研華MAS控制器

Motion Basic程式設計手冊

Version 1.6.1

第 1 章　研華 MOTION BASIC 介紹

1.1　MOTION BASIC 概述

Motion Basic是一種用於研華MAS控制器的多工程式設計語言，它的語法與標準Basic相似。在裝有windows作業系統的電腦上運行Motion Studio就可以進行開發和測試所有MAS控制器功能。Motion Studio提供了Basic程式編輯和豐富的調試工具來進行應用程式的調試，使設備應用程式開發變得簡單高效。

1.2　特點

- 基本程式指令和Visual Basic相容，運動控制指令十分簡單，初學者也很容易上手。
- 支持10個任務獨立執行、同時執行。
- 支持所有研華softmotion功能指令，並可整合進外部演算法和函數到Motion Basic指令，非常適合設備整合控制和高效開發。
- 編譯執行方式，使得在簡單易用的前提下，同時具有高階語言高效執行的優點。
- 可以同時運行多個子任務，強大的多工處理功能提高軟體結構和易維護性。
- 穩定性高：當使用者程式出現錯誤時，不會出現系統崩潰。

常見的三種指令特點比較表

	MOTION BASIC	G 代碼	梯形圖
可理解方式	較容易	容易	困難
執行方式	編譯執行	解釋執行	編譯執行
執行速度	快	較慢	慢
功能	全	較少	全
操作硬體能力	所有	只支持電機、IO	所有
副程式	支持	支持	不支持
任務數	10	1 主任務/3 從任務	不支持
變數	支持	不支持	支持
陣列	支持	不支持	不支持
數學函數	支持	不支持	支持

第 2 章　BASIC 指令詳解

MOTION BASIC 規則說明

研華 Motion Basic 指令主要包含運動控制指令和其他指令兩大類。根據研華運動控制的特點，我們把運動控制指令分為命令和屬性兩部分指令，在第 2 章 Basic 指令詳解中有說明每條運動控制指令屬於命令還是屬性。其他指令的規則基本上類似於標準 Basic 指令規則。本節將研華 MOTION BASIC 指令說明的一些重要規則描述如下：

- **研華 MOTION BASIC 指令集不區分字母大小寫**
- **指令語法說明方法：語法說明中包含( )的部分，必須要使用( )，不可省略。說明中有[ ]的部分，表示[ ]中的內容根據實際需求可以填或不填，不會造成語法錯誤。**

  如下 LINE 指令說明：

  LINE　distance1,distance2 [,distance3]；做兩軸直線插補時，LINE 指令後面的參數填 distance1 和 distance2 就可以了。做三軸直線插補時，LINE 指令後面的參數就填 distance1,distance2,distance3。[,distance3]這個部分是根據實際需求選填的，選填的部分我們用[ ]說明。

- **運動控制屬性類指令的賦值說明**

  運動控制屬性類指令賦值是用 “=” 來做賦值的，如要設置加速度，要用 ACC=value 這樣的語句，ACC value 這樣的語句則不符合語法規則。

- **運動控制指令中對指定軸的操作說明**

  運動控制指令對指定軸操作時，指定軸號時要使用 “AX” 這個關鍵字，僅用數字代表軸號不符合語法規則。如將鎖存到的軸 0 理論位置值賦給一個變數 A，會用到指令 LDPOS，語句是 A=LDPOS(AX(0))，A=LDPOS(0)則不符合語法規則。

- **系統已定義運動控制關鍵字**

  注意:使用者自訂的變數名不能與以下表格中的關鍵字或枚舉名一樣，也不能與 Motion Basic 的所有指令名一樣。

關鍵字或枚舉	說明	例子
AX	指定軸號的關鍵字，AX(0)代表軸 0，AX(5)代表軸 5	MVOE AX(0),1000 表示命令軸 0 運動 1000 個脈衝當量
CW	順時針方向	CIRC 0,5000,0,10000,0 可以用 CIRC CW,5000,0,10000,0 來代替
CCW	逆時針方向	CIRC 1,5000,0,10000,0 可以用 CIRC CCW,5000,0,10000,0 來代替

S	S 型曲線	JK=0 可以用 JK=T 來代替
T	T 型曲線	JK=1 可以用 JK=S 來代替
DONE	軸運動完成事件	WAIT DONE；參考 WAIT 指令說明
COMPARED	比較觸發完成事件	WAIT COMPARED；參考 WAIT 指令說明
LATCHED	鎖存完成事件	WAIT LATCHED ；參考 WAIT 指令說明
AXIS	系統內部定義的軸關鍵字	
RUN	系統內部定義的運行關鍵字	
STOP	系統內部定義的停止關鍵字	
TASK	系統內部定義的任務關鍵字	
DIR	系統內部定義的方向關鍵字	

2.1 流程控制語句

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.1.1	DIM...As...	定義資料類型	×	×
2.1.2	CONST	定義常量	×	×
2.1.3	IF...THEN...ELSEIF ...ELSE...END IF	IF 條件陳述式	×	×
2.1.4	FOR...TO... STEP...NEXT	FOR 迴圈語句	×	×
2.1.5	Select Case...End Select	CASE 語句	×	×
2.1.6	WHILE...WEND	While 迴圈語句	×	×
2.1.7	WAIT	等待事件結束	×	×
2.1.8	CancelWAIT	退出事件的等待	×	×
2.1.9	SLEEP	延時	×	×
2.1.10	TYPE	定義類	×	×
2.1.11	EXIT	退出一個控制流語句塊或函數體	×	×

2.1.1 DIM...As...

語　法 1：DIM　varname1 As　DataType [,varname2 As DataType,...]

語　法 2：DIM　As DataType varname1 [,varname2, ...]

描　述：　定義變數，陣列

參　數：　varname　　變數名

注　意：　在一個 Task 裡用 DIM 定義的區域變數只在該 Task 當前的程式段起到聲明的作用，如果要聲明到該 Task 的副程式段（如 SUB 段程式），那需要在 DIM 後面加上關鍵字 shared，如這樣定義一個變數 x：DIM shared x As Integer。

例　程

```
DIM a AS DOUBLE                    '定義一個變數名為 a 的 64 位元浮點型變數
DIM text AS STRING ="Hi,MAS"       '定義一個變數名為 text 的字串，並賦值為 HI,MAS
DIM Array(3) AS BYTE={1,2,5}       '定義一個長度為 3，名為 Array 的 BYTE 類型陣列,並賦值
DIM AS ULONG b,c,d,e               '一次定義多個同類型變數
```

2.1.2 CONST

語　法：CONST　varname = value

描　述：定義常量。採用常量名可避免在程式中多處修改同一個數值

參　數：　varname　　變數名

value　　常數值

例　程

```
CONST max_speed=50000          '定義速度常量
CONST Distance=10000           '定義位移常量
BASE 0
SVON
VH = max_speed                 '將速度常量賦給運行速度屬性
MOVE Distance                  '將位移常量賦給位移指令參數
```

2.1.3 IF...THEN...ELSEIF...ELSE...END IF

語　法：IF <condition1> THEN

commands1

[ELSEIF <condition2> THEN]

commands2

[ELSE commands3]

END IF

描　述：該指令為條件陳述式。首先判斷條件運算式 1；如果條件運算式 1 成立，則執行指令塊 1，然後跳轉至 END IF，退出條件陳述式。如果條件運算式 1 不成立，則判斷條件運算式 2；如果條件運算式 2 成立，則執行指令塊 2，然後跳轉至 endif，退出條件陳述式。如果條件運算式 2 不成立，則執行指令塊 3

參　數：　condition　　條件運算式

commands　　指令塊（可單條指令也可多條指令）

例　程

```
DIM a AS LONG
IF(a=0)    THEN                    '判斷條件 1：a 是否為 0
    DOUT (2)=1                     '條件 1 成立，則 DO2, DO6 分別賦值 1,1
    DOUT (6)=1
ELSEIF( a>0 AND a<=5) THEN         '判斷條件 2：a 是否大於 0 且小於等於 5
    DOUT(2)=0                      '條件 2 成立，則 DO2, DO6 分別賦值 0,1
    DOUT(6)=1
ELSE                               '條件 1、2 都不成立的情況，則 DO2, DO6 分別賦值 1,0
    DOUT(2) =1
    DOUT(6)=0
END IF
```

2.1.4　FOR...TO...STEP...NEXT

語　法：For varname = startvalue TO endvalue [STEP stepvalue]

[comands]

NEXT　varname

描　述：FOR 迴圈語句。如果迴圈變數小於迴圈結束值，則執行指令塊到 NEXT 時，迴圈變數自動加一個增量，再一次執行指令塊；當迴圈變數大於等於迴圈結束值時，則停止迴圈。

參　數：

varname	循環體變數名
startvalue	迴圈起始值
endvalue	迴圈結束值
stepvalue	迴圈變數的增量，可選；缺省時，增量為 1。增量也可以是小數或負數。增量為負數時，迴圈起始值要大於迴圈結束值。
Comands	指令塊

注　意：　該指令最多嵌套八層

例　程

```
DIM i AS ULONG
'將 DO0~DO7 全部置 1
FOR i=0 To 7
        DOUT (i) =1
```

```
NEXT i
'將 DO0、DO2、DO4、DO6 置 0
FOR i=0 To 7 STEP 2
        DOUT (i) =0
NEXT i
```

2.1.5 Select Case...End Select

語　法：Select Case expression
[Case expressionlist]
[commands]
.....
[Case expressionlist]
[commands]
[Case Else]
[commands]
End Select

描　述：類似 C 語言的 Switch...Case 條件陳述式。根據運算式的內容選擇執行哪個指令塊，各分支 CASE 運算式條件內容需互斥，運算式內容會逐一匹配各 CASE 的條件內容，直到匹配成功，一旦匹配成功，相應分支 CASE 下的指令塊只執行一次，然後程式跳轉到 End select，結束 CASE 條件陳述式。如果寫了 Case Else，則等 Case Else 以上的各分支 CASE 都沒有匹配成功時，就會執行 Case Else 下的指令塊。

參　數：
expression	運算式
expressionlist	case 分支運算式條件內容
comands	指令塊

注　意：　各 case 分支運算式內容需互斥

例　程

```
Dim choice As LONG
Select Case choice
Case 1
    Print "number is 1"                    'choice 為 1 時，列印 number is 1
Case 2
    Print "number is 2"                    'choice 為 2 時，列印 number is 2
```

```
Case 3, 4
    Print "number is 3 or 4"                    'choice 為 3 或 4 時，列印 number is 3 or 4
Case 5 To 10
    Print "number is 5 to 10"                   'choice 為 5~10 時，列印 number is 5 to 10
Case Else
    Print "number is outside the 1-10 range"    '非以上情況，列印 number is outside the 1-10 range
End Select
```

2.1.6 WHILE...WEND

語　法：WHILE condition
　　commands
　　WEND

描　述：迴圈語句。當 condition 條件成立時，執行循環體內的指令塊；否則，結束循環體。

參　數：condition　條件運算式
　　comands指令塊

例　程

```
Dim i AS LONG
WHILE(i=5)                  '如果 i=5，則在 WHILE 和 WEND 之間的指令段迴圈執行
   BASE 0
   SVON
   MOVE 1000
   WAIT DONE
   MOVE -1000
   WAIT DONE
WEND
```

2.1.7 WAIT

語　法 1：WAIT [AX(no),]DONE

語　法 2：WAIT [AX(no),] LATCHED

語　法 3：WAIT [AX(no),] COMPARED

語　法 4：WAIT [CYL(no),] CYLDONE

語　法 5：WAIT [AX(no),] LTCBUFDONE

描　述：WAIT 指令表示等待某個事件結束，WAIT 後面未指定軸或氣/油缸時，等待的是當前 BASE 列表中的所有軸或氣/油缸的相應事件。WAIT 指令後面跟的事件暫只支援 DONE、LATCHED、COMPARED、CYLDONE、LTCBUFDONE 五種事件。

參　數：

AX(no)	指定軸號，no 填軸號數字
DONE	指運動結束事件
LATCHED	指鎖存發生事件
COMPARED	指比較觸發事件
CYLDONE	指氣缸動作完成事件
LTCBUFDONE	指鎖存緩存中資料個數達到設定數量時觸發，完整用法可參考 LBUF_DATA 指令

注　意：WAIT 指令使用時要小心，未等到事件發生，程式會一直停在 WAIT 這行。特別是等待多個事件時，要確保事件都會發生。

例　程

```
BASE 0,1,2
MOVE 10000,20000,30000
WAIT AX(2),DONE              '等待軸 2 運動結束後，程式執行下一行，否則一直等在該行
BASE 0,1
MOVE 20000,5000
WAIT DONE                    '等待軸 0,1 運動都結束後，程式執行下一行，否則一直等在該行
BASE 0
MOVE 10000
WAIT DONE                    '等待軸 0 運動結束後，程式執行下一行，否則一直等在該行
DOUT (2) =1
```

2.1.8　CancelWAIT

語　法 1：CancelWAIT [AX(no),]DONE

語　法 2：CancelWAIT [AX(no),] LATCHED

語　法 3：CancelWAIT [AX(no),] COMPARED

語　法 4：CancelWAIT [CYL(no),] CYLDONE

語　法 5：CancelWAIT [AX(no),] LTCBUFDONE

描　述： CancelWAIT 指令表示退出等待某個事件結束，一般是對應 WAIT 指令來使用的。當執行了 CancelWAIT 指令後，當前系統所有 Task 正在執行的 Wait 事件語句會退出等待，各 Task 程式會立刻接著執行 Wait 語句後面的程式列。

參　數：

AX(no)	指定軸號，no 填軸號數字
DONE	指運動結束事件
LATCHED	指鎖存發生事件
COMPARED	指比較觸發事件
CYLDONE	指氣缸動作完成事件
LTCBUFDONE	指鎖存緩存中資料個數達到設定數量時觸發，完整用法可參考 LBUF_DATA 指令

例　程

```
'下麵例程有兩個 Task 同時執行，Task 2 開始執行後會，軸 0 一直處於正向連續運動狀態，程式會等
'待在 WAIT DONE 指令行。這時候如果 VR(0)值變為 1，Task 1 中會執行 CancelWAIT DONE'，
'Task 2 中 WAIT DONE 指令就會退出等待，程式會繼續執行 STOPDEC 這行指令，然後 DOUT(0)就
'會置成 ON 狀態
'***Task 1***'
WHILE 1
  IF(VR(0)=1)    Then
      BASE 0
      CancelWAIT DONE
      VR(0)=0
  End IF
  Sleep 10
WEND
'***Task 1***'
'***Task 2***'
BASE 0
SVON
FORWARD          '執行正向連續運動
WAIT DONE        '等待運動結束
STOPDEC          '減速停止
DOUT(0)=1        'DO0 ON
```

```
'***Task 2***'
```

2.1.9 SLEEP

語　法：SLEEP delay_time

描　述：延時指令

參　數：delay_time　延時時間，單位：ms

例　程

```
BASE 0
MOVE 10000
WAIT DONE
SLEEP 500              '延時 500 毫秒
MOVE -10000
```

2.1.10 TYPE

語　法：TYPE type_name

描　述：定義一個類（類似物件導向語言中的類）

參　數：type_name　類的名稱

例　程

```
'最終列印出來的結果如下
' Number of Values =   4
'Average              =   19.2
'以下為例程
Type Statistics
  count As Single
  sum As Single
  Declare Sub AddValue( ByVal x As Single )
  Declare Sub ShowResults( )
End Type
Sub Statistics.AddValue( ByVal x As Single )
  count += 1
  sum += x
End Sub
Sub Statistics.ShowResults( )
  Print "Number of Values = "; count
```

```
  Print "Average              = ";
  If( count > 0 ) Then
    Print sum / count
  Else
    Print "N/A"
  End If
End Sub
Dim stats As Statistics
stats.AddValue 17.5
stats.AddValue 20.1
stats.AddValue 22.3
stats.AddValue 16.9
stats.ShowResults
```

2.1.11 EXIT

語　法：EXIT　Do | For | While | Select

EXIT　Sub | Function

EXIT　DO[, DO[,...]]

EXIT　For[, For[,...]]

EXIT　While[, While[,...]]

EXIT　Select[, Select[,...]]

描　述：退出一個控制流語句塊或函數體。

例　程

```
DIM i AS USHORT
FOR i=0 To 7
        DOUT (i) =1
        IF(i=5) THEN
          EXIT FOR '當 i 為 5 時，結束 FOR 循環體
        END IF
NEXT i

DIM As Integer i, j
For i = 1 To 10
  For j = 1 To 10
    Exit For, For    '嵌套的控制流語句塊退出指令，結束 FOR 嵌套循環體
```

```
    Next j
    Print "I will never be shown"
Next i
```

2.2 副程式、多工控制語句

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.2.1	SUB	定義一個函數體	×	×
2.2.2	END	結束當前任務或程式	×	×
2.2.3	RUN_TASK	運行指定的 Task	√	×
2.2.4	STOP_TASK	停止運行指定的 Task	√	×
2.2.5	STOP_ALL	停止運行所有 Task，並停止所有軸的運動	√	×
2.2.6	Task_Status	讀取 Task 的狀態	√	×

2.2.1 SUB

語　法：SUB　name

描　述：定義一個函數體

參　數：name　函數體名

注　意：調用函數時，SUB 函數體必須要寫在調用該函數之前，否則編譯時會出錯。如果想將定義的 SUB 函數體寫在任意位置，可以在 TASK 開頭用 **Declare Sub name** 語句先定義。

Declare 指令特別說明如下：

Declare 為聲明的指令，一般用於聲明 SUB（無返回值的函數體）和 Function（有返回值的函數體）,語法如下：

Declare Sub name [param_list]

Declare Function name [param_list] As return_type

例　程

```
'定義一個函數體，名為 abc
SUB abc
BASE 0
```

```
SVON
MOVE 50000
WAIT DONE
END SUB

BASE 1
SVON
MOVE 10000
WAIT DONE
abc                '調用函數名為 abc 的函數體
BASE 1
MOVE -10000
```

2.2.2 END

語　法：END

描　述：結束當前任務或程式，END 指令後面可以跟 SUB，IF 等

例　程

```
'可以參考 IF、SUB 等指令例程
```

2.2.3 RUN_TASK

所　屬：命令

語　法：RUN_TASK　“task_name”

描　述：運行指定的 Task

參　數：　task_name　TASK 文本名稱，此處名稱不能加 .bas 尾碼，只需填.bas 前面的名稱

例　程

```
'如用戶創建了 3 個 TASK，分別為 test1.bas，test2.bas,test3.bas，需在 test3.bas 裡運行 test1.bas，
test2.bas 這兩個 TASK，可在 test3.bas 裡寫如下例程
RUN_TASK "test1"
```

```
RUN_TASK "test2"
```

2.2.4 STOP_TASK

所　屬：命令

語　法：STOP_TASK　“task_name”

描　述：停止運行指定的 Task

參　數： task_name　TASK 文本名稱，此處名稱不能加 .bas 尾碼，只需填.bas 前面的名稱

例　程

```
STOP_TASK "test"        ' 停止運行名為 test 的 TASK
```

2.2.5 STOP_ALL

所　屬：命令

語　法：STOP_ALL

描　述：停止運行所有 Task，並停止所有軸的運動

例　程

```
RUN_TASK "test1"        ' 運行名為 test1 的 TASK
RUN_TASK "test2"        ' 運行名為 test2 的 TASK
Sleep 1000              '延時 1 秒
STOP_ALL                ' 停止運行所有 TASK，並停止所有軸運動
```

2.2.6 Task_Status

語　法：value=Task_Status (taskname)

描　述：讀取 Task 的狀態

參　數：taskname 要讀取狀態的 Task 名稱；類型：String

返回值：0：停止；1：運行中　　類型：ULONG

例　程

```
DIM   A AS ULONG
A=Task_Status("test1")      'test1 為其中一個 Task 的名稱
```

2.3 運算子及數學函數

2.3.1 運算子

當運算式包含多種運算子時，首先計算算術運算子，然後計算比較運算子，最後計算邏輯運算子。所有比較運算子的優先順序相同，即按照從左到右的順序計算比較運算子。

當乘號與除號同時出現在一個運算式中時，按從左到右的順序計算乘、除運算子。同樣當加與減同時出現在一個運算式中時，按從左到右的順序計算加、減運算子。

算術運算子說明如表 3.1 所示。

表 3.1 運算子說明

算術運算子		比較運算子		邏輯運算子	
描述	符號	描述	符號	描述	符號
加	+	等於	=	邏輯非	NOT
減	-	不等於	<>	邏輯與	AND
乘	*	小於	<	邏輯或	OR
除	/	大於	>	邏輯異或	XOR
整除	\	小於等於	<=	邏輯等價	EQV
求餘數	Mod	大於等於	>=		
求相反數	-				
冪	^				

2.3.1.1 NOT

語　法：NOT expression

描　述：對運算式進行非操作或對數值按二進位的位元進行“非”運算

參　數：expression　　運算式

注　意：只針對運算式的值的整數部分運算

2.3.1.2 AND

語　法：expression1 AND expression2

描　述：對兩個運算式進行與操作或對兩個數值按二進位的位元進行“與”運算

參　數：expression1　　運算式 1

expression2　　運算式 2

注　意：只針對運算式的值的整數部分運算

2.3.1.3　OR

語　法：expression1 OR expression2

描　述：對兩個運算式進行或操作或對兩個數值按二進位的位元進行“或”運算

參　數：expression1　　運算式 1

expression2　　運算式 2

注　意：只針對運算式的值的整數部分運算

2.3.1.4　XOR

語　法：expression1 XOR expression2

描　述：對兩個運算式進行異或操作或對兩個數值按二進位的位元進行“異或”運算

參　數：expression1　　運算式 1

expression2　　運算式 2

注　意：只針對運算式的值的整數部分運算

2.3.1.5　EQV

語　法：expression1 EQV expression2

描　述：對兩個運算式進行同或操作或對兩個數值按二進位的位元進行“同或”運算

參　數：expression1　　運算式 1

expression2　　運算式 2

注　意：只針對運算式的值的整數部分運算

2.3.2 數學函數

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.3.2.1	ABS	求絕對值	×	×
2.3.2.2	SIN	正弦函數	×	×
2.3.2.3	ASIN	反正弦函數	×	×
2.3.2.4	COS	余弦函數	×	×
2.3.2.5	ACOS	反余弦函數	×	×
2.3.2.6	TAN	正切函數	×	×
2.3.2.7	ATN	反正切函數	×	×
2.3.2.8	ATAN2	用比值求反正切函數	×	×
2.3.2.9	SQR	求平方根	×	×
2.3.2.10	LOG	自然對數	×	×
2.3.2.11	BITRESET	位操作置 0	×	×
2.3.2.12	BIT	位操作讀值	×	×
2.3.2.13	BITSET	位操作置 1	×	×
2.3.2.14	FRAC	求小數部分的值	×	×
2.3.2.15	INT	求整數部分的值	×	×
2.3.2.16	SGN	判斷值的符號	×	×

2.3.2.1 ABS

語　法：ABS(expression)

描　述：求運算式的絕對值

參　數：expression　　運算式

2.3.2.2 SIN

語　法：SIN (expression)

描　述：計算運算式的正弦函數

參　數：expression　　運算式，單位：弧度

例　程

```
CONST PI AS DOUBLE = 3.1415926535897932
DIM a AS DOUBLE                 '定義一個變數用於存角度值
DIM r AS DOUBLE                 '定義一個變數用於存弧度值
DIM sin_value as DOUBLE         '定義一個變數用於取正弦值
a=90
r = a*PI/180                    '把 a 轉換成弧度值存於 r
sin_value=SIN (r)
PRINT r
PRINT sin_value
```

運行結果：r 為 1.570796326794897

sin_value 為 1

2.3.2.3　ASIN

語　法：ASIN (expression)

描　述：計算運算式的反正弦函數，返回值單位：弧度

參　數：expression　　運算式

2.3.2.4　COS

語　法：COS (expression)

描　述：計算運算式的余弦函數

參　數：expression　　運算式，單位：弧度

2.3.2.5　ACOS

語　法：ACOS (expression)

描　述：計算運算式的反余弦函數　，返回值單位：弧度

參　數：expression　　運算式

2.3.2.6　TAN

語　法：TAN (expression)

描　述：計算運算式的正切函數

參　數：expression　　運算式，單位：弧度

2.3.2.7 ATN

語　法：ATN (expression)

描　述：計算運算式的反正切函數，返回值單位：弧度

參　數：expression

2.3.2.8 ATAN2

語　法：ATAN2　(number1，number2)

描　述：計算 number1/number2 比值的反正切函數，返回值單位：弧度

參　數：number1　　比值分子

number2　　比值分母

2.3.2.9 SQR

語　法：SQR (expression)

描　述：計算運算式的平方根

參　數：expression　　運算式

2.3.2.10 LOG

語　法：LOG (expression)

描　述：計算運算式的自然對數（以 e 為底的對數）

參　數：expression　　運算式

2.3.2.11 BITRESET

語　法：BITRESET(value，bit_num)

描　述：將運算元的二進位的第 bit 位清 0

參　數：bit_num　　位編號：0~31（二進位數字的位，從低（右）至高（左）排列）

value　　運算元

注　意：只針對運算元的整數部分操作

例　程

```
DIM AS ULONG a,b
a=5
b=BITRESET(a,0)          'a 為 5，二進位即 101，清掉 0 位，即 b 得到 100
PRINT b                  'b 為 100，列印出來即為 4
b=BITRESET(a,2)          'a 為 5，二進位即 101，清掉 2 位，即 b 得到 001
```

```
PRINT b                    'b 為 001，列印出來即為 1
```

2.3.2.12 BIT

語　法：BIT(value，bit_num)

描　述：讀取運算元的二進位的第 bit 位的值，讀到 bit 位為 0 值，返回 0；讀到 bit 位為 1 值，返回-1

參　數：bit_num　　位編號：0~31（二進位數字的位，從低（右）至高（左）排列）

value　　運算元

注　意：只針對運算元的整數部分操作

2.3.2.13 BITSET

語　法：BITSET(value，bit_num)

描　述：將運算元的二進位的第 bit 位的置 1

參　數：bit_num　　位編號：0~31（二進位數字的位，從低（右）至高（左）排列）

value　　運算元

注　意：只針對運算元的整數部分操作

2.3.2.14 FRAC

語　法：FRAC(expression)

描　述：返回運算式的小數部分

參　數：expression　　運算式

注　意：此函數僅支援大於 0 的運算式

2.3.2.15 INT

語　法：INT(expression)

描　述：返回運算式的整數部分

參　數：expression　　運算式

注　意：當運算式的值小於 0 時，返回值比其整數小 1

2.3.2.16 SGN

語　法：SGN(expression)

描　述：判斷運算式是大於 0、等於 0，還是小於 0。當運算式大於 0，函數的返回值為 1；當運算式等於 0 時，函數的返回值為 0；當運算式小於 0，函數的返回值為－1

參　數：expression　　運算式

2.4 控制器系統指令

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.4.1	BASE	該指令後面所有的軸指令和軸參數設置和讀取都基於該指令選定的軸號	√	×
2.4.2	UNIT_NUM	脈衝當量分子	√	√
2.4.3	UNIT_DENOM	脈衝當量分母	√	√
2.4.4	POUT_MODE	指令脈衝輸出類型	√	√
2.4.5	POUT_REVERSE	指令脈衝輸出邏輯反相	√	√
2.4.6	PIN_MODE	編碼器脈衝輸入類型	√	√
2.4.7	PIN_MAXFREQ	編碼器脈衝輸入的最高頻率限值	√	√
2.4.8	PIN_LOGIC	編碼器脈衝輸入邏輯反相	√	√
2.4.9	DPOS	理論位置值（指令脈衝）	√	√
2.4.10	MPOS	實際位置值（編碼器回饋脈衝）	√	√
2.4.11	SVON	使能伺服	√	×
2.4.12	SVOFF	禁用使能伺服	√	×
2.4.13	ERROR_AXIS	當前哪些軸發生了軸狀態錯誤	√	×
2.4.14	RUN_ERROR	軸錯誤資訊	√	×
2.4.15	SYSTEM_ERROR	系統級錯誤資訊	√	×
2.4.16	CLEAR_ERROR	清除系統錯誤狀態	√	×
2.4.17	PRINT	在 Motion Studio 中的資訊輸出表單列印資訊	×	×
2.4.18	DATE	獲取當前控制器日期：月-日-年	×	×
2.4.19	TIME	獲取當前控制器日期：小時：分鐘：秒	×	×
2.4.20	SETDATE	給控制器系統設置新的日期	×	×
2.4.21	SETTIME	給控制器系統設置新的時間	×	×

2.4.22	TIMER	返回程式段程式執行的時間	×	×

2.4.1 BASE

所　屬：命令

語　法：BASE　axis no [,second axis][,third axis] …

描　述：為了簡化程式設計，可以用該指令選擇要參與運動的軸號，其後的指令就沒必要填寫所有軸的參數，只填寫參與運動的軸參數即可。軸號要按順序填寫，軸號可以是 1 個，也可以是 2 個、3 個…

參　數：axis no 軸號；範圍：根據控制器實際硬體決定。

例　程

```
BASE 0,1,2,3
VH=8000                          '軸 0,1,2,3 的最大運行速度都設置為 8000
MOVE 10000，4000，2000，6000    '軸 0,1,2,3 都執行單軸相對點位運動
BASE 1
VH=1000                          '軸 1 的最大運行速度設置為 1000
MOVE 10000                       '軸 1 執行單軸相對點位運動
```

2.4.2 UNIT_NUM

所　屬：屬性

語　法：UNIT_NUM = value

類　型：ULONG

描　述：設置/讀取脈衝當量分子

範　圍：大於 0，預設值 1

例　程

```
BASE 0
UNIT_NUM　=10　'設置軸 0 的脈衝當量分子
```

2.4.3 UNIT_DENOM

所　屬：屬性

語　法：UNIT_DENOM = value

類　型：ULONG

描　述：設置/讀取脈衝當量分母

範　圍：（0, MAX_PULSE），預設值 1

例　程

```
BASE 0
UNIT_DENOM=40     '設置軸 0 的脈衝當量分母為 40
```

2.4.4 POUT_MODE

所　屬：屬性

語　法：POUT_MODE= value

類　型：ULONG

描　述：設置/讀取指令脈衝輸出類型

範　圍 :如下設定值，預設值 5

0：OUT/DIR

1：OUT/DIR，OUT 負邏輯

2：OUT/DIR，DIR 負邏輯

3：OUT/DIR，OUT&DIR 負邏輯

4：CW/CCW

5：CW/CCW，CW&CCW 負邏輯

例　程

```
BASE 0
POUT_MODE =2      '設置指令脈衝輸出類型為 OUT/DIR，DIR 負邏輯
```

2.4.5 POUT_REVERSE

所　屬：屬性

語　法：POUT_REVERSE = value

類　型：ULONG

描　述：啟用/禁用指令脈衝輸出埠信號對調

範　圍 :如下設定值，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
POUT_REVERSE   =1 '啟用指令脈衝輸出埠信號對調
```

2.4.6 PIN_MODE

所　屬：屬性

語　法：PIN_MODE= value

類　型：ULONG

描　述：設置/讀取編碼器輸入脈衝類型

範　圍 :如下設定值，預設值 2

0：1XAB

1：2XAB

2：4XAB

3：CCW/CW

例　程

```
BASE 0
PIN_MODE =3 '設置編碼器輸入脈衝類型為 CCW/CW
```

2.4.7 PIN_MAXFREQ

所　屬：屬性

語　法：PIN_MAXFREQ = value

類　型：ULONG

描　述：設置/讀取編碼器輸入脈衝的最高頻率

範　圍 :如下設定值，預設值 0

0：500KHz

1：1MHz

2：2MHz

3：4MHz

例　程

```
BASE 0
PIN_MAXFREQ =1  '設置編碼器輸入脈衝的最高頻率為 1MHz
```

2.4.8 PIN_LOGIC

所　屬：屬性

語　法：PIN_LOGIC = value

類　型：ULONG

描　述：設置/讀取編碼器輸入脈衝的邏輯

範　圍 :如下設定值，預設值 0

0：不反轉方向

1：反轉方向

例　程

```
BASE 0
PIN_LOGIC =1   '設置編碼器輸入脈衝邏輯為反轉方向
```

2.4.9 DPOS

所　屬：屬性

語　法：DPOS = value

類　型：DOUBLE

描　述：設置/讀取軸當前的理論位置

範　圍：64 位浮點資料類型範圍

例　程

```
BASE 0
DIM A AS DOUBLE
A=DPOS    '將軸 0 的當前理論位置賦值給變數 A
BASE 1
VR(10)=DPOS '將軸 1 的當前理論位置賦值給全域變數 VR（10）
DPOS=0 '將軸 1 的當前理論位置計數器賦 0
```

2.4.10 MPOS

所　屬：屬性

語　法：MPOS = value

類　型：DOUBLE

描　述：設置/讀取軸當前的編碼器回饋位置

範　圍：64 位浮點資料類型範圍

例　程

```
BASE 0
DIM A AS DOUBLE
A=MPOS    '將軸 0 的當前實際位置賦值給變數 A
BASE 1
VR(10)=MPOS '將軸 1 的當前實際位置賦值給全域變數 VR（10）
MPOS=0 '將軸 1 的當前實際位置計數器賦 0
```

2.4.11 SVON

所　屬：命令

語法 1：SVON

語法 2：SVON AX(axis no)

描　述： BASE 軸列表的軸或指定軸，使能軸

參　數：axis no　軸號； 範圍：根據控制器實際硬體決定

例　程

```
'對 BASE 列表中的軸使能，即使能伺服
BASE 2
SVON              '使能軸 2
BASE 0,1,3,5
SVON              '使能軸 0、1、3、5
'指定軸使能
SVON AX(2)        '使能軸 2
```

2.4.12 SVOFF

所　屬：命令

語法 1：SVOFF

語法 2：SVOFF AX(axis no)

描　述： BASE 軸列表的軸或指定軸，禁用軸使能

參　數：axis no　軸號； 範圍：根據控制器實際硬體決定

例　程

```
'對 BASE 列表中的禁用軸使能
BASE 2
SVOFF             '禁用軸 2 使能
BASE 0,1,3,5
SVOFF             '禁用軸 0、1、3、5 使能
```

```
SVOFF AX(2)    '禁用軸 2 使能
```

2.4.13 ERROR_AXIS

所　屬：屬性 (唯讀)

語　法：value=ERROR_AXIS

類　型：ULONG

描　述：讀取當前哪些軸發生了軸狀態錯誤。該屬性為 32 位寄存器，每一位代表一個軸。位值為 0 表示該軸無錯誤發生，位元值為 1 表示該軸發生了軸狀態錯誤。當發生軸狀態錯誤時，可以用 RESETERR 指令清除軸狀態錯誤。

返回值：0：無錯誤發生；1：發生了軸狀態錯誤

例　程

```
DIM ErrorReturn As ULONG
ErrorReturn=ERROR_AXIS
IF (ErrorReturn=4) THEN   ' ErrorReturn 為 4 時，表示軸 2 發生了軸狀態錯誤
RESETERR AX(2)            '清除軸 2 的軸狀態錯誤
End if
```

2.4.14 RUN_ERROR

所　屬：屬性 (唯讀)

語　法：value=RUN_ERROR

類　型：ULONG

描　述：根據 BASE 軸清單中的軸，讀取軸錯誤資訊。錯誤代碼資訊請參照章節 2.15 RUN_ERROR 錯誤代碼資訊表。

返回值：錯誤代碼

例　程

```
DIM ErrorCode As ULONG
BASE 0
ErrorCode = RUN_ERROR
```

2.4.15 SYSTEM_ERROR

所　屬：屬性 (唯讀)

語　法：value=SYSTEM_ERROR

類　型：ULONG

描　述：讀取系統級錯誤資訊。錯誤代碼資訊請參考章節 2.15 SYSTEM_ERROR 錯誤代碼資訊表

返回值：錯誤代碼

例　程

```
DIM ErrorCode As ULONG
ErrorCode = SYSTEM_ERROR
```

2.4.16 CLEAR_ERROR

所　屬：命令

語　法：CLEAR_ERROR

描　述：清除系統錯誤狀態。該命令僅用於清除 SYSTEM_ERROR 對應的系統錯誤狀態

2.4.17 PRINT

所　屬：命令

描　述：在 Motion Studio 中的資訊輸出表單列印資訊。

例　程

```
Dim A As ULONG
A=42
Print   "Hello"   ‘列印字串
Print   VL     ‘列印初速度屬性值
Print   A      ‘變數值
Print   3*4   ‘列印一個運算式結果，列印結果為 12
```

2.4.18 DATE

語　法：value=DATE

類　型：String

描　述：獲取當前控制器日期：月-日-年

例　程

```
DIM str1 AS string
str1=DATE
print str1        '如現在是 2016 年 3 月 15 日，列印結果為：03-15-2016
Print "the current date is: "; DATE        '列印結果：the current date is: 03-15-2016
```

2.4.19 TIME

語　法：value=TIME

類　型：String

描　述：獲取當前控制器日期：小時 ： 分鐘 ： 秒

例　程

```
DIM str1 AS string
str1=TIME
print str1        '如現在時間是 14 點 22 分 51 秒，列印結果為：14:22:51
Print "the current time is: "; TIME          '列印結果：the current date is: 14:22:51
```

2.4.20 SETDATE

語　法：SETDATE(newdate)

描　述：給控制器系統設置新的日期

參　數：newdate 新的日期，類型為 string

例　程

```
SETDATE "03/15/2016"  '將當前控制器系統日期設置為 2016 年 3 月 15 號
```

2.4.21 SETTIME

語　法：SETTIME(newtime)

描　述：給控制器系統設置新的時間

參　數：newtime 新的時間，類型為 string

例　程

```
SETTIME "14:20:31"      '將當前控制器系統時間設置為 14 點 20 分 31 秒
```

2.4.22 TIMER

語　法：value=TIMER

描　述：以秒為單位，返回開始參考的時間到現在消逝的時間總和。該功能主要用於計算 TASK 裡一個程式段跑了多少時間，從而去找出從程式一行執行到另一行所花的時間。

例　程

```
Dim Start As Double
Print "Wait 2.5 seconds."
Start = Timer
Do
    Sleep 1, 1
Loop Until (Timer – Start) > 2.5
Print "Done."
```

2.5 單軸點位運動

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.5.1	VL	單軸初速度	√	√
2.5.2	VH	單軸運行速度	√	√
2.5.3	ACC	單軸加速度	√	√
2.5.4	DEC	單軸減速度	√	√
2.5.5	JK	單軸速度曲線類型	√	√
2.5.6	DSPEED	當前軸指令速度值	√	√
2.5.7	STATE	軸當前運動狀態	√	√
2.5.8	MOVE	單軸相對點位運動	√	×
2.5.9	MOVEABS	單軸絕對點位運動	√	×
2.5.10	PCHANGE	單軸運動過程中改變終點位置	√	×
2.5.11	STOPDEC	減速停止	√	×
2.5.12	STOPEMG	立即停止	√	×
2.5.13	BACKLASH_EN	背隙補償功能使能	√	×
2.5.14	BACKLASH_PULSE	背隙補償距離	√	√
2.5.15	BACKLASH_VEL	背隙補償過程運動速度	√	√
2.5.16	MAXVEL	單軸最大速度限值	√	√
2.5.17	MAXACC	單軸最大加速度限值	√	√
2.5.18	MAXDEC	單軸最大減速度限值	√	√
2.5.19	RESETERR	清除當前軸錯誤狀態	√	×

2.5.1 VL

所　屬：屬性

語　法：VL = value

類　型：DOUBLE

描　述：設置/讀取軸的初速度，VL 的單位為脈衝當量/s

範　圍：【0, MAXVEL】，預設值 2000

例　程

```
BASE 0

VL=2000 '設置軸 0 的初速度為 2000 個脈衝當量/s
```

2.5.2 VH

所　屬：屬性

語　法：VH = value

類　型：DOUBLE

描　述：設置/讀取軸的最大運行速度，VH 的單位為脈衝當量/s

範　圍：（VL, MAXVEL），預設值 8000

例　程

```
BASE 0

VH=2000 '設置軸 0 的運行速度為 2000 個脈衝當量/s
```

2.5.3 ACC

所　屬：屬性

語　法：ACC = value

類　型：DOUBLE

描　述：設置/讀取軸的加速度，ACC 的單位為脈衝當量/s^2

範　圍：（0, MAXACC），預設值 10000

例　程

```
BASE 0

ACC=20000 '設置軸 0 的加速度為 20000 個脈衝當量/s^2
```

2.5.4 DEC

所　屬：屬性

語　法：DEC = value

類　型：DOUBLE

描　述：設置/讀取軸的減速度，DEC 的單位為脈衝當量/s^2

範　圍：（0, MAXDEC），預設值 10000

例　程

```
BASE 0
DEC=20000 '設置軸 0 的減速度為 20000 個脈衝當量/s^2
```

2.5.5 JK

所　屬：屬性

語　法：JK = value

類　型：ULONG

描　述：設置/讀取單軸點位元運動的速度曲線類型

範　圍：【0,1】，0：T 型曲線；1：S 型曲線，預設值 0

例　程

```
BASE 0
JK=1        '設置軸 0 單軸點位元運動的速度曲線類型為 S 型曲線
```

2.5.6 DSPEED

所　屬：屬性（唯讀）

語　法：value = DSPEED

類　型：DOUBLE

描　述：根據當前 BASE 清單中的軸，讀取軸當前指令理論速度

例　程

```
BASE 0
DIM A AS DOUBLE
A=DSPEED    '將軸 0 的當前指令速度賦值給變數 A
```

2.5.7 STATE

所　屬：屬性(唯讀)

語　法：value = STATE

類　型：ULONG

描　述：讀取當前軸運動狀態。

返回值：如下

0：軸不可用狀態

1：Ready 狀態

2：軸停止狀態，但未 Ready

3：軸處於錯誤狀態，軸被停止運動

4：軸正在執行回原點運動中

5：軸正在執行單軸點位運動中

6：軸正在執行單軸連續運動中

7：軸正在參與插補運動中或同步運動中

8：軸處於外部 JOG 模式中

9：軸處於外部 MPG 模式中

例　程

```
BASE 0
DIM A AS ULONG
A=STATE    '將軸 0 的當前軸運動狀態對應的值賦值給變數 A
A=STATE(AX(2))    '將軸 2 的當前軸運動狀態對應的值賦值給變數 A。A=STATE AX(2)這種語法不對。
```

2.5.8 MOVE

所　屬：命令

語　法 1：MOVE　distance1[,distance2][,distance3].....

語　法 2：MOVE AX(axis no)，distance

描　述：BASE 軸列表的軸，以當前位置為原點，在相對座標下運動到指定距離的位置。

參　數：distance　相對移動距離；類型：DOUBLE

axis no　軸號；　範圍：根據控制器實際硬體決定。

例　程

```
'選擇要操作的軸, 並設定速度等相關參數
BASE 0,1,2
VL=2000
VH=8000
ACC=10000
DEC=10000
'開始軸 0 的相對運動
BASE 0
```

```
MOVE 1000
WAIT DONE
'開始軸 2,3 的相對運動
BASE 1,2
MOVE 2000, 3000
WAIT DONE
'指定軸 1，開始相對運動
MOVE AX(1),5000
```

2.5.9 MOVEABS

所　屬：命令

語　法 1：MOVEABS position1[，position2] [，position3]……

語　法 2：MOVEABS AX(axis no)，positon

描　述： BASE 軸列表的軸，在絕對座標下運動到的指定位置。

參　數：position　絕對位置；類型：DOUBLE

axis no　軸號；　範圍：根據控制器實際硬體決定。

例　程

```
'選擇要操作的軸, 並設定速度等相關參數
BASE 0,1,2
VL=2000
VH=8000
ACC=10000
DEC=10000
'開始軸 0 的絕對運動
BASE 0
MOVEABS 1000
WAIT DONE
'開始軸 2,3 的絕對運動
BASE 1,2
MOVEABS 2000, 3000
WAIT DONE
'指定軸 1，開始絕對運動
```

```
MOVEABS AX(1),5000
```

2.5.10 PCHANGE

所　屬：命令

語　法 1：PCHANGE distance

語　法 2：PCHANGE AX(axis no) ,distance

描　述： 修改當前運動的終點位置，如果當前軸不在運動中，下該指令會報錯。

參　數：distance　改變相對運動距離；類型：DOUBLE

　　axis no　軸號； 範圍：根據控制器實際硬體決定。

例　程

```
BASE 0
MOVE 20000
SLEEP 200
PCHANGE 25000          '改變位移為 25000
WAIT DONE
BASE 0,1
SVON
MOVE 10000,10000
SLEEP 200
PCHANGE AX(1),5000   '改變位移為 5000
WAIT DONE
```

2.5.11 STOPDEC

所　屬：命令

語　法 1：STOPDEC

語　法 2：STOPDEC　AX(axis no)，dec

語　法 3：STOPDEC　dec1，dec2，dec3，……，dec32

描　述： BASE 軸列表中的軸或指定軸、指定減速度對軸下減速停止運動命令

參　數：dec　減速度；類型：DOUBLE

　　axis no　軸號； 範圍：根據控制器實際硬體決定。

例　程

```
BASE 0,1,2
SVON
VL=1000
VH=10000
ACC=50000
DEC=50000
'多個軸下減速停止運動命令
MOVE 40000,20000,35000
SLEEP 1000
STOPDEC
WAIT DONE
'指定軸下減速停止運動命令
FORWARD AX(0)
SLEEP 2000
STOPDEC AX(0)
WAIT AX(0),DONE
'指定多個軸用新的減速度下減速停止運動命令
MOVE 40000,30000,35000
SLEEP 2000
STOPDEC 200000,200000,200000
```

2.5.12 STOPEMG

所　屬：命令

語　法 1：STOPEMG

語　法 2：STOPEMG　AX(axis no)

描　述： BASE 軸列表中的軸或指定軸對軸下立即停止運動命令

參　數： axis no　軸號；　範圍：根據控制器實際硬體決定。

例　程

```
BASE 0,1,2
SVON
```

```
VL=1000
VH=10000
ACC=50000
DEC=50000
'多個軸下立即停止運動命令
MOVE 40000,20000,35000
SLEEP 1000
STOPEMG
WAIT DONE
'指定軸下立即停止運動命令
FORWARD AX(0)
SLEEP 2000
STOPEMG AX(0)
WAIT AX(0),DONE
```

2.5.13 BACKLASH_EN

所　屬：屬性

語　法：BACKLASH_EN = value

類　型：ULONG

描　述：啟用/禁用背隙補償功能

範　圍：如下設定值，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
BACKLASH_EN=1   '啟用軸 0 的背隙補償功能
```

2.5.14 BACKLASH_PULSE

所　屬：屬性

語　法：BACKLASH_PULSE = value

類　型：ULONG

描　述：設置/讀取背隙補償的脈衝個數

範　圍：【0, 4095】，預設值 10

例　程

```
BASE 0

BACKLASH_PULSE=10 '設置軸 0 的背隙補償的脈衝個數為 10 個
```

2.5.15 BACKLASH_VEL

所　屬：屬性

語　法：BACKLASH_VEL = value

類　型：ULONG

描　述：設置/讀取進行補償距離移動時的速度，單位為脈衝/s

範　圍：（0, MAXVEL），預設值 1000

例　程

```
BASE 0

BACKLASH_VEL=1000 '設置軸 0 的背隙補償速度為 1000 個脈衝/s
```

2.5.16 MAXVEL

所　屬：屬性

語　法：MAXVEL = value

類　型：DOUBLE

描　述：設定/讀取運動軸的運行速度限值，單位為脈衝當量/s 。VL、VH、HOME_VH 等跟軸速度相關的設定值都不能超過該限值，否則設定會不成功。

範　圍：【1,5000000】，預設值 1000000

例　程

```
MAXVEL=200000 '設置軸的最大運行速度限值為 200000 脈衝當量/s
```

2.5.17 MAXACC

所　屬：屬性

語　法：MAXACC = value

類　型：DOUBLE

描　述：設定/讀取運動軸的加速度限值，單位為脈衝當量/s^2 。ACC、HOME_ACC 等跟軸加速度相關的設定值都不能超過該限值，否則設定會不成功。

範　圍：【1,500000000】，預設值 500000000

例　程

```
MAXACC=200000 '設置軸的加速度限值為 200000 脈衝當量/s^2
```

2.5.18 MAXDEC

所　屬：屬性

語　法：MAXDEC = value

類　型：DOUBLE

描　述：設定/讀取運動軸的減速度限值，單位為脈衝當量/s^2 。DEC、HOME_DEC 等跟軸減速度相關的設定值都不能超過該限值，否則設定會不成功。

範　圍：【1,500000000】，預設值 500000000

例　程

```
MAXDEC=200000 '設置軸的減速度限值為 200000 脈衝當量/s^2
```

2.5.19 RESETERR

所　屬：命令

語　法 1：RESETERR

語　法 2：RESETERR AX(axis no)

描　述：BASE 軸清單的軸或指定軸，清除軸錯誤

參　數：axis no　軸號；　範圍：根據控制器實際硬體決定。

例　程

```
BASE 0,1,2
RESETERR    '清除軸 0、1、2 的錯誤
RESETERR AX(1)   '清除軸 1 的錯誤
```

2.6 單軸定速運動

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.6.1	FORWARD	正向恒速連續運動	√	×
2.6.2	REVERSE	反向恒速連續運動	√	×
2.6.3	VCHANGE	運動中改變速度	√	×
2.6.4	VCHANGE_RATE	運動中改變速度（百分比）	√	×

2.6.1 FORWARD

所　屬：命令

語　法 1：FORWARD

語　法 2：FORWARD AX(axis no)

語　法 3：FORWARD dir1[，dir2][，dir3]……　dir 為 0 時，方向與 FORWARD 同向 dir 為 1 時，方向與 FORWARD 反向

描　述： BASE 軸列表的軸或指定軸，開始正向連續運動

參　數：dir　方向；類型：ULONG

axis no　軸號； 範圍：根據控制器實際硬體決定。

例　程

```
BASE 0,1
SVON
VL=1000                 '設置初速度
VH=10000                '設置運行速度
ACC=100000              '設置加速度
DEC=ACC                 '設置減速度
'單軸或多軸同方向執行連續運動
FORWARD     '軸 0,1 都執行正向連續運動
SLEEP 2000      '延時 2000ms
STOPDEC         '減速停止軸 0,1 運動
```

```
WAIT DONE     '等待運動停止,如運動未停止的狀態，下面語句再對該軸操作會執行不成功
REVERSE       '軸 0,1 都執行負向連續運動
SLEEP 2000
STOPDEC       '減速停止軸 0,1 運動
WAIT DONE     '等待運動停止
'指定多個軸按不同方向執行連續運動
FORWARD 0,1   '軸 0 正向連續運動，軸 1 負向連續運動
SLEEP 2000
STOPEMG       '立即停止軸 0,1 運動
WAIT DONE
'指定一個軸執行連續運動
FORWARD AX(0)   '軸 0 執行正向連續運動
SLEEP 1000
STOPDEC AX(0)   '指定軸 0 下減速停止命令
WAIT AX(0),DONE   '指定軸等待運動停止
```

2.6.2 REVERSE

所　屬：命令

語　法 1：REVERSE

語　法 2：REVERSE　AX(axis no)

語　法 3：REVERSE　dir1[，dir2][，dir3]……　dir 為 0 時，方向與 REVERSE 同向 dir 為 1 時，方向與 REVERSE 反向

描　述： BASE 軸列表的軸或指定軸，開始反向連續運動

參　數：dir　方向；類型：ULONG

axis no　軸號； 範圍：根據控制器實際硬體決定。

例　程

```
BASE 0,1
SVON
VL=1000          '設置初速度
VH=10000         '設置運行速度
```

```
ACC=100000          '設置加速度
DEC=ACC             '設置減速度
'單軸或多軸同方向執行連續運動
FORWARD     '軸 0,1 都執行正向連續運動
SLEEP 2000    '延時 2000ms
STOPDEC     '減速停止軸 0,1 運動
WAIT DONE    '等待運動停止,如運動未停止的狀態，下面語句再對該軸操作會執行不成功
REVERSE     '軸 0,1 都執行負向連續運動
SLEEP 2000
STOPDEC     '減速停止軸 0,1 運動
WAIT DONE    '等待運動停止
'指定多個軸按不同方向執行連續運動
REVERSE 0,1    '軸 0 負向連續運動，軸 1 正向連續運動
SLEEP 2000
STOPEMG     '立即停止軸 0,1 運動
WAIT DONE
'指定一個軸執行連續運動
REVERSE AX(0)    '軸 0 執行負向連續運動
SLEEP 1000
STOPDEC AX(0)    '指定軸 0 下減速停止命令
WAIT AX(0),DONE    '指定軸等待運動停止
```

2.6.3 VCHANGE

所　屬：命令

語　法 1：VCHANGE vel

語　法 2：VCHANGE AX（axis no），vel

語　法 3：VCHANGE vel, acc, dec

語　法 4：VCHANGE AX（axis no）, vel，acc, dec

描　述： BASE 軸清單的第一個軸，開始更改速度運動；或指定軸和新的速度開始更改速度運動。該指令可以對 MOVE、MOVEABS、FORWARD、REVERSE 起作用，如果當前軸不在運動中，下該指令會報錯

參　數： vel　　運行速度；類型：DOUBLE

acc　　改變速度時的加速度；類型：DOUBLE

dec　　改變速度時的減速度；類型：DOUBLE

axis no　軸號；　範圍：根據控制器實際硬體決定

例　程

```
BASE 0,1
SVON
VL=1000              '設置初速度
VH=10000             '設置運行速度
ACC=100000           '設置加速度
DEC=ACC              '設置減速度
FORWARD      '軸 0,1 都執行正向連續運動
SLEEP 2000     '延時 2000ms
VCHANGE 30000        '對 BASE 列表中第一個軸起作用，將軸 0 的速度改為 30000
SLEEP 2000
VCHANGE AX(1),5000   '將軸 1 的速度改為 50000
SLEEP 2000
VCHANGE 20000,10000,10000   '將軸 0 的速度改為 20000，加、減速度都改為 10000
SLEEP 3000
VCHANGE AX(1),20000,50000,50000 '將軸 1 的速度改為 20000，加、減速度都改為 50000
SLEEP 2000
STOPDEC
```

2.6.4 VCHANGE_RATE

所　屬：命令

語　法 **1**：VCHANGE_RATE　rate

語　法 **2**：VCHANGE_RATE　AX（axis no），rate

語　法 **3**：VCHANGE_RATE　rate, acc, dec

語　法 **4**：VCHANGE_RATE　AX（axis no）, rate，acc, dec

描　述： BASE 軸列表的第一個軸，開始按百分比更改速度運動；或指定軸和新的百分比速度開始更改速度運動。該指令可以對 MOVE、MOVEABS、FORWARD、REVERSE 起作用，如果當前軸不在運動中，下該指令會報錯

參　數： rate　　原設置速度的百分比速度；類型：DOUBLE

acc　　改變速度時的加速度；類型：DOUBLE

dec　　改變速度時的減速度；類型：DOUBLE

axis no　軸號；　範圍：根據控制器實際硬體決定

例　程

```
BASE 0,1
SVON
VL=1000              '設置初速度
VH=10000             '設置運行速度
ACC=100000           '設置加速度
DEC=ACC              '設置減速度
FORWARD              '軸 0,1 都執行正向連續運動
SLEEP 2000           '延時 2000ms
'運動中改變速度的功能應用中：新設定的速度要高於 VL
VCHANGE_RATE 200     '對 BASE 列表中第一個軸起作用，將軸 0 的速度改為設定的 VH 的 200%
SLEEP 2000
VCHANGE_RATE AX(1),30   '將軸 1 的速度改為設定的 VH 的 30%
SLEEP 2000
VCHANGE_RATE 50,10000,10000   '將軸 0 的速度改為 VH 的 50%，加、減速度都改為 10000
SLEEP 3000
VCHANGE_RATE AX(1),400,50000,50000 '將軸 1 的速度改為 VH 的 400%，加、減速度都改為 50000
SLEEP 2000
STOPDEC
```

2.7 多軸插補運動

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.7.1	GVL	插補初速度	×	×
2.7.2	GVH	插補運行速度	×	×
2.7.3	GACC	插補加速度	×	×
2.7.4	GDEC	插補減速度	×	×
2.7.5	GJK	插補速度曲線類型	×	×
2.7.6	GDSPEED	當前插補運動指令速度值	×	×
2.7.7	GSTATE	當前插補運動狀態	×	×
2.7.8	LINE	2-3 軸直線相對插補運動	×	×
2.7.9	LINEABS	2-3 軸直線絕對插補運動	×	×
2.7.10	DIRECT	2-8 軸線性相對插補運動	×	×
2.7.11	DIRECTABS	2-8 軸線性絕對插補運動	×	×
2.7.12	CIRC	2 軸相對圓弧插補（指定圓心、終點）	×	×
2.7.13	CIRCABS	2 軸絕對圓弧插補（指定圓心、終點）	×	×
2.7.14	CIRC_3P	2 軸相對圓弧插補(指定圓上 3 點）	×	×
2.7.15	CIRCABS_3P	2 軸絕對圓弧插補(指定圓上 3 點）	×	×
2.7.16	CIRC_A	2 軸相對圓弧插補（指定圓弧角度、終點）	×	×
2.7.17	CIRCABS_A	2 軸絕對圓弧插補（指定圓弧角度、終點）	×	×
2.7.18	HELIX	3 軸相對螺旋插補（指定圓弧中心、終點、高度）	×	×
2.7.19	HELIXABS	3 軸絕對螺旋插補（指定圓弧中心、終點、高度）	×	×
2.7.20	HELIX_3P	3 軸相對螺旋插補（指定螺旋線上	×	×

		3 點）		
2.7.21	HELIXABS_3P	3 軸絕對螺旋插補（指定螺旋線上 3 點）	×	×
2.7.22	HELIX_A	3 軸相對螺旋插補（指定圓弧角度、終點、高度）	×	×
2.7.23	HELIXABS_A	3 軸絕對螺旋插補（指定圓弧角度、終點、高度）	×	×
2.7.24	GPAUSE	插補運動暫停指令	×	×
2.7.25	GRESUME	插補運動暫定後恢復運動指令	×	×

2.7.1 GVL

所　屬：屬性

語　法：GVL = value

類　型：DOUBLE

描　述：設置/讀取插補運動的初速度，GVL 的單位為脈衝當量/s

範　圍：（0, MAXVEL），預設值 2000

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
GVL=2000 '設置軸 0,1 的插補運動的初速度為 2000 個脈衝當量/s
```

2.7.2 GVH

所　屬：屬性

語　法：GVH = value

類　型：DOUBLE

描　述：設置/讀取插補運動的最大運行速度，GVH 的單位為脈衝當量/s

範　圍：（GVL, MAXVEL），預設值 8000

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
GVH=10000 '設置軸 0,1 的插補運動的最大運行速度為 10000 個脈衝當量/s
```

2.7.3 GACC

所　屬：屬性

語　法：GACC = value

類　型：DOUBLE

描　述：設置/讀取插補運動的加速度，GACC 的單位為脈衝當量/s^2

範　圍：（0, MAXACC），預設值 10000

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
GACC=20000 '設置軸 0,1 的插補運動的加速度為 20000 個脈衝當量/s^2
```

2.7.4 GDEC

所　屬：屬性

語　法：GDEC = value

類　型：DOUBLE

描　述：設置/讀取插補運動的減速度，GDEC 的單位為脈衝當量/s^2

範　圍：（0, MAXDEC），預設值 10000

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
GDEC=20000 '設置軸 0,1 的插補運動的減速度為 20000 個脈衝當量/s^2
```

2.7.5 GJK

所　屬：屬性

語　法：GJK = value

類　型：ULONG

描　述：設置/讀取插補運動的速度曲線類型

範　圍：【0,1】，0：T 型曲線；1：S 型曲線，預設值 0

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
GJK=1    '設置軸 0、1 的插補運動的速度曲線類型為 S 型曲線
```

2.7.6 GDSPEED

所　屬：屬性（唯讀）

語　法：value = GDSPEED

類　型：DOUBLE

描　述：讀取當前插補指令理論速度

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
DIM A AS DOUBLE
A=GDSPEED    '將軸 0,1 插補運動的當前指令速度賦值給變數 A
```

2.7.7 GSTATE

所　屬：屬性（唯讀）

語　法：value = GSTATE

類　型：ULONG

描　述：讀取當前插補運動狀態。

返回值：如下

0：插補不可用狀態

1：插補運動處於 Ready 狀態

2：插補運動處於停止狀態，但未 Ready

3：插補運動處於錯誤狀態，插補運動被停止運動

4：BASE 軸正在執行插補運動中

5：保留

6：BASE 軸正在執行連續插補運動中

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
DIM A AS USHORT
```

```
A=GSTATE    '將軸 0、1 當前的插補運動狀態對應的值賦值給變數 A
```

2.7.8 LINE

所　屬：命令

語　法：LINE　distance1,distance2 [,distance3]

描　述：指定插補軸的移動距離，開始 2 軸或 3 軸的相對直線插補運動。LINE 指令僅支援 2 軸或 3 軸的直線插補運動，3 軸以上不支持。

參　數：distance　各軸的相對移動距離；類型：DOUBLE

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
SVON
GVL=1000              '設置插補初速度
GVH=10000             '設置插補運行速度
GACC=100000           '設置插補加速度
GDEC=GACC             '設置插補減速度
GJK=0                 '設置插補速度曲線為 T 型
 '絕對直線插補運動
LINEABS 0,5000   '運動到目標位置（0,5000）
WAIT DONE    '等待 LINEABS 運動走完
 '相對直線插補運動
LINE 8000,-15000    '軸 0、1 方向的運動距離分別為 8000、-15000
```

2.7.9 LINEABS

所　屬：命令

語　法：LINEABS　position1, position2[,position3]

描　述：指定插補軸的終點，開始 2 軸或 3 軸的絕對直線插補運動。LINEABS 指令僅支援 2 軸或 3 軸的直線插補運動，3 軸以上不支持。

參　數：position　各軸的終點位置；類型：DOUBLE

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
SVON
GVL=1000                '設置插補初速度
GVH=10000               '設置插補運行速度
GACC=100000             '設置插補加速度
GDEC=GACC               '設置插補減速度
GJK=0                   '設置插補速度曲線為 T 型
 '絕對直線插補運動
LINEABS 0,5000   '運動到目標位置（0,5000）
WAIT DONE     '等待 LINEABS 運動走完
 '相對直線插補運動
LINE 8000,-15000     '軸 0、1 方向的運動距離分別為 8000、-15000
```

2.7.10 DIRECT

所　屬：命令

語　法：DIRECT　distance1,distance2[,distance3]... [,distance8]

描　述：指定插補軸的移動距離，開始 2 軸-8 軸的相對線性插補運動。最多支援到 8 個軸的 DIRECT 線性插補運動

參　數：distance　各軸的相對移動距離；類型：DOUBLE

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1,2,3
SVON
'DIRECT 插補運動中，以下速度、加減速參數設置的是參與插補運動中移動距離最長軸的參數
GVL=1000                '設置插補初速度
GVH=10000               '設置插補運行速度
GACC=100000             '設置插補加速度
GDEC=GACC                '設置插補減速度
GJK=0                   '設置插補速度曲線為 T 型
'絕對線性插補
```

```
DIRECTABS 0,5000,-500,1000    '運動到目標位置（0,5000,-500,1000）
WAIT DONE                     '等待 DIRECTABS 運動走完
'相對線性插補
DIRECT 8000,-15000,0,2000     '軸 0、1、2、3 方向的運動距離分別為 8000、-15000、0、2000
```

2.7.11 DIRECTABS

所　屬：命令

語　法：DIRECTABS　position1, position2[,position3]... [,position8]

描　述：指定插補軸的終點，開始 2 軸-8 軸的絕對線性插補運動。最多支援到 8 個軸的 DIRECTABS 線性插補運動

參　數：position　各軸的終點位置；類型：DOUBLE

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1,2,3
SVON
'DIRECT 插補運動中，以下速度、加減速參數設置的是參與插補運動中移動距離最長軸的參數
GVL=1000              '設置插補初速度
GVH=10000             '設置插補運行速度
GACC=100000           '設置插補加速度
GDEC=GACC             '設置插補減速度
GJK=0                 '設置插補速度曲線為 T 型
'絕對線性插補
DIRECTABS 0,5000,-500,1000    '運動到目標位置（0,5000,-500,1000）
WAIT DONE                     '等待 DIRECTABS 運動走完
'相對線性插補
DIRECT 8000,-15000,0,2000     '軸 0、1、2、3 方向的運動距離分別為 8000、-15000、0、2000
```

2.7.12 CIRC

所　屬：命令

語　法：CIRC　dir,center1,center2,end1,end2

描　述：指定圓方向、圓心、終點，開始兩軸的相對圓弧插補運動

參　數：dir　圓弧運動方向：0-順時針；1-逆時針　；類型：ULONG

center1　第一個軸圓心的相對座標　；類型：DOUBLE

center2　第二個軸圓心的相對座標；類型：DOUBLE

end1　第一個軸圓弧終點的相對座標；類型：DOUBLE

end2　第二個軸圓弧終點的相對座標；類型：DOUBLE

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
GVL=1000            '設置插補初速度
GVH=10000           '設置插補運行速度
GACC=100000     '設置插補加速度
GDEC=GACC           '設置插補減速度
GJK=0                   '設置插補速度曲線為 T 型
DPOS=0              '當前理論位置清零
SVON
'執行相對圓弧插補運動
CIRC 0,5000,0,10000,0       '順時針，圓心相對距離為（5000,0）,圓弧終點相對距離為（10000,0）
WAIT DONE
'執行絕對圓弧插補運動
CIRCABS 1,5000,0,0,0        '逆時針，圓心位置為（5000,0）,圓弧終點位置為（0,0）
WAIT DONE
```

2.7.13 CIRCABS

所　屬：命令

語　法：CIRCABS　dir,center1,center2,end1,end2

描　述：指定圓方向、圓心、終點，開始兩軸的絕對圓弧插補運動

參　數：dir 圓弧運動方向：0-順時針；1-逆時針　；類型：ULONG

center1　第一個軸圓心的絕對座標　；類型：DOUBLE

center2　第二個軸圓心的絕對座標；類型：DOUBLE

end1　第一個軸圓弧終點的絕對座標；類型：DOUBLE

end2　　第二個軸圓弧終點的絕對座標；類型：DOUBLE

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
GVL=1000            '設置插補初速度
GVH=10000           '設置插補運行速度
GACC=100000     '設置插補加速度
GDEC=GACC           '設置插補減速度
GJK=0                   '設置插補速度曲線為 T 型
DPOS=0              '當前理論位置清零
SVON
'執行相對圓弧插補運動
CIRC 0,5000,0,10000,0       '順時針，圓心相對距離為（5000,0）,圓弧終點相對距離為（10000,0）
WAIT DONE
'執行絕對圓弧插補運動
CIRCABS 1,5000,0,0,0        '逆時針，圓心位置為（5000,0）,圓弧終點位置為（0,0）
WAIT DONE
```

2.7.14 CIRC_3P

所　屬：命令

語　法：CIRC_3P　dir,ref1,ref2,end1,end2

描　述：指定圓方向和圓上 3 點，開始兩軸的相對圓弧插補運動

參　數：dir 圓弧運動方向：0-順時針；1-逆時針　；類型：ULONG

ref1　第一個軸中間點的相對座標　；類型：DOUBLE

ref2　第二個軸中間點的相對座標；類型：DOUBLE

end1　第一個軸圓弧終點的相對座標；類型：DOUBLE

end2　第二個軸圓弧終點的相對座標；類型：DOUBLE

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
```

```
GVL=1000          '設置插補初速度
GVH=10000         '設置插補運行速度
GACC=100000       '設置插補加速度
GDEC=GACC         '設置插補減速度
GJK=0             '設置插補速度曲線為 T 型
DPOS=0            '當前理論位置清零
SVON
'執行相對圓弧插補運動
CIRC_3P  0,10000,10000,20000,0      '順時針，圓上第二點相對距離為（10000,10000）,圓弧終點相對距離為（20000,0）
WAIT DONE
'執行絕對圓弧插補運動
CIRCABS_3P 1,10000,10000,0,0        '逆時針，圓心第二點位置為（10000,10000）,圓弧終點位置為（0,0）
WAIT DONE
```

2.7.15 CIRCABS_3P

所　屬：命令

語　法：CIRC_3P　dir,ref1,ref2,end1,end2

描　述：指定圓方向和圓上 3 點，開始兩軸的絕對圓弧插補運動

參　數：dir 圓弧運動方向：0-順時針；1-逆時針　；類型：ULONG

ref1　第一個軸中間點的絕對座標　；類型：DOUBLE

ref2　第二個軸中間點的絕對座標；類型：DOUBLE

end1　第一個軸圓弧終點的絕對座標；類型：DOUBLE

end2　第二個軸圓弧終點的絕對座標；類型：DOUBLE

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
GVL=1000          '設置插補初速度
GVH=10000         '設置插補運行速度
GACC=100000       '設置插補加速度
```

```
GDEC=GACC          '設置插補減速度
GJK=0              '設置插補速度曲線為 T 型
DPOS=0             '當前理論位置清零
SVON
'執行相對圓弧插補運動
CIRC_3P  0,10000,10000,20000,0      '順時針，圓上第二點相對距離為（10000,10000）,圓弧終點相對距離為（20000,0）
WAIT DONE
'執行絕對圓弧插補運動
CIRCABS_3P 1,10000,10000,0,0        '逆時針，圓心第二點位置為（10000,10000）,圓弧終點位置為（0,0）
WAIT DONE
```

2.7.16 CIRC_A

所　屬：命令

語　法：CIRC_A　dir,center1,center2,degree

描　述：指定圓方向、圓心、角度，開始兩軸的相對圓弧插補運動

參　數：dir 圓弧運動方向：0-順時針；1-逆時針　；類型：ULONG

center1　第一個軸圓心的相對座標　；類型：DOUBLE

center2　第二個軸圓心的相對座標；類型：DOUBLE

degree　圓弧角度，單位為角度；類型：DOUBLE

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
GVL=1000           '設置插補初速度
GVH=10000          '設置插補運行速度
GACC=100000    '設置插補加速度
GDEC=GACC          '設置插補減速度
GJK=0              '設置插補速度曲線為 T 型
DPOS=0             '當前理論位置清零
SVON
```

```
'執行相對圓弧插補運動
CIRC_A 0,10000,0,180      '順時針，圓心相對距離為（10000,0）,圓弧角度為 180 度
WAIT DONE
'執行絕對圓弧插補運動
CIRCABS_A 1,10000,0,90      '逆時針，圓心位置為（10000,0）,圓弧角度為 90 度
WAIT DONE
```

2.7.17 CIRCABS_A

所　屬：命令

語　法：CIRC_A　dir,center1,center2,degree

描　述：指定圓方向、圓心、角度，開始兩軸的絕對圓弧插補運動

參　數：dir 圓弧運動方向：0-順時針；1-逆時針　；類型：ULONG

center1　第一個軸圓心的絕對座標　；類型：DOUBLE

center2　第二個軸圓心的絕對座標；類型：DOUBLE

degree　圓弧角度，單位為角度；類型：DOUBLE

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
GVL=1000            '設置插補初速度
GVH=10000           '設置插補運行速度
GACC=100000     '設置插補加速度
GDEC=GACC           '設置插補減速度
GJK=0               '設置插補速度曲線為 T 型
DPOS=0              '當前理論位置清零
SVON
'執行相對圓弧插補運動
CIRC_A 0,10000,0,180      '順時針，圓心相對距離為（10000,0）,圓弧角度為 180 度
WAIT DONE
'執行絕對圓弧插補運動
CIRCABS_A 1,10000,0,90      '逆時針，圓心位置為（10000,0）,圓弧角度為 90 度
```

WAIT DONE

2.7.18 HELIX

所　屬：命令

語 法：HELIX　dir,center1,center2,end1,end2,distance

描　述：指定螺旋旋轉方向、中心、螺旋高度，開始 3 軸的相對螺旋插補運動

參　數：dir　螺旋運動方向：0：　順時針；1：逆時針

center1　第一個軸圓心的相對座標　；

center2　第二個軸圓心的相對座標；

end1　第一個軸螺旋終點的相對座標；

end2　第二個軸螺旋終點的相對座標；

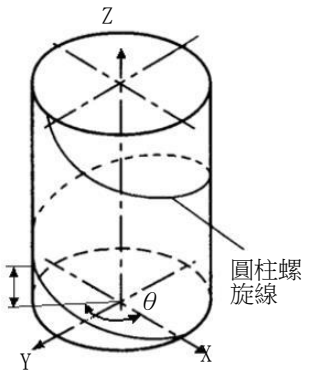
distance　螺旋高度，單位為脈衝當量

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1,2
GVL=1000            '設置插補初速度
GVH=10000           '設置插補運行速度
GACC=100000         '設置插補加速度
GDEC=GACC           '設置插補減速度
GJK=0               '設置插補速度曲線為 T 型
DPOS=0              '當前理論位置清零
MPOS=0              '當前回饋位置清零
SVON
'相對螺旋插補，順時針，圓心相對座標為（5000,0）,圓弧終點相對座標為（10000,0）,螺旋高度為 5000
HELIX 0,5000,0,10000,0,5000
WAIT DONE

'絕對螺旋插補，逆時針，圓心絕對座標為（5000,0）,圓弧終點絕對座標為（0,0）,螺旋 Z 軸終點位置為 0
HELIXABS 1,5000,0,0,0,0
```



```
WAIT DONE
```

2.7.19 HELIXABS

所　屬：命令

語　法：HELIXABS　dir,center1,center2,end1,end2,position

描　述：指定螺旋方向、中心、終點，開始 3 軸的絕對螺旋插補運動

參　數：dir　螺旋運動方向：0：　順時針；1：逆時針

center1　第一個軸圓心的絕對座標

center2　第二個軸圓心的絕對座標

end1　第一個軸螺旋終點的絕對座標

end2　第二個軸螺旋終點的絕對座標

position　第三個軸終點位置座標

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1,2
GVL=1000            '設置插補初速度
GVH=10000           '設置插補運行速度
GACC=100000         '設置插補加速度
GDEC=GACC           '設置插補減速度
GJK=0               '設置插補速度曲線為 T 型
DPOS=0              '當前理論位置清零
MPOS=0              '當前回饋位置清零
SVON
'相對螺旋插補，順時針，圓心相對座標為（5000,0）,圓弧終點相對座標為（10000,0）,螺旋高度為 5000
HELIX 0,5000,0,10000,0,5000
WAIT DONE
'絕對螺旋插補，逆時針，圓心絕對座標為（5000,0）,圓弧終點位置為（0,0）,螺旋 Z 軸終點位置為 0
HELIXABS 1,5000,0,0,0,0
WAIT DONE
```

2.7.20 HELIX_3P

所　屬：命令

語　法：HELIX_3P　dir,ref1,ref2,ref3,end1,end2,end3

描　述：指定螺旋方向和螺旋線上的 3 點，開始 3 軸的相對螺旋插補運動

參　數：　dir　螺旋運動方向：0：　順時針；1：逆時針

ref1　第一個軸中間點的相對座標　；

ref2　第二個軸中間點的相對座標；

ref3　第三個軸中間點的相對座標；

end1　第一個軸螺旋終點的相對座標；

end2　第二個軸螺旋終點的相對座標；

end3　第三個軸螺旋終點的相對座標；

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1,2
GVL=1000            '設置插補初速度
GVH=10000           '設置插補運行速度
GACC=100000         '設置插補加速度
GDEC=GACC           '設置插補減速度
GJK=0               '設置插補速度曲線為 T 型
DPOS=0              '當前理論位置清零
MPOS=0              '當前回饋位置清零
SVON
'相對螺旋插補。順時針，中間參考點相對座標為（0,5000,0），終點相對座標為（10000,0,5000）
HELIX_3P 0,0,5000,0,10000,0,5000
WAIT DONE
'絕對螺旋插補。逆時針，中間參考點絕對座標為（0,5000,2500），終點絕對座標為（0,0,0）
HELIXABS_3P 1,0,5000,2500,0,0,0
WAIT DONE
```

2.7.21 HELIXABS_3P

所　屬：命令

語　法：HELIXABS_3P　dir,ref1,ref2,ref3,end1,end2,end3

描　述：指定螺旋方向和螺旋線上的 3 點，開始 3 軸的絕對螺旋插補運動

參　數：　dir　螺旋運動方向：0：　順時針；1：逆時針

ref1　第一個軸中間點的絕對座標　；

ref2　第二個軸中間點的絕對座標；

ref3　第三個軸中間點的絕對座標；

end1　第一個軸螺旋終點的絕對座標；

end2　第二個軸螺旋終點的絕對座標；

end3　第三個軸螺旋終點的絕對座標；

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1,2
GVL=1000            '設置插補初速度
GVH=10000           '設置插補運行速度
GACC=100000         '設置插補加速度
GDEC=GACC           '設置插補減速度
GJK=0               '設置插補速度曲線為 T 型
DPOS=0              '當前理論位置清零
MPOS=0              '當前回饋位置清零
SVON
'相對螺旋插補。順時針，中間參考點相對座標為（0,5000,0），終點相對座標為（10000,0,5000）
HELIX_3P 0,0,5000,0,10000,0,5000
WAIT DONE
'絕對螺旋插補。逆時針，中間參考點絕對座標為（0,5000,2500），終點絕對座標為（0,0,0）
HELIXABS_3P 1,0,5000,2500,0,0,0
WAIT DONE
```

2.7.22 HELIX_A

所　屬：命令

語　法：HELIX_A　dir,center1,center2,degree,distance

描　述：指定螺旋方向、螺旋圓面上的旋轉角度、螺旋高度，開始 3 軸的相對螺旋插補運動

參　數：dir　螺旋運動方向：0：　順時針；1：逆時針

center1　第一個軸圓心的相對座標

center2　第二個軸圓心的相對座標

degree　螺旋線形成的圓柱截面上的投影圓弧運動的圓心角度

distance　螺旋高度，單位為脈衝當量

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1,2
GVL=1000          '設置插補初速度
GVH=10000         '設置插補運行速度
GACC=100000       '設置插補加速度
GDEC=GACC         '設置插補減速度
GJK=0             '設置插補速度曲線為 T 型
DPOS=0            '當前理論位置清零
MPOS=0            '當前回饋位置清零
SVON
'相對螺旋插補，順時針，圓心相對座標為（5000,0）,運動的圓心角度為 180 度，,螺旋高度為 5000
HELIX_A 0,5000,0,180,5000
WAIT DONE
'絕對螺旋插補，逆時針，圓心絕對座標為（5000,0）,運動的圓心角度為 180 度，,螺旋 Z 軸終點位置為 0
HELIXABS_A 1,5000,0,180,0
WAIT DONE
```

2.7.23 HELIXABS_A

所　屬：命令

語　法：HELIXABS_A　dir,center1,center2,degree,position

描　述：指定螺旋方向、螺旋圓面上的旋轉角度、終點位置，開始 3 軸的絕對螺旋插補運動

參　數：dir　螺旋運動方向：0：　順時針；1：逆時針

center1　第一個軸圓心的絕對座標

center2　第二個軸圓心的絕對座標

degree　螺旋線形成的圓柱截面上的投影圓弧運動的圓心角度

position　第三個軸終點位置座標

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1,2
GVL=1000            '設置插補初速度
GVH=10000           '設置插補運行速度
GACC=100000         '設置插補加速度
GDEC=GACC           '設置插補減速度
GJK=0               '設置插補速度曲線為 T 型
DPOS=0              '當前理論位置清零
MPOS=0              '當前回饋位置清零
SVON
'相對螺旋插補，順時針，圓心相對座標為（5000,0）,運動的圓心角度為 180 度，,螺旋高度為 5000
HELIX_A 0,5000,0,180,5000
WAIT DONE
'絕對螺旋插補，逆時針，圓心絕對座標為（5000,0）,運動的圓心角度為 180 度，,螺旋 Z 軸終點位置為 0
HELIXABS_A 1,5000,0,180,0
WAIT DONE
```

2.7.24 GPAUSE

所　屬：命令

語　法：GPAUSE

描　述：暫定當前 TASK 的插補運動。該指令對插補運動和連續插補運動都起作用。

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
LINE 11000,20000
```

```
SLEEP 500
GPAUSE          '暫停插補運動
IF(DIN(1)=1) THEN
GRESUME         '恢復插補運動，繼續執行未執行的插補運動
END IF
```

2.7.25 GRESUME

所　屬：命令

語　法：GRESUME

描　述：恢復當前 TASK 的插補運動暫定狀態。該指令對插補運動和連續插補運動都起作用。

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
LINE 11000,20000
SLEEP 500
GPAUSE          '暫停插補運動
IF(DIN(1)=1) THEN '如果 DIN(1)為 1 時，恢復插補運動
GRESUME         '恢復插補運動，繼續執行未執行的插補運動
END IF
```

2.8 多軸連續插補運動

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.8.1	PATHBEGIN	開始進入連續插補狀態	×	×
2.8.2	PATHEND	結束連續插補狀態路徑緩存添加	×	×
2.8.3	MERGEON	用 fly mode 交接模式執行連續插補運動	×	×
2.8.4	MERGEOFF	用buffer mode交接模式執行連續插補運動	×	×
2.8.5	DELAY	連續插補路徑中的延時段指令	×	×
2.8.6	PATHRESET	清除連續插補路徑緩存中的路徑資料	×	×
2.8.7	PATH_Status	讀取連續插補運動中相關參數	×	×

2.8.1 PATHBEGIN

所　屬：命令

語　法：PATHBEGIN [num]

描　述：指定路徑緩存裡添加多少段插補指令後，開始進入連續插補模式。Num 值不填或 0 時，會將 PATHBEGIN 與 PATHEND 間所有的段都添加到緩存區裡，再開始執行連續插補運動。

參　數：num 預先添加到連續插補緩存區段數；類型：ULONG；範圍【0,10000】

注　意：連續插補段不允許存在點位元運動指令的段

該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

```
BASE 0,1
SVON
GVL=1000              '設置插補初速度
GVH=10000             '設置插補運行速度
GACC=100000           '設置插補加速度
```

```
GDEC=ACC              '設置插補減速度
LINEABS 0,0           '運動到目標位置（0,0）
WAIT DONE
PATHRESET             '清除路徑緩存
'PATHBEGIN 開始到 PATHEND 之前的路徑段為連續插補運動
'PATHBEGIN 後面跟的編號不寫或寫 0 時，表明路徑添加完再開始執行連續插補，其他數值代表添加該數值段後開始執行運動
PATHBEGIN     '進入連續插補狀態
MERGEON       '連續插補速度交接模式設置為 fly mode
CIRCABS 1,10000,0,10000,-10000   '圓弧插補段
LINEABS 25000,-10000             '直線插補段
DELAY=500                        '延時段，延時 500ms
CIRCABS 1,25000,0,35000,0        '圓弧插補段
CIRCABS 1,25000,0,25000,10000    '圓弧插補段
LINEABS 10000,10000              '直線插補段
CIRCABS 1,10000,0,0,0            '圓弧插補段
PATHEND       '連續插補路徑段結束指令，退出連續狀態
```

2.8.2 PATHEND

所　屬：命令

語　法：PATHEND

描　述：PATHBEGIN 對應的結束指令，PATHBEGIN 和 PATHEND 之間用於路徑緩存添加設定。

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

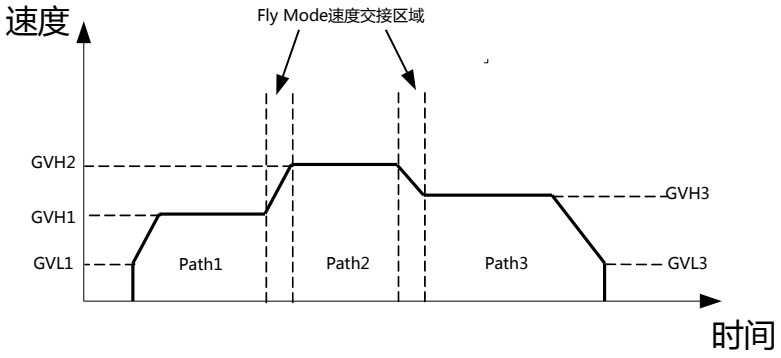
請參考 PATHBEGIN 指令中例程。

2.8.3 MERGEON

所　屬：命令

語　法：MERGEON

描　述：用 fly mode 速度交接模式執行連續插補運動。fly mode 速度交接模式，是前一段路徑的 VH 加速或減速到後一段路徑的 VH 的速度交接方式。



注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

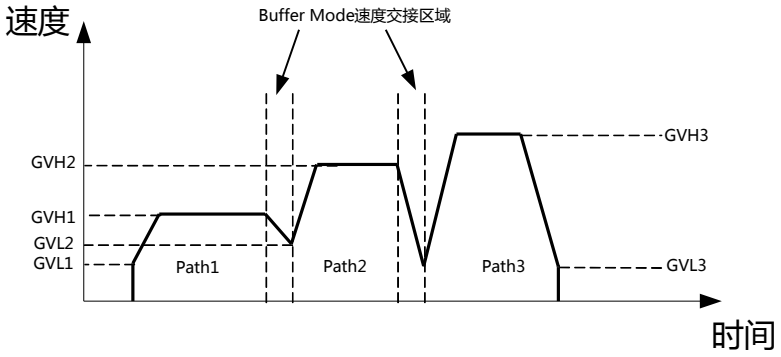
請參考 PATHBEGIN 指令中例程。

2.8.4 MERGEOFF

所　屬：命令

語　法：MERGEOFF

描　述：用 buffer mode 速度交接模式執行連續插補運動。buffer mode 速度點交接模式，是前一段路徑的 VH 加速或減速到後一段路徑的 VL 的速度交接方式。



注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

請參考 PATHBEGIN 指令中例程。

2.8.5 DELAY

所　屬：命令

語　法：DELAY= time

描　述：連續插補中，延時段指令。表示 DELAY 指令上一段完成與 DELAY 指令下一段開始間延時的事件，該延時時間非常精准，單位為 ms。

注　意：該指令不能在 Motion Studio 中的“終端”和“觀察變數”工具中使用

例　程

請參考 PATHBEGIN 指令中例程。

2.8.6 PATHRESET

所　屬：命令

語　法：PATHRESET

描　述：清除連續插補路徑緩存中的路徑資料。

注　意：PATHBEGIN 與 PATHEND 之間的路徑段為路徑緩存中的路徑。如果連續插補執行過程中被停止，下次執行連續插補將從路徑緩存中上次剩餘的未執行路徑開始執行。如果使用者不想從剩餘未執行的路徑段開始執行連續插補，則需先用 PATHRESET 指令清除路徑緩存。然後重新添加需執行的路徑到緩存中，再執行連續插補。

2.8.7 PATH_Status

所　屬：命令

語　法：PATH_Status　RemainPath,FreeBuffer [,curIndex] [,curCmd]

描　述：讀取連續插補運動中相關參數。

參　數：RemainPath　剩餘未執行的路徑段　類型：ULONG

FreeBuffer　總路徑緩存中的剩餘緩存數　類型：ULONG

curIndex　當前執行的路徑索引　類型：ULONG

curCmd　當前執行的路徑指令　類型：ULONG

curCmd 的枚舉如下：

0：EndPath ；

1：Abs2DLine ；

2：Rel2DLine ；

3：Abs2DArcCW ；

4：Abs2DArcCCW ；

5：Rel2DArcCW ；

6：Rel2DArcCCW ；

7：Abs3DLine ；

8：Rel3DLine ；

9：Abs4DLine ；（不支持）

10：Rel4DLine ；（不支持）

11：Abs2DDirect ；

12：Rel2DDirect ；

13：Abs3DDirect ；

14：Rel3DDirect ；

15：Abs4DDirect ；

16：Rel4DDirect ；

17：Abs5DDirect ；

18：Rel5DDirect ；

19：Abs6DDirect ；

20：Rel6DDirect ；

21：Abs3DArcCW ；（不支持）

22：Rel3DArcCW ；（不支持）

23：Abs3DArcCCW ；（不支持）

24：Rel3DArcCCW ；（不支持）

25：Abs3DhelixCW；

26：Rel3DhelixCW；

27：Abs3DHelixCCW；

28：Rel3DHelixCCW；

29：GPDELAY （單位：ms）

例 程

```
DIM AS ULONG RemainPath, FreeBuffer, curIndex,curCmd
BASE 0,1
SVON
GVL=1000                    '設置插補初速度
GVH=10000                   '設置插補運行速度
GACC=100000                 '設置插補加速度
GDEC=ACC                    '設置插補減速度
LINEABS 0,0                 '運動到目標位置（0,0）
WAIT DONE
PATHRESET                   '清除路徑緩存
'PATHBEGIN 開始到 PATHEND 之前的路徑段為連續插補運動
'PATHBEGIN 後面跟的編號不寫或寫 0 時，表明路徑添加完再開始執行連續插補，其他數值代表添加該數值段後開始執行運動

PATHBEGIN                   '進入連續插補狀態
MERGEON                     '連續插補速度交接模式設置為 fly mode
CIRCABS 1,10000,0,10000,-10000      '圓弧插補段
LINEABS 25000,-10000                '直線插補段
DELAY=500                           '延時段，延時 500ms
CIRCABS 1,25000,0,35000,0           '圓弧插補段
CIRCABS 1,25000,0,25000,10000       '圓弧插補段
LINEABS 10000,10000                 '直線插補段
CIRCABS 1,10000,0,0,0               '圓弧插補段
PATHEND
SLEEP 500
PATH_Status  RemainPath, FreeBuffer, curIndex, curCmd
Print  RemainPath, FreeBuffer, curIndex, curCmd     '列印出 6，9994,1,4
```

2.9 同步運動

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.9.1	STA_SRC	同步啟/停觸發源	√	√
2.9.2	SETSTA	以單軸相對運動模式狀態等待同步啟動信號	√	×
2.9.3	SETSTA_ABS	以單軸絕對運動模式狀態等待同步啟動信號	√	×
2.9.4	SETSTA_VEL	以恒速連續運動模式狀態等待同步啟動信號	√	×
2.9.5	STARTSTA	STA 或同張板卡同步軸開始啟動運動	√	×
2.9.6	STOPSTA	STA 或同張板卡同步軸開始停止運動	√	×
2.9.7	GANTRY	龍門運動	√	×
2.9.8	GEAR	電子齒輪運動	√	×
2.9.9	PHASE	電子齒輪運動過程中超前或落後相位運動	√	×
2.9.10	TANGENT	切向跟隨運動	√	×
2.9.11	MODULO	切向跟隨運動中旋轉刀軸旋轉一周的指令脈衝數	√	√
2.9.12	CAMTABLE	設定凸輪表數據	×	×
2.9.13	CAMSET	設定凸輪表相關配置	×	×
2.9.14	CAMIN	啟動電子凸輪，建立主從關係	×	×
2.9.15	CAMSTOP	解除主從凸輪關係	×	×

2.9.1 STA_SRC

所　屬：屬性

語　法：STA_SRC=value

類　型：ULONG

描　述：設置/讀取軸的同步啟停運動的觸發源，觸發源被觸發後，處於等待同步啟/停的軸會根據等待的模式，開始執行相應的運動。STA 信號可以由運動控制卡上硬體 STA 針腳觸發產生，也可以由 STARTSTA 指令觸發產生。

範　圍：設定值和返回值如下，預設值 1

0：禁用

1：板卡 STA 信號

2：軸 0 的比較信號

3：軸 1 的比較信號

4：軸 2 的比較信號

5：軸 3 的比較信號

6：軸 4 的比較信號

7：軸 5 的比較信號

8：軸 6 的比較信號

9：軸 7 的比較信號

10：軸 0 的停止信號

11：軸 1 的停止信號

12：軸 2 的停止信號

13：軸 3 的停止信號

14：軸 4 的停止信號

15：軸 5 的停止信號

16：軸 6 的停止信號

17：軸 7 的停止信號

例　程

```
BASE 1
STA_SRC = 1   '設置軸 1 的同步啟停運動觸發源為板卡 STA 信號
```

2.9.2 SETSTA

所　屬：命令

語　法：SETSTA　distance 1[,distance 2] [,distance 3]……

描　述：設置同步軸為點位元運動模式並處於等待觸發狀態，同時設置同步軸的相對移動距離。

參　數：distance　相對移動距離；類型：DOUBLE

例　程

```
BASE 0,1,2,3
STA_SRC=1    '設置同步源為板卡 STA 信號
SETSTA 10000,5000,30000,-8000 '設置軸 0,1,2,3 處於等待同步相對運動狀態，並設置運動距離
STARTSTA      '同步啟，同時啟動 4 軸運動
WAIT DONE
SETSTA_ABS 0,2000,5000,20000   '設置軸 0,1,2,3 處於等待同步絕對運動狀態，並設置目標位置
SLEEP 500
STARTSTA          '同步啟，同時啟動 4 軸運動
WAIT DONE
SETSTA_VEL 0,1,0,1 '設置軸 0,1,2,3 處於等待同步連續運動狀態，並設置各軸方向
STARTSTA          '同步啟，同時啟動 4 軸運動
Sleep 3000
STOPSTA           '同步停，同時停止 4 軸運動
```

2.9.3 SETSTA_ABS

所　屬：命令

語　法：SETSTA_ABS　position1[,position2] [,position3]……

描　述：設置同步軸為點位元運動模式並處於等待觸發狀態，同時設置同步軸絕對移動位置

參　數：position　絕對位置；類型：DOUBLE

例　程

```
BASE 0,1,2,3
STA_SRC=1    '設置同步源為板卡 STA 信號
SETSTA 10000,5000,30000,-8000 '設置軸 0,1,2,3 處於等待同步相對運動狀態，並設置運動距離
STARTSTA      '同步啟，同時啟動 4 軸運動
```

```
WAIT DONE
SETSTA_ABS 0,2000,5000,20000   '設置軸 0,1,2,3 處於等待同步絕對運動狀態，並設置目標位置
SLEEP 500
STARTSTA          '同步啟，同時啟動 4 軸運動
WAIT DONE
SETSTA_VEL 0,1,0,1 '設置軸 0,1,2,3 處於等待同步連續運動狀態，並設置各軸方向
STARTSTA          '同步啟，同時啟動 4 軸運動
Sleep 3000
STOPSTA           '同步停，同時停止 4 軸運動
```

2.9.4 SETSTA_VEL

所　屬：命令

語　法：SETSTA_VEL dir1[,dir2][,dir3]……

描　述：設置同步軸為連續運動模式並處於等待觸發狀態，同時設置同步軸的連續運動方向。

參　數：dir　方向，0：正向，1：反向；類型：ULONG

例　程

```
BASE 0,1,2,3
STA_SRC=1    ''設置同步源為板卡 STA 信號
SETSTA 10000,5000,30000,-8000 '設置軸 0,1,2,3 處於等待同步相對運動狀態，並設置運動距離
STARTSTA       '同步啟，同時啟動 4 軸運動
WAIT DONE
SETSTA_ABS 0,2000,5000,20000   '設置軸 0,1,2,3 處於等待同步絕對運動狀態，並設置目標位置
SLEEP 500
STARTSTA          '同步啟，同時啟動 4 軸運動
WAIT DONE
SETSTA_VEL 0,1,0,1 '設置軸 0,1,2,3 處於等待同步連續運動狀態，並設置各軸方向
STARTSTA          '同步啟，同時啟動 4 軸運動
Sleep 3000
```

```
STOPSTA              '同步停，同時停止 4 軸運動
```

2.9.5 STARTSTA

所　屬：命令

語　法：STARTSTA

描　述：當軸的同步啟/停觸發源為 STA 信號時，使用該指令開始同步啟動運動

例　程

請參考 SETSTA、SETSTA_ABS、SETSTA_VEL 等指令例程

2.9.6 STOPSTA

所　屬：命令

語　法：STOPSTA

描　述：當軸的同步啟/停觸發源為 STA 信號時，使用該指令開始同步停止運動

例　程

請參考 SETSTA、SETSTA_ABS、SETSTA_VEL 等指令例程

2.9.7 GANTRY

所　屬：命令

語　法：GANTRY AX(slave axis no),refsrc,dir[,max_diffvalue]

描　述：以 BASE 軸列表中的第一個軸為主軸，指定龍門運動從軸、參考源、龍門運動方向，建立龍門同步關係

參　數：slave axis no　從軸的軸號；範圍：根據控制器實際硬體決定

refsrc　從軸跟隨主軸的位置源；範圍：0：理論位置，1：實際位置（暫不支持）

dir　從軸與主軸的運動方向一致性；範圍：0：相同，1：相反

max_diffvalue　主軸和從軸讀到的編碼器位置誤差值限值，該值可以不設定。如果設定值後，控制器會即時比對誤差值，一旦超過 max_diffvalue 值，控制器會控制馬達停止。

注　意：龍門一旦建立龍門關係，主從軸狀態會變成同步狀態，要解除龍門關係，需對

從軸下 STOPDEC 或 STOPEMG 命令，龍門關係即解除。

例　程

```
BASE 0,1
SVON
'龍門軸的速度等參數由主軸參數決定
VL=1000
VH=40000
ACC=200000
DEC=200000
STOPDEC AX(1) '龍門運動中，對從軸下 STOPDEC 或 STOPEMG 可以解除龍門關係
'如果已有龍門關係存在，再下 GANTRY 命令，會報無效軸狀態錯誤
GANTRY AX(1),0,0    '設定龍門關係：從軸為軸 1，參考源為主軸的理論位置，從軸方向與主軸同向
BASE 0              '設定軸 0 為龍門主軸
MOVE 20000    '主軸執行距離為 20000 的正向相對運動，從軸這時會與跟隨主軸一起執行龍門運動
```

2.9.8 GEAR

所　屬：命令

語　法：GEAR　AX(slave axis no), numerator, denominator, refsrc, mode

描　述：以 BASE 軸清單中的第一個軸為主軸，指定電子齒輪運動從軸、齒輪分子、齒輪分母、參考源、運動模式。建立齒輪同步關係

參　數：slave axis no　從軸的軸號；範圍：根據控制器實際硬體決定

numerator　電子齒輪比分子；類型 ULONG

denominator　電子齒輪比分子；類型 ULONG

refsrc　從軸跟隨主軸的位置源；範圍：0：理論位置，1：實際位置

mode　主從軸齒輪關係模式；範圍：0：相對位置主從模式，1：絕對位置主從模式

注　意：一旦建立電子齒輪關係，主從軸狀態會變成同步狀態，要解除電子齒輪關係，需對從軸下 STOPDEC 或 STOPEMG 命令，齒輪關係即解除。

例　程

```
BASE 0,1
SVON
'齒輪關係的所有軸速度等參數由主軸參數決定
VL=1000
VH=40000
ACC=200000
DEC=200000
STOPDEC AX(1) '電子齒輪運動中，對從軸下 STOPDEC 或 STOPEMG 可以解除齒輪關係
'如果已有齒輪關係存在，再下 GEAR 命令，會報無效軸狀態錯誤
GEAR AX(1),1,1,0,0    '從軸為軸 1，齒輪分子分母分別為 1,1，從軸參考主軸理論位置，相對位置模式
BASE 0
MOVE 20000    '主軸執行距離為 20000 的正向相對運動，從軸這時會與跟隨主軸一起執行齒輪運動
```

2.9.9 PHASE

所　屬：命令

語　法：PHASE　AX(slave axis no), acc, dec, phase_speed, phase_dist

描　述：電子齒輪過程中使從軸進行相位超前或落後運動

參　數：slave axis no　從軸的軸號；範圍：根據控制器實際硬體決定

acc　　相位運動的加速度；類型 DOUBLE

dec　　相位運動的減速度；類型 DOUBLE

phase_speed　　相位運動的運行速度；類型 DOUBLE

phase_dist　　相位運動的超前或落後距離；類型 DOUBLE。Phase_dist >0, 從軸做相位超前運動；phase_dist <0,從軸做相位落後運動。

例　程

```
BASE 0,1
SVON
'齒輪關係的所有軸速度等參數由主軸參數決定
VL=1000
VH=40000
```

```
ACC=200000
DEC=200000
STOPDEC AX(1) '電子齒輪運動中，對從軸下 STOPDEC 或 STOPEMG 可以解除齒輪關係
'如果已有齒輪關係存在，再下 GEAR 命令，會報無效軸狀態錯誤
GEAR AX(1),1,1,0,0    '從軸為軸 1，齒輪分子分母分別為 1,1，從軸參考主軸理論位置，相對位置模式
BASE 0
MOVE 80000    '主軸執行距離為 20000 的正向相對運動，從軸這時會與跟隨主軸一起執行齒輪運動
Sleep 1000
Phase AX(1),50000,50000,30000,10000 '軸 1 進行距離為 10000 個脈衝當量的相位超前運動。
```

2.9.10 TANGENT

所　屬：命令

語　法：TANGENT AX(axis no), start_vector*, dir[,module_range]

描　述：指定切向跟隨軸、起始切向向量、運動方向，建立切向跟隨關係

參　數：AX(axis no)　切向跟隨軸的軸號

start_vector*　　起始切向向量陣列位址

dir　跟隨軸旋轉方向；範圍：0：與切向方向相同，1：與切向方向相反

module_range　刀向旋轉軸旋轉一周的指令脈衝數

注　意：一旦建立切向跟隨，主從軸狀態會變成同步狀態，要解除切向跟隨關係，需對從軸下 STOPDEC 或 STOPEMG 命令，切向跟隨關係即解除。

例　程

```
BASE 0,1
DIM StartArray(3) as SHORT
StartArray(0)=0
StartArray(1)=1
StartArray(2)=0
'跟隨的旋轉刀控制軸為軸 2，起始刀向向量為（0,1,0），軸 1 組成的平面，旋轉方向與刀向方向相同，旋轉刀軸運動一圈需要 3600 個脈衝
TANGENT AX(2),StartArray(),0,3600
CIRC 0,8000, 0,16000, 0  ' 軸 0，軸 1 進行圓弧插補運動，這時軸 2 會同時執行刀向跟隨運動
```

```
WAIT DONE
STOPDEC AX(2)
```

2.9.11 MODULO

所　屬：屬性

語　法：MODULO=value

類　型：ULONG

描　述：設置/讀取 ModuleRange 值。切向跟隨功能中，該值為刀向旋轉軸旋轉一周的指令脈衝數

範　圍：【0，8000000】，該值必須為 4 的倍數；預設值為 0

例　程

```
BASE 0
MODULO =10000 '設定軸 0 的 MODULO 值為 10000
```

2.9.12 CAMTABLE

語　法：CAMTABLE CamIndex, AX(MasAxisNo), AX(SlvAxisNo),VR_Start, DataCount[,CamID][, IsSpdSet]

描　述：設定凸輪表數據。凸輪表可通過 IDE 上的凸輪工具產生，詳細請參考 MotionStudio 使用手冊。

參　數：

CamIndex：0~7.（使用不同的凸輪表需指定不同的 index）

AX(MasAxisNo)：指定主軸軸號

AX(SlvAxisNo)：指定從軸軸號

VR_Start: 存放凸輪關係資料的起始 Index，在 IDE 上的凸輪工具上設定的 VR Index 一致。

VR(VR_Start) --- 第一個點的 X 值， 單位：pulse

VR(VR_Start +1) ---第二個點的 Y 值， 單位：pulse

VR(VR_Start +2) ---第一個點的 X 值， 單位：pulse

VR(VR_Start +3) ---第二個點的 Y 值， 單位：pulse

……

VR(VR_Start + 2* DataCount -2)---最後一個點的 X 值，單位：pulse

VR(VR_Start + 2*DataCount -1)---最後一個點的 Y 值，單位：pulse

VR(VR_Start + 2* DataCount) --第一個點的主從速度比

VR(VR_Start + 2* DataCount +1) --第二個點的主從速度比

……

VR(VR_Start + 3* DataCount - 1) --最後一個點的主從速度比

DataCount：凸輪表中的點數

CamID：指定 CamIndex 與 Device 上的實際 ID 對應關係。默認為 0.（每張 板卡支持兩個 CamID：0/1）

IsSpdSet: 使用自訂速度還是自動計算速度。在 MotionStudio 上可在凸輪工具中修改每個點主從速度比， 缺省為 FALSE。

FALSE---使用板卡內部自動計算速度。

TRUE---使用使用者規劃速度，則使用 VR(VR_Start + 2* DataCount)~ VR(VR_Start + 2* DataCount +1)中速度規劃。

例 程：

注：如下例程已先使用 MotionStudio 上的凸輪工具產生了凸輪表，並將資料寫入了 VR（1000）開始的位置。

```
base 0,1
'CAMINEX 為 1,AX(0)為主軸,AX(1)為從軸,
'VR(1000)~VR(11)記錄 6 個凸輪表的值，使用板卡內部速度規劃
CAMTABLE(1, AX(0), AX(1), 1000, 6, 0)
CAMSET 1,0,0,1   '設定主軸和從軸都是相對模式,週期性跟隨
CAMIN 1 '建立主從跟隨關係
BASE 0
MOVE 10000 '主軸開始移動，從軸按照凸輪表跟隨移動
WAIT DONE
CAMSTOP AX(1)   '解除從軸的跟隨關係
```

2.9.13 CAMSET

語 法：CAMSET CamIndex[,MasAbsOrRel][, SlvAbsOrRel][,Periodic]

描 述：設定凸輪表，需先使用 CAMTABLE 將資料寫入硬體中。

參　數：CamIndex：0~7. 與 CAMTABLE 一致

MasAbsOrRel：主軸絕對/相對匹配凸輪表，0—相對，1—絕對。

SlvAbsOrRel：從軸絕對/相對匹配凸輪表，0—相對，1—絕對。

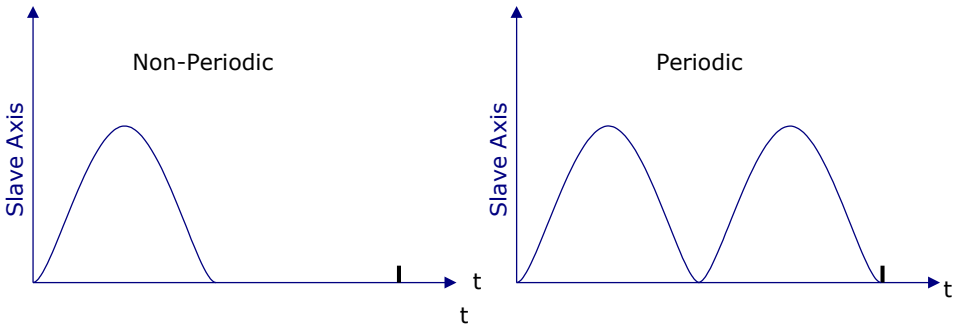
絕對：　當絕對凸輪系統設置後，基於 CamTable 的控制值或從值將被認為是絕對的。系統會補償同步過程中主從軸之間的任何偏差。當達到同步時，控制值和從值之間將建立一種確定的相位關係。

相對：當相對凸輪系統設置後，則表示系統不會補償在同步過程中控制值和從值之間的任何偏移。

Periodic：

0—非週期跟隨，隨著主軸的移動，凸輪表完成一個週期後將從軸停止動作；

1—週期跟隨，隨著主軸的移動，從軸週期性地跟隨主軸依照凸輪表動作。



例　程：參考 CAMTABLE 例程。

2.9.14 CAMIN

語　　法：CAMIN CamIndex, [MasOffset=0,][SlvOffset=0,][MasRatioNum =1,][MasRatioDemo =1,][SlvRatioNum =1,][SlvRatioDemo = 1,][RefSrc = 0]

描　述：建立主從軸的凸輪關係，並設定相關資料，調用此指令後，主從軸會建立跟隨關係，從軸的狀態會從 ready 狀態變成同步狀態。

參　數：CamIndex：0~7. 與 CAMTABLE 一致

MasOffset：主軸方向的座標偏移值。

SlvOffset：從軸方向的座標偏移值。

MasRatioNum：主軸座標中 CAMTABLE 的比例因數的分子。

MasRatioDemo：主軸座標中 CAMTABLE 的比例因數的分母。

SlvRatioNum：從軸座標中 CAMTABLE 的比例因數的分子。

SlvRatioDemo：從軸座標中 CAMTABLE 的比例因數的分母。

RefSrc：凸輪跟隨參照源。0—理論位置，1—回饋位置。

例　程：參考 CAMTABLE 例程。

2.9.15 CAMSTOP

語　法 1：CAMSTOP

語　法 2：CAMSTOP　AX(SlvAxNo)

描　述：解除從軸的跟隨關係。

參　數：SlvAxNo 從軸軸號

例　程：參考 CAMTABLE 例程。

2.10 輸入輸出埠控制

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.10.1	DIN	數位量輸入	√	√
2.10.2	DOUT	數位量輸出	√	√
2.10.3	ALM_EN	伺服報警使能	√	×
2.10.4	ALM_FILTER	伺服報警埠濾波	√	√
2.10.5	ALM_LOGIC	伺服報警埠邏輯電平	√	√
2.10.6	ALM_MODE	伺服報警觸發後運動停止模式	√	√
2.10.7	IN1STOP_EN	IN1STOP 功能使能	√	×
2.10.8	IN1_FILTER	IN1 埠濾波	√	√
2.10.9	IN1STOP_EDGE	IN1STOP 功能的觸發條件	√	√
2.10.10	IN1STOP_MODE	IN1STOP 觸發後運動停止模式	√	√
2.10.11	IN2STOP_EN	IN2STOP 功能使能	√	×
2.10.12	IN2_FILTER	IN2 埠濾波	√	√
2.10.13	IN2STOP_EDGE	IN2STOP 功能的觸發條件	√	√
2.10.14	IN2STOP_MODE	IN2STOP 觸發後運動停止模式	√	√
2.10.15	IN4STOP_EN	IN4STOP 功能使能	√	×
2.10.16	IN4_FILTER	IN4 埠濾波	√	√
2.10.17	IN4STOP_EDGE	IN4STOP 功能的觸發條件	√	√
2.10.18	IN4STOP_MODE	IN4STOP 觸發後運動停止模式	√	√
2.10.19	IN5STOP_EN	IN5STOP 功能使能	√	×
2.10.20	IN5_FILTER	IN5 埠濾波	√	√

2.10.21	IN5STOP_EDGE	IN5STOP 功能的觸發條件	√	√
2.10.22	IN5STOP_MODE	IN5STOP 觸發後運動停止模式	√	√
2.10.23	INSTOP_DEC	INSTOP 功能觸發後減速停止	√	×
2.10.24	INP_EN	伺服到位功能使能	√	×
2.10.25	INP_LOGIC	伺服到位埠邏輯電平	√	√
2.10.26	LTC_EN	鎖存功能使能	√	×
2.10.27	LTC_LOGIC	鎖存埠邏輯電平	√	√
2.10.28	LDPOS	鎖存到的理論位置	√	×
2.10.29	LMPOS	鎖存到的實際位置	√	×
2.10.30	TRIGLTC	軟體觸發鎖存功能	√	×
2.10.31	LTC_FLAG	軸鎖存信號標誌	√	×
2.10.32	RESETLTC	清除鎖存位置和標識	√	×
2.10.33	CMP_EN	比較觸發功能使能	√	×
2.10.34	CMP_LOGIC	比較觸發埠邏輯電平	√	√
2.10.35	CMP_METHOD	比較觸發功能比較方法	√	√
2.10.36	CMP_MODE	比較觸發的 DO 輸出模式	√	√
2.10.37	CMP_SRC	比較觸發的比較源	√	√
2.10.38	CMP_WIDTH	比較觸發的 DO 輸出模式為脈衝模式時的電平寬度	√	√
2.10.39	CPOS	當前比較位置資料	√	×
2.10.40	CMP	比較觸發模式選擇	√	×
2.10.41	CMP_FLAG	比較觸發信號標誌	√	×
2.10.42	RESETCMP	清除比較觸發信號標誌	√	×
2.10.43	CAMDO_EN	CAMDO 功能使能	√	×
2.10.44	CAMDO_LOGIC	CAMDO 埠邏輯電平	√	√
2.10.45	CAMDO_LPOS	CAMDO 低限位位置值	√	√
2.10.46	CAMDO_HPOS	CAMDO 高限位位置值	√	√
2.10.47	CAMDO_SRC	CAMDO 比較觸發源	√	√

2.10.48	MIO	運動控制相關的 I/O 狀態	×	×
2.10.49	EMG_LOGIC	EMG 邏輯電平	√	√
2.10.50	EMG_FILTER	EMG 埠濾波	√	×
2.10.51	DORESET	將指定區域 DO 值設為初始值	×	√
2.10.52	VRRESET	將指定區域 VR 值設為初始值	×	√
2.10.53	MCMP_EN	多軸比較使能	×	×
2.10.54	MCMP_CH	多軸比較對應軸的 OUT5 是否輸出	×	×
2.10.55	MCMP_MODE	多軸比較 DO 輸出模式	×	×
2.10.56	MCMP_DEVIA	多軸比較誤差範圍	×	×
2.10.57	MCMP_EMPTY	自動清除多軸比較資料	×	×
2.10.58	MCMP_LOGIC	多軸比較邏輯電平	×	√
2.10.59	MCMP_WIDTH	多軸比較脈衝模式時的脈寬	×	×
2.10.60	MCMP_PWMFREQ	多軸比較 PWM 模式時的頻率	×	×
2.10.61	MCMP_PWMDUTY	多軸比較 PWM 的占空比	×	×
2.10.62	MCMPPWMTIMETABLE	多軸比較的 PWM 輸出時間	×	×
2.10.63	CMPSETDO	手動控制 CMP_DO	×	√
2.10.64	MCMPSETDO	手動控制 MCMP_DO	×	√
2.10.65	MCMPFORCEOUT	手動控制 MCMP 輸出	×	√
2.10.66	MCMP_PWMMAXFREQ	多軸比較 PWM 的最大頻率	×	×
2.10.67	MCMP_PWMMINFREQ	多軸比較 PWM 的最小頻率	×	×
2.10.68	MCMP_PWMMAXDUTY	多軸比較 PWM 的最大占空比	×	×
2.10.69	MCMP_PWMMINDUTY	多軸比較 PWM 的最小占空比	×	×
2.10.70	GPWM_LINKEN	根據群組速度改變 PWM 輸	×	×

		出		
2.10.71	GPWM_MODE	群組 PWM 更改模式	×	×
2.10.72	GPWM_REFVEL	多軸比較 PWM 模式時的基本速度	×	×
2.10.73	LBUF_RESET	清除 Latch Buffer 中所有資料	×	×
2.10.74	LBUF_EN	Latch Buffer 使能	×	×
2.10.75	LBUF_DIST	Latch Buffer 鎖存距離間隔	×	×
2.10.76	LBUF_SRC	Latch Buffer 比較觸發源	×	×
2.10.77	LBUF_ID	設定/讀取 LatchBuffer 中的資料來源	×	×
2.10.78	LBUF_EVTNUM	產生 LTCBUFDONE 事件鎖存到的數據個數	×	×
2.10.79	LBUF_STATUS	獲取 latchbuffer 中資料個數以及空間大小	×	×
2.10.80	LBUF_DATA	獲取 Latchbuffer 中的資料	×	×

2.10.1 DIN

所　屬：命令（唯讀）

語　法：DIN (no)

描　述：讀取一個通用數位量輸入埠的狀態

參　數：no，數位量輸入的編號，在控制器配置時，系統會根據控制器硬體分配對應的編號；類型：ULONG

返回值：0：低電平，1：高電平

例　程

```
WHILE 1
IF DIN (0)=1 THEN        '判斷 DI0 信號是否有信號
    DOUT (0)=1           'DO0,DO1 置 1，DO2 置 0
    DOUT (1)=1
    DOUT (2)=0
ELSE
```

```
    DOUT (2)=1
END IF
WEND
```

2.10.2 DOUT

所　屬：命令

語　法：DOUT (no)=value

描　述：設置/讀取一個通用數位量輸出埠的狀態

參　數：no　數位量輸出的編號，在控制器配置時，系統會根據控制器硬體分配對應的編號；類型：ULONG

value　數位量輸出埠的狀態，範圍：0 或 1

返回值：輸出口的電平，0 或 1

例　程

```
WHILE 1
IF DIN (0)=1 THEN        '判斷 DI0 信號是否有信號
    DOUT (0)=1           'DO0,DO1 置 1，DO2 置 0
    DOUT (1)=1
    DOUT (2)=0
ELSE
    DOUT (2)=1
END IF
WEND
```

2.10.3 ALM_EN

所　屬：屬性

語　法：ALM_EN = value

類　型：ULONG

描　述：啟用/禁用運動報警功能，報警是當電機驅動處於報警狀態時，電機驅動器生成的信號

範　圍：設定值和返回值如下，預設值 0

0：禁用(預設值)

1：啟用

例　程

BASE 0

ALM_EN=1　'啟用軸 0 檢測報警功能

2.10.4 ALM_FILTER

所　屬：屬性

語　法：ALM_FILTER = value

類　型：ULONG

描　述：設置/讀取軸的報警(ALM) 輸入埠的濾波參數

範　圍 :設定值和返回值如下，預設值 0

0：5us

1：100us

2：200us

3：500us

例　程

BASE 0

ALM_FILTER=1 '設定軸 0 的 ALM 輸入埠濾波時間為 100us

2.10.5 ALM_LOGIC

所　屬：屬性

語　法：ALM_LOGIC = value

類　型：ULONG

描　述：設置/讀取報警輸入信號的有效邏輯電平

範　圍 :設定值和返回值如下，預設值 0

0：低電平

1：高電平

例　程

BASE 0

ALM_LOGIC =1　'設置軸 0 的報警輸入信號高電平有效

2.10.6 ALM_MODE

所　屬：屬性

語　法：ALM_MODE　= value

類　型：ULONG

描　述：設置/讀取接收報警信號時電機的停止模式

範　圍 :設定值和返回值如下，預設值 0

0：立即停止

1：減速停止

例　程

```
BASE 0
ALM_MODE=1   '設置接收到報警信號後電機減速停止
```

2.10.7 IN1STOP_EN

所　屬：屬性

語　法：IN1STOP_EN = value

類　型：ULONG

描　述：啟用/禁用 IN1 的觸發停止功能，該功能啟動用後，IN1 信號一旦有效，在運動

中的指定電機會被控制停止運動。研華運動控制的每個軸都關聯著 4 個 DI 埠，分別稱為 IN1，IN2，IN4，IN5。4 個埠都可以指定啟用 INSTOP 功能。

範　圍 :設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
IN1STOP_EN =1   '啟用軸 0 的 IN1 觸發停止功能
```

2.10.8 IN1_FILTER

所　屬：屬性

語　法：IN1_FILTER = value

類　型：ULONG

描　述：設置/讀取 IN1 埠的濾波時間

範　圍:設定值和返回值如下，預設值 0

0：5us

1：100us

2：200us

3：500us

例　程

BASE 0

IN1_FILTER =1　'設置軸 0 的 IN1 信號濾波時間 100us

2.10.9 IN1STOP_EDGE

所　屬：屬性

語　法：IN1STOP_EDGE = value

類　型：ULONG

描　述：設置/讀取 IN1STOP 功能的觸發條件。設置上升沿觸發時，IN1 埠有上升沿信號時， IN1STOP 功能被觸發。設置下降沿觸發時，IN1 埠有下降沿信號時，IN1STOP 功能被觸發。

範　圍:設定值和返回值如下，預設值 0

0：上升沿

1：下降沿

例　程

BASE 0

IN1STOP_EDGE =1 '設置 IN1STOP 功能的觸發條件為下降沿觸發有效。

2.10.10 IN1STOP_MODE

所　屬：屬性

語　法：IN1STOP_MODE　= value

類　型：ULONG

描　述：設置/讀取 IN1 觸發時電機的停止模式

範　圍:設定值和返回值如下，預設值 1

0：立即停止

1：減速停止

例　程

```
BASE 0
IN1STOP_MODE =1 '設置 IN1 觸發時電機作減速停止運動
```

2.10.11 IN2STOP_EN

所　屬：屬性

語　法：IN2STOP_EN = value

類　型：ULONG

描　述：啟用/禁用 IN2 的觸發停止功能，該功能啟動用後，IN2 信號一旦有效，在運動中的指定電機會被控制停止運動。研華運動控制的每個軸都關聯著 4 個 DI 埠，分別稱為 IN1，IN2，IN4，IN5。4 個埠都可以指定啟用 INSTOP 功能。

範　圍 :設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
IN2STOP_EN =1   '啟用軸 0 的 IN2 觸發停止功能
```

2.10.12 IN2_FILTER

所　屬：屬性

語　法：IN2_FILTER = value

類　型：ULONG

描　述：設置/讀取 IN2 埠的濾波時間

範　圍 :設定值和返回值如下，預設值 0

0：5us

1：100us

2：200us

3：500us

例　程

```
BASE 0
IN2_FILTER =1   '設置軸 0 的 IN2 信號濾波時間 100us
```

2.10.13 IN2STOP_EDGE

所　屬：屬性

語　法：IN2STOP_EDGE = value

類　型：ULONG

描　述：設置/讀取 IN2STOP 功能的觸發條件。設置上升沿觸發時，IN2 埠有上升沿信號時， IN2STOP 功能被觸發。設置下降沿觸發時，IN2 埠有下降沿信號時，IN2STOP 功能被觸發。

範　圍 :設定值和返回值如下，預設值 0

0：上升沿

1：下降沿

例　程

```
BASE 0
IN2STOP_EDGE =1 '設置 IN2STOP 功能的觸發條件為下降沿觸發有效。
```

2.10.14 IN2STOP_MODE

所　屬：屬性

語　法：IN2STOP_MODE　= value

類　型：ULONG

描　述：設置/讀取 IN2 觸發時電機的停止模式

範　圍 :設定值和返回值如下，預設值 1

0：立即停止

1：減速停止

例　程

```
BASE 0
IN2STOP_MODE =1 '設置 IN2 觸發時電機作減速停止運動
```

2.10.15 IN4STOP_EN

所　屬：屬性

語　法：IN4STOP_EN = value

類　型：ULONG

描　述：啟用/禁用 IN4 的觸發停止功能，該功能啟動用後，IN4 信號一旦有效，在動中的指定電機會被控制停止運動。研華運動控制的每個軸都關聯著 4 個 DI 端口，分別稱為 IN1，IN2，IN4，IN5。4 個埠都可以指定啟用 INSTOP 功能。

範　圍：設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
IN4STOP_EN =1   '啟用軸 0 的 IN4 觸發停止功能
```

2.10.16 IN4_FILTER

所　屬：屬性

語　法：IN4_FILTER = value

類　型：ULONG

描　述：設置/讀取 IN4 埠的濾波時間

範　圍：設定值和返回值如下，預設值 0

0：5us

1：100us

2：200us

3：500us

例　程

```
BASE 0
IN4_FILTER =1   '設置軸 0 的 IN4 信號濾波時間 100us
```

2.10.17 IN4STOP_EDGE

所　屬：屬性

語　法：IN4STOP_EDGE = value

類　型：ULONG

描　述：設置/讀取 IN4STOP 功能的觸發條件。設置上升沿觸發時，IN4 埠有上升沿信號時， IN4STOP 功能被觸發。設置下降沿觸發時，IN4 埠有下降沿信號時，IN4STOP 功能被觸發。

範　圍:設定值和返回值如下，預設值 0

0：上升沿

1：下降沿

例　程

```
BASE 0
IN4STOP_EDGE =1 '設置 IN4STOP 功能的觸發條件為下降沿觸發有效。
```

2.10.18 IN4STOP_MODE

所　屬：屬性

語　法：IN4STOP_MODE　= value

類　型：ULONG

描　述：設置/讀取 IN4 觸發時電機的停止模式

範　圍：設定值和返回值如下，預設值 1

0：立即停止

1：減速停止

例　程

```
BASE 0
IN4STOP_MODE =1 '設置 IN4 觸發時電機作減速停止運動
```

2.10.19 IN5STOP_EN

所　屬：屬性

語　法：IN5STOP_EN = value

類　型：ULONG

描　述：啟用/禁用 IN5 的觸發停止功能，該功能啟動用後，IN5 信號一旦有效，在運動中的指定電機會被控制停止運動。研華運動控制的每個軸都關聯著 4 個 DI 埠，分別稱為 IN1，IN2，IN4，IN5。4 個埠都可以指定啟用 INSTOP 功能。

範　圍:設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
IN5STOP_EN =1   '啟用軸 0 的 IN5 觸發停止功能
```

2.10.20 IN5_FILTER

所　屬：屬性

語　法：IN5_FILTER = value

類　型：ULONG

描　述：設置/讀取 IN5 埠的濾波時間

範　圍:設定值和返回值如下，預設值 0

0：5us

1：100us

2：200us

3：500us

例　程

```
BASE 0
IN5_FILTER =1   '設置軸 0 的 IN5 信號濾波時間 100us
```

2.10.21 IN5STOP_EDGE

所　屬：屬性

語　法：IN5STOP_EDGE = value

類　型：ULONG

描　述：設置/讀取 IN5STOP 功能的觸發條件。設置上升沿觸發時，IN5 埠有上升沿信號時， IN5STOP 功能被觸發。設置下降沿觸發時，IN5 埠有下降沿信號時，IN5STOP 功能被觸發。

範　圍:設定值和返回值如下，預設值 0

0：上升沿

1：下降沿

例　程

```
BASE 0
IN5STOP_EDGE =1 '設置 IN5STOP 功能的觸發條件為下降沿觸發有效。
```

2.10.22 IN5STOP_MODE

所　屬：屬性

語　法：IN5STOP_MODE　= value

類　型：ULONG

描　述：設置/讀取 IN5 觸發時電機的停止模式

範　圍 :設定值和返回值如下，預設值 1

0：立即停止

1：減速停止

例　程

```
BASE 0
IN5STOP_MODE =1 '設置 IN5 觸發時電機作減速停止運動
```

2.10.23 INSTOP_DEC

所　屬：屬性

語　法：INSTOP_DEC　= value

類　型：DOUBLE

描　述：設置/讀取 INSTOP 用減速停止模式時的減速度，單位為脈衝當量/s^2

範　圍：（0, MAXDEC），預設值 100000

例　程

```
BASE 0
INSTOP_DEC=20000 '設置軸 0 的 INSTOP 減速度為 20000 個脈衝當量/s^2
```

2.10.24 INP_EN

所　屬：屬性

語　法：INP_EN = value

類　型：ULONG

描　述：啟用/禁用檢測電機運動到位功能，到位信號是當電機運動到位時，電機驅動器生成到位信號。

範　圍 :設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
INP_EN =1 '啟用軸 0 的到位檢測功能
```

注　意

啟用到位檢測功能後，控制軸進行運動時，脈衝命令輸出完後，軸狀態不會立即變為 ready，要等到軸的 INP 埠接收到伺服送出的到位信號，軸狀態才會變為 ready。軸狀態未變為 ready 時，對該軸下運動指令將不成功。

2.10.25 INP_LOGIC

所　屬：屬性

語　法：INP_LOGIC = value

類　型：ULONG

描　述：設置/讀取到位輸入信號的有效邏輯電平

範　圍:設定值和返回值如下，預設值 1

0：低電平

1：高電平

例　程

```
BASE 0
INP_LOGIC =1    '設置軸 0 的到位輸入信號高電平有效
```

2.10.26 LTC_EN

所　屬：屬性

語　法：LTC_EN = value

類　型：ULONG

描　述：　啟用/禁用軸的鎖存功能，研華運動控制的每個軸都關聯著 4 個 DI 埠，分別稱為 IN1，IN2，IN4，IN5，每個軸的鎖存信號由 IN1 輸入埠產生，啟用鎖存功能後，一旦 IN1 埠接收到有效電平，控制器會立即鎖存指令理論位置值和編碼器實際位置值。

範　圍:設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
LTC_EN=1 '啟用軸 0 鎖存功能
```

2.10.27 LTC_LOGIC

所　屬：屬性

語　法：LTC_LOGIC = value

類　型：ULONG

描　述：設置/讀取鎖存輸入信號的有效邏輯電平

範　圍：設定值和返回值如下，預設值 0

0：低電平

1：高電平

例　程

```
BASE 0
LTC_LOGIC =0   '設置軸 0 鎖存輸入信號低電平有效
```

2.10.28 LDPOS

所　屬：命令

語　法：value=LDPOS(AX(no))

類　型：DOUBLE

描　述：讀取觸發鎖存得到的理論位置值

返回值：軸指令理論位置值

例　程

```
'完整例程可參考 TrigLTC 指令
DIM B AS DOUBLE
B=LDPOS(AX(1)) '獲取軸 1 的鎖存理論位置值
```

2.10.29 LMPOS

所　屬：命令

語　法：value=LMPOS(AX(no))

類　型：DOUBLE

描　述：讀取觸發鎖存得到的編碼器實際位置值

返回值：軸編碼器實際位置值

例　程

```
'完整例程可參考 TrigLTC 指令
DIM B AS DOUBLE
B=LMPOS(AX(1)) '獲取軸 1 的鎖存編碼器實際位置值
```

2.10.30 TrigLTC

所　屬：命令

語　法：TrigLTC AX(axis no)

描　述：用軟體命令觸發產生鎖存信號。實際應用中，鎖存信號基本上由硬體信號觸發，該命令多用於測試。

參　數：axis no　軸號；　範圍：根據控制器實際硬體決定。

例　程

```
DIM LTC_CmdDATA AS DOUBLE
DIM LTC_FBDATA AS DOUBLE
BASE 1
SVON
LTC_EN=1
MOVE 50000
SLEEP 2000
TrigLTC AX(1)        '觸發軸 1 的鎖存信號，進行位置鎖存
LTC_CmdDATA=LDPOS(AX(1))     '將鎖存到的指令理論位置讀取出來
PRINT LTC_CmdDATA
LTC_FBDATA=LMPOS(AX(1))      '將鎖存到的編碼器實際位置讀取出來
PRINT LTC_FBDATA
RESETLTC AX(1)               '清除鎖存標記，鎖存位置值
```

2.10.31 LTC_Flag

所　屬：命令

語　法：value=LTC_Flag(AX(no))

類　型：ushort

描　述：讀取軸鎖存標誌，捕捉到鎖存信號時，LTC_Flag 會置 1。要清除鎖存標誌，需要用 ResetLTC 指令清除。

返回值：鎖存標誌信號

例　程

```
DIM B AS USHORT
B= LTC_Flag(AX(1)) '獲取軸 1 的鎖存標誌信號
```

2.10.32 RESETLTC

所　屬：命令

語　法：RESETLTC AX(axis no)

描　述：清除鎖存資料、鎖存標記。未清除鎖存標記，下次接收到鎖存信號時，將不進行位置值的鎖存。

參　數：axis no　軸號；　範圍：根據控制器實際硬體決定。

例　程

```
'參考 TrigLTC 指令例程
```

2.10.33 CMP_EN

所　屬：屬性

語　法：CMP_EN = value

類　型：ULONG

描　述：　啟用/禁用軸比較觸發功能，研華運動控制的每個軸都關聯著 4 個 DO 埠，分別稱為 OUT4，OUT5，OUT6，OUT7，每個軸的比較觸發輸出由 OUT5 輸出埠產生，啟用比較觸發功能後，一旦比較觸發產生，OUT5 即輸出信號。

範　圍:設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
CMP_EN =1 '啟用軸 0 比較觸發功能
```

2.10.34 CMP_LOGIC

所　屬：屬性

語　法：CMP_LOGIC = value

類　型：ULONG

描　述：設置/讀取軸比較觸發有效輸出時的 DO 邏輯電平

範　圍 :設定值和返回值如下，預設值 0

0：低電平

1：高電平

例　程

```
BASE 0
CMP_LOGIC   =0 '設置比較觸發輸出的 DO 邏輯電平為低電平
```

2.10.35 CMP_METHOD

所　屬：屬性

語　法：CMP_METHOD = value

類　型：ULONG

描　述：設置/讀取軸比較觸發的比較方法

範　圍 :設定值和返回值如下，預設值 0

0：>= 位置計數器

1：<= 位置計數器

2：= 計數器 （無方向）

例　程

```
BASE 0
CMP_METHOD   =1'設置軸 0 的比較方法為小於等於位置計數器
```

2.10.36 CMP_MODE

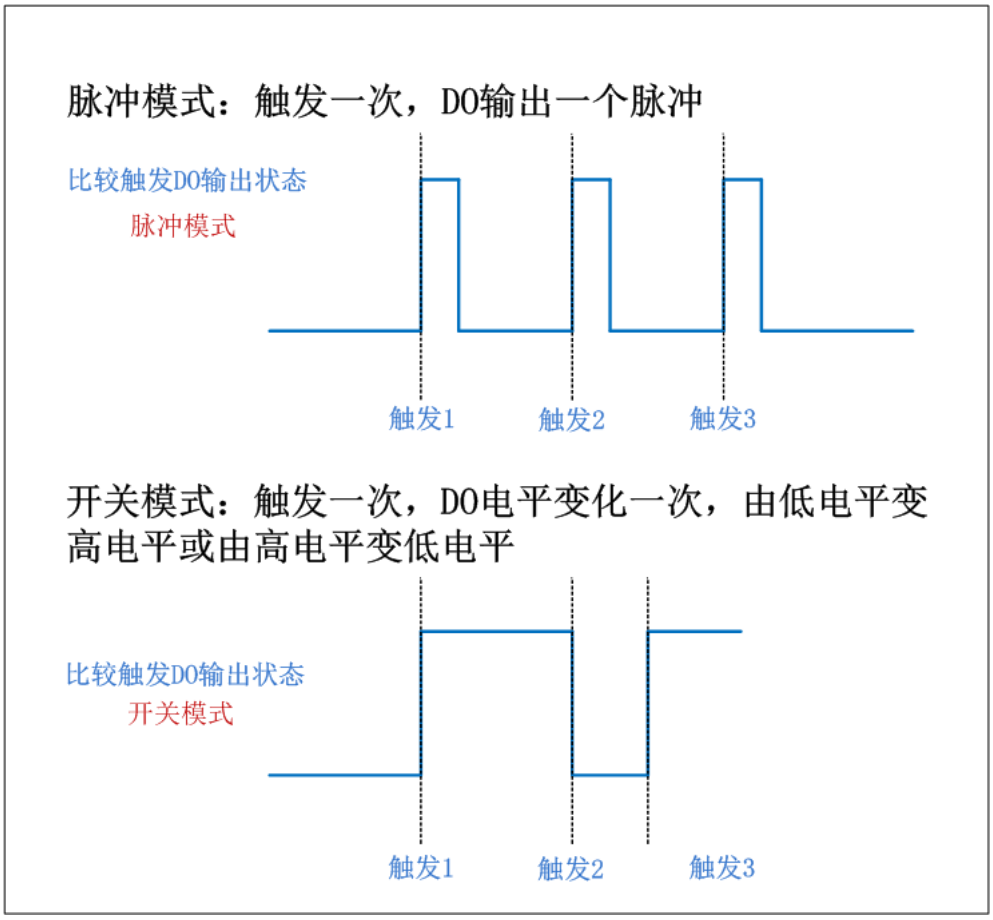
所　屬：屬性

語　法：CMP_MODE = value

類　型：ULONG

描　述：設置/讀取軸比較觸發的 DO 輸出模式，DO 輸出模式說明如下圖。

比較觸發 DO 輸出模式說明



範　圍:設定值和返回值如下，預設值 0

0：脈衝模式

1：開關模式

例　程

BASE 0

CMP_MODE　=1 '設置軸 0 的比較觸發 DO 輸出模式為開關模式

2.10.37 CMP_SRC

所　屬：屬性

語　法：CMP_SRC = value

類　型：ULONG

描　述：設置/讀取軸比較觸發的比較源

範　圍:設定值和返回值如下，預設值 0

0：理論位置

1：實際位置

例　程

```
BASE 0
CMP_SRC   =1   '設置軸 0 的位置比較源為實際位置
```

2.10.38 CMP_WIDTH

所　屬：屬性

語　法：CMP_WIDTH = value

類　型：ULONG

描　述：設置/讀取軸比較觸發的 DO 輸出模式為脈衝模式時的電平寬度，單位為微秒

範　圍 :1~85,000,000，預設值 0

例　程

```
BASE 0
CMP_WIDTH   =50 '設置軸 0 比較觸發的 DO 電平寬度為 50us
```

2.10.39 CPOS

所　屬：命令

語　法：value=CPOS(AX(no))

類　型：DOUBLE

描　述：讀取比較觸發中的當前比較位置資料

返回值：軸當前設定的比較位置資料值

例　程

```
DIM B AS DOUBLE
B=CPOS(AX(1))    '獲取軸 1 的當前筆比較位置值
```

2.10.40 CMP

所　屬：命令

語　法 1：CMP AX(axis no)， position

語　法 2：CMP AX(axis no)，start_position，end_position, interval

語　法 3：CMP AX(axis no)，tablearray( ),array_num

描　述：比較觸發的模式分 3 種。

✧ 語法 1 用於單點比較觸發，設置一個比較點的比較位置值。

✧ 語法 2 用於等距間隔多點比較觸發，設置起始比較點位置，終點比較點位置，間隔距離。

✧ 語法 3 用於隨機點多點比較觸發，設置多個要比較的位置值。

參　數：axis no　軸號；　範圍：根據控制器實際硬體決定。

Position　比較觸發的位置值

start_position　起始比較點位置值，用於等距間隔多點比較模式

end_position　終點比較點位置值，用於等距間隔多點比較模式

interval　間隔距離，用於等距間隔多點比較模式

tablearray（）　隨機比較點陣列，將要比較的點依次賦值到一個陣列裡，用於隨機點多點比較模式

array_num　比較點個數，用於隨機點多點比較模式

例　程

```
BASE 0
SVON
DIM cmp_table(3) AS double
cmp_table(0)=70000
cmp_table(1)=73400
cmp_table(2)=79424
CMP_EN=1
CMP_METHOD=0 '大於等於位置源時觸發
CMP_LOGIC=0 '觸發電平為低電平
CMP_SRC=0   '參考源為理論位置
CMP_WIDTH=100 '脈衝模式輸出脈寬為 100us
CMP_MODE=0 'DO 輸出模式脈衝模式
'單點比較觸發
CMP AX(0),10000 '設置比較位置為 10000
MOVEABS 0 '先運動到位置 0
WAIT DONE
```

```
MOVEABS 20000 '運到到 10000 的瞬間，比較觸發的 DO 輸出
WAIT Done
'均勻間隔比較觸發
CMP AX(0),30000,50000,4000 '比較位置從 30000 到 50000，每隔 4000，觸發一次 DO
MOVEABS 60000
WAIT DONE
'隨機表格位置比較觸發
CMP AX(0),cmp_table(),3
MOVEABS 80000
```

2.10.41 CMP_Flag

所　屬：命令

語　法：value=CMP_Flag(AX(no))

類　型：ushort

描　述：讀取軸比較觸發標誌，發生軸比較觸發時，CMP_Flag 會置 1。要清除鎖存標志，需要用 ResetCMP 指令清除。

返回值：比較觸發標誌信號

例　程

```
DIM B AS USHORT
B= CMP_Flag(AX(1)) '獲取軸 1 的比較觸發標誌信號
```

2.10.42 RESETCMP

所　屬：命令

語　法：RESETCMP AX(axis no)

描　述：清除比較觸發標誌。

參　數：axis no　軸號；　範圍：根據控制器實際硬體決定。

2.10.43 CAMDO_EN

所　屬：屬性

語　法：CAMDO_EN = value

類　型：ULONG

描　述： 啟動/禁用 CAMDO 功能，研華運動控制的每個軸都關聯著 4 個 DO 埠，分別稱為 OUT4，OUT5，OUT6，OUT7，每個軸的 CAMDO 輸出由 OUT4 輸出埠產生，啟用 CAMDO 功能後，一旦觸發產生，OUT4 即輸出信號。

範　圍:設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
CAMDO_EN=1    '啟用軸 0 上的 CAMDO 功能
```

2.10.44 CAMDO_LOGIC

所　屬：屬性

語　法：CAMDO_LOGIC = value

類　型：ULONG

描　述：設置/讀取 CAMDO 有效輸出時 DO 的電平邏輯

範　圍：設定值和返回值如下，預設值 1

0：低電平

1：高電平

例　程

```
BASE 0
CAMDO_LOGIC   =1 '設置 CAMDO 有效輸出時的電平為高電平
```

2.10.45 CAMDO_LPOS

所　屬：屬性

語　法：CAMDO_LPOS = value

類　型：ULONG

描　述： 設置/讀取 CAMDO 低限位元位置值，單位為脈衝當量

範　圍:設定值和返回值為【-2147483647，2147483647】，預設值 10000

例　程

```
BASE 0
```

CAMDO_LPOS=1000 '設置 CAMDO 低限位位置值為 1000

2.10.46 CAMDO_HPOS

所　屬：屬性

語　法：CAMDO_HPOS = value

類　型：ULONG

描　述：設置/讀取 CAMDO 高限位元位置值，單位為脈衝當量

範　圍：設定值和返回值為【-2147483647，2147483647】，預設值 20000

例　程

BASE 0

CAMDO_HPOS=1000 '設置 CAMDO 高限位位置值為 1000

2.10.47 CAMDO_SRC

所　屬：屬性

語　法：CAMDO_SRC = value

類　型：ULONG

描　述：設置/讀取 CAMDO 的位置比較源

範　圍：設定值和返回值如下，預設值 0

0：理論位置

1：實際位置

例　程

BASE 0

CAMDO_SRC=1 '設置軸 0 的 CAMDO 位置比較源為實際位置

2.10.48 MIO

所　屬：屬性（唯讀）

語　法：如下枚舉

✧ value=MIO.RDY，讀取伺服驅動器 Ready 信號

✧ value=MIO.ALM，讀取伺服驅動器報警信號

✧ value=MIO.PEL，讀取正方向硬極限信號

✧ value=MIO.NEL，讀取負方向硬極限信號

- ✧ value=MIO.ORG，讀取硬體原點信號
- ✧ value=MIO.DIR，讀取軸運動方向信號
- ✧ value=MIO.EMG，讀取軸卡上 EMG 信號
- ✧ value=MIO.PCS，暫不支持
- ✧ value= MIO.ERC，暫不支持
- ✧ value= MIO.EZ，讀取編碼器 Z 相輸入信號
- ✧ value= MIO.CLR，暫不支持
- ✧ value= MIO.LTC，讀取鎖存信號
- ✧ value= MIO.SD，暫不支持
- ✧ value= MIO.INP，讀取伺服驅動器到位信號
- ✧ value= MIO.SVON，讀取伺服使能輸出信號
- ✧ value= MIO.ALRM，讀取報警重定信號輸出狀態
- ✧ value= MIO.SPEL，讀取正方向軟極限信號
- ✧ value= MIO.SNEL，讀取負方向軟極限信號
- ✧ value= MIO.CMP，讀取比較觸發輸出信號
- ✧ value= MIO.CAMDO，讀取 CAM DO 輸出信號

類　型：ULONG

描　述：讀取跟運動控制相關的 I/O 狀態

返回值：0 或 1

2.10.49 EMG_LOGIC

所屬：屬性

語　法：EMG_LOGIC = value

類　型：ULONG

描　述：設置/讀取板卡的 EMG 邏輯電平

範　圍：設定值和返回值如下，預設值 0

0：低電平

1：高電平

例　程

```
BASE 0
EMG_LOGIC=1 '設置板卡的 EMG 邏輯電平為高電平
```

2.10.50 EMG_FILTER

所屬：屬性

語　法：EMG_FILTER = value

類　型：ULONG

描　述：設置/讀取 EMG 埠的濾波時間

範　圍：設定值和返回值如下，預設值 0

0：5us

1：100us

2：200us

3：500us

例　程

```
BASE 0
EMG_FILTER =1   '設置 EMG 信號的濾波時間為 100us
```

2.10.51 DORESET

所屬：命令

語　法：DORESET (StartIndex, DoCnt)

類　型：ULONG

描　述：將指定區域的 DO 值設置為初始值

參　數：StartIndex：起始 DO index。（第一個 Index 為 0）

DoCnt：要初始化的 DO 個數　。

例　程

```
DORESET (0, 10)   '將 0~9 這 10 個 DO 的值設置為初始值
```

2.10.52 VRRESET

所屬：命令

語　法：VRReset(start, cnt)

類　型：ULONG

描　述：將指定區域的 VR 值設置為初始值

參　數：start：起始 VR 區域起始位置。

cnt：要初始化的 VR 個數。

例　程

```
VRRESET(100,20) '將 100~119 這 20 個 VR 的值設置為初始值
```

2.10.53 MCMP_EN

所屬：屬性

語　法：MCMP_EN = value

類　型：ULONG

描　述：啟用/禁用多軸比較功能，需搭配 BASE 使用，以指定同時比較哪些軸的位置（BASE 中的軸需在同一個板卡類型中）。

範　圍：設定值和返回值如下，預設值 0

0：禁用(預設值)

1：啟用

例　程

```
BASE 0,1,2
VH = 400
DPOS=0
'PWM 模式時,需要先設置比較位置再設置其他參數,PWM 模式才有效.
CMP AX(0),2000,14000,3000
CMP AX(1),2000,14000,3000
CMP AX(2),2000,14000,3000
MCMP_EN=1          '使能多軸比較功能
MCMP_MODE=2 '0:Pulse 1:Toggle 2:PWM 3:PWM Toggle
MCMP_LOGIC=1 'PWM 模式下不起作用
MCMP_CH=7
MCMP_WIDTH= 1000000 '單位:um,PWM 和 toggle 模式下不起作用
MCMP_PWMMINFREQ=1
MCMP_PWMMAXFREQ=200 '1-250000
```

```
MCMP_PWMMINDUTY=10
MCMP_PWMMAXDUTY=80
MCMP_PWMFREQ=100     'PWM 頻率
MCMP_PWMDUTY=50      '占空比
GPWM_LINKEN=1 '啟用根據群組速度改變 PWM 輸出功能
GPWM_MODE=0     '0：頻率     1：占空比
GPWM_REFVEL=1000
MCMP_EMPTY = 1
MCMP_DEVIA=10
CMP_METHOD=2
CMP_SRC=1
DIM onTimeArray(5) as ULONG
onTimeArray(0)=1000
onTimeArray(1)=500
onTimeArray(2)=1000
onTimeArray(3)=5000
onTimeArray(4)=2000
MCMPPWMTIMETABLE onTimeArray(),3
MOVEABS 15000,15000,15000,15000
WAIT DONE
```

2.10.54 MCMP_CH

所屬：屬性

語　法：按位輸出, Bit0~3 分別對應軸 0~3, 如下枚舉

value=0　，禁用 OUT5 輸出比較信號

value=1　，軸 0　的 OUT5 輸出比較信號

value=2　，軸 1　的 OUT5 輸出比較信號

value=3　，軸 0, 1 的 OUT5 輸出比較信號

value=4　，軸 2　的 OUT5 輸出比較信號

value=5　，軸 0, 2 的 OUT5 輸出比較信號

value=6 ，軸 1, 2 的 OUT5 輸出比較信號

value= 7 ，軸 0, 1, 2 的 OUT5 輸出比較信號

value= 8 ，軸 3 的 OUT5 輸出比較信號

value= 9 ，軸 0,3 的 OUT5 輸出比較信號

value= 10，軸 1, 3 的 OUT5 輸出比較信號

value= 11，軸 0, 1, 3 的 OUT5 輸出比較信號

value= 12，軸 2, 3 的 OUT5 輸出比較信號

value= 13，軸 0, 2, 3 的 OUT5 輸出比較信號

value= 14，軸 1, 2, 3 的 OUT5 輸出比較信號

value= 15，軸 0,1,2,3 的 OUT5 輸出比較信號

類 型：ULONG

描 述：啟用/禁用多軸比較對應軸的 OUT5 輸出

2.10.55 MCMP_MODE

所屬：屬性

語 法：MCMP_MODE = value

類 型：ULONG

描 述：設置/讀取多軸比較觸發的 DO 輸出模式

範 圍：設定值和返回值如下，預設值 0

0：脈衝模式

1：開關模式

2：PWM 模式

3：PWM-TOGGLE 模式

例 程

```
BASE 0,1
MCMP_MODE   =1 '設置軸 0,1 的多軸比較 DO 輸出模式為開關模式
```

2.10.56 MCMP_DEVIA

所屬：屬性

語 法：MCMP_DEVIA = value

類 型：ULONG

描　述：設置/讀取多軸比較誤差範圍，單位為脈衝當量

範　圍：設定值和返回值為【0，65535】，預設值 0

例　程

```
BASE 0,1
MCMP_DEVIA=100   '設置多軸比較誤差範圍
```

2.10.57 MCMP_EMPTY

所屬：屬性

語　法：MCMP_EMPTY = value

類　型：ULONG

描　述：啟用/禁用自動清除多軸比較資料

範　圍：設定值和返回值如下，預設值 0

0：禁用(預設值)

1：啟用

例　程

```
BASE 0,1
MCMP_EN=1    '比較結束後自動清除多軸比較資料
```

2.10.58 MCMP_LOGIC

所屬：屬性

語　法：MCMP_LOGIC = value

類　型：ULONG

描　述：設置/讀取多軸比較邏輯電平

範　圍：設定值和返回值如下，預設值 0

0：低電平

1：高電平

例　程

```
BASE 0,1
MCMP_LOGIC=1 '設置多軸比較的輸出電平為高電平
```

2.10.59 MCMP_WIDTH

所屬：屬性

語　法：MCMP_WIDTH = value

類　型：ULONG

描　述：設置/讀取多軸比較為脈衝模式時的脈衝寬度，單位為微秒

範　圍：【0，85,899,000】，預設值 1

例　程

```
BASE 0,1
MCMP_WIDTH   =1000000 '設置多軸比較脈衝寬度為 1000000us
```

2.10.60 MCMP_PWMFREQ

所屬：屬性

語　法：MCMP_PWMDUTY = value

類　型：ULONG

描　述：設置/讀取多軸比較為 PWM 模式時的頻率

範　圍：【1Hz – 250,000Hz】，預設值 100000Hz。

例　程

```
BASE 0,1
MCMP_PWMFREQ   =10000 '設置多軸比較 PWM 頻率為 10000
```

2.10.61 MCMP_PWMDUTY

所屬：屬性

語　法：MCMP_PWMDUTY = value

描　述：設置/讀取多軸比較為 PWM 模式時的占空比

範　圍：【0，100】，預設值 0

例　程

```
BASE 0,1
MCMP_PWMDUTY   =50 '設置多軸比較 PWM 占空比為 50
```

2.10.62 MCMPPWMTIMETABLE

所屬：命令

語　法：MCMPPWMTIMETABLE　OnTimeArray(),　Cnt

描　述：設置多軸比較 PWM 與 PWM-Toggle 模式時的 PWM 輸出時間

參數：　OnTimeArray()　每個比較位置啟動後的持續時間(單位：us)一次最多設定 30000 筆。

Cnt：有效資料的個數。

注　意：

1. 當所設置的比較位置的個數大於所設置的比較時間個數時,剩餘比較位置的 PWM 輸出時間會沿用上個比較位置設置的輸出時間.

2.需要先設置比較位置再設置時間表才有效。

範　圍：設定值和返回值為【-2147483647，2147483647】，預設值 20000

例　程

```
BASE 0,1
DIM onTimeArray(5) as ULONG
onTimeArray(0)=1000
onTimeArray(1)=5000
onTimeArray(2)=1000
onTimeArray(3)=5000
onTimeArray(4)=2000
MCMPPWMTIMETABLE onTimeArray(),3
'設置 5 筆 PWM 輸出時間，其中前 3 個有效
```

2.10.63 CMPSETDO

所屬：命令

語　法：CMPSETDO　AX(id),　OnOrOff

類　型：ULONG

描　述：手動設置軸的 CMP_DO 輸出狀態

參　數：ax(id)：軸號

OnOrOff：打開或關閉 DO，0：關閉，1：打開

例　程

```
CMPSETDO AX(0),1    '觸發軸 0 上的 CAMDO 輸出
```

2.10.64 MCMPSETDO

所屬：命令

語　法：MCMPSETDO　OnOrOff

類　型：ULONG

描　述：手動設置多軸比較各軸的 CMP_DO 輸出狀態

參　數：OnOrOff：打開或關閉 DO，0：關閉，1：打開

例　程

```
BASE 0,1,2
MCMPSETDO (1)  '觸發軸 0,1,2 上的 CAMDO 輸出
```

2.10.65 MCMPFORCEOUT

所屬：命令

語　法：MCMPFORCEOUT

類　型：ULONG

描　述：手動觸發多軸比較輸出，輸出方式以 MCMP_MODE 設定決定。

例　程

```
BASE 0,1,2
MCMP_EN=1
MCMP_MODE=0
MCMP_LOGIC=1
MCMP_WIDTH= 1000000
MCMPFORCEOUT '手動控制多軸比較輸出
```

2.10.66 MCMP_PWMMAXFREQ

所屬：屬性

語　法：MCMP_PWMMAXFREQ =　value

類　型：ULONG

描　述：設置/讀取多軸比較為 PWM 模式時的最大頻率，單位為赫茲

範　圍：【1，250,000】

例　程

```
BASE 0,1
MCMP_PWMMAXFREQ  =200 '設置多軸比較 PWM 最大頻率為 200Hz
```

2.10.67 MCMP_PWMMINFREQ

所屬：屬性

語　法：MCMP_PWMMINFREQ =　value

類　型：ULONG

描　述：設置/讀取多軸比較為 PWM 模式時的最小頻率，單位為赫茲

範　圍：【1，250,000】

例　程

```
BASE 0,1
MCMP_PWMMINFREQ　=10 '設置多軸比較 PWM 頻率為 10Hz
```

2.10.68 MCMP_PWMMAXDUTY

所屬：屬性

語　法：MCMP_PWMMAXDUTY = value

類　型：ULONG

描　述：設置/讀取多軸比較為 PWM 模式時的最大占空比

範　圍：【0，100】，預設值 0

例　程

```
BASE 0,1
MCMP_PWMMAXDUTY　=80 '設置多軸比較 PWM 最大占空比為 80
```

2.10.69 MCMP_PWMMINDUTY

所屬：屬性

語　法：MCMP_PWMMINDUTY = value

類　型：ULONG

描　述：設置/讀取多軸比較為 PWM 模式時的最小占空比

範　圍：【0，100】，預設值 0

例　程

```
BASE 0,1
MCMP_PWMMINDUTY　=20 '設置多軸比較 PWM 最小占空比為 20
```

2.10.70 GPWM_LINKEN

所屬：屬性

語　法：GPWM_LINKEN= value

類　型：ULONG

描　述：啟用/禁用根據群組速度改變 PWM 輸出功能

範　圍：設定值和返回值如下，預設值 0

0：禁用(預設值)

1：啟用

例　程

```
BASE 0,1
GPWM_LINKEN=1   '啟用根據群組速度改變 PWM 輸出功能
```

2.10.71 GPWM_MODE

所屬：屬性

語　法：GPWM_MODE = value

類　型：ULONG

描　述：設置/讀取根據群組速度改變依照何種模式更改 PWM 輸出

範　圍：設定值和返回值如下

0：頻率

1：占空比

例　程

```
BASE 0,1
GPWM_MODE   =0 '最終頻率會根據設置的基準速度與原先設置的頻率計算得到
```

2.10.72 GPWM_REFVEL

所屬：屬性

語　法：GPWM_REFVEL = value

類　型：ULONG

描　述：設置/讀取多軸比較為 PWM 模式時的基本速度，以計算 PWM 的改變因數

例如如果設定模式是修改 PWM 頻率，則當前 PwmFreq=(當前群組速度/客戶設置基準速度)* MCMP_PWMFREQ，如果 PwmFreq 超過最大 MCMP_PWMMAXFREQ，

則使用 MCMP_PWMMAXFREQ 作為當前頻率，如果 PwmFreq 低於最小 MCMP_PWMMINFREQ，則使用 MCMP_PWMMINFREQ 作為當前頻率。

例　程

```
BASE 0,1
GPWM_REFVEL   =1000 '設置多軸比較 PWM 模式時的基本速度為 1000
```

2.10.73 LBUF_RESET

所屬：命令

語　法：LBUF_RESET　AX(axis no)

類　型：ULONG

描　述：清除鎖存緩存中的資料。

參　數：axis no　　軸號

範　圍：根據控制器實際硬體決定

例　程

```
'完整例程可參考 LBUF_EN 指令
LBUF_RESET AX(0)     '清除軸 0 鎖存數據
```

2.10.74 LBUF_EN

所屬：屬性

語　法：LBUF_EN = value

類　型：ULONG

描　述：啟用/禁用 latch buffer 功能，使用此功能，需現將 LTC_EN 打開。

範　圍：設定值和返回值如下，預設值 0

0：禁用(預設值)

1：啟用

例　程

```
BASE 3
LBUF_RESET AX(3)
LTC_EN=1
LTC_LOGIC =0
LBUF_EN=1          '啟用軸 3latch buffer 功能
```

```
LBUF_DIST=1000
LBUF_SRC=0
LBUF_ID=0
LBUF_EVTNUM=5
MOVE 100000
WAIT DONE
```

2.10.75 LBUF_DIST

所屬：屬性

語　法：LBUF_DIST = value

類　型：ULONG

描　述：設置/讀取相鄰兩筆鎖存資料的間隔距離，單位為 Pulse.

如果兩次相鄰鎖存資料的間隔長度小於設定值，則第二個 latch 會被忽略。

範　圍：【0，2147483647】，預設值 1000

例　程

```
'完整例程可參考 LBUF_EN 指令
BASE 0
LBUF_DIST   =2000 '設置軸 0 相鄰兩筆鎖存資料的間隔距離為 2000
```

2.10.76 LBUF_SRC

所屬：屬性

語　法：LBUF_SRC = value

類　型：ULONG

描　述：設置/讀取 LatchBuffer 比較觸發的比較源。

範　圍：設定值和返回值如下，預設值 0

0：理論位置

1：實際位置

例　程

```
BASE 0
LBUF_SRC   =1   '設置軸 0 的位置比較源為實際位置
```

2.10.77 LBUF_ID

所屬：屬性

語　法：LBUF_ID = value

類　型：ULONG

描　述：設置/讀取 LatchBuffer 中的資料來自哪一軸的位置，目前只支持板卡 PCI1285-MAS

注　意：該 ID 號是基於程式中所 BASE 的軸號來設定的，用法見例程

範　圍：軸號，根據控制器實際硬體決定

例　程

```
'完整例程可參考 LBUF_EN 指令
BASE 4,5,6
LBUF_ID=1    '軸 4、5、6 鎖存的都是軸 1 的位置資料
```

2.10.78 LBUF_EVTNUM

所屬：屬性

語　法：LBUF_EVTNUM= value

類　型：ULONG

描　述：當鎖存的筆數達到 LBUF_EVTNUM 時，產生 LTCBUFDONE 事件

範　圍：【0，128】，預設值 128

例　程

```
'完整例程可參考 LBUF_EN 指令
BASE 0
LBUF_EVTNUM   =10 '設置軸 0LatchBuffer 中鎖存到 10 個數據時，產生 LTCBUFDONE 事件
```

2.10.79 LBUF_STATUS

所屬：命令

語　法：LBUF_STATUS AX(AxNo), RemCnt, SpaceCnt

描　述：獲取 latchbuffer 中尚未讀取的資料個數以及剩餘空間，總空間大小為 128

參　數：AxNo：軸號

RemCnt：latchbuffer 中尚未讀取的數據個數

SpaceCnt：latchbuffer 中剩餘空間大小

例　程

'請參考 LBUF_DATA 指令

2.10.80 LBUF_DATA

所屬：命令

語　法：LBUF_DATA AX(AxNo), DataArray(), DataCnt

描　述：從 Latch buffer 中讀取指定數量的資料

參　數：AxNo：軸號

DataArray()：讀取到的 LatchBuffer 中的資料清單，單位：脈衝當量

DataCnt：指定讀取的數據個數

例　程

```
DIM DataArray(0 to 30) as Double
DIM AS ULONG RemCnt,SpaceCnt,DataCnt
BASE 4
WAIT LTCBUFDONE '等待 latch buffer 中資料個數大於或等於 LBUF_EVTNUM 的設定時觸發事件。
LBUF_STATUS(AX(4),RemCnt,SpaceCnt)
LBUF_DATA(AX(4),DataArray(),30)
PRINT RemCnt
PRINT SpaceCnt
PRINT DataArray(0)
PRINT DataArray(1)
PRINT DataArray(2)
PRINT DataArray(3)
PRINT DataArray(4)
```

2.11 回原點與限位

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.11.1	HOME_VL	原點運動初速度	√	√
2.11.2	HOME_VH	原點運動運行速度	√	√
2.11.3	HOME_ACC	原點運動加速度	√	√
2.11.4	HOME_DEC	原點運動減速度	√	√
2.11.5	HOME_JK	原點運動速度曲線類型	√	√
2.11.6	HOME_MODE	原點運動模式	√	√
2.11.7	HOMEP	正向原點運動	√	×
2.11.8	HOMEN	反向原點運動	√	×
2.11.9	HOME_CROSS	原點運動跨越距離	√	√
2.11.10	HOME_OFFSETDIST	原點運動完成後再移動的偏移距離	√	√
2.11.11	HOME_OFFSETVEL	原點運動完成後移動偏移距離的速度	√	√
2.11.12	HOME_RESET	原點運動後清零位置值功能	√	×
2.11.13	ORG_LOGIC	原點埠邏輯電平	√	√
2.11.14	ORG_MODE	原點運動結束時的停止模式	√	√
2.11.15	ORG_FILTER	原點埠的濾波	√	√
2.11.16	EZ_LOGIC	Z 相埠邏輯電平	√	√
2.11.17	EL_EN	硬限位功能使能	√	×
2.11.18	EL_LOGIC	硬限位元埠邏輯電平	√	√
2.11.19	EL_MODE	硬限位元觸發時的停止模式	√	√
2.11.20	PEL_FILTER	正方向硬限位埠濾波	√	√

2.11.21	NEL_FILTER	負方向硬限位埠濾波	√	√
2.11.22	SPEL	正方向軟限位值	√	√
2.11.23	SPEL_EN	正方向軟限位功能使能	√	×
2.11.24	SPEL_MODE	正方向軟限位元觸發時的停止模式	√	√
2.11.25	SNEL	負方向軟限位值	√	√
2.11.26	SNEL_EN	負方向軟限位功能使能	√	×
2.11.27	SNEL_MODE	負方向軟限位元觸發時的停止模式	√	√
2.11.28	PEL_TOL_EN	正方向硬極限容差功能使能	√	×
2.11.29	PEL_TOL	正方向硬極限容差值	√	√
2.11.30	NEL_TOL_EN	負方向硬極限容差功能使能	√	×
2.11.31	NEL_TOL	負方向硬極限容差值	√	√
2.11.32	SPEL_TOL_EN	正方向軟極限容差功能使能	√	×
2.11.33	SPEL_TOL	正方向軟極限容差值	√	√
2.11.34	SNEL_TOL_EN	負方向軟極限容差功能使能	√	×
2.11.35	SNEL_TOL	負方向軟極限容差值	√	√

2.11.1 HOME_VL

所　屬：屬性

語　法：HOME_VL = value

類　型：DOUBLE

描　述：設置/讀取回原點運動的初速度，單位為脈衝當量/s

範　圍：（0, MAXVEL），預設值 2000

例　程

```
BASE 0
```

```
HOME_VL=1000 '設定軸 0 回原點運動的初速度為 1000 個脈衝當量/s
```

2.11.2 HOME_VH

所　屬：屬性

語　法：HOME_VH = value

類　型：DOUBLE

描　述：設置/讀取回原點運動的最大運行速度，單位為脈衝當量/s

範　圍：（HOME_VL, MAXVEL），預設值 8000

例　程

```
BASE 0
HOME_VH=5000 '設定軸 0 回原點運動的運行速度為 5000 個脈衝當量/s
```

2.11.3 HOME_ACC

所　屬：屬性

語　法：HOME_ACC = value

類　型：DOUBLE

描　述：設置/讀取回原點運動的加速度，單位為脈衝當量/s^2

範　圍：（0, MAXACC），預設值 10000

例　程

```
BASE 0
HOME_ACC=20000 '設置軸 0 回原點的加速度為 20000 個脈衝當量/s^2
```

2.11.4 HOME_DEC

所　屬：屬性

語　法：HOME_DEC = value

類　型：DOUBLE

描　述：設置/讀取回原點運動的減速度，單位為脈衝當量/s^2

範　圍：（0, MAXDEC），預設值 10000

例　程

```
BASE 0
HOME_DEC=20000 '設置軸 0 回原點的減速度為 20000 個脈衝當量/s^2
```

2.11.5 HOME_JK

所　屬：屬性

語　法：HOME_JK = value

類　型：ULONG

描　述：設置/讀回原點運動的速度曲線類型

範　圍：【0,1】，0：T 型曲線；1：S 型曲線，預設值 0

例　程

```
BASE 0
HOME_JK=1   '設置軸 0 回原點的速度曲線為 S 型曲線
```

2.11.6 HOME_MODE

所　屬：屬性

語　法：HOME_MODE = value

類　型：ULONG

描　述：設置/讀取回原點運動的模式

範　圍 :設定值和返回值如下，預設值 0

0：MODE1_Abs

1：MODE2_Lmt

2：MODE3_Ref

3：MODE4_Abs_Ref

4：MODE5_Abs_NegRef

5：MODE6_Lmt_Ref

6：MODE7_AbsSearch

7：MODE8_LmtSearch

8：MODE9_AbsSearch_Ref

9：MODE10_AbsSearch_NegRef

10：MODE11_LmtSearch_Ref

11：MODE12_AbsSearchReFind

12：MODE13_LmtSearchReFind

13：MODE14_AbsSearchReFind_Ref

14：MODE15_AbsSearchReFind_NegR

15：MODE16_LmtSearchReFind_Ref

例　程

```
BASE 0
HOME_MODE=7     '設定軸 0 的回原點模式未 MODE8_LmtSearch
```

2.11.7 HOMEP

所　屬：命令

語　法 1：HOMEP

語　法 2：HOMEP AX(axis no)

語　法 3：HOMEP dir1[，dir2][，dir3]……

描　述：BASE 軸列表的軸或指定軸、方向，開始正向回原點運動。HOMEP 分 3 種方法使用如下。

✧　語法 1 用於對 BASE 軸列表的軸執行正向回原點運動

✧　語法 2 用於指定某個軸執行正向回原點運動

✧　語法 3 用於對 BASE 軸列表的軸，指定不同方向，執行回原點運動。Dir 為 0 時，方向與 HOMEP 同向；dir 為 1 時，方向與 HOMEP 反向

參　數：dir　0：正向；1：反向

axis no　軸號；　範圍：根據控制器實際硬體決定。

相關指令參考：HOME_MODE；HOME_CROSS；HOME_RESET

例　程

```
BASE 0,1,2
HOME_VL=500
HOME_VH=10000
HOME_ACC=50000
HOME_DEC=50000
HOME_CROSS=2000   '設置原點運動中跨越距離
HOME_RESET=1      '原點運動結束後清零理論位置、實際位置
HOME_MODE=6       '6：MODE7_AbsSearch
HOMEP AX(1)       '指定軸 1 執行正向回原點運動
WAIT AX(1),DONE   '等待軸 1 運動停止
```

```
BASE 0,1                '基於軸 0,軸 1
HOMEP                   '軸 0，軸 1 都執行正向回原點運動
WAIT DONE               '等待兩個軸的回原點運動停止
HOMEN                   '軸 0，軸 1 都執行負向回原點運動
WAIT DONE
HOMEP 0,1               '軸 0 執行正向回原點運動，軸 1 執行反向回原點運動
WAIT DONE
MOVE 5000,5000    '原點運動停止後，執行兩軸相對點位運動
```

2.11.8 HOMEN

所　屬：命令

語　法 1：HOMEN

語　法 2：HOMEN AX(axis no)

語　法 3：HOMEN dir1[，dir2][，dir3]……

描　述：BASE 軸列表的軸或指定軸、方向，開始負向回原點運動。HOMEN 分 3 種方法使用如下。

✧ 語法 1 用於對 BASE 軸列表的軸執行負向回原點運動

✧ 語法 2 用於指定某個軸執行負向回原點運動

✧ 語法 3 用於對 BASE 軸列表的軸，指定不同方向，執行回原點運動。Dir 為 0 時，方向與 HOMEN 同向；dir 為 1 時，方向與 HOMEN 反向

參　數：dir　0：反向；1：正向

axis no　軸號；　範圍：根據控制器實際硬體決定。

相關指令參考：HOME_MODE；HOME_CROSS；HOME_RESET

例　程

```
BASE 0,1,2
HOME_VL=500
HOME_VH=10000
HOME_ACC=50000
HOME_DEC=50000
HOME_CROSS=2000   '設置原點運動中跨越距離
HOME_RESET=1      '原點運動結束後清零理論位置、實際位置
```

```
HOME_MODE=6        '6：MODE7_AbsSearch
HOMEP AX(1)        '指定軸 1 執行正向回原點運動
WAIT AX(1),DONE    '等待軸 1 運動停止
BASE 0,1           '基於軸 0,軸 1
HOMEP              '軸 0，軸 1 都執行正向回原點運動
WAIT DONE          '等待兩個軸的回原點運動停止
HOMEN              '軸 0，軸 1 都執行負向回原點運動
WAIT DONE
HOMEP 0,1          '軸 0 執行正向回原點運動，軸 1 執行反向回原點運動
WAIT DONE
MOVE 5000,5000     '原點運動停止後，執行兩軸相對點位運動
```

2.11.9 HOME_CROSS

所　屬：屬性

語　法：HOME_CROSS= value

類　型：DOUBLE

描　述：設置/讀取回原點運動時的跨越距離。HOME_MODE 裡有幾種模式會用到 HOME_CROSS，請參考 HOME_MODE 指令說明。

例　程

```
BASE 0
HOME_CROSS=100 '設定軸 0 回原點運動時的跨越距離為 100 個脈衝當量
```

2.11.10 HOME_OFFSETDIST

所　屬：屬性

語　法：HOME_OFFSETDIST = value

類　型：DOUBLE

描　述：設置/讀取回原點運動完成後再移動的偏移距離。

例　程

```
BASE 0
HOME_OFFSETDIS =1000 '設置軸 0 的回原點偏移距離為 1000 個脈衝當量
```

2.11.11 HOME_OFFSETVEL

所　屬：屬性

語　法：HOME_OFFSETVEL = value

類　型：DOUBLE

描　述：設置/讀取回原點運動完成後移動偏移距離的速度

範　圍：（0, MAXVEL），預設值 8000

例　程

```
BASE 0
HOME_OFFSETVEL =1000 '設置軸 0 的回原點偏移速度為 1000 個脈衝當量/s
```

2.11.12 HOME_RESET

所　屬：屬性

語　法：HOME_RESET= value

類　型：ULONG

描　述：啟用或禁用回原點後清零位置值功能

範　圍 :設定值和返回值如下，預設值 1

0：禁用

1：啟用

例　程

```
BASE 0
HOME_RESET=1   '啟用軸 0 回完原點後清零位置值功能
```

2.11.13 ORG_LOGIC

所　屬：屬性

語　法：ORG_LOGIC = value

類　型：ULONG

描　述：設置/讀取 ORG 信號的有效邏輯電平。ORG 專用數位量輸入埠用於回原點運動中的幾種用到 ORG 信號的模式，請參考 HOME_MODE 指令說明。

範　圍 :設定值和返回值如下，預設值 0

0：低電平

1：高電平

例　程

```
BASE 0
ORG_LOGIC =1   '設置軸 0 的 ORG 輸入信號高電平有效
```

2.11.14 ORG_MODE

所　屬：屬性

語　法：ORG_MODE　= value

類　型：ULONG

描　述：設置/讀取回原點運動結束時的停止模式。

範　圍：設定值和返回值如下，預設值 1

0：立即停止

1：減速停止

例　程

```
BASE 0
ORG_MODE =1   '設置軸 0 的回原點運動停止模式未減速停止模式
```

2.11.15 ORG_FILTER

所　屬：屬性

語　法：ORG_FILTER = value

類　型：ULONG

描　述：設置/讀取軸的 ORG 輸入信號的濾波時間

範　圍：設定值和返回值如下，預設值 0

0：5us

1：100us

2：200us

3：500us

例　程

```
BASE 0
ORG_FILTER =1   '設置 ORG 信號的濾波時間為 100us
```

2.11.16 EZ_LOGIC

所　屬：屬性

語　法：EZ_LOGIC = value

類　型：ULONG

描　述：設置/讀取電機編碼器 Z 相輸入信號的有效邏輯電平。運動控制中，Z 相信號常常用於回原點運動中，請參考 HOME_MODE 的指令說明。

範　圍 :設定值和返回值如下，預設值 0

0：低電平

1：高電平

例　程

```
BASE 0
EZ_LOGIC =0   '設置軸 0 的 Z 相輸入信號低電平有效
```

2.11.17 EL_EN

所　屬：屬性

語　法：EL_EN = value

類　型：ULONG

描　述：啟用/禁用硬體限位元功能，啟用後限位元開關被觸發，相應方向上運動的電機會被控制停下來

範　圍 :設定值和返回值如下，預設值 1

0：禁用

1：啟用

例　程

```
BASE 0
EL_EN =1 '啟用硬體限位元功能
```

2.11.18 EL_LOGIC

所　屬：屬性

語　法：EL_LOGIC = value

類　型：ULONG

描　述：設置/讀取硬體限位元輸入信號的有效邏輯電平

範　圍 :設定值和返回值如下，預設值 0

0：低電平

1：高電平

例　程

BASE 0

EL_LOGIC =1 '設置軸 0 的硬極限輸入信號高電平有效

2.11.19 EL_MODE

所　屬：屬性

語　法：EL_MODE　= value

類　型：ULONG

描　述：設置/讀取接收硬體限位元信號時電機的停止模式

範　圍 :設定值和返回值如下，預設值 0

0：立即停止

1：減速停止

例　程

BASE 0

EL_MODE =0 '設置軸 0 碰到硬極限時電機立即停止

2.11.20 PEL_FILTER

所　屬：屬性

語　法：PEL_FILTER = value

類　型：ULONG

描　述：設置/讀取軸的正方向硬限位元輸入信號的濾波時間

範　圍 :設定值和返回值如下，預設值 0

0：5us

1：100us

2：200us

3：500us

例　程

BASE 0

PEL_FILTER =1;設置軸 0 的正方向硬限位元信號濾波時間為 100us

2.11.21 NEL_FILTER

所　屬：屬性

語　法：NEL_FILTER = value

類　型：ULONG

描　述：設置/讀取軸的負方向硬限位元輸入信號的濾波時間

範　圍 :設定值和返回值如下，預設值 0

0：5us

1：100us

2：200us

3：500us

例　程

BASE 0

NEL_FILTER =1;設置軸 0 的負方向硬限位元信號濾波時間為 100us

2.11.22 SPEL

所　屬：屬性

語　法：SPEL = value

類　型：LONG

描　述：設置/讀取正方向軟限位元的值，單位為脈衝

例　程

BASE 0

SPEL =100 '設置正方向軟體限位元的值為 100

2.11.23 SPEL_EN

所　屬：屬性

語　法：SPEL_EN = value

類　型：ULONG

描　述：啟用/禁用正方向軟限位功能，啟用正方向軟限位功能後，正向移動的電機指令位置到達 SPEL 設定的值後，馬達會被控制停止運動。

範　圍 :設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0

SPEL_EN =1   '啟用軸 0 的正方向軟限位功能
```

2.11.24 SPEL_MODE

所　屬：屬性

語　法：SPEL_MODE　= value

類　型：ULONG

描　述：設置/讀取正方向軟體限位元功能被觸發時電機被控制停止的模式

範　圍 :設定值和返回值如下，預設值 1

0：立即停止

1：減速停止

例　程

```
BASE 0

SPEL_MODE =1   '設置軸 0 的正方向軟體限位元被觸發時，電機被控制的停止模式為減速停止
```

2.11.25 SNEL

所　屬：屬性

語　法：SNEL = value

類　型：LONG

描　述：設置/讀取負方向軟限位元的值，單位為脈衝

例　程

```
BASE 0

SNEL =100 '設置負方向軟體限位元的值為 100
```

2.11.26 SNEL_EN

所　屬：屬性

語　法：SNEL_EN = value

類　型：ULONG

描　述：啟用/禁用負方向軟限位功能，啟用負方向軟限位功能後，負向移動的電機指令位置到達 SNEL 設定的值後，馬達會被控制停止運動。

範　圍:設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
SNEL_EN =1   '啟用軸 0 的負方向軟限位功能
```

2.11.27 SNEL_MODE

所　屬：屬性

語　法：SNEL_MODE　= value

類　型：ULONG

描　述：設置/讀取負方向軟體限位元功能被觸發時電機被控制停止的模式

範　圍:設定值和返回值如下，預設值 1

0：立即停止

1：減速停止

例　程

```
BASE 0
SNEL_MODE =1   '設置軸 0 的負方向軟體限位元被觸發時，電機被控制的停止模式為減速停止
```

2.11.28 PEL_TOL_EN

所　屬：屬性

語　法：PEL_TOL_EN = value

類　型：ULONG

描　述： 啟用/禁用正方向硬極限容差功能。該功能僅在外部手輪操作時使用。當手輪控制電機運動時，碰到極限信號後，由於極限會限制某方向的運動，而且觸碰極限會發生軸錯誤報警，導致手輪不能正常控制電機移出極限。該指令功能開啟後，會允許在極限附近的某段範圍，觸碰極限不產生軸錯誤報警，使得手輪可以正常控制電機。

範　圍:設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
PEL_TOL_EN =1 '啟用正方向硬極限容差功能
```

2.11.29 PEL_TOL

所　屬：屬性

語　法：PEL_TOL = value

類　型：ULONG

描　述： 設置/讀取軸的正方向硬極限容差值。

範　圍:設定值和返回值為【0，2147483647】，預設值 5000

例　程

```
BASE 0
PEL_TOL =100 '設置軸 0 的正方向硬極限容差值為 100 個脈衝
```

2.11.30 NEL_TOL_EN

所　屬：屬性

語　法：NEL_TOL_EN = value

類　型：ULONG

描　述： 啟用/禁用負方向硬極限容差功能。該功能僅在外部手輪操作時使用。當手輪控制電機運動時，碰到極限信號後，由於極限會限制某方向的運動，而且觸碰極限會發生軸錯誤報警，導致手輪不能正常控制電機移出極限。該指令功能開啟後，會允許在極限附近的某段範圍，觸碰極限不產生軸錯誤報警，使得手輪可以正常控制電機。

範　圍:設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
NEL_TOL_EN =1 '啟用負方向硬極限容差功能
```

2.11.31 NEL_TOL

所　屬：屬性

語　法：NEL_TOL = value

類　型：ULONG

描　述： 設置/讀取軸的負方向硬極限容差值。

範　圍 :設定值和返回值為【0，2147483647】，預設值 5000

例　程

```
BASE 0
NEL_TOL =100 '設置軸 0 的負方向硬極限容差值為 100 個脈衝
```

2.11.32 SPEL_TOL_EN

所　屬：屬性

語　法：SPEL_TOL_EN = value

類　型：ULONG

描　述： 啟用/禁用正方向軟極限容差功能。該功能僅在外部手輪操作時使用。當手輪控制電機運動時，碰到極限信號後，由於極限會限制某方向的運動，而且觸碰極限會發生軸錯誤報警，導致手輪不能正常控制電機移出極限。該指令功能開啟後，會允許在極限附近的某段範圍，觸碰極限不產生軸錯誤報警，使得手輪可以正常控制電機。

範　圍 :設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0
SPEL_TOL_EN =1 '啟用正方向軟極限容差功能
```

2.11.33 SPEL_TOL

所　屬：屬性

語　法：SPEL_TOL = value

類　型：ULONG

描　述： 設置/讀取軸的正方向軟極限容差值。

範　圍:設定值和返回值為【0，2147483647】，預設值 5000

例　程

```
BASE 0

SPEL_TOL =100 '設置軸 0 的正方向軟極限容差值為 100 個脈衝
```

2.11.34 SNEL_TOL_EN

所　屬：屬性

語　法：SNEL_TOL_EN = value

類　型：ULONG

描　述： 啟用/禁用負方向軟極限容差功能。該功能僅在外部手輪操作時使用。當手輪控制電機運動時，碰到極限信號後，由於極限會限制某方向的運動，而且觸碰極限會發生軸錯誤報警，導致手輪不能正常控制電機移出極限。該指令功能開啟後，會允許在極限附近的某段範圍，觸碰極限不產生軸錯誤報警，使得手輪可以正常控制電機。

範　圍:設定值和返回值如下，預設值 0

0：禁用

1：啟用

例　程

```
BASE 0

SNEL_TOL_EN =1 '啟用負方向軟極限容差功能
```

2.11.35 SNEL_TOL

所　屬：屬性

語　法：SNEL_TOL = value

類　型：ULONG

描　述： 設置/讀取軸的負方向軟極限容差值。

範　圍:設定值和返回值為【0，2147483647】，預設值 5000

例　程

```
BASE 0

SNEL_TOL =100 '設置軸 0 的負方向軟極限容差值為 100 個脈衝
```

2.12 JOG 與手輪

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.12.1	JOG_VL	JOG 運動低速段速度	√	√
2.12.2	JOG_VH	JOG 運動高速段速度	√	√
2.12.3	JOG_ACC	JOG 運動加速度	√	√
2.12.4	JOG_DEC	JOG 運動減速度	√	√
2.12.5	JOG_VLTIME	JOG 運動低段速度運行的時間	√	√
2.12.6	JOGP	正向軟體 JOG 運動	√	×
2.12.7	JOGN	負向軟體 JOG 運動	√	×
2.12.8	JOGON	使能外部驅動的 JOG 功能	√	×
2.12.9	JOGOFF	禁用外部驅動的 JOG 功能	√	×
2.12.10	MPGON	使能外部驅動的手輪功能	√	×
2.12.11	MPGOFF	禁用外部驅動的手輪功能	√	×
2.12.12	EXT_MODE	手輪模式外部驅動的脈衝輸入模式	√	√
2.12.13	EXT_PULSE	手輪模式外部驅動時，每個手輪脈衝輸入對應多少個指令脈衝輸出值	√	√
2.12.14	EXT_SRC	外部驅動的信號接入哪個軸的外部驅動輸入埠	√	√

2.12.1 JOG_VL

所　屬：屬性

語　法：JOG_VL = value

類　型：DOUBLE

描　述：設置/讀取 JOG 運動的低速段速度，單位為脈衝當量/s。當 JOG_VLTIME 值不為 0 時，JOG_VL 將起作用。

範　圍：（0, MAXVEL），預設值 2000

例　程

```
BASE 0
JOG_VL=1000   '設置軸 0 的 JOG 運動低速段速度為 1000 個脈衝當量/s
```

2.12.2 JOG_VH

所　屬：屬性

語　法：JOG_VH = value

類　型：DOUBLE

描　述：設置/讀取 JOG 運動的高速段速度，單位為脈衝當量/s

範　圍：（JOG_VL, MAXVEL），預設值 8000

例　程

```
BASE 0
JOG_VH=1000   '設置軸 0 的 JOG 運動高速段速度為 1000 個脈衝當量/s
```

2.12.3 JOG_ACC

所　屬：屬性

語　法：JOG_ACC = value

類　型：DOUBLE

描　述：設置/讀取 JOG 運動的加速度，單位為脈衝當量/s^2

範　圍：（0, MAXACC），預設值 10000

例　程

```
BASE 0
JOG_ACC=20000 '設置軸 0 的 JOG 運動加速度為 20000 個脈衝當量/s^2
```

2.12.4 JOG_DEC

所　屬：屬性

語　法：JOG_DEC = value

類　型：DOUBLE

描　述：設置/讀取 JOG 運動的減速度，單位為脈衝當量/s^2

範　圍：（0, MAXDEC），預設值 10000

例　程

```
BASE 0
JOG_DEC=20000 '設置軸 0 的 JOG 運動減速度為 20000 個脈衝當量/s^2
```

2.12.5 JOG_VLTIME

所　屬：屬性

語　法：JOG_VLTIME =value

類　型：ULONG

描　述：設置/讀取 JOG 運動低段速度運行的時間，單位為 ms。研華規劃的 JOG 運動分兩段速度。JOG 指令下達後，先控制電機的速度為 JOG_VL，JOG_VL 運動 JOG_VLTIME 值的時間後，控制電機加速到 JOG_VH。如果 JOG_VLTIME 值設置為 0，JOG 指令下達後，直接控制電機加速到 JOG_VH，JOG_VL 將不起作用。

範　圍：大於等於 0，預設值 5000

例　程

```
BASE 0
JOG_VLTIME=1000 '設置軸 0 的 JOG 運動高低速切換時間為 1000ms
```

2.12.6 JOGP

所　屬：命令

語　法 1：JOGP

語　法 2：JOGP AX(axis no)

語　法 3：JOGP dir1[，dir2][，dir3]……

描　述：BASE 軸列表的軸或指定軸、方向，開始正向 JOG 運動。JOGP 分 3 種方法使用如下。

✧　語法 1 用於對 BASE 軸列表的軸執行正向 JOG 運動

✧　語法 2 用於指定某個軸執行正向 JOG 運動

✧　語法 3 用於對 BASE 軸列表的軸，指定不同方向，執行 JOG 運動。Dir 為 0 時，方向與 JOGP 同向；dir 為 1 時，方向與 JOGP 反向

參　數：dir　0：正向；1：反向

axis no　軸號；　範圍：根據控制器實際硬體決定。

例　程

```
BASE 0,1,2
JOG_VL=500
JOG_VH=10000
JOG_ACC=50000
JOG_DEC=50000
JOG_VLTIME=2000      '設定低段速度運行的時間為 2 秒
SLEEP 5000
STOPDEC
JOGP                 '軸 0、1、2 都執行正向 JOG 運動
SLEEP 3000
STOPDEC      JOGP AX(1)          '指定軸 1 執行正向 JOG 運動，軸速度會先加速到 JOG_VL 運行 2 秒，
再加速到 JOG_VH

WAIT DONE
JOGP 0,1,0      '軸 0,2 執行正向 JOG 運動，軸 1 執行負向 JOG 運動
SLEEP 4000
STOPDEC
```

2.12.7 JOGN

所　屬：命令

語　法 1：JOGN

語　法 2：JOGN AX(axis no)

語　法 3：JOGN dir1[，dir2][，dir3]……

描　述：BASE 軸列表的軸或指定軸、方向，開始負向 JOG 運動。JOGN 分 3 種方法使用如下。

- ✧ 語法 1 用於對 BASE 軸列表的軸執行負向 JOG 運動
- ✧ 語法 2 用於指定某個軸執行負向 JOG 運動
- ✧ 語法 3 用於對 BASE 軸列表的軸，指定不同方向，執行 JOG 運動。Dir 為 0 時，方向與 JOGN 同向；dir 為 1 時，方向與 JOGN 反向

參　數：dir　0：反向；1：正向

axis no　軸號；　範圍：根據控制器實際硬體決定。

例　程

```
BASE 0,1,2
JOG_VL=500
JOG_VH=10000
JOG_ACC=50000
JOG_DEC=50000
JOG_VLTIME=2000      '設定低段速度運行的時間為 2 秒
JOGN AX(1)    '指定軸 1 執行負向 JOG 運動，軸速度會先加速到 JOG_VL 運行 2 秒，再加速到 JOG_VH
SLEEP 5000
STOPDEC
JOGN          '軸 0、1、2 都執行負向 JOG 運動
SLEEP 3000
STOPDEC
WAIT DONE
JOGN 0,1,0      '軸 0,2 執行負向 JOG 運動，軸 1 執行正向 JOG 運動
SLEEP 4000
STOPDEC
```

2.12.8 JOGON

所　屬：命令

語　法：JOGON

描　述：BASE 軸列表的第一個軸，使能外部驅動的 JOG 功能。該指令對外部硬體接線控制的 JOG 運動起作用。研華運動控制的每個軸都關聯著 4 個 DI 埠，分別稱為 IN1，IN2，IN4，IN5。當使用外部驅動的 JOG 功能時，IN4 和 IN5 分別控制 JOG+和 JOG-。

2.12.9 JOGOFF

所　屬：命令

語　法：JOGOFF

描　述：BASE 軸列表的第一個軸，禁用外部驅動的 JOG 功能。該指令對外部硬體接線

控制的 JOG 運動起作用。研華運動控制的每個軸都關聯著 4 個 DI 埠，分別稱為 IN1，IN2，IN4，IN5。當使用外部驅動的 JOG 功能時，IN4 和 IN5 分別控制 JOG+和 JOG-。

2.12.10 MPGON

所　屬：命令

語　法：MPGON

描　述：BASE 軸列表的第一個軸，使能外部驅動的 MPG 功能。該指令對外部硬體接線控制的 MPG 運動起作用。研華運動控制的每個軸都關聯著 4 個 DI 埠，分別稱為 IN1，IN2，IN4，IN5。當使用外部驅動的 MPG 功能時，IN4 和 IN5 分別控制對應手輪脈衝輸入的 A 相和 B 相。

2.12.11 MPGOFF

所　屬：命令

語　法：MPGOFF

描　述：BASE 軸列表的第一個軸，禁用外部驅動的 MPG 功能。該指令對外部硬體接線控制的 MPG 運動起作用。研華運動控制的每個軸都關聯著 4 個 DI 埠，分別稱為 IN1，IN2，IN4，IN5。當使用外部驅動的 MPG 功能時，IN4 和 IN5 分別控制對應手輪脈衝輸入的 A 相和 B 相。

2.12.12 EXT_MODE

所　屬：屬性

語　法：EXT_MODE = value

類　型：ULONG

描　述：設置/讀取手輪模式外部驅動的脈衝輸入模式

範　圍：設定值和返回值如下，預設值 2

0：1XAB

1：2XAB

2：4XAB

3：CCW/CW

例　程

```
EXT_MODE =1    '設置手輪外部驅動的脈衝輸入模式為 2XAB
```

2.12.13 EXT_PULSE

所　屬：屬性

語　法：EXT_PULSE = value

類　型：ULONG

描　述：設置/讀取手輪模式外部驅動時，每個手輪脈衝輸入對應多少個指令脈衝輸出值

範　圍：【1,1000】；預設值為 1

例　程

```
EXT_PULSE =2    '設置手輪脈衝輸入對應 2 個指令脈衝輸出
```

2.12.14 EXT_SRC

所　屬：屬性

語　法：EXT_SRC = value

類　型：ULONG

描　述：設置/讀取外部驅動的信號接入哪個軸的外部驅動輸入埠

範　圍：設定值和返回值如下，預設值 0

0： 0 軸

1： 1 軸(暫不支持)

2： 2 軸(暫不支持)

3： 3 軸(暫不支持)

例　程

```
BASE 0
EXT_SRC =0    '設置外部驅動信號接到軸 0 的外部驅動埠
```

2.13 通信指令

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.13.1	COM_OPEN	打開串口	×	×
2.13.2	COM_CLOSE	關閉串口	×	×
2.13.3	COM_SET	設置串口通訊參數	×	×
2.13.4	COM_ReadStream	串口自由協定讀操作，通過串口讀數據	×	×
2.13.5	COM_WriteStream	串口自由協議寫操作，通過串口寫資料	×	×
2.13.6	COM_ResetBuf	清除串口緩存區資料	×	×
2.13.7	TCP_OPEN	打開一個 TCP 通訊連接	×	×
2.13.8	TCP_CLOSE	關閉一個 TCP 通訊連接	×	×
2.13.9	TCP_STATUS	檢查 TCP 連接狀態	×	×
2.13.10	TCP_WAIT	等待 TCP 連接完成	×	×
2.13.11	TCP_Check	獲取 TCP 通信接收到的字元個數	×	×
2.13.12	TCP_ ReadSTR	控制器接收字串指令	×	×
2.13.13	TCP_WriteSTR	控制器發送字串指令	×	×
2.13.14	TCP_Read	控制器接收資料	×	×
2.13.15	TCP_Write	控制器發送資料	×	×
2.13.16	TCP_ReadVR	控制器用 VR 變數接收資料	×	×
2.13.17	TCP_WriteVR	控制器把 VR 變數中的資料發送出去	×	×
2.13.18	TCP_ResetBuf	清除 TCP 緩存區數據	×	×

2.13.1 COM_OPEN

所　屬：命令

語　法：COM_OPEN port

描　述：指定串口編號，打開串口。相應串口埠被打開後，才可以對該串口操作。該指令需要根據本地串口資源進行操作。

參　數：port　串口埠號

注　意：打開串口操作僅適用於未打開的串口，如果串口資源已經被打開，下該指令操會執行不成功，並返回錯誤。

例　程

```
COM_OPEN 2              '打開串口 2
COM_SET 2, 9600, 0, 1, 8  '設置串口串列傳輸速率 9600，校驗位無，停止位 1 位元，數據位元 8 位
COM_CLOSE 2             '關閉串口 2
```

2.13.2 COM_CLOSE

所　屬：命令

語　法：COM_ClOSE port

描　述：指定串口編號，關閉串口。

參　數：port　串口埠號

例　程

```
COM_OPEN 2              '打開串口 2
COM_SET 2, 9600, 0, 1, 8  '設置串口串列傳輸速率 9600，校驗位無，停止位 1 位元，數據位元 8 位
COM_CLOSE 2             '關閉串口 2
```

2.13.3 COM_SET

所　屬：命令

語　法：COM_SET port , baudrate, parity, stopbits, databits

描　述：設置串口通訊參數。

參　數：port　　串口埠號；

Baudrate　串列傳輸速率；　範圍：4800、9600、19200、38400、57600、115200

Parity　　校驗方式；　範圍：無（NONE）、奇（ODD）、偶（EVEN）

Stopbits　停止位；　範圍：1、2

Databits　數據位元；　範圍：7、8

例　程

```
COM_OPEN 2              '打開串口 2
```

```
COM_SET 2, 9600, 0, 1, 8   '設置串口串列傳輸速率 9600，校驗位無，停止位 1 位元，數據位元 8 位
COM_CLOSE 2                '關閉串口 2
```

2.13.4 COM_ReadStream

所　屬：命令

語　法：COM_READSTREAM port, *strarray, num

描　述：串口自由協定讀操作，通過串口讀數據。執行到該指令時，控制器程式會等在該行，直到讀到的位元組個數和 num 參數指定的個數一致時，程式才會執行到下一行。

參　數：port　　串口埠號；

*strarray　存放讀到的資料變數位址，一般為陣列的位址或字串位址

num　　讀取的位元組個數或字元個數

例　程

```
DIM WriteArray(2) AS BYTE={1,2}
DIM ReadArray(2) AS BYTE
DIM WriteStr AS STRING= “OK”
DIM ReV AS ULONG
COM_Open 2   '打開串口 2
COM_SET 2,9600,0,1,8 '設置串口串列傳輸速率 9600，校驗位無，停止位 1 位元，數據位元 8 位
BASE 0,1
SVON
MOVE 2000,30000
WAIT DONE
COM_WriteStream 2,WriteArray(),2 '控制器發出陣列 WriteArray()裡的 2 個位元組資料
COM_WriteStream 2,WriteStr,2     '控制器寫出 WriteStr 中字串
COM_ReadStream 2,ReadArray(),2   '控制器讀兩個位元組資料，未讀到 2 個位元組，程式會停在該行讀
IF(ReadArray(1)=8) THEN     '判斷讀到的 ReadArray(1)值是否為 8
MOVEABS 0,0
END IF
COM_Close 2                '關閉串口 2
```

2.13.5 COM_WriteStream

所　屬：命令

語　法：COM_WriteStream　port, *strarray, num

描　述：串口自由協議寫操作，通過串口寫資料。

參　數：port　　串口埠號；

*strarray　存放寫出的資料變數位址，一般為陣列的位址或字串位址

num　　寫出的位元組個數或字元個數

例　程

```
DIM WriteArray(2) AS BYTE={1,2}
DIM ReadArray(2) AS BYTE
DIM WriteStr AS STRING= “OK”
DIM ReV AS ULONG
COM_Open 2   '打開串口 2
COM_SET 2,9600,0,1,8 '設置串口串列傳輸速率 9600，校驗位無，停止位 1 位元，數據位元 8 位
BASE 0,1
SVON
MOVE 2000,30000
WAIT DONE
COM_WriteStream 2,WriteArray(),2 '控制器發出陣列 WriteArray()裡的 2 個位元組資料
COM_WriteStream 2,WriteStr,2     '控制器寫出 WriteStr 中字串
COM_ReadStream 2,ReadArray(),2   '控制器讀兩個位元組資料，未讀到 2 個位元組，程式會停在該行讀
IF(ReadArray(1)=8) THEN    '判斷讀到的 ReadArray(1)值是否為 8
MOVEABS 0,0
END IF
COM_Close 2              '關閉串口 2
```

2.13.6 COM_ResetBuf

所　屬：命令

語　法：COM_ResetBuf　port

描　述：清除串口緩存區資料。

參　數：port　　串口埠號

2.13.7 TCP_OPEN

所　屬：命令

語　法：TCP_OPEN　no，mode，port[，ipaddress]

描　述：指定 TCP/IP 通訊編號、模式、網路埠號[、IP 地址]，打開一個 TCP 通信連接。相應 TCP 通訊連接埠被打開後，才可以對該網口操作。該指令需要根據本地網口資源進行操作。

參　數：no　TCP 通訊編號。用於控制器內部識別不同 TCP/IP 連接。類型為 ULONG，沒用過的編號可以隨意指定，比如 0,1,2,3,4,5.....

mode　連接模式；範圍：0：控制器作為伺服器，1：控制器作為用戶端。

Port　網路埠號

ipaddress　：IP 位址，控制器作為伺服器時，不需要填該參數。控制器作為客戶端時，該參數填伺服器端網口 IP 地址

例　程

```
TCP_Open 2,1,5024,"192.168.0.11"   '打開一個 TCP 用戶端連接，對接 IP 為 192.168.0.11 的伺服器
TCP_Close 2                        '關閉編號為 2 的網路用戶端端
TCP_Open(1,0,5025)                 '打開一個 TCP 伺服器連接，伺服器處於監聽狀態
TCP_Close 1                        '關閉編號為 1 的網路服務器
```

2.13.8 TCP_CLOSE

所　屬：命令

語　法：TCP_ClOSE no

描　述：指定 TCP 通訊編號，關閉對應 TCP 通訊連接埠

參　數：no　TCP 通訊編號；類型：ULONG

例　程

```
TCP_Open 2,1,5024,"192.168.0.11"   '打開一個 TCP 用戶端連接，對接 IP 為 192.168.0.11 的伺服器
TCP_Close 2                        '關閉編號為 2 的網路用戶端端
TCP_Open(1,0,5025)                 '打開一個 TCP 伺服器連接，伺服器處於監聽狀態
TCP_Close 1                        '關閉編號為 1 的網路服務器
```

2.13.9 TCP_STATUS

所　屬：命令

語　法：value=TCP_STATUS (no)

類　型：ULONG

描　述：檢查 TCP 通訊連接狀態

參　數：no　　TCP 通訊編號

返回值：0：連接不成功；1：連接成功

例　程

```
'請參考 TCP_ReadSTR 或 TCP_WriteSTR 指令
```

2.13.10 TCP_WAIT

所　屬：命令

語　法：TCP_WAIT no [,timeout]

描　述：等待 TCP 連接完成。執行該指令時，程式會等待在該行直到 TCP 通訊連接成功或 timeout 超時，程式才會繼續下一行的執行。

參　數：no　　TCP 通訊編號；

timeout　等待超時時間，單位為 ms。Timeout 時間到後，TCP 通訊連接還未成功，程式會繼續下一行的執行。

例　程

```
'請參考 TCP_ReadSTR 或 TCP_WriteSTR 指令
```

2.13.11 TCP_Check

所　屬：命令

語　法：value=TCP_Check(no)

描　述：獲取 TCP 通訊接收到的字串字元個數。

參　數：no　　TCP 通訊編號；

返回值：字元個數；類型：ULONG

例　程

```
'請參考 TCP_ReadSTR 或 TCP_WriteSTR 指令
```

2.13.12 TCP_ReadSTR

所　屬：命令

語　法：TCP_ReadSTR　no ,strData ,numChars [,strEnd] [,timeout]

描　述：TCP/IP 自由協定讀操作，控制器接收字串指令。執行到該指令時，程式會等在該指令行，直到接收到字元或 timeout 超時，程式才會繼續下一行的執行。

參　數：no　　TCP 通訊編號；類型：ULONG

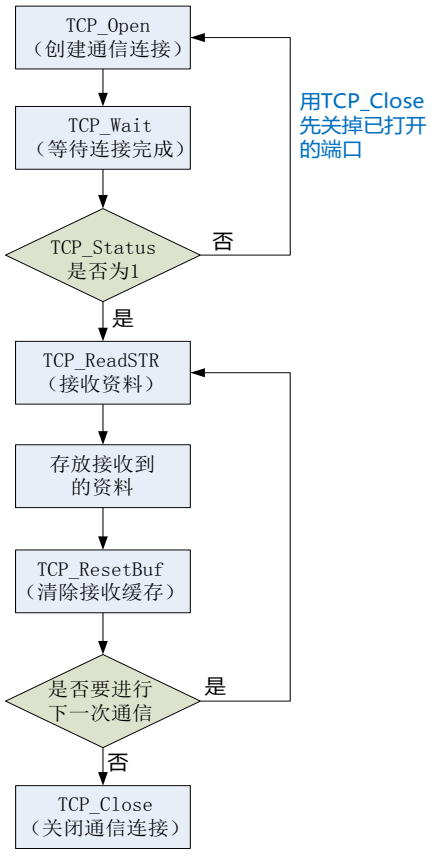
strData　存放接收的字串；類型：String

numChars　接收的字元個數；類型：ULONG。接收字元個數可由 TCP_check 指令獲取得到。

StrEnd　接收字串的結束符。類型：String 該參數不填時，控制器會根據參數 numChars 來確定接收多少個字元。指定結束符時，控制器接收到結束符就會停止這一次接收。

Timeout　接收的超時時間，單位為 ms。Timeout 時間到後，還未接收到字符，系統會斷開當前通訊編號的 TCP 連接。如還需要進行通訊，需重新創建 TCP 連接。

注　意：TCP 接收的相關指令，如 TCP_ReadSTR、TCP_Read、TCP_ReadVR 要注意通信接收緩存的處理。為避免通信接收緩存中的遺留資料影響新的接收資料，接收指令前需用 TCP_ResetBuf 清除通信緩存，不然接收指令會先收到遺留在通信緩存中的資料，導致接收的資料不對。TCP 接收指令的操作可以參照以下流程圖處理。



例　程

```
Dim NumChars as ULONG = 0
Dim StrData as string
TCP_Open (0, 1, 8080, "127.0.0.1")      '創建一個用戶端連接，對接 IP 為"127.0.0.1"的伺服器
TCP_Wait 0                              '等待連接完成
If TCP_STATUS(0) > 0 Then               '確認通訊編號為 0 的連接是否連接成功
    TCP_WriteSTR 0,"I'm ready"          '連接成功後，控制器發送出 I'm ready
    WHILE(1)
        NumChars = TCP_Check(0)                 '取接收到字元個數
        If NumChars > 0 Then
            Print "charNum = ", NumChars        '列印出接收到的字元個數
            TCP_ReadSTR(0, StrData, NumChars)   '將接收到字元放入 StrData
            Print StrData                       '列印出接到到的字串
            If(StrData="ok") THEN       '確認接收到的字串是否是 ok
                EXIT WHILE              'ok：退出接收字元；不是 ok：繼續接收字元
            END If
        End If
        SLEEP 10
    WEND
End If
TCP_CLOSE 0             '斷開通訊編號為 0 的 TCP 連接
```

2.13.13 TCP_WriteSTR

所　屬：命令

語　法：TCP_WriteSTR　no, strData

描　述：TCP/IP 自由協定寫操作，控制器發送出字串指令

參　數：no　　TCP 通訊編號；

　　strData　　發送出去的字串；類型：String

例　程

```
Dim NumChars as ULONG = 0
```

```
Dim StrData as string
TCP_Open (0, 1, 8080, "127.0.0.1")      '創建一個用戶端連接，對接 IP 為"127.0.0.1"的伺服器
TCP_Wait 0                              '等待連接完成
If TCP_STATUS(0) > 0 Then               '確認通訊編號為 0 的連接是否連接成功
    TCP_WriteSTR 0,"I'm ready"          '連接成功後，控制器發送出 I'm ready
    WHILE(1)
        NumChars = TCP_Check(0)                 '取接收到字元個數
        If NumChars > 0 Then
            Print "charNum = ", NumChars       '列印出接收到的字元個數
            TCP_ReadSTR(0, StrData, NumChars)   '將接收到字元放入 StrData
            Print StrData                       '列印出接到到的字串
            If(StrData="ok") THEN       '確認接收到的字串是否是 ok
                EXIT WHILE              'ok：退出接收字元；不是 ok：繼續接收字元
            END If
        End If
        SLEEP 10
    WEND
End If
TCP_CLOSE 0             '斷開通訊編號為 0 的 TCP 連接
```

2.13.14 TCP_Read

所　屬：命令

語　法：TCP_Read　no ,array( ) ,arrayCnt [,timeout]

描　述：TCP/IP 自由協定讀操作，控制器接收資料。控制器會根據 array( )定義的資料類型（byte 或 short 或 long），將每接收到的資料按 1 個位元組或 2 個位元組或 4 個位元組為一個資料依次放入 array( )中。執行到該指令時，程式會等在該指令行，直到接收到資料或 timeout 超時，程式才會繼續下一行的執行。

參　數：no　　TCP 通訊編號；類型：ULONG

Array( )　存放接收資料的陣列；類型：byte、short、long

arrayCnt　接收的資料個數；類型：ULONG。

Timeout　接收的超時時間，單位為 ms。Timeout 時間到後，還未接收到數據，系統會斷開當前通訊編號的 TCP 連接。如還需要進行通訊，需重新創建 TCP 連接。

注　意：TCP 接收的相關指令，如 TCP_ReadSTR、TCP_Read、TCP_ReadVR 要注意通信接收緩存的處理。為避免通信接收緩存中的遺留資料影響新的接收資料，接收指令前需用 TCP_ResetBuf 清除通信緩存，不然接收指令會先收到遺留在通信緩存中的資料，導致接收的資料不對。TCP 接收指令的操作可以參照 TCP_ReadSTR 指令說明中的流程圖來處理。

例　程

```
Dim R_ByteArray(0 to 1) as BYTE
Dim R_ShortArray(0 to 1) as SHORT
Dim R_LongArray(0 to 1) as  LONG
TCP_Open (0, 1, 8080, "127.0.0.1")      '創建一個用戶端連接，對接 IP 為"127.0.0.1"的伺服器
TCP_Wait 0                              '等待連接完成
If TCP_STATUS(0) > 0 Then               '確認通訊編號為 0 的連接是否連接成功
    '接收到的 2 個 Byte 資料分別存入 R_ByteArray(0)，R_ByteArray(1)
    TCP_READ 0,R_ByteArray(),2
    PRINT R_ByteArray(0),R_ByteArray(1)
    '接收到的前 2 個 Byte 資料組成 Short 類型資料存入 R_ShortArray(0)
    '接收到的後 2 個 Byte 資料組成 short 類型資料存入 R_ShortArray(1)
    TCP_READ 0,R_ShortArray(),2
    PRINT R_ShortArray(0),R_ShortArray(1)
    '接收到的前 4 個 Byte 資料組成 long 類型資料存入 R_LongArray(0)
    '接收到的後 4 個 Byte 資料組成 long 類型資料存入 R_LongArray(1)
    TCP_READ 0,R_LongArray(),2
    PRINT R_LongArray(0),R_LongArray(1)
End If
TCP_CLOSE 0             '斷開通訊編號為 0 的 TCP 連接
```

2.13.15 TCP_Write

所　屬：命令

語　法：TCP_Write　no, array( ),arrayCnt

描　述：TCP/IP 自由協定寫操作，控制器發送出資料。發送的資料可以選擇 byte、short、long 三種資料類型。控制器會根據資料類型按 byte 的數據送出。如果是 short、long 類型，發出去的 byte 會以低位元組到高位元組的順序發送。

參　數：no　　TCP 通訊編號；

Array( )　　發送出去資料的陣列；類型：byte，short，long

ArrayCnt　　發送出去的資料個數。

例　程

```
Dim W_ByteArray(0 to 1) as BYTE ={-75,121}
Dim W_ShortArray(0 to 1) as SHORT ={135,32753}
Dim W_LongArray(0 to 1) as  LONG={-24,175024}
TCP_Open (0, 1, 8080, "127.0.0.1") '創建一個用戶端連接，對接 IP 為"127.0.0.1"的伺服器
TCP_Wait 0                          '等待連接完成

'相隔 3 秒，依次發送 W_ByteArray(),W_ShortArray(),W_LongArray()
If TCP_STATUS(0) > 0 Then           '確認通訊編號為 0 的連接是否連接成功
    '伺服器端收到的資料為十六進位數：B5 79。對應-75,121
    TCP_Write 0,W_ByteArray(),2
    SLEEP 3000
    '伺服器端收到的資料為十六進位數：87 00 F1 7F。對應 135,32753
    TCP_Write 0,W_ShortArray(),2
    SLEEP 3000
    '伺服器端收到的資料為十六進位數：E8 FF FF FF B0 AB 02 00。對應-24,175024
    TCP_Write 0,W_LongArray(),2
End If
TCP_CLOSE 0             '斷開通訊編號為 0 的 TCP 連接
```

2.13.16 TCP_ReadVR

所　屬：命令

語　法：TCP_ReadVR　no ,VR_StartIndex ,VRCnt, format [,timeout]

描　述：TCP/IP 自由協定讀操作，控制器用 VR 變數接收資料。控制器會根據 format 指定的資料類型（byte 或 short 或 long），將每接收到的資料按 1 個位元組或 2 個位元組或 4 個位元組為一個資料依次放入 VR 變數中。執行到該指令時，程式會等在該指令行，直到接收到資料或 timeout 超時，程式才會繼續下一行的執行。

參　數：

no	TCP 通訊編號；類型：ULONG
VR_StartIndex	存放接收資料的起始 VR；類型：byte、short、long
VRCnt	接收的資料個數；類型：ULONG
format	指定存放接收資料的類型。 0：byte 1：short 2：long
timeout	接收的超時時間，單位為 ms。Timeout 時間到後，還未接收到資料，系統會斷開當前通訊編號的 TCP 連接。如還需要進行通訊，需重新創建 TCP 連接。

注　意：TCP 接收的相關指令，如 TCP_ReadSTR、TCP_Read、TCP_ReadVR 要注意通信接收緩存的處理。為避免通信接收緩存中的遺留資料影響新的接收資料，接收指令前需用 TCP_ResetBuf 清除通信緩存，不然接收指令會先收到遺留在通信緩存中的資料，導致接收的資料不對。TCP 接收指令的操作可以參照 TCP_ReadSTR 指令說明中的流程圖來處理。

例　程

```
TCP_Open (0, 1, 8080, "127.0.0.1") '創建一個用戶端連接，對接 IP 為"127.0.0.1"的伺服器
TCP_Wait 0                          '等待連接完成
If TCP_STATUS(0) > 0 Then           '確認通訊編號為 0 的連接是否連接成功

    '接收到的 2 個 Byte 資料分別存入 VR(0)，VR(1)
    TCP_ReadVR 0,0,2,0
    PRINT VR(0),VR(1)

    '接收到的前 2 個 Byte 資料組成 Short 類型資料存入 VR(2)
    '接收到的後 2 個 Byte 資料組成 short 類型資料存入 VR(3)
```

```
    TCP_ReadVR 0,2,2,1
    PRINT VR(2),VR(3)

    '接收到的前 4 個 Byte 資料組成 long 類型資料存入 VR(4)
    '接收到的後 4 個 Byte 資料組成 long 類型資料存入 VR(5)
    TCP_ReadVR 0,4,2,2
    PRINT VR(4),VR(5)
End If
TCP_CLOSE 0              '斷開通訊編號為 0 的 TCP 連接
```

2.13.17 TCP_WriteVR

所　屬：命令

語　法：TCP_WriteVR　no, VR_StartIndex ,VRCnt, format

描　述：TCP/IP 自由協定寫操作，控制器把 VR 變數中的資料發送出去。發送的資料可以選擇 byte、short、long 三種資料類型。控制器會根據資料類型按 byte 的數據送出。

參　數：no　　TCP 通訊編號；

VR_StartIndex　發送資料的起始 VR；類型：byte、short、long

VRCnt　　發送的資料個數；類型：ULONG

format　　指定發送資料的類型。

0：byte

1：short

2：long

注　意　如果發送出去的 VR 變數資料是浮點型資料，接收端接收到的資料會失真。如要發送浮點數據，請用 TCP_WriteSTR 指令，用字串形式發送出去，接收端接收到字串後再轉資料類型來接收浮點數據。

例　程

```
VR(0)=-75
VR(1)=121
VR(2)=135
VR(3)=32753
```

```
VR(4)=-24
VR(5)=175024
TCP_Open (0, 1, 8080, "127.0.0.1") '創建一個用戶端連接，對接 IP 為"127.0.0.1"的伺服器
TCP_Wait 0                          '等待連接完成

'相隔 3 秒，依次發送 VR(0)、VR(1) ; VR(2)、VR(3) ; VR(4)、VR(5)
If TCP_STATUS(0) > 0 Then           '確認通訊編號為 0 的連接是否連接成功
'將 VR(0),VR(1)發送出去，伺服器端收到的資料為十六進位數：B5 79。對應-75,121
    TCP_WriteVR 0,0,2,0
    SLEEP 3000
'將 VR(2),VR(3)發送出去，伺服器端收到的資料為十六進位數：87 00 F1 7F。對應 135,32753
    TCP_WriteVR 0,2,2,1
    SLEEP 3000
'將 VR(4),VR(5)發送出去，伺服器端收到的資料為十六進位數：E8 FF FF FF B0 AB 02 00。對應-24,175024
    TCP_WriteVR 0,4,2,2
End If
TCP_CLOSE 0             '斷開通訊編號為 0 的 TCP 連接
```

2.13.18 TCP_ResetBuf

所　屬：命令

語　法：TCP_ResetBuf　no

描　述：清除 TCP 緩存區數據。

參　數：no　TCP 通訊連接編號

注　意：TCP 接收的相關指令，如 TCP_ReadSTR、TCP_Read、TCP_ReadVR 要注意通信接收緩存的處理。為避免通信接收緩存中的遺留資料影響新的接收資料，接收指令前需用 TCP_ResetBuf 清除通信緩存，不然接收指令會先收到遺留在通信緩存中的資料，導致接收的資料不對。

2.13.19 MB_OPEN

所屬：命令

語　法：VALUE = MB_OPEN (mbindex, connectmode, ip/comid, port/baudrate,

deviceID[,parity,stopbits,databits])

類　型：BOOLEAN。

描　述：指定 modbus 通訊編號、模式、網路埠號、IP port 或串列傳輸速率、設備 ID，打開一個 modbus tcp 連接或 modbus rtu 串口。相應通訊連接埠或串口被打開後，才可以對該埠或串口進行操作。該指令需要根據本地資源進行操作。

參　數：mbindex 設定一個 modbus 通信序號，0~4294967294

Connectmode 連接模式　0：Modbus RTU，1：Modbus Tcp client

ip/comid IP　地址或 com 埠號

port/baudrateIP　port 或串列傳輸速率

deviceID　Device ID 範圍 1~247

parity 奇偶位　0：none，1：even

stopbits 停止位　0-1，1-1.5，2-2

databits 數據位元　7/8

返回值：　TRUE—打開成功，FALSE—打開失敗

例　程

```
'完整例程可參考 MB_GETHDREG 指令
MB_OPEN(0, 1, "127.0.0.1", 502, 1)   '打開一個 modbus tcp 用戶端連接，
                                      '對接 IP 為 127.0.0.1 的伺服器
MB_CLOSE(0)                          '關閉編號為 0 的網路用戶端端
```

2.13.20 MB_CLOSE

所屬：命令

語　法：VALUE = MB_CLOSE(mbindex)

類　型：BOOLEAN。

描　述：關閉指定序號的 modbus 連接。

參　數：mbindex　通訊編號

返回值：　TRUE—關閉成功，FALSE—關閉失敗

例　程

```
'完整例程可參考 MB_GETHDREG 指令
MB_CLOSE(0)        '關閉編號為 0 的網路用戶端端
```

2.13.21 MB_STATUS

所屬：命令

語　法：VALUE = MB_STATUS(mbindex)

類　型：ULONG。

描　述：獲取 modbus 連接狀態

參　數：mbindex　通訊編號。

返回值：0：連接不成功；1：連接成功

例　程

```
MB_STATUS(0)        '獲取編號為 0 的 modbus 連接狀態
```

2.13.22 MB_SETCOIL

所屬：命令

語　法 1：設置單個線圈數值:

VALUE = MB_SETCOIL(mbindex, m_start_address, Value)

語　法 2：設置多個線圈數值:

VALUE = MB_SETCOIL(mbindex, m_start_address, ValueArray(),　DataCnt)

類　型：BOOLEAN。

描　述：設置單個或多個線圈數值。

參　數：mbindex　通訊編號

m_start_address　modbus 相對起始位址(首地址為 0）

Value　　設定單個值

ValueArray()　　設定多個值

DataCnt　需傳輸的數值個數

返回值：TRUE—設置成功，FALSE—設置失敗

例　程

```
'請參考 MB_GETCOIL 指令
```

2.13.23 MB_GETCOIL

所屬：命令

語　法 1：獲取單個線圈數值:

VALUE =MB_GETCOIL(mbindex, m_start_address, OutputValue)

語　法 2：獲取多個線圈數值:

VALUE=MB_GETCOIL(mbindex, m_start_address, OutputValueArray(),
DataCnt)

類　型：BOOLEAN。

描　述：獲取單個或多個線圈數值。

參　數：mbindex　通訊編號

m_start_address　modbus 相對起始位址(首地址為 0）

OutputValue　讀取時用於接收值

OutputValueArray　用於接收獲取到的多個數值的陣列

DataCnt　需傳輸的數值個數

返回值：TRUE—獲取成功，FALSE—獲取失敗

例　程

```
'設置或獲取單個線圈數值
DIM coil_data AS BYTE = 0
DIM As INTEGER mb_Index =0, i
DIM As USHORT startAddress = 0, data_count = 3
MB_SETCOIL(mb_Index, startAddress, 1)
MB_GETCOIL(mb_Index, startAddress, coil_data)
IF coil_data<>1 THEN
 PRINT "Sigle coil failed."
END IF
'設置或獲取多個線圈數值
DIM temp_in(11) As BYTE = {1,1,1,0,0,1,1,1,0,0,1}
MB_SETCOIL(mb_Index, startAddress, temp_in(), data_count)
DIM temp_out(11) As BYTE
MB_GETCOIL(mb_Index, startAddress, temp_out(), data_count)
FOR i As INTEGER = 0 to data_count-1
 PRINT "Coil address ";startAddress+i;" data = ";temp_out(i)
 IF temp_out(i)<>temp_in(i) THEN
```

```
  PRINT "Multiple coil failed."
 END IF
NEXT i
```

2.13.24 MB_GETINPUT

所屬：命令

語　法 1：獲取單個離散輸入值:

VALUE=MB_GETINPUT(mbindex, m_start_address, OutputValue)

語　法 2： 獲取多個離散輸入值:

VALUE=MB_GETINPUT (mbindex, m_start_address,OutputValueArray(),
DataCnt)

類　型：BOOLEAN。

描　述：獲取單個或多個離散輸入值。

返回值：TRUE—獲取成功，FALSE—獲取失敗

參　數：mbindex　通訊編號

m_start_address　modbus 相對起始位址(首地址為 0）

OutputValue　讀取時用於接收值

OutputValueArray　用於接收獲取到的多個數值的陣列

DataCnt　需傳輸的數值個數(非地址個數)，實際地址個數依照傳入的資料類型而定

例　程

```
'設置或獲取多個離散輸入值
DIM temp_input(11) As BYTE
MB_GETINPUT(mb_Index, startAddress, temp_input(), data_count)
FOR i As INTEGER = 0 to data_count-1
 PRINT "Input bit address ";startAddress + i;" data = ";temp_input(i)
NEXT i
```

2.13.25 MB_SETHDREG

所屬：命令

語　法 1：設置單個 Holding register 值:

VALUE=MB_SETHDREG(mbindex,m_start_address,Value[, DataType])

語　法 2：設置多個 Holding register 值:

VALUE =MB_SETHDREG(mbindex, m_start_address, ValueArray(),　DataCnt)

類　型：BOOLEAN。

描　述：設置單個或多個 Holding register 值。

參　數：mbindex　　通訊編號

　　m_start_address　modbus 相對起始位址(首地址為 0）

Value　　設定單個值

ValueArray()　　設定多個值

DataCnt 需傳輸的數值個數(非地址個數) ，實際地址個數依照傳入的資料類型而定。

DataType：資料類型，目前支持以下幾種

DATATYPE_U16　0

DATATYPE_I16　 1

DATATYPE_U32　2

DATATYPE_I32　 3

DATATYPE_F32　4

DATATYPE_F64　5

返回值：TRUE—設置成功，FALSE—設置失敗

例　程

```
'也可參考 MB_GETHDREG 指令例程
DIM As INTEGER mb_Index =0, i
DIM As USHORT startAddress = 3, data_count = 3
DIM sData As SHORT
DIM fData AS SINGLE
MB_OPEN(mb_Index, 1, "127.0.0.1", 502, 1)
MB_SETHDREG(mb_Index, startAddress, -10, DATATYPE_I16)
MB_GETHDREG(mb_Index, startAddress, sData)
PRINT "Short data:";sData
MB_SETHDREG(mb_Index, startAddress, -10.123, DATATYPE_F32)
MB_GETHDREG(mb_Index, startAddress, fData)
PRINT "Float data:";fData
```

```
MB_CLOSE(0)
SLEEP 1000
```

2.13.26 MB_GETHDREG

所屬：命令

語　法 1：獲取單個 Holding register 值:

VALUE =MB_GETHDREG(mbindex, m_start_address, OutputValue)

語　法 2：獲取多個 Holding register 值:

VALUE=MB_GETHDREG(mbindex,m_start_address,OutputValueArray(), DataCnt)

類　型：BOOLEAN。

描　述：獲取單個或多個 Holding register 值。

參　數：mbindex　通訊編號

m_start_address　modbus 相對起始位址(首地址為 0）

OutputValue　讀取時用於接收值

OutputValueArray　用於接收獲取到的多個數值的陣列

DataCnt　需傳輸的數值個數(非地址個數) ，實際地址個數依照傳入的資料類型而定

返回值：TRUE—獲取成功，FALSE—獲取失敗

例　程

```
Dim As INTEGER mb_Index =0, i
Dim As USHORT startAddress = 0, data_count = 3
IF MB_OPEN(mb_Index, 1, "127.0.0.1", 502, 1)=FALSE THEN
 PRINT "Open modbus failed."
END IF
'Write &Read Ushort register value
Dim usData As USHORT
MB_SETHDREG(mb_Index, startAddress, 65534)
MB_GETHDREG(mb_Index, startAddress, usData)
PRINT "Holding register address";startAddress;", Ushort data =";usData
IF usData<>65534 THEN
```

```
 PRINT "Ushort register failed."
END IF
usData = 0
MB_GETINREG(mb_Index, startAddress, usData)
PRINT "Input register address";startAddress;", Uhsortdata =";usData
MB_CLOSE(0)
SLEEP 1000
```

2.13.27 MB_GETINREG

所屬：命令

語　法 1：獲取單個 Input register 值:

VALUE=MB_GETINREG(mbindex, m_start_address, OutputValue)

語　法 2：獲取多個 Input register 值:

VALUE=MB_GETINREG(mbindex,m_start_address, OutputValueArray(),
DataCnt)

類　型：ULONG。

描　述：獲取單個或多個 Input register 值。

參　數：mbindex　通訊編號

m_start_address　modbus 相對起始位址(首地址為 0）

OutputValue　讀取時用於接收值

OutputValueArray　用於接收獲取到的多個數值的陣列

DataCnt　需傳輸的數值個數(非地址個數)，實際地址個數依照傳入的資料類型而定

返回值：TRUE—獲取成功，FALSE—獲取失敗

例　程

```
'請參考 MB_GETHDREG 指令例程
```

2.14 字串處理

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.14.1	ASC	返回字串中字元的 ASCII 碼	×	×
2.14.2	CHR	返回用 ASCII 碼表達的值對應的字元	×	×
2.14.3	HEX	返回數值的十六進位結果	×	×
2.14.4	INSTR	查找字串中第一次出現的字元或者字串	×	×
2.14.5	LCASE	將字串中的字母全部轉變成小寫字母返回	×	×
2.14.6	LEFT	返回字串從左開始指定字元個數的子串	×	×
2.14.7	LEN	返回字串的長度（字元個數）或者資料類型的長度（位元組數）	×	×
2.14.8	MID	返回一個字串的子字串	×	×
2.14.9	RIGHT	返回字串從右開始指定字元個數的子串	×	×
2.14.10	STR	將一個數轉換成字串	×	×
2.14.11	UCASE	將字串中的字母全部轉變成大寫字母返回	×	×
2.14.12	VAL	將字串轉換成一個數值	×	×
2.14.13	PARSESTR	按使用者指定的分隔符號解析字串	×	×

2.14.1 ASC

語　法：value=ASC(string [,position])

描　述：返回字串中字元的 ASCII 碼

參　數：string　　字串

position　需返回 ASCII 碼字元在字串中的位置，缺省值為 1

例　程

```
PRINT ASC("A")          '結果為 65
PRINT ASC("ABC",1)      '列印第一個字母 A 的 ASCII 碼，結果為 65
PRINT ASC("ABC",2)      '列印第二個字母 B 的 ASCII 碼，結果為 66
PRINT ASC("ABC",3)      '列印第三個字母 C 的 ASCII 碼，結果為 65
PRINT ASC("ABC")        '缺省位置值為 1，即列印 A 的 ASCII 碼，結果為 65
```

2.14.2　CHR

語　法：value=CHR(number)

描　述：返回用 ASCII 碼表達的值對應的字元

參　數：number　ASCII 碼值

例　程

```
PRINT CHR(97)           '97 對應的字元為 a，列印結果為 a
PRINT CHR(65)           '65 對應的字元為 A，列印結果為 A
```

ASCII 碼表

32	空格	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	

2.14.3 HEX

語　法：value=HEX(number [,digits])

描　述：返回數值的十六進位結果

參　數：number　數值

digits　返回由低位元到高位的位元數

例　程

'十進位 54321 對應的十六進位數為 D431

Print Hex(54321)　'列印結果為 D431

Print Hex(54321, 2)　'列印結果為 31

Print Hex(54321, 5)　'列印結果為 0D431

2.14.4 INSTR

語　法：value=INSTR([start,]　string,　[Any]　substring)

描　述：查找字串中第一次出現的字元或者字串

參　數：start　　從第幾個字元開始查找

string　在 string 這個字串中查找字元或字串

Any　　加上這個關鍵字後，string 中先找到 substring 中的任意一個字元就會返回相應值

substring　需查找的字元或字串

例　程

```
Print InStr(2,"abcdefg", "a")   '列印資訊為 0，因從字串的第 2 位開始找，找不到 a，返回 0
Print InStr("abcdefg", "de")    '列印資訊為 4，第 4 位找到 de
Print InStr("abcdefg", "h")     '列印資訊為 0，字串中沒有 h
Print InStr("abcdefg", Any "fbc")   '列印資訊為 2，因加了 any 關鍵字，所以先找到 b，b 為第 2 位
```

2.14.5 LCASE

語　法：value=LCASE(string)

描　述：將字串中的字母全部轉變成小寫字母返回

參　數：string　需要轉換的字串

例　程

```
Print Lcase("AeeE")     '列印結果為 aeee
```

2.14.6 LEFT

語　法：value=LEFT(string,number)

描　述：返回字串從左開始指定字元個數的子串

參　數：string　　需要轉換的字串

number　字元個數

例　程

```
Print LEFT("Hello Advantech",5)          '列印資訊為 Hello
```

2.14.7 LEN

語　法：value=LEN(expression)

描　述：返回字串的長度（字元個數）或者資料類型的長度（位元組數）

參　數：expression　如果是字串，返回字元個數；如果是資料類型，返回位元組數

例　程

```
Print Len("hello world") '列印結果為 11，共 11 個字元
Print Len(Integer)    '列印結果為 4，integer 這個資料類型為 4 個位元組
```

2.14.8 MID

語　法：value=MID(string，start [,number])

描　述：返回一個字串的子字串

參　數：string　需要轉換的字串

start　返回的子字串的起始轉換位元

number　子字串的字元個數。如不填，則返回從 start 位元後的所有字元

例　程

```
Print Mid("abcdefg", 3, 2)    '列印結果為 cd
Print Mid("abcdefg", 3)       '列印結果為 cdefg
Print Mid("abcdefg", 2, 1)    '列印結果為 b
```

2.14.9 RIGHT

語　法：value=RIGHT(string,number)

描　述：返回字串從右開始指定字元個數的子字串

參　數：string　需要轉換的字串

number　字元個數

例　程

```
Print RIGHT("Hello Advantech",9)       '列印資訊為 Advantech
```

2.14.10 STR

語　法：value=STR(Numeric)

描　述：將一個數轉換成字串

參　數：Numeric　數值運算式

例　程

```
VR(100)=100.32
PRINT STR(VR(100))      '列印結果為字串"100.32"
```

2.14.11 UCASE

語　法：value=UCASE(string)

描　述：將字串中的字母全部轉變成大寫字母返回

參　數：string　需要轉換的字串

例　程

```
Print Ucase("AeeE")      '列印結果為 AEEE
```

2.14.12 VAL

語　法：value=VAL(string)

描　述：將字串轉換成一個數值，字串轉換將從左到右按字元轉換，如果先遇到非數值的字元，轉換出來的數值將是 0。

參　數：string　字串

例　程

```
DIM AS STRING str1,str2
str1="e3t"                    '因先遇到非數值字元 e，所有列印結果為 0
str2="325.32"
PRINT VAL(str1),VAL(str2)     '列印結果為 0,325.32
```

2.14.13 PARSESTR

語　法：NumStr=ParseSTR(StrInput, StrTokens( ), StrDelimits)

描　述：按使用者指定的分隔符號解析字串。

參　數：StrInput　　輸入的需分隔的字串

StrTokens( )　存放分隔出的有效字串陣列

StrDelimits　指定的分隔符號

返回值：NumStr　　分隔出的有效字串個數。類型：ULONG

例　程

```
Dim StrInput as string = "Hi,MAS,Controller,!"
Dim StrDelimits as string = ","
Dim NumStr as ULONG
Dim StrTokens(0 to 3) as string
NumStr = ParseStr(StrInput, StrTokens(), StrDelimits)
print "num = ", NumStr        '列印出 num=4，有效分隔出 4 個字串
Dim i as Integer = 0
for i= 0 to (NumStr-1)
    print   StrTokens(i)      '字串陣列依次列印出 Hi MAS Controller !
next i
```

2.15 工藝模組指令

2.15.1 氣/油缸控制

氣/油缸在自動化設備中非常常見，很好的對氣/油缸進行控制在系統開發中顯得很重要。本章節介紹了 Motion Basic 簡單易使用的氣/油缸控制指令，通過簡單配置，可以很方便的實現設備中常見的氣/油缸控制。為簡要說明，本章節指令說明中統一用“氣缸”來代替“氣/油缸”，“用氣缸前進”、“氣缸後退來”表示氣缸動作的兩個方向運動。

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.15.1.1	CYL_BASE	該指令後面所有的氣缸指令和參數設置、讀取都基於該指令選定的氣缸	√	×
2.15.1.2	CYL_FwDoneType	氣缸前進到位方式	√	×
2.15.1.3	CYL_BwDoneType	氣缸後退到位方式	√	×
2.15.1.4	CYL_FwTime	CYL_FwDoneType 中涉及到延時到位方式的延時時間	√	×
2.15.1.5	CYL_BwTime	CYL_BwDoneType 中涉及到延時到位方式的延時時間	√	×
2.15.1.6	CYL_FwAlmTime	氣缸前進開始到到位的最大時間	√	×
2.15.1.7	CYL_BwAlmTime	氣缸後退開始到到位的最大時間	√	×
2.15.1.8	CYL_FwEncValue	氣缸前進到位編碼器值	√	×
2.15.1.9	CYL_BwEncValue	氣缸後退到位編碼器值	√	×
2.15.1.10	CYL_Status	氣缸當前狀態	√	×
2.15.1.11	CYL_AlmReset	重定氣缸的狀態到重定模式	√	×
2.15.1.12	CYL_Move	執行氣缸前進或後退動作	√	×
2.15.1.13	CYL_Stop	停止氣缸動作	√	×

2.15.1.1 CYL_BASE

語　法：CYL_BASE　(cyl_no) [,second cyl][,third cyl] …

描　述：為了簡化程式設計，可以用該指令選擇要參與運動的氣缸號，其後的指令就沒必

要填寫所有氣缸控制的參數，只填寫參與運動的氣缸參數即可。氣缸號要按順序填寫，氣缸號可以是 1 個，也可以是 2 個、3 個...

參　數： cyl_no 氣缸號，由硬體設定決定對應的實體氣缸控制；範圍：根據控制器實際硬體決定

例　程

```
CYL_BASE 0,1,2,3
Cyl_FwDoneType=0          '設置 4 個氣缸前進到位方式為延時到位
Cyl_BwDoneType=1          '設置 4 個氣缸後退到位方式為限位元到位
CYL_MOVE 1,0,1,1          '氣缸 0,1,2,3 分別執行前進、後退、前進、前進動作
Wait CYLDONE              '等待氣缸 0,1,2,3 動作到位完成
CYL_BASE 1
CYL_MOVE 1                '氣缸 1 執行前進動作
```

2.15.1.2 CYL_FwDoneType

語　法：CYL_FwDoneType= value

類　型：ULONG

描　述：設置/讀取氣缸前進到位方式

範　圍 :如下設定值，預設值 0

0：延時到位：氣缸動作後，延時指定時間到，即認為氣缸動作到位

1：限位到位：氣缸動作後，遇到指定限位有效，即認為氣缸動作到位

2：（限位+延時）到位：氣缸動作後，遇到指定限位有效，再延時指定時間後，即認為氣缸動作到位

3：（延時+限位）到位：氣缸動作後，延時指定時間後，再檢測到指定限位有效，即認為氣缸動作到位

4： 編碼器到位：氣缸動作後，編碼器到限定數值後，即認為氣功動作到位

例　程

```
CYL_BASE 0
CYL_FwDoneType=0 '設置氣缸 0 前進到位方式為延時到位
```

2.15.1.3 CYL_BwDoneType

語　法：CYL_BwDoneType= value

類　型：ULONG

描　述：設置/讀取氣缸後退到位方式

範　圍 :如下設定值，預設值 0

0：延時到位：氣缸動作後，延時指定時間到，即認為氣缸動作到位

1：限位到位：氣缸動作後，遇到指定限位有效，即認為氣缸動作到位

2：（限位+延時）到位：氣缸動作後，遇到指定限位有效，再延時指定時間後，即認為氣缸動作到位

3：（延時+限位）到位：氣缸動作後，延時指定時間後，再檢測到指定限位有效，即認為氣缸動作到位

4： 編碼器到位：氣缸動作後，編碼器到限定數值後，即認為氣功動作到位

例　程

```
CYL_BASE 0
CYL_BwDoneType=0 '設置氣缸 0 後退到位方式為延時到位
```

2.15.1.4 CYL_FwTime

語　法：CYL_FwTime= value

類　型：ULONG

描　述：設置/讀取 CYL_FwDoneType 中涉及到延時到位方式的延時時間。

範　圍 :ULONG 類型範圍，預設值 5000（ms）

例　程

```
CYL_BASE 0
CYL_FwTime =1000 '設置氣缸 0 前進到位方式中的延時時間為 1000 毫秒
```

2.15.1.5 CYL_BwTime

語　法：CYL_BwTime= value

類　型：ULONG

描　述：設置/讀取 CYL_BwDoneType 中涉及到延時到位方式的延時時間。

範　圍 :ULONG 類型範圍，預設值 5000（ms）

例　程

```
CYL_BASE 0
CYL_BwTime =1000 '設置氣缸 0 後退到位方式中的延時時間為 1000 毫秒
```

2.15.1.6 CYL_FwAlmTime

語　法：CYL_FwAlmTime= value

類　型：ULONG

描　述：設置/讀取氣缸前進開始到到位的最大時間，如超過該時間前進動作還未到位，會發生內部報警，CYL_Status 屬性值變為 9：氣缸到位超時。

範　圍：ULONG 類型範圍，預設值 20000（ms）

例　程

```
CYL_BASE 0
CYL_FwAlmTime =1000 '設置氣缸 0 前進最大到位時間為 1000 毫秒
```

2.15.1.7 CYL_BwAlmTime

語　法：CYL_BwAlmTime= value

類　型：ULONG

描　述：設置/讀取氣缸後退開始到到位的最大時間，如超過該時間前進動作還未到位，會發生內部報警，CYL_Status 屬性值變為 9：氣缸到位超時。

範　圍：ULONG 類型範圍，預設值 20000（ms）

例　程

```
CYL_BASE 0
CYL_BwAlmTime =1000 '設置氣缸 0 後退最大到位時間為 1000 毫秒
```

2.15.1.8 CYL_FwEncValue

語　法：CYL_FwEncValue= value

類　型：Double

描　述：設置/讀取氣缸前進動作的到位方式為編碼器到位時，指定的到位編碼器值。

範　圍：Double 類型範圍，預設值 0

例　程

```
CYL_BASE 0
CYL_FwEncValue =1000 '設置氣缸 0 前進到位編碼器值為 1000 個脈衝當量
```

2.15.1.9 CYL_BwEncValue

語　法：CYL_BwEncValue= value

類　型：Double

描　述：設置/讀取氣缸後退動作的到位方式為編碼器到位時，指定的到位編碼器值。

範　圍：Double 類型範圍，預設值 0

例　程

```
CYL_BASE 0
CYL_BwEncValue =1000 '設置氣缸 0 後退到位編碼器值為 1000 個脈衝當量
```

2.15.1.10 CYL_Status

語　法：value=CYL_Status（唯讀）

類　型：ULONG

描　述：讀取氣缸當前狀態。狀態為 9 時，氣缸不能再正常執行動作。

返回值：如下

0：復位：原始狀態。執行 CYL_Stop、CYL_AlmReset 後的氣缸狀態都為重定模式

1：前進到位：

2：後退到位

3：前進中

4：後退中

5：保留

6：保留

7：保留

8：保留

9：超過到位時間報警

例　程

```
Dim A As ULONG
CYL_BASE 0
A = CYL_Status   '將氣缸 0 的當前狀態賦值給變數 A
```

2.15.1.11 CYL_AlmReset

語　法 1：CYL_AlmReset

語　法 2：CYL_AlmReset CYL(no)

描　述：BASE 氣缸列表的氣缸或指定氣缸，重定氣缸的狀態到重定模式。

參　數：no　氣缸號；　範圍：根據控制器實際硬體決定。

例　程

```
CYL_BASE 0,1,2
CYL_AlmReset          '復位氣缸 0、1、2 的狀態
CYL_AlmReset cyl(1)   '復位氣缸 1 的狀態
```

2.15.1.12 CYL_Move

語　法 1：CYL_Move dir

語　法 2：CYL_Move CYL(no) , dir

描　述：BASE 氣缸列表的氣缸或指定氣缸，執行氣缸動作。

參　數：dir　氣缸動作方向。0：　氣缸後退　；　1：氣缸前進

no　氣缸號；　範圍：根據控制器實際硬體決定。

例　程

```
CYL_BASE 0,1,2,3
CYL_MOVE 1,0,1,1          '氣缸 0,1,2,3 分別執行前進、後退、前進、前進動作
Wait CYLDONE              '等待氣缸 0,1,2,3 動作到位完成
CYL_Move cyl(1),1         '氣缸 1 執行前進動作
```

2.15.1.13 CYL_Stop

語　法 1：CYL_Stop

語　法 2：CYL_Stop CYL(no)

描　述：BASE 氣缸列表的氣缸或指定氣缸，停止氣缸動作。

參　數：no　氣缸號；　範圍：根據控制器實際硬體決定。

注　意：該指令僅適用對雙線圈電磁閥控制的氣缸控制。

例　程

```
CYL_BASE 0,1,2,3
CYL_Stop              '停止氣缸 0,1,2,3 的動作
CYL_Stop cyl(6),1     '停止氣缸 6 的動作
```

2.15.2 PATHLINK

本文主要說明如何實現 XYTable 追隨傳送帶上的工件進行加工。加工動作包含：點膠，鎖螺絲，取放等動作。追隨加工的動作，我們簡稱 Pathlink。

整個操作流程大致如下：

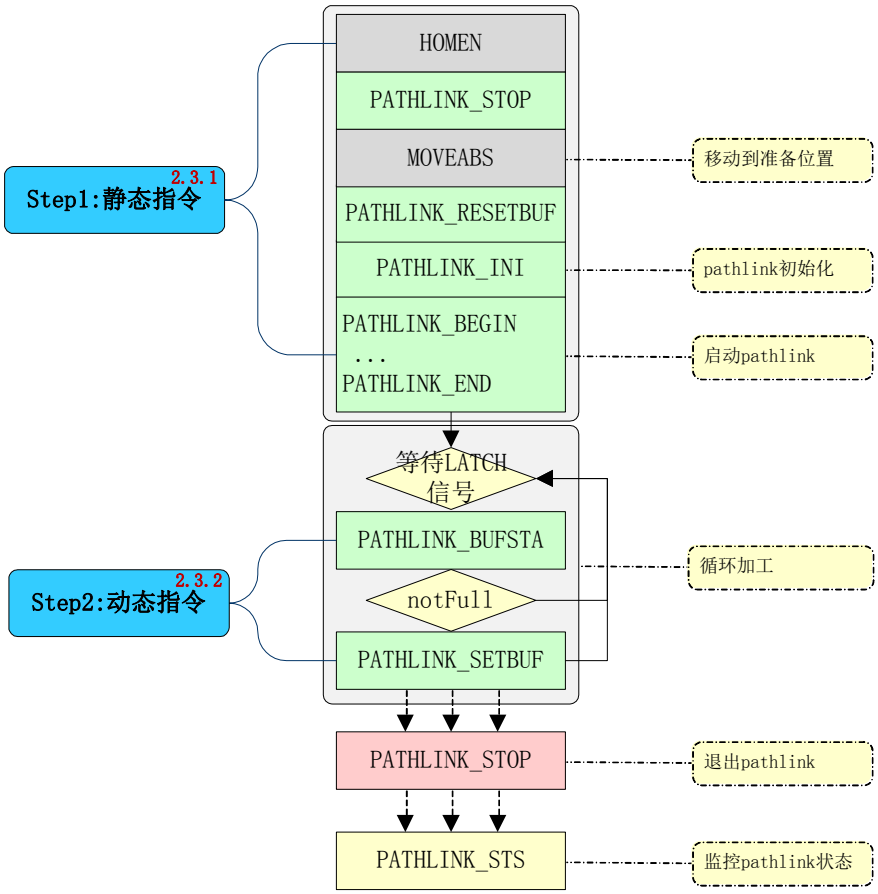


圖 2.15.1 PATHLINK 操作流程圖

靜態指令：把只需操作一次的指令記為靜態指令。

動態指令：把需要多次操作的指令記為動態指令。每來一個工件，就需要操作一次。

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.15.2.1	PATHLINK_INI	設置 PATH LINK 初始信息,包含同步位置設定, 圖像 MARK 點座標信息等	×	×
2.15.2.2	PATHLINK_BEGIN	Pathlink 插補軌跡路徑起始符	×	×
2.15.2.3	PATHLINK_END	Pathlink 插補軌跡路徑結束符	×	×
2.15.2.4	PATHLINK_SETBUF	設定加工時的每次相機拍照時得到 MARK 的位置和角度資訊以及鎖存位置(理論位置/編碼器位置)寫入 Buffer 中	×	×
2.15.2.5	PATHLINK_STOP	調用 Acm_PathLinkStop, 解除主軸和從軸的同步關係	√	×
2.15.2.6	PATHLINK_BUFSTATUS	用於獲取加工時用於存儲拍照時得到 MARK 信息和 latch 數據的 buffer 的狀態	×	×
2.15.2.7	PATHLINK_RESETBUF	清空用於存儲拍照時得到 MARK 信息和 latch 數據的 buffer	×	×
2.15.2.8	PATHLINK_STATUS	獲取當前 PATHLINK 的運動狀態	×	×
2.15.2.9	PATHLINK_RDYPOINT	計算 XYTable 跟隨之前的等待位置	×	×

2.15.2.1 PATHLINK_INI

語　法：PATHLINK_INI AX(MasAxisNo), SYNINFO_SartVR [, MasOffsetPos] [, MARK1_SartVR] [,MAangle] [, PAngle]

描　述：設置 PATH LINK 初始信息。包含同步位置資料, 圖像 MARK 點位置, 各坐標系對應關係等。這些資訊需在示教階段獲取，在程式啟動配置階段進行此部分的配置。需搭配 BASE 使用，例如 BASE 0,1 則有軸 0 和軸 1 建立了 XYTable。

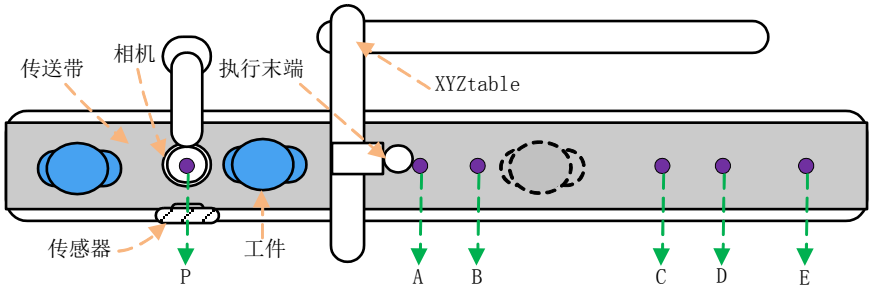


圖 2.15.2 機台示意圖

說明：

A： 影像處理完畢後，啟動同步關係

B：XYTable 開始塗膠

C：XYTable 塗膠完成

D：XYTable 停止

E：XYTable 回退等待位置

參　數：AX(MasAxisNo) 主軸號。

SYNINFO_SartVR: 同步關係曲線資訊表對應的 VR 的起始 index，即 VR[SYNINFO_SartVR] ~ VR[SYNINFO_SartVR+3]為同步關係曲線設定資訊，單位：PPU。

VR[SYNINFO_SartVR]: XYTable 從等待到開始加工(即進入同步)的這個過程主軸的移動距離，對應圖 2.15.1 傳送帶上 A 到 B 的距離；

VR[SYNINFO_SartVR+1]: XYTable 從等待到加工完成主軸的移動距離，對應圖 2.15.1 傳送帶上 A 到 C 的距離；

VR[SYNINFO_SartVR+2]: XYTable 從等待到減速停止的過程中主軸的移動距離，對應圖 2.15.1 傳送帶上 A 到 D 的距離；

VR[SYNINFO_SartVR+3]: XYTable 從等待經過加工完成並返回到等待位置的過程中主軸的移動距離，對應圖 2.15.1 傳送帶上 A 到 E 的距離；

MasOffsetPos: XY Table 開始進行同步時相對與 MARK 點的距離。缺省為 0.

對應圖上 A 點位置相對與標定時 Mark 點的距離（P 點），單位 PPU。

MARK1_SartVR: 標定時 MARK 點的示教位置資訊所在 VR 起始位置。

即 VR[MARK1_SartVR]~ VR[MARK1_SartVR+2]。缺省則示教位置資訊都為 0.由相機拍照獲得。

VR[MARK1_SartVR]： 示教時 MARK 點所在世界坐標系(WCS)中的 X 位置。

VR[MARK1_SartVR+1]：示教時 MARK 點所在世界坐標系（WCS）中的 Y 位置。

VR[MARK1_SartVR+2]： 示教時工件偏轉(相對於標定)弧度。順時針為正，逆時針為負。

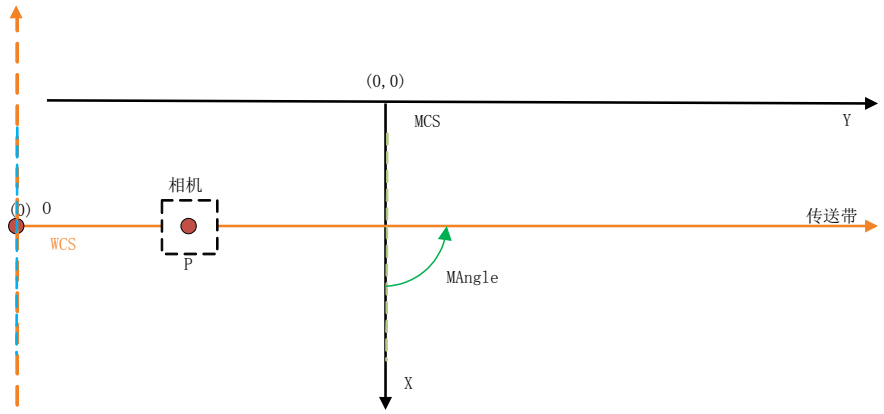


圖 2.15.3 坐標系示意圖

MAangle: WCS(世界坐標系)與 MCS(機械坐標系)之間的夾角弧度，單位為 rad。缺省為 0. 數值為由 MCS 的 X 軸正方向逆時針旋轉到 WCS 的 X 軸正方向。逆時針為正。如圖 5.15.2 所示，為 pi/2；

PAngle：PCS(工件坐標系)與 MCS(機械坐標系)之間的夾角弧度，單位為 rad。缺省為 0.（注：只有加工 CAD 導入軌跡時，才使用此值）

例 程

```
Dim As DOUBLE point_x, point_y
BASE 0,1 '軸 0,1 組成 XY Table
GVH= 20000
GACC = 100000
GDEC = 100000
'軸 2 為主軸,VR(0)~VR(3)存放同步資料{2000,20000,30000,40000}
'主軸運行到距離相機 Mark 點 1000 位置，XYTable 開始動作。
'VR(6)~VR(8)存放相機 Mark 點信息
'世界坐標系與機械坐標系夾角弧度值為 0.7853982.
PATHLINK_INI AX(2),0,1000,6,0.7853982,0
```

```
PATHLINK_RDYPOINT(point_x, point_y, 0)   '獲取 XYTable 的等待位置,讓 XYTable 移動到等待位置
MOVE point_x, point_y
WAIT DONE
PATHLINK_RESETBUF AX(2) '清空存放 latch 數據和 mark 信息的 buffer
PATHLINK_BEGIN AX(2) '設定加工軌跡
LINE 0,0
LINE 0,10000
LINE 10000,0
LINE 0,-10000
LINE -10000,0
PATHLINK_END
VR(20) = PATHLINK_STATUS
IF(VR(20)<1) then
    print "Pathlink failed."
END IF
```

2.15.2.2 PATHLINK_BEGIN

語　法：PATHLINK_BEGIN AX(MasAxisNo)

描　述：Pathlink 插補軌跡路徑起始符。需結合 PATHLINK_END，設定加工軌跡，所有軌跡皆為相對位置，第一段需為加工件的起點相對於 MARK 點的距離，第二段為加工件的第二點相對於第一點的距離，依次類推。軌跡通常由示教所得，有兩種示教方法：①機台示教；②圖像示教。

參　數：AX(MasAxisNo) 主軸號。

例　程：參考 PATHLINK_INI 例程

2.15.2.3 PATHLINK_END

語　法：PATHLINK_END AX(MasAxisNo)

描　述：Pathlink 插補軌跡路徑結束符。結合 PATHLINK_BEGIN 設定加工軌跡，執行完此命令，XYTable 和主軸便建立了跟隨關係。

參　數：AX(MasAxisNo) 主軸號。

例　程：參考 PATHLINK_INI 例程

2.15.2.4 PATHLINK_SETBUF

語　法：PATHLINK_SETBUF AX(MasAxisNo), MARK2_StartVR,LatchData

描　述：設定加工時的每次相機拍照時得到 MARK 的位置和角度資訊以及鎖存位置(理論位置/編碼器位置)寫入 Buffer 中，每次開始加工時，會從 buffer 中讀取 mark 資訊和鎖存位置，buffer 的空閒位置+1，buffer 總大小為 50。

參　數：AX(MasAxisNo) 主軸號。

MARK2_StartVR：每次加工前由相機得到的 MARK 點的新位置和偏轉角度資訊所在 VR 的起始 index。如果不設定，則 VR[MARK2_SartVR]~ VR[MARK2_SartVR+2]都為 0.

VR[MARK2_SartVR]：加工時 MARK 點所在 MCS 中的 X 位置，單位：PPU。

VR[MARK2_SartVR+1]：加工時 MARK 點所在 MCS 中的 Y 位置，單位：PPU

VR[MARK2_SartVR+2]：加工時工件偏轉弧度,相對於標定時的角度，單位：rad。

LatchData: 拍照時鎖存到的傳送帶的位置(理論位置/實際位置)，單位：PPU

注　意：

1. 需搭配 BASE X,X 使用。用來指定 XYTable，在啟用此指令之前，必須先設定 PATHLINK_INI 以及 PATHLINK_BEGIN，PATHLINK_END。
2. 調用此指令之前，需先通過調用 PATHLINK_BUFSTA 判斷 buffer 是否滿，如果滿則無法寫入 buffer。

例　程：

```
VR(15) = PATHLINK_BUFSTATUS(AX(2))
IF VR(15)<1 THEN
BASE 0,1
PATHLINK_SETBUF AX(2),10,0
END IF
```

2.15.2.5 PATHLINK_STOP

語　法 **1**：PATHLINK_STOP

語　法 **2**：PATHLINK_STOP AX(MasAxisNo)

描　述：解除主軸和從軸的同步關係，且插補運動都停止，主軸不停止。

參　數：AX(MasAxisNo) 主軸號。

例　程：參考 PATHLINK_INI 例程

2.15.2.6 PATHLINK_BUFSTATUS

語　法 **1**：PATHLINK_BUFSTATUS

語　法 2：PATHLINK_BUFSTATUS AX(MasAxisNo)

描　述：用於獲取加工時用於存儲拍照時得到MARK信息和latch數據的buffer的狀態。

當 Buffer 滿了之後，需等待有空閒之後再寫入，Buffer 總大小為 50。

參　數：AX(MasAxisNo) 主軸號。

返回值： 0—未滿，尚有空間，1—已滿，不可再寫入。

例　程：參考 PATHLINK_SETBUF 例程。

2.15.2.7 PATHLINK_RESETBUF

語　法 1：PATHLINK_RESETBUF

語　法 2：PATHLINK_RESETBUF AX(MasAxisNo)

描　述：清空用於存儲拍照時得到 MARK 信息和 latch 數據的 buffer。

參　數：AX(MasAxisNo) 主軸號。

例　程：參考 PATHLINK_INI 例程。

2.15.2.8 PATHLINK_STATUS

語　法 1：PATHLINK_STATUS

語　法 2：PATHLINK_STATUS AX(MasAxisNo)

描　述：獲取當前 PATHLINK 的運動狀態，方便與其他工藝配合運動。

參　數：AX(MasAxisNo) 主軸號。

返回值：

0	未進入 pathlink
1	等待工件，pathlink 已啟動，但是相機未識別到未加工的工件
2	等待加工，相機識別到工件，但是還沒有到達加工位置
3	加速區，加速到與傳送帶同步
4	同步區，加工過程中
5	減速區，加工完成，XYTable 減速過程中
6	回程區，XYTable 回到等待位置
7	異常，運動過程中異常停止。已退出同步區，但 path 還沒有走完即執行末端到達 C 的時，path 還沒有走完，則報警。

例　程：參考 PATHLINK_INI 例程。

2.15.2.9 PATHLINK_RDYPOINT

語　法：PATHLINK_RDYPOINT point_x, point_y[, mark_offset]

描　述：在 PATHLINK_INI 之後，啟動 pathlink 之前，用此指令可獲取等待位置，用於將執行末端移動指定位置，即等待位置。

參　數：point_x　返回得到的等待位置的 X 座標值，單位 PPU

Point_y　返回得到的等待位置的 Y 座標值，單位 PPU

mark_offset　輸入相機標定位置的偏移量，單位 PPU

例　程：參考 PATHLINK_INI 例程

2.16 全域變數 VR 操作

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.16.1	VR	實數型全域變數	√	√
2.16.2	FILE_WRITEVR	寫 VR 檔到本地	×	×
2.16.3	FILE_READVR	將本地 VR 檔載入專案	×	×
2.16.4	VRCopy	將一段 VR 區域的資料拷貝到另一段 VR 區域	×	×
2.16.5	VRExchange	將一段 VR 區域的資料與另一段長度相等 VR 區域的資料做交換	×	×
2.16.6	VRClear	將一段 VR 區域的數值清零	×	×
2.16.7	StrToVR	將字串塞入一段 VR 區域	×	×
2.16.8	VRToStr	將一段 VR 區域轉成字串	×	×

2.16.1 VR

語　法：VR(no)=value

類　型：Double

描　述：VR 變數是實數型全域變數。軟體平臺共提供 10000 個 VR 變數給使用者操作：VR(0)~VR(9999)。當 VR 變數用於 Modbus 通訊的自訂變數時，可以選擇將 VR 變數對應一個 16 位資料類型寄存器或 32 位資料類型寄存器。

參　數：no　VR 變數的索引號；範圍：0~9999，共 10000 個

例　程

```
Dim A As ULONG
A=15
VR(A)=200.525        '將 200.525 賦值給 VR(15)
BASE 0
VR(25)=1000
VL=VR(25)            '將 VR(15)的數值賦給軸 0 的初速度。
```

2.16.2 FILE_WRITEVR

語　法：FILE_WRITEVR　file_name，vr_start no，vr_end no

描　述：將一段 VR 的資料保存到本地文本，目前僅支援 bas 和 csv 兩種檔案類型。

參　數：file_name　　保存到本地文本的檔案名

vr_start no　VR 起始編號

vr_end no　　VR 結束編號

例　程

```
'將 VR(0)~VR(20)的數據寫到本地名為 VR_data 的 bas 文件
FILE_WRITEVR "VR_data.bas",0,20
'將 VR(0)~VR(100)的數據寫到本地名為 P_data 的 csv 文件
FILE_WRITEVR "P_data.csv",0,100
```

2.16.3 FILE_READVR

語　法：FILE_READVR　file_name

描　述：將本地 VR 文本的資料寫到當前工程的 VR 變數中

參　數：file_name　　保存到本地文本的檔案名

例　程

```
'將本地名為 VR_data 的 bas 文件 VR 數據讀到控制器裡，該 bas 文件裡的 VR 資料將覆蓋控制器裡對應的 VR 數據
FILE_READVR "VR_data.bas"
'將本地名為 P_data 的 csv 文件 VR 數據讀到控制器裡，該 csv 文件裡的 VR 資料將覆蓋控制器裡對應的 VR 數據
FILE_READVR "P_data.csv"
```

2.16.4 VRCopy

語　法：VRCopy　src_vr_start , dst_vr_start,count

描　述：將一段 VR 區域的資料拷貝到另一段 VR 區域

參　數：src_vr_start　　VR 源區域起始編號；類型：ULONG

dst_vr_start　　VR 目的地區域起始編號；類型：ULONG

count　　拷貝 VR 的個數；類型：ULONG

例　程

```
'將 VR(17), VR(18) , VR(19) , VR(20)的資料拷貝到 VR(30), VR(31) , VR(32) , VR(33)
VR(17)=1
```

```
VR(18)=2.2
VR(19)=3
VR(20)=4.5
VRCopy(17,30,4)    '拷貝 VR(17)~VR(20)的數據到 VR(30)~VR(33)
Print VR(30)       'VR(30)的數值為 1
Print VR(31)       'VR(31)的數值為 2.2
Print VR(32)       'VR(32)的數值為 3
Print VR(33)       'VR(33)的數值為 4.5
```

2.16.5 VRExchange

語　法：VRExchange　src_vr_start , dst_vr_start,count

描　述：將一段 VR 區域的資料與另一段長度相等 VR 區域的資料做交換

參　數：src_vr_start　　VR 源區域起始編號；類型：ULONG

dst_vr_start　　VR 目的地區域起始編號；類型：ULONG

count　　拷貝 VR 的個數；類型：ULONG

例　程

```
'將 VR(17), VR(18) , VR(19) , VR(20)的數值與 VR(30), VR(31) , VR(32) , VR(33)的資料做交換
VR(17)=1
VR(18)=2
VR(19)=3
VR(20)=4
VR(30)=6
VR(31)=7
VR(32)=8
VR(33)=9
VRExchange(17,30,4)    '將 VR(17)~VR(20)的數據與 VR(30)~VR(33)的資料做交換
Print VR(17)       'VR(17)的數值為 6
Print VR(18)       'VR(18)的數值為 7
Print VR(19)       'VR(19)的數值為 8
Print VR(20)       'VR(20)的數值為 9
Print VR(30)       'VR(30)的數值為 1
```

```
Print VR(31)        'VR(31)的數值為 2
Print VR(32)        'VR(32)的數值為 3
Print VR(33)        'VR(33)的數值為 4
```

2.16.6 VRClear

語　法：VRClear　vr_start , count

描　述：將一段 VR 區域的數值清零

參　數：vr_start　　VR 區域起始編號；類型：ULONG

　　count　　拷貝 VR 的個數；類型：ULONG

例　程

```
'將 VR(17), VR(18) , VR(19) , VR(20)的數值清零
VRClear(17,4)
Print VR(17)        'VR(17)的數值為 0
Print VR(18)        'VR(18)的數值為 0
Print VR(19)        'VR(19)的數值為 0
Print VR(20)        'VR(20)的數值為 0
```

2.16.7 StrToVR

語　法：StrToVR　StrInput, vr_start

描　述：將字串依次拆分成單個字元，以 ASCII 碼值塞入以 vr_start 為起始的 VR 區域，一個 VR 變數對應一個字元。

參　數：StrInput　　需轉換的字串；類型：String

　　vr_start　　存放轉換後字元的 VR 區域起始編號；類型　：ULONG

例　程

```
'將字串"mas"拆分成字元塞入以 VR(0)起始的 VR 區域
StrToVR("mas",0)
Print VR(0)         'VR(0)的數值為 109，"m"對應的 ASCII 碼值為 109
Print VR(1)         'VR(1)的數值為 97，"a"對應的 ASCII 碼值為 97
Print VR(2)         'VR(2)的數值為 115，"s"對應的 ASCII 碼值為 115
```

2.16.8 VRToStr

語　法：outstr=VRToStr　(vr_start, count)

描　述：將一段 VR 區域的每個 VR 值對應的字元組成字串返回。

參　數：vr_start　　VR 區域起始編號；類型：ULONG

　　count　　存放轉換後字元的 VR 區域起始編號

返回值：組成的字串；類型：String

例　程

```
Dim A as string
StrToVR("mas",0)    '將“mas”字串塞入 VR(0)~VR(2)
Print VR(0)         'VR(0)的數值為 109，"m"對應的 ASCII 碼值為 109
Print VR(1)         'VR(1)的數值為 97，"a"對應的 ASCII 碼值為 97
Print VR(2)         'VR(2)的數值為 115，"s"對應的 ASCII 碼值為 115
A=VRToStr(0,3)      '將 VR(0)~VR(2)值對應的字元組成字串返回給 A 變數
Print A             'A 列印出來為“mas”
```

2.17 檔操作

本節指令概覽

章節	指令	說明	終端工具	觀察變數工具
2.17.1	GetFilesCnt	讀取目的檔案夾下指定檔案類型的檔個數	×	×
2.17.2	GetFileName	讀取目的檔案夾下指定索引的檔案名	×	×
2.17.3	File_delete	刪除預設路徑或指定路徑下的檔	×	×
2.17.4	File_Find	讀取指定檔案名在資料夾中的索引	×	×
2.17.5	File_Rename	重命名指定資料夾下的指定檔案名	×	×
2.17.6	File_Copy	拷貝指定資料夾下的指定檔，並命名新拷貝出來的檔	×	×

2.17.1 GetFilesCnt

語　法：value=GetFilesCnt　([extension] [, isFullName][,folder_path])

描　述：讀取目的檔案夾下指定檔案類型的檔個數。

參　數：extension　　文件副檔名；類型：String

isFullName　　決定 GetFileName 指令取得的檔案名是否要包含副檔名；

0：不包含副檔名

1：包含副檔名

folder_path　　目的檔案夾的路徑；類型：String

返回值：指定檔案類型的個數；類型：ULONG

注　意：資料夾路徑不允許有中文。資料夾路徑可以寫絕對路徑，也可以寫相對路徑。相對路徑的預設路徑為：C:\Advantech\Motion_Runtime\AMI\AMI_User_Files

例　程

```
DIM A AS ULONG
A=GetFilesCnt( )     '讀預設路徑 C:\Advantech\Motion_Runtime\AMI\AMI_User_Files 下的文件個數
A=GetFilesCnt(".txt") '讀預設路徑下的 txt 檔案類型的文件個數
```

A= GetFilesCnt(".txt",0,"/abc") '讀預設路徑下名為 abc 資料夾下的 txt 檔案類型的文件個數

A= GetFilesCnt(".txt",0,"C:\Users\Administrator\Desktop\FILE_TEST") '用絕對路徑讀檔，讀桌面 FILE_TEST 資料夾下的 txt 檔案類型的文件個數。

2.17.2 GetFileName

語　法：value =GetFileName　(index)

描　述：讀取目的檔案夾下指定索引的檔案名。

參　數：index　檔索引；類型：ULONG

返回值：指定檔的檔案名；類型：String

注　意：該指令讀出來的檔案名字串是否包含副檔名是由 GetFilesCnt 中的參數 isFullName 決定的；該指令索引的資料夾路徑是由 GetFilesCnt 中的參數 folder_path 確定的；該指令參數 index 的編號是 windows 系統自動分配的，所以一般會把指定類型的檔全部讀出再針對檔案名做操作。綜上所述，該指令需配合 GetFilesCnt 指令使用。

例　程

```
'如預設路徑下有 3 個類型為 txt 的檔，名稱分別為 a1, a2, a3
DIM A AS ULONG
DIM txt_filename(0 to 2) AS STRING
DIM i AS INTEGER
A=GetFilesCnt(".txt") '讀預設路徑下的 txt 檔案類型的文件個數
FOR i=0 to 2
    txt_filename(i)=GetFileName(i)
    Print txt_filename(i)      'For 迴圈列印出來 3 個檔案名為 a1，a2，a3
NEXT i
```

2.17.3 File_delete

語　法：File_delete (file_path)

描　述：刪除預設路徑或指定路徑下的檔。

參　數：file_path　檔路徑；類型：String

例　程

```
FILE_DELETE("a1.txt") '刪除預設路徑下的“a1.txt”文件
```

```
FILE_DELETE("C:\Users\Administrator\Desktop\FILE_TEST\a1.txt") '刪除指定路徑下的“a1.txt”文件
```

2.17.4 File_Find

語　法：value =File_Find　(filename_list( ),filename)

描　述：讀取指定檔案名在資料夾中的索引

參　數：filename_list( )　目的檔案夾中檔案名列表；類型：String

filename　要讀取檔的檔案名；類型：String

返回值：讀取檔案名在資料夾中的索引；類型：ULONG

例　程

```
'如預設路徑下有 3 個類型為 txt 的檔，名稱分別為 a1, a2, a3
DIM A AS ULONG
DIM txt_filename(0 to 2) AS STRING
DIM i AS INTEGER
A=GetFilesCnt(".txt") '讀預設路徑下的 txt 檔案類型的文件個數
FOR i=0 to 2
    txt_filename(i)=GetFileName(i)
    Print txt_filename(i)      'For 迴圈列印出來 3 個檔案名為 a1，a2，a3
NEXT i
A=File_find(txt_filename( ),"a3")
Print A                  '列印出 A 的值為 2，代表 a3 這個檔在 txt_filename( )陣列中的索引為 2
```

2.17.5 File_Rename

語　法：File_Rename　src_filename,dst_filename

描　述：重命名指定資料夾下的指定檔案名

參　數：src_filename　需重命名的檔的檔案名；類型：String

dst_filename　新的檔案名；類型：String

例　程

```
'如預設路徑下有 3 個類型為 txt 的檔，名稱分別為 a1, a2, a3
DIM A AS ULONG
```

```
A=GetFilesCnt(".txt") '讀預設路徑下的 txt 檔案類型的文件個數
File_Rename "a3.txt","b3.txt"'預設路徑下的 a3.txt 檔重命名為 b3.txt
```

2.17.6 File_Copy

語　法：File_Copy　src_filename,dst_filename

描　述：拷貝指定資料夾下的指定檔，並命名新拷貝出來的檔

參　數：src_filename　需被拷貝檔的檔案名；類型：String

dst_filename　新拷貝出的檔案名；類型：String

例　程

```
'如預設路徑下有 3 個類型為 txt 的檔，名稱分別為 a1, a2, a3
DIM A AS ULONG
A=GetFilesCnt(".txt") '讀預設路徑下的 txt 檔案類型的文件個數
File_Copy "a3.txt","b3.txt"'預設路徑下的 a3.txt 檔拷貝一份，並將拷貝出來的檔命名為 b3.txt
```

第 3 章　附錄

3.1　錯誤代碼說明

3.1.1 RUN_ERROR 錯誤代碼表

錯誤代碼	說明
0x00000000	SUCCESS
0x80000000	InvalidDevNumber
0x80000001	DevRegDataLost
0x80000002	LoadDllFailed
0x80000003	GetProcAddrFailed
0x80000004	MemAllocateFailed
0x80000005	InvalidHandle
0x80000006	CreateFileFailed
0x80000007	OpenEventFailed
0x80000008	EventTimeOut
0x80000009	InvalidInputParam
0x8000000a	PropertyIDNotSupport
0x8000000b	PropertyIDReadOnly
0x8000000c	ConnectWinIrqFailed
0x8000000d	InvalidAxCfgVel
0x8000000e	InvalidAxCfgAcc
0x8000000f	InvalidAxCfgDec
0x80000010	InvalidAxCfgJerk
0x80000011	InvalidAxParVelLow
0x80000012	InvalidAxParVelHigh
0x80000013	InvalidAxParAcc
0x80000014	InvalidAxParDec
0x80000015	InvalidAxParJerk
0x80000016	InvalidAxPulseInMode

0x80000017	InvalidAxPulseOutMode
0x80000018	InvalidAxAlarmEn
0x80000019	InvalidAxAlarmLogic
0x8000001a	InvalidAxInPEn
0x8000001b	InvalidAxInPLogic
0x8000001c	InvalidAxHLmtEn
0x8000001d	InvalidAxHLmtLogic
0x8000001e	InvalidAxHLmtReact
0x8000001f	InvalidAxSLmtPEn
0x80000020	InvalidAxSLmtPReact
0x80000021	InvalidAxSLmtPValue
0x80000022	InvalidAxSLmtMEn
0x80000023	InvalidAxSLmtMReact
0x80000024	InvalidAxSLmtMValue
0x80000025	InvalidAxOrgLogic
0x80000026	InvalidAxOrgEnable
0x80000027	InvalidAxEzLogic
0x80000028	InvalidAxEzEnable
0x80000029	InvalidAxEzCount
0x8000002a	InvalidAxState
0x8000002b	InvalidAxInEnable
0x8000002c	InvalidAxSvOnOff
0x8000002d	InvalidAxDistance
0x8000002e	InvalidAxPosition
0x8000002f	InvalidAxHomeModeKw
0x80000030	InvalidAxCntInGp
0x80000031	AxInGpNotFound
0x80000032	AxIsInOtherGp
0x80000033	AxCannotIntoGp

0x80000034	GpInDevNotFound
0x80000035	InvalidGpCfgVel
0x80000036	InvalidGpCfgAcc
0x80000037	InvalidGpCfgDec
0x80000038	InvalidGpCfgJerk
0x80000039	InvalidGpParVelLow
0x8000003a	InvalidGpParVelHigh
0x8000003b	InvalidGpParAcc
0x8000003c	InvalidGpParDec
0x8000003d	InvalidGpParJerk
0x8000003e	JerkNotSupport
0x8000003f	ThreeAxNotSupport
0x80000040	DevIpoNotFinished
0x80000041	InvalidGpState
0x80000042	OpenFileFailed
0x80000043	InvalidPathCnt
0x80000044	InvalidPathHandle
0x80000045	InvalidPath
0x80000046	IoctlError
0x80000047	AmnetRingUsed
0x80000048	DeviceNotOpened
0x80000049	InvalidRing
0x8000004a	InvalidSlaveIP
0x8000004b	InvalidParameter
0x8000004c	InvalidGpCenterPosition
0x8000004d	InvalidGpEndPosition
0x8000004e	InvalidAddress
0x8000004f	DeviceDisconnect
0x80000050	DataOutBufExceeded

0x80000051	SlaveDeviceNotMatch
0x80000052	SlaveDeviceError
0x80000053	SlaveDeviceUnknow
0x80000054	FunctionNotSupport
0x80000055	InvalidPhysicalAxis
0x80000056	InvalidVelocity
0x80000057	InvalidAxPulseInLogic
0x80000058	InvalidAxPulseInSource
0x80000059	InvalidAxErcLogic
0x8000005a	InvalidAxErcOnTime
0x8000005b	InvalidAxErcOffTime
0x8000005c	InvalidAxErcEnableMode
0x8000005d	InvalidAxSdEnable
0x8000005e	InvalidAxSdLogic
0x8000005f	InvalidAxSdReact
0x80000060	InvalidAxSdLatch
0x80000061	InvalidAxHomeResetEnable
0x80000062	InvalidAxBacklashEnable
0x80000063	InvalidAxBacklashPulses
0x80000064	InvalidAxVibrationEnable
0x80000065	InvalidAxVibrationRevTime
0x80000066	InvalidAxVibrationFwdTime
0x80000067	InvalidAxAlarmReact
0x80000068	InvalidAxLatchLogic
0x80000069	InvalidFwMemoryMode
0x8000006a	InvalidConfigFile
0x8000006b	InvalidAxEnEvtArraySize
0x8000006c	InvalidAxEnEvtArray
0x8000006d	InvalidGpEnEvtArraySize

0x8000006e	InvalidGpEnEvtArray
0x8000006f	InvalidIntervalData
0x80000070	InvalidEndPosition
0x80000071	InvalidAxisSelect
0x80000072	InvalidTableSize
0x80000073	InvalidGpHandle
0x80000074	InvalidCmpSource
0x80000075	InvalidCmpMethod
0x80000076	InvalidCmpPulseMode
0x80000077	InvalidCmpPulseLogic
0x80000078	InvalidCmpPulseWidth
0x80000079	InvalidPathFunctionID
0x8000007a	SysBufAllocateFailed
0x8000007b	SpeedFordFunNotSpported
0x8000007c	InvalidNormVector
0x8000007d	InvalidCmpTimeTableCount
0x8000007e	InvalidCmpTime
0x8000007f	FWDownLoading
0x80000080	FWVersionNotMatch
0x80000081	InvalidAxParHomeVelLow
0x80000082	InvalidAxParHomeVelHigh
0x80000083	InvalidAxParHomeAcc
0x80000084	InvalidAxParHomeDec
0x80000085	InvalidAxParHomeJerk
0x80000086	InvalidAxCfgJogVelLow
0x80000087	InvalidAxCfgJogVelHigh
0x80000088	InvalidAxCfgJogAcc
0x80000089	InvalidAxCfgJogDec
0x8000008a	InvalidAxCfgJogJerk

0x8000008b	InvalidAxCfgKillDec
0x8000008c	NotOpenAllAxes
0x8000008d	NotSetServoComPort
0x8000008e	OpenComPortFailed
0x8000008f	ReadComPortTimeOut
0x80000090	SetComPortStateFailed
0x80000091	SevroTypeNotSupport
0x80000092	ReadComBufFailed
0x80000096	SlaveIOUpdateError
0x80000097	NoSlaveDevFound
0x80000098	MasterDevNotOpen
0x80000099	MasterRingNotOpen
0x800000c8	InvalidDIPort
0x800000c9	InvalidDOPort
0x800000ca	InvalidDOValue
0x800000cb	CreateEventFailed
0x800000cc	CreateThreadFailed
0x800000cd	InvalidHomeModeEx
0x800000ce	InvalidDirMode
0x800000cf	AxHomeMotionFailed
0x800000d0	ReadFileFailed
0x800000d1	PathBufIsFull
0x800000d2	PathBufIsEmpty
0x800000d3	GetAuthorityFailed
0x800000d4	GpIDAllocatedFailed
0x800000d5	FirmWareDown
0x800000d6	InvalidGpRadius
0x800000d7	InvalidAxCmd
0x800000d8	InvalidaxExtDrv

0x800000d9	InvalidGpMovCmd
0x800000da	SpeedCurveNotSupported
0x800000db	InvalidCounterNo
0x800000dc	InvalidPathMoveMode
0x800000dd	PathSelStartCantRunInSpeedForwareMode
0x800000de	InvalidCamTableID
0x800000df	InvalidCamPointRange
0x800000e0	CamTableIsEmpty
0x800000e1	InvalidPlaneVector
0x800000e2	MasAxIDSameSlvAxID
0x800000e3	InvalidGpRefPlane
0x800000e4	InvalidAxModuleRange
0x800000e5	DownloadFileFailed
0x800000e6	InvalidFileLength
0x800000e7	InvalidCmpCnt
0x800000e8	JerkExceededMaxValue
0x800000e9	AbsMotionNotSupport
0x800000ea	invalidAiRange
0x800000eb	AIScaleFailed
0x800000ec	AxInRobot
0x800000ed	Invalid3DarcFlat
0x800000ee	InvalidIpoMap
0x800000ef	DataSizeNotCorrect
0x800000f0	AxisNotFound
0x800000f1	InvalidPathVelHigh
0x80002000	HlmtPExceeded
0x80002001	HlmtNExceeded
0x80002002	SlmtPExceeded

0x80002003	SlmtNExceeded
0x80002004	AlarmHappened
0x80002005	EmgHappened
0x80002006	TimeLmtExceeded
0x80002007	DistLmtExceeded
0x80002008	InvalidPositionOverride
0x80002009	OperationErrorHappened
0x8000200a	SimultaneousStopHappened
0x8000200b	OverflowInPAPB
0x8000200c	OverflowInIPO
0x8000200d	STPHappened
0x8000200e	SDHappened
0x8000200f	AxisNoCmpDataLeft
0x10000001	Warning_AxWasInGp
0x10000002	Warning_GpInconsistRate
0x10000003	Warning_GpInconsistPPU
0x10000004	Warning_GpMoveDistanceCanntBeZero
0x80004001	DevEvtTimeOut
0x80004002	DevNoEvt
0x80005001	ERR_SYS_TIME_OUT
0x80005002	Dsp_PropertyIDNotSupport
0x80005003	Dsp_PropertyIDReadOnly
0x80005004	Dsp_InvalidParameter
0x80005005	Dsp_DataOutBufExceeded
0x80005006	Dsp_FunctionNotSupport
0x80005007	Dsp_InvalidConfigFile

0x80005008	Dsp_InvalidIntervalData
0x80005009	Dsp_InvalidTableSize
0x8000500a	Dsp_InvalidTableID
0x8000500b	Dsp_DataIndexExceedBufSize
0x8000500c	Dsp_InvalidCompareInterval
0x8000500d	Dsp_InvalidCompareRange
0x8000500e	Dsp_PropertyIDWriteOnly
0x8000500f	Dsp_NcError
0x80005010	Dsp_CamTableIsInUse
0x80005011	Dsp_EraseBlockFailed
0x80005012	Dsp_ProgramFlashFailed
0x80005013	Dsp_WatchdogError
0x80005014	Dsp_ReadPrivateOverMaxTimes
0x80005015	Dsp_InvalidPrivateID
0x80005016	Dsp_DataNotReady
0x80005017	Dsp_LastOperationNotOver
0x80005018	Dsp_WritePrivateTimeout
0x80005019	Dsp_FwIsDownloading
0x80005020	Dsp_FwDownloadStepError
0x80005101	Dsp_InvalidAxCfgVel
0x80005102	Dsp_InvalidAxCfgAcc
0x80005103	Dsp_InvalidAxCfgDec
0x80005104	Dsp_InvalidAxCfgJerk
0x80005105	Dsp_InvalidAxParVelLow
0x80005106	Dsp_InvalidAxParVelHigh
0x80005107	Dsp_InvalidAxParAcc
0x80005108	Dsp_InvalidAxParDec
0x80005109	Dsp_InvalidAxParJerk

0x8000510a	Dsp_InvalidAxPptValue
0x8000510b	Dsp_InvalidAxState
0x8000510c	Dsp_InvalidAxSvOnOff
0x8000510d	Dsp_InvalidAxDistance
0x8000510e	Dsp_InvalidAxPosition
0x8000510f	Dsp_InvalidAxHomeMode
0x80005110	Dsp_InvalidPhysicalAxis
0x80005111	Dsp_HlmtPExceeded
0x80005112	Dsp_HlmtNExceeded
0x80005113	Dsp_SlmtPExceeded
0x80005114	Dsp_SlmtNExceeded
0x80005115	Dsp_AlarmHappened
0x80005116	Dsp_EmgHappened
0x80005117	Dsp_CmdValidOnlyInConstSec
0x80005118	Dsp_InvalidAxCmd
0x80005119	Dsp_InvalidAxHomeDirMode
0x8000511a	Dsp_AxisMustBeModuloAxis
0x8000511b	Dsp_AxIdCantSameAsMasId
0x8000511c	Dsp_CantResetPosiOfMasAxis
0x8000511d	Dsp_InvalidAxExtDrvOperation
0x8000511e	Dsp_AxAccExceededMaxAcc
0x8000511f	Dsp_AxVelExceededMaxVel
0x80005120	Dsp_NotEnoughPulseForChgV
0x80005121	Dsp_NewVelMustGreaterThanVelLow
0x80005122	Dsp_InvalidAxGearMode
0x80005123	Dsp_InvalidGearRatio
0x80005124	Dsp_InvalidPWMDataCount
0x80005125	Dsp_InvalidAxPWMFreq
0x80005126	Dsp_InvalidAxPWMDuty

0x80005127	Dsp_AxGantryExceedMaxDiffValue
0x80005128	Dsp_ChanelIsDisable
0x80005129	Dsp_ChanelBufferIsFull
0x80005130	Dsp_ChanelBufferIsEmpty
0x80005131	Dsp_InvalidDoChanelID
0x80005132	Dsp_LatchHappened
0x80005201	Dsp_InvalidAxCntInGp
0x80005202	Dsp_AxInGpNotFound
0x80005203	Dsp_AxIsInOtherGp
0x80005204	Dsp_AxCannotIntoGp
0x80005205	Dsp_GpInDevNotFound
0x80005206	Dsp_InvalidGpCfgVel
0x80005207	Dsp_InvalidGpCfgAcc
0x80005208	Dsp_InvalidGpCfgDec
0x80005209	Dsp_InvalidGpCfgJerk
0x8000520a	Dsp_InvalidGpParVelLow
0x8000520b	Dsp_InvalidGpParVelHigh
0x8000520c	Dsp_InvalidGpParAcc
0x8000520d	Dsp_InvalidGpParDec
0x8000520e	Dsp_InvalidGpParJerk
0x8000520f	Dsp_JerkNotSupport
0x80005210	Dsp_ThreeAxNotSupport
0x80005211	Dsp_DevIpoNotFinished
0x80005212	Dsp_InvalidGpState
0x80005213	Dsp_OpenFileFailed
0x80005214	Dsp_InvalidPathCnt
0x80005215	Dsp_InvalidPathHandle
0x80005216	Dsp_InvalidPath

0x80005217	Dsp_GpSlavePositionOverMaster
0x80005218	Dsp_GpPathBufferOverflow
0x80005219	Dsp_InvalidPathFunctionID
0x8000521a	Dsp_SysBufAllocateFailed
0x8000521b	Dsp_InvalidGpCenterPosition
0x8000521c	Dsp_InvalidGpEndPosition
0x8000521d	Dsp_InvalidGpCmd
0x8000521e	Dsp_AxHasBeenInInGp
0x8000521f	Dsp_ThreeAxNotSupport
0x80005220	Dsp_InvalidPathRange
0x80005221	Dsp_InvalidNormVector

3.1.2 SYSTEM_ERROR 錯誤代碼表

錯誤代碼	說明
0	SUCCESS
0x90000000	AMI_NULL_PROJECT_EXIST
0x90000001	AMI_INVALID_INPUT_PARAMS
0x90000002	AMI_INVALID_RETURN
0x90000003	AMI_INVALID_CTRL_MODE
0x90000004	AMI_CONTROLLER_LOCKED
0x90000005	AMI_GET_MAC_FAILED
0x90000006	AMI_INVALID_COMMAND
0x90000007	AMI_SET_MEM_FAILED
0x90000008	AMI_GET_VERSION_FAILED
0x9000000a	AMI_CTRL_ENCODED_ALREADY
0x9000000b	AMI_CTRL_INVALID_PASSWORD
0x9000000c	AMI_GET_VARIABLE_FAILED
0x9000000d	AMI_NUM_CONVERT_FAILED
0x90000032	AMI_ACTION_NOT_ALLOWED

0x90000064	AMI_SOCK_TIME_OUT
0x90000065	AMI_LOAD_FILE_FAILED
0x90000066	AMI_DOWN_FILE_FAILED
0x90000067	AMI_LOAD_PROJECT_FAILED
0x90000068	AMI_DOWN_PROJECT_FAILED
0x90000069	AMI_SOCK_ALREADY_CONNECTED
0x9000006A	AMI_SOCK_COMMU_FAILED
0x90000096	AMI_CONNECTION_FAILED
0x90000097	AMI_DISCONNECTION_FAILED
0x90000098	AMI_SEND_COMMAND_TIMEOUT
0x900000C8	AMI_OPEN_FILE_FAILED
0x900000C9	AMI_CREATE_FILE_FAILED
0x900000CA	AMI_REMOVE_FILE_FAILED
0x900000CB	AMI_PATH_NOT_EXIST
0x900000CC	AMI_SET_NON_BLOCK_FAILED
0x900000CD	AMI_SET_BLOCK_FAILED
0x900000CE	AMI_CFG_FILE_NOT_EXISTS
0x900000CF	AMI_REF_FILE_NOT_EXISTS
0x900000D0	AMI_HEAD_FILE_NOT_EXISTS
0x900000D1	AMI_FILE_NOT_EXISTS
0x900000D2	AMI_FILE_INVALID_FORMAT
0x900000DC	AMI_PRJ_FILE_LOAD_FAILED
0x900000DD	AMI_SOURCE_FILE_NOT_EXISTS
0x900000DE	AMI_DST_FILE_EXISTS_ALREADY
0x900000FA	AMI_XML_LOAD_FAILED
0x900000FB	AMI_XML_CHECK_FAILED
0x900000FC	AMI_XML_SAVE_FAILED

0x900000FD	AMI_XML_ADD_FAILED
0x900000FE	AMI_XML_DELETE_FAILED
0x900000FF	AMI_XML_CREATE_FAILED
0x90000100	AMI_XML_INVALID_ELEMENT
0x9000012C	AMI_TASK_NOT_EXIST
0x9000012D	AMI_FORK_PROCESS_FAILED
0x9000012E	AMI_POPEN_FILE_FAILED
0x9000012F	AMI_STILL_RUNNING
0x90000130	AMI_NOT_IN_IDLE
0x90000131	AMI_GET_NO_ERROR
0x90000132	AMI_GET_NO_INFO
0x90000133	AMI_GET_MSG_NOTFINISHED
0x90000134	AMI_CREAT_PIPE_FAILED
0x90000136	AMI_RUN_FAILED
0x90000137	AMI_STOP_FAILED
0x90000138	AMI_NOT_RUNNING
0x90000140	AMI_DB_INIT_FAILED
0x90000141	AMI_DB_COMPILE_FAILED
0x90000142	AMI_DB_BREAKPOINT_FAILED
0x90000143	AMI_DB_CLEARPOINT_FAILED
0x90000144	AMI_DB_DELETEPOINTS_FAILED
0x90000145	AMI_DB_RUN_FAILED
0x90000146	AMI_DB_CONTINUE_FAILED
0x90000147	AMI_DB_NEXT_FAILED
0x90000148	AMI_DB_PROGRAM_NOT_RUN
0x90000149	AMI_DB_STOP_FAILED
0x9000014A	AMI_DB_OUT_OF_RANGE
0x9000014B	AMI_DB_EXIT_NOMARLLY

0x9000014C	AMI_DB_GET_LOCAL_VAR_FAILED
0x9000014D	AMI_DB_NOT_READY
0x9000014E	AMI_DB_ALREADY_PAUSED
0x9000014F	AMI_GET_RUNNING_TASKLIST_FAILED
0x90000190	AMI_MB_ILLEGAL_FUNCTION
0x90000191	AMI_MB_CRC_FAILED
0x90000192	AMI_MB_ILLEGAL_LENGTH
0x900001F4	AMI_MEM_UPDATE_VR_MBADDR_ERROR
0x900001F5	AMI_MEM_UPDATE_TABLE_MBADDR_ERROR
0x900001F6	AMI_MEM_UPDATE_DO_INIT_VALUE_ERROR
0x90000258	AMI_BASIC_RESET_ERROR
0x90000259	AMI_BASIC_INITIAL_ERROR
0x9000025A	AMI_BASIC_REFRESH_ERROR
0x9000025B	AMI_BASIC_GET_OFFSET_VALUE_FAILED
0xb0000000	AMI_GetSharedMemFailed
0xb0000001	AMI_GetTaskNameFailed
0xb0000002	AMI_IsNotInitialized
0xb0000003	AMI_IsAlreadyInitialized
0xb0000004	AMI_LoadXMLFailed
0xb0000005	AMI_ParseXMLFailed
0xb0000006	AMI_CreateDevListFailed
0xb0000007	AMI_InitializeDeviceFailed
0xb0000008	AMI_InitializeSharedMemFailed
0xb0000009	AMI_RefreshSharedMemFailed
0xb000000a	AMI_SetDeviceCfgFailed

0xb000000b	AMI_IncorrectCommand
0xb000000c	AMI_DeviceLargerList
0xb000000d	AMI_SerialPortError
0xb000000e	AMI_EthernetError
0xb000000f	AMI_LogOpenFailed
0xb0000010	AMI_StartTaskFailed
0xb0000011	AMI_StopTaskFailed
0xb0000012	AMI_CreatEventFailed
0xb0000013	AMI_CreatEThreadsFailed
0xb0000014	AMI_AllocatePMotionInfoFiled
0xb0000015	AMI_FAILEDTOCHECKEVENT
0xb0000016	AMI_FailedCloseCheckingEventThread
0xb0001000	AMI_CardsNotFound
0xb0001001	AMI_MotionBoardIDNotFound
0xb0001002	AMI_AxesOrGroupCountNotFound
0xb0001003	AMI_AxisIDorPhyIDNotFound
0xb0001004	AMI_GroupNotFound
0xb0001005	AMI_AxisInfoError
0xb0001006	AMI_MotionDeviceCountError
0xb0001007	AMI_InputBoardIDNotFound
0xb0001008	AMI_InputDeviceCountError
0xb0001009	AMI_InDAQdeviceCountError
0xb000100a	AMI_OutputBoardIDNotFound
0xb000100b	AMI_OutputDeviceCountError
0xb000100c	AMI_OutDAQdeviceCountError
0xb000100d	AMI_ActualDeviceCountError
0xb0002000	AMI_WrongAxisIndex

0xb0002001	AMI_WrongDoIndex
0xb0002002	AMI_WrongDiIndex
0xb0002003	AMI_NoAxis
0xb0002004	AMI_AxisInDifferentDevice
0xb0002005	AMI_WaitModeNotMatch
0xb0002006	AMI_MasterAxisIndexError
0xb0002007	AMI_GANTRYAxisNotInSameDev
0xb0002008	AMI_GEARAxisNotInSameDev
0xb0002009	AMI_AddPathAxisCntError
0xb000200a	AMI_AddPathHELIX3PnotSupport
0xb0003000	AMI_EthernetModeError
0xb0003001	AMI_EthernetOpened
0xb0003002	AMI_EthernetOpenFailed
0xb0003003	AMI_EthernetCloseFailed
0xb0003004	AMI_EthernetWrongNum
0xb0003005	AMI_EthernetNotOpen
0xb0003006	AMI_EthernetReadFailed
0xb0003007	AMI_EthernetResetFailed
0xb0003008	AMI_EthernetWriteFailed
0xb0003009	AMI_EthernetReadVRFailed
0xb000300a	AMI_EthernetWriteVRFailed
0xb0003100	AMI_SerialPortWrongID
0xb0003101	AMI_SerialPortOpenFailed
0xb0003102	AMI_SerialPortCloseFailed
0xb0003103	AMI_SerialPortNotOpen
0xb0003104	AMI_SerialPortWrongCfg
0xb0003105	AMI_SerialPortSetCfgFailed
0xb0003106	AMI_SerialPortWriteFailed

0xb0003107	AMI_SerialPortReadFailed
0xb0003108	AMI_SerialPortResetFailed
0xb0003109	AMI_SerialPortWriteVRFailed
0xb000310a	AMI_SerialPortReadVRFailed

3.2 MAS 控制器運動功能支持列表

項目	說明	PCI-1245L-MAS	PCI-1245-MAS	PCI-1285-MAS	MVP-3245/85-MAS
單軸運動	單軸點位運動	✓	✓	✓	✓
	單軸定速運動	✓	✓	✓	✓
	變位運動	✓	✓	✓	✓
	變速運動	✓	✓	✓	✓
	背隙補償	✓	✓	✓	✓
	T&S 形速度曲線	✓	✓	✓	✓
	運動中重設軸的加速度和減速度	✓	✓	✓	✓
	同步運動	不支持	✓	✓	✓
	疊加運動	不支持	不支持	不支持	不支持
	JOG 運動	✓	✓	✓	✓
	手輪運動	✓	✓	✓	✓
	回原點運動(16 種模式)	✓	✓	✓	✓
	DI 觸發停止	✓	✓	✓	✓
插補功能	直線插補	2 軸	2~3 軸	2~3 軸	2~3 軸
	直接線性插補	2 軸	2~4 軸	2~8 軸	2~4 軸
	2 軸圓弧插補	不支持	✓	✓	✓
	3 軸圓弧插補	不支持	不支持	不支持	不支持
	螺旋插補	不支持	✓	✓	✓
	運動中改變組的運行速度	不支持	✓	✓	✓
同步運動	電子齒輪	不支持	✓	✓	✓
	電子凸輪	不支持	✓	不支持	不支持
	龍門	不支持	✓	✓	✓
	CAM DO	不支持	✓	✓	✓
	切向跟隨	不支持	✓	✓	✓

	PathLink	不支持	✓	不支持	不支持
路徑運動	載入路徑功能	不支持	支援最多 10000 個點	支援最多 7000 個點	支援最多 10000 個點
	添加直線運動	不支持	✓	✓	✓
	添加圓弧運動	不支持	✓	✓	✓
	添加螺旋運動	不支持	✓	✓	✓
	延遲功能	不支持	✓	✓	✓
	添加 DO 開關運動	不支持	✓	✓	✓
	路徑速度交接功能	不支持	✓	✓	✓
	路徑速度前瞻功能	不支持	不支持	不支持	不支持
比較觸發功能	單點比較	不支持	✓	✓	✓
	等距線性比較	不支持	✓	✓	✓
	不等距隨機點比較	不支持	✓	✓	✓
	多軸比較觸發	不支持	不支持	不支持	✓
等待事件	單軸/插補運動的停止運行功能	✓	✓	✓	✓
	軸的比較	不支持	✓	✓	✓
	軸的鎖存	不支持	✓	✓	✓
	軸的鎖存緩存	不支持	✓	✓	✓
輸入/輸出功能	DI	16	16	32	32
	DO	16	16	32	32
	AI	不支持	不支持	不支持	不支持
	AO	不支持	不支持	不支持	不支持

3.3 資料類型及類型轉換指令

Motion Basic 常用資料類型說明

資料類型	說明
BOOLEAN	布林資料類型，True 或者 False
BYTE	長度為 8 位元的整數資料類型
UBYTE	長度為 8 位的不帶正負號的整數資料類型
DOUBLE	長度為 64 位的雙精度浮點資料類型
INTEGER	長度為 32 位或 64 位元的整數資料類型
UINTEGER	長度為 32 位或 64 位的不帶正負號的整數資料類型
LONG	長度為 32 位元的整數資料類型
ULONG	長度為 32 位的不帶正負號的整數資料類型
SHORT	長度為 16 位元的整數資料類型
USHORT	長度為 16 位的不帶正負號的整數資料類型
UNSIGNED	整數類型有無符號的修飾符
STRING	字串類型
SINGLE	長度為 32 位的單精確度浮點資料類型

資料類型轉換指令

指令	說明
CBYTE	將數位或字串的運算式轉換成位元組類型資料
CDBL	將數位或字串的運算式轉換成雙精度浮點數
CHR	返回用 ASCII 碼表達的值對應的字元
CINT	將數位或字串運算式轉換成整型資料
CLNG	將數位或字串運算式轉換成長整型資料
CLNGINT	將數位或字串運算式轉換成 64 位元長整型數據
CSNG	將數位或字串的運算式轉換成單精確度浮點數
CUBYTE	將數位或字串運算式轉換為無符號位元組類型資料
CUINT	將數位或字串運算式轉換為無符號整型資料
CULNG	將數位或字串運算式轉換為無符號長整型資料

CULNGINT	將數位或字串運算式轉換為無符號 64 位元長整型數據
CUNSG	將一個運算式轉換成對應的無符號類型
CUSHORT	將數位或字串運算式轉換為無符號短整型資料
VALINT	將一個字串轉換為一個 INTEGER 類型資料
VAL	將一個字串轉換為一個浮點數
HEX	將一個數用十六進位數返回
OCT	將一個數用八進位數返回
VALLNG	將一個字串轉換為一個 LONG 類型資料
VALUINT	將一個字串轉換為一個 UINTEGER 類型資料
VALULNG	將一個字串轉換為一個 ULONG 類型資料
ASC	返回字串中字元的 ASCII 碼

3.4 Modbus 地址

Motion Studio 中的變數內建有 Modbus 位址，用於與外部的通信。變數與 Modbus 位址的對應關係分五大區塊，如下表：

區塊	Modbus 組別	Modbus 地址範圍	讀/寫
標準變數—DOUT	COIL STATUS	1~1024	可讀/可寫
標準變數—DI	INPUT STATUS	10001~11024	唯讀
標準變數—狀態（運動控制相關）	INPUT REGISTER	30001~31600	唯讀
標準變數—參數、配置（運動控制相關）	HOLDING REGISTER	45001~49800	可讀/可寫
內建自訂變數—VR	HOLDING REGISTER	40001~44000	可讀/可寫

注：以下各區塊變數與 **Modbus** 對應關係中資料類型說明如下：

- **Bit**： 位
- **UINT16**: 16 位元無符號整型
- **INT16:** 16 位整型
- **UINT32**：32 位元無符號整型
- **INT 32**: 32 位整型
- **F32**：32 位浮點型

3.4.1 DOUT 對應的 Modbus 地址

變數	說明	Modbus 地址	資料類型
DOUT(0)	索引為 0 的數位量輸出	1	Bit
DOUT(1)	索引為 1 的數位量輸出	2	Bit
……			Bit
DOUT(1023)	索引為 1023 的數位量輸出	1024	Bit

3.4.2 DI 對應的 Modbus 地址

變數	說明	Modbus 地址	資料類型
DI(0)	索引為 0 的數位量輸入	10001	Bit
DI(1)	索引為 1 的數位量輸入	10002	Bit
.....			Bit
DI(1023)	索引為 1023 的數位量輸入	11024	Bit

3.4.3 軸狀態對應的 Modbus 地址

軸狀態	Modbus 地址範圍
軸 0	30001~30100
軸 1	30101~30200
.....	
軸 15	31501~31600

- 軸 0 狀態的 Modbus 地址表

變數	說明	Modbus 地址	資料類型
DPOS	累計指令位置（理論位置）	30001	F32
MPOS	累計回饋位置（實際位置）	30003	F32
STATE	當前軸運動狀態	30005	UINT16
M_STATUS	暫無作用，預留	30006	UINT32
DSPEED	當前軸運動指令速度（理論速度）	30008	F32
MIO	運動相關 I/O 的狀態	30010	UINT32
RUN_ERROR	軸錯誤資訊	30012	UINT32

- 軸 1 狀態的 Modbus 地址表

變數	說明	Modbus 地址	資料類型
DPOS	累計指令位置（理論位置）	30101	F32
MPOS	累計回饋位置（實際位置）	30103	F32
STATE	當前軸運動狀態	30105	UINT16
M_STATUS	暫無作用，預留	30106	UINT32
DSPEED	當前軸運動指令速度（理論速度）	30108	F32
MIO	運動相關 I/O 的狀態	30110	UINT32
RUN_ERROR	軸錯誤資訊	30112	UINT32

- .....

- 軸 15 狀態的 Modbus 地址表

變數	說明	Modbus 地址	資料類型
DPOS	累計指令位置（理論位置）	31501	F32
MPOS	累計回饋位置（實際位置）	31503	F32
STATE	當前軸運動狀態	31505	UINT16
M_STATUS	暫無作用，預留	31506	UINT32
DSPEED	當前軸運動指令速度（理論速度）	31508	F32
MIO	運動相關 I/O 的狀態	31510	UINT32
RUN_ERROR	軸錯誤資訊	31512	UINT32

3.4.4 軸參數、配置對應的 Modbus 地址

軸參數、配置	Modbus 地址範圍
軸 0	45001~45300
軸 1	45301~45600
.....	
軸 15	49501~49800

- 軸 0 參數、配置的 Modbus 地址表

變數	說明	Modbus 地址	資料類型
VL	軸初速度	45001	F32
VH	軸運行速度	45003	F32
ACC	軸加速度	45005	F32
DEC	軸減速度	45007	F32
JK	速度曲線類型：S 型或 T 型	45009	F32
MAXVEL	軸運行速度限值	45011	F32
MAXACC	軸加速度限值	45013	F32
MAXDEC	軸減速度限值	45015	F32
MAXJK	暫無作用，預留	45017	F32
INSTOP_DEC	INSTOP 功能中的減速度	45019	F32
UNIT_NUM	脈衝當量分子	45031	UINT32
UNIT_DENOM	脈衝當量分母	45033	UINT32
HOME_CROSS	回原點運行中的跨越距離	45051	F32
HOME_EX	暫無作用，預留	45053	UINT16
ORG_LOGIC	ORG 信號的有效邏輯電平	45054	UINT16
ORG_MODE	回原點運動結束時的停止模式	45055	UINT16
EZ_LOGIC	Z 相輸入信號的有效邏輯電平	45056	UINT16
HOME_RESET	回原點後清零位置值功能	45057	UINT16
HOME_OFFSETDIST	回原點成後的偏移距離	45058	F32
HOME_OFFSETVEL	回原點成後偏移距離的移動速	45060	F32

	度		
HOME_MODE	回原點模式	45062	UINT16
HOME_VL	回原點運動的初速度	45063	F32
HOME_VH	回原點運動的運行速度	45065	F32
HOME_ACC	回原點運動的加速度	45067	F32
HOME_DEC	回原點運動的減速度	45069	F32
HOME_JK	回原點運動的速度曲線類型	45071	F32
JOG_VLTIME	JOG 運動低段速度運行的時間	45081	UINT32
JOG_VL	JOG 運動低段速度	45083	F32
JOG_VH	JOG 運動高段速度	45085	F32
JOG_ACC	JOG 運動的加速度	45087	F32
JOG_DEC	JOG 運動的減速度	45089	F32
EL_EN	使能硬體限位元功能	45101	UINT16
EL_LOGIC	硬體限位元信號的有效邏輯電平	45102	UINT16
EL_MODE	硬限位元觸發時的停止模式	45103	UINT16
PEL_TOL_EN	正方向硬極限容差功能使能	45104	UINT16
PEL_TOL	正方向硬極限容差值	45105	UINT32
NEL_TOL_EN	負方向硬極限容差功能使能	45107	UINT16
NEL_TOL	負方向硬極限容差值	45108	UINT32
PIN_MODE	編碼器脈衝輸入類型	45121	UINT16
PIN_LOGIC	編碼器脈衝輸入邏輯反相	45122	UINT16
PIN_MAXFREQ	編碼器脈衝輸入的最高頻率限值	45123	UINT16
POUT_MODE	指令脈衝輸出類型	45131	UINT16
POUT_REVERSE	指令脈衝輸出邏輯反相	45132	UINT16
ALM_FILTER	報警(ALM) 埠的濾波時間	45141	UINT16
NEL_FILTER	負方向硬限位埠濾波時間	45142	UINT16
PEL_FILTER	正方向硬限位埠濾波時間	45143	UINT16

ORG_FILTER	原點信號埠濾波時間	45144	UINT16
IN1_FILTER	IN1 埠的濾波時間	45145	UINT16
IN2_FILTER	IN2 埠的濾波時間	45146	UINT16
IN4_FILTER	IN4 埠的濾波時間	45147	UINT16
IN5_FILTER	IN5 埠的濾波時間	45148	UINT16
EXT_SRC	哪個軸的外部驅動輸入埠作為外部驅動信號源	45161	UINT16
EXT_EN	暫無作用，預留	45162	UINT16
EXT_MODE	手輪模式外部驅動的脈衝輸入模式	45163	UINT16
EXT_PULSE	手輪模式外部驅動時，每個手輪脈衝輸入對應多少個指令脈衝輸出值	45164	UINT32

- 軸 1 參數、配置的 Modbus 地址表

變數	說明	Modbus 地址	資料類型
VL	軸初速度	45301	F32
VH	軸運行速度	45303	F32
ACC	軸加速度	45305	F32
DEC	軸減速度	45307	F32
JK	速度曲線類型：S 型或 T 型	45309	F32
MAXVEL	軸運行速度限值	45311	F32
MAXACC	軸加速度限值	45313	F32
MAXDEC	軸減速度限值	45315	F32
MAXJK	暫無作用，預留	45317	F32
INSTOP_DEC	INSTOP 功能中的減速度	45319	F32
UNIT_NUM	脈衝當量分子	45331	UINT32
UNIT_DENOM	脈衝當量分母	45333	UINT32
HOME_CROSS	回原點運行中的跨越距離	45351	F32
HOME_EX	暫無作用，預留	45353	UINT16

ORG_LOGIC	ORG 信號的有效邏輯電平	45354	UINT16
ORG_MODE	回原點運動結束時的停止模式	45355	UINT16
EZ_LOGIC	Z 相輸入信號的有效邏輯電平	45356	UINT16
HOME_RESET	回原點後清零位置值功能	45357	UINT16
HOME_OFFSETDIST	回原點成後的偏移距離	45358	F32
HOME_OFFSETVEL	回原點成後偏移距離的移動速度	45360	F32
HOME_MODE	回原點模式	45362	UINT16
HOME_VL	回原點運動的初速度	45363	F32
HOME_VH	回原點運動的運行速度	45365	F32
HOME_ACC	回原點運動的加速度	45367	F32
HOME_DEC	回原點運動的減速度	45369	F32
HOME_JK	回原點運動的速度曲線類型	45371	F32
JOG_VLTIME	JOG 運動低段速度運行的時間	45381	UINT32
JOG_VL	JOG 運動低段速度	45383	F32
JOG_VH	JOG 運動高段速度	45385	F32
JOG_ACC	JOG 運動的加速度	45387	F32
JOG_DEC	JOG 運動的減速度	45389	F32
EL_EN	使能硬體限位元功能	45401	UINT16
EL_LOGIC	硬體限位元信號的有效邏輯電平	45402	UINT16
EL_MODE	硬限位元觸發時的停止模式	45403	UINT16
PEL_TOL_EN	正方向硬極限容差功能使能	45404	UINT16
PEL_TOL	正方向硬極限容差值	45405	UINT32
NEL_TOL_EN	負方向硬極限容差功能使能	45407	UINT16
NEL_TOL	負方向硬極限容差值	45408	UINT32
PIN_MODE	編碼器脈衝輸入類型	45421	UINT16
PIN_LOGIC	編碼器脈衝輸入邏輯反相	45422	UINT16
PIN_MAXFREQ	編碼器脈衝輸入的最高頻率限值	45423	UINT16

POUT_MODE	指令脈衝輸出類型	45431	UINT16
POUT_REVERSE	指令脈衝輸出邏輯反相	45432	UINT16
ALM_FILTER	報警(ALM) 埠的濾波時間	45441	UINT16
NEL_FILTER	負方向硬限位埠濾波時間	45442	UINT16
PEL_FILTER	正方向硬限位埠濾波時間	45443	UINT16
ORG_FILTER	原點信號埠濾波時間	45444	UINT16
IN1_FILTER	IN1 埠的濾波時間	45445	UINT16
IN2_FILTER	IN2 埠的濾波時間	45446	UINT16
IN4_FILTER	IN4 埠的濾波時間	45447	UINT16
IN5_FILTER	IN5 埠的濾波時間	45448	UINT16
EXT_SRC	哪個軸的外部驅動輸入埠作為外部驅動信號源	45461	UINT16
EXT_EN	暫無作用，預留	45462	UINT16
EXT_MODE	手輪模式外部驅動的脈衝輸入模式	45463	UINT16
EXT_PULSE	手輪模式外部驅動時，每個手輪脈衝輸入對應多少個指令脈衝輸出值	45464	UINT32

- ……

- 軸 15 參數、配置的 Modbus 地址表

變數	說明	Modbus 地址	資料類型
VL	軸初速度	49501	F32
VH	軸運行速度	49503	F32
ACC	軸加速度	49505	F32
DEC	軸減速度	49507	F32
JK	速度曲線類型：S 型或 T 型	49509	F32
MAXVEL	軸運行速度限值	49511	F32
MAXACC	軸加速度限值	49513	F32

MAXDEC	軸減速度限值	49515	F32
MAXJK	暫無作用，預留	49517	F32
INSTOP_DEC	INSTOP 功能中的減速度	49519	F32
UNIT_NUM	脈衝當量分子	49531	UINT32
UNIT_DENOM	脈衝當量分母	49533	UINT32
HOME_CROSS	回原點運行中的跨越距離	49551	F32
HOME_EX	暫無作用，預留	49553	UINT16
ORG_LOGIC	ORG 信號的有效邏輯電平	49554	UINT16
ORG_MODE	回原點運動結束時的停止模式	49555	UINT16
EZ_LOGIC	Z 相輸入信號的有效邏輯電平	49556	UINT16
HOME_RESET	回原點後清零位置值功能	49557	UINT16
HOME_OFFSETDIST	回原點成後的偏移距離	49558	F32
HOME_OFFSETVEL	回原點成後偏移距離的移動速度	49560	F32
HOME_MODE	回原點模式	49562	UINT16
HOME_VL	回原點運動的初速度	49563	F32
HOME_VH	回原點運動的運行速度	49565	F32
HOME_ACC	回原點運動的加速度	49567	F32
HOME_DEC	回原點運動的減速度	49569	F32
HOME_JK	回原點運動的速度曲線類型	49571	F32
JOG_VLTIME	JOG 運動低段速度運行的時間	49581	UINT32
JOG_VL	JOG 運動低段速度	49583	F32
JOG_VH	JOG 運動高段速度	49585	F32
JOG_ACC	JOG 運動的加速度	49587	F32
JOG_DEC	JOG 運動的減速度	49589	F32
EL_EN	使能硬體限位元功能	49601	UINT16
EL_LOGIC	硬體限位元信號的有效邏輯電平	49602	UINT16
EL_MODE	硬限位元觸發時的停止模式	49603	UINT16

PEL_TOL_EN	正方向硬極限容差功能使能	49604	UINT16
PEL_TOL	正方向硬極限容差值	49605	UINT32
NEL_TOL_EN	負方向硬極限容差功能使能	49607	UINT16
NEL_TOL	負方向硬極限容差值	49608	UINT32
PIN_MODE	編碼器脈衝輸入類型	49621	UINT16
PIN_LOGIC	編碼器脈衝輸入邏輯反相	49622	UINT16
PIN_MAXFREQ	編碼器脈衝輸入的最高頻率限值	49623	UINT16
POUT_MODE	指令脈衝輸出類型	49631	UINT16
POUT_REVERSE	指令脈衝輸出邏輯反相	49632	UINT16
ALM_FILTER	報警(ALM) 埠的濾波時間	49641	UINT16
NEL_FILTER	負方向硬限位埠濾波時間	49642	UINT16
PEL_FILTER	正方向硬限位埠濾波時間	49643	UINT16
ORG_FILTER	原點信號埠濾波時間	49644	UINT16
IN1_FILTER	IN1 埠的濾波時間	49645	UINT16
IN2_FILTER	IN2 埠的濾波時間	49646	UINT16
IN4_FILTER	IN4 埠的濾波時間	49647	UINT16
IN5_FILTER	IN5 埠的濾波時間	49648	UINT16
EXT_SRC	哪個軸的外部驅動輸入埠作為外部驅動信號源	49661	UINT16
EXT_EN	暫無作用，預留	49662	UINT16
EXT_MODE	手輪模式外部驅動的脈衝輸入模式	49663	UINT16
EXT_PULSE	手輪模式外部驅動時，每個手輪脈衝輸入對應多少個指令脈衝輸出值	49664	UINT32

3.4.5 特殊運動指令對應的 Modbus 地址

3.4.5.1 MOVE 運動指令

當上位機寫 1 到“指令對應的 Modbus 位址”時，對應的軸會執行單軸相對運動，運動的距離為該軸對應的“運動距離對應的 Modbus 地址”裡的值；上位機寫 0 到“指令對應的 Modbus 位址”時，控制器不做任何處理。

說明	指令對應的 Modbus 地址 (資料類型：Bit)	運動距離對應的 Modbus 地址 (資料類型：F32)
軸 0 相對運動	9001	44501
軸 1 相對運動	9002	44503
……		
軸 15 相對運動	9016	44531

例：上位機寫 1 到地址 9002，此時如果地址 44503 裡的值為 1000，則軸 1 執行單軸相對運動一次，運動距離為 1000 個脈衝當量。

3.4.5.2 JOG 運動指令

當上位機寫 1 到“指令對應的 Modbus 位址”時，對應的軸會執行 JOG 運動。當上位機寫 0 到“指令對應的 Modbus 位址”時，對應的軸會停止當前的 JOG 運動。

說明	指令對應的 Modbus 地址(資料類型：Bit)
軸 0 正向 JOG 運動	9033
軸 1 正向 JOG 運動	9034
……	
軸 15 正向 JOG 運動	9048
軸 0 負向 JOG 運動	9065
軸 1 負向 JOG 運動	9066
……	
軸 15 負向 JOG 運動	9080

例：上位機做一個按鈕。按鈕按下時，上位機寫 1 到地址 9034，軸 1 開始持續執行正向 JOG 運動；按鈕抬起時，位元機寫 0 到地址 9034，軸 1 停止正向 JOG 運動。

3.4.5.3 回原點運動指令

當上位機寫 1 到“指令對應的 Modbus 位址”時，對應的軸會執行回原點運動。當上位機寫 0 到“指令對應的 Modbus 位址”時，控制器不做任何處理。

說明	指令對應的 Modbus 地址(資料類型：Bit)
軸 0 正向回原點運動	9097
軸 1 正向回原點運動	9098
……	
軸 15 正向回原點運動	9112
軸 0 負向回原點運動	9129
軸 1 負向回原點運動	9130
……	
軸 15 負向回原點運動	9144

例：上位機寫 1 到地址 9098，控制會根據系統裡設置的回原點相關參數，控制軸 1 執行正向回原點運動一次。

3.4.5.3 停止運動指令

當上位機寫 1 到“指令對應的 Modbus 位址”時，控制器會控制對應的軸停止運動。當上位機寫 0 到“指令對應的 Modbus 位址”時，控制器不做任何處理。

說明	指令對應的 Modbus 地址(資料類型：Bit)
停止軸 0 運動	9161
停止軸 1 運動	9162
……	
停止軸 15 運動	9176

例：上位機寫 1 到地址 9162，控制器會控制軸 1 停止運動。