ABB机器人高级指令

ABB机器人高级指令

NWE技术研发部 自动化应用

目录

运动控制指令

AccSet

VelSet

ConfJ

ConfL

SingArea

PathReasol

SoftAct

SoftDeact

运动控制指令-AccSet

AccSet Acc,Ramp;

Acc:机器人加速度百分率**(num)**

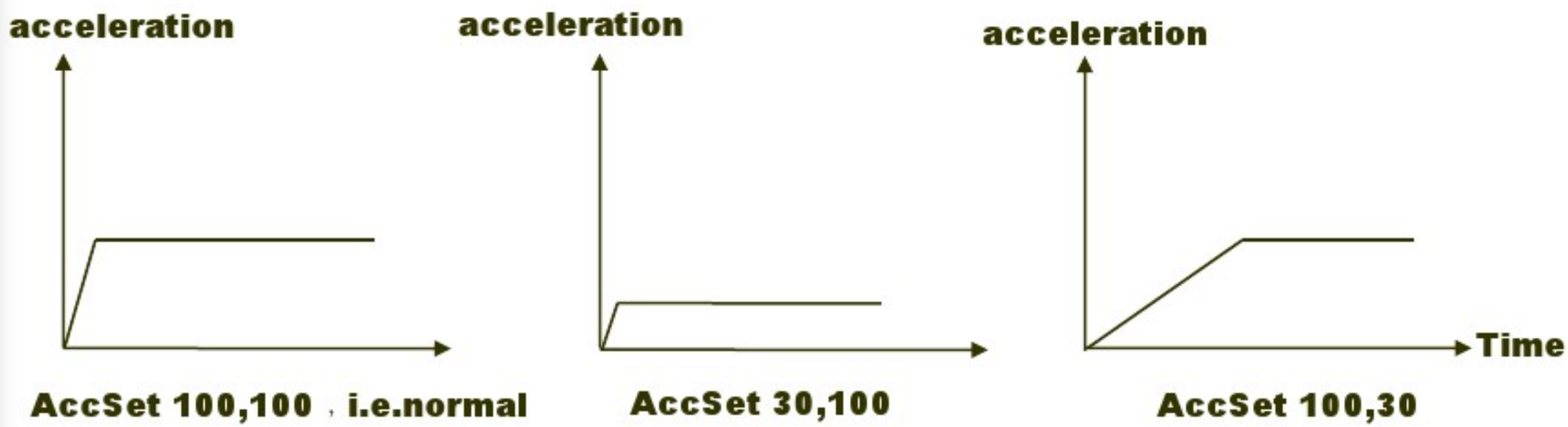
Ramp:机器人加速度坡度**(num)**

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：**AccSet 100,100;**

运动控制指令-AccSet

实例：



限制：

●机器人加速度百分率最小值为20，小于20，以20计，机器人加速度坡度最小值为10，小于10以10计；

●机器人冷启动，新程序加载与程序重置后，系统自动设置为默认值

运动控制指令-VelSet

VelSet Override,Max;

Override:机器人运行速度% **(num)**

Max:最大运行速度**mm/s (num)**

应用：

对机器人的运行速度进行限制，机器人运动指令中均带有运行速度，在执行运动速度控制指令**VelSet**后，实际运行速度为运动指令规定的运行速度乘以机器人运行速率，并且不超过机器人最大运行速度，系统默认值为：**VelSet 100,5000;**

運動控制指令-VelSet

實例：

```
VelSet  50,800;                          ←———— 500mm/s
MoveL  p1,v1000,z10,tool1;               ←———— 800mm/s
MoveL   p2,v1000\v:=2000,z10,tool1;      ←———— 10s
MoveL   p3,v1000\T:=5,z10,tool1;
VelSet   80,1000;
MoveL   p1,v1000,z10,tool1;              ←———— 800mm/s
MoveL   p2,v5000,z10,tool1;              ←———— 1000smm/s
MoveL   p3,v1000\v:=2000,z10,tool1;      ←———— 1000mm/s
MoveL   p4,v1000\T:=5,z10,tool1;         ←———— 6.25s
```

運動控制指令-VelSet

限製：

- 機器人冷啟動，新程序載入與程序重置後，系統自動設置為默認值；
- 機器人使用參變量[\T]時，最大運行速度將不起作用；
- **Override**對數據(**speeddate**)內所有項都起作用，例如：**TCP**、方位及外軸，但對焊接參數**welddate**與**seamdate**內機器人運動速度不起作用；
- **Max**只對速度數據(**speeddate**)內**TCP**這項起作用；

運動控制指令-ConfJ

ConfJ[\On]|[\Off];

[\On]: 啟動軸配置數據 **(switch)**

關節運動時，機器人移動至絕對**ModPos**點，如果無法到達，程序將停止運行；

[\Off]: 默認**H**軸配置數據 **(switch)**

關節運動時，機器人移動至**ModPos**點，軸配置數據默認為當前最接近值；

運動控制指令-ConfJ

應用：

對機器人運行姿態進行限製與調整，程序運行時，使機器人運行姿態得到控制，系統默認值為：**ConfJ\On;**

實例：

ConfJ\On;

…..

ConfJ\Off;

限製：

- 機器人冷啟動，新程序載入與程序重置後，系統自動設置為默認值

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**H**轴配置数据**(switch)**

直线运动时，机器人移动至**ModPos**点，轴配置数据默认为当前最接近值；

運動控制指令-ConfL

應用：

對機器人運行姿態進行限製與調整，程序運行時，使機器人運行姿態得到控制，系統默認值為：**ConfL\On;**

實例：

ConfL\On;

…..

ConfL\Off;

限製：

- 機器人冷啟動，新程序載入與程序重置後，系統自動設置為默認值

運動控制指令-SingArea

SingArea[\Wrist] [\Off];

[\Wrist]: 啟動位置方位調整 **(switch)**

機器人運動時，為了避免頻繁死機，位置點允許其方位值有些許改變，例如：在五軸零度時，機器人四六軸平行；

[\Off]: 關閉位置方位調整 **(switch)**

機器人運動時，不允許位置點方位改變，是機器人的默認狀態；

運動控制指令-SingArea

應用：

當前指令通過對機器人位置點姿態進行些許改變，可以絕對避免機器人運行時死機，但是，機器人運行路徑會受影響，姿態得不到控制，通常使用與通過複雜姿態點，絕對不能作為工作點使用；

實例：

SingArea\Wrist;

…..

SingArea\Off;

運動控制指令-SingArea

限製：

- 以下情況機器人將自動恢復默認值 **SingArea\Off**
  - -機器人冷啟動
  - -系統重新載入新的程序
  - -系統重置(Start From Beginning)

運動控制指令-PathResol

PathResol PathSampleTime;

PathSampleTime：路徑控制%　**(num)**

應用：

當前指令用於更改機器人主機系統參數;調整機器人路徑采樣時間，從而達到控制機器人運行路徑的效果，通過此指令可以提高機器人運動精度或縮短循環時間，路徑控制默認值為**100%**，調整範圍為**25%-400%**路徑控制百分比越小，運動精度越高，占用**CPU**資源也越多

運動控制指令-PathResol

實例：

MoveJ p1,v1000,fine,tool1;

PathResol 150;

機器人在臨界運動狀態(重載、高速、路徑變化複雜情況下接近最大工作區域)；增加路徑控制值，可避免頻繁死機；外軸以很低的速度與機器人聯動，增加路徑控制值，可避免頻繁死機；

機器人進行高頻率擺動弧焊時，需要很高的路徑采樣時間，需要減小路徑控制值；機器人進行小範圍複雜運動時，需要很高的精度，需要減小路徑控制值；

運動控制指令-PathResol

限製：

- 機器人必須在完全停止後才能更改路徑控制值，否則，機器人將默認一個停止點，并且顯示錯誤信息**50146**；
- 機器人在更改路徑控制值時，機器人被強製停止運行，機器人將不能立刻恢復正常運行**(Restart)**
- 以下情況機器人將自動恢復默認值**100%**

  -機器人冷啟動

  -系統載入新的程序

  -程序重置**(Start From Beginning)**

運動控制指令-SoftAct

SoftAct[\MechUnit],Axis,Softness[\Ramp]

[\MechUnit]：	軟化外軸名稱	**(mechunit)**
Axis:	軟化外軸號碼	**(num)**
Softness:	軟化值%	**(num)**
[\Ramp]:	軟化坡度%	**(num)**

應用：

當前指令用於軟化機器人主機或外軸伺服系統，軟化值範圍**0%-100%**,軟化坡度範圍**>=100%**,此指令必須與指令**SoftDeact**同時使用，通常不使用於工作位置。

運動控制指令-SoftAct

實例：

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

限製：

- 機器人被強製停止運行後，軟伺服設置將自動失效；
- 同一轉軸軟化伺服不允許被連續設置兩次；

```
.....
SoftAct 3,20;
SoftAct 3,20;
.....
```

```
.....
SoftAct 3,20;
MoveJ *,v100,fine,tool;
SoftAct 3,20;
.....
```

運動控制指令-SoftDeact

SoftDeact [\Ramp]

[\Ramp]: 軟化坡度**>=100%** **(num)**

應用：

當前指令用於使軟化機器人主機或外軸伺服系統指令**SoftAct**失效。

實例：

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

外軸激活指令

ActUnit

DeactUnit

外軸激活指令-ActUnit

ActUnit MecUnit

MecUnit: 外軸名 **(mecunit)**

應用：

將機器人一個外軸激活，例如：當多個外軸公用一個驅動板時，通過外軸激活指令**ActUnit**選擇當前所使用的外軸；

外軸激活指令-ActUnit

實例：

```
MoveL p10,v100,fine,tool1;
ActUnit  track_motion;
MoveL p20,v100,z10,tool1;
DeactUnit  track_motion;
ActUnit  orbit_a;
MoveL p30,v100,z10,tool1;
```

P10,外軸不動

P20,外軸聯動

Track_motion

P30,外軸聯動

Orbit_a

限製：

- 不能在指令**StorePath…RestorePath**內使用
- 不能在預置程序**RESTART**內使用
- 不能在機器人轉軸處於獨立狀態時使用

外軸激活指令-DeactUnit

DeactUnit MecUnit

MecUnit: 外軸名 **(mecunit)**

應用：

將機器人一個外軸失效，例如：當多個外軸公用一個驅動板時，通過外軸激活指令**DeactUnit**使當前所使用的外軸失效；

外軸激活指令-DeactUnit

實例：

```
MoveL p10,v100,fine,tool1;
ActUnit  track_motion;
MoveL p20,v100,z10,tool1;
DeactUnit  track_motion;
ActUnit  orbit_a;
MoveL p30,v100,z10,tool1;
```

P10,外軸不動

P20,外軸聯動 Track_motion

P30,外軸聯動 Orbit_a

限製：

- 不能在指令**StorePath…RestorePath**內使用
- 不能在預置程序**RESTART**內使用

計數指令

Add

Clear

Incr

Decr

計數指令-Add

Add Name,AddValur

Name: 數據名稱 **(num)**

AddValue: 增加的值 **(num)**

應用：

在一個數字數據值是增加相應的值，可以用賦值指令替代；

實例：

Add reg1,3; 等同於 **reg1:=reg1+3;**

Add reg1,reg2; 等同於**reg1:=reg1+reg2;**

計數指令-Clear

Clear Name;

Name: 數據名稱 **(num)**

應用：

將一個數字數據的值歸零，可以用賦值指令替代；

實例：

Clear reg1; 等同於 **reg1:=0**

計數指令-Incr

Incr Name;

Name: 數據名稱 **(num)**

應用：

在一個數字數據的值上加**1**，可以用賦值指令替代；一般用於產量計數；

實例：

Incr reg1; 等同於 **reg1:=reg1+1;**

計數指令-Decr

Decr Name;

Name: 數據名稱 **(num)**

應用：

在一個數字數據的值減**1**，可以用賦值指令替代；一般用於產量計數；

實例：

Incr reg1; 等同於 **reg1:=reg1-1;**

輸入輸出指令

AliasIO	InvertDO
IODisable	IOEnable
PluseDO	Reset
Set	SetAO
SetDO	SetGO
WaitDI	WaitDO

輸入輸出指令-AliasIO

AliasIO FromSignal,ToSignal;

FromSignal: 機器人系統參數內所定義的信號名稱；**(SignalXX or string)**

ToSignal: 機器人程序內所使用的信號名稱 **(SignalXX)**

應用：

對機器人系統參數內定義的信號名稱進行化名，給機器人程序使用，一般使用與**LoadedModule**或**Built-in Module**內，例如：多台機器人使用相同系統參數

輸入輸出指令-AliasIO

實例：

```
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,在系統參數內定義

Alias_do,在機器人程序內定義

輸入輸出指令-AliasIO

限製：

- 指令**AliasIO**必須放置在預置程序**START**內或程序內使用相應的信號之前；
- 指令**AliasIO**在示教器上無法輸入，只能通過離線編輯輸入；
- 指令**AliasIO**需要軟件**Develop’s Functions**的支持

輸入輸出指令-InvertDO

InvertDO Signal;

Signal: 輸出信號名稱； **(SignalDO)**

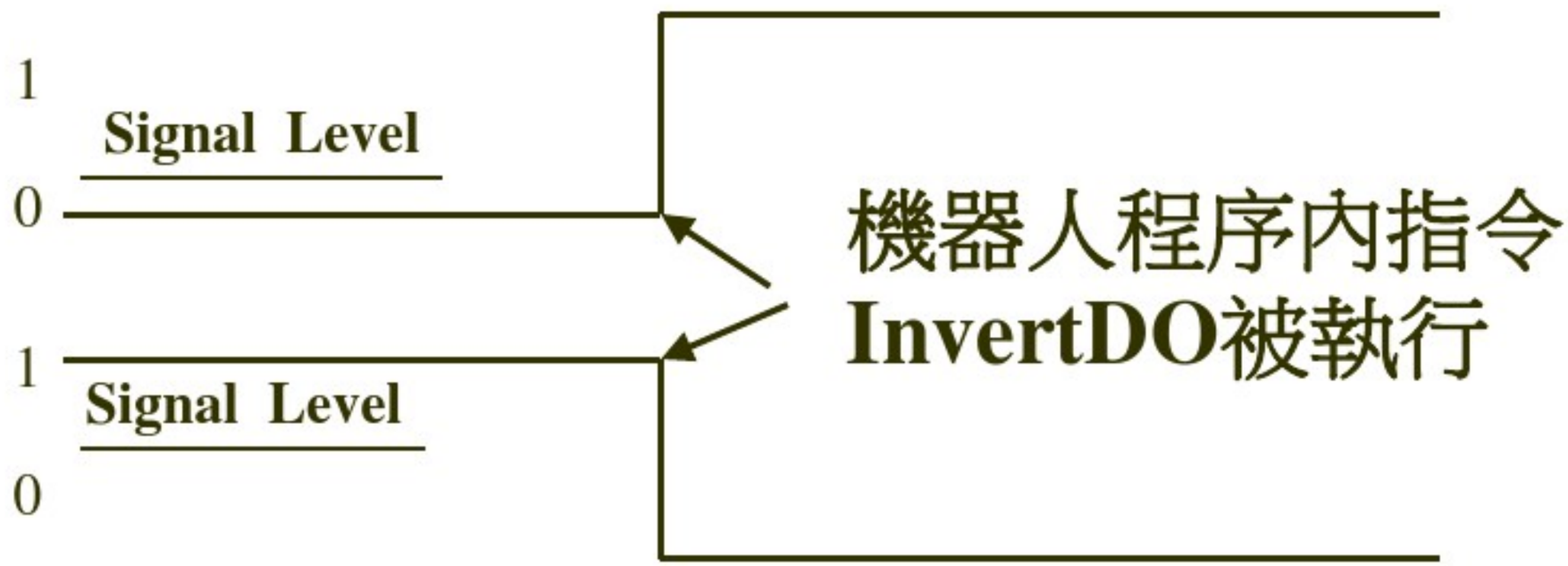
應用：

將機器人輸出信號值反轉，**0**為**1**，**1**為**0**，在系統參數內也可定義；

實例:

InvertDO do15;

輸入輸出指令-InvertDO



輸入輸出指令-IODisable

IODisable UnitName,MaxTime;

UnitName: 輸入輸出板名稱；**(num)**

MaxTime: 最長等待時間 **(num)**

應用：

通過指令可以使機器人輸入板在程序運行時自動失效，系統將一塊輸入輸出板失效需要**2-5**秒，如果失效時間超過最長等待時間，系統將進入**ErrorHandler**處理，錯誤代碼為**ERR_IODISABLE**，如果例行程序沒有**ErrorHandler**，機器人將停機報錯。

輸入輸出指令-IODisable

實例：

```
PROC go_home( )
  recover_flag:=1;
  IODisable “cell”,0’
  MoveJ home,v1000,fine,tool1;
  recover_cover:=2;
  IODisable “cell”,5;
ERROR
  IF ERRNO=ERR_IODISABLE THEN
    IF recover_flag=1 THEN
      TRYNEXT;
    ELSEIF recover_flag=2 THEN
      RETRY;
    ENDIF
  ELSEIF ERRNO=ERR_EXCRPTYMAX THEN
    ErrorWrite “IODisable error”,”Restart the program”;
    Ttop;
  ENDIF
ENDPROC
```

輸入輸出板cell開始失效，最長等待時間為0，肯定進入Error Handler處理

利用機器人移動到home的時間完成輸入輸出板失效

確認輸入輸出板失效

連續5次RETRY後，仍無法完成輸入輸出板的失效

輸入輸出指令-IODisable

Error Handing

- **ERR_IODISABLE**

  超過最長等待時間，系統仍未完成輸入輸出板失效；

- **ERR_CALLIO_INTER**

  系統在執行輸入輸出板失效與激活時，當前輸入輸出板與再次被失效或激活，形成沖突

- **ERR_NAME_INV ALID**

  輸入輸出板名稱錯誤或無法進行失效與激活操作

輸入輸出指令-IOEnable

IOEnable UnitName,MaxTime;

UnitName: 輸入輸出板名稱；**(num)**

MaxTime: 最長等待時間 **(num)**

應用：

通過指令可以使機器人輸入板在程序運行時自動激活，系統將一塊輸入輸出板失效需要**2-5**秒，如果失效時間超過最長等待時間，系統將進入**ErrorHandler**處理，錯誤代碼為**ERR_IOENABLE**，如果例行程序沒有**ErrorHandler**，機器人將停機報錯。

輸入輸出指令-IOEnable

實例：

```
  VAR num max_retry:=0;
  .....
  IOEnable "cell:,0;
  SetDO cell_sig3,1;
ERROR
  IF ERRNO=ERR_IOENABLE THEN
     IF max_retry<5 THEN
        WaitTime 1;
        max_retry:=max_retry+1;
        RETRY;
     ELSE
        RAISE;
     ENDIF
  ENDIF
```

輸入輸出板cell開始激活，最長等待時間為0，肯定進入Error Handler處理

通過每次1秒進行計數，連續5次仍無法激活輸入輸出板，執行指令RAISE

輸入輸出指令-IOEnable

Error Handing

- **ERR_IOENABLE**

  超過最長等待時間，系統仍未完成輸入輸出板激活；

- **ERR_CALLIO_INTER**

  系統在執行輸入輸出板失效與激活時，當前輸入輸出板與再次被失效或激活，形成沖突

- **ERR_NAME_INV ALID**

  輸入輸出板名稱錯誤或無法進行失效與激活操作

輸入輸出指令-PulseDO

PulseDO[\High][\PLength]Signal;;

[\High]: 輸出脈沖時，輸出信號可以處在高電平；**(swtich)**

[\PLength]: 脈沖長度，**0.1s-32s,,**默認為 **0.2s (num)**

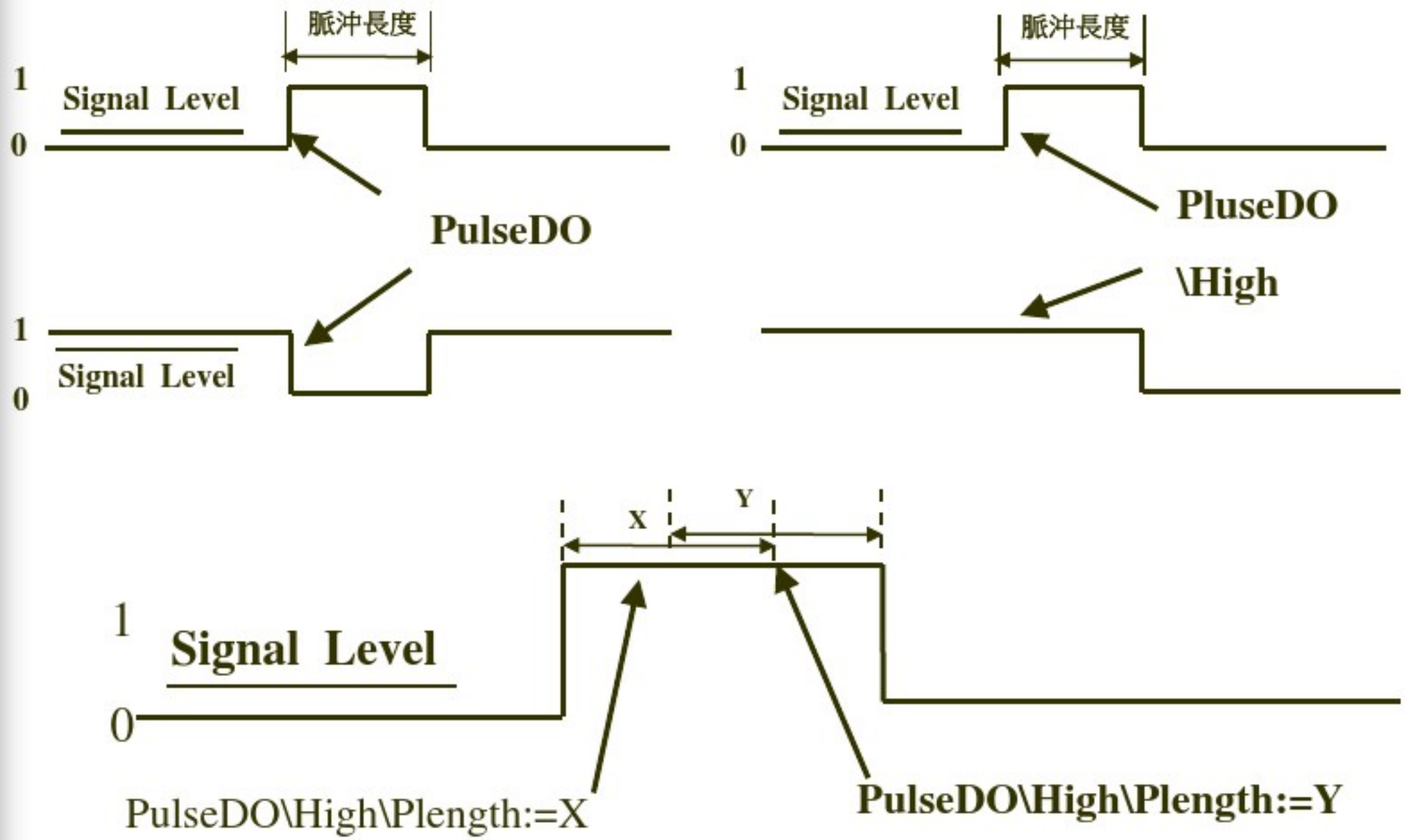
Signal: 輸出信號名稱 **(signaldo)**

應用：

機器人輸出數字脈沖信號，一般作為運輸鏈完成信號或計數信號

輸入輸出指令-PulseDO

實例：



輸入輸出指令-PulseDO

限製：

- 機器人脈沖輸出長度小於0.01秒，系統將報錯，不得不熱啟動

例如：

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

輸入輸出指令-Reset

Reset Signal;

Signal: 輸入輸出信號名稱；**(signaldo)**

應用：

將機器人相應數字輸出信號置為**0**，與指令**Set**對應，是自動化重要組成部分。

實例：

Reset do12;

輸入輸出指令-Set

Set Signal;

Signal: 輸入輸出信號名稱；**(signaldo)**

應用：

將機器人相應數字輸出信號置為**1**，與指令**Reset**對應，是自動化重要組成部分。

實例：

Set do12;

輸入輸出指令-SetAo

SetAo Signal,Value;

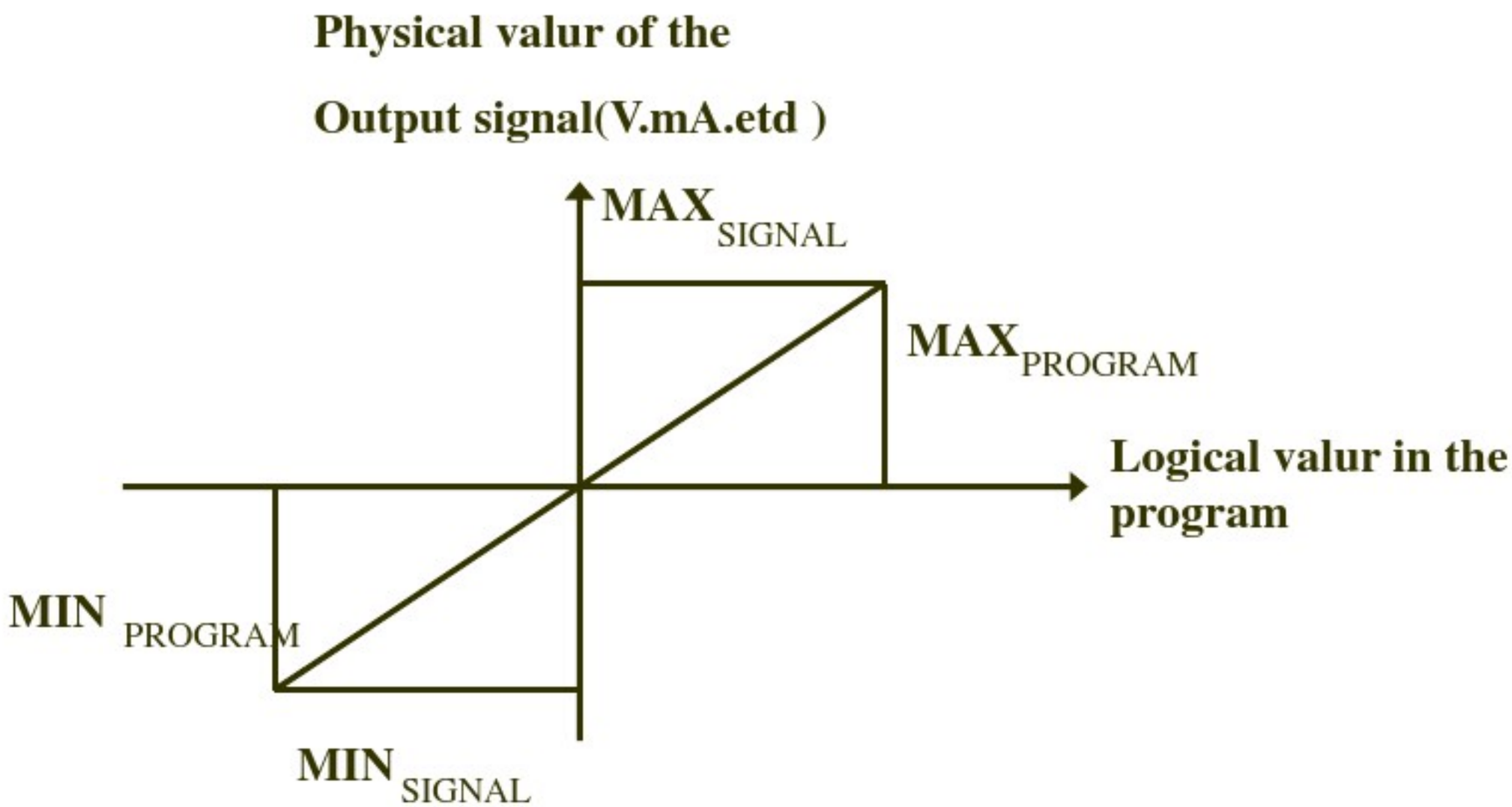
Signal: 模擬量輸出信號名稱；**(signaldo)**

Value: 模擬量輸出信號值； **(num)**

應用：

機器人當前模擬量輸出信號輸出相應的值;例如：機器人焊接時，通過模擬量輸出控制焊接電壓和送絲速度。

輸入輸出指令-SetAo



實例：

SetAO ao2,5.5;

SetAo weldcurr,curr_outp

輸入輸出指令-SetDo

SetDo[\SDelay] Signal,Value;

[\SDelay]:延時輸出時間**s** **(num)**

Signal: 輸出信號名稱；**(signaldo)**

Value: 輸出信號值； **(num)**

應用：

設置機器人相應組合輸出信號的值(采用**8421**碼)，可以設置延時輸出，延時範圍為**0.1s-32s,**默認狀態為沒有延時。

實例：

SetGo\SDelay:=0.2,go_Type,10;

輸入輸出指令-SetGo

SetGo[\SDelay] Signal,Value;

[\SDelay]:延時輸出時間**s** **(num)**

Signal: 輸出信號名稱；**(signaldo)**

Value: 輸出信號值； **(num)**

應用：

設置機器人相應數字輸出信號的值，與指令**Set**和**Reset**雷同，并且可以設置延時，延時的範圍為**0.1s-32s,**默認狀態為沒有延時。

實例：

SetDo\SDelay:=0.2,weld,high;

輸入輸出指令-WaitDI

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

Signal: 輸入信號名稱；**(signaldo)**

Value: 輸入信號值； **(num)**

[\MaxTime]: 最長等待時間**s** **(num)**

[\TimeFlag]: 超時邏輯量 **(bool)**

應用：

等待數字輸入信號滿足相應值，達到通信目的，是自動化生產重要組成部分，例如：機器人等待工件到位信號。

輸入輸出指令-WaitDI

實例：

```
PROC  PickPart(   )
   MoveJ pPrePick,vFastEmpty,zBig,tool1;
   WaitDI di_Ready,1;
   …..
ENDPROC
PROC  PickPart(   )
   MoveJ pPrePick,vFastEmpty,zBig,tool1;
   WaitDI di_Ready,1\MaxTime：=5；
   …..
ERROR
   IF  ERRNO=ERR_WAIT_MAXTIME  THEN
      TRWrite "…..";
      RETRY;
   ELSE
     RAISE;
   ENDIF
ENDPROC
```

機器人等待輸入信號，直到信號di_Ready值為1，才執行隨後指令

機器人等待相應輸入信號，如果5秒內仍沒有等到信號di_Ready值為1，自動進行Error Handler處理，如果沒有Error Handler ,機器人停機報錯

輸入輸出指令-WaitDI

實例：

```
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 nCounter;
   ENDWHOLE
      .....
ENDPROC
```

機器等待到位信號，如果1秒內仍沒有等到信號di_Ready值為1，機器人執行隨後指令，但此時TimeFlag值為TRUE，機器人等到di_Ready值為1，此時TimeFlag值為FALSE

輸入輸出指令-WaitDO

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

Signal: 輸出信號名稱；**(signaldo)**

Value: 輸出信號值； **(num)**

[\MaxTime]: 最長等待時間**s** **(num)**

[\TimeFlag]: 超時邏輯量 **(bool)**

應用：

等待數字輸出信號滿足相應值，達到通信目的，因為輸出信號一般情況下受程序控制此指令很少使用。

輸入輸出指令-WaitDO

實例：

```
PROC  Grip(   )
   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
      TRWrite "…..";
      RETRY;
   ELSE
     RAISE;
   ENDIF
ENDPROC
```

機器人等待輸出信號，直到信號do03_Grip為1，才執行隨後指令；

機器人等待相應輸出信號，如果5秒內仍沒有等到信號do03_Grip值為1，自動進行Error Handler處理，如果沒有Error Handler ,機器人停機報錯

輸入輸出指令-WaitDO

實例：

```
PROC  Grip(   )
   Set do03_Grip;
   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 nCounter;
   ENDWHOLE
      .....
ENDPROC
```

機器等待到位信號，如果1秒內仍沒有等到信號do03_Grip值為1，機器人執行隨後指令，但此時TimeFlag值為TRUE，機器人等到di_Ready值為1，此時TimeFlag值為FALSE

程序運行停止指令

Break

Exit

Stop

ExitCycle

程序運行停止指令-Break

Break;

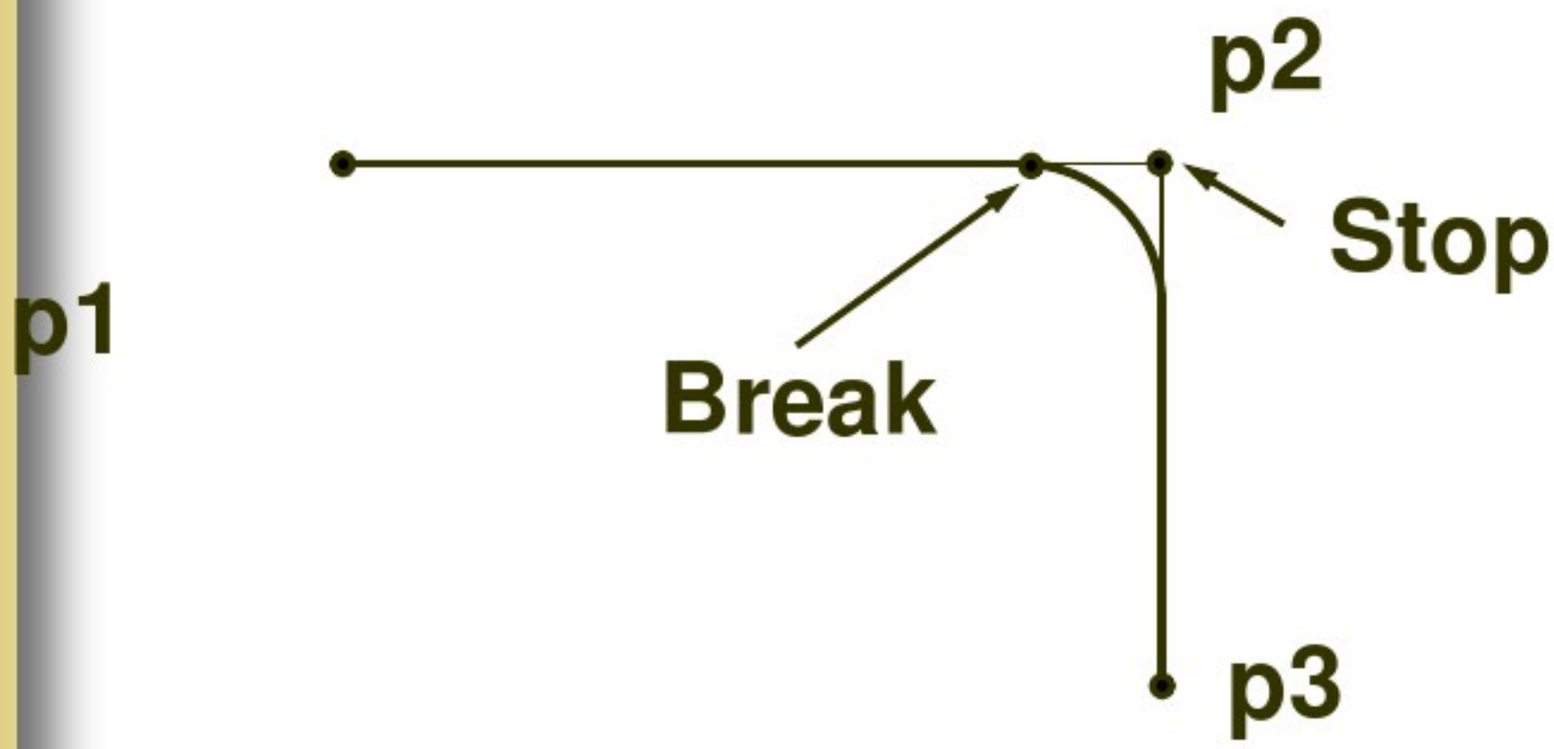
應用：

機器人在當前指令行立刻停止運行，程序運行指針停留在下一行指令，可以用**Start**鍵繼續運行機器人。

實例：

```
……
Break;
……
```

程序運行停止指令-Break



區別：

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

程序運行停止指令-EXIT

EXIT;

應用：

機器人在當前指令行停止運行，并且程序重置，程序運行指針停留在主程序第一行。

實例：

```
……
EXIT;
……
```

程序運行停止指令-STOP

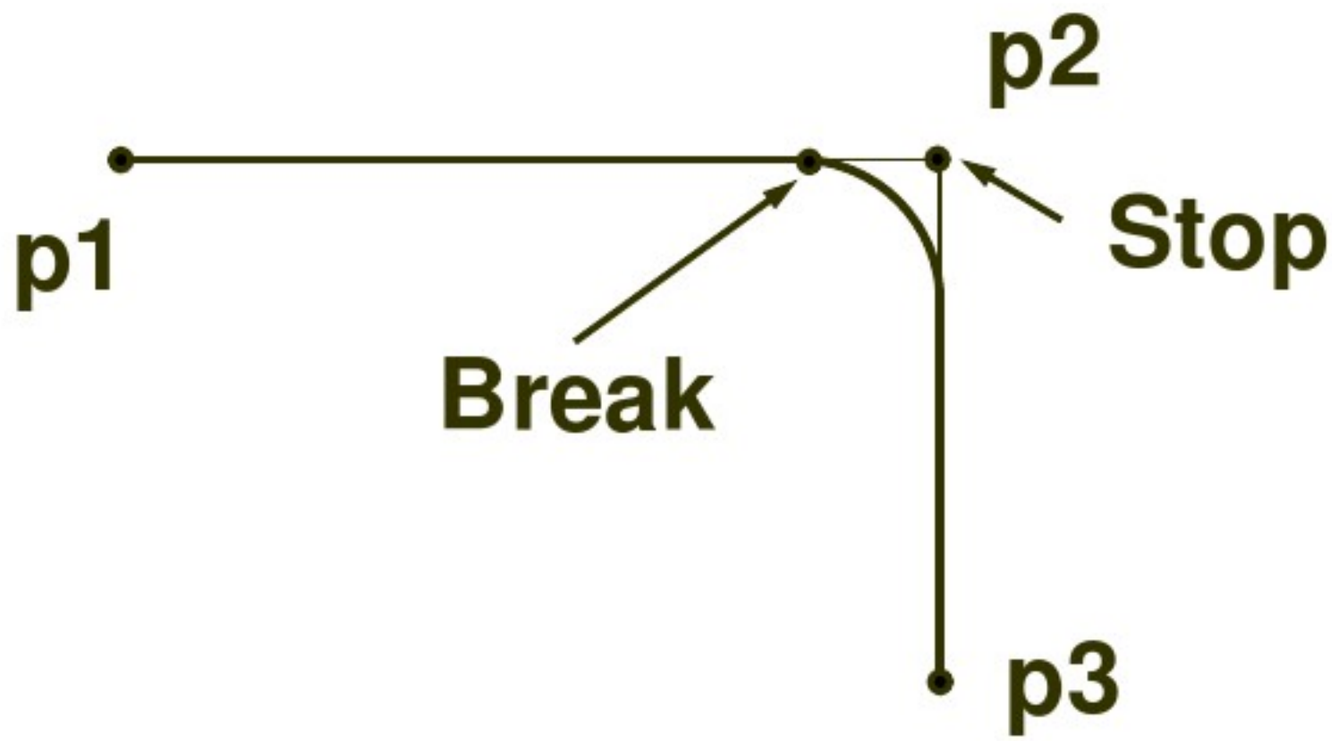
Stop [\NoRegain];

[\NoRegain]: 路徑恢復參數 **(num)**

應用：

機器人在當前指令停止運行，程序運行指針停留在下一行指令，可以用**Start**鍵繼續運行機器人，屬於臨時性停止，如果機器人停止期間被手動移動後，然後直接啟動機器人，機器人將警告確認路徑，如果此時采用參變量**[\NoRegain]**，機器人將直接運行。

程序運行停止指令-STOP



區別：

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

程序運行停止指令-ExitCycle

ExitCycle;

應用：

機器人在當前指令行立刻停止運行，并且設定當前循環結束，機器人自動從主程序第一行繼續運行下一個循環。

程序運行停止指令-ExitCycle

實例：

```
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 statr on the next item”;
    ExitCycle;
ENDTRAP
```

例行程序調用指令

ProcCall **CallByVar**

例行程序調用指令-ProcCall

ProcCall Procedure{Argument}

Procedure: 例行程序名稱 **(Identifier)**

{Argument}: 例行程序參數 **(All)**

應用：

機器人調用相應例行程序，同時給帶有參數的例行程序中相應的參數賦值。

實例：

Weldpipe1;

Weldpipe2 10,lowspeed;

Weldpipe3 10\speed:=20;

例行程序調用指令-ProcCall

限製：

- 機器人調用帶參數的例行程序時，必須包括所有強製性參數；
- 例行程序所有參數位置次序必須與例行程序設置一致；
- 例行程序所有參數數據類型必須與例行程序設置一致；
- 例行程序所有參數數據性質必須為**Input,Variable**或**Persistent;**

例行程序調用指令-CallByVar

CallByVar Name,Number;

Name: 例行程序名稱第一部分 **(string)**

Number: 例行程序名稱第二部分 **(num)**

應用：

通過指令中相應數據，機器人調用相應例行程序，但無法調用帶有參數的例行程序。

實例：

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

例行程序調用指令-CallByVar

限製：

- 不能調用帶參數的例行程序；
- 所有被調用的例行程序名稱第一部分必須相同，例如：**proc1,proc2,proc3;**
- 使用**CallByVar**指令調用例行程序比直接采用**ProcCall**調用例行程序需要更長時間；

Error Handling:

- **ERR_REFUNKPRC**

  系統無法找到例行程序名稱第一部分

- **ERR_CALLPROC**

  系統無法找到例行程序名稱第二部分

例行程序調用指令-CallByVar

實例比較：

```
TEST reg1
CASE 1:
    lf _door  door_loc;
CASE 2:
   rf _door  door_loc;
CASE 3:
   lr _door  door_loc;
CASE 4:
   rr _door  door_loc;
DEFAULT
   EXIT;
ENDTEST
CallByVar “proc”, reg1;
%”proc”+NumTostr(reg1,0)%door_loc;
```

指令CallByVar
不能調用帶有參
數的例行程序

通過RAPID結構
仍可以調用帶有
參數的例行程序

計時指令

ClkReset

ClkStart

ClkStop

計時指令-ClkReset

ClkReset Clock;

Clock: 時鐘名稱 **(clock)**

應用：

將機器人相應時鐘復位，常用於記錄循環時間或機器人跟蹤運輸鏈。

實例：

```
ClkReset clock1;
ClkStart  clock1;
RunCycle;
ClkStop clock;
nCycleTime:=ClkRead(clock1);
TPWrite “Last cycletime:’\Num:=nCycleTime
```

計時指令-ClkStart

ClkStart　Clock;

Clock:　時鐘名稱　**(clock)**

應用：

啟動機器人相應時鐘，常用於記錄循環時間或機器人跟蹤運輸鏈，機器人時鐘啟動後，時鐘不會因為機器人停止運行或關機而停止計時，在機器人時鐘運行時，指令**ClkStop**與**ClkReset**仍起作用。

計時指令-ClkStart

實例：

```
ClkReset clock1;
ClkStart  clock1;
RunCycle;
ClkStop clock;
nCycleTime:=ClkRead(clock1);
TPWrite “Last cycletime:’\Num:=nCycleTime
```

限製：

- 機器人時鐘計時超過4,294,967秒，即49天17小時2分47秒，機器人將出錯，Error Handler代碼為ERR_OVERFLOW

計時指令-ClkStop

ClkStop Clock;

Clock: 時鐘名稱 **(clock)**

應用：

停止機器人相應時鐘，常用於記錄循環時間或機器人跟蹤運輸鏈。

實例：

```
ClkReset clock1;
ClkStart  clock1;
RunCycle;
ClkStop clock;
nCycleTime:=ClkRead(clock1);
TPWrite “Last cycletime:’\Num:=nCycleTime
```

中斷指令

CONNECT	IDelete
ISignalDI	ISignalDO
ISignalAI	ISignalAO
ISleep	IWatch
IDisable	IEnable
ITimer	

中斷指令-CONNECT

CONNECT Interrupt

WITH Trap routine

Interrupt: 中斷數據名稱 **(intnum)**

Trap routine: 中斷處理程序 **(Identifier)**

應用：

將機器人相應中斷數據連接到相應的中斷處理程序是機器人中斷功能必不可少的組成部分，必須同指令**ISignalDI**、**ISignalDO**、**ISignalAI**、**ISignalAO**、或**ITmer**聯合使用

中斷指令-CONNECT

實例：

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

中斷指令-CONNECT

限製：

●中斷數據的數據類型必須為變量(VAR)，

●一個中斷數據不允許同時連接到多個中斷處理程序，但多個中斷數據可以共享一個中斷處理程序，

●當一個中斷數據完成連接後， 這個中斷數據不允許再次連接到任何中斷處理程序，(包括已經連接的中斷處理程序)，如果需要再次連接到任何中斷處理程序，必須先使用指令IDelete將原連接去除。

中斷指令-CONNECT

Error Handler:

- **ERR_ALRDYCNT**

  中斷數據已經被連接到中斷處理程序；

- **ERR_CNTNOTVAR**

  中斷數據的數據類型不是變量；

- **ERR_INOMAX**

  沒有更多的中斷數據可以使用；

中斷指令-IDelete

IDelete　Interrupt

Interrupt: 中斷數據名稱 **(intnum)**

應用：

將機器人相應中斷數據與相應的中斷處理程序之間的連接去除；

實例：

```
…..
CONNECT IntInspect WITH  rAlarm;
ISingalDI  di01_Vacuum,0,intInspect;
…..
IDelete intInspect;
```

中斷指令-IDelete

限製：

●執行指令**IDelete**後 ，當前中斷數據的連接被完全清除，如需再次使用這個中斷數據必須重新使用指令**CONNECT**連接到相應的中斷處理程序；

●在下列情況下，中斷程序將自動去除：

-重新載入新的運行程序；

-機器人運行程序被重置，程序指針回到主程序的第一行**(Start From Beginning);**

-機器人程序指針被移到任意一個例行程序的第一行**(Move PP to Routine);**

中斷指令-ISignalDI

ISingalDI [\Single],Signal,

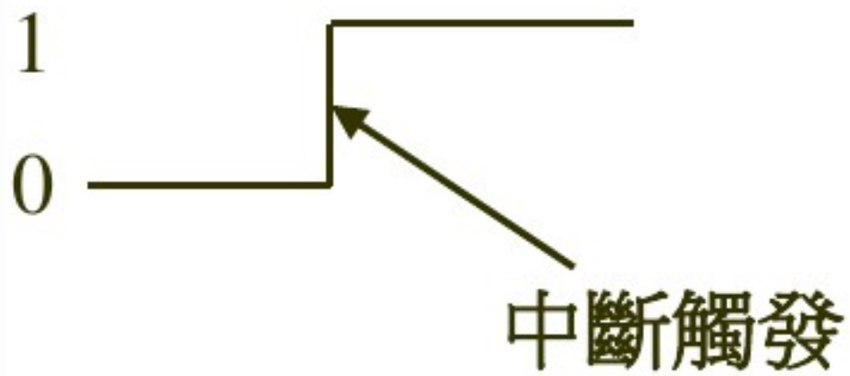
TriggValue,Interrupt

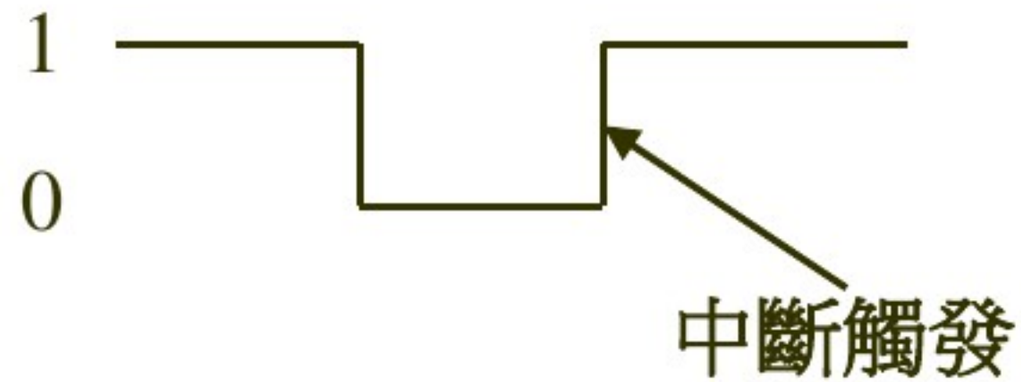
[\single]:	單次中斷信號開關	**(switch)**
Signal:	觸發中斷信號	**(singaldi)**
TriggValue:	觸發信號值	**(dionum)**
Interrupt:	中斷數據名稱	**(intnum)**

應用：

使用相應的數字輸入信號觸發相應的中斷功能，必須同指令**CONNECT**聯合使用

中斷指令-ISignalDI





實例：

…..

CONNECT int1 WITH iroutine1;

IsignalDI\single di01,1,int1; ← 中斷功能在單次觸發後失效

…..

CONNECT int2 WITH iroutine2;

ISignalDI di02,1,int1 ← 中斷功能持續有效，只有在程序重置或運行指令IDelete後才失效

中斷指令-ISignalDI

限製：

●當一個中斷數據完成連接後， 這個中斷數據不允許再次連接到任何中斷處理程序，(包括已經連接的中斷處理程序)，如果需要再次連接到任何中斷處理程序，必須先使用指令**IDelete**將原連接去除;

```
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
```

中斷指令-ISignalDO

ISingalDO [\Single],Signal,

TriggValue,Interrupt

[\single]: 單次中斷信號開關 **(switch)**

Signal: 觸發中斷信號 **(singaldi)**

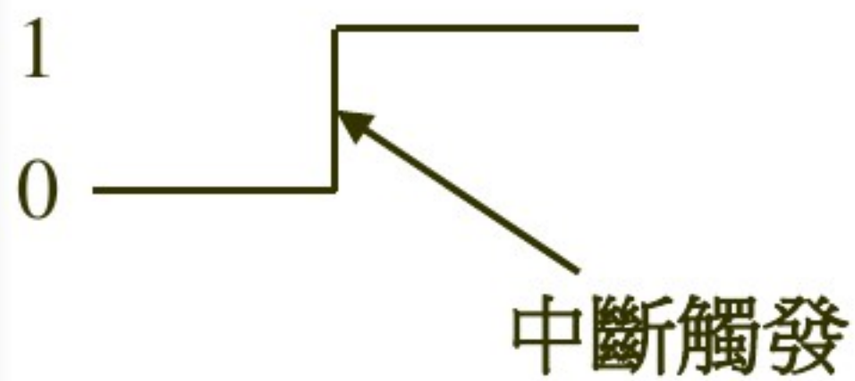
TriggValue: 觸發信號值 **(dionum)**

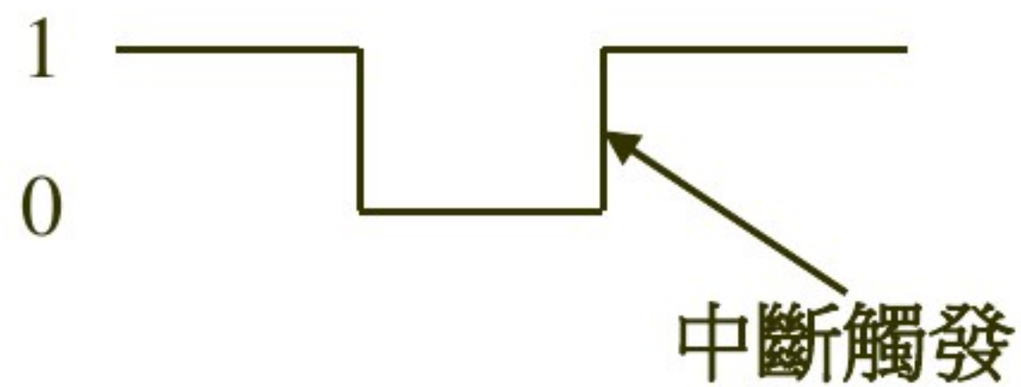
Interrupt: 中斷數據名稱 **(intnum)**

應用：

使用相應的數字輸入信號觸發相應的中斷功能，必須同指令**CONNECT**聯合使用

中斷指令-ISignalDO





實例：

…..

CONNECT int1 WITH iroutine1;

IsignalDO\single di01,1,int1; ← 中斷功能在單次觸發後失效

…..

CONNECT int2 WITH iroutine2;

ISignalDO di02,1,int1 ← 中斷功能持續有效，只有在程序重置或運行指令**IDelete**後才失效

…..

中斷指令-ISignalDO

限製：

●當一個中斷數據完成連接後， 這個中斷數據不允許再次連接到任何中斷處理程序，(包括已經連接的中斷處理程序)，如果需要再次連接到任何中斷處理程序，必須先使用指令**IDelete**將原連接去除;

```
PROC main(  )
  CONNECT int1 WITH  r1;
  ISignalDO  dO01,1,int1;
  …..
  IDelete int1;
ENDPROC
```

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

中斷指令-ISignalAI

ISingalAI [\Single],Signal,Condition,HighValue LowValeu,DeltaValue,[\Dpos][\DNeg],Interrupt

[\single]:	單次中斷信號開關	**(switch)**
Signal:	觸發中斷信號	**(singaldi)**
Condition:	中斷觸發狀態	**(aiotrigg)**
HighValue:	最大邏輯值	**(num)**
LowValue:	最小邏輯值	**(num)**

中斷指令-ISignalAI

ISingalAI [\Single],Signal,Condition,HighValue LowValeu,DeltaValue,[\Dpos][\DNeg],Interrupt

DeltaValue:	中斷恢復差值	**(num)**
[\DPos]:	正值中斷開關	**(switch)**
[\DNeg]:	負值中斷開關	**(switch)**
Interrupt:	中斷數據名稱	**(intnum)**

中斷指令-ISignalAI

中斷觸發狀態：

- **AIO_ABOVE_HIGH**
  模擬量信號邏輯值大於最大邏輯值(HighValue)
- **AIO_BELOW_HIGH**
  模擬量信號邏輯值小於最大邏輯值(HighValue)
- **AIO_ABOVE_LOW**
  模擬量信號邏輯值大於最小邏輯值(LowValue)
- **AIO_BELOW_LOW**
  模擬量信號邏輯值小於最小邏輯值(LowValue)
- **AIO_BETWEEN**
  模擬量信號邏輯值處於最小邏輯值(LowValue)和最大邏輯值(HighValue)之間

中斷指令-ISignalAI

中斷觸發狀態：

- **AIO_OUTSIDE**

  模擬量信號邏輯值大於最大邏輯值(HighValue)或者小於最小邏輯值(LowValue)

- **AIO_ALWAYS**

  總是觸發中斷，與模擬量信號邏輯值處於最小邏輯值(LowValue)與最大邏輯值(HighValue)無關

應用：

使用相應的模擬量輸入信號觸發相應的中斷功能，必須同指令CONNECT聯合使用；

中斷指令-ISignalAI

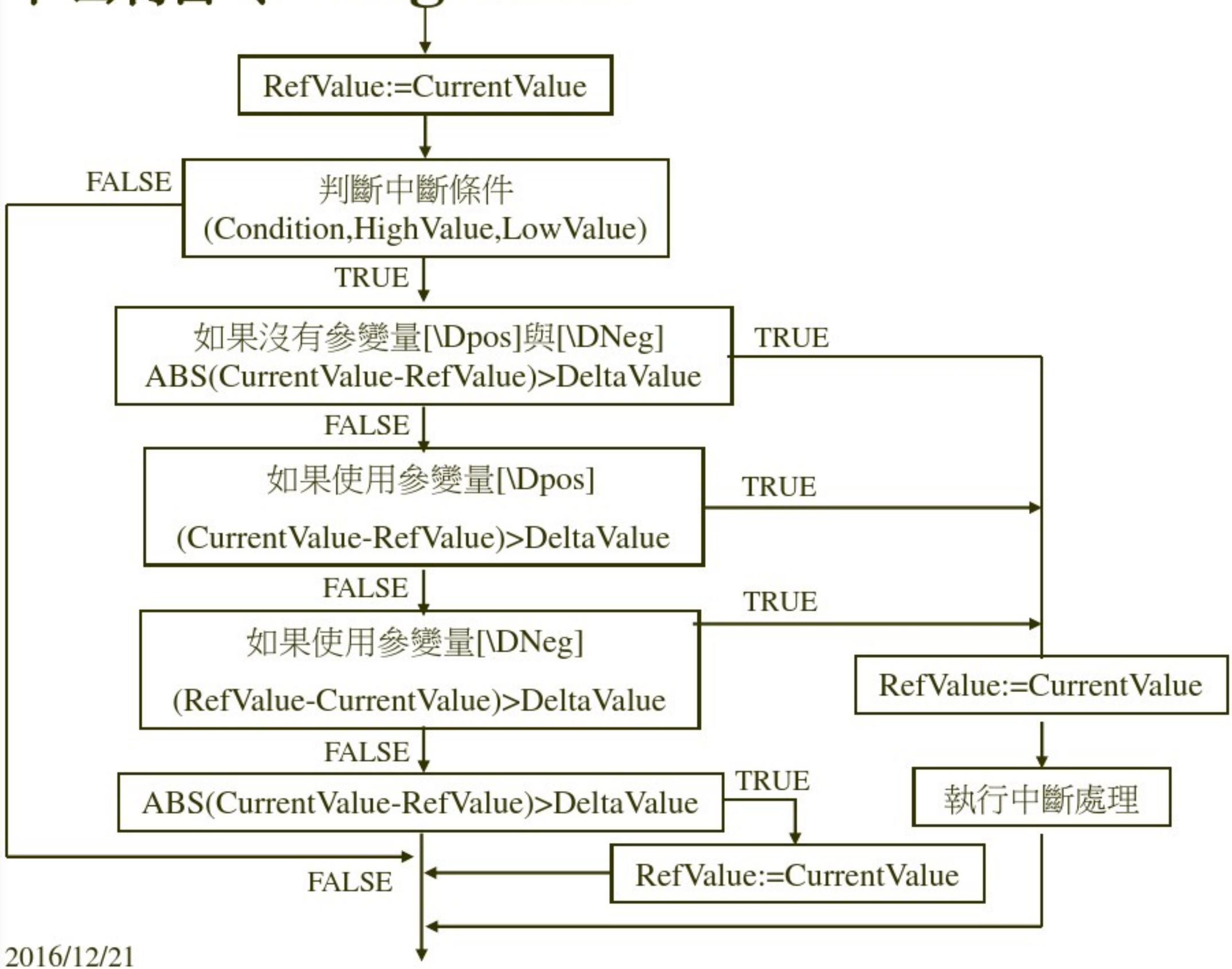
實例：

中斷功能在單次觸發後失效

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

中斷功能持續有效，只有在程序重置或運行IDelete後才失效

中斷指令-ISignalAI



中斷指令-ISignalAI

限製：

●當前最大邏輯值(HighValue)與最小邏輯值(LowValue)必須是模擬量信號所定義的邏輯值範圍內；

●最大邏輯值(HighValue)必須大於最小邏輯值(LowValue);

●中斷復位差值(DeltaValue)必須為正數或0；

●指令ISignalDI的限製，仍然適用；

中斷指令-ISignalAO

ISingalAO [\Single],Signal,Condition,HighValue LowValeu,DeltaValue,[\Dpos][\DNeg],Interrupt

[\single]:	單次中斷信號開關	**(switch)**
Signal:	觸發中斷信號	**(singaldi)**
Condition:	中斷觸發狀態	**(aiotrigg)**
HighValue:	最大邏輯值	**(num)**
LowValue:	最小邏輯值	**(num)**

中斷指令-ISignalAO

ISingalAO [\Single],Signal,Condition,HighValue LowValeu,DeltaValue,[\Dpos][\DNeg],Interrupt

DeltaValue:	中斷恢復差值	**(num)**
[\DPos]:	正值中斷開關	**(switch)**
[\DNeg]:	負值中斷開關	**(switch)**
Interrupt:	中斷數據名稱	**(intnum)**

中斷指令-ISignalAO

中斷觸發狀態：

AIO_ABOVE_HIGH

模擬量信號邏輯值大於最大邏輯值(HighValue)

AIO_BELOW_HIGH

模擬量信號邏輯值小於最大邏輯值(HighValue)

- AIO_ABOVE_LOW

模擬量信號邏輯值大於最小邏輯值(LowValue)

- AIO_BELOW_LOW

模擬量信號邏輯值小於最小邏輯值(LowValue)

- AIO_BETWEEN

模擬量信號邏輯值處於最小邏輯值(LowValue)和最大邏輯值(HighValue)之間

中斷指令-ISignalAO

中斷觸發狀態：

- **AIO_OUTSIDE**
  模擬量信號邏輯值大於最大邏輯值(HighValue)或者小於最小邏輯值(LowValue)
- **AIO_ALWAYS**
  總是觸發中斷，與模擬量信號邏輯值處於最小邏輯值(LowValue)與最大邏輯值(HighValue)無關

應用：

使用相應的模擬量輸入信號觸發相應的中斷功能，必須同指令CONNECT聯合使用；

中斷指令-ISignalAO

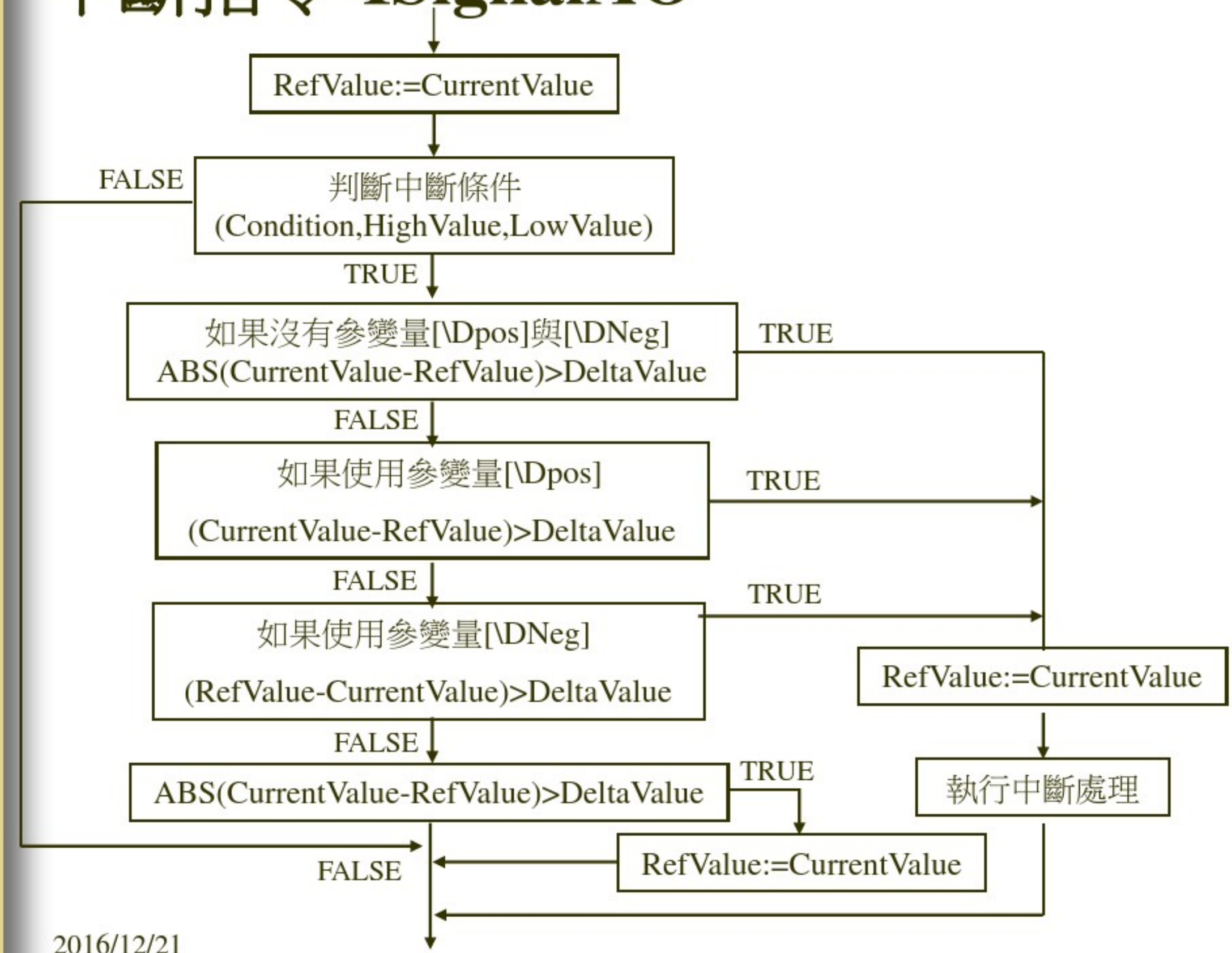
實例：

中斷功能在單次觸發後失效

```
…..
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,int1;
…..
CONNECT  int3 WITH iroutine3;
ISignalAO  ao3，AIO_BETWEEN,1.5,0.5,0.1,int3;
```

中斷功能持續有效，只有在程序重置或運行IDelete後才失效

中斷指令-ISignalAO



中斷指令-ISignalAO

限製：

●當前最大邏輯值(HighValue)與最小邏輯值(LowValue)必須是模擬量信號所定義的邏輯值範圍內；

●最大邏輯值(HighValue)必須大於最小邏輯值(LowValue);

●中斷復位差值(DeltaValue)必須為正數或0；

●指令ISignalDI的限製，仍然適用；

中斷指令-ISleep

ISleep　Interrupt;

Interrupt:　中斷數據名稱　(intnum)

應用：

　使機器人相應中斷數據暫時失效，直到執行指令**IWatch**後才恢復。

中斷指令-ISleep

實例：

…..

CONNECT intInspect WITH rAlarm

ISingalDI di01_Vacuum,0,intInspect;

← 中斷監控

…..

ISleep intInspect;

← 中斷失效

…..

IWatch intInspect;

← 中斷監控

…..

Error Handler:

- **ERR_UNKINO**

  無法找到當前的中斷數據

中斷指令-IWatch

IWatch　Interrupt;

Interrupt:　中斷數據名稱　(intnum)

應用：

　激活機器人已失效的相應中斷數據，正常情況下，與指令**ISleep**配合使用。

中斷指令-IWatch

實例：

…..

CONNECT intInspect WITH rAlarm

ISingalDI di01_Vacuum,0,intInspect;

….. ←———— 中斷監控

ISleep intInspect;

….. ←———— 中斷失效

IWatch intInspect;

….. ←———— 中斷監控

Error Handler:

- **ERR_UNKINO**

  無法找到當前的中斷數據

中斷指令-IDisable

IWatch Interrupt;

Interrupt: 中斷數據名稱 (intnum)

應用：

使機器人相應中斷功能暫時不執行，直到執行**IEnable**後，才進入中斷處理程序，此指令使用於機器人正在執行不希望被打斷的操作其間，例如：通過通信口讀寫數據。

中斷指令-IDisable

實例：

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

中斷指令-IEnable

IEnable　Interrupt;

Interrupt:　中斷數據名稱　(intnum)

應用：

　開始執行被機器人暫停的相應中斷功能，正常情況下，與指令**IDisable**配合使用，此指令用於機器人正在執行不希望被打斷的操作期間，例如：通過通信口讀寫數據。

中斷指令-IEnable

實例：

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

中斷指令-ITimer

IWatch [\Single],Time,Interrupt;

[\Single]: 單次中斷開關 **(switch)**

Time: 觸發中斷時間s **(num)**

Interrupt: 中斷數據名稱 **(intnum)**

應用：

定時處理機器人相應中斷數據，此指令常使用於通過通信口讀寫數據等場合。

中斷指令-ITimer

實例：

```
…..
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(人機對話)

TPErase	**TPWrite**
TPReadFK	**TPReadNum**
ErrWrite	**TPShow**

通信指令-TPErase

TPErase;

應用：

清屏指令，將機器人示教器屏幕上所有顯示清除，是機器人屏幕顯示重要組成部分。

實例：

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

通信指令-TPWrite

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

String: 屏幕顯示的字符串。 **(string)**
[\Num]: 屏幕顯示數字數據值 **(num)**
[\Bool]: 屏幕顯示邏輯量數據 **(bool)**
[\Pos]: 顯示位置**XYZ** **(pos)**
[\Orient]: 顯示方位**q1q2q3q4** **(orient)**

應用：

在示教器屏幕上顯示相應字符串，字符串最長**80**個字節，屏幕每行可顯示**40**個字節，在字符串後可顯示相應參變量。

通信指令-TPWrite

實例：

TPWrite string1;

TPWrite "Cycle Time= "\Num:=nTime;

限製：

- 每個**TPWrite**指令只允許單獨使用參變量，不允許同時使用；
- 參變量值**<0.000005**或**>0.999995**將圓整；

Argument	Value	Text string
\num	23	"23"
\num	1.141367	"1.141367"
\bool	TURE	"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: 數字賦值**1-5;** **(num)**
Text: 屏幕字符串； **(string)**
FKx: 功能鍵字符串； **(string)**
[\MaxTime]: 最長等待時間**s; (num)**
[\DIBreak]： 輸入信號控制； **(signaldi)**
[\BreakFlag]：指令狀態控制； **(errnum)**

通信指令-TPReadFK

應用：

在示教器屏幕上顯示相應字符串(Text),字符串最長80個字節，屏幕每行可顯示40個字節，同時在5個功能鍵上顯示相應字符串(FKx)，字符串最長7個字節，通過選擇安相應的功能鍵，給數字變量(Answer)賦值1-5，通過這種功能，當前指令可以進行數據選擇，但必須有人參與，無法達到自動化，已被輸入輸出信號代替，另外，在執行當前指令時，必須等到功能鍵輸入，才執行以后指令，除非選擇相應參變量，通過這種功能，當前指令常用于錯誤處理等場合。

通信指令-TPReadFK

實例：

```
TPReadFK  reg1,”more?”,stEmpty,stEmpty,stEmpty,
                “yes”,”no”;
```

通信指令-TPReadFK

參變量：

- **[\MaxTime]**

  機器人執行當前指令等待時間超過最長等待時間，機器人將停機報錯，如果同時采用參變量**[\BreakFlag]**，機器人將繼續執行以后指令，并且給出相應錯誤數據；

- **[\DIBreak]**

  機器人通過輸入信號來繼續執行以后指令，并且給出相應錯誤數據；

- **[\BreakFlag]**

--ERR_TP_MAXTIME

--ERR_TP_DIBREAK

通信指令-TPReadNum

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

Answer: 數字賦值**1-5;** **(num)**
String: 屏幕字符串； **(string)**
[\MaxTime]: 最長等待時間**s; (num)**
[\DIBreak]： 輸入信號控制； **(signaldi)**
[\BreakFlag]：指令狀態控制； **(errnum)**

通信指令-TPReadNum

應用：

在示教器屏幕上顯示相應字符串(String),字符串最長80個字節，屏幕每行可顯示40個字節，同時在功能鍵上顯示OK，通過數字鍵輸入相應數值，給數字變量(Answer)賦值，通過這種功能，當前指令可以進行數字數據賦值，但是必須有人參與，無法達到自動化，已被輸入輸出指令替代。

實例：

```
TPReadNum reg1,"How many units?"
FOR I FROM 1 TO reg1 DO
    produce_part;
ENDFOR
```

通信指令-TPReadNum

參變量：

- **[\MaxTime]**

  機器人執行當前指令等待時間超過最長等待時間，機器人將停機報錯，如果同時采用參變量**[\BreakFlag]**，機器人將繼續執行以后指令，并且給出相應錯誤數據；

- **[\DIBreak]**

  機器人通過輸入信號來繼續執行以后指令，并且給出相應錯誤數據；

- **[\BreakFlag]**

--ERR_TP_MAXTIME

--ERR_TP_DIBREAK

通信指令-ErrWrite

ErrWrite [\W],Header,Reason
[\RL2][\RL3] [\RL4],

[\W]:	事件記錄開關；	**(switch)**
Header:	錯誤信息標題	**(string)**
Reason:	錯誤信息原因	**(string)**
[\RL2]:	附加錯誤信息原因	**(string)**
[\RL3]:	附加錯誤信息原因	**(string)**
[\RL4]:	附加錯誤信息原因	**(string)**

通信指令-ErrWrite

應用：

在示教器屏幕上顯示標准出錯截面，錯誤代碼為80001，標題最長24個字符，原因最長40個字符，如果由多種錯誤原因，可使用參變量[\RL2] [\RL3] [\RL4],每種原因最長40個字符，使用參變量[\W],錯誤代碼為80002，并且只在事件清單中記錄，不中示教器屏幕上顯示，當前指令只顯示或記錄出錯信息，不影響機器人正常運行，在示教器屏幕上顯示的信息，需要按功能鍵OK確認并清除，如果需要影響機器人運行，使用指令Stop、EXIT、TRReadFK等 。

通信指令-ErrWrite

實例：

```
.....
ErrWrite\W,"Search error","No hit for the first search";
ErrWrite\W,"PLC error","Fatal error in PLC"\RL2:="Call service";
.....
```

限制：

- 每個**ErrWrite**指令最多能顯示**145**個字節；**(Header+Reason+\RL2+\RL3+RL4)**

通信指令-TPShow

TPShow Window;

Window:顯示相應示教器窗口； **(tpnum)**

- **TP_PROGRAM**

  自動模式下，顯示生產窗口；

  手動模式下，顯示測試窗口；
- **TP_LATEST**

  顯示當前窗口的前一個窗口；
- **TP_SCREENVIEWER**

  顯示**Screen Viewer**窗口，需要相應軟件；

通信指令-TPShow

應用：

機器人在示教器屏幕上顯示相應截面，通常情況下與機器人附加軟件**Screen Viewer**配合使用。

實例：

```
……
TPShow TP_PROGRAM;
TPShow TP_TEST
```

程序流程指令

FOR

WaitTime

IF

WHILE

Lable

WaitUntil

Compact IF

TEST

GOTO

程序流程指令-FOR

```
FOR Loop counter FROM
Start value TO End value
[STEP step value] DO
……
ENDFOR;
```

Loop counter: 循環計數標識； (Identifier)

Start value: 標識初始值； (num)

End value: 標識最終值； (num)

[Step value]: 計數更改值； (num)

程序流程指令-FOR

應用：

當前指令通過循環判斷標識從初始值逐漸更改至最終值，從而控制相應循環次數，如果不使用參變量[Step],循環標識每次更改值為1，如果使用參變量[Step],循環標識每次更改值為參變量相應設置，通常情況下，初始值，最終值與更改值為整數，循環判斷標識使用ijk等小寫字母，是標準的機器人循環指令，常在通訊口讀寫，數組數據賦值等數據處理時使用。

程序流程指令-FOR

實例：

```
 FOR i FROM 1TO 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 FROM 1 TO 2 DO
         nCount{i,j}:=0;
      ENDFOR
   ENDFOR
ENDPROC
```

程序流程指令-FOR

限製：

- 循環標識只能自動更改，不允許賦值；
- 在程序循環內，循環標識可以作為數字數據(num)使用，但只能讀取相應值，不允許賦值；
- 如果循環標識、初始值、最終值與更改值使用小數形式，必須為精確值；

程序流程指令-WaitUntil

WaitUntil [\InPos],Cond [\MaxTime][\TimeFlag]

[\InPos]:	提前量開關；	**(switch)**
Cond:	判斷條件；	**(bool)**
[\MaxTime]:	最長等待時間s;	**(num)**
[\TimeFlag]:	超時邏輯量；	**(bool)**

程序流程指令-WaitUntil

應用：

當前指令用於等待滿足相應判斷條件後，才執行以後指令，使用參變量[\InPos],機器人及其外軸必須在完全停止的情況下，才進行條件判斷，此指令比指令WaitDI的功能更廣，可以替代其所有功能；

限製：

- 當前指令在使用參變量[\InPos]時，遇到程序突然停止運行，機器人不能其保証停在最終停止點進行條件判斷。

程序流程指令-WaitUntil

實例：

```
PROC  PickPart(   )
   MoveJ pPrePick,vFastEmpty,zBig,tool1;
   (WaitDI di_Ready,1;)
   …..
ENDPROC
PROC  PickPart(   )
   MoveJ pPrePick,vFastEmpty,zBig,tool1;
   WaitUntil nCounter=4\MaxTime:=5;
   …..
ERROR
   IF  ERRNO=ERR_WAIT_MAXTIME  THEN
      TRWrite "…..";
      RETRY;
   ELSE
     RAISE;
   ENDIF
ENDPROC
```

機器人等待輸入信號，直到信號di_Ready值為1，才執行隨後指令

機器人等待相應輸入信號，如果5秒內仍沒有得到相應數據值4，自動進行Error Handler處理，如果沒有Error Handler ,機器人停機報錯

程序流程指令-WaitUntil

實例：

```
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;
   ENDWHOLE
      .....
ENDPROC
```

機器等待到位信號，如果1秒內仍沒有等到信號di_Ready值為1，機器人執行隨後指令，但此時TimeFlag值為TRUE，機器人等到di_Ready值為1，此時TimeFlag值為FALSE

程序流程指令-WaitTime

WaitTime [\InPos],Time;

[\InPos]: 程序運行提前量開關； **(switch)**

Time: 相應等待時間s; **(num)**

應用：

當前指令只用於機器人等待相應時間後，才執行以後指令，使用參變量**[\InPos]**，機器人及其外軸必須在完全停止的情況下，才進行等待時間計時，此指令會延長循環時間；

程序流程指令-WaitTime

實例：

WaitTime 3;

WaitTime\Inpos, 0.5;

WaitTime\InPos,0;

限製：

- 當前指令在使用參變量**[\InPos]**時，遇到程序突然停止運行，機器人不能保証停在最終停止點進行等待計時；
- 當前指令參變量**[\InPos]**不能與機器人指令**SoftServo**同時使用。

程序流程指令-Compact IF

IF Condition…..;

Condition: 判斷條件； **(switch)**

應用：

當前指令是指令**IF**的簡化版，判斷條件後只允許跟一句指令，如果有多句指令需要執行，必須采用指令**IF**；

實例：

IF reg1>5 GOTO next

IF counter>10 set do1;

程序流程指令- IF

IF Condition….. THEN

{ELSEIF Condition THEN…..}

[\ELSE…..]

ENDIF;

Condition: 判斷條件； **(switch)**

應用：

當前指令通過判斷相應條件，控制需要執行的相應指令，是機器人程序流程基本指令；

程序流程指令- IF

實例：

```
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
    rountin2;
ELSEIF reg2=3 THEN
    rountine3;
ELSEIF reg2=4 THEN
    rountine4;
ELSE
    error;
ENDIF
```

程序流程指令- TEST

TEST Test date

{CASE Test value ,{Test value}:…}

[DEFAULT:….}

ENDTEST

Test date: 判斷數據變量； **(All)**

Test value: 判斷數據值； **(same as)**

應用：

當前指令通過判斷相應數據變量與其所對應的值，控制需要執行的相應指令；

程序流程指令- TEST

實例：

```
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
  rountin2;
ELSEIF reg2=3 THEN
  rountine3;
ELSEIF reg2=4 or reg2=5 THEN
  rountine4;
ELSE
  error;
ENDIF
```

程序流程指令- GOTO

GOTO Label

Label: 程序執行位置標簽； **(Identifier)**

應用：

當前指令必須與指令**Label**同時使用，執行當前指令後，機器人將從相應標簽位置**Label**處繼續運行程序指令；

程序流程指令- GOTO

實例：

```
IF reg1>100 GOTO highvalue
lowvalue;
.....
GOTO ready;
highvalue;
.....
ready:
.....
reg1:=1;
next:
reg1:=reg1+1;
IF reg1<=5 GOTO next;
```

程序流程指令- GOTO

限製：

- 只能使用當前指令跳至同一例行程序內相應的位置標簽**Label;**
- 如果相應位置標簽**Label**處於指令**TEST**或**IF**內，相應指令**GOTO**必須同處於相同的判斷指令內或其分支內；
- 如果相應位置標簽**Label**處於指令**WHILE**或**FOR**內，相應指令**GOTO**必須同處於相同循環指令內；

程序流程指令- Label

Label

Label: 程序執行位置標簽； **(Identifier)**

應用：

當前指令必須與指令**GOTO**同時使用，執行當前指令**GOTO**後，機器人將從相應標簽位置**Label**處繼續運行程序指令；當前指令用后，程序內不會顯示**Label**字樣，直接顯示相應標簽。

程序流程指令- Label

實例：

```
IF reg1>100 GOTO highvalue
lowvalue;
.....
GOTO ready;
highvalue;
.....
ready:
.....
```

限制:

- 在同一例行程序內，程序位置標簽**Label**的名稱必須唯一。

程序流程指令-WHILE

WHILE Condition DO

…

ENDWHILE

…

Condition: 判斷條件； **(bool)**

應用：

當前指令通過判斷相應條件，如果符合判斷條件執行循環內指令，直至判斷條件不滿足才跳出循環指令，繼續執行循環指令以后的指令，需要注意，當前指令存在死循環。

程序流程指令-WHILE

實例：

```
WHILE reg1<reg2  DO
.....
reg1:=reg1+1;
ENDWHILE
```

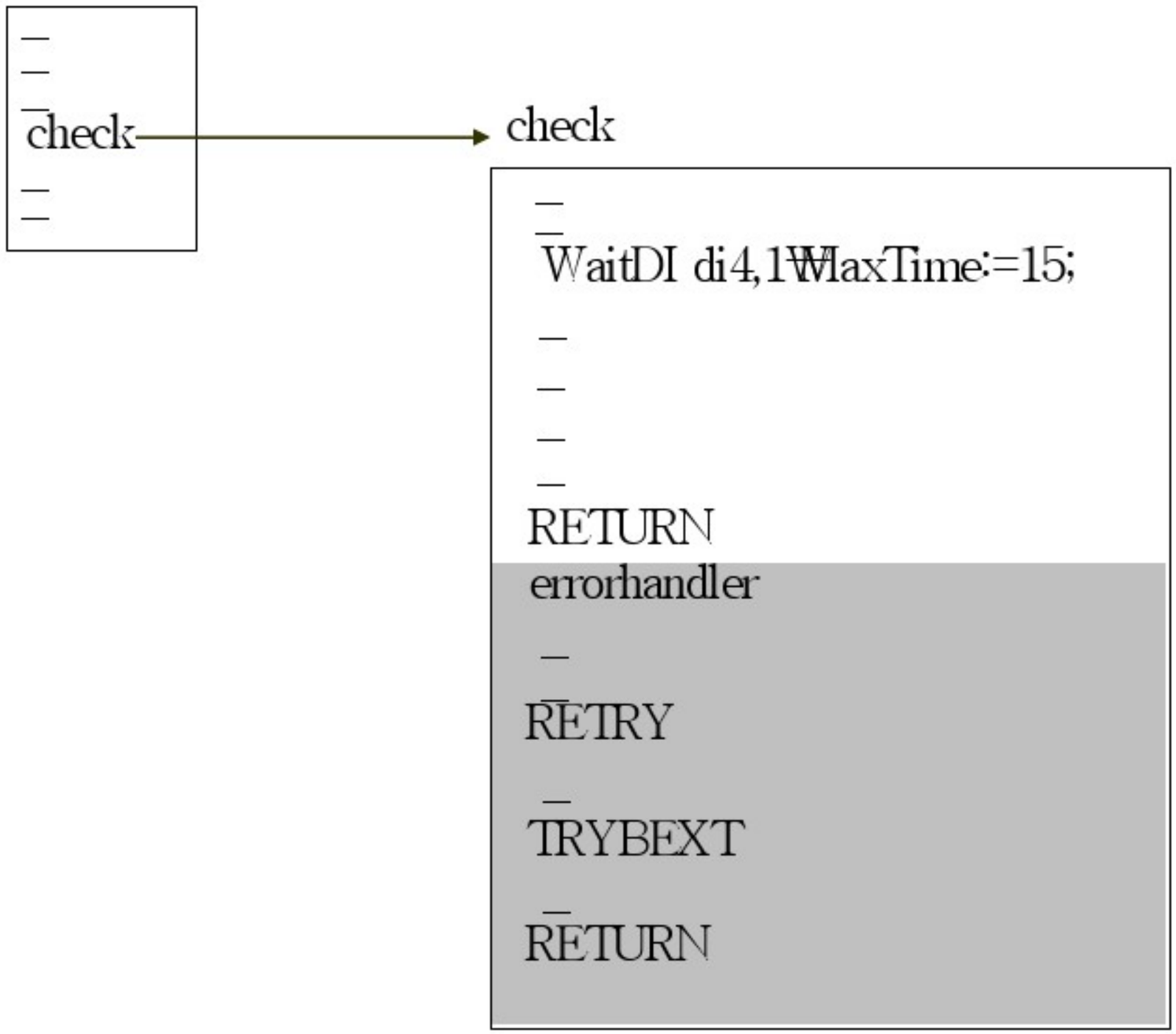
```
PROC main ( )
   rInitial;
   WHILE TRUE DO
      …
   ENDWHILE
ENDPROC
```

故障處理指令

RETREY **TRYNEXT**

RAISE **RETURN**

故障處理指令-RETRY



故障處理指令-RETRY

RETRY

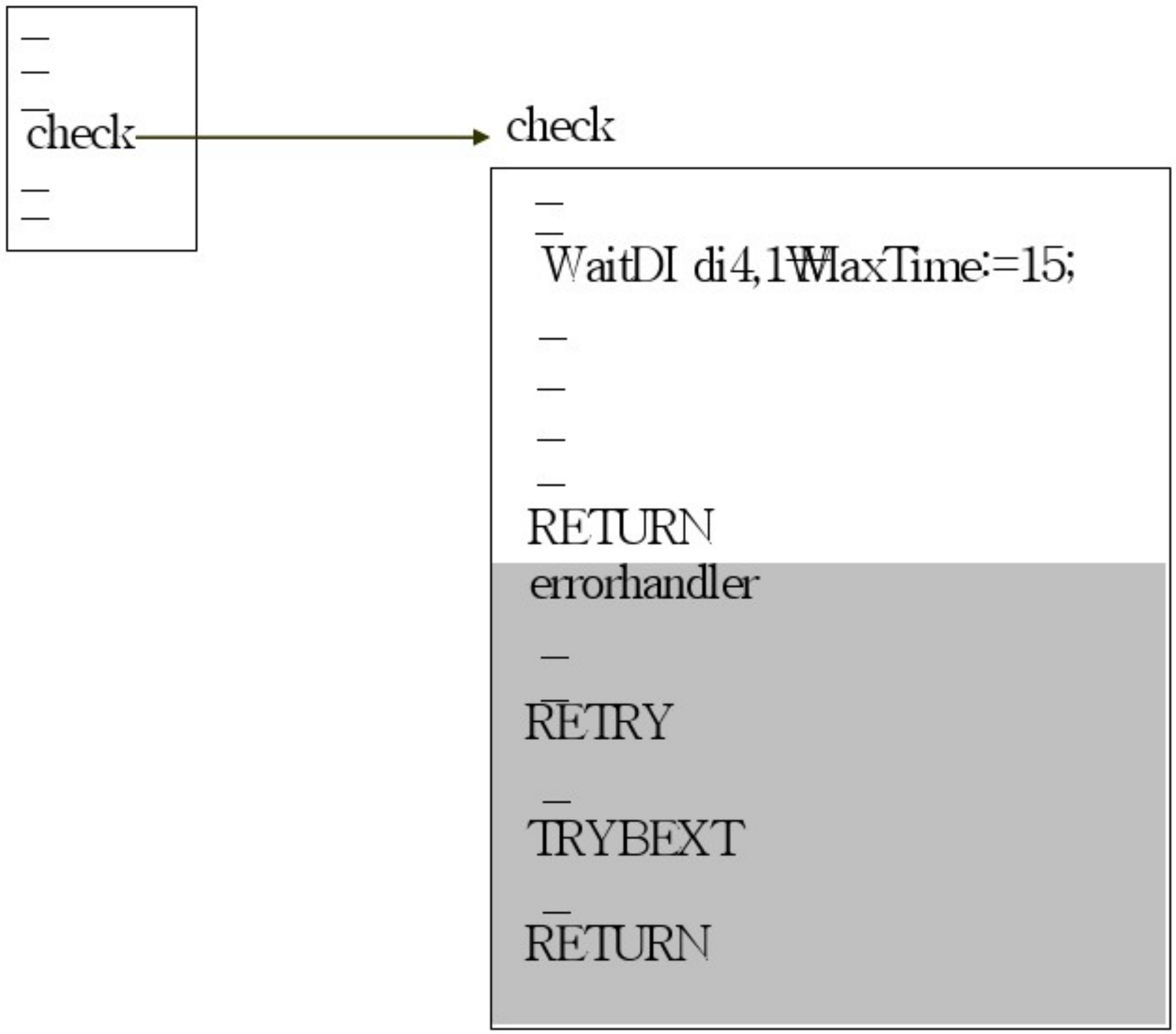
應用：

當前指令只用于機器人故障處理程序**ErrorHandler**內，經過故障處理后，再次對故障的指令進行再次運行。

限制：

- 通過當前指令對所發生故障的指令進行再次運行連續**4**次嘗試后，故障仍無法解決，機器人將停止運行，示教器顯示錯誤信息**ERR_EXCRTYMAX**

故障處理指令-TRYNEXT



故障處理指令-TRYNEXT

TRYNEXT;

應用：

當前指令只用于機器人故障處理程序**ErrorHandler**內，經過故障處理后，跳過發生故障的指令繼續運行程序。

故障處理指令-RAISE

RAISE [Error no.];

[Error no.]: 錯誤信息編號； **(errnum)**

應用：

當前指令如果不使用參變量，只用于機器人故障處理程序**ErrorHandler**內，經過故障處理后，跳至上一層例行程序故障處理程序內，繼續運行故障處理，如果使用參變量，只用于例行程序內，直接進入當前例行程序故障處理程序進行 處理。

故障處理指令-RAISE

限制：

- 參變量錯誤代碼范圍為**1-90**，超出范圍，示教器顯示錯誤信息**ERR_ILLRAISE**；
- 如果需要更多的錯誤代碼，超出**1-90**，必須使用機器人指令**BookErrNo.**。

故障處理指令-RETURN

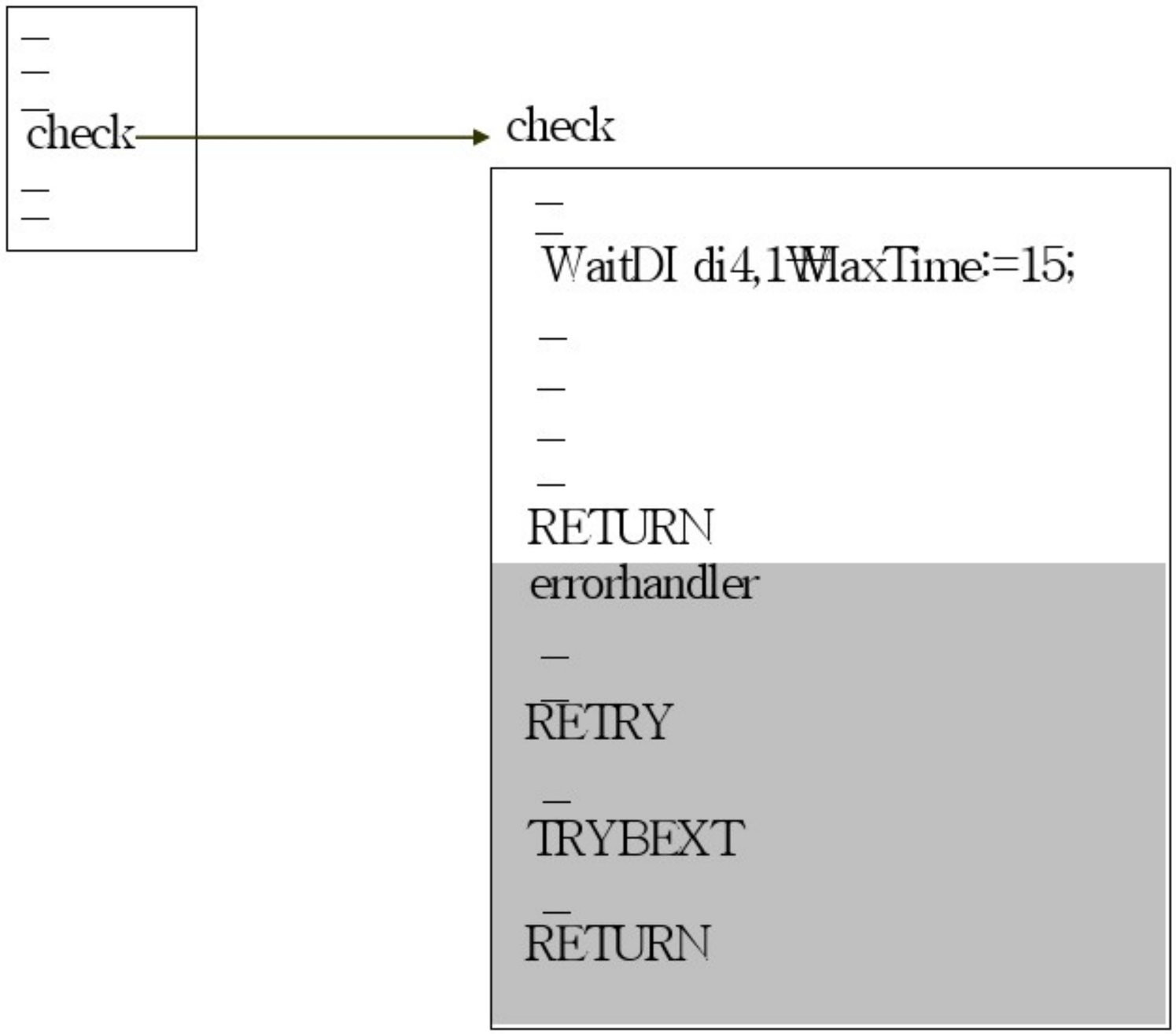
RETURN [Return value];

[Return value] : 返回時間值； **(all)**

應用：

當前指令如果使用參變量，只用于機器人函數例行程序內，經過運行返回相應的值，通常情況下，在不使用參變量時，機器人運行到此指令時，無論主程序**main ,**標准例行程序**PROC**，中斷例行程序**TRAP**，故障處理程序**ErrorHandler** 都代表當前例行程序結束。。

故障處理指令-RETURN



故障處理指令-RETURN

實例：

```
PROC rPick ( )
  …
  RETURN;
  …
ENDPROC
FUNC num abs_value( num  value)
  IF value<0 THEN
    RETURN -value;
  ELSE
    RETURN value;
  ENDIF
ENDFUNC
```

中斷運動指令

StopMove **StartMove**

StorePath **RestoPath**

中斷運動指令-StopMove

StopMove

應用：

當前指令使機器人運動臨時停止，直到運行指令**StartMove**后，才繼續恢復被臨時停止的運動，此指令通常被用于處理牽涉到機器人運動的中斷程序。

實例：

```
StopMove;
WaitDI ready_input,1;
StartMove;
```

中斷運動指令-StopMove

實例：

```
...
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
```

機器人完成當前運動指令后停止運動，并記錄運動路徑，在Home位置等待di1為0后，繼續原運動狀態；

機器人臨時停止運動，并記錄運動路徑，在Home位置等待di1為0后，繼續原運動狀態。

中斷運動指令-StartMove

StartMove

應用：

當前指令必須與指令**StopMove**聯合使用，使機器人臨時停止的運動恢復，此指令通常被用于處理牽涉到機器人運動的中斷程序。

實例：

```
StopMove;
WaitDI ready_input,1;
StartMove;
```

Error Handling

- **ERR_PATHDIST**

偏離原來路徑(大于**10mm**或**20**度)

中斷運動指令-StorePath

StorePath

應用：

當前指令用來記錄機器人當前運動狀態，通常與指令**RestoPath**聯合使用，此指令通常被用于處理牽涉到機器人運動的中斷程序。

限制：

- 當前指令只能用來記錄機器人運動路徑；
- 機器人臨時停止后，需要執行新的運動，必須記錄當前運動路徑；
- 機器人系統只能記錄一個運動路徑；

中斷運動指令-StorePath

實例：

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

機器人臨時停止運動，并記錄運動路徑，在Home位置等待di1為0后，繼續原運動狀態。

中斷運動指令-RestoPath

RestoPath

應用：

當前指令用來恢復已經被記錄的機器人運動狀態，必須與指令**StorePath**聯合使用，此指令通常被用于機器人錯誤處理與處理牽涉到運動的中斷程序。

限制：

- 當前指令只能用來恢復機器人運動路徑；
- 機器人臨時停止后，需要執行新的運動，必須記錄當前運動路徑；
- 機器人系統只能記錄一個運動路徑；

中斷運動指令-RestoPath

實例：

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

機器人臨時停止運動，并記錄運動路徑，在Home位置等待di1為0后，繼續原運動狀態。

坐標轉換指令

PDispOn

PDispOff

PDispSet

EOffsOn

EOffsOff

EOffsSet

坐標轉換指令-PDispOn

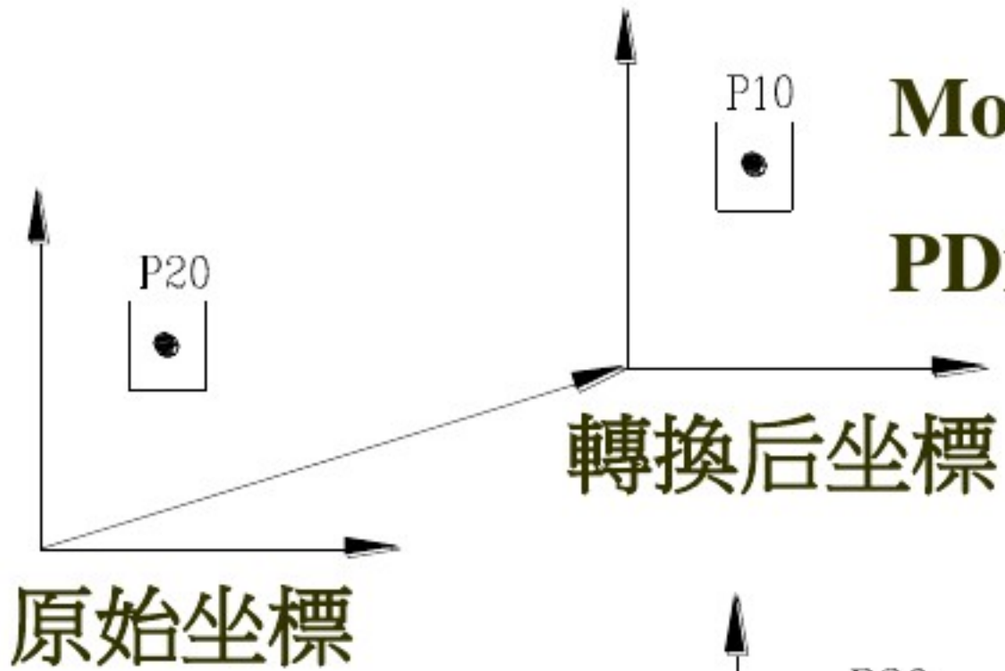
PDispOn [\Rot][\ExeP]
ProgPoint, Tool[\Wobj];

[\Rot]:	坐標旋轉開關，	**(switch)**
[\Exep]:	運行起始點，	**(robtarget)**
ProgPoint:	坐標原始點，	**(robtarget)**
Tool:	工具坐標系，	**(Tooldate)**
[\Wobj]:	工件坐標系，	**(wobjdate)**

應用：

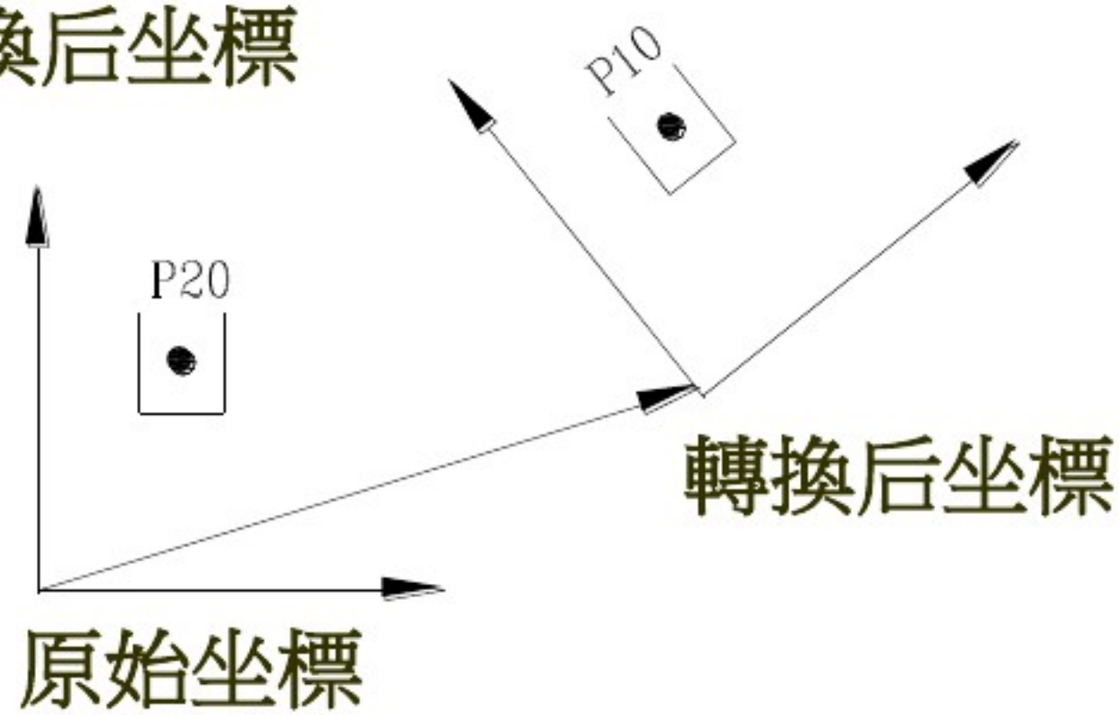
當前指令可以使機器人坐標通過編程進行即時轉換，通常用于水切割等運動軌跡保持不變的場合，可以快捷的完成工作位置修正。

坐標轉換指令-PDispOn



MoveL p10,v500,z10,tool1;

PDispOn\Exep:=p10,p20,tool1;



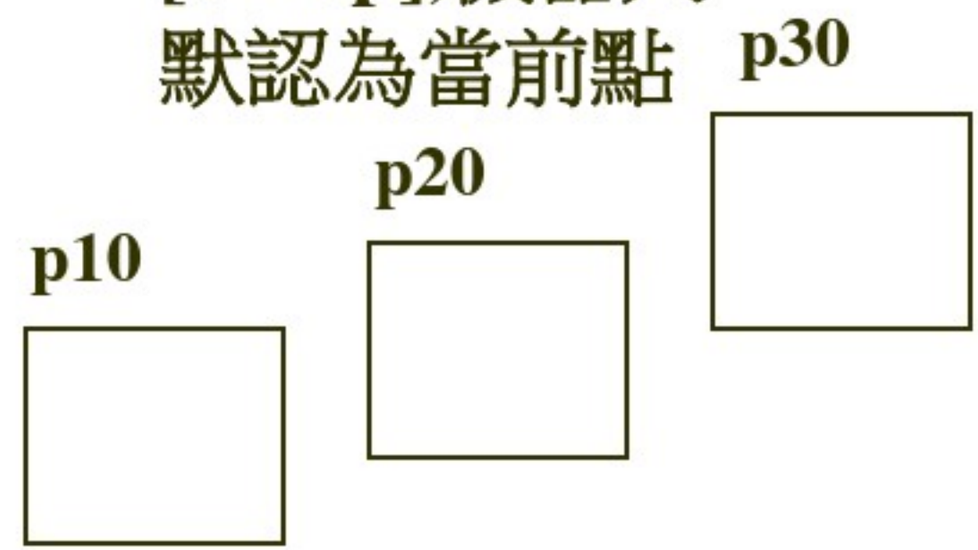
MoveL p10,v500,fine\Inpos:=inpos50,tool;

PDispOn\Rot\Exep:=p10,p20,tool1;

坐標轉換指令-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:=inpos50,tool1;
Draw_square;
MoveL p20,v500,fine\Inpos:=inpos50,tool1;
Draw_square;
MoveL p30,v500,fine\Inpos:=inpos50,tool1;
Drar_square;
```

不使用參變量
[\Exep],機器人
默認為當前點



坐標轉換指令-PDispOn

實例：

SearchL sen1,psearch,p10,v100,tool1;

PDispOn\Exep:=psearch,*,tool1;

限制：

●當前指令在使用后 ，機器人坐標將被轉換，直到使用指令**PDispOff**后才失效；

●在下列情況下，機器人坐標轉換功能將自動失效

--機器人系統冷啟動

--載入新機器人程序

--程序重置

坐標轉換指令-PDispOff

PDispOff

應用：

當前指令用于使機器人通過編程達到的坐標轉換功能失效，必須于指令**PDispOn**或**PDispSet**同時使用。

實例：

MoveL p10,v500,z10,tool1; ← 坐標轉換指令失效

PDispOn\Exep:=p10,p11,tool1;

MoveL p20,v500,z10,tool1; ← 坐標轉換指令生效

MoveL p30,v500,z10,tool1;

PDispOn

MoveL p50,v500,z10,tool1; ← 坐標轉換指令失效

坐標轉換指令-PDispSet

PDispSet DispFrame

DispFrame: 坐標偏差量 **(pose)**

應用：

當前指令通過輸入坐標偏差量，使機器人坐標通過編程進行即時轉換，通常用于水切割等運行軌跡保持不變的場合，可以快捷的完成工作位置修正。

坐標轉換指令-PDispSet

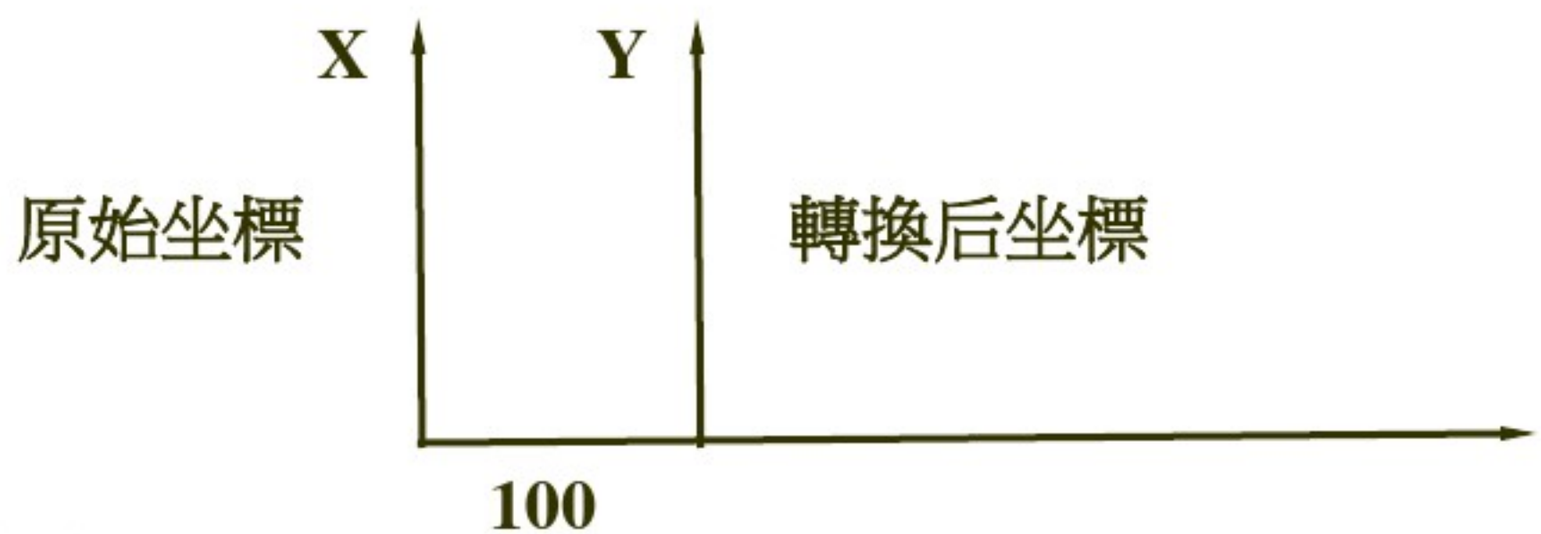
實例：

```
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;
```

坐標轉換指令失效

坐標轉換指令生效

坐標轉換指令失效



坐標轉換指令-PDispSet

限制：

- 當前指令在使用后 ，機器人坐標將被轉換，直到使用指令**PDispOff**后才失效；
- 在下列情況下，機器人坐標轉換功能將自動失效

--機器人系統冷啟動

--載入新機器人程序

--程序重置(Start From Beginning)

坐標轉換指令-EOffsOff

EOffsOff

應用：

當前指令用于使機器人通過編程達到的外軸位置更改功能失效，必須于指令**EOffsOn**或**EOffsSet**同時使用。

實例：

MoveL p10,v500,z10,tool1; ← 外軸位置更改失效

EOffsOn\Exep:=p10,p11,tool1;

MoveL p20,v500,z10,tool1; ← 外軸位置更改生效

MoveL p30,v500,z10,tool1;

EOffsOff;

MoveL p50,v500,z10,tool1; ← 外軸位置更改失效

坐標轉換指令-EOffsOn

PDispOn [\ExeP]ProgPoint;

[\Exep]: 運行起始點， **(robtarget)**

ProgPoint:坐標原始點， **(robtarget)**

應用：

當前指令可以使機器人外軸通過編程進行即時更改，帶導軌的機器人。

實例：

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

坐標轉換指令-EOffsOn

限制：

- 當前指令在使用后 ，機器人外軸位置將被更改，直到使用指令**EOffsOff**后才失效；
- 在下列情況下，機器人坐標轉換功能將自動失效

--機器人系統冷啟動

--載入新機器人程序

--程序重置(Start From Beginning)

坐標轉換指令-EOffsSet

EOffsSet EAxOffs;

EAxOffs: 外軸位置偏差量 **(extjoint)**

應用：

當前指令通過輸入外軸位置偏差量，使機器人外軸位置通過編程進行即時更改，對于導軌類外軸，偏差值單位為**mm,**對于轉軸類外軸，偏差值單位為角度。

坐標轉換指令-EOffsSet

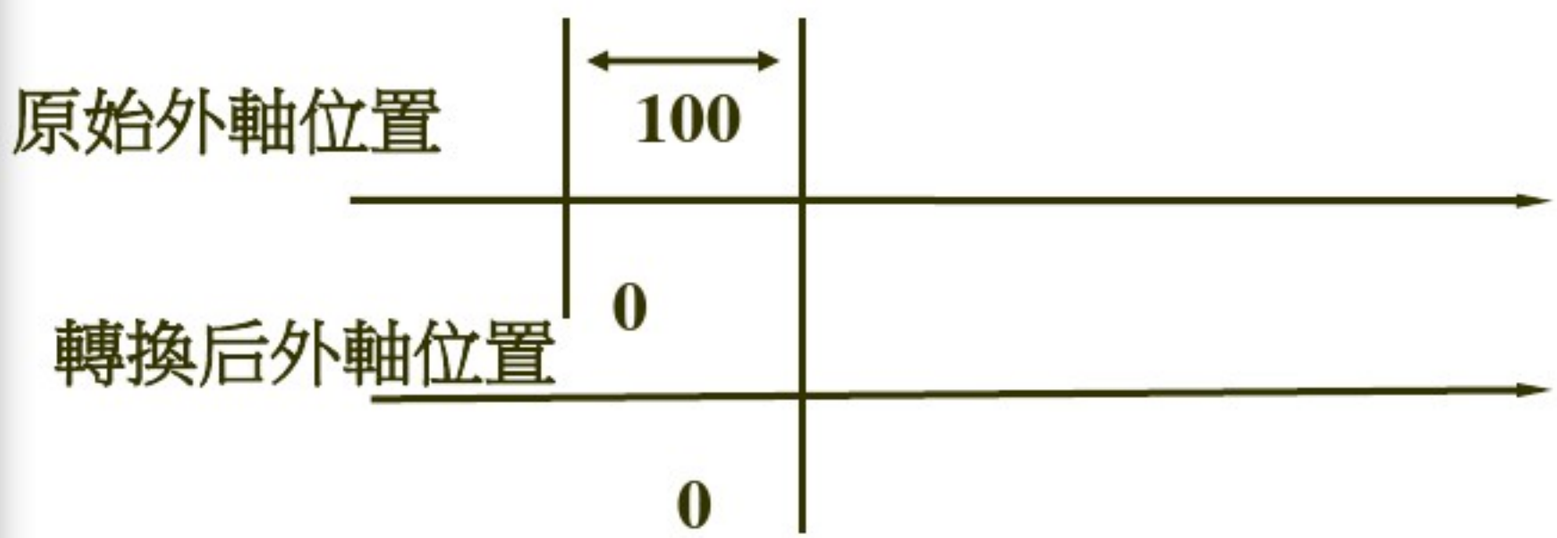
實例：

```
VAR extjoint eax_a_p100:=[[100,0,0,0,0];
MoveL p10,v500,z10,tool1;
EOffsSet eax_a_p100;;
MoveL p20,v500,z10,tool1;
EOffsOff;
MoveL p30,v500,z10,tool1;
```

外軸位置更改失效

外軸位置更改生效

外軸位置更改失效



坐標轉換指令-EOffsSet

限制：

- 當前指令在使用后 ，機器人外軸位置將被更改，直到使用指令**EOffsOff**后才失效；
- 在下列情況下，機器人坐標轉換功能將自動失效

--機器人系統冷啟動

--載入新機器人程序

--程序重置(Start From Beginning)

運動觸發指令

TriggIO

TriggInt

TriggEquipp

TriggJ

TriggL

TriggC

運動觸發指令-TriggIO

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

[\TriggDate]:	觸發變量名稱，	**(triggdate)**
Distance:	觸發距離**mm**，	**(num)**
[\Start]:	觸發起始開關，	**(switch)**
[\Time]:	時間觸發開關，	**(switch)**
[\DOp]:	觸發數字輸出，	**(signaldo)**

運動觸發指令-TriggIO

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

[\GOp]:	觸發組合輸出，	**(signalgo)**
[\AOp] :	觸發模擬輸出，	**(signalao)**
[\ProcID]:	過程處理觸發，	**(num)**
SetValue:	相應信號值，	**(num)**
[\DODelay]:	數字輸出延遲，	**(num)**

運動觸發指令-TriggIO

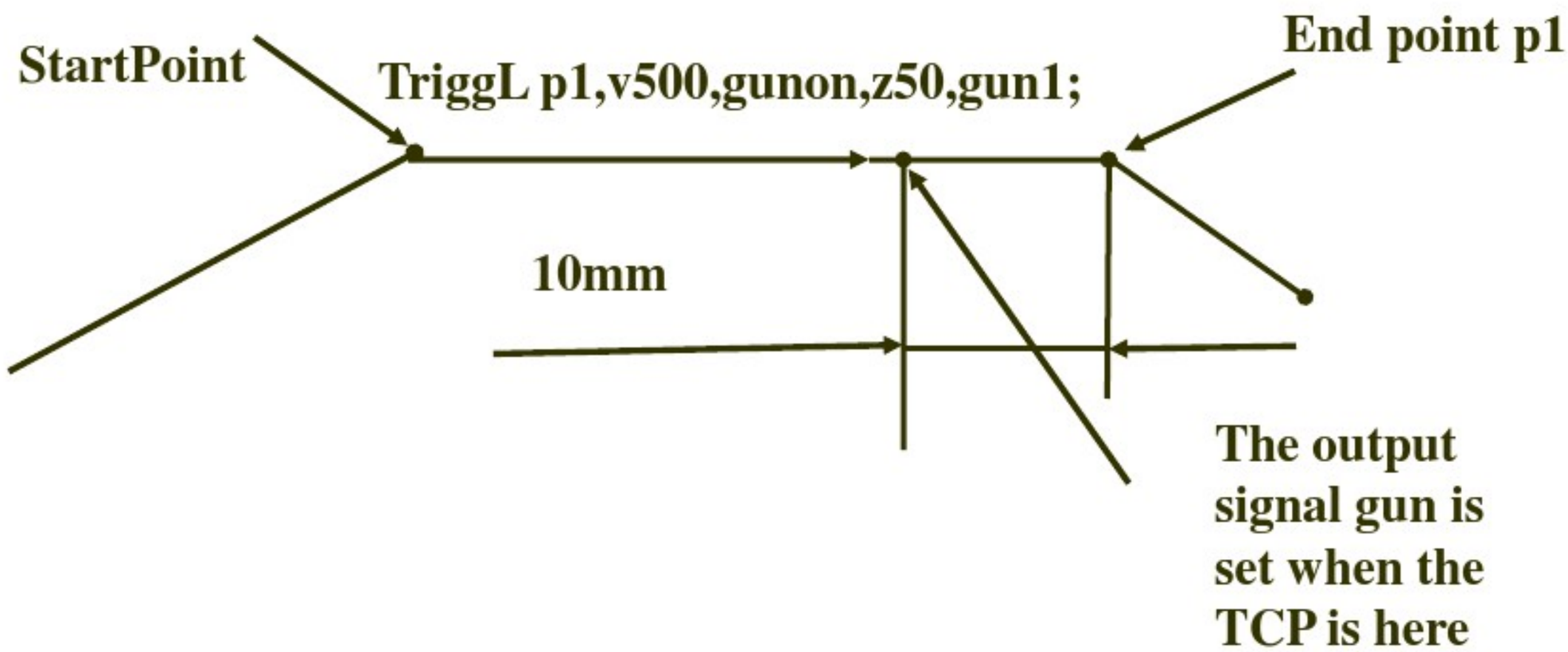
應用：

機器人可以在運動時通過觸發指令精確的輸出相應信號，當前指令用于定義觸發性質，此指令必須與其他觸發指令**TriggJ,TriggL**或**TriggC**同時使用才有意義，同機器人指令**TriggEquip**比較，多了時間控制功能，少了外部設備觸發延遲功能，通常用于噴涂、涂膠等行業，使用參變量**[\Start]**，表示以運動起始點觸發基准點，默認為運動終止點，使用參變量**[\Time],**以時間來控制觸發，允許最大時間為**0.5s,**詳見限制，參變量**[\ProcID],**正常情況用戶無法自行使用，此參變量用于**IPM**過程處理。

運動觸發指令-TriggIO

實例：

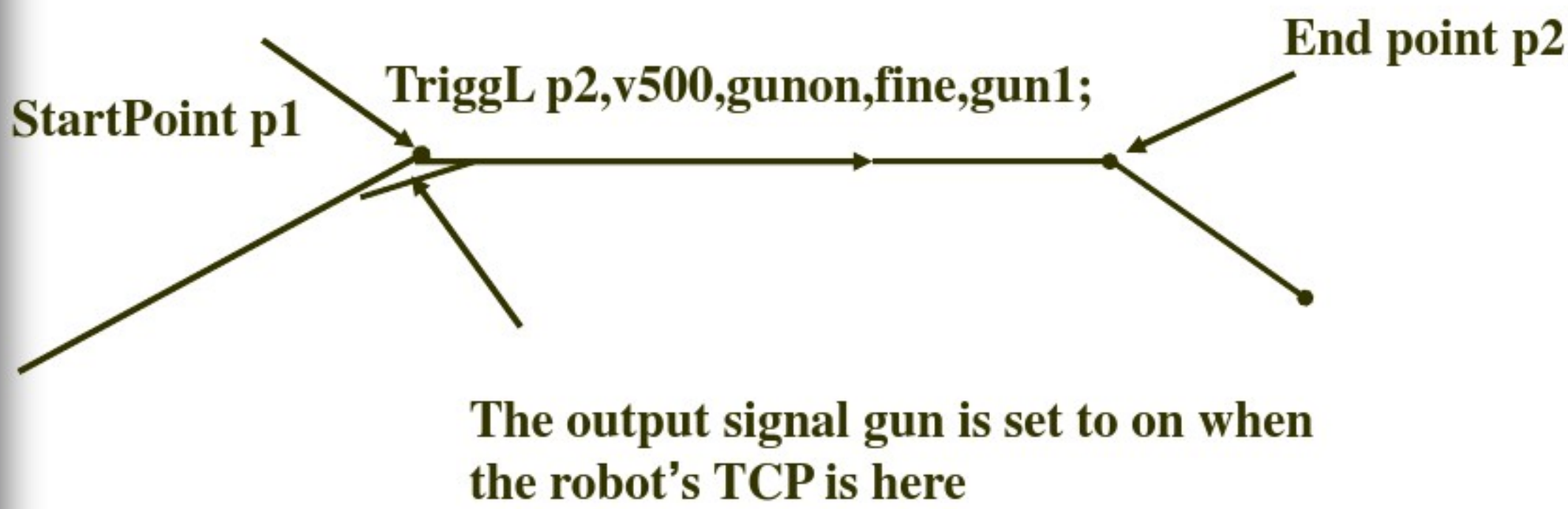
```
VAR triggdate gunon;
TriggIO gunon,10\Dop:=gun,1;
TriggL p1,v500,gunon,z50,gun1;
```



運動觸發指令-TriggIO

實例：

```
VAR triggdate gunon;
TriggIO gunon,0\Start\Dop:=gun,1;
MoveJ p1,v500,z50,gun1;
TriggL p1,v500,gunon,z50,gun1;
```



運動觸發指令-TriggIO

限制：

- 當前指令使用參變量[\Time]可以提高信號輸出精度，此參變量以目標點為基准，使用固定的目標點fine比轉角zone精度高，一般情況下，此 參變量采用固定目標點；
- 參變量[\Time]設置的時間應小于機器人開始減速時間(最大0.5s),例如：運行速度 為500mm/s, IRB2400為150ms,IRB6400為250ms,在設置時間超過減速時間的情況下，實際控制時間會縮短，但不會對正常的運行造成影響；

運動觸發指令-TriggInt

TriggInt TriggDate,Distance [\Start][\Time],Interrupt;

[\TriggDate]:	觸發變量名稱，	**(triggdate)**
Distance:	觸發距離**mm**，	**(num)**
[\Start]:	觸發起始開關，	**(switch)**
[\Time]:	時間觸發開關，	**(switch)**
interrupt:	觸發中斷名稱，	**(intnum)**

運動觸發指令-TriggInt

應用：

機器人可以在運動時通過觸發指令精確的進入中斷處理，當前指令用于定義觸發性質，此指令必須與其他觸發指令**TriggJ,TriggL**或**TriggC**同時使用才有意義，通常用于噴涂、涂膠等行業，使用參變量**[\Start]**，表示以運動起始點觸發基准點，默認為運動終止點，使用參變量**[\Time],**以時間來控制觸發，允許最大時間為**0.5s,**詳見限制。

限制：

- 正常情況下，當前指令從觸發中斷到得到響應，有**5-120ms**的延遲，用指令**TriggIO**或**TriggEquip**控制信號輸出效果更佳。

運動觸發指令-TriggInt

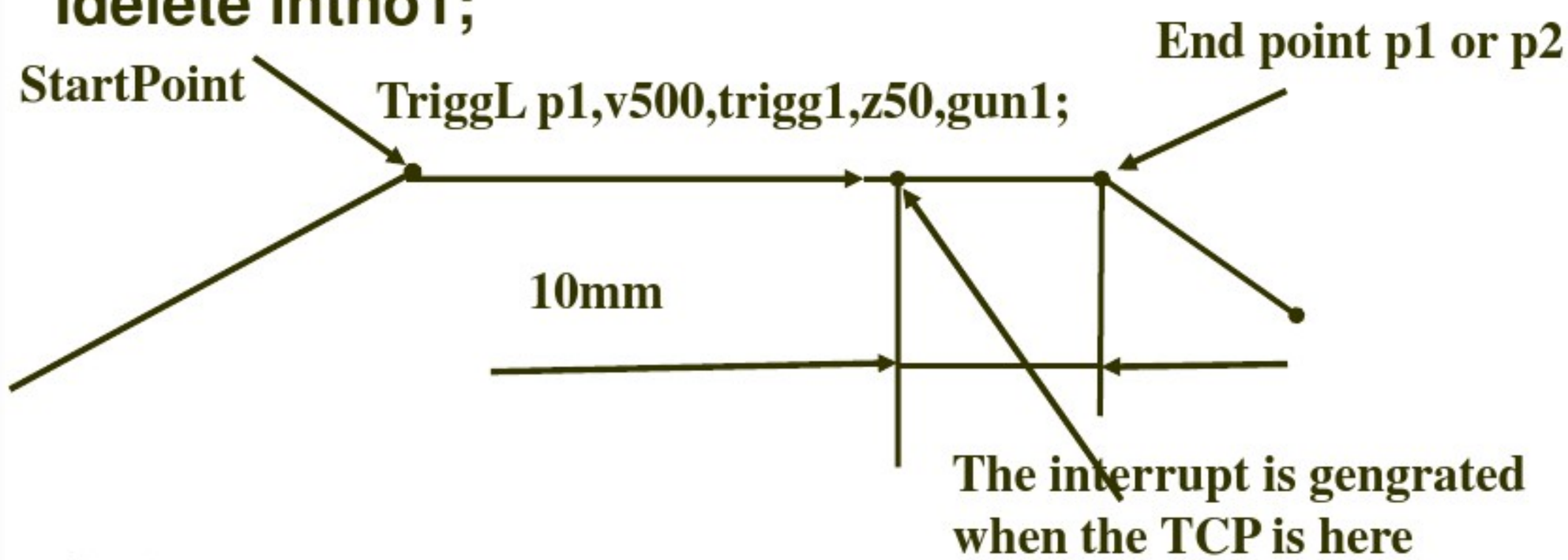
限制：

- 當前指令使用參變量[\Time]可以提高中斷觸發精度，此參變量以目標點為基准，使用固定的目標點fine比轉角zone精度高，一般情況下，此 參變量采用固定目標點；
- 參變量[\Time]設置的時間應小于機器人開始減速時間(最大0.5s),例如：運行速度 為500mm/s, IRB2400為150ms,IRB6400為250ms,在設置時間超過減速時間的情況下，實際控制時間會縮短，但不會對正常的運行造成影響；

運動觸發指令-TriggInt

實例：

```
VAR intnum intno1;
VAR triggdate Trigg1;
CONNECT intno1 WITH trap1;
TriggInt trigg1,5,intno1;
TriggL p1,v500,trigg1,z50,gun1;
TriggL p2,v500,gunon,z50,gun1;
Idelete intno1;
```



運動觸發指令-TriggEquip

TriggEquip TriggDate,Distance [\Start],EquipLag,[\Dop][\Gop] [\Aop][\ProcID],SetValue,[\Inhib];

[\TriggDate]:	觸發變量名稱，	**(triggdate)**
Distance:	觸發距離**mm**，	**(num)**
[\Start]:	觸發起始開關，	**(switch)**
EquipLag:	觸發延遲補償**s**，	**(num)**
[\DOp]:	觸發數字輸出，	**(signaldo)**

運動觸發指令-TriggEquip

TriggEquip TriggDate,Distance [\Start],EquipLag,[\Dop][\Gop] [\Aop][\ProcID],SetValue,[\Inhib];

[\GOp]:	觸發組合輸出，	**(signalgo)**
[\AOp] :	觸發模擬輸出，	**(signalao)**
[\ProcID]:	過程處理觸發，	**(num)**
SetValue:	相應信號值，	**(num)**
[\Inhib]:	信號抑止數據，	**(bool)**

運動觸發指令-TriggEquip

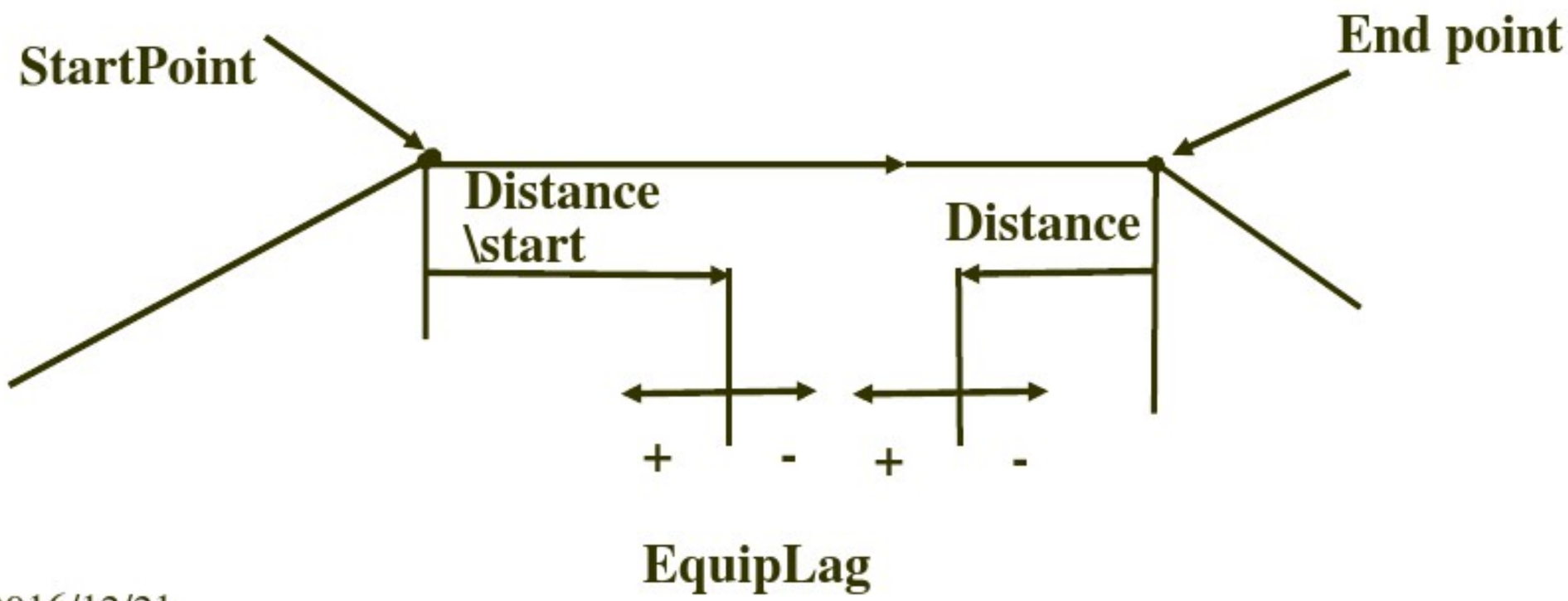
應用：

機器人可以在運動時通過觸發指令精確的輸出相應信號，當前指令用于定義觸發性質，此指令必須與其他觸發指令**TriggJ,TriggL**或**TriggC**同時使用才有意義，同機器人指令**TriggIO**比較，多了外部設備觸發延遲功能，少了時間控制功能，通常用于噴涂、涂膠等行業。

運動觸發指令-TriggEquip

應用：

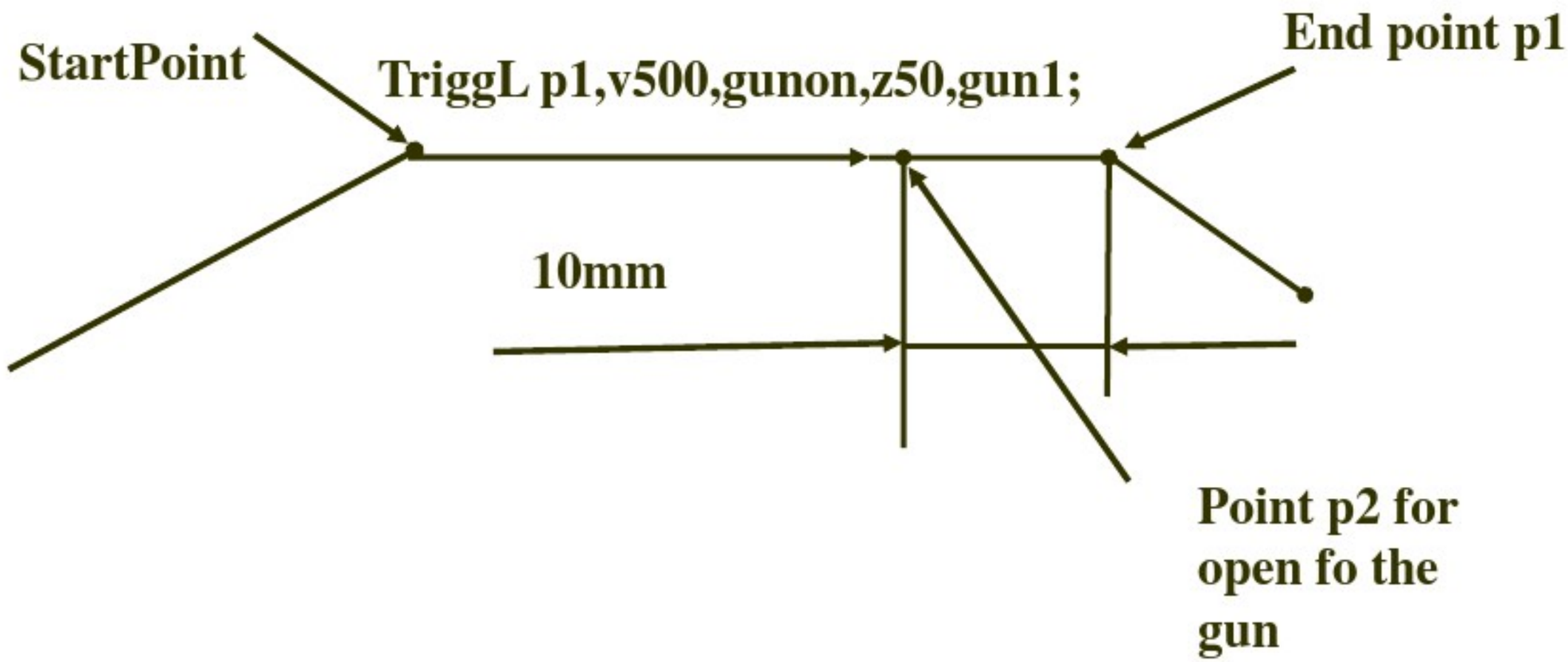
使用參變量[\Start]，表示以運動起始點觸發基准點，默認為運動終止點，參變量[\ProcID],正常情況用戶無法自行使用，此參變量用于IPM過程處理,當參變量[\Inhib]值為TRUE，在觸發點所有輸出信號(AO，GO，DO)將被置為0。



運動觸發指令-TriggEquip

實例：

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



運動觸發指令-TriggEquip

限制：

- 當前指令通過觸發延遲可以提高信號輸出精度，設置的時間應小于機器人開始減速的時間(最大0.5s),例如：運行速度 為500mm/s, IRB2400為150ms,IRB6400為250ms,在設置時間超過減速時間的情況下，實際時間會縮短，但不會對正常的運行造成影響；
- 觸發延遲EquipLag值應小于系統參數內Event Preset Time配置值，默認為60ms;
- 如果觸發延遲EquipLag值大于系統參數內Event Preset Time配置值，需要使用指令SingArea\Wrist。

運動觸發指令-TriggJ

TriggJ [\Conc]ToPoint speed [\T] Trigg_1[\T2][\T3][\T4] Zone Tool [\Wobj]

Trigg_1: 觸發變量名稱， **(triggdate)**
[\T2] : 觸發變量名稱， **(triggdate)**
[\T3]: 觸發變量名稱， **(triggdate)**
[\T4]: 觸發變量名稱, **(triggdate)**

運動觸發指令-TriggJ

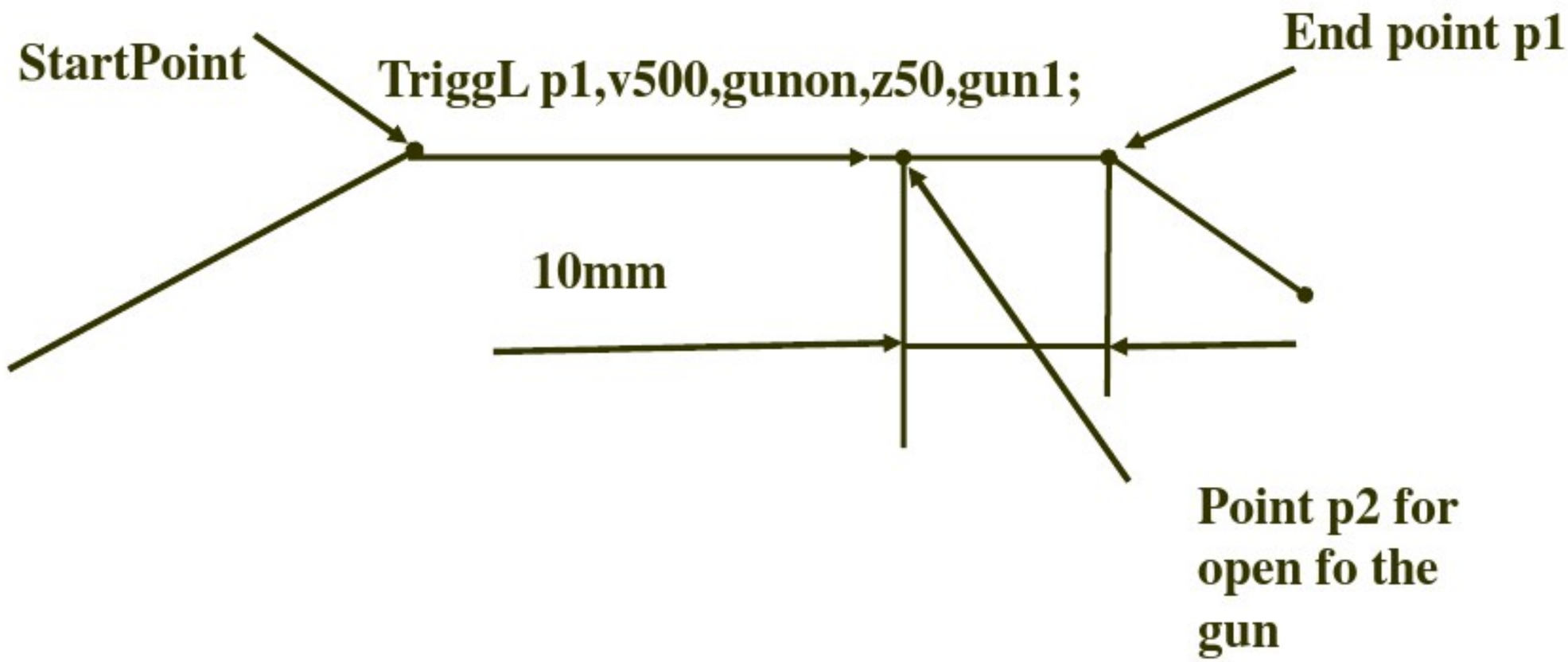
應用：

機器人可以在運動時通過該觸發指令在確定的位置輸出某個信號或出發某個中斷，同時機器人進行一個圓滑的過度路徑；需與指令**TriggIO,TriggInt**，**TriggEquip**等聯合使用才有意義，總共可定義**4**個觸發事件；

運動觸發指令-TriggJ

實例：

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



運動觸發指令-TriggJ

限制：

- 當前指令通過觸發延遲可以提高信號輸出精度，設置的時間應小于機器人開始減速的時間(最大0.5s),例如：運行速度 為500mm/s, IRB2400為150ms,IRB6400為250ms,在設置時間超過減速時間的情況下，實際時間會縮短，但不會對正常的運行造成影響；
- 觸發延遲EquipLag值應小于系統參數內Event Preset Time配置值，默認為60ms;
- 如果觸發延遲EquipLag值大于系統參數內Event Preset Time配置值，需要使用指令SingArea\Wrist。