中断程序TRAP

在RAPID程序执行过程中，如果出现需要紧急处理的情况，机器人会中断当前的执行，程序指针pp马上跳转到专门的程序中对紧急的情况进行相应的处理，处理结束后程序指针pp返回到原来被中断的地方，继续往下执行程序。这种专门用来处理紧急情况的专门程序，称作中断程序（TRAP）。

中断程序经常会用于出错处理、外部信号的响应这种实时响应要求高的场合。

现以对一个传感器的信号进行实时监控为例编写一个中断

程序：

在正常情况下，di1的信号为0

如果di1的信号从0变为1，就对reg1数据进行加1的操作

中断程序TRAP

- 中断的设定
- 中断的控制


1、中断设定

指令	说明
CONNECT	连接一个中断符号到中断程序
ISignalDI	使用一个数字输入信号触发中断
ISignalDO	使用一个数字输出信号触发中断
ISignalGI	使用一个组输入信号触发中断
ISignalGO	使用一个组输出信号触发中断
ISignalAI	使用一个模拟输入信号触发中断
ISignalAO	使用一个模拟输出信号触发中断
ITimer	计时中断
TriggInt	在一个指定的位置触发中断
IPers	使用一个可变量触发中断
IError	当一个错误发生时触发中断
IDelete	取消中断

(1)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。


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
```

限制：

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将原连接去除。

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

(2)ISignalDI

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

[\Single]： 单次中断开关。 （switch）

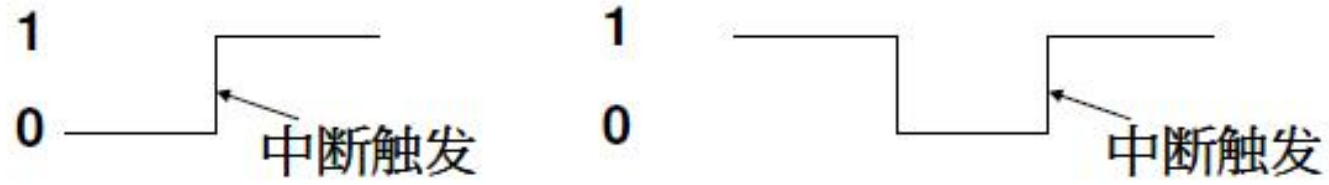
Signal： 触发中断信号。 （signaldi）

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。



实例：

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

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Idelete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
```

(3)ISignalDO

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

[\Single] : 单次中断开关。 (switch)

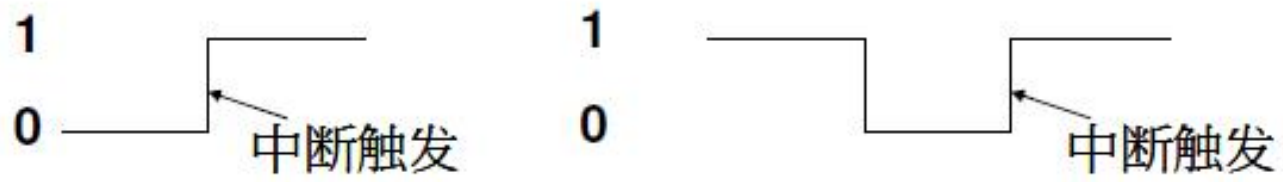
Signal : 触发中断信号。 (signaldi)

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。



实例：

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

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Idelete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
```

(4)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）或小于最小逻辑值（Low Value）

AIO_ALWAYS

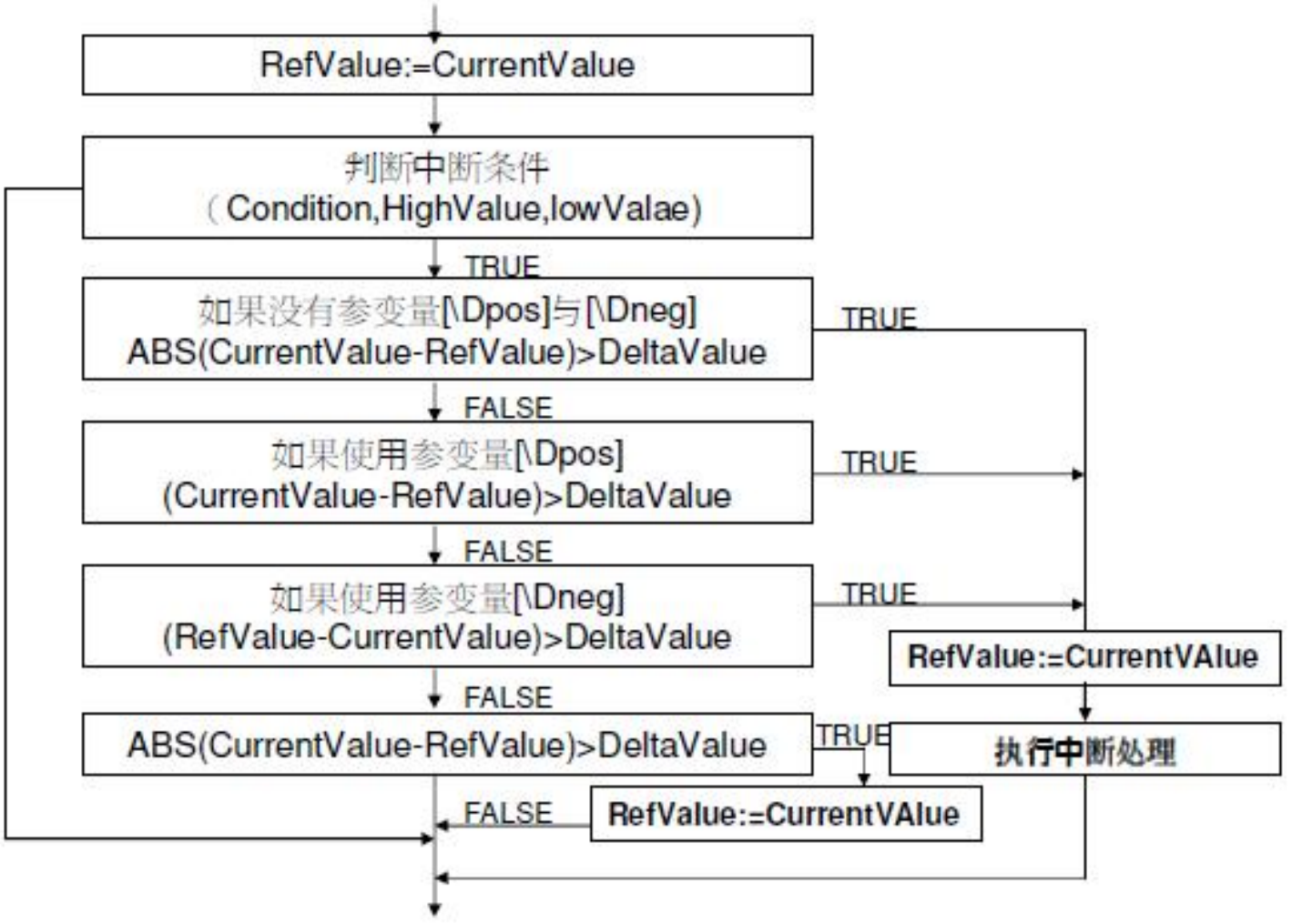
总是触发中断，与模拟量信号逻辑值处于最小逻辑值（Low Value）与最大逻辑值（High Value）无关

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。

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;
```


RefValue:=CurrentValue
判断中断条件
（Condition,HighValue,lowValae)
TRUE
如果没有参变量[\Dpos]与[\Dneg]
ABS(CurrentValue-RefValue)>DeltaValue
TRUE
FALSE
如果使用参变量[\Dpos]
(CurrentValue-RefValue)>DeltaValue
TRUE
FALSE
如果使用参变量[\Dneg]
(RefValue-CurrentValue)>DeltaValue
TRUE
RefValue:=CurrentVAlue
FALSE
ABS(CurrentValue-RefValue)>DeltaValue
TRUE
执行中断处理
FALSE
RefValue:=CurrentVAlue

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）必须为整数或0

指令ISignalDIr的限制，仍适用。

(5)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）或小于最小逻辑值（Low Value）

AIO_ALWAYS

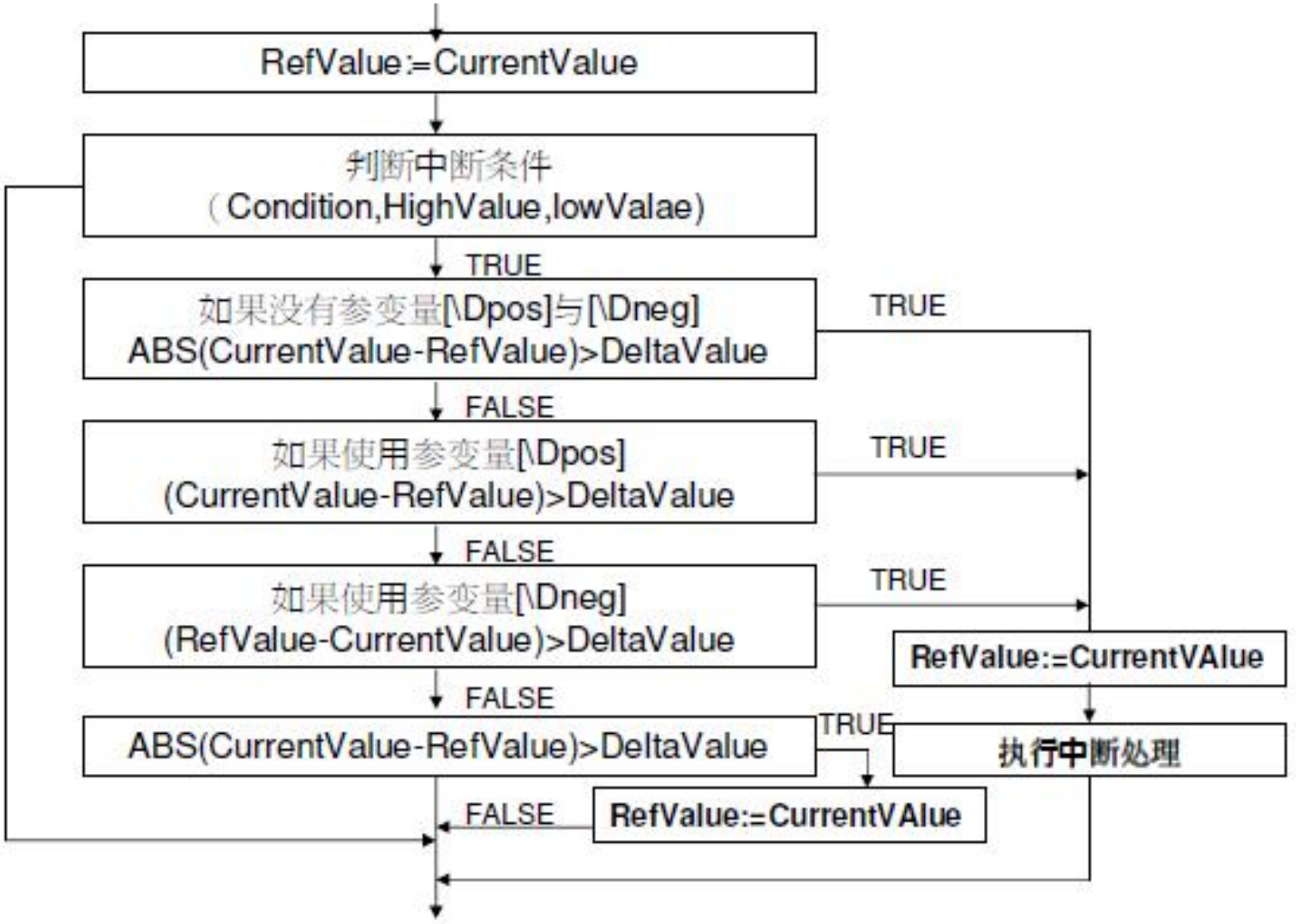
总是触发中断，与模拟量信号逻辑值处于最小逻辑值（Low Value）与最大逻辑值（High Value）无关

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。

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;
```


RefValue:=CurrentValue
判断中断条件
(Condition,HighValue,lowValae)
TRUE
如果没有参变量[\Dpos]与[\Dneg]
ABS(CurrentValue-RefValue)>DeltaValue
TRUE
FALSE
如果使用参变量[\Dpos]
(CurrentValue-RefValue)>DeltaValue
TRUE
FALSE
如果使用参变量[\Dneg]
(RefValue-CurrentValue)>DeltaValue
TRUE
RefValue:=CurrentVAlue
FALSE
ABS(CurrentValue-RefValue)>DeltaValue
TRUE
执行中断处理
FALSE
RefValue:=CurrentVAlue

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）必须为整数或0
- 指令ISignalDIr的限制，仍适用。

(5)ITimer

Itimer [\Single],Time,Interrupt;

[\Single] : 单次中断开关。 (swtich)

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，此指令常用于通过通信口读写数据等场合。

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
```

(6)TriggInt

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

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

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。

限制：

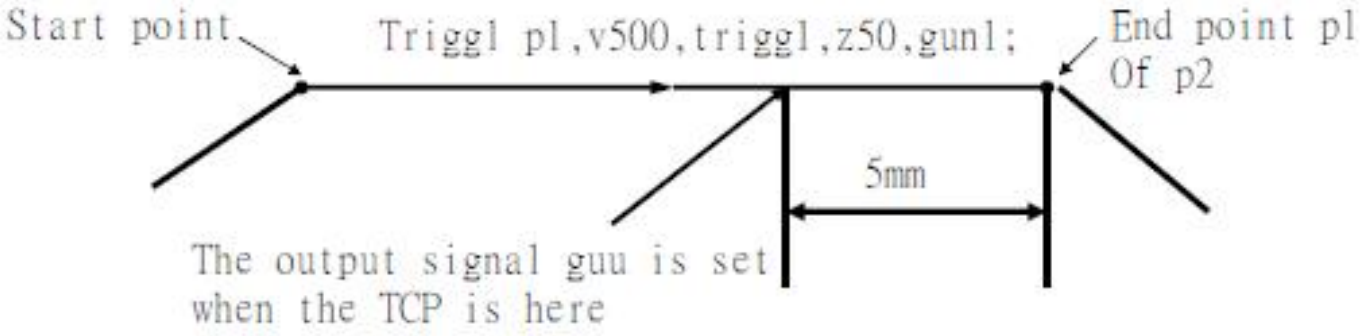
正常情况下，当前指令从触发中断到得到响应，有5-120ms延迟，用指令TriggIO或TriggEquit控制信号输出效果最佳。

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。

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;
```



(7)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;
```

限制：

执行指令Idelete后，当前中断数据的连接被完全清除，如需要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。

在以下情况下，中断将被自动去除：

（1）重新载入新的运行程序

（2）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）

（3）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）

2、中断的控制

指令	说明
ISleep	关闭一个中断
IWatch	激活一个中断
IDisable	关闭所有中断
IEnable	激活所以中断

(1)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。

实例：

```
...
CONNECT intInspect WITH rAlarm;
ISignalDI di01_vacuum,0,intInspect;
...
...
ISleep intInspect;
... 
IWatch intInspect;
... 
Error Handler :
ERR_UNKNO
```

中断监控

中断失效

中断监控

(2)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。

实例：

```
...
CONNECT intInspect WITH rAlarm;
ISignalDI di01_vacuum,0,intInspect;
...
```
← 中断监控
```
...
ISleep intInspect;
...
```
← 中断失效
```
IWatch intinspect;
...
```
← 中断监控
```
Error Handler :
ERR_UNKINO
```

(3)IDisable和IEnable

应用：

使机器人相应中断功能暂时不执行，直到执行指令IEnable，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。

实例：

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