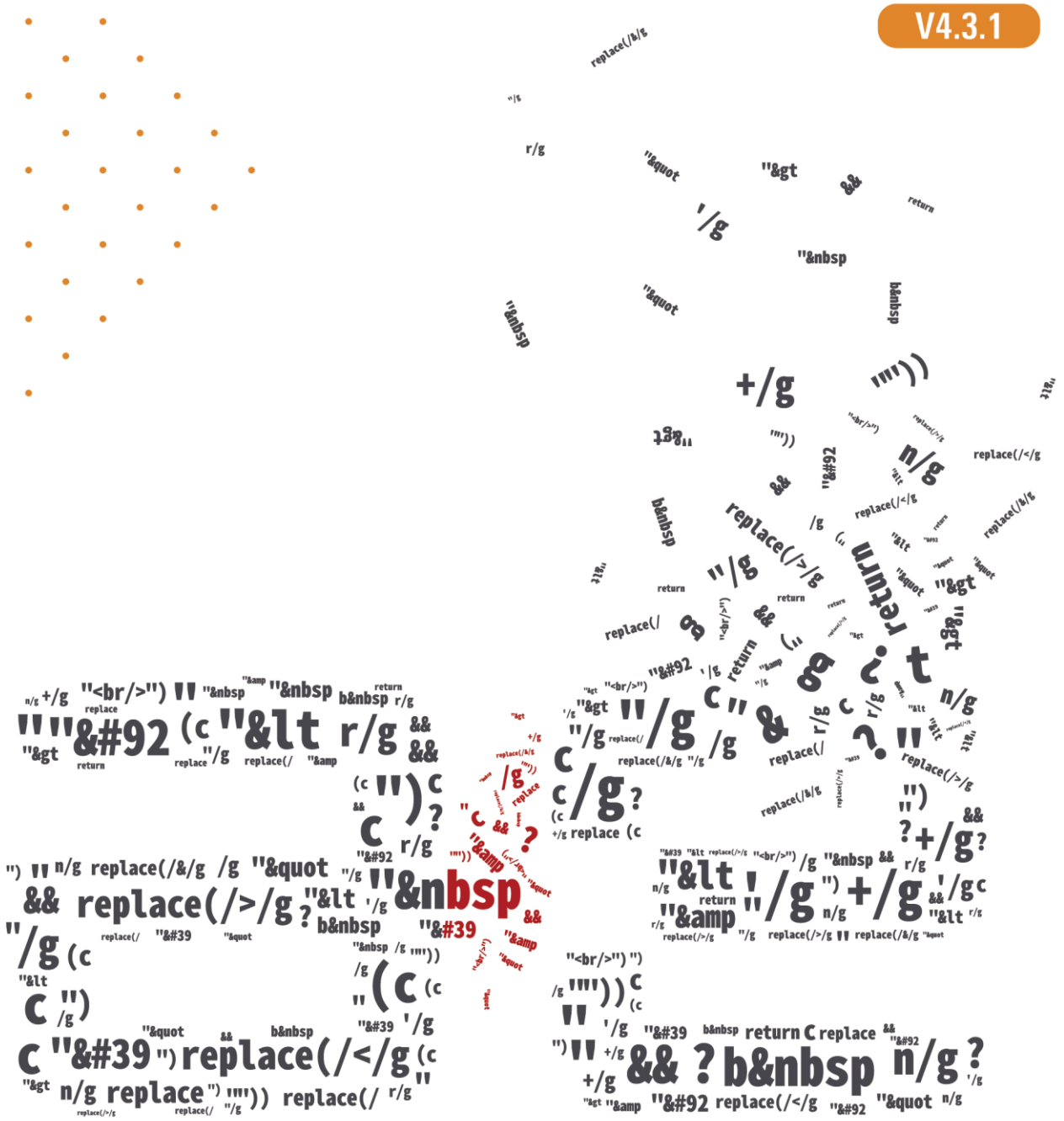
ARL编程手册

V4.3.1



SMARTER·STRONGER 配天机器人

前言

关于本手册

ARL 编程手册详细描述了 ARL 编程格式规范以及 ARL 支持的编程指令。

目标群体

- 操作人员
- 产品技术人员
- 技术服务人员
- 机器人示教员

常见标识含义

手册中出现标识及其含义详见下表 1。

表 1 本文中使用的标识

标志	含义
 危险	如不按照说明进行操作，就会发生事故，导致严重或致命的人员伤害，或严重的物品损坏
 警告	如不按照说明进行操作，可能发生事故，导致严重或致命的人员伤害，或严重的物品损坏
 注意	提示您需要注意的环境条件和重要事项，或快捷操作方法
 提示	提示您参阅其他文献和说明，以便获取附加信息或更加详细的操作说明

手册说明

文档相关信息见表 2。

表 2 文档相关信息

文档编号	BJM/SS-UG-02-003
文档版本	V4.3.1
软件版本号	2.6.4

本手册内容会有补充和修改，请定时留意我公司网站的“下载中心”，及时获取最新版本的手册。

我公司网站网址：http://robot.peitian.com/

通用安全说明

感谢贵公司购买本公司产品，本说明资料为安全使用本公司产品而需要遵守的内容，在使用之前，请务必仔细阅读相关手册，并且在理解该内容的前提下正确使用。

编程时尽可能在安全栅栏外进行，因不得已情形而需要在安全栅栏内进行时，应注意下列事项：

1. 仔细查看安全栅栏内情况，确认没有危险后再进入栅栏内部。
2. 要做到随时都可以按下急停按钮。
3. 应以低速运行操作机。
4. 应在确认整个系统的状态后进行作业，避免由于针对外围设备的遥控指令或动作等而导致作业人员陷入危险境地。

在编程结束后，务必按照规定步骤进行测试运转，此时，作业人员务必在安全栅栏外进行操作。

进行编程的作业人员，务必通过本公司的相关培训接受适当的培训。

符号约定

如下为一种含指令 TPWrite 的简化语法示例。

Movej j:,[v: | vp:],[s: | sp: | sl:],[t:],[dura:]

- 括号中不含强制性参数。
- 用方括号“[ ]”将可选参数括起来，可忽略这些参数。
- 互相排斥的参数不能同时存在于同一指令中，在同一指令中就要用竖线“|”隔开。
- 用波形括号“{ }”将可重复任意次的参数括起来。

上述示例采用了如下参数：

- j 为强制性参数。
- v、vp、s、sp、sl、t 和 dura 为可选参数。
- v 和 vp 互相排斥。
- s、sp 和 sl 互相排斥。

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

表 2 本文中使用的标识

版本	发布时间	修改说明
V4.2.0	2020/10/30	**第五次正式发布** 软件版本升级到 V2.6.3 ■ 新增结构体类型：Centroid_Pos（工具负载质心）/Inertia_Tensor（工具负载惯性主轴方向）/ToolInertiaPara（工具负载类型） ■ 新增运动指令：ccir(连续圆弧运动) ■ 新增 IO 相关函数：getnostopdi（不停前瞻异步获取单路 DI）/syncao（同步输出模拟量信号）/getao（获取模拟量输出信号） ■ 新增系统变量：$WOBJ_OFFSET(工件坐标系偏移)/$TOOL_OFFSET(工具坐标系偏移)
V4.3.0	2021/09/02	**第六次正式发布** 软件版本升级到 V2.6.4 ■ 新增平滑运动指令 spl ■ 新增碰撞检测开启指令 startdetect 和碰撞检测关闭 enddetect ■ 新增记录前台运行程序名及行号的函数见第“5.13.8~5.13.11”节 ■ 新增“5.6.4 ModBusTCP 相关函数” ■ 新增“5.7.5 ftobytes(浮点转换成字节)”和“5.7.6 tofloat(字节转换成浮点)”
V4.3.1	2022/05/05	**第七次正式发布** ■ 修正已知 Bug ■ 新增“文件大小查询函数、文件重命名函数、删除文件函数”

安全预防措施

在运行操作机和外围设备及其组成的操作机系统前，必须充分研究作业人员和系统的安全预防措施，图 1 为工业机器人安全工作示意图。

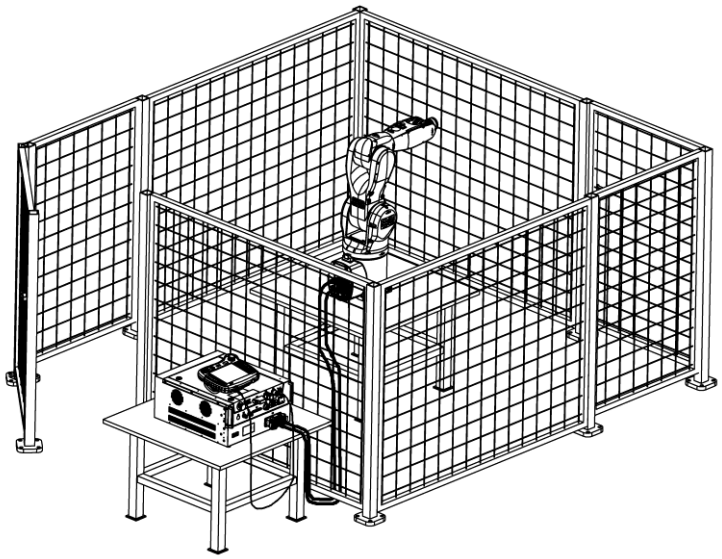


图 1 工业机器人安全工作示意图

作业人员定义

操作机的作业人员主要分为操作员、示教员、维护工程师三种，这三种作业人员需满足的条件描述如下：

操作员

- 进行操作机电源 ON/OFF 的操作。
- 通过操作面板来启动操作机程序。
- 无权进行安全栅栏内的作业。

示教员

- 具备操作员的职能。
- 可以在安全栅栏内进行操作机示教等。

维护工程师

- 具备示教员的职能。
- 可以进行操作机维护（修理、调整、更换等）作业。

作业人员的安全

在进行操作机操作、编程、维护时，操作员、示教员、维护工程师必须注意安全，至少应穿戴下列物品进行作业：

- 适合于作业内容的工作服
- 安全鞋
- 安全帽

在运用自动系统时，必须设法确保作业人员安全，进入操作机作业范围是十分危险的，应采取防止作业人员进入操作机动作范围的措施。

下面列出一般性注意事项，请妥善采取确保作业人员安全的相应措施：

- 运行操作机系统的作业人员，应接受本公司的培训并通过相关考核。
- 在设备运行时，即使操作机看上去已经停止，也有可能是因为操作机在等待启动信号而处在即将动作的状态。此状态也应该视为操作机处在操作状态。为了确保作业人员安全，应当以警报灯等的显示或响声等来确认操作机处在停止状态或安全状态。
- 务必在系统周围设置安全栅栏和安全门，使得不打开安全门，作业人员就不能够进入安全栅栏内。安全门上应该设置互锁开关、安全插销等，以使作业人员打开安全门时，操作机就会停下。
- 外围设备均应电气接地。
- 应尽可能地将外围设备设置在操作机动作范围之外。
- 应采用在地板上画上线条等方式来标清操作机动作范围，使得操作者清楚包括操作机上配备的机械手等工具在内的操作机动作范围。
- 应在地板上设置垫片开关或者安装光电开关等，以便当作业人员将要进入操作机动作范围时，通过蜂鸣器和光等发出警报，使得操作机停下，由此确保作业人员安全。
- 应根据需要，设置一把锁，除负责操作的作业人员外，不能接通操作机电源。
- 在进行外围设备的单个调试时，务必断开操作机的电源。

示教员的安全

在进行操作机示教作业时，某些情况下需要进入操作机工作范围内，此时尤其要注意安全：

- 在不需要进入操作机动作范围的情况下，务必在操作机动作范围外进行作业。
- 在进行示教作业之前，应确认操作机或外围设备处在安全状态。
- 在迫不得已情况下需要进入操作机动作范围内进行示教作业时，应事先确认安全装置（如急停按钮，示教器紧急自动停机开关等）的位置和状态等。
- 示教员应特别注意，勿使其他人员进入操作机动作范围。

- 在操作机启动前，应充分确认操作机动作范围内没有人员且没有异常后再执行。
- 在示教结束后，务必按照下列步骤执行测试运转：
  1. 在低速下，单步执行至少执行一个循环，确认没有异常。
  2. 在低速下，连续运转至少一个循环，确认没有异常。
  3. 在中速下，连续运转至少一个循环，确认没有异常。
  4. 在运转速度下，连续运转一个循环，确认没有异常。
  5. 自动运行模式下执行程序。
- 示教员在操作机进行自动运转时，务必撤离到安全栅栏外。

目录

1 ARL 概述

1.1 术语和缩写词

- ARL AE Robot Language，即 AE 机器人编程语言。
- CP 运动 笛卡尔路径运动，主要包括直线和圆弧运动，对应运动指令中的 lin 和 cir。
- PTP 运动 点到点运动，对应运动指令中的 ptp 和 movej。
- DI digital input，数字输入信号。
- DO digital output，数字输出信号。
- AO analog output，模拟输出信号。
- RPP return to path point，RPP 运动是指机器人从当前位置返回到路径点的运动。

1.2 程序文件

ARL 为 AE Robot Language 缩写，即 AE 机器人编程语言。用户可以通过 ARL 编写控制机器人的用户程序。

ARL 程序文件后缀名为 arl，例如：AEMoveObj.arl。

1.3 代码注释

ARL 中的代码注释的 2 种格式：

- 使用“//”代表注释该行，“//”后面的部分即为注释

程序示例：

```
//waittime 5
waituntil getdi(2) //等待通道 2 DI 信号变为 true
```

编译器读入一行后，首先将“//”后面的部分忽略，之后再进行解析编译执行。

- 使用“/*”和“*/”注释一个区域

程序示例：

```
/* this prog is for moving object*/
movej j:{j1 20}
waittime 5
```

1.4 换行连接符

当程序中一行太长，希望将其分为两行或者多行编写时，可以采用换行连接符“\”。

程序示例：

```
if(~x == -6 && y == 3 && A == 60 && (A&B) == 12 \&& (A|B) == 61 && (A^B) == 49 && (~A) == -61
\&& (A<<2) == 240 && (A>>2) == 15)
return true
else
```

return false

endif

1.5 子程序

利用子程序技术可以让机器人程序更加模块化，可帮助编程人员有效的结构化设计程序。目的是不必将所有指令写入一个程序，而是将特定的流程、计算或过程放到单独的程序中，以达到模块化复用的目的。


提示

- 当子程序和主程序处于同一目录下时的调用方法：

  SubProg::func()
- 当子程序和主程序处于不同目录下时的调用方法：

  import "/home/ae/.../SubProg.arl" //导入子程序文件

  SubProg::func()

  程序指针执行到上述程序段时，会跳转到 SubProg 程序的 func 函数中。
- 子程序结构与一般程序没有明显区别，只是可以不包含主函数。
- 子程序被调用的函数结束后（即执行到 endfunc 后）程序指针会返回调用处，如果想提前结束子程序，可以在需要终止的地方插入 return 指令，这样会提前终止子程序的运行。

1.6 函数

当处理相似的，通常是重复的程序部分时，为了减少字符的数量和程序的长度，引入函数功能。

- 一个 ARL 程序可以由一个或多个函数组成

函数定义格式如下：

func returntype funcname(argtype argname,argtype argname……)

内部为函数体实现

endfunc

其中，returntype 为返回值类型，funcname 为函数名，argtype 为参数类型，argname 为参数名。

- 实现两个整数加法的函数的示例

程序示例：

func int add(int a,int b)

return a+b

endfunc

函数的调用格式如下：

funcname(arg1,arg2,……)

- 调用之前定义的 add 函数的方式

例如：

int c = add(1,2) //c 的值为 3

- 函数的返回值可以继续用在表达式中

例如：

int c = add(1,2) * 3 //c 的值为 9

- ARL 程序没有指针的概念

ARL 程序没有指针的概念，函数参数只是值传递，如果要某个函数内修改传入参数的值，需要将该参数声明为全局变量。

下述程序示例中，print 打印结果是 1，而不是 2。因为 add 函数对 a 进行的自加操作不会传递给变量 a。

程序示例：

```
func void add(int a)
a++
endfunc
func void main()
init()
int a=1
add(a)
print a
endfunc
```

- main 函数

ARL 程序中必须要实现一个 main 函数，也就是程序的入口函数。程序复位后，程序指针将指向 main 函数中的第一行代码。程序运行时，将从 main 函数的第一行代码开始执行。

main 函数没有参数也没有返回值，所以按如下形式定义：

main 函数定义

```
func void main()
……
endfunc
```

其中，void 表示函数没有返回值。

- 使用 ARL 语言编写 HelloWorld 程序

程序示例：

```
//HelloWorld.arl
func void main()
print "Hello world!"
endfunc
```

print 为打印输出函数，默认输出到 HMI 的消息栏中。加载并运行该程序，可以看到 HMI 的消息栏中打印出“Hello world!”。

注意

需要注意的是如果在同一个文件中定义不同的函数，则被调用的函数必须在调用函数之前定义，否则加载时会报出找不到被调用函数的错误。

2 变量与运算

2.1 常量

程序执行过程中不能修改其值的量称为常量。

ARL 支持四种类型的基本常量，分别是：

- 整型（int）
- 浮点型（double）
- 布尔型（bool）
- 字符串类型（string）

其中，整型常量可以使用十进制、二进制（数字后面跟 b）或者十六进制（数字后面跟 h）来表达；浮点型常量可以以小数形式或科学计数法来表达。

例如，下面都是合法的常量：

- 1 //十进制整型
- 10b //2 进制整型，值为 2
- 3eh //16 进制整型，值为 62
- 3.245 //浮点型
- 0.15e2 //科学计数法表达的浮点型，表示 0.15 乘以 10 的平方，值为 15
- true //布尔型，值为真
- false //布尔型，值为假

2.2 变量和名称

程序执行过程中可以修改其值的量称为变量。ARL 支持 6 种类型的基本变量，分别是：

- 整型 (int)
- 无符号整形 (uint)
- 字节型 (byte)
- 浮点型 (double)
- 布尔型 (bool)
- 字符串类型 (string)

使用变量前必须要对该变量进行声明，变量名为用户自定义，可由字母，下划线，数字的组合组成。变量名不能以数字开头。变量名不能命名为系统关键字。ARL 系统关键字参见附录 A ARL 关键字。

- 在使用一个变量之前需要首先声明它，变量声明语句的一般格式如下：

变量类型 变量名

或

变量类型 变量名 = 初值

程序示例：

```
int a = 1
```

该声明语句声明了一个 int 类型，名字为 a 的变量并赋予了初始值 1。

- 同一类型的变量可以在同一行中声明，格式为：

变量类型 变量名 = 初值，变量名 = 初值，……

程序示例：

int x = 1,y =3

该声明语句声明了两个 int 类型的变量，名字为 x 的变量并赋予了初始值 1，名字为 y 的变量并赋予了初始值 3。

- 如果在定义的变量之前加上 const 关键字，则表示该变量不允许被再次赋值。

程序示例：

const double pi = 3.1415926

pi = 3.4

则加载程序时会报错。

2.3 基本数据类型

ARL 支持 6 种类型的基本变量：

- 整型 (int)
- 无符号整形 (uint)
- 字节型 (byte)
- 浮点型 (double)
- 布尔型 (bool)
- 字符串类型 (string)

2.3.1 int(整型)

int 类型数据为一个 32bit 数，可以表达一个范围在-2^{31}~2^{31}-1 的整数。

程序示例：

int counter = 4

表示定义了一个整型变量 counter，且它的数值等于 4。

2.3.2 uint(无符号整型)

uint 类型数据为一个 32bit 数，可以表达一个范围在 0~2^{32}-1 的整数。

程序示例：

uint ae = 5

表示定义了一个无符号整型变量 ae，且它的数值等于 5。

2.3.3 byte(字节型)

byte 类型数据为一个 8bit 数，可以表达一个范围在 0~255 的整数。

程序示例：

byte b = ffh

表示定义了一个 byte 型变量 b，且它的数值等于 ffh，也就是十进制的 255。

2.3.4 double(浮点型)

double 类型数据为一个 64bit 数，可以表达一个范围在$-1.7*10^{-308}$~$1.7*10^{308}$的小数。

程序示例：

```
double scbd=1
```

表示定义了一个浮点型变量 scbd，且值等于 1。

double 类型数据存在精度问题。请尽量避免两个浮点数之间比较是否相等的表达式。

程序示例（错误用法）：

```
double data = 0
……
if(data == 0)
……
endif
```

程序示例（正确用法）：

```
double data = 0
……
if(abs(data) < 0.000001)
……
endif
```

2.3.5 bool(布尔型)

bool 型变量表示逻辑真或假，一个 bool 类型变量的值只能为 true 或者 false。

例如：

```
bool io16 = true
```

表示定义了一个 bool 型变量 io16，且值为 true。

2.3.6 string(字符串类型)

字符串类型数据表达了一个由 1 个或多个字符组成的字符串。

例如：

```
string name = "Vincent"
```

这条语句定义了一个字符串变量 name，并初始化为“Vincent”。

但是某些字符没有对应的单独符号，例如换行符，所以需要依赖以反斜杠“\”开头的转义字符来表达。

ARL 中支持的转义字符如下：

- \n 换行（LF），将当前位置移到下一行开头
- \r 回车（CR），将当前位置移到本行开头

- \t 水平制表（HT），跳到下一个 tab 位置
- \" 代表一个双引号字符
- \\ 代表一个反斜杠字符"\"

2.3.7 基本数据隐式类型转换

某些情况下，系统会对基本类型变量做隐式类型转换，允许进行隐式类型转换的类型如表 2-1 所示：

表 2-1 隐式类型转换表

from \ to	byte	int	double	bool
byte		√	√	√
int			√	√

表中画有"√"的项表示两种类型之间允许从左侧的类型隐式类型转换为上方的类型。如 byte 类型就允许隐式类型转换为 int 类型。

例如：

```
int a = 3
double b = 4
double c = a+b //系统将隐式地把 a 转换成 double 类型再与 b 相加。
int counter = 5
while(counter--)
……
Endwhile //系统会将 counter 隐式地转换成 bool 型数据。非 0 值会被转为 true，0 值会被转为 false。所以此段代码循环将会执行 5 次。
```

2.4 结构体类型（适用于六轴机器人）

ARL 中支持系统预定义的结构体类型，结构体类型是由基本类型组成的复合类型。

2.4.1 结构体变量的声明，初始化及引用

以 pos 类型为例，空间的任意一个点坐标由 x，y，z 3 个坐标分量组成，所以 pos 类型，也就是空间点坐标类型由 3 个 double 类型的子分量组成。

pos 类型的定义如下：

```
struct pos
{
double x
double y
double z
}
```

结构体类型变量的声明与基本类型变量的声明相同，格式为：

变量类型 变量名

例如：

```
pos a
```

声明了一个名为 a 的 pos 类型变量，如果我们用 print 指令打印 a 的值：

```
print a
```

系统将输出：

```
{9e+09,9e+09,9e+09}
```

逗号隔开的 3 个数字分别为 a 变量中每个分量的值，可见，默认情况下，pos 类型变量 a 的分量 x，y，z 分别被初始化为 9e+09，9e+09，9e+09。

可以在变量声明的同时对变量进行初始化。结构体变量初始化的格式有以下 2 种方式：

- 初始化结构体成员的前 n 个成员，格式为：

结构体类型 变量名 = {分量 1 的值，分量 2 的值，……分量 n 的值}

- 初始化结构体成员的某 1 个或几个成员：

结构体类型 变量名 = {分量名字 分量值，分量名字 分量值，……}

对于以上两种结构体变量初始化方式，未初始化的分量将被系统初始化为默认值。

程序举例如下：

```
pos a = {1,2}
print a //输出“{1,2,9e+09}”
pos b = {x 3,z 4}
print b //输出“{3,9e+09,4}”
```

使用成员操作符“.”访问一个结构体变量的成员分量，格式为：

结构体变量名.分量名

程序示例：

```
pos a
a.x = 1
a.y = 2
a.z = 3
double b = a.x
print a //输出“{1,2,3}”
print b //输出“1”
```

结构体支持嵌套，也就是说某个结构体类型的某个成员可能是另外一个结构体类型。考虑如下结构体类型 tool 的定义：

```
struct tool
{
frame t_frame
bool stationary
}
```

声明一个 tool 类型的变量并输出：

```
tool a
```

```
print a
```

系统将会输出：

```
{{0,0,0,0,0,0},false}
```

这里结构体变量将会以嵌套的花括号的形式输出。同样初始化结构体变量的时候也要以嵌套的花括号形式。

程序举例如下：

```
tool a = {{10,10,0,0,90,0},false}
print a //输出“{{10,10,0,0,90,0},false}”
print a.t_frame //输出“{10,10,0,0,90,0}”
print a.t_frame.x //输出“10”
tool b = {{x 10,b 90},false}
print b //输出“{{10,0,0,0,90,0},false}”
```

2.4.2 pos(空间点坐标)

描述

pos 结构体类型用于描述空间任意一点的坐标。

定义

```
struct pos
{
double x
double y
double z
}
```

成员

pos 结构体类型的成员详见表 2-2。

表 2-2 pos 结构体类型的成员

名称	说明
x	类型：double
	空间一点坐标的 x 分量，单位毫米
y	类型：double
	空间一点坐标的 y 分量，单位毫米
z	类型：double
	空间一点坐标的 z 分量，单位毫米

用法举例

```
pos p1 = {0,0,0}
print p1 //输出“{0,0,0}”
```

```
p1.x = p1.x +50 //以对结构体成员进行运算
print p1 //输出“{50,0,0}”
```

2.4.3 frame(坐标系)

描述

frame 结构体类型用于描述空间两个直角坐标系之间的转换关系。

定义

```
struct frame
{
double x
double y
double z
double a
double b
double c
}
```

成员

frame 结构体类型的成员详见表 2-3。

表 2-3 frame 结构体类型的成员

名称	说明
x	类型：double
	转换后坐标系原点在原坐标系中坐标的 x 分量，单位毫米
y	类型：double
	转换后坐标系原点在原坐标系中坐标的 y 分量，单位毫米
z	类型：double
	转换后坐标系原点在原坐标系中坐标的 z 分量，单位毫米
a	类型：double
	转换后坐标系姿态在原坐标系中欧拉角表达的 a 分量，单位度
b	类型：double
	转换后坐标系姿态在原坐标系中欧拉角表达的 b 分量，单位度
c	类型：double
	转换后坐标系姿态在原坐标系中欧拉角表达的 c 分量，单位度


注意

欧拉角按照旋转所绕轴的次序不同，共有 12 种不同的欧拉角表达方式，我们这里采用的是 ZYX 型欧拉角，即从初始坐标系姿态转换为变换坐标系姿态的顺序为先绕 z 轴旋转 a 角度，再绕新的 y 轴旋转 b 角度，最后绕新的 x 轴旋转 c 角度。

用法举例

frame f = {10,10,0,0,90,0}

2.4.4 pose(笛卡尔目标点)

描述

pose 结构体类型使用笛卡尔坐标的方式来描述机器人各轴以及外轴的位置。

定义

```
struct pose
{
double x
double y
double z
double a
double b
double c
int cfg
int turn
double ej1
double ej2
double ej3
double ej4
double ej5
double ej6
}
```

成员

pose 结构体类型的成员详见表 2-4。

表 2-4 pose 结构体类型的成员

名称	说明
x	类型：double
	机器人 TCP 点相对当前工件坐标系下的 x 方向分量，单位毫米
y	类型：double
	机器人 TCP 点相对当前工件坐标系下的 y 方向分量，单位毫米
z	类型：double
	机器人 TCP 点相对当前工件坐标系下的 z 方向分量，单位毫米
a	类型：double
	机器人工具姿态在当前工件坐标系中欧拉角表达的 a 分量，单位度
b	类型：double

名称	说明
	机器人工具姿态在当前工件坐标系中欧拉角表达的 b 分量，单位度
c	类型：double
	机器人工具姿态在当前工件坐标系中欧拉角表达的 c 分量，单位度
cfg	类型：int
	机器人轴配置。由于机器人可能存在几种不同的方式使 TCP 到达同一个位姿，所以这里需要通过 cfg 参数（取值 0-7，代表可能的 8 种方式，参考图 2-1，其中，“beta”指代五轴的角度）来指定其中一种方式，从而唯一确定一组机器人轴位置。参数取值及位置关系见表 2-5 和图 2-1.所示。
turn	类型：int
	配合 cfg 用于确定反解的轴位置。这里只用到 turn 的低 6 个 bit 位，bit0 表示 1 轴，bit1 表示 2 轴，以此类推。 ■ 当 turn 值为-1 时，表示自动选取距离起点最近的解； ■ 当 turn 值为 1 时，表示该轴选小于 0 的解； ■ 当 turn 值为 0 时，表示该轴选大于 0 的解。 当轴的运动范围大于 360 度时可能会用到这个参数辅助选解，其他大部分情况下都不需要设置该值。
ej1~ej6	类型：double
	外 1 轴~外 6 轴的位置，直线轴单位为 mm，旋转轴单位为度

表 2-5 cfg 取值及位姿关系

cfg 取值	相对于轴 1 的腕中心	相对于大臂的腕中心	5 轴角度
0	在右侧	在右侧	正
1	在右侧	在右侧	负
2	在右侧	在左侧	正
3	在右侧	在左侧	负
4	在左侧	在左侧	正
5	在左侧	在左侧	负
6	在左侧	在右侧	正
7	在左侧	在右侧	负

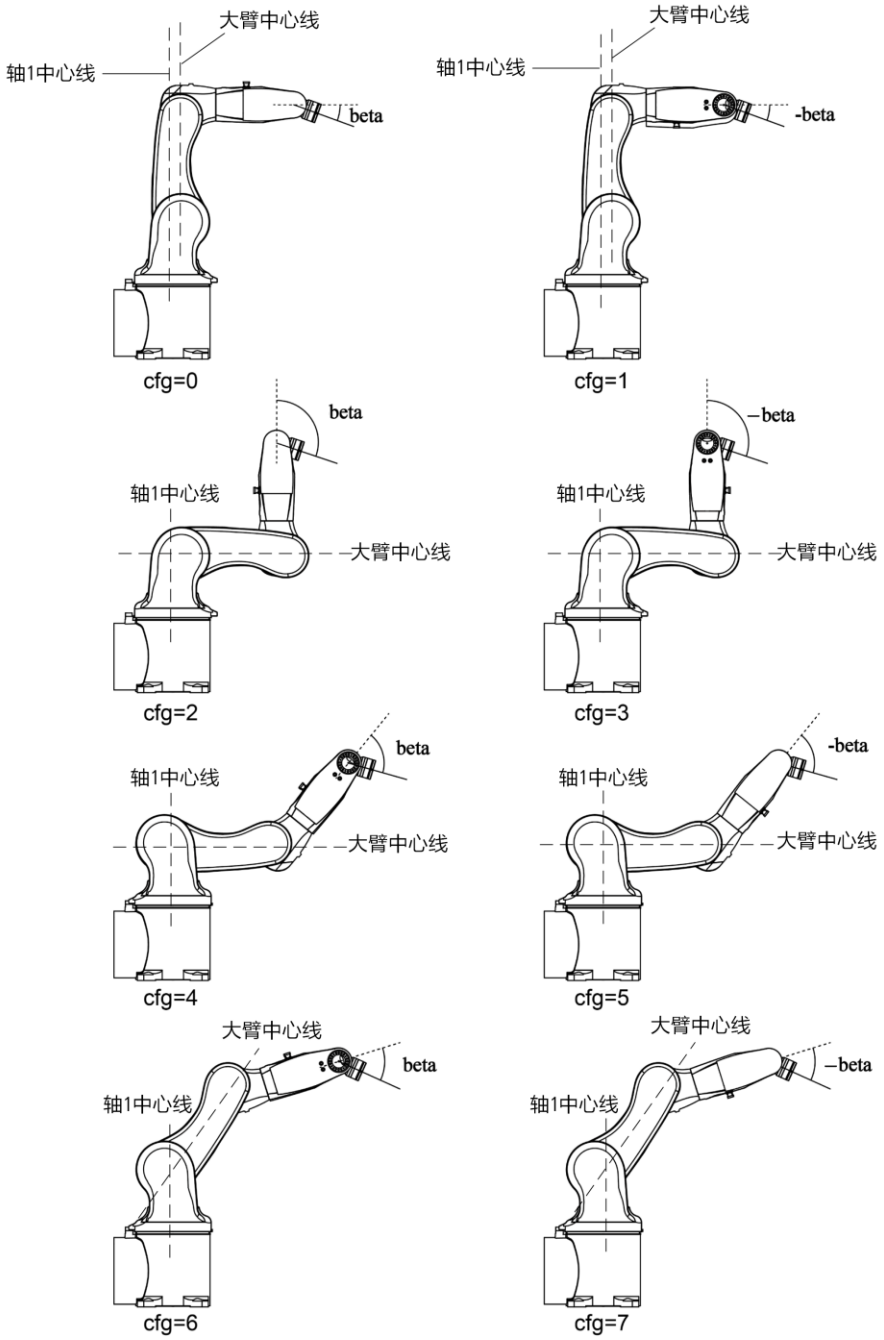


图 2-1 cfg 参数值姿态说明

注意

欧拉角定义参见第 2.4.3 章节。

用法举例

```
pose p1 = {x 0,y 10,z 15,a 0,b 90,c 0,cfg 0,ej1 20}
pose p2 = {0,-10,15,0,90,0,0,-1,10}
```

2.4.5 joint(轴目标点)

描述

joint 结构体类型用于直接描述机器人各轴以及外轴的位置。

定义

```
struct joint
{
double j1
double j2
double j3
double j4
double j5
double j6
double ej1
double ej2
double ej3
double ej4
double ej5
double ej6
}
```

成员

joint 结构体类型的成员详见表 2-6。

表 2-6 joint 结构体类型的成员

名称	说明
j1~j6	类型：double
	机器人 1 轴~6 轴的轴位置，单位度
ej1~ej6	类型：double
	外 1 轴~外 6 轴的位置，直线轴单位为 mm，旋转轴单位为度

用法举例

```
joint p1 = {j1 0,j3 15,j5 0,ej1 20}
print p1 //输出“{0,9e+09,15, 9e+09,0,9e+09,20,9e+09,9e+09,9e+09,9e+09,9e+09}”
joint p2 = {0,10,20,30,40,50,10}
print p2 //输出“{0, 10,20,30,40,50,10,9e+09,9e+09,9e+09,9e+09,9e+09}”
```

注意

当所有轴位置均指定时，joint 中的参数名称可缺省，但必须全部缺省。例如以下的写法就是错误的：j:{0,0,90,0,0,0,ej1 0}。

2.4.6 tool(工具)

描述

tool 结构体类型用于描述一个工具。

定义

```
struct tool
{
frame t_frame
bool stationary
}
```

成员

tool 结构体类型的成员详见表 2-7。

表 2-7 tool 结构体类型的成员

名称	说明
t_frame	类型：frame
	在工具上定义的工具坐标系
stationary	类型：bool
	■ false：指定该工具为安装在机器人法兰末端的工具。此时工具坐标系参照法兰坐标系定义 ■ true：指定该工具为外部的固定工具。此时工具坐标系参照世界坐标系定义

用法举例

```
frame f = {10,10,0,0,90,0}
tool t1
t1.t_frame = f
t1.stationary = false
print t1 //输出“{{10,10,0,0,90,0},false}”
tool t2 = {{0,10,20,0,0,0},true}
print t2 //输出“{{0,10,20,0,0,0},true}”
```

2.4.7 wobj(工件坐标系)

描述

wobj 结构体类型用于描述一个工件坐标系。

定义

```
struct wobj
{
frame w_frame
bool robhold
string mu_name
}
```

成员

wobj 结构体类型的成员详见表 2-8。

表 2-8 wobj 结构体类型的成员

名称	说明
w_frame	类型：frame
	工件坐标系
robhold	类型：bool
	■ false：指定工件坐标系参照世界坐标系定义 ■ true：指定该工件由机器人抓取。此时工件坐标系参照法兰坐标系定义
mu_name	string
	机械单元名称

用法举例

```
frame f = {10,10,0,0,90,0}
wobj w1
w1.w_frame = f
print w1 //输出“{{10,10,0,0,90,0},false}”
wobj w2 = {{0,10,20,0,0,0},false}
print w2 //输出“{{0,10,20,0,0,0},false}”
```

2.4.8 weavedata(摆动图形参数)

描述

weavedata 结构体类型用于描述摆动的图形参数。

定义

```
struct weavedata
{
enum weaveshape
double frequency
double amplitude
double dwell_left
double dwell_right
double dwell_middle
bool track
double swing_angle
double radius
enum weaverotaxis
rotation_angle
bool vibrat
}
```

成员

weavedata 结构体类型的成员详见表 2-9。

表 2-9 weavedata 结构体类型的成员

名称	说明
weaveshape	类型：enum
	枚举类型，包含了所有叠加轨迹支持的图形类型，详见*第 2.6.8 章节*
frequency	类型：double
	摆动的循环频率，单位为赫兹
amplitude	类型：double
	摆动振幅，单位毫米
dwell_left	类型：double
	左侧停留长度，循环类型为频率时，该参数表示停留时长，单位秒，循环类型为波长时，该参数表示停留距离，单位毫米。该参数的最小值为 0
dwell_right	类型：double
	右侧停留长度，循环类型为频率时，该参数表示停留时长，单位秒，循环类型为波长时，该参数表示停留距离，单位毫米。该参数的最小值为 0
dwell_middle	类型：double
	中心停留长度，循环类型为频率时，该参数表示停留时长，单位秒，循环类型为波长时，该参数表示停留距离，单位毫米。该参数的最小值为 0
track	类型：bool
	是否进行焊缝跟踪，true 表示跟踪，false 表示不跟踪，可缺省，默认值为 false
swing_angle	类型：double
	空间三角摆和空间 v 型摆的摆动角度，单位度
radius	类型：double
	图形半径，单位毫米
weaverotaxis	类型：enum
	该枚举类型数据包含了叠加轨迹支持的摆动平面偏转所参考的坐标轴
Vibrat	类型：bool
	是否开启起振功能。
rotation_angle	类型：double
	摆动平面的偏转角度，单位度

用法举例

```
weavedata weavedatalin1 = {2,15,1,1,0,true}
weavedata weavedatalin2 = {2,15}
```

2.4.9 speed(速度)

描述

speed 结构体类型用于描述一条运动指令的速度参数。

定义

```
struct speed
{
double per
double tcp
double ori
double exj
double exl
}
```

成员

speed 结构体类型的成员详见表 2-10。

表 2-10 speed 结构体类型的成员

名称	说明
per	类型：double
	速度百分比，ptp 和 movej 指令使用该速度参数，表示轴最大速度百分比，取值范围 0.001~100
tcp	类型：double
	TCP 点移动速度，lin 和 cir 指令使用该速度参数，表示 TCP 点平动速度，单位毫米/秒，不同机型取值范围如下： ■ AIR3/4/6L/7L/10/20：TCP=2500 ■ AIR50、165：TCP=1300
ori	类型：double
	工具坐标系姿态转动速度，lin 和 cir 指令使用该速度参数，表示 TCP 点转动速度，单位度/秒，不同机型取值范围如下： ■ AIR3/4/6L/7L/10：ORI=500 ■ AIR20：ORI=500 ■ AIR50、165：ORI=250
exj	类型：double
	旋转外轴速度。当有旋转外轴时，移动时使用该速度参数，单位度/秒
exl	类型：double
	直线外轴速度。当有直线外轴时，移动时使用该速度参数，单位毫米/秒

用法举例

```
speed v = {per 10}
print v //输出“{10,9e+09,9e+09,9e+09,9e+09}”
```

2.4.10 slip(平滑参数)

描述

slip 结构体类型用于描述一条运动指令的平滑参数。

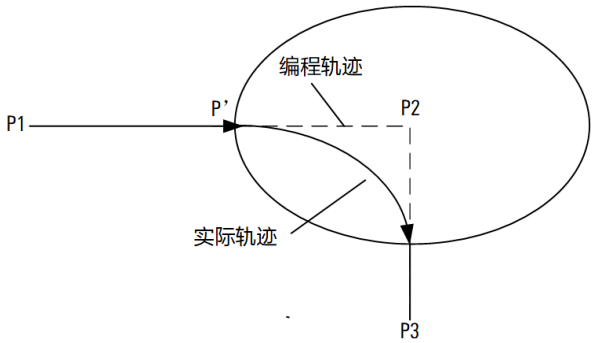
定义

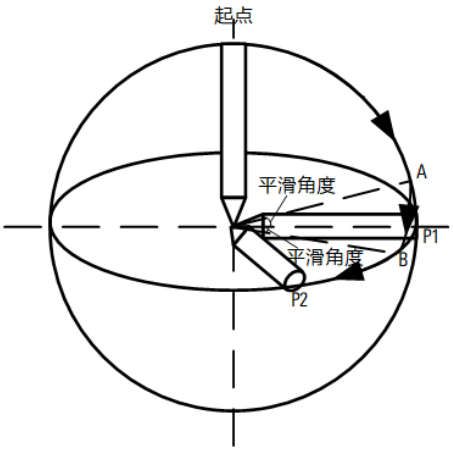
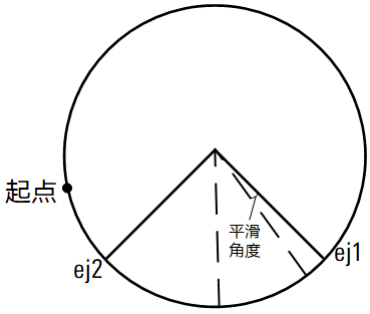
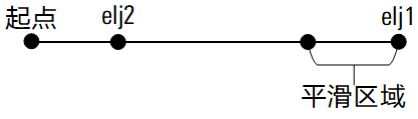
```
struct slip
{
double perdis
double pdis
double odis
double ejdis
double eldis
}
```

成员

slip 结构体类型的成员详见表 2-11。

表 2-11 slip 结构体类型的成员

名称	说明
perdis	类型：double
	Percentage distance，所有运动指令均可以使用此平滑参数。当此平滑参数生效时，其他平滑参数将会无效。表示轨迹在可平滑段开启平滑的百分比，100%为完全平滑，0%为不平滑
pdis	类型：double
	pos distance，表示 TCP 距离目标点多少毫米（平滑区域半径）时开始平滑，单位毫米。 如下图所示，编程轨迹为“P1→P2→P3”，当 TCP 进入平滑区域后，沿着平滑的圆弧轨迹运动，离开平滑区域后沿直线运动到 P3。  提示 平滑距离（直线 P'P2）不能大于轨迹长度（直线 P1P2）的一半，如果大于某侧轨迹长度一半，则该侧轨迹的平滑距离减小为轨迹长度的一半。

名称	说明
odis	类型：double
	orientation distance，表示姿态距离目标点多少角度（平滑角度）时开始平滑，单位度。 如下图所示，工具的编程轨迹为“起点→P1→P2”，当工具沿圆弧运动到 A 点（即平滑角度内）时，在平滑区域内开始平滑运动到 B 点，再从 B 点运动到 P2。  提示 平滑角度不能大于轨迹起、终点夹角的一半，如果大于某侧轨迹起、终点夹角的一半，则该侧轨迹的平滑角度减小为该侧轨迹起、终点夹角的一半。
ejdis	类型：double
	ex-joint distance，有旋转型外轴的运动指令使用这项平滑参数，表示最慢的外轴距离目标点多少度（平滑角度）时开始平滑，外轴为旋转轴时，单位为度；外轴为直线轴时，单位为毫米。 如下图所示，外轴的编程轨迹为“起点→ej1→ej2”，当外轴旋转到平滑角度范围内开始平滑运动，并开始向着 ej2 点的方向平滑运动。 
eldis	类型：double
	exlinearjoint distance，有移动型外轴的运动指令使用这项平滑参数，表示最慢的外轴距离目标点多少毫米（平滑距离）时开始平滑，外轴为旋转轴时，单位为度；外轴为直线轴时，单位为毫米。 如下图所示，外轴的编程轨迹为“起点→elj1→elj2”，当外轴直线运动到平滑区域内开始平滑运动，并开始向着 ej2 点方向平滑运动。 

用法举例

slip s = {100,2,3,4,5,5}

注意

- 当 pdis 参数小于等于 0 时，表示使用它的运动语句不平滑。
- 请尽量在初始化结构体变量时将结构体的所有成员都写全，以防止由于用户的疏忽导致实际的运动轨迹或者运动速度和用户预期不符，进而导致一些不必要的损失或伤害。
- perdis 优先级最高，在有 perdis 的情况下，其余参数无效；当输入的 pdis，odis，ejdis 生效时，平滑点更靠近目标点的值生效。

2.4.11 jvel(关节速度)

描述

jvel 结构体类型用于描述机器人各关节速度值。

定义

```
struct jvel
{
double jv1
double jv2
double jv3
double jv4
double jv5
double jv6
double ev1
double ev2
double ev3
double ev4
double ev5
double ev6
}
```

成员

jvel 结构体类型的成员详见表 2-12。

表 2-12 jvel 结构体类型的成员

名称	说明
jv1~jv6	类型：double
	机器人 1 轴~6 轴的轴速度，单位：rad/s
ev1~ev6	类型：double
	机器人外 1 轴~外 6 轴的轴速度，单位：rad/s

用法举例

```
jvel jvel1 = {jv1 1, jv3 2, ev1 3}
print jvel1 //输出“{1,9e+09,2,9e+09,9e+09,9e+09,3,9e+09,9e+09,9e+09,9e+09,9e+09,9e+09}”
jvel jvel2 = {0,1,2,3,4,5,1}
```

print jvel2 //输出“{0,1,2,3,4,5,1,9e+09,9e+09,9e+09,9e+09,9e+09}”

2.4.12 Centroid_Pos(工具负载质心)

描述

Centroid_Pos 结构体类型用于描述工具负载质心的位置。

格式

```
struct Centroid_Pos
{
double x
double y
double z
}
```

参数

Centroid_Pos 指令的参数详见表 2-13。

表 2-13 Centroid_Pos 指令的参数

名称	说明
x	类型：double
	工具负载质心在参考坐标系中坐标的 x 分量，单位毫米。
y	类型：double
	工具负载质心在参考坐标系中坐标的 y 分量，单位毫米。
z	类型：double
	工具负载质心在参考坐标系中坐标的 z 分量，单位毫米。

用法举例

```
Centroid_Pos cen_pos = {10, 20, 30}
print cen_pos //输出“{10, 20, 30}”
```

2.4.13 Inertia_Tensor(工具负载惯性主轴方向)

描述

Inertia_Tensor 结构体类型用于描述工具负载的惯性参数。

格式

```
struct Inertia_Tensor
{
double Ixx
double Ixy
```

double Ixz
double Iyy
double Iyz
double Izz
}

参数

Inertia_Tensor 指令的参数详见表 2-15。

表 2-14 Inertia_Tensor 指令的参数

名称	说明
Ixx	类型：double
	负载惯性矩阵的 xx 分量，单位 g*mm^2。
Ixy	类型：double
	负载惯性矩阵的 xy 分量，单位 g*mm^2。
Ixz	类型：double
	负载惯性矩阵的 xz 分量，单位 g*mm^2。
Iyy	类型：double
	负载惯性矩阵的 yy 分量，单位 g*mm^2。
Iyz	类型：double
	负载惯性矩阵的 yz 分量，单位 g*mm^2。
Izz	类型：double
	负载惯性矩阵的 zz 分量，单位 g*mm^2。

用法举例

Inertia_Tensor I_T = {10, 20, 30,40,50,60}
print I_T //输出“{10, 20, 30,40,50,60}”

2.4.14 ToolInertiaPara(工具负载类型)

描述

ToolInertiaPara 结构体类型包含了负载质量、质心、惯性主轴方向及转动惯量。

格式

struct ToolInertiaPara
{

```
double m
Centroid_Pos centroidpos
Inertia_Tensor inertiatensor
}
```

参数

ToolInertiaParah 指令的参数详见表 2-15。

表 2-15 ToolInertiaPara 指令的参数

名称	说明
m	类型：double
	工具负载质量，单位 g。
centroidpos	Centroid_Pos
	工具负载质心位置
inertiatensor	Inertia_Tensor
	描述工具负载的惯性参数

用法举例

```
ToolInertiaPara tip //定义工具负载
tip.m = 30 //质量
tip.centroid_pos = {15, 25, 100} //定义质心位置
tip. Inertia_Tensor = {10, 20, 30,40,50,60}//定义惯性主轴方向
print tip //输出“{30,{15,25,100},{10, 20, 30,40,50,60}}”
```

2.5 结构体类型（适用于 SCARA 机器人）

ARL 中支持系统预定义的结构体类型，结构体类型是由基本类型组成的复合类型。

2.5.1 结构体变量的声明，初始化及引用

以 pos 类型为例，空间的任意一个点坐标由 x，y，z 3 个坐标分量组成，所以 pos 类型，也就是空间点坐标类型由 3 个 double 类型的子分量组成。

pos 类型的定义如下：

```
struct pos
{
double x
double y
double z
}
```

结构体类型变量的声明与基本类型变量的声明相同，格式为：

变量类型 变量名

例如：

pos a

声明了一个名为 a 的 pos 类型变量，如果我们用 print 指令打印 a 的值：

print a

系统将输出：

{9e+09,9e+09,9e+09}

逗号隔开的 3 个数字分别为 a 变量中每个分量的值，可见，默认情况下，pos 类型变量 a 的分量 x，y，z 分别被初始化为 9e+09，9e+09，9e+09。

可以在变量声明的同时对变量进行初始化。结构体变量初始化的格式有以下 2 种方式：

- 初始化结构体成员的前 n 个成员，格式为：

结构体类型 变量名 = {分量 1 的值，分量 2 的值，……分量 n 的值}

- 初始化结构体成员的某 1 个或几个成员：

结构体类型 变量名 = {分量名字 分量值，分量名字 分量值，……}

对于以上两种结构体变量初始化方式，未初始化的分量将被系统初始化为默认值。

程序举例如下：

```
pos a = {1,2}
print a //输出“{1,2,9e+09}”
pos b = {x 3,z 4}
print b //输出“{3,9e+09,4}”
```

使用成员操作符“.”访问一个结构体变量的成员分量，格式为：

结构体变量名.分量名

程序示例：

```
pos a
a.x = 1
a.y = 2
a.z = 3
double b = a.x
print a //输出“{1,2,3}”
print b //输出“1”
```

结构体支持嵌套，也就是说某个结构体类型的某个成员可能是另外一个结构体类型。考虑如下结构体类型 tool 的定义：

```
struct tool
{
frame t_frame
bool stationary
}
```

声明一个 tool 类型的变量并输出：

```
tool a
print a
```

系统将会输出：

```
{{0,0,0,0,0,0},false}
```

这里结构体变量将会以嵌套的花括号的形式输出。同样初始化结构体变量的时候也要以嵌套的花括号形式。

程序举例如下：

```
tool a = {{0,0,0,0,0,0},false}
print a //输出“{{0,0,0,0,0,0},false}”
print a.t_frame //输出“{0,0,0,0,0,0}”
print a.t_frame.x //输出 “0”
tool b = {{x 10},false}
print b //输出“{{10,0,0,0,0,0},false}”
```

2.5.2 pos(空间点坐标)

描述

pos 结构体类型用于描述空间任意一点的坐标。

定义

```
struct pos
{
double x
double y
double z
}
```

成员

pos 结构体类型的成员详见表 2-16。

表 2-16 pos 结构体类型的成员

名称	说明
x	类型：double
	空间一点坐标的 x 分量，单位毫米
y	类型：double
	空间一点坐标的 y 分量，单位毫米
z	类型：double
	空间一点坐标的 z 分量，单位毫米

用法举例

```
pos p1 = {0,0,0}
print p1 //输出“{0,0,0}”
p1.x = p1.x +50
print p1 //输出“{50,0,0}”
```

2.5.3 frame(坐标系)

描述

frame 结构体类型用于描述空间两个直角坐标系之间的转换关系。

定义

```
struct frame
{
double x
double y
double z
double a
double b
double c
}
```

成员

frame 结构体类型的成员详见表 2-17。

表 2-17 frame 结构体类型的成员

名称	说明
x	类型：double
	转换后坐标系原点在原坐标系中坐标的 x 分量，单位毫米
y	类型：double
	转换后坐标系原点在原坐标系中坐标的 y 分量，单位毫米
z	类型：double
	转换后坐标系原点在原坐标系中坐标的 z 分量，单位毫米
a	类型：double
	转换后坐标系姿态在原坐标系中欧拉角表达的 a 分量，单位度
b	类型：double
	转换后坐标系姿态在原坐标系中欧拉角表达的 b 分量，单位度
c	类型：double
	转换后坐标系姿态在原坐标系中欧拉角表达的 c 分量，单位度


注意

欧拉角按照旋转所绕轴的次序不同，共有 12 种不同的欧拉角表达方式，我们这里采用的是 ZYX 型欧拉角，即从初始坐标系姿态转换为变换坐标系姿态的顺序为先绕 z 轴旋转 a 角度，再绕新的 y 轴旋转 b 角度，最后绕新的 x 轴旋转 c 角度。

用法举例

```
frame f = {10,10,0,0,0,0}
print f //输出“{10,10,0,0,0,0}”
```

2.5.4 pose(笛卡尔目标点)

描述

pose 结构体类型使用笛卡尔坐标的方式来描述机器人各轴以及外轴的位置。

定义

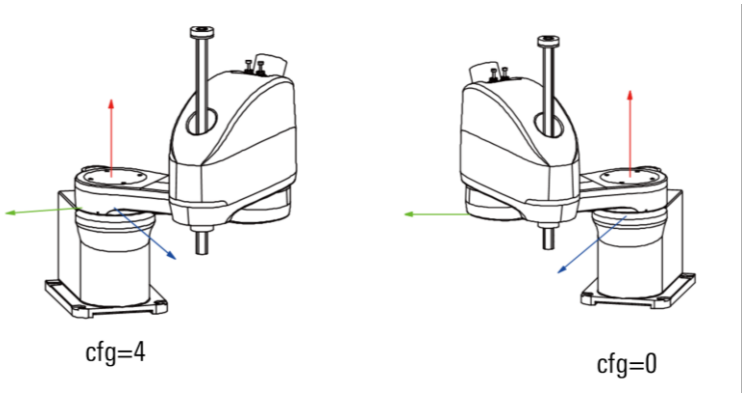
```
struct pose
{
double x
double y
double z
double a
double b
double c
int cfg
int turn
double ej1
double ej2
double ej3
double ej4
double ej5
double ej6
}
```

成员

pose 结构体类型的成员详见表 2-18。

表 2-18 pose 结构体类型的成员

名称	说明
x	类型：double
	机器人 TCP 点相对当前工件坐标系下的 x 方向分量，单位毫米
y	类型：double
	机器人 TCP 点相对当前工件坐标系下的 y 方向分量，单位毫米
z	类型：double

名称	说明
	机器人 TCP 点相对当前工件坐标系坐标的 z 分量，单位毫米
a	类型：double
	机器人工具姿态在当前工件坐标系中欧拉角表达的 a 分量，单位度
b	类型：double
	机器人工具姿态在当前工件坐标系中欧拉角表达的 b 分量，单位度
c	类型：double
	机器人工具姿态在当前工件坐标系中欧拉角表达的 c 分量，单位度
cfg	类型：int
	机器人轴配置。由于机器人可能通过左手系/右手系两种方式使 TCP 到达同一个位姿，所以这里需要通过 cfg 参数（参考图 2-2，4 表示左手系，0 表示右手系）来指定其中一种方式，从而唯一确定一组机器人轴位置  图 2-2 cfg 参数值姿态说明
turn	类型：int
	配合 cfg 用于确定反解的轴位置。当 turn 值为-1 时，表示自动选取距离起点最近的解；当某个位值为 1 时，表示该轴选小于 0 的解；当某个位值为 0 时，表示该轴选大于 0 的解。当轴的运动范围大于 360 度时可能会用到这个参数辅助选解，其他大部分情况下都不需要设置该值，取默认值-1 即可
ej1~ej6	类型：double
	外 1 轴~外 6 轴的位置，直线轴单位为 mm，旋转轴单位为度


注意

欧拉角定义参见第 2.5.3 章节。

用法举例

```
pose p1 = {x 266,y 88,z -50,a -34,b 90,c 0,cfg 0,ej1 20}
print p1 //输出“{266, 88, -50, -34,90,0,0,-1,20,9e+09,9e+09,9e+09,9e+09,9e+09}”
pose p2 = {308, 52, -50, -33,90,0,0,-1,10}
print p2 //输出“{308, 52, -50, -33,90,0,0,-1,10,9e+09,9e+09,9e+09,9e+09,9e+09}”
```

2.5.5 joint(轴目标点)

描述

joint 结构体类型用于直接描述机器人各轴以及外轴的位置。

定义

```
struct joint
{
double j1
double j2
double j3
double j4
double ej1
double ej2
double ej3
double ej4
double ej5
double ej6
}
```

成员

joint 结构体类型的成员详见表 2-19。

表 2-19 joint 结构体类型的成员

名称	说明
j1~j4	类型：double
	机器人 1 轴~4 轴的轴位置，直线轴单位为 mm，旋转轴单位为度
ej1~ej6	类型：double
	外 1 轴~外 6 轴的位置，直线轴单位为 mm，旋转轴单位为度

用法举例

```
joint j1={j1 0, j2 0, j3 -10, j4 0}
print to:hmi,argtoprint:j1 //输出“{0,0，-10,0,9e+9, 9e+9, 9e+9, 9e+9, 9e+9, 9e+9, 9e+9, 9e+9}”
```


注意

当所有轴位置均指定时，joint 中的参数名称可缺省，但必须全部缺省。例如以下的写法就是错误的：j:{0,0,0,0,0,0,ej1 0}。

2.5.6 tool(工具)

描述

tool 结构体类型用于描述一个工具。

定义

```
struct tool
{
frame t_frame
bool stationary
}
```

成员

tool 结构体类型的成员详见表 2-20。

表 2-20 tool 结构体类型的成员

名称	说明
t_frame	类型：frame
	在工具上定义的工具坐标系
stationary	类型：bool
	■ false：指定该工具为安装在机器人法兰末端的工具。此时工具坐标系参照法兰坐标系定义 ■ true：指定该工具为外部的固定工具。此时工具坐标系参照世界坐标系定义

用法举例

```
frame f = {10,10,0,0,0,0}
tool t1
t1.t_frame = f
t1.stationary = false
print t1 //输出“{{10,10,0,0,0,0},false}”
tool t2 = {{0,10,0,0,0,0},true}
print t2 //输出“{{0,10,0,0,0,0},true}”
```

2.5.7 wobj(工件坐标系)

描述

wobj 结构体类型用于描述一个工件坐标系。

定义

```
struct wobj
{
frame w_frame
bool robhold
string mu_name
}
```

成员

wobj 结构体类型的成员详见表 2-21。

表 2-21 wobj 结构体类型的成员

名称	说明
w_frame	类型：frame
	工件坐标系
robhold	类型：bool
	■ false：指定工件坐标系参照世界坐标系定义 ■ true：指定该工件由机器人抓取。此时工件坐标系参照法兰坐标系定义
mu_name	string
	机械单元名称

用法举例

```
frame f = {10,10,0,0,0,0}
wobj w1
w1.w_frame = f
print w1 //输出“{{10,10,0,0,0,0},false}”
wobj w2 = {{0,10,0,0,0,0},false}
print w2 //输出“{{0,10,0,0,0,0},false}”
```

2.5.8 speed(速度)

描述

speed 结构体类型用于描述一条运动指令的速度参数。

定义

```
struct speed
{
double per
double tcp
double ori
double exj
double exl
}
```

成员

speed 结构体类型的成员详见表 2-22。

表 2-22 speed 结构体类型的成员

名称	说明
per	类型：double
	速度百分比，ptp 和 movej 指令使用该速度参数，表示轴最大速度百分比，取值范围 0.001~100
tcp	类型：double

名称	说明
	TCP 点移动速度，lin 和 cir 指令使用该速度参数，表示 TCP 点平动速度，单位毫米/秒
ori	类型：double
	工具坐标系姿态转动速度，lin 和 cir 指令使用该速度参数，表示 TCP 点转动速度，单位度/秒
exj	类型：double
	旋转外轴速度。当有旋转外轴时，移动时使用该速度参数，单位度/秒
exl	类型：double
	直线外轴速度。当有直线外轴时，移动时使用该速度参数，单位毫米/秒

用法举例

```
speed v = {per 10}
print v //输出“{10,9e+09,9e+09,9e+09,9e+09}”
```

2.5.9 slip(平滑参数)

描述

slip 结构体类型用于描述一条运动指令的平滑参数。

定义

```
struct slip
{
double pdis
double ejdis
double eldis
double odis
double perdis
}
```

成员

slip 结构体类型的成员详见表 2-23。

表 2-23 slip 结构体类型的成员

名称	说明
pdis	类型：double
	pos distance，lin 和 cir 指令使用这项平滑参数，表示 TCP 距离目标点多少 mm 时开始平滑，单位毫米
ejdis	类型：double
	ex-joint distance，有旋转型外轴的运动指令使用这项平滑参数，表示最慢的外轴距离目标点多少度时开始平滑，外轴为旋转轴时，单位为度；外轴为直线轴时，单位为毫米
eldis	类型：double

名称	说明
	exlinearjoint distance，有移动型外轴的运动指令使用这项平滑参数，表示最慢的外轴距离目标点多少毫米时开始平滑，外轴为旋转轴时，单位为度；外轴为直线轴时，单位为毫米
odis	类型：double
	orientation distance，lin 和 cir 指令使用这项平滑参数，表示姿态距离目标点多少角度时开始平滑，单位度
perdis	类型：double
	Percentage distance, 所有运动指令均可以使用此平滑参数。当此平滑参数生效时，其他平滑参数将会无效。表示轨迹在可平滑段开启平滑的百分比，100%为完全平滑，0%为不平滑

用法举例

```
slip s = {100,2,3,4,5,5}
print s //输出“{100,2,3,4,5,5}”
```

注意

- 当 pdis 参数小于等于 0 时，表示使用它的运动语句不平滑。
- 请尽量在初始化结构体变量时将结构体的所有成员都写全，以防止由于用户的疏忽导致实际的运动轨迹或者运动速度和用户预期不符，进而导致一些不必要的损失或伤害。
- perdis 优先级最高，在有 perdis 的情况下，其余参数无效；当输入的 pdis，odis，ejdis 生效时，平滑点更靠近目标点的值生效。

2.5.10 jvel(关节速度)

描述

jvel 结构体类型用于描述机器人各关节速度值。

定义

```
struct jvel
{
double jv1
double jv2
double jv3
double jv4
double ev1
double ev2
double ev3
double ev4
double ev5
double ev6
}
```

成员

jvel 结构体类型的成员详见表 2-24。

表 2-24 jvel 结构体类型的成员

名称	说明
jv1~jv4	类型：double
	机器人 1 轴~4 轴的轴速度，单位：rad/s
ev1~ev6	类型：double
	机器人外 1 轴~外 6 轴的轴速度，单位：rad/s

用法举例

```
jvel jvel1 = {jv1 1, jv3 2, ev1 3}
print jvel1 //输出“{1,9e+09,2,9e+09,9e+09,9e+09,3,9e+09,9e+09,9e+09,9e+09,9e+09}”
jvel jvel2 = {0,1,2,3,4}
print jvel2 //输出“{0,1,2,3,4, 9e+09,9e+09,9e+09,9e+09,9e+09,9e+09,9e+09}”
```

2.5.11 control(门型动作的控制参数)

描述

control 结构体类型用于描述门型动作直升和直降阶段的速度、加速度、减速度。

定义

```
struct control
{
double rising_vel
double rising_acc
double rising_dec
double falling_vel
double falling_acc
double falling_dec
}
```

成员

control 结构体类型的成员详见表 2-25。

表 2-25 control 结构体类型的成员

名称	说明
rising_vel	类型：double
	直升阶段最大速度百分比，不指定则参考当前速度参数，指定则代表最大轴速度的百分比。
rising_acc	类型：double
	直升阶段最大加速度百分比，不指定则参考当前加速度参数，指定则代表最大轴加速度百分比。
rising_dec	类型：double

名称	说明
	直升阶段最大减速度百分比，不指定则参考当前加速度参数，指定则代表最大轴加速度百分比。
falling_vel	类型：double
	直降阶段最大速度百分比，不指定则参考当前速度参数，指定则代表最大轴加速度的百分比。
falling_acc	类型：double
	直降阶段最大加速度百分比，不指定则参考当前加速度参数，指定则代表最大轴加速度的百分比。
falling_dec	类型：double
	直降阶段最大减速度百分比，不指定则参考当前加速度参数，指定则代表最大轴加速度的百分比。

用法举例

```
control ctr1 = { rising_vel 0, rising_acc 0, rising_dec 0, falling_vel 0, falling_acc 0, falling_dec 100}
print ctr1 //输出”{0, 0, 0, 0, 0, 100}”
```

2.6 枚举类型

ARL 支持系统预定义枚举类型，枚举类型变量的值只能从固定的几个值中选取。

例如如下枚举类型定义：

```
enum $VEL_PROFILE
{
T_type,
S_type,
O_type
}
```

该枚举类型定义了速度曲线的不同类型，这里 T_type，S_type，O_type 分别代表 T 型速度曲线、S 型速度曲线和 O 型速度曲线。枚举类型是一种特殊的整型，T_type，S_type，O_type 实际分别对应整型的 0，1，2。

引用枚举变量的格式为：

枚举类型名::枚举常量名

例如：

```
print $VEL_PROFILE::S_TYPE //输出“S_TYPE”
```

通常情况下，程序中可以省略枚举类型名，直接使用枚举常量名来引用枚举常量。

例如：

```
print S_TYPE //输出“S_TYPE”
```

但是当程序中定义了与枚举常量名相同的变量名时，系统将优先将该名字识别为变量名而不是枚举常量名。

例如：

```
int S_TYPE = 6
```

print S_TYPE //输出“6”

此时，引用枚举常量时则不能省略枚举类型名。

例如：

int S_TYPE = 6

print $VEL_PROFILE::S_TYPE //输出“S_TYPE”

2.6.1 $VEL_PROFILE(速度曲线)

描述

$VEL_PROFILE 枚举类型包含了所有运动轨迹支持的速度曲线类型。

定义

enum $VEL_PROFILE

{

T_type,

S_type,

O_type

}

成员

$VEL_PROFILE 枚举类型的成员详见表 2-26。

表 2-26 $VEL_PROFILE 枚举类型的成员

名称	说明
T_type	T 型速度曲线
S_type	S 型速度曲线
O_type	O 型速度曲线

用法举例

$VEL_PROFILE = S_type //指定以下运动指令采用 S 型速度规划

movej j:j1,vp:5%,sp:-1%

关于速度曲线更详细内容请参见相关运动指令。

2.6.2 printto(输出定向)

描述

printto 枚举类型包含了所有 print 指令可以设置的输出定向。

定义

enum printto

{

off,

hmi,
file,
console
}

成员

printto 枚举类型的成员详见表 2-27。

表 2-27 printto 枚举类型的成员

名称	说明
off	关闭输出，即不输出到任何地方
hmi	输出到 HMI(人机界面)的消息栏
file	输出到指定文件
console	输出到终端，用于开发调式，一般用户不会用到此设置项

用法举例

print to:hmi, "hello world" //HMI 消息栏输出“hello world”

关于输出定向更详细内容参见第 3.5 章节 。

2.6.3 num_base(输出数制)

描述

num_base 枚举类型包含了所有 print 指令可以设置的输出数制。

定义

enum num_base
{
hex,
dec
}

成员

num_base 枚举类型的成员详见表 2-28。

表 2-28 num_base 枚举类型的成员

名称	说明
hex	16 进制
dec	10 进制

用法举例

print hex,ffh //输出“ff”

print dec,ffh //输出“255”

关于输出数制更详细内容请参见第 3.5 章节。

2.6.4 stoptype(停止类型)

描述

stoptype 枚举类型包含了所有运动轨迹的停止方式。

定义

```
enum stoptype
{
general,
fast
}
```

成员

stoptype 枚举类型的成员详见表 2-29。

表 2-29 stoptype 枚举类型的成员

名称	说明
general	普通减速停止方式
fast	快速减速停止方式，是普通减速停止方式加速度的 5 倍。

用法举例

stopmove fast //以快速停止方式停止当前运动

详细信息参见 stopmove 指令。

2.6.5 controlmode(控制模式)

描述

controlmode 枚举类型定义了系统的控制模式。

定义

```
enum controlmode
{
MANUAL,
AUTO,
MANUFAST
}
```

成员

controlmode 枚举类型的成员详见表 2-30。

表 2-30 controlmode 枚举类型的成员

名称	说明
MANUAL	手动低速模式
AUTO	AUTO
MANUFAST	手动高速模式

2.6.6 stopbits(停止位)

描述

stopbits 枚举类型定义了串口通信中的停止位类型。

定义

```
enum stopbits
{
one,
two
}
```

成员

stopbits 枚举类型的成员详见表 2-31。

表 2-31 stopbits 枚举类型的成员

名称	说明
one	表示 1 位停止位
two	表示 2 位停止位

2.6.7 parity(奇偶校验类型)

描述

parity 枚举类型定义了串口通信中的奇偶校验类型。

定义

```
enum parity
{
none,
odd,
even
}
```

成员

parity 枚举类型的成员详见表 2-32。

表 2-32 parity 枚举类型的成员

名称	说明
none	表示不使用奇偶校验位
odd	表示使用奇校验
even	表示使用偶校验

2.6.8 weaveshape(叠加轨迹类型)

描述

weaveshape 枚举类型包含了所有叠加轨迹支持的图形类型。

定义

```
enum weaveshape
{
simple,
V_shape,
triangle,
ellipse,
eight
}
```

成员

weaveshape 枚举类型的成员详见表 2-33。

表 2-33 weaveshape 枚举类型的成员

名称	说明
simple	横摆
V_shape	V 型摆
triangle	空间三角摆
ellipse	椭圆摆
eight	“8”字摆

用法举例

```
weavedata weavedata = {V_shape,2,15,90} //指定该叠加图形为 V 型摆
```

2.6.9 weaverotaxis(叠加轨迹摆动平面偏转轴)

描述

weaverotaxis 枚举类型包含了叠加轨迹支持的摆动平面偏转所参考的坐标轴。

定义

```
enum weaverotaxis
{
rot_x,
rot_y,
rot_z
}
```

成员

weaverotaxis 枚举类型的成员详见表 2-34。

表 2-34 weaverotaxis 枚举类型的成员

名称	说明
rot_x	参考 x 轴偏转
rot_y	参考 y 轴偏转
rot_z	参考 z 轴偏转

用法举例

```
weavedata weavedata = {V_shape,2,15,90, rot_x,10} //指定该叠加轨迹的摆动平面参考 x 轴进行偏转
```

2.7 其他类型

2.7.1 socket(套接字类型)

描述

socket 套接字类型用于通过网口与外部设备通信。该数据类型配合 connect，accept，write，read，readuntil 等函数使用。

用法举例

参见函数 connect，accept，write，read，readuntil。

2.7.2 iodev(IO 设备类型)

描述

IO 设备类型用于 IO 读写，例如通过串口读写与外部设备通信。该数据类型配合 open，write，read，readuntil 等函数使用。

用法举例

参见函数 open，write，read，readuntil。

2.8 数组

相同类型的变量的集合，放在一起处理比较方便。这种情况下，可以使用数组。数组，就是相同数据类型的元素按一定顺序排列的集合。数组可以是一维的、也可以是多维的：

- 一维数组

元素不是数组的数组。

- 多维数组

二维数组及以上的数组。以数组作为元素的数组是二维数组，以二维数组为元素的数组是三维数组，当然也可以生成更高的数组。

2.8.1 数组的声明（使用数组前的准备）

声明一个数组变量的格式为：

变量类型名 数组变量名[第 1 维大小][第 2 维大小]……

例如：

double a[10] //声明了一个 double 类型的一维数组，数组大小为 10

double b[2][3] //声明了一个 double 类型的二维数组，第 1 维大小为 2，第 2 维大小为 3。

2.8.2 数组初始化

数组在声明的同时可以进行初始化，初始化数据的格式为：

{第 1 维的第 1 个数据值，第 1 维的第 2 个数据值，……第 n 维的第 1 个数据值，第 n 维的第 2 个数据值……第 n 维的第 n 个数据值}

例如：

double a[3] = {1,2,3}

double b[2][3] = {1,2,3,1,2,3}

允许只初始化数组的前 m 个元素，未初始化的元素保持默认值。

2.8.3 访问数组（数组的使用方法）

使用下标来访问数组中的元素，数组每一维的下标从 0 开始。

例如：

```
double a[3] = {1,2,3}
double b[2][3] = {1,2,3,4,5,6}
print a[0] //输出“1”
print b[0][1] //输出“2”
double c = a[0] + b[0][1]
print c //输出“3”
```

注意

一维数组的下标限制为 10000，多维数组下标乘积限制为 10000。

2.8.4 数组管理结构体变量

结构体变量也可以组成数组来管理。

例如：

```
pos p[3] = {{0,0,0},{10,10,10},{20,20,20}}
print p[0] //输出“{0,0,0}”
print p[1].x //输出“10”
```

2.9 运算符

2.9.1 算术运算符

算术运算符的分类及作用详见表 2-35：

表 2-35 算数运算符

操作符	作用	适用数据类型
+	加	int,byte,double,string,joint
-	减/取负	int,byte,double,joint
*	乘	int,byte,double,frame,pose
/	除	int,byte,double,
%	取余	int,byte,double,
--	自减 1	int
++	自加 1	int

string 类型的数据相加表示字符串连接。

frame 类型的数据相乘表示坐标系变换。

各种算术运算符使用举例如下：

```
int sa = 1
int sb = 2
int sc = -sb //取负
int sac = sa * sc //乘法
sac = -2
```

“++”和“--”用法如下：

“x=m++”表示将 m 的值赋给 x 后, m 加 1；

“x=++m”表示 m 先加 1 后，再将新值赋给 x。

“--”操作符和“++”用法相同。

2.9.2 按位运算符

按位运算符的分类及作用详见表 2-36：

表 2-36 按位运算符

操作符	作用	适用数据类型
<<	向左移位	int,byte
>>	向右算术移位	int,byte
&	按位与	int,byte
|	按位或	int,byte
^	按位异或	int,byte
~	按位取非	int,byte

各种按位运算符使用举例如下：

int A = 00111100b

int B = 00001101b

print A," ",A&B," ",A|B," ",A^B," ",~A," ",A<<2," ",A>>2 //输出“ 60 12 61 49 -61 240 15”

2.9.3 逻辑运算符

逻辑运算符的分类及作用详见表 2-37：

表 2-37 逻辑运算符

操作符	作用	适用数据类型
&&	逻辑与	int,byte,bool
||	逻辑或	int,byte,bool
<	小于	int,byte,double,string
>	大于	int,byte,double,string
<=	小于等于	int,byte,double,string
>=	大于等于	int,byte,double,string
==	等于	int,byte,double,bool,string
!=	不等于	int,byte,double,bool,string
!	取逻辑非	int,byte, bool

逻辑与“&&”表达式为真的条件是两边的结果都为真，而逻辑或“||”为真的条件是两边只要有一个条件为真即可。

各种逻辑运算符使用举例如下：

```
int i = 0
while(getdi(1) && !getdi(2))
if(i<100)
i++
endif
endwhile
```

2.9.4 赋值运算符

赋值运算符的分类及作用详见表 2-38：

表 2-38 赋值运算符

操作符	作用
=	赋值
+=	加等
-=	减等
*=	乘等
/=	除等
%=	取模等
<<=	左移等
>>=	算术右移等
&=	按位与等
|=	按位或等

所有数据类型均支持赋值操作，XX=运算符支持的数据类型与 XX 运算符支持的数据类型一致。

“a += b”等同于“a = a + b”。其他同理。

各种赋值运算符使用举例如下：

```
int a = 1
print a //输出“1”
a += 1
print a //输出“2”
a <<= 2
print a //输出“8”
```

2.9.5 其他运算符

其他运算符的分类及作用详见表 2-39：

表 2-39 其他运算符

操作符	作用
[ ]	数组下标 数组下标操作符的用法参见“2.8 数组”一节
( )	圆括号 取结构体成员变量操作符参见“2.4 结构体类型（适用于六轴机器人）”一节
.	取结构体成员变量

圆括号用于指定运算的优先级，如：

```
int num = (1+2)*3
print num //输出“9”
```

2.9.6 运算符优先级

不同运算符的优先级顺序如下表 2-40 所示，该表中优先级数字越小，表示优先级越高。同一优先级的运算符，运算次序由结合方向所决定。

表 2-40 运算符优先级

优先级	运算符	名称或含义	使用形式	结合方向	说明
1	[ ]	数组下标	数组名[常量表达式]	左到右	-
	( )	圆括号	（表达式）/函数名(形参表)		-
	.	结构体变量成员选择	结构体变量.成员名		-
2	-	负号运算符	-表达式	右到左	单目运算符
	++	自增运算符	++变量名/变量名++		单目运算符
	--	自减运算符	--变量名/变量名--		单目运算符
	!	逻辑非运算符	!表达式		单目运算符
	~	按位取反运算符	~表达式		单目运算符
3	/	除	表达式/表达式	左到右	双目运算符
	*	乘	表达式*表达式		双目运算符
	%	余数（取模）	整型表达式/整型表达式		双目运算符
4	+	加	表达式+表达式	左到右	双目运算符
	-	减	表达式-表达式		双目运算符
5	<<	左移	变量<<表达式	左到右	双目运算符
	>>	右移	变量>>表达式		双目运算符
6	>	大于	表达式>表达式	左到右	双目运算符
	>=	大于等于	表达式>=表达式		双目运算符
	<	小于	表达式<表达式		双目运算符
	<=	小于等于	表达式<=表达式		双目运算符
7	==	等于	表达式==表达式	左到右	双目运算符
	!=	不等于	表达式!= 表达式		双目运算符
8	&	按位与	表达式&表达式	左到右	双目运算符
9	^	按位异或	表达式^表达式		双目运算符
10	|	按位或	表达式|表达式		双目运算符
11	&&	逻辑与	表达式&&表达式	左到右	双目运算符
12	||	逻辑或	表达式||表达式	左到右	双目运算符
13	=	赋值运算符	变量=表达式	右到左	-

优先级	运算符	名称或含义	使用形式	结合方向	说明
	/=	除后赋值	变量/=表达式		
	*=	乘后赋值	变量*=表达式		
	%=	取模后赋值	变量%=表达式		
	+=	加后赋值	变量+=表达式		
	-=	减后赋值	变量-=表达式		
	<<=	左移后赋值	变量<<=表达式		
	>>=	右移后赋值	变量>>=表达式		
	&=	按位与后赋值	变量&=表达式		
	^=	按位异或后赋值	变量^=表达式		
	|=	按位或后赋值	变量|=表达式		


注意

实际应用中如果不确定运算符的优先级大小，请尽量使用圆括号显示表达希望的运算顺序，否则表达式的计算结果可能和用户的预期不符。

2.10 变量的作用域

按变量的作用域来划分，变量可以分为局部变量，全局变量和系统变量。ARL 中，在函数内部声明的变量为局部变量，在函数外部声明的变量默认为全局变量，系统变量为系统预定义的变量，不需要声明，可以直接通过”$”符号引用。

局部变量只在当前函数内有效。

例如下面的程序：

```
func void myfunc()
for(int i=1;i<2;i++)
int a = 1 //该变量只在 for 和 endfor 范围内有效
print a //这里可以访问到 a
endfor
print a //这里将报出变量 a 不存在
endfunc
```

在同一个作用域范围内声明的变量名不能重复，否则系统将报出变量重复定义的错误。但是在不同的作用域范围内声明的变量名可以重复，此时使用该名字访问的变量是最近作用域范围内的变量。如：

```
func void myfunc()
int a = 6
for(int i=1;i<2;i++)
int a = 1 //该变量只在 for 和 endfor 范围内有效
print a //这里输出“1”
endfor
print a //这里输出“6”
```

```
endfunc
```

全局变量在声明后的当前程序中都可以访问。如：

```
int a = 1
func void myfunc()
a = 2
print a //这里输出“2”
endfunc
func void main()
print a //这里输出“1”
myfunc()
endfunc
```

系统变量是任意通道、函数、程序都共用的，如：

```
func void main()
$DFSPPED = {10,50,5,5,5} //这里将默认速度设置为：{10,50,5,5,5}
print $DFSPPED //这里输出“{10,50,5,5,5}”
……
endfunc
```

3 顺序指令

3.1 顺序指令的一般格式

除流程控制指令以外，ARL 顺序指令具有如下所示的一般格式：

指令名 参数名：参数值，参数名：参数值，……

其中有些参数是强制参数，如果不写，系统会报出指令格式错误。有些参数是可选参数，可选参数允许缺省，缺省时将取默认值或者保持之前设定的值。以下指令格式中[ ]内的参数表示该参数为可选参数。

某些参数的参数名部分也可以缺省而只写参数值即可，而某些参数的参数名部分则不能缺省。以下指令格式中使用“参数名：”的形式表示该参数的参数名不能缺省。在允许参数名缺省的语法规则有一种特例，当参数值是以花括号“{ }”形式表达的结构体常量而不是变量时，参数名部分不能缺省，否则系统将会报出指令格式错误。

3.2 运动指令（适用于六轴机器人）

3.2.1 movej(移动轴)

描述

movej 指令用于将机器人轴或外轴移动到一个指定的轴位置。所有轴同时到达目标轴位置。

格式

movej j:,[v: | vp:],[s: | sp: | sl:],[t:],[dura:]

参数

movej 指令的参数详见表 3-1。

表 3-1 movej 指令的参数

名称	说明
j	数据类型：joint
	该参数指定机器人各轴和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。程序中如果第一条执行的运动指令为 movej，则该 movej 指令的目标点 j1~j6 以及配置的外轴均不允许缺省
v	数据类型：speed
	该参数指定运动速度。movej 指令使用 speed 结构体变量中的 per 和 exj 分量，分别用于指定轴运动速度百分比和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 代替，格式见用法举例
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%

名称	说明
s	数据类型：slip
	该参数指定平滑距离。movej 指令使用 slip 结构体的 perdis 和 ejdis 分量，用于描述脱离原轨迹进入平滑轨迹点。其中，perdis 用于指定距离目标点百分比，ejdis 用于指定外轴距离目标点轴位置角度。这里如果缺省 s 参数，则默认使用系统变量$DFSLIP 作为 s 参数的值；如果指定了 s 参数，则成员分量的值会被更新到$DFSLIP 对应的分量中 在简化编程中，s 可由平滑百分比参数 sp 代替，格式见用法举例 注： ■ 当 perdis>=0 时，其余分量将被忽略； ■ 当同时指定 ejdis 时，更靠近目标点的分量将会被选用； ■ 当上述分量均为 0 时，表示不平滑，但插补点不一定与目标点重合； ■ 当上述分量均小于 0 时，表示不平滑，目标点肯定会有插补点，即准停
sp	数据类型：double
	该参数指定平滑百分比。可替代平滑参数 s，用于不需要精确指定平滑大小的场合。格式为 sp:10%，表示当前行目标点的平滑距离是最大平滑距离的 10%
sl	数据类型：double
	该参数指定平滑距离。可替代平滑参数 s，用于需要精确指定平滑大小的场合。格式为 sl:10mm，表示当前行的平滑距离是 10mm
t	该参数指定工具。用户需要在法兰末端安装的工具上自定义工具坐标系，该工具坐标系与工具固连，目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。这里如果缺省了 t 参数，则默认使用系统变量$DFTOOL 作为 t 参数的值；如果指定了 t 参数，则成员分量的值会被更新到$DFTOOL 对应的分量中，该参数用于限制手动低速模式下运行程序时，工具末端的线速度不大于 250 毫米/秒。
dura	数据类型：double
	该参数指定轨迹时间。用户可以直接指定运动时间而不是运动速度。如果用户指定了 dura 参数，系统将忽略 speed 参数，而是通过自动调整速度来满足 dura 参数指定的时间要求。dura 参数单位是秒

用法举例

```
//模糊指定速度和平滑大小
joint j1 = {j1 10,j2 20,j3 30,j4 40,j5 50,j6 60}
movej j:j1,vp:5%,sp:5%
movej j:{j1 10,j2 20,j3 30,j4 40,j5 50,j6 60}
movej j:{j1 20,ej1 30}
joint p = {0,20,30,40,50,60}
speed v = {per 50}
movej p,v
```

3.2.2 ptp(点到点)

描述

ptp 指令用于将机器人从一个点快速运动到另一个点而又不要求 TCP 点所走轨迹形状时。所有轴同时到达目标点。

格式

ptp p:,[v: | vp:],[s: | sl: | sp:],[t:],[w:],[dura:]

参数

ptp 指令的参数详见表 3-2。

表 3-2 ptp 指令的参数

名称	说明
p	数据类型：pose
	该参数指定机器人 TCP 目标位姿和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。cfg 分量缺省默认值为 0，turn 分量缺省默认值为-1。程序中如果第一条执行的运动指令为 ptp，则该 ptp 指令的目标点 X,Y,Z,A,B,C 以及配置的外轴均不允许缺省
v	数据类型：speed
	该参数指定运动速度。ptp 指令使用 speed 结构体变量中的 per 和 exj 分量，分别用于指定轴运动速度百分比和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 代替，格式见用法举例
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%
s	数据类型：slip
	该参数指定平滑距离。PTP 指令使用 slip 结构体的 perdis 和 ejdis 分量，用于描述脱离原轨迹进入平滑轨迹点。其中，perdis 用于指定距离目标点百分比，ejdis 用于指定外轴距离目标点轴位置角度。这里如果缺省 s 参数，则默认使用系统变量$DFSLIP 作为 s 参数的值；如果指定了 s 参数，则成员分量的值会被更新到$DFSLIP 对应的分量中 在简化编程中，s 可由平滑百分比参数 sp 代替，格式见用法举例 注： ■ 当 perdis>=0 时，其余分量将被忽略； ■ 当同时指定 ejdis 时，更靠近目标点的分量将会被选用； ■ 当上述分量均为 0 时，表示不平滑，但插补点不一定与目标点重合； ■ 当上述分量均小于 0 时，表示不平滑，目标点肯定会有插补点，即准停
sp	数据类型：double
	该参数指定平滑百分比。可替代平滑参数 s，用于不需要精确指定平滑大小的场合。格式为 sp:10%，表示当前行目标点的平滑距离是最大平滑距离的 10%
sl	数据类型：double
	该参数指定平滑距离。可替代平滑参数 s，用于需要精确指定平滑大小的场合。格式为 sl:10mm，表示当前行的平滑距离是 10mm
t	数据类型：tool 结构体
	该参数指定工具。用户需要在法兰末端安装的工具上自定义工具坐标系，该工具坐标系与工具固连，目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。这里如果缺省了 t 参数，则默认使用系统变量$DFTOOL 作为 t 参数的值；如果指定了 t 参数，则成员分量的值会被更新到$DFTOOL 对应的分量中

名称	说明
w	数据类型：wobj 结构体
	该参数指定工件坐标系。用户可以自定义工件坐标系。目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。在某些情况下，用户可能通过在特定的坐标系中编程更方便。另外，当工件移动位置时，只需重新标定工件坐标系即可，不需要修改用户程序。这里如果缺省了 w 参数，则默认使用系统变量$DFWOBJ 作为 w 参数的值；如果指定了 w 参数，则成员分量的值会被更新到$DFWOBJ 对应的分量中
dura	数据类型：double
	该参数指定轨迹时间。用户可以直接指定运动时间而不是运动速度。如果用户指定了 dura 参数，系统将忽略 speed 参数，通过自动调整速度来满足 dura 参数指定的时间要求。dura 参数单位是秒

以上结构体数据详细信息请参考第 2 章节。

用法举例

```
pose p = {x 1500,y 500,z 500,a 0,b 90,c 0,cfg 0}
//模糊指定速度和平滑大小
ptp p:p1,vp:5%,sp:5%
ptp p,v:{per 10}
ptp p:{x 1000,y 500,z 500,a 0,b 90,c 0}
ptp p,dura:10 //10s 运动至 p 点
tool t = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w = {{x 1000,y 0,z 500,a 0,b 10,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
pose p2 = {x 10,y 10,z 10,a 0,b 90,c 0,cfg 0}
ptp p2,v,t,w
```

3.2.3 lin(直线运动)

描述

lin 指令用于将机器人 TCP 点沿直线路径运动到目标点位姿；位置移动和姿态转动同步。

格式

```
lin p:,[ v: | vl: | vp: ],[ s: | sl: | sp: ],[ t: ],[ w: ],[ dura: ]
```

参数

lin 指令的参数详见表 3-3。

表 3-3 lin 指令的参数

名称	说明
p	数据类型：pose
	该参数指定机器人 TCP 目标位姿和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。cfg 参数被忽略，与起点 cfg 参数保持一致，turn 分量缺省默认值为-1
v	数据类型：speed

名称	说明
	该参数指定运动速度。lin 指令使用 speed 结构体中的 tcp、ori 和 exj 分量，分别用于指定 TCP 点运动速度、TCP 姿态变化速度和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 或速度绝对值参数 vl 代替
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%
vl	数据类型：double
	该参数指定运动线速度。可替代速度参数 v，用于需要精确指定速度大小的场合。格式为 vl:200mm/s，表示当前行的 TCP 最大速度的 200mm/s
s	数据类型：slip
	该参数指定平滑距离。LIN 指令使用 slip 结构体的 perdis、pdis、odis 和 ejdis 分量，用于描述脱离原轨迹进入平滑轨迹点。其中，perdis 用于指定距离目标点百分比，pdis 用于指定距离目标点轴路径距离，odis 用于指定距离目标点姿态角度，ejdis 用于指定外轴距离目标点轴位置角度。这里如果缺省 s 参数，则默认使用系统变量$DFSLIP 作为 s 参数的值；如果指定了 s 参数，则成员分量的值会被更新到$DFSLIP 对应的分量中 在简化编程中，s 可由平滑百分比参数 sp 或平滑绝对值参数 sl 代替，格式见用法举例 注： ■ 当 perdis>=0 时，其余分量将被忽略； ■ 当同时指定 pdis、odis 和 ejdis 时，更靠近目标点的分量将会被选用； ■ 当上述分量均为 0 时，表示不平滑，但插补点不一定与目标点重合； ■ 当上述分量均小于 0 时，表示不平滑，目标点肯定会有插补点，即准停； 使用笛卡尔空间轨迹（即 LIN、CIR）平滑时，建议使用 pdis 平滑，可以使平滑轨迹在前后轨迹上均匀
sp	数据类型：double
	该参数指定平滑百分比。可替代平滑参数 s，用于不需要精确指定平滑大小的场合。格式为 sp:10%，表示当前行目标点的平滑距离是最大平滑距离的 10%
sl	数据类型：double
	该参数指定平滑距离。可替代平滑参数 s，用于需要精确指定平滑大小的场合。格式为 sl:10mm，表示当前行的平滑距离是 10mm
t	数据类型：tool 结构体
	该参数指定工具。用户需要在法兰末端安装的工具上自定义工具坐标系，该工具坐标系与工具固连，目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。这里如果缺省了 t 参数，则默认使用系统变量$DFTOOL 作为 t 参数的值；如果指定了 t 参数，则成员分量的值会被更新到$DFTOOL 对应的分量中 该参数是为了保证手动低速运行程序时限制 TCP 速度不大于 250mm/s
w	数据类型：wobj 结构体
	该参数指定工件坐标系。用户可以自定义工件坐标系。目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。在某些情况下，用户可能通过在特定的坐标系中编程更方便。另外，当工件移动位置时，只需重新标定工件坐标系即可，不需要修改用户程序。这里如果缺省了 w 参数，则默认使用系统变量$DFWOBJ 作为 w 参数的值；如果指定了 w 参数，则成员分量的值会被更新到$DFWOBJ 对应的分量中

名称	说明
dura	数据类型：double
	该参数指定轨迹时间。用户可以直接指定运动时间而不是运动速度。如果用户指定了 dura 参数，系统将忽略 speed 参数，而是通过自动调整速度来满足 dura 参数指定的时间要求。dura 参数单位是秒

以上结构体数据详细信息请参考第 2 章节。

用法举例

```
pose p = {x 1500,y 500,z 500,a 0,b 90,c 0,cfg 0}
ptp p,v:{per 10}
//模糊指定速度和平滑大小
lin p:p1,vp:5%,sp:5%
//精确指定速度和平滑距离
lin p:p1, vl:100mm/s,sl:5mm
lin p:{x 1000,y 500,z 500,a 0,b 90,c 0}
lin p,dura:10
//精确指定速度、平滑结构体
tool t = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w = {{x 1000,y 0,z 500,a 0,b 10,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
slip s = {pdis 10, ejdis 10, odis 10}
pose p2 = {x 10,y 10,z 10,a 0,b 90,c 0,cfg 0}
lin p2,t,w,v,s
```

3.2.4 spl(样条曲线运动)

描述

spl 指令使机器人平滑不停顿地经过示教点。


注意

如果两个示教点间距过小，容易导致 spl 轨迹出现较大偏移。在这种情况下，请根据运行时的告警信息适当增加示教点，确保轨迹符合预期。

格式

```
spl p: , [ v: | vp: ],[ s: | sl: | sp: ],[ t: ],[ w: ]
```

参数

spl 指令的参数详见表 3-5。

表 3-4 spl 指令的参数

名称	说明
p	数据类型：pose

名称	说明
	该参数指定机器人 TCP 目标位姿和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。cfg 参数被忽略，与起点 cfg 参数保持一致，turn 分量缺省默认值为-1
v	数据类型：speed
	该参数指定运动速度。lin 指令使用 speed 结构体中的 tcp、ori 和 exj 分量，分别用于指定 TCP 点运动速度、TCP 姿态变化速度和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 或速度绝对值参数 vl 代替
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%
s	数据类型：slip
	该参数指定平滑距离。LIN 指令使用 slip 结构体的 perdis、pdis、odis 和 ejdis 分量，用于描述脱离原轨迹进入平滑轨迹点。其中，perdis 用于指定距离目标点百分比，pdis 用于指定距离目标点轴路径距离，odis 用于指定距离目标点姿态角度，ejdis 用于指定外轴距离目标点轴位置角度。这里如果缺省 s 参数，则默认使用系统变量$DFSLIP 作为 s 参数的值；如果指定了 s 参数，则成员分量的值会被更新到$DFSLIP 对应的分量中 在简化编程中，s 可由平滑百分比参数 sp 或平滑绝对值参数 sl 代替，格式见用法举例 注： ■ 当 perdis>=0 时，其余分量将被忽略； ■ 当同时指定 pdis、odis 和 ejdis 时，更靠近目标点的分量将会被选用； ■ 当上述分量均为 0 时，表示不平滑，但插补点不一定与目标点重合； ■ 当上述分量均小于 0 时，表示不平滑，目标点肯定会有插补点，即准停； 使用笛卡尔空间轨迹（即 LIN、CIR）平滑时，建议使用 pdis 平滑，可以使平滑轨迹在前后轨迹上均匀
sl	数据类型：double
	该参数指定平滑距离。可替代平滑参数 s，用于需要精确指定平滑大小的场合。格式为 sl:10mm，表示当前行的平滑距离是 10mm
sp	数据类型：double
	该参数指定平滑百分比。可替代平滑参数 s，用于不需要精确指定平滑大小的场合。格式为 sp:10%，表示当前行目标点的平滑距离是最大平滑距离的 10%
t	数据类型：tool 结构体
	该参数指定工具。用户需要在法兰末端安装的工具上自定义工具坐标系，该工具坐标系与工具固连，目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。这里如果缺省了 t 参数，则默认使用系统变量$DFTOOL 作为 t 参数的值；如果指定了 t 参数，则成员分量的值会被更新到$DFTOOL 对应的分量中 该参数是为了保证手动低速运行程序时限制 TCP 速度不大于 250mm/s
w	数据类型：wobj 结构体
	该参数指定工件坐标系。用户可以自定义工件坐标系。目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。在某些情况下，用户可能通过在特定的坐标系中编程更方便。另外，当工件移动位置时，只需重新标定工件坐标系即可，不需要修改用户程序。这里如果缺省了 w 参数，则默认使用系统变量$DFWOBJ 作为 w 参数的值；如果指定了 w 参数，则成员分量的值会被更新到$DFWOBJ 对应的分量中

用法举例

示例程序：

```
lin P:P1
lin P:P2
spl P:P3
spl P:P4
spl P:P5
spl P:P6
lin P:P7
```

程序运行如图 3-1 所示，从程序点 2 平滑移动到程序点 6。

- 沿根据程序点 2、3、4 的示教点形成的轨迹运行到程序点 3。
- 沿根据程序点 2、3、4 的示教点形成的 3、4 区间的轨迹和根据程序点 3、4、5 的示教点形成的 3、4 区间的轨迹合成的轨迹运行到程序点 4。
- 沿根据程序点 3、4、5 的示教点形成的 4、5 区间的轨迹和根据程序点 4、5、6 的示教点形成的 4、5 区间的轨迹合成的轨迹运行到程序点 5。
- 沿根据程序点 4、5、6 的示教点形成的轨迹运行到程序点 6。

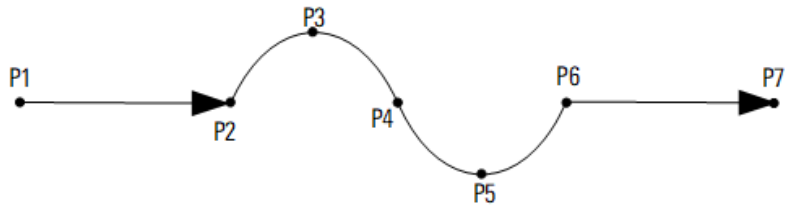


图 3-1 程序执行示意图

3.2.5 cir(圆弧运动)

描述

cir 指令用于将机器人 TCP 点沿圆弧路径运动到目标点；平移运动和旋转运动同步。

圆弧路径坐标系

圆弧路径坐标系如下图 3-2 所示，X 轴为路径的切线方向，Y 轴指向圆心方向，用右手定则确定 Z 轴方向。

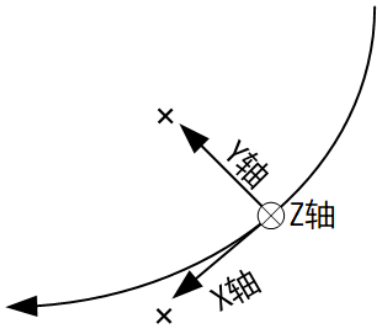


图 3-2 圆弧路径坐标系示意图

格式

cir m:,p:,[v: | vl: | vp:],[s: | sl: | sp:],[t:],[w:],[CA:],[dura:]

参数

cir 指令的参数详见表 3-5。

表 3-5 cir 指令的参数

名称	说明
m	数据类型：pose
	该参数指定圆弧辅助点。起点、辅助点和目标点，这 3 点唯一确定了一段圆弧。辅助点中只有 x，y，z 分量被使用，他们中值为 9e+09 的分量将保持起点位置值，其他分量被忽略
p	数据类型：pose
	该参数指定机器人 TCP 目标位姿和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。cfg 参数被忽略，与起点 cfg 参数保持一致，turn 分量缺省默认值为-1
v	数据类型：speed
	该参数指定运动速度。cir 指令使用 speed 结构体中的 tcp、ori 和 exj 分量，分别用于指定 TCP 点运动速度、TCP 姿态变化速度和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 或速度绝对值参数 vl 代替，格式见用法举例
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%
vl	数据类型：double
	该参数指定运动线速度。可替代速度参数 v，用于需要精确指定速度大小的场合。格式为 vl:500mm/s，表示当前行的 TCP 最大速度的 500mm/s
s	数据类型：slip
	该参数指定平滑距离。LIN 指令使用 slip 结构体的 perdis、pdis、odis 和 ejdis 分量，用于描述脱离原轨迹进入平滑轨迹点。其中，perdis 用于指定距离目标点百分比，pdis 用于指定距离目标点轴路径距离，odis 用于指定距离目标点姿态角度，ejdis 用于指定外轴距离目标点轴位置角度。这里如果缺省 s 参数，则默认使用系统变量$DFSLIP 作为 s 参数的值；如果指定了 s 参数，则成员分量的值会被更新到$DFSLIP 对应的分量中

名称	说明
	在简化编程中，s 可由平滑百分比参数 sp 或平滑绝对值参数 sl 代替，格式见用法举例 注： ■ 当 perdis>=0 时，其余分量将被忽略； ■ 当同时指定 pdis、odis 和 ejdis 时，更靠近目标点的分量将会被选用； ■ 当上述分量均为 0 时，表示不平滑，但插补点不一定与目标点重合； ■ 当上述分量均小于 0 时，表示不平滑，目标点肯定会有插补点，即准停； 使用笛卡尔空间轨迹（即 LIN、CIR）平滑时，建议使用 pdis 平滑，可以使平滑轨迹在前后轨迹上均匀
sp	数据类型：double
	该参数指定平滑百分比。可替代平滑参数 s，用于不需要精确指定平滑大小的场合。格式为 sp:10%，表示当前行目标点的平滑距离是最大平滑距离的 10%。
sl	数据类型：double
	该参数指定平滑距离。可替代平滑参数 s，用于需要精确指定平滑大小的场合。格式为 sl:10mm，表示当前行的平滑距离是 10mm。
w	数据类型：wobj 结构体
	该参数指定工件坐标系。用户可以自定义工件坐标系。目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。在某些情况下，用户可能通过在特定的坐标系中编程更方便。另外，当工件移动位置时，只需重新标定工件坐标系即可，不需要修改用户程序。这里如果缺省了 w 参数，则默认使用系统变量$DFWOBJ 作为 w 参数的值；如果指定了 w 参数，则成员分量的值会被更新到$DFWOBJ 对应的分量中
CA	数据类型：double
	该参数指定圆心角（Circle Angle）。用户可以不直接指定目标点，而通过指定圆弧转过的圆心角的方式来指定目标点。如果用户指定了 CA 参数，则 p 参数只用来和辅助点一起确定圆弧的几何形状，而不是真正的目标点，真正的目标点系统通过用户指定的圆心角自动计算。CA 参数单位：度
dura	数据类型：double
	该参数指定轨迹时间。用户可以直接指定运动时间而不是运动速度。如果用户指定了 dura 参数，系统将忽略 speed 参数，而是通过自动调整速度来满足 dura 参数指定的时间要求。dura 参数单位：秒

以上结构体数据详细信息请参考第 2 章节。

用法举例

```
pose p3={x 1000,y 200,z 500,a 90,b 0,c 90}
pose p4={x 800,y 0,z 500,a 90,b 0,c 90}
tool tool1 = {{x 0,y 10,z 15,a 0,b 90,c 0}}
cir p3,p4,tool1
//指定圆弧圆心角为 360 度
cir p3,p4,CA:360
//模糊指定速度和平滑大小
cir m:p1,p:p2,vp:5%,sp:5%
//精确指定速度和平滑距离
cir m:p3,p:p4,vl:500mm/s,sl:5mm
//精确指定速度和平滑结构体
```

```
tool t1 = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w1 = {{x 1000,y 0,z 500,a 0,b 10,c 0}}
speed v2 = {per 10,tcp 50,ori 5,exj 10}
slip s = { pdis 10, ejdis 10, odis 10 }
cir p3,p4,t1,w1,v2, s
```

注意

- 在单步或段调试模式下，cir 指令分为两步执行，第一步运行至辅助点，第二步运行至目标点。
- 在程序调试器内，当执行<跳转>到 cir 指令动作的时候，机器人为走 lin 指令的方式运动到目标点，而不是走圆弧轨迹。
- 虽然用户可以通过使用参数缺省来简化程序，但仍建议用户在编写运动指令时尽量将参数写全，以防止由于编程人员的疏忽而导致的实际的运动与用户预期不符，进而导致一些不必要的损失或伤害。

3.2.6 ccir(连续圆弧运动)

描述

在 cir 指令中，用户需要对经由点和终点的 2 个位置进行示教。在 ccir 指令中，只需示教一个点，但至少需要连续示教两条 ccir 指令，用于确定圆弧路径。

ccir 指令与 cir 指令相比具有如下特征：

- 可以在圆弧动作的经由点和终点分别指定速度和平滑。
- 可以在经由点和终点之间示教逻辑指令。但可进行示教的逻辑指令会受到限制。
- 可在两条 ccir 之间插入 ccir，用于微调圆弧轨迹。

注意

当出现以下情况时，无法创建圆弧，且系统报告“[12002][0]非法圆弧平面”。

- 当创建的圆弧点数小于 3 个时，无法构成圆弧。
- 当创建的圆弧上的 3 点，构成一条直线时，无法创建圆弧。
- 当 ccir 指令中出现连续的相同点时，无法创建圆弧。
- ccir 指令之间不允许有停前瞻的语句，例如：ccir 指令之间有 waittime 0 的时候，会报警非法圆弧平面

注意

如果两个示教点间距过小，容易导致 ccir 轨迹出现较大偏移。在这种情况下，请根据运行时的告警信息适当增加示教点，确保轨迹符合预期。

格式

```
ccir p:,[ v: | vl: | vp: ],[ s: | sl: | sp: ],[ t: ],[ w: ]
```

参数

ccir 指令的参数详见表 3-3。

表 3-6 ccir 指令的参数

名称	说明
p	数据类型：pose
	该参数指定机器人 TCP 目标位姿和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。cfg 参数被忽略，与起点 cfg 参数保持一致，turn 分量缺省默认值为-1

名称	说明
v	数据类型：speed
	该参数指定运动速度。lin 指令使用 speed 结构体中的 tcp、ori 和 exj 分量，分别用于指定 TCP 点运动速度、TCP 姿态变化速度和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 或速度绝对值参数 vl 代替
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%
vl	数据类型：double
	该参数指定运动线速度。可替代速度参数 v，用于需要精确指定速度大小的场合。格式为 vl:200mm/s，表示当前行的 TCP 最大速度的 200mm/s
s	数据类型：slip
	该参数指定平滑距离。LIN 指令使用 slip 结构体的 perdis、pdis、odis 和 ejdis 分量，用于描述脱离原轨迹进入平滑轨迹点。其中，perdis 用于指定距离目标点百分比，pdis 用于指定距离目标点轴路径距离，odis 用于指定距离目标点姿态角度，ejdis 用于指定外轴距离目标点轴位置角度。这里如果缺省 s 参数，则默认使用系统变量$DFSLIP 作为 s 参数的值；如果指定了 s 参数，则成员分量的值会被更新到$DFSLIP 对应的分量中 在简化编程中，s 可由平滑百分比参数 sp 或平滑绝对值参数 sl 代替，格式见用法举例 注： ■ 当 perdis>=0 时，其余分量将被忽略； ■ 当同时指定 pdis、odis 和 ejdis 时，更靠近目标点的分量将会被选用； ■ 当上述分量均为 0 时，表示不平滑，但插补点不一定与目标点重合； ■ 当上述分量均小于 0 时，表示不平滑，目标点肯定会有插补点，即准停； 使用笛卡尔空间轨迹（即 LIN、CIR）平滑时，建议使用 pdis 平滑，可以使平滑轨迹在前后轨迹上均匀
sp	数据类型：double
	该参数指定平滑百分比。可替代平滑参数 s，用于不需要精确指定平滑大小的场合。格式为 sp:10%，表示当前行目标点的平滑距离是最大平滑距离的 10%
sl	数据类型：double
	该参数指定平滑距离。可替代平滑参数 s，用于需要精确指定平滑大小的场合。格式为 sl:10mm，表示当前行的平滑距离是 10mm
t	数据类型：tool 结构体
	该参数指定工具。用户需要在法兰末端安装的工具上自定义工具坐标系，该工具坐标系与工具固连，目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。这里如果缺省了 t 参数，则默认使用系统变量$DFTOOL 作为 t 参数的值；如果指定了 t 参数，则成员分量的值会被更新到$DFTOOL 对应的分量中 该参数是为了保证手动低速运行程序时限制 TCP 速度不大于 250mm/s
w	数据类型：wobj 结构体
	该参数指定工件坐标系。用户可以自定义工件坐标系。目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。在某些情况下，用户可能通过在特定的坐标系中编程更方便。另外，当工件移动位置时，只需重新标定工件坐标系即可，不需要修改用户程序。这里如果缺省了 w 参数，则默认使用系统变量$DFWOBJ 作为 w 参数的值；如果指定了 w 参数，则成员分量的值会被

名称	说明
	更新到$DFWOBJ 对应的分量中

以上结构体数据详细信息请参考第 2 章节。


注意

在程序调试器内，当执行<跳转>到 ccir 指令动作的时候，机器人为走 lin 指令的方式运动到目标点，而不是走圆弧轨迹。

用法举例

例一：常用方法

示例程序：

```
lin P:P1
lin P:P2
ccir P:P3
ccir P:P4
ccir P:P5
lin P:P6
```

程序执行示意图如图 3-3 所示。

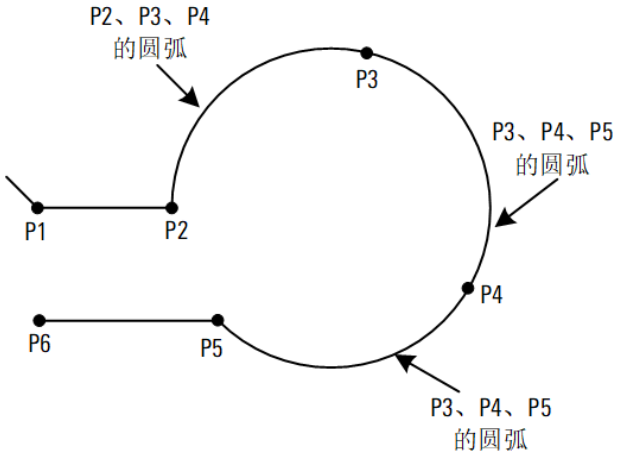


图 3-3 程序执行示意图

最初的 lin 指令为直线运动，如图 3-4（a）所示。在执行第 1 个 ccir 指令时，机器人在通过现在位置、该指令的目标位置、下一个 ccir 的目标位置的 3 点的圆周上运动（如图 3-4（b）所示）。当程序执行到下一个动作指令不是 ccir 运动指令时，最后的 ccir 运动指令的目标点被视为圆弧的终点。在向着终点运动时，机器人在通过之前一个 ccir 指令的目标位置、现在位置、该指令的目标位置的 3 点的圆周上运动（如图 3-4（c）所示）。

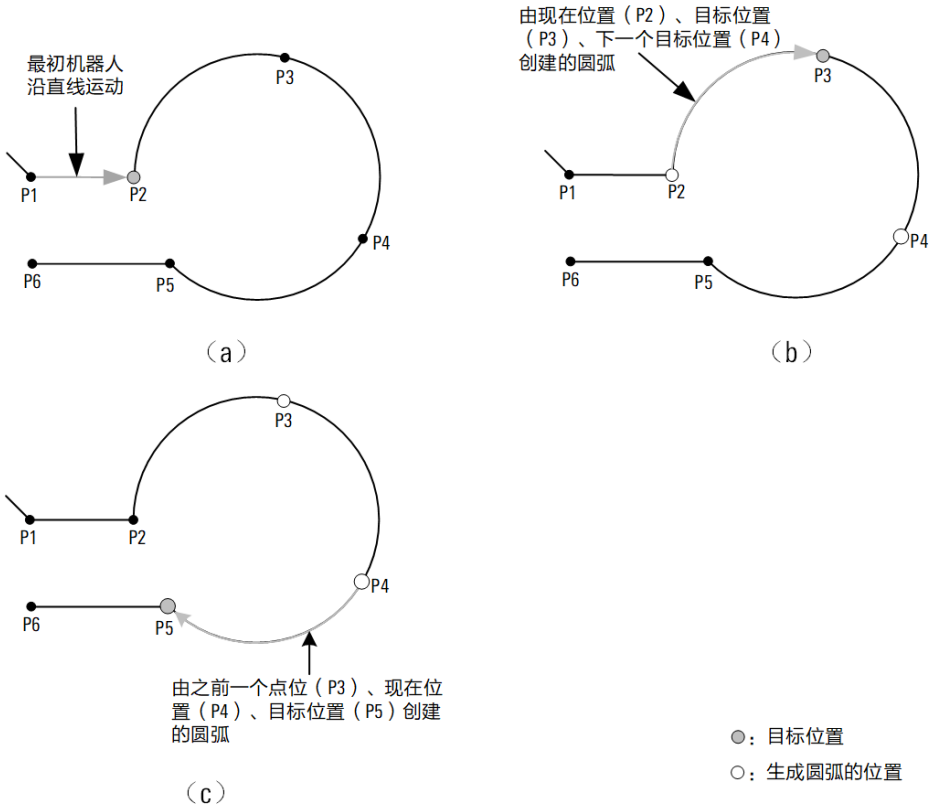


图 3-4 程序分布执行示意图

例二：圆弧朝向的判断

示例程序：

```
lin P:P1
lin P:P2
ccir P:P3
ccir P:P4
```

执行上述程序的第三行时，机器人通过经由 P2、P3、P4 的 3 点的圆弧，并向着 P3 运动。通过此圆弧向着 P3 的路径如图 3-5 所示，机器人沿着 P2→P3→P4 朝向的圆弧一直运动到 P3。

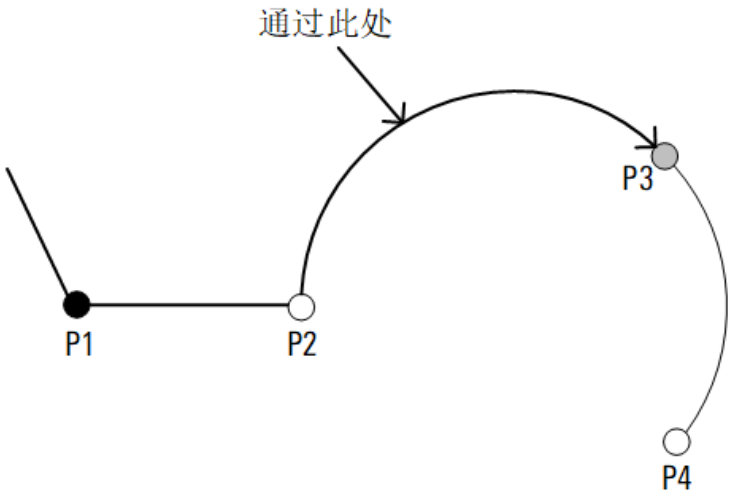


图 3-5 圆弧朝向判断程序示意图

例三：暂停并点动后再继续移动

机器人在 ccir 指令的中途暂停，在该状态下再继续移动时，会沿着之前的运动轨迹再继续移动。若机器人在 ccir 指令的中途暂停并进行点动，当再继续移动时，机器人沿直线返回到暂停位置，然后按照之前的路径继续运动。

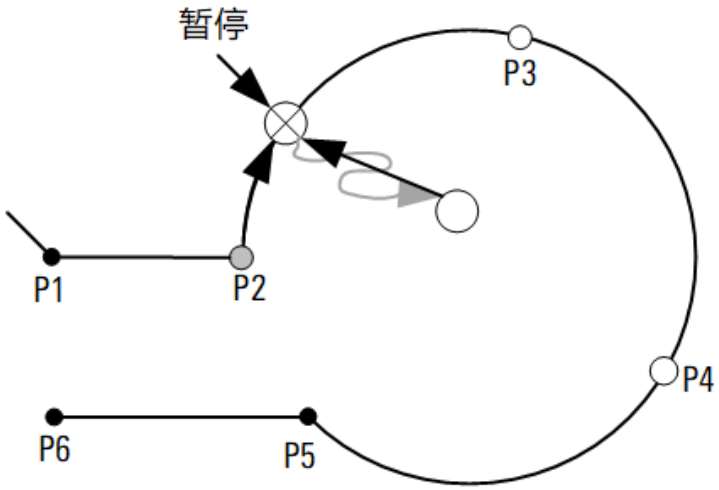


图 3-6 暂停并点动后再继续移动示意图

例四：下一个目标点的修正

将例一中的 P4 进行变更时，机器人沿着通过修正后的 P4 的圆弧再继续移动。

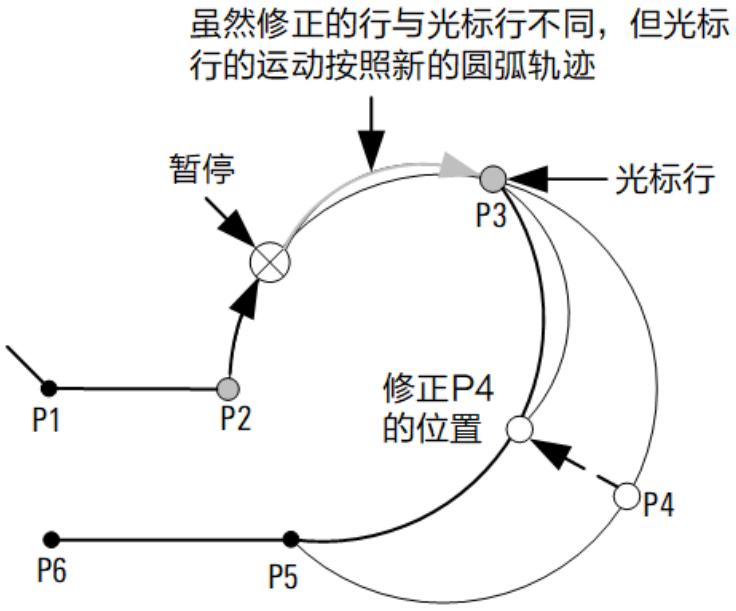


图 3-7 下一个目标点的修正示意图

例五：暂停后跳转

图 3-8 所示是编程为在向着 P2 的中途暂停，从 P3 再继续移动的示例。机器人从暂停位置，直线跳转到 P3，再按照 P3→P4→P5 的顺序移动。

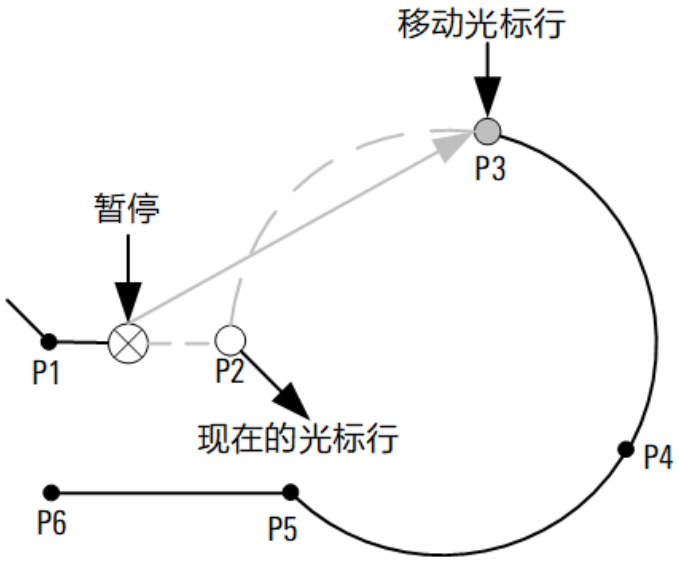


图 3-8 暂停后跳转示意图

3.2.7 叠加摆动指令

3.2.7.1 startweave(开启叠加摆动)

描述

startweave 指令用户开启叠加摆动。

格式

startweave weave:weavedata

参数

startweave 指令的参数详见表 3-7。

表 3-7 startweavelin 指令的参数

名称	说明
weavedata	数据类型：weavedata
	用于指定摆动轨迹的参数，包括： ■ weave_type（摆动类型） ■ frequency（摆动频率） ■ amplitude（振幅） ■ dwell_left（左停留时间） ■ dwell_right（右停留时间） ■ dwell_middle（中间停留时间） ■ swing_angle（夹角） ■ radius（半径） ■ axis（偏转轴） ■ rotation_angle（偏转角度）

名称	说明
	详细描述见 2.4.8 节

提示

- axis（偏转轴）可以缺省。
- rotation_angle （偏转角度）可以缺省。
- 当振幅较大时，可能出现超速告警情况，此时可尝试降低轨迹期望速度或振幅，避免告警。

用法举例

横摆：

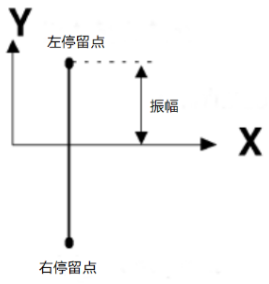
```
weavedata weavedata1
startweave weavedata1={simple,10,15,0,45}
tool t = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w = {{x 1000,y 0,z 500,a 0,b 10,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
pose p2 = {x 10,y 10,z 10,a 0,b 90,c 0,cfg 0}
lin p2,t,w,v
endweave
```

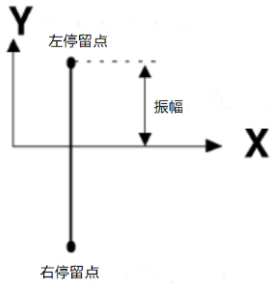
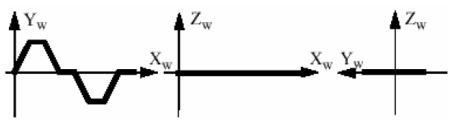
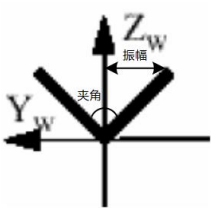
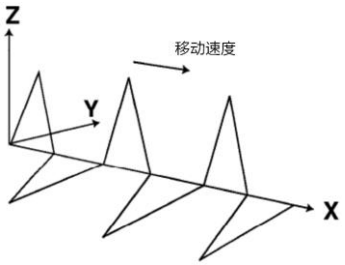
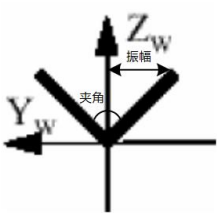
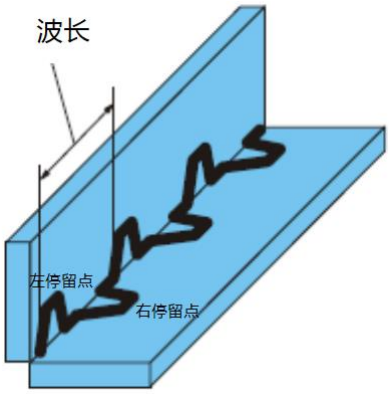
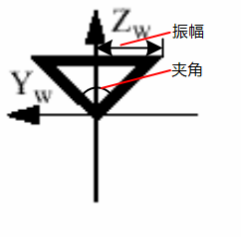
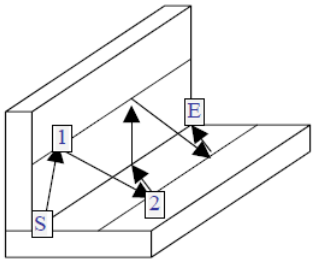
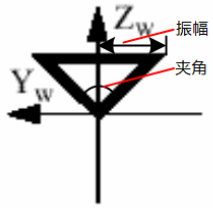
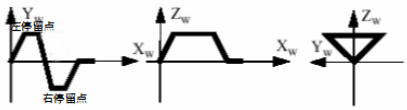
三角摆：

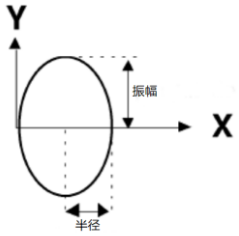
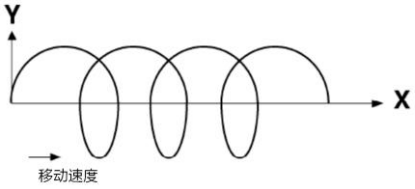
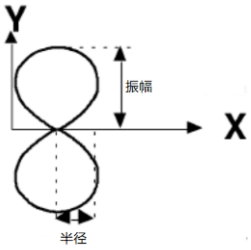
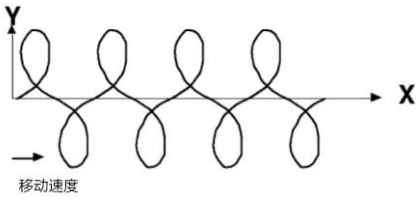
```
weavedata weavedata1 = {V_shape,2,15,90,1,1, 0,1}
startweavespa weavedata1
tool t = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w = {{x 1000,y 0,z 500,a 0,b 10,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
pose p2 = {x 10,y 10,z 10,a 0,b 90,c 0,cfg 0}
lin p2,t,w,v
endweave
```

叠加摆动需支持对叠加图样的图形参数进行调节，具体参考表 3-8：

表 3-8 叠加轨迹的图形参数

名称	叠加图样	组合图样	图形参数	备注
横摆			幅值	

名称	叠加图样	组合图样	图形参数	备注
线性锯齿横摆			幅值 左侧停留长度/时间 右侧停留长度/时间 中间停留长度/时间	若基于频率循环，则采用停留时间，若基于波长，则采用停留长度
“V”型摆			幅值 夹角	当夹角为 0 时，摆动方向与 z 轴重合
“V”型锯齿摆			幅值 夹角 左侧停留长度/时间 右侧停留长度/时间 中间停留长度/时间	若基于频率循环，则采用停留时间，若基于波长，则采用停留长度
空间三角摆			幅值 夹角	
空间三角锯齿摆			幅值 夹角 左侧停留长度/时间 右侧停留长度/时间 中间停留长度/时间	若基于频率循环，则采用停留时间，若基于波长，则采用停留长度

名称	叠加图样	组合图样	图形参数	备注
螺旋线			幅值 半径	
“8”字形			幅值 半径	

3.2.7.2 endweave(结束叠加摆动)

描述

endweave 指令用户结束叠加摆动。

格式

endweave

用法举例

```
weavedatalin weavedatalin1 = {2,15}
startweavelin weavedatalin1
lin p:{x 1000,y 500,z 500,a 0,b 90,c 0}
endweave
```

3.2.8 组合指令

“组合指令”的使用方法请参考本公司的《多机联动使用说明书》。

3.2.9 传送带

“传送带”相关指令的使用方法请参考本公司的《传送带跟踪使用说明书》。

3.2.10 软浮动

“软浮动”相关指令的使用方法请参考本公司的《软浮动使用说明书》。

3.2.11 轨迹补偿

3.2.11.1 startcompen(开始轨迹补偿)

描述

startcompen 指令用于开启机器人的轨迹补偿功能。

提示

角度补偿用于轨迹微调，累积的最大值请不要超过 90°。

格式

startcompen data:data

参数

startcompen 指令的参数详见表 3-9。

表 3-9 startcompen 指令的参数

名称	说明
data	数据类型：compendata
	该参数指定机器人轨迹补偿过程中 TCP 的最大速度、加速度、加加速度、角速度、角加速度、角加加速度

用法举例

```
const compendata data3 = {tcp_max_vel 200 ,tcp_max_acc 600 ,tcp_max_jerk 1000 ,ori_max_vel 200 ,ori_max_acc 50 ,ori_max_jerk 1000}
startcompen data:data3
```

3.2.11.2 compen(轨迹补偿)

描述

compen 指令用于 startcompen 和 endcompen 之间，用于对机器人的轨迹进行补偿。

格式

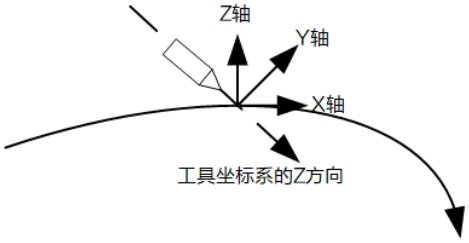
compen [x:],[y:],[z:],[a:],[b:],[c:],type:

参数

compen 指令的参数详见表 3-10。

表 3-10 compen 指令的参数

名称	说明
type	数据类型：枚举型
	可选择 TOOL，WOBJ、TOOL_PATH、MODIFY_PATH 或 WORLD 表示轨迹补偿量所参考的坐标系，各个坐标系类型说明如下： ■ TOOL：工具坐标系 ■ WOBJ：工件坐标系 ■ TOOL_PATH：工具-路径坐标系，定义如下图所示。

名称	说明
	 a) 坐标系 x 方向为主轨迹的切线方向； b) 坐标系 y 方向由坐标系 x 方向和工具坐标系的 z 方向叉乘确定； c) 坐标系 z 方向由坐标系 x 方向和坐标系 y 方向叉乘确定； 提示 如果主轨迹的切线方向和工具坐标系的 z 方向平行，则无法求出坐标系 y 方向，此时需给出告警。 ■ WORLD：世界坐标系 ■ MODIFY_PATH：一般情况下，MODIFY_PATH 与 TOOL_PATH 定义一致，当轨迹中存在折返点时，根据工具-路径坐标系定义，x 轴会反向，同时 y 轴也会反向。此时，MODIFY_PATH 的 y 轴并不反向，从而保证在进行 y 方向的补偿时，经过折返轨迹后补偿方向不变（a,b,c 在此坐标系下不补偿）
x	TCP 沿参考坐标系的 x 轴的平移补偿量
y	TCP 沿参考坐标系的 y 轴的平移补偿量
z	TCP 沿参考坐标系的 z 轴的平移补偿量
a	TCP 沿参考坐标系的 z 轴的旋转补偿量
b	TCP 沿参考坐标系的 y 轴的旋转补偿量
c	TCP 沿参考坐标系的 x 轴的旋转补偿量

用法举例

图 3-9 所示示例为机器人的轨迹相对于工件坐标系 X 方向偏移了 100 毫米。

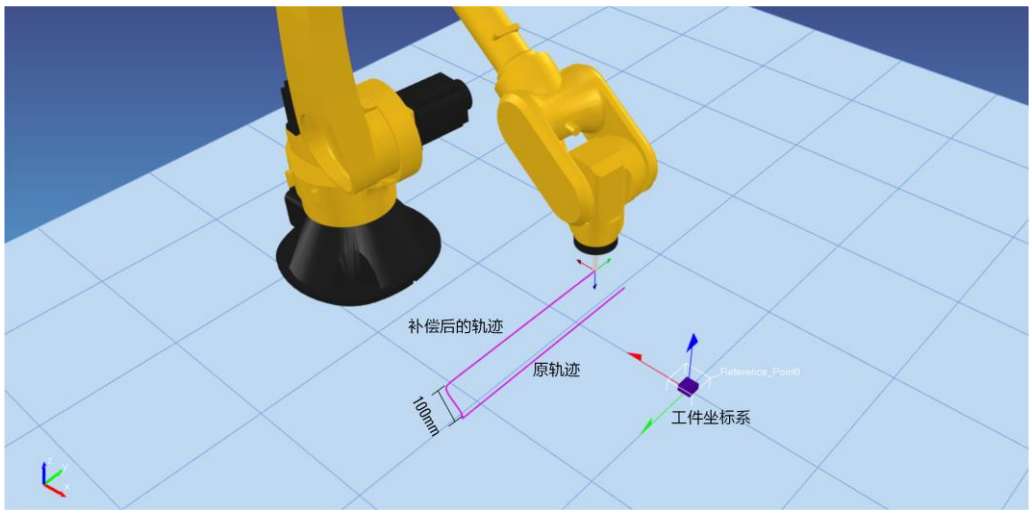


图 3-9 轨迹补偿示意图

图片中对应的程序示例为：

const compendata data1 = { tcp_max_vel 250 , tcp_max_acc 900 , tcp_max_jerk 6000 , ori_max_vel 50 , ori_max_acc 75 , ori_max_jerk 375 }

const pose p1 = { x 911.138, y -5.704, z 739.959, a -178.159, b 43.881, c 2.201, cfg 1, turn 000000b, ej1 9.000e+09, ej2 0.000, ej3 9.000e+09, ej4 9.000e+09, ej5 9.000e+09, ej6 9.000e+09 }

startcompen data:data1

compen x:100,y:0,z:0,a:0,b:0,c:0,type:"WOBJ" //相对于工件坐标系 X 方向偏移了 100mm。

lin p:p1,vl:50mm/s,sl:0mm,t:$tool0,w:$WORLD

endcompen


注意

单步或段调试模式下，轨迹补偿不生效。

3.2.11.3 endcompen(结束轨迹补偿)

描述

endcompen 指令用于结束机器人的轨迹补偿功能。

格式

endcompen

用法举例

参考第 3.2.11.2 章节。


注意

endcompen 后，直接将补偿后点位作为起点开始运行下一条不补偿的指令。如需先回到原轨迹，可在 endcompen 后增加一条 lin 指令返回原路径。

3.3 运动指令（适用于 SCARA 机器人）

3.3.1 movej(移动轴)

描述

movej 指令用于将机器人轴或外轴移动到一个指定的轴位置。所有轴同时到达目标轴位置。

格式

movej j:,[v: | vp:],[s: | sl: | sp:],[dura:]

参数

movej 指令的参数详见表 3-11。

表 3-11 movej 指令的参数

名称	说明
j	数据类型：joint

名称	说明
	该参数指定机器人各轴和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。程序中如果第一条执行的运动指令为 movej，则该 movej 指令的目标点 j1~j4 以及配置的外轴均不允许缺省
v	数据类型：speed
	该参数指定运动速度。movej 指令使用 speed 结构体变量中的 per 和 exj 分量，分别用于指定轴运动速度百分比和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 代替，格式见用法举例
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%
s	数据类型：slip
	该参数指定平滑距离。Movej 指令使用 slip 结构体的 perdis 和 ejdis 分量，用于描述脱离原轨迹进入平滑轨迹点。其中，perdis 用于指定距离目标点百分比，ejdis 用于指定外轴距离目标点轴位置角度。这里如果缺省 s 参数，则默认使用系统变量$DFSLIP 作为 s 参数的值；如果指定了 s 参数，则成员分量的值会被更新到$DFSLIP 对应的分量中 在简化编程中，s 可由平滑百分比参数 sp 代替，格式见用法举例 注： ■ 当 perdis>=0 时，其余分量将被忽略； ■ 当同时指定 ejdis 时，更靠近目标点的分量将会被选用； ■ 当上述分量均为 0 时，表示不平滑，但插补点不一定与目标点重合； ■ 当上述分量均小于 0 时，表示不平滑，目标点肯定会有插补点，即准停
sp	数据类型：double
	该参数指定平滑百分比。可替代平滑参数 s，用于不需要精确指定平滑大小的场合。格式为 sp:10%，表示当前行目标点的平滑距离是最大平滑距离的 10%
sl	数据类型：double
	该参数指定平滑距离。可替代平滑参数 s，用于需要精确指定平滑大小的场合。格式为 sl:10mm，表示当前行的平滑距离是 10mm
dura	数据类型：double
	该参数指定轨迹时间。用户可以直接指定运动时间而不是运动速度。如果用户指定了 dura 参数，系统将忽略 speed 参数，而是通过自动调整速度来满足 dura 参数指定的时间要求。dura 参数单位：秒

用法举例

```
//模糊指定速度和平滑大小
movej j:j1,vp:5%,sp:5%
movej j:{j1 0,j20,j3 0,j4 0, }
movej j:{j1 0}
movej j:{j1 20,ej1 30}
movej j:{ej1 10}
joint p = {0,0,0,0}
speed v = {per 50}
```

movej p,v

3.3.2 ptp(点到点)

描述

ptp 指令用于将机器人从一个点快速运动到另一个点而又不要求 TCP 点所走轨迹形状时。所有轴同时到达目标点。

格式

ptp p:,[v: | vp:],[s: | sl: | sp:],[t:],[w:],[dura:]

参数

ptp 指令的参数详见表 3-12。

表 3-12 ptp 指令的参数

名称	说明
p	数据类型：pose
	该参数指定机器人 TCP 目标位姿和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。cfg 分量缺省默认值为 0，turn 分量缺省默认值为-1。程序中如果第一条执行的运动指令为 ptp，则该 ptp 指令的目标点 X,Y,Z,A,B,C 以及配置的外轴均不允许缺省
v	数据类型：speed
	该参数指定运动速度。ptp 指令使用 speed 结构体变量中的 per 和 exj 分量，分别用于指定轴运动速度百分比和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 代替，格式见用法举例
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%
s	数据类型：slip
	该参数指定平滑距离。PTP 指令使用 slip 结构体的 perdis 和 ejdis 分量，用于描述脱离原轨迹进入平滑轨迹点。其中，perdis 用于指定距离目标点百分比，ejdis 用于指定外轴距离目标点轴位置角度。这里如果缺省 s 参数，则默认使用系统变量$DFSLIP 作为 s 参数的值；如果指定了 s 参数，则成员分量的值会被更新到$DFSLIP 对应的分量中 在简化编程中，s 可由平滑百分比参数 sp 代替，格式见用法举例 注： ■ 当 perdis>=0 时，其余分量将被忽略； ■ 当同时指定 ejdis 时，更靠近目标点的分量将会被选用； ■ 当上述分量均为 0 时，表示不平滑，但插补点不一定与目标点重合； ■ 当上述分量均小于 0 时，表示不平滑，目标点肯定会有插补点，即准停
sp	数据类型：double
	该参数指定平滑百分比。可替代平滑参数 s，用于不需要精确指定平滑大小的场合。格式为 sp:10%，表示当前行目标点的平滑距离是最大平滑距离的 10%
sl	数据类型：double

名称	说明
	该参数指定平滑距离。可替代平滑参数 s，用于需要精确指定平滑大小的场合。格式为 sl:10mm，表示当前行的平滑距离是 10mm
t	数据类型：tool 结构体
	该参数指定工具。用户需要在法兰末端安装的工具上自定义工具坐标系，该工具坐标系与工具固连，目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。这里如果缺省了 t 参数，则默认使用系统变量$DFTOOL 作为 t 参数的值；如果指定了 t 参数，则成员分量的值会被更新到$DFTOOL 对应的分量中 该参数是为了保证手动低速运行程序时限制 TCP 速度不大于 250mm/s
w	数据类型：wobj 结构体
	该参数指定工件坐标系。用户可以自定义工件坐标系。目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。在某些情况下，用户可能通过在特定的坐标系中编程更方便。另外，当工件移动位置时，只需重新标定工件坐标系即可，不需要修改用户程序。这里如果缺省了 w 参数，则默认使用系统变量$DFWOBJ 作为 w 参数的值；如果指定了 w 参数，则成员分量的值会被更新到$DFWOBJ 对应的分量中
dura	数据类型：double
	该参数指定轨迹时间。用户可以直接指定运动时间而不是运动速度。如果用户指定了 dura 参数，系统将忽略 speed 参数，通过自动调整速度来满足 dura 参数指定的时间要求。dura 参数单位：秒

以上结构体数据详细信息请参考第 2 章节。

用法举例

```
pose p = {x 266,y 88,z -50,a -34,b 0,c 0,cfg 4}
//模糊指定速度和平滑大小
ptp p:p1,vp:5%,sp:5%
ptp p,v:{per 10}
ptp p:{x 266,y 88,z -50,a 0,b 0,c 0}
ptp p,dura:10
tool t = {{x 0,y 0,z 0,a 0,b 0,c 0}}
wobj w = {{x 0,y 0,z 500,a 0,b 0,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
pose p2 = {x 266,y 88,z -50,a 0,b 0,c 0,cfg 0}
ptp p2,t,w
```

3.3.3 lin(直线运动)

描述

lin 指令用于将机器人 TCP 点沿直线路径运动到目标点位姿；位置移动和姿态转动同步。

格式

```
lin p:,[ v: | vl: | vp: ],[ s: | sl: | sp: ],[ t: ],[ w: ],[ dura: ]
```

参数

lin 指令的参数详见表 3-13。

表 3-13 lin 指令的参数

名称	说明
p	数据类型：pose
	该参数指定机器人 TCP 目标位姿和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。cfg 参数被忽略，与起点 cfg 参数保持一致，turn 分量缺省默认值为-1
v	数据类型：speed
	该参数指定运动速度。lin 指令使用 speed 结构体中的 tcp、ori 和 exj 分量，分别用于指定 TCP 点运动速度、TCP 姿态变化速度和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 或速度绝对值参数 vl 代替
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%
vl	数据类型：double
	该参数指定运动线速度。可替代速度参数 v，用于需要精确指定速度大小的场合。格式为 vl:500mm/s，表示当前行的 TCP 最大速度的 500mm/s
s	数据类型：slip
	该参数指定平滑距离。LIN 指令使用 slip 结构体的 perdis、pdis、odis 和 ejdis 分量，用于描述脱离原轨迹进入平滑轨迹点。其中，perdis 用于指定距离目标点百分比，pdis 用于指定距离目标点轴路径距离，odis 用于指定距离目标点姿态角度，ejdis 用于指定外轴距离目标点轴位置角度。这里如果缺省 s 参数，则默认使用系统变量$DFSLIP 作为 s 参数的值；如果指定了 s 参数，则成员分量的值会被更新到$DFSLIP 对应的分量中 在简化编程中，s 可由平滑百分比参数 sp 或平滑绝对值参数 sl 代替，格式见用法举例 注： ■ 当 perdis>=0 时，其余分量将被忽略； ■ 当同时指定 pdis、odis 和 ejdis 时，更靠近目标点的分量将会被选用； ■ 当上述分量均为 0 时，表示不平滑，但插补点不一定与目标点重合； ■ 当上述分量均小于 0 时，表示不平滑，目标点肯定会有插补点，即准停； 使用笛卡尔空间轨迹（即 LIN、CIR）平滑时，建议使用 pdis 平滑，可以使平滑轨迹在前后轨迹上均匀
sp	数据类型：double
	该参数指定平滑百分比。可替代平滑参数 s，用于不需要精确指定平滑大小的场合。格式为 sp:10%，表示当前行目标点的平滑距离是最大平滑距离的 10%
sl	数据类型：double
	该参数指定平滑距离。可替代平滑参数 s，用于需要精确指定平滑大小的场合。格式为 sl:10mm，表示当前行的平滑距离是 10mm
t	数据类型：tool 结构体
	该参数指定工具。用户需要在法兰末端安装的工具上自定义工具坐标系，该工具坐标系与工具固连，目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。这里如果缺省了 t 参数，则默认使用系统变量$DFTOOL 作为 t 参数的值；如果指定了 t 参数，则成员分量的值会被更新到$DFTOOL 对应的分量中 该参数是为了保证手动低速运行程序时限制 TCP 速度不大于 250mm/s

名称	说明
w	数据类型：wobj 结构体
	该参数指定工件坐标系。用户可以自定义工件坐标系。目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。在某些情况下，用户可能通过在特定的坐标系中编程更方便。另外，当工件移动位置时，只需重新标定工件坐标系即可，不需要修改用户程序。这里如果缺省了 w 参数，则默认使用系统变量$DFWOBJ 作为 w 参数的值；如果指定了 w 参数，则成员分量的值会被更新到$DFWOBJ 对应的分量中
dura	数据类型：double
	该参数指定轨迹时间。用户可以直接指定运动时间而不是运动速度。如果用户指定了 dura 参数，系统将忽略 speed 参数，而是通过自动调整速度来满足 dura 参数指定的时间要求。dura 参数单位：秒

以上结构体数据详细信息请参考第 2 章节。

用法举例

```
pose p = {x 266,y 88,z -50,a 0,b 0,c 0,cfg 0}
ptp p,v:{per 10}
//模糊指定速度和平滑大小
lin p:p1,vp:5%,sp:5%
//精确指定速度和平滑距离
lin p:p1, vl:100mm/s,sl:5mm
lin p:{x 266,y 88,z -50,a 0,b 90,c 0}
lin p,dura:10
//精确指定速度、平滑结构体
tool t = {{x 0,y 0,z 10,a 0,b 0,c 0}}
wobj w = {{x 0,y 0,z 0,a 0,b 0,c 0}}
speed v = {per 10,tcp 50,ori 5,exj 10}
slip s = { pdis 10, ejdis 10, odis 10 }
pose p2 = {x 266,y 88,z -50,a 0,b 0,c 0,cfg 0}
lin p2,t,w,v,s
```

3.3.4 cir(圆弧运动)

描述

cir 指令用于将机器人 TCP 点沿圆弧路径运动到目标点；平移运动和旋转运动同步。

格式

```
cir m:,p:,[ v: ],[ s: ],[ t: ],[ w: ],[ CA: ],[ dura: ]
```

参数

cir 指令的参数详见表 3-14。

表 3-14 cir 指令的参数

名称	说明
m	数据类型：pose
	该参数指定圆弧辅助点。起点、辅助点和目标点，这 3 点唯一确定了一段圆弧。辅助点中只有 x，y，z 分量被使用，他们中值为 9e+09 的分量将保持起点位置值，其他分量被忽略
p	数据类型：pose
	该参数指定机器人 TCP 目标位姿和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。cfg 参数被忽略，与起点 cfg 参数保持一致，turn 分量缺省默认值为-1
v	数据类型：speed
	该参数指定运动速度。cir 指令使用 speed 结构体中的 tcp、ori 和 exj 分量，分别用于指定 TCP 点运动速度、TCP 姿态变化速度和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 或速度绝对值参数 vl 代替，格式见用法举例
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%
vl	数据类型：double
	该参数指定运动线速度。可替代速度参数 v，用于需要精确指定速度大小的场合。格式为 vl:500mm/s，表示当前行的 TCP 最大速度的 500mm/s
s	数据类型：slip
	该参数指定平滑距离。cir 指令使用 slip 结构体的 perdis、pdis、odis 和 ejdis 分量，用于描述脱离原轨迹进入平滑轨迹点。其中，perdis 用于指定距离目标点百分比，pdis 用于指定距离目标点轴路径距离，odis 用于指定距离目标点姿态角度，ejdis 用于指定外轴距离目标点轴位置角度。这里如果缺省 s 参数，则默认使用系统变量$DFSLIP 作为 s 参数的值；如果指定了 s 参数，则成员分量的值会被更新到$DFSLIP 对应的分量中 在简化编程中，s 可由平滑百分比参数 sp 或平滑绝对值参数 sl 代替，格式见用法举例 注： ■ 当 perdis>=0 时，其余分量将被忽略； ■ 当同时指定 pdis、odis 和 ejdis 时，更靠近目标点的分量将会被选用； ■ 当上述分量均为 0 时，表示不平滑，但插补点不一定与目标点重合； ■ 当上述分量均小于 0 时，表示不平滑，目标点肯定会有插补点，即准停； 使用笛卡尔空间轨迹（即 LIN、CIR）平滑时，建议使用 pdis 平滑，可以使平滑轨迹在前后轨迹上均匀
sp	数据类型：double
	该参数指定平滑百分比。可替代平滑参数 s，用于不需要精确指定平滑大小的场合。格式为 sp:10%，表示当前行目标点的平滑距离是最大平滑距离的 10%。
sl	数据类型：double
	该参数指定平滑距离。可替代平滑参数 s，用于需要精确指定平滑大小的场合。格式为 sl:10mm，表示当前行的平滑距离是 10mm。
w	数据类型：wobj 结构体
	该参数指定工件坐标系。用户可以自定义工件坐标系。目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。在某些情况下，用户可能通过在特定的坐标系中编程更方便。另外，当

名称	说明
	工件移动位置时，只需重新标定工件坐标系即可，不需要修改用户程序。这里如果缺省了 w 参数，则默认使用系统变量$DFWOBJ 作为 w 参数的值；如果指定了 w 参数，则成员分量的值会被更新到$DFWOBJ 对应的分量中
CA	数据类型：double
	该参数指定圆心角（Circle Angle）。用户可以不直接指定目标点，而通过指定圆弧转过的圆心角的方式来指定目标点。如果用户指定了 CA 参数，则 p 参数只用来和辅助点一起确定圆弧的几何形状，而不是真正的目标点，真正的目标点系统通过用户指定的圆心角自动计算。CA 参数单位：度
dura	数据类型：double
	该参数指定轨迹时间。用户可以直接指定运动时间而不是运动速度。如果用户指定了 dura 参数，系统将忽略 speed 参数，通过自动调整速度来满足 dura 参数指定的时间要求。dura 参数单位：秒

以上结构体数据详细信息请参考第 2 章节。

用法举例

```
pose p3={x 266,y 88,z -50,a 0,b 0,c 0}
pose p4={x 308,y 52,z -50,a 0,b 0,c 0}
tool tool1 = {{x 0,y 0,z 0,a 0,b 0,c 0}}
cir p3,p4,tool1
//指定圆弧圆心角为 360 度
cir p3,p4,CA:360
//模糊指定速度和平滑大小
cir m:p3,p:p4,vp:5%,sp:5%
//精确指定速度和平滑距离
cir m:p3,p:p4,vl:500mm/s,sl:5mm
//精确指定速度和平滑结构体
tool t1 = {{x 0,y 0,z 0,a 0,b 0,c 0}}
wobj w1 = {{x 0,y 0,z 0,a 0,b 0,c 0}}
speed v2 = {per 10,tcp 50,ori 5,exj 10}
slip s = { pdis 10, ejdis 10, odis 10 }
cir p3,p4,t1,w1,v2, s
```

注意

- 在单步或段调试模式下，cir 指令分为两步执行，第一步运行至辅助点，第二步运行至目标点。
- 虽然用户可以通过使用参数缺省来简化程序，但仍建议用户在编写运动指令时尽量将参数写全，以防止由于编程人员的疏忽而导致的实际的运动与用户预期不符，进而导致一些不必要的损失或伤害。

3.3.5 spl(样条曲线运动)

描述

spl 指令使机器人平滑不停顿地经过示教点。

格式

```
spl p: , [ v: | vp: ],[ s: | sl: | sp: ],[ t: ],[ w: ]
```

参数

spl 指令的参数详见表 3-5。

表 3-15 spl 指令的参数

名称	说明
p	数据类型：pose
	该参数指定机器人 TCP 目标位姿和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。cfg 参数被忽略，与起点 cfg 参数保持一致，turn 分量缺省默认值为-1
v	数据类型：speed
	该参数指定运动速度。lin 指令使用 speed 结构体中的 tcp、ori 和 exj 分量，分别用于指定 TCP 点运动速度、TCP 姿态变化速度和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 或速度绝对值参数 vl 代替
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%
s	数据类型：slip
	该参数指定平滑距离。LIN 指令使用 slip 结构体的 perdis、pdis、odis 和 ejdis 分量，用于描述脱离原轨迹进入平滑轨迹点。其中，perdis 用于指定距离目标点百分比，pdis 用于指定距离目标点轴路径距离，odis 用于指定距离目标点姿态角度，ejdis 用于指定外轴距离目标点轴位置角度。这里如果缺省 s 参数，则默认使用系统变量$DFSLIP 作为 s 参数的值；如果指定了 s 参数，则成员分量的值会被更新到$DFSLIP 对应的分量中 在简化编程中，s 可由平滑百分比参数 sp 或平滑绝对值参数 sl 代替，格式见用法举例 注： ■ 当 perdis>=0 时，其余分量将被忽略； ■ 当同时指定 pdis、odis 和 ejdis 时，更靠近目标点的分量将会被选用； ■ 当上述分量均为 0 时，表示不平滑，但插补点不一定与目标点重合； ■ 当上述分量均小于 0 时，表示不平滑，目标点肯定会有插补点，即准停； 使用笛卡尔空间轨迹（即 LIN、CIR）平滑时，建议使用 pdis 平滑，可以使平滑轨迹在前后轨迹上均匀
sl	数据类型：double
	该参数指定平滑距离。可替代平滑参数 s，用于需要精确指定平滑大小的场合。格式为 sl:10mm，表示当前行的平滑距离是 10mm
sp	数据类型：double
	该参数指定平滑百分比。可替代平滑参数 s，用于不需要精确指定平滑大小的场合。格式为 sp:10%，表示当前行目标点的平滑距离是最大平滑距离的 10%
t	数据类型：tool 结构体
	该参数指定工具。用户需要在法兰末端安装的工具上自定义工具坐标系，该工具坐标系与工具固连，目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。这里如果缺省了 t 参数，则默认使用系统变量$DFTOOL 作为 t 参数的值；如果指定了 t 参数，则成员分量的值会被更新到$DFTOOL 对应的分量中 该参数是为了保证手动低速运行程序时限制 TCP 速度不大于 250mm/s

名称	说明
w	数据类型：wobj 结构体
	该参数指定工件坐标系。用户可以自定义工件坐标系。目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。在某些情况下，用户可能通过在特定的坐标系中编程更方便。另外，当工件移动位置时，只需重新标定工件坐标系即可，不需要修改用户程序。这里如果缺省了 w 参数，则默认使用系统变量$DFWOBJ 作为 w 参数的值；如果指定了 w 参数，则成员分量的值会被更新到$DFWOBJ 对应的分量中

用法举例

示例程序：

```
lin P:P1

lin P:P2

spl P:P3

spl P:P4

spl P:P5

spl P:P6

lin P:P7
```

程序运行如图 3-1 所示，从程序点 2 平滑移动到程序点 6。

- 沿根据程序点 2、3、4 的示教点形成的轨迹运行到程序点 3。
- 沿根据程序点 2、3、4 的示教点形成的 3、4 区间的轨迹和根据程序点 3、4、5 的示教点形成的 3、4 区间的轨迹合成的轨迹运行到程序点 4。
- 沿根据程序点 3、4、5 的示教点形成的 4、5 区间的轨迹和根据程序点 4、5、6 的示教点形成的 4、5 区间的轨迹合成的轨迹运行到程序点 5。
- 沿根据程序点 4、5、6 的示教点形成的轨迹运行到程序点 6。

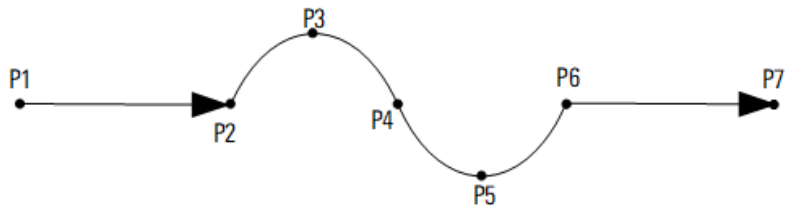


图 3-10 程序执行示意图

3.3.6 jump(门形运动)

描述

jump 指令用于将机器人从一个点快速抬起并运动到另一个点而又不要求 TCP 点所走轨迹形状时。所有轴同时到达目标点。

格式

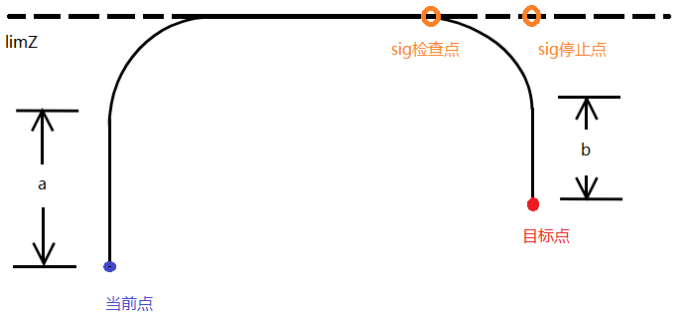
```
jump p:,[ v: | vp: ],[ t: ],[ w: ],[ a: ],[ b: ],[ Z: ],[ ctr: ],[ s: ],[ sig: ]
```

参数

jump 指令的参数详见表 3-16。

表 3-16 jump 指令的参数

名称	说明
p	数据类型：pose
	该参数指定机器人 TCP 目标位姿和外轴的目标点位置。参数中值为 9e+09 的结构体分量将保持起点位置不变。cfg 分量缺省默认值为 0，表示右手系，turn 分量缺省默认值为-1，表示运行至最近的 turn 值。程序中如果第一条执行的运动指令为 jump，则该 jump 指令的目标点 X,Y,Z,A,B,C 以及配置的外轴均不允许缺省
v	数据类型：speed
	该参数指定运动速度。ptp 指令使用 speed 结构体变量中的 per 和 exj 分量，分别用于指定轴运动速度百分比和外轴运动速度。这里如果缺省了 v 参数，则默认使用系统变量$DFSPEED 作为 v 参数的值；如果指定了 v 参数，则成员分量的值会被更新到$DFSPEED 对应的分量中 在简化编程中，v 可由速度百分比参数 vp 代替，格式见用法举例
vp	数据类型：double
	该参数指定运动速度百分比。可替代速度参数 v，用于不需要精确指定速度大小的场合。格式为 vp:10%，表示当前行速度是机器人最大速度的 10%
t	数据类型：tool 结构体
	该参数指定工具。用户需要在法兰末端安装的工具上自定义工具坐标系，该工具坐标系与工具固连，目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。这里如果缺省了 t 参数，则默认使用系统变量$DFTOOL 作为 t 参数的值；如果指定了 t 参数，则成员分量的值会被更新到$DFTOOL 对应的分量中 该参数是为了确定机器人 TCP 位置，同时保证手动低速运行程序时限制 TCP 速度不大于 250mm/s
w	数据类型：wobj 结构体
	该参数指定工件坐标系。用户可以自定义工件坐标系。目标点 p 指定的就是工具坐标系在 w 参数指定的坐标系中的位姿。在某些情况下，用户可能通过在特定的坐标系中编程更方便。另外，当工件移动位置时，只需重新标定工件坐标系即可，不需要修改用户程序。这里如果缺省了 w 参数，则默认使用系统变量$DFWOBJ 作为 w 参数的值；如果指定了 w 参数，则成员分量的值会被更新到$DFWOBJ 对应的分量中
a	数据类型：double
	拱形指令从起始点垂直上升的相对距离，单位 mm，缺省值为 25mm
b	数据类型：double
	拱形指令到达目标点前垂直下降的距离，单位 mm，缺省值为 25mm
Z	数据类型：double
	拱形指令 Z 方向允许达到的最大绝对高度，单位 mm，缺省值为 0（最大限高）
s	数据类型：bool
	是否平滑（不到达目标点，而是在降速之前拐入下一条语句），true 表示平滑，缺省值为 false
ctr	数据类型：control
	设置拱形指令垂直上升段和下降段的速度、加速度、减速度。当需要单独指定上升或下降段的速

名称	说明
	度、稳定性时需要指定，详见第二章 control 结构体
sig	数据类型：bool
	在 jump 指令下降前（图中 sig 检查点），系统会检查 sig 参数，true 则停在目标点正上方（图中 sig 停止点），三轴不下降，false 表示到达目标点。缺省为 false。  该处必须为一个 bool 型或者能隐式转换为 bool 型的表达式。该表达式与 if，while 语句中的判断语句相同 举例来说，以下表达式都属于这种类型的表达式： ■ 1 ■ true ■ getdi(5) ■ getdi(5) == true ■ getdi(5) && getdi(6) ■ counter>=20 ■ T(3.4)

以上结构体数据详细信息请参考第 2 章节。

用法举例

pose p3={x 266,y 88,z -50,a 0,b 0,c 0}

jump p:p1,vp:5%,t:$FLANGE,w:$WORLD,ctr:ctr1,s:false

注意

- jump 指令只适用于 scara 机器人，对六轴机器人无效。
- jump 指令为追求节拍最快，低速动作时的轨迹会低于高速动作时的轨迹，因此，即使确认高速没有发生碰撞，低速动作也可能发生碰撞。
- 如果起始点、目标点超过了 Z 限高，则会告警“12042：Jump 轨迹始末点超笛卡尔 Z 轴限制”。
- 如果垂直上升距离 a 或垂直下降距离 b 设置超过了 Z 限高，则 Z 限高被满足，a 或 b 不会被满足，且轨迹退化为直角门型轨迹。

3.3.7 传送带

“传送带”相关指令的使用方法请参考本公司的《传送带跟踪使用说明书》。

3.4 过程控制

3.4.1 waittime(延时等待)

描述

程序等待指定的时长。执行 waittime 指令时，将自动暂停运动指令的提前规划。

格式

waittime time:

参数

waittime 指令的参数详见表 3-17。

表 3-17 waittime 指令的参数

名称	说明
time	数据类型：double
	该参数指定等待的时间，参数取值应大于 0，单位为秒

用法举例

waittime time:5 //表示等待 5 秒的时间

3.4.2 waituntil(条件等待)

描述

程序等待，直到某个条件表达式的值为真或者超时，则继续向下执行。

格式

waituntil cond:,[maxtime:],[timeoutflag:]

参数

waituntil 指令的参数详见表 3-18。

表 3-18 waituntil 指令的参数

名称	说明
cond	数据类型：bool
	该参数为一个 bool 的条件表达式，指定程序等待的条件
maxtime	数据类型：double
	最大等待时间，单位秒。当等待的时间超过该参数设置的时间时，即使 cond 条件表达式的值没有变为 true，程序也将继续向下执行 该参数缺省时默认值为无穷大，即如果程序中不写这个参数，程序会一直等待，直到 cond 条件表达式的值变为 true
timeoutflag	数据类型：bool
	当 waituntil 指令中指定了 maxtime 参数时，则可同时指定 timeoutflag 参数。这个参数的值部分必须是一个 bool 型变量。waituntil 指令执行完后可以根据这个变量的值判断 waituntil 指令是否超时。如果该变量值为 true，表示超时 cond 条件表达式的值仍然没有变为 true；如果该变量值为 false，表示超时之前 cond 条件表达式的值就已经变为 true

用法举例

waituntil getdi(6) //等待第 6 通道 di 信号值为 true

```
waituntil getdi(6),maxtime:5 //等待第 6 通道 di 信号值为 true 时,才会执行后面的语句。最大等待时间为 5s
bool time_flag
waituntil getdi(6),maxtime:2,timeoutflag:time_flag //等待第 6 通道 di 信号值为 true 时,才会执行后面的语句。最大等待时间为 2s
if(time_flag)
print "timeout." //等待时间超过 2s
else
print "cond satisfy within 2 sec." //等待时间小于 2s
endif
```

3.4.3 pause(暂停)

描述

暂停程序运行。程序会在 pause 语句的前一句执行完毕后进入暂停状态，必须通过示教器的开始或外部控制按键继续程序运行。

格式

```
pause
```

用法举例

```
func void main()
movej j:j1,vp:5%,sp:-1%
pause //程序运行到这里将暂停，通过示教器开始按键继续程序//执行
movej j:j2,vp:5%,sp:-1%
endfunc
```


注意

当 pause 指令后有运动指令，且程序运行至 pause 指令时，程序不会进入暂停状态，当已前瞻的轨迹全部运行后，程序才会进入暂停状态。故当机器人需要立即停止时，请不要使用 pause 指令。

3.4.4 exit(退出程序)

描述

程序退出执行。

exit 与 return 区别为：return 返回到上一级函数继续执行；而 exit 则不管当前程序在执行哪个函数，整个程序都将直接退出，再次运行程序时从主函数第一行开始运行。

格式

```
exit
```

用法举例

```
func void main()
movej j:j1,vp:5%,sp:-1%
//如果第 6 通道 di 信号为 true，则退出程序执行
```

```
if(getdi(6))
exit
endif
movej j:j2,vp:5%,sp:-1%
endfunc
```

3.4.5 restart(重启程序)

描述

The program is executed again. When the program executes to this instruction, the program pointer will return to the first line of the main function to restart execution.

格式

```
restart
```

用法举例

```
func void main()
movej j:{j1 10}
movej j:{j1 20}
restart
endfunc
```

3.4.6 stopmove(停止当前运动)

描述

停止前瞻运动语句。该指令一般用在中断处理函数中，用于停止被中断函数中正在执行的运动。与 startmove 配合使用，与需根据 startmove 后面参数来决定启动后轨迹要跳的条数。

格式

```
stopmove [ type: ]
```

stopmove 指令的详细用法参见第 6.6 章节。

参数

stopmove 指令的参数详见表 3-19。

表 3-19 stopmove 指令的参数

名称	说明
type	数据类型：stoptype
	该参数指定运动停止的减速类型，包括普通停止和快速停止，参见第 2.6.4 章节 stoptype 类型定义。该参数可缺省，缺省值为 general 普通停止。 ■ 普通停止是按照当前最大加速度和加加速度停止。 ■ 快速停止与普通停止相比停止的更快，采用 5 倍加速度和加加速度。

3.4.7 startmove(重新启动停止的运动)

描述

重新启动停止的运动。该指令一般用在中断处理函数中，与 stopmove 配合使用，继续执行被 stopmove 指令停止的运动，需根据 startmove 后面参数来决定启动后轨迹要跳的条数。

格式

startmove [skip:]

startmove 指令的详细用法参见第 6.6 章节。

参数

startmove 指令的参数详见表 3-20。

表 3-20 startmove 指令的参数

名称	说明
skip	数据类型：int
	skip 后的数字表示从停止的行数起，重新启动后轨迹要跳的条数。其中：0 表示不跳，即先回到停止点（CP 轨迹以直线回，PTP 轨迹以 PTP 轨迹回），再继续原轨迹。1 表示跳一条，则会直接回到被中断的轨迹目标点（CP 轨迹以直线回，PTP 轨迹以 PTP 轨迹回）。2 表示跳 2 条，以此类推。该参数缺省时默认值为 0  注意 当 skip 数值大于当前运动指令条数的时候，会运动到起始点后再运行后续的程序，即重新开始运行程序。

3.5 辅助指令

3.5.1 print(打印输出)

描述

print 指令用于打印输出到某个位置。可以使用该函数打印一个或多个表达式的值到 HMI 消息栏、优盘、某个指定的文件或者一个字符串，该指令多用于程序调试，当然也可用于用户输出日志。

格式

print [to:],[to:],[to:],[to:],[tostr:],[filepath:],[precision:],[numbase:],{ argtoprint: }

参数

print 指令的参数详见表 3-21。

表 3-21 print 指令的参数

名称	说明
to	数据类型：printto
	指定输出定向，可以是 off，hmi，file 或 console。 ■ off 表示关闭任何打印输出；

名称	说明
	■ hmi 表示打印输出到 HMI 的消息栏； ■ file 表示打印输出到指定文件，文件路径由 filepath 参数指定； ■ console 表示打印输出到终端，该选项用户一般用不到。 可以同时存在多个 to 参数（最多 4 个），用于同时输出到 hmi 和 file，print 指令中只要存在值为 off 的 to 参数，off 前指定的参数即无效。to 参数为模态参数，只要在一个 print 指令中指定了 to 参数，之后将一直保持该输出定向设置直到下一次指定 to 参数。程序被加载后默认的配置为只输出到 hmi
tostr	数据类型：string
	输出到某个 string 变量。可以使用该参数将变量对应的字符串输出到某个 string 类型的变量中
filepath	数据类型：string
	当某个 to 参数的值为 file 时，该参数用于指定输出到文件的路径。该路径以 script 文件夹作为根文件夹。如指定文件路径为“mylog/mylog.log”，则实际输出到“script/mylog/mylog.log”中。此外，用户需确保目录“script/mylog”存在，否则运行时会报错
precision	数据类型：int
	用于指定 double 类型变量输出的有效数字位数 d
numbase	数据类型：num_base
	用于指定 int 类型变量输出的数制格式。可以为 hex(16 进制)，dec(10 进制)
argtoprint	数据类型：anytype
	要打印的表达式，argtoprint 参数可以有 1 个或多个，最终系统将这些表达式的值转为字符串并连接起来打印输出到指定位置。有一个比较特殊的常量“endl”表示换行符，打印 endl 将会使打印的字符串换到新的一行

用法举例

```
print to:hmi, "hello world!"
int i = 1
while(i <= 3)
print to:file,filepath:"mylog","loop ",i,endl
i++
endwhile
print precision:3,"PI:",PI
print "255 =",hex,255,"h"
```

程序运行将输出：

HMI 消息栏：

hello world!

mylog 文件中：

loop 1

loop 2

loop 3

PI:3.14

255 = ffh

3.5.2 scan(扫描输入)

描述

scan 指令用于扫描一个字符串，将其中使用某个分隔符分隔的一系列子串按类型读入到一系列的变量中。

格式

scan from:,delimiter:,{ argtosave: }

参数

scan 指令的参数详见表 3-22。

表 3-22 scan 指令的参数

名称	说明
from	数据类型：string
	待扫描的字符串
delimiter	数据类型：string
	分隔符
argtosave	数据类型：int/double/bool/string
	要保存到的变量，argtosave 参数可以有 1 个或多个，这些变量必须是之前定义过的，系统会将 from 参数中一系列字符串按照每个变量的类型解析为对应的数值并存储到每个变量中

用法举例

```
double x,y,z
bool b
scan from:"1.1,1.2,1.3,true",delimiter:",",x,y,z,b
print x, " ",y, " ",z, " ",b
```

程序运行将输出：

```
1.1 1.2 1.3 true
```

3.5.3 import(导入 ARL 模块)

描述

导入一个 arl 模块。

格式

import modpath:

参数

import 指令的参数详见表 3-23。

表 3-23 import 指令的参数

名称	说明
modpath	数据类型：string
	该参数指定导入的 arl 文件路径，如果是和当前文件在同一个目录下，可以直接输入文件名。

import 指令的具体用法参见第 8 章节。

3.5.4 velset(速度调节)

描述

velset 指令可用于降低或提升之后所有运动指令的编程规划速度倍率，也可用于设置运动段最大速度。

格式

velset override:,max:

参数

velset 指令的参数详见表 3-24。

表 3-24 velset 指令的参数

名称	说明
override	数据类型：int
	编程规划速度倍率的百分比值，包括轴速度、TCP 速度、ORI 速度，100%表示使用程序设定速度，参数取值范围为 0~100 编程规划速度倍率用于运动规划，程序运行中操作示教器进行速度倍率调节在此基础上生效，即机器人运行的实际速度倍率是规划倍率和示教器速度倍率的乘积
max	数据类型：int
	编程规划最大 TCP 速度，单位 mm/s 指令中的最大速度限制的是编程规划速度，不是对实际运行速度的限制，即当编程设定速度大于指令中设置的最大速度时，系统以指令中的 max 值进行速度规划，但实际运行速度还受示教器速度倍率影响，由于速度倍率一定小于 100%，因此实际运行速度一定不会超过 max。

用法举例

velset override:50, max:800

所有编程规划速度降低到程序设定速度的 50%，TCP 速度在任何情况下不允许超过 800mm/s。

3.5.5 accset(加速度调节)

描述

accset 指令调节机器人运动的加速度和加加速度，常用于机器人加持易碎负载时，可允许较低的加速度和减速度，结果使机器人运动更加柔顺，也可适当提速。

格式

accset acc:,ramp:

参数

accset 指令的参数详见表 3-25。

表 3-25 accset 指令的参数

名称	说明
acc	数据类型：double
	机器人实际加速度和减速度相对于最大值的百分比形式，100%表示使用系统最大加速度。最大值：300%，指令输入小于 20%时，取 20%做为实际值
ramp	数据类型：double
	机器人实际加加速度相对于最大值的百分比形式，100%表示使用系统最大加加速度。最大值：300%，指令输入小于 10%时，取 10%做为实际值

用法举例

accset acc:50, ramp:300

加速度限制到最大值的 50%.

accset acc:300, ramp:50

加加速度限制到最大值的 50%.

3.5.6 toolload(工具负载设置)

描述

toolload 指令用于指定机器人的工具负载。

格式

toolload toolinertia:

参数

toolload 指令的参数详见表 3-26。

表 3-26 toolload 指令的参数

名称	说明
toolinertia	数据类型：ToolInertiaPara
	该参数指定机器人工具负载的质量、质心、惯性主轴方向及转动惯量。可以自定义 ToolInertiaPara 类型的数据作为 toolload 的参数。也可以将系统变量$TOOL_INERTIA[i]直接作为参数，切换到第 i 号工具负载

用法举例

CentroidPos cen_pos = {x 15, y 25, z 100} //定义质心位置

InertiaTensor inertia_tensor = {Ixx 30, Ixy 40, Ixz 50, Iyy 60, Iyz 70, Izz 80} //定义惯性张量

```
ToolInertiaPara tip //定义工具负载
tip.m = 30 //质量
tip.centroid_pos = cen_pos
tip.centroid_dir = cen_dir
tip.moment_inertia = inertia
toolload tip //切换工具负载

//定义工具负载并初始化
ToolInertiaPara tip1 = {20, {30, 35, 40}, {45, 50, 55}, {60, 65, 70}}
toolload toolinertia:tip1

toolload $TOOL_INERTIA[3] //
切换到系统中[3]号工具的负载
```

3.5.7 toolswitch(工具负载切换)

描述

toolswitch 指令用于切换当前机器人的工具负载序号。

格式

toolswitch toolindex: ,mu:

参数

toolswitch 指令的参数详见表 3-27。

表 3-27 toolswitch 指令的参数

名称	说明
toolindex	数据类型：int
	该参数指定机器人工具负载的质量、质心、惯性主轴方向及转动惯量。将系统变量 $TOOL_INERTIA[i]直接作为参数，切换到第 i 号工具负载
mu	切换负载的机械单元名称

用法举例

```
toolswitch toolindex:2
movej j:{j1 10,j2 20,j3 30,j4 40,j5 50,j6 60}
```

3.5.8 startdetect（碰撞检测开启）

描述

碰撞检测开启指令。

格式

startdetect cid: ,mu:

参数

参数	名称	含义
cid	碰撞检测条件号	取值范围是 0~16，说明如下： ■ 0：代表手动 jog 对应的条件号 ■ 1-16：代表程序运行对应的条件号
mu	机械单元名称	开始碰撞检测功能的机械单元名称。

用法举例

startdetect cid:1 ,mu:R1

3.5.9 enddetect（碰撞检测关闭）

描述

碰撞检测关闭指令。

格式

enddetect mu:

参数

参数	名称	含义
mu	机械单元名称	关闭碰撞检测功能的机械单元名称。

用法举例

enddetect mu:R1

3.6 功能包

关于弧焊、码垛、折弯等功能的相关指令，其具体用法请参考本公司的各个功能包对应的使用说明书。


提示

SCARA 机器人目前不支持上述功能包相关指令。

4 逻辑控制指令

4.1 if (条件语句)

格式

```
if(bool 型表达式)
……
elseif(bool 型表达式)
…….
elseif(bool 型表达式)
……
else
…….
endif
```

描述

条件执行语句。

系统将从上向下依次计算 if 后的 bool 型表达式的值，直到某一个表达式的值为真，则执行这个 if 和下一个 elseif 或 else 之间的指令，执行完后跳到 endif 后继续执行。

其中 elseif 的个数不限，也可以没有 elseif 和/或 else 的部分。

用法举例

```
int count = 1
if(count == 1)
setdo(5,true)
endif
if(count > 2)
setdo(5,true)
elseif(count < 2)
setdo(6,true)
else
setdo(7,true)
endif
if(count > 2)
setdo(5,true)
else
setdo(6,true)
endif
```

4.2 compact if (紧凑条件语句)

格式

if(bool 型表达式) ……

描述

紧凑条件语句。

当条件语句只有一条 if，并且当条件表达式值为 true 时，执行的指令行只有一行时，此时可以使用紧凑型条件语句将 3 行指令缩减为 1 行。

用法举例

if(getdi(3))　　setdo(3,true)

4.3 while(while 循环)

格式

while (bool 型表达式)
……
endwhile

描述

循环执行语句。

当 while 后的 bool 型表达式的值为真时，则执行 while 到 endwhile 之间的指令，执行完后重新计算 bool 型表达式的值，如果为真，则重新执行 while 到 endwhile 之间的指令，如此循环，直到 while 后的 bool 型表达式的值为假，则跳到 endwhile 后继续执行。

用法举例

int a = 0
while(a<3)
a++
endwhile
print a //while 循环体会被执行 3 次，结果输出“3”

4.4 repeat(repeat 循环)

格式

repeat
……
until (bool 型表达式)

描述

循环执行语句。

repeat 和 until 循环体内的指令至少会被执行 1 次。当 until 后的 bool 型表达式的值为真时，则退出循环；反之，如果 until 后的 bool 型表达式的值为假时，则继续 repeat 循环。

用法举例

```
int a = 0
repeat
a++
until(a >= 3)
print a //repeat 循环体会被执行 3 次，结果输出“3”
```

4.5 loop(无限循环)

格式

```
loop
……
endloop
```

描述

无限循环执行语句。

该循环指令相当于 while(1)或 while(true)。循环永远都不会停止，除非内部执行 break/exit/restart/return 指令。

用法举例

```
int a = 5
loop
if(a-- == 0) break
endloop
print a //结果输出“-1”
```

4.6 for(for 循环)

格式

```
for (初始化语句；bool 型表达式；迭代表达式)
……
endfor
```

描述

循环执行语句。

首先执行初始化语句，然后判断 bool 型表达式是否为真，如果为真，则执行 for 和 endfor 之间的指令，之后执行迭代表达式，然后再次判断 bool 型表达式是否为真，如果为真，则执行 for 和 endfor 之间的指令。直到 bool 型表达式的值为假时跳到 endfor 之后继续执行。

用法举例

```
int b = 0
for(int i = 0;i<5;i++)
b++
endfor
print b //for 循环体会被执行 5 次，结果输出“5”
```


注意

- while 循环和 for 循环并不完全等同于循环运行模式。使用 while 循环或 for 循环时，所有的系统变量以及允许缺省的参数并不会被系统重置，而使用循环模式时，当一次循环执行完毕后，程序会执行复位动作。这意味着那些在程序复位时会被重置的系统变量，以及可以缺省的参数将被恢复为默认值。

  例如：

  ```
  ptp P0
  lin P1,s:{pdis 10}
  ```

- 如果设置了循环运行模式，当第 1 次循环执行完毕后，程序会执行复位的动作。当第 2 次执行 PTP 语句时，slip 参数已经被复位，所以这条 PTP 运动将不会有平滑行为。如果希望在循环运行的过程中，PTP 语句目标点要平滑，请按如下方式明确指定 PTP 的 slip 参数而不要缺省它。

  例如：

  ```
  ptp P0,s:{pdis 10}
  lin P1,s:{pdis 10}
  ```

4.7 break(跳出循环)

格式

```
break
```

描述

跳出循环。在 while，for，loop，repeat 循环体内，如果执行到该语句，程序将退出最近一层循环继续向下执行。

用法举例

```
int counter = 0
while(1)
if(counter == 5)
break //跳出 while 循环。
else
counter++
    endif
endwhile
print count //输出“5”
```

4.8 continue(继续下一个循环)

格式

```
continue
```

描述

结束当前循环，继续下一次循环。在 while，for，loop，repeat 循环体内，如果执行到该语句，程序将结束当前循环，继续执行最近一层循环的下一次循环。

用法举例

```
int count = 0
while(1)
count++
if(count == 1)
continue //执行到这里则直接跳回执行 count++
else
break //执行到这里则退出 while 循环，继续执行 endwhile 下面的指令
endif
endwhile
```

4.9 switch(条件分支)

格式

```
switch(表达式)
case 常量表达式：
……
case 常量表达式：
……
……
default：
……
endswitch
```

描述

条件分支指令是 if 语句的另一种写法。系统将首先计算 switch 后的表达式的值，然后从上到下依次与每一个 case 后面的常量表达式做比较，直到比较相等，则执行这个 case 与下一个 case 之间的指令。执行完后跳到 endswitch 后继续执行。如果没有与任何一个 case 匹配，则执行 default 后面的指令。

用法举例

```
func void TestSwitch()
int j = 3
int i = 0
switch(j)
case 0:
i = 0
case 1:
i = 1
```

```
case 2:
i = 2
case 3:
i = 3 //程序会执行到这里
default:
i = 4
endswitch
endfunc
```

4.10 goto(跳转)

格式

```
label:
goto label
```

描述

可以在一个函数内的任意一行定义一个 label 标签，标签名的命令规则同变量。执行到 goto label 一行时，程序将跳转到 label 标签所在行的下一行继续执行。

用法举例

```
func void TestGoto()
int i = 0
next:
i++
if(i<5)
goto next
else
goto end
endif
end:
print i //输出“5”
endfunc
```

4.11 return(函数返回)

格式

```
return
或
return  表达式
```

描述

函数返回。

程序遇到 return 指令时，如果程序当前处于被调用的函数中，则程序将返回到上一级函数中。如果程序当前处于主函数中，则程序直接结束。

在有返回值的函数中，return 后面需要接一个函数返回值类型的表达式，系统会计算该表达式的值并将结果返回给调用该函数的表达式中。

用法举例

```
func int add (int x,int y)
return x+y //return 语句
endfunc
func void main()
print add(2,3) //输出“5”
endfunc
```

5 系统预定义函数

系统预先定义了很多功能函数，这些函数可以在程序中直接调用。

5.1 数学函数

5.1.1 abs(求绝对值)

函数原型

double abs(double x)

或者 int abs(int x)

描述

计算参数 x 的绝对值。

参数

表 5-1 abs 函数的参数

名称	说明
x	类型：double or int
	输入值

返回值

类型：double or int

返回参数 x 的绝对值。

用法举例

```
double x = -11.11
int y = -10
double resultx = abs(x)
int resulty = abs(y)
print resultx //输出“11.11”
print resulty //输出“10”
```

5.1.2 sin(正弦函数)

函数原型

double sin(double x)

描述

计算参数 x 对应的正弦值。

参数

表 5-2 sin 函数的参数

名称	说明
x	类型：double
	角度值，单位弧度

返回值

类型：double

返回参数 x 对应的正弦值，范围[-1,1]。

用法举例

```
double x = PI/6
double y = sin(x)
print y //输出“0.5”
```

5.1.3 cos(余弦函数)

函数原型

```
double cos(double x)
```

描述

计算参数 x 对应的余弦值。

参数

表 5-3 cos 函数的参数

名称	说明
x	类型：double
	角度值，单位弧度

返回值

类型：double

返回参数 x 对应的余弦值，范围[-1,1]。

用法举例

```
double x = PI/3
double y = cos(x)
print y //输出“0.5”
```

5.1.4 tan(正切函数)

函数原型

double tan(double x)

描述

计算参数 x 对应的正切值。

参数

表 5-4 tan 函数的参数

名称	说明
x	类型：double
	角度值，单位弧度

返回值

类型：double

返回参数 x 对应的正切值。

用法举例

```
double x = PI/4
double y = tan(x)
print y //输出“1”
```

5.1.5 asin(反正弦函数)

函数原型

double asin(double x)

描述

计算参数 x 对应的反正弦值。

参数

表 5-5 asin 函数的参数

名称	说明
x	类型：double
	参数范围 [-1,1]

返回值

类型：double

返回参数 x 对应的反正弦值，单位弧度，范围[-PI/2，PI/2]。

用法举例

```
double x = 0.5
double y = asin(x)
print y //输出“0.523599”
```

5.1.6 acos(反余弦函数)

函数原型

```
double acos(double x)
```

描述

计算参数 x 对应的反余弦值。

参数

表 5-6 acos 函数的参数

名称	说明
x	类型：double
	参数范围[-1,1]

返回值

类型：double

返回参数 x 对应的反余弦值，单位弧度，范围[0，PI]。

用法举例

```
double x = 0.5
double y = acos(x)
print y //输出“1.0472”
```

5.1.7 atan(反正切函数)

函数原型

```
double atan(double x)
```

描述

计算参数 x 对应的反正切值。

参数

表 5-7 atan 函数的参数

名称	说明
x	类型：double

名称	说明
	输入值

返回值

类型：double

返回参数 x 对应的反正切值，单位弧度，范围(-PI/2，+PI/2)。

用法举例

```
double x = 1
double y = atan(x)
print y //输出“0.785398”
```

5.1.8 atan2(反正切函数)

函数原型

```
double atan2(double y,double x)
```

描述

计算参数 y/x 对应的反正切值。

参数

表 5-8 atan2 函数的参数

名称	说明
y	类型：double
	分子数值
x	类型：double
	分母数值

返回值

类型：double

返回参数 y/x 对应的反正切值，单位弧度，范围[-PI,+PI]。

用法举例

```
double x = 1
double y = 1
double z = atan2(y,x)
print z //输出“0.785398”
```

5.1.9 sinh(双曲正弦函数)

函数原型

double sinh(double x)

描述

计算参数 x 对应的双曲线正弦值，其中 sinh(x)=(exp(x)-exp(-x))/2。

参数

表 5-9 sinh 函数的参数

名称	说明
x	类型：double
	输入值，单位是弧度

返回值

类型：double

返回参数 x 对应的双曲线正弦值。

用法举例

```
double x = 1
double y = sinh(x)
print y //输出“1.1752”
```

5.1.10 cosh(双曲余弦函数)

函数原型

double cosh(double x)

描述

计算参数 x 对应的双曲余弦值，其中 cosh(x)=(exp(x)+exp(-x))/2。

参数

表 5-10 cosh 函数的参数

名称	说明
x	类型：double
	输入值，单位是弧度

返回值

类型：double

返回参数 x 对应的双曲余弦值。

用法举例

```
double x = 1
double y = cosh(x)
print y //输出“1.54308”
```

5.1.11 tanh(双曲正切函数)

函数原型

```
double tanh(double x)
```

描述

计算参数 x 对应的双曲线正切值，其中 tanh(x)=sinh(x)/cosh(x)。

参数

表 5-11 tanh 函数的参数

名称	说明
x	类型：double
	输入值，单位是弧度

返回值

类型：double

返回参数 x 对应的双曲线正切值，范围(-1,+1)。

用法举例

```
double x = 1
double y = tanh(x)
print y //输出“0.76159”
```

5.1.12 exp(指数函数)

函数原型

```
double exp(double x)
```

描述

计算参数 x 的以 e 为底的 x 次方值。

参数

表 5-12 exp 函数的参数

名称	说明
x	类型：double

名称	说明
	指数值。

返回值

类型：double

返回参数 x 对应的 e 为底的 x 次方值，范围(0,+∞)。

用法举例

```
double x = 1
double y = exp(x)
print y //输出“2.71828”
```

5.1.13 pow(指数函数)

函数原型

```
double pow(double x,double y)
```

描述

计算以 x 为底的 y 次方值。

参数

表 5-13 pow 函数的参数

名称	说明
x	类型：double
	底数部分
y	类型：double
	指数部分

返回值

类型：double

返回以 x 为底的 y 次方值。

用法举例

```
double x = 2
double y = 3
double z = pow(x,y)
print z //输出“8”
```

5.1.14 pow10(指数函数)

函数原型

double pow10(double x)

描述

计算参数 x 的以 10 为底的 x 次方值。

参数

表 5-14 pow10 函数的参数

名称	说明
x	类型：double
	指数值

返回值

类型：double

返回参数 x 的以 10 为底的 x 次方值。

用法举例

```
double x = -1
double y = pow10(x)
print y //输出“0.1”
```

5.1.15 log(对数函数)

函数原型

double log(double x)

描述

计算以 e 为底的 x 对数值。

参数

表 5-15 log 函数的参数

名称	说明
x	类型：double
	参数范围 x>0

返回值

类型：double

返回以 e 为底的 x 对数值，即 ln(x)的值。

用法举例

```
double x = 2
double y = log(x)
print y //输出“0.69315”
```

5.1.16 log10(对数函数)

函数原型

```
double log10(double x)
```

描述

计算以 10 为底的 x 对数值。

参数

表 5-16 log10 函数的参数

名称	说明
x	类型：double
	参数范围 x>0

返回值

类型：double

返回以 10 为底的 x 对数值，即 $\log_{10}(x)$的值。

用法举例

```
double x = 100
double y = log10(x)
print y //输出“2”
```

5.1.17 sqrt(开方函数)

函数原型

```
double sqrt(double x)
```

描述

计算参数 x 的平方根值。

参数

表 5-17 sqrt 函数的参数

名称	说明
x	类型：double

名称	说明
	参数范围 x>=0

返回值

类型：double

返回参数 x 的平方根值，即$\sqrt{x}$的值。

用法举例

```
double x = 2
double y = sqrt(x)
print y //输出“1.41421”
```

5.1.18 floor(向下取整)

函数原型

```
double floor(double x)
```

描述

计算不大于参数 x 的最大整数值。

参数

表 5-18 floor 函数的参数

名称	说明
x	类型：double
	输入值

返回值

类型：double

返回不大于参数 x 的最大整数值。

用法举例

```
double x = 2.36
double y = floor(x)
print y //输出“2”
```

5.1.19 ceil(Rounded up)

函数原型

```
double ceil(double x)
```

描述

计算不小于参数 x 的最小整数值。

参数

表 5-19 ceil 函数的参数

名称	说明
x	类型：double
	输入值

返回值

类型：double

返回不小于参数 x 的最小整数值。

用法举例

```
double x = 2.36
double y = ceil(x)
print y //输出“3”
```

5.1.20 frexp(分解浮点数)

函数原型

double frexp(double val,int &exp)

描述

将参数 val 分解成数字部分 x 和以 2 为底的指数部分 n，即 val=x*2^n，其中 n 存放在 exp 指向的变量中。

参数

表 5-20 frexp 函数的参数

名称	说明
val	类型：double
	输入值
exp	类型：int
	输出指数值

返回值

类型：double

返回参数 val 分解成数字部分 x 的值，范围[0.5,1)。

用法举例

```
double val = 64
int exp
double y = frexp(val,exp)
print y //输出“0.5”
print exp //输出“7”
```

5.1.21 ldexp(装载浮点数)

函数原型

```
double ldexp(double val,int exp)
```

描述

计算参数 val 与 2 的 exp 次方值的乘积，即计算 val*2exp。

参数

表 5-21 ldexp 函数的参数

名称	说明
val	类型：double
	输入值
exp	类型：int
	输入指数值

返回值

类型：double

返回参数 val 与 2 的 exp 次方值的乘积。

用法举例

```
double val = 3
int exp = 3
double y = ldexp(val,exp)
print y //输出“24”
```

5.1.22 fmod(求模)

函数原型

```
double fmod(double x,double y)
```

描述

计算参数 x 对 y 的模，即 x/y 的余数。

参数

表 5-22 fmod 函数的参数

名称	说明
x	类型：double
	分子参数
y	类型：double
	分母参数，y 不能为 0

返回值

类型：double

返回参数 x/y 的余数。

用法举例

```
double x = 12.58
double y = 2.6
double z = fmod(x,y)
print z //输出“2.18”
```

5.1.23 modf(分解浮点数)

函数原型

```
double modf(double x,int &y)
```

描述

把浮点数 x 分解成整数部分和小数部分，并将整数部分存到 y 变量中。

参数

表 5-23 modf 函数的参数

名称	说明
x	类型：double
	被分解的浮点数
y	类型：int
	返回的浮点数的整数部分

返回值

类型：double

返回双精度数 x 的小数部分，范围(0,1)。

用法举例

```
double x = 123.456
double y
double z = modf(x,y)
print y //输出“123”
print z //输出“0.456”
```

5.1.24 hypot(求直角三角形斜边长)

函数原型

```
double hypot(double x,double y)
```

描述

计算直角三角形的两个直角边分别是 x，y 时，其斜边的长度。

参数

表 5-24 hypot 函数的参数

名称	说明
x	类型：double
	直角三角形的一条直角边长度
y	类型：double
	直角三角形的另一条直角边长度

返回值

类型：double

返回 x 平方与 y 平方的和的平方根，即$\sqrt{x^2+y^2}$。

用法举例

```
double x = 6
double y = 8
double z = hypot(x,y)
print z //输出“10”
```

5.1.25 rand(产生随机数)

函数原型

```
int rand(void)
```

描述

随机产生一个整数。

参数

无

返回值

类型：int

返回一个随机整数，范围[0,+2147483647]。

用法举例

```
int x = rand()
print x //随机输出一个整数，比如输出“15”
```

5.1.26 norm(求向量长度)

函数原型

double norm(pos p)

描述

计算当前位置点距离坐标系原点的长度。

参数

表 5-25 norm 函数的参数

名称	说明
p	类型：pos
	向量坐标值

返回值

类型：double

返回当前位置点 p 距离坐标系原点的长度。

用法举例

```
pos p = {x 300,y 400,z 500}
double len = norm(p)
print len //输出“707.107”
```

5.1.27 trunc(截断浮点数)

函数原型

double trunc(double num,int n,bool round)

描述

将某一数值按照是否四舍五入的要求截断成小数点后指定位数的数值。

参数

表 5-26 trunc 函数的参数

名称	说明
num	类型：double
	被截断的浮点数
y	类型：double
	保留小数的位数。n>=0
round	类型：bool
	true 表示四舍五入，false 表示直接截断

返回值

类型：double

返回截断后的结果。

用法举例

```
double num = 3.1415926
int n = 4
double result1 = trunc(num,n,true)
print result1 //输出“3.1416”
double result2 = trunc(num,n,false)
print result2 //输出“3.1415”
```

5.2 位操作函数

5.2.1 bitclear(位清 0)

函数原型

```
void bitclear(byte &data,int pos)
```

或者 void bitclear(int &data,int pos)

描述

将一个整型或字节型数 data 的第 pos 位清为零。

参数

表 5-27 bitclear 函数的参数

名称	说明
data	类型：byte 或 int
	输入值
pos	类型：int

名称	说明
	参数范围[0,7]（当 data 类型是 byte 时）或者[0,31]（当 data 类型是 int 时）

返回值

无

用法举例

```
byte data1= 00001111b
bitclear(data1,2)
print data1 //输出“0bh”
int data2= 15
bitclear(data2,2)
print data2 //输出“11”
```

5.2.2 bitset(位置 1)

函数原型

```
void bitset(byte &data,int pos)
```

或者 void bitset(int &data,int pos)

描述

将一个整型或字节型数 data 的第 pos 位设置成 1。

参数

表 5-28 bitset 函数的参数

名称	说明
data	类型：byte 或 int
	待设置位的数。
pos	类型：int
	参数范围[0,7]（当 data 类型是 byte 时）或者[0,31]（当 data 类型是 int 时）

返回值

无

用法举例

```
byte data1= 00001000
bitset(data1,2)
print data1 //输出“0ch”
int data2 = 8
bitset(data2,2)
print data2 //输出“12”
```

5.2.3 bitcheck(位检查)

函数原型

bool bitcheck(byte &data,int pos)

或者 bool bitcheck(int &data,int pos)

描述

检查一个整型或字节型数 data 的第 pos 位是否为 1，如果是则函数返回 true，如果不是则函数返回 false。

参数

表 5-29 bitcheck 函数的参数

名称	说明
data	类型：byte 或 int
	待检查位的数。
pos	类型：int
	参数范围[0,7]（当 data 类型是 byte 时）或者[0,31]（当 data 类型是 int 时）

返回值

类型：bool

返回参数 data 的第 pos 位是否等于 1，等于 1 函数返回 true，不等于 1 函数返回 false

用法举例

```
byte data1= 00001000b
bool result = bitcheck(data1,2)
print result //输出“false”
int data2= 8
print bitcheck(data2,3) //输出“true”
```

5.2.4 bitlcs(循环左移多位)

函数原型

int bitlcs(int data,int n)

或者 byte bitlcs(byte data,int n)

描述

将一个整型或字节型数 data 循环左移 n 位。

参数

表 5-30 bitlcs 函数的参数

名称	说明
data	类型：byte 或 int
	待移位的数。
n	类型：int
	n >= 0。参数 n 可以缺省，缺省默认值为 1

返回值

类型：int 或 byte

返回参数 data 循环左移位后的结果。

用法举例

```
byte data= 11000000b
print bitlcs(data,2) //输出“03h”
print bitlcs(data) //输出“81h”
```

5.2.5 bitrcs(循环右移多位)

函数原型

int bitrcs(int data,int n)

或者 byte bitrcs(byte data,int n)

描述

将一个整型或字节型数 data 循环右移 n 位。

参数

表 5-31 bitrcs 函数的参数

名称	说明
data	类型：byte 或 int
	待移位的数。
n	类型：int
	n >= 0。参数 n 可以缺省，缺省默认值为 1

返回值

类型：int 或 byte

返回参数 data 循环右移位后的结果。

用法举例

byte data= 00000011b
print bitrcs(data,2) //输出“c0h”
print bitrcs(data) //输出“81h”

5.3 时钟函数

5.3.1 clock(时钟类型)

描述

clock 时钟类型用于程序中计时。该数据类型配合 clkreset，clkstart，clkstop 和 clkread 函数使用。

用法举例

clock c
clkreset(c)
clkstart(c)
waittime 3
clkstop(c)
print clkread(c) //输出“3.00098”,表示从 clkstart 到 clkstop 共耗时约 3 秒。

5.3.2 clkstart(启动时钟计时)

函数原型

void clkstart(clock &c)

描述

启动时钟计时。一个时钟被启动后，可以对他进行 clkread，clkstop 和 clkreset 动作。

参数

表 5-32 clkstart 函数的参数

名称	说明
c	类型：clock
	参见 *clock*(时钟类型) 数据类型

返回值

无

用法举例

参见 clkread 函数。

5.3.3 clkstop(停止时钟计时)

函数原型

void clkstop(clock &c)

描述

停止时钟计时。一个时钟被停止后，可以对他进行 clkread，clkstart 和 clkreset 动作。

参数

表 5-33 clkstop 函数的参数

名称	说明
c	类型：clock
	参见 *clock*(时钟类型) 数据类型

返回值

无

用法举例

参见 clkread 函数。

5.3.4 clkreset(时钟清 0)

函数原型

void clkreset(clock &c)

描述

将时钟变量重新设置为 0。

参数

表 5-34 clkreset 函数的参数

名称	说明
c	类型：clock
	参见 *clock*(时钟类型) 数据类型

返回值

无

用法举例

参见 clkread 函数。

5.3.5 clkread(读取时钟)

函数原型

double clkread(clock &c)

描述

读取时钟变量 c 的数值。

参数

表 5-35 clkread 函数的参数

名称	说明
c	类型：clock
	参见 *clock*(时钟类型) 数据类型

返回值

类型：double

返回时钟变量 c 的数值，单位是秒

用法举例

```
clock c
clkstart(c)
byte data= 00001100b
int n = 2
byte result = bitrcs1(data,n)
clkstop(c)
print clkread(c) //输出从上一个 clkstart 函数到 clkstop 函数之间运行花费的时间，例如运行时间为 0.001s,则输出“0.001”
clkreset(c)
print clkread(c) //输出“0”
```

5.4 字符串相关函数

5.4.1 strlen(获取字符串长度)

函数原型

int strlen(string s)

描述

计算字符串 s 的长度。

参数

表 5-36 strlen 函数的参数

名称	说明
s	类型：string
	字符数量有待统计的字符串。

返回值

类型：int

返回字符串 s 的长度值。

用法举例

```
string s = "hello world!"
int len = strlen(s)
print len //输出“12”
```

5.4.2 substr(截取字符串)

函数原型

```
string substr(string s,int startpos,int len)
```

描述

从字符串的指定位置处开始截取特定长度的字符子串。

参数

表 5-37 substr 函数的参数

名称	说明
s	类型：string
	待截取的字符串。
startpos	类型：int
	截取字符串的起始位置。startpos 从 0 开始，如果 startpos 为负数或者大于等于字符串总长度，将会产生一个运行时错误
len	类型：int
	截取字符串的长度。如果没有指定该参数，则子字符串将延续到字符串的结尾。如果 len 为负数，将会产生一个运行时错误

返回值

类型：string

返回从字符串 s 的指定位置 startpos 处开始截取 len 长度的字符子串。

用法举例

```
s1 = "hello world!"
int statpos = 0
int len = 5
string s2 = substr(s1,startpos,len)
print s2 //输出“hello”
```

5.4.3 toascii(获取字符对应的 ASCII 码)

函数原型

```
int toascii(string s)
```

描述

该函数返回字符 s 对应的 ASCII 码。

参数

表 5-38 toascii 函数的参数

名称	说明
s	类型：string
	输入字符串 s，该字符串只能包含 1 个字符

返回值

类型：int

字符 s 对应的 ASCII 码值。

用法举例

```
print toascii("a") //输出“97”
```

5.5 IO 相关函数及指令

5.5.1 setdo(异步输出单路 DO)

函数原型

```
void setdo(int chan,bool value)
```

描述

设置一路 DO 信号为 true 或者 false。

参数

表 5-39 setdo 函数的参数

名称	说明
chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）
value	类型：bool
	指定输出值

返回值

无

用法举例

setdo(3,true) //第 3 通道置为 true

5.5.2 setdo(异步输出多路 DO)

函数原型

void setdo(int from_chan,int to_chan,int value)

描述

设置连续多路 DO 信号。

参数

表 5-40 setdo 函数的参数

名称	说明
from_chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）
to_chan	类型：int
	结束通道号。参数范围[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,40*PLC_MF 从站数]（针对 RTEX 总线版控制系统） from_chan <= to_chan 即使 to_chan 超过实际可控地址范围，只要满足上述条件，仍旧可以正常输出可控地址范围内的 DO，不会产生报警
value	类型：int
	因为 int 型数据长度为 32 位，所以这里最多只能同时设置连续的 32 路 DO

返回值

无

用法举例

setdo(5,8,1100b)
//将第 5 和第 6 路 DO 信号置为 false，将第 7 和第 8 路信号置为 true

5.5.3 setdoinmv(不停前瞻异步输出单路 DO)

函数原型

setdoinmv channel:int chan,value:bool value

描述

设置一路 DO 信号为 true 或者 false，在下一行运动指令开始前输出。

参数

表 5-41 setdoinmv 指令的参数

名称	说明
chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1, 40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）
value	类型：bool
	指定输出值

返回值

无

用法举例

movej j:j1,vp:5%,sp:5%
setdoinmv channel:3,value:true //将第 3 通道置为 true
movej j:j2,vp:5%,sp:5%

5.5.4 syncdo(同步输出单路 DO)

函数原型

void syncdo(int chan,bool value)

描述

同步输出一路 DO 信号为 true 或者 false。同步输出与异步输出的区别为同步输出保证 IO 端口信号已经输出后再继续向下执行程序，而异步输出则将数据发送出去后便继续向下执行程序。

参数

表 5-42 syncdo 函数的参数

名称	说明
chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）
value	类型：bool
	指定输出值

返回值

无

用法举例

syncdo(3,true) //将第 3 通道置为 true

5.5.5 syncdo(同步输出多路 DO)

函数原型

void syncdo(int from_chan,int to_chan,int value)

描述

同步输出连续多路 DO 信号。同步输出与异步输出的区别为同步输出保证 IO 端口信号已经输出后再继续向下执行程序，而异步输出则将数据发送出去后便继续向下执行程序。

参数

表 5-43 syncdo 函数的参数

名称	说明
from_chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）
to_chan	类型：int
	结束通道号。参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,40*PLC_MF 从站数]（针对 RTEX 总线版控制系统） from_chan <= to_chan
value	类型：int
	因为 int 型数据长度为 32 位，所以这里最多只能同时设置连续的 32 路 DO

返回值

无

用法举例

syncdo(5,8,1100b)

//将第 5 和第 6 路 DO 信号置为 false，将第 7 和第 8 路信号置为 true

5.5.6 pulsedo(输出单路脉冲信号)

函数原型

void pulsedo(int chan,bool value,double width)

描述

该函数用于输出单路指定脉宽的脉冲信号。相比使用 setdo 函数配合 waittime 指令实现的脉冲信号，使用 pulsedo 函数更加高效，因为它不会等待脉冲信号执行完，就会继续向下执行程序，也就是说脉冲输出的执行和程序的执行是并行的。

参数

表 5-44 pulsedo 函数的参数

名称	说明
chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）
value	类型：bool
	true 表示输出高脉冲，false 表示输出低脉冲。
width	类型：double
	指定脉冲宽度，单位 s。该值必须大于等于 0，最小的输出脉宽约 25ms 左右

返回值

无

用法举例

pulsedo(5,true,0.2)

//第 5 路 DO 输出脉宽为 200ms 的高脉冲信号

注意事项

- 当一个脉冲输出函数还未执行完时，在该路 DO 又执行了另一个 pulsedo 函数，则尚未执行完的那个 pulsedo 信号会终止。

  例如，对于如下程序：

  pulsedo(5,true,3)

  waittime 2

  pulsedo(5,false,1)

  将在第 5 路 DO 上输出如下图 5-1 中的波形：

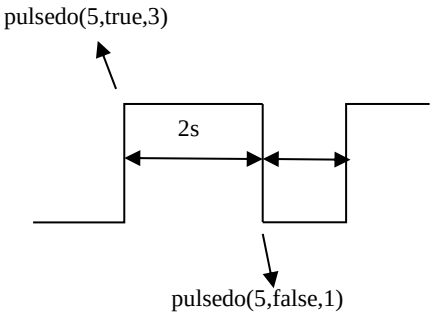


图 5-1 程序示例对应波形示意图

- 某个通道中如果存在尚未执行完毕的脉冲信号时，则该通道将一直处于运行或暂停状态，直到所有的脉冲输出信号全部执完毕才会变成停止状态。
- 程序暂停时脉冲输出信号不会停止执行。
- 程序复位时，如果存在未执行完毕的脉冲输出信号，则这些信号会立即停止执行。

5.5.7 getdo(获取单路 DO)

函数原型

bool getdo(int chan)

描述

获取单路 DO 信号值。

参数

表 5-45 getdo 函数的参数

名称	说明
chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）

返回值

类型：bool

获取到的第 chan 路 DO 信号值。

用法举例

```
int chan = 3
setdo(chan,true)
bool value = getdo(chan)
print value //输出“true”
```

5.5.8 getdo(获取多路 DO)

函数原型

int getdo(int from_chan,int to_chan)

描述

获取连续多路 DO 信号值。

参数

表 5-46 getdo 函数的参数

名称	说明
from_chan	类型：int
	起始通道号。参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1, 40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）
to_chan	类型：int
	结束通道号。参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1, 40*PLC_MF 从站数]（针对 RTEX 总线版控制系统） from_chan <= to_chan

返回值

类型：int

因为 int 型数据长度为 32 位，所以这里最多只能同时获取连续 32 路 DO 信号的值。

用法举例

```
int from = 5
int to = 8
setdo(from,to,1001b)
print getdo(from,to) //输出“9”
```

5.5.9 getdi(获取单路 DI)

函数原型

bool getdi(int chan)

描述

获取单路 DI 信号值。

参数

表 5-47 getdi 函数的参数

名称	说明
chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,

名称	说明
	40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）

返回值

类型：bool

获取到的第 chan 路 DI 信号值

用法举例

//假设第 4 通道 DI 输入了高电平
print getdi(4) //输出“true”

5.5.10 getdi(获取多路 DI)

函数原型

int getdi(int from_chan,int to_chan)

描述

获取连续多路 DI 信号值。

参数

表 5-48 getdi 函数的参数

名称	说明
from_chan	类型：int
	起始通道号。参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1, 40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）
to_chan	类型：int
	结束通道号。参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1, 40*PLC_MF 从站数]（针对 RTEX 总线版控制系统） from_chan <= to_chan

返回值

类型：int

因为 int 型数据长度为 32 位，所以这里最多只能同时获取连续 32 路 DI 信号的值。

用法举例

//假设第 6 到第 9 通道的 DI 输入了高电平
int value = getdi(6,9)
print value //输出“15”

5.5.11 getai(获取模拟量输入信号)

函数原型

double getai(int chan)

描述

获取一路模拟量信号输入值。其中，AI 信号类型包括：电流型和电压型，通过 PLC 从站类型配置 AI 信号类型。当输入信号为电流型时，返回值单位为毫安；当输入信号为电压型时，返回值单位为伏。

参数

表 5-49 getai 函数的参数

名称	说明
chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）

用法举例

print getai(2) //输出“第 2 通道的模拟量输入值”（单位与配置的 AI 信号类型有关）

5.5.12 getnostopdi（不停前瞻异步获取单路 DI）

函数原型

bool getnostopdi(int chan)

描述

获取单路 DI 信号值，在下一行运动指令开始前获取。


提示

与 **getdi** 相比，该指令不停前瞻，所以不会导致机器人停止卡顿等问题。但是如果前瞻条数太多，可能会使机器人在未执行完前面的运动语句时就触发该指令，导致产生不期望的动作。

参数

名称	说明
chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1, 40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）

返回值

类型：bool

获取到的第 chan 路 DI 信号值

用法举例

//假设第 4 通道 DI 输入了高电平

print getnostopdi(4) //输出“true”

5.5.13 setao(异步输出模拟量信号)

函数原型

void setao(int chan,double value)

描述

设置一路模拟量信号输出。其中，AO 信号类型包括：电流型和电压型，通过 PLC 从站类型配置 AO 信号类型。

参数

表 5-50 setao 函数的参数

名称	说明
chan	类型：int
	模拟量输出端口号，参数范围[1,2*AO 从站数]（针对 MIII 总线版控制系统）或[1,2*PLC_MF 从站数]（针对 RTEX 总线版控制系统）
value	类型：double
	当输出信号为电流型时，参数范围[4,20]，单位毫安；当输出信号为电压型时，参数范围[-10,10]，单位伏

用法举例

setao(2,5.2) //若第 2 路 AO 配置为电流型时，则将第 2 通道置为 5.2mA
setao(2,5.2) //若第 2 路 AO 配置为电压型时，则将第 2 通道置为 5.2V

5.5.14 syncao(同步输出模拟量信号)

函数原型

void syncao(int chan,double value)

描述

同步输出一路模拟量信号。同步输出与异步输出的区别为同步输出保证 IO 端口信号已经输出后再继续向下执行程序，而异步输出则将数据发送出去后便继续向下执行程序。

参数

表 5-51 syncaov 函数的参数

名称	说明
chan	类型：int
	模拟量输出端口号，参数范围[1,2*AO 从站数]（针对 MIII 总线版控制系统）或[1,2*PLC_MF 从站数]（针对 RTEX 总线版控制系统）
value	类型：double

名称	说明
	当输出信号为电流型时，参数范围[4,20]，单位毫安；当输出信号为电压型时，参数范围[-10,10]，单位伏

返回值

无

用法举例

```
syncao(2,5.2) //若第 2 路 AO 配置为电流型时，则将第 2 通道置为 5.2mA
syncao(2,5.2) //若第 2 路 AO 配置为电压型时，则将第 2 通道置为 5.2V
```

5.5.15 getao(获取模拟量输出信号)

函数原型

```
double getao(int chan)
```

描述

获取一路模拟量信号输出值。其中，AO 信号类型包括：电流型和电压型，通过 PLC 从站类型配置 AO 信号类型。当输出信号配置为电流型时，返回值单位为毫安；当输出信号配置为电压型时，返回值单位为伏。

参数

表 5-52 getao 函数的参数

名称	说明
chan	类型：int
	模拟量输出端口号，参数范围[1,2*AO 从站数]（针对 MIII 总线版控制系统）或[1,2*PLC_MF 从站数]（针对 RTEX 总线版控制系统）

返回值

类型：double

返回第 chan 路模拟量端口信号的值。

用法举例

```
int chan = 1
setao(chan,5.2)
double value = getao(chan)
print value //输出“5.2”
```

5.5.16 getintdo(读取单路 PLC_INT 的 DO 信号值)

函数原型

```
bool getintdo(int chan)
```

描述

获取 INT 单路 DO 信号值。

参数

表 5-53 getintdo 函数的参数

名称	说明
chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）

返回值

类型：bool

获取到的 INT 第 chan 路 DO 信号值。

用法举例

```
bool value = getintdo(chan)
print value // DO status of channel chan on output INT
```

5.5.17 getintdi(读取单路 PLC_INT 的 DI 信号值)

函数原型

bool getintdi(int chan)

描述

获取 INT 单路 DI 信号值。

参数

表 5-54 getintdi 函数的参数

名称	说明
chan	类型：int
	参数范围为[1,DO 端口数]。如：[1,32*DO 从站数]（针对 MIII 总线版控制系统）或[1,40*PLC_MF 从站数]（针对 RTEX 总线版控制系统）

返回值

类型：bool

获取到的 INT 第 chan 路 DI 信号值。

用法举例

```
bool value = getintdi(chan)
```

print value //输出 INT 上第 chan 路 DI 状态

5.5.18 setpwm(设置一路 PWM 通道的频率和占空比)

函数原型

bool setpwm(int channel, int freq, int ratio)

描述

设置一路 PWM 通道的频率和占空比。

参数

表 5-55 setpwmi 函数的参数

名称	说明
channel	类型：int
	PWM 通道号，参数范围：[1,3]
freq	类型：int
	PWM 频率，单位：Hz，参数范围：[10,10000]
chan	类型：int
	PWM 占空比，单位：%，参数范围：[0,100]

返回值

类型：bool

用法举例

setpwm(1,20,50) //对第 1 路 PWM 通道进行设置，频率设为 20Hz，占空比设为 0.5

5.6 通信相关函数

5.6.1 socket 通信相关函数

5.6.1.1 setip(设置对外网口 IP)

函数原型

bool setip(string ip,string gate,string mask,string if_name)

bool setip(string ip,string gate,string mask)

描述

如果想要通过对外网口与外部设备通信，首先需要设置网口 IP，网关，子网掩码和网口名称。可以通过此函数来进行设置。

参数

表 5-56 setip 函数的参数

名称	说明
ip	类型：string
	要设置的 IP 字符串
gate	类型：string
	要设置的网关
mask	类型：string
	要设置的子网掩码
if_name	类型：string
	要设置的网口名称。注：对于 SCARA 控制柜，有三个用户网口：eth1、eth2、eth3，对于其他控制柜，只有一个用户网口：eth1(当该参数缺省时，默认为 eth1)

返回值

类型：bool

true 表示设置成功，false 表示设置失败。

设置 IP 时，网关地址也必须是真实存在的 IP，否则返回 false。

用法举例

```
setip("10.20.210.93","10.20.210.255","255.255.255.0","eth1") //设置的用户网口名称为 eth1
setip("10.20.210.93","10.20.210.255","255.255.255.0") //设置的用户网口名称为 eth1
```

5.6.1.2 getip(获取对外网口 IP)

函数原型

```
bool getip(string& ip,string if_name)
bool getip(string& ip)
```

描述

如果想要通过获取对外网口 IP 字符串。可以通过此函数来进行设置。

参数

表 5-57 getip 函数的参数

名称	说明
ip	类型：string
	要设置的 IP 字符串
if_name	类型：string
	指定获取哪个用户网口的 IP。注：对于 SCARA 控制柜，有三个用户网口：eth1、eth2、eth3，对于其他控制柜，只有一个用户网口：eth1(当该参数缺省时，默认为 eth1)

返回值

类型：bool

true 表示获取成功，false 表示获取失败。

用法举例

```
string ip
getip(ip,"eth2") //获取名称为"eth2"的用户网口的 IP
getip(ip) //获取名称为"eth1"的用户网口的 IP
```

5.6.1.3 connect(发起 socket 连接)

函数原型

bool connect(socket s,string ip,int port)

描述

如果希望机器人控制器作为发起连接的一方，也就是 client 端，则需要使用此函数发起 socket 连接。须配合 waituntil 指令实现同步连接，也就是说等待直到连接成功程序才继续运行。

参数

表 5-58 connect 函数的参数

名称	说明
s	类型：socket
	预先定义好的套接字变量
ip	类型：string
	要连接的对方设备 IP 字符串
port	类型：int
	要连接的对方设备的端口

返回值

类型：bool

未连接成功时，返回 false；当返回 true 时表示连接成功。

用法举例

```
socket s
//等待，直到连接成功，这里也可以使用 waituntil 指令的 maxtime
//等参数实现超时机制
waituntil connect(s,"192.168.0.40",2888)
```

5.6.1.4 accept(接受 socket 连接)

函数原型

bool accept(socket s,string ip,int port)

描述

如果希望机器人控制器作为被动接受连接的一方，也就是 server 端，则需要使用此函数接受 socket 连接。须配合 waituntil 指令实现同步连接，也就是说等待直到对方发起连接才继续运行。

参数

表 5-59 accept 函数的参数

名称	说明
s	类型：socket
	预先定义好的套接字变量
ip	类型：string
	本机 IP 字符串
port	类型：int
	本机端口

返回值

类型：bool

未建立连接时，返回 false；当返回 true 时表示建立连接成功。

用法举例

```
socket s
//等待，直到建立连接，这里也可以使用 waituntil 指令的 maxtime
//等参数实现超时机制
waituntil accept(s,"192.168.0.40",2888)
```

5.6.1.5 write(通过 socket 发送数据)

函数原型

bool write(socket s,string data)或
bool write(socket s,byte[] data,int len)

描述

不管是 server 端还是 client 端均通过此函数发送数据。须配合 waituntil 指令实现同步发送，也就是说等待直到数据发送完成才继续运行。

参数

表 5-60 write 函数的参数

名称	说明
s	类型：socket
	预先定义好的套接字变量

名称	说明
data	类型：string/byte[]
	要发送的字符串数据或二进制数据
len	类型：int
	当发送的数据为字节数组时，该参数表示写入的字节长度

返回值

类型：bool

发送未完成时，返回 false；当返回 true 时表示发送完成。

用法举例

```
socket s
waituntil connect(s,"192.168.0.40",2888)
waituntil write(s,"hello")
byte bdata[3] = {1,2,3}
waituntil write(s,bdata,3)
```

socket 断线重连功能

当 write 过程中出现 socket 断开时，ARCS 会产生报警并停止运行，如果此时不希望程序停止，可在断线后尝试重连的方法，可在 ARL 程序中进行设置：$DETECT_SOCK_CLOSE=true。

断线后设置当前 socket 的错误状态为$ERR_SOCK_CLOSED，用户可以通过 geterror(s)来获取 socket s 的错误状态。


提示

- 当需要使用断线重连功能时，使用定时中断（timer）和清空缓冲区（clearbuff）会影响断线状态的检测，建议使用 while 和 waittime 替代定时中断 timer。
- 当机器人为 Server 时，在断线重连子函数中，需要先执行 close，再执行 accept。

实现断线重连的方法如下：

用户可以声明中断，当某个 socket 的错误状态为$ERR_SOCK_CLOSED 时，触发某个中断处理函数，在该中断处理函数中重新进行 socket 连接。

用法举例：

```
socket s
func void handler()
close(s)
waituntil connect(s,"10.20.220.1",8080)
endfunc
func viod main()
init()
$DETECT_SOCK_CLOSE=true
interrupt 1, when:geterror(s)==$ERR_SOCK_CLOSED, do:handler()
setip(.....)
```

```
waituntil connect(s,.....)
waituntil write (s,.....)
endfunc
```

5.6.1.6 read(通过 socket 接收数据)

函数原型

```
bool read(socket s,string data,int len)
bool read(socket s,byte[] data,int len)
```

描述

不管是 server 端还是 client 端均通过此函数接收指定长度的数据。须配合 waituntil 指令实现同步接收，也就是说等待直到数据接收完成才继续运行。

参数

表 5-61 read 函数的参数

名称	说明
s	类型：socket
	预先定义好的套接字变量
data	类型：string/byte[]
	预先定义的字符串变量或字节数组变量
len	类型：int
	接收数据长度，当 data 参数为 string 类型时，该长度指字符串中字符个数；当 data 参数为字节数组类型时，该长度指接收的字节个数

返回值

类型：bool

未接收到指定长度的字符串时，返回 false；当返回 true 时表示接收完成。

用法举例

```
socket s
waituntil connect(s,"192.168.0.40",2888)
string data
waituntil read(s,data,5)
print data
byte bdata[3]
waituntil read(s,bdata,3)
```

socket 断线重连功能

当 read 过程中出现 socket 断开时，ARCS 会产生报警并停止运行，如果此时不希望程序停止，可在断线后尝试重连的方法，可在 ARL 程序中进行设置，$DETECT_SOCK_CLOSE=true。

断线后设置当前 socket 的错误状态为$ERR_SOCK_CLOSE，用户可以通过 geterror(s)来获取 socket s 的错误状态。

实现断线重连的方法如下：

用户可以声明中断，当某个 socket 的错误状态为$ERR_SOCK_CLOSED 时，触发某个中断处理函数，在该中断处理函数中重新进行 socket 连接。

用法举例：

```
socket s
func void handler()
close(s)
waituntil connect(s, "10.20.220.1", 8080)
endfunc
func viod main()
init()
$DETECT_SOCK_CLOSE=true
interrupt 1, when:geterror(s)== $ERR_SOCK_CLOSED, do:handler()
setip(.....)
waituntil connet(s,.......)
waituntil read (s,......)
endfunc
```

5.6.1.7 readuntil(通过 socket 接收数据)

函数原型

bool readuntil(socket s,string data,string cut)

描述

不管是 server 端还是 client 端均通过此函数接收直到读到某个指定的字符或字符串的数据。须配合 waituntil 指令实现同步接收，也就是说等待直到接收到指定的字符或字符串数据才继续运行。

参数

表 5-62 readuntil 函数的参数

名称	说明
s	类型：socket
	预先定义好的套接字变量
data	类型：string
	预先定义的字符串变量
cut	类型：string
	指定截断字符或字符串，如果希望该字符为换行符，则使用\n 作为这里的 cut 参数

返回值

类型：bool

未接收到指定字符或字符串时，返回 false；当返回 true 时表示接收到指定的字符或字符串。

用法举例

```
socket s
waituntil connect(s,"192.168.0.40",2888)
string data
//如果对方发过来的数据是“hello”,则这里 data 将打印出 he
waituntil readuntil(s, data, "e")
print data
```

5.6.1.8 clearbuff(清空 socket 的输入缓冲区)

函数原型

bool clearbuff(socket s)

描述

清空 socket 的输入缓冲区（当输入缓冲区为空时，通过 read 函数接受不到任何数据）。

参数

表 5-63 clearbuff 函数的参数

名称	说明
s	类型：socket
	预先定义好的套接字变量

返回值

类型：bool

返回 true 表示操作成功，false 表示操作失败。

用法举例

```
socket s
waituntil accept(s,"192.168.1.1",8080), maxtime:10,timeoutflag:time_flag
waituntil read(s,data,3), maxtime:10,timeoutflag:time_flag
clearbuff（s）
……
```

5.6.2 串口通信相关函数

5.6.2.1 open(打开串口设备)

函数原型

bool open(iodev& dev,string devname,int baud_rate,int character_size,stopbits stop_bits,parity prt)

描述

打开并配置一个串口设备。

参数

表 5-64 open 函数的参数

名称	说明
dev	类型：iodev
	预先定义好的 IO 设备
devname	类型：string
	柜子类型为 ARCC 系列时： 设备名，“/dev/ttyS0”表示 com1，“/dev/ttyS1”表示 com2 柜子类型为 ARCCD 系列时： 设备名，“/dev/ttyS0”表示 com1
baud_rate	类型：int
	波特率，可以设置为 300，600，1200，2400，4800，9600，19200，38400，57600,115200，单位 bps
character_size	类型：int
	字符大小，可以设置为 5，6，7，8
stop_bits	类型：stopbits
	停止位类型，参见 stopbits 枚举类型，可以设置为 1 位和 2 位
prt	类型：parity
	奇偶校验类型，参见 parity 枚举类型，可以设置为不校验，奇校验或偶校验

返回值

类型：bool

返回 true 表示打开设备成功。

用法举例

```
iodev s
open(s,"/dev/ttyS0",9600,8,one,none)
waituntil write(s, "hello")
```

5.6.2.2 close(关闭串口设备)

函数原型

void close(iodev& dev)

描述

关闭一个之前打开的串口设备。

参数

表 5-65 close 函数的参数

名称	说明
dev	类型：iodev
	之前打开的 IO 设备

返回值

无

用法举例

```
iodev s
open(s,"/dev/ttyS0",9600,8,one,none)
waituntil write(s,"hello")
close(s)
```

5.6.2.3 read/write/readuntil(读写串口设备)

描述

以上 3 个函数的用法与 socket 通信函数中的 read/write/readuntil 用法完全相同，只要将第一个参数换成 iodev 即可。

用法举例

```
iodev s
//打开 com1 口
open(s,"/dev/ttyS0",9600,8,one,none)
waittime 1
//写字符串
write(s,"hello")
write(s,"world")
string data
//按结束符读字符串
waituntil readuntil(s,data,"o")
print "data:",data
//按数量读字符串
waituntil read(s,data,4)
//写二进制数据
byte sdata[4]={1,2,3,4}
write(s,sdata,4)
close(s)
```

5.6.2.4 clearbuff(清空串口设备的输入缓冲区)

函数原型

```
bool clearbuff(iodev& dev)
```

描述

清空串口设备的输入缓冲区。

参数

表 5-66 clearbuff 函数的参数

名称	说明
dev	类型：iodev
	之前打开的 IO 设备

返回值

类型：bool

返回 true 表示操作成功，false 表示操作失败。

用法举例

```
iodev s
open(s,"/dev/ttyS0",9600,8,one,none)
waituntil write(s,"hello")
clearbuff（s）
……
close(s)
```

5.6.3 devicenet 总线通信相关函数

5.6.3.1 dnwrite(向 devicenet 总线发送数据)

函数原型

int dnwrite(int offset,int len, byte[] data)

描述

向 devicenet 总线 buffer 写入数据。

参数

表 5-67 dnwrite 函数的参数

名称	说明
offset	类型：int
	起始字节的偏移地址
len	类型：int
	写入数据的长度，单位：字节数
data	类型：byte[]
	要发送的二进制数据数组

返回值

类型：int

返回 0 时，表示成功写入。非 0 表示写入失败，值表示错误码。

用法举例

byte bdata[3] = {1,2,3}
dnwrite(0,3,bdata) //从字节 0 开始，向 devicenet 设备写 3 个字节数据

5.6.3.2 dnread(从 devicenet 总线读入数据)

函数原型

int dnread(int offset,int len, byte[] data)

描述

从 devicenet 总线 buffer 读入数据。

参数

表 5-68 dnread 函数的参数

名称	说明
offset	类型：int
	起始字节的偏移地址
len	类型：int
	读入数据的长度，单位：字节数
data	类型：byte[]
	要读入的二进制数据数组

返回值

类型：int

返回 0 时，表示成功读出。非 0 表示读失败，值表示错误码。

用法举例

byte bdata[3]
dnread(0,3,bdata) //从字节 0 开始，从 devicenet 设备读出 3 个字节数据

5.6.4 ModBusTCP 相关函数

5.6.4.1 readregisters(从 modbus 寄存器中读取数据)

函数原型

bool readregisters(string data, int start, int len)
bool readregisters(byte[] data, int start, int len)

描述

在以 start 为起始地址、长度为 len（单位：寄存器）的 modbus 寄存器中，读取并转换为字节型或字符串型数据，储存在预先定义的变量中。

参数

参数名称	参数类型	说明
data	string/byte[]	预先定义的字节/字符串类型变量
start	int	读取数据的起始寄存器地址
len	int	读寄存器的个数

返回值

类型：bool

读取成功时，返回 true；读取失败时，返回 false。

用法举例

```
byte bdata[4]
waituntil readregisters(bdata,100,2) //在以 100 为起始地址、长度为 2（单位：寄存器）的 modbus 寄存器中，读取并转换为字节型数据，储存在 bdata 变量中。
print bdata
string s
waituntil readregisters(s,200,2) //在以 200 为起始地址、长度为 2（单位：寄存器）的 modbus 寄存器中，读取并转换为字符串型数据，储存在 s 变量中。
print s
```

5.6.4.2 writeregisters(向 modbus 寄存器中写入数据)

函数原型

```
bool writeregisters(string data, int start)
bool writeregisters(byte[] data, int start, int len)
```

描述

将字符串或字节型数据，写入起始地址为 start、长度为 len（单位：寄存器）的 modbus 寄存器中。

参数

参数名称	参数类型	说明
data	string/byte[]	预先定义的字节/字符串类型变量
start	int	写入数据的起始寄存器地址
len	int	写寄存器的个数

返回值

类型：bool

写入成功时，返回 true；写入失败时，返回 false。

用法举例

byte bdata[4] = {1,2,3,4}

waituntil writeregisters(bdata,100,2) //将字节型数据 bdata，写入起始地址为 100、长度为 2（单位：寄存器）的 modbus 寄存器中。

string s = "abcd"

waituntil writeregisters(s,200,2) //将字符串型数据 s，写入起始地址为 200、长度为 2（单位：寄存器）的 modbus 寄存器中。

5.6.4.3 readcoils(从 modbus 线圈中读取数据)

函数原型

bool readcoils(byte[], int start, int length)

bool readcoils(byte[], int start, int length)

描述

将要发送的二进制数据，从起始地址为 start、长度为 length（单位：bit）的 modbus 线圈中读取。

参数

参数名称	参数类型	说明
byte[]	byte[]	要发送的二进制数据
start	int	读取数据的起始寄存器地址
length	int	读取线圈个数，用十进制数表示

返回值

类型：bool

读取成功时，返回 true；读取失败时，返回 false。

用法举例

byte bdata[2]

waituntil readcoils(bdata,100,16) // bdata 表示要发送的二进制数据，100 表示读线圈的起始地址，这里表示从第 100 个线圈开始读，16 表示读 16 个线圈。

print bdata

5.6.4.4 writecoils(向 modbus 线圈中写入数据)

函数原型

bool writecoils(byte[], int start, int length)

bool writecoils(byte[], int start, int length)

描述

将二进制数据，写入起始地址为 start、长度为 length（单位：bit）的 modbus 线圈中。

参数

参数名称	参数类型	说明
byte[]	byte[]	要发送的二进制数据

参数名称	参数类型	说明
start	int	写入数据的起始寄存器地址
length	int	写入线圈个数，用十进制数表示

返回值

类型：bool

写入成功时，返回 true；写入失败时，返回 false。

用法举例

byte bdata[2] = {1,2}

waituntil writecoils(bdata,100,16) // bdata 表示要发送的二进制数组，100 表示写线圈的起始地址，这里表示从第 100 个线圈开始写，16 表示写 16 个线圈。

5.6.5 Melsec 通讯相关的 ARL 函数

5.6.5.1 open(打开 melsec 从站设备)

函数原型

bool open(melsec_dev dev, string ip, int port, int slave_id)

描述

打开并配置一个 Melsec 协议 PLC 设备

参数

表 5-69 open 函数的参数

名称	说明
dev	类型：melsec_dev
	预先定义好的 melsec 套接字变量
ip	类型：string
	设备 ip,比如取值“192.168.1.101”
port	类型：int
	要连接的对方设备的端口
slave_id	类型：int
	Melsec 从站号，取值范围:0~247 以及 255；其中 0 是 Melsec 广播地址;在 TCP 模式下，255 能被用来恢复默认值。

返回值

类型：bool

true 表示打开成功，false 表示打开失败。

用法举例

melsec_dev mc //定义套接字，与 socket 功能类似

waituntil open(mc,"10.20.220.92",8080,255)

5.6.5.2 close(关闭 melsec 从站设备)

函数原型

void close(melsec_dev dev)

描述

关闭一个已打开的 Melsec 协议 PLC 设备

参数

表 5-70 close 函数的参数

名称	说明
dev	类型：melsec_dev
	预先定义好的 melsec 套接字变量

返回值

类型：void

用法举例

close(mc) //关闭 Melsec 协议 PLC 设备

5.6.5.3 read(通过 melsec 从主站接收数据)

函数原型

bool read(melsec_dev dev, string data, in start, int len)
bool read(melsec_dev dev, byte[] data, in start, int len)

描述

以 start 为起始地址、len 为长度(单位：字节)，将 plc 寄存器内容读取至预先定义的字符串变量或字节数组变量中。

不管是 server 端还是 client 端均通过此函数接收指定长度的数据。须配合 waituntil 指令实现同步接收，也就是说等待直到数据接收完成才继续运行。

参数

表 5-71 read 函数的参数

名称	**说明**
dev	类型：melsec_dev
	预先定义好的 melsec 套接字变量
data	类型：string 或 byte[]
	预先定义的字符串变量或字节数组变量

名称	说明
start	类型：int
	起始偏移地址
len	类型：int
	接收数据长度，当 data 参数为 string 类型时，该长度指字符串中字符个数；当 data 参数为字节数组类型时，该长度指接收的字节个数。

返回值

类型：bool

未接收到指定长度的字符串时，返回 false；当返回 true 时表示接收完成。

用法举例

```
melsec mc
waituntil open(mc,"10.20.220.92",8080,255)
string data
waituntil read(mc,data,6000,10)
byte bdata24]
waituntil read(mc,bdata,6000,24)
```

5.6.5.4 write(通过 melsec 向主站发送数据)

函数原型

```
bool write(melsec_dev dev, string data, in start, int len)
bool write(melsec_dev dev, string data, in start).
bool write(melsec_dev dev, byte[] data, in start, int len)
```

描述

以 start 为起始地址、len 为长度(单位：字节)，把输入发送的字符串变量或字节数组变量写入 plc 寄存器。

不管是 server 端还是 client 端均通过此函数发送数据。须配合 waituntil 指令实现同步发送，也就是说等待直到数据发送完成才继续运行。

参数

表 5-72 write 函数的参数

名称	说明
dev	类型：melsec_dev
	预先定义好的 melsec 套接字变量
data	类型：string 或 byte[]
	预先定义的字符串变量或字节数组变量
start	类型：int

名称	说明
	起始偏移地址
len	类型：int
	发送数据长度，当 data 参数为 string 类型时，该长度指字符串中字符个数(参数缺省时默认为 string 的长度)；当 data 参数为字节数组类型时，该长度指接收的字节个数。

返回值

类型：bool

发送未完成时，返回 false；当返回 true 时表示发送完成。

用法举例

```
melsec mc
waituntil open(mc,"10.20.220.92",8080,255)
string data = "abcd"
waituntil write(mc1,data,6000)
byte bdata[4] = {1,2,3,4}
waituntil write (mc,bdata,6000,4)
```

5.6.6 modbus-rtu 通信相关函数

5.6.6.1 open(打开 modbus 从站设备)

函数原型

bool open(modbus_dev& dev, string devname, int slave_id, int baud_rate, ParityType parity, int databits, StopBitType stop_bits)

描述

打开并配置一个 modbus 从站设备。

参数

表 5-73 参数说明

参数名称	参数类型	说明
dev	modbus_dev	预先定义好的 modbus 设备
devname	string	设备名，对于二代柜来说，设备名为”rtserMB0”
slave_id	int	Modbus 从站号，取值范围 1-247
baud_rate	int	波特率，可以设置为 4800，9600，19200，38400，57600,115200，单位是 bps
parity	ParityType	奇偶校验类型，参见 modbus_parity 枚举类型，可以设置为不校验，奇校验或偶校验
databits	int	数据位，可以设置为 5，6，7，8
stop_bits	StopBitType	停止位类型，参见 modbus_stopbits 枚举类型，可以设置为 1 位或 2 位

返回值

类型：bool

返回 true 表示操作成功，false 表示操作失败。

用法举例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
```

5.6.6.2 close(关闭 modbus 从站设备)

函数原型

```
void close(modbus_dev& dev)
```

描述

关闭一个之前打开的 modbus 从站设备。

参数

表 5-74 参数说明

参数名称	参数类型	说明
dev	modbus_dev	之前打开的 modbus 设备

返回值

无

用法举例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
close(m)
```

5.6.6.3 read(通过 modbus 从主站接收数据)

函数原型

```
bool read(modbus_dev dev, string data, in start, int len)
bool read(modbus_dev dev, byte[] data, in start, int len)
```

描述

以 start 为起始地址、len 为长度(单位：字节)，将 modbus 从站设备寄存器内容读取至预先定义的字符串变量或字节数组变量中。

参数

表 5-75 参数说明

参数名称	参数类型	说明
dev	modbus_dev	已经打开的 modbus 设备
data	string/byte[]	预先定义的字符串变量或字节数组变量
start	int	所要读取数据的起始地址
len	int	所要读取数据的长度

返回值

类型：bool

未接收到指定长度的字符串时，返回 false；当返回 true 时表示接收完成。

用法举例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
string data
read(m, data, 0, 4)
close(m)
```

5.6.6.4 write(通过 modbus 向主站发送数据)

函数原型

```
bool write(modbus_dev dev,string data,in start)
bool write(modbus_dev dev,byte[] data,in start,int len)
```

描述

以 start 为起始地址、len 为长度(单位：字节)，将输入发送的字符串变量或字节数组变量写入 modbus 从站设备寄存器中。

参数

表 5-76 参数说明

参数名称	参数类型	说明
dev	modbus_dev	已经打开的 modbus 设备
data	string/byte[]	要发送的字符串变量或字节数组变量
start	int	所要写入数据的起始地址
len	int	所要写入数据的长度，写入 string 类型数据不需要该参数

返回值

类型：bool

发送未完成时，返回 false；当返回 true 时表示发送完成。

用法举例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
byte sdata[4]={31h,32h,33h,34h}
write(m, sdata, 0, 4)
close(m)
```

5.6.6.5 clearbuff(清空 modbus 从站的数据缓冲区)

函数原型

```
bool clearbuff(modbus_dev dev)
```

描述

清空 modbus 从站的数据缓冲区（当输入缓冲区为空时，通过 read 函数接受不到任何数据）。

参数

表 5-77 参数说明

参数名称	参数类型	说明
dev	modbus_dev	已经打开的 modbus 设备

返回值

类型：bool

返回 true 表示操作成功，false 表示操作失败。

用法举例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
string data
read(m, data, 0, 4)
clearbuff(m)
close(m)
```

5.6.6.6 setcycle(设置与 modbus 设备通讯的读写周期)

函数原型

```
void setcycle(modbus_dev& dev,int cycle)
```

描述

设置控制柜与 modbus 设备通讯的读写周期。

参数

表 5-78 参数说明

参数名称	参数类型	说明
dev	modbus_dev	已经打开的 modbus 设备

cycle	int	读写周期，单位：μs，参数范围：>500000

返回值

无

用法举例

```
modbus_dev m
open(m,"rtserMB0",1,115200,one,8,none)
setcycle(m,1000000)
clearbuff(m)
close(m)
```

5.7 数据类型转换函数

5.7.1 toint(转换成整型)

函数原型

```
int toint(double d)或
int toint(byte[] data,int start)
int toint(string number_string, base number_base)
```

描述

将 double 型数据类型、byte 数组或者字符串数据转换成整型。对于 byte 数组，系统会将从 data 数组的第 start 个字节开始的 4 个字节转换为整型数据。

参数

表 5-79 toint 函数的参数

名称	说明
d	类型：double
	被转换的 double 型数据
data	类型：byte[]
	被转换的 byte 数组
start	类型：int
	起始偏移地址
number_string	类型：string
	被转换的字符串数据
number_base	类型：base
	字符串数据的显示数制

返回值

类型：int

转换后的整型数据。

用法举例

```
double value =3.1415
int s = toint(value)
print s //输出“3”
byte data[4] = {01h,02h,03h,04h}
print hex,toint(data,0) //输出“4030201h”
```

5.7.2 todouble(转换成浮点型)

函数原型

```
double todouble(byte[] data,int start)
```

描述

将 byte 数组转换成浮点型。系统会将从 data 数组的第 start 个字节开始的 8 个字节转换为浮点型数据。

参数

表 5-80 todouble 函数的参数

名称	说明
data	类型：byte[]
	被转换的 byte 数组
start	类型：int
	起始偏移地址

返回值

类型：double

转换后的浮点型数据。

用法举例

```
byte data[8] = {00h,00h,00h,00h,00h,00h,08h,40h}
print data //输出 3
byte data[8] = {00h,00h,00h,00h,00h,00h,08h,C0h}
print data //输出-3
byte data[8] = {00h,00h,00h,00h,00h,00h,00h,40h}
print data //输出 2
```

5.7.3 tobytes(转换成字节数组)

函数原型

void tobytes(string/int/double src,byte[] dest,int start)

描述

将 string 型、int 型或 double 型数据类型转换成 byte 数组。int 数据会被转换为 4 个字节，double 数据会被转换为 8 个字节，string 类型转换成 byte，长度无法确定，要根据转换的字符串来看，不同的字符串长度是不同的。

参数

表 5-81 tobytes 函数的参数

名称	说明
dest	类型：byte[]
	转换后数据存入的 byte 数组
start	类型：int
	存储的起始偏移地址

返回值

类型：void

用法举例

```
byte result[4]
tobytes(14030201h,result,0)
print hex,result //输出“{01h,02h,03h,14h}”
```

5.7.4 tostr(强制转换成字符串)

函数原型

string tostr(anytype v)
string tostr(double v,int precision)
string tostr(int v, enum number_base) //将整型转化为 10 进制/16 进制显示的字符串

描述

将任意数据类型的数据转换成字符串类型。

参数

表 5-82 tostr 函数的参数

名称	说明
v	类型：可以是任意数据类型
	被转为字符串的表达式

名称	说明
	■ 当 v 是 double 类型时，可以指定第二个参数 precision 设置保留几位有效数字，如果缺省该参数，则默认保留 6 位有效数字 ■ 当 v 是 int 类型时，可以指定第二个参数 number_base 设置显示数制，如果缺省该参数，则默认显示十进制
precision	类型：int
	精度，保留的有效数字位数。
number_base	类型：enum 枚举类型
	进制转换类型，取值如下： ■ hex：十六进制 ■ dec：十进制

返回值

类型：string

返回转换后的字符串。

用法举例

```
double value =3.1415
string s = tostr(value)
print s //输出“3.1415”
```

5.7.5 ftobytes(浮点转换成字节)

函数原型

void ftobytes(double src, byte[] data,int start)

描述

将 double 型数据类型转换成 byte 数组，长度为 4 个字节，该转换会损失数值精度。

参数

名称	说明
src	类型 double 待转换的 double 型数据
data	类型：byte[] 转换后数据存入的 byte 数组
start	类型：int 存储的起始偏移地址

返回值

类型：void

用法举例

```
byte result[4]
double b = 1.23
ftobytes(b,result,0)
```

5.7.6 tofloat(字节转换成浮点)

函数原型

```
double tofloat(byte[] data,int start)
```

描述

将 byte 数组转换成浮点型。系统会将从 data 数组的第 start 个字节开始的 4 个字节转换为 double 型数据。

参数

名称	说明
data	类型：byte[] 被转换的 byte 数组
start	类型：int 起始偏移地址

返回值

类型：double

转换后的浮点型数据。

用法举例

```
byte result[4]
double b = 1.23
ftobytes(b,result,0)
print tofloat(result, 0) //输出 1.23
```

5.8 机器人位姿函数

5.8.1 cjoint(获取当前轴位置)

函数原型

```
joint cjoint()
```

描述

获取当前轴位置，包括外轴。后台不能使用。

参数

无

返回值

类型：joint

返回当前轴位置。

用法举例

```
joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}
movej a
waittime 0
print cjoint() //输出“{ 0, 10, 20, 30, 40, 50, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
//实际打印结果可能因误差原因与设定值略有偏差（位置精度范围内）。
```

5.8.2 cpose(获取当前 TCP 位姿)

函数原型

```
pose cpose(tool t,wobj w)
```

描述

获取当前 TCP 位姿以及外轴位置。

参数

表 5-83 cpose 函数的参数

名称	说明
t	类型：tool
	当前工具
w	类型：wobj
	参考的工件坐标系

返回值

类型：pose

返回当前 TCP 位姿以及外轴位置。

用法举例

```
pose p = { x 763.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}
ptp p
waittime 0
wobj wobj1 = {{10,0,0,0,0,0}}
```

```
print cpose($FLANGE,wobj1) //输出“{753.869,207.323,1422.841,129.538,0.480,92.084,0,-1,9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

5.8.3 getpose(获取某组轴位置对应的 TCP 位姿，即运动学正解)

函数原型

```
pose getpose(joint j,tool t,wobj w)
```

描述

获取轴位置 j 对应的 TCP 位姿以及外轴位置。

参数

表 5-84 getpose 函数的参数

名称	说明
j	类型：joint
	机器人及外轴轴位置
t	类型：tool
	指定的工具
w	类型：wobj
	参考的工件坐标系

返回值

类型：pose

返回对应的 TCP 位姿以及外轴位置。

用法举例

```
joint j = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}
wobj wobj1 = {{10,0,0,0,0,0}}
print getpose(j,$FLANGE,wobj1)
```

5.8.4 getjoint(获取某个 TCP 位姿对应的机器人轴位置，即运动学逆解)

函数原型

```
joint getjoint(pose p,tool t,wobj w)
```

描述

获取位姿 p 对应的机器人各轴位置。

参数

表 5-85 getjoint 函数的参数

名称	说明
p	类型：pose
	机器人位姿，参考第 2.4.4 章节
t	类型：tool
	指定的工具，参考第 2.4.6 章节
w	类型：wobj
	参考的工件坐标系，参考第 2.4.7 章节

返回值

类型：joint

返回对应的机器人轴位置数据。

用法举例

```
pose p = {x 763.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}
print getjoint(p,$FLANGE,$WORLD)
```

5.8.5 poseinv(计算一个 pose 的逆 pose)

函数原型

```
pose poseinv(pose p)
```

描述

计算一个 pose 的逆 pose。

参数

表 5-86 poseinv 函数的参数

名称	说明
p	类型：pose
	输入一个坐标系 A 在另一个坐标系 B 中的位姿

返回值

类型：pose

返回坐标系 B 在坐标系 A 中的位姿。

用法举例

```
pose p = {x 1,y 2,z 3,a 0,b 0,c 0}
```

```
print poseinv(p)
//输出“{-1,-2,-3,0,0,0,0,-1,9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

5.8.6 offset(目标点相对于工件坐标系的移位函数)

函数原型

```
pose offset(pose p,double dx,double dy,double dz,double rz,double ry,double rx)
```

描述

offset 函数用于将指定的目标点相对于工件坐标系进行平移及旋转，并获取平移后的新的目标点。

参数

表 5-87 offset 函数的参数

名称	说明
p	类型：pose
	当前目标点
dx	类型：double
	沿工件坐标系 x 方向的平移量，单位毫米（mm）
dy	类型：double
	沿工件坐标系 y 方向的平移量，单位毫米（mm）
dz	类型：double
	沿工件坐标系 z 方向的平移量，单位毫米（mm）
rz	类型：double
	绕工件坐标系 z 方向的旋转量，单位度（°）
ry	类型：double
	绕工件坐标系 y 方向的旋转量，单位度（°）
rx	类型：double
	绕工件坐标系 x 方向的旋转量，单位度（°）

注意

如果同时指定了 rz、ry、rx 中的多个值，则采用的是 ZYX 型欧拉角进行旋转，即旋转顺序为先绕 z 轴旋转 rz 角度，再绕新的 y 轴旋转 ry 角度，最后绕新的 x 轴旋转 rx 角度。

返回值

类型：pose

平移后的目标点。

用法举例

```
pose p = {x 1500,y 500,z 500,a 0,b 0,c 0}
double dx = 10
double dy = 20
double dz = 30
double rz = 60
double ry = 50
double rx = 40
p=offset (p,dx,dy,dz,rz,ry,rx)
ptp p:p,vp:5%,sp:5%
waittime 0
print cpose() //输出“{1510, 520, 530, 40, 50, 60, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”，实际打印结果可能因误差原因与设定值略有偏差（位置精度范围内）。
```

5.8.7 reltool(目标点相对于工具坐标系的移位函数)

函数原型

pose reltool(pose p,double dx,double dy,double dz,double rz,double ry,double rx)

描述

reltool 函数用于将指定的目标点相对于工具坐标系进行平移及旋转，并获取移位后的新的目标点。

参数

表 5-88 reltool 函数的参数

名称	说明
p	类型：pose
	当前目标点
dx	类型：double
	沿工具坐标系 x 方向的平移量，单位毫米（mm）
dy	类型：double
	沿工具坐标系 y 方向的平移量，单位毫米（mm）
dz	类型：double
	沿工具坐标系 z 方向的平移量，单位毫米（mm）
rz	类型：double
	绕工具坐标系 z 方向的旋转量，单位度（°）
ry	类型：double
	绕工具坐标系 y 方向的旋转量，单位度（°）
rx	类型：double
	绕工具坐标系 x 方向的旋转量，单位度（°）


注意

这里采用的是 ZYX 型欧拉角进行旋转，即旋转顺序为先绕 z 轴旋转 rz 角度，再绕新的 y 轴旋转 ry 角度，最后绕新的 x 轴旋转 rx 角度。

返回值

类型：pose

移位后的目标点。

用法举例

```
pose p1= {x 1500,y 500,z 500,a 0,b 0,c 0}
double dx = 10
double dy = 20
double dz = 30
double rz = 40
double ry = 50
double rx = 60
pose p2 = reltool(p1, dx, dy, dz, rz, ry, rx)
ptp p:p2,vp:5%,sp:5%
waittime 0
print cpose($FLANGE,WORLD) //输出“{1510,520,530,40,50,60,0,-1,9e+09,9e+09, 9e+09, 9e+09, 9e+09, 9e+09}，实际打印结果可能因误差原因与设定值略有偏差（位置精度范围内）。
```

5.8.8 cjttq(获取机器人各个轴的输出力矩)

描述

jttq 结构体类型用于直接描述机器人各轴的输出力矩。该函数用于获取指定前台通道当前各个轴的输出力矩（单位 N*m）。在前台通道、后台通道均可使用。

格式

```
jttq cjttq(int chan_no, bool Is_cmd)
```

定义

```
struct jttq
{
double jt1
double jt2
double jt3
double jt4
double jt5
double jt6
double et1
double et2
double et3
```

double et4

double et5

double et6

}

成员

cjttq 指令的成员详见表 5-89。

表 5-89 cjttq 指令的成员

名称	说明
j1~j6	类型：double
	机器人 1 轴~6 轴的输出力矩，单位 N*M（牛米）
et1~et6	类型：double
	外轴 1 轴～6 轴的输出力矩，单位 N*M（牛米）

参数

cjttq 指令的参数详见表 5-90。

表 5-90 cjttq 指令的参数

名称	说明
chan_no	类型：int
	指定前台通道号
Is_cmd	类型：bool
	true—输出指令力矩，false—输出反馈力矩

返回值

类型：jttq

返回指定前台通道当前各个轴的输出力矩（单位 N*m）。

用法举例

//前台通道 1 执行程序

pose p = {x 763.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}

ptp p

print cjttq(1,true) //输出“{*,*,*,*,*,*,*,*,*,*,*,*,*}”，*表示力矩是未知状态，反映当下时刻各个轴的力矩数值。

5.8.9 cjtci(获取机器人各个轴的电机电流)

描述

jtci 结构体类型用于直接描述机器人各轴的电机电流 Iq。该函数用于获取指定前台通道当前各个轴的电机电流（单位 A 安培）。在在前台通道、后台通道均可使用。

格式

jtci cjtci(int chan_no, bool Is_cmd)

定义

```
struct jtci
{
double ji1
double ji2
double ji3
double ji4
double ji5
double ji6
double ei1
double ei2
double ei3
double ei4
double ei5
double ei6
}
```

成员

cjtci 指令的成员详见表 5-91。

表 5-91 cjtci 指令的参数

名称	说明
ji1~ji6	类型：double
	机器人 1 轴~6 轴的电机电流 Iq，单位 A 安培
ei1~ei6	类型：double
	外轴 1 轴～6 轴的电机电流 Iq，单位 A 安培

参数

cjtci 指令的参数详见表 5-92。

表 5-92 cjtci 指令的参数

名称	说明
chan_no	类型：int
	指定前台通道号

名称	说明
Is_cmd	类型：bool
	是否为指令电流：true—输出指令电流，false—输出反馈电流

返回值

类型：jtci

返回指定前台通道当前各个轴的电机电流（单位 A 安培）。

用法举例

//前台通道 1 执行程序

pose p = {x 763.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}

ptp p

print cjtci(1,true) //输出“{*,*,*,*,*,*,*,*,*,*,*,*,*}”，*表示电流是未知状态，反映当下时刻各个轴的电机电流。

5.8.10 channeltojoint(获取某通道的机器人运行目标点的轴位置)

描述

获取指定前台通道运行目标点的轴位置，包括外轴，在前台通道、后台通道均可使用。

格式

joint channeltojoint(int chan_no)

参数

channeltojoint 指令的参数详见表 5-93。

表 5-93 channeltojoint 指令的参数

名称	说明
chan_no	类型：const int
	指定前台通道号

返回值

类型：joint

返回指定通道运行目标点各个轴的位置。

用法举例

//前台通道 1 执行程序

joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}

movej a

print channeltojoint(1) //输出“{ 0, 10, 20, 30, 40, 50, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”

5.8.11 channeltopose(获取某通道的机器人运行目标点 TCP 位姿)

描述

获取指定前台通道运行目标点 TCP 位姿以及外轴位置。再在前台通道、后台通道均可使用。

格式

pose channeltopose(int chan_no,tool t,wobj w)

参数

channeltopose 指令的参数详见表 5-94。

表 5-94 channeltopose 指令的参数

名称	说明
chan_no	类型：int
	指定前台通道号
t	类型：tool
	当前工具
w	类型：wobj
	参考的工件坐标系

返回值

类型：pose

返回指定前台通道当前 TCP 位姿以及外轴位置。

用法举例

//前台通道 1 执行程序

pose p = {x 753.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}

ptp p

wobj wobj1 = {{10,0,0,0,0,0}}

print channeltopose(1,$FLANGE,wobj1) //输出“{753.869,207.323, 1422.841,129.538,0.480,92.084,0,-1,9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”

5.8.12 channeljoint(获取指定前台通道当前轴位置)

描述

获取指定前台通道当前轴位置，包括外轴，在前台通道、后台通道均可使用。

格式

joint channeljoint(int chan_no,bool Is_cmd)

参数

channeljoint 指令的参数详见表 5-95。

表 5-95 channeljoint 指令的参数

名称	说明
chan_no	类型：int
	指定前台通道号
Is_cmd	类型：bool
	是否为指令位置： ■ true—输出指令位置，即当本体处于运行状态时，则反回的指令位置为当前轨迹的目标点。 ■ false—输出反馈位置，即反馈本体当前的点位。

返回值

类型：joint

返回指定前台通道当前轴位置。

用法举例

```
//前台通道 1 执行程序
joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}
movej a
waittime 0
//后台通道执行程序
print channeljoint(1,true) //输出“{0, 10, 20, 30, 40, 50, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

5.8.13 channelpose(获取指定前台通道当前 TCP 位姿)

描述

获取指定前台通道当前 TCP 位姿以及外轴位置，在前台通道、后台通道均可使用。

格式

pose channelpose(int chan_no,tool t,wobj w,bool Is_cmd)

参数

channelpose 指令的参数详见表 5-96。

表 5-96 channelpose 指令的参数

名称	说明
chan_no	类型：int
	指定前台通道号
t	类型：tool

名称	说明
	当前工具
w	类型：wobj
	参考的工件坐标系
Is_cmd	类型：bool
	是否为指令位置： ■ true—输出指令位置，即当本体处于运行状态时，则反回的指令位置为当前轨迹的目标点。 ■ false—输出反馈位置，即反馈本体当前的点位。

返回值

类型：pose

返回指定前台通道当前 TCP 位姿以及外轴位置。

用法举例

```
//前台通道 1 执行程序
pose p = {x 753.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}
ptp p
waittime 0
wobj wobj1 = {{10,0,0,0,0,0}}
//后台通道执行程序
print channelpose(1,$FLANGE,wobj1,true) //输出“{753.869,207.323, 1422.841,129.538,0.480,92.084,0,-1,9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

5.8.14 channeljointvel(获取机器人各个轴的速度)

描述

获取指定前台通道当前各个轴的速度（单位 rad/s），包括外轴。在前台通道、后台通道均可使用。

格式

jvel channeljointvel(int chan_no,bool Is_cmd)

参数

channeljointvel 指令的参数详见表 5-97。

表 5-97 channeljointvel 指令的参数

名称	说明
chan_no	类型：const int
	指定前台通道号
Is_cmd	类型：bool
	是否为指令速度：

名称	说明
	■ true—输出指令位置，即当本体处于运行状态时，则反回的指令位置为当前轨迹的目标点。 ■ false—输出反馈位置，即反馈本体当前的点位。

返回值

类型：jvel

返回指定前台通道当前各个轴的速度（单位 rad/s），jvel 结构体定义请参考第 2.4.11 章节。

用法举例

//前台通道 1 执行程序

joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}

movej a

print channeljointvel(1,$FLANGE,wobj1,true) //输出“{ *,*, *, *, *, *, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”，*表示打印当下的轴速度是未知状态，*的具体数值反映当下时刻速度。

5.8.15 channeltcpvel(获取机器人 TCP 点速度)

描述

获取指定前台通道当前 TCP 点的速度（单位 mm/s）。在前台通道、后台通道均可使用。

格式

double channeltcpvel(int chan_no, tool t,wobj w,bool Is_cmd)

参数

channeltcpvel 指令的参数详见表 5-98。

表 5-98 channeltcpvel 指令的参数

名称	说明
chan_no	类型：const int
	指定前台通道号
t	类型：tool
	当前使用的工具
w	类型：wobj
	当前参考的工件坐标系
Is_cmd	类型：bool
	是否为指令速度：true—输出指令位置，即当本体处于运行状态时，则反回的指令位置为当前轨迹的目标点。false—输出反馈位置，即反馈本体当前的点位。

返回值

类型：double

返回指定前台通道当前 TCP 点的速度（单位 mm/s）。

用法举例

```
//前台通道 1 执行程序
joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}
movej a
print channeltcpvel(1,$FLANGE,wobj1,true) //输出“*”，*表示打印当下的 TCP 速度是未知状态，*的具体数值反映当下时刻速度。
```

5.8.16 ctcpforce(获取机器人 TCP 点六维力矢量)

描述

tcpforce 结构体类型使用笛卡尔坐标的方式来描述机器人 TCP 点输出力和力矩。该函数用于获取指定前台通道当前 TCP 点输出力（单位 N）和输出力矩（单位 N*m）。在前台通道使用。

格式

```
tcpforce ctcpforce(int chan_no, tool t,wobj w)
```

定义

```
struct tcpforce
{
double fx
double fy
double fz
double tx
double ty
double tz
}
```

成员

ctcpforce 指令的成员详见表 5-99。

表 5-99 ctcpforce 指令的参数

名称	说明
fx	类型：double
	机器人 TCP 点输出力相对当前工件坐标系坐标的 x 分量，单位 N(牛)
fy	类型：double
	器人 TCP 点输出力相对当前工件坐标系坐标的 y 分量，单位 N(牛)
fz	类型：double
	器人 TCP 点输出力相对当前工件坐标系坐标的 z 分量，单位 N(牛)
tx	类型：double
	机器人工具输出力矩在当前工件坐标系的 x 分量，单位 N*m(牛米)

名称	说明
ty	类型：double
	机器人工具输出力矩在当前工件坐标系的 y 分量，单位 N*m(牛米)
tz	类型：double
	机器人工具输出力矩在当前工件坐标系的 z 分量，单位 N*m(牛米)

参数

ctcpforce 指令的参数详见表 5-100。

表 5-100 ctcpforce 指令的参数

名称	说明
chan_no	类型：int
	指定前台通道号
t	类型：tool
	当前使用的工具
w	类型：wobj
	当前参考的工件坐标系

返回值

类型：tcpforce

返回指定前台通道当前 TCP 点输出力（单位 N）和输出力矩（单位 N*m）。

用法举例

//前台通道 1 执行程序

pose p = {x 763.869,y 207.323,z 1422.841,a 129.538,b 0.480,c 92.084,cfg 0}

ptp p

wobj wobj1 = {{10,0,0,0,0,0}}

print ctcpforce(1,$FLANGE,wobj1) //输出“{*,*,*,*,*,*}”，*表示力和力矩是未知状态，表示当下时刻 TCP 点的力和力矩数值。

5.9 坐标系标定函数

5.9.1 getwobj_3p(3 点法标定工件坐标系)

函数原型

wobj getwobj_3p(joint j1, joint j2, joint j3,tool t)

描述

通过 3 个示教点，标定工件坐标系。

参数

表 5-101 getwobj_3p 函数的参数

名称	说明
j1	类型：joint
	待标定的工件坐标系的原点对应的示教点
j2	类型：joint
	待标定的工件坐标系的 X 轴正向一点对应的示教点
j3	类型：joint
	待标定的工件坐标系的 XY 平面上 Y 轴为正的一点对应的示教点
t	类型：tool
	标定时使用的工具

返回值

类型：wobj

返回计算出的工件坐标系标定结果。

用法举例

```
joint j1 = {j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015, j6 -0.139}
joint j2 = {j1 10,j2 0.005,j3 120.001,j4 0.001,j5 30.015, j6 -0.139}
joint j3 = {j1 15,j2 0.005,j3 120.001,j4 0.001,j5 30.015, j6 -0.139}
print getwobj_3p(j1,j2,j3,$TOOL0)
```

5.9.2 getwobj_indi(间接法标定工件坐标系)

函数原型

```
wobj getwobj_indi(joint j1, joint j2, joint j3,pos p1,pos p2,pos p3,tool t)
```

描述

通过已知在待标定工件坐标系中坐标的 3 个测试点，间接标定工件坐标系。

参数

表 5-102 getwobj_indi 函数的参数

名称	说明
j1	类型：joint
	p1 点对应的示教点
j2	类型：joint
	p1 点对应的示教点

名称	说明
j3	类型：joint
	p3 点对应的示教点
p1	类型：pos
	p1 点在待标定工件坐标系中的坐标
p2	类型：joint
	p2 点在待标定工件坐标系中的坐标
p3	类型：joint
	p3 点在待标定工件坐标系中的坐标
t	类型：tool
	标定时使用的工具

返回值

类型：wobj

返回计算出的工件坐标系标定结果。

用法举例

```
joint j1 = j:{j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j2 = j:{j1 10,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j3 = j:{j1 15,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
pos p1 = {10,20,20}
pos p2 = {40,60,20}
pos p3 = {50,20,40}
print getwobj_indi(j1,j2,j3,p1,p2,p3, $TOOL0)
```

5.9.3 getwobj_flange(法兰参照法标定工件坐标系)

函数原型

wobj getwobj_flange(joint j1, joint j2, tool t,bool xydefault)

描述

该函数通过一个标准工具标定工件坐标系或外部固定工具坐标系原点，通过机器人法兰坐标系参照标定工件坐标系或外部固定工具坐标系方向。该标定方法主要用于外部工具坐标系的标定。

标定方法：

- 在机器人法兰上安装好标准工具。
- 用标准工具的 TCP 点驶向待测坐标系的原点，记下此时各轴位置角度。
- 将法兰坐标系的 Z-方向与待测坐标系的 Z+方向移至平行。
- 如果希望标定待测坐标系的 X，Y 方向，则还需要将法兰坐标系的 X+方向与待测坐标系的 X+方向移至平行。如果不关心待测坐标系的 X，Y 方向，则这步可以省略。

- 记下此时各轴位置角度。
- 执行该函数计算得到待测工件坐标系。

参数

表 5-103 getwobj_flange 函数的参数

名称	说明
j1	类型：joint
	标定原点时记录的轴位置
j2	类型：joint
	标定方向时记录的轴位置
t	类型：tool
	标定原点时法兰端安装的标准工具，这里只用到工具中的 xyz 分量，abc 分量忽略
xydefault	类型：bool
	是否默认 xy 轴方向 ■ true：只标定 z 轴方向，xy 轴方向由系统默认 ■ false：xyz 轴方向均标定

返回值

类型：wobj

返回计算出的工件坐标系标定结果。

用法举例

```
tool knowntool = {{0,0,0,0,0,0}}
joint j1 = { j1 4.229 ,j2 42.310 ,j3 84.665, j4 88.315, j5 84.614 ,j6 -36.429}
joint j2 = { j1 3.456, j2 41.824, j3 86.464 ,j4 92.073, j5 84.982, j6 -36.506}
wobj w = getwobj_flange(j1,j2,knowntool,true)
print w
$TOOLS[0].t_frame = w.w_frame
$TOOLS[0].stationary = true
```

5.9.4 gettooltcp_ref(标准工具参照法标定工具坐标系 xyz)

函数原型

void gettooltcp_ref(joint j1, joint j2, tool ref,tool& t)

描述

通过标准工具标定工具坐标系 x，y，z。

标定方法：

- 选取一个参考点和一个已知 TCP 坐标的参考工具。

- 将该参考工具安装于机器人法兰，手动控制机器人靠近参考点，记录下该示教点。
- 将参考工具卸下，将待测工具安装于机器人法兰，手动控制机器人靠近参考点，记录下该示教点。

参数

表 5-104 gettooltcp_ref 函数的参数

名称	说明
j1	类型：joint
	安装参考工具时对应的示教点
j2	类型：joint
	安装待测工具时对应的示教点
tool	类型：ref
	参考工具，只需知道 x，y，z 值
tool	类型：tool
	待标定的工具坐标系，标定结果将会保存到该变量的 x，y，z 中

返回值

无

用法举例

```
joint j1 = j:{j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j2 = j:{j1 10,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
tool ref = {{x 10,y 10,z 50}}
tool t
gettooltcp_ref(j1,j2,ref,t)
print t
```

5.9.5 gettoolrot_world(世界坐标系参照法测量工具坐标系 abc)

函数原型

```
void gettoolrot_world(joint j,tool& t)
```

描述

通过将工具坐标系的轴调整为与世界坐标系的轴平行，由此计算工具坐标系的 abc。

标定方法：

- 将 $+X_{TOOL}$ 与 $-Z_{WORLD}$ 调成平行，$+Y_{TOOL}$ 与 $+Y_{WORLD}$ 平行，$+Z_{TOOL}$ 与 $+X_{WORLD}$ 平行，记下此时的示教点。

参数

表 5-105 gettoolrot_world 函数的参数

名称	说明
j	类型：joint
	记录的示教点
tool	类型：tool
	待标定的工具坐标系，标定结果将会保存到该变量的 a，b，c 中

返回值

无

用法举例

```
joint j = j:{j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
tool t
gettoolrot_world (j,t)
print t
```

5.9.6 gettoolrot_3p(3 点法测量工具坐标系 abc)

函数原型

void gettoolrot_3p(joint j1,joint j2,joint j3,tool& t)

描述

通过 3 点法标定工具坐标系 abc。

标定方法：

- 用工具 TCP 驶向任何一个参考点，记下此时示教点。
- 用工具坐标系 X 轴负向的一个点驶向参考点，记下此时示教点。
- 用工具坐标系 XY 平面上的 Y 轴为负值的一点驶向参考点，记下此时示教点。

参数

表 5-106 gettoolrot_3p 函数的参数

名称	说明
j1	类型：joint
	第 1 步获取的示教点
j2	类型：joint
	第 2 步获取的示教点
j3	类型：joint
	第 3 步获取的示教点

名称	说明
tool	类型：tool
	待标定的工具坐标系，t_frame 分量的 x，y，z 分量需传入正确的值，标定结果将会保存到该变量的 t_frame 分量的 a，b，c 分量中

返回值

无

用法举例

```
joint j1 = j:{j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j2 = j:{j1 10,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j3 = j:{j1 15,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
tool t
gettoolrot_3p (j1,j2,j3,t)
print t
```

5.9.7 gettool_3p(3 点法标定工具坐标系)

函数原型

```
tool gettool_3p(joint j1, joint j2, joint j3,wobj w)
```

描述

该函数通过一个已知坐标的参考点来标定相对法兰坐标系定义的工具坐标系或机器人抓取工件坐标系。该标定方法主要用于标定机器人抓取工件坐标系。

标定方法：

- 用坐标系原点驶向已知参考点，记下此时示教点。
- 用坐标系 X 轴正向的一个点驶向已知参考点，记下此时示教点。
- 用坐标系 XY 平面上的 Y 轴为正值的一点驶向已知参考点，记下此时示教点。

参数

表 5-107 gettool_3p 函数的参数

名称	说明
j1	类型：joint
	第 1 步获取的示教点
j2	类型：joint
	第 2 步获取的示教点
j3	类型：joint
	第 3 步获取的示教点
w	类型：wobj

名称	说明
	已知在世界坐标系中坐标值的参考点，这里只用到工件坐标系中的 xyz 分量，abc 分量忽略

返回值

类型：tool

返回计算出的工具坐标系标定结果。

用法举例

```
wobj w = {{935,0,1300,0,0,0}}
joint j1 = {0,0,90,0,90,0}
joint j2 = {0,0,90,0,-90,0}
joint j3 = {0,0,90,90,90,0}
tool t = gettool_3p(j1,j2,j3,w)
print t
$WOBJS[0].w_frame = t.t_frame
$WOBJS[0].robhold = true
```

5.9.8 getbase_3p(3 点法标定基础坐标系)

函数原型

void getbase_3p(joint j1, joint j2, joint j3,tool t, int index)

描述

通过 3 个示教点，标定基础坐标系。

参数

表 5-108 getbase_3p 函数的参数

名称	说明
j1	类型：joint
	世界坐标系原点对应的示教点
j2	类型：joint
	世界坐标系 X 轴正向一点对应的示教点
j3	类型：joint
	世界坐标系 XY 平面上 Y 轴为正的一点对应的示教点
t	类型：tool
	标定时使用的工具
index	类型：int
	机械单元序号，参数范围：[1,n]。其中 n 为当前通道的机械单元总数

返回值

无

用法举例

```
joint j1 = j:{j1 1,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j2 = j:{j1 10,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
joint j3 = j:{j1 15,j2 0.005,j3 120.001,j4 0.001,j5 30.015 ,j6-0.139}
getbase_3p(j1,j2,j3, $TOOL0,1)
print $BASE
```

5.10 轨迹触发相关函数

5.10.1 T(当前运动轨迹是否到达某个时间点)

函数原型

bool T(double t)

描述

返回当前运动轨迹距离起点是否已经走了 t 秒。该函数主要用于轨迹触发中的事件定义。

参数

表 5-109 T 函数的参数

名称	说明
t	类型：double
	单位秒

返回值

类型：bool

- false：当前运动轨迹尚未到达 t 时间点。
- true：当前运动轨迹已经到达 t 时间点。

用法举例

```
trigger 1,when:T(1),do:setdo(1,true) //在下面的轨迹上声明一个轨迹触发：当距离起点时间 1 秒时第 1 路 DO 输出信号 true
movej j:{j1 30},v:{per 2}
```

5.10.2 S(当前运动轨迹是否到达某个路程点)

函数原型

bool S(double s)

描述

返回当前运动轨迹距离起点是否已经走了 s 毫米。该函数主要用于轨迹触发中的事件定义。

参数

表 5-110 S 函数的参数

名称	说明
s	类型：double
	单位毫米

返回值

类型：bool

- false：当前运动轨迹尚未到达距离起点 s 的位置点。
- true：当前运动轨迹已经到达距离起点 s 的位置点。

用法举例

trigger 1,when:S(100),do:setdo(1,true) //在下面的轨迹上声明一个轨迹触发：当距离起点时间 100 毫米时第 1 路 DO 输出信号 true

lin p1,v:{tcp 20,ori 10}

5.10.3 StoEnd(当前运动轨迹是否到达距离目标点某个距离的位置点)

函数原型

bool StoEnd(double s)

描述

返回当前运动轨迹距离目标点是否还剩 s 毫米。该函数主要用于轨迹触发中的事件定义。

参数

表 5-111 StoEnd 函数的参数

名称	说明
s	类型：double
	单位毫米

返回值

类型：bool

- false：当前运动轨迹尚未到达距离目标点还剩 s 毫米的点。
- true：当前运动轨迹已经到达距离目标点还剩 s 毫米的点。

用法举例

trigger 1,when:StoEnd(100),do:setdo(1,true) //在下面的轨迹上声明一个轨迹触发：当距离目标点 100 毫米时第 1 路 DO 输出信号 true

lin p1,v:{tcp 20,ori 10}

5.11 点位配方修改的相关 ARL 函数

5.11.1 savearl(复制 arl 文件)

函数原型

bool savearl(const string& file_src, const string& file_dst, bool mode)

描述

复制当前文件夹下指定名称的 arl 文件，并对新 arl 文件进行命名

参数

表 5-112 savearl 函数的参数

名称	说明
file_src	类型：string
	原 arl 文件名称
file_dst	类型：string
	复制后的 arl 文件名称
mode	类型：bool
	是否覆盖同名文件。true：覆盖同名文件；false：不覆盖同名文件

返回值

类型：bool

复制成功则返回 true，复制失败则返回 false。

用法举例

print savearl("Recipe1.arl",Recipe2.arl",true) //当前文件夹下已有 Recipe2.arl 文件，则输出 true，并覆盖 Recipe2.arl 文件

print savearl("Recipe1.arl","Recipe2.arl",false) //当前文件夹下已有 Recipe2.arl 文件，则输出 false，不做任何操作

5.11.2 savefilepose(将点位信息保存至 data 文件)

函数原型

bool savefilepose(const string& file_name,const string& pose_name,const pose& p)

描述

将 pose 点位信息保存至指定 arl 文件的指定点位中（重新加载后生效）

参数

表 5-113 savefilepose 函数的参数

名称	说明
file_name	类型：string
	存入点位的_data.arl 文件名称
p_name	类型：string
	存入点位信息的 pose 变量名称
p	类型：pose
	待存入的 pose 变量

返回值

类型：bool

存入成功则返回 true，存入失败则返回 false。

用法举例

```
pose a = {x 0,y 10,z 15,a 0,b 90,c 0,cfg 0,ej1 20}
savefilepose("Recipe1_data.arl", p1, a) //将{x 0,y 10,z 15,a 0,b 90,c 0,cfg 0,ej1 20}保存至 Recipe1_data.arl 的 p1 中。
```

5.11.3 savefilejoint(将点位信息保存至 data 文件)

函数原型

bool savefilejoint(const string& file_name,const string& pose_name,const joint& j)

描述

将 joint 点位信息保存至指定 arl 文件的指定点位中（重新加载后生效）

参数

表 5-114 savefilejoint 函数的参数

名称	说明
file_name	类型：string
	存入点位的_data.arl 文件名称
joint_name	类型：string
	存入点位信息的 joint 变量名称
j	类型：joint
	待存入的 joint 变量

返回值

类型：bool

存入成功则返回 true，存入失败则返回 false。

用法举例

joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}

savefilejoint("Recipe1_data.arl", j1, a) //将{j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}保存至 Recipe1_data.arl 中的 j1 点位。

5.11.4 saveposenow(将点位信息保存至某通道的程序)

函数原型

bool saveposenow(unsigned int channel, const string& file_name, const string& pose_name, const pose& p)

描述

将 pose 类型点位信息保存至指定前台通道下指定程序的指定点位中（立即生效）。

参数

表 5-115 saveposenow 函数的参数

名称	说明
channel	类型：unsigned int
	存入点位的通道号
file_name	类型：string
	存入点位的 arl 程序名
pose_name	类型：string
	存入点位信息的 pose 变量名称
p	类型：pose
	待存入的 pose 变量

返回值

类型：bool

存入成功则返回 true，存入失败则返回 false。

用法举例

pose a = {x 0,y 10,z 15,a 0,b 90,c 0,cfg 0,ej1 20}

saveposenow(1,"prog1.arl","p1", a) //将{x 0,y 10,z 15,a 0,b 90,c 0,cfg 0,ej1 20}保存至前台通道 1 下 prog1.arl 的 p1 中。

5.11.5 savejointnow(将点位信息保存至某通道的程序)

函数原型

bool savejointnow(unsigned int channel,const string& file_name,const string& pose_name,const joint& j)

描述

将 joint 类型点位信息保存至指定前台通道下指定程序的指定点位中（立即生效）。

参数

表 5-116 savejointnow 函数的参数

名称	说明
channel	类型：unsigned int
	存入点位的通道号
file_name	类型：string
	存入点位的 arl 程序名
joint_name	类型：string
	存入点位信息的 joint 变量名称
j	类型：joint
	待存入的 joint 变量

返回值

类型：bool

存入成功则返回 true，存入失败则返回 false。

用法举例

joint a = {j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}

savejointnow(1,"prog1.arl","j1", a) //将{j1 0,j2 10,j3 20,j4 30,j5 40,j6 50}保存至前台通道 1 下 prog1.arl 的 j1 中。

5.11.6 switcharl(切换前台通道加载的 arl 程序)

函数原型

bool switcharl(unsigned int channel,const string& file_name)

描述

将指定通道的程序暂停、复位、卸载，并将指定名称的 arl 文件加载到该通道中并运行。仅允许在后台通道使用。

参数

表 2-6 switcharl 函数的参数

名称	说明
channel	类型：unsigned int
	前台通道号，范围：[1,6]
file_name	类型：string
	待加载的 arl 程序文件名

返回值

类型：bool

切换成功则返回 true，失败则返回 false。

用法举例

print switcharl(1,"Recipe1.arl") //输出 true，并将 Recipe1.arl 加载至前台通道 1

5.12 动力学函数

5.13 其他函数

5.13.1 typeof(获取参数类型名)

函数原型

string typeof(anytype v)

描述

返回任意数据类型的类型名。

参数

表 5-117 typeof 函数的参数

名称	说明
v	类型：可以是任意数据类型的表达式
	待获取类型的表达式

返回值

类型：string

返回该类型的类型名字符串。

用法举例

double value =3.1415

print typeof(value) //输出“double”

5.13.2 ctime(获取当前时间字符串)

函数原型

string ctime()

描述

以字符串的形式表示当前的时间。

参数

无

返回值

类型：string

返回当前时间，用“时：分：秒”的字符串形式表示。

用法举例

print ctime() //假设当前时间是 10 点 10 分 10 秒，则输出“10:10:10”

5.13.3 cdate(获取当前日期字符串)

函数原型

string cdate()

描述

以字符串的形式表示当前日期。

参数

无

返回值

类型：string

返回当前日期，用“年-月-日”的字符串形式表示。

用法举例

print cdate() //假设当前日期是 2014 年 10 月 25 日，则输出“2014-11-25”

5.13.4 assert(断言)

函数原型

void assert(bool x)

描述

判断 x 的值是否为真，如果不为真，则报出警告，并说明 assert 函数所在的文件和行号。

参数

表 5-118 assert 函数的参数

名称	说明
x	类型：bool
	被断言的 bool 型表达式

返回值

无

用法举例

```
int a = 6
assert(a == 5) //程序执行到这一行将退出，并报出错误
```

5.13.5 savesv(存储系统变量)

函数原型

```
void savesv(string svname)
```

描述

存储变量名为 svname 的系统变量。

参数

表 5-119 savesv 函数的参数

名称	说明
svname	类型：string
	预保存的系统变量的名字。如果输入的变量名不存在，则会产生一个运行时错误

返回值

类型：void

用法举例

```
savesv("TOOLS") //保存工具系统变量，这样下次再启动系统时 TOOLS 的值不会丢失。
```

5.13.6 init(恢复系统变量为默认值)

函数原型

```
void init()
```

描述

恢复第 10.2 章节中说明的系统变量为默认值。

参数

无

返回值

无

用法举例

```
$DFSPPED = {10,50,5,5,5}
init()
print $DFSPPED //这里输出“{5,50,5,5,5}”
```

5.13.7 gettextstr(读取文本文件中的某一行内容)

函数原型

gettextstr(string file_path,int line_no,bool external_file)

描述

用于读取文本文件中的某一行内容。

参数

表 5-120 gettextstr 函数的参数

名称	说明
file_path	类型：string
	文件名，例如 script 目录下的文件 1.txt，则输入文件名为 1.txt，如果 script 目录下的文件 subdir/2.txt，则输入文件名为 subdir/2.txt
line_no	类型：int
	行号，从 1 开始，当输入文件名不存在时，会报警 10064，文件不存在或损坏，当输入行号大于文件内容最大行号时，报警 10065，输入行号大于文件最大行号
external_file	类型：bool
	文件存储位置类型，缺省默认值为 false。true 代表外部存储文件，读取/home/USB 目录下的文件；false 代表内部存储文件，读取/home/ae/script 目录下的文件

返回值

类型：string

用法举例

gettextstr("test1.txt",5, false) //读取路径/home/ae/script 目录下 test1.txt 第 5 行的内容；

gettextstr("test/test1.txt",5, true) //读取路径/home/ae/USB/test 目录下 test1.txt 第 5 行的内容；

5.13.8 curmpfile(获取运动指针所在 arl 文件名)

函数原型

string curmpfile(int channel_index)

描述

用于获取运动指针所在 arl 文件名。


注意

控制柜断电时会保存 arl 文件名，当控制柜重启后，程序名维持不变；当前台通道运行后，程序名会随着程序的运行被更新。

参数

表 5-121 curmpfile 函数的参数

名称	说明
channel_index	类型：int
	为前台通道号，起始值为 1

返回值

类型：string

用法举例

string s = curmpfile(1) //读取前台通道 1 运动指针所在的 arl 文件名。

5.13.9 curmpline(获取运动指针行号)

函数原型

int curmpline(int channel_index)

描述

用于获取运动指针行号。


注意

控制柜断电时会保存指针行号，当控制柜重启后，指针行号维持不变；当前台通道运行后，指针行号会随着程序的运行被更新。

参数

表 5-122 currpline 函数的参数

名称	说明
channel_index	类型：int
	为前台通道号，起始值为 1

返回值

类型：int

用法举例

int i = currpline(1) //读取前台通道 1 的运动指针行号。

5.13.10curppfile(获取程序指针所在 arl 文件名)

函数原型

string curppfile(int channel_index)

描述

用于获取程序指针所在 arl 文件名。


注意

控制柜断电时会保存 arl 文件名，当控制柜重启后，程序名维持不变；当前台通道运行后，程序名会随着程序的运行被更新。

参数

表 5-123 curppfile 函数的参数

名称	说明
channel_index	类型：int
	为前台通道号，起始值为 1

返回值

类型：string

用法举例

string s = curppfile(1) //读取前台通道 1 的程序指针所在的 arl 文件名。

5.13.11curppline(获取程序指针行号)

函数原型

int curppline(int channel_index)

描述

用于获取程序指针行号。


注意

控制柜断电时会保存指针行号，当控制柜重启后，指针行号维持不变；当前台通道运行后，指针行号会随着程序的运行被更新。

参数

表 5-124 curppline 函数的参数

名称	说明
channel_index	类型：int
	为前台通道号，起始值为 1

返回值

类型：int

用法举例

int i = curppline(1) //读取前台通道 1 的程序指针行号。

5.13.12filesize(文件大小查询)

函数原型

int filesize(string file_name)

描述

用于查询文件大小。

参数

表 5-125 curppline 函数的参数

名称	说明
file_name	类型：string
	被查询的文件名称

返回值

类型：int

用法举例

int i = filesize(prog1.arl) //读取 script/prog1.arl 文件的大小，单位是字节。

5.13.13renamefile(文件重命名)

函数原型

bool renamefile(string file_name, string new_file_name)

描述

用于将文件重新命名。

参数

表 5-126 curppline 函数的参数

名称	说明
file_name	类型：string
	原文件名称
new_file_name	类型：string
	新文件名称

返回值

类型：bool

返回 true 表示操作成功，false 表示操作失败。

用法举例

bool result = renamefile(prog1.arl, prog2.arl) //将 script/prog1.arl 文件的名称改为 prog2.arl。

5.13.14removefile(删除文件)

函数原型

bool removefile(string file_name)

描述

用于删除文件。

参数

表 5-127 curppline 函数的参数

名称	说明
file_name	类型：string

名称	说明
	被删除的文件名称

返回值

类型：bool

返回 true 表示操作成功，false 表示操作失败。

用法举例

bool result = removefile(prog2.arl) //删除名称为 script/prog2.arl 的文件。

6 中断

当在复杂的应用中使用机器人时，需要机器人可以立即对某些外部或内部事件作出反应，必须中断正在运行的机器人程序，启动中断处理函数。中断处理函数执行完成后，再回到被中断的程序中继续执行。

6.1 中断声明

格式

interrupt [name:],[priority:],when:,do:

参数

表 6-1 interrupt 的参数说明

名称	说明
name	数据类型：string
	指定中断名，该名字如果指定则不允许为空，并且不能和其他声明过的中断重名 之后用户可以通过该中断名使能或者屏蔽该中断 该参数可以缺省，缺省值为空字符串
priority	数据类型：int
	指定中断优先级 中断优先级必须为范围在 0~255 的整数 该参数可缺省，缺省值为 0 关于中断优先级更详细的信息见 第 6.2 章节
when	数据类型：bool
	定义中断事件 该中断事件为一个 bool 型表达式，只要该 bool 型表达式的值为 true，则中断事件就执行，表示一个中断事件发生 关于中断事件更详细的信息见 第 6.3 章节
do	数据类型：any
	定义中断处理动作 当中断事件发生时，则会执行该表达式动作 关于中断处理动作更详细的信息见 第 6.4 章节

6.2 中断优先级

中断优先级为一个 0~255 的整数，0 的优先级最高，255 优先级最低。

中断优先级表示当在同一个中断扫描周期同时有 2 个事件发生时，则优先响应优先级高的那一个，也就是说先执行高优先级中断的中断处理动作，执行完毕后，再执行低优先级中断的中断处理动作，最后再回到被中断函数继续执行。如果 2 个事件的优先级相同，则按声明的先后顺序依次响应。

相关示例程序如下：

```
func void inthandler1()
```

```
print "-->inthandler1"
endfunc
func void inthandler2()
print "-->inthandler2"
endfunc
func void main()
//声明中断，中断优先级为 1
interrupt 1,when:getdi(5),do:inthandler1()
//声明中断，中断优先级为 0
interrupt 0,when:getdi(6),do:inthandler2()
while(1)
waittime 1
endwhile
endfunc
```

假设某一时刻 DI 板的通道 5 与通道 6 同时有信号产生时，则系统会优先响应中断优先级为 0 的中断，再响应中断优先级为 1 的中断。

所以最终的输出结果为：

```
-->inthandler2
-->inthandler1
```

中断允许嵌套。在系统执行中断处理函数的过程中，如果发生了更高优先级的中断事件，则系统会暂停当前中断函数的执行，进入新的中断处理函数执行，执行完毕后，再回到上一级中断处理函数中继续执行。

相关示例程序如下：

```
func void inthandler1()
print "-->inthandler1"
endfunc
func void inthandler2()
print "-->inthandler2"
movej j:{j2 -20}
movej j:{j2 -30}
waittime 0
endfunc
func void main()
//声明中断，中断优先级为 0
interrupt 0,when:getdi(5),do:inthandler1()
//声明中断，中断优先级为 1
interrupt 1,when:getdi(6),do:inthandler2()
while(1)
waittime 1
endwhile
endfunc
```

假设在 t1 时刻 DI 的第 6 通道产生信号，则程序进入到 inthandler2 中执行，执行过程中，如果 DI 的第 5 通道又产生了信号，则程序会直接进入到 inthandler1 中执行，执行完毕后再回到 inthandler2 中接着被中断的程序继续执行。

6.3 中断事件

中断事件必须为一个 bool 型或者能隐式转换为 bool 型的表达式。该表达式与 if，while 语句中的判断语句相同。

举例来说，以下表达式都属于这种类型的表达式：

- 1
- true
- getdi(5)
- getdi(5) == true
- getdi(5) && getdi(6)
- counter>=20
- T(3.4)

有中断事件发生的定义为：

在上一个中断扫描周期中，中断事件表达式的值为 false，在本中断扫描周期中，中断事件表达式的值为 true。也就是说 ARL 中中断为边沿触发，只有当中断事件表达式的值从 false 变为 true 时，才表示该中断事件发生。

6.4 中断处理动作

中断处理动作为一个表达式。当中断事件发生时，如果满足执行条件，该表达式会被执行。

interrupt 声明指令中的 do 参数表达式一般为调用一个函数，该函数既可以是用户定义的中断处理函数，也可以是系统预定义函数。

如果中断处理动作比较简单，例如只是输出 1 路或多路 DO，则可以像如下形式声明中断：

interrupt 1,when:getdi(6),do:setdo(1,true) //声明一个优先级为 1 的中断，当 DI 的第 6 通道有信号时，则将 DO 的第一个通道信号置为 true。

如果中断处理动作要执行比较复杂的事情，则只能定义中断处理函数。中断处理函数与普通函数的定义没有区别，只是一般的中断处理函数都没有参数和返回值。

6.5 中断使能，屏蔽与删除

当一个中断声明过后该中断默认就已经使能。

如果希望在某些情况下屏蔽该中断，可以使用 disableint 指令。如果希望再次使能该中断，可以使用 enableint 指令。如果希望删除某个声明的中断，可以使用 delint 指令。

disableint，enableint 和 delint 指令用法如下：

格式

disableint/enableint/delint [name:],[priority:]

参数

disableint/enableint/delint 指令单独使用时，即不带任何参数时表示屏蔽/使能/删除所有中断。

表 6-2 disableint/enableint/delint 的参数说明

名称	说明
name	数据类型：string
	指定要屏蔽/使能/删除的中断的中断名 如果希望通过中断名来屏蔽/使能/删除某个中断。该中断在声明时必须执行 name 参数为该中断取个合适的名字 如果这里指定的中断名的中断并不存在，则会产生一个运行时错误
priority	数据类型：int
	指定屏蔽/使能/删除某个中断优先级的中断 disableint 指令允许屏蔽/使能/删除某个中断优先级的中断 如果这里指定的中断优先级的中断并不存在，则会产生一个运行时错误

如果 name 和 priority 参数同时指定，则表示既使能名字为 name 的中断，也使能优先级为 prioriy 的所有中断。

中断屏蔽与使能的用法举例如下：

```
func void inthandler1
disalbeint//屏蔽任何中断，即执行该中断函数期间不响应任何其他中断
……
enableint//使能全部中断
endfunc
func void main
//声明中断，中断优先级为 0
interrupt 0,when:getdi(5),do:inthandler1()
while(1)
waittime 1
endwhile
endfunc
```

6.6 在中断处理函数中停止当前机器人动作

默认情况下，当发生中断事件时，主函数中提前规划的轨迹仍然会运行完之后才会执行中断处理函数中的运动。

如果希望中断事件发生时即停止当前的运动，则需要用到 stopmove 指令。

退出中断处理函数时如果希望继续被中断的运动，则需要用到 startmove 指令。

stopmove 和 startmove 指令的用法举例如下：

```
func void inthandler1()
stopmove fast //快速停止当前运动
waituntil getdi(6) //等待第 6 通道 DI 信号
startmove skip:1 //跳过停止的程序段，并重新启动当前运动
```

```
endfunc
func void main()
//声明中断，中断优先级为 0
interrupt 0,when:getdi(5),do:inthandler1()
pose p = {x 1500,y 500,z 500,a 0,b 90,c 0,cfg 0}
ptp p,v:{per 10}
while(true)
lin p:{x 1000,y 500,z 500,a 0,b 90,c 0},v:{tcp 10}
lin p:{x 1100,y 500,z 500,a 0,b 90,c 0},v:{tcp 10}
endwhile
endfunc
```


提示

stopmove 会沿编程所设定的路径减速停止，需要一定的停止时间和停止距离，速度较快时甚至可能停止到下一条轨迹上，编程时需要注意。

6.7 定时中断

描述

定时中断指令是一种特殊的中断声明。它以时钟作为中断源，可以应用于需要实现一段时间之后触发一次中断，或者每隔一段时间就触发一次中断的场合。

格式

timer [name:],[priority:],interval:,[rmode:],do:

参数

表 6-3 timer 的参数说明

名称	说明
name	数据类型：string
	指定中断名，该名字如果指定则不允许为空，并且不能和其他声明过的中断重名 之后用户可以通过该中断名使能或者屏蔽该中断。该参数可以缺省，缺省值为空字符串
priority	数据类型：int
	指定中断优先级。中断优先级必须为范围在 0~255 的整数，该参数可缺省，缺省值为 0 关于中断优先级更详细的信息见 第 6.2 章节
interval	数据类型：double
	定义触发中断的时间间隔，单位：秒 当 rmode 参数值为 true 时，interval 的值最小可以指定为 0.001，当 rmode 参数值为 false 时，interval 的值最小可以指定为 0
rmode	数据类型：bool
	定义 repeat 模式 ■ false 表示只在时间间隔 interval 后触发一次中断

名称	说明
	■ true 表示每隔时间间隔 interval 后反复触发中断
do	数据类型：any
	定义中断处理动作 当中断事件发生时，则会执行该表达式动作

用法举例

timer name:"t1",priority:1,interval:1,rmode:true,do:setdo(1,true) //每隔 1s 执行一次 setdo(1,true)

7 轨迹触发

在某些情况下，用户希望在一条运动轨迹的中间某个点触发一些事件而又不希望破坏这条运动轨迹的动态特性。这时可以使用轨迹触发指令。ARL 轨迹触发指令可以实现在一条运动轨迹的多个时间点或者多个距离点触发多个触发函数来达到用户希望的行为。

7.1 轨迹触发声明

ARL 中轨迹触发实际是一种特殊形式的中断，所以轨迹触发的声明与中断声明指令格式差不多，唯一的区别是轨迹触发声明中没有名字参数。

指令格式

trigger [priority:],when:,do:

参数

轨迹触发声明参数详见表 7-1。

表 7-1 轨迹触发声明参数

名称	说明
priority	数据类型：int
	指定轨迹触发优先级 该优先级的定义同中断优先级，并且当一个触发事件和一个中断事件在一个中断扫描周期同时发生时也会优先响应具有较高优先级的中断或者触发事件 该参数可缺省，缺省值为 10
when	数据类型：bool
	定义触发事件 触发事件与中断事件的定义相同。只是因为轨迹触发是在轨迹运动过程中扫描的中断，所以轨迹触发事件一般都与轨迹信息相关，如轨迹的运行时间和运行距离 关于轨迹触发事件更详细的信息见 第 6.3 章节
do	数据类型：any
	定义轨迹触发动作 当轨迹触发事件发生时，则会执行该表达式动作。轨迹触发动作的定义与中断触发动作的定义相同

声明的轨迹触发将作用在该触发声明语句下面最近的一条运动指令。

例如：

```
trigger 1,when:T(1),do:setdo(2,true)//该声明作用于下一条 movej 指令
movej j:{j1 30},v:{per 2}
ptp p0,v:{per 50}
trigger 1,when:S(100),do:setdo(3,true) //该声明作用于下一条 lin 指令
lin p1,v:{tcp 20,ori 10}
```

7.2 轨迹触发事件

一般以 4 个系统预定义函数来定义触发事件：

- 轨迹经过时间（T）
- 轨迹经过距离（S）
- 轨迹剩余距离（StoEnd）
- 轨迹经过百分比（P）

例如：

trigger 0,when:T(0.2),do:setdo(2,true)

该声明的含义是在下一条运动语句的轨迹从轨迹起点出发 0.2s 时将 DO 的第 2 路信号置为 true。

例如：

trigger 0,when:StoEnd(10),do:setdo(2,true)

该声明的含义是在下一条运动语句的轨迹运动到距离目标点 10 毫米时将 DO 的第 2 路信号置为 true。

更复杂的轨迹触发事件可以通过系统变量$TRAJ_ ELAPSE _TIME，$TRAJ_LEFT_TIME，$TRAJ_ ELAPSE _DIS 和$TRAJ_LEFT_DIS 实现。

例如：

trigger 0,when: P(50),do:setdo(2,true)

该声明的含义是在下一条运动语句的轨迹运动到轨迹长度 50%处时将 DO 的第 2 路信号置为 true。

7.3 使用轨迹触发注意事项

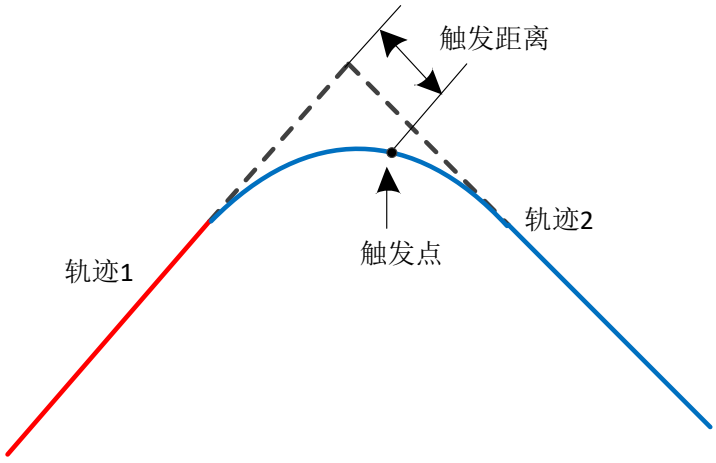


图 7-1 轨迹触发

使用轨迹触发注意事项主要有以下几点：

- PTP 和 MOVEJ 指令因为不关心 TCP 点轨迹，所以对于 PTP 和 MOVEJ 指令如果使用 S 和 StoEnd 函数指定轨迹触发事件，触发函数将不会被触发。
- 由于插补精度以及信号输出速度的问题，指定的触发时间或者触发距离与实际的触发事件或者触发距离可能会有几个毫秒或者几个毫米的误差。
- 当指定的时间或者距离参数不在 0 至轨迹总时间或总距离范围内时，声明的轨迹触发将不会被触发。

- 在指定了平滑行为的情况下，平滑部分的轨迹将被认为属于后一条轨迹，所以如果希望在平滑轨迹过程中触发事件，需要在第 2 条轨迹前声明触发事件。轨迹路程仍会按照前一条轨迹未平滑时的目标点作为起点计算，如下图 7-1 所示。
- T()函数的触发针对的是倍率为 100%。当倍率非 100%时，触发点的绝对位置不会改变，因此触发时间与指定的实际时间不同。

7.4 并行处理

描述

执行运动指令的过程中，并行处理一些动作，功能上与轨迹触发一致。并列处理格式无法指定优先级（默认为 10），其余功能与轨迹触发完全一致。

指令格式

运动指令;触发事件,触发动作

运动指令;触发事件 1,触发动作 1;...;触发事件 i,触发动作 i;...;触发事件 n,触发动作 n

参数

名称	说明
触发事件	定义触发事件 触发事件与中断事件的定义相同。只是因为轨迹触发是在轨迹运动过程中扫描的中断，所以轨迹触发事件一般都与轨迹信息相关，如轨迹的运行时间和运行距离关于轨迹触发事件更详细的信息见 第 7.3 章节
触发动作	定义轨迹触发动作 当轨迹触发事件发生时，则会执行该表达式动作。轨迹触发动作的定义与中断触发动作的定义相同

使用举例

举例如下：

```
movej j:{j1 30},v:{per 2};T(1)，setdo(2,true)
//功能完全等同于以下 2 条指令
trigger 10,when:T(1),do:setdo(2,true)
movej j:{j1 30},v:{per 2}


lin p1,v:{tcp 20,ori 10};T(1)，setdo(2,true)；S(100),setdo(3,true)
//功能完全等同于以下 3 条指令
trigger 10,when:T(1),do:setdo(2,true)
trigger 10,when:S(100),do:setdo(3,true)
lin p1,v:{tcp 20,ori 10}
```

8 模块化编程

ARL 支持模块化编程，即可以在一个 arl 文件中调用另一个 arl 文件中定义的函数或者访问另一个 arl 文件中定义的全局变量。

该功能通过“::”作用域操作符实现。

例如：

a.arl 文件

```
double gc = 1
func double add(double x,double y)
return x+y
endfunc
```

该文件中定义了一个全局变量 gc 和一个函数 add。

如果想在 b.arl 文件中调用 add 函数或者引用变量 gc，可以使用如下方式：

b.arl 文件

```
func void main()
print a::add(1,2) + a::gc
endfunc
```

运行程序后，将输出 4。

以上程序假设 a.arl 文件与 b.arl 文件在同一目录下。

如果 a.arl 文件与 b.arl 文件不在同一个目录下，则需要借助 import 指令：

b.arl 文件

```
import “/XXX/XXX/a.arl”
func void main()
print a::add(1,2) + a::gc
endfunc
```

其中，/XXX/XXX/a.arl 指 a.arl 文件的绝对路径。


注意

- 加载一个 arl 文件时，系统自动导入一个名字为该 arl 文件+_data.arl 模块（如果该文件存在的话），例如对于 b.arl 文件，如果同目录下存在一个叫 b_data.arl 的文件，则系统会自动将它导入，此时引用 b_data.arl 模块中的全局变量时则可以缺省“模块名::”，而直接通过变量名引用。
- 模块名即为不带.arl 的文件名。
- 被调用函数所在的文件名命名必须符合变量命名规则，且被调用 arl 文件名不允许为纯数字或中文。
- 引用其他模块的函数或者变量时，必须通过“模块名::”的方式。

9 外部自动控制

可通过外部自动控制功能实现通过其他控制器（如机床 PLC）控制机器人系统。

使用外部自动控制功能需要完成以下工作：

- 外部自动控制功能配置。
- 根据配置连接外部控制器和机器人控制系统端口。
- 编写外部自动控制主程序并测试运行。

9.1 外部自动控制功能配置

由于外部自动控制功能使用用户 IO 端口实现和外部控制器之间的交互，所以首先必须对各种外部自动控制功能配置相应的端口号。

这里包括对输入端信号以及输出端信号的配置。这里的输入输出端是从机器人控制系统的角度来说。

9.1.1 输入端信号配置

$EXT_CTL_ACT_DI

- 变量类型：int
- 变量功能：定义外部自动控制激活信号 DI 端口号。

如果将该系统变量定义为[1~40*系统接入的 MF 个数]之间的一个整数时，则外部控制器必须通过将该端口信号置 true 来激活机器人控制系统的外部自动控制功能；定义为 0 时，表示不使用该激活功能，即机器人控制系统默认激活外部自动控制功能。

$SERVO_ON_DI

- 变量类型：int
- 变量功能：定义伺服上电信号 DI 端口号。

如果将该系统变量定义为一个有效端口号整数时，在此输入端口上给入脉宽至少 30ms 的高脉冲，则机器人控制系统会进行伺服上电动作；定义为 0 时表示不启用该功能。

$SERVO_OFF_DI

- 变量类型：int
- 变量功能：定义伺服断电信号 DI 端口号。

如果将该系统变量定义为一个有效端口号整数时，在此输入端口上给入脉宽至少 30ms 的高脉冲，则机器人控制系统会进行伺服断电动作；定义为 0 时表示不启用该功能。

$START_PROG_DI

- 变量类型：int
- 变量功能：定义启动程序信号 DI 端口号。

如果将该系统变量定义为一个有效端口号整数时，在此输入端口上给入一个上升沿信号，则机器人控制系统会启动通道中加载的程序的执行（这个程序通常为用户编写的外部自动控制主程序）；定义为 0 时表示不启用该功能。

$PAUSE_PROG_DI

- 变量类型：int
- 变量功能：定义暂停程序信号 DI 端口号。

如果将该系统变量定义为一个有效端口号整数时，在此输入端口上给入一个上升沿信号，则机器人控制系统会暂停通道中加载的程序的执行（这个程序通常为用户编写的外部自动控制主程序）；定义为 0 时表示不启用该功能。

$RESET_PROG_DI

- 变量类型：int
- 变量功能：定义复位程序信号 DI 端口号。

如果将该系统变量定义为一个有效端口号整数时，在此输入端口上给入 true 信号，则机器人控制系统会复位通道中加载的程序；定义为 0 时表示不启用该功能。注意，如果使用该信号，如要启动程序时，需要先给该 DI 信号输入 false 信号，否则程序将不能够启动。

$CLEAR_ALARM_DI

- 变量类型：int
- 变量功能：定义清除报警信号 DI 端口号。

如果将该系统变量定义为一个有效端口号整数时，在此输入端口上给入一个上升沿信号，则机器人控制系统会清除当前报警；定义为 0 时表示不启用该功能。

$PGNO_TYPE

- 变量类型：int
- 变量功能：定义程序号格式。取值范围为 0，1，2。含义如下表 9-1 所示。

表 9-1 $PGNO_TYPE 变量取值范围含义

值	说明	示例
0	以二进制格式读取	0 0 0 0 0 1 1 1 $PGNO = 7
1	以 BCD 编码格式读取，每 4 个二进制位表示一个 BCD 码。该格式下每 4 个二进制位对应的十进制数不能大于 9	0 0 0 1 0 1 0 0 $PGNO = 14
2	以“N 选 1”格式读取。该格式下 N 位长度中必须只能有 1 位是 1，其他位都是 0	0 0 0 0 0 0 0 1 $PGNO = 0 0 0 0 1 0 0 0 0 $PGNO = 4

$PGNO_LENGTH

- 变量类型：int
- 变量功能：定义程序号位宽

取值范围为 1~16。当$PGNO_TYPE 值为 1，也就是 BCD 编码格式读取程序号时，$PGNO_LENGTH 取值只能为 4、8、12、16。

$PGNO_FBIT_DI

- 变量类型：int
- 变量功能：定义程序号第一位所在的 DI 端口号。

例如当此变量设置为 21，并且程序号位宽设置为 4 时，则程序号由第 21、22、23、24 这 4 路 DI 信号决定。

$PGNO_PARITY_DI

- 变量类型：int
- 变量功能：定义奇偶校验信号 DI 端口号。

该变量值可以为正数，也可以为负数。该变量值的绝对值大小表示所用的端口号，负值表示使用奇校验，正值表示使用偶校验。其他无效值表示不对程序号进行奇偶校验。当$PGNO_TYPE 值为 2，也就是“N 选 1”格式读取程序号时，不管$PGNO_PARITY_DI 值为何值都不进行奇偶校验。

$PGNO_VALID_DI

- 变量类型：int
- 变量功能：定义程序号有效信号 DI 端口号。

外部控制器给入该信号时，机器人控制器开始读入程序号。该变量值为正数，值的大小表示所用的端口号，且信号上升沿有效。

$EXT_CTL_CHAN_DI

- 变量类型：int
- 变量功能：外部控制通道选择起始位 DI 端口号

当机器人控制系统配置了多个前台通道时，可通过该功能选择外部控制功能生效的通道号。此参数为外部自动控制通道选择 DI 的第一个逻辑地址，当该参数定义为有效 DI 逻辑地址时，以二进制格式读取连续三路 DI 状态，序号范围为：0~5，对应第 1~6 个前台通道。

举例

若设置$EXT_CTL_CHAN_DI = 1 时：

- 当 1~3 路 DI 的电平状态分别为 000（对应序号 1）时，则前台第 1 通道的外部控制功能生效。
- 当 1~3 路 DI 的电平状态分别为 011（对应序号 3）时，则前台第 4 通道的外部控制功能生效。

9.1.2 输出端信号配置

$EXT_CTL_ACT_CONF_DO

- 变量类型：int
- 变量功能：定义外部自动控制功能激活确认信号 DO 端口号。

当外部自动控制被激活时，该信号输出 true，否则该信号输出 false。激活外部自动控制功能需要进行如下操作：

通过 HMI 的设置页面使能外部自动控制功能；外部控制器给入激活信号；将该变量设置为一个不存在的通道号时表示不启用该功能。

$SERVO_ON_DO

- 变量类型：int
- 变量功能：定义伺服上电信号 DO 端口号。

当机器人控制系统处于伺服上电状态时，该信号输出 true，否则该信号输出 false。将该变量设置为一个不存在的通道号时表示不启用该功能。

$PGNO_REQ_DO

- 变量类型：int
- 变量功能：定义请求程序号信号 DO 端口号。

当用户将系统变量$PGNO_REQ 置为 true 时，系统会将$PGNO_REQ_DO 定义的 DO 信号置 true 表示向外部控制器请求程序号。将该变量设置为一个不存在的通道号时表示不启用该功能。

$AT_HOME_DO

- 变量类型：int[5]
- 变量功能：定义处于 HOME 点信号 DO 端口号。

系统可设置 5 个 home 点，每个 HOME 点代表位置可在实时位置界面修改。该 DO 信号表明机器人当前是否处于某个 home 点的位置。

$AT_T1_DO

- 变量类型：int
- 变量功能：定义系统处于手动低速模式信号 DO 端口号。

当系统处于手动低速模式时，该信号输出 true，否则该信号输出 false。将该变量设置为一个不存在的通道号时表示不启用该功能。

$AT_T2_DO

- 变量类型：int
- 变量功能：定义系统处于手动高速模式信号 DO 端口号。

当系统处于手动高速模式时，该信号输出 true，否则该信号输出 false。将该变量设置为一个不存在的通道号时表示不启用该功能。

$AT_AUT_DO

- 变量类型：int
- 变量功能：定义系统处于自动模式信号 DO 端口号。

当系统处于自动模式时，该信号输出 true，否则该信号输出 false。将该变量设置为一个不存在的通道号时表示不启用该功能。

$PGNO_ACK_FBIT_DO

- 变量类型：int
- 变量功能：定义程序号确认信号第一位所在的 DO 端口号。

当机器人系统收到外部通过 DI 信号给入的程序号后，如果该程序号合法，则会通过此系统变量定义的 DO 信号组将程序号输出给外部控制器，外部控制器可以通过这个程序号反馈来校验机器人控制器是否收到程序号。将该变量设置为一个不存在的通道号时表示不启用该功能。

$CHAN_STATE_DO

- 变量类型：int
- 变量功能：当前通道状态起始逻辑地址号。

如值为 5 时，表示使用 5、6 地址号作为通道状态输出。00:未加载；10:运行；01:暂停；11:停止。将该变量设置为一个不存在的通道号时表示不启用该功能。

10 系统变量

ARL 中变量分为 3 种类型（参见第 2 章节）：

- 局部变量
- 全局变量
- 系统变量

系统变量为系统预定义的有特殊含义的变量。用户在使用时不需要声明，可以直接使用。使用系统变量和使用用户自定义变量没有区别。

每个系统变量都有其自己的变量类型，默认值和含义。有些系统变量还有最大值和最小值的限制。用户只能向这些系统变量写入最大值和最小值之间的值，如果试图向这些系统变量赋超限值时，系统会报错。

以下说明可以在用户程序中引用的系统变量。

10.1 数据类型系统变量

10.1.1 $I(整型系统变量)

表 10-1 $I 变量

属性	说明
变量名	整型系统变量
数据类型	int 数组[100]
最大值	整型数据的最大值
最小值	整型数据的最小值
默认值	出厂默认值为全 0
何时恢复默认值	用户在程序中给数组的某些元素赋予新值后，可以通过 savesv(“I”)函数将数组所有元素的值存储到配置文件中，使得数据掉电不丢失。如果不保存，则下次启动系统时数据恢复为上一次保存的值
功能描述	系统预定义的整型变量数组，其含义用户可以自定义
值含义	用户自定义

10.1.2 $I_NAME(整形系统变量名称)

表 10-2 $I_NAME 变量

属性	说明
变量名	整形系统变量名称
数据类型	string 数组[100]
最大值	\
最小值	\
默认值	

属性	说明
何时恢复默认值	该变量改动立即生效，掉电重启后仍为上次修改值
功能描述	系统预定义的整形变量名称数组，其含义由用户自定义
值含义	用户自定义

10.1.3 $B(布尔型系统变量)

表 10-3 $B 变量

属性	说明
变量名	布尔型系统变量
数据类型	bool 数组[100]
最大值	-
最小值	-
默认值	出厂默认值为全 false
何时恢复默认值	用户在程序中给数组的某些元素赋予新值后，可以通过 savesv(“B”)函数将数组所有元素的值存储到配置文件中，使得数据掉电不丢失。如果不保存，则下次启动系统时数据恢复为上一次保存的值
功能描述	系统预定义的布尔型变量数组，其含义用户可以自定义
值含义	用户自定义

10.1.4 $B_NAME(布尔形系统变量名称)

表 10-4 $B_NAME 变量

属性	说明
变量名	布尔形系统变量名称
数据类型	string 数组[100]
最大值	\
最小值	\
默认值	
何时恢复默认值	该变量改动立即生效，掉电重启后仍为上次修改值
功能描述	系统预定义的布尔形变量名称数组，其含义由用户自定义
值含义	用户自定义

10.1.5 $D(浮点型系统变量)

表 10-5 $D 变量

属性	说明
变量名	浮点型系统变量
数据类型	double 数组[100]

属性	说明
最大值	浮点型数据的最大值
最小值	浮点型数据的最小值
默认值	出厂默认值为全 0
何时恢复默认值	用户在程序中给数组的某些元素赋予新值后，可以通过 savesv(“D”)函数将数组所有元素的值存储到配置文件中，使得数据掉电不丢失。如果不保存，则下次启动系统时数据恢复为上一次保存的值
功能描述	系统预定义的浮点型变量数组，其含义用户可以自定义
值含义	用户自定义

10.1.6 $D_NAME(浮点形系统变量名称)

表 10-6 $D_NAME 变量

属性	说明
变量名	浮点形系统变量名称
数据类型	string 数组[100]
最大值	\
最小值	\
默认值	
何时恢复默认值	该变量改动立即生效，掉电重启后仍为上次修改值
功能描述	系统预定义的浮点形变量名称数组，其含义由用户自定义
值含义	用户自定义

10.1.7 $P(结构体类型系统变量 P)

表 10-7 $P 变量

属性	说明
变量名	结构体类型系统变量 P
数据类型	pose 数组[100]
最大值	-
最小值	-
默认值	出厂默认值为全 false
何时恢复默认值	用户在程序中给数组的某些元素赋予新值后，可以通过 savesv(“P”)函数将数组所有元素的值存储到配置文件中，使得数据掉电不丢失。如果不保存，则下次启动系统时数据恢复为上一次保存的值
功能描述	系统预定义的结构体类型变量数组，其含义用户可以自定义
值含义	用户自定义

10.1.8 $J(结构体类型系统变量 J)

表 10-8 $J 变量

属性	说明
变量名	结构体类型系统变量 J
数据类型	joint 数组[100]
最大值	-
最小值	-
默认值	出厂默认值为全 false
何时恢复默认值	用户在程序中给数组的某些元素赋予新值后，可以通过 savesv(“J”)函数将数组所有元素的值存储到配置文件中，使得数据掉电不丢失。如果不保存，则下次启动系统时数据恢复为上一次保存的值
功能描述	系统预定义的结构体类型变量数组，其含义用户可以自定义
值含义	用户自定义

10.1.9 $TOOLS(工具坐标系)

表 10-9 $TOOLS 变量

属性	说明
变量名	工具坐标系
数据类型	tool 结构体数组[32]
最大值	-
最小值	-
默认值	-
何时恢复默认值	该系统变量通过 HMI 修改将会永久保存，如果通过程序修改则在系统重启后将恢复为之前保存的值
功能描述	存储用户自定义工具坐标系
值含义	用户自定义工具坐标系

10.1.10$TOOLS_NAME(工具坐标系名称)

表 10-10 $TOOLS_NAME 变量

属性	说明
变量名	工具坐标系名称
数据类型	tool 结构体数组[32]
最大值	-
最小值	-
默认值	-
何时恢复默认值	该变量改动立即生效，掉电重启后仍为上次修改值
功能描述	存储用户自定义工具坐标系

属性	说明
值含义	用户自定义工具坐标系

10.1.11$WOBJS(工件坐标系)

表 10-11 $WOBJS 变量

属性	说明
变量名	工件坐标系
数据类型	wobj 结构体数组[32]
最大值	-
最小值	-
默认值	-
何时恢复默认值	该系统变量通过 HMI 修改将会永久保存，如果通过程序修改则在系统重启后将恢复为之前保存的值
功能描述	存储用户自定义工件坐标系
值含义	用户自定义工件坐标系

10.1.12$WOBJS_NAME(工件坐标系名称)

表 10-12 $WOBJS_NAME 变量

属性	说明
变量名	工件坐标系名称
数据类型	wobj 结构体数组[32]
最大值	-
最小值	-
默认值	-
何时恢复默认值	该变量改动立即生效，掉电重启后仍为上次修改值
功能描述	存储用户自定义工件坐标系
值含义	用户自定义工件坐标系

10.1.13$BASE(基础坐标系)

表 10-13 $BASE[]变量

属性	说明
变量名	基础坐标系
数据类型	wobj
取值范围	0,1,2
最大值	-

属性	说明
最小值	-
默认值	{{0,0,0,0,0,0},false} {{0,0,0,0,0,0},false} {{0,0,0,0,0,0},false}
何时恢复默认值	该系统变量通过 HMI 修改将会永久保存，如果通过程序修改则在系统重启后将恢复为之前保存的值
功能描述	定义在机器人足底的基础坐标系
值含义	基础坐标系，相对世界坐标系定义

10.1.14$FLANGE(法兰坐标系)

表 10-14 $FLANGE 常量

属性	说明
变量名	法兰坐标系
数据类型	tool
值	{{0,0,0,0,0,0,},false,}
何时恢复默认值	-
功能描述	存储系统预定义的特殊工具坐标系：法兰坐标系
值含义	系统预定义法兰坐标系

10.1.15$WORLD(世界坐标系)

表 10-15 $WORLD 常量

属性	说明
变量名	世界坐标系
数据类型	wobj
值	{{0,0,0,0,0,0,},false, WORLD}
何时恢复默认值	-
功能描述	存储系统预定义的特殊工件坐标系：世界坐标系
值含义	系统预定义世界坐标系

10.2 功能类型系统变量

10.2.1 $WRIST(开启腕部奇异点避让)

表 10-16 $WRIST 变量

属性	说明
变量名	布尔型系统变量
数据类型	bool

属性	说明
最大值	\
最小值	\
默认值	出厂默认值为 false
何时恢复默认值	软件启动时、程序复位时
功能描述	机器人轨迹理论上不能穿越奇异点，在接近奇异点过程中会报警超速。其中，腕部奇异点是指机器人 5 轴为 0 的位置。当需要穿越 5 轴为 0 的位置进行 lin、cir 等笛卡尔轨迹运动时，可将系统变量设置为 true，开启腕部奇异点避让功能。此时，机器人会部分牺牲姿态精度，保证 TCP 精度，从而穿过奇异点。完成穿越后，将系统变量设置为 false，可继续进行普通运动。
值含义	■ true：开启 ■ false：关闭

10.2.2 $Config_check (轴配置检查使能)

表 10-17 $ config_check 变量

属性	说明
变量名	轴配置检查使能
数据类型	bool
默认值	true
何时恢复默认值	软件启动时，程序复位时
功能描述	机器人的 TCP 到达同一个空间中的点位，对应的轴位置可能是不同的，可参考 2.4.4 节 pose 结构体中 turn 值的说明。如果在示教时，两个点位的轴位置不同（比如出现了六轴差了 1 圈的情况），实际运行时需要用该参数决定是否要运动到示教时所获取的轴位置。
值含义	■ true：按照示教时轴配置运动 ■ false：按照距当前轴位置最近的轴配置运动

10.2.3 $DFSPEED(默认速度参数)

表 10-18 $DFSPEED 变量

属性	说明
变量名	默认速度参数
数据类型	speed
最大值	-
最小值	-
默认值	{5,50,400,5,5}
何时恢复默认值	软件启动时、程序复位时
功能描述	当指令中未指定速度参数 v 时则使用此默认参数。
值含义	参见 speed 类型定义

10.2.4 $DFSLIP(默认平滑参数)

表 10-19 $DFSLIP 变量

属性	说明
变量名	默认平滑参数
数据类型	slip
最大值	-
最小值	-
默认值	{-1.0,-1.0}
何时恢复默认值	软件启动时、程序复位时
功能描述	当指令中未指定平滑参数 s 时则使用此默认参数
值含义	参见 slip 类型定义

10.2.5 $DFTOOL(默认工具参数)

表 10-20 $DFTOOL 变量

属性	说明
变量名	默认工具参数
数据类型	tool
最大值	-
最小值	-
默认值	{{0,0,0,0,0,0},false}
何时恢复默认值	软件启动时、程序复位时
功能描述	当指令中未指定工具参数时则使用此默认参数
值含义	参见 tool 类型定义

10.2.6 $DFWOBJ(默认工件坐标系参数)

表 10-21 $DFWOBJ 变量

属性	说明
变量名	默认工件坐标系参数
数据类型	wobj
最大值	-
最小值	-
默认值	{{0,0,0,0,0,0},false}
何时恢复默认值	软件启动时、程序复位时
功能描述	当指令中未指定工件坐标系参数时则使用此默认参数

属性	说明
值含义	参见 wobj 类型定义

10.2.7 $IGNORE_ORI(方向忽略使能)

表 10-22 $IGNORE_ORI 变量

属性	说明
变量名	方向忽略使能
数据类型	bool
最大值	-
最小值	-
默认值	false
何时恢复默认值	软件启动时、程序复位时
功能描述	该值为 true 时，将忽略运动语句中的目标点的 A,B,C 方向参数，目标点方向将保持和起点一致
值含义	true：使能

10.2.8 $ORI_REF_PATH(圆弧方向参照路径坐标系)

表 10-23 $ORI_REF_PATH 变量

属性	说明
变量名	圆弧方向参照路径坐标系
数据类型	bool
最大值	-
最小值	-
默认值	false
何时恢复默认值	软件启动时、程序复位时
功能描述	设置圆弧轨迹方向是否参照圆弧路径坐标系
值含义	true：cir 语句的轨迹方向参照圆弧路径坐标系

10.2.9 $VEL_PROFILE(速度轮廓类型)

表 10-24 $VEL_PROFILE 变量

属性	说明
变量名	速度轮廓类型
数据类型	$VEL_PROFILE
最大值	-
最小值	-

属性	说明	
默认值	不同的机型（对应示教器中的机械单元型号）默认值不同	3A、6A、6L、10A 的默认值为 O_type
		其它机型默认值为 S_type
何时恢复默认值	软件启动时、程序复位时	
功能描述	设置运动轨迹的速度轮廓 每条运动指令都会以该参数中设置了速度轮廓类型来进行规划 ■ S 型速度规划优点为平稳无冲击，缺点为规划出来的运动时间会比较长 ■ T 型速度规划的优点为规划出来的运动时间比较短，缺点为对机械有冲击 ■ O 型速度规划优点为节拍快且机械冲击小	
值含义	■ S_type：S 型速度轮廓 ■ T_type：T 型速度轮廓 ■ O_type：O 型速度轮廓	

10.2.10$TRAJ_ELAPSE_TIME(轨迹经过时间)

表 10-25 $TRAJ_ELAPSE_TIME 变量

属性	说明
变量名	轨迹经过时间
数据类型	double
最大值	-
最小值	-
默认值	0
何时恢复默认值	程序复位时由系统恢复为默认值
功能描述	存储轨迹经过时间，一般用于轨迹触发声明中
值含义	轨迹经过时间，单位秒

10.2.11$TRAJ_LEFT_TIME(轨迹剩余时间)

表 10-26 $TRAJ_LEFT_TIME 变量

属性	说明
变量名	轨迹剩余时间
数据类型	double
最大值	-
最小值	-
默认值	0
何时恢复默认值	程序复位时由系统恢复为默认值
功能描述	存储轨迹剩余时间，一般用于轨迹触发声明中
值含义	轨迹剩余时间，单位秒

10.2.12$TRAJ_ELAPSE_DIS(轨迹经过路程)

表 10-27 $TRAJ_ELAPSE_DIS 变量

属性	说明
变量名	轨迹经过路程
数据类型	double
最大值	-
最小值	-
默认值	0
何时恢复默认值	程序复位时由系统恢复为默认值
功能描述	存储轨迹经过路程，一般用于轨迹触发声明中
值含义	轨迹经过路程，单位毫米

10.2.13$TRAJ_LEFT_DIS(轨迹剩余路程)

表 10-28 $TRAJ_LEFT_DIS 变量

属性	说明
变量名	轨迹剩余路程
数据类型	double
最大值	-
最小值	-
默认值	0
何时恢复默认值	程序复位时由系统恢复为默认值
功能描述	存储轨迹剩余路程，一般用于轨迹触发声明中
值含义	轨迹剩余路程，单位毫米

10.2.14$CJOINT(当前轴位置点)

表 10-29 $CJOINT 变量

属性	说明
变量名	当前轴位置点
数据类型	joint
最大值	-
最小值	-
默认值	-
何时恢复默认值	程序复位时由系统恢复为默认值
功能描述	该系统变量的值始终由系统赋值为上一条运动轨迹的目标点轴位置，所以可使用该系统变量实现增量编程

属性	说明
值含义	当前轴位置点

10.2.15$RPP_ENABLE(RPP 使能)

表 10-30 $RPP_ENABLE 变量

属性	说明
变量名	RPP 使能
数据类型	bool
最大值	-
最小值	-
默认值	false
何时恢复默认值	软件启动时、程序复位时
功能描述	RPP(return to path point)轨迹是指自动程序运行的第一条轨迹。 ■ 当 RPP 功能生效时，RPP 轨迹会退化为 movej 轨迹，且速度为手动 jog 的速度； ■ 当 RPP 功能失效时，RPP 轨迹按照客户设定的当条指令类型进行运动，但运行速度受手动运行限速的控制，由于机器人再 RPP 运动前的位置不能确定，因此可能出现无法到达的告警
值含义	■ true：使能 RPP 功能 ■ false：屏蔽 RPP 功能

10.2.16$AT_HOME(是否处于 HOME 位置)

表 10-31 $AT_HOME 变量

属性	说明
变量名	是否处于 HOME 位置
数据类型	bool 型数组[5]
最大值	-
最小值	-
默认值	{false,false,false,false,false}
何时恢复默认值	-
功能描述	系统可设置 5 个 home 点，每个 HOME 点代表位置可在实时位置界面修改。该系统变量表明机器人当前是否处于某个 home 点的位置。该系统变量对用户只读，由系统内部修改
值含义	■ true：机器人处于 HOME 位置 ■ false：机器人不处于 HOME 位置

10.2.17$EXT_CTL_ACT(外部自动控制被激活)

表 10-32 $EXT_CTL_ACT 变量

属性	说明
变量名	外部自动控制被激活
数据类型	bool
最大值	-
最小值	-
默认值	-
何时恢复默认值	-
功能描述	该系统变量表明当前外部自动控制是否被激活。该系统变量对用户只读，由系统内部修改
值含义	■ true：外部自动控制被激活 ■ false：外部自动控制未激活

10.2.18$PGNO_REQ(请求程序号状态)

表 10-33 $PGNO_REQ 变量

属性	说明
变量名	请求程序号状态
数据类型	bool
最大值	-
最小值	-
默认值	false
何时恢复默认值	软件启动时，程序复位时
功能描述	该系统变量表明当前系统处于请求程序号状态。该系统变量用于使能外部自动控制时，将该系统变量值置为 true 时，系统变量$PGNO_REQ_DO 定义的 DO 输出端口会输出 true，表明当前系统处于请求程序号状态
值含义	■ true：系统处于请求程序号状态 ■ false：系统不处于请求程序号状态

10.2.19$PGNO(从外部获取的程序号)

表 10-34 $PGNO 变量

属性	说明
变量名	从外部获取的程序号
数据类型	int
最大值	-
最小值	-
默认值	-1
何时恢复默认值	软件启动时

属性	说明
功能描述	当使能外部自动控制，外部控制系统通过 DI 端口给入程序号有效信号时，系统会从 DI 端口读入程序号，如果该程序号为有效的程序号，则系统会将该系统变量置为对应的程序号。用户可以在外部自动控制主程序中根据该系统变量的不同值来执行不同的子程序
值含义	从外部获取的有效程序号

10.2.20$PI(圆周率)

表 10-35 $PI 常量

属性	说明
变量名	圆周率
数据类型	double
值	3.1415926535897932
何时恢复默认值	-
功能描述	系统预定义圆周率常数
值含义	圆周率

10.2.21$CTL_MODE(当前控制模式)

表 10-36 $CTL_MODE 变量

属性	说明
变量名	当前控制模式
数据类型	controlmode
最大值	-
最小值	-
默认值	-
何时恢复默认值	-
功能描述	该系统变量表明当前系统处于何种控制模式。该系统变量对用户只读，由系统内部修改
值含义	■ T1：手动低速模式 ■ T2：手动高速模式 ■ AUT：自动模式

10.2.22$WOBJ_OFFSET(工件坐标系偏移)

表 10-37 $WOBJ_OFFSET 变量

属性	说明
变量名	工件坐标系偏移，此次的工件坐标系是运动指令参考的坐标系。可以为：世界坐标系、工件坐标系、基座标系、工具坐标系和法兰坐标系。

属性	说明
数据类型	double 数组[6]
最大值	-
最小值	-
默认值	{0,0,0,0,0,0}
何时恢复默认值	软件启动时、程序复位时
功能描述	通过修改此系统变量，可实现目标点位的批量偏移
值含义	用户自定义工件坐标系偏移值

使用示例一如下：

```
pose p1 = { x 400, y 0, z 800, a 0, b 0, c 00}

$WOBJ_OFFSET={50,50,50,0,0,0}

lin p:p1,vp:50%,sp:-1%,t:$FLANGE,w:$WORLD //此时，$WORLD = {{50,50,50,0,0,0},false,WORLD}

print cpose($FLANGE,$WORLD) //输出“{ 450, 50, 850, 0, 0, 0, 0, -1, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”
```

10.2.23$TOOL_OFFSET(工具坐标系偏移)

表 10-38 $TOOL_OFFSET 变量

属性	说明
变量名	工具坐标系偏移
数据类型	double 数组[6]
最大值	-
最小值	-
默认值	{0,0,0,0,0,0}
何时恢复默认值	软件启动时、程序复位时
功能描述	通过修改此系统变量，可实现目标点位的批量偏移
值含义	用户自定义工具坐标系偏移值

使用示例如下：

```
pose p1 = { x 400, y 0, z 800, a 0, b 0, c 00}

$TOOL_OFFSET={50,0,0,0,0,0}
```

lin p:p1,vp:50%,sp:-1%,t:$FLANGE,w:$WORLD //此时，$FLANGE = {{50,0,0,0,0,0},false}

print cpose($FLANGE,$WORLD) //输出“{ 350, 0, 800, 0, 0, 0, 0, -1, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09, 9e+09}”

10.2.24$RESET_POS_TYPE（上电时位置复位方式）

表 10-39 $RESET_POS_TYPE 变量

属性	**说明**
变量名	上电时位置复位方式
数据类型	string 数组[100]
最大值	-
最小值	-
默认值	-
何时恢复默认值	该变量改动立即生效，掉电重启后仍为上次修改值
功能描述	系统上使能时的位置复位方式
值含义	■ feedback：表示使用实际反馈来更新指令位置 ■ cmd：表示当满足指令反馈误差门限要求时，指令维持位置不变

10.2.25$RESET_POS_THESHOLD（上电时位置复位判断门限）

表 10-40 10.2.28$RESET_POS_THESHOLD 变量

属性	**说明**
变量名	上电时位置复位判断门限
数据类型	double 数组[100]
最大值	0.5
最小值	0
默认值	0
何时恢复默认值	该变量改动立即生效，掉电重启后仍为上次修改值
功能描述	上使能时的恢复位置门限值
值含义	当上使能时的恢复位置门限值 RESET_POS_TYPE 设为 cmd，且各轴指令位置与反馈位置偏差小于门限值时，上使能时会恢复上次下使能之前的指令位置。单位为度(°)

附录 A ARL 关键字

附表 A ARL 关键字一览表

序号	名称	序号	名称	序号	名称
1	accset	23	joint	45	stopbits
2	bool	24	jvel	46	stopmove
3	break	25	lin	47	stoptype
4	byte	26	loop	48	string
5	byte	27	movej	49	switch
6	ccir	28	num_base	50	timer
7	cir	29	parity	51	tool
8	clock	30	pause	52	ToolInertiaPara
9	compen	31	pos	53	toolload
10	continue	32	pose	54	toolswitch
11	controlmode	33	print	55	trigger
12	double	34	printto	56	uint
13	endcompen	35	ptp	57	velset
14	endweave	36	repeat	58	waittime
15	exit	37	restart	59	waituntil
16	for	38	return	60	weavedata
17	frame	39	scan	61	weaverotaxis
18	goto	40	slip	62	weaveshape
19	if	41	speed	63	while
20	import	42	startcompen	64	wobj
21	int	43	startmove		
22	interrupt	44	startweave		

附录 B 指令索与变量引表

B

变量

配天机器人

微信公众号　　官方网站

服务热线：400-990-0909
官方网站：http://robot.peitian.com

BJM/SS-UG-02-003 / V4.3.1 / 2022.05.05
© 版权所有 2011-2022 配天机器人保留所有权利.

有关产品特性和可用性说明并不构成性能保证，仅供参考。所交付产品和所执行的服务范围以具体合同为准。