JAKA®

节卡机器人

脚本编程手册

翻译版本（zh）

文档版本：1.1

软件版本：1.5

注意：

协作机器人的定义遵循国际 ISO 标准及国标相关规定来保护作业者的安全，我们不推荐直接将机器人本体应用于作业对象为人体的场合。但机器人应用者或应用开发者确有需要涉及机器人作业对象为人体的场合时，需要在应用者或应用开发者充分评估并在人员安全得到保障的前提下，为机器人本体配置安全可靠、经过充分测试及认证的安全防护系统，以保护人员安全。

节卡会定期对手册进行修正和完善，其内容可能会更新，恕不另行通知。使用本手册前请认真核对实际产品信息。

本手册适用于节卡推出的全部产品和/或服务（后文统称为“产品”），手册所包含的信息按产品“现状”提供，受中华人民共和国法律（不包括香港、澳门与台湾地区法律）的约束并依据其解释，在法律允许的最大范围内，本手册不构成节卡任何形式的明示或暗示的陈述或保证，亦不构成对节卡有关产品的适销性、适用于特定目的、达到预期效果以及不侵权的保证。节卡对本手册中仍然可能出现的任何错误、遗漏以及对使用本手册及其所介绍产品而引起的意外或间接伤害概不负责。安装、使用产品前，请仔细阅读本手册。

本手册图片仅供参考，请以实物为准。

若节卡机器人产品出现被改造或者拆卸的情况，节卡不负责无偿售后工作。

节卡提醒用户在使用、维修节卡机器人时必须使用安全设备，必须遵守安全条款。

节卡机器人的程序设计者、机器人系统的设计和调试者，必须熟悉节卡机器人的编程方式和系统应用安装。

手册使用说明

本手册主要介绍了机器人的脚本编程指令及使用示例。

本手册面向的用户应接受过基本的编程培训，这将更加有助于机器人的使用。

更多信息

如果您还想了解更多的产品信息，请扫描右侧二维码访问我们的官网 www.jaka.com。

目录

第 1 章 JAKA 编程脚本简介

JAKA 编程脚本是控制节卡机器人的专用编程语言。用户可以根据 JAKA 编程脚本所规定的语法进行编程以达到控制机器人的目的。

第 2 章 JAKA 编程脚本基础

2.1 JAKA 基础语法

2.1.1 标识符

在 JAKA 编程脚本中，标识符不区分大小写，且其命名需满足以下规则：

1. 只能使用英文字母、数字和下划线
2. 第一个字符不能是数字
3. 不能将关键字用作标识符
4. 最长支持 255 个字符，建议不要超过 30 个字符

示例：

```
#合法
_var1 = 1
var2  = 1
VAR3  = 1
_2KDDinKAEld74Zl8WzKP = 1
#不合法
4VAR = 1
if = 1
```

2.1.2 预留关键字

下面的列表显示了在 JAKA 机器人编程脚本中的保留字。这些保留字不能用作常数或变数，或任何其他标识符名称。

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

2.1.3 脚本注释

JAKA 机器人编程脚本支持单行注释，不支持多行注释。单行注释采用 # 开头，注释可以在语句或表达式行末。

示例：

```
# the first comment
str = "Hello, Python!"    # the second comment
```

2.1.4 语句

JAKA 机器人编程脚本仅支持单行语句。不支持一行语句拆分到多行书写，也不支持一行中书写多条语句。

2.2 基本类型

JAKA 机器人编程脚本提供标量、字符串与数组三种基本类型数据支持。在数组的基础上，预定义系统变量供用户使用。

2.2.1 标量

JAKA Script 不区分布尔类型、整形与浮点型数据。对布尔类型，false 对应 0， true 对应 1。

示例：

```
var = 1
#或
var = 1.0
#或
var = (expr1 > expr2)
```

2.2.2 字符串

用户定义字符串时需要使用英文双引号（""）进行包裹。目前 JAKA Script 支持的转义字符（见下表），用户在定义字符串时需要注意，否则会造成解析错误。

转义字符	描述
\\	反斜杠符号
\'	单引号
\"	双引号
\n	换行
\t	横向制表符
\r	回车

示例：

```
string =   "this is a \"string\""
string =   "This is a string \n"
```

2.2.3 数组

数组是存放一组相同数据类型数据的容器。目前仅支持标量类型的数据，不支持字符串数组或数组的嵌套。

数组定义

语法形式：

```
arr = [...]   #定义一个数组
arr = []      #定义一个空数组
```

单元素的负索引访问

假设数组变量 array 的长度为 N，对于单个元素的访问，支持以下形式：

array[index]

元素的索引支持负值，即 index 范围支持 [-N, N-1]。当 index 为非负值时，元素访问满足以下关系：

array[index] = array[-N+index]

当 index 取值不在支持范围内，则报错数组访问越界并终止程序执行。

示例：

```
a = [1,2,3,4,5,6,7,8,9,0]
b = a[-10] # b = a[0]
b = a[-11] #  报错数组访问超界
```

数组的子区间访问

支持访问指定数组的区间内指定间隔的子序列，并以数组形式返回。对数组的子区间的访问，具体格式为：

array[startIndx ： endIndex ： step]

当 step 为 0 时，程序报错，并会终止程序执行；

其他情况下，即使 startIndex, endIndex，step 不满足逻辑条件时不会报错（超过数组边界），仅在指定数组有效的范围内返回满足条件的值，如不存在则返回空数组。

提供一种 Step 为 1 的子区间访问的特殊形式，即 Step 参数省略：

array[startIndex, endIndex]

示例：

```
a = [1,2,3,4,5,6,7,8,9,0]
b = a[0：5]            #[1, 2, 3, 4, 5]
b = a[-5：10：1]      #[6, 7, 8, 9, 0]
```

数组与位姿表示

在 JAKA 机器人编程脚本中，采用 6 元素数组表示机器人关节位置或空间位姿，长度单位为 mm，角度单位为°。

示例：

```
endPosJ = [90,90,90,90,90,90] #关节空间位置数组
endPosL = [663.5,8.159996,6.950005,90,0,0] #笛卡尔空间位置数组
```

2.2.4 系统变量

在作业程序中定义的变量在程序执行结束之后将被释放。对于希望值能够持久保存的变量，JAKA 编程脚本提供系统变量机制。系统变量可在程序中直接使用，在程序中修改值后作业终止或系统关机后，其值仍然可以保留。系统变量只支持标量类型，目前支持用户使用的系统变量共 100 个，其访问方式如下：

sysvar[id]，其中 id∈[5500, 5599]

示例：

```
sysvar[5500] = 100
a = sysvar[5500]
```

注：系统变量暂不支持负索引与区间访问。

2.3 表达式

2.3.1 算术运算符

算术运算符用来进行四则运算，按照运算符优先级将其分组，（*、/、%、**）的优先级高于（+、-）。优先级高的运算符比优先级低的运算符结合得更加紧密，下表的运算符满足左结合律，即当运算符优先级相同时按照从中向右的顺序进行。

算术运算符		
运算符	功能	用法
* / % **	乘法 除法 求余 求幂	expr * expr expr / expr expr % expr expr ** expr

+	加法	+expr
-	减法	-expr

2.3.2 逻辑与关系运算符

关系运算符运算符作用于算术类型，逻辑运算符作用于任意能转换成布尔值的类型。逻辑运算符和关系运算符的返回值类型都是布尔类型。

逻辑运算符和关系运算符			
结合律	运算符	功能	用法
右	！	逻辑非	！ expr
左	&&	逻辑与	expr && expr
左	||	逻辑或	expr || exprr
左	<	小于	expr < expr
左	>	大于	expr > expr
左	==	等于	expr == expr
左	!=	不等于	expr != expr
左	<=	小于等于	expr <= expr
左	>=	大于等于	expr >= expr

2.3.3 位运算符

位运算符作用于整数类型的运算对象，并把运算对象看作是二进制位的集合，目前仅支持异或操作。

位运算符(左结合律)		
运算符	功能	用法
^	异或	expr1 ^ expr2

2.4 语句

通常情况下，语句是顺序执行的。但除非是最简单的程序，否则仅有顺序执行远远不够，因此 JAKA 编程脚本提供了一组控制流语句以支持更复杂的执行控制。

2.4.1 简单语句

JAKA 机器人编程脚本中，要求语句单独成行，且大多数语句无需结束符。简单语句包括表达式语句、函数调用语句等。

2.4.2 条件语句

if..end 语句

if 语句的作用是：判断一个指定的条件是否为真，根据判断结果决定是否执行另外一条语句。if 语句包括两种形式，一种含有 else 分支，另外一种没有。

语法形式如下：

```
if(condition)：
statement
end
```

示例：

```
condition = get_digital_output(0,1)
if(condition)：
endPosJ = [0,0,0,0,0,0    ]
movj(endPosJ,0,60,200,0)
end
```

if..else..end 语句

语法形式如下：

```
if(condition)：
statement
else：
statement
end
```

if..elif..else..end 语句

语法形式如下：

```
if(condition1)：
statement
elif(condition2)：
statement
else：
statement
end
```

示例：

```
condition =    get_digital_output(0,1)
if(condition)：
endPosJ = [0,0,0,0,0,0]
movj(endPosJ,0,60,200,0)
else：
endPosL =[663.5,8.159996,6.950,90,0    ,0]
movl(endPosL,0,250,250,0)
end
```

2.4.3 循环语句

while 循环语句

只要条件为真，while 语句就会重复地执行循环体，语法形式如下：

```
while(condition)：
statement
end
```

在 while 结构中，只要 condition 的求值结果为真，就会一直执行 statement (通常是一个语法块)，condition 不能为空，如果 condition 第一次求值就是 false ，statement 一次都不执行。

示例：

```
while(i <= 4)：
endPosJ =[0,0,0,0,0,0    ]
endPosL =[663.5,8.159996,6.950005,90,0    ,0]
movl(endPosL,0,250,2    50,0)
i = (i+1)
end
```

2.4.4 跳转语句

跳转语句可以中断当前的 **while** 语句的执行过程，JAKA Script 提供了两种跳转语句 **break** 和 **continue**。

break 语句

break 语句负责终止离它最近的 **while** 语句，并从这些语句之后的第一条语句开始继续执行。**break** 的作用范围仅限于最近的 **while** 循环。

示例：

statement 是语句块，condition1 和 condition1 是判断条件

```
while(condition1)：
Statement
...
if(condition2)：
break
end
...
statement
end
```

continue 语句

continue 语句终止终止最近的循环中的当前迭代并立即开始下一次迭代。**continue** 语句只能出现在 **while** 循环体的内部。和 **break** 类似，**continue** 的作用范围仅限于最近的 **while** 循环。**continue** 中断当前的迭代，但是会继续执行当前的循环，对于 **while** 语句来说即继续判断条件的值。

示例：

statement 是语句块，condition1 和 condition1 是判断条件

```
while(condition1)：
Statement
...
if(condition2)：
continue
end
...
statement
end
```

第 3 章 JAKA 脚本预置函数

3.1 运动相关指令

3.1.1 基本运动

函数原型	函数描述
movl(var_pos, rel_flag, vel, acc,tol,end_cond)	该函数用于控制机器人沿直线运动。 var_pos： 六元素数组变量，用于指定笛卡尔空间目标位置，需提前定义。 rel_flag： 指定该段运动是相对运动还是绝对运动。 0 为绝对运动，var_pos 是当前用户或世界坐标系下的绝对位置； 1 为相对用户坐标系的增量运动，此时目标指令位置满足： target_pos.P = curr_pos.P + var_pos.P target_pos.R = var_pos.R * curr_pos.R 2 为相对工具坐标系的增量运动，此时目标位置指令满足： target_pos.P = curr_pos.R * var_pos.P + curr_pos.P target_pos.R = curr_pos.R * var_pos.R vel： 运动速度，单位 mm/s。 Acc ： 直线加速度 单位： mm/ s^2。 tol： 用于指定终点误差。若为 0，则将准确到达目标点；若为正值，则可能会与后续段混合，不准确到达目标点。 End_cond：该段指令停止条件，可选参数。为一个包括 3 个参数的数组，包括： di_type：输入类型，-1 表示不使用， 0 表示 basic io，1 表示 TIO， 2 表示 MODBUS IO; di_index： IO 地址; di_state ： IO 满足条件的状态 0/1.
movj(var_pos, rel_flag, vel, acc,tol, end_cond)	var_pos： 六元素数组变量，用于指定关节空间目标位置，需提前定。 rel_flag： 指定该段运动是相对运动还是绝对运动： 1 为相对运动，此时 var_pos 给定值将被解释为关节空间位置增量； 0 位绝对运动，此时 var_pos 给定值将被解释为关节空间某绝对位置。 vel： 各关节指令速度，单位 deg/s。实际运动时，将受各关节实际最大速度约束。 acc ： 直线加速度 单位： deg/ s^2 tol： 用于指定终点误差。若为 0，则将准确到达目标点；若为正值，则可能会与后续段混合，不准确到达目标点。 End_cond：该段指令停止条件，可选参数。为一个包括 3 个参数的数组，包括： di_type：输入类型，-1 表示不使用， 0 表示 basic io，1 表示 TIO， 2 表示 MODBUS IO; di_index： IO 地址; di_state ： IO 满足条件的状态 0/1.
movc(var_pos_mid,var_pos_e	var_pos_mid： 六元素数组变量，用于指定空间圆弧的中间经过点，需提前定义。

函数原型	函数描述
nd,rel_flag, vel, acc, tol, turn_cnt, end_cond)	var_pos_end： 六元素数组变量，用于指定空间圆弧的最终停止点，需提前定义。 rel_flag： 指定该段运动是相对运动还是绝对运动。1 为相对运动，此时 var_pos 给定值将被解释为关节空间位置增量；0 位绝对运动，此时 var_pos 给定值将被解释为关节空间某绝对位置。 vel： 各关节指令速度，单位 deg/s。实际运动时，将受各关节实际最大速度约束。 Acc：用于指定运动的加速度，单位： mm/s^2 tol： 用于指定终点误差。若为 0，则将准确到达目标点；若为正值，则可能会与后续段混合，不准确到达目标点。 turn_cnt： 用户指定圆弧运动圈数（非负实数），可选参数。当圈数为 0 时，则按三点确定圆弧运动；如果圈数为非 0，则按三点确定的圆弧运动指定圈数，姿态保持初始点姿态与转轴的夹角相同。 End_cond：该段指令停止条件，可选参数。为一个包括 3 个参数的数组，包括： di_type：输入类型，-1 表示不适用， 0 表示 basic io，1 表示 TIO， 2 表示 MODBUS IO; di_index： IO 地址; di_state ： IO 满足条件的状态 0/1.
atlJntPos = get_atl_joint_pose()	该函数用于获取当前机器人实际关节位置。 atlJntPos 为返回的当前实际关节位置。
atlTcpPos = get_atl_tcp_pose()	该函数用于获取当前机器人实际末端工具中心点位置。 atlTcpPos 为返回的当前实际工具末端中心点位置。
atlFlangePos = get_atl_flange_pose()	该函数用于获取当前机器人末端法兰中心点位姿。 atlFlangePos 为返回的当前实际末端法兰中心点位姿。
enable_speed_override(type,vel,acc)	该函数设置函数块的速度与加速度。 type： 0 表示笛卡尔空间运动类型，1 表示关节空间运动类型。 vel： double 类型数据，表示设置的覆盖速度。 acc： double 类型数据，表示设置的覆盖加速度。
disable_speed_override(type)	该函数取消函数块的速度与加速度设置。 type： 0 表示笛卡尔空间运动类型，1 表示关节空间运动类型。

3.1.2 传送带跟踪

函数原型	函数描述
enable_conveyor_track(dir, mm_per_pulse)	使能传送带跟踪. dir：跟踪方向，可以为 X+, Y+, Z+, X-, Y-, Z-重的任意一个。 mm_per_pulse：该参数用于指定脉冲当量，即跟踪传送带上的编码器反馈每个脉冲对应传送带方向的位移（单位为 mm）。
disable_convayor_track()	关闭传送带跟踪。

3.1.3 运动柔顺控制

函数原型	函数描述
set_compliance_ft_config(id.admitctrlconfig,immed)	该函数用于恒力柔顺控制配置。admitctrlconfig 时配置参数数组。 id： 代表配置轴编号[0,5] admitctrlconfig=[opt,FTuser,FTreboundK,FTconstant,FTnormalTrack] (1)opt： 柔顺方向，可选值为 1 2 3 4 5 6 分别对应 fx fy fz mx my mz，0 代表没有勾选； (2)FTuser：用户用多大的力才能让机器人沿着某个方向以最大速度进行运动。设置的力和力矩值范围是需大于 0； (3)FTreboundK：回弹是机器人回到初始状态的能力。设置的力和力矩值范围需大于 0； (4)FTconstant： 代表恒力； (5)FTnormalTrack：法向跟踪,1 代表跟踪，0 则相反； immed： 1 为 即时指令， 0 为 非即时指令。若为非即时指令，则该指令的实际执行则在下一条运动指令之前执行
velCompliantCtrl = [level,rate1,rate2，rate3,rate4] set_compliance_velocity_level(velCompliantCtrl)	该函数用于设置速度柔顺控制阶梯速度。 在速度模式下，当机器人末端受力大于控制器设定值时，机器人会按照设定的倍率进行减速，直至传感器检测值小于控制力设定值。速度柔顺控制阶梯速度分三个等级，并且 1 > rate1 > rate2 > rate3 > rate4 >0 (1)等级为 1 时，只能设置 rate1,rate2 两个等级。rate3,rate4 的值为 0 (2)等级为 2 时，只能设置 rate1,rate2，rate3 三个等级。rate4 的值为 0 (3)等级为 3 时，能设置 rate1,rate2，rate3,rate4 4 个等级
compliantCondition = [fx,fy,fz,tx,ty,tz] set_compliance_condition(compliantCondition)	该函数用于设置速度柔顺控制力。 (1)set_compliance_condition 的参数是一个数组，设置力和力矩的停止条件。当机器人的实际力大于这个条件的时候，机器人开始按 set_compliance_velocity_level 设定的倍率减速。 (2)fx,fy,fz,tx,ty,tz 的范围是：大于等于 0。
disable_force_control()	该函数用于关闭力控。
enable_force_control(type, compensation)	柔顺控制类型配置和传感器补偿 (1)type： 0 代表不使用任何一种柔顺控制；1 代表 恒力柔顺控制；2 代表 速度柔顺控制； (2)compensation： 1 代表开启传感器补偿；0 代表关闭传感器补偿。
set_end_force_condition(axis,condition) condition=[enableEndCond,lowFlag,lowLimitForce,upFlag,	该函数用于设置力控终止条件。设置力控终止条件,不能立即生效,默认是非即时指令。 axis：代表轴号，取值范围是[0,5] enableEndCond：是否使能，1 代表生效，0 则相反 lowFlag：运动终止力下限生效标志位，1 代表生效，0 则相反

函数原型	函数描述
upLimitForce]	lowLimitForce：运动终止力下限 upFlag：运动终止力上限生效标志位，1 代表生效， 0 则相反 upLimitForce：运动终止力上限
set_force_control_frame (ftFrame)	该函数用于设置力控坐标系。 0 表示工具坐标系，1 代表世界坐标系
torq = get_sensor_torque(type)	该函数用于获取力矩底座或末端力矩传感器的力矩值。该值为补偿传感器末端负载后的净力矩值。 type： 0 为末端力矩传感器，1 为底座力矩传感器（暂不支持）

3.2 接口设置与查询

3.2.1 IO 控制与查询

函数原型	函数描述
set_digital_output(type, index, tarState, immed)	该函数用于控制数字输出信号。 type：0--标准 IO ; 1--工具 IO； 2 为扩展 IO; index： 所控制数字输出的编号，0-based。 tarState： 1 为 ON，0 为 OFF immed： 1 为 即时指令， 0 为 非即时指令。若为非即时指令，则该指令的实际执行则在下一条运动指令之前执行。 **注：即时 IO 指令会打断不同运动段之间的转接。**
set_analog_output(type, index, tarValue, immed)	该函数用于控制模拟输出信号。 type：0 标准 IO ; 1 工具 IO；2 为扩展 IO; index： 所控制模拟输出的编号，0-based。 tarValue： Real 类型值，用于指定模拟输出的目标值。 immed： 1 为 即时指令， 0 为 非即时指令。若为非即时指令，则该指令的实际执行则在下一条运动指令之前执行。 **注：即时 IO 指令会打断不同运动段之间的转接。**
get_digital_output(type, index)	该函数用于获取指定数字输出的状态。 type：0 标准 IO ; 1 工具 IO；2 为扩展 IO; index 为数字输出索引。 函数返回查询结果：0 -> off, 1 -> on
get_analog_output(type, index)	该函数用于获取指定模拟输出的状态。 type：0 标准 IO ; 1 工具 IO；2 为扩展 IO； index 为模拟输出索引。 函数返回 double 类型的查询结果。
get_digital_input(type, index)	获取指定数字输入的状态。 type：0 标准 IO ; 1 工具 IO；2 为扩展 IO； index 为数字输入索引。 函数返回查询结果：0 为 off, 1 为 on
get_analog_input(type, index)	获取指定模拟输入的状态。 type：0 标准 IO ; 1 工具 IO；2 为扩展 IO； index 为模拟输入索引。 函数返回 double 类型的查询结果。

函数原型	函数描述
wait_input(type, index, stat, time)	该函数用于等待指定输入变成指定状态，若超出最大等待时间，则置 TIMEOUT 状态标志，可用 GETTIMEOUT 函数查询。 type：0 标准 IO ; 1 工具 IO；2 为扩展 IO； ndex：输入的索引 tat： 期待的输入状态 time：最长等待时间，0 为无限等待
get_timeout()	该函数用于获取 wait_input 指令的执行结果， 1 为等待超时，0 为在指定时间内成功等到该信号。

3.2.2 TIO+控制与查询

函数原型	函数描述
tio_update_signal(strId, freq)	该函数用于以一定周期刷新指定的 TIO Modbus RTU 信号。 strId：信号量的标识符； freq：为 0 时，仅刷新一次；为其他正值时为刷新频率，刷新频率内部自动受限于 CAN 通讯瓶颈。
res = tio_get_signal_value(strId)	该函数用于获取指定信号的值。 strId：信号量的标识符。
tio_send_command(chnId, cmdBuf, crcType)	该函数用于向指定的 TIO RS485 通道发送字节数据。 chnId 为通道编号，0 位 RS485H， 1 为 RS485L； cmdBuf 为字节数组； crcType：发送数据类型，当且指定通道为 RS485 透传时生效，Modbus RTU 时写 0.

3.2.3 网络通讯

函数原型	函数描述
sockid = socket_open(ip, port)	该函数用于打开指定的 IP 与端口号，并将创建的 SOCKET 句柄保存在变量并返回。 ip：为字符串形式 TCP 服务器地址，如”192.168.1.10”； port：为 TCP 服务器端口号.
socket_close(sockid)	该函数用于关闭指定的套接字。 sockid： 需要关闭的套接字连接。
varname = socket_get_var(sockid, type, argname)	该函数用于请求远端对参数的设置。 sockid： 套接字 ID，需事先创建； type： 参数类型, 0 为整型，1 为浮点数，2 为字符串； argname： 字符串类型的变量名，表示想要获取的变量的值，形式可为”argname”，或 argname>； varname： 返回值存储变量。 该函数将通过 socket 发送字符串： get <arg_varname>，并期望接收数据格式：<arg_varname><value>，2s 时间的超时，超时后返回 0。 当期望发送一个数组时，上位机发送格式为： <arrName><{num1, num2, ..., numN}>

函数原型	函数描述
	当期望发送一个字符串时，上位机发送格式为： <strName><”stringValue”>
Res = socket_read_real(sockid, num)	该函数用于通过制定的套接字从外界获取（一组）数，并存储在数组变量中。 sockid： 套接字 ID，需事先创建; num： 期望接收的数的数量； res： 接收的数据结果，若接收失败，则返回标量 0，成功接收则均存在数组中。2s 的超时，若接收失败，将返回 0. 函数发送格式为 get#real#num#，期望接收数据格式为{num1, num2, ..., numN}
res = socket_read_string(sockid, prefix, suffix)	该函数用于通过指定的套接字从外界获取一段字符串，并存储在变量中。 sockid： 套接字 ID，需事先创建; prefix： 期望接收的字符串的前缀要求； suffix： 期望接收的字符串的后缀要求； res： 接收的数据结果。2s 的超时，若接收失败，将返回空的字符串. 函数发送格式为 get#string#prefix#suffix#，期望接收数据格式为”prefixSTRINGsuffix”
res = socket_send(sockid, var)	该函数用于通过指定的 SOCKET 以字符串形式发送一个变量的值。 sockid： 套接字 ID，需事先创建; var： 期望发送的变量，变量目前支持 Number，Number 数组，以及 String 类型。发送数组时，发送的字符串格式为”{Num1, Num2, ...}” res>： 发送结果标志。2s 的发送超时，如果发送成功，返回 1，否则返回 0。 发送的数据格式为 <varName><varValue> <intVar><12> <strVar><”string”> <listVar><{ele1, ele2, ele3, ...}>
res = socket_recv(sockid, timeout)	该函数用于从指定的 socket 接收数据，用户可设置超时时间。 注：该指令单纯用于接收数据，不会向另一连接端发送请求。 sockid： 套接字 ID，需事先创建; timeout： 接收超时设置，超时未接受到数据，将返回空字符串。正常接收，函数返回接收的字符串。如果超时未接收到数据，返回为空字符串，如果网络问题，则会返回标量-1。

3.3 参数设置与查询

函数原型	函数描述
set_payload(m, centriod)	该函数用于设置当前机器人负载： m： 负载质量，单位 kg, centriod： 负载质心，包含三个分量的数组,单位：mm

函数原型	函数描述
	注：该指令会同步运动，打断 Blending。
payload = get_payload()	该函数用于获取当前机器人负载； payload 为返回的机器人法兰末端总负载， payload[0]为负载质量（单位 kg），payload[1]~[3]为负载质心（单位 mm）。
set_collision_level(clsnLevel)	该函数用于设置当前碰撞检测灵敏度： clsnLevel： 碰撞灵敏度等级，可允许设置值为 0,1,2,3,4,5，与实际碰撞力阈值关系如下： 0：关闭； 1：25N； 2：50N； 3：75N； 4：100N； 5：125N 注：该指令会同步运动，打断 Blending。
res= get_collision_level()	该函数用于获取当前碰撞检测灵敏度。
set_tool(var_tool)	该函数用于设置工具末端相对于末端法兰的位置偏置。法兰坐标系垂直端面向外为 z 轴正方向， 工具 IO 连接端口为 y 轴负方向。var_tool 为工具末端相对于法兰坐标系的偏置。
set_tool_id(id)	该函数用于设置工具坐标系 id，取值[0,10)。控制系统内最大支持 10 个工具，通过设置 ID 指定当前使用的工具偏置值。
res= get_tool_offsets()	该函数用于获取当前工具坐标系偏置值.
res= get_tool_offsets_of(id)	该函数用于获取指定工具坐标系偏置值，id 为指定的工具坐标系，取值范围是[1, 10]。
set_user_frame(user_frame)	该函数用于设置用户坐标系。设置时工具 ID 将设置为-1，程序中设置后不会被保存。
set_user_frame_id(id)	该函数用于设置用户坐标系 id，取值[0,10)。控制系统内最大支持 10 个工具，通过设置 ID 指定当前使用的工具偏置值。
res= get_user_frame()	该函数用于获取当前用户坐标系偏置值。
res=get_user_frame_of(id)	该函数用于获取指定用户坐标系偏置值，id 为指定的用户坐标系，取值范围是[1, 10]。

3.4 辅助函数库

3.4.1 数学计算库

函数原型	函数描述
res = atan2(y，x)	反正切函数，返回以角度表示的 y/x 的反正切
res = abs(arg)	求表达式绝对值
res = acos(arg)	反余弦函数，返回为角度值
res = asin(arg)	反正弦函数，返回为角度值
res = cos(arg)	余弦函数

函数原型	函数描述
res = sin(arg)	正弦函数
res = tan(arg)	正切函数
res = floor(arg)	向下取整
res = ceil(arg)	向上取整
res = round(arg)	四舍五入取整
res = sqrt(arg)	取平方根
res = rad2deg(arg)	弧度转角度
res = deg2rad(arg)	角度转弧度

3.4.2 位姿运算库

函数原型	函数描述
res = pose_add(pos1, pos2)	该函数用于计算两个位姿之间的简单累加，并返回计算结果。运算满足： res.P = pos1.P + pos2.P res.R = pos2.R *pos1.R
res = pose_sub(pos1, pos2)	该函数用于计算两个位姿之间的简单相减，并返回计算结果。运算满足： res.P = pos1.P - pos2.P res.R = inv(pos2.R) *pos1.R
res = pose_dist(pos1, pos)	该函数用于计算两个位姿之间的距离。仅考虑位置坐标。函数返回计算结果。
res = pose_inv(pos)	该函数用于计算某个位置的逆变换， 并返回 pose 形式的计算结果。
res = pose_trans(p_from, p_from_to)	该函数计算位置变换，并返回位置结果。计算公式如下： p_res = p_from * p_from_to
res = pose_intpl(pos1, pos2, alpha)	该函数按照给定的系数计算两点之间的插补点： p_res = pos1 + (pos2 - pos1) * alpha, 其中 alpha >= 0 且 alpha <= 1
res = xy_plane_trans(posebase, dx, dy, drz)	该函数将刚体在 XY 平面内做变换，在 posebase 处先绕 Z 轴旋转 drz 度，然后沿 X 轴平移 dx，沿 Y 轴平移 dy。 函数返回变换后的位姿。
res = yz_plane_trans(posebase, dy, dz, drx)	该函数将刚体在 YZ 平面内做变换，在 posebase 处先绕 X 轴旋转 drx 度，然后沿 Y 轴平移 dy，沿 Z 轴平移 dz。 函数返回变换后的位姿。
res = zx_plane_trans(posebase, dz, dx, dry)	该函数将刚体在 ZX 平面内做变换，在 posebase 处先绕 Y 轴旋转 dry 度，然后沿 Z 轴平移 dz，沿 X 轴平移 dx。

函数原型	函数描述
	函数返回变换后的位姿。
res = kine_inverse(posJ, posP)	该函数用于计算运动学逆解。功能为计算输入的笛卡尔空间位置 posP 对应的关节空间位置。posJ 为这个逆解关节值附近的一组关节值，用来确定逆解函数多解的选取。 函数返回逆运动学计算结果。当逆解失败时，程序报错终止。
res = kine_forward(posJ)	该函数用户计算运动学正解。功能为计算输入的关节角度的所在笛卡尔空间位置。 函数返回正运动学计算结果。

3.4.3 字符串操作库

函数原型	函数描述
resLen = sprintf(bufferOut, stringFormat, ...)	该函数用于对字符串的格式化输出，即将指定数据按指定格式输出为字符串。 bufferOut： 格式化后的字符串结果，变量需事先定义； stringFormat： 定义格式字符串，目前仅支持变量形式。支持对%f, %d, %s 三种类型变量的格式化。格式如：“%f, %s, %d, ...”； ...： 待格式化变量，可变参数。支持整形，浮点型以及字符串类型的变量，或者常数。 strLen： 如果格式化成功返回字符串的长度，否则返回-1。
resNum = sscanf(bufferIn, stringFormat, ...)	该函数用于对字符串的格式化输入，即将匹配一定格式的字符串将数据输入到变量中。 bufferIn： 待匹配的字符串，目前仅支持变量格式； stringFormat： 定义格式字符串，目前仅支持变量形式。支持对%f, %d, %s 三种类型变量的格式化。格式如：“%f, %s, %d, ...”； %s 匹配字符串，读取连续字符直到遇到一个空格字符（空白、换行和制表符等）或格式化字符串中下一个匹配字符（非%s、%f、%d）。 %d 匹配十进制整数：数字前面的+或-号可选 %f 匹配一个浮点数。 ...： 格式化结果变量，可变参数。支持整形，浮点型以及字符串类型的变量，或者常数。 resNum： 如果格式化成功返回匹配的变量的数目，否则返回-1。
strRes = string_concat(str1, str2)	该函数用于两个字符串的拼接，并返回拼接后的字符串。 str1： 带拼接的子字符串，变量或者常量； str2： 待拼接的子字符串，变量或者常量； strRes： 字符串拼接结果。

函数原型	函数描述
resLen = get_string_from_array(arrayIn, splittingToken, stringOut)	该函数用于将满足一定格式的字符串转为为数组变量。 arrayIn 为用户给定的数组； splittingToken 为字符串格式的分隔符; stringOut 为转换成功后字符串。 如果转换成功，函数返回字符串的长度（不包含截止符），失败则**返回-1.**
resNum = get_array_from_string(stringIn, splittingToken, arrayOut)	该函数用于将给定字符串转换为数组变量。 stringIn 为用户给定的字符串（包含个数组中的数据）; splittingToken 为字符串格式的分隔符; arrayOut 为转换成功后的数组。 如果转换成功，函数返回 arrayOut 的元素数量，失败则**返回-1.**
len = get_length(str_or_arr)	该指令用于获取字符串或数组的长度。 如果参数数据类型不匹配，则会报错。
res = strcmp(str1, str2)	该指令用于两个字符串间的比较。

3.4.4 程序控制与调试

函数原型	函数描述
log_message(level, message)	该函数用于用户自行添加日志信息。 level，日志消息类型。具体定义如下： INFO：1; WARNING：2; ERROR：3; message：日志消息文本。不支持 Unicode 类型。
clock = get_system_clock()	该函数用户获取系统实时时钟。 返回值为 double 类型浮点数，单位为 ms（毫秒），精确到 ns. 底层使用 CLOCK_MONOTONIC。
sleep(time)	该函数用于延时一段时间。 time： 延时时间长度，单位 s。 **注：该指令会打断不同运动段之间的混合。**
pause()	该函数用于控制作业程序暂停。
exit()	该函数用于控制作业程序终止。

节卡机器人股份有限公司
地址：上海市闵行区剑川路 610 号 33-35 幢
电话：400-006-2665
网址：www.jaka.com