二次开发说明书

4.3

中科新松有限公司

2025 年 09 月 26 日

Contents

CHAPTER 1

远程控制 APIV2.9

1.1 简介

此函数手册适用于新松协作机器人远程调用使用。

平台支持：Windows、Linux

语言支持：C++、Python、C#

测试环境：Windows(C++ VC14)

Windows(Python python3)

Linux(C++ ubuntu16.04 gcc5.4)

Linux(Python ubuntu16.04 python3)

适用 Core 版本：V3.4.1 及以上

1.2 数据类型和变量 (以 C++ 为例)

枚举变量的含义：

机器人状态信息：

```
enum StateRobot {
 SR_Start = 0, // 机器人启动
 SR_Initialize = 1, // 机器人初始化
 SR_Logout = 2, // 机器人登出，暂未使用
 SR_Login = 3, // 机器人登陆，暂未使用
 SR_PowerOff = 4, // 机器人下电
 SR_Disable = 5, // 机器人下使能
```

```
 SR_Enable = 6, // 机器人上使能
 SR_Update=7 // 机器人更新
};
```

程序状态信息:

```
enum StateProgram {
 SP_Stopped = 0, // 程序停止
 SP_Stopping = 1, // 程序正在停止中
 SP_Running = 2, // 程序正在运行
 SP_Paused = 3, // 程序已经暂停
 SP_Pausing = 4, // 程序暂停中
 SP_TaskRuning = 5 // 手动示教任务执行中
};
```

机器人操作模式信息:

```
enum OperationMode {
 kManual = 0, // 手动模式
 kAuto = 1, // 自动模式
 kRemote = 2 // 远程模式
};
```

任务状态信息:

```
enum TaskState {
 ST_Idle = 0, // 任务未执行
 ST_Running = 1, // 任务正在执行
 ST_Paused = 2, // 任务已经暂停
 ST_Stopped = 3, // 任务已经停止
 ST_Finished = 4, // 任务已经正常执行完成, 唯一表示任务正常完成（任务已经结束）
 ST_Interrupt = 5, // 任务被中断（任务已经结束）
 ST_Error = 6, // 任务出错（任务已经结束）
 ST_Illegal = 7, // 任务非法, 当前状态下任务不能执行（任务已经结束）
 ST_ParameterMismatch = 8 // 任务参数错误（任务已经结束）
};
```

安全控制器状态信息:

```
enum SafetyState {
```

```
    SS_INIT = 0, // 初始化
    SS_WAIT = 2, // 等待
    SS_CONFIG = 3, // 配置模式
    SS_POWER_OFF = 4, // 下电状态
    SS_RUN = 5, // 正常运行状态
    SS_RECOVERY = 6, // 恢复模式
    SS_STOP2 = 7, // Stop2
    SS_STOP1 = 8, // Stop1
    SS_STOP0 = 9, // Stop0
    SS_MODEL = 10, // 模型配置状态
    SS_REDUCE = 12, // 缩减模式状态
    SS_BOOT = 13, // 引导
    SS_FAIL = 14, // 致命错误状态
    SS_UPDATE = 99 // 更新状态
};
```

结构体变量含义:

机器人相关信息:

```
struct RobotStatusList{
    std::vector<double> jointExpectPosition; // 目标关节位置
    std::vector<double> jointExpectVelocity; // 目标角速度
    std::vector<double> jointExpectAccelera; // 目标角加速度
    std::vector<double> jointActualPosition; // 实际关节位置
    std::vector<double> jointActualVelocity; // 实际角速度
    std::vector<double> jointActualAccelera; // 实际角加速度
    std::vector<double> jointActualCurrent; // 实际关节电流
    std::vector<double> jointTemperature; // 时间关节温度
    std::vector<double> driverTemperature; // 未使用
    std::vector<double> cartExpectPosition; // 目标末端位姿
    std::vector<double> cartExpectVelocity; // 目标末端速度
    std::vector<double> cartExpectAccelera; // 目标末端加速度
    std::vector<double> cartActualPosition; // 实际末端位姿
    std::vector<double> cartActualVelocity; // 实际末端速度
    std::vector<double> cartActualAccelera; // 实际末端加速度
    std::vector<bool> slaveReady; // 从站状态
```

```
 bool collision; // 是否发生碰撞
 int8_t collisionAxis; // 发生碰撞的关节
 bool emcStopSignal; // 未使用
 int8_t robotState; // 机器人状态
 int32_t robotError; // 机器人错误码
 std::vector<double> jointAuxiliaryPosition; // 关节辅助位置
 std::vector<double> jointDynamicTorque; // 关节动态扭矩
 std::vector<double> jointGravityTorque; // 关节重力扭矩
 std::vector<double> jointActualTorque; // 关节实际扭矩
 std::vector<double> jointExtraTorque; // 关节额外扭矩
};
```

IO 和寄存器相关信息:

```
struct IOStatusList{
 std::vector<double> analogCurrentOutputs; // 模拟电流输出
 std::vector<double> analogVoltageOutputs; // 模拟电压输出
 std::vector<double> analogCurrentInputs; // 模拟电流输入
 std::vector<double> analogVoltageInputs; // 模拟电压输入
 std::vector<bool> digitalInputs; // 通用数字输入
 std::vector<bool> digitalOutputs; // 通用数字输出
 std::vector<bool> toolIOIn; // 工具数字输入
 std::vector<bool> toolIOOut; // 工具数字输出
 std::vector<bool> toolButton; // 工具末端按键
 std::vector<bool> funRegisterInputs; // 功能寄存器输入
 std::vector<bool> funRegisterOutputs; // 功能寄存器输出
 std::vector<bool> boolRegisterInputs; // bool 寄存器输入
 std::vector<bool> boolRegisterOutputs; // bool 寄存器输出
 std::vector<int16_t> wordRegisterInputs; // word 寄存器输入
 std::vector<int16_t> wordRegisterOutputs; // word 寄存器输出
 std::vector<double> floatRegisterInputs; // float 寄存器输入
 std::vector<double> floatRegisterOutputs; // float 寄存器输出
};
```

机器人点动相关信息:

```
struct MoveJogTaskParams{
```

```
 int32_t jog_direction; // 运动方向, 0: 正方向, 1: 负方向
 int32_t jog_type; // 1: 空间点动, 2: 关节点动
 int32_t axis_num; // 关节索引号, 0--5
 double vel; // 速度百分比
 int32_t jog_coordinate; // 参考坐标系, 0: 世界, 1: 基座, 2: 工具, 3: 工件
 bool use_step; // 是否步进模式, true: 步进, false: 连续
 double step_jointValue; // 关节和空间 RX/RY/RZ 步进弧度, 单位: rad
 double step_cartvalue; // 空间 X/Y/Z 步进距离, 单位: m
};
```

可达性检测相关信息:

```
struct ReachabilityParams{
 bool result; // 可达性确认结果, true: 可达, false: 不可达
 int32_t index; // 不可达时不可达点的索引号
 int32_t singularity_type; // 奇异类型, 0: 不奇异, 1: 肩部奇异, 2: 肘部奇异, 3: 腕部奇异, 4: 超出空间
};
```

外部轴反馈信息:

```
struct EAxisInfo{
 std::string scheme_name; // 外部轴方案名称
 int32_t status; // 激活状态
 double pos; // 当前位置
 double vel; // 当前移动速度
 double acc; // 当前加速度
};
```

实时数据:

```
struct RealTimeData{
 std::vector<double> joint_pos_cmd; // 关节位置实时指令, 仅在实时控制模式为关节位置时生效
 std::vector<double> joint_vel_cmd; // 关节速度实时指令, 仅在实时控制模式为关节速度时生效
 std::vector<double> joint_torq_cmd; // 关节力矩控制指令, 预留
 std::vector<double> cart_pos_tool_wobj_cmd; // 笛卡尔位置实时指令, 对应当前机器人工具坐标系在工件坐标系下的位置, 仅在实时控制模式未笛卡尔位置时生效
```

```
 std::vector<double> cart_vel_tool_wobj_cmd; // 笛卡尔速度实时指令，对应当前机器人工具坐标系在工件坐标系下的速度，仅在实时控制模式未笛卡尔速度时生效
 std::vector<double> cart_ft_cmd; // 笛卡尔力和力矩控制指令，预留
 bool status; // 控制指令刷新状态，更新机器人控制指令时给 true
};
```

路径点和 OP 操作:

```
struct PointOP{
 std::vector<double> pos; // 笛卡尔位置
 OP op; // 该位置对应的 OP 操作
 double blend_time; // 融合时间
 double dwell_time; // 驻留时间
};
```

计算几何路径运动所需要的时间:

```
struct TrajTimeParams{
 std::vector<double> start; // 起点
 std::vector<double> target; // 终点
 std::vector<double> middle; // 中间点
 std::vector<double> qnear; // 目标点附近位置对应的关节角度
 int32_t type; // 运动类型 0:movej, 1:movej2, 2:movej_pose, 3:movej_pose2, 4:movel, 5:movec, 6:movetcp
 double vel; // 速度
 double acc; // 加速度
 bool circle; // 是否整圆
 int32_t scale; // 笛卡尔空间检查奇点尺度
 std::string tool; // 工具坐标系名称
 std::string wobj; // 工件坐标系名称
};
```

偏移路径点序列:

```
struct PathOffsetResult {
 std::vector< std::vector<double> > path_result; // 偏移之后的路径点序列
 int32_t status_code; // 任务状态信息，参考 TaskState
};
```

为了方便使用，上述的枚举类型在实际使用时为 int 类型，上述枚举类型仅提供使用时 int 值对应的状态信息。

c++ 函数中 vector<double> 类型对应 python 类型 list。

1.3 函数说明

1.3.1 实例连接控制

DucoCobot (string ip, unsigned int port)

函数说明:

创建机器人远程连接的实例。

参数说明:

ip : 远程连接机器人的 ip 地址。

port : 远程连接机器人的端口地址，固定为 7003。

返回值:

机器人实例。

---

open ()

函数说明:

打开机器人连接。

参数说明:

无。

返回值:

Int 类型，-1：打开失败；0：打开成功。

---

close ()

函数说明:

关闭机器人连接。

参数说明:

无。

返回值:

Int 类型，-1：关闭失败；0：关闭成功。

---

rpc_heartbeat (int time)

函数说明:

该指令用于确保机器人远程连接断开时，机器人自动产生一条 stop 指令以停止当前运动。使用该函数需要单独创建一个线程周期性调用，且在线程中新创建一个 DucoCobot 对象。不使用该函数时，远程连接断开后，机器人不再主动产生 stop 指令。

参数说明:

`time` : 心跳延时时间，单位 (ms)。

返回值:

无。

1.3.2 系统控制

当通讯断开连接，所有函数的返回值变为-1

---

power_on (bool block)

函数说明:

机器人上电。

参数说明:

`block` : 指令是否阻塞型指令，如果为 `false` 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 `id` 信息，用户可以调用 `get_noneblock_taskstate(id)` 函数查询当前任务的执行状态。

---

power_off (bool block)

函数说明:

机器人下电。

参数说明:

`block` : 指令是否阻塞型指令，如果为 `false` 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 `id` 信息，用户可以调用 `get_noneblock_taskstate(id)` 函数查询当前任务的执行状态。

---

enable (bool block)

函数说明:

机器人上使能。

参数说明:

`block` : 指令是否阻塞型指令，如果为 `false` 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

disable (bool block)

函数说明：

机器人下使能。

参数说明：

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

shutdown (bool block)

函数说明：

机器人关机。

参数说明：

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

restart (bool block)

函数说明：

机器人重启。

参数说明：

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

1.3.3 任务控制

stop(bool block)

函数说明:

停止所有任务。

参数说明:

block : 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

pause (bool block)

函数说明:

暂停所有任务。

参数说明:

block : 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

resume (bool block)

函数说明:

恢复所有暂停的任务。

参数说明:

block : 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

abort (bool block)

函数说明:

中止当前正在执行的运动任务。如果还提前预读取了下一条或多条运动指令进行融合，此时预读取的指令同样会被中止。

参数说明:

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

! 注意

由于运动类脚本调用大多使用阻塞型指令，因此任务控制指令需要在新建的线程中，重新实例化 DucoCobot(string ip,unsigned int port)，并在新的实例中调用任务控制函数。

1.3.4 脚本控制

run_program (string name, bool block)

函数说明:

运行程序脚本。

参数说明:

name ： 脚本程序名称。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

get_current_project (string &path)

函数说明:

获取当前工程的路径。

参数说明:

path ： 当前工程路径。

返回值:

无。

---

get_files_list (map<string, int> &fileslist, const string &path)

函数说明:

获取指定路径下的文件列表。

参数说明:

filelist ： 文件列表和类型; 0: 文件夹; 1: 文件。

path ： 当前工程路径。

返回值:

无。

示例:

```
string project_path( “” );
get_current_project(project_path);
map<string, int> file_map;
get_files_list(file_map, project_path + “program” );
```

1.3.5 移动控制

```
struct Op {
    char time_or_dist_1;
    char trig_io_1;
    bool trig_value_1;
    double trig_time_1;
    double trig_dist_1;
    string trig_event_1;
    char time_or_dist_2;
    char trig_io_2;
    bool trig_value_2;
    double trig_time_2;
    double trig_dist_2;
    string trig_event_2;
    char time_or_dist_3;
    char trig_io_3;
    bool trig_value_3;
    double trig_time_3;
    double trig_dist_3;
    string trig_event_3;
}
```

特殊类型说明:

该参数类型用于控制机械臂在移动过程中控制机器人系统 IO 与寄存器输出。

参数说明：

time_or_dist_1：轨迹起始点触发类型，0：不启用，1：时间触发，2：距离触发。

trig_io_1：：轨迹起始点目标触发 IO 或寄存器索引号，定义如下：1-16 为控制柜通用 IO 输出 1-16，101-164 为 bool 寄存器输出 1-64，201-232 为 word 寄存器输出 1-32，301-332 为 float 寄存器输出 1-32，401-402 为机器人末端 IO 输出 1-2。

trig_value_1：：轨迹起始点目标触发 IO 或寄存器目标值。当目标触发类型为 IO 时，0 为低电平，非零为搞定平。当目标触发类型为寄存器时，trig_value_1 的值会根据目标寄存器数据类型进行强制转换后输出。

trig_time_1：轨迹起始点触发时间参数。当且仅当 time_or_dist_1 为时间触发时有效，代表轨迹运行多少时间长度触发 IO，单位 ms。

trig_dist_1：轨迹起始点触发距离参数。当 time_or_dist_1 为距离触发时，代表轨迹运行多少距离长度触发 IO，单位 m。

trig_event_1 ：轨迹起始点触发目标用户自定义事件名称。目标触发触发事件需要预先在机器人系统事件功能中定义且名称匹配。事件触发优先级高于 IO 或寄存器触发优先级，即若存在目标触发自定义事件，则不会触发目标 IO 或寄存器。若目标触发 IO 或寄存器，该参数写空。

time_or_dist_2：轨迹结束点触发类型，0：不启用，1：时间触发，2：距离触发。

trig_io_2：轨迹结束点目标触发 IO 或寄存器索引号，定义如下：1-16 为控制柜通用 IO 输出 1-16，101-164 为 bool 寄存器输出 1-64，201-232 为 word 寄存器输出 1-32，301-332 为 float 寄存器输出 1-32，401-402 为机器人末端 IO 输出 1-2。

trig_value_2：轨迹结束点目标触发 IO 或寄存器目标值。当目标触发类型为 IO 时，0 为低电平，非零为搞定平。当目标触发类型为寄存器时，trig_value_2 的值会根据目标寄存器数据类型进行强制转换后输出。。trig_time_2：轨迹结束点触发时间参数。对于 time_or_dist_2 为时间触发时有效。当 trig_time_2 > =0 时，代表轨迹运行剩余多少时间长度触发目标 IO 或寄存器，单位 ms；当 trig_time_2 < 0 时，代表代表轨迹运行结束后多少时间长度后触发目标 IO 与寄存器。

trig_dist_2：轨迹结束点触发距离参数。对于 time_or_dist_2 为距离触发时有效，当 trig_dist_2 > =0 时，代表轨迹运行剩余多少距离长度触发 IO，单位 m；当 trig_dist _2 < 0 时，代表代表轨迹运行结束后多少距离长度后触发目标 IO 与寄存器。

trig_event_2：轨迹结束点触发目标用户自定义事件名称。目标触发触发事件需要预先在机器人系统事件功能中定义且名称匹配。事件触发优先级高于 IO 或寄存器触发优先级，即若存在目标触发自定义事件，则不会触发目标 IO 或寄存器。若目标触发 IO 或寄存器，该参数写空。

time_or_dist_3：轨迹暂停或停止触发类型，0：不启用，1：时间触发。

trig_io_3：轨迹触发目标触发 IO 或寄存器索引号，定义如下：1-16 为控制柜通用 IO 输出 1-16，101-164 为 bool 寄存器输出 1-64，201-232 为 word 寄存器输出 1-32，301-332 为 float 寄存器输出 1-32，401-402 为机器人末端 IO 输出 1-2。

trig_value_3: 轨迹触发目标触发 IO 或寄存器目标值。当目标触发类型为 IO 时，0 为低电平，非零为搞定平。当目标触发类型为寄存器时，trig_value_2 的值会根据目标寄存器数据类型进行强制转换后输出。

trig_time_3: 轨迹暂停或停止触发时间参数。对于 time_or_dist_3 为时间触发时有效。当 trig_time_3 > =0 时，代表轨迹运行剩余多少时间长度触发目标 IO 或寄存器，单位 ms；当 trig_time_3 < 0 时，代表代表轨迹运行结束后多少时间长度后触发目标 IO 与寄存器。

trig_dist_3: 当前暂停与停止触发不支持基于距离的触发方式，该参数无效。

trig_event_3: 轨迹暂停或停止触发目标用户自定义事件名称。目标触发触发事件需要预先在机器人系统事件功能中定义且名称匹配。事件触发优先级高于 IO 或寄存器触发优先级，即若存在目标触发自定义事件，则不会触发目标 IO 或寄存器。若目标触发 IO 或寄存器，该参数写空。

---

movej2 (vector<double> joints_list, double v, double a, double rad, bool block, Op op = op_, bool def_acc = true)

函数说明:

该指令控制机械臂从当前状态，按照关节运动的方式移动到目标关节角状态。

参数说明:

joints_list: axis 数组对应 1-6 关节的目标关节角度，范围 [-2*PI, 2*PI]，单位 rad。

v : 最大关节角速度指令，范围 [0.01*PI/180, 1.25*PI]，单位 (rad/s)。

a : 最大关节角加速度指令，范围 [0.01*PI/180, ∞]，单位 (rad/s2)。

rad : 轨迹融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。产生融合运动时，机器人程序会根据融合预读取设定参数在运动指令执行过程中尝试提前获取后续运动函数。

block : 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op : 详见上方 Op 特殊类型说明 (距离触发无效)，可缺省参数。

def_acc : 是否使用默认加速度，默认为 false，可缺省参数。当开启默认加速度时，执行运动时最大加速度参数不在生效，系统将会根据实际工况计算机器人可以产生的最大加速度指令进行运动，从而提升节拍。

返回值:

当配置为阻塞，返回值代表当前任务结束时状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

movej_pose2 (vector<double> p, double v, double a, double r, vector<double> q_near, string tool, string wobj, bool block, Op op = op_, bool def_acc = true)

函数说明:

该指令控制机械臂从当前状态，按照各关节相位同步运动的方式移动到末端目标位置。

参数说明：

p ：目标机器人工具在参考机器人工件坐标系中的位姿，位置单位 m，姿态以 Rx、Ry、Rz 表示，范围 [-2*PI, 2*PI]，单位 (rad)。

v ：最大关节角速度，范围 [0.01*PI/180, 1.25*PI]，单位 (rad/s)。

a ：最大关节加速度，范围 [0.01*PI/180, ∞]，单位 (rad/s2)。

r ：轨迹融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。产生融合运动时，机器人程序会根据融合预读取设定参数在运动指令执行过程中尝试提前获取后续运动函数。

q_near：目标点附近位置对应的关节角度，用于确定逆运动学选解。

tool ：设置使用的工具的名称，为空时默认为当前使用的工具。

wobj ：设置使用的工件坐标系的名称，为空时默认为当前使用的工件坐标系。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op：详见上方 Op 特殊类型说明 (距离触发无效)，可缺省参数。

def_acc ：是否使用默认加速度，默认为 false，可缺省参数。当开启默认加速度时，执行运动时最大加速度参数不在生效，系统将会根据实际工况计算机器人可以产生的最大加速度指令进行运动，从而提升节拍。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

movel (vector<double> p, double v, double a, double rad, vector<double> q_near, string tool, string wobj, bool block, Op op = op_, bool def_acc = true)

函数说明：

该指令控制机械臂末端从当前状态按照直线路径移动到目标位姿。

参数说明：

p ：目标机器人工具在参考机器人工件坐标系中的位姿，位置单位 m，姿态以 Rx、Ry、Rz 表示，范围 [-2*PI, 2*PI]，单位 (rad)。

v ：最大末端线速度，范围 [0.01, 5]，单位 (m/s)。

a ：最大末端线加速度，范围 [0.01, ∞]，单位 (m/s2)。

rad ：轨迹融合半径，单位 (m)，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。产生融合运动时，机器人程序会根据融合预读取设定参数在运动指令执行过程中尝试提前获取后续运动函数。

qnear ：目标点附近位置对应的关节角度，用于校验机器人运动过程中逆运动学选解空间。

tool ： 设置使用的工具的名称，为空时默认为当前使用的工具。

wobj ： 设置使用的工件坐标系的名称，为空时默认为当前使用的工件坐标系。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op: 详见上方 Op 特殊类型说明，可缺省参数。

def_acc ： 是否使用默认加速度，默认为 false，可缺省参数。当开启默认加速度时，执行运动时最大加速度参数不在生效，系统将会根据实际工况计算机器人可以产生的最大加速度指令进行运动，从而提升节拍。

arc_rad ： 圆弧弧度，为 0 时运动到 p2 结束点，按照原始圆弧规划，其余情况按照数据圆弧角规划 (此模式姿态仅支持起点一致和受圆心约束)，单位: rad. 可缺省，默认为 0

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

movec (vector<double> p1, vector<double> p2, double v, double a, double r, int mode, vector<double> q_near, string tool, string wobj, bool block, Op op = op_, bool def_acc = true , double arc_rad = 0)

函数说明:

该指令控制机械臂做圆弧运动，起始点为当前位姿点，途径 p1 点，终点为 p2 点。

参数说明:

p1 ： 圆弧运动过程中任意机器人工具在参考机器人工件坐标系中的位姿中间点，位置单位 m，姿态以 Rx、Ry、Rz 表示范围 [-2*PI, 2*PI]，单位 (rad) 。

p2 ： 目标机器人工具在参考机器人工件坐标系中的位姿，位置单位 m，姿态以 Rx、Ry、Rz 表示范围 [-2*PI, 2*PI]，单位 (rad) 。

v ： 最大末端线速度，范围 [0.01, 5]，单位 (m/s)。

a ： 最大末端线加速度，范围 [0.01, ∞]，单位 (m/s2)。

r ： 轨迹融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。产生融合运动时，机器人程序会根据融合预读取设定参数在运动指令执行过程中尝试提前获取后续运动函数。

mode ： 姿态控制模式。

0: 姿态与终点保持一致，即机器人会以 p2 点的姿态为目标姿态，平滑运动到目标姿态。

1: 姿态与起点保持一致，即机器人会以开始执行 movec 函数时机器人末端工具坐标系在工件坐标系中的姿态为准，始终保持该姿态值。

2：姿态受圆心约束，即机器人会以开始执行 movec 函数时机器人末端工具坐标系与目标圆弧路径起点处切线方向间关系为参考，在圆弧运动过程中始终保持末端工具与圆弧实时运动所处位置切线方向参考关系。

qnear ： 目标点附近位置对应的关节角度，用于校验机器人运动过程中逆运动学选解空间。

tool ： 设置使用的工具的名称，为空时默认为当前使用的工具。

wobj ： 设置使用的工件坐标系的名称，为空时默认为当前使用的工件坐标系。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op ： 详见上方 Op 特殊类型说明，可缺省参数。

def_acc ： 是否使用默认加速度，默认为 false，可缺省参数。当开启默认加速度时，执行运动时最大加速度参数不在生效，系统将会根据实际工况计算机器人可以产生的最大加速度指令进行运动，从而提升节拍。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

move_circle (vector<double> p1, vector<double> p2, double v, double a, double rad, int mode, vector<double> qnear, string tool, string wobj, bool block, Op op = op_, bool def_acc = true)

函数说明：

该指令控制机械臂做圆周运动，起始点为当前位姿点，途径 p1 点和 p2 点

参数说明：

p1 圆周运动过程中任意机器人工具在参考机器人工件坐标系中的位姿中间点 1，位置单位 m，姿态以 Rx、Ry、Rz 表示范围 [-2*PI, 2*PI]，单位 (rad) 。

p2 ： 圆周运动过程中任意机器人工具在参考机器人工件坐标系中的位姿中间点 2，最终以机器人初始运动位置-p1-p2 的顺序决定最终整圆轨迹，位置单位 m，姿态以 Rx、Ry、Rz 表示范围 [-2*PI, 2*PI]，单位 (rad) 。

v ： 最大末端线速度，范围 [0.01, 5]，单位 (m/s)。

a ： 最大末端线加速度，范围 [0.01, ∞]，单位 (m/s2)。

rad ： 轨迹融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。产生融合运动时，机器人程序会根据融合预读取设定参数在运动指令执行过程中尝试提前获取后续运动函数。

mode ： 姿态控制模式。

1：姿态与起点保持一致，即机器人会以开始执行 movec 函数时机器人末端工具坐标系在工件坐标系中的姿态为准，始终保持该姿态值。

2: 姿态受圆心约束，即机器人会以开始执行 movec 函数时机器人末端工具坐标系与目标圆弧路径起点处切线方向间关系为参考，在圆弧运动过程中始终保持末端工具与圆弧实时运动所处位置切线方向参考关系。

qnear ： 目标点附近位置对应的关节角度，用于校验机器人运动过程中逆运动学选解空间。

tool ： 设置使用的工具的名称，默认为当前使用的工具。

wobj ： 设置使用的工件坐标系的名称，默认为当前使用的工件坐标系。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回，默认为阻塞。

op ： 详见上方 Op 特殊类型说明，可缺省参数。

def_acc ： 是否使用默认加速度，默认为 false，可缺省参数。当开启默认加速度时，执行运动时最大加速度参数不在生效，系统将会根据实际工况计算机器人可以产生的最大加速度指令进行运动，从而提升节拍。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

tcp_move (vector<double> pose_offset, double v, double a, double r, string tool, bool block, Op op = op_, bool def_acc = true)

函数说明:

该指令控制机械臂沿工具坐标系直线移动一个增量。

参数说明:

pose_offset:pose 数据类型，或者长度为 6 的 number 型数组，表示工具坐标系下的位姿偏移量。偏移量将会转换为齐次变换矩阵右乘于当前机器人末端位姿之上。

v: 最大末端线速度，范围 [0.01, 5]，单位 m/s，当 x、y、z 均为 0 时，线速度按比例换算成角速度。

a ： 最大末端线加速度，范围 [0.01, ∞]，单位 (m/s2)。

r ： 轨迹融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。产生融合运动时，机器人程序会根据融合预读取设定参数在运动指令执行过程中尝试提前获取后续运动函数。

tool ： 设置使用的工具的名称，为空时默认为当前使用的工具。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op ： 详见上方 Op 特殊类型说明，可缺省参数。

def_acc ： 是否使用默认加速度，默认为 false，可缺省参数。当开启默认加速度时，执行运动时最大加速度参数不在生效，系统将会根据实际工况计算机器人可以产生的最大加速度指令进行运动，从而提升节拍。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

tcp_move_2p (vector<double> p1, vector<double> p2, double v, double a, double r, string tool, string wobj , bool block, Op op = op_, bool def_acc = true)

函数说明:

该指令控制机器人沿工具坐标系直线移动一个增量，增量为 p1 与 p2 点之间的差，运动的目标点为：当前点 *p1-1*p2。

参数说明:

p1 ：表示工具坐标系下的位姿偏移量计算点 1。

p2 ：表示工具坐标系下的位姿偏移量计算点 2。

v：最大末端线速度，范围 [0.01, 5]，单位 (m/s)，当 x、y、z 均为 0 时，线速度按比例换算成角速度。

a ：最大末端线加速度，范围 [0.01, ∞]，单位 (m/s2)。

r ：轨迹融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。产生融合运动时，机器人程序会根据融合预读取设定参数在运动指令执行过程中尝试提前获取后续运动函数。

tool ：设置使用的工具的名称，默认为当前使用的工具。

wobj ：设置使用的工件坐标系的名称，默认为当前使用的工件坐标系。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op ：详见上方 Op 特殊类型说明，可缺省参数。

def_acc ：是否使用默认加速度，默认为 false，可缺省参数。当开启默认加速度时，执行运动时最大加速度参数不在生效，系统将会根据实际工况计算机器人可以产生的最大加速度指令进行运动，从而提升节拍。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

wobj_move(vector<double> pose_offset, double v ,double a, double r, string wobj, bool block, Op op = op _, bool def_acc = true)

函数说明:

该指令控制机械臂沿工件坐标系直线移动一个增量。

参数说明:

pose_offset:pose 数据类型，或者长度为 6 的 number 型数组，表示工件坐标系下的位姿偏移量。

v: 最大末端线速度，范围 [0.01, 5]，单位 m/s，当 x、y、z 均为 0 时，线速度按比例换算成角速度。

a : 最大末端线加速度，范围 [0.01, ∞]，单位 (m/s2)。

r : 轨迹融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。产生融合运动时，机器人程序会根据融合预读取设定参数在运动指令执行过程中尝试提前获取后续运动函数。

wobj : 设置使用的工件的名称，为空时默认为当前使用的工件。

block : 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op: 详见上方 Op 特殊类型说明，可缺省参数。

def_acc : 是否使用默认加速度，默认为 false，可缺省参数。当开启默认加速度时，执行运动时最大加速度参数不在生效，系统将会根据实际工况计算机器人可以产生的最大加速度指令进行运动，从而提升节拍。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

wobj_move_2p (vector<double> p1, vector<double> p2, double v, double a, double r, string tool, string wobj , bool block, Op op = op _, bool def_acc = true)

函数说明:

该指令控制机器人沿工件坐标系直线移动一个增量，增量为 p1 与 p2 点之间的位姿偏移在工件坐标系下的描述 $offset_{wobj}$，运动的目标点为: 当前点 * $offset_{wobj}$。

参数说明:

p1 : 表示工件坐标系下的位姿偏移量计算点 1。

p2 : 表示工件坐标系下的位姿偏移量计算点 2。

v: 最大末端线速度，范围 [0.01, 5]，单位 (m/s)，当 x、y、z 均为 0 时，线速度按比例换算成角速度。

a : 最大末端线加速度，范围 [0.01, ∞]，单位 (m/s2)。

rad : 轨迹融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。产生融合运动时，机器人程序会根据融合预读取设定参数在运动指令执行过程中尝试提前获取后续运动函数。

tool : 参考点使用的工具的名称，默认为当前使用的工具。

wobj : 参考点使用的工件坐标系的名称，默认为当前使用的工件坐标系。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回，默认为阻塞。

op：详见上方 Op 特殊类型说明，可缺省参数。

def_acc ：是否使用默认加速度，默认为 false，可缺省参数。当开启默认加速度时，执行运动时最大加速度参数不在生效，系统将会根据实际工况计算机器人可以产生的最大加速度指令进行运动，从而提升节拍。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

speedj (vector<double> joints_list, double a, int time, bool block)

函数说明：

该指令控制机械臂每个关节按照给定的速度一直运动，函数执行后会直接运行后续指令。运行 speedj 函数后，机械臂会持续运动并忽略后续运动指令，直到接收到 speed_stop() 函数后停止。

参数说明：

joints_list ：每个关节的速度，范围 [0.01*PI/180, 1.25*PI]，单位 (rad/s)。

a ：主导轴的关节加速度，范围 [0.01*PI/180, ∞]，单位 (rad/s2)。

time ：运行时间，到达时间后会停止运动，单位 (ms)。默认-1 表示一直运行。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

speedl (vector<double> pose_list, double a, int time, bool block)

函数说明：

该指令控制机械臂末端按照给定的速度一直运动，函数执行后会直接运行后续指令。运行 speedl 函数后，机械臂会持续运动并忽略后续运动指令，直到接收到 speed_stop() 函数后停止。

参数说明：

pose_list ：末端速度向量，线速度范围 [0.00001, 5]，线速度单位 (m/s)，角速度范围 (0, 1.25*PI]，角速度范围 (0, 2*PI]，角速度单位 (rad/s)。

a ：末端的线性加速度，范围 [0.00001, ∞]，单位 (rad/s2)。

time ：运行时间，到达时间会停止运动，单位 (ms)。默认-1 表示一直运行。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

speed_stop (bool block)

函数说明：

停止 speedj 及 speedl 函数的运动。

参数说明：

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

teach_mode (bool block)

函数说明：

该函数用于控制机器人进入牵引示教模式。

参数说明：

block：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

end_teach_mode (bool block)

函数说明：

该函数用于控制机器人退出牵引示教模式。

参数说明：

block：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

set_hand_teach_parameter (int32_t space, const std::vector<int32_t> & joint_scale, const std::vector<int32_t> & cart_scale, int32_t coord_type, const std::vector<bool> & direction)

函数说明:

设置牵引时的参数。

参数说明:

space: 牵引类型, 0: 关节空间, 1: 笛卡尔空间。

joint_scale: 关节柔顺度, 元素数量必须为机器人关节数量。

cart_scale: 笛卡尔柔顺度, 元素数量必须为 6 个。

coord_type: 笛卡尔示教参考坐标系类型, 0: 世界, 1: 基座, 2: 工具, 3: 工件。

direction: 牵引方向激活, true: 激活, false: 不激活。

返回值:

返回值代表当前任务结束时的状态。

---

replay (string name, int v, int mode)

函数说明:

该函数用于对记录的轨迹基于关节空间（或基于笛卡尔空间）复现。

参数说明:

name: 轨迹名称。

v: 轨迹速度,(系统设定速度的百分比%), 取值范围 (0,100]。

mode : 复现方式, 0: 基于关节空间, 1: 基于笛卡尔空间。

返回值:

无。

---

spline (vector< vector<double> > p_list, double v, double a, string tool, string wobj, bool block, Op op = op_, double r = 0, bool def_acc = true)

函数说明:

样条运动函数, 该指令控制机器人按照空间样条进行运动。

参数说明:

p_list: 在设置工件坐标系下的末端位姿列表, 最多不超过 50 个点, 格式如下:

[p1,p2,p3,…,pi,…]

其中 pi 为空间位姿, 如 [0.4,0.5,0.5,1.2,0.717,1.414]。

v ： 最大末端线速度，范围 [0.01, 5]，单位 (m/s)。

a ： 最大末端线加速度，范围 [0.01, ∞]，单位 (m/s2)。

tool ： 设置使用的工具的名称，为空时默认为当前使用的工具。

wobj ： 设置使用的工件坐标系的名称，为空默认为当前使用的工件坐标系

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op ： 详见上方 Op 特殊类型说明，可缺省参数。

r ： 轨迹融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。

def_acc ： 是否使用默认加速度，默认为 false，可缺省参数。当开启默认加速度时，执行运动时最大加速度参数不在生效，系统将会根据实际工况计算机器人可以产生的最大加速度指令进行运动，从而提升节拍。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

警告: 连续两条 spline 一起使用时，需要注意前一条的 spline 结束点与后一条的 spline 起始点，不能是同一个点。

---

spline_op (vector<PointOP> & pose_list, double v, double a, const string& tool, const string& wobj, bool block, const OP &op = op_, double r = 0, bool def_acc = false)

函数说明:

样条运动函数，控制机器人按照空间样条进行运动，在运动过程中触发对应点位的 OP 操作。

参数说明:

pose_list ： 在设置工件坐标系下的末端位姿列表，最多不超过 50 个点，格式如下:

{{p_1,op_1},{p_2,op_2},.....,{p_i,op_i}}

v ： 末端速度，范围 [0.00001, 5]，单位 (m/s)。

a ： 末端加速度，范围 [0.00001, ∞]，单位 (m/s2)。

tool ： 设置使用的工具的名称，为空时默认为当前使用的工具。

wobj ： 设置使用的工件坐标系的名称，为空默认为当前使用的工件坐标系

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op ： 详见上方 Op 特殊类型说明，可缺省参数。

r ：融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合，可缺省参数。

def_acc ：是否使用系统默认加速度，false 表示使用自定义的加速度值，true 表示使用系统自动规划的加速度值，可缺省，默认为 false。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

警告： 连续两条 spline 一起使用时，需要注意前一条的 spline 结束点与后一条的 spline 起始点，不能是同一个点。

---

servoj (vector<double> joints_list, double v, double a, bool block, double kp, double kd, double smooth_vel, double smooth_acc)

函数说明：

该指令控制机械臂从当前状态，按照机器人各关节独立运动的方式移动到目标关节角状态，运动过程中不考虑笛卡尔空间路径且不保证各关节间的运动规划相位同步。当在 servoj 运动过程中收到一条新的 servoj 运动指令时，机器人将会以当前运动规划状态向新的关节目标位置移动，舍弃之前的目标位置。

参数说明：

joints_list: axis 数组对应 1-6 关节的目标关节角度，范围 [-2*PI, 2*PI]，单位 rad。

v ：最大关节角速度，范围 [0.01*PI/180, 1.25*PI]，单位 (rad/s)。当用户使用 servoj 指令连续发送较为密集的关节位置指令时，此时默认用户对轨迹路径以及速度存在软实时插补（无法保证加速度及速度平滑连续），此时推荐用户将 v 设置为关节最大速度保护值，并通过调整连续离散点位的发送时间间隔来控制机器人最终运动速度。当用户使用 servoj 指令发送非连续目标位置以控制机器人到达不同的关节位置时，此时 servoj 指令作用同时涵盖路径插补以及运动学插补，此时推荐用户将 v 设置为目标关节产生运动的最大速度，直接通过 servoj 函数实现对最终运动速度指令的规划。

a ：最大关节加速度，范围 [0.01*PI/180, ∞]，单位 (rad/s2)。最大关节加速度的设定值方法与最大关节角速度设定方法雷同，根据用户使用 servoj 运动产生目标运动方案的不同存在一定差异性。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回，默认值 false，可缺省。

kp: 比例参数，默认值 200，可缺省，建议使用默认参数。

kd: 微分参数，默认值 25，可缺省，建议使用默认参数。

smooth_vel: 速度平滑系数，范围 [1,100]，当用户连续使用 servoj 指令发送连续离散关节位置点时，若由于发送的连续点位间经过时间差分不连续或发送周期抖动较大无法有效保证最终速度稳定性时，可以通过调整该平滑系数进行最终

运动速度指令的稳定性。平滑系数越大，对速度的平滑效果越强，同时会引入更大的跟随滞后。

smooth_acc：加速度平滑系数，范围 [0,1]，与速度平滑系数雷同，当用户连续使用 servoj 指令发送连续离散关节位置点时，若由于发送的连续点位间经过时间差分不连续或发送周期抖动较大无法有效保证最终加速度稳定性时，可以通过调整该平滑系数进行最终运动加速度指令的稳定性。平滑系数越大，对速度的加平滑效果越强，同时会引入更大的跟随滞后。

返回值：

返回值代表当前任务的id信息，用户可以调用get_noneblock_taskstate(id)函数查询当前任务的执行状态。

---

servoj_pose (vector<double> p, double v, double a, vector<double> qnear, string tool, string wobj, bool block, double kp, double kd, double smooth_vel, double smooth_acc)

函数说明：

该指令控制机械臂从当前状态，按照机器人各关节独立运动的方式移动到目标机器人末端在参考工件坐标系中的位姿，运动过程中不考虑笛卡尔空间路径且不保证各关节间的运动规划相位同步。当在 servoj_pose 运动过程中收到一条新的servoj_pose 运动指令时，机器人将会以当前运动规划状态向新的关节目标位置移动，舍弃之前的目标位置。

参数说明：

p ：目标工具在参考工件坐标系下的末端位姿，单位 (m/rad)。

v ：最大关节角速度，范围 [0.01*PI/180, 1.25*PI]，单位 (rad/s)。当用户使用 servoj_pose 指令连续发送较为密集的关节位置指令时，此时默认用户对轨迹路径以及速度存在软实时插补（无法保证加速度及速度平滑连续），此时推荐用户将 v 设置为关节最大速度保护值，并通过调整连续离散点位的发送时间间隔来控制机器人最终运动速度。当用户使用 servoj_pose 指令发送非连续目标位置以控制机器人到达不同的关节位置时，此时 servoj_pose 指令作用同时涵盖路径插补以及运动学插补，此时推荐用户将 v 设置为目标关节产生运动的最大速度，直接通过 servoj_pose 函数实现对最终运动速度指令的规划。

a ：最大关节角加速度，范围 [0.01*PI/180, ∞]，单位 (rad/s2)。最大关节加速度的设定值方法与最大关节角速度设定方法雷同，根据用户使用servoj_pose 运动产生目标运动方案的不同存在一定差异性。

qnear ：目标点位置对应的关节角度，用于确定逆运动学选解，单位 rad

tool ：设置使用的工具的名称，为空时默认为当前使用的工具。

wobj ：设置使用的工件坐标系的名称，为空默认为当前使用的工件坐标系。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回，默认值 false，可缺省。

kp：比例参数，默认值 200，可缺省，建议使用默认参数。

kd：微分参数，默认值 25，可缺省，建议使用默认参数。

smooth_vel：速度平滑系数，范围 [1,100]，当用户连续使用 servoj_pose 指令发送连续离散关节位置点时，若由于发送的连续点位间经过时间差分不连续或

发送周期抖动较大无法有效保证最终速度稳定性时，可以通过调整该平滑系数进行最终运动速度指令的稳定性。平滑系数越大，对速度的平滑效果越强，同时会引入更大的跟随滞后。

smooth_acc：加速度平滑系数，范围 [0,1]，与速度平滑系数雷同，当用户连续使用 servoj_pose 指令发送连续离散关节位置点时，若由于发送的连续点位间经过时间差分不连续或发送周期抖动较大无法有效保证最终加速度稳定性时，可以通过调整该平滑系数进行最终运动加速度指令的稳定性。平滑系数越大，对速度的加平滑效果越强，同时会引入更大的跟随滞后。

返回值：

非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

servo_tcp (vector<double> pose_offset, double v, double a, string tool, bool block, double kp, double kd, double smooth_vel, double smooth_acc)

函数说明：

该指令控制机械臂从当前状态，按照机器人各关节独立运动的方式移动到参考当前机器人末端在参考工件坐标系中的位姿叠加目标位姿增量后的位姿，运动过程中不考虑笛卡尔空间路径且不保证各关节间的运动规划相位同步。当在 servoj_pose 运动过程中收到一条新的 servoj_pose 运动指令时，机器人将会以当前运动规划状态向新的关节目标位置移动，舍弃之前的目标位置。

参数说明：

p ：目标工具在参考工件坐标系下参考机器人当前目标位姿的参考增量，单位(m/rad)。

v ：最大末端速度，范围 [0.00001, 5]，单位：m/s。当用户使用 servo_tcp 指令连续发送较为密集的位姿指令时，此时默认用户对轨迹路径以及速度存在软实时插补（无法保证加速度及速度平滑连续），此时推荐用户将 v 设置为最大速度保护值，并通过调整连续离散点位的发送时间间隔来控制机器人最终运动速度。当用户使用 servo_tcp 指令发送非连续目标位置以控制机器人到达不同的位姿时，此时 servo_tcp 指令作用同时涵盖路径插补以及运动学插补，此时推荐用户将 v 设置为运动的最大速度，直接通过 servo_tcp 函数实现对最终运动速度指令的规划。

a ：最大末端加速度，范围 [0.00001, ∞)，单位：m/s^2。最大末端加速度的设定值方法与最大末端速度设定方法雷同，根据用户使用 servo_tcp 运动产生目标运动方案的不同存在一定差异性。

tool ：设置使用的工具的名称，为空时默认为当前使用的工具。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回，默认值 false，可缺省。

kp：比例参数，默认值 200，可缺省，建议使用默认参数。

kd：微分参数，默认值 25，可缺省，建议使用默认参数。

smooth_vel：速度平滑系数，范围 [1,100]，当用户连续使用 servo_tcp 指令发送连续离散关节位置点时，若由于发送的连续点位间经过时间差分不连续或发送周期抖动较大无法有效保证最终速度稳定性时，可以通过调整该平滑系数进行最

终运动速度指令的稳定性。平滑系数越大，对速度的平滑效果越强，同时会引入更大的跟随滞后。

smooth_acc：加速度平滑系数，范围 [0,1]，与速度平滑系数雷同，当用户连续使用 servo_tcp 指令发送连续离散关节位置点时，若由于发送的连续点位间经过时间差分不连续或发送周期抖动较大无法有效保证最终加速度稳定性时，可以通过调整该平滑系数进行最终运动加速度指令的稳定性。平滑系数越大，对速度的加平滑效果越强，同时会引入更大的跟随滞后。

返回值：

非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

servol(std::vector<double> &pose_list, double v, double a, std::vector<double> &q_near, string &tool, string &wobj, bool block, double kp, double kd, double smooth_vel, double smooth_acc)

函数说明：

该指令控制机械臂从当前状态，按照机器人各关节独立运动的方式移动到目标机器人末端在参考工件坐标系中位姿，运动过程中不考虑笛卡尔空间路径且不保证各笛卡尔方向的运动规划相位同步。当在 servol 运动过程中收到一条新的 servol 运动指令时，机器人将会以当前运动规划状态向新的关节目标位置移动，舍弃之前的目标位置。

参数说明：

p ：目标工具在参考工件坐标系下参考机器人当前目标位姿，单位 (m/rad)。

v ：最大末端速度，范围 [0.00001, 5]，单位：m/s。当用户使用 servol 指令连续发送较为密集的位姿指令时，此时默认用户对轨迹路径以及速度存在软实时插补（无法保证加速度及速度平滑连续），此时推荐用户将 v 设置为最大速度保护值，并通过调整连续离散点位的发送时间间隔来控制机器人最终运动速度。当用户使用 servol 指令发送非连续目标位置以控制机器人到达不同的位姿时，此时 servol 指令作用同时涵盖路径插补以及运动学插补，此时推荐用户将 v 设置为运动的最大速度，直接通过 servol 函数实现对最终运动速度指令的规划。

a ：最大末端加速度，范围 [0.00001, ∞)，单位：m/s^2。最大末端加速度的设定值方法与最大末端速度设定方法雷同，根据用户使用 servol 运动产生目标运动方案的不同存在一定差异性。

qnear ：目标点位置对应的关节角度，用于确定逆运动学选解，单位 rad

tool ：设置使用的工具的名称，为空时默认为当前使用的工具。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回，默认值 false，可缺省。

kp：比例参数，默认值 200，可缺省，建议使用默认参数。

kd：微分参数，默认值 25，可缺省，建议使用默认参数。

smooth_vel：速度平滑系数，范围 [1,100]，当用户连续使用 servol 指令发送连续离散关节位置点时，若由于发送的连续点位间经过时间差分不连续或发送周期抖动较大无法有效保证最终速度稳定性时，可以通过调整该平滑系数进行最终

运动速度指令的稳定性。平滑系数越大，对速度的平滑效果越强，同时会引入更大的跟随滞后。

smooth_acc：加速度平滑系数，范围 [0,1]，与速度平滑系数雷同，当用户连续使用 servol 指令发送连续离散关节位置点时，若由于发送的连续点位间经过时间差分不连续或发送周期抖动较大无法有效保证最终加速度稳定性时，可以通过调整该平滑系数进行最终运动加速度指令的稳定性。平滑系数越大，对速度的加平滑效果越强，同时会引入更大的跟随滞后。

返回值：

非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

备注： 当前所有 servo 运动仅支持同类型 servo 运动间连续控制，当接收到一条不同类型的 servo 运动时，新收到的 servo 运动会被暂时挂起，直到当前正在执行的 servo 运动运行完成后，继续执行新收到的 servo 运动。

---

move_spiral (vector<double> p1, vector<double> p2, double rev, double len, double rad, int mode, double v, double a, vector<double> qnear, string tool, string wobj, bool block, Op op = op_, bool def_acc = true)

函数说明：

该指令通过参数或者结束点两种设置方式，在笛卡尔空间做螺旋轨迹运动。

参数说明：

p1 ： 螺旋线中心点位姿。

p2 ： 螺旋线的目标点位姿，参数设置模式时不参考此参数。

rev ： 总旋转圈数，rev < 0，表示顺时针旋转；rev > 0，表示逆时针旋转。

len ： 轴向移动距离，正负号遵循右手定则，结束点设置模式时不参考此参数，单位 (m)。

red ： 目标点半径，结束点设置模式时不参考此参数，单位 (m)。

mode ： 螺旋线示教模式，0：参数设置，1：结束点设置。

v ： 最大末端线速度，范围 [0.01, 5]，单位 (m/s)。

a ： 最大末端线加速度，范围 [0.01, ∞]，单位 (m/s2)。

qnear ： 目标点位置对应的关节角度，用于确定逆运动学选解，单位 (rad)

tool ： 设置使用的工具的名称，为空时默认为当前使用的工具。

wobj ： 设置使用的工件坐标系的名称，为空默认为当前使用的工件坐标系。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op ： 详见上方 Op 特殊类型说明，可缺省参数。

def_acc ： 是否使用默认加速度，默认为 false，可缺省参数。当开启默认加速度时，执行运动时最大加速度参数不在生效，系统将会根据实际工况计算机器人可以产生的最大加速度指令进行运动，从而提升节拍。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

move_jog (const MoveJogTaskParams& param, bool block)

函数说明：

该指令控制机械臂在关节或者笛卡尔空间做点动。

参数说明：

param ： Jog 运动的相关参数，参考 MoveJogTaskParams。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

stop_manual_move (bool block)

函数说明：

该指令结束机械臂的关节或者笛卡尔 Jog。

参数说明：

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

set_blend_ahead (int per, int num)

函数说明：

该指令设置融合预读取的百分比。

参数说明：

per ： 融合预读取的百分比，单位：%，通常设为 0 或者 50。

num：当前仅当融合预读取百分比参数设置为 0 时生效，可缺省，范围 [1,3]，默认为 1，即仅额外预读取一行运动指令。在机器人目标工作在高速连续短轨迹融合运动时，若不存在过程逻辑问题，应尽可能设置较大的预读取轨迹数量以保证融合稳定性。

返回值：

返回值代表当前任务结束时的状态。

---

move_fixed (const std::string& tool, const std::string& wobj, int32_t axis, int32_t time, std::vector<std::string> scheme_name, const std::vector<std::vector<double> > epose, const std::vector<double> eaxis_v, bool block, const OP& op = _op, bool def_acc = false)

函数说明：

该指令控制机械臂做原地摆运动。

参数说明：

tool ： 工具坐标系名称，为空字符串默认为当前使用的坐标系。

wobj ： 设置使用的工件坐标系的名称，为空时默认为当前使用的工件坐标系。

axis ： 复合运动参考轴向：0 为参考平面轴向 1，1 为参考平面轴向 2. 例如 XOY 0 代表 X 轴 1 代表 Y 轴。

time ： 延迟时间参数单位 ms 当且仅当 eaxis_scheme_name 为空时生效。

scheme_name ： 外部轴方案名称列表。

epose ： 目标外部轴方案所对应自由度位置（三维）列表，记录位置自由度及单位根据外部轴方案所设置自由度及外部轴方案类型改变，单位：rad，m。

eaxis_v ： 外部轴最大规划速度列表，根据对应外部轴方案类型改变，单位：rad/s，m/s。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op ： 详见上方 Op 特殊类型说明 (距离触发无效)，可缺省参数。

def_acc ： 是否使用系统默认加速度，false 表示使用自定义的加速度值，true 表示使用系统自动规划的加速度值，可缺省，默认为 false。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，若无融合为 Finished，若有融合为 Interrupt。当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

1.3.6 can 及 485 总线

set_baudrate_485 (int value, bool block)

函数说明:

该函数用于设置 485 的波特率。

参数说明:

value : 波特率。

block : 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

read_raw_data_485 (vector<int8_t> _return, int len)

函数说明:

485 端口读取长度为 len 的字节数据。

参数说明:

_return : 读取到的数据，未读到数据返回空列表。

len : 需要读取的长度。

返回值:

无。

---

read_raw_data_485_h (vector<int8_t > _return, vector<int8_t > head, int len)

函数说明:

匹配头 head 后读取到长度为 len 的一帧数据。

参数说明:

_return : 读取到的数据，未读到数据返回空列表。

head : 需要匹配的头数据。

len : 需要读取的长度。

返回值:

无。

示例:

```
vector<int8_t> _return;
vector<int8_t> head = {255,1};
read_raw_data_485_h (_return, head, 10)
```

---

read_raw_data_485_ht (vector< int8_t > _return, vector<int8_t> head, vector<int8_t> tail)

函数说明:

匹配头 head 和尾 tail 读取到一帧匹配的数据。

参数说明:

_return : 读取到的数据，未读到数据返回空列表。

head : 需要匹配的头数据。

tail : 需要匹配的尾数据。

返回值:

无。

示例:

```
vector<int8_t> _return;
vector<int8_t> head = {255,1};
vector<int8_t> tail = {255,1};
read_raw_data_485_ht (_return, head, tail);
```

---

write_raw_data_485 (vector<int8_t> value)

函数说明:

485 写原生数据，将表 value 中的数据写入 485 端口。

参数说明:

data : 需要写入的数据列表。

返回值:

true : 成功。

false : 失败。

示例:

```
vector<int8_t> data={255,1};
bool rlt = write_raw_data_485(data);
```

---

write_raw_data_485_h (vector<int8_t> value, vector<int8_t> head)

函数说明:

485 写原生数据，将列表 value 中的数据加上 head 写入 485 端口。

参数说明:

value : 需要写入的数据列表。

head ： 需要添加的头。

返回值:

true ： 成功。

false ： 失败

示例:

```
vector<int8_t> data={1,2,3};
vector<int8_t> head={255,255};
bool rlt = write_raw_data_485_h (data, head)
```

---

write_raw_data_485_ht (vector<int8_t> value, vector<int8_t> head, vector<int8_t> tail)

函数说明:

485 写原生数据，将列表 value 中的数据加上头 head 和尾 tail 写入 485 端口。

参数说明:

value ： 需要写入的数据列表。

head ： 需要添加的头。

tail ： 需要添加的尾。

返回值:

true ： 成功。

false ： 失败。

示例:

```
vector<int8_t> data={1,2,3};
vector<int8_t> head={255,255};
vector<int8_t> tail={255,255};
bool rlt = write_raw_data_485_ht (data, head, tail)
```

---

tool_read_raw_data_485 (vector<int8_t> _return , int len)

函数说明:

末端 485 端口读取长度为 len 的字节数据。

参数说明:

_return ： 读取到的数据，未读到数据返回空列表。

len ： 需要读取的长度。

返回值:

无。

示例:

```
vector<int8_t> data;
tool_read_raw_data_485 (data, 10);
```

---

tool_read_raw_data_485_h (vector<int8_t> _return, vector<int8_t> head, int:len)

函数说明:

末端 485 匹配头 head 后读取到长度为 len 的一帧数据。

参数说明:

_return : 读取到的数据，未读到数据返回空列表。

head : 需要匹配的头数据。

len : 需要读取的长度。

返回值:

无。

示例:

```
vector<int8_t> data;
vector<int8_t> head={255,255};
tool_read_raw_data_485_h (data, head, 10);
```

---

tool_read_raw_data_485_ht (vector<int8_t> _return, vector<int8_t> head, vector<int8_t> tail)

函数说明:

末端 485 匹配头 head 和尾 tail 读取到一帧匹配的数据。

参数说明:

_return : 读取到的数据，未读到数据返回空列表。

head : 需要匹配的头数据。

tail : 需要匹配的尾数据。

返回值:

无。

示例:

```
vector<int8_t> data;
vector<int8_t> head={255,1};
vector<int8_t> tail={1,255};
tool_read_raw_data_485_ht (data, head, tail);
```

---

tool_write_raw_data_485 (vector<int8_t> data)

函数说明:

末端 485 写原生数据，将表 data 中的数据写入 485 端口。

参数说明:

data : 需要写入的数据列表。

返回值:

true : 成功。

false : 失败。

示例:

```
vector<int8_t> data = {1,2,3};
tool_write_raw_data_485 (data);
```

---

tool_write_raw_data_485_h (vector<int8_t> value, vector<int8_t> head)

函数说明:

末端 485 写原生数据，将表 value 中的数据加上 head 写入 485 端口。

参数说明:

value : 需要写入的数据列表。

head : 需要添加的头。

返回值:

true : 成功。

false : 失败

示例:

```
vector<int8_t> data = {1,2,3};
vector<int8_t> head={255,1};
tool_write_raw_data_485_h (data, head)
```

---

tool_write_raw_data_485_ht (vector<int8_t> value, vector<int8_t> head, vector<int8_t> tail)

函数说明:

末端 485 写原生数据，将表 value 中的数据加上头 head 和尾 tail 写入 485 端口。

参数说明:

value : 需要写入的数据列表。

head : 需要添加的头。

tail : 需要添加的尾。

返回值:

true : 成功。

false : 失败。

示例:

```
vector<int8_t> data = {1,2,3};
vector<int8_t> head={255,1};
vector<int8_t> tail={1,255};
tool_write_raw_data_485_ht (data, head, tail)
```

---

set_baudrate_can (int value, bool block)

函数说明:

该函数用于设置 CAN 的波特率。

参数说明:

value : 波特率;

block : 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

read_raw_data_can (vector<int8_t> data)

函数说明:

读取一帧 can 的字节数据。

参数说明:

data : 读取到的数据，未读到数据返回空列表，读到数据时，列表的第一个数据为发送端的 can 帧 id。

返回值:

无。

---

write_raw_data_can (double id, vector<int8_t> data)

函数说明:

can 写帧为 id，数据为 data 的原生数据。

参数说明:

id : 数据帧的 id。

data : 要发送的数据列表。

返回值:

true : 成功。

false : 失败。

1.3.7 系统函数及外设

! 注意

若范围参数越界，函数将返回 false 或 0

---

set _digital_out_mode (int16_t num, int16_t type, int freq, int duty_cycle)

函数说明:

该函数可设置控制柜上的通用 IO 输出的信号类型。

参数说明:

num : 控制柜上的 IO 输出口序号，范围从 1-16。

type : 0 为电平模式，1 为周期脉冲模式。

freq : 频率，单位：Hz，范围从 1-100。

duty_cycle : 占空比，单位：%，1-100。

参数错误时函数不改变 IO 输出信号类型。

类型设置完成后，若设置未电平迷失，则需要主动发送输出信号才能生效。若设置成周期脉冲模式后，需要发送一个高电平的启动信号。

返回值:

当前任务结束时的状态。

备注: 若要将通用 output 修改为脉冲输出，在对通用 IO 输出类型完成配置后，仍然需要使用 set_standard_digital_out 函数来控制脉冲的有无。当前仅允许对脉冲类型统一设置，无法对不同通用 output 口，设置不同的脉冲类型。脉冲脉宽的最小识别率为 1ms。即当频率为 100hz 时，占空比最小值为 10。若超出范围将关闭脉冲输出。

---

set_standard_digital_out (int num, bool val, bool block)

函数说明:

该函数可控制控制柜上的 IO 输出口的高低电平。

参数说明:

num : 控制柜上的 IO 输出口序号，范围从 1-16。

val : true 为高电平，false 为低电平。

参数错误时函数不改变 IO 输出。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

get_standard_digital_out (int num)

函数说明：

该函数可获取控制柜上通用 IO 输出口的高低电平，返回 true 为高电平，false 为低电平。

参数说明：

num ：控制柜上的 IO 输出口序号，范围从 1-16。

返回值：

bool，返回 true 为高电平，false 为低电平。

---

get_standard_digital_in (int num)

函数说明：

该函数可读取控制柜上的用户 IO 输入口的高低电平，返回 true 为高电平，false 为低电平。

参数说明：

num ：控制柜上的 IO 输入口序号，范围从 1-16。

返回值：

bool，返回 true 为高电平，false 为低电平。

---

set_tool_digital_out (int num, bool val, bool block)

函数说明：

该函数可控制机械臂末端的 IO 输出口的高低电平。

参数说明：

num ：机械臂末端的 IO 输出口序号，范围从 1-2。

val ：true 为高电平，false 为低电平。

参数错误时函数不改变 IO 输出。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

get_tool_digital_out (int num)

函数说明:

该函数可读取机械臂末端的 IO 输入口的高低电平,返回 true 为高电平,false 为低电平。

参数说明:

num : 机械臂末端的 IO 输出口序号,范围从 1-2。

返回值:

bool,返回 true 为高电平,false 为低电平。

---

get_tool_digital_in (int num)

函数说明:

该函数可读取机械臂末端的 IO 输入口的高低电平,返回 true 为高电平,false 为低电平。

参数说明:

num : 机械臂末端的 IO 输出口序号,范围从 1-2。

返回值:

bool,返回 true 为高电平,false 为低电平。

---

get_function_digital_in (int portnum)

函数说明:

该函数可读取控制柜功能输入 IO 高低电平,返回 true 为高电平,false 为低电平。

参数说明:

ortnum : 控制柜功能 IO 输入口序号,范围从 1-8。

返回值:

ool,返回 true 为高电平,false 为低电平。

---

get_function_digital_out (int portnum)

函数说明:

该函数可读取控制柜功能输出 IO 高低电平,返回 true 为高电平,false 为低电平。

参数说明:

portnum : 控制柜功能 IO 输出口序号,范围从 1-8。

返回值:

bool,返回 true 为高电平,false 为低电平。

get_standard_analog_voltage_in (int num)

函数说明:

该函数可读取控制柜上的模拟电压输入。

参数说明:

num ： 控制柜上的模拟电压通道序号，范围从 1-4。

返回值:

对应通道的模拟电压值。

---

set_standard_analog_voltage_out (int num, double value, bool block)

函数说明:

该函数可设置控制柜上的模拟电压输出。

参数说明:

num ： 控制柜上的模拟电压通道序号，范围从 1-4;

value ： 设置的模拟电压值，范围 [0,10]，单位 V;

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

get_tool_analog_voltage_in (int num)

函数说明:

该函数可读取机械臂末端的模拟电压输入，单位 V。

参数说明:

num ： 机械臂末端的模拟电压通道序号，范围从 1-2。

返回值:

对应通道的模拟电压值。

---

get_standard_analog_current_in (int num)

函数说明:

该函数可读取控制柜上的模拟电流输入，单位 mA。

参数说明:

num ： 控制柜上的模拟电流通道序号，范围从 1-4。

返回值:

对应通道的模拟电流值

---

set_standard_analog_current_out (int num, double value, bool block)

函数说明:

该函数可设置控制柜上的模拟电流输出。

参数说明:

num : 控制柜上的模拟电流通道序号，范围从 1-4;

value : 设置的模拟电流值，范围 [4,20]，单位 mA;

block : 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

write_bool_reg (int num, bool value)

函数说明:

该函数可修改内部 bool 寄存器的值。

参数说明:

num : 内部寄存器序号，num 范围为 1-64。

val : true 表示真，false 表示假。

参数错误时函数不改变内部寄存器数值。

返回值:

返回当前任务结束时的状态。

---

write_word_reg (int num, int value)

函数说明:

该函数可修改内部 word 寄存器的值。

参数说明:

num : 内部寄存器序号，num 范围为 1-32。

val : 修改的寄存器的值。

参数错误时函数不改变内部寄存器数值。

返回值:

返回当前任务结束时的状态。

---

write_float_reg (int num, double value)

函数说明:

该函数可修改内部 float 寄存器的值。

参数说明:

num : 内部寄存器序号, num 范围为 1-32。

val : 修改的寄存器的值。

参数错误时函数不改变内部寄存器数值。

返回值:

返回当前任务结束时的状态。

---

read_bool_reg (int num)

函数说明:

该函数可返回内部 bool 寄存器的值。

参数说明:

num : 内部寄存器序号, num 范围为 1-64

返回值:

true 表示真, false 表示假。

---

read_word_reg(int num)

函数说明:

该函数可返回内部 word 寄存器的值。

参数说明:

num : 内部寄存器序号, num 范围为 1-32

返回值:

word 寄存器的值

---

read_float_reg (int num)

函数说明:

该函数可返回内部 float 寄存器的值。

参数说明:

num : 内部寄存器序号, num 范围为 1-32

返回值:

float 寄存器的值

---

get_function_reg_in (int num)

函数说明:

该函数可读取功能输入寄存器的值。

参数说明:

num : 内部寄存器序号，范围从 1-16。

返回值:

bool 寄存器的值。

---

get_function_reg_out (int num)

函数说明:

该函数可读取功能输出寄存器的值。

参数说明:

num : 内部寄存器序号，范围从 1-16。

返回值:

bool 寄存器的值。

---

set_wobj_offset (std::vector<double> & wobj, bool active)

函数说明:

基于当前的工件坐标系设置一个偏移量，后续的 move 类脚本的参考工件坐标系上都将添加这个偏移量。该偏移量以右乘的形式叠加在当前工件坐标系的初始值上。

参数说明:

wobj_offset : {x, y, z, rx, ry, rz} 工件坐标系偏移量(单位:m,rad)。

active : true 为启用偏移量，false 为取消偏移量。

返回值:

无

---

set_tool_data (string name, vector<double> tool_offset, vector<double> payload, vector<double> inertia_tensor)

函数说明:

设置工具末端相对于法兰面坐标系的偏移，设置成功后，后续运动函数中的 TCP 设置为该 TCP 或者为空时，使用该 TCP 参数。

参数说明:

name : 工具坐标系的名字，类型 string，最长不超过 32 个字节长度。

tool_offset : 工具 TCP 偏移量 [x_off, y_off,z_off,rx,ry,rz]，单位 (m, rad)。

payload ： 末端负载质量，质心，[mass,x_cog,y_cog,z_cog]，单位（kg，m）。

inertia_tensor ： 末端工具惯量矩阵参数，参数 1-6 分别对应矩阵 xx、xy、xz、yy、yz、zz 元素，单位 kg*m^2。

返回值：

返回当前任务结束时的状态。

---

cal_ikine (vector<double> _return, vector<double> p, vector<double> q_near, vector<double> tool, vector<double> wobj)

函数说明：

基于目标工具在工件坐标系中的笛卡尔空间位姿，通过机器人逆运动学计算机器人对应的关节角位置，在求解过程中，会选取靠近参考关节角位置或当前机械臂关节位置的解。

参数说明：

_return ： 关节位置。

p：需要计算的末端位姿在设置工件坐标系的值，包含当前有效的工具偏移量，位置单位 m，姿态单位 rad。

q_near ： 用于计算逆运动学的参考关节位置，为空时使用当前关节值。

tcp ： 工具坐标系信息,tcp 偏移量 {x_off,y_off,z_off,rx,ry,rz},(单位：m，rad)，为空使用当前工具。

wobj：工件坐标系相对于世界坐标系的位移 {x，y，z，rx，ry，rz}，(单位：m，rad)，为空使用当前工件坐标系。

返回值：

无。

---

cal_fkine (vector<double> _return, vector<double> joints_position, vector<double> tool, vector<double> wobj)

函数说明：

基于目标机器人关节角位置，通过机器人正运动学计算目标工具在目标工件坐标系中的笛卡尔空间位姿。

参数说明：

_return ： 末端姿态。

joints_position ： 需要计算正解的关节角，单位 rad。

tool ： 工具坐标系信息，tcp 偏移量 [x_off,y_off,z_off,rx,ry,rz]，(单位：m，rad)，为空使用当前 tcp 值。

wobj ： 工件坐标系相对于基坐标系的位移 [x，y，z，rx，ry，rz]，(单位：m，rad)，为空使用当前 wobj。

返回值:

无。

---

get_tcp_pose (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂工具末端点在基坐标系下的位姿。

参数说明:

data：末端位置。

返回值:

无。

---

get_tcp_pose_coord (vector<double> &data, const string& tool, const string& wobj)

函数说明:

该函数可获取末端法兰在工具坐标系和工件坐标系下的位姿。

参数说明:

data：末端法兰的位姿。

tool ：工具坐标系名称，默认为当前使用的坐标系。

user ：工件坐标系名称，默认为当前使用的坐标系。

返回值:

无。

---

get_tcp_speed (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂当前生效工具末端原点的空间速度。

参数说明:

data ：末端速度列表，单位 m/s,rad/s。

返回值:

无。

---

get_tcp_acceleration (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂工具当前生效工具末端原点的空间加速度。

参数说明:

data：末端加速度列表，单位 m/ s2，rad/ s2。

返回值:

无。

---

get_tcp_force (vector<double> data)

函数说明:

该函数可获取当前机械臂工具当前生效工具末端原点的空间力信息。该函数当且仅当安装了末端力传感器并正确配置启用时有效。

参数说明:

data: 末端力矩信息，[Fx,Fy,Fz,Mx,My,Mz]，单位 N、N.m。

返回值:

无。

---

get_tcp_force_tool (vector<double>& data, const string& tool)

函数说明:

该函数可获取机械臂工具末端在目标工具坐标系下的力矩信息。该函数当且仅当安装了末端力传感器并正确配置启用时有效。

参数说明:

data ：末端力矩信息，[Fx,Fy,Fz,Mx,My,Mz]，单位 N、N.m。

tool ：工具坐标系名称，默认为当前使用的坐标系。

返回值:

无。

---

get_tcp_offset (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂有效的末端工具的偏移量。

参数说明:

data: [x_off，y_off,z_off,rx,ry,rz] 返回 TCP 偏移量信息，单位 m，rad。

返回值:

无。

---

get_tool_load (vector<double> data)

函数说明:

该函数可获取当前设置工具的负载质量及质心位置。

参数说明:

data: 质量单位 kg，质心位置单位 m，[mass,x_cog,y_cog,z_cog]。

返回值:

无。

---

get_wobj (vector<double> data)

函数说明:

该函数可获取当前设置的工件坐标系的值。

参数说明:

data: [x, y, z, rx, ry, rz] 工件坐标系相对于基坐标系的位移（单位: m, rad)。

返回值:

无。

---

get_actual_joints_position (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂各关节的角度。

参数说明:

data : 1-6 轴关节角度列表，单位 rad。

返回值:

无。

---

get_target_joints_position (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂各关节的规划角度。

参数说明:

data : 1-6 轴目标关节角度列表，单位 rad。

返回值:

无。

---

get_actual_joints_speed (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂各关节角速度。

参数说明:

data : 1-6 轴关节速度列表，单位 rad/s。

返回值:

无。

---

get_target_joints_speed (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂各关节规划角速度。

参数说明:

data : 1-6 轴目标关节速度列表，单位 rad/s。

返回值:

无。

---

get_actual_joints_acceleration (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂各关节角加速度。

参数说明:

data : 1-6 轴关节加速度列表，单位 rad/ s2。

返回值:

无。

---

get_target_joints_acceleration (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂各关节角规划加速度。

参数说明:

data : 1-6 轴目标关节加速度列表，单位 rad/ s2。

返回值:

无。

---

get_actual_joints_torque (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂各关节力矩。

参数说明:

data : 1-6 轴关节力矩列表，单位 N.m。

返回值:

无。

---

get_target_joints_torque (vector<double> data)

函数说明:

该函数可获取当前状态下机械臂各关节目标力矩。

参数说明:

data : 1-6 轴关节加速度列表，单位 rad/ s2。

返回值:

无。

---

get_flange_pose (vector<double> pose)

函数说明:

该函数可获取当前状态下机械臂末端法兰在基坐标系下的位姿。

参数说明:

pose: 末端法兰位置 pose。

返回值:

无。

---

get_flange_speed (vector<double> vel)

函数说明:

该函数可获取当前状态下机械臂末端法兰在基坐标系下的速度。

参数说明:

vel: 末端法兰速度列表，单位 m/s,rad/s。

返回值:

无。

---

get_flange_acceleration (vector<double> acc)

函数说明:

该函数可获取当前状态下机械臂末端法兰在基坐标系下的加速度。

参数说明:

acc : 末端法兰加速度列表，单位 m/ s2，rad/ s2。

返回值:

无。

---

get_robot_state (vector<int8_t> data)

函数说明:

该函数可获取当前机器人状态。

参数说明:

data : 机器人状态信息列表，data[0] 表示机器人状态，data [1] 表示程序状态，data [2] 表示安全控制器状态，data [3] 表示操作模式。(各状态列表详见第 1.2 章节)

返回值:

无。

---

simulation (bool sim, bool block)

函数说明:

切换机器人到仿真或者真机模式。在使用该函数前，需要保证机器人完全处于静止状态，否则会引起机器人异常运动并停机报警。

参数说明:

sim : true: 仿真，false: 真机

block : 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

speed (double val)

函数说明:

设置机器人全局速度百分比。

参数说明:

val : 设置机器人全局速度，范围 [1,100]。

返回值:

返回当前任务结束时的状态。

---

set_load_data (vector<double> load)

函数说明:

设置抓取负载。可以在程序运行过程中设置机器人当前的负载（质量、质心）。

参数说明:

load : 末端工具抓取负载质量，质心，{mass,x_cog,y_cog,z_cog}，相对于工具坐标系，质量范围 [0, 35]，单位 (kg, m)。

返回值:

无

---

stop_record_track ()

函数说明:

该函数停止轨迹记录。

参数说明:

无。

返回值:

无

---

start_record_track (string name, number mode, string tool, string wobj)

函数说明:

该函数开启轨迹记录，当超过允许记录的轨迹长度（针对基于位置记录）或允许记录的时长时（针对基于时间记录），会自动停止文件记录，并且暂停当前运行的程序。文件会记录机器人的各关节弧度值和选定工具、工件坐标系下的笛卡尔空间位姿。

参数说明:

name ： 轨迹名称。

mode ： 轨迹类型，mode=0 基于位置记录（与上一记录点所有关节偏移总量到达 5° 时记录新点）；mode=1 基于时间记录（与上一记录点间隔 250ms 记录新点）

tool ： 工具坐标系名称。

wobj ： 工件坐标系名称。

返回值:

当前任务的 id。

---

collision_detect (int level)

函数说明:

设置碰撞检测等级。

参数说明:

level ： 0：关闭碰撞检测，1-5：对应设置碰撞检测等级 1 到等级 5。

返回值:

任务结束时状态。

---

collision_detection_reset ()

函数说明:

重置碰撞检测警告。

参数说明:

无。

返回值:

无

---

set_teach_pendant (bool enable)

函数说明:

启用或禁用示教器的物理按键。

参数说明:

`enable : true` 启动示教器物理按键，`false` 禁用示教器物理按键。

返回值:

任务结束时状态。

---

set_teach_speed (int v)

函数说明:

设置示教速度的百分比。

参数说明:

`v :` 示教速度的百分比，范围 `[1,100]`。

返回值:

任务结束时状态。

---

get_teach_speed ()

函数说明:

获取示教速度的百分比。

参数说明:

无。

返回值:

示教速度的百分比。

---

get_global_speed ()

函数说明:

获取全局速度的百分比。

参数说明:

无。

返回值:

全局速度的百分比。

---

reach_check (std::vector<double> &base, std::vector<double> &tool, std::vector<double> &wobj, std::vector<double> &ref_pos, std::vector<std::vector<double> > &check_points)

函数说明:

计算目标机器人末端工具在参考工件坐标系在机器人以特定安装方向的可达性。

参数说明:

_return: 可达性检查结果，返回所有指令发送的目标位姿所对应的可达性检测结果。

base: 机器人安装参考坐标系，姿态描述为 Rz*Ry*Rx，参考全局世界坐标系，单位 m/rad。

tool: 机器人末端工具坐标系，姿态描述为 Rz*Ry*Rx，参考机器人法兰坐标系，单位 m/rad。

wobj: 参考工件坐标系，姿态描述为 Rz*Ry*Rx，参考全局世界坐标系，单位 m/rad。

ref_pos: 待检查的机器人末端工具在工件坐标系下的目标笛卡尔空间位姿所使用的参考选解关节位置，单位 rad。

check_points: 待检查的机器人末端工具在工件坐标系下的目标笛卡尔空间位姿，单位 m/rad。

返回值:

ReachabilityParam 类型。详细参考 ReachabilityParam 类型定义。

---

switch_mode (int32_t mode)

函数说明:

手自动模式切换，当且仅当安全设置中未启用通过外部 IO 切换模式时有效，否则在调用接口时会报错。

参数说明:

mode : 0: 手动模式, 1: 自动模式。

返回值:

阻塞执行，返回值代表当前任务结束时的状态。

```
struct RobotStatusList {
 std::vector<double> jointExpectPosition;
 std::vector<double> jointExpectVelocity;
 std::vector<double> jointExpectAccelera;
 std::vector<double> jointActualPosition;
 std::vector<double> jointActualVelocity;
 std::vector<double> jointActualAccelera;
 std::vector<double> jointActualCurrent;
 std::vector<double> jointTemperature;
 std::vector<double> driverTemperature;
 std::vector<double> cartExpectPosition;
 std::vector<double> cartExpectVelocity;
 std::vector<double> cartExpectAccelera;
 std::vector<double> cartActualPosition;
 std::vector<double> cartActualVelocity;
 std::vector<double> cartActualAccelera;
 std::vector<bool> slaveReady;
 bool collision;
 int collisionAxis;
 bool emcStopSignal;
 int robotState;
 int robotError;
}
```

参数说明：

jointExpectPosition：关节位置控制指令，单位 rad；

jointExpectVelocity：关节速度控制指令，单位 rad/s；

jointExpectAccelera：关节加速度控制指令，单位 rad/s2；

jointActualPosition：关节实际位置，单位 rad；

jointActualVelocity：关节实际速度，单位 rad/s；

jointActualAccelera：关节实际加速度，单位 rad/s2；

jointActualCurrent：关节实际电流，单位为参考各关节电机额定电流的千分比比例；

jointTemperature：关节实际温度，单位 °；

driverTemperature：关节驱动板温度，单位 °；

cartExpectPosition: 机器人末端工具在世界坐标系下的位姿指令，单位 m/rad;

cartExpectVelocity: 机器人末端工具在世界坐标系下的位姿速度指令，单位 m/s / rad/s;

cartExpectAccelera: 机器人末端工具在世界坐标系下的位姿加速度指令，单位 m/s2 rad/s2;

cartActualPosition: 机器人末端工具在世界坐标系下的实际位姿，单位 m/rad;

cartActualVelocity: 机器人末端工具在世界坐标系下的实际位姿速度，单位 m/s / rad/s;

cartActualAccelera: 机器人末端工具在世界坐标系下的实际位姿加速度，单位 m/s2 /rad/s2;

slaveReady: 机器人关节使能状态，当且仅当关节处于使能模式下为 true，否则为 false;

collision: 机器人碰撞检测触发状态，当且仅当机器人触发碰撞且未被复位时为 true，否则为 false;

collisionAxis: 触发机器人碰撞检测状态的关节号，当且仅当机器人触发碰撞且未被复位时有效，否则为 0;

emcStopSignal: 机器人触发急停信号状态，当机器人示教器急停被按下或外部预留急停断开时为 true，否则为 false;

robotState: 机器人状态机;

robotError: 获取当前机器人错误代码;

---

getRobotStatus (RobotStatusList& status)

函数说明:

获取机器人当前的位姿等信息，具体信息定义参考 RobotStatusList 数据结构。

参数说明:

status : 机器人位姿等信息，参考 RobotStatusList。

返回值:

无。

```
struct IOStatusList {
 std::vector<double> analogCurrentOutputs;
 std::vector<double> analogVoltageOutputs;
 std::vector<double> analogCurrentInputs;
 std::vector<double> analogVoltageInputs;
 std::vector<bool> digitalInputs;
 std::vector<bool> digitalOutputs;
 std::vector<bool> toolIOIn;
```

```
    std::vector<bool> toolIOOut;
    std::vector<bool> toolButton;
    std::vector<bool> funRegisterInputs;
    std::vector<bool> funRegisterOutputs;
    std::vector<bool> boolRegisterInputs;
    std::vector<bool> boolRegisterOutputs;
    std::vector<int16_t> wordRegisterInputs;
    std::vector<int16_t> wordRegisterOutputs;
    std::vector<double> floatRegisterInputs;
    std::vector<double> floatRegisterOutputs;
}
```

参数说明：

analogCurrentOutputs：模拟量电流输出，单位 mA；

analogVoltageOutputs：模拟量电压输出，单位 V；

analogCurrentInputs：模拟量电流输入。单位 mA；

analogVoltageInputs：模拟量电压输入，单位 V；

digitalInputs：通用 IO 输入；

digitalOutputs：通用 IiO 输出；

toolIOIn：末端 IO 输入；

toolIOOut：末端 IO 输出；

toolButton：末端 T 按钮与 S 按钮状态；

funRegisterInputs：功能输入寄存器；

funRegisterOutputs：功能输出寄存器；

boolRegisterInputs：bool 输入寄存器；

boolRegisterOutputs：bool 输出寄存器；

wordRegisterInputs：word 输入寄存器；

wordRegisterOutputs：word 输出寄存器；

floatRegisterInputs：float 输入寄存器；

floatRegisterOutputs：float 输出寄存器；

---

getRobotIOStatus (IOStatusList& status)

函数说明：

获取机器人当前 IO 和寄存器信息，具体信息定义参考 IOStatusList 数据结构。

参数说明：

status : 当前 IO 和寄存器信息, 参考 IOStatusList。

返回值:

无。

---

read_encoder_count ()

函数说明:

读取外接 INC 的实际 count 值。

参数说明:

无。

返回值:

外接 INC 的实际 count 值。

---

get_origin_DH (std::vector<std::vector<double> >& dh)

函数说明:

当前机器人型号原始 DH 参数。

参数说明:

dh : 原始 DH 参数, 参数顺序 a, alpha, d, theta, 单位 m/rad。

返回值:

无。

---

get_calib_DH (std::vector<std::vector<double> >& calib_dh)

函数说明:

当前机器人型号标定补偿后 DH 参数。

参数说明:

calib_dh : 补偿后 DH 参数, 参数顺序 a, alpha, d, theta, 单位 m/rad。

返回值:

无。

---

get_robot_type (std::vector<std::string>& type)

函数说明:

获取机器人系列号, 型号, ext, SN。

参数说明:

type : 机器人系列号, 型号, ext, SN。

返回值:

无。

---

get_ext_torque (std::vector<double>& torque)

函数说明:

获取关节观测外力矩。

参数说明:

torque : 关节观测外力矩。

返回值:

无。

---

set_system_value_bool (std::string& name, bool value)

函数说明:

设置 bool 类型的系统变量。

参数说明:

name : 系统变量名称。如果变量在域里，名称格式为: " 域名. 变量名"。

value : 系统变量的值。

返回值:

0 : 设置成功，其它: 设置不成功。

---

set_system_value_double (std::string& name, double value)

函数说明:

设置 double 类型的系统变量。

参数说明:

name : 系统变量名称。如果变量在域里，名称格式为: " 域名. 变量名"。

value : 系统变量的值。

返回值:

0 : 设置成功，其它: 设置不成功。

---

set_system_value_str (std::string& name, std::string& value)

函数说明:

设置 string 类型的系统变量。

参数说明:

name : 系统变量名称。如果变量在域里，名称格式为: " 域名. 变量名"。

value : 系统变量的值。

返回值:

0 : 设置成功，其它: 设置不成功。

---

set_system_value_list (std::string& name, std::vector<double> value)

函数说明:

设置 num_list 类型的系统变量。

参数说明:

name : 系统变量名称。如果变量在域里，名称格式为: " 域名. 变量名"。

value : 系统变量的值。

返回值:

0 : 设置成功，其它: 设置不成功。

---

get_system_value_bool (std::string& name)

函数说明:

获取 bool 类型系统变量的值。

参数说明:

name : 系统变量名称。如果变量在域里，名称格式为: " 域名. 变量名"。

返回值:

系统变量的值。

---

get_system_value_double (std::string& name)

函数说明:

获取 double 类型系统变量的值。

参数说明:

name : 系统变量名称。如果变量在域里，名称格式为: " 域名. 变量名"。

返回值:

系统变量的值。

---

get_system_value_str (std::string& value, std::string& name)

函数说明:

获取 string 类型系统变量的值。

参数说明:

value : 系统变量的值。

name : 系统变量名称。如果变量在域里，名称格式为: " 域名. 变量名"。

返回值:

无。

---

get_system_value_list (std::vector<double> value, std::string& name)

函数说明:

获取 num_list 类型系统变量的值。

参数说明:

value : 系统变量的值。

name : 系统变量名称。如果变量在域里，名称格式为: " 域名. 变量名"。

返回值:

无。

---

get_simulation_state ( )

函数说明:

获取当前是否是仿真模式。

参数说明:

无。

返回值:

true : 仿真, false : 真机。

---

get_system_value_lists (std::vector<std::vector<double> > & value, const std::string& name)

函数说明:

获取机器人 pose_list 类型全局变量的值。

参数说明:

value : pose_list 类型全局变量的值。

name : 系统变量名称。如果变量在域里，名称格式为: " 域名. 变量名"。

返回值:

无。

---

get_tool_data (std::vector<double> & tool, const std::string& name)

函数说明:

获取工具坐标系的值。

参数说明:

tool : 工具坐标系的值 (长度: 10)。

name : 空字符串: 返回当前坐标系的值, 不为空: 返回指定工具坐标系的值。

返回值:

无。

---

get_base (std::vector<double> & base)

函数说明:

获取机器人安装位姿。

参数说明:

pose : Home 位置关节角。

返回值:

无。

---

get_home_pose (std::vector<double> & pose)

函数说明:

获取 Home 位置。

参数说明:

value : pose_list 类型全局变量的值。

pose : Home 位置关节角。

返回值:

无。

---

set_tool_data_workcell (const std::string& name, const std::vector<double> & tool, const std::vector<double> & payload, const std::vector<double> & inertia_tensor)

函数说明:

设置工具末端相对于法兰面坐标系的偏移及质量、质心，设置成功后，后续运动函数中的 TCP 设置为该 TCP 或者为空时，使用该 TCP 参数，此函数会将工程文件中的值同步修改。

参数说明:

name : 工具坐标系的名称。

tool : 工具 TCP 偏移量 {x_off, y_off, z_off, rx, ry, rz}，单位 (m, rad)。

payload : 末端工具质量，质心，{mass,x_cog,y_cog,z_cog}，相对于法兰坐标系，单位 (kg, m)。

inertia_tensor : 末端工具惯量矩阵参数，参数 1-6 分别对应矩阵 xx、xy、xz、yy、yz、zz 元素，单位 kg*m^2。

返回值:

任务结束时的状态。

---

set_wobj_workcell (const std::string& name, const std::vector<double> & wobj)

函数说明:

根据名称修改要配置的工件坐标系，此函数会将工程文件中的值同步修改。

参数说明:

name : 工件坐标系的名称。

wobj : 工件坐标系的值 {x, y, z, rx, ry, rz}, 单位 (m, rad)。

返回值:

任务结束时的状态。

---

path_offset_cal (const int32_t path_type, const std::vector<std::vector<double> > & path_point, const std::string& tool, const std::string& wobj, const std::vector<double> & path_offset, const std::vector<double> & path_retract, const std::vector<double> & tool_rotation, const double path_rotation, const std::vector<double> & weld_rotation = std::vector<double>(), const int32_t tool_z_offset = 0, const std::vector<int32_t> & path_type_list = std::vector<int32_t>(), const std::vector<double> & plane_normals = std::vector<double>(), bool average_height = false)

函数说明:

按照输入的偏移量和和偏移方向计算出偏移后的路径点序列。

参数说明:

path_type : 路径类型, 0: 直线, 1: 圆弧 2: 直线、直线, 3: 直线、圆弧, 4: 直线、圆弧、直线, 5: 圆弧、直线、圆弧, 6: 参考输入路径类型队列(path_type)。

path_point : 轨迹路点序列。

tool : 使用的工具坐标系的名称，为空时默认为当前使用的工具坐标系。

wobj : 使用的工件坐标系的名称，为空时默认为当前使用的工件坐标系。

path_offset : 路径在路径坐标系起点处的偏移，两个元素。

path_retract : 路径在路径坐标系终点处的缩进，两个元素。

tool_rotation : 工具自身参考 tool 的旋转量，两个元素。

path_rotation : 工具参考路径坐标系起点处路径切线方向的旋转量。

weld_rotation : 相对于焊道坐标系的 RY、RZ，可缺省，两个元素。

tool_z_offset : 工具 z 偏移量，可缺省。

path_type : 路径类型序列，可缺省。

plane_normals : 法向量，可缺省，三个元素。

average_height ： 根据法向量高度平均，可缺省。

返回值：

PathOffsetResult 类型数据，偏移后的路径点序列。

---

set_installation(std::vector<double> installation)

函数说明：

设置机械臂的安装方式。

参数说明：

installation ： 安装方式。两个元素 {ry, rz}，ry ： 机器人绕基座倾斜弧度，rz ： 机器人绕基座旋转弧度。

返回值：

任务结束时的状态。

---

get_installation( )

函数说明：

获取机械臂的安装方式。

参数说明：

无。

返回值：

安装方式。std::vector<double> 类型，两个元素 {ry, rz}，ry ： 机器人绕基座倾斜弧度，rz ： 机器人绕基座旋转弧度。

1.3.8 调试相关

log_info (string message)

函数说明：

该函数可插入 log 日志，记录运行问题。

参数说明：

message ： 日志描述。

返回值：

无。

---

log_error (string message)

函数说明：

该函数可在运行过程中产生弹窗，并暂停当前所有任务。

参数说明：

message : 弹窗描述。

返回值:

无。

---

get_last_error (vector<string>:_return)

函数说明:

该函数返回机器人最新的错误列表

参数说明:

_return: 错误列表。

返回值:

无。

---

get_noneblock_taskstate (int id)

函数说明:

根据 id 查询当前的任务状态。

参数说明:

id: 任务的 id。

返回值:

任务的当前执行状态（任务状态列表详见第 1.2 章节)。

---

robotmoving ()

函数说明:

该函数用于判断机器人是否在运动。

参数说明:

无。

返回值:

True: 机器人在运动;

False: 机器人没有运动。

---

clear_error_message ()

函数说明:

清空报错信息并关闭弹窗。

参数说明:

无。

返回值:

无。

cal_traj_time (const TrajTimeParams& param)

函数说明:

计算目标运动类型从指定位置以某一几何路径运动到目标位置在指定最大速度/最大加速度约束下所需要的时间。

参数说明:

param : 参考 TrajTimeParams。

返回值:

无。

---

get_hardware_clock ( )

函数说明:

获取主板硬件时钟, 精确到 ms。

参数说明:

无。

返回值:

主板硬件时钟, 单位: ms。

1.3.9 Modbus

modbus_read (string signal_name)

函数说明:

该函数可读取 modbus 节点的数据, 返回值为 double 类型。

参数说明:

signal_name : modbus 节点名。

返回值:

modbus 节点返回值

---

modbus_write (string signal_name, int value)

函数说明:

该函数可对 modbus 节点进行写操作。

参数说明:

signal_name : modbus 节点名。

value : 要写入的数值, 寄存器节点取值为 0-65535 内的整数, 线圈节点取值为 0 或 1。

返回值:

返回当前任务结束时的状态。

---

modbus_set_frequency (string signal_name, int frequence)

函数说明:

该函数可修改 modbus 节点的刷新频率，默认频率为 10Hz。

参数说明:

signal_name : modbus 节点名。

frequence : 频率值，取值范围: 1~100Hz。

返回值:

无

示例:

```
modbus_set_frequency ("mbus1", 20)
```

---

modbus_write_multiple_regs (int slave_num, string name, int len, vector<int8_t> word_list)

函数说明:

该函数可对多寄存器进行写操作。

参数说明:

slave_num : modbus 节点号。

name : modbus 节点名。

len : 需要写入数据的寄存器长度。

word_list : 需要写入的数据。

返回值:

无

示例:

```
modbus_write_multiple_regs (1, "mbus1", 5, {1,2,3,4,5})
```

---

modbus_write_multiple_coils (int slave_num, string name, int len, vector<int16_t> byte_list)

函数说明:

该函数可对多线圈进行写操作。

参数说明:

slave_num : modbus 节点号。

name : modbus 节点名。

len : 需要写入数据的线圈长度。

byte_list ： 需要写入的数据。

返回值:

无

示例:

```
modbus_write_multiple_coils (2, "mbus1", 5, {1,1,1,1,1})
```

1.3.10 力控函数

fc_start ()

函数说明:

该指令控制机械臂开启机器人末端力控。开启末端力控后所有运动函数除正常运动外，会额外基于已配置的末端力控参数叠加末端力控运动实现力位混合运动。

参数说明:

无。

返回值:

返回值代表当前任务的 id 信息。

警告： 在使用 fc_start 函数启用末端力控功能前，需要严格确保机器人末端已正常安装力传感器，并在正确完成其通讯配置及安装位置设置启用。错误的通讯配置或未正确启用传感器会导致使用 fc_start 函数后机器人报错。错误的传感器安装位置参数会导致异常的末端力控运动，从而造成安全风险。

---

fc_stop ()

函数说明:

该指令控制机械臂退出末端力控，结束力位混合控制。

参数说明:

无。

返回值:

返回值代表当前任务的 id 信息。

警告： 当前在调用 fc_stop 函数退出末端力控前，需要严格保证所有机器人绝对位置运动已结束，即所有来自脚本以及外部接口所触发的机器人运动已结束。机器人运动过程中能够使用 fc_stop 函数结束末端力控，会引起机器人产生异常运动并报错。

---

fc_config (vector<bool> direction, vector<double> ref_ft, vector<double> damp, vector<double> max_vel, vector<double> number_list, string tool, string wobj, int type)

函数说明:

该指令修改并配置机器人末端力控参数。

参数说明:

direction:: 6 个笛卡尔空间方向末端力控开关，开为 true，关为 false。开启力控的笛卡尔空间方向，除了机器人末端原有绝对运动外，同时还会根据末端传感器感知到的外部力，基于其他力控参数做出对应力控运动调整。未开启力控的笛卡尔空间方向则保持原始机器人末端绝对运动。

ref_ft: 6 个笛卡尔空间方向末端力控目标维持力，范围 [-1000, 1000]，X/Y/Z 方向单位 N，RX/RY/RZ 方向单位 Nm，方向符号参考末端力控参考坐标系方向。该目标维持力会使机器人末端力控产生额外运动，直到机器人末端传感器感知到与外界接触力且达到目标维持力的大小。

damp: 6 个笛卡尔空间方向末端力控阻尼，X/Y/Z 方向单位 (N/(m/s)，RX/RY/RZ 方向单位 (Nm/(rad/s))。末端力控所产生的叠加运动最大速度，会根据传感器在笛卡尔空间启用力控方向上感知到的外力与目标维持力之间的偏差值，除以对应方向的阻尼值计算得到。

max_vel: 6 个笛卡尔空间方向末端力控最大调整速度，范围 [-5, 5]，X/Y/Z 方向单位 (m/s)，RX/RY/RZ 方向单位 (rad/s)。若基于末端传感器感知到的外力与目标维持力的偏差除以阻尼后计算得到的速度超过力控最大调整速度，则会以最大调整速度进行末端力控运动。该参数可以使末端力控运动在使用较小阻尼提升响应的基础上降低调整最大速度，从而保证末端力控运动的安全及稳定性。

number_list:6 个笛卡尔空间方向末端接触力死区，范围 [-1000, 1000]，X/Y/Z 方向单位 N，RX/RY/RZ 方向单位 Nm。若末端力传感器感知到外力与目标维持力的偏差绝对值小于接触力死区时，则会停止对于力控运动，直到由于外力变化再次使偏差值大于死区。

tool ：设置使用的末端力控工具的名称，默认为当前使用的工具。所有末端力控运动所产生的叠加运动，都会以 tool 坐标系的原点产生。若同时末端力控参考坐标系配置为工具坐标系，则力控运动笛卡尔方向的定义参考 tool 坐标系的笛卡尔方向。

wobj ：设置使用的末端力控工件坐标系的名称，默认为当前使用的工件坐标系。若末端力控参考坐标系配置为工件坐标系，则力控运动笛卡尔方向的定义参考 wobj 坐标系的笛卡尔方向。

type: 末端力控参考坐标系选择标志位，0 为参考工具坐标系，1 为参考工件坐标系。

返回值:

返回值代表当前任务的 id 信息。

---

fc_move ()

函数说明:

该指令控制机械臂没有绝对位置运动的基础上产生仅基于末端力控产生的补偿运动。

参数说明:

无。

返回值:

返回值代表任务结束时状态。

---

fc_guard_act (vector<bool> direction, vector<double> ref_ft, string tool, string wobj, int type,int force_property)

函数说明:

该指令控制机械臂在末端力控过程中进行力安全力监控。当监控力拆过所设定范围时，机器人会触发急停并报警。

参数说明:

direction:: 6 个笛卡尔空间方向末端力安全监控开关，开为 true，关为 false。

ref_ft: 6 个笛卡尔空间方向末端力安全监控最大力，X/Y/Z 方向单位 N，RX/RY/RZ 方向单位 Nm。该值未监控最大力范围绝对值，不区分所在自由度上的 ± 方向。

tool ： 设置使用的末端力安全监控工具的名称，默认为当前使用的工具。若末端力控参考坐标系配置为工具坐标系，则安全力监控的笛卡尔方向的定义参考 tool 坐标系的笛卡尔方向。

wobj ： 设置使用的末端力安全监控工件坐标系的名称，默认为当前使用的工件坐标系。若末端力控参考坐标系配置为工件坐标系，则安全力监控的笛卡尔方向的定义参考 wobj 坐标系的笛卡尔方向。

type: 末端力安全监控参考坐标系选择标志位，0 为参考工具坐标系，1 位参考工件坐标系。

force_property ： 监控力属性，0 为末端负载力及外力，1 为末端外力（不含负载），可缺省，默认为 0。

返回值:

返回值代表当前任务的 id 信息。

---

fc_guard_deact ()

函数说明:

该指令控制机械臂在末端力控过程中禁用力安全力监控。

参数说明:

无。

返回值:

返回值代表当前任务的 id 信息。

---

fc_force_set_value (vector<bool> direction, vector<double> ref_ft)

函数说明:

该指令控制机械臂末端力传感器读数设置为指定值。

参数说明:

direction: 6 个末端力传感器输出力设置标志位，需要设置为 true，不需要设置为 false。需要注意的是，该方向定义未传感器自身力坐标系方向。

ref_ft: 6 个末端力传感器输出力设置目标值，X/Y/Z 方向单位 N，RX/RY/RZ 方向单位 Nm。

返回值:

返回值代表当前任务的 id 信息。

---

fc_wait_pos (vector<double> middle_value, vector<double> range, bool absolute, int duration, int timeout)

函数说明:

该指令控制机械臂在执行 fc_start() 函数后的末端力控过程中满足指定位置判断条件时自动停止当前运动函数并跳过后续运动函数，直到 fc_stop() 函数被执行停止末端力控或再次调用 fc_wait_logic 函数复位力控条件判断。

参数说明:

middle_value: 位置判断条件绝对值，X/Y/Z 方向单位 m，RX/RY/RZ 方向单位 (rad)。

range: 位置判断条件范围大小，X/Y/Z 方向单位 m，RX/RY/RZ 方向单位 (rad)。由于位置信息存在波动与偏差，适当设定范围大小可以有效保证逻辑触发成功率。

absolute: 绝对/增量条件判断标志位，true 为绝对位置判断，false 为增量位置判断。增量判断会以最后一次调用 fc_wait_logic 函数复位力控条件判断时机器人所在位置为参考点。

duration: 条件满足触发保持时间，单位 ms。

timeout: 条件满足触发超时时间，单位 ms。起始计算时间未最后一次调用 fc_wait_logic 函数复位力控条件判断时刻。

返回值:

返回值代表当前任务的 id 信息。

---

fc_wait_vel (vector<double> middle_value, vector<double> range, bool absolute, int duration, int timeout)

函数说明:

该指令控制机械臂在执行 fc_start() 函数后的末端力控过程中满足指定速度判断条件时自动停止当前运动函数并跳过后续运动函数，直到 fc_stop() 函数被执行停止末端力控或再次调用 fc_wait_logic 函数复位力控条件判断。

参数说明:

middle_value: 速度判断条件绝对值，X/Y/Z 方向速度范围 [-5, 5]，单位 (m/s)，RX/RY/RZ 方向速度范围 [-2*PI, 2*PI]，单位 (rad/s)。

range: 速度判断条件偏移范围大小，X/Y/Z 方向单位 (m/s)，RX/RY/RZ 方向单位 (rad/s)。由于速度信息存在波动与偏差，适当设定范围大小可以有效保证逻辑触发成功率。

absolute: 绝对/增量条件判断标志位，true 为绝对速度判断，false 为增量速度判断。增量判断会以最后一次调用 fc_wait_logic 函数复位力控条件判断时机器人末端速度为参考基准。

duration: 条件满足触发保持时间，单位 ms。

timeout: 条件满足触发超时时间，单位 ms。起始计算时间未最后一次调用 fc_wait_logic 函数复位力控条件判断时刻。

返回值:

返回值代表当前任务的 id 信息。

---

fc_wait_ft (vector<double> middle_value, vector<double> range, bool absolute, int duration, int timeout)

函数说明:

该指令控制机械臂在执行 fc_start() 函数后的末端力控过程中满足指定力判断条件时自动停止当前运动函数并跳过后续运动函数，直到 fc_stop() 函数被执行停止末端力控或再次调用 fc_wait_logic 函数复位力控条件判断。

参数说明:

middle_value: 力判断条件绝对值，范围 [-1000, 1000]，X/Y/Z 方向单位 N，RX/RY/RZ 方向单位 Nm。

range: 力判断条件偏移范围大小，X/Y/Z 方向单位 N，RX/RY/RZ 方向单位 Nm。。由于力信息存在波动与偏差，适当设定范围大小可以有效保证逻辑触发成功率。

absolute: 绝对/增量条件判断标志位，true 为绝对力判断，false 为增量力判断。增量判断会以最后一次调用 fc_wait_logic 函数复位力控条件判断时机器人末端感知力为参考基准。

duration: 条件满足触发保持时间，单位 ms。

timeout: 条件满足触发超时时间，单位 ms。起始计算时间未最后一次调用 fc_wait_logic 函数复位力控条件判断时刻。

返回值:

返回值代表当前任务的 id 信息。

---

fc_wait_logic (vector<int> logic)

函数说明:

该指令控制机械臂在执行 fc_start() 函数后的末端力控过程中位置条件判断、速度条件判断与力条件判断间的逻辑关系。不配置时默认三个条件判断都禁用。当已通过

参数说明:

logic: 三维整形列表，0 代表不启用，1 代表与逻辑，2 代表或逻辑。例如开启位置条件判断，禁用速度条件判断，开启力条件判断，并且位置与力的关系为或，则输入 [1,0,2]。

返回值:

返回值代表当前任务的 id 信息。

---

fc_get_ft (vector<double> data)

函数说明:

该指令用以获取当前机器人末端传感器的原始反馈读数。该读数经过去皮操作，当机器人上下使能或调用 fc_force_set_value 时，会根据机器人系统中设置的负载参数（默认负载安装在传感器力检测端）以及目标参考传感器输出度数的大小进行去皮。

参数说明:

data: 6 自由度末端力读数，X/Y/Z 方向单位 (N，RX/RY/RZ 方向单位 (Nm)。

返回值:

无。

---

fc_mode_is_active ()

函数说明:

该指令用以获取当前机器人末端力控功能启用状态。需要注意的是，即使通过该函数获取到当前机器人末端处于力控状态下，也需要机器人当前存在正在执行的绝对运动函数，例如 movel、movec、fc_move 等，才会产生对应的末端力控运动。

参数说明:

无。

返回值:

机器人末端力控启用返回 true，未启用返回 false。

1.3.11 运动优化函数

enable_speed_optimization ()

函数说明:

该指令控制机械臂开启速度优化功能。开启该功能后，机器人会参考当前模式下的安全限制参数，以不会违反安全限制参数为基础实时优化机器人速度控制指令，以达到最优速度节拍。速度优化功能仅在机器人自动模式下有效。当机器人开启速度优化功能并且处于自动模式下，原始机器人程序中所设定的关节最大速度与末端最大速度参数不再生效。

参数说明:

无。

返回值:

阻塞执行，返回值代表当前任务结束时的状态。

---

disable_speed_optimization ()

函数说明:

该指令用以控制机械臂退出速度优化。

参数说明:

无。

返回值:

阻塞执行，返回值代表当前任务结束时的状态。

---

enable_acc_optimization ()

函数说明:

该指令控制机械臂开启加速度优化功能。开启该功能后，系统会根据机器人动力学模型、机械功率模型及电功率模型计算机器人起停最优加速度，在满足速度约束前提下，机械臂以尽可能高的加速度进行规划。当速度优化同时打开后，系统默认同时开启加速度优化功能，该函数不在生效。

参数说明:

无。

返回值:

阻塞执行，返回值代表当前任务结束时的状态。

---

disable_acc_optimization ()

函数说明:

该指令用以控制机械臂退出加速度优化。

参数说明:

无。

返回值:

阻塞执行，返回值代表当前任务结束时的状态。

---

enable_singularity_control()

函数说明:

该指令用以开启机械臂奇异点规避功能。开启奇异规避功能后，若机器人后续运动过程为笛卡尔空间运动且运动过程中会穿过机器人预先定义好的奇异空间，则会自动切换到关节空间过度运动，从而避免机器人经过奇异空间过程中引发的失速问题。在奇异规避过程中，机器人末端仅能尽可能保证末端与原始路径的一致性，实际会产生一定程度的精度损失。

参数说明:

无。

返回值:

任务结束时状态。

---

disable_singularity_control()

函数说明:

该指令用以关闭机械臂奇异点规避功能。关闭该功能时，若机器人在笛卡尔空间运动过程中经过奇异区域，则会有失速风险，且机器人在接近奇异区域后会触发停机报警。

参数说明:

无。

返回值:

任务结束时状态。

! 注意

由于奇异点规避功能与振动抑制功能都是通过对指令轨迹进行优化处理而实现，因此这两个功能无法同时使用。当前默认振动抑制的优先级高于奇异点规避。因此，使用奇异点规避功能需保证振动抑制功能关闭。

---

enable_vibration_control()

函数说明:

该指令用以开启对机械臂起停振动控制功能进行优化。该功能在机器人系统启动时默认开启，以最大程度的保证机器人运动起停稳定性。需要注意的是，该功能开启后，机器人每一次独立运动都会有短暂的节拍降低（约 200-400ms）。

参数说明:

无。

返回值:

任务结束时状态。

---

disable_vibration_control()

函数说明:

该指令用以退出对机械臂末端振动的优化。若在机器人实际调试过程中受限于连续独立短轨迹运行节拍，且对机器人末端起停稳定性要求较低，可以通过使用该函数临时禁用振动控制功能。需要注意的是，调用该函数后，机器人在程序运行结束后并不会自动复位并开启振动控制功能，因此若需要依旧保持手动试教过程中机器人运动起停稳定性，应再次调用 enable_vibration_control 函数开启振动控制功能。

参数说明:

无。

返回值:

任务结束时状态。

1.3.12 轨迹池函数

机器人控制器中，内置一组最大数量为 1000 个 points 的轨迹池，轨迹池遵循先入先出原则，可通过以下函数对轨迹池进行操作。

---

trackEnqueue (vector< vector<double> > points, bool block)

函数说明:

该函数将一组 points 点位信息输入到机器人控制器中的轨迹池。轨迹池最大可容纳点位信息个数为 1000。在使用轨迹池跟随过程中，轨迹池中的点位会阶段性被提取并执行。因此为了保证使用轨迹池跟随过程中，需要周期性确认当前轨迹池中现有点位数量，并进行及时补充，从而避免产生中间产生不连续的（速度降速到 0）的跟随运动。

参数说明:

points : 一组 points 点位信息。每个 point 以 6 个 double 类型数据构成。point 的定义最终由所调用的轨迹池跟随运动类型决定。

block : 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。如果返回值为-1，表示轨迹池已满，无法加入新点位。

---

trackClearQueue ()

函数说明:

该函数用于将机器人控制器中的轨迹池清空。

参数说明:

无。

返回值:

阻塞执行，返回值代表当前任务结束时的状态。

---

getQueueSize ()

函数说明:

该函数用于获取机器人控制器中的当前轨迹池未被提取的点位数量大小。

参数说明:

无。

返回值:

阻塞执行，返回值为当前轨迹池大小。

---

trackJointMotion (double speed, double acc, bool block)

函数说明:

该函数执行时，关节轨迹池跟随功能将会周期性的从轨迹池队列中现有点位中提取点位信息，并将其定义为关节位置信息处理。机器人的各关节将顺序到达轨迹池中的点位值直到轨迹池中无新的点位。执行过程中，主导关节（关节位置变化最大的关节）将以 speed 与 acc 规划运动，其他关节按比例缩放。

注：如果已经开始执行停止规划，将不再重新获取轨迹池中的数据，直到完成停止。停止后如果轨迹池中有新的点位，将重新执行跟随。为保证运动连续性，建议至少保证轨迹池中有 10 个数据。

参数说明:

speed ： 最大关节速度，范围 [0.01*PI/180, 1.25*PI]，单位 (rad/s)。

acc ： 最大关节加速度，范围 [0.01*PI/180, 12.5*PI]，单位 (rad/s2)。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态;

当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

trackCartMotion (double speed, double acc, bool block, string wobj, string tool)

函数说明:

该函数执行时，笛卡尔轨迹池跟随功能将会周期性的从轨迹池队列中现有点位中提取点位信息，并将其定义为机器人末端工具在参考工件坐标系中位姿位置信息处理。机器人的工具末端 tool 将顺序到达轨迹池中的点位值直到轨迹池中无新的点位。执行过程中，工具末端 tool 将以 speed 与 acc 在工件坐标系 wobj 下规划运动。

注：如果已经开始执行停止规划，将不再重新获取轨迹池中的数据，直到完成停止。停止后如果轨迹池中有新的点位，将重新执行跟随。为保证运动连续性，建议至少保证轨迹池中有 10 个数据。

参数说明:

speed ： 最大末端线速度，范围 [0.00001, 5]，单位 (m/s)。

acc ： 最大末端线加速度，范围 [0.00001, ∞]，单位 (m/s2)。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

wobj ： 设置使用的工件坐标系的名称，为空时默认为当前使用的工件坐标系。

tool ： 设置使用的工具的名称，为空时默认为当前使用的工具。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态;

当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

trackEnqueueOp (vector<PointOP> & points, bool block)

函数说明:

该函数将一组 PointOP 点位信息输入到机器人控制器中的轨迹池。该韩函数与 trackEnqueue 函数功能使用方法保持一致，区别在于在点位的基础上额外添加了每个点位的 Op 信息。该函数与 trackEnqueue 函数共享同一轨迹池，因此无论使用 trackEnqueue 函数还是 trackEnqueueOp 函数向轨迹池中添加点位信息时，需要同步周期性通过 getQueueSize 获取当前队列中未被执行点位数量信息，以确保轨迹池跟随运动的连续性以及不超过队列池大小。

参数说明:

points ： 一组 PointOP 点位信息和对应的 OP 操作。每个元素以 6 个 double 类型的点位和该点的 OP 操作组成。point 的定义最终由所调用的轨迹池跟随运动类型决定。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。如果返回值为-1，表示轨迹池已满，无法加入新点位。

---

track_enqueue_op_vel (const std::vector<PointOP> & track, bool block)

函数说明:

将一组 points 点位及对应速度 (m/s)、融合时间 (ms)、停留时间 (ms) 和该点位下的 OP 信息输入到机器人控制器中的轨迹池。

参数说明:

track ： 一组 PointOP 点位信息和对应的 OP 操作。每个元素以 6 个 double 类型的点位和该点的 OP 操作组成。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id)

函数查询当前任务的执行状态。如果返回值为-1，表示轨迹池已满，无法加入新点位。

---

track_cart_vel_motion (double acc, const std::string& tool, const std::string& wobj, bool block)

函数说明：

执行时，机器人的工具末端 tool 将顺序到达轨迹池中的点位值直到轨迹池中无新的点位。执行过程中，工具末端 tool 将以 speed 与 acc 在工件坐标系 wobj 下规划运动。

> 注：如果已经开始执行停止规划，将不再重新获取轨迹池中的数据，直到完成停止。停止后如果轨迹池中有新的点位，将重新执行跟随．为保证运动连续性，建议至少保证轨迹池中有 10 个数据。

参数说明：

acc ：最大末端加速度，单位：m/s^2

tool ：设置使用的工件坐标系的名称，为空字符串时默认为当前使用的工件坐标系。

wobj ：设置使用的工具的名称，为空字符串时默认为当前使用的工具。

block ：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值：

当配置为阻塞执行，返回值代表当前任务结束时的状态，当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。如果返回值为-1，表示轨迹池已满，无法加入新点位。

1.3.13 复合运动函数

combine_motion_config (int type, int ref_plane, int fq, double amp, double el_offset, double az_offset, double up_height, const vector<int>& time, bool path_dwell = false, const vector<Op>& op_list = {})

函数说明：

该指令控制机器人在原始运动轨迹上额外叠加一个复合运动，被叠加的符合运动为一周期性运动轨迹，且可以被几何描述。

参数说明：

type ：复合运动轨迹几何类型。1：平面三角形轨迹，2：平面正弦轨迹，3：平面圆形轨迹，4：平面梯形轨迹，5：平面 8 字形轨迹

ref_plane ：参考平面，0：工具 XOY 平面，1：工具 XOZ 平面，2：工具 YOZ 平面，3：工件 XOY 平面，4：工件 XOZ 平面，5：工件 YOZ 平面。

fq ：频率，单位（Hz）。

amp ：振幅，单位（m）。

el_offset ： 仰角偏移，单位（m）。（参数预留）

az_offset ： 方向角偏移，单位（m）。（参数预留）

up_height ： 中心隆起高度，单位（m）。（参数预留）

time ： 左右停留时间。

path_dwell ： 主路径同步停留，默认为 false，可缺省。

op_list ： 二维的 OP 参数列表，默认为空，可缺省。

返回值：

任务结束时状态。

---

enable_combined_motion ()

函数说明：

该指令会以最近一次使用 combine_motion_config 脚本设置的复合运动参数开启复合运动。

参数说明：

无。

返回值：

任务结束时状态。

---

disable_combined_motion ()

函数说明：

该指令用以结束复合运动。

参数说明：

无。

返回值：

任务结束时状态。

备注： 当且仅当机器人停止时，可以对复合运动是否启用进行开关，无法在轨迹运动过程中动态开启或关闭。

1.3.14 外部轴控制函数说明

enable_eaxis_scheme (string scheme_name)

函数说明：

该指令用于启动外部轴方案。

参数说明：

scheme_name ： 外部轴方案名称。

返回值:

任务结束时的状态。

---

disable_eaxis_scheme (string scheme_name)

函数说明:

该指令用于结束外部轴方案。

参数说明:

scheme_name : 外部轴方案名称。

返回值:

任务结束时的状态。

---

move_eaxis (std::vector<std::string> scheme_name, const std::vector<std::vector<double> > epose, const std::vector<double> vel, bool block, const Op op = op_, bool def_acc = true)

函数说明:

该指令用于控制单个或多个外部轴移动。

参数说明:

scheme_name : 目标外部轴方案名称列表。

epose : 目标外部轴方案所对应自由度位置 (三维) 列表, 记录位置自由度及单位根据外部轴方案所设置自由度及外部轴方案类型改变, 单位 (rad 或 m)。

vel : 外部轴最大规划速度列表, 根据对应外部轴方案类型改变, 单位 (rad/s 或 m/s)。

block : 指令是否阻塞型指令, 如果为 false 表示非阻塞指令, 指令会立即返回。

op : 详见上方 Op 特殊类型说明 (距离触发无效), 可缺省参数。

def_acc : 是否使用自定义加速度, 默认为 true, 可缺省参数。

返回值:

当配置为阻塞执行, 返回值代表当前任务结束时的状态;

当配置为非阻塞执行, 返回值代表当前任务的 id 信息, 用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

move_jog_eaxis (vector<double> const string& scheme_name, int direction, double vel, bool block)

函数说明:

该指令用于控制外部轴和机器人执行点动。

参数说明:

scheme_name ： 目标外部轴方案名称。

direction ： 运动方向，-1：负方向，1：正方向。

vel ： 速度百分比。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态;

当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

movej2_eaxis (vector<double> joints_list, double v, double a, double rad, std::vector<std::string> scheme_name, const std::vector<std::vector<double> > epose, const std::vector<double> eaxis_v, bool block, Op op = op_, bool def_acc = true)

函数说明:

该指令用于控制外部轴和机器人执行关节运动。

参数说明:

joint_list ： 目标关节位置，单位 (rad)。

v ： 关节角速度，单位 (rad/s)。

a ： 关节加速度，单位 (rad/s^2)。

rad ： 融合半径，单位 (m)。

scheme_name ： 目标外部轴方案名称列表。

epos ： 目标外部轴方案所对应自由度位置 (三维) 列表，记录位置自由度及单位根据外部轴方案所设置自由度及外部轴方案类型改变，单位 (rad 或 m)。

eaxis_v ： 外部轴最大规划速度列表，根据对应外部轴方案类型改变，单位 (rad/s 或 m/s)。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op ： 详见上方 Op 特殊类型说明 (距离触发无效)，可缺省参数。

def_acc ： 是否使用自定义加速度，默认为 true，可缺省参数。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态;

当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

movej2_pose_eaxis (vector<double> p, double v, double a, double rad, vector<double> q_near, string tool, string wobj, std::vector<std::string> scheme_name, const std::vector<std::vector<double> > epose, const std::vector<double> eaxis_v, bool block, Op op = op_, bool def_acc = true)

函数说明:

该指令用于控制外部轴和机器人从当前状态，按照关节运动的方式移动到末端目标位置。

参数说明:

p ： 对应末端的位姿，位置单位 (m)，姿态以 Rx、Ry、Rz 表示，范围 [-2*PI, 2*PI]，单位 (rad)。

v ： 关节角速度，单位 (rad/s)。

a ： 关节加速度，单位 (rad/s^2)。

rad ： 融合半径，单位 (m)。

q_near ： 目标点附近位置对应的关节角度，用于确定逆运动学选解，为空时使用当前位置。

tool ： 设置使用的工具的名称，为空时默认为当前使用的工具。

wobj ： 设置使用的工件坐标系的名称，为空时默认为当前使用的工件坐标系。

scheme_name ： 目标外部轴方案名称列表。

epos ： 目标外部轴方案所对应自由度位置（三维）列表，记录位置自由度及单位根据外部轴方案所设置自由度及外部轴方案类型改变，单位 (rad 或 m)。

eaxis_v ： 外部轴最大规划速度列表，根据对应外部轴方案类型改变，单位 (rad/s 或 m/s)。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op ： 详见上方 Op 特殊类型说明 (距离触发无效)，可缺省参数。

def_acc ： 是否使用自定义加速度，默认为 true，可缺省参数。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态;

当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

movel_eaxis (vector<double> p, double v, double a, double rad, vector<double> q_near, string tool, string wobj, std::vector<std::string> scheme_name, const std::vector<std::vector<double> > epose, const std::vector<double> eaxis_v, bool block, Op op = op_, bool def_acc = true)

函数说明:

该指令用于控制外部轴和机器人从当前状态按照直线路径移动到目标状态。

参数说明:

p : 对应末端的位姿，位置单位 (m)，姿态以 Rx、Ry、Rz 表示，范围 [-2*PI, 2*PI]，单位 (rad)。

v : 末端速度，范围 (0, 5]，单位 (m/s)。

a : 末端加速度，范围 (0, ∞]，单位 (m/s^2)。

rad : 关节融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。

q_near : 目标点附近位置对应的关节角度，用于确定逆运动学选解，为空时使用当前位置。

tool : 设置使用的工具的名称，为空时默认为当前使用的工具。

wobj : 设置使用的工件坐标系的名称，为空时默认为当前使用的工件坐标系。

scheme_name : 目标外部轴方案名称列表。

epos : 目标外部轴方案所对应自由度位置 (三维) 列表，记录位置自由度及单位根据外部轴方案所设置自由度及外部轴方案类型改变，单位 (rad 或 m)。

eaxis_v : 外部轴最大规划速度列表，根据对应外部轴方案类型改变，单位 (rad/s 或 m/s)。

block : 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op : 详见上方 Op 特殊类型说明 (距离触发无效)，可缺省参数。

def_acc : 是否使用自定义加速度，默认为 true，可缺省参数。

arc_rad : 圆弧弧度，为 0 时运动到 p2 结束点，按照原始圆弧规划，其余情况按照数据圆弧角规划 (此模式姿态仅支持起点一致和受圆心约束)，单位: rad. 可缺省，默认为 0。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态;

当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

movec_eaxis (vector<double> p1, vector<double> p2, double v, double a, double rad, vector<double> q_near, string tool, string wobj, std::vector<std::string> scheme_name, const std::vector<std::vector<double> > epose, const std::vector<double> eaxis_v, bool block, Op op = op_, bool def_acc = true)

函数说明:

该指令用于控制外部轴和机器人做圆弧运动，起始点为当前位姿点，途径 p1 点，终点为 p2 点。

参数说明:

p1 ： 圆弧运动中间点位姿。

p2 ： 圆弧运动结束点位姿。

v ： 末端速度，范围 (0, 5]，单位 (m/s)。

a ： 末端加速度，范围 (0, ∞]，单位 (m/s^2)。

rad ： 关节融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合。

q_near ： 目标点附近位置对应的关节角度，用于确定逆运动学选解，为空时使用当前位置。

tool ： 设置使用的工具的名称，为空时默认为当前使用的工具。

wobj ： 设置使用的工件坐标系的名称，为空时默认为当前使用的工件坐标系。

scheme_name ： 目标外部轴方案名称列表。

epos ： 目标外部轴方案所对应自由度位置 (三维) 列表，记录位置自由度及单位根据外部轴方案所设置自由度及外部轴方案类型改变，单位 (rad 或 m)。

eaxis_v ： 外部轴最大规划速度列表，根据对应外部轴方案类型改变，单位 (rad/s 或 m/s)。

block ： 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回。

op ： 详见上方 Op 特殊类型说明 (距离触发无效)，可缺省参数。

def_acc ： 是否使用自定义加速度，默认为 true，可缺省参数。

返回值:

当配置为阻塞执行，返回值代表当前任务结束时的状态;

当配置为非阻塞执行，返回值代表当前任务的 id 信息，用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

---

get_eaxis_info (std::vector<EAxisInfo> & info)

函数说明:

获取外部轴的当前信息。

参数说明:

info ： 所有外部轴方案，以及对应的位置，状态，速度，加速度。

返回值:

无。

1.3.15 实时控制函数

1. 用户需确保客户端的实时性和通讯质量;
2. 由于存在不同电脑间实时性同步的问题, 该功能无法做到强实时;
3. Python 版本不提供实时控制接口。

---

start_realtime_mode (int32_t mode, double filter_bandwidth, double com_lost_time)

函数说明:

开启实时控制模式。

参数说明:

mode : 实时控制模式, 0: 关节位置, 1: 关节速度, 2: 关节力矩, 3: 空间位置, 4: 空间速度。

filter_bandwidth : 实时控制指令滤波器带宽, 单位 Hz, 默认 100Hz。

com_lost_time : 实时控制通讯数据丢失监控保护时间, 单位 s, 默认 0.02s。

返回值:

阻塞执行, 返回值代表当前任务结束时的状态。

---

end_realtime_mode ()

函数说明:

结束实时控制模式。

参数说明:

无。

返回值:

阻塞执行, 返回值代表当前任务结束时的状态。

---

realtime_data_enqueue (const std::vector<RealTimeData> &realtime_data, bool block)

函数说明:

实时数据入队。

参数说明:

realtime_data : 实时数据。

block : 是否阻塞, 如果为 false 表示非阻塞指令, 指令会立即返回。

返回值:

当配置为非阻塞执行, 返回值代表当前任务的 id 信息, 用户可以调用 get_noneblock_taskstate(id) 函数查询当前任务的执行状态。

clear_realtime_data_queue ()

函数说明:

清空实时数据队列。

参数说明:

无。

返回值:

阻塞执行，返回值代表当前任务结束时的状态。

---

get_realtime_data_queue_size ()

函数说明:

获取当前实时队列池数据的数量。

参数说明:

无。

返回值:

当前实时队列池数据的数量。

调用逻辑

1. 调用 clear_realtime_data_queue() 清空当前实时控制指令队列;
2. RPC 控制端本地生成对应的实时控制指令数据，并将其赋值给 RealTimeData 中对应指令数据，status 指令赋 true;
3. 调用 realtime_data_enqueue 将赋值好的数据推送进实时控制指令队列，确保推送进实时控制队列池的第一个指令与当前机器人实际状态相符，避免开启实时控制模式后出现飞车报错;
4. 重复步骤 2-3 直至完整控制指令推送至实时控制队列中或队列已满;
5. 调用 start_realtime_mode 启用实时控制模式，机器人进入实时控制模式，并参考对应实时控制模式解析控制指令队列中内容并进行跟随;
6. 若还有未完成推送的实时指令，在机器人运动过程中可以通过 get_realtime_data_queue_size() 在线查询实时控制队列池的大小，并进行新指令的在线推送更新。

示例

以关节速度控制为例:

```
std::vector< std::vector<double> > joints_vel = {
{ -6.70021746007427193e-04, -4.07213725421849866e-03,
7.06659303126689307e-03, -2.97873519057115406e-03,
2.18309444235197414e-05, -6.69850452906270801e-04 },
```

```
{ -1.589154109004034272e-03, -1.011421833810334688e-02,
1.754193265331701820e-02, -7.390421055007160601e-03,
5.177329125210710173e-05, -1.588748280197681967e-03 },

{ -2.755305715766451745e-03, -1.811209469461632846e-02,
3.140150170886479852e-02, -1.322473808616756764e-02,
8.975891577376789718e-05, -2.754602640998529955e-03 },

{ -3.442737990430701396e-03, -2.315600008944551300e-02,
4.013747069438520360e-02, -1.690065907568836720e-02,
1.121474983595938492e-04, -3.441859692247766726e-03 },

{ -3.157588346910660156e-03, -2.189245123375168842e-02,
3.793907058813744682e-02, -1.597249170914913363e-02,
1.028521444952658879e-04, -3.156782645645519157e-03 },

{ -2.864607599016891562e-03, -2.071522367525847202e-02,
3.588861821868741253e-02, -1.510613530488653065e-02,
9.330159342950848850e-05, -2.863877683525377273e-03 },

{ -2.820140424052270788e-03, -2.131761007776102057e-02,
3.692116291809591222e-02, -1.553733028238165649e-02,
9.184682028243494134e-05, -2.819424392840391355e-03 },

{ -2.812250163455366943e-03, -2.205552289748904243e-02,
3.818995917565090603e-02, -1.606839114406481001e-02,
9.158444035047173999e-05, -2.811536045264771428e-03 },

{ -2.728426654789113497e-03, -2.205617043400752084e-02,
3.818376395297849724e-02, -1.606350627152726418e-02,
8.884967501704240018e-05, -2.727728984672162828e-03 },

{ -1.936938097461126220e-03, -2.279194264874751311e-02,
3.938230929954751602e-02, -1.654479502939042515e-02,
6.304779778496766665e-05, -1.936393526134143209e-03 } };

// 机器人已经上电上使能

// 清空实时数据队列

duco_cobot.clear_realtime_data_queue();

vector<RealTimeData> real_data_vec;

// 将关节速度赋值给实时数据结构体变量

for (int i = 0; i < joints_vel.size(); i++)

{

    RealTimeData rt_data;

    rt_data.joint_vel_cmd = joints_vel[i];

    // 数据更新状态为 true

    rt_data.status = true;

    real_data_vec.push_back (rt_data);

}
```

```
// 将数据传进队列
duco_cobot.realtime_data_enqueue( real_data_vec, true );
// 以关节速度控制方式打开实时控制模式
result = duco_cobot.start_realtime_mode( 1, 100, 0.04 );
// 不断地向队列填充数据
while ( ! stop_realtime_control )
{
    real_data_vec.clear();
    RealTimeData rt_data;
    rt_data.joint_vel_cmd = joints_vel[joints_vel.size()
    - 1];
    rt_data.status = true;
    real_data_vec.push_back(rt_data);
    duco_cobot.realtime_data_enqueue(real_data_vec,
    true);
    usleep(1000)
}
// 关闭实时控制模式
duco_cobot.end_realtime_mode();
```

1.3.16 激光跟踪控制

enable_laser_track ()

函数说明:

开启激光跟踪功能。

参数说明:

无。

返回值:

非阻塞执行，返回当前任务的 id 信息。

---

disable_laser_track ()

函数说明:

关闭激光跟踪功能。

参数说明:

无。

返回值:

任务结束时的状态。

---

set_laser_track_params(const int32_t plane, const int32_t freq, const int32_t lead_len, const std::vector<double> & active, const std::vector<double> & kp, const std::vector<double> & kd, const std::vector<double> & offset, const std::vector<double> & offset_vel, const std::vector<double> & filter_coe, const bool path_length_correction, const bool track_type)

函数说明:

设置激光跟踪功能相关参数。

参数说明:

plane : 补偿量在工具坐标系中所在平面, 0 : X-Y 平面, 1 : X-Z 平面, 2 : Y-Z 平面。

freq : 数据更新参考频率。

lead_len : 获取的偏差量在主路径上的提前距离, 单位 (m) , (暂时不需要)。

active : 补偿量方向开关, 参考对应补偿平面, 变量顺序为 X-Y-Z 中两个平面所在方向顺序, 0 : 关闭, 1 : 打开。

kp : 补偿方向上的控制 kp 系数。

kd : 补偿方向上的控制 kd 系数。

offset : 补偿方向上最大补偿偏移量, 单位 (m)。

offset_vel : 补偿方向上最大补偿速度, 单位 (m/s)。

filter_coe : 补偿方向上补偿量滤波系数。

path_length_correction : 路径长度修正。

track_type true : 表示使用记录的数据。

返回值:

返回值代表任务结束时状态。

---

reset_laser_track ()

函数说明:

清空记录的偏移数据。

参数说明:

无。

返回值:

返回值代表任务结束时状态。

---

write_laser_search_pos(std::vector<double> pos)

函数说明:

添加基于世界坐标系偏移的点位。

参数说明:

pos : 点位。

返回值:

返回值代表任务结束时状态。

1.3.17 其他信息

get_version ()

函数说明:

获取当前 RPC 库的版本号。

参数说明:

无。

返回值:

当前 RPC 库的版本号。

---

get_robot_version(string& version)

函数说明:

获取机器人控制器的软件版本号。

参数说明:

version : 软件版本号。

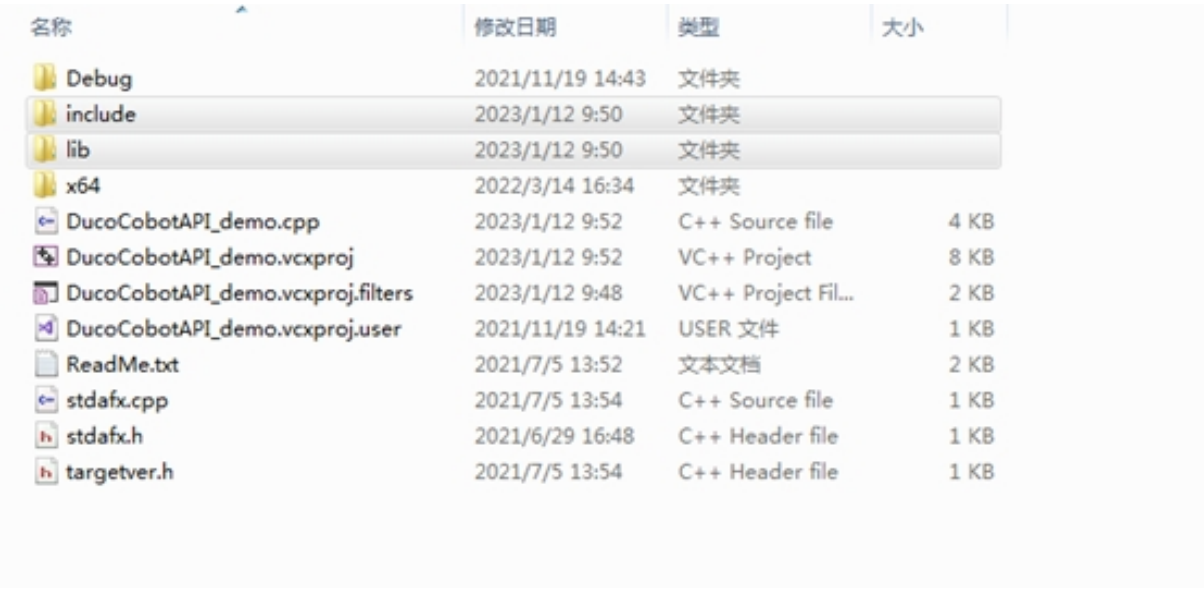
返回值:

无。

1.4 Visual Studio 引用远程接口库的说明

使用 Visual Stuadio 开发时，可参考如下设置方式调用远程接口库:

1、文件结构

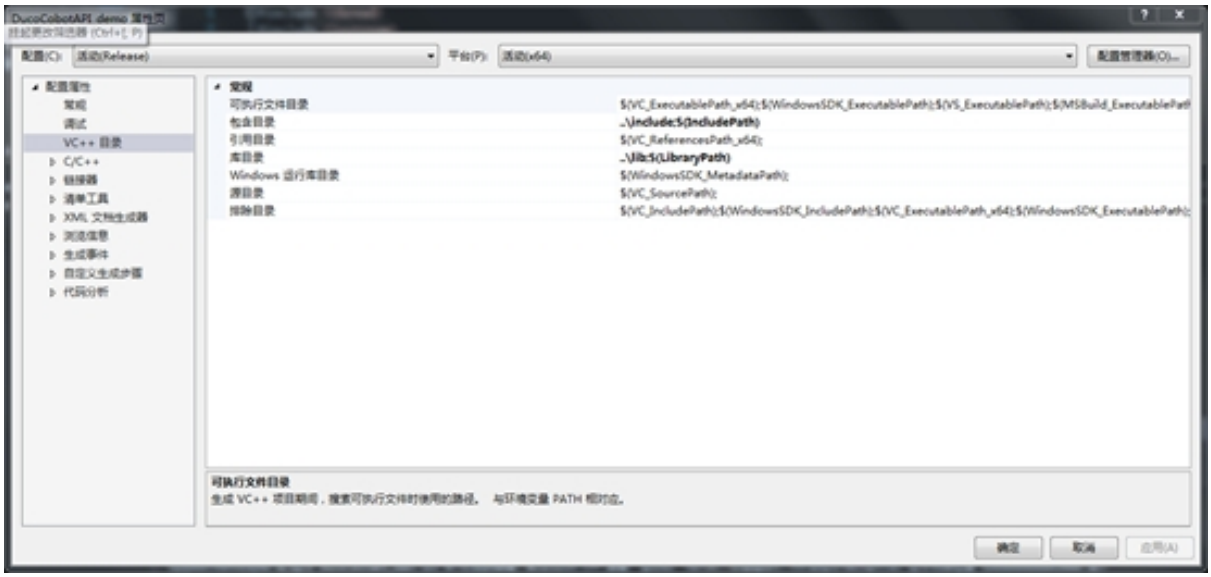


名称	修改日期	类型	大小
Debug	2021/11/19 14:43	文件夹	
include	2023/1/12 9:50	文件夹	
lib	2023/1/12 9:50	文件夹	
x64	2022/3/14 16:34	文件夹	
DucoCobotAPI_demo.cpp	2023/1/12 9:52	C++ Source file	4 KB
DucoCobotAPI_demo.vcxproj	2023/1/12 9:52	VC++ Project	8 KB
DucoCobotAPI_demo.vcxproj.filters	2023/1/12 9:48	VC++ Project Fil...	2 KB
DucoCobotAPI_demo.vcxproj.user	2021/11/19 14:21	USER 文件	1 KB
ReadMe.txt	2021/7/5 13:52	文本文档	2 KB
stdafx.cpp	2021/7/5 13:54	C++ Source file	1 KB
stdafx.h	2021/6/29 16:48	C++ Header file	1 KB
targetver.h	2021/7/5 13:54	C++ Header file	1 KB

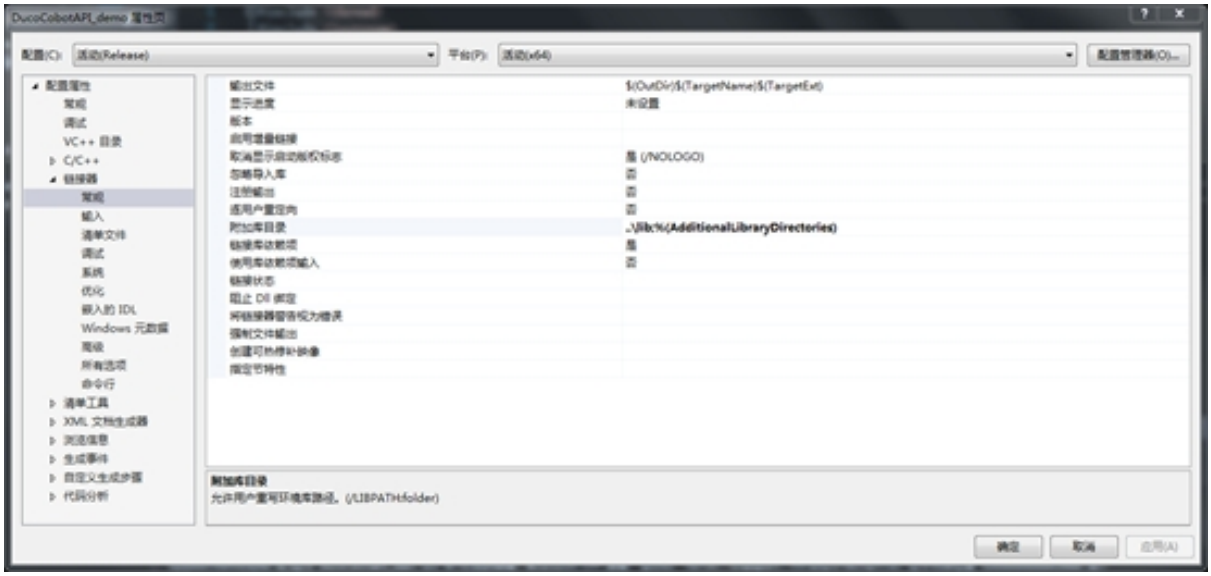
include 文件夹里是远程接口的头文件，lib 文件里是远程接口的库文件。

2、属性设置

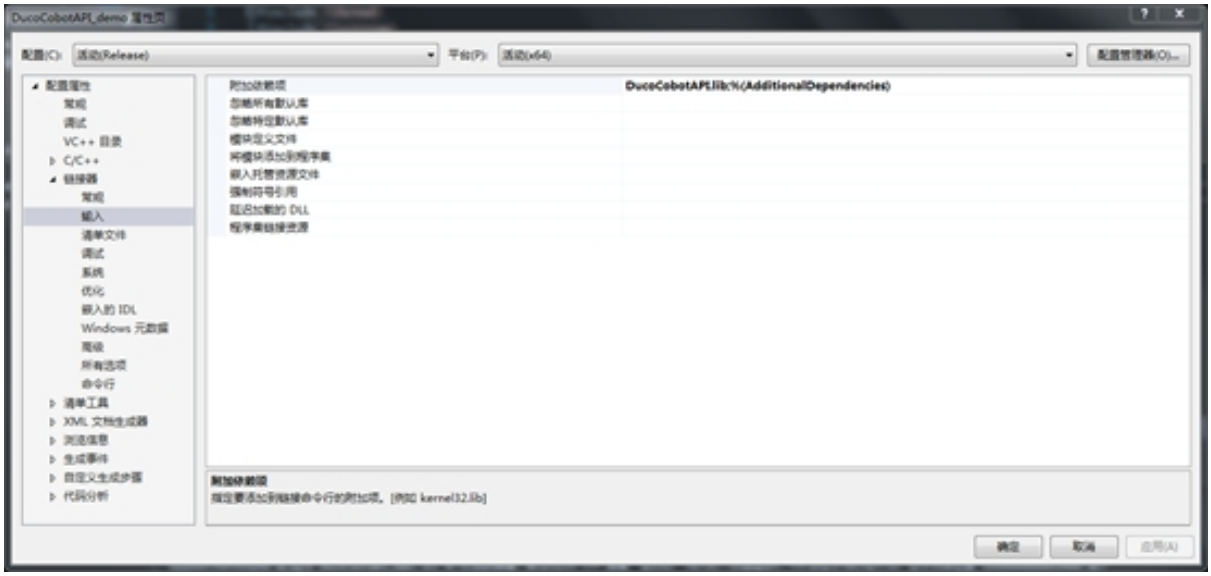
1）“VC++ 目录”中添加引用目录和库目录：



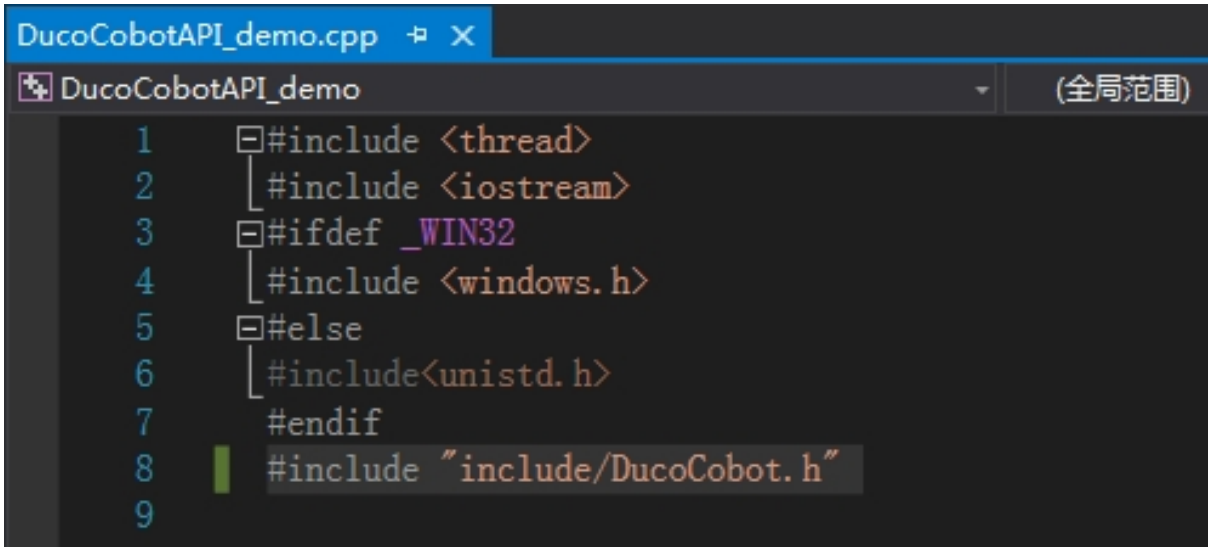
2）“链接器 → 常规”中添加附加库目录：



3）“链接器 → 输入”中添加库名称：



4）在程序中头文件的引用路径：



1.5 常见问题

1.5.1 RPC 客户端多线程崩溃

现象:

同一个 `DucoCobot` 对象在多个线程里调用阻塞函数，运行过程中客户端崩溃。

解决方法:

多线程代码里，务必使用不同的 `DucoCobot` 对象。

CHAPTER 2

ROS 开发说明

2.1 ROS 环境安装

2.1.1 安装 ROS 前准备

4.34KB 将软件与更新中的软件源改成国内的，比如清华

2.1.2 安装过程

第一步：更新软件源

- sudo apt-get update

第二步：安装 Melodic 版本 ROS

- sudo apt-get install ros-melodic-desktop-full
- sudo apt-get install ros-melodic-rqt*

第三步：初始化 rosdep

- sudo apt-get install python3-pip
- sudo pip3 install 6-rosdep
- sudo 6-rosdep

第四步：解决 rosdep update time out

- sudo rosdep init
- rosdep update

第五步：安装 ros install

- sudo apt-get install python-rosinstall

2.1.3 环境配置

第一步：加载 ROS 环境设置文件

- source /opt/ros/melodic/setup.bash

第二步：创建并初始化工作目录

- mkdir -p ~/catkin_ws/src
- cd ~/catkin_ws/src
- catkin_init_workspace

第三步：编译工作目录

- cd ~/ catkin _ws/
- catkin_make

第四步：设置环境变量

- sudo apt install net-tools
- gedit ~/.bashrc

```
# Set ROS melodic
source /opt/ros/melodic/setup.bash
source ~/catkin_ws/devel/setup.bash
source ~/duco_ws/devel/setup.bash

# Set ROS Network
#ifconfig查看你的电脑ip地址
export ROS_HOSTNAME=192.168.12.36
export ROS_MASTER_URI=http://${ROS_HOSTNAME}:11311

# Set ROS alias command 快捷指令
alias cw='cd ~/catkin_ws'
alias cs='cd ~/catkin_ws/src'
alias cm='cd ~/catkin_ws && catkin_make'

export PATH=/usr/lib/ccache:$PATH
```

2.1.4 小海龟测试

第一步：打开三个终端

第一个终端输入

- roscore

第二个终端输入

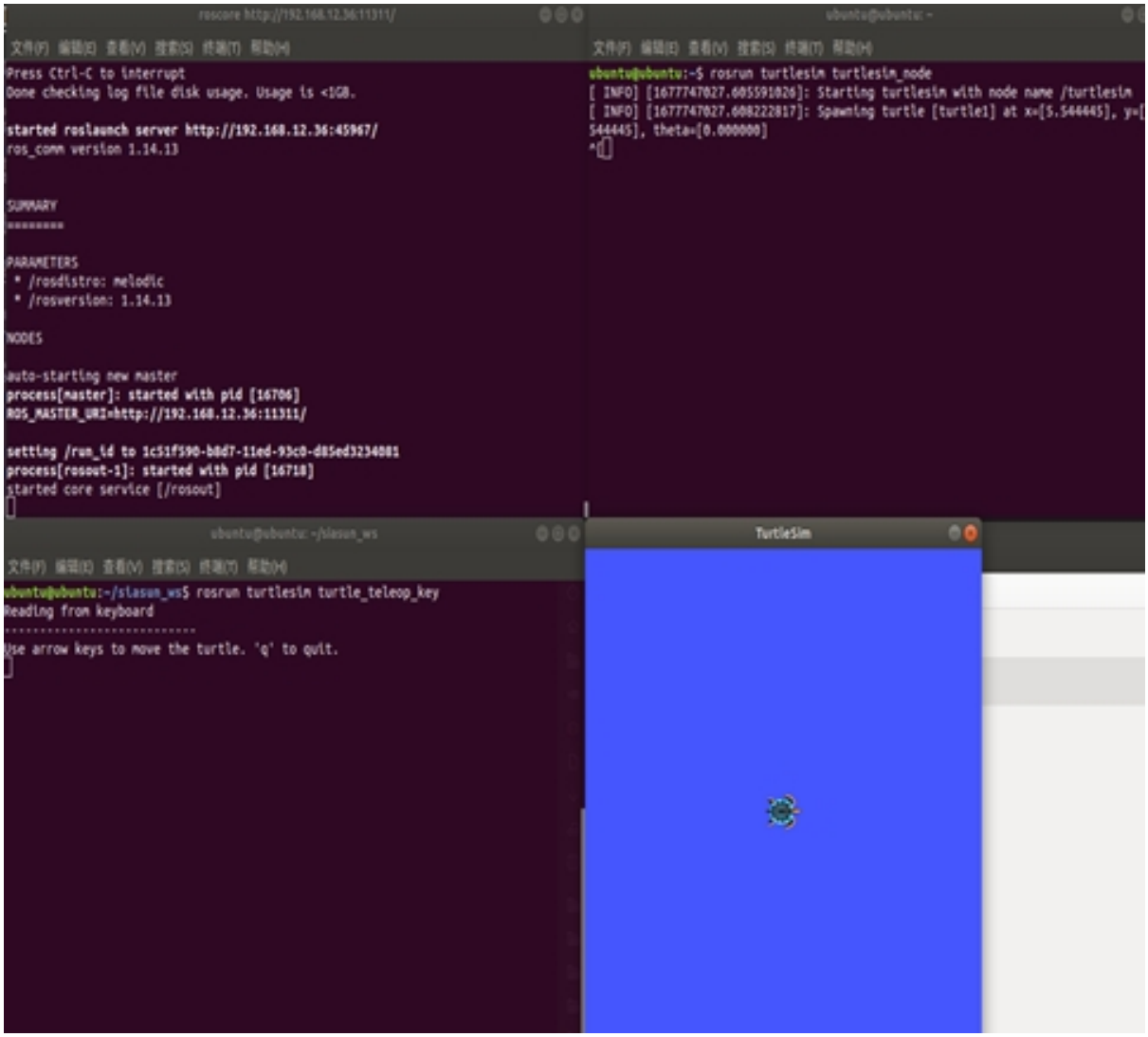
- rosrun turtlesim turtlesim_node

第三个终端输入

- rosrun turtlesim turtle_teleop_key

第二步：

确认是否可以通过键盘上的上下左右键控制小海龟运动，如果可以则完成 ROS 安装，否则出错。如果出错重复上述安装过程。



2.2 MoveIt 环境安装

MoveIt 由 ROS（机器人操作系统）中一系列功能包组成，主要包含运动规划，操作控制，3D 感知，运动学，碰撞检测等等，目前已经广泛应用于工业、商业、研发和其他领域，是目前 ROS 下最受欢迎的功能包之一。MoveIt 还提供了一系列成熟的插件和工具，可以实现机械臂控制的快速配置；并且封装了大量 API，方便用户在 MoveIt! 模块上进行二次开发。MoveIt 支持源码和二进制安装。如无必要建议用户采用二进制安装。详细安装说明可登陆 MoveIt 官方文档查看具体操作。需要注意的是针对不同的 ROS 版本 MoveIt 有相对应的版本，用户安装时需要确认和匹配。下面简单说明下安装步骤：

第一步：打开终端

第二步：在终端中输入

- rosdep update

第三步：在终端中输入

- sudo apt-get update

第四步：在终端中输入

- sudo apt-get dist-upgrade

第五步：在终端中输入

- sudo apt-get install ros-melodic-catkin python-catkin-tools

2.3 ROS 机器人开发包使用说明

2.3.1 系统要求

系统软件：Ubuntu 18.04.1

ROS 版本：Melodic

机器人控制器程序 V2.7 及以上版本

2.3.2 下载安装包并解压

第一步：加载 ROS 环境设置文件

打开终端输入

- source /opt/ros/melodic/setup.bash

第二步：创建并初始化工作目录

- mkdir -p ~/duco_ws/src
- cd ~/duco_ws/src
- catkin_init_workspace

第三步：编译工作目录

- cd ~/duco_ws/

- catkin_make

第四步：下载 ROS 二次开发包，并解压后

将所得的源码复制到指定的/duco_ws/src 下。

具体代码目录结构：

duco_ws

--src

--CMakeLists.txt

--duco_controller

--duco_demo

--duco_driver

--duco_gcr5_moveit_config

--duco_gcr5_moveit_config

--duco_gcr5_moveit_config

--duco_gcr5_moveit_config

--duco_gcr5_moveit_config

--duco_gcr5_moveit_config

--duco_msgs

--duco_support

第五步：编译 ROS 二次开发包

打开终端依次输入下列指令

- cd ~/duco_ws
- source ./devel/setup.bash
- catkin_make

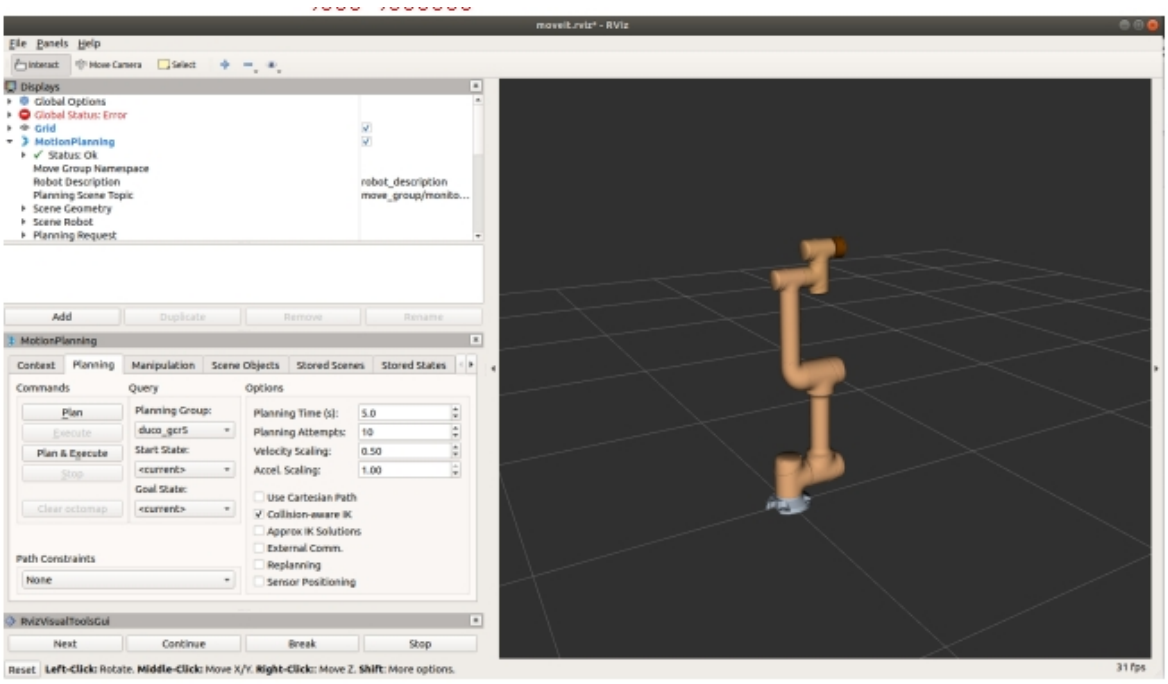
第六步：启动脚本控制机器人

确认当前 DucoCore 控制器是否已经启动

打开终端输入下列指令：

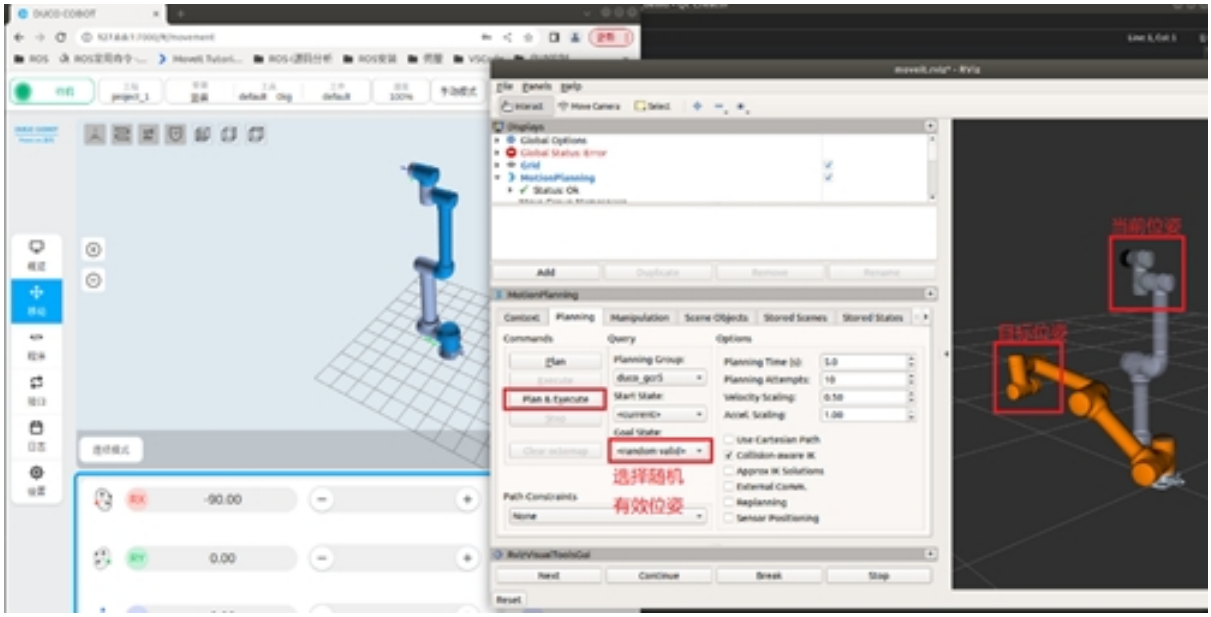
- cd ~/duco_ws
- source ./devel/setup.bash
- roslaunch duco_gcr5_moveit_config moveit_planning_execution.launch robot_ip:=< 控制器 IP>

等待一会即可启动 rviz+moveit 界面，如下图所示。



可通过设置 Goal State 为随机有效位姿，并确认该位姿处于安全状态下，点击 Plan & Execute 按钮。

即可规划路径，并控制机械臂进行相应的运动。

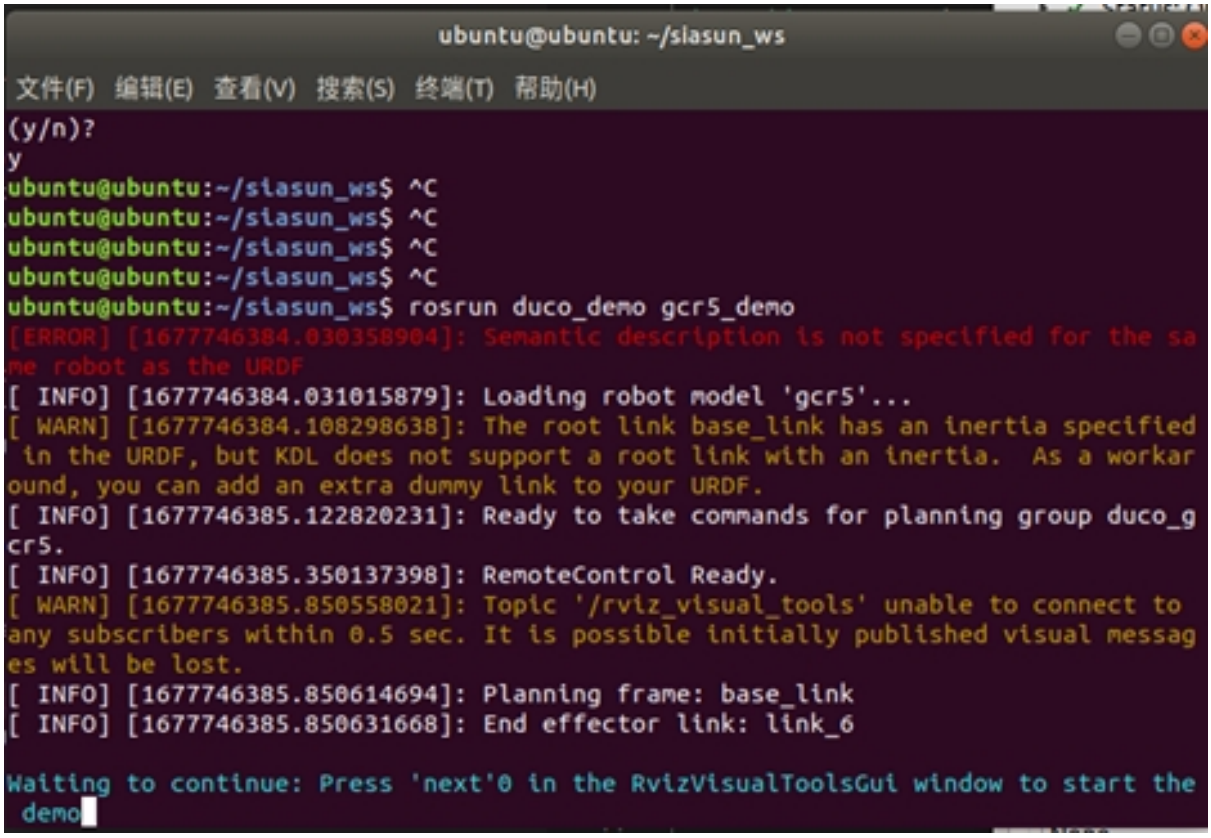


2.3.3 duco_demo 使用

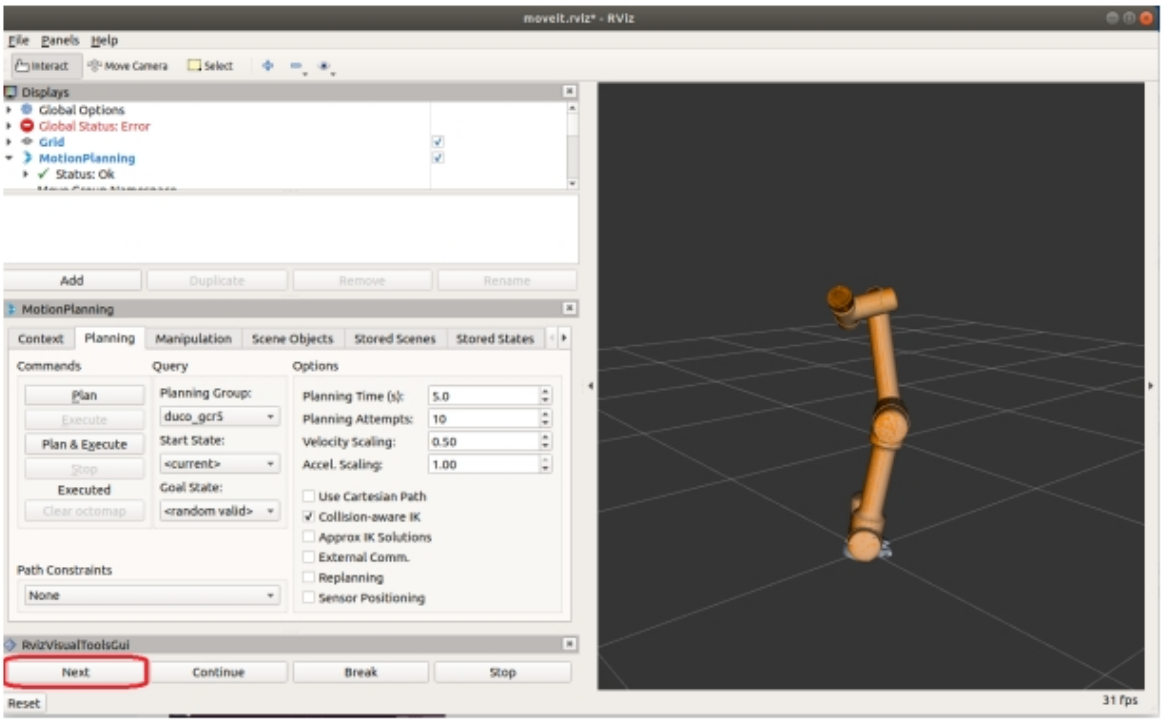
duco_demo 里面包含 gcr5_demo 节点，该节点主要是通过 MoveGroup 提供的接口实现对机器人控制。对于有一定 C++ 编程基础且对机器人比较熟悉的用户可以修改该示例程序从而实现自己想要的功能。

第一步：打开终端输入

- rosrun duco_demo gcr5_demo



第二步：点击 RVIZ 窗口里的 next 实现对机器人控制。


moveit.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Error
Grid
MotionPlanning
Status: Ok
Add
Duplicate
Remove
Rename
MotionPlanning
Context
Planning
Manipulation
Scene Objects
Stored Scenes
Stored States
Commands
Plan
Execute
Plan & Execute
Stop
Executed
Clear octomap
Query
Planning Group:
duco_gcr5
Start State:
<current>
Goal State:
<random valid>
Options
Planning Time (s):
5.0
Planning Attempts:
10
Velocity Scaling:
0.50
Accel. Scaling:
1.00
Use Cartesian Path
Collision-aware IK
Approx IK Solutions
External Comm.
Replanning
Sensor Positioning
Path Constraints
None
RvizVisualToolsGui
Next
Continue
Break
Stop
Reset
31 fps

CHAPTER 3

ROS2 开发说明

3.1 基本环境安装

推荐在 Ubuntu 22.04.4 LTS 上搭配环境，以及推荐使用 ros2-humble。

（注意：需要安装官方 ros2_control 包，再正式编译 duco 功能包）

3.1.1 ros2 基础安装

下面对 ROS2(humble) 的过程详细阐述：

1. 打开 shell 窗口

```
locale  # check for UTF-8

sudo apt update && sudo apt install locales
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8

locale  # verify settings
```

2. 设置源

```
sudo apt install software-properties-common
sudo add-apt-repository universe
sudo apt update && sudo apt install curl -y
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/share/keyrings/ros-archive-keyring.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-keyring.gpg] http://packages.ros.org/ros2/ubuntu $(. /etc/os-release && echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

3. 安装 ROS2

```
sudo apt update
sudo apt upgrade
sudo apt install ros-humble-desktop
```

4. 最后安装 dev 工具

```
sudo apt install ros-dev-tools
```

下面对 ros2_control 的安装过程详细阐述:

1. 首先 source ROS2 的资源

```
source /opt/ros/humble/setup.bash
```

2. 创建 ros2_control 工作空间，下载资源

```
mkdir -p ~/ros2_control_ws/
cd ~/ros2_control_ws/
```

创建好之后，将 duco 的 src 文件放入此目录中。

然后开始下载资源:

```
wget https://raw.githubusercontent.com/ros-controls/ros2_control_ci/master/
↪ros_controls.humble.repos
vcs import src < ros_controls.humble.repos
```

如果网络不行，需要切换软件源再重新下载资源:

```
# 设置软件源
sudo sh -c 'echo "deb [arch=$(dpkg --print-architecture)] http://mirrors.
↪tuna.tsinghua.edu.cn/ros2/ubuntu/ $(lsb_release -cs) main" > /etc/apt/
↪sources.list.d/ros2.list'

# 添加密钥
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys␣
↪F42ED6FBAB17C654
sudo apt update
```

3.1.2 rosdep

安装依赖包:

```
sudo rosdep init
rosdep update --rosdistro=$ROS_DISTRO
sudo apt-get update
rosdep install --from-paths src --ignore-src -r -y
```

编译 ros2_control:

```
. /opt/ros/humble/setup.sh
```

3.1.3 补充安装 colcon

如若 colcon 缺失，补充安装 colcon。没有缺失 colcon 文件，则可接着去安装下面的 Moveit。

```
# 安装必要组件
sudo apt update
sudo apt install python3-colcon-common-extensions

# 确保 pip 版安装（系统级安装）
pip3 install --upgrade colcon-common-extensions
```

3.1.4 Moveit

安装完整 MoveIt 套件 (针对 Humble)

```
sudo apt install ros-humble-moveit ros-humble-moveit-ros-planning-interface
```

验证安装:

```
ros2 pkg prefix moveit_ros_planning_interface
```

3.1.5 安装环境配置包:

安装 test-msgs 包:

```
sudo apt install ros-humble-test-msgs
```

安装 control-toolbox:

```
sudo apt install ros-humble-filters ros-humble-control-toolbox

colcon build --symlink-install --packages-select control_toolbox
source install/setup.bash

colcon build --symlink-install --packages-up-to hardware_interface
```

安装 ackermann-msgs 包:

```
# 官方 ROS 2 ackermann-msgs 包
sudo apt install ros-humble-ackermann-msgs

# 验证安装
ros2 pkg list | grep ackermann_msgs
```

完成后，重置环境:

```
# 重置环境变量
unset AMENT_PREFIX_PATH
unset CMAKE_PREFIX_PATH

# 设置基础 ROS 环境
source /opt/ros/humble/setup.bash
```

(续下页)

（接上页）

```
# 手动指定ackermann_msgs路径
export ackermann_msgs_DIR=~/ackermann_ws/install/ackermann_msgs/share/
↪ackermann_msgs/cmake
```

进行最终编译：

```
cd ~/ros2_control_ws

# 1. 清除所有构建文件
rm -rf build install log

# 2. 首先单独编译消息包
colcon build --symlink-install --packages-select duco_msg

# 3. 加载系统ROS和刚刚编译的消息包环境
source /opt/ros/humble/setup.bash
source install/setup.bash  # 关键：加载刚编译的duco_msg

# 4. 编译其他所有包（允许覆盖必要的消息）
colcon build --symlink-install --allow-overriding ackermann_msgs,control_
↪msgs
```

3.2 功能包编译与基础配置

3.2.1 克隆 DUCO 资源包

克隆 DUCO 的 src 资源包到本地，接下来 cd 到目标目录下，其中主要文件包括 duco_msg，duco_arm，duco_support 和 duco_ros_driver 等多可机器人功能包。

我们有两种方式对机械臂进行配置，第二种在下一章节。现在我们介绍下第一种配置方式。

3.2.2 编译 duco 功能包

首先，需要进入到 duco_ros2_driver 目录，在此目录下运行如下命令：

第一步：分步编译核心包（msg/support/arms/driver）

```
colcon build --packages-select duco_msg
colcon build --packages-select duco_support
colcon build --packages-select duco_arms
colcon build --packages-select duco_ros_driver
colcon build
```

```
oem@robot:~/git/ros2_demo$ colcon build --packages-select duco_msg
Starting >>> duco_msg
Finished <<< duco_msg [6.38s]

Summary: 1 package finished [6.70s]
```

```
oem@robot:~/git/ros2_demo$ colcon build --packages-select duco_support
Starting >>> duco_support
Finished <<< duco_support [0.53s]

Summary: 1 package finished [0.78s]
```

```
oem@robot:~/git/ros2_demo$ colcon build --packages-select duco_arms
Starting >>> duco_arms
Finished <<< duco_arms [0.44s]

Summary: 1 package finished [0.69s]
```

```
oem@robot:~/git/ros2_demo$ colcon build --packages-select duco_ros_driver
Starting >>> duco_ros_driver
--- stderr: duco_ros_driver
```

```
oem@robot:~/git/ros2_demo$ colcon build
Starting >>> duco_support
Starting >>> duco_msg
Starting >>> robot_control
Finished <<< duco_support [0.08s]
Starting >>> duco_arms
Starting >>> double_arm_moveit_config
Starting >>> duco_gcr10_1300_moveit_config
Starting >>> duco_gcr10_2000_moveit_config
Starting >>> duco_gcr12_1300_moveit_config
Starting >>> duco_gcr14_1400_moveit_config
Starting >>> duco_gcr16_2000_moveit_config
Starting >>> duco_gcr16_960_moveit_config
Starting >>> duco_gcr20_1100_moveit_config
```

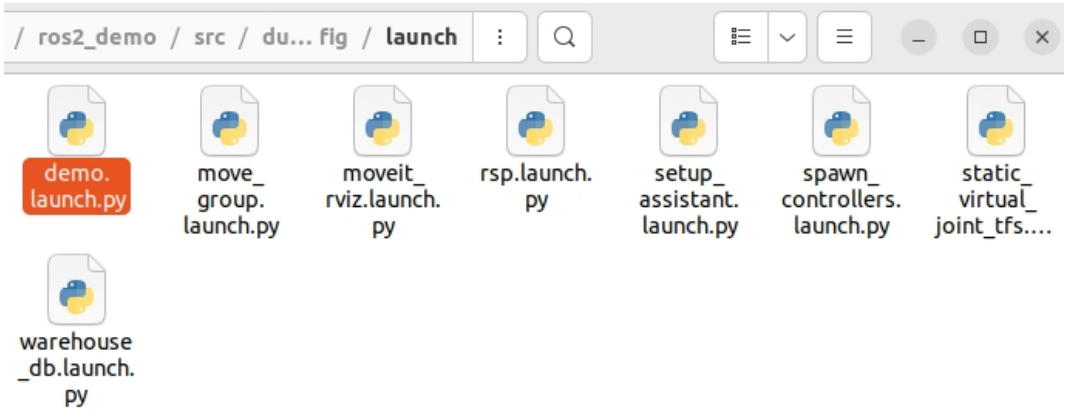
第二步：资源加载

```
source install/setup.bash
colcon build
```

```
oem@robot:~/git/ros2_demo$ source install/setup.bash
oem@robot:~/git/ros2_demo$ colcon build
Starting >>> duco_support
Starting >>> duco_msg
Starting >>> robot_control
Finished <<< duco_support [0.08s]
Starting >>> duco_arms
Starting >>> double_arm_moveit_config
Starting >>> duco_gcr10_1300_moveit_config
```

第三步：修改接口地址

进入 src 文件夹/使用的 gcr 文件夹/launch 文件夹/demo.launch.py



找到：

```
ld.add_action(
    DeclareLaunchArgument(
    'server_host_1',
    default_value='',
    )
)
```

修改其中的 default_value 为使用的域名地址。

```
ld.add_action(
    DeclareLaunchArgument(
    'server_host_1',
    default_value='192.168.120.163',
    )
)
```

第四步：启动所需 moveit 配置包的 launch 文件：

这里以 duco_gcr5_910_moveit_config 为例。

可更换为其他文件，例如 duco_gcr3_618_moveit_config。

```
source install/setup.bash
ros2 launch duco_gcr5_910_moveit_config demo.launch.py
```

```
oem@robot:~/git/ros2_demo$ ros2 launch duco_gcr5_910_moveit_config demo.launch.py
[INFO] [launch]: All log files can be found below /home/oem/.ros/log/2025-05-21-09-34-05-408574-robot-48990
[INFO] [launch]: Default logging verbosity is set to INFO
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
[INFO] [robot_state_publisher-1]: process started with pid [48991]
[INFO] [move_group-2]: process started with pid [48993]
[INFO] [rviz2-3]: process started with pid [48995]
[INFO] [DucoDriver-4]: process started with pid [48997]
[INFO] [DucoRobotStatus-5]: process started with pid [48999]
[INFO] [DucoTrajectoryAction-6]: process started with pid [49001]
[INFO] [DucoRobotControl-7]: process started with pid [49003]
[DucoRobotControl-7] [INFO] [1747791245.936443036] [DucoRobotControl]: Duco Robot Control Init Done
[DucoDriver-4] [INFO] [1747791245.937161274] [DucoDriver]: DucoDriver::moveItPosCallback msg is empty 192.168.120.163
[DucoRobotStatus-5] [INFO] [1747791245.938744336] [DucoRobotStatus]: Duco Robot Status Init Done
```

3.2.3 检查链接状况

第五步：检查连接

首先新开一个位于同目录下的终端，初始化并查看节点：

```
source install/setup.bash
ros2 service list
```

```
oem@robot:~/git/ros2_demo$ source install/setup.bash
oem@robot:~/git/ros2_demo$ ros2 service list
/DucoDriver/describe_parameters
/DucoDriver/get_parameter_types
/DucoDriver/get_parameters
/DucoDriver/list_parameters
/DucoDriver/set_parameters
/DucoDriver/set_parameters_atomically
/DucoRobotControl/describe_parameters
/DucoRobotControl/get_parameter_types
/DucoRobotControl/get_parameters
/DucoRobotControl/list_parameters
```

第六步：

检查是否存在以下节点：

```
/duco_robot/robot_control
/duco_robot/robot_io_control
/duco_robot/robot_move
```

3.3 Moveit_setup_assistant 环境配置

3.3.1 moveit_setup_assistant 配置过程

以下内容将通过 moveit_setup_assistant 来配置自定义机器人 moveit2_config 的配置包。

第一步：

安装上一章节所说的，对功能包进行编译：

```
colcon build --packages-select duco_msg
colcon build --packages-select duco_support
colcon build --packages-select duco_arms
colcon build --packages-select duco_ros_driver
colcon build
```

创建工作空间以及功能包：

```
mkdir -p test_duco_ws/src
cd test_duco_ws/src
mkdir duco_test_robot_moveit_config
```

（续下页）

（接上页）

```
cd ..
cd ..
```

编译功能包后进行 source:

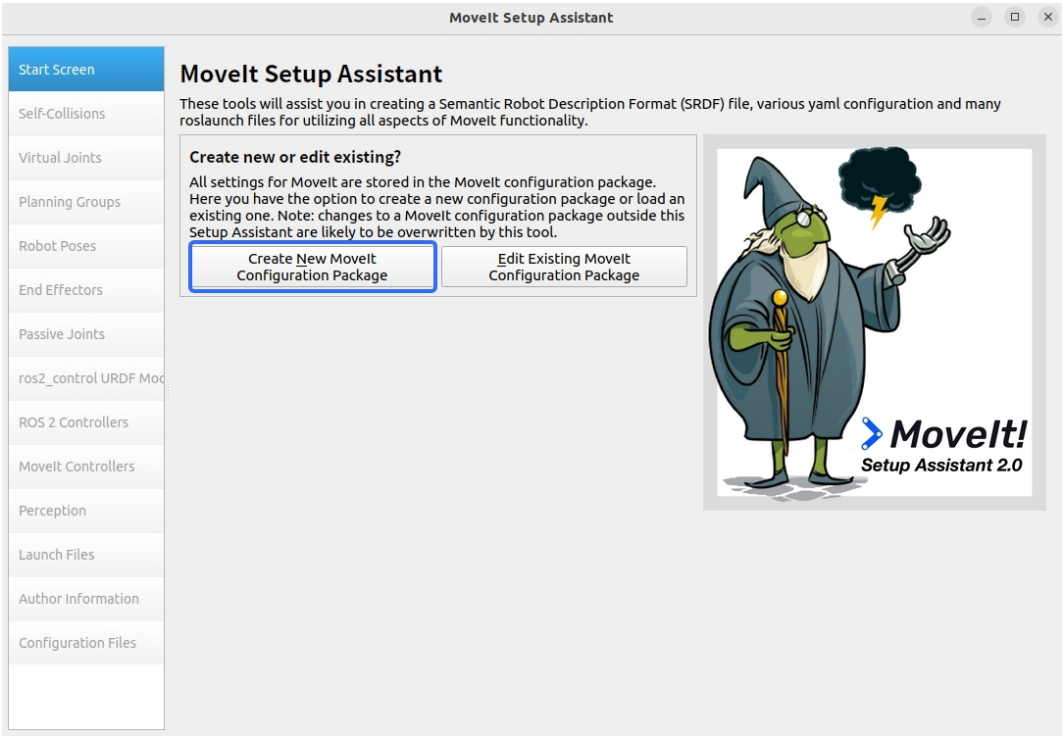
```
colcon build
source install/setup.bash
```

来到 test_duco_ws，启动 moveit_setup_assistant 对机器人进行配置

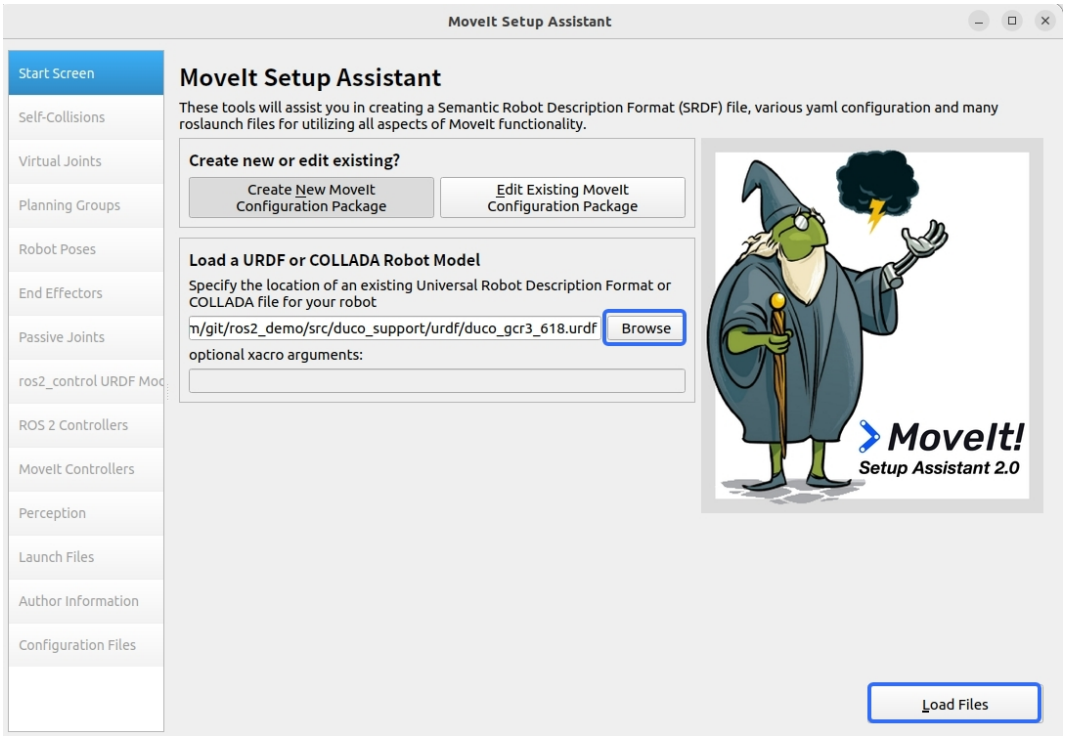
```
cd test_duco_ws
ros2 launch moveit_setup_assistant setup_assistant.launch.py
```

第二步：进行配置

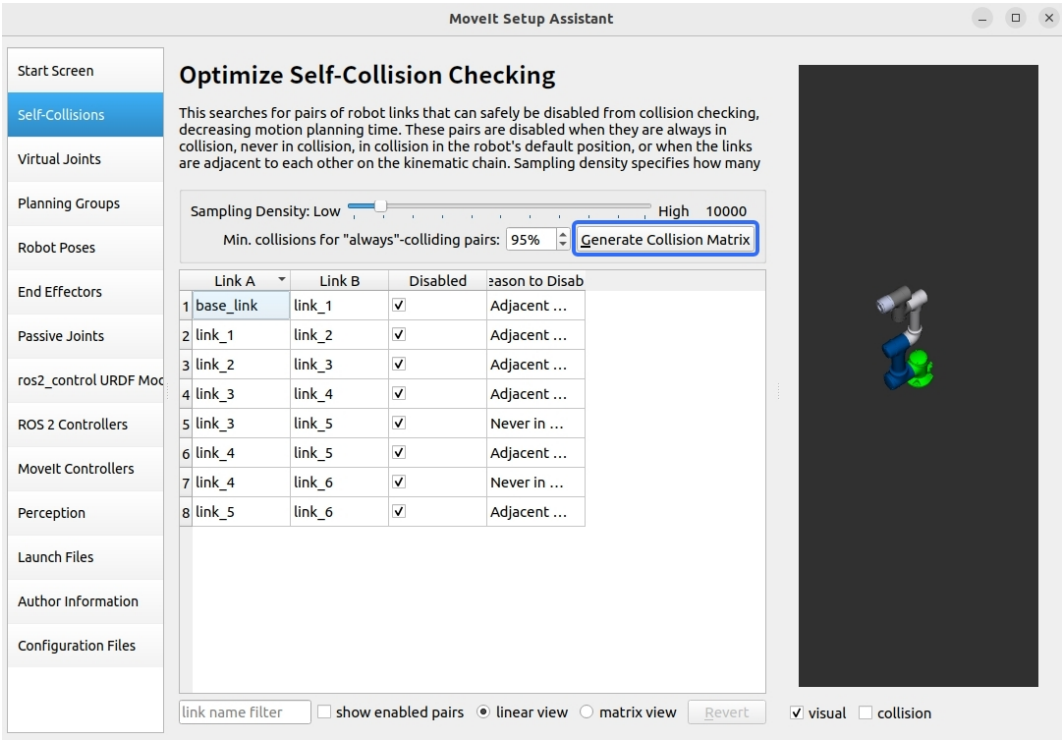
当启动配置界面后，在配置界面中点击“Create New Moveit Configuration Package”来创建新 Moveit 配置功能包。



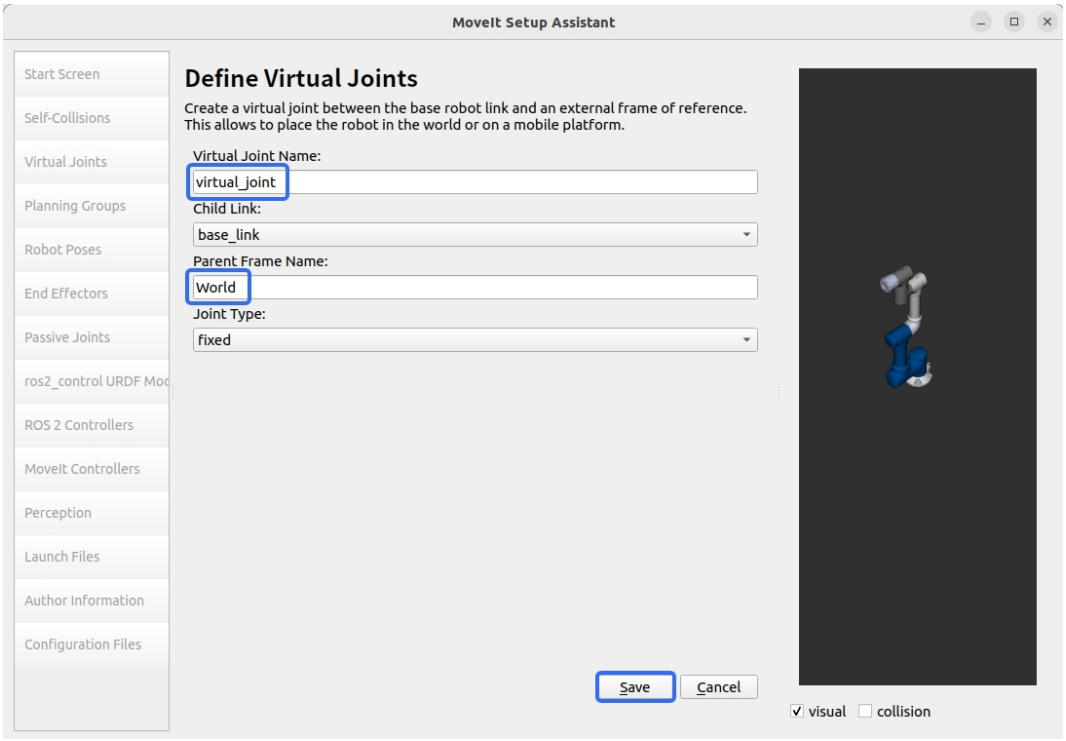
然后点击 Browse，从之前导入的目录 src/duco_support/urdf 中，选中所需要机械臂 gcr 的.urdf 描述文件。选好后点击 Load Files 进行加载。(可以在界面右侧看到加载出来的机械臂)



在左侧选中 Self-Collisions 界面，点击 Generate Collision Matrix 既可。点击后将自动生成关节碰撞矩阵，避免计算两接触连杆和接触不到的连杆间的碰撞。

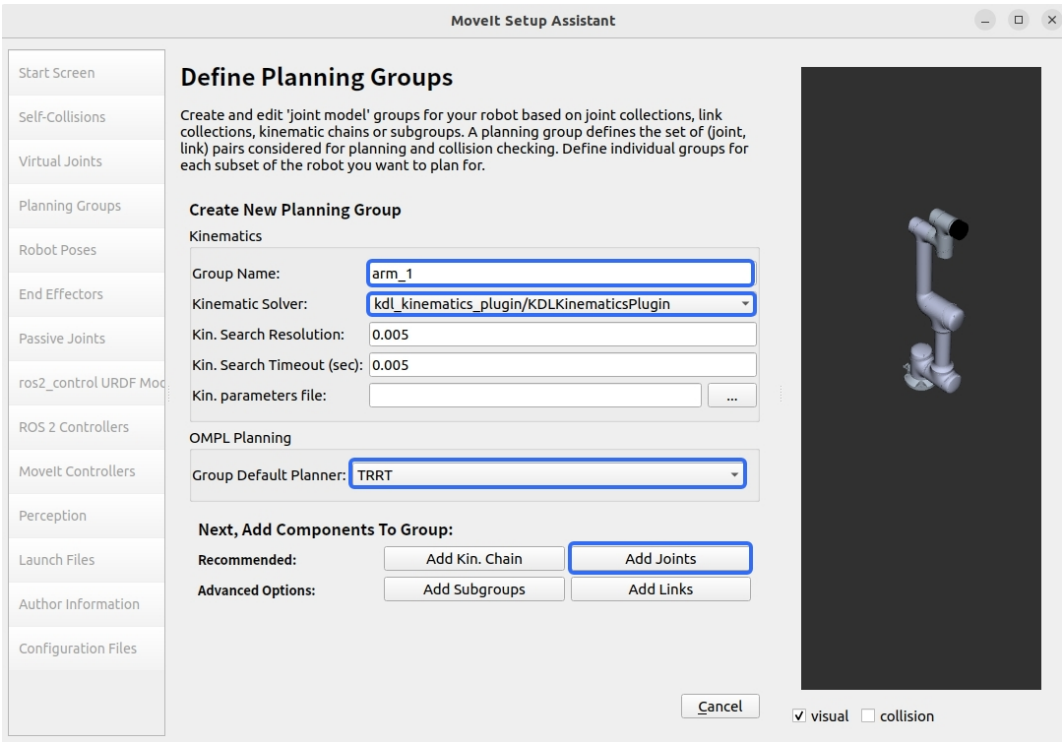


接下来设置虚拟关节 Virtual Joints，这是用于机器人安装在移动平台时的虚拟轴配置。当机器人需要随移动平台协同运动时，需配置虚拟关节参数，包括名称和关节类型等。通过虚拟关节的同步运动，可实现机器人本体随移动平台的联动。在本演示中，机器人被直接放置在全局坐标系 world 下，虚拟关节命名为 virtual_joint。

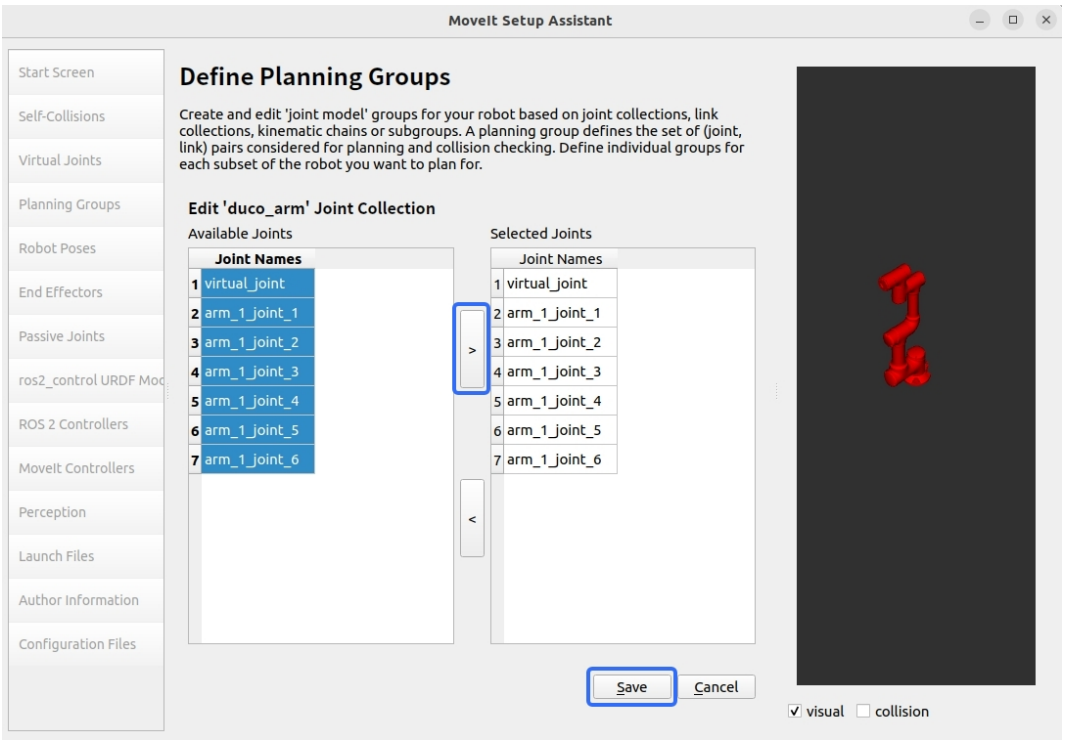


规划组（Planning Groups）用于将机器人系统中需要协同进行运动学计算的关节划分为逻辑单元。每个规划组内的关节将作为一个整体进行正或逆运动学计算。本演示仅配置机器人本体的关节规划组（arm 组），具体步骤如下：

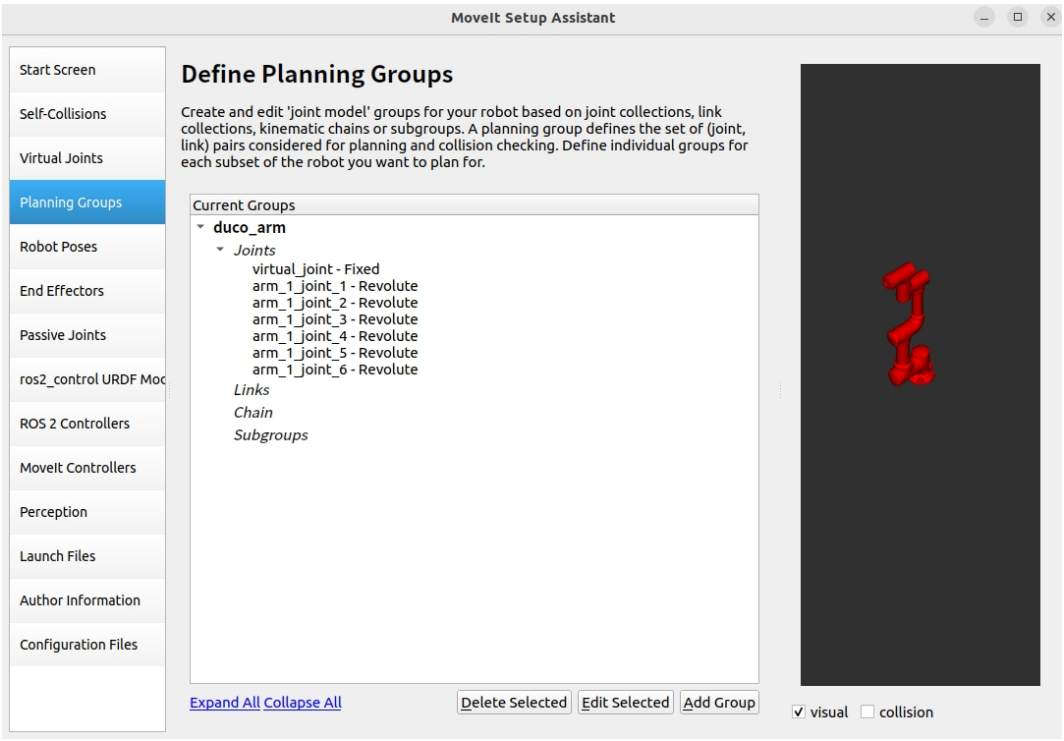
1. 点击 Add Groups
2. 填写 Group name（支持 arm_1 作为名字）
3. 选择运动学求解器 kinetic Solver 为：kdl_kinematics_plugin/KDLKinematicsPlugin
4. 设置默认规划器 Group Default Planner 为：TRRT
5. 通过 Add Joints 功能添加相关关节



选中所有关节，通过">" 进行添加后点击 save 保存。

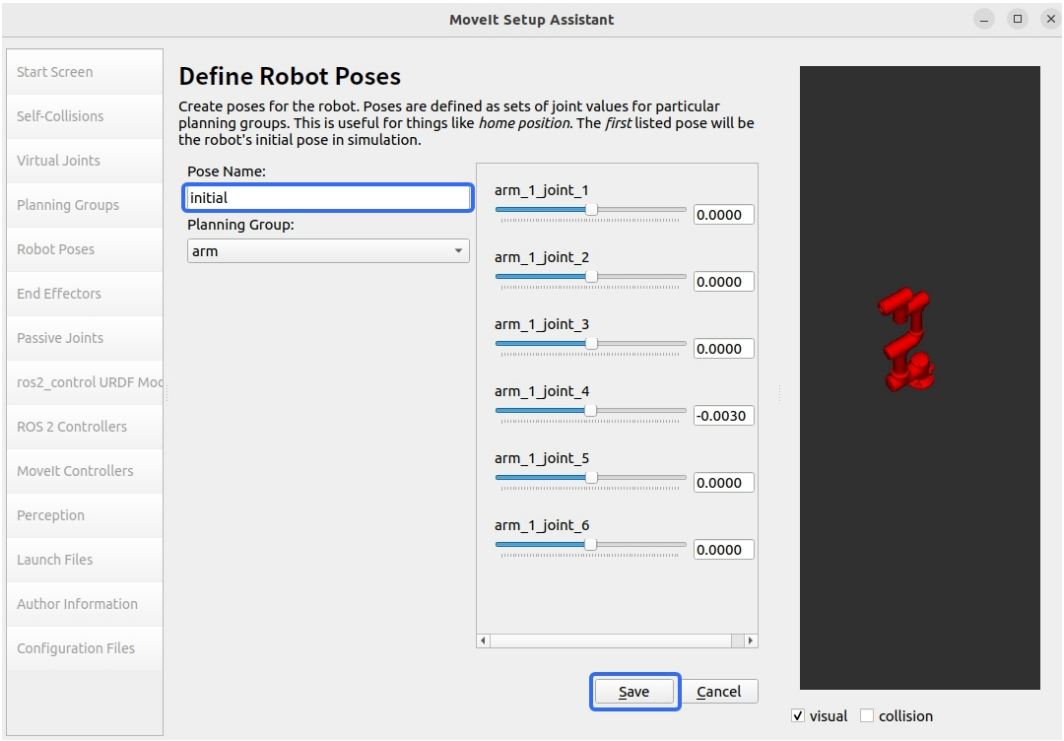


定义好后，会如下图所示：

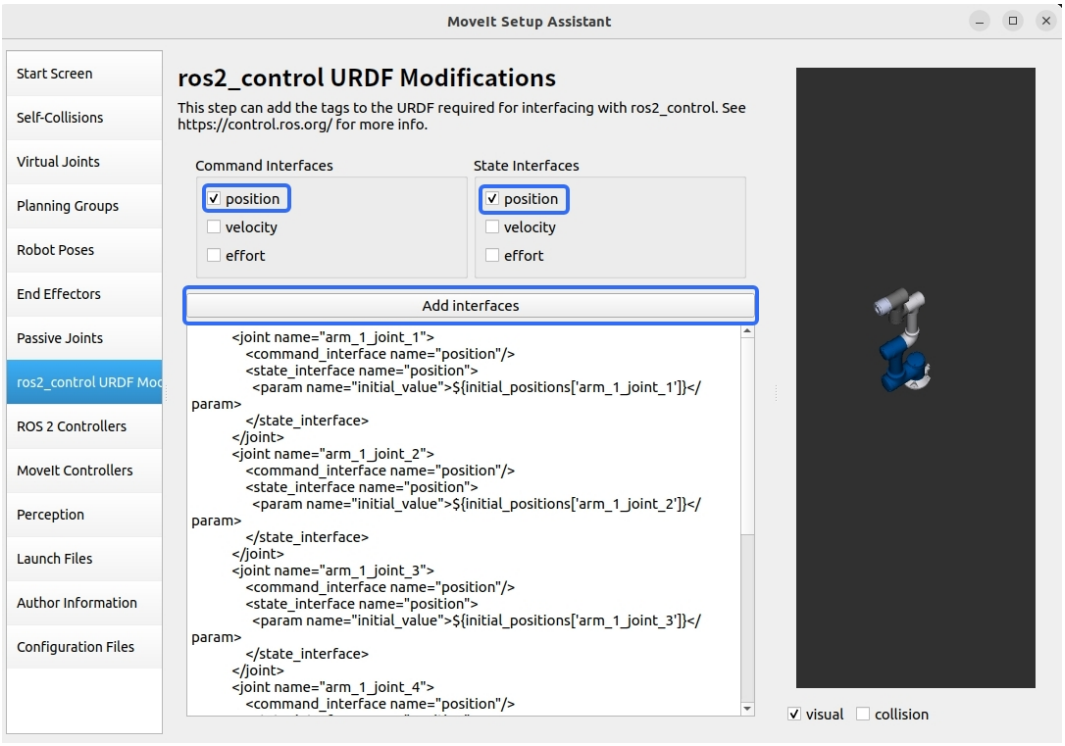


进入左侧 Robot Poses 界面后，我们可以通过为机械臂的每个规划组定义写预设位姿，姿态没有限制，可以随需求调整。

首先点击 Add Pose，然后在 Pose Name 中输入名字（在此以 initial 作为名字），最后点击 save 保存。



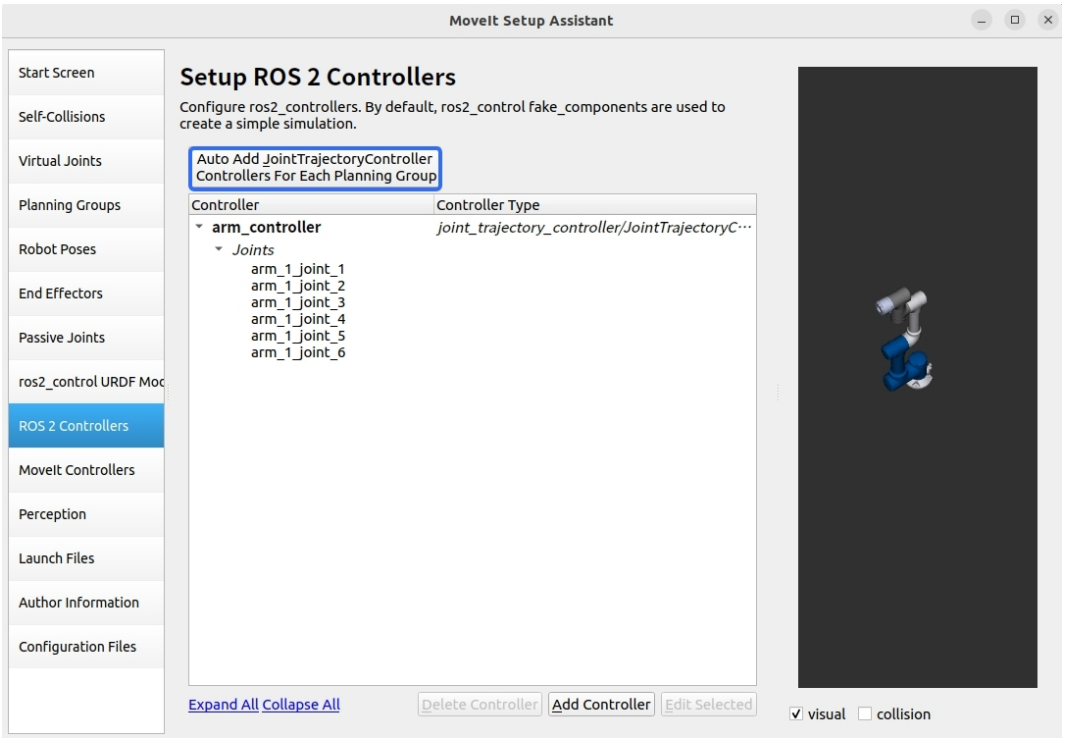
接下来配置 ros2_control URDF Modifications。我们这次为 command Interfaces 和 State Interfaces 勾上 position。然后点击 Add Interfaces 添加配置。



ROS 2 Controllers 的核心功能是自动生成 ros2_controllers.yaml 配置文件。该文件包含控制器类型选择等关键参数配置。

为每个规划组 Planning Group 指定对应的控制器

我 们 通 过 点 击"Auto Add JointTrajectoryController Controllers For Each Planning Group" 功能自动生成轨迹控制器配置即可。



MoveIt Controllers 的核心功能是生成 moveit_controllers 配置文件，该文件主要定义以下关键参数：

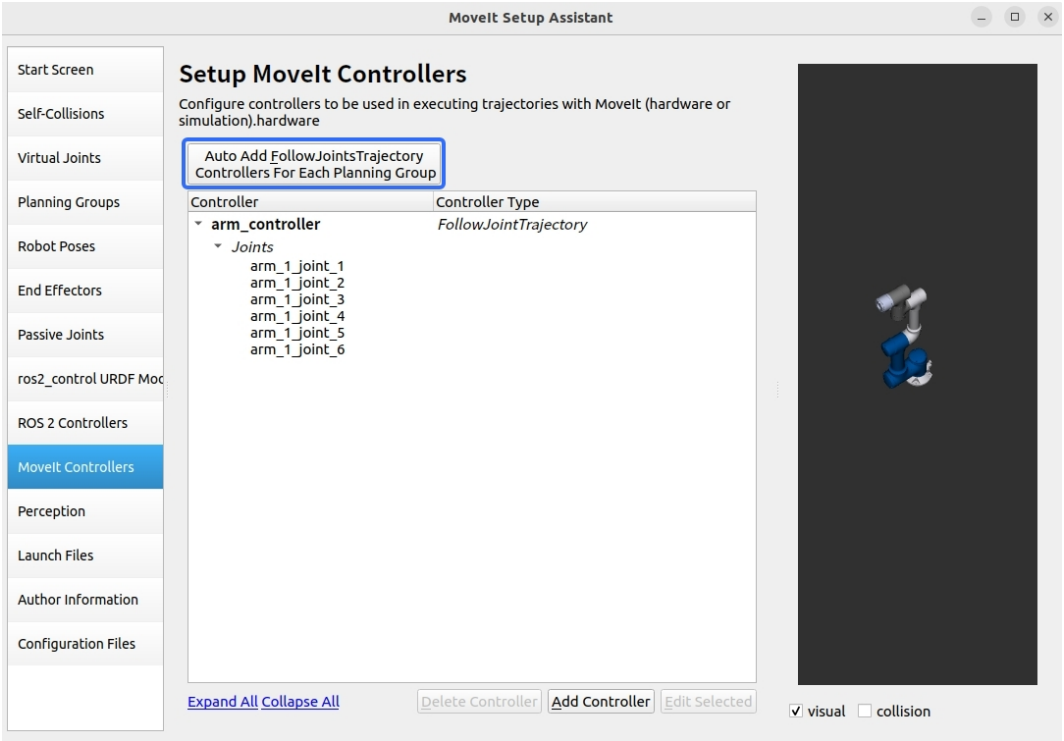
控制器命名（controller name）

控制器类型（controller type）

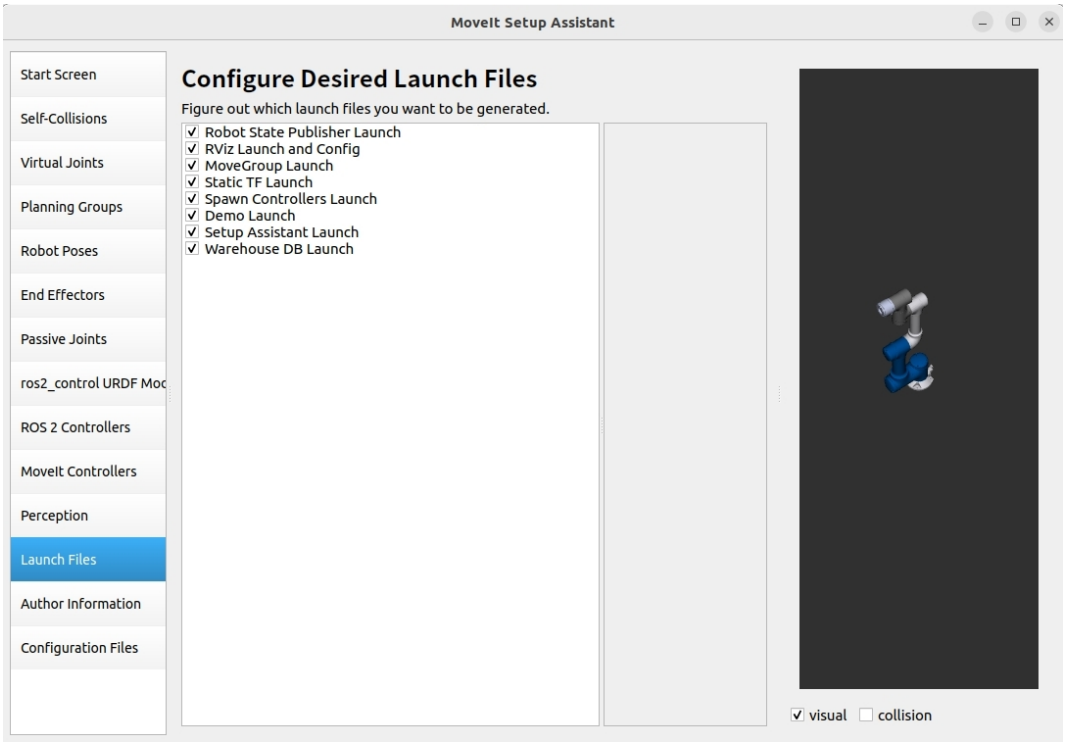
重要配置注意事项：

命名一致性要求：moveit_controllers 中配置的控制器名称必须与 ros2_controllers 中的控制器名称完全一致，这是系统正常运行的必要条件。

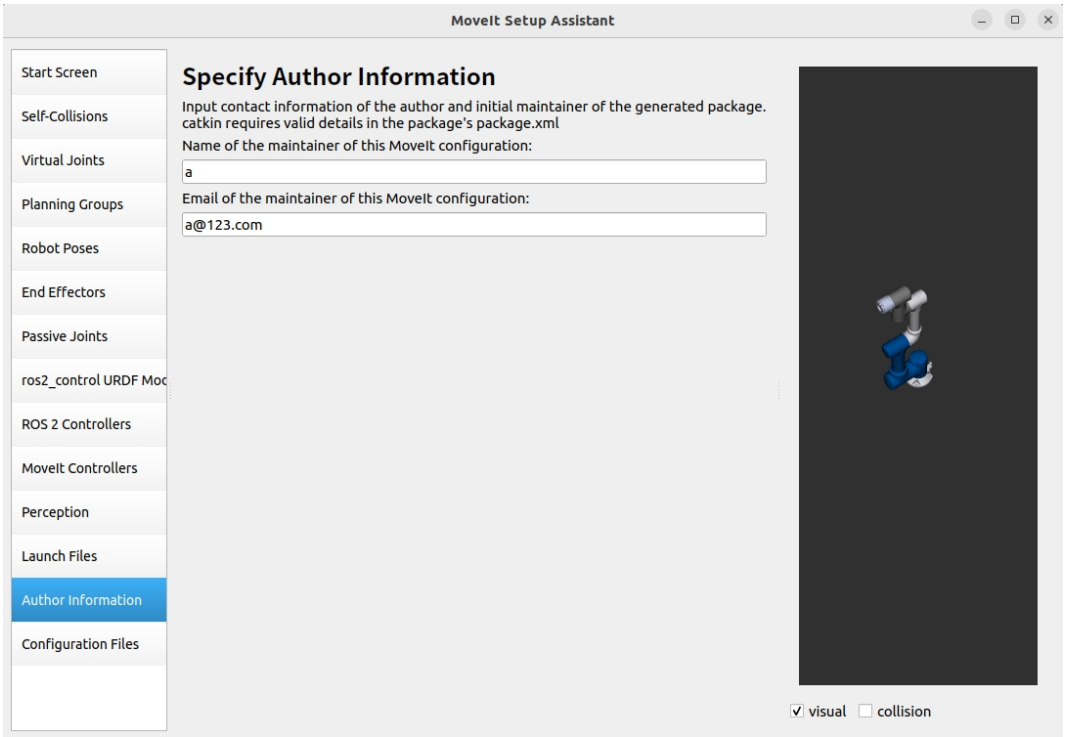
自动类型映射机制：当名称匹配时，系统会自动建立控制器类型映射关系。



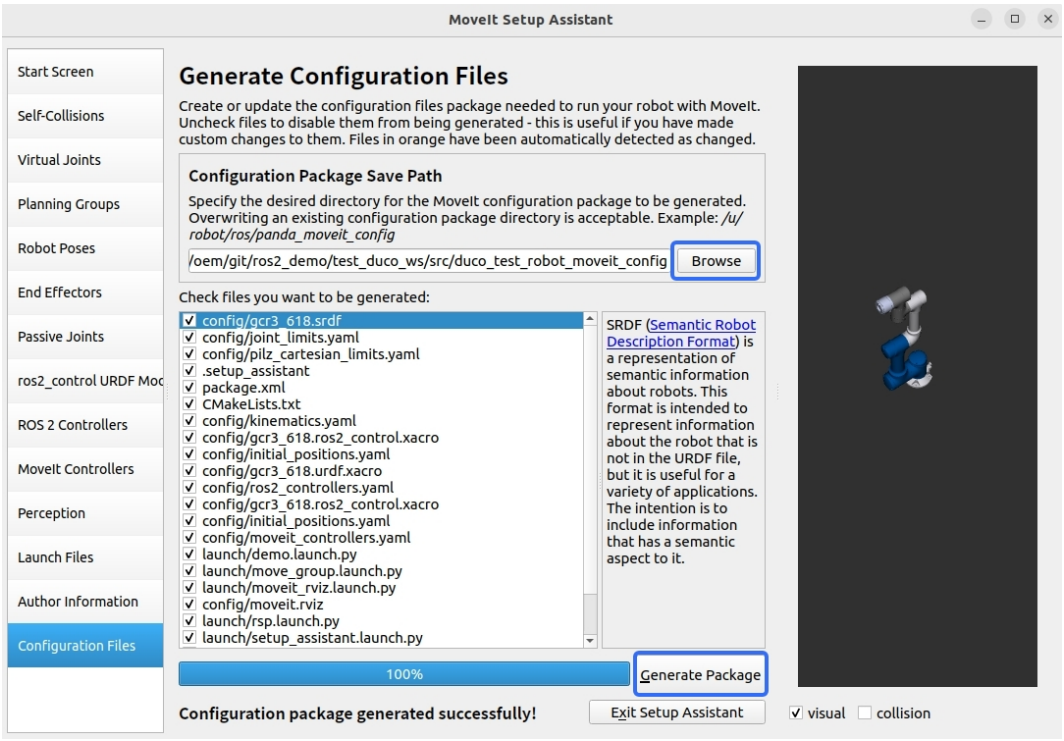
进入到 Launch Files 界面中，我们使用默认配置即可。



对于作者信息来说，我们照常填写即可 （注意需要填写有效邮箱）。



最后来到了 `Configuration Files` 的界面，我们点击 `Browse` 进行生成位置的选择。在这次演示中选择了前面我们创建的 `duco_test_robot_moveit_config` 文件夹来存放配置文件。



我们配置完后可以对我们的新功能包进行编译，用我们自己定义的 moveit2 配置包进行替换，增强兼容性。

3.3.2 运行过程

首先，我们来到 test_duco_ws 目录下的终端，输入以下指令：.. code-block:

```
:linenos:

colcon build --packages-select duco_test_robot_moveit_config
source install/setup.bash
```

然后，回到上一层的目录，运行 source:

```
cd ..
source install/setup.bash
```

然后，再回到我们刚刚的目录，运行 launch 文件，即可打开配置好的 RVIZ。

```
cd test_duco_ws
ros2 launch duco_test_robot_moveit_config demo.launch.py
```

```
oem@robot:~/Desktop/ros2_demo_desk$ source install/setup.bash
oem@robot:~/Desktop/ros2_demo_desk$ cd test_duco_ws
oem@robot:~/Desktop/ros2_demo_desk/test_duco_ws$ ros2 launch duco_test_robot_mov
eit_config demo.launch.py
[INFO] [launch]: All log files can be found below /home/oem/.ros/log/2025-05-23-
15-16-20-921344-robot-1908903
[INFO] [launch]: Default logging verbosity is set to INFO
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
```

3.4 配置机器人连接

通过“Moveit_setup_assistant 环境配置”步骤生成的配置文件为仅为机器人仿真测试，为与真实机械臂进行通信连接，可通过以下步骤实现。

当前提供的开发包 duco_ros2_driver 中，在 duco_support 文件夹下中已包含当前所有臂型 urdf，并提供相应 moveit_config 配置文件，型号包含：

--duco_gcr3_618_moveit_config

--duco_gcr5_910_moveit_config

--duco_gcr7_910_moveit_config

--duco_gcr10_1300_moveit_config

--duco_gcr10_2000_moveit_config

--duco_gcr12_1300_moveit_config

--duco_gcr14_1400_moveit_config

--duco_gcr16_960_moveit_config

--duco_gcr16_2000_moveit_config

--duco_gcr20_1100_moveit_config

--duco_gcr20_1400_moveit_config

--duco_gcr25_1800_moveit_config

--duco_gcr30_1100_moveit_config

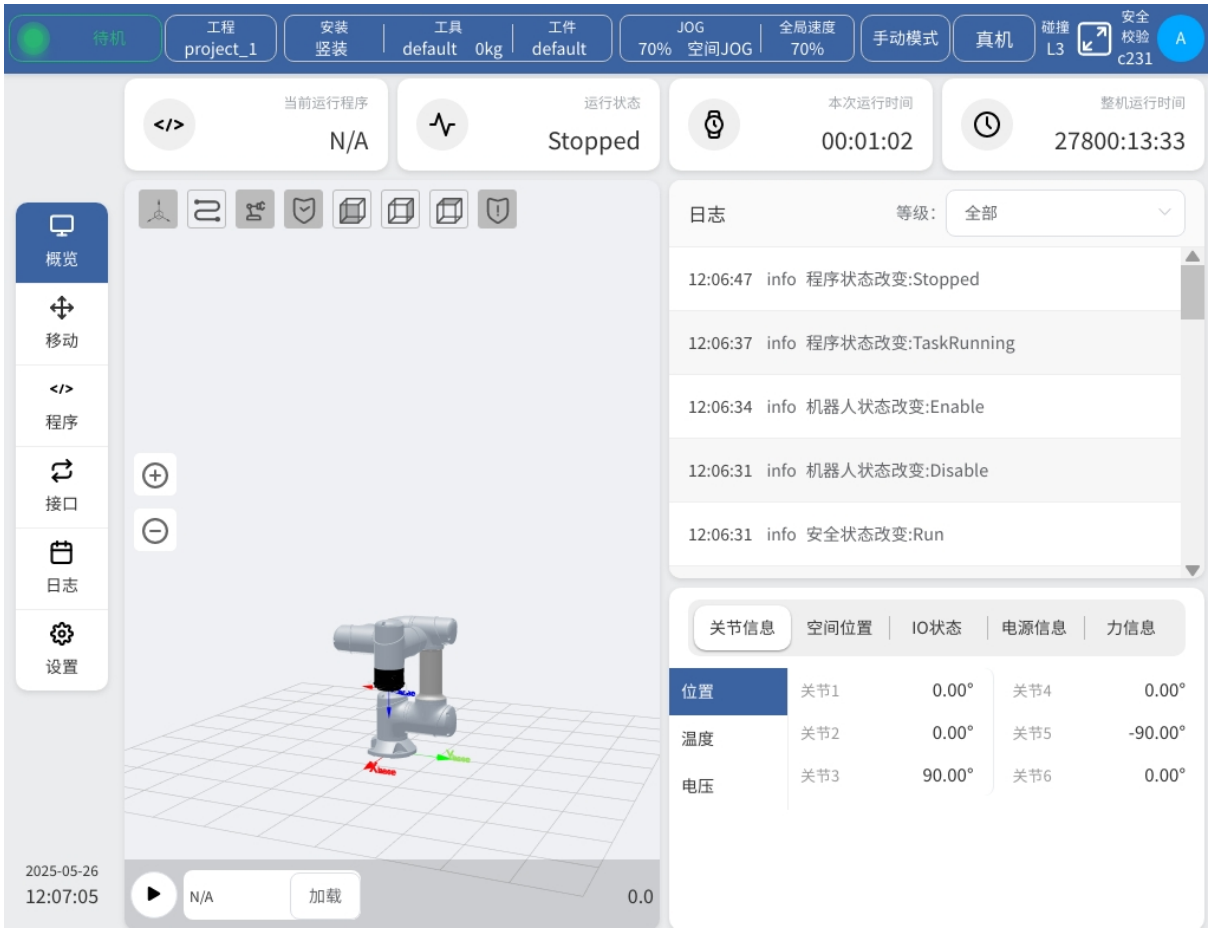
--double_arm_moveit_config

同时开发包中已包含与机器人数据接口和通信接口文件：

```
--robot_control
--duco_ros_driver
--duco_msg
```

以 GCR3-618 型号机械臂真机连接为例：

确认当前 DucoCore 控制器是否已经启动，并在“网络设置”界面查看当前控制柜 IP



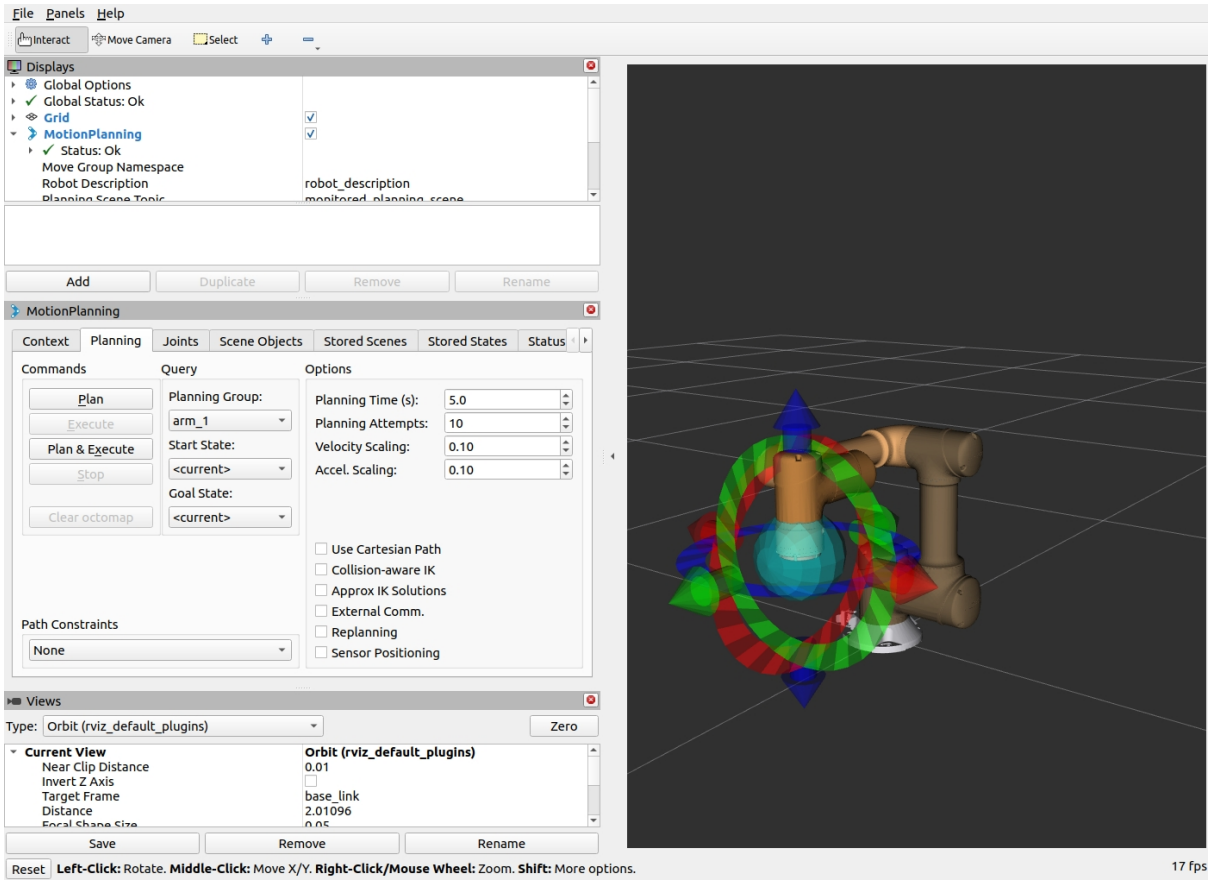
在 duco_ros2_driver 下完整编译功能包后进行 source，运行该型号的 demo.launch.py 文件，并输入对应控制柜 IP：

```
colcon build
source install/setup.bash
ros2 launch duco_gcr3_618_moveit_config demo.launch.py
server_host_1:=< 控制器 IP>
```

```
Starting >>> duco_gcr3_618_moveit_config
Finished <<< duco_gcr20_1400_moveit_config [0.16s]
Starting >>> duco_gcr5_910_moveit_config
Finished <<< duco_gcr30_1100_moveit_config [0.13s]
Starting >>> duco_gcr7_910_moveit_config
Finished <<< duco_gcr25_1800_moveit_config [0.16s]
Starting >>> duco_ros_driver
Finished <<< duco_gcr3_618_moveit_config [0.15s]
Finished <<< duco_gcr5_910_moveit_config [0.13s]
Finished <<< duco_gcr7_910_moveit_config [0.12s]
Finished <<< duco_ros_driver [0.15s]

Summary: 19 packages finished [1.29s]
ubuntu@ubuntu:~/Documents/git/duco_ros2_driver$ source install/setup.bash
ubuntu@ubuntu:~/Documents/git/duco_ros2_driver$ ros2 launch duco_gcr3_618_moveit_config demo.launch.py
 server_host_1:=192.168.120.227
[INFO] [launch]: All log files can be found below /home/ubuntu/.ros/log/2025-05-26-12-01-32-096516-ubu
ntu-6413
[INFO] [launch]: Default logging verbosity is set to INFO
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
Using load_yaml() directly is deprecated. Use xacro.load_yaml() instead.
```

确认打开的 RViz 仿真界面机械臂状态与当前实际状态一致



3.5 moveit 示例运行及操作

3.5.1 控制机械臂方案

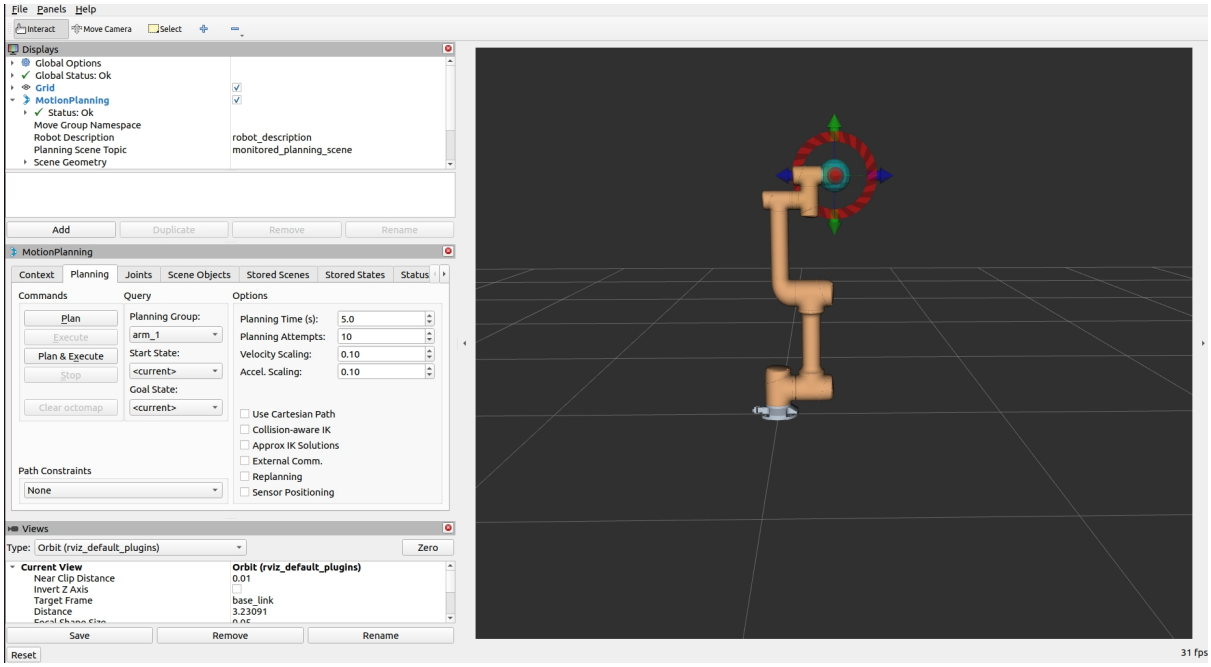
若需要控制械臂的运动，我们有两种方法。

- 第一种，通过 Rviz 软件控制。
- 第二种，通过指令在终端进行控制。

首先我们来介绍第一种方案，使用 RVIZ 控制机械臂，第二种方案将在下一章节讲述：

3.5.2 方案一：在 Rviz 软件中对机械臂进行控制

进入 Rviz 程序后，我们可以看到机械臂的模型及其末端的小球。



控制视角介绍：

1. 鼠标滚轮控制视角远近。
2. 鼠标左键可以拖拽小球，及进行视角的旋转。
3. 鼠标右键控制我们视角旋转的中心点。

此时，我们有两种方法控制机械臂。

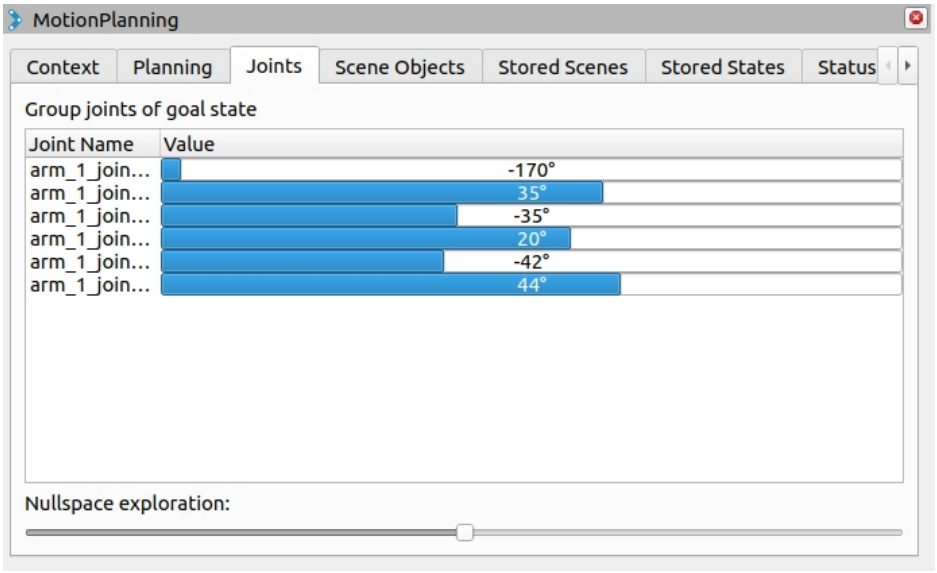
第一种：

直接拖拽小球至我们所需位置。

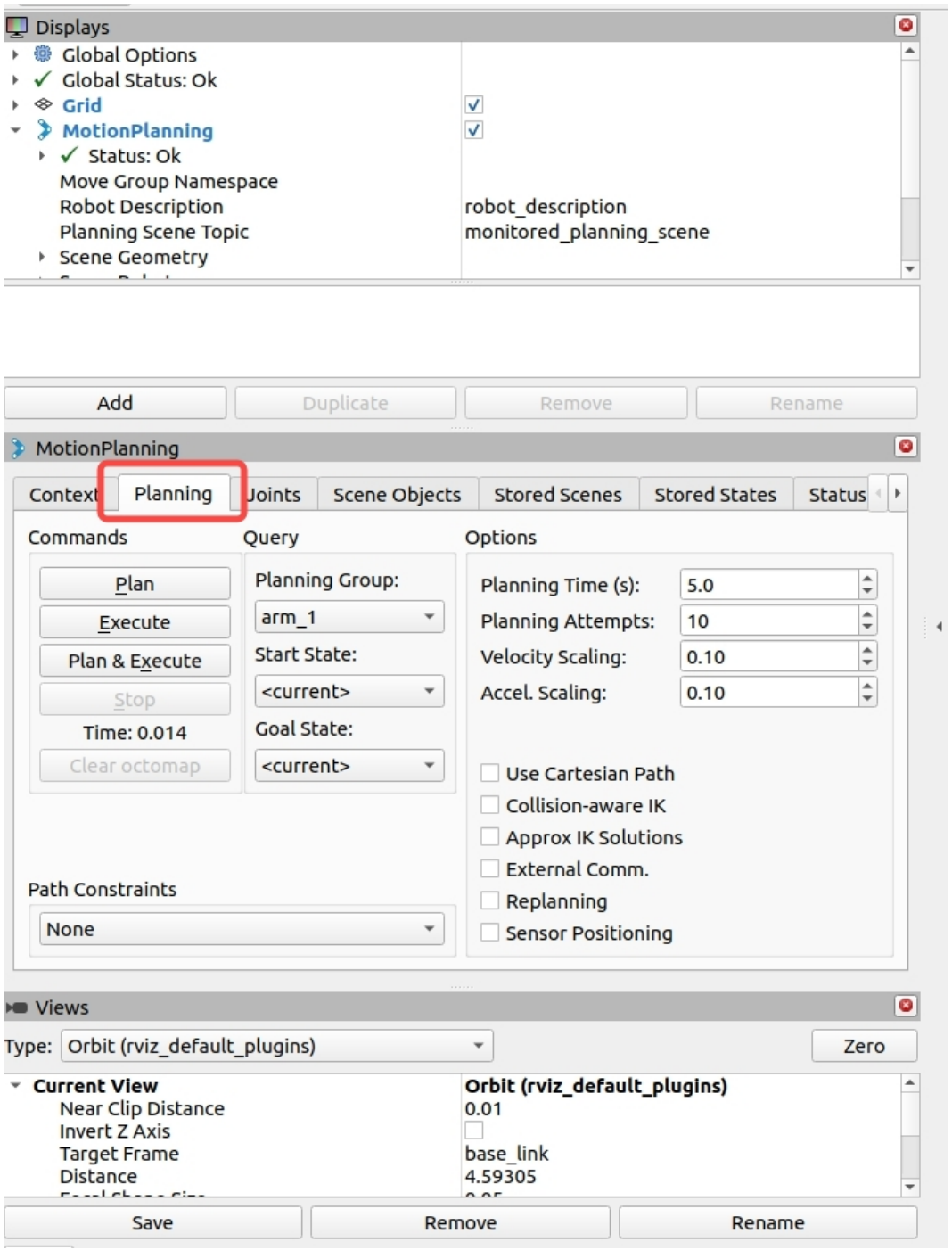
RIVIZ 中的机械臂建模会自动模拟机械臂末端处于小球位置时，各关节角度和模拟机械臂位姿。

第二种：

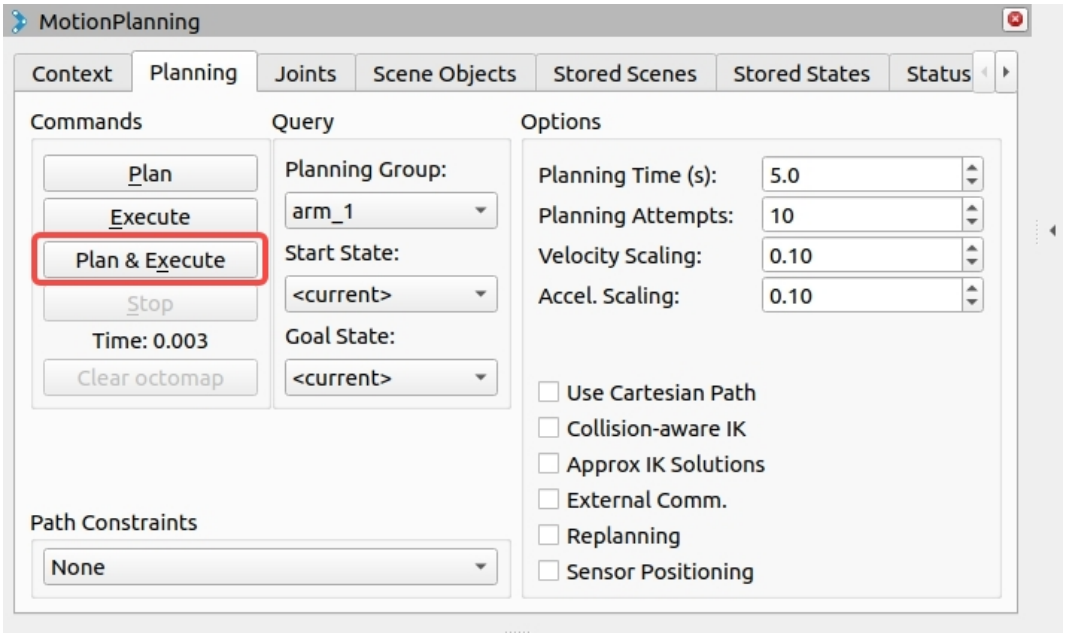
进入左侧的 MtionPlanning 中进入 Joints 界面。在此界面中可以单独控制每个机械臂关节的角度。



当调节好机械臂位置后，进入左侧的 `MtionPlanning` 中进入 `Planning` 界面。



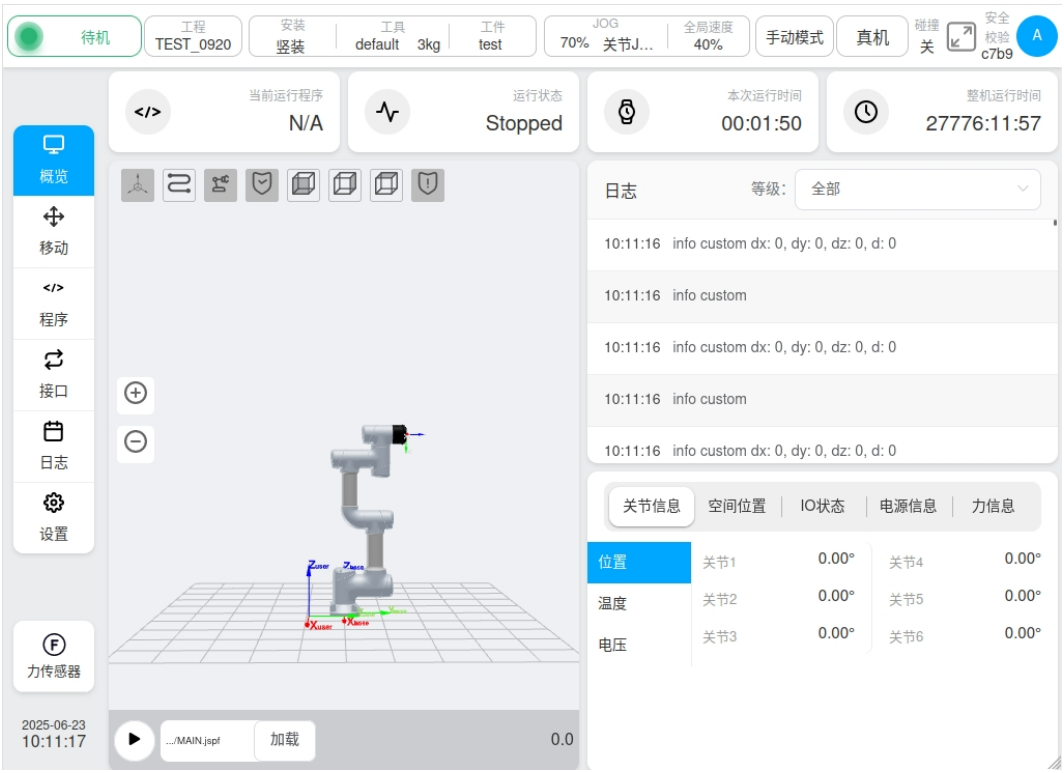
点击 plan & Execute。



plan 会自动计算好机械臂的运行轨迹。

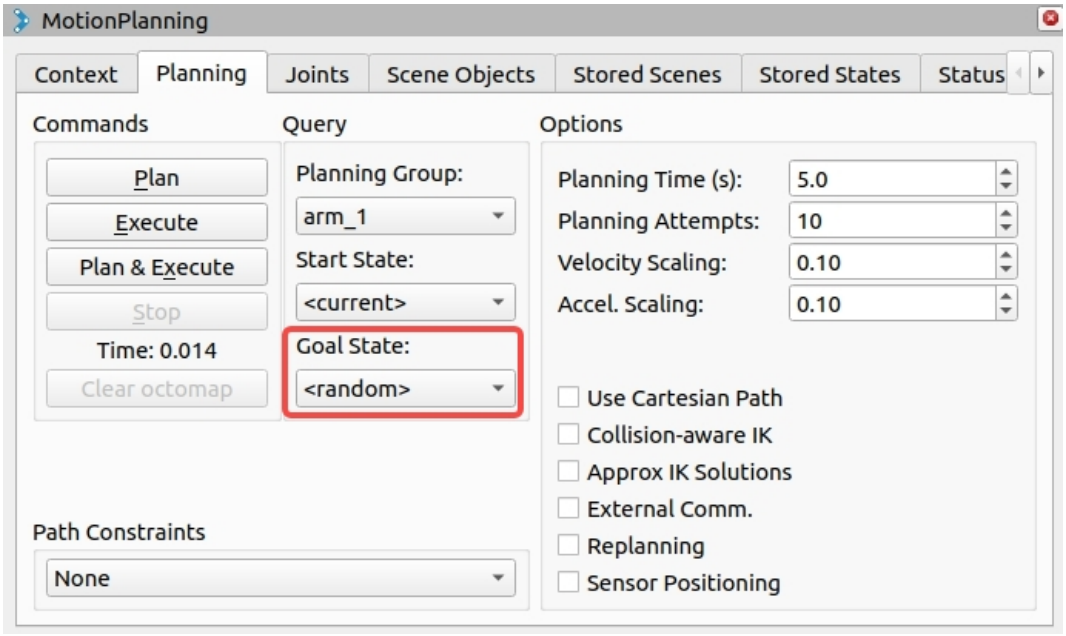
然后 Execute 会将实际的机械臂通过计算好的路径，按照设定的速度，加速度等参数，最终运行至我们在 RVIZ 中设置的机械臂位置和位姿。

（可以在浏览器中打开使用的域名地址观测实际机械臂位姿，接口，日志等信息）



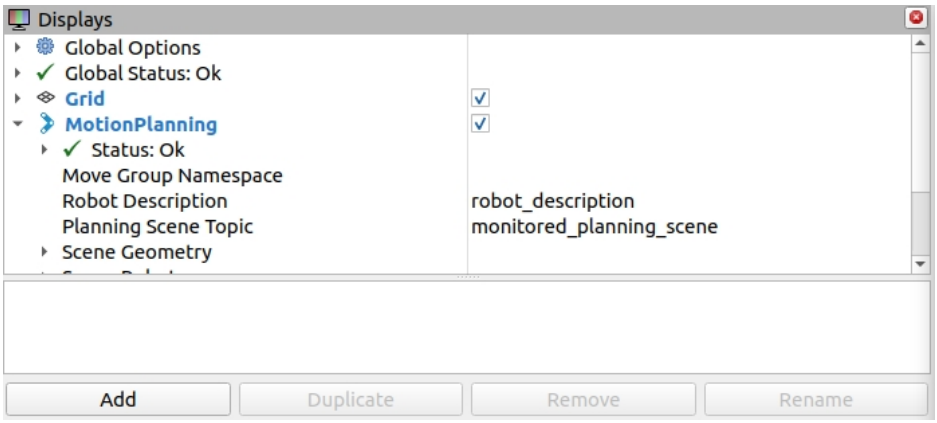
也可以单独使用 plan 功能查看规划路径，此时机械臂会模拟运动。然后单独使用 Execute 功能，让实际机械臂运动至规定位置。

若希望采用随机位姿，也可通过改变 Goal State 的参数为 random，机械臂将随机选取一个位姿。

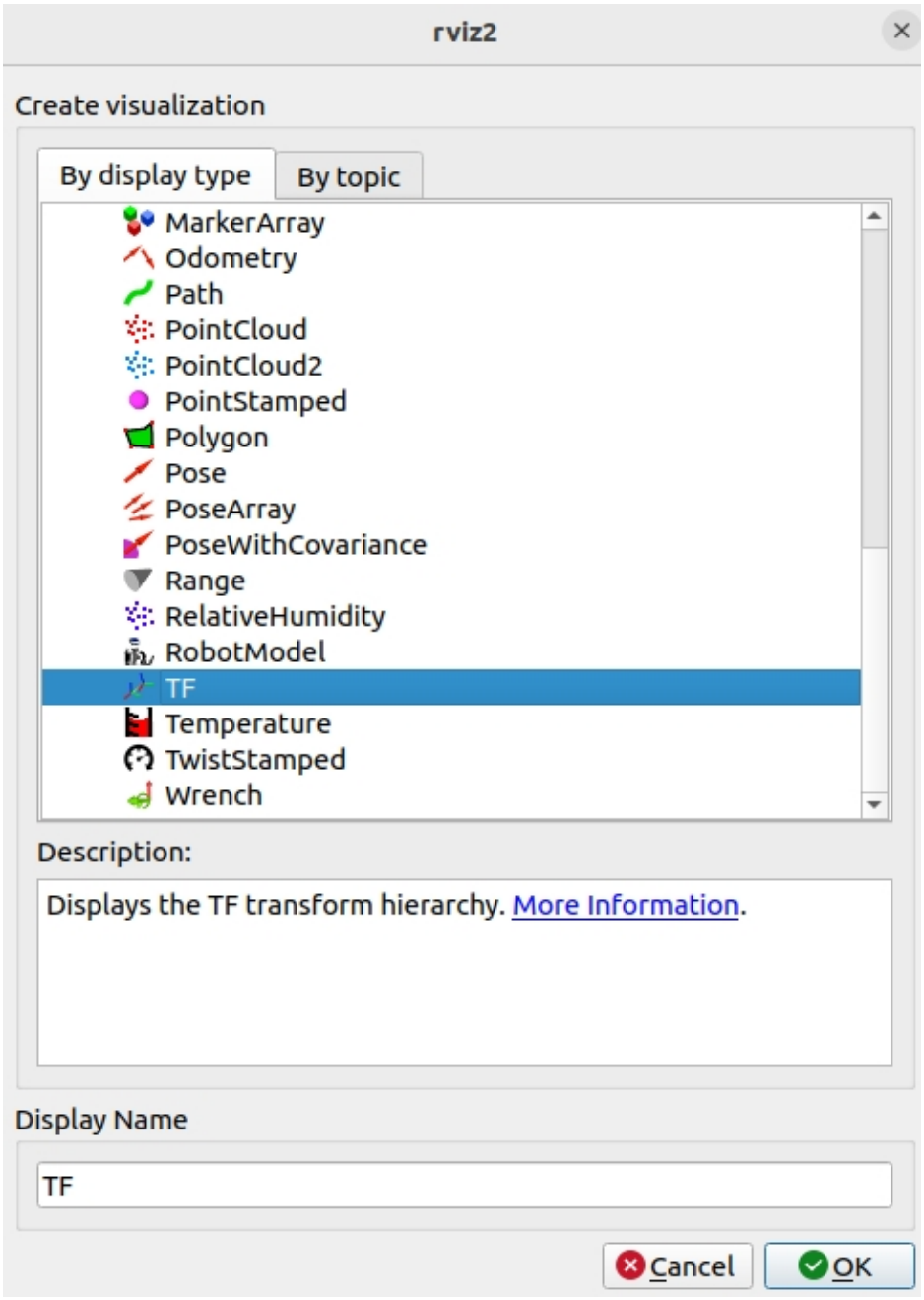


如果希望有更多功能，比如 TF 等，如何增加呢?

在 RVIZ 的左侧界面中，有一个 Displays 界面。



点击界面左下角的 Add 按钮。



选中我们所需功能后，点击 OK 按钮即可使用。

（在下一章节将介绍如何通过指令来控制机械臂）

3.6 指令控制机械臂及相关接口调用说明

3.6.1 方案二：在终端中使用指令对机械臂进行控制

我们可以通过 RobotControl，RobotIoControl 和 RobotMove 的指令，分别对机械臂的电源使能，输入输出和移动进行控制。

注意:

1. 以下指令中的‘:’后都需要有空格

2. 指令中的参数可以从 src/duco_msg/srv 中寻找相关的 srv 文件查看所需参数和参数类型

我们使用指令前，需要完成之前章节的配置过程及功能包的编译。

完成之前准备工作后，我们来到 src 资源包的目录下，打开目录下终端，将所需指令输入终端进行运行。

下面我们将详细介绍可用指令：

RobotControl

对于机器的电源和使能进行控制。

输入参数：

string command：指令

int8 arm_num：机械臂编号，设置为 0

bool block：指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回，默认为阻塞

string response：返回值为 String

command 列表：

1. poweron（上电）
2. enable（使能）
3. disable（断使能）
4. poweroff（断电）

指令模版：

```
ros2 service call /duco_robot/robot_control duco_msg/srv/RobotControl "
↪{command: "", arm_num: , block: }"
```

指令示例：

```
ros2 service call /duco_robot/robot_control duco_msg/srv/RobotControl "
↪{command: "poweron", arm_num: 0, block: true}"
```

```
oem@robot:~/git/duco_ros2_driver$ ros2 service call /duco_robot/robot_control duco_msg
/srv/RobotControl "{command: "poweron", arm_num: 0, block: true}"
requester: making request: duco_msg.srv.RobotControl_Request(command='poweron', arm_nu
m=0, block=True)

response:
duco_msg.srv.RobotControl_Response(response='4')
```

RobotIoControl

对于通用输出和输入分别进行设置和获取。

输入参数:

string command: 指令

int8 arm_num: 机械臂编号，设置为 0

int8 type: IO 类型，0 为 gen io，1 为 tool io

int8 port: IO 端口，GEN IO 范围 1-16 TOOL IO 范围 0-1

bool value: SetIO 值

bool block: 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回，默认为阻塞

string response: 返回值为 String

command 列表:

1. setIo（设置通用输出）
2. getIo（获取通用输入）

指令模版:

```
ros2 service call /duco_robot/robot_io_control duco_msg/srv/
↪RobotIoControl "{command: "", arm_num: , type: , port: , value:␣
↪, block: }"
```

指令示例:

```
ros2 service call /duco_robot/robot_io_control duco_msg/srv/
↪RobotIoControl "{command: "setIo", arm_num: 0, type: 0, port: 1,
↪ value: true, block: true}"
```

```
oem@robot:~/git/duco_ros2_driver$ ros2 service call /duco_robot/robot_io_control duco_
msg/srv/RobotIoControl "{command: "setIo", arm_num: 0, type: 0, port: 1, value: true,
block: true}"
requester: making request: duco_msg.srv.RobotIoControl_Request(command='setIo', arm_nu
m=0, type=0, port=1, value=True, block=True)

response:
duco_msg.srv.RobotIoControl_Response(response='4')
```

RobotMove

控制机械臂的移动。

输入参数:

string command: 指令

int8 arm_num: 机械臂编号，设置为 0

float32[] p: 笛卡尔目标位置

float32[] q: 目标机器人关节角位置，单位 (rad)

float32 v: 最大末端线速度，范围 [0.01, 5]，单位 m/s，当 x、y、z 均为 0 时，线速度按比例换算成角速度

float32 a: 最大末端线加速度，范围 [0.01, ∞]，单位 (m/s2)

float32 r: 轨迹融合半径，单位 m，默认值为 0，表示无融合。当数值大于 0 时表示与下一条运动融合

string tool: 设置使用的工具的名称，默认为当前使用的工具 ("default")

string wobj: 设置使用的工件坐标系的名称，默认为当前使用的工件坐标系 ("default")

bool block: 指令是否阻塞型指令，如果为 false 表示非阻塞指令，指令会立即返回，默认为阻塞

string response: 返回值为 String

command 列表:

1. movej (控制机械臂从当前状态，按照关节运动的方式移动到目标关节角状态)
2. movej2(控制机械臂从当前状态，按照各关节相位同步运动的方式移动到目标关节角状态)
3. movejpose (控制机械臂从当前状态，按照关节运动的方式移动到末端目标位置)
4. movejpose2 (控制机械臂从当前状态，按照各关节相位同步运动的方式移动到末端目标位置)
5. movel (控制机械臂末端从当前状态按照直线路径移动到目标位姿)
6. movetcp (控制机械臂沿工具坐标系直线移动一个增量)

指令模版:

```
ros2 service call /duco_robot/robot_move duco_msg/srv/RobotMove "{command:
↪"", arm_num: , v: , a: , r: , tool: "", wobj: "", block: , q: [,,,,,],␣
↪p: [,,,,,]}"
```

指令示例:

```
ros2 service call /duco_robot/robot_move duco_msg/srv/RobotMove "{command:
↪"movejpose",arm_num: 0,v: 1,a: 1,r: 0,tool: "default", wobj: "default",␣
↪block: true, q: [1,1,1,1,1,1], p: [0.49,0.14,0.44,-1.14,0,-1.57]}"
```

```
oem@robot:~/git/duco_ros2_driver$ ros2 service call /duco_robot/robot_move duco_msg/sr
v/RobotMove "{command: "movejpose",arm_num: 0,v: 1,a: 1,r: 0,tool: "default", wobj: "d
efault", block: true, q: [1,1,1,1,1,1], p: [0.45,0.15,0.4,0,0,0]}"
requester: making request: duco_msg.srv.RobotMove_Request(command='movejpose', arm_num
=0, p=[0.44999998807907104, 0.15000000596046448, 0.4000000059604645, 0.0, 0.0, 0.0], q
=[1.0, 1.0, 1.0, 1.0, 1.0, 1.0], v=1.0, a=1.0, r=0.0, tool='default', wobj='default',
block=True)

response:
duco_msg.srv.RobotMove_Response(response='4')
```

3.6.2 API 说明

RobotControl

```
/**
 * @brief 机器人上电
 * @param block 是否阻塞, 如果为false表示非阻塞指令, 指令会立即返回
 * @return 阻塞执行代表任务结束时状态, 非阻塞执行代表任务的ID
 */
int32_t power_on(bool block)
// 例子:
power_on(true)

/**
 * @brief 机器人下电
 * @param block 是否阻塞, 如果为false表示非阻塞指令, 指令会立即返回
 * @return 阻塞执行代表任务结束时状态, 非阻塞执行代表任务的ID
 */
int32_t power_off(bool block)
// 例子:
power_off(true)

/**
 * @brief 机器人上使能
 * @param block 是否阻塞, 如果为false表示非阻塞指令, 指令会立即返回
 * @return 阻塞执行代表任务结束时状态, 非阻塞执行代表任务的ID
 */
int32_t enable(bool block) ;
// 例子:
enable(true)

/**
 * @brief 机器人下使能
 * @param block 是否阻塞, 如果为false表示非阻塞指令, 指令会立即返回
 * @return 阻塞执行代表任务结束时状态, 非阻塞执行代表任务的ID
 */
int32_t disable(bool block) ;
// 例子:
disable(true)
```

RobotIoControl

```
 /**
 * @brief 读取机械臂末端的IO输入口的高低电平, 返回true为高电平,␣
↪false为低电平
 * @param num 机械臂末端的IO输出口序号, 范围从1-2
 * @return true为高电平, false为低电平
 */
bool get_tool_digital_in(int16_t num) ;
//例子:
get_tool_digital_in(1)

/**
 * @brief 读取控制柜上的用户IO输入口的高低电平, 返回true为高电平,␣
```

(续下页)

(接上页)

```
 ↪false为低电平
 * @param num 控制柜上的IO输入口序号, 范围从1-16
 * @return true为高电平, false为低电平
 */
bool get_standard_digital_in(int16_t num) ;
// 例子:
get_standard_digital_in(1)

 /**
 * @brief set_tool_digital_out
 * @param num   机械臂末端的IO输出口序号, 范围从1-2
 * @param value true为高电平, false为低电平
 * @param block 是否阻塞, 如果为false表示非阻塞指令, 指令会立即返回
 * @return 阻塞执行代表任务结束时状态, 非阻塞执行代表任务的ID
 */
int32_t set_tool_digital_out(int16_t num, bool value, bool block) ;
//例子:
set_tool_digital_out(1, true, true)

/**
 * @brief 该函数可控制控制柜上的IO输出口的高低电平
 * @param num   控制柜上的IO输出口序号, 范围从1-16
 * @param value true为高电平, false为低电平
 * @param block 是否阻塞, 如果为false表示非阻塞指令, 指令会立即返回
 * @return 阻塞执行代表任务结束时状态, 非阻塞执行代表任务的ID
 */
int32_t set_standard_digital_out(int16_t num, bool value, bool block) ;
// 例子:
set_standard_digital_out(1, true, true)
```

RobotMove

```
 /**
 * @brief 控制机械臂从当前状态, 按照关节运动的方式移动到目标关节角状态
 * @param joints_list 1-6关节的目标关节角度, 单位: rad
 * @param v 关节角速度, 单位: 系统设定速度的百分比%, 取值范围(0,100]
 * @param a 关节角加速度, 单位: 系统设定加速度的百分比%, 取值范围(0,100]
 * @param r 融合半径, 单位: 系统设定最大融合半径的百分比%, 默认值为 0,␣
↪表示无融合, 取值范围[0,50)
 * @param block 是否阻塞, 如果为false表示非阻塞指令, 指令会立即返回
 * @param op 可缺省参数
 * @param def_acc 是否使用系统默认加速度, false表示使用自定义的加速度值,␣
↪true表示使用系统自动规划的加速度值, 可缺省, 默认为false
 * @return 当配置为阻塞执行, 返回值代表当前任务结束时的状态,␣
↪若无融合为Finished, 若有融合为Interrupt.
     *          当配置为非阻塞执行, 返回值代表当前任务的id, 用户可以调用get_
↪noneblock_taskstate(id)函数查询当前任务的执行状态
 */
int32_t movej(const std::vector<double> & joints_list, double v, double a,␣
↪double r, bool block, const OP &op = _op, bool def_acc = false) ;
// 例子:
movej([1,1,1,1,1,1], 1, 1, 0, true)

/**
```

(续下页)

(接上页)

```
18  * @brief 控制机械臂从当前状态，按照关节运动的方式移动到目标关节角状态
19  * @param joints_list 1-6关节的目标关节角度，单位：rad
20  * @param v 关节角速度，范围[0.01*PI/180, 1.25*PI]，单位：rad/s
21  * @param a 关节角加速度，范围[0.01*PI/180, 12.5*PI]，单位：rad/s^2
22  * @param r 融合半径，单位：m，默认值为 0，表示无融合.
    ↪当数值大于0时表示与下一条运动融合
23  * @param block 是否阻塞，如果为false表示非阻塞指令，指令会立即返回
24  * @param op 可缺省参数
25  * @param def_acc 是否使用系统默认加速度，false表示使用自定义的加速度值，
    ↪true表示使用系统自动规划的加速度值，可缺省，默认为false
26  * @return 当配置为阻塞执行，返回值代表当前任务结束时的状态，
    ↪若无融合为Finished，若有融合为Interrupt.
27      *          当配置为非阻塞执行，返回值代表当前任务的id，用户可以调用get_
    ↪noneblock_taskstate(id)函数查询当前任务的执行状态
28  */
29 int32_t movej2(const std::vector<double> & joints_list, double v, double a,
    ↪ double r, bool block, const OP &op = _op, bool def_acc = false) ;
30 // 例子：
31  movej2([1,1,1,1,1,1], 1, 1, 0, true)
32
33  /**
34  * @brief 控制机械臂从当前状态，按照关节运动的方式移动到末端目标位置
35  * @param p 对应末端的位姿，位置单位：m，姿态以Rx、Ry、Rz表示，单位：rad
36  * @param v 关节角速度，单位：系统设定速度的百分比%，取值范围(0,100]
37  * @param a 关节加速度，单位：系统设定加速度的百分比%，取值范围(0,100]
38  * @param r 融合半径，单位：系统设定最大融合半径的百分比%，默认值为 0，
    ↪表示无融合，取值范围[0,50)
39  * @param q_near 目标点附近位置对应的关节角度，用于确定逆运动学选解，
    ↪为空时使用当前位置
40  * @param tool   设置使用的工具的名称，为空时默认为当前使用的工具
41  * @param wobj   设置使用的工件坐标系的名称，
    ↪为空时默认为当前使用的工件坐标系
42  * @param block  是否阻塞，如果为false表示非阻塞指令，指令会立即返回
43  * @param op 可缺省参数
44  * @param def_acc 是否使用系统默认加速度，false表示使用自定义的加速度值，
    ↪true表示使用系统自动规划的加速度值，可缺省，默认为false
45  * @return 当配置为阻塞执行，返回值代表当前任务结束时的状态，
    ↪若无融合为Finished，若有融合为Interrupt.
46      *          当配置为非阻塞执行，返回值代表当前任务的id，用户可以调用get_
    ↪noneblock_taskstate(id)函数查询当前任务的执行状态
47  */
48 int32_t movej_pose(const std::vector<double> & p, double v, double a,
    ↪double r, const std::vector<double> & q_near, const std::string& tool,
    ↪const std::string& wobj, bool block, const OP &op = _op, bool def_acc =
    ↪false) ;
49 // 例子：
50 movej_pose([0.49,0.14,0.44,-1.14,0,-1.57], 1, 1, 0, [1,1,1,1,1,1], "default
    ↪", "default", true)
51
52 /**
53  * @brief 控制机械臂从当前状态，按照关节运动的方式移动到末端目标位置
54  * @param p 对应末端的位姿，位置单位：m，姿态以Rx、Ry、Rz表示，单位：rad
55  * @param v 关节角速度，范围[0.01*PI/180, 1.25*PI]，单位：rad/s
56  * @param a 关节加速度，范围[0.01*PI/180, 12.5*PI]，单位：rad/s^2
57  * @param r 融合半径，单位：m，默认值为 0，表示无融合.
    ↪当数值大于0时表示与下一条运动融合
```

(续下页)

(接上页)

```
58  * @param q_near 目标点附近位置对应的关节角度，用于确定逆运动学选解，
    ↪为空时使用当前位置
59  * @param tool   设置使用的工具的名称，为空时默认为当前使用的工具
60  * @param wobj   设置使用的工件坐标系的名称，
    ↪为空时默认为当前使用的工件坐标系
61  * @param block  是否阻塞，如果为false表示非阻塞指令，指令会立即返回
62  * @param op 可缺省参数
63  * @param def_acc 是否使用系统默认加速度，false表示使用自定义的加速度值，
    ↪true表示使用系统自动规划的加速度值，可缺省，默认为false
64  * @return 当配置为阻塞执行，返回值代表当前任务结束时的状态，
    ↪若无融合为Finished，若有融合为Interrupt.
65      *         当配置为非阻塞执行，返回值代表当前任务的id，用户可以调用get_
    ↪noneblock_taskstate(id)函数查询当前任务的执行状态
66  */
67 int32_t movej_pose2(const std::vector<double> & p, double v, double a,
    ↪double r, const std::vector<double> & q_near, const std::string& tool,
    ↪const std::string& wobj, bool block, const OP &op = _op, bool def_acc =
    ↪false) ;
68 // 例子:
69 movej_pose2([0.49,0.14,0.44,-1.14,0,-1.57], 1, 1, 0, [1,1,1,1,1,1],
    ↪"default", "default", true)
70
71 /**
72  * @brief 控制机械臂末端从当前状态按照直线路径移动到目标状态
73  * @param p 对应末端的位姿，位置单位：m，姿态以Rx、Ry、Rz表示，单位：rad
74  * @param v 末端速度，范围[0.00001, 5]，单位：m/s
75  * @param a 末端加速度，范围[0.00001, ∞]，单位：m/s^2
76  * @param r 融合半径，单位：m，默认值为 0，表示无融合.
    ↪当数值大于0时表示与下一条运动融合
77  * @param q_near 目标点附近位置对应的关节角度，用于确定逆运动学选解，
    ↪为空时使用当前位置
78  * @param tool   设置使用的工具的名称，为空时默认为当前使用的工具
79  * @param wobj   设置使用的工件坐标系的名称，
    ↪为空时默认为当前使用的工件坐标系
80  * @param block  是否阻塞，如果为false表示非阻塞指令，指令会立即返回
81  * @param op 可缺省参数
82  * @param def_acc 是否使用系统默认加速度，false表示使用自定义的加速度值，
    ↪true表示使用系统自动规划的加速度值，可缺省，默认为false
83  * @return 当配置为阻塞执行，返回值代表当前任务结束时的状态，
    ↪若无融合为Finished，若有融合为Interrupt.
84      *         当配置为非阻塞执行，返回值代表当前任务的id，用户可以调用get_
    ↪noneblock_taskstate(id)函数查询当前任务的执行状态
85  */
86 int32_t movel(const std::vector<double> & p, double v, double a, double r,
    ↪const std::vector<double> & q_near, const std::string& tool = "default",
    ↪const std::string& wobj = "default", bool block = true, const OP &op = _
    ↪op, bool def_acc = false) ;
87 // 例子:
88 movel([0.49,0.14,0.44,-1.14,0,-1.57], 1, 1, 0, [1,1,1,1,1,1])
89
90 /**
91  * @brief 控制机械臂沿工具坐标系直线移动一个增量
92  * @param pose_offset 工具坐标系下的位姿偏移量
93  * @param v 直线移动的速度，范围[0.00001, 5]，单位：m/s，当x、y、z均为0时，
    ↪线速度按比例换算成角速度
94  * @param a 加速度，范围[0.00001, ∞]，单位：m/s^2
```

(续下页)

(接上页)

```
95  * @param r 融合半径，单位：m，默认值为 0，表示无融合.
    ↪当数值大于0时表示与下一条运动融合
96  * @param tool   设置使用的工具坐标系的名称,␣
    ↪为空时默认为当前使用的工具坐标系
97  * @param block  是否阻塞，如果为false表示非阻塞指令，指令会立即返回
98  * @param op 可缺省参数
99  * @param def_acc 是否使用系统默认加速度，false表示使用自定义的加速度值,␣
    ↪true表示使用系统自动规划的加速度值，可缺省，默认为false
100 * @return 当配置为阻塞执行，返回值代表当前任务结束时的状态,␣
    ↪若无融合为Finished，若有融合为Interrupt.
101     *          当配置为非阻塞执行，返回值代表当前任务的id，用户可以调用get_
    ↪noneblock_taskstate(id)函数查询当前任务的执行状态
102 */
103 int32_t tcp_move(const std::vector<double> & pose_offset, double v, double␣
    ↪a, double r, const std::string& tool, bool block, const OP &op = _op,␣
    ↪bool def_acc = false) ;
104 // 例子：
105 movetcp([0.49,0.14,0.44,-1.14,0,-1.57], 1, 1, 0, "default", "default")
```

3.7 附录

3.7.1 资源链接

全 ROS2 安装说明可参考教程：https://docs.ros.org/en/humble/index.html。

需要安装官方 ros2_control 包，再正式编译 duco 功能包，ros2_control 安装可以参考教程：https://control.ros.org/humble/index.html。

官方提供的两种安装方式中，我们推荐使用源码编译安装的方式，指令安装的方式可能导致功能包安装不齐全的问题出现。