onbase = new AuboRobot.Pos();
    pos_onbase.x = -0.161831;
    pos_onbase.y = -0.209027;
    pos_onbase.z = 0.950235;

    //法兰盘中心在基坐标系下的姿态（欧拉角）
    AuboRobot.Rpy rpy_onbase = new AuboRobot.Rpy();
    rpy_onbase.rx = 98.142464 / 180 * M_PI;
    rpy_onbase.ry = -19.882261 / 180 * M_PI;
    rpy_onbase.rz = -22.816816 / 180 * M_PI;

    //法兰盘中心在基坐标系下的姿态（四元数）
    AuboRobot.Ori ori_onbase = new AuboRobot.Ori();
    AuboRobot.rs_rpy_to_quaternion(rshd, ref rpy_onbase, ref ori_onbase);

    //基坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 0;

    //工具描述
    AuboRobot.ToolInEndDesc tool_desc = new AuboRobot.ToolInEndDesc();
    tool_desc.cartPos.x = 0.2;
    tool_desc.cartPos.y = 0.1;
    tool_desc.cartPos.z = 0.1;
    tool_desc.orientation.w = 1;
    tool_desc.orientation.x = 0;
    tool_desc.orientation.y = 0;
    tool_desc.orientation.z = 0;

    //工具在基坐标系下的位置
```

```

AuboRobot.Pos tool_pos_onbase = new AuboRobot.Pos();

//工具在基坐标系下的姿态
AuboRobot.Ori tool_ori_onbase = new AuboRobot.Ori();

//坐标系转换
AuboRobot.rs_base_to_user(rshd, ref pos_onbase, ref ori_onbase, ref user_coord, ref tool_desc, ref tool_pos_onbase, ref tool_ori_onbase);

Console.Out.WriteLine("工具在基坐标系下的位置：（{0}，{1}，{2}）", tool_pos_onbase.x, tool_pos_onbase.y, tool_pos_onbase.z);
Console.Out.WriteLine("工具在基坐标系下的姿态（四元数）：（{0}，{1}，{2}，{3}）", tool_ori_onbase.w, tool_ori_onbase.x, tool_ori_onbase.y, tool_ori_onbase.z);
AuboRobot.Rpy tool_rpy_onbase = new AuboRobot.Rpy();
AuboRobot.rs_quaternion_to_rpy(rshd, ref tool_ori_onbase, ref tool_rpy_onbase);
Console.Out.WriteLine("工具在基坐标系下的姿态（欧拉角）：（{0}，{1}，{2}）", tool_rpy_onbase.rx * 180 / M_PI, tool_rpy_onbase.ry * 180 / M_PI, tool_rpy_onbase.rz * 180 / M_PI);
}

```

4.4.2 基坐标系转基坐标系 rs_base_to_base_additional_tool()

rs_base_to_base_additional_tool 函数示例

本示例是将法兰盘中心在基坐标系下的位置和姿态转成工具在基坐标系下的位置和姿态。

代码如下：

```

static void base2BaseExample(UInt16 rshd)
{
    //法兰盘中心在基坐标系下的位置
    AuboRobot.Pos flange_pos_onbase = new AuboRobot.Pos();
    flange_pos_onbase.x = -0.161831;
    flange_pos_onbase.y = -0.209027;
    flange_pos_onbase.z = 0.950235;

    //法兰盘中心在基坐标系下的姿态（欧拉角）
    AuboRobot.Rpy flange_rpy_onbase = new AuboRobot.Rpy();
    flange_rpy_onbase.rx = 98.142464 / 180 * M_PI;
    flange_rpy_onbase.ry = -19.882261 / 180 * M_PI;
}

```

```

flange_rpy_onbase.rz = -22.816816 / 180 * M_PI;

//法兰盘中心在基坐标系下的姿态（四元数）
AuboRobot.Ori flange_ori_onbase = new AuboRobot.Ori();
AuboRobot.rs_rpy_to_quaternion(rshd, ref flange_rpy_onbase, ref flange_ori_o
nbase);

//工具描述
AuboRobot.ToolInEndDesc tool_desc = new AuboRobot.ToolInEndDesc();
tool_desc.cartPos.x = 0.2;
tool_desc.cartPos.y = 0.1;
tool_desc.cartPos.z = 0.1;
tool_desc.orientation.w = 1;
tool_desc.orientation.x = 0;
tool_desc.orientation.y = 0;
tool_desc.orientation.z = 0;

//工具在基坐标系下的位置
AuboRobot.Pos tool_pos_onbase = new AuboRobot.Pos();

//工具在基坐标系下的姿态
AuboRobot.Ori tool_ori_onbase = new AuboRobot.Ori();

//坐标系转换
AuboRobot.rs_base_to_base_additional_tool(rshd, ref flange_pos_onbase, ref fl
ange_ori_onbase, ref tool_desc, ref tool_pos_onbase, ref tool_ori_onbase);

Console.Out.WriteLine("工具在基坐标系下的位置：（{0}，{1}，{2}）", tool
_pos_onbase.x, tool_pos_onbase.y, tool_pos_onbase.z);
Console.Out.WriteLine("工具在基坐标系下的姿态（四元数）：（{0}，{1}，
{2}，{3}）", tool_ori_onbase.w, tool_ori_onbase.x, tool_ori_onbase.y, tool_ori_onbas
e.z);
AuboRobot.Rpy tool_rpy_onbase = new AuboRobot.Rpy();
AuboRobot.rs_quaternion_to_rpy(rshd, ref tool_ori_onbase, ref tool_rpy_onbas
e);
Console.Out.WriteLine("工具在基坐标系下的姿态（欧拉角）：（{0}，{1}，
{2}）", tool_rpy_onbase.rx * 180 / M_PI, tool_rpy_onbase.ry * 180 / M_PI, too
l_rpy_onbase.rz * 180 / M_PI);
}

```

4.4.3 用户坐标系转基坐标系 rs_user_to_base()

rs_user_to_base 函数示例 1

本示例是将工具末端在用户坐标系下的位置和姿态转成法兰盘中心在基坐标系下的位置和姿态。

代码如下：

```
static void user2BaseExample1(UInt16 rshd)
{
    //工具在用户坐标系下的位置
    AuboRobot.Pos pos_onuser = new AuboRobot.Pos();
    pos_onuser.x = 0.2917;
    pos_onuser.y = -0.4095;
    pos_onuser.z = -0.7179;

    //工具在用户坐标系下的姿态（欧拉角）
    AuboRobot.Rpy rpy_onuser = new AuboRobot.Rpy();
    rpy_onuser.rx = -75.6677 * M_PI / 180.0;
    rpy_onuser.ry = 20.9749 * M_PI / 180.0;
    rpy_onuser.rz = -64.6939 * M_PI / 180.0;

    //工具在用户坐标系下的姿态（四元数）
    AuboRobot.Ori ori_onuser = new AuboRobot.Ori();
    AuboRobot.rs_rpy_to_quaternion(rshd, ref rpy_onuser, ref ori_onuser);

    //用户坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 2;
    user_coord.methods = 0;
    user_coord.jointPara[0].jointRadian[0] = -14.717415 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[1] = -9.423585 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[2] = -74.757117 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[3] = 22.657165 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[4] = -84.449238 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[5] = -14.946778 * M_PI / 180.0;

    user_coord.jointPara[1].jointRadian[0] = 24.363602 * M_PI / 180.0;
```

```

user_coord.jointPara[1].jointRadian[1] = 7.233866 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[2] = -55.242389 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[3] = 22.464115 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[4] = -86.957003 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[5] = 24.171302 * M_PI / 180.0;

user_coord.jointPara[2].jointRadian[0] = -12.492231 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[1] = 0.324244 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[2] = -65.331736 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[3] = 22.120198 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[4] = -84.531432 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[5] = -12.712789 * M_PI / 180.0;

//坐标系的工具参数
AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
());
tooluser_coord.cartPos.x = 0.1;
tooluser_coord.cartPos.y = 0.1;
tooluser_coord.cartPos.z = 0.2;
tooluser_coord.orientation.w = 1;
tooluser_coord.orientation.x = 0;
tooluser_coord.orientation.y = 0;
tooluser_coord.orientation.z = 0;

user_coord.toolDesc = tooluser_coord;

//工具描述
AuboRobot.ToolInEndDesc tool_desc = new AuboRobot.ToolInEndDesc();
tool_desc.cartPos.x = 0.2;
tool_desc.cartPos.y = 0.1;
tool_desc.cartPos.z = 0.1;
tool_desc.orientation.w = 1;
tool_desc.orientation.x = 0;
tool_desc.orientation.y = 0;
tool_desc.orientation.z = 0;

//法兰盘中心在基坐标系下的位置
AuboRobot.Pos pos_onbase = new AuboRobot.Pos();

//法兰盘中心在基坐标系下的姿态
AuboRobot.Ori ori_onbase = new AuboRobot.Ori();

//坐标系转换
AuboRobot.rs_user_to_base(rshd, ref pos_onuser, ref ori_onuser, ref user_coord

```

```

d, ref tool_desc, ref pos_onbase, ref ori_onbase);

    Console.Out.WriteLine("法兰盘中心在基坐标系下的位置: ({0}, {1}, {2})", pos_onbase.x, pos_onbase.y, pos_onbase.z);
    Console.Out.WriteLine("法兰盘中心在基坐标系下的姿态 (四元数): ({0}, {1}, {2}, {3})", ori_onbase.w, ori_onbase.x, ori_onbase.y, ori_onbase.z);
    AuboRobot.Rpy rpy_onbase = new AuboRobot.Rpy();
    AuboRobot.rs_quaternion_to_rpy(rshd, ref ori_onbase, ref rpy_onbase);
    Console.Out.WriteLine("法兰盘中心在基坐标系下的姿态 (欧拉角): ({0}, {1}, {2})", rpy_onbase.rx * 180 / M_PI, rpy_onbase.ry * 180 / M_PI, rpy_onbase.rz * 180 / M_PI);
}

```

rs_user_to_base 函数示例 2

本示例是将法兰盘中心在用户坐标系下的位置和姿态转成法兰盘中心在基坐标系下的位置和姿态。

代码如下:

```

static void user2BaseExample2(UInt16 rshd)
{
    //法兰盘中心在用户坐标系下的位置
    AuboRobot.Pos pos_onuser = new AuboRobot.Pos();
    pos_onuser.x = 0.112965;
    pos_onuser.y = -0.315997;
    pos_onuser.z = -0.578977;

    //法兰盘中心在用户坐标系下的姿态 (欧拉角)
    AuboRobot.Rpy rpy_onuser = new AuboRobot.Rpy();
    rpy_onuser.rx = -75.667717 * M_PI / 180.0;
    rpy_onuser.ry = 20.974926 * M_PI / 180.0;
    rpy_onuser.rz = -64.693947 * M_PI / 180.0;

    //法兰盘中心在用户坐标系下的姿态 (四元数)
    AuboRobot.Ori ori_onuser = new AuboRobot.Ori();
    AuboRobot.rs_rpy_to_quaternion(rshd, ref rpy_onuser, ref ori_onuser);

    //用户坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord

```



```

d, typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 2;
    user_coord.methods = 0;
    user_coord.jointPara[0].jointRadian[0] = -14.717415 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[1] = -9.423585 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[2] = -74.757117 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[3] = 22.657165 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[4] = -84.449238 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[5] = -14.946778 * M_PI / 180.0;

    user_coord.jointPara[1].jointRadian[0] = 24.363602 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[1] = 7.233866 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[2] = -55.242389 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[3] = 22.464115 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[4] = -86.957003 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[5] = 24.171302 * M_PI / 180.0;

    user_coord.jointPara[2].jointRadian[0] = -12.492231 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[1] = 0.324244 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[2] = -65.331736 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[3] = 22.120198 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[4] = -84.531432 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[5] = -12.712789 * M_PI / 180.0;

    //坐标系的工具参数
    AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
    ();
    tooluser_coord.cartPos.x = 0.1;
    tooluser_coord.cartPos.y = 0.1;
    tooluser_coord.cartPos.z = 0.2;
    tooluser_coord.orientation.w = 1;
    tooluser_coord.orientation.x = 0;
    tooluser_coord.orientation.y = 0;
    tooluser_coord.orientation.z = 0;

    user_coord.toolDesc = tooluser_coord;

    //工具描述
    AuboRobot.ToolInEndDesc tool_desc = new AuboRobot.ToolInEndDesc();
    tool_desc.cartPos.x = 0;
    tool_desc.cartPos.y = 0;
    tool_desc.cartPos.z = 0;
    tool_desc.orientation.w = 1;
    tool_desc.orientation.x = 0;

```

```
tool_desc.orientation.y = 0;
tool_desc.orientation.z = 0;

//法兰盘中心在基坐标系下的位置
AuboRobot.Pos pos_onbase = new AuboRobot.Pos();

//法兰盘中心在基坐标系下的姿态
AuboRobot.Ori ori_onbase = new AuboRobot.Ori();

//坐标系转换
AuboRobot.rs_user_to_base(rshd, ref pos_onuser, ref ori_onuser, ref user_coord, ref tool_desc, ref pos_onbase, ref ori_onbase);

Console.Out.WriteLine("法兰盘中心在基坐标系下的位置：（{0}，{1}，{2}）", pos_onbase.x, pos_onbase.y, pos_onbase.z);
Console.Out.WriteLine("法兰盘中心在基坐标系下的姿态（四元数）：（{0}，{1}，{2}，{3}）", ori_onbase.w, ori_onbase.x, ori_onbase.y, ori_onbase.z);
AuboRobot.Rpy rpy_onbase = new AuboRobot.Rpy();
AuboRobot.rs_quaternion_to_rpy(rshd, ref ori_onbase, ref rpy_onbase);
Console.Out.WriteLine("法兰盘中心在基坐标系下的姿态（欧拉角）：（{0}，{1}，{2}）", rpy_onbase.rx / M_PI * 180.0, rpy_onbase.ry / M_PI * 180.0, rpy_onbase.rz / M_PI * 180.0);
}
```

rs_user_to_base 函数示例 3

本示例是将工具末端在基坐标系下的位置和姿态转成法兰盘中心在基坐标系下的位置和姿态。

代码如下：

```
static void user2BaseExample3(UInt16 rshd)
{
    //工具在基坐标系下的位置
    AuboRobot.Pos tool_pos_onbase = new AuboRobot.Pos();
    tool_pos_onbase.x = -0.058941;
    tool_pos_onbase.y = -0.375075;
    tool_pos_onbase.z = 1.098025;

    //工具在基坐标系下的姿态（欧拉角）
    AuboRobot.Rpy tool_rpy_onbase = new AuboRobot.Rpy();
    tool_rpy_onbase.rx = 98.142464 / 180 * M_PI;
    tool_rpy_onbase.ry = -19.882261 / 180 * M_PI;
    tool_rpy_onbase.rz = -22.816816 / 180 * M_PI;

    //工具在基坐标系下的姿态（四元数）
    AuboRobot.Ori tool_ori_onbase = new AuboRobot.Ori();
    AuboRobot.rs_rpy_to_quaternion(rshd, ref tool_rpy_onbase, ref tool_ori_onbase);

    //基坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 0;

    //工具描述
    AuboRobot.ToolInEndDesc tool_desc = new AuboRobot.ToolInEndDesc();
    tool_desc.cartPos.x = 0.2;
    tool_desc.cartPos.y = 0.1;
    tool_desc.cartPos.z = 0.1;
    tool_desc.orientation.w = 1;
    tool_desc.orientation.x = 0;
    tool_desc.orientation.y = 0;
    tool_desc.orientation.z = 0;
}
```

```

//法兰盘中心在基坐标系下的位置
AuboRobot.Pos flange_pos_onbase = new AuboRobot.Pos();

//法兰盘中心在基坐标系下的姿态
AuboRobot.Ori flange_ori_onbase = new AuboRobot.Ori();

//坐标系转换
AuboRobot.rs_user_to_base(rshd, ref tool_pos_onbase, ref tool_ori_onbase, ref
user_coord, ref tool_desc, ref flange_pos_onbase, ref flange_ori_onbase);

Console.Out.WriteLine("法兰盘中心在基坐标系下的位置：（{0}，{1}，
{2}）", flange_pos_onbase.x, flange_pos_onbase.y, flange_pos_onbase.z);
Console.Out.WriteLine("法兰盘中心在基坐标系下的姿态（四元数）：（{0}，
{1}，{2}，{3}）", flange_ori_onbase.w, flange_ori_onbase.x, flange_ori_onbase.y, fla
nge_ori_onbase.z);
AuboRobot.Rpy flange_rpy_onbase = new AuboRobot.Rpy();
AuboRobot.rs_quaternion_to_rpy(rshd, ref flange_ori_onbase, ref flange_rpy_o
nbase);
Console.Out.WriteLine("法兰盘中心在基坐标系下的姿态（欧拉角）：
（{0}，{1}，{2}）", flange_rpy_onbase.rx * 180 / M_PI, flange_rpy_onbase.ry *
180 / M_PI, flange_rpy_onbase.rz * 180 / M_PI);
}

```

4.5 设置和获取机械臂相关参数

4.5.1 获取当前路点信息

本示例是获取机械臂当前路点信息。

代码如下：

```

static void getCurrentWaypoint(UInt16 rshd)
{
    AuboRobot.wayPoint_S waypoint = new AuboRobot.wayPoint_S();

    int ret = AuboRobot.rs_get_current_waypoint(rshd, ref waypoint);
    if(ret == RS_SUCC)
    {
        Console.Out.WriteLine("获取当前路点信息成功！");
        AuboRobot.PrintWaypoint(waypoint);
    }
    else

```

```

    {
        Console.Error.WriteLine("获取当前路点信息失败！错误码：{0}", ret);
    }
}

```

4.5.2 获取关节状态

本示例是获取机械臂当前的关节状态。

代码如下：

```

static void getJointStatus(UInt16 rshd)
{
    int size = Marshal.SizeOf(typeof(AuboRobot.JointStatus)) * 6;
    byte[] bytes = new byte[size];
    IntPtr pBuff = Marshal.AllocHGlobal(size);
    AuboRobot.JointStatus[] jointStatus = new AuboRobot.JointStatus[6];
    int ret = AuboRobot.rs_get_joint_status(rshd, pBuff);
    if(ret == RS_SUCC)
    {
        Console.Out.WriteLine("获取关节状态成功！\n");
        for (int i = 0; i < 6; i++)
        {
            IntPtr pPonitor = new IntPtr(pBuff.ToInt64() + Marshal.SizeOf(type
of(AuboRobot.JointStatus)) * i);
            jointStatus[i] = (AuboRobot.JointStatus)Marshal.PtrToStructure(pPonit
or, typeof(AuboRobot.JointStatus));
        }
        for (int i = 0; i < 6; i++)
        {
            Console.Out.WriteLine("关节{0}:", i + 1);
            Console.Out.WriteLine("关节电流：{0} 关节速度：{1} 关节角：{2}
关节电压：{3} 当前温度：{4} ", jointStatus[i].jointCurrentI, jointStatus[i].jointSpe
edMoto, jointStatus[i].jointPosJ * 180 / M_PI, jointStatus[i].jointCurVol, jointStatu
s[i].jointCurTemp);
            Console.Out.WriteLine("电机目标电流：{0} 电机目标速度：{1} 目
标关节角：{2} 关节错误码：{3} ", jointStatus[i].jointTagCurrentI, jointStatus[i].joi
ntTagSpeedMoto, jointStatus[i].jointTagPosJ * 180 / M_PI, jointStatus[i].jointError
Num);
        }
        Marshal.FreeHGlobal(pBuff);
    }
    else

```

```

    {
        Console.Error.WriteLine("获取关节状态失败！ 错误码： {0}", ret);
    }
}

```

4.5.3 获取诊断信息

本示例是获取机械臂的诊断信息。

代码如下：

```

static void getDiagnosisExample(UInt16 rshd)
{
    int result = 0xffff;
    AuboRobot.RobotDiagnosis robotDiagnosis = new AuboRobot.RobotDiagnosis
();
    result = AuboRobot.rs_get_diagnosis_info(rshd, ref robotDiagnosis);

    if (result == RS_SUCC)
    {
        Console.Out.WriteLine("\n 获取诊断信息成功！ ");
        Console.Out.WriteLine("-----诊断信息-----");
        Console.Out.WriteLine("CAN 通信状态:{0}", robotDiagnosis.armCanbusSt
atus);
        Console.Out.WriteLine("当前电流:{0}", robotDiagnosis.armPowerCurrent);
        Console.Out.WriteLine("当前电压:{0}", robotDiagnosis.armPowerVoltage);
        Console.Out.WriteLine("电源状态（开、关）:{0}", robotDiagnosis.armPo
werStatus);
        Console.Out.WriteLine("控制箱温度:{0}", robotDiagnosis.contorllerTemp);
        Console.Out.WriteLine("控制箱湿度:{0}", robotDiagnosis.contorllerHumidi
ty);
        Console.Out.WriteLine("远程关机信号:{0}", robotDiagnosis.remoteHalt);
        Console.Out.WriteLine("机械臂软急停:{0}", robotDiagnosis.softEmergenc
y);
        Console.Out.WriteLine("远程急停信号:{0}", robotDiagnosis.remoteEmergenc
y);
        Console.Out.WriteLine("碰撞检测位:{0}", robotDiagnosis.robotCollision);
        Console.Out.WriteLine("机械臂进入力控模式标志位:{0}", robotDiagnosis.
forceControlMode);
        Console.Out.WriteLine("刹车状态:{0}", robotDiagnosis.brakeStuats);
        Console.Out.WriteLine("末端速度:{0}", robotDiagnosis.robotEndSpeed);
        Console.Out.WriteLine("最大加速度:{0}", robotDiagnosis.robotMaxAcc);
        Console.Out.WriteLine("上位机软件状态位:{0}", robotDiagnosis.orpeStatu
s);
    }
}

```

```

        Console.Out.WriteLine("位姿读取使能位:{0}", robotDiagnosis.enableRead
Pose);
        Console.Out.WriteLine("安装位置状态:{0}", robotDiagnosis.robotMounting
PoseChanged);
        Console.Out.WriteLine("磁编码器错误状态:{0}", robotDiagnosis.encoderE
rrorStatus);
        Console.Out.WriteLine("静止碰撞检测开关:{0}", robotDiagnosis.staticColl
isionDetect);
        Console.Out.WriteLine("关节碰撞检测:{0}", robotDiagnosis.jointCollision
Detect);
        Console.Out.WriteLine("光电编码器不一致错误:{0}", robotDiagnosis.enco
derLinesError);
        Console.Out.WriteLine("关节错误状态:{0}", robotDiagnosis.jointErrorStatu
s);
        Console.Out.WriteLine("机械臂奇异点过速警告:{0}", robotDiagnosis.sing
ularityOverSpeedAlarm);
        Console.Out.WriteLine("机械臂电流错误警告:{0}", robotDiagnosis.robotC
urrentAlarm);
        Console.Out.WriteLine("工具 IO 错误:{0}", robotDiagnosis.toolIoError);
        Console.Out.WriteLine("机械臂安装位置错位（只在力控模式下起作用）:
{0}", robotDiagnosis.robotMountingPoseWarning);
        Console.Out.WriteLine("mac 缓冲器长度:{0}", robotDiagnosis.macTargetP
osBufferSize);
        Console.Out.WriteLine("mac 缓冲器有效数据长度:{0}", robotDiagnosis.m
acTargetPosDataSize);
        Console.Out.WriteLine("mac 数据中断 :{0}", robotDiagnosis.macDataInter
ruptWarning);
        Console.Out.WriteLine("主控板(接口板)异常状态标志:{0}", robotDiagnosi
s.controlBoardAbnormalStateFlag);

    }
    else
    {
        Console.Error.WriteLine("获取诊断信息失败！ 错误码: {0}", result);
    }
}

```

4.5.4 获取设备信息

本示例是获取机械臂的设备信息。

代码如下：

```
static void getDeviceExample(UInt16 rshd)
{
    int result = 0xffff;
    AuboRobot.RobotDevInfo robotDevice = new AuboRobot.RobotDevInfo();
    result = AuboRobot.rs_get_device_info(rshd, ref robotDevice);
    if (result == RS_SUCC)
    {
        Console.Out.WriteLine("\n 获取设备信息成功! ");
        Console.Out.WriteLine("-----设备信息-----");
        string revision = new string(robotDevice.revision);
        Console.Out.WriteLine("Revision: {0}", revision);
        string slave_version = new string(robotDevice.slave_version);
        Console.Out.WriteLine("Slave Version: {0}", slave_version);
        string extio_version = new string(robotDevice.extio_version);
        Console.Out.WriteLine("Extend IO Version: {0}", extio_version);
        string manu_id = new string(robotDevice.manu_id);
        Console.Out.WriteLine("ManuId: {0}", manu_id);
        string joint_type = new string(robotDevice.joint_type);
        Console.Out.WriteLine("Joint Type: {0}", joint_type);
        for(int i = 0; i < 8; i++)
        {
            if(i == 6)
            {
                string hw_version = new string(robotDevice.joint_ver[i].hw_veri
on);
                Console.Out.WriteLine("Tool hardware version: {0}", hw_veri
on);
                string sw_version = new string(robotDevice.joint_ver[i].sw_veri
on);
                Console.Out.WriteLine("Tool software version: {0}", sw_versi
on);
            }
            else if(i == 7)
            {
                string hw_version = new string(robotDevice.joint_ver[i].hw_veri
on);
                Console.Out.WriteLine("Base hardware version: {0}", hw_veri
on);
                string sw_version = new string(robotDevice.joint_ver[i].sw_veri
```



```

on);

        Console.Out.WriteLine("Base software version: {0}", sw_version);
n);
    }
    else
    {

        string hw_version = new string(robotDevice.joint_ver[i].hw_version);
on);

        Console.Out.WriteLine("Joint({0}) hardware version: {1}", i +
1, hw_version);
        string sw_version = new string(robotDevice.joint_ver[i].sw_version);
on);

        Console.Out.WriteLine("Joint({0}) software version: {1}", i +
1, sw_version);
    }
}
for (int i = 0; i < 8; i++)
{
    if (i == 0)
    {
        string productID = new string(robotDevice.jointProductID[i].productID);
uctID);

        Console.Out.WriteLine("Interface Board ID: {0}", productID);
    }
    else if (i == 7)
    {

        string productID = new string(robotDevice.jointProductID[i].productID);
uctID);

        Console.Out.WriteLine("Tool ID: {0}", productID);
    }
    else
    {
        string productID = new string(robotDevice.jointProductID[i].productID);
uctID);

        Console.Out.WriteLine("Joint({0}) ID: {1}", i, productID);
    }
}

}
else

```

```
    {  
        Console.Error.WriteLine("获取设备信息失败！ 错误码： {0}", result);  
    }  
}
```

4.5.5 设置和获取工作模式：仿真或真实

本示例是设置和获取机械臂的工作模式。

代码如下：

```
static void workMode(UInt16 rshd)  
{  
    //设置工作模式  
    int state = 0;  
    int ret = AuboRobot.rs_set_work_mode(rshd, state);  
    if(ret == RS_SUCC)  
    {  
        Console.Out.WriteLine("设置工作模式成功！");  
    }  
    else  
    {  
        Console.Error.WriteLine("设置工作模式失败！ 错误码： {0}", ret);  
    }  
    //获取工作模式  
    ret = AuboRobot.rs_get_work_mode(rshd, ref state);  
    if (ret == RS_SUCC)  
    {  
        Console.Out.WriteLine("获取工作模式成功！ 工作模式： {0}", state);  
    }  
    else  
    {  
        Console.Error.WriteLine("获取工作模式失败！ 错误码： {0}", ret);  
    }  
}
```

4.5.6 设置和获取碰撞等级

本示例是设置和获取机械臂的碰撞等级。

代码如下：

```
static void collisionMode(UInt16 rshd)  
{  
    //设置碰撞等级
```

```

int grade = 7;
int ret = AuboRobot.rs_set_collision_class(rshd, grade);
if (ret == RS_SUCC)
{
    Console.Out.WriteLine("设置碰撞等级成功！");
}
else
{
    Console.Error.WriteLine("设置碰撞等级失败！ 错误码： {0}", ret);
}
//获取碰撞等级
ret = AuboRobot.rs_get_collision_class(rshd, ref grade);
if (ret == RS_SUCC)
{
    Console.Out.WriteLine("获取碰撞等级成功！ 碰撞等级： {0}", grade);
}
else
{
    Console.Error.WriteLine("获取碰撞等级失败！ 错误码： {0}", ret);
}
}

```

4.5.7 根据错误码获取错误信息

本示例是根据错误码来获取错误信息。

代码如下：

```

static void getErrorInfo(UInt16 rshd)
{
    int err_code = 10003;
    IntPtr _errorinfo = AuboRobot.rs_get_error_information_by_errcode(rshd, err_code);
    string err_information = Marshal.PtrToStringAnsi(_errorinfo);
    Console.Out.WriteLine("错误信息： {0}", err_information);
}

```

4.5.8 获取 socket 的连接状态

本示例是获取 socket 的连接状态。

代码如下：

```

static void getSocketStatus(UInt16 rshd)
{
    Byte status = 2;
}

```

```
int ret = AuboRobot.rs_get_socket_status(rshd, ref status);
if (ret == RS_SUCC)
{
    Console.Out.WriteLine("获取 socket 连接状态成功! socket 状态: {0}", status);
}
else
{
    Console.Error.WriteLine("获取 socket 连接状态失败! 错误码: {0}", ret);
}
}
```

4.6 IO

4.6.1 工具 IO

本示例是关于工具 IO。

代码如下：

```
//工具端 IO 类型
//
//工具端 DI
const int Robot_Tool_DI = 8;
//工具端 DO
const int Robot_Tool_DO = 9;
//工具端 AI
const int Robot_Tool_AI = 10;
//工具端 AO
const int Robot_Tool_AO = 11;
//工具端 DI
const int Robot_ToolIoType_DI = Robot_Tool_DI;
//工具端 DO
const int Robot_ToolIoType_DO = Robot_Tool_DO;

//工具端 IO 名称
const string TOOL_IO_0 = ("T_DI/O_00");
const string TOOL_IO_1 = ("T_DI/O_01");
const string TOOL_IO_2 = ("T_DI/O_02");
const string TOOL_IO_3 = ("T_DI/O_03");

//工具端数字 IO 类型
//
```

```
//输入
const int TOOL_IO_IN = 0;
//输出
const int TOOL_IO_OUT = 1;

//工具端电源类型
//
const int OUT_0V = 0;
const int OUT_12V = 1;
const int OUT_24V = 2;

//IO 状态-无效
const double IO_STATUS_INVALID = 0.0;
//IO 状态-有效
const double IO_STATUS_VALID = 1.0;
```

```
static void toolIOExample(UInt16 rshd)
{
    int result = 0xffff;

    //设置工具 IO 类型:输入 (0) 或者输出 (1)
    int addr = 3;
    int io_type = 1;
    result = AuboRobot.rs_set_tool_io_type(rshd, addr, io_type);
    if (result == RS_SUCC)
    {
        Console.Out.WriteLine("设置工具 IO 类型成功!");
    }
    else
    {
        Console.Error.WriteLine("设置工具 IO 类型失败! 错误码: {0}", result);
    }

    //设置工具 IO 状态
    string io_name = "T_DI/O_03";
    int val = 1;
    result = AuboRobot.rs_set_tool_do_status(rshd, io_name, val);
    Thread.Sleep(1000);
    if (result == RS_SUCC)
    {
        Console.Out.WriteLine("设置工具 IO 状态成功!");
    }
    else
```

```
{
    Console.Error.WriteLine("设置工具 IO 状态失败！ 错误码： {0}", result);
}

//获取工具 IO 状态
string name = "T_DI/O_03";
double value = -1;
result = AuboRobot.rs_get_tool_io_status(rshd, name, ref value);
if (result == RS_SUCC)
{
    Console.Out.WriteLine("获取工具 IO 状态成功!");
    Console.Out.WriteLine("IO 状态： {0}", value);
}
else
{
    Console.Error.WriteLine("获取工具 IO 状态失败！ 错误码： {0}", result);
}

//设置工具电源类型
int power_type = 1;
result = AuboRobot.rs_set_tool_power_type(rshd, power_type);
Thread.Sleep(1000);
if (result == RS_SUCC)
{
    Console.Out.WriteLine("设置工具 IO 电源类型成功!");
}
else
{
    Console.Error.WriteLine("设置工具 IO 电源类型失败！ 错误码： {0}", result);
}

//获取工具电源类型
int type = -1;
result = AuboRobot.rs_get_tool_power_type(rshd, ref type);
if (result == RS_SUCC)
{
    Console.Out.WriteLine("获取工具 IO 电源类型成功!");
    Console.Out.WriteLine("工具 IO 电源类型： {0}", type);
}
else
{
    Console.Error.WriteLine("获取工具 IO 电源类型失败！ 错误码： {0}", result);
}
```

```
sult);  
    }  
}
```

4.6.2 用户 IO

本示例是关于工具 IO。

代码如下：

```
//接口板用户 DI 地址
const int ROBOT_IO_F1 = 30;
const int ROBOT_IO_F2 = 31;
const int ROBOT_IO_F3 = 32;
const int ROBOT_IO_F4 = 33;
const int ROBOT_IO_F5 = 34;
const int ROBOT_IO_F6 = 35;
const int ROBOT_IO_U_DI_00 = 36;
const int ROBOT_IO_U_DI_01 = 37;
const int ROBOT_IO_U_DI_02 = 38;
const int ROBOT_IO_U_DI_03 = 39;
const int ROBOT_IO_U_DI_04 = 40;
const int ROBOT_IO_U_DI_05 = 41;
const int ROBOT_IO_U_DI_06 = 42;
const int ROBOT_IO_U_DI_07 = 43;
const int ROBOT_IO_U_DI_10 = 44;
const int ROBOT_IO_U_DI_11 = 45;
const int ROBOT_IO_U_DI_12 = 46;
const int ROBOT_IO_U_DI_13 = 47;
const int ROBOT_IO_U_DI_14 = 48;
const int ROBOT_IO_U_DI_15 = 49;
const int ROBOT_IO_U_DI_16 = 50;
const int ROBOT_IO_U_DI_17 = 51;

//接口板用户 DO 地址
const int ROBOT_IO_U_DO_00 = 32;
const int ROBOT_IO_U_DO_01 = 33;
const int ROBOT_IO_U_DO_02 = 34;
const int ROBOT_IO_U_DO_03 = 35;
const int ROBOT_IO_U_DO_04 = 36;
const int ROBOT_IO_U_DO_05 = 37;
const int ROBOT_IO_U_DO_06 = 38;
const int ROBOT_IO_U_DO_07 = 39;
const int ROBOT_IO_U_DO_10 = 40;
const int ROBOT_IO_U_DO_11 = 41;
const int ROBOT_IO_U_DO_12 = 42;
const int ROBOT_IO_U_DO_13 = 43;
const int ROBOT_IO_U_DO_14 = 44;
const int ROBOT_IO_U_DO_15 = 45;
```



```

const int ROBOT_IO_U_DO_16 = 46;
const int ROBOT_IO_U_DO_17 = 47;

//接口板用户 AI 地址
const int ROBOT_IO_VI0 = 0;
const int ROBOT_IO_VI1 = 1;
const int ROBOT_IO_VI2 = 2;
const int ROBOT_IO_VI3 = 3;

//接口板用户 AO 地址
const int ROBOT_IO_VO0 = 0;
const int ROBOT_IO_VO1 = 1;
const int ROBOT_IO_CO0 = 2;
const int ROBOT_IO_CO1 = 3;

//接口板 IO 类型
//
//接口板用户 DI(数字量输入) 可读可写
const int Robot_User_DI = 4;
//接口板用户 DO(数字量输出) 可读可写
const int Robot_User_DO = 5;
//接口板用户 AI(模拟量输入) 可读可写
const int Robot_User_AI = 6;
//接口板用户 AO(模拟量输出) 可读可写
const int Robot_User_AO = 7;

```

```

static void userIOExample(UInt16 rshd)
{
    int result = 0xffff;

    //根据地址设置接口板 IO 状态
    int io_type = 5;
    int addr = 33;
    double val = 1;
    result = AuboRobot.rs_set_board_io_status_by_addr(rshd, io_type, addr, val);
    Thread.Sleep(1000);
    if (result == RS_SUCC)
    {
        Console.Out.WriteLine("设置接口板 IO 状态成功!");
    }
    else
    {
        Console.Error.WriteLine("设置接口板 IO 状态失败！错误码： {0}", resul

```

```
t);
    }

    //根据地址获取接口板 IO 状态
    int type = 5;
    int address = 33;
    double value = -1;
    result = AuboRobot.rs_get_board_io_status_by_addr(rshd, type, address, ref value);
    if (result == RS_SUCC)
    {
        Console.Out.WriteLine("获取接口板 IO 状态成功!");
        Console.Out.WriteLine("IO 状态: {0}", value);
    }
    else
    {
        Console.Error.WriteLine("获取接口板 IO 状态失败! 错误码: {0}", result);
    }
}
```

4.7 关节运动

4.7.1 rs_move_joint 函数

本示例是关节运动到指定关节角。

代码如下：

```
static void jointMoveExample(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 10 * M_PI / 180;
    jointMaxAcc[1] = 10 * M_PI / 180;
    jointMaxAcc[2] = 10 * M_PI / 180;
    jointMaxAcc[3] = 10 * M_PI / 180;
    jointMaxAcc[4] = 10 * M_PI / 180;
    jointMaxAcc[5] = 10 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);
}
```

```
//设置关节运动最大速度
double[] jointMaxVelc = new double[6];
jointMaxVelc[0] = 10 * M_PI / 180;
jointMaxVelc[1] = 10 * M_PI / 180;
jointMaxVelc[2] = 10 * M_PI / 180;
jointMaxVelc[3] = 10 * M_PI / 180;
jointMaxVelc[4] = 10 * M_PI / 180;
jointMaxVelc[5] = 10 * M_PI / 180;
AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

//关节角
double[] initPos = {
    -0.000172 / 180 * M_PI,
    -7.291862 / 180 * M_PI,
    -75.604718 / 180 * M_PI,
    21.596727 / 180 * M_PI,
    -89.999982 / 180 * M_PI,
    -0.00458 / 180 * M_PI
};

//关节运动
int ret = AuboRobot.rs_move_joint(rshd, initPos, true);
if (ret == RS_SUCC)
{
    Console.Out.WriteLine("关节运动成功!");
}
else
{
    Console.Error.WriteLine("关节运动失败!错误码: {0}", ret);
}
}
```

4.7.2 rs_move_joint_to 函数

rs_move_joint_to 函数示例 1

本示例是轴动到法兰盘中心在基坐标系下的位置。

代码如下：

```
static void jointMoveToExample1(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 5 * M_PI / 180;
    jointMaxAcc[1] = 5 * M_PI / 180;
    jointMaxAcc[2] = 5 * M_PI / 180;
    jointMaxAcc[3] = 5 * M_PI / 180;
    jointMaxAcc[4] = 5 * M_PI / 180;
    jointMaxAcc[5] = 5 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 5 * M_PI / 180;
    jointMaxVelc[1] = 5 * M_PI / 180;
    jointMaxVelc[2] = 5 * M_PI / 180;
    jointMaxVelc[3] = 5 * M_PI / 180;
    jointMaxVelc[4] = 5 * M_PI / 180;
    jointMaxVelc[5] = 5 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置基坐标系
    AuboRobot.rs_set_base_coord(rshd);

    //目标位置
    AuboRobot.Pos pos = new AuboRobot.Pos();
    pos.x = -0.502071;
    pos.y = -0.104238;
    pos.z = 0.476826;

    //工具参数，法兰盘中心
    AuboRobot.ToolInEndDesc tool_pos = new AuboRobot.ToolInEndDesc();
```

```

    tool_pos.orientation.w = 1;
    tool_pos.orientation.x = 0;
    tool_pos.orientation.y = 0;
    tool_pos.orientation.z = 0;
    tool_pos.cartPos.x = 0;
    tool_pos.cartPos.y = 0;
    tool_pos.cartPos.z = 0;

    int ret = AuboRobot.rs_move_joint_to(rshd, ref pos, ref tool_pos, true);
    if (ret == RS_SUCC)
    {
        Console.Out.WriteLine("关节运动到目标位置成功!");
    }
    else
    {
        Console.Error.WriteLine("关节运动到目标位置失败!错误码: {0}", ret);
    }
}

```

rs_move_joint_to 函数示例 2

本示例是轴动到工具在基坐标系下的位置。

代码如下：

```

static void jointMoveToExample2(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 5 * M_PI / 180;
    jointMaxAcc[1] = 5 * M_PI / 180;
    jointMaxAcc[2] = 5 * M_PI / 180;
    jointMaxAcc[3] = 5 * M_PI / 180;
    jointMaxAcc[4] = 5 * M_PI / 180;
    jointMaxAcc[5] = 5 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 5 * M_PI / 180;
}

```

```

jointMaxVelc[1] = 5 * M_PI / 180;
jointMaxVelc[2] = 5 * M_PI / 180;
jointMaxVelc[3] = 5 * M_PI / 180;
jointMaxVelc[4] = 5 * M_PI / 180;
jointMaxVelc[5] = 5 * M_PI / 180;
AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

//设置基坐标系
AuboRobot.rs_set_base_coord(rshd);

//目标位置
AuboRobot.Pos pos = new AuboRobot.Pos();
pos.x = -0.6;
pos.y = -0.3;
pos.z = 0.37;

//工具参数，法兰盘中心
AuboRobot.ToolInEndDesc tool_pos = new AuboRobot.ToolInEndDesc();
tool_pos.orientation.w = 1;
tool_pos.orientation.x = 0;
tool_pos.orientation.y = 0;
tool_pos.orientation.z = 0;
tool_pos.cartPos.x = 0.2;
tool_pos.cartPos.y = 0.1;
tool_pos.cartPos.z = 0.1;

int ret = AuboRobot.rs_move_joint_to(rshd, ref pos, ref tool_pos, true);
if (ret == RS_SUCC)
{
    Console.Out.WriteLine("关节运动到目标位置成功!");
}
else
{
    Console.Error.WriteLine("关节运动到目标位置失败!错误码: {0}", ret);
}
}

```

rs_move_joint_to 函数示例 3

本示例是轴动到法兰盘中心在用户坐标系下的位置和姿态。

代码如下：

```
static void jointMoveToExample3(UInt16 rshd)
```

```

{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 5 * M_PI / 180;
    jointMaxAcc[1] = 5 * M_PI / 180;
    jointMaxAcc[2] = 5 * M_PI / 180;
    jointMaxAcc[3] = 5 * M_PI / 180;
    jointMaxAcc[4] = 5 * M_PI / 180;
    jointMaxAcc[5] = 5 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 5 * M_PI / 180;
    jointMaxVelc[1] = 5 * M_PI / 180;
    jointMaxVelc[2] = 5 * M_PI / 180;
    jointMaxVelc[3] = 5 * M_PI / 180;
    jointMaxVelc[4] = 5 * M_PI / 180;
    jointMaxVelc[5] = 5 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //目标位置
    AuboRobot.Pos pos = new AuboRobot.Pos();
    pos.x = -0.075;
    pos.y = -0.002;
    pos.z = 0.067;

    //工具参数
    AuboRobot.ToolInEndDesc tool_pos = new AuboRobot.ToolInEndDesc();
    tool_pos.orientation.w = 1;
    tool_pos.orientation.x = 0;
    tool_pos.orientation.y = 0;
    tool_pos.orientation.z = 0;
    tool_pos.cartPos.x = 0;
    tool_pos.cartPos.y = 0;
    tool_pos.cartPos.z = 0;

    //用户坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
}

```

```

    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord,
d, typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 2;
    user_coord.methods = 0;
    user_coord.jointPara[0].jointRadian[0] = -14.717415 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[1] = -9.423585 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[2] = -74.757117 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[3] = 22.657165 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[4] = -84.449238 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[5] = -14.946778 * M_PI / 180.0;

    user_coord.jointPara[1].jointRadian[0] = 24.363602 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[1] = 7.233866 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[2] = -55.242389 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[3] = 22.464115 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[4] = -86.957003 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[5] = 24.171302 * M_PI / 180.0;

    user_coord.jointPara[2].jointRadian[0] = -12.492231 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[1] = 0.324244 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[2] = -65.331736 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[3] = 22.120198 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[4] = -84.531432 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[5] = -12.712789 * M_PI / 180.0;

    //坐标系的工具参数
    AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
());
    tooluser_coord.cartPos.x = 0.1;
    tooluser_coord.cartPos.y = 0.1;
    tooluser_coord.cartPos.z = 0.2;
    tooluser_coord.orientation.w = 1;
    tooluser_coord.orientation.x = 0;
    tooluser_coord.orientation.y = 0;
    tooluser_coord.orientation.z = 0;

    user_coord.toolDesc = tooluser_coord;

    AuboRobot.rs_set_user_coord(rshd, ref user_coord);

    int ret = AuboRobot.rs_move_joint_to(rshd, ref pos, ref tool_pos, true);
    if (ret == RS_SUCC)
    {
        Console.Out.WriteLine("关节运动到目标位置成功!");
    }

```



```
    }  
    else  
    {  
  
        Console.Error.WriteLine("关节运动到目标位置失败!错误码: {0}", ret);  
  
    }  
}
```

rs_move_joint_to 函数示例 4

本示例是轴动到工具在用户坐标系下的位置。

代码如下：

```
static void jointMoveToExample4(UInt16 rshd)  
{  
    //初始化运动属性  
    AuboRobot.rs_init_global_move_profile(rshd);  
  
    //设置关节运动最大加速度  
    double[] jointMaxAcc = new double[6];  
    jointMaxAcc[0] = 5 * M_PI / 180;  
    jointMaxAcc[1] = 5 * M_PI / 180;  
    jointMaxAcc[2] = 5 * M_PI / 180;  
    jointMaxAcc[3] = 5 * M_PI / 180;  
    jointMaxAcc[4] = 5 * M_PI / 180;  
    jointMaxAcc[5] = 5 * M_PI / 180;  
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);  
  
    //设置关节运动最大速度  
    double[] jointMaxVelc = new double[6];  
    jointMaxVelc[0] = 5 * M_PI / 180;  
    jointMaxVelc[1] = 5 * M_PI / 180;  
    jointMaxVelc[2] = 5 * M_PI / 180;  
    jointMaxVelc[3] = 5 * M_PI / 180;  
    jointMaxVelc[4] = 5 * M_PI / 180;  
    jointMaxVelc[5] = 5 * M_PI / 180;  
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);  
  
    //目标位置  
    AuboRobot.Pos pos = new AuboRobot.Pos();  
    pos.x = 0.115;  
    pos.y = 0.092;  
    pos.z = 0.189;
```

```

//工具参数
AuboRobot.ToolInEndDesc tool_pos = new AuboRobot.ToolInEndDesc();
tool_pos.orientation.w = 1;
tool_pos.orientation.x = 0;
tool_pos.orientation.y = 0;
tool_pos.orientation.z = 0;
tool_pos.cartPos.x = 0.2;
tool_pos.cartPos.y = 0.1;
tool_pos.cartPos.z = 0.1;

//用户坐标系
AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
user_coord.coordType = 2;
user_coord.methods = 0;
user_coord.jointPara[0].jointRadian[0] = -14.717415 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[1] = -9.423585 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[2] = -74.757117 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[3] = 22.657165 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[4] = -84.449238 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[5] = -14.946778 * M_PI / 180.0;

user_coord.jointPara[1].jointRadian[0] = 24.363602 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[1] = 7.233866 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[2] = -55.242389 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[3] = 22.464115 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[4] = -86.957003 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[5] = 24.171302 * M_PI / 180.0;

user_coord.jointPara[2].jointRadian[0] = -12.492231 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[1] = 0.324244 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[2] = -65.331736 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[3] = 22.120198 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[4] = -84.531432 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[5] = -12.712789 * M_PI / 180.0;

//坐标系的工具参数
AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
();
tooluser_coord.cartPos.x = 0.1;
tooluser_coord.cartPos.y = 0.1;

```

```

    tooluser_coord.cartPos.z = 0.2;
    tooluser_coord.orientation.w = 1;
    tooluser_coord.orientation.x = 0;
    tooluser_coord.orientation.y = 0;
    tooluser_coord.orientation.z = 0;

    user_coord.toolDesc = tooluser_coord;

    AuboRobot.rs_set_user_coord(rshd, ref user_coord);

    int ret = AuboRobot.rs_move_joint_to(rshd, ref pos, ref tool_pos, true);
    if (ret == RS_SUCC)
    {
        Console.Out.WriteLine("关节运动到目标位置成功!");
    }
    else
    {
        Console.Error.WriteLine("关节运动到目标位置失败!错误码: {0}", ret);
    }
}

```

4.8 跟随模式

4.8.1 跟随模式之提前到位

本示例是跟随模式之提前到位。

代码如下：

```

static void arrivalAheadExample(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 10 * M_PI / 180;
    jointMaxAcc[1] = 10 * M_PI / 180;
    jointMaxAcc[2] = 10 * M_PI / 180;
    jointMaxAcc[3] = 10 * M_PI / 180;
    jointMaxAcc[4] = 10 * M_PI / 180;
    jointMaxAcc[5] = 10 * M_PI / 180;
}

```

```
AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

//设置关节运动最大速度
double[] jointMaxVelc = new double[6];
jointMaxVelc[0] = 10 * M_PI / 180;
jointMaxVelc[1] = 10 * M_PI / 180;
jointMaxVelc[2] = 10 * M_PI / 180;
jointMaxVelc[3] = 10 * M_PI / 180;
jointMaxVelc[4] = 10 * M_PI / 180;
jointMaxVelc[5] = 10 * M_PI / 180;
AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

//起始路点
double[] initPos = new double[6];
initPos[0] = 173.108713 / 180 * M_PI;
initPos[1] = -12.075005 / 180 * M_PI;
initPos[2] = -83.663342 / 180 * M_PI;
initPos[3] = -15.641249 / 180 * M_PI;
initPos[4] = -89.140000 / 180 * M_PI;
initPos[5] = -28.328713 / 180 * M_PI;

//关节运动到起始路点
AuboRobot.rs_move_joint(rshd, initPos, true);

double[] jointAngle = new double[6];
for (int i = 0; i < 5; i++)
{
    if (i % 2 == 0)
    {
        //跟随模式之提前到位 当前仅适用于关节运动
        //设置提前到位的距离模式
        AuboRobot.rs_set_arrival_ahead_distance(rshd, 0.2);
    }
    else
    {
        AuboRobot.rs_set_no_arrival_ahead(rshd);
    }
}

jointAngle[0] = 20.0 / 180.0 * M_PI;
jointAngle[1] = 0.0 / 180.0 * M_PI;
jointAngle[2] = 90.0 / 180.0 * M_PI;
jointAngle[3] = 0.0 / 180.0 * M_PI;
jointAngle[4] = 90.0 / 180.0 * M_PI;
```

```

        jointAngle[5] = 0.0 / 180.0 * M_PI;

        int ret = AuboRobot.rs_move_joint(rshd, jointAngle, true);
        if (ret != RS_SUCC)
        {
            Console.Error.WriteLine("运动 1 失败。错误号为: {0}", ret);
            break;
        }
        else
        {
            Console.Out.WriteLine("运动 1 成功。i = {0}", i);
        }

        jointAngle[0] = 50.0 / 180.0 * M_PI;
        jointAngle[1] = 40.0 / 180.0 * M_PI;
        jointAngle[2] = 78.0 / 180.0 * M_PI;
        jointAngle[3] = 20.0 / 180.0 * M_PI;
        jointAngle[4] = 66.0 / 180.0 * M_PI;
        jointAngle[5] = 0.0 / 180.0 * M_PI;

        ret = AuboRobot.rs_move_joint(rshd, jointAngle, true);
        if (ret != RS_SUCC)
        {
            Console.Error.WriteLine("运动 2 失败。错误号为: {0}", ret);
            break;
        }
        else
        {
            Console.Out.WriteLine("运动 2 成功。 i = {0}", i);
        }
    }
}

```

4.9 直线运动

4.9.1 rs_move_line 函数

本示例是直线运动到指定关节角。

代码如下：

```
static void lineMoveExample(UInt16 rshd)
```

```
{  
    //初始化运动属性  
    AuboRobot.rs_init_global_move_profile(rshd);  
  
    //设置关节运动最大加速度  
    double[] jointMaxAcc = new double[6];  
    jointMaxAcc[0] = 30 * M_PI / 180;  
    jointMaxAcc[1] = 30 * M_PI / 180;  
    jointMaxAcc[2] = 30 * M_PI / 180;  
    jointMaxAcc[3] = 30 * M_PI / 180;  
    jointMaxAcc[4] = 30 * M_PI / 180;  
    jointMaxAcc[5] = 30 * M_PI / 180;  
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);  
  
    //设置关节运动最大速度  
    double[] jointMaxVelc = new double[6];  
    jointMaxVelc[0] = 30 * M_PI / 180;  
    jointMaxVelc[1] = 30 * M_PI / 180;  
    jointMaxVelc[2] = 30 * M_PI / 180;  
    jointMaxVelc[3] = 30 * M_PI / 180;  
    jointMaxVelc[4] = 30 * M_PI / 180;  
    jointMaxVelc[5] = 30 * M_PI / 180;  
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);  
  
    //设置末端型运动最大加速度  
    double lineMaxAcc = 0.2;  
    AuboRobot.rs_set_global_end_max_line_acc(rshd, lineMaxAcc);  
  
    //设置末端型运动最大速度  
    double lineMaxVelc = 0.2;  
    AuboRobot.rs_set_global_end_max_line_velc(rshd, lineMaxVelc);  
  
    //初始位置  
    double[] initPos = {  
        173.108713 / 180 * M_PI,  
        -12.075005 / 180 * M_PI,  
        -83.663342 / 180 * M_PI,  
        -15.641249 / 180 * M_PI,  
        -89.140000 / 180 * M_PI,  
        -28.328713 / 180 * M_PI  
    };  
  
    //关节运动到初始位置
```

```
AuboRobot.rs_move_joint(rshd, initPos, true);

double[] jointAngle = new double[6];
jointAngle[0] = 173.108617 * M_PI / 180;
jointAngle[1] = -11.163866 * M_PI / 180;
jointAngle[2] = -122.428532 * M_PI / 180;
jointAngle[3] = -55.317091 * M_PI / 180;
jointAngle[4] = -89.139597 * M_PI / 180;
jointAngle[5] = -28.328648 * M_PI / 180;

//直线运动
int ret = AuboRobot.rs_move_line(rshd, jointAngle, true);
if (ret == RS_SUCC)
{
    Console.Out.WriteLine("直线运动成功!");
}
else
{
    Console.Error.WriteLine("直线运动失败! 错误码: {0}", ret);
}
}
```

4.9.2 rs_move_line_to 函数

rs_move_line_to 函数示例 1

本示例是直线运动到法兰盘中心在基坐标系下的位置。

代码如下：

```
static void lineMoveToExample1(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 10 * M_PI / 180;
    jointMaxAcc[1] = 10 * M_PI / 180;
    jointMaxAcc[2] = 10 * M_PI / 180;
    jointMaxAcc[3] = 10 * M_PI / 180;
    jointMaxAcc[4] = 10 * M_PI / 180;
    jointMaxAcc[5] = 10 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 10 * M_PI / 180;
    jointMaxVelc[1] = 10 * M_PI / 180;
    jointMaxVelc[2] = 10 * M_PI / 180;
    jointMaxVelc[3] = 10 * M_PI / 180;
    jointMaxVelc[4] = 10 * M_PI / 180;
    jointMaxVelc[5] = 10 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动最大加速度
    double lineMaxAcc = 0.1;
    AuboRobot.rs_set_global_end_max_line_acc(rshd, lineMaxAcc);

    //设置末端型运动最大速度
    double lineMaxVelc = 0.1;
    AuboRobot.rs_set_global_end_max_line_velc(rshd, lineMaxVelc);

    //初始位置
    double[] initPos = {
```



```
        173.108713 / 180 * M_PI,  
        -12.075005 / 180 * M_PI,  
        -83.663342 / 180 * M_PI,  
        -15.641249 / 180 * M_PI,  
        -89.140000 / 180 * M_PI,  
        -28.328713 / 180 * M_PI  
    };  
  
    //关节运动到初始位置  
    AuboRobot.rs_move_joint(rshd, initPos, true);  
  
    //目标位置  
    AuboRobot.Pos pos = new AuboRobot.Pos();  
    pos.x = 0.361276;  
    pos.y = -0.208021;  
    pos.z = 0.577276;  
  
    //设置基坐标系  
    AuboRobot.rs_set_base_coord(rshd);  
  
    //工具参数，即法兰盘中心  
    AuboRobot.ToolInEndDesc tool_pos = new AuboRobot.ToolInEndDesc();  
    tool_pos.orientation.w = 1;  
    tool_pos.orientation.x = 0;  
    tool_pos.orientation.y = 0;  
    tool_pos.orientation.z = 0;  
    tool_pos.cartPos.x = 0;  
    tool_pos.cartPos.y = 0;  
    tool_pos.cartPos.z = 0;  
  
    //直线运动到目标位置  
    int ret = AuboRobot.rs_move_line_to(rshd, ref pos, ref tool_pos, true);  
    if (ret == RS_SUCC)  
    {  
        Console.Out.WriteLine("直线运动到目标位置成功!");  
    }  
    else  
    {  
        Console.Error.WriteLine("直线运动到目标位置失败! 错误码: {0}", ret);  
    }  
}
```

rs_move_line_to 函数示例 2

本示例是直线运动到工具在基坐标系下的位置。

代码如下：

```
static void lineMoveToExample2(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 10 * M_PI / 180;
    jointMaxAcc[1] = 10 * M_PI / 180;
    jointMaxAcc[2] = 10 * M_PI / 180;
    jointMaxAcc[3] = 10 * M_PI / 180;
    jointMaxAcc[4] = 10 * M_PI / 180;
    jointMaxAcc[5] = 10 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 10 * M_PI / 180;
    jointMaxVelc[1] = 10 * M_PI / 180;
    jointMaxVelc[2] = 10 * M_PI / 180;
    jointMaxVelc[3] = 10 * M_PI / 180;
    jointMaxVelc[4] = 10 * M_PI / 180;
    jointMaxVelc[5] = 10 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动最大加速度
    double lineMaxAcc = 0.2;
    AuboRobot.rs_set_global_end_max_line_acc(rshd, lineMaxAcc);

    //设置末端型运动最大速度
    double lineMaxVelc = 0.2;
    AuboRobot.rs_set_global_end_max_line_velc(rshd, lineMaxVelc);

    //初始位置
    double[] initPos = {
        173.108713 / 180 * M_PI,
        -12.075005 / 180 * M_PI,
        -83.663342 / 180 * M_PI,
```

```
-15.641249 / 180 * M_PI,  
-89.140000 / 180 * M_PI,  
-28.328713 / 180 * M_PI  
};  
  
//关节运动到初始位置  
AuboRobot.rs_move_joint(rshd, initPos, true);  
  
//设置基坐标系  
AuboRobot.rs_set_base_coord(rshd);  
  
//目标位置  
AuboRobot.Pos pos = new AuboRobot.Pos();  
pos.x = 0.436330;  
pos.y = 0.009522;  
pos.z = 0.493359;  
  
//工具参数  
AuboRobot.ToolInEndDesc tool_pos = new AuboRobot.ToolInEndDesc();  
tool_pos.orientation.w = 1;  
tool_pos.orientation.x = 0;  
tool_pos.orientation.y = 0;  
tool_pos.orientation.z = 0;  
tool_pos.cartPos.x = 0.2;  
tool_pos.cartPos.y = 0.1;  
tool_pos.cartPos.z = 0.1;  
  
//直线运动到目标位置  
int ret = AuboRobot.rs_move_line_to(rshd, ref pos, ref tool_pos, true);  
if (ret == RS_SUCC)  
{  
    Console.Out.WriteLine("直线运动到目标位置成功!");  
}  
else  
{  
    Console.Error.WriteLine("直线运动到目标位置失败! 错误码: {0}", ret);  
}  
}
```

rs_move_line_to 函数示例 3

本示例是直线运动到法兰盘中心在用户坐标系下的位置。

代码如下：

```
static void lineMoveToExample3(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 10 * M_PI / 180;
    jointMaxAcc[1] = 10 * M_PI / 180;
    jointMaxAcc[2] = 10 * M_PI / 180;
    jointMaxAcc[3] = 10 * M_PI / 180;
    jointMaxAcc[4] = 10 * M_PI / 180;
    jointMaxAcc[5] = 10 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 10 * M_PI / 180;
    jointMaxVelc[1] = 10 * M_PI / 180;
    jointMaxVelc[2] = 10 * M_PI / 180;
    jointMaxVelc[3] = 10 * M_PI / 180;
    jointMaxVelc[4] = 10 * M_PI / 180;
    jointMaxVelc[5] = 10 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动最大加速度
    double lineMaxAcc = 0.2;
    AuboRobot.rs_set_global_end_max_line_acc(rshd, lineMaxAcc);

    //设置末端型运动最大速度
    double lineMaxVelc = 0.2;
    AuboRobot.rs_set_global_end_max_line_velc(rshd, lineMaxVelc);

    //初始位置
    double[] initPos = {
        173.108713 / 180 * M_PI,
        -12.075005 / 180 * M_PI,
        -83.663342 / 180 * M_PI,
```

```

        -15.641249 / 180 * M_PI,
        -89.140000 / 180 * M_PI,
        -28.328713 / 180 * M_PI
    };

    //关节运动到初始位置
    AuboRobot.rs_move_joint(rshd, initPos, true);

    //目标位置
    AuboRobot.Pos pos = new AuboRobot.Pos();
    pos.x = 0.575115;
    pos.y = 0.649255;
    pos.z = 0.171612;

    //用户坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 2;
    user_coord.methods = 0;
    user_coord.jointPara[0].jointRadian[0] = 41.122933 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[1] = -8.439166 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[2] = -77.604753 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[3] = 12.691872 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[4] = -89.793205 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[5] = -6.778675 * M_PI / 180.0;

    user_coord.jointPara[1].jointRadian[0] = 56.384957 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[1] = -15.511293 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[2] = -82.897186 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[3] = 14.701128 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[4] = -91.937030 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[5] = 8.334642 * M_PI / 180.0;

    user_coord.jointPara[2].jointRadian[0] = 22.662699 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[1] = -22.522148 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[2] = -87.122177 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[3] = 17.736179 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[4] = -87.233154 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[5] = -25.068271 * M_PI / 180.0;

    //坐标系的工具参数

```

```
AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
());
tooluser_coord.cartPos.x = 0.2;
tooluser_coord.cartPos.y = 0.1;
tooluser_coord.cartPos.z = 0.1;
tooluser_coord.orientation.w = 1;
tooluser_coord.orientation.x = 0;
tooluser_coord.orientation.y = 0;
tooluser_coord.orientation.z = 0;

user_coord.toolDesc = tooluser_coord;

AuboRobot.rs_set_user_coord(rshd, ref user_coord);

//工具参数, 法兰盘中心
AuboRobot.ToolInEndDesc tool_pos = new AuboRobot.ToolInEndDesc();
tool_pos.orientation.w = 1;
tool_pos.orientation.x = 0;
tool_pos.orientation.y = 0;
tool_pos.orientation.z = 0;
tool_pos.cartPos.x = 0;
tool_pos.cartPos.y = 0;
tool_pos.cartPos.z = 0;

//直线运动到目标位置
int ret = AuboRobot.rs_move_line_to(rshd, ref pos, ref tool_pos, true);
if (ret == RS_SUCC)
{
    Console.Out.WriteLine("直线运动到目标位置成功!");
}
else
{
    Console.Error.WriteLine("直线运动到目标位置失败!错误码: {0}", ret);
}
}
```

rs_move_line_to 函数示例 4

本示例是直线运动到工具在用户坐标系下的位置。

代码如下：

```
static void lineMoveToExample4(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 10 * M_PI / 180;
    jointMaxAcc[1] = 10 * M_PI / 180;
    jointMaxAcc[2] = 10 * M_PI / 180;
    jointMaxAcc[3] = 10 * M_PI / 180;
    jointMaxAcc[4] = 10 * M_PI / 180;
    jointMaxAcc[5] = 10 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 10 * M_PI / 180;
    jointMaxVelc[1] = 10 * M_PI / 180;
    jointMaxVelc[2] = 10 * M_PI / 180;
    jointMaxVelc[3] = 10 * M_PI / 180;
    jointMaxVelc[4] = 10 * M_PI / 180;
    jointMaxVelc[5] = 10 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动最大加速度
    double lineMaxAcc = 0.2;
    AuboRobot.rs_set_global_end_max_line_acc(rshd, lineMaxAcc);

    //设置末端型运动最大速度
    double lineMaxVelc = 0.2;
    AuboRobot.rs_set_global_end_max_line_velc(rshd, lineMaxVelc);

    //初始位置
    double[] initPos = {
        173.108713 / 180 * M_PI,
        -12.075005 / 180 * M_PI,
        -83.663342 / 180 * M_PI,
```

```

        -15.641249 / 180 * M_PI,
        -89.140000 / 180 * M_PI,
        -28.328713 / 180 * M_PI
    };

    //关节运动到初始位置
    AuboRobot.rs_move_joint(rshd, initPos, true);

    //目标位置
    AuboRobot.Pos pos = new AuboRobot.Pos();
    pos.x = 0.650169;
    pos.y = 0.866797;
    pos.z = 0.112075;

    //用户坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 2;
    user_coord.methods = 0;
    user_coord.jointPara[0].jointRadian[0] = 41.122933 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[1] = -8.439166 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[2] = -77.604753 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[3] = 12.691872 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[4] = -89.793205 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[5] = -6.778675 * M_PI / 180.0;

    user_coord.jointPara[1].jointRadian[0] = 56.384957 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[1] = -15.511293 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[2] = -82.897186 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[3] = 14.701128 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[4] = -91.937030 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[5] = 8.334642 * M_PI / 180.0;

    user_coord.jointPara[2].jointRadian[0] = 22.662699 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[1] = -22.522148 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[2] = -87.122177 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[3] = 17.736179 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[4] = -87.233154 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[5] = -25.068271 * M_PI / 180.0;

    //坐标系的工具参数

```



```
AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
());
tooluser_coord.cartPos.x = 0.2;
tooluser_coord.cartPos.y = 0.1;
tooluser_coord.cartPos.z = 0.1;
tooluser_coord.orientation.w = 1;
tooluser_coord.orientation.x = 0;
tooluser_coord.orientation.y = 0;
tooluser_coord.orientation.z = 0;

user_coord.toolDesc = tooluser_coord;

AuboRobot.rs_set_user_coord(rshd, ref user_coord);

//工具参数
AuboRobot.ToolInEndDesc tool_pos = new AuboRobot.ToolInEndDesc();
tool_pos.orientation.w = 1;
tool_pos.orientation.x = 0;
tool_pos.orientation.y = 0;
tool_pos.orientation.z = 0;
tool_pos.cartPos.x = 0.2;
tool_pos.cartPos.y = 0.1;
tool_pos.cartPos.z = 0.1;

//直线运动到目标位置
int ret = AuboRobot.rs_move_line_to(rshd, ref pos, ref tool_pos, true);
if (ret == RS_SUCC)
{
    Console.Out.WriteLine("直线运动到目标位置成功!");
}
else
{
    Console.Error.WriteLine("直线运动到目标位置失败!错误码: {0}", ret);
}
}
```

4.10 偏移运动

4.10.1 rs_set_relative_offset_on_base 函数

示例 1：法兰盘中心在基坐标系下

本示例是机械臂做位置偏移运动。

代码如下：

```
static void relativeMoveOnBase1(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节型运动的最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[1] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[2] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[3] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[4] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节型运动的最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[1] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[2] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[3] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[4] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动的最大加速度
    double endMoveMaxAcc;
    endMoveMaxAcc = 0.2; //单位米每秒
    AuboRobot.rs_set_global_end_max_line_acc(rshd, endMoveMaxAcc);
    AuboRobot.rs_set_global_end_max_angle_acc(rshd, endMoveMaxAcc);

    //设置末端型运动的最大速度
    double endMoveMaxVelc;
    endMoveMaxVelc = 0.2; //单位米每秒
```

```

AuboRobot.rs_set_global_end_max_line_velc(rshd, endMoveMaxVele);
AuboRobot.rs_set_global_end_max_angle_velc(rshd, endMoveMaxVele);

//基坐标系
AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
user_coord.coordType = 0;

//偏移量
AuboRobot.MoveRelative relative = new AuboRobot.MoveRelative();
IntPtr pt_relative = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.MoveRelative)));
relative = (AuboRobot.MoveRelative)Marshal.PtrToStructure(pt_relative, typeof(AuboRobot.MoveRelative));
relative.enable = 1;
relative.orientation.w = 1;
relative.orientation.x = 0;
relative.orientation.y = 0;
relative.orientation.z = 0;
relative.pos[0] = 0;
relative.pos[1] = 0;
relative.pos[2] = 0.1F;

double[] target0 = new double[6]; //注意这个里面的值是弧度!
target0[0] = -0.000172 / 180 * M_PI;
target0[1] = -7.291862 / 180 * M_PI;
target0[2] = -75.694718 / 180 * M_PI;
target0[3] = 21.596727 / 180 * M_PI;
target0[4] = -89.999982 / 180 * M_PI;
target0[5] = -0.00458 / 180 * M_PI;

//移动到坐标系原点
AuboRobot.rs_move_joint(rshd, target0, true);

//相对坐标系原点沿z轴正向运动
AuboRobot.rs_set_relative_offset_on_user(rshd, ref relative, ref user_coord);
AuboRobot.rs_move_line(rshd, target0, true);
}

```

示例 2：工具末端在基坐标系下

本示例是机械臂做位置偏移运动。

代码如下：

```
static void relativeMoveOnBase2(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节型运动的最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[1] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[2] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[3] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[4] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节型运动的最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[1] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[2] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[3] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[4] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动的最大加速度
    double endMoveMaxAcc;
    endMoveMaxAcc = 0.2; //单位米每秒
    AuboRobot.rs_set_global_end_max_line_acc(rshd, endMoveMaxAcc);
    AuboRobot.rs_set_global_end_max_angle_acc(rshd, endMoveMaxAcc);

    //设置末端型运动的最大速度
    double endMoveMaxVelc;
    endMoveMaxVelc = 0.2; //单位米每秒
    AuboRobot.rs_set_global_end_max_line_velc(rshd, endMoveMaxVelc);
    AuboRobot.rs_set_global_end_max_angle_velc(rshd, endMoveMaxVelc);

    //基坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
```

```

    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 0;

    //偏移量
    AuboRobot.MoveRelative relative = new AuboRobot.MoveRelative();
    IntPtr pt_relative = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.MoveRelative)));
    relative = (AuboRobot.MoveRelative)Marshal.PtrToStructure(pt_relative, typeof(AuboRobot.MoveRelative));
    relative.enable = 1;
    relative.orientation.w = 1;
    relative.orientation.x = 0;
    relative.orientation.y = 0;
    relative.orientation.z = 0;
    relative.pos[0] = 0;
    relative.pos[1] = 0;
    relative.pos[2] = 0.1F;

    double[] target0 = new double[6]; //注意这个里面的值是弧度!
    target0[0] = -0.000172 / 180 * M_PI;
    target0[1] = -7.291862 / 180 * M_PI;
    target0[2] = -75.694718 / 180 * M_PI;
    target0[3] = 21.596727 / 180 * M_PI;
    target0[4] = -89.999982 / 180 * M_PI;
    target0[5] = -0.00458 / 180 * M_PI;

    //设置工具参数
    AuboRobot.ToolInEndDesc tool = new AuboRobot.ToolInEndDesc();
    tool.cartPos.x = 0.2;
    tool.cartPos.y = 0.1;
    tool.cartPos.z = 0.1;
    tool.orientation.w = 1;
    tool.orientation.x = 1;
    tool.orientation.y = 1;
    tool.orientation.z = 1;
    AuboRobot.rs_set_tool_kinematics_param(rshd, ref tool);

    //移动到坐标系原点
    AuboRobot.rs_move_joint(rshd, target0, true);

```

```

//相对坐标系原点沿z 轴正向运动
AuboRobot.rs_set_relative_offset_on_user(rshd, ref relative, ref user_coord);
AuboRobot.rs_move_line(rshd, target0, true);
}

```

4.10.2 rs_set_relative_offset_on_user 函数

示例 1：法兰盘中心在用户坐标系下

本示例是机械臂做位置偏移运动。

代码如下：

```

static void relativeMoveOnUser1(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节型运动的最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[1] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[2] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[3] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[4] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节型运动的最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[1] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[2] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[3] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[4] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动的最大加速度
    double endMoveMaxAcc;
    endMoveMaxAcc = 0.2; //单位米每秒
    AuboRobot.rs_set_global_end_max_line_acc(rshd, endMoveMaxAcc);
    AuboRobot.rs_set_global_end_max_angle_acc(rshd, endMoveMaxAcc);
}

```

```

//设置末端型运动的最大速度
double endMoveMaxVelc;
endMoveMaxVelc = 0.2; //单位米每秒
AuboRobot.rs_set_global_end_max_line_velc(rshd, endMoveMaxVelc);
AuboRobot.rs_set_global_end_max_angle_velc(rshd, endMoveMaxVelc);

//用户坐标系
AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
user_coord.coordType = 2;
user_coord.methods = 0;
user_coord.jointPara[0].jointRadian[0] = -14.717415 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[1] = -9.423585 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[2] = -74.757117 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[3] = 22.657165 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[4] = -84.449238 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[5] = -14.946778 * M_PI / 180.0;

user_coord.jointPara[1].jointRadian[0] = 24.363602 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[1] = 7.233866 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[2] = -55.242389 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[3] = 22.464115 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[4] = -86.957003 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[5] = 24.171302 * M_PI / 180.0;

user_coord.jointPara[2].jointRadian[0] = -12.492231 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[1] = 0.324244 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[2] = -65.331736 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[3] = 22.120198 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[4] = -84.531432 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[5] = -12.712789 * M_PI / 180.0;

//坐标系的工具参数
AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc();
tooluser_coord.cartPos.x = 0.1;
tooluser_coord.cartPos.y = 0.1;
tooluser_coord.cartPos.z = 0.2;
tooluser_coord.orientation.w = 1;
tooluser_coord.orientation.x = 0;

```

```
tooluser_coord.orientation.y = 0;
tooluser_coord.orientation.z = 0;

user_coord.toolDesc = tooluser_coord;

//偏移量
AuboRobot.MoveRelative relative = new AuboRobot.MoveRelative();
IntPtr pt_relative = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.
MoveRelative)));
relative = (AuboRobot.MoveRelative)Marshal.PtrToStructure(pt_relative, typeof
(AuboRobot.MoveRelative));
relative.enable = 1;
relative.orientation.w = 1;
relative.orientation.x = 0;
relative.orientation.y = 0;
relative.orientation.z = 0;
relative.pos[0] = 0;
relative.pos[1] = 0;
relative.pos[2] = 0.01F;

double[] target0 = new double[6]; //注意这个里面的值是弧度!
target0[0] = -0.000172 / 180 * M_PI;
target0[1] = -7.291862 / 180 * M_PI;
target0[2] = -75.694718 / 180 * M_PI;
target0[3] = 21.596727 / 180 * M_PI;
target0[4] = -89.999982 / 180 * M_PI;
target0[5] = -0.00458 / 180 * M_PI;

//移动到坐标系原点
AuboRobot.rs_move_joint(rshd, target0, true);

//相对坐标系原点沿z轴正向运动
AuboRobot.rs_set_relative_offset_on_user(rshd, ref relative, ref user_coord);
AuboRobot.rs_move_line(rshd, target0, true);
}
```


示例 2：工具末端在用户坐标系下

本示例是机械臂做偏移运动。

代码如下：

```
static void relativeMoveOnUser2(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节型运动的最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[1] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[2] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[3] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[4] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节型运动的最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[1] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[2] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[3] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[4] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动的最大加速度
    double endMoveMaxAcc;
    endMoveMaxAcc = 0.2; //单位米每秒
    AuboRobot.rs_set_global_end_max_line_acc(rshd, endMoveMaxAcc);
    AuboRobot.rs_set_global_end_max_angle_acc(rshd, endMoveMaxAcc);

    //设置末端型运动的最大速度
    double endMoveMaxVelc;
    endMoveMaxVelc = 0.2; //单位米每秒
    AuboRobot.rs_set_global_end_max_line_velc(rshd, endMoveMaxVelc);
    AuboRobot.rs_set_global_end_max_angle_velc(rshd, endMoveMaxVelc);

    //用户坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
}
```

```

    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord,
    d, typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 2;
    user_coord.methods = 0;
    user_coord.jointPara[0].jointRadian[0] = -14.717415 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[1] = -9.423585 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[2] = -74.757117 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[3] = 22.657165 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[4] = -84.449238 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[5] = -14.946778 * M_PI / 180.0;

    user_coord.jointPara[1].jointRadian[0] = 24.363602 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[1] = 7.233866 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[2] = -55.242389 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[3] = 22.464115 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[4] = -86.957003 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[5] = 24.171302 * M_PI / 180.0;

    user_coord.jointPara[2].jointRadian[0] = -12.492231 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[1] = 0.324244 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[2] = -65.331736 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[3] = 22.120198 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[4] = -84.531432 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[5] = -12.712789 * M_PI / 180.0;

    //坐标系的工具参数
    AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
    ();
    tooluser_coord.cartPos.x = 0.1;
    tooluser_coord.cartPos.y = 0.1;
    tooluser_coord.cartPos.z = 0.2;
    tooluser_coord.orientation.w = 1;
    tooluser_coord.orientation.x = 0;
    tooluser_coord.orientation.y = 0;
    tooluser_coord.orientation.z = 0;

    user_coord.toolDesc = tooluser_coord;

    //偏移量
    AuboRobot.MoveRelative relative = new AuboRobot.MoveRelative();
    IntPtr pt_relative = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.
    MoveRelative)));

```

```
relative = (AuboRobot.MoveRelative)Marshal.PtrToStructure(pt_relative, typeof
(AuboRobot.MoveRelative));
relative.enable = 1;
relative.orientation.w = 1;
relative.orientation.x = 0;
relative.orientation.y = 0;
relative.orientation.z = 0;
relative.pos[0] = 0;
relative.pos[1] = 0;
relative.pos[2] = 0.01F;

double[] target0 = new double[6]; //注意这个里面的值是弧度!
target0[0] = -0.000172 / 180 * M_PI;
target0[1] = -7.291862 / 180 * M_PI;
target0[2] = -75.694718 / 180 * M_PI;
target0[3] = 21.596727 / 180 * M_PI;
target0[4] = -89.999982 / 180 * M_PI;
target0[5] = -0.00458 / 180 * M_PI;

//设置工具参数
AuboRobot.ToolInEndDesc tool = new AuboRobot.ToolInEndDesc();
tool.cartPos.x = 0.2;
tool.cartPos.y = 0.1;
tool.cartPos.z = 0.1;
tool.orientation.w = 1;
tool.orientation.x = 1;
tool.orientation.y = 1;
tool.orientation.z = 1;
AuboRobot.rs_set_tool_kinematics_param(rshd, ref tool);

//移动到坐标系原点
AuboRobot.rs_move_joint(rshd, target0, true);

//相对坐标系原点沿z轴正向运动
AuboRobot.rs_set_relative_offset_on_user(rshd, ref relative, ref user_coord);
AuboRobot.rs_move_line(rshd, target0, true);
}
```

示例 3：工具末端在工具坐标系下

本示例是机械臂做位置偏移运动。

代码如下：

```
static void relativeMoveOnUser3(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节型运动的最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[1] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[2] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[3] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[4] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节型运动的最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[1] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[2] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[3] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[4] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动的最大加速度
    double endMoveMaxAcc;
    endMoveMaxAcc = 0.2; //单位米每秒
    AuboRobot.rs_set_global_end_max_line_acc(rshd, endMoveMaxAcc);
    AuboRobot.rs_set_global_end_max_angle_acc(rshd, endMoveMaxAcc);

    //设置末端型运动的最大速度
    double endMoveMaxVelc;
    endMoveMaxVelc = 0.2; //单位米每秒
    AuboRobot.rs_set_global_end_max_line_velc(rshd, endMoveMaxVelc);
    AuboRobot.rs_set_global_end_max_angle_velc(rshd, endMoveMaxVelc);

    //工具坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
```

```

    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord,
    typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 1;

    //坐标系的工具参数
    AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
    ();
    tooluser_coord.cartPos.x = 0.1;
    tooluser_coord.cartPos.y = 0.1;
    tooluser_coord.cartPos.z = 0.2;
    tooluser_coord.orientation.w = 1;
    tooluser_coord.orientation.x = 0;
    tooluser_coord.orientation.y = 0;
    tooluser_coord.orientation.z = 0;

    user_coord.toolDesc = tooluser_coord;

    //偏移量
    AuboRobot.MoveRelative relative = new AuboRobot.MoveRelative();
    IntPtr pt_relative = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.
    MoveRelative)));
    relative = (AuboRobot.MoveRelative)Marshal.PtrToStructure(pt_relative,
    typeof(AuboRobot.MoveRelative));
    relative.enable = 1;
    relative.orientation.w = 1;
    relative.orientation.x = 0;
    relative.orientation.y = 0;
    relative.orientation.z = 0;
    relative.pos[0] = 0;
    relative.pos[1] = 0;
    relative.pos[2] = 0.01F;

    double[] target0 = new double[6]; //注意这个里面的值是弧度!
    target0[0] = -0.000172 / 180 * M_PI;
    target0[1] = -7.291862 / 180 * M_PI;
    target0[2] = -75.694718 / 180 * M_PI;
    target0[3] = 21.596727 / 180 * M_PI;
    target0[4] = -89.999982 / 180 * M_PI;
    target0[5] = -0.00458 / 180 * M_PI;

    //设置工具参数
    AuboRobot.ToolInEndDesc tool = new AuboRobot.ToolInEndDesc();

```

```
tool.cartPos.x = 0.2;
tool.cartPos.y = 0.1;
tool.cartPos.z = 0.1;
tool.orientation.w = 1;
tool.orientation.x = 1;
tool.orientation.y = 1;
tool.orientation.z = 1;
AuboRobot.rs_set_tool_kinematics_param(rshd, ref tool);

//移动到坐标系原点
AuboRobot.rs_move_joint(rshd, target0, true);

//相对坐标系原点沿z 轴正向运动
AuboRobot.rs_set_relative_offset_on_user(rshd, ref relative, ref user_coord);
AuboRobot.rs_move_line(rshd, target0, true);
}
```

4.11 旋转运动

4.11.1 rs_move_rotate 函数

示例 1：法兰盘中心在基坐标系下

本示例是法兰盘中心在基坐标系下做旋转运动。

代码如下：

```
static void rotateMoveExample1(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 30 * M_PI / 180;
    jointMaxAcc[1] = 30 * M_PI / 180;
    jointMaxAcc[2] = 30 * M_PI / 180;
    jointMaxAcc[3] = 30 * M_PI / 180;
    jointMaxAcc[4] = 30 * M_PI / 180;
    jointMaxAcc[5] = 30 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 30 * M_PI / 180;
    jointMaxVelc[1] = 30 * M_PI / 180;
    jointMaxVelc[2] = 30 * M_PI / 180;
    jointMaxVelc[3] = 30 * M_PI / 180;
    jointMaxVelc[4] = 30 * M_PI / 180;
    jointMaxVelc[5] = 30 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动最大加速度
    double lineMaxAcc = 0.2;
    AuboRobot.rs_set_global_end_max_line_acc(rshd, lineMaxAcc);

    //设置末端型运动最大速度
    double lineMaxVelc = 0.2;
    AuboRobot.rs_set_global_end_max_line_velc(rshd, lineMaxVelc);
}
```

```

//初始位置
double[] initPos = {
    173.108713 / 180 * M_PI,
    -12.075005 / 180 * M_PI,
    -83.663342 / 180 * M_PI,
    -15.641249 / 180 * M_PI,
    -89.140000 / 180 * M_PI,
    -28.328713 / 180 * M_PI
};

//关节运动到初始位置
AuboRobot.rs_move_joint(rshd, initPos, true);

//设置基坐标系
AuboRobot.CoordCalibrate baseCoord = new AuboRobot.CoordCalibrate();
IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
baseCoord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord,
    typeof(AuboRobot.CoordCalibrate));
baseCoord.coordType = 0;

//旋转轴和旋转角度
AuboRobot.MoveRotateAxis axis = new AuboRobot.MoveRotateAxis();
IntPtr pt_axis = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.MoveRotateAxis)));
axis = (AuboRobot.MoveRotateAxis)Marshal.PtrToStructure(pt_axis, typeof(AuboRobot.MoveRotateAxis));
axis.rotateAxis[0] = 1;
axis.rotateAxis[1] = 0;
axis.rotateAxis[2] = 0;
double rotateAngle = 5.0 * M_PI / 180;

//旋转运动
int ret = AuboRobot.rs_move_rotate(rshd, ref baseCoord, ref axis, rotateAngle, true);
if (ret == RS_SUCC)
{
    Console.Out.WriteLine("旋转运动成功！");
}
else
{
    Console.Error.WriteLine("旋转运动失败！错误码：{0}", ret);
}
}

```


示例 2：末端工具在基坐标系下旋转

本示例是末端工具在基坐标系下做旋转运动。

代码如下：

```
static void rotateMoveExample2(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 30 * M_PI / 180;
    jointMaxAcc[1] = 30 * M_PI / 180;
    jointMaxAcc[2] = 30 * M_PI / 180;
    jointMaxAcc[3] = 30 * M_PI / 180;
    jointMaxAcc[4] = 30 * M_PI / 180;
    jointMaxAcc[5] = 30 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 30 * M_PI / 180;
    jointMaxVelc[1] = 30 * M_PI / 180;
    jointMaxVelc[2] = 30 * M_PI / 180;
    jointMaxVelc[3] = 30 * M_PI / 180;
    jointMaxVelc[4] = 30 * M_PI / 180;
    jointMaxVelc[5] = 30 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动最大加速度
    double lineMaxAcc = 0.2;
    AuboRobot.rs_set_global_end_max_line_acc(rshd, lineMaxAcc);

    //设置末端型运动最大速度
    double lineMaxVelc = 0.2;
    AuboRobot.rs_set_global_end_max_line_velc(rshd, lineMaxVelc);

    //初始位置
    double[] initPos = {
        173.108713 / 180 * M_PI,
        -12.075005 / 180 * M_PI,
        -83.663342 / 180 * M_PI,
```

```

        -15.641249 / 180 * M_PI,
        -89.140000 / 180 * M_PI,
        -28.328713 / 180 * M_PI
    };

    //关节运动到初始位置
    AuboRobot.rs_move_joint(rshd, initPos, true);

    //设置坐标系
    AuboRobot.CoordCalibrate baseCoord = new AuboRobot.CoordCalibrate();
    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    baseCoord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord,
        typeof(AuboRobot.CoordCalibrate));
    baseCoord.coordType = 0;

    //旋转轴和旋转角度
    AuboRobot.MoveRotateAxis axis = new AuboRobot.MoveRotateAxis();
    IntPtr pt_axis = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.MoveRotateAxis)));
    axis = (AuboRobot.MoveRotateAxis)Marshal.PtrToStructure(pt_axis, typeof(AuboRobot.MoveRotateAxis));
    axis.rotateAxis[0] = 1;
    axis.rotateAxis[1] = 0;
    axis.rotateAxis[2] = 0;
    double rotateAngle = 5.0 * M_PI / 180;

    //工具参数
    AuboRobot.ToolInEndDesc tool = new AuboRobot.ToolInEndDesc();
    tool.cartPos.x = 0.2;
    tool.cartPos.y = 0.1;
    tool.cartPos.z = 0.1;
    tool.orientation.w = 1;
    tool.orientation.x = 0;
    tool.orientation.y = 0;
    tool.orientation.z = 0;

    AuboRobot.rs_set_tool_kinematics_param(rshd, ref tool);

    //旋转运动
    int ret = AuboRobot.rs_move_rotate(rshd, ref baseCoord, ref axis, rotateAngle, true);
    if (ret == RS_SUCC)
    {

```

```

        Console.Out.WriteLine("旋转运动成功! ");
    }
    else
    {
        Console.Error.WriteLine("旋转运动失败! 错误码: {0}", ret);
    }
}

```

示例 3：法兰盘中心在用户坐标系下旋转

本示例是法兰盘中心在用户坐标系下做旋转运动。

代码如下：

```

static void rotateMoveExample3(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 30 * M_PI / 180;
    jointMaxAcc[1] = 30 * M_PI / 180;
    jointMaxAcc[2] = 30 * M_PI / 180;
    jointMaxAcc[3] = 30 * M_PI / 180;
    jointMaxAcc[4] = 30 * M_PI / 180;
    jointMaxAcc[5] = 30 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 30 * M_PI / 180;
    jointMaxVelc[1] = 30 * M_PI / 180;
    jointMaxVelc[2] = 30 * M_PI / 180;
    jointMaxVelc[3] = 30 * M_PI / 180;
    jointMaxVelc[4] = 30 * M_PI / 180;
    jointMaxVelc[5] = 30 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动最大加速度
    double lineMaxAcc = 0.2;
    AuboRobot.rs_set_global_end_max_line_acc(rshd, lineMaxAcc);

    //设置末端型运动最大速度
}

```

```

double lineMaxVelc = 0.2;
AuboRobot.rs_set_global_end_max_line_velc(rshd, lineMaxVelc);

//初始位置
double[] initPos = {
    173.108713 / 180 * M_PI,
    -12.075005 / 180 * M_PI,
    -83.663342 / 180 * M_PI,
    -15.641249 / 180 * M_PI,
    -89.140000 / 180 * M_PI,
    -28.328713 / 180 * M_PI
};

//关节运动到初始位置
AuboRobot.rs_move_joint(rshd, initPos, true);

//用户坐标系
AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
user_coord.coordType = 2;
user_coord.methods = 0;
user_coord.jointPara[0].jointRadian[0] = -14.717415 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[1] = -9.423585 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[2] = -74.757117 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[3] = 22.657165 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[4] = -84.449238 * M_PI / 180.0;
user_coord.jointPara[0].jointRadian[5] = -14.946778 * M_PI / 180.0;

user_coord.jointPara[1].jointRadian[0] = 24.363602 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[1] = 7.233866 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[2] = -55.242389 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[3] = 22.464115 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[4] = -86.957003 * M_PI / 180.0;
user_coord.jointPara[1].jointRadian[5] = 24.171302 * M_PI / 180.0;

user_coord.jointPara[2].jointRadian[0] = -12.492231 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[1] = 0.324244 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[2] = -65.331736 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[3] = 22.120198 * M_PI / 180.0;
user_coord.jointPara[2].jointRadian[4] = -84.531432 * M_PI / 180.0;

```

```

user_coord.jointPara[2].jointRadian[5] = -12.712789 * M_PI / 180.0;

//坐标系的工具参数
AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
());
tooluser_coord.cartPos.x = 0.1;
tooluser_coord.cartPos.y = 0.1;
tooluser_coord.cartPos.z = 0.2;
tooluser_coord.orientation.w = 1;
tooluser_coord.orientation.x = 0;
tooluser_coord.orientation.y = 0;
tooluser_coord.orientation.z = 0;

user_coord.toolDesc = tooluser_coord;

//旋转轴和旋转角度
AuboRobot.MoveRotateAxis axis = new AuboRobot.MoveRotateAxis();
IntPtr pt_axis = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.Move
eRotateAxis)));
axis = (AuboRobot.MoveRotateAxis)Marshal.PtrToStructure(pt_axis, typeof(Au
boRobot.MoveRotateAxis));
axis.rotateAxis[0] = 1;
axis.rotateAxis[1] = 0;
axis.rotateAxis[2] = 0;
double rotateAngle = 5.0 * M_PI / 180;

//旋转运动
int ret = AuboRobot.rs_move_rotate(rshd, ref user_coord, ref axis, rotateAngl
e, true);
if (ret == RS_SUCC)
{
    Console.Out.WriteLine("旋转运动成功！");
}
else
{
    Console.Error.WriteLine("旋转运动失败！ 错误码： {0}", ret);
}
}

```

示例 4：末端工具在用户坐标系下旋转

本示例是末端工具在用户坐标系下做旋转运动。

代码如下：

```
static void rotateMoveExample4(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 30 * M_PI / 180;
    jointMaxAcc[1] = 30 * M_PI / 180;
    jointMaxAcc[2] = 30 * M_PI / 180;
    jointMaxAcc[3] = 30 * M_PI / 180;
    jointMaxAcc[4] = 30 * M_PI / 180;
    jointMaxAcc[5] = 30 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 30 * M_PI / 180;
    jointMaxVelc[1] = 30 * M_PI / 180;
    jointMaxVelc[2] = 30 * M_PI / 180;
    jointMaxVelc[3] = 30 * M_PI / 180;
    jointMaxVelc[4] = 30 * M_PI / 180;
    jointMaxVelc[5] = 30 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动最大加速度
    double lineMaxAcc = 0.2;
    AuboRobot.rs_set_global_end_max_line_acc(rshd, lineMaxAcc);

    //设置末端型运动最大速度
    double lineMaxVelc = 0.2;
    AuboRobot.rs_set_global_end_max_line_velc(rshd, lineMaxVelc);

    //初始位置
    double[] initPos = {
        173.108713 / 180 * M_PI,
        -12.075005 / 180 * M_PI,
        -83.663342 / 180 * M_PI,
```

```

        -15.641249 / 180 * M_PI,
        -89.140000 / 180 * M_PI,
        -28.328713 / 180 * M_PI
    };

    //关节运动到初始位置
    AuboRobot.rs_move_joint(rshd, initPos, true);

    //用户坐标系
    AuboRobot.CoordCalibrate user_coord = new AuboRobot.CoordCalibrate();
    IntPtr pt_user_coord = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.CoordCalibrate)));
    user_coord = (AuboRobot.CoordCalibrate)Marshal.PtrToStructure(pt_user_coord, typeof(AuboRobot.CoordCalibrate));
    user_coord.coordType = 2;
    user_coord.methods = 0;
    user_coord.jointPara[0].jointRadian[0] = -14.717415 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[1] = -9.423585 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[2] = -74.757117 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[3] = 22.657165 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[4] = -84.449238 * M_PI / 180.0;
    user_coord.jointPara[0].jointRadian[5] = -14.946778 * M_PI / 180.0;

    user_coord.jointPara[1].jointRadian[0] = 24.363602 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[1] = 7.233866 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[2] = -55.242389 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[3] = 22.464115 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[4] = -86.957003 * M_PI / 180.0;
    user_coord.jointPara[1].jointRadian[5] = 24.171302 * M_PI / 180.0;

    user_coord.jointPara[2].jointRadian[0] = -12.492231 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[1] = 0.324244 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[2] = -65.331736 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[3] = 22.120198 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[4] = -84.531432 * M_PI / 180.0;
    user_coord.jointPara[2].jointRadian[5] = -12.712789 * M_PI / 180.0;

    //坐标系的工具参数
    AuboRobot.ToolInEndDesc tooluser_coord = new AuboRobot.ToolInEndDesc
    ();
    tooluser_coord.cartPos.x = 0.1;
    tooluser_coord.cartPos.y = 0.1;
    tooluser_coord.cartPos.z = 0.2;
    tooluser_coord.orientation.w = 1;

```

```
tooluser_coord.orientation.x = 0;
tooluser_coord.orientation.y = 0;
tooluser_coord.orientation.z = 0;

user_coord.toolDesc = tooluser_coord;

//旋转轴和旋转角度
AuboRobot.MoveRotateAxis axis = new AuboRobot.MoveRotateAxis();
IntPtr pt_axis = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.MoveRotateAxis)));
axis = (AuboRobot.MoveRotateAxis)Marshal.PtrToStructure(pt_axis, typeof(AuboRobot.MoveRotateAxis));
axis.rotateAxis[0] = 1;
axis.rotateAxis[1] = 0;
axis.rotateAxis[2] = 0;
double rotateAngle = 5.0 * M_PI / 180;

//工具参数
AuboRobot.ToolInEndDesc tool = new AuboRobot.ToolInEndDesc();
tool.cartPos.x = 0.2;
tool.cartPos.y = 0.1;
tool.cartPos.z = 0.1;
tool.orientation.w = 1;
tool.orientation.x = 0;
tool.orientation.y = 0;
tool.orientation.z = 0;

AuboRobot.rs_set_tool_kinematics_param(rshd, ref tool);

//旋转运动
int ret = AuboRobot.rs_move_rotate(rshd, ref user_coord, ref axis, rotateAngle, true);
if (ret == RS_SUCC)
{
    Console.Out.WriteLine("旋转运动成功! ");
}
else
{
    Console.Error.WriteLine("旋转运动失败! 错误码: {0}", ret);
}
}
```


示例 5：在工具坐标系下旋转

本示例是在工具坐标系下做旋转运动。

代码如下：

```
static void rotateMoveExample5(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节运动最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 30 * M_PI / 180;
    jointMaxAcc[1] = 30 * M_PI / 180;
    jointMaxAcc[2] = 30 * M_PI / 180;
    jointMaxAcc[3] = 30 * M_PI / 180;
    jointMaxAcc[4] = 30 * M_PI / 180;
    jointMaxAcc[5] = 30 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节运动最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 30 * M_PI / 180;
    jointMaxVelc[1] = 30 * M_PI / 180;
    jointMaxVelc[2] = 30 * M_PI / 180;
    jointMaxVelc[3] = 30 * M_PI / 180;
    jointMaxVelc[4] = 30 * M_PI / 180;
    jointMaxVelc[5] = 30 * M_PI / 180;
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动最大加速度
    double lineMaxAcc = 0.2;
    AuboRobot.rs_set_global_end_max_line_acc(rshd, lineMaxAcc);

    //设置末端型运动最大速度
    double lineMaxVelc = 0.2;
    AuboRobot.rs_set_global_end_max_line_velc(rshd, lineMaxVelc);

    //初始位置
    double[] initPos = {
        173.108713 / 180 * M_PI,
        -12.075005 / 180 * M_PI,
        -83.663342 / 180 * M_PI,
```

```

        -15.641249 / 180 * M_PI,
        -89.140000 / 180 * M_PI,
        -28.328713 / 180 * M_PI
    };

    //关节运动到初始位置
    AuboRobot.rs_move_joint(rshd, initPos, true);

    //设置末端坐标系
    AuboRobot.CoordCalibrate userCoord = new AuboRobot.CoordCalibrate();
    userCoord.coordType = 1;

    AuboRobot.ToolInEndDesc toolUserCoord = new AuboRobot.ToolInEndDesc
0);
    toolUserCoord.cartPos.x = 0.2;
    toolUserCoord.cartPos.y = 0.1;
    toolUserCoord.cartPos.z = 0.1;
    toolUserCoord.orientation.w = 1;
    toolUserCoord.orientation.x = 0;
    toolUserCoord.orientation.y = 0;
    toolUserCoord.orientation.z = 0;
    userCoord.toolDesc = toolUserCoord;

    //旋转轴和旋转角度
    AuboRobot.MoveRotateAxis axis = new AuboRobot.MoveRotateAxis();
    IntPtr pt_axis = Marshal.AllocHGlobal(Marshal.SizeOf(typeof(AuboRobot.Move
eRotateAxis)));
    axis = (AuboRobot.MoveRotateAxis)Marshal.PtrToStructure(pt_axis, typeof(Au
boRobot.MoveRotateAxis));
    axis.rotateAxis[0] = 1;
    axis.rotateAxis[1] = 0;
    axis.rotateAxis[2] = 0;
    double rotateAngle = 5.0 * M_PI / 180;

    //旋转运动
    int ret = AuboRobot.rs_move_rotate(rshd, ref userCoord, ref axis, rotateAngl
e, true);
    if (ret == RS_SUCC)
    {
        Console.Out.WriteLine("旋转运动成功！");
    }
    else
    {
        Console.Error.WriteLine("旋转运动失败！ 错误码： {0}", ret);
    }

```

```

    }
}

```

4.12 轨迹运动

4.12.1 rs_move_track 函数

示例 1：圆运动

本示例是机械臂做轨迹运动之圆运动。

代码如下：

```

static void trackMoveExample1(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节型运动的最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[1] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[2] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[3] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[4] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节型运动的最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[1] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[2] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[3] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[4] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动的最大加速度
    double endMoveMaxAcc;
    endMoveMaxAcc = 0.2; //单位米每秒
    AuboRobot.rs_set_global_end_max_line_acc(rshd, endMoveMaxAcc);
}

```

```
AuboRobot.rs_set_global_end_max_angle_acc(rshd, endMoveMaxAcc);

//设置末端型运动的最大速度
double endMoveMaxVelc;
endMoveMaxVelc = 0.2; //单位米每秒
AuboRobot.rs_set_global_end_max_line_velc(rshd, endMoveMaxVelc);
AuboRobot.rs_set_global_end_max_angle_velc(rshd, endMoveMaxVelc);

//准备点
double[] jointAngle = new double[6];
jointAngle[0] = -0.000003;
jointAngle[1] = -0.127267;
jointAngle[2] = -1.321122;
jointAngle[3] = 0.376934;
jointAngle[4] = -1.570796;
jointAngle[5] = -0.000008;

//关节运动到准备点
int ret = AuboRobot.rs_move_joint(rshd, jointAngle, true);
if (ret != RS_SUCC)
{
    Console.Error.WriteLine("关节运动到准备点失败。 错误号: {0}", ret);
}

//添加圆轨迹路点
jointAngle[0] = -0.000003;
jointAngle[1] = -0.127267;
jointAngle[2] = -1.321122;
jointAngle[3] = 0.376934;
jointAngle[4] = -1.570796;
jointAngle[5] = -0.000008;
AuboRobot.rs_add_waypoint(rshd, jointAngle);

jointAngle[0] = -0.211675;
jointAngle[1] = -0.325189;
jointAngle[2] = -1.466753;
jointAngle[3] = 0.429232;
jointAngle[4] = -1.570794;
jointAngle[5] = -0.211680;
AuboRobot.rs_add_waypoint(rshd, jointAngle);

jointAngle[0] = -0.037186;
jointAngle[1] = -0.224307;
jointAngle[2] = -1.398285;
```

```

jointAngle[3] = 0.396819;
jointAngle[4] = -1.570796;
jointAngle[5] = -0.037191;
AuboRobot.rs_add_waypoint(rshd, jointAngle);

//设置圆的圈数
AuboRobot.rs_set_circular_loop_times(rshd, 3);

//开始轨迹运动
ret = AuboRobot.rs_move_track(rshd, ARC_CIR, true);
if (RS_SUCC != ret)
{
    Console.Error.WriteLine("圆轨迹运动失败。 错误号:{0}", ret);
}
else
{
    Console.Out.WriteLine("圆轨迹运动成功。");
}
}

```

示例 2：圆弧运动

本示例是机械臂做轨迹运动之圆弧运动。

代码如下：

```

static void trackMoveExample2(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节型运动的最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[1] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[2] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[3] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[4] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节型运动的最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 50.0 / 180.0 * M_PI;
}

```

```
jointMaxVelc[1] = 50.0 / 180.0 * M_PI;
jointMaxVelc[2] = 50.0 / 180.0 * M_PI;
jointMaxVelc[3] = 50.0 / 180.0 * M_PI;
jointMaxVelc[4] = 50.0 / 180.0 * M_PI;
jointMaxVelc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

//设置末端型运动的最大加速度
double endMoveMaxAcc;
endMoveMaxAcc = 0.2; //单位米每秒
AuboRobot.rs_set_global_end_max_line_acc(rshd, endMoveMaxAcc);
AuboRobot.rs_set_global_end_max_angle_acc(rshd, endMoveMaxAcc);

//设置末端型运动的最大速度
double endMoveMaxVelc;
endMoveMaxVelc = 0.2; //单位米每秒
AuboRobot.rs_set_global_end_max_line_velc(rshd, endMoveMaxVelc);
AuboRobot.rs_set_global_end_max_angle_velc(rshd, endMoveMaxVelc);

//准备点
double[] jointAngle = new double[6];
jointAngle[0] = -0.000003;
jointAngle[1] = -0.127267;
jointAngle[2] = -1.321122;
jointAngle[3] = 0.376934;
jointAngle[4] = -1.570796;
jointAngle[5] = -0.000008;

//关节运动到准备点
int ret = AuboRobot.rs_move_joint(rshd, jointAngle, true);
if (ret != RS_SUCC)
{
    Console.Error.WriteLine("关节运动到准备点失败。    错误号:{0}", ret);
}

//添加圆弧轨迹路点
jointAngle[0] = -0.000003;
jointAngle[1] = -0.127267;
jointAngle[2] = -1.321122;
jointAngle[3] = 0.376934;
jointAngle[4] = -1.570796;
jointAngle[5] = -0.000008;
AuboRobot.rs_add_waypoint(rshd, jointAngle);
```

```
jointAngle[0] = 0.200000;  
jointAngle[1] = -0.127267;  
jointAngle[2] = -1.321122;  
jointAngle[3] = 0.376934;  
jointAngle[4] = -1.570794;  
jointAngle[5] = -0.000008;  
AuboRobot.rs_add_waypoint(rshd, jointAngle);  
  
jointAngle[0] = 0.600000;  
jointAngle[1] = -0.127267;  
jointAngle[2] = -1.321122;  
jointAngle[3] = 0.376934;  
jointAngle[4] = -1.570796;  
jointAngle[5] = -0.000008;  
AuboRobot.rs_add_waypoint(rshd, jointAngle);  
  
//设置圆弧  
AuboRobot.rs_set_circular_loop_times(rshd, 0);  
  
//开始轨迹运动  
ret = AuboRobot.rs_move_track(rshd, ARC_CIR, true);  
if (RS_SUCC != ret)  
{  
    Console.Error.WriteLine("圆弧轨迹运动失败。 错误号:{0}", ret);  
}  
else  
{  
    Console.Out.WriteLine("圆弧轨迹运动成功。");  
}  
}
```

示例 3: MOVEP

本示例是机械臂做轨迹运动之 MOVEP 运动。

代码如下：

```
static void trackMoveExample3(UInt16 rshd)
{
    //初始化运动属性
    AuboRobot.rs_init_global_move_profile(rshd);

    //设置关节型运动的最大加速度
    double[] jointMaxAcc = new double[6];
    jointMaxAcc[0] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[1] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[2] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[3] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[4] = 50.0 / 180.0 * M_PI;
    jointMaxAcc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxacc(rshd, jointMaxAcc);

    //设置关节型运动的最大速度
    double[] jointMaxVelc = new double[6];
    jointMaxVelc[0] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[1] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[2] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[3] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[4] = 50.0 / 180.0 * M_PI;
    jointMaxVelc[5] = 50.0 / 180.0 * M_PI;    //接口要求单位是弧度
    AuboRobot.rs_set_global_joint_maxvelc(rshd, jointMaxVelc);

    //设置末端型运动的最大加速度
    double endMoveMaxAcc;
    endMoveMaxAcc = 0.2; //单位米每秒
    AuboRobot.rs_set_global_end_max_line_acc(rshd, endMoveMaxAcc);
    AuboRobot.rs_set_global_end_max_angle_acc(rshd, endMoveMaxAcc);

    //设置末端型运动的最大速度
    double endMoveMaxVelc;
    endMoveMaxVelc = 0.2; //单位米每秒
    AuboRobot.rs_set_global_end_max_line_velc(rshd, endMoveMaxVelc);
    AuboRobot.rs_set_global_end_max_angle_velc(rshd, endMoveMaxVelc);

    //准备点
    double[] jointAngle = new double[6];
```



```
jointAngle[0] = -0.000003;
jointAngle[1] = -0.127267;
jointAngle[2] = -1.321122;
jointAngle[3] = 0.376934;
jointAngle[4] = -1.570796;
jointAngle[5] = -0.000008;

//关节运动到准备点
int ret = AuboRobot.rs_move_joint(rshd, jointAngle, true);
if (ret != RS_SUCC)
{
    Console.Error.WriteLine("关节运动到准备点失败。 错误号:{0}", ret);
}

//添加 MOVEP 轨迹路点
jointAngle[0] = -0.000003;
jointAngle[1] = -0.127267;
jointAngle[2] = -1.321122;
jointAngle[3] = 0.376934;
jointAngle[4] = -1.570796;
jointAngle[5] = -0.000008;
AuboRobot.rs_add_waypoint(rshd, jointAngle);

jointAngle[0] = 0.200000;
jointAngle[1] = -0.127267;
jointAngle[2] = -1.321122;
jointAngle[3] = 0.376934;
jointAngle[4] = -1.570796;
jointAngle[5] = -0.000008;
AuboRobot.rs_add_waypoint(rshd, jointAngle);

jointAngle[0] = 0.600000;
jointAngle[1] = -0.127267;
jointAngle[2] = -1.321122;
jointAngle[3] = 0.376934;
jointAngle[4] = -1.570796;
jointAngle[5] = -0.000008;
AuboRobot.rs_add_waypoint(rshd, jointAngle);

//设置交融半径
AuboRobot.rs_set_blend_radius(rshd, 0.03);

//开始轨迹运动
ret = AuboRobot.rs_move_track(rshd, CARTESIAN_MOVEP, true);
```

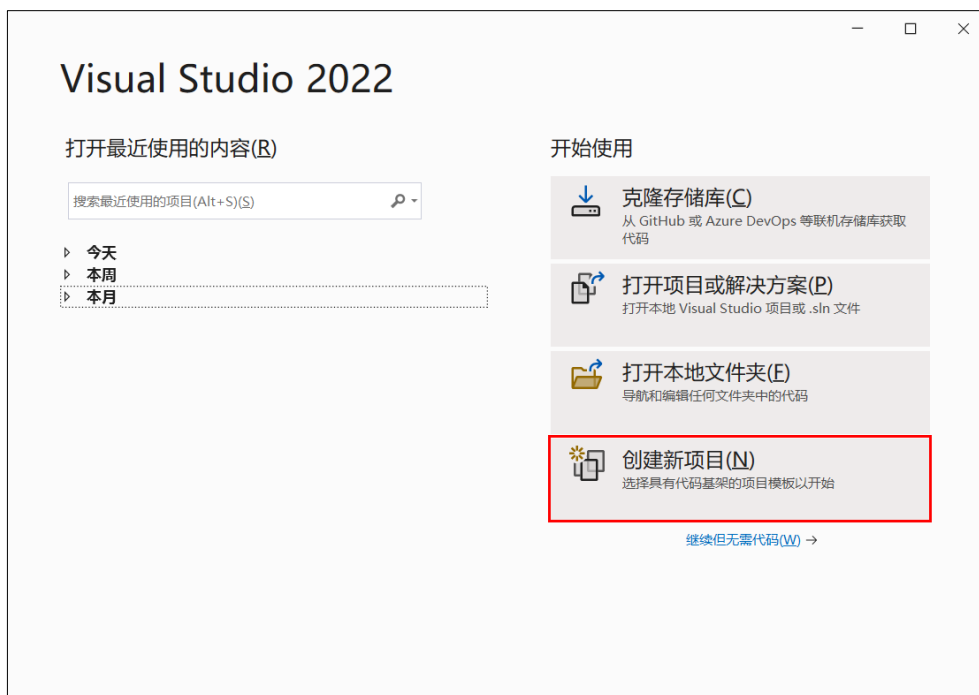
```
if (RS_SUCC != ret)
{
    Console.Error.WriteLine("MOVEP 轨迹运动失败。错误号:{0}", ret);
}
else
{
    Console.Out.WriteLine("MOVEP 轨迹运动成功。");
}
}
```

5 环境配置说明

注意：该环境配置说明是在 Windows 10 64 位操作系统里进行的。

5.1.1 新建工程

打开 Visual Studio，点击“创建新项目”。



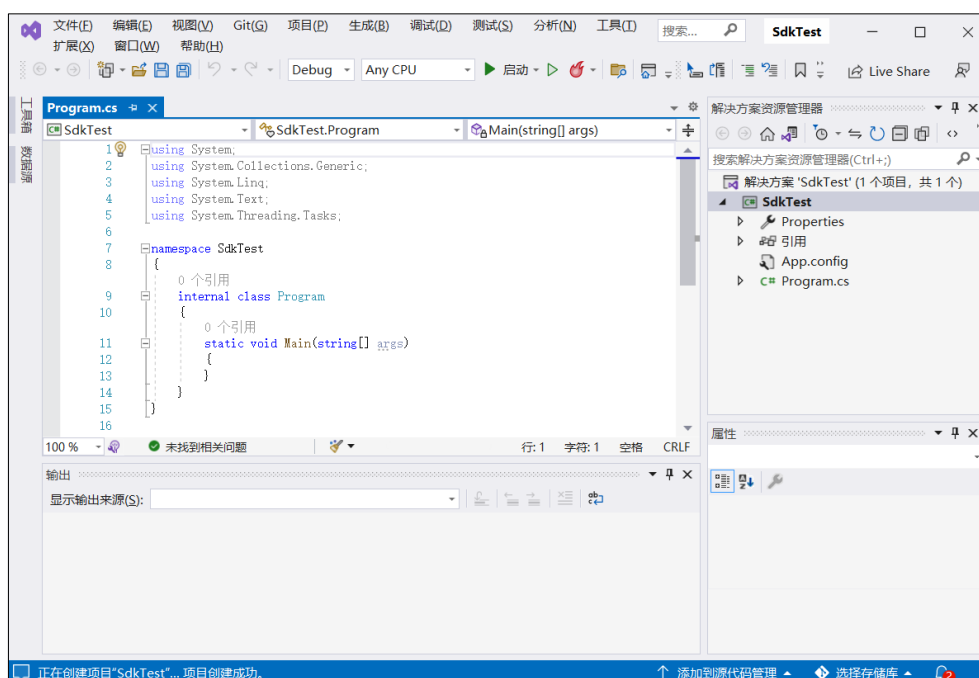
选择“C#”→选择“控制台应用(.NET Framework)”→点击“下一步”。



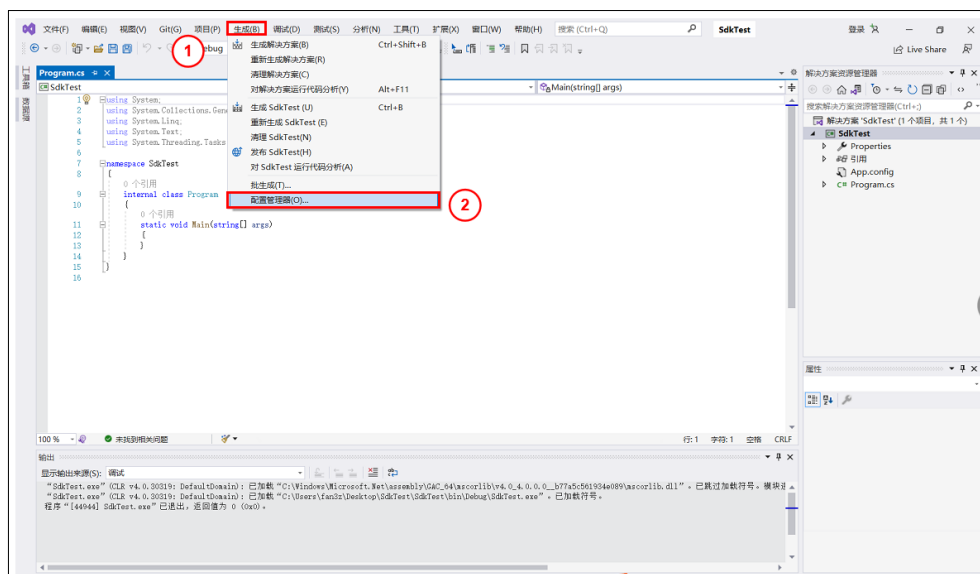
输入“项目名称”→点击“创建”。此处项目名称为《SdkTest》。



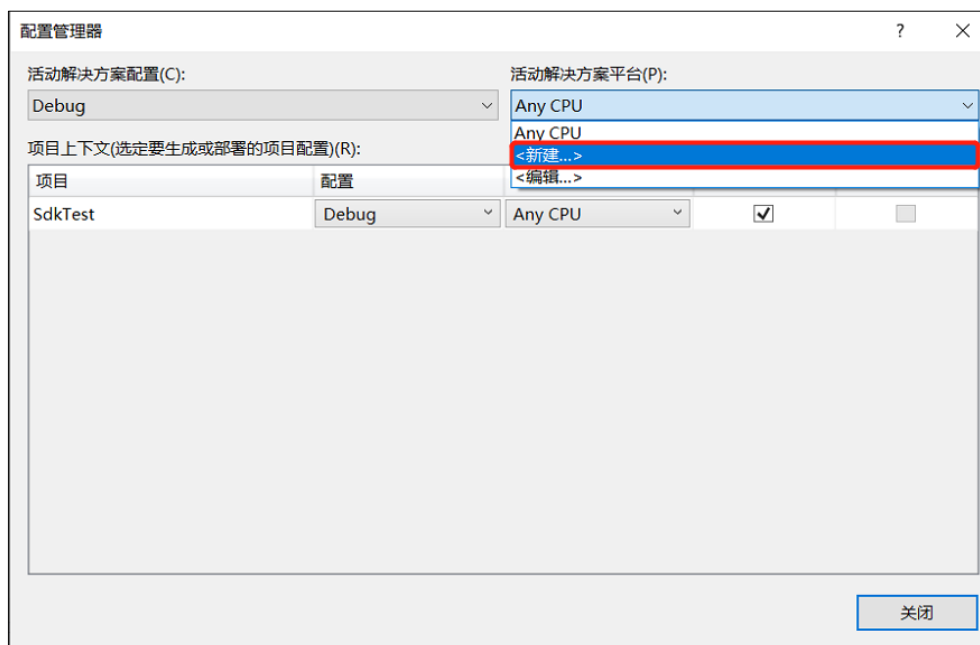
创建项目之后，界面如下。



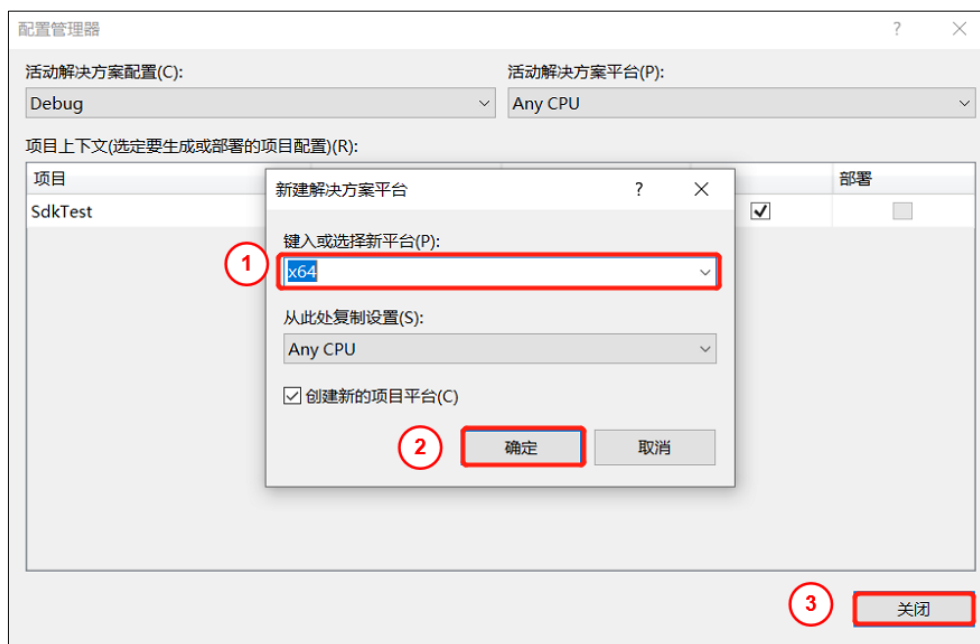
点击菜单栏中的“生成”→“配置管理器”。



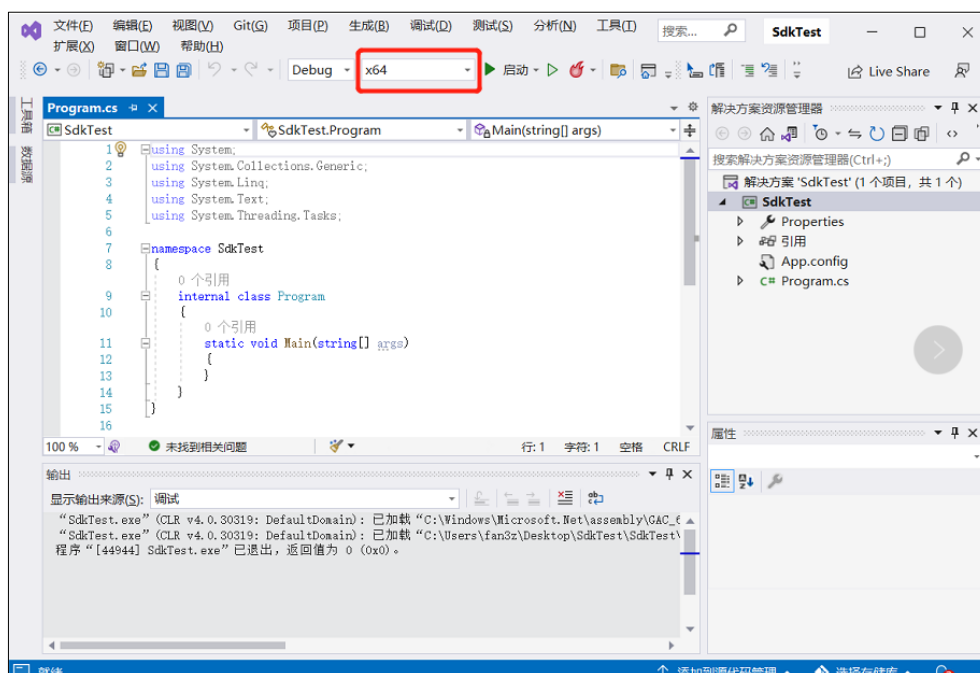
在“活动解决方案平台”中点击“新建”。



平台选择“x64”→点击“确定”→点击“关闭”。



如下图所示，配置管理器由“Any CPU”改为“x64”。



5.1.2 配置 DLL

将配置文件拷贝到 SdkTest/SdkTest/bin/x64/Debug 路径下。



5.1.3 编写并运行工程

在 Program.cs 中的代码如下：

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

using System.Runtime.InteropServices;

namespace SdkTest
{
    class Program
    {
        const string service_interface_dll = "serviceinterface2.dll";
        const string ip = "192.168.219.132";
        const int port = 8899;
        const int RS_SUCC = 0;
        static UInt16 rshd = 0xffff;

        //初始化机械臂控制库
        [DllImport(service_interface_dll, EntryPoint = "rs_initialize", CharSet = CharSet.Auto, CallingConvention = CallingConvention.Cdecl)]
        public static extern int rs_initialize();
    }
}
```

```

//反初始化机械臂控制库
[DllImport(service_interface_dll, EntryPoint = "rs_uninitialize", CharSet =
CharSet.Auto, CallingConvention = CallingConvention.Cdecl)]
public static extern int rs_uninitialize();

//创建机械臂控制上下文句柄
[DllImport(service_interface_dll, EntryPoint = "rs_create_context", CharSe
t = CharSet.Auto, CallingConvention = CallingConvention.Cdecl)]
public static extern int rs_create_context(ref UInt16 rshd);

//注销机械臂控制上下文句柄
[DllImport(service_interface_dll, EntryPoint = "rs_destory_context", CharS
et = CharSet.Auto, CallingConvention = CallingConvention.Cdecl)]
public static extern int rs_destory_context(UInt16 rshd);

//链接机械臂服务器
[DllImport(service_interface_dll, EntryPoint = "rs_login", CharSet = Char
Set.Auto, CallingConvention = CallingConvention.Cdecl)]
public static extern int rs_login(UInt16 rshd, [MarshalAs(UnmanagedType
e.LPStr)] string addr, int port);

static void Main(string[] args)
{
    //初始化机械臂控制库
    int result = rs_initialize();
    if (result == RS_SUCC)
    {
        Console.Out.WriteLine("机械臂初始化成功! ");
    }
    else
    {
        Console.Error.WriteLine("机械臂初始化失败! 错误码: {0}", re
sult);
    }

    //创建机械臂控制上下文句柄
    result = rs_create_context(ref rshd);
    if (result == RS_SUCC)
    {
        Console.Out.WriteLine("创建上下文句柄成功! ");
    }
    else

```



```

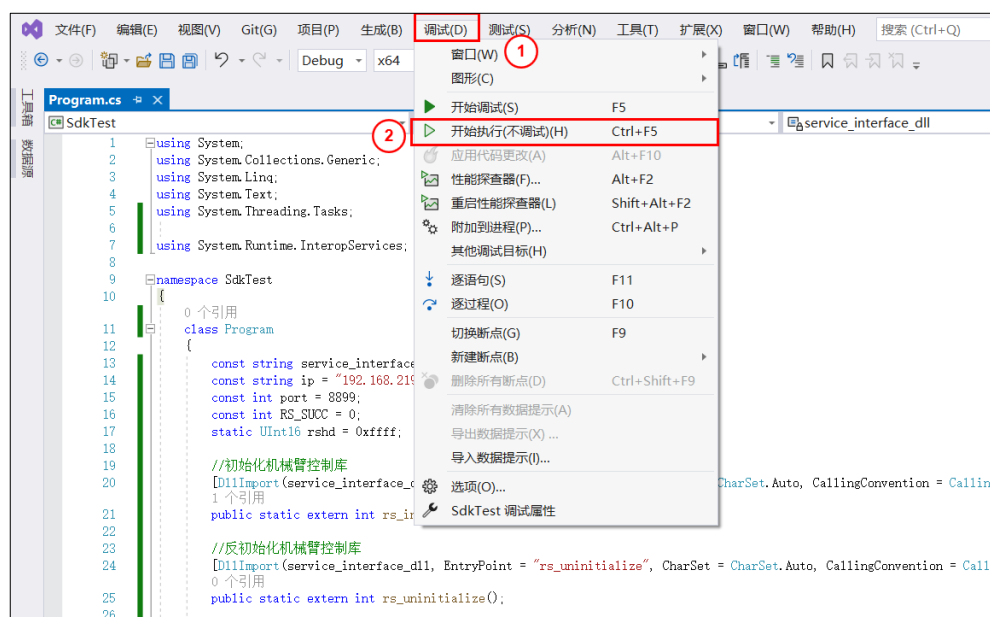
    {
        Console.Error.WriteLine("创建上下文句柄失败！错误码：{0}",
result);
    }

    //机械臂登录
    result = rs_login(rshd, ip, port);
    if (result == RS_SUCC)
    {
        Console.Out.WriteLine("登录成功！");
    }
    else
    {
        Console.Error.WriteLine("登录失败！错误码： {0}", result);
    }
}
}
}

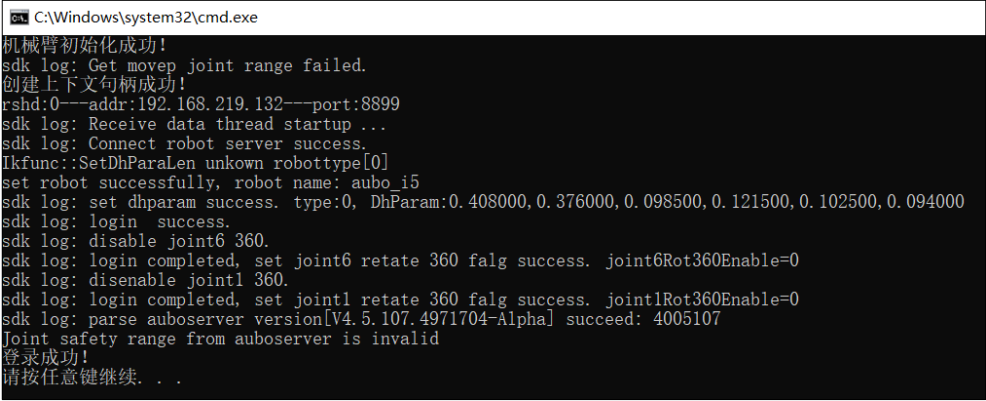
```

修改代码中的 IP 地址，连接 aobo 虚拟机或者真实机械臂。

菜单栏“调试”→“开始执行（不调试）”，或者“Ctrl+F5”来运行程序。



控制台中打印出“登录成功！”，说明与机械臂服务器通信成功，即环境配置成功。



```
C:\Windows\system32\cmd.exe
机械臂初始化成功!
sdk log: Get movep joint range failed.
创建上下文句柄成功!
rshd:0---addr:192.168.219.132---port:8899
sdk log: Receive data thread startup ...
sdk log: Connect robot server success.
Iikfunc::SetDhParaLen unkown robottype[0]
set robot successfully, robot name: aubo_i5
sdk log: set dhparam success. type:0, DhParam:0.408000, 0.376000, 0.098500, 0.121500, 0.102500, 0.094000
sdk log: login success.
sdk log: disable joint6 360.
sdk log: login completed, set joint6 retate 360 falg success. joint6Rot360Enable=0
sdk log: disenable joint1 360.
sdk log: login completed, set joint1 retate 360 falg success. joint1Rot360Enable=0
sdk log: parse auboserver version[V4.5.107.4971704-Alpha] succeed: 4005107
Joint safety range from auboserver is invalid
登录成功!
请按任意键继续. . .
```