智殷自动化
GENE AUTOMATION

机器人编程指令参考手册

智殷工业机器人专用

文档编号：G-UM-SW00001-A7

版本：A

版次：7

本手册是智殷机器人编程及指令系统的用户说明书。本手册适用于控制器软件V1.5 及以上版本。我们尽最大努力确保本手册的内容准确无误，但是后续的版本可能包含有对规范和操作的更改，这些更改可能是细微改变或者重大变动，也可能会新增本手册中不包括的全新章节和模块。

为了改进产品的可靠性、设计和功能，本手册中的信息如有更改，恕不另行通知，且本手册中的信息并不代表制造商所作的承诺。在产品或文档的使用过程中，发生的直接、间接、特殊、意外或从属损坏（即使已告知可能造成这种损坏），制造商将不承担任何责任。

本手册中提到的产品名称仅用于标识目的，可能是其各自所属公司的商标和/或注册商标。

目录

概览

关于此手册

此手册包含智殷机器人编程及指令系统的以下信息

- 机器人基础
- 机器人编程基础概念
- 常用数据类型
- 常用指令
- 焊接专用指令
- 激光切割专用指令
- 视觉专用指令
- 折弯专用指令
- I/O 系统配置
- 编程示例
- 术语表

用途

此手册在执行以下操作时使用：

- 对机器人系统编程

使用对象

以下人员应当使用此文档:

- 操作人员
- 维护和维修人员

前提条件

阅读此文档的人应当:

- 熟悉工业机器人及相关术语
- 熟悉此类设备

参考文档

文档名称	文档编号	版本

版本

版本号	修订描述
A	**第六版 201911** 增加了 11 条常用指令，增加了第 8 章折弯指令。
	第七版 201912 增加第四章 IO 别名及 GIGO 相关指令，电弧跟踪增加多道指令等 第九章增加总线配置流程。

1 机器人基础

1.1 机器人坐标系

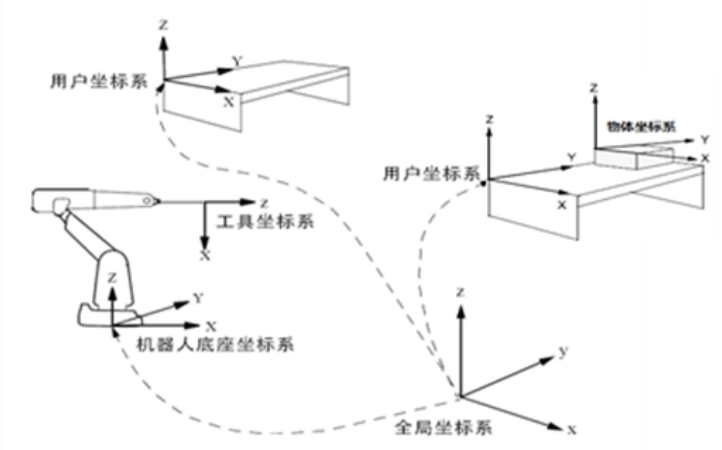
坐标系类别

工业机器人通过编程可以让机器人末端沿着确定的路径运动，运动路径是由一系列目标点构成，这些目标点都是基于特定的坐标系。为了方便灵活的控制机器人的运动，我们提供了多种坐标系，各个坐标系之间满足一定的关联关系。智殷机器人提供如下几种坐标系：

- 世界坐标系（全局坐标系）： 世界坐标系是系统中所有坐标系的参照基准，其他坐标系都是直接或者间接参考这个坐标系通过平移和旋转而得。
- 基坐标系（机器人底座坐标系）： 基坐标系位于机器人底座，它描述了机器人的安装位置信息，即机器人底座在世界坐标系中的位置。世界坐标系理论上和基坐标系可以不同，但是在单个机器人的控制系统中，通常这两个坐标系是一致的。
- 工件坐标系（用户/物体坐标系）： 工件坐标系用于描述待加工工件的安装位置和姿态。如果工件安装在固定工作台或者地面上，则工件坐标系描述了工件相对于世界坐标系的位置信息；如果工件是安装在机器人末端法兰盘上，则工件坐标系描述了工件相对于机器人末端法兰盘的位置信息。
- 工具坐标系： 工具坐标系用于描述工具中心点 (TCP) 的安装位置和姿态。一般情况下，将工具中心点定义在工具的加工点处，如弧焊焊枪的尖端、喷胶胶枪的枪口或者机械手爪的中心等。如果工具是安装在机器人末端法兰盘上，则工具坐标系描述了工具相对于机器人末端法兰盘的位置信息；如果工具安装在固定工作台或者地面上，则工具坐标系描述了工具相对于世界坐标系的位置信息。

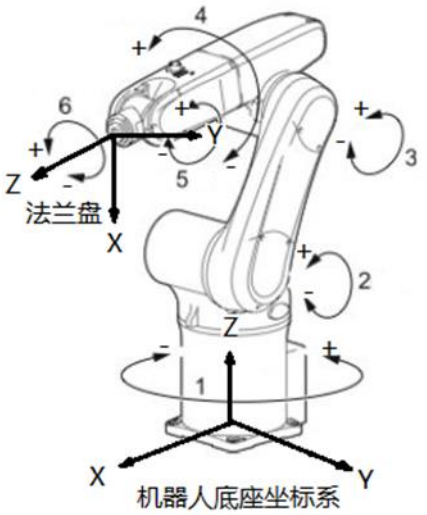
位置信息和姿态信息

坐标系的表述包含位置信息和姿态信息。位置信息（x, y, z）描述该坐标系原点在父坐标系下的位置；姿态信息（q1, q2, q3, q4）描述坐标系姿态在父坐标系下的旋转或姿态信息，姿态信息可以有多种表述如欧拉角，四元数等，在智殷机器人系统中，选用四元数来表达空间点的姿态。详见以下示意图：



1.2 关节和坐标系定义

标准六关节机器人的关节转动方向的定义，如下图所示。下图同时也标出了机器人基坐标系 wobj0 和法兰盘腕坐标系 tool0 的定义。机器人基坐标系 wobj0 的坐标原点位于机器人安装基座的中心点，X 轴指向正前方，Y 轴从机器人的右侧指向左侧，Z 轴垂直于安装基座向上。



智殷机器人的各坐标系都遵从三维笛卡尔坐标系的右手定则。伸出右手的拇指、食指和中指，并使三指成两两垂直状。如下图所示右手定则满足以下两条：

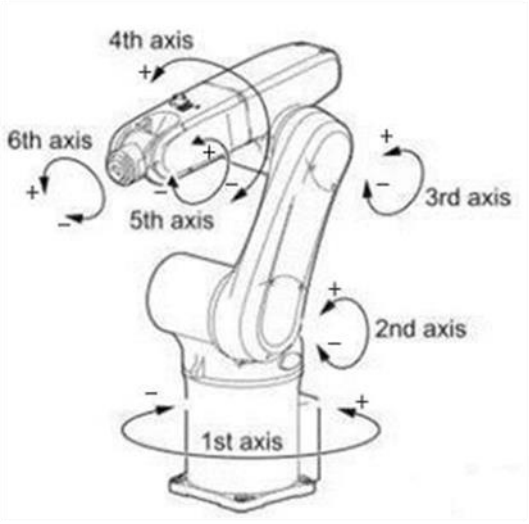
- 如果将食指指向 X 轴的正方向，将中指指向 Y 轴的正方向，那么拇指的指向就是 Z 轴的正方向；
- 如果将右手拇指指向某轴（X/Y/Z 轴）的正方向，那么四指弯曲的方向就是绕该轴旋转（Rx/Ry/Rz）的正方向。

1.3 机器人负载

机器人负载，通常指机器人在移动过程中使用的总负载，即末端法兰盘上安装的工具负载加上该工具正在处理的工件的有效负载，包含质量，重心和惯性张量信息。其中，重心位置相对于机器人法兰盘的 tool0 坐标系；惯性张量相对于以重心点为坐标系原点、姿态和 tool0 一致的参考坐标系计算得到。为了简化计算，通常只需要提取惯性张量的主惯量信息 Ixx，Iyy，Izz。

1.4 机器人奇异点

机器人各轴的转动方向示意图如下图所示。

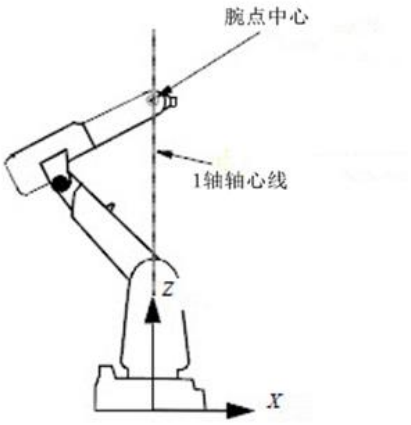


机器人在运动过程中可能会遇到奇异点。奇异点会导致机器人直线和圆弧运动控制计算失效，但是不影响机器人关节运动。

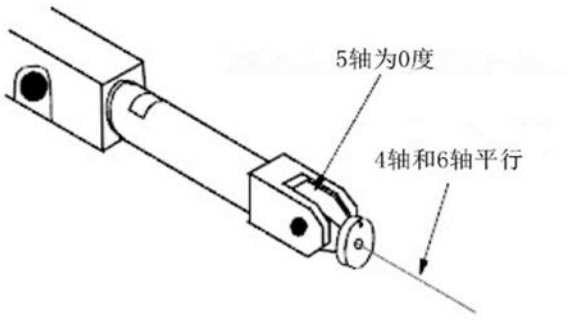
当用户通过示教器手动示教机器人直线或者圆弧运动时，如果接近奇异点位置，机器人会停止运动。

奇异点包括两种类型，一种是机器人在三维空间中到达了极限位置，无法再向更远的地方运动；另一种是和机器人机械结构有关，对于三维空间某些位置，机器人各个关节可能有无穷多种方式到达。

针对标准 6 轴机器人，4，5，6 三轴的轴线交汇于空间的一个点称为腕点。当机器人伸直时候，2 轴轴心，3 轴轴心和腕点处在同一直线上，此时 3 轴为某一特定角度，机器人处于极限位置（机器人手臂已经伸直达到极限位置），如下图所示。所以当机器人 3 轴接近这一特定角度时，机器人处于奇异点位置，此时直线和圆弧运动由于计算失效而无法正常工作。



当 5 轴为 0 度时，4 轴与 6 轴的轴线方向在三维空间共线，4，6 轴可以有无穷多种组合使机器人末端点的位置和姿态保持不变。所以当机器人 5 轴接近 0 度时，机器人处于奇异点位置，此时直线和圆弧运动由于计算失效而无法正常工作，如下图所示。



针对标准 6 轴机器人，还有一种奇异点位置，当腕点经过 1 轴的轴线时 1，2，3 轴可以有无穷多种组合使机器人腕点的位置保持不变，此时直线和圆弧运动由于计算失效而无法正常工作。

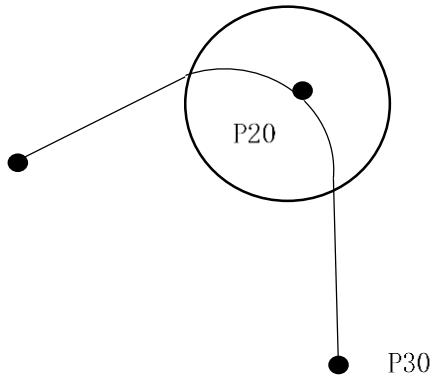
操作人员在操作机器人的过程中，或者机器人程序执行过程中，应该尽量避免经过或者靠近机器人的奇异点位置。当机器人经过或者靠近奇异点位置时，某些关节的角度可能发生快速变化，请注意安全；也有可能发生机器人运动失败的情况，此时需要通过关节运动方式使机器人离开奇异点位置。

1.5 机器人运动路径

机器人运动方式通常包含以下方式：线性运动、关节运动、圆弧运动。线性运动是指机器人从当前点以直线方式运动到目标点，机器人末端或者工具中心点在运动过程中始终处于一条直线上；关节运动指的是机器人从当前点通过转动关节位置来最终到达目标点，在运动过程中机器人末端或者工具中心点不会像线性运动一样始终处于一条直线上；圆弧运动指的是机器人在运动过程中，机器人末端或者工具中心点处于一条圆弧上。

机器人运动点位有 ROBTARGET 和 JOINTTARGET 两种格式表示。ROBTARGET 记录的是目标点的笛卡尔坐标值，由位置和姿态组成；JOINTTARGET 记录的是机器人在目标点时的各关节位置。

对于如何到达目标点，这里提供了两种模式：停止点和经过点。用户可以通过设定运动指令中的转弯路径数据（zonedata）来选择不同模式，如图 1-7 所示。当选择停止点的时候机器人会精确的到达目标点并停止在目标点，下图中机器人会精确停在 P20 点的位置；当选择经过点时，机器人会以一定的速度经过以 P20 点为球心，以指定转弯路径为半径的一个球形区域，不会准确停止在目标点，除非该点为路径上的最后一个点。



2 机器人编程

2.1 程序结构

定义及框架

智殷机器人提供了一种脚本编程语言 GScript，它基于主流的脚本语言 Lua 的语法和特性，可以在一定的范围内兼容 Lua 代码的类型和函数。该语言提供了机器人的运动控制、IO 控制及基本的逻辑处理能力。

基本的程序框架包括：模块、程序数据、指令和函数。模块指的是一个程序文件，它包含程序数据、指令及函数。对于一个简单的机器人应用一个模块就能涵盖所有的数据和逻辑；但是对于复杂的应用，可以创建多个模块文件，模块文件通过特定的 API 来连接其它模块，在加载程序的时候，只需要加载主模块即可，所有其它模块必须放在主模块的同一路径下。

2.2 程序数据

定义

程序数据，指程序运行过程中所需要的数据，包含全局变量和局部变量。全局变量不需要声明，给一个变量赋值后即创建了这个全局变量，如 a = 10.

需要特别说明的是，智殷机器人指令系统中存在一些关键的全局数据变量，它们的数据需要在系统内部保持同步，必须使用专门提供的关键全局变量的定义函数来初始化。常用的关键全局变量包括 JOINTTARGET, ROBTARGET, LOADDATA, TOOLDATA, WOBJDATA, SPEEDDATA, ZONEDATA,NUMDATA 等，它们分别用 JOINTTARGET( )、ROBTARGET( )、LOADDATA( )、TOOLDATA( )、WOBJDATA( )、SPEEDDATA( )、ZONEDATA( )、NUMDATA( )、STRINGDATA( )在主模块的开始进行定义和初始化，其定义方法将在*第 3.2 节关键全局变量*中详细介绍。

关键全局变量定义示例

ROBTARGET("P10",{-19.090000,31.870001,-135.070007},{0.348341,-0.322035,0.590760,0.652651},{-1,1,-1,0},{0,0,0,0,0,0,0},131.214996)

JOINTTARGET("J_Home",{-152.747,59.202,-62.744,-108.506,20.267,-34.303,12.879},{0,0,0,0,0,0,0})

WOBJDATA("wobjC",false,true,"" ,{{434.516,4.953,0},{1,0,0,-0.014}},{{0,0,0},{1,0,0,0}})

TOOLDATA("tool1",true,{{0,0,0},{1,0,0,0}},{0,{0,0,0},{1,0,0,0},0,0,0,0,0,0})

SPEEDDATA("v500",500,500,5000,1000)

ZONEDATA("zone5",0,5,8,8,0.8,8,0.8)

ZONEDATA("finex",1,0,0,0,0,0,0)

局部变量需要使用 local 创建，如 local b = 20。与全局变量不同，局部变量的数据作用域只限制在被声明的那个代码块内部。代码块，指的是一个控制结构、一个函数体或者变量被声明的那个文件。除了在主模块开始处定义的全局变量，应该尽可能的使用局部变量，以避免命名冲突。

2.3 表达式

2.3.1 算术运算符

+，-，*，/，^和 -

二元运算符：+（加法），-（减法），*（乘法），/（除法），^（幂运算）

一元运算符：-（负值）

这些运算符的操作数都是实数。

2.3.2 关系运算符

<，>，<=，>=，==和~=

<（小于），>（大于），<=（小于等于），>=（大于等于），==（等于）以及 ~=（不等于）的返回结果为 false 或者 true。

==和~=用于比较两个值，如果两个值类型不同，则两者不同；nil 只和自己相等。

比较数字时按数字大小进行，比较字符串按字母的顺序进行。比如，2<15 返回值为 true，但是“2”<“15”则返回值为 false。

当比较不同类型的值时需要特别注意，比如，“0” == 0 返回值为 false。

2.3.3 逻辑运算符

and（与） or（或） not（非）

逻辑运算符认为 false 和 nil 都是假（false），而其他值为真（true），注意 0 也是真值。

and 和 or 的运算结果不是 true 和 false，而是和它的两个操作数相关。

a and b -- 如果 a 为 false，则返回 a，否则返回 b

a or b -- 如果 a 为 true，则返回 a，否则返回 b

示例

4 and 5 返回值为 5；nil and 13 返回值为 nil；4 or 5 返回值为 4；false or 5 则返回值为 5。

一个很实用的语句：如果 x 为 false 或者 nil，则给 x 赋初始值 v。可以表示为：x = x or v，等价于：

```
if not x then
 x = v
end
```

2.3.4 字符串连接运算符

..（连接）

字符串连接，“Hello ” .. “World”返回值为 Hello World。

如果操作数为数字，则将数字也转成字符串处理。

示例

```
local  X = 1.8
local  Y = -2.3
local  Z = 30
print ("X = " .. X .. ",Y = " .. Y .. ",Z = " .. Z)
```

打印出的结果是：X = 1.8,Y = -2.3,Z = 30。

2.4 控制结构

控制结构的条件表达式结果，理论上可以是任何值。注意，仅有 false 和 nil 为假，其他值都为真。

2.4.1 if 语句

if 语句

用于判断条件里面的内容是否满足。若条件满足，则执行下面的程序；若条件不满足，则程序不执行 if---end 所包含的内容。若有多个条件进行判断，还可以采用 if…else if…else…end。

示例

假如满足条件 LVAR1=0，就执行 LVAR2 加 1，若满足条件 LVAR1=1，就执行 LVAR2 减 1，若两个条件都不满足，则将 LVAR2 设置为 0。

程序

```
local LVAR1 = 0
local LVAR2 = 0
if (LVAR1  ==  0) then
      LVAR2 = LVAR2 + 1
elseif (LVAR1  ==  1) then
      LVAR2 = LVAR2 - 1
else
      LVAR2 = 0
end
```

2.4.2 while 语句

while 语句

用于判断条件里面的内容是否满足。当 while 条件满足要求时，执行 do---end 里面的程序，直到 while 条件不满足要求，退出该循环。

示例

当 LVAR1< 10 时，执行 do---end 里面的程序。(LVAR1 的初始值为 0，循环体里面的程序一共执行了 10 次)

程序

```
local LVAR1 = 0
while (LVAR1 < 10) do
    LVAR1 = LVAR1 + 1
end
```

在循环体中，一定要对 LVAR1 进行加一操作，否则该程序将无法离开此循环。

2.4.3 for 语句

for 语句

用于判断条件里面的内容是否满足。当 for 条件满足要求时，执行 do---end 里面的程序，直到 for 条件不满足要求，退出该循环。

示例 1

LVAR1 的值从 1 增加到 10，每次递增 1，do---end 里面的程序一共被执行 10 次。

程序

```
local LVAR1 = 0
local LVAR2 = 0
for LVAR2 = 1,10,1 do
    LVAR1 = LVAR1 + 1
end
```

当循环结束后，LVAR1 的值是 10。

示例 2

LVAR1 的值从 1 增加到 10，每次递增 2，do---end 里面的程序一共被执行 5 次。

程序

```
local LVAR1 = 0
local LVAR2 = 0
for LVAR2 = 1,10,2 do
      LVAR1 = LVAR1 + 1
end
```

当循环结束后，LVAR1 的值是 5。

2.4.4 break 和 return 语句

break 语句

用于跳出当前的 while 或者 for 循环。在循环外部不可以使用。

示例

```
local i = 1
while a[i] ~= nil do
    if a[i] == v then
        break
    end
    i = i + 1
end
```

说明

查找数组中等于 v 的位置，判断 i 位置的 a[i]是否等于 v，等于 v 时跳出循环。

return 语句

用于从函数返回结果，当一个函数自然结束，结尾会有一个默认的 return。

注意！

break 和 return 只能出现在代码块的结尾一句，或者在 end 之前，或者 else 之前。有时候为了调试或者其他目的需要在代码块的中间使用 break 和 return，可以显示的用 do…end 来实现：

```
function PickPart ( )
    …
    do return end      -- OK
    …                  -- statements not reached
end
```

2.5 数学函数

智殷机器人编程语言，可以兼容 LUA 的 math 库里的一些特殊变量（比如 pi 等） 和数学函数（比如三角函数、指数和对数函数、取整函数、max 和 min、随机数函数等）。

下面列出了这些常用的数学函数。

函数名	描述	示例	结果
pi	圆周率	math.pi	3.1415926535898
abs	取绝对值	math.abs(-2012)	2012
ceil	向上取整	math.ceil(9.1)	10
floor	向下取整	math.floor(9.9)	9
max	取参数最大值	math.max(2,4,6,8)	8
min	取参数最小值	math.min(2,4,6,8)	2
pow	计算 x 的 y 次幂	math.pow(2,16)	65536
sqrt	开平方	math.sqrt(65536)	256
mod	取模	math.mod(65535,2)	1
modf	取整数和小数部分	math.modf(20.12)	20　0.12
randomseed	设随机数种子	math.randomseed(os.time())	
random	取随机数	math.random(5,90)	5~90
rad	角度转弧度	math.rad(180)	3.1415926535898
deg	弧度转角度	math.deg(math.pi)	180
exp	e 的 x 次方	math.exp(4)	54.598150033144
log	计算 x 的自然对数	math.log(54.598150033144)	4
log10	计算 10 为底，x 的对数	math.log10(1000)	3
frexp	将参数拆成 x*(2^y)的形式	math.frexp(160)	0.625　8
ldexp	计算 x*(2^y)	math.ldexp(0.625,8)	160
sin	正弦	math.sin(math.rad(30))	0.5
cos	余弦	math.cos(math.rad(60))	0.5
tan	正切	math.tan(math.rad(45))	1

asin	反正弦	math.deg(math.asin(0.5))	30
acos	反余弦	math.deg(math.acos(0.5))	60
atan	反正切	math.deg(math.atan(1))	45

2.6 文件操作

通过使用外部的文件句柄，可以很方便的实现一些文件的常用操作。

2.6.1 OpenFile 打开文件

函数功能

按指定的模式打开一个文件。如果打开文件成功，则返回文件句柄；如果打开文件失败，则返回 nil 和一个错误代码。

文件默认访问路径：控制器 Program 目录。

函数结构

fd,err = OpenFile (fileName, [mode])

返回值

fd： 为句柄。

err：错误代码。

函数参数

参数	说明
filename	要打开文件的文件名
mode	指定打开文件的模式，为可选参数。 如果省略，默认为"r"。
	"r":以只读方式打开文件。 该文件必须存在(默认)。
	"w":打开只写文件。

	若文件存在则文件长度清为 0，即该文件内容会消失。若文件不存在则建立该文件。
	"a":以附加的方式打开只写文件。 若文件不存在，则会建立该文件。 如果文件存在，写入的数据会被加到文件尾，即文件原先的内容会被保留。（EOF 符保留）。
	"r+":以可读写方式打开文件。 该文件必须存在。
	"w+":打开可读写文件。 若文件存在则文件长度清为零，即该文件内容会消失。若文件不存在则建立该文件。
	"a+":与 a 类似，但此文件可读可写。
	"b":二进制模式，如果文件是二进制文件，可以加上 b。 “+”: 号表示对文件既可以读也可以写。

调用示例

local fd, err = OpenFile (fileName, ”a+”)

说明

以添加模式打开文件，如果打开文件成功则返回文件句柄 fd，失败则返回 nil + 错误信息。

2.6.2 ReadFile 读取文件数据

函数功能

根据所给的格式来读取文件内容，相当于对标准输入文件的操作。如果读取文件成功，则返回所指定格式的读取内容；如果读取文件失败，则会返回 nil。

函数结构

str = ReadFile (handle, [format],[num])

返回值

str：返回 str 的内容由 format 的参数决定。

函数参数

参数	说明
handle	读取文件的句柄。
format	指定读取文件的格式，为可选参数。如果省略，默认为"*l"。 "*n"表示读取一个数字，这是唯一返回数字而不是字符串的读取格式； "*a"表示从当前位置读取余下的所有内容，如果在文件尾，则返回空串""； "*l"表示读取下一个行内容，如果在文件尾部则会返回 nil。
num	可选参数，读取 num 个字符的字符串。如果 number 为 0，则返回一个空串""；如果在文件尾，则会返回 nil。

调用示例

打开文件

local fd, err = OpenFile ("readtest.txt", "r")

读取数字

local number = ReadFile (fd, "*n")

读取行

local content = ReadFile (fd, "*l")

读取 6 个字节

```
local sixbyte = ReadFile (fd, 6)
```

读取所有内容

```
local readall = ReadFile (fd, "*a")
……
```

2.6.3 ReadFileLines 读取文件行

函数功能

以"r"读模式打开指定的文件，会返回一个迭代函数。每次调用 ReadFileLines 将获得该文件中的一行内容；当到文件尾时，将返回 nil，但结束时不关闭文件。

函数结构

val = ReadFileLines (handle)

返回值

val：返回一行的内容，如返回 nil，则到文件尾。

函数参数

参数	说明
Handle	读取文件的句柄

调用示例

打开文件

```
local fd, err = OpenFile(fileName,"r")
if nil == fd then
    print("fail to open file!")
    return false
end
```

逐行读取文件并打印

```
for line in ReadFileLines(fd) do
    print(line)
end
……
```

2.6.4 SeekFile 设置和获取当前文件位置

函数功能

获取或者设置一个文件的当前位置。如果获取或者设置成功，则返回文件的当前位置，即相对于描述位置处的偏移字节数；如果失败，则返回 nil 和一个错误信息。

函数结构

val = SeekFile (handle, [whence], [offset])

返回值

val： 返回当前位置。如果失败，则返回 nil 和一个错误信息

函数参数

参数	说明
handle	所操作文件的句柄。
whence	表示描述参数，为可选参数。如果省略，默认为"cur" "set"表示设置从文件头开始 "cur"表示设置从当前位置开始 "end"表示设置从文件尾开始。
offset	表示偏移的字节数，为可选参数。如果省略，默认为 0 当 whence 取值 set，offset 表示为相对于文件起始的偏移量。 当 whence 取值 cur，offset 表示为相对于当前位置的偏移量。 当 whence 取值 end，offset 表示为相对于文件末尾的偏移量。

调用示例

打开文件

local fd, err = OpenFile("log.txt","r")

获取文件的当前位置

local current = SeekFile(fd)

获取文件的大小

```
local size = SeekFile(fd, "end")
```

恢复文件的位置

```
SeekFile(fd, "set", current)
```

2.6.5 WriteFile 写入文件数据

函数功能

将一个或者多个数据写入到文件中，可以有多个参数。如果写入文件成功，则返回 true；如果写入文件失败，则返回 nil。

函数结构

val = WriteFile (handle, param1, param2, …)

返回值

val：如果写入文件成功，则返回 true；如果写入文件失败，则返回 nil。

函数参数

参数	说明
handle	打开文件的句柄
param1	参数的类型必须是字符串或者数字。如果要写入其他类型，则需要使用 tostring(arg)函数或者 string.format()函数转换成字符串

调用示例

打开文件

```
local fd, err = OpenFile ("writetest.txt", "w")
if nil == fd then
    print("fail to open file!")
    return false
end
```

写入字符串

```
WriteFile (fd, "test WriteFile\n")
```

写入数字

```
WriteFile (fd, 2017)
```

写入分隔符

```
WriteFile (fd, " ")
```

写入回车符

```
WriteFile (fd, "\n")
```

继续写入其他类型

```
WriteFile (fd, tostring(os.date("%x %X")))
WriteFile (fd, "\n")
```

2.6.6 CloseFile 关闭文件

函数功能

关闭文件。当文件描述符被垃圾回收时，对应的文件也会自动关闭，但是这个时间是不确定的。

函数结构

CloseFile (handle)

函数参数

参数	说明
handle	所操作文件的句柄

调用示例

打开文件

```
local fd, err = = OpenFile (fileName, 'a+')
if nil == fd then
    print("fail to open file!")
    return false
end
```

写入系统日期

```
local date = os.date("%x %X")
WriteFile (fd, date)
```

写入回车符

```
WriteFile (fd, "\n")
```

关闭文件

```
CloseFile (fd)
```

3 常用数据类型

3.1 基础数据变量

3.1.1 pos 笛卡尔坐标

pos 数据结构定义

pos 数据结构定义为：

{ x, y, z}

笛卡尔坐标下的位置定义为{x, y, z}。 x, y, z 数据类型为：float，单位：mm。用于表示机器人的位置，包括工具和轴的坐标配置。标记坐标系的位移。

定义示例

pos1 = {100,0,0}

机器人在笛卡尔坐标系下的位置。

3.1.2 orient 姿态

orient 数据结构定义

orient 数据结构定义为：

{ q1，q2，q3，q4}

Orient 数据表示机器人的姿态，由四元数的形式描述：{q1，q2，q3，q4}。

q1，q2，q3，q4 数据类型为 float。

定义示例

orient1={1, 0, 0, 0}；

机器人姿态用四元数描述成{1, 0, 0, 0}。

3.1.3 pose 位姿

pose 数据结构定义

pose 数据结构定义为：

{ pos, orient}

它包含两个子数据结构：一个笛卡尔坐标和一个姿态数据。

pose 坐标数据可以描述在坐标系下的一个点的位置和姿态，也可以用来描述从一个坐标系到另一个坐标系的变换。

trans 笛卡尔坐标

trans 笛卡尔坐标下的位置，单位为 mm。

数据类型: pos

坐标系中的位置{x, y, z}。

rot 姿态变化

数据类型: orient

坐标系姿态，数据结构使用四元数{q1,q2,q3,q4}。

定义示例

Pose = {{100, 0, 0},{1,0,0,0}}

3.1.4 config 关节配置

config 数据结构定义

config 数据结构定义为：

{ j1, j4, j6, jx}

config 是指机器人关节位置的配置信息，用于定义机器人的第一、四、六轴及外部轴的所在象限。

0：第一象限

1： 第二象限

2： 第三象限

-1：第四象限

定义示例

conf1 = {1, -1, 0, 0}。

定义机器人的一轴在第二象限，四轴在第四象限，六轴在第一象限，外部轴在第一象限。

3.1.5 triggdata 触发数据

triggdata 是一种非数值的数据类型，用于储存有关机器人运动期间定位事件的数据。一起定位事件的具体形式，既可以是设置一个输出信号，也可以是在机器人移动路径上的某特定位置处运行一则中断例程。

为定义有关定位事件应对措施的条件，使用 triggdata 类变量。通过使用指令

TriggIO、TriggCheckIO、TriggSpeed、TriggEquip 之一，在程序中形成变量的

数据内容，并由指令 TriggL、TriggC、TriggJ之一或焊接指令 ArcCStart、ArcC、ArcCEnd、SpotL、SpotJ 指令使用。

3.2 关键全局变量

在智殷机器人指令系统中存在一些关键的全局数据变量，比如 JOINTTARGET, ROBTARGET, LOADDATA, TOOLDATA, WOBJDATA, SPEEDDATA, ZONEDATA 等.

它们需要在系统内部进行实时同步，为此专门提供了一种全局变量的定义格式。

用全部大写字母命名的函数 JOINTTARGET( )、ROBTARGET( )、LOADDATA( )、TOOLDATA( )、WOBJDATA( )、SPEEDDATA( )、ZONEDATA( )、NUMDATA( ) 、STRINGDATA( )来定义它们，并且必须定义在主模块的开始。

3. 2. 1 JOINTTARGET 关节位置数据

数据结构

关节位置数据，用于定义机器人的各轴和外轴所在的关节空间位置，单位为角度（可能是毫米）。其数据结构定义为：

```
{
   robax {rax1, rax2, rax3, rax4, rax5, rax6, rax7},
   extax {eax1, eax2, eax3, eax4, eax5, eax6, eax7}
}
```

参数包含两个 float（Num）型数组，一组数据为机器人各轴的位置数据，另一组数据为外部轴的各轴位置数据。可以支持 6 关节和 7 关节机器人，当为 6 关节机器人时，最后一个参数无效。

使用示例

作为关键全局变量，关节位置数据用 JOINTTARGET( ) 在主模块的开始进行定义和初始化，示例如下：

```
JOINTTARGET("JHome",{0,0,0,-0,49.445,-0.001,-0},{0,0,0,0,0,0,0});
```

定义一个名为“JHome”的 JOINTTARGET 类型的全局变量并赋值。

3.2.2 ROBTARGET 机器人位置数据

机器人位置数据，表示机器人的运动目标点，单位为毫米。RobTarget 数据可以单独使用，也可以通过 MPointArray 指令创建 RobTarget 点位数组来使用。

数据结构

ROBTARGET 数据结构定义为：

```
{
    pos trans,
    orient rot,
    config robconf,
    extax {eax1, eax2, eax3, eax4, eax5, eax6, eax7},
    armAngle
}
```

机器人位置数据参数

机器人位置数据包含五个参数：

trans 笛卡尔位置

数据类型: pos；tool 中心点的位置{x，y 和 z}, 单位为 mm。

rot 姿态

数据类型: orient；tool 转动方向，数据结构使用 4 个参数 q1,q2,q3 和 q4。

robconf 关节配置

数据类型: config；机器人的某些轴所在的象限，这些轴通常为 1 轴、4 轴、6 轴和一个其他轴，其他轴取决于特定型号的机器人。（cf1, cf4, cf6, and cfx）。

extax 外部轴位置

数据结构: joints；外部的轴的位置。轴的位置定义为 float 类型数组(eax1,eax2...eax7)。

armAngle 臂型角

七轴机器人特有参数，可通过臂型角调整机械臂的位置组合，保持机器人末端不动。浮点型数据，单位为角度。

使用示例

作为关键全局变量，机器人位置数据用 ROBTARGET( ) 在主模块的开始进行定义和初始化，示例如下：

ROBTARGET("P10",{900,0,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0,-0},-0)

定义一个名为“P10”的 ROBTARGET 类型变量并赋值。

ROBTARGET("PArray10_1",{925,0,-385},{0,0,1,0},{0,0,0,0},{0,0,0,0,0,0,0,0},0)

ROBTARGET("PArray10_2",{925,0,-385},{0,0,1,0},{0,0,0,0},{0,0,0,0,0,0,0,0},0)

ROBTARGET("PArray10_3",{925,0,-385},{0,0,1,0},{0,0,0,0},{0,0,0,0,0,0,0,0},0)

ROBTARGET("PArray10_4",{925,0,-385},{0,0,1,0},{0,0,0,0},{0,0,0,0,0,0,0,0},0)

ROBTARGET("PArray10_5",{925,0,-385},{0,0,1,0},{0,0,0,0},{0,0,0,0,0,0,0,0},0)

定义一个名为“PArray10”的 ROBTARGET 类型的点位数组。

3.2.3 LOADDATA 负载数据

数据结构

负载数据，用于描述附加到机械臂的安装法兰的负载。负载数据常常定义机械臂的有效负载或者支配负载，即机械臂夹具所施加的负载。同时将 loaddata 作为 tooldata 的组成部分，以描述工具负载。其数据结构定义如下：

```
{
    mass,
    pos cog,
    rot aom,
    ix,iy,iz,ixy,iyz,izx
}
```

负载数据参数

负载数据包含九个参数：

mass 负载质量

数据类型: float；重量单位为 kg。

cog　质心坐标系笛卡尔位置

数据类型: pos；{x，y 和 z}, 单位为 mm。

aom 质心坐标系姿态

数据类型: orient；数据结构使用 4 个参数{q1,q2,q3,q4}。

ix,iy,iz,ixy,iyz,izx 惯量坐标

数据类型：float；数据结构：ix,iy,iz,ixy,iyz,izx。

使用示例

作为关键全局变量，负载数据用 LOADDATA( ) 在主模块的开始进行定义和初始化，示例如下：

LOADDATA("Load_6kg",0,{74.3,0,0},{1,0,0,0},0.0418,0.0418,0.0102,0,0,0)

定义一个名为“Load_6kg”的 LOADDATA 数据类型变量。

3.2.4 TOOLDATA 工具数据

数据结构

工具数据，是描述工具特征的数据结构，它包含工具工作点的位置和姿态、工具的质量、质心位置等物理特征，以及工具是被机器人抓着还是固定在工作空间里。其数据结构定义如下：

```
{
  bool robhold,
  pose tframe,
  loaddata tload
}
```

工具数据参数

工具数据包含三个参数：

robhold

数据类型：bool ，定义工具是否被机器人抓着。

true: 机器人拿着工具。

false: 机器人不拿着工具，也就是一个固定的工具。

tframe

数据类型： pose，该工具安装坐标系，即：

在安装坐标系中，工具中心点的位置{x，y，z}单位 mm；

在安装坐标系中，工具的姿态{q1,q2,q3,q4}。

pose 数据结构： {{x,y,z}，{q1,q2,q3,q4}}

tload

数据类型： loaddata，描述机器人工具或夹具的负载参数。

工具的重量，单位 kg。

在安装坐标系中负载的重心（x,y,z），单位 mm。

在安装坐标系中表示的工具的主轴的姿态{q1,q2,q3,q4}。

惯性主轴的惯量（x,y,z）,单位 kgm^2。如果所有轴惯量定义为 0，该工具是作为质点处理。

使用示例

作为关键全局变量，工具数据用 TOOLDATA( ) 在主模块的开始进行定义和初始化，示例如下：

TOOLDATA("tool1",true,{{0,0,0},{1,0,0,0}},{0,{0,0,0},{1,0,0,0},0,0,0,0,0,0})

3.2.5 WOBJDATA 工件数据

数据结构

工件数据，用于描述工件坐标系，它的内部数据包含两个空间坐标系，即用户坐标系和物体坐标系。其数据结构定义如下：

```
{
   bool  robhold,
   int  ufprog,
   string ufmec,
   pose uframe;
   pose oframe;
}
```

数据参数

工件数据包含五个参数：

robhold

数据类型：bool；机器人是否抓着该工件坐标系。

true ： 机器人拿着工件坐标系。

false ： 机器人不拿着工件坐标系， 工件坐标系固定。

ufprog

用户坐标(user frame)是固定的还是程序控制的。

数据类型：boolean；定义是否使用固定的用户坐标系统。

true ： 固定的用户坐标系。

false : 用户坐标系可编程。

ufmec

user frame mechanical unit

数据类型: string；机器人运动协调的机械装置。

只有在可移动用户坐标系统的情况下指定（ufprog 为 false）。

uframe

user frame

数据类型: pose；用户坐标系，即当前工作面或夹具的位置

坐标系原点的位置（x，x，x）单位 mm。

坐标系的姿态，表示为{q1，q2，q3, q4}。

oframe

object frame

数据结构: pose；物体坐标系，即当前工作对象的位置：

坐标系原点的位置（x，y, z）。

坐标系的姿态，表示为一个四元数（q1，q2，q3，q4）。

使用示例

作为关键全局变量，工件数据用 WOBJDATA( ) 在主模块的开始进行定义和初始化，示例如下：

```
WOBJDATA("wobj1",false,true,"",{{-0,-0,-0},{1,-0,-0,-0}},{{-0,-0,-0},{1,-0,-0,-0}})
```

3.2.6 SPEEDDATA 速度数据

数据结构

速度数据，用于设置机器人的移动速度。其数据结构定义如下：

```
{
  float v_tcp,
  float v_ori,
  float v_leax,
  float v_reax
}
```

数据参数

速度数据包含四个参数：

v_tcp

tcp 在指定坐标系下的运动速度

数据类型: float

单位：mm/s。

v_ori

姿态改变的速度

数据类型: float

单位：degrees/s。

v_leax

外部轴线性移动速度

数据类型: float

单位：mm/s。

v_reax

外部轴旋转速度

数据类型: float

单位：degrees/s。

速度数据也是关键全局变量，但是用户不需要在应用程序里自行定义，可以直接使用系统中预定义的各种常用的速度数据，如下表所示：

速度编号	Tcp 运动速度 (mm/s)	姿态改变速度 (°/s)	外部轴线性移动速度(mm/s)	外部轴旋转速度(°/s)
v5	5	500	5000	1000
v10	10	500	5000	1000
v20	20	500	5000	1000
v30	30	500	5000	1000
v40	40	500	5000	1000
v50	50	500	5000	1000
v60	60	500	5000	1000
v80	80	500	5000	1000
v100	100	500	5000	1000
v150	150	500	5000	1000
v200	200	500	5000	1000
v300	300	500	5000	1000
v400	400	500	5000	1000
v500	500	500	5000	1000
v600	600	500	5000	1000
v800	800	500	5000	1000
v1000	1000	500	5000	1000
v1500	1500	500	5000	1000
v2000	2000	500	5000	1000

3.2.7 ZONEDATA 区域数据

数据结构

区域数据，用于指定转弯半径。其数据结构定义如下：

```
{
   int finep;
   float pzone_tcp;
   float pzone_ori;
   float pzone_eax;
   float zone_ori;
   float zone_leax;
   float zone_reax;
}
```

数据参数

区域数据包含 7 个参数：

finep 是否精确到达目标点

数据类型：bool；若 finep 值为 true， 机器人会精确走到目标点并停止，如果为 false，机器人会走到一目标点位球心，以指定转弯半径的球型区域，并且保持一定的速度向下一个目标点移动。

pzone_tcp 路径精度半径

数据类型: float；TCP 转弯半径大小，单位：mm。

pzone_ori 路径姿态半径

数据类型: float；工具姿态调整的半径大小，从指定点到 TCP 的距离，单位：mm。

此值必须大于 pzone_tcp 值。

pzone_eax 外部轴路径半径

数据类型: float；外轴的区域大小（半径）。

大小定义为 TCP 到指定点的距离，单位：mm。此值必须大于 pzone_tcp 值。

zone_ori 工具姿态精度半径

数据类型: float；tool 区域定位大小，单位：degrees。

zone_leax 外部轴线性移动精度半径

数据类型: float；线性外部轴的区域大小，单位：mm。

区域数据也是关键全局变量，但是用户不需要在应用程序里自行定义，可以直接使用系统中预定义的各种常用的区域数据，如下表所示：

路径 zone				Zone		
Zone 编号	TCP 路径精度半径(mm)	路径姿态半径(mm)	外部轴路径半径(mm)	工具姿态精度半径(°)	外部轴线性移动精度半径（mm）	外部轴旋精度半径(mm)
fine	0	0	0	0	0	0
z0	0.3	0.3	0.3	0.03	0.3	0.03
z1	1	1	1	0.1	1	0.1
z5	5	8	8	0.8	8	0.8
z10	10	15	15	1.5	15	1.5
z15	15	23	23	2.3	23	2.3
z20	20	30	30	3	30	3
z30	30	45	45	4.5	45	4.5
z40	40	60	60	6	60	6
z50	50	75	75	7.5	75	7.5
z60	60	90	90	9	90	9
z80	80	120	120	12	120	12
z100	100	150	150	15	150	15
z150	150	225	225	22.5	225	22.5
z200	200	300	300	30	300	30

zone_reax 外部轴旋转精度半径

数据类型: float；外部轴旋转区域大小单位：degrees。

3.2.8 NUMDATA 字符数据

数据结构

数值数据，用于定义数值变量。其数据结构定义如下：

```
{
    string name;
    int/double data;
}
```

数据参数

数值数据包含两个参数：

Name 数据名称

数据类型：string。

data 字符串数据值

数据类型：int 或 double。

使用示例

作为关键全局变量，字符数据用 NUMDATA( ) 在主模块的开始进行定义和初始化，示例如下：

```
NUMDATA("var1",10)
```

3.2.9 STRINGDATA 字符数据

数据结构

字符数据，用于定义字符变量。其数据结构定义如下：

```
{
   string  strData;
   string dataVal
}
```

数据参数

字符数据包含两个参数：

strData 字符串数据名称

数据类型：string。字符数据名称。

dataVal 字符串数据值

数据类型：string。字符数据名称。

使用示例

作为关键全局变量，字符数据用 STRINGDATA( ) 在主模块的开始进行定义和初始化，示例如下：

```
STRINGDATA("teststr1","Hahahaha1")
```

4 常用指令

指令，指一个具体的操作，如一条运动指令、IO 指令。本章将对智殷机器人编程语言的常用指令进行详细说明。

4.1 MoveL 线性运动

指令功能

MoveL 用于将工具中心点（TCP）以直线插补移动至给定目标位置。当 TCP 保持固定时，则该指令亦可用于调整工具方位。

指令结构

MoveL([Motgrp/Mec_name], ToPoint, Speed, Zone, Tool, WObj, [Load])

指令参数

参数	说明
Motgrp/Mec_name	这个参数是可选参数，可以选择 Motgrp 赋值，也可以选择 Mec_name 赋值。若选择 Motgrp：0—左臂，1—右臂。如果不设置，则默认为左臂。若选择 Mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”，对于单个机器人系统，该参数通常省略。
ToPoint	数据类型为 robtarget。机器人和外部坐标轴的目标点，定义为一个指定位置。
Speed	数据类型为 speeddata。运动的速度数据。定义的 tcp 的运动速度。
Zone	数据类型 zonedata。运动区域数据，描述生成的区域半径的大小。
Tool	数据类型 tooldata。机器人移动时使用的 tool 数据。该 tool 中心点是指向指定目标位置的点。
WObj	数据类型 wobjdata。指定机器人运动的坐标系。如果使用固定 tool 参数或协同的外部轴，则此参数必须指定以执行相对于工作对象的线性运动。
Load	数据类型 loaddata。可选参数，指定机器人移动中使用的总负载，即末端法兰盘上安装的工具负载加上该工具正在处理工件的有效负载。如果使用了本参数，那么就不再考虑当前 tooldata 中的 loaddata；如果省略掉本参数，就表示可以忽略处理工件的负荷。

指令示例

以下两个示例介绍了指令 MoveL 的基本用法：

例 1：

工具 tool1 将线性运动至位置 P10，其速度数据为 v200，且经过 fine 点。

```
ROBTARGET("P10",{900,0,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0},-0)
TOOLDATA("tool1",true,{{0,0,0},{1,-0,-0,-0}},{-0,{-0,-0,-0},{1,-0,-0,-0},-0,-0,-0,-0,-0,-0})
WOBJDATA("wobj1",false,true,"",{{-0,-0,-0},{1,-0,-0,-0}},{{-0,-0,-0},{1,-0,-0,-0}})
MoveL(P10,v200,fine,tool1,wobj1)
```

例 2：

工具 tool2 将线性运动至位置 P20，其速度数据为 v500，且区域数据为 z10。

```
ROBTARGET("P20",{900,0,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0},-0)
TOOLDATA("tool2",true,{{0,0,0},{1,-0,-0,-0}},{-0,{-0,-0,-0},{1,-0,-0,-0},-0,-0,-0,-0,-0,-0})
WOBJDATA("wobj1",false,true,"",{{-0,-0,-0},{1,-0,-0,-0}},{{-0,-0,-0},{1,-0,-0,-0}})
MoveL(0,P20,v500,z10,tool2,wobj1)
```

4.2 MoveC 圆弧运动

指令功能

MoveC 用于将工具中心点（TCP）沿圆弧运动至给定目的地。移动期间，该周期的方位通常相对保持不变。

指令结构

MoveC([Motgrp/Mec_name], CirPoint, ToPoint, Speed, Zone, Tool, WObj, [Load])

指令参数

参数	说明
Motgrp/Mec_name	可选参数，可以选择 Motgrp 作参数，也可以选择 Mec_name 作参数。若选择 Motgrp：0—左臂，1—右臂，如果不设置，则默认为左臂。若选择 Mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”，对于单个机器人系统，该参数通常省略。
CirPoint	数据类型 robtarget。相关机器人的圆弧点。圆弧点，是指相关起点与终点间的圆弧上的某个位置。为了获得最佳的准确度，应该把该点放在相关起点和终点的正中间处。如果该点太靠近起点或终点，机器人可能会发出警告。将圆弧点定义为一个已命名的位置，或直接存储在相关指令中。
ToPoint	数据类型 robtarget。机器人和外部坐标轴的目标点，定义为一个指定位置，或者直接存储在指令中。
Speed	数据类型 speeddata。运动的速度数据。定义的工具中心点、工具方位调整和外轴的速度。
Zone	数据类型 zonedata。运动区域数据,描述生成的角路径的大小。
Tool	数据类型 tooldata。机器人移动时使用的 tool 数据。该 tool 中心点是指向指定目标位置的点。
WObj	数据类型 wobjdata。指定机器人在坐标系中的工作位置。位置与坐标系有关，可以省略。如果使用固定 tool 参数或协同的外部轴，则此参数必须指定以执行相对于工作对象的线性运动。
Load	数据类型 loaddata。可选参数，指定机器人移动中使用的总负载，即末端法兰盘上安装的工具负载加上该工具正在处理工件的有效负载。如果使用了本参数，那么就不再考虑当前 tooldata 中的 loaddata；如果省略掉本参数，就表示可以忽略处理工件的负荷。

指令示例

以下两个示例，介绍了 MoveC 指令的基本用法。

例 1：

工具 tool1 沿圆弧运动至位置 P30，其速度数据为 v200 且区域数据为 z10。根据起始位置 P10、圆弧点 P20 和目的点 P30，确定该圆弧路径。

```
ROBTARGET("P10",{900,0,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0},-0)
ROBTARGET("P20",{1000,-100,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0},-0)
ROBTARGET("P30",{1100,0,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0},-0)
TOOLDATA("tool1",true,{{0,0,0},{1,-0,-0,-0}},{-0,{-0,-0,-0},{1,-0,-0,-0},-0,-0,-0,-0,-0,-0})
WOBJDATA("wobj1",false,true,"",{{-0,-0,-0},{1,-0,-0,-0}},{{-0,-0,-0},{1,-0,-0,-0}})
MoveL(P10,v200,fine,tool1,wobj1)
MoveC(P20,P30,v200,z10,tool1,wobj1)
```

例 2：

如何通过两个 MoveC 指令，实现一个完整的圆形路径。

```
ROBTARGET("P10",{900,0,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0},-0)
ROBTARGET("P20",{1000,-100,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0},-0)
ROBTARGET("P30",{1100,0,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0},-0)
ROBTARGET("P40",{1000,100,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0},-0)
TOOLDATA("tool1",true,{{0,0,0},{1,-0,-0,-0}},{-0,{-0,-0,-0},{1,-0,-0,-0},-0,-0,-0,-0,-0,-0})
WOBJDATA("wobj1",false,true,"",{{-0,-0,-0},{1,-0,-0,-0}},{{-0,-0,-0},{1,-0,-0,-0}})
MoveL(P10,v200,fine,tool1,wobj1)
MoveC(P20,P30,v200,z10,tool1,wobj1)
MoveC(P40,P10,v200,z10,tool1,wobj1)
```

工具 tool1 从起点 P10 经中间点 P20 沿圆弧运动至位置 P30，再从 P30 点经 P40 中间点运动到 P10 点，其速度数据为 v200 且区域数据为 z10。通过两个 MoveC 指令，实现一个完整的圆形路径

4.3 MoveJ 关节运动

指令功能

本指令用于将机器人以关节插补迅速地从一点移动至另一点。机器人和外轴沿非线性路径运动至目标位置。所有轴均同时达到目标位置。

指令结构

MoveJ([Motgrp/Mec_name], ToPoint, Speed, Zone, Tool, WObj, [Load])

指令参数

参数	说明
Motgrp/Mec_name	这个参数是可选参数，可以选择 Motgrp 赋值，也可以选择 Mec_name 赋值。若选择 Motgrp：0—左臂，1—右臂，如果不设置，则默认为左臂。若选择 Mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”，对于单个机器人系统，该参数通常省略。
ToPoint	数据类型 robtarget。机器人和外部坐标轴的目标点，定义为一个指定位置。
Speed	数据类型 speeddata。运动的速度数据。定义的工具中心点、工具方位调整和外轴的速度。
Zone	数据类型 zonedata。运动区域数据,描述生成的转弯路径的大小。
Tool	数据类型 tooldata。机器人移动时使用的 tool 数据。
WObj	数据类型 wobjdata。指定机器人运动的坐标系。
Load	数据类型 loaddata。可选参数，指定机器人移动中使用的总负载，即末端法兰盘上安装的工具负载加上该工具正在处理工件的有效负载。如果使用了本参数，那么就不再考虑当前 tooldata 中的 loaddata；如果省略掉本参数，就表示可以忽略处理工件的负荷。

指令示例

以下示例介绍了指令 MoveJ 指令的基本使用：

例 1 ：

工具 tool1 将关节运动至位置 P10，其速度数据为 v200，且经过 fine 点。

```
ROBTARGET("P10",{900,0,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0,-0},-0)
TOOLDATA("tool1",true,{{0,0,0},{1,-0,-0,-0}},{-0,{-0,-0,-0},{1,-0,-0,-0},-0,-0,-0,-0,-0,-0})
WOBJDATA("wobj1",false,true,"",{{-0,-0,-0},{1,-0,-0,-0}},{{-0,-0,-0},{1,-0,-0,-0}})
MoveJ(P10,v200,fine,tool1,wobj1)
```

例 2：

工具 tool2 将关节运动至位置 P20，其速度数据为 v500，且区域数据为 z10。

```
ROBTARGET("P20",{900,0,485},{0,0,1,0},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0,-0},-0)
TOOLDATA("tool2",true,{{0,0,0},{1,-0,-0,-0}},{-0,{-0,-0,-0},{1,-0,-0,-0},-0,-0,-0,-0,-0,-0})
WOBJDATA("wobj1",false,true,"",{{-0,-0,-0},{1,-0,-0,-0}},{{-0,-0,-0},{1,-0,-0,-0}})
MoveJ(P20,v500,z10,tool2,wobj1)
```

4.4 MoveAbsJ 关节绝对运动

指令功能

用于将机器人和外轴移动至轴位置中指定的绝对位置。使用 MoveAbsJ 运动期间，机器人的位置不会受到给定工具和工件以及有效程序位移的影响。机器人运用该数据，以计算负载、TCP 速度和拐角路径。可在邻近运动指令中使用相同的工具。机器人和外轴沿非线性路径运动至目标位置。所有轴均同时达到目标位置。

指令结构

MoveAbsJ([Motgrp/Mec_name], ToJointPos, Speed, Zone, Tool, WObj,[Load])

指令参数

参数	说明
Motgrp/Mec_name	可选参数，可以选择 Motgrp 作参数，也可以选择 Mec_name 作参数。若选择 Motgrp：0—左臂，1—右臂，如果不设置，则默认为左臂。若选择 Mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”，对于单个机器人系统，该参数通常省略。
ToJointPos	数据类型 robjoints。机器人和外部坐标轴的目标绝对位置。
Speed	数据类型 speeddata。运动的速度数据。定义的工具中心点、工具方位调整和外轴的速度。
Zone	数据类型 zonedata。运动区域数据,描述生成的角路径的大小。
Tool	数据类型 tooldata。机器人移动时使用的 tool 数据。该 tool 中心点是指向指定目标位置的点。
WObj	数据类型 wobjdata。用于指定机器人在坐标系中的工作位置。位置与坐标系有关，可以省略。如果使用固定 tool 参数或协同的外部轴，则此参数必须指定以执行相对于工作对象的线性运动。
Load	数据类型 loaddata。可选参数，指定机器人移动中使用的总负载，即末端法兰盘上安装的工具负载加上该工具正在处理工件的有效负载如果使用了本参数，那么就不再考虑当前 tooldata 中的 loaddata；如果省略掉本参数，就表示可以忽略处理工件的负荷。

指令示例

以下两个示例介绍了指令 MoveAbsJ 的基本用法：

例 1：

工具 tool1 将做关节绝对运动至位置 J10，其速度数据为 v200，且经过 fine 点。

```
JOINTTARGET("K10",{-0,19.592,25.504,-0,49.445,-0.001,-0},{0,0,0,0,0,0,0})
LOADDATA("Load_6kg",0,{74.3,0,0},{1,0,0,0},0.0418,0.0418,0.0102,0,0,0)
TOOLDATA("tool1",true,{{0,0,0},{1,-0,-0,-0}},{-0,{-0,-0,-0},{1,-0,-0,-0},-0,-0,-0,-0,-0,-0})
WOBJDATA("wobj1",false,true,"",{{-0,-0,-0},{1,-0,-0,-0}},{{-0,-0,-0},{1,-0,-0,-0}})
MoveAbsJ(K10,v200,fine,tool1,wobj1,Load_6kg)
```

例 2：

工具 tool2 做关节绝对运动至位置 J20，其速度数据为 v500，且区域数据为 z50。

```
JOINTTARGET("K20",{-0,19.592,25.504,-0,49.445,-0.001,-0},{0,0,0,0,0,0,0})
LOADDATA("Load_6kg",0,{74.3,0,0},{1,0,0,0},0.0418,0.0418,0.0102,0,0,0)
TOOLDATA("tool2",true,{{0,0,0},{1,-0,-0,-0}},{-0,{-0,-0,-0},{1,-0,-0,-0},-0,-0,-0,-0,-0,-0})
WOBJDATA("wobj1",false,true,"",{{-0,-0,-0},{1,-0,-0,-0}},{{-0,-0,-0},{1,-0,-0,-0}})
MoveAbsJ(K20,v500, z50,tool2,wobj1,Load_6kg)
```

4.5 ConfJOn, ConfJOff 关节运动期间监测配置

指令功能

用于确定在关节运动期间是否控制机器人的主轴配置参数，不设置时默认为开启监测。

ConfJOff 指令被调用后，在后续的关节运动期间将不再监测机器人运动指令编程的主轴参数。如果可以以若干种不同的方式到达该位置，它会寻找一种同机器人当前配置相同的主轴配置方案，并选择最接近第 4 轴和第 6 轴的配置参数。

ConfJOn 指令被调用后，后续的关节运动将再次开启对机器人主轴配置参数的监测。

指令结构

ConfJOn( )

ConfJOff( )

指令参数

无参数。

指令示例

ConfJOff( )

MoveJ (0, P10, v50, fine, tool0, wobj0)

机器人关节运动至编程的位置、方位参数，与机器人先前的主轴配置参数最为接近，但是可能与编程的主轴配置参数有所不同。

ConfJOn( )

MoveJ (0, P20, v50, fine, tool0, wobj0)

机器人关节运动至编程的位置、方位和主轴配置参数。如果不可能，则停止程序执行。

4.6 ConfLOn, ConfLOff 线性运动期间监测配置

指令功能

用于确定在线性运动期间是否控制机器人的主轴配置参数，不设置时默认为开启监测。

ConfLOff 指令被调用后，在后续的线性运动期间将不再监测机器人运动指令编程的主轴参数。如果可以以若干种不同的方式到达该位置，它会寻找一种同机器人当前配置相同的主轴配置方案，并选择最接近第 4 轴和第 6 轴的配置参数。

ConfLOn 指令被调用后，后续的线性运动将再次开启对机器人主轴配置参数的监测。

指令结构

ConfLOn( )

ConfLOff( )

指令参数

无参数。

指令示例

ConfLOff( )

MoveL (0, P10, v50, fine, tool0, wobj0)

机器人线性运动至编程的位置、方位参数，与机器人先前的主轴配置参数最为接近，但是可能与编程的主轴配置参数有所不同。

ConfLOn( )

MoveL (0, P20, v50, fine, tool0, wobj0)

机器人线性运动至编程的位置、方位和主轴配置参数。如果不可能，则停止程序执行。

4.7 CorrectOframe 修改工件坐标系

指令功能

修改工件坐标系中的物体坐标系，从而实现偏移、旋转。

指令结构

newWobj = CorrectOframe (Wobj, x, y, z, rx, ry, rz)

返回值

newWobj：返回的新的 wobj 值。

指令参数

参数	数据类型	说明
wobj	wobjdata	工件坐标系
x	num	X 方向偏移
y	num	Y 方向偏移
z	num	Z 方向偏移
rx	num	Rx 方向旋转
ry	num	Ry 方向旋转
rz	num	Rz 方向旋转

调用示例

local newWobj = CorrectOframe (wobj1, 100, 0, 0, 0, 0, 0)

在 wobj1 的基础上创建了一个新的 wobj 命名为 newWobj ， 该 wobj 的 oframe 是在 wobj0 的 oframe 基础上沿着 x 方向偏移 100 得到。

4.8 CreateRobT 创建新的 Robtarget 点

指令功能

基于已知的 RobTarget 点和末端位姿，创建新的 RobTarget 点。

指令结构

newRobT = CreateRobT (RobT, x, y, z, rx, ry, rz)

返回值

newRobT： 返回一个新的 RobTarget 点。

指令参数

参数	数据类型	说明
RobT	robtarget	已知的机器人位置数据
x	num	X 方向位置
y	num	Y 方向位置
z	num	Z 方向位置
rx	num	Rx 方向角度
ry	num	Ry 方向角度
rz	num	Rz 方向角度

调用示例

local newRobT = CreateRobT (P1, 100, 100, 100, 0, 0, 0)

在 P1 的基础上创建了一个新的 Robtarget 点，该 Robtarget 的关节配置、外部轴位置、臂形角和 P1 点保持一致，其末端的位置和姿态在 6 个方向分别为 100、100、100、0、0、0。

4.9 CorrectTool 修改工具坐标系

指令功能

修改工具坐标系，从而实现偏移、旋转。

指令结构

newTool = CorrectTool (Tool, x, y, z, rx, ry, rz)

返回值

newTool: 返回一个新的 tooldata。

指令参数

参数	数据类型	说明
Tool	tooldata	工具坐标系
x	num	X 方向偏移
y	num	Y 方向偏移
z	num	Z 方向偏移
rx	num	Rx 方向旋转
ry	num	Ry 方向旋转
rz	num	Rz 方向旋转

调用示例

local newTool = CorrectTool (tool1, 100, 0, 0, 0, 0, 0)

在 tool1 的基础上创建了一个新的 tool，该 tool 是在 tool0 的基础上沿着 x 方向偏移 100 得到。

4.10 CirPathMode 圆周路径期间的工具方位调整

指令功能

在圆周运动期间，对工具进行再定位。

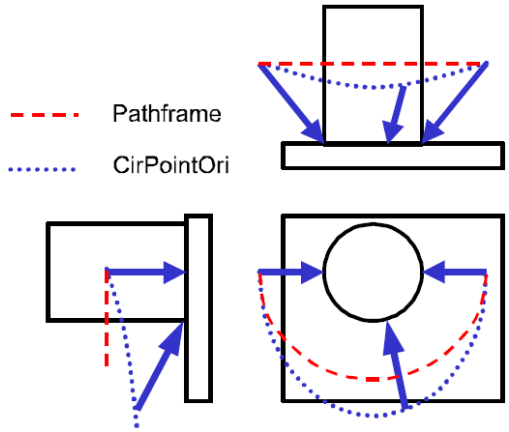
指令结构

CirPathMode(mode)

注意：

只对指令调用后的圆弧运动指令生效，当程序重新加载时，将恢复到系统标准的模式。

指令参数

参数	数据类型	说明
mode	num	mode 参数目前支持以下取值： PathFrame: 1，在圆周运动期间，从起点到 ToPoint，持续对工具实施再定位，中间过渡点姿态对姿态变化没有影响，这便是系统中的标准模式。 CirPointOri: 2，在圆周运动期间，从起点到 CirPoint 点，再到 ToPoint 点，持续对工具实施再定位。  箭头表明了适用于编程点、由腕中心点到工具中心点的工具。图中，腕中心点的不同路径以虚线表示。

调用示例

CirPathMode (1)

设置圆弧运动模式设置为经圆弧过渡点工具姿态持续变化，中间过渡点姿态对姿态变化没有影响，即系统标准。

4.11 GetAI 获取 AI 信号值

指令功能

GetAI 用于获取模拟输入信号的值。返回值范围与 IO 硬件模拟采样位数有关。

指令结构

val = GetAI (name)

返回值

val：返回 AI 的数值，整型值。

指令参数

参数	数据类型	说明
name	string	模拟输入信号的名称

指令示例

Value = GetAI(“AO10_0”)

获取 AO10_0 信号的数字值。

4.12 GetAO 获取 AO 信号值

指令功能

GetAO 用于获取模拟输出信号的值。返回值范围与 IO 硬件模拟采样位数有关。

指令结构

val = GetAO (name)

返回值

val：返回 AO 的数值，整型值。

指令参数

参数	数据类型	说明
name	string	模拟输出信号的名称

指令示例

Value = GetAO(“AO10_0”)

获取 AO10_0 信号的数字值。

4.13 GetDI 获取 DI 信号值

指令功能

GetDI 用于获取数字输入信号的值。电平值：1 为高电平，0 为低电平。

指令结构

val = GetDI (name)

返回值

val：返回 DI 的电平值。

指令参数

参数	数据类型	说明
name	string	数字输入信号的名称

指令示例

level = GetDI("DO10_6")

获得 DO10_6 电平值，保存在 level。

4.14 GetDO 获取 DO 信号值

指令功能

GetDO 用于获取数字输出信号的值。电平值：1 为高电平，0 为低电平。

指令结构

val = GetDO (name)

返回值

val：返回 DO 的电平值。

指令参数

参数	数据类型	说明
name	string	数字输出信号的名称

指令示例

level = GetDO("DO10_6")

获得 DO10_6 电平值，保存在 level。

4.15 SetAO 设置 AO 信号值

指令功能

SetAO 用于设置模拟输出信号的值。返回值范围与 IO 硬件模拟采样位数有关。

指令结构

SetAO (name, value)

指令参数

参数	数据类型	说明
name	string	模拟输出信号的名称
value	num	模拟输出信号的设定值

指令示例

SetAO(“AO10_0”, 1000)

设置模拟信号 AO10_0 数字值为 1000。

4.16 SetDO 设置 DO 信号值

指令功能

SetDO 用于设置数字输出信号的值。电平值：1 为高电平，0 为低电平。

指令结构

SetDO (name, level)

指令参数

参数	数据类型	说明
name	string	数字输出信号的名称
level	num	电平值

指令示例

SetDO("DO10_6",1)

设置 DO10_6 为高电平。

4.17 SetDI 设置 DI 信号值

指令功能

当打开信号仿真时，用于设置 DI 的值。

指令结构

SetDI (Signal,Value)

指令参数

参数	数据类型	说明
Signal	string	DI 信号字符串名称
Value	num	1/0

指令示例

SimInput(“DI_4”, true)

SetAI(“DI_4”,1)

SimInput(“DI_4”, false)

设置信号” DI_4”进入仿真模式，设置 DI 值为 1，设置完成后关掉信号仿真。

4.18 SetAI 设置 AI 信号值

指令功能

当打开信号仿真时，用于设置 AI 的值。

指令结构

SetAI (Signal,Value)

指令参数

参数	数据类型	说明
Signal	string	AI 信号字符串名称
Value	num	信号数值。

指令示例

SimInput(“AI_4”, true)

SetAI(“AI_4”,200)

SimInput(“AI_4”, false)

设置信号” AI_4”进入仿真模式，设置 AI 值为 200，设置完成后关掉信号仿真。

4.19 SimInput 设置 DI/AI 信号仿真开关

指令功能

设置一个 DI 或 AI 信号进入仿真状态，以设置 DI/AI 的值，用于仿真调试。

当打开信号仿真时，可以设置 DI/AI 的值，关闭后不能设置。

指令结构

SimInput(Signal,Flag)

指令参数

参数	数据类型	说明
Signal	string	信号字符串名称
Flag	bool	true：打开仿真； false：关闭仿真。

指令示例

SimInput(“DI_8”, true)
设置信号”DI_8”进入仿真模式。

4.20 ConnectDI 连接用户输入信号和系统输入信号

指令功能

ConnectDI 用于连接用户数字输入信号与系统数字输入信号。系统重启后，关联失效。

指令结构

ret = ConnectDI(signal, system_signal, trig_type)

返回值

ret： 0，设置不成功，1，设置成功。

指令参数

参数	数据类型	说明
signal	string	用户数字输入信号，在 EIO 中配置的 DI 信号。
system_signal	string	系统数字输入信号， 如”SYS_LOAD_PGM”,”SYS_START_PGM”, ”SYS_STOP_PGM”。 注意：”SYS_EM_STOP”不能连接。
trig_type	num	触发类型取值如下： TRIG_RISE_EDGE TRIG_FALL_EDGE TRIG_HIGH_LEVEL TRIG_LOW_LEVEL TRIG_LEVEL_CHANGE

指令示例

ConnectDI(“DI10_6”, “SYS_STOP_PGM”, TRIG_RISE_EDGE)

SetDI(“DI10_6”, 1)

DI10_6 信号产生上升沿，则系统信号 SYS_STOP_PGM 也会被触发。

4.21 ConnectDO 连接用户输出信号和系统输出信号

指令功能

ConnectDO 用于连接用户数字输入信号与系统数字输入信号。系统重启后，关联失效。

指令结构

ret = ConnectDO(signal, system_signal)

返回值

ret： 0，设置不成功，1，设置成功。

指令参数

参数	数据类型	说明
signal	string	用户数字输出信号，在 EIO 中配置的 DO 信号。
system_signal	string	系统数字输出信号， 如”SYS_MANUAL_MODE”,”SYS_AUTO_MODE”, ”SYS_MOTOR_STATE”。

指令示例

ConnectDO(“DO10_6”, “SYS_AUTO_MODE”)

当系统信号 SYS_AUTO_MODE 为 1 时，DO10_6 会被设置成 1；系统信号 SYS_AUTO_MODE 为 0 时，DO10_6 会被设置成 0.

4.22 AliasIO 设置 IO 信号别名

指令功能

AliasIO 把信号量和一个字符串别名关联， 在 IO 指令中可以通过该别名来访问信号量。

指令结构

AliasIO(Signal, Alias)

指令参数

参数	数据类型	说明
signal	string	IO 信号名称
alias	string	信号别名

指令示例

AliasIO ("da_doArcOn", "doArcOn")

Sleep(2000)

SetDO("doArcOn", 0)

将 DO 信号 “da_doArcOn”设置别名为"doArcOn"，通过设置"doArcOn"的值可以改变“da_doArcOn”的值。

4.23 AliasIOReset 重置 I/O 信号别名

指令功能

AliasIOReset 用于重置由 AliasIO 指令设置的信号别名。

指令结构

AliasIOReset(alias)

指令参数

参数	数据类型	说明
alias	string	信号别名，在先前指令中通过 AliasIO(signal, alias)设置的信号别名 alias。

指令示例

```
local alias = “doGrap”
AliasIO(“DO10_6”, alias)
SetDO(alias, 0)
AliasIOReset(alias)
```

将 DO 信号“DO10_6”设置别名为 “doGrap”，通过别名 “doGrap”可设置 DO 信号 “DO10_6”，通过 AliasIOReset 指令重置别名 “doGrap”，解除“DO10_6”与 “doGrap”的关联关系。

4.24 AliasIOResetAll 重置所有 I/O 信号别名

指令功能

AliasIOResetAll 用于重置所有由 AliasIO 指令设置的信号别名。

指令结构

AliasIOResetAll()

指令示例

AliasIO(“DO10_6”, “doGrap”)
AliasIO(“DO10_7”, “doPut”)
SetDO(“doGrap”, 1)
SetDO(“doPut”, 1)
AliasIOResetAll()
AliasIOResetAll()指令可以解除该指令之前所有信号与别名之间的关联关系。

4.25 GetGI 获取一组数字输入信号值

指令功能

GetGI 用于获取一组数字输入信号的值。

GI 信号可以设置包含最多 32 个数字输入信号，可通过 EIO 来配置。如果其中第一个 DI 为 1，其他信号为 0，则 GI_1 值为 1（2^0）。

第一个和第二个为 1，其他为 0，则 GI_1 值为 3（2^0+2^1）。

全部信号为 1，则 GI_1 值为 255（$2^0+2^1+2^2+2^3+2^4+2^5+2^6+2^7$）。

指令结构

val = GetGI (gi_name)

返回值

val：整型，代表一组输入信号的数值，其每个二进制位对应一个 DI。

指令参数

参数	数据类型	说明
gi_name	string	数字输入信号组的名称，通过在 EIO 中配置信号组名称及对应的 DI 信号组（通过起始 DI 和 DI 个数指定）。

指令示例

local value = GetGI("GI_1")

获得 GI_1 对应的数字输入信号值，保存到 value。

4.26 GetGO 获取一组数字输出信号值

指令功能

GetGO 用于获取一组数字输出信号的值。

GO 信号可以设置包含最多 32 个数字输出信号，可通过 EIO 配置 GO 信号。

指令结构

val = GetGO (go_name)

返回值

val：整型，代表一组输出信号的数值，其每个二进制位对应一个 DO。

指令参数

参数	数据类型	说明
go_name	string	数字输出信号组的名称，通过在 EIO 中配置信号组名称及对应的 DO 信号组（通过起始 DO 和 DO 个数指定）。

指令示例

local value = GetGO("GO_1")

获得 GO_1 对应的数字输入信号值，保存到 value。

4.27 SetGO 改变一组数字输出信号的值

指令功能

SetGO 用于改变一组数字输出信号的值。

GO 信号可以设置包含最多 32 个数字输出信号，可通过 EIO 配置 GO 信号。

指令结构

SetGO (go_name, value)

指令参数

参数	数据类型	说明
go_name	string	数字输出信号组的名称，通过在 EIO 中配置信号组名称及对应的 DO 信号组（通过起始 DO 和 DO 个数指定）。
value	整型	整数，如果信号个数为 n，则允许取值范围为[0, 2^n-1]。value 为 0 时所有 DO 信号都置为 0，value 为 2^n-1 时所有 DO 信号都为 1。

指令示例

SetGO("GO_1", 7)

把 GO_1 对应的前三个 DO 信号设置成 1，其他设置成 0。

SetGO(“GO_1”, 3)

把 GO_1 对应的第一个 DO 信号和第二个 DO 信号设置成 1，其余信号设置成 0。

4.28 WaitAI 等待单个 AI 信号被设置

指令功能

等待模拟输入信号被设置。当执行本指令时，如果信号值正确，则本程序仅仅继续以下指令。如果信号值不正确，则机器人进入等待状态，且当信号改变为正确值时，程序继续。

指令结构

ValueAtTimeout = WaitAI(name, relation,value,[MaxTime])

返回值

ValueAtTimeout：数据类型 num。如果指令超时，则将当前信号值储存在该变量中。

指令参数

参数	数据类型	说明
name	string	数字输入信号的名称
relation	num	条件参数 "LT"（小于）或者"GT"（大于）
value	num	模拟信号值
MaxTime	num	可选参数，允许的最长等待时间，单位：ms。设置参数值，程序超时则停止，并报错。 不设置该参数时程序一直等待。

指令示例

timeoutvalue = WaitAI("AI10_1", “LT”, 400, 500)

等待 AI10_1 数字值为小于 400,等待最大时间 0.5 秒，如果超时，则将程序运行停止，并在示教器显示超时信息。

4.29 WaitAO 等待单个 AO 信号被设置

指令功能

等待模拟输出信号被设置。当执行本指令时，如果信号值正确，则本程序仅仅继续以下指令。如果信号值不正确，则机器人进入等待状态，且当信号改变为正确值时，程序继续。

指令结构

ValueAtTimeout = WaitAO(name, relation,value, [MaxTime])

返回值

ValueAtTimeout：数据类型 num。返回值，如果指令超时，则将当前信号值储存在该变量中。

指令参数

参数	数据类型	说明
name	string	模拟输出信号的名称
relation	num	条件参数 "LT"（小于）或者"GT"（大于）
value	num	模拟信号值
MaxTime	num	可选参数，设置改参数表示允许的最长等待时间，单位：ms。如果超时，则程序停止，并报错。 不设置该参数时程序停止并一直等待。

指令示例

timeoutvalue = WaitAO("AO10_1", “LT”, 400, 500)

等待 AO10_1 数字值为小于 400,等待最大时间 0.5 秒，如果超时，则程序停止运行，并在示教器显示等待超时。

4.30 WaitDI 等待单个 DI 信号被设置

指令功能

等待数字输入信号被设置。当执行本指令时，如果信号值正确，则本程序仅仅继续以下指令。如果信号值不正确，则机器人进入等待状态，且当信号改变为正确值时，程序继续。

指令结构

TimeOutFlag = WaitDI (name,level,[MaxTime], [TimeFlag])

返回值

TimeOutFlag：数据类型 bool。 超出时间时设置为 true，否则，其将设置为 false。

指令参数

参数	数据类型	说明
name	string	数字输入信号的名称
level	num	电平值
MaxTime	num	可选参数，允许的最长等待时间，单位：ms。
TimeFlag	bool	可选参数，使用该参数基于使用 MaxTime。**如果在**满**足条件之前**超出该时间，**且** TimeFlag 为 false，则程序停止，并提示超时。如果 TimeFlag 为 true，则超时发生时，程序不停止，进行执行。 该参数不填时，**与** TimeFlag 为 false 时执行效果相同。

指令示例

flag = WaitDI("DI10_1", 1,500)

等待 DI10_1 为高电平,等待最大时间 0.5 秒，如果超时，则 flag 为 true, 否则 flag 为 false。如果超时，则程序停止，并提示超时。

4.31 WaitMultiDI 等待多个 DI 信号被设置

指令功能

等待多个数字输入信号被设置。当执行本指令时，如果所监测的信号值符合设置的条件，则本程序仅仅继续以下指令。如果信号值不满足条件，则机器人进入等待状态，且当信号改变满足所设置的条件时，程序继续。

指令结构

TimeOutFlag = WaitMultiDI (value1,value2，…, relation,MaxTime)

返回值

TimeOutFlag：数据类型 bool。 等待超出时间时返回 true，否则，返回 false。

指令参数

参数	数据类型	说明
value1,value2，…	string	数字输入信号的名称，格式为"DI10_1=1"，"DI10_2=0"
relation	num	条件参数: OR，或，当所设的指令有一个或一个上满足设置的条件，程序继续执行。 AND，与，当所设的指令都满足设置的条件，程序继续执行。
MaxTime	num	允许的最长等待时间，单位：ms

指令示例

flag = WaitMultiDI("DI10_1=1","DI10_2=0","and", 300)

等待 DI10_1 信号值为 1 和 DI10_2 信号值为 0，等待最大时间 0.3 秒，如果超时，则 flag 为 true, 否则 flag 为 false。

flag = WaitMultiDI("DI10_1=1","DI10_2=0","OR", 0.3)

等待 DI10_1 信号值为 1 或 DI10_2 信号值为 0，有一个值满足条件时则继续执行。等待最大时间 0.3 秒，如果超时，则 flag 为 true, 否则 flag 为 false。

4.32 WaitDO 等待直至已设置数字信号输出信号

指令功能

等待数字输出信号被设置。当执行本指令时，如果信号值正确，则本程序仅仅继续以下指令。如果信号值不正确，则机器人进入等待状态，且当信号改变为正确值时，程序继续。

指令结构

ret = WaitDO (name,level,[MaxTime],[TimeFlag])

返回值

ret：数据类型 bool。返回值，可选参数，超出时间时设置为 true，否则，其将设置为 false。

指令参数

参数	数据类型	说明
name	string	数字输入信号的名称
level	num	电平值
MaxTime	num	可选参数，允许的最长等待时间，单位：ms。
TimeFlag	bool	可选参数，该参数结合 MaxTime 来使用。**如果在**满**足条件之前**超出该时间，**且** TimeFlag 为 false，则程序停止，并提示超时。如果 TimeFlag 为 true，则超时发生时，程序不停止，继续执行。 该参数不填时，**与** TimeFlag 为 false 时执行效果相同。

指令示例

flag = WaitDO("DO10_1", 1,500)

等待 DO10_1 为高电平,等待最大时间 0.5 秒，如果超时，则 flag 为 true, 程序停止，并提示超时，否则 flag 为 false。

4.33 PulseDO 设置 DO 脉冲信号

指令功能

PulseDO 产生数字输出信号的脉冲。

指令结构

PulseDO(do_name,length,high)

指令参数

参数	数据类型	说明
do_name	string	数字输出信号的名称
length	int	脉冲信号的时间长度，单位为毫秒 值范围：（1-2000000）
high	bool	是否始终将脉冲信号设置为高电平； 当 high=true 时，脉冲信号始终为高电平； 当 high=false 时，脉冲信号为原信号的反转电平。

指令示例

PulseDO("DO1",10,false)

反转当前 DO1 信号的电平，持续时间为 10 毫秒，10 毫秒后恢复 DO1 电平。

PulseDO("DO2",2555,true)

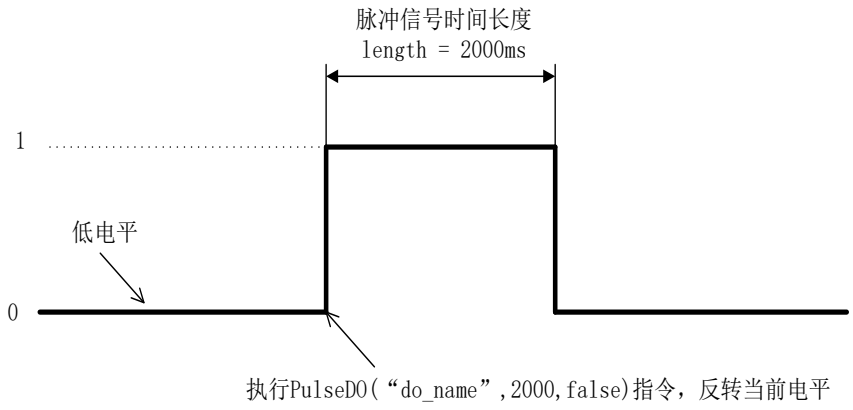
无论当前 DO2 信号电平，始终将 DO2 信号置为高电平 2555 毫秒，2555 毫秒后将 DO2 设置为低电平。

指令执行说明

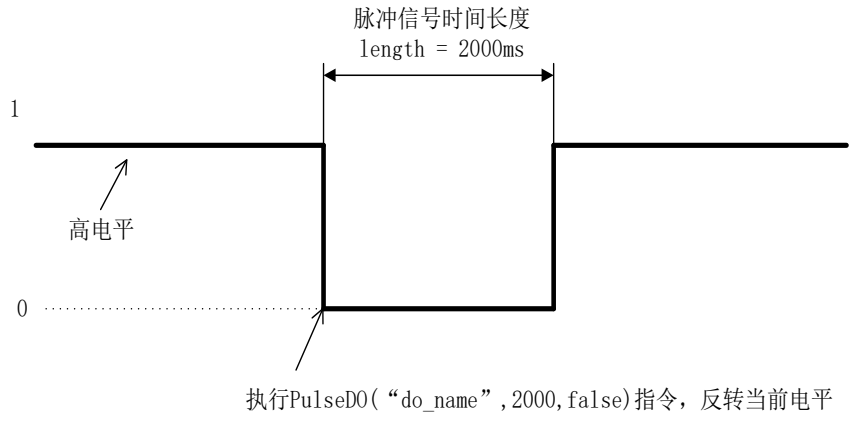
在产生脉冲信号后，直接执行 PulseDO 后的下一条指令。可在不影响程序执行的情况下，设置/重置脉冲。

关于设置 DO 脉冲信号的实例如下。

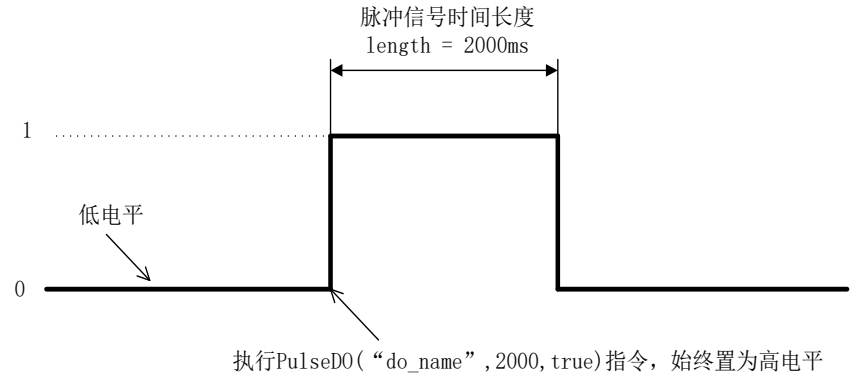
1 当信号”do_name”为低电平。设置参数 length 为 2000 毫秒，参数 high 设置为 false，执行 PulseDO 指令，产生的脉冲信号如下：



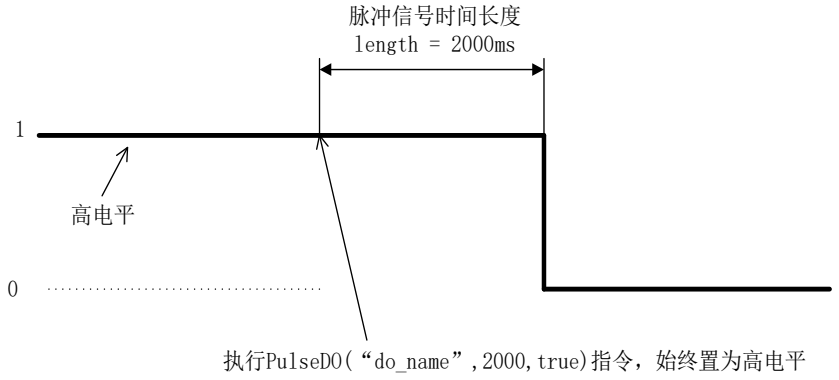
2 当信号”do_name”为高电平。设置参数 length 为 2000 毫秒，参数 high 设置为 false，执行 PulseDO 指令，产生的脉冲信号如下：



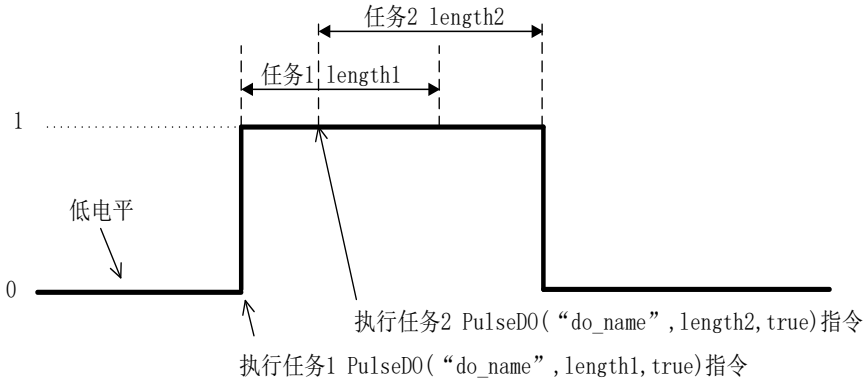
3 当信号”do_name”为低电平。设置参数 length 为 2000 毫秒，参数 high 设置为 true，执行 PulseDO 指令，产生的脉冲信号如下：



4 当信号”do_name”为高电平。设置参数 length 为 2000 毫秒，参数 high 设置为 true，执行 PulseDO 指令，产生的脉冲信号如下：



5 当两个任务设置同一个脉冲信号”do_name”时，产生的脉冲信号如下：



4.34 WaitGI 等待直至数字输入组信号满足指定条件

指令功能

WaitGI 用于等待，直至数字输入组信号满足特定条件。

指令结构

ret = WaitGI(gi_name, relation, expected_value,[timeout])

返回值

ret：数据类型 bool。满足返回 true，其它返回 false。

指令参数

参数	数据类型	说明
gi_name	string	数字输入信号组的名称，通过在 EIO 中配置信号组名称及对应的 DI 信号组（通过起始 DI 和 DI 个数指定）。
relation	string	字符串形式的关系运算符。 “EQ”相等，”GT”大于，”LT”小于， “NOEQ” 不相等, “GTEQ”大于等于, “LTEQ”小于等于
expected_value	num	整数，期望值。3 即前两个信号为 1，7 即前三个信号为 1，以此类推。
timeout	num	可选参数，最大等待时间，单位毫秒。如果 timeout 为空或者 0，则一直等待，直至信号满足条件。如果 timeout 大于 0，等待时间超过最大等待时间，则程序停止，并报错。

指令示例

WaitGI("GI_1", “EQ”, 7)

一直等待，直至 GI_1 对应的前三个 DI 信号都为 1。

WaitGI("GI_1", “GT”, 7, 2000)

等待，直至第四个及之后的信号有一个为 1。如果在 2000ms 内信号不符合要求，则系统报错。

4.35 WaitGO 等待直至数字输出组信号满足指定条件

指令功能

WaitGO 用于等待，直至数字输出组信号满足特定条件。

指令结构

ret = WaitGO(go_name, relation, expected_value, [timeout])

返回值

ret： 数据类型 bool。

满足返回 true，否则返回 false。

指令参数

参数	数据类型	说明
go_name	string	数字输出信号组的名称，通过在 EIO 中配置信号组名称及对应的 DO 信号组（通过起始 DO 和 DO 个数指定）。
relation	string	字符串形式的关系运算符。 “EQ”相等，”GT”大于，”LT”小于， “NOEQ” 不相等, “GTEQ”大于等于, “LTEQ”小于等于
expected_value	num	整数，期望值。3 即前两个信号为 1，7 即前三个信号为 1，以此类推。
timeout	num	可选参数，最大等待时间，单位毫秒。如果 timeout 为空或者 0，则一直等待，直至信号满足条件。如果 timeout 大于 0，等待时间超过最大等待时间，则程序停止，并报错。

指令示例

WaitGO("GO_1", “EQ”, 7)

一直等待，直至 GO_1 对应的前三个 DO 信号都为 1。

WaitGO("GO_1", “GT”, 7, 2000)

等待，直至第四个及之后的信号有一个为 1。如果在 2000ms 内信号不符合要求，则系统报错。

4.36 TriggIO 定义停止点附近的固定位置或时间 I/O 事件

指令功能

TriggIO 用于定义有关设置机器人运动路径沿线固定位置处的数字或模拟信号输出信号的条件和行动。如果停止点附近的 I/O 设置需要较高精度，则应始终使用 TriggIO。

指令结构

event = TriggIO (Distance,Start,Time,DO/AO,setValue,DODelay)

返回值

event ：数据类型 TriggData 数据结构。

用于储存 triggdata 的变量，随后将此类 triggdata 用于 TriggL、TriggC 或 TriggJ 指令。

指令参数

参数	数据类型	说明
Distance	num	定义在路径上应出现 I/O 事件的位置
Start	bool	true: Distance 始于起点 false: Distance 提前于结束点
Time	bool	Distance 的指定值为时间或距离 true：Distance 值为时间，单位：ms。时间事件只相对于结束点生效，最大值为 250ms。Start 值设置为 false。 false：Distance 值为距离，单位：mm
DO/ AO	signaldo	DO/AO/GO 信号名称
SetValue	num	信号的期望值
DODelay	num	机械臂已达到指定位置后延时输出信号, 单位：ms

指令示例

Event1 = TriggIO(5, false, false, "DORedLight", 1, 0.3)

距离结束点 5mm 处触发 DORedLight 置一。

Event2 = TriggIO(20, false, false, "DOGreenLigh", 1, 0.3)

距离结束点 20mm 处触发 DOGreenLigh 置一。

4.37 TriggCheckIO 定义位于固定位置的 I/O 检查

指令功能

TriggCheckIO 用于定义有关测试机器人运动路径沿线固定位置处的数字或模拟的输入或输出信号的条件。如果本条件得以满足，则将不会存在特定行动。否则，在机器人已尽快在路径上随意停止后，将运行中断程序。

指令结构

event = TriggCheckIO (Distance, Start, Time,Signal, Relation, CheckValue / CheckDvalue , StopMove，Interrupt)

返回值

event : 数据类型 TriggData 数据结构。

用于储存 triggdata 的变量，随后将此类 triggdata 用于 TriggL、TriggC 或 TriggJ 指令。

指令参数

参数	数据类型	说明
Distance	num	
Start	bool	关于参数 Distance 的距离始于移动起点而非终点时使用
Time	bool	true：Distance 值为时间，单位：ms。 时间事件只相对于结束点生效。在机械臂达到本指令终点前，以时间表示的固定位置 I/O 仅可用于短时间（<0.25 s）。 false：Distance 值为距离，单位：mm。
Signal	signalxx	将测试的信号的名称，DI/DO/AI/AO/GI/GO
Relation	opnum	规定如何将信号的实际值与参数 CheckValue 定义的值进行比较
CheckValue/ CheckDvalue	num/dnum	与输入或输出信号的实际值进行比较的值
StopMove	bool	规定如果未满足条件，则在运行中断程序之前，机器人将在路径上尽快停止
Interrupt	intnum	用于确定运行中断程序的变量

指令示例

```
local function errprocess()
    SetDO("DO10_6",1)
end
local trap_errprocess = 1
RegisterTrap (trap_errprocess, errprocess)
BVoltage = {}
BVoltage=TriggCheckIO(50,false, false, AIBWeldingVoltage, "GT", AIMaxVoltage, true, trap_errprocess)
```

定义中断函数 errprocess（），中断号为 trap_errprocess。定义 TriggCheckIO，距离结束点 50mm 处判断 AI AIBWeldingVoltage 是否大于 AIMaxVoltage，如果判断条件成立则机械臂在路径上停止，并运行中断程序 errprocess（）。

4.38 TriggEquip 定义路径上的固定位置和时间 I/O 事件

指令功能

TriggEquip 用于定义设置机械臂移动路径沿线固定位置处的数字、数字组或模拟信号输出的条件和行动以及对外部设备中的滞后时间进行补偿的可能性。如果停止点附近的 I/O 设置需要较高精度，则应始终使用 TriggIO。

指令结构

event = TriggEquip(Distance, Start, EquipLag, IO, Value, Inhib)

返回值

event : 数据类型 TriggData 数据结构。。

用于储存 triggdata 的变量，随后将此类 triggdata 用于 TriggL、TriggC 或 TriggJ 等指令。

指令参数

参数	数据类型	说明
Distance	num	定义在路径上应出现 I/O 事件的位置
Start	bool	true: Distance 始于起点 false: Distance 提前于结束点
EquipLag	num	指定外部设备的滞后时间，单位：ms。 用于外部设备滞后补偿，使用正或负参数值。正参数值意味着 I/O 信号由 TCP 达到相对于移动起点或终点的指定距离前一指定时间。 负参数值意味着 I/O 信号由 TCP 已通过相对于移动起点或终点的指定距离后一指定时间。
IO	string	DO/GO/AO 信号名称
Value	num	信号的期望值
Inhib	bool	用于约束运行时信号的设置。 true：指定信号设置为 0。

指令示例

local Event1 = TriggEquip(50, true, 50, "DO10_8", 1, false)

距离起点 50mm 处触发 DO10_8 置 1，信号提前输出补偿时间为 50ms。

4.39 TriggSpeed 定义与固定位置、时间尺度事件成比例的 TCP 速度模拟信号输出

指令功能

TriggSpeed 用于定义模拟信号输出条件和行动，其输出值与实际 TCP 速度成比例。

可在沿机械臂移动路径的固定位置和时间处，指定模拟信号输出跟随速度的变化而改变输出值。针对外部设备中的滞后、模拟信号输出的开始、缩放和结束跟随机械臂的速度而变化，并可使用时间补偿。

指令结构

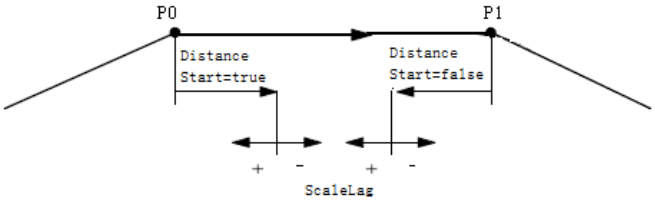
event = TriggSpeed(Distance, Start,ScaleLag,Ao, ScaleValue, DipLag, ErrDo, Inhib, Toplimit, Lowlimit)

返回值

event ：数据类型 TriggData 数据结构。

用于储存 triggdata 的变量，随后将此类 triggdata 用于 TriggL、TriggC 或 TriggJ 指令。

指令参数

参数	数据类型	说明
Distance	num	定义为距移动路径起点或终点的距离，以改变模拟信号输出值。单位：mm（正值）。
Start	bool	定义参数 Distance 的距离始于移动起点或终点时使用。 true：起点；false：终点。
ScaleLag	num	指定滞后外部设备中的时间，单位：ms（正值）。 
Ao	string	模拟输出信号的名称。
ScaleValue	num	模拟信号输出的比例刻度值。 计算关于模拟信号的实际输出值：实际输出值 = 比例刻度值 * 实际 TCP 速度（mm/s）。模拟信号输出范围：0 – 4095。
DipLag	num	由于机械臂速度下降，因此，当改变模拟信号输出值时，指定滞后作用于外部设备的时间，单位：ms（正值）。

ErrDo	string	用于报告模拟值溢出的数字输出信号的名称。 如果在移动期间，参数 DO 中定义的信号模拟输出值计算结果因超速而产生溢出，则设置该信号，并将实际模拟信号输出值降低至最大值。如果不再存在溢出，则重置信号。
Inhib	bool	用于约束运行时模拟信号设置的永久变量标志的名称。 false：设置 AO 信号跟随 tcp 速度变化。 true：指定信号 AOp 设置为 0 而非计算出的值。

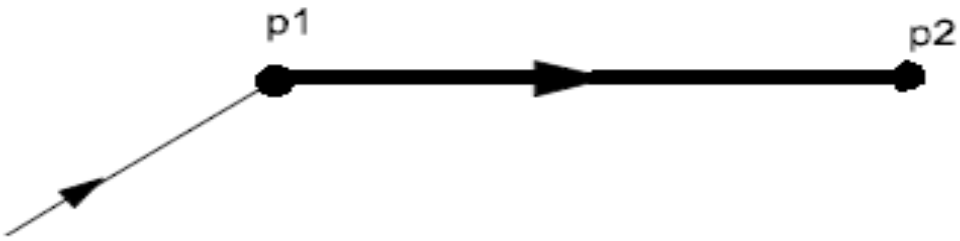
注意

如果程序紧急停止，则本过程将立即把模拟信号输出设置为 0。如果程序停止，则模拟信号输出信号将始终与 TCP 速度成比例，直至机械臂保持静止。指令运行暂停过程中始终“启用”，并准备重启。当机械臂重启时，信号从一开始便直接与 TCP 速度成比例。

指令示例

```
ts = {}
ts = TriggSpeed(10, true, 0.05, AoLsPower, 40, 0.01, "DO10_8", false)
MoveL(P1,v100,fine, tool0,wobj0,load0)
TriggL(P2, v100, ts, fine, tool0, wobj0, load0)
```

当 TCP 位于点 p1 后 10 mm 加 0.05 s 时，启用刻度值 40，向模拟信号 AoLsPower 输出 TCP 速度*40 的值。模拟信号 AoLsPower 输出在 TCP 速度出现下降前 0.01 s 受到影响。



4.40 TriggJ 基于事件的关节运动

指令功能

当无须以线性移动时，机器人以关节插补迅速从一点移动至另一点；同时，在大致固定位置设置输出信号和/或运行中断程序。

指令结构

TriggJ(MecName, ToPoint, Speed, Trigg_1/TriggArray,Zone, Tool, Wobj, TLoad)

指令参数

参数	数据类型	说明
MecName	string	机械单元名
ToPoint	robtarget	目标点
Speed	speeddata	运动的速度数据
Trigg_1/TriggArray	triggdata	trigg 变量或数组,数组最多 8 个
Zone	zonedata	相关移动的区域数据
Tool	tooldata	当机器人移动时工具
WObj	wobjdata	机器人位置关联的工件（坐标系）
TLoad	loaddata	移动中使用的总负载

指令示例

event11 = TriggIO(5, false, false, "DORedLight", 1, 0)

TriggJ(0, P10, v200, event11, z10, tool0, wobj0, load0)

到达位置 P10 前 5mm 处拉高 DO 信号"DORedLight"。

4.41 TriggL 基于事件的线性运动

指令功能

当机器人以直线插补移动时，TriggL（TriggLinear）用于设置输出信号和/或在固定位置运行中断程序。

指令结构

TriggL(MecName,ToPoint,Speed,Trigg_1/TriggArray,Zone, Tool,Wobj,TLoad)

指令参数

参数	数据类型	说明
MecName	string	机械单元名
ToPoint	robtarget	目标点
Speed	speeddata	运动的速度数据
Trigg_1/TriggArray	triggdata	trigg 变量或数组,数组最多 8 个
Zone	zonedata	相关移动的区域数据
Tool	tooldata	当机器人移动时工具
WObj	wobjdata	机器人位置关联的工件（坐标系）
TLoad	loaddata	移动中使用的总负载

指令示例

event11 = TriggIO(5, false, false, "DORedLight", 1, 0)

TriggL(0, P10, v200, event11, z10, tool0, wobj0, load0)

到达位置 P10 前 5mm 处拉高 DO 信号"DORedLight"。

4.42 TriggC 基于事件的圆弧运动

指令功能

当机器人正在圆弧路径上移动时，TriggC（Trigg Circular）用于设置输出信号和/或在固定位置运行中断程序。

指令结构

TriggC(MecName,CirPoint,ToPoint,Speed,Trigg_1/TriggArray,Zone, Tool,Wobj,TLoad)

指令参数

参数	数据类型	说明
MecName	string	机械单元名
CirPoint	robtarget	机器人圆弧点
ToPoint	robtarget	目标点
Speed	speeddata	运动的速度数据
Trigg_1/TriggArray	triggdata	变量
Zone	zonedata	相关移动的区域数据
Tool	tooldata	目标点
WObj	wobjdata	机器人位置关联的工件（坐标系）
TLoad	loaddata	移动中使用的总负载

指令示例

event11 = TriggIO(5, false, false, "DORedLight", 1, 0)

TriggC(P10,P20,v200,event11,fine,tool0,wobj0,load0)

到达位置 P10 前 5mm 处拉高 DO 信号"DORedLight"。

4.43 Sleep 用于等待给定的时间

指令功能

用于等待给定的时间。将进程挂起一段时间。

指令结构

Sleep (time_ms)

函数参数

参数	数据类型	说明
time_ms	num	休眠或者挂起的时间，单位：ms。

指令示例

Sleep(500)

休眠 500ms

4.44 GetJointTarget 获取当前关节位置

指令功能

获取指定机器人当前的关节位置数据，单位为角度。

指令结构

val = GetJointTarget (motgrp)

返回值

val：返回一个 JOINTTARGET 数据结构。

指令参数

参数	数据类型	说明
motgrp	string	机械单元名称

调用示例

local databody = GetJointTarget ("ROL_L")

print(databody.name)

print(databody.robax.rax_1,databody.robax.rax_2,databody.robax.rax_3,databody.robax.rax_4,databody.robax.rax_5,databody.robax.rax_6,databody.robax.rax_7)

获取机械单元 ROL_L 的位置并打印出来

4.45 GetRobTarget 获取当前机器人位置

指令功能

获取指定机器人当前的位置数据。

指令结构

val = GetRobTarget ([motgrp],tool,wobj)

返回值

val：返回一个 RobTarget 数据结构。

指令参数

参数	数据类型	说明
Motgrp	string	可选参数，机械单元名称
tool	tooldata	使用的工具坐标系
wobj	wobjdata	工件坐标系

调用示例

```
local databody = GetRobTarget("ROL_L", tool0, wobj0)
print(databody.name)
print(databody.trans.x,databody.trans.y,databody.trans.z)
```

获取变量 ROL_L 的位置并打印出来

4.46 MPointArray 创建点位数组

指令功能

创建一个 RobTarget 点位数组。

指令结构

array = MPointArray(arg1,arg2,arg3, ...)

返回值

array：数组，一个由 Robtarget 组成的数组。

指令参数

参数	数据类型	说明
arg1,arg2,arg3, ...	RobTarget	RobTarget 点位数据

指令示例

local array1 = MPointArray(P10,P20,P30)

在示教器上添加“点位数组定义”指令，输入需要插入的点位个数，示例中为 3，指令中插入 3 个 RobTarget 点，合并成 array1 作为返回值 。

4.47 Offs 对指定点位进行偏移

指令功能

在指定的工件坐标系中对指定点位添加一个偏移量。

指令结构

val = Offs(point, XOffset, YOffset, ZOffset)

返回值

val：返回一个 RobTarget 数据结构。

指令参数

参数	数据类型	说明
point	robtarget	待移动的位置数据
XOffset	num	工件坐标系中 x 方向的位移
YOffset	num	工件坐标系中 y 方向的位移
ZOffset	num	工件坐标系中 z 方向的位移

指令示例

MoveL(Offs(P10, 0, 0, 10), v500, z10, tool2, wobj1)

P10 沿 Z 轴方向移动 10 个单位，速度为 V500。

4.48 SearchL 机器人沿直线进行搜索

指令功能

当机器人以直线插补移动时，SearchL（SearchLinear）用于搜寻监测信号并在搜到信号后机器人立即读取当前位置并执行下一条指令。SearchL 和其上一条运动指令执行的都是 fine 点。

指令结构

ret = SearchL([Motgrp/Mec_name], DI, Flag, SearchPoint, ToPoint, Speed, Tool, WObj, [Load],[StopType])

返回值

ret： 数据类型 bool。搜寻到信号返回 true，否则返回 false。

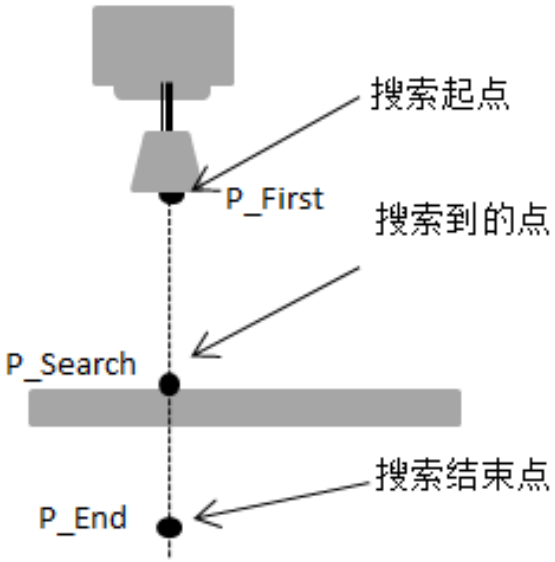
指令参数

参数	数据类型	说明
Motgrp/Mec_name	string	这个参数是可选参数，可以选择 Motgrp 赋值，也可以选择 Mec_name 赋值。若选择 Motgrp：0—左臂，1—右臂。如果不设置，则默认为左臂。若选择 Mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”，对于单个机器人系统，该参数通常省略。
DI	string	要监测的信号量。
Flag	num	监测信号为 Flag 的值时有效，有效值为 0 或 1，其他值无效
SearchPoint	robtarget	监测信号有效时的记录点，该点位将在搜寻到信号时被修改。
ToPoint	robtarget	目标点。
Speed	speeddata	运动的速度数据。
Tool	tooldata	当机器人移动时工具。
WObj	wobjdata	机器人位置关联的工件（坐标系）。
TLoad	loaddata	移动中使用的总负载，可缺省。

StopType	num	停止类型。 0， 硬停止，监测信号有效时机器人会尽快停止动作，不会保持 TCP 在路径上，机器人在停止前会移动很小的一段距离； 1， 1，软停止，监测信号有效时机器人会尽快停止移动同时保持 TCP 靠近或在路径上，机器人在停止前会移动一小段距离。参数可缺省，缺省值为 0，硬停止。

指令示例

以下为典型的搬运示例：




```
local bFirstTime = true
while true do
    if bFirstTime == true then
        MoveL(P_First, v500, z10, tool0, wobj0)
        bFirstTime = false
    else
        MoveL(P_Search, v500, z10, tool0, wobj0)
    end
    local ret = SearchL(“DI10_4”, 0, P_Search, P_End, v50, tool0, wobj0, load0)
    if ret == true then
        SetDO("DO_Grab", 1)
    else
        HandleErr()
    end
```

```
end
```

说明

程序开始时会判断是否为第一次搜寻，若为第一次搜寻则移动至 P_First 点，同时将第一次搜寻的标志 bFirstTime 置为 false，否则将移动至上一次搜寻到达点。

机器人到位后往 P_End 点运行过程中搜寻“DI10_4”上的信号变为 0，若搜寻到信号则将"DO_Grab"信号置为 1， 进行抓取， 否则执行用户自定义的错误处理程序。

4.49 RelTool 相对于工具进行位移

指令功能

RelTool（Relative Tool）用于将通过有效工具坐标系表达的位移和/或旋转增加至机械臂位置。

指令结构

newtarget = RelTool(target,dx, dy, dz, Rx, Ry, Rz)

返回值

newtarget：返回一个新的 robtaget 数据结构。

指令参数

参数	类型	说明
target	robtarget	输入机械臂位置。该位置的方位规定了工具坐标系的当前方位。
dx	num	工具坐标系 x 方向的位移，单位：mm。
dy	num	工具坐标系 y 方向的位移，单位：mm。
dz	num	工具坐标系 z 方向的位移，单位：mm。
Rx	num	围绕工具坐标系 x 轴的旋转，单位：度。
Ry	num	围绕工具坐标系 y 轴的旋转，单位：度。
Rz	num	围绕工具坐标系 z 轴的旋转，单位：度。

注意

如果同时指定两次或三次旋转，则将首先围绕 x 轴旋转，随后围绕新的 y 轴旋转，然后围绕新的 z 轴旋转。

指令示例

local newtarget = RelTool (p1, 10, 0, 0, 10, 0 , 0)

p1 点 TCP 沿工具坐标系移动 10mm，并绕工具坐标系 x 轴旋转 10 度，生成点 newtarget。

4.50 RegisterTrap 注册中断

指令功能

注册中断。中断服务程序中只能做最简单的逻辑运算或者 IO 操作，不能进行运动等动作。

指令结构

RegisterTrap(trap_num, trap_routine)

指令参数

参数	数据类型	说明
trap_num	num	中断号
trap_routine	string	中断服务程序名称

指令示例

trap_errprocess = 1

RegisterTrap (trap_errprocess, “errprocess”)

将子程序 errprocess（）注册为中断，中断号为 1。

4.51 SetAcc 设置加速度

指令功能

SetAcc 用于设置 TCP 运动的加速度。

指令结构

SetAcc (max_acc_defined, max_acc, dacc, ddec)

指令参数

参数	数据类型	说明
max_acc_defined	bool	若此参数为 true，表示限定 TCP 运动的最大加速度。 若此参数为 false，表示不限定 TCP 运动的最大加速度。
max_acc	num	此参数给出了限定 TCP 运动的最大加速度值，仅当第一参数为 true 时生效。范围：0~10000。
dacc	num	表示 TCP 运动加加速度占正常值的比率，最大值为 1.0。
ddec	num	表示 TCP 运动减加速度的增速占正常值的比率，最大值为 1.0。

指令示例

```
JOINTTARGET("K10",{-0,-0,-0,-0,-0,-0,-0},{-0,-0,-0,-0,-0,-0,-0})
TOOLDATA("tool1",true,{{-0,-0,-0},{1,-0,-0,-0}},{-0,{-0,-0,-0},{1,-0,-0,-0},-0,-0,-0,-0,-0,-0})
WOBJDATA("wobj1",false,true,"",{{-0,-0,-0},{1,-0,-0,-0}},{{-0,-0,-0},{1,-0,-0,-0}})
LOADDATA("Load_6kg",6,{74.3,0,0},{1,0,0,0},0.0418,0.0418,0.0102,0,0,0)
SetAcc(true, 1000, 0.1, 0.1)
MoveAbsJ(0, K10,v500,z50,tool1,wobj1,Load_6kg)
```

设置加速度，TCP 最大加速度为 1000m/s2,加加速度为 0.1，减加速度为 0.1。

4.52 PathAccLim 降低路径沿线的 TCP 加速度

指令功能

PathAccLim 用于限制沿运动路径方向的 TCP 加速度或减速度限值 。对沿运动路径的加速度进行限制，路径方向中的加速度或减速度将会受到限制。

PathAccLim 未限制设备的总加速度，即世界坐标系中的加速度，因此，无法直接用于保护设备，以免出现较大的加速度。

指令结构

PathAccLim(acclim, max_acc, declim, max_dec)

指令参数

参数	数据类型	说明
acclim	bool	如果加速度存在限制，则为 true，否则为 false。
max_acc	num	加速度限制的绝对值，以 m/s2 计。仅当 acclim 为 true 时生效。范围：大于 0，建议 0~100 之间。
declim	bool	如果减速度存在限制，则为 true，否则为 false。
max_dec	num	减速度限制的绝对值，以 m/s2 计。仅当 declim 为 true 时生效。范围：大于 0，建议 0~100 之间。

指令示例

PathAccLim(true, 50, false, 10)

运动路径方向上加速度限制为 50。

4.53 MotionSup 运动监控

指令功能

用于在程序执行期间，停用或启用机械臂的运动监控功能。

指令结构

MotionSup(switch,sensitivity)

指令参数

参数	数据类型	说明
switch	bool	是否开启路径监控功能，开启功能设置为 true，否则为 false。
sensitivity	num	灵敏度调试比例，取值范围 0-100。数值越大，灵敏度越低。

指令示例

MotionSup(true,90)

打开路径监控，设置灵敏度调试比例为 90.

4.54 SocketCreate 创建新套接字

指令功能

SocketCreate 用于创建一个 server 端。如果连接成功，返回 true；如果连接不成功，返回 false。

指令结构

ret = SocketCreate(serverName, ip, port)

返回值

ret： 数据类型 bool。返回状态，创建成功返回 true；创建失败返回 false。

指令参数

参数	数据类型	说明
serverName	string	要建立连接的服务端套接字名
ip	string	需要绑定的 IP 地址。可以输入 nil 表示不指定 IP，客户端可以通过控制器的 IP 访问到改服务。
port	num	port 数

指令示例

```
local serverName =  “SSocket”
local port = 5062
local err = SocketCreate(serverName, nil, port)
    --连接失败，处理
if false == err then
    print(“ create failed !”)
    return false
end
```

4.55 SocketClose 关闭套接字

指令功能

SocketClose 用于关闭一个套接字。如果连接成功，返回 true；如果连接不成功，返回 false。

指令结构

ret = SocketClose (socketName)

返回值

ret：数据类型 bool。返回状态，如果成功返回 true；如果失败则返回 false。

指令参数

参数	数据类型	说明
socketName	string	要建立连接的服务端套接字名

指令示例

```
local err = SocketClose (serverName)
    --连接失败，处理
if false == err then
    print(“ create failed !”)
    return false
end
```

4.56 SocketAccept 开始监听来自于客户端的连接。

指令功能

SocketAccept 用于开始监听输入连接。等待客户请求到来；当请求到来后，接受连接请求，返回一个新的对应于此次连接的客户端名称。

指令结构

local ret，socketName = SocketAccept(serverName)

返回值

ret: 监听结果 数据类型：bool

socketName：数据类型：字符串类型，套接字名称，用于描述本次连接。

指令参数

参数	数据类型	说明
serverName	string	正在监听的服务端套接字名称。

指令示例

```
local serverName =  “SocketServer”
local ret = false
local socketName = nil
ret = SocketCreate(serverName,“192.168.89.125”,1080)
ret, socketName = SocketAccept (serverName)
if not ret then
      print(“SocketAccept failed!”)
     return
end
```

4.57 SocketConnect 连接远程 TCP/IP 应用程序

指令功能

SocketConnect 用于将套接字与客户端 TCP/IP 应用程序相连，创建连接。

建立一个 TCP 连接，如果连接成功，返回 1；如果连接不成功，返回-1。

指令结构

ret = SocketConnect(socketName, host, port, [timeout])

返回值

ret: 数据类型 num。连接结果，连接成功，返回 1；连接失败，返回非 1。

指令参数

参数	数据类型	说明
socketName	string	要建立连接的套接字名
host	string	要连接的主机 IP 地址
port	num	要连接的主机端口号
timeout	num	超时时间，缺省值为-1，即最大超时 75s。

指令示例

```
local socketName = “mSocket”
--建立连接
local err = SocketConnect(socketName, “192.168.89.126”, 6000, 1000)
--连接失败，处理
if err ~= 1 then
    print(“connection failed !”)
    return false
end
```

4.58 SocketDisconnect 断开与远程 TCP/IP 应用程序的连接

指令功能

SocketDisconnect 用于断开与远程计算机的连接。

指令结构

ret = SocketDisconnect(socketName)

返回值

ret：数据类型 bool。返回状态，如果成功返回 true；如果失败则返回 false。

指令参数

参数	数据类型	说明
socketName	string	要断开连接的套接字名

指令示例

```
local socketName = “mSocket”
local ret = 0
ret = SocketConnect(socketName, "192.168.90.2", 8000, 2000)
if ret ~= 1 then
    print(“SocketConnect failed!”)
    return
End
--断开 mSocket 的连接
SocketDisconnect(socketName)
```

4.59 SocketReceive 接收来自远程 TCP/IP 应用程序的数据

指令功能

SocketReceive 用于从远程计算机接收数据。如果接收成功，返回 1，接收数据和接受字节数；如果接收不成功，返回-1。

指令结构

ret, recvData, recvLength = SocketReceive (socketName, [type],[timeout])

返回值

返回值	数据类型	说明
ret	num	返回值，为 1 表示接收成功，为-1 表示接收失败
recvData	string/table	接收到的数据，可接受字符串或字节接收失败时为 nil。最大支持 1024 字节。
recvLength	num	接收到的字节数，接收失败时为 0

指令参数

参数	数据类型	说明
socketName	string	已连接的套接字名称
type	num	接收数据类型，缺省值为 0。为 0 时表示接受字符串，为 1 时表示接受字节数组。
timeout	num	超时时间，缺省值为-1，即一直阻塞

指令示例

```
local socketName = “mSocket”
local ret = 0
ret = SocketConnect(socketName, "192.168.90.2", 8000, 2000)
if ret ~= 1 then
    print(“SocketConnect failed!”)
    return
end
```

```
-- 发送字符串
ret = SocketSend(socketName, “getMessage”)
ret, recvMsg, recvLength = SocketReceive(socketName,0,2000)
print(string.format(“Recevied message: %s”, recvMsg))
SocketDisconnect(socketName)
```

4.60 SocketSend 向远程 TCP/IP 应用程序发送数据

指令功能

SocketSend 用于向远程计算机发送数据。

如果发送成功，返回 1；如果发送不成功，返回-1。

指令结构

ret = SocketSend(socketName, sendData, [type],[timeout])

返回值

ret：数据类型 bool。如果发送成功，返回 1；如果发送不成功，返回-1。

指令参数

参数	数据类型	说明
socketName	string	要发生数据的已连接套接字名
sendData	string/table	要发送的数据，支持字符串和字节，最大支持 1024 字节。
type	num	接收数据类型，缺省值为 0。为 0 时表示发送字符串，为 1 时表示发送字节数组。
timeout	num	超时时间，缺省值为-1，即一直阻塞

指令示例

```
local socketName = “mSocket”
local ret = 0
ret = SocketConnect(socketName, "192.168.90.2", 8000, 2000)
if ret ~= 1 then
    print(“SocketConnect failed!”)
    return
end

-- 发送字符串
local sendMsg = "hello"
```

```
ret = SocketSend(socketName, sendMsg)

-- 发送字节数组
local sendMsg = {20, 154, 1, 13, 90, 84}
ret = SocketSend(socketName, sendMsg)
SocketDisconnect(socketName)
```

4.61 Stop 停止程序执行

指令功能

Stop 用于停止程序执行。在 Stop 指令就绪之前，将完成当前执行的所有移动。

指令结构

Stop ( )

指令示例

MoveL (0, P10, v50, fine, tool0, wobj0)

Stop ( )

停止直线运动。

4.62 WZBoxDef 定义一个箱形安全区域

指令功能

用于定义拥有直线箱形状，且各侧均与世界坐标系各轴平行的安全区域。

指令结构

block = WZBoxDef (InOutside, LowPoint, HighPoint)

返回值

block：返回箱子的定义，储存在 shapedata 型变量（参数 Shape）中，以便将来在 WZLimSup 或 WZDOSet 指令中使用。

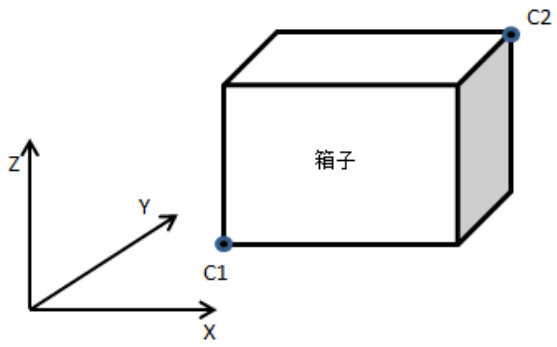
指令参数

参数	数据类型	说明
InOutside	num	数据类型：num 0，定义箱内的体积。 1，定义箱外的体积（相反体积）。
LowPoint	pos	用于定义箱子的一个较低角的位置（x、y、z），以 mm 计。
HighPoint	pos	用于定义与先前角相对的角的位置（x、y、z），以 mm 计。

注意

LowPoint 和 HighPoint 位置必须适用于相对的角（采用不同的 x、y 和 z 坐标值）。

指令示例



local c1 = {503.650024,-59.200000,557.720001}

local c2= {643.650024,59.200000,357.720001}

local block = WZBoxDef(0, c1, c2)

以 C1,C2 为对角线，定义一个箱体安全区域。

4.63 WZCylDef 定义圆柱体安全区域

指令功能

用于定义外形为圆柱形，且圆柱轴与世界坐标系 z 轴平行的安全区域。

指令结构

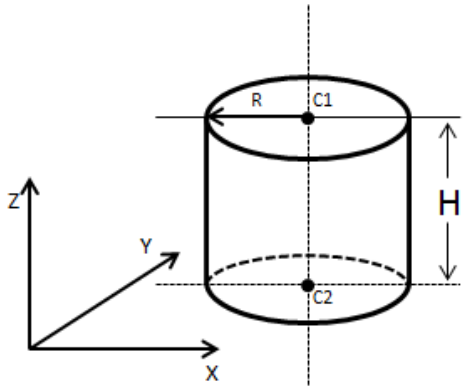
cyl = WZCylDef (InOutside, CentrePoint, Radius,Height)

返回值

cyl：返回圆柱体的定义，储存在 shapedata 型变量（参数 Shape）中，以便将来在 WZLimSup 或 WZDOSet 指令中使用。

指令参数

参数	数据类型	说明
InOutside	num	0, 定义圆柱体内的体积。 1, 定义圆柱体外的体积（相反体积）。
CentrePoint	pos	用于定义圆柱体的上底面或下底面的圆心位置（x、y、z），以 mm 计。
Radius	num	圆柱体的半径，以 mm 计。返回：用于储存指定体积的变量。
Height	num	圆柱体的高度，以 mm 计。如果其值为正（+z 方向），则 CentrePoint 参数为圆柱体下底面圆中心。如果其值为负（-z 方向），则 CentrePoint 参数为圆柱体上底面圆心。



如上图圆柱体空间，CentrePoint 若设置为上底面圆心 C1，则 Height 设置为-H；CentrePoint 若设置为下底面圆心 C2，则 Height 设置为-H。

指令示例

```
local C2 = {200,300,400}
local R= 500
local H= 600
local sph = WZCylDef(0, C2, R, H)
```

定义半径为 500，高度为 600 的圆柱体为机器人运动安全区域。

4.64 WZSphDef 定义球形安全区域

指令功能

用于定义外形为球形的安全区域。

指令结构

shp = WZSphDef (InOutside, CentrePoint, Radius)

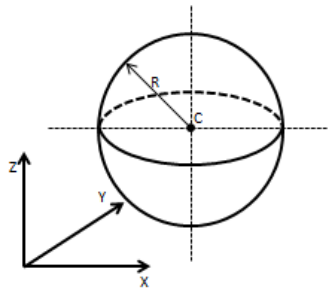
返回值

shp：返回球体的定义，储存在 shapedata 型变量（参数 Shape）中，以便将来在 WZLimSup 或 WZDOSet 指令中使用。

指令参数

参数	数据类型	说明
InOutside	num	0, 定义球内的体积。 1, 定义球体外的体积（相反体积）。
CentrePoint	pos	用于球体中心的位置（x、y、z），以 mm 计。
Radius	num	球体的半径，以 mm 计。

指令示例



local C = {200,300,400}

local R= 500

local sph = WZSphDef(0, C, R)

以 C 为球心，以 R 为半径定义一个球形安全区域

4.65 WZLimJointDef 定义有关关节内限制的安全区域

指令功能

用于定义用于定义关节坐标系中的安全区域，以便将机械臂和外轴用于工作区域的限制。通过 WZLimJointDef，有可能限制各机械臂和外轴的工作区域。

指令结构

wzljoint = WZLimJointDef (num, MiddleJointVal, DeltaJointVal)

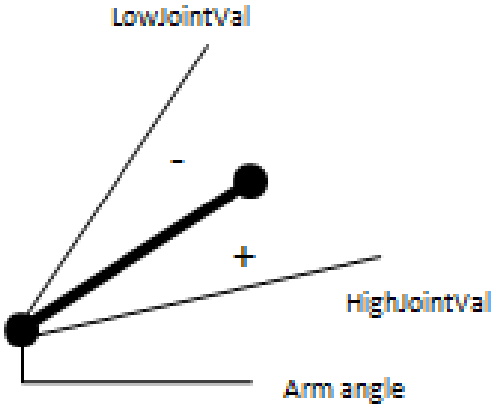
返回值

wzljoint：返回关节空间的定义，储存在 shapedata 型变量（参数 Shape）中，以便将来在 WZLimSup 或 WZDOSet 指令中使用。

指令参数

参数	数据类型	说明
InOutside	num	0, 定义 LowJointVal ... HighJointVal 内的关节空间。 1, 定义 LowJointVal ... HighJointVal 外的关节空间（相反关节空间）。
LowJointVal	JOINTTARGET	确定有关关节空间下限的关节坐标位置。针对各机械臂轴和外轴进行说明（旋转轴以度计，线性轴以毫米计）。在绝对关节中进行说明（而非在针对外轴的偏移量坐标系EOffsSet或EOffsOn中）。有关一些轴的值9E9意味着应当监控该轴的下限。编程期间，无效外轴同时得出9E9。
HighJointVal	JOINTTARGET	确定有关关节空间上限的关节坐标位置。针对各机械臂轴和外轴进行说明（旋转轴以度计，线性轴以毫米计）。在绝对关节中进行说明（而非在针对外轴的偏移量坐标系 EOffsSet 或 EOffsOn 中）。有关一个轴的值 9E9 意味着应当监控该轴的上限。编程期间，无效外轴同时得出 9E9。针对用于监控的所有轴，各轴的 HighJointVal 减去 LowJointVal，所得结果必须大于 0。

下图显示了有关旋转轴关接空间的定义。



- 如果使用 WZLimJointDef 连同 WZDOSet，则仅当所有有效轴以及关节空间监控位于关节空间前或内部时，方才设置数字信号输出信号。
- 如果使用包含外部关节空间的 WZLimJointDef（参数 InOutside = 1）连同 WZLimSup，则当一个包含关节空间监控的有效轴达到关节空间时，机械臂随即停止。
- 如果使用包含内部关节空间的 WZLimJointDef（参数 InOutside = 0）连同 WZLimSup，则当包含关节空间监控的最后一个有效轴达到关节空间时，机械臂随即停止。这意味着关节空间内部可同时拥有一个或多个轴，但是并非所有轴都是有效并接受监控的轴。

指令示例

```
local low_pos = {{-90,9E9,9E9,9E9,9E9,9E9,9E9,9E9},{-1000,9E9,9E9,9E9,9E9,9E9,9E9,9E9}}
local high_pos = {{90,9E9,9E9,9E9,9E9,9E9,9E9,9E9},{9E9,9E9,9E9,9E9,9E9,9E9,9E9,9E9}}
local joint = WZLimJointDef(0,low_pos,high_pos)
```

4.66 WZLimSup 启用安全区域限制监控

指令功能

用于定义行动，并启用安全区域，以监控机械臂或外轴的工作区域。

执行该指令后，在程序执行和点动期间，当机械臂 TCP 达到规定全局区域，或当机械臂/外轴达到关节中的规定安全区域时，移动得以停止。

指令结构

WZLimSup(motgrp,wztype,wzname,Shape)

指令参数

参数	数据类型	说明
motgrp/mec_name	num	这个参数可以选择 motgrp 赋值，也可以选择 mec_name 赋值。若选择 motgrp：0—左臂，1—右臂。如果不设置，则默认为左臂。若选择 mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”。
Wztype	num	0, 用于定义的安全区域为临时安全区域。 1, 用于定义的安全区域为固定式安全区域。 必须指定参数0或1之一
Wzname	string	安全区域的识别号（字符串）。
Shape	shapedata	用以定义安全区域体积的变量。

指令示例

```
local C1 = {200,300,400}
local r1= 500
local h1= 600
local wz2= "wz2"
local sph = WZSphDef(0, C1, r1)
WZLimSup(1, 1, wz2, sph)
```

4.67 WZDOSet 启用安全区域，设置数字信号输出

指令功能

用于定义行动，并启用安全区域，以监控机械臂移动。

在执行该指令后，当机械臂 TCP 或机械臂/外轴（关节中的区域）位于指定安全区域内，或正在接近安全区域时，将数字信号输出信号设置为指定值。

指令结构

WZDOSet(motgrp,wztype,wzname,side,Shape,Signal,SetValue)

指令参数

参数	数据类型	说明
motgrp/mec_name	num/ string	这个参数可以选择 motgrp 赋值，也可以选择 mec_name 赋值。 若选择 motgrp：0—左臂，1—右臂。如果不设置，则默认为左臂。 若选择 mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”。
Wztype	num	0, 用于定义的安全区域为临时安全区域。 1, 用于定义的安全区域为固定式安全区域。 必须指定参数 0 或 1 之一。
Wzname	string	安全区域的识别号（字符串）。
Side	num	0，停止,不输出信号。 1，在机械臂 TCP 或指定轴达到规定体积前（尽可能在该体积前），将设置数字信号输出信号。 2, 当机械臂 TCP 或指定轴位于指定体积内时，将设置数字信号输出信号。
Shape	shapedata	用以定义安全区域体积的变量。
Signal	string	待改变数字信号输出信号的名称。 如果使用固定式安全区域，则必须写入信号，以作为用户访问保护。针对系统参数或指定轴中的信号，设置 Access Level
SetValue	num	当机械臂TCP位于体积内或刚好在进入体积之前的信号期望值（0或1）。 当位于体积外或恰好位于体积外时，将信号设置为相反值。

指令示例

```
local C1 = {200,300,400}
local r1= 500
local sph = WZSphDef(0, C1, r1)
WZDOSet( 0,1, "wz2",1, sph,"DO10_7",1)
```

定义球形安全区域，并关联 DO10_7 信号监控机器人是否接近或处于安全区域

4.68 WZDisable 停用临时安全区域监控

指令功能

用于停用对临时安全区域的监控，其预先定义以便停止移动或设置输出。

指令结构

WZDisable(motgrp,wzname)

指令参数

参数	数据类型	说明
motgrp/mec_name	num	这个参数可以选择 motgrp 赋值，也可以选择 mec_name 赋值。 若选择 motgrp：0—左臂，1—右臂。如果不设置，则默认为左臂。 若选择 mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”。
Wzname	string	安全区域的识别号（字符串）。

指令示例

WZDisable(0,“wz1”)

停止对“wz1”的监控

4.69 WZEnable 启用临时安全区域监控

指令功能

用于重新启用对临时安全区域的监控，其预先定义，以便停止移动或设置输出。

指令结构

WZEnable(motgrp,wzname)

指令参数

参数	数据类型	说明
motgrp/mec_name	num/ string	这个参数可以选择 motgrp 赋值，也可以选择 mec_name 赋值。若选择 motgrp：0—左臂，1—右臂。如果不设置，则默认为左臂。若选择 mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”。
Wzname	string	安全区域的识别号（字符串）。

指令示例

WZEnable (0,“wz1”)

启动对“wz1”的监控。

4.70 WZFree 擦除临时安全区域监控

指令功能

用于擦除临时安全区域的定义，其预先定义，以便停止移动或设置输出。

指令结构

WZFree (motgrp,wzname)

指令参数

参数	数据类型	说明
motgrp/mec_name	num/ string	这个参数可以选择 motgrp 赋值，也可以选择 mec_name 赋值。 若选择 motgrp：0—左臂，1—右臂。如果不设置，则默认为左臂。 若选择 mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”。
Wzname	string	安全区域的识别号（字符串）。

指令示例

WZFree (0, “wz1”)

擦除对“wz1”安全区域的监控。

4.71 TPWrite 在示教器上显示信息

指令功能

用于在 TPU 上显示信息。

指令结构

TPWrite(arg1,arg2,arg3, ...)

指令参数

参数	数据类型	说明
arg1,arg2,arg3, ...	string	字符串格式符

指令示例

TPWrite("SocketCreate failed! error code: " .. retVal)

在示教器上打印信息“SocketCreate failed! error code: 10”。

4.72 ClkStart 启动时钟指令

指令功能

用于启动时钟开始计时。

指令结构

ClkStart(clockName)

指令参数

参数	数据类型	说明
clockName	string	时钟名称

指令示例

ClkStart("clock10")

如果当前时钟 clock10 存在，启动时钟 clock10, 并开始计时。

如果当前时钟 clock10 不存在，则创建 clock10 时钟，并启动时钟 clock10, 并开始计时。

4.73 ClkReset 复位时钟指令

指令功能

用于复位时钟计数值为 0。

指令结构

ClkReset(clockName)

指令参数

参数	数据类型	说明
clockName	string	时钟名称

指令示例

ClkReset("clock10")

如果当前时钟 clock10 存在，复位 ClkRead 值为 0。

如果当前时钟 clock1 不存在，则创建 clock10 时钟，并将 clock10 复位使得 ClkRead 值为 0。

4.74 ClkStop 停止时钟指令

指令功能

用于停止时钟计数。

指令结构

ClkStop(clockName)

指令参数

参数	数据类型	说明
clockName	string	时钟名称

指令示例

ClkStop("clock10")

如果当前时钟 clock10 存在, 停止时钟 clock10 计时。

如果当前时钟 clock10 不存在，则报错。

4.75 ClkRead 读取时钟指令

指令功能

用于读取时钟计时值。

指令结构

val = ClkRead(clockName)

返回值

val: 返回值，返回读取的时钟计时值。val 返回值单位为 ms。

指令参数

参数	数据类型	说明
clockName	string	时钟名称

指令示例

local val = ClkRead("clock10")

如果当前时钟 clock10 存在,读取时钟 clock10 的计时。

如果当前时钟 clock10 不存在，则报错。

4.76 CallScript 调用子程序

指令功能

用于在程序中调用子程序。

指令结构

status, err = CallScript(fileName)

返回值

status：数据类型 bool，返回状态，成功返回 true，失败返回 false.

err：数据类型 string，如果失败，则返回错误信息。

指令参数

参数	数据类型	说明
fileName	string	需要调用的程序名称

指令示例

CallScript("job3.lua")

在程序中调用“job3.lua”子程序。

4.77 ByteToStr 字节数据转字符串

指令功能

将字节数据按照 AscII 码表和指定格式转换成对应的字符。

指令结构

str = ByteToStr(value,format)

返回值

str： 数据类型 string, 返回转换后的字符串。

指令参数

参数	数据类型	说明
value	num	待转换的数据值，取值范围 0-2^8-1
format	string	数据转化格式，数据转换格式取值如下： • HEX; • OKT; • DEC; • BIN; • CHAR;

指令示例

local str1 = ByteToStr(11, “DEC”);

把数值 11 按照十进制格式转化成对应的字符，返回值 str1 为 “11”。

local str2 = ByteToStr(11, “HEX”);

把数值 11 按照十六进制格式转化成对应的字符，返回值 str2 为 “0b”。

local str3 = ByteToStr(0x3A, “CHAR”)

把十六进制数据 0x3A 按照 ASCII 码表转化成对应的字符，返回值 str3 为 ‘:’ 。

4.78 StrToByte 字符串转字节数据

指令功能

将字符串按照指定格式和 AscII 码表转换成对应的数值数据。

指令结构

byte = StrToByte (str,format)

返回值

byte： 数据类型 num, 返回转换后的数值数据。

指令参数

参数	数据类型	说明
value	string	待转换的字符串
format	string	数据转化格式，数据转换格式取值如下： • HEX; • OKT; • DEC; • BIN; • CHAR;

指令示例

local var= StrToByte("11","BIN");

把字符串按照二进制格式转化成对应的数值，返回值 var 为 3。

local var= StrToByte("11","OKT");

把字符串按照八进制格式转化成对应的数值，返回值 var 为 9。

local var= StrToByte(";","CHAR");

把字符串按照 ASCII 码转化成对应的十进制数值，返回值 var 为 59。

4.79 WordToStr 字数据转字符串

指令功能

将字数据按照 AscII 码表和指定格式转换成对应的字符。

指令结构

str = WordToStr(value,format)

返回值

str：数据类型 string, 返回转换后的字符串。

指令参数

参数	数据类型	说明
value	num	待转换的数据值，取值范围 0-2^16-1
format	string	数据转化格式，数据转换格式取值如下： • HEX; • OKT; • DEC; • BIN; • CHAR;

指令示例

local str = WordToStr(11, “BIN”);

把数据 11 按照二进制格式转化成对应的字符串，返回值 str 为 “0000000000001011” 。

local str = WordToStr(11, “OKT”);

把数据 11 按照八进制格式转化成对应的字符串，返回值 str 为 “000013”。

local str =WordToStr(0x3A3B, “CHAR”);

把数据 0x3A3B 转化成对应的 ASCII 码字符，返回值 str 为 “:;”。

4.80 StrToWord 字符串转字数据

指令功能

将字符串按照指定格式或 AscII 码表转换成对应的数据。

指令结构

var = StrToWord (str,format)

返回值

var：返回转换后的数据，整型值。

指令参数

参数	数据类型	说明
value	string	待转换的字符串
format	string	数据转化格式，数据转换格式取值如下： • HEX; • OKT; • DEC; • BIN; • CHAR;

指令示例

local var= StrToWord(“11”, “DEC”) ;

把字符串 “11”按照十进制格式转化成对应的数值，返回值 var 为 11。

local var= StrToWord(“11”, “HEX”) ;

把字符串 “11”按照十六进制格式转化成对应的数值，返回值 var 为 17。

local var=StrToWord(";","CHAR") ;

把字符串 “:;”查询 ASCII 码表转化成对应的十进制数值，返回值 var 为 59。

4.81 DWordToStr 双字数据转字符串

指令功能

将字数据按照 AscII 码表和指定格式转换成对应的字符串。

指令结构

str = DWordToStr(value,format)

返回值

str：数据类型 string, 返回转换后的字符串。

指令参数

参数	数据类型	说明
value	num	待转换的数据值，取值范围 0-2^32-1
format	string	数据转化格式，数据转换格式取值如下： • HEX; • OKT; • DEC; • BIN; • CHAR;

指令示例

local str = DwordToStr(11, “BIN”);

把数据 11 按照二进制格式转化成对应的字符串，返回值 str 为 "00000000000000000000000000001011"。

local str = DwordToStr(11, “OKT”);

把数据 11 按照八进制格式转化成对应的字符串，返回值 str 为"000000000013"。

local str = DwordToStr(0x3A3B6664, “CHAR”);

把数据 0x3A3B6664 转化成对应的 ASCII 码字符，返回值 str 为 ":;fd"。

4.82 StrToDWord 字符串转双字数据

指令功能

将字符串按照指定格式和 AscII 码表转换成对应的数据。

指令结构

var = StrToDWord (str,format)

返回值

var：返回转换后的数据，整型值。

指令参数

参数	数据类型	说明
value	string	待转换的字符串
format	string	数据转化格式，数据转换格式取值如下： • HEX; • OKT; • DEC; • BIN; • CHAR;

指令示例

local var= StrToDword(":;fd","CHAR") ;

把字符串按照 ASCII 码表转化成对应的数值，返回值 var 为 0x3A3B6664。

local var= StrToDword("11","BIN");

把字符串按照二进制格式转化成对应的数值，返回值 var 为 3。

local var= StrToDword("11","HEX");

把字符串按照十六进制格式转化成对应的数值，返回值 var 为 17。

4.83 ValToStr 变量转化成字符串

指令功能

将数字变量转化成对应的字符串

指令结构

str = ValToStr(value)

返回值

str：数据类型 string, 返回转换后的字符串。

指令参数

参数	数据类型	说明
value	num	待转换的数据值

指令示例

local str = ValToStr(65);

将数字 65 直接转化成字符串“65”。

local str =ValToStr(nil);

将值 nil 直接转化成字符串“nil”。

4.84 StrToVal 字符串转化成数字

指令功能

将字符串转化成对应的数字值。

指令结构

val = StrToVal(str)

返回值

val：返回转换后的数值。

指令参数

参数	数据类型	说明
str	string	待转换的字符串

指令示例

local var = StrToVal(“67”);

将字符串“67”直接转化成数字 67。

local var = StrToVal("0x11") == 17;

将字符串"0x11"转化成十进制数字 17。

4.85 StrPart 求字符串的字串

指令功能

求字符串指定位置开始指定长度的子字符串。

指令结构

StrPart(str,loc,len)

返回值

str：数据类型 string，返回截取后的字符串。

指令参数

参数	数据类型	说明
str	string	字符串
loc	num	指定截取字符串的起始位置
len	num	指定截取字符串的长度

指令示例

local str = StrPart(“abcdefgth”,3,4)

从字符串“abcdefgth”的第 3 个字符开始，取 4 个字符，即返回 str = “cdef”。

local str = StrPart("abcdefghi",9,1)

从字符串"abcdefghi"的第 9 个字符开始，取 1 个字符，即返回 str = ‘i’。

4.86 StrLen 求字符串长度

指令功能

求取字符串长度

指令结构

var = strLen(str)

返回值

var：数据类型 num, 返回字符串的长度。

指令参数

参数	数据类型	说明
str	string	待求长度的字符串

指令示例

local var = strLen(“abcdfge”)

字符串“abcdfge”的长度 var 为 7。

4.87 Incr 数值数据值增加 1

指令功能

用于给数值数据增加 1。

指令结构

Incr (name)

函数参数

参数	数据类型	说明
name	NUMDATA	待改变永久数据对象的名称。

指令示例

Incr(CycleCount)

数值数据 CycleCount 的数值增加 1。

4.88 Decr 数值数据值减少 1

指令功能

用于给数值数据减去 1。

指令结构

Decr (name)

函数参数

参数	数据类型	说明
name	NUMDATA	待改变永久数据对象的名称。

指令示例

Decr(CycleCount)

数值数据 CycleCount 的数值减少 1。

5 焊接专用指令

5.1 焊接数据类型

5.1.1 SEAMDATA 焊缝数据

焊缝数据，用于控制焊接的开始和结束阶段，定义了起弧、加热和收弧等参数。如果焊接操作被中断后，焊接过程重新开始，SEAMDATA 也被使用。

数据结构

```
SEAMDATA
{
        purge_time;
        preflow_time;
        ign_arc;            //ignition_on
        ign_move_delay;     //ignition_on No and ign_move_delay_on
        scrape_start;       //scrape_on and scrape_opt_on
        heat_speed;         //heat_on and heat_as_time
        heat_time;          //heat_on and heat_as_time
        heat_distance;      //heat_on
        heat_arc;           //heat_on
        cool_time;          //cool_time_on and fill_on
        fill_time;          //fill_on
        fill_arc;           //fill_on
        bback_time;         //burnback_on
        rback_time;         //burnback_on
        bback_arc;          //burnb_volt_on and burnback_on
        postflow_time;
}
```

参数说明

参数名	单位	默认值	说明
purge_time	ms	100	焊机气泵的充气时间，气体充满气管和焊枪。
preflow_time	ms	100	焊机启动时的提前送气时间，机器人在此时间内保持不动。
ign_sched	无	0	焊机起弧的job号。
ign_mode	无	0	焊机起弧的程序号。
ign_volt	0.1V(%)	20 (0)	焊机起弧的设定电压或者设定电压修正值。当焊机配置参数的电压设定方式为 0 时，该参数表示给定电压；当电压设定方式为 1 时，该参数表示设定电压修正值。
ign_wirefeed	mm/s	10	焊机起弧的设定送丝速度。当焊机配置参数的电流设定方式为 1 时，该参数有效。
ign_current	A	100	焊机起弧的设定电流。当焊机配置参数的电流设定方式为 0 时，该参数有效。
ign_move_delay	ms	100	焊机起弧成功后运动延迟时间，为起弧稳定之后到加热开始之前的时间。
scrape_start（预留）		0	预留参数。
heat_speed（预留）	mm/s	100	焊机加热过程中机器人的运动速度。 目前此参数不生效。
heat_time	ms	100	焊机加热的持续时间。当加热持续距离参数heat_distance为 0 时，该参数有效。
heat_distance	mm	10	焊机加热的持续距离。当此参数>0 时，heat_time参数无效。
heat_sched	无	0	焊机加热的job号。
heat_mode	无	0	焊机加热的程序号。
heat_volt	0.1V(%)	20 (0)	焊机加热的设定电压或者设定电压修正值。当焊机配置参数的电压设定方式为 0 时，该参数表示给定电压；当电压设定方式为 1 时，该参数表示设定电压修正值。
heat_wirefeed	mm/s	10	焊机加热的设定送丝速度。当焊机配置参数的电流设定方式为 1 时，该参数有效。
heat_current	A	100	焊机加热的设定电流。当焊机配置参数的电流设定方式为 0 时，该参数有效。
cool_time	ms	100	焊机收弧的冷却时间。如果收弧阶段支持填弧功能，此时间表示第一次断弧到填弧起弧的冷却时间。

fill_time	ms	100	焊机填弧的时间。
fill_sched	无	0	焊机填弧的job号。
fill_mode	无	0	焊机填弧的程序号。
fill_volt	0.1V(%)	20 (0)	焊机填弧的设定电压或者设定电压修正值。当焊机配置参数的电压设定方式为 0 时，该参数表示给定电压；当电压设定方式为 1 时，该参数表示设定电压修正值。
fill_wirefeed	mm/s	10	焊机填弧的设定送丝速度。当焊机配置参数的电流设定方式为 1 时，该参数有效。
fill_current	A	100	焊机填弧的设定电流。当焊机配置参数的电流设定方式为 0 时，该参数有效。
bback_time	ms	100	焊机收弧的回烧时间。
rback_time	ms	100	焊机填弧的回烧时间。
bback_sched	无	0	焊机收弧的job号。
bback_mode	无	0	焊机收弧的程序号。
bback_volt	0.1V(%)	20 (0)	焊机收弧的设定电压或者设定电压修正值。当焊机配置参数的电压设定方式为 0 时，该参数表示给定电压；当电压设定方式为 1 时，该参数表示设定电压修正值。
bback_wirefeed	mm/s	10	焊机收弧的设定送丝速度。当焊机配置参数的电流设定方式为 1 时，该参数有效。
bback_current	A	100	焊机收弧的设定电流。当焊机配置参数的电流设定方式为 0 时，该参数有效。
postflow_time	ms	100	焊接停止后的送气时间。

5.1.2 WELDDATA 焊接数据

焊接数据，定义了焊机的焊接速度、给定焊接电压和给定焊接电流等焊接行为。

数据结构

```
WELDDATA
{
    weld_speed;
    main_arc;
}
```

参数说明

参数名	子参数	单位	默认值	说明
weld_speed		mm/s	100	焊机焊接的速度。
main_arc	sched	无	0	焊机焊接的Job号。焊机工作在job模式该设置才能够使用。
	mode	无	0	焊机焊接的程序号。
	volt	V(%)	20 (0)	焊机焊接的设定电压或者设定电压修正值。当焊机配置参数的电压设定方式为 0 时，该参数表示给定电压；当电压设定方式为 1 时，该参数表示设定电压修正值。
	wirefeed	mm/s	10	焊机焊接的设定送丝速度。当焊机配置参数的电流设定方式为 1 时，该参数有效。
	current	A	100	焊机焊接的设定电流。当焊机配置参数的电流设定方式为 0 时，该参数有效。

5.1.3 WEAVEDATA 摆弧数据

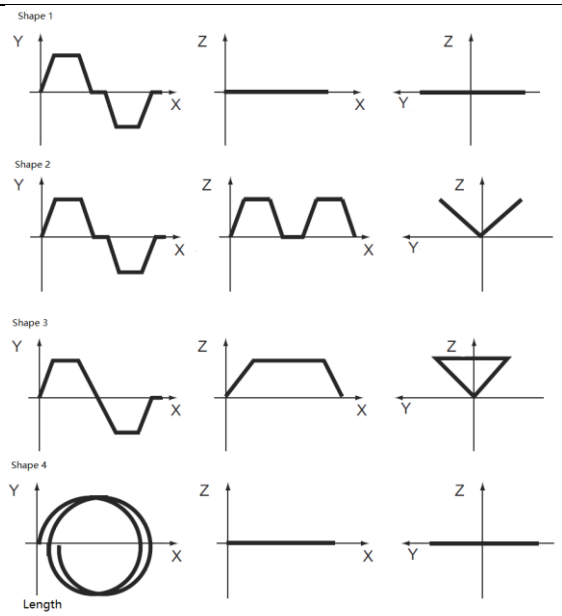
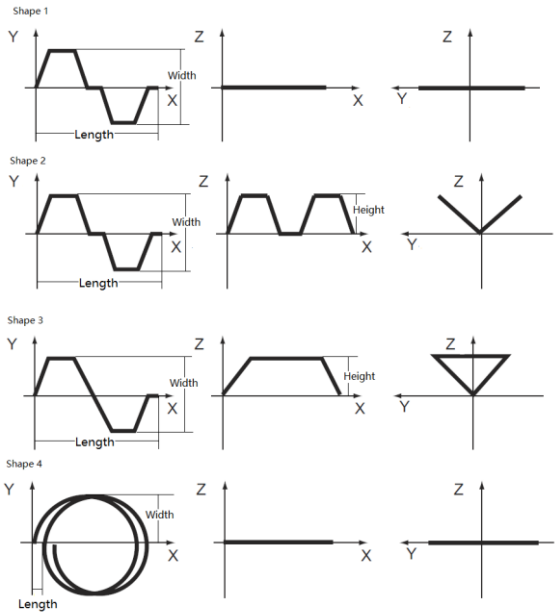
摆弧数据，用于加热和焊接过程，定义了摆弧的路径。

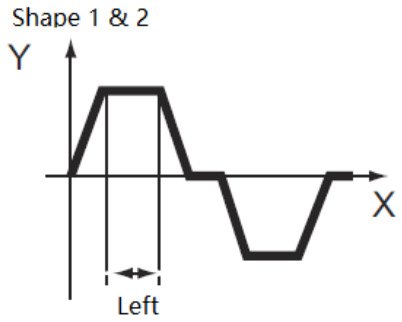
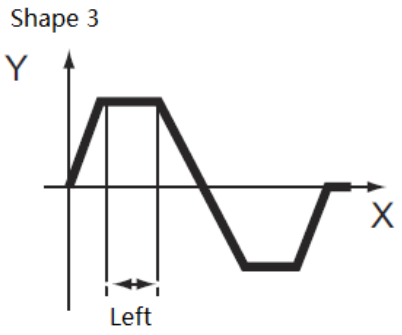
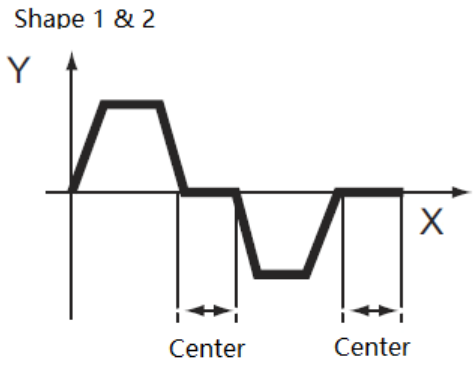
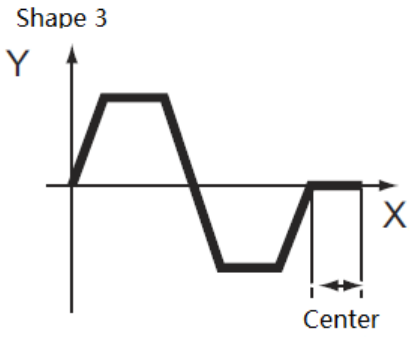
数据结构

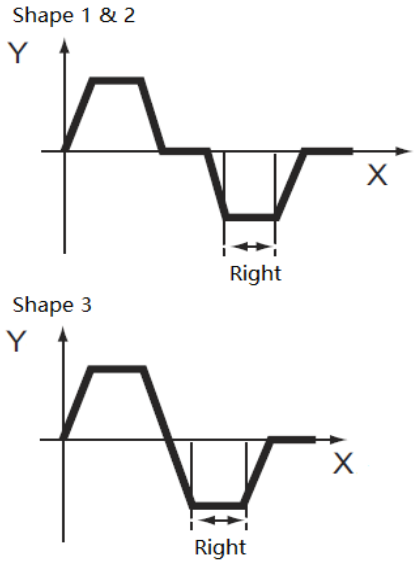
```
WEAVEDATA
{
    weave_shape;
    weave_type;
    weave_length;
    weave_width;
    weave_height;
    dwell_left;
    dwell_center;
    dwell_right;
    weave_dir;
    weave_tilt;
    weave_ori;
    weave_bias;
}
```

参数说明

参数名	单位	默认值	说明
weave_shape	无	0	定义摆弧的形状。 NO_WEAVE：0， 表示没有摆弧； WEAVE_XY_Z：1， 表示在XY平面做Z型摆弧； WEAVE_YZ_V：2， 表示在和焊缝垂直的YZ平面做V型摆弧； WEAVE_YZ_TRI：3， 表示在和焊缝垂直的YZ平面做三角型摆弧； WEAVE_XY_CIR：4， 表示在XY平面做圆型摆弧； WEAVE_XY_SIN：5， 表示在XY平面做正玄波摆弧。

			
weave_type	无	0	WEAVE_ARM：0，一般的臂摆动，机器人的 6 个轴都参与摆动； WEAVE_WRIST：1，腕摆动，第 5 轴、第 6 轴参与摆动。
weave_length	mm	10	定义摆弧一个周期在原轨迹上的运动距离，参考例图中的Length参数。 注意： 对应相应的摆弧形状，其设定的参数dwell_left、dwell_center 和 dwell_right 各段值的总和，须小于weave_length的设定值；否则，会导致机器人报错。 

weave_width	mm	4	定义摆弧的宽度。
weave_height	mm	0	定义摆弧的高度，摆动高度，只有在三角摆动和V字摆动时此参数才有效。
dwell_left	mm	0	定义摆弧的左边平移距离。  
dwell_center	mm	0	定义摆弧的中间平移距离。  

dwell_right	mm	0	定义摆弧的右边平移距离。 
weave_dir (预留)		0	摆动倾斜的角度，焊缝的X方向。
weave_tilt (预留)		0	摆动倾斜的角度，焊缝的Y方向。
weave_ori (预留)		0	摆动倾斜的角度，焊缝的Z方向。
weave_bias (预留)		0	

数据定义示例

WEAVEDATA("weave_z",1,0,10,10,0,2,2,1,0,0,0,0)

- 1： 摆焊形状；
- 0： 摆焊类型；
- 10：摆焊一个周期的长度；
- 10：摆焊振幅的宽度；
- 0：摆焊高度（Z 方向不参与摆焊）；
- 2：摆弧左边平移距离；
- 2：摆弧中间平移距离；
- 1：摆弧右边平移距离；
- 0,0,0,0：最后四个参数目前不起作用。

5.1.4 MULTIPASSDATA 多道数据

数据结构

```
MULTIPASSDATA
{
    Direction,
    ApproachDistance,
    DepartDistance,
    StartOffset,
    EndOffset,
    SeamOffs_y,
    SeamOffs_z,
    SeamRot_x,
    SeamRot_y,
    SeamRot_z
  }
```

参数说明

参数	单位	说明
Direction	num	1，从焊缝起点到结束点； -1，从焊缝结束点到起点。
ApproachDistance	num	单位：mm，起点高度
DepartDistance	num	单位：mm，结束点高度
StartOffset	num	单位：mm，路径x向起点偏移
EndOffset	num	单位：mm，路径x向结束点偏移
SeamOffs_y	num	单位：mm，路径y向偏移
SeamOffs_z	num	单位：mm，路径z向偏移
SeamRot_x	num	单位：度，绕路径x旋转
SeamRot_y	num	单位：度，绕路径y旋转
SeamRot_z	num	不使用

5.2 焊接指令

5.2.1 SetActivePara 设置焊接数据（精简指令模式）

指令功能

设置 seamdata, welddata, weavedata 等焊接相关的数据参数。

使用该指令可以设置全局的焊接数据，定义的数据类型为以上三种，定义的数据类型数量不固定。调用该指令定义后，在程序中的所有焊接数据都使用该指令设置的焊接数据，如果 SetActivePara 没有设置相关的焊接数据，则在调用焊接指令时需要调用。

也可以在调用焊接指令时指定需要的焊接数据，如果没有调用则使用 SetActivePara 所设置的焊接数据。

注意：

该指令用于精简指令设置时调用，否则焊接相关指令不能正常工作。

精简指令设置：

在 TPU 界面选择 “配置”->“其它配置”->“精简指令”，在“在添加指令时是否使用精简模式”选择“true”或“false”。

选择 true 时部分常用焊接指令中将不显示 seamdata, welddata, weavedata 等参数，使用 SetActivePara 设置的焊接参数。如需在焊接过程中修改参数，则在焊接指令编辑中勾选焊接相关参数，或者调用 SetActivePara 设置参数。

选择 false，将不会在指令中省略参数。

指令结构

SetActivePara ([Seam], [Weld], [Weave])

指令参数

参数	数据类型	说明
Seam	seamdata	可选参数，起弧收弧控制参数
Weld	welddata	可选参数，焊接过程控制参数
Weave	weavedata	可选参数，焊接过程摆弧参数

指令示例

精简指令模式

SetActivePara(seam0, weld0, weave0)

ArcLStart(ToPoint, speed0, fine, tool0, wObj0,load0)

ArcL (ToPoint, speed0, z0, tool0, wObj0,load0)

ArcL (ToPoint, speed0, seam0, weld1, weave1, z0, tool0, wObj0,load0)

--此处所设置的焊接参数只在当前指令生效

ArcC (CirPoint ,ToPoint, speed0, z0, tool0, wObj0,load0)

ArcLEnd (ToPoint, speed0, fine, tool0, wObj0,load0)

说明

设置全局焊接数据并开始焊接过程。并可在焊接指令中设置焊接参数，所设置的参数只在当前指令中生效。

非精简指令模式：

ArcLStart(P370,v200,shareSeam0,shareWeld2,shareWeave0,fine,shareTool0,wobj0,nil,load0)

ArcL(P380,v200,shareSeam0,shareWeld2,shareWeave0,z0,shareTool0,wobj0,nil,load0)

ArcL(P390,v200,shareSeam0,shareWeld2,shareWeave0,z0,shareTool0,wobj0,nil,load0)

ArcL(P720,v200,shareSeam0,shareWeld2,shareWeave0,z0,shareTool0,wobj0,nil,load0)

ArcLEnd(P400,v200,shareSeam0,shareWeld2,shareWeave0,fine,shareTool0,wobj0,nil,load0)

说明

在非精简指令模式下，所有指令必须包含 seamdata, welddata, weavedata 等参数，不能省略。

5.2.2 ArcLStart 基于线性运动的弧焊开始

指令功能

本指令用于启动沿直线焊缝的弧焊。

指令结构

ArcLStart(ToPoint, Speed, Seam, Weld, Weave, Zone, Tool, WObj, Tarray, TLoad)

指令参数

参数	数据类型	说明
ToPoint	robtarget	目标点
Speed	speeddata	运动速度（焊接时无效）
Seam	seamdata	起弧收弧控制参数，精简模式下该参数可选择设置。
Weld	welddata	焊接过程控制参数，精简模式下该参数可选择设置。
Weave	weavedata	焊接过程摆弧参数，精简模式下该参数可选择设置。
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
Tarray	triggdata 数据数组	跟踪数据
TLoad	loaddata	负载

指令示例

ArcLStart(P10, v200, seam1, weld1, weave1, fine, tTorch, wobj0, tarr, load0)

启动沿直线焊缝的弧焊运动到 P10 点。

5.2.3 ArcL 基于线性运动的弧焊过程

指令功能

本指令用于沿直线焊缝的弧焊（定义了弧焊的线性运动过程点）。

指令结构

ArcL(ToPoint, Speed, Seam, Weld, Weave, Zone, Tool, WObj, Tarray, TLoad)

指令参数

参数	数据类型	说明
ToPoint	robtarget	目标点位置
Speed	speeddata	运动速度（焊接时无效）
Seam	seamdata	起弧收弧控制参数，精简模式下该参数可选择设置。
Weld	welddata	焊接过程控制参数，精简模式下该参数可选择设置。
Weave	weavedata	焊接过程摆弧参数，精简模式下该参数可选择设置。
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
Tarray	triggdata 数据数组	跟踪数据
TLoad	loaddata	负载

指令示例

ArcL(P20, v10, seam1, weld1, weave1, fine, tTorch, wobj0, event11, load0)

机器人沿直线焊缝的弧焊运动到 P20 点。

5.2.4 ArcLEnd 基于线性运动的弧焊结束

指令功能

本指令用于结束沿直线焊缝的弧焊。

指令结构

ArcLEnd(ToPoint, Speed, Seam, Weld, Weave, Zone, Tool, WObj, Tarray, TLoad)

指令参数

参数	数据类型	说明
ToPoint	robtarget	目标点位置
Speed	speeddata	运动速度（焊接时无效）
Seam	seamdata	焊机启停参数，精简模式下该参数可选择设置。
Weld	welddata	焊接过程参数，精简模式下该参数可选择设置。
Weave	weavedata	焊接过程摆弧参数，精简模式下该参数可选择设置。
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
Tarray	triggdata 数据数组	跟踪数据
TLoad	loaddata	负载

指令示例

ArcLEnd(P30, v10, seam1, weld1, weave1, fine, tTorch, wobj0, event11 , load0)

机器人沿直线焊缝的弧焊运动到 P30 结束。

5.2.5 ArcCStart 基于圆弧运动的弧焊开始

指令功能

本指令用于启动沿圆弧形焊缝的弧焊。

指令结构

ArcCStart(CirPoint, ToPoint, Speed, Seam, Weld, Weave, Zone, Tool, WObj, Tarray, TLoad)

指令参数

参数	数据类型	说明
CirPoint	robtarget	路径点
ToPoint	robtarget	目标点
Speed	speeddata	运动速度（焊接时无效）
Seam	seamdata	起弧收弧控制参数，精简模式下该参数可选择设置。
Weld	welddata	焊接过程控制参数，精简模式下该参数可选择设置。
Weave	weavedata	焊接过程摆弧参数，精简模式下该参数可选择设置。
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
Tarray	triggdata 数据数组	跟踪数据
TLoad	loaddata	负载

指令示例

ArcCStart(P20, P30, v200, seam1, weld1, weave1, fine, tTorch, wobj0, event11, load0)

机器人启动沿圆弧形焊缝的弧焊经过 P20 运动到 P30。

5.2.6 ArcC 基于圆弧运动的弧焊过程

指令功能

本指令用于沿圆弧形焊缝的弧焊（定义了弧焊的圆弧形运动过程点）。

注意：加速度的设置在一些姿态下可能会导致 ArcC 在运动结束点停顿，如果影响焊接效果，可以在示教器上选择“焊接”->“焊接系统参数配置”->“加速度”里面调整加速度以避免停顿问题。

指令结构

ArcC(CirPoint, ToPoint, Speed, Seam, Weld, Weave, Zone, Tool, WObj, Tarray, TLoad)

指令参数

参数	数据类型	说明
CirPoint	robtarget	路径点
ToPoint	robtarget	目标点
Speed	speeddata	运动速度（焊接时无效）
Seam	seamdata	焊机启停参数，精简模式下该参数可选择设置。
Weld	welddata	焊接过程控制参数，精简模式下该参数可选择设置。
Weave	weavedata	焊接过程摆弧参数，精简模式下该参数可选择设置。
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
Tarray	triggdata 数据数组	跟踪数据
TLoad	loaddata	负载

指令示例

ArcC(P40, P50, v10, seam1, weld1, weave1, fine, tTorch, wobj0, event11 , load0)

机器人沿圆弧形焊缝的弧焊经过 P40 运动到 P50。

5.2.7 ArcCEnd 基于圆弧运动的弧焊结束

指令功能

本指令用于结束沿圆弧形焊缝的弧焊。

指令结构

ArcCEnd(CirPoint, ToPoint, Speed, Seam, Weld, Weave, Zone, Tool, WObj, Tarray, TLoad)

指令参数

参数	数据类型	说明
CirPoint	robtarget	路径点
ToPoint	robtarget	目标点
Speed	speeddata	运动速度（焊接时无效）
Seam	seamdata	焊机启停参数，精简模式下该参数可选择设置。
Weld	welddata	焊接过程控制参数，精简模式下该参数可选择设置。
Weave	weavedata	焊接过程摆弧参数，精简模式下该参数可选择设置。
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
Tarray	triggdata 数据数组	触发数据
TLoad	loaddata	负载

指令示例

ArcCEnd(P60, P70, v10, seam1, weld1, weave1, fine, tTorch, wobj0, event11 , load0)

机器人沿圆弧形焊缝的弧焊经过 P60 运动到 P70 点结束。

5.2.8 SpotJ 基于关节运动目标位置的点焊

指令功能

本指令用于工具中心点（TCP）以关节插补运动到给定目标位置，并且在目标位置激活点焊的过程。

指令结构

SpotJ (ToPoint, Speed, Weld, Zone, Tool, WObj, Tarr, TLoad, Time)

指令参数

参数	数据类型	说明
ToPoint	robtarget	目标点
Speed	speeddata	运动速度
Weld	welddata	焊接过程控制参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
Tarr	triggdata 数据数组	触发数据
TLoad	loaddata	负载
Time	num	焊接时间

指令示例

SpotJ (P10, v200, weld1, fine, tTorch, wobj0,nil, load0,500)

关节运动到 P10 点，焊接 500ms。

5.2.9 SpotL 基于线性运动目标位置的点焊

指令功能

本指令用于工具中心点（TCP）以直线插补运动到给定目标位置，并且在目标位置激活点焊的过程。

指令结构

SpotL (ToPoint, Speed, Weld, Zone, Tool, WObj, Tarr, TLoad, Time)

指令参数

参数	数据类型	说明
ToPoint	robtarget	目标点
Speed	speeddata	运动速度
Weld	welddata	焊接过程控制参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
Tarr	triggdata 数据数组	触发数据
TLoad	loaddata	负载
Time	num	焊接时间

指令示例

SpotL (P10, v200, weld1, fine, tTorch, wobj0,nil, load0,500)

直线运动到 P10 点，焊接 500ms。

5.2.10 ArcDotL 基于线性运动目标位置的连续点焊

指令功能

本指令用于工具中心点（TCP）以直线插补运动到给定目标位置，并且在运动过程中按照设置的条件等距离或者指定长度停顿点焊的过程。

指令结构

ArcDotL (StartPoint,ToPoint, Average,Len,Time, Weld, Speed,Zone, Tool, WObj, TLoad)

指令参数

参数	数据类型	说明
StartPoint	robtarget	焊缝起点
ToPoint	robtarget	焊缝结束点
Average	num	AwSegment： AwSegment =1,平均分段点焊 AwDistance： AwDistance =2,按设置的距离点焊
Len	num	Average = AwSegment，将焊缝平均分成 Len 段； Average = AwDistance，Len 为段的间隔距离，单位：mm
Time	num	点焊的时间，单位：ms
Weld	welddata	焊接过程控制参数
Speed	speeddata	运动速度
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

指令示例

ArcDotL (P10,P20, AwSegment,5,200, weld0, v200,,fine, tool0, wobj0, load0)

机器人沿直线运动（MoveL）到焊缝起点 P10，在沿直线运动到焊缝结束点 P20 的过程中将焊缝分为等距离的五段，并在每段的结束点停顿焊接 200ms。

5.2.11 ArcDotC 基于圆弧运动到目标位置的连续点焊

指令功能

本指令用于工具中心点（TCP）以圆弧插补运动到给定目标位置，并且在运动过程中按照设置的条件等距离或者指定长度停顿点焊的过程。

指令结构

ArcDotC (StartPoint,CirPoint,ToPoint, Average,Len,Time, Weld, Speed,Zone, Tool, WObj, TLoad)

指令参数

参数	数据类型	说明
StartPoint	robtarget	焊缝起点
CirPoint	robtarget	圆弧焊缝上的点
ToPoint	robtarget	焊缝结束点
Average	num	AwSegment： AwSegment =1,平均分段点焊 AwDistance： AwDistance =2,按设置的距离点焊
Len	num	Average = AwSegment，将焊缝平均分成 Len 段； Average = AwDistance，Len 为段的间隔距离，单位：mm
Time	num	点焊的时间，单位：ms
Weld	welddata	焊接过程控制参数
Speed	speeddata	运动速度
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

指令示例

ArcDotC (P10,P20,P30, AwDistance ,5,200,v200, weld1, fine, tTorch, wobj0, load0)

机器人关节运动（MoveJ）到焊缝起点 P10，在沿圆弧运动经过 P20 点到焊缝结束点 P30 的过程中，每间隔 5mm 停顿焊接 200ms。

5.2.12 ArcCircleL 使用 3 点焊接圆

指令功能

本指令用于工具中心点（TCP）以圆上 3 点规划出圆型运动轨迹，并且在焊接运动过程中改变焊枪姿态。

使用该指令焊接圆时，如果第六轴是从负到正运动，第六轴需要调整到 0 度以下，如果第六轴是从正到负运动，第六轴需要调整到 0 度以上，确保焊接时不会限位。

注意

示教点位时使 3 点在圆上均匀分布，这样能够确保圆的弧度更规整。

机器人位置变化大于 180，姿态变化小于 180。

机器人位置变化小于 180，姿态变化大于 180。

指令结构

ArcCircleL (StartPoint, CirPoint, ToPoint, Seam, Weld, Weave, Speed, Zone, Tool, WObj, TLoad, Flag)

指令参数

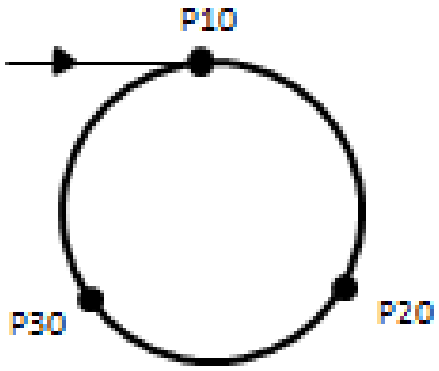
参数	数据类型	说明
StartPoint	robtarget	焊缝起点
CirPoint	robtarget	圆弧焊缝上的经过点
ToPoint	robtarget	焊缝结束点
Seam	seamdata	焊接启停参数，精简模式下该参数可选择设置。
Weld	welddata	焊接过程控制参数，精简模式下该参数可选择设置。
Weave	weavedata	焊接过程摆弧参数，精简模式下该参数可选择设置。
Speed	speeddata	运动速度
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

Flag	bool	true 或 false，默认：false. true：三点的姿态都参与圆的运动规划，运动过程中焊枪姿态随示教的点姿态变化； false：三点中只有起点的姿态参与圆的运动规划，运动时只以起点的姿态焊接圆，焊枪与半径成固定角度。

指令示例

ArcCircleL(P10,P20,P30, seam1, weld1, weave1, v200,fine, tTorch, wobj0, load0,false)

机器人直线运动到焊缝起点 P10 并起弧，再沿圆弧运动经过 P20，P30 点，再运动到焊缝结束点 P10，焊接成一个圆。如图所示。



5.2.13 ArcSegL 断续焊接直线焊缝

指令功能

本指令用于断续焊接直线型焊缝，并且可在焊接运动过程中改变焊枪姿态。

指令结构

ArcSegL(StartPoint, EndPoint, WeldLen, WeldNum, Speed, Seam, Weld, Weave, Zone, Tool, WObj, TLoad)

指令参数

参数	数据类型	说明
StartPoint	robtarget	焊缝起点
EndPoint	robtarget	焊缝结束点
WeldLen	num	焊缝上每段焊接的长度
WeldNum	num	焊缝上需要焊接的段数
Speed	speeddata	运动速度
Seam	seamdata	焊接启停参数。可选
Weld	welddata	焊接过程控制参数。可选
Weave	weavedata	焊接过程摆弧参数。可选
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载。可选

指令示例

ArcSegL (P10,P20,20,4, v200, seam0,weld0,weavw0,fine, tool0, wobj0, load0)

说明：

P10,P20 之间焊接 4 段 20mm 长的焊缝。

5.2.14 Multoffs 对指定点位进行位置，姿态偏移补偿

指令功能

在指定的工件坐标系中对指定点位补偿一个位置和姿态偏移量。

指令结构

point = Multoffs(point, offset)

返回值

point: 数据类型 Robtarget。

指令参数

参数	数据类型	说明
ToPoint	robtarget	机器人和外部坐标轴的目标点，定义为一个指定位置。
offset	pose	位置的坐标，姿态偏差，含一个笛卡尔坐标和一个姿态数据

指令示例

Pose = {{100, 0, 0},{1,0,0,0}}

MoveL(Multoffs(P20,Pose), v500, z10, tool2, wobj0)

说明：对 P20 点进行位置姿态插补。

5.2.15 Search_1D 机器人沿直线进行触觉搜索（智能寻位）

指令功能

Search_1D 是用触觉进行搜索的指令。当机器人以直线寻位时，Search_1D 用于搜寻监测信号并在搜到信号后机器人立即读取当前位置并返回当前位置距离期望结束点的姿态数据。Search_1D 和其上一条运动指令执行的都是 fine 点。

在焊接寻位时，在寻位模式下系统给焊丝通电，在机器人沿寻位轨迹移动过程中，一旦焊丝和工件接触时会产生接触信号，机器人停止移动。利用当前位置与程序设置位置的偏差值对路径进行修正，从而得出真实目标位置。焊枪会行走起点到希望结束点的距离的 2 倍长，如果移动过程中没有接触工件就会报错。

指令结构

Search_1D (Motgrp, Poffset, StartPoint, SearchPoint, Speed, Zone, Tool, WObj,[Load])

返回值

Poffset： 数据类型 pose。

寻位成功返回沟槽宽度，Poffset；失败将弹出错误提示信息。

指令参数

参数	数据类型	说明
Motgrp	num	0，左臂；1，右臂。
Poffset	pose	返回值。引入的偏移，寻位成功后返回的结果。
StartPoint	robtarget	搜索运动的起点。
SearchPoint	robtarget	机器人希望接触的部位。
Speed	speeddata	运动的速度数据。定义的 tcp 的运动速度。
Zone	zonedata	运动区域数据，描述生成的区域半径的大小。
Tool	tooldata	机器人移动时使用的 tool 数据。该 tool 中心点是指向指定目标位置的点。
WObj	wobjdata	指定机器人运动的坐标系。如果使用固定 tool 参数或协同的外部轴，则此参数必须指定以执行相对于工作对象的线性运动。

Load	loaddata	可选参数，指定机器人移动中使用的总负载，即末端法兰盘上安装的工具负载加上该工具正在处理工件的有效负载。如果使用了本参数，那么就不再考虑当前 tooldata 中的 loaddata；如果省略掉本参数，就表示可以忽略处理工件的负荷。

说明

在寻位模式下，使能焊机的寻位功能，焊机给焊丝通低压电，工件接地。

在机器人沿着寻位轨迹（直线）移动过程中，一旦焊丝和工件接触时，焊机会产生接触信号并反馈给机器人，机器人停止移动。利用机器人 TCP 的当前位置与程序设定位置的偏差值对路径进行修正，从而得出真实的目标位置。

使用条件：

1 工件表面必须没有铁锈、氧化层、油漆，或其他绝缘的涂层；

2 寻位前需要进行清枪和剪丝处理，以保证焊丝与工件的导电灵敏度。

指令示例

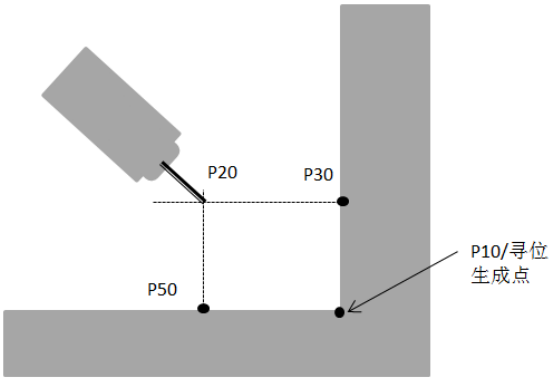
```
local pose20={}
Search_1D(pose20,P20,P30,v20,fine,shareTool3,wobj0,load0)
Search_1D(pose20,P20,P50,v20,fine,shareTool3,wobj0,load0)
MoveL(Multoffs(P10,pose20),v200,fine,shareTool3,wobj0,load0)
```

说明

1 运动到 P20 点，向 P30 搜索，接触到工件表面后回到 P20，记录位姿偏移；

2 带入位姿偏移，向 P50 点搜索，接触到工件表面后回到 P20，记录位姿偏移；

3 将获取的位姿偏移补偿到 P10，并运到的工件拐角点。



5.2.16 Search_Groove 机器人搜索沟槽的位置和宽度

指令功能

Search_Groove 是在智能寻位的基础上，焊枪针对沟槽进行工件表面寻位、沟槽边沿探查，沟槽左右边沿探查等一系列运动，计算出沟槽实际位置及宽度与编程时的偏移值，以得到真实沟槽位置及宽度尺寸。

指令结构

Search_Groove ([Motgrp], Poffset, SearchStop, StartPoint, CentrePoint, NomWidth, NomDepth, InitSchL, Speed, Zone, Tool, WObj,[Load])

返回值

Poffset： 数据类型 pose。

寻位成功返回沟槽宽度，Poffset；失败将弹出错误提示信息。

指令参数

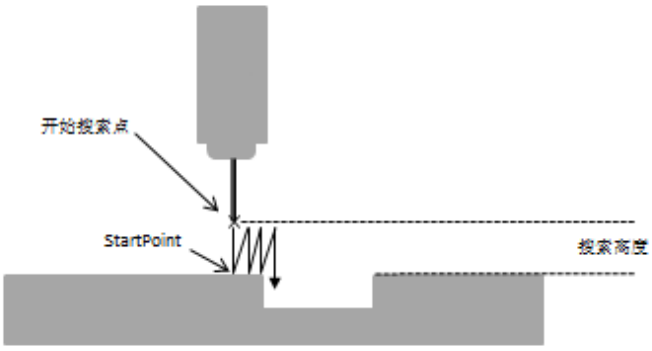
参数	数据类型	说明
Motgrp	num/string	可选参数，可以选择 Motgrp 作参数，也可以选择 Mec_name 作参数。若选择 Motgrp：0—左臂，1—右臂，如果不设置，则默认为左臂。若选择 Mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”。
Poffset	pose	搜寻成功后返回的相对于 CentrePoint 的位姿偏移结果。
SearchStop	bool	搜寻结束后是否停在沟槽中间。 false: 停在示教的 CentrePoint， true: 停在搜索产生的中间点。
StartPoint	robtarget	搜索运动的起点。
CentrePoint	robtarget	沟槽的中心点。
NomWidth	num	单位：mm。期望的沟槽宽度。
NomDepth	num	单位：mm。搜索深度，这个参数影响沟槽宽度的搜索。
InitSchL	num	单位：mm。搜索高度。起始点上方的开始搜索高度。
Speed	speeddata	运动的速度数据。定义的 tcp 的运动速度。
Zone	zonedata	运动区域数据，描述生成的区域半径的大小。

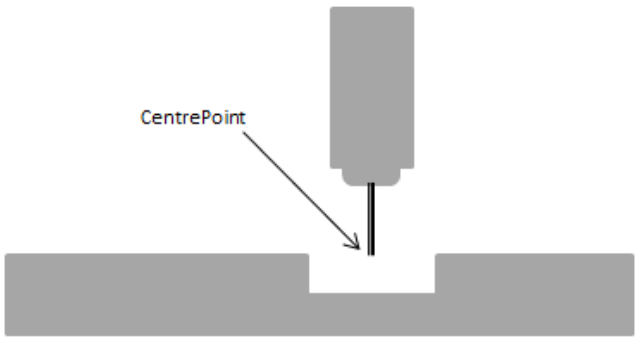
Tool	tooldata	机器人移动时使用的 tool 数据。该 tool 中心点是指向指定目标位置的点。
WObj	wobjdata	指定机器人运动的坐标系。如果使用固定 tool 参数或协同的外部轴，则此参数必须指定以执行相对于工作对象的线性运动。
Load	loaddata	可选参数，指定机器人移动中使用的总负载，即末端法兰盘上安装的工具负载加上该工具正在处理工件的有效负载。如果使用了本参数，那么就不再考虑当前 tooldata 中的 loaddata；如果省略掉本参数，就表示可以忽略处理工件的负荷。

说明

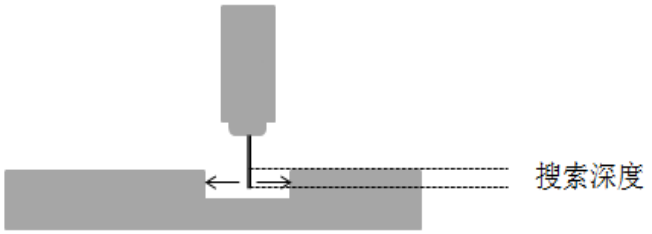
在寻位模式下，使能焊机的寻位功能，焊机给焊丝通低压电，工件接地。

1 焊枪在距离 StartPoint(开始点)上方（InitSchL 设置的高度）开始向工件表面运动，接触工件后向 CentrePoint(中心点)方向拉起再重复以上动作，直到进入沟槽后焊枪在原始接触高度上将无法触到工件，视为进入沟槽。搜索的最大距离为开始点到中心点长度的两倍。





2 搜索到沟槽后，运动到中心点下面所设置的搜索深度位置，分别向左右运动，计算出实际的中心位置。向左或向右的最大搜索距离为所设置期望宽度的两倍。



指令示例

```
local poffset = {}
local width, poffset = Search_Groove (poffset, true, P30, P40, 10, 3, 8,
v200,fine,shareTool3,wobj0,load0)
MoveL(Multoffs(P40,poffset),v200,fine,shareTool3,wobj0,load0)
```

在 P30 点上方 10mm 位置开始跳跃式向 P40 点跳跃运动。搜索到沟槽后运动到 P40 点下方 3mm 处，向左搜寻最大距离 8mm，再向右搜寻最大距离 8mm，最好停在沟槽中心点位置。

5.2.17 SetArcMode 设置焊机工作模式

指令功能

在 DeviceNet 通信模式下，设置焊机的工作模式。

指令结构

SetArcMode(num)

指令参数

参数	数据类型	说明
num	num	该值对应焊机的工作模式定义，不同厂家可能定义不同。 麦格米特焊机定义如下： 1 直流一元化； 2 脉冲一元化； 3 JOB 模式； 4 近控模式； 5 分别模式。

指令示例

SetArcMode(0)

对应麦格米特焊机使用 DeviceNet 焊接通信，将焊机的工作模式设置为一元化直流。

5.2.18 SetJobNumber 设置焊机工作号

指令功能

在 DeviceNet 通信模式下，通过 job 号选择焊机的焊接参数。

指令结构

SetJobNumber (num)

指令参数

参数	数据类型	说明
num	num	该值对应焊机的中保存的焊接相关的电流，电压参数代码。

指令示例

SetJobNumber (20)

对应麦格米特焊机使用 DeviceNet 焊接通信，使用焊机的保存的第 20 号参数焊接。

5.2.19 SetVAS 设置焊接参数

指令功能

设置焊接电压，焊接电流，焊接速度等。

使用该指令可以设置全局的焊接电压，电流，速度，以代替 welddata 中的电压，电流，速度。

如果要取消该指令的设置值，需要再次调用该指令，并将电压，电流，速度都设置为 0.
如： SetVAS(0, 0, 0)。

注意

如果对智殷的焊接指令不熟悉，不建议使用该指令，否则可能会在使用中导致出差错。

指令结构

SetVAS (V, A, S)

指令参数

参数	数据类型	说明
V	num	焊接电压
A	num	焊接电流
S	num	焊接速度

指令示例

SetVAS (18, 90, 9)

设置焊接电压为 18V，电流为 90A，速度 9mm/s。

SetVAS (0, 0, 0)

取消以上 SetVAS 的设置。

5.2.20 焊接指令的错误描述

当焊接系统使用过程中发生故障时，错误描述会推送到示教器上显示出来，焊接系统常见的错误描述如下表所示。

错误描述

序号	错误说明
1	"焊接异常"：在焊接过程中断弧，并且重复起弧失败。请检查工作环境后，重新起弧焊接。
2	"检测到焊枪碰撞"：焊枪疑似碰到障碍物，请关闭焊机，检查工作环境后，重新起弧焊接。
3	"焊机电压异常"：检测到焊接电压信号异常，请检查 IO 设备和焊机。
4	"焊机电流异常"：检测到焊接电流信号异常，请检查 IO 设备和焊机。
5	"焊接异常"：到在焊接过程中，焊接异常信号触发，请检查工作环境后，重新起弧焊接。
6	"焊接异常"：检测到送气信号异常，请检查 IO 设备和焊接。
7	"焊接异常"：检测到送丝信号异常，请检查 IO 设备和焊机。
8	"焊接异常"：检测到焊接失败，请检查工作环境。
9	"焊接指令运行出错"
10	"焊机异常"：检测到焊机异常，请检查焊机。
11	"焊机异常"：检测到焊机停止时出错，请关闭电源，检查焊机。
12	"焊机异常"：检测到在焊接过程中出错，请关闭电源，检查焊机。
13	"焊机异常"：检测到焊机与机器人通信异常。
14	"焊机异常"：检测到配置的电流电压校准范围错误。
15	"焊机异常"：检测给定的电流电压超出范围。

5.3 激光跟踪指令

5.3.1 LaserOn 开启激光跟踪

指令功能

本指令用于开启激光焊缝跟踪，建立与激光器的连接。

指令结构

LaserOn()

指令参数

无

指令示例

LaserOn()
LaserOff()

5.3.2 LaserOff 关闭激光跟踪

指令功能

本指令用于关闭激光焊缝跟踪，断开与激光器的连接。

指令结构

LaserOff()

指令参数

无

指令示例

LaserOn()
LaserOff()

5.3.3 LaserSet 设置激光焊缝跟踪参数

指令功能

本指令用于设置激光焊缝跟踪参数

指令结构

LaserSet (ParaIndex,YOffset,ZOffset,FilterEnable,FilterThreshold,FilterLength,EndDetect, PassLength)

指令参数

参数	数据类型	说明
ParaIndex	num	焊缝参数序号，与上位机焊缝跟踪系统中的参数序号对应，默认为 1。
YOffset	num	y 方向偏移量补偿，默认为 0。
ZOffset	num	z 方向偏移量补偿，默认为 0。
FilterEnable	bool	是否开启噪点过滤。
FilterThreashold	num	噪点过滤的阈值，默认为 10，即抖动如果限制在 10 以内，则会被过滤掉，大于该值则系统会报错并停止焊接。
FilterLength	num	最大过滤长度。实际抖动大于该长度，系统会报错。
EndDetect	bool	自动检测结束点，根据焊缝识别结果，自动识别结束点。
PassLength	num	在 passLength 以内不检测结束点，默认为 100。

指令示例

```
LaserOn()
LaserSet(1, 0, 0, false, 10, 0, false, 100)
LaserOff()
```

5.3.4 LaserSearch 激光焊缝搜索

指令功能

本指令用于焊缝搜索，搜索到的结果会保存到指令中。

指令结构

LaserSearch(P10, Distance, P20, Speed, Tool, Wobj, Load)

指令参数

参数	数据类型	说明
P10	robtarget	寻位目标点，和当前点一起决定寻位方向。
Distance	num	寻位距离，当机器人进行寻位时，沿指定的寻位方向前进的最大距离。超过此距离未找到焊缝，则告警。默认为 0，表示只原地搜索。
P20	robtarget	用于保存寻位结果。
Speed	speeddata	搜索时 TCP 运行的速度。
Tool	tooldata	指定机器人运动时参照的工具。
Wobj	wobjdata	指定机器人运动时参照的工件坐标系。
Load	loaddata	指定机器人工具的重量以及相关参数。

指令示例

```
LaserOn()
LaserSet(1, 0, 0, false, 10, 0, false, 100)
LaserSearch(P10, 0, P20, v10, tool1, wobj1, load0)
MoveL(P20, v50, z0, tool1, wobj1)
LaserOff()
```

5.3.5 ArcLOnline 激光焊缝跟踪焊接

指令功能

本指令用于激光焊缝跟踪焊接。

指令结构

ArcLOnline(P10, Speed, Zone, Tool, Wobj)

指令参数

参数	数据类型	说明
P10	robtarget	指定的是指令运行的目标位置点。
Speed	speeddata	搜索时 TCP 运行的速度。
Zone	zonedata	指定机器人运动路径的转弯半径。
Tool	tooldata	指定机器人运动时参照的工具。
Wobj	wobjdata	指定机器人运动时参照的工件坐标系。

指令示例

```
LaserOn()
LaserSet(1, 0, 0, false, 10, 0, false, 100)
ArcLOnline(P10, v10, z0, tool1, wobj1)
LaserOff()
```

5.4 电弧跟踪指令

5.4.1 ArcVolTrackOn 开启电弧跟踪

指令功能

本指令用于开启电弧跟踪，开启电弧跟踪后，机器人将根据摆焊过程中采集的电流数据，对示教路径进行修正，开启前确保焊机电流通过 DeviceNet 进行采样，焊接指令采用的是摆焊指令。

指令结构

ArcVolTrackOn([sn])

指令参数

参数	数据类型	说明
sn	num	可选参数，如果不设置或者设置为 0，则不保存路径。 用于记录焊接过程中的路径保存的内存号。范围：1~10。 机器人系统提供 10 块内存用于保存焊接路径，每块内存保存 20000 个点，两点之间的间隔为 5mm。

指令示例

ArcVolTrackOn()

ArcVolTrackOff()

开启电弧跟踪，不保存路径到内存，再关闭电弧跟踪。

5.4.2 ArcVolTrackOff 关闭电弧跟踪

指令功能

本指令用于关闭电弧跟踪，关闭电弧跟踪后，机器人以示教路径进行焊接作业，而不对路径进行修正。

指令结构

ArcVolTrackOff()

指令参数

无

指令示例

```
ArcVolTrackOn()
    --开启弧压跟踪
ArcLStart(P4,v2000,shareSeam8,shareWeld1,shareWeave8,fine,tool0,wobj0,nil,load1)
ArcL(P5,v20,shareSeam8,weld2,weave0,z0,tool0,wobj0,nil,load1)
ArcLEnd(P20,v20,shareSeam8,shareWeld1,shareWeave8,fine,tool0,wobj0,nil,load1)
ArcVolTrackOff()
    --关闭弧压跟踪
local function GLOBALDATA_DEFINE()
ROBTARGET("P4",{10.900000,3.380000,5.170000},{0.189157,-0.966237,0.173172,-0.024801},{1,-2,1,0},{0,0,0,0,0,0,0},0)
ROBTARGET("P5",{5.820000,76.070000,9.680000},{0.189148,-0.966238,0.173180,-0.024797},{1,-2,1,0},{0,0,0,0,0,0,0},0)
ROBTARGET("P6",{3.950000,265.350006,9.680000},{0.274671,0.916425,0.289344,-0.031631},{1,-1,0,0},{0,0,0,0,0,0,0},0)
end
print("The end!")
```

5.4.3 ArcRepL 多道指令

指令功能

本指令 ArcRepL 用于焊接存储的运动轨迹路径。可以用于一个完整的焊缝或一个焊缝的一部分。机器人按照轨迹数据中的存储路径运动焊接。

ArcRepL 指令用于多道焊，或重走存储的焊接路径，不设定每个后续的焊道参数。multipass 数据结构中设置运动路径方向、焊缝开始和结束的偏移、Y 和 Z 路径的偏移以及 Y 和 X 焊枪旋转角度信息。

指令结构

ArcRepL(Start,End,NoProcess,Offset,StartInd,EndInd,Speed,Seam,Weld,Weave,Zone,Tool,Wobj,Sn,Load)

指令参数

参数	数据类型	说明
Start	bool	用于定义存储路径的起点是否为焊接的起始点，为 true 时则在该点停止运动并起弧，false 则不起弧。
End	bool	用于定义存储路径的结束是否为焊接的结束点，为 true 时则在该点停止运动并熄弧，false 则不熄弧。
NoProcess	bool	true: 机器人之走存储路径的轨迹，不焊接。 false: 机器人之走存储路径的轨迹并焊接。
Offset	multipass	用于设置焊缝的首尾偏移，路径偏移，焊枪角度等信息。
StartInd	num	设置从存储路径开始运动的起始点号，默认 1。
EndInd	num	设置从存储路径运动结束的存储点号，默认 1。若为负数，则从存储的最好一个点倒数计算。
Speed	speeddata	运动速度
Seam	seamdata	焊接启停参数　可选
Weld	welddata	焊接过程控制参数　可选
Weave	weavedata	焊接过程摆弧参数　可选
Zone	zonedata	转角参数
Tool	tooldata	TCP 数据
WObj	wobjdata	工件数据

Sn	num/string	焊缝存储的内存号或名称
Load	loaddata	指定机器人工具的重量以及相关参数。

指令示例

ArcVolTrackOn(1) --开启弧压跟踪，并将焊缝路径保存在内存号 1 的区域

ArcLStart(P4,v2000,shareSeam8,shareWeld1,shareWeave8,fine,tool0,wobj0,nil,load1)

ArcL(P5,v20,shareSeam8,weld2,weave0,z0,tool0,wobj0,nil,load1)

ArcLEnd(P20,v20,shareSeam8,shareWeld1,shareWeave8,fine,tool0,wobj0,nil,load1)

ArcVolTrackOff() --关闭弧压跟踪

ArcRepL(true,true,false,m1,1,1,v200,shareSeam0,shareWeld0,shareWeave0,z5,tool0,wobj0,1,load1)

以上程序是将焊缝 P4-P5-P20 的路径，以 5mm 的间隔保存在内存号为 1 的位置。

焊接完成后使用定义为 m1 的 multipass 参数执行 ArcRepL 焊接。

5.4.4 MpSavePath 保存路径到文件

指令功能

本指令用于将内存中存储的路径保存到文件。然后可以使用 MpLoadPath 指令将该文件加载到内存中。如果必须稍后执行复焊操作，则可以使用此功能。只有在使用 ArcRepL 指令复焊此路径之前存储另一个路径时，才需要将路径保存到文件。保存快速文件的默认路径是 usr/log。

指令结构

MpSavePath(FileName, SN)

指令参数

参数	数据类型	说明
FileName	string	需要保持的文件名称
sn	num	用于记录焊接过程中的路径保存的内存号。范围：1~10。 机器人系统提供 10 块内存用于保存焊接路径，每块内存保存 20000 个点，两点之间的间隔为 5mm。

指令示例

ArcVolTrackOn(1)　--开启弧压跟踪，并将焊缝路径保存在内存号 1 的区域

ArcLStart(P4,v2000,shareSeam8,shareWeld1,shareWeave8,fine,tool0,wobj0,nil,load1)

ArcL(P5,v20,shareSeam8,weld2,weave0,z0,tool0,wobj0,nil,load1)

ArcLEnd(P20,v20,shareSeam8,shareWeld1,shareWeave8,fine,tool0,wobj0,nil,load1)

ArcVolTrackOff()　--关闭弧压跟踪

MpSavePath(“seam1”, 1)　--保存路径到文件“seam1”中

以上程序是将焊缝 P4-P5-P20 的路径，以 5mm 的间隔保存在内存号为 1 的位置，并将存储的路径保存到文件“seam1”中。

5.4.5 MpLoadPath 从文件加载路径

指令功能

MpLoadPath 用于将存储在文件中的路径加载到内存中，该文件先前是用 MpSavePath 指令存储的。

指令结构

MpLoadPath (FileName)

指令参数

参数	数据类型	说明
FileName	string	需要保持的文件名称

指令示例

MpLoadPath (“seam1”) --加载文件“seam1”中保存的路径到内存

ArcRepL(true,true,false,m1,1,1,v200,shareSeam0,shareWeld0,shareWeave0,z5,tool0,wobj0, “seam1”,load1)

以上程序为从文件“seam1”中加载保存的路径， ArcRepL 指令使用定义为 m1 的 multipass 参数执行轨迹运动。

6 激光切割专用指令

6.1 激光切割数据类型

6.1.1 LSCUTDATA 激光切割数据

激光切割数据，用于控制激光切割工艺，包含穿孔和切割阶段用于控制头、激光、切割气体等切割参数。

数据结构

```
LSCUTDATA
{
        CircleDiam;
    SideLength1;
    SideLength2;
    CutSpeed;
    LaserPower;
    DutyCycle;
    LaserOnDelay;
    LaserOffDelay;
    GasFlow;
    CutInType;
    CutInScale;
    CutOutType;
    CutOutScale;
    CutType;
    KerfComp;
    StopTime;
    FollowHeight;
    FollowSensit;
}
```

参数说明

序号	参数名	单位	默认值	说明
1	CircleDiam	mm	10	切割圆的直径
2	SideLength1	mm	0	边长 1，若为 0 或负值则为第 1 点和第 2 点的距离
3	SideLength2	mm	0	边长 2，若为 0 或负值则为第 2 点和第 3 点距离
4	CutSpeed	mm/s	100	切割速度
5	LaserPower	%	100	切割功率百分比,范围：0~100
6	DutyCycle	%	100	切割 pwm 百分比,范围：0~100
7	LaserOnDelay	ms	200	切割开光延时时间
8	LaserOffDelay	ms	200	切割关光延时时间
9	GasFlow	%	100	切割气体压力百分比,范围：0~100
10	CutInType		1	0：无引入，1：圆弧，2：直线
11	CutInScale	%	100	引线长度和角度百分比
12	CutOutType		0	0：无引出，1：圆弧，2：直线
13	CutOutScale	%	0	引线长度和角度百分比
14	CutType		1	1：内切，2：外切
15	KerfComp	mm	0	数值为正图形越大，为负越小
16	StopTime	ms	0	切割到达启动点停顿时间（预留）
17	FollowHeight	mm	2	切割头离工件距离(预留)
18	FollowSensit	%	100	切割头灵敏度百分比（预留）

6.1.2 MULTILAYERDATA 多层多道阵列数据

数据结构

多层多道阵列数据，用于激光熔覆，切割多层，多道，阵列的参数指定设置。其数据结构定义如下：

```
{
    int LsYMoveNum;
    BOOL LsYLenSel;
    float  LsYMoveDistance;
    int LsZMoveNum;
    float  LsZMoveDistance;
    float  LsZPowerDecScale;
    int LsYSuspendNum;
    float  LsYSuspendDeviation;
    int LsYSuspendTime;
    int LsZSuspendNum;
    int LsZSuspendTime;
    int LsXArrNum;
    float  LsXArrDistance;
    int LsYArrNum;
    float  LsYArrDistance;
    float  LsPowerScale;
}
```

数据参数

多层多道阵列数据包含 16 个数据：

1、LsYMoveNum Y 方向道数或长度

数据类型：int

复用参数，X-Y 平面，Y 方向一道道平移行走的数量或者熔覆的长度。用于长度时单位：mm。

2、LsYLenSel 道数或长度选择

数据类型：bool

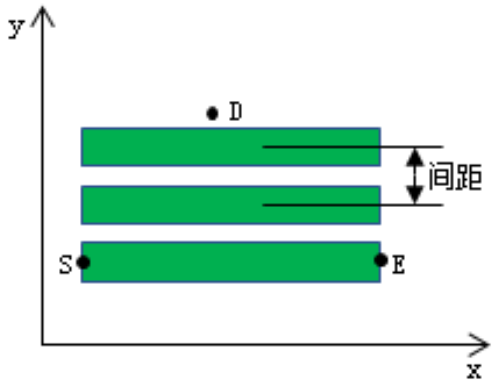
用于选择 LsYMoveNum 所设置的值是融覆道数或长度。

true：长度； false：道数。

3、LsYMoveDistance Y 方向平移距离

数据类型：float

Y 方向每次平移的距离，单位：mm。



4、LsZMoveNum Z 方向数量

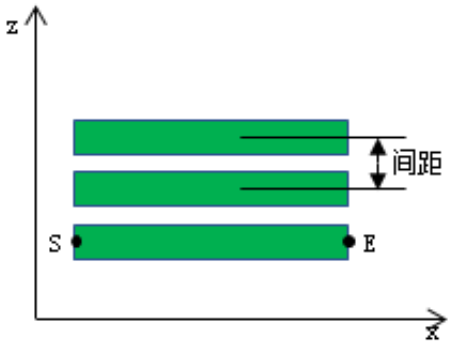
类型：int

Z 方向一道道平移行走的数量。

5、LsZMoveDistance Z 方向平移距离

数据类型：float

Z 方向每次平移的距离, 单位：mm。



6、LsZPowerDecScale 激光功率衰减比例系数

数据类型：float

每递增一层，激光功率衰减比例，单位：%。

例如：

10，第一层输出 CUT 数据中所设置的光功率的 100%，第二层输出 110%，第三层输出 120%。

-10，第一层输出 CUT 数据中所设置的光功率的 100%，第二层输出 90%，第三层输出 80%。

7、LsYSuspendNum Y 方向运动的停顿道数百分值（中断点）

数据类型：int

用于设置沿着 Y 方向行走 n 道后暂停，单位：%。

例如：

总道数 100，设置值 50，机器人将在 100*50%处暂停。

8、LsYSuspendDeviation Y 方向停顿偏移比例

数据类型：float

每递增一层，相对上一层停顿点的偏移比例系数，单位：%。

例如：

如果 LsYSuspendNum 设置为 50，LsYSuspendDeviation 设置为：

10，第一层 Y 方向停顿点 50%，第二层 60%，第三层输出 70%，最大 100%。

-10，第一层 Y 方向停顿点 50%，第二层 40%，第三层输出 30%，最小 0。

9、LsYSuspendTime Y 方向运动的停顿时间

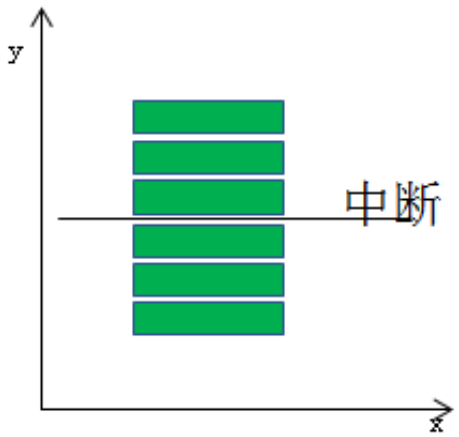
数据类型：int

沿着 Y 方向行走 n 道后暂停，等待设置的时候后继续下一道。单位：ms。

10、LsZSuspendNum Z 方向运动的停顿层数（中断点）

数据类型：int

用于设置沿着 Z 方向行走 n 层后暂停。



11、LsZSuspendTime Z 方向运动的停顿时间

数据类型：int

沿着 Z 方向行走 n 层后暂停，等待设置的时候后继续下一层。单位：ms。

12、LsXArrNum 阵列 X 方向个数

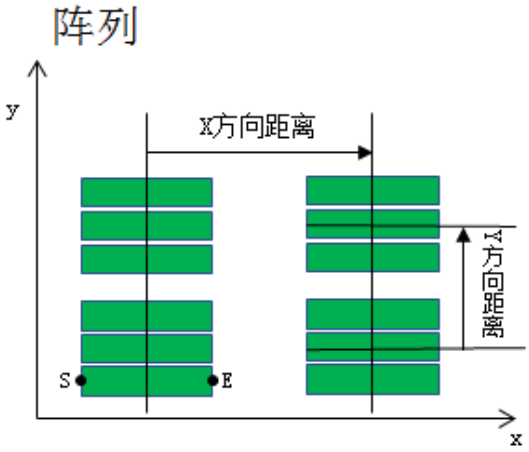
数据类型：int

用于设置 X-Y 平面在 X 方向重复操作的次数。

13、LsXArrDistance 阵列 X 方向距离

数据类型：float

用于设置 X-Y 平面在 X 方向执行重复操作时每次平移的间隔。单位：mm。



14、LsYArrNum 阵列 Y 方向个数

数据类型：int

用于设置 X-Y 平面在 Y 方向重复操作的次数。

15、LsYArrDistance 阵列 Y 方向距离

数据类型：float

用于设置 X-Y 平面在 Y 方向执行重复操作时每次平移的间隔。单位：mm。

16、LsPowerScale 多道衔接时激光功率输出比例系数（范围 0-100%）

数据类型：float

用于激光熔覆时完成一道运动后向下一道平移时候时输出的激光功率乘上一个比例系数。

输出的激光功率 = 当前激光功率 * LsPowerScale%。

指令示例

MULTILAYERDATA("multilayer1",10,false,2,2,2,0,50,0,1000,0,0,2,350,2,50,100)

LsCladdingPlane (S, E, D, v100, multilayer1, cut0, fine, tool0, wobj0, load0)

说明

1 在待熔覆的物体平面试教第一道的起点(S)，结束点(E)，再向平移方向试教第三点(D)。

2 一个熔覆面 Y 方向由 10 道 2mm 宽组成，Z 方向有 2 层，每层 2mm;

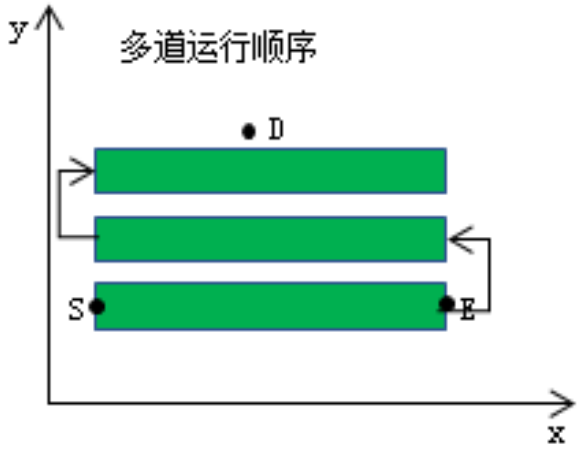
3 Y 方向第 5 道完成后关激光等待 1000ms，再继续完成后面的 5 道；

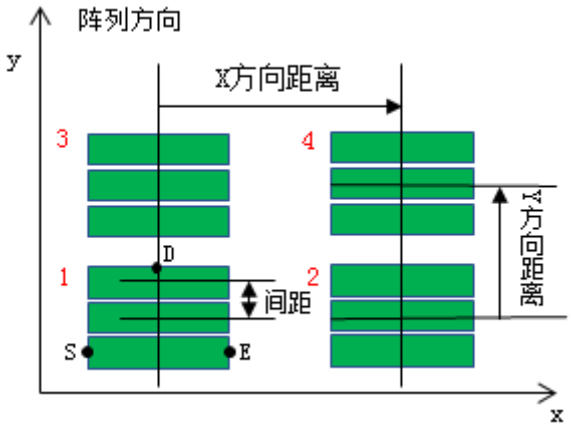
4 第一层完成后，再回到 S 点，激光枪抬高 2mm，开始熔覆第 2 层，并且到第 5 道完成后等待 1000ms；

5 第一个熔覆面熔覆完成后，沿着 X 方向平移 350mm，开始下一个熔覆面的工作；

6 待 X 方向两个熔覆面完成后，激光器回到 X 起点并沿 Y 方向平移 50mm，开始熔覆 Y 方向上的两个熔覆面。

7 熔覆阵列顺序：1 - 2 - 3 – 4。





6.2 激光切割指令

6.2.1 LsCutCircleL 基于圆形轨迹的激光切割

指令功能

本指令用于沿直线路径走到切割接近点，然后沿圆形轨迹进行的激光切割。

指令结构

LsCutCircleL (AprPoint, CenPoint, Speed, Cut, Zone, Tool, WObj, TLoad)

指令参数

参数	数据类型	说明
AprPoint	robtarget	切割接近点
CenPoint	robtarget	圆心点
Speed	speeddata	运动速度（切割时无效）
Cut	cutdata	切割工艺参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

指令示例

LsCutCircleL (p0, p1, v1000, Cut1, fine, tLaser, wobj0, load0)

沿直线路径走到切割接近点 P0，然后以 P1 为圆心沿圆形轨迹进行的激光切割

6.2.2 LsCutShapeL 基于规则几何轨迹的激光切割

指令功能

本指令用于沿直线路径走到切割接近点，然后沿规则几何形状轨迹进行的激光切割。

指令结构

LsCutShapeL (AprPoint, ToPoint1, ToPoint2, ToPoint3, Speed, Shape, Cut, Zone, Tool, WObj, Tload)

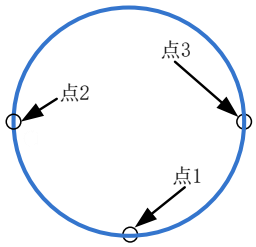
指令参数

参数	数据类型	说明
AprPoint	robtarget	切割接近点
ToPoint1	robtarget	几何形状控制点 1
ToPoint2	robtarget	几何形状控制点 2
ToPoint3	robtarget	几何形状控制点 3
Speed	speeddata	运动速度（切割时无效）
Shape	shapedata	0: 圆形；1：腰孔；3：三角形；4：矩形；5：正五边形；6：正六边形
Cut	cutdata	切割工艺参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

指令算法

1 圆算法

----圆必经点 1、点 2 和点 3，此时 CutData 中的直径数据不生效。



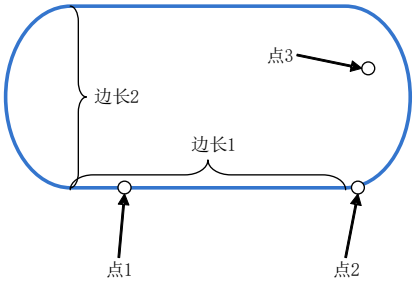
2 腰型孔算法

-----腰形孔直线和圆弧相切的地方肯定在点 2 位置

-----腰形孔的直线边肯定穿过点 1，直线边的方向就是点 1 和点 2 的连线，长度为边长 1。

-----点 3 主要为了指明跑道的方向，轨迹不一定会穿过该点。

-----跑道的圆弧直径为边长 2

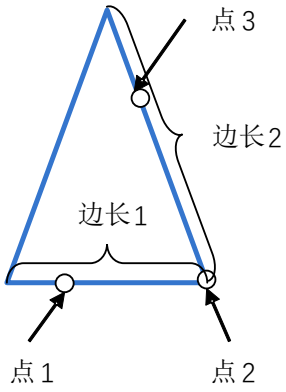


3 三角形算法

-----三角形的其中一个顶点就在点 2 位置

-----三角形的其中一边肯定穿过点 1，边的方向就是点 1 和点 2 的连线，长度为边长 1。

-----三角形的另外一边肯定穿过点 3，边的方向就是点 3 和点 2 的连线，长度为边长 2。



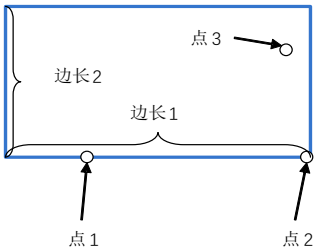
4 矩形算法

-----矩形的一个顶点肯定在点 2 位置

-----矩形的一个边肯定穿过点 1，边的方向就是点 1 和点 2 的连线，长度为边长 1。

-----点 3 主要为了指明矩形的方向，轨迹不一定会穿过该点。

-----矩形的另外一边长度边长 2



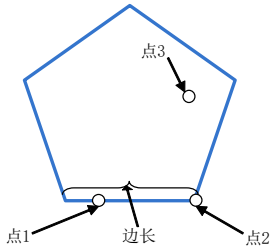
5 正五边形算法

-----正五边形的一个顶点肯定在点 2 位置

-----正五边形的一个边肯定穿过点 1，边的方向就是点 1 和点 2 的连线。

-----点 3 主要为了指明矩形的方向，轨迹不一定会穿过该点。

-----正五边形的边长为(边长 1+边长 2)/2



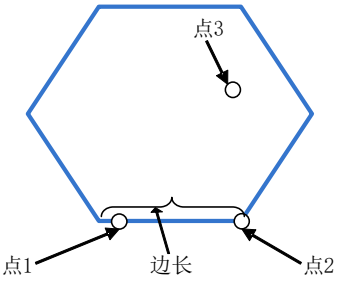
6 正六边形算法

-----正六边形的一个顶点肯定在点 2 位置

-----正六边形的一个边肯定穿过点 1，边的方向就是点 1 和点 2 的连线。

-----点 3 主要为了指明矩形的方向，轨迹不一定会穿过该点。

-----正六边形的边长为(边长 1+边长 2)/2



指令示例

LsCutShapeL (p0, p1, p2, p3, v1000, 4, Cut1, z10, tool1, wobj0, load0)

沿直线路径走到切割接近点 P0，然后沿以 p1 p2 p3 确定的矩形轨迹进行的激光切割。

6.2.3 LsCutLStart 基于自由轨迹的激光切割开始

指令功能

本指令用于开始沿自由轨迹进行的激光切割，自由轨迹路径灵活性很大，可以包括线性运动、圆弧运动和关节运动。

指令结构

LsCutLStart(ToPoint, Speed, Cut, Zone, Tool, WObj, TLoad)

指令参数

参数	数据类型	说明
ToPoint	robtarget	目标位置
Speed	speeddata	运动速度（切割时无效）
Cut	cutdata	切割工艺参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

指令示例

LsCutStart(P10, v200, cut, fine, tool1, wobj0, load0)

机器人开始沿自由轨迹运动到 p10 进行激光切割。

6.2.4 LsCutL 基于线性运动的激光切割

指令功能

本指令用于沿线性运动轨迹进行的激光切割过程。

指令结构

LsCutL(ToPoint, Speed, Cut, Zone, Tool, WObj, TLoad)

指令参数

参数	数据类型	说明
ToPoint	robtarget	目标位置
Speed	speeddata	运动速度（切割时无效）
Cut	cutdata	切割工艺参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

指令示例

LsCutL (P10, v20, cut1, fine, tLaser, wobj0, load0)

机器人沿直线运动到 P10 点。

6.2.5 LsCutLEnd 基于线性运动的激光切割结束

指令功能

本指令用于以线性运动结束沿自由轨迹进行的激光切割。

指令结构

LsCutLEnd (ToPoint, Speed, Cut, Zone, Tool, WObj, TLoad)

指令参数

参数	数据类型	说明
ToPoint	robtarget	目标位置
Speed	speeddata	运动速度（切割时无效）
Cut	cutdata	切割工艺参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

指令示例

LsCutLEnd (P10, v20, cut1, fine, tLaser, wobj0, load0)

线性激光切割运动到 P10 点结束。

6.2.6 LsCutC 基于圆弧运动的激光切割

指令功能

本指令用于沿圆弧运动轨迹进行的激光切割过程。

指令结构

LsCutC (CirPoint, ToPoint, Speed, Cut, Zone, Tool, WObj, TLoad)

指令参数

参数	数据类型	说明
CirPoint	robtarget	圆形路径圆周点
ToPoint	robtarget	目标位置
Speed	speeddata	运动速度（切割时无效）
Cut	cutdata	切割工艺参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

指令示例

LsCutC (P10, P20, v20, cut1, fine, tLaser, wobj0, load0)

机器人以圆弧经过 P10 运动到 P20 点。

6.2.7 LsCutCEnd 基于圆弧运动的激光切割结束

指令功能

本指令用于以圆弧运动结束沿自由轨迹进行的激光切割。

指令结构

LsCutCEnd (CirPoint, ToPoint, Speed, Cut, Zone, Tool, WObj, TLoad)

指令参数

参数	数据类型	说明
CirPoint	robtarget	圆形路径圆周点
ToPoint	robtarget	目标位置
Speed	speeddata	运动速度（切割时无效）
Cut	cutdata	切割工艺参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

指令示例

LsCutLEnd (P10, P20, v20, cut1, fine, tLaser, wobj0, load0)

机器人以圆弧经过 P10 运动到 P20 点结束。

6.2.8 LsWristCutCL 基于腕部运动的圆形轨迹的激光切割

指令功能

本指令用于沿直线路径走到切割接近点，然后沿圆形轨迹进行腕部运动的激光切割。

指令结构

LsWristCutCL (AprPoint, CenPoint, Speed, Cut, Zone, Tool, WObj, TLoad)

指令参数

参数	数据类型	说明
AprPoint	robtarget	切割接近点
CenPoint	robtarget	圆心点
Speed	speeddata	运动速度（切割时无效）
Cut	cutdata	切割工艺参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

指令示例

LsWristCutCL (p0, p1, v1000, Cut1, fine, tLaser, wobj0, load0)

沿直线路径走到切割接近点 P0，然后以 p1 为圆心，沿圆形轨迹进行腕部运动的激光切割。

6.2.9 LsCladdingPlane 激光熔覆

指令功能

本指令用于沿直线路径走到熔覆的起点，然后沿直线从起点到结束点，平移，再从结束点到起点一道道运行。

在一个 X-Y 面熔出一定宽度的涂层，然后根据 multilayerdata 的 Z 向参数设置，在 Z 方向上完成多层平移操作，熔出一定高度的涂层。

根据 multilayerdata 的阵列参数设置，在 X 方向或 Y 方向完成整体平移操作，将指定宽度的涂层平移到另外一个区域。

指令结构

LsCladdingPlane(StartPoint, EndPoint, DirPoint, Speed, Multilayer, Cut, Zone, Tool, WObj, TLoad)

指令参数

参数	数据类型	说明
StartPoint	robtarget	熔覆第一道的起点
EndPoint	robtarget	熔覆第一道的结束点
DirPoint	robtarget	方向点
Speed	speeddata	运动速度（切割时无效）
Multilayer	multilayerdata	多层多道阵列参数
Cut	cutdata	切割工艺参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据
TLoad	loaddata	负载

指令示例

MULTILAYERDATA("multilayer1",10,2,2,2,5,1000,0,0,2,350,2,50,50)

LsCladdingPlane (S, E, D, v100, multilayer1, cut0, fine, tool0, wobj0, load0)

说明

在待熔覆的物体平面试教第一道的起点(S)，结束点(E)，再向平移方向试教第三点(D)。

1 一个熔覆面 Y 方向由 10 道 2mm 宽组成，Z 方向有 2 层，每层 2mm;

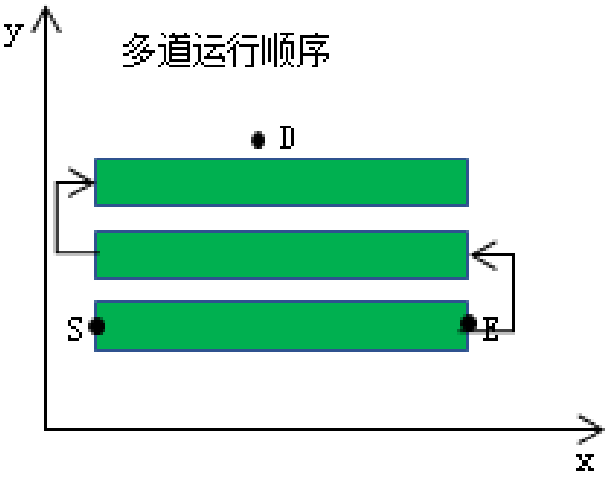
2 Y 方向第 5 道完成后关激光等待 1000ms，再继续完成后面的 5 道；

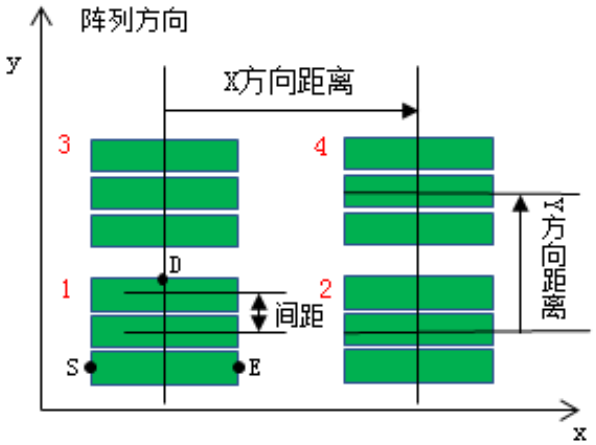
3 第一层完成后，再回到 S 点，激光枪抬高 2mm，开始熔覆第 2 层，并且到第 5 道完成后等待 1000ms；

4 第一个熔覆面熔覆完成后，沿着 X 方向平移 350mm，开始下一个熔覆面的工作；

5 待 X 方向两个熔覆面完成后，激光器回到 X 起点并沿 Y 方向平移 50mm，开始熔覆 Y 方向上的两个熔覆面。

6 熔覆阵列顺序：1 - 2 - 3 - 4





7 视觉专用指令

7.1 VisionOn 开启视觉系统

指令功能

本指令用于打开视觉系统，建立与指定视觉系统的连接。

指令结构

VisionOn(Id)

指令参数

参数	数据类型	说明
Id	num	视觉系统编号，该编号与视觉配置中的编号对应。编号从 0 开始。

指令示例

VisionOn (0)

VisionTrig(0,0,0)

VisionOff(0)

开启视觉系统，触发视觉拍照后关闭视觉系统。

7.2 VisionOff 关于视觉系统

指令功能

本指令用于关闭视觉系统并断开与指定视觉系统的连接。

指令结构

VisionOff(Id)

指令参数

参数	数据类型	说明
Id	num	视觉系统编号，该编号与视觉配置中的编号对应。编号从 0 开始。

指令示例

VisionOn(0)

VisionTrig(0,0,0)

VisionOff (0)

开启视觉系统，触发视觉拍照后关闭视觉系统。

7.3 VisionTrig 触发视觉拍照

指令功能

本指令用于触发视觉系统进行拍照并分析。

触发拍照成功返回 true，失败返回 false。

指令结构

flag = VisionTrig(Id, ProcId, ImgId)

返回值

flag: 数据类型 bool。

触发拍照成功返回 true，失败返回 false。

指令参数

参数	数据类型	说明
Id	num	视觉系统编号，该编号与视觉配置中的编号对应。编号从 0 开始。
ProcId	num	流程单元编号
ImgId	num	图像单元编号

指令示例

VisionOn(0)

VisionTrig (0, 0, 0)

VisionOff(0)

开启视觉系统，触发视觉拍照后关闭视觉系统。

7.4 VisionGetTarget 获取视觉识别的点位信息

指令功能

本指令用于获取视觉识别后的点位信息。点位信息的位置由视觉定位的 x，y 以及视觉配置中的抓取点 z 确定，点位信息的姿态由视觉配置中的抓取点姿态经过视觉定位的 rz 旋转后确定。

指令结构

VisionGetTarget(Id, P10)

指令参数

参数	数据类型	说明
Id	num	视觉系统编号，该编号与视觉配置中的编号对应。编号从 0 开始。
P10	robtarget	该点位由视觉系统自动计算，用户无需对该点位进行示教。

指令示例

```
VisionOn(0)    --连接视觉系统
VisionTrig(0)    --触发视觉拍照
VisionGetTarget (0, P10)    --获取视觉识别后的指令信息
MoveL(P10, v1000, z0, tool1, wobj1)    --线性运动到 P10 点
VisionOff(0)    --断开视觉系统连接
```

7.5 VisionGetString 获取视觉识别的字符串结果

指令功能

本指令用于获取视觉识别后的字符串结果。

指令结构

str = VisionGetString(Id, DataId)

返回值

str: 数据类型 string。

指令参数

参数	数据类型	说明
Id	Num	视觉系统编号，该编号与视觉配置中的编号对应。编号从 0 开始。
DataId	Num	数值序号，序号从 1 开始。

指令示例

```
VisionOn(0)
VisionTrig(0)
local str = VisionGetString (0, 1)
if str == “Rect” then
  -- do something.
elseif str == “Cube” then
  -- do another thing.
end
VisionOff(0)
```

如果 VisionTrig 后视觉系统返回的结果为“Rect,false,0,1”，系统会以“，”对字符串进行切割。

VisionGetString(0,1)返回“Rect”，VisionGetString(0,2)返回“false”。

以此类推。

7.6 VisionGetNumber 获取视觉识别的数值结果

指令功能

本指令用于获取视觉识别后的数值结果。

指令结构

num = VisionGetNumber(Id, DataId)

返回值

num: 数据类型 num。

指令参数

参数	数据类型	说明
Id	Num	视觉系统编号，该编号与视觉配置中的编号对应。编号从 0 开始。
DataId	Num	数值序号，序号从 1 开始。

指令示例

```
VisionOn(0)
VisionTrig(0)
local num = VisionGetNumber (0, 1)
if num == 100 then
  -- do something.
else
  -- do another thing.
end
VisionOff(0)
```

获取 0 号摄像头的序号 1 的数值，根据获取的结果做对应处理。

7.7 VisionSendCmd 向视觉系统发送

指令功能

本指令用于向视觉系统发送字符串命令。

指令结构

VisionSendCmd(Id, CmdString)

指令参数

参数	数据类型	说明
Id	num	视觉系统编号，该编号与视觉配置中的编号对应。编号从 0 开始。
CmdString	string	字符串格式的命令。

指令示例

VisionOn(0)

VisionSendCmd(0, “Record reference point 1”)

VisionOff(0)

开启视觉系统，向视觉系统发送字符串后关闭

8 折弯专用指令

8.1 BendTrack 折弯跟随指令

指令功能

BendTrack 用于围绕 wobj 的 x 轴做旋转。

参考坐标系，原点为刀口与槽口平面交线中点，折弯机折弯运动方向为 Z 轴正向，刀口方向为 Y 轴方向，朝外（正面面对折弯机）为 X 轴正向。

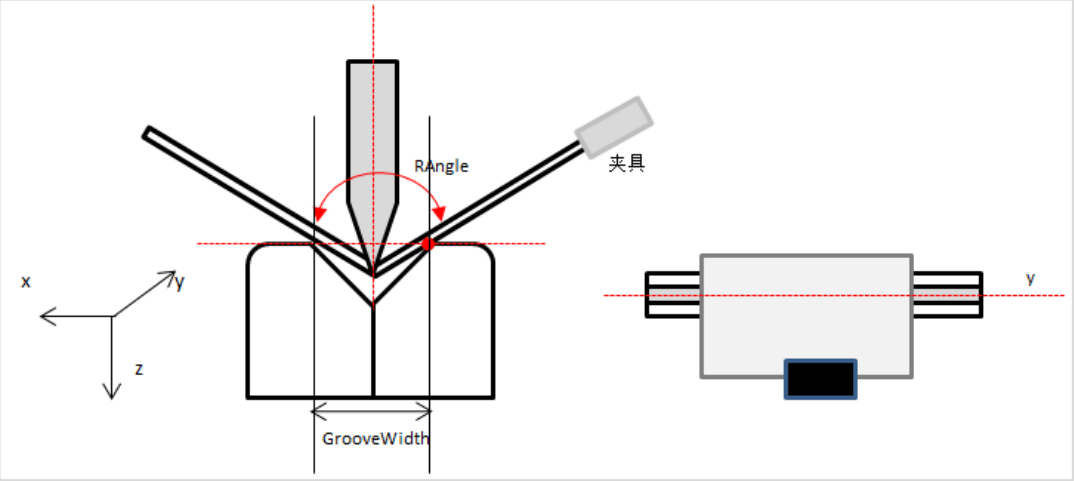
指令结构

BendTrack ([Motgrp/Mec_name], ToPoint, RAngle, BendSpeed, BoardThickness, GrooveWidth, Zone, Tool, WObj, [Load])

指令参数

参数	数据类型	说明
Motgrp/Mec_name	string	这个参数是可选参数，可以选择 Motgrp 赋值，也可以选择 Mec_name 赋值。若选择 Motgrp：0—左臂，1—右臂。如果不设置，则默认为左臂。若选择 Mec_name，即直接把机械单元的名称作为第一个参数，如“RL7”。
ToPoint	RobTarget	数据类型为 robtarget。机器人基于工件坐标系的一个指定位置。
RAngle	num	数据类型为 num。 折弯的旋转角度，为两个面的夹角，单位：度。
BendSpeed	num	数据类型为 num。上模沿着中心线向下运行的速度。单位：mm/s。
BoardThickness	num	数据类型为 num。板材厚度，单位：mm。
GrooveWidth	num	数据类型为 num。折弯机沟槽的宽度。
Zone	zoneData	数据类型 zonedata。运动区域数据，描述生成的区域半径的大小。

Tool	Tooldata	数据类型 tooldata。机器人移动时使用的 tool 数据。
WObj	wobjData	数据类型 wobjdata。指定机器人运动的工件坐标系。
Load	loadData	数据类型 loaddata。可选参数，指定机器人移动中使用的总负载，即末端法兰盘上安装的工具负载加上该工具正在处理工件的有效负载。如果使用了本参数，那么就不再考虑当前 tooldata 中的 loaddata；如果省略掉本参数，就表示可以忽略处理工件的负荷。



指令示例

BendTrack (P40, 90, 20, 0, 20, fine, shareTool3,rotwobj, load0)

示教工件坐标系 rotwobj，原点为刀口与槽口平面交线中点，折弯机上面运动方向为 Z 轴正向，刀口方向为 Y 轴方向，朝外（正面面对折弯机）为 X 轴正向。

设置所需折弯的角度 RAngle，上模折弯时运动的速度 BendSpeed，板材厚度 BoardThickness，沟槽宽度 GrooveWidth 等。

9 IO 系统配置

9.1 IO 系统概述

IO 系统是机器人控制器和外部设备(如 PLC)进行通信和数据交换的接口，数据的扫描和刷新是实时的。

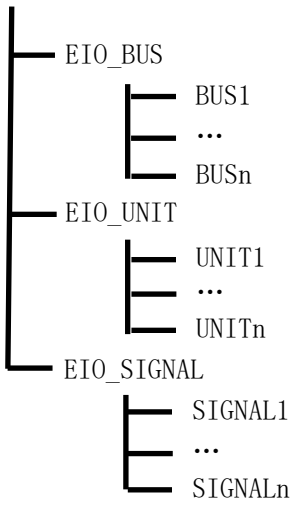
IO 系统目前支持以 EhterCAT 和 Modbus 为通信协议的 IO 设备，控制器在出厂的时候内嵌了华太的 IO 板，配置文件也相应的支持该板；如果需要支持外部的 PLC 主站，需要对系统的 IO 配置做出相应的修改。

IO 系统可以通过修改配置文件 EIO.cfg 来满足客户定制化的需求， 具体配置方法参见接下来的章节。

用户可通过指令系统中的 IO 指令来实现对 IO 数据的访问，具体的操作方法参见指令系统手册。

9.2 拓扑结构

IO 系统是一个树状结构， 它的一类根节点包括数据总线(bus)、数据采集板(unit)、数据点；数据总线描述控制器对外的通信接口，包括通信协议、通信参数；数据采集板描述基于特定数据总线的一个逻辑单元，一块数据采集板容纳多个数据采集点；数据点描述的是一个具体的数据，它可以是数字量(DI, DO)、模拟量(AI, AO)。



IO 系统的配置通过配置文件 EIO.cfg 来实现，可以参考缺省的 EIO.cfg 来定制该配置文件。

9.3 IO 系统配置

IO 的配置通过文本文件 EIO.cfg 来实现，在该文件中包含如下关键字：

- 根节点名字：EIO_BUS，EIO_UNIT， EIO_SIGNAL；
- 属性关键字：每种数据类型的属性，如 Name、BusType、Address、Port 等；

EIO 配置语法要求：

- 根节点名字后面必须跟“：”，以表示接下来的内容是描述该节点的信息。
- 所有属性关键字必须以”-“开头
- 根节点信息及每种数据必须是单独一行；
- 所有注释必须以”#”开始；
- 所有的名字必须是英文字符，字符长度小于 32。

9.3.1 数据总线

控制器支持以下数据总线：

总线	功能	说明
ECAT	通过网络接口把外部支持 EtherCAT 的 IO 设备连接到控制器，IO 设备作为从站设备；IO 设备必须提供相应的 XML 描述文件	目前支持华太 IO 板/新时达 IO 板
MODBUS TCP	通过网络接口和 PLC 主站实现通信	支持 Siemens PLC，适配其它支持 MODBUS TCP 通信的厂商 PLC
DEVICENET	通过 USB 和 DeviceNet 主站实现通信	广州致远 DeviceNet 设备已支持，其他支持 ZLG 通信定义的厂商也能支持
VIRTUAL	虚拟总线	虚拟 IO 信号
ECATGATEWAY	通过 EtherCAT 网关适配其它总线	目前支持 KUNBUS 品牌的 EtherCAT 转 Profibus/Profinet/Ethernet IP 三种总线转换网关
ETHERNETIP	通过网络接口，与 EthernetIP Scanner 扫描器设备或者 Adapter 适配器设备实现通信	机器人控制器作为 Scanner，可以与 KUNBUS 品牌的 Adapter IO 设备、米加尼克 EthernetIP 焊机通讯； 机器人控制器作为 Adapter，可以与基恩士 KV 系列 PLC 通讯。

在定义总线的时候，可以定义多个总线，但是总线类型（BusType）必须是以上类型之一；

需要配置的信息包括：总线名字、类型及其他相关信息：

总线	名字(Name)	类型(BusType)	地址(Address)	端口(Port)
EtherCAT	用户自定义	ECAT	NA	NA
Modbus	用户自定义	MODBUS	NA	NA
DeviceNet	用户自定义	DEVICENET	NA	NA

Virtual	用户自定义	VIRTUAL	NA	NA
ECATGATEWAY	用户自定义	ECATGATEWAY	NA	NA
ETHERNETIP	用户自定义	ETHERNETIP	NA	NA

示例

EIO_BUS:

-Name "Virtual" -BusType "VIRTUAL"

-Name "Ecat" -BusType "ECAT"

-Name "Modbus” -BusType "MODBUS"

-Name "DeviceNet” -BusType "DEVICENET"

-Name "EIPADAPTER" -BusType "ETHERNETIP"

-Name "EIPSCANNER" -BusType "ETHERNETIP"

9.3.2 数据采集板

数据采集板(unit)依附于一个特定的总线，板的配置信息如下：

名字(Name)	总线(Bus)	节点(Node)	地址(MacID)
用户可随意定义	总线名字，必须是之前定义的总线中的一个	节点序号，一条总线可以有多个节点，节点序号从 0 开始	物理地址，一个数据采集板，对应从站设备的物理地址

示例

EIO_UNIT:

-Name "BOARD10" -Bus "Virtual"

-Name "BOARD20" -Bus "Ecat" -Node "0"
-Name "BOARD30" -Bus "Modbus"

-Name "BOARD40" -Bus " DeviceNet" –MacID "0"

-Name "BOARD50" -Bus "EIPADAPTER" -Node "0"

-Name "BOARD60" -Bus "EIPSCANNER" -Node "0"

不是所有的数据采集板都需要这些配置信息，具体规则如下：

总线(Bus)	名字 (Name)	节点(Node)	地址(MacID)
VIRTUAL	√		
ECAT	√	√	
MODBUS	√		
DEVICENET	√		√
ECATGATEWAY	√	√	
ETHERNETIP	√		

9.3.3 数据点

定义具体的 IO 信号，具体信息保护如下：

名称	说明
名字(Name)	信号名字，英文字符串，字符数量小于 32，各信号名称不能重复
信号类型(SignalType)	以下字符串之一：DI、DO、AI、AO
数据采集板（Unit）	指定该信号位于的板(unit)
地址映射（UnitMap）	指定该信号的地址
缺省值（DefaultValue）	指定该信号的缺省值，该值仅对输出有效，系统启动后输出该指定值
关联（Connect）	关联到的系统信号，必须是同一种信号类型
数据长度（DataLen）	指定该信号在设备中占用的数据长度(单位：bit), 缺省为 1
关联触发类型（TrigType）	关联信号（Connect）的触发机制（上升沿、下降沿、高电平、低电平、电平变化）
交叉（Cross）	把一个输入信号的值交叉改变一个输出信号
交叉触发类型（TrigType）	交叉信号（Cross）的触发机制取值： RISE（上升沿）、FALL（下降沿）、HIGH（高电平）、LOW（低电平）、LEVEL（电平变化），默认值上升沿。
取反（Invert）	取反值为 TRUE 或者 FALSE，TRUE 表示取反生效，默认为 FALSE
最大值（MaxBitValue）	用于模拟量设置最大值
最小值（MinBitValue）	用于模拟量设置最小值，必须小于等于默认值

UnitMap 代表一个信号的地址或者是地址起始， 在不同总线类型中， 其含义是不一样的，所有地址的起始序号都是 0。

VIRTUAL 总线上的不同型号类型的映射地址都可以从 0 开始依次往后排，同一个节点(node)中的地址不能冲突；同一个节点上的信号（数字信号、模拟信号、输入、输出）共用一块地址空间，映射地址(unitmap)的值不能有冲突。

ECAT 总线的不同型号类型的映射地址基于不同的品牌单元映射地址从不同的数字开始，同一个节点(node)中的地址不能冲突；同一个节点上的信号（数字信号、模拟信号、输入、输出）共用一块地址空间， 映射地址(unitmap)的值不能有冲突。

注意

如果在出厂控制器自带华太 IO 的基础上增加 IO 模块，需要更换设备描述文件（xml 文件），请与技术支持联系。

以华太 IO 为例说明 ECAT 配置

- EIO 配置中 DI 的单元映射第一个模块从 0 到 7 共 8 个点，第二个模块从 10 到 17，后续依次类推。
- EIO 配置中 DO 的单元映射第一个模块从 0 到 7 共 8 个点，第二个模块从 8 到 15，后续依次类推。
- EIO 配置中 AI 的单元映射第一个模块从 3 到 6 共 4 个点，第二个模块从 10 到 13，后续依次类推。
- EIO 配置中 AO 的单元映射第一个模块从 1 到 4 共 4 个点，第二个模块从 6 到 9，后续依次类推。
- DI 两个模块之间的连接点不是衔接的，中间有两个状态位，DO 两个模块的点之间是衔接的；AI、AO 模块前面有几个点是状态位，其单元映射不是从 0 开始的。

MODBUS 总线地址数字信号和模拟信号对应的地址都是以 bit 为单位的，一个数字信号对应一个 bit 位；一个模拟信号对应于 PLC 中的 2 个 16 位寄存器， 共 32 个 bit， 地址（unitmap）对应一个信号的开始的 bit 地址； Modbus 所有的数据类型（模拟量和数字量的输入输出）共用一个地址空间，且模拟量和数字量的输入定义在一起，输出定义在一起。

注意

1 和西门子 PLC 通信，模拟信号的映射地址只能从 PLC 的第偶数个寄存器开始映射。

2 和西门子 PLC 通信时存在大小端问题，比如西门子 PLC 一个寄存器是 16 位，分为 0.0~0.7 和 1.0~1.7，EIO 文件中的映射地址是从 0~15，那么 0.0~0.7 和 8~15 一一映射，1.0~1.7 和 0~7 一一映射。

DEVICENET 的总线地址数字信号和模拟信号对应的地址也是基于 bit 位的，输出信号(DO/AO)共享一个地址空间, 输入信号(DI/AO)共享一个地址空间；数字信号占用一个 bit 位，模拟信号占用多个 bit 位。

示例

#Virtual 总线：

EIO_SIGNAL:

-Name "DI10_1" -SignalType "DI" -Unit "BOARD10" -UnitMap "0" -DefaultValue "1"

-Name "DI10_2" -SignalType "DI" -Unit "BOARD10" -UnitMap "1" -DefaultValue "1"

#EtherCAT 总线：

#DI:

-Name "DI20_01" -SignalType "DI" -Unit "BOARD20" -UnitMap "0" -DefaultValue "0" -DataLen "1"

-Name "DI20_02" -SignalType "DI" -Unit "BOARD20" -UnitMap "1" -DefaultValue "0" -DataLen "1"

-Name "DI20_03" -SignalType "DI" -Unit "BOARD20" -UnitMap "2" -DefaultValue "0" -DataLen "1"

-Name "DI20_04" -SignalType "DI" -Unit "BOARD20" -UnitMap "3" -DefaultValue "0" -DataLen "1"

-Name "DI20_05" -SignalType "DI" -Unit "BOARD20" -UnitMap "4" -DefaultValue "0" -DataLen "1"

-Name "DI20_06" -SignalType "DI" -Unit "BOARD20" -UnitMap "5" -DefaultValue "0" -DataLen "1"

-Name "DI20_07" -SignalType "DI" -Unit "BOARD20" -UnitMap "6" -DefaultValue "0" -DataLen "1"

-Name "DI20_08" -SignalType "DI" -Unit "BOARD20" -UnitMap "7" -DefaultValue "0" -DataLen "1"

#DO:

-Name "DO20_01" -SignalType "DO" -Unit "BOARD20" -UnitMap "0" -DefaultValue "0" -DataLen "1"

-Name "DO20_02" -SignalType "DO" -Unit "BOARD20" -UnitMap "1" -DefaultValue "0" -DataLen "1"

-Name "DO20_03" -SignalType "DO" -Unit "BOARD20" -UnitMap "2" -DefaultValue "0" -DataLen "1"

-Name "DO20_04" -SignalType "DO" -Unit "BOARD20" -UnitMap "3" -DefaultValue "0" -DataLen "1"

-Name "DO20_05" -SignalType "DO" -Unit "BOARD20" -UnitMap "4" -DefaultValue "0" -DataLen "1"

-Name "DO20_06" -SignalType "DO" -Unit "BOARD20" -UnitMap "5" -DefaultValue "0" -DataLen "1"

-Name "DO20_07" -SignalType "DO" -Unit "BOARD20" -UnitMap "6" -DefaultValue "0" -DataLen "1"

-Name "DO20_08" -SignalType "DO" -Unit "BOARD20" -UnitMap "7" -DefaultValue "0" -DataLen "1"

#AI

-Name "AI20_01" -SignalType "AI" -Unit "BOARD20" -UnitMap "3" -DefaultValue "0"

-Name "AI20_02" -SignalType "AI" -Unit "BOARD20" -UnitMap "4" -DefaultValue "0"

-Name "AI20_03" -SignalType "AI" -Unit "BOARD20" -UnitMap "5" -DefaultValue "0"

-Name "AI20_04" -SignalType "AI" -Unit "BOARD20" -UnitMap "6" -DefaultValue "0"

#AO

-Name "AO20_01" -SignalType "AO" -Unit "BOARD20" -UnitMap "1" -DefaultValue "0"

-Name "AO20_02" -SignalType "AO" -Unit "BOARD20" -UnitMap "2" -DefaultValue "0"

-Name "AO20_03" -SignalType "AO" -Unit "BOARD20" -UnitMap "3" -DefaultValue "0"

-Name "AO20_04" -SignalType "AO" -Unit "BOARD20" -UnitMap "4" -DefaultValue "0"

#Modbus 总线：

-Name "MB_DO_0" -SignalType "DO" -Unit "BOARD30" -UnitMap “49" -DefaultValue "0" –DataLen 1

-Name "MB_AO_0" -SignalType "AO" -Unit "BOARD30" -UnitMap "80" -DefaultValue "0" –DataLen 32

-Name "MB_DI_0" -SignalType "DI" -Unit "BOARD30" -UnitMap "160" -DefaultValue "0" –DataLen 1

-Name "MB_AI_1" -SignalType "AI" -Unit "BOARD30" -UnitMap "196" -DefaultValue "0" –DataLen 32

#DeviceNet 总线：

-Name "D_DO_0"-SignalType "DO"-Unit"BOARD30" -UnitMap "0" -DefaultValue "0" –DataLen 1

-Name "D_AO_1" -SignalType"AO"-Unit "BOARD30"-UnitMap "16" -DefaultValue "0" –DataLen 32

-Name"D_DI_0"-SignalType "DI"-Unit "BOARD30" -UnitMap "0"-DefaultValue"0" –DataLen 1

-Name "D_AI_1" -SignalType"AI"-Unit "BOARD30"-UnitMap "16" -DefaultValue "0" –DataLen 32

9.3.4 总线配置流程

1. Ethernet/IP Scanner

机器人控制器作为 Scanner，与 Adapter 设备进行通讯。

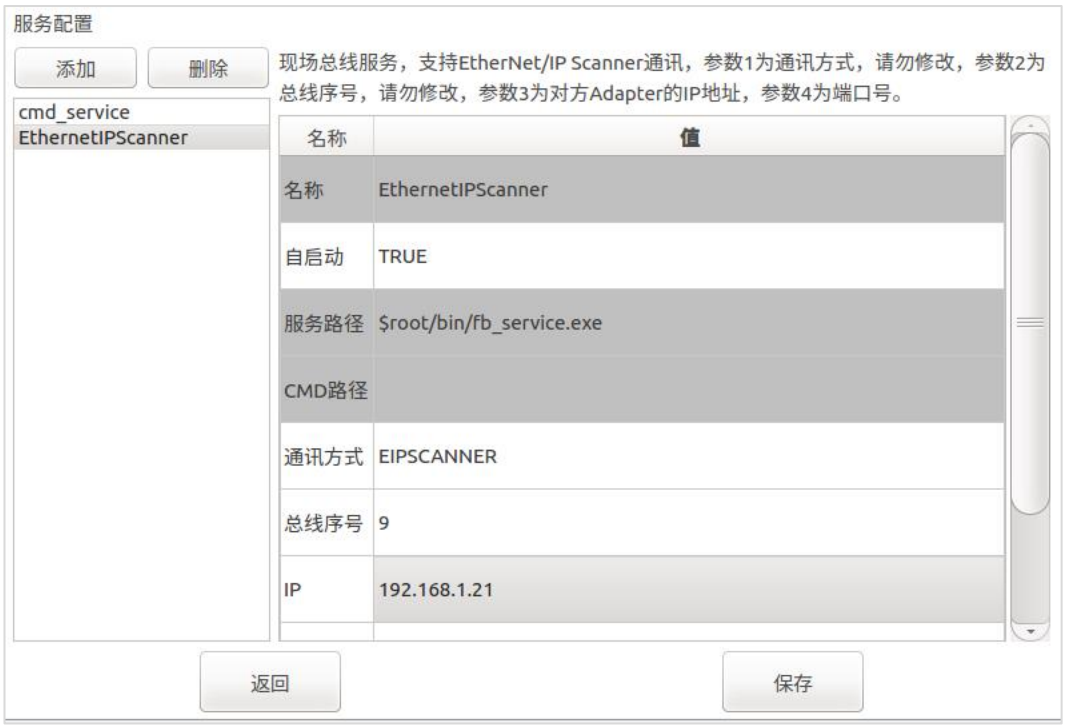
以下以 KUNBUS 品牌的 Adapter IO 设备为例：

1.接线与网络配置

使用带屏蔽的网线（以减少通讯过程中的电磁干扰），通过控制器第三或第四个网口与 Adapter IO 设备连接。

打开示教器主菜单界面，点击“配置”-“网络配置”，配置控制器 Scanner 网络，如下图所示：

打开示教器主菜单界面，点击“配置”-“服务配置”-“添加”，添加 EthernetIPScanner 服务。根据 Adapter IO 设备的硬件配置，设置 Adapter IP 地址（控制器 IP 地址与 Adapter IP 地址需在同一网段且不能重复），保持默认端口号 “2222”不变，设置自启动，如下图所示。



2.Ethernet IP 工业以太网通讯参数配置

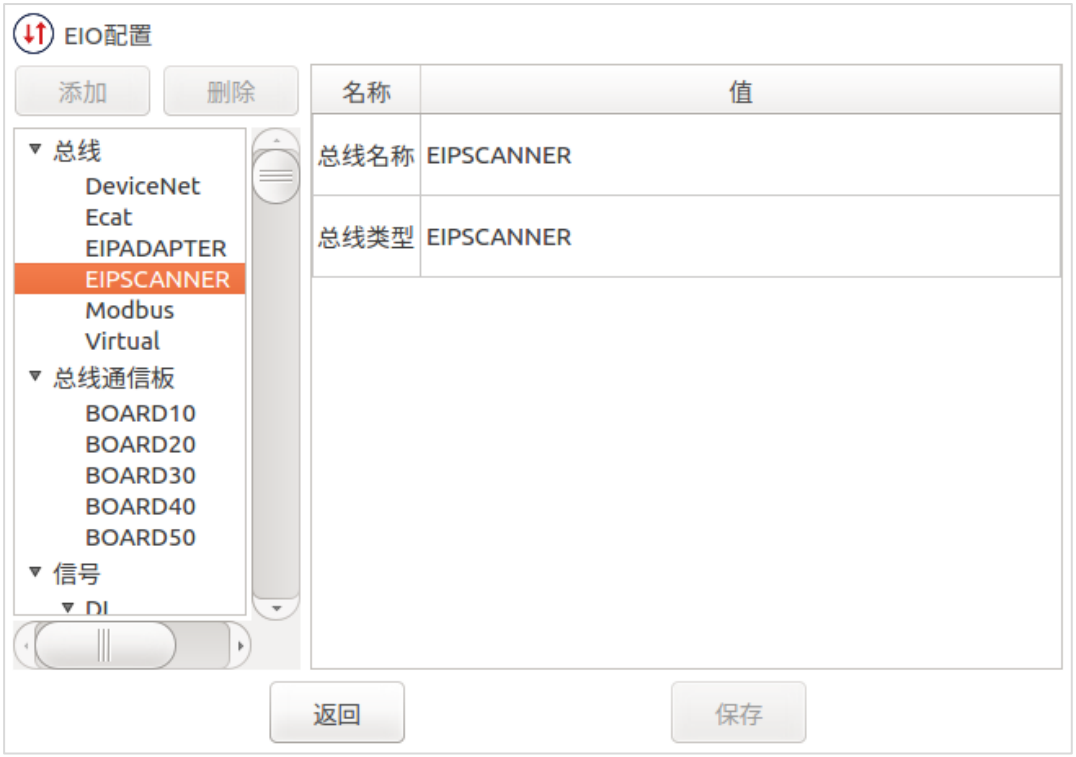
根据 Adapter 设备的 eds 文件，配置 Ethernet IP 通讯特有的参数。目前，控制器 Scanner 无法直接读取 eds 文件的配置信息，需要手动修改 FieldBus.cfg 文件中的参数值，如下图所示。

由于 Ethernet IP 参数配置涉及专业知识，修改相关配置时请联系技术支持。

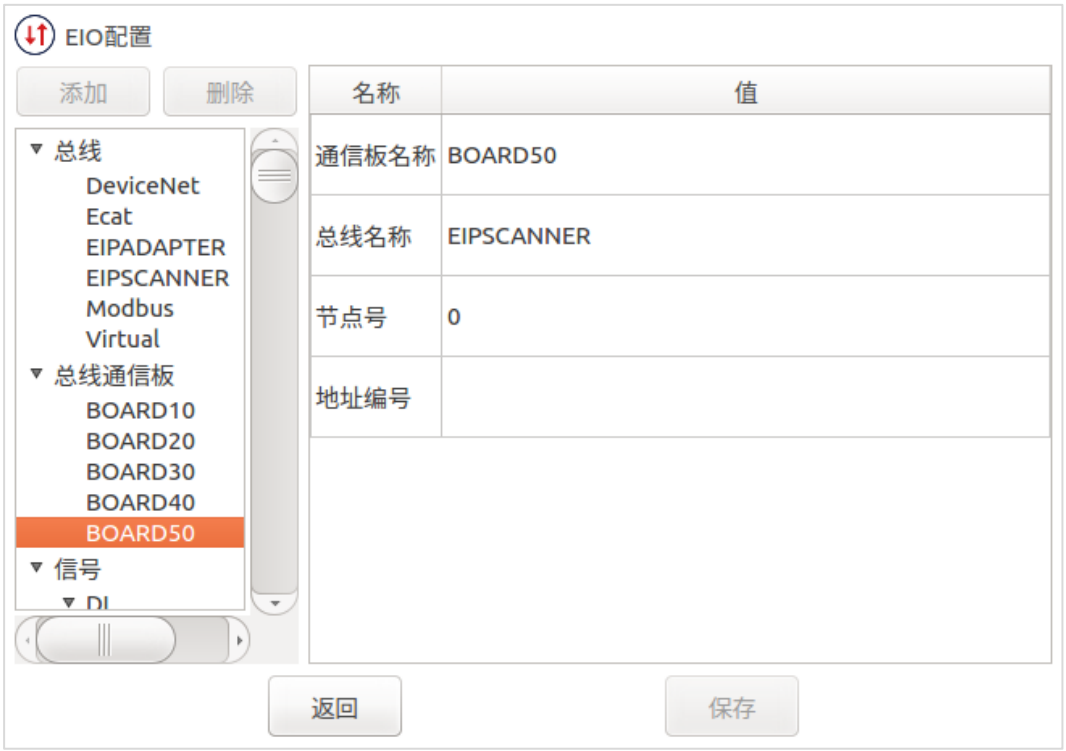
```
"EthernetIP":
[
    {
        "Scanner":
        [
            {
                "O_T_InstanceID":150,
                "O_T_Length":480,
                "O_T_RealTimeFormat":3,
                "O_T_OwnerRedundant":0,
                "O_T_Priority":2,
                "O_T_VariableLength":0,
                "O_T_ConnectionType":2,
                "RequestedPacketRate_O_T":10,
                "T_O_InstanceID":100,
                "T_O_Length":480,
                "T_O_RealTimeFormat":0,
                "T_O_OwnerRedundant":0,
                "T_O_Priority":2,
                "T_O_VariableLength":0,
                "T_O_ConnectionType":2,
                "RequestedPacketRate_T_O":10
            }
        ],
        "Adapter":
        [
            {
                "APP_Scan_Board_Time":15,
                "T_O_Length":480
            }
        ]
    }
],
```

3.EIO 配置

打开示教器主菜单界面，点击“输入输出”-“EIO 配置”-“添加”，添加 ETHERNETIP 总线，如下图所示。



添加“总线通讯板”，如下图所示。



根据 Adapter 设备的应用映射定义，添加 DI、DO、AI、AO 等“信号”。

4.所有配置保存后、重启控制器

注意！

控制器作为 Scanner 需要安装.Net 4.0 库，安装.Net 库前需要关闭 win7 的 EWF 保护

2. Ethernet/IP Adapter

机器人控制器作为 Adapter，与 Scanner 设备进行通讯。

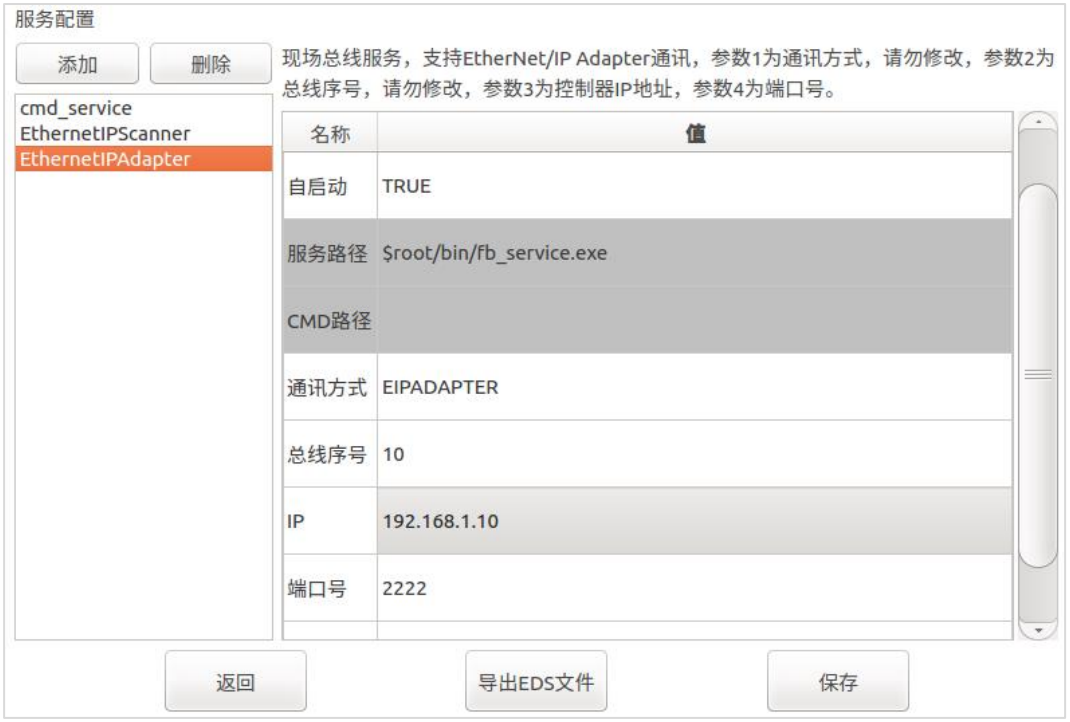
以下以基恩士品牌的 PLC 设备为例：

1.接线与网络配置

使用带屏蔽的网线（以减少通讯过程中的电磁干扰），通过控制器第三或第四个网口与 PLC Ethernet IP 网口连接。

打开示教器主菜单界面，点击“配置”-“网络配置”，配置控制器网络，如下图所示：

打开示教器主菜单界面，点击“配置”-“服务配置”-“添加”，添加 EthernetIPAdapter 服务。设置控制器 Adapter IP 地址（控制器 IP 地址与 Scanner IP 地址需在同一网段且不能重复），设置自启动，如下图所示。



在示教器上插入 U 盘，点击“导出 EDS 文件”按钮，可以导出 EDS 文件供 Scanner 使用，Ethernet IP 通讯相关的参数配置见 EDS 文件。

Adapter 默认传输最大支持 480 字节。

2.EIO 配置

打开示教器主菜单界面，点击“输入输出”-“EIO 配置”-“添加”，添加 ETHERNETIP 总线，如下图所示。

添加“总线通讯板”，如下图所示。

根据应用映射定义，添加 DI、DO、AI、AO 等“信号”。

3.所有配置保存后、重启控制器

注意！

控制器不能同时作为 Scanner 和 Adapter。

3. TCPMODBUS Client/Server

机器人控制器通讯中，可以选用 Modbus TCP 通讯中 Client 或 Server 与其他 Modbus 设备进行通讯。

以下以基恩士品牌的 PLC 设备为例：

1.接线与网络配置

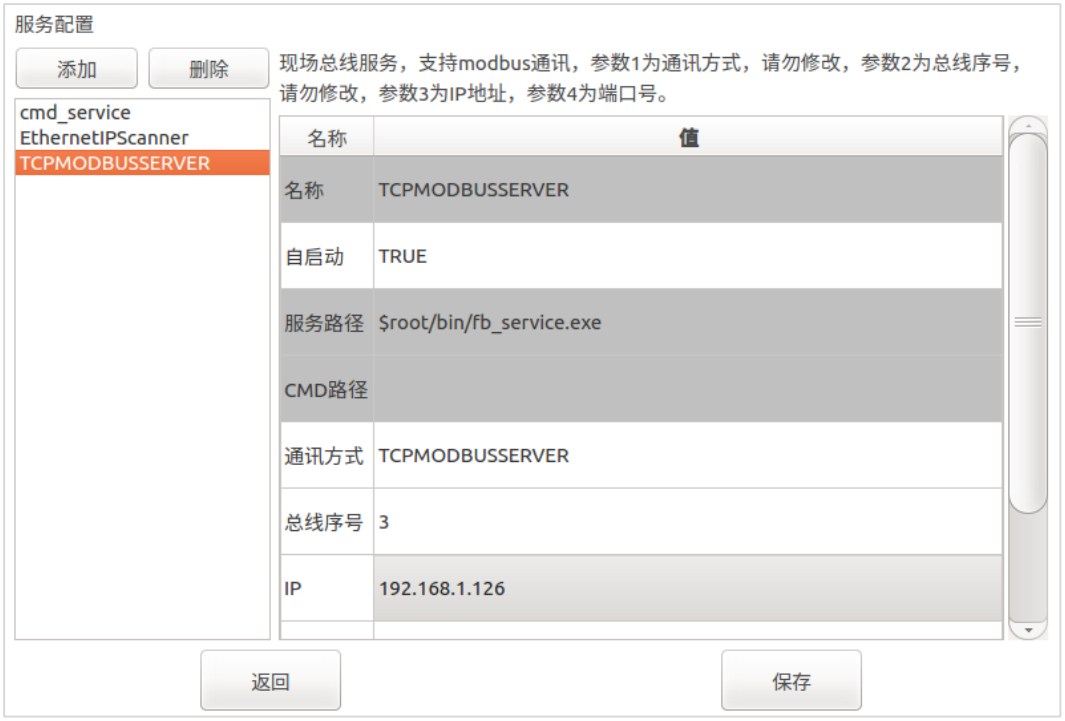
使用带屏蔽的网线（以减少通讯过程中的电磁干扰），通过控制器第三或第四个网口与 Server IO 设备连接。

打开示教器主菜单界面，点击“配置”-“网络配置”，配置控制器通讯接口网络，如下图所示：

打开示教器主菜单界面，点击“配置”-“服务配置”-“添加”，添加 TCPMODBUSCLIE 服务或添加 TCPMODBUSSERVER 服务。

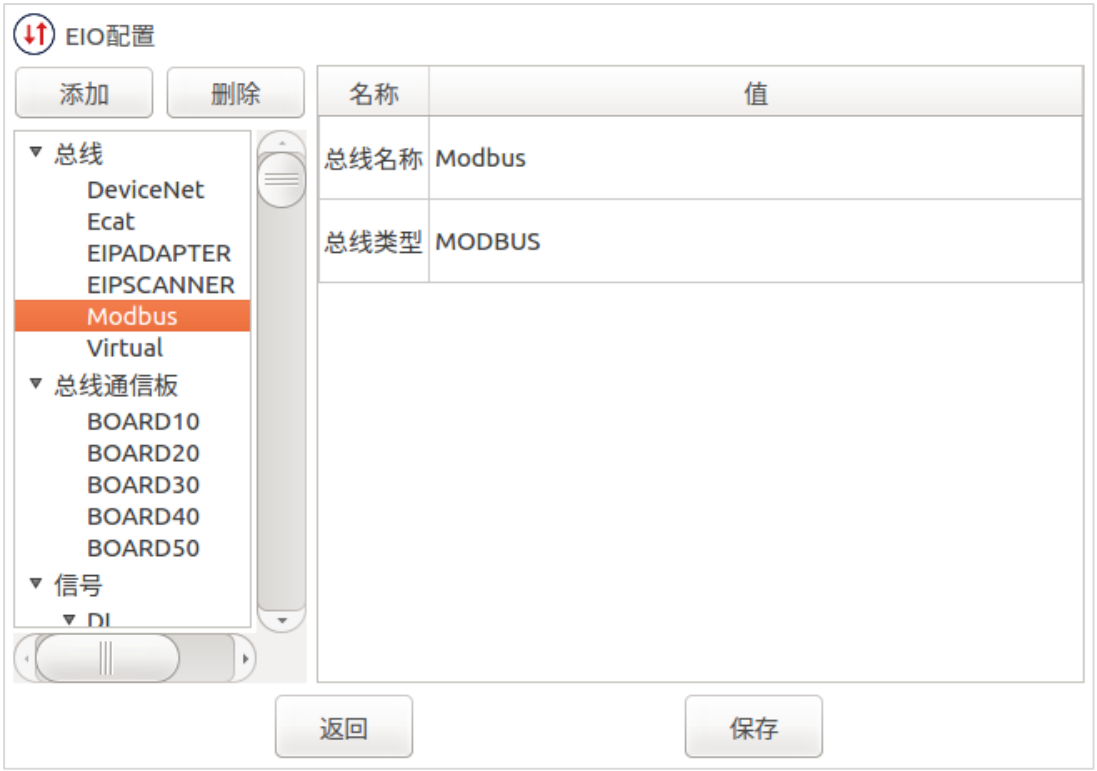
如果选择 TCPMODBUSCLIE 服务，IP 地址需要设置为 Server IO 设备的 IP 地址；如果选择 TCPMODBUSSERVER 服务，IP 地址需要设置为本地通讯网口的 IP 地址。

此外控制器 IP 地址与 Server IP 地址需在同一网段且不能重复，保持默认端口号”502”不变，设置自启动；如下图所示：



2.EIO 配置

打开示教器主菜单界面，点击“输入输出”-“EIO 配置”-“添加”，添加“总线类型” MODBUS，如下图所示。



添加“总线通讯板”，如下图所示。

根据 Server 设备的应用映射定义，添加 DI、DO、AI、AO 等信号。

3.所有配置保存后、重启控制器

注意！

在使用 MODBUS 进行通讯中，控制器只能作为 Client 或 Server 其中的一状态，如需切换其通讯状态，需要在“服务配置”删除原先 TCPMODBUS 配置或将“自启动”改为“FALSE”，重新添加新的类型即可。

9.4 系统 IO

系统 IO 是跟控制器内部相关的一些信号，用户不需要配置；

系统 IO 包含两类：

- 系统输出：系统数字量输出，反映控制器内部的状态
- 系统输入：系统数字量输入，可以通过系统输入信号的变化来触发系统命令，比如程序加载、运行、停止等。

9.4.1 系统输出

名称	说明
SYS_AUTO_MODE	1 - 系统处于自动模式， 0 - 非自动模式
SYS_MANUAL_MODE	1 - 系统处于手动模式， 0 - 非手动模式
SYS_REMOTE_MODE	1 - 系统处于远程模式， 0 - 非远程模式
SYS_MOTOR_STATE	1 – 电机上电状态， 0 – 电机掉电状态
SYS_PGM_LOAD_STATE	1 – 程序已加载， 0 – 程序没有加载
SYS_PGM_RUN_STATE	1 – 程序处于运行状态， 0 – 程序处于非运行状态
SYS_PGM_EXEC_ERROR	1–自动模式下程序执行报错，0–程序处于非报错
SYS_EM_STOP_STATE	1 – 程序处于停止状态， 0 – 程序处于非停止状态
SYS_SERVO_BUSY	1 – 电机处于运动状态， 0 – 电机处于静止状态

系统输出实时反映系统当前状态， 为了方便远程设备了解机器人控制器状态信息，所有以上信息可以关联到外部 IO 的输出，这样，该外部 IO 就可以和系统 IO 进行实时同步。

示例

-Name "DO_MOTOR_ON" -SignalType "DO" -Unit "BOARD10" -UnitMap "0" -DefaultValue "0" -Connect "SYS_MOTOR_STATE"

以上配置把外部输出信号 “DO_MOTOR_ON”和系统 IO “SYS_MOTOR_STATE”关联，当电机上电时，SYS_MOTOR_STATE 就会被置为 1； 当电机掉电时， 该信号就会被置为 0。

9.4.2 系统输入

名称	说明
SYS_MOTOR_ON	从 0 到 1 的上升沿触发电机上电， 该信号为 0 时不影响系统行为
SYS_MOTOR_OFF	从 0 到 1 的上升沿触发电机掉电， 该信号为 0 时不影响系统行为
SYS_LOAD_PGM	从 0 到 1 的上升沿触发控制器加载程序， 该信号为 0 时不影响系统行为；程序必须是位于 Program 目录下，并且名字为 remoter_entry.lua, 其它名字无效
SYS_START_PGM	从 0 到 1 的上升沿触发启动程序， 该信号为 0 时不影响系统行为
SYS_STOP_PGM	从 0 到 1 的上升沿停止程序， 该信号为 0 时不影响系统行为
SYS_EM_STOP	从 0 到 1 的上升沿触发机器人急停， 该信号为 0 时不影响系统行为
SYS_PGM_PP2MAIN	从 0 到 1 的上升沿触发程序指针回到主程序入口， 该信号为 0 时不影响系统行为
SYS_STOP_AFTER_CYCLE	暂时无效
SYS_SWITCH2_MANUAL	从 0 到 1 的上升沿触发控制器切换到手动模式， 该信号为 0 时不影响系统行为
SYS_SWITCH2_AUTO	从 0 到 1 的上升沿触发控制器切换到自动模式， 该信号为 0 时不影响系统行为
SYS_CLEAR_ERROR	从 0 到 1 的上升沿触发远程清除伺服错误，该信号为 0 时不影响系统行为

所有的系统信号都是上升沿有效，控制器检测到系统信号的上升沿后会调用相应的系统服务程序；为了方便远程设备控制机器人，可以通过信号关联把一个外部的输入信号关联到某个系统输入信号，这样通过外部设备比如 PLC 就可以远程控制机器人。外部输入信号触发系统输入的触发机制可以有多种，参见 TrigType 中的描述。

示例

-Name "DI_MOTOR_ON" -SignalType "DI" -Unit "BOARD10" -UnitMap "0" -DefaultValue "0" -Connect "SYS_MOTOR_ON"

以上配置把外部输入信号 “DI_MOTOR_ON”和系统输入 “SYS_MOTOR_ON”关联，当 DI_MOTOR_ON 被置为 1 时， 电机就会上电；该信号仅上升沿有效，当 DI_MOTOR_ON 被置为 0 时，电机不会掉电，必须通过其它信号来控制电机掉电。

9.5 信号关联和交叉

9.5.1 信号关联

信号关联包含两类关联：

- 系统输出关联到通用输出： 把系统的状态信息输出到通用输出，让外部设备了解机器人系统状态， 通过关联外部设备可以知道把机器人的运行模式、电机状态、程序状态
- 通用输入关联到系统输入：外部通用输入触发系统输入，从而可以执行特定的系统命令，如切换手动/自动模式， 运行/停止程序

示例

把手动状态信息关联到通用输出 DO_1 的配置如下：

-Name "DO_1" -SignalType "DO" -Unit "BOARD10" -UnitMap "16" -DefaultValue "0" -DataLen "1" -Connect "SYS_MANUAL_MODE"

通过外部输入信号 DI_StopPGM 来停止程序的配置如下：

-Name "DI_StopPGM" -SignalType "DI" -Unit "BOARD20" -UnitMap "64" -DefaultValue "0" -Connect "SYS_STOP_PGM"

信号关联在配置文件中的关键字如下：

名称	说明
Connect	指定需要关联的系统信号名字
TriggType	触发条件

信号关联的触发条件包含如下 2 种类型：

名称	说明
RISE	上升沿
FALL	下降沿

9.5.2 信号交叉

通用的外部输入信号交叉影响一个通用的外部输出，比如机器人检测到焊枪发生碰撞后，点亮一个报警灯。

示例

-Name "DI_GunCollision" -SignalType "DI" -Unit "BOARD20" -UnitMap "14" -DefaultValue "0" -Cross "DO_WarningLight" -CrossType "LEVEL"

信号交叉在配置文件中的关键字如下：

名称	说明
Cross	指定需要交叉的通用外部输出信号名字
CrossType	触发条件如下表，默认值“LEVEL”

信号交叉触发条件包含如下 3 种类型：

名称	说明
RISE	上升沿
FALL	下降沿
LEVEL	电平变化

10 编程示例

10.1 运动程序示例

关键全局变量定义

JOINTTARGET("JHome",{-0,19.592,25.504,-0,49.445,-0.001,-0},{-0,-0,-0,-0,-0,-0,-0})

LOADDATA("Load_6kg",0,{74.3,0,0},{1,0,0,0},0.0418,0.0418,0.0102,0,0,0)

ROBTARGET("P10",{1112.170044,-424.820007,544.450012},{0.039628,0.000017,-0.999214,-0.000001},{-1,-1,-1,0},{-0,-0,-0,-0,-0,-0,-0},-0)

ROBTARGET("P20",{921.460022,811.289978,544.450012},{0.039626,0.000016,-0.999214,0.000001},{0,0,0,0},{-0,-0,-0,-0,-0,-0,-0},-0)

TOOLDATA("tool1",true,{{-0,-0,-0},{1,-0,-0,-0}},{-0,{-0,-0,-0},{1,-0,-0,-0},-0,-0,-0,-0,-0,-0})

WOBJDATA("wobj1",false,true,"",{{-0,-0,-0},{1,-0,-0,-0}},{{-0,-0,-0},{1,-0,-0,-0}})

关节绝对运动至 JHome 点位

MoveAbsJ(JHome,v200,fine,tool1,wobj1,Load_6kg)

线性运动以 1000mm/s 速度滑过 P10 点位不停留

MoveL(P10,v1000,z0,tool1,wobj1)

关节运动以 1000mm/s 速度、10mm 转弯半径滑过 P20 点位

MoveJ(P20,v1000,z10,tool1,wobj1)

线性运动以 1000mm/s 速度、5mm 转弯半径滑过 P10 点位

MoveL(P10,v1000,z5,tool1,wobj1)

关节绝对运动至 JHome 点位

MoveAbsJ(JHome,v200,fine,tool1,wobj1,Load_6kg)

10.2 弧焊程序示例

关键全局变量定义

```
ROBTARGET("P00",{1036.910034,-43.840000,834.940002},{0.318425,-0.109922,0.940730,0.039360},{-1,0,-1,0},{0,0,0,0,0,0,0},0)

ROBTARGET("P10",{1036.910034,-55.790001,834.659973},{0.318344,-0.109969,0.940751,0.039383},{-1,0,-1,0},{0,0,0,0,0,0,0},0)

ROBTARGET("P20",{1086.910034,-89.970001,834.599976},{0.318293,-0.110007,0.940763,0.039389},{-1,0,-1,0},{0,0,0,0,0,0,0},0)

ROBTARGET("P30",{1056.910034,-130,834.599976},{0.318295,-0.109993,0.940764,0.039386},{-1,0,-1,0},{0,0,0,0,0,0,0},0)

ROBTARGET("P40",{1056.910034,-130,874.599976},{0.318295,-0.109993,0.940764,0.039386},{-1,0,-1,0},{0,0,0,0,0,0,0},0)

SEAMDATA("seam1",1000,2000,{0,0,18,10,85},200,0,10,100,2,{0,0,18,10,85},2000,100,{0,0,18,10,80},400,10,{0,0,18,10,80},100)

WELDDATA("weld1",10,{0,0,18,10,80})

WEAVEDATA("weave1",0,0,5,4,2,1,1,1)

local Tarr1 = nil
```

线性运动以 1000mm/s 速度至 P00 点位

```
MoveL( P00, v1000, fine, tool1, wobj0, load0)
```

线性运动至 P10 点位。在到达 P10 途中，为开始焊接做准备，比如预送气等

```
ArcLStart(P10, v1000, seam1, weld1, weave1, fine, tool1, wobj0, Tarr1, load0)
```

从当前点位出发，途径 P20 到 P30 点位做圆弧形焊接，seam1 定义了焊缝数据，weld1 描述了焊接数据

```
ArcCEnd(P20, P30, v100, seam1, weld1, weave1, fine, tool1, wobj0, Tarr1, load0)
```

结束焊接，并线性运动以 1000mm/s 速度至 P40 点位

```
MoveL(P40, v1000, fine, tool1, wobj0, load0)
```

11 术语表

术语	描述
世界坐标系	世界坐标系是系统中所有坐标系的参照基准，其他坐标系都是直接或者间接参考这个坐标系通过平移和旋转而得。
基坐标系	基坐标系位于机器人底座，它描述了机器人的安装位置信息，即机器人底座在世界坐标系中的位置。世界坐标系理论上和基坐标系可以不同，但是在单个机器人的控制系统中，通常这两个坐标系是一致的。
工件坐标系	工件坐标系用于描述待加工工件的安装位置和姿态。如果工件安装在固定工作台或者地面上，则工件坐标系描述了工件相对于世界坐标系的位置信息；如果工件是安装在机器人末端法兰盘上，则工件坐标系描述了工件相对于机器人末端法兰盘的位置信息。
工具坐标系	工具坐标系用于描述工具中心点 (TCP) 的安装位置和姿态。一般情况下，将工具中心点定义在工具的加工点处，如弧焊焊枪的尖端、喷胶胶枪的枪口或者机械手爪的中心等。如果工具是安装在机器人末端法兰盘上，则工具坐标系描述了工具相对于机器人末端法兰盘的位置信息；如果工具安装在固定工作台或者地面上，则工具坐标系描述了工具相对于世界坐标系的位置信息。
机器人负载	机器人负载，通常指机器人在移动过程中使用的总负载，即末端法兰盘上安装的工具负载加上该工具正在处理的工件的有效负载，包含质量，重心和惯性张量信息。
关节运动	关节运动指的是机器人从当前点通过转动关节位置来最终到达目标点，在运动过程中机器人末端或者工具中心点不会像线性运动一样始终处于一条直线上。
线性运动	线性运动是指机器人从当前点以直线方式运动到目标点，机器人末端或者工具中心点在运动过程中始终处于一条直线上。
圆弧运动	圆弧运动指的是机器人在运动过程中，机器人末端或者工具中心点处于一条圆弧上。
IO系统	IO系统是智殷机器人控制器和外部设备(如PLC)进行通信和数据交换的接口。数据的扫描和刷新是实时的，每隔 4ms，整个IO系统会进行一次刷新。

END

G-UM-SW00001-A7 版本号：A

上海智殷自动化科技有限公司
中国，上海
电话：+86 021- 6485 5368
传真：+86 021- 6485 5368