研华MAS控制器

Motion Basic编程手册

Version 1.4

第 1 章　研华 MOTION BASIC 介绍

1.1　MOTION BASIC 概述

Motion Basic是一种用于研华MAS控制器的多任务编程语言，它的语法与标准Basic相似。在装有windows操作系统的电脑上运行Motion Studio就可以进行开发和测试所有MAS控制器功能。Motion Studio提供了Basic程序编辑和丰富的调试工具来进行应用程序的调试，使设备应用程序开发变得简单高效。

1.2　特点

- 基本程序指令和Visual Basic兼容，运动控制指令十分简单，初学者也很容易上手。
- 支持10个任务独立执行、同时执行。
- 支持所有研华softmotion功能指令，并可整合进外部算法和函数到Motion Basic指令，非常适合设备整合控制和高效开发。
- 编译执行方式，使得在简单易用的前提下，同时具有高级语言高效执行的优点。
- 可以同时运行多个子任务 ,强大的多任务处理功能提高软件结构和易维护性。
- 稳定性高：当用户程序出现错误时，不会出现系统崩溃。

常见的三种指令特点比较表

	MOTION BASIC	G 代码	梯形图
可理解方式	较容易	容易	困难
执行方式	编译执行	解释执行	编译执行
执行速度	快	较慢	慢
功能	全	较少	全
操作硬件能力	所有	只支持电机、IO	所有
子程序	支持	支持	不支持
任务数	10	1 主任务/3 从任务	不支持
变量	支持	不支持	支持
数组	支持	不支持	不支持
数学函数	支持	不支持	支持

第 2 章　BASIC 指令详解

MOTION BASIC 规则说明

研华 Motion Basic 指令主要包含运动控制指令和其他指令两大类。根据研华运动控制的特点，我们把运动控制指令分为命令和属性两部分指令，在第 2 章 Basic 指令详解中有说明每条运动控制指令属于命令还是属性。其他指令的规则基本上类似于标准 Basic 指令规则。本节将研华 MOTION BASIC 指令说明的一些重要规则描述如下：

- **研华 MOTION BASIC 指令集不区分字母大小写**
- **指令语法说明方法：语法说明中包含( )的部分，必须要使用( )，不可省略。说明中有[ ]的部分，表示[ ]中的内容根据实际需求可以填或不填，不会造成语法错误。**

  如下 LINE 指令说明：

  LINE　distance1,distance2 [,distance3]；做两轴直线插补时，LINE 指令后面的参数填 distance1 和 distance2 就可以了。做三轴直线插补时，LINE 指令后面的参数就填 distance1,distance2,distance3。[,distance3]这个部分是根据实际需求选填的，选填的部分我们用[ ]说明。
- **运动控制属性类指令的赋值说明**

  运动控制属性类指令赋值是用“=”来做赋值的，如要设置加速度，要用 ACC=value 这样的语句，ACC value 这样的语句则不符合语法规则。
- **运动控制指令中对指定轴的操作说明**

  运动控制指令对指定轴操作时，指定轴号时要使用“AX”这个关键字，仅用数字代表轴号不符合语法规则。如将锁存到的轴 0 理论位置值赋给一个变量 A，会用到指令 LDPOS，语句是 A=LDPOS(AX(0))，A=LDPOS(0)则不符合语法规则。
- **系统已定义运动控制关键字**

  注意:用户自定义的变量名不能与以下表格中的关键字或枚举名一样，也不能与 Motion Basic 的所有指令名一样。

关键字或枚举	说明	例子
AX	指定轴号的关键字 ,AX(0)代表轴 0，AX(5)代表轴 5	MVOE AX(0),1000 表示命令轴 0 运动 1000 个脉冲当量
CW	顺时针方向	CIRC 0,5000,0,10000,0 可以用 CIRC CW,5000,0,10000,0 来代替
CCW	逆时针方向	CIRC 1,5000,0,10000,0 可以用 CIRC CCW,5000,0,10000,0 来代替

S	S 型曲线	JK=0 可以用 JK=T 来代替
T	T 型曲线	JK=1 可以用 JK=S 来代替
DONE	轴运动完成事件	WAIT DONE；参考 WAIT 指令说明
COMPARED	比较触发完成事件	WAIT COMPARED；参考 WAIT 指令说明
LATCHED	锁存完成事件	WAIT LATCHED ；参考 WAIT 指令说明
AXIS	系统内部定义的轴关键字	
RUN	系统内部定义的运行关键字	
STOP	系统内部定义的停止关键字	
TASK	系统内部定义的任务关键字	
DIR	系统内部定义的方向关键字	

2.1 流程控制语句

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.1.1	DIM...As...	定义数据类型	×	×
2.1.2	CONST	定义常量	×	×
2.1.3	IF...THEN...ELSEIF ...ELSE...END IF	IF 条件语句	×	×
2.1.4	FOR...TO... STEP...NEXT	FOR 循环语句	×	×
2.1.5	Select Case...End Select	CASE 语句	×	×
2.1.6	WHILE...WEND	While 循环语句	×	×
2.1.7	WAIT	等待事件结束	×	×
2.1.8	CancelWAIT	退出事件的等待	×	×
2.1.9	SLEEP	延时	×	×
2.1.10	TYPE	定义类	×	×
2.1.11	EXIT	退出一个控制流语句块或函数体	×	×

2.1.1 DIM...As...

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

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

描　述：　定义变量，数组

参　数：　varname　　变量名

注　意：　在一个 Task 里用 DIM 定义的局部变量只在该 Task 当前的程序段起到声明的作用，如果要声明到该 Task 的子程序段（如 SUB 段程序），那需要在 DIM 后面加上关键字 shared，如这样定义一个变量 x：DIM shared x As Integer。

例　程

```
DIM a AS DOUBLE                    '定义一个变量名为 a 的 64 位浮点型变量
DIM text AS STRING ="Hi,MAS"       '定义一个变量名为 text 的字符串，并赋值为 HI,MAS
DIM Array(3) AS BYTE={1,2,5}       '定义一个长度为 3，名为 Array 的 BYTE 类型数组,并赋值
DIM AS ULONG b,c,d,e               '一次定义多个同类型变量
```

2.1.2 CONST

语　法：CONST　varname = value

描　述：定义常量。采用常量名可避免在程序中多处修改同一个数值

参　数：　varname　　变量名

　　value　　常数值

例　程

```
CONST max_speed=50000          '定义速度常量
CONST Distance=10000           '定义位移常量
BASE 0
SVON
VH = max_speed                 '将速度常量赋给运行速度属性
MOVE Distance                  '将位移常量赋给位移指令参数
```

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

语　法：IF <condition1> THEN

　　commands1

　[ELSEIF <condition2> THEN]

　　commands2

　[ELSE commands3]

　END IF

描　述：该指令为条件语句。首先判断条件表达式 1；如果条件表达式 1 成立，则执行指令块 1，然后跳转至 END IF，退出条件语句。如果条件表达式 1 不成立，则判断条件表达式 2；如果条件表达式 2 成立，则执行指令块 2，然后跳转至 endif，退出条件语句。如果条件表达式 2 不成立，则执行指令块 3

参　数：　condition　　条件表达式

　　commands　　指令块（可单条指令也可多条指令）

例　程

```
DIM a AS LONG
IF(a=0)    THEN                    '判断条件 1：a 是否为 0
    DOUT (2)=1                     '条件 1 成立，则 DO2, DO6 分别赋值 1,1
    DOUT (6)=1
ELSEIF( a>0 AND a<=5) THEN         '判断条件 2：a 是否大于 0 且小于等于 5
    DOUT(2)=0                      '条件 2 成立，则 DO2, DO6 分别赋值 0,1
    DOUT(6)=1
ELSE                               '条件 1、2 都不成立的情况，则 DO2, DO6 分别赋值 1,0
    DOUT(2) =1
    DOUT(6)=0
END IF
```

2.1.4　FOR...TO...STEP...NEXT

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

[comands]

NEXT　varname

描　述：FOR 循环语句。如果循环变量小于循环结束值，则执行指令块到 NEXT 时，循环变量自动加一个增量，再一次执行指令块；当循环变量大于等于循环结束值时，则停止循环。

参　数：

varname	循环体变量名
startvalue	循环起始值
endvalue	循环结束值
stepvalue	循环变量的增量，可选；缺省时，增量为 1。增量也可以是小数或负数。增量为负数时，循环起始值要大于循环结束值。
Comands	指令块

注　意：　该指令最多嵌套八层

例　程

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

```
NEXT i
'将 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

语　法：Select Case expression

[Case expressionlist]

[commands]

.....

[Case expressionlist]

[commands]

[Case Else]

[commands]

End Select

描　述：类似 C 语言的 Switch...Case 条件语句。根据表达式的内容选择执行哪个指令块 ,各分支 CASE 表达式条件内容需互斥 ,表达式内容会逐一匹配各 CASE 的条件内容，直到匹配成功，一旦匹配成功，相应分支 CASE 下的指令块只执行一次，然后程序跳转到 End select，结束 CASE 条件语句。如果写了 Case Else，则等 Case Else 以上的各分支 CASE 都没有匹配成功时，就会执行 Case Else 下的指令块。

参　数：　expression　　表达式

expressionlist　case 分支表达式条件内容

comands　　指令块

注　意：　各 case 分支表达式内容需互斥

例　程

```
Dim choice As LONG
Select Case choice
Case 1
    Print "number is 1"                    'choice 为 1 时，打印 number is 1
Case 2
    Print "number is 2"                    'choice 为 2 时，打印 number is 2
```

```
Case 3, 4
    Print "number is 3 or 4"                    'choice 为 3 或 4 时，打印 number is 3 or 4
Case 5 To 10
    Print "number is 5 to 10"                   'choice 为 5~10 时，打印 number is 5 to 10
Case Else
    Print "number is outside the 1-10 range"    '非以上情况 ,打印 number is outside the 1-10 range
End Select
```

2.1.6 WHILE...WEND

语　法： WHILE condition
commands
WEND

描　述：循环语句。当 condition 条件成立时，执行循环体内的指令块；否则，结束循环体。

参　数： condition　　条件表达式
comands指令块

例　程

```
Dim i AS LONG
WHILE(i=5)                  '如果 i=5，则在 WHILE 和 WEND 之间的指令段循环执行
   BASE 0
   SVON
   MOVE 1000
   WAIT DONE
   MOVE -1000
   WAIT DONE
WEND
```

2.1.7 WAIT

语　法 1：WAIT [AX(no),]DONE

语　法 2：WAIT [AX(no),] LATCHED

语　法 3：WAIT [AX(no),] COMPARED

语　法 4：WAIT [CYL(no),] CYLDONE

描　述：WAIT 指令表示等待某个事件结束，WAIT 后面未指定轴或汽油缸时，等待的是当前 BASE 列表中的所有轴或汽油缸的相应事件。WAIT 指令后面跟的事件暂只支持 DONE、LATCHED、COMPARED、CYLDONE 四种事件。

参　数：

AX(no)	指定轴号，no 填轴号数字
DONE	指运动结束事件
LATCHED	指锁存发生事件
COMPARED	指比较触发事件
CYLDONE	指气缸动作完成事件

注　意：WAIT 指令使用时要小心，未等到事件发生，程序会一直停在 WAIT 这行。特别是等待多个事件时，要确保事件都会发生。

例　程

```
BASE 0,1,2
MOVE 10000,20000,30000
WAIT AX(2),DONE              '等待轴 2 运动结束后，程序执行下一行，否则一直等在该行
BASE 0,1
MOVE 20000,5000
WAIT DONE                    '等待轴 0,1 运动都结束后，程序执行下一行，否则一直等在该行
BASE 0
MOVE 10000
WAIT DONE                    '等待轴 0 运动结束后，程序执行下一行，否则一直等在该行
DOUT (2) =1
```

2.1.8　CancelWAIT

语　法 1：CancelWAIT [AX(no),]DONE

语　法 2：CancelWAIT [AX(no),] LATCHED

语　法 3：CancelWAIT [AX(no),] COMPARED

描　述：CancelWAIT 指令表示退出等待某个事件结束，一般是对应 WAIT 指令来使用的。当执行了 CancelWAIT 指令后，当前系统所有 Task 正在执行的 Wait 事件语句会退出等待 ,各 Task 程序会立刻接着执行 Wait 语句后面的程序行。

参　数：

AX(no)	指定轴号，no 填轴号数字
DONE	指运动结束事件
LATCHED	指锁存发生事件

COMPARED 指比较触发事件

例　程

```
'下面例程有两个 Task 同时执行，Task 2 开始执行后会，轴 0 一直处于正向连续运动状态，程序会等
'待在 WAIT DONE 指令行。这时候如果 VR(0)值变为 1，Task 1 中会执行 CancelWAIT DONE'，
'Task 2 中 WAIT DONE 指令就会退出等待，程序会继续执行 STOPDEC 这行指令，然后 DOUT(0)就
'会置成 ON 状态
'***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          '执行正向连续运动
WAIT DONE        '等待运动结束
STOPDEC          '减速停止
DOUT(0)=1        'DO0 ON
'***Task 2***'
```

2.1.9　SLEEP

语　法：SLEEP delay_time

描　述：延时指令

参　数：delay_time　延时时间，单位：ms

例　程

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

2.1.10 TYPE

语　法：TYPE type_name

描　述：定义一个类（类似面向对象语言中的类）

参　数：type_name　类的名称

例　程

```
'最终打印出来的结果如下
' Number of Values =  4
'Average             =  19.2
'以下为例程
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

语　法：EXIT　Do | For | While | Select

EXIT　Sub | Function

EXIT　DO[, DO[,...]]

EXIT　For[, For[,...]]

EXIT　While[, While[,...]]

EXIT　Select[, Select[,...]]

描　述：退出一个控制流语句块或函数体。

例　程

```
DIM i AS USHORT
FOR i=0 To 7
        DOUT (i) =1
        IF(i=5) THEN
           EXIT FOR '当 i 为 5 时，结束 FOR 循环体
        END IF
NEXT i


DIM As Integer i, j
For i = 1 To 10
   For j = 1 To 10
      Exit For, For    '嵌套的控制流语句块退出指令，结束 FOR 嵌套循环体
   Next j
   Print "I will never be shown"
Next i
```

2.2 子程序、多任务控制语句

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.2.1	SUB	定义一个函数体	×	×
2.2.2	END	结束当前任务或程序	×	×
2.2.3	RUN_TASK	运行指定的 Task	√	×
2.2.4	STOP_TASK	停止运行指定的 Task	√	×
2.2.5	STOP_ALL	停止运行所有 Task ,并停止所有轴的运动	√	×

2.2.1 SUB

语　法：SUB　name

描　述：定义一个函数体

参　数：name　函数体名

注　意：调用函数时，SUB 函数体必须要写在调用该函数之前，否则编译时会出错。如果想将定义的 SUB 函数体写在任意位置，可以在 TASK 开头用 **Declare Sub name** 语句先定义。

Declare 指令特别说明如下：

Declare 为声明的指令，一般用于声明 SUB（无返回值的函数体）和 Function（有返回值的函数体）,语法如下：

Declare Sub name [param_list]

Declare Function name [param_list] As return_type

例　程

```
'定义一个函数体，名为 abc
SUB abc
BASE 0
SVON
```

```
MOVE 50000
WAIT DONE
END SUB

BASE 1
SVON
MOVE 10000
WAIT DONE
abc                    '调用函数名为 abc 的函数体
BASE 1
MOVE -10000
```

2.2.2 END

语　法：END

描　述：结束当前任务或程序，END 指令后面可以跟 SUB，IF 等

例　程

```
'可以参考 IF、SUB 等指令例程
```

2.2.3 RUN_TASK

所　属：命令

语　法：RUN_TASK　“task_name”

描　述：运行指定的 Task

参　数： task_name　TASK 文本名称，此处名称不能加 .bas 后缀，只需填.bas 前面的名称

例　程

```
'如用户创建了 3 个 TASK，分别为 test1.bas，test2.bas,test3.bas，需在 test3.bas 里运行 test1.bas，
test2.bas 这两个 TASK，可在 test3.bas 里写如下例程
RUN_TASK "test1"
RUN_TASK "test2"
```

2.2.4 STOP_TASK

所　属：命令

语　法：STOP_TASK　“task_name”

描　述：停止运行指定的 Task

参　数： task_name　TASK 文本名称，此处名称不能加 .bas 后缀，只需填.bas 前面的名称

例　程

```
STOP_TASK "test"          ' 停止运行名为 test 的 TASK
```

2.2.5 STOP_ALL

所　属：命令

语　法：STOP_ALL

描　述：停止运行所有 Task，并停止所有轴的运动

例　程

```
RUN_TASK "test1"          ' 运行名为 test1 的 TASK
RUN_TASK "test2"          ' 运行名为 test2 的 TASK
Sleep 1000                '延时 1 秒
STOP_ALL                  ' 停止运行所有 TASK，并停止所有轴运动
```

2.3 运算符及数学函数

2.3.1 运算符

当表达式包含多种运算符时，首先计算算术运算符，然后计算比较运算符，最后计算逻辑运算符。所有比较运算符的优先级相同，即按照从左到右的顺序计算比较运算符。

当乘号与除号同时出现在一个表达式中时，按从左到右的顺序计算乘、除运算符。同样当加与减同时出现在一个表达式中时，按从左到右的顺序计算加、减运算符。

算术运算符说明如表 3.1 所示。

表 3.1 运算符说明

算术运算符		比较运算符		逻辑运算符	
描述	符号	描述	符号	描述	符号
加	+	等于	=	逻辑非	NOT
减	-	不等于	<>	逻辑与	AND
乘	*	小于	<	逻辑或	OR
除	/	大于	>	逻辑异或	XOR
整除	\	小于等于	<=	逻辑等价	EQV
求余数	Mod	大于等于	>=		
求相反数	-				
幂	^				

2.3.1.1 NOT

语　法：NOT expression

描　述：对表达式进行非操作或对数值按二进制的位进行“非”运算

参　数：expression　　表达式

注　意：只针对表达式的值的整数部分运算

2.3.1.2 AND

语　法：expression1 AND expression2

描　述：对两个表达式进行与操作或对两个数值按二进制的位进行“与”运算

参　数：expression1　　表达式 1

expression2　　表达式 2

注　意：只针对表达式的值的整数部分运算

2.3.1.3　OR

语　法：expression1 OR expression2

描　述：对两个表达式进行或操作或对两个数值按二进制的位进行“或”运算

参　数：expression1　　表达式 1

expression2　　表达式 2

注　意：只针对表达式的值的整数部分运算

2.3.1.4　XOR

语　法：expression1 XOR expression2

描　述：对两个表达式进行异或操作或对两个数值按二进制的位进行“异或”运算

参　数：expression1　　表达式 1

expression2　　表达式 2

注　意：只针对表达式的值的整数部分运算

2.3.1.5　EQV

语　法：expression1 EQV expression2

描　述：对两个表达式进行同或操作或对两个数值按二进制的位进行“同或”运算

参　数：expression1　　表达式 1

expression2　　表达式 2

注　意：只针对表达式的值的整数部分运算

2.3.2 数学函数

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.3.2.1	ABS	求绝对值	×	×
2.3.2.2	SIN	正弦函数	×	×
2.3.2.3	ASIN	反正弦函数	×	×
2.3.2.4	COS	余弦函数	×	×
2.3.2.5	ACOS	反余弦函数	×	×
2.3.2.6	TAN	正切函数	×	×
2.3.2.7	ATN	反正切函数	×	×
2.3.2.8	ATAN2	用比值求反正切函数	×	×
2.3.2.9	SQR	求平方根	×	×
2.3.2.10	LOG	自然对数	×	×
2.3.2.11	BITRESET	位操作置 0	×	×
2.3.2.12	BIT	位操作读值	×	×
2.3.2.13	BITSET	位操作置 1	×	×
2.3.2.14	FRAC	求小数部分的值	×	×
2.3.2.15	INT	求整数部分的值	×	×
2.3.2.16	SGN	判断值的符号	×	×

2.3.2.1 ABS

语　法：ABS(expression)

描　述：求表达式的绝对值

参　数：expression　　表达式

2.3.2.2 SIN

语　法：SIN (expression)

描　述：计算表达式的正弦函数

参　数：expression　　表达式，单位：弧度

例程

```
CONST PI AS DOUBLE = 3.1415926535897932
DIM a AS DOUBLE                 '定义一个变量用于存角度值
DIM r AS DOUBLE                 '定义一个变量用于存弧度值
DIM sin_value as DOUBLE         '定义一个变量用于取正弦值
a=90
r = a*PI/180                    '把 a 转换成弧度值存于 r
sin_value=SIN (r)
PRINT r
PRINT sin_value
```

运行结果：r 为 1.570796326794897

sin_value 为 1

2.3.2.3　ASIN

语　法：ASIN (expression)

描　述：计算表达式的反正弦函数，返回值单位：弧度

参　数：expression　　表达式

2.3.2.4　COS

语　法：COS (expression)

描　述：计算表达式的余弦函数

参　数：expression　　表达式，单位：弧度

2.3.2.5　ACOS

语　法：ACOS (expression)

描　述：计算表达式的反余弦函数　，返回值单位：弧度

参　数：expression　　表达式

2.3.2.6　TAN

语　法：TAN (expression)

描　述：计算表达式的正切函数

参　数：expression　　表达式，单位：弧度

2.3.2.7 ATN

语　法：ATN (expression)

描　述：计算表达式的反正切函数，返回值单位：弧度

参　数：expression

2.3.2.8 ATAN2

语　法：ATAN2　(number1，number2)

描　述：计算 number1/number2 比值的反正切函数，返回值单位：弧度

参　数：number1　　比值分子

number2　　比值分母

2.3.2.9 SQR

语　法：SQR (expression)

描　述：计算表达式的平方根

参　数：expression　　表达式

2.3.2.10 LOG

语　法：LOG (expression)

描　述：计算表达式的自然对数（以 e 为底的对数）

参　数：expression　　表达式

2.3.2.11 BITRESET

语　法：BITRESET(value，bit_num)

描　述：将操作数的二进制的第 bit 位清 0

参　数：bit_num　　位编号：0~31（二进制数的位，从低（右）至高（左）排列）

value　　操作数

注　意：只针对操作数的整数部分操作

例　程

```
DIM AS ULONG a,b
a=5
b=BITRESET(a,0)          'a 为 5，二进制即 101，清掉 0 位，即 b 得到 100
PRINT b                  'b 为 100，打印出来即为 4
b=BITRESET(a,2)          'a 为 5，二进制即 101，清掉 2 位，即 b 得到 001
```

```
PRINT b                    'b 为 001，打印出来即为 1
```

2.3.2.12 BIT

语　法：BIT(value，bit_num)

描　述：读取操作数的二进制的第 bit 位的值，读到 bit 位为 0 值，返回 0；读到 bit 位为 1 值，返回-1

参　数：bit_num　　位编号：0~31（二进制数的位，从低（右）至高（左）排列）

value　　操作数

注　意：只针对操作数的整数部分操作

2.3.2.13 BITSET

语　法：BITSET(value，bit_num)

描　述：将操作数的二进制的第 bit 位的置 1

参　数：bit_num　　位编号：0~31（二进制数的位，从低（右）至高（左）排列）

value　　操作数

注　意：只针对操作数的整数部分操作

2.3.2.14 FRAC

语　法：FRAC(expression)

描　述：返回表达式的小数部分

参　数：expression　　表达式

注　意：此函数仅支持大于 0 的表达式

2.3.2.15 INT

语　法：INT(expression)

描　述：返回表达式的整数部分

参　数：expression　　表达式

注　意：当表达式的值小于 0 时，返回值比其整数小 1

2.3.2.16 SGN

语　法：SGN(expression)

描　述：判断表达式是大于 0、等于 0，还是小于 0。当表达式大于 0，函数的返回值为 1；当表达式等于 0 时，函数的返回值为 0；当表达式小于 0，函数的返回值为 - 1

参　数：expression　　表达式

2.4 控制器系统指令

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.4.1	BASE	该指令后面所有的轴指令和轴参数设置和读取都基于该指令选定的轴号	√	×
2.4.2	UNIT_NUM	脉冲当量分子	√	√
2.4.3	UNIT_DENOM	脉冲当量分母	√	√
2.4.4	POUT_MODE	指令脉冲输出类型	√	√
2.4.5	POUT_REVERSE	指令脉冲输出逻辑反相	√	√
2.4.6	PIN_MODE	编码器脉冲输入类型	√	√
2.4.7	PIN_MAXFREQ	编码器脉冲输入的最高频率限值	√	√
2.4.8	PIN_LOGIC	编码器脉冲输入逻辑反相	√	√
2.4.9	DPOS	理论位置值（指令脉冲）	√	√
2.4.10	MPOS	实际位置值（编码器反馈脉冲）	√	√
2.4.11	SVON	使能伺服	√	×
2.4.12	SVOFF	禁用使能伺服	√	×
2.4.13	FILE_WRITEVR	写 VR 文件到本地	×	×
2.4.14	FILE_READVR	将本地 VR 文件载入项目	×	×
2.4.15	ERROR_AXIS	当前哪些轴发生了轴状态错误	√	×
2.4.16	RUN_ERROR	轴错误信息	√	×
2.4.17	SYSTEM_ERROR	系统级错误信息	√	×
2.4.18	CLEAR_ERROR	清除系统错误状态	√	×
2.4.19	PRINT	在 Motion Studio 中的信息输出窗体打印信息	×	×
2.4.20	VR	实数型全局变量	√	√
2.4.21	DATE	获取当前控制器日期：月-日-年	×	×
2.4.22	TIME	获取当前控制器日期：小时：分	×	×

		钟：秒		
2.4.23	SETDATE	给控制器系统设置新的日期	×	×
2.4.24	SETTIME	给控制器系统设置新的时间	×	×
2.4.25	TIMER	返回程序段程序执行的时间	×	×

2.4.1 BASE

所　属：命令

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

描　述：为了简化编程，可以用该指令选择要参与运动的轴号，其后的指令就没必要填写所有轴的参数，只填写参与运动的轴参数即可。轴号要按顺序填写，轴号可以是 1 个，也可以是 2 个、3 个…

参　数：axis no 轴号；范围：根据控制器实际硬件决定。

例　程

```
BASE 0,1,2,3
VH=8000                          '轴 0,1,2,3 的最大运行速度都设置为 8000
MOVE 10000，4000，2000，6000    '轴 0,1,2,3 都执行单轴相对点位运动
BASE 1
VH=1000                          '轴 1 的最大运行速度设置为 1000
MOVE 10000                       '轴 1 执行单轴相对点位运动
```

2.4.2 UNIT_NUM

所　属：属性

语　法：UNIT_NUM = value

类　型：ULONG

描　述：设置/读取脉冲当量分子

范　围：大于 0，默认值 1

例　程

```
BASE 0
UNIT_NUM　=10　'设置軸 0 的脉冲当量分子
```

2.4.3 UNIT_DENOM

所　属：属性

语　法：UNIT_DENOM = value

类　型：ULONG

描　述：设置/读取脉冲当量分母

范　围：(0, MAX_PULSE) ，默认值 1

例　程

```
BASE 0
UNIT_DENOM=40     '设置軸 0 的脉冲当量分母为 40
```

2.4.4 POUT_MODE

所　属：属性

语　法：POUT_MODE= value

类　型：ULONG

描　述：设置/读取指令脉冲输出类型

范　围:如下设定值，默认值 5

0：OUT/DIR

1：OUT/DIR，OUT 负逻辑

2：OUT/DIR，DIR 负逻辑

3：OUT/DIR，OUT&DIR 负逻辑

4：CW/CCW

5：CW/CCW，CW&CCW 负逻辑

例　程

```
BASE 0
POUT_MODE =2      '设置指令脉冲输出类型为 OUT/DIR，DIR 负逻辑
```

2.4.5 POUT_REVERSE

所　属：属性

语　法：POUT_REVERSE = value

类　型：ULONG

描　述：启用/禁用指令脉冲输出端口信号对调

范　围:如下设定值，默认值 0

0：禁用

1：启用

例　程

BASE 0

POUT_REVERSE　=1 '启用指令脉冲输出端口信号对调

2.4.6 PIN_MODE

所　属：属性

语　法：PIN_MODE= value

类　型：ULONG

描　述：设置/读取编码器输入脉冲类型

范　围：如下设定值，默认值 2

0：1XAB

1：2XAB

2：4XAB

3：CCW/CW

例　程

BASE 0

PIN_MODE =3 '设置编码器输入脉冲类型为 CCW/CW

2.4.7 PIN_MAXFREQ

所　属：属性

语　法：PIN_MAXFREQ = value

类　型：ULONG

描　述：设置/读取编码器输入脉冲的最高频率

范　围：如下设定值，默认值 0

0：500KHz

1：1MHz

2：2MHz

3：4MHz

例　程

BASE 0

```
PIN_MAXFREQ =1  '设置编码器输入脉冲的最高频率为 1MHz
```

2.4.8 PIN_LOGIC

所　属：属性

语　法：PIN_LOGIC = value

类　型：ULONG

描　述：设置/读取编码器输入脉冲的逻辑

范　围：如下设定值，默认值 0

0：不反转方向

1：反转方向

例　程

```
BASE 0
PIN_LOGIC =1   '设置编码器输入脉冲逻辑为反转方向
```

2.4.9 DPOS

所　属：属性

语　法：DPOS = value

类　型：DOUBLE

描　述：设置/读取轴当前的理论位置

范　围：64 位浮点数据类型范围

例　程

```
BASE 0
DIM A AS DOUBLE
A=DPOS    '将轴 0 的当前理论位置赋值给变量 A
BASE 1
VR(10)=DPOS '将轴 1 的当前理论位置赋值给全局变量 VR（10）
DPOS=0 '将轴 1 的当前理论位置计数器赋 0
```

2.4.10 MPOS

所　属：属性

语　法：MPOS = value

类　型：DOUBLE

描　述：设置/读取轴当前的编码器反馈位置

范　围：64 位浮点数据类型范围

例　程

```
BASE 0
DIM A AS DOUBLE
A=MPOS     '将轴 0 的当前实际位置赋值给变量 A
BASE 1
VR(10)=MPOS '将轴 1 的当前实际位置赋值给全局变量 VR（10）
MPOS=0 '将轴 1 的当前实际位置计数器赋 0
```

2.4.11 SVON

所　属：命令

语法 1：SVON

语法 2：SVON AX(axis no)

描　述： BASE 轴列表的轴或指定轴，使能轴

参　数：axis no　 轴号； 范围：根据控制器实际硬件决定

例　程

```
'对 BASE 列表中的轴使能，即使能伺服
BASE 2
SVON                '使能轴 2
BASE 0,1,3,5
SVON                '使能轴 0、1、3、5
'指定轴使能
SVON AX(2)          '使能轴 2
```

2.4.12 SVOFF

所　属：命令

语法 1：SVOFF

语法 2：SVOFF AX(axis no)

描　述： BASE 轴列表的轴或指定轴，禁用轴使能

参　数：axis no　 轴号； 范围：根据控制器实际硬件决定

例　程

```
'对 BASE 列表中的禁用轴使能
BASE 2
```

```
SVOFF            '禁用轴 2 使能
BASE 0,1,3,5
SVOFF            '禁用轴 0、1、3、5 使能
SVOFF AX(2)      '禁用轴 2 使能
```

2.4.13 FILE_WRITEVR

所　属：命令

语　法：FILE_WRITEVR　file_name，vr_start no，vr_end no

描　述：将一段 VR 的数据保存到本地文本，目前仅支持 bas 和 csv 两种文件类型。

参　数：file_name　　保存到本地文本的文件名

vr_start no　VR 起始编号

vr_end no　　VR 结束编号

例　程

```
'将 VR(0)~VR(20)的数据写到本地名为 VR_data 的 bas 文件
FILE_WRITEVR "VR_data.bas",0,20
'将 VR(0)~VR(100)的数据写到本地名为 P_data 的 csv 文件
FILE_WRITEVR "P_data.csv",0,100
```

2.4.14 FILE_READVR

所　属：命令

语　法：FILE_READVR　file_name

描　述：将本地 VR 文本的数据写到当前工程的 VR 变量中

参　数：file_name　　保存到本地文本的文件名

例　程

```
'将本地名为 VR_data 的 bas 文件 VR 数据读到控制器里 ,该 bas 文件里的 VR 数据将覆盖控制器里对应的 VR 数据
FILE_READVR "VR_data.bas"
'将本地名为 P_data 的 csv 文件 VR 数据读到控制器里，该 csv 文件里的 VR 数据将覆盖控制器里对应的 VR 数据
FILE_READVR "P_data.csv"
```

2.4.15 ERROR_AXIS

所　属：属性 (只读)

语　法：value=ERROR_AXIS

类　型：ULONG

描　述：读取当前哪些轴发生了轴状态错误。该属性为 32 位寄存器，每一位代表一个轴。位值为 0 表示该轴无错误发生，位值为 1 表示该轴发生了轴状态错误。当发生轴状态错误时，可以用 RESETERR 指令清除轴状态错误。

返回值：0：无错误发生；1：发生了轴状态错误

例　程

```
DIM ErrorReturn As ULONG
ErrorReturn=ERROR_AXIS
IF (ErrorReturn=4) THEN   ' ErrorReturn 为 4 时，表示轴 2 发生了轴状态错误
RESETERR AX(2)            '清除轴 2 的轴状态错误
End if
```

2.4.16 RUN_ERROR

所　属：属性 (只读)

语　法：value=RUN_ERROR

类　型：ULONG

描　述：根据 BASE 轴列表中的轴，读取轴错误信息。错误代码信息请参照章节 2.15 RUN_ERROR 错误代码信息表。

返回值：错误代码

例　程

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

2.4.17 SYSTEM_ERROR

所　属：属性 (只读)

语　法：value=SYSTEM_ERROR

类　型：ULONG

描　述：读取系统级错误信息。错误代码信息请参考章节 2.15 SYSTEM_ERROR 错误代码信息表

返回值：错误代码

例　程

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

2.4.18 CLEAR_ERROR

所　属：命令

语　法：CLEAR_ERROR

描　述：清除系统错误状态。该命令仅用于清除 SYSTEM_ERROR 对应的系统错误状态

2.4.19 PRINT

所　属：命令

描　述：在 Motion Studio 中的信息输出窗体打印信息。

例　程

```
Dim A As ULONG
A=42
Print  "Hello"  '打印字符串
Print  VL    '打印初速度属性值
Print  A     '变量值
Print  3*4   '打印一个表达式结果，打印结果为 12
```

2.4.20 VR

语　法：VR(no)=value

类　型：Double

描　述：VR 变量是实数型全局变量。软件平台共提供 10000 个 VR 变量给用户操作：VR(0)~VR(9999)。当 VR 变量用于 Modbus 通讯的自定义变量时，可以选择将 VR 变量对应一个 16 位数据类型寄存器或 32 位数据类型寄存器。

参　数：no　VR 变量的索引号；范围：0~9999，共 10000 个

例　程

```
Dim A As ULONG
A=15
VR(A)=200.525       '将 200.525 赋值给 VR(15)
BASE 0
VR(25)=1000
VL=VR(25)           '将 VR(15)的数值赋给轴 0 的初速度。
```

2.4.21 DATE

语　法：value=DATE

类　型：String

描　述：获取当前控制器日期：月-日-年

例　程

```
DIM str1 AS string
str1=DATE
print str1        '如现在是 2016 年 3 月 15 日，打印结果为：03-15-2016
Print "the current date is: "; DATE        '打印结果：the current date is: 03-15-2016
```

2.4.22 TIME

语　法：value=TIME

类　型：String

描　述：获取当前控制器日期：小时 ： 分钟 ： 秒

例　程

```
DIM str1 AS string
str1=TIME
print str1        '如现在时间是 14 点 22 分 51 秒，打印结果为：14:22:51
Print "the current time is: "; TIME        '打印结果：the current date is: 14:22:51
```

2.4.23 SETDATE

语　法：SETDATE(newdate)

描　述：给控制器系统设置新的日期

参　数：newdate 新的日期，类型为 string

例　程

```
SETDATE "03/15/2016"  '将当前控制器系统日期设置为 2016 年 3 月 15 号
```

2.4.24 SETTIME

语　法：SETTIME(newtime)

描　述：给控制器系统设置新的时间

参　数：newtime 新的时间，类型为 string

例　程

```
SETTIME "14:20:31"      '将当前控制器系统时间设置为 14 点 20 分 31 秒
```

2.4.25 TIMER

语　法：value=TIMER

描　述：以秒为单位，返回开始参考的时间到现在消逝的时间总和。该功能主要用于计算 TASK 里一个程序段跑了多少时间，从而去找出从程序一行执行到另一行所花的时间。

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 单轴点位运动

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.5.1	VL	单轴初速度	√	√
2.5.2	VH	单轴运行速度	√	√
2.5.3	ACC	单轴加速度	√	√
2.5.4	DEC	单轴减速度	√	√
2.5.5	JK	单轴速度曲线类型	√	√
2.5.6	DSPEED	当前轴指令速度值	√	√
2.5.7	STATE	轴当前运动状态	√	√
2.5.8	MOVE	单轴相对点位运动	√	×
2.5.9	MOVEABS	单轴绝对点位运动	√	×
2.5.10	PCHANGE	单轴运动过程中改变终点位置	√	×
2.5.11	STOPDEC	减速停止	√	×
2.5.12	STOPEMG	立即停止	√	×
2.5.13	BACKLASH_EN	背隙补偿功能使能	√	×
2.5.14	BACKLASH_PULSE	背隙补偿距离	√	√
2.5.15	BACKLASH_VEL	背隙补偿过程运动速度	√	√
2.5.16	MAXVEL	单轴最大速度限值	√	√
2.5.17	MAXACC	单轴最大加速度限值	√	√
2.5.18	MAXDEC	单轴最大减速度限值	√	√
2.5.19	RESETERR	清除当前轴错误状态	√	×

2.5.1 VL

所　属：属性

语　法：VL = value

类　型：DOUBLE

描　述：设置/读取轴的初速度，VL 的单位为脉冲当量/s

范　围：【0, MAXVEL】，默认值 2000

例　程

```
BASE 0

VL=2000 '设置軸 0 的初速度为 2000 个脉冲当量/s
```

2.5.2 VH

所　属：属性

语　法：VH = value

类　型：DOUBLE

描　述：设置/读取轴的最大运行速度，VH 的单位为脉冲当量/s

范　围：（VL, MAXVEL），默认值 8000

例　程

```
BASE 0

VH=2000 '设置軸 0 的运行速度为 2000 个脉冲当量/s
```

2.5.3 ACC

所　属：属性

语　法：ACC = value

类　型：DOUBLE

描　述：设置/读取軸的加速度，ACC 的单位为脉冲当量/s^2

范　围：（0, MAXACC），默认值 10000

例　程

```
BASE 0

ACC=20000 '设置軸 0 的加速度为 20000 个脉冲当量/s^2
```

2.5.4 DEC

所　属：属性

语　法：DEC = value

类　型：DOUBLE

描　述：设置/读取軸的减速度，DEC 的单位为脉冲当量/s^2

范　围：（0, MAXDEC），默认值 10000

例　程

```
BASE 0
DEC=20000 '设置軸 0 的减速度为 20000 个脉冲当量/s^2
```

2.5.5 JK

所　属：属性

语　法：JK = value

类　型：ULONG

描　述：设置/读取单轴点位运动的速度曲线类型

范　围：【0,1】，0：T 型曲线；1：S 型曲线，默认值 0

例　程

```
BASE 0
JK=1        '设置軸 0 单轴点位运动的速度曲线类型为 S 型曲线
```

2.5.6 DSPEED

所　属：属性（只读）

语　法：value = DSPEED

类　型：DOUBLE

描　述：根据当前 BASE 列表中的轴，读取轴当前指令理论速度

例　程

```
BASE 0
DIM A AS DOUBLE
A=DSPEED     '将轴 0 的当前指令速度赋值给变量 A
```

2.5.7 STATE

所　属：属性(只读)

语　法：value = STATE

类　型：ULONG

描　述：读取当前轴运动状态。

返回值：如下

0：轴不可用状态

1：Ready 状态

2：轴停止状态，但未 Ready

3：轴处于错误状态，轴被停止运动

4：轴正在执行回原点运动中

5：轴正在执行单轴点位运动中

6：轴正在执行单轴连续运动中

7：轴正在参与插补运动中或同步运动中

8：轴处于外部 JOG 模式中

9：轴处于外部 MPG 模式中

例　程

```
BASE 0
DIM A AS ULONG
A=STATE     '将轴 0 的当前轴运动状态对应的值赋值给变量 A
A=STATE(AX(2))     '将轴 2 的当前轴运动状态对应的值赋值给变量 A。A=STATE AX(2)这种语法不对。
```

2.5.8 MOVE

所　属：命令

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

语　法 2：MOVE AX(axis no)，distance

描　述：BASE 轴列表的轴 ,以当前位置为原点 ,在相对坐标下运动到指定距离的位置。

参　数：distance　相对移动距离；类型：DOUBLE

axis no　轴号；　范围：根据控制器实际硬件决定。

例　程

```
'选择要操作的軸，並設定速度等相关参数
BASE 0,1,2
VL=2000
VH=8000
ACC=10000
DEC=10000
'开始轴 0 的相对运动
BASE 0
```

```
MOVE 1000
WAIT DONE
'开始轴 2,3 的相对运动
BASE 1,2
MOVE 2000, 3000
WAIT DONE
'指定轴 1，开始相对运动
MOVE AX(1),5000
```

2.5.9 MOVEABS

所　属：命令

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

语　法 2：MOVEABS AX(axis no)，positon

描　述： BASE 轴列表的轴，在绝对坐标下运动到的指定位置。

参　数：position　绝对位置；类型：DOUBLE

axis no　轴号； 范围：根据控制器实际硬件决定。

例　程

```
'选择要操作的軸，並設定速度等相关参数
BASE 0,1,2
VL=2000
VH=8000
ACC=10000
DEC=10000
'开始轴 0 的绝对运动
BASE 0
MOVEABS 1000
WAIT DONE
'开始轴 2,3 的绝对运动
BASE 1,2
MOVEABS 2000, 3000
WAIT DONE
'指定轴 1，开始绝对运动
```

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

2.5.10 PCHANGE

所　属：命令

语　法 1：PCHANGE distance

语　法 2：PCHANGE AX(axis no) ,distance

描　述： 修改当前运动的终点位置，如果当前轴不在运动中，下该指令会报错。

参　数：distance　改变相对运动距离；类型：DOUBLE

　　axis no　轴号； 范围：根据控制器实际硬件决定。

例　程

```
BASE 0
MOVE 20000
SLEEP 200
PCHANGE 25000          '改变位移为 25000
WAIT DONE
BASE 0,1
SVON
MOVE 10000,10000
SLEEP 200
PCHANGE AX(1),5000   '改变位移为 5000
WAIT DONE
```

2.5.11 STOPDEC

所　属：命令

语　法 1：STOPDEC

语　法 2：STOPDEC　AX(axis no)，dec

语　法 3：STOPDEC　dec1，dec2，dec3，……，dec32

描　述： BASE 轴列表中的轴或指定轴、指定减速度对轴下减速停止运动命令

参　数：dec　　减速度；类型：DOUBLE

　　axis no　轴号； 范围：根据控制器实际硬件决定。

例　程

```
BASE 0,1,2
SVON
VL=1000
VH=10000
ACC=50000
DEC=50000
'多个轴下减速停止运动命令
MOVE 40000,20000,35000
SLEEP 1000
STOPDEC
WAIT DONE
'指定轴下减速停止运动命令
FORWARD AX(0)
SLEEP 2000
STOPDEC AX(0)
WAIT AX(0),DONE
'指定多个轴用新的减速度下减速停止运动命令
MOVE 40000,30000,35000
SLEEP 2000
STOPDEC 200000,200000,200000
```

2.5.12 STOPEMG

所　属：命令

语　法 1：STOPEMG

语　法 2：STOPEMG　AX(axis no)

描　述：　BASE 轴列表中的轴或指定轴对轴下立即停止运动命令

参　数：　axis no　　轴号；　范围：根据控制器实际硬件决定。

例　程

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

```
VL=1000
VH=10000
ACC=50000
DEC=50000
'多个轴下立即停止运动命令
MOVE 40000,20000,35000
SLEEP 1000
STOPEMG
WAIT DONE
'指定轴下立即停止运动命令
FORWARD AX(0)
SLEEP 2000
STOPEMG AX(0)
WAIT AX(0),DONE
```

2.5.13 BACKLASH_EN

所　属：属性

语　法：BACKLASH_EN = value

类　型：ULONG

描　述：启用/禁用背隙补偿功能

范　围：如下设定值，默认值 0

0：禁用

1：启用

例　程

```
BASE 0
BACKLASH_EN=1   '启用轴 0 的背隙补偿功能
```

2.5.14 BACKLASH_PULSE

所　属：属性

语　法：BACKLASH_PULSE = value

类　型：ULONG

描　述：设置/读取背隙补偿的脉冲个数

范　围：【0, 4095】，默认值 10

例　程

```
BASE 0

BACKLASH_PULSE=10 '设置轴 0 的背隙补偿的脉冲个数为 10 个
```

2.5.15 BACKLASH_VEL

所　属：属性

语　法：BACKLASH_VEL = value

类　型：ULONG

描　述：设置/读取进行补偿距离移动时的速度，单位为脉冲/s

范　围：（0, MAXVEL），默认值 1000

例　程

```
BASE 0

BACKLASH_VEL=1000 '设置轴 0 的背隙补偿速度为 1000 个脉冲/s
```

2.5.16 MAXVEL

所　属：属性

语　法：MAXVEL = value

类　型：DOUBLE

描　述：设定/读取运动轴的运行速度限值，单位为脉冲当量/s 。VL、VH、HOME_VH 等跟轴速度相关的设定值都不能超过该限值，否则设定会不成功。

范　围：【1,5000000】，默认值 1000000

例　程

```
MAXVEL=200000 '设置轴的最大运行速度限值为 200000 脉冲当量/s
```

2.5.17 MAXACC

所　属：属性

语　法：MAXACC = value

类　型：DOUBLE

描　述：设定/读取运动轴的加速度限值，单位为脉冲当量/s^2 。ACC、HOME_ACC 等跟轴加速度相关的设定值都不能超过该限值，否则设定会不成功。

范　围：【1,500000000】，默认值 500000000

例　程

```
MAXACC=200000 '设置軸的加速度限值为 200000 脉冲当量/s^2
```

2.5.18 MAXDEC

所　属：属性

语　法：MAXDEC = value

类　型：DOUBLE

描　述：设定/读取运动轴的减速度限值，单位为脉冲当量/s^2 。DEC、HOME_DEC 等跟轴减速度相关的设定值都不能超过该限值，否则设定会不成功。

范　围：【1,500000000】，默认值 500000000

例　程

```
MAXDEC=200000 '设置軸的减速度限值为 200000 脉冲当量/s^2
```

2.5.19 RESETERR

所　属：命令

语　法 1：RESETERR

语　法 2：RESETERR AX(axis no)

描　述：BASE 轴列表的轴或指定轴，清除轴错误

参　数：axis no　 轴号； 范围：根据控制器实际硬件决定。

例　程

```
BASE 0,1,2
RESETERR    '清除轴 0、1、2 的错误
RESETERR AX(1)   '清除轴 1 的错误
```

2.6 单轴定速运动

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.6.1	FORWARD	正向恒速连续运动	√	×
2.6.2	REVERSE	反向恒速连续运动	√	×
2.6.3	VCHANGE	运动中改变速度	√	×
2.6.4	VCHANGE_RATE	运动中改变速度（百分比）	√	×

2.6.1 FORWARD

所　属：命令

语　法 1：FORWARD

语　法 2：FORWARD AX(axis no)

语　法 3：FORWARD dir1[，dir2][，dir3]……　　dir 为 0 时，方向与 FORWARD 同向 dir 为 1 时，方向与 FORWARD 反向

描　述：　BASE 轴列表的轴或指定轴，开始正向连续运动

参　数：dir　方向；类型：ULONG

　　axis no　　轴号；　范围：根据控制器实际硬件决定。

例　程

```
BASE 0,1
SVON
VL=1000                 '设置初速度
VH=10000                '设置运行速度
ACC=100000              '设置加速度
DEC=ACC                 '设置减速度
'单轴或多轴同方向执行连续运动
FORWARD       '轴 0,1 都执行正向连续运动
SLEEP 2000       '延时 2000ms
STOPDEC          '减速停止轴 0,1 运动
```

```
WAIT DONE     '等待运动停止,如运动未停止的状态，下面语句再对该轴操作会执行不成功
REVERSE       '轴 0,1 都执行负向连续运动
SLEEP 2000
STOPDEC       '减速停止轴 0,1 运动
WAIT DONE     '等待运动停止
'指定多个轴按不同方向执行连续运动
FORWARD 0,1   '轴 0 正向连续运动，轴 1 负向连续运动
SLEEP 2000
STOPEMG       '立即停止轴 0,1 运动
WAIT DONE
'指定一个轴执行连续运动
FORWARD AX(0)   '轴 0 执行正向连续运动
SLEEP 1000
STOPDEC AX(0)   '指定轴 0 下减速停止命令
WAIT AX(0),DONE   '指定轴等待运动停止
```

2.6.2 REVERSE

所　属：命令

语　法 1：REVERSE

语　法 2：REVERSE　AX(axis no)

语　法 3：REVERSE　dir1[，dir2][，dir3]……　dir 为 0 时，方向与 REVERSE 同向
dir 为 1 时，方向与 REVERSE 反向

描　述：　BASE 轴列表的轴或指定轴，开始反向连续运动

参　数：dir　方向；类型：ULONG
axis no　轴号；　范围：根据控制器实际硬件决定。

例　程

```
BASE 0,1
SVON
VL=1000          '设置初速度
VH=10000         '设置运行速度
```

```
ACC=100000              '设置加速度
DEC=ACC                 '设置减速度
'单轴或多轴同方向执行连续运动
FORWARD      '轴 0,1 都执行正向连续运动
SLEEP 2000     '延时 2000ms
STOPDEC      '减速停止轴 0,1 运动
WAIT DONE     '等待运动停止,如运动未停止的状态，下面语句再对该轴操作会执行不成功
REVERSE      '轴 0,1 都执行负向连续运动
SLEEP 2000
STOPDEC      '减速停止轴 0,1 运动
WAIT DONE     '等待运动停止
'指定多个轴按不同方向执行连续运动
REVERSE 0,1     '轴 0 负向连续运动，轴 1 正向连续运动
SLEEP 2000
STOPEMG      '立即停止轴 0,1 运动
WAIT DONE
'指定一个轴执行连续运动
REVERSE AX(0)     '轴 0 执行负向连续运动
SLEEP 1000
STOPDEC AX(0)     '指定轴 0 下减速停止命令
WAIT AX(0),DONE     '指定轴等待运动停止
```

2.6.3 VCHANGE

所　属：命令

语　法 1：VCHANGE vel

语　法 2：VCHANGE AX（axis no），vel

语　法 3：VCHANGE vel, acc, dec

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

描　述： BASE 轴列表的第一个轴，开始更改速度运动；或指定轴和新的速度开始更改速度运动。该指令可以对 MOVE、MOVEABS、FORWARD、REVERSE 起作用，如果当前轴不在运动中，下该指令会报错

参　数： vel　　运行速度；类型：DOUBLE

acc　　改变速度时的加速度；类型：DOUBLE

dec　　改变速度时的减速度；类型：DOUBLE

axis no　轴号；　范围：根据控制器实际硬件决定

例　程

```
BASE 0,1
SVON
VL=1000              '设置初速度
VH=10000             '设置运行速度
ACC=100000           '设置加速度
DEC=ACC              '设置减速度
FORWARD      '轴 0,1 都执行正向连续运动
SLEEP 2000     '延时 2000ms
VCHANGE 30000        '对 BASE 列表中第一个轴起作用，将轴 0 的速度改为 30000
SLEEP 2000
VCHANGE AX(1),5000   '将轴 1 的速度改为 50000
SLEEP 2000
VCHANGE 20000,10000,10000   '将轴 0 的速度改为 20000，加、减速度都改为 10000
SLEEP 3000
VCHANGE AX(1),20000,50000,50000 '将轴 1 的速度改为 20000，加、减速度都改为 50000
SLEEP 2000
STOPDEC
```

2.6.4 VCHANGE_RATE

所　属：命令

语　法 1：VCHANGE_RATE　rate

语　法 2：VCHANGE_RATE　AX（axis no），rate

语　法 3：VCHANGE_RATE　rate, acc, dec

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

描　述： BASE 轴列表的第一个轴，开始按百分比更改速度运动；或指定轴和新的百分比速度开始更改速度运动。该指令可以对 MOVE、MOVEABS、FORWARD、REVERSE 起作用，如果当前轴不在运动中，下该指令会报错

参　数： rate　　原设置速度的百分比速度；类型：DOUBLE

acc　　改变速度时的加速度；类型：DOUBLE

dec　　改变速度时的减速度；类型：DOUBLE

axis no　轴号；　范围：根据控制器实际硬件决定

例　程

```
BASE 0,1
SVON
VL=1000              '设置初速度
VH=10000             '设置运行速度
ACC=100000           '设置加速度
DEC=ACC              '设置减速度
FORWARD              '轴 0,1 都执行正向连续运动
SLEEP 2000           '延时 2000ms
'运动中改变速度的功能应用中：新设定的速度要高于 VL
VCHANGE_RATE 200     '对 BASE 列表中第一个轴起作用，将轴 0 的速度改为设定的 VH 的 200%
SLEEP 2000
VCHANGE_RATE AX(1),30   '将轴 1 的速度改为设定的 VH 的 30%
SLEEP 2000
VCHANGE_RATE 50,10000,10000   '将轴 0 的速度改为 VH 的 50%，加、减速度都改为 10000
SLEEP 3000
VCHANGE_RATE AX(1),400,50000,50000 '将轴 1 的速度改为 VH 的 400%，加、减速度都改为 50000
SLEEP 2000
STOPDEC
```

2.7 多轴插补运动

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.7.1	GVL	插补初速度	×	×
2.7.2	GVH	插补运行速度	×	×
2.7.3	GACC	插补加速度	×	×
2.7.4	GDEC	插补减速度	×	×
2.7.5	GJK	插补速度曲线类型	×	×
2.7.6	GDSPEED	当前插补运动指令速度值	×	×
2.7.7	GSTATE	当前插补运动状态	×	×
2.7.8	LINE	2-3 轴直线相对插补运动	×	×
2.7.9	LINEABS	2-3 轴直线绝对插补运动	×	×
2.7.10	DIRECT	2-8 轴线性相对插补运动	×	×
2.7.11	DIRECTABS	2-8 轴线性绝对插补运动	×	×
2.7.12	CIRC	2 轴相对圆弧插补（指定圆心、终点）	×	×
2.7.13	CIRCABS	2 轴绝对圆弧插补（指定圆心、终点）	×	×
2.7.14	CIRC_3P	2 轴相对圆弧插补（指定圆上 3 点）	×	×
2.7.15	CIRCABS_3P	2 轴绝对圆弧插补（指定圆上 3 点）	×	×
2.7.16	CIRC_A	2 轴相对圆弧插补（指定圆弧角度、终点）	×	×
2.7.17	CIRCABS_A	2 轴绝对圆弧插补（指定圆弧角度、终点）	×	×
2.7.18	HELIX	3 轴相对螺旋插补（指定圆弧中心、终点、高度）	×	×
2.7.19	HELIXABS	3 轴绝对螺旋插补（指定圆弧中心、终点、高度）	×	×
2.7.20	HELIX_3P	3 轴相对螺旋插补（指定螺旋线上	×	×

		3 点）		
2.7.21	HELIXABS_3P	3 轴绝对螺旋插补（指定螺旋线上 3 点）	×	×
2.7.22	HELIX_A	3 轴相对螺旋插补(指定圆弧角度、终点、高度）	×	×
2.7.23	HELIXABS_A	3 轴绝对螺旋插补(指定圆弧角度、终点、高度）	×	×
2.7.24	GPAUSE	插补运动暂停指令	×	×
2.7.25	GRESUME	插补运动暂定后恢复运动指令	×	×

2.7.1 GVL

所　属：属性

语　法：GVL = value

类　型：DOUBLE

描　述：设置/读取插补运动的初速度，GVL 的单位为脉冲当量/s

范　围：（0, MAXVEL），默认值 2000

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
GVL=2000 '设置轴 0,1 的插补运动的初速度为 2000 个脉冲当量/s
```

2.7.2 GVH

所　属：属性

语　法：GVH = value

类　型：DOUBLE

描　述：设置/读取插补运动的最大运行速度，GVH 的单位为脉冲当量/s

范　围：（GVL, MAXVEL），默认值 8000

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
GVH=10000 '设置轴 0,1 的插补运动的最大运行速度为 10000 个脉冲当量/s
```

2.7.3 GACC

所　属：属性

语　法：GACC = value

类　型：DOUBLE

描　述：设置/读取插补运动的加速度，GACC 的单位为脉冲当量/s^2

范　围：（0, MAXACC），默认值 10000

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
GACC=20000 '设置軸 0,1 的插补运动的加速度为 20000 个脉冲当量/s^2
```

2.7.4 GDEC

所　属：属性

语　法：GDEC = value

类　型：DOUBLE

描　述：设置/读取插补运动的减速度，GDEC 的单位为脉冲当量/s^2

范　围：（0, MAXDEC），默认值 10000

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
GDEC=20000 '设置軸 0,1 的插补运动的减速度为 20000 个脉冲当量/s^2
```

2.7.5 GJK

所　属：属性

语　法：GJK = value

类　型：ULONG

描　述：设置/读取插补运动的速度曲线类型

范　围：【0,1】，0：T 型曲线；1：S 型曲线，默认值 0

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
GJK=1     '设置轴 0、1 的插补运动的速度曲线类型为 S 型曲线
```

2.7.6 GDSPEED

所　属：属性（只读）

语　法：value = GDSPEED

类　型：DOUBLE

描　述：读取当前插补指令理论速度

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
DIM A AS DOUBLE
A=GDSPEED     '将轴 0,1 插补运动的当前指令速度赋值给变量 A
```

2.7.7 GSTATE

所　属：属性（只读）

语　法：value = GSTATE

类　型：ULONG

描　述：读取当前插补运动状态。

返回值：如下

0：插补不可用状态

1：插补运动处于 Ready 状态

2：插补运动处于停止状态，但未 Ready

3：插补运动处于错误状态，插补运动被停止运动

4：BASE 轴正在执行插补运动中

5：保留

6：BASE 轴正在执行连续插补运动中

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

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

```
A=GSTATE    '将轴 0、1 当前的插补运动状态对应的值赋值给变量 A
```

2.7.8 LINE

所　属：命令

语　法：LINE　distance1,distance2 [,distance3]

描　述：指定插补轴的移动距离，开始 2 轴或 3 轴的相对直线插补运动。LINE 指令仅支持 2 轴或 3 轴的直线插补运动，3 轴以上不支持。

参　数：distance　各轴的相对移动距离；类型：DOUBLE

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
SVON
GVL=1000                '设置插补初速度
GVH=10000               '设置插补运行速度
GACC=100000             '设置插补加速度
GDEC=GACC               '设置插补减速度
GJK=0                   '设置插补速度曲线为 T 型
 '绝对直线插补运动
LINEABS 0,5000   '运动到目标位置（0,5000）
WAIT DONE    '等待 LINEABS 运动走完
 '相对直线插补运动
LINE 8000,-15000    '轴 0、1 方向的运动距离分别为 8000、-15000
```

2.7.9 LINEABS

所　属：命令

语　法：LINEABS　position1, position2[,position3]

描　述：指定插补轴的终点，开始 2 轴或 3 轴的绝对直线插补运动。LINEABS 指令仅支持 2 轴或 3 轴的直线插补运动，3 轴以上不支持。

参　数：position　各轴的终点位置；类型：DOUBLE

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
SVON
GVL=1000                '设置插补初速度
GVH=10000               '设置插补运行速度
GACC=100000             '设置插补加速度
GDEC=GACC               '设置插补减速度
GJK=0                   '设置插补速度曲线为 T 型
 '绝对直线插补运动
LINEABS 0,5000   '运动到目标位置（0,5000）
WAIT DONE     '等待 LINEABS 运动走完
 '相对直线插补运动
LINE 8000,-15000     '轴 0、1 方向的运动距离分别为 8000、-15000
```

2.7.10 DIRECT

所　属：命令

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

描　述：指定插补轴的移动距离，开始 2 轴-8 轴的相对线性插补运动。最多支持到 8 个轴的 DIRECT 线性插补运动

参　数：distance　各轴的相对移动距离；类型：DOUBLE

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1,2,3
SVON
'DIRECT 插补运动中，以下速度、加减速参数设置的是参与插补运动中移动距离最长轴的参数
GVL=1000                '设置插补初速度
GVH=10000               '设置插补运行速度
GACC=100000             '设置插补加速度
GDEC=GACC               '设置插补减速度
GJK=0                   '设置插补速度曲线为 T 型
'绝对线性插补
```

```
DIRECTABS 0,5000,-500,1000     '运动到目标位置（0,5000,-500,1000）
WAIT DONE                      '等待 DIRECTABS 运动走完
'相对线性插补
DIRECT 8000,-15000,0,2000      '轴 0、1、2、3 方向的运动距离分别为 8000、-15000、0、2000
```

2.7.11 DIRECTABS

所　属：命令

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

描　述：指定插补轴的终点，开始 2 轴-8 轴的绝对线性插补运动。最多支持到 8 个轴的 DIRECTABS 线性插补运动

参　数：position　各轴的终点位置；类型：DOUBLE

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1,2,3
SVON
'DIRECT 插补运动中，以下速度、加减速参数设置的是参与插补运动中移动距离最长轴的参数
GVL=1000               '设置插补初速度
GVH=10000              '设置插补运行速度
GACC=100000            '设置插补加速度
GDEC=GACC              '设置插补减速度
GJK=0                  '设置插补速度曲线为 T 型
'绝对线性插补
DIRECTABS 0,5000,-500,1000     '运动到目标位置（0,5000,-500,1000）
WAIT DONE                      '等待 DIRECTABS 运动走完
'相对线性插补
DIRECT 8000,-15000,0,2000      '轴 0、1、2、3 方向的运动距离分别为 8000、-15000、0、2000
```

2.7.12 CIRC

所　属：命令

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

描　述：指定圆方向、圆心、终点，开始两轴的相对圆弧插补运动

参　数：dir　圆弧运动方向：0-顺时针；1-逆时针　；类型：ULONG

center1　第一个轴圆心的相对坐标　；类型：DOUBLE

center2　第二个轴圆心的相对坐标；类型：DOUBLE

end1　第一个轴圆弧终点的相对坐标；类型：DOUBLE

end2　第二个轴圆弧终点的相对坐标；类型：DOUBLE

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
GVL=1000            '设置插补初速度
GVH=10000           '设置插补运行速度
GACC=100000     '设置插补加速度
GDEC=GACC           '设置插补减速度
GJK=0                   '设置插补速度曲线为 T 型
DPOS=0              '当前理论位置清零
SVON
'执行相对圆弧插补运动
CIRC 0,5000,0,10000,0       '顺时针，圆心相对距离为（5000,0）,圆弧终点相对距离为（10000,0）
WAIT DONE
'执行绝对圆弧插补运动
CIRCABS 1,5000,0,0,0        '逆时针，圆心位置为（5000,0）,圆弧终点位置为（0,0）
WAIT DONE
```

2.7.13 CIRCABS

所　属：命令

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

描　述：指定圆方向、圆心、终点，开始两轴的绝对圆弧插补运动

参　数：dir 圆弧运动方向：0-顺时针；1-逆时针　；类型：ULONG

center1　第一个轴圆心的绝对坐标　；类型：DOUBLE

center2　第二个轴圆心的绝对坐标；类型：DOUBLE

end1　第一个轴圆弧终点的绝对坐标；类型：DOUBLE

end2　　第二个轴圆弧终点的绝对坐标；类型：DOUBLE

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
GVL=1000              '设置插补初速度
GVH=10000             '设置插补运行速度
GACC=100000     '设置插补加速度
GDEC=GACC             '设置插补减速度
GJK=0                 '设置插补速度曲线为 T 型
DPOS=0                '当前理论位置清零
SVON
'执行相对圆弧插补运动
CIRC 0,5000,0,10000,0       '顺时针，圆心相对距离为（5000,0）,圆弧终点相对距离为（10000,0）
WAIT DONE
'执行绝对圆弧插补运动
CIRCABS 1,5000,0,0,0        '逆时针，圆心位置为（5000,0）,圆弧终点位置为（0,0）
WAIT DONE
```

2.7.14 CIRC_3P

所　属：命令

语　法：CIRC_3P　dir,ref1,ref2,end1,end2

描　述：指定圆方向和圆上 3 点，开始两轴的相对圆弧插补运动

参　数：dir　圆弧运动方向：0-顺时针；1-逆时针　；类型：ULONG

ref1　　第一个轴中间点的相对坐标　；类型：DOUBLE

ref2　　第二个轴中间点的相对坐标；类型：DOUBLE

end1　第一个轴圆弧终点的相对坐标；类型：DOUBLE

end2　第二个轴圆弧终点的相对坐标；类型：DOUBLE

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
```

```
GVL=1000            '设置插补初速度
GVH=10000           '设置插补运行速度
GACC=100000         '设置插补加速度
GDEC=GACC           '设置插补减速度
GJK=0               '设置插补速度曲线为 T 型
DPOS=0              '当前理论位置清零
SVON
'执行相对圆弧插补运动
CIRC_3P  0,10000,10000,20000,0      '顺时针，圆上第二点相对距离为（10000,10000）,圆弧终点相对距离为（20000,0）
WAIT DONE
'执行绝对圆弧插补运动
CIRCABS_3P 1,10000,10000,0,0        '逆时针 ,圆心第二点位置为( 10000,10000 ),圆弧终点位置为( 0,0 )
WAIT DONE
```

2.7.15 CIRCABS_3P

所　属：命令

语　法：CIRC_3P　dir,ref1,ref2,end1,end2

描　述：指定圆方向和圆上 3 点，开始两轴的绝对圆弧插补运动

参　数：dir 圆弧运动方向：0-顺时针；1-逆时针　；类型：ULONG

ref1　第一个轴中间点的绝对坐标　；类型：DOUBLE

ref2　第二个轴中间点的绝对坐标；类型：DOUBLE

end1　第一个轴圆弧终点的绝对坐标；类型：DOUBLE

end2　第二个轴圆弧终点的绝对坐标；类型：DOUBLE

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
GVL=1000            '设置插补初速度
GVH=10000           '设置插补运行速度
GACC=100000         '设置插补加速度
```

```
GDEC=GACC          '设置插补减速度
GJK=0              '设置插补速度曲线为 T 型
DPOS=0             '当前理论位置清零
SVON
'执行相对圆弧插补运动
CIRC_3P  0,10000,10000,20000,0      '顺时针，圆上第二点相对距离为（10000,10000）,圆弧终点相对距离为（20000,0）
WAIT DONE
'执行绝对圆弧插补运动
CIRCABS_3P 1,10000,10000,0,0        '逆时针 ,圆心第二点位置为( 10000,10000 ),圆弧终点位置为( 0,0 )
WAIT DONE
```

2.7.16 CIRC_A

所　属：命令

语　法：CIRC_A　dir,center1,center2,degree

描　述：指定圆方向、圆心、角度，开始两轴的相对圆弧插补运动

参　数：dir 圆弧运动方向：0-顺时针；1-逆时针 ；类型：ULONG

center1　第一个轴圆心的相对坐标 ；类型：DOUBLE

center2　第二个轴圆心的相对坐标；类型：DOUBLE

degree　圆弧角度，单位为角度；类型：DOUBLE

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
GVL=1000           '设置插补初速度
GVH=10000          '设置插补运行速度
GACC=100000    '设置插补加速度
GDEC=GACC          '设置插补减速度
GJK=0              '设置插补速度曲线为 T 型
DPOS=0             '当前理论位置清零
SVON
```

```
'执行相对圆弧插补运动
CIRC_A 0,10000,0,180      '顺时针，圆心相对距离为（10000,0）,圆弧角度为 180 度
WAIT DONE
'执行绝对圆弧插补运动
CIRCABS_A 1,10000,0,90      '逆时针，圆心位置为（10000,0）,圆弧角度为 90 度
WAIT DONE
```

2.7.17 CIRCABS_A

所　属：命令

语　法：CIRC_A　dir,center1,center2,degree

描　述：指定圆方向、圆心、角度，开始两轴的绝对圆弧插补运动

参　数：dir 圆弧运动方向：0-顺时针；1-逆时针 ；类型：ULONG

center1　第一个轴圆心的绝对坐标 ；类型：DOUBLE

center2　第二个轴圆心的绝对坐标；类型：DOUBLE

degree　圆弧角度，单位为角度；类型：DOUBLE

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
GVL=1000          '设置插补初速度
GVH=10000         '设置插补运行速度
GACC=100000     '设置插补加速度
GDEC=GACC         '设置插补减速度
GJK=0             '设置插补速度曲线为 T 型
DPOS=0            '当前理论位置清零
SVON
'执行相对圆弧插补运动
CIRC_A 0,10000,0,180      '顺时针，圆心相对距离为（10000,0）,圆弧角度为 180 度
WAIT DONE
'执行绝对圆弧插补运动
CIRCABS_A 1,10000,0,90      '逆时针，圆心位置为（10000,0）,圆弧角度为 90 度
```

WAIT DONE

2.7.18 HELIX

所　属：命令

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

描　述：指定螺旋旋转方向、中心、螺旋高度，开始 3 轴的相对螺旋插补运动

参　数：dir　螺旋运动方向：0：　顺时针；1：逆时针

center1　第一个轴圆心的相对坐标　；

center2　第二个轴圆心的相对坐标；

end1　第一个轴螺旋终点的相对坐标；

end2　第二个轴螺旋终点的相对坐标；

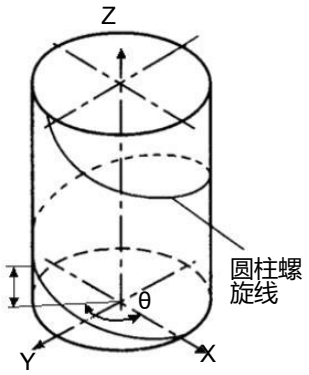
distance　螺旋高度，单位为脉冲当量

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1,2
GVL=1000            '设置插补初速度
GVH=10000           '设置插补运行速度
GACC=100000         '设置插补加速度
GDEC=GACC           '设置插补减速度
GJK=0               '设置插补速度曲线为 T 型
DPOS=0              '当前理论位置清零
MPOS=0              '当前反馈位置清零
SVON
'相对螺旋插补，顺时针，圆心相对坐标为（5000,0）,圆弧终点相对坐标为（10000,0）,螺旋高度为 5000
HELIX 0,5000,0,10000,0,5000
WAIT DONE

'绝对螺旋插补，逆时针，圆心绝对坐标为（5000,0）,圆弧终点绝对坐标为（0,0）,螺旋 Z 轴终点位置为 0
HELIXABS 1,5000,0,0,0,0
```



```
WAIT DONE
```

2.7.19 HELIXABS

所　属：命令

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

描　述：指定螺旋方向、中心、终点，开始 3 轴的绝对螺旋插补运动

参　数：dir　螺旋运动方向：0：　顺时针；1：逆时针

center1　第一个轴圆心的绝对坐标

center2　第二个轴圆心的绝对坐标

end1　第一个轴螺旋终点的绝对坐标

end2　第二个轴螺旋终点的绝对坐标

position　第三个轴终点位置坐标

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1,2
GVL=1000            '设置插补初速度
GVH=10000           '设置插补运行速度
GACC=100000         '设置插补加速度
GDEC=GACC           '设置插补减速度
GJK=0               '设置插补速度曲线为 T 型
DPOS=0              '当前理论位置清零
MPOS=0              '当前反馈位置清零
SVON
'相对螺旋插补，顺时针，圆心相对坐标为（5000,0）,圆弧终点相对坐标为（10000,0）,螺旋高度为 5000
HELIX 0,5000,0,10000,0,5000
WAIT DONE
'绝对螺旋插补，逆时针，圆心绝对坐标为（5000,0）,圆弧终点位置为（0,0）,螺旋 Z 轴终点位置为 0
HELIXABS 1,5000,0,0,0,0
WAIT DONE
```

2.7.20 HELIX_3P

所　属：命令

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

描　述：指定螺旋方向和螺旋线上的 3 点，开始 3 轴的相对螺旋插补运动

参　数：　dir　螺旋运动方向：0：　顺时针；1：逆时针

ref1　第一个轴中间点的相对坐标　；

ref2　第二个轴中间点的相对坐标；

ref3　第三个轴中间点的相对坐标；

end1　第一个轴螺旋终点的相对坐标；

end2　第二个轴螺旋终点的相对坐标；

end3　第三个轴螺旋终点的相对坐标；

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1,2
GVL=1000            '设置插补初速度
GVH=10000           '设置插补运行速度
GACC=100000         '设置插补加速度
GDEC=GACC           '设置插补减速度
GJK=0               '设置插补速度曲线为 T 型
DPOS=0              '当前理论位置清零
MPOS=0              '当前反馈位置清零
SVON
'相对螺旋插补。顺时针，中间参考点相对坐标为（0,5000,0），终点相对坐标为（10000,0,5000）
HELIX_3P 0,0,5000,0,10000,0,5000
WAIT DONE
'绝对螺旋插补。逆时针，中间参考点绝对坐标为（0,5000,2500），终点绝对坐标为（0,0,0）
HELIXABS_3P 1,0,5000,2500,0,0,0
WAIT DONE
```

2.7.21 HELIXABS_3P

所　属：命令

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

描　述：指定螺旋方向和螺旋线上的 3 点，开始 3 轴的绝对螺旋插补运动

参　数：　dir　螺旋运动方向：0：　顺时针；1：逆时针

ref1　第一个轴中间点的绝对坐标　；

ref2　第二个轴中间点的绝对坐标；

ref3　第三个轴中间点的绝对坐标；

end1　第一个轴螺旋终点的绝对坐标；

end2　第二个轴螺旋终点的绝对坐标；

end3　第三个轴螺旋终点的绝对坐标；

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1,2
GVL=1000            '设置插补初速度
GVH=10000           '设置插补运行速度
GACC=100000         '设置插补加速度
GDEC=GACC           '设置插补减速度
GJK=0               '设置插补速度曲线为 T 型
DPOS=0              '当前理论位置清零
MPOS=0              '当前反馈位置清零
SVON
'相对螺旋插补。顺时针，中间参考点相对坐标为（0,5000,0），终点相对坐标为（10000,0,5000）
HELIX_3P 0,0,5000,0,10000,0,5000
WAIT DONE
'绝对螺旋插补。逆时针，中间参考点绝对坐标为（0,5000,2500），终点绝对坐标为（0,0,0）
HELIXABS_3P 1,0,5000,2500,0,0,0
WAIT DONE
```

2.7.22 HELIX_A

所　属：命令

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

描　述：指定螺旋方向、螺旋圆面上的旋转角度、螺旋高度，开始 3 轴的相对螺旋插补运动

参　数：dir　螺旋运动方向：0：　顺时针；1：逆时针

center1　第一个轴圆心的相对坐标

center2　第二个轴圆心的相对坐标

degree　螺旋线形成的圆柱截面上的投影圆弧运动的圆心角度

distance　螺旋高度，单位为脉冲当量

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1,2
GVL=1000            '设置插补初速度
GVH=10000           '设置插补运行速度
GACC=100000         '设置插补加速度
GDEC=GACC           '设置插补减速度
GJK=0               '设置插补速度曲线为 T 型
DPOS=0              '当前理论位置清零
MPOS=0              '当前反馈位置清零
SVON
'相对螺旋插补，顺时针，圆心相对坐标为（5000,0）,运动的圆心角度为 180 度，,螺旋高度为 5000
HELIX_A 0,5000,0,180,5000
WAIT DONE
'绝对螺旋插补，逆时针，圆心绝对坐标为（5000,0）,运动的圆心角度为 180 度，,螺旋 Z 轴终点位置为 0
HELIXABS_A 1,5000,0,180,0
WAIT DONE
```

2.7.23 HELIXABS_A

所　属：命令

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

描　述：指定螺旋方向、螺旋圆面上的旋转角度、终点位置，开始 3 轴的绝对螺旋插补运动

参　数：dir　螺旋运动方向：0：　顺时针；1：逆时针

center1　第一个轴圆心的绝对坐标

center2　第二个轴圆心的绝对坐标

degree　螺旋线形成的圆柱截面上的投影圆弧运动的圆心角度

position　第三个轴终点位置坐标

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1,2
GVL=1000            '设置插补初速度
GVH=10000           '设置插补运行速度
GACC=100000         '设置插补加速度
GDEC=GACC           '设置插补减速度
GJK=0               '设置插补速度曲线为 T 型
DPOS=0              '当前理论位置清零
MPOS=0              '当前反馈位置清零
SVON
'相对螺旋插补，顺时针，圆心相对坐标为（5000,0）,运动的圆心角度为 180 度，,螺旋高度为 5000
HELIX_A 0,5000,0,180,5000
WAIT DONE
'绝对螺旋插补，逆时针，圆心绝对坐标为（5000,0）,运动的圆心角度为 180 度，,螺旋 Z 轴终点位置为 0
HELIXABS_A 1,5000,0,180,0
WAIT DONE
```

2.7.24 GPAUSE

所　属：命令

语　法：GPAUSE

描　述：暂定当前 TASK 的插补运动。该指令对插补运动和连续插补运动都起作用。

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

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

```
SLEEP 500
GPAUSE              '暂停插补运动
IF(DIN(1)=1) THEN
GRESUME             '恢复插补运动，继续执行未执行的插补运动
END IF
```

2.7.25 GRESUME

所　属：命令

语　法：GRESUME

描　述：恢复当前 TASK 的插补运动暂定状态。该指令对插补运动和连续插补运动都起作用。

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
LINE 11000,20000
SLEEP 500
GPAUSE              '暂停插补运动
IF(DIN(1)=1) THEN '如果 DIN(1)为 1 时，恢复插补运动
GRESUME             '恢复插补运动，继续执行未执行的插补运动
END IF
```

2.8 多轴连续插补运动

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.2.1	PATHBEGIN	开始进入连续插补状态	×	×
2.2.2	PATHEND	结束连续插补状态	×	×
2.2.3	MERGEON	用 fly mode 交接模式执行连续插补运动	×	×
2.2.4	MERGEOFF	用buffer mode交接模式执行连续插补运动	×	×
2.2.5	DELAY	连续插补路径中的延时段指令	×	×

2.8.1 PATHBEGIN

所　属：命令

语　法：PATHBEGIN [num]

描　述：指定路径缓存里添加多少段插补指令后，开始进入连续插补模式。Num 值不填或 0 时，会将 PATHBEGIN 与 PATHEND 间所有的段都添加到缓存区里，再开始执行连续插补运动。

参　数：num 预先添加到连续插补缓存区段数；类型：ULONG；范围【0,10000】

注　意：连续插补段不允许存在点位运动指令的段

该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

```
BASE 0,1
SVON
GVL=1000              '设置插补初速度
GVH=10000             '设置插补运行速度
GACC=100000           '设置插补加速度
GDEC=ACC              '设置插补减速度
LINEABS 0,0           '运动到目标位置（0,0）
```

```
WAIT DONE
'PATHBEGIN 开始到 PATHEND 之前的路径段为连续插补运动
'PATHBEGIN 后面跟的编号不写或写 0 时，表明路径添加完再开始执行连续插补，其他数值代表添加该数值段后开始执行运动
PATHBEGIN       '进入连续插补状态
MERGEON         '连续插补模式设置为 fly mode
CIRCABS 1,10000,0,10000,-10000    '圆弧插补段
LINEABS 25000,-10000              '直线插补段
DELAY=500                         '延时段，延时 500ms
CIRCABS 1,25000,0,35000,0         '圆弧插补段
CIRCABS 1,25000,0,25000,10000     '圆弧插补段
LINEABS 10000,10000               '直线插补段
CIRCABS 1,10000,0,0,0             '圆弧插补段
PATHEND         '连续插补路径段结束指令，退出连续状态
```

2.8.2 PATHEND

所　属：命令

语　法：PATHEND

描　述：当前插补运动退出连续插补状态

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

请参考 PATHBEGIN 指令中例程。

2.8.3 MERGEON

所　属：命令

语　法：MERGEON

描　述：使能 Blending，用 fly mode 交接模式执行连续插补运动

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

请参考 PATHBEGIN 指令中例程。

2.8.4 MERGEOFF

所　属：命令

语　法：MERGEOFF

描　述：禁用 Blending，用 buffer mode 交接模式执行连续插补运动

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

请参考 PATHBEGIN 指令中例程。

2.8.5 DELAY

所　属：命令

语　法：DELAY= time

描　述：连续插补中，延时段指令。表示 DELAY 指令上一段完成与 DELAY 指令下一段开始间延时的事件，该延时时间非常精准，单位为 ms。

注　意：该指令不能在 Motion Studio 中的“终端”和“观察变量”工具中使用

例　程

请参考 PATHBEGIN 指令中例程。

2.9 同步运动

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.9.1	STA_SRC	同步启/停触发源	√	√
2.9.2	SETSTA	以单轴相对运动模式状态等待同步启动信号	√	×
2.9.3	SETSTA_ABS	以单轴绝对运动模式状态等待同步启动信号	√	×
2.9.4	SETSTA_VEL	以恒速连续运动模式状态等待同步启动信号	√	×
2.9.5	STARTSTA	STA 或同张板卡同步轴开始启动运动	√	×
2.9.6	STOPSTA	STA 或同张板卡同步轴开始停止运动	√	×
2.9.7	GANTRY	龙门运动	√	×
2.9.8	GEAR	电子齿轮运动	√	×
2.9.9	PHASE	电子齿轮运动过程中超前或落后相位运动	√	×
2.9.10	TANGENT	切向跟随运动	√	×
2.9.11	MODULO	切向跟随运动中旋转刀轴旋转一周的指令脉冲数	√	√

2.9.1 STA_SRC

所　属：属性

语　法：STA_SRC=value

类　型：ULONG

描　述：设置/读取轴的同步启停运动的触发源，触发源被触发后，处于等待同步启/停的轴会根据等待的模式，开始执行相应的运动。STA 信号可以由运动控制卡上硬件 STA 针脚触发产生，也可以由 STARTSTA 指令触发产生。

范　围:设定值和返回值如下，默认值 1

0：禁用

1：板卡 STA 信号

2：轴 0 的比较信号

3：轴 1 的比较信号

4：轴 2 的比较信号

5：轴 3 的比较信号

6：轴 4 的比较信号

7：轴 5 的比较信号

8：轴 6 的比较信号

9：轴 7 的比较信号

10：轴 0 的停止信号

11：轴 1 的停止信号

12：轴 2 的停止信号

13：轴 3 的停止信号

14：轴 4 的停止信号

15：轴 5 的停止信号

16：轴 6 的停止信号

17：轴 7 的停止信号

例　程

```
BASE 1
STA_SRC = 1   '设置轴 1 的同步启停运动触发源为板卡 STA 信号
```

2.9.2 SETSTA

所　属：命令

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

描　述：设置同步轴为点位运动模式并处于等待触发状态，同时设置同步轴的相对移动距离。

参　数：distance　相对移动距离；类型：DOUBLE

例　程

```
BASE 0,1,2,3
STA_SRC=1    '设置同步源为板卡 STA 信号
SETSTA 10000,5000,30000,-8000 '设置轴 0,1,2,3 处于等待同步相对运动状态，并设置运动距离
STARTSTA      '同步启，同时启动 4 轴运动
WAIT DONE
SETSTA_ABS 0,2000,5000,20000   '设置轴 0,1,2,3 处于等待同步绝对运动状态，并设置目标位置
SLEEP 500
STARTSTA          '同步启，同时启动 4 轴运动
WAIT DONE
SETSTA_VEL 0,1,0,1 '设置轴 0,1,2,3 处于等待同步连续运动状态，并设置各轴方向
STARTSTA          '同步启，同时启动 4 轴运动
Sleep 3000
STOPSTA           '同步停，同时停止 4 轴运动
```

2.9.3 SETSTA_ABS

所　属：命令

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

描　述：设置同步轴为点位运动模式并处于等待触发状态，同时设置同步轴绝对移动位置

参　数：position　绝对位置；类型：DOUBLE

例　程

```
BASE 0,1,2,3
STA_SRC=1    '设置同步源为板卡 STA 信号
SETSTA 10000,5000,30000,-8000 '设置轴 0,1,2,3 处于等待同步相对运动状态，并设置运动距离
STARTSTA      '同步启，同时启动 4 轴运动
WAIT DONE
SETSTA_ABS 0,2000,5000,20000   '设置轴 0,1,2,3 处于等待同步绝对运动状态，并设置目标位置
SLEEP 500
```

```
STARTSTA          '同步启，同时启动 4 轴运动
WAIT DONE
SETSTA_VEL 0,1,0,1 '设置轴 0,1,2,3 处于等待同步连续运动状态，并设置各轴方向
STARTSTA          '同步启，同时启动 4 轴运动
Sleep 3000
STOPSTA           '同步停，同时停止 4 轴运动
```

2.9.4 SETSTA_VEL

所　属：命令

语　法：SETSTA_VEL dir1[,dir2][,dir3]……

描　述：设置同步轴为连续运动模式并处于等待触发状态，同时设置同步轴的连续运动方向。

参　数：dir　方向，0：正向，1：反向；类型：ULONG

例　程

```
BASE 0,1,2,3
STA_SRC=1    ''设置同步源为板卡 STA 信号
SETSTA 10000,5000,30000,-8000 '设置轴 0,1,2,3 处于等待同步相对运动状态，并设置运动距离
STARTSTA      '同步启，同时启动 4 轴运动
WAIT DONE
SETSTA_ABS 0,2000,5000,20000   '设置轴 0,1,2,3 处于等待同步绝对运动状态，并设置目标位置
SLEEP 500
STARTSTA          '同步启，同时启动 4 轴运动
WAIT DONE
SETSTA_VEL 0,1,0,1 '设置轴 0,1,2,3 处于等待同步连续运动状态，并设置各轴方向
STARTSTA          '同步启，同时启动 4 轴运动
Sleep 3000
STOPSTA           '同步停，同时停止 4 轴运动
```

2.9.5 STARTSTA

所　属：命令

语　法：STARTSTA

描　述：当轴的同步启/停触发源为 STA 信号时，使用该指令开始同步启动运动

例　程

请参考 SETSTA、SETSTA_ABS、SETSTA_VEL 等指令例程

2.9.6 STOPSTA

所　属：命令

语　法：STOPSTA

描　述：当轴的同步启/停触发源为 STA 信号时，使用该指令开始同步停止运动

例　程

请参考 SETSTA、SETSTA_ABS、SETSTA_VEL 等指令例程

2.9.7 GANTRY

所　属：命令

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

描　述：以 BASE 轴列表中的第一个轴为主轴，指定龙门运动从轴、参考源、龙门运动方向，建立龙门同步关系

参　数：slave axis no　从轴的轴号；范围：根据控制器实际硬件决定

refsrc　从轴跟随主轴的位置源；范围：0：理论位置，1：实际位置（暂不支持）

dir　从轴与主轴的运动方向一致性；范围：0：相同，1：相反

max_diffvalue　主轴和从轴读到的编码器位置误差值限值，该值可以不设定。如果设定值后，控制器会实时比对误差值，一旦超过 max_diffvalue 值，控制器会控制马达停止。

注　意：龙门一旦建立龙门关系，主从轴状态会变成同步状态，要解除龙门关系，需对从轴下 STOPDEC 或 STOPEMG 命令，龙门关系即解除。

例　程

```
BASE 0,1
SVON
```

```
'龙门轴的速度等参数由主轴参数决定
VL=1000
VH=40000
ACC=200000
DEC=200000
STOPDEC AX(1) '龙门运动中，对从轴下 STOPDEC 或 STOPEMG 可以解除龙门关系
'如果已有龙门关系存在，再下 GANTRY 命令，会报无效轴状态错误
GANTRY AX(1),0,0    '设定龙门关系：从轴为轴 1，参考源为主轴的理论位置，从轴方向与主轴同向
BASE 0              '设定轴 0 为龙门主轴
MOVE 20000    '主轴执行距离为 20000 的正向相对运动，从轴这时会与跟随主轴一起执行龙门运动
```

2.9.8 GEAR

所　属：命令

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

描　述：以 BASE 轴列表中的第一个轴为主轴，指定电子齿轮运动从轴、齿轮分子、齿轮分母、参考源、运动模式。建立齿轮同步关系

参　数：slave axis no　从轴的轴号；范围：根据控制器实际硬件决定

numerator　　电子齿轮比分子；类型 ULONG

denominator　电子齿轮比分子；类型 ULONG

refsrc　从轴跟随主轴的位置源；范围：0：理论位置，1：实际位置

mode　主从轴齿轮关系模式；范围：0：相对位置主从模式，1：绝对位置主从模式

注　意：一旦建立电子齿轮关系，主从轴状态会变成同步状态，要解除电子齿轮关系，需对从轴下 STOPDEC 或 STOPEMG 命令，齿轮关系即解除。

例　程

```
BASE 0,1
SVON
'齿轮关系的所有轴速度等参数由主轴参数决定
VL=1000
```

```
VH=40000
ACC=200000
DEC=200000
STOPDEC AX(1) '电子齿轮运动中，对从轴下 STOPDEC 或 STOPEMG 可以解除齿轮关系
'如果已有齿轮关系存在，再下 GEAR 命令，会报无效轴状态错误
GEAR AX(1),1,1,0,0    '从轴为轴 1，齿轮分子分母分别为 1,1，从轴参考主轴理论位置，相对位置模式
BASE 0
MOVE 20000    '主轴执行距离为 20000 的正向相对运动，从轴这时会与跟随主轴一起执行齿轮运动
```

2.9.9 PHASE

所　属：命令

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

描　述：电子齿轮过程中使从轴进行相位超前或落后运动

参　数：slave axis no　从轴的轴号；范围：根据控制器实际硬件决定

acc　　相位运动的加速度；类型 DOUBLE

dec　　相位运动的减速度；类型 DOUBLE

phase_speed　　相位运动的运行速度；类型 DOUBLE

phase_dist　　相位运动的超前或落后距离；类型 DOUBLE。Phase_dist >0, 从轴做相位超前运动；phase_dist <0,从轴做相位落后运动。

例　程

```
BASE 0,1
SVON
'齿轮关系的所有轴速度等参数由主轴参数决定
VL=1000
VH=40000
ACC=200000
DEC=200000
STOPDEC AX(1) '电子齿轮运动中，对从轴下 STOPDEC 或 STOPEMG 可以解除齿轮关系
'如果已有齿轮关系存在，再下 GEAR 命令，会报无效轴状态错误
```

```
GEAR AX(1),1,1,0,0    '从轴为轴 1，齿轮分子分母分别为 1,1，从轴参考主轴理论位置，相对位置模式
BASE 0
MOVE 80000    '主轴执行距离为 20000 的正向相对运动，从轴这时会与跟随主轴一起执行齿轮运动
Sleep 1000
Phase AX(1),50000,50000,30000,10000 '轴 1 进行距离为 10000 个脉冲当量的相位超前运动。
```

2.9.10 TANGENT

所　属：命令

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

描　述：指定切向跟随轴、起始切向向量、参考源平面、运动方向，建立切向跟随关系

参　数：AX(axis no)　切向跟随轴的轴号

start_vector*　起始切向向量数组地址

working_plane　切向跟随轴跟随的参考平面；范围：0：轴 0 和轴 1 组成的平面（该值暂只支持赋值 0）

dir　跟随轴旋转方向；范围：0：与切向方向相同，1：与切向方向相反

module_range　刀向旋转轴旋转一周的指令脉冲数

注　意：一旦建立切向跟随，主从轴状态会变成同步状态，要解除切向跟随关系，需对从轴下 STOPDEC 或 STOPEMG 命令，切向跟随关系即解除。

例　程

```
BASE 0,1
DIM StartArray(3) as SHORT
StartArray(0)=0
StartArray(1)=1
StartArray(2)=0
'跟随的旋转刀控制轴为轴 2，起始刀向向量为（0,1,0），参考平面为轴 0，轴 1 组成的平面，旋转方向与刀向方向相同，旋转刀轴运动一圈需要 3600 个脉冲
TANGENT AX(2),StartArray(),0,0,3600
CIRC 0,8000, 0,16000, 0   ' 轴 0，轴 1 进行圆弧插补运动，这时轴 2 会同时执行刀向跟随运动
WAIT DONE
STOPDEC AX(2)
```

2.9.11 MODULO

所　属：属性

语　法：MODULO=value

类　型：ULONG

描　述：设置/读取 ModuleRange 值。切向跟随功能中，该值为刀向旋转轴旋转一周的指令脉冲数

范　围：【0，8000000】，该值必须为 4 的倍数；默认值为 0

例　程

```
BASE 0

MODULO =10000 '設定軸 0 的 MODULO 值为 10000
```

2.10 输入输出端口控制

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.10.1	DIN	数字量输入	√	√
2.10.2	DOUT	数字量输出	√	√
2.10.3	ALM_EN	伺服报警使能	√	×
2.10.4	ALM_FILTER	伺服报警端口滤波	√	√
2.10.5	ALM_LOGIC	伺服报警端口逻辑电平	√	√
2.10.6	ALM_MODE	伺服报警触发后运动停止模式	√	√
2.10.7	IN1STOP_EN	IN1STOP 功能使能	√	×
2.10.8	IN1_FILTER	IN1 端口滤波	√	√
2.10.9	IN1STOP_EDGE	IN1STOP 功能的触发条件	√	√
2.10.10	IN1STOP_MODE	IN1STOP 触发后运动停止模式	√	√
2.10.11	IN2STOP_EN	IN2STOP 功能使能	√	×
2.10.12	IN2_FILTER	IN2 端口滤波	√	√
2.10.13	IN2STOP_EDGE	IN2STOP 功能的触发条件	√	√
2.10.14	IN2STOP_MODE	IN2STOP 触发后运动停止模式	√	√
2.10.15	IN4STOP_EN	IN4STOP 功能使能	√	×
2.10.16	IN4_FILTER	IN4 端口滤波	√	√
2.10.17	IN4STOP_EDGE	IN4STOP 功能的触发条件	√	√
2.10.18	IN4STOP_MODE	IN4STOP 触发后运动停止模式	√	√
2.10.19	IN5STOP_EN	IN5STOP 功能使能	√	×
2.10.20	IN5_FILTER	IN5 端口滤波	√	√
2.10.21	IN5STOP_EDGE	IN5STOP 功能的触发条件	√	√
2.10.22	IN5STOP_MODE	IN5STOP 触发后运动停止模式	√	√
2.10.23	INSTOP_DEC	INSTOP 功能触发后减速停止	√	×
2.10.24	INP_EN	伺服到位功能使能	√	×
2.10.25	INP_LOGIC	伺服到位端口逻辑电平	√	√

2.10.26	LTC_EN	锁存功能使能	√	×
2.10.27	LTC_LOGIC	锁存端口逻辑电平	√	√
2.10.28	LDPOS	锁存到的理论位置	√	×
2.10.29	LMPOS	锁存到的实际位置	√	×
2.10.30	TRIGLTC	软件触发锁存功能	√	×
2.10.31	LTC_FLAG	轴锁存信号标志	√	×
2.10.32	RESETLTC	清除锁存位置和标识	√	×
2.10.33	CMP_EN	比较触发功能使能	√	×
2.10.34	CMP_LOGIC	比较触发端口逻辑电平	√	√
2.10.35	CMP_METHOD	比较触发功能比较方法	√	√
2.10.36	CMP_MODE	比较触发的 DO 输出模式	√	√
2.10.37	CMP_SRC	比较触发的比较源	√	√
2.10.38	CMP_WIDTH	比较触发的 DO 输出模式为脉冲模式时的电平宽度	√	√
2.10.39	CPOS	当前比较位置数据	√	×
2.10.40	CMP	比较触发模式选择	√	×
2.10.41	CMP_FLAG	比较触发信号标志	√	×
2.10.42	RESETCMP	清除比较触发信号标志	√	×
2.10.43	CAMDO_EN	CAMDO 功能使能	√	×
2.10.44	CAMDO_LOGIC	CAMDO 端口逻辑电平	√	√
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 比较触发源	√	√
2.10.48	MIO	运动控制相关的 I/O 状态	×	×

2.10.1 DIN

所　属：命令（只读）

语　法：DIN (no)

描　述：读取一个通用数字量输入端口的状态

参　数：no，数字量输入的编号，在控制器配置时，系统会根据控制器硬件分配对应的编号；类型：ULONG

返回值：0：低电平，1：高电平

例　程

```
WHILE 1
IF DIN (0)=1 THEN        '判断 DI0 信号是否有信号
    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　属：命令

语　法：DOUT (no)=value

描　述：设置/读取一个通用数字量输出端口的状态

参　数：no　数字量输出的编号，在控制器配置时，系统会根据控制器硬件分配对应的编号；类型：ULONG

value　数字量输出端口的状态，范围：0 或 1

返回值：输出口的电平，0 或 1

例　程

```
WHILE 1
IF DIN (0)=1 THEN        '判断 DI0 信号是否有信号
    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　属：属性

语　法：ALM_EN = value

类　型：ULONG

描　述：启用/禁用运动报警功能，报警是当电机驱动处于报警状态时，电机驱动器生成的信号

范　围:设定值和返回值如下，默认值 0

0：禁用(默认值)

1：启用

例　程

```
BASE 0
ALM_EN=1    '启用轴 0 检测报警功能
```

2.10.4 ALM_FILTER

所　属：属性

语　法：ALM_FILTER = value

类　型：ULONG

描　述：设置/读取軸的报警(ALM) 輸入端口的滤波参数

范　围:设定值和返回值如下，默认值 0

0：5us

1：100us

2：200us

3：500us

例　程

```
BASE 0
ALM_FILTER=1 '设定轴 0 的 ALM 输入端口滤波时间为 100us
```

2.10.5 ALM_LOGIC

所　属：属性

语　法：ALM_LOGIC = value

类　型：ULONG

描　述：设置/读取报警輸入信号的有效逻辑电平

范　围：设定值和返回值如下，默认值 0

0：低电平

1：高电平

例　程

```
BASE 0
ALM_LOGIC =1   '设置轴 0 的报警输入信号高电平有效
```

2.10.6 ALM_MODE

所　属：属性

语　法：ALM_MODE　= value

类　型：ULONG

描　述：设置/读取接收报警信号时电机的停止模式

范　围：设定值和返回值如下，默认值 0

0：立即停止

1：减速停止

例　程

```
BASE 0
ALM_MODE=1   '设置接收到报警信号后电机减速停止
```

2.10.7 IN1STOP_EN

所　属：属性

语　法：IN1STOP_EN = value

类　型：ULONG

描　述：启用/禁用 IN1 的触发停止功能，该功能启动用后，IN1 信号一旦有效，在运动

中的指定电机会被控制停止运动。研华运动控制的每个轴都关联着 4 个 DI 端口，分别称为 IN1，IN2，IN4，IN5。4 个端口都可以指定启用 INSTOP 功能。

范　围:设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

BASE 0

IN1STOP_EN =1　'启用轴 0 的 IN1 触发停止功能

2.10.8 IN1_FILTER

所　属：属性

语　法：IN1_FILTER = value

类　型：ULONG

描　述：设置/读取 IN1 端口的滤波时间

范　围:设定值和返回值如下，默认值 0

0：5us

1：100us

2：200us

3：500us

例　程

BASE 0

IN1_FILTER =1　'设置轴 0 的 IN1 信号滤波时间 100us

2.10.9 IN1STOP_EDGE

所　属：属性

语　法：IN1STOP_EDGE = value

类　型：ULONG

描　述：设置/读取 IN1STOP 功能的触发条件。设置上升沿触发时，IN1 端口有上升沿信号时， IN1STOP 功能被触发。设置下降沿触发时，IN1 端口有下降沿信号时，IN1STOP 功能被触发。

范　围:设定值和返回值如下，默认值 0

0：上升沿

1：下降沿

例　程

```
BASE 0
IN1STOP_EDGE =1 '设置 IN1STOP 功能的触发条件为下降沿触发有效。
```

2.10.10 IN1STOP_MODE

所　属：属性

语　法：IN1STOP_MODE　= value

类　型：ULONG

描　述：设置/读取 IN1 触发时电机的停止模式

范　围:设定值和返回值如下，默认值 1

0：立即停止

1：减速停止

例　程

```
BASE 0
IN1STOP_MODE =1 '设置 IN1 触发时电机作减速停止运动
```

2.10.11 IN2STOP_EN

所　属：属性

语　法：IN2STOP_EN = value

类　型：ULONG

描　述：启用/禁用 IN2 的触发停止功能，该功能启动用后，IN2 信号一旦有效，在运动中的指定电机会被控制停止运动。研华运动控制的每个轴都关联着 4 个 DI 端口，分别称为 IN1，IN2，IN4，IN5。4 个端口都可以指定启用 INSTOP 功能。

范　围:设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

```
BASE 0
IN2STOP_EN =1　'启用轴 0 的 IN2 触发停止功能
```

2.10.12 IN2_FILTER

所　属：属性

语　法：IN2_FILTER = value

类　型：ULONG

描　述：设置/读取 IN2 端口的滤波时间

范　围：设定值和返回值如下，默认值 0

0：5us

1：100us

2：200us

3：500us

例　程

```
BASE 0

IN2_FILTER =1   '设置轴 0 的 IN2 信号滤波时间 100us
```

2.10.13 IN2STOP_EDGE

所　属：属性

语　法：IN2STOP_EDGE = value

类　型：ULONG

描　述：设置/读取 IN2STOP 功能的触发条件。设置上升沿触发时，IN2 端口有上升沿信号时， IN2STOP 功能被触发。设置下降沿触发时，IN2 端口有下降沿信号时，IN2STOP 功能被触发。

范　围：设定值和返回值如下，默认值 0

0：上升沿

1：下降沿

例　程

```
BASE 0

IN2STOP_EDGE =1 '设置 IN2STOP 功能的触发条件为下降沿触发有效。
```

2.10.14 IN2STOP_MODE

所　属：属性

语　法：IN2STOP_MODE　= value

类　型：ULONG

描　述：设置/读取 IN2 触发时电机的停止模式

范　围：设定值和返回值如下，默认值 1

0：立即停止

1：减速停止

例　程

```
BASE 0
IN2STOP_MODE =1 '设置 IN2 触发时电机作减速停止运动
```

2.10.15 IN4STOP_EN

所　属：属性

语　法：IN4STOP_EN = value

类　型：ULONG

描　述：启用/禁用 IN4 的触发停止功能，该功能启动用后，IN4 信号一旦有效，在动中的指定电机会被控制停止运动。研华运动控制的每个轴都关联着 4 个 DI 端、口，分别称为 IN1，IN2，IN4，IN5。4 个端口都可以指定启用 INSTOP 功能。

范　围：设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

```
BASE 0
IN4STOP_EN =1   '启用轴 0 的 IN4 触发停止功能
```

2.10.16 IN4_FILTER

所　属：属性

语　法：IN4_FILTER = value

类　型：ULONG

描　述：设置/读取 IN4 端口的滤波时间

范　围：设定值和返回值如下，默认值 0

0：5us

1：100us

2：200us

3：500us

例　程

BASE 0

IN4_FILTER =1　'设置轴 0 的 IN4 信号滤波时间 100us

2.10.17 IN4STOP_EDGE

所　属：属性

语　法：IN4STOP_EDGE = value

类　型：ULONG

描　述：设置/读取 IN4STOP 功能的触发条件。设置上升沿触发时，IN4 端口有上升沿信号时， IN4STOP 功能被触发。设置下降沿触发时，IN4 端口有下降沿信号时，IN4STOP 功能被触发。

范　围:设定值和返回值如下，默认值 0

0：上升沿

1：下降沿

例　程

BASE 0

IN4STOP_EDGE =1 '设置 IN4STOP 功能的触发条件为下降沿触发有效。

2.10.18 IN4STOP_MODE

所　属：属性

语　法：IN4STOP_MODE　= value

类　型：ULONG

描　述：设置/读取 IN4 触发时电机的停止模式

范　围：设定值和返回值如下，默认值 1

0：立即停止

1：减速停止

例　程

BASE 0

IN4STOP_MODE =1 '设置 IN4 触发时电机作减速停止运动

2.10.19 IN5STOP_EN

所　属：属性

语　法：IN5STOP_EN = value

类　型：ULONG

描　述：启用/禁用 IN5 的触发停止功能，该功能启动用后，IN5 信号一旦有效，在运动中的指定电机会被控制停止运动。研华运动控制的每个轴都关联着 4 个 DI 端口，分别称为 IN1，IN2，IN4，IN5。4 个端口都可以指定启用 INSTOP 功能。

范　围:设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

```
BASE 0
IN5STOP_EN =1   '启用轴 0 的 IN5 触发停止功能
```

2.10.20 IN5_FILTER

所　属：属性

语　法：IN5_FILTER = value

类　型：ULONG

描　述：设置/读取 IN5 端口的滤波时间

范　围:设定值和返回值如下，默认值 0

0：5us

1：100us

2：200us

3：500us

例　程

```
BASE 0
IN5_FILTER =1   '设置轴 0 的 IN5 信号滤波时间 100us
```

2.10.21 IN5STOP_EDGE

所　属：属性

语　法：IN5STOP_EDGE = value

类　型：ULONG

描　述：设置/读取 IN5STOP 功能的触发条件。设置上升沿触发时，IN5 端口有上升沿信号时， IN5STOP 功能被触发。设置下降沿触发时，IN5 端口有下降沿信号时，IN5STOP 功能被触发。

范　围：设定值和返回值如下，默认值 0

0：上升沿

1：下降沿

例　程

```
BASE 0
IN5STOP_EDGE =1 '设置 IN5STOP 功能的触发条件为下降沿触发有效。
```

2.10.22 IN5STOP_MODE

所　属：属性

语　法：IN5STOP_MODE　= value

类　型：ULONG

描　述：设置/读取 IN5 触发时电机的停止模式

范　围：设定值和返回值如下，默认值 1

0：立即停止

1：减速停止

例　程

```
BASE 0
IN5STOP_MODE =1 '设置 IN5 触发时电机作减速停止运动
```

2.10.23 INSTOP_DEC

所　属：属性

语　法：INSTOP_DEC　= value

类　型：DOUBLE

描　述：设置/读取 INSTOP 用减速停止模式时的减速度，单位为脉冲当量/s^2

范　围：（0, MAXDEC），默认值 100000

例　程

```
BASE 0
```

```
INSTOP_DEC=20000 '设置轴 0 的 INSTOP 减速度为 20000 个脉冲当量/s^2
```

2.10.24 INP_EN

所　属：属性

语　法：INP_EN = value

类　型：ULONG

描　述：启用/禁用检测电机运动到位功能，到位信号是当电机运动到位时，电机驱动器生成到位信号。

范　围:设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

```
BASE 0
INP_EN =1 '启用轴 0 的到位检测功能
```

注　意

启用到位检测功能后，控制轴进行运动时，脉冲命令输出完后，轴状态不会立即变为 ready，要等到轴的 INP 端口接收到伺服送出的到位信号，轴状态才会变为 ready。轴状态未变为 ready 时，对该轴下运动指令将不成功。

2.10.25 INP_LOGIC

所　属：属性

语　法：INP_LOGIC = value

类　型：ULONG

描　述：设置/读取到位输入信号的有效逻辑电平

范　围:设定值和返回值如下，默认值 1

0：低电平

1：高电平

例　程

```
BASE 0
INP_LOGIC =1    '设置轴 0 的到位输入信号高电平有效
```

2.10.26 LTC_EN

所　属：属性

语　法：LTC_EN = value

类　型：ULONG

描　述： 启用/禁用轴的锁存功能，研华运动控制的每个轴都关联着 4 个 DI 端口，分别称为 IN1，IN2，IN4，IN5，每个轴的锁存信号由 IN1 输入端口产生，启用锁存功能后，一旦 IN1 端口接收到有效电平，控制器会立即锁存指令理论位置值和编码器实际位置值。

范　围:设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

```
BASE 0
LTC_EN=1 '启用轴 0 锁存功能
```

2.10.27 LTC_LOGIC

所　属：属性

语　法：LTC_LOGIC = value

类　型：ULONG

描　述：设置/读取锁存输入信号的有效逻辑电平

范　围:设定值和返回值如下，默认值 0

0：低电平

1：高电平

例　程

```
BASE 0
LTC_LOGIC =0   '设置轴 0 锁存输入信号低电平有效
```

2.10.28 LDPOS

所　属：命令

语　法：value=LDPOS(AX(no))

类　型：DOUBLE

描　述：读取触发锁存得到的理论位置值

返回值：轴指令理论位置值

例　程

```
'完整例程可参考 TrigLTC 指令
DIM B AS DOUBLE
B=LDPOS(AX(1)) '获取轴 1 的锁存理论位置值
```

2.10.29 LMPOS

所　属：命令

语　法：value=LMPOS(AX(no))

类　型：DOUBLE

描　述：读取触发锁存得到的编码器实际位置值

返回值：轴编码器实际位置值

例　程

```
'完整例程可参考 TrigLTC 指令
DIM B AS DOUBLE
B=LMPOS(AX(1)) '获取轴 1 的锁存编码器实际位置值
```

2.10.30 TrigLTC

所　属：命令

语　法：TrigLTC AX(axis no)

描　述:用软件命令触发产生锁存信号。实际应用中 ,锁存信号基本上由硬件信号触发，该命令多用于测试。

参　数：axis no　轴号；　范围：根据控制器实际硬件决定。

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)          '触发轴 1 的锁存信号，进行位置锁存
LTC_CmdDATA=LDPOS(AX(1))      '将锁存到的指令理论位置读取出来
PRINT LTC_CmdDATA
LTC_FBDATA=LMPOS(AX(1))       '将锁存到的编码器实际位置读取出来
PRINT LTC_FBDATA
RESETLTC AX(1)                '清除锁存标记，锁存位置值
```

2.10.31 LTC_Flag

所　属：命令

语　法：value=LTC_Flag(AX(no))

类　型：ushort

描　述：读取轴锁存标志，捕捉到锁存信号时，LTC_Flag 会置 1。要清除锁存标志，需要用 ResetLTC 指令清除。

返回值：锁存标志信号

例　程

```
DIM B AS USHORT
B= LTC_Flag(AX(1)) '获取轴 1 的锁存标志信号
```

2.10.32 RESETLTC

所　属：命令

语　法：RESETLTC AX(axis no)

描　述：清除锁存数据、锁存标记。未清除锁存标记，下次接收到锁存信号时，将不进行位置值的锁存。

参　数：axis no　轴号；　范围：根据控制器实际硬件决定。

例　程

```
'参考 TrigLTC 指令例程
```

2.10.33 CMP_EN

所　属：属性

语　法：CMP_EN = value

类　型：ULONG

描　述： 启用/禁用轴比较触发功能，研华运动控制的每个轴都关联着 4 个 DO 端口，分别称为 OUT4，OUT5，OUT6，OUT7，每个轴的比较触发输出由 OUT5 输出端口产生，启用比较触发功能后，一旦比较触发产生，OUT5 即输出信号。

范　围:设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

BASE 0

CMP_EN =1 '启用轴 0 比较触发功能

2.10.34 CMP_LOGIC

所　属：属性

语　法：CMP_LOGIC = value

类　型：ULONG

描　述：设置/读取轴比较触发有效输出时的 DO 逻辑电平

范　围:设定值和返回值如下，默认值 0

0：低电平

1：高电平

例　程

BASE 0

CMP_LOGIC　=0 '设置比较触发输出的 DO 逻辑电平为低电平

2.10.35 CMP_METHOD

所　属：属性

语　法：CMP_METHOD = value

类　型：ULONG

描　述：设置/读取轴比较触发的比较方法

范　围:设定值和返回值如下，默认值 0

0：>= 位置计数器

1：<= 位置计数器

2：= 计数器 （无方向）

例　程

BASE 0

CMP_METHOD　=1'设置轴 0 的比较方法为小于等于位置计数器

2.10.36 CMP_MODE

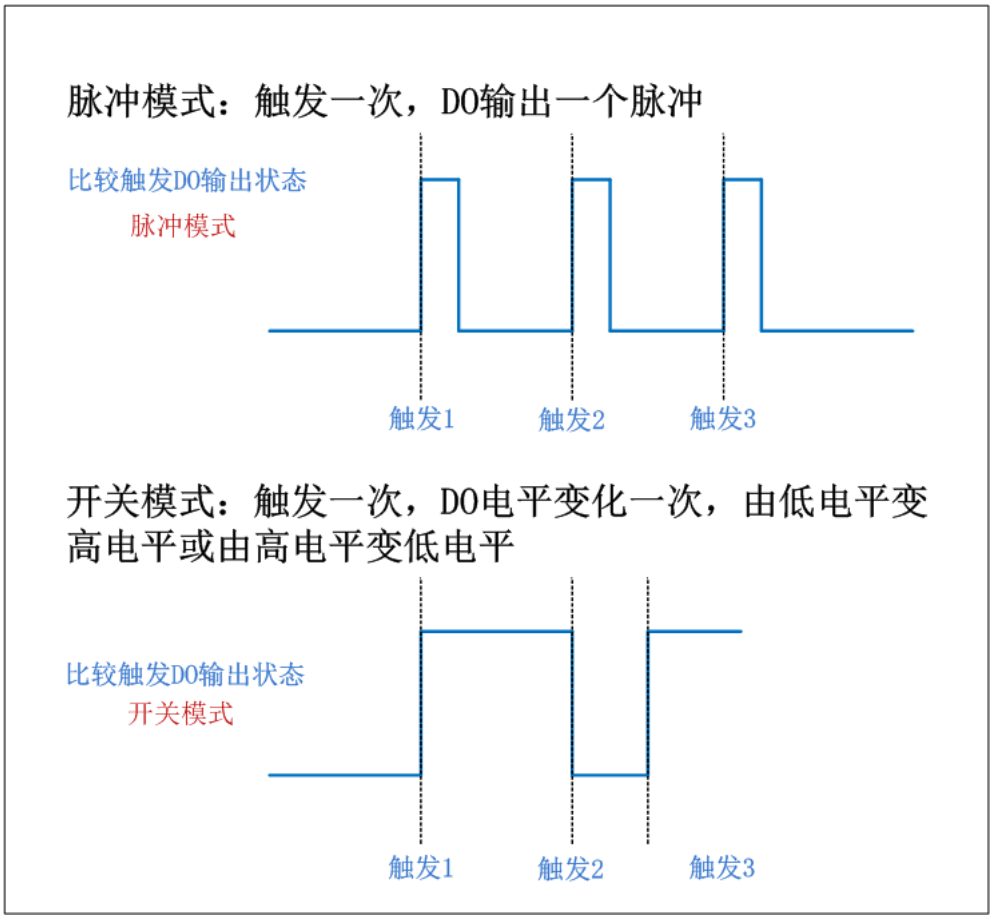
所　属：属性

语　法：CMP_MODE = value

类　型：ULONG

描　述：设置/读取轴比较触发的 DO 输出模式，DO 输出模式说明如下图。

比较触发 DO 输出模式说明



范　围：设定值和返回值如下，默认值 0

0：脉冲模式

1：开关模式

例　程

BASE 0

```
CMP_MODE   =1 '设置軸 0 的比较触发 DO 输出模式为开关模式
```

2.10.37 CMP_SRC

所　属：属性

语　法：CMP_SRC = value

类　型：ULONG

描　述：设置/读取轴比较触发的比较源

范　围：设定值和返回值如下，默认值 0

0：理论位置

1：实际位置

例　程

```
BASE 0
CMP_SRC   =1   '设置轴 0 的位置比较源为实际位置
```

2.10.38 CMP_WIDTH

所　属：属性

语　法：CMP_WIDTH = value

类　型：ULONG

描　述：设置/读取轴比较触发的 DO 输出模式为脉冲模式时的电平宽度，单位为微秒

范　围：设定值和返回值如下，默认值 0

0： 5 Microsecond(us)

1： 10 Microsecond(us)

2： 20 Microsecond(us)

3： 50 Microsecond(us)

4： 100 Microsecond(us)

5： 200 Microsecond(us)

6： 500 Microsecond(us)

7： 1000 Microsecond(us)

例　程

```
BASE 0
CMP_WIDTH   =1 '设置轴 0 比较触发的 DO 电平宽度为 10us
```

2.10.39 CPOS

所　属：命令

语　法：value=CPOS(AX(no))

类　型：DOUBLE

描　述：读取比较触发中的当前比较位置数据

返回值：轴当前设定的比较位置数据值

例　程

```
DIM B AS DOUBLE
B=CPOS(AX(1))    '获取轴 1 的当前笔比较位置值
```

2.10.40 CMP

所　属：命令

语　法 1：CMP AX(axis no)， position

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

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

描　述：比较触发的模式分 3 种。

✧ 语法 1 用于单点比较触发，设置一个比较点的比较位置值。

✧ 语法 2 用于等距间隔多点比较触发，设置起始比较点位置，终点比较点位置，间隔距离。

✧ 语法 3 用于随机点多点比较触发，设置多个要比较的位置值。

参　数：axis no　轴号； 范围：根据控制器实际硬件决定。

Position　比较触发的位置值

start_position　起始比较点位置值，用于等距间隔多点比较模式

end_position　终点比较点位置值，用于等距间隔多点比较模式

interval　间隔距离，用于等距间隔多点比较模式

tablearray（ ）　随机比较点数组，将要比较的点依次赋值到一个数组里，用于随机点多点比较模式

array_num　比较点个数，用于随机点多点比较模式

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 '大于等于位置源时触发
CMP_LOGIC=0 '触发电平为低电平
CMP_SRC=0   '参考源为理论位置
CMP_WIDTH=4 '脉冲模式输出脉宽为 100us
CMP_MODE=0 'DO 输出模式脉冲模式
'单点比较触发
CMP AX(0),10000 '设置比较位置为 10000
MOVEABS 0 '先运动到位置 0
WAIT DONE
MOVEABS 20000 '运到到 10000 的瞬间，比较触发的 DO 输出
WAIT Done
'均匀间隔比较触发
CMP AX(0),30000,50000,4000 '比较位置从 30000 到 50000，每隔 4000，触发一次 DO
MOVEABS 60000
WAIT DONE
'随机表格位置比较触发
CMP AX(0),cmp_table(),3
MOVEABS 80000
```

2.10.41 CMP_Flag

所　属：命令

语　法：value=CMP_Flag(AX(no))

类　型：ushort

描　述：读取轴比较触发标志，发生轴比较触发时，CMP_Flag 会置 1。要清除锁存标志，需要用 ResetCMP 指令清除。

返回值：比较触发标志信号

例　程

```
DIM B AS USHORT
B= CMP_Flag(AX(1)) '获取轴 1 的比较触发标志信号
```

2.10.42 RESETCMP

所　属：命令

语　法：RESETCMP AX(axis no)

描　述：清除比较触发标志。

参　数：axis no　轴号；　范围：根据控制器实际硬件决定。

2.10.43 CAMDO_EN

所　属：属性

语　法：CAMDO_EN = value

类　型：ULONG

描　述：　启动/禁用 CAMDO 功能，研华运动控制的每个轴都关联着 4 个 DO 端口，分别称为 OUT4，OUT5，OUT6，OUT7，每个轴的 CAMDO 输出由 OUT4 输出端口产生，启用 CAMDO 功能后，一旦触发产生，OUT4 即输出信号。

范　围:设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

```
BASE 0
CAMDO_EN=1    '启用轴 0 上的 CAMDO 功能
```

2.10.44 CAMDO_LOGIC

所　属：属性

语　法：CAMDO_LOGIC = value

类　型：ULONG

描　述：设置/读取 CAMDO 有效输出时 DO 的电平逻辑

范　围：设定值和返回值如下，默认值 1

0：低电平

1：高电平

例　程

```
BASE 0
CAMDO_LOGIC　=1 '设置 CAMDO 有效输出时的电平为高电平
```

2.10.45 CAMDO_LPOS

所　属：属性

语　法：CAMDO_LPOS = value

类　型：ULONG

描　述：　设置/读取 CAMDO 低限位位置值，单位为脉冲当量

范　围：设定值和返回值为【-2147483647，2147483647】，默认值 10000

例　程

```
BASE 0
CAMDO_LPOS=1000　'设置 CAMDO 低限位位置值为 1000
```

2.10.46 CAMDO_HPOS

所　属：属性

语　法：CAMDO_HPOS = value

类　型：ULONG

描　述：设置/读取 CAMDO 高限位位置值，单位为脉冲当量

范　围：设定值和返回值为【-2147483647，2147483647】，默认值 20000

例　程

```
BASE 0
CAMDO_HPOS=1000　'设置 CAMDO 高限位位置值为 1000
```

2.10.47 CAMDO_SRC

所　属：属性

语　法：CAMDO_SRC = value

类　型：ULONG

描　述：设置/读取 CAMDO 的位置比较源

范　围：设定值和返回值如下，默认值 0

0：理论位置

1：实际位置

例　程

BASE 0

CAMDO_SRC=1 '设置轴 0 的 CAMDO 位置比较源为实际位置

2.10.48 MIO

所　属：属性（只读）

语　法：如下枚举

- value=MIO.RDY，读取伺服驱动器 Ready 信号
- value=MIO.ALM，读取伺服驱动器报警信号
- value=MIO.PEL，读取正方向硬极限信号
- value=MIO.NEL，读取负方向硬极限信号
- value=MIO.ORG，读取硬件原点信号
- value=MIO.DIR，读取轴运动方向信号
- value=MIO.EMG，读取轴卡上 EMG 信号
- value=MIO.PCS，暂不支持
- value= MIO.ERC，暂不支持
- value= MIO.EZ，读取编码器 Z 相输入信号
- value= MIO.CLR，暂不支持
- value= MIO.LTC，读取锁存信号
- value= MIO.SD，暂不支持
- value= MIO.INP，读取伺服驱动器到位信号
- value= MIO.SVON，读取伺服使能输出信号
- value= MIO.ALRM，读取报警复位信号输出状态
- value= MIO.SPEL，读取正方向软极限信号

- value= MIO.SNEL，读取负方向软极限信号
- value= MIO.CMP，读取比较触发输出信号
- value= MIO.CAMDO，读取 CAM DO 输出信号

类　型：ULONG

描　述：读取跟运动控制相关的 I/O 状态

返回值：0 或 1

2.11 回原点与限位

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.11.1	HOME_VL	原点运动初速度	√	√
2.11.2	HOME_VH	原点运动运行速度	√	√
2.11.3	HOME_ACC	原点运动加速度	√	√
2.11.4	HOME_DEC	原点运动减速度	√	√
2.11.5	HOME_JK	原点运动速度曲线类型	√	√
2.11.6	HOME_MODE	原点运动模式	√	√
2.11.7	HOMEP	正向原点运动	√	×
2.11.8	HOMEN	反向原点运动	√	×
2.11.9	HOME_CROSS	原点运动跨越距离	√	√
2.11.10	HOME_OFFSETDIST	原点运动完成后再移动的偏移距离	√	√
2.11.11	HOME_OFFSETVEL	原点运动完成后移动偏移距离的速度	√	√
2.11.12	HOME_RESET	原点运动后清零位置值功能	√	×
2.11.13	ORG_LOGIC	原点端口逻辑电平	√	√
2.11.14	ORG_MODE	原点运动结束时的停止模式	√	√
2.11.15	ORG_FILTER	原点端口的滤波	√	√
2.11.16	EZ_LOGIC	Z 相端口逻辑电平	√	√
2.11.17	EL_EN	硬限位功能使能	√	×
2.11.18	EL_LOGIC	硬限位端口逻辑电平	√	√
2.11.19	EL_MODE	硬限位触发时的停止模式	√	√
2.11.20	PEL_FILTER	正方向硬限位端口滤波	√	√
2.11.21	NEL_FILTER	负方向硬限位端口滤波	√	√

2.11.22	SPEL	正方向软限位值	√	√
2.11.23	SPEL_EN	正方向软限位功能使能	√	×
2.11.24	SPEL_MODE	正方向软限位触发时的停止模式	√	√
2.11.25	SNEL	负方向软限位值	√	√
2.11.26	SNEL_EN	负方向软限位功能使能	√	×
2.11.27	SNEL_MODE	负方向软限位触发时的停止模式	√	√
2.11.28	PEL_TOL_EN	正方向硬极限容差功能使能	√	×
2.11.29	PEL_TOL	正方向硬极限容差值	√	√
2.11.30	NEL_TOL_EN	负方向硬极限容差功能使能	√	×
2.11.31	NEL_TOL	负方向硬极限容差值	√	√
2.11.32	SPEL_TOL_EN	正方向软极限容差功能使能	√	×
2.11.33	SPEL_TOL	正方向软极限容差值	√	√
2.11.34	SNEL_TOL_EN	负方向软极限容差功能使能	√	×
2.11.35	SNEL_TOL	负方向软极限容差值	√	√

2.11.1 HOME_VL

所　属：属性

语　法：HOME_VL = value

类　型：DOUBLE

描　述：设置/读取回原点运动的初速度，单位为脉冲当量/s

范　围：（0, MAXVEL），默认值 2000

例　程

```
BASE 0
HOME_VL=1000 '設定軸 0 回原点运动的初速度为 1000 个脉冲当量/s
```

2.11.2 HOME_VH

所　属：属性

语　法：HOME_VH = value

类　型：DOUBLE

描　述：设置/读取回原点运动的最大运行速度，单位为脉冲当量/s

范　围：（HOME_VL, MAXVEL），默认值 8000

例　程

```
BASE 0
HOME_VH=5000 '設定軸 0 回原点运动的运行速度为 5000 个脉冲当量/s
```

2.11.3 HOME_ACC

所　属：属性

语　法：HOME_ACC = value

类　型：DOUBLE

描　述：设置/读取回原点运动的加速度，单位为脉冲当量/s^2

范　围：（0, MAXACC），默认值 10000

例　程

```
BASE 0
HOME_ACC=20000 '设置轴 0 回原点的加速度为 20000 个脉冲当量/s^2
```

2.11.4 HOME_DEC

所　属：属性

语　法：HOME_DEC = value

类　型：DOUBLE

描　述：设置/读取回原点运动的减速度，单位为脉冲当量/s^2

范　围：（0, MAXDEC），默认值 10000

例　程

```
BASE 0
HOME_DEC=20000 '设置轴 0 回原点的减速度为 20000 个脉冲当量/s^2
```

2.11.5 HOME_JK

所　属：属性

语　法：HOME_JK = value

类　型：ULONG

描　述：设置/读回原点运动的速度曲线类型

范　围：【0,1】，0：T 型曲线；1：S 型曲线，默认值 0

例　程

```
BASE 0
HOME_JK=1   '设置轴 0 回原点的速度曲线为 S 型曲线
```

2.11.6 HOME_MODE

所　属：属性

语　法：HOME_MODE = value

类　型：ULONG

描　述：设置/读取回原点运动的模式

范　围：设定值和返回值如下，默认值 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 的回原点模式未 MODE8_LmtSearch
```

2.11.7 HOMEP

所　属：命令

语　法 1：HOMEP

语　法 2：HOMEP AX(axis no)

语　法 3：HOMEP dir1[，dir2][，dir3]……

描　述：BASE 轴列表的轴或指定轴、方向，开始正向回原点运动。HOMEP 分 3 种方法使用如下。

- ✧ 语法 1 用于对 BASE 轴列表的轴执行正向回原点运动
- ✧ 语法 2 用于指定某个轴执行正向回原点运动
- ✧ 语法 3 用于对 BASE 轴列表的轴，指定不同方向，执行回原点运动。Dir 为 0 时，方向与 HOMEP 同向；dir 为 1 时，方向与 HOMEP 反向

参　数：dir　0：正向；1：反向

axis no　轴号；　范围：根据控制器实际硬件决定。

相关指令参考：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   '设置原点运动中跨越距离
HOME_RESET=1      '原点运动结束后清零理论位置、实际位置
HOME_MODE=6       '6：MODE7_AbsSearch
HOMEP AX(1)       '指定轴 1 执行正向回原点运动
WAIT AX(1),DONE   '等待轴 1 运动停止
BASE 0,1          '基于轴 0,轴 1
```

```
HOMEP                  '轴 0，轴 1 都执行正向回原点运动
WAIT DONE              '等待两个轴的回原点运动停止
HOMEN                  '轴 0，轴 1 都执行负向回原点运动
WAIT DONE
HOMEP 0,1              '轴 0 执行正向回原点运动，轴 1 执行反向回原点运动
WAIT DONE
MOVE 5000,5000     '原点运动停止后，执行两轴相对点位运动
```

2.11.8 HOMEN

所　属：命令

语　法 1：HOMEN

语　法 2：HOMEN AX(axis no)

语　法 3：HOMEN dir1[，dir2][，dir3]……

描　述：BASE 轴列表的轴或指定轴、方向，开始负向回原点运动。HOMEN 分 3 种方法使用如下。

- ✧ 语法 1 用于对 BASE 轴列表的轴执行负向回原点运动
- ✧ 语法 2 用于指定某个轴执行负向回原点运动
- ✧ 语法 3 用于对 BASE 轴列表的轴，指定不同方向，执行回原点运动。Dir 为 0 时，方向与 HOMEN 同向；dir 为 1 时，方向与 HOMEN 反向

参　数：dir　0：反向；1：正向

axis no　轴号；　范围：根据控制器实际硬件决定。

相关指令参考：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    '设置原点运动中跨越距离
HOME_RESET=1       '原点运动结束后清零理论位置、实际位置
HOME_MODE=6        '6：MODE7_AbsSearch
```

```
HOMEP AX(1)          '指定轴 1 执行正向回原点运动
WAIT AX(1),DONE      '等待轴 1 运动停止
BASE 0,1             '基于轴 0,轴 1
HOMEP                '轴 0，轴 1 都执行正向回原点运动
WAIT DONE            '等待两个轴的回原点运动停止
HOMEN                '轴 0，轴 1 都执行负向回原点运动
WAIT DONE
HOMEP 0,1            '轴 0 执行正向回原点运动，轴 1 执行反向回原点运动
WAIT DONE
MOVE 5000,5000       '原点运动停止后，执行两轴相对点位运动
```

2.11.9 HOME_CROSS

所　属：属性

语　法：HOME_CROSS= value

类　型：DOUBLE

描　述：设置/读取回原点运动时的跨越距离。HOME_MODE 里有几种模式会用到 HOME_CROSS，请参考 HOME_MODE 指令说明。

例　程

```
BASE 0
HOME_CROSS=100 '設定軸 0 回原点运动时的跨越距离为 100 个脉冲当量
```

2.11.10 HOME_OFFSETDIST

所　属：属性

语　法：HOME_OFFSETDIST = value

类　型：DOUBLE

描　述：设置/读取回原点运动完成后再移动的偏移距离。

例　程

```
BASE 0
HOME_OFFSETDIS =1000 '设置軸 0 的回原点偏移距离为 1000 个脉冲当量
```

2.11.11 HOME_OFFSETVEL

所　属：属性

语　法：HOME_OFFSETVEL = value

类　型：DOUBLE

描　述：设置/读取回原点运动完成后移动偏移距离的速度

范　围：（0, MAXVEL），默认值 8000

例　程

```
BASE 0
HOME_OFFSETVEL =1000 '设置轴 0 的回原点偏移速度为 1000 个脉冲当量/s
```

2.11.12 HOME_RESET

所　属：属性

语　法：HOME_RESET= value

类　型：ULONG

描　述：启用或禁用回原点后清零位置值功能

范　围：设定值和返回值如下，默认值 1

0：禁用

1：启用

例　程

```
BASE 0
HOME_RESET=1   '启用轴 0 回完原点后清零位置值功能
```

2.11.13 ORG_LOGIC

所　属：属性

语　法：ORG_LOGIC = value

类　型：ULONG

描　述：设置/读取 ORG 信号的有效逻辑电平。ORG 专用数字量输入端口用于回原点运动中的几种用到 ORG 信号的模式，请参考 HOME_MODE 指令说明。

范　围：设定值和返回值如下，默认值 0

0：低电平

1：高电平

例　程

```
BASE 0
ORG_LOGIC =1   '设置轴 0 的 ORG 输入信号高电平有效
```

2.11.14 ORG_MODE

所　属：属性

语　法：ORG_MODE　= value

类　型：ULONG

描　述：设置/读取回原点运动结束时的停止模式。

范　围：设定值和返回值如下，默认值 1

0：立即停止

1：减速停止

例　程

```
BASE 0
ORG_MODE =1   '设置轴 0 的回原点运动停止模式未减速停止模式
```

2.11.15 ORG_FILTER

所　属：属性

语　法：ORG_FILTER = value

类　型：ULONG

描　述：设置/读取轴的 ORG 输入信号的滤波时间

范　围：设定值和返回值如下，默认值 0

0：5us

1：100us

2：200us

3：500us

例　程

```
BASE 0
ORG_FILTER =1   '设置 ORG 信号的滤波时间为 100us
```

2.11.16 EZ_LOGIC

所　属：属性

语　法：EZ_LOGIC = value

类　型：ULONG

描　述：设置/读取电机编码器 Z 相输入信号的有效逻辑电平。运动控制中，Z 相信号常常用于回原点运动中，请参考 HOME_MODE 的指令说明。

范　围：设定值和返回值如下，默认值 0

0：低电平

1：高电平

例　程

```
BASE 0
EZ_LOGIC =0   '设置轴 0 的 Z 相输入信号低电平有效
```

2.11.17 EL_EN

所　属：属性

语　法：EL_EN = value

类　型：ULONG

描　述：启用/禁用硬件限位功能，启用后限位开关被触发，相应方向上运动的电机会被控制停下来

范　围：设定值和返回值如下，默认值 1

0：禁用

1：启用

例　程

```
BASE 0
EL_EN =1 '启用硬件限位功能
```

2.11.18 EL_LOGIC

所　属：属性

语　法：EL_LOGIC = value

类　型：ULONG

描　述：设置/读取硬件限位输入信号的有效逻辑电平

范　围：设定值和返回值如下，默认值 0

0：低电平

1：高电平

例　程

BASE 0

EL_LOGIC =1 '设置轴 0 的硬极限输入信号高电平有效

2.11.19 EL_MODE

所　属：属性

语　法：EL_MODE　= value

类　型：ULONG

描　述：设置/读取接收报警信号时电机的停止模式

范　围：设定值和返回值如下，默认值 0

0：立即停止

1：减速停止

例　程

BASE 0

EL_MODE =0 '设置轴 0 碰到硬极限时电机立即停止

2.11.20 PEL_FILTER

所　属：属性

语　法：PEL_FILTER = value

类　型：ULONG

描　述：设置/读取轴的正方向硬限位输入信号的滤波时间

范　围：设定值和返回值如下，默认值 0

0：5us

1：100us

2：200us

3：500us

例　程

BASE 0

PEL_FILTER =1;设置轴 0 的正方向硬限位信号滤波时间为 100us

2.11.21 NEL_FILTER

所　属：属性

语　法：NEL_FILTER = value

类　型：ULONG

描　述：设置/读取轴的负方向硬限位输入信号的滤波时间

范　围：设定值和返回值如下，默认值 0

0：5us

1：100us

2：200us

3：500us

例　程

BASE 0

NEL_FILTER =1;设置轴 0 的负方向硬限位信号滤波时间为 100us

2.11.22 SPEL

所　属：属性

语　法：SPEL = value

类　型：LONG

描　述：设置/读取正方向软限位的值，单位为脉冲

例　程

BASE 0

SPEL =100 '设置正方向软件限位的值为 100

2.11.23 SPEL_EN

所　属：属性

语　法：SPEL_EN = value

类　型：ULONG

描　述：启用/禁用正方向软限位功能，启用正方向软限位功能后，正向移动的电机指令位置到达 SPEL 设定的值后，马达会被控制停止运动。

范　围：设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

```
BASE 0
SPEL_EN =1   '启用轴 0 的正方向软限位功能
```

2.11.24 SPEL_MODE

所　属：属性

语　法：SPEL_MODE　= value

类　型：ULONG

描　述：设置/读取正方向软件限位功能被触发时电机被控制停止的模式

范　围：设定值和返回值如下，默认值 1

0：立即停止

1：减速停止

例　程

```
BASE 0
SPEL_MODE =1   '设置轴 0 的正方向软件限位被触发时，电机被控制的停止模式为减速停止
```

2.11.25 SNEL

所　属：属性

语　法：SNEL = value

类　型：LONG

描　述：设置/读取负方向软限位的值，单位为脉冲

例　程

```
BASE 0
SNEL =100 '设置负方向软件限位的值为 100
```

2.11.26 SNEL_EN

所　属：属性

语　法：SNEL_EN = value

类　型：ULONG

描　述：启用/禁用负方向软限位功能，启用负方向软限位功能后，负向移动的电机指令位置到达 SNEL 设定的值后，马达会被控制停止运动。

范　围:设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

BASE 0

SNEL_EN =1　'启用轴 0 的负方向软限位功能

2.11.27 SNEL_MODE

所　属：属性

语　法：SNEL_MODE　= value

类　型：ULONG

描　述：设置/读取负方向软件限位功能被触发时电机被控制停止的模式

范　围:设定值和返回值如下，默认值 1

0：立即停止

1：减速停止

例　程

BASE 0

SNEL_MODE =1　'设置轴 0 的负方向软件限位被触发时，电机被控制的停止模式为减速停止

2.11.28 PEL_TOL_EN

所　属：属性

语　法：PEL_TOL_EN = value

类　型：ULONG

描　述：　启用/禁用正方向硬极限容差功能。该功能仅在外部手轮操作时使用。当手轮控制电机运动时，碰到极限信号后，由于极限会限制某方向的运动，而且触碰极限会发生轴错误报警，导致手轮不能正常控制电机移出极限。该指令功能开启后，会允许在极限附近的某段范围，触碰极限不产生轴错误报警，使得手轮可以正常控制电机。

范　围:设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

BASE 0

PEL_TOL_EN =1 '启用正方向硬极限容差功能

2.11.29 PEL_TOL

所　属：属性

语　法：PEL_TOL = value

类　型：ULONG

描　述： 设置/读取轴的正方向硬极限容差值。

范　围:设定值和返回值为【0，2147483647】，默认值 5000

例　程

BASE 0

PEL_TOL =100 '设置轴 0 的正方向硬极限容差值为 100 个脉冲

2.11.30 NEL_TOL_EN

所　属：属性

语　法：NEL_TOL_EN = value

类　型：ULONG

描　述： 启用/禁用负方向硬极限容差功能。该功能仅在外部手轮操作时使用。当手轮控制电机运动时，碰到极限信号后，由于极限会限制某方向的运动，而且触碰极限会发生轴错误报警，导致手轮不能正常控制电机移出极限。该指令功能开启后，会允许在极限附近的某段范围，触碰极限不产生轴错误报警，使得手轮可以正常控制电机。

范　围:设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

BASE 0

NEL_TOL_EN =1 '启用负方向硬极限容差功能

2.11.31 NEL_TOL

所　属：属性

语　法：NEL_TOL = value

类　型：ULONG

描　述： 设置/读取轴的负方向硬极限容差值。

范　围：设定值和返回值为【0，2147483647】，默认值 5000

例　程

```
BASE 0
NEL_TOL =100 '设置轴 0 的负方向硬极限容差值为 100 个脉冲
```

2.11.32 SPEL_TOL_EN

所　属：属性

语　法：SPEL_TOL_EN = value

类　型：ULONG

描　述： 启用/禁用正方向软极限容差功能。该功能仅在外部手轮操作时使用。当手轮控制电机运动时，碰到极限信号后，由于极限会限制某方向的运动，而且触碰极限会发生轴错误报警，导致手轮不能正常控制电机移出极限。该指令功能开启后，会允许在极限附近的某段范围，触碰极限不产生轴错误报警，使得手轮可以正常控制电机。

范　围：设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

```
BASE 0
SPEL_TOL_EN =1 '启用正方向软极限容差功能
```

2.11.33 SPEL_TOL

所　属：属性

语　法：SPEL_TOL = value

类　型：ULONG

描　述： 设置/读取轴的正方向软极限容差值。

范　围：设定值和返回值为【0，2147483647】，默认值 5000

例　程

BASE 0

SPEL_TOL =100 '设置轴 0 的正方向软极限容差值为 100 个脉冲

2.11.34 SNEL_TOL_EN

所　属：属性

语　法：SNEL_TOL_EN = value

类　型：ULONG

描　述： 启用/禁用负方向软极限容差功能。该功能仅在外部手轮操作时使用。当手轮控制电机运动时，碰到极限信号后，由于极限会限制某方向的运动，而且触碰极限会发生轴错误报警，导致手轮不能正常控制电机移出极限。该指令功能开启后，会允许在极限附近的某段范围，触碰极限不产生轴错误报警，使得手轮可以正常控制电机。

范　围:设定值和返回值如下，默认值 0

0：禁用

1：启用

例　程

BASE 0

SNEL_TOL_EN =1 '启用负方向软极限容差功能

2.11.35 SNEL_TOL

所　属：属性

语　法：SNEL_TOL = value

类　型：ULONG

描　述： 设置/读取轴的负方向软极限容差值。

范　围:设定值和返回值为【0，2147483647】，默认值 5000

例　程

BASE 0

SNEL_TOL =100 '设置轴 0 的负方向软极限容差值为 100 个脉冲

2.12 JOG 与手轮

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.12.1	JOG_VL	JOG 运动低速段速度	√	√
2.12.2	JOG_VH	JOG 运动高速段速度	√	√
2.12.3	JOG_ACC	JOG 运动加速度	√	√
2.12.4	JOG_DEC	JOG 运动减速度	√	√
2.12.5	JOG_VLTIME	JOG 运动低段速度运行的时间	√	√
2.12.6	JOGP	正向软件 JOG 运动	√	×
2.12.7	JOGN	负向软件 JOG 运动	√	×
2.12.8	JOGON	使能外部驱动的 JOG 功能	√	×
2.12.9	JOGOFF	禁用外部驱动的 JOG 功能	√	×
2.12.10	MPGON	使能外部驱动的手轮功能	√	×
2.12.11	MPGOFF	禁用外部驱动的手轮功能	√	×
2.12.12	EXT_MODE	手轮模式外部驱动的脉冲输入模式	√	√
2.12.13	EXT_PULSE	手轮模式外部驱动时，每个手轮脉冲输入对应多少个指令脉冲输出值	√	√
2.12.14	EXT_SRC	外部驱动的信号接入哪个轴的外部驱动输入端口	√	√

2.12.1 JOG_VL

所　属：属性

语　法：JOG_VL = value

类　型：DOUBLE

描　述：设置/读取 JOG 运动的低速段速度，单位为脉冲当量/s。当 JOG_VLTIME 值不为 0 时，JOG_VL 将起作用。

范　围：（0, MAXVEL），默认值 2000

例　程

```
BASE 0
JOG_VL=1000   '设置軸 0 的 JOG 运动低速段速度为 1000 个脉冲当量/s
```

2.12.2 JOG_VH

所　属：属性

语　法：JOG_VH = value

类　型：DOUBLE

描　述：设置/读取 JOG 运动的高速段速度，单位为脉冲当量/s

范　围：（JOG_VL, MAXVEL），默认值 8000

例　程

```
BASE 0
JOG_VH=1000   '设置軸 0 的 JOG 运动高速段速度为 1000 个脉冲当量/s
```

2.12.3 JOG_ACC

所　属：属性

语　法：JOG_ACC = value

类　型：DOUBLE

描　述：设置/读取 JOG 运动的加速度，单位为脉冲当量/s^2

范　围：（0, MAXACC），默认值 10000

例　程

```
BASE 0
JOG_ACC=20000 '设置轴 0 的 JOG 运动加速度为 20000 个脉冲当量/s^2
```

2.12.4 JOG_DEC

所　属：属性

语　法：JOG_DEC = value

类　型：DOUBLE

描　述：设置/读取 JOG 运动的减速度，单位为脉冲当量/s^2

范　围：（0, MAXDEC），默认值 10000

例　程

```
BASE 0
JOG_DEC=20000 '设置轴 0 的 JOG 运动减速度为 20000 个脉冲当量/s^2
```

2.12.5 JOG_VLTIME

所　属：属性

语　法：JOG_VLTIME =value

类　型：ULONG

描　述：设置/读取 JOG 运动低段速度运行的时间，单位为 ms。研华规划的 JOG 运动分两段速度。JOG 指令下达后，先控制电机的速度为 JOG_VL，JOG_VL 运动 JOG_VLTIME 值的时间后，控制电机加速到 JOG_VH。如果 JOG_VLTIME 值设置为 0，JOG 指令下达后，直接控制电机加速到 JOG_VH，JOG_VL 将不起作用。

范　围：大于等于 0，默认值 5000

例　程

```
BASE 0
JOG_VLTIME=1000 '设置軸 0 的 JOG 运动高低速切换时间为 1000ms
```

2.12.6 JOGP

所　属：命令

语　法 1：JOGP

语　法 2：JOGP AX(axis no)

语　法 3：JOGP dir1[，dir2][，dir3]……

描　述：BASE 轴列表的轴或指定轴、方向，开始正向 JOG 运动。JOGP 分 3 种方法使用如下。

- ✧ 语法 1 用于对 BASE 轴列表的轴执行正向 JOG 运动
- ✧ 语法 2 用于指定某个轴执行正向 JOG 运动
- ✧ 语法 3 用于对 BASE 轴列表的轴，指定不同方向，执行 JOG 运动。Dir 为 0 时，方向与 JOGP 同向；dir 为 1 时，方向与 JOGP 反向

参　数：dir　0：正向；1：反向

axis no　轴号；　范围：根据控制器实际硬件决定。

例　程

```
BASE 0,1,2
JOG_VL=500
JOG_VH=10000
JOG_ACC=50000
JOG_DEC=50000
JOG_VLTIME=2000      '设定低段速度运行的时间为 2 秒
SLEEP 5000
STOPDEC
JOGP            '轴 0、1、2 都执行正向 JOG 运动
SLEEP 3000
STOPDEC      JOGP AX(1)          '指定轴 1 执行正向 JOG 运动 ,轴速度会先加速到 JOG_VL 运行 2 秒，
再加速到 JOG_VH

WAIT DONE
JOGP 0,1,0      '轴 0,2 执行正向 JOG 运动，轴 1 执行负向 JOG 运动
SLEEP 4000
STOPDEC
```

2.12.7 JOGN

所　属：命令

语　法 1：JOGN

语　法 2：JOGN AX(axis no)

语　法 3：JOGN dir1[，dir2][，dir3]……

描　述：BASE 轴列表的轴或指定轴、方向，开始负向 JOG 运动。JOGN 分 3 种方法使用如下。

- ✧ 语法 1 用于对 BASE 轴列表的轴执行负向 JOG 运动
- ✧ 语法 2 用于指定某个轴执行负向 JOG 运动
- ✧ 语法 3 用于对 BASE 轴列表的轴，指定不同方向，执行 JOG 运动。Dir 为 0 时，方向与 JOGN 同向；dir 为 1 时，方向与 JOGN 反向

参　数：dir　0：反向；1：正向

axis no　轴号； 范围：根据控制器实际硬件决定。

例　程

```
BASE 0,1,2
JOG_VL=500
JOG_VH=10000
JOG_ACC=50000
JOG_DEC=50000
JOG_VLTIME=2000      '设定低段速度运行的时间为 2 秒
JOGN AX(1)    '指定轴 1 执行负向 JOG 运动，轴速度会先加速到 JOG_VL 运行 2 秒，再加速到 JOG_VH
SLEEP 5000
STOPDEC
JOGN          '轴 0、1、2 都执行负向 JOG 运动
SLEEP 3000
STOPDEC
WAIT DONE
JOGN 0,1,0     '轴 0,2 执行负向 JOG 运动，轴 1 执行正向 JOG 运动
SLEEP 4000
STOPDEC
```

2.12.8 JOGON

所　属：命令

语　法：JOGON

描　述：BASE 轴列表的第一个轴，使能外部驱动的 JOG 功能。该指令对外部硬件接线控制的 JOG 运动起作用。研华运动控制的每个轴都关联着 4 个 DI 端口，分别称为 IN1，IN2，IN4，IN5。当使用外部驱动的 JOG 功能时，IN4 和 IN5 分别控制 JOG+和 JOG-。

2.12.9 JOGOFF

所　属：命令

语　法：JOGOFF

描　述：BASE 轴列表的第一个轴，禁用外部驱动的 JOG 功能。该指令对外部硬件接线

控制的 JOG 运动起作用。研华运动控制的每个轴都关联着 4 个 DI 端口，分别称为 IN1，IN2，IN4，IN5。当使用外部驱动的 JOG 功能时，IN4 和 IN5 分别控制 JOG+和 JOG-。

2.12.10 MPGON

所　属：命令

语　法：MPGON

描　述：BASE 轴列表的第一个轴，使能外部驱动的 MPG 功能。该指令对外部硬件接线控制的 MPG 运动起作用。研华运动控制的每个轴都关联着 4 个 DI 端口，分别称为 IN1，IN2，IN4，IN5。当使用外部驱动的 MPG 功能时，IN4 和 IN5 分别控制对应手轮脉冲输入的 A 相和 B 相。

2.12.11 MPGOFF

所　属：命令

语　法：MPGOFF

描　述：BASE 轴列表的第一个轴，禁用外部驱动的 MPG 功能。该指令对外部硬件接线控制的 MPG 运动起作用。研华运动控制的每个轴都关联着 4 个 DI 端口，分别称为 IN1，IN2，IN4，IN5。当使用外部驱动的 MPG 功能时，IN4 和 IN5 分别控制对应手轮脉冲输入的 A 相和 B 相。

2.12.12 EXT_MODE

所　属：属性

语　法：EXT_MODE = value

类　型：ULONG

描　述：设置/读取手轮模式外部驱动的脉冲输入模式

范　围：设定值和返回值如下，默认值 2

0：1XAB

1：2XAB

2：4XAB

3：CCW/CW

例　程

```
EXT_MODE =1    '设置手轮外部驱动的脉冲输入模式为 2XAB
```

2.12.13 EXT_PULSE

所　属：属性

语　法：EXT_PULSE = value

类　型：ULONG

描　述：设置/读取手轮模式外部驱动时，每个手轮脉冲输入对应多少个指令脉冲输出值

范　围：【1,1000】；默认值为 1

例　程

```
EXT_PULSE =2    '设置手轮脉冲输入对应 2 个指令脉冲输出
```

2.12.14 EXT_SRC

所　属：属性

语　法：EXT_SRC = value

类　型：ULONG

描　述：设置/读取外部驱动的信号接入哪个轴的外部驱动输入端口

范　围：设定值和返回值如下，默认值 0

0： 0 轴

1： 1 轴(暂不支持)

2： 2 轴(暂不支持)

3： 3 轴(暂不支持)

例　程

```
BASE 0
EXT_SRC =0    '设置外部驱动信号接到轴 0 的外部驱动端口
```

2.13 通信指令

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.13.1	COM_OPEN	打开串口	×	×
2.13.2	COM_CLOSE	关闭串口	×	×
2.13.3	COM_SET	设置串口通讯参数	×	×
2.13.4	COM_ReadStream	串口自由协议读操作，通过串口读数据	×	×
2.13.5	COM_WriteStream	串口自由协议写操作，通过串口写数据	×	×
2.13.6	COM_ResetBuf	清除串口缓存区数据	×	×
2.13.7	TCP_OPEN	打开一个 TCP 通讯连接	×	×
2.13.8	TCP_CLOSE	关闭一个 TCP 通讯连接	×	×
2.13.9	TCP_STATUS	检查 TCP 连接状态	×	×
2.13.10	TCP_WAIT	等待 TCP 连接完成	×	×
2.13.11	TCP_Check	获取 TCP 通信接收到的字符个数	×	×
2.13.12	TCP_ ReadSTR	控制器接收字符串指令	×	×
2.13.13	TCP_WriteSTR	控制器发送字符串指令	×	×
2.13.14	TCP_Read	控制器接收数据	×	×
2.13.15	TCP_Write	控制器发送数据	×	×
2.13.16	TCP_ReadVR	控制器用 VR 变量接收数据	×	×
2.13.17	TCP_WriteVR	控制器把 VR 变量中的数据发送出去	×	×
2.13.18	TCP_ResetBuf	清除 TCP 缓存区数据	×	×

2.13.1 COM_OPEN

所　属：命令

语　法：COM_OPEN port

描　述：指定串口编号，打开串口。相应串口端口被打开后，才可以对该串口操作。该指令需要根据本地串口资源进行操作。

参　数：port　串口端口号

注　意：打开串口操作仅适用于未打开的串口，如果串口资源已经被打开，下该指令操会执行不成功，并返回错误。

例　程

```
COM_OPEN 2              '打开串口 2
COM_SET 2, 9600, 0, 1, 8  '设置串口波特率 9600，校验位无，停止位 1 位，数据位 8 位
COM_CLOSE 2             '关闭串口 2
```

2.13.2 COM_CLOSE

所　属：命令

语　法：COM_ClOSE port

描　述：指定串口编号，关闭串口。

参　数：port　串口端口号

例　程

```
COM_OPEN 2              '打开串口 2
COM_SET 2, 9600, 0, 1, 8  '设置串口波特率 9600，校验位无，停止位 1 位，数据位 8 位
COM_CLOSE 2             '关闭串口 2
```

2.13.3 COM_SET

所　属：命令

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

描　述：设置串口通讯参数。

参　数：port　　串口端口号；

Baudrate　波特率；　范围：4800、9600、19200、38400、57600、115200

Parity　　校验方式；　范围：无（NONE）、奇（ODD）、偶（EVEN）

Stopbits　停止位；　范围：1、2

Databits　数据位；　范围：7、8

例　程

```
COM_OPEN 2              '打开串口 2
COM_SET 2, 9600, 0, 1, 8  '设置串口波特率 9600，校验位无，停止位 1 位，数据位 8 位
```

```
COM_CLOSE 2                '关闭串口 2
```

2.13.4 COM_ReadStream

所　属：命令

语　法：COM_READSTREAM port, *strarray, num

描　述：串口自由协议读操作，通过串口读数据。执行到该指令时，控制器程序会等在该行，直到读到的字节个数和 num 参数指定的个数一致时，程序才会执行到下一行。

参　数：port　　串口端口号；

*strarray　存放读到的数据变量地址，一般为数组的地址或字符串地址

num　　读取的字节个数或字符个数

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   '打开串口 2
COM_SET 2,9600,0,1,8 '设置串口波特率 9600，校验位无，停止位 1 位，数据位 8 位
BASE 0,1
SVON
MOVE 2000,30000
WAIT DONE
COM_WriteStream 2,WriteArray(),2 '控制器发出数组 WriteArray()里的 2 个字节数据
COM_WriteStream 2,WriteStr,2      '控制器写出 WriteStr 中字符串
COM_ReadStream 2,ReadArray(),2   '控制器读两个字节数据，未读到 2 个字节，程序会停在该行读
IF(ReadArray(1)=8) THEN     '判断读到的 ReadArray(1)值是否为 8
MOVEABS 0,0
END IF
COM_Close 2                '关闭串口 2
```

2.13.5 COM_WriteStream

所　属：命令

语　法：COM_WriteStream　port, *strarray, num

描　述：串口自由协议写操作，通过串口写数据。

参　数：port　　串口端口号；

*strarray　存放写出的数据变量地址，一般为数组的地址或字符串地址

num　　写出的字节个数或字符个数

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   '打开串口 2
COM_SET 2,9600,0,1,8 '设置串口波特率 9600，校验位无，停止位 1 位，数据位 8 位
BASE 0,1
SVON
MOVE 2000,30000
WAIT DONE
COM_WriteStream 2,WriteArray(),2 '控制器发出数组 WriteArray()里的 2 个字节数据
COM_WriteStream 2,WriteStr,2     '控制器写出 WriteStr 中字符串
COM_ReadStream 2,ReadArray(),2   '控制器读两个字节数据，未读到 2 个字节，程序会停在该行读
IF(ReadArray(1)=8) THEN    '判断读到的 ReadArray(1)值是否为 8
MOVEABS 0,0
END IF
COM_Close 2              '关闭串口 2
```

2.13.6 COM_ResetBuf

所　属：命令

语　法：COM_ResetBuf　port

描　述：清除串口缓存区数据。

参　数：port　　串口端口号

2.13.7 TCP_OPEN

所　属：命令

语　法：TCP_OPEN　no，mode，port[，ipaddress]

描　述：指定 TCP/IP 通讯编号、模式、网络端口号[、IP 地址]，打开一个 TCP 通信连接。相应 TCP 通信端口被打开后，才可以对该网口操作。该指令需要根据本地网口资源进行操作。

参　数：no　TCP 通讯编号。用于控制器内部识别不同 TCP/IP 连接。类型为 ULONG，没用过的编号可以随意指定，比如 0,1,2,3,4,5.....

mode　连接模式；范围：0：控制器作为服务器，1：控制器作为客户端。

Port　网络端口号

ipaddress　：IP 地址，控制器作为服务器时，不需要填该参数。控制器作为客户端时，该参数填服务器端网口 IP 地址

例　程

```
TCP_Open 2,1,5024,"192.168.0.11"   '打开一个 TCP 客户端连接，对接 IP 为 192.168.0.11 的服务器
TCP_Close 2                        '关闭编号为 2 的网络客户端
TCP_Open(1,0,5025)                 '打开一个 TCP 服务器连接，服务器处于监听状态
TCP_Close 1                        '关闭编号为 1 的网络服务器
```

2.13.8 TCP_CLOSE

所　属：命令

语　法：TCP_ClOSE no

描　述：指定 TCP 通讯编号，关闭对应 TCP 通信端口

参　数：no　TCP 通讯编号；类型：ULONG

例　程

```
TCP_Open 2,1,5024,"192.168.0.11"   '打开一个 TCP 客户端连接，对接 IP 为 192.168.0.11 的服务器
TCP_Close 2                        '关闭编号为 2 的网络客户端
TCP_Open(1,0,5025)                 '打开一个 TCP 服务器连接，服务器处于监听状态
TCP_Close 1                        '关闭编号为 1 的网络服务器
```

2.13.9 TCP_STATUS

所　属：命令

语　法：value=TCP_STATUS (no)

类　型：ULONG

描　述：检查 TCP 通讯连接状态

参　数：no　　TCP 通讯编号

返回值：0：连接不成功；1：连接成功

例　程

'请参考 TCP_ReadSTR 或 TCP_WriteSTR 指令

2.13.10 TCP_WAIT

所　属：命令

语　法：TCP_WAIT no [,timeout]

描　述：等待 TCP 连接完成。执行该指令时，程序会等待在该行直到 TCP 通讯连接成功或 timeout 超时，程序才会继续下一行的执行。

参　数：no　　TCP 通讯编号；

timeout　等待超时时间，单位为 ms。Timeout 时间到后，TCP 通讯连接还未成功，程序会继续下一行的执行。

例　程

'请参考 TCP_ReadSTR 或 TCP_WriteSTR 指令

2.13.11 TCP_Check

所　属：命令

语　法：value=TCP_Check(no)

描　述：获取 TCP 通讯接收到的字符串字符个数。

参　数：no　　TCP 通讯编号；

返回值：字符个数；类型：ULONG

例　程

'请参考 TCP_ReadSTR 或 TCP_WriteSTR 指令

2.13.12 TCP_ReadSTR

所　属：命令

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

描　述：TCP/IP 自由协议读操作，控制器接收字符串指令。执行到该指令时，程序会等在该指令行 ,直到接收到字符或 timeout 超时 ,程序才会继续下一行的执行。

参　数：no　　TCP 通讯编号；类型：ULONG

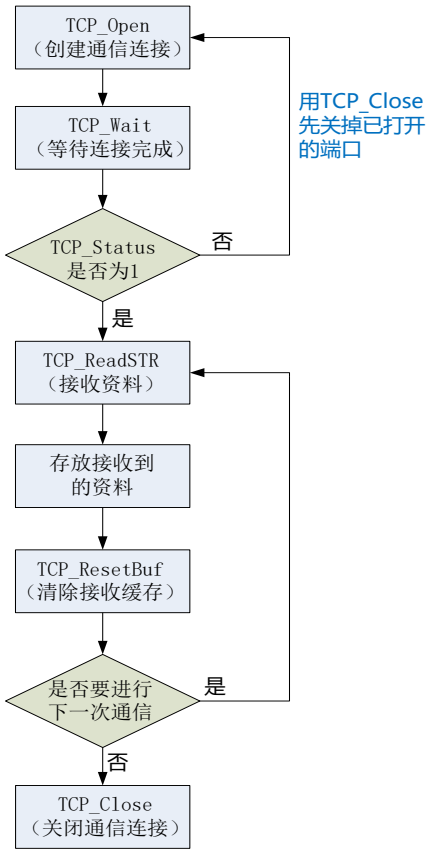
strData　存放接收的字符串；类型：String

numChars　接收的字符个数；类型：ULONG。接收字符个数可由 TCP_check 指令获取得到。

StrEnd　接收字符串的结束符。类型：String 该参数不填时，控制器会根据参数 numChars 来确定接收多少个字符。指定结束符时，控制器接收到结束符就会停止这一次接收。

Timeout　接收的超时时间，单位为 ms。Timeout 时间到后，还未接收到字符，系统会断开当前通讯编号的 TCP 连接。如还需要进行通讯，需重新创建 TCP 连接。

注　意：TCP 接收的相关指令，如 TCP_ReadSTR、TCP_Read、TCP_ReadVR 要注意通信接收缓存的处理。为避免通信接收缓存中的遗留数据影响新的接收资料，接收指令前需用 TCP_ResetBuf 清除通信缓存，不然接收指令会先收到遗留在通信缓存中的资料，导致接收的资料不对。TCP 接收指令的操作可以参照以下流程图处理。



例　程

```
Dim NumChars as ULONG = 0
Dim StrData as string
TCP_Open (0, 1, 8080, "127.0.0.1")      '创建一个客户端连接，对接 IP 为"127.0.0.1"的服务器
TCP_Wait 0                              '等待连接完成
If TCP_STATUS(0) > 0 Then               '确认通讯编号为 0 的连接是否连接成功
    TCP_WriteSTR 0,"I'm ready"          '连接成功后，控制器发送出 I'm ready
    WHILE(1)
        NumChars = TCP_Check(0)                     '取接收到字符个数
        If NumChars > 0 Then
            Print "charNum = ", NumChars            '打印出接收到的字符个数
            TCP_ReadSTR(0, StrData, NumChars)   '将接收到字符放入 StrData
            Print StrData                           '打印出接到到的字符串
            If(StrData="ok") THEN           '确认接收到的字符串是否是 ok
                EXIT WHILE                  'ok：退出接收字符；不是 ok：继续接收字符
            END If
        End If
        SLEEP 10
    WEND
End If
TCP_CLOSE 0             '断开通讯编号为 0 的 TCP 连接
```

2.13.13 TCP_WriteSTR

所　属：命令

语　法：TCP_WriteSTR　no, strData

描　述：TCP/IP 自由协议写操作，控制器发送出字符串指令

参　数：no　　TCP 通讯编号；

　　strData　　发送出去的字符串；类型：String

例　程

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

```
Dim StrData as string
TCP_Open (0, 1, 8080, "127.0.0.1")      '创建一个客户端连接，对接 IP 为"127.0.0.1"的服务器
TCP_Wait 0                              '等待连接完成
If TCP_STATUS(0) > 0 Then               '确认通讯编号为 0 的连接是否连接成功
    TCP_WriteSTR 0,"I'm ready"          '连接成功后，控制器发送出 I'm ready
    WHILE(1)
        NumChars = TCP_Check(0)                     '取接收到字符个数
        If NumChars > 0 Then
            Print "charNum = ", NumChars           '打印出接收到的字符个数
            TCP_ReadSTR(0, StrData, NumChars)   '将接收到字符放入 StrData
            Print StrData                           '打印出接到到的字符串
            If(StrData="ok") THEN           '确认接收到的字符串是否是 ok
                EXIT WHILE                  'ok：退出接收字符；不是 ok：继续接收字符
            END If
        End If
        SLEEP 10
    WEND
End If
TCP_CLOSE 0             '断开通讯编号为 0 的 TCP 连接
```

2.13.14 TCP_Read

所　属：命令

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

描　述：TCP/IP 自由协议读操作，控制器接收数据。控制器会根据 array( )定义的数据类型（byte 或 short 或 long），将每接收到的数据按 1 个字节或 2 个字节或 4 个字节为一个数据依次放入 array( )中。执行到该指令时，程序会等在该指令行，直到接收到数据或 timeout 超时，程序才会继续下一行的执行。

参　数：no　　TCP 通讯编号；类型：ULONG

Array( )　存放接收数据的数组；类型：byte、short、long

arrayCnt　接收的数据个数；类型：ULONG。

Timeout　接收的超时时间，单位为 ms。Timeout 时间到后，还未接收到数据，系统会断开当前通讯编号的 TCP 连接。如还需要进行通讯，需重新创建 TCP 连接。

注　意：TCP 接收的相关指令，如 TCP_ReadSTR、TCP_Read、TCP_ReadVR 要注意通信接收缓存的处理。为避免通信接收缓存中的遗留数据影响新的接收资料，接收指令前需用 TCP_ResetBuf 清除通信缓存，不然接收指令会先收到遗留在通信缓存中的资料，导致接收的资料不对。TCP 接收指令的操作可以参照 TCP_ReadSTR 指令说明中的流程图来处理。

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")      '创建一个客户端连接，对接 IP 为"127.0.0.1"的服务器
TCP_Wait 0                              '等待连接完成
If TCP_STATUS(0) > 0 Then               '确认通讯编号为 0 的连接是否连接成功
    '接收到的 2 个 Byte 数据分别存入 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 个 Byte 数据组成 Short 类型数据存入 R_ShortArray(0)
    '接收到的后 2 个 Byte 数据组成 short 类型数据存入 R_ShortArray(1)
    TCP_READ 0,R_ShortArray(),2
    PRINT R_ShortArray(0),R_ShortArray(1)
    '接收到的前 4 个 Byte 数据组成 long 类型数据存入 R_LongArray(0)
    '接收到的后 4 个 Byte 数据组成 long 类型数据存入 R_LongArray(1)
    TCP_READ 0,R_LongArray(),2
    PRINT R_LongArray(0),R_LongArray(1)
End If
TCP_CLOSE 0             '断开通讯编号为 0 的 TCP 连接
```

2.13.15 TCP_Write

所　属：命令

语　法：TCP_Write　no, array( ),arrayCnt

描　述：TCP/IP 自由协议写操作 ,控制器发送出数据。发送的数据可以选择 byte、short、long 三种数据类型。控制器会根据数据类型按 byte 的数据送出。如果是 short、long 类型，发出去的 byte 会以低字节到高字节的顺序发送。

参　数：no　　TCP 通讯编号；

Array( )　　发送出去数据的数组；类型：byte，short，long

ArrayCnt　　发送出去的数据个数。

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") '创建一个客户端连接，对接 IP 为"127.0.0.1"的服务器
TCP_Wait 0                          '等待连接完成

'相隔 3 秒，依次发送 W_ByteArray(),W_ShortArray(),W_LongArray()
If TCP_STATUS(0) > 0 Then           '确认通讯编号为 0 的连接是否连接成功
    '服务器端收到的数据为十六进制数：B5 79。对应-75,121
    TCP_Write 0,W_ByteArray(),2
    SLEEP 3000
    '服务器端收到的数据为十六进制数：87 00 F1 7F。对应 135,32753
    TCP_Write 0,W_ShortArray(),2
    SLEEP 3000
    '服务器端收到的数据为十六进制数：E8 FF FF FF B0 AB 02 00。对应-24,175024
    TCP_Write 0,W_LongArray(),2
End If
TCP_CLOSE 0             '断开通讯编号为 0 的 TCP 连接
```

2.13.16 TCP_ReadVR

所　属：命令

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

描　述：TCP/IP 自由协议读操作，控制器用 VR 变量接收数据。控制器会根据 format 指定的数据类型（byte 或 short 或 long），将每接收到的数据按 1 个字节或 2 个字节或 4 个字节为一个数据依次放入 VR 变量中。执行到该指令时，程序会等在该指令行，直到接收到数据或 timeout 超时，程序才会继续下一行的执行。

参　数：

no	TCP 通讯编号；类型：ULONG
VR_StartIndex	存放接收数据的起始 VR；类型：byte、short、long
VRCnt	接收的数据个数；类型：ULONG
format	指定存放接收数据的类型。 0：byte 1：short 2：long
timeout	接收的超时时间，单位为 ms。Timeout 时间到后，还未接收到数据，系统会断开当前通讯编号的 TCP 连接。如还需要进行通讯，需重新创建 TCP 连接。

注　意：TCP 接收的相关指令，如 TCP_ReadSTR、TCP_Read、TCP_ReadVR 要注意通信接收缓存的处理。为避免通信接收缓存中的遗留数据影响新的接收资料，接收指令前需用 TCP_ResetBuf 清除通信缓存，不然接收指令会先收到遗留在通信缓存中的资料，导致接收的资料不对。TCP 接收指令的操作可以参照 TCP_ReadSTR 指令说明中的流程图来处理。

例　程

```
TCP_Open (0, 1, 8080, "127.0.0.1") '创建一个客户端连接，对接 IP 为"127.0.0.1"的服务器
TCP_Wait 0                          '等待连接完成
If TCP_STATUS(0) > 0 Then           '确认通讯编号为 0 的连接是否连接成功

    '接收到的 2 个 Byte 数据分别存入 VR(0)，VR(1)
    TCP_ReadVR 0,0,2,0
    PRINT VR(0),VR(1)

    '接收到的前 2 个 Byte 数据组成 Short 类型数据存入 VR(2)
    '接收到的后 2 个 Byte 数据组成 short 类型数据存入 VR(3)
```

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

    '接收到的前 4 个 Byte 数据组成 long 类型数据存入 VR(4)
    '接收到的后 4 个 Byte 数据组成 long 类型数据存入 VR(5)
    TCP_ReadVR 0,4,2,2
    PRINT VR(4),VR(5)
End If
TCP_CLOSE 0              '断开通讯编号为 0 的 TCP 连接
```

2.13.17 TCP_WriteVR

所　属：命令

语　法：TCP_WriteVR　no, VR_StartIndex ,VRCnt, format

描　述：TCP/IP 自由协议写操作，控制器把 VR 变量中的数据发送出去。发送的数据可以选择 byte、short、long 三种数据类型。控制器会根据数据类型按 byte 的数据送出。

参　数：no　　TCP 通讯编号；

VR_StartIndex　发送数据的起始 VR；类型：byte、short、long

VRCnt　　发送的数据个数；类型：ULONG

format　　指定发送数据的类型。

0：byte

1：short

2：long

注　意　如果发送出去的 VR 变量数据是浮点型数据，接收端接收到的数据会失真。如要发送浮点数据，请用 TCP_WriteSTR 指令，用字符串形式发送出去，接收端接收到字符串后再转数据类型来接收浮点数据。

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") '创建一个客户端连接，对接 IP 为"127.0.0.1"的服务器
TCP_Wait 0                          '等待连接完成

'相隔 3 秒，依次发送 VR(0)、VR(1) ; VR(2)、VR(3) ; VR(4)、VR(5)
If TCP_STATUS(0) > 0 Then           '确认通讯编号为 0 的连接是否连接成功
'将 VR(0),VR(1)发送出去，服务器端收到的数据为十六进制数：B5 79。对应-75,121
    TCP_WriteVR 0,0,2,0
    SLEEP 3000
'将 VR(2),VR(3)发送出去，服务器端收到的数据为十六进制数：87 00 F1 7F。对应 135,32753
    TCP_WriteVR 0,2,2,1
    SLEEP 3000
'将 VR(4),VR(5)发送出去 ,服务器端收到的数据为十六进制数 :E8 FF FF FF B0 AB 02 00。对应-24,175024
    TCP_WriteVR 0,4,2,2
End If
TCP_CLOSE 0             '断开通讯编号为 0 的 TCP 连接
```

2.13.18 TCP_ResetBuf

所　属：命令

语　法：TCP_ResetBuf　no

描　述：清除 TCP 缓存区数据。

参　数：no　TCP 通讯连接编号

注　意：TCP 接收的相关指令，如 TCP_ReadSTR、TCP_Read、TCP_ReadVR 要注意通信接收缓存的处理。为避免通信接收缓存中的遗留数据影响新的接收资料，接收指令前需用 TCP_ResetBuf 清除通信缓存，不然接收指令会先收到遗留在通信缓存中的资料，导致接收的资料不对。

2.14 字符串处理

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.14.1	ASC	返回字符串中字符的 ASCII 码	×	×
2.14.2	CHR	返回用 ASCII 码表达的值对应的字符	×	×
2.14.3	HEX	返回数值的十六进制结果	×	×
2.14.4	INSTR	查找字符串中第一次出现的字符或者字串	×	×
2.14.5	LCASE	将字符串中的字母全部转变成小写字母返回	×	×
2.14.6	LEFT	返回字符串从左开始指定字符个数的子串	×	×
2.14.7	LEN	返回字符串的长度（字符个数）或者数据类型的长度（字节数）	×	×
2.14.8	MID	返回一个字符串的子字符串	×	×
2.14.9	RIGHT	返回字符串从右开始指定字符个数的子串	×	×
2.14.10	STR	将一个数转换成字符串	×	×
2.14.11	UCASE	将字符串中的字母全部转变成大写字母返回	×	×
2.14.12	VAL	将字符串转换成一个数值	×	×
2.14.13	PARSESTR	按用户指定的分隔符解析字符串	×	×

2.14.1 ASC

语　法：value=ASC(string [,position])

描　述：返回字符串中字符的 ASCII 码

参　数：string　　字符串

position　需返回 ASCII 码字符在字符串中的位置，缺省值为 1

例　程

```
PRINT ASC("A")          '结果为 65
PRINT ASC("ABC",1)      '打印第一个字母 A 的 ASCII 码，结果为 65
PRINT ASC("ABC",2)      '打印第二个字母 B 的 ASCII 码，结果为 66
PRINT ASC("ABC",3)      '打印第三个字母 C 的 ASCII 码，结果为 65
PRINT ASC("ABC")        '缺省位置值为 1，即打印 A 的 ASCII 码，结果为 65
```

2.14.2　CHR

语　法：value=CHR(number)

描　述：返回用 ASCII 码表达的值对应的字符

参　数：number　ASCII 码值

例　程

```
PRINT CHR(97)           '97 对应的字符为 a，打印结果为 a
PRINT CHR(65)           '65 对应的字符为 A，打印结果为 A
```

ASCII 码表

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

语　法：value=HEX(number [,digits])

描　述：返回数值的十六进制结果

参　数：number　数值

digits　返回由低位到高位的位数

例　程

```
'十进制 54321 对应的十六进制数为 D431
Print Hex(54321)          '打印结果为 D431
Print Hex(54321, 2)       '打印结果为 31
Print Hex(54321, 5)       '打印结果为 0D431
```

2.14.4 INSTR

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

描　述：查找字符串中第一次出现的字符或者字串

参　数：start　　从第几个字符开始查找

string　在 string 这个字符串中查找字符或字串

Any　　加上这个关键字后，string 中先找到 substring 中的任意一个字符就会返回相应值

substring　需查找的字符或字串

例　程

```
Print InStr(2,"abcdefg", "a")　'打印信息为 0，因从字符串的第 2 位开始找，找不到 a，返回 0
Print InStr("abcdefg", "de")　'打印信息为 4，第 4 位找到 de
Print InStr("abcdefg", "h")　'打印信息为 0，字符串中没有 h
Print InStr("abcdefg", Any "fbc")　'打印信息为 2，因加了 any 关键字，所以先找到 b，b 为第 2 位
```

2.14.5 LCASE

语　法：value=LCASE(string)

描　述：将字符串中的字母全部转变成小写字母返回

参　数：string　需要转换的字符串

例　程

```
Print Lcase("AeeE")　'打印结果为 aeee
```

2.14.6 LEFT

语　法：value=LEFT(string,number)

描　述：返回字符串从左开始指定字符个数的子串

参　数：string　　需要转换的字符串

number　字符个数

例　程

```
Print LEFT("Hello Advantech",5)　　'打印信息为 Hello
```

2.14.7 LEN

语　法：value=LEN(expression)

描　述：返回字符串的长度（字符个数）或者数据类型的长度（字节数）

参　数：expression　如果是字符串，返回字符个数；如果是数据类型，返回字节数

例　程

```
Print Len("hello world") '打印结果为 11，共 11 个字符
Print Len(Integer)    '打印结果为 4，integer 这个数据类型为 4 个字节
```

2.14.8 MID

语　法：value=MID(string，start [,number])

描　述：返回一个字符串的子字符串

参　数：string　需要转换的字符串

start　返回的子字符串的起始转换位

number　子字符串的字符个数。如不填，则返回从 start 位后的所有字符

例　程

```
Print Mid("abcdefg", 3, 2)     '打印结果为 cd
Print Mid("abcdefg", 3)        '打印结果为 cdefg
Print Mid("abcdefg", 2, 1)     '打印结果为 b
```

2.14.9 RIGHT

语　法：value=RIGHT(string,number)

描　述：返回字符串从右开始指定字符个数的子串

参　数：string　需要转换的字符串

number　字符个数

例　程

```
Print RIGHT("Hello Advantech",9)        '打印信息为 Advantech
```

2.14.10 STR

语　法：value=STR(Numeric)

描　述：将一个数转换成字符串

参　数：Numeric　数值表达式

例　程

```
VR(100)=100.32
PRINT STR(VR(100))      '打印结果为字符串"100.32"
```

2.14.11 UCASE

语　法：value=UCASE(string)

描　述：将字符串中的字母全部转变成大写字母返回

参　数：string　需要转换的字符串

例　程

```
Print Ucase("AeeE")      '打印结果为 AEEE
```

2.14.12 VAL

语　法：value=VAL(string)

描　述：将字符串转换成一个数值，字符串转换将从左到右按字符转换，如果先遇到非数值的字符，转换出来的数值将是 0。

参　数：string　字符串

例　程

```
DIM AS STRING str1,str2
str1="e3t"                    '因先遇到非数值字符 e，所有打印结果为 0
str2="325.32"
PRINT VAL(str1),VAL(str2)     '打印结果为 0,325.32
```

2.14.13 PARSESTR

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

描　述：按用户指定的分隔符解析字符串。

参　数：StrInput　　输入的需分隔的字符串

StrTokens( )　存放分隔出的有效字符串数组

StrDelimits　指定的分隔符

返回值：NumStr　　分隔出的有效字符串个数。类型：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        '打印出 num=4，有效分隔出 4 个字串
Dim i as Integer = 0
for i= 0 to (NumStr-1)
    print   StrTokens(i)      '字符串数组依次打印出 Hi MAS Controller !
next i
```

2.15 工艺模块指令

2.15.1 汽油缸控制

汽油缸在自动化设备中非常常见，很好的对汽油缸进行控制在系统开发中显得很重要。本章节介绍了 Motion Basic 简单易使用的汽油缸控制指令，通过简单配置，可以很方便的实现设备中常见的汽油缸控制。为简要说明，本章节指令说明中统一用“气缸”来代替“汽油缸”，“用气缸前进”、“气缸后退来”表示气缸动作的两个方向运动。

本节指令概览

章节	指令	说明	终端工具	观察变量工具
2.15.1.1	CYL_BASE	该指令后面所有的气缸指令和参数设置、读取都基于该指令选定的气缸	√	×
2.15.1.2	CYL_FwDoneType	气缸前进到位方式	√	×
2.15.1.3	CYL_BwDoneType	气缸后退到位方式	√	×
2.15.1.4	CYL_FwTime	CYL_FwDoneType 中涉及到延时到位方式的延时时间	√	×
2.15.1.5	CYL_BwTime	CYL_BwDoneType 中涉及到延时到位方式的延时时间	√	×
2.15.1.6	CYL_FwAlmTime	气缸前进开始到到位的最大时间	√	×
2.15.1.7	CYL_BwAlmTime	气缸后退开始到到位的最大时间	√	×
2.15.1.8	CYL_FwEncValue	气缸前进到位编码器值	√	×
2.15.1.9	CYL_BwEncValue	气缸后退到位编码器值	√	×
2.15.1.10	CYL_Status	气缸当前状态	√	×
2.15.1.11	CYL_AlmReset	复位气缸的状态到复位状态	√	×
2.15.1.12	CYL_Move	执行气缸前进或后退动作	√	×
2.15.1.13	CYL_Stop	停止气缸动作	√	×

2.15.1.1 CYL_BASE

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

描　述：为了简化编程，可以用该指令选择要参与运动的气缸号，其后的指令就没必要填写所有气缸控制的参数，只填写参与运动的气缸参数即可。气缸号要按顺序填写，气缸号可以是 1 个，也可以是 2 个、3 个...

参　数： cyl_no 汽缸号，由硬件配置决定对应的实体汽缸控制；范围：根据控制器实际硬件决定

例　程

```
CYL_BASE 0,1,2,3
Cyl_FwDoneType=0          '设置 4 个气缸前进到位方式为延时到位
Cyl_BwDoneType=1          '设置 4 个气缸后退到位方式为限位到位
CYL_MOVE 1,0,1,1          '气缸 0,1,2,3 分别执行前进、后退、前进、前进动作
Wait CYLDONE              '等待气缸 0,1,2,3 动作到位完成
CYL_BASE 1
CYL_MOVE 1                '气缸 1 执行前进动作
```

2.15.1.2 CYL_FwDoneType

语　法：CYL_FwDoneType= value

类　型：ULONG

描　述：设置/读取气缸前进到位方式

范　围:如下设定值，默认值 0

0：延时到位：气缸动作后，延时指定时间到，即认为气缸动作到位

1：限位到位：气缸动作后，遇到指定限位有效，即认为气缸动作到位

2：（限位+延时）到位：气缸动作后，遇到指定限位有效，再延时指定时间后，即认为气缸动作到位

3：（延时+限位）到位：气缸动作后，延时指定时间后，再检测到指定限位有效，即认为气缸动作到位

4： 编码器到位：气缸动作后，编码器到限定数值后，即认为气功动作到位

例　程

```
CYL_BASE 0
CYL_FwDoneType=0 '设置气缸 0 前进到位方式为延时到位
```

2.15.1.3 CYL_BwDoneType

语　法：CYL_BwDoneType= value

类　型：ULONG

描　述：设置/读取气缸后退到位方式

范　围:如下设定值，默认值 0

0：延时到位：气缸动作后，延时指定时间到，即认为气缸动作到位

1：限位到位：气缸动作后，遇到指定限位有效，即认为气缸动作到位

2：（限位+延时）到位：气缸动作后，遇到指定限位有效，再延时指定时间后，即认为气缸动作到位

3：（延时+限位）到位：气缸动作后，延时指定时间后，再检测到指定限位有效，即认为气缸动作到位

4： 编码器到位：气缸动作后，编码器到限定数值后，即认为气功动作到位

例　程

```
CYL_BASE 0
CYL_BwDoneType=0 '设置气缸 0 后退到位方式为延时到位
```

2.15.1.4 CYL_FwTime

语　法：CYL_FwTime= value

类　型：ULONG

描　述：设置/读取 CYL_FwDoneType 中涉及到延时到位方式的延时时间。

范　围:ULONG 类型范围，默认值 5000（ms）

例　程

```
CYL_BASE 0
CYL_FwTime =1000 '设置气缸 0 前进到位方式中的延时时间为 1000 毫秒
```

2.15.1.5 CYL_BwTime

语　法：CYL_BwTime= value

类　型：ULONG

描　述：设置/读取 CYL_BwDoneType 中涉及到延时到位方式的延时时间。

范　围:ULONG 类型范围，默认值 5000（ms）

例　程

```
CYL_BASE 0
CYL_BwTime =1000 '设置气缸 0 后退到位方式中的延时时间为 1000 毫秒
```

2.15.1.6 CYL_FwAlmTime

语　法：CYL_FwAlmTime= value

类　型：ULONG

描　述:设置/读取气缸前进开始到到位的最大时间,如超过该时间前进动作还未到位，会发生内部报警，CYL_Status 属性值变为 9：气缸到位超时。

范　围:ULONG 类型范围，默认值 20000（ms）

例　程

```
CYL_BASE 0
CYL_FwAlmTime =1000 '设置气缸 0 前进最大到位时间为 1000 毫秒
```

2.15.1.7 CYL_BwAlmTime

语　法：CYL_BwAlmTime= value

类　型：ULONG

描　述:设置/读取气缸后退开始到到位的最大时间,如超过该时间前进动作还未到位，会发生内部报警，CYL_Status 属性值变为 9：气缸到位超时。

范　围:ULONG 类型范围，默认值 20000（ms）

例　程

```
CYL_BASE 0
CYL_BwAlmTime =1000 '设置气缸 0 后退最大到位时间为 1000 毫秒
```

2.15.1.8 CYL_FwEncValue

语　法：CYL_FwEncValue= value

类　型：Double

描　述：设置/读取气缸前进动作的到位方式为编码器到位时，指定的到位编码器值。

范　围：Double 类型范围，默认值 0

例　程

```
CYL_BASE 0
CYL_FwEncValue =1000 '设置气缸 0 前进到位编码器值为 1000 个脉冲当量
```

2.15.1.9 CYL_BwEncValue

语　法：CYL_BwEncValue= value

类　型：Double

描　述：设置/读取气缸后退动作的到位方式为编码器到位时，指定的到位编码器值。

范　围：Double 类型范围，默认值 0

例　程

```
CYL_BASE 0
CYL_BwEncValue =1000 '设置气缸 0 后退到位编码器值为 1000 个脉冲当量
```

2.15.1.10 CYL_Status

语　法：value=CYL_Status（只读）

类　型：ULONG

描　述：读取气缸当前状态。状态为 9 时，气缸不能再正常执行动作。

返回值：如下

0：复位：原始状态。执行 CYL_Stop、CYL_AlmReset 后的气缸状态都为复位状态

1：前进到位：

2：后退到位

3：前进中

4：后退中

5：保留

6：保留

7：保留

8：保留

9：超过到位时间报警

例　程

```
Dim A As ULONG
CYL_BASE 0
A = CYL_Status   '将气缸 0 的当前状态赋值给变量 A
```

2.15.1.11 CYL_AlmReset

语　法 1：CYL_AlmReset

语　法 2：CYL_AlmReset CYL(no)

描　述：BASE 气缸列表的气缸或指定气缸，复位气缸的状态到复位状态。

参　数：no　气缸号；　范围：根据控制器实际硬件决定。

例　程

```
CYL_BASE 0,1,2
CYL_AlmReset          '复位气缸 0、1、2 的状态
CYL_AlmReset cyl(1)   '复位气缸 1 的状态
```

2.15.1.12 CYL_Move

语　法 1：CYL_Move dir

语　法 2：CYL_Move CYL(no) , dir

描　述：BASE 气缸列表的气缸或指定气缸，执行气缸动作。

参　数：dir　气缸动作方向。0：　气缸后退　；　1：气缸前进

no　气缸号；　范围：根据控制器实际硬件决定。

例　程

```
CYL_BASE 0,1,2,3
CYL_MOVE 1,0,1,1          '气缸 0,1,2,3 分别执行前进、后退、前进、前进动作
Wait CYLDONE              '等待气缸 0,1,2,3 动作到位完成
CYL_Move cyl(1),1         '气缸 1 执行前进动作
```

2.15.1.13 CYL_Stop

语　法 1：CYL_Stop

语　法 2：CYL_Stop CYL(no)

描　述：BASE 气缸列表的气缸或指定气缸，停止气缸动作。

参　数：no　气缸号；　范围：根据控制器实际硬件决定。

注　意：该指令仅适用对双线圈电磁阀控制的气缸控制。

例　程

```
CYL_BASE 0,1,2,3
CYL_Stop              '停止气缸 0,1,2,3 的动作
CYL_Stop cyl(6),1     '停止气缸 6 的动作
```

第 3 章　附录

3.1　错误代码说明

3.1.1 RUN_ERROR 错误代码表

错误代码	说明
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 错误代码表

错误代码	说明
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 控制器运动功能支持列表

项目	说明	PCI-1245L-MAS	PCI-1245-MAS	PCI-1285-MAS	MVP-3245-MAS
单轴运动	点到点运动	✓	✓	✓	✓
	定速运动	✓	✓	✓	✓
	变位运动	✓	✓	✓	✓
	变速运动	✓	✓	✓	✓
	背隙补偿	✓	✓	✓	✓
	T&S 形速度曲線	✓	✓	✓	✓
	运动中重设轴的加速度和减速度	✓	✓	✓	✓
	同步起、同步停运动	不支持	✓	✓	✓
	叠加运动	不支持	不支持	不支持	不支持
	JOG 运动	✓	✓	✓	✓
	手轮运动	✓	✓	✓	✓
	原点复归(16 种模式)	✓	✓	✓	✓
	DI 触发停止	✓	✓	✓	✓
插补功能	直线插补	2 轴	2~3 轴	2~3 轴	2~3 轴
	直接线性插补	2 轴	2~4 轴	2~8 轴	2~4 轴
	2 轴圆弧插补	不支持	✓	✓	✓
	3 轴圆弧插补	不支持	不支持	不支持	不支持
	螺旋插补	不支持	✓	✓	✓
	运动中改变组的运行速度	不支持	✓	✓	✓
跟随运动	电子齿轮	不支持	✓	✓	✓
	电子凸轮	不支持	不支持	不支持	不支持
	龙门	不支持	✓	✓	✓
	CAM DO	不支持	✓	✓	✓
	切向跟随	不支持	✓	✓	✓

路径运动	加载路径功能	**不支持**	支持最多 10000 个点	支持最多 7000 个点	支持最多 10000 个点
	添加直线运动	**不支持**	✓	✓	✓
	添加圆弧运动	**不支持**	✓	✓	✓
	添加螺旋运动	**不支持**	✓	✓	✓
	延迟功能	**不支持**	✓	✓	✓
	添加 DO 开关运动	**不支持**	✓	✓	✓
	路径速度交接功能	**不支持**	✓	✓	✓
	路径速度前瞻功能	**不支持**	**不支持**	**不支持**	**不支持**
比较触发功能	单点比较	**不支持**	✓	✓	✓
	等距线性比较	**不支持**	✓	✓	✓
	不等距随机点比较	**不支持**	✓	✓	✓
等待事件	单轴/插补运动的停止运行功能	✓	✓	✓	✓
	轴的比较	**不支持**	✓	✓	✓
	轴的锁存	**不支持**	✓	✓	✓
输入/输出功能	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 数据类型及类型转换指令

Motion Basic 常用数据类型说明

数据类型	说明
BOOLEAN	布尔数据类型，True 或者 False
BYTE	长度为 8 位的整数数据类型
UBYTE	长度为 8 位的无符号整数数据类型
DOUBLE	长度为 64 位的双精度浮点数据类型
INTEGER	长度为 32 位或 64 位的整数数据类型
UINTEGER	长度为 32 位或 64 位的无符号整数数据类型
LONG	长度为 32 位的整数数据类型
ULONG	长度为 32 位的无符号整数数据类型
SHORT	长度为 16 位的整数数据类型
USHORT	长度为 16 位的无符号整数数据类型
UNSIGNED	整数类型有无符号的修饰符
STRING	字符串类型
SINGLE	长度为 32 位的单精度浮点数据类型

数据类型转换指令

指令	说明
CBYTE	将数字或字符串的表达式转换成字节类型数据
CDBL	将数字或字符串的表达式转换成双精度浮点数
CHR	返回用 ASCII 码表达的值对应的字符
CINT	将数字或字符串表达式转换成整型数据
CLNG	将数字或字符串表达式转换成长整型数据
CLNGINT	将数字或字符串表达式转换成 64 位长整型数据
CSNG	将数字或字符串的表达式转换成单精度浮点数
CUBYTE	将数字或字符串表达式转换为无符号字节类型数据
CUINT	将数字或字符串表达式转换为无符号整型数据
CULNG	将数字或字符串表达式转换为无符号长整型数据

CULNGINT	将数字或字符串表达式转换为无符号 64 位长整型数据
CUNSG	将一个表达式转换成对应的无符号类型
CUSHORT	将数字或字符串表达式转换为无符号短整型数据
VALINT	将一个字符串转换为一个 INTEGER 类型数据
VAL	将一个字符串转换为一个浮点数
HEX	将一个数用十六进制数返回
OCT	将一个数用八进制数返回
VALLNG	将一个字符串转换为一个 LONG 类型数据
VALUINT	将一个字符串转换为一个 UINTEGER 类型数据
VALULNG	将一个字符串转换为一个 ULONG 类型数据
ASC	返回字符串中字符的 ASCII 码