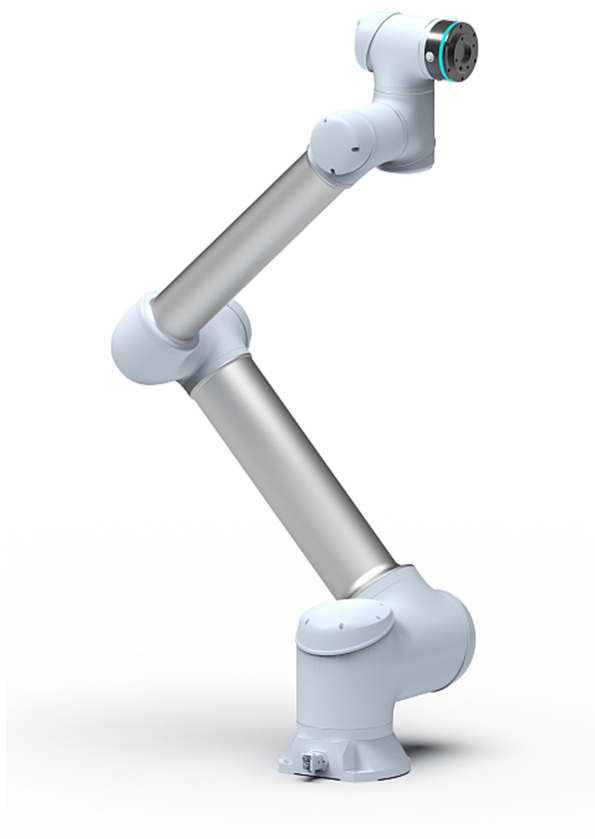


DUCO CORE 脚本手册

文档版本 V2.7



中科新松有限公司

2023 年 11 月 29 日

版权所有©中科新松有限公司 2023。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制文档内容的部分或全部，并不得以任何形式传播。

商标声明

“SIASUN”、“新松”、“SIASUN 新松”等文字或形象均进行了商标注册保护，注册商标信息可见于公开的商标注册信息中。

本手册提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受中科新松有限公司商业合同和条款的约束，本手册中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围内。除非合同另有约定，中科新松有限公司对本手册内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本手册内容会不定期进行更新。除非另有约定，本手册仅作为使用指导，本手册中的所有陈述、信息和建议不构成任何明示或暗示的担保。

中科新松有限公司

地址：上海市浦东新区金藏路 257 号 邮编：201206

网址：<http://www.siasun-in.com>

修订历史

1. 2021 年 7 月 30 日 文档版本号 V1.0 软件版本 V1.1.0 首次修订

2. 2021 年 10 月 09 日 文档版本号 V1.1 软件版本 V1.2.0
 - 1) 1.5.1 章节增加 OP 特殊参数描述,move1,movec,tcp_move,tcp_move_2p,spline 增加 op 参数
 - 2) 1.5.1 章节, rad 的定义描述修改
 - 3) 1.5.1 章节增加 move_circle 脚本
 - 4) 1.5.1 章节增加 hand_teach 脚本
 - 5) 1.5.2 章节, socket_open 修正描述
 - 6) 1.5.2 章节, socket_read_str_num, socket_read_num 修正示例
 - 7) 1.5.3 章节, write_raw_data_485 修正描述, 增加参数说明 tail
 - 8) 1.5.3 章节, tool_read_raw_data_485 等, 与工具末端相关的 485 函数修正示例
 - 9) 1.5.4 章节增加 get_function_digital_in/out
 - 10) 1.5.4 章节修正 write_reg,read_reg 描述及示例
 - 11) 1.5.4 章节增加 start/stop_record_track
 - 12) 1.5.4 章节增加 set_wobj_offset
 - 13) 1.5.4 章节增加 set_load_data
 - 14) 1.5.4 章节增加 collision_detect
 - 15) 1.5.7 章节, 修改 num2str 输入
 - 16) 1.5.10 章节, 修改 fc_wait_logic 参数说明

3. 2021 年 11 月 19 日 文档版本号 V1.2 软件版本 V1.3.0
 - 1) 1.4.5 章节增加 goto 语句
 - 2) 1.4.6 章节增加 break 语句
 - 3) 1.5.4 章节增加 get_standard_digital_out 脚本
 - 1) 1.5.4 章节增加 get_tool_digital_out 脚本
 - 2) 1.5.4 章节增加 get_function_reg_in 脚本
 - 3) 1.5.4 章节增加 get_function_reg_out 脚本
 - 4) 1.5.4 章节增加 get_value 脚本
 - 5) 1.5.4 章节增加 set_value 脚本
 - 6) 1.5.4 章节 set_tool_data 增加惯量矩阵参数
 - 7) 1.5.8 章节增加脚本 str_split 脚本
 - 8) 1.5.8 章节增加脚本 str_at 脚本
 - 9) 1.5.8 章节, 修正涉及的字符串相关函数索引参数从 1 开始
 - 10) 1.5.10 章节 fc_config 函数添加死区参数

4. 2021 年 12 月 25 日 文档版本号 V1.3 软件版本 V1.4.0
 - 1) 1.5.1 章节 movej,movej_pose 增加 OP 参数及注意事项;
 - 2) 1.5.1 章节增加 movej2,movej_pose2 脚本;
 - 3) 增加 1.5.11 速度优化函数章节;

5. 2022 年 5 月 30 日 文档版本号 V1.4 软件版本 V1.4.4
 - 1) 1.5.7 章节, num2str 函数增加小数点后保留位数的设置参数;
6. 2022 年 7 月 7 日 文档版本号 V1.5 软件版本号 V2.2.1
 - 1) 1.5.1 章节 Op 参数含义更改, movec 和 move_circle 函数模式参数取值更改;
 - 2) 1.5.1 章节增加 move_spiral 螺旋轨迹脚本函数;
 - 3) 1.5.3 章节 read_raw_data_485 函数增加等待时间参数;
 - 4) 1.5.4 章节增加 get_target_joints_position、get_target_joints_speed、get_target_joints_acceleration 和 get_target_joints_torque 脚本函数;
 - 5) 1.5.11 章节增加 enable_acc_optimization 和 disable_acc_optimization 加速度优化函数;
7. 2022 年 7 月 29 日 文档版本号 V1.6 软件版本号 V2.3.0
 - 1) 1.5.1 章节 tcp_move 函数增加工具坐标系参数, tcp_move_2p 函数增加工具和工件坐标系参数;
 - 2) 修改 1.5.11 章节名称, 新增包含开启和退出机械臂末端振动抑制的函数;
8. 2022 年 9 月 14 日 文档版本号 V1.7 软件版本号 V2.4.0
 - 1) 1.5.1 章节增加 is_moving 函数;
 - 2) 1.5.9 章节增加 float2word 和 word2float 函数;
 - 3) 增加 1.5.12 复合运动章节, 包含 combine_motion_config、enable_combined_motion 和 disable_combined_motion 相关函数;
9. 2022 年 9 月 21 日 文档版本号 V1.8 软件版本号 V2.4.1
 - 1) 1.5.4 章节删除 set_wobj 函数;
 - 2) 1.5.2 章节增加设置 2001 端口频率的函数;
10. 2022 年 10 月 19 日 文档版本号 V1.9 软件版本号 V2.5.0
 - 1) 1.5.1 章节 spline 增加注意事项;
 - 2) 1.5.2 章节增加 socket_write_byte_list, socket_read_byte_list 函数;
 - 3) 1.5.4 章节 start_record_track 函数增加轨迹类型参数;
 - 4) 1.5.11 章节增加奇异规避开启与关闭函数;
11. 2022 年 11 月 18 日 文档版本号 V2.0 软件版本号 V2.6.0
 - 1) 1.5.1 章节 OP 参数增加自定义事件名称参数, replay 函数增加复现方式参数;
 - 2) 1.5.4 章节 start_record_track 函数增加坐标系参数;
 - 3) 1.5.6 章节增加 modbus_write_multiple_regs 和 modbus_write_multiple_coils 函数;
 - 4) 1.5.11 章节增加融合段姿态约束功能相关函数;
12. 2023 年 1 月 10 日 文档版本号 V2.1 软件版本号 V2.7.0
 - 1) Joint_list 替换为 joints;
 - 2) 增加函数的示例;
13. 2023 年 3 月 22 日 文档版本号 V2.2 软件版本号 V2.8.0

-
- 1) 1.5.4 章节增加 `get_tcp_pose_coord`, `get_tcp_force_tool`, `single_brake_control`, `timer_start`, `timer_end`;
 - 2) 1.2.3 章节增加 `pose_speed`, `pose_acc`, `joint_speed`, `joint_acc`, `timer` 等变量类型;
14. 2023 年 4 月 26 日 文档版本 V2.3 软件版本号 V2.8.1
- 1) 增加移动控制和力控函数的参数限制;
15. 2023 年 6 月 21 日 文档版本 V2.4 软件版本号 V2.9.0
- 1) 1.5.4 章节修改 `get_tcp_pose_coord` 函数的参数名称;
16. 2023 年 8 月 9 日 文档版本 V2.5 软件版本号 V2.10.0
- 1) 1.5.4 章节增加设置控制柜通用 IO 输出信号类型的函数 `set_digital_output_mode`;
 - 2) 1.5.7 章节增加数学计算的函数;
17. 2023 年 8 月 9 日 文档版本 V2.6 软件版本号 V2.10.0
- 1) 1.5.12 章节 `combine_motion_config` 参考平面纠正勘误, 改为工件坐标系;
18. 2023 年 11 月 29 日 文档版本 V2.7 软件版本号 V3.1.0
- 1) 1.5.1 章节 `spline` 函数增加融合半径参数, 增加 `set_blend_ahead` 融合预读取函数
 - 2) `movej`、`movej_pose`、`movej2`、`movej_pose2`、`movec`、`movel`、`move_circle`、`tcp_move`、`tcp_move_2p`、`move_spiral`、`spline` 增加是否使用自定义加速度的可缺省参数;

目录

修订历史	3
1 脚本函数说明	7
1.1 简介	7
1.2 数据类型和变量	7
1.2.1 内置数据类型	7
1.2.2 系统常量	8
1.2.3 变量	8
1.3 表达式	9
1.3.1 算术运算表达式	9
1.3.2 关系运算表达式	9
1.3.3 逻辑运算表达式	10
1.3.4 赋值表达式	10
1.3.5 函数调用表达式	10
1.4 语句	10
1.4.1 循环语句 <code>while</code>	10
1.4.2 函数定义	10
1.4.3 <code>return</code> 语句	11
1.4.4 条件控制语句	11
1.4.5 <code>goto</code> 语句	13
1.4.6 <code>break</code> 语句	13
1.5 系统函数说明	14
1.5.1 移动控制	14
1.5.2 网络通信	24
1.5.3 <code>can/485</code> 总线	28
1.5.4 系统函数及外设	34
1.5.5 调试相关	53
1.5.6 <code>Modbus</code>	54
1.5.7 数学运算函数	56
1.5.8 字符串相关函数	64
1.5.9 辅助函数	67
1.5.10 力控函数	72
1.5.11 运动优化函数	78
1.5.12 复合运动函数	81

1 脚本函数说明

1.1 简介

此脚本函数手册适用于新松协作机器人编程使用。



注意

- 脚本函数名不能使用如下划线、标点符号等特殊字符作为起始字符。

1.2 数据类型和变量

1.2.1 内置数据类型

空类型: nil, 表示没有任何有效值。

算术类型: 系统内置算术类型为双精度实浮点数(number)。

字符串类型 string: 例如 “siasun robot”。

布尔类型 boolean: 例如 true、false。

列表类型 list: 可容纳其他数据类型的表(数组结构): 例如

{1,2,3,4,5}、{“a”,“b”,“c”}。

注: 为了方便以下函数说明, 定义以下类型:

joints: 特指长度为 6 的描述关节相关数据(位置信息、速度信息等)的双精度实浮点数列表, 单位 rad, 例如{1,2,3,4,5,6}。

pose: 特指长度为 6 的描述位姿数据的双精度实浮点数列表, 单位 m、rad, 例如{1,2,3,4,5,6}。

num_list: 特指双精度实浮点数列表, 例如{1.1,2.2,3,4}。

pose_list: 特指描述一组位姿数据的 pose 列表, 例如 {{1,2,3,4,5,6},{2,3,4,5,6,7}}。

joints_list: 特指描述一组关节相关数据的 joints 列表, 例如 {{1,2,3,4,5,6},{2,3,4,5,6,7}}。

pose_speed: 特指描述末端速度的双精度浮点数, 单位 mm/s、m/s, 例如 2.5。

pose_acc: 特指描述末端加速度的双精度浮点数, 单位 mm/s²、m/s², 例如 2.5。

joint_speed: 特指描述关节速度的双精度浮点数, 单位 deg/s、deg/s, 例如 2.5。

joint_acc: 特指描述关节加速度的双精度浮点数, 单位 deg/s²、deg/s², 例如 2.5。

timer: 特指描述时间间隔的双精度浮点数列表, 单位 s, 例如 1。

1.2.2 系统常量

一个形如 42 的值被称作字面值常量。每个字面值常量都对应一种数据类型, 字面值常量的形式和值决定了它的数据类型和在脚本中的使用方式。系统支持的常量如下:

基本类型常量:

算数类型常量、true、false。

算数类型常量通常指的双精度实浮点数常量, 该类型常量可以参与关系运算、算数运算、逻辑运算, 并可以为基本类型变量进行赋值和初始化, 亦可作为函数的实参进行参数传递。

true 在系统中代指真。

false 在系统中代指假。

字符串常量:

字符串常量是字符的集合, 字符串常量可用于字符串变量的定义, 函数参数传递。

1.2.3 变量

变量在系统中通常用一个唯一的标识符来表示 (id)。并且变量需要在定义之后才可以使用。

基本类型变量

a=1.23 --变量赋值 上面定义了 a 的类型为双精度实浮点数, 初始化 a 的值为 1.23

b = true	-- b 布尔变量定义，初始化为 真
c = false	--c 布尔变量定义，初始化为 假

字符串类型变量

s1= "siasun"	--s1 变量定义 字符串类型 初始化为 "siasun"
s2="SiasunCobot"	--s2 变量定义 字符串类型 初始化为"SiasunCobot"
s2=s1	--s2 变量赋值，值修改为"siasun"

列表类型变量

a1={}	--a1 变量定义，空列表，长度为 0
a2={1.1,2.2,3.3,4.4}	--a2 变量定义，双精度实浮点数列表，长度为 4
a3={"a","b","c"}	--a3 变量定义，字符串列表，长度为 3
a4=a3	#a4 变量定义，a4 和 a3 指向同一个列表，修改 a4 的值，a3 同时被修改

可以通过索引操作具体的元素，索引下标从 1 开始，比如，

```
a2[1]=12
a3[2]="siasun"
```

1.3 表达式

表达式在系统中指的是有返回值的运算集合，是一个递归的定义。

在此用 exp 表示表达式，用 op 表示运算符。

1.3.1 算术运算表达式

系统支持 加 (+) 减 (-) 乘 (*) 除 (/) 四类算数运算，算数运算表达式返回算数类型常量。

1.3.2 关系运算表达式

系统支持 > < >= <= == ~= 类型的关系运算，关系运算表达式返回常量真或假。

1.3.3 逻辑运算表达式

系统支持 `and` 、 `or` 、 `not` 类型的逻辑运算。

`exp1 and exp2`, 若 `exp1` 为真 而且 `exp2` 为真, 那么表达式返回 `true`, 否则返回 `false`。

`exp1 or exp2`, 若 `exp1` 为真 或者 `exp2` 为真, 那么表达式返回 `true`, 否则返回 `false`。

`not exp1`, 若 `exp1` 为真, 那么表达式返回 `false`, 否则返回 `true`。

1.3.4 赋值表达式

变量名 (`id`) = `exp`

赋值表达式的左值是一个变量的表示符, 只有在变量已经定义过的情况下该表达式才是一个赋值表达式, 若变量没有定义, 那么就是一个变量定义语句。

1.3.5 函数调用表达式

函数名 (实参列表)

1.4 语句

1.4.1 循环语句 `while`

`while (exp)`

`do`

`exp`

`...`

`end`

1.4.2 函数定义

`function fun(参数列表)`

语句列表

`...`

`return exp`

```
end
```

例：function fun1(a,b,c)

```
...
```

```
return a+b+c
```

```
end
```

参数类型：以实际传入参数为准

返回值类型：以实际返回参数为准

函数可以多参数返回，以“,”分割 列如：

```
function fun1(a,b,c)
```

```
...
```

```
return a,b,c
```

```
end
```

1. 4. 3return 语句

```
return exp
```

return 语句只能用在函数内部，用来返回函数的运算结果。

return 语句返回的数据类型决定了函数的返回值。若函数中没有包含 return 语句，那么函数不返回任何值。

1. 4. 4条件控制语句

形式一：

```
if (exp)
```

```
then
```

```
...
```

```
elseif (exp)
```

```
then
```

```
    ...  
  
    else  
  
    ...  
  
end
```

形式二:

```
if (exp)  
  
then  
  
    ...  
  
end
```

形式三:

```
if (exp)  
  
then  
  
    ...  
  
else  
  
    ...  
  
end
```

同大多数语言一样，系统支持 if 条件控制语句，当 if / elseif 中的判定条件表达式为真时，就执行语句块中的内容。

注意：控制结构的条件表达式结果可以是任何值， false 和 nil 为假， true 和非 nil 为真。特别注意， 0 为 true:

```
if (0)  
  
then  
  
    print("0 is true")  
  
end
```

结果： 0 is true

1. 4. 5 goto 语句

```
goto Label

:: Label::

例：

a=1

::label:: print(“----goto----”)

if a<3 then

    goto label

end
```

1. 4. 6 break 语句

使用 break 语句终止循环。

```
例：

a=10

while( a<20 )

do

    print(“a=”,a)

    a=a+1

    if(a>15) then


        break

    end

end
```

1.5 系统函数说明

1.5.1 移动控制

<p>Op={number:time_or_dist_1,number:trig_io_1,boolean:trig_value_1, number:trig_time_1,number:trig_dist_1, string:trig_event_1,number:time_or_dist_2,number:trig_io_2,boolean:trig_value_2, number:trig_time_2,number:trig_dist_2, string:trig_event_2}</p>	
<p>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。</p> <p>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：对于 time_or_dist_2 为 1，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：对于 time_or_dist_2 为 2，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。</p>	
<div> 注意</div>	<ul style="list-style-type: none">Op 参数在所有移动脚本中均为可缺省参数，即当没有 Op 指令时，不会再运动过程中触发 IO

```
movej2( joints : axis, number : v, number : a, number : rad = 0, boolean : block =
true, Op, boolean : 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:

axis : 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 $(0, 1.25*PI]$, 单位(rad/s)。

a : 关节角加速度, 范围 $(0, \infty]$, 单位 (rad/s^2) 。

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。

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 : 是否使用自定义加速度, 默认为 true, 可缺省参数。

返回值:

无

示例:

```
movej2({0,0,1.57,0,-1.57,0},0.523,5.23,0)
```

```
movej_pose2(pose : p, number : v, number : a, number : rad, joints : qnear,
string : tool, string : wobj, boolean : block=true, Op, boolean : 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 $[-2*PI, 2*PI]$, 单位(rad)。

v : 关节角速度, 范围 $(0, 1.25*PI]$, 单位 (rad/s)。

a : 关节加速度, 范围 $(0, \infty]$, 单位 (rad/s^2) 。

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。

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 : 是否使用自定义加速度, 默认为 true, 可缺省参数。

返回值:

无

示例:

```
movej_pose2({0.49,0.14,0.44,-3.14,0,-1.57},0.5,5,0,{0,0,1.57,0,-1.57,0},"default","default")
```

```
movel( pose : p, number : v, number : a, number : rad, joints : qnear, string : tool, string : wobj, boolean : block=true,Op, boolean : 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 $[-2\pi, 2\pi]$, 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: 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。

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: 是否使用自定义加速度, 默认为 **true**, 可缺省参数。

返回值:

无

示例:

```
movel({0.49,0.14,0.35,-3.14,0,-1.57},0.5,5,0,{0,0.02,1.77,-0.22,-1.57,0},"default","default")
```



```
movec( pose : p1, pose : p2, number : v, number : a, number : rad, number: mode,
joints : qnear, string : tool, string : wobj, boolean : block=true,Op, boolean :
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，位置单位 m，姿态以 Rx、Ry、Rz 表示范围 $[-2\pi, 2\pi]$ ，单位(rad)。

p2 :圆弧运动结束点，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 : 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。

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。

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 : 是否使用自定义加速度，默认为 true，可缺省参数。

返回值:

无

示例:

```
movec({0.397,0.14,0.35,-3.14,0,-1.57},{0.397,0.04,0.35,-3.14,0,-1.57},0.5,
5,0,{-0.47,-0.26,2.02,-0.19,-1.57,-0.47},"default","default")
```

`move_circle (pose : p1, pose : p2, number : v, number : a, number : rad, number: mode, joints : qnear, string : tool, string : wobj, boolean : block=true, Op, boolean : 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，位置单位 m，姿态以 Rx、Ry、Rz 表示范围 $[-2\pi, 2\pi]$ ，单位(rad)。

p2 :圆周运动经过点，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 : 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 : 是否使用自定义加速度，默认为 true，可缺省参数。

返回值:

无

示例:

`move_circle({0.5,0.0114,0.35,-3.14,0,-1.57},{0.358,0.144,0.35,-3.14,0,-1.57},0.5,5,0,{-0.01,-0.3,2.05,-0.179,-1.57,-0.012},"default","default")`

tcp_move(pose:pose_offset, number : v, number : a, number : rad, string : tool, boolean : block=true, Op, boolean : 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。

v: 直线移动的速度, 范围(0, 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, ∞], 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。

tool: 设置使用的工具的名称, 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: 是否使用自定义加速度, 默认为 true, 可缺省参数。

返回值:

无

示例:

```
tcp_move({0,0,0.1,0,0,0},0.5,5,0,"default")
```

tcp_move_2p (pose:p1, pose:p2 , number : v, number : a, number : rad, string : tool, string : wobj,boolean : block=true,Op, boolean : 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, 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, ∞], 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。

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: 是否使用自定义加速度, 默认为 true, 可缺省参数。

返回值:

无

示例:

```
tcp_move_2p({0.397,0.04,0.25,-3.14,0,-1.57},{0.397,-0.045,0.43,-3.14,0,-1.57},0.5,5,0,"default","default")
```

speedj (joints : axis_speed, number: a, number: t=-1)

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：

- axis_speed:每个关节的速度，范围(0, 1.25*PI]，单位(rad/s)。
- a: 主导轴的关节加速度，范围(0, ∞]，单位(rad/ s²)。
- t: 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。

返回值：

无

示例：

```
speedj({0.2,0.2,0,0,0,0},5,3000)
```

speedl (pose : position_speed, number: a, number: t=-1)

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：

- position_speed:末端速度向量，线速度范围(0, 5]，单位(m/s)，角速度范围(0, 1.25*PI]，单位(rad/s)。
- a:末端的线性加速度，范围(0, ∞]，单位(rad/ s²)。
- t: 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。

返回值：

无

示例：

```
speedl({0.2,0.2,0,0,0,0},5,3000)
```

speed_stop ()

函数说明：

停止 speedj 及 speedl 函数的运动。

参数说明：


无

返回值：

无

示例：

```
speed_stop()
```

<pre>spline(pose_list : p_list, number : v, number : a, string : tool, string : wobj, boolean : block=true,Op,number : rad, boolean : def_acc=true)</pre>	
<p>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。</p> <p>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, 5], 单位(m/s)。 a : 末端加速度, 范围(0, ∞], 单位(m/s²)。 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。 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。 def_acc : 是否使用自定义加速度, 默认为 true, 可缺省参数。</p> <p>返回值： 无</p> <p>示例： spline({{0.397,-0.11,0.5,-3.14,0,-1.57},{0.397,-0.11,0.43,-3.14,0,-1.57},{0.316,-0.11,0.43,-3.14,0,-1.57}},0.5,5,"default","default")</p>	
<div> 注意</div>	<ul style="list-style-type: none">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。

```
move_spiral(pose_list : p1, pose_list : p1, number : rev, number : len, number :
rad, number : mode, number : v, number : a, joints : qnear, string : tool, string :
wobj, boolean : block=true, Op, boolean : 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, 5]，单位(m/s)。
a：末端加速度，范围(0, ∞)，单位(m/s^2)。
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，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：是否使用自定义加速度，默认为 true，可缺省参数。

返回值:

无

示例:

```
move_spiral({{0.41,0.008,0.43,-3.14,0,-1.57},{},2,0.2,0,0.1,nil,"default","default"})
```

```
hand_teach()
```

函数说明:

当运行脚本程序时, 该函数调用并且触发末端牵引按钮时, 可以启用手动牵引功能。

参数说明:

无

返回值:

无

示例:

```
hand_teach()
```

replay(string : name, number : v, number : 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

示例：

replay("tmp", 100,0)

is_moving()

函数说明：

该函数判断机器人是否在运动。

参数说明：

无。

返回值：

True：机器人正在运动；

False：机器人没有运动。

示例：

is_moving()

set_blend_ahed (number :per)

函数说明：

该函数用于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：

无

示例：

set_blend_ahed (50)

1.5.2 网络通信

```
socket_open( string : ip, number : port, string : name ="socket1" , number :
timeout=3000)
```

函数说明:

打开一个以太网的通信链接,如果超时时间内没有创建成功,那么链接创建失败。

参数说明:

ip: 服务器端的 IP 地址,需要用“”括起来。

port: 服务器端的端口号。

name: 连接名称,可省略,默认为“socket1”。

tiomeout : 设置超时时间,单位(ms),默认 3000ms。

返回值:

true: 操作成功。

false: 操作失败。

示例:

```
socket_open("192.168.67.60",2003,"socket2",3000)
```

```
socket_write (msg_send,string : name="socket1", number : timeout=3000)
```

函数说明:

发送数据给已连接的服务器,函数根据不同数据类型自动判断。

参数说明:

msg_send : 需要发送的数据,接受以下类型:

<1> 字符串,需要用“”括起来

<2> number列表,列表中每个数据为32位float。

name: 连接名称,默认为“socket1”。

tiomeout : 设置超时时间,单位(ms),默认 3000ms。

返回值:

true: 操作成功。

false: 操作失败。

示例:

```
socket_write ("send message", "socket1",1000)
```

```
socket_write ({1,2,3.14},"socket1",1000)
```



```
socket_read_string(number:length,string:prefix="",string:subfix="", string:
name="socket1", number:timeout = 3000 )
```

函数说明:

从已连接的服务器接收一定长度的字符串。

参数说明:

length:需要接收的字符串长度，当设置后缀时，长度参数无效。

prefix:接收的字符串前缀，默认为空，当设置前缀后，从设置前缀开始读取，直至达到设置长度或者设置后缀。

subfix:接收的字符串的后缀，默认为空，当设置后缀后，从字符串开始位置读取，直至读取到设置后缀，当达到最大限制长度255时，返回255字节长度的字符串。

name: socket连接的名字,字符串格式。

timeout: 超时时间，单位ms。不填写时默认为3000ms。

返回值:

收到字符串信息，如果读取失败，返回空 string。

示例:

```
socket_read_string (10, " ", " ", "socket1",3000)
```

```
socket_read_str_num (string:name="socket1",number: timeout = 3000 )
```

函数说明:

从已连接的服务器接收一组字符串，解析成 num 数值，所有数据应当在括号内，数据之间通过“,” 隔开，数据格式为: (num1,num2,num3)

参数说明:

name: socket 连接的名字,字符串格式。

timeout: 超时时间，单位(ms)。不填写时默认为 3000ms。

返回值:

收到的 number 型列表，如果读取失败返回空列表。

示例:

```
list=socket_read_str_num("socket",1000)
```

```
socket_read_num(number:count, name="socket1",timeout = 3000 )
```

函数说明:

从已连接的服务器接收一组浮点数。

参数说明:

count:需要接收的数据个数，每个数据为单精度浮点数。

name: socket 连接的名字，字符串格式。

timeout: 超时时间，单位(ms)。不填写时默认为 3000ms。

返回值:

收到的 number 列表，如果读取失败返回空列表。

示例:

```
list= socket_read_num (10, "socket",1000)
```

```
socket_close( string : name = "socket1" )
```

函数说明:

关闭和服务器端的以太网通信链接。

参数说明:

name: 连接名称，默认为“socket1”。

返回值:

无

示例:

```
list= socket_close ("socket")
```

```
set_data_frequence( number : freq)
```

函数说明:

设置 2001 端口数据的发送频率。

参数说明:

freq: 发送频率，($1 \leq \text{freq} \leq 100$)。

返回值:

无

示例:

```
list= set_data_frequence (10)
```

```
socket_write_byte_list
```

```
(table:list,table:head,table:tail,name="socket1",timeout=3000)
```

函数说明:

发送 byte 类型数据给已连接的服务器。

参数说明:

list: 发送的 byte 类型数据列表

head:数据头。该数据可缺省。

tail:数据尾。该数据可缺省。

name: socket 连接的名字,字符串格式，默认为 socket1。

timeout: 超时时间，单位(ms)。不填写时默认为 3000ms。

返回值:

true: 操作成功。

false: 操作失败。

示例:

```
socket_write_byte_list ({255,254,1,2,3,253,252},"socket1",3000)
```

```
socket_write_byte_list ({1,2,3},{255,254},{253,252}"socket1",1000)
```

socket_read_byte_list

(table:head, int:len(table:tail),name="socket1",timeout=3000)

函数说明:

从已连接的服务器接收一组 byte 类型数据。

参数说明:

head: 数据头。该数据可缺省，如果缺省，则数据长度是必要参数；如果使用数据头，则数据长度或数据尾，可任选一种。

len:数据长度。可以用数据头与数据尾代替。

tail:数据尾。该数据可缺省。

name: socket 连接的名字,字符串格式，默认为 socket1。

timeout: 超时时间，单位(ms)。不填写时默认为 3000ms。

返回值:

收到的 byte_list 列表，如果读取失败返回空列表。

示例:

```
list=socket_read_byte_list(10,"socket1",3000);
```

```
list=socket_read_byte_list({255,254},10,"socket1",3000);
```

```
list=socket_read_byte_list({255,254},{253,252},"socket1",3000);
```

1.5.3 can/485 总线

`read_raw_data_485 (number:len, number:time=3000)`

函数说明：

485 端口在设置的时间内读取长度为len的字节数据。

参数说明：

len:需要读取的长度。

time:等待时间，默认 3000ms，单位(ms)。

返回值：

number_list 读取到的数据，在规定的时间内，接收到指定长度的数据会立即返回，未接收到指定长度的数据时，等待结束后，返回收到的数据。

示例：

```
list= read_raw_data_485 (10,3000)
```

`read_raw_data_485 (number_list:head, number_list:tail, number:time=3000)`

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：

head:需要匹配的头数据。

tail:需要匹配的尾数据。

time:等待时间，默认 3000ms，单位(ms)。

返回值：

number_list 读取到的数据，在规定的时间内，接收到指定长度的数据会立即返回，未接收到指定长度的数据时，等待结束后，返回收到的数据。

示例：

```
list= read_raw_data_485 ({255,1},{1,255})
```

`read_raw_data_485 (number_list:head, number:len, number:time=3000)`

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：

head:需要匹配的头数据。

len:需要读取的长度。

time:等待时间，默认 3000ms，单位(ms)。

返回值：

number_list 读取到的数据，在规定的时间内，接收到指定长度的数据会立即返回，未接收到指定长度的数据时，等待结束后，返回收到的数据。

示例：

```
list= read_raw_data_485 ({255,1},10)
```

read_data_485 ()**函数说明:**

配方 485 读数据，按照配方将输入的数据转换成配方后的数据（自定义配方情况下使用）。

参数说明:

无

返回值:

number_list 读取到的数据，未读到数据返回空列表。

示例:

```
list= read_data_485 ()
```

write_raw_data_485 (number_list:value)**函数说明:**

485 写原生数据，将列表 value 中的数据写入 485 端口。

参数说明:

value:需要写入的数据列表。

返回值:

true:成功。

false:失败。

示例:

```
write_raw_data_485 ({1,2,3})
```

write_raw_data_485 (number_list:value,number_list: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:

```
write_raw_data_485 ({1,2,3},{255,255})
```

write_raw_data_485 (number_list:value,number_list:head, number_list: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:

write_raw_data_485 ({1,2,3},{255,255},{255,255})

write_data_485 (number_list:value)

函数说明:

配方 485 写数据，按照配方的输出配置将转换后的流数据写入 485 端口（自定义配方情况）。

参数说明:

value:需要写入的数据列表。

返回值:

true:成功。

false:失败。

示例:

write_data_485 ({255,255,1,1,1,238,238})

tool_read_raw_data_485 (number: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:

len:需要读取的长度。

返回值:

number_list 读取到的数据，未读到数据返回空列表。

示例:

list= tool_read_raw_data_485 (10)

<p>tool_read_raw_data_485 (number_list:head, number_list:tail)</p> <p>函数说明： 末端 485 匹配头 head 和尾 tail 读取到一帧匹配的数据。</p> <p>参数说明： head:需要匹配的头数据。 tail:需要匹配的尾数据。</p> <p>返回值： number_list 读取到的数据，未读到数据返回空列表。</p> <p>示例： list= tool_read_raw_data_48 ({255,1},{1,255})</p>
<p>tool_read_raw_data_485 (number_list:head, number:len)</p> <p>函数说明： 末端 485 匹配头 head 后读取到长度为 len 的一帧数据。</p> <p>参数说明： head:需要匹配的头数据。 len:需要读取的长度。</p> <p>返回值： number_list 读取到的数据，未读到数据返回空列表。</p> <p>示例： list= tool_read_raw_data_485 ({255,1},10)</p>
<p>tool_read_data_485 ()</p> <p>函数说明： 末端配方 485 读数据，按照配方将输入的数据转换成配方后的数据（自定义配方情况下使用）。</p> <p>参数说明： 无</p> <p>返回值： number_list 读取到的数据，未读到数据返回空列表。</p> <p>示例： list= tool_read_data_485 ()</p>

<p><code>tool_write_raw_data_485 (number_list:value)</code></p> <p>函数说明: 末端 485 写原生数据，将表 value 中的数据写入 485 端口。</p> <p>参数说明: value:需要写入的数据列表。</p> <p>返回值: true:成功。 false:失败。</p> <p>示例: <code>tool_write_raw_data_485 ({1,2,3})</code></p>
<p><code>tool_write_raw_data_485 (number_list:value,number_list:head)</code></p> <p>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。</p> <p>参数说明: value:需要写入的数据列表。 head:需要添加的头。</p> <p>返回值: true:成功。 false:失败</p> <p>示例: <code>tool_write_raw_data_485 ({1,2,3},{255,255})</code></p>
<p><code>tool_write_raw_data_485 (number_list:value,number_list:head, number_list:tail)</code></p> <p>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。</p> <p>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</p> <p>返回值: true:成功。 false:失败。</p> <p>示例: <code>tool_write_raw_data_485 ({1,2,3},{255,255},{255,255})</code></p>

`tool_write_data_485 (number_list:value)`

函数说明：

末端配方 485 写数据，按照配方的输出配置将转换后的流数据写入 485 端口（自定义配方情况）。

参数说明：

value:需要写入的数据列表。

返回值：

true:成功。

false:失败。

示例：

`tool_write_data_485 ({255,255,0,0,0,255,255})`

`read_raw_data_can ()`

函数说明：

读取一帧 can 的字节数据。

参数说明：

无

返回值：

number_list 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。

示例：

`list = read_raw_data_can ()`

`read_raw_data_can (number:id)`

函数说明：

根据 id 读取一帧 can 的字节数据。

参数说明：

id:需要读取的帧 id。

返回值：

number_list 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 id。

示例：

`list = read_raw_data_can (1)`

`read_data_can()`

函数说明：

配方 can 读数据，按照配方将输入的数据转换成配方后的数据（自定义配方情况下）。

参数说明：

无

返回值：

`number_list` 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 id。

示例：

```
list = read_data_can()
```

`write_raw_data_can(number:id, number_list:value)`

函数说明：

can 写帧为 id，数据为 value 的原生数据。

参数说明：

id:数据帧的 id。

value:要发送的数据列表。

返回值：

true:成功。

false:失败。

示例：

```
write_raw_data_can(1,{1,2,3})
```

`write_data_can(number: id, number_list: value)`

函数说明：

can 写帧为 id，数据为 value 的配方数据。

参数说明：

id:数据帧的 id。

value:要发送的数据列表。

返回值：

true:成功。

false:失败。

示例：

```
write_data_can(1,{1,2,3})
```

1.5.4 系统函数及外设



注意

- 若范围参数越界，函数将返回 false 或 0

sleep(number : time)

函数说明：

该函数会让程序暂停一定时间。

参数说明：

time：需要暂停的时间，单位(ms)。

返回值：

无

示例：

sleep (1000)

set_digital_output_mode (number : portnum, number: type, number: freq, number: 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：

portnum：控制柜上的 IO 输出口序号，范围从 1-16。

type：0 为高电平，1 为低电平。

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，需要主动发送输出信号才能生效。设置成脉冲信号后，需要发送一个高电平的启动信号。

返回值：

无

示例：

set_digital_out_mode (1,1,100,50)

set_standard_digital_out (1,true)



注意

- 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 (number : portnum, boolean: val)`

函数说明：

该函数可控制控制柜上的通用 IO 输出口的高低电平。

参数说明：

portnum：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。

返回值：

无

示例：

`set_standard_digital_out (1,true)`

`get_standard_digital_out (number : portnum)`

函数说明：

该函数可读取控制柜上的通用 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：

portnum：控制柜上的 IO 输出口序号，范围从 1-16。

返回值：

boolean，返回 true 为高电平，false 为低电平。

示例：

`value = get_standard_digital_out (1)`

`get_standard_digital_in (number : portnum)`

函数说明：

该函数可读取控制柜上的通用 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：

portnum：控制柜上的 IO 输入口序号，范围从 1-16。

返回值：

boolean，返回 true 为高电平，false 为低电平。

示例：

`value = get_standard_digital_in (1)`

`set_tool_digital_out (number : portnum, boolean : val)`

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：

portnum：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。

返回值：

无

示例：

`set_tool_digital_out (1, true)`

`get_tool_digital_in (number : portnum)`

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：

portnum：机械臂末端的 IO 输入口序号，范围从 1-2。

返回值：

boolean，返回 true 为高电平，false 为低电平。

示例：

`value = get_tool_digital_in (1)`

`get_tool_digital_out (number : portnum)`

函数说明：

该函数可读取机械臂末端的 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：

portnum：机械臂末端的 IO 输出口序号，范围从 1-2。

返回值：

boolean，返回 true 为高电平，false 为低电平。

示例：

`value = get_tool_digital_out (1)`

`get_function_digital_in (number : 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：

boolean，返回 true 为高电平，false 为低电平。

示例：

```
value = get_function_digital_in (1)
```

`get_function_digital_out (number : 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：

boolean，返回 true 为高电平，false 为低电平。

示例：

```
value = get_function_digital_out (1)
```

`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。

示例：

```
value = get_standard_analog_voltage_in (1)
```

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

示例：

`value = set_standard_analog_voltage_out (1, 1.5, true)`

`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。

示例：

`value = get_tool_analog_voltage_in (1)`

`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序号，范围从 1-4。

返回值：

对应通道的模拟电流值

示例：

`value = get_standard_analog_current_in (1)`

`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；

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。

示例：

`value = set_standard_analog_current_out (1, 1.5,true)`

`get_standard_analog_current_out (int : num)`

函数说明：

该函数可获取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；

返回值：

对应通道的模拟电流值。

示例：

`value = get_standard_analog_current_out (1)`

`get_standard_analog_voltage_out (int : num)`

函数说明：

该函数可获取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；

返回值：

对应通道的模拟电压值。

示例：

`value = get_standard_analog_voltage_out (1)`

`write_reg(number: num, number: type, val)`

函数说明：

该函数可修改内部寄存器的值。

参数说明：

num: 内部寄存器序号。

type: 修改的寄存器类型 1 为 bool 寄存器, 2 为 word 寄存器, 3 为 float 寄存器。

val: 根据 type 类型确定 val 类型。

当 type 为 1 时, val 类型为 boolean, true 表示真, false 表示假, num 范围为 1-64

当 type 为 2 时, val 类型为 number, 并且为整数, 范围 0-65535, num 范围为 1-32

当 type 为 3 时, val 类型为 number, num 范围为 1-32

参数错误时函数不改变内部寄存器数值。

返回值：

无

示例：

`write_reg(5, 1,true)`

`read_reg (number: num, number: type, number: in_out)`

函数说明：

该函数可读取内部寄存器的值。

参数说明：

num: 内部寄存器序号。

type: 寄存器类型 1 为 bool 寄存器, 2 为 word 寄存器, 3 为 float 寄存器。

当 type 为 1 时, num 范围为 1-64。

当 type 为 2 时, num 范围为 1-32。

当 type 为 3 时, num 范围为 1-32。

参数错误时函数不改变内部寄存器数值。

in_out: 为 0 时代表读取输入寄存器, 1 代表读取输出寄存器。

返回值：

当 type 为 1 时, 返回值类型为 boolean, true 表示真, false 表示假。

当 type 为 2 时, 返回值类型为 number, 并且为整数, 范围为 0-65535。

当 type 为 3 时, 返回值类型为 number。

示例：

`ret=read_reg(10,1,1)`

`get_function_reg_in (number : portnum)`

函数说明：

该函数可读取功能输入寄存器值。

参数说明：

portnum：功能输入寄存器序号，范围从 1-16。

返回值：

boolean，返回 true 为真，false 为假。

示例：

ret= get_function_reg_in (1)

`get_function_reg_out (number : portnum)`

函数说明：

该函数可读取功能输出寄存器值。

参数说明：

portnum：功能输出寄存器序号，范围从 1-16。

返回值：

boolean，返回 true 为真，false 为假。

示例：

ret= get_function_reg_out (1)

`set_wobj_offset (boolean : val , number_list: 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：

val : true 为启用偏移量，false 为取消偏移量。

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

返回值：

无

示例：

ret= set_wobj_offset (true, {0.1,0,0,0,0,0})

```
set_tool_data(string: name, number_list: tool_offset, number_list: load,
number_list 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。

参数说明：

name:工具的名字，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)。

load: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，可缺省，缺省时默认为 0。

返回值：

无

示例：

```
ret= set_tool_data ("default", {0.1,0,0,0,0,0}, {5,0,0.05,0},{0,0,0,0,0,0})
```

```
set_load_data (number_list: 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

示例：

```
ret= set_load_data ({5,0,0.05,0})
```

`cal_ikine(pose : p, joints : q_near, number_list:tcp, number_list: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。

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。

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wobj。

返回值：

返回的关节位置列表 {q1,q2,q3,q4,q5,q6}

示例：

```
ret= cal_ikine({0.492,0.140,0.442,-3.14,0,-1.57},{},{},{})
```

`cal_fkine(joints:axis, number_list:tcp, number_list: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：

axis :需要计算正解的关节角，单位(rad)。

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 TCP 值。

wobj:工件坐标系相对于世界坐标系的偏移量 {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}

示例：

```
ret= cal_fkine({0,0,1.57,0,-1.57,0},{},{})
```

`get_tcp_pose()`

函数说明：

该函数可获取当前状态下机械臂 TCP 在基坐标系下的位姿。

参数说明：无

返回值：

返回末端位置 pose。

示例：

```
ret= get_tcp_pose ()
```

`get_tcp_pose_coord(string : tool, 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：

tool: 工具坐标系名称，默认为当前使用的坐标系。

wobj: 工件坐标系名称，默认为当前使用的坐标系。

返回值：

末端法兰的位姿，单位(m, rad)。

示例：

```
ret= get_tcp_pose_coord ("tool_1", "wobj_1")
```

`get_tcp_speed()`

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：无

返回值：

末端速度列表，单位 m/s, rad/s。

示例：

```
ret= get_tcp_speed ()
```

`get_tcp_acceleration()`

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：无

返回值：

末端加速度列表，单位(m/s², rad/s²)。

示例：

```
ret= get_tcp_acceleration ()
```

`get_tcp_force()`

函数说明：

该函数可获取当前机械臂工具末端的力矩信息。

参数说明：无

返回值：

返回末端力矩信息，{Fx,Fy,Fz,Mx,My,Mz}, 单位 N、N.m。

示例：

```
ret= get_tcp_force ()
```

get_tcp_force_tool(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：

tool: 工具坐标系名称，默认为当前使用的坐标系。

返回值：

返回工具坐标系下的力矩信息，{Fx,Fy,Fz,Mx,My,Mz}, 单位(N、N.m)。

示例：

```
ret= get_tcp_force_tool ("tool_1")
```

get_tcp_offset()

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： 无

返回值：

{x_off, y_off,z_off,rx,ry,rz} 返回 TCP 偏移量信息，单位(m, rad)。

示例：

```
ret= get_tcp_offset ()
```

get_tool_load()

函数说明：

该函数可获取当前设置工具的负载质量、质心位置和惯性张量。

参数说明： 无

返回值：

质量单位 kg，质心位置单位(m)，{mass,x_cog,y_cog,z_cog,t_1,t_2,t_3,t_4,t_5,t_6,}。

示例：

```
ret= get_tcp_load ()
```

get_wobj()

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： 无

返回值：

{x, y, z, rx, ry, rz} 工件坐标系相对于世界坐标系的偏移量，单位(m, rad)。

示例：

```
ret= get_wobj()
```

get_actual_joints_position ()

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: 无

返回值:

返回 1-6 轴关节角度列表, 单位(rad)。

示例:

```
ret= get_actual_joints_position()
```

get_target_joints_position ()

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: 无

返回值:

返回 1-6 轴关节角度列表, 单位(rad)。

示例:

```
ret= get_target_joints_position()
```

get_actual_joints_speed ()

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: 无

返回值:

返回 1-6 轴关节速度列表, 单位(rad/s)。

示例:

```
ret= get_actual_joints_speed ()
```

get_target_joints_speed ()

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速度。

参数说明: 无

返回值:

返回 1-6 轴关节速度列表, 单位(rad/s)。

示例:

```
ret= get_target_joints_speed()
```

get_actual_joints_acceleration ()**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: 无**返回值:**

返回 1-6 轴关节加速度列表, 单位 (rad/s^2)。

示例:

```
ret= get_actual_joints_acceleration()
```

get_target_joints_acceleration ()**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加速度。

参数说明: 无**返回值:**

返回 1-6 轴关节加速度列表, 单位 (rad/s^2)。

示例:

```
ret= get_target_joints_acceleration()
```

get_actual_joints_torque ()**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: 无**返回值:**

返回 1-6 轴关节力矩列表, 单位 (N.m)。

示例:

```
ret= get_actual_joints_torque ()
```

get_target_joints_torque ()**函数说明:**

该函数可获取当前状态下机械臂各关节的目标力矩。

参数说明: 无**返回值:**

返回 1-6 轴关节力矩列表, 单位 (N.m)。

示例:

```
ret= get_target_joints_torque ()
```


`get_flange_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: 无

返回值:

末端法兰位置 pose。

示例:

```
ret= get_flange_pose ()
```

`get_flange_speed()`

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: 无

返回值:

末端法兰速度列表，单位 (m/s, rad/s)。

示例:

```
ret= get_flange_speed ()
```

`get_flange_acceleration()`

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: 无

返回值:

末端法兰加速度列表，单位 (m/ s², rad/ s²)。

示例:

```
ret= get_flange_acceleration ()
```

`sub_program(string : program_name, string: tree_name)`

函数说明:

调用子程序。

参数说明:

program_name : 子程序名。

tree_name:子程序树节点名称，显示使用。

返回值:

无

示例:

```
sub_program("program1.jspf", " ")
```

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:

无

示例:

start_record_track ("track",0,"default","default")

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

示例:

stop_record_track ()

collision_detect (number: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:

无

示例:

collision_detect (1)

set_value (string:name,var:data)

函数说明：

设置系统变量值。

参数说明：

name:系统变量名称

var:设置的系统变量值，根据对应的系统变量类型确定其类型

当系统变量类型为:

boolean,data 类型为 boolean;

number, data 类型为 number;

string, data 类型为 string;

number_list, data 类型为 number_list;

pose, data 类型为 number_list;

joints, data 类型为 number_list;

返回值：

无

示例：

```
set_value ("g_data",true)
```

get_value (string:name)

函数说明：

获取系统变量值。

参数说明：

name:系统变量名称

返回值：

ret:根据对应的系统变量类型确定其返回类型,当系统变量类型为:

boolean, ret 类型为 boolean;

number, ret 类型为 number;

string, ret 类型为 string;

number_list, ret 类型为 number_list;

pose, ret 类型为 number_list;

joints, ret 类型为 number_list;

示例：

```
ret = get_value ("g_data")
```

timer_start ()

函数说明：

开始/重置计时器。

参数说明：

无

返回值：

无

示例：

timer_start ()

timer_end (string : timer_list)

函数说明：

计算时间间隔。

参数说明：

timer_list: 存放时间间隔的 timer 类型系统变量的名称。

返回值：

无

示例：

timer_end (timer_1)

1.5.5 调试相关

log(string: describe, val)

函数说明:

该函数可在程序中插入 log 日志，记录运行问题。

参数说明:

describe : 日志描述。

val: 变量值, 可以是 number, string, list 等任意类型。

返回值:

无

示例:

```
log("this is log",{1,2,3,4})
```

message(string: describe)

函数说明:

该函数可在程序运行过程中产生弹窗，并暂停当前程序。

参数说明:

describe : 弹窗描述。

返回值:

无

示例:

```
message ("this is message")
```

1.5.6 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 `number` 类型。

参数说明：

`signal_name`: modbus 节点名。

返回值：

`number`

示例：

```
val=modbus_read("mbus1")
```

`modbus_write(string : signal_name, number : val)`

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。

`val`: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：

无

示例：

```
modbus_write("mbus1", 1)
```

`modbus_set_frequency(string : signal_name, number: 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(number : slave_num, string : name, number : len,  
number_list : 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(number : slave_num, string : name, number : len,  
number_list : 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})
```

1.5.7 数学运算函数

abs (number: number)

函数说明：

计算绝对值。

参数说明：

number: number 数据类型。

返回值：

number 类型。

示例：

```
ret = abs (2)
```

cos(number: number)

函数说明：

计算余弦值。

参数说明：

number: number 数据类型，单位(rad)。

返回值：

number 类型。

示例：

```
ret = cos (1.57)
```

acos(number: number)

函数说明：

计算反余弦值。

参数说明：

number: number 数据类型，余弦值，取值范围[-1,1]。

返回值：

number 类型，单位(rad)。

示例：

```
ret = acos (0.5)
```

sin(number: number)

函数说明：

计算正弦值。

参数说明：

number: number 数据类型，单位(rad)。

返回值：

number 类型。

示例：

```
ret = sin (1.57)
```


asin(number: number)

函数说明：

计算反正弦值。

参数说明：

number: number 数据类型，正弦值，取值范围[-1,1]。

返回值：

number 类型，单位(rad)。

示例：

```
ret = asin (0.5)
```

tan(number: number)

函数说明：

计算正切值。

参数说明：

number: number 数据类型，单位(rad)。

返回值：

number 类型。

示例：

```
ret = tan (1.57)
```

atan(number: number)

函数说明：

计算反正切值。

参数说明：

number: number 数据类型，正切值。

返回值：

number 类型，范围(-pi/2,+pi/2)，单位(rad)。

示例：

```
ret = atan (0.5)
```

atan2(number: number1, number: number2)

函数说明：

计算 number1/number2 的反正切值，根据 number1、number2 的符号，确定返回值所在的象限。

参数说明：

number1: number 数据类型。

number2: number 数据类型。

返回值：

number 类型，范围（-pi,pi），单位(rad)。

示例：

```
ret = atan2 (1,2)
```

pow(number: base,number: exponent)

函数说明：

幂运算。

参数说明：

base: 底数。

exponent: 指数。

注: 如果 base 为负值并且 exponent 不是一个整数值, 则发生定义域错误。

当 base 为 0 且 exponent 小于 0 时, 会发生定义域错误。

返回值：

number 类型。

示例：

```
ret = pow (10,2)
```

sqrt(number: number)

函数说明：

计算平方根。

参数说明：

number: number 数据类型, 取值大于等于 0。

返回值：

number 类型。

示例：

```
ret = sqrt (10)
```

deg2rad(number: number)

函数说明：

角度值转换成弧度值。

参数说明：

number: number 数据类型, 单位度。

返回值：

number 类型, 单位(rad)。

示例：

```
ret = deg2rad (90)
```

rad2deg(number: number)

函数说明：

弧度值转换成角度值。

参数说明：

number: number 数据类型, 单位(rad)。

返回值：

number 类型, 单位度。

示例：

```
ret = rad2deg (1.57)
```

pose_trans (pose: p1,pose: p2)

函数说明：

坐标变换函数，位姿 p1，经过 p2 变换得到一个新的位姿，返回的位姿 $p3=p1*p2$ 。

参数说明：

p1:pose 类型，姿态信息按照 $Rz*Ry*Rx$ 方式计算旋转矩阵，位置单位 m，姿态单位(rad)。

p2:pose 类型，姿态信息按照 $Rz*Ry*Rx$ 方式计算旋转矩阵，位置单位 m，姿态单位(rad)。

返回值：

pose 类型，经过坐标变换后的位姿，位置单位 m，姿态单位(rad)。

示例：

```
ret = pose_trans ({492,140.5,442,-3.14,0,-1.57}, {231.4,140.5,582.3,-3.14,0,-1.57} )
```

pose_inv (pose: p1)

函数说明：

坐标逆变换函数，求解一个变换的逆变换。

参数说明：

p1:pose 类型，姿态信息按照 $Rz*Ry*Rx$ 方式计算旋转矩阵，位置单位 m，姿态单位(rad)。

返回值：

pose 类型，逆变换的解。姿态信息按照 $Rz*Ry*Rx$ 方式计算旋转矩阵，位置单位 m，姿态单位(rad)。

示例：

```
ret = pose_inv ({492,140.5,442,-3.14,0,-1.57} )
```

pose_distance(pose: p1, pose: p2)

函数说明：

计算两点之间的空间距离，姿态不参与计算。

参数说明：

p1: 要计算的点 1。

p2: 要计算的点 2。

返回值：

number: 两点间的距离。

示例：

```
ret = pose_distance ({492,140.5,442,-3.14,0,-1.57}, {231.4,140.5,582.3,-3.14,0,-1.57} )
```

pose_offset(pose: p1,pose: p2)

函数说明：

计算两个位姿之间的偏移量。

参数说明：

p1: 要计算的点 1。

p2: 要计算的点 2。

返回值：

两个位姿之间的偏移量。

示例：

```
ret = pose_offset ({492,140.5,442,-3.14,0,-1.57}, {231.4,140.5,582.3,-3.14,0,-1.57})
```

num2str (number: num, number: precision)

函数说明：

将数值按照指定的精度转换为字符串。比如 num2str(1.23456, 3)转换成“1.235”，num2str(1.23)转换成“1.230000”。

参数说明：

num: 要转换的数值。

precision: 保留的精度，可缺省，默认为 6。

返回值：

string

示例：

```
ret = num2str (1.234, 2)
```

str2num (string:s)

函数说明：

将字符串转换成数值，比如“1.23”转换成 1.23。

参数说明：

s: 需要转译的字符串。

返回值：

number 类型，转译后的数。

示例：

```
ret = str2num (“1.234”)
```

list2str (number_list:a)

函数说明：

将 number 类型列表转换为固定格式字符串，字符串以 “(” 开头，“)” 结尾，中间的数据之间使用 “,” 分隔。比如：列表{1.23,1.57,2.3}会转换成字符串“(1.23,1.57,2.3)”。

参数说明：

a : number 类型的列表。

返回值：

string

示例：

```
ret = list2str ({1.2,3.4,5.6})
```

str2list(string:s)

函数说明：

将字符串转换为列表，字符串以 “(” 开头，“)” 结尾，中间的数据之间使用“,”分隔。比如：字符串“(1.23,1.57,2.3)” 会转换成列表{1.23,1.57,2.3}。

参数说明：

s : 需要转换的格式化字符串。

返回值：

number_list, 如果转换错误，则返回空列表。

示例：

```
ret = str2list (“(1.2,3.4,5.6)”)
```

get_pose_trans(pose:p)

函数说明：

获取机器人位姿中关于位置的偏移量。

参数说明：

p : pose 类型，机器人位姿数据，单位：m、rad。

返回值：

number_list, 返回 pose 的 3 维位置偏移量信息，单位：m。

示例：

```
ret = get_pose_trans ({1,2,3,4,5,6})
```

期望结果:ret=(1,2,3)

get_pose_rpy(pose:p)

函数说明：

获取机器人位姿中关于姿态的偏移量。

参数说明：

p: pose 类型，机器人位姿数据，单位：m、rad。

返回值：

number_list, 返回 pose 的 3 维姿态偏移量信息，单位：rad。

示例：

```
ret = get_pose_rpy({1,2,3,4,5,6})
```

期望结果:ret=(4,5,6)

```
get_number_list_norm(number_list : nl)
```

函数说明：

获取 nl 中所有数据的模长，方式为取 nl 所有元素的平方和。

参数说明：

nl : number_list 类型，输入参数。

返回值：

number，输入参数的模长。

示例：

```
ret = get_number_list_norm ({1,2,3,4,5,6})
```

```
normalize_number_list(number_list:nl)
```

函数说明：

将输入的 nl 归一化并返回输出，方式为 nl 的所有元素除以其模长。

参数说明：

nl : number_list 类型，输入参数。

返回值：

number_list，归一化后的数据。

示例：

```
ret = normalize_number_list ({1,2,3,4,5,6})
```

```
number_list_cross(number_list : nl1, number_list : nl2)
```

函数说明：

计算两个维度为 3 的 number_list 的叉乘，计算方式为 $nl1 \times nl2$ 。

参数说明：

nl1 : number_list 类型，输入参数，维度为 3。

nl2 : number_list 类型，输入参数，维度为 3。

返回值：

number_list，叉乘结果。

示例：

```
ret = number_list_cross ({1,2,3},{4,5,6})
```

```
number_list_cross(list : nl)
```

函数说明：

计算的 nl 中所有 number_list 的叉乘。

参数说明：

nl : list 类型，{number_list: nl1, number_list: nl2, ...}，所有 number_list 的维度必须大于等于 3 且相等，且 number_list 的参数个数必须为 number_list 的维度-1。

返回值：

number_list，叉乘结果，维度与 nl 的元素的维度相同。

示例：

```
ret = number_list_cross({{1,2,3},{4,5,6}})
```

`number_list_dot(number_list : nl1, number_list : nl2)`

函数说明：

计算 nl1 和 nl2 的点积。

参数说明：

nl1: number_list 类型，输入参数。

nl2: number_list 类型，维度与 nl1 相同。

返回值：

number, nl1·nl2 的值。

示例：

```
ret = number_list_dot ({1,2,3,4,5,6},{1,2,3,4,5,6})
```

`rpy_to_axial_angle(number_list : rpy)`

函数说明：

计算输入 rpy 所对应的轴角，rpy 默认为动态 ZYX。

参数说明：

rpy : number_list 类型，默认参考动态 ZYX，单位 rad，必须为 3 维。

返回值：

number_list, rpy 所对应的轴角，维度为 4 维，前三个值为所对应的旋转轴矢量，第 4 个值对应参考旋转轴矢量的旋转角度，单位：rad。

示例：

```
ret = rpy_to_axial_angle({1,2,3})
```

`rpy_to_rot(number_list : rpy)`

函数说明：

计算输入 rpy 所对应的旋转矩阵，rpy 默认为动态 ZYX。

参数说明：

rpy : number_list 类型，默认参考动态 ZYX，单位 rad，必须为 3 维。

返回值：

list, {number_list, number_list, number_list}, 按 RX、RY、RZ 的顺序返回 rpy 所对应的 3 组正交基，组成对应的旋转矩阵。

示例：

```
ret = rpy_to_rot ({1,2,3})
```

`rot_to_rpy(list : rot)`

函数说明：

计算输入旋转矩阵 rot 所对应 rpy，rpy 默认为动态 ZYX。

参数说明：

rpy : list 类型，{number_list, number_list, number_list}, 按 RX、RY、RZ 的顺序输入三组正交基，numberl_list 必须为 3 维。

返回值：

number_list, RX、RY、RZ 值。

示例：

```
ret = rot_ro_rpy ({1,2,3},{4,5,6},{7,8,9})
```

1.5.8 字符串相关函数

<code>str_cat(string : str1, string : str2)</code>
<p>函数说明： 拼接两个字符串。</p> <p>参数说明： str1：要拼接的左侧字符串。 str2：要拼接的右侧字符串。</p> <p>返回值： 拼接后的字符串。</p> <p>示例： ret = str_cat (“a”, ”b”)</p>
<code>str_cmp(string : str1, string : str2)</code>
<p>函数说明： 比较两个字符串是否相等，系统支持的字符串最大长度为 255。</p> <p>参数说明： str1：要比较的字符串 1。 str2：要比较的字符串 2。</p> <p>返回值： false：不相等。 true：相等。</p> <p>示例： ret = str_cmp (“a”, ”b”)</p>
<code>str_del(string : str, number: index, number: num)</code>
<p>函数说明： 删除字符串中的一段。</p> <p>参数说明： str：要进行操作的字符串。 index：表示从第几位开始删除，从 1 开始计数。 num：表示删除几位。</p> <p>返回值： 删除后的字符串。</p> <p>示例： ret = str_del (“abcde”, 2, 2)</p>


```
str_insert(string : str1, number: index, number:num, string : str2)
```

函数说明:

在一个字符串中插入另一个字符串。

参数说明:

str1 : 进行操作的基准字符串。

index: 从基准字符串哪一位开始插入，从 1 开始计数。

num: 插入多少位。

str2 : 插入的字符串。

返回值:

操作后的字符串。

示例:

```
ret = str_insert ("abcde", 2, "fff")
```

```
str_substr(string: str, number: index, number: num)
```

函数说明:

取字符串的子集。

参数说明:

str : 要操作的字符串。

index: 从哪一位开始取，从 1 开始计数。

num: 取多少位。

返回值:

操作后的字符串。

示例:

```
ret = str_substr ("abcde", 2, 2)
```

```
str_len(string : str)
```

函数说明:

字符串长度。

参数说明:

str1 : 要计算长度的字符串。

返回值:

number, 字符串长度。

示例:

```
ret = str_len ("abcde")
```

```
str_find(string : str1, string : str2)
```

函数说明:

字符串 str1 中首次出现 str2 的位置。

参数说明:

str1 : 要操作的字符串。

str2 : 要操作的字符串。

返回值:

number: 返回位置, 从 1 开始计数, 未找到返回-1。

示例:

```
ret = str_find ("abcde", "ab")
```

```
str_split(string : str1, string : separator)
```

函数说明:

按照字符 separator 分割 str1 字符串, 返回列表。

参数说明:

str1 : 要操作的字符串。

separator:分割字符。

返回值:

num_list, 字符串列表。

示例:

```
ret = str_split ("abcde","c")
```

```
str_at(string : str1, number : index)
```

函数说明:

获取字符串 str1 中 index 处的字符, index 从 1 开始。

参数说明:

str1 : 要操作的字符串。

index : 索引, 从 1 开始计数。

返回值:

string, index 超出范围返回空字符串。

示例:

```
ret = str_at ("abcde", 2)
```

1.5.9 辅助函数

<code>list_len (list:value)</code>
<p>函数说明： 该函数返回输入列表的长度。</p> <p>参数说明： value: 输入的列表。</p> <p>返回值： number: 列表的长度。</p> <p>示例： ret = list_len ({1,1,1,1,1})</p>
<code>is_valid (value)</code>
<p>函数说明： 该函数判断输入的条件是否合法。</p> <p>参数说明： value: 输入的数据，会根据数据类型自动推断，判断逻辑如下 当 value 为 boolean 类型时 true 为合法，false 为非法 当 value 为 list 类型时非空为合法，空为非法 当 value 为 string 类型时非空为合法，空字符串为非法 当 value 为 number 类型时非 0 为合法，0 为非法 当 value 为 nil 类型时为非法。</p> <p>返回值： false: 不合法。 true: 合法。</p> <p>示例： ret = is_valid (“”)</p>
<code>int16_to_byte_list (number:value)</code>
<p>函数说明： 将 int16 数据类型按照内存结构转换成 byte list。</p> <p>参数说明： value: int16 类型数据。</p> <p>返回值： byte 列表。</p> <p>示例： ret = int16_to_byte_list (5)</p>

uint16_to_byte_list (number:value)

函数说明:

将 uint16 数据类型按照内存结构转换成 byte list。

参数说明:

value: uint16 类型数据。

返回值:

byte 列表。

示例:

```
ret = uint16_to_byte_list (5)
```

int32_to_byte_list (number:value)

函数说明:

将 int32 数据类型按照内存结构转换成 byte list。

参数说明:

value: int32 类型数据。

返回值:

byte 列表。

示例:

```
ret = int32_to_byte_list (5)
```

uint32_to_byte_list (number:value)

函数说明:

将 uint32 数据类型按照内存结构转换成 byte list。

参数说明:

value: uint32 类型数据。

返回值:

byte 列表。

示例:

```
ret = uint32_to_byte_list (5)
```

float_to_byte_list (number:value)

函数说明:

将 float 数据类型按照内存结构转换成 byte list。

参数说明:

value: float 类型数据。

返回值:

byte 列表。

示例:

```
ret = float_to_byte_list (5.0)
```

`double_to_byte_list (number:value)`

函数说明:

将 double 数据类型按照内存结构转换成 byte list。

参数说明:

value: double 类型数据。

返回值:

byte 列表。

示例:

```
ret = int16_to_byte_list (5.0005)
```

`byte_list_to_int16 (byte_list:value)`

函数说明:

将 byte list 数据类型按照内存结构转换成 int16 类型数据。

参数说明:

value: byte list(number list)类型数据。

返回值:

int16 类型数据。

示例:

```
ret = byte_list_to_int16 ({0x01,0x05})
```

`byte_list_to_uint16 (byte_list:value)`

函数说明:

将 byte list 数据类型按照内存结构转换成 uint16 类型数据。

参数说明:

value: byte list(number list)类型数据。

返回值:

uint16 类型数据。

示例:

```
ret = byte_list_to_uint16 ({0x01,0x05})
```

`byte_list_to_int32 (byte_list:value)`

函数说明:

将 byte list 数据类型按照内存结构转换成 int32 类型数据。

参数说明:

value: byte list(number list)类型数据。

返回值:

int32 类型数据。

示例:

```
ret = byte_list_to_int32 ({0x01,0x05})
```

byte_list_to_uint32 (byte_list:value)
<p>函数说明: 将 byte list 数据类型按照内存结构转换成 uint32 类型数据。</p> <p>参数说明: value: byte list(number list)类型数据。</p> <p>返回值: uint32 类型数据。</p> <p>示例: ret = byte_list_to_uint32 ({0x01,0x05})</p>
byte_list_to_float (byte_list:value)
<p>函数说明: 将 byte list 数据类型按照内存结构转换成 float 类型数据。</p> <p>参数说明: value: byte list(number list)类型数据。</p> <p>返回值: float 类型数据。</p> <p>示例: ret = byte_list_to_float ({0x01,0x05})</p>
byte_list_to_double (byte_list:value)
<p>函数说明: 将 byte list 数据类型按照内存结构转换成 double 类型数据。</p> <p>参数说明: value: byte list(number list)类型数据。</p> <p>返回值: double 类型数据。</p> <p>示例: ret = byte_list_to_double ({0x01,0x05})</p>
float2word (number:value)
<p>函数说明: 将 float 数据类型按照内存结构转换成两个 word 类型数据。</p> <p>参数说明: value: float 类型数据。</p> <p>返回值: {word_H, word_L}。</p> <p>示例: ret = float2word (5.5)</p>

`word2float(number:word_H, number:word_L)`

函数说明:

将两个 word 类型数据按照内存结构转换成一个 float 类型数据。

参数说明:

word_H, word_L: word 类型数据。

返回值:

float 类型数据。

示例:

```
ret = word2float (5)
```

1.5.10 力控函数

fc_start()
<p>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。</p> <p>参数说明： 无</p> <p>返回值： 无</p> <p>示例： fc_start ()</p>
fc_stop()
<p>函数说明： 该指令控制机械臂退出末端力控。</p> <p>参数说明： 无</p> <p>返回值： 无</p> <p>示例： fc_stop ()</p>
fc_move()
<p>函数说明： 该指令控制机械臂仅产生末端力控运动。</p> <p>参数说明： 无</p> <p>返回值： 无</p> <p>示例： fc_move ()</p>

<code>fc_config(number_list: direction, number_list: ref_ft, number_list: damp, number_list: max_vel, string: tool, string: wobj, number: type, number_list:ft_deadzone)</code>
<p>函数说明: 该指令修改并配置机器人末端力控参数。</p> <p>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，X/Y/Z 方向单位(N/(m/s)，RX/RX/RZ 方向单位(Nm/(rad/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)，RX/RX/RZ 方向单位(rad/s)。 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。 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/RX/RZ 方向单位 Nm。</p> <p>返回值: 无</p> <p>示例: <code>fc_config ({true,false,false,false,false,false},{0.1,0,0,0,0,0},{1000,1000,1000,57.29,57.29,57.29},{0.15,0.15,0.15,1.04,1.04,1.04}, "", "",0, {0,0,0,0,0,0})</code></p>
<code>fc_guard_deact()</code>
<p>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。</p> <p>参数说明: 无</p> <p>返回值: 无</p> <p>示例: <code>fc_guard_deact ()</code></p>

<code>fc_guard_act(number_list: direction, number_list: ref_ft, string: tool, string: wobj, number: type)</code>
<p>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。</p> <p>参数说明: <code>direction</code>: 6 个笛卡尔空间方向末端力安全监控开关, 开为 <code>true</code>, 关为 <code>false</code>。 <code>ref_ft</code>: 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。 <code>tool</code>: 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。 <code>wobj</code>: 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。 <code>type</code>: 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。</p> <p>返回值: 无</p> <p>示例: <code>fc_guard_act({true,false,false,false,false,false},{1,0,0,0,0,0}, "", "", 0)</code></p>
<code>fc_force_set_value(number_list: direction, number_list: ref_ft)</code>
<p>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。</p> <p>参数说明: <code>direction</code>: 6 个末端力传感器出力设置标志位, 需要设置为 <code>true</code>, 不需要设置为 <code>false</code>。 <code>ref_ft</code>: 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。</p> <p>返回值: 无</p> <p>示例: <code>fc_force_set_value({true,false,false,false,false,false},{1,0,0,0,0,0})</code></p>

`fc_wait_pos(number_list: middle_value, number_list: range, boolean: absolute, number: duration, number: 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。

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:

无

示例:

`fc_wait_pos ({0.005,0.005,0,0,0,0},{0.001,0.001,0,0,0,0},false,1000,5000)`

`fc_wait_vel(number_list: middle_value, number_list: range, boolean: absolute, number: duration, number: 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 方向速度范围[-5, 5]，单位 (m/s)，RX/RX/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/RX/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:

无

示例:

`fc_wait_vel ({0.01,0.01,0,0,0,0},{0.001,0.001,0,0,0,0},false,0,5000)`

<code>fc_wait_ft(number_list: middle_value, number_list: range, boolean: absolute, number: duration, number: timeout)</code>
<p>函数说明:</p> <p>该指令控制机械臂在执行 <code>fc_start()</code> 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 <code>fc_stop()</code> 函数被执行停止末端力控。</p> <p>参数说明:</p> <p><code>middle_value</code>: 力判断条件绝对值，范围[-1000, 1000]，X/Y/Z 方向单位 N，RX/RX/RZ 方向单位 Nm。</p> <p><code>range</code>: 力判断条件偏移范围大小，X/Y/Z 方向单位 N，RX/RX/RZ 方向单位 Nm。</p> <p><code>absolute</code>: 绝对/增量条件判断标志位，<code>true</code> 为绝对力判断，<code>false</code> 为增量力判断。</p> <p><code>duration</code>: 条件满足触发保持时间，单位 ms。</p> <p><code>timeout</code>: 条件满足触发超时时间，单位 ms。</p> <p>返回值:</p> <p>无</p> <p>示例:</p> <p><code>fc_wait_ft ({1,1,0,0,0,0},{0.1,0.1,0,0,0,0},false,0,5000)</code></p>

<code>fc_wait_logic(number : pos, number : vel, number : force)</code>
<p>函数说明:</p> <p>该指令控制机械臂在执行 <code>fc_start()</code>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。</p> <p>参数说明:</p> <p><code>pos</code>: 位置条件判断, 0 代表不启用, 1 代表与逻辑, 2 代表或逻辑;</p> <p><code>vel</code>: 速度条件判断;</p> <p><code>force</code>: 力条件判断;</p> <p>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}。</p> <p>返回值:</p> <p>无</p> <p>示例:</p> <p><code>fc_wait_logic (1,0,2)</code></p>

fc_get_ft()
<p>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。</p> <p>参数说明: 无</p> <p>返回值: 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</p> <p>示例: ret = fc_get_ft ()</p>

fc_mode_is_active()
<p>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。</p> <p>参数说明: 无</p> <p>返回值: 机器人末端力控启用返回 true, 未启用返回 false。</p> <p>示例: ret = fc_mode_is_active ()</p>

1.5.11 运动优化函数

enable_speed_optimization()
<p>函数说明:</p> <p>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。</p> <p>参数说明:</p> <p>无</p> <p>返回值:</p> <p>无</p> <p>示例:</p> <p>enable_speed_optimization ()</p>
disable_speed_optimization()
<p>函数说明:</p> <p>该指令用以控制机械臂退出速度优化。</p> <p>参数说明:</p> <p>无</p> <p>返回值:</p> <p>无</p> <p>示例:</p> <p>disable_speed_optimization ()</p>
enable_acc_optimization()
<p>函数说明:</p> <p>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。</p> <p>参数说明:</p> <p>无</p> <p>返回值:</p> <p>无</p> <p>示例:</p> <p>enable_acc_optimization ()</p>

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:

无

示例:

disable_acc_optimization ()

enable_vibration_control()

函数说明:

该指令用以开启对机械臂末端振动进行优化。

参数说明:

无

返回值:

无

示例:

enable_vibration_control ()

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。

参数说明:

无

返回值:

无

示例:

disable_vibration_control ()

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:

无

示例:

enable_singularity_control ()

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:

无

示例:

disable_singularity_control ()

enable_posture_control()

函数说明:

该指令用以开启融合段姿态约束功能。

参数说明:

无

返回值:

无

示例:

enable_posture_control ()

disable_posture_control()

函数说明:

该指令用以关闭融合段姿态约束功能。

参数说明:

无

返回值:

无

示例:

disable_posture_control ()

1.5.12 复合运动函数

<code>combine_motion_config(number:type, number:ref_plane, number:fq, number:amp, number:el_offset, number:az_offset, number:up_height, numble_list:time)</code>
<p>函数说明： 该指令控制机械臂执行复合运动。</p> <p>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：工件 XOY，1：工件 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</p> <p>返回值： 无</p> <p>示例： <code>combine_motion_config (1,0,1,0.1,0,0,0,{0,0})</code></p>

<code>enable_combined_motion ()</code>
<p>函数说明： 该指令用以开启复合运动。</p> <p>参数说明： 无</p> <p>返回值： 无</p> <p>示例： <code>enable_combined_motion ()</code></p>

<code>disable_combined_motion ()</code>
<p>函数说明： 该指令用以关闭复合运动。</p> <p>参数说明： 无</p> <p>返回值： 无</p> <p>示例： <code>disable_combined_motion ()</code></p>