
二次开发说明书

中科新松有限公司

Apr 26, 2024

CONTENTS

1	远程控制 APIV2.9	1
1.1	简介	1
1.2	数据类型和变量 (以 C++ 为例)	1
1.3	函数说明	4
1.3.1	实例连接控制	4
1.3.2	系统控制	5
1.3.3	任务控制	7
1.3.4	脚本控制	8
1.3.5	移动控制	9
1.3.6	can 及 485 总线	21
1.3.7	系统函数及外设	27
1.3.8	调试相关	43
1.3.9	Modbus	44
1.3.10	力控函数	46
1.3.11	运动优化函数	50
1.3.12	轨迹池函数	52
1.3.13	复合运动函数	54
1.3.14	外部轴控制函数说明	55
1.3.15	实时控制函数	60
1.3.16	其他信息	61
1.4	常见问题	62
1.4.1	RPC 客户端多线程崩溃	62
2	ROS 开发说明	63
2.1	ROS 环境安装	63
2.1.1	安装 ROS 前准备	63
2.1.2	安装过程	64
2.1.3	环境配置	64
2.1.4	小海龟测试	65
2.2	Movelt 环境安装	66
2.3	ROS 机器人开发包使用说明	67
2.3.1	系统要求	67
2.3.2	下载安装包并解压	67
2.3.3	duco_demo 使用	69

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: V3.2.0 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
};

// 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, -1: 负方向, 1: 正方向 int32_t jog_type; // 1: 空间点动, 2: 关节点动 int32_t axis_num; // 关节索引号 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 double step_jointValue; // 关节步进距离, 单位:m、rad double step_cartvalue; // 末端步进距离, 单位:m、rad

};

// 可达性检测相关信息 struct ReachabilityParams {

    bool result; // 可达性确认结果 std::vector<std::vector<double> > joints_pos; // 可达性检测成功时所对应的所有关节位置

};

// 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

};

// 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

};

```

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_nonblock_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_nonblock_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_nonblock_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_nonblock_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_nonblock_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_nonblock_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_nonblock_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_nonblock_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_nonblock_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_nonblock_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> &filelist, 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; }
```

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控制柜上的 IO 的输出。

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: 轨迹触发控制柜 IO 的输出序号，范围 1-16。

trig_value_1: 轨迹触发控制柜 IO 的电平高低，false: 低电平，true: 高电平。

trig_time_1: 当 time_or_dist_1 为 1 时，代表轨迹运行多少时间长度触发 IO, 单位 ms。

trig_dist_1: 当 time_or_dist_1 为 2 时，代表轨迹运行多少距离长度触发 IO, 单位 m。

trig_event_1: 轨迹触发的用户自定义事件名称。

`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`: 轨迹触发控制柜 IO 的输出序号, 范围 1-16。
`trig_value_2`: 轨迹触发控制柜 IO 的电平高低, `false`: 低电平, `true`: 高电平。
`trig_time_2`: 当 `time_or_dist_2` 为 1 时, 当 `trig_time_2 >= 0` 时, 代表轨迹运行剩余多少时间长度触发 IO, 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触发 IO。
`trig_dist_2`: 当 `time_or_dist_2` 为 2 时, 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触发 IO。
`trig_event_2`: 轨迹触发的用户自定义事件名称。

```
struct MoveJogTaskParams { int jog_direction; int jog_type; int axis_num; double vel; int jog_coordinate; bool use_step; double step_jointValue; double step_cartValue; }
```

特殊类型说明:

该参数类型用于控制机械臂执行关节和笛卡尔空间点动。

参数说明:

`jog_direction`: 运动方向, -1: 负方向, 1: 正方向。
`jog_type`: 运动方式, 1: 空间 Jog, 2: 关节 Jog。
`axis_num`: 关节的索引号。
`vel`: 速度百分比。
`jog_coordinate`: 参考的坐标系, 0: 世界, 1: 基座, 2: 工具, 3: 工件。
`use_step`: 步进模式, `true`: 步进模式, `false`: 连续模式。
`step_jointValue`: 关节步进距离, 单位: m、rad。
`step_cartValue`: 空间步进距离, 单位: m、rad。

```
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`: 关节角速度, 范围 $[0.01*PI/180, 1.25*PI]$, 单位 (rad/s)。
`a`: 关节角加速度, 范围 $[0.01*PI/180, \infty]$, 单位 (rad/s²)。
`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。
`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, 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 $[-2*\text{PI}, 2*\text{PI}]$, 单位 (rad)。

v: 关节角速度, 范围 $[0.01*\text{PI}/180, 1.25*\text{PI}]$, 单位 (rad/s)。

a: 关节加速度, 范围 $[0.01*\text{PI}/180, \infty]$ 单位 (rad/s²)。

r: 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。

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。

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 $[-2*\text{PI}, 2*\text{PI}]$, 单位 rad。

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)。

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。

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。

```
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)
```

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: 圆弧运动中间点。

p2: 圆弧运动结束点。

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)。

r: 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。

mode: 姿态控制模式。

0: 姿态与终点保持一致;

1: 姿态与起点保持一致;

2: 姿态受圆心约束;

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。

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。

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: 圆周运动经过点。

p2: 圆周运动经过点。

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)。

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。

mode: 姿态控制模式。

1: 姿态与终点保持一致;

2: 姿态受圆心约束。

qnear: 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, 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。

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`: 表示工具坐标系下的位姿偏移量。

`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。

`a`: 加速度, 范围 $[0.00001, \infty]$, 单位 (m/s²)。

`r`: 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。

`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。

`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`: 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。

`a`: 加速度, 范围 $[0.00001, \infty]$, 单位 (m/s²)。

`r`: 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。

`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_nonblock_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 $[0.01*PI/180, \infty]$ ，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_nonblock_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 $[0.00001, \infty]$ ，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_nonblock_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_nonblock_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_nonblock_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_nonblock_taskstate(id)** 函数查询当前任务的执行状态。

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: 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。

! 注意

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_nonblock_taskstate(id)` 函数查询当前任务的执行状态。

! 注意

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。

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, 运动过程中不考虑笛卡尔空间路径。

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**)。

a : 最大关节加速度, 范围 $[0.01*PI/180, \infty]$, 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, 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。

返回值:

返回值代表当前任务的 **id** 信息, 用户可以调用 `get_nonblock_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)

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笛卡尔状态, 通过关节空间运动。

参数说明:

p : 目标工件坐标系下的末端位姿, 单位 (**m/rad**)。

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

a : 关节加速度, 范围 $[0.01 \cdot \pi / 180, \infty]$, 单位 (rad/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, 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。

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_nonblock_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)

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一个增量, 通过关节空间运动。

参数说明:

p : 目标工件坐标系下的末端位姿, 单位 (m/rad)。

v : 关节速度, 范围 $[0.01 \cdot \pi / 180, 1.25 \cdot \pi]$, 单位 (rad/s)。

a : 关节加速度, 范围 $[0.01 \cdot \pi / 180, 12.5 \cdot \pi]$, 单位 (rad/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。

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。

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_nonblock_taskstate(id)` 函数查询当前任务的执行状态。

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: 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)。

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。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_nonblock_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_nonblock_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)

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。

返回值:

返回值代表当前任务结束时的状态。

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_nonblock_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 为高低电平, 1 为脉冲。

freq: 脉冲频率, 单位: HZ。

duty_cycle: 脉冲占空比, 单位: %。

返回值:

当前任务结束时的状态。

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_nonblock_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_nonblock_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: *

portnum : 控制柜功能 IO 输入口序号，范围从 1-8。

返回值: *

bool，返回 **true** 为高电平，**false** 为低电平。

get_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;

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_nonblock_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。

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。

参数说明:

num : 控制柜上的模拟电流通道的序号, 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的序号, 范围从 1-4;

value : 设置的模拟电流值;

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_nonblock_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 (vector<double>: wobj_offset)

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。

参数说明:

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

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*m²。

返回值:

返回当前任务结束时的状态。

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

函数说明:

计算运动学逆解, 在求解过程中, 会选取靠近当前机械臂关节位置的解。

参数说明:

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, 为空使用当前关节值。

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:

返回的关节位置列表 [q1,q2,q3,q4,q5,q6]

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

函数说明:

计算机械臂的正运动学, 求解给定 TCP 在给定 wobj 下的值。

参数说明:

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:

返回末端姿态列表 [x,y,z,rx,ry,rz]

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工具末端点的速度。

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末端点的加速度。

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:

该函数可获取当前末端的力矩信息。

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工具坐标系下的力矩信息。

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。

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。

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()

函数说明:

可达性检测。

参数说明:

无。

返回值:

全局速度的百分比。

switch_mode(int32_t mode)

函数说明:

手自动模式切换。

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。

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;

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: frequency)
```

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。

frequency: 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进行末端力控运动。

参数说明:

无

返回值:

返回值代表当前任务的 id 信息。

fc_stop()

函数说明:

该指令控制机械臂退出末端力控。

参数说明:

无

返回值:

返回值代表当前任务的 id 信息。

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**。

ref_ft: 6 个笛卡尔空间方向末端力控参考力，范围 [-1000, 1000]，X/Y/Z 方向单位 (N)，RX/RX/RZ 方向单位 (Nm)，方向符号参考末端力控参考坐标系方向。

damp: 6 个笛卡尔空间方向末端力控阻尼，范围 [-10000, 10000]，X/Y/Z 方向单位 (N/(m/s))，RX/RX/RZ 方向单位 (Nm/(°/s))。

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)，范围 [-2*PI, 2*PI]，RX/RX/RZ 方向单位 (rad/s)。

number_list:6 个笛卡尔空间方向末端与环境接触力死区, 范围 [-1000, 1000], X/Y/Z 方向单位 (N), RX/RX/RZ 方向单位 (Nm)。

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。

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。

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 位参考工件坐标系。

返回值:

返回值代表当前任务的 id 信息。

fc_move()

函数说明:

该指令控制机械臂仅产生末端力控运动, 为阻塞型指令。

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)

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监控。

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参考力, X/Y/Z 方向单位 (N), RX/RX/RZ 方向单位 (Nm), 方向符号参考末端力安全监控参考坐标系方向。

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。

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。

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。

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 个末端力传感器出力设置标志位, 需要设置为 **true**, 不需要设置为 **false**。

ref_ft: 6 个末端力传感器出力设置目标值, X/Y/Z 方向单位 (N), RX/RX/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调过后续运动函数, 直到 **fc_stop()** 函数被执行停止末端力控。

参数说明:

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

range: 位置判断条件偏移范围大小, X/Y/Z 方向单位 (m), RX/RX/RZ 方向单位 (rad)。

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。

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)。

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。

参数说明:

middle_value: 速度判断条件绝对值, X/Y/Z 方向范围 [-5, 5], 单位 (m/s), RX/RX/RZ 方向范围 [-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)。

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。

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)。

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。

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)。

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。

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)。

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。

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反馈读数。

参数说明:

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

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, 在满足系统约束前提下, 机械臂以尽可能高的速度跟踪路径。

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、电功率模型计算得到最优加速度大小，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，该函数不起作用。

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。

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。

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：

该指令用以开启机械臂振动抑制功能。

参数说明：

无。

返回值：

任务结束时状态。

disable_vibration_control()

函数说明：

该指令用以关闭机械臂振动抑制功能。

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。

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。

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_nonblock_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，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 $[0.01 \cdot \pi / 180, 1.25 \cdot \pi]$ ，单位 (rad/s)。

acc : 最大关节加速度，范围 $[0.01 \cdot \pi / 180, 12.5 \cdot \pi]$ ，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，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`：最大末端速度，范围 [0.00001, 5]，单位 (m/s)。

`acc`：最大末端加速度，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。

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。

`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, int amp, int el_offset, int az_offset, int up_height, const vector<int>& time, const vector<Op>& op_list)

函数说明：

该指令设置复合运动的相关参数。

参数说明：

`type`：复合运动类型。1：平面三角形轨迹，2：平面正旋轨迹，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，1：工具 XOZ。

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。
op_list : 二维的 OP 参数列表
返回值:
任务结束时状态。

enable_combined_motion ()

函数说明:
该指令用以开启复合运动。
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。

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

函数说明:

该指令用于控制外部轴移动。

参数说明:

scheme_name : 目标外部轴方案名称。

epose : 目标外部轴方案所对应自由度位置 (三维), 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, 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_nonblock_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, const string&
scheme_name, vector<double> epos, 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/s²)。

`rad`：融合半径，单位 (m)。

`scheme_name`：目标外部轴方案名称。

`epos`：目标外部轴方案所对应自由度位置 (三维)，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)。

`epos_v`：外部轴最大规划速度，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_nonblock_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, const string& scheme_name, vector<double> epos, 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/s²)。

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。

epos : 目标外部轴方案所对应自由度位置 (三维), 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)。

epos_v : 外部轴最大规划速度, 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_l_eaxis (vector<double> p, double v, double a, double rad, vector<double> q_near, string tool, string wobj, const string& scheme_name, vector<double> epos, 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/s²)。

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。

epos : 目标外部轴方案所对应自由度位置 (三维), 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)。

epos_v : 外部轴最大规划速度, 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。

```
movec_eaxis (vector<double> p1, vector<double> p2, double v, double a, double rad, vector<double> q_near, string tool, string wobj, const string& scheme_name, vector<double> epos, 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: *

1: 圆弧运动中间点位姿。

2: 圆弧运动结束点位姿。

 : 末端速度, 范围 (0, 5], 单位 (m/s)。

 : 末端加速度, 范围 (0, ∞], 单位 (m/s²)。

ad : 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。

_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。

ool : 设置使用的工具的名称, 为空时默认为当前使用的工具。

obj : 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。

cheme_name : 目标外部轴方案名称。

pos : 目标外部轴方案所对应自由度位置 (三维), 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**)。

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

lock : 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。

p : 详见上方 **Op** 特殊类型说明 (距离触发无效), 可缺省参数。

ef_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。

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, 1: 关节位置, 2: 关节速度, 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_noblock_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 `realtimeControlData` 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.

1.3.16 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 : 存放软件版本号的字符串。

返回值:

当前机器人控制器的软件版本号。

1.4 常见问题

1.4.1 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。

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

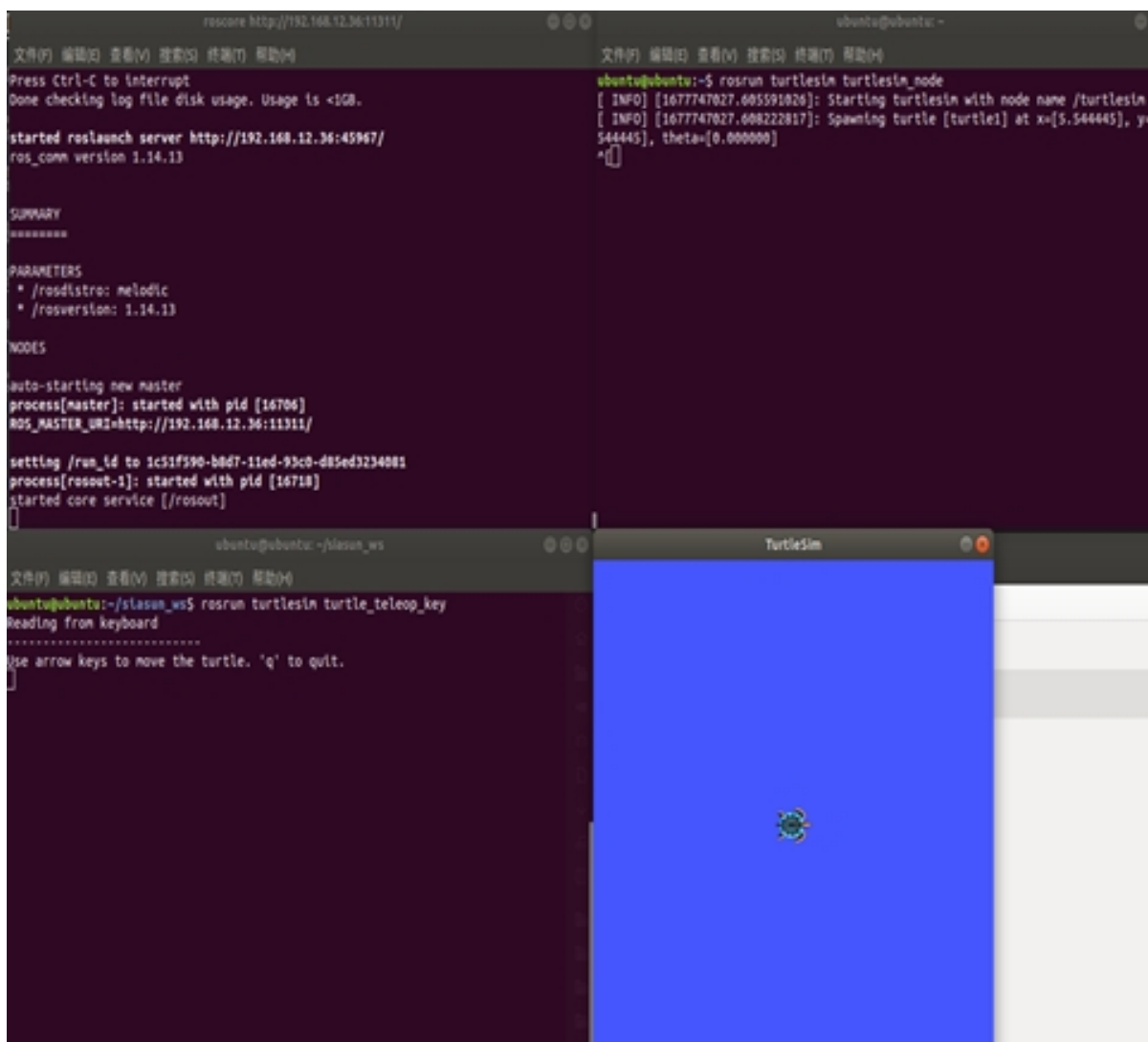
- rosrn turtlesim turtlesim_node

第三个终端输入

- rosrn 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

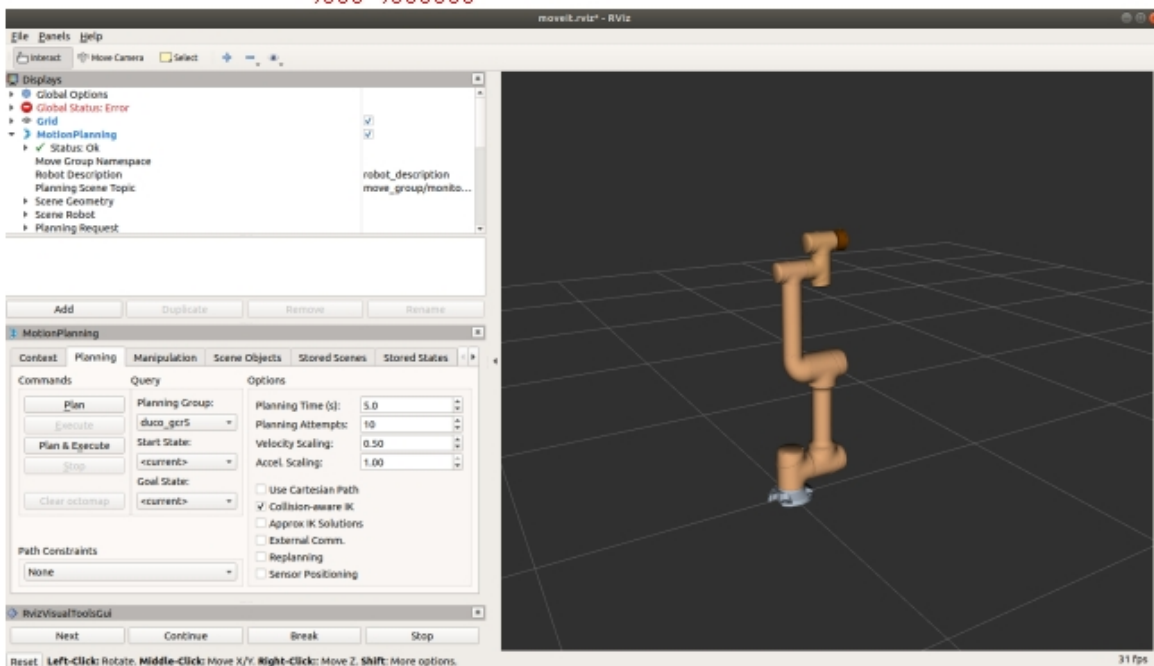
- `cd ~/duco_ws`
- `source ./devel/setup.bash`
- `catkin_make`

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

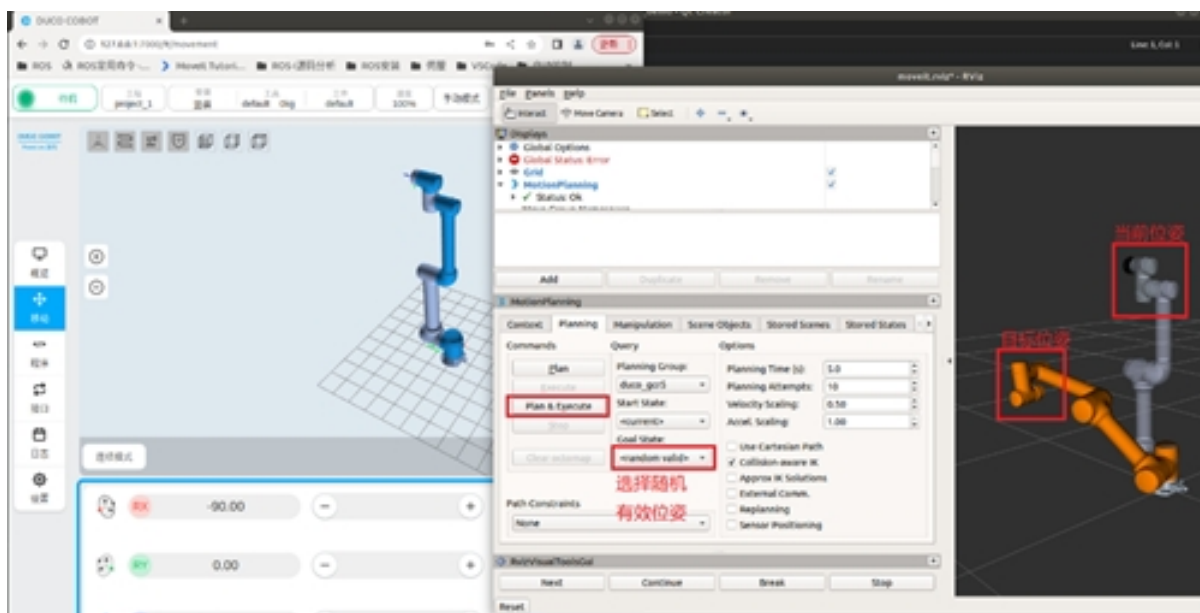
打开终端输入下列指令：

- `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

- `roslaunch duco_demo gcr5_demo`

```

ubuntu@ubuntu: ~/slasun_ws
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
(y/n)?
y
ubuntu@ubuntu:~/slasun_ws$ ^C
ubuntu@ubuntu:~/slasun_ws$ ^C
ubuntu@ubuntu:~/slasun_ws$ ^C
ubuntu@ubuntu:~/slasun_ws$ ^C
ubuntu@ubuntu:~/slasun_ws$ roslaunch duco_demo gcr5_demo
[ERROR] [1677746384.030358904]: Semantic description is not specified for the same robot as the URDF
[ INFO] [1677746384.031015879]: Loading robot model 'gcr5'...
[ WARN] [1677746384.108298638]: The root link base_link has an inertia specified in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an extra dummy link to your URDF.
[ INFO] [1677746385.122820231]: Ready to take commands for planning group duco_gcr5.
[ INFO] [1677746385.350137398]: RemoteControl Ready.
[ WARN] [1677746385.850558021]: Topic '/rviz_visual_tools' unable to connect to any subscribers within 0.5 sec. It is possible initially published visual messages will be lost.
[ INFO] [1677746385.850614694]: Planning frame: base_link
[ INFO] [1677746385.850631668]: End effector link: link_6
Waiting to continue: Press 'next' in the RvizVisualToolsGui window to start the demo

```

第二步：点击 RVIZ 窗口里的 next 实现对机器人控制。

