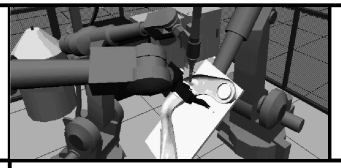
IRC5 RAPID

ABB Robot RAPID Reference manual



动动控制指令

Accset

Velset

ConfJ

ConflL

SingArea

PathResol

SoftAct

SoftDeact

运动控制指令-AccSet

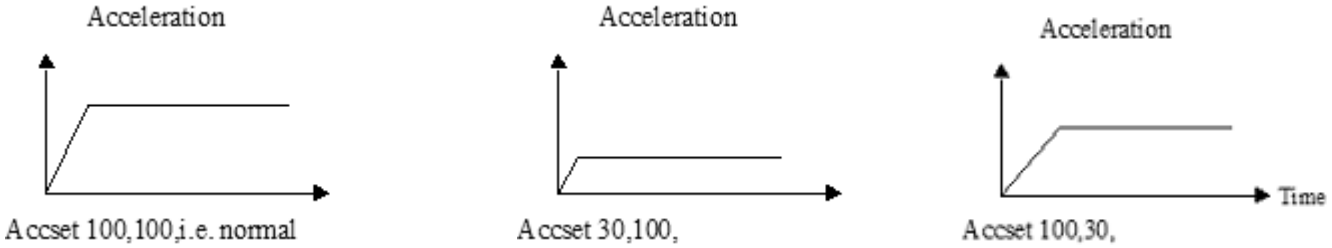
AccSet Acc,Ramp;

Acc: 机器人加速度百分比（num)
Ramp: 机器人加速度坡度（num)

应用：
当机器人运行速度改变时，对所产生的相应加速度进行限制，使机器人高速运行时更平缓，但会延长循环时间，系统默认值为AccSet100,100;

运动控制指令-AccSet

实例：



限制：

机器人加速度百分率值最小为**20**，小于**20**以**20**计，机器人加速度坡度值最小为**10**，小于**10**以**10**计

机器人冷启动，新程序载入与程序重置后，系统自动设置为默认值。

运动控制指令-Velset

限制：
机器人冷启动，新程序载入与程序重置后，系统自动设置为默认值。
机器人运动使用参变量**[\T]**时，最大运行速度将不起作用。
Override对速度数据**(speeddata)**内所有项都起作用，例如：**TCP.**方位及外轴。但对焊接参数**welddata**与**Seamdata**内机器人运行速度不起作用
Max只对速度数据**(speeddata)**内**TCP**这项起作用。

运动控制指令-Velset

实例：

```
Velset 50,800;
Movel p1,v1000,z10,tool1;                    ----500mm/s
Movel p2,v1000\V:=2000,z10,tool1;            ----800mm/s
Movel p2,v1000\T:=5,z10,tool1;               ----10 s
Velset 80,1000
MoveL p1,v1000,z10,tool1;                    ----800mm/s
MoveL p2,v5000,z10,tool1;                    ----1000mm/s
MoveL p3,v1000\V:=2000,z10,tool1;            ----1000mm/s
MoveL p3,v1000\T:=5,z10,tool1;               ----6.25 s
```

运动控制指令-Confj;

Confj [\on] [\off];

[\on]启动轴配置数据。 **(switch)**
关节运动时，机器人移动至绝对**Modpos**点，如果无法到达，程序将停止运行。
[\off]默认轴配置数据。 **(switch)**
关节运动时，机器人移运至**Modpos**点，轴配值数据默认为当前最接近值。

运动控制指令-Confj

应用：
对机器人运行姿态进行限制与调整，程序运行时，使机器人运行姿态得到控制。系统默认值为**Confj**\on。

实例：
Confj\on;

Confj\off
限制：
机器人冷启动，新程序载入与程序重置后，系统自动设置为默认值。

运动控制指令-ConfL

ConfL [\on] [\off];

[\on]启动轴配置数据。 **(switch)**
直线运动时，机器人移动至绝对**Modpos**点，如果无法到达，程序将停止运行。
[\off]默认轴配置数据。 **(switch)**
直线运动时，机器人移运至**Modpos**点，轴配值数据默认为当前最接近值。

运动控制指令-ConfL

应用：
对机器人运行姿态进行限制与调整，程序运行时，使机器人运行姿态得到控制。系统默认值为**ConfL**\on。

实例：
ConfL\on;

ConfL\off;
限制：
机器人冷启动，新程序载入与程序重置后，系统自动设置为默认值。

运动控制指令-SingArea

SingArea[\Wrist][\off];

[\Wrist]:启用位置方位调整。（**switch)**
机器人运动时，为了避免死机，位值点允许其方位有些改变，例如：在五轴零度时，机器人四六轴平行。
[\off]：关闭位置方位调整。（**switch)**
机器人运动时，不允许位置点方位改变，是机器人默认状态。

运动控制指令-SingArea

应用：

当前指令通过对机器人位置点姿态进行些改变，可以绝对避免机器人运行时死机，但是，机器人运行路径会受影响，姿态得不到控制，通常使用于复杂姿态点。绝对不能作为工作点使用。

实例：

SingArea\wrist;

SingArea\off;

限制：

以下情况机器人将自动恢复默认值

SingArea\off

-机器人冷启动

-系统载入新的程序

-程序重置（**Start From Beginning)**

运动控制指令-PathResol

PathResol PathSample Time;
PathSample Time:路径控制**%**。（**num)**

应用：
当前指令用于更改机器人主机系统参数，调整机器人路径采样时间，从而达到控制机器人运行路径的效果，通过此指令可以提高机器人运行精度或缩短循环时间，路径控制默认值为**100%**，调整范围为**25%-400%**，路径控制百分比越小，运动精度越高，占用**CPU**资源也越多

ABB

运动控制指令-PathResol

实例：

Movej p1,v1000,fine,tool1;
Pathresol 150;

机器人在临界运动状态（重载.高速.路径变化复杂情况下接近最大工作区域），增加路径控制值，可以避免频繁死机。
外轴以很底的速度与机器人联动，增加路径控制值，可以避免频繁死机。
机器人进行高频率摆动弧焊时，需要很高的路径采样时间，需要减小路径控制值。
机器人进行小圆周或小范围复杂运动时，需要很高精度，需要减小路径控制值。

运动控制指令-PathResol

限制：
机器人必须在完全停止后才能更改路径控制值，否侧，机器人将默认一个停止点，并且显示错误信息**50146**
机器人正在更改路径控制值时，机器人被强制停止运行，机器人将不能立刻恢复正常运行（**Restart)**
以下情况机器人将自动恢复默认值**100%**。

--机器人冷启动

--系统载入新的程序

--程序重置（**Start From Beginning)**

运动控制指令-SoftAct

SoftAct[\MechUnit,]Axis,
Softness[\Ramp];
[\MechUnit]; 软化外轴名称。（**mecunnit)**
Axis: 软化外轴号码。（**mum)**
Softness: 软化值**%**（**mum)**
[\Ramp]: 软化坡度**%**。（**mum)**

应用：
当前指令用于软化机器人主机或外轴伺服系统，软化值范围**0%-%,**软化坡度范围**>=100%,**此指令必须与指令**SoftDeact**同时使用，通常不使用工作位置。

运动控制指令-SoftAct

实例：

Softact 3,20;
Softact 1,90\Ramp:=150;
Softact \MechUnit:=Orbit1,1,40\Ramp:=120;

限制：

机器人被强制停止运行后，软伺服设置将自动失效。
同一转轴软化伺服不允许被连续设置两次

错误：

```
Softact 3,20;
Softact 3,30;
```

正确：

```
Softact 3,20;
Movej *,v100,fine,tool1;
Softact 3,30;
```

运动控制指令-**SoftDeact**

SoftDeact [\Ramp];
[\Ramp]:软化坡度，>=100%（unm)

应用：
当前指令用于使软化机器人主机或外轴伺服系统指令**SoftAct**失效。

实例：
SoftAct 3,20
SoftDeact;
SoftAct 1,90;
SoftDeact\Ramp:=150;

外轴激活指令

ActUnit

DeactUnit

外轴激活指令-ActUnit

ActUnit MecUnit;

MecUnit: 外轴名。 （**mecunit)**

应用：

将机器人一个外轴激活，例如：当多个外轴公用一个驱动板时，通过外轴激活指令 ActUnit**选择当前所使用的外轴。**

ABB

外轴激活指令-ActUnit

实例：

```
MoveL p10,v100,fine,tool1          --p10,外轴不动
ActUnit track_motion;
MoveL p20,v100,z10,tool1;          --p20,外轴联动
DeactUnit track_motion;             track_motion
ActUnit orbit_a;
MoveL p30,v100,z10,tool1;          --P30,外轴联动
                                    orbit_a
```

限制：

不能在指令StorePath…Restopath**内使用。**

不能在预置程序RESTART**内使用**

不能在机器人转轴处于独立状态使用。

外轴激活指令-DeactUnit

DeactUnit MecUnit;

MecUnit: 外轴名。 （**mecunit)**

应用：

使机器人外轴失效，例如，当多个外轴公用一个驱动板时，通过外轴激活指令 DeactUnit**使当前所使用的外轴失效。**

外轴激活指令- **DeactUnit**

实例：

```
MoveL p10,v100,fine,tool1          --p10,外轴不动
ActUnit track_motion;
MoveL p20,v100,z10,tool1;          --p20,外轴联动
DeactUnit track_motion;             track_motion
ActUnit orbit_a;
MoveL p30,v100,z10,tool1;          --P30,外轴联动
                                    orbit_a
```

限制：

不能在指令StorePath…Restopath**内使用。**

不能在预置程序RESTART**内使用**

ABB

计数指令

Add

Clear

Incr

Decr

计数指令-Add

Add Name, AddValue;

Name: 数据名称。（num)
AddValue: 增加的值。（num)

应用：

在一个数字数据上增加相应的值，可以用赋值指令替代。

实例：

```
Add reg1,3;       等同于   reg1:=reg1+3;
Add reg1,-reg2;   等同于   reg1:=reg1-reg2;
```

计数指令-Clear

Clear Name;

Name: 数据名称。（num)

应用：

将一个数字数据的值归零，可以用赋值指令替代。

实例：

Clear reg1; 等同于 reg1:=0;

计数指令-Incr

Incr Name;

Name: 数据名称。（num)

应用：

将一个数字数据值上增加1，可以用赋值指令替代。一般用于产量计数。

实例：

Incr reg1; 等同于 reg1:=reg1+1;

计数指令-Decr

Decr Name;

Name: 数据名称。（num)

应用：

在一数字数据值上增加1，可以用赋值指令替代，一般用于产量计数

实例：

Decr reg1; 等同于 reg1:=reg1-1;

输入输出指令

AliasIO

InvertDO

IODisable

IOEnable

PluseDO

Reset

Set

SetAO

SetDO

SetGO

WaitDI

WaitDO

输入输出指令-AliasIO

AliasIO FromSignal, ToSignal;

FromSignal :机器人系统参数内所定义的信号名称。
。 （**sigunlxx or string)**
ToSignal:机器人程序内所使用的信号名称（**sigunlxx** ）

应用：

对机器人系统参数内定义的信号名称进行化名，给机器人程序使用，一般使用与 Loaded Modules**或**Built-in Modules**内。例如：多台机器人使用相同系统参数。**

ABB

输入输出指令-AliasIO

实例：

```
VAR signaldo alias_do;
CONST string config_string:=“config_do”;
PROC prog_start()
AliasIO config_do,alias_do;
AliasIO config_string,alias_do;
ENDPROC
```

Config_do，在系统参数内定义。

alias_do，机器人程序内定义

输入输出指令-AliasIO

限制：
指令**AliasIO**必须放置在预置程序**START**内或程序内使用相应信号之前。
指令**AliasIO**在示教器上无法输入，只能通过离线编程输入。
指令**AliasIO**需要软件**Developer's Functions**支持。

输入输出指令-InvertDO

InvertDO Singnal;

Singnal:输出信号名称 **(signaldo)**

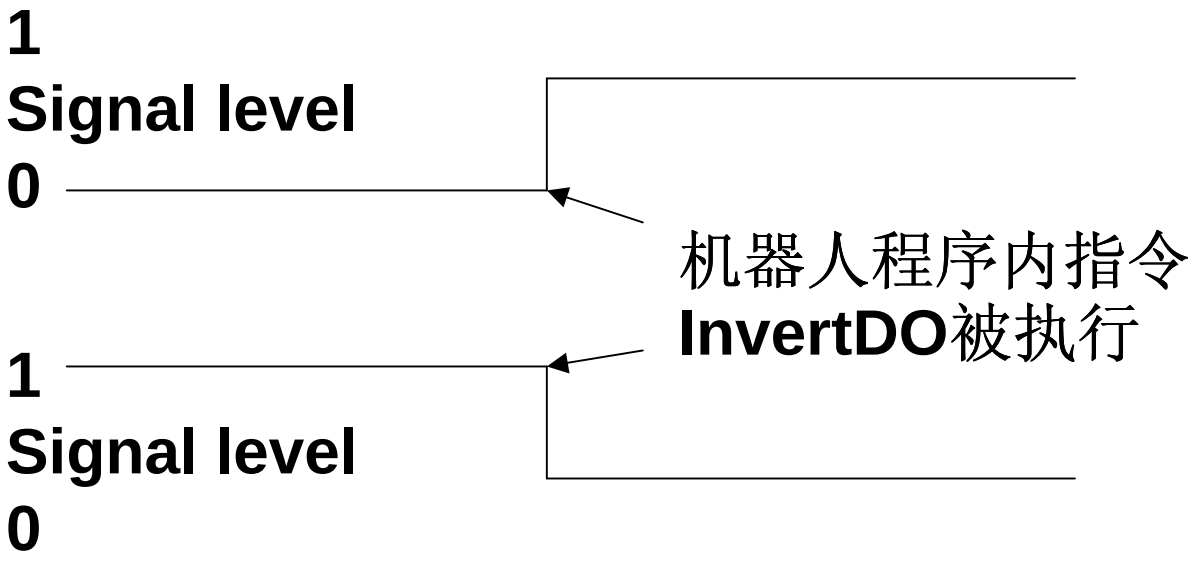
应用：

将机器人输出信号值反转，0为1，1为0，在系统参数内也可定义。

实例：

InvertDO do15;

输入输出指令-InvertDO



输入输出指令-IODisable

IODisable UnitName,MaxTime;

UnitName :输入输出板名称。 **(num)**

MaxTime : 最长等待时间。 **(num)**

应用：

通过指令可以使机器人输入输出板在程序运行时自动失效，系统将一块输入输出板失效需要2-5秒，如果失效时间超过最长等待时间，系统将进入Error Handler,错误代码为ERR_IODISABLE,如果例行程序没有Error Handler机器人将停机报错。

ABB

输入输出指令-IODisable

实例：

```
PROC go_home()
    recover_flag:=1
    IODisable “cell1”,0;
    MoveJ home,v1000,fine,tool1;
    recover_flag:=2
    IODisable “cell1”,5;
ERROR
    IF ERRNO=ERR_IODISABLE THEN
        IF recover_flag=1 THEN
            TRYNEXT;
        ELXEIF recover_flag=2 THEN
             RETRY;
        ENDIF
    ELSEIF ERRNO=ERR_EXCRTYMAX THEN
         RETRY;
         ENDIF;
    ErrWrite “IODisable error”, “Restart the program”;
    Stop;
    ENDIF
ENDPROC
```

输入输出板cell1开始失效，最长等待时间为0，肯定进入Error Handler处理。

利用机器人移动至home时间完成输入输出失效。

确认输入输出板cell失效

连续5次RETRY,仍无法完成输入输出板失效。

ABB

输入输出指令-IODisable

Error Handling:

ERR_IODISABLE

超过最长等待时间，系统仍未完成输入输出板失效。

ERR_CALLIO_INTER

系统在执行输入输出板失效与激活时，当前输入输出板再次被失效或激活。

ERR_NAME_INVALID

输入输出板名称错误或无法进行失效与激活操作。

输入输出指令-IOEnable

IOEnable UnitName,MaxTime

Unit Name:输入输出板名称。 （num)

MaxTime：最长等待时间。 （num)

应用：

通过指令可以使机器人输入输出板在程序运行时自动激活，系统将一块输入输出板失效需要2-5秒，如果激活时间超过最长等待时间，系统将进入Error Handler,错误代码为ERR_IODISABLE,如果例行程序没有Error Handler机器人将停机报错。

ABB

输入输出指令-IOEnable

实例：

```
WAR num max_retry:=0;
...
IOEnable “cell1”,0;
setDO cell1_sig3,1;
ERROR
IF ERRNO=ERR_IOENABLE THEN
  IF max_retry<5 THEN
    WaitTime 1;
    max_retry:=max_retry+1;
    RETRY;
  ELSE
    RAISE;
  ENDIF
```

输入输出板cell开始激活，最长等待时间为0，肯定进入Hrror Handler处理

通过每次1秒进行计数，连续5次无法激活输入输出板，执行指令RAISE

输入输出指令- IOEnable

Error Handling:

ERR_IODISABLE

超过最长等待时间，系统仍未完成输入输出板激活。

ERR_CALLIO_INTER

系统在执行输入输出板失效与激活时，当前输入输出板再次被失效或激活，形成冲突。

ERR_NAME_INVALID

输入输出板名称错误或无法进行失效与激活操作。

输入输出指令- PluseDO

PluseDO [\High][\Plenggh] Signal;

[\High] :输出脉冲时，输出信号可以处在高电平（switch)

[\Plenggh] : 脉冲长度，0.1s-32s,默认值为0.2s（num)

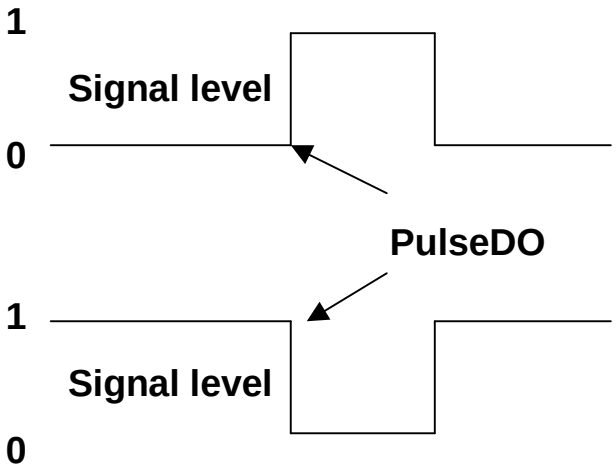
Signal :输出信号名称（signaldo)

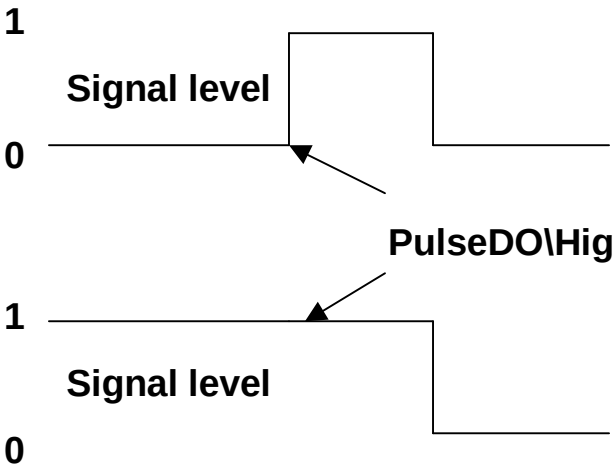
应用：

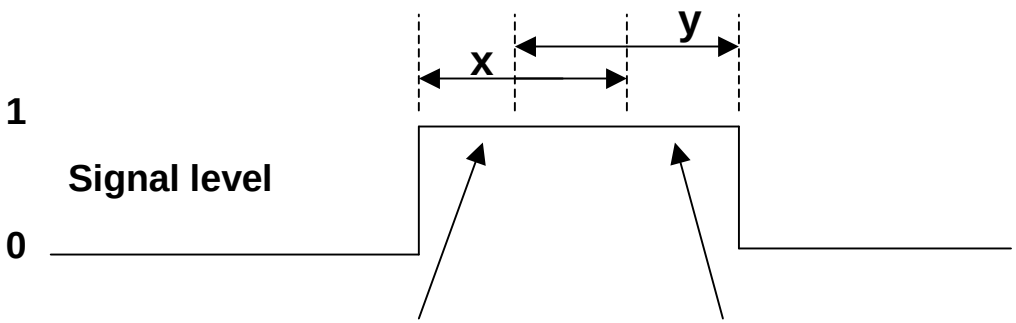
机器人输出数字脉冲信号，一般作为运输链完成信号或计数信号。

输入输出指令- **PluseDO**

实例：







PulseDO\High\Plength:=x

PulseDO\High\plength:=y

输入输出指令- **PluseDO**

限制：

机器人脉冲输出长度小于0.01秒，系统将报错，不得不重新热启动。

例如：

```
WHILE TRUE DO
    PulseDO do5;
ENDWHILE
```

输入输出指令-Reset

Reset signal;

signal :输出信号名称。（signaldo)

应用：

将机器人相应数字输出信号值置为0，与指令Set相对应，是自动化重要组成部分。

实例：

Reset do12;

输入输出指令-Set

Set signal;

signal :机器人输出信号名称。（signaldo)

应用：

将机器人相应数字输出信号值置为1，与指令Reset相对应，是自动化重要组成部分。

实例：

Set do12;

输入输出指令-SetAO

SetAO signal,Value;

signal :模拟量输出信号名称。（signaldo)

Value :模拟量输出信号值。（num)

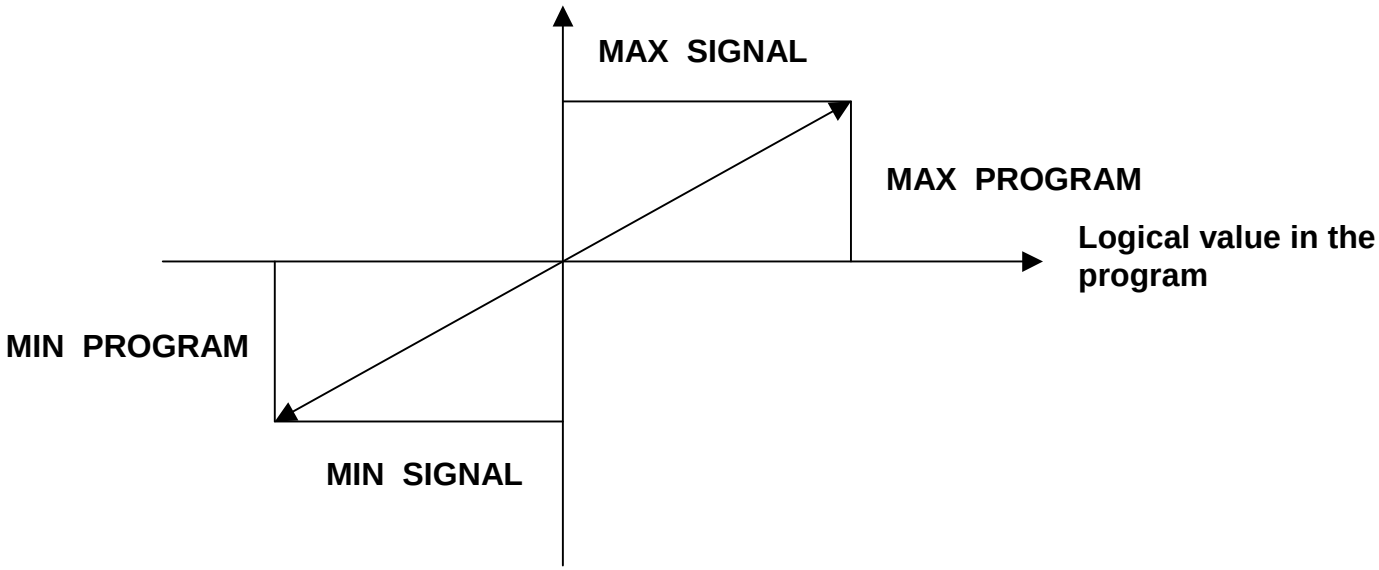
应用：

将机器人当前模拟量输出信号输出相应的值，例如：如机器人焊接时，通过模拟量输出控制焊接电压与送丝速度。

ABB

输入输出指令-SetAO

Value :模拟量输出信号值。 （num)



实例：

SetAO ao2,5.5;

SetAO weldcurr,curr_outp;

输入输出指令-SetDO

SetDO [\Sdelay] signal,Value;

[\Sdelay] :延迟输出时间s。 （num)

signal : 输出信号名称。 （signaldo)

Value : 输出信号值。 （num)

应用：

设置机器人相应数字输出信号的值，与指令Set与Reset雷同，并且可以设置延时，延时范围为0.1s-32s，默认状态没有延时。

例如：

SetDO\Sdelay:=0.2,weld,high;

输入输出指令-SetGO

SetGO [\Sdelay] signal,Value;

[\Sdelay] :延迟输出时间s。 （num)

signal : 输出信号名称。 （signaldo)

Value : 输出信号值。 （num)

应用：

设置机器人相应组合数字输出信号的值，（采用8421码），可以设置延时，延时范围为0.1s-32s，默认状态没有延时。

例如：

SetGO\Sdelay:=0.2,go_Type,10;

输入输出指令-WaitDI

WaitDI Signal,Value [\MaxTime][\TimeFlag];

signal： 输出信号名称。 （signaldo)

Value： 输出信号值。 （dionum)

[\MaxTime]：最长等待时间。 （num)

[\TimeFlag] :超出逻辑量。 （bool)

应用：

等待数字输入信号满足相应值，达到通信目的，是自动化生产的重要组成部分，例如：机器人等待工件到位信号。

输入输出指令-WaitDI

实例：

```
PROC Pickpart()
    MoveJ pPrePick,vFastempty,zBig,tool1;
    WaitDI di_Ready,1;
    …
    ENDPROC
PROC PickPart()
    MoveJ pPrepick,vFastEmpty,zBig,tool1;
    WaitDI di_Ready,1\WMaxTime:=5
    …
    IF ERRNO=ERR_WAIT_MAXTIME THEN
       TPWrite "……";
          RETRY;
       ELSE
          RAISE;
       ENDIF
ENDPROC
```

机器人等待输入信号，直到信号di_Ready值为1，才执行随后指令。

机器人等待相应输入信号，如果5秒内仍没有等到信号di_Ready值为1，自动进行Error Handler处理，如果没有Error Handler,机器人停机报错。

输入输出指令-WaitDI

实例：

```
PROC Pickpart()
    MoveJ pPrePick,vFastempty,zBig,tool1;
    bTimeout:=TRUE;
    nCounter:=0;
    WHILE bTimeout DO
        IF nCounter >3 THEN
            TPWrite "……";
        ENDIF
        IF nCounter >30 THEN
            Stop;
        ENDIF
        WaitDI di_Ready,1\MaxTime:=1\TimeFlag:=bTimeout;
        Incr nCunter;
      ENDWHILE
      …
ENDPROC
```

机器人等待输入信号，直到信号di_Ready值为1，机器人执行随后指令。但此时TimeFlag值为TRUE;机器人等到信号di_Ready值为1，此时，TimeFlag值为FALSE

ABB

输入输出指令-WaitDO

WaitDO Signal,Value [\MaxTime][\TimeFlag];

signal： 输出信号名称。 （signaldo)

Value： 输出信号值。 （dionum)

[\MaxTime]：最长等待时间。 （num)

[\TimeFlag] :超出逻辑量。 （bool)

应用：

等待数字输出信号满足相应值，达到通信目的，因为输出信号一般情况下受程序控制，此指令很少使用。

ABB

输入输出指令-WaitDO

实例：

```
PROC Pickpart()
    Set do03_Grip;
    WaitDO do03_Grip,1;
    …
ENDPROC
PROC Grip()
    Set do03_Grip;
    WaitDO do03_Grip,1\MaxTime:=5
    …
ERROR
  IF ERRNO=ERR_WAIT_MAXTIME THEN
     TPWrite “……”
     RETPY;
  ELSE
     RAISE;
   ENDIF
ENDPROC
```

机器人等待输出信号，直到信号do03_Grip值为1，才执行随后指令。

机器人等待相应输入信号，如果5秒内仍没有等到信号do03_Ready值为1，自动进行Error Handler处理，如果没有Error Handler，机器人停机报错。

输入输出指令-WaitDO

实例：

```
PROC Pickpart()
    Set do03_Grip;
    bTimeout:=TRUE
    nCunter:=0;
    WHILE bTimeout DO
        IF nCounter >3 THEN
          TPWrite "……";
        ENDIF
        IF nCounter >30 THEN STOP
        WaitDO do03_Grip.1\Maxtime:=1\TimeFlag:=bTimeout;
        Incr nCounter;
     ENDWHILE
     …
ENDPROC
```

机器人等待输入信号，直到信号di_Grip值为1，机器人执行随后指令。但此时TimeFlag值为TRUE;机器人等到信号di_Grip值为1，此时，TimeFlag值为FALSE

程序运行停止指令

Break

Exit

STOP

ExitCycle

程序运行停止指令-Break

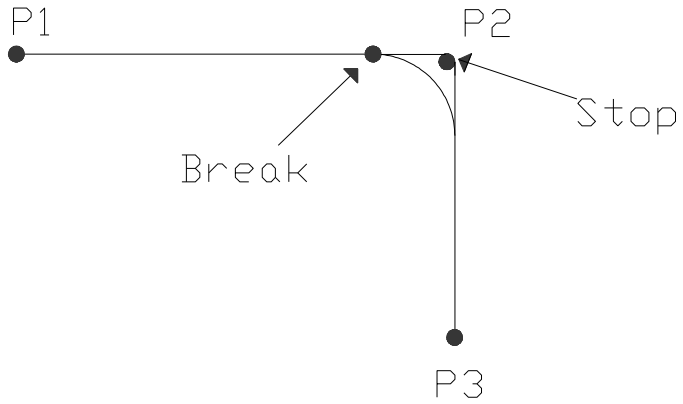
Break;

应用：
机器人在当前指令行立刻停止运行，程序运行指针停留在下一行指令，可以用Start键继续运行机器人

实例：

```
...
Break
...
```

程序运行停止指令-Break



区别：

```
MoveL p2,v100,z30,tool0;

Break; (Stop;)

MoveL p3.v100,fine,tool0;
```

程序运行停止指令-Exit

Exit;

应用：
机器人在当前指令行停止运行，并且程序重置，程序运行指针停留在主程序第一行。

实例：

```
...
Exit
...
```

程序运行停止指令-Stop

Stop [\NoRegain];

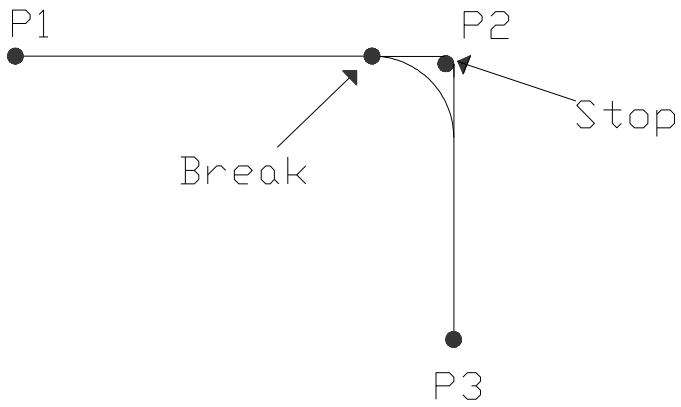
[\NoRegain]:路径恢复参数

应用：
机器人在当前指令行停止运行，程序运行指针停留在下一行指令。可以用Start键继续运行机器人，属于临时性停止，如果机器人停止期间被手动移动后，然后直接启动机器人，机器人将警告确认路径，如果此时采用参变量**[\NoRegain]**，机器人将直接运行。

程序运行停止指令-Stop

实例：

```
...
Stop;
...
```



区别：

```
MoveL p2,v100,z30,tool0;
Stop; (Break;)
MoveL p3,v100,fine,tool0;
```

程序运行停止指令-ExitCycle

ExitCycle;

应用：
机器人在当前指令行停止运行，并且设定当前循环结束，机器人自动从主程序第一行继续运行下一个循环。

程序运行停止指令-ExitCycle

实例：

```
PROC main()
  IF cyclecount=0 THEN
     CONNECT error_intno WITH error_trap;
     ISignalDI di_error,1,error_intno;
  ENDIF
  cyclecount:=cyclecount+1;
  ! Start to do something intelligent
   . . . .
ENDPROC
TRAP error_trap
     TPWrite “I will start on the next item”
     ExitCycle;
     ENDTRAP
```

例行程序指令

ProcCall **CallByvar**

例行程序指令-ProcCall

Procedure {Argument};

Procedure : 例行程序名称。（**Identifier)**

{Argument} :例行程序参数。（**all)**

应用：

机器人调用相应例行程序，同时给带有参数的例行程序中相应参数赋值。

实例：

```
Weldpipe1;

Weldpipe2 10,lowspeed;

Weldpipe3 10\speed:=20;
```

例行程序指令-ProcCall

限制：
机器人调用带参数的例行程序时，必须包括所有强制性参数。
例行程序所有参数位置次序必须与例行程序设置一致。
例行程序所有参数数据类型必须与例行程序设置一致。
例行程序所有参数数据性质必须为Input,Variable或Persistent.

例行程序指令-CallByVar

CallByVar Name,Number;

Name： 例行程序名称第一部分。（**string)**

Number :例行程序名称第二部分。（**num)**

应用：

通过指令中相应数据，机器人调用相应例行程序，但无法调用带有参数的例行程序。

实例：

```
reg1:=Ginput(gi_Type);
callByVar “proc”,reg1;
```

例行程序指令-CallByVar

限制：
不能调用带参数的例行程序。

所有被调用的例行程序名称第一部分必须相同，例如：
proc1.proc2.proc3.

使用CallByVar指令调用例行程序比直接采用ProcCall调用例行程序需要更长时间。

Error Handling

ERR_REFUNKPRC

系统无法找到例行程序名称第一部分。

ERR_CALLPROC

系统无法找到例行程序名称第二部分。

例行程序指令-CallByVar

实例比较：

```
TEST reg1
CASE 1:
  lf_door door_Loc;
CASE 2:
  rf_door door_loc;
CASE 3:
  lf_door door_loc;
CASE 4:
  rr_door door_loc;
 EDFAULT:
     EXIT;
ENDTEST
CallByVar “proc”,reg1;
%“proc”+NumTostr(reg1,0)% door_loc;
```

指令CallByVar不能调用带有参数的例行程序

通过PAPID结构仍可以调用带有参数的例行程序

计时指令

ClkReset

ClkStart

ClkStop

计时指令-ClkReset

ClkReset Clock;

Clock： 时钟名称。（**clock)**

应用：
将机器人相应的时钟复位，常用于记录循环时间或机器人跟踪运输链。
实例：

```
ClkReset clock1;
ClkStart clock1;
RunCycle;
ClkStop clock1;
nCycleTime:=ClkRead(clock1);
TPWrite “Last Cycle Time: “\Num:=nCycleTime
```


计时指令-ClkStart

ClkStart Clock;

Clock： 时钟名称。（**clock)**

应用：
启动机器人相应时钟，常用于记录循环时间或机器人跟踪输链。机器人时钟启动后，时钟不会因为机器人停止运行或关机而停止计时，在机器人时钟运行时，指令ClkStop与ClkReset仍起作用。

计时指令-ClkStart

实例：

```
ClkReset clock1:
ClkStart clock1:
RunCycle;
ClkStop clock1;
nCycleTime:=ClkRead(clock1);
TPWrite “last Cycle Time: ”\Num:=nCycleTime;
```

限制：

机器人时钟计时超过4，294，967秒，既49天17小时2分47秒，机器人蒋出错。Error Handler代码为ERR_OVERFLOW.

ABB

中断指令

CONNECT

IDelete

ISingnalDI

ISignalDO

ISignalAI

ISignalAO

ISleep

Iwatch

Idisable

Ienable

ITimer

中断指令-CONNECT

CONNECT Interrupt
WITH Trap routine;

Interrupt : 中断数据名称。（**intnum)**

Trap routine : 中断数据程序。（**identifier)**

应用：
将机器人相应中断数据连接到相应的中断处理程序，是机器人中断功能必不可少的组成部分，必须同指令ISignalDI,ISignalDO,ISignalAI,ISignalAO或ITimer联合使用。

中断指令-CONNECT

实例：

```
VAR intnum,intInspect;
PROC main()
    ...
     CONNECT intInspect WITH rAlarm;
     ISignalDI di01_Vacuum,0,intInspect;
       ...
ENDPROC
TRAP rAlarm
      TPWrite “Grip Error”;
       Stop;
      WaitDI di01_Vacuum,1;
ENDTRAP
```

中断指令-CONNECT

限制：
中中断数据的数据类型必须为变量（VAR）
一个中断数据不允许同时连接到多个中断处理程序，但多个中断数据可以共享一个中断处理程序
当一个中断数据完成连接后，这个中断数据不允许再次连接到任何中断处理程序（包括已经连接的中断处理程序）。如果需要再次连接至任何中断程序，必须先使用指令IDelete将原连接支除。

中断指令-CONNECT

Error Handler：

ERR_ALRDYCNT
中断数据已经被连接至中断处理程序

ERR_CNTNOTVAR
中断数据的数据类型不是变量（VAR）

ERR_INOMAX
没有更多的中断数据可以使用

中断指令-IDelete

IDelete Interrupt;

Interrupt : 中断数据名称。（**intnum)**

应用：

将机器人相应中断数据与相应的中断处理程序之间原连接去除。

实例：

```
...
CONNECT intInspect WITH rAlarm;
ISigalDI di01_Vacuum,0,intInspect;
...
Idelete intInspect;
```


ABB

中断指令-IDelete

限制：
执行指令Idelete后，当前中断数据的连接被完全清除，如需再次使用这个中断数据必须重新用指令CONNECT连接至相应的中断处理程序。

在以下列情况下，中断将被自动去除;
-重新载入新的运行程序
-机器人运行程序被重置，程序指针回到主程序第一行（Start from Beginning).
-机器人程序指针被移至任意一个例行程序第一行（Move pp to Routine)

中断指令-ISignalDI

ISignalDI [\Single],Signal, TriggValue,Interrupu;

[\Single]：　单次中断开关。（**switch)**

Signal：　触发中断信号。（**signaldi)**

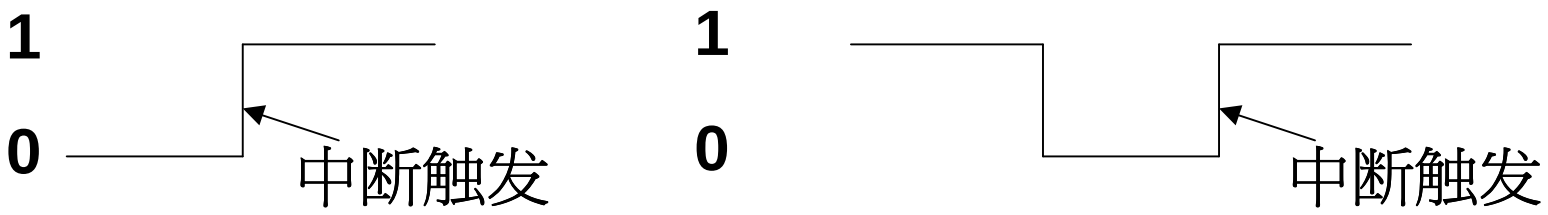
TriggValue：触发信号值。　（**dionum)**

Interrupu：　中断信号名称。（**intnum)**

应用：

使用相应的数字信号输入信号触发相应的中断功能，必须同指令CONNECT联合使用。

中断指令-ISignalDI



实例：

```
…
CONNECT int1 WITH iroutine1;
ISignalDI\Signal di01,1,int1;
…
CONNECT int2 WITH iroutine2;
ISignalDI di02,1,int1;
```

中断功能在单次触发触发后失效

中断功能持续有效，只有在程序得置或运行指令IDelete后才失效

中断指令-ISignalDI

限制：

当一个中断数据完成连接后，这个中断数据不允许再次连接到任何中断处理程序（包括已经连接的中断处理程序）。如果需要再次连接至任何中断处理程序，必须先使用指令Ieelete将原连接去除。

```
PROC main()
    CONNECT int1 WITH r1;
    ISignalDI di01,1,int1;
    ...
    Idelete int1;
ENDPROC
```

```
PROC main()
    CONNECT int1 WITH r1;
    ISignalDI di01,1,int1;
    WHILE TRUE DO
    ...
    ENDWHILE
ENDPROC
```

中断指令-ISignalDO

ISignalDO [\Single],Signal, TriggValue,Interrupu;

[\Single]：　单次中断开关。（**switch)**

Signal：　触发中断信号。（**signaldi)**

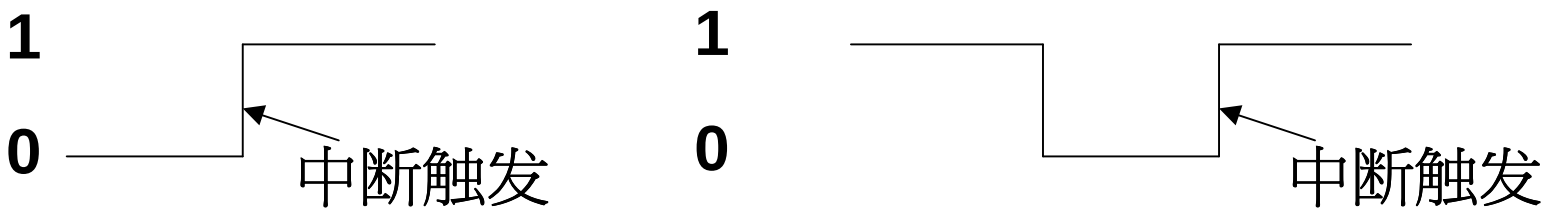
TriggValue：触发信号值。　（**dionum)**

Interrupu：　中断信号名称。（**intnum)**

应用：

使用相应的数字信号输出信号触发相应的中断功能，必须同指令CONNECT联合使用。

中断指令-ISignalDO



实例：

```
…
CONNECT int1 WITH iroutine1;
ISignalDO\Signal do01,1,int1;
…
CONNECT int2 WITH iroutine2;
ISignalDO do02,1,int1;
```

中断功能在单次触发触发后失效

中断功能持续有效，只有在程序得置或运行指令IDelete后才失效

ABB

中断指令-ISignalDI

限制：

当一个中断数据完成连接后，这个中断数据不允许再次连接到任何中断处理程序（包括已经连接的中断处理程序）。如果需要再次连接至任何中断处理程序，必须先使用指令Ieelete将原连接去除。

```
PROC main()
    CONNECT int1 WITH r1;
    ISignalDO do01,1,int1;
    ...
    IDelete int1;
ENDPROC
```

```
PROC main()
    CONNECT int1 WITH r1;
    ISignalDO do01,1,int1;
    WHILE TRUE DO
    ...
    ENDWHILE
ENDPROC
```

中断指令-ISignalAI

ISgnalAI [\Single],Signal,
Condition,HighValue,lowValue,
DeltaValue,[\DPos][\Dneg]
Interrupt;

[\Single] :	单次中断开关。（**switch)**
Signal :	触发中断信号。（**signaldi)**
Condition :	中断触发状态。（**adotrigg)**
HighValue :	最大逻辑值。（**num)**
lowValue :	最小逻辑值。（**num)**
DeltaValue :	中断复位差值。（**num)**
[\Dpos] :	正值中断开关。（ **switch)**
[\Dneg] :	负值中断开关。（**switch)**
Interrupt; :	中断数据名称 （**intnum)**

中断指令-ISignalAI

中断触发状态：

AIO_ABOVE_HIGH
模拟量信号逻辑值大于最大逻辑值（**High Value)**

AIO_BELOW_HIGH
模拟量信号逻辑值小于最大逻辑值（**High Value)**

AIO_ABOVE_LOW
模拟量信号逻辑值大于最小逻辑值（**Low Value)**

AIO_BELOW _LOW
模拟量信号逻辑值小于最小逻辑值（**Low Value)**

AIO_BETWEEN
模拟量信号逻辑值处于最小逻辑值（Low Value)与最大逻辑值（High Value)之间。

AIO_OUTSIDE
模拟量信号逻辑值大于最大逻辑值（High Value）或小于最小（Low Value)

AIO_ALWAYS
总是触发中断，与模拟量信号逻辑值处于最小逻辑值（Low Value）与最大逻辑值（High Value）无关。

应用：

使用相应的模拟量输入信号触发相应的中断功能，必须同指令CONNECT联合使用

中断指令-ISignalAI

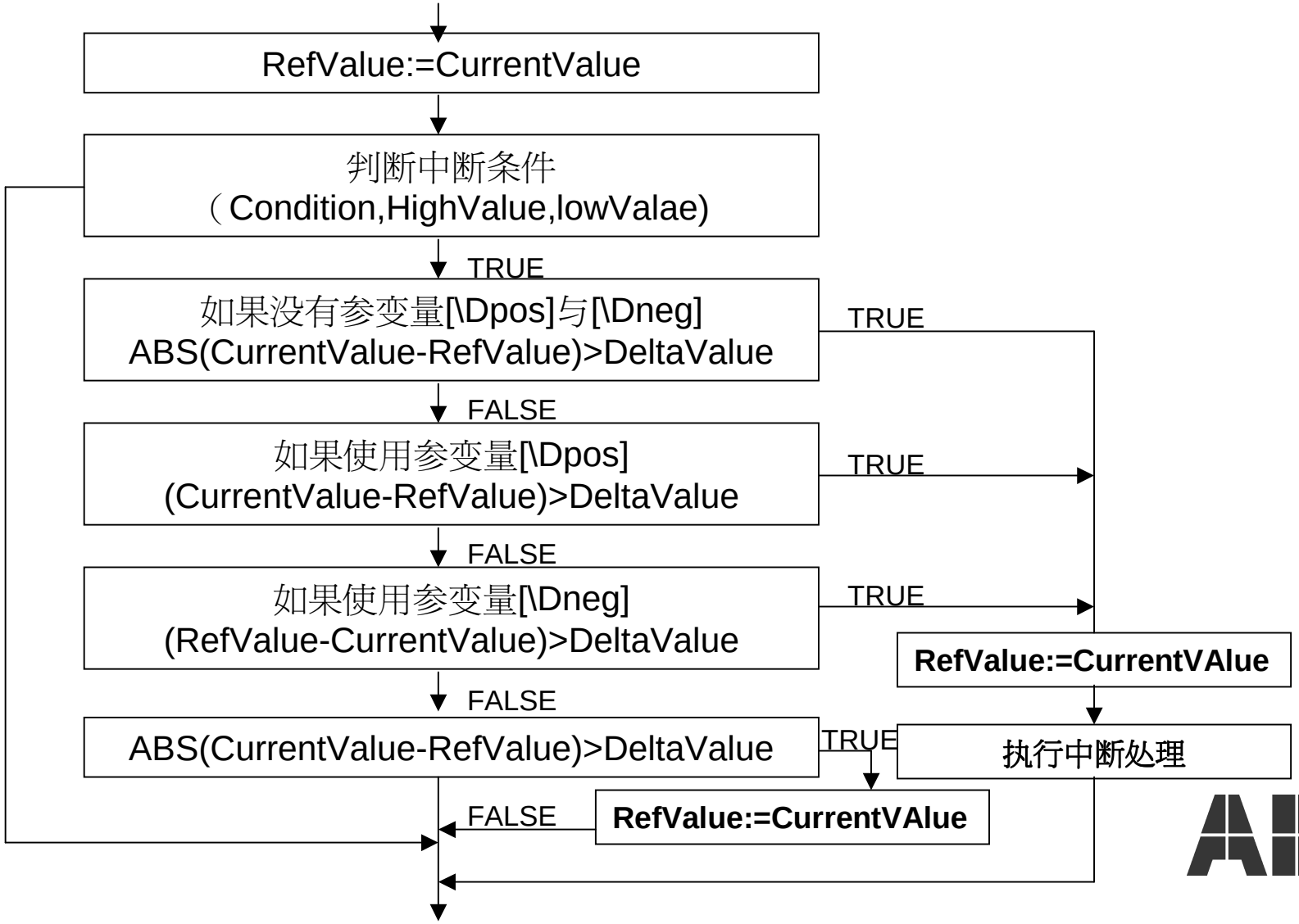
实例：

```
...
CONNECT int1 WITH iroutine1;
ISignalAI\signal ai1,AIO_ BETWEEN,2,1,0,int1;
...
CONNECT int2 WITH iroutine2;
ISignalAI ai2,AIO_BETWEEN,1.5,0.5,0,int1;
...
CONNECT int3 iroutine3;
ISignalAI ai3,AIO_BETWEEN,1.5,0.5,0.1,init3;
```

中断功能在单次触发触后失效

中断功能持续有效，只有在程序重置或运行指令IDelete后才失效

中断指令-ISignalAI



中断指令-ISignalAI

限制：

当前最大逻辑值（HighValue)与最小逻辑值LowValue)必须在模拟量信号所定义的逻辑范围内

最大逻辑值（HighValue)必须大于最小逻辑值LowValue)

中断复位差值（DeltaValue)必须为正数或0.

指令ISignalDIr的限制，仍适用。

中断指令-ISignalAO

ISgnalAO [\Single],Signal,
Condition,HighValue,lowValue,
DeltaValue,[\DPos][\Dneg]
Interrupt;

DeltaValue : 中断复位差值。（**num)**

[\Dpos] : 正值中断开关。（ **switch)**

[\Dneg] : 负值中断开关。（**switch)**

Interrupt; : 中断数据名称 （**intnum)**

中断指令-ISignalAO

中断触发状态：

AIO_ABOVE_HIGH
模拟量信号逻辑值大于最大逻辑值（**High Value)**

AIO_BELOW_HIGH
模拟量信号逻辑值小于最大逻辑值（**High Value)**

AIO_ABOVE_LOW
模拟量信号逻辑值大于最小逻辑值（**Low Value)**

AIO_BELOW _LOW
模拟量信号逻辑值小于最小逻辑值（**Low Value)**

AIO_BETWEEN
模拟量信号逻辑值处于最小逻辑值（Low Value)与最大逻辑值（High Value)之间。

AIO_OUTSIDE
模拟量信号逻辑值大于最大逻辑值（High Value）或小于最小（Low Value)

AIO_ALWAYS
总是触发中断，与模拟量信号逻辑值处于最小逻辑值（Low Value）与最大逻辑值（High Value）无关。

应用：

使用相应的模拟量输出信号触发相应的中断功能，必须同指令CONNECT联合使用

中断指令-ISignalAO

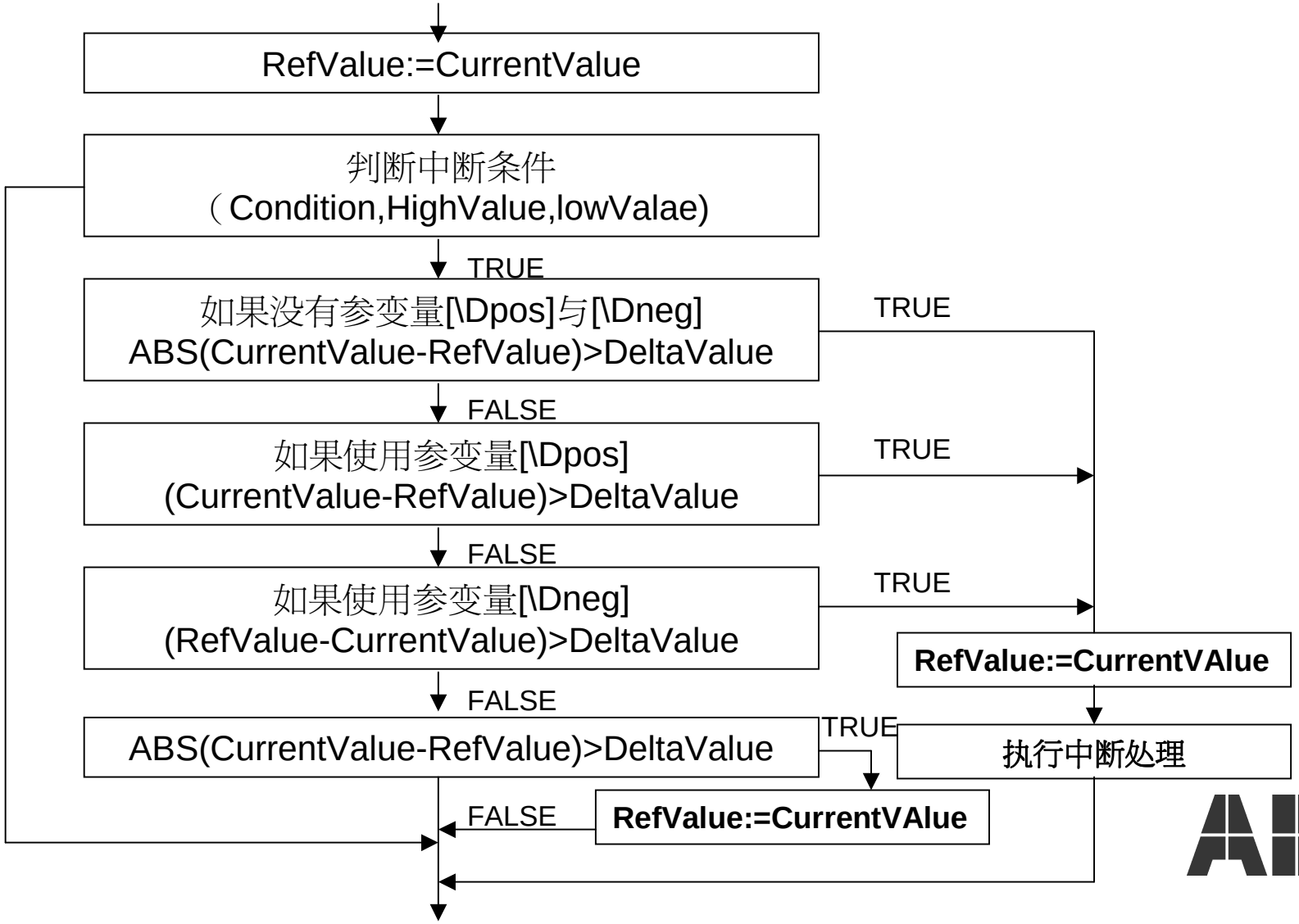
实例：

```
...
CONNECT int1 WITH iroutine1;
ISignalAO\single ao1,AIO_BETWEEN,2,1,0,int1;
...
CONNECT int2 WITH iroutine2;
ISignalAO ao2,AIO_BETWEEN,1.5,0.5,0,int2;
...
CONNECT int3 iroutine3;
ISignalAO ao3,AIO_BETWEEN,1.5,0.5,0.1,init3;
```

中断功能在单次触发触后失效

中断功能持续有效，只有在程序重置或运行指令IDelete后才失效

中断指令-ISignalAO



中断指令-ISignalAO

限制：

当前最大逻辑值（HighValue)与最小逻辑值LowValue)必须在模拟量信号所定义的逻辑范围内

最大逻辑值（HighValue)必须大于最小逻辑值LowValue)

中断复位差值（DeltaValue)必须为正数或0.

指令ISignalDIr的限制，仍适用。

中断指令-ISleep

Isleep Interrupt;

Interrupt : 中断数据名称。（**intnum)**

应用：
机器人相应中断数据暂时失效，直到执行指令IWatch后才恢复。

ABB

中断指令-ISleep

实例：

```
...
CONNECT intInspect WITH rAlarm;
ISignalDI di01_vacuum,0,intInspect;
...
...
ISleep intInspect;
... 
IWatch intinspect;
... 
```

中断监控

中断失效

中断监控

Error Handler：

ERR_UNKINO

无法找到当前的中断数据。

中断指令-IWatch

IWatch Interrupt;

Interrupt : 中断数据名称。（**intnum)**

应用：
激活机器人已失效的相应中断数据，正常情况下，
与指令ISleep配合使用。

ABB

中断指令- IWatch

实例：

```
...
CONNECT intInspect WITH rAlarm;
ISignalDI di01_vacuum,0,intInspect;
...
...
ISleep intInspect;
... 
IWatch intinspect;
... 
```

← 中断监控

← 中断失效

← 中断监控

Error Handler：

ERR_UNKINO

无法找到当前的中断数据。

中断指令-IDisable

IDisable;

应用：
使机器人相应中断功能暂时不执行，直到执行指令IEnable后，才进入中断处理程序，此指令使用于机器人正在执行不希望被打断的操作期间，例如：通过通信口读写数据。

ABB

中断指令-IDisable

实例：

```
...
IDisable;
FOR i FROM 1 TO 100 D0
    character [i]:=ReadBin(sensor);
ENDFOR
IEnable;
...
```


ABB

中断指令-IEisable

实例：

```
...
IDisable;
FOR i FROM 1 TO 100 D0
    character [i]:=ReadBin(sensor);
ENDFOR
IEnable;
...
```


ABB

中断指令-ITimer

Itimer [\Single],Time,Interrupt;

[\Single] : 单次中断开关。（**switch)**

Time : 触发中断时间。（**num)**

Interrupt : 中断数据名称。（**intnum)**

应用：
定时处理机器人相应中断数据，此指令常使用于通过通信口读写数据等场合。

ABB

中断指令-ITimer

实例：

```
...
CONNECT timeint WITH check_serialch;
Itimer 60,timeint;
...
TRAP check_serialch
   WriteBin ch1,buffer,1;
   IF ReadBin(ch1\Time:=5)>0 THEN
      TPWrite “Communication is broken”;
      EXIT;
ENDIF
ENDTRAP
```

通信指令（人机对话）

TPEerase

TPWrite

TPReadFK

TPReadNum

ErrWrite

TPShow

通信指令- TPEerase

TPEerase;

应用：
清屏指令，将机器人示教器屏幕上所有显示清除，是机器人屏幕显示重要组成部分。
实例：

```
TPErase;
TPWrite “    ABB Robotics      ”;
TPWrite “_____________________”;
```

通信指令- TPWrite

TPWrite String [\Num][\Bool][\POS][\Orient];

String: 屏幕显示的字符串。 **(string)**

[\Num] : 屏幕显示数字数据值。 **(string)**

[\Bool] : 屏幕显示逻辑量数据。 **(string)**

[\POS] : 显示位置值。**X Y Z.** **(string)**

[\Orient] : 显示方位q1 q2 q3 q4 **(string)**

应用：

在示教器屏幕上显示相应字符串，字符串最长80个字节，屏幕每行可显示40个字节。在字符串后可显示相应参变量。

通信指令- TPWrite

实例：

TPWrite string1;
TPWrite “Cycle Time=　　”\Num:=nTime;

限制：

每个TPWrite指令只允许单独使用参变量，不允许同时使用。
参变量值<0.000005或0.999995将圆整。

Argument	Value	Text string
\Num	23	“23”
\Num	1.141367	“1.14137”
\Bool	TRUE	“TRUE’
\Pos	[1817.3,905.17,879.11]	“[1817.3,905.17,879.11]”
\Orient	[0.96593,0,0.25882,0]	“[0.96593,0,0.25882,0]”

通信指令- TPReadFK

TPReadFK Answer,Text,FK1,FK2,FK3,FK4,FK5,
[\MaxTime][\DIBreak][\BreakFlag];

Answer :	数字赋值1-5。	**(num)**
Text :	屏幕字符串。	**(string)**
FKx:	功能键字符串。	**(string)**
[\MaxTime] :	最长等带时间	**(num)**
[\DIBreak] :	输入信号控制	**(signaldi)**
[\BreakFlag] :	指令状态控制	**(errnum)**

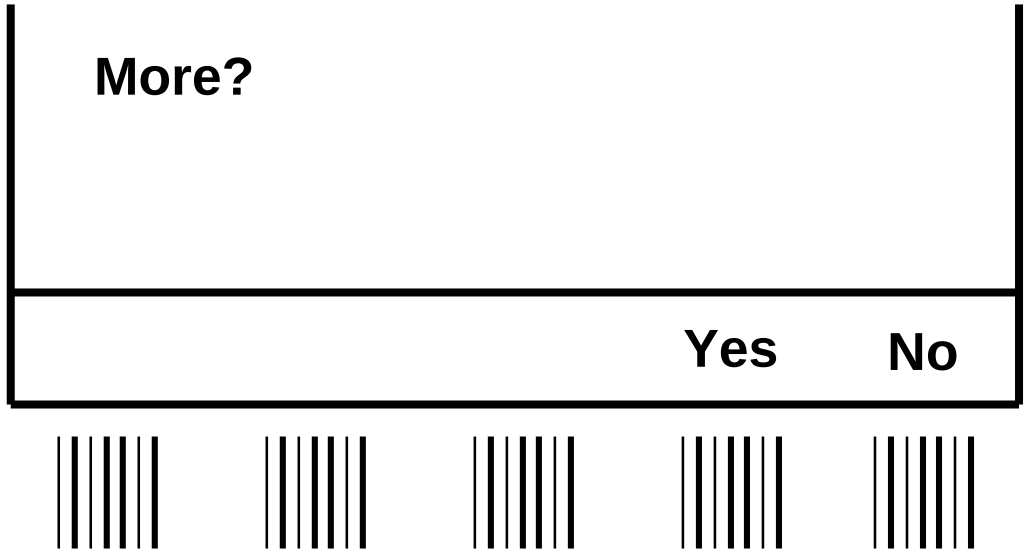
通信指令- **TPReadFK**

应用：
在示教器屏幕上显示相应字符串（Text),字符串最长80个字节，屏幕每行可显示40个字节，同时在5个功能键上显示相应字符串(FKx)，字符串最长7个字节，通过选择按相应的功能键，给数字变量（Answer)赋值1-5，通过这种功能，当前指令可以进行数据选择，但必须有人参与，无法达到自动化，已被输入输出信号替代，另外，在执行以后指令，除非选择相应参变量。通过这种功能，当前指令常用于错误处理等场合。

通信指令- TPReadFK

实例：

TPReadFK reg1,“More?”,stEmpty,stempty,“Yes”, “No”;



通信指令- **TPReadFK**

参变量：

[\MaxTime]
机器人执行当前指令等待时间超过最长等待时间，机器人将停机报错，如果同时采用参变量[BreakFlag],机器人将继续执行以后指令，并且给出相应错误数据。
[\DIBREAK]
机器人通过输入信号来继续执行以后指令，并且给出相应错误数据。
[\BreakFlag]
-ERR_TP_MAXTIME
-ERR_TP_DIBREAK

通信指令- TPReadNum

TPReadNum Answer,String,
[\MaxTime][\DIBreak][\BreakFlag];

Answer :	数字赋值。	**(num)**
String :	屏幕字符串。	**(string)**
[\MaxTime] :	最长等带时间	**(num)**
[\DIBreak] :	输入信号控制	**(signaldi)**
[\BreakFlag] :	指令状态控制	**(errnum)**

通信指令- **TPReadNum**

应用：

在示教器屏幕上显示相应字符串（String),字符串最长80个字节，屏幕每行可显示40个字节，同时在在功能键上显示OK,通过数字键输入相应数值，给数字变量（Answer)赋值，通过这种功能，当前指令可以进行数字数据赋值，但必须有人参与，无法达到自动化，已被输入输出信号替代。

实例：

```
TPReaRun reg1,“How many units ?”,
FOR i FROM 1 TO reg1 DO
    produce_part;
```

通信指令- **TPReadNum**

参变量：

[\MaxTime]
机器人执行当前指令等待时间超过最长等待时间，机器人将停机报错，如果同时采用参变量[BreakFlag],机器人将继续执行以后指令，并且给出相应错误数据。
[\DIBREAK]
机器人通过输入信号来继续执行以后指令，并且给出相应错误数据。
[\BreakFlag]
-ERR_TP_MAXTIME
-ERR_TP_DIBREAK

通信指令- ErrWrite

ErrWrite [\w],Header,Reason[\RL2] [\RL3] [\RL4];

[\w] :	事件记录开关。	**(switch)**
Header :	错误信息标题。	**(string)**
Reason :	错误信息原因。	**(string)**
[\RL2] :	附加错误信息原因	**(string)**
[\RL2] :	附加错误信息原因	**(string)**
[\RL2] :	附加错误信息原因	**(string)**

通信指令- **ErrWrite**

应用：
在示教器屏幕上显示标准出错界面，错误代码为80001，标题最长24个字符，原因最长40个字符，如果由多种错误原因，可以使用参变量[\RL2] [\RL2] [\RL2],每种原因最长40个字符，使用参变量[\W]，错误代码为80002，并且只在事件清单中记录，不在示教器屏幕上显示。当前指令只显示或记录出错误信息，需要按功能键OK确认并清除，如需影响机器人运行，使用指令Stop,EXIT,TPReadFK等。

通信指令- **ErrWrite**

实例：

```
...
ErrWrite\W,“search error”,“No hit for the first search”;
ErrWrite “PLC error”, “Fatal error in PLC”₩RL2:=“Call service”;
Stop;
...
```

限制：

每个ErrWrite指令最多能显示145个字节。
（Header+Reason+\RL2+ \ RL3+ \ RL4)

通信指令- **TPShow**

TPShow Window;

Window:显示相应示教器窗口。（**tpnum)**

TP_PROGRAM
自动模式下，显示生产窗口。
手动模式下，显示测试窗口
TP_LATEST
显示当前窗口的前一个窗口。
TP_SCREENVIEWER
显示Screen Viewer窗口，需要相应软件。

ABB

通信指令- TPShow

应用：

机器人在示教器屏幕显示界面，通常情况下与机器人附加软件ScreenViewer配合使用。

实例：

```
...
TPShow TP_PROGRAM;
TPShow TP_LATEST;
...
```

运动指令

MoveJ

MoveL

MoveC

MoveJDO

MoveLDO

MoveCDO

MoveJSync

MoveLSync

MoveCSync

MoveAbsJ

运动指令- MoveJ

MoveJ [\Conc,] ToPoint,Speed [\V]| [\T],Zone [\Z] [\Inpos],Tool [\Wobj];

[\Conc] : 协作运动开关。 （**switch)**

ToPoint : 目标点，默认为* （**robotarget)**

Speed : 动行速度数据。 （**speeddata)**

[\V] : 特殊运行速度mm/s （**unm)**

[\T] : 运行时间控制s （**unm)**

运动指令- MoveJ

MoveJ [\Conc,] ToPoint,Speed [\V]| [\T],Zone [\Z] [\Inpos],Tool [\Wobj];

Zone :	动行转角数据。	**（zonedata)**
[\Z] :	特殊运行转角mm	**（num)**
[\Inpos] :	动行停止点数据。	**（stoppointdata)**
Tool :	工具中心点（TCP）	**（tooldata)**
[\Wobj] :	工件座标系	**（wobjadta)**

运动指令- **MoveJ**

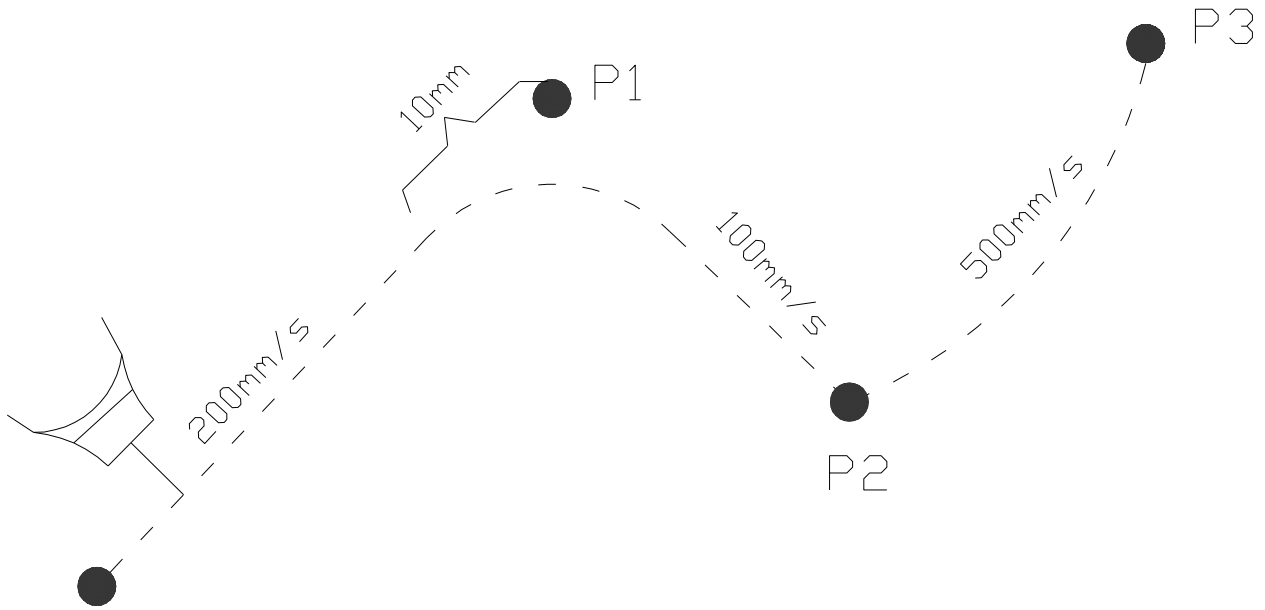
应用：

机器人以最快捷的方式运动至目标点，机器人运动状态不完全可控，但运动路径保持唯一，常用于机器人在空间大范围移动。

实例：

```
MoveJ p1,v2000,fine,grip1;
MoveJ\Conc,p1,v2000,fine,grip1;
MoveJ p1,v2000\V:=2200,z40\z:45,grip1;
MoveJ p1,v2000,z40,grip1\Wobj:=wobjTable;
MoveJ p1,v2000,fine\Inpos:=inpos50,grip1;
```

运动指令- **MoveJ**



MoveL p1,v200,z10,tool1
MoveL p2,v100,fine,tool1
MoveJ p3,v500,fine,tool1

运动指令- MoveL

MoveL [\Conc,] ToPoint,Speed [\V]| [\T],Zone [\Z] [\Inpos],Tool [\Wobj] [\Corr];

[\Conc] : 协作运动开关。 （**switch)**

ToPoint : 目标点，默认为* （**robotarget)**

Speed : 动行速度数据。 （**speeddata)**

[\V] : 特殊运行速度mm/s （**unm)**

[\T] : 运行时间控制s （**unm)**

运动指令- MoveL

MoveL [\Conc,] ToPoint,Speed [\V]| [\T],Zone [\Z] [\Inpos],Tool [\Wobj] [\Corr];

Zone :	动行转角数据。	（**zonedata)**
[\Z] :	特殊运行转角mm	（**num)**
[\Inpos] :	动行停止点数据。	（**stoppointdata)**
Tool :	工具中心点（TCP）	（**tooldata)**
[\Wobj] :	工件座标系	（**wobjadta)**
[\Corr] :	修正目标点开关	（**switch)**

运动指令- **MoveL**

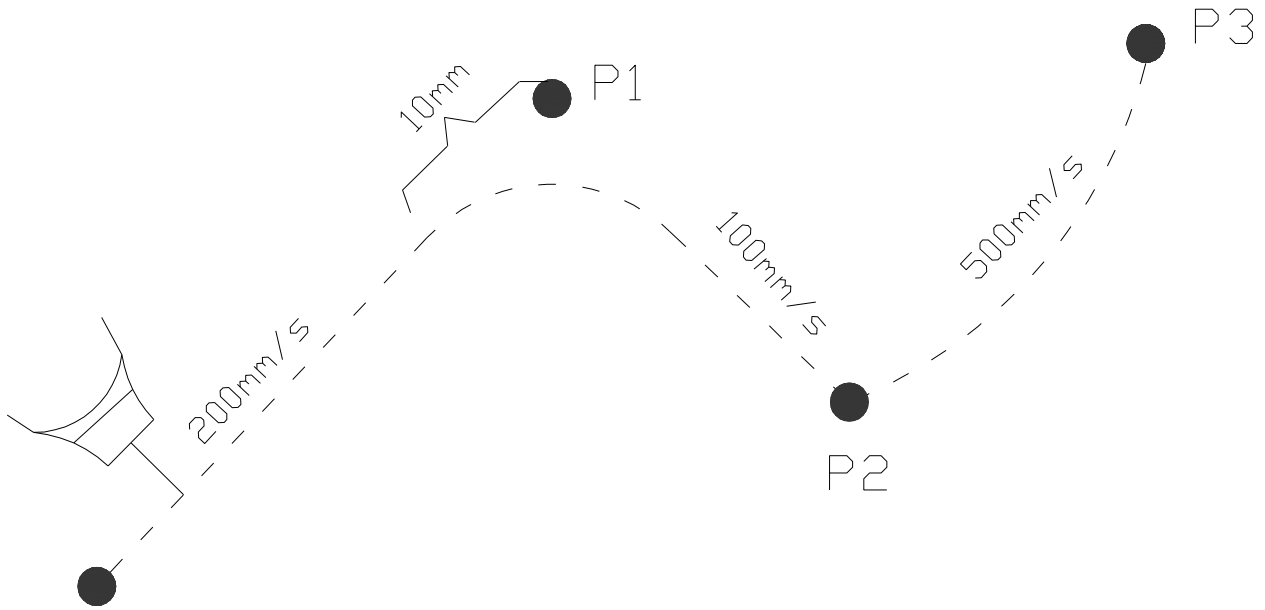
应用：

机器人以线性方式运动至目标点，当前点与目标点两点决定一条直线，机器人运动状态可控，运动路径保持唯一，可能出现死点，常用于机器人在工作状态移动。

实例：

```
MoveL p1,v2000,fine,grip1;
MoveL\Conc,p1,v2000,fine,grip1;
MoveL p1,v2000\V:=2200,z40\z:45,grip1;
MoveL p1,v2000,z40,grip1\Wobj:=wobjTable;
MoveL p1,v2000,fine\Inpos:=inpos50,grip1;
MoveL p1,v2000,fine,grip1\corr;
```

运动指令- **MoveL**




```
MoveL p1,v200,z10,tool1
MoveL p2,v100,fine,tool1
MoveJ p3,v500,fine,tool1
```

运动指令- MoveC

MoveC [\Conc,] CirPoint,ToPoint,Speed [\V]| [\T],Zone [\Z] [\Inpos],Tool [\Wobj] [\Corr];

[\Conc] :	协作运动开关。	**（switch)**
CirPoint :	中间点，默认为*	**（robotarget)**
ToPoint :	目标点，默认为*	**（robotarget)**
Speed :	动行速度数据。	**（speeddata)**
[\V] :	特殊运行速度mm/s	**（unm)**
[\T] :	运行时间控制s	**（unm)**

运动指令- MoveC

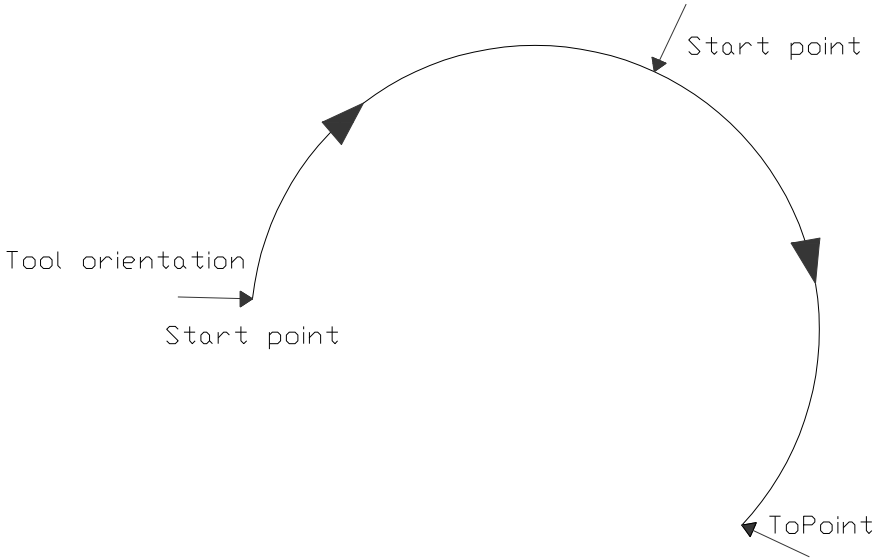
MoveC [\Conc,] CirPoint, ToPoint,Speed [\V]| [\T],Zone [\Z] [\Inpos],Tool [\Wobj] [\Corr];

Zone :	动行转角数据。	**（zonedata)**
[\Z] :	特殊运行转角mm	**（num)**
[\Inpos] :	动行停止点数据。	**（stoppointdata)**
Tool :	工具中心点（TCP）	**（tooldata)**
[\Wobj] :	工件座标系	**（wobjadta)**
[\Corr] :	修正目标点开关	**（switch)**

运动指令- **MoveC**

应用：

机器人通过中心点以圆弧移动方式动动至目标点，当前点.中间点与目标点三点决定一断圆弧，机器人运动状态可控，运动路径保持唯一，常用于机器人在工作状态移动。



运动指令- **MoveC**

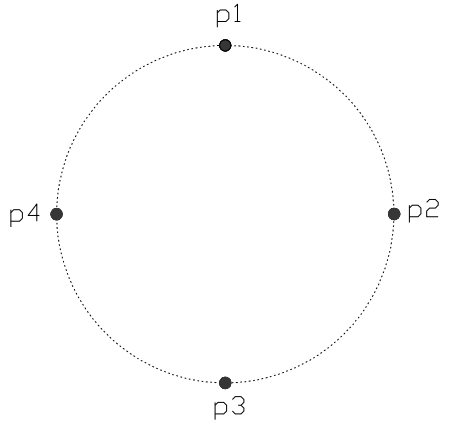
实例：

```
MoveC p1,p2,v2000,fine,grip1;
MoveC\Conc,p1,p2,v200,\v:=500,z1\z:=5,grip1;
MoveC p1,p2,v2000,z40,grip1\Wobj:=wobjTable;
MoveC p1,p2,v2000,fine\Inpos:=50,grip1
MoveC p1,p2,v2000,fine,grip1\corr;
```

限制：

不可能通过一个MoveC指令完成一个圆。

```
MoveL p1,v500,fine,tool1
Movec p2,p3,v500,z20,tool1
Movec p4,p1,v500,fine,tool1
```



运动指令- MoveJDO

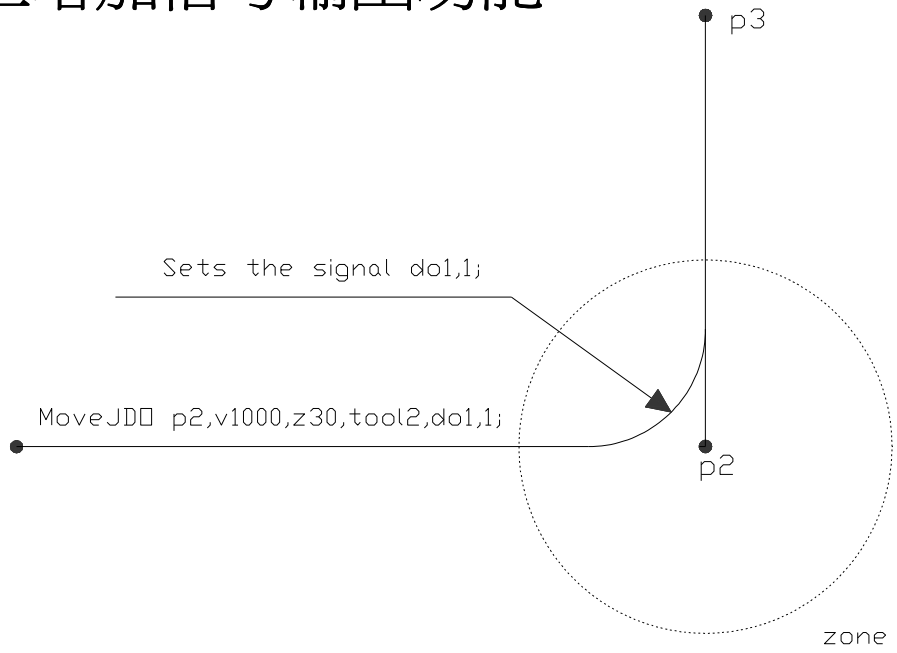
MoveDO ToPoint,Speed [\T],Zone,Tool [\Wobj], Signal,Value;

ToPoint :	目标点，默认为*	（**robotarget)**
Speed :	动行速度数据。	（**speeddata)**
[\T] :	运行时间控制s	（**unm)**
Zone :	运行转角数据	（**zonedata)**
Tool :	工具中心点（TCP)	（**tooldata)**
[\Wobj] :	工件座标系	（**wlbjdata)**
Signal :	数字输出信号名称	（**signaldo)**
Value :	数字输出信号值	（**dionum)**

运动指令- MoveJDO

应用：

机器人以最快捷的方式运动至目标点，并且在目标点将相应输出信号设置为相应值，在指令MoveJ基础上增加信号输出功能。



运动指令- MoveLDO

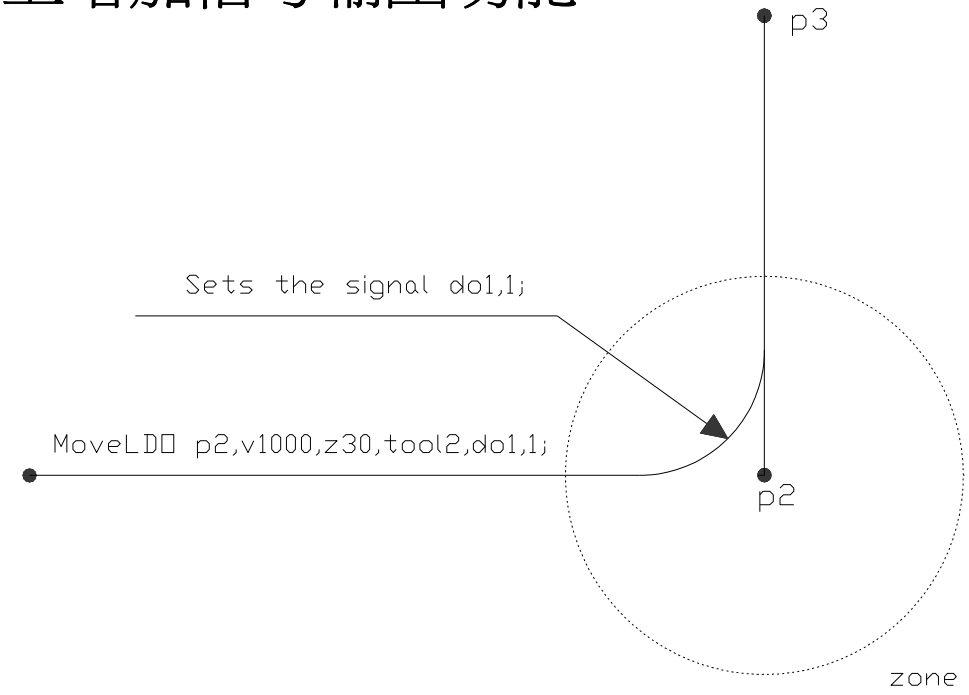
MoveLDO ToPoint,Speed [\T],Zone,Tool [\Wobj], Signal,Value;

ToPoint :	目标点，默认为*	（**robotarget)**
Speed :	动行速度数据。	（**speeddata)**
[\T] :	运行时间控制s	（**unm)**
Zone :	运行转角数据	（**zonedata)**
Tool :	工具中心点（TCP)	（**tooldata)**
[\Wobj] :	工件座标系	（**wlbjdata)**
Signal :	数字输出信号名称	（**signaldo)**
Value :	数字输出信号值	（**dionum)**

运动指令- MoveLDO

应用：

机器人以线性运动方式运动至目标点，并且在目标点将相应输出信号设置为相应值，在指令MoveL基础上增加信号输出功能。



运动指令- MoveCDO

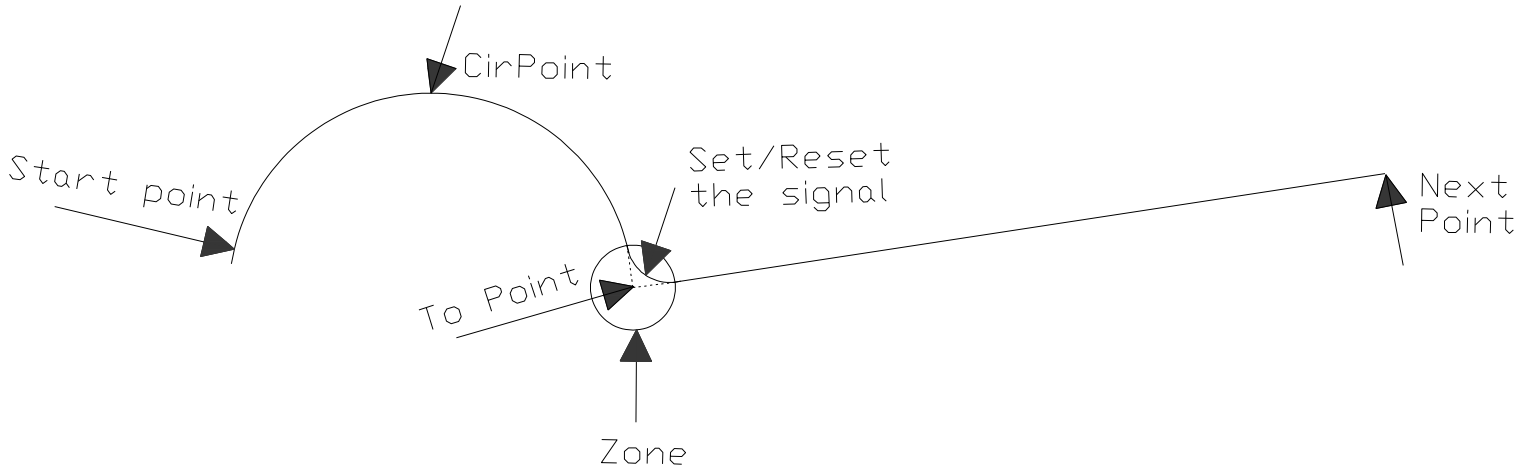
MoveCDO CiPoint,ToPoint,Speed [\T],Zone,Tool [\Wobj], Signal,Value;

CiPoint :	中间点，默认为*	（**robotarget)**
ToPoint :	目标点，默认为*	（**robotarget)**
Speed :	动行速度数据	（**speeddata)**
[\T] :	运行时间控制s	（**unm)**
Zone :	运行转角数据	（**zonedata)**
Tool :	工具中心点（TCP)	（**tooldata)**
[\Wobj] :	工件座标系	（**wlbjdata)**
Signal :	数字输出信号名称	（**signaldo)**
Value :	数字输出信号值	（**dionum)**

运动指令- MoveCDO

应用：

机器人通过中间点以圆弧移动方式运动至目标点，并且在目标点将相应输出信号设置为相应值，在指令MoveC基础上增加信号输出功能。



运动指令- MoveJSync

MoveJSync ToPoint,Speed [\T], Tool Zone,[\Wobj], Proc;

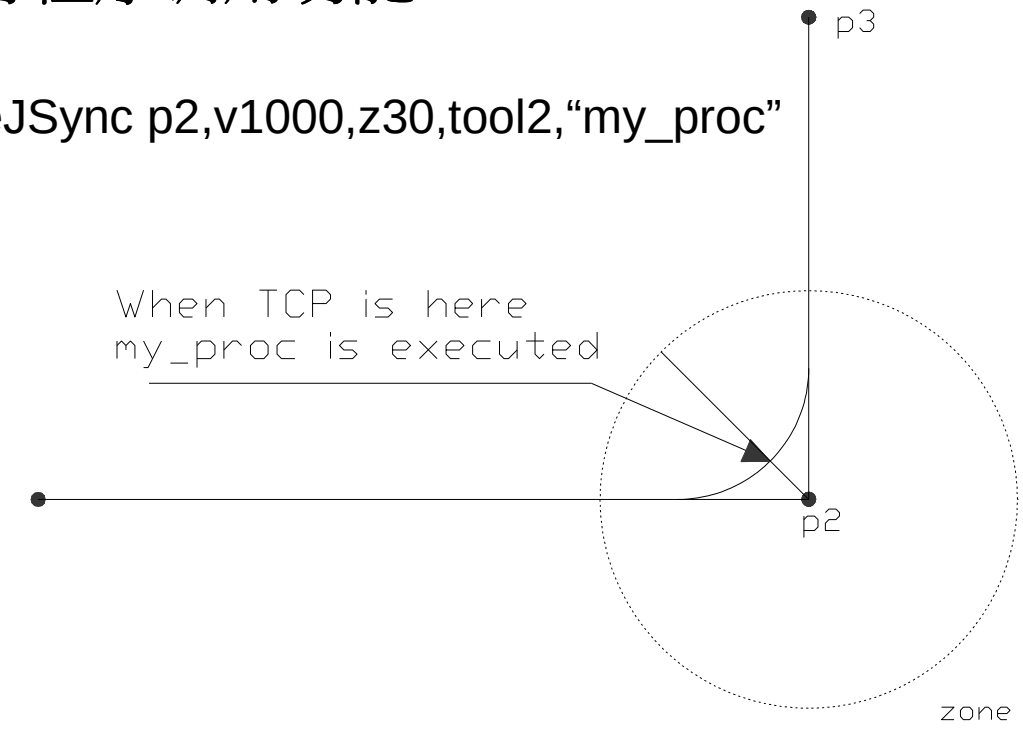
ToPoint :	目标点，默认为*	（**robotarget)**
Speed :	动行速度数据	（**speeddata)**
[\T] :	运行时间控制s	（**unm)**
Zone :	运行转角数据	（**zonedata)**
Tool :	工具中心点（TCP)	（**tooldata)**
[\Wobj] :	工件座标系	（**wlbjdata)**
Proc:	例行程序名称	（**string)**

运动指令- **MoveJSync**

应用：

机器人以最快捷的方式运动至目标点，并且在目标点调用相应的例行程序，在指令MoveJ基础上增加例行程序调用功能

MoveJSync p2,v1000,z30,tool2,“my_proc”



运动指令- **MoveJSync**

限制：

用指令Stop停止当前指令运行，会出现一个错误信息，如需要避免，采用指令StopInstr.

不能使用指令MoveJSync来调用中断处理程序TRAP.

不能单步执行指令MoveJSync所调用的例行程序PROC。

运动指令- MoveLSync

MoveLSync ToPoint,Speed [\T], Tool Zone,[\Wobj], Proc;

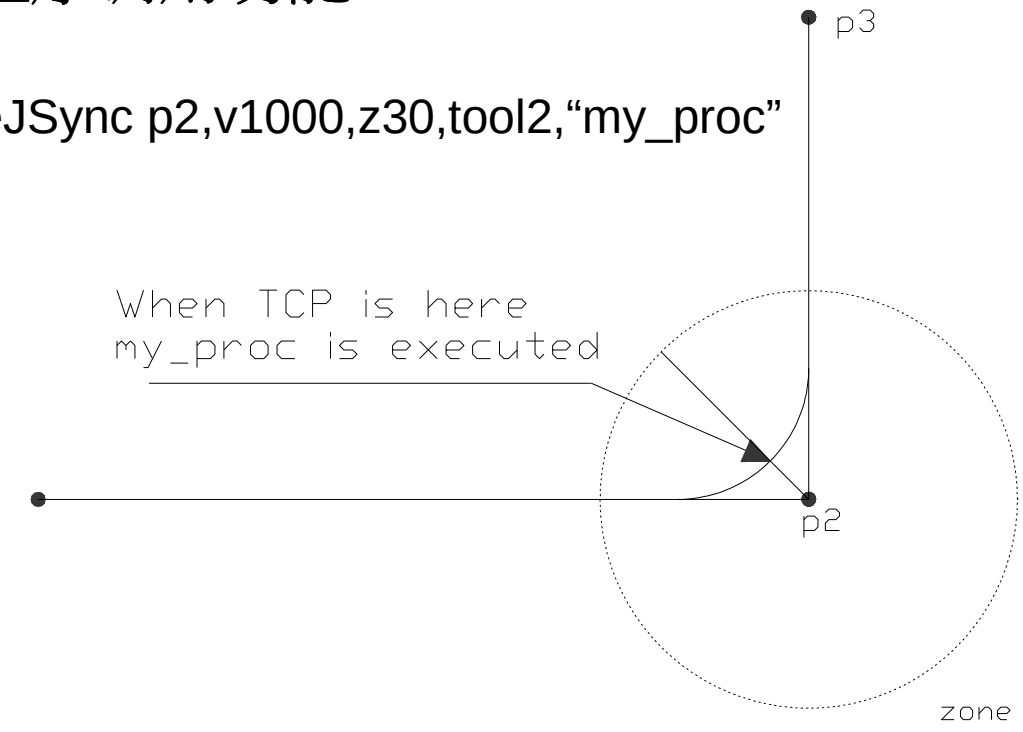
ToPoint :	目标点，默认为*	（**robotarget)**
Speed :	动行速度数据	（**speeddata)**
[\T] :	运行时间控制s	（**unm)**
Zone :	运行转角数据	（**zonedata)**
Tool :	工具中心点（TCP)	（**tooldata)**
[\Wobj] :	工件座标系	（**wlbjdata)**
Proc:	例行程序名称	（**string)**

运动指令- **MoveLSync**

应用：

机器人以线性的方式运动至目标点，并且在目标点调用相应的例行程序，在指令MoveL基础上增加例行程序调用功能

MoveJSync p2,v1000,z30,tool2,“my_proc”



运动指令- **MoveLSync**

限制：

用指令Stop停止当前指令运行，会出现一个错误信息，如需要避免，采用指令StopInstr.

不能使用指令MoveLSync来调用中断处理程序TRAP.

不能单步执行指令MoveLsync所调用的例行程序PROC。

运动指令- MoveCSync

MoveCSync Cirpoint,ToPoint,Speed [\T], Zooe,Tool [\Wobj], Proc;

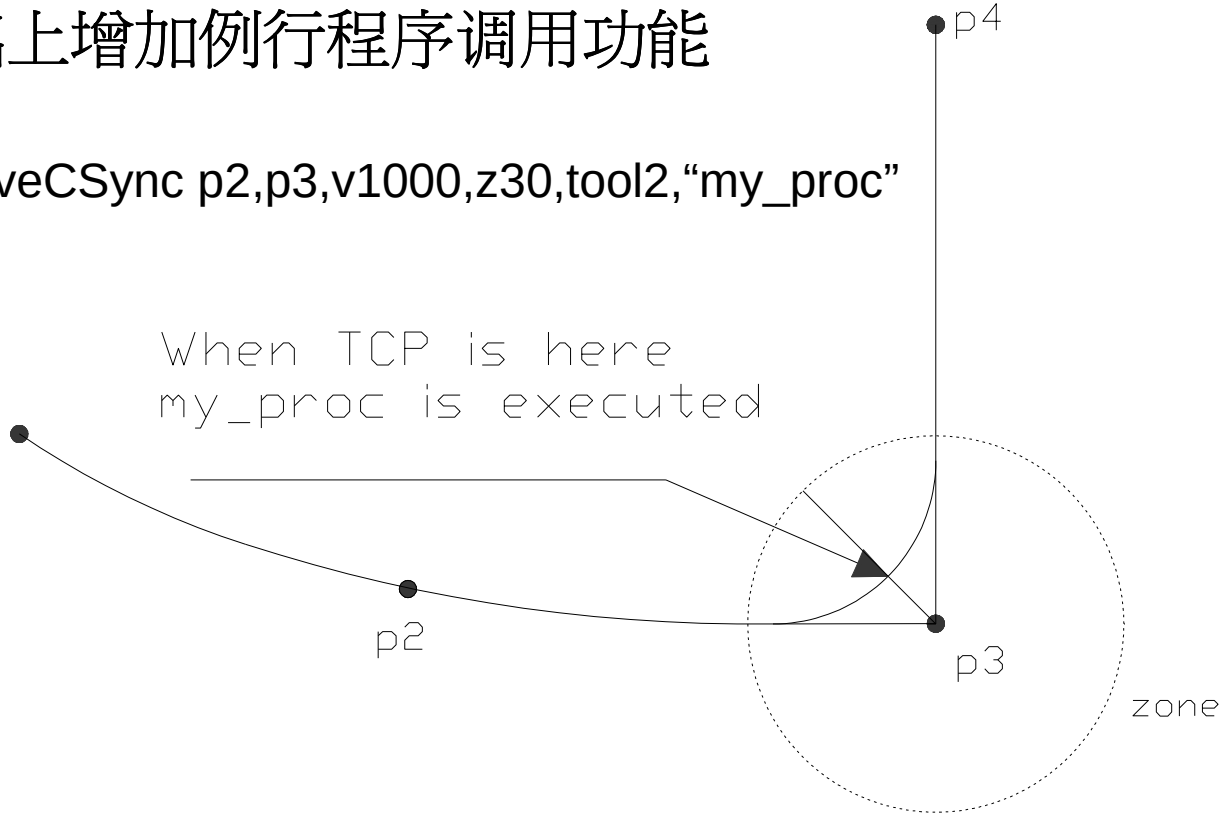
CiPoint :	中间点，默认为*	（**robotarget)**
ToPoint :	目标点，默认为*	（**robotarget)**
Speed :	动行速度数据	（**speeddata)**
[\T] :	运行时间控制s	（**unm)**
Zone :	运行转角数据	（**zonedata)**
Tool :	工具中心点（TCP)	（**tooldata)**
[\Wobj] :	工件座标系	（**wlbjdata)**
Proc:	例行程序名称	（**string)**

运动指令- **MoveCSync**

应用：

机器人通过中间点以圆弧移动方式至运动至目标点，并且在目标点调用相应的例行程序，在指令MoveC基础上增加例行程序调用功能

MoveCSync p2,p3,v1000,z30,tool2,“my_proc”



运动指令- **MoveCSync**

限制：

用指令Stop停止当前指令运行，会出现一个错误信息，如需要避免，采用指令StopInstr.

不能使用指令MoveCSync来调用中断处理程序TRAP.

不能单步执行指令MoveCsync所调用的例行程序PROC。

运动指令- MoveAbsJ

MoveAbsJ [\Conc,] ToJointPos [\NoEoffs],Speed [\V][\T], Zooe [\Z] [\InPOS],Tool [\Wobj];

[\Conc] :	协作运动开关	**（switch)**
ToJointPos :	目标点	**（jointtarget)**
[\NoEoffs] :	外轴偏差开关	**（switch)**
Speed :	动行速度数据	**（speeddata)**
[\V] :	特殊运行速度	**（num)**
[\V] :	运行时间控制	**（num)**

运动指令- MoveAbsJ

MoveAbsJ [\Conc,] ToJointPos [\NoEoffs],Speed [\V]|[\T], Zooe [\Z] [\InPOS],Tool [\Wobj];

Zooe :	运行转角数据	**（zonedata)**
[\Z] :	特殊运行转角mm	**（unm)**
[\InPOS] :	运行停止点数据	**（switch)**
Tool :	工具中心点TCP	**（tooldata)**
[\Wobj] :	工件座标系	**（wobjdata)**

运动指令- MoveAbsj

应用：

机器人以单轴运行的方式运动至目标点，绝对不存在死点，运动状态完全不可控，避免在正常生产中使用此指令，常用于检查机器人零点位置，指令中TCP与Wobj只与运行速度有关，与运动位置无关。

实例：

```
MoveAbsJ p1,v2000,fine,grip1;
MoveAbsJ\Conc,p1\NoEoffs,v2000,fine,grip1;
MoveAbsJ p1,v2000\v:=2200,z40\z:=45,grip1;
MoveAbsJ p1,v2000,z40,grip1\Wobj:=Wobj1;
MoveAbsJ p1,v2000,fine\Inpos:=inpos50,grip1;
```

中断运动指令

StopMove

StartMove

StorePath

RestoPath

中断运动指令-StopMove

应用：

当前指令使机器人运动临时停止，直到运行指令StartMove后，才继续恢复被临时停止的运动，此指令通常被用于处理牵涉到机器人运动的中断程序。

实例：

```
StopMove;
WaitDI ready_input,1;
StartMome;
```


ABB

中断运动指令-StopMove

实例：

```
…
  CONNECT intno1 WITH go_to_home_pos;
  ISignalDI di1,1,intno1;
…
TRAP go_to_home_pos

  Storepath;
  P10:=Crobt();
  MoveL Home,v500,fine,tool1;
  WaitDI di1,0;
  MoveL p10,v500,fine,tool1;
  Restopath;
  StartMove;
ENDTRAP
```

机器人完成当前运动指令后停止运动，并记录运动路径，在Home位置等待di1为0后，继续原运动状态。

机器人临时停止运行，并记录运动路径，在Home位置等待di1为0后继续原运动状态。

中断运动指令-StartMove

应用：

当前指令必须与StopMove联合使用，使机器人临时停止运动的恢复 ，此指令通常被用于处理牵涉到机器人运动的中断程序。

实例：

```
StopMove;
WaitDI ready_input,1;
StartMome;
```

Error Handling：

ERR_PATHDIST
偏离原路径（大于10mm或20度 ）

中断运动指令-StorePath

StorePath;

应用：

当前指令用来记录机器人当前运动状态，通常与指令RestoPath联合使用。此指令通常被用于机器人故障处理与处理牵涉到机器人运动的中断程序。

限制：

当前指令只能用来记录机器人运动路经。
机器人临时停止后，需要执行新的运动，必须记录当前运动路径。
机器人系统只能记录一个运动路径。

中断运动指令-StorePath

实例：

```
TRAP go_to_home_pos
   StopMove;
   StorePath;
   p10:=CRobT();
   MoveL Home,v500,fine,tool1;
   WaitDI di1,0;
   MoveL p10,v500,fine,tool1;
   RestoPath;
   StartMove;
ENDTRAP
```

机器人临时停止运动，并记录运动路径，在Home位置等待di1为0后，继续原运动状态。

ABB

中断运动指令-RestoPath

RestoPath;

应用：

当前指令用来恢复机器人当前运动状态，必须与指令StorePath联合使用。此指令通常被用于机器人错误处理与处理牵涉到机器人运动的中断程序。

限制：

当前指令只能用来恢复机器人运动路经。
机器人临时停止后，需要执行新的运动，必须记录当前运动路径。
机器人系统只能记录一个运动路径。

中断运动指令-RestoPath

实例：

```
ArcL p100,v100,seam1,weld5,weave1,z10,gum1;
ERROR
    IF ERRNO=AW_WELD_ERR THEN
        gun_cleaning;
        RETRY;
    ENDIF
PROC gun_cleaning()
    StorePath;
    p1:=CRobT();
    MoveL pclean,v100,fine,gun1;
    …
    MoveL p1,v100,fine,gun1;
    RestoPath;
ENDPROC
```

程序流程指令

IF

TEST

GOTO

label

WHILE

FOR

WaitUntil

WaitTime

Compact IF

程序流程指令-IF

IF Condition THEN…
{ELSEIF Condition THEN…}
[ELST…]
ENDIF

Condition: 判断条件 **(bool)**

应用：
当前指令通过判断相应条件，控制需要执行的相应指令，是机器人程序流程基本指令。

程序流程指令-IF

实例：

```
IF reg1>5 THEN
   set do1;
   set do2;
ENDIF

IF reg1>5 THEN
   set do1;
   set do2;
ELSE
   Reset do1;
   Reset do2;
ENDIF
```

```
IF reg2=1 THEN
  routine1;
ELSEIF reg2=2 THEN
  routine2;
ELSEIF reg2=3 THEN
  routine3;
ELSEIF reg2=4 THEN
  routine4;
ELSE
  Error;
ENDIF
```

程序流程指令-TEST

TEST Test data
{CASE Test value {,Test value}:…}
[DEFAULT:…]
ENDTEST

Test data: 判断数据变量 **(all)**

Test value: 判断数据值 **(Same as)**

应用：
当前指令通过判断相应数据变量与其所对应的值，控制需要执行的相应指令，

程序流程指令-TEST

实例：

```
TEST reg2
CASE 1:
    routine1;
CASE 2:
    routine2;
CASE 3:
    routine3;
CASE 4,5:
    routine9;
DEFAULT:
    Error;
ENDTEST
```

```
IF reg2=1 THEN
  routine1;
ELSEIF reg2=2 THEN
  routine2;
ELSEIF reg2=3 THEN
  routine3;
ELSEIF reg2=4 OR reg2=5 THEN
  routine4;
ELSE
  Error;
ENDIF
```

程序流程指令-GOTO

GOTO Label:

Label: 程序执行位置标签 **(identifier)**

应用：
当前指令必须与指令Label同时例用，执行当前指令后，机器人将从相应标签位置Label处继续运行程序指令。

程序流程指令-GOTO

实例：

```
IF reg1>100 GOTO highvalue;
Lowvalue;
…
GOTO ready;
Highvalue:
…
Reg1:=1;
next：
Reg1:=reg1+1
IF reg1<=5 GOTO next;
```

程序流程指令-GOTO

限制：

只能使用当前指令跳跃至同一例行程序内相应位置标签Label.

如果相应位置标签label处于指令TEST或IF内，相应指令GOTO必须同处于相同的判断指令内或其分支内。

如果相应位置标签Label处于指令WHILE或FOR内，相应指令GOTO必须同处于相同的循环指令内。

程序流程指令-label

Label:

Label: 程序执行位置标签 **(identifier)**

应用：
当前指令必须与指令GOTO同时例用，执行当前指令GOTO后，机器人将从相应标签位置Label处继续运行程序指令,当前指令使用后，程序内不会显示Label字样，直接显示相应标签。

ABB

程序流程指令- **label**

实例：

```
IF reg1>100 GOTO highvalue;
Lowvalue;
…
GOTO ready;
Highvalue:
…
Ready:
…
```

限制：

在同一个例行程序内，程序位置标签Label的名称必须唯一。

程序流程指令-WHILE

WHILE Condition DO
…
ENDWHILE

Condition: 判断条件 **(bool)**

应用：
当前指令通这判断相应条件，如果符合判断条件执行循环内指令，直至判断条件不满足才跳出循环，继续执行循环以后指令，需要注意，当前指令存在死循环。

程序流程指令- **WHILE**

实例：

```
WHILE reg1<reg2 DO
    …
    reg1:=reg1+1;
ENDWHILE
PROC main()
    rInitial;
    WHILE TRUE DO
        …
    ENDWHILE
ENDPROC
```

程序流程指令-FOR

```
FOR Loop counter FROM
Start value TO End value
[STEP Step value] DO
...
ENDFOR
```

Loop counter： 循环计数标识 (identifier)

Start value： 标识初始值 (num)

End value： 标识最终值 (num)

[STEP Step value]： 计数更改值 (num)

程序流程指令- FOR

应用：

当前指令通过循环判断标识从初始值逐渐更改最终值，从而控制程序相应循环次数，如果不使用参变量[STEP]，循环标识每次更改值为1，如果例用参变量[STEP],循环标识每次更改值为参变量相应设置，通常情况下，初始值.最终值与更改值为整数，循环判断标识使用i k j等小写字母，是标准的机器人循环指令，常在通讯口读写，数组数据赋值等数据处理时例用。

ABB

程序流程指令- **FOR**

实例：

```
FOR i FROM 1 TO 10 DO
    routine1;
ENDFOR
FOR i FROM 10 TO 2 STEP –1 DO
    a{i}:=a{i -1};
ENDFOR
PROC ResetCount()
    FOR i FROM 1 TO 20 DO
        FOR j FORM 1 TO 2 DO
            nCount{i,j}:=0
        ENDFOR
    ENDFOR
ENDPROC
```


ABB

程序流程指令- FOR

限制：

循环标识只能自动更改，不允许赋值。
在程序循环内，循环标识可以作为数字数据（num)使用，但只能读取相应值，不允许赋值。
如果循环标识.初始值.最终值与更改值使用小数形式，必须为精确值。

程序流程指令-WaitUntil

WaitUntil [\Inpos,]Cond [\MaxTime][\TimeFlag];

[\Inpos]： 提前量开关 (switch)

Cond： 判断条件 (bool)

[\MaxTime]： 最长等待时间 (num)

[\TimeFlag]: 超时逻辑量 (bool)

程序流程指令- **WaitUntil**

应用：

当前指令用于等待满足相应判断条件后，才执行以后指令，使用参变量[\InPOS],机器人及其外轴必须在完全停止的情况下，才进行条件判断，此指令比可以指令WaitDI的功能更广，可以替代其所有功能。

限制：

当前指令在使用参变量[\InPOS]时，遇到程序突然停止运行，机器人不能其保证停在最终停止点进行条件判断。

程序流程指令- **WaitUntil**

实例：

```
PROC PickPart()
    MoveJ pPrePick,vFastEmpty,zBig,tool1;
    bTimeout:=TRUE;
    nCounter:=0;
    WHILE bTimeout DO
        IF nCounter>3 THEN
            TPWrite "……";
        ENDIF
        IF nCounter>30 THEN
           stop;
        ENDIF
        WaitUntil bok=TRUE\MaxTime:=1\TimeFlag:=bTimeout;
        Incr nCounter;
    ENDWHILE
    …
ENDPROC
```

机器人等待相应逻辑值，如果1秒内仍没有得到相应值，机器人自动执行随后指令，但此时TimeFlag值为TRUE;机器人得到相应逻辑值，TimeFlag值为FALSE

程序流程指令- **WaitUntil**

实例：

```
PROC PickPart()
    MoveJ pPrePick,vFastEmpty,zBig,tool1;
    WaitUntil di_Ready=1;
    (WaitDI di_Ready,1;)
…
ENDPROC
PROC PickPart()
    MoveJ pPrePick,vFastEmpty,zBig,tool1;
    WaitUntil nCounter=4\MaxTime:=5;
    …
ERROR
    IF ERRNO+ERR_WAIT_MAXTIME THEN
       TPWrite “……”;
       RETRY;
    ENDIF
ENDPROC
```

机器人等待输入信号，直到信号di_Ready值为1，才执行随后指令。

机器人等待相应数据输入，如果5秒内仍没有得到相应数据值为4，自动进行Error Handler处进，如果没有Error Handler,机器人停止报错。

程序流程指令-WaitTime

WaitTime [\Inpos,] Time;

[\Inpos]：　　程序提前量开关（switch)

Time：　　相应等待时间 **S.**（num)

应用：

当前指令只用于机器人等待相应时间后，才执行以后指令，使用参变量[\Inpos],机器人及其外轴必须在完全停止的情况下，才进行等待时间计时，此指令会延长循环时间。

程序流程指令-WaitTime

实例：

```
WaitTime 3;
WaitTime\InPos,0.5;
WaitTime\InPos,0;
```

限制：

当前指令在使用参变量[\InPos]时，遇到程序突然停止运行，机器人不能保证停止在最终停止点进行等待计时，

当前指令参变量 [\InPos]不能与机器人指令SoftServo同进使用。

程序流程指令-Compact IF

IF Condition…

Condition：　　判断条件　　(bool)

应用：

当前指令是指令IF的简单化，判断条件后只允许跟一名指令，如果有多句指令需要执行，必须采用指令IF.

实例：

```
IF reg1>5 GOTO next;
IF counter>10 set do1;
```

故障处理指令

RETRY

TRYNEXT

RAISE

RETURN

故障处理指令

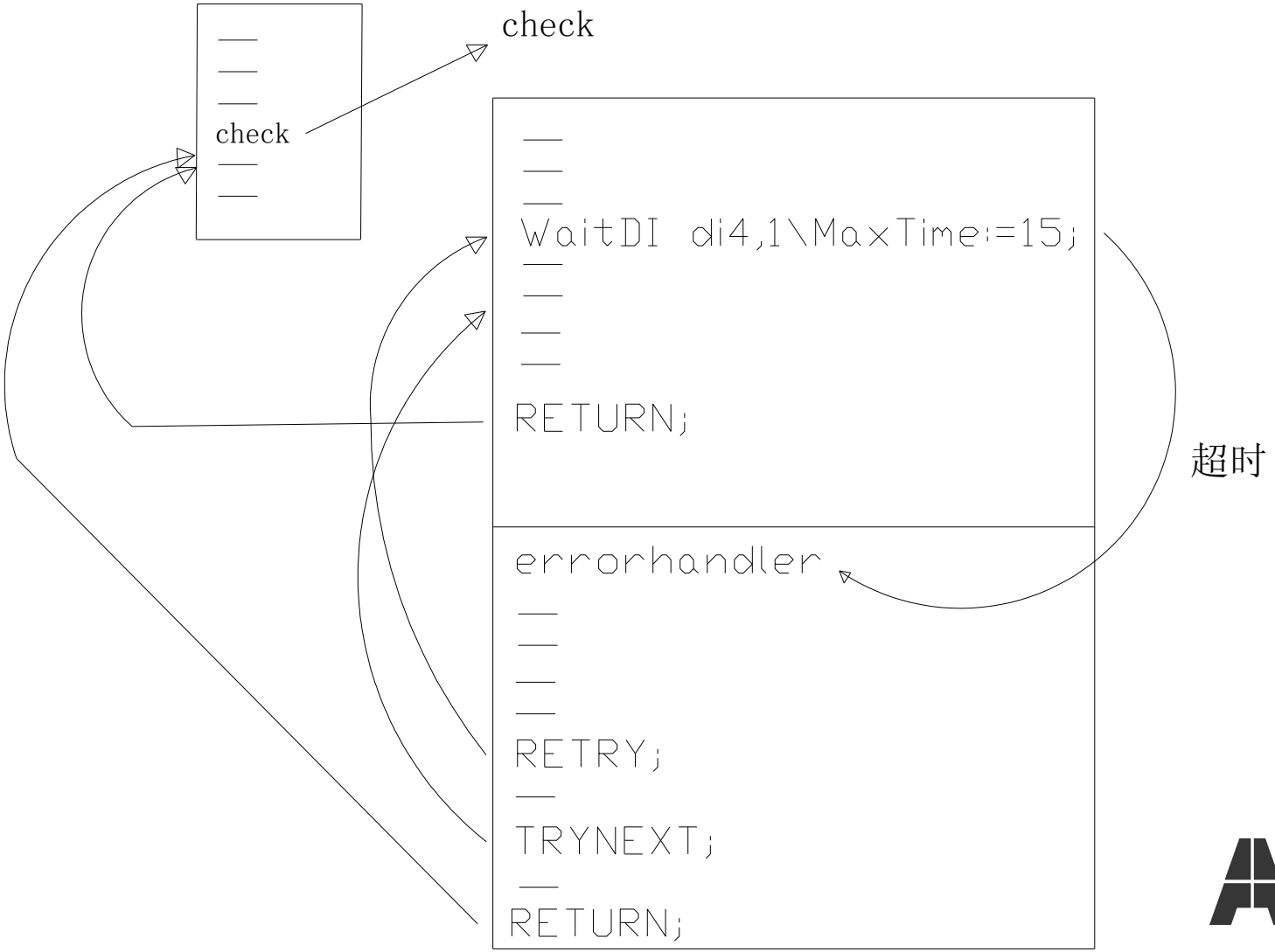
RETRY

TRYNEXT

RAISE

RETURN

故障处理指令- RETRY



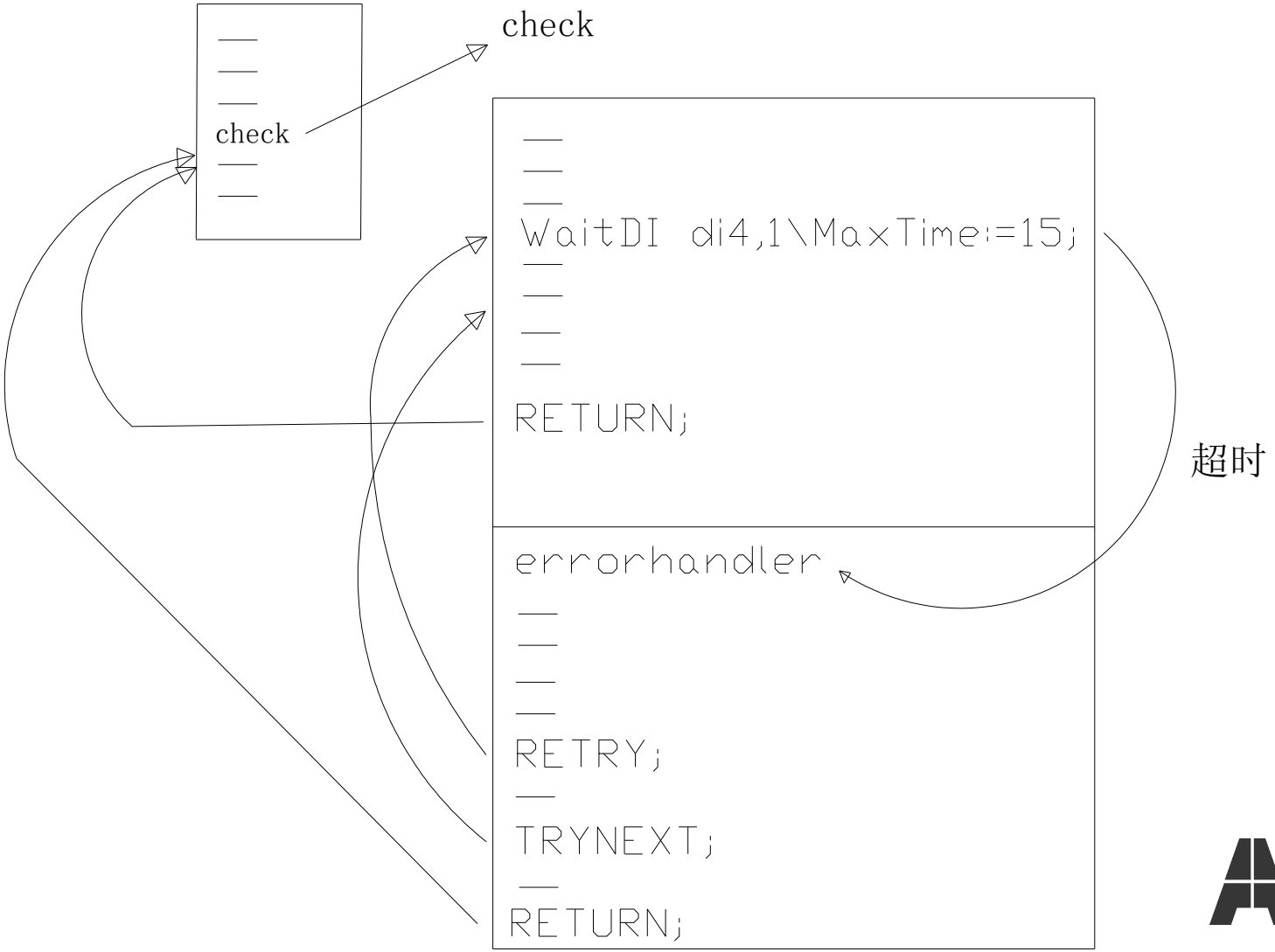
故障处理指令- **RETRY**

RETRY;

应用：
当前指令只用于机器人故障处理程序Error Handler内，经过故障处理后，再次对发生故障的指令进行再次运行。
限制：
通过当前指令对所发生故障的指令进行再次运行，连续4次尝试后，故障仍无法解决，机器人将停止运行，示教器显示错误信息ERR_EXCRTYMAX

ABB

故障处理指令- TRYNEXT



故障处理指令- TRYNEXT

TRYNEXT;

应用：
当前指令只用于机器人故障处理程序Error Handler内，经过故障处理后，跳过发生故障的指令继续运行程序。

故障处理指令- **RAISE**

RAISE[Error no.];

RAISE[Error no.]: 错误信息编号 （**errnum)**

应用：

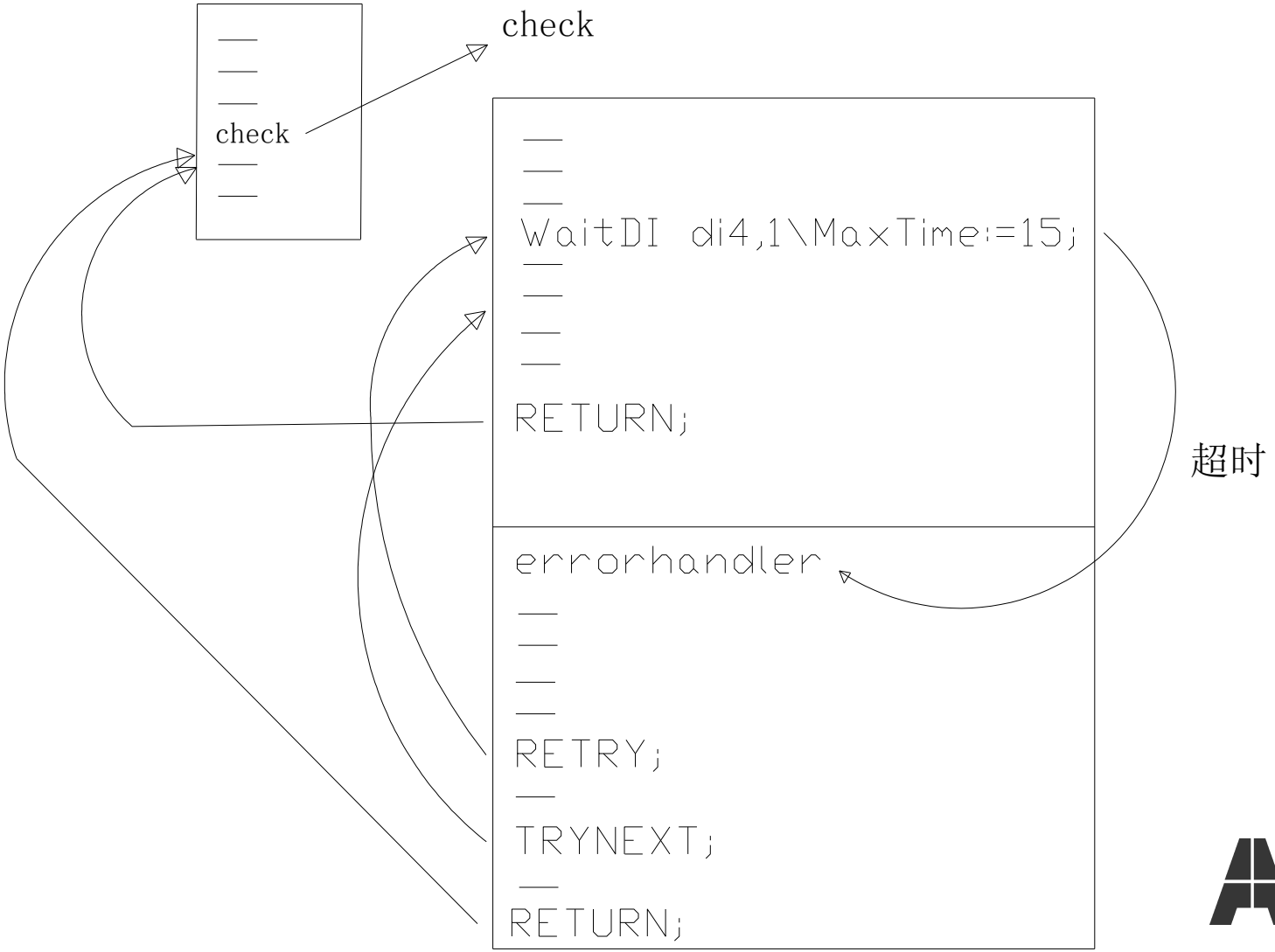
当前指令如果不使用参变量，只用于机器人故障处理程序Error Handler内，经过故障处理后，跳至上一层例行程序故障处理程序内，继续进行故障处理;如果使用参变量，只用于例行程序内，直接进入当前例行程序故障处理程序进行处理。

限制：

参变量错误代码范围为1-90，超出范围示教器显示错误信息ERR_ILLRAISE

如果需要更多的错误代码，超出1-90，必须使机器人指令BookErrNo.

故障处理指令- RUTURN



故障处理指令- **RETURN**

RETURN [Return value];

[Return value]: 返回时间值 （**all)**

应用：
当前指令如果使用参变量，只用于机器人函数例行程序内，经过运行返回相应的值;通常情况下，在不使用参变量时，机器人运行至此指令时，无论是主程序main.标准例行程续PROC.中断程序TRAP.故障程序Error handler都代表当前例行程序结束。

座标转换指令

PDispOn

PDispOff

PDispset

EOffsOn

EOffsOff

EOffsSet

座标转换指令- **PDispOn**

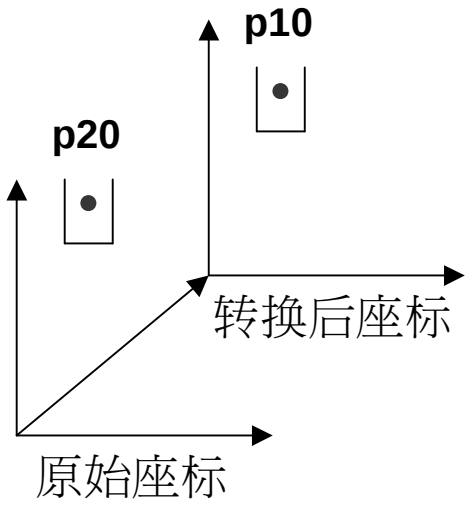
PDispOn [\Rot][\Exep,]
ProgPoint,tool [\Wobj];

[\Rot] :	座标旋转开关	（**switch)**
[\Exep] :	运行起始点	（**robtarget)**
ProgPoint :	座标原始点	（**robtarget)**
tool :	工具坐标系	（**tooldata)**
tool :	工件坐标系	（**wobjdata)**

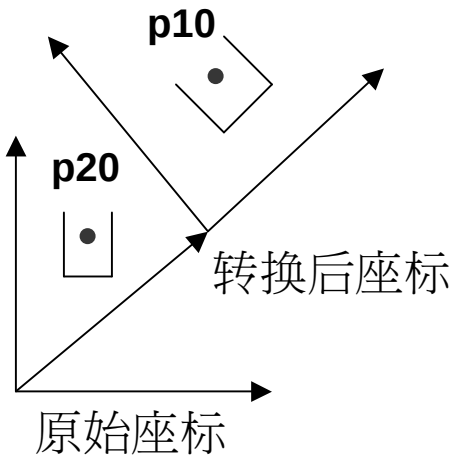
应用：

当前指令可以使机器人座标通过编程进行即时转换，通常用于水切割等运行轨迹保持不变的场合，可以快捷的完成工作位置修正。

座标转换指令- **PDispOn**



MoveL p10,v500,z10,tool1;
PDispOn\Exep:=p10,p20,tool1;

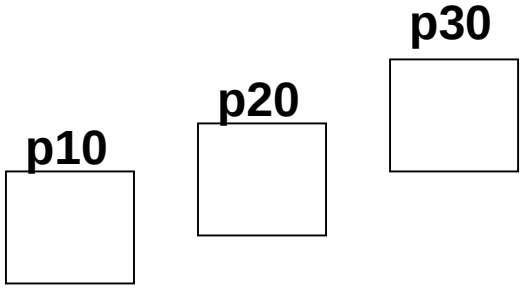


MoveL p10,v500,fine\Inpos:=inpos50,tool1;
PDispOn\Rot\ExeP:=p10,p20,tool1;

座标转换指令- **PDispOn**

```
PROC draw_square()
    PDispOn *,tool1;
    Movel *,v500,z10,tool1;
    Movel *,v500,z10,tool1;
    Movel *,v500,z10,tool1;
    Movel *,v500,z10,tool1;
    PDispoff;
ENDPROC
MoveL p10,v500,fine\Inpos:=ippos50,tool1;
draw_square;
MoveL p20,v500,fine\Inpos:=ippos50,tool1;
draw_square;
MoveL p30,v500,fine\Inpos:=ippos50,tool1;
draw_square;
```

不使用参变量[\ExeP]
机器人默认为当前点



ABB

座标转换指令- **PDispOn**

实例：

```
SearchL sen1,psearch,p10,v100,tool1;
PDispOn\ExeP:=psearch,*,tool1;
```

限制**:**

当前指令在使用后，机器人座标将被转换，直到使用指令**PDispOff**后才失效。

在下列情况下，机器人座标转换功能将自动失效：

- **-**机器人系统冷启动
- **-**载入新机器人程序
- **-**程序重置（**Start From Beginning)**

座标转换指令- PDispOff

PDispOff;

应用：

当前指令用于使机器人通过编程达到座标转换功能失效，必须与指令PDispOn或指令PDispSet同时使用。

限制：

```
MoveL p10,v500,z10,tool1;  ←——— 座标转换失效
PDispOn\ExeP:=p10,p11,tool1;
MoveL P20,v500,z10,tool1;←——— 座标转换生效
MoveL p30,v500,z10,tool1;
PDispOff;
MoveL p40,v500,z10,tool1;←———座标转换失效
```

座标转换指令- **PDispSet**

PDispSet DispFrame;

应用：
当前指令通过输入坐标偏差量，使机器人座标通过编程进行即时转换，通常用于切歌等运行轨迹保持不变的场合，可以快捷的完成工作位置修正

座标转换指令- **PDispSet**

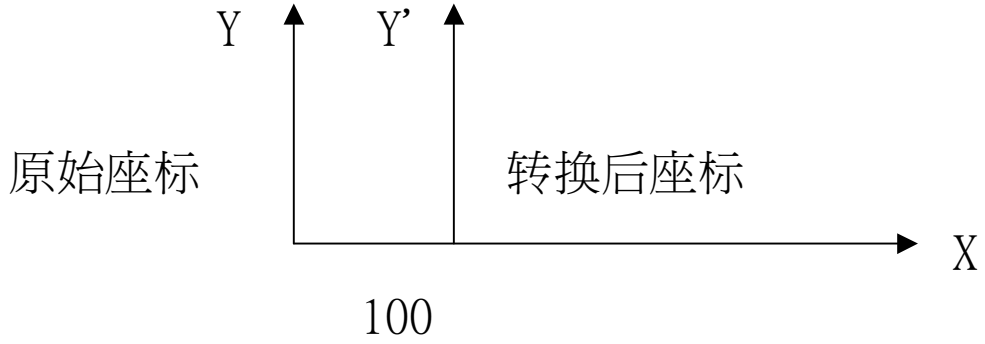
实例：

```
VAR pose xp100:=[[100,0,0],[1,0,0,0]];
MoveL p10,v500,z10,tool1;
PDispSet xp100
MoveL p20,v500,z10,tool1;
PDispOff;
MoveL p30,v500,z10,tool1;
```

座标转换失效

座标转换失效

座标转换失效



座标转换指令- **PDispSet**

限制：

当前指令在使用后，机器人座标将被转换，直到使用指令PDispOff后才失效。

在下列情况下，机器人座标转换功能将自动失效

- -机器人系统冷启动
- -载入新机器人程序
- -程序重置（**Start From Beginning)**

ABB

座标转换指令- EOffsOn

EOffsOn [\ExeP]ProgPoint;

[\Exep] : 运行起始点 （**robtarget)**

ProgPoint; : 座标原始点 （**robtarget)**

应用：

当前指令用于机器人外轴位置通过编程进行即时更改，通常用于带导轨的机器人。

实例：

```
searchL sen1,psearch,p10,v100,tool1;
PDispOn\ExeP:=psearch,*,tool1;
EOffsOn\ExeP:=psearch,*;
```

座标转换指令- **EOffsOn**

限制：
当前指令在使用后，机器人外轴位置将被更改，
直到使用指令EOffsOff后才失效。
在下列情况下，机器人座标转换功能将自动失效

- -机器人系统冷启动
- -载入新机器人程序
- -程序重置（**Start From Beginning)**

ABB

座标转换指令- EOffsOff

EOffsOff;

应用：

当前指令用于使机器人通过编程达到的外轴位置更改功能失效，必须与指令EOffsOn或指令EOffsSet同时使用。

限制：

```
MoveL p10,v500,z10,tool1;
EOffsOn\ExeP:=p10,p11,tool1;
MoveL P20,v500,z10,tool1;
MoveL p30,v500,z10,tool1;
EOffsOff;
MoveL p40,v500,z10,tool1;
```

座标转换失效

座标转换生效

座标转换失效

座标转换指令- **EOffsSet**

EOffsSet EAxOffs;

EAxOffs:　　外轴位置偏差量　　（**extjoint)**

应用：
当前指令通过输入外轴位置偏差量，使机器人外轴位置通过编程即时更改，对于导轨类外轴，偏差值单位为mm，对于转轴类外轴，偏差值单位为角度。

座标转换指令- **EOffsSet**

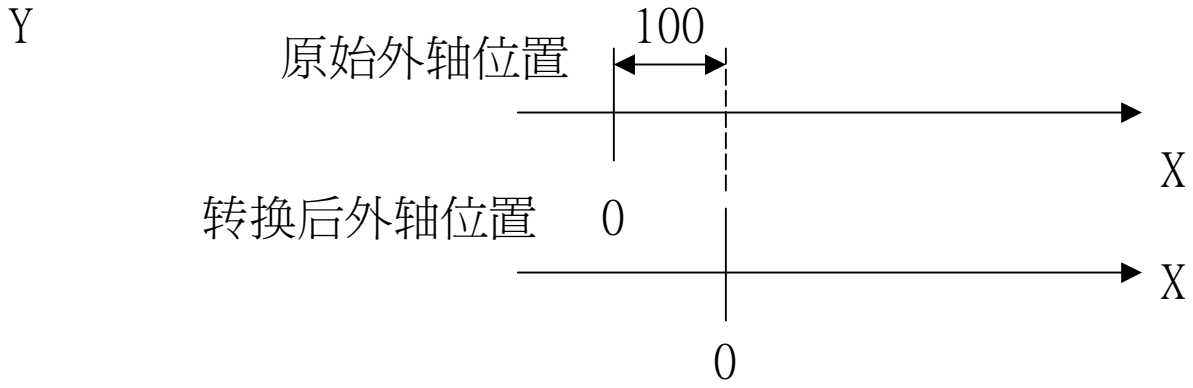
实例：

```
VAR extjoint eax_a_p:=[100,0,0,0,0,0];
MoveL p10,v500,z10,tool1;
EOffsSet xp100
MoveL p20,v500,z10,tool1;
EOffsOff;
MoveL p30,v500,z10,tool1;
```

座标转换失效

座标转换失效

座标转换失效



ABB

座标转换指令- **EOffsSet**

限制：
当前指令在使用后，机器人外轴位置将被更改，
直到使用指令EOffsOff后才失效。
在下列情况下，机器人座标转换功能将自动失效

- -机器人系统冷启动
- -载入新机器人程序
- -程序重置（**Start From Beginning)**

ABB

运动触发指令

TriggIO

TriggInt

TriggEquip

Triggj

TriggL

Triggc

运动触发指令- TriggIO

TriggIO TriggData,Distance
[\Start][\Time][\Dop][\Gop]|
[\Aop][\ProcID],SetValue
[\DODelay];

[TriggData]: 触发变量名称。（**triggdata)**

Distance: 触发距离mm。（**unm)**

[\Start]: 触发起始开关。（**switch)**

[\Time]: 时间触发开关。（**switch)**

[\Dop]: 时间数字输出。（**signaldo)**

运动触发指令- TriggIO

TriggIO TriggData,Distance
[\Start][\Time][\Dop][\Gop]|
[\Aop][\ProcID],SetValue
[\DODelay];

[\Gop] : 触发组合输出。（**signalgo)**

[\Aop] : 触发模拟输出。（ **signalao)**

[\ProcID] : 过程处理触发。（**num)**

SetValue : 相应信号值。 （**num)**

[\DODelay] : 数字输出延迟。 （**num)**

运动触发指令- TriggIO

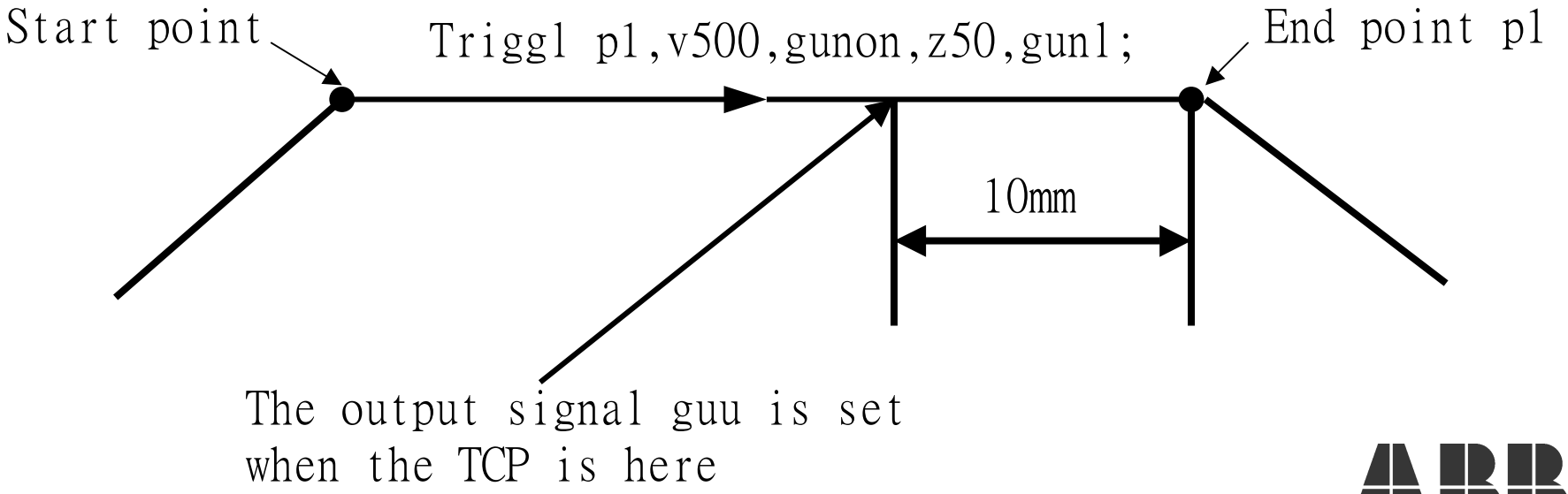
应用：

机器人可以在运动时通过触发指令精确的输出相应信号，当前指令用于定义触发性质，此指令必须与其他触发指令TriggJ.TriggL.或TriggC同时使用才有意义，同机器人指令TriggEquip比较，多了时间控制功能，少了外部设备触发延迟功能，通常用于喷涂.涂胶等行业，使用参变量[\Start],表示以运动起始点触发基准点，默认为运动终止点;使用参变量[\Time],以时间来控制触发，允许最大时间为0.5s，详见限制;参变量[\ProcID],正常情况下用户无法自行使用，此参变量用于IPM过程处理。

运动触发指令- TriggIO

实例：

```
VAR triggdata gunon;
TriggIO gunon,10\Dop:=gun,1;
TriggL p1,v500,gunon,z50,gun1;
```



运动触发指令- TriggIO

限制：

当前指令使有参变量[\Time]可以提高信号输出精度，此参变量以目标点为基准，使用固定的目标点fine此转角zone精度高，一般情况下，此参变量采用固定目标点。

参变量[\Time]设置的时间小于机器人开始减速时间（最大0.5s),例如：运行速度500mm/s,IRB2400为150ms,IRB6400为250ms,机器人在设置时间超过减速时间的情况下，实际控制时间会缩短，但不会对正常运行造成影响。

ABB

运动触发指令- TriggInt

TriggInt TriggData,Distance
[\Start]|[\Time],Interrupt;

[TriggData]:	触发变量名称。	**（triggdata)**
Distance:	触发距离mm。	**（unm)**
[\Start]:	触发起始开关。	**（switch)**
[\Time]:	时间触发开关。	**（switch)**
Interrupt :	触发中断名称。	**（signaldo)**

运动触发指令- TriggInt

应用：

机器人可以在运动时通过触发指令精确的输出相应信号，当前指令用于定义触发性质，此指令必须与其他触发指令TriggJ.TriggL.或TriggC同时使用才有意义，通常用于喷涂.涂胶等行业，使用参变量[\Start],表示以运动起始点触发基准点，默认为运动终止点;使用参变量[\Time],以时间来控制触发，允许最大时间为0.5s，详见限制.

运动触发指令- TriggInt

限制：

正常情况下，当前指令从触发中断到得到响应，有5-120ms延迟，用指令TriggIO或TriggEquit控制信号输出效果更佳。

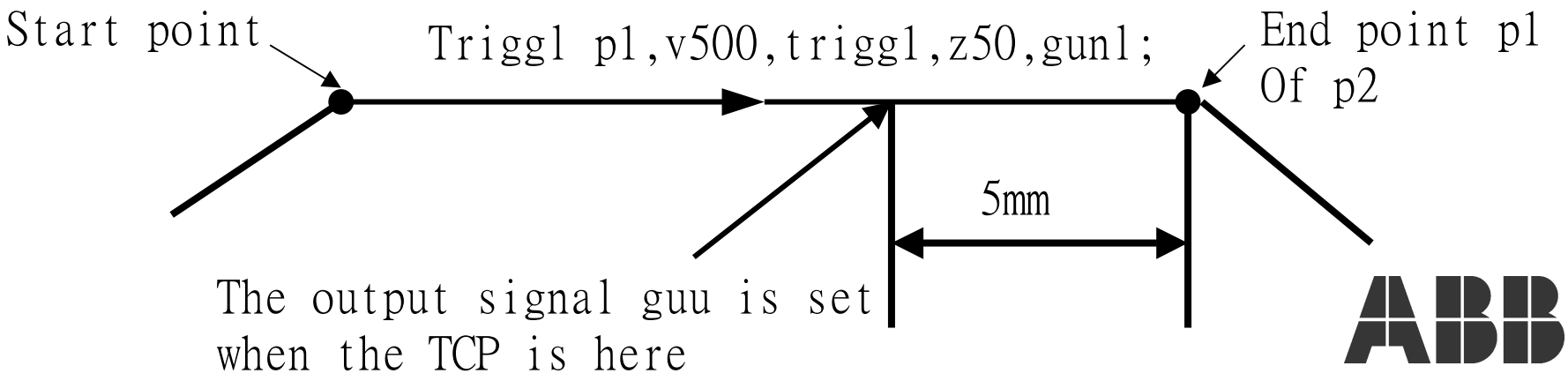
当前指令使用参变量， [\Time]可以提高信号输出精度 ,此参变量以目标点为基准，使用固定的目标点fine此转角zone精度高，一般情况下，此参变量采用固定目标点。

参变量[\Time]设置的时间小于机器人开始减速时间（最大0.5s),例如：运行速度500mm/s,IRB2400为150ms,IRB6400为250ms,机器人在设置时间超过减速时间的情况下，实际控制时间会缩短，但不会对正常运行造成影响。

运动触发指令- TriggInt

实例：

```
VAR intnum intno1;
VAR triggdata trigg1;
CONNECT intno1 WITH trap1;
Triggint trigg1,5,intno1;
TriggL p1,v500,trigg1,z50,gun1;
TriggL p2,v500,trigg1,z50,gun1;
Idelete intno1;
```



运动触发指令- TriggEquip

TriggEquip TriggData,
Distance [\Start],EquipLag
[\DOp][GOp][AOp][\ProcID],
SetValue [\Inhib];

[TriggData]: 触发变量名称。（**triggdata)**

Distance: 触发距离mm。（**unm)**

[\Start]: 触发起始开关。（**switch)**

EquipLag: 触发延迟补偿s。（**switch)**

[\Dop]: 触发数字输出。（**signaldo)**

运动触发指令- TriggEquip

TriggEquip TriggData,
Distance [\Start],EquipLag
[\DOp][GOp][AOp][\ProcID],
SetValue [\Inhib];

[\Gop] : 触发组合输出。（**signalgo)**

[\Aop] : 触发模拟输出。（ **signalao)**

[\ProcID] : 过程处理触发。（**num)**

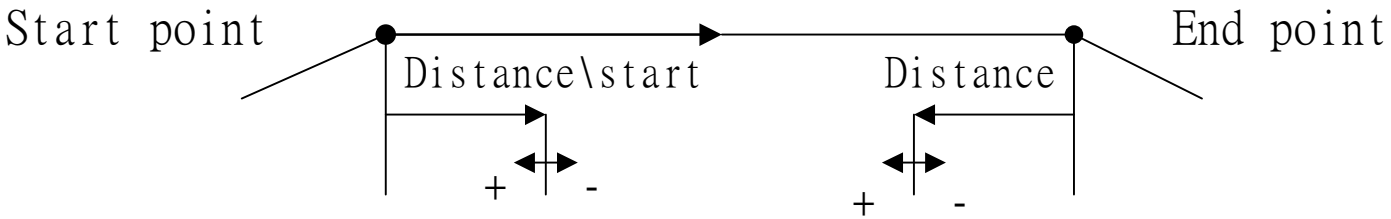
SetValue : 相应信号值。 （**num)**

[\Inhib] : 信号抑止数据。 （**bool)**

运动触发指令- TriggEquip

应用：

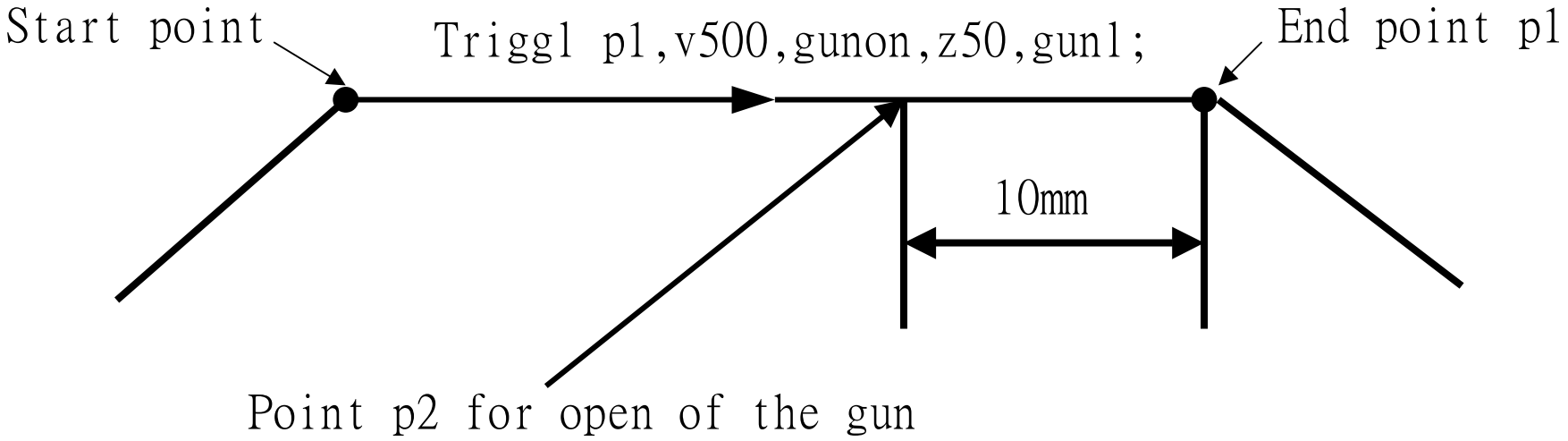
机器人可以在运动时通过触发指令精确的输出相应信号，当前指令用于定义触发性质，此指令必须与其他触发指令TriggJ.TriggL.或TriggC同时使用才有意义，同机器人指令TriggIO比较，多了外部设备触发延迟功能，少了时间控制功能，通常用于喷涂.涂胶等行业。 使用参变量[\Start],表示以运动起始点触发基准点，默认为运动终止点; 参变量[\ProcID],正常情况下用户无法自行使用，此参变量用于IPM过程处理。当参变量[\Inhib]值为TRUE，在触发点所有输出信号（AO GO DO)将被置为0



运动触发指令- TriggEquip

实例：

```
VAR triggdata gunon;
TriggEquip gunon,10,0.1\Dop:=gun,1;
TriggL p1,v500,gunon,z50,gun1;
```



运动触发指令- TriggEquip

限制：

当前指令通过触发延迟可以提高信号输出精度 设置的时间应小于机器人 开始减速时间（最大0.5s)

例如：IRB2400为150ms,IRB6400为250ms,机器人在设置时间超过减速时间的情况下，实际控制时间会缩短，但不会对正常运行造成影响。

触发延迟EquipLag值应小于系统参数内Event preset Time配值值，默认为60ms

如果触发延迟EquipLag值应大于系统参数内Event preset Time配值值，需要使用指令SingArea\Wrist.

运动触发指令- TriggJ

WaitTime [\InPos,] Time;

[\InPos] : 程序运行提前量开关。 （**switch)**

Time : 相应等待时间。 （**num)**

应用：

当前指令只用于机器人等待相应时间后，才执行以后指令，使用参变量[\Inpos],机器人及外轴必须在完全停止的情况下，才进行等待进间计时，此指令会延长循环时间

运动触发指令- TriggL

WaitTime [\InPos,] Time;

[\InPos] : 程序运行提前量开关。 （**switch)**

Time : 相应等待时间。 （**num)**

应用：

当前指令只用于机器人等待相应时间后，才执行以后指令，使用参变量[\Inpos],机器人及外轴必须在完全停止的情况下，才进行等待进间计时，此指令会延长循环时间