适用机型：全系列
出版状态：标准
产品版本：G
软件版本：V1.5.8

九众九机器人有限公司全权负责本控制系统用户操作及编程指南的编制、印刷

版权所有，保留一切权利。

未得到九众九机器人有限公司的许可，任何单位和个人不得擅自复制，不得以任何形式（包括资料和出版物）进行传播。

版权所有，侵权必究。内容如有改动，恕不另行通知。

序言

内容提要

本操作手册对九众九机器人编程进行了全面系统的阐述，可作为使用九众九机器人编程的参考资料。

为了确保能够正确的使用九众九机器人软件功能，请在使用前仔细阅读本使用操作手册。

阅读对象

操作人员

机器人编程人员

工程维护人员

用户技术支持人员

内容说明

本说明书内容会有补充和修改，请经常留意我公司网站，更新说明书。

主要特点

九众九机器人示教器界面采用逐级分类式的管理形式，用户可以通过示教器上的物理按键及触摸屏上的虚拟热键对机器人进行控制。操作界面简洁明了，使用方法符合人类感观认知，简单易懂。

安全标记

本使用说明书中，与安全相关的内容，使用下列标记。附有安全标记的叙述、内容重要，请务必遵守。

危险 错误使用时，会引起危险情况，可能导致人身伤亡。

注意 错误使用时，会引起危险，可能导致人身轻度或重度伤害和设备损坏。

重要 用户需要遵守和重点注意的部分。

目 录

第一章 前言

1.1 目的

这篇文档描述了终端用户如何在九众九机器人上编写程序，我们假设用户已经对机器人有了一定的了解。

1.2 编程前提

- 电气技术相关经历
- PLC 编程经验
- 面向对象编程经验
- 对机器人语言及机器人应用有所了解

1.3 文件架构

1）工程文件

工程文件是一个目录文件夹，后缀名为：“.sr”，工程文件新建时会自动生成一个名为_project.srd 的工程变量数据文件。

2）用户程序文件

用户程序文件包含用户指令的文件，文件后缀为“srp”。

3）用户数据文件

数据文件包含用户程序文件中的变量及对应的值，文件后缀为“srd”。

第二章 数据类型

2.1 基本数据类型

类型	含义	取值范围	对应 C++类型
BOOL	布尔类型	TRUE / FALSE (1/0)	Bool
INT	整型	-2^15 ~ 2^15-1	Short
UINT	无符号整型	0 ~ 2^16-1	Unsigned short
LINT	长整型	-2^31 ~ 2^31-1	Long int
ULINT	无符号长整型	0 ~ 2^32-1	Unsigned long int
REAL	单精度浮点型		Float
LREAL	双精度浮点型		Double
PERCENT	百分数	0-100 的整数	
BOOLEXT	外部 BOOL 变量	TRUE / FALSE (1/0) 端口号为 0-31	
DINTEXT	外部 INT 变量	-2^15 ~ 2^15-1 端口号为 0-31	
REALEXT	外部 REAL 变量	端口号为 0-31	
DWORDEXT	外部 DWORD 变量	-2^31 ~ 2^31-1 端口号为 0-31	

2.1.1 BOOL

例：BOOL bool0 := TRUE

2.1.2 INT

例：INT int1 := -10

2.1.3 UINT

例：UINT uint1 := 100

2.1.4 LINT

例：LINT lint1 := -1000

2.1.5 ULINT

例：ULINT ulint1 := 1000

2.1.6 REAL

例：REAL real0 := 1.1

2.1.7 LREAL

例：LREAL lreal0 := 1.1

2.1.8 PERCENT

百分比变量，范围为 0-100 中的整数

例：PERCENT per0:=70

2.1.9 BOOLEXT

外部（PLC）变量类型。当需要和外部 bool 变量交互时，新建并使用该类型的变量作为参数。该类型变量由 Port 和 Value 组成，Port 是数字(0-31)，Value 是 bool (即上文的 BOOL)型。定义好的数据如下格式：

例：BOOLEXT ext1 := {0 , TRUE}

2.1.10 DINTEXT

外部（PLC）变量类型。当需要和外部 bool 变量交互时，新建并使用该类型的变量作为参数。该类型变量由 Port 和 Value 组成，Port 是数字(0-31),Value 是 DINT(即上文的 LINT)型.定义好的数据如下格式：

例：DINTEXT ext1 := {0 , 12}

2.1.11 REALEXT

外部（PLC）变量类型。当需要和外部 REAL 变量交互时，新建并使用该类型的变量作为参数。该类型变量由 Port 和 Value 组成，Port 是数字(0-31),Value 是 REAL(即上文的 FLOAT)型.定义好的数据如下格式：

例：REALEXT ext1 := {0 , 1.1}

2.1.12 DWORDEXT

外部（PLC）变量类型。当需要和外部 DWORD 变量交互时，新建并使用该类型的变量作为参数。该类型变量由 Port 和 Value 组成，Port 是数字(0-31),Value 是 DWORD(即上文的 ULINT)型.定义好的数据如下格式：

例：DWORDEXT ext1 := {0 , 222}。

2.1.13 CLOCK

时钟变量，本质是无符号长整型，用来存储时间长度，单位是毫秒。最大时间间隔 49 天（49*24*3600 秒）

例如：

CLOCK ck := 0;

2.1.14 ARRAY

数组变量，包含类型和数组长度两个参数，类型包含 BOOL/INT/UINT/LINT/ULINT/REAL/LREAL/AXISPOS/CARTPOS/ROBOTAXISPOS/ROBOTCARTPOS，数组长度为 1-100。

例如：

ARRAY array0 := {BOOL,10};

2.1.15 STRING

字符串变量，长度为 0-60。

例如：

STRING string0 =sr;

2.1.16 LINE

线方程变量。

例如：

LINE line0 ={0.0,0.0,0.0,1.0,1.0,1.0};

这 6 个 real 类型的变量分别为：

线上点 X；线上点 Y；线上点 Z；线方向向量 X；线方向向量 Y；线方向向量 Z

2.1.17 PLANE

平面方程变量。

例如：

PLANE plane0 ={0.0,0.0,0.0,1.0,1.0,1.0};

这 6 个 real 类型的变量分别为：

平面上点 X；平面上点 Y；平面上点 Z；平面法向量 X；平面法向量 Y；平面法向量 Z

2.2 位置数据类型

复合数据类型的定义参照已经定义好的 ROBOT 数据结构类型中的结构，在用户创建变量时，把它们作为选项提供给用户，不允许用户自定义数据类型。

数据类型	含义
AXISPOS	关节位置（带附加轴）
CARTPOS	笛卡尔位置（带附加轴）
AXISPOSEXT	外部关节位置（带附加轴）
CARTPOSEXT	外部笛卡尔位置（带附加轴）
ROBOTAXISPOS	机器人关节位置（不带附加轴）
AUXAXISPOS	附加轴关节位置（不带机器人轴）
ROBOTCARTPOS	机器人笛卡尔位置（不带附加轴）
ROBOTAXISPOSEXT	外部机器人关节位置（不带附加轴）
ROBOTCARTPOSEXT	外部机器人笛卡尔位置（不带附加轴）
AXISDIST	关节类型相对距离（带附加轴）
CARTDIST	笛卡尔类型相对距离（带附加轴）
CARTPOSEE	带 XYZ 偏移量的笛卡尔位置(带附加轴)
POSOFFSET	位置偏差量（不带附加轴）

POSITION_

该类型可以是：AXISPOS、CARTPOS、AXISPOSEXT、CARTPOSEXT、ROBOTAXISPOS、AUXAXISPOS、ROBOTCARTPOS、ROBOTAXISPOSEXT、ROBOTCARTPOSEXT、CARTPOSEE

2.2.1 AXISPOS

带附加轴的关节位置，最多 6 个机器人轴+6 个附加轴，每个量的类型是 REAL，单位是度（°）。

例：AXISPOS ap1 := {10.1 , 10.1 , 10.1 , 10.1 , 10.1 , 10.1, 10.1, 10.1, 10.1, 10.1, 10.1, 10.1}

2.2.2 CARTPOS

笛卡尔坐标系下的空间位置，包含附加轴的位置值，最多 6 个附加轴。变量由 X、Y、Z、A、B、C、Aux1~6、Mode 组成，其中 X~Aux6 是 REAL 类型，Mode 是 UINT 类型，范围为[0，31]。

例：CARTPOS cp1 := {800 , 10.1 ,1027 , 0 , 90 , -180 , 10.1 , 10.1 , 10.1 , 10.1 , 10.1 , 10.1 , 0}

2.2.3 AXISPOSEXT

外部关节位置，包含附加轴的位置值，端口号为 UINT 类型，范围为[0，31]。

例：AXISPOSEXT ape0:= 5

2.2.4 CARTPOSEXT

外部笛卡尔位置，包含附加轴的位置值，端口号为 UINT 类型，范围为[0，127]。

例：CARTPOSEXT cpe1:= 5

2.2.5 ROBOTAXISPOS

机器人关节位置，最多 6 个关节，每个量的类型是 REAL，单位是度（°）。

例：ROBOTAXISPOS rap1 := {10.1 , 10.1 , 10.1 , 10.1 , 10.1 , 10.1}

2.2.6 AUXAXISPOS

附加轴关节位置值，最多 6 个附加轴，每个量的类型是 REAL，单位是度（°）。

例：AUXAXISPOS raxp1 := {10.1 , 10.1 , 10.1 , 10.1 , 10.1 , 10.1}

2.2.7 ROBOTCARTPOS

笛卡尔坐标系下的空间位置，不包含附加轴的值，变量由 X、Y、Z、A、B、C、Mode 组成，其中 X~C 是 REAL 类型，Mode 是 UINT 类型，范围为[0,31]。

例：ROBOTCARTPOS　rcp1 := {10.1 , 10.1 , 10.1 , 10.1 , 10.1 , 10.1 , 1}

2.2.8 ROBOTAXISPOSEXT

外部机器人轴位置，端口号为 UINT 类型，范围为[0,31]。

例：ROBOTAXISPOSEXT rape1 := 5

2.2.9 ROBOTCARTPOSEXT

外部笛卡尔空间位置，端口号为 UINT 类型，范围为[0,127]。

例：ROBOTCARTPOSEXT rcpe1 := 5

2.2.10 AXISDIST

关节类型的相对距离，包含附加轴，最多 6 个机器人轴+6 个附加轴，每个量的类型是 REAL，单位是度（°）。

例：AXISDIST ad1 := {10.1 , 10.1 , 10.1 , 10.1 , 10.1 , 10.1, 10.1, 10.1, 10.1, 10.1, 10.1, 10.1}

2.2.11 CARTDIST

笛卡尔类型的相对距离，包含附加轴，最多 6 个附加轴，变量由 X、Y、Z、A、B、C、Aux1~6 组成，都为 REAL 类型。

例：CARTDIST cd1 := {800 , 10.1 ,1027 , 0 , 90 , -180 , 10.1 , 10.1 , 10.1 , 10.1 , 10.1 , 10.1 }

2.2.12 POSOFFSET

位置偏差量，不带附加轴的值，变量由 VAR1，VAR2，VAR3，VAR4，VAR5，VAR6 组成，都为 REAL 类型。

2.3 坐标系和工具数据类型

数据类型	含义
CARTREF	参考坐标系
CARTREFEXT	外部固定参考坐标系
CARTREFVAR	外部跟踪坐标系
TOOL	工具
TOOLEXT	外部工具

2.3.1 CARTREF

坐标系相对其参考的坐标系的位置，变量由 X、Y、Z、A、B、C、参考坐标系名组成，X~C 均是 REAL 类型。

例：CARTREF ref1 := {1.1 , 1.2 , 1.3 , 1.4 , 1.5 , 1.6, ref0}

2.3.2 CARTREFEXT

外部固定坐标系，包含一个端口号和一个参考坐标系，端口号的取值范围为[0,31]，根据端口号，在 PLC 端设置外部固定坐标系相对于参考坐标系的值。被参考的坐标系只能是固定坐标系。

例：CARTREFEXT refe0:= {5, ref0}

2.3.3 CARTREFVAR

外部跟踪坐标系，包含一个端口号和一个参考坐标系，端口号的取值范围为[0,31]，根据端口号，在 PLC 端设置外部固定坐标系相对于参考坐标系的值，且该值是变化的。被参考的坐标系只能是固定坐标系。

例：CARTREFVAR refv1 := {1, ref0}

2.3.4 TOOL

代表工具的坐标系，即工具的末端相对于法兰坐标系的为值。由 X、Y、Z、A、B、C 组成，均是 REAL 类型。

例：TOOL tool0:= {1.1, 1.2, 1.3,1.4,1.5,1.6}

2.3.5 TOOLEXT

外部工具坐标系，即在 PLC 端设置工具的值，通过端口号传输给示教器，端口号的取值范围为[0,31]。

例：TOOLEXT toolext0:= 3

2.3.6 INTNUM

中断类型变量，只能是全局变量，通过绑定中断程序和 DI 信号实现中断功能。该变量的值没有用到，默认是 0 即可。

例：INTNUM intnum0 := 0

2.4 动态特性与圆滑类型

数据类型	含义
DYNAMIC	动态特性
OVLREL	圆滑百分比
OVLABS	圆滑绝对量
ORITYPE	姿态类型
RAMPTYPE	加速度类型

2.4.1 DYNAMIC

由位置、旋转和关节的速度、加速度、减速度和加加速度组成，一共 12 个量。实际根据执行的路径类型（针对笛卡尔或关节的），使用其中不同的量。12 个量依次是：

PathVel	REAL	位置速度(mm/s)
PathAcc	REAL	位置加速度(mm/s^2)
PathDec	REAL	位置减速度(mm/s^2)
PathJerk	REAL	位置加加速度(mm/s^3)
OriVel	REAL	旋转速度(deg/s)
OriAcc	REAL	旋转加速度(deg/s^2)
OriDec	REAL	旋转减速度(deg/s^2)
OriJerk	REAL	旋转加加速度(deg/s^3)
JointVel	REAL	关节速度(%)
JointAcc	REAL	关节加速度(%)
JointDed	REAL	关节减速度(%)
JointJerk	REAL	关节加加速度(%)

例：DYNAMIC dyn1 := {1000 , 2000 , 2000 , 5000 , 100 , 200 , 200 , 500 , 80 , 80 , 80 , 80}

2.4.2 OVLREL

表示圆滑百分比，为 INT 类型的量，代表百分之几。

例：OVERLAPREL olr0 := 50

2.4.3 RAMPTYPE

由两种类型组成：Sshape 和 Ssine;

2.5 操作运算符

2.5.1 数学操作符

\+ 加法操作

\- 减法操作

* 乘法操作

/ 除法操作

用法：左右两侧可以是数字或基本数据类型变量

2.5.2 关系运算符

< 小于

<= 小于等于

> 大于

>= 大于等于

== 等于

!= 不等于

用法：左右两侧可以是数字或基本数据类型变量

2.5.3 逻辑运算符

&& 与

|| 或

! 非

2.5.4 分隔符及其他符号

: 冒号

用法：用在赋值符号中

, 逗号

用法：用在多个参数之间

() 括号（适用于参数，初始化列表，算术表达式）

用法：用在需要参数的指令后

2.6 打磨变量

打磨变量用在 CLin 指令中，变量由打磨类型（枚举字符串）、打磨个数（整形）、圆半径（REAL）、椭圆长半轴（REAL）、椭圆短半轴（REAL）、椭圆倾斜角（REAL）、幅值（REAL）组成。

例：POLISHDATA polishdata1 := {　eCircle, 1, 0.0, 0.0, 0.0, 0.0,0.0 }

当选择 eCircle 类型时，椭圆相关的参数不需添加，由系统给出默认值。

第三章 指令及编写示例

3.1 机器人运动指令

3.1.1 PTP

1、使用格式：

```
PTP（pos，dyn，ovl，tool，refsys）
```

2、参数：

pos : POSITION_ / /语句的目的位置，必填

OPTIONAL dyn : DYNAMIC_ //动态特性，选填项

OPTIONAL ovl : OVERLAP_ //圆滑数据，选填项

OPTIONAL tool: TOOL_ //工具数据，选填项

OPTIONAL refsys: REFSYS_ //坐标系数据，选填项

3、功能描述：

实现点到点运动，即机器人从当前位置以最快的方式运动到目标点 pos，运行过程中的动态参数和圆滑参数由 dyn 和 ovl 确定，工具和坐标系参数由 tool 和 refsys 确定。

4、编写示例：

PTP(ap1);

机器人从当前位置以最快的方式运动到 ap1 点，动态和圆滑，工具，坐标系参数为默认值。

3.1.2 Lin

1、使用格式：

```
Lin（pos，dyn，ovl，ori，tool，refsys）
```

2、参数：

pos : POSITION_ / /目的位置，必填

OPTIONAL dyn : DYNAMIC_ //动态特性，选填

OPTIONAL ovl : OVERLAP_ //圆滑数据，选填

OPTIONAL ori : ORITYPE //姿态参数，选填

OPTIONAL tool: TOOL_ //工具数据，选填项

OPTIONAL refsys: REFSYS_　//坐标系数据，选填项

3、功能描述：

机器人的 TCP 将会以一个直线运动的方式从初始位置运动到目标位置，运行过程中的动态参数、圆滑数据以及姿态由 dyn、ovl 和 ori 确定，工具和坐标系参数由 tool 和 refsys 确定。

4、编写示例：

Lin (ap1,dyn1,ovl1,ori1);

机器人从当前位置以直线的方式运动到 ap1，运行过程中的动态参数由 dyn1 决定，圆滑数据由 ovl1 确定，姿态根据 ori1 做变化，工具，坐标系参数为默认值。

3.1.3 Circ

1、语句格式：

Circ（hlpPos，pos，dyn，ovl，ori，tool，refsys）

2、参数：

hlpPos：POSITION_　//圆弧上的辅助位置，中间点必填

pos : POSITION_　/ /目的位置，必填

OPTIONAL dyn : DYNAMIC_　//动态特性，选题

OPTIONAL ovl : OVERLAP_　//圆滑数据，选填

OPTIONAL ori : ORITYPE　//姿态参数，选填

OPTIONAL tool: TOOL_　//工具数据，选填项

OPTIONAL refsys: REFSYS_　//坐标系数据，选填项

3、功能描述：

控制 TCP 执行一段圆弧运动，同时动态参数和圆滑数据由 dyn 和 ovl 确定，姿态按照 ori 的选项做变化，工具和坐标系参数由 tool 和 refsys 确定。

4、编写示例：

Circ(ap0,ap1);

通过当前位置点和 ap0、ap1 三点确定唯一的一条弧线，机器人从当前位置沿着这条弧线运动到 ap1 点。

3.1.4 CircleAngle

1、语句格式：

```
CircAngle（hlpPos，pos，angle，dyn，ovl，ori，tool，refsys）
```

2、参数：

hlpPos：POSITION_ //圆弧上的辅助位置，必填

pos : POSITION_ //目的位置，必填

angle: REAL/ /用户指定的角度，必填

OPTIONAL dyn : DYNAMIC_ //动态特性，选填

OPTIONAL ovl : OVERLAP_ //圆滑数据，选填

OPTIONAL ori : ORITYPE //姿态参数，选填

OPTIONAL tool: TOOL_ //工具数据，选填项

OPTIONAL refsys: REFSYS_ //坐标系数据，选填项

3、功能描述：

控制 TCP 执行一段圆弧运动，圆弧的角度由用户指定，动态、圆滑、姿态参数由 dyn,ovl,ori 指定，工具和坐标系参数由 tool 和 refsys 确定。

4、编写示例：

CircleAngle(ap0,ap1,angle0);

通过当前位置点和 ap0、ap1 三点确定唯一的一条弧线，再由 angle0，确定这段圆滑的角度。机器人从当前位置沿着这条弧线运动 angle0 度。

3.1.5 PTPRel

1、语句格式：

```
PTPRel（pos，dyn，ovl，tool，refsys）
```

2、参数：

pos : DISTANCE_ //移动的距离，必填

OPTIONAL dyn : DYNAMIC_ //动态数据，选填

OPTIONAL ovl : OVERLAP_ //圆滑数据，选填

OPTIONAL tool: TOOL_ //工具数据，选填项

OPTIONAL refsys: REFSYS_ //坐标系数据，选填项

3、功能描述：

PTPRel 是一个点到点的相对运动命令，移动的距离是相对机器人初始位置（该位置取决于前一个运动指令）。

4、编写示例：

PTPRel(apd0);

机器人从当前位置以点到点的方式移动 apd0 的距离。

3.1.6 LinRel

1、语句格式：

```
LinRel（dist，dyn，ovl，ori，tool，refsys）
```

2、参数：

dist : DISTANCE_ //移动的距离，必填

OPTIONAL dyn : DYNAMIC_ //动态数据，选填

OPTIONAL ovl : OVERLAP_ //圆滑数据，选填

OPTIONAL ori : ORITYPE //姿态参数，选填

OPTIONAL tool: TOOL_ //工具数据，选填项

OPTIONAL refsys: REFSYS_ //坐标系数据，选填项

3、功能描述：

LinRel 是一个相对直线运动命令，移动的距离是相对机器人初始位置（该位置取决于前一个运动指令）。

4、编写示例：

LinRel(cpd0);

机器人从当前位置直线的方式移动 cpd0 的距离。

3.1.7 WaitTime

1、语句格式：

```
WaitTime (timeMs);
```

2、参数：

timeMs : UINT//时间(单位：毫秒 ms)

3、功能描述：

该指令暂停机器人 timeMs 时间。

4、编写示例：

WaitTime(unit0);

机器人暂停 unit0 毫秒。

3.1.8 LinISR

1、使用格式：

```
BOOLVAR：=LinISR（pos，dyn，ovl，ori，di，respondpos，stoppos，tool，refsys）
```

2、参数：

pos : POSITION_　　/ /目的位置，必填

OPTIONAL dyn : DYNAMIC_　　//动态特性，选填

OPTIONAL ovl : OVERLAP_　　//圆滑数据，选填

OPTIONAL ori : ORITYPE　　//姿态参数，选填

di：DI　//数字输入端口号

respondpos：POSITION_　//接收到信号的位置

stoppos：POSITION_　　//接收到信号后机器人停止的位置

OPTIONAL tool: TOOL_　　//工具数据，选填项

OPTIONAL refsys: REFSYS_　　//坐标系数据，选填项

3、功能描述：

机器人的 TCP 以一个直线运动的方式从初始位置运动到目标位置过程中，若收到 di 信号，则立马降速停止，收到信号的位置信息存储在 respondpos 中，机器人停止的位置信息存储到 stoppos 中，运行过程中的动态参数、圆滑数据以及姿态由 dyn、ovl 和 ori 确定。此外，若在这条直线运行过程中，收到了 di 信号，则将左值赋值为 TRUE，否则，赋值为 FALSE。

注意：respondpos 和 stoppos 的存储的响应位置和停止位置的信息在界面上无法看到，界面上显示的始终是初始值。

4、编写示例：

```
PTP(ap0);
bool0:=LinISR (ap1,dyn1,ovl1,ori1,di0,cp0,cp1);
Lin(cp0);
```

机器人以直线的方式从 ap0 运行到 ap1 的过程中，若收到了 di0 的信号，则机器人立马降速停止，并把接收到的位置信息存储到 cp0 里，机器人停止的位置信息存储到 cp1 里，将 bool0 置为 TRUE。运行直线过程中的动态参数由 dyn1 决定，圆滑数据由 ovl1 确定，姿态根据 ori1 做变化。然后运行第 3 句，就可以将机器人移动到接收信号的位置。

若运行从 ap0 到 ap1 的直线过程中，没有收到 di0 信号，则直接运行到 ap1，且 bool0 的值为 FALSE。运行第 3 行时，机器人会运行到 cp0 的初值位置，即变量 cp0 显示的位置值。

3.1.9 CLin

1、使用格式：

```
CLin（pos，polishdata，press，dyn，ovl，ori，tool，refsys）
```

2、参数：

pos : POSITION_　　//目的位置，必填

Polishdata//打磨变量，必填项

press//下压量，选填项

OPTIONAL dyn : DYNAMIC_　　//动态特性，选填

OPTIONAL ovl : OVERLAP_　　//圆滑数据，选填

OPTIONAL ori : ORITYPE　　//姿态参数，选填

OPTIONAL tool: TOOL_　　//工具数据，选填项

OPTIONAL refsys: REFSYS_　　//坐标系数据，选填项

3、功能描述：

机器人的 TCP 将会按照打磨变量里设定的轨迹类型运动，可能是圆，椭圆，sin 形。如果下压量不为空，则会在上述轨迹做做整体的偏移，偏移方向是沿着工具的 Z 轴方向。

4、编写示例：

CLin (ap1,polishdata，press，dyn1,ovl1,ori1,tool,ref);

如果 polishdata 内是类型是圆，则机器人从当前位置边画圆边向 ap1 运动，运行过程中的动态参数由 dyn1 决定，圆滑数据由 ovl1 确定，姿态根据 ori1 做变化，工具为 tool，坐标系参数为 ref。

3.1.10 WaitIsFinished

1、语句格式：

```
WaitIsFinished ();
```

2、参数：

无；

3、功能描述：

该指令用于打断预读和圆滑。

4、编写示例：

WaitIsFinished ();

3.2 机器人设置指令

设置指令用来修改参数设置，并应用到后续的运动命令中。

3.2.1 Dyn

1、语句格式：

```
Dyn( dyn);
```

2、参数：

dyn : DYNAMIC_//指定动态特性参数，必填

3、功能：

速度，加速度，减速度，加加速度值为后续运动语句动态参数的默认值。PTP 中 dynamic 使用百分比形式，笛卡尔下运动的动态特性使用绝对数值形式。

4、编写示例：

Dyn(dynamic0);

后面运动语句的动态参数默认值为 dynamic0。

3.2.2 Ovl

1、语句格式：

```
Ovl ( ovl );
```

2、参数：

ovl : OVERLAP_//圆滑参数数据，必填

3、功能描述：

设置运动过程中的圆滑特性，应用到后续的运动中。

4、编写示例：

Ovl(ovearlap0);

后面运动语句的圆滑数据的默认值为 overlap0。

注意：不使用圆滑语句时，程序里使用的圆滑参数是机器人配置文件里配置的圆滑参数。

3.2.3 Ramp

1、语句格式：

```
Ramp( type );
```

2、参数：

type : RAMPTYPE　//速率类型，必填

3、功能描述：

设置速度类型，可选 Sshape 和 Ssine。

4、编写示例：

Ramp(type0);

后面运动语句的速度类型为 type0。

3.2.4 RefSys

1、语句格式：

```
RefSys ( refSys );
```

2、参数：

refSys : REFSYS_　//参考系，必填

3、功能描述：

设置坐标系，应用到后面的运动语句中。

4、编写示例：

RefSys(ref0);

后面机器人位置的参考坐标系为 ref0。

3.2.5 Tool

1、语句格式：

```
Tool( tool );
```

2、参数：

tool : TOOL_　//工具数据，必填

3、功能描述：

指定工具末端相对于法兰的位置和方向，应用到后面的运动语句中。

4、编写示例：

Tool(tool0);

后面语句都是在 tool0 下运动的。

3.2.6 OriMode

1、语句格式

```
OriMode ( ori );
```

2、参数：

ori : ORITYPE_ //姿态方式，枚举类型

3、功能描述：

设置连续运动指令的姿态变化方式，应用到后面的连续运动语句中。

4、编写示例：

OriMode(eVar);

后面连续运动指令的姿态默认方式为 eVar。

3.2.7 CartSpaceActivate

1、语句格式：

```
CartSpaceActivate(space,val);
```

2、参数：

Space：SPACE //工作空间的序号，值为 1-8，必填项

val：BOOL //工作空间的激活状态，TRUE 表示激活，FALSE 表示不激活，必填项。

3、功能描述：

激活某一个笛卡尔工作空间或者是关闭某一个笛卡尔工作空间。

4、编写示例：

CartSpaceActivate(1,TRUE);

激活笛卡尔工作空间 1。

3.2.8 AxisSpaceActivate

1、语句格式：

```
AxisSpaceActivate(space,val);
```

2、参数：

space: SPACE //工作空间的序号，值为 1-8，必填项

val： BOOL //工作空间的激活状态，TRUE 表示激活，FALSE 表示不激活，必填项

3、功能描述：

激活某一个关节工作空间或者是关闭某一个关节工作空间。

4、编写示例：

AxisSpaceActivate(1,TRUE);

激活关节工作空间 1。

3.2.9 SetCartSpaceMode

1、语句格式：

```
SetCartSpaceMode(space,val);
```

2、参数：

space: SPACE //工作空间的序号，值为 1-8，必填项

val： //枚举类型，工作空间的 5 种模式，必填项

eAreaInside---进入工作空间里面会触发工作空间的信号

eAreaOutside---到工作空间外面会触发工作空间的信号

eAreaInside_Stop---进入工作空间里面时机器人会报错停止，且触发信号

eAreaOutside_Stop---到空间空间外面时机器人会报错停止，且触发信号

eAreaOFF---关闭模式，在该模式下不会触发工作空间

3、功能描述：

在程序里面设置笛卡尔工作空间的模式。

4、编写示例：

SetCartSpaceMode(1,eAreaInside);

将笛卡尔工作空间 1 的模式设为 inside 模式。

3.2.10 SetAxisSpaceMode

1、语句格式：

```
SetAxisSpaceMode(space,val);
```

2、参数：

space: SPACE //工作空间的序号，值为 1-8，必填项

val： //枚举类型，工作空间的 5 种模式，必填项

eAreaInside---进入工作空间里面会触发工作空间的信号

eAreaOutside---到工作空间外面会触发工作空间的信号

eAreaInside_Stop---进入工作空间里面时机器人会报错停止，且触发信号

eAreaOutside_Stop---到空间空间外面时机器人会报错停止，且触发信号

eAreaOFF---关闭模式，在该模式下不会触发工作空间

3、功能描述：

在程序里面设置关节工作空间的模式。

4、编写示例：

SetAxisSpaceMode(1,eAreaInside);

将关节工作空间 1 的模式设为 inside 模式。

3.2.11 ReadCartSpaceState

1、语句格式：

```
bool:=ReadCartSpaceState(space);
```

2、参数：

space: SPACE　　//工作空间的序号，值为 1-8，必填项

val：BOOL　　//布尔类型变量，必填项

3、功能描述：

读取笛卡尔工作空间状态的值，若工作空间被触发，则读取的值为 1，否则为 0，并将该值赋给左值变量。

4、编写示例：

bool0:=ReadCartSpaceState(1);

读取笛卡尔工作空间 1 的状态，并将其值赋给 bool0。

3.2.12 ReadAxisSpaceState

1、语句格式：

```
bool:=ReadAxisSpaceState(space);
```

2、参数：

space: SPACE　　//工作空间的序号，值为 1-8，必填项

val：BOOL　　//布尔类型变量，必填项

3、功能描述：

读取关节工作空间状态的值，若工作空间被触发，则读取的值为 1，否则为 0，并将该值赋给左值变量。

4、编写示例：

bool0:=ReadAxisSpaceState(1);

读取关节工作空间 1 的状态，并将其值赋给 bool0。

3.2.13 Override

1、语句格式

```
Override ( percent);
```

2、参数：

percent: PERCENT //百分比

3、功能描述：

动态设置运动速度百分比。

4、编写示例：

Override (percent);

讲当前程序的速度百分比改成 percent 的值。

3.2.14 IndReset

1、语句格式：

```
IndReset();
```

2、参数：

无参数。

3、功能描述：

该指令的功能是当前开启独立轴功能后，对当前六轴的角度进行重置（计算方法：除 360 去余），如果独立轴没有开启，执行该语句会报错“独立轴开启后才能使用”。如果独立轴已开启，当前六轴角度为 370°，则执行 IndReset 后，位置界面显示的当前六轴角度为 10°

3.3 系统函数指令

3.3.1 :=（赋值指令）

1、语句格式：

```
Left:= right;
```

2、参数：

无

3、功能描述：

给变量赋一个值，变量在符号左侧，值（表达式）在右侧。

4、编写示例：

int0:=1;

将 1 赋给整型变量 int0；

3.3.2 //（注释指令）

1、语句格式：

```
//comment;
```

2、参数：

无

3、功能描述：

该指令后是注释内容（同一行）。

4、编写示例：

//PTP(ap0);

该运动指令不再执行。

3.3.3 SaveData

1、语句格式：

```
SaveData(localvar);
```

2、参数：

localvar: 局部变量，类型不限

3、功能描述：

SaveData 只为下班断电前保存重要数据使用。用于将变量的当前值保存到控制器的.srd 文件中。建议该功能用于程序末尾，在断电操作前执行一次。

若在程序内持续循环使用，则.srd 文件中的变量值会不断被写入，若频率太高，影响磁盘寿命。

4、编写示例：

IF(int1= =1)THEN //int1 的初始值为 0

PTP(ap1); //ap1:=10,20,10,0,0,30

PTP(ap0); //ap0:=0,0,0,0,0,0

END_IF

int1:=1; //该语句将 int1 的值改为 1

SaveData(int1); //保存 int1 的过程值，变量监视界面的变量值会同步更新为 1

运行完该程序，即使断电重启，int1 的值仍旧为 1，不会变为初始值 0.

3.3.4 RecordList

1、语句格式：

```
RecordList(Var1, Var2, Var3, Var4, Var5, Var6, Var7, Var8, Var9, Var10);
```

2、参数：

Var: 局部变量，类型为所有的局部变量

3、功能描述：

该语句的功能与 SaveData 语句的功能类似，用于保存变量的过程值，但是用法有所不同，将要记录过程值的变量名作为 RecordList 的参数，最多可以记录 10 个变量的过程值，然后将 RecordList 语句放在程序的第一行，在程序运行过程中，只要要记录的变量的值发生了变化，就会被记录下来，即使断电重启，也会保存断电前的值。

RecordList 只在特殊情况下使用，即对需要保存过程值的变量使用，否则读写文件太频繁，会影响磁盘的寿命。

4、编写示例：

RecordList(int1,int2)

While(int1) DO //int1 的初始值为 100

int1:=int1-1;

PTP(ap0);

PTP(ap1);

PTP(ap2);

int2:=int2+1; //int2 的初始值为 0

WaitTime(unit3); //unit3 为 1000

END_WHILE

该程序循环运行两次后，int1 的值会变为 98，int2 的值会变为 2，此时无论是卸载程序还是断电重启，int1 的值会保持为 98，int2 的值会保持为 2。

3.3.5 Message

1、语句格式：

```
Message(message,type,var1,var2);
```

2、参数：

message：STRING　//自定义的信息，为字符串类型

type：枚举类型，eInfo/eWarining/eError　//自定义信息的类型

var1：任何变量类型　//替换 message 里的“%1”

var2：任何变量类型　//替换 message 里的“%2”

3、功能描述：

用户可以利用 Message 语句自定义信息，自定义的信息类型可以是 Information、Error 或 Warning。

如果是 Error，运行到该语句时会报错停止，而且会弹出一个错误提示框，错误提示框里显示为 message 的值，且 message 里的“%1”被 var1 的变量名或值替换，“%2”被 var2 的变量名或值替换，若 var1、var2 变量的参数只有一个，则 message 里的“%1”“%2”就会被变量的值替换；若 var1、var2 变量的参数有两个及以上，则 message 里的“%1”“%2”就会被变量名替换。此外，点击错误弹出框里的确认按钮，信息界面的所有错误都会被清除掉。

如果是 Warning 或者 Information，则运行到 message 语句时，会在示教器上的错误状态栏显示该信息，但是不会影响机器人的运行状态。

4、编写示例：

```
int1:=3;
PTP(ap0);
PTP(ap1);
Message(“outint %1 outpos %2 ”,eError,int1,ap0);
PTP(ap2);
```

该程序运行到 message 语句时会报错停止，且会弹出一个错误信息框，里面的错误描述为“outint 3 outpos ap1 ”，点击错误弹出框里的确认按钮，就能清除所有的错误信息。

3.3.6 MessageErase

1、语句格式：

```
MessageErase();
```

2、参数：

无参数

3、功能描述：

该语句用于清除信息栏里的所有信息，包括 Waring 和 Information。

4、编写示例：

```
int1:=3;
PTP(ap0);
PTP(ap1);
Message(“outint %1 outpos %2 ”,eWarning,int1,ap0);
WaitTime(unit0);
MessageErase();
```

运行到最后一句时，会将 Message 语句产生的警告信息清除掉。

3.3.7 Pause

1、语句格式：

```
Pause();
```

2、参数：

无参数。

3、功能描述：

该语句的作用与 Stop 按键的功能一样，用于暂停程序，在程序运行过程中，运行到该语句时，机器人停止，程序处于暂停状态，再按 start 键时，程序继续运行。

注意：

该语句会中断预读，即不会预读该语句之后的程序语句；

pause 语句后面若没有接运动语句，直接接同步 IO 语句，同步 IO 语句会失效；

手动模式下连续运行程序时，遇到 pause 语句，需要松开 start 键后再运行，否则会一直停留在 pause 语句。

4、编写示例：

```
PTP(ap0);
PTP(ap1);
Pause();
PTP(ap2);
```

运行到第 3 行时，机器人会降速停止，程序运行状态变为暂停状态。再按 start 键时，程序会接着运行 PTP(ap2)语句。

3.3.8 GetCurrentRobotPos

1、语句格式：

```
GetCurrentRobotPos (Pos,Tool,Ref);
```

2、参数：

Pos："AXISPOS", "CARTPOS", "ROBOTAXISPOS", "ROBOTCARTPOS"类型。

Tool：工具变量（可选，默认使用最近设置的工具）。

Ref：坐标系变量（可选，默认使用最近设置的坐标系）。

3、功能描述：

获取机器人当前位置和姿态，并转换为语句中设置的坐标系和工具下的值。

注意：

该语句会中断预读，即不会预读该语句之后的程序语句；

4、编写示例：

示例 1：

```
GetCurrentRobotPos(pos);//获取 Default,WORLD 下的位置存储到 pos
```

示例 2：

```
Tool(too10);
Refsys(ref0);
Lin(ap1,tool2);
GetCurrentRobotPos(pos);//获取 tool0,ref0 下的位置存储到 pos
```

示例 3：

Tool(too10);

Refsys(ref0);

Lin(ap1,tool2);

GetCurrentRobotPos(pos,tool3,ref3);//获取 too13,ref3 下的位置存储到 pos

3.3.9 ReturnMainProgram

1、语句格式：

```
ReturnMainProgram (int);
```

2、参数：

int：程序行号，必选参数，范围为 0-10000。

3、功能描述：

ReturnMainProgram 语句用于从中断程序或子程序，返回到主程序，可以指定返回到主程序的某一行

注意：

该语句重新加载程序，程序变量会变为初始值。

4、编写示例：

主程序：

Connect(intnum0,program0);

ISignal(jia_close,TRUE,intnum0);

PTP(ap0);

PTP(ap1);

中断程序 program0：

PTP(cp1);

ReturnMainProgram(3);

PTP(cp2);

当运行中断程序 program0 的 ReturnMainProgram(3)语句时，程序不会再执行 ReturnMainProgram(3)后面的 PTP(cp2),会直接跳到主程序的第三行，执行 PTP(ap0)，此时给中断信号不会响应中断。

3.4 数学函数指令

3.4.1 SIN

1、语句格式：

```
real:= SIN(deg);
```

2、参数：

参数 deg 可以是 INT，FLOAT，DOUBLE 类型（以°为单位）。

3、功能描述：

将一个实数的正旋值赋给另一个实数。

4、编写示例：

a:=SIN(b);

将实数 b 的正旋值赋给实数 a。

3.4.2 COS

1、语句格式：

```
real:= COS(deg);
```

2、参数：

参数 deg 可以是 INT，FLOAT，DOUBLE 类型（以°为单位）。

3、功能描述：

将一个实数的余弦值赋给另一个实数。

4、编写示例：

a:=COS(b);

将实数 b 的余旋值赋给实数 a。

3.4.3 TAN

1、语句格式：

```
real:= TAN(deg);
```

2、参数：

参数 deg 可以是 INT，FLOAT，DOUBLE 类型（以°为单位）。

3、功能描述：

将一个实数的正切值赋给另一个实数。

4、编写示例：

a:=TAN(b);

将实数 b 的正切值赋给实数 a。

3.4.4 LN

1、语句格式：

```
real:= LN(deg);
```

2、参数：

参数 deg 可以是 INT，FLOAT，DOUBLE 类型（以°为单位）。

3、功能描述：

将一个实数的对数值赋给另一个实数。

4、编写示例：

a:=LN(b);

将实数 b 的对数值赋给实数 a。

3.4.5 PosReset

1、语句格式：

```
PosReset(pos,distance,outpos);
```

2、参数：

pos : POSITION_　　//语句的初始位置，必填

diatance: OFFSET　　//位置偏移量，必填

outpos: POSITION_　　//语句的输出位置，必填

3、功能描述：

PosReset 语句能实现位置点的加法运算，即将 pos 的值加上 distance 的偏差后，赋值给 outpos，该语句不是运行语句，运行该语句时，机器人不会动。

4、编写示例：

PosReset(ap0,posoffset0,ap1); //ap0=0,0,0,0,0,0　posoffset0=10,10,10,10,10,10

PosReset(cp0,posoffset0,cp1); //cp0=872,0,920,0,70,-80

运行完第 1 行语句后，ap1 的值变为：10,10,10,10,10,10

运行完第 2 行语句后，cp0 的值变为：882,10,930,10,80,-70

3.4.6 SHL

1、语句格式:

```
int:=SHL（Expr，digit）；
```

2、参数：

Expr 将要被移动的值

digits 该值中将要移动的位数

3、功能描述：

将 Expr 按位左移 digit 位后赋给 int。

4、编写示例：

c:=SHL(a,b);

将 a 按位左移 b 位后赋给 c。

3.4.7 SHR

1、语句格式：

```
int:=SHR（Expr，digit）；
```

2、参数：

Expr 将要被移动的值

digits 该值中将要移动的位数

3、功能描述：

将 Expr 按位右移 digit 位后赋给 int。

4、编写示例：

c:=SHR(a,b);

将 a 按位右移 b 位后赋给 c。

3.4.8 SetBit

1、语句格式：

```
int:=SetBit（val,bitNr）；
```

2、参数：

val 将要被用来设置的值

bitNr 需要设置的位

3、功能描述：

将 val 的 bitNr 位的值设为 1，再赋值给 int。

4、编写示例：

C:=SetBit(a,b);

将 a 的第 b 位置为 1，再赋值给 c。

3.4.9 GetPointBy2Lines

1、语句格式：

```
pos := GetPointBy2Lines (line1,line2);
```

2、参数：

pos : CARTPOS/ROBOTCARTPOS //两条直线的交点或公垂线的中间点，必填

line1: LINE //直线方程或 XAxis/YAxis/ZAxis，必填

line2: LINE //直线方程或 XAxis/YAxis/ZAxis，必填

3、功能描述：

GetPointBy2Lines 语句获取两条直线的交点。若两条直线相交，则返回交点的位置 XYZ，若两条直线异面，则返回公垂线的中间点的位置 XYZ，若两条直线平行，则报错。

4、编写示例：

cp0 := GetPointBy2Lines(line0, line1)

返回直线 line0 和直线 line1 的交点位置，并赋给 cp0 的 XYZ。

3.4.10 GetPointByLinePlane

1、语句格式：

```
pos := GetPointByLinePlane (line,plane);
```

2、参数：

pos : CARTPOS/ROBOTCARTPOS //直线与平面的交点，必填

line: LINE //直线方程或 XAxis/YAxis/ZAxis，必填

plane: PLANE //平面方程或 XOY/YOZ/ZOX，必填

3、功能描述：

GetPointByLinePlane 语句获取直线和平面的交点。若直线与平面平行，则报错。

4、编写示例：

cp0 := GetPointByLinePlane(line0, plane0)

返回直线 line0 和平面 plane0 的交点位置，并赋给 cp0 的 XYZ。

3.4.11 GetPointBy3Points

1、语句格式：

```
posA := GetPointBy3Points (P1,P2,P3);
```

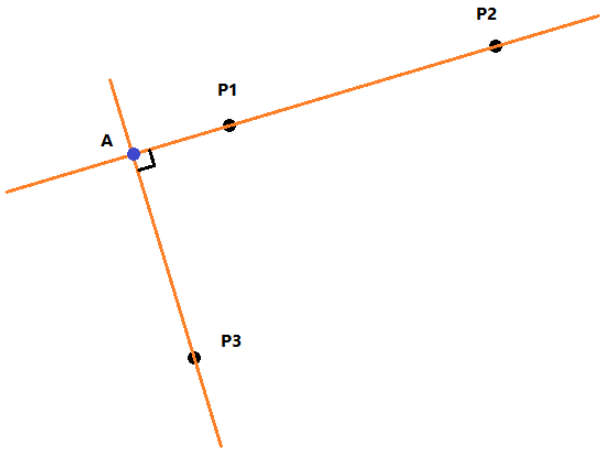
2、参数：

posA : CARTPOS/ROBOTCARTPOS //垂线交点，必填

P1: CARTPOS/ROBOTCARTPOS //直线上的点 1，必填

P2: CARTPOS/ROBOTCARTPOS //直线上的点 2，必填

P3: CARTPOS/ROBOTCARTPOS //直线外的点 3，必填



3、功能描述：

GetPointBy3Points 语句用于获取已知平面上三点的垂线交点，如上图，语句的三个参数分别是 P1，P2，P3，然后返回经过 P3 且与 P1P2 直线垂直的垂足。

4、编写示例：

cp0 := GetPointBy3Points (cp1,cp2,cp3)

返回过点 cp3 且与过点 cp1 和 cp2 的直线垂直的垂足，并将垂足的 XYZ 赋值给 cp0 的 XYZ。

3.4.12 GetPointBy4Points

1、语句格式：

```
posA := GetPointBy4Points (P1,P2,P3,P4);
```

2、参数：

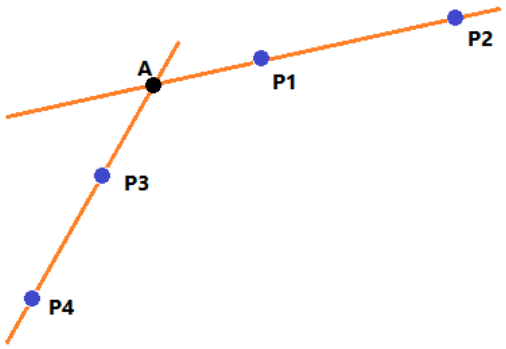
posA : CARTPOS/ROBOTCARTPOS //直线交点，必填

P1: CARTPOS/ROBOTCARTPOS //直线 1 上的点 1，必填

P2: CARTPOS/ROBOTCARTPOS //直线 1 上的点 2，必填

P3: CARTPOS/ROBOTCARTPOS //直线 2 上的点 1，必填

P4: CARTPOS/ROBOTCARTPOS //直线 2 上的点 2，必填



3、功能描述：

GetPointBy4Points 语句用于获取平面四点确定的两条直线的交点，如上图，语句的三个参数分别是 P1，P2，P3，P4，然后返回直线 P1P2 和直线 P3P4 的交点。若构成的两条直线不在同一个平面，则返回公垂线的中点，若两条直线平行，则报错。

4、编写示例：

cp0 := GetPointBy4Points (cp1,cp2,cp3,cp4)

返回经过 cp1 和 cp2 与经过 cp3 和 cp4 两条直线的交点，并将交点的 XYZ 赋值给 cp0 的 XYZ。

3.4.13 GetVPointByPlanePoint

1、语句格式：

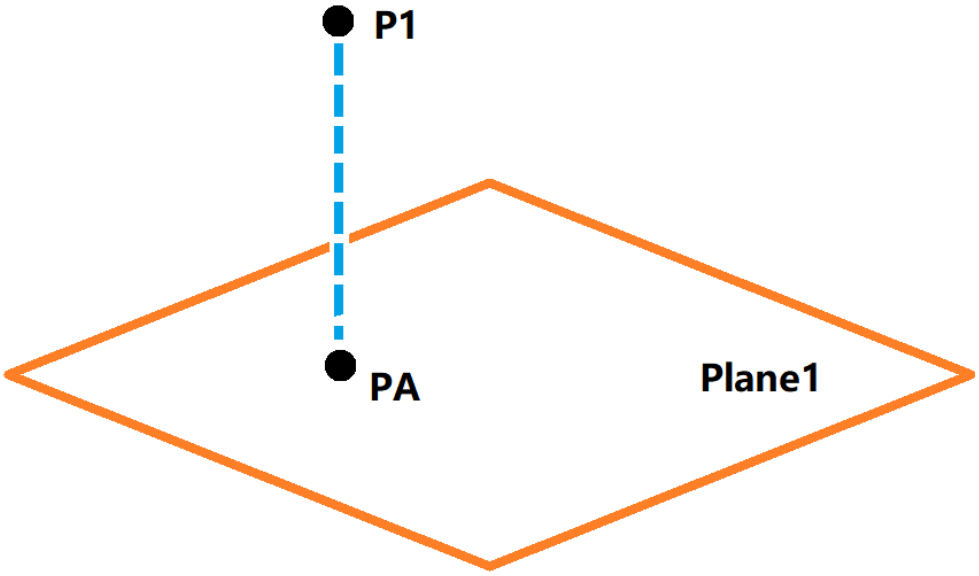
```
posA := GetVPointByPlanePoint (Pos,plane);
```

2、参数：

posA : CARTPOS/ROBOTCARTPOS //直线交点，必填

Pos: CARTPOS/ROBOTCARTPOS //空间上的点，必填

plane: PLANE //平面方程或 XOY/YOZ/ZOX，必填



3、功能描述：

GetVPointByPlanePoint 语句用于获取点在平面上的投影点的 XYZ，若点在给定的平面上，则直接返回该点的 XYZ。

4、编写示例：

cp0 := GetVPointByPlanePoint (cp1,plane1)

返回 cp1 到平面 plane1 的投影点，并将投影点的 XYZ 赋值给 cp0 的 XYZ。

3.4.14 GetVPointByLinePoint

1、语句格式：

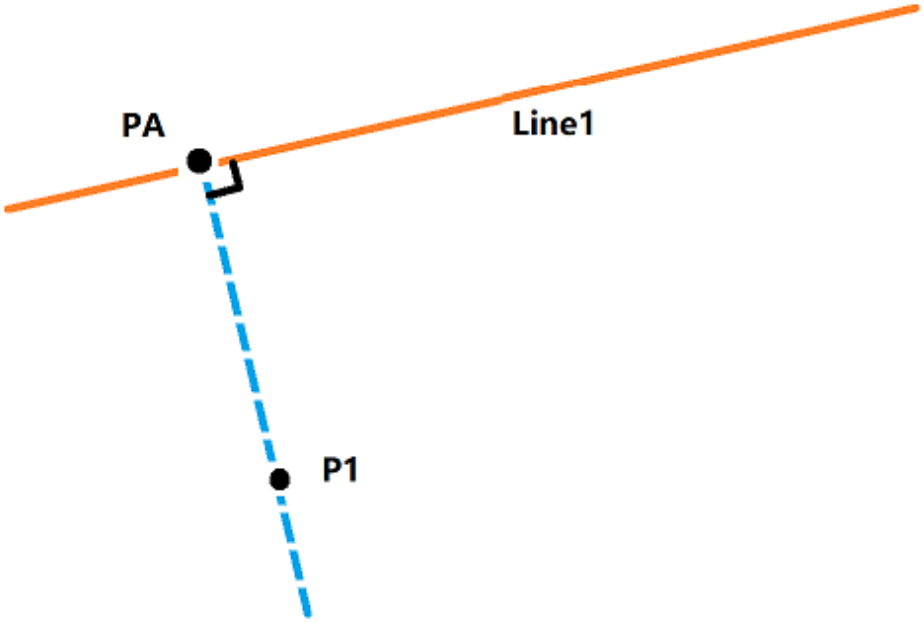
```
posA := GetVPointByLinePoint (Pos,line);
```

2、参数：

posA : CARTPOS/ROBOTCARTPOS //直线交点，必填

Pos: CARTPOS/ROBOTCARTPOS //空间上的点，必填

line: LINE //直线方程或 XAxis/YAxis/ZAxis，必填



3、功能描述：

GetVPointByLinePoint 语句用于获取点在直线上的投影点的 XYZ，若点在给定的直线上，则直接返回该点的 XYZ。

4、编写示例：

cp0 := GetVPointByLinePoint (cp1,line1)

返回 cp1 到直线 line1 的投影点，并将投影点的 XYZ 赋值给 cp0 的 XYZ。

3.4.15 GetCenterByPoints

1、语句格式：

```
posA := GetCenterByPoints (P1，P2，P3…P10);
```

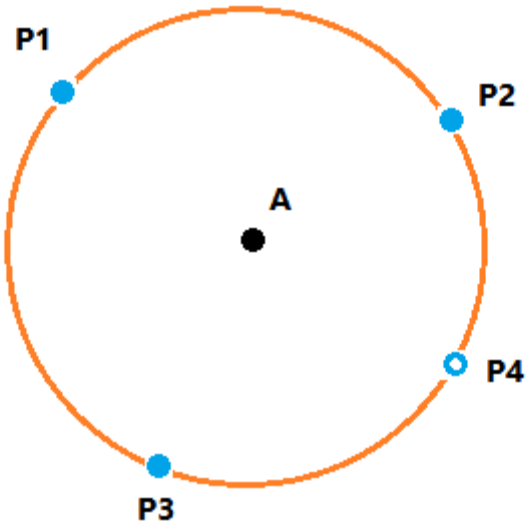
2、参数：

posA : CARTPOS/ROBOTCARTPOS //圆心，必填

P1: CARTPOS/ROBOTCARTPOS //平面上的点 1

…

P10: CARTPOS/ROBOTCARTPOS //平面上的点 10



3、功能描述：

GetCenterByPoints 语句用于获取平面上多点拟合出来的圆的圆心，平面上的位置点，需要提供至少 3 个，至多 10 个，若拟合圆的误差超过 10mm，会报错。

4、编写示例：

cp0 := GetCenterByPoints (cp1,cp2,cp3,cp4)

返回 cp1/cp2/cp3/cp4 拟合出来的圆的圆心，并将投影点的 XYZ 赋值给 cp0 的 XYZ。

3.4.16 GetOffsetPointByLine

1、语句格式：

```
posA := GetOffsetPointByLine (Pos，line，offset);
```

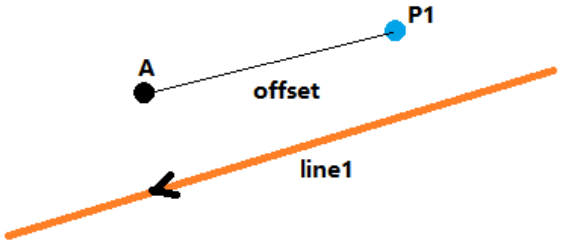
2、参数：

posA : CARTPOS/ROBOTCARTPOS //偏移点，必填

Pos: CARTPOS/ROBOTCARTPOS //空间上的点，必填

line: LINE //直线方程或 XAxis/YAxis/ZAxis，必填

offset:REAL//偏移距离，必填



3、功能描述：

GetOffsetPointByLine 语句用于获取点在直线（沿着直线方向）上偏移 offset 距离的位置点。

4、编写示例：

cp0 := GetOffsetPointByLine (cp1,line1,real0)

返回 cp1 沿着直线 line1 偏移 real0 的位置点，并将其 XYZ 赋值给 cp0 的 XYZ。

3.4.17 GetLineBy2Points

1、语句格式：

```
line := GetLineBy2Points (pos1,pos2);
```

2、参数：

line: LINE //直线方程，必填

pos1 : CARTPOS/ROBOTCARTPOS //空间上的任意一点，必填

pos2 : CARTPOS/ROBOTCARTPOS //空间上的任意一点，必填

3、功能描述：

GetLineBy2Points 语句获取通过给定的两个位置点的直线，若两个位置点相同，则报错，此外，获取的直线方程的向量是归一化后的值。

4、编写示例：

Line0 := GetLineBy2Points (cp1, cp2)

返回经过 cp1 和 cp2 的直线方程，并赋给 Line0。

3.4.18 GetLineByPointLine

1、语句格式：

```
line2 := GetLineByPointLine (Pos,line1);
```

2、参数：

Pos : CARTPOS/ROBOTCARTPOS //空间上的任意一点，必填

line1: LINE //已知直线的直线方程，必填

line2: LINE //返回的直线方程，必填

3、功能描述：

GetLineByPointLine 语句获取经过给定点，且与已知直线平行的直线，此外，获取的直线方程的向量是归一化后的值。

4、编写示例：

line2 := GetLineByPointLine (cp1, line1)

返回经过 cp1 且与直线 line1 平行的直线方程，并赋给 line2。

3.4.19 GetVLineByPointLine

1、语句格式：

```
line2 := GetVLineByPointLine (Pos,line1);
```

2、参数：

Pos : CARTPOS/ROBOTCARTPOS //空间上的任意一点，必填

line1: LINE //已知直线的直线方程，必填

line2: LINE //返回的直线方程，必填

3、功能描述：

GetVLineByPointLine 语句获取通过给定点，并且与已知直线垂直的直线方程，若点在直线上，则报错。此外，获取的直线方程的向量是归一化后的值。

4、编写示例：

line2 := GetVLineByPointLine (cp1, line1)

返回经过 cp1 且与直线 line1 垂直的直线方程，并赋给 line2。

3.4.20 GetVLineByPointPlane

1、语句格式：

```
line := GetVLineByPointPlane (Pos,plane);
```

2、参数：

line: LINE //返回的直线方程，必填

Pos : CARTPOS/ROBOTCARTPOS //空间上的任意一点，必填

plane: PLANE //平面方程或 XOY/YOZ/ZOX，必填

3、功能描述：

GetVLineByPointPlane 语句获取通过给定点且与已知平面垂直的直线，此外，获取的直线方程的向量是归一化后的值。

4、编写示例：

line := GetVLineByPointPlane (cp1, plane)

返回经过 cp1 且与 plane 平面垂直的直线方程，并赋给 line。

3.4.21 GetLineBy2Planes

1、语句格式：

```
line := GetLineBy2Planes (plane1,plane2);
```

2、参数：

line: LINE //返回的直线方程，必填

plane1: PLANE //平面方程或 XOY/YOZ/ZOX，必填

plane2: PLANE //平面方程或 XOY/YOZ/ZOX，必填

3、功能描述：

GetLineBy2Planes 语句获取两个平面的交线的直线方程，若两个平面平行，则报错，此外，获取的直线方程的向量是归一化后的值。

4、编写示例：

line0 := GetLineBy2Planes (plane1, plane2)

返回 plane1 和 plane2 两个平面的交线的直线方程，并赋给 line0。

3.4.22 GetPlaneByPointLine

1、语句格式：

```
plane:= GetPlaneByPointLine (Pos,line);
```

2、参数：

Pos : CARTPOS/ROBOTCARTPOS //空间上的任意一点，必填

line: LINE //已知直线的直线方程，必填

plane: PLANE //返回的平面方程，必填

3、功能描述：

GetPlaneByPointLine 语句获取点和直线确定的平面的平面方程，若点在直线上，则报错。此外，获取的平面方程的向量是归一化后的值。

4、编写示例：

plane0:= GetPlaneByPointLine (cp0, line0)

返回点 cp0 和直线 line0 确定的平面的平面方程，并赋给 plane0。

3.4.23 GetPlaneBy2Lines

1、语句格式：

```
plane:= GetPlaneBy2Lines (line1,line2);
```

2、参数：

line1: LINE //已知直线的直线方程，必填

line2: LINE //已知直线的直线方程，必填

plane: PLANE //返回的平面方程，必填

3、功能描述：

GetPlaneBy2Lines 语句获取经过两条直线的平面的平面方程，若两条直线共线，则报错，若两条直线异面，则返回过公垂线中点且与两条直线平行的平面。此外，获取的平面方程的向量是归一化后的值。

4、编写示例：

plane0:= GetPlaneBy2Lines (line0, line1)

返回 line0 和 line1 确定的平面的平面方程，并赋给 plane0。

3.4.24 FittingPlaneByPoints

1、语句格式：

```
plane:= FittingPlaneByPoints (P1，P2，P3…P10);
```

2、参数：

P1: CARTPOS/ROBOTCARTPOS //平面上的点 1

…

P10: CARTPOS/ROBOTCARTPOS //平面上的点 10

plane: PLANE //返回的平面方程，必填

说明：至少需要填写 3 个位置点。

3、功能描述：

FittingPlaneByPoints 语句获取平面上的多个点拟合出来的平面的平面方程，若拟合误差过大，即超过 10mm，则报错。此外，获取的平面方程的向量是归一化后的值。

4、编写示例：

plane0:= FittingPlaneByPoints (cp0, cp1,cp2,cp3)

返回 cp0/cp1/cp2/cp3 四个点拟合出来的平面的平面方程，并赋给 plane0。

3.6 程序流控制指令

3.6.1 IF ( )THEN… END_IF

1、语句格式：

```
IF(…)THEN
…
END_IF
```

2、参数：

无

3、功能描述：

该指令控制程序流进入条件分支。

4、编写示例：

```
IF(bool1==0)THEN
   PTP(ap1);
END_IF
PTP(ap2);
```

当 bool1 为 0 时，机器人运动到 ap1，否则直接运行 PTP(ap2)语句。

3.6.2 IF( ) THEN …ELSE… END_IF

1、语句格式：

```
IF(…)THEN
…
ELSE
…
END_IF
```

2、参数：

无

3、功能描述：

该指令控制程序流进入条件分支。

4、编写示例：

```
IF(bool1==0)THEN
   PTP(ap1);
ELSE
   PTP(ap2);
END_IF
```

当 bool1 为 0 时，机器人运动到 ap1，否则机器人运动到 ap2。

3.6.3 WHILE ( )DO… END_WHILE

1、语句格式：

```
WHILE(…)DO
…
END_WHILE
```

2、参数：

无

3、功能描述：

该指令用来在条件满足的情况下，执行一个指令序列

4、编写示例：

```
WHILE ( bool0)DO
   PTP(ap0);
   PTP(ap1);
 END_WHILE
```

当 bool0 为 TRUE 时，机器人执行两条 PTP 语句，否则不执行。

3.6.4 LOOP ( )DO… END_LOOP

1、语句格式：

```
LOOP(…)DO
…
END_LOOP
```

2、参数：

无

3、功能描述：

该指令用于对指令序列执行给定的次数。

4、编写示例：

```
LOOP(int0)DO
   PTP(ap0);
   PTP(ap1);
END_LOOP
```

循环执行两条 PTP 语句 int0 次。

3.6.5 LP

1、语句格式：

```
LP: variable
```

2、参数：

任何变量名

3、功能描述：

该指令用来定义跳转标签点，与 GOTO 语句配合使用。

3.6.6 GOTO

1、语句格式：

```
GOTO（variable）;
```

2、参数：

任何变量名

3、功能描述：

用来在程序的不同部分跳转

4、编写示例：

```
LP:int0
PTP(ap1);
PTP(ap2);
PTP(ap3);
GOTO (int0);
```

运行到 GOTO 行时跳转到 LP 处继续执行 LP 与 GOTO 之间的指令序列，该程序会一直循环运行 3 条 PTP 语句。

3.6.7 SWITCH...CASE

1、语句格式：

```
SWITCH(uint)
CASE 1:
…
CASE 2:
…
CASE 3,4,5:
…
END_SWITCH
```

2、参数：

SWITCH 的参数只能为无符号整形，即 Uint 型变量。

CASE 语句的参数只能为非负整数，或者由逗号连接的非负整数字符串。

3、功能描述：

根据 unit 的值来选择需要执行的语句，即根据 unit 的值选择要执行的 CASE 分支。

4、编写示例：

```
SWITCH(uint0)
CASE 0:
   PTP(ap0);
CASE 1:
   PTP(ap1);
CASE 4:
  PTP(ap2);
CASE 2,5,8:
  PTP(ap3);
END_SWITCH
```

当 unit0 为 0 时，机器人运行到 ap0 点；当 unit0 为 1 时，机器人运行到 ap1 点；当 unit0 为 4 时，机器人运行到 ap2 点；当 unit0 为 2 或 5 或 8 时，机器人运行到 ap3 点。

3.6.8 ElSIF

1、语句格式：

```
IF()
…
```

```
ELSIF()
…
ELSEIF()
…
ELSE
…
END_IF
```

2、使用方法：

该语句在示教器上作为一个整体使用，ELSIF 的个数没有限制，ELSIF 中的输入字符格式和 IF 中相同。

3、编程用例：

```
IF(int0>100)THEN
    PTP(ap1);
ELSEIF(int0>50)
    PTP(ap2);
ELSE
    PTP(ap3);
END_IF
```

当 int0 大于 100 时，机器人运行到 ap1 点；当 50<int0<=100 时，机器人运行到 ap2 点；否则机器人运行到 ap3 点。

3.6.9 ProgCall

1、语句格式：

```
ProgCall(var)
```

2、参数：

var : label　　//同一个工程下的程序名称或者 global 工程下的程序名

3、功能描述：

将同工程下(或全局)的程序作为子程序调用。

4、编写示例：

ProgCall(test);

ProgCall(_global.sr\test);

调用同工程下的（或全局）名为 test 的程序作为子程序。

3.7 输入输出设备 IO 指令

3.7.1 DIRead

1、语句格式：

```
BOOLVAR=DIRead(di,CONT);
```

2、参数：

di : DI　　/ /数字输入信号端口号；

CONT: 枚举　//可选参数

3、功能描述：

该指令用于从外部数据端口读取数字量输入，并将值返给左值变量 BOOLVAR。若不带 CONT 参数，DIRead 会打断预读，并且运行到该语句时读取 di 端口的值，并赋值给 BOOLVAR；

若带 CONT 参数，则在预读该语句时就会直接读取 di 端口的值，并继续往后预读，但是要运行到这一条语句时才将值赋给左值变量 BOOLVAR。

4、编写示例：

PTP(ap0);

bool0:=DIRead(di0);

运行到 DIRead 语句时，才会读取 di0 的值，并将 di0 端口的值赋给 bool0；

PTP(ap0);

bool0:=DIRead(di0,CONT);

连续模式下，运行 PTP 时，就会读取 di0 的值，等运行到 DIRead 语句时，将 di0 的值赋给 bool0；

3.7.2 DIWAIT

1、语句格式：

```
BOOLVAR=DIWAIT(di,val,time,CONT);
```

2、参数：

BOOLVAR：BOOL

di：DI

val：BOOL

time：UINT (毫秒)可选

CONT：枚举类型，可选

3、功能描述：

该指令用于在指定等待时间内等待触发信号并赋值，返回值为 bool 型。

若不带 CONT 参数，则运行到该 DIWAIT 语句的时候才开始判断 di 的值与 val 值是否相等，若相等，则给左值 BOOLVAR 赋值 TRUE，若不等，则等待 time 时间，在这段时间之内，等到了 val 信号，也给左值 BOOLVAR 赋值 TRUE，没有等到 val 信号，则给左值 BOOLVAR 赋值 FALSE,并往下运行，如果没有设置 time 参数，则会一直等待，直到等到 val 信号。

若带 CONT 参数，则在预读时就判断 di 的值是否与 val 值相等，若相等，则继续往下预读，运行到 DIWAIT 语句时给 BOOLVAR 赋值 TRUE，若不等，则会打断预读，并且运行到 DIWAIT 语句时再读取 di 的值，判断 di 的值与 val 的值，若相等，则给左值 BOOLVAR 赋值 TRUE，若不等，则等待 time 时间，在这段时间之内，等到了 val 信号，也给左值 BOOLVAR 赋值 TRUE，没有等到 val 信号，则给左值 BOOLVAR 赋值 FALSE,并往下运行，如果没有设置 time 参数，则会一直等待，直到等到 val 信号。

4、编写示例：

PTP(ap0);

bool0=DIWAIT(di0,val0,time0);

运行到 DIWAIT 语句时，在 time0 时间内，若从 di0 数字量输入端得到了 val 信号，就将 bool0 置 true，若没有，则等待 time0 时间后，给 bool0 置 false；

PTP(ap0);

bool0=DIWAIT(di0,val0);

运行到 DIWAIT 语句时，判断 di0 数字输入端的值与 val 的值，若相等，则给 bool0 置 TRUE,若不等，则一直等待，直到 di0 数字输入端的值与 val 相等；

PTP(ap0);

bool0=DIWAIT(di0,val0,time0,CONT);

运行 PTP 语句时，就判断 di0 的值与 val0 的值是否相等，若相等，继续往下预读，运行到 DIWAIT 时给 bool0 置 TRUE，否则，等程序运行到 DIWAIT 时，再开始判断，在 time0 时间内，若从 di0 数字量输入端得到了 val 信号，就将 bool0 置 true，若没有，则等待 time0 时间后，给 bool0 置 false；

PTP(ap0);

bool0=DIWAIT(di0,val0，NULL,CONT);

运行 PTP 语句时，就判断 di0 的值与 val0 的值是否相等，若相等，继续往下预读，运行到 DIWAIT 时给 bool0 置 TRUE，若不等，则等程序运行到 DIWAIT 语句时，再进行判断，一直等待，直到 di0 数字输入端得到 val 信号；

3.7.3 DIWaitPath

1、语句格式：

```
BOOLVAR=DIWaitPath(di,val,time);
```

2、参数：

BOOLVAR：BOOL

di：DI

val：BOOL

time：UINT (毫秒)可选

3、功能描述：

从 DIWaitPath 的上一条运动语句运行时开始判断 di，如果 di 的值等于 val，则继续后续语句的预读，等运行到 DIWaitPath 时将 BOOLVAR 置为 TRUE。反之则持续判断，当上一条运动运动语句执行完以后，再持续等待 time 时间（无 time 参数时，则无限等待），若等待到 di 的值等于 val，则 BOOLVAR 等于 TRUE；反之，则 BOOLVAR 等于 FALSE。

4、编写示例：

PTP(ap0);

PTP(ap1);

bool0=DIWaitPath(di0,val0,time0);

PTP(ap2);

开始运行 PTP(ap1)语句时，就开始读取 di0 的值，并判断 di0 的值是否与 val0 相等，若相等，则直接往下预读，并等到运行 DIWaitPath 语句时将 bool0 置为 TRUE，若不相等，则持续判断，直到 PTP(ap1)语句运行完，若运行 PTP(ap1)过程中 di0 的值一直与 val0 不等，则在运行 DIWaitPath 时再进行判断，且判断 time0 时间内

是否等待到 di0 与 val0 相等的情况，若等到，直接将 bool0 置 TRUE 并往下运行，若等不到，直接将 bool0 置为 FALSE 并往下运行。

PTP(ap0);

PTP(ap1);

bool0=DIWaitPath(di0,val0);

PTP(ap2);

开始运行 PTP(ap1)语句时，就开始读取 di0 的值，并判断 di0 的值是否与 val0 相等，若相等，则直接往下预读，并等到运行 DIWaitPath 语句时将 bool0 置为 TRUE，若不相等，则持续判断，直到 PTP(ap1)语句运行完，若运行 PTP(ap1)过程中 di0 的值一直与 val0 不等，则在运行 DIWaitPath 时再进行判断，直到等到 di0 的值与 val0 的值相等为止。

3.7.4 DOSet

1、语句格式：

```
DOSet(do,val,CONT);
```

2、参数：

do ： DO

val：BOOL

CONT: 枚举，可选参数

3、功能描述：

该指令用于给外部 DO 端口设置值；

若不带 CONT 参数，则运行到该语句时才会给 DO 端口设置值，但是不会中断语句的预读；

若带 CONT 参数，则在预读时就直接给 DO 端口设置值，即有可能还没有运行到该语句时就直接输出 DO 了；

4、编写示例：

PTP(ap0);

DOSet(do0,val0);

运行到 DOSet 语句时给 do0 输出 val0；

PTP(ap0);

DOSet(do0,val0,CONT);

连续模式下，运行 PTP 时就会直接给 do0 输出 val0；

3.7.5 AIRead

1、语句格式：

```
int:= AIRead(ai,CONT);
```

2、参数：

int: INT 类型变量

ai :AI 端口号

CONT：枚举类型，可选参数

3、功能描述：

从指定端口中读取对应端口的模拟量值，并赋值给左值。

若不带 CONT 参数，会中断预读，并且运行到 AIRead 语句时才会读取对应的模拟端口的值，并将值赋给左值变量 int；

若带 CONT 参数，则预读到 AIRead 语句时就会读取对应的模拟量端口的值，并继续往后预读，等运行到该语句时，再将值赋给左值变量 int；

4、编写示例：

PTP(ap0);

int0:=AIRead(ai0);

运行到 AIRead 语句时，读取 ai0 端口的值，并将该值赋给 int0；

PTP(ap0);

int0:=AIRead(ai0,CONT);

连续模式下，运行 PTP 时就会直接读取 ai0 的值，等运行到 AIRead 语句时再将值赋给 int0；

3.7.6 AIWaitGreater

1、语句格式：

```
bool:= AIWaitGreater (ai, val, time,CONT);
```

2、参数：

bool: BOOL

ai：AI

val：INT

time：UINT (毫秒)可选

CONT: 枚举类型，可选参数

3、功能描述：

监视指定端口的模拟量，直到该端口的值大于或等于设置的 val 值；

若不带 CONT 参数，则运行到 AIWaitGreater 语句时，才开始读取 ai 的值，并判断在 time 时间内，ai 的值是否大于或等于 val，若满足条件，则给 bool 赋值 TRUE，否则赋值 FALSE；

若带 CONT 参数，则在预读时就直接判断 ai 的值是否大于等于 val 的值，若满足条件，就直接往下预读，且等运行到 AIWaitGreater 时将 bool 赋值 TRUE，否则运行到 AIWaitGreater 语句时再进行判断，判断在 time 时间内，ai 的值是否大于或等于 val，若满足条件，则给 bool 赋值 TRUE，否则赋值 FALSE；

4、编写示例：

PTP(ap0);

bool0:= AIWaitGreater (ai0, val0, time0);

运行到 AIWaitGreater 语句时，在 time0 时间内，判断 ai0 的值是否大于等于 val0 的值，若满足条件，则给 bool0 赋值 TRUE，并直接往下运行，否则等待 time0 时间后，给 bool0 赋值 FALSE，再往下运行；

PTP(ap0);

bool0:= AIWaitGreater (ai0, val0);

运行到 AIWaitGreater 语句时，开始判断 ai0 的值是否大于等于 val0 的值，若满足条件，则给 bool0 赋值 TRUE，并往下运行，若不满足，则一直等待，直到满足为止；

PTP(ap0);

bool0:= AIWaitGreater (ai0, val0, time0,CONT);

连续模式下，运行 PTP 时就会直接读取 ai0 的值，并判断 ai0 是否大于等于 val0，若满足条件，则继续往下预读，等到运行 AIWaitGreater 时给 bool0 赋值 true，若不满足，则等程序运行到 AIWaitGreater 语句时再进行判断，判断在 time0 时间内，ai0 的值是否大于等于 val0 的值，若满足条件，则给 bool0 赋值 TRUE，并直接往下运行，否则等待 time0 时间后，给 bool0 赋值 FALSE，再往下运行；

PTP(ap0);

bool0:= AIWaitGreater (ai0, val0, NULL,CONT);

连续模式下，运行 PTP 时就会直接读取 ai0 的值，并判断 ai0 是否大于等于 val0，若满足条件，则继续往下预读，等到运行 AIWaitGreater 时给 bool0 赋值 true，若不满足，则等程序运行到 AIWaitGreater 语句时再进行判断，一直等待，直到条件满足为止；

3.7.7 AIWaitLess

1、语句格式：

```
bool:= AIWaitLess (ai, val, time,CONT);
```

2、参数：

bool: BOOL

ai：AI

val：INT

time：UINT (毫秒)可选

CONT: 枚举类型，可选参数

3、功能描述：

监视指定端口的模拟量，直到该端口的值小于或等于设置的 val 值；

若不带 CONT 参数，则运行到 AIWaitLess 语句时，才开始读取 ai 的值，并判断在 time 时间内，ai 的值是否小于或等于 val，若满足条件，则给 bool 赋值 TRUE，否则赋值 FALSE；

若带 CONT 参数，则在预读时就直接判断 ai 的值是否小于等于 val 的值，若满足条件，就继续往下预读，并且运行到该语句时给 bool 赋值 TRUE，否则运行到 AIWaitLess 语句时再进行判断，判断在 time 时间内，ai 的值是否小于或等于 val，若满足条件，则给 bool 赋值 TRUE，否则赋值 FALSE；

4、编写示例：

PTP(ap0);

bool0:= AIWaitLess (ai0, val0, time0);

运行到 AIWaitLess 语句时，在 time0 时间内，判断 ai0 的值是否小于等于 val0 的值，若满足条件，则给 bool0 赋值 TRUE，并直接往下运行，否则等待 time0 时间后，给 bool0 赋值 FALSE，再往下运行；

PTP(ap0);

bool0:= AIWaitLess (ai0, val0);

运行到 AIWaitLess 语句时，开始判断 ai0 的值是否小于等于 val0 的值，若满足条件，则给 bool0 赋值 TRUE，并往下运行，若不满足，则一直等待，直到满足为止；

PTP(ap0);

bool0:= AIWaitLess (ai0, val0, time0,CONT);

连续模式下，运行 PTP 时就会直接读取 ai0 的值，并判断 ai0 是否小于等于 val0，若满足条件，则继续往下预读，并且运行到 AIWaitLess 语句时给 bool0 赋值 true，若不满足，则等程序运行到 AIWaitLess 语句时再进行判断，判断在 time0 时间内，ai0 的值是否小于等于 val0 的值，若满足条件，则给 bool0 赋值 TRUE，并直接往下运行，否则等待 time0 时间后，给 bool0 赋值 FALSE，再往下运行；

PTP(ap0);

bool0:= AIWaitLess (ai0, val0, NULL,CONT);

连续模式下，运行 PTP 时就会直接读取 ai0 的值，并判断 ai0 是否小于等于 val0，若满足条件，则继续往下预读，并等运行到 AIWaitLess 语句时给 bool0 赋值 true，若不满足，则等程序运行到 AIWaitLess 语句时再进行判断，一直等待，直到条件满足为止；

3.7.8 AOSet

1、语句格式：

```
AOSet(ao,int,CONT);
```

2、参数：

ao: AO

int: INT

CONT: 枚举类型，可选参数

3、功能描述：

给 ao 端口设置值，即将 ao 对应端口的值设为 int。

若不带 CONT 参数，则运行到该语句时，将 ao 对应端口的值置为 int，但是不中断程序语句的预读；

若带 CONT，则在预读时就直接将 ao 对应端口的值置为 int；

4、编写示例：

PTP(ap0);

AOSet (ao0,int0);

运行到 AOSet 语句时，将 ao0 端口的值置为 int0；

PTP(ap0);

AOSet (ao0,int0,CONT);

连续模式下，运动 PTP 语句时就会将 ao0 端口的值置为 int0；

3.7.9 AOSyncOn

1、语句格式：

AOSyncOn(ao,vel0,vel1,int0,int1);

2、参数：

ao: AO

vel0: REAL

vel1: REAL

int0: INT

int1: INT

3、功能描述：

ao 的值随着机器人的运行速度变化而变化，即 ao 的值与当前机器人的速度产生一个线性关系，(vel0, int0)、(vel1,int1)分别是线性关系上的两个对应点，也就是说 ao0=(int1-int0)/(vel1-vel0)*v，其中 v 是机器人末端的当前移动速度。

4、编写示例：

AOSyncOn (ao0,vel0,vel1,int0,int1);

PTP(ap0);

PTP(ap1);

运行 PTP 期间，ao0 的值会随着机器人的速度变化而线性变化，线性关系由

(vel0, int0)、(vel1,int1)两个点决定，即 ao0=(int1-int0)/(vel1-vel0)*v，其中 v 是机器人末端的当前移动速度。

3.7.10 AOSyncOff

1、语句格式：

```
AOSyncOff(ao);
```

2、参数：

ao: AO

3、功能描述：

将 ao 与速度的线性关系关闭掉，与 AOSyncOn 配合使用。

4、编写示例：

```
AOSyncOn (ao0,vel0,vel1,int0,int1);
PTP(ap0);
PTP(ap1);
AOSyncOff(ao0);
PTP(ap2);
```

运行 PTP(ap0)和 PTP(ap1)期间，ao0 的值会随着机器人的当前速度变化，但是运行 PTP(ap2)期间，ao0 的值与机器人的速度无关。

3.7.11 GDIRead

1、语句格式：

```
uint:= GDIRead(gdi,CONT);
```

2、参数：

uint: UINT

gdi: GDI　　/ /端口组

CONT：枚举类型，可选参数

3、功能描述：

从指定端口组中读取对应的数字量值，并把这些端口的值组合后赋值给左值。如 gdi 设定端口 3—7，即读取 3、4、5、6、7 号端口的 di 值，7 号端口为最高位；那么如果此时 3、6、7 号端口为 1，其它为 0，则此时左值大小为 $25=2^4+2^3+0+0+1$ 二进制表示为 11001。

端口号	7	6	5	4	3	.
对应值	1	1	0	0	1	

组 IO 的端口号需要是连续的，但是个数可以变化，从 2---16；对应的值范围分别是 0---3；0---65535。

若没有 CONT 参数，则在运行该语句时才会读取 gdi 的值，并将读取的值赋给左值变量；若有 CONT 参数，则在预读时就读取 gdi 的值，并在运行到该语句时赋给左值变量。

4、编写示例：

PTP(ap0);

uint0:= GDIRead(gdi0); //gdi0 的端口号为 1-3

运行到 GDIRead 语句时，若读取到端口号 1 和 2 为 true，端口 3 为 false，则 uint0 的值为 3；

PTP(ap0);

uint0:= GDIRead(gdi0,CONT); //gdi0 的端口号为 1-3

连续模式下，运动 PTP 语句时，就读取端口号 1-3 的值，若读取到端口号 1 和 2 为 true，端口 3 为 false，则在运行 GDIRead 语句时，将 unit0 的值置为 3；

3.7.12 GDOSet

1、语句格式：

```
GDOSet(gdo,uint,CONT);
```

2、参数：

gdo: GDO / /端口组

uint: UINT

CONT: 枚举类型，可选参数

3、功能描述：

把指定的数值设置到某个端口组中，如 gdo 设定端口 3—7，第二个参数值为 6，其对应的二进制是 00110。那么将 3 号口置为 0；4 号口置为 1,5 号口置为 1；6、7 号端口置为 0。

端口号	7	6	5	4	3	.
对应值	0	0	1	1	0	

若不带 CONT 参数，则在运行 GDOSet 语句时才给 GDO 中的各端口号设值；

若带 CONT 参数，则在预读 GDOSet 语句时就给 GDO 中的各端口号设值；

4、编写示例：

PTP(ap0);

GDOSet (gdo0,uint0);　//gdo0 的端口号为 1-3，unit0 为 3

运行到 GDOSet 语句时，将 DO 端口号 1 和 2 置为 true，端口 3 置为 false；

PTP(ap0);

GDOSet (gdo0,uint0,CONT);　//gdo0 的端口号为 1-3，unit0 为 3

连续模式下，运动 PTP 语句时，就将 DO1 和 DO2 置为 TRUE，将 DO3 置为 FALSE；

3.7.13 DOPulse

1、语句格式：

```
DOPulse(do,bool,pulsetime,CONT);
```

2、参数：

do: DO

bool: BOOL

pulsetime: INT

CONT: 枚举类型，可选参数

3、功能描述：

给 do 输出一个持续 pulsetime 时间的脉冲，脉冲信号值由 bool 参数确定。若 bool 为 true，则给 do 输出一个持续时间为 pulsetime 的高电平脉冲，若为 false，则给 do 输出一个长达 pulsetime 时间的低电平脉冲。

若不带 CONT 参数，则在运行 DOPulse 语句时才输出 DO；

若带 CONT 参数，则在预读 DOPulse 语句时就给 DO 输出一个脉冲；

4、编写示例：

PTP(ap0);

DOPulse (do0,bool0,time0);　//bool0 为 TRUE,time0 为 1000

运行到 DOPulse 语句时，给 do0 输出一个高电平脉冲，高电平持续时间为 1s；

PTP(ap0);

DOPulse (do0,bool0,time0,CONT);　//bool0 为 TRUE,time0 为 1000

连续模式下，运动 PTP 语句时，就给 do0 输出一个高电平的脉冲，高电平持

续时间为 1s；

3.7.14 DOSetSyncTime

1、语句格式：

```
DOSetSyncTime(do, boolval, start/end, offsettime);
```

2、参数：

do: DO

boolvar: BOOL

start/end： 枚举类型

offsettime：INT

3、功能描述：

与上一条运动语句同步输出 DO，在距离 start/end 点 offsettime 的时刻，将 boolval 输出到 do 上。若选择 start 模式，offsettime 设置的时间比 0 小时，在开始运行时刻输出 do，offsettime 设置的时间比上面一条运动语句的运行时间长时，在运动语句结束时输出 do；若选择 end 模式，offsettime 设置的过小，导致超过起点时刻时，在运行开始时输出，offsettime 设置的大于 0 时，在上一条运动语句结束时输出 do。

说明：在单步模式下，若该语句前面有运动语句，则运行该语句时直接输出 do。

4、编写示例：

PTP(ap0);

PTP(ap1);

DOSetSyncTime(do0, bool0, start, time0);//bool0 为 TRUE,time0 为 1000

连续模式下运行时，在 PTP(ap1)运行 1s 后，将 do0 的值置为 TRUE；

PTP(ap0);

PTP(ap1);

DOSetSyncTime(do0, bool0, end, time0);//bool0 为 TRUE,time0 为-1000

连续模式下运行时，在 PTP(ap1)语句结束前 1 秒，将 do0 的值置为 TRUE；

3.7.15 DOSetSyncPath

1、语句格式：

```
DOSetSyncPath(do, boolval, start/end, length);
```

2、参数：

do: DO

boolvar: BOOL

start/end: 枚举类型

length：REAL

3、功能描述：

与上一条运动语句同步输出 DO，在距离 start/end 点 length 距离时，将 boolval 输出到 do 上。若选择 start 模式，lenght 设置的路径长度小于 0 时，在上一条运动语句开始运行时刻输出 do，lenght 设置长度比上面一条运动语句的轨迹长时，在运动语句结束时输出 do；若选择 end 模式，lenght 设置的过小，导致超过运动语句的起点，在运行开始时输出 do，lenght 设置的大于 0 时，在上一条运动语句结束时输出 do。

说明：在单步模式下，若该语句前面有运动语句，则运行该语句时直接输出 do。且前面运动语句是 PTP 时，length 的单位是度，按照运行的最长的一个轴来计算，前面运动语句是 LIN 或 CIRC 时，length 的单位是 mm，按照 TCP 末端的移动距离来计算。

4、编写示例：

PTP(ap0); //ap0=0,0,0,0,0,0

PTP(ap1); //ap1=100,20,10,0,0,0

DOSetSyncPath(do0, bool0, start, lenght0);//bool0 为 TRUE, lenght0 为 10°

连续模式下运行时，机器人的 1 轴运行 10° 后，将 do0 的值置为 TRUE；

PTP(ap0); //ap0=0,0,0,0,0,0

PTP(ap1); //ap1=100,20,10,0,0,0

DOSetSyncPath(do0, bool0, end, lenght0);//bool0 为 TRUE, lenght0 为-10°

连续模式下运行时，机器人的 1 轴运行到 90 度时，将 do0 的值置为 TRUE；

3.7.16 DOPulseSyncTime

1、语句格式：

```
DOPulseSyncTime(do, boolval, pulsetime, start/end, offsettime);
```

2、参数：

do: DO

boolvar: BOOL

pulsetime: INT

start/end： 枚举类型

offsettime：INT

3、功能描述：

与上一条运动语句同步输出脉冲，在距离 start/end 点 offsettime 的时刻，给 do 输出一个宽为 pulsetime，值为 boolval 的脉冲。若选择 start 模式，offsettime 设置的时间比 0 小时，在开始运行时刻输出脉冲，offsettime 设置的时间比上面一条运动语句的运行时间长时，在运动语句结束时输出脉冲；若选择 end 模式，offsettime 设置的过小，导致超过起点时刻时，在开始运行时输出脉冲，offsettime 设置的大于 0 时，在上一条运动语句结束时输出脉冲。

说明：在单步模式下，若该语句前面有运动语句，则运行该语句时直接输出 do。

4、编写示例：

PTP(ap0);

PTP(ap1);

DOPulseSyncTime (do0, bool0,pulsetime0, start, time0);

//bool0 为 TRUE, pulsetime0 为 500，time0 为 1000

连续模式下运行时，在 PTP(ap1)运行 1s 后，给 do0 输出一个宽为 500ms 的高电平脉冲；

PTP(ap0);

PTP(ap1);

DOPulseSyncTime (do0, bool0, pulsetime0, end, time0);

//bool0 为 TRUE, pulsetime0 为 500，time0 为-1000

连续模式下运行时，在 PTP(ap1)语句结束前 1 秒，给 do0 输出一个宽为 500ms 的高电平脉冲；

3.7.17 DOPulseSyncPath

1、语句格式：

DOPulseSyncPath (do, boolvar, pulsetime, start/end, length);

2、参数：

do: DO

boolvar: BOOL

pulsetime: INT

start/end: 枚举类型

length：REAL

3、功能描述：

与上一条运动语句同步输出脉冲，在距离 start/end 点 length 距离时，给 do 输出一个宽为 pulsetime，值为 boolval 的脉冲。若选择 start 模式，lenght 设置的路径长度小于 0 时，在上一条运动语句开始运行时刻输出脉冲，lenght 设置长度比上面一条运动语句的轨迹长时，在运动语句结束时输出脉冲；若选择 end 模式，lenght 设置的过小，导致超过运行的起点，在运行开始时输出脉冲，lenght 设置的大于 0 时，在上一条运动语句结束时输出脉冲。

说明：在单步模式下，若该语句前面有运动语句，则运行该语句时直接输出 do。且前面运动语句是 PTP 时，length 的单位是度，按照运行的最长的一个轴来计算，前面运动语句是 LIN 或 CIRC 时，length 的单位是 mm，按照 TCP 末端的移动距离来计算。

4、编写示例：

PTP(ap0); //ap0=0,0,0,0,0,0

PTP(ap1); //ap1=100,20,10,0,0,0

DOPulseSyncPath(do0, bool0, pulsetime0,start, lenght0);

//bool0 为 TRUE, pulsetime0 为 800，lenght0 为 10°

连续模式下运行时，机器人的 1 轴运行 10° 后，输出一个宽为 800ms 的高电平脉冲；

PTP(ap0); //ap0=0,0,0,0,0,0

PTP(ap1); //ap1=100,20,10,0,0,0

DOPulseSyncPath (do0, bool0, pulsetime0,end, lenght0);

//bool0 为 TRUE, pulsetime0 为 800，lenght0 为-10°

连续模式下运行时，机器人的 1 轴运行到 90 度时，输出一个宽为 800ms 的高电平脉冲；

3.7.18 BOOLEXTRead

1、语句格式：

```
BOOLVAR= BOOLEXTRead(ext);
```

2、参数：

ext : BOOLEXT　　//结构体，包含数字输入信号端口号、值；

3、功能描述：

该指令用于读取在 PLC 已经定义的变量，返回值为 bool 型；

4、编写示例：

bool0= BOOLEXTRead(ext0);

将外部数据结构体 ext0 中的数据赋给变量 bool0。

3.7.19 BOOLEXTSet

1、语句格式：

```
BOOLEXTSet(ext, val);
```

2、参数：

ext : BOOLEXT　　//结构体，包含数字输入信号端口号、值

val：BOOL

3、功能描述：

该指令设置在 PLC 已经定义的变量

4、编写示例：

BOOLEXTSet(ext0, val0);

将 val0 的值赋给外部数据结构体 ext0。

3.7.20 DINTEXTRead

1、语句格式：

```
DINTVAR= DINTEXTRead(ext);
```

2、参数：

ext : BOOLEXT　　//结构体，包含数字输入信号端口号、值

3、功能描述：

该指令用于从 PLC 读取已经定义的变量，返回值为 DINT 型

4、编写示例：

int0=DINTEXTRead(ext0);

将 ext0 中的数据赋给变量 int0。

3.7.21 DINTEXTSet

1、语句格式

```
DINTEXTSet(ext, val);
```

2、参数：

ext：BOOLEXT　　//结构体，包含数字输入信号端口号、值

val：DINT(LINT)

3、功能描述：

该指令设置在 PLC 已经定义的变量的值，其中 val0 为 DINT 型；

4、编写示例：

DINTEXTSet(ext0, val0);

将 val0 中的数据赋给 ext0。

3.7.22 REALEXTRead

1、语句格式：

```
REALVAR= REALEXTRead(ext);
```

2、参数：

ext : BOOLEXT　　/ /结构体，包含数字输入信号端口号、值；

3、功能描述：

该指令用于读取在 PLC 已经定义的变量，返回值为 REAL 型；

4、编写示例：

real0= REALEXTRead(ext0);

将从 ext0 中读取的数据赋给 real0。

3.7.23 REALEXTSet

1、语句格式：

```
REALEXTSet(ext, val);
```

2、参数：

ext : BOOLEXT　　//结构体，包含数字输入信号端口号、值

val：REAL

3、功能描述：

该指令设置在 PLC 已经定义的变量的值；

4、编写示例：

REALEXTSet(ext0, val0);

将 val0 的值赋给 ext0。

3.7.24 DWORDEXTRead

1、语句格式：

```
DWORDVAR= DWORDEXTRead(ext);
```

2、参数：

ext : BOOLEXT　　//结构体，包含数字输入信号端口号、值；

3、功能描述：

该指令用于读取在 PLC 已经定义的变量，返回值类型为 DWORD；

4、编写示例：

int0= DWORDEXTRead(ext0);

读取 ext0 中的数据并赋给变量 int0。

3.7.25 DWORDEXTSet

1、语句格式：

```
DWORDEXTSet(ext, val);
```

2、参数：

ext : BOOLEXT　　//结构体，包含数字输入信号端口号、值

val：DWORD（ULINT）

3、功能描述：

该指令用于设置在 PLC 已经定义的变量

4、编写示例：

DWORDEXTSet(ext0, val0);

将变量 val0 的值赋给 ext0。

3.7.26 AXISPOSEXTSet

1、语句格式：

```
AXISPOSEXTSet(port, axispos);
```

2、参数：

port: 枚举类型，0-31 / /端口号

axispos：AXISPOS //关节位置点

3、功能描述：

该指令与 codesys 中的 Step_Robot_GetAxisPosFromHMI 配合使用，可实现将示教器端的关节位置点读到 codesys 中。

4、编写示例：

AXISPOSEXTSet(1,ap0);

将 ap0 的位置值通过 1 号端口给到 codesys 端。

3.7.27 CARTPOSEXTSet

1、语句格式：

```
CARTPOSEXTSet(port, cartpos);
```

2、参数：

port: 枚举类型，0-127 / /端口号

cartpos：CARTPOS //笛卡尔位置点

3、功能描述：

该指令与 codesys 中的 Step_Robot_GetCartPosFromHMI 配合使用，可实现将示教器端的笛卡尔位置点读到 codesys 中。

4、编写示例：

CARTPOSEXTSet (2,cp0);

将 cp0 的位置值通过 2 号端口给到 codesys 端。

3.7.28 ROBOTAXISPOSEXTSet

1、语句格式：

```
ROBOTAXISPOSEXTSet(port, robotaxispos);
```

2、参数：

port: 枚举类型，0-31 / /端口号

robotaxispos：ROBOTAXISPOS //机器人关节位置点

3、功能描述：

该指令与 codesys 中的 Step_Robot_GetRobotAxisPosFromHMI 配合使用，可实现将示教器端的机器人位置点读到 codesys 中。

4、编写示例：

ROBOTAXISPOSEXTSet(3,rap1);

将 rap1 的位置值通过 3 号端口给到 codesys 端。

3.7.29 ROBOTCARTPOSEXTSet

1、语句格式：

```
ROBOTCARTPOSEXTSet(port, robotcartpos);
```

2、参数：

port: 枚举类型，0-127 / /端口号

robotcartpos：ROBOTCARTPOS //机器人笛卡尔位置点

3、功能描述：

该指令与 codesys 中的 Step_Robot_GetRobotCartPosFromHMI 配合使用，可实现将示教器端的机器人笛卡尔位置点读到 codesys 中。

4、编写示例：

ROBOTCARTPOSEXTSet(4,rcp1);

将 rcp1 的位置值通过 4 号端口给到 codesys 端。

3.7.30 AXISPOSEXTRead

1、语句格式：

```
axispos:=AXISPOSEXTRead (port);
```

2、参数：

axispos：AXISPOS //关节位置点

port: 枚举类型，0-31 / /端口号

3、功能描述：

该指令与 codesys 中的 Step_Robot_SetAxisPos 配合使用，可实现将 codesys 端的关节位置点读到示教器的位置变量中。

4、编写示例：

ap0:=AXISPOSEXTRead (1);

将 codesys 里 1 号端口的关节位置点传递给示教器变量 ap0.

3.7.31 CARTPOSEXTRead

1、语句格式：

```
cartpos:=CARTPOSEXTRead (port);
```

2、参数：

port: 枚举类型，0-127 / /端口号

cartpos：CARTPOS //笛卡尔位置点

3、功能描述：

该指令与 codesys 中的 Step_Robot_SetCartPos 配合使用，可实现将 codesys 里的笛卡尔位置点传递给示教器上的位置变量中。

4、编写示例：

cp0:=CARTPOSEXTRead (2);

将 codesys 里 2 号端口的笛卡尔位置传递给示教器上的 cp0 变量。

3.7.32 ROBOTAXISPOSEXTRead

1、语句格式：

```
robotaxispos:=ROBOTAXISPOSEXTRead (port);
```

2、参数：

port: 枚举类型，0-31 / /端口号

robotaxispos：ROBOTAXISPOS //机器人关节位置点

3、功能描述：

该指令与 codesys 中的 Step_Robot_SetRobotAxisPos 配合使用，可实现将 codesys 端的机器人位置点传递给示教器的位置变量中。

4、编写示例：

rap1：=ROBOTAXISPOSEXTRead (3);

将 codesys 中 3 号端口的值传递给示教器变量 rap1。

3.7.33 ROBOTCARTPOSEXTRead

1、语句格式：

```
robotcartpos:=ROBOTCARTPOSEXTRead (port);
```

2、参数：

port: 枚举类型，0-127 / /端口号

robotcartpos：ROBOTCARTPOS　　//机器人笛卡尔位置点

3、功能描述：

该指令与 codesys 中的 Step_Robot_SetRobotCartPos 配合使用，可实现将 codesys 里的机器人笛卡尔位置传递给示教器的位置变量。

4、编写示例：

rcp1：=ROBOTCARTPOSEXTRead (4);

将 codesys 里的 4 号端口的机器人笛卡尔位置传递给示教器位置变量 rcp1。

3.8 时钟指令

3.8.1 ClkStart

1、语句格式:

```
ClkStart(clock);
```

2、参数：

clock 时钟变量，没有值

3、功能描述：

ClkStart 用于启动一个计时器。

4、编写示例:

先定义一个 Clock 类型变量 ck1。Ck1 本质是一个基本类型的变量，其值不会使用。

ClkStart(ck1);

运行到 ClkStart(ck1)语句时会启动计时。

3.8.2 ClkStop

1、语句格式:

```
ClkStop (clock);
```

2、参数：

clock 时钟变量，没有值

3、功能描述：

ClkStop 用于计时器停止计时。

4、编写示例:

先定义一个 Clock 类型变量 ck1。Ck1 本质是一个基本类型的变量，其值不会使用。

```
ClkStart(ck1);
……
ClkStop(ck1);
```

运行到 ClkStop 语句时会停止计时。

3.8.3 ClkReset

1、语句格式:

```
ClkReset (clock);
```

2、参数：

clock 时钟变量，没有值

3、功能描述：

ClkReset 用于重置一个计时器，即将定时器的值置为 0。

4、编写示例:

先定义一个 Clock 类型变量 ck1。Ck1 本质是一个基本类型的变量，其值不会使用。

```
ClkStart(ck1);
……
ClkStop(ck1);
ClkReset(ck1);
```

运行到 ClkReset 语句时会重置 ck1。

3.8.4 ClkRead

1、语句格式:

```
luint:=ClkRead (clock);
```

2、参数：

clock 时钟变量，没有值

3、功能描述：

ClkRead 用于读取时钟变量所统计的所有时间间隔，可以是一段也可以是多段。

4、编写示例：

先定义一个 Clock 类型变量 ck1。Ck1 本质是一个基本类型的变量，其值不会使用。

```
ClkStart(ck1);
……
ClkStop(ck1);
ulint0:=ClkRead(ck1);
```

运行到 ClkRead 语句时，ulint0 记录了 ClkStart 和 ClkStop 之间的时间间隔。

3.9 中断指令

3.9.1 Connect

1、语句格式：

```
Connect(intnum0,program1);
```

2、参数：

intnum0：INTNUM（中断类型变量）

program1：程序名；

3、功能描述：

用于将中断变量和中断程序连接起来。

4、编写示例：

```
Connect(Intnum0,programName);
ISignal(move_open,1,Intnum0);
```

建立中断变量 Intnum0 和程序 programName 之间的映射关系。

3.9.2 ISignal

1、语句格式：

```
ISignal(move_open,1,Intnum0);
```

2、参数：

move_open：DI　　//被监视的输入信号

TRUE(FALSE):bool 类型　//用于判断是上升沿（TRUE 代表监视上升沿，FALSE 监视下降沿）

Intnum0：INTNUM（中断类型变量）//用于查找关联的中断程序

Type：枚举（单次，循环，安全）（可选参数）英文名称：Single，Cycle，Safe//此参数可选，不选则默认使用循环中断类型

3、功能描述：

绑定中断变量和 DI 信号（数字量输入信号）。

4、编写示例：

Connect(Intnum0,programName);

ISignal(move_open,1,Intnum0);

建立中断变量 Intnum0 和程序 programName 之间的映射关系。当 move_open 信号变成上升沿时，执行中断程序 programName。

3.9.3 StorePath

1、语句格式：

```
StorePath();
```

2、参数：

无

3、功能描述：

存储中断前机器人内部路径信息。

4、编写示例：

StorePath();//存储当前机器人内部信息（工具，坐标系）

GetCurrentRobotPos(pos,tool,ref);//获取机器人末端位置;

3.9.4 RestorePath

1、语句格式：

```
RestorePath();
```

2、参数：

无

3、功能描述：

恢复 StorePath 存储的路径信息，不允许单独使用。

4、编写示例：

StorePath();//存储当前机器人内部信息（工具，坐标系）

GetCurrentRobotPos(pos,tool,ref);//获取机器人末端位置;

3.9.5 IDelete

1、语句格式：

```
IDelete(intnum);
```

2、参数：

intnum：INTNUM（中断变量名）

3、功能描述：

IDelete 执行后，参数中的对应的中断就不会再响应。如果被删除的中断已经在待执行队列中，则不会同步删除。

4、编写示例：

IDelete(intnum);

3.9.6 ISleep

1、语句格式：

```
ISleep(intnum);
```

2、参数：

intnum：INTNUM（中断变量名）

3、功能描述：

ISleep 执行后，参数中的对应的中断就不会再响应，直到 IWatch 指令解除休眠。已经在待执行队列的中断不受 ISleep 影响。ISleep 不会删除已有中断。

4、编写示例：

ISleep(intnum);

...

IWatch(intnum);

3.9.7 IWatch

1、语句格式：

```
IWatch(intnum);
```

2、参数：

intnum：INTNUM（中断变量名）

3、功能描述：

IWatch 执行后，参数中对应的中断会被唤醒。该中断会再次置于后台监控。

4、编写示例：

ISleep(intnum);

...

IWatch(intnum);

3.9.8 IDisable

1、语句格式：

```
IDisable();
```

2、参数：

无

3、功能描述：

该指令用来禁用所有中断的执行。该指令会将 IDisable 指令生效期间出现的中断列入等待队列。再次允许中断时（执行 IEnable），立即开始响应中断，并在队列中以“先进先出”的顺序执行。

4、编写示例：

IDisable();

...

IEnable();

3.9.9 IEnable

1、语句格式：

```
IEnable();
```

2、参数：

无

3、功能描述：

该指令用来启用所有被禁用的中断及之后到来的中断。该指令会将 IDisable 指令生效期间出现的中断状态变成可执行，并以“先进先出”的顺序执行。IEnable 指令执行之后的中断也依次插入队列排序执行。

4、编写示例：

IDisable();

...

IEnable();

3.10 码垛指令

3.10.1 PalletEntryPoint

1、语句格式：

PalletEntryPoint (pallet0,cp0,dyn0,olr0,tool0,ref0);

2、参数：

pallet：PALLETDATA（码垛变量）

cp0：CARTPOS　（笛卡尔位置点）

dyn0：DYNAMIC　（动态特性）

olr0：　OVERLAPREL（圆滑百分比）

tool0: TOOL(工具)

ref0：CARTREF(参考坐标系)

3、功能描述：

用来设定码垛进入点，进入点会随着垛层的高度升高或者降低。

4、编写示例：

PalletEntryPoint (pallet0,cp0,dyn0,olr0);

设定码垛进入点，PalletEntryPoint 语句要放在调用生成的码垛程序之前。示教进入点时需要选择工具和坐标系，此工具和坐标系要与托盘使用的工具和坐标系一致，否则第一个进入点会有偏差

3.10.2 UpdatePalletizing

1、语句格式：

```
UpdatePalletizing (pallet0);
```

2、参数：

pallet：PALLETDATA（码垛变量）

3、功能描述：

更新指定码垛的状态。

4、编写示例：

UpdatePalletizing (pallet0);

ReadPalletizing(pallet0);

UpdatePalletizing 语句是更新码垛的语句，在码垛码完一个产品后，需要更新，即需要放在码垛子函数调用语句的后面，且 UpdatePalletizing 语句后面必须紧接着添加一条 ReadPalletizing 语句。

3.10.3 SetPalletizing

1、语句格式：

```
SetPalletizing (pallet0,uint0,uint1,uint2);
```

2、参数：

pallet：PALLETDATA（码垛变量）

uint0：UINT (设置当前层数)

uint1：UINT (设置当前层产品数)

uint2：UINT（设置任务量）

3、功能描述：

用来设定码垛的层计数、层内的计数以及循环次数，同时可以初始化码垛变量。当调用码垛子程序之前，必须用此语句初始化码垛变量。

4、编写示例：

SetPalletizing (pallet0,uint0,uint1,uint2);

在使用码垛变量前，即码垛程序的开头，需要使用 SetPalletizing 语句初始化码垛变量。

3.10.4 ReadPalletizing

1、语句格式：

```
ReadPalletizing (pallet0,uint0,uint1,uint2,bool0,bool1);
```

2、参数：

pallet：PALLETDATA（码垛变量）

uint0：UINT (层数)

uint1：UINT (产品数)

uint2：UINT（任务量）

bool0：BOOL（是否满）

bool1：BOOL（是否完成）

3、功能描述：

用来读取码垛计数和状态，包括当前垛进行到了哪一层，哪一个，循环了多少次，是否满，是否完成。

4、编写示例：

ReadPalletizing (pallet0,uint0,uint1,uint2,bool0,bool1);

3.10.5 PLin

1、语句格式：

```
PLin(pcp0,dyn0,olr0);
```

2、参数：

pcp0：PCARTPOS（码垛专用类型位置点）

dyn0：DYNAMIC （动态特性）

olr0： OVERLAPREL（圆滑百分比）

3、功能描述：

码垛专用点到点运动语句。

4、编写示例：

PLin(pcp0,dyn0,olr0);

3.10.6 PPTP

1、语句格式：

```
PPTP(pcp0,dyn0,olr0);
```

2、参数：

pcp0：PCARTPOS（码垛专用类型位置点）

dyn0：DYNAMIC　（动态特性）

olr0：　OVERLAPREL（圆滑百分比）

3、功能描述：

码垛专用直线运动语句。

4、编写示例：

PPTP(pcp0,dyn0,olr0);

技术支持

◆ 技术服务

九众九机器人有限公司乐于提供有关机器运行及操作的信息，并可帮助您排除故障和提供详细咨询，如果您的机器人生产过程中出现故障，可立即联系我们的服务机构，并尽可能提供以下信息：

- ✧ 机器人型号及序列号
- ✧ 控制系统型号及序列号
- ✧ 控制系统系统版本号
- ✧ 额外的软件功能包（可选）
- ✧ 现有的应用程序
- ✧ 其他附加装置（变位机、导轨等，可选）
- ✧ 问题描述、故障持续时间及频率等

◆ 联系方式

九众九机器人有限公司

地址：江苏省无锡市滨湖区胡埭工业园银杏路6号

电话：400-828-6378

传真：0510-83899219