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-A5

版本：A

版次：5

本手册是智殷机器人编程及指令系统的用户说明书。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变动或者重大变动，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

- 机器人基础概念
- 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
- 常用函数
- I/O 系统配置
- 编程示例
- 术语表和索引

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	第五版

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。智殷机器人提供了如下几种坐标系：

- 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。

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）描述坐标系原点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。

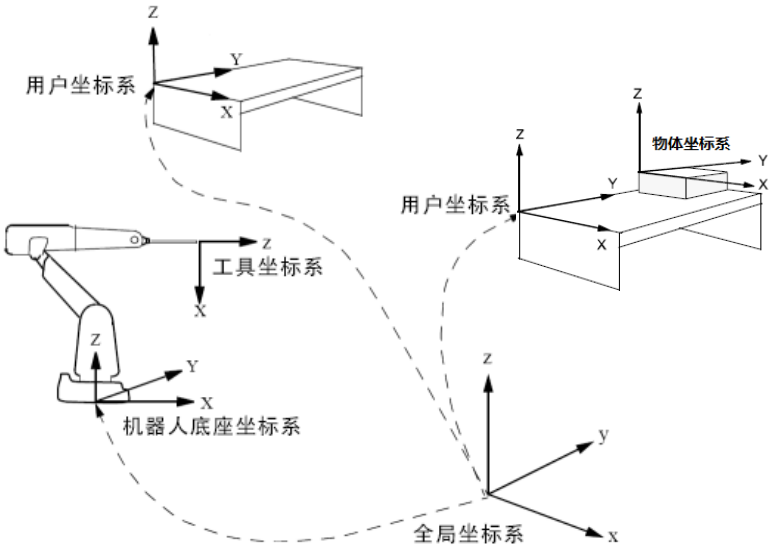


图 1 坐标系示意图

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，如图 2 所示。

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，右手定则满足以下两条：

1 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；

2 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。

图 2 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。

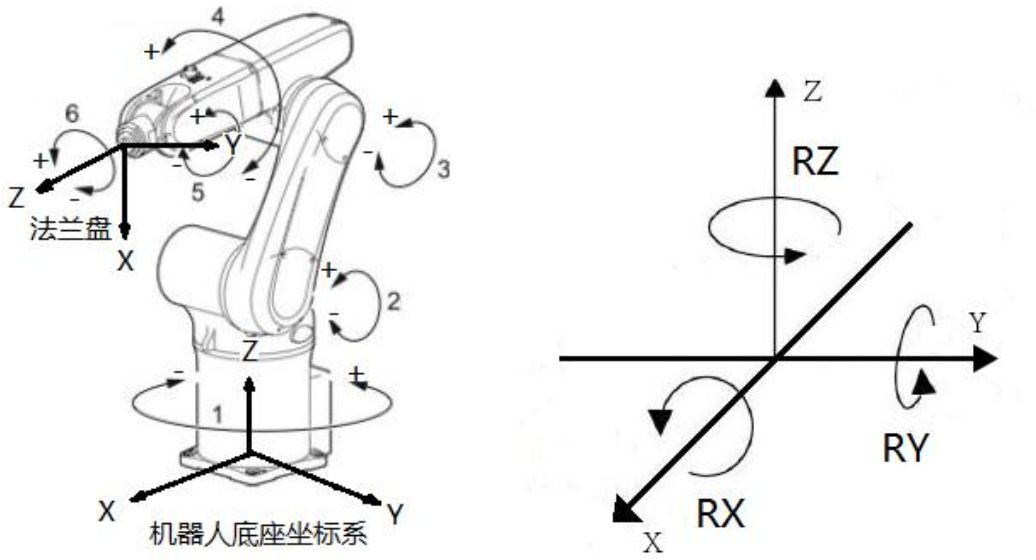


图 2 标准 6 轴机器人的关节和坐标系

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）。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，如图 3 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。

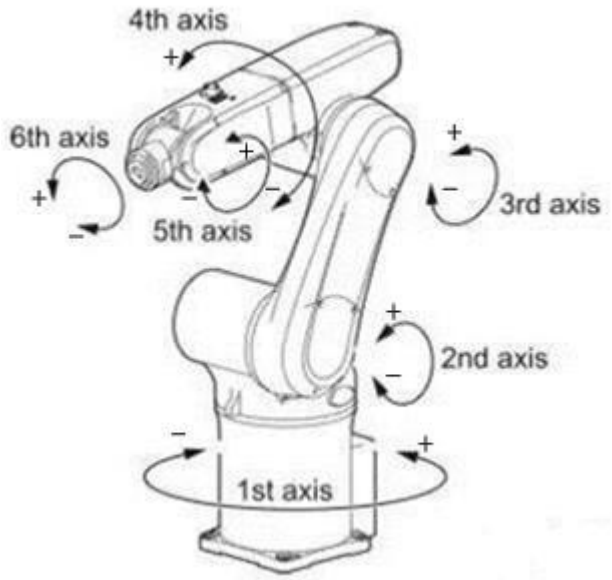


图 3 关节方向和奇异点位置

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。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；当选择经过点时，机器人会以一定的速度经过以目标点为球心，以指定转弯路径为半径的一个球形区域，不会停止在目标点，除非该点位路径上的最后一个点。

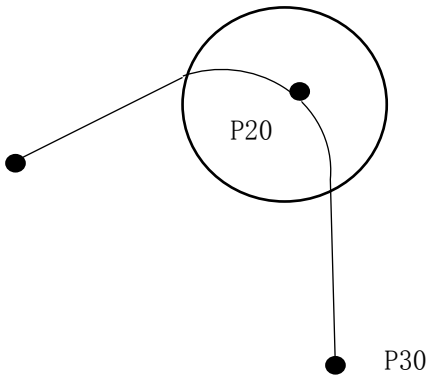


图 4 转弯路径示意图

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预处理 (require) 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, num 等，它们分别用 JOINTTARGET( )、ROBTARGET( )、LOADDATA( )、TOOLDATA( )、WOBJDATA( )、SPEEDDATA( )、ZONEDATA( )、NUMDATA( )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("fine",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

例如：

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 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。例如：

```
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…elseif…else…end。

例如：

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。

程序如下：

```
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。

例如：

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)

程序如下：

```
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。

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。

程序如下：

```
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。

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。

程序如下：

```
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。

例如：

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

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。

函数结构

OpenFile (fileName [, mode])

函数参数

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": 读模式 (默认)。 "w": 写模式。 "a": 添加模式。 "r+": 更新模式，所有之前的数据将被保存。 "w+": 更新模式，所有之前的数据将被清除。 "a+": 添加更新模式，所有之前的数据将被保存，只允许在文件尾进行添加。 "b": 某些系统支持二进制方式。

调用示例

```
local fd, err = OpenFile (fileName, 'a+')
```

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

ReadFile (handle[, format])

函数参数

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。
number	读取 number 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

```
ReadFileLines (handle)
```

函数参数

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

SeekFile (handle[, whence, offset])

函数参数

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

WriteFile (handle, param1[, param2, …])

函数参数

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 ([filename])
```

函数参数

filename: 要关闭文件的文件名

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}

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}；

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，单位是 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}。

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。

使用示例

```
BVoltage = {}
BVoltage = TriggCheckIO(4, 0, 0, AIBWeldingVoltage, "GT", AIMaxVoltage, 1, trap_int_er
rprocess)
```

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, num 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( ) 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，用于定义机器人的轴和外轴所在的关节空间位置，单位是角度。其数据结构定义为：

```
{
  {rax1, rax2, rax3, rax4, rax5, rax6, rax7},
  {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});
```

3.2.2 ROBTARGET 机器人位置数据

数据结构

机器人位置数据，表示机器人的运动目标点，其数据结构定义为：

```
{
pos trans,
orient rot,
config cfg,
{eax1, eax2, eax3, eax4, eax5, eax6, eax7},
arm_angle
}
```

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}, 单位是 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。

cfg 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)。

arm_angle 臂型角

机器人的臂型角。浮点型数据，单位是角度。

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)

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,

pose pose,

centralinertia central_inertia

}

负载数据参数

负载数据包含三个参数：

mass 负载质量

数据类型: float

重量单位为 kg。

pose 质心坐标系

数据类型: pose

centralinertia 惯量坐标

数据类型：float

数据结构：{Ixx, Iyy, Izz, 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)

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 tool_frame,
  loaddata load
}
```

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: 拿着 tool。

false: 机器人不拿着工具，也就是一个固定的工具。

tool_frame

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{u0,u1,u2,u3}。

pose 数据结构： {{x,y,z}，{u0,u1,u2,u3}}

load

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（u0,u1,u2,u3）。

惯性主轴的惯量（x,y,z）,单位 kgm²。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_name,
   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
v2500	2500	500	5000	1000
v3000	3000	500	5000	1000
v4000	4000	500	5000	1000
v5000	5000	500	5000	1000
v6000	6000	500	5000	1000
v7000	7000	500	5000	1000
vmax	*)	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 tcp_m;
   float ori_tcp_m;
   float ext_tcp_m;
   float ori_rad;
   float ext_m;
   float ext_rad;
}
```

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。

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。本章将按照字母序，对智殷机器人编程语言的常用指令进行详细说明。

4.1 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。

4.2 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。

4.3 GetAI 获取 AI 信号值

指令功能

GetAI 用于获取模拟输入信号的值。输入范围跟 AD 位数有关。

指令结构

GetAI (name)

指令参数

name	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。

4.4 GetAO 获取 AO 信号值

指令功能

GetAO 用于获取模拟输出信号的值。输出范围跟 DA 位数有关。

指令结构

GetAO (name)

指令参数

name	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。

4.5 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

GetDI (name)

指令参数

name	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。

4.6 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

GetDO (name)

指令参数

name	数字输出信号名称

指令示例

level = GetDO("DO10_6")

获得 DO10_6 电平值，保存在 level。

4.7 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

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”。
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("J10",{-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(0,J10,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("J20",{-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(0,J20,v500, z50,tool1,wobj1,Load_6kg)

4.8 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, CirPoint, ToPoint, Speed, Zone, Tool, WObj)

指令参数

Support arrary format or structure format

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”。
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(0,P10,v200,fine,tool1,wobj1)
MoveC(0,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(0,P10,v200,fine,tool1,wobj1)
MoveC(0,P20,P30,v200,z10,tool1,wobj1)
MoveC(0,P40,P10,v200,z10,tool1,wobj1)
```

4.9 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

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”。

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)
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(0,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)
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(0,P20,v500,z10,tool2,wobj1)
```

4.10 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

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”。
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(0,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)
```

4.11 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

Offs(point, XOffset, YOffset, ZOffset)

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(P20, 0, 0, 10), v500, z10, tool2, wobj1)

4.12 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

SearchL([MecName], DI, Flag, SearchPoint, ToPoint, Speed, Tool, WObj, [Load])

指令参数：

参数	数据类型	说明
MecName	string	机械单元名，缺省值为 0
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
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

指令示例

SearchL(“DI10_4”, 0, P10, P20, v50, tool0, wobj0, loaddata0)

4.13 RegisterTrap 注册中断

指令功能

注册中断。

指令结构

RegisterTrap(trap_num, trap_routine)

指令参数

trap_num	中断号
trap_routine	中断服务程序

指令示例

```
trap_errprocess = 1
RegisterTrap (trap_errprocess, errprocess)
```

4.14 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, acc, dacc)

指令参数

max_acc_defined	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	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。
acc	表示 TCP 运动加速度占正常值的比率，最大值为 1.0。
dacc	表示 TCP 运动加速度的增速占正常值的比率，最大值为 1.0。

指令示例

```
JOINTTARGET("J10",{-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(false, 1, 1, 1)
MoveAbsJ(0, J10,v500,z50,tool1,wobj1,Load_6kg)
```

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。输出范围跟 DA 位数有关。

指令结构

SetAO (name, value)

指令参数

name	模拟输出信号的名称
value	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

name	数字输出信号名称
level	电平值

指令示例

SetDO("DO10_6",1)

设置 DO10_6 为高电平。

4.17 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

time_ms 休眠或者挂起的时间，单位：ms。

指令示例

Sleep(500) 休眠 500ms

4.18 SocketConnect 连接远程计算机

指令功能

SocketConnect 用于将套接字与客户端应用中的远程计算机相连。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])

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”
local host = “127.0.0.1”
local port = 6000
local timeout = 1000
--建立连接
local err = SocketConnect (socketName, host, port, timeout)
--连接失败，处理
if -1 == err then
    print(“connection failed !”)
    return false
```

end

4.19 SocketDisconnect 断开与远程计算机的连接

指令功能

SocketDisconnect 用于断开与远程计算机的连接。

指令结构

SocketDisconnect(socketName)

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

SocketDisconnect(socketName)

4.20 SocketReceive 接收来自远程计算机的数据

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])

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
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
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

指令示例

ret, recvMsg, recvLength = SocketReceive(socketName,1)

4.21 SocketSend 向远程计算机发送数据

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])

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

(1)发送字符串

local sendMsg = "hello"

ret = SocketSend(socketName, sendMsg)

(2)发送字节数组

local sendMsg = {20, 154, 1, 13, 90, 84}

ret = SocketSend(socketName, sendMsg)

4.22 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 ( )

4.23 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

TriggC(P10,P20,v200,event11,fine,tool0,wobj0,loaddata0)

4.24 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

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

指令参数

参数	数据类型	
Distance	Num	
Start	switch	关于参数 Distance 的距离始于移动起点而非终点时使用
Time	switch	单位：s
Signal	signalxx	将测试的信号的名称
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	switch	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

返回值

TriggData

类型

triggdata

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

指令示例

```
event11[3]=TriggCheckIO(6,0,0,"DORedLight","LT",1,0,1)
BVoltage = {}
BVoltage=TriggCheckIO(50,0, 0, AIBWeldingVoltage, "GT", AIMaxVoltage, 1,
trap_int_errprocess)
```

4.25 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

TriggIO (Distance,Start,Time,Dop/AOp,setValue,DODelay)

指令参数

参数	数据类型	说明
Distance	num	距移动路径终点的距离
Start	Switch	Distance 始于起点
Time	Switch	Distance 的指定值实际为时间而不是距离（单位：s）
DOp	signaldo	DO 信号名称
AOp	signalao	AO 信号名称
SetValue	num	信号的期望值
DODelay	num	输出信号延时

返回值

TriggData

类型

triggdata

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

指令示例

Event1 = TriggIO(5, 0, 0, "DORedLight", 1, 0.3)

Event2 = TriggIO(20, 0, 0, "DOGreenLigh", 1, 0.3)

4.26 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

TriggJ(0, P10, v200, event11, z10, tool0, wobj0, loaddata0)

4.27 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

TriggL(0, P10, v200, event11, z10, tool0, wobj0, loaddata0)

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)

指令参数

name	数字输入信号的名称
relation	条件参数 "LT"（小于）或者"GT" （大于）
level	电平值
MaxTime	可选参数，允许的最长等待时间，以秒计
ValueAtTimeout	可选参数，如果指令超时，则将当前信号值储存在该变量中。

指令示例

timeoutvalue = WaitAI("AI10_1", “LT”, 400, 0.5)

等待 AI10_1 数字值为小于 400,等待最大时间 0.5 秒，如果超时，则将当前的 AI 值存入 timeoutvalue。

4.29 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)

指令参数

name	数字输入信号的名称
level	电平值
MaxTime	可选参数，允许的最长等待时间，以秒计
TimeOutFlag	可选参数，超出时间时设置为 true，否则，其将设置为 false。

指令示例

flag = WaitDI("DI10_1", 1,0.5)

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。

4.30 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 = WaitAI(value1,value2，…, relation,value,MaxTime)

指令参数

value1,value2，…	数字输入信号的名称，格式为"DI10_1=1"，"DI10_2=0"
relation	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。
level	电平值
MaxTime	允许的最长等待时间，以秒计
TimeOutFlag	超出时间时设置为 true，否则，其将设置为 false。

指令示例

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

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。

4.31 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)

指令参数

InOutside	数据类型：number 0，定义箱内的体积。 1，定义箱外的体积（相反体积）。

LowPoint	用于定义箱子的一个较低角的位置（x、y、z），以 mm 计。
HighPoint	用于定义与先前角相对的角的位置（x、y、z），以 mm 计。
返回：block	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。

指令示例

```
local corner1 = {503.650024,-59.200000,557.720001}
local corner2= {643.650024,59.200000,357.720001}
local block = WZBoxDef( 0, corner1, corner2)
```

4.32 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)

指令参数

InOutside	数据类型：number 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	数据类型：pos 用于定义圆柱体的一个圆周端中心的位置（x、y、z），以 mm 计。
Radius	数据类型：number 圆柱体的半径，以 mm 计。返回：用于储存指定体积的变量。
Height	数据类型：number 圆柱体的高度，以 mm 计。如果其值为正（+z 方向），则 CentrePoint 参数为圆柱体下端中心。如果其值为负（-z 方向），则 CentrePoint 参数为圆柱体上端中心。
返回：cyl	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。

指令示例

```
local C1 = {200,300,400}
```

```
local r1= 500
local h1= 600
local sph = WZCylDef(0, C1, r1, h1)
```

4.33 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)

指令参数

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

指令示例

```
local C1 = {200,300,400}
local r1= 500
local sph = WZSphDef(0, C1, r1)
```

4.34 WZHomeJointDef 定义内部关节的安全区域

指令功能

用于定义关节坐标系中的安全区域，以便将机械臂和外轴作为 HOME 或 SERVICE 位置的安全区域。

指令结构

wzhjoint = WZHomeJointDef (InOutside, MiddleJointVal, DeltaJointVal)

指令参数

InOutside	数据类型：number 0，定义 MiddleJointVal+/-DeltaJointVal 内的关节空间。 1，定义 MiddleJointVal+/-DeltaJointVal 外的关节空间（相反关节空间）。
MiddleJointVal	数据类型： jointtarget 确定关节空间中心的关节坐标位置。针对各机械臂轴和外轴进行说明（旋转轴以度计，线性轴以mm计）。在绝对关节中进行说明（而非在有关外轴的偏移量坐标系EOffsSet-EOffsOn中）。有关某些轴的值9E9意味着不应该监控该轴。编程期间，无效外轴亦得出9E9。
DeltaJointVal	数据类型：jointtarget 距离关节空间中心的关节坐标中的+/-δ位置。针对用于监控的所有轴，该值必须大于 0。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。
返回：wzhjoint	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。

指令示例

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

4.35 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)
```

指令参数

InOutside	数据类型：number 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	数据类型： 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	数据类型：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。
返回：wzljoint	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。

指令示例

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

4.36 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

motgrp	数据类型：number 0，左臂 1，右臂
Wztype	数据类型： number 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	数据类型：string 安全区域的识别号（字符串）。

Shape	数据类型： 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)
```

4.37 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

motgrp	数据类型：number 0，左臂 1，右臂
Wztype	数据类型： number 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	数据类型：string 安全区域的识别号（字符串）。
Side	数据类型：number 0，停止，将设置数字信号输出信号。 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	数据类型： shapedata 用以定义安全区域体积的变量。
Signal	数据类型：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	数据类型：number 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 wz2= "wz2"
local sph = WZSphDef(0, C1, r1)
WZDOSet( 0,1, wz2,1, sph,"DO10_7",1)
```

4.38 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

motgrp	数据类型：number 0, 左臂 1, 右臂
Wzname	数据类型：string 安全区域的识别号（字符串）。

指令示例

```
WZDisable(0,”wz1”)
```

4.39 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

motgrp	数据类型：number
	0, 左臂
	1, 右臂
Wzname	数据类型：string
	安全区域的识别号（字符串）。

指令示例

WZEnable (0,”wz1”)

4.40 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

motgrp	数据类型：number
	0, 左臂
	1, 右臂
Wzname	数据类型：string
	安全区域的识别号（字符串）。

指令示例

WZFree (0,”wz1”)

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

序号	参数名	单位	默认值	说明
1	purge_time	ms	100	焊机气泵的充气时间，气体充满气管和焊枪。
2	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。

3	ign_sched（预留）	无	0	焊机起弧的程序号。
	ign_mode（预留）	无	0	焊机起弧的模式。
	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。
4	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。
5	scrape_start（预留）		0	预留参数。
6	heat_speed	mm/s	100	焊机加热过程中机器人的运动速度。
7	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。
8	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。
9	heat_sched（预留）	无	0	焊机加热的程序号。
	heat_mode（预留）	无	0	焊机加热的模式。
	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。
10	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。
11	fill_time	ms	100	焊机填弧的时间。
12	fill_sched（预留）	无	0	焊机填弧的程序号。
	fill_mode（预留）	无	0	焊机填弧的模式。
	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。
13	bback_time	ms	100	焊机收弧的回烧时间。
14	rback_time	ms	100	焊机填弧的回烧时间。
15	bback_sched（预留）	无	0	焊机收弧的程序号。
	bback_mode（预留）	无	0	焊机收弧的模式。
	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。
16	postflow_time	ms	100	焊接停止后的送气时间。

参数说明

Purge_time

单位：秒，定义了保护气管路的预充气时间，焊接开始前清理枪管中空气的时间。

Preflow_time

单位：秒，定义了保护气的预吹气时间，焊枪到达焊接位置对工件进行保护。

Ign_Arc

起弧。

- Voltage: 起弧电压。
- Wirefees:送丝速度。
- Heat_speed：焊接开始加热时的焊接速度。
- Heat_time：加热时间。
- Heat_distance: 加热的距离。

Heat_arc

- Voltage:加热时的电压。
- Wirefeed:加热时的送丝速度。
- Cool_time:焊接过程结束时，冷却时间。
- Fill_time:填充弧坑时间。

fill_arc

- Voltage: 填充弧坑电压。
- Wirefeed: 填充时送丝速度。
- Bback_time:回烧时间，防止粘丝。
- Rback_time:焊丝回卷时间。

Bback_arc

- Voltage:回烧电压。
- Wirefeed:回烧送丝速度。
- Postflow_time: 秒，在焊接结束后，保护气体延长吹气时间，防止电极和焊缝在最后冷却过程中氧化。

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	单位	默认值	说明
weld_speed	mm/s	100	焊机焊接的速度。
weld_sched（预留）	无	0	焊机焊接的程序号。
weld_mode（预留）	无	0	焊机焊接的模式。
weld_volt	0.1V(%)	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。
weld_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。
weld_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

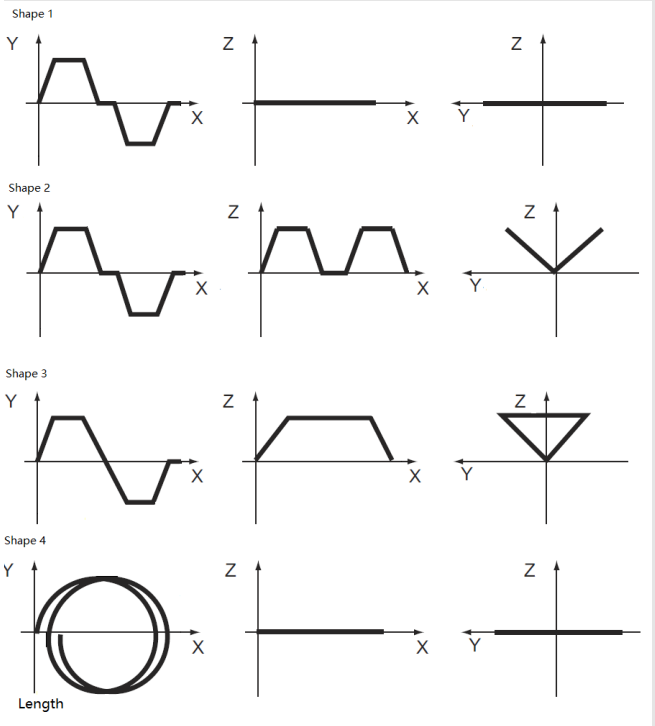
摆弧数据，用于加热和焊接过程，定义了摆弧的路径。

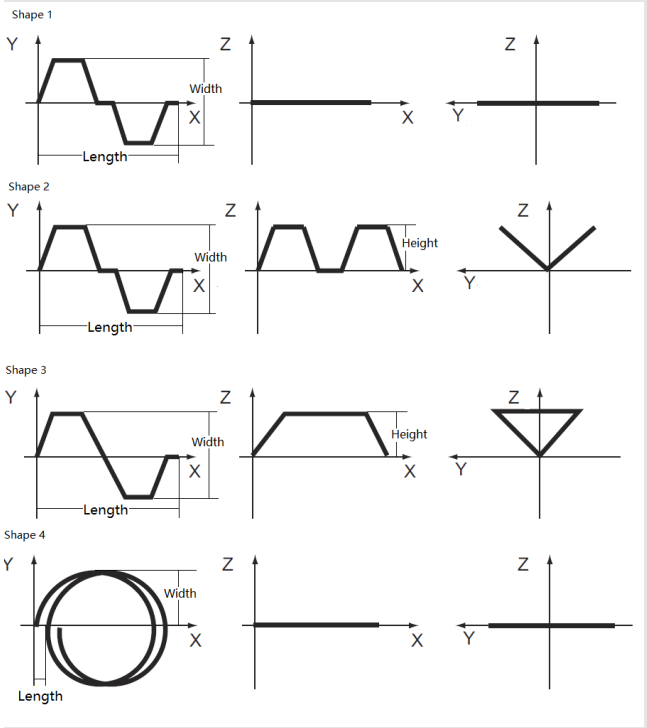
数据结构

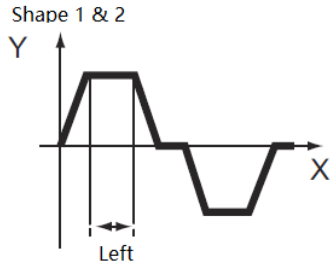
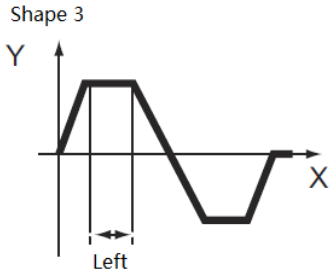
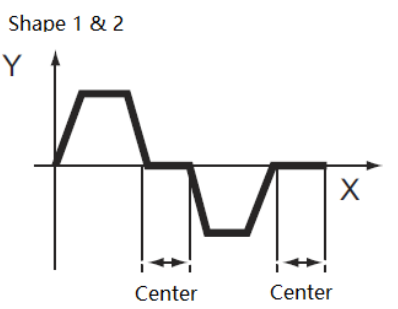
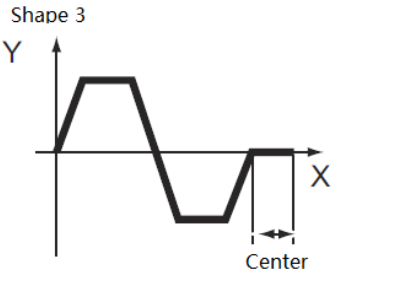
```
weavedata
{
    weave_shape;
    weave_type;
    weave_length;
    weave_width;
    weave_height;
```

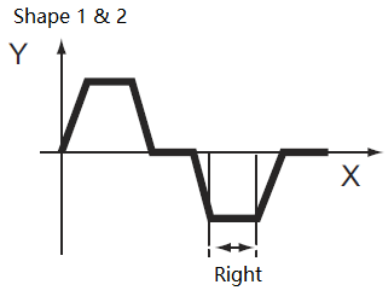
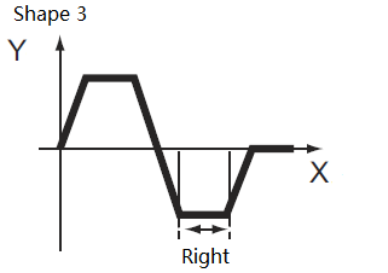
```
        dwell_left;
        dwell_center;
        dwell_right;
        weave_dir;
        weave_tilt;
        weave_ori;
        weave_bias;
}
```

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。 0 表示没有摆弧； 1 表示在XY平面做Z型摆弧； 2 表示在和焊缝垂直的YZ平面做V型摆弧； 3 表示在和焊缝垂直的YZ平面做三角型摆弧； 4 表示在XY平面做圆型摆弧。 

weave_type	无	0	0：一般的摆动，机器人的 6 个轴都参与摆动； 1：腕摆动，第 5 轴、第 6 轴参与摆动； 2：简易的臂摆动，第 1 轴、第 2 轴、第 3 轴参与摆动； 3：简易的腕摆动，第 4 轴、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 trackdata 跟踪数据

数据结构

```
trackdata
{
        track_system;
        store_path;
        max_corr;
        arctrack;       // arctrackdata,track_system = 0
        opttrack;       // opttrackdata,track_system = 1
}
arctrackdata
{
        track_type;
        gain_y;
        gain_z;
        weld_penetration;
        track_bias;
        min_weave;
        max_weave;
        min_speed;
        max_speed;
}
opttrackdata
{
        joint_no;
        filter;
        seamoffs_y;
        seamoffs_z;
        seamadapt_y;
        seamadapt_z;
        track_mode;
}
```

5.1.5 arcdata 弧焊数据

数据结构

```
arcdata
{
    sched
    mode
    voltage
    wirefeed
    current
}
```

arcdata 数据结构参数说明：

参数	单位	说明
sched		程序号（预留）
mode		模式（预留）
Voltage	0.1V(%)	电压
wirefeed	mm/s	速度
current	A	电流

5.2 焊接指令

5.2.1 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
Weld	welddata	焊接过程控制参数
Weave	weavedata	焊接过程摆弧参数
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, loaddata0)

5.2.2 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
Weld	welddata	焊接过程控制参数
Weave	weavedata	焊接过程摆弧参数
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, loaddata0)

5.2.3 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
Weld	welddata	焊接过程参数
Weave	weavedata	焊接过程摆弧参数
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 , loaddata0)

5.2.4 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
Weld	welddata	焊接过程控制参数
Weave	weavedata	焊接过程摆弧参数
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, loaddata0)

5.2.5 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）。

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
Weld	welddata	焊接过程控制参数
Weave	weavedata	焊接过程摆弧参数
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 , loaddata0)

5.2.6 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
Weld	welddata	焊接过程控制参数
Weave	weavedata	焊接过程摆弧参数
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

ArcCEnd(P60, P70, v10, seam1, weld1, weave1, fine, tTorch, wobj0, event11 , loaddata0)

5.2.7 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, TLoad)

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
TLoad	loaddata	负载

指令示例

SpotJ (P10, v200, weld1, fine, tTorch, wobj0, loaddata0)

5.2.8 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, TLoad)

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
TLoad	loaddata	负载

指令示例

SpotL (P10, v200, weld1, fine, tTorch, wobj0, loaddata0)

5.2.9 ArcLOffLine 基于线性运动的弧焊跟踪指令

指令功能

本指令用于沿直线焊缝的激光跟踪弧焊。

指令结构

ArcLOffLine (ScanPoint1, ScanPoint2, OffsetZ, Startpoint , Endpoint , WeldNo , ScaleNO, Speed, Seam, Weld, Weave, Zone, Tool, WObj)

指令参数

参数	数据类型	说明
ScanPoint1	robtarget	焊缝扫描起点焊枪位置
ScanPoint2	robtarget	焊缝扫描终点焊枪位置
OffsetZ	num	两点间 TCP 行走高度
Startpoint	robtarget	焊缝起点位置
Endpoint	robtarget	焊缝终点位置
WeldNo	num	焊缝号（从 1 开始计数）
ScaleNO	num	系数号，焊缝对应的模板号（从 1 开始计数）
Speed	speeddata	运动速度（焊接时无效）
Seam	seamdata	起弧收弧控制参数
Weld	welddata	焊接过程控制参数
Weave	weavedata	焊接过程摆弧参数
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据

指令示例

ArcLOffLine (P1, P2, 3, P3 , P4 , 1 ,1, v200, seam1, weld1, weave1, fine, tool1, wobj0)

说明：P1，P2 点为焊枪停止位置，激光扫描线在 P3，P4 如下图所示：

5.2.10 ArcLOnLine 基于线性运动的激光在线弧焊跟踪过程

指令功能

本指令用于沿直线焊缝的激光在线弧焊跟踪（定义了弧焊的线性运动过程点）。

指令结构

ArcLOnLine(motgrp,Startpoint,Endpoint,WeldNo,ScaleNo,Speed,Seam,Weld,Weave,Zone,Tool,WObj,Tarr,TLoad)

指令参数

参数	数据类型	说明
motgrp	num	0，左臂；1，右臂。对于单臂，默认缺省。
Startpoint	robtarget	焊缝跟踪起点
Endpoint	robtarget	焊缝跟踪终点
WeldNo	num	焊缝号，需和激光跟踪仪配合使用
ScaleNo	num	系数号，需和标定时使用的系数配合使用
Speed	speeddata	运动速度（焊接时无效）
Seam	seamdata	起弧收弧控制参数
Weld	welddata	焊接过程控制参数
Weave	weavedata	焊接过程摆弧参数
Zone	zonedata	转角参数
Tool	tooldata	工具数据
WObj	wobjdata	工件数据
Tarray	triggdata 数据数组	跟踪数据
TLoad	loaddata	负载数据

指令示例

ArcLOnLine(P3,P2,1,1,v100,shareSeam0,shareWeld0,weave0,fine,shareTool0,wobj0,nil,load0)

说明：如图所示，P3-P2 为焊接起始点和终点。跟踪过程中，TCP 将自动运行到 P3-P2 反向延长线的 P1 点，此时激光线的位置在 P3 点，然后开始寻位，运动到 P3 点后起弧并开始跟踪，焊接到 P2 点后断弧，此时激光线的位置在 P4 点。

因此，请确保有足够的工作空间，以使机器人在跟踪过程中不会因此发生碰撞。

5.2.11 TrackCalibrate 激光跟踪仪标定指令

指令功能

本指令用于激光跟踪仪沿着焊缝两侧做标定，生成标定系数。

指令结构

TrackCalibrate(ToPoint1, ToPoint2, WeldNo, ScaleNO, Speed, Zone, Tool, WObj)

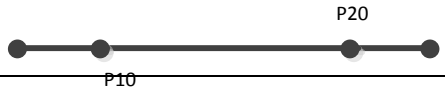
指令参数

参数	数据类型	说明
ToPoint1	robtarget	焊枪标定点位置 1
ToPoint2	robtarget	焊枪标定点位置 2
WeldNo	num	焊缝号（从 1 开始计数）
ScaleNO	num	系数号，焊缝对应的模板号（从 1 开始计数）
Speed	speeddata	运动速度
Zone	zonedata	转角参数
Tool	tooldata	TCP
WObj	wobjdata	工件数据

指令示例

TrackCalibrate (P10, P20, 1, 1 , v200, fine, tool1, wobj0)

说明：P10，P20 点为焊枪标定点停止位置，如下图所示：



5.2.12 TrackCreateTemplate 创建用于激光焊缝跟踪的模板

指令功能

本指令用于创建激光焊缝跟踪的模板。

指令结构

TrackCreateTemplate(motgrp, Startpoint, Endpoint, Speed, Zone, Tool, WObj, Load)

指令参数

参数	数据类型	说明
motgrp	num	0，左臂；1，右臂。对于单臂，默认缺省。
Startpoint	robtarget	焊缝跟踪起点
Endpoint	robtarget	焊缝跟踪终点
Speed	speeddata	运动速度
Zone	zonedata	转角参数
Tool	tooldata	工具数据
WObj	wobjdata	工件数据
TLoad	loaddata	负载数据

指令示例

TrackCreateTemplate(P3,P2,v100,fine,shareTool0,wobj0,load0)

说明：如图所示，P3-P2 为焊接起始点和终点。创建模板过程中，TCP 将自动运行到 P3-P2 反向延长线的 P1 点，此时激光线的位置在 P3 点，机器人将在此处停止运动，直到用户再次点击“运行”按钮。

因此，请确保有足够的工作空间，以使机器人在跟踪过程中不会因此发生碰撞。

5.2.13 焊接指令的错误返回值

错误返回值

错误代码	错误常量	错误说明
600	AW_ERR_BREAK	"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。
601	AW_ERR_GUN_COLLISION	"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。
602	AW_ERR_VOLTAGE	"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。
603	AW_ERR_CURRENT	"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。
604	AW_ERR_WARNING	"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。
605	AW_ERR_GAS	"焊接异常"：检测到送气信号异常，请检查 IO 设备和焊接。
606	AW_ERR_WIRE_FEED	"焊接异常"：检测到送丝信号异常，请检查 IO 设备和焊机。
607	AW_ERR_ARC_FAILED	"焊接异常"：检测到焊接失败，请检查工作环境。
608	AW_ERR_ARC_CODE	"焊接指令运行出错"
609	AW_ERR_EQUIP	"焊机异常"：检测到焊机异常，请检查焊机。
610	AW_ERR_EQUIP_STOP	"焊机异常"：检测到焊机停止时出错，请关闭电源，检查焊机。
611	AW_ERR_ARCING	"焊机异常"：检测到在焊接过程中出错，请关闭电源，检查焊机。
612	AW_ERR_COMMUNCTION	"焊机异常"：检测到焊机与机器人通信异常。
613	AW_ERR_PARAMENTS	"焊机异常"：检测到配置的电流电压校准范围错误。

614	AW_ERR_OVER	"焊机异常"：检测给定的电流电压超出范围。

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）

6.2 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, loaddata0)

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)

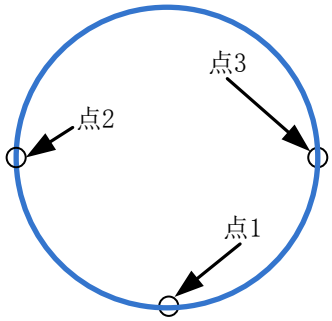
指令参数

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

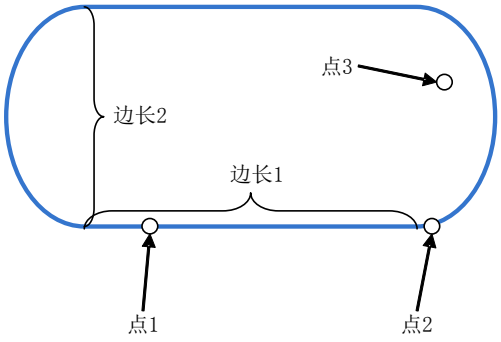
指令算法

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



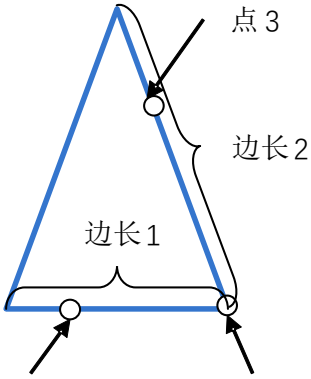
-----腰形孔直线和圆弧相切的地方肯定在点 2 位置

-----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

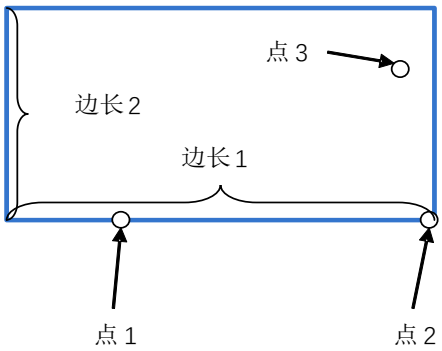


-----三角形的其中一个顶点就在点 2 位置

-----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



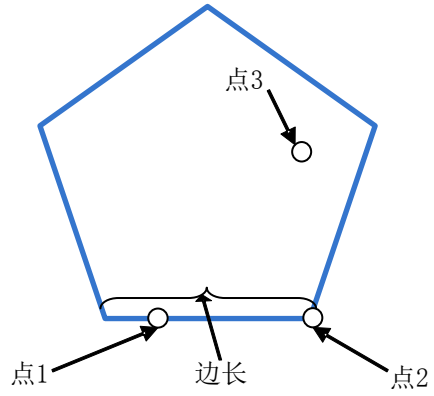
-----矩形的一个顶点肯定在点 2 位置

-----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



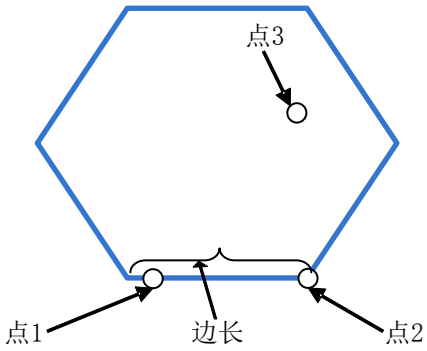
-----正五边形的一个顶点肯定在点 2 位置

-----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, loaddata0)

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, loaddata0)

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, P20, v20, cut1, fine, tLaser, wobj0, loaddata0)

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, P20, v20, cut1, fine, tLaser, wobj0, loaddata0)

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, loaddata0)

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, loaddata0)

7 常用函数

函数，指的是对一系列的指令的一个封装，用户在做类似的事情时，只需要调用相应的函数即可。

7.1 GetJointTarget 获取当前关节位置

函数功能

获取指定机器人当前的关节位置数据，单位是角度。

函数结构

GetJointTarget (motgrp)

函数参数

motgrp

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)
```

7.2 GetRobTarget 获取当前机器人位置

函数功能

获取指定机器人当前的位置数据。

函数结构

GetRobTarget (motgrp,tool,wobj)

函数参数

Motgrp

机械单元名称

tool

数据类型：tooldata

使用的工具坐标系

wobj

数据类型：wobjdata

工件坐标系

调用示例

```
local databody = GetRobTarget("ROL_L", tool0, wob0)
print(databody.name)
print(databody.trans.x,databody.trans.y,databody.trans.z)
```

8 IO 系统配置

8.1 IO 系统概述

所谓 IO 系统，是指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。

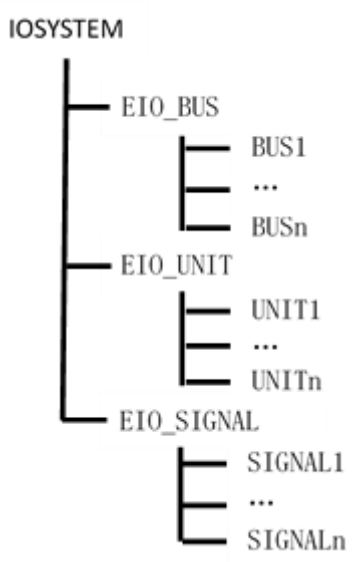
IO 系统目前支持以 EhterCAT 和 ModBus 为通信协议的 IO 设备，控制器在出厂时内嵌了华泰的 IO 板，相应的也需要在配置文件里增加对该板的支持；如果需要支持外部的 PLC 主站，还需要对系统的 IO 配置做出相应的修改。

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*第 4 章 常用指令*。

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，本章将详述具体的配置方法。

8.2 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（如 IP 地址、端口号等）；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。

8.3 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。

注意：

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。

跟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。

8.3.1 数据总线

控制器支持以下数据总线：

EtherCAT	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	目前仅内嵌支持华泰 IO 板
Modbus	通过网络接口和 PLC 主站实现通信	支持 Siemens 的 PLC，以及支持 MODBUS 的其它厂商 PLC
Virtual	虚拟总线	不需要实际的 IO 从站就可以模拟 IO 信号

在定义总线的时候，可以定义多个总线，但是总线类型必须（BusType）必须是三种类型（ECAT, MODBUS, VIRTUAL）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	用户可随意定义	ECAT	NA	NA
Modbus	用户可随意定义	MODBUS	PLC 主站 ip 地址	PLC 主站连接的端口
Virtual	用户可随意定义	VIRTUAL	NA	NA

示例

EIO_BUS:

-Name "Virtual" -BusType "VIRTUAL"

-Name "Ecat" -BusType "ECAT"

-Name "Modbus" -BusType "MODBUS" -Address "192.168.1.10" -Port "502"

-Name "DeviceNet" -BusType "DEVICENET"

8.3.2 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，必须是已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，一个数据采集版，对应从站设备的物理地址

示例

EIO_UNIT:

-Name "BOARD10" -Bus "Virtual" -Node "0" –MacID “1”

-Name "BOARD20" -Bus "Ecat" -Node "0" –MacID “2”

-Name "BOARD30" -Bus "Modbus" -Node "0" –MacID “1”

8.3.3 数据点

定义具体的 IO 信号，具体信息保护如下。

名字(Name)	信号名字，英文字符串，字符数量小于 32
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
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)

所有地址的起始序号都是 0。

EtherCAT 和 Virtual 总线的地址都是从 0 开始，依次往后排，不区分数字量和模拟量，地址映射(unitmap)只代表一个变量（数字量或模拟量）的索引值。

Modbus 总线的地址是从 0 开始，数字量和模拟量的地址映射(unitmap)都以 bit 为单位，对应一个变量的开始 bit 地址。即一个数字量对应一个 bit 位，一个模拟量对应于 PLC 中的

两个 16 位寄存器， 共 32 个 bit 位。寄存器的序号是从 0 开始计的，寄存器内的位数也是从 0 开始计的。举例说明：若一个数字量对应的是 1 号寄存器的第 0 位，则它对应的地址映射(unitmap) 是 16×1+0 = 16；若一个数字量对应的是 2 号寄存器的第 1 位，则它对应的地址映射(unitmap)应该是 16×2 +1 = 33；若一个模拟量对应的是 6 号寄存器和 7 号寄存器，则它对应的地址映射(unitmap)是 16× 6 = 96。Modbus 所有的数据类型公用一个地址空间，一般前面是输出量，后面是输入量。

DeviceNet 总线的地址也是从 0 开始，数字量和模拟量对应的地址映射(unitmap)也是基于 bit 位的，输出量(DO/AO)共享一个地址空间，输入量(DI/AI)共享一个地址空间；数字量占用一个 bit 位，模拟量占用多个 bit 位。

示例

EIO_SIGNAL:

#Virtual 总线

-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

-Name "AI10_0" -SignalType "AI" -Unit "BOARD20" -UnitMap "3" -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"DI"-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

8.4 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。

8.4.1 系统输出

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	暂时无效
SYS_EM_STOP_STATE	1 – 程序处于停止状态， 0 – 程序处于非停止状态

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 就可以系统和系统 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，DO_MOTOR_ON 就会被置为 1； 当电机掉电时， 该信号就会被置为 0。

8.4.2 系统输入

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

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。一旦系统输入和某个外部输入关联了，就无法单独控制系统输入来控制机器人行为。

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 编程示例

9.1 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(0,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(0,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(0,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(0,P10,v1000,z5,tool1,wobj1)

关节绝对运动至 JHome 点位

MoveAbsJ(0,JHome,v200,fine,tool1,wobj1,Load_6kg)

9.2 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(0, P00, v1000, fine, tool1, wobj0, loaddata0)

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, loaddata0)

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, loaddata0)

结束焊接，并线性运动以 1000mm/s 速度至 P40 点位

MoveL(0,P40, v1000, fine, tool1, wobj0, loaddata0)

10 术语表和索引

10.1 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。

10.2 索引

X

Y

Z

ZONEDATA

G-UM-SW00001-A5 版本 A

智殷自动化科技有限公司
中国，上海
电话：+86 021- xxxxx
传真：xxxxxxx