SUSTAINABLE DEVELOPMENT GROUP

SHARE WELL

用戶手冊

指令應用篇

蘇州協緯自動化科技有限公司

SUZHOU SHARE WELL AUTOMATION TECHNOLOGY CO.,LTD

目錄

PowerNexOS 指令一覽

通用語句

Call	調用函數
Do..Loop	迴圈語句，根據條件在 Do..Loop 之間反復執行
Else/Elself	搭配判斷語句使用，If 不成立則執行 Else...Endlf 之間的代碼
Exit	強制結束迴圈或函數
For..To..Next	反復執行一定次數 For...Next 之間的代碼
Function..Fend	函數定義
Global	聲明全局變數
GoSub..Return	跳轉到副程式
GoTo	跳轉到標籤處
If..Then..Else..Endlf	判斷語句，判斷 If 成立則執行 Then...Endlf 之間的代碼
Select..Case..Default..Send	根據 Case 結果執行對應語句

數據類型

Boolean	布爾類型，True/False
Byte	位元組類型，-128~127
Double	雙精度浮點數
Int32	32 位整型，-2147483648~2147483647
Integer	16 位整形，-32768~32767
Long	長整型，-2147483648~2147483647
Real	單精確度浮點數
Short	短整型，-32768~32767
String	字串，最多包含 255 個字元
UByte	無符號位元組，0~255
UInt32	無符號 32 位整形，0~4294967295
UShort	無符號短整形，0~65535
Preserve	將全局變數定義為掉電保持狀態，停止程式或重啟控制器保持變數記憶。

運動指令

!...!	並行處理運動過程中 I/O 的輸入輸出
Motor	整體上下使能
ServoOff	單軸下使能
ServoOn	單軸上使能
Power	設置高低功率模式
SpeedFactor	用於設置/顯示全局運行速度，會影響 Speed 和 SpeedS
Speed	用於設置/顯示 Go、Jump、Pulse 命令等 PTP 運動速度
Accel	設置/顯示 PTP 運動的加減速度
Go	用於在當前位置和指定位置之間進行 PTP 運動
BGo	用於在已選擇的用戶坐標系上執行偏移 PTP 運動
TGo	用於在已選擇的工具坐標系上執行偏移 PTP 運動
SpeedS	用於設置/顯示 Move、Arc、Arc3、Jump3、Jump3CP 等 CP 運動速度
AccelS	設置/顯示直線插補運動的加減速度
Move	用於在當前位置和指定位置之間進行直線插補運動
BMove	用於在已選擇的用戶坐標系上執行偏移直線插補運動
TMove	用於在已選擇的工具坐標系上執行偏移直線插補運動
Jump	PTP 門型運動
Arc	用於在 XY 平面上進行圓弧插補運動
Arc3	用於在立體空間上進行圓弧插補運動
CP	用於設置/顯示運動平滑
CPZone	用於設置/顯示連續運動平滑參數
Here	用於示教/返回機器人當前座標
Home	返回 Home 點
Local	用於設置/顯示用戶坐標系
TLSet	用於設置/顯示工具坐標系
Tool	用於選擇/顯示指定編號的工具座標
Till	用於在運動過程中判斷條件成立並停止運動
Agl	用於返回指定關節的角度或位置
AglToPls	用於將機器人各關節角度轉換為脈衝
Arch	用於設置/顯示 Jump、Jump3、Jump3CP 命令的 Arch 參數
Find	用於設置/顯示在動作命令中保存座標的條件

FindPos	用於在執行動作命令過程中通過 Find 返回保存的座標
InPos	返回機器人是否到位
JA	以指定關節角度返回機器人座標
Joint	用於在關節坐標系顯示機器人當前座標
JS	用於返回 Sense 輸入是否成立
LimZ	用於設置/顯示 Jump 命令時第 3 關節高度(Z 座標值)初始值
Pass	用於進行穿過指定點附近且不停止的 PTP 運動
Pls	用於返回當前位置各關節的脈衝值
PosFound	用於返回 Find 命令時執行的狀態
Pulse	以 PTP 運動到各關節脈衝指定位置處，或根據脈衝值返回點位
RobLoad	用於選擇/顯示指定編號的負載
Sense	用於設置/顯示 Jump、Jump3、Jump3CP 停在目標座標上方的條件
TillOn	用於返回 Till 的狀態
WaitPos	等待機器人運動停止
XY	以指定數值返回點位
CX	用於設置/獲取點位的 X 座標
CY	用於設置/獲取點位的 Y 座標
CZ	用於設置/獲取點位的 Z 座標
CU	用於設置/獲取點位的 U 座標
CV	用於設置/獲取點位的 V 座標
CW	用於設置/獲取點位的 W 座標
CR	用於設置/獲取點位的 R 座標
CS	用於設置/獲取點位的 S 座標
CT	用於設置/獲取點位的 T 座標
ColSup	用於設置/獲取碰撞檢測係數
ZeroPls	用於設置/獲取機器人零位
Hand	用於設置/獲取當前手系
Elbow	用於設置/獲取肘姿態
Wrist	用於設置/獲取肘姿態
PDef	用於獲取點位是否存在定義
PDel	用於刪除點位
PLabel	用於設置點位數據的標籤

PLabel$	用於獲取點位標籤
PNumber	用於獲取點編號
PList	用於獲取點位數據資訊
PLocal	用於返回點位的用戶座標
PDescription	用於設置點位的描述
PDescription$	用於返回點的描述

I/O 指令

Wait	延時指令，或在一定時間內等待條件成立
Mask	以位為單位，用於對表示 Wait 輸入條件的值進行位 And 運算。
On	將 DO 置為 ON，或將 DO 置 ON 一定時間後置 OFF
Off	將 DO 置為 OFF，或將 DO 置 ON 一定時間後置 ON
Oport	用於返回指定 DO 的狀態
Sw	用於返回指定 DI 的狀態
SetSw	用於將指定 DI 置 ON 或置 OFF
In	根據 8421 返回 8 位 DI 的狀態，0~255
SetIn	根據 8421 設置 8 位 DI 的狀態，0~255
Out	根據 8421 設置/返回 8 位 DO 的狀態，0~255
InW	根據 8421 返回 16 位 DI 的狀態，0~65535
SetInW	根據 8421 設置 16 位 DI 的狀態，0~65535
OutW	根據 8421 設置/返回 16 位 DO 的狀態，0~65535
InBCD	根據 8421 返回 8 位 DI 的狀態，0~99
OpBCD	根據 8421 設置 8 位 DO 的狀態，0~99
InReal	以 32 位浮點數返回 DI 值
OutReal	以 32 位浮點數設置/返回 DI 值
AIO_InW	返回模擬量輸入的值
AIO_OutW	設置/返回模擬量輸出的值
MemOn	設置用戶寄存器 1 位為 1
MemOff	設置用戶寄存器 1 位為 0
MemSw	返回用戶寄存器 1 位的值
MemIn	返回用戶寄存器 8 位的值

MemOut	設置用戶寄存器 8 位的值
MemInW	返回用戶寄存器 16 位的值
MemOutW	設置用戶寄存器 16 位的值
MemIn32	返回用戶寄存器 32 位的值
MemOut32	設置用戶寄存器 32 位的值
MemInReal	以 32 位浮點數返回用戶寄存器 32 位的值
MemOutReal	以 32 位浮點數設置用戶寄存器 32 位的值
SysMemOn	設置系統寄存器 1 位為 1
SysMemOff	設置系統寄存器 1 位為 0
SysMemSw	返回系統寄存器 1 位的值
SysMemIn	返回系統寄存器 8 位的值
SysMemOut	設置系統寄存器 8 位的值
SysMemInW	返回系統寄存器 16 位的值
SysMemOutW	設置系統寄存器 16 位的值
SysMemIn32	返回系統寄存器 32 位的值
SysMemOut32	設置系統寄存器 32 位的值
SysMemInReal	以 32 位浮點數返回系統寄存器 32 位的值
SysMemOutReal	以 32 位浮點數設置系統寄存器 32 位的值

通訊指令

SetNet	用於設置 TCP/IP 端口參數
OpenNet	用於打開 TCP/IP 端口
WaitNet	用於等待 TCP/IP 端口建立連接
ChkNet	用於返回網路端口的接收緩衝器內的位元組數
ClrNet	用於清空網路端口接收緩衝器
CloseNet	用於關閉 OpenNet 打開的 TCP/IP 端口
SetCom	用於設置串口通訊參數
OpenCom	用於打開串口通訊參數
ChkCom	用於串口端口的接收緩存器內的位元組數

ClrCom	用於清空串口端口接收緩衝器
CloseCom	用於關閉 OpenCom 打開的串口端口
Read	用於從通訊端口讀取指定的位元組數
ReadBin	用於從通訊端口讀取二進位數據
Write	將字串寫入到通訊端口中
WriteBin	將二進位數據寫入到通訊端口中
Print #	將數據輸出到指定的通訊端口中
Input #	用於從通訊端口中讀取字串或數據
Line Input #	用於從通訊端口中讀取 1 行數據

系統指令

Print	列印字串或數據
Xqt	啟動多線程
Pause	暫停所有線程
Halt	暫停指定線程
Quit	停止所有或指定線程
Resume	繼續所有或指定線程
Reset	將控制器重置為初始狀態
MyTask	返回當前線程編號
TaskDone	用於確認線程是否結束
TaskState	用於返回指定線程的狀態
TaskWait	用於等待指定線程結束
SyncLock	用於相互排他鎖定，使多個線程同步
SyncUnLock	用於解鎖 SyncLock 鎖定的信號
Signal	用於向正在執行 WaitSig 命令的任務發送信號
WaitSig	用於等待其他任務的 Signal 命令發出的同步信號
TW	用於返回 Wait、WaitNet、WaitSig 命令的狀態
ElapsedTime	以秒為單位返回計算節拍時間計時器開始計時之後經過的時間
ResetElapsedTime	重置 ElapsedTime 計時器
Tmr	以秒為單位返回計時器開始計時之後經過的時間
TmReset	重置 Tmr 計時器
Eval	用於執行命令窗口的語句。

碼垛指令

Pallet	定義陣列、獲取陣列點位、列印陣列
PalletClr	清空陣列

傳送帶跟蹤指令

SyStart	開始傳送帶跟蹤功能
SyEnd	結束傳送帶跟蹤功能
SyReset	清空當前傳送帶跟蹤佇列數據
SyGetPoint	獲取可進行傳送帶跟蹤的點位數據
SyGetUserData	獲取當前跟蹤目標的附加資訊
SyGetPose	獲取當前跟蹤目標在傳送帶上的位置
SyGetNum	獲取傳送帶跟蹤佇列中的目標個數
SyMove	傳送帶跟蹤直線插補運動
SyArc	傳送帶跟蹤圓弧插補運動
SyTrig	記錄指定傳動帶編碼器脈衝數值
SyPoint	根據當前傳送帶追蹤生成追蹤點位
SyAddVisTarget	將視覺座標註冊到傳送帶跟蹤佇列中
SyRejectDis	設置/獲取傳送帶跟蹤功能濾重間距
SySave	保存程式中對傳送帶跟蹤功能的修改

數學運算指令

+	加
-	減
*	乘
/	除
**	乘方
Mod	整數取模
And	與
Or	或
Xor	異或
Not	非

>	大於
>=	大於等於
<	小於
<=	小於等於
<>	不等於
=	等於、賦值
DegToRad	角度轉弧度
RadToDeg	弧度轉角度
Sin	正弦函數
Cos	余弦函數
Tan	正切函數
Asin	反正弦函數
Acos	反余弦函數
Atan	反正切函數
Atan2	反正切函數 2
HexToFloat	十六進制浮點數轉單精確度浮點數
FloatToHex	單精確度浮點數轉十六進制浮點數
Abs	絕對值
Sqr	平方根
Sgn	返回數值的符號
LShift	左移
RShift	右移
BClr	將數值指定的 Bit 置 0 並返回該數值
BSet	將數值指定的 Bit 置 1 並返回該數值
BTst	返回數值指定 Bit 的值
Ubound	獲取數組長度

字串運算指令

ParseStr	字串分割
+	字串拼接
<>	不等於
=	等於、賦值
Val	字串轉數值
Str$	數值轉字串
Len	獲取字串長度
Asc	字元轉 ASCII 碼
Chr$	ASCII 碼轉字元
Left$	從字串左側開始提取指定的字元
Mid$	在字串中提取指定範圍的字元或字串
Right$	從字元有右側開始提取指定的字元
LSet$	從字串左側開始獲取指定長度的字串
RSet$	從字串右側開始獲取指定長度的字串
Space$	返回指定長度的空格字串
LCase$	返回小寫字母的字串
UCase$	返回大寫字母的字串
LTrim$	刪除字串左側的空格並返回
RTrim$	刪除字串右側的空格並返回
Trim$	刪除字串兩側的空格並返回
InStr	從字串中檢索指定字元的位置並返回
Tab$	返回指定數量的跳位字元字串
Hex$	將十六進制數值轉換為字串並返回

點位運算指令

+	相對座標增量運動
-	相對座標負增量運動
/	修改手系、修改用戶坐標系
:	絕對座標運動
=	賦值
@	轉換到指定用戶坐標系
X	X 方向分量運算
Y	Y 方向分量運算
Z	Z 方向分量運算
U	U 方向分量運算
V	V 方向分量運算
W	W 方向分量運算
RX	用戶坐標系繞 X 軸旋轉分量運算
RY	用戶坐標系繞 Y 軸旋轉分量運算
RZ	用戶坐標系繞 Z 軸旋轉分量運算
TLX	工具坐標系繞 X 軸旋轉分量運算
TLY	工具坐標系繞 Y 軸旋轉分量運算
TLZ	工具坐標系繞 Z 軸旋轉分量運算
TLU	工具坐標系繞 U 軸旋轉分量運算
TLV	工具坐標系繞 V 軸旋轉分量運算
TLW	工具坐標系繞 W 軸旋轉分量運算
SavePoints	保存點位檔
LoadPoints	加載點位檔
ClearPoints	清空點位

文本符號聲明：

1. {參數} 大括弧指大括弧中的參數是可省略不寫的，並不是強制的語法要求。
2. [參數] 中括弧指中括弧中的參數必須任選其一，必須要填入，屬於語法強制要求。

通用語句詳細說明

[功能]

調用函數

[語法]

Call 函數名

[參數說明]

函數名 需要調用的 Function 名

[使用案例]

```
Function main
    Call Task2
    Task2
    Task2()
Fend
```

[功能]

根據條件是否成立，在 Do..Loop 之間反復執行代碼

[語法]

```
Do {While / Until 條件運算式}
    ...
Loop
```

或

```
Do
    ...
Loop {While/Until 條件運算式}
```

[參數說明]

While/Until	可在 Do 或 Loop 後面追加關鍵字來使用條件運算式
條件運算式	根據條件運算式是否成立來跳出 Do..Loop 迴圈

[使用案例]

```
Do                      '一直迴圈
    Go P1
    Wait 1
    Go P2
    Wait 2
Loop

    Do Until Sw(1) <> 1 '當 DI1 不等於 1 時進入迴圈
    Go P1
    Wait 1
    Go P2
    Wait 2
Loop

Do                      '一直迴圈
    Go P1
    Wait 1
    Go P2
    Wait 2
Loop Until ChkNet(201) < 0 '當 TCP/IP 201 端口通訊異常時跳出迴圈
```

[功能]

搭配判斷語句使用，當 If 條件不成立時，執行 Else/Elself 內的代碼。

[語法]

```
If 條件運算式 Then
    ...
Elself 條件運算式 Then
    ...
Else
    ...
Endlf
```

[參數說明]

條件運算式　　根據條件運算式是否成立來執行 Else/Elself 內代碼

[使用案例]

```
If Sw(1) = 1 Then
    Print "DI1 為 On"

Elself Sw(1) = 1  Then
    Print "DI1 為 Off"

Else
    Print "DI1 異常"
Endlf
```

[功能]

用於強制結束迴圈或函數

[語法]

```
Exit [Do/For/Function]
```

[參數說明]

Exit Do 退出 Do..Loop 迴圈。如存在多層 Do..Loop 嵌套，Exit Do 只會退出一級迴圈。

Exit For 退出 For 迴圈。如存在多層 For 嵌套，Exit For 只會退出一級迴圈。

Exit Function 退出 Function。如存在多層 Function 嵌套，Exit Function 只會退出至上一級 Function

[使用案例]

```
Do                      '一直迴圈
    Go P1
    Go P2
    If Sw(1) = 1 Then '當 DI1 等於 1 時進入判斷
        Exit Do         '退出 Do 迴圈
    EndIf
Loop

Integer var
For var =  1 To 1     '迴圈 5 次
    Go P1
    Go P2
    If Sw(1) =  1 Then '當 DI1 等於 1 時進入判斷
        Exit For        '退出 For 迴圈
    EndIf
Next

Function main
    Exit Function      '退出函數
Fend
```

[功能]

反復執行一定次數 For..Next 之間的代碼

[語法]

```
For 變數 = 起始值 To 結束值 {Step 增量值}

    ...
Next
```

[參數說明]

變數	用於在 For 迴圈中迭代的變數，該變數需要提前定義。
起始值	變數在 For 迴圈開始時的起始值。
結束值	變數在 For 迴圈結束時的結束值。
增量值	每完成一次迴圈後的疊加的數值，默認為疊加 1 步。

[使用案例]

```
Integer var
For var = 1 To 5 Step 1  '迴圈 5 次
    Go P1
    Go P2
    Print var
Next
```

[功能]

全局函數定義

[語法]

```
Function 函數名{(引數 As 數據類型,引數  數據類型, ...)}{As  數據類型}
    ...
Fend
```

[參數說明]

函數名　　必須定義的 Function 名

引數列表　　需要在 Function 中傳遞數值或數值運算的形式參數，多個變數時用逗號進行分隔。

ByRef　調用形參為數組時需配合該指令使用，可修改引數索引值。

ByVal　調用形參為數組時需配合該指令使用，不可修改引數索引值。

引數 As 數據類型　必要的參數。請聲明引數的類型。

函　數 As 數據類型　將函數名生成為變數並存儲返回值。請聲明函數名所屬的變數類型。

[使用案例]

```
Function aa          '普通的 Function
    Print "123"
    Print "456"
    Print "789"
Fend

aa              '調用

Function Trig_CCD(IO_Idx As Integer, Integer_Time As Double) '帶引數的 Function
    Do
        On IO_Idx
        Wait Integer_Time
        Off IO_Idx
    Loop
Fend

Function Add(Num1 As Integer, Num2 As Integer) As Integer '帶引數且帶返回值的 Function
    Integer result
    result = Num1 + Num2
    Add = result
Fend
```

[功能]

全局變數定義

[語法]

Global 數據類型 變數名

[參數說明]

變數名 必須定義的變數名稱

注意事項:

1. 全局變數只能在函數外定義，不能在函數內定義。
2. 全局變數總數不得超過 100000 個。String 類型不得超過 10000 個。

[使用案例]

```
Global String    PointList$(300,24)    '字串二維數組
Global Boolean   P_Mode                '布爾類型
Global String    CCD_Dev_X$            '字串類型
Global Double    SafeHight             '雙精度浮點數類型
Global Integer   ABCD                  '整形
Global Integer   AABB(10)              '整形一維數組
```

[功能]

GoSub 到標籤處，將程式控制權移交給副程式，副程式通過 Return 將程式返回至原來 GoSub 的位置。

[語法]

```
GoSub [標籤]
...
[標籤]:
...
Return
```

[參數說明]

標籤	必須定義的標籤名稱

[使用案例]

```
Function main
    GoSub checkin            '跳轉至標籤處
        On 1
        On 2
    Exit Function

Checkin:                     '標籤
    If In(0) = 1 And In(1) = 1 Then
        On  1
    Else
        Off 1
    EndIf
    Return                   '返回 GoSub 處
Fend
```

[功能]

跳轉到標籤處繼續執行程式

[語法]

GoTo [標籤]

...

[標籤]:

...

[參數說明]

標籤 必須定義的標籤名稱

注意事項:

1. 與 GoSub 的區別在於，GoTo 沒有 Return 操作。

[使用案例]

```
Function main
    If Sw(1) = Off Then
        GoTo mainAbort
    EndIf
    Print    "DI1 為 On"

    Exit Function

    mainAbort:
    Print    "DI1 為 Off"
Fend
```

[功能]

判斷語句，根據條件運算式執行對應的程式

[語法]

```
If 條件運算式 Then
    ...
ElseIf 條件運算式 Then
    ...
Else
    ...
EndIf
```

[參數說明]

條件運算式　　根據條件運算式是否成立來執行對應的程式

[使用案例]

```
Function main
    String AA$
    Integer num

    Input #201, AA$

    Num = val(AA$)          '接收內容轉換成數值並賦值給 Num

    If num > 1 Then
        Print    "num 大於 1"
    ElseIf num < 1 Then
        Print    "num 小於 1"
    Else
        Print    "num 等於 1"

    EndIf
Fend
```

[功能]

選擇語句，根據值來決定執行那個段落的程式

[語法]

```
Select 變數
Case 值 1
    ...
{Case 值 2
    ...}
Default
    ...
Send
```

[參數說明]

變數	用來作為條件判斷的變數名
值 1、值 2	判斷變數是否等於該值，進入該程式段
Default	當所有 Case 不成立時，執行 Default 的程式段

[使用案例]

```
Function main
    Integer I
    For I = 0 To 10
        Select I
            Case 0
                Print    "I = 1"
                Off 1; On 2; Jump P1
            Case 3
                Print    "I = 3"
                On 1; Off 2
                Jump P2;Move P3; On 3
            Case 7
                Print    "I = 3"
                On 4
            Default
                On 7
        Send
    Next
Fend
```

數據類型詳細說明

[功能]

布爾類型，2 位元組

[語法]

```
Boolean 變數{(數組元素個數)}
Global Boolean 變數{(數組元素個數)}
```

[參數說明]

變數	必須定義的變數名
取值範圍	True/False

注意事項：

1. 局部變數最大數量：2000
2. 全局變數最大數量：100000

[使用案例]

```
Function main
    Boolean AA
    If AA = True Then
        Print    "AA 為真"
    Elself AA = False Then
        Print    "AA 為假"
    Endlf
Fend
```

[功能]

位元組類型，2 位元組

[語法]

Byte 變數[(數組元素個數)]

Global Byte 變數[(數組元素個數)]

[參數說明]

變數	必須定義的變數名
取值範圍	-128~127

注意事項：

3. 局部變數最大數量：2000
4. 全局變數最大數量：100000

[使用案例]

```
Function main
    Byte A1(10)          'Byte 類型的一維數組
    Byte B1(10,10)       'Byte 類型的二維數組
    Byte CC(5,5,5)       'Byte 類型的三維數組
    Byte Test_ok

    Test_ok = 15

    Test_ok =  (test_ok And 8)

    If   test_ok <> 8 Then
        Print   "高位 bit 為 ON"
    Else
        Print   "高位 bit 為 OFF"
    EndIf
Fend
```

[功能]

雙精度浮點數，8 位元組

[語法]

Double 變數{(數組元素個數)}

Global Double 變數{(數組元素個數)}

[參數說明]

變數	必須定義的變數名
取值範圍	可取值到小數點後 14 位

注意事項：

5. 局部變數最大數量：2000
6. 全局變數最大數量：100000

[使用案例]

```
Function main
    Double var1
    Double A1(10)            'Double 類型的一維數組
    Double B1(5,5)           'Double 類型的二維數組
    Double CC(5,5,5)         'Double 類型的三維數組
    Double arrayvar(10)
    Integer i

    For  i = 1  To 5
        Print arrayvar(i)
    Next
Fend
```

[功能]

32 位整型，4 位元組

[語法]

Int32 變數{(數組元素個數)}

Global Int32 變數{(數組元素個數)}

[參數說明]

變數	必須定義的變數名
取值範圍	-2147483648~2147483647

注意事項：

7. 局部變數最大數量：2000
8. 全局變數最大數量：100000

[使用案例]

```
Function main
    Int32 A1(10)          'Int32 類型的一維數組
    Int32 B1(5,5)         'Int32 類型的二維數組
    Int32 CC(5,5,5)       'Int32 類型的三維數組
    Int32 var1,arrayvar(10)
Fend
```

[功能]

整型類型，2 位元組

[語法]

Integer 變數{(數組元素個數)}

Global Integer 變數{(數組元素個數)}

[參數說明]

變數	必須定義的變數名
取值範圍	-32768~32767

注意事項：

9. 局部變數最大數量：2000

10. 全局變數最大數量：100000

[使用案例]

```
Function main
    Integer A1(10)          'Interger 類型的一維數組
    Integer B1(5,5)         'Interger 類型的二維數組
    Integer CC(5,5,5)       'Interger 類型的三維數組
    Integer var1,arrayvar(10)
Fend
```

[功能]

長整型類型，4 位元組

[語法]

Long 變數{(數組元素個數)}

Global Long 變數{(數組元素個數)}

[參數說明]

變數	必須定義的變數名
取值範圍	-2147483648~2147483647

注意事項：

11. 局部變數最大數量：2000
12. 全局變數最大數量：100000

[使用案例]

```
Function main
    Long A1(10)          'Long 類型的一維數組
    Long B1(5,5)         'Long 類型的二維數組
    Long CC(5,5,5)       'Long 類型的三維數組
    Long var1,arrayvar(10)
Fend
```

[功能]

單精確度浮點數，4 位元組

[語法]

Real 變數[(數組元素個數)]

Global Real 變數[(數組元素個數)]

[參數說明]

變數	必須定義的變數名
取值範圍	小數點後六位

注意事項：

13. 局部變數最大數量：2000
14. 全局變數最大數量：100000

[使用案例]

```
Function main
    Real A1(10)          'Real 類型的一維數組
    Real B1(5,5)         'Real 類型的二維數組
    Real CC(5,5,5)       'Real 類型的三維數組
    Real var1,arrayvar(10)
Fend
```

[功能]

短整型，2 位元組

[語法]

Short 變數[(數組元素個數)]

Global Short 變數[(數組元素個數)]

[參數說明]

變數	必須定義的變數名
取值範圍	-32768~32767

注意事項：

15. 局部變數最大數量：2000
16. 全局變數最大數量：100000

[使用案例]

```
Function main
    Short A1(10)              'Short 類型的一維數組
    Short B1(5,5)             'Short 類型的二維數組
    Short CC(5,5,5)           'Short 類型的三維數組
    Short var1,arrayvar(10)
Fend
```

[功能]

字串類型

[語法]

String 變數${(數組元素個數)}

Global String 變數${(數組元素個數)}

[參數說明]

變數$	必須定義的變數名
取值範圍	最大 255 個字元

注意事項：

17. 局部變數最大數量：200
18. 全局變數最大數量：10000

[使用案例]

```
Function main
    String A1$(10)              'String 類型的一維數組
    String B1$(5,5)             'String 類型的二維數組
    String CC$(1,2,3)           'String 類型的三維數組
    String var1$,arrayvar$(10)
Fend
```

[功能]

無符號位元組類型，2 位元組。

[語法]

UByte 變數{(數組元素個數)}

Global UByte 變數{(數組元素個數)}

[參數說明]

變數	必須定義的變數名
取值範圍	0~255

注意事項：

19. 局部變數最大數量：2000
20. 全局變數最大數量：100000

[使用案例]

```
Function main
    UByte A1(10)                'UByte 類型的一維數組
    UByte B1(5,5)               'UByte 類型的二維數組
    UByte CC(1,2,3)             'UByte 類型的三維數組
    UByte var1,arrayvar(10)
Fend
```

[功能]

無符號 32 位整型，4 位元組

[語法]

UInt32 變數[(數組元素個數)]

Global UInt32 變數[(數組元素個數)]

[參數說明]

變數	必須定義的變數名
取值範圍	0~4294967295

注意事項：

21. 局部變數最大數量：2000
22. 全局變數最大數量：100000

[使用案例]

```
Function main
    UInt32 A1(10)                    'UInt32 類型的一維數組
    UInt32 B1(5,5)                   'UInt32 類型的二維數組
    UInt32 CC(1,2,3)                 'UInt32 類型的三維數組
    UInt32 var1,arrayvar(10)
Fend
```

[功能]

無符號短整型，2 位元組

[語法]

UShort 變數[(數組元素個數)]

Global UShort 變數[(數組元素個數)]

[參數說明]

變數	必須定義的變數名
取值範圍	0~65535

注意事項：

23. 局部變數最大數量：2000
24. 全局變數最大數量：100000

[使用案例]

```
Function main
    UShort A1(10)                    'UShort 類型的一維數組
    UShort B1(5,5)                   'UShort 類型的二維數組
    UShort CC(1,2,3)                 'UShort 類型的三維數組
    UShort var1,arrayvar(10)
Fend
```

[功能]

將全局變數定義為掉電保持狀態，停止程式或重啟控制器保持變數記憶。

[語法]

```
Global Preserve Integer 變數{(數組元素個數)}
```

[參數說明]

變數	必須定義的變數名
取值範圍	-2147483648~2147483647

注意事項：

25. 局部變數最大數量：2000
26. 全局變數最大數量：100000

[使用案例]

```
Global Preserve Integer Hi
Global Preserve Integer PowerNex
```

運動指令詳細說明

[功能]

並行處理運動過程中 I/O 等的輸入輸出

[語法]

運動指令 !處理語句!

[參數說明]

運動指令 Go、Move、Arc、Jump 等相關的運動指令

處理語句 如下

1. D：百分比處理語句

2. On/Off、MemOff、Out、OutW、MemOut、Signal、Wait、Print、Print#等 I/O 指令。詳情請查閱 I/O 指令詳細說明。

[使用案例]

```
Function main
    Go P1 !D50; On 1!
    'PTP 運動至 P1 點過程中，運動至 50%路徑時 DO1 置 ON

    Jump P2 !D30; On 1; D70; Off 1!
    '門型運動至 P2 點過程中，運動至 30%路程時 DO1 置 ON, 70%時 DO1 置 OFF

    Move P3 !D10; On 5; Wait 0.5; Off 5!
    '直線運動至 P3 過程中，運動至 10%路程時 DO5 置 ON, 並在 0.5 秒後 DO5 置 OFF
Fend
```

[功能]

機器人整體上/下使能、或返回機器人使能狀態

[語法]

```
Motor {On/Off}
```

[參數說明]

On/Off On 為整體上使能，Off 為整體下使能。省略不填時則返回機器人使能狀態。

[使用案例]

```
Function main
    If Motor =  Off Then
        Motor On
    EndIf
Fend
```

[回傳說明]

回傳數據 1 0 / 1 ， 0 為 Off,1 為 On。

[功能]

單軸上/下使能、或返回單軸上/下使能狀態

[語法]

上下使能：

ServoOn 關節編號 1 {,關節編號 2...}}

ServoOff 關節編號 1 {,關節編號 2...}}

獲取使能狀態：

ServoOn(關節編號)

ServoOff(關節編號)

[參數說明]

關節編號 以 1、2、3、4、5、6 來指定需要上下使能的關節。或需要返回使能狀態的關節。

[使用案例]

```
Function main
    If ServoOff =  True Then     '判斷 1 關節使能狀態
        ServoOn 1                '1 關節上使能
    EndIf
Fend
```

[回傳說明]

回傳數據 1 TRUE / FALSE ， TRUE 為真，FALSE 為假。

[功能]

切換高低功率模式、或返回當前功率模式

[語法]

```
Power {High/Low}
```

[參數說明]

High/Low	High 為高功率、Low 為低功率。省略不填則返回當前功率狀態。

[使用案例]

```
Function main
    If Power = Low Then     '判斷當前是否處於低功率
        Power High          '切換高功率
    EndIf
Fend
```

[回傳說明]

回傳數據 1	0 / 1，0 為低功率模式，1 為高功率模式

[功能]

設置全局運動速度，或返回當前全局速度

[語法]

SpeedFactor {百分比}

[參數說明]

百分比 以 1~100%來設定當前的全局速度。省略不填則返回當前的全局速度。

[使用案例]

```
Function main
    Motor On
    Power High

    Speed 100         'PTP 速度
    Accel 100,100
    SpeedS 2000       'Line 速度
    AccelS 100,100

    SpeedFactor 80   '全局速度

    Print SpeedFactor    '列印當前全局速度
Fend
```

[回傳說明]

回傳數據 1 百分比數值 1 -100 % ，數據類型默認為浮點數。

[功能]

設置 PTP 運動的運行速度，或返回當前 PTP 運動的運行速度

[語法]

Speed {百分比{, 轉速速度百分比, 接近速度百分比}}

[參數說明]

百分比 以 1~100%來設定當前的 PTP 運動速度。

轉速速度百分比 以 1~100%來設定 Jump 命令時的轉移動作速度，可省略。

接近速度百分比 以 1~100%來設定 Jump 命令時的轉移動作速度，可省略。

注意事項：若程式中未使用該指令定義 PTP 運動速度時，將使用默認速度：20%

[使用案例]

```
Function main
    Motor On
    Power High

    Speed 100         'PTP 速度
    Accel 100,100
    SpeedS 2000       'Line 速度
    AccelS 100,100

    SpeedFactor 80    '全局速度

    Print SpeedFactor     '列印當前全局速度
Fend
```

[回傳說明]

回傳數據 1 百分比數值 1 -100 % ，數據類型默認為浮點數。

[功能]

設置 PTP 運動的運行加減速度，或返回當前 PTP 運動的運行加減速度

[語法]

Accel {加速度百分比,減速度百分比,{轉速加速度百分比, 轉速減速度百分比, 接近加速度百分比, 接近減速度百分比}}

[參數說明]

加速度百分比 以 1~100%來設定當前的 PTP 運動加速度。

減速度百分比 以 1~100%來設定當前的 PTP 運動減速度。

轉速加速度百分比 以 1~100%來設定 Jump 命令時的轉移動作加速度，可省略。

轉速減速度百分比 以 1~100%來設定 Jump 命令時的轉移動作減速度，可省略。

接近加速度百分比 以 1~100%來設定 Jump 命令時的接近動作加速度，可省略。

接近減速度百分比 以 1~100%來設定 Jump 命令時的接近動作減速度，可省略。

[使用案例]

```
Function main
    Accel 100,100,50,50,50,50    '設置 PTP 運動加減速度

    Print Accel(1)               '設置 PTP 運動加速度
    Print Accel(2)               '設置 PTP 運動減速度
    Print Accel(3)               '設置 JUMP 運動轉移動作加速度
    Print Accel(4)               '設置 JUMP 運動轉移動作減速度
    Print Accel(5)               '設置 JUMP 運動接近動作加速度
    Print Accel(6)               '設置 JUMP 運動接近動作減速度
Fend
```

[回傳說明]

回傳數據 百分比數值 1 -100 % ,數據類型默認為浮點數。

[功能]

用於在當前位置到指定位置之間以 PTP 運動。

[語法]

Go 目標座標 {CP} {Till/Find} {!...!}

[參數說明]

目標座標	以點數據指定目標位置
CP	設置運動平滑。可省略
Till/Find	Till 運算式 = On/Off。Find 運算式 = On/Off。可省略。詳情請參考 Till/Find 的詳細說明
!...!	在運動過程中並行處理 I/O 等輸入輸出。可省略。詳情請參考並行處理的詳細說明。

[使用案例]

```
Function main
    Go P1
    Go P2 CP

    Go P1+X(20)
    '運動 P1 點且 X 方向相對偏移 20mm

    Go P2:X(200)
    '運動 P2 點且 X 座標改變為 200mm 處

    Go P3 Till Sw(1) = On And Sw(5) = Off
    '運動 P3 點過程中，當 DI1 為 On，DI5 為 Off 時停止運動

    Go P4 Till Sw(1) = On !D50; On 5!
    '運動 P4 點過程中，當 DI1 為 On 時停止運動，且運動到 50%路徑時將 DO5 置 On

    Go P5 /R          '以右手系運動至 P5 點
Fend
```

[功能]

用於在已選擇的用戶坐標系上執行偏移 PTP 運動

[語法]

BGo 目標座標 {CP}{Till/Find}{!...!}

[參數說明]

目標座標	以點數據指定目標位置
CP	設置運動平滑。可省略
Till/Find	Till 運算式 = On/Off。Find 運算式 = On/Off。可省略。詳情請參考 Till/Find 的詳細說明
!...!	在運動過程中並行處理 I/O 等輸入輸出。可省略。詳情請參考並行處理的詳細說明。

[使用案例]

```
Function main
    BGo XY(100, 0, 0, 0)              '在已選擇用戶座標上，向 X 方向移動 100mm

    P1 = XY(300, 300, -20, 0)
    P2 = XY(300, 300, -20, 0) /L
    Local 1, XY(0, 0, 0, 45)

    Go P1
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /R /0
    BGo XY(0, 50, 0, 0)
    Print Here
    'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /R /0
    Go P2
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0
    BGo XY(0, 50, 0, 0)
    Print Here
    'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0
    BGo XY(0, 50, 0, 0) /L
    Print Here
    'X:264.645 Y:385.355 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0
Fend
```

[功能]

用於在當前工具坐標系上執行偏移 PTP 動作

[語法]

TGo 目標座標 {CP}{Till/Find}{!...!}

[參數說明]

目標座標	以點數據指定目標位置
CP	設置運動平滑。可省略
Till/Find	Till 運算式 = On/Off。Find 運算式 = On/Off。可省略。詳情請參考 Till/Find 的詳細說明
!...!	在運動過程中並行處理 I/O 等輸入輸出。可省略。詳情請參考並行處理的詳細說明。

[使用案例]

```
Function main
    TGo XY(100, 0, 0, 0)              '在已選擇用戶座標上，向 X 方向移動 100mm

    Tool 0
    P1 = XY(300, 300, -20, 0)
    P2 = XY(300, 300, -20, 0) /L

    Go P1
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /R /0

    TGo XY(0, 0, -30, 0)
    Print Here
    'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000 /R /0

    Go P2
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000 /L /0

    TGo XY(0, 0, 30, 0)
    Print Here
    'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000 /L /0
Fend
```

[功能]

設置直線插補運動的運行速度，或返回當前直線插補運動的運行速度

[語法]

SpeedS {速度[, 轉速速度, 接近速度]}

[參數說明]

速度 以 0.1~2000mm/s 來設定當前的直線插補運動速度。

轉速速度 以 0.1~2000mm/s 來設定 Jump3 命令時的轉移動作速度，可省略。

接近速度 以 0.1~2000mm/s 來設定 Jump3 命令時的轉移動作速度，可省略。

[使用案例]

```
Function main
    Motor On
    Power High

    Speed 100         'PTP 速度
    Accel 100,100
    SpeedS 2000       'Line 速度
    AccelS 100,100

    SpeedFactor 80   '全局速度

    Print SpeedFactor    '列印當前全局速度
Fend
```

[回傳說明]

回傳數據 1 百分比數值 1 -100 % ，數據類型默認為浮點數。

[功能]

設置直線插補運動的運行加減速度，或返回當前直線插補運動的運行加減速度

[語法]

AccelS {加速度百分比,減速度百分比,{轉速加速度百分比, 轉速減速度百分比, 接近加速度百分比, 接近減速度百分比}}

[參數說明]

加速度百分比 以 1~100%來設定當前的直線插補運動加速度。

減速度百分比 以 1~100%來設定當前的直線插補運動減速度。

轉速加速度百分比 以 1~100%來設定 Jump3 和 Jum3CP 命令時的轉移動作加速度，可省略。

轉速減速度百分比 以 1~100%來設定 Jump3 和 Jum3CP 命令時的轉移動作減速度，可省略。

接近加速度百分比 以 1~100%來設定 Jump3 和 Jum3CP 命令時的接近動作加速度，可省略。

接近減速度百分比 以 1~100%來設定 Jump3 和 Jum3CP 命令時的接近動作減速度，可省略。

[使用案例]

```
Function main
    Motor On
    Power High

    Speed 100        'PTP 速度
    Accel 100,100
    SpeedS 2000      'Line 速度
    AccelS 100,100

    SpeedFactor 80   '全局速度

    Print SpeedFactor    '列印當前全局速度
Fend
```

[功能]

用於在當前位置到指定位置之間以直線插補運動。

[語法]

Move 目標座標 {CP} {Till/Find} {!...!}

[參數說明]

目標座標	以點數據指定目標位置
CP	設置運動平滑。可省略
Till/Find	Till 運算式 = On/Off。Find 運算式 = On/Off。可省略。詳情請參考 Till/Find 的詳細說明
!...!	在運動過程中並行處理 I/O 等輸入輸出。可省略。詳情請參考並行處理的詳細說明。

[使用案例]

```
Function main
    Move P1          '直線運動到 P1 點
    Move P2 CP       '直線運動到 P2 點，且在 P2 點處平滑過渡

    Move P3 Till Sw(1) = On And Sw(5) = Off
    '運動 P3 點過程中，當 DI1 為 On，Di5 為 Off 時停止運動

    Move P4 Till Sw(1) = On !D50; On 5!
    '運動 P4 點過程中，當 DI1 為 On 時停止運動，且運動到 50%路徑時將 DO5 置為 On
Fend
```

[功能]

用於在已選擇的用戶坐標系上執行偏移直線插補運動

[語法]

BMove 目標座標 {CP} {Till/Find} {!...!}

[參數說明]

目標座標	以點數據指定目標位置
CP	設置運動平滑。可省略
Till/Find	Till 運算式 = On/Off。Find 運算式 = On/Off。可省略。詳情請參考 Till/Find 的詳細說明
!...!	在運動過程中並行處理 I/O 等輸入輸出。可省略。詳情請參考並行處理的詳細說明。

[使用案例]

```
Function main
BMove XY(100 , 0 , 0 , 0)              '在已選擇用戶座標上，向 X 方向移動 100mm
    P1 = XY(300 , 300 , -20 , 0)
    P2 = XY(300 , 300 , -20 , 0) /L
    Local 1 , XY(0 , 0 , 0 , 45)

    Go P1
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000

    BMove XY(0 , 50 , 0 , 0)
    Print Here
    'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000

    Go P2
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000

    BMove XY(0 , 50 , 0 , 0)
    Print Here
    'X:300.000 Y:350.000 Z:-20.000 U:0.000 V:0.000 W:0.000

Fend
```

[功能]

用於在當前工具坐標系上執行偏移直線動作

[語法]

TMove 目標座標 {CP} {Till/Find} {!...!}

[參數說明]

目標座標	以點數據指定目標位置
CP	設置運動平滑。可省略
Till/Find	Till 運算式 = On/Off。Find 運算式 = On/Off。可省略。詳情請參考 Till/Find 的詳細說明
!...!	在運動過程中並行處理 I/O 等輸入輸出。可省略。詳情請參考並行處理的詳細說明。

[使用案例]

```
Function main
    TMove XY(100 , 0 , 0 , 0) '在已選擇工具座標上，向 X 方向移動 100mm

    Tool 0          '選擇工具 0
    P1 = XY(300 , 300 , -20 , 0)
    P2 = XY(300 , 300 , -20 , 0) /L

    Go P1
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000

    TMove XY(0 , 0 , -30 , 0)
    Print Here
    'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000

    Go P2
    Print Here
    'X:300.000 Y:300.000 Z:-20.000 U:0.000 V:0.000 W:0.000

    TMove XY(0 , 0 , -30 , 0)
    Print Here
    'X:300.000 Y:350.000 Z:-50.000 U:0.000 V:0.000 W:0.000
Fend
```

[功能]

用於在當前位置到指定位置之間以門型 PTP 運動。先上升，再平移，再下降的門型運動。

[語法]

Jump 目標座標 {C Arch 編號} {LimZ 座標值} {CP} {Till/Find/Sense} {!...!}

[參數說明]

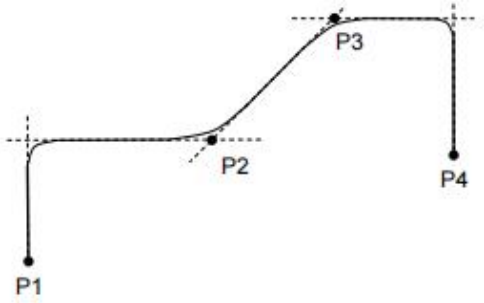
目標座標	以點數據指定目標位置
C Arch 編號	以 C0~C7 來選定不同的 Arch。詳情請參考 Arch 詳細說明。
LimZ 座標值	設定 Jump 運動過程中，第 3 關節可移動的最大值(限制值)。詳情請參考 LimZ 詳細說明。可省略
CP	設置運動平滑。可省略
Till/Find/Sense	Till 運算式 = On/Off。Find 運算式 = On/Off。Sense 運算式 = On/Off。可省略。詳情請參考 Till/Find/Sense 的詳細說明
!...!	在運動過程中並行處理 I/O 等輸入輸出。可省略。詳情請參考並行處理的詳細說明。

[使用案例]

```
Function main
    Go P1
    Jump P2:Z(-50) C0 LimZ -50 CP
    Go P3:Z(0) CP
    Jump P4 C0 LimZ 0

    Jump P1
    Jump P2 CP
    Jump P3 Till Sw(1) = On And Sw(5) = Off
    '運動 P3 點過程中，當 DI1 為 ON，DI5 為 OFF 時停止運動

    Jump P4 Till Sw(1) = On !D50; On 5!
    '運動 P4 點過程中，當 DI1 為 On 時停止運動，且運動到 50%路徑時將 DO5 置 On
    Jump P5 /R C1 LimZ-10 Sense Sw(2) = On
    '選擇 Arch1 為接近參數，以右手系運動 P5 點過程中，Z 軸最大只能-10MM
    '當 DI2 為 On 時停在目標點上方，且運動到 50%路徑時將 DO5 置 On
Fend
```



[功能]

Arc 用於在 XY 平面上以圓弧插補動作將機械臂從當前位置移至指定位置

Arc3 用於在三維空間上以圓弧插補動作將機械臂從當前位置移至指定位置

[語法]

Arc 經過座標, 目標座標 {CP} {Till/Find} {!...!}

Arc3 經過座標, 目標座標 {CP} {Till/Find} {!...!}

[參數說明]

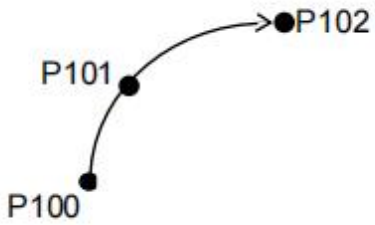
經過座標	以點數據指定指定圓弧會經過的位置
目標座標	以點數據指定目標位置
CP	設置運動平滑。可省略
Till/Find	Till 運算式 = On/Off。Find 運算式 = On/Off。詳情請參考 Till/Find/Sense 的詳細說明
!...!	在運動過程中並行處理 I/O 等輸入輸出。可省略。詳情請參考並行處理的詳細說明。

[使用案例]

```
Function main

    Go P100
    Arc P101 , P102

Fend
```



[功能]

設置運動平滑

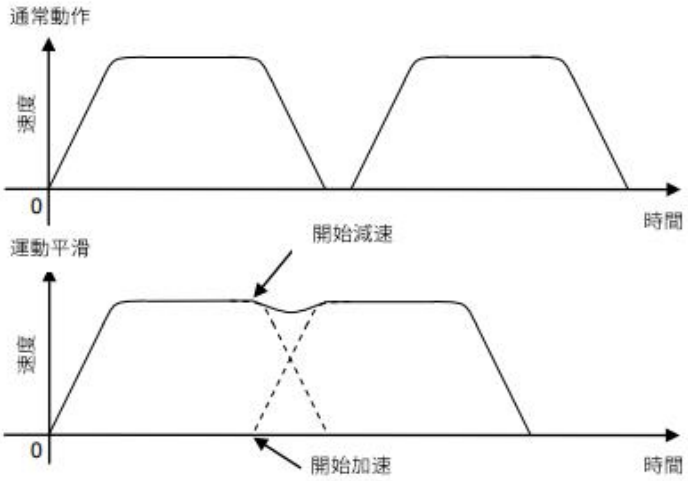
[語法]

CP {On/Off}
運動指令 目標座標 CP

[參數說明]

On/Off	On 為打開運動平滑，Off 為關閉。
運動指令	Go、Move、Arc、Jump 等運動指令

[使用案例]




```
Function main
    Go P1 CP
    Move P2 CP

    CP On           'CP On 以後的運動指令默認平滑過渡
    Move P3
    Move P4
    CP Off

    If CP = Off Then
        CP On
    EndIf
Fend
```

[功能]

設置運動平滑的比例程度

[語法]

CPZone {On/Off} {PTP 運動平滑比例, 直線插補平滑比例}

[參數說明]

On/Off	On 為開啟運動平滑比例，Off 為關閉平滑比例。Off 後則採用系統默認比例。
PTP 運動平滑比例	以 1~200 mm 來設定 PTP 運動平滑的融合程度。
直線插補平滑比例	以 1~200 mm 來設定直線插補運動平滑的融合程度。

[使用案例]

```
Function main

    CPZone On , 50 , 50

Fend
```

[功能]

用於將機器人當作座標示教至點位中，或獲取機器人當前座標。

[語法]

```
Here {點編號/點名稱}
```

[參數說明]

點編號/點名稱	將當前座標示教至該點位

[使用案例]

```
Function main

    Here P1          '將當前座標示教到 P1 點
    Go Here + X(10)  '在當前位置向 X 方向偏移 10mm

Fend
```

[功能]

將機器人移動至用戶定義的 Home 點處

[語法]

```
Home
```

[參數說明]

[使用案例]

```
Function main

    Home        '回到 Home 點

Fend
```

[功能]

用於定義和顯示用戶坐標系

[語法]

一點示教：Local 用戶座標編號, 點位
兩點示教：Local 用戶座標編號, 點位 1, 點位 2, {X/Y}
三點示教：Local 用戶座標編號, 原點, X 軸點, Y 軸點, {X/Y}
四點示教：Local 用戶座標編號, (點位 1:點位 2), (點位 3:點位號 4), {L/R}

[參數說明]

用戶座標編號	以 1~64 指定需要示教的用戶坐標系。
原點、點位 1...	以點變數指定用戶座標的點數據
L/R	將用戶座標原點對準左(1 號)或右(2 號)。可省略
X/Y	將連接原點與 X 軸或 Y 軸指定的直線定義為本地坐標系的 X 軸或 Y 軸。可省略。 默認 X

[使用案例]

```
Function main

    '以原點和相對於基礎坐標系的角度定義用戶坐標系
    Local 1 , P1
    Local 2 , XY(100 , 200 , 300 , 60)

    Local 3 , P1 , P2 , X              '兩點示教，兩點指定 X 軸

    Local 4 , P1 , P2 , P3             '三點示教

    Local 5 , (P1:P11) , (P2:P12)      '四點示教

Fend
```

[功能]

用於定義和顯示工具坐標系

[語法]

TLSet {工具座標編號, {工具設置數據}}

[參數說明]

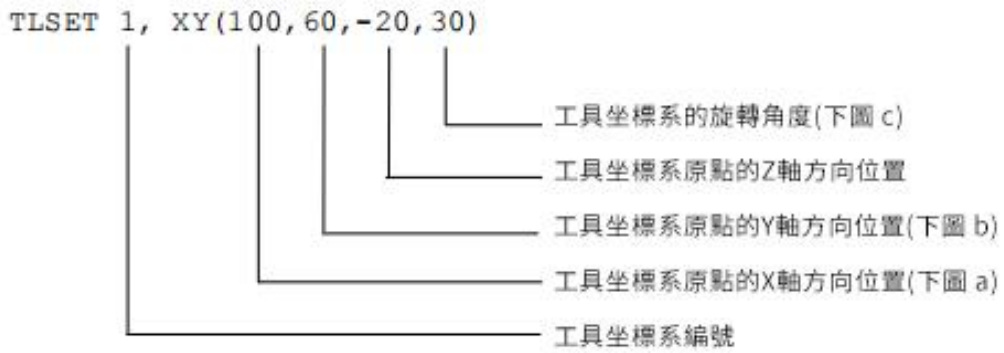
工具座標編號 以 1~64 指定需要示教的工具坐標系。

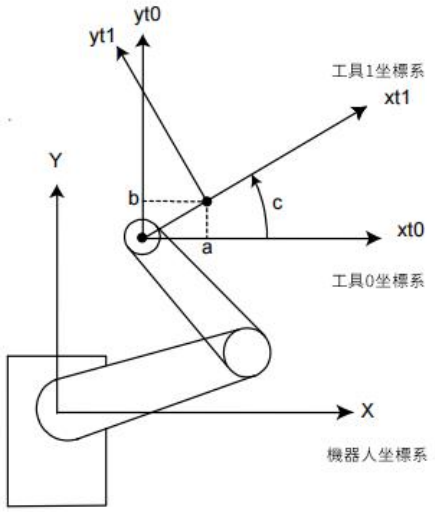
工具設置數據 以點編號、點名稱、點變數來設置工具坐標系的原點和方向。

注意事項：省略所有參數，則顯示所有 TLSet 的設置

[使用案例]

```
Function main
    TLSet 1,XY(50 , 100 , -20 , 30)
    TLSet 2 , P10 +X(20)
Fend
```





[功能]

用於選擇工具坐標系，或返回當前已選擇的工具坐標系

[語法]

Tool {工具座標編號}

[參數說明]

工具座標編號　　以 1~64 指定選擇的工具坐標系。

[使用案例]

```
Function main

    Tool 1                                  '選擇 1 號工具坐標系

    If Tool = 0 Then
        Tool 2
    EndIf

Fend
```

[功能]

使用三點法定義工具坐標系

[語法]

TCalib 工具座標編號, 工具輔助點 1, 工具輔助點 2, 工具輔助點 3

[參數說明]

工具座標編號 以 1~64 指定選擇的工具坐標系。

工具輔助點 1 以安裝在機械手絲杆的治具末端中心對準示教標識物。

工具輔助點 2 在工具輔助點 1 的基礎上 U 軸順時針或逆時針旋轉 60 ~ 120°，並使用寸動功能將治具末端中心再次對準示教標識物。

工具輔助點 3 在工具輔助點 2 的基礎上 U 軸向同一旋轉方向再次旋轉 60 ~ 120°，並使用寸動功能將治具末端中心再次對準示教標識物。

[使用案例]

```
Function main

    TCalib 1 , P0 , P1 , P2                    '以 P0、P1、P2 作為輔助點示教 1 號工具座標

Fend
```

[功能]

用於設置 Jump、Go、Move 等運動指令，運動過程中停止運動的條件。

[語法]

Till {條件運算式}

[參數說明]

條件運算式　　運動過程中停止運動的條件。可使用各種 I/O 指令、運算符等。

[使用案例]

```
Function main
    Till Sw(1) = Off      '設置 Till 條件(DI1 為 OFF)
        Go P1 Till        '滿足前一行的條件時停止

    Till Sw(1) = On And Sw(1) = On '設置新的 Till 條件
        Move P2 Till                '滿足前一行的條件時停止

        Move P5 Till Sw(10) = On    '滿足該行條件時停止
Fend
```

[功能]

用於返回指定關節的角度或位置的函數

[語法]

Agl(關節編號)

[參數說明]

關節編號	以整數指定關節編號。Scara 機器人範圍 1~4，六軸 1~6。如有拓展軸則根據拓展軸增加編號

[使用案例]

```
Function main
    Print Agl(1) , Agl(2)
Fend
```

[功能]

用於將機器人各關節角度轉換為脈衝值

[語法]

AglToPls(關節位置 1, 關節位置 2, 關節位置 3, 關節位置 4, {關節位置 5, 關節位置 6}....)

[參數說明]

關節位置 根據順序，依次指定各個關節的關節角度。可根據機型和拓展軸酌情增加形參數量。

指令配合 做為 Go 參數時執行絕對關節運動，作為 Print 參數時輸出脈衝值，其餘用法均返回空間點位。

[使用案例]

```
Function main

    P1 = AglToPls(0 , 0 , 0 , 90 , 0 , 0)
    Go P1

    Move AglToPls(0 , 0 , 0 , 90 , 0 , 0)

    Go AglToPls(0 , 0 , 0 , 90 , 0 , 0)

Fend
```

[功能]

用於設置/顯示 Jump、Jump3、Jump3CP 命令的 Arch 參數

[語法]

Arch {Arch 編號{, 轉移距離, 接近距離}}
Arch(Arch 編號, 參數編號)

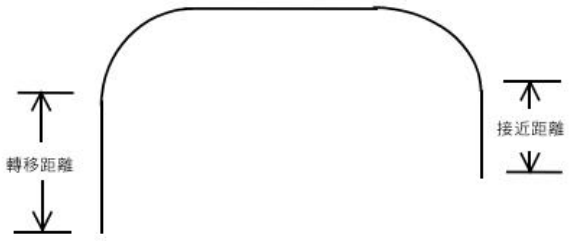
[參數說明]

Arch 編號 以整數 0~6 指定 Arch 編號。不指定的情況下，系統有默認值，詳情看下表格。

轉移距離 從出發點算起的垂直距離，單位 mm

轉移距離 從目標點算起的垂直距離，單位 mm

參數編號 1:轉移距離。2:接近距離



Arch 表格默認值

Arch 編號	轉移距離	接近距離
0	30	30
1	40	40
2	50	50
3	60	60
4	70	70
5	80	80
6	90	90

[使用案例]

```
Function main

    Arch 0 , 20 , 20                '給 Arch 0 設置兩個距離參數

    Jump P1 C0                      '在 Jump 中啟用 Arch0

    If Arch(0,1) = 30 And Arch(0,2) = 30 Then
        '判斷 Arch0 的兩個距離參數是否符合條件
        Arch 0 , 25 , 25
    EndIf

Fend
```

[功能]

用於設置在運動過程中根據條件運算式成立保存座標

[語法]

Find 條件運算式

[參數說明]

條件運算式　　運動過程中保存座標的條件。可使用各種 I/O 指令、運算符等。

[使用案例]

```
Function main

    Go P10 Find Sw(5) = On              '運動過程中 DI5 置為 On 時保存座標

    If PosFound Then
        Go FindPos                      '通過 FindPos 獲取 Find 保存的座標
    Else
        Print "運動過程中 DI5 未置為 ON"
    EndIf

Fend
```

[功能]

用於返回 Find 保存的座標

[語法]

FindPos
Go FindPos

[參數說明]

[使用案例]

```
Function main

    Go P10 Find Sw(5) = On              '運動過程中 DI5 置為 On 時保存座標

    If PosFound Then
        Go FindPos                      '通過 FindPos 獲取 Find 保存的座標
    Else
        Print "運動過程中 DI5 未置為 ON"
    EndIf

Fend
```

[功能]

返回機器人的到位狀態

[語法]

InPos

[參數說明]

注意事項:

1. 完成定位時返回 True
2. 機器人運動中返回 False

[使用案例]

```
Function main

    P0 = XY(0 , -100 , 0 , 0)
    P1 = XY(0 , 100 , 0 , 0)
    Xqt MonitorPosition
    Do
        Jump P0
        Wait 0.5
        Jump P1
        Wait 0.5
    Loop
Fend

Function MonitorPosition
    Boolean oldInPos , Pos

    Do
        Pos = InPos
        If Pos <> oldInPos Then
            Print "InPos = " , Pos
        EndIf

        oldInPos = Pos

    Loop
Fend
```

[功能]

用於根據關節角度返回機器人座標

[語法]

JA(關節角度 1, 關節角度 2, 關節角度 3, 關節角度 4, {關節角度 5, 關節角度 6}....)

[參數說明]

關節角度	根據順序依次指定各個關節的關節角度。單位：旋轉關節 deg。移動關節 mm。 可根據機型和拓展軸酌情增加形參數量。

[使用案例]

```
Function main
    P10 = JA(60 , 30 , -50 , 45)
    Go JA(135 , 90 , -50 ,90)
    P3 = JA(0 , 0 , 0 , 0 , 0 , 0)
Fend
```

[功能]

用於在關節坐標系中顯示當前機器人位置。

[語法]

Joint

[參數說明]

[使用案例]

```
Function main
    Joint
    '列印資訊:Joint: 1:-11.880deg 2:30.432deg 3: -21.902mm 4:-18.552deg
Fend
```

[功能]

返回 Jump、Jump3、Jump3CP 運動過程中 Sense 條件是否成立

[語法]

JS

[參數說明]

注意事項：

1. Sense 條件成立停在目標點上方時，返回 True
2. 正常完成了 Jump 動作並到達目標點，返回 False

[使用案例]

```
Function main
    Print searchSensor
Fend

Function searchSensor As Boolean
    Sense Sw(5) = On
    Jump P0
    Jump P1 Sense

    If JS = True Then
        Print "Sense 條件成立"
        searchSensor = True
    EndIf
Fend
```

[功能]

用於設置/顯示 Jump 運動過程中第 3 關節高度(Z 座標)初始值

[語法]

LimZ {Z 座標值}

[參數說明]

Z 座標值 指定第 3 關節在動作範圍內的座標值

[使用案例]

```
Function main
    LimZ -10                    '設置 Limz 的默認值
    Jump P1                     '執行 JUMP 時水準移動 Z 軸高度保持為-10
    Jump P2 LimZ -20            '執行 JUMP 時水準移動 Z 軸高度保持為-20
    Jump P3                     '執行 JUMP 時水準移動 Z 軸高度保持為-10
    If LimZ > -10 Then          '當設置值大於-10 時
        LimZ -30                '將 Limz 默認值設置為-30
    EndIf
Fend
```

[功能]

用於進行穿過指定點附近(不停止)的 PTP 運動。

[語法]

Pass 目標座標 1{, 目標座標 2, 目標座標 3....}

[參數說明]

目標座標 以點編號、點名稱設定需要移動到的位置。可根據需求後面酌情增加點位形參。

[使用案例]

```
Function main
    Pass P1 , P2 , P3       '從 P1 → P2 → P3 連續平滑過渡運動
Fend
```

[功能]

用於返回當前位置指定關節的脈衝值

[語法]

Pls(關節編號)

[參數說明]

關節編號　　以整數 1~關節數量指定需要獲取脈衝值的關節。

[使用案例]

```
Function main
    Real Axle_1 , Axle_2 , Axle_3 , Axle_4
    Axle_1 = pls(1)
    Axle_2 = pls(2)
    Axle_3 = pls(3)
    Axle_4 = pls(4)
    Print "一軸脈衝: ",Axle_1
    Print "二軸脈衝: ",Axle_2
    Print "三軸脈衝: ",Axle_3
    Print "四軸脈衝: ",Axle_4
Fend
```

[功能]

用於返回 Find 命令時執行的狀態

[語法]

PosFound

[參數說明]

注意事項:

1. 運動過程中 Find 成立則返回 True，否則返回 False

[使用案例]

```
Function main

    Go P10 Find Sw(5) = On '運動過程中 DI5 置 On 時保存座標

    If PosFound Then
        Go FindPos          '通過 FindPos 獲取 Find 保存的座標
    Else
        Print "運動過程中 DI5 未置為 ON"
    EndIf
Fend
```

[功能]

將機械臂以 PTP 移動到指定的各關節脈衝值的位置。或返回脈衝值對應的點位。

[語法]

Pulse {關節 1 脈衝值, 關節 2 脈衝值, 關節 3 脈衝值, 關節 4 脈衝值{, 關節 5 脈衝值, ...}}

Pulse(關節 1 脈衝值, 關節 2 脈衝值, 關節 3 脈衝值, 關節 4 脈衝值{, 關節 5 脈衝值, ...})

[參數說明]

關節脈衝值	根據順序依次填入各關節的脈衝值，結合機器人實際軸數可酌情增加形參

[使用案例]

```
Function main
    Pulse 123456 , 222333 , 333444 , 111111
    '移動到指定脈衝位置

    P1 = Pulse(123456 , 222333 , 333444 , 111111)
    '將指定脈衝的點位保存到 P1

    Pulse '列印當前所有軸脈衝值
Fend
```

[功能]

用於選擇指定的負載參數，或返回指定的負載參數

[語法]

RobLoad {負載編號}

[參數說明]

負載編號	以整數 1~64 指定需要選擇的負載

[使用案例]

```
Function main
    RobLoad 3
    Go P1

    If RobLoad = 0 Then
        RobLoad 1
    EndIf
Fend
```

[功能]

用於設置/顯示 Jump、Jump3、Jump3CP 停在目標點上方的條件

[語法]

Sense {條件運算式}

[參數說明]

條件運算式 運動過程中停在目標點上方的條件。可使用各種 I/O 指令、運算符等。

[使用案例]

```
Function main
    Sense Sw(5) = Off
    Jump P1 C2 Sense
    '當 DI5 為 Off 時停在目標點上方

    If JS = True Then
        Print "機器人已停在目標點上方"
    EndIf
    On 1; Wait 0.2; Off 1
Fend
```

[功能]

用於返回 Till 的狀態

[語法]

TillOn

[參數說明]

注意事項:

1. 運動過程中 Till 條件成立則返回 True。否則返回 False

[使用案例]

```
Function main
    Go P1 Till Sw(1) = On

    If TillOn Then
        Print "因為 DI1 置為 On, 已停止運動"
    EndIf
Fend
```

[功能]

用於在運動平滑指令後等待機器人進行減速停止後再將程式移交給後續程式

[語法]

WaitPos

[參數說明]

注意事項：

1. 通常情況下，在運動平滑有效的情況下(CP On 或運動指令帶 CP 時)，動作命令會在開始減速的同時將控制提前移交給後續的程式。如果不想提前移交，可以在後面插入 WaitPos，則會完成減速動作之後才移交。

[使用案例]

```
Function main
    Off 1
    CP On
    Move P1
    Move P2
    WaitPos          '等待機器人減速
    On 1
    CP Off
Fend
```

[功能]

根據指定的數據返回點位

[語法]

XY(X 座標值, Y 座標值, Z 座標值, U 座標值{, V 座標值, W 座標值})

[參數說明]

座標值 根據順序填入各個軸的座標。根據實際機器人型號酌情增加 V 軸和 W 軸的座標值。

單位 mm

[使用案例]

```
Function main
    P10 = XY(60 , 30 , -50 , 45) + P20
    '索引 P20 點位的數據加上指定偏差座標生成 P10 點位

    P10 = XY(60 , 30 , -50 , 45) /L
    '將指定的座標以及手系生成為 P10 點位

    Go XY(10 , 20 , 30 , 40)
    '圓弧移動至指定座標位置

Fend
```

[功能]

用於設置(修改)點的座標值。

[語法]

CX(點位) {= 座標值}

CY(點位) {= 座標值}

CZ(點位) {= 座標值}

....

[參數說明]

點位	點編號、點名稱
座標值	設置的座標值，單位 mm

[使用案例]

```
Function main
    Double New_Y

    CX(P1) = 123                '將 P1 的 X 座標改為 123mm
    CU(P1) = 321                '將 P1 的 U 座標改為 321deg

    If CY(P1) > 200 Then
        CY(P1) = 200            '當 P1 的 Y 座標大於 200，強制等於 200
    EndIf

    P1 = Here                   '獲取當前位置並賦值 P1 點
    Print CX(P1)                '列印當前位置 X 座標
    New_Y = CY(P1)              '將當前 Y 座標賦值給 New_Y 變數
Fend
```

[功能]

用於設置(修改)碰撞檢測係數。

[語法]

TargetOK(點位)

....

[參數說明]

點位　　點編號、點名稱

[使用案例]

```
Function main
    If TargetOK(P1) = False Then
        Print "該點位無法到達"
    ElseIf TargetOK(P1) = True Then
        Go P1
    EndIf
Fend
```

[功能]

用於設置(修改)碰撞檢測係數。

[語法]

```
ColSup    On/Off [,level]
ColSup
....
```

[參數說明]

On/Off	是否需要設定碰撞檢測係數
level	需要設定的碰撞檢測係數值，取值範圍：1 - 300

[使用案例]

```
Function main
    ColSup On,50          '設置碰撞檢測係數
    ColSup Off            '關閉碰撞檢測設置
    Print ColSup          '列印當前碰撞檢測係數
Fend
```

[功能]

用於設置(修改)機器人零位。

[語法]

ZeroPls　脈衝 1 , 脈衝 2 , 脈衝 3 [, 脈衝 4 , 脈衝 5 , 脈衝 6]

ZeroPls

....

[參數說明]

脈衝 1 , 脈衝 2 , 脈衝 3 [, 脈衝 4 , 脈衝 5 , 脈衝 6]	各成員軸需要設置零位的脈衝值

[使用案例]

```
Function main

    ZeroPls 0 , 0  , 0  , 0
    '將 XYZU 四軸零點位置設置為脈衝值為 0 處

    Print ZeroPls
    '列印當前的零位

Fend
```

[功能]

用於設置(修改)手系。

[語法]

Hand [點位 , 手系]
Hand

....

[參數說明]

點位	點編號、點名稱
手系	修改點位手系：Lefty 為左手、Righty 為右手。當設定為空時則返回點位手系

[使用案例]

```
Function main

    Print Hand(P0)          '列印 P0 點手系

    Hand P0,Righty          '設置 P0 點為右手系

    Print Hand              '列印當前手系

Fend
```

[功能]

用於設置(修改)肘姿態。

[語法]

```
Elbow [點 , 姿態]
Elbow
....
```

[參數說明]

點	點編號、點名稱
姿態	點位所屬得肘部姿態：Above 為肘部朝上、Below 為肘部朝下。

[使用案例]

```
Function main
    Print Elbow(P0)          '列印 P0 點姿態
    Elbow P0 , Above         '設置 P0 點姿態，肘部朝上
    Print Elbow              '列印當前姿態
Fend
```

[功能]

用於設置(修改)手腕姿態。

[語法]

```
Wrist [點 , 姿態]
Wrist
....
```

[參數說明]

點	點編號、點名稱
姿態	點位所屬得手腕姿態：Flip 為手腕翻轉、NoFlip 為手腕不翻轉。

[使用案例]

```
Function main
    Print Wrist(P0)                 '列印 P0 點手腕姿態
    Wrist P0 , NoFlip               '設置 P0 點姿態，肘部朝上
    Print Wrist                     '列印當前姿態
Fend
```

[功能]

用於返回點是否已定義。

[語法]

PDef（點編號）

PDef（P 點編號）

PDef（P（點編號））

PDef（標籤名稱）

[參數說明]

點編號	以點位檔中的排序號做索引。
P 點編號	以點位檔中的排序號做索引。
P(點編號)	以點位檔中的排序號做索引。
標籤名稱	賦予標籤變數後，依據變數值在點位檔中的排序號做索引。

[使用案例]

```
Function main

    Print "0:" , PDef(0)                  '列印 P0 號點是否存在定義
    Print "P1:" , PDef(P1)                '列印 P1 號點是否存在定義

    Print "P(1):" , PDef(P(1))            '列印 P1 號點是否存在定義

    Integer Point_Num
    Point_Num =  0
    Print "Point_Num:" , PDef(Point_Num) '列印對應點是否存在定義
Fend
```

[回傳說明]

回傳數據 1	TRUE / FALSE , TRUE 為點位存在定義，FALSE 為點位未定義。

[功能]

用於刪除點位。

[語法]

PDel 點編號

PDel 起始點編號 , 結束點編號

[參數說明]

點編號	以點位檔中的排序號做索引。
起始點編號	從指定的編號開始刪除點位。
結束點編號	從指定的編號結束刪除點位。

[使用案例]

```
Function main

    PDel 1                  '刪除 P1 號點位資訊

    PDel 1 , 5              '刪除 P1 - P5 號點位資訊

    SavePoints "robot1"     '保存至點位表

Fend
```

[功能]

用於設置指定點數據的標籤。

[語法]

```
PLabel 點編號,新標籤
```

[參數說明]

點編號　　以點位檔中的排序號做索引。

新標籤　　為指定的點編號賦予新的標籤(暫不支持中文)。

[使用案例]

```
Function main

    PLabel 1 , "New_Point"        '為 P1 號點賦予新標籤

    Go P(New_Point)

    SavePoints "robot1"           '保存至點位檔

Fend
```

[功能]

用於返回指定點數據的標籤。

[語法]

PLabel$ (點編號)

PLabel$ (P 點編號)

PLabel$ (P(點編號))

[參數說明]

點編號	以點位檔中的排序號做索引。
P 點編號	以點位檔中的排序號做索引。
P(點編號)	以點位檔中的排序號做索引。

注意事項:

1. 若使用該指令索引空點位或索引的點位不存在標籤時，不做任何輸出。

[使用案例]

```
Function main

    Print PLabel$(1)                '列印 1 號點位標籤

    Print PLabel$(P1)               '列印 1 號點位標籤

    Print PLabel$(P(1))             '列印 1 號點位標籤

    If PLabel$(2) = "" Then         '判斷 2 號點標籤等於空
        Print "該點位不存在標籤"
    EndIf
Fend
```

[功能]

用於返回指定標籤名稱的點編號。

[語法]

PNumber（標籤名稱）

PNumber（“標籤名稱”）

[參數說明]

標籤名稱	點位數據中的其中一項，標籤。

[使用案例]

```
Function main

    Print PNumber(EA)           '列印點位標籤為 EA 的點編號
    Print PNumber("AA")         '列印點位標籤為 AA 的點編號

Fend
```

[功能]

用於顯示指定點數據。

[語法]

PList

PList 點編號

PList 起始點編號 , 結束點編號

[參數說明]

點編號	以點位檔中的排序號做索引。
P 點編號	以點位檔中的排序號做索引。
P(點編號)	以點位檔中的排序號做索引。

[使用案例]

```
Function main

    PList               '列印所有點位數據

    PList 1             '列印 1 號點位數據

    PList 2 , 10        '列印 2 號點至 10 號點數據

Fend
```

[功能]

用於指定 / 獲取點位的用戶座標資訊。

[語法]

```
PLocal  (點編號)
PLocal  (點編號) = Value
```

[參數說明]

點編號	以點位檔中的排序號做索引。
Value	指定的用戶坐標系。

[使用案例]

```
Function main

    Print PLocal(1)          '列印 1 號點所屬的用戶座標

    PLocal(1) = 1            '設置 1 號點所屬的用戶座標

Fend
```

[功能]

用於設置指定點的描述。

[語法]

```
PDescription  點編號 ,  新描述
```

[參數說明]

點編號　　以點位檔中的排序號做索引。

新描述　　顯示於點位檔中的描述數據。

[使用案例]

```
Function main

    PDescription 1 , "新的描述"          '將 P1 號點的描述內容改為 "新的描述"

    Print PDescription$(1)               '列印 P1 號點的描述

Fend
```

[功能]

用於返回指定點的描述。

[語法]

PDescription$ （點編號）

PDescription$ （P 點編號）

PDescription$ （P（點編號））

PDescription$ （標籤名稱）

[參數說明]

點編號	以點位檔中的排序號做索引。
P 點編號	以點位檔中的排序號做索引。
P(點編號)	以點位檔中的排序號做索引。
標籤名稱	賦予標籤變數後，依據變數值在點位檔中的排序號做索引。

[使用案例]

```
Function main

    PDescription 1 , "新的描述"      '將 P1 號點的描述內容改為 "新的描述"

    Print PDescription$(1)           '列印 P1 號點的描述

    Print PDescription$(P1)          '列印 P1 號點的描述

    Print PDescription$(P(1))        '列印 P1 號點的描述

Fend
```

I/O 指令詳細說明

[功能]

延時指令、或在一定時間內等待條件成立。

[語法]

Wait 時間

Wait 條件運算式

Wait 條件運算式, 時間

[參數說明]

時間	以 0~2147483 之間的數指定延時的時間。單位：秒。最小延時 0.01 秒。
條件運算式	運動過程中停在目標點上方的條件。可使用各種 I/O 指令、運算符等。

[使用案例]

```
Function main

    '等待輸入 0 變為 On 狀態
    Wait Sw(0) = On

    '等待 60.5 秒後繼續執行
    Wait 60.5

    '等待輸入 0 變為 OFF、輸入 1 變為 On 狀態
    Wait Sw(0) = Off And Sw(1) = On

    '等待寄存器 bit 位 0 變為 On 或寄存器 bit 位 1 變為 On
    Wait MemSw(0) = On Or MemSw(1) = On

    '等待 1 秒鐘，然後將輸出 1 設為 On
    Wait 1; On 1

Fend
```

[功能]

以位為單位，用於對表示 Wait 輸入條件的值進行位 And 運算。

[語法]

```
Wait 條件運算式 Mask Bit 位最大值 = 條件返回值
```

[參數說明]

Wait	延時指令、或在一定時間內等待條件成立。
條件運算式	指定的 DIO bit 位或組別、寄存器地址或 bit 位。
Mask	用於對表示 Wait 輸入條件的值進行位 And 運算。
Bit 位最大值	索引的 Bit 位最大值。
條件達成值	滿足當前條件運算式的目標值。

[使用案例]

```
Function main
    Do
        Wait InW(0) Mask 1 = 0
        '等待 16 位 DI 組 0(前 1bit 位)返回值為 0 時滿足跳出條件

        Wait InW(0) Mask 3 = 2
        '等待 16 位 DI 組 0(前 2bit 位)返回值為 2 時滿足跳出條件

        Wait InW(0) Mask 7 = 4
        '等待 16 位 DI 組 0(前 3bit 位)返回值為 4 時滿足跳出條件

        Wait InW(0) Mask 15 = 8
        '等待 16 位 DI 組 0(前 4bit 位)返回值為 8 時滿足跳出條件

        Wait InW(0) Mask 31 = 16
        '等待 16 位 DI 組 0(前 5bit 位)返回值為 16 時滿足跳出條件

        Wait InW(0) Mask 63 = 32
        '等待 16 位 DI 組 0(前 6bit 位)返回值為 32 時滿足跳出條件

    Loop
Fend
```

[功能]

控制 DO 置 On，或置 On 一定時間後置 Off

[語法]

On DO 編號/DO 標籤 {, 時間}

[參數說明]

DO 編號/DO 標籤	以整數指定 DO 編號，或使用 DO 標籤指定
時間	以秒指定 DO 置 On 的時長。達到該時長後自動置 Off。

[使用案例]

```
Function main

    On 1                'Do1 至為 on
    On 3                'Do3 至為 on

    On 5 , 1            'Do5 至為 on，1 秒後置 off

Fend
```

[功能]

控制 DO 置 Off，或置 Off 一定時間後置 On

[語法]

Off DO 編號/DO 標籤 {, 時間}

[參數說明]

DO 編號/DO 標籤	以整數指定 DO 編號，或使用 DO 標籤指定
時間	以秒指定 DO 置 Off 的時長。達到該時長後自動置 On。

[使用案例]

```
Function main

    Off 1               'Do1 至為 on
    Off 3               'Do3 至為 on

    Off 5 , 1           'Do5 至為 on，1 秒後置 off

Fend
```

[功能]

用於返回指定 DO 狀態

[語法]

Oport(DO 編號)

[參數說明]

DO 編號　　以整數指定需要獲取狀態的 DO

注意事項:

1. DO 為 On 返回 1
2. DO 為 Off 返回 0

[使用案例]

```
Function main

    If Oport(1) = 0 Then
            On 1
    EndIf

    Wait Oport(3)
Fend
```

[功能]

用於返回指定 DI 狀態

[語法]

Sw(DI 編號)

[參數說明]

DI 編號　　以整數指定需要獲取狀態的 DI

注意事項:

3. DO 為 On 返回 1

4. DO 為 Off 返回 0

[使用案例]

```
Function main

    If Sw(1) = 0 Then
        SetSw 1 ,  On
    EndIf

    Wait Sw(1)
Fend
```

[功能]

用於將指定 DI 置 On 或置 Off

[語法]

SetSw DI 編號, 狀態

[參數說明]

DI 編號	以整數指定需要控制的 DI
狀態	0 為 Off，1 為 On。填入 Off 或 On 也能生效。

[使用案例]

```
Function main

    SetSw 1 , On
    SetSw 4 , Off


Fend
```

[功能]

根據 8421 返回 8 位 DI 的狀態，0~255

[語法]

In(DI 組編號)

[參數說明]

DI 組編號　　以整數指定需要讀取狀態的一組 DI。

注意事項：

1. DI 組編號取 0，返回 DI0~DI7 的狀態。輸入 1，返回 DI8~DI15 的狀態。以此類推。

2. 8421 原理示意圖如下：

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[使用案例]

```
Function main

    Print In(0)

    Integer A1
    A1 =  In(1)

    Select A1
    Case 0
        Go P1

    Case 8
        Go P2

    Default
        Go P3

    Send
Fend
```

[功能]

根據 8421 設定 8 位 DI 的狀態，0~255

[語法]

SetIn DI 組編號, 狀態值

[參數說明]

DI 組編號　　以整數指定需要設定狀態的一組 DI。

狀態值　　根據 8421 設定該組 DI 的狀態

注意事項：

1. DI 組編號取 0，是 DI0~DI7。輸入 1，是 DI8~DI15。以此類推。

2. 8421 原理示意圖如下：

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[使用案例]

```
Function main
    SetIn 0 ,85              '將 DI0、2、4、6 置 On
Fend
```

[功能]

根據 8421 設置/返回 8 位 DO 的狀態，0~255

[語法]

Out DO 組編號, 狀態值

Out(DO 組編號)

[參數說明]

DO 組編號　　以整數指定需要設置或讀取狀態的一組 DO。

狀態值　　根據 8421 設定該組 DO 的狀態

注意事項：

1. DO 組編號取 0，是 DO0~DO7。輸入 1，是 DO8~DO15。以此類推。

2. 8421 原理示意圖如下：

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[使用案例]

```
Function main
    Out 0 ,85                    '將 DO0、2、4、6 置 On

    If Out(0) = 0 Then           '讀取 DO0~DO7 的狀態
        Out 0 ,85
    EndIf
Fend
```

[功能]

根據 8421 返回 16 位 DI 的狀態，0~65535

[語法]

InW(DI 組編號)

[參數說明]

DI 組編號　　以整數指定需要讀取狀態的一組 DI。

注意事項：

1. DI 組編號取 0，是 DI0~DI15 的狀態。輸入 1，是 DI16~DI31 的狀態。以此類推。

2. 8421 原理示意圖如下：

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[使用案例]

```
Function main

    Print InW(0)

    Integer A1
    A1 =  InW(1)

    Select A1
    Case 0
        Go P1

    Case 8
        Go P2

    Default
        Go P3

    Send
Fend
```

[功能]

根據 8421 設定 16 位 DI 的狀態，0~65535

[語法]

SetInW DI 組編號 ， 狀態值

[參數說明]

DI 組編號　　以整數指定需要設定狀態的一組 DI。

狀態值　　根據 8421 設定該組 DI 的狀態

注意事項：

1. DI 組編號取 0，是 DI0~DI15。輸入 1，是 DI16~DI31。以此類推。

2. 8421 原理示意圖如下：

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[使用案例]

```
Function main
    SetInW 0 ,85              '將 DI0、2、4、6 置 On
Fend
```

[功能]

根據 8421 設置/返回 16 位 DO 的狀態，0~65535

[語法]

OutW DO 組編號, 狀態值
OutW(DO 組編號)

[參數說明]

DO 組編號　　以整數指定需要設置或讀取狀態的一組 DO。

狀態值　　根據 8421 設定該組 DO 的狀態

注意事項：

1. DO 組編號取 0，是 DO0~DO15。輸入 1，是 DO16~DO31。以此類推。

2. 8421 原理示意圖如下：

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[使用案例]

```
Function main
    OutW 0 ,85                  '將 D00、2、4、6 置 On

    If OutW(0) = 0 Then         '讀取 D00~D015 的狀態
        OutW 0 ,85
    EndIf
Fend
```

[功能]

根據 8421 返回 8 位 DI 的狀態，0~99

[語法]

InBCD(DI 組編號)

[參數說明]

DI 組編號　　以整數指定需要讀取狀態的一組 DI。

注意事項：

1. DI 組編號取 0，返回 DI0~DI7 的狀態。輸入 1，返回 DI8~DI15 的狀態。以此類推。
2. 該函數與 In 的區別在於取值範圍不同。In 取值範圍是 0~255。
3. 8421 原理示意圖如下：

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[使用案例]

```
Function main

    Print InBCD(0)

    Integer A1
    A1 =  InBCD(1)

    Select A1
    Case 0
        Go P1

    Case 8
        Go P2

    Default
        Go P3

    Send
Fend
```

[功能]

根據 8421 設置 8 位 DO 的狀態，0~99

[語法]

OpBCD DO 組編號, 狀態值

[參數說明]

DO 組編號　　以整數指定需要設置或讀取狀態的一組 DO。

狀態值　　根據 8421 設定該組 DO 的狀態

注意事項：

1. DO 組編號取 0，是 DO0~DO7。輸入 1，是 DO8~DO15。以此類推。
2. 該函數與 Out 的區別在於取值範圍不同。Out 取值範圍是 0~255。
3. 8421 原理示意圖如下：

返回值	7	6	5	4	3	2	1	0
1	Off	Off	Off	Off	Off	Off	Off	On
5	Off	Off	Off	Off	Off	On	Off	On
15	Off	Off	Off	Off	On	On	On	On
255	On	On	On	On	On	On	On	On

[使用案例]

```
Function main
    OpBCD 0 ,85                    '將 DO0、2、4、6 置 On
Fend
```

[功能]

以 32 位浮點數返回 DI 狀態。

[語法]

InReal(DI 組編號)

[參數說明]

DI 組編號　　以整數指定需要設置或讀取狀態的一組 DO。

注意事項：

1. DI 組編號取 0，是 DI0~DI31。輸入 1，是 DI32~DI63。以此類推。

[使用案例]

```
Function main
    Real realVal

    realVal =  InReal(32)
Fend
```

[功能]

以 32 位浮點數設置/返回 DO 狀態。

[語法]

OutReal DO 組編號, 狀態值

OutReal(DO 組編號)

[參數說明]

DO 組編號　　以整數指定需要設置或讀取狀態的一組 DO。

狀態值　　根據 32 位浮點數設定該組 DO 的狀態

注意事項：

1. DO 組編號取 0，是 DO0~DO31。輸入 1，是 DO32~DO63。以此類推。

[使用案例]

```
Function main
    OutReal 32 , 2.543
Fend
```

[功能]

讀取模擬量 AI 的值

[語法]

AIO_InW(AI 編號)

[參數說明]

AI 編號　　以整數指定需要讀取狀態的模擬量 AI。

[使用案例]

```
Function main
    If AIO_InW(1) > 123456 Then
        Print "模擬量大於 123456"
    EndIf
Fend
```

[功能]

設置/讀取模擬量 AO 的值

[語法]

```
AIO_OutW AI 編號, 模擬量值
AIO_OutW(AI 編號)
```

[參數說明]

AI 編號　　以整數指定需要設置或讀取狀態的模擬量 AO。

模擬量值　　為該 AO 設定值

[使用案例]

```
Function main
    If AIO_OutW(1) > 123456 Then
        AIO_OutW 1 , 0          'A01 模擬量大於 123456 清零
    EndIf
Fend
```

[功能]

指定用戶寄存器中的一位 bit 為 1

[語法]

```
MemOn bit 位
```

[參數說明]

bit 位　　指定第幾個 bit 為 1

注意事項：

1. 一個用戶寄存器是 16 位，因此 0~15bit 是第 0 個寄存器。16~31bit 是第 1 個寄存器，以此類推。

[使用案例]

```
Function main
    MemOn 3          '將第一個寄存器的第 3 個 Bit 置 1
    MemOn 16         '將第二個寄存器的第 0 個 Bit 置 1
Fend
```

[功能]

指定用戶寄存器中的一位 bit 為 0

[語法]

```
MemOff bit 位
```

[參數說明]

bit 位　　指定第幾個 bit 為 0

注意事項：

1. 一個用戶寄存器是 16 位，因此 0~15bit 是第 0 個寄存器。16~31bit 是第 1 個寄存器，以此類推。

[使用案例]

```
Function main
    MemOff 3            '將第一個寄存器的第 3 個 Bit 置 0
    MemOff 16           '將第二個寄存器的第 0 個 Bit 置 0
Fend
```

[功能]

獲取用戶寄存器中的一位 bit 的值

[語法]

MemSw(bit 位)

[參數說明]

bit 位　　指定第幾個 bit

注意事項：

1. 一個用戶寄存器是 16 位，因此 0~15bit 是第 0 個寄存器。16~31bit 是第 1 個寄存器，以此類推。

[使用案例]

```
Function main

    If MemSw(16) = 1 Then                    '獲取第二個寄存器第 0 個 Bit 值
        Go P1
    Else
        Go P2
    EndIf

Fend
```

[功能]

獲取用戶寄存器中 8 位的值

[語法]

```
MemIn(bit 編號, {Signed/Unsigned})
```

[參數說明]

bit 編號　　指定第幾組 bit

Signed/Unsigned　　讀取有符號數值或無符號數值。默認無符號。

注意事項：

1. 一個用戶寄存器是 16 位，因此 0~1 分別是第 0 個寄存器的低八位和高八位。2~3 是第 1 個寄存器的低八位和高八位。以此類推。

[使用案例]

```
Function main

    Print MemIn(2)
    '獲取第一個寄存器的低八位

    Print MemIn(1 , Signal)
    '以有符號數值的形式獲取第 0 個寄存器的高八位

Fend
```

[功能]

設置用戶寄存器中 8 位的值

[語法]

```
MemOut bit 編號, 值
```

[參數說明]

bit 編號　　指定第幾組 bit

值　　給寄存器設定的值

注意事項：

1. 一個用戶寄存器是 16 位，因此 0~1 分別是第 0 個寄存器的低八位和高八位。2~3 是第 1 個寄存器的低八位和高八位。以此類推。

[使用案例]

```
Function main
    MemOut 1 , 123
    '給第 0 個寄存器的高八位設定為 123
Fend
```

[功能]

獲取用戶寄存器中 16 位的值

[語法]

```
MemInW(寄存器編號, {Signed/Unsigned})
```

[參數說明]

寄存器編號　　指定第幾個寄存器

Signed/Unsigned　　讀取有符號數值或無符號數值。默認無符號。

注意事項：

1. 一個用戶寄存器是 16 位，因此 0 是第 0 個寄存器。1 是第 1 個寄存器。以此類推。

[使用案例]

```
Function main

    Print MemInW(1)
    '獲取第一個寄存器的值

    Print MemInW(2 , Signal)
    '以有符號數值的形式獲取第 2 個寄存器的高八位

Fend
```

[功能]

設置用戶寄存器中 16 位的值

[語法]

```
MemOutW 寄存器編號, 值
```

[參數說明]

寄存器編號　　指定第幾個寄存器

值　　給寄存器設定的值

注意事項：

1. 一個用戶寄存器是 16 位，因此 0 是第 0 個寄存器。1 是第 1 個寄存器。以此類推。

[使用案例]

```
Function main
    MemOutW 5 , 1234
    '給第 5 個寄存器的值設定為 1234
Fend
```

[功能]

獲取用戶寄存器中 32 位的值

[語法]

```
MemIn32(寄存器組, {Signed/Unsigned})
```

[參數說明]

寄存器組	指定第幾組寄存器
值	給寄存器設定的值

注意事項：

1. 一個用戶寄存器是 16 位，因此 0 是第 0 個寄存器+第 1 個寄存器合併的 32 位數值。1 是第 2 個寄存器+第 3 個寄存器合併的 32 位數值。以此類推。

[使用案例]

```
Function main

    Print MemIn32(0)
    '獲取第 0 個寄存器和第 1 個寄存器合併的 32 位數值

    Print MemIn32(1 , Signal)
    '以有符號數值的形式，獲取第 2 個寄存器和第 3 個寄存器合併的 32 位數值

Fend
```

[功能]

設置用戶寄存器中 32 位的值

[語法]

```
MemOut32 寄存器組, 值
```

[參數說明]

寄存器組	指定第幾組寄存器
值	給寄存器設定的值

注意事項：

1. 一個用戶寄存器是 16 位，因此 0 是第 0 個寄存器+第 1 個寄存器合併的 32 位數值。1 是第 2 個寄存器+第 3 個寄存器合併的 32 位數值。以此類推。

[使用案例]

```
Function main
    MemOut32 1 , 1234
    '給第 2 個寄存器和第 3 個寄存器合併的 32 位數值設置為 1234
Fend
```

[功能]

以 32 位浮點數形式獲取用戶寄存器中 32 位的值

[語法]

MemInReal(寄存器組)

[參數說明]

寄存器組　　指定第幾組寄存器

注意事項：

1. 一個用戶寄存器是 16 位，因此 0 是第 0 個寄存器+第 1 個寄存器合併的 32 位數值。1 是第 2 個寄存器+第 3 個寄存器合併的 32 位數值。以此類推。

[使用案例]

```
Function main

    Print MemInReal(0)
    '以 32 位浮點數獲取第 0 個寄存器和第 1 個寄存器合併的 32 位數值

Fend
```

[功能]

設置用戶寄存器中 32 位的值

[語法]

```
MemOutReal 寄存器組, 值
```

[參數說明]

寄存器組	指定第幾組寄存器
值	給寄存器設定的值

注意事項：

1. 一個用戶寄存器是 16 位，因此 0 是第 0 個寄存器+第 1 個寄存器合併的 32 位數值。1 是第 2 個寄存器+第 3 個寄存器合併的 32 位數值。以此類推。

[使用案例]

```
Function main
    MemOutReal 1 , 123.321
    '給第 2 個寄存器和第 3 個寄存器合併的 32 位數值設置為 123.321
Fend
```

[功能]

指定系統寄存器中的一位 bit 為 1

[語法]

SysMemOn bit 編號

[參數說明]

bit 編號　　指定第幾個 bit 為 1

注意事項：

1. 一個系統寄存器是 16 位，因此 0~15bit 是第 0 個寄存器。16~31bit 是第 1 個寄存器，以此類推。

[使用案例]

```
Function main
    SysMemOn 0          '將第 0 個系統寄存器的第 1 個 Bit 置 On
    SysMemOn 15         '將第 0 個系統寄存器的第 16 個 Bit 置 On
    SysMemOn 16         '將第 1 個系統寄存器的第 1 個 Bit 置 On
Fend
```

[功能]

指定系統寄存器中的一位 bit 為 0

[語法]

SysMemOff bit 編號

[參數說明]

bit 編號　　指定第幾個 bit 為 0

注意事項：

1. 一個系統寄存器是 16 位，因此 0~15bit 是第 0 個寄存器。16~31bit 是第 1 個寄存器，以此類推。

[使用案例]

```
Function main
    SysMemOff 0             '將第 0 個系統寄存器的第 1 個 Bit 置 Off
    SysMemOff 15            '將第 0 個系統寄存器的第 16 個 Bit 置 Off
    SysMemOff 16            '將第 1 個系統寄存器的第 1 個 Bit 置 Off
Fend
```

[功能]

獲取系統寄存器中的一位 bit 的值

[語法]

SysMemSw bit 編號

[參數說明]

bit 編號　　指定第幾組 bit

注意事項：

1. 一個系統寄存器是 16 位，因此 0~15bit 是第 0 個寄存器。16~31bit 是第 1 個寄存器，以此類推。

[使用案例]

```
Function main

    If SysMemSw(0) = On Then                    '第 0 個系統寄存器的第一位 bit 值為 on 時成立
        Go P1

    ElseIf SysMemSw(15) = On Then               '第 0 個系統寄存器的第十六位 bit 值為 on 時成立
        Go P2
    EndIf

Fend
```

[功能]

獲取系統寄存器中 8 位的值

[語法]

SysMemIn(bit 編號, {Signed/Unsigned})

[參數說明]

bit 編號	指定第幾組 bit
Signed/Unsigned	讀取有符號數值或無符號數值。默認無符號。

注意事項:

1. 一個系統寄存器是 16 位，因此 0~1 分別是第 0 個寄存器的低八位和高八位。2~3 是第 1 個寄存器的低八位和高八位。以此類推。

[使用案例]

```
Function main

    Print SysMemIn(0)
    '列印第 0 個寄存器的低八位

    Print SysMemIn(1 , Unsigned)
    '以無符號數值的形式'列印第 0 個寄存器的高八位

Fend
```

[功能]

設置系統寄存器中 8 位的值

[語法]

```
SysMemOut bit 編號, 值
```

[參數說明]

bit 編號	指定第幾組 bit
值	給寄存器設定的值

注意事項：

1. 一個系統寄存器是 16 位，因此 0~1 分別是第 0 個寄存器的低八位和高八位。2~3 是第 1 個寄存器的低八位和高八位。以此類推。

[使用案例]

```
Function main
    SysMemOut 1 , 123
    '設置第 0 個寄存器高八位為 123
Fend
```

[功能]

獲取系統寄存器中 16 位的值

[語法]

SysMemInW 寄存器編號, {Signed/Unsigned}])

[參數說明]

寄存器編號 指定第幾個寄存器

Signed/Unsigned 讀取有符號數值或無符號數值。默認無符號。

注意事項:

1. 一個系統寄存器是 16 位，因此 0 是第 0 個寄存器。1 是第 1 個寄存器。以此類推。

[使用案例]

```
Function main

    Print SysMemInW(0,Unsigned)
    '以無符號整形格式列印 0 號系統寄存器

    Print SysMemInW(1 , Signal)
    '以有符號數值的形式列印 1 號系統寄存器

Fend
```

[功能]

設置系統寄存器中 16 位的值

[語法]

```
SysMemOutW 寄存器編號, 值
```

[參數說明]

寄存器編號	指定第幾個寄存器
值	給寄存器設定的值

注意事項:

1. 一個系統寄存器是 16 位，因此 0 是第 0 個寄存器。1 是第 1 個寄存器。以此類推。

[使用案例]

```
Function main
    SysMemOutW 0 , 123
    '將第 0 個寄存器地址寫入值 123
Fend
```

[功能]

獲取系統寄存器中 32 位的值

[語法]

SysMemIn32 (寄存器組, {Signed/Unsigned})

[參數說明]

寄存器組	指定第幾組寄存器
Signed/Unsigned	讀取有符號數值或無符號數值。默認無符號。

注意事項:

1. 一個系統寄存器是 16 位，因此 0 是第 0 個寄存器+第 1 個寄存器合併的 32 位數值。1 是第 2 個寄存器+第 3 個寄存器合併的 32 位數值。以此類推。

[使用案例]

```
Function main

    Print SysMemIn32(0)
    '獲取第 0 個寄存器和第 1 個寄存器合併的 32 位數值

    Print SysMemIn32(1 , Unsigned)
    '以有符號數值的形式，獲取第 2 個寄存器和第 3 個寄存器合併的 32 位數值

Fend
```

[功能]

設置系統寄存器中 32 位的值

[語法]

SysMemOut32 寄存器組, 值

[參數說明]

寄存器組	指定第幾組寄存器
值	給寄存器設定的值

注意事項：

1. 一個系統寄存器是 16 位，因此 0 是第 0 個寄存器+第 1 個寄存器合併的 32 位數值。1 是第 2 個寄存器+第 3 個寄存器合併的 32 位數值。以此類推。

[使用案例]

```
Function main
    SysMemOut32 1 , 1234
    '給第 2 個系統寄存器和第 3 個系統寄存器合併的 32 位數值設置為 1234
Fend
```

[功能]

以 32 位浮點數形式獲取系統寄存器中 32 位的值

[語法]

SysMemInReal（寄存器組）

[參數說明]

寄存器組　　指定第幾組寄存器

注意事項：

1. 一個系統寄存器是 16 位，因此 0 是第 0 個寄存器+第 1 個寄存器合併的 32 位數值。1 是第 2 個寄存器+第 3 個寄存器合併的 32 位數值。以此類推。

[使用案例]

```
Function main

    Print SysMemInReal(0)
    '以 32 位浮點數獲取第 0 個寄存器和第 1 個寄存器合併的 32 位數值

Fend
```

[功能]

以 32 位浮點數形式設置系統寄存器中 32 位的值

[語法]

```
SysMemOutReal 寄存器組, 值
```

[參數說明]

寄存器組	指定第幾組寄存器
值	給寄存器設定的值

注意事項：

1. 一個系統寄存器是 16 位，因此 0 是第 0 個寄存器+第 1 個寄存器合併的 32 位數值。1 是第 2 個寄存器+第 3 個寄存器合併的 32 位數值。以此類推。

[使用案例]

```
Function main
    SysMemOutReal 1 , 123.321
    '給第 2 個寄存器和第 3 個寄存器合併的 32 位數值設置為 123.321
Fend
```

通訊指令詳細說明

[功能]

用於設置 TCP/IP 的通訊參數

[語法]

SetNet #通訊端口編號, IP 地址, 端口號{, 結束符, 流控制, 超時時間, 通訊協議}

[參數說明]

#通訊端口編號	指定需要修改通訊參數的端口編號。範圍：#201~216
IP 地址	用於 TCP/IP 通訊的 IP 地址。例如：192.168.1.1。
端口號	用於 TCP/IP 通訊的端口號。例如：8090。範圍：0~65535
結束符	通訊數據的末尾字元。可以設定 CR、LF、CRLF(回車、換行、回車換行)。可省略
流控制	指軟體流控制。設定為 NONE。可省略
超時時間	以秒設定收發的最長時間。0 為永不超時。可省略
通訊協議	當前只支持 TCP 協議。可省略

[使用案例]

```
Function main
    ReConnect:
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    If ChkNet(201) =  -1 Then
        Print "視覺 TCP 通訊錯誤，端口已打開，但是未建立通訊"
        Wait 1
        GoTo ReConnect
    ElseIf ChkNet(201) = -2 Then
        Print "視覺 TCP 通訊錯誤，其他任務正在使用端口"
        Wait 1
        GoTo ReConnect
    ElseIf ChkNet(201) = -3 Then
        Print "視覺 TCP 通訊錯誤，未打開端口"
        Wait 1
        GoTo ReConnect
    EndIf
Fend
```

[功能]

用於開啟 TCP/IP 通訊，或獲取 OpenNet 的線程編號。

[語法]

```
OpenNet #通訊端口編號 As [Client/Server]
OpenNet(通訊端口編號)
```

[參數說明]

#通訊端口編號　　指定需要修改通訊參數的端口編號。範圍：#201~216

Client/Server　　Client 為客戶端。Server 為伺服器。

[使用案例]

```
Function main
    ReConnect:
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    If ChkNet(201) =  -1 Then
        Print "視覺 TCP 通訊錯誤，端口已打開，但是未建立通訊"
        Wait 1
        GoTo ReConnect
    Elself ChkNet(201) = -2 Then
        Print "視覺 TCP 通訊錯誤，其他任務正在使用端口"
        Wait 1
        GoTo ReConnect
    Elself ChkNet(201) = -3 Then
        Print "視覺 TCP 通訊錯誤，未打開端口"
        Wait 1
        GoTo ReConnect
    Endlf
Fend
```

[功能]

用於等待 TCP/IP 通訊端口建立連接。

[語法]

```
WaitNet #通訊端口編號{, 超時時間}
```

[參數說明]

#通訊端口編號　　指定需要修改通訊參數的端口編號。範圍：#201~216

超時時間　　以秒設定等待時間。省略則無限等待。

[使用案例]

```
Function main
    ReConnect:
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    If ChkNet(201) =  -1 Then
        Print "視覺 TCP 通訊錯誤，端口已打開，但是未建立通訊"
        Wait 1
        GoTo ReConnect
    Elself ChkNet(201) = -2 Then
        Print "視覺 TCP 通訊錯誤，其他任務正在使用端口"
        Wait 1
        GoTo ReConnect
    Elself ChkNet(201) = -3 Then
        Print "視覺 TCP 通訊錯誤，未打開端口"
        Wait 1
        GoTo ReConnect
    Endlf
Fend
```

[功能]

用於返回通訊端口接收緩存器內的位元組數

[語法]

ChkNet(通訊端口編號)

[參數說明]

通訊端口編號　　指定需要修改通訊參數的端口編號。範圍：201~216

注意事項：

1. 以整數返回位元組數。如果數據不存在，以負值返回通訊端口狀態。-1 為端口已開啟，但沒有通訊連接。-2 為其他線程正在使用該通訊端口。-3 為通訊端口未開啟。

[使用案例]

```
Function main
    ReConnect:
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    If ChkNet(201) =  -1 Then
        Print "視覺 TCP 通訊錯誤，端口已打開，但是未建立通訊"
        Wait 1
        GoTo ReConnect
    Elself ChkNet(201) = -2 Then
        Print "視覺 TCP 通訊錯誤，其他任務正在使用端口"
        Wait 1
        GoTo ReConnect
    Elself ChkNet(201) = -3 Then
        Print "視覺 TCP 通訊錯誤，未打開端口"
        Wait 1
        GoTo ReConnect
    EndIf
Fend
```

[功能]

用於關閉 OpenNet 打開的 TCP/IP 通訊端口

[語法]

CloseNet [#通訊端口編號/All]

[參數說明]

#通訊端口編號　　指定需要修改通訊參數的端口編號。範圍：#201~216。All 為關閉所有通訊端口。

[使用案例]

```
Function main
    ReConnect:
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    If ChkNet(201) =  -1 Then
        Print "視覺 TCP 通訊錯誤，端口已打開，但是未建立通訊"
        Wait 1
        GoTo ReConnect
    Elself ChkNet(201) = -2 Then
        Print "視覺 TCP 通訊錯誤，其他任務正在使用端口"
        Wait 1
        GoTo ReConnect
    Elself ChkNet(201) = -3 Then
        Print "視覺 TCP 通訊錯誤，未打開端口"
        Wait 1
        GoTo ReConnect
    EndIf
Fend
```

[功能]

用於設置 RS232/485 串口通訊參數

[語法]

SetCom #通訊端口編號{, 串列傳輸速率, 數據位長度, 結束位長度, 奇偶性, 結束符, H/W 流控制, S/W 流控制, 超時時間}

[參數說明]

#通訊端口編號 指定需要修改通訊參數的端口編號。範圍：#1~16

串列傳輸速率 可指定 4800、9600、14400、19200、38400、56000、57600、115200、128000、230400、256000。省略時默認 19200。

數據位長度 以 7 或 8 指定每個字元的數據位長度。可省略。

結束位長度 以 1 或 2 指定每個字元的結束位長度。可省略。

結束符 通訊數據的末尾字元。可以設定 CR、LF、CRLF(回車、換行、回車換行)。可省略

H/W 流控制 硬體控制有效時指定 RTS，無效時指定 NONE。可省略。

S/W 流控制 軟體控制有效時指定 XON，無效時指定 NONE。可省略。

超時時間 以秒設定收發的最長時間。0 為永不超時。可省略

[使用案例]

```
Function main
    Integer AA
    ReConnect:
    SetCom #2,38400,8,2,N,CRLF,NONE,NONE,0
    OpenCom #2
    Do
        Print #2 ,"OK"
        Input #2,AA
        Print AA
    Loop

Fend
```

[功能]

用於開啟 RS232/485 串口通訊，或獲取 RS232/485 串口的線程編號。

[語法]

OpenCom #通訊端口編號

OpenCom(通訊端口編號)

[參數說明]

#通訊端口編號　　指定需要修改通訊參數的端口編號。範圍：#1~16

[使用案例]

```
Function main
    Integer AA
    ReConnect:
    SetCom #2,38400,8,2,N,CRLF,NONE,NONE,0
    OpenCom #2
    Do
        Print #2 ,"OK"
        Input #2,AA
        Print AA
    Loop

Fend
```

[功能]

用與返回串口通訊端口接收緩衝器內的位元組數。

[語法]

ChkCom(通訊端口編號)

[參數說明]

通訊端口編號　　指定需要修改通訊參數的端口編號。範圍：1~16

注意事項：

1. 以整數返回位元組數。如果數據不存在，以負值返回通訊端口狀態。-2 為其他線程正在使用該通訊端口。-3 為通訊端口未開啟。

[使用案例]

```
Function main
    Integer numChars
    numChars = ChkCom(1)
Fend
```

[功能]

用與返回串口通訊端口接收緩衝器內的位元組數。

[語法]

ClrCom #通訊端口編號

[參數說明]

#通訊端口編號　　指定需要修改通訊參數的端口編號。範圍：#1~16

[使用案例]

```
Function main
    ClrCom #1
Fend
```

[功能]

用與關閉 OpenCom 打開的串口通訊端口。

[語法]

```
CloseCom [#通訊端口編號/All]
```

[參數說明]

#通訊端口編號　　指定需要修改通訊參數的端口編號。範圍：#1~16。All 為關閉所有。

[使用案例]

```
Function main
    CloseCom #1
Fend
```

[功能]

用於從通訊端口讀取指定的字元數，不包含結束符號。

[語法]

```
Read #通訊端口編號, 字串變數$, 字元數
```

[參數說明]

#通訊端口編號　　指定需要修改通訊參數的端口編號。TCP/IP 範圍：#201~216。串口範圍：#1~16。

字串變數$　　用於接收端口內字元的字串變數。

字元數　　讀取字元的數量

[使用案例]

```
Function main
    String strBin$
    Read #201,strBin$,4          '從#201 端口讀取四個字元
Fend
```

[功能]

用於從 TCP/IP 通訊端口讀取二進位數據

[語法]

ReadBin #通訊端口編號, 變數名

ReadBin #通訊端口編號, 數組變數名(), 位元組數

[參數說明]

#通訊端口編號 指定需要修改通訊參數的端口編號。TCP/IP 範圍：#201~216。串口範圍：#1~16。

變數名 用於接收端口內二進位數據的變數。

數組變數名() 用於接收端口內二進位數據的數組變數。

位元組數 讀取位元組的數量

[使用案例]

```
Function main
    Integer num
    ReadBin #201,num
    Integer arr(5)
    ReadBin #201,arr(),5
Fend
```

[功能]

將字串寫入到通訊端口中，不包含結束符號。

[語法]

Write #通訊端口編號, 字串

[參數說明]

#通訊端口編號 指定需要修改通訊參數的端口編號。TCP/IP 範圍：#201~216。串口範圍：#1~16。

字串 需要寫入通訊端口的字串

[使用案例]

```
Function main
    Write #201 , "abcd"
Fend
```

[功能]

將二進位數據寫入通訊端口，不包含結束符號。

[語法]

WriteBin #通訊端口編號, 數據

WriteBin #通訊端口編號, 數組變數名(), 位元組數

[參數說明]

#通訊端口編號 指定需要修改通訊參數的端口編號。TCP/IP 範圍：#201~216。串口範圍：#1~16。

變數名 用於接收端口內二進位數據的變數。

數組變數名() 用於接收端口內二進位數據的數組變數。

位元組數 讀取位元組的數量

[使用案例]

```
Function main
    WriteBin #201 , 123456
    Integer arr(5)
    arr(0) = 22
    arr(1) = 43
    arr(2) = 67
    arr(3) = 45
    arr(4) = 33
    WriteBin #201 , arr() , 5
Fend
```

[功能]

將數據或字串輸出到指定通訊端口中，包含結束符號。

[語法]

```
Print #通訊端口編號, 數據 1{, 數據 2...}
```

[參數說明]

#通訊端口編號　　指定需要修改通訊參數的端口編號。TCP/IP 範圍：#201~216。串口範圍：#1~16。

數據　　可用變數、數值、字串等來輸出想輸出的內容。可根據需求酌情增加數據量。

[使用案例]

```
Function main
    String CCD_X$,CCD_Y$,CCD_U$
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    Print #201 , "EXW,1"
    Input #201 , CCD_X$,CCD_Y$,CCD_U$
    Print CCD_X$,CCD_Y$,CCD_U$
Fend
```

[功能]

用於從通訊端口讀取數據或字串，包含結束符號。

[語法]

Input #通訊端口編號, 變數 1[, 變數 2...]

[參數說明]

#通訊端口編號 指定需要修改通訊參數的端口編號。TCP/IP 範圍：#201~216。串口範圍：#1~16。

變數 用於接收數據或字串的變數名。可根據數據長度酌情增加變數。

注意事項：

1. Input #默認字串的分割為“,”(逗號)或“ ”(空白)。其餘分隔符號請使用令牌字串指令。

[使用案例]

```
Function main
    String CCD_X$,CCD_Y$,CCD_U$
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    Print #201 , "EXW,1"
    Input #201 , CCD_X$,CCD_Y$,CCD_U$
    Print CCD_X$,CCD_Y$,CCD_U$
Fend
```

[功能]

用於從通訊端口讀取一行數據或字串，包含結束符號。

[語法]

```
Line Input #通訊端口編號, 變數
```

[參數說明]

#通訊端口編號　　指定需要修改通訊參數的端口編號。TCP/IP 範圍：#201~216。串口範圍：#1~16。

變數　　用於接收數據或字串的變數名

[使用案例]

```
Function main
    String CCD_TxT$
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    Print #201 , "EXW,1"
    Line Input #201 , CCD_TxT$
    Print CCD_TxT$
Fend
```

系統指令詳細說明

[功能]

列印指令，列印字串、數值、變數

[語法]

Print 內容 1{, 內容 2.....}

[參數說明]

內容	想要列印的字串、數值或變數。可根據需求酌情增加變數。

[使用案例]

```
Function main
    ReConnect:
    CloseNet #201
    SetNet #201 , "192.168.1.220" , 9000 ,CRLF
    OpenNet #201 As Client
    WaitNet #201, 5
    If ChkNet(201) =  -1 Then
        Print "視覺 TCP 通訊錯誤，端口已打開，但是未建立通訊"
        Wait 1
        GoTo ReConnect
    ElseIf ChkNet(201) = -2 Then
        Print "視覺 TCP 通訊錯誤，其他任務正在使用端口"
        Wait 1
        GoTo ReConnect
    ElseIf ChkNet(201) = -3 Then
        Print "視覺 TCP 通訊錯誤，未打開端口"
        Wait 1
        GoTo ReConnect
    EndIf

    Print ChkNet(201)

Fend
```

[功能]

啟動多線程

[語法]

Xqt {任務編號, } 函數名(形參 1...) {, Normal/NoPause/NoEmgAbort}

[參數說明]

任務編號	以整數指定多線程的編號。可省略。
函數名	要執行多線程的 Function 名。如果 Function 有形參則需要填上()並補上形參，如沒有則直接填函數名即可。
Normal	普通的多線程。三者省略不填，默認 Normal。
NoPause	發生 Pause、或有外部暫停信號、以及安全門打開的狀態下不會暫停的多線程。
NoEmgAbort	緊急停止、或發生錯誤時仍能繼續處理的多線程。

注意事項:

1. 前臺多線程，任務編號 1~16。後臺多線程，任務編號 66~80。

[使用案例]

```
Function main
    Xqt A1
Fend

Function A1
    Print "Runing....."
Fend
```

[功能]

用於暫停可暫停的所有多線程。

[語法]

Pause

[參數說明]

注意事項:

1. 前臺多線程，任務編號 1~16。後臺多線程，任務編號 66~80。

[使用案例]

```
Function main
    Xqt A1
    Wait 1
    Pauser                '暫停程式
Fend

Function A1
    Print "Runing....."
Fend
```

[功能]

用於暫停指定的正在執行的多線程

[語法]

Halt [函數名/任務編號]

[參數說明]

函數名　　正在執行的多線程 Function 名。

任務編號　　正在執行的多線程任務編號。

注意事項：

1. 前臺多線程，任務編號 1~16。後臺多線程，任務編號 66~80。

[使用案例]

```
Function main
    Xqt A1
    Wait 1
    Halt A1          '暫停 A1 線程
Fend

Function A1
    Do
        Print "Runing....."
    Loop
Fend
```

[功能]

停止所有或指定的線程

[語法]

Quit [函數名/任務編號/All]

[參數說明]

函數名　　正在執行的多線程 Function 名。

任務編號　　正在執行的多線程任務編號。

All　　停止所有線程。

注意事項：

1. 前臺多線程，任務編號 1~16。後臺多線程，任務編號 66~80。

[使用案例]

```
Function main
    Xqt A1
    Wait 1
    Quit A1          '停止 A1 線程
Fend

Function A1
    Do
        Print "Runing....."
    Loop
Fend
```

[功能]

用於啟用主任務。

[語法]

StartMain [主任務名稱]

[參數說明]

主任務名稱　　main - main63

[使用案例]

```
Function bgmain                '後臺任務
    If MemIn W(0) = 1 Then
        StartMain main
        Wait MemIn W(0) <> 1      '等待信號消失，防止重複觸發
    EndIf
Fend
```

[功能]

用於繼續運行被 Halt 命令而暫停的多線。

[語法]

Resume [函數名/任務編號/All]

[參數說明]

函數名　　正在執行的多線程 Function 名。

任務編號　　正在執行的多線程任務編號。

All　　繼續所有線程。

注意事項：

1. 前臺多線程，任務編號 1~16。後臺多線程，任務編號 66~80。

[使用案例]

```
Function main
    Xqt A1
    Wait 1
    Halt A1          '暫停 A1 線程
    Wait Sw(1) = 1
    Resume A1        '繼續運行 A1 線程
Fend

Function A1
    Do
        Print "Runing....."
    Loop
Fend
```

[功能]

用於重置控制器異警狀態。

[語法]

```
Resume [Error]
```

[參數說明]

Error　　指所有錯誤將被指令清除。

注意事項：

1. 調用指令時填寫 Error 則清除所有報警，若未填寫 Error 則只清除急停狀態。

[使用案例]

```
Function main
    Resume          '清除急停狀態
Fend

Function A1
    Resume Error    '清除異警狀態
Fend
```

[功能]

用於返回當前線程的任務編號

[語法]

MyTask

[參數說明]

注意事項：

1. 前臺多線程，任務編號 1~16。後臺多線程，任務編號 66~80。

[使用案例]

```
Function main
    Xqt 2 , A1
    Xqt 3 , A1
    Xqt 4 , A1
Fend

Function A1
    On MyTask       '將編號與當前任務編號相同的 IO 端口設為 On
    Wait 1
    Off MyTask      '將編號與當前任務編號相同的 IO 端口設為 Off
Fend
```

[功能]

確認線程是否結束

[語法]

TaskDone(函數名/任務編號)

[參數說明]

函數名　　多線程 Function 名。

任務編號　　多線程任務編號。

注意事項:

1. 前臺多線程，任務編號 1~16。後臺多線程，任務編號 66~80。
2. 線程結束返回 True。否則返回 False。

[使用案例]

```
Function main
    Xqt A1
    Do
        Print "A1"
    Loop Until TaskDone(A1)
Fend

Function A1
    Go P1
Fend
```

[功能]

返回線程當前的狀態

[語法]

TaskState(函數名/任務編號)

[參數說明]

函數名　　多線程 Function 名。

任務編號　　多線程任務編號。

注意事項:

1. 前臺多線程，任務編號 1~16。後臺多線程，任務編號 66~80。

2. 0 為未執行。4 為待機。5 為啟動中。6 為運行中。7 為錯誤狀態。8 為暫停狀態。9 為恢復中狀態。10 為停止中狀態。

[使用案例]

```
Function main
    If TaskState(A1) = 0 Then
        Xqt 2 , A1
    EndIf
Fend

Function A1
    Go P1
Fend
```

[功能]

用於等待指定線程結束

[語法]

TaskWait(函數名/任務編號)

[參數說明]

函數名　　多線程 Function 名。

任務編號　　多線程任務編號。

注意事項:

1. 前臺多線程，任務編號 1~16。後臺多線程，任務編號 66~80。

[使用案例]

```
Function main
    Xqt 2 , A1
    TaskWait A1
Fend

Function A1
    Go P1
Fend
```

[功能]

用於使用相互排他鎖定，使多個線程同步

[語法]

SyncLock 信號編號[, 超時時間]

[參數說明]

信號編號　　以運算式或數值指定要接收的信號編號。範圍 0~63。

超時時間　　以運算式或數值指定鎖定之前等待的時間。可省略。

[使用案例]

```
'下例所示為使用 SyncLock 和 SyncUnlock 設為 1 次只有 1 個任務將資訊寫入到記錄檔中。
Function Main
    Xqt Func1
    Xqt Func2
Fend
Function Func1
    Long count
    Do
    Wait 0.5
        count = count+1
        LogMsg("Msg from Func1,"+ Str$(count))
    Loop
Fend

Function Func2
    Long count
    Do
    Wait 0.5
        count = count+1
        LogMsg("Msg from FuncmsgSCloseCom2,"+ Str$(count))
    Loop
Fend

Function LogMsg(msg$ As String)
    SyncLock 1
    Print msg$
    Wait 1
    SyncUnlock 1
Fend
```

[功能]

用於解除 SyncLock 鎖定的信號編號。

[語法]

SyncUnLock 信號編號

[參數說明]

信號編號　　以運算式或數值指定要接收的信號編號。範圍 0~63。

[使用案例]

```
'下例所示為使用 SyncLock 和 SyncUnlock 設為 1 次只有 1 個任務將資訊寫入到記錄檔中。
Function Main
    Xqt Func1
    Xqt Func2
Fend
Function Func1
    Long count
    Do
    Wait 0.5
        count = count+1
        LogMsg("Msg from Func1,"+ Str$(count))
    Loop
Fend

Function Func2
    Long count
    Do
    Wait 0.5
        count = count+1
        LogMsg("Msg from FuncmsgSCloseCom2,"+ Str$(count))
    Loop
Fend

Function LogMsg(msg$ As String)
    SyncLock 1
    Print msg$
    Wait 1
    SyncUnlock 1
Fend
```

[功能]

用於向正在執行 WaitSig 命令的任務發送信號

[語法]

```
Signal 信號編號
```

[參數說明]

信號編號	以運算式或數值指定要接收的信號編號。範圍 0~63。

[使用案例]

```
Function main
    Xqt 2 , A1
    Signal 1
Fend

Function A1
    WaitSig 1
Fend
```

[功能]

用於等待其他任務的 Signal 命令發送的同步信號

[語法]

WaitSig 信號編號{, 超時時間}

[參數說明]

信號編號　　以運算式或數值指定要接收的信號編號。範圍 0~63。

超時時間　　以運算式或數值指定最長等待的時間。可省略。

[使用案例]

```
Function main
    Xqt 2 , A1
    Signal 1
Fend

Function A1
    WaitSig 1
Fend
```

[功能]

用於返回 Wait、WaitNet、WaitSig 指令的狀態

[語法]

TW

[參數說明]

注意事項:

1. 在指定時間內 Wait、WaitNet、WaitSig 條件成立範圍 False。超時返回 True。

[使用案例]

```
Function main
    Wait Sw(0) = On , 5
    If TW = True Then
        '輸入 DIO 變為 ON 狀態之前待機 5 秒
        Print "DIO 並沒有置為 On"
    EndIf
Fend
```

[功能]

以秒為單位返回計算節拍時間計時器開始計時之後經過的時間，不包含程式暫停時間。

[語法]

ElapsedTime

[參數說明]

注意事項：

1. 計時器計量範圍 0~1.7E+31。最小單位是 0.001 秒。

[使用案例]

```
Function Main
    Integer i
    ResetElapsedTime                    '重置計算節拍時間用的計時器
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "次数：",i,ElapsedTime / 10    '計算並列印節拍時間
    If i < 10 Then
        Return
    EndIf
Fend
```

[功能]

用於重置由 ElapsedTime 使用的計算節拍時間用的計時器。

[語法]

ResetElapsedTime

[參數說明]

[使用案例]

```
Function Main
    Integer i
    ResetElapsedTime                    '重置計算節拍時間用的計時器
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "次数: ",i,ElapsedTime / 10    '計算並列印節拍時間
    If i < 10 Then
        Return
    EndIf
Fend
```

[功能]

以秒為單位返回計時器開始計時之後經過的時間，包含程式暫停時間。

[語法]

Tmr(計時器編號)

[參數說明]

計時器編號　　以運算式或數值指定返回那個計時器的時間。範圍 0~63。

[使用案例]

```
Function Main
    Integer i
    TmReset 0                          '重置計算節拍時間用的計時器
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "次數: ",i,Tmr(0) / 10      '計算並列印節拍時間
    If i < 10 Then
        Return
    EndIf
Fend
```

[功能]

用於重置 Tmr 計時器

[語法]

TmReset 計時器編號

[參數說明]

計時器編號　　以運算式或數值指定返回那個計時器的時間。範圍 0~63。

[使用案例]

```
Function Main
    Integer i
    TmReset 0                          '重置計算節拍時間用的計時器
    For i = 1 To 10
        GoSub Cycle
        Wait 0.1
    Next
    Cycle:
    Print "次數: ",i,Tmr(0) / 10     '計算並列印節拍時間
    If i < 10 Then
        Return
    EndIf
Fend
```

[功能]

用於執行命令窗口的語句。

[語法]

```
Eval 命令
```

[參數說明]

命令　　以字串指定要執行的命令。
返回值　　返回通過執行命令返回的錯誤代碼。 當命令正常結束時，返回 0。

[使用案例]

```
Function Main

Integer errCMD
String CMD$
OpenCom #1

Do
    Line Input #1, CMD$
    errCMD = EVal(CMD$)
    Print #1, errCMD
Loop

Fend
```

碼垛語句詳細說明

[功能]

用於定義/顯示託盤

[語法]

定義託盤:

Pallet {OutSide,} 託盤編號, 點位 1, 點位 2, 點位 3{, 點位 4,} 個數 1, 個數 2

列印單個託盤:

Pallet 託盤編號

列印所有託盤:

Pallet

獲取託盤點位:

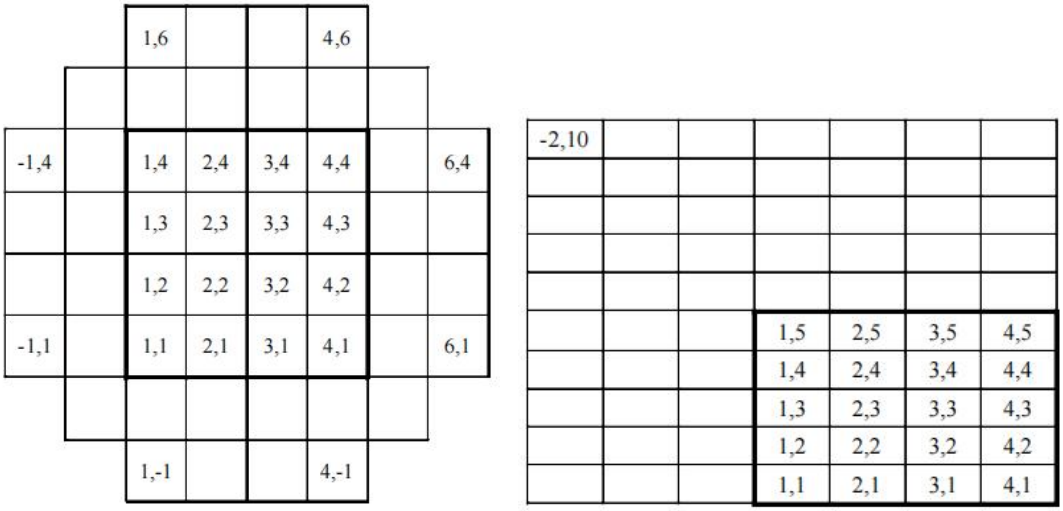
Pallet(託盤編號, 託盤位置編號)
Pallet(託盤編號, 行座標, 列座標)

[參數說明]

OutSide	在指定的行和列的範圍以外生成額外的託盤。可省略。
託盤編號	以運算式或數值指定託盤的編號。範圍 0~15。
點位 1~3	標準 3 點法定義託盤。
點位 4	可使用 4 點法定義託盤進一步提升託盤點位的精度。
個數 1	以整數指定點位 1 與點位 2 之間有多少個點位。範圍 1~32767。
個數 2	以整數指定點位 1 與點位 3 之間有多少個點位。範圍 1~32767。
行座標	以整數指定託盤的第幾行
列座標	以整數指定託盤的第幾列

注意事項：

1. OutSide 的效果如下圖：在原有託盤的基礎上拓展託盤外的點位。



[使用案例]

```
Function main
    Integer var

    '定義 5*3 的託盤
    Pallet 1 , P1 , P2 , P3 , 5 , 3

    '列印 2 號託盤
    Pallet 2

    '列印所有託盤
    Pallet

    'for 迴圈遍曆託盤點
    For var = 1 To 15
        Go Pallet(1 , var)
    Next

    Go Pallet(1, 2, 3) '移動到 1 號託盤第二行第三列的位置
Fend
```

[功能]

用於清除已設置的 Pallet

[語法]

```
PalletClr 託盤編號
```

[參數說明]

託盤編號	以運算式或數值指定返回那個計時器的時間。範圍 0~15。

[使用案例]

```
Function main
    Integer var

    '定義 5*3 的託盤
    Pallet 1 , P1 , P2 , P3 , 5 , 3

    '清除 1 號託盤
    PalletClr 1

Fend
```

傳送帶跟蹤語句詳細說明

[功能]

開始傳送帶跟蹤功能

[語法]

SyStart 傳送帶編號

[參數說明]

傳送帶編號　　以運算式或數值指定傳動帶編號。範圍 1~4。

注意事項:

1. 當機器人需要開始傳送帶跟蹤跟蹤物料時使用該指令。

[使用案例]

```
Recrt:
    SyStart Cvt_One '開啟皮帶跟蹤

If SyGetPose(Cvt_One) > 700 Then
    Print "物料不夠加工距離"
    SyEnd Cvt_One '關閉皮帶跟蹤
    GoTo Recry
EndIf

SyStart Cvt_One    '開啟皮帶跟蹤進行生產流程
.
.
.
.
SyEnd Cvt_One      '結束皮帶跟蹤
```

[功能]

結束傳送帶跟蹤功能

[語法]

SyEnd 傳送帶編號

[參數說明]

傳送帶編號　　以運算式或數值指定傳動帶編號。範圍 1~4。

注意事項:

1. 當機器人傳送帶跟蹤完畢，不需要再跟蹤時使用該指令。

[使用案例]

```
Recrt:
    SyStart Cvt_One '開啟皮帶跟蹤

If SyGetPose(Cvt_One) > 700 Then
    Print "物料不夠加工距離"
    SyEnd Cvt_One '關閉皮帶跟蹤
    GoTo Recry
EndIf

SyStart Cvt_One    '開啟皮帶跟蹤進行生產流程
.
.
.
.
SyEnd Cvt_One      '結束皮帶跟蹤
```

[功能]

清除當前傳送帶跟蹤目標佇列，複位傳送帶跟蹤

[語法]

SyReset 傳送帶編號

[參數說明]

傳送帶編號　　以運算式或數值指定傳動帶編號。範圍 1~4。

[使用案例]

```
SyReset Cvt_One        '初始化皮帶跟蹤
Recrt:
    SyStart Cvt_One '開啟皮帶跟蹤

If SyGetPose(Cvt_One) > 700 Then
    Print "物料不夠加工距離"
    SyEnd Cvt_One '關閉皮帶跟蹤
    GoTo Recry
EndIf

SyStart Cvt_One    '開啟皮帶跟蹤進行生產流程
.
.
.
.
SyEnd Cvt_One      '結束皮帶跟蹤
```

[功能]

獲取指定傳送帶目標佇列中的點位資訊

[語法]

SyGetPoint(傳送帶編號)

[參數說明]

傳送帶編號　　以運算式或數值指定傳動帶編號。範圍 1~4。

[使用案例]

```
SyMove SyGetPoint(Cvt_One):X(X1) :Y(Y1) :Z(Z1) :U(RZ1) CP

SyMove SyGetPoint(Cvt_One):X(X1) :Y(Y1) :Z(Z1) :U(RZ1)

On 0
Wait 1

Off 0
Wait 1
```

[功能]

獲取指定傳送帶目標佇列中點位的附加資訊。

[語法]

SyGetUserData(傳送帶編號)

[參數說明]

傳送帶編號　　以運算式或數值指定傳動帶編號。範圍 1~4。

注意事項:

1. 該資訊由 SyAddVisTarget 指令添加。

[使用案例]

```
Print SyGetUserData(1)
```

[功能]

獲取指定傳送帶目標當前在傳送帶上的位置。單位 mm。

[語法]

SyGetPose(傳送帶編號)

[參數說明]

傳送帶編號　　以運算式或數值指定傳動帶編號。範圍 1~4。

[使用案例]

```
Recrt:
    SyStart Cvt_One '開啟皮帶跟蹤

If SyGetPose(Cvt_One) > 700 Then
    Print "物料不夠加工距離"
    SyEnd Cvt_One '關閉皮帶跟蹤
    GoTo Recry
EndIf

SyStart Cvt_One    '開啟皮帶跟蹤進行生產流程
.
.
.
.
SyEnd Cvt_One      '結束皮帶跟蹤
```

[功能]

獲取指定傳送帶的佇列中目標的個數

[語法]

SyGetNum(傳送帶編號)

[參數說明]

傳送帶編號　　以運算式或數值指定傳動帶編號。範圍 1~4。

[使用案例]

```
If SyGetNum(1)  > 1 Then
    Print "跟蹤佇列中存在目標"
EndIf
```

[功能]

傳送帶跟蹤直線插補運動

[語法]

SyMove 目標座標 {CP} {Till/Find} {!...!}

[參數說明]

目標座標	以傳送帶跟蹤指令獲取的點數據指定目標位置
CP	設置運動平滑。可省略
Till/Find	Till 運算式 = On/Off。Find 運算式 = On/Off。可省略。詳情請參考 Till/Find 的詳細說明
!...!	在運動過程中並行處理 I/O 等輸入輸出。可省略。詳情請參考並行處理的詳細說明。

[使用案例]

```
SyMove SyGetPoint(Cvt_One):X(X1) :Y(Y1) :Z(Z1) :U(RZ1) CP

SyMove SyGetPoint(Cvt_One):X(X1) :Y(Y1) :Z(Z1) :U(RZ1)

On 0
Wait 1

Off 0
Wait 1
```

[功能]

傳送帶跟蹤圓弧插補運動

[語法]

SyArc 經過座標, 目標座標 {CP}{Till/Find}{!...!}

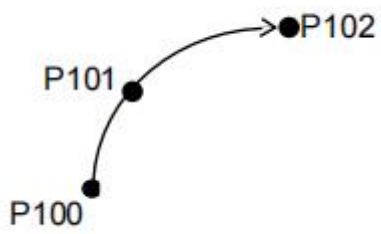
[參數說明]

經過座標	以傳送帶跟蹤指令獲取的點數據指定圓弧會經過的位置
目標座標	以傳送帶跟蹤指令獲取的點數據指定目標位置
CP	設置運動平滑。可省略
Till/Find	Till 運算式 = On/Off。Find 運算式 = On/Off。可省略。詳情請參考 Till/Find 的詳細說明
!...!	在運動過程中並行處理 I/O 等輸入輸出。可省略。詳情請參考並行處理的詳細說明。

[使用案例]

```
On 0
Wait 1
SyArc SyGetPoint(Cvt_One):X(X1) :Y(Y1) :Z(Z1) :U(RZ1)

On 0
Wait 1
```



[功能]

記錄當前傳動帶的編碼器脈衝值

[語法]

SyTrig 傳送帶編號

[參數說明]

傳送帶編號　　以運算式或數值指定傳動帶編號。範圍 1~4。

注意事項:

1. 後續的 SyAddVisTarget 將會使用此指令記錄的編碼器脈衝值。

[使用案例]

```
SyTrig 1
```

[功能]

將普通座標或點位轉換為傳送帶跟蹤座標或點位。

[語法]

SyPoint(傳送帶編號, 座標 X, 座標 Y, 座標 U)
SyPoint(傳送帶編號, 點位)

[參數說明]

傳送帶編號	以運算式或數值指定傳動帶編號。範圍 1~4。
座標	以 mm 為單位，指定需要轉換為傳送帶跟蹤的座標。
點位	以運算式、點名稱、點編號指定需要轉換為傳送帶跟蹤的座標。

[使用案例]

```
SyAddVisTarget 1 , SyPoint(1 , 10 , 20 , 30)
SyMove SyPoint(1 , 5 , 5 , 5)
```

[功能]

將視覺拍照結果註冊進傳送帶跟蹤佇列中

[語法]

SyAddVisTarget 傳送帶編號, 點位[, 附加資訊]

[參數說明]

傳送帶編號	以運算式或數值指定傳動帶編號。範圍 1~4。
點位	以運算式、點名稱、點編號指定需要註冊的點位
附加資訊	以整數指定附加資訊

[使用案例]

```
Do
    Input #TCP_Id , temp1$(0) , temp1$(1) , temp1$(2) , temp1$(3)
    Print temp1$(0) ,",", temp1$(1) ,",", temp1$(2) ,",", temp1$(3)
    CCD_X$ = temp1$(1)
    CCD_Y$ = temp1$(2)
    CCD_U$ = temp1$(3)
    If Val(temp1$(0)) = 1 Then
        Print "註冊點位"
        SyAddVisTarget Cvt_One , SyPoint(Cvt_One , Val(CCD_X$) , Val(CCD_Y$) , Val(CCD_U$)
    EndIf
Loop Until ChkNet(TCP_Id) < 0
```

[功能]

設置/獲取傳送帶跟蹤功能濾重間距

[語法]

SyRejectDis 傳送帶編號 , 距離
SyRejectDis (傳送帶編號)

[參數說明]

傳送帶編號 以運算式或數值指定傳動帶編號。範圍 1~4。

距離 以毫米為單位區分物料間距

注意事項:

1. 請勿在跟蹤過程中使用該函數。

[使用案例]

```
Recrt:
    SyRejectDis Cvt_One , 20
    SySave

    SyRejectDis Cvt_One

    SyStart Cvt_One '開啟皮帶跟蹤

If SyGetPose(Cvt_One) > 700 Then
    Print "物料不夠加工距離"
    SyEnd Cvt_One '關閉皮帶跟蹤
    GoTo Recry
EndIf

SyStart Cvt_One    '開啟皮帶跟蹤進行生產流程
.
.
.
.
SyEnd Cvt_One      '結束皮帶跟蹤
```

[功能]

保存程式中對傳送帶跟蹤功能的修改。

[語法]

SySave

[參數說明]

注意事項:

1. 所有在程式中對傳送帶跟蹤功能的參數修改都需要使用該指令進行保存。

[使用案例]

```
Recrt:
    SyRejectDis Cvt_One , 20
    SySave

    SyRejectDis Cvt_One

    SyStart Cvt_One '開啟皮帶跟蹤

If SyGetPose(Cvt_One) > 700 Then
    Print "物料不夠加工距離"
    SyEnd Cvt_One '關閉皮帶跟蹤
    GoTo Recry
EndIf

SyStart Cvt_One    '開啟皮帶跟蹤進行生產流程
.
.
.
.
SyEnd Cvt_One      '結束皮帶跟蹤
```

數學運算指令詳細說明

[功能]

加、減、乘、除、乘方、取模、與、或、異或、非、大於、大於等於、小於、小於等於、不等於、等於

[語法]

運算符	格式示例	說明
+	A+B	加法
-	A-B	減法
*	A*B	乘法
/	A/B	除法
**	A**B	乘方
=	A=B	A 等於 B
>	A>B	A 大於 B
<	A<B	A 小於 B
>=	A>=B	A 大於等於 B
<=	A<=B	A 小於等於 B
<>	A<>B	A 不等於 B
And	A And B	邏輯與
Mod	A Mod B	整數的取模
Not	Not A	非
Or	A Or B	邏輯域
Xor	A Xor B	邏輯異域

[參數說明]

[使用案例]

[功能]

角度轉弧度

[語法]

DegToRad(角度)

[參數說明]

角度	以數值指定需要轉換為弧度的角度。單位 deg。

[使用案例]

```
Function main
    Double rad_num , cos_num
    rad_num =  DegToRad(30)
    cos_num =  Cos(rad_num)
Fend
```

[功能]

弧度轉角度

[語法]

```
RadToDeg(弧度)
```

[參數說明]

弧度	以數值指定需要轉換為角度的弧度。

[使用案例]

```
Function main
    Double deg_num
    deg_num =  RadToDeg(1.21)
Fend
```

[功能]

正弦函数、余弦函数、正切函数、反正弦函数、反余弦函数、反正切函数、反正切函数 2

[語法]

```
Sin(弧度)
Cos(弧度)
TaVn(弧度)
Asin(數值)
Acos(數值)
Atan(數值)
Atan2(y, x)
```

[參數說明]

弧度	以整數或浮點數指定進行運算的弧度。
數值	以整數或浮點數指定進行運算的數值。

[使用案例]

```
Function main
    Double num

    num = Sin(DegToRad(30))
    num = Cos(DegToRad(30))
    num = Tan(DegToRad(30))

    num = Asin(0.5)
    num = Acos(0.5)
    num = Atan(0.5)
    num = Atan2(1,2)
Fend
```

[功能]

十六進制轉單精確度浮點數。符合 IEEE754。

[語法]

HexToFloat(數值)

[參數說明]

數值 可以十進位整數或十六進制數(&H 數值)指定

[使用案例]

```
Function main
    Print HexToFloat(&H40490FDA)          '輸出 3.1415926
Fend
```

[功能]

單精確度浮點數轉十六進制數。符合 IEEE754。

[語法]

FloatToHex(數值)

[參數說明]

數值 可以十進位整數或單精確度浮點數指定

[使用案例]

```
Function main
    Print Hex$(FloatToHex(3.1415926))          '輸出 40490FDA
Fend
```

[功能]

取絕對值

[語法]

Abs(數值)

[參數說明]

數值	可以十進位整數或浮點數指定

[使用案例]

```
Function main
    Abs(-1.234)
Fend
```

[功能]

取平方根

[語法]

Sqr(數值)

[參數說明]

數值	可以十進位整數或浮點數指定

[使用案例]

```
Function main
    Sqr(4)
Fend
```

[功能]

返回數值的符號

[語法]

Sgn(數值)

[參數說明]

數值	可以十進位整數或浮點數指定

[使用案例]

```
Function main
    Sgn(4)
Fend
```

[功能]

左移

[語法]

LShift(數值, 位數)

[參數說明]

數值	可以十進位整數或浮點數指定
位數	以整數指定左移幾位

[使用案例]

```
Function main
    LShift(4433 , 3)
Fend
```

[功能]

右移

[語法]

RShift(數值, 位數)

[參數說明]

數值	可以十進位整數或浮點數指定
位數	以整數指定左移幾位

[使用案例]

```
Function main
    RShift(4433 , 3)
Fend
```

[功能]

將數值指定的 Bit 置 0 並返回該數值

[語法]

BClr(數值, 位數)

[參數說明]

數值	可以十進位整數或浮點數指定
位數	以整數指定第幾位

[使用案例]

```
Function main
    BClr(123 , 2)
Fend
```

[功能]

將數值指定的 Bit 置 1 並返回該數值

[語法]

BSet(數值, 位數)

[參數說明]

數值	可以十進位整數或浮點數指定
位數	以整數指定第幾位

[使用案例]

```
Function main
    BSet(123 , 2)
Fend
```

[功能]

返回數值指定 Bit 的值

[語法]

BTst(數值, 位數)

[參數說明]

數值	可以十進位整數或浮點數指定
位數	以整數指定第幾位

[使用案例]

```
Function main
    BTst(123 , 2)
Fend
```

[功能]

返回數組長度

[語法]

Ubound(數組名{, 維度})

[參數說明]

數組名	可以十進位整數或浮點數指定
維度	1 為一維。2 為二維。3 為三維。可省略，默認一維。

[使用案例]

```
Function main
    Integer i , a(10)
    For i = 0 To UBound(a)
        a(i) = i
    Next
Fend
```

字串運算指令詳細說明

[功能]

字串分割

[語法]

```
ParseStr 字串, 字串數組$(), 分割符$
字串數組長度 = ParseStr(字串, 字串數組$(), 分割符$)
```

[參數說明]

字串	需要用於分割的字串
字串數組$()	用於接收被分隔符號分割的字串
分割符$	字串根據該分隔符號進行分割
字串數組長度	字串分割後得到的數組的長度

[使用案例]

```
Function main
    String myStr$ , strArr$(0)
    Integer i
    myStr$ = "1$2$3$4"
    ParseStr myStr$ , strArr$() , "$"
    For i = 0 To UBound(strArr$())
        Print "數組內容 = " , strArr$(i)
    Next
Fend
```

[功能]

字串拼接

[語法]

字串 + 字串

[參數說明]

字串　　需要用於拼接的字串

[使用案例]

```
Function main
    String myStr$
    myStr$ = "Pro" + "Easy"
Fend
```

[功能]

不等於、等於、賦值

[語法]

字串 <> 字串

字串 = 字串

字串變數 = 字串

[參數說明]

字串　　需要用於運算的字串

[使用案例]

```
Function main
    String myStr$
    myStr$ = "stop"
    IF myStr$ = "Run" Then
        Print "Run"
    ElseIF myStr$ <> "Run" Then
        Print "stop"
    EndIF
Fend
```

[功能]

字串轉數值

[語法]

Val(字串)

[參數說明]

字串	需要用於轉換的字串

[使用案例]

```
Function main
    String myStr$
    myStr$ = "3.1415926"

    Double num
    num = Val(myStr$)
Fend
```

[功能]

數值轉字串

[語法]

Str$(數值)

[參數說明]

數值	需要用於轉換的數值

[使用案例]

```
Function main
    String myStr$
    Double num
    num    = Val(myStr$)
    myStr$ = Str$(num)
Fend
```

[功能]

獲取字串長度

[語法]

Len(字串)

[參數說明]

字串　　需要用於轉換的字串

[使用案例]

```
Function main
    String myStr$
    Print Len(myStr$)
Fend
```

[功能]

字串轉 ASCII 碼

[語法]

Asc(字串)

[參數說明]

字串　　需要用於轉換的字串

[使用案例]

```
Function Main
    Integer AA1 , BB1 , CC1
    AA1 = Asc("a")
    BB1 = Asc("b")
    CC1 = Asc("c")
    Print "The ASCII Value of AA1 is",AA1
    Print "The ASCII Value of BB1 is",BB1
    Print "The ASCII Value of CC1 is",CC1
Fend
```

[功能]

ASCII 碼轉字串

[語法]

Chr$(ASCII 碼)

[參數說明]

ASCII 碼	1~255 指定 ASCII 碼

[使用案例]

```
Function Main
    String Test$
    Test$ = Chr$(&H41) + Chr$(&H42) + Chr$(&H43)
    Print "The value of Test= " , Test$
Fend
```

[功能]

從字串左側開始提取指定的字串

[語法]

Left$(字串, 數量)

[參數說明]

字串	需要被提取的字串
數量	提取數量

[使用案例]

```
Function Main
    Print Left$("ABCDEFG" , 2)
    '>>> AB
    Print Left$("ABCDEFG" , 3)
    '>>> ABC
Fend
```

[功能]

從字串指定範圍開始提取指定的字串

[語法]

```
Mid$(字串, 起始位置{, 數量})
```

[參數說明]

字串	需要被提取的字串
起始位置	提取字串的起始位置
數量	提取數量。省略則從起始位置開始提取到結束。

[使用案例]

```
Function main
    Print Mid$("123456" , 3 , 3)
Fend
```

[功能]

從字串右側開始提取指定的字串

[語法]

```
Right$(字串, 數量)
```

[參數說明]

字串	需要被提取的字串
數量	提取數量

[使用案例]

```
Function Main
    Print Right$("ABCDEFG" , 2)
    '>>> FG
    Print Right$("ABC" , 3)
    '>>> ABC
Fend
```

[功能]

從字串左側開始提取指定長度的字串，字串長度不足時補充空格

[語法]

```
LSet$(字串, 數量)
```

[參數說明]

字串	需要被提取的字串
數量	提取數量

[使用案例]

```
Function Main
    String temp$
    temp$ = "123"
    temp$ = LSet$(temp$ , 10)        'temp$ = "123"
Fend
```

[功能]

從字串右側開始提取指定長度的字串，字串長度不足時補充空格

[語法]

```
RSet$(字串, 數量)
```

[參數說明]

字串	需要被提取的字串
數量	提取數量

[使用案例]

```
Function Main
    String temp$
    temp$ = "123"
    temp$ = RSet$(temp$ , 10)        'temp$ = "123"
Fend
```

[功能]

返回指定數量的空格字串

[語法]

Space$(數量)

[參數說明]

數量 空格數量

[使用案例]

```
Function Main
    Print "XYZ" + Space$(1) + "ABC"
    '>>>XYZ ABC
    Print Space$(3) + "ABC"
    '>>>ABC
Fend
```

[功能]

返回小寫字母的字串

[語法]

LCase$(字串)

[參數說明]

字串 需要用於轉換的字串

[使用案例]

```
Function Main
    String Test$
    Test$ = "Data"
    Test$ = LCase$(Test$)          'Test$ = "Data"
Fend
```

[功能]

返回大寫字母的字串

[語法]

```
UCase$(字串)
```

[參數說明]

字串　　需要用於轉換的字串

[使用案例]

```
Function Main
    String Test$
    Test$ = "Data"
    Test$ = UCase$(Test$)          'Test$ = "Data"
Fend
```

[功能]

用於刪除字串左側空格並返回

[語法]

```
LTrim$(字串)
```

[參數說明]

字串　　需要用於轉換的字串

[使用案例]

```
Function Main
    String Test$
    Test$ = "   Data   "
    Test$ = LTrim$(Test$)          'Test$ = "Data   "
Fend
```

[功能]

用於刪除字串右側空格並返回

[語法]

RTrim$(字串)

[參數說明]

字串　　需要用於轉換的字串

[使用案例]

```
Function Main
    String Test$
    Test$ = "   Data   "
    Test$ = RTrim$(Test$)          'Test$ = "   Data"
Fend
```

[功能]

用於刪除字串兩側空格並返回

[語法]

Trim$(字串)

[參數說明]

字串　　需要用於轉換的字串

[使用案例]

```
Function Main
    String Test$
    Test$ = "   Data   "
    Test$ = Trim$(Test$)           'Test$ = "Data"
Fend
```

[功能]

從字串中檢索指定字元的位置並返回

[語法]

InStr(字串, 檢索字串)

[參數說明]

字串 需要用於檢索的字串

檢索字串 需要檢索的字串

[使用案例]

```
Function main
    Print InStr("ABCDEF" , "C")
Fend
```

[功能]

用於返回指定數量跳位字元的字串

[語法]

Tab$(數量)

[參數說明]

數量 指定跳位字元數量

[使用案例]

```
Function Main
    Integer i
    Print "X" , Tab$(1) , "Y"
    For i = 1 To 10
        Print x(i) , Tab$(1) , y(i)
    Next i
Fend
```

[功能]

用於將 16 進制數轉成字串

[語法]

Hex$(數值)

[參數說明]

數值　　需要轉成字串的數值

[使用案例]

```
Function Main
    Integer stat
    stat = 10
    Print Hex$(stat)
    '>>>A
    Print  Hex$(255)
    '>>>FF
Fend
```

點位運算指令詳細說明

[功能]

相對座標增量運動

[語法]

運動指令 點位 + 分量

運動指令 點位 - 分量

[參數說明]

分量 方向分量

[使用案例]

```
Function main
    Go P1 + X(10)
    Move P2 - U(5)
Fend
```

[功能]

修改手系、修改用戶座標

[語法]

運動指令 /用戶坐標系編號

運動指令 /[R/L]

[參數說明]

用戶坐標系編號	指定切換的用戶坐標系編號。範圍 1~64
R/L	指定切換的手系，R 右手，L 左手

[使用案例]

```
Function main
    Go P1 + X(10) /R        '以右手系移動到 P1 點 X+5mm 處
    Go P1 + X(10) /3        '以用戶坐標系 3 移動到 P1 點 X+5mm 處
Fend
```

[功能]

絕對座標運動

[語法]

運動指令 點位 : 分量

[參數說明]

分量	方向分量

[使用案例]

```
Function main
    Go P1 :X(10)     '移動到 P1 點且 X=10mm 處
Fend
```

[功能]

賦值

[語法]

點位 = 指令

[參數說明]

[使用案例]

```
Function main
    P1 = XY(10 , 20 , 30 ,40)
Fend
```

[功能]

轉換到指定用戶坐標系

[語法]

@用戶坐標系編號

[參數說明]

用戶坐標系編號　　指定切換的用戶坐標系編號。範圍 1~64

[使用案例]

```
Function main
    GO XY(10 , 20 , 30 ,40)  @2
Fend
```

[功能]

方向分量

[語法]

+/-/:方向分量

[參數說明]

[使用案例]

```
Function main
    Go P1 + X(10)
    Move P2 - U(5)
    Go P1 +X(5) /R              '以右手系移動到 P1 點 X+5mm 處
    Go P1 +X(5) /3              '以用戶坐標系 3 移動到 P1 點 X+5mm 處
    Go P1 :X(10)                '移動到 P1 點且 X=10mm 處
Fend
```

[功能]

指定用戶坐標系繞 X、Y、Z 軸的旋轉分量

[語法]

+/-[RX/RY/RZ]

[參數說明]

[使用案例]

```
Function main
      Go P1 + X(10) + RX(10)
    'P1 點的用戶座標繞 X 軸旋轉 10°，並移動到 P1 點 X+5mm 處
Fend
```

[功能]

指定工具坐標系繞 X、Y、Z、U、V、W 軸的旋轉分量

[語法]

+/-[TLX/TLY/TLZ/TLU/TLV/TLW]

[參數說明]

[使用案例]

```
Function main
      Go P1 + X(10) + TLX(10)
    'P1 點的用戶座標繞 X 軸旋轉 10°，並移動到 P1 點 X+5mm 處
Fend
```

[功能]

將當前程式記憶體中的點位數據保存至點位檔中。檔不存在則會新建檔。

[語法]

SavePoints 檔案名

[參數說明]

檔案名　　以字串命名點位檔案名

[使用案例]

```
Function main
    P1 = XY(10 , 20 , 30 , 40)
    P2 = XY(40 , 30 , 20 , 10)
    SavePoints "ABCD" '將 P1、P2 保存至 ABCD 點位檔中
Fend
```

[功能]

加載點位檔

[語法]

LoadPoints 檔案名{, Merge}

[參數說明]

檔案名	以字串命名點位檔案名
Merge	用於讀入新的點之前，不想清除當前的點時進行設置。如果進行設置，則將新的點添加到設置的點中。檔中已存在要添加的點時，執行覆寫。可省略。

[使用案例]

```
Function main

    '將通用的點讀入到當前的機器人中
    LoadPoints "R1Common.pts"

    '合併部件模型 1 的點
    LoadPoints "R1Model.pts" , Merge

    '讀入機器人 2 的點檔
    LoadPoints "R2Model.pts"

Fend
```

[功能]

用於程式記憶體儲存的點位數據

[語法]

```
ClearPoints
```

[參數說明]

[使用案例]

```
Function main
    P1 = XY(10 , 20 , 30 , 40)
    P2 = XY(40 , 30 , 20 , 10)
    ClearPoints       '前面定義的 P1、P2 點將不復存在
Fend
```