深圳市雷赛控制技术有限公司
SHENZHEN LEADSHINE CONTROL TECHNOLOGY CO.,LTD

雷赛运动控制器
BASIC 指令编程手册

Version 1.5

2017.04.07

版权说明

调试机器要注意安全！用户必须在机器中设计有效的安全保护装置，在软件中加入出错处理程序。否则所造成的损失，雷赛公司没有义务或责任负责

目 录

文档版本

版本号	修订日期	备注

第 1 章 雷赛 BASIC 指令概述

1.1 雷赛 BASIC 编程语言

BASIC 程序的英文名称为 Beginner's All-Purpose Symbolic Instruction Code（适用于初学者的多功能符号指令码）。取全称首字母简称为“BASIC”，其含意就是基础的、简单的、实用的、易学的软件。事实上 BASIC 语言确实达到了设计者的初衷，成为计算机发展史上应用最为广泛的、最简单的编程语言。

BASIC 语言有以下特点：

- 构成简单。BASIC 语言的最基本语句只有 20 多个，而且它们都是常见的英文单词或其变形，如 IF、THEN、FOR、NEXT、END 等，很容易学习和掌握。
- 是一种“人机会话”式的语言。通过键盘操作，用 BASIC 语言编写程序，可以在计算机上边编写、边修改、边运行。BASIC 在运行中向人们提示信息、指出错误，帮助编程者修改，即实现了人和机器的对话。
- 功能丰富，适用面广。BASIC 语言除了能进行科学计算和数据处理外，还能进行字符处理、图形处理、音频处理、网络通讯等。

许多厂商生产的运动控制器都配有 BASIC 指令集，具有代表性的有英国的 Trio 公司、美国的 Baldor 公司。目前国内只有雷赛公司的运动控制器提供 BASIC 指令集。它们的共同特点是：大多数程序语句和标准的 BASIC 语言相同，只是各家的运动控制指令略有区别。

运动控制器的 BASIC 指令集与 PLC 的梯形图指令、数控机床的 G 代码相比，具有如下特点：

- 简单易学：基本的程序指令和 Visual BASIC 兼容；运动控制指令十分简单。
- 交互性强：程序为解释执行方式，可以实时交互控制，用户可以随时了解程序执行的状态。
- 实时多任务：可以同时运行多个子任务，所以具有强大的实时多任务处理功能。
- 中断响应：定时器信号、通用数字输入信号、限位信号可作为中断源，中断处理程序使控制器的响应速度更快。
- 稳定性高：当用户程序出现错误时，不会出现系统崩溃。

1.2 雷赛 BASIC 编程执行原理

雷赛 BASIC 程序采用解释执行的方式逐行解析代码行并执行相应的功能，见下图 1.1。

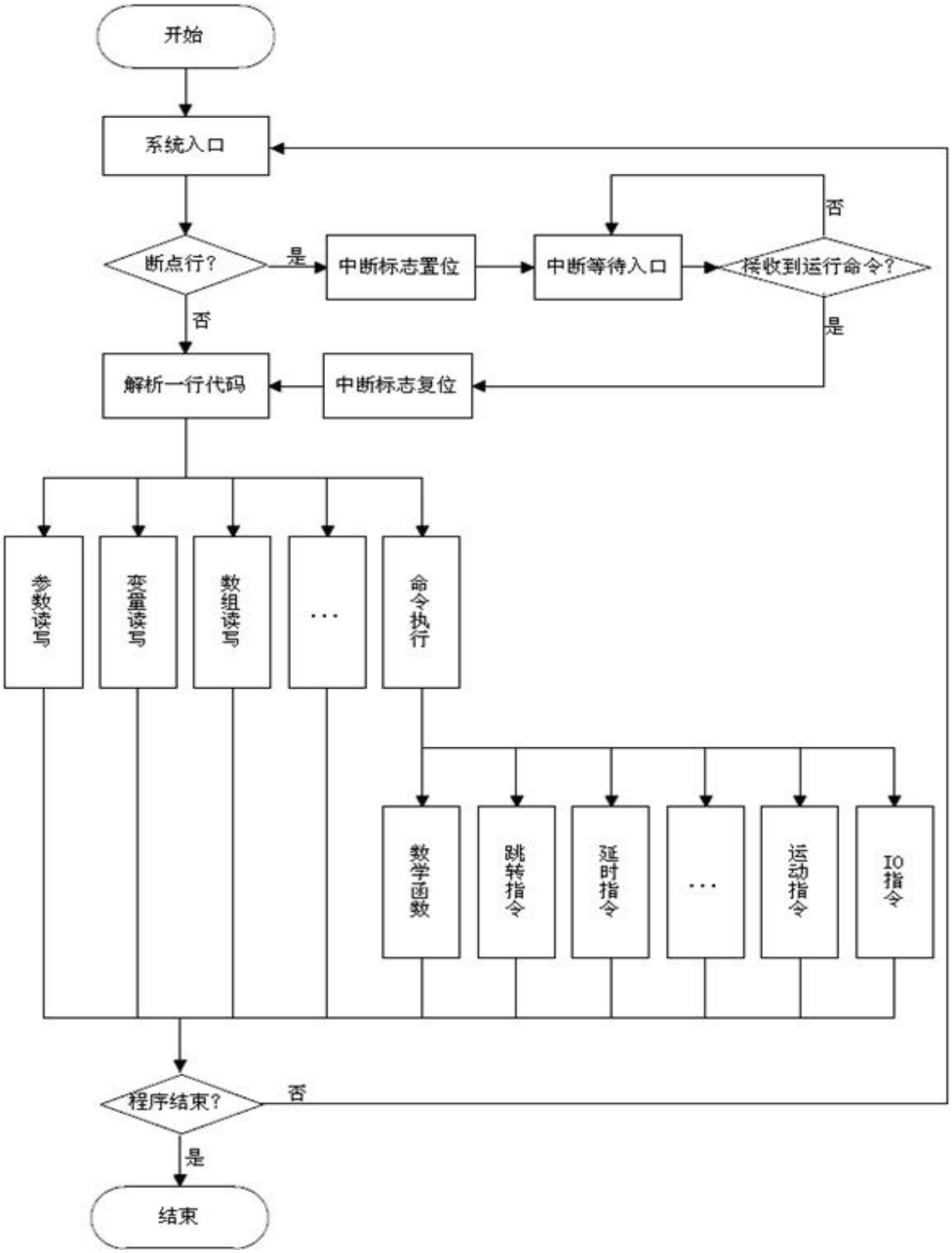


图 1.1 BASIC 编程执行示意图

1.3 雷赛 BASIC 编辑器介绍

BASIC 编辑器是在编程开发测试工具 SMC BASIC STUDIO 上实现的，此工具集合测试、调试、IO 监控等多种功能。同时也可执行 G 代码、BASIC 和 G 代码混合编程等。

对控制器编程前需先连接好控制器再在编程区编辑代码，编辑完代码再下载到控制器实现对电机等负载的驱动。软件具体介绍见 《SMC BASIC STUDIO 软件使用手册》。

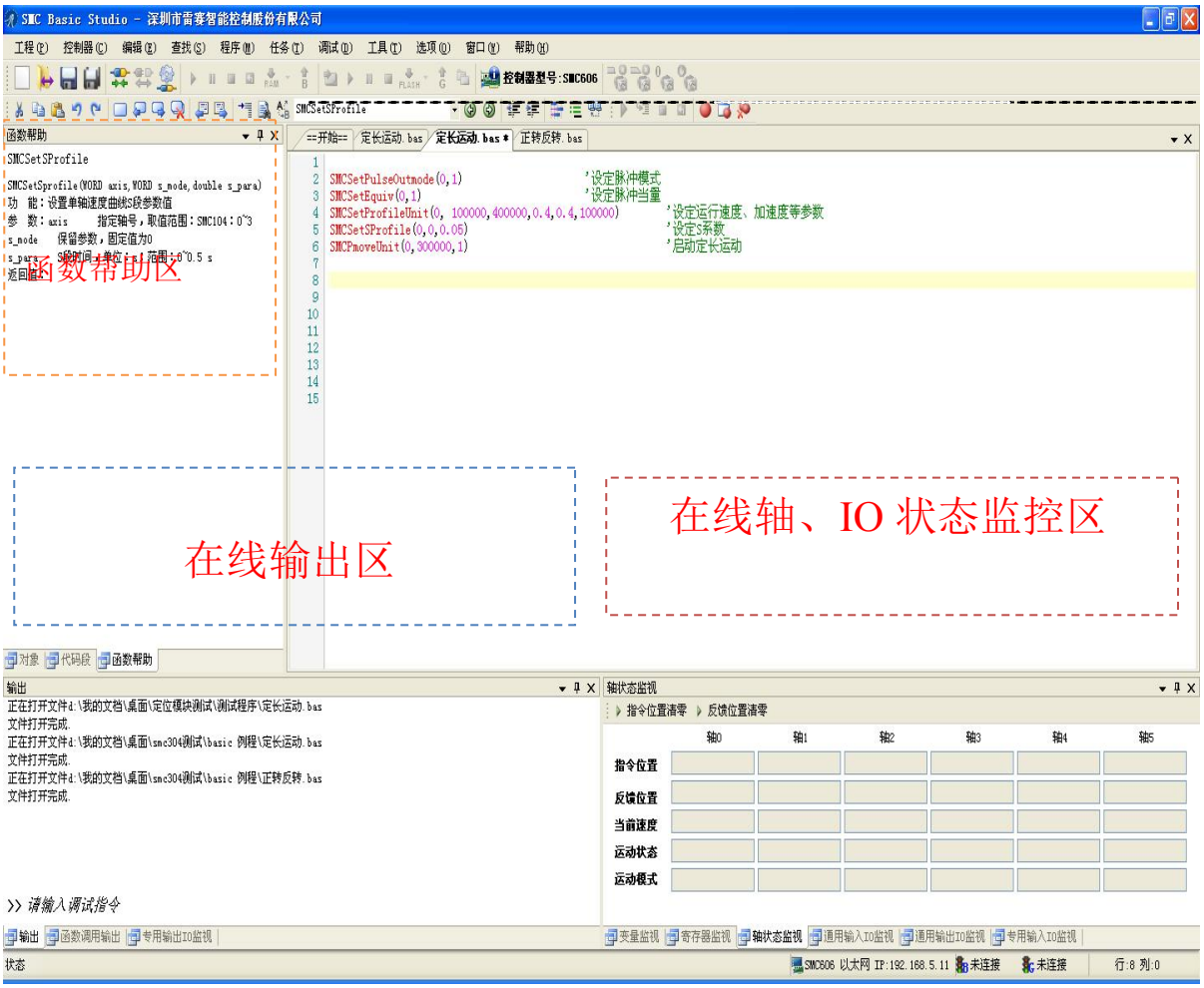


图 1.2 SMC BASIC STUDIO 软件编辑示意图

第 2 章 功能描述

2.1 参数设置

在使用控制器之前，我们需要初始化某些参数，如脉冲当量、脉冲模式等配置参数。

2.1.1 控制器初始化

BASIC 代码是在编程开发软件 SMC BASIC STUDIO 的编程器上编辑的，打开 SMC BASIC STUDIO 软件后，其与雷赛控制器的连接初始化可分为串口、网口两种类型。

2.1.1.1 串口设置

串口的连接串口号默认“COM1”，波特率默认值为 115200，连接前确认串口线已连接好，SMC BASIC STUDIO 软件上连接图标及参数设置见图 2.1，先设置连接参数，再连接控制器：

图 2.1 串口连接示意图

连接好控制器后，可以在软件 SMC BASIC STUDIO 编辑区编辑指令，可以修改与串口连接的一些参数。

相关指令：

名称	功能	参考
Setcom	设置串口的通讯参数	4.2 节
Com	读取 com 口参数	

⚠注意：修改串口参数后需要重新启动控制器，且连接参数需按照设定值输入否则可能会出现无法与计算机通讯的情况。

例　程：SETCOM(115200,8,1,2,1) '设置 COM1 端口的波特率为 115200

'数据位为 8，停止位为 1，校验方式为偶校验

Print　COM'打印 com 口数据

2.1.1.2 网口设置

网口的连接 IP 默认值为 192.168.5.11，连接前确认电脑 IP 地址与设定 IP 在同一网段,同时最后一位地址不能一样；如下图电脑 IP 设定为 192.168.5.122，与控制器的 IP 值都在 5 网段，最后一位地址（范围：1-254）分别为 11 和 122，见图 2.2。

图 2.2 网口连接示意图

在软件 SMC BASIC STUDIO 编辑区，我们可以编辑指令来修改与网口连接的一些参数。

相关指令：

名称	功能	参考
IPADDR	回读网口的通讯参数	4.2 节
SETIP	设置网口的通讯参数	

例　程：设置读取控制器 IP 地址

Setip (192,168,5,11) '设置控制器的 IP 地址为 192.168.5.11

Print　ipaddr'读取控制器设置的 IP 地址，如显示结果为 192.168.5.11

⚠注意：修改参数网口参数后需要重新启动控制器，且连接参数需按照设定值输入，电脑网口IP地址与控制器需在同一网段，否则可能会出现无法与计算机通讯的情况。

2.1.2 脉冲输出模式设置

雷赛控制器可采用多种脉冲输出模式。由于市面上的众多电机驱动器厂家信号接口要求各有不同（常用的有六种类型，下图 2.3），所以在使用运动控制器控制具体的电机驱动器时，必须根据电机驱动器的接收信号类型，对运动控制器的脉冲输出模式进行正确的设定，电机才能正常工作。

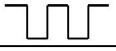
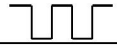
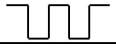
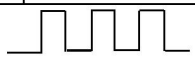
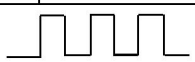
脉冲输出模式	正方向脉冲		负方向脉冲	
	PULSE 输出端	DIR 输出端	PULSE 输出端	DIR 输出端
0		高电平		低电平
1		高电平		低电平
2		低电平		高电平
3		低电平		高电平
4		高电平	高电平	
5		低电平	低电平	
6	PULSE 输出端  DIR 输出端 		PULSE 输出端  DIR 输出端 	

图 2.3 脉冲模式标准示意图

相关指令：

名称	功能	参考
SMCSetPulseOutmode	设置脉冲输出模式	4.3 节
SMCGetPulseOutmode	读取脉冲输出模式	

例程 1：某设备电机驱动器只接受脉冲+方向模式的脉冲，我们可设定控制器输出脉冲+方向模式 0

```
Dim MyAxis,outmode, outmode1,result
MyAxis = 0'轴号 0
outmode =0'脉冲输出模式为 0
result=SMCSetPulseOutmode ( MyAxis, outmode)   '设置脉冲输出模式
```

```
result=SMCGetPulseOutmode ( MyAxis, outmode1) '读取脉冲输出模式参数
Print outmode1                                  '打印脉冲模式输出，结果为 0
```

例程 2：某设备电机驱动器只接受双脉冲模式的脉冲，我们可设定控制器输出双脉冲，模式 4

```
Dim MyAxis,MyEquiv,result
MyAxis = 0'轴号 0
outmode =4'双脉冲输出模式为 4
result=SMCSetPulseOutmode ( MyAxis, outmode)   '设置脉冲输出模式
```

2.1.3 脉冲当量设置

雷赛控制器提供了脉冲当量设置功能，该功能可以自定义位置（位移）单位；并且提供了相应的各种高级运动函数。

相关指令：

名称	功能	参考
SMCSetEquiv	设置脉冲当量值	4.4 节
SMCGetEquiv	读取脉冲当量值	

⚠注意 1）该函数适用于高级运动函数（包括点位、插补、连续插补运动）

2）当使用高级运动函数进行运动前，必须先使用该函数设置各运动轴的脉冲当量值，该值不能设置为 0

例程 1：某设备中运动控制器每发送 100pulse，设备平台前进 1mm。用户可以通过脉冲当量设置功能将运动的位移单位设置为 mm，速度单位则为 mm/s。脉冲当量为 100pulse/mm。

```
Dim MyAxis,MyEquiv,MyEquiv1,result
MyAxis = 0'轴号
MyEquiv = 100'脉冲当量设置为  100pulse/mm
result =SMCSetEquiv(MyAxis, MyEquiv)      '设置脉冲当量
result = SMCGetEquiv(MyAxis, MyEquiv1)    '读取脉冲当量
Print MyEquiv1                             '打印脉冲当量值为 100
```

例程 2：如果用户希望以脉冲（pulse）为单位，那么用户可以将脉冲当量值设置为 1，即此时运动的位移单位为 pulse，速度单位则为 pulse/s。

```
Dim MyAxis,MyEquiv ,result
```

```
MyAxis = 0'轴号
MyEquiv = 1'脉冲当量设置为 1pulse/unit
result =SMCSetEquiv( MyAxis, MyEquiv)    '设置脉冲当量
```

例程 3：电机运动一圈需 10000pulse ，丝杆导程 5mm。运动距离 2mm 则需脉冲 4000 个。

换算公式为： 脉冲当量=电机一圈脉冲数/丝杆导程,即 2000 pulse/mm

运动 2mm 则发送了 2000*2=4000 个脉冲

```
Dim MyAxis,MyEquiv ,dos,cic_pos,result
MyAxis = 0'轴号
cic_pos=10000'电机运动一圈需脉冲数
dos= 5'丝杆导程
MyEquiv = cic_pos/ dos                    '换算成脉冲当量值
result =SMCSetEquiv( MyAxis, MyEquiv)    '设置脉冲当量
```

2.1.4 反向间隙设置

雷赛控制器提供了反向间隙补偿功能，以降低机械传动反向间隙的影响。

相关指令：

名称	功能	参考
SMCSetBacklashUnit	设置反向间隙值	4.5 节
SMCGetBacklashUnit	读取反向间隙设置值	

⚠注意：反向间隙目前只支持插补反向间隙，不支持单轴。

例 程：反向间隙补偿功能。

```
Dim MyCardNo, MyCrd, MyaxisNum, MyBacklash,outmode1
Dim AxisArray(2),Pos(2) ,Cen(2) , result
Dim MyBacklash
result=SMCsetequiv (0, 100) '设置 X 轴脉冲当量为 100pulse/unit
result=SMCsetequiv (1, 100) '设置 Y 轴脉冲当量为 100pulse/unit
MyBacklash = 10'反向间隙设置值为 10unit
MyCrd =0'参与插补运动的坐标系
AxisArray(0)=0'定义 0 轴为插补 X 轴
```

```
AxisArray(1)=1'定义 1 轴为插补 Y 轴
result=SMCSetBacklashUnit(AxisArray(1),MyBacklash)   'Y 轴进行反向间隙补偿设置
result=SMCGetBacklashUnit (AxisArray(1),outmode1)    '读取反向间隙参数
print outmode1                                        '打印反向间隙设定值，结果为 10
MyaxisNum = 2'插补运动轴数为 2
Pos(0) = 0'设置终点 X 坐标
Pos(1) = 0'设置终点 Y 坐标
Cen(0) = 100'设置圆心 X 坐标
Cen(1) = 0'设置圆心 Y 坐标
result=SMCSetVectorProfileUnit(MyCrd,0,1000,0.1,0.1,0) '设置插补速度曲线参数
result=SMCArcMoveCenterUnit(MyCrd,MyaxisNum,AxisArray(0),Pos(0),Cen(0),0,0,1)
'XY 轴执行顺时针方向圆弧插补运动，圆弧圈数为 0，相对坐标模式
```

运行结果：

Y 轴设置反向间隙为 10unit,当 Y 轴开始反向运动时，为了补偿间隙误差，Y 轴先反向运动 10unit 然后继续进行圆弧插补运动。运动轨迹图如下：

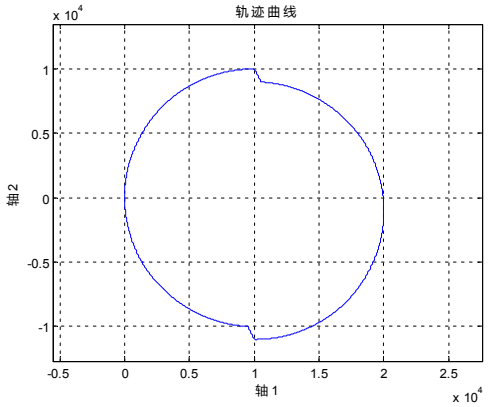


图 2.4 反向间隙模式下的轨迹图

2.1.5 轴 IO 映射

雷赛控制器支持轴 IO 映射配置功能，支持将轴专用 IO 信号配置到任意一个硬件输入口，如：可将限位接口当原点信号。该功能可减少现场接线、换线的困难等。

相关指令：

名称	功能	参考

SMCSetAxisIoMap	设置 IO 轴映射参数	4.25 节
SMCGetAxisIoMap	读取 IO 轴映射参数	

⚠注意：SMCSetAxisIoMap 的映射主要参数如下表：

Axis 指定轴号	IoType：指定需映射的功能	MapIoType 轴 IO 映射端口	MapIoIndex 轴 IO 映射索引号
SMC606：0-5 SMC604：0-3 SMC604A：0-3 SMC306：0-5	0：正限位信号 1：负限位信号 2：原点信号 3：急停信号 4：减速停止信号（保留） 5：伺服报警信号 6：伺服准备信号（保留） 7：伺服到位信号	0：正限位输入端口 1：负限位输入端口 2：原点输入端口 3：伺服报警输入端口 4：伺服准备输入端口 5：伺服到位输入端口 6：通用输入端口	1）当轴 IO 映射类型设置为 6 时，此参数可设置为 0-控制器最大 IO 输入值-1，表示该映射对应的具体通用输入端口号 2）当轴 IO 映射类型设置为 0-5 时，此参数可设置 0-5 整数，表示该映射所对应的具体轴号

例程 1：设置通用输入口 0 接口作为所有轴的急停信号，并且设置急停信号为低电平有效。

```
Dim Axis,IoType, MapIoType, MapIoIndex ,filter_time,result
Axis = 0'指定轴号：第 0 轴
IoType = 3'指定轴的 IO 信号类型为：急停信号
MapIoType = 6'轴 IO 映射类型：通用输入端口
MapIoIndex = 0'轴 IO 映射索引号：通用输入口 0
filter_time = 0.01'0.01 秒滤波
For Axis=0 to 5 '循环，依次对 0~5 号轴进行设置
    result=SMCSetEMGMode(Axis,1,0)   '设置 EMG 信号使能，低电平有效
    result=SMCSetAxisIoMap(Axis,IoType,MapIoType,MapIoIndex,filter_time)
'设置通用输入口 0 接口作为所有轴的急停信号，并且设置急停信号为低电平有效
Next Axis
```

例程 2：设置第 2 轴原点接口作为第 0 轴的正限位信号

```
Dim CardNo,Axis,IoType, MapIoType, MapIoIndex ,filter_time
Dim MapIoType1, MapIoIndex1 ,filter_time1, result
Axis = 0'指定轴号：第 0 轴
IoType = 0'指定轴的 IO 信号类型为：正限位信号
MapIoType = 2'轴 IO 映射类型：原点信号
MapIoIndex = 2'轴 IO 映射索引号：第 2 轴
filter_time = 0.01'0.01 秒滤波
result=SMCSetAxisIoMap(Axis,IoType,MapIoType,MapIoIndex,filter_time)
'设置第 2 轴原点接口作为第 0 轴的正限位信号参数
result=SMCGetAxisIoMap(Axis,IoType,MapIoType1,MapIoIndex1,filter_time1)
'读取设置参数
Print MapIoType1, MapIoIndex1 ,filter_time1   '打印 IO 映射值，结果为 2，2，0.01
```

运行结果：当第 2 轴原点信号有效后，轴 0 的正限位信号立即响应。

2.2 运动功能

2.2.1 点位运动

主要包括定长运动、恒速运动、在线变速度、在线变位置运动功能。

2.2.1.1 参数设置

雷赛控制器在执行点位运动控制指令时，可使电机按照 T 形速度曲线或 S 形速度曲线进行点位运动。同时可以设置起始速度、停止速度，不同的加速时间、减速时间等速度参数。

相关指令：

名称	功能	参考
SMCGetProfileUnit	读取单轴运动速度曲线参数	4.7 节
SMCSetProfileUnit	设置点位运动速度参数	
SMCSetSprofile	设置 S 段时间参数	
SMCGetSprofile	读取 S 段时间参数	
SMCSetDecStopTime	设置单轴异常减速停止参数	

SMCGetDecStopTime	读取单轴异常减速停止参数	

⚠注意：（1）由于运动控制器的最大脉冲输出频率为 1MHz(SMC104)或者 4MHz(SMC300、SMC600 系列)，故设置的最大速度与脉冲当量设置值的乘积必须小于该最大脉冲输出频率。

（2）单轴速度曲线 S 平滑时间，若值为 0 则为 T 形曲线，不为 0 则为 S 平滑曲线，S 平滑时间值范围为 0-1S。

为了让平台在运动快速加速、准确停止，一般采用梯形速度曲线控制运动过程，如图 2.5 所示。即：电机以起始速度开始运动，加速至最大速度后保持速度不变，结束前减速至停止速度，并停止。运动的距离由点位运动指令决定。

“起始速度”：设置单轴运动的初始速度，单位：unit/s 。

“运行速度”：设置单轴运动的最大运行速度，单位：unit/s 。

“停止速度”：设置单轴运动的停止速度，单位：unit/s 。

“加速时间”：设置单轴运动时从起始速度加速到最大运行速度需要的时间，见下图 Tacc。

“减速时间”：设置单轴运动时从最大运行速度减速到停止速度需要的时间，见下图 Tdec。

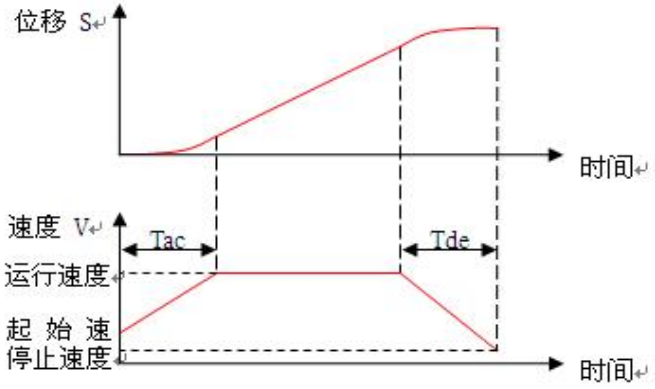


图2.5　梯形速度曲线及对应的位移曲线

为改善平台运动的平稳性，雷赛运动控制器还提供了 S 形速度控制曲线。

在 S 形速度控制过程中，指令脉冲频率从一个内部设定的速度快速加速到起始速度，然后作 S 形加速运动；运动结束前，指令脉冲频率作 S 形减速运动到停止速度，然后再快速减速到一个内部设定的速度，这时脉冲输出停止，如下图 2.6 所示。

“起始速度”：设置单轴运动的初始速度，单位：unit/s 。

“运行速度”：设置单轴运动的最大运行速度，单位：unit/s 。

“停止速度”：设置单轴运动的停止速度，单位：unit/s 。

“加速时间”：设置单轴运动时从起始速度加速到最大运行速度需要的时间，见下图 Tacc。

“减速时间”：设置单轴运动时从最大运行速度减速到停止速度需要的时间，见下图 Tdec。

“S 段时间”：设置单轴速度曲线 S 段的时间参数，见下图 spara。

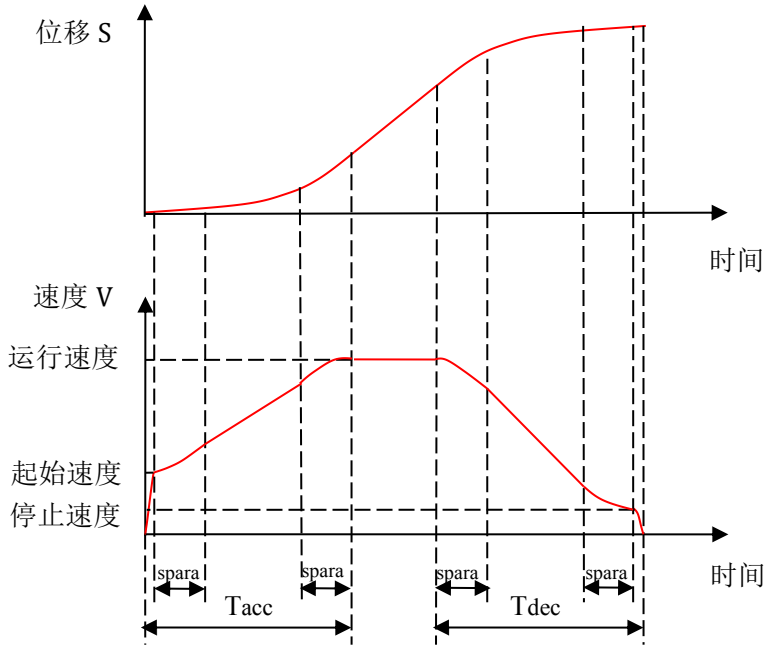


图2.6 S形速度曲线及对应的位移曲线

2.2.1.2 定长运动

定长运动指位置值确定，其速度曲线包含 T 和 S 形速度规划。

相关指令：

名称	功能	参考
SMCPmoveUnit	启动点位运动	4.7 节

例程 1：轴 0 执行 T 形，起始、停止速度不相等的点位运动，运动距离 1000units

```
''''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''''
Dim MyCardNo,Myaxis,Mys_mode,Myposi_mode
Dim Mys_para,MyDist
Myaxis =0'轴号
result=SMCSetEquiv(Myaxis,100)  '设置脉冲当量为 100pulse/unit
MyMin_Vel = 100'设置起始速度为 100unit/s
MyMax_Vel = 1000'设置最大速度为 1000unit/s
```

```
MyTacc = 0.1'设置加速时间为 0.1s
MyTdec = 0.1'设置减速时间为 0.1s
MyStop_Vel = 200'设置停止速度为 200unit/s
Mys_mode = 0'保留参数
Mys_para =0'S 段时间为 0s 即速度曲线不平滑,T 形曲线
MyDist = 1000'设置运动距离为 1000unit
Myposi_mode = 0'设置运动模式为相对坐标模式
"""""""""""""""""""""指令调用执行"""""""""""""""""""""
'第一步、设置 0 号轴速度曲线参数
result=SMCSetProfileUnit(Myaxis,MyMin_Vel,MyMax_Vel,MyTacc,MyTdec,MyStop_Vel)
'第二步、设置 0 号轴 S 段参数
result=SMCSetsprofile(Myaxis,Mys_mode,Mys_para)
'第三步、启动 0 号轴梯形定长运动
result=SMCPMoveUnit(Myaxis,MyDist,Myposi_mode)
```

例程 2：轴 0 执行 S 形，起始、停止速度为 0 的非对称点位运动，运动距离 1000units。

```
"""""""""""""""""""""变量定义"""""""""""""""""""""
Dim MyCardNo,Myaxis,Mys_mode,Myposi_mode
Dim Mys_para,MyDist, result
Myaxis =0'轴号
SMCSetEquiv(Myaxis,100)                    '设置脉冲当量为  100pulse/unit
MyMin_Vel = 0'设置起始速度为 0unit/s
MyMax_Vel = 1000'设置最大速度为 1000unit/s
MyTacc = 0.1'设置加速时间为 0.1s
MyTdec = 0.3'设置减速时间为 0.3s
MyStop_Vel = 0'设置停止速度为 0unit/s
Mys_mode = 0'保留参数
Mys_para =0.05'平滑系数 S 段时间为 0.05s
MyDist = 1000'设置运动距离为 1000unit
Myposi_mode = 0'设置运动模式为相对坐标模式
"""""""""""""""""""""指令调用执行"""""""""""""""""""""
```

```
'第一步、设置 0 号轴速度曲线参数
result=SMCSetProfileUnit(Myaxis,MyMin_Vel,MyMax_Vel,MyTacc,MyTdec,MyStop_Vel)
'第二步、设置 0 号轴 S 段参数
result=SMCSetsprofile(Myaxis,Mys_mode,Mys_para)
'第三步、启动 0 号轴 S 形定长运动
result=SMCPMoveUnit(Myaxis,MyDist,Myposi_mode)'0 号轴 S 形定长运动
```

2.2.1.3 恒速运动

恒速运动模式中，雷赛控制器可以控制电机以 T 形或 S 形速度曲线在指定的加速时间内从起始速度加速至最大速度，然后以该速度一直运行，直至调用停止指令或者该轴遇到限位信号才会按启动时的速度曲线减速停止。

相关指令：

名称	功能	参考
SMCVmove	指定轴恒速运动	4.7 节
SMCStop	指定轴停止运动	

⚠注意：恒速运动需手动停止，否则运动将一直持续。

例　程：轴 0 执行 T 形恒速运动，当通用输入口 1 为低电平时恒速运动停止；

```
"""""""""""""""""""""变量定义"""""""""""""""""""
Dim MyCardNo,Myaxis,Mys_mode,dir
Myaxis =0'轴号
dir = 0'设置运动方向为 0，负方向
"""""""""""""""""""""指令调用执行"""""""""""""""""""
'第一步、设置脉冲当量、0 号轴速度曲线参数
result=SMCSetEquiv(Myaxis,100)
result=SMCSetProfileUnit(Myaxis,0,1000,0.1,0.1,0)
'第二步、设置 0 号轴平滑 S 段时间参数
result=SMCSetsprofile(Myaxis,0,0)
'第三步、启动 0 号轴负方向恒速运动
result=SMCVmove (Myaxis, dir)     '启动 0 号轴负方向恒速运动
```

```
'第四步、等待 IO 口 1 为低电平时，停止恒速运动
WhileSMCReadInBit(1)=1
Wend
SMCStop(Myaxis,0)                    '输入口 1 电平为低时，0 轴恒速减速停止
```

2.2.1.4 在线变速变位运动

在点位运行过程中，最大速度 Max_Vel 和目标位置 Dist 均可以实时改变，在恒速运动中只能改变运动速度，不能改变位置。

相关指令：

名称	功能	参考
SMCResetTargetPositionUnit	在线改变指定轴的当前目标位置	4.7 节
SMCUpdateTargetPositionUnit	强制改变指定轴的当前目标位置	
SMCChangeSpeedUnit	在线改变指定轴的当前运动速度	

⚠注意：1）在线变位适用于点位运动；在线变速适用于点位和恒速运动。

2）在线变位后的目标位置为绝对坐标位置值，无论当前的运动模式为绝对坐标还是相对坐标模式。

3）在线变速可以设置变速时间，设置的变速时间是从当前速度变速到新速度的时间。此时控制器会重新计算起始速度加速到最高速度所需的时间以及最高速度减速到停止速度所需的时间，即加减速时间会被重新计算。变速一旦成立，该轴的默认运行速度将会被改写为 New_Vel，加减速时间也会被控制器新计算的值所覆盖。

4）在线变位需运动状态处于未停止状态才会执行，而强制变位，不管当前轴是否在运动都可以改变位置。

例程 1：定长反向变位

```
''''''''''''''''''''''''''''''''指令调用执行''''''''''''''''''''''''''''''''
'第一步、设置起始速度、停止速度为 0, 加减速时间为 0.1s, 最大运行速度 10000
result=SMCSetProfileUnit (0,0,10000,0.1,0.1,0)
'第二步、设置轴 0 平滑系数 s 为 0.1s
result=SMCSetSprofile (0,0, 0.1)
'第三步、启动轴定长运动
```

```
result=SMCPmoveunit (0,10000, 1)
'第四步、启动在线变速度
Delay(600)                                    '延时 600ms 后再变速度
result=SMCResetTargetPositionUnit(0,0)   '轴 0 变位到 0
```

运行结果：定长运动 600ms 后位置变到 0，运动立即反向运行，回到位置点 0，见图 2.7。

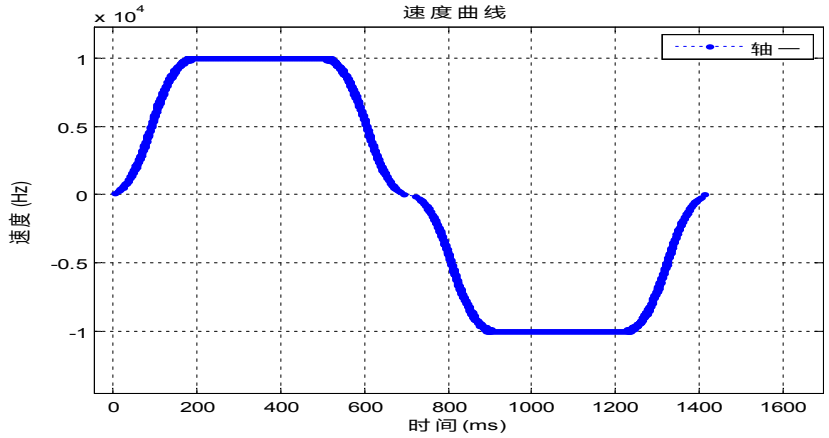


图 2.7 例程运行速度曲线

例程 2：定长运动同向强制变位

```
''''''''''''''''''''''''''''''指令调用执行''''''''''''''''''''''''''''''
'第一步、设置速度曲线参数
result=SMCSetProfileUnit (0,0,10000,0.2,0.1,0)
'第二步、设置速度平滑 S 时间参数，设为 T 形
result=SMCSetSprofile (0,0, 0)
'第三步、启动定长运动
result=SMCPmoveunit (0,5000, 1)
'第四步、启动强制变位
delay(600)   '延时 600ms
result=SMCUpdateTargetPositionUnit (0,10000)'轴 0 强制变位到 10000
```

运行结果：定长运动，当即将到达设定位置点 5000 时，再强制把位置点设置到 10000.运动将加速到最大速度，最终停止到位置点 10000。见图 2.8

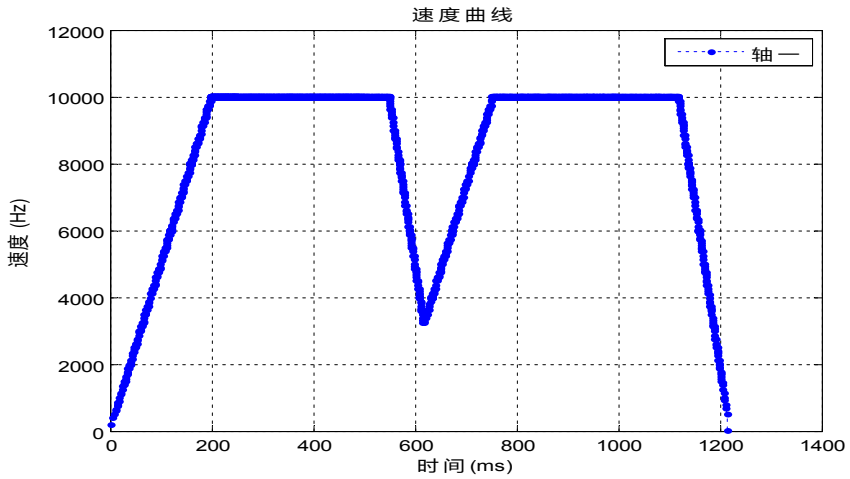


图 2.8 例程 2 运行速度曲线

例程 3：轴 0 负方向恒速运动，最大运行速度 10K，300ms 后最大速度变为 2K（运动将会往正方向运动），再 300ms 后最大速度变为 10K（运动方向为正方向），再 300ms 后停止该轴运动；

```
''''''''''''''''''''''变量定义''''''''''''''''''''
Dim MyCardNo,Myaxis,Mys_mode,Myposi_mode
Dim Mys_para,MyDist,dir, New_ve2, New_vel,result
Myaxis =0'轴号
SMCSetEquiv(Myaxis,10) '设置脉冲当量为 10pulse/unit
MyMin_Vel = 0'设置起始速度为 0unit/s
MyMax_Vel = 10000'设置最大速度为 1000unit/s
MyTacc = 0.1'设置加速时间为 0.1s
MyTdec = 0.1'设置减速时间为 0.1s
MyStop_Vel = 0'设置停止速度为 0unit/s
Mys_mode = 0'保留参数
Mys_para =0'平滑系数 S 段时间为 0s
dir = 0'设置运动方向为 0，负方向
New_vel=2000'新速度为 2K
New_ve2=10000'新速度为 10K
''''''''''''''''''''''指令调用执行''''''''''''''''''''
'第一步、设置速度曲线参数
result=SMCSetProfileUnit(Myaxis,MyMin_Vel,MyMax_Vel,MyTacc,MyTdec,MyStop_Vel)
```

```
'第二步、设置速度平滑 S 参数
result=SMCSetsprofile(Myaxis,Mys_mode,Mys_para)
'第三步、'启动 0 号轴负方向恒速运动
SMCVmove (Myaxis, dir)
'第四步、'启动在线变速度
delay(300)                                          '延时 300ms
result=SMCChangeSpeedUnit(Myaxis,New_vel,0.1)
'第五步、'再次启动在线变速度
delay(300)                                          '延时 300ms
result=SMCChangeSpeedUnit(Myaxis,New_ve2,0.1)
'第六步、'减速停止恒速运动
delay(300)                                          '延时 300ms
result=SMCStop(Myaxis,0)
```

例程 4：定长运动在线变速、在线变位、强制变位。轴 0 执行运行速度 10K，运动位置 100K，300ms 后速度变为 2K，再 300ms 后位置变为 5K；当运动停止后，再变位到 0。

```
''''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''''
Dim MyCardNo,Myaxis,Mys_mode,Myposi_mode
Dim Mys_para,MyDist,New_Pos,New_Pos1,result
Myaxis =0'轴号
result=SMCSetEquiv(Myaxis,10)      '设置脉冲当量为  10pulse/unit
MyMin_Vel = 0'设置起始速度为 0unit/s
MyMax_Vel = 10000'设置最大速度为 1000unit/s
MyTacc = 0.1'设置加速时间为 0.1s
MyTdec = 0.1'设置减速时间为 0.1s
MyStop_Vel = 0'设置停止速度为 0unit/s
Mys_mode = 0'保留参数
Mys_para =0.05'平滑系数 S 段时间为 0.05s
MyDist = 100000'设置运动距离为 100000unit
Myposi_mode = 0'设置运动模式为相对坐标模式
New_vel=2000'新速度为 2K
```

```
New_Pos=5000'新位置为 5K
New_Pos1 =0'新位置为 0
""""""""""""""""""""""指令调用执行""""""""""""""""""""""
'第一步、设置速度曲线参数
result=SMCSetProfileUnit(Myaxis,MyMin_Vel,MyMax_Vel,MyTacc,MyTdec,MyStop_Vel)
'第二步、设置 0 号轴平滑 S 时间参数
result=SMCSetsprofile(Myaxis,Mys_mode,Mys_para)
'第三步、启动定长运动
result=SMCPMoveUnit(Myaxis,MyDist,Myposi_mode)
'第四步、启动在线变速
delay(300)                              '延时 300ms
result=SMCChangeSpeedUnit(Myaxis,New_vel,0.1)
'第五步、启动在线变位
delay(300)                              '延时 300ms
result=SMCResetTargetPositionUnit(Myaxis, New_Pos)
'第六步、等待运动停止，强制变位到 0
whileSMCCheckDone(0)=0'等待轴 0 运动停止
wend
result=SMCUpdateTargetPositionUnit(Myaxis,New_Pos1) '强制运动到新位置 0
```

2.2.2 回原点运动

在进行精确的运动控制之前，需要设定运动坐标系的原点。运动平台上都设有原点传感器（也称为原点开关），可作为原点信号的输入源。

雷赛控制器提供了 10 种回原点运动的方式：

方式 0：一次回零

该方式以设定速度回原点；适合于行程短、安全性要求高的场合。动作过程为：电机从初始位置以恒定速度向原点方向运动，当到达原点开关位置，原点信号被触发，电机立即停止（过程 0）；将停止位置设为原点位置，如图 2.9 所示。

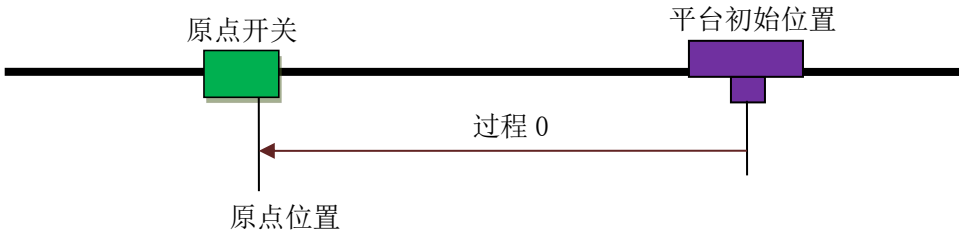


图 2.9　一次回零方式示意图

方式 1：一次回零加回找

该方式先进行方式 1 运动，完成后再反向回找原点开关的边缘位置，当原点信号第一次无效的时候，电机立即停止；将停止位置设为原点位置如图 2.10 所示。

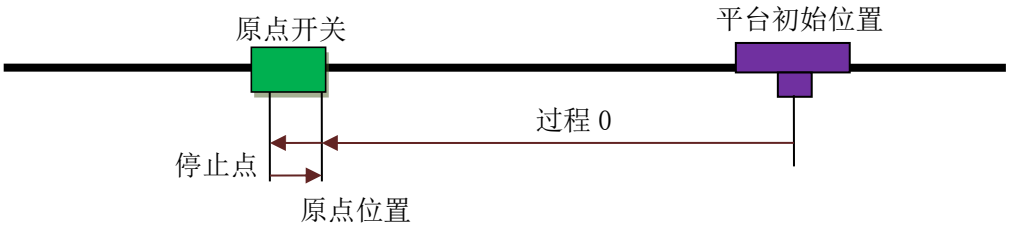


图 2.10　一次回零加回找方式示意图

方式 2：两次回零

如图 2.11 所示，该方式为方式 0 和方式 1 的组合。先进行方式 1 的回零加反找，完成后再进行方式 0 的一次回零。可参见方式 0 和方式 1 的说明。

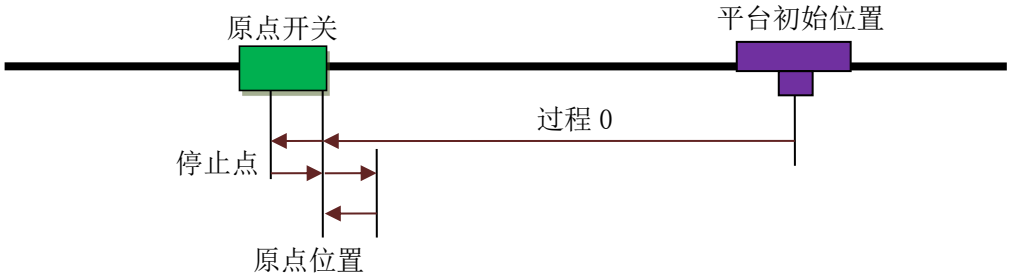


图 2.11 二次回零方式示意图

方式 3：一次回零后再找 EZ 信号

该方式在回原点运动过程中，当找到原点信号后，还要等待该轴的 EZ 信号出现，此时电机停止。回原点过程如图 2.12 所示。

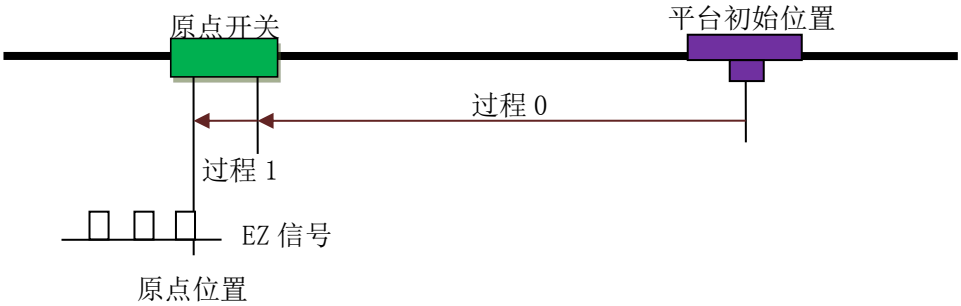


图 2.12 一次回零后再找 1 个 EZ 信号回零方式示意图

方式 4：记 1 个 EZ 信号回零

该方式在回原点运动过程中，当检测到该轴的 EZ 信号出现一次后，此时电机停止。回原点过程如图 2.13 所示。

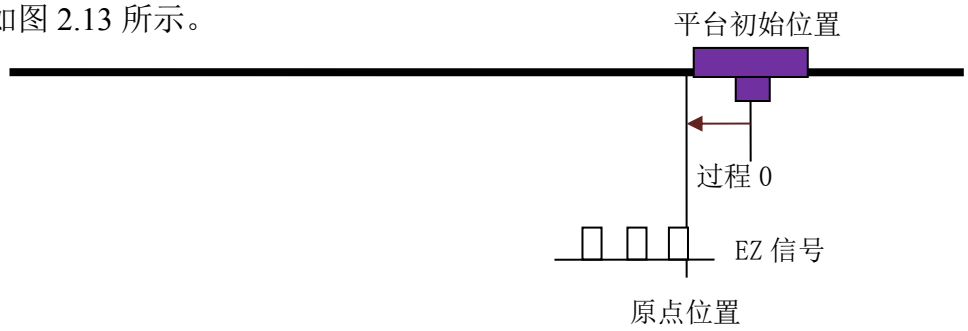


图 2.13 记 1 个 EZ 信号回零方式示意图

方式 5：一次回零再反找 EZ 信号

该方式在回原点运动过程中，当找到原点信号后，减速停止，然后以反找速度反向找到 EZ 生效此时电机停止。回原点过程如图 2.14 所示。

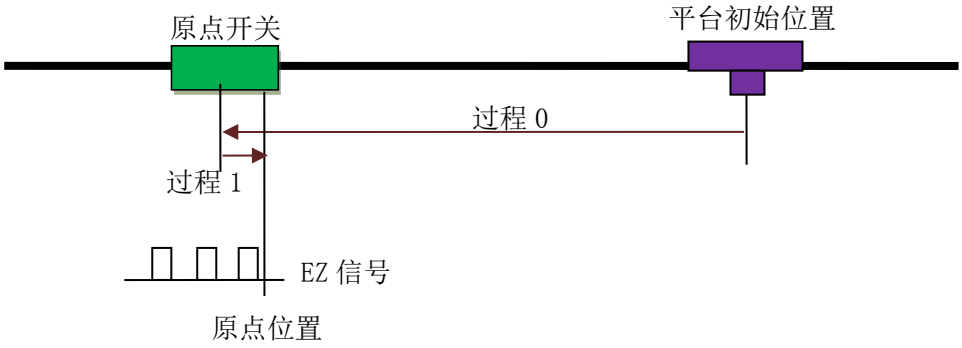


图 2.14 一次回零反找一个 EZ 进行回零

方式 6：原点锁存

如图 2.15 所示，电机先以设定速度回原点，当原点开关边沿触发时，将当前位置锁存下来，同时电机减速停止。电机减速停止完成后再反向回找锁存位置，运动到锁存位置，电机停止。

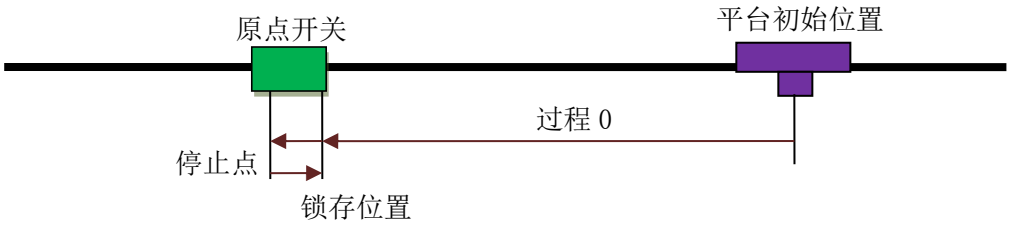


图 2.15 原点锁存回零方式示意图

方式 7：原点锁存加同向 EZ 锁存

此模式先以方式 6 的方式执行一次原点锁存回零，完成后继续沿设定回零方向运行到 EZ 信号产生，EZ 信号产生时锁存当前位置并执行减速停，电机减速停止之后再反向回找 EZ 的锁存位置，运动到锁存位置，电机停止。回原点过程如图 2.16 所示。

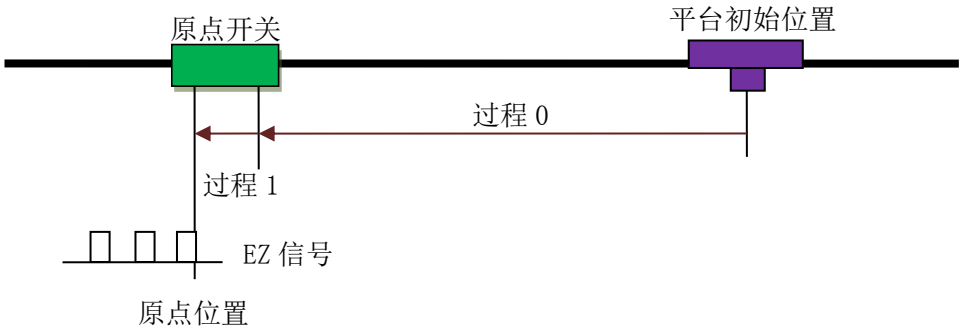


图 2.16 原点锁存加同向 EZ 锁存回零方式示意图

方式 8：单独记一个 EZ 锁存

在回零过程中检测到 EZ 有效边沿出现，锁存当前位置，执行减速停，电机减速停止之后再反向回找 EZ 的锁存位置，运动到锁存位置，电机停止。回原点过程如图 2.17 所示。

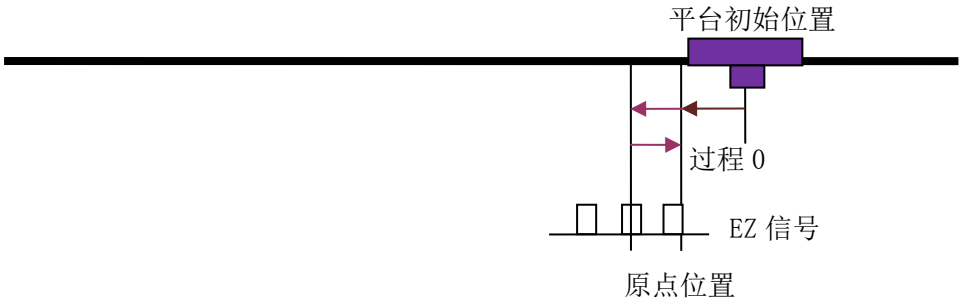


图 2.17 单独记一个 EZ 锁存回零方式示意图

方式 9：原点锁存加反向 EZ 锁存

此模式先以模式 6 的方式执行一次原点锁存回零，完成后以与设定回零方向相反的方向运行到 EZ 信号产生，EZ 信号产生时锁存当前位置并执行减速停，电机减速停止之后再反向回找 EZ 的锁存位置，运动到锁存位置，电机停止。回原点过程如图 2.18 所示。

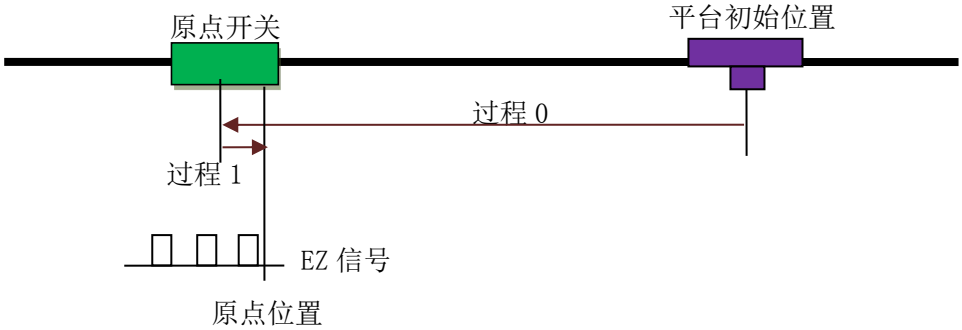


图 2.18 原点锁存加反向 EZ 锁存进行回零

相关指令：

名称	功能	参考
SMCSetHomePinLogic	设置 ORG 原点信号有效电平参数	4.8 节
SMCGetHomePinLogic	读取 ORG 原点信号有效电平参数	
SMCSetHomemode	设置回原点模式参数	
SMCGetHomemode	读取回原点模式参数	
SMCSetEZCount	设置回零 EZ 个数	
SMCGetEZCount	回读回原点 EZ 个数	
SMCSetHomePositionUnit	设置回原点完成后偏移位置值	
SMCGetHomePositionUnit	读取回原点完成后偏移位置值	
SMCSetHomeProfileUnit	设置回原点速度参数值	
SMCGetHomeProfileUnit	读取回原点速度参数值	
SMCHomeMove	启动回原点运动	
SMCGetHomeResult	读取回原点运动状态	

例　程：轴 0 执行回原点运动。

```
''''''''''''''''''''''变量定义''''''''''''''''''''
Dim Myaxis,logic,mode,state,result
Dim Low_Vel,enable,position,home_tacc,home_speed
Myaxis = 0'轴号
logic = 0'ORG 信号的有效电平为低 0
home_dir=0'回零方向，0：负向，1：正向
mode=1'回原点模式 1：一次回原点+反找
Low_Vel=500'回原点启始速度值
home_speed =2000'回原点速度值
enable =1'回原点完成后使能设置位置
position=100'回原点完成后设置位置值
home_tacc =0.1'回原点加减速时间
''''''''''''''''''''指令调用执行''''''''''''''''''
'第一步、设置 ORG 有效电平为低电平
```

```
result=SMCSetHomePinLogic (Myaxis,logic,0)
'第二步、设置轴 0 回原点模式、方向等参数
result=SMCSetHomemode(Myaxis, home_dir,1,mode,0)
'第三步、设轴 0 回原点速度参数
result= SMCSetHomeProfileUnit(Myaxis,Low_Vel,home_speed, home_tacc,0)
'第四步、设轴 0 回原点完成后设置位置值 100
result= SMCSetHomePositionUnit(Myaxis, enable,position)
'第五步、启动轴 0 回原点
result= SMCHomeMove(Myaxis)                    '启动轴 0 回原点
'第六步、读取并打印回原点状态
While SMCCheckDone(0) = 1'等待运动停止
      result=SMCGetHomeResult(Myaxis ,state)    '读取回原点运动状态
      Print state                                '打印状态值，0 未完成，1 完成
wend
```

运行结果：当原点信号有效电平为低电平，回原点运动反向低速运行，再次回到原点时，回原点运动停止。一般回原点信号输入点是连接到运动平台上的原点传感器上。

2.2.3 PVT 运动

雷赛控制器共提供四种 PVT 模式，分别为 PTT、PTS、PVT、PVTS 模式。其中 PTT、PTS 运动模式用于单轴速度规划功能，PVT、PVTS 运动则用于多轴轨迹规划功能。当用户需要规划一些特殊的运动轨迹而使用单轴运动及插补运动无法满足需求时，可以尝试使用 PVT 来规划自己的运动轨迹，根据实际需求选择适合的 PVT 模式。

相关指令：

名称	功能	参考
SMCPttTableUnit	向指定数据表传送数据，采用 PTT 模式	4.9 节
SMCPtsTableUnit	向指定数据表传送数据，采用 PTS 模式	
SMCPvtTableUnit	向指定数据表传送数据，采用 PVT 模式	
SMCPvtsTableUnit	向指定数据表传送数据，采用 PVTS 模式	
SMCPvtMove	启动 PVT 运动	

2.2.3.1 单轴速度规划功能

雷赛控制器中提供了两种 PVT 模式来实现单轴速度规划，分别为 PTT 运动模式和 PTS 运动模式。

（a）PTT 运动模式

PTT 模式中第一个字母 P 表示位置（Position）、第二个字母 T 表示时间（Time），最后一个字母 T 表示梯形（Trapezoid）；PTT 模式表示在梯形速度曲线下规划点位运动。

⚠ 注意：（1）下载的第一组（即起始点）数据中位置、时间必须为 0；数组中的数据都是以起始点的数据为参考点。

（2）调用该指令向数据表中传递数据时，会删除数据表中原先的数据，因此所有数据应当一次传送完毕。如果使用数据表的轴正在运动，禁止更新数据表。

例　程：执行 PTT 模式运动。需运行图 2.19 的 PTT 规划曲线。

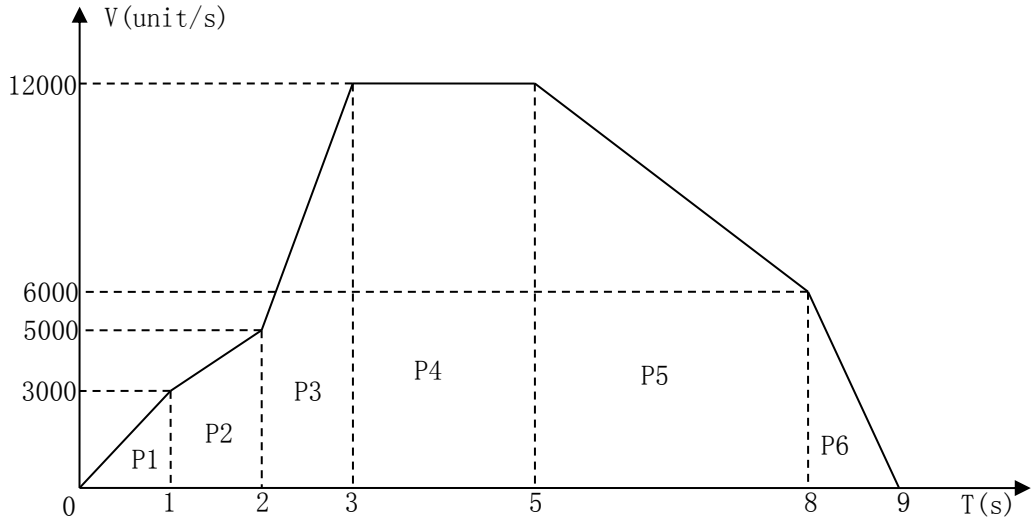


图 2.19 PTT 模式下的 V-T 曲线规划

通过图形我们首先计算各段的位移量，即速度曲线和时间轴所围面积：P1=1500(unit)，P2=4000(unit)，P3=8500(unit)，P4=24000(unit)，P5=27000(unit)，P6=3000(unit)。

将各段位移量累加，得到 PTT 模式下各点的位置和时间数据，如表图 2.20 所示。

序号	位置 P(unit)	时间 T(s)
0	0	0
1	1500	1
2	5500	2
3	14000	3

4	38000	5
5	65000	8
6	68000	9

图 2.20 PTT 模式数组数据

程序如下：

```
''''''''''''''''''''''变量定义''''''''''''''''''''''
Dim MyCardNo,MyAxisNum,My_AxisList(1),MyCount,result
Dim MyPTime(7)                    '定义 PTT 的时间数组
Dim MyPPos(7)                     '定义 PTT 的位置数组
My_AxisList(0) = 0'0 号轴参与 PTT 运动
MyCount = 7'有 7 组数据
MyPPos(0) = 0'定义 PVT 数组数据 0
MyPTime(0) = 0
MyPPos(1) = 1500'定义 PVT 数组数据 1
MyPTime(1) = 1
MyPPos(2) = 5500'定义 PVT 数组数据 2
MyPTime(2) = 2
MyPPos(3) = 14000'定义 PVT 数组数据 3
MyPTime(3) = 3
MyPPos(4) = 38000'定义 PVT 数组数据 4
MyPTime(4) = 5
MyPPos(5) = 65000'定义 PVT 数组数据 5
MyPTime(5) = 8
MyPPos(6) = 68000'定义 PVT 数组数据 6
MyPTime(6) = 9
MyAxisNum = 1'参与 PVT 运动的轴数为 1
''''''''''''''''''''''指令调用执行''''''''''''''''''''''
'第一步、以 PTT 描述方式，向 0 号轴传送 PVT 数据
result=SMCPttTableUnit(My_AxisList(0), MyCount, MyPTime(0), MyPPos(0))
```

```
'第二步、启动 PTT 运动
result=SMCPvtMove(MyAxisNum,My_AxisList(0))
```

位置曲线、加速度曲线见下图 2.21、图 2.22 所示

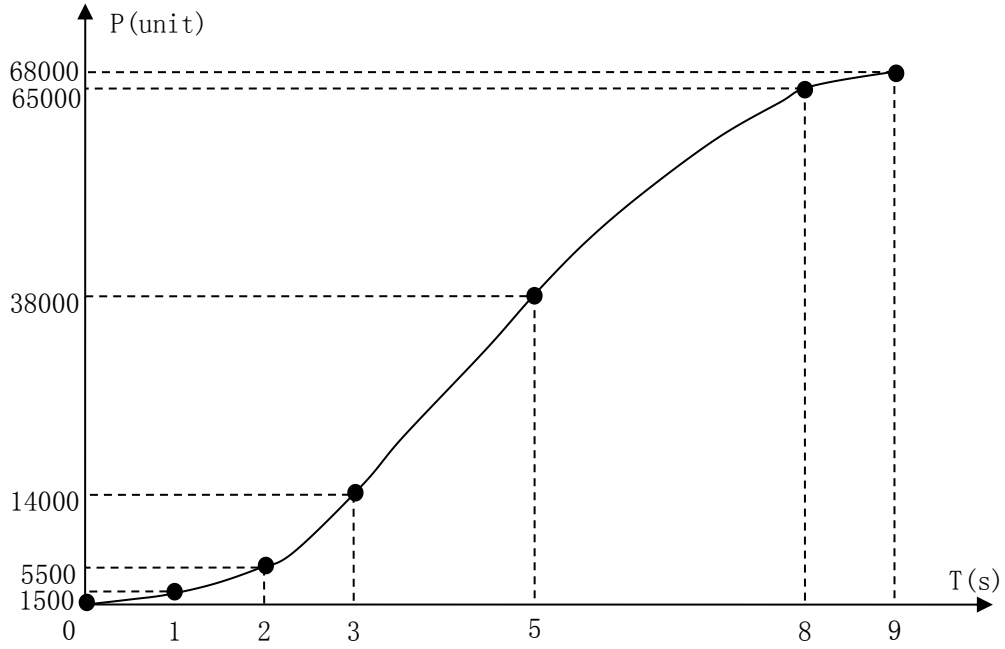


图 2.21 PTT 运动得到的位移曲线

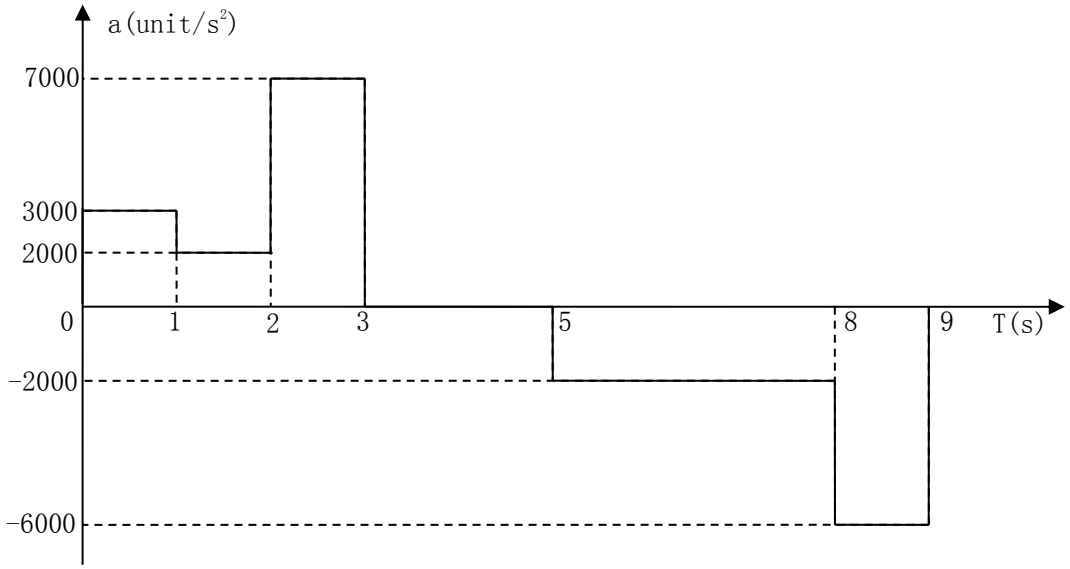


图 2.22　PTT 运动的加速度曲线

（b）PTS 运动模式

PTS 运动模式是 PTT 的扩展功能模式。PTS 的最后一个字母 S 表示 S 形速度曲线；PTS 模式是在 S 形速度曲线下规划点位运动；和 PTT 模式相比，其各段速度过渡更加平滑。

用户通过输入一系列位置、时间、百分比参数，自定义单轴的运动规律。其中位置、时间参数定义和 PTT 模式相同；“百分比”参数是指：相邻 2 个数据点之间加速度的变化时间占速度变化时间的百分比。

以图 2.23 为例说明。数据点 P2 和 P3 之间加速度不变，因此数据点 P2 的百分比为 0。数据点 P3 和 P4 之间加速度变化时间为 2×ta，速度变化时间为 2×ta+te，因此数据点 P3 的百分比为[2×ta/(2×ta+te)]×100%。

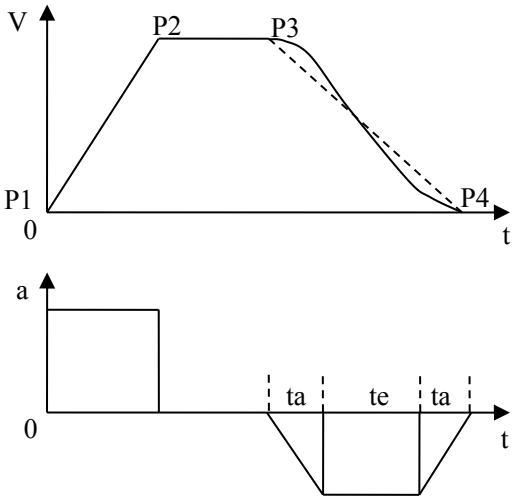


图 2.23 加速度变化时间百分比定义

调整百分比参数，可以改变 S 形速度曲线的形状。如图 2.23 所示，当数据点 P3 的百分比为 0 时，数据点 P3 和 P4 之间的速度曲线为直线（如图 2.23 中虚线所示）；当数据点 P3 的百分比不为 0 时，数据点 P3 和 P4 之间的速度曲线为 S 形曲线（如图 2.23 中实线所示）。“百分比”参数越大，S 段曲线越长。

⚠注意：（1）下载的第一组（即起始点）数据中位置、时间必须为 0；数组中的数据都是以起始点的数据为参考点。

（2）调用该指令向数据表中传递数据时，会删除数据表中原先的数据，因此所有数据应当一次传送完毕。如果使用数据表的轴正在运动，禁止更新数据表。

例　程：执行 PTS 模式运动。运行参数见下图 2.24。

序号	P(unit)	T(s)	Percent(%)
0	0	0	0
1	1500	1	20
2	5500	2	40

3	14000	3	60
4	38000	5	0
5	65000	8	20
6	68000	9	80

图 2.24 PTS 运行参数

程序如下：

```
''''''''''''''''''''''变量定义''''''''''''''''''''''
Dim MyCardNo, MyAxisNum, My_AxisList(1),MyCount,result
Dim MyPTime(7) ,MyPPer(7)     '定义 PTT 的时间数组、百分比数组
Dim MyPPos(7)                 '定义 PTT 的位置数组
My_AxisList(0) = 0'0 号轴参与 PTT 运动
MyCount = 7'有 7 组数据
MyPPos(0) = 0'定义 PVT 数组数据 0
MyPTime(0) = 0
MyPPer(0)=0
MyPPos(1) = 1500'定义 PVT 数组数据 1
MyPTime(1) = 1
MyPPer(1)=20
MyPPos(2) = 5500'定义 PVT 数组数据 2
MyPTime(2) = 2
MyPPer(2)=0
MyPPos(3) = 14000'定义 PVT 数组数据 3
MyPTime(3) = 3
MyPPer(3)=60
MyPPos(4) = 38000'定义 PVT 数组数据 4
MyPTime(4) = 5
MyPPer(4)=0
MyPPos(5) = 65000'定义 PVT 数组数据 5
MyPTime(5) = 8
```

```
MyPPer(5)=20
MyPPos(6) = 68000'定义 PVT 数组数据 6
MyPTime(6) = 9
MyPPer(6)=80
MyAxisNum = 1'参与 PVT 运动的轴数为 1
"""""""""""""""""""""指令调用执行"""""""""""""""""""""
'第一步、以 PTS 描述方式，向 0 号轴传送 PVT 数据
result=SMCPtsTableUnit(My_AxisList(0), MyCount, MyPTime(0), MyPPos(0),MyPPer(0))
'第二步、启动 PTS 运动
result=SMCPvtMove(MyAxisNum,My_AxisList(0))
```

运行结果:与 PTT 相比，PTS 运行中过冲小，速度平滑。大致加速度曲线、位置曲线、速度曲线如下图 2.25、图 2.26、图 2.27：

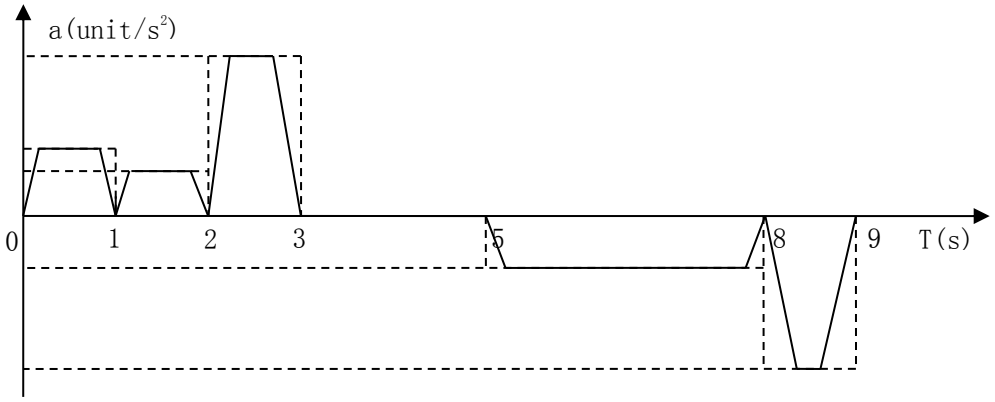


图 2.25　PTS 模式下的加速度曲线

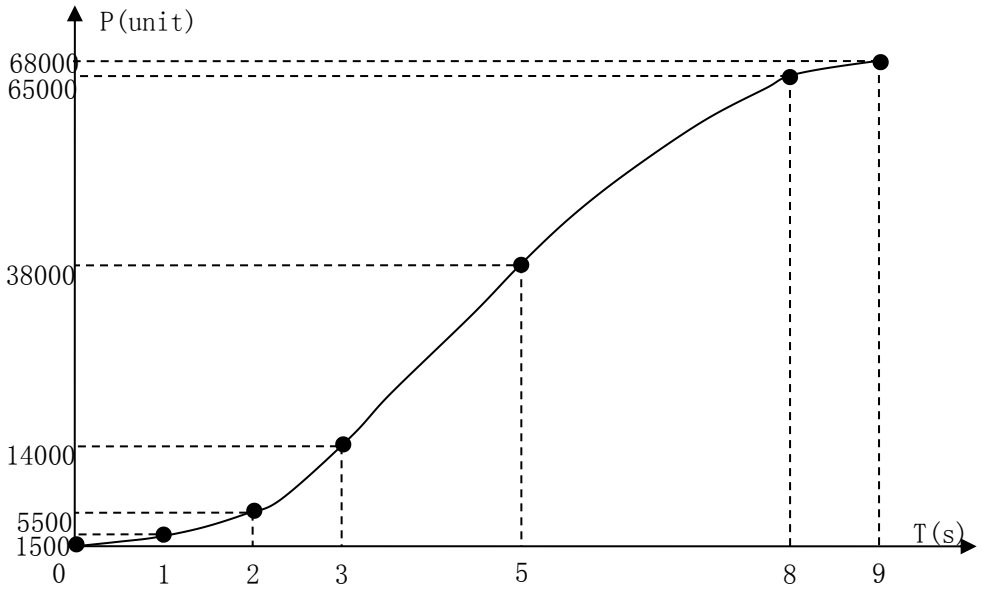
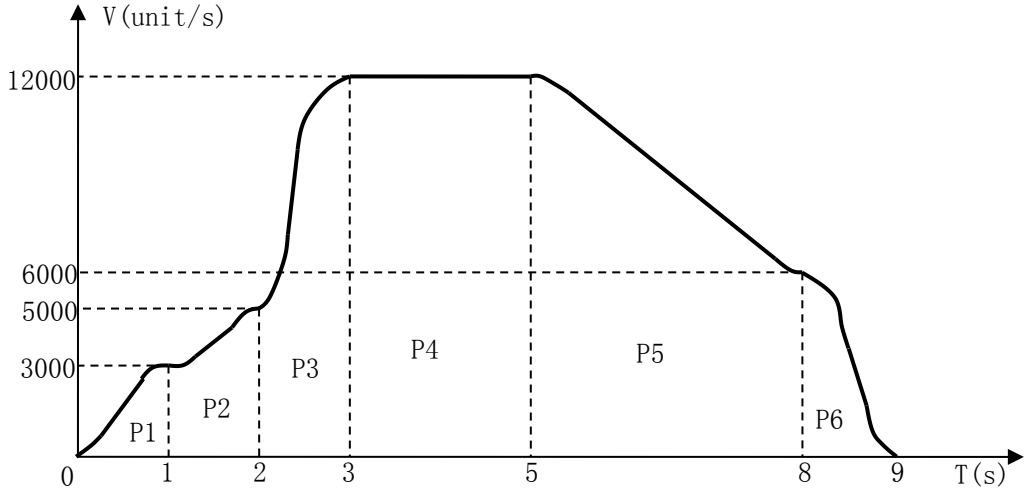


图 2.26 PTS 运动得到的位移曲线



2.27　PTS 运动得到的速度曲线

2.2.3.2 多轴高级轨迹规划功能

雷赛控制器提供的两种多轴轨迹规划的功能分别为 PVT、PVTS 运动模式。当直线插补、圆弧插补不能满足轨迹规划的需求时，雷赛控制器可以使用上述两种高级 PVT 运动功能。PVT 和 PVTS 第二字母 V 表示速度（Velocity），最后一个字母 S 表示平滑。

PVT 模式用于对各点的位置、速度、时间都有要求的轨迹规划；PVTS 模式用于只对各点的位置、时间有要求，而对各点的速度无严格要求的轨迹规划。

（a）PVT 运动模式

PVT 模式使用一系列数据点的位置、速度、时间参数自定义运动规律。

雷赛控制器采用 3 次样条插值算法对位置、速度和时间参数进行曲线拟合。即位置、速度曲线满足 3 次多项式函数关系。

位移方程为：　$p = at^3 + bt^2 + ct + d$

速度方程为：　$v = \frac{dp}{dt} = 3at^2 + 2bt + c$

如果给定轨迹上的一组“位置、速度、时间”参数，即可用 3 次样条函数逼近该轨迹。如图 2.46 所示为采用 PVT 模式拟合平面椭圆的轨迹图。

⚠注意：当设置的各点 P、V、T 数据不合理时，很难得到理想的轨迹曲线。理想轨迹上取点越多，实际轨迹越接近理想轨迹。

例　程：执行 PVT 模式运动。需求控制运动平台按椭圆轨迹运动，使用 PVT 函数自己设计该

轨迹。设该椭圆的长半轴长 9000unit，短半轴长 7000unit；椭圆轨迹的角速度ω恒定，轨迹运动的总时间为 10s。

编程步骤：

1、根据提示可知该椭圆的方程为：

$$\begin{cases} x = 9000\cos(\theta) + 9000 \\ y = 7000\sin(\theta) \end{cases}$$

2、将该轨迹分成圆弧角相等的十段轨迹，如图 2.28 所示；计算各点坐标值，即得 P 值。轨迹分段图如下：

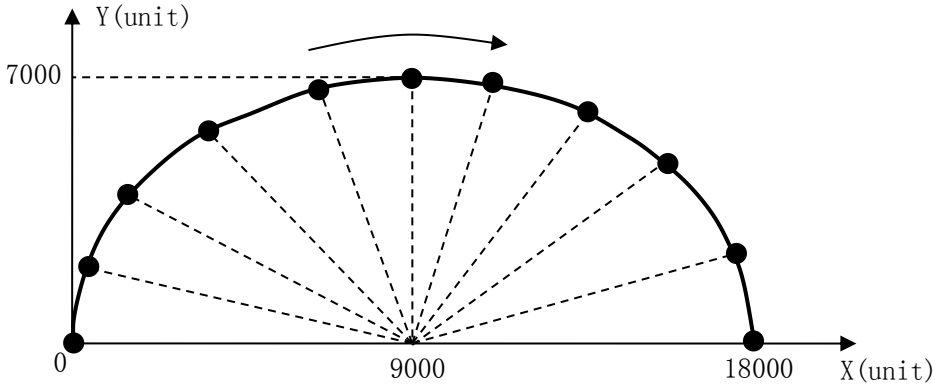


图 2.28 轨迹参数图

程序如下：

```
undim *                                  '删除之前定义的所有变量及数组
    Dim MyPPosX(11),MyPPosY(11)   '定义数组, 用于存储 PVT 的位置数据 X\Y 轴)
    Dim a, b, i
    a = 9000
    b = 7000'定义椭圆长半轴、短半轴长
    For i = 0 To 10'计算各点的 X、Y 坐标
        MyPPosX(i) = a * Cos((10 - i) * 3.14159 / 10) + a
        MyPPosY(i) = b * Sin((10 - i) * 3.14159 / 10)
    Nexti
```

3、根据各点坐标（即 P 值），计算出各点对应的速度的（V 值）和时间（T 值）。导入椭圆方程式求导，可得 X、Y 轴方向的速度分量为：

上式中 $\frac{d\theta}{dt}$ 即为角速度ω。程序如下：

```
Dim MyPVelX(11)    '定义数组，用于存储 PVT 中的速度数据（X 轴）
Dim MyPTimeX(11)   '定义数组，用于存储 PVT 中的时间数据（X 轴）
Dim MyPVelY(11)    '定义数组，用于存储 PVT 中的速度数据（Y 轴）
Dim MyPTimeY(11)   '定义数组，用于存储 PVT 中的时间数据（Y 轴）
Dim MyWVel          ' 定义角速度
For i = 0 To 10
    MyPTimeX(i) = i    ' 存储 X 轴各点时间数据
    MyPTimeY(i) = i    ' 存储 Y 轴各点时间数据
Next i
MyWVel = -3.14159 / 10'计算角速度
MyPVelX(0) = 0
MyPVelX(10) = 0'起始点与终止点 X 轴速度设为 0
MyPVelY(0) = 0
MyPVelY(10) = 0'起始点与终止点 Y 轴速度设为 0
For i = 0 To 8
    MyPVelX(i + 1) = -a * Sin((10 - i - 1) * 3.14159 / 10) * MyWVel
'计算其他点 X 轴速度
    MyPVelY(i + 1) = b * Cos((10 - i - 1) * 3.14159 / 10) * MyWVel
'计算其他点 Y 轴速度
Next i
```

计算结果如下表：

	X 轴			Y 轴		
序号	P(unit)	V(unit/s)	T(s)	P(unit)	V(unit/s)	T(s)
0	0	0	0	0	0	0
1	440	873.731	1	2163	2091.479	1
2	1719	1661.927	2	4115	1779.117	2
3	3710	2287.443	3	5663	1292.603	3

4	6219	2689.048	4	6657	679.560	4
5	9000	2827.431	5	7000	-0.003	5
6	11781	2689.046	6	6657	-679.566	6
7	14290	2287.438	7	5663	-1292.608	7
8	16281	1661.921	8	4114	-1779.120	8
9	17560	873.724	9	2163	-2091.481	9
10	18000	0	10	0	0	10

4、使用 SMCPvtTableUnit 指令向数据表传递数组数据后，再执行 PVT 启动。

```
Dim My_AxisList(2) '定义 PVT 运动的轴列表变量
Dim MyCountX                    '定义 X 轴的 PVT 数据点编号变量
Dim MyCountY                    '定义 Y 轴的 PVT 数据点编号变量
My_AxisList(0) = 0
My_AxisList(1) = 1'0、1 号轴(即 X、Y 轴)参与 PVT 运动
MyCountX = 11
MyCountY = 11'11 组数据
result=SMCPvtTableUnit(My_AxisList(0),MyCountX, MyPTimeX(0), MyPPosX(0), MyPVelX(0))
' 以 PVT 描述方式向 X 轴传送 PVT 数据
result=SMCPvtTableUnit(My_AxisList(1), MyCountY, MyPTimeY(0), MyPPosY(0), MyPVelY(0))
' 以 PVT 描述方式向 Y 轴传送 PVT 数据
Dim My_AxisNum
My_AxisNum = 2'参与 PVT 运动的轴数为 2
result=SMCPvtMove(My_AxisNum, My_AxisList(0)) '启动两轴 PVT 运动
```

运行结果轨迹图如下图 2.29：

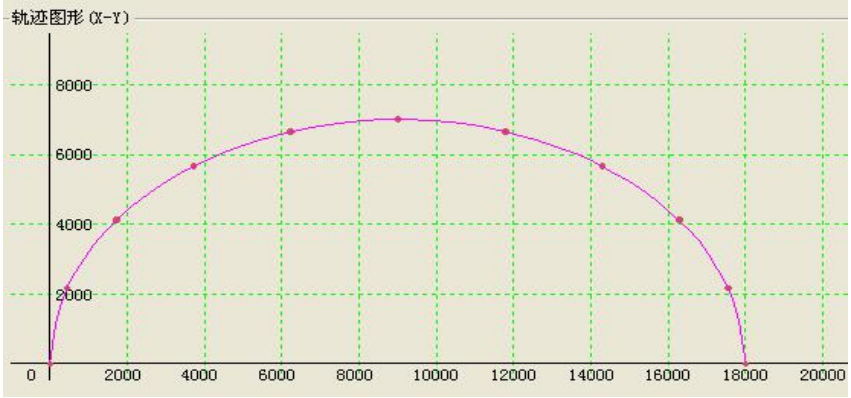


图 2.29　PVT 模式运动得到的上半椭圆轨迹

（b）PVTS 运动模式

PVTS 运动模式是 PVT 模式的简化模式。PVTS 运动模式只需要定义各数据点的位置、时间参数，以及起点速度和终点速度。运动控制器根据各数据点的位置、时间参数计算运动轨迹的速度，确保各数据点速度连续和加速度连续。由于对各点速度没有特殊要求，因此使用 PVTS 运动模式可以得到更平滑的运动轨迹。

例　程：PVTS 模式运动. 设计一空间圆弧轨迹,其半径 R=15000unit，其映射在 XY 平面上的轨迹与 X 轴的夹角α=π/6，轨迹运动总时间为 10s。大体轨迹图如下图 2.30：

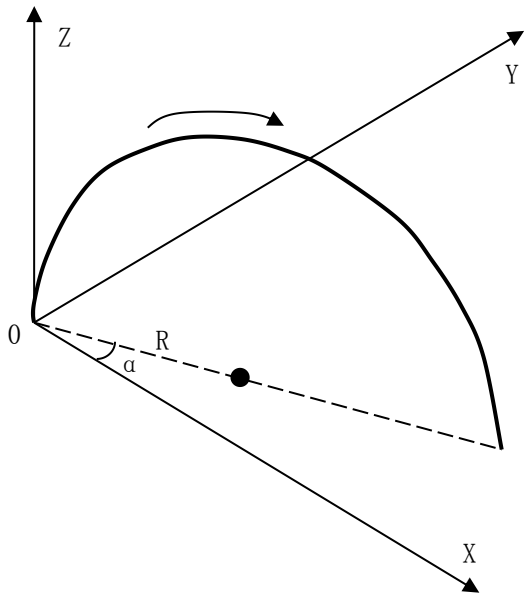


图 2.30　空间圆弧轨迹

编程步骤：

1、根据提该空间圆弧的方程为：

$$\begin{cases} x = 15000\cos(\frac{\pi}{6})\cos(\theta) + 15000\cos(\frac{\pi}{6}) \\ y = 15000\sin(\frac{\pi}{6})\cos(\theta) + 15000\sin(\frac{\pi}{6}) \\ z = 15000\sin(\theta) \end{cases}$$

2、根据该空间圆弧的方程，我们将轨迹分为 10 段，计算出各点的位置：

```
Dim MyPPosX(11),MyPPosY(11) ,MyPPosZ(11)
Dim MyCountX,MyCountY,MyCountZ
Dim R,I                          '定义空间圆弧半径，循环变量
R = 15000'定义圆半径
```

```
MyCountX = 11'定义轨迹点数
MyCountY = 11
MyCountZ =11'设置 X、Y、Z 轴的数据点数
For i = 0To10
    MyPPosX(i) = R * Cos(pi / 6) * Cos((10 - i) * pi / 10) + R * Cos(pi / 6)
'计算 X 轴各点位置坐标
    MyPPosY(i) = R * Sin(pi / 6) * Cos((10 - i) * pi / 10) + R * Sin(pi / 6)
'计算 Y 轴各点位置坐标
    MyPPosZ(i) = R * Sin((10 - i) * pi / 10) '计算 Z 轴各点位置坐标
Next i
```

2、根据提示，每段数据时间为 1s，共 10 段。

```
Dim MyPTimeX(11), MyPTimeY(11),MyPTimeZ(11),i
For i = 0 To 10
    MyPTimeX(i) = i              '计算 X 轴各点时间
    MyPTimeY(i) = i              '计算 Y 轴各点时间
    MyPTimeZ(i) = i              '计算 Z 轴各点时间
Next i
```

3、根据提示，每段数据时间为 1s，共 10 段。

```
Dim MyPVelBeginX,MyPVelEndX
Dim MyPVelBeginY,MyPVelEndY
Dim MyPVelBeginZ,MyPVelEndZ
Dim My_AxisNum,My_AxisList(3)
MyPVelBeginX = 0'X 轴的起点速度及终点速度为 0
MyPVelEndX = 0
MyPVelBeginY = 0'Y 轴的起点速度及终点速度为 0
MyPVelEndY = 0
MyPVelBeginZ = 0'Z 轴的起点速度及终点速度为 0
MyPVelEndZ = 0
My_AxisList(0) = 0'0、1、2 号轴（即 X、Y、Z 轴）参加 PVT 运动
My_AxisList(1) = 1
```

```
My_AxisList(2) = 2
SMCPvtsTableUnit(My_AxisList(0), MyCountX, MyPTimeX(0), MyPPosX(0), MyPVelBeginX,
MyPVelEndX ) '以 PVTS 描述方式向 X 轴传送 PVT 数据
SMCPvtsTableUnit(My_AxisList(1), MyCountY, MyPTimeY(0), MyPPosY(0), MyPVelBeginY,
MyPVelEndY ) '以 PVTS 描述方式向 Y 轴传送 PVT 数据
SMCPvtsTableUnit(My_AxisList(2), MyCountZ, MyPTimeZ(0), MyPPosZ(0), MyPVelBeginZ,
MyPVelEndZ ) '以 PVTS 描述方式向 Z 轴传送 PVT 数据
My_AxisNum = 3'3 个轴参与 PVT 运动
SMCPvtMove(My_AxisNum, My_AxisList(0))                    '启动 PVT 运动
```

运动结果：以 PVTS 模式得到的空间圆弧轨迹如下图 2.31 所示。

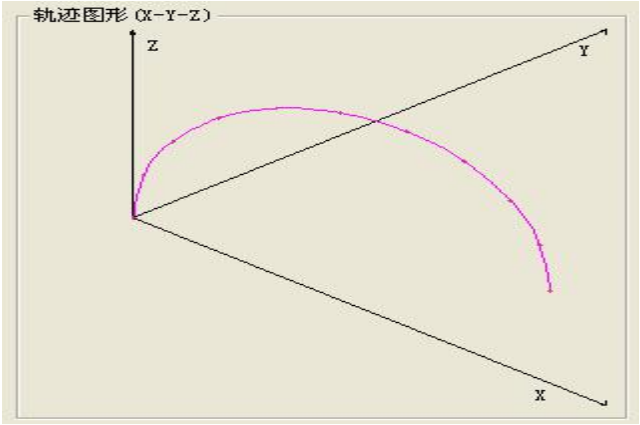


图 2.31 PVTS 空间圆弧运行轨迹图

2.2.4 插补运动

雷赛控制器可进行多轴的直线插补、圆弧插补、三轴螺旋线插补、空间圆弧插补等。

2.2.4.1 参数设置

插补运动的速度、加减速时间、平滑 S 段时间等参数的设置，由不同的函数实现。见图 2.32：

“起始速度”：设置插补运动的起始速度。

“插补速度”：设置插补运动时的最大运行速度，为插补轴合速度值。

“终止速度”：设置插补运动的停止速度。

“加速时间”：设置插补运动时从起始速度加速到最大运行速度需要的时间（Tacc）。

“减速时间”：设置插补运动时从最大运行速度减速到停止速度需要的时间（Tdec）。

“S 段时间”：设置插补合速度曲线 S 段的时间参数（spara）。

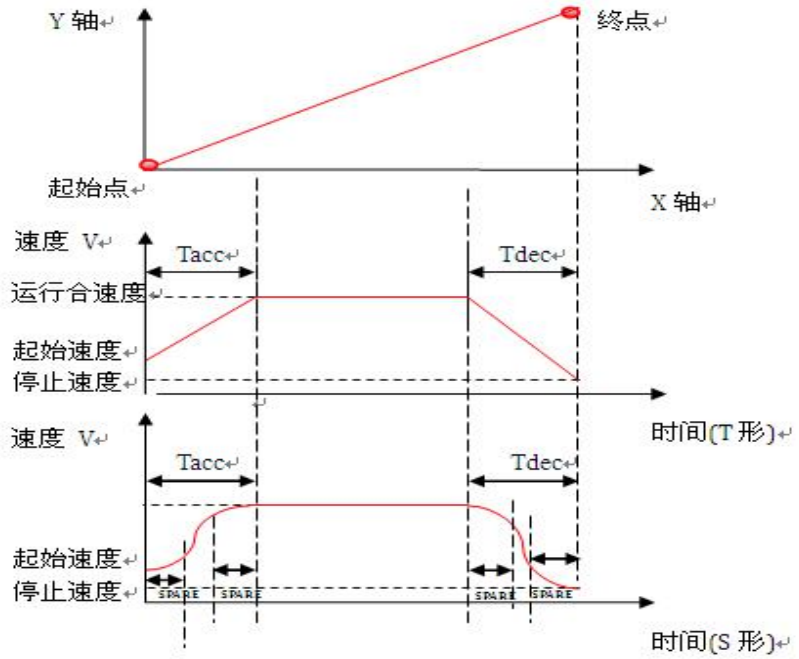


图 2.32 T 形、S 形速度曲线及对应位置图

相关指令：

名称	功能	参考
SMCSetVectorProfileUnit	设置插补运动速度等参数	4.10 节
SMCGetVectorProfileUnit	读取插补运动速度等参数	
SMCSetVectorSprofile	设置插补运动速度曲线的 S 段时间	
SMCGetVectorSprofile	读取插补运动速度曲线的 S 段时间	
SMCSetVectorDecStopTime	设置插补异常减速停止参数	
SMCGetVectorDecStopTime	读取插补异常减速停止参数	

⚠注意：插补速度曲线 S 段平滑时间，若值为 0 则为 T 形曲线，不为 0 则为 S 平滑曲线。

2.2.4.2 单段插补

单段插补指运行轨迹段只有一段，运动轴可多个，且同时启动和停止运动。

（a）单段直线插补

控制器提供了 2 个坐标系的多轴插补，单个插补系最多可同时插补 6 个轴。

相关指令:

名称	功能	参考
SMCLineUnit	启动直线插补运动	4.11 节

例　程：执行 XY 两轴直线插补 ，S 曲线

```
''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2)
Dim Dist(2),MySpara,MyMax_Vel,MyTacc,result
MyCrd = 0'参与插补运动的坐标系 0
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
SMCSetEquiv(0,10)  '设置 X 轴脉冲当量为  10pulse/unit
SMCSetEquiv(1,10)  '设置 Y 轴脉冲当量为  10pulse/unit
MyMax_Vel = 3000'最大速度为 3000unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0.05'平滑时间为 0.05s
MyaxisNum = 2'参与插补运动轴数为 2
Dist(0) = 100'定义 X 轴运动距离为 100unit
Dist(1) = 80'Y 轴运动距离为 80unit
Myposi_mode = 0'插补运动模式为相对坐标模式
''''''''''''''''''''''''''指令调用执行''''''''''''''''''''''''''
'第一步、设置插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第二步、设置平滑 S 参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第三步、启动直线插补运动
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
```

运行插补轨迹图形结果如下图 2.33：

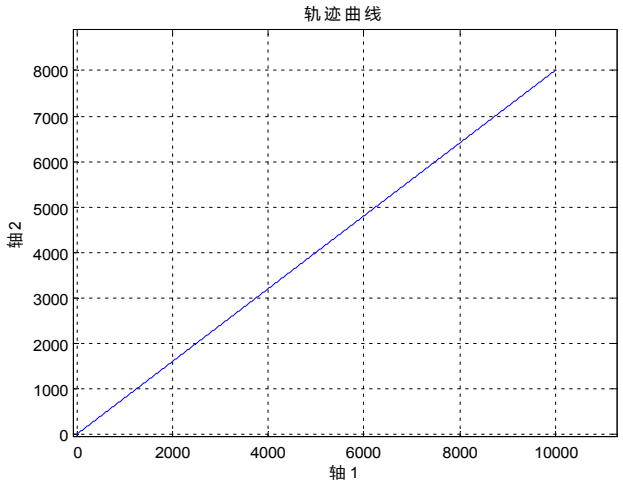


图 2.33 直线插补轨迹

（b）单段圆弧插补

控制器可执行圆弧插补功能，圆弧实现有 3 种实现模式，见图 2.34：

1） 圆心+终点模式，即需知道圆心点（auxpoint）和终点（EndPoint），执行相应指令后运行圆弧轨迹。

2） 半径+终点模式，即需终点圆的半径值 R 和终点（EndPoint），执行相应指令后运行圆弧轨迹。

3） 三点模式，即需知道圆经过 3 个点，起点、中间点（auxpoint）、终点（EndPoint），执行相应指令后运行圆弧轨迹。

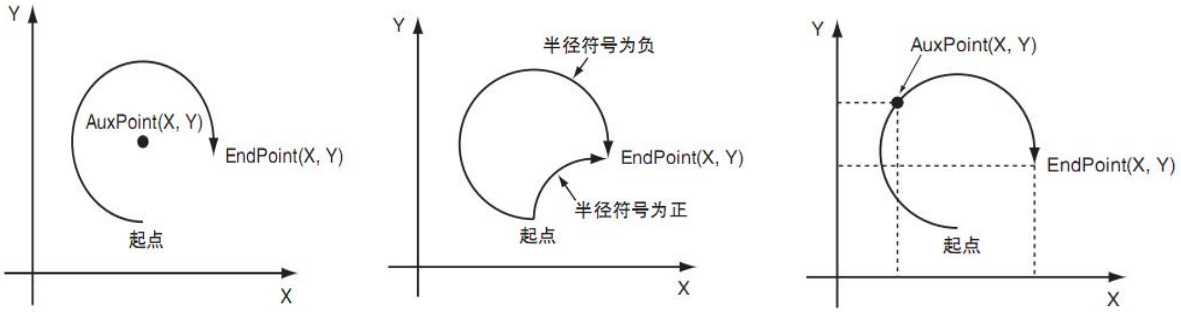


图 2.34 分别对应以上 3 中圆弧模式（起点表示起始位置点）

相关指令：

名称	功能	参考
SMCArcMoveCenterUnit	圆心+终点模式的圆弧插补运动	4.11 节
SMCArcMoveRadiusUnit	半径+终点模式的圆弧插补运动	

SMCArcMove3pointsUnit	三点模式的圆弧插补运动	

例程 1：执行 XY 两轴圆弧顺时针方向插补运动，插补模式为圆心+终点模式

```
'''''''''''''''''''''''''变量定义'''''''''''''''''''''''''
undim *                                    '除之前定义的所有变量及数组
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode,AxisArray(2)
Dim Dist(2),MySpara,MyMax_Vel,MyTacc,cen(2),result,MyCircle,MyArc_Dir
MyCrd = 0'参与插补运动的坐标系 0
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyMax_Vel = 300'最大速度为 300unit/s
MyTacc = 0.3'加减速时间为 0.3s
MySmode = 0'保留参数，固定值为 0
MySpara = 0.1'平滑时间为 0.1
MyaxisNum = 2'参与插补运动轴数为 2
Dist(0) = 50'定义 X 轴运动终点位置
Dist(1) = -50'定义 Y 轴运动终点位置
Cen(0)=50'定义 X 轴运动圆心位置
Cen(1)=0'定义 Y 轴运动圆心位置
Myposi_mode = 0'插补运动模式为相对坐标模式
MyCircle=0'设置圆弧圈数为 0
MyArc_Dir = 0'设置圆弧方向为顺时针
''''''''''''''''''''''''指令调用执行''''''''''''''''''''''''
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)            '设置 X 轴脉冲当量为 10pulse/unit
result=SMCSetEquiv(1,10)            '设置 Y 轴脉冲当量为 10pulse/unit
'第二步、设置插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第三步、设置平滑 S 参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
```

'第四步、启动圆弧插补运动

result=SMCArcMoveCenterUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Cen(0),MyArc_Dir,MyCircle,Myposi_mode)

运行插补轨迹图形结果如下图 2.25：

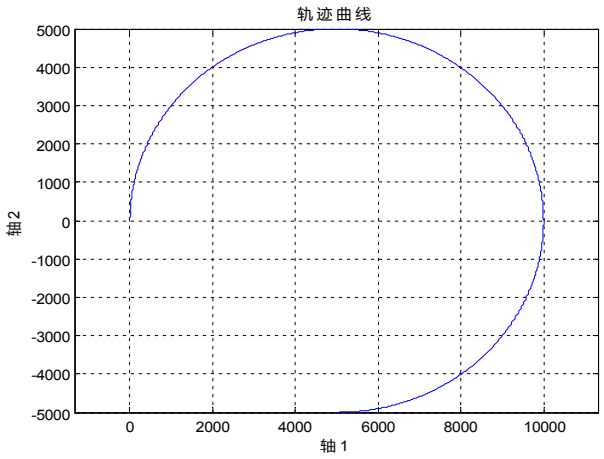


图 2.25 例程圆弧插补轨迹图

例程 2：执行 XY 两轴圆弧逆时针方向插补运动，插补模式为半径+终点模式

```
''''''''''''''''''''''变量定义''''''''''''''''''''
undim *                                    '除之前定义的所有变量及数组
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2)
Dim Dist(2),MySpara,MyMax_Vel,MyTacc,Radius,result,MyCircle,MyArc_Dir
MyCrd = 0'参与插补运动的坐标系 0
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyMax_Vel = 500'最大速度为 500unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0'平滑时间为 0
MyaxisNum = 2'参与插补运动轴数为 2
Dist(0) = 0'定义 X 轴运动终点位置
Dist(1) = 60'定义 Y 轴运动终点位置
Radius=30'定义圆弧半径
Myposi_mode = 0'插补运动模式为相对坐标模式
```

MyCircle=0'设置圆弧圈数

MyArc_Dir = 1'设置圆弧方向为逆时针

''指令调用执行''

'第一步、设置轴脉冲当量值

result=SMCSetEquiv(0,10)　　　　　　'设置 X 轴脉冲当量

result=SMCSetEquiv(1,10)　　　　　　'设置 Y 轴脉冲当量

'第二步、设置插补速度参数

result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)

'第三步、设置平滑 S 参数

result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)

'第四步、启动螺旋插补方向为逆时针

result=SMCArcMoveRadiusUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Radius,MyArc_Dir,MyCircle,My posi_mode)

运行结果：按逆时针方向运行一个圆弧，运行插补轨迹图形结果如下 2.25：

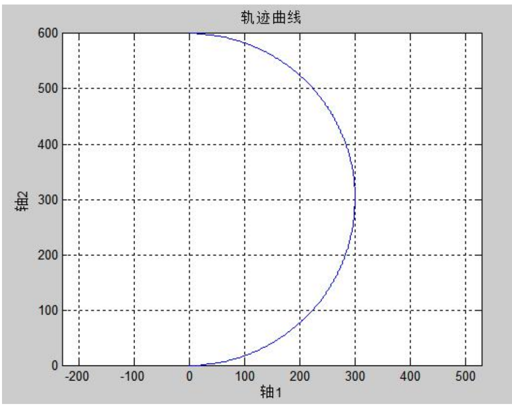


图 2.25 例程 2 圆弧插补轨迹图

例程 3：执行 XY 两轴圆弧插补运动，插补模式为三点成圆模式

''变量定义''

undim *

Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2)

Dim Dist(2),MySpara,MyMax_Vel,MyTacc,Mid(2),result, MyCircle,MyArc_Dir

MyCrd = 0'参与插补运动的坐标系 0

AxisArray(0) = 0'定义插补 0 轴为 X 轴

AxisArray(1) = 1'定义插补 1 轴为 Y 轴

```
MyMax_Vel = 500'最大速度为 500unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0.05'平滑时间为 0.05S
MyaxisNum = 2'参与插补运动轴数为 2
Dist(0) = 15'定义 X 轴运动终点位置
Dist(1) = -10'定义 Y 轴运动终点位置
Mid(0)=10'定义 X 轴中间点位置
Mid(1)=20'定义 Y 轴中间点位置
Myposi_mode = 0'插补运动模式为相对坐标模式
MyCircle=0'设置圆弧圈数
''''''''''''''''''''''''''''''''指令调用执行''''''''''''''''''''''''''''''''
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)              '设置 X 轴脉冲当量
result=SMCSetEquiv(1,10)              '设置 Y 轴脉冲当量
'第二步、设置插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第三步、设置平滑 S 参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第四步、启动圆弧插补
result=SMCArcMove3pointsUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Mid(0),MyCircle,Myposi_mode)
```

运行插补轨迹图形结果如下 2.26：

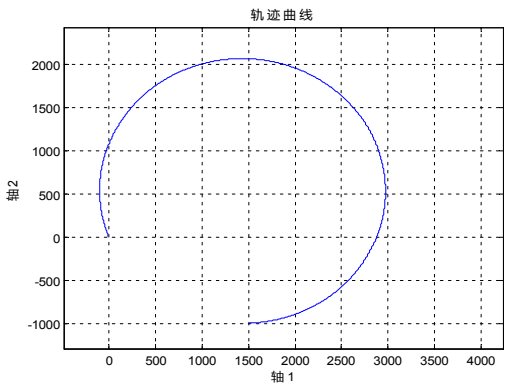


图 2.26 例程 3 圆弧插补轨迹图

例程 4：执行基于 XYZ 三轴空间圆弧的插补运动，插补模式为三点成圆模式

```
"""""""""""""""""""""变量定义"""""""""""""""""""""
undim *
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(3)
Dim Dist(3),MySpara,MyMax_Vel,MyTacc,Mid(3),result
MyCrd = 0'参与插补运动的坐标系 0
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
AxisArray(2) = 2'定义插补 2 轴为 Z 轴
MyMax_Vel = 500'最大速度为 500unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0'平滑时间为 0
MyaxisNum = 3'参与插补运动轴数为 3
Dist(0) = 15'定义 X 轴运动终点位置
Dist(1) = -10'定义 Y 轴运动终点位置
Dist(2) = -15'定义 Z 轴运动终点位置
Mid(0)=10'定义 X 轴中间点位置
Mid(1)=20'定义 Y 轴中间点位置
Mid(2)=15'定义 Z 轴中间点位置
Myposi_mode = 0'插补运动模式为相对坐标模式
MyCircle=-1'该值的绝对值减 1 表示空间圆弧的圈数
"""""""""""""""""""""指令调用执行"""""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)        '设置 X 轴脉冲当量
result=SMCSetEquiv(1,10)        '设置 Y 轴脉冲当量
result=SMCSetEquiv(2,10)        '设置 Z 轴脉冲当量
'第二步、设置插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第三步、设置平滑 S 参数
```

```
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第四步、启动圆弧插补
result=SMCArcMove3pointsUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Mid(0),MyCircle,Myposi_mode)
```

运行插补轨迹图形结果如下图 2.27：

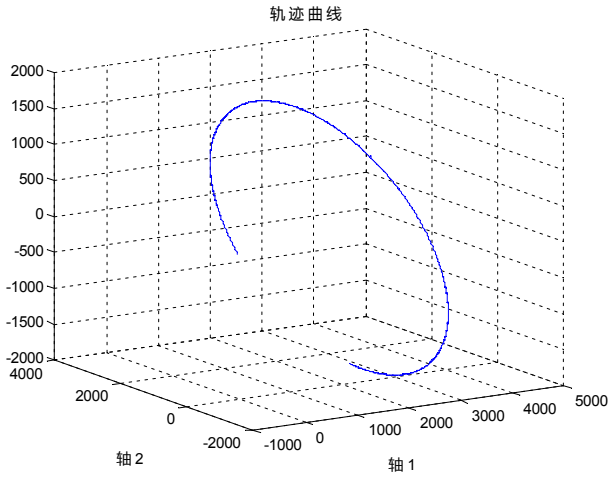


图 2.27 例程 4 空间圆弧插补轨迹图

（c）单段螺旋插补

螺旋插补是基于圆弧插补上完成的，需使用三个轴。螺旋插补与圆弧插补运动指令相同，只是圈数可自由设置。

例　程：执行 XYZ 三轴圆弧顺时针方向插补运动，插补模式为圆心+终点模式

```
"""""""""""""""""""""变量定义"""""""""""""""""""""
undim *
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(3)
Dim Dist(3),MySpara,MyMax_Vel,MyTacc,cen(3),result, MyCircle,MyArc_Dir
MyCrd = 0'参与插补运动的坐标系 0
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
AxisArray(2) = 2'定义插补 2 轴为 Z 轴
MyMax_Vel = 500'最大速度为 500unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0'平滑时间为 0
MyaxisNum = 3'参与插补运动轴数为 3
```

```
Dist(0) = 50'定义 X 轴运动终点位置
Dist(1) = -50'定义 Y 轴运动终点位置
Dist(2) = 100'定义 Z 轴运动终点位置
Cen(0)=50'定义 X 轴运动圆心位置
Cen(1)=0'定义 Y 轴运动圆心位置
Cen(2)=0'定义 Z 轴运动圆心位置
Myposi_mode = 0'插补运动模式为相对坐标模式
MyCircle=4'设置圆弧圈数为 4
MyArc_Dir = 0'设置圆弧方向为顺时针
''''''''''''''''''''''''''''''指令调用执行''''''''''''''''''''''''''''''
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)        '设置 X 轴脉冲当量
result=SMCSetEquiv(1,10)        '设置 Y 轴脉冲当量
result=SMCSetEquiv(2,10)        '设置 Z 轴脉冲当量
'第二步、设置插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第三步、设置平滑 S 参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第四步、启动螺旋插补方向为顺时针
result=SMCArcMoveCenterUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Cen(0),MyArc_Dir,MyCircle,Myposi_mode)
```

运行插补轨迹图形结果如下图 2.28：

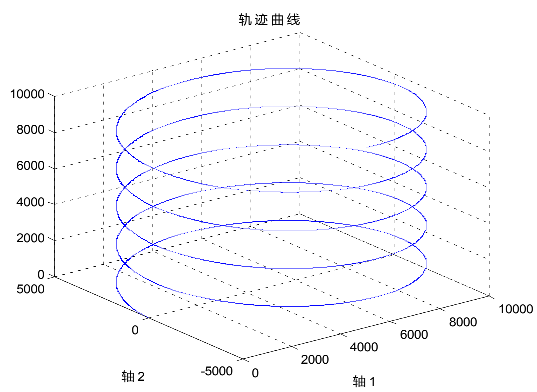


图 2.28 例程螺旋插补轨迹图

（d）单段其它插补

当设定圆弧参数不构成一整圆时，将会出现一些类似渐进线、圆锥等弧形。不规则圆时，最大运行速度可能超出设定值值范围。

例　程：执行 XYZ 三轴螺旋插补 ，参数不构成标准螺旋线，图形类似锥形。

```
"""""""""""""""""""变量定义"""""""""""""""""""
undim *
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(3)
Dim Dist(3),MySpara,MyMax_Vel,MyTacc,Cen(3), MyCircle,MyArc_Dir
MyCrd = 0'参与插补运动的坐标系 0
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
AxisArray(2) = 2'定义插补 2 轴为 Z 轴
MyMax_Vel = 500'最大速度为 500unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0'平滑时间为 0
MyaxisNum = 3'参与插补运动轴数为 3
Dist(0) = 50'定义 X 轴运动终点位置
Dist(1) = -50'定义 Y 轴运动终点位置
Dist(2) =100'定义 Z 轴运动终点位置
Cen(0)=20'定义 X 轴运动圆心位置
Cen(1)=10'定义 Y 轴运动圆心位置
Cen(2)=0'定义 Z 轴运动圆心位置
Myposi_mode = 0'插补运动模式为相对坐标模式
MyCircle=2'设置圆弧圈数为 0
MyArc_Dir = 0'设置圆弧方向为顺时针
"""""""""""""""""""指令调用执行"""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)     '设置 X 轴脉冲当量
result=SMCSetEquiv(1,10)     '设置 Y 轴脉冲当量
```

```
result=SMCSetEquiv(2,10)     '设置 Z 轴脉冲当量为
'第二步、设置插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第三步、设置平滑 S 参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第四步、启动螺旋插补方向为顺时针
result=SMCArcMoveCenterUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Cen(0),MyArc_Dir,MyCircle,My
posi_mode)
```

运行插补轨迹图形结果如下图 2.29：

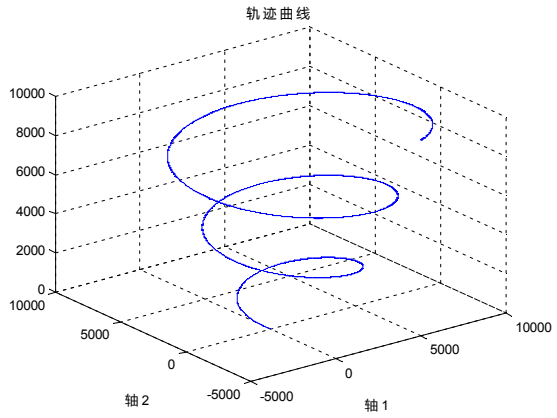


图 2.29 特殊螺旋插补轨迹

2. 2. 4. 3 连续插补

在运动控制中，采用连续插补可实现速度的平滑过渡，减少机器的振动，可以提高机器的加工精度和加工速度。

控制器提供了连续插补运动功能，包含前瞻（模式 1）和非前瞻运动（模式 0、模式 2，两种算法不一样）。连续插补指令支持直线插补，圆弧插补，螺旋线插补，IO 控制等，前瞻和非前瞻主要区别在于前瞻可很好应用于小线段轨迹，其轨迹连接处更平滑。

连续插补对于圆弧提供了圆弧限速功能功能，主要为限制运行速度，使加速度值不超出设定范围。 前瞻（模式 1）默认为圆弧限速启用，非前瞻模式 0 不支持圆弧限速，非前瞻模式 2 可自由设置圆弧限速功能。具体功能见下表：

连续插补功能	**非前瞻(模式 0)**	**前瞻(模式 1)**	**非前瞻(模式 2)**
插补系	4 个	2 个	2 个
长线段轨迹	支持	支持	支持

小线段轨迹	不支持	支持	不支持
圆弧限速	不支持	支持	可自由设定
Blend 功能	可自由设定	不支持	不支持
T\S 型曲线	支持 T、S 型	支持 T 型	支持 T、S 型
加、减速时间	可自由设定	对称曲线	可自由设定
起始、停止速度	可自由设定	不支持	可自由设定
速度倍率	支持(下一轨迹生效)	支持(立即生效)	支持(立即生效)
插补延时	支持	支持	支持
IO 等待输入	支持	支持	支持
IO 立即输出	支持	支持	支持
IO 超前、滞后输出	支持	支持	支持

此外，雷赛控制器支持两个坐标系，每个坐标系的连续缓冲区最多可缓存 5000 条指令。两个坐标系的速度可独立设置，执行连续插补时两个坐标系可独立进行连续插补运动，即可同时进行两组连续插补运动。

实现基本连续插补运动的一般步骤如下：

1)如需要小线段前瞻功能，使用 SMCContiSetLookAheadMode 指令设置前瞻模式参数；

2)使用函数 SMCContiOpenList 打开连续插补缓冲区；

3）使用 SMCSetVectorProfileUnit、SMCSetVectorSprofile 等指令设置连续插补速度参数；

4）编写连续插补运动指令；

5）使用函数 SMCContiStartList 启动连续插补运动；

6）使用函数 SMCContiCloseList 关闭连续插补缓冲区。

需要注意的是，SMCContiSetLookAheadMode 设置前瞻模式参数指令必须在 SMCContiOpenList 打开连续插补缓冲区指令前调用。

连续插补设置相关指令：

名称	功能	参考
SMCContiStartList	开始连续插补	4.12 节
SMCContiCloseList	关闭连续插补缓冲区	
SMCContiPauseList	暂停连续插补	
SMCContiDelay	连续插补中暂停延时指令	

SMCContiStopList	停止连续插补	
SMCContiSetLookAheadMode	设置小线段前瞻模式及参数	
SMCContiGetLookAheadMode	读取小线段前瞻模式及参数	
SMCContiSetBlend	设置 Blend 是否使能	
SMCContiGetBlend	读取插补是否使用了 Blend 功能	
SMCContiChangeSpeedRatio	动态调整连续插补速度比例	

连续插补运动状态相关指令：

名称	功能	参考
SMCContiRemainSpace	读取插补缓存去剩余空间	4.13 节
SMCContiReadCurrentMark	读取连续插补缓冲区当前插补段号	
SMCContiGetRunState	读取连续插补运动状态	

连续插补运动 IO 相关指令：

名称	功能	参考
SMCContiSetPauseOutput	设置连续插补暂停及异常停止时 IO 输出状态	4.14 节
SMCContiGetPauseOutput	读取连续插补暂停及异常停止时 IO 输出状态参数	
SMCContiWaitInput	连续插补等待 IO 输入	
SMCContiDelayOutbitToStart	连续插补中相对于轨迹段起点 IO 滞后输出（段内执行）	
SMCContiDelayOutbitToStop	连续插补中相对于轨迹段终点 IO 滞后输出	
SMCContiAheadOutbitToStop	连续插补中相对于轨迹段终点 IO 提前输出（段内执行）	
SMCContiWriteOutbit	连续插补中缓冲区立即 IO 输出	
SMCContiClearIoAction	清除段内未执行完的 IO 动作	

⚠注意：连续插补轨迹运动指令与单段插补轨迹运动指令一样。

（a）连续插补 Blend 功能

控制器执行多段连续插补时， 当不使用拐角平滑过渡功能时，连续插补运动以位置控制为主要原则，即连续插补运动中的每段轨迹都会运行到目标位置才会进行下一段轨迹运动，此时每段运动的速度曲线都有加减速过程。

如下图为未设置 Blend 拐角平滑功能的两段轨迹及速度曲线示意图 2.30，左图为轨迹曲线，右图为速度曲线。

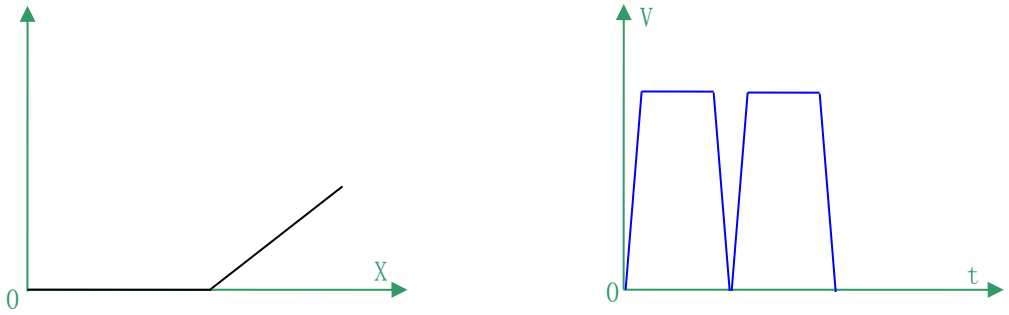


图 2.30 未设置拐角平滑功能的运动轨迹及速度曲线

当设置了拐角平滑过渡功能后，连续插补运动则以速度平滑过渡为主要原则，即连续插补运动中的每段轨迹的速度曲线都是平滑过渡的，但此时各段运动轨迹之间的拐角也是平滑过渡的，拐角弧度的大小由拐角过渡时的速度大小及加减速时间所决定。

如图 2.31 为设置 Blend 拐角平滑功能的轨迹及速度曲线示意图，左图为轨迹曲线，右图为速度曲线。

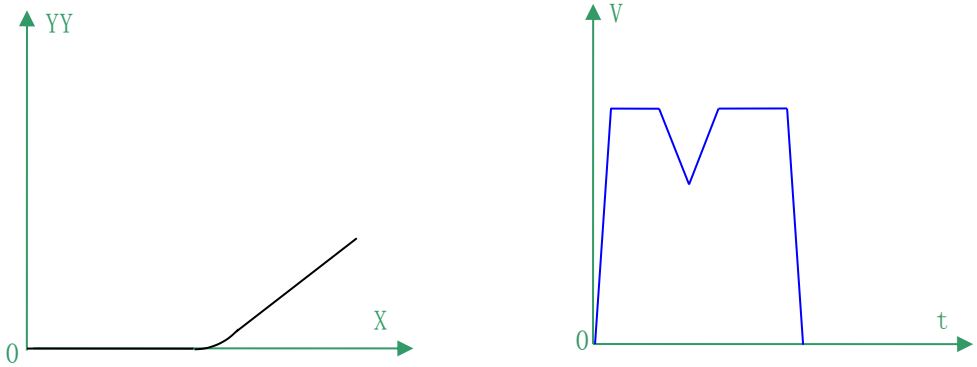


图 2.31 设置拐角平滑功能的运动轨迹及速度曲线

⚠注意：Blend 功能指令需在运动指令前使能，下段指令才会执行拐角平滑。

例程 1：执行直线+圆弧+直线连续插补，禁用 Blend 功能。

```
''''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''''
undim *        '删除之前定义的所有变量及数组
```

```
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2),result
Dim Dist(2),MySpara,MyMax_Vel,MyTacc,Radius,Dist_1(2),Dist_2(2)
MyCrd = 0'参与插补运动的坐标系 0
Myposi_mode = 1'插补运动模式为绝对坐标模式
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyaxisNum = 2'参与插补运动轴数为 2
MyMax_Vel = 300'最大速度为 300unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0.05'平滑时间为 0.05s
Dist(0) = 100'定义直线插补 X 轴运动终点距离
Dist(1) = 50'定义直线插补 Y 轴运动终点距离
Dist_1(0) = 200'定义圆弧 X 轴运动终点距离
Dist_1(1) = 50'定义圆弧 Y 轴运动终点距离
Radius=50'定义圆弧半径距离
Dist_2(0) = 250'定义轨迹 2 直线 X 轴运动终点距离
Dist_2(1) = 0'定义轨迹 2 直线 Y 轴运动终点距离
""""""""""""""""""""指令调用执行""""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10'设置 X 轴脉冲当量为 10pulse/unit
result=SMCSetEquiv(1,10)        '设置 Y 轴脉冲当量为 10pulse/unit
'第二步、设置插补模式，此处设置为模式 0，该模式支持 BLEND
result=SMCContiSetLookAheadMode(MyCrd,0,10,1,100000)
'第三步、打开缓存区
result=SMCContiOpenList(MyCrd,MyaxisNum,AxisArray(0))
'第四步、设置插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第五步、设置平滑 S 参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
```

```
'第六步、启动连续插补
result=SMCContiStartList(MyCrd)
'第七步、禁用插补 Blend 功能
result= SMCContiSetBlend(0,0)   '不启动插补 Blend 功能
'第八步、将直线插补段放入缓存区
result= SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
'第九步、将圆弧插补段放入缓存区
result=SMCArcMoveRadiusUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_1(0),Radius,0,0,Myposi_mode)
'第十步、将直线插补段 2 放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_2(0),Myposi_mode)
'第十一步、关闭直线插补缓存区
result=SMCContiCloseList(MyCrd)
```

运行连续插补无 Blend 功能轨迹图形结果如下图 2.32：

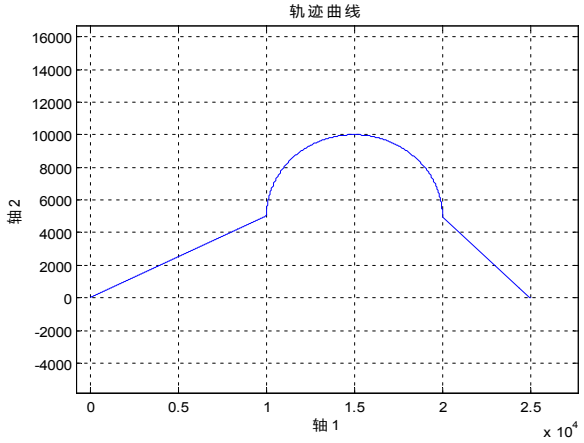


图 2.32 连续插补无 Blend 功能轨迹图

例程 2：执行直线+圆弧+直线连续插补，启用 Blend 功能。

```
'''''''''''''''''''''''''''''''''变量定义'''''''''''''''''''''''''''''''''
undim *                    '删除之前定义的所有变量及数组
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2),result
Dim Dist(2),MySpara,MyMax_Vel,MyTacc,Radius,Dist_1(2),Dist_2(2)
MyCrd = 0'参与插补运动的坐标系 0
Myposi_mode = 1'插补运动模式为绝对坐标模式
AxisArray(0) = 0'定义插补 0 轴为 X 轴
```

```
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyaxisNum = 2'参与插补运动轴数为 2
MyMax_Vel = 300'最大速度为 300unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0.05'平滑时间为 0.05s
Dist(0) = 100'定义直线插补 X 轴运动终点距离
Dist(1) = 50'定义直线插补 Y 轴运动终点距离
Dist_1(0) = 200'定义圆弧 X 轴运动终点距离
Dist_1(1) = 50'定义圆弧 Y 轴运动终点距离
Radius=50'定义圆弧半径距离
Dist_2(0) = 250'定义 2 直线 X 轴运动终点距离
Dist_2(1) = 0'定义 2 直线 Y 轴运动终点距离
""""""""""""""""""""指令调用执行""""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)      '设置 X 轴脉冲当量为 10pulse/unit
result=SMCSetEquiv(1,10)      '设置 Y 轴脉冲当量为 10pulse/unit
'第二步、设置插补模式，此处设置为模式 0，该模式支持 BLEND
result=SMCContiSetLookAheadMode(MyCrd,0,10,1,100000)
'第三步、打开缓存区
result=SMCContiOpenList(MyCrd,MyaxisNum,AxisArray(0))
'第四步、设置插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第五步、设置平滑 S 参数、前瞻模式及相关参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第六步、启动连续插补
result=SMCContiStartList(MyCrd)
'第七步、开启插补 Blend 功能
result=SMCContiSetBlend(0,1)
'第八步、将直线插补段放入缓存区
```

```
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
'第九步、将圆弧插补段放入缓存区
result=SMCArcMoveRadiusUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_1(0),Radius,0,0,Myposi_mode)
'第十步、将直线插补段放入缓存区
result= SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_2(0),Myposi_mode)
'第十一步、关闭缓存区
result=SMCContiCloseList(MyCrd)
```

运行连续插补有 Blend 功能轨迹图形结果如下图 2.33：

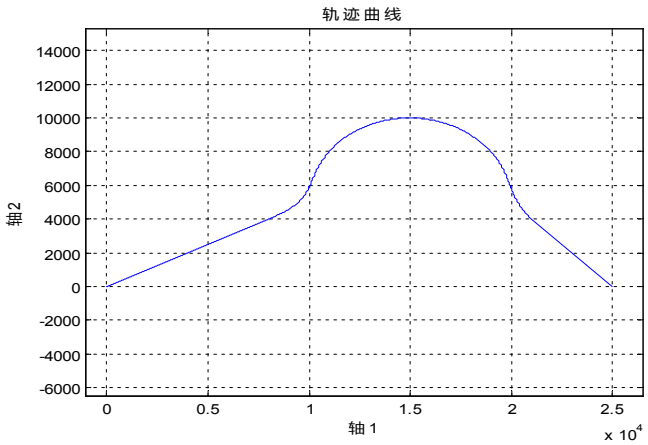


图 2.33 连续插补 Blend 功能轨迹图

例程 3：执行直线+圆弧+直线连续插补，使用小线段前瞻模式。

```
''''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''''
undim *                    '删除之前定义的所有变量及数组
Dim MyCrd,MyaxisNum,Myposi_mode ,AxisArray(2),result
Dim Dist(2),MyMax_Vel,MyTacc,Radius,Dist_1(2),Dist_2(2)
MyCrd = 0'参与插补运动的坐标系 0
Myposi_mode = 1'插补运动模式为绝对坐标模式
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyaxisNum = 2'参与插补运动轴数为 2
MyMax_Vel = 300'最大速度为 300unit/s
MyTacc = 0.1'加减速时间为 0.1s
Dist(0) = 100'定义直线插补 X 轴运动终点距离
```

```
Dist(1) = 50'定义直线插补 Y 轴运动终点距离
Dist_1(0) = 200'定义圆弧 X 轴运动终点距离
Dist_1(1) = 50'定义圆弧 Y 轴运动终点距离
Radius=50'定义圆弧半径距离
Dist_2(0) = 250'定义 2 直线 X 轴运动终点距离
Dist_2(1) = 0'定义 2 直线 Y 轴运动终点距离
""""""""""""""""""""指令调用执行""""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)      '设置 X 轴脉冲当量为 10pulse/unit
result=SMCSetEquiv(1,10)      '设置 Y 轴脉冲当量为 10pulse/unit
'第二步、设置插补模式，此处设置为模式 1，该模式支持小线段前瞻功能
result=SMCContiSetLookAheadMode(MyCrd,1,10,1,100000)
'第三步、打开缓存区
result=SMCContiOpenList(MyCrd,MyaxisNum,AxisArray(0))
'第四步、设置插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第五步、启动连续插补
result=SMCContiStartList(MyCrd)
'第六步、将直线插补段放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
'第七步、将圆弧插补段放入缓存区
result=SMCArcMoveRadiusUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_1(0),Radius,0,0,Myposi_mode)
'第八步、将直线插补段放入缓存区
result= SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_2(0),Myposi_mode)
'第九步、关闭缓存区
result=SMCContiCloseList(MyCrd)
```

（b）连续插补延时功能

雷赛控制器对连续插补运动提供了延时功能，即在运动过程中实时延时一段时间后继续运动。

⚠注意：1）延时时间为运动停止时的等待时间。

2）当延时时间设置为 0 时，延时时间将无限长。

例　程：执行直线+圆弧连续插补 ，连续插补延时功能。

```
"""""""""""""""""""""变量定义"""""""""""""""""""""
undim *                    '删除之前定义的所有变量及数组
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2),delay_time
Dim Dist(2),MySpara,MyMax_Vel,MyTacc,Radius,Dist_1(2),Dist_2(2),result
MyCrd = 0'参与插补运动的坐标系 0
Myposi_mode = 1'插补运动模式为绝对坐标模式
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyaxisNum = 2'参与插补运动轴数为 2
MyMax_Vel = 300'最大速度为 300unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0.05'平滑时间为 0.05s
Dist(0) = 100'定义直线插补 X 轴运动终点距离
Dist(1) = 50'定义直线插补 Y 轴运动终点距离
delay_time=5'延时时间，单位:S
Dist_1(0) = 200'定义圆弧 X 轴运动终点距离
Dist_1(1) = 50'定义圆弧 Y 轴运动终点距离
Radius=50'定义圆弧半径距离
"""""""""""""""""""""指令调用执行"""""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)     '设置 X 轴脉冲当量为  10pulse/unit
result=SMCSetEquiv(1,10)     '设置 Y 轴脉冲当量为  10pulse/unit
'第二步、打开缓存区
result=SMCContiOpenList(MyCrd,MyaxisNum,AxisArray(0))
'第三步、设置连续插补速度参数
```

```
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第四步、设置平滑 S 时间参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第五步、启动连续插补
result=SMCContiStartList(MyCrd)
'第六步、将直线插补段放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
'第七步、启动延时指令
SMCContiDelay(MyCrd,delay_time,0)
'第八步、将圆弧插补段放入缓存区
result=SMCArcMoveRadiusUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_1(0),Radius,0,0,Myposi_mode)
'第九步、关闭插补缓冲器
result=SMCContiCloseList(MyCrd)
```

运行结果：第一段直线完成后延时 5s，再继续圆弧插补，运行速度结果如下 2.34：

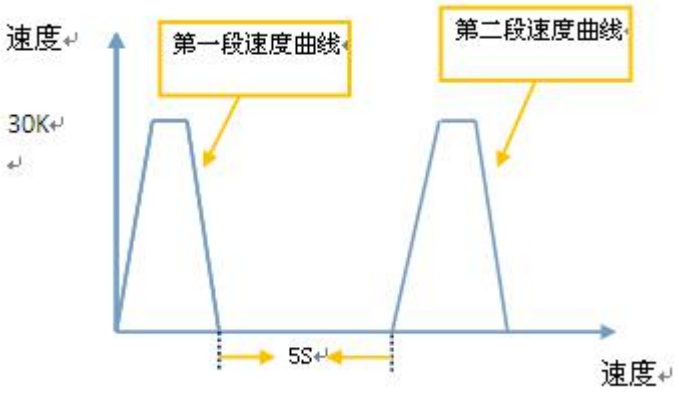


图 2.34 连续插补延时功能示意图

（c）连续插补速度比例调整功能

雷赛控制器对连续插补运动提供了连续插补速度比例调整功能。

例　程：执行直线+圆弧连续插补，连续插补速度变化

```
''''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''''
undim *                          '删除之前定义的所有变量及数组
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2),delay_time
Dim Dist(2),MySpara,MyMax_Vel,MyTacc,Radius,Dist_1(2),Dist_2(2),result
```

```
MyCrd = 0'参与插补运动的坐标系 0
Myposi_mode = 1'插补运动模式为绝对坐标模式
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyaxisNum = 2'参与插补运动轴数为 2
MyMax_Vel = 300'最大速度为 300unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0.05'平滑时间为 0.05s
Dist(0) = 100'定义直线插补 X 轴运动终点距离
Dist(1) = 50'定义直线插补 Y 轴运动终点距离
Ration=1.5'速度倍率变化为 1.5 倍
Dist_1(0) = 200'定义圆弧 X 轴运动终点距离
Dist_1(1) = 50'定义圆弧 Y 轴运动终点距离
Radius=50'定义圆弧半径距离
""""""""""""""""""""指令调用执行""""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)     '设置 X 轴脉冲当量为 10pulse/unit
result=SMCSetEquiv(1,10)     '设置 Y 轴脉冲当量为 10pulse/unit
'第二步、打开缓存区
result=SMCContiOpenList(MyCrd,MyaxisNum,AxisArray(0))    '打开缓冲区
'第三步、设置连续插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第四步、设置平滑 S 时间参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第五步、启动连续插补
result=SMCContiStartList(MyCrd)
'第六步、将直线插补段放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
'第七步、启动连续插补速度比例调整功能
```

```
Delay(500)                          '程序运行延时 200ms
result=SMCContiChangeSpeedRatio(MyCrd,Ration)
'第八步、将圆弧插补段放入缓存区
result=SMCArcMoveRadiusUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_1(0),Radius,0,0,Myposi_mode)
'第九步、关闭插补缓冲器
result=SMCContiCloseList(MyCrd)
```

运行结果：第一段轨迹以设定速度运行，再调用改变速率函数，第二段轨迹将以设定速度的 1.5 倍运行。

（d）连续插补暂停功能

雷赛控制器在连续插补运动过程中支持 IO 控制，通过 IO 控制函数的调用，用户可以轻易实现各种 IO 控制功能。插补暂停指令执行后需再启动 SMCContiStartList 指令才能运动，运动指令是继续原先位置点继续运动。

当暂停、停止连续插补，或遇到其他异常停止（如碰到 EMG 信号）时，运动控制器可以按照用户预先设置的 IO 输出状态进行 IO 控制

例　程：执行直线+圆弧连续插补暂停，IO 输出功能。

```
'''''''''''''''''''''''''变量定义'''''''''''''''''''''''''
undim *                             '删除之前定义的所有变量及数组
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2), action ,Mask,state
Dim Dist(2),MySpara,MyMax_Vel,MyTacc,Radius,Dist_1(2),Dist_2(2),result
MyCrd = 0'参与插补运动的坐标系 0
Myposi_mode = 1'插补运动模式为绝对坐标模式
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyaxisNum = 2'参与插补运动轴数为 2
MyMax_Vel = 300'最大速度为 300unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0.05'平滑时间为 0.05s
Action=1'激活模式 1:恢复运行时不恢复暂停前的 IO 状态
Mask=9'端口值，bit 值对应输出 IO 口，位值为 1 时输出，为 0 不输出
```

```
State=0'输出电平状态 0,低电平
Dist(0) = 100'定义直线插补 X 轴运动终点距离
Dist(1) = 50'定义直线插补 Y 轴运动终点距离
flg=1'标志位
Dist_1(0) = 200'定义圆弧 X 轴运动终点距离
Dist_1(1) = 50'定义圆弧 Y 轴运动终点距离
Radius=50'定义圆弧半径距离
""""""""""""""""""""指令调用执行""""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)          '设置 X 轴脉冲当量为  10pulse/unit
result=SMCSetEquiv(1,10)          '设置 Y 轴脉冲当量为  10pulse/unit
'第二步、打开缓存区
result=SMCContiOpenList(MyCrd,MyaxisNum,AxisArray(0))
'第三步、设置连续插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第四步、设置平滑 S 时间参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第五步、设置暂停 IO 输出，对应输出口 out0、out3 在暂停时输出低电平（LED 亮）
result=SMCContiSetPauseOutput(MyCrd, Action,Mask,State)
'第六步、启动连续插补
result=SMCContiStartList(MyCrd)
'第七步、将直线插补段放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
'第八步、启动插补暂停
Delay(200)                                  '程序运行延时 200ms
result=SMCContiPauseList(MyCrd)
'第九步、判断 IO 输入口 1 电平状态是否为 0，来确定是否再次启动连续插补
While flg
    If  SMCReadInBit(1)=0  then'判断输入口电平状态是否为 0
        If  SMCContiGetRunState(0)=1 then'判断插补状态是否为暂停 1
```

```
                result=SMCContiStartList(MyCrd)      '启动连续插补
              flg=0
            endif
          endif
    wend
    '第十步、将圆弧插补段放入缓存区，启动插补暂停
    result=SMCArcMoveRadiusUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_1(0),Radius,0,0,Myposi_mode)
    '第十一步、关闭插补缓冲器
    result=SMCContiCloseList(MyCrd)
```

运行结果：当暂停运动后 IO 输出口 0 和 3 输出低电平，若 IN1 口为低电平时，插补继续运行，输出口 0 和 3 电平状态保持不变。

（e）连续插补等待 IO 输入功能

当进行连续插补运动时，用户可以在缓冲区插入等待 IO 输入指令。当运动控制器执行到此指令时，其只有在接受到输入 IO 信号或超出超时时间后，才会执行后续运动，超时时间可以自由设置。

例　程：执行直线+圆弧连续插补，连续插补等待 IO 输入。

```
    '''''''''''''''''''''''变量定义'''''''''''''''''''''''
    undim *                '删除之前定义的所有变量及数组
    Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2), action ,Mask,state
    Dim Dist(2),MySpara,MyMax_Vel,MyTacc,Radius,Dist_1(2),Dist_2(2),result
    MyCrd = 0'参与插补运动的坐标系 0
    Myposi_mode = 1'插补运动模式为绝对坐标模式
    AxisArray(0) = 0'定义插补 0 轴为 X 轴
    AxisArray(1) = 1'定义插补 1 轴为 Y 轴
    MyaxisNum = 2'参与插补运动轴数为 2
    MyMax_Vel = 300'最大速度为 300unit/s
    MyTacc = 0.1'加减速时间为 0.1s
    MySmode = 0'保留参数，固定值为 0
    MySpara = 0.05'平滑时间为 0.05s
    Dist(0) = 100'定义直线插补 X 轴运动终点距离
```

```
Dist(1) = 50'定义直线插补 Y 轴运动终点距离
bitno=1'输入口 1
state=0'输入口电平状态为低
TimeOut=2'超时时间，单位：s
Dist_1(0) = 200'定义圆弧 X 轴运动终点距离
Dist_1(1) = 50'定义圆弧 Y 轴运动终点距离
Radius=50'定义圆弧半径距离
""""""""""""""""""""指令调用执行""""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)                    '设置 X 轴脉冲当量为 10pulse/unit
result=SMCSetEquiv(1,10)                    '设置 Y 轴脉冲当量为 10pulse/unit
'第二步、打开缓存区
result=SMCContiOpenList(MyCrd,MyaxisNum,AxisArray(0))
'第三步、设置连续插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第四步、设置平滑 S 时间参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第五步、启动连续插补
result=SMCContiStartList(MyCrd)     '启动连续插补
'第六步、将直线插补段放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
'第七步、启动连续插补等待输入
result=SMCContiWaitInput(MyCrd,bitno,state,TimeOut,0)
'第八步、将圆弧插补段放入缓存区
result=SMCArcMoveRadiusUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_1(0),Radius,0,0,Myposi_mode)
'第九步、关闭插补缓冲器
result=SMCContiCloseList(MyCrd)
```

运行结果：连续插补中，先运行一段直线插补，然后等待通用输入口 1 为低电平时或等待时间超过 2 秒后，才执行后续运动。

（f）连续插补超前滞后输出 IO 功能

当进行连续插补运动时，用户可以在缓冲区插入普通 IO 输出控制指令。在每段运动轨迹中，至多可添加 10 个 IO 操作，包括普通 IO 控制及精确位置 CMP 输出控制。普通 IO 输出的分辨率为 1ms（误差在 0.2ms 以内）。

其中，SMCContiDelayOutbitToStart 指令可以设置轨迹段内 IO 的输出位置或时间，该位置或时间是相对于该轨迹段起点的滞后值。

其中，SMCContiDelayOutbitToStop 指令可以设置轨迹段执行完后 IO 的输出时间，该时间是相对于该轨迹段终点的滞后值。

其中，SMCContiAheadOutbitToStop 指令可以设置轨迹段内 IO 的输出位置或时间，该位置或时间是相对于该轨迹段终点的提前值。

其中，SMCContiWriteOutbit 指令可以实现连续插补运动中的 IO 立即输出。

其中，SMCContiClearIoAction 指令可以实现当本段轨迹运行完成时，清除仍未执行完的 IO 操作，使其不会在后续轨迹段中被继续执行。

⚠注意：超前和滞后的输出 IO 时间、电平翻转时间大小值都需与实际插补段运行时间相匹配。如果设定超前 IO 输出时间为 1S，而本段轨迹完成时间只需 500ms 那么可能指令会报错，可以用指令 SMCContiClearIoAction 清除段内的 IO 操作或者需要设定超前输出 IO 时间在 500ms 内，即轨迹未完成前。

例程 1：执行直线+圆弧连续插补，连续插补中以时间输出 IO 控制。

```
"""""""""""""""""""变量定义"""""""""""""""""""
undim *                    '删除之前定义的所有变量及数组
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2), action ,Mask,state
Dim Dist(2), result,MyMax_Vel,MyTacc,Radius,Dist_1(2),Dist_2(2),on_off
MyCrd = 0'参与插补运动的坐标系 0
Myposi_mode = 1'插补运动模式为绝对坐标模式
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyaxisNum = 2'参与插补运动轴数为 2
MyMax_Vel = 300'最大速度为 300unit/s
MyTacc = 0.1'加减速时间为 0.1s
```

```
MyBitno = 1'通用输出口 1
MyLevel = 0'输出电平为低电平
MyDelayMode   = 0'滞后模式为滞后时间
MyDelayVal = 0.4'滞后时间为 0.4s
MyRevTime = 0.5'电平延时翻转时间为 0.5s
Dist(0) = 100'定义直线插补 X 轴运动终点距离
Dist(1) = 50'定义直线插补 Y 轴运动终点距离
Dist_1(0) = 200'定义圆弧 X 轴运动终点距离
Dist_1(1) = 50'定义圆弧 Y 轴运动终点距离
Radius=50'定义圆弧半径距离
bitno=1'输出口 1
on_off=0'输出低电平 0
"""""""""""""""""""""指令调用执行"""""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,10)          '设置 X 轴脉冲当量为  10pulse/unit
result=SMCSetEquiv(1,10)          '设置 Y 轴脉冲当量为  10pulse/unit
'第二步、打开缓存区
result=SMCContiOpenList(MyCrd,MyaxisNum,AxisArray(0))
'第三步、设置连续插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第四步、设置平滑 S 时间参数
result=SMCSetVectorSprofile(MyCrd,0,0.05)
'第五步、相对于轨迹段起点,滞后 0.4 秒，输出口 1 输出低电平，持续时间为 0.5s
result=SMCContiDelayOutbitToStart(MyCrd,MyBitno,MyLevel,MyDelayVal,
MyDelayMode,MyRevTime)
'第六步、启动连续插补
result=SMCContiStartList(MyCrd)
'第七步、将直线插补段放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
'第八步、将圆弧插补段放入缓存区
```

```
result=SMCArcMoveRadiusUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_1(0),Radius,0,0,Myposi_mode)
'第九步、连续插补中输出 IO 翻转
result=SMCContiWriteOutbit(MyCrd,bitno,on_off,MyRevTime)
'第十步、关闭插补缓冲器
result=SMCContiCloseList(MyCrd)
```

运行结果如下：

相对于第一段轨迹段起点,滞后 0.4 秒，输出口 1 输出低电平，低电平持续时间为 0.5s。第 2 段轨迹完成后，输出口 1 输出低电平，持续时间 0.5s

例程 2：执行直线+圆弧连续插补，连续插补中以位置输出 IO 控制。

```
''''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''''
undim *                         '删除之前定义的所有变量及数组
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2), action ,Mask,state
Dim Dist(2), result,MyMax_Vel,MyTacc,Radius,Dist_1(2),Dist_2(2),on_off
MyCrd = 0'参与插补运动的坐标系 0
Myposi_mode = 1'插补运动模式为绝对坐标模式
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyaxisNum = 2'参与插补运动轴数为 2
MyMax_Vel = 300'最大速度为 300unit/s
MyTacc = 0.1'加减速时间为 0.1s
MyBitno = 1'通用输出口 1
MyLevel = 0'输出电平为低电平
MyDelayMode   =1'滞后模式为滞后位置
MyDelayVal = 10'滞后位置为 10 unit
MyRevTime = 0.5'电平延时翻转时间为 0.5s
Dist(0) = 100'定义直线插补 X 轴运动终点距离
Dist(1) = 50'定义直线插补 Y 轴运动终点距离
Dist_1(0) = 200'定义圆弧 X 轴运动终点距离
Dist_1(1) = 50'定义圆弧 Y 轴运动终点距离
Radius=50'定义圆弧半径距离
```

```
bitno=1'输出口 1
on_off=0'输出低电平 0
"""""""""""""""""""""指令调用执行"""""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,100)          '设置 X 轴脉冲当量为 100pulse/unit
result=SMCSetEquiv(1,100)          '设置 Y 轴脉冲当量为 100pulse/unit
'第二步、打开缓存区
result=SMCContiOpenList(MyCrd,MyaxisNum,AxisArray(0))
'第三步、设置连续插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第四步、设置平滑 S 时间参数
result=SMCSetVectorSprofile(MyCrd,0,0.05)
'第五步、相对于轨迹段起点,滞后 10unit，输出口 1 输出低电平，持续时间为 0.5s
result=SMCContiDelayOutbitToStart(MyCrd,MyBitno,MyLevel,MyDelayVal,MyDelayMode,MyRevTime)
'第六步、启动连续插补
result=SMCContiStartList(MyCrd)
'第七步、将直线插补段放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
'第八步、将圆弧插补段放入缓存区
result=SMCArcMoveRadiusUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_1(0),Radius,0,0,Myposi_mode)
'第九步、连续插补中输出 IO 翻转
result=SMCContiWriteOutbit(MyCrd,bitno,on_off,MyRevTime)
'第十步、关闭插补缓冲器
result=SMCContiCloseList(MyCrd)
```

运行结果如下：

相对于第一段轨迹段起点，运行距离 10unit 后，输出口 1 输出低电平，低电平持续时间为 0.5s。第 2 段轨迹完成后，输出口 1 输出低电平，持续时间 0.5s。

例程 3：执行直线+圆弧连续插补，清除段内未执行完的 IO 动作功能。

```
"""""""""""""""""""""变量定义"""""""""""""""""""""
undim *          '删除之前定义的所有变量及数组
```

```
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2),action ,Mask,state
Dim Dist(2),MySpara,MyMax_Vel,MyTacc,Radius,Dist_1(2), IoMask,on_off
MyCrd = 0'参与插补运动的坐标系 0
Myposi_mode = 1'插补运动模式为绝对坐标模式
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyaxisNum = 2'参与插补运动轴数为 2
MyMax_Vel = 300'最大速度为 300unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0.05'平滑时间为 0.05s
MyBitno = 1'通用输出口 1
MyLevel = 0'输出电平为低电平
MyDelayMode   = 0'滞后模式为滞后时间
MyDelayVal = 0.1'滞后时间为 0.1s
MyRevTime = 3'电平延时翻转时间为 3s
Dist(0) = 100'定义直线插补 X 轴运动终点距离
Dist(1) = 50'定义直线插补 Y 轴运动终点距离
IoMask=2'bit 值，清除输出口 1 的 IO 动作
Dist_1(0) = 200'定义圆弧 X 轴运动终点距离
 Dist_1(1) = 50'定义圆弧 Y 轴运动终点距离
 Radius=50'定义圆弧半径距离
''''''''''''''''''''''''''指令调用执行''''''''''''''''''''''''''
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,100)        '设置 X 轴脉冲当量为 100pulse/unit
result=SMCSetEquiv(1,100)        '设置 Y 轴脉冲当量为 100pulse/unit
'第二步、打开缓存区
result=SMCContiOpenList(MyCrd,MyaxisNum,AxisArray(0))
'第三步、设置连续插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
```

```
'第四步、设置平滑 S 时间参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第五步、启动连续插补
result=SMCContiStartList(MyCrd)
'第六步、相对于轨迹段起点,滞后 0.1 秒，输出口 1 输出低电平
result=SMCContiDelayOutbitToStart(MyCrd,MyBitno,MyLevel,MyDelayVal,
MyDelayMode,MyRevTime)
'第七步、清除段内未执行的通用输出口动作
result=SMCContiClearIoAction(MyCrd,IoMask)
'第八步、将直线插补段放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
'第九步、将圆弧插补段放入缓存区
result=SMCArcMoveRadiusUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_1(0),Radius,0,0,Myposi_mode)
'第十步、关闭插补缓冲器
result=SMCContiCloseList(MyCrd)
```

运行结果：

当第一段直线插补开始 0.1s 后，通用输出口 1 输出低电平，低电平持续时间为 3s。但在运行 3s 后，已进入了第二段圆弧插补运动。此时，由于使用了函数 SMCContiClearIoAction 清除段内未执行完的 IO 操作，所以通用输出口 1 的电平将持续保持低电平，不会翻转。如图 2.35 所示。

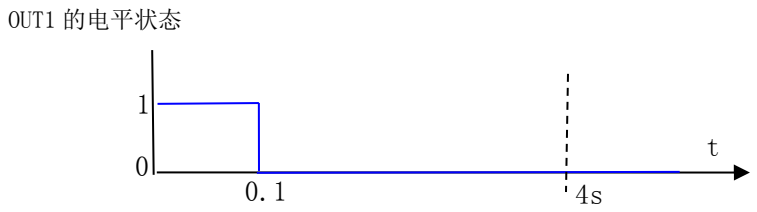


图 2.35 通用输出口 1 的电平时序图(清除段内未执行完的 IO 操作)

如果在上例中没有插入指令 SMCContiClearIoAction 清除段内未执行完的 IO 操作，那么通用输出口 1 的电平将会在到达设置的延时翻转时间后翻转，尽管此时已进入了第二段圆弧插补运动。如图 2.36 所示。如果翻转时间在直线轨迹段内可实现，那么清除指令也是无效的。

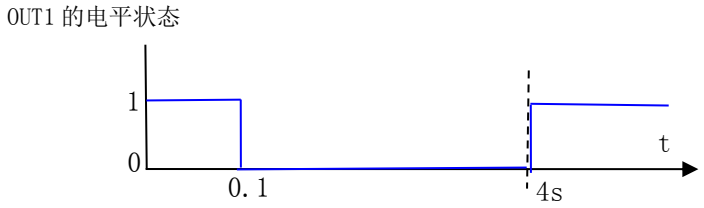


图 2.36 通用输出口 1 的电平时序图(未执行清除段内未执行完的 IO 操作)

（g）连续插补速度参数可设

雷赛控制器对连续插补运动提供了对下段轨迹加速度、加减速时间、最大运行速度、起始停止速度可设功能，客户应用范围更广、灵活实用。

⚠注意：设定其运行速度、加速度、加减速时间后将在下段轨迹开始生效。参数设置必须是成双出现的，如设置了加速时间则必须设置减速时间。一般不设置中间轨迹段的起始、停止速度，否则在轨迹连接处可能加速度有较大的跳变，震荡加剧。可以配合 Blend 功能来使用，改变各段的运行速度，来提高效率等。

例　程：执行连续插补，各段速度加速度、运行速度可设

```
''''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''''
undim *          '删除之前定义的所有变量及数组
Dim MyCrd,MyaxisNum,MySmode,Myposi_mode ,AxisArray(2),delay_time
Dim Dist(2),MySpara,MyMax_Vel,MyTacc,Radius,Dist_1(2),Dist_2(2),result
MyCrd = 0'参与插补运动的坐标系 0
Myposi_mode = 1'插补运动模式为绝对坐标模式
AxisArray(0) = 0'定义插补 0 轴为 X 轴
AxisArray(1) = 1'定义插补 1 轴为 Y 轴
MyaxisNum = 2'参与插补运动轴数为 2
MyMax_Vel = 300'最大速度为 300unit/s
MyTacc = 0.1'加减速时间为 0.1s
MySmode = 0'保留参数，固定值为 0
MySpara = 0.05'平滑时间为 0.05s
Dist(0) = 150'定义直线插补 X 轴运动终点距离
Dist(1) = 150'定义直线插补 Y 轴运动终点距离
Dist_1(0) = 250'定义直线 1 插补 X 轴运动终点距离
Dist_1(1) = 250'定义直线 1 插补 Y 轴运动终点距离
```

```
Dist_2(0) = 500'定义直线 2 插补 X 轴运动终点距离
Dist_2(1) = 120'定义直线 2 插补 Y 轴运动终点距离
""""""""""""""""""""指令调用执行""""""""""""""""""""
'第一步、设置轴脉冲当量值
result=SMCSetEquiv(0,100) '设置 X 轴脉冲当量为 100pulse/unit
result=SMCSetEquiv(1,100) '设置 Y 轴脉冲当量为 100pulse/unit
'第二步、打开缓冲区
result=SMCContiOpenList(MyCrd,MyaxisNum,AxisArray(0))
'第三步、设置插补速度参数值
result=SMCSetVectorProfileUnit(MyCrd,0,MyMax_Vel,MyTacc,MyTacc,0)
'第四步、设置插补平滑 S 参数
result=SMCSetVectorSprofile(MyCrd,MySmode,MySpara)
'第五步、使能 Blend 功能
result=SMCContiSetBlend(0,1)
'第六步、启动连续插补
result=SMCContiStartList(MyCrd)
'第七步、将直线插补段放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist(0),Myposi_mode)
'第八步、改变连续插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,100,0.2,0.1,0)
'第九步、将直线 1 插补段放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_1(0),Myposi_mode)
'第十步、再次改变连续插补速度参数
result=SMCSetVectorProfileUnit(MyCrd,0,200,0.1,0.2,0)
'第十一步、将直线 2 插补段放入缓存区
result=SMCLineUnit(MyCrd,MyaxisNum,AxisArray(0),Dist_2(0),Myposi_mode)
'第十二步、关闭插补缓冲器
result=SMCContiCloseList(MyCrd)
```

运行结果：

第一段轨迹运行速度 150、加速时间 0.1s、减速时间 0.1s，第二段轨迹运行速度 100、加速时间 0.2s、加速时间 0.1s，第三段轨迹运行速度 200、加速时间 0.1s、减速时间 0.2s。

2.2.5 手轮运动

雷赛控制器提供了一个强大的手轮脉冲控制功能，虽然接口只有一个，但是通过灵活配置，就能达到用户理想的手轮效果。控制器手轮功能提供了两种模式，一种默认模式，一种高级模式。

在默认模式下，手轮功能与手轮上的图标说明一一对应，例如 x，y，z，4，5，6 轴的单轴控制，3 个速率档位可选：1，10，100 倍的控制。

在高级模式下，轴档位不在是固定的单轴控制，而是可以通过函数来设定每一个轴档位关联的轴组，速率档位也不再是固定的 1，10，100 倍。举例说明：如果有三种情况需要用到手轮，1：x, y 轴同时运动，2：z, u 轴同时运动，3: 4 轴同时运动。 那么就可以设定为 x 轴手轮信号关联 x,y 轴，同时可以单独设置 3 个档位的速率，不与其他轴手轮档位(y, z, u)的共用。y 轴手轮信号关联 z,u 轴，速率也可单独设置。z 轴则关联 x,y,z,u 4 个轴，速率也单独设置。这样，当手轮轴选打到 x 轴开关时，可以同时控制 x,y 轴运动， 打到 z 轴开关时，则可以控制 4 个轴同时运动。

不同的模式可以满足不同的使用情况，使得手轮功能大大增强，满足使用者的多种需求。

相关指令：

名称	功能	参考
SMCHandwheelSetAxislist	设置同一轴选档位下具体运动轴列表	4.18 节
SMCHandwheelGetAxislist	读取同一轴选档位下具体运动轴列表	
SMCHandwheelSetRatiolist	设置同一轴选下手轮倍率档位	
SMCHandwheelGetRatiolist	读取同一轴选下手轮倍率档位	
SMCHandwheelSetMode	设置手轮运动模式，硬件或者软件模式下运动	
SMCHandwheelGetMode	读取手轮运动模式，硬件或者软件模式下运动	
SMCHandwheelSetIndex	选择或更换手轮运动轴选、倍率档位	
SMCHandwheelGetIndex	读取选择手轮运动轴选、倍率档位	

SMCHandwheelMove	启动手轮运动	
SMCHandwheelStop	停止手轮运动	

例　程：启动手轮运动

```
'''''''''''''''''''''变量定义'''''''''''''''''''''
undim *        '清除数组，变量等
dim AxisSelIndex,RatioSelIndex,result
dim AxisSelIndex1,RatioSelIndex1
AxisSelIndex = 0'手轮轴选档位为 0
RatioSelIndex = 0'手轮倍选档位为 0,1 倍速度
AxisSelIndex1 = 1'手轮轴选档位为 1
RatioSelIndex1= 1'手轮倍选档位为 1,10 倍速度
dim InMode,IfHardEnable
InMode = 0'输入脉冲模式，0：脉冲+方向，1：AB 相脉冲
IfHardEnable = 0'运行模式，0：软件控制，1：硬件控制
Dim AxisNum,AxisList(3), AxisNum1,AxisList1(2)
AxisNum = 3'手轮档位 0 运动轴为 3 轴
AxisList(0) = 0'手轮档位 0 运动轴列表
AxisList(1) = 1
AxisList(2) = 2
AxisNum1 = 2'手轮档位 1 运动轴为 2 轴
AxisList1(0) = 4'手轮档位 1 运动轴列表
AxisList1(1) = 5
dim StartRatioIndex,RatioSelNum,RatioList(3)
StartRatioIndex = 0'倍选起始值
RatioSelNum = 3'需设定倍率值 3
RatioList(0) = 1'设定倍率列表
RatioList(1) = 10
RatioList(2) = 100
''''''''''''''''''''指令调用执行''''''''''''''''''''
```

```
'第一步、设置手轮运动模式
result=SMCHandwheelSetMode(InMode,IfHardEnable)
'第二步、设置手轮轴选倍率档位
result=SMCHandwheelSetIndex(AxisSelIndex,RatioSelIndex)
'第三步、设置手轮档位及运动轴列表
result=SMCHandwheelSetAxislist(AxisSelIndex,AxisNum,AxisList(0))
result=SMCHandwheelSetAxislist(AxisSelIndex1,AxisNum1,AxisList1(0))
'第四步、设置手轮倍率档位及对应轴的倍率值
result=SMCHandwheelSetRatiolist(AxisSelIndex,StartRatioIndex,RatioSelNum,RatioList(0))
result=SMCHandwheelSetRatiolist(AxisSelIndex1,StartRatioIndex,RatioSelNum,RatioList(0))
'第五步、启动手轮运动
result=SMCHandwheelMove(0)
'第六步、5S 后换到手轮档位 1
delay(5000)
result=SMCHandwheelSetIndex(AxisSelIndex1,RatioSelIndex1)
```

运行结果：

运行程序，外部有脉冲输入时，轴选和倍选档位都为 0，轴 0、1、2 分别运动，速度倍率为 1 倍。5s 后换轴选档位 1，轴 4、5 运动，速度倍率为 10 倍。

2.2.6 电子凸轮

雷赛控制器支持单轴或者多轴按凸轮关系表跟随主轴的指令位置或编码器反馈位置运动。

电子凸轮数据表，采用相对位置模式，Mpos1 和 Spos1 固定为 0，主轴位置必须朝一个方向递增或者递减。

主轴位置	Mpos1	Mpos2	.....	Mposn-1	MposN
从轴位置	Spos1	Spos2	.....	Sposn-1	SposN

电子凸轮控制流程

1、下载凸轮数据表

2、启动从轴凸轮运动，从轴进入凸轮运动模式并为运动状态，当前点即为跟随起始点，最大跟随距离为凸轮表中主轴最大距离，主轴停止状态下方可启动从轴凸轮运动。

3、启动主轴运动，从轴开始跟随，超出跟随范围后从轴不在跟随。

4、停止从轴凸轮运动，从轴退出凸轮跟随模式。

相关指令：

名称	功能	参考
SMCCamTableUnit	设置凸轮表	4.18 节
SMCCamMove	启动电子凸轮运动	

例　程：

```
dim result,MasterAxisNo,SlaveAxisNo,Count,MasterPos(100),SlavePos(100)
dim SrcMode,MasterTargetPos
MasterAxisNo = 0'主轴轴号
SlaveAxisNo = 1'从轴轴号
SrcMode = 0'主轴位置模式：0-指令位置，1-反馈位置
Count = 11'数据个数
'填充电子凸轮数据表，最大 1000 组，第一组数据保留，必须固定为（0，0）
For  i=0  to Count-1
    MasterPos(i) = i*-1000
    SlavePos(i) = -100*(i mod2)
next
'添加电子凸轮表
result=SMCCamTableUnit(MasterAxisNo,SlaveAxisNo,Count,MasterPos(0),SlavePos(0),SrcMode)
'启动从轴电子凸轮运动
result=SMCCamMove(SlaveAxisNo)
'控制主轴运动
result=SMCPMoveUnit(MasterAxisNo,MasterPos(Count-1),0)
```

2.3 通用 IO 功能

用户可以使用雷赛控制器上的数字 I/O 口用于检测开关信号、传感器信号等输入信号，或者控制继电器、电磁阀等输出设备的信号。

2.3.1 通用输入 IO

通用输入 IO 口常用于检测开关信号、传感器信号等输入信号。

相关指令：

名称	功能	参考
SMCReadInbit	读取某个输入端口的电平	4.16 节
SMCReadInport	读取全部输入端口的电平	

例程 1：读取控制器的通用输入口 1 的电平值

```
Dim MyInbitno,MyInValue
MyInbitno = 1'定义通用输入口 1
MyInValue = SMCReadInBit(MyInbitno)      '读取当前输入口 1 的电平状态
print MyInValue                          '打印状态值
```

例程 2：读取控制器的所有输入口电平值

```
Dim MyInbitno,MyInValue
MyInbitno = 0'固定口 0,输入 IO 口 32 位内
MyInValue = SMCReadInPort(MyInbitno)        '读取所有输入 IO 口的电平状态
print MyInValue     '打印结果转成 2 进制后，位 0 表示低电平、位 1 高电平
```

⚠注意：在使用 SMCReadInport 进行 IO 电平读显示时，可将返回值转成 2 进制后，位 0 表示低电平、导通状态，位 1 表示高电平、断开状态。2 进制 bit 值从低到高对应 IO 口数。

例程 3：读取控制器 IO 大于 32 位的电平状态，如读取 SMC604A 中 IN32-45 的 IO 电平

```
Dim MyInbitno,MyInValue
MyInbitno = 1'固定口 1，输入 IO 口大于 32 位
MyInValue = SMCReadInPort(MyInbitno)        '读取所有输入 IO 口的电平状态
print MyInValue         '打印结果转成 2 进制后，位 0 低电平、位 1 表示高电平
```

⚠注意：读取小于 32 位的 IO 输入输出所有状态值时，我们使用 port 口 0，读取大于 32 位的 IO 输入输出所有状态值时，我们使用 port 口 1

2.3.2 通用输出 IO

通用输出 IO 口常用于控制继电器、电磁阀等输出设备的信号。

相关指令：

名称	功能	参考
SMCWriteOutbit	设置某个输出端口的电平	4.16 节
SMCReadOutbit	读取某个输出端口的电平	
SMCWriteOutport	设置全部输出口的电平	
SMCReadOutport	读取全部输出口的电平	
SMCReverseOutbit	IO 输出延时翻转	
SMCSetIoCountMode	设置 IO 计数模式	
SMCGetIoCountMode	读取 IO 计数模式设置值	
SMCSetIoCountValue	设置 IO 计数值	
SMCGetIoCountValue	读取 IO 计数值	

例程 1：设置输出口 1 的电平为低电平，再读取控制器的通用输出口 1 的电平值

```
Dim MyInbitno,MyInValue
MyInbitno = 1'定义通用输出口 1
SMCWriteOutBit(MyInbitno,0)                    '设置通用输出口 1 电平为低电平 0
MyInValue = SMCReadOutBit(MyInbitno)       '读取当前 IO 1 口的电平状态
print MyInValue                                  '打印结果为 0，表示当前输出口状态为 0
```

例程 2：设置输出口 0、1、2、3 的电平为低电平，其它通用输出口的电平值为高电平

```
Dim portno,MyInValue
portno = 0'固定值 0
SMCWriteOutPort (portno,262128)          '设置通用所有输出口电平参数
MyInValue = SMCReadOutPort (portno)      '读取当前所有输出口口的电平状态
print MyInValue                                  '打印结果转成 2 进制后为
'11111111111110000，bit0 表示低电平，bit1 表示高电平，从低位开始
```

⚠注意：在使用 SMCWriteOutport、SMCReadOutport 对运动控制器的全部输出口进行置位，使用的是十进制数进行赋值，0 表示低电平、1 表示高电平。使用方法类似 IN 口。

例程 3：设置输出口 0 翻转时间 1s

```
Dim MyInbitno,reverse_time,MyInValue
MyInbitno = 0'定义通用输出口 0
```

```
reverse_time=1'翻转时间 1s，单位 s
SMCReverseOutbit(MyInbitno,reverse_time)    '设置通用输出口 0，翻转延时 1s
```

例程 4：设置输输入口 IO 计数

```
Dim bitno,mode,count
bitno=1'计数输入口 1
mode=1'IO 计数模式，0：禁用，1：上升沿计数，2：下降沿计数
SMCSetIoCountMode(bitno,mode,0)    '设置计数模式
SMCSetIoCountValue(bitno,0)        'IO 计数值置 0
While 1
    SMCGetIoCountValue(bitno,count)      '读取 IO 计数值
    print count                          '打印 IO 计数值
    delay(200)
wend
```

2.3.3 虚拟 IO 映射

雷赛控制器提供了虚拟 IO 映射功能，该功能允许用户对虚拟 IO 口的硬件输入接口进行任意配置。通过该功能函数的设置可以专用通用 IO 输入接口的滤波功能。

相关指令：

名称	功能	参考
SMCSetIoMapVirtual	设置虚拟 IO 轴映射参数	4.27 节
SMCGetIoMapVirtual	读取虚拟 IO 轴映射参数	
SMCReadInbitVirtual	读取虚拟 IO 电平状态	

⚠注意：SMCReadInbitVirtual 与 SmcReadInbit 函数的区别是：SmcReadInbit 函数是不经过滤波直接读取硬件端口的电平状态，SMCReadInbitVirtual 函数通过虚拟 IO 映射后读取相应端口滤波后的电平状态

例　程：设置第 2 轴原点信号输入口作为虚拟 IO 口 3 输入信号。

```
Dim result,bitno, MapIoType, MapIoIndex ,MyVirIOLogic,filter_time
bitno = 3'虚拟 IO 口号：IO 口 3
MapIoType = 2'轴 IO 映射类型：原点信号口
```

```
MapIoIndex = 2'轴 IO 映射索引号：第 2 轴
filter_time = 0.05'0.05 秒滤波
result=SMCSetIoMapVirtual(bitno,MapIoType,MapIoIndex,filter_time)
'虚拟 IO 口 3，作为第 2 轴原点输入接口。
result=SMCGetIoMapVirtual(bitno,MapIoType,MapIoIndex,filter_time)'读设置参数值
print MapIoType,,MapIoIndex,filter_time    '打印结果 2,2,0.05
printSMCReadInbitVirtual(bitno)                '读取虚拟 IO 口 3 电平状态
```

运行结果：运行程序，当原点信号发生变化后，我们可读取虚拟 IO 口 3 滤波后的电平状态。

2.4 特殊 IO 功能

2.4.1 编码器检测

雷赛控制器每轴都有一个编码器输入接口用于检测平台的位移或电机的转角。编码器有 EA、EB、EZ 三个信号，脉冲计数信号由 EA 和 EB 端口输入；它可以接收两种类型的脉冲信号：正负脉冲输入或 A/B 相正交信号；EZ 信号是编码器零位信号。编码器外形如图 2.11 所示。

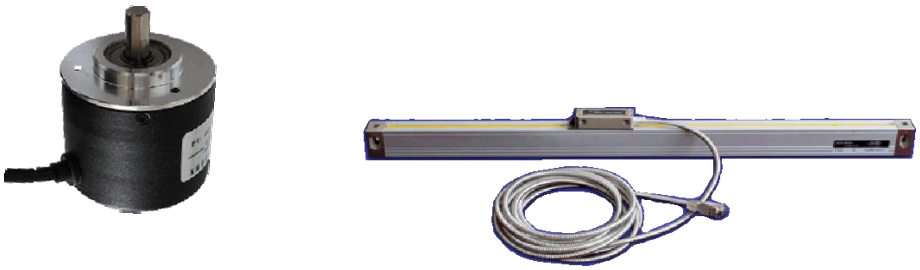


图2.11　编码器外形

采用探针和编码器配合使用，雷赛控制器通过位置触发功能，可完成对工件的位置检测工作，如图 2.12 所示。即：当探针接触到工件时，产生一个触发信号；雷赛控制器接收该信号后，立即将编码器当前位置记录下来；通过记录工件的一系列数据，然后再通过软件处理，即可获得该工件的外形尺寸。

图2.12 被测工件与探针

相关指令：

名称	功能	参考
SMCSetCounterInmode	设置编码器信号参数	4.19 节
SMCGetCounterInmode	读取编码器信号参数	
SMCSetEncoderUnit	设置当前编码器计数值	
SMCGetEncoderUnit	读取当前编码器计数值	

⚠注意：编码器设置模式切换后，编码器值将自动清零。

例　程：显示0号轴编码器的位移量。（编码器是一个最小分辨率为0.005mm的光栅尺）

```
Dim MyCardNo,MyAxis, mode, mode_r,dis2 ,c
Dim MyHomelatchValue , result
MyAxis = 0'轴号
mode = 3'编码器参数模式为 4 倍频
result=SMCSetCounterInmode (MyAxis , mode )   '设置编码器参数
result=SMCGetCounterInmode(MyAxis , mode_r)   '读取编码器参数
print   mode_r                                  '打印编码器模式参数
While 1
    result=SMCGetEncoderUnit(MyAxis, dis2)          '读取 0 号轴编码计数值
    c= dis2*0.005'读取 0 号轴编码器的值，并转换为 mm
    print c,"mm"'打印编码器的值
   delay(500)
wend
```

运行结果：运行程序前需确认 0 号轴有 AB 相脉冲输入，编码计数值将会打印编码计数返回值。

2.4.2 位置锁存

2.4.2.1 高速位置锁存

雷赛控制器支持多种高速位置锁存功能，可实现精确定位功能，其包括单次锁存、连续锁存。单次锁存在锁存信号有效后可锁存一次，再次锁存需复位锁存标志。

连续锁存可实现对多个位置依次进行锁存。在每次锁存后，不需要再复位锁存标志。锁存次数超过 1000 次，剔除最先的触发位置保存最新的触发位置。连续锁存间隔时间需大于 1ms。

相关指令：

名称	功能	参考
SMCSetLtcMode	设置指定轴的 LTC 信号参数	4.20 节
SMCGetLtcMode	读取指定轴的 LTC 信号参数	
SMCSetLatchMode	设置锁存方式参数	
SMCGetLatchMode	读取锁存方式参数	
SMCGetLatchValueUnit	从控制器内读取锁存器的值	
SMCGetLatchFlag	从控制器内读取指定轴的锁存次数	
SMCResetLatchFlag	复位锁存器的标志位	

例　程：高速单次位置锁存

```
''''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''''
Dim Myaxis,i, result
Dim My_latch_Value '定义锁存值
Dim My_latch_flag   '定义锁存个数
Myaxis = 0'轴号
i=1'循环
''''''''''''''''''''''''''''指令调用执行''''''''''''''''''''''''''''
'第一步、设置 0 号轴的 LTC 信号，下降沿触发
result=SMCSetLtcMode(Myaxis,0,0,0)
'第二步、设置 0 轴的锁存源是指令位置，单次锁存
result=SMCSetLatchMode(Myaxis,0,0,0)
'第三步、复位 0 号轴的锁存状态
```

```
result=SMCResetLatchFlag(Myaxis)
'第四步、置 0 号轴速度参数
result=SMCSetProfileUnit(Myaxis,0,5000,0.1,0.1,0)
'第五步、启动运动
result=SMCPMoveUnit(Myaxis,50000,0)
'第六步、判断锁存器是否已锁存，若锁存有效，则打印锁存值
While (i)
      If SMCGetLatchFlag(Myaxis)>0 then'
          i=0'跳出循环
          My_latch_Value= SMCGetLatchValueUnit(Myaxis)      '从 PC 缓存中读取锁存器已锁
'存值，并赋值给变量 My_latch_Value
          Print My_latch_Value                  '打印锁存值
      Endif
Wend
```

运行结果：当轴 0 锁存信号 IN20 有效后，将会锁存，锁存值在变量 My_latch_Value 里。

例 2：高速连续位置锁存

```
"""""""""""""""""""变量定义"""""""""""""""""""
undim *                                  '删除变量，数组
Dim Myaxis,i, ltc_mode, result
Dim My_latch_Value(20)                  '定义锁存次数 20 次
Dim My_latch_flag                        '定义锁存个数
Myaxis = 0'轴号
ltc_mode=2'锁存模式，0 单次锁存，2 连续锁存
"""""""""""""""""""指令调用执行"""""""""""""""""""
'第一步、设置 0 号轴的 LTC 信号，下降沿触发
result=SMCSetLtcMode(Myaxis,0,0,0)
'第二步、设置 0 号轴的锁存参数
   result=SMCSetLatchMode(Myaxis, ltc_mode,0,0)
'第三步、复位 0 号轴的锁存状态
result=SMCResetLatchFlag(Myaxis)
```

```
'第四步、置 0 号轴速度参数
result=SMCSetProfileUnit(Myaxis,0,5000,0.1,0.1,0)
'第五步、启动运动
result=SMCPMoveUnit(Myaxis,50000,0)
'第六步、判断 0 号轴状态，等待 0 轴运动停止
While(SMCCheckDone(Myaxis) = 0) '
Wend
'第七步、从缓存中读取锁存器已锁存个数，并打印锁存值
My_latch_flag= SMCGetLatchFlag (Myaxis)    '
For    i = 0 To My_latch_flag - 1
        My_latch_Value(i) = SMCGetLatchValueUnit(Myaxis)
'按索引号读取缓冲区中已保存的锁存值，并赋值给变量 My_latch_Value
        print My_latch_Value(i)                    '打印锁存值
Next i
```

运行结果：当轴 0 锁存信号有效后，将会自动锁存，所有锁存值为在变量 My_latch_Value（i）里。锁存信号下降沿有效，则可锁存一次数值，可锁存多个数值。

2.4.2.2 原点位置锁存

雷赛控制器提供了原点锁存功能，该功能可以实现在碰到原点信号时将当前位置锁存，使用该功能可以实现精确回原点运动，一般步骤如下：

1）使用 SMCSetHomelatchMode 函数设置原点锁存模式；

2）使用 SMCResetHomelatchFlag 函数清除原点锁存标志；

3）设置轴运动参数，并启动运动；

4）判断原点锁存标志是否有效，有效则停止运动；

5）重新设置轴运动速度为低速度，并使用定长运动到原点锁存位置；

6）回到原点锁存位置后，指令脉冲计数器清零。

相关指令：

名称	功能	参考
SMCResetHomelatchFlag	清除原点锁存标志	4.21 节

SMCGetHomelatchFlag	读取原点锁存标志	
SMCGetHomelatchValueUnit	读取原点锁存值	
SMCSetHomelatchMode	设置原点锁存模式	
SMCGetHomelatchMode	读取原点锁存模式	

例　程：使用原点锁存功能进行精确回原点运动

```
"""""""""""""""""""变量定义"""""""""""""""""""
Dim MyCardNo,MyAxis,Myenable,Mylogic,Mysource , result
DimLatchValue                    '锁存值
MyAxis = 0'轴号
Myenable = 1'使能原点锁存
Mylogic = 0'锁存方式为下降沿锁存
Mysource = 0'锁存位置源为指令位置
"""""""""""""""""""指令调用执行"""""""""""""""""""
'第一步、设置设置原点锁存参数
result=SMCSetHomelatchMode(MyAxis,Myenable,Mylogic,Mysource)
'第二步、清除原点锁存标志位
result=SMCResetHomelatchFlag(MyAxis)
'第三步、设置 0 号轴速度等参数
result=SMCSetProfileUnit(Myaxis,0,1000,0.1,0.1,0)
'第四步、启动 0 号轴执行恒速运动，负方向
result=SMCVmove(MyAxis,0)
'第五步、判断原点锁存标志位是否有效
While(SMCGetHomelatchFlag(Myaxis) = 0)
Wend
'第六步、原点锁存标志位有效后，运动停止
result= SMCSetDecStopTime(Myaxis,0.15)   '设定减速停止速度时间
result=SMCStop(MyAxis,0)
'第七步、判断当前运动是否停止
While (SMCCheckDone(MyAxis) = 0)
```

```
Wend
'第八步、设置 0 号轴速度等参数
result=SMCSetProfileUnit(Myaxis,0,1000,0.1,0.1,0)
'第九步、获取原点锁存值，并赋值给变量 LatchValue
LatchValue = SMCGetHomelatchValueUnit(Myaxis)
'第十步、0 轴定长运动到原点锁存位置点
result=SMCPmoveUnit(MyAxis, LatchValue,1)
'第十一步、判断当前运动状态，等待停止
While(SMCCheckDone(MyAxis) = 0)
Wend
'第十二步、设置 0 轴的指令脉冲计数器绝对位置为 0，作为原点位置
result=SMCSetPositionUnit(MyAxis,0)
```

运行结果：

负方向恒速运动，当遇到原点信号，锁存当前位置值，运动停止。再次启动定长运动，回到锁存位置值处，将当前计数值清 0。

2.4.2.3 EZ 位置锁存

雷赛控制器提供了 EZ 锁存功能，该功能可以实现在碰到 EZ 信号时将当前位置锁存，使用该功能可以实现精确回原点运动，一般步骤如下：

1）使用 SMCSetEzLatchMode 函数设置 EZ 锁存模式；

2）使用 SMCResetEzLatchFlag 函数清除 EZ 锁存标志；

3）设置轴运动参数，并启动运动；

4）判断 EZ 锁存标志是否有效，有效则停止运动；

5）重新设置轴运动速度为低速度，并使用定长运动到 EZ 锁存位置；

6）回到 EZ 锁存位置后，指令脉冲计数器清零。

相关指令：

名称	功能	参考
SMCResetEzLatchFlag	清除 EZ 锁存标志	4.21 节
SMCGetEzLatchFlag	读取 EZ 锁存标志	

SMCGetEzLatchValueUnit	读取 EZ 锁存值	
SMCSetEzLatchMode	设置 EZ 锁存模式	
SMCGetEzLatchMode	读取 EZ 锁存模式	

例　程：使用 EZ 锁存功能进行精确回原点运动

```
"""""""""""""""""""""变量定义"""""""""""""""""""
Dim MyCardNo,MyAxis,Myenable,Mylogic,Mysource,result
DimLatchValue '锁存值
MyAxis = 0'轴号
Myenable = 1'使能原点锁存
Mylogic = 0'锁存方式为下降沿锁存
Mysource = 0'锁存位置源为指令位置
"""""""""""""""""""""指令调用执行"""""""""""""""""""
'第一步、设置设置 EZ 锁存参数
result=SMCSetEzLatchMode(MyAxis,Myenable,Mylogic,Mysource)
'第二步、清除 EZ 锁存标志位
result=SMCResetEzLatchFlag(MyAxis)
'第三步、设置 0 号轴速度等参数
result=SMCSetProfileUnit(Myaxis,0,1000,0.1,0.1,0)
'第四步、启动 0 号轴执行恒速运动，负方向
result=SMCVmove(MyAxis,0)
'第五步、判断 EZ 锁存标志位是否有效
While(SMCGetEzLatchFlag(Myaxis) = 0)
Wend
'第六步、EZ 锁存标志位有效后，运动停止
result=SMCSetDecStopTime(Myaxis,0.15)   '设定减速停止速度时间
result=SMCStop(MyAxis,0)
'第七步、判断当前运动是否停止
While (SMCCheckDone(MyAxis) = 0)
Wend
```

```
'第八步、设置 0 号轴速度等参数
result=SMCSetProfileUnit(Myaxis,0,1000,0.1,0.1,0)
'第九步、获取 EZ 锁存值，并赋值给变量 LatchValue
LatchValue = SMCGetEzLatchValueUnit(Myaxis)
'第十步、0 轴定长运动到 EZ 锁存位置点
result=SMCPmoveUnit(MyAxis, LatchValue,1)
'第十一步、判断当前运动状态，等待停止
While (SMCCheckDone(MyAxis) = 0)
Wend
'第十二步、设置 0 轴的指令脉冲计数器绝对位置为 0，作为原点位置
result=SMCSetPositionUnit(MyAxis,0)
```

运行结果：

负方向恒速运动，当遇到原点信号，锁存当前位置值，运动停止。再次启动定长运动，回到锁存位置值处，将当前计数值清 0。

2.4.3 位置比较输出

雷赛控制器提供了位置比较功能，通过位置比较可输出 IO 等功能。包括单轴低速位置比较、单轴高速位置比较和一组二维低速位置比较。当电机运动到预先设置的位置时，自动触发特定的输出口。该功能在轨迹运动中用于控制点胶阀的开关、触发照相机快门等动作十分方便。

2.4.3.1 一维低速位置比较

雷赛控制器对每个轴都提供了一组一维低速位置比较，每轴最多都可以添加 256 个比较点。一维低速位置比较的触发延时时间大于 1ms。比较模式支持支持“大于等于”或“小于等于”，支持 IO 翻转、高低电平等动作。

相关指令：

名称	功能	参考
SMCCompareSetConfig	设置一维位置比较器参数	4.23 节
SMCCompareGetConfig	读取一维位置比较器参数	
SMCCompareClearPoints	清除已添加的所有一维位置比较点	

SMCCompareAddPointUnit	添加一维位置比较点	
SMCCompareGetCurState	读取比较状态	

⚠注意：（1）每轴的位置比较都是独立进行的。

（2）执行位置比较时，每个比较点的触发是按照添加的比较点顺序执行的，即如果有一个比较点没有被触发比较动作，那么后面的比较点是不会起作用的。

例　程：一维低速位置比较。

```
Dim MyCardNo,Myaxis,Myenable,Mycmp_source,Mydir,Myaction,result
Dim Mypos,Myactpara
Myaxis = 0'轴号
Myenable = 1'设置比较器使能
Mycmp_source = 0'设置比较源为指令位置
result=SMCCompareSetConfig(Myaxis,Myenable,Mycmp_source) '设置 0 号轴比较器
result=SMCCompareClearPoints(Myaxis)          '清除 0 号轴的比较点及比较点个数
Mypos = 10000'设置比较位置为 10000pulse
Mydir = 1'设置比较模式为大于等于
Myaction = 3'设置触发功能为 IO 电平取反
Myactpara = 0'设置输出 IO 端口 0 触发功能
result=SMCCompareAddPointUnit(Myaxis, Mypos, Mydir, Myaction, Myactpara)
'添加比较点，位置 10000pulse，模式大于等于，触发时动作为输出端口 0 电平取反
Mypos = 30000'设置比较位置为 30000pulse
Mydir = 1'设置比较模式为大于等于
Myaction = 1'设置触发功能为 IO 端口置高电平
Myactpara = 3'设置输出 IO 端口 3 触发功能
result=SMCCompareAddPointUnit(Myaxis, Mypos, Mydir, Myaction, Myactpara)
'添加比较点，位置 30000pulse，模式大于等于，触发时动作为输出端口 3 置高电平
result=SMCSetProfileUnit(Myaxis,0,10000,0.1,0.1,0)'设置速度曲线参数
result=SMCPMoveUnit(Myaxis,50000,0) '0 号轴定长运动，相对模式
```

运行结果：

当运动到 10000pulse 时，通用输出口 0 电平将取反，即原来为低电平则反转为高电平，

原来为高电平则反转为低电平；当运动到 30000pulse 时，通用输出口 3 输出高电平。

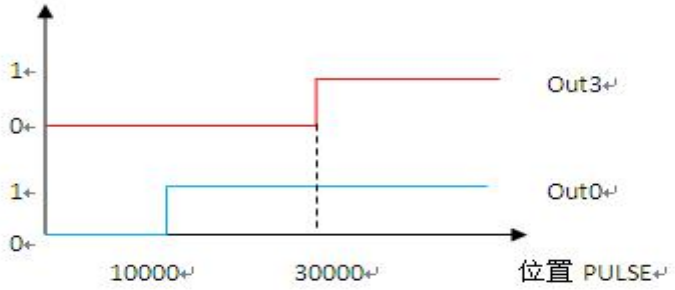


图 2.8　例程运行结果示意图

2.4.3.2 一维高速位置比较

雷赛控制器共有 2 个高速位置比较器，每个高速位置比较器均配有 1 个硬件位置比较输出接口。CMP1、CMP2 端口电路使用的是高速光耦（1MHz），对应硬件输出口 OUT16 和 OUT17。

每个 CMP 输出端口都可以与任意轴关联，支持多种比较模式，用户只需在函数里设置便可以使用，非常灵活方便。

其中“等于”、“小于”、“大于”比较模式为单次比较模式。“队列”模式提供 127 个比较点，采用先添加先比较，比较完可追加比较点，也可一次性添加多个比较点。“线性”模式适用于每个比较点位置都是按照线性增量依次增加的情况，用户只需要设置起始比较点位置以及线性增量值即可，非常方便。位置间时间间隔最小可达几微秒。

相关指令：

名称	功能	参考
SMCHcmpSetMode	设置高速比较模式参数	4.23 节
SMCHcmpGetMode	读取高速比较模式参数	
SMCHcmpSetConfig	配置高速比较器参数	
SMCHcmpGetConfig	读取配置高速比较器参数	
SMCHcmpClearPoints	清除已添加的所有高速位置比较点	
SMCHcmpAddPointUnit	添加/更新高速比较位置	
SMCHcmpSetLinerUnit	设置高速比较线性模式参数	
SMCHcmpGetLinerUnit	读取高速比较线性模式参数	
SMCHcmpGetCurState	读取高速比较状态	

⚠注意：（1）每个比较器的位置比较都是独立进行的。如不再使用位置比较下比较口 OUT16 和 OUT17 时，我们需启用指令 SMCHcmpSetMode(Myhcmp,0)下禁止模式，释放比较端口，否则 0UT16、0UT17 单独作为输出口时，可能无法使用。

（2）在队列及线性比较模式中，执行位置比较时，每个比较点的触发是按照添加的比较点顺序执行的，即如果有一个比较点没有被触发比较动作，那么后面的比较点是不会被触发的。

例程 1：一维高速位置比较，当定长运动到计数值大于 5000 时比较器 0（OTU116）端口输出低电平并保持。

```
Dim MyCardNo,Myhcmp,Myaxis,MyCmpSource,MyCmpLogic,MyCmpMode, result
Dim MyTime,MyCmpPos
Myhcmp = 0'高速比较器，对应硬件 CMP 端口 0
Myaxis = 2'轴号
MyCmpSource=0'比较位置源，0：计数值，1：编码器值
MyCmpLogic=0'有效电平，0：低电平，1：高电平
MyTime=0'电平翻转时间,只有比较模式为 4 或 5 时起作用
result=SMCHcmpSetConfig(Myhcmp,Myaxis,MyCmpSource,MyCmpLogic,MyTime)
'设置 2 号轴高速比较器 0，比较源为指令位置，有效电平为低电平
MyCmpMode =3'比较模式 3：大于
result=SMCHcmpSetMode(Myhcmp, MyCmpMode) '设置比较器 0 比较模式为模式 3（大于）
result=SMCHcmpClearPoints(Myhcmp)          '清除位置比较器
MyCmpPos = 50000'比较位置
result=SMCHcmpAddPointUnit(Myhcmp, MyCmpPos)'更新高速比较位置为 50000 pulse
result=SMCSetProfileUnit(Myaxis,2000,10000,0.1,0.1,3000)'设置速度曲线参数
result=SMCPMoveUnit(Myaxis,100000,1)      '2 号轴定长运动、绝对模式
```

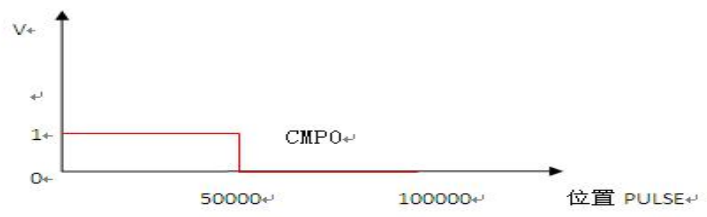


图 2.9　例程运行结果示意图

例程 2：队列比较模式，模式 4，高速比较口 0，输出电平翻转时间 500ms。

```
Dim MyCardNo,Myhcmp,Myaxis,MyCmpSource,MyCmpLogic,MyCmpMode
Dim MyTime,MyCmpPos,result
Dim remained,current,runned
Myhcmp = 0'高速比较器，对应硬件 CMP 端口 0
Myaxis = 0'轴号
MyCmpSource=0'比较位置源，0：计数值，1：编码器值
MyCmpLogic=0'有效电平，0：低电平，1：高电平
MyTime=500000'电平翻转时间,单位：us
result=SMCHcmpSetConfig(Myhcmp,Myaxis,MyCmpSource,MyCmpLogic,MyTime)
'设置 0 号轴的高速比较器 0，比较源为指令位置，有效电平为低电平
MyCmpMode =4'比较模式 4：队列模式
result=SMCHcmpSetMode(Myhcmp, MyCmpMode) '设置比较器 0 比较模式为模式 4
result=SMCHcmpClearPoints(Myhcmp)              '清除位置比较器
MyCmpPos = 10000'比较位置数值
result=SMCHcmpAddPointUnit(Myhcmp, MyCmpPos) '添加位置比较点参数
MyCmpPos = 30000'比较位置数值
result=SMCHcmpAddPointUnit(Myhcmp, MyCmpPos) '添加位置比较点参数
MyCmpPos = 70000'比较位置数值
result=SMCHcmpAddPointUnit(Myhcmp, MyCmpPos) '添加位置比较点参数
result=SMCSetProfileUnit(Myaxis,0,10000,0.1,0.1,0)'设置速度曲线参数
result=SMCPMoveUnit(Myaxis,100000,1)            '0 号轴定长运动，绝对模式
delay(100)
while(SMCCheckDone(Myaxis)=0)
      delay(300)
      printSMCHcmpGetCurState(Myhcmp,remained,current,runned)   '读取比较状态参数
      print remained,current,runned        '打印剩余比较点，当前比较点值，已比较点数
wend
```

运行结果：

当 0 号轴运动经过位置 10000、30000、70000 pulse 时，CMP0 端口输出低电平，低电平持续时间为 500ms。同时可打印剩余比较点，当前比较点值，已比较点数等信息。

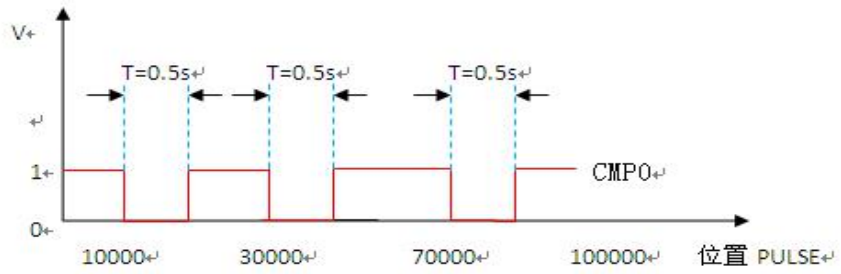


图 2.10　例程 2 运行结果示意图

例程 3：线性比较模式，模式 5，高速比较口 0，输出电平翻转时间 500ms。

```
Dim MyCardNo,Myhcmp,Myaxis,MyCmpSource,MyCmpLogic,MyCmpMode , result
Dim MyTime,MyCmpPos
Myhcmp = 0'高速比较器，对应硬件 CMP 端口 0
Myaxis = 0'轴号
MyCmpSource=0'比较位置源，0：计数值，1：编码器值
MyCmpLogic=0'有效电平，0：低电平，1：高电平
MyTime=500000'翻转时间 500ms,只有比较模式为 4 或 5 时起作用
result=SMCHcmpSetConfig(Myhcmp,Myaxis,MyCmpSource,MyCmpLogic,MyTime)
'设置 0 号轴的高速比较器 0，比较源为指令位置，有效电平为低电平
MyCmpMode =5'比较模式 5，线性模式
result=SMCHcmpSetMode(Myhcmp, MyCmpMode) '设置比较器 0 比较模式为模式 5
result=SMCHcmpClearPoints(Myhcmp)   '清除位置比较器
MyCmpInc = 20000'线性增量，单位 pulse
MyCmpCount = 5'比较次数
result=SMCHcmpSetLinerUnit(Myhcmp,MyCmpInc, MyCmpCount)
'设置比较器 0 线性模式参数，线性增量为 20000 pulse，比较次数为 5
MyCmpPos = 10000'比较点初始位置值
result=SMCHcmpAddPointUnit(Myhcmp, MyCmpPos) '添加位置比较点参数
result=SMCSetProfileUnit(Myaxis,0,10000,0.1,0.1,0)'设置速度曲线参数
result=SMCPMoveUnit(Myaxis,100000,1)          '0 号轴定长运动，绝对模式
```

运行结果：当 0 号轴运动经过位置 10000、30000、50000、70000、90000 pulse 时，CMP0 端口输出低电平，低电平持续时间为 500ms。

2.4.3.3 二维低速位置比较

二维低速位置比较相关函数：

名称	功能	
SMCCompareSetConfigExtern	设置二维位置比较器	4.23 节
SMCCompareGetConfigExtern	回读二维比较器参数	
SMCCompareClearPointsExtern	清除二维位置比较点	
SMCCompareAddPointExternUnit	添加二维位置比较点	
SMCCompareGetCurrentPointExternUnit	读取当前二维位置比较点位置	
SMCCompareGetPointRunnedExtern	查询已经比较过的二维比较点个数	
SMCCompareGetPointsRemainedExtern	查询可以加入的二维比较点个数	

2.4.4 PWM 输出

雷赛控制器支持 PWM 立即输出功能，用户只需要简单设置 PWM 输出通道、频率及占空比参数即可实现 PWM 输出功能，非常方便。如下图所示，控制器输出的 PWM 波形的周期为 t2（频率即为 1/t2），占空比为 t1/t2，幅值为 V1 = 5V。

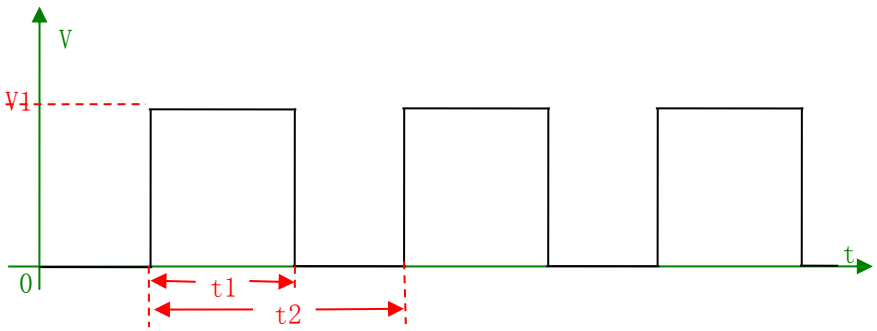


图 2.6 PWM 输出示意图

相关指令：

名称	功能	参考
SMCSetPwmOutput	设置PWM输出信号的频率、占空比参数	4.15 节
SMCGetPwmOutput	读取 PWM 输出信号的频率、占空比参数	

例　程：执行 PWM 立即输出功能。

```
Dim ionum,value,ration,value1,ration1, result
ionum =0'PWM 输出口 0
value=1000'PWM 输出频率
ration=0.4'PWM 输出占空比
result=SMCSetPwmOutput(ionum,ration,value)  'PWM 输出占空比、频率参数设置
result=SMCGetPwmOutput(ionum,ration1,value1) '回读 PWM 参数设置
print ration1,value1  '打印回读 PWM 参数
```

运行结果：　PWM0 输出口立即发出频率为 1000Hz，占空比为 0.4 的输出信号

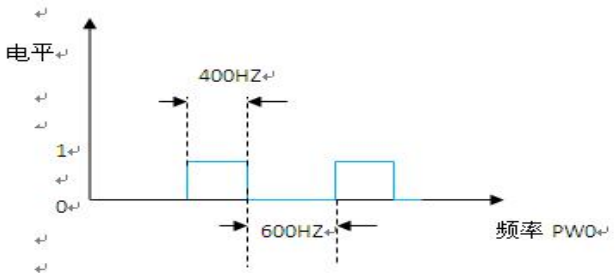


图 2.7　例程运行结果图

2.4.5 伺服专用功能

设备中有交流伺服电机时，使用雷赛控制器上的伺服电机控制信号 SEVON、RDY、ALM、INP 和 ERC 就显得十分方便。

2.4.5.1 ALM 报警信号

ALM 信号是从伺服电机驱动器发给控制器的状态信号，用来报告伺服驱动器或电机出错。控制器接收到 ALM 信号时，将立即停止发出脉冲，该过程是一个硬件处理过程。

相关指令：

名称	功能	参考
SMCSetAlmMode	设置报警信号参数	4.17 节
SMCGetAlmMode	读取报警信号参数	
SMCReadAlarmPin	读取报警信号电平	

例　程：设置轴 0，报警信号有效电平为低，报警信号有效后，运动立即停止

```
Dim Myaxis,Myel_enable, alm_logic,Myel_mode,Myel_enable1
Dim alm_logic1,Myel_mode1
Myaxis = 0'轴号
Myel_enable = 1'报警信号使能
alm_logic = 0'ALM 信号的有效电平为低 0
Myel_mode = 0'ALM 信号有效后立即停止，目前只支持立即停止
result=SMCSetAlmMode(Myaxis,Myel_enable,alm_logic,Myel_mode) '报警信号参数设置
result=SMCGetAlmMode(Myaxis,Myel_enable1,alm_logic1,Myel_mode1)'读报警信号参数
print   Myel_enable1, alm_logic1,Myel_mode1     '打印报警信号参数值,结果 1、0、0
SMCVMove(Myaxis,1)                  '启动恒速运动
```

运行结果：

运行程序，当报警信号接口输入低电平时，轴 0 运动立即停止。

2.4.5.2 INP 到位信号

INP 信号是从伺服电机驱动器发给控制器的状态信号，告知运动控制器伺服电机已经停止。伺服电机驱动器通常有一个位置偏差计数器，记录指令脉冲和位置反馈脉冲之间的偏差。伺服电机驱动器将控制电机运动使位置偏差趋于 0，但是电机实际位置总是滞后于指令脉冲。所以当运动控制器的指令脉冲发送完毕时，伺服电机并没有立即停止，而是继续运动，如图 2.2.3.5 所示，直到位置偏差趋于 0；这时驱动器将发出一个 INP 信号。

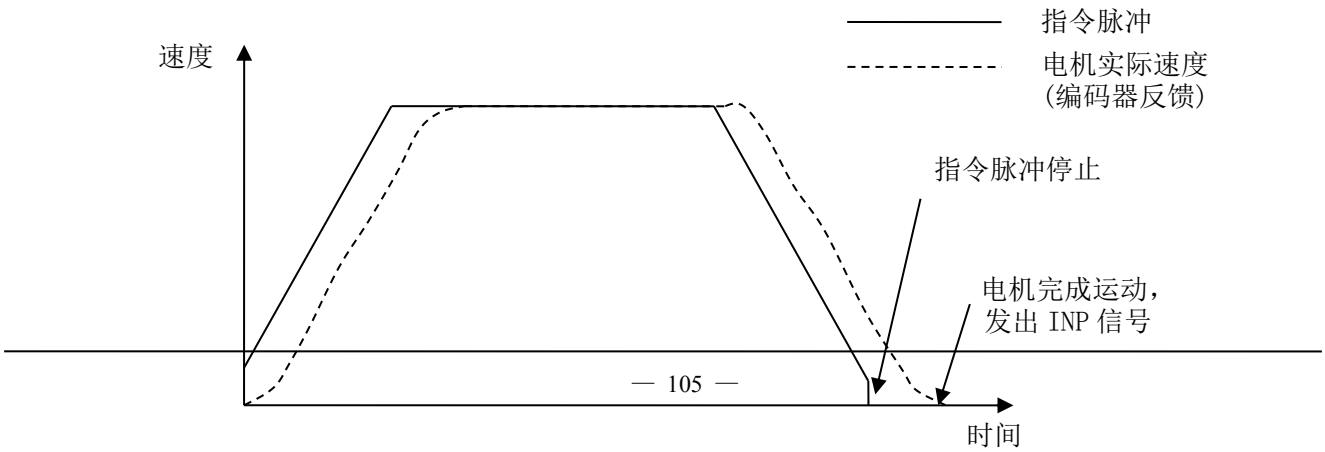


图 2.5　伺服定位完成时的 INP 信号

相关指令：

名称	功能	参考
SMCSetInpMode	设置 INP 到位信号	4.17 节
SMCGetInpMode	回读 INP 到位信号参数	
SMCReadInpPin	读取 INP 到位信号	

⚠注意：SMC606 控制器本身没有 INP 硬件接口，使用前必须进行 IO 映射。

例　程：使能轴 0 的到位 INP 信号。

```
Dim Myaxis,Myel_enable, alm_logic,Myel_mode
Myaxis = 0'轴号
Myel_enable = 1'INP 信号使能
alm_logic = 0'INP 信号的有效电平为低 0
result=SMCSetAxisIoMap(Myaxis,7,6,0,0) '输入 IO 口 0 作为 INP 信号输入口
result=SMCSetInpMode (Myaxis,Myel_enable,alm_logic)   '使能 INP 信号
result=SMCSetProfileUnit(0,0,100,0.1,0.1,0) '设置轴 0 速度参数
result=SMCPMoveUnit(0,1000,1)                    '设置轴 0 定长运动 1000unit
while (SMCCheckDone(Myaxis)=0)                    '等待运动停止
wend
While   SMCReadInpPin(Myaxis)=1'判断 INP 信号是否有效
      SMCWriteOutBit(1,0)                          'INP 信号有效后，IO 口 1 输出低电平
Wend
```

运行结果

运行程序，当 INP 信号有效后，可看到输出口 1 输出低电平。

2.4.5.3 SEVON 伺服驱动使能

SEVON 是控制器输出给伺服电机驱动器的控制信号，当 SEVON 信号为无效状态时，伺

服驱动器不使能，电机处于自由状态；当 SEVON 信号有效时，伺服驱动器使能，电机锁紧；等待指令脉冲信号。一般驱动使能信号为低电平时，SEVON 信号处于有效状态。

相关指令：

名称	功能	参考
SMCWriteSevonPin	设置伺服驱动使能信号参数	4.17 节
SMCReadSevonPin	读取伺服驱动使能信号参数	

例　程：读取轴 0-5 的驱动使能信号，若没有启动则启动驱动使能信号，有效电平为高。

```
Dim Myaxis,Myel_logic, my_value
Myaxis = 0'轴号
Myel_logic = 0'  端口电平，0：低电平，1：高电平
For Myaxis=0 to 5'轴号 0-5
    my_value=SMCReadSevonPin(Myaxis)     '读取当前轴的使能电平状态
     If   my_value=1then'若使能处于无效电平状态
          SMCWriteSevonPin(Myaxis, Myel_logic)'设置当前轴的使能电平状态 0 有效
     endif
next
```

运行结果：程序运行后，控制器上驱动伺服使能 LED 灯亮。

2.4.5.4 ERC 伺服误差清除

ERC 信号是控制器输出给伺服电机驱动器的控制信号。伺服驱动器依赖电机目标位置（即要求电机达到的位置）和当前位置（即电机已经到达的位置）之间的误差来驱动电机运动，若这个误差为零电机将停止运动。当伺服驱动器接收到运动控制器发出的 ERC 信号后，会立即清除误差并停止电机运转。

相关指令：

名称	功能	参考
SMCWriteErcPin	设置伺服驱动清除使能信号参数	4.17 节
SMCReadErcPin	读取伺服驱动清除使能信号参数	

例　程：轴 0-5 运动，运动一段时间后（或发生运动错误等），启动伺服驱动清除信号，有效电平为高，读取驱动清零状态.

```
Dim Myaxis,Myel_logic
Myaxis = 0 '轴号
Myel_logic = 1 '输出电平，0：低电平，1：高电平
For   Myaxis=0 to 5 '轴号 0-5
        printSMCWriteErcPin (Myaxis, Myel_logic)   '设置当前轴的伺服清除信号
next
For Myaxis= 0 to 5 '轴号 0-5
        Print SMCReadErcPin (Myaxis)          '读取当前轴的伺服清除信号
next
```

运行结果：程序运行后，控制器上 ERC 输出高电平

2.4.6 限位功能

2.4.6.1 EL 硬件限位

雷赛控制器提供了限位开关来设置限位功能。用户必须根据设备的限位开关硬件接线，来设置限位开关工作的有效电平。

相关指令：

名称	功能	参考
SMCSetElMode	设置 EL 限位信号参数	4.24 节
SMCGetElMode	读取 EL 限位信号参数	

例　程：设置轴 0，限位电平为低电平有效，限位停止方式为立即停止

```
Dim Myaxis,Myel_enable,Myel_logic,Myel_mode
Dim Myel_enable1,Myel_logic1,Myel_mode1
Myaxis = 0'轴号
Myel_enable = 1'正负限位使能
Myel_logic = 0'正负限位低电平有效
Myel_mode = 0'正负限位停止方式为立即停止
```

```
result=SMCSetelmode(Myaxis,Myel_enable,Myel_logic,Myel_mode) '设置硬限位参数
result=SMCGetelmode(Myaxis,Myel_enable1,Myel_logic1,Myel_mode1)'读硬限位参数
print   Myel_enable1,Myel_logic1,Myel_mode1                    '打印硬件限位参数值
result=SMCSetProfileUnit(Myaxis,0,1000,0.1,0.1,0)              '设置运行速度参数
result=SMCVMove(Myaxis,0)                                      '启动恒速运动
```

运行结果：当控制器负向硬件限位信号输入接口为低电平时，运动停止。

2.4.6.2 SL 软件限位

雷赛控制器提供了软件限位功能。当计数值在限位设定值外时，限位信号有效，运动将会自动停止。

相关指令：

名称	功能	参考
SMCSetSoftlimitUnit	设置软件限位信号参数	4.24 节
SMCGetSoftlimitUnit	读取软件限位信号参数	

例　程：设置轴 0，软件限位位置分别为+1000，-1000，限位信号有效后，运动立即停止

```
Dim Myaxis,Myel_enable,Myel_logic,Myel_mode
Myaxis = 0'轴号
Myel_enable = 1'正负限位使能
source_sel = 0'以指令计数器作为比较源 0
Myel_mode = 0'正负限位停止方式为立即停止
N_limit=-1000'负限位位置
P_limit =1000'正限位位置
result=SMCSetSoftlimitUnit(Myaxis,Myel_enable,source_sel,Myel_mode,N_limit,P_limit)'软限位参数设置
result=SMCSetProfileUnit(Myaxis,0,1000,0.1,0.1,0)        '设置运行速度参数
result=SMCVMove(Myaxis,0)                                '启动恒速运动
```

运行结果：当运动轴 0 的计数值大于 1000 或小于-1000 时，轴 0 运动将停止。

2.4.7 EMG 急停功能

雷赛控制器提供了急停开关功能。对硬件接口电路进行接线，然后调用急停开关设置函数进行设置。

相关指令：

名称	功能	参考
SMCSetEMGMode	EMG 急停信号使能、电平参数设置	4.25 节
SMCGetEMGMode	读取 EMG 急停信号使能、电平参数	

⚠注意：若控制器本身没有急停开关硬件接口，使用前必须进行 IO 映射。

例　程：设置通用输入口 0 接口为所有轴的急停信号，低电平有效

```
Dim CardNo,Axis,IoType, MapIoType, MapIoIndex,Myenable,Mylogic, Myenable1,Mylogic1,result
Dim Filter
For Axis = 0 To 5'循环，依次对 0~5 号轴进行轴映射配置
    result=SMCSetAxisIoMap(Axis, 3, 6, 0, 0.01 )
'设置轴 IO 映射，将通用输入 0 作为各轴的急停信号，EMG 信号滤波时间为 0.01 秒
Next Axis
Myenable = 1'急停信号使能
Mylogic = 0'急停信号低电平有效
For Axis = 0 To 5'循环，依次对 0~5 号轴进行使能、电平设置
    result=SMCSetEMGMode(Axis,Myenable,Mylogic)     '设置 EMG 信号使能，低电平有效
    result=SMCGetEMGMode(Axis,Myenable1,Mylogic1) '读取 EMG 信号使能参数
   print Myenable1,Mylogic1                                  '打印设置的 EMG 参数
   SMCVMove(Axis,1)                                            '恒速运动
Next Axis
```

运行结果：　当输入信号 IN0 为低电平后，各轴的急停开关有效，所有轴运动停止。

2.5 系统功能

2.5.1 系统时间

控制器系统自带系统时间，掉电后不会丢失。

相关指令：

名称	功能	参考
RTC_TIMES	读取 RTC 的运行秒数，掉电后不会丢失	4.1.10 节
RTC_TIME	读取 RTC 的当前时间，掉电后不会丢失	
RTC_DATE	读取 RTC 的当前日期，掉电后不会丢失。	
TICKS	读取系统开机后毫秒计时时间。	

⚠ 注意：系统时间等需在有电池供电情况下断电才不会丢失。

例程 1：打印当前时间，当前日期。

```
Print RTC_TIME'打印控制器当前时间
Print RTC_DATE'打印控制器当前日期
```

例程 2：运行一段程序，打印所需时间。

```
dim t1,t2,i,m
t1=ticks '起始时间
For i=0 to 1000
   m=i+1
next i
t2=ticks '结束时间
print t2-t1                    '打印运行时间
```

运行结果：打印结果运行循环时间值，单位为 ms。

2.5.2 中断功能

系统自带中断功能包括定时器、输入口中断、报警中断、软件限位中断、硬件限位中断功能等。中断功能出厂默认为禁用状态，建议在全局变量定义完成之后开启中断功能，开启指

令“Int_Enable(1)”。

相关指令：

名称	功能	参考
OnTimer0～控制器最大定时器-1	定时器计数结束中断入口	4.1.5 节
OnInOn0～最大输入 IO 口数-1	通用数字输入信号口，上升沿产生中断的程序入口	4.1.6 节
OnInOff0～最大输入 IO 口数-1	通用数字输入信号口，下降沿产生中断的程序入口	
OnSoftEl0～最大电机轴数-1	各轴软件限位信号的中断处理程序入口	
OnEl0～最大电机轴数-1	硬件限位信号的中断处理程序入口	
OnAlarm0～最大电机轴数-1	各轴伺服电机的报警信号ALM的中断处理程序入口	

例　程：中断程序应用

```
"""""""""""""""""""变量定义"""""""""""""""""""
dim result              '函数返回值
dim axis=0'轴号
dim pos                 '位置


Int_Enable(1)           '开启中断，建议变量定义完成之后开启
"""""""""""""""""""指令调用执行"""""""""""""""""""
'第一步、开启 2 号定时器，定时时间为 500 毫秒
timer_start(2, 500)
'第二步、设置软限位参数
result = SMCSetSoftLimitUnit(axis,1,0,0,-1000,1000)
'第三步、设置硬限位参数
result = SMCSetELMode(axis,1,0,1)
'第四步、时刻关注运动位置状况
While 1
    pos = SMCGetPositionUnit(axis)
```

```
        print"position=",pos
        delay (500)
wend
end' 程序结束
"""""""""""""""""""中断子程序"""""""""""""""""
'2 号定时器中断入口子程序
ontimer2:
dim state=0
result = SMCWriteOutBit(3,state)       '操作 3 号输出口
state = not state                      '状态取反
timer_start(2, 500)                    '重新启动 2 号定时器
end
'1 号输入信号上升沿触发中断入口子程序
oninon1:
        result=SMCPMoveUnit(axis,2000,0)          '电机正转 2000 个脉冲
end
'2 号输入信号下降沿触发中断入口子程序
oninoff2:
        result=SMCPMoveUnit(axis,-2000,0)         '电机反转 2000 个脉冲
end
'0 号轴硬限位中断入口子程序
OnEl0:
dim state=0
state= SMCAxisIoStatus(axis)                    '0 轴运动信号相关状态
if   read_bit(1,state)=1then'检测是否为正限位
        print"limit at forward!"'提示：是正限位
endif
if     read_bit(2,state)=1then'检测是否为负限位
        print"limit at backward!"'提示：是负限位
endif
```

```
end
'0 号轴软限位中断入口子程序
OnSoftEl0:
dim state=0
state = SMCAxisIoStatus(axis)                    '0 轴运动信号相关状态
if   read_bit(6,state)=1 then'检测是否为软件正限位
      print"soft limit at forward!""提示：是软件正限位
endif
if   read_bit(7,state)=1 then'检测是否为软件负限位
      print"soft limit at backward!""提示：是软件负限位
endif
end
```

运行结果：用定时中断方式控制输出口每 500 毫秒闪烁一次；用数字输入端口 1、2 的中断方式控制电机正、反转；软限位、硬限位信号触发中断处理程序发出限位报警。

2.5.3 多任务功能

雷赛控制器运行程序，可同时开启多个任务，最多同时运行 6 个任务。

相关指令：

名称	功能	参考
Call	调用过程（子程序）	4.1.4节
SUB	定义子程序	
RUN	启动多任务	
END	结束当前任务或程序	
STOP	任务强制停止	
PAUSE	暂停当前任务	
HALT	停止全部任务	

例　程：多任务控制指令程序。

```
auto:                    '控制器开机后，自动启动程序
undim *                  '删除之前定义的所有变量及数组
```

```
SMCSetProfileUnit(0,0,1000,0.1,0.1,0)        ' 设置运动速度参数
run1,check_stop            '启动“check_stop”任务，任务号为 1
while1
        while   SMCReadInBit(1)=0'如果输入口 1 为低电平，则运行恒速运动
                SMCVMove(0,0)                    'X 轴恒速运动
                While   SMCCheckDone(0)=0'等待 X 轴运动停止
                wend
        wend
        if SMCReadInBit(3)=0 then'如果输入口 3 为低电平，则
             stop1'停止任务 1 运行，即:之后输入口 2 不可控制电机停止
       endif
wend
end'程序结束
check_stop:                                    '任务名
while1
        if   SMCReadInBit(2)=0 then'如果输入口 2 为低电平，
             SMCSetDecStopTime(0,0.15)          '设定减速停止速度时间
             SMCStop(0,0)                       'X 轴电机减速停止
             print"stop"
        endif
wend
end
```

运行结果：用输入口1为低电平时轴0电机运动，用输入口2为低电平控制电机停止；用输入口3为低电平则关闭任务1，再次启动电机运动，电机不能再由输入口2来控制电机停止的功能。

2.5.4 断电保存功能

系统自带断电保存功能，可将重要数据保存于寄存器中，数据将不会丢失。重启控制器后，可将重要数据读取出来。

相关指令：

名称	功能	参考
STATICREG/NVRAM_INT	读写断电保护寄存器，数据类型为32位整数	4.1.8节
NVRAM/NVRAM_FLOAT	读写断电保护寄存器，数据类型为32位浮点数,5位固定小数	
NVRAM_BYTE	读写断电保护寄存器，数据类型为8位整数	
NVRAM_SHORT	读写断电保护寄存器，数据类型为16位整数	

在BASIC控制器中，断电保持数据区域共分4块铁电保存，每块字节数固定，均为32768个。对于INT型数据(4个字节)，能够保存8192个数据，范围为NVRAM_INT(0)~NVRAM_INT(8191)；对于FLOAT型数据(4个字节)，能够保存8192个数据，范围为NVRAM_FLOAT(0) ~NVRAM_FLOAT(8191)；对于BYTE型数据(1个字节)，能够保存32768个数据，范围为NVRAM_BYTE(0)~NVRAM_BYTE(32767)；对于SHORT型数据(2个字节)，能够保存16384个数据，范围为NVRAM_SHORT(0)~NVRAM_SHORT(16383)。

例程:突然断电后再次运行该程序，坐标自动恢复到上次断电时的坐标值。

```
auto:
dim pos_x,pos_y,result
SMCSetProfileUnit(0,0,1000,0.1,0.1,0)
SMCSetProfileUnit(1,0,1000,0.1,0.1,0)
If   staticreg(1)=33 then
     pos_x=staticreg(20)          '非第一次运行，读取铁电存储器内的坐标
     pos_y=staticreg(30)
     print pos_x, pos_y
else
     staticreg(1)=33'  第一次运行
     result=SMCSetPositionUnit(0,0) '设置 0 号轴坐标为 0
     result=SMCSetPositionUnit(1,0) '  设置 1 号轴坐标为 0
endif
While 1
```

```
    if SMCReadInBit(1)=0 then' 1 号输入口为 0 时，电机运动
        result=SMCPMoveUnit(0,3000,0'0 号轴正转、1 号轴反转
        result=SMCPMoveUnit(1,-3000,0)
        while   SMCCheckDone(0)=0andSMCCheckDone(1)=0' 等待 2 个电机停止
            pos_x =SMCGetPositionUnit(0)     ' 将当前坐标值写入铁电存储器
            pos_y =SMCGetPositionUnit(1)
             staticreg(20)= pos_x                ' 将当前坐标值写入铁电存储器
            staticreg(30)= pos_y
            printstaticreg(20), staticreg(30) ' 显示 0 号轴、1 号轴坐标
        wend
    endif
wend
end
```

运行结果：当运动间突然断电时，当前位置值将自动保存在 staticreg(20), staticreg(30)中，控制器重新上电后找到断电时的位置值 pos_x、pos_y。

2.6 存储功能

运动控制器具有 FLASH 存储和 U 盘存储以及 NAND 存储功能。

2.6.1 U 盘存储

控制器提供了 U 盘的读、写存储功能，以及相应的文件管理功能。

相关 U 盘读写指令：

名称	功能	参考
U_WRITE	存储变量或者数组至U盘中	4.31节
U_READ	从U盘里面读取变量或者数组	
U_STATE	检查U盘是否插入	
U_CHECKFILE	检查U盘存储文件是否存在。不用后缀名	

相关 U 盘文件指令：

名称	功能	参考
SMCUdiskGetFirstFile	读取第一个文件参数	4.31节
SMCUdiskGetNextFile	读取下一个文件参数	
SMCUdiskCheckFile	检测文件是否存在	
SMCUdiskCopyFile	复制文件内容	

⚠注意：U 盘插入时，控制器需处于上电状态。否则上电后可能找不到 U 盘。

例程:　保存变量n、数组a至U盘、或从U盘中读取变量n、数组a，检测U盘中文件属性等

```
''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''
undim *                '删除之前定义的所有变量及数组
dim i,n,a(10)     '定义变量 n、数组 a
dim filename(20),filesize,fileid,filetype,filename1(20)
n=9'变量值
filetype=0'文件类型，0-basic,1-gcode,2-set 参数设置值,3-固件文件
''''''''''''''''''''''''''指令调用执行''''''''''''''''''''''''''
'第一步、在控制器上插入 U 盘
'第二步、将数据存入 U 盘中，保存为 mydata.sav 文件
While1
    If   SMCReadInBit(1)=0 then'1 号输入口为 0 时，存入数据
        If   U_STATE= true then'如果 U 已插上
            for i=0 to n
                a(i)=i* i                    '写入值
                Print"write:",a(i)    '打印写入值
            Next i
            U_WRITE"mydata",n,a    '存入数据至 U 盘，名称为"mydata"
        Else    ' 若 U 未插上，给出提示
            Print"U disk is unavailable."
        endif
    endif
    '第三步、读取 U 盘中 mydata.sav 文件数组值
```

```
        If  SMCReadInBit(2)=0  then'2 号输入口为 0 时，读取数据
            If  U_STATE=true  and  U_CHECKFILE("mydata")=true then
                U_READ"mydata",n,a '从 U 盘读取数据，名称为"mydata"
                For i=0 to n
                    Print"read:",a(i) '打印数据
                Next i
            Else   'U 未插上或文件不存在，给出提示
                Print "U disk is unavailable or the file doesn't exist."
            endif
        endif
    '第四步、读取 U 盘中第一个文件属性值，随机读取
        If SMCReadInBit(3)=0 then' 3 号输入口为 0 时，读取文件参数
            result=SMCUdiskGetFirstFile(filename(0),filesize,fileid,filetype)'读取第一个文件
            Print*filename,filesize,fileid,filetype '打印文件名，大小，类型
    '第五步、读取 U 盘中下一个文件属性值
            result=SMCUdiskGetNextFile(filename1(0),filesize,fileid,filetype)'读取下一个文件
            Print*filename1,filesize,fileid,filetype   '打印值
    '第六步、检测 U 盘中文件是否存在，需加上后缀名
            result=SMCUdiskCheckFile(filename(0),filesize,filetype) '检测文件是否存在
            Print   filesize,filetype '打印文件名属性，若;filesize 值为-1
    '第七步、U 盘中文件之间复制，需加上后缀名
            result=SMCUdiskCopyFile("123.bas","153.bas",0,2)   '将 U 盘中文件 123.bas 内容复制到文件
153.bas 中
    '第八步、U 盘中文件复制到 FLASH 中，需加上后缀名
            result=SMCUdiskCopyFile("123.bas","",0,2)   '将 U 盘中文件 123.bas 内容复制到控制器 FLASH
中，由于 FLASH 中文件名为固定文件名，可以省略。
        endif
    wend
    end
```

运行结果：

当IO口1为低电平时，在U盘根目录下生成一个mydata.SAV文件并存储一组数组a，当IO口2为低电平时，读取并打印U盘存储值，当IO口3为低电平时，读取U盘中文件属性等参数。

2.6.2 FLASH 按块存储

控制器提供了FLASH存储功能，不同控制器有不同的FLASH块，如SMC104:0-9,SMC606:0-99。

相关指令：

名称	功能	参考
FLASH_REG	直接读取 FLASH 的存储变量	4.31节
FLASH_READ	从内部 FLASH 里面读取变量或者数组	
FLASH_WRITE	存储变量或者数组至内部 FLASH 中	
FLASH_SECTSIZE	读取内部 FLASH 一个块可以存储的变量数	
FLASH_SECTES	读取内部 FLASH 的总块数	

用例1：写入和读取flash中值

```
''''''''''''''''''''变量定义''''''''''''''''''
dim x,y,a,b
x=1
y=2
''''''''''''''''''''指令调用执行''''''''''''''''''
'第一步、读取控制器 FLASH 参数
printFLASH_SECTSIZE'读取内部 FLASH 一个块可以存储的变量数
printFLASH_SECTES'读取内部 FLASH 的总块数
'第二步、FLASH 中写入数值
flash_write0,x,y
'第三步、读取 FLASH 中数值
flash_read0,a,b
print a,b
```

运行结果：打印a、b值分别为1,2

2.6.3 FLASH 按文件名存储

控制器提供了 FLASH 按文件名存储功能，相关指令如下所示。

名称	功能	参考
NAND_WRITE	存储变量或者数组到 NAND FLASH	4.31 节
NAND_READ	从 NAND FLASH 里面读取变量或者数组	
NAND_CHECKFILE	检查 NAND 存储文件是否存在	
NAND_DELETEFILE	删除 NAND 存储文件	

例　程：保存变量 n、数组 a 至 FLASH、或从 FLASH 中读取变量 n、数组 a。

```
dim i,n,a(10) ' 定义变量 n、数组 a
While 1
If  in(1)=1 then' 1 号输入口为 1 时，存入数据
    n=9
    For i=0 to n
        a(i)=i*i
        Print"write:",a(i)
    Next i
   NAND_WRITE"mydata", n,a ' 存入数据，名称为“ mydata”
endif
If  in(2)=1 then' 2 号输入口为 1 时，读取数据
    NAND_READ"mydata", n,a ' 读取数据，名称为“ mydata”
    For i=0 to n
        Print"read:",a(i) ' 打印数据
    Next i
endif
wend
```

运行结果： 运行该程序一次后，断电。再次上电运行此程序，并置输入口 2 为 1，则能显示保存在 flash 中的数据： 0,1,4,9,16,25,36,49,64,81 。

2.7 扩展功能

雷赛控制器提供控制器间的 CAN 总线级联，主控制器可控制多个从控制器或扩展模块，同时从控制器也可控制各扩展模块。见其连接示意图 2.7：

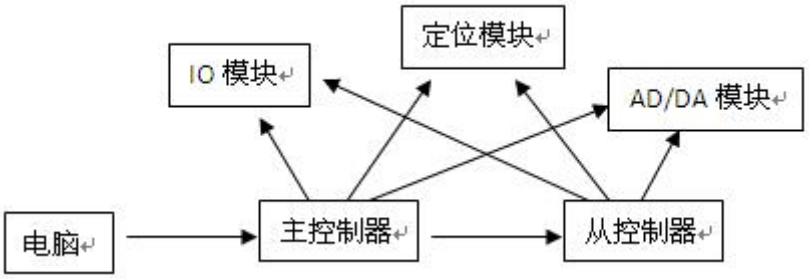


图 2.7 控制器 CAN 总线级联

2.7.1 CAN 总线级联

雷赛控制器提供多个控制器之间 CAN 总线级联，最多可以连接 16 个。如设置 2 个控制器间连接，主要步骤如下：

（1） 设置从控制器的波特率和 ID 号，可先通过 SMC Basic Studio 软件单独连接从控制器，编辑指令 setcan(baudrate，canid)设置波特率和 CAN ID 号。

（2） 将主控制器和从控制器的 CAN 口通过网线连接好。

（3） 打开 SMC Basic Studio 软件，设置主控制器波特率和 CAN ID 号，编辑不同的运动指令等，如 SMCPmoveCan，则可控制从控制器做定长运动。

例 程：控制器连接另一个控制器，从控制器做点位运动。

```
''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''
dim canid,axis,result
canid=1'从控制器 CAN ID 号
axis=0'轴号
''''''''''''''''''''''''''指令调用执行''''''''''''''''''''''''''
'第一步、设置主控制器波特率和 CAN ID 号
setcan(500,2)  '设置主控制器波特率为 500Kbps/S,CAN ID 号为 2
'第三步、将主控制器与从控制器的 CAN 口通过网线相连
'第四步、设置点位运动速度参数，S 段时间参数
```

```
SMCSetProfileCan(canid, axis,0,100, 0.1,0.1,0, result)
SMCSetSprofileCan(canid,axis,0,0, result)
'第五步、启动点位运动参数
SMCPmoveCan(canid,axis,1000,0, result)
```

⚠注意：运行前需设置从控制器的波特率和 ID 号（可先通过 SMC Basic Studio 软件单独连接从控制器，编辑指令 setcan(boute，canid)设置波特率和 CAN ID 号），主控制器上的 CAN ID 号与从控制器的 ID 号不能相同。

2.7.2 功能模块扩展

2.7.2.1 IO 模块

雷赛控制器提供了多种 IO 模块，其 IO 输出输入口数量各有不同。其与控制器连接大致步骤如下：

(1) 设置 IO 模块波特率、CAN ID 号(可在 IO 模块硬件接口上手动设置)。

(2) 将控制器和 IO 模块通过网线连接好。

(3) 打开 SMC Basic Studio 软件连接控制器，设置控制器波特率、CAN ID 号。

(4) 在SMC Basic Studio软件中编辑调用CAN IO指令等,如指令SMCReadInbitCan。

⚠注意：运行前需设置 CAN IO 模块的波特率（可直接在 IO 模块波特率和 CAN ID 硬件接口上手动设置），控制器上的 CAN ID 号与 IO 模块的 ID 后不能相同。

例　程：控制器连接 IO 模块，IO 模块输出口 1 输出低电平。

```
"""""""""""""""""""""变量定义"""""""""""""""""""
dim canid,bitno,state,result
canid=1'IO 模块 ID 号
bitno =1'IO 口号
state=0'输出 IO 状态值
"""""""""""""""""""""指令调用执行"""""""""""""""""""
'第一步、设置 IO 模块波特率为 500kbps/S 和 CAN ID 为 1
'第一步、设置主控制器波特率
setcan(500,3)  '设置主控制器波特率为 500Kbps/S，CAN ID 为 3
```

```
'第三步、将主控制器与 IO 模块的 CAN 口通过网线相连
'第四步、设置 IO 模块输出口 1 输出低电平
SMCWriteOutbitCan(canid,bitno,state,result)
```

2.7.2.2 定位模块

雷赛控制器提供了多种定位模块，各种定位模块具有不同的轴及 IO 等配置。如定位模块 EM02DP-L1，具有 2 个定位轴，普通输入输出 IO 各 6 个。各轴具有硬件限位、软件限位、INP、ERC、ORG、ALM、EMG 等功能。与控制器连接使用步骤大致如下：

（1） CAN 定位模块波特率和 CAN ID 设置。

定位模块上的波特率开关 SW0,其对应于不同的波特率设定值，目前我们使用的只有 badu1 和 badu2，具体对应见下表（0 表示 off、1 表示 on）：

SW0-badu1	SW0-badu2	波 特 率 值
0	0	1000KHz
1	0	500KHz
0	1	250KHz
1	1	125KHz

定位模块 CAN ID 号设定开关 SW1，其对应不同的 ID 值，目前只使用了 Addr1、Addr2、Addr3，具体设置见下表（0 表示 off、1 表示 on）：

Addr4	Addr3	Addr2	Addr1	CAN ID
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

（2） 控制器波特率和 CAN ID 号设置。

SMC104 需在控制器上设置 CAN ID 拨码开关 S2，目前只使用了 A1、A2、A3 对应 ID 范围为 1-7,其它作为备用，波特率是通过软件指令来设置，如 SMC104 设置波特率为 500K,即发送指令 SETCAN(500)

SMC104 Can ID 号设置

A4	A3	A2	A1	CAN ID 号
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

SMC300 及 SMC600 系列 CAN ID 和波特率都是在软件编辑器上写入指令，SETCAN(Baudrate,id)，id 为设定的 ID 号，波特率需与定位模块上波特率设定一致。

⚠注意：控制器上的 CAN ID 号与定位模块上的 ID 号不能相同。

（3）控制器的 CAN 口与定位模块的 CAN 口使网线连接，接好 24V 电源。

⚠注意：有的控制器有几个 CAN 口，需先确认使用哪个 CAN 口与定位模块连接。

（4）打开 SMC Basic Studio 软件，与控制器通讯。进入指令编辑界面，写入 BASIC 指令就可以实现对 CAN 定位模块的控制，类似对控制器的操作，只是指令不同。定位模块指令具体见第 4 章指令列表。

（5）常见连接错误说明。

1. “Send canid:3 set num:0 ack time out.” ，定位模块与控制器 CAN 通讯超时。检查网线是否连接，控制器与定位模块波特率是否一样。

2. “Cannot use can command for itself,please use modbus command”，定位模块与控制器 CAN ID 号重复。检查控制器与定位模块 ID 号，两者不能一样。

（6）以下为具体用例。

例　程：SMC606 与定位模块上通讯，波特率为 500K，当 IO 口 0 为高电平时，实现定位轴 0 的回原点，当 IO 口 2 为 1 时实现定长运动。

```
''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''
undim *          '清除变量
```

```
dim axis=0, outmode=0, result        '定义轴号 0，脉冲模式 0
Dim equiv=100,equiv1                 '定义脉冲当量
dim home_speed=1000'回原点速度 1000
dim org_logic=0, filter=0, result '设置回原点电平有效状态为 0
dim home_dir=1, vel_mode=1, mode=0, EZ_count=0'回原点模式参数
dim low_speed=2000，t_acc=0.1，t_dec=0.1
dim bitno1=1'IO 口
canid=3'定位模块的 ID 号，此值必须与硬件接口上配置值一样
"""""""""""""""""""""指令调用执行"""""""""""""""""""""
'第一步、设置控制器波特率和 CAN ID 号
setcan(500,1)   '设置控制器 CAN ID，波特率 500Kbps/S。若控制器为 SMC104  则 ID 号是通过拨码开关实现，波特率通过指令 setcan(boute)实现
'第二步、设置定位模块输出脉冲模式，脉冲当量等
SMCSetPulseOutModeCan(canid,axis,outmode,result)
SMCSetEquivCan(canid,axis,equiv,result)     '模块脉冲当量为 100
SMCGetEquivCan(canid,axis,equiv1,result)   '读取定位模块脉冲当量参数
print"脉冲当量", equiv1                          '打印定位模块脉冲当量值
'第三步、设置轴 0 的初始位置位置，读取 CAN IO 状态等
SMCSetPositionCan(canid, axis, 0, result)        '设置轴 0 的初始位置位置
dim bitno=0, state,result                        '定义 IO 口 0
SMCReadInbitCan( canid, bitno, state, result) '读取 IO 口 0 的电平状态
'第四步、判断 CAN IO 状态，进行回原点操作
If   state=1 then'若 IO 口 0 的电平状态为高
      SMCSetHomeProfileCan(canid,axis,low_speed,home_speed,t_acc,t_dec,result)
      SMCSetHomePinLogicCan( canid, axis, org_logic, filter, result)
      SMCSetHomeModeCan( canid, axis, home_dir, vel_mode, mode, EZ_count, result)
      '第五步、启动回原点运动
      SMCHomeMoveCan(canid, axis, result)
endif
'第六步、判断 IO 口 1 的电平状态，进行回定长运动
```

```
SMCReadInbitCan( canid, bitno, state, result) '读取 IO 口 1 的电平状态
If state=1then
    Dist=10000
    SMCPmoveCan(canid, axis, Dist, 1, result)        '启动定长运动
Endif
```

⚠注意：以上用例运行前，需设置定位模块的波特率为 500Kbps/S ，CAN ID 号为 3

2.7.2.3 模拟量模块

雷赛控制器具有模拟量模块功能，其有电流和电压两种模式。

如控制器连接模拟量模块，主要步骤如下：

(1) 设置模拟量模块波特率、CAN ID 号。

(2) 将控制器和模拟量模块通过网线连接好

(3) 打开 SMC Basic Studio 软件连接控制器，设置波特率、CAN ID 号。

(4) 在SMC Basic Studio软件中编辑调用模拟量指令等,如指令.SMCSetAinModeCan等

⚠注意：运行前需设置模拟量模块的波特率（可直接在模拟量模块波特率和 CAN ID 硬件接口上手动设置），控制器上的 CAN ID 号与模拟量模块的 ID 后不能相同。

例　程：控制器连接模拟量模块，输出电压。

```
''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''
dim canid,result,fvalue
dim channel,enable,filter,workmode
canid=1'AD 模块 ID 号
channel=0'AD 输出口 0
enable=1'AD 模块使能
workmode=0'AD 输出：电压
filter=10'滤波时间
''''''''''''''''''''''''''指令调用执行''''''''''''''''''''''''''
'第一步、设置 AD 模块波特率为 500kbps/S 和 CAN ID 为 1
'第一步、设置控制器波特率
setcan(500,3)   '设置主控制器波特率为 500Kbps/S，CAN ID 号为 3
```

```
'第三步、将主控制器与 AD 模块的 CAN 口通过网线相连
'第四步、设置 AD 输出电压使能等参数
SMCSetAinModeCan( canid, channel, enable, workmode, filter, result)
第五步、读取 AD 输出电压值
While 1
    SMCGetAinCan( canid, channel,fvalue,result)
    print"SMCGetAinCan",canid,channel,fvalue,result
    Delay 500
wend
```

⚠注意：上述程序运行完后，我们需手动给 AD 模块输入口 0 输入电压值。

2.8 BASIC 与 G 代码混合编程功能

BASIC 编辑开发软件 SMC BASIC STUDIO 里可以直接编辑 G 代码文件来运行程序。同时提供了 G 代码程序控制功能，可以下载 G 代码文件到控制器，通过指令来调用 G 代码文件来实现各运动功能。

相关指令：

名称	功能	参考
SMCGcodeSetCurrentFile	设置为当前加工文件	4.31.1 节
SMCGcodeStart	启动程序	
SMCGcodePause	暂停程序	
SMCGcodeStop	停止程序	
SMCGcodeState	读取运动状态	
SMCGcodeSetStepState	设置单步运行状态	
SMCGcodeGetStepState	读取单步运行状态	
SMCGcodeStopReason	读取停止原因	
GCode_LinerNO	设置 G 插补坐标系	
GCode_LocateSpeed	设置 G 指令定位运行速度	
GCode_LocateTAcc	设置 G 指令定位加减速时间	

GCode_PathSpeed	设置 G 指令插补运行速度	
GCode_PathTAcc	设置 G 指令插补加减速时间	
SMCGcodeCurretLine	读取当前行状态	4.31.2 节
SMCGcodeGetLine	读取行字符数组	
SMCGcodeGetLineArray	读取行数据数组	
SMCGcodeDeleteLine	删除一行 G 代码	
SMCGcodeAddLine	添加一行 G 代码字符串或字符数字	
SMCGcodeAddLineArray	添加一行 G 代码数据数组	
SMCGcodeModifyLine	修改一行 G 代码字符串或字符数组	
SMCGcodeModifyLineArray	修改一行 G 代码数据数组	
SMCGcodeInsertLine	插入 G 代码字符	
SMCGcodeInsertLineArray	插入 G 代码数组	
SMCGcodeCheckFile	检查 G 文件状态	4.31.3 节
SMCGcodeSetCurFile	设置 G 代码运行文件	
SMCGcodeSaveFile	保存 G 代码文件	
SMCGcodeCreatFile	创建 G 代码文件	
SMCGcodeDeleteFile	删除 G 代码文件	
SMCGcodeDeleteFileId	按文件号删除 G 代码文件	
SMCGcodeCopyFile	复制 G 代码文件	
SMCGcodeGetCurFile	读取当前 G 代码文件信息	
SMCGcodeGetFirstFile	读取第一个 G 代码文件	
SMCGcodeGetNextFile	读取下一个 G 代码文件	
SMCGcodeGetFile	设置 G 代码运行文件	

例程 1：在编辑器直接写 G 代码执行定长、插补运动

```
G00 X100 Y200'执行定长运动
G01 X200 Y100    '执行插补运动
```

例程 2：通过指令来调用 G 代码文件，来达到执行 G 代码运动功能。

```
''''''''''''''''''''''''''变量定义''''''''''''''''''''''''''
dim result
```

```
dim Gcode_State 'G 代码执行状态
dim Gcode_LastLineNo 'G 代码上次行号
dim Gcode_CurLineNo 'G 代码当前行号
dim Gcode_CurLineString(200) 'G 代码当前行字符串
dim Gcode_TotalLineNum 'G 代码总行数
""""""""""""""""""""指令调用执行""""""""""""""""""
'第一步、设置轴脉冲当量值
SMCSetEquiv(0,100) '设置 0 轴脉冲当量值
SMCSetEquiv(1,100) '设置 1 轴脉冲当量值
SMCSetEquiv(2,100) '设置 1 轴脉冲当量值
'第二步、设置 G 代码运行参数
GCode_LinerNO=0'选择坐标系 0
Gcode_AxisList(0)=0'X 轴对应 0 轴
Gcode_AxisList(1)=1'Y 轴对应 1 轴
Gcode_AxisList(2)=2'Z 轴对应 2 轴
GCode_LocateSpeed(Gcode_AxisList(0))=40'0 轴定位速度
GCode_LocateSpeed(Gcode_AxisList(1))=40'1 轴定位速度
GCode_LocateSpeed(Gcode_AxisList(2))=40'2 轴定位速度
GCode_LocateTacc(Gcode_AxisList(0))=0.1'0 轴定位加减速时间
GCode_LocateTacc(Gcode_AxisList(1))=0.1'1 轴定位加减速时间
GCode_LocateTacc(Gcode_AxisList(2))=0.1'2 轴定位加减速时间
GCode_PathSpeed(GCode_LinerNO)=50'插补速度
GCode_PathTAcc(GCode_LinerNO)=0.1'插补加减速时间
'第三步、设置加工文件
 result=SMCGCodeSetCurFile("3.g")      '确保控制器保存有 G 代码文件"3.g"
'第四步、启动 G 代码程序
 result=SMCGCodeStart()
Print"G 代码程序启动"
 Gcode_LastLineNo = -1
While 1
```

```
        SMCGcodeState(Gcode_State)'读取 G 代码运行状态
        SMCGcodeCurrentLine(Gcode_CurLineNo,Gcode_TotalLineNum,Gcode_CurLineString(0))'读取 G 代码当前执行行
        If  Gcode_LastLineNo<>Gcode_CurLineNo then
            Gcode_LastLineNo = Gcode_CurLineNo
            Print"状态:"Gcode_State,"行数:"Gcode_TotalLineNum,"行号:"Gcode_CurLineNo,"行字符串:"str(Gcode_CurLineString(0))
        endif
        delay(100)
    wend
```

运行结果：以设定插补速度、定位速度等参数运行加工 G 代码文件 sibx.g

⚠ 注意：例程 2 运行程序前，需确认已将需加工 G 代码文件“sibx.g”下载到了控制器。G 代码文件的创建可直接在编辑器中写入 G 代码，写好代码后保存名为“sibx.g”。“sibx.g”文件中代码可为 G00、GO1 等任意运行轨迹 G 代码。

2.9 总线控制功能

雷赛控制器支持总线功能，包括 CANopen 总线、EtherCAT 总线。总线控制和脉冲控制大部分的用法一样，但是存在一些区别，具体从以下几个部分来详细说明。

2.9.1 电机使能

在总线模式下，所有轴在运动之前都需要进行使能操作，不管是总线伺服还是总线步进。和脉冲控制器对电机使能即打开伺服使能信号不同，总控制器的轴使能，需要发送设置轴使能指令，并且在总线轴状态机（state_machine）变成”操作使能”状态后才完成。以下是具体代码部分：

注意：为更好说明该功能，该部分代码使用 BASIC 语言为例，仅供参考，请根据实际情况使用。

```
Dim Myaxis,
Dim TotalAxis,cnts,state_machine
Myaxis =0       '轴号
TotalAxis=0                          '当前总线上的连接轴数
cnts=0
delay(100)
if BusState=0 then                          '总线正常状态下使能操作
        NMCSGetTotalAxes(TotalAxis)          '获取当前总线上连接的轴数
        if TotalAxis > 0 then                ‘连接轴号大于 0 才能操作
            for cnts=0 To TotalAxis-1        '按轴号每轴使能
                NMCSSetAxisEnable(cnts)    '设置指定轴使能
                NMCSGetAxisStateMachine(cnts,state_machine)    '获取轴状态机
                ticks=3000                   '3 秒未使能完成，提示出错
                while (state_machine<>4)
                    NMCSSetAxisEnable(cnts)
                    NMCSGetAxisStateMachine(cnts,state_machine)
                    if   ticks<0   and   state_machine<>4   then
                        print "使能失败！请重试"
                        Exit for
                    endif
                wend
            next cnts
         endif
    else
        print "总线异常，无法进行使能操作！"
    endif
end
```

2.9.2 电机复位

脉冲控制器的复位方式和总线控制器的复位方式有比较大的区别，脉冲控制器对复位的控制（读取原点限位信号，控制方式等）都在控制器部分，总线控制器对复位的控制只有发起复位和等待复位结束，中间的复位过程全部交由驱动器来处理。雷赛控制器支持标准的 34 种回零模式，具体请参考 EtherCAT 回零标准。具体由以下的示例说明。

注意：为更好说明该功能，该部分代码使用 BASIC 语言为例，仅供参考，请根据实际情况

使用。

```
Dim Myaxis,Mode,state,result,BusState,axistemp,tempstate,Axis_StateMachine
Dim Low_Vel,Offset,Home_Tacc,Home_Tdcc,High_Vel,home_mode
home_mode=21
Myaxis = 0                              '轴号
Mode=21                                 '回原点模式
Low_Vel=1000                            '回原点低速
High_Vel =20000                         '回原点高速
Offset=0                                '回原点完成后设置位置值
Home_Tacc =0.1                          '回原点加减速时间
Home_Tdcc =0.1                          '回原点加减速时间
run 2,reflashstate
delay(100)

if BusState=0 then                      '总线正常状态下使能操作
    NMCSGetTotalAxes(TotalAxis)         '获取当前总线上连接的轴数
    if TotalAxis > 0 then
        for cnts=0 To TotalAxis-1       '按轴号每轴使能
            NMCSSetAxisEnable(cnts)   '设置指定轴使能
            NMCSGetAxisStateMachine(cnts,Axis_StateMachine)   '获取轴状态机
            ticks=3000                  '3 秒未使能完成，提示出错
            while (Axis_StateMachine<>4)
                NMCSSetAxisEnable(cnts)
                NMCSGetAxisStateMachine(cnts,Axis_StateMachine)
                if ticks<0 then
                    print "使能失败！请重试"
                    Exit for
                endif
            wend
        next cnts
    endif
else
    print "总线异常，无法进行使能操作！"
endif
```

```
if BusState=0 then"总线正常
    NMCSGetAxisStateMachine(Myaxis,Axis_StateMachine)   '获取轴状态机
    if Axis_StateMachine=4 then"轴处于使能可操作状态
        '设置轴 0 回原点模式、速度等参数
        result=SMCSetHomeMode(Myaxis,0,0,home_mode,0)
        result=SMCSetHomeProfileUnit(Myaxis,Low_Vel,High_Vel,Home_Tacc,0)
        result=SMCHomeMove(Myaxis)
        '读取并打印回原点状态
        While SMCCheckDone(0)=1                          '等待运动停止
            result=SMCGetHomeResult(Myaxis ,state)     '读取回原点运动状态
            Print state                                  '打印状态值，0 未完成，1 完成
        wend
    endif
endif

reflashstate:"读取总线状态和轴状态机
while 1
    NMCSGetErrcode(2,BusState)
wend
```

2.9.3 IO 控制及电机运动

总线控制器和脉冲控制器在 IO 控制以及轴运动方式上并无实质上的区别，控制函数两者都是一样的，以下以定长以及连续运动、位置清零、减速停止为例做说明。

```
//执行定长运动以及连续运动
Dim MyCardNo,Myaxis,Mys_mode,Myposi_mode
Dim Mys_para,MyDist,Axis_StateMachine,tempstate,BusState
Dim TotalAxis,cnts
Myaxis =0       '轴号
result=SMCSetEquiv(Myaxis,100)   '设置脉冲当量为  100pulse/unit
MyMin_Vel = 100                   '设置起始速度为 100unit/s
MyMax_Vel = 1000                  '设置最大速度为 1000unit/s
MyTacc = 0.1                     '设置加速时间为 0.1s
MyTdec = 0.1                     '设置减速时间为 0.1s
MyStop_Vel = 200                 '设置停止速度为 200unit/s
```

```
Mys_mode = 0                    '保留参数
Mys_para =0                     'S 段时间为 0s 即速度曲线不平滑,T 形曲线
MyDist = 1000                   '设置运动距离为 1000unit
Myposi_mode = 0                 '设置运动模式为相对坐标模式
TotalAxis=0                     '当前总线上的连接轴数
cnts=0

run 2,reflashstate
delay(100)
    if BusState=0 then                  '总线正常状态下使能操作
        NMCSGetTotalAxes(TotalAxis)     '获取当前总线上连接的轴数
        if TotalAxis > 0 then
            for cnts=0 To TotalAxis-1   '按轴号每轴使能
                NMCSSetAxisEnable(cnts)   '设置指定轴使能
                NMCSGetAxisStateMachine(cnts,Axis_StateMachine)   '获取轴状态机
                ticks=3000                '3 秒未使能完成，提示出错
                while (Axis_StateMachine<>4)
                    NMCSSetAxisEnable(cnts)
                    NMCSGetAxisStateMachine(cnts,Axis_StateMachine)
                    if ticks<0 and Axis_StateMachine<>4 then
                        print "使能失败！请重试"
                        Exit for
                    endif
                wend
            next cnts
        endif
    else
        print "总线异常，无法进行使能操作！"
    Endif

 if BusState=0 then                 '总线正常
    if Axis_StateMachine=4   then   '轴处于使能可操作状态
                                    '第一步、设置 0 号轴速度曲线参数
        result=SMCSetProfileUnit(Myaxis,MyMin_Vel,MyMax_Vel,MyTacc,MyTdec,MyStop_Vel)
```

```
        result=SMCSetsprofile(Myaxis,Mys_mode,Mys_para)   '第二步、设置 0 号轴 S 段参数
        result=SMCPMoveUnit(Myaxis,MyDist,Myposi_mode)   '第三步、启动 0 号轴梯形定长运动
        while(SMCCheckDone(Myaxis)=0)   '第四步、等待轴运行停止
        wend
    endif
else                                    '总线异常处理
    result=SMCEmgStop()                 '紧急停止所有轴
    print "总线异常！ "
endif

end

reflashstate:                           '读取总线状态
while 1
    NMCSGetErrcode(2,BusState)          ‘读取总线状态
wend

//执行清除指令位置，脉冲方式和总线方式一致
result=SMCSetPositionUnit(0,0)

//执行减速停止
while1
    if   SMCReadInBit(2)=0 then                 '如果输入口 2 为低电平，
            SMCSetDecStopTime(0,0.15)           '设定减速停止速度时间
            SMCStop(0,0)                        '0 轴电机减速停止
            print"stop"
    endif
wend
```

2.9.4 总线状态

总线控制器由于是主从结构，并且主从结构之间通过网线连接，所以需要在程序中实时扫描总线状态(一般采用定时器或者独立的任务来扫描)，以应对可能出现的异常。当总线状态正常时才能进行相应的操作，当总线状态异常时需要做相应的处理，如停止当前运行、提示报警等。如下例程所示：

//扫描总线状态

总线控制器在使用过程中可能会遇到以下错误信息，有部分错误信息需要给控制器重新上电初始化总线才能消除，有些错误信息可以通过函数操作消除。

序号	总线错误	原因及解决方法
1	000e	总线初始化，若正常会自动消失，若一直报错，则需要检查线路。
2	001e	增加或者丢失从站；需要重新扫描
3	0009	控制器与第一个从站丢失连接需要重新连接
4		

//清除总线错误操作

```
while1
    if   SMCReadInBit(2)=0 then          '如果输入口 2 为低电平，
            NMCSClearErrCode()           ‘清除总线错误
            print"stop"
    endif
wend
```

第 3 章 G 代码指令列表

G 指令主要是指数控加工代码，雷赛控制系统的数控程序采用的数控代码符合 ISO-1056-1975E 标准，G 代码、M 代码分别表示准备功能字和辅助功能字。

为了提高 G 代码的灵活性，我们增加了一些特殊 G、M 代码，如：条件判断、循环控制、变量控制、子程序、多任务程序等等。合理地使用这些功能，可以编写出功能强大的运动控制程序。

3.1 坐标系介绍

系统按照以右手法则确定的笛卡儿直角坐标系（the Right-handed Rectanglar Cartesian Coordinate System）作为编程的标准坐标系，如下图 3.1。

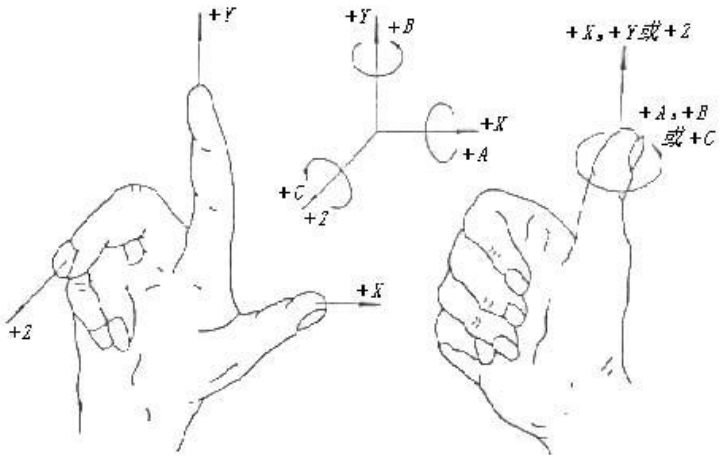


图3.1　右手法则

系统坐标系是系统内部唯一的坐标系，一般与设备机架固连，其坐标系原点通常与设备的机械原点开关位置对应。

绝对坐标和相对坐标：

运动模式分为绝对运动和相对运动两种模式，如图 3-2 所示。所谓相对运动，就是用一系列点来定义一条曲线，改变其中某一点的坐标将会影响其后续点的坐标。而绝对运动则是用一系列点来定义一条曲线，改变其中某一点的坐标不会影响其后续点的坐标。

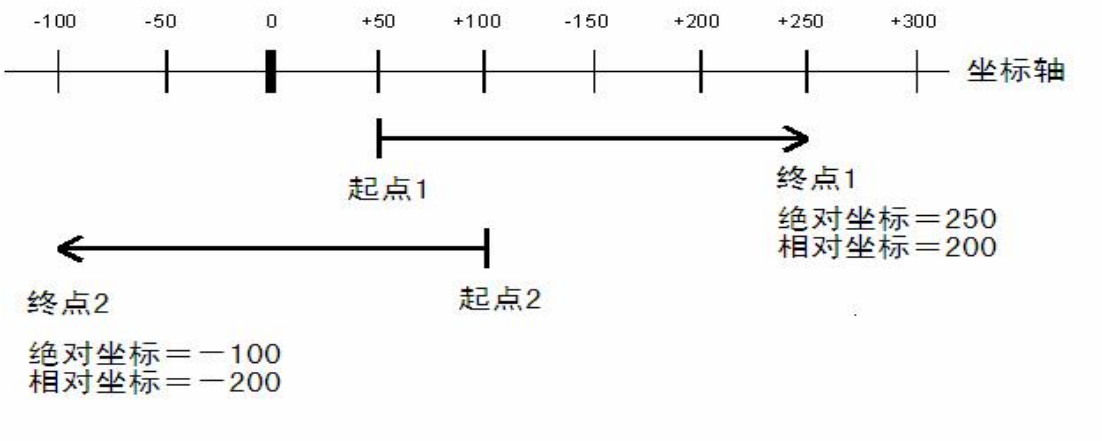


图 3.2 绝对运动和相对运动模式图

3.2 程序段格式

所谓程序段格式，是指程序段书写规则，它包括机床所要求执行的功能和运动所需要的所有几何数据和工艺数据。一个零件加工程序是由若干以段号大小次序排列的程序段组成，每个程序段一般由顺序号（N）、准备功能（G）、坐标字（X，Y，Z，U，V，W，I，J，K，R）、进给速度（F）、主轴功能（S）、刀具功能（T）、辅助功能（M）等组成。每个程序段不一定都必须具有上面这些指令，但在每个程序段中，指令要遵照上述格式来排列。

一个程序段由一个或多个程序字组成，程序字通常由地址和地址字后的数据组成。

例如：X-46.38

其中：X——地址功能字

-46.38——数据字

雷赛数控系统采用的程序段格式是可变程序段格式，所谓可变程序段格式就是程序段的数据的个数和长度都是可变的。这一格式，如上一程序段已写明，本程序段里不产生变化的那些字仍然有效，可以不再重写。数据字中，可只写有效数字，不规定每个字要写满固定位数。

例如：N100 G03 X70 Y-36.5 I0 J-2　F100

程序中 N，G，X，Y，I，J，F 均为地址功能字。100，03，70，-36.5，0，-2，100 均为数据字。数据字尺寸字地址可用 X，Y，Z，U，V，W，I，J，K 等字母表示。非尺寸地址用 N，S，T，G，F，M，P 等字母表示。

在通常情况下，程序段是零件加工的一个工步，NC 程序段是一个程序语句系列，程序语句作为程序贮存在存储器里。加工零件时，这些语句从存储器整体读出并一次性解释成可执行的数据格式，然后加以执行。

3.3 编程方式

G 代码编程可在编程开发测试工具 SMC BASIC STUDIO 上实现，具体过程如下：

（1） 软件连接控制器，新建 G 代码文件（**.g），并编辑代码；

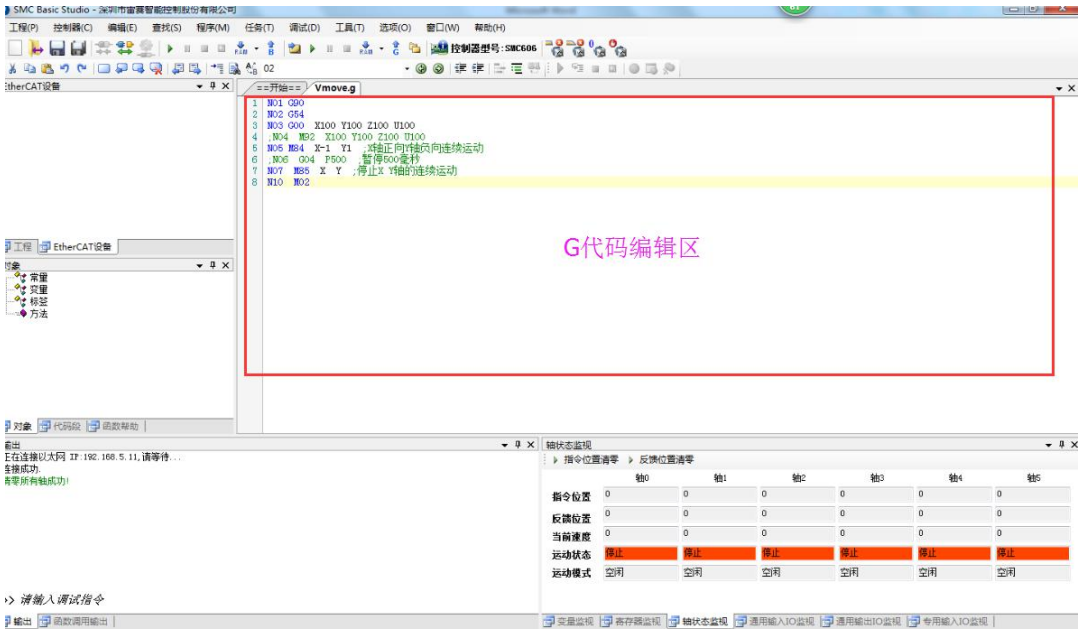


图 3.3 G 代码软件编辑示意图

（2） 下载 G 代码到控制器

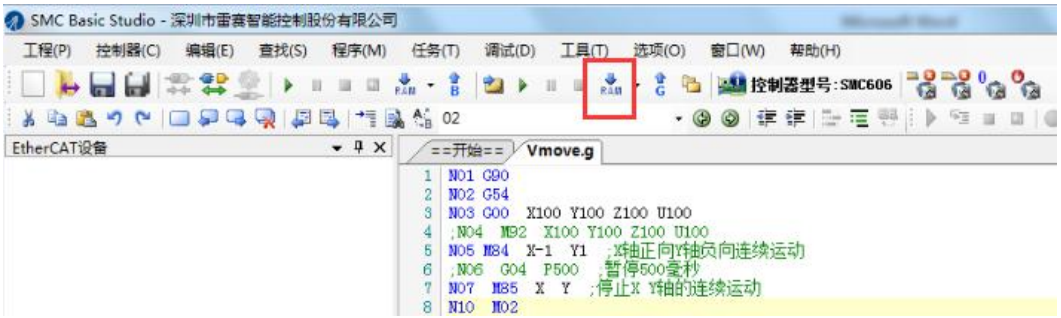


图 3.4 G 代码程序下载示意图

（3） 运行 G 代码。

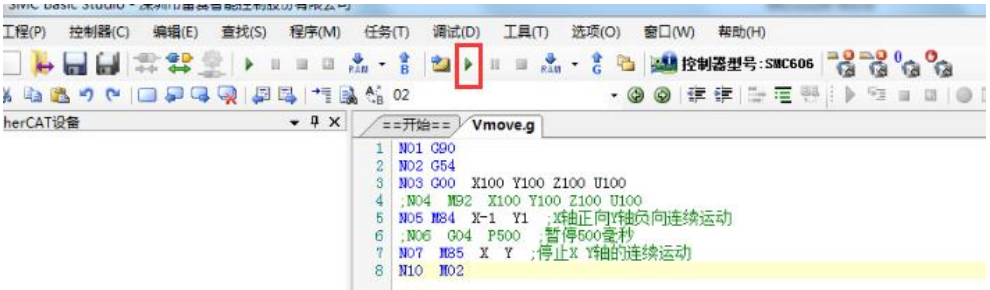


图 3.5 G 代码软件运行示意图

3.4 指令详细介绍

3.4.1 坐标系指令

G90 绝对坐标

参数选择：无参数

功能： 绝对坐标，设置后面的坐标都为绝对坐标

参数： 无

⚠注意：采用本指令后，后续程序段的坐标值都应按绝对方式编程，即所有点的表示数值都是在编程坐标系中的坐标值，直到执行G91为止。

示例：N00G90;设置后面的坐标都为绝对坐标

G91 相对坐标

参数选择：无参数

功能： 相对坐标，设置后面的坐标都为相对坐标

参数： 无

⚠注意：采用本指令后，后续程序段的坐标值都按相对方式编程，即所有点的坐标均以前一个坐标值作为起点来计算运动的位置矢量，直到执行G90指令为止。

示例：N00G91;设置后面的坐标都为相对坐标

G92 重定义坐标

参数选择：X Y Z U V W

功能： 用于重新定义坐标

参数：X 可选。设定X轴的当前坐标位置，单位：unit。

Y 可选。设定Y轴的当前坐标位置，单位：unit。

Z 可选。设定Z轴的当前坐标位置，单位：unit。

U 可选。设定U轴的当前坐标位置，单位：unit。

V 可选。设定V轴的当前坐标位置，单位：unit。

W 可选。设定W轴的当前坐标位置，单位：unit。

⚠注意：使用本指令后，工件机械坐标被修改为此设置值。

示例：G92 X100 Y500 Z1000 ;将当前坐标定义为 X100、Y500、Z1000

G53 机械坐标

参数选择：无参数

功能：切换为机械坐标

参数：无

⚠注意：采用本指令后，程序的坐标系原点切换为机械原点。

示例：N00G53;转换为机械坐标

G54 还原为工件坐标

参数选择：无参数

功能：还原为工件坐标

参数：无

⚠注意：程序坐标系设置指令。一般以工件原点作为程序的坐标原点。本指令指定程序的坐标原点为工件原点。

示例：N00G54;还原为工件坐标

M92 强制修改坐标

参数选择：X Y Z U V W

功能： 强制修改指定轴的工件坐标

参数： X 可选。指定X轴从当前位置移动到的位置，单位：unit。

Y 可选。指定Y轴从当前位置移动到的位置，单位：unit。

Z 可选。指定Z轴从当前位置移动到的位置，单位：unit。

U 可选。指定U轴从当前位置移动到的位置，单位：unit。

V 可选。指定V轴从当前位置移动到的位置，单位：unit。

W 可选。指定W轴从当前位置移动到的位置，单位：unit。

⚠注意：使用本指令后，当前点的工件坐标值被修改为此设置值，工件原点会发生相应变化。

示例：N10M92 X0 ;强制修改 0 为当前 X 工件坐标。

3.4.2 运动指令

G00 快速定位

参数选择：X Y Z U V W F

功能： 快速定位

参数： X 可选。指定X轴的移动位置或距离，单位：unit。

Y 可选。指定Y轴的移动位置或距离，单位：unit。

Z 可选。指定Z轴的移动位置或距离，单位：unit。

U 可选。指定U轴的移动位置或距离，单位：unit。

V 可选。指定V轴的移动位置或距离，单位：unit。

W 可选。指定W轴的移动位置或距离，单位：unit。

F 可选。速度参数，单位：百分比

⚠注意：轴以最快速度沿直线或折线（GCODE_IFGOOUSELINE参数设定）移动到目标位置。不运动的坐标可以省略不写，不参与运动。

示例：N00G00 X100 Y100 Z100 U100

G01 直线轨迹

参数选择：X Y Z U V W F

功能： 直线插补

参数： X 可选。指定X轴的移动位置或距离，单位：unit。

Y 可选。指定Y轴的移动位置或距离，单位：unit。

Z 可选。指定Z轴的移动位置或距离，单位：unit。

U 可选。指定U轴的移动位置或距离，单位：unit。

V 可选。指定V轴的移动位置或距离，单位：unit。

W 可选。指定W轴的移动位置或距离，单位：unit。

F 可选。速度参数，单位：百分比

示例：N00G01 X100 Y100

G02 顺圆

参数选择：X Y Z U V W I J K R C F

功能： 圆心或者半径模式顺圆插补

参数： X 可选。指定X轴的移动位置或相对距离，单位：unit。

Y 可选。指定Y轴的移动位置或相对距离，单位：unit。

Z 可选。指定Z轴的移动位置或相对距离，单位：unit。

U 可选。指定U轴的移动位置或相对距离，单位：unit。

V 可选。指定V轴的移动位置或相对距离，单位：unit。

W 可选。指定W轴的移动位置或相对距离，单位：unit。

I 可选。指定X轴的圆心的位置或相对距离，单位：unit。

J 可选。指定Y轴的圆心的位置或相对距离，单位：unit。

K 可选。指定Z轴的圆心的位置或相对距离，单位：unit。

R 半径圆必选。圆弧半径，单位：unit。

C 可选。圆弧圈数，正整圈数。

F 可选。速度参数，单位：百分比。

⚠注意1：XYZ必须且只能选择两个做圆弧插补，如果需要做螺旋，则选择三个轴，前两个轴做圆弧插补，第三个轴走直线。

⚠注意2：当有R参数时将会走半径圆，否则走圆心圆，当R为负时，表示大于180度的圆；为正时，小于180度的圆。

示例：N00G02 X100 Y0 R50 F50 ;以高速的 50%作一半径为 50 的顺时针圆弧

G03 逆圆

参数选择：

半径模式：X Y Z U V W R F

圆心模式：X Y Z U V W I J F

功能：圆心或者半径模式逆圆插补

参数： X 可选。指定X轴的移动位置或相对距离，单位：unit。

Y 可选。指定Y轴的移动位置或相对距离，单位：unit。

Z 可选。指定Z轴的移动位置或相对距离，单位：unit。

U 可选。指定U轴的移动位置或相对距离，单位：unit。

V 可选。指定V轴的移动位置或相对距离，单位：unit。

W 可选。指定W轴的移动位置或相对距离，单位：unit。

I 可选。指定X轴的圆心的位置或相对距离，单位：unit。

J 可选。指定Y轴的圆心的位置或相对距离，单位：unit。

K 可选。指定Z轴的圆心的位置或相对距离，单位：unit。

R 半径圆必选。圆弧半径，单位：unit

C 可选。圆弧圈数，正整圈数。

F 可选。速度参数，单位：百分比

⚠注意：同G02

示例：N00G03 X0 Y-100 R50 F50 ;以高速的 50%作一半径为 50 的逆时针圆弧

G04 延时

参数选择：P

功能：延时

参数： P 必选。 暂停时间。单位：毫秒

⚠注意：最长可延时9.999秒。

示例：N00G04 P500 ;暂停 500 毫秒

G05 三点圆弧

参数选择：X Y Z U V W I J K C F

功能：三点模式圆弧插补

参数： X 可选。指定X轴的移动位置或相对距离，单位：unit。

Y 可选。指定Y轴的移动位置或相对距离，单位：unit。

Z 可选。指定Z轴的移动位置或相对距离，单位：unit。

U 可选。指定U轴的移动位置或相对距离，单位：unit。

V 可选。指定V轴的移动位置或相对距离，单位：unit。

W 可选。指定W轴的移动位置或相对距离，单位：unit。

I 可选。指定X轴的经过点的位置或相对距离，单位：unit。

J 可选。指定Y轴的经过点的位置或相对距离，单位：unit。

K 可选。指定Z轴的经过点的位置或相对距离，单位：unit。

C 可选。圆弧圈数，正整圈数。

F 可选。速度参数，单位：百分比

示例：N00G05 X100 Y100 Z100 I50 J50 K70 F50 ;以高速的 50%作一经过（50，50，70）的圆弧。

G06 三点螺旋

参数选择：X Y Z U V W I J K C F

功能：三点模式螺旋插补

参数： X 可选。指定X轴的移动位置或相对距离，单位：unit。

Y 可选。指定Y轴的移动位置或相对距离，单位：unit。

Z 可选。指定Z轴的移动位置或相对距离，单位：unit。

U 可选。指定U轴的移动位置或相对距离，单位：unit。

V 可选。指定V轴的移动位置或相对距离，单位：unit。

W 可选。指定W轴的移动位置或相对距离，单位：unit。

I 可选。指定X轴的经过点的位置或相对距离，单位：unit。

J 可选。指定Y轴的经过点的位置或相对距离，单位：unit。

K 可选。指定Z轴的经过点的位置或相对距离，单位：unit。

C 可选。螺旋圈数，正整圈数。

F 可选。速度参数，单位：百分比

⚠注意：XYZ必须且只能选择两个做圆弧插补，其他轴走直线

示例：N00G06 X0 Y-100 Z100 I50 J-50 C2 F50 ;以高速的 50%作一半径为 50 的螺旋线，运动圈数为 2 圈。

G26 回原点

参数选择：X Y Z U V W

功能：回机械零原点

参数： X Y Z U V W

示例：N00G26 X Y Z U ;X Y Z U 轴回原点

G28 回工件零点

参数选择：X Y Z U V W

功能：回工件零点

参数： X Y Z U V W

示例：N00G28 X Y Z U ;X Y Z U 轴回工件零点

M84 恒速运动

参数选择：X Y Z U V W

功能：vmove，恒速运动指令

参数：

X 可选。X轴转动方向，正是按正向转动，非正则按负向转动

Y 可选。Y轴转动方向，正是按正向转动，非正则按负向转动

Z 可选。Z轴转动方向，正是按正向转动，非正则按负向转动

U 可选。U轴转动方向，正是按正向转动，非正则按负向转动

V 可选。V轴转动方向，正是按正向转动，非正则按负向转动

W 可选。W轴转动方向，正是按正向转动，非正则按负向转动

示例：N00M84 X1 Y -1 ;X 轴正向 Y 轴负向恒速运动

M85 停止恒速运动

参数选择：X Y Z U V W

功能：停止vmove

参数：X Y Z U V W

示例：N00M85 X Y ;停止 X Y 轴的恒速运动

3.4.3 输入指令

M82 等待输入口有效

参数选择：S

功能：IO等待开

参数：S 必选。输入口号,0-23。

示例：N00M82 S5 ;等待输入口 5 有效

M83 等待输入口无效

参数选择：S

功能：IO等待关

参数：S 必选。输入口号

示例：N00M83 S5 ;等待输入口 5 无效

3.4.4 输出指令

M07 输出口0开

参数选择：无

功能：输出口0开

参数：无

示例：N00M07

M08 输出口0关

参数选择：无

功能：输出口0关

参数：无

示例：N00M08

M09 输出口1开

参数选择：无

功能：输出口1开

参数：无

示例：N00M09;开气阀

M10 输出口1关

参数选择：无

功能：输出口1关

参数：无

示例：N00M10;关气阀

M11 输出口2开

参数选择：无

功能：　输出口2开

参数：　无

示例：N00M11;开光闸

M12 输出口2关

参数选择：无

功能：输出口2关

参数：无

示例：N00M12;关光闸

M80 输出口开

参数选择：S

功能：IO开，执行该指令指定输出口立即开

参数：S　必选。输出口号

示例：N00M80　S5　;指定输出口 5 开

M81 输出口关

参数选择：S

功能：IO关，执行该指令指定输出口立即关

参数：S　必选。输出口号

示例：N00M81　S5;指定输出口 5 关

3.4.5 系统指令

F 速度设置

参数选择：F

功能：F指令，设置速度

参数：F　必选。速度百分比

示例：F　F52　;设定速度为高速的 52%

M86 变量加一个数

参数选择：S　V

功能：对应变量加一个数

参数：　S 变量编号，意义如下：

0-17 输出口

41-42 PWM占空比的修改，范围0-1

61-62 指示灯状态切换

71-72 PWM频率修改

101-200 内部变量

V IO，0不变，1翻转；变量具体值。

示例：假定输出口 5 初始为高电平

N00M86 S5 V1　　; 输出口 5 变量加 1 变为低电平

说明：上述程序被再次执行时，输出口 5 将变为高电平；若为 M86 S5 V0 时，执行此程序时输出口 5 初始电平无变化

M87 变量赋值

参数选择：S　V

功能：对应变量设置为一个值

参数： S 变量编号，意义如下：

0-17 输出口

41-42 PWM占空比的修改，范围0-1

61-62 指示灯状态输出，0-灭，1-亮

71-72 PWM频率修改

101-200 内部变量

V IO，0不变，1翻转；变量具体值。

示例：N00M87 S5 V1 ；输出口 5 有效

说明：上述程序执行后，输出口 5 有效；若为 M86 S5 V0 时，执行此程序时输出口 5 无效。执行此程序可通过故障诊断来观察输出口 5 的变化。

M00 程序暂停

参数选择：无

功能： 程序暂停

参数： 无

示例：N00M00;程序暂停

M02 程序结束

参数选择： 无

功能： 程序结束

参数： 无

示例：N00M02;程序全部结束

M30 程序结束并循环

参数选择：无

功能： 程序结束并循环

参数： 无

⚠注意：执行此指令后，程序即进入无限循环，可通过急停按钮将程序强制停止

示例：N00M30;程序结束并循环

M96 根据条件调用子程序

参数选择：Sm Vn Ni

功能：根据条件调用指定行号的子程序

参数：Sm，m：1-23 输入IO； 101-200 内部变量

Vn，n：IO，0无效，1有效；变量具体值。

Ni，i：跳转行号。

⚠注意：子程序的最后要使用代码M99。

示例：N10M96 S5 V1 N100;当输入口 5 有效时调用行号为 100 的子程序

M98 子程序调用

参数选择： N

功能： 子程序调用

参数： N 子程序首行的行号

⚠注意：使用此指令后，在子程序的末尾要使用M99指令

示例：N00M98N20;调用首行号为 20 的子程序

M99 子程序返回

参数选择： 无

功能：子程序返回

参数：无

示例：N100M99;子程序返回

3.4.6 逻辑指令

M90 局部循环结束

参数选择： 无

功能：局部循环结束

参数：无

示例：N00M90;局部循环结束

M91 局部循环开始

参数选择：C

功能：局部循环开始

参数：C， 循环次数

示例：N00M91 C10 ;局部循环 10 次

M94 根据条件跳转

参数选择：Sm Vn Ni

功能：根据条件跳到指定的行号

参数：Sm，m：0-23 输入IO； 101-200 内部变量Sys_Var_00~Sys_Var_99

Vn，n：IO，0-低电平，1-高电平；变量具体值。

Ni，i：跳转行号。

示例：N10M94 S3 V1 N50;当输入口 3 有效时跳转到行号为 50 的程序中去

M95 强制跳转

参数选择：Nm

功能：强制跳转到指定的行号

参数：Nm，m：跳转行号。

示例：N10M95N100;强制跳转到行号为 100 的程序中去

3.5 G 代码例程

3.5.1 直线插补例程

下面的例子从（0，0）直线焊接到（100，100）的位置。

```
N00G28  X Y                ; 回工件零点
N01G91; 程序后面的 G 代码参数都使用相对坐标
N02M07; 开光
N03G01 X100 Y100 F50    ; 以高速的 50%进行直线插补
N04M08; 关光
N05M02; 结束
```

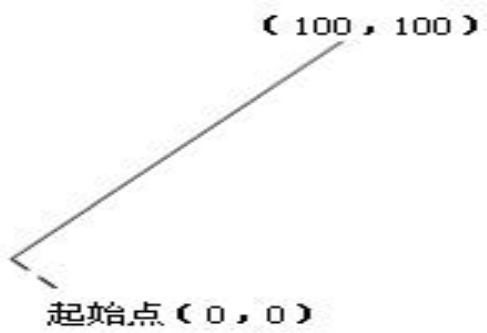


图 3.6 直线插补运行轨迹图

3.5.2 圆弧插补例程

下面的例子从（0，0）圆弧焊接到（100，100）的位置。

```
N00G28   X Y                    ; 回工件零点
N02M07; 开光
N03G02 X100 Y100 R100   ; 顺时针圆弧插补，
N04M08; 关光
N05M02; 结束
```

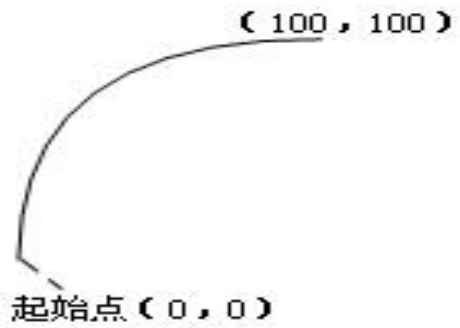


图 3.7 圆弧插补运行轨迹图

3.5.3 连续轨迹例程

下面工件只需事先移动到起始点，然后执行程序就能够走需要的轨迹。

```
N00G92 X0 Y0          ; 将当前的位置坐标定义为（0，0）
N01M07; 开激光
N02G01 X100
N03G01 Y100
N04G01 X0
N05G01 Y0
N06M08; 关激光
N07M02; 结束
```

⚠注意：连续轨迹通过连续的运行 G01,G02,G03,G05,G06 等轨迹指令来实现，如果需要调整速度加减速时间等可以通过 G 代码参数来调整，参数具体的设置可参见本手册第 4 章系统默认初始值小节的内容。

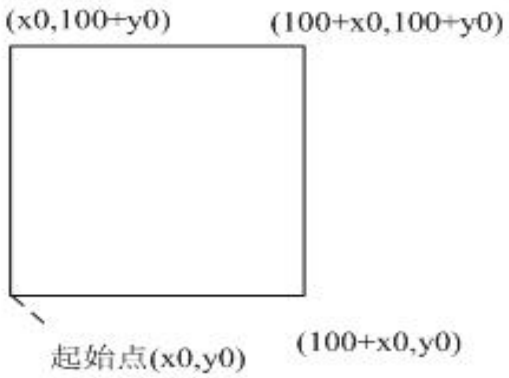


图 3.8 连续插补运行轨迹图

第 4 章 BASIC 指令列表

4.1 基本指令

雷赛全系列控制器均支持一些基本的运算指令、逻辑指令等。

4.1.1 运算指令

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。

算术运算符和逻辑运算符的优先级如下表所示。

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
求幂	^	等于	=	逻辑非	Not
负号	-	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	Mod (%)	大于等于	>=		
加	+				
减	—				

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。

各指令详细说明见下：

AND

语　法：expression1 AND expression2

描　述：将两个表达式的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，以2进制来比较。

例　程：对 a 值与 b 值进行“与”运算

```
dim a,b,c
a=11' a=10 = 1011B
b=13' b=12 = 1101B
c=a andb ' c =1001B＝9
Printc'  打印值 c，打印值为十进制
```

运行打印结果：9

OR

语　法：expression1 OR expression2

描　述：将两个表达式的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，以2进制来运算。

例　程：对 a 值与 b 值进行“或”运算

```
dim a,b,c
a=11' a=10 = 1011B
b=13' b=12 = 1101B
c=a orb   'c = 1111B＝15
Printc '  打印值 c，打印值为十进制
```

运行打印结果：15

NOT

语　法：NOT expression

描　述：对表达式的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

例　程：将 a 值的非值转化给 b 值

```
dim a,b
a=0' a = 0 =00000000B（以一个字节长度为例）
b=not a   ' b =11111111B，'计算机中存储数据是以补码的形式存储的，
print b 'b 为有符号数，故 b 的补码为 10000001B，转换成十进制数为-1
```

运行打印结果：-1

XOR

语　法：expression1 XOR expression2

描　述：将两个表达式的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

例　程：将 a 值与 b 值的异或值付给 c 值

```
dim a,b,c
a=11' a=10 = 1011B
b=13' b=12 = 1101B
c=a   xor   b ' c = 0110B  ＝  6
Print c
```

运行打印结果：6

EQV

语　法：expression1 EQV expression2

描　述：将两个表达式的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

例　程：将 a 值与 b 值的同或值付给 c 值

```
dim a,b,c
a=10' a=10 = 00001010B
b=12' b=12 = 00001100B
```

```
c=a  eqv  b     ' c =11111001B   c 为有符号数，故 c 的补码为 10000111B，转换成十进制数为-7
Print c
```

运行打印结果：-7

ABS

语　法：ABS(expression)

描　述：求表达式的绝对值

参　数：expression　　表达式

例　程：将 a 值的绝对值赋给 a1，b 值的绝对值赋给 b1

```
dim a,b,a1,b1
a=10
b=12
a1= abs(a)      '取 a 的绝对值
b1= abs(b)      '取 b 的绝对值
Print a1,b1   '打印 a1、b1 值
```

运行打印结果：10 12

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

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

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

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

ATAN

语　法：ATAN(expression)

描　述：计算表达式的反正切函数，返回值单位：弧度

参　数：expression　　表达式

SQR

语　法：SQR(expression)

描　述：计算表达式的平方根

参　数：expression　　表达式

LN

语　法：LN(expression)

描　述：计算表达式的自然对数

参　数：expression　　表达式

LOG

语　法：LOG(expression)

描　述：计算表达式以10为底的对数

参　数：expression　　表达式

CLEAR_BIT

语　法：CLEAR_BIT(bit，int)

描　述：将操作数的二进制的第bit位清0

参　数：bit　　位编号：0~31

　　int　　操作数

⚠注意：只针对操作数的整数部分操作

例　程：将 a 的二进制数第 2 位清 0 后，将值赋给 b 值

```
dim a,b
a=15'a = 01111B
b=clear_bit(2,a)  ' 将 a 的二进制数第 2 位清 0
Print b              ' b = 01011B = 11
```

运行结果： b = 11

READ_BIT

语　法：READ_BIT(bit，int)

描　述：读取操作数的二进制的第bit位的值

参　数：bit　　位编号：0~31

　　int　　操作数

⚠注意：只针对操作数的整数部分操作

例　程：读取 a 的二进制数的第 1 位，并将值赋给 b 值

```
dim a,b
a=13'a = 01101B
b=read_bit(1,a) ' 将 a 的二进制数第 1 位数赋值给 b
Print b
```

运行结果： b = 0

SET_BIT

语　法：SET_BIT(bit，int)

描　述：将操作数的二进制的第bit位置1

参　数：bit　　位编号：0~31

　　int　　操作数

⚠注意：只针对操作数的整数部分操作

例　程：读取 a 的二进制数的第 1 位置 1 后，将值赋给 b 值

```
dim a,b
a=13'a = 01101B
b=set_bit(1,a)                  ' 将 a 的二进制数第 1 位置 1 后，赋值给 b
Print b                          'b =  01111B = 15
```

运行结果： b = 15

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

例　程：读取 a 的小数值，并将值赋给 b 值

```
dim a,b
a=11.33333'a 必须大于 0
b=frac(a)' 将 a 的小数部分赋值给 b
Print b
```

运行结果：　b = 0.33333

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

例　程：取 a 和比值的整数，并将值打印出来

```
dim a,b
a=6.66666
b=-2.22222
a=int(a)
b=int(b)
Print a,b
```

运行结果：　a = 6　　b = -3

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

例　程：判断 a、b、c 返回值，并打印数值

```
dim a,b,c
a=6.66666
```

```
b=0
c=-2.22222
a=sgn(a)
b=sgn(b)
c=sgn(c)
Print a,b,c
```

运行结果： a = 1 b = 0 c = -1

CHR

语 法：CHR(expression)

描 述：返回表达式的值对应的ASCII码

参 数：expression 表达式

⚠注意：该指令只用于PRINT指令后。ASCII码字符集参见下表

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	

⚠注意： 0~31对应的ACSII码为一些制表符。

例 程： 打印32～127对应的ASCII码字符

```
dim a
For a=32 to 127
Print a,chr(a)' 打印结果参见表 1.2
Next a
```

运行结果：具体如上表

STR

语 法：STR(variable)

描 述：打印变量中存储的字符串

参 数：variable 存储字符串的变量

⚠注意：该指令只用于PRINT指令后

STRING

语　法：STRING (“string”)

描　述：把字符串存储在一个变量中。字符串长度最多8个字符

参　数：“string”　　字符串

例　程：将字符串赋值给state，再打印出来

```
dim state(2)
state(0) = string("stop")
state(1) = string("run")
Print "motor:",str(state(0))                    ' 打印 motor: stop
Print "motor:",str(state(1))                    ' 打印 motor: run
```

运行结果：motor: stop ， motor: run

ASC

语　法：ASC(“string”)

描　述：获取字符串首字符的ASCII十进制值

参　数：“string”　　字符串

例　程：打印G的十进制ASCII值

```
Print ASC("G")
```

运行结果：71

HEX

语　法：HEX(“string”)

描　述：十六进制字符串转换为十进制数

参　数：“string”　　十六进制字符串

例　程：打印“0x10”的十进制值

```
Print HEX("0x10")
```

运行结果：16

4.1.2 系统默认常量与变量

PI

描　述：pi为常数，是圆周率，等于3.14159

TRUE

描　述：true为常数：-1　　（-1用二进制表示为；11111111B（以8位数为例））

FALSE

描　述：false为常数：0

ON

描　述：on为常数：1

OFF

描　述：off为常数：0

SYS_VAR_00~SYS_VAR_99

描　述：系统保留变量，可通过 BASIC 程序读写，也可通过 G 代码程序读写

4.1.3 流程控制指令

DIM

语　法：DIM　varname, arrayname(space)

描　述：定义变量，数组

参　数：varname　　变量名

arrayname　数组名

space　　数组长度

⚠注意：1）变量名、数组名不用定义也可以直接使用，但推荐先定义后使用

2）当定义了数组长度后，如果要重新定义该数组长度，必须先使用UNDIM删除该数组后再重新定义

3）数组元素由0开始编号

4）定义后可直接赋值，比如dim var=100.00，重复调用会重新赋值，区别于static

STATIC

语　法：STATIC　varname, arrayname(space)

描　述：定义变量，数组

参　数：varname　　变量名

arrayname　　数组名

space　　数组长度

⚠注意：1）变量名、数组名不用定义也可以直接使用，但推荐先定义后使用

2）当定义了数组长度后，如果要重新定义该数组长度，必须先使用 UNDIM 删除该数组后再重新定义

3）数组元素由 0 开始编号

4）定义后可直接赋值，比如 static var=100.00，重复调用不会重新赋值，区别于 dim

例　程：定义各变量、数值等，并进行赋值

```
dim a,b,c,r(3)                ' 定义变量 a，b，c；定义数组 r，长度为 3
a=1' 给变量赋值
b=2
c=3
r(0)=a                        ' 给数组赋值
r(1)=b
r(2)=c
Print r(0),r(1),r(2)
```

运行打印结果为：　1 2 3

UNDIM

语　法：UNDIM　varname, arrayname

描　述：删除变量，数组

参　数：varname　　变量名

arrayname　数组名

⚠注意：1）undim*　表示将清除之前定义的所有变量及数组

2）删除变量后不对寄存器进行清零，故当再次定义相同名称的变量后，其初始值可能并不是0。建议定义变量后都给变量赋值一个初始值，否则该变量的初始值可能不确定。

CONST

语　法：CONST　varname = value

描　述：定义符号表示的常数。采用常量名可避免在程序中多处修改同一个数值

参　数：　varname　常量名

value　　常数值

⚠注意：CONST变量在程序不可修改其数值，否则在编译中会报错

例　程：定义各常量值

```
const max_speed = 10000'定义速度常量
const pulse = 1600'定义脉冲常量
```

DMINS

语　法：DMINS　arrayname(pos)

描　述：数组的链表操作。插入后所有元素往后移

参　数：arrayname　　数组名

　　pos　　数组索引

例　程：在原数值1处插入数值a（1）

```
dim a(3)    '定义数组 a,包含 3 个元素
a(0)=1
a(1)=2
a(2)=3
dmins a(1) '在 a(1)处插入一个数组元素
Printa(0),a(1),a(2)
```

运行结果为：　1 2 2,原来的a(2)中的3被2覆盖。

DMDEL

语　法：DMDEL　arrayname(pos)

描　述：数组的链表操作。删除数组元素，删除后所有元素向前移动

参　数：arrayname　　数组名

　　pos　　数组索引

例　程：在原数值1处插入数值a（1）

```
dim a(3)
a(0)=1
a(1)=2
a(2)=3
Dmdel a(0) '将 a(0)中的内容删除，后面的元素前移
Print a(0),a(1),a(2)
```

运行结果为： 2 3 3，删除a(0)后，后面数值前移

IF THEN ELSEIF ENDIF

语　法：IF　<condition1> THEN
　　commands1
　ELSEIF <condition2> THEN
　　commands2
　ELSE
　　commands3
　ENDIF

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endif，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：condition1　　条件表达式1
　condition2　　条件表达式2
　commands　　指令块（即有多条指令）

例　程：判断变量a是大于10，还是小于1，还是大于等于1且小于等于10。

```
dim a
a=8
If  a>10  then          ' 判断条件 1
    Print"a>10"' 即 a 大于 10；打印：a >10
Elseif  a<1 then        ' 条件 1 不成立，即 a 小于 10；判断条件 2
    Print"a<1"' 条件 2 成立，即 a 小于 1；打印：a <1
else
    Print"1<=a<=10"' 条件 2 不成立，即 a<= 1，且 a <= 10
endif
```

运行结果为： 1<=a<=10

FOR TO STEP NEXT

语　法：FOR variable=start TO end [STEP increment]
　　commands

NEXT variable

描　述：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：variable　　循环变量名

start　　循环起始值

end　　循环结束值

increment　循环变量的增量，可选；缺省时，增量为1。增量可以是小数或负数。增量为负数时，循环起始值要大于循环结束值

commands　　指令块

⚠注意：该指令最多可重叠嵌套八层，不支持子程序调用

例　程：从1累加至100。

```
dim a,sum
sum=0
For a=1 to 100
    sum =sum+a
next a
Print sum
```

运行结果为：　5050

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

参　数：commands　　指令块

condition　条件表达式

⚠注意：while循环语句可能一次也不执行循环体内的指令块；而repeat指令，至少要执行一次循环体内的指令块

例　程：　从1开始累加自然数，当和大于等于5050时停止，并显示最后一个自然数。

```
dim a=0
```

```
dim sum=0
While sum<5050' 如果 sum<5050，则循环
    a=a+1
    sum=sum+a
wend
Print sum,a
```

运行结果为：5050　100

REPEAT UNTIL

语　法：REPEAT
　　commands
　　UNTIL condition

描　述：循环语句，循环执行commands指令块，当condition为真时退出循环

参　数：commands　　指令块
　　condition　　条件表达式

⚠注意：while循环语句可能一次也不执行循环体内的指令块；而repeat指令，至少要执行一次循环体内的指令块

例　程：从1开始累加自然数，当和大于等于5050时停止，并显示最后一个自然数。

```
dim a=0
dim sum=0
repeat
    a=a+1
    sum=sum+a
Until sum>=5050' 如果 sum>=5050，则退出循环
Print sum,a
```

运行结果为：5050　100

WAIT UNTIL

语　法：WAIT UNTIL condition

描　述：原地等待，直到条件满足

参　数：condition　　条件表达式

⚠注意：不支持子程序调用

例　程：当电机开始运动时，控制输出口3为低电平0；当运动至6400处时，控制输出口3为1。

```
SMCSetProfileUnit(0,0,5000,0.1,0.1,0)'设置速度参数
SMCPMoveUnit(0,10000,1) '定长运动
SMCWriteOutBit(3,0)      ' 输出口 3 输出高电平。即电机开始运动时，LED 亮
Wait until mpos>6400'  判断位置是否大于 6400
SMCWriteOutBit(3,1)   ' 输出口 3 输出高电平。LED 灭
```

运行结果为：当脉冲计数值大于6400时，输出口3输出高电平

GOTO

语　法：GOTO　label

描　述：强制跳转，也称为无条件跳转

参　数：label　　跳转目标的标号

⚠注意：1）使用该指令容易使程序混乱，尽量少用

2）该指令与子程序调用指令不同，该指令也无法使用return返回

3）不支持子程序调用

ON GOTO

语　法：ON expression GOTO label

描　述：条件跳转，当expression条件为真时，跳转

参　数：expression　　条件表达式

label　　跳转目标的标号

⚠注意：1）使用该指令容易使程序混乱，尽量少用

2）该指令与子程序调用指令不同，该指令也无法使用return返回

3）不支持子程序调用

DELAY

语　法：DELAY(delay_time)

参　数：delay_time　　延时时间，单位：ms

描　述：延时指令

4.1.4 子程序、多任务控制指令

GOSUB (CALL)

别　名：CALL

语　法：GOSUB　label

描　述：调用过程（子程序）

参　数：label　　过程名（子程序名）

RETURN(END SUB)

别　名：END SUB

语　法：RETURN

描　述：用户过程返回

SUB

语　法：SUB label(para1,para2,...,para8)，也可以写为 label：

描　述：定义过程（子程序）

参　数：label　　过程名（子程序名）

para1 参数 1

para2 参数 2

...

para8 参数 8

⚠注意：1、SUB 内部 DIM 的变量(包括参数变量)为局部变量，可与全局变量重名

2、通过参数变量可实现数据的双向传递

例程1：　计算1到10、1到100的和。

```
dim m,sum,i
m=10
gosub  sigma_m' 调用过程 sigma_m
Print sum
m=100
gosub sigma_m' 调用过程 sigma_m
Print sum
end
```

```
sigma_m:        ' 定义过程 sigma_m
    sum=0
For i=1to m    ' 计算 1 到 m 的和
    sum=sum+i
Next i
return' 过程结束
```

运动结果为： 55 ， 5050

例程 2：参数传递举例 100+200=300

```
dim para1 = 100
dim para2 = 200
dim result
Call fun(para1,para2,result)
Print para1,para2,result
Sub fun(para1,para2,byref result)
    result = para1 + para2
endsub
```

执行结果：100 200 300

备注：传递值需增加 byref ， 不然无法传递计算值

ON GOSUB

语　法：ON expression GOSUB label

描　述：当expression 条件为真时，调用过程label

参　数：expression　条件表达式

label　　过程（子程序）的标号

⚠注意：不支持子程序调用

RUN

语　法：RUN tasknum, label

描　述：启动多任务程序。多任务操作指令还有：END、STOP、AUTO、HALT等

参　数：tasknum　　任务号(1-4)，缺省则为当前任务

label　　任务名

END

语　法：END

描　述：结束当前任务或程序

STOP

语　法：STOP [tasknum]

描　述：任务强制停止

参　数：tasknum　　任务号，缺省当前任务

AUTO

描　述：主程序标号。如果有auto标号，开机自动运行该程序

PAUSE

语　法：PAUSE

描　述：暂停任务

HALT

语　法：HALT

描　述：停止全部任务

4.1.5 定时器控制指令

TIMER_START

语　法：TIMER_START(num, ms)

描　述：启动定时器

参　数：num　　定时器编号：0～控制器最大定时器数量-1

　　ms　　定时长度，单位：ms

TIMER_STOP

语　法：TIMER_STOP(num)

描　述：强制停止定时器

参　数：num　　定时器编号, 0～控制器最大定时器数量-1

TIMER_IFEND

语　法：TIMER_IFEND(num)

描　述：查询定时器是否停止

参　数：num　　定时器编号：0～控制器最大定时器数量-1

返回值：0：未停，1：停止

例程: 定时器功能应用

```
dim a,b
a=off' LED 状态
b=0' 开关状态置 0
While 1
  If timer_ifend(0)=1and b=1 then' 定时器停止且开关状态为 1 时，
     SMCWriteOutBit(3,a)
     a=not a                                    'LED 状态取反
     timer_start(0, 500)                        ' 再次开定时器 0
  endif
  If SMCReadInBit(1)=0 then
     timer_start(0,500)                         ' 输入口 1 为低电平，开定时器 0
     b=1' 开关状态置 1
  endif
  If SMCReadInBit(2)=0 then
     timer_stop(0)                              ' 输入口 2 为 1，关定时器 0
     b=0' 开关状态置 0
  endif
wend
```

运行结果：输入口1为低电平，开定时器0，控制LED灯500毫秒闪烁一次；输入口2为低电平，关定时器0，LED保持不变。

4.1.6 中断控制指令

BASIC系列运动控制器具有中断处理功能。中断源有：定时器、通用数字输入信号、软件限位信号、硬件限位信号、电机报警信号等。

中断处理程序属于子任务程序，在运行中断处理程序时，主程序不会停止运行。所有的中断处理程序只算作一个任务（不建议在中断任务中写过多的代码）。

Int_Enable

语　法：Int_Enable(enable)

描　述：开启或关闭中断

参　数:enable:0-禁用中断，1-启用中断

OnTimer

语　法：OnTimer0～控制器最大定时器-1

描　述：定时器计数结束中断入口

⚠注意:定时器0~控制器最大定时器-1均可使用,使用TIMER_STOP指令停止指定定时器后，不会进入该定时器的中断程序

OnInOn

语　法：OnInOn0～最大输入端口数-1

描　述：通用数字输入信号由0变为1时，产生中断的程序入口。即信号的上升沿触发中断

⚠注意:所有数字输入端口均可使用，上升沿定义：信号由无效切换为有效。

OnInOff

语　法：OnInOff0～最大输入端口数-1

描　述：通用数字输入信号由1变为0时，产生中断的程序入口。即信号的下降沿触发中断

⚠注意:所有数字输入端口均可使用，下降沿定义：信号由有效切换为无效。

OnSoftEl

语　法：OnSoftEl0～最大电机轴数-1

描　述：软件限位信号的中断处理程序入口

⚠注意:所有电机轴的软件限位均可使用

OnEl

语　法：OnEl0～最大电机轴数-1

描　述：硬件限位信号的中断处理程序入口

⚠注意:所有电机轴的正、负限位信号EL+、EL－均可使用

OnAlarm

语　法：OnAlarm0～最大电机轴数-1

描　述：伺服电机的报警信号ALM的中断处理程序入口

⚠注意:所有电机轴的ALM信号均可使用

4.1.7 信息输出设置指令

PRINT

描　述：在终端打印参数。通常用于输出一些信息。

例　程：打印下列指令

```
PRINT *SET ‘输出所有参数设置
PRINT *VAR ‘输出所有变量名及内容
PRINT *SUB ‘输出所有子过程名称
PRINT *ARRAY‘输出所有数组名
PRINT CHR(54) ‘ ASCII码字符输出。54对应于字符“6”
PRINT STR(字符串变量名) ‘ 输出字符串
PRINT  变量名‘输出变量值
PRINT “abcdef” ‘输出字符
PRINT *STOP        ‘输出当前程序停止的位置信息，如：停止的行号，任务等。
PRINT *RUN         ‘输出当前程序运行的位置信息，如：运行的行号，任务等。
PRINT BASIC_MAX_INF‘输出BASIC相关配置信息，如：最大任务数、定时器数等。
```

TRACE

描　述：显示字符串

VERSION

描　述：读取运动控制器软件版本

例　程：打印当前软件版本

```
Print version‘ 显示：601150707 固件版本
```

ERROR WORN TRACE

描　述：与PRINT指令类似，定义用户特殊输出，受ERRSWITCH开关控制

ERRSWITCH

描　述：出错输出开关。通过开关可以屏蔽ERROR等的输出

0：不输出

1：输出ERROR

2：输出WORN ERROR

3：输出TRACE WORN ERROR

4.1.8 断电数据保存指令

STATICREG/NVRAM_INT

语　法：STATICREG(num)
描　述：读写断电保护寄存器，数据类型为32位整数
参　数：num　　寄存器编号，0～4096

NVRAM/NVRAM_FLOAT

语　法：NVRAM(num)
描　述：读写断电保护寄存器，数据类型为32位浮点数,5位固定小数
参　数：num　　寄存器编号，0～4096

NVRAM_BYTE

语　法：NVRAM_BYTE(num)
描　述：读写断电保护寄存器，数据类型为8位整数
参　数：num　　寄存器编号，0～4096

NVRAM_SHORT

语　法：NVRAM_SHORT(num)
描　述：读写断电保护寄存器，数据类型为16位整数
参　数：num　　寄存器编号，0～4096

4.1.9 字符串操作指令

STRCMP

语　法 1：short strcmp(const char* string1,const char* string2)

语　法 2：short strcmp(const char* string1,const char* string2, int value)

描　述：字符串间比较

参　数：string1　比较的字符串 1

String2　比较的字符串 2

Value　比较的个数

返回值：返回值，0 相同，非 0 为不同

例　程：print strcmp("string1","string1")　　'比较两字符，结果为 0 表示两字符相同

例　程：print strcmp("aac","aad",2)　　'比较两字符串的前面两个字符，结果为 0 表示两字符串前面两个字符相同

STRCPY

语　法：short strcpy(char* dest,const char* src)

描　述：字符串间的复制

参　数：dest 目标字符串

　　Src 源字符串

返回值：返回整数

例程 1：将 b 的值复制到 a

```
dim a(4),b(4)
b(0)=1
b(1)=2
b(2)=3
b(3)=4
strcpy(a(0),b(0)) '将 b 的值复制到 a
Print *b              '打印结果数组对应值为 1、2、3、4
```

例程 2：将字符串的值复制到 a

```
dim a(10)
strcpy(a(0),"12345678")
Printstr(a)  '打印结果为 12345678
```

备注：输出应该书写为 print str（a）

STRTOF

语　法：float strtof(const char* string)

描　述：字符串转换为浮点数

参　数：string　需转换的字符串

返回值：返回浮点数

举例：数值字符串”1.3”转换为数值 1.3

```
dim  b(4)
b(0)=49'对应字符为"1"
b(1)=46'对应字符为"."
```

```
b(2)=51'对应字符为"3"
b(3)=0'对应字符串结束符
Print strtof(b(0))  '打印结果为 1.3
```

STRTOL

语　法：int strtol(const char* string)

描　述：字符串转换为长整数

参　数：string　需转换的字符串

返回值：返回整数

举例：数值字符串”123”转换为数值 123

```
dim  b(4)
b(0)=49'对应字符为"1"
b(1)=50'对应字符为"2"
b(2)=51'对应字符为"3"
b(3)=0'对应字符串结束符
Printstrtol(b(0))  '打印结果为 123
```

SPRINTF_L

语法：int sprintf_l(char* string,char* format,int value,int value1,int value2,
int value3)

描　述：字符串格式化命令，把整形数据按格式写入字符串

参　数：string　char 型指针，指向将要写入的字符串

Format　格式化字符串，例如“%d”“%c”等

Value　可选参数，类型为整数的数据

value1　可选参数 1，类型为整数的数据

value2　可选参数 2，类型为整数的数据

value3　可选参数 3，类型为整数的数据

返回值：返回整数

例　程：

```
dim strBuf(100)
sprintf_l(strBuf(0),"CurTick=%06d",ticks)'假设 ticks=1000
Printstr(strBuf)
```

打印结果 CurTick=001000

SPRINTF_F

语　法：int sprintf_f(char* string,char* format,float value,float value1,float value2,float value3)

描　述：字符串格式化命令，把浮点数据按格式写入字符串

参　数：string　char 型指针，指向将要写入的字符串

Format　格式化字符串，如“%f”“%02.3f”等

Value　可选参数，类型为浮点数

value1　可选参数 1，类型为浮点数

value2　可选参数 2，类型为浮点数

value3　可选参数 3，类型为浮点数

返回值：返回整数

例　程：

```
dim strBuf(100)
sprintf_f(strBuf(0),"CurPos=%06.3f",mpos(0))'假设 mpos(0)=1.01
Print str(strBuf)
```

打印结果 CurPos=01.010

STRCAT

语　法：short strcat(char *dest,char *src)

描　述：连接字符串，把 src 所指字符串添加到 dest 结尾处

参　数：dest　目标字符串

src　源字符串

返回值：返回整数

例　程：

```
dim strBuf(100),strBuf2(100)
strcpy(strBuf(0),"123")
strcpy(strBuf2(0),"456")
strcat(strBuf(0),strBuf2(0))
Print str(strBuf)
```

打印结果 123456

⚠注意：dest 必须有足够的空间来容纳 src 的字符串。

STRLEN

语　法：int strlen(char *s）

描　述：计算给定字符串的长度（不包括'\0'在内）

参　数：s　　需要计算长度的字符串

返回值：字符串长度

例　程：

```
dim strBuf(100)
strcpy(strBuf(0),"123")
Print strlen(strBuf(0))
```

打印结果 3

MEMSET

语　法：short memset(void *s, int ch, size_t n)

描　述：将 s 数组中的前 n 个数用 ch 替换

参　数：s　　需要替换的数组

ch　　类型为整数的数据

n　　需要替换的数组的数据个数

返回值：返回整数

例　程：将数组 b（10）清零

```
dim  b(10)
memset(b(0),0,10)
```

4.1.10 实时时钟

TICKS

描　述：读取系统开机后时钟的计时时间。单位：毫秒

例　程：每500毫秒显示一次时钟值。

```
While1
    Print ticks
    delay (500)
```

```
wend
```

运行结果：每隔500ms软件打印一次系统开机时间

RTC_TIMES

语　法：RTC_TIMES

描　述：读取 RTC 的运行秒数；掉电后不会丢失。

RTC_TIME

语　法：RTC_TIME

描　述：读取 RTC 的当前时间；掉电后不会丢失。

参　数：无

返回值：当前时间，例如：print RTC_TIME，结果为 104130 表示 10:41:30

例　程：设置时间

```
RTC_TIME=104130'设置时间 10:41:30
```

RTC_DATE

语　法：RTC_DATE

描　述：读取 RTC 的当前日期；掉电后不会丢失。

参　数：无

返回值：当前日期。例如：print RTC_DATE,结果为 20150316，表示 2015 年 3 月 16 日

例　程：设置日期

```
RTC_DATE=20150316'设置时间 2015 年 3 月 16 日
```

4.1.11 系统参数

SMCDefaultSet

语　法：void SMCDefaultSet()

描　述：恢复默认参数设置，参数总表请查看附表。

参　数：

返回值：无

⚠注意：恢复系统默认参数值，先执行指令 SMCEnterPassword 输入密码登陆，再恢复默认参数指令 SMCDefaultSet。

BURNSET

语　法：BURNSET

描　述：保存所有参数的当前值至FLASH，下次开机时会自动从FLASH读取保存的参数，参数总表请查看附表。

适用范围：全系列控制器

4.2 通讯连接指令

IPADDR

描　述：查询IP地址

例　程：print IPADDR'显示结果为：192.168.5.11

适用范围:全系列控制器

SETIP

语　法：SETIP(dot1,dot2,dot3,dot4)

描　述：修改IP地址

参　数：dot1　　第1个域

dot2　　第2个域

dot3　　第3个域

dot4　　第4个域

例　程：SETIP(192,168,5,11)　　'改变控制器 IP 地址为：192.168.5.11

⚠注意：IP参数之间用逗号隔开，修改IP地址后需重新连接控制器。特别需注意与电脑连接时，控制器IP段号与电脑的IP地址段相同（控制器默认段号为5）。

SETCOM

语　法：SETCOM (Baudrate,Databits,Stopbits,Parity,Port)

描　述：设置串口的通讯参数

参　数：Baudrate　波特率，选项为：2400, 4800, 9600, 19200, 38400, 57600, 115200

Databits　数据位，默认值为8。　位数范围：　7，　8。

Stopbits　停止位，范围：1, 2.默认值为1

Parity　校验方式，0-不校验,1-奇数，2-偶数

Port　通讯端口：1-RS232，2-RS485

⚠注意：1）连接设置中默认值：波特率115200，数据位固定为8，停止位为1，校验方式为偶校验。各参数可修改，如果该运动控制器是通过串口与计算机相连的话，数据位、停止位及校验方式必须设置为与编程软件中的一样。否则可能会出现无法与计算机通讯的情况。

2）使用 API 函数动态库时，数据位必须为 8 位。

COM

描　述：查询com口参数

例　程：print COM

显示结果

RS232:115200,8,1,2

RS485:9600,8,1,2

CAN:250,1

4.3 脉冲模式指令

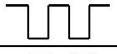
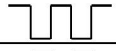
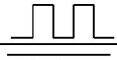
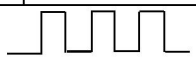
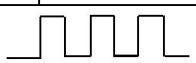
SMCSetPulseOutmode

语　法：short SMCSetPulseOutmode(WORD axis,WORD outmode)

描　述：设置指定轴的脉冲输出模式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

Outmode 脉冲输出方式选择，脉冲+方向（0、1、2、3）双脉冲（4、5）AB 相（6）

脉冲输出模式	正方向脉冲		负方向脉冲	
	PULSE 输出端	DIR 输出端	PULSE 输出端	DIR 输出端
0		高电平		低电平
1		高电平		低电平
2		低电平		高电平
3		低电平		高电平
4		高电平	高电平	
5		低电平	低电平	
6	PULSE 输出端  DIR 输出端 		PULSE 输出端  DIR 输出端 	

返回值：见错误代码

适用范围:脉冲型全系列控制器

⚠注意：1、请按照驱动器能接收的脉冲模式设置

2、AB 相输出 300 系列、600 系列支持

SMCGetPulseOutmode

语　法：short SMCGetPulseOutmode(WORD axis,WORD* outmode)

描　述：读取指定轴的脉冲输出模式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

Outmode 返回脉冲输出方式选择，脉冲+方向（0、1、2、3）双脉冲（4、5）AB 相（6）

返回值：见错误代码

适用范围:脉冲型全系列控制器

4.4 脉冲当量指令

SMCSetEquiv

语　法：short SMCSetEquiv(WORD axis, double equiv)

描　述：设置脉冲当量值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

equiv 脉冲当量，单位：pulse/unit

返回值：错误代码

适用范围：全系列控制器

⚠注意：运动关联此参数，必须在运动前设置！

SMCGetEquiv

语　法：short SMCGetEquiv(WORD axis, double* equiv)

描　述：读取脉冲当量值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

equiv 返回脉冲当量，单位：pulse/unit

返回值：错误代码

适用范围：全系列控制器

4.5 反向间隙指令

SMCSetBacklashUnit

语　法：short SMCSetBacklashUnit(WORD axis,double backlash)

描　述：设置反向间隙值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

backlash 反向间隙值，单位：unit

返回值：错误代码

适用范围：脉冲型全系列控制器

⚠注意：只有运动过程中有反向运动，指令才会生效！暂时适用只支持插补运动；

SMCGetBacklashUnit

语　法：short SMCGetBacklashUnit(WORD axis,double* backlash)

描　述：读取反向间隙设定值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

backlash　返回反向间隙值，单位：unit

返回值：错误代码

适用范围：脉冲型全系列控制器

4.6 状态监控指令

SMCStop

语　法：short SMCStop(WORD axis,WORD stop_mode)

描　述：指定轴停止运动

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

stop_mode 制动方式，0：减速停止，1：紧急停止

返回值：错误代码

适用范围：全系列控制器

⚠注意：此函数只适用于单轴、PVT 运动

SMCStopMulticoor

语　法：short SMCStopMulticoor(WORD Crd,WORD stop_mode)

描　述：停止坐标系内所有轴的运动

参　数：Crd 指定控制器上的坐标系号（取值范围：0~1）

stop_mode 制动方式，0：减速停止，1：立即停止

返回值：错误代码错误代码

适用范围：脉冲型全系列控制器

⚠注意：此函数只适用于插补运动

SMCEmgStop

语　法：short SMCEmgStop()

描　述：紧急停止所有轴

参　数：无

返回值：错误代码

适用范围：全系列控制器

⚠注意：此函数适用于所有运动模式

SMCCheckDone

语　法：short SMCCheckDone(WORD axis)

描　述：检测指定轴的运动状态

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

适用范围：全系列控制器

返回值：0：指定轴正在运行，1：指定轴已停止

SMCCheckDoneMulticoor

语　法：short SMCCheckDoneMulticoor(WORD Crd)

描　述：检测坐标系的运动状态

参　数：Crd 指定控制器上的坐标系号（取值范围：0~1）

返回值：坐标系状态，0：正在使用中，1：正常停止

适用范围：脉冲型全系列控制器

SMCAxisIoStatus

语　法：DWORD SMCAxisIoStatus(WORD axis)

描　述：读取指定轴有关运动信号的状态

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

返回值：见下表。

位号	信号名称	描述
0	ALM	1：表示伺服报警信号 ALM 为 ON；　0：OFF
1	EL+	1：表示正硬限位信号 +EL 为 ON；　0：OFF
2	EL-	1：表示负硬限位信号–EL 为 ON；　0：OFF
3	EMG	1：表示急停信号 EMG 为 ON；　0：OFF
4	ORG	1：表示原点信号 ORG 为 ON；　0：OFF
6	SL+	1：表示正软限位信号+SL 为 ON；　0：OFF
7	SL-	1：表示负软件限位信号-SL 为 ON；　0：OFF
8	INP	1：表示伺服到位信号 INP 为 ON；　0：OFF
9	EZ	1：表示 EZ 信号为 ON；　0：OFF
10	RDY 保留	1：表示伺服准备信号 RDY 为 ON；
11	DSTP	1：表示减速停止信号 DSTP 为 ON；
其他位	保留	

适用范围：全系列控制器

SMCSetPositionUnit

语　法：short SMCSetPositionUnit(WORD axis, double pos)

描　述：设置当前指令位置计数器值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

　　Pos　位置值，单位：unit

返回值：错误代码

适用范围：全系列控制器

SMCGetPositionUnit

语　法：double SMCGetPositionUnit(WORD axis)

描　述：读取当前指令位置值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

返回值：位置值，单位：unit

适用范围：全系列控制器

SMCGetTargetPositionUnit

语　法：double SMCGetTargetPositionUnit(WORD axis)

描　述：读取当前目标位置

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

返回值：位置值，单位：unit

适用范围：全系列控制器

SMCSetWorkPosUnit

语　法：short SMCSetWorkPosUnit(WORD axis, double pos)

描　述：设置当前工件原点

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

pos　位置值，单位：unit

返回值：错误代码

适用范围：SMC104A 控制器除外

SMCGetWorkPosUnit

语　法：double SMCGetWorkPosUnit(WORD axis)

描　述：读取当前工件原点坐标

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

返回值：工件位置值，单位：unit

适用范围：SMC104A 控制器除外

SMCReadCurrentSpeedUnit

语　法：double SMCReadCurrentSpeedUnit(WORD axis)

描　述：读取当前轴运动速度

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

返回值：当前轴运动速度值，单位：unit/s. 回读速度读取值带符号，正表示正向运动，负表示负向运动

适用范围：全系列控制器

4.7 点位运动指令

SMCSetProfileUnit

语　法：short SMCSetProfileUnit(WORD axis,double Min_Vel,double Max_Vel,double Tacc,double Tdec,double Stop_Vel)

描　述：设置单轴运动速度参数

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

Min_Vel 起始速度，单位：unit/s, 取值范围：0-2000000

Max_Vel 最大速度，单位：unit/s, 取值范围：0-2000000

Tacc 加速时间，单位：s，取值范围：大于等于 0.001

Tdec 减速时间，单位：s，取值范围：大于等于 0.001

Stop_Vel 停止速度，单位：unit/s，取值范围：0-2000000

返回值：错误代码

适用范围：全系列控制器

⚠注意：该函数需运动指令前设置才有效，可设置起始、停止速度，加减速时间决定加速段

例　程：

```
SMCSetProfileUnit (1,0,1000,0.1,0.2,0)
'设置轴 1 起、停速度为 0，加速时间为 0.1s，减速时间 0.2s，最大运行速度 1000
SMCSetProfileUnit (0,1000,2000,0.1,0.1,2000)
'设置轴 0 起始速度 100、停止速度为 200，加减速时间为 0.1s，最大运行速度 1000
```

SMCGetProfileUnit

语　法：short SMCGetProfileUnit(WORD axis,double* Min_Vel,double* Max_Vel,double* Tacc,double* Tdec,double* Stop_Vel)

描　述：读取单轴运动速度曲线参数

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

Min_Vel 返回起始速度，单位：unit/s, 取值范围：0-2000000

Max_Vel 返回最大速度，单位：unit/s, 取值范围：0-2000000

Tacc 返回加速时间，单位：s，取值范围：大于等于 0.001

Tdec 返回减速时间，单位：s，取值范围：大于等于 0.001

Stop_Vel 返回停止速度，单位：unit/s，取值范围：0-2000000

返回值：错误代码

适用范围：全系列控制器

SMCSetSprofile

语　法：short SMCSetSprofile(WORD axis,WORD s_mode,double s_para)

描　述：设置单轴速度曲线 S 段参数值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

s_mode　　保留参数，固定值为 0

s_para　　S 段时间，单位：s；范围：0~1

返回值：错误代码

适用范围：脉冲型全系列控制器

⚠注意：单轴速度曲线 S 平滑时间，若值为 0 则为 T 形曲线，不为 0 则为 S 平滑曲线。

例　程：

```
SMCSetSprofile (1,0, 0.1)   '设置轴 1 平滑系数 s 为 0.1s
SMCSetSprofile (0,0, 0)     '设置轴 0，速度曲线为 T 形曲线
```

SMCGetSprofile

语　法：short SMCGetSprofile(WORD axis,WORD* s_mode,double* s_para)

描　述：读取单轴速度曲线 S 段参数值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

s_mode　　返回保留参数，固定值为 0

s_para　　返回 S 段时间，单位：s；范围：0~1

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCSetDecStopTime

语　法：short SMCSetDecStopTime(WORD axis,double time)

描　述：设置减速停止时间

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

time 减速停止时间，单位：s

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCGetDecStopTime

语　法：short SMCGetDecStopTime(WORD axis,double* time)

描　述：读取减速停止时间

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

time 返回减速停止时间，单位：s

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCPmoveUnit

语　法：short SMCPmoveUnit(WORD axis,double Dist,WORD posi_mode)

描　述：定长运动，加/减速度恒定

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

Dist 目标位置，单位：unit

posi_mode　　运动模式，0：相对坐标模式，1：绝对坐标模式

返回值：错误代码

适用范围：全系列控制器

⚠ 注意：　需在运动指令执行前设置速度、加速度等相关参数才有效。

例　程：　绝对、相对定长运动

```
SMCPmoveunit (1,10000, 1)    '轴 1 绝对运动 10000 个脉冲
SMCPmoveunit (0,10000, 0)   '轴 0 相对运动 10000 个脉冲
```

SMCVmove

语　法：short SMCVmove(WORD axis,WORD dir)

描　述：指定轴恒速运动

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

dir　　运动方向，0：负方向，1：正方向

返回值：错误代码

适用范围：全系列控制器

⚠注意：　执行该函数时，需设定好速度等参数，当 dir 值大于等于 1 时为正向运动，小于等于 0 时为负向运动

例　程：

```
SMCvmove(1,0)                          '轴 1 负方向速度运动
SMCvmove(1,1)                          '轴 1 正方向速度运动
```

SMCResetTargetPositionUnit

语　法：short SMCResetTargetPositionUnit(WORD axis, double New_Pos)

描　述：在线改变指定轴的当前目标位置，位置值为运动的绝对位置点

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

New_Pos 新目标位置，单位：unit

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：只支持轴处于定长运动且运动处于未停止状态。最终位置点为运动的绝对位置值。

SMCUpdateTargetPositionUnit

语　法：short SMCUpdateTargetPositionUnit(WORD axis, double New_Pos)

描　述：强制改变指定轴的当前目标位置

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

New_Pos 新目标位置，单位：unit

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠ 注意：支持处于空闲或停止或正运行状态下的定长运动，最终位置点为运动的绝对位置值

SMCChangeSpeedUnit

语　法：short SMCChangeSpeedUnit(WORD axis, double New_Vel,double Taccdec)

描　述：在线改变指定轴的当前运动速度

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

New_Vel 新的运动速度，单位：unit/s

Taccdec 最大加减速时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：(1)支持轴处于定长或者恒速运动状态下执行。

(2) 定长运动下变速只改变速度大小，不改变运动方向；为了到达新的运动速度，实际加减速时间会根据当前速度以及剩余距离自动调整。

(3) 恒速运动下变速值与其运动方向有关。 若起始运行方向为正，变速值为负值时，运动将往负方向运动。变速值为正时，则运动方向不变，速度都会以设定变速值运行。若起始运行方向为负，变速值为正值时，运动将往正方向运动，变速值为负时，则运动方向不变，速

度都会以设定变速值运行。

4.8 回原点运动指令

SMCSetHomePinLogic

语　法：short SMCSetHomePinLogic(WORD axis,WORD org_logic,double filter)

描　述：设置 ORG 原点电平信号

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

org_logic　ORG 信号有效电平，0：低有效，1：高有效

filter　保留参数，固定值为 0

返回值：错误代码

适用范围：脉冲型全系列控制器

⚠注意：该函数设置轴为 255 时，表示所有轴都将统一被设置相应的参数。

SMCGetHomePinLogic

语　法：short SMCGetHomePinLogic(WORD axis,WORD* org_logic,double* filter)

描　述：读取 ORG 原点电平信号

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

org_logic　返回 ORG 信号有效电平，0：低有效，1：高有效

filter　返回保留参数，固定值为 0

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCSetHomePositionUnit

语　法：short SMCSetHomePositionUnit(WORD axis, WORD enable,double position)

描　述：回零完成后，设置位置计数值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

enable　使能参数

0. 禁止。

1: 先清 0，然后运动到指定位置（相对位置）

2: 先运动到指定位置（相对位置），再清 0

position　　设置计数偏移位置值

返回值：错误代码

备　注：enable 为 1,2 时，偏移运动的参数为点位运动参数，相关函数为 SMCSetProfileUnit, SMCSetSprofileUnit

适用范围：全系列控制器

SMCGetHomePositionUnit

语　法：short SMCGetHomePositionUnit(WORD axis, WORD *enable,double *position)

描　述：读取回零完成后，位置计数设置值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

enable　　使能参数

0. 禁止。

1: 先清 0，然后运动到指定位置（相对位置）。

2: 先运动到指定位置（相对位置），再清 0

position　　回读位置偏移计数值

返回值：错误代码

适用范围：全系列控制器

SMCSetHomeProfileUnit

语　法：short SMCSetHomeProfileUnit(WORD axis,double Low_Vel,double High_Vel,double Tacc,double Tdec)

描　述：设置回原点速度参数

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数-1

Low_Vel　设置回原点起始速度

High_Vel　设置回原点运行速度

Tacc　　设置回原点加速、减速时间，单位：s

Tdec　　保留值 0

返回值，见错误代码，可省略不写

适用范围：全系列控制器

SMCGetHomeProfileUnit

语　法：short SMCGetHomeProfileUnit(WORD axis,double* Low_Vel,double* High_Vel,double* Tacc,double* Tdec)

描　述：回读回原点速度参数

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数-1

Low_Vel　回读设置回原点起始速度

High_Vel　回读设置回原点运行速度

Tacc　　回读设置回原点加速、减速时间，单位：s

Tdec　　保留值 0

返回值，见错误代码，可省略不写

适用范围：全系列控制器

SMCSetEZCount

语　法：short SMCSetEZCount(WORD axis,WORD Count)

描　述：设置回零 EZ 个数

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数-1

Count　　EZ 个数值

返回值：错误代码

适用范围：SMC306A、SMC600 系列

SMCGetEZCount

语　法：short SMCGetEZCount(WORD axis,WORD *Count)

描　述：回读回零 EZ 个数

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数-1

Count　　回读 EZ 个数设置

返回值：错误代码

适用范围：SMC300、SMC600 系列

SMCSetHomeMode

语　法：short SMCSetHomeMode(WORD axis,WORD home_dir,double vel_mode,WORD mode,WORD Source)

描　述：设置回原点模式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

home_dir　回原点方向，0：负向，1：正向

vel_mode　回原点速度模式，默认值：1

Mode　　回原点模式：

① 总线型控制器回零模式参考驱动器设置

② SMC100 系列支持的回零模式如下：

0：一次回零，即平台高速向原点传感器方向运动，当原点传感器 触发，电机立即停止

1：一次回零加回找，即平台高速向原点传感器方向运动，当原点 传感器触发后，电机低速反转；退出原点传感器触发区域后， 电机立即停止

2：二次回零，即平台高速向原点传感器方向运动，当原点传感器 触发后，电机低速反转；退出原点传感器触发区域后，再次以 低速向原点传感器方向运动；当原点传感器触发，电机立即停 止。

3：一次回零后再记一个 EZ 脉冲进行回零，即平台高速向原点传 感器方向运动，当原点传感器触发后，电机以低速继续向前运 动，当编码器上的 EZ 信号触发后，电机立即停止

4：记一个 EZ 脉冲进行回零，EZ 信号的触发次数为 1。电机以低 速向前运动，当编码器上的 EZ 信号触发后，电机立即停止。

5：一次回零后反向再记一个 EZ 脉冲进行回零，即平台高速向原 点传感器方向运动，当原点传感器触发后，电机以低速反向运动，当编码器上的 EZ 信号触发后，电机立即停止。（保留）

③ SMC300、SMC600 系列支持的回零模式如下：

0：一次回原点，即平台高速向原点传感器方向运动，当原点传感器触发，电机立即停止。

1：一次回原点加反找，即平台高速向原点传感器方向运动，当原点传感器触发后，电机低速反转；退出原点传感器触发区域后，电机立即停止。

2：二次回原点，即平台高速向原点传感器方向运动，当原点传感器触发后，电机低速反转；退出原点传感器触发区域后，再次以低速向原点传感器方向运动；当原点传感器触发，电机立即停止。

3：一次回原点 ＋EZ回原点方式，即平台高速向原点传感器方向运动，当原点传感器触发后，电机以低速继续向前运动，当编码器上的EZ信号触发后，电机立即停止。

4：EZ单独回原点。EZ信号的触发次数为1。电机以低速向前运动，当编码器上的EZ信号触发后，电机立即停止。

5：一次回零再反找 EZ。该方式在回原点运动过程中，当找到原点信号后，减速停止，

然后以反找速度反向找到 EZ 生效此时电机停止

6：原点锁存。电机先以设定速度回原点，当原点开关边沿触发时，将当前位置锁存下来，同时电机减速停止。电机减速停止完成后再反向回找锁存位置，运动到锁存位置，电机停止。

7：原点锁存加同向 EZ 锁存。此模式先以模式 3 的方式执行一次原点锁存回零，完成后继续沿设定回零方向运行到 EZ 信号产生，EZ 信号产生时锁存当前位置并执行减速停，电机减速停止之后再反向回找 EZ 的锁存位置，运动到锁存位置，电机停止。

8：单独记一个 EZ 锁存。在回零过程中检测到 EZ 有效边沿出现，锁存当前位置，执行减速停，电机减速停止之后再反向回找 EZ 的锁存位置，运动到锁存位置，电机停止。

9：原点锁存加反向 EZ 锁存。此模式先以模式 3 的方式执行一次原点锁存回零，完成后以与设定回零方向相反的方向运行到 EZ 信号产生，EZ 信号产生时锁存当前位置并执行减速停，电机减速停止之后再反向回找 EZ 的锁存位置，运动到锁存位置，电机停止

Source　回零计数源，0：指令位置计数器，1：编码器计数器

返回值：错误代码

适用范围：全系列控制器

⚠注意：当回原点模式 mode=4 时，回原点速度模式将固定为低速回原点

SMCGetHomeMode

语　法：short SMCGetHomeMode(WORD axis,WORD* home_dir,double* vel_mode,WORD* mode,WORD* Source)

描　述：读取回原点模式参数

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

home_dir　返回回原点方向，0：负向，1：正向

vel_mode　返回回原点速度模式，默认值：1

mode　返回回原点模式

Source　返回保留参数，固定值为 0

返回值：错误代码

适用范围：全系列控制器

SMCHomeMove

语　法：short SMCHomeMove (WORD axis)

描　述：回原点运动

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

返回值：错误代码

适用范围：全系列控制器

SMCGetHomeResult

语　法：short SMCGetHomeResult(WORD axis,WORD* state)

描　述：读取回原点运动状态

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

state　返回回原点运动状态，0：未完成，1：完成

返回值：错误代码

适用范围：全系列控制器

4.9 PVT 运动指令

SMCPttTableUnit

语　法：short SMCPttTableUnit(WORD axis,DWORD count,double*pTime,double*pPos)

描　述：向指定数据表传送数据，采用 PTT 模式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

count 数据点个数，每个数据表具有 1000 个存储空间，每个数据点占用 1 个存储空间

pTime　数据点时间数组，单位：s

pPos　数据点位置数组，单位：unit

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：（1）下载的第一组（即起始点）数据中位置、时间必须为 0；数组中的数据都是以起始点的数据为参考点

（2）调用该指令向数据表中传递数据时，会删除数据表中原先的数据，因此所有数据应当一次传送完毕。如果使用数据表的轴正在运动，禁止更新数据表

SMCPtsTableUnit

语　法：Short SMCPtsTableUnit (WORD axis,DWORD count, double*pTime, double*pPos,double*pPercent)

描　述：向指定数据表传送数据，采用 PTS 模式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

count 数据点个数，每个数据表具有 1000 个存储空间，每个数据点占用 1 个存储空间

pTime 数据点时间数组，单位：s

pPos 数据点位置数组，单位：unit

pPercent 数据点百分比数组，百分比的取值范围：[0,100]

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：（1）下载的第一组（即起始点）数据中位置、时间必须为 0，数组中的数据都是以起始点的数据为参考点

（2）调用该指令向数据表中传递数据时，会删除数据表中原有数据，因此所有数据应当一次传送完毕。如果使用数据表的轴正在运动，禁止更新数据表

SMCPvtTableUnit

语　法：short SMCPvtTableUnit(WORD axis,DWORD count, double *pTime,double *pPos,double *pVel)

描　述：向指定数据表传送数据，采用 PVT 模式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

count 数据点个数，每个数据表具有 5000 个存储空间，每个数据点占用 1 个存储空间

pTime 数据点时间数组，单位：s

pPos 数据点位置数组，单位：unit

pVel 数据点速度数组，单位：unit/s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：（1）下载的第一组（即起始点）数据中位置、时间、速度必须为 0；数组中的数据都是以起始点的数据为参考点

（2）调用该指令向数据表中传递数据时，会删除数据表中原有数据，因此所有数据应当一次传送完毕。如果使用数据表的轴正在运动，禁止更新数据表

SMCPvtsTableUnit

语　法：short SMCPvtsTableUnit (WORD axis,DWORD count,double*pTime,double*pPos,double velBegin,double velEnd)

描　述：向指定数据表传送数据，采用 PVTS 模式

参　数：　axis　指定轴号，取值范围：0-控制器最大轴数-1

Count 数据点个数，每个数据表具有 5000 个存储空间，每个数据点占用 1 个存储空间

pTime 数据点时间数组，单位：s

pPos 数据点位置数组，单位：unit

velBegin　设置的第一点的速度，单位：unit/s

velEnd　　设置的最后一点的速度，单位：unit/s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：（1）下载的第一组（即起始点）数据中位置、时间、速度必须为 0；数组中的数据都是以起始点的数据为参考点

（2）调用该指令向数据表中传递数据时，会删除数据表中原有数据，因此所有数据应当一次传送完毕。如果使用数据表的轴正在运动，禁止更新数据表

SMCPvtMove

语　法：short SMCPvtMove(WORD AxisNum,WORD* AxisList)

描　述：启动 PVT 运动

参　数：AxisNum 轴数，取值范围：1-控制器最大轴数

AxisList 轴列表

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

4.10 插补运动参数指令

SMCSetVectorProfileUnit

语　法：Short SMCSetVectorProfileUnit(WORD crd,double Min_Vel,double Max_Vel,double Tacc,double Tdec,double Stop_Vel)

描　述：设置插补运动速度参数

参　数：Crd 坐标系号，取值范围：0~1

Min_Vel 起始速度，单位：unit/s

Max_Vel 最大速度，单位：unit/s

Tacc 加速时间，单位：s

Tdec 减速时间，单位：s

Stop_Vel 停止速度，单位：unit/s

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCGetVectorProfileUnit

语　法：short SMCGetVectorProfileUnit (WORD crd,double* Min_Vel,

double* Max_Vel,double* Tacc,double* Tdec,double* stop_vel)

描　述：读取插补运动速度参数

参　数：Crd 坐标系号，取值范围：0~1

Min_Vel 起始速度，单位：unit/s

Max_Vel 最大速度，单位：unit/s

Tacc 加速时间，单位：s

Tdec 减速时间，单位：s

Stop_Vel 停止速度，单位：unit/s

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCSetVectorSprofile

语　法：short SMCSetVectorSprofile(WORD Crd,WORD s_mode,double s_para)

描　述：设置插补运动速度曲线的 S 段时间

参　数：Crd　　坐标系号，取值范围：0~1

s_mode　　保留参数，固定值为 0

s_para　　S 段时间，单位：s，范围：0~1

返回值：错误代码

适用范围：除 SMC100 系列外的控制器

SMCGetVectorSprofile

语　法：short SMCGetVectorSprofile(WORD Crd,WORD* s_mode,double* s_para)

描　述：读取插补运动速度曲线的 S 段时间

参　数：Crd 坐标系号，取值范围：0~1

s_mode 返回保留参数，固定值为 0

s_para　返回 S 段时间，单位：s，范围：0~1

返回值：错误代码

适用范围：除 SMC100 系列外的控制器

SMCSetVectorDecStopTime

语　法：short SMCSetVectorDecStopTime(WORD Crd,double time)

描　述：设置插补异常减速停止时间参数

参　数：Crd 坐标系号，取值范围：0~1

time 减速停止时间，单位：s

返回值：错误代码

适用范围：除 SMC100 系列外的控制器

SMCGetVectorDecStopTime

语　法：short SMCGetVectorDecStopTime(WORD Crd,double* time)

描　述：读取插补异常减速停止时间参数

参　数：Crd 坐标系号，取值范围：0~1

time 返回减速停止时间，单位：s

返回值：错误代码

适用范围：除 SMC100 系列外的控制器

SMCSetArcLimit

语　法：short SMCSetArcLimit(WORD Crd,WORD Enable)

描　述：设置圆弧插补限速功能

参　数：Crd　　坐标系号，取值范围：0~1

Enable　　使能参数，0：不限速，1：圆弧限速

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：圆弧限速只用于连续插补模式 1、和模式 2。

SMCGetArcLimit

语　法：short SMCGetArcLimit(WORD Crd,WORD* Enable);

描　述：回读圆弧插补限速功能

参　数：Crd　　坐标系号，取值范围：0~1

Enable　返回使能参数，0：不限速，1：圆弧限速

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

4.11 插补运动指令

SMCLineUnit

语　法：short SMCLineUnit(WORD Crd,WORD AxisNum,WORD* AxisList,double* Target_Pos,WORD posi_mode)

描　述：启动直线插补运动

参　数：Crd 坐标系号，取值范围：0~1

AxisNum 轴数，取值范围：2-控制器最大轴数

AxisList 轴号列表

Target_Pos 目标位置列表，单位：unit

posi_mode 运动模式，0：相对坐标模式，1：绝对坐标模式

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCArcMoveCenterUnit

语　法：ShortSMCArcMoveCenterUnit(WORD crd,WORD AxisNum,WORD* AxisList,double *Target_Pos,double *Cen_Pos,WORD Arc_Dir,long Circle,WORD posi_mode)

描　述：圆心+圆弧终点模式扩展的螺旋线插补运动（可作两轴圆弧插补）

参　数：Crd 坐标系号，取值范围：0~1

AxisNum 轴数，取值范围：2-控制器最大轴数

AxisList 轴号列表

Target_Pos 目标位置数组，单位：unit

Cen_Pos 圆心位置数组，单位：unit

Arc_Dir 圆弧方向，0：顺时针，1：逆时针

Circle 圈数，取值范围：大于等于 0，该值表示螺旋线的圈数。如，0 即表示 0 圈螺旋线插补，1 表示 1 圈螺旋线插补

posi_mode 运动模式，0：相对坐标模式，1：绝对坐标模式

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCArcMoveRadiusUnit

语 法：short SMCArcMoveRadiusUnit(WORD crd,WORD AxisNum,WORD* AxisList,double *Target_Pos,double Arc_Radius,WORD Arc_Dir,long Circle,WORD posi_mode)

描 述：半径+圆弧终点模式扩展的螺旋线插补运动（可作两轴圆弧插补）

参 数：Crd 坐标系号，取值范围：0~1

AxisNum 轴数，取值范围：2-控制器最大轴数

AxisList 轴号列表

Target_Pos 目标位置数组，单位：unit

Arc_Radius 圆弧半径值，单位：unit

Arc_Dir 圆弧方向，0：顺时针，1：逆时针

Circle 圈数，取值范围：大于等于 0

该值表示螺旋线的圈数。如，0 即表示 0 圈螺旋线插补，1 即表示 1 圈螺旋线插补

posi_mode 运动模式，0：相对坐标模式，1：绝对坐标模式

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCArcMove3pointsUnit

语 法：short SMCArcMove3pointsUnit(WORD Crd,WORD AxisNum,WORD* AxisList,double *Target_Pos,double *Mid_Pos,long Circle,WORD posi_mode)

描 述：三点圆弧模式扩展的螺旋线插补运动（可作两轴及三轴空间圆弧插补）

参 数：Crd 坐标系号，取值范围：0~1

AxisNum 轴数，取值范围：2-控制器最大轴数（若为空间圆弧则轴数为 3）

AxisList 轴号列表

Target_Pos 目标位置数组，单位：unit

Mid_Pos 中间位置数组，单位：unit

Circle 圈数：

负数：表示此时执行的为空间圆弧插补

该值的绝对值减 1 表示空间圆弧的圈数。如，-1 即表示 0 圈空间圆弧，-2 即表示 1 圈空间圆弧…

非负数：表示此时执行的为圆柱螺旋线插补

该值表示螺旋线的圈数。如，0 即表示 0 圈螺旋线插补，1 即表示 1 圈螺旋线插补…

posi_mode 运动模式，0：相对坐标模式，1：绝对坐标模式

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

4.12 连续插补相关指令

SMCContiSetLookAheadMode

语 法：short SMCContiSetLookAheadMode (WORD Crd, WORD Mode, DWORD LookaheadSegment, double PathError, double LookaheadAcc)

描 述：设置插补模式及小线段前瞻参数

参 数：Crd 坐标系号，取值范围： 0~1

Mode 插补模式：0-非前瞻模式 0，1-前瞻模式 1，2-非前瞻模式 2

LookaheadSegment 前瞻插补段数，范围：[2,5000]

PathError 前瞻插补轨迹误差，单位：unit

LookaheadAcc 前瞻拐弯加速度值，单位：unit/s^2，该值越大，拐角速度越大，可能造成设备振动越大。

返回值：错误代码

适用范围：SMC600 系列控制器

⚠ 注意：前瞻段数、轨迹误差、拐弯加速度都只有在前瞻运动时才有效。

SMCContiGetLookAheadMode

语　法：short SMCContiGetLookAheadMode(WORD Crd, WORD* Mode, DWORD* LookaheadSegment,double* PathError, double* LookaheadAcc)

描　述：　读取插补模式及小线段前瞻参数

参 数：　Crd 坐标系号，取值范围：　0~1

Mode 回读插补模式：0-非前瞻模式 0，1-前瞻模式 1，2-非前瞻模式 2

LookaheadSegment 回读前瞻插补段数，范围：[2,5000]

PathError 回读前瞻插补轨迹误差，单位：unit

LookaheadAcc 回读拐弯加速度值，单位：unit/s^2

返回值：错误代码

适用范围：SMC600 系列控制器

SMCContiSetBlend

语　法：short SMCContiSetBlend(WORD Crd,WORD blend_mode)

描　述：设置连续插补 Blend 拐角过渡模式使能状态

参　数：Crd 坐标系号，取值范围：0~1

blend_mode 拐角平滑过渡，0：禁用，1：允许

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：

1）当插补模式为 0 时起作用，用 SMCContiSetLookaheadMode 设置

2）当使能 Blend 拐角平滑过渡后，各段运动轨迹之间的拐角将平滑过渡，从而得到更加平滑的速度曲线。

3）当计算机执行到此指令时，该函数设置将存入缓冲区，从该函数的下一条运动函数开始运动时起作用。

4）当使能拐角平滑后，除非再次调用该函数禁止拐角平滑过渡，否则后续的运动过程中拐角都将平滑过渡。

SMCContiGetBlend

语　法：short SMCContiGetBlend(WORD Crd,WORD* blend_mode)

描　述：读取连续插补 Blend 拐角过度模式使能状态设置

参　数：Crd 坐标系号，取值范围：0~1

blend_mode 返回拐角平滑过渡值，0：禁用，1：允许

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCContiOpenList

语　法：short SMCContiOpenList(WORD Crd,WORD AxisNum,WORD* AxisList)

描　述：打开连续插补缓冲区

参　数：Crd 坐标系号，取值范围：0~1

AxisNum　轴数，取值范围：2-控制器最大轴数

AxisList 轴号列表

返回值：错误代码

⚠注意：1）连续缓冲区最多可缓存 5000 条指令

2）在执行圆弧或螺旋线运动时，XYZ 为主动轴，UVW 为辅助轴；主动轴执行圆弧或螺旋线运动，辅助轴不执行圆弧及螺旋线运动，但跟随主动轴做线性运动

3）当打开连续插补缓冲区后，则进入连续插补模式；此时，除非当执行完缓冲区中的指令或是调用停止连续插补指令 SMCContiStopList 后，参与连续插补的运动轴才能退出连续插补模式

适用范围：SMC300、SMC600 系列控制器

SMCContiStartList

语　法：short SMCContiStartList(WORD Crd)

描　述：开始连续插补

参　数：Crd 坐标系号，取值范围：0~1

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCContiCloseList

语　法：short SMCContiCloseList(WORD Crd)

描　述：关闭连续插补缓冲区

参　数：Crd 坐标系号，取值范围：0~1

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCContiPauseList

语　法：short SMCContiPauseList(WORD Crd)

描　述：暂停连续插补

参　数：Crd 坐标系号，取值范围：0~1

返回值：错误代码

⚠注意：当暂停连续插补后，连续插补运动将减速停止，当再次调用 SMCContiStartList 指令时运动控制器将继续运行之前未完成的连续插补轨迹，暂停减速停止时间由 SMCSetVectorDecStopTime 设置。

适用范围：SMC300、SMC600 系列控制器

SMCContiStopList

语　法：short SMCContiStopList(WORD Crd,WORD stop_mode)

描　述：停止连续插补

参　数：Crd 坐标系号，取值范围：0~1

stop_mode 停止模式，0：减速停止，1：立即停止

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：当正在执行连续插补运动时，通过此指令可以中止连续插补运动，并使参与连续插补的运动轴退出连续插补模式

SMCContiChangeSpeedRatio

语　法：short SMCContiChangeSpeedRatio (WORD Crd,double Percent)

描　述：动态调整连续插补速度比例

参　数：Crd 坐标系号，取值范围：0~1

Percent 速度比例，取值范围：0~2.0

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：当计算机执行到此指令时，运动控制器将在下段轨迹调整连续插补速度比例

SMCContiDelay

语　法：short SMCContiDelay(WORD Crd, double delay_time,long mark)

描　述：连续插补中暂停延时指令

参　数：Crd 坐标系号，取值范围：0~1

delay_time 延时时间，单位：秒

mark 标号，任意指定，0 表示自动编号

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：1）延时时间为运动停止时的等待时间

2）当延时时间设置为 0 时，延时时间将无限长

4.13 连续插补状态指令

SMCContiRemainSpace

语　法：long SMCContiRemainSpace(WORD Crd)

描　述：查询连续插补缓冲区剩余插补空间

参　数：Crd　　坐标系号，取值范围：0~1

返回值：连续插补缓冲区剩余大小，范围：1-5000

适用范围：SMC300、SMC600系列控制器

SMCContiReadCurrentMark

语　法：long SMCContiReadCurrentMark(WORD Crd)

描　述：读取连续插补缓冲区当前插补段号

参　数：Crd　　坐标系号，取值范围：0~1

返回值：当前连续插补段号

适用范围：SMC300、SMC600系列控制器

SMCContiGetRunState

语　法：short SMCContiGetRunState(WORD crd)

描　述：读取连续插补运动状态

参　数：Crd　　坐标系号，取值范围：0~1

返回值：运动状态，0：运动中，1：暂停中，2：正常停止，3：未启动，4：空闲

5：异常减速停止中

适用范围：SMC300、SMC600系列控制器

4.14 连续插补 IO 控制指令

SMCContiSetPauseOutput

语　法：short SMCContiSetPauseOutput(WORD crd,WORD action,long mask,long state)

描　述：设置连续插补暂停及异常停止时 IO 输出状态

参　数：Crd　　坐标系号，取值范围：0~1

action　　激活模式：

0：保持原状

1：暂停连续插补时输出设定的 IO 状态，恢复运行时不恢复暂停前的 IO 状态

2：暂停连续插补时输出设定的 IO 状态，继续运行时恢复暂停前的 IO 状态

3：当暂停、停止连续插补，或遇到其他异常停止（如碰到 EMG 信号）时，输出设定的 IO 状态

mask　　选择输出端口标志：bit0~bit31 代表 Out0~Out31，位值为 1 时输出，位值为 0 不输出

state　　输出电平状态：bit0~bit31 代表 Out0~Out31，位值为 1 时输出高电平，位值为 0 时输出低电平

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠激活模式 3 的说明：1）暂停连续插补时，运动控制器输出设定的 IO 状态，继续运行时恢复暂停前的 IO 状态

2）停止连续插补、或遇到其他异常停止时，运动控制器输出设定的 IO 状态，但是再次启动连续插补时不会恢复之前的 IO 状态

SMCContiGetPauseOutput

语　法：short SMCContiGetPauseOutput(WORD crd,WORD* action,long* mask,long* state)

描　述：读取连续插补暂停及异常停止时 IO 输出状态

参　数：Crd　　坐标系号，取值范围：0~1

action　　返回激活模式：

0：保持原状

1：暂停连续插补时输出设定的 IO 状态，恢复运行时不恢复暂停前的 IO 状态

2：暂停连续插补时输出设定的 IO 状态，继续运行时恢复暂停前的 IO 状态

3：当暂停、停止连续插补，或遇到其他异常停止（如碰到 EMG 信号）时，输出设定的 IO 状态

mask　　返回选择输出端口标志：bit0~bit31 代表 Out0~Out31，位值为 1 时输出，位值为 0 不输出

state　　返回输出电平状态：bit0~bit31 代表 Out0~Out31，位值为 1 时输出高电平，位值为 0 时输出低电平

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCContiWaitInput

语　法：short SMCContiWaitInput(WORD Crd,WORD bitno,WORD on_off,double TimeOut,long mark)

描　述：连续插补等待 IO 输入

参　数：Crd　　坐标系号，取值范围：0~1

bitno　　输入口号，取值范围：0-控制器本机输入口数目-1

on_off　　电平状态，0：低电平，1：高电平

TimeOut　　超时时间，单位：s

mark　　标号，任意指定，0 表示自动递增编号

返回值：错误代码

⚠注意：1）当超时时间设为 0 时，运动控制器将一直等待 IO 输入信号，超时时间为无限长

2）超时时间不为 0 时，在超时时间内，一旦有相应的 IO 响应，马上运行下段轨迹。若无 IO 响应，等待时间大于超时时间后，程序将自动运行下段轨迹。

适用范围：SMC300、SMC600 系列控制器

SMCContiDelayOutbitToStart

语　法：short SMCContiDelayOutbitToStart(WORD Crd, WORD bitno,WORD on_off,double delay_value,WORD delay_mode,double ReverseTime)

描　述：连续插补中相对于轨迹段起点 IO 滞后输出（段内执行）

参　数：ConnectNo 指定链接号：0-7

Crd　　坐标系号，取值范围：0~1

bitno　　输出口号，取值范围：0-控制器本机输出口数目-1

on_off　　电平状态，0：低电平，1：高电平

delay_value　滞后值，单位：s（滞后时间模式）或 unit（滞后距离模式）

delay_mode　滞后模式，0：滞后时间，1：滞后距离

ReverseTime　电平输出后的延时翻转时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：1）设置的IO操作，将在该指令的下一条轨迹中起作用

2）当ReverseTime参数设置为0时，相应IO端口电平将不会翻转，保持设置值不变

3）当滞后模式选择为滞后距离时，位置源为指令位置计数器

SMCContiDelayOutbitToStop

语　法：short SMCContiDelayOutbitToStop(WORD Crd, WORD bitno,WORD on_off,double delay_time)

描　述：连续插补中相对于轨迹段终点 IO 滞后输出

参　数：Crd　　坐标系号，取值范围：0~1

bitno　　输出口号，取值范围：0-控制器本机输出口数目-1

on_off　　电平状态，0：低电平，1：高电平

delay_time　滞后时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：1）设置的 IO 操作，将在该指令的下一条轨迹结束后起作用

2）如果使用了 SMCContiClearIoAction 函数清除段内未执行完的 IO 动作时，那么该

指令将不会被执行

SMCContiAheadOutbitToStop

语　法：short SMCContiAheadOutbitToStop(WORD Crd, WORD bitno,WORD on_off,double ahead_value,WORD ahead_mode,double ReverseTime)

描　述：连续插补中相对于轨迹段终点 IO 提前输出（段内执行）

参　数：Crd　　坐标系号，取值范围：0~1

bitno　　输出口号，取值范围：0-控制器本机输出口数目-1

on_off　　电平状态，0：低电平，1：高电平

ahead_value　提前值，单位：s（提前时间模式）或 unit（提前距离模式）

ahead_mode　提前模式，0：提前时间，1：提前距离

ReverseTime　电平输出后的延时翻转时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠ 注意：1）设置的 IO 操作，将在该指令的下一条轨迹中起作用

2）当ReverseTime参数设置为0时，相应IO端口电平将不会翻转，保持设置值不变

3）当滞后模式选择为滞后距离时，位置源为指令位置计数器

SMCContiWriteOutbit

语　法：short SMCContiWriteOutbit(WORD Crd, WORD bitno,WORD on_off,double ReverseTime)

描　述：连续插补中缓冲区立即 IO 输出

参　数：Crd　　坐标系号，取值范围：0~1

bitno　　输出口号，取值范围：0-控制器本机输出口数目-1

on_off　　电平状态，0：低电平，1：高电平

ReverseTime　电平输出后的延时翻转时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：1）当控制器执行到此指令时，将该指令存入缓冲区，当缓冲区中的上一段运动指令执行完毕时，该指令将执行

2）该指令调用之后，前一段轨迹与下一段轨迹的速度曲线将不连续，即 Blend 平滑

模式在这两段轨迹之间不起作用

3）当 ReverseTime 参数设置为 0 时，相应 IO 端口电平将不会翻转，保持设置值不变

SMCContiClearIoAction

语　法：short SMCContiClearIoAction(WORD Crd, DWORD IoMask)

描　述：清除段内未执行完的 IO 动作

参　数：Crd　　坐标系号，取值范围：0~1

IoMask　清除标志：bit0~bit31 分别表示 Out0~Out31 输出口；位值：1：清除对应输出口段内未执行完的动作，比如翻转时间到达后 IO 翻转动作；0：不操作

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

注　意：该指令对 SMCContiDelayOutbitToStart、SMCContiDelayOutbitToStop、SMCContiAheadOutbitToStop 指令起作用。

4.15 PWM 立即输出指令

SMCSetPwmOutput

语　法：short SMCSetPwmOutput(WORD PwmNo,double fDuty, double fFre)

描　述：设置PWM输出信号的频率、占空比参数

参　数：PwmNo　PWM输出口编号，0或1

fDuty　PWM信号的占空比：0～1

fFre　PWM输出频率，单位：Hz，范围：1HZ-500KHZ

返回值：错误代码

适用范围：除SMC100系列控制器外的控制器

SMCGetPwmOutput

语　法：short SMCGetPwmOutput(WORD PwmNo,double* fDuty, double* fFre)

描　述：读取PWM输出信号的频率、占空比参数

参　数：PwmNo　PWM输出口编号，0或1

fDuty　返回PWM信号的占空比：0～1

fFre　返回PWM输出频率，单位：Hz，范围：1HZ-500KHZ

返回值：错误代码

适用范围：除SMC100系列控制器外的控制器

4.16 通用 IO 接口指令

SMCReadInbit

语　法：short SMCReadInbit(WORD bitno)

描　述：读取某个输入端口的电平

参　数：bitno 输入端口号，取值范围：0-控制器本机输入口数目-1

返回值：指定的输入端口电平：0：低电平，导通状态；1：高电平，断开状态

适用范围：全系列控制器

SMCWriteOutbit

语　法：short SMCWriteOutbit(WORD bitno,WORD on_off)

描　述：设置指定控制器的某个输出端口的电平

参　数：bitno 输出端口号，取值范围：0-控制器本机输出口数目-1

on_off 输出电平，0：低电平，1：高电平

返回值：错误代码

适用范围：全系列控制器

SMCReadOutbit

语　法：short SMCReadOutbit(WORD bitno)

描　述：读取某个输出端口的电平

参　数：bitno 输出端口号，取值范围：0-控制器本机输出口数目-1

返回值：指定输出端口的电平，0：低电平，1：高电平

适用范围：全系列控制器

SMCReadInport

语　法：DWORD SMCReadInport(WORD portno)

描　述：读取全部输入端口的电平

参　数：portno　　IO 组号，从 0 开始编号

返回值：各 IO 口的 bit 值

⚠注意：在 IO 口在 32 位内 PORT 号为 0.超出 32 位 PORT 为 1。返回值为 10 进制，转换为 2

进制后 bit 值对应各端口状态值

适用范围：全系列控制器

SMCWriteOutport

语　法：short SMCWriteOutport(WORD portno,DWORD port_value)

描　述：设置指定控制器的全部输出口的电平

参　数：portno　　IO 组号，从 0 开始编号

port_value　　输入端口电平数值

返回值：错误代码

适用范围：全系列控制器

SMCReadOutport

语　法：DWORD SMCReadOutport(WORD portno)

描　述：设置指定控制器的全部输出口的电平

参　数：portno　　IO 组号，从 0 开始编号

返回值：各 IO 口的 bit 值

适用范围：全系列控制器

⚠注意：返回值为 10 进制数，将其转换为 2 进制后各 bit 值为对应 OUT 口的状态值

SMCReverseOutbit

语　法：short SMCReverseOutbit(WORD bitno,double reverse_time)

描　述：IO 输出延时翻转

参　数：bitno　　输出端口号，取值范围：0-控制器本机输出口数-1

reverse_time　　延时翻转时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：1）该函数只能对 OUT0-各控制器端口数值，进行操作

2）当延时翻转时间参数设置为 0 时，此时延时翻转时间将为无限大

SMCSetIoCountMode

语　法：short SMCSetIoCountMode(WORD bitno,WORD mode,double filter_time)

描　述：设置 IO 计数模式；

参　数：bitno　　输入端口号，取值范围：0-控制器本机输入口数-1

mode　　IO 计数模式，0：禁用，1：上升沿计数，2：下降沿计数

filter_time　　滤波时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGetIoCountMode

语　法：short SMCGetIoCountMode(WORD bitno,WORD *mode,double* filter_time)

描　述：读取 IO 计数模式设置；

参　数：bitno　　输入端口号，取值范围：0-控制器本机输入口数-1

mode　　返回 IO 计数模式，0：禁用，1：上升沿计数，2：下降沿计数

filter_time　　返回滤波时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCSetIoCountValue

语　法：short SMCSetIoCountValue(WORD bitno,DWORD CountValue)

描　述：设置 IO 计数值；

参　数：bitno　　输入端口号，取值范围：0-控制器本机输入口数-1

CountValue　　IO 计数值

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGetIoCountValue

语　法：short SMCGetIoCountValue(WORD bitno,DWORD* CountValue)

描　述：读取 IO 计数值；

参　数：bitno　　输入端口号，取值范围：0-控制器本机输入口数-1

CountValue　　返回 IO 计数值

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

4.17 专用 IO 接口指令

SMCSetAlmMode

语　法：short SMCSetAlmMode(WORD axis,WORD enable,WORD alm_logic,WORD alm_action)

描　述：设置指定轴的 ALM 信号参数

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

enable　　ALM 信号使能，0：禁止，1：允许

alm_logic　　ALM 信号的有效电平，0：低有效，1：高有效

alm_action　　ALM 信号的制动方式，0：立即停止（只支持该方式）

返回值：错误代码

适用范围：脉冲型全系列控制器

⚠注意：该函数设置轴为 255 时，表示所有轴都将统一被设置相应的参数。

SMCGetAlmMode

语　法：short SMCGetAlmMode(WORD axis,WORD* enable,WORD* alm_logic,WORD* alm_action)

描　述：读取指定轴的 ALM 信号参数

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

enable　　返回 ALM 信号使能，0：禁止，1：允许

alm_logic　　返回 ALM 信号的有效电平，0：低有效，1：高有效

alm_action　　返回 ALM 信号的制动方式，0：立即停止（只支持该方式）

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCSetInpMode

语　法：short SMCSetInpMode(WORD axis,WORD enable,WORD inp_logic)

描　述：设置指定轴的 INP 信号

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

enable　　INP 信号使能，0：禁止，1：允许

inp_logic　　INP 信号的有效电平，0：低有效，1：高有效

返回值：错误代码

⚠注意：（1）若控制器本身没有 INP 借口，需通过 IO 映射来实现。

（2）该函数设置轴为 255 时，表示所有轴都将统一被设置相应的参数。

适用范围：脉冲型全系列控制器

SMCGetInpMode

语　法：short SMCGetInpMode(WORD axis,WORD* enable,WORD*inp_logic)

描　述：读取指定轴的 INP 信号

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

enable　　返回 INP 信号使能，0：禁止，1：允许

inp_logic　　返回 INP 信号的有效电平，0：低有效，1：高有效

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCSetEzMode

语　法：short SMCSetEzMode(WORD axis,WORD ez_logic,WORD ez_mode,double filter)

描　述：设置指定轴的 EZ 信号

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

ez_logicEZ 信号有效电平，0：低有效，1：高有效

ez_mode　　保留参数，固定值为 0

filter 保留参数，固定值为 0

返回值：错误代码

⚠注意：该函数设置轴为 255 时，表示所有轴都将统一被设置相应的参数。

适用范围：脉冲型全系列控制器

SMCGetEzMode

语　法：short SMCGetEzMode(WORD axis,WORD* ez_logic,WORD* ez_mode,double* filter)

描　述：读取指定轴的 EZ 信号

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

ez_logicEZ 返回信号有效电平，0：低有效，1：高有效

ez_mode　　返回保留参数，固定值为 0

filter 返回时间过滤参数，固定值为 0

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCWriteSevonPin

语　法：short　SMCWriteSevonPin(WORD axis,WORD on_off)

描　述：设置指定轴的伺服使能端口的输出

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

on_off　设置伺服使能端口电平，0：低电平，1：高电平

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCReadSevonPin

语　法：short　SMCReadSevonPin(WORD axis)

描　述：读取指定轴的伺服使能端口的电平

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：伺服使能端口电平，0：低电平，1：高电平

适用范围：SMC300、SMC600 系列控制器

SMCWriteErcPin

语　法：short　SMCWriteErcPin(WORD axis,WORD sel)

描　述：设置指定轴的 ERC 信号输出

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

sel　　输出电平，0：低电平，1：高电平

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCReadErcPin

语　法：short SMCReadErcPin(WORD axis)

描　述：读取指定轴的 ERC 端口电平

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：ERC 端口电平，0：低电平，1：高电平

适用范围：SMC300、SMC600 系列控制器

SMCReadAlarmPin

语　法：short SMCReadAlarmPin(WORD axis)

描　述：读取指定轴的 ALARM 端口电平

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：ALARM 端口电平，0：低电平，1：高电平

适用范围：SMC300、SMC600 系列控制器

SMCReadInpPin

语　法：short SMCReadInpPin(WORD axis)

描　述：读取指定轴的 INP 端口电平

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：INP 端口电平，0：低电平，1：高电平

适用范围：SMC300、SMC600 系列控制器

SMCReadOrgPin

语　法：short SMCReadOrgPin(WORD axis)

描　述：读取指定轴的 ORG 端口电平

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数-1

返回值：ORG 端口电平，0：低电平，1：高电平

适用范围：SMC300、SMC600 系列控制器

SMCReadElpPin

语　法：short SMCReadElpPin(WORD axis)

描　述：读取指定轴的正硬限位端口电平

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数-1

返回值：EL+端口电平，0：低电平，1：高电平

适用范围：SMC300、SMC600 系列控制器

SMCReadElnPin

语　法：short SMCReadElnPin(WORD axis)

描　述：读取指定轴的负硬限位端口电平

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数-1

返回值：EL-端口电平，0：低电平，1：高电平

适用范围：SMC300、SMC600 系列控制器

SMCReadEzPin

语　法：short SMCReadEzPin(WORD axis)

描　述：读取指定轴的 EZ 端口电平

参　数：axis 指定轴号，取值范围：0-控制器最大轴数-1

返回值：EZ 端口电平，0：低电平，1：高电平

适用范围：SMC300、SMC600 系列控制器

SMCReadEmgPin

语　法：short SMCReadEmgPin(WORD axis)

描　述：读取指定轴的急停端口电平

参　数：axis 指定轴号，取值范围：0-控制器最大轴数-1

返回值：EMG 端口电平，0：低电平，1：高电平

适用范围：SMC300、SMC600 系列控制器

4.18 电子凸轮指令

SMCCamTableUnit

语　法：short SMCCamTableUnit(WORD MasterAxisNo,WORD SlaveAxisNo,DWORD Count, double* MasterPos,double* SlavePos,WORD SrcMode)

描　述：添加电子凸轮表

参　数：MasterAxisNo　　主轴轴号

SlaveAxisNo　　从轴轴号

Count　　位置数据个数

MasterPos　　主轴位置数组，主轴位置必须递增或者递减变化

SlavePos　　从轴位置数组

SrcMode　　主轴位置模式：0-指令位置，1-反馈位置

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：跟随起点为启动从轴进入电子凸轮时刻的位置，跟随范围为跟随起点主轴位置加上凸轮表中主轴的最大（或者最小）位置值。

SMCCamMove

语　法:short SMCCamMove(WORD SlaveAxisNo)

描　述：启动从轴电子凸轮运动

参　数：SlaveAxisNo 从轴轴号

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

4.19 手轮指令

SMCHandwheelSetIndex

语　法：WORD SMCHandwheelSetIndex (WORD AxisSelIndex,WORD RatioSelIndex)

描　述：选择或更换手轮运动轴选、倍率档位

参　数：AxisSelIndex 轴选档位，取值范围：0-控制器最大轴数目-1

RatioSelIndex　返回倍率档位，取值范围：0、1、2，分别对应倍率 1、10、100

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：此指令用于启动手轮前选择档位和手轮运动过程中更换档位。一个轴选档位可对应多个运动轴（SMCHandwheelSetAxislist）。同一个轴选档位下各个运动轴倍率值是一样的，但可以修改倍率值（SMCHandwheelSetRatiolist）。

SMCHandwheelGetIndex

语　法：WORD SMCHandwheelGetIndex (WORD* AxisSelIndex,WORD* RatioSelIndex)

描　述：读取选择或更换手轮运动轴选、倍率档位

参　数：AxisSelIndex　返回轴选档位，取值范围：0-控制器最大轴数目-1

RatioSelIndex　返回倍率档位，取值范围：0、1、2，分别对应倍率 1、10、100

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCHandwheelSetAxislist

语　法：short SMCHandwheelSetAxislist(WORD AxisSelIndex,WORD AxisNum, WORD AxisList)

描　述：设置同一轴选档位下具体运动轴列表

参　数：AxisSelIndex　轴选档位，取值范围：0-控制器最大轴数目-1

AxisNum　运动轴数，取值范围：1-控制器最大轴数目

AxisList　轴号列表

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCHandwheelGetAxislist

语　法：short SMCHandwheelGetAxislist(WORD AxisSelIndex,WORD *AxisNum, WORD* AxisList)

描　述：读取同一轴选档位下具体运动轴列表

参　数：AxisSelIndex　轴选档位，取值范围：0-控制器最大轴数目-1

AxisNum　　返回运动轴数，取值范围：1-控制器最大轴数目

AxisList　　返回轴号

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCHandwheelSetRatiolist

语　法：short SMCHandwheelSetRatiolist(WORD AxisSelIndex, WORD StartRatioIndex, WORD RatioSelNum, double RatioList)

描　述：设置同一轴选下手轮倍率档位

参　数：AxisSelIndex　　轴选档位，取值范围：0-控制器最大轴数目-1

StartRatioIndex　倍选起始索引值，取值范围：0-2

RatioSelNum　　倍率值数目

RatioList　　倍率列表，取值范围：[1,100]

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：此指令默认参数 StartRatioIndex=0，RatioSelNum=3，数组 RatioList[3]为 3 个分别对应 1、10、100,。即同一轴选下，3 种倍率值都可以有。

SMCHandwheelGetRatiolist

语　法：short SMCHandwheelGetRatiolist(WORD AxisSelIndex, WORD StartRatioIndex, WORD RatioSelNum, double* RatioList)

描　述：读取同一轴选下手轮倍率档位

参　数：AxisSelIndex　　轴选档位，取值范围：0-控制器最大轴数目-1

StartRatioIndex　倍选起始索引值，取值范围：0-2

RatioSelNum 返回倍率值数目

RatioList 倍率列表，取值范围：[1,100]

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCHandwheelSetMode

语　法：short SMCHandwheelSetMode(WORD InMode, WORD IfHardEnable)

描　述：设置手轮运动模式，硬件、还是软件模式下运动

参　数：InMode 输入脉冲模式，0：脉冲+方向，1：AB 相脉冲

IfHardEnable 运行模式，0：软件控制，1：硬件控制

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：硬件模式下运动，需配置好控制器上的硬件连接线等。软件模式下则可不用。

SMCHandwheelGetMode

语　法：short SMCHandwheelGetMode(WORD* InMode, WORD* IfHardEnable)

描　述：读取手轮运动模式，硬件、还是软件模式下运动

参　数：InMode 返回输入脉冲模式，0：脉冲+方向，1：AB 相脉冲

IfHardEnable 返回运行模式，0：软件控制，1：硬件控制

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCHandwheelMove

语　法：WORD SMCHandwheelMove(WORD ForceMove)

描　述：启动手轮运动

参　数：ForceMove 参数保留，固定为 0

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCHandwheelStop

语　法：short SMCHandwheelStop(void)

描　述：停止手轮运动

参　数：无

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

4.20 编码器指令

SMCSetCounterInmode

语　法：short SMCSetCounterInmode(WORD axis,WORD mode)

描　述：设置编码器的计数方式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

Mode　设置编码器信号参数模式

0：非A/B相 (脉冲/方向)

1：A/B信号，1倍频

2：A/B信号，2倍频

3：A/B信号，4倍频

返回值：错误代码

适用范围：全系列控制器

⚠注意：编码器参数设置需与外部接入信号一致，否则可能计数不准或不计数。

SMCGetCounterInmode

语　法：short SMCGetCounterInmode(WORD axis,WORD* mode)

描　述：读取编码器的计数方式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

Mode　返回设置编码器信号参数模式

0：非A/B相 (脉冲/方向)

1：A/B信号，1倍频

2：A/B信号，2倍频

3：A/B信号，4倍频

返回值：错误代码

适用范围：全系列控制器

SMCSetEncoderUnit

语　法：short SMCSetEncoderUnit(WORD axis,double dist)

描　述：设置编码器计数值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

Dist　位置值，单位：unit

返回值：错误代码

适用范围：全系列控制器

SMCGetEncoderUnit

语　法：short SMCGetEncoderUnit(WORD axis,double* dist)

描　述：读取编码器计数值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

Dist　返回位置值，单位：unit

返回值：错误代码

适用范围：全系列控制器

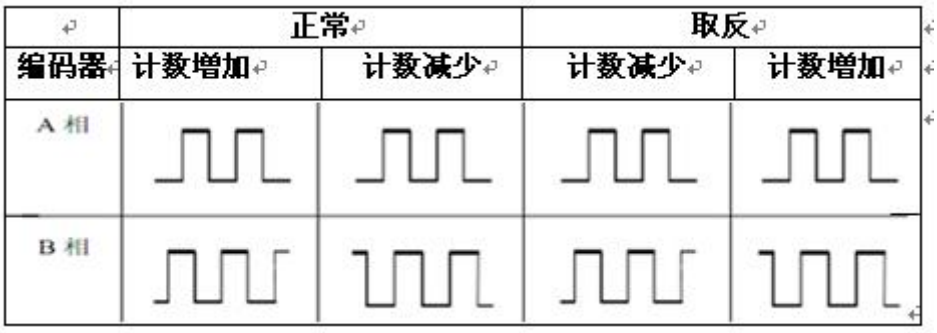
SMCSetCounterReverse

语　法：short SMCSetCounterReverse(WORD axis,WORD reverse);

描　述：设置 AB 相计数反相

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

reverse　0-A 超前 B 增计数，1-B 超前 A 增计数



返回值：错误代码

适用范围：全系列控制器

⚠注意：该函数默认为模式 0.在编码器硬件连接线不变的情况下，我们将模式 0 变为 1，可使其编码计数为负值（取反）。

SMCGetCounterReverse

语　法：short SMCGetCounterReverse(WORD axis,WORD *reverse);

描　述：读取 AB 相计数反相值

参　数：　axis　　指定轴号，取值范围：0-控制器最大轴数-1

reverse　　0-A 超前 B 增计数，1-B 超前 A 增计数

返回值：错误代码

适用范围：全系列控制器

4.21 高速位置锁存指令

SMCSetLtcMode

语　法：short SMCSetLtcMode(WORD axis,WORD ltc_logic,WORD ltc_mode,double filter)

描　述：设置指定轴的 LTC 信号

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

ltc_logic LTC 信号的触发方式，0：下降沿锁存，1：上升沿锁存

ltc_mode　固定值为 0

filter　　保留参数，固定值为 0

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGetLtcMode

语　法：short SMCGetLtcMode(WORD axis,WORD* ltc_logic,WORD* ltc_mode,double* filter)

描　述：读取指定轴的 LTC 信号参数

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

ltc_logic　　返回 LTC 信号的触发方式，0：下降沿锁存，1：上升沿锁存

ltc_mode　　固定值 0

filter　　保留参数，固定值为 0

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCSetLatchMode

语　法：short SMCSetLatchMode(WORD axis,WORDltc_mode,WORD latch_source,

WORD triger_chunnel)

描　述：设置锁存方式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

ltc_mode　设置锁存方式：

0：单次锁存

2：连续锁存

latch_source　锁存源，0：指令位置计数器，1：编码器计数器

triger_chunnel　触发通道，固定值 0

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGetLatchMode

语　法：short SMCGetLatchMode(WORD axis, WORD* ltc_mode,WORD* latch_source, WORD* triger_chunnel)

描　述：读取锁存方式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

ltc_mode　返回锁存方式：

0：单次锁存

2：连续锁存

latch_source　返回锁存源，0：指令位置计数器，1：编码器计数器

triger_chunnel　触发通道，固定值 0

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGetLatchValueUnit

语　法：doubleSMCGetLatchValueUnit(WORD axis)

描　述：从控制器内读取锁存器的值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：锁存值,单位：unit

适用范围：SMC300、SMC600 系列控制器

⚠注意：当选择锁存方式为连续锁存时，用此函数成功读取锁存值后，锁存次数会减 1 并剔除该锁存值

SMCGetLatchFlag

语　法：short SMCGetLatchFlag(WORD axis)

描　述：从控制器内读取指定轴的锁存次数

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：有效锁存次数

适用范围：SMC300、SMC600 系列控制器

SMCResetLatchFlag

语　法：short SMCResetLatchFlag(WORD axis)

描　述：复位锁存器的标志位

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：当使用锁存功能前，必须先调用此函数复位锁存器的标志位

4.22 原点锁存指令

SMCSetHomelatchMode

语　法：short SMCSetHomelatchMode(WORD axis,WORD enable,WORD logic,
WORD source)

描　述：设置原点锁存模式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

Enable 原点锁存使能，0：禁止，1：允许

Logic　触发方式，0：下降沿，1：上升沿

Source 位置源选择，0：指令位置计数器，1：编码器计数器

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGetHomelatchMode

语　法：short SMCGetHomelatchMode(WORD axis,WORD* enable,WORD* logic,WORD*
source)

描　述：读取原点锁存模式参数

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

enable　　返回原点锁存使能，0：禁止，1：允许

logic　　返回触发方式，0：下降沿，1：上升沿

source　　返回位置源选择，0：指令位置计数器，1：编码器计数器

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCResetHomelatchFlag

语　法：short SMCResetHomelatchFlag(WORD axis)

描　述：清除原点锁存标志

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGetHomelatchFlag

语　法：long SMCGetHomelatchFlag(WORD axis)

描　述：读取原点锁存标志

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：原点锁存标志，0：未锁存，1：锁存

适用范围：SMC300、SMC600 系列控制器

SMCGetHomelatchValueUnit

语　法：doubleSMCGetHomelatchValueUnit(WORD axis)

描　述：读取原点锁存值

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：锁存值，单位：unit

适用范围：SMC300、SMC600 系列控制器

4.23 EZ 锁存指令

SMCSetEzlatchMode

语　法：short SMCSetEzlatchMode(WORD axis,WORD enable,WORD logic,WORD source)

描　述：设置 EZ 锁存模式

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

Enable EZ 锁存使能，0：禁止，1：允许

Logic　触发方式，0：下降沿，1：上升沿

Source 位置源选择，0：指令位置计数器，1：编码器计数器

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGetEzlatchMode

语　法：short SMCGetEzlatchMode(WORD axis,WORD* enable,WORD* logic,

WORD* source)

描　述：读取 EZ 锁存模式参数

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

enable　返回 EZ 锁存使能，0：禁止，1：允许

logic　返回触发方式，0：下降沿，1：上升沿

source　返回位置源选择，0：指令位置计数器，1：编码器计数器

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCResetEzlatchFlag

语　法：short SMCResetEzlatchFlag(WORD axis)

描　述：清除 EZ 锁存标志

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGetEzlatchFlag

语　法：long SMCGetEzlatchFlag(WORD axis)

描　述：读取 EZ 锁存标志

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：EZ 锁存标志，0：未锁存，1：锁存

适用范围：SMC300、SMC600 系列控制器

SMCGetEzlatchValueUnit

语　法：double SMCGetEzlatchValueUnit(WORD axis)

描　述：读取 EZ 锁存值

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：锁存值，单位：unit

适用范围：SMC300、SMC600 系列控制器

4.24 位置比较指令

SMCCompareSetConfig

语　法：short SMCCompareSetConfig(WORD axis,WORD enable, WORD cmp_source)

描　述：设置一维位置比较器

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

Enable 比较功能状态，0：禁止，1：使能

cmp_source 比较源，0：指令位置计数器，1：编码器计数器

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCCompareGetConfig

语　法：short SMCCompareGetConfig(WORD axis,WORD* enable, WORD* cmp_source)

描　述：读取一维位置比较器参数

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

enable　返回比较功能状态，0：禁止，1：使能

cmp_source　返回比较源，0：指令位置计数器，1：编码器计数器

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCCompareClearPoints

语　法：short SMCCompareClearPoints(WORD axis)

描　述：清除已添加的所有一维位置比较点

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCCompareAddPointUnit

语　法：short SMCCompareAddPointUnit(WORD axis, double pos, WORD dir, WORD action, DWORD actpara)

描　述：添加一维位置比较点

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

Pos 比较位置，单位：unit

dir　比较模式，0：小于等于，1：大于等于

Action 比较点触发功能编号，见下表

actpara 比较点触发功能参数，见下表

返回值：错误代码

action	actpara	功能
1	IO 号	IO 置为高电平
2	IO 号	IO 置为低电平
3	IO 号	取反 IO
5	IO 号	输出 500us 脉冲
6	IO 号	输出 1ms 脉冲
7	IO 号	输出 10ms 脉冲
8	IO 号	输出 100ms 脉冲
13	轴号	停止指定轴

适用范围：SMC300、SMC600 系列控制器

SMCCompareGetCurState

语　法：short SMCCompareGetCurState(WORD axis,long *remained_points,double *current_point,long *runned_points)

描　述：读取比较状态

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

remained_points　返回可添加比较点数,最大点数 256

current_point　返回当前比较点位置，单位：unit

runned_points　返回已比较点数

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCHcmpSetMode

语　法：short SMCHcmpSetMode(WORD hcmp, WORD cmp_mode)

描　述：设置高速比较模式

参　数：hcmp　高速比较器，取值范围：0-1（对应控制器最后两个输出端口）

cmp_mode　比较模式：

0：禁止（默认值）

1：等于

2：小于

3：大于

4：队列，提供 127 个点比较空间，采用先添加先比较，比较完可追加比较点，也可一次性添加多个比较点

5：线性，提供起始比较点，位置增量，比较次数

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：1）当选择模式 1 时，只有当前位置等于比较位置时，CMP 端口才输出有效电平

2）当选择模式 2 时，只要当前位置小于比较位置时，CMP 端口就一直保持有效电平

3）当选择模式 3 时，只要当前位置大于比较位置时，CMP 端口就一直保持有效电平

4）当选择模式 4 或 5 时，CMP 端口输出有效电平的时间通过 SMCHcmpSetConfig 函数的 time 参数（脉冲宽度）设置

SMCHcmpGetMode

语　法：short SMCHcmpSetMode(WORD hcmp, WORD* cmp_mode)

描　述：读取高速比较模式参数

参　数：hcmp　　高速比较器，取值范围：0-1（对应控制器最后两个输出端口）

cmp_mode　返回比较模式：

0：禁止（默认值）

1：等于

2：小于

3：大于

4：队列，提供 127 个点比较空间，采用先添加先比较，比较完可追加比较点，也可一次性添加多个比较点

5：线性，提供起始比较点，位置增量，比较次数

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCHcmpSetConfig

语　法：short SMCHcmpSetConfig(WORD hcmp,WORD axis, WORD cmp_source, WORD cmp_logic,long time)

描　述：能配置高速比较器

参　数：hcmp　　高速比较器，取值范围：0-1（对应控制器最后两个输出端口）

axis　指定轴号，取值范围：0-控制器最大轴数目-1

cmp_source　比较位置源：0：指令位置计数器，1：编码器计数器

cmp_logic　有效电平：0：低电平，1：高电平

time　　脉冲宽度，单位：us，取值范围：1us~20s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：该函数的 time 参数（脉冲宽度）只对队列和线性比较模式起作用

SMCHcmpGetConfig

语　法：short SMCHcmpGetConfig(WORD hcmp,WORD axis, WORD* cmp_source, WORD* cmp_logic,long* time)

描　述：读取高速比较器参数

参　数：hcmp　　高速比较器，取值范围：0-1（对应控制器最后两个输出端口）
　　axis　　指定轴号，取值范围：0-控制器最大轴数目-1
　　cmp_source　返回比较位置源：0：指令位置计数器，1：编码器计数器
　　cmp_logic　返回有效电平：0：低电平，1：高电平
　　time　　返回脉冲宽度，单位：us，取值范围：1us~20s
返回值：错误代码
适用范围：SMC300、SMC600 系列控制器

SMCHcmpClearPoints
语　法：short SMCHcmpClearPoints(WORD hcmp)
描　述：清除已添加的所有高速位置比较点
参　数：hcmp　　高速比较器，取值范围：0-1（对应控制器最后两个输出端口）
返回值：错误代码
适用范围：SMC300、SMC600 系列控制器

SMCHcmpAddPointUnit
语　法：short SMCHcmpAddPointUnit(WORD hcmp, double cmp_pos)
描　述：添加/更新高速比较位置
参　数：hcmp　　高速比较器，取值范围：0-1（对应控制器最后两个输出端口）
　　cmp_pos　　队列模式下：添加比较位置，单位：unit
　　线性模式下：更新起始比较位置，单位：unit
　　其他模式下：更新比较位置，单位：unit
返回值：错误代码
适用范围：SMC300、SMC600 系列控制器

⚠注意：执行线性比较时该指令作为设置比较起始位置的同时也是启动命令，需在配置完比较器后调用执行。

SMCHcmpSetLinerUnit
语　法：short SMCHcmpSetLinerUnit(WORD hcmp, double Increment,long Count)

描　述：设置高速比较线性模式参数

参　数：hcmp　　高速比较器，取值范围：0-1（对应控制器最后两个输出端口）

Increment　位置增量值，单位：unit（正值表示位置递增，负值表示位置递减）

Count　　比较次数，取值范围：1~65535

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCHcmpGetLinerUnit

语　法：short SMCHcmpGetLinerUnit(WORD hcmp, double* Increment,long* Count)

描　述：读取高速比较线性模式参数

参　数：hcmp　　高速比较器，取值范围：0-1（对应控制器最后两个输出端口）

Increment　返回位置增量值，单位：unit（正值表示位置递增，负值表示位置递减）

Count　　返回比较次数，取值范围：1~65535

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCHcmpGetCurState

语　法：short SMCHcmpGetCurState(WORD hcmp,long *remained_points,double *current_point,long *runned_points)

描　述：读取高速比较状态

参　数：hcmp　　高速比较器，取值范围：0-1（对应控制器最后两个输出端口）

remained_points　返回可添加比较点数

current_point　返回当前比较点位置，单位：unit

runned_points　返回已比较点数

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

short SMCCompareSetConfigExtern(WORD enable, WORD cmp_source)

功　能：设置二维位置比较器

参　数：　ConnectNo　指定链接号：0-7,默认值0

enable　二维位置比较功能状态，0：禁止，1：使能

cmp_source　二维位置比较源，0：指令位置计数器，1：编码器计数器

返回值：错误代码

适用范围：SMC300、SMC600系列控制器

short SMCCompareGetConfigExtern(WORD* enable,WORD* cmp_source)

功　能：读取二维位置比较器设置

参　数：　ConnectNo　指定链接号：0-7,默认值0

Enable　返回比较功能状态

cmp_source　返回比较源

返回值：错误代码

适用范围：SMC300、SMC600系列控制器

short SMCCompareClearPointsExtern(WORD ConnectNo)

功　能：清除已添加的所有二维位置比较点

参　数：无

返回值：错误代码

适用范围：SMC300、SMC600系列控制器

short SMCCompareAddPointExternUnit(WORD* axis, double* pos, WORD* dir, WORD action, DWORD actpara)

功　能：添加二维位置比较点

参　数：　ConnectNo　指定链接号：0-7,默认值0

axis　指定控制器上的即将进行位置比较的轴列表(两个轴)

pos　二维位置比较位置列表，单位：unit

dir　比较模式列表，0：小于等于，1：大于等于

action　二维位置比较点触发功能编号，见表3.6

actpara　二维位置比较点触发功能参数，见表3.6

返回值：错误代码

适用范围：SMC300、SMC600系列控制器

表3.6 SMCCompareAddPointExternUnit 函数action, actpara 参数值

Action	actpara	功能
1	IO号	IO置为高电平
2	IO号	IO置为低电平
3	IO号	取反IO
5	IO号	输出500us 脉冲
6	IO号	输出1ms 脉冲
7	IO号	输出10ms 脉冲
8	IO号	输出100ms 脉冲
13	轴号	停止指定轴

short SMCCompareGetCurrentPointExternUnit(double *pos)

功　能：读取当前二维位置比较点位置

参　数：　　ConnectNo　指定链接号：0-7,默认值0

　　pos　　返回当前二维位置比较点位置，单位：unit

返回值：错误代码

适用范围：SMC300、SMC600系列控制器

short SMCCompareGetPointRunnedExtern(long *PointNum)

功　能：查询已经比较过的二维比较点个数

参　数：　　ConnectNo　指定链接号：0-7,默认值0

　　PointNum　返回已经比较过的二维位置比较点数

返回值：错误代码

适用范围：SMC300、SMC600系列控制器

short SMCCompareGetPointsRemainedExtern(WORD ConnectNo,long *PointNum)

功　能：查询可以加入的二维比较点个数

参　数：　　ConnectNo　指定链接号：0-7,默认值0

　　PointNum　返回可以加入的二维位置比较点数

返回值：错误代码

适用范围：SMC300、SMC600系列控制器

4.25 软硬件限位指令

SMCSetElMode

语　法：short SMCSetElMode(WORD axis,WORD el_enable,WORD el_logic,WORD el_mode)

描　述：设置 EL 限位(硬件限位)信号参数

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

　　el_enable　　EL 信号的使能状态：

　　0：正负限位禁止

　　1：正负限位允许

　　2：正限位禁止、负限位允许

　　3：正限位允许、负限位禁止

　　el_logic EL 信号的有效电平：

　　0：正负限位低电平有效

　　1：正负限位高电平有效

　　2：正限位低有效，负限位高有效

　　3：正限位高有效，负限位低有效

　　el_modeEL 制动方式：

　　0：正负限位立即停止

　　1：正负限位减速停止

　　2：正限位立即停止，负限位减速停止

　　3：正限位减速停止，负限位立即停止

返回值：错误代码

适用范围：脉冲型全系列控制器

⚠注意：该函数设置轴为 255 时，表示所有轴都将统一被设置相应的参数。

SMCGetElMode

语　　法：short SMCGetElMode(WORD axis,WORD* el_enable,WORD* el_logic,WORD* el_mode)

描　述：读取 EL 限位(硬件限位)信号设置参数

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数-1

el_enable　　返回 EL 信号的使能状态：

0：正负限位禁止

1：正负限位允许

2：正限位禁止、负限位允许

3：正限位允许、负限位禁止

el_logic 返回 EL 信号的有效电平：

0：正负限位低电平有效

1：正负限位高电平有效

2：正限位低有效，负限位高有效

3：正限位高有效，负限位低有效

el_mode返回 EL 制动方式：

0：正负限位立即停止

1：正负限位减速停止

2：正限位立即停止，负限位减速停止

3：正限位减速停止，负限位立即停止

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCSetSoftlimitUnit

语　　法：short SMCSetSoftlimitUnit(WORD axis,WORD enable, WORD source_sel,WORD SL_action, doubel N_limit, double P_limit)

描　述：设置软限位

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

Enable 使能状态，0：禁止，1：允许

source_sel 计数器选择，0：指令位置计数器，1：编码器计数器

SL_action 限位停止方式，0：立即停止，1：减速停止

N_limit 负限位位置，单位：unit

P_limit 正限位位置，单位：unit

返回值：错误代码

适用范围：脉冲型全系列控制器

⚠注意：正、负限位位置可为正数也可为负数，但正限位位置应大于负限位位置

SMCGetSoftlimitUnit

语　法：short SMCGetSoftlimitUnit(WORD axis,WORD* enable, WORD* source_sel,WORD* SL_action, double* N_limit,double* P_limit)

描　述：读取软限位参数

参　数：axis　指定轴号，取值范围：0-控制器最大轴数-1

Enable　返回使能状态，0：禁止，1：允许

source_sel　返回计数器选择，0：指令位置计数器，1：编码器计数器

SL_action　返回限位停止方式，0：立即停止，1：减速停止

N_limit　返回负限位位置，单位：unit

P_limit　返回正限位位置，单位：unit

返回值：错误代码

适用范围：脉冲型全系列控制器

4.26 运动异常停止指令

SMCSetEmgMode

语　法：short SMCSetEmgMode(WORD axis,WORD enable,WORD emg_logic)

描　述：设置 EMG 急停信号参数

参　数：axis　指定轴号，取值范围：0-控制器最大轴数目-1

（SMC104：保留参数，固定值为 0（2 轴 4/轴共用 EMG 信号））

Enable 允许/禁止信号功能，0：禁止，1：允许

emg_logicEMG 信号有效电平，0：低有效，1：高有效

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCGetEmgMode

语　法：short SMCGetEmgMode(WORD axis,WORD* enable,WORD* emg_logic)

描　述：读取 EMG 急停信号参数

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

（SMC104：保留参数，固定值为 0（2 轴/4 轴共用 EMG 信号））

Enable　返回允许/禁止信号功能，0：禁止，1：允许

emg_logic 返回 EMG 信号有效电平，0：低有效，1：高有效

返回值：错误代码

适用范围：脉冲型全系列控制器

SMCSetIoDstpMode

语　法：short SMCSetIoDstpMode(WORD axis,WORD enable,WORD logic)

描　述：设置 IO 触发减速停止电平信号；

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数-1

enable　　允许/禁止硬件信号功能，0：禁止，1：允许

logic　　外部减速停止信号有效电平，0：低有效，1：高有效

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：（1）减速停止信号（DSTP）的减速时间由函数 SMCSetDecStopTime 设置

（2）该函数轴号设为 255 时，所有轴停止信号参数都被设置。

SMCGetIoDstpMode

语　法：short SMCGetIoDstpMode(WORD axis,WORD *enable,WORD *logic)

描　述：读取 IO 减速停止电平信号

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数-1

enable　　返回 DSTP 硬件信号功能状态

logic　　返回设置的外部减速停止信号有效电平

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCSetDecStopTime

语　法：short SMCSetDecStopTime(WORD axis,double stop_time)

描　述：设置减速停止时间；

参　数：axis　　指定轴号，取值范围：　0-控制器最大轴数-1

　　stop_time　　减速时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：当发生异常停止时，如：调用 SMCStop 函数、限位信号（软硬件）被触发、减速停止信号(DSTP)被触发等进行减速停止时，减速停止时间都为 SMCSetDecStopTime 函数里设置的减速时间

SMCGetDecStopTime

语　法：short SMCGetDecStopTime(WORD axis,double *stop_time)

描　述：读取减速停止时间设置；

参　数：axis　　指定轴号，取值范围：　0-控制器最大轴数-1

　　stop_time　　返回设置的减速时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCSetVectorDecStopTime

语　法：short SMCSetVectorDecStopTime(WORD Crd,double stop_time)

描　述：设置减速停止时间；

参　数：Crd　指定插补系，取值范围：　0-1

　　stop_time　　减速时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：当发生异常停止时，如：调用 SMCStopMulticoor 函数、限位信号（软硬件）被触发、减速停止信号(DSTP)被触发等进行减速停止时，减速停止时间都为 SMCSetVectorDecStopTime 函数里设置的减速时间

SMCGetVectorDecStopTime

语　法：short SMCGetVectorDecStopTime(WORD Crd,double *stop_time)

描　述：读取减速停止时间设置；

参　数：Crd　指定插补系，取值范围：　0-1

stop_time　　返回设置的减速时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGetStopReason

语　法：short SMCGetStopReason(WORD axis,long* stop_reason)

描　述：轴异常停止原因

参　数：axis　　指定轴号，取值范围：0~控制器最大轴数-1

stop_reason：返回当前轴异常停止原因,具体见以下：

0：正常停止

1：ALM 立即停止，IMD_STOP_AT_ALM

2：ALM 减速停止，DEC_STOP_AT_ALM

3：LTC 外部触发立即停止，IMD_STOP_AT_LTC

4：EMG 立即停止，IMD_STOP_AT_EMG

5：正硬限位立即停止，IMD_STOP_AT_ELP

6：负硬限位立即停止，IMD_STOP_AT_ELN

7：正硬限位减速停止，DEC_STOP_AT_ELP

8：负硬限位减速停止，DEC_STOP_AT_ELN

9：正软限位立即停止，IMD_STOP_AT_SOFT_ELP

10：负软限位立即停止，IMD_STOP_AT_SOFT_ELN

11：正软限位减速停止，DEC_STOP_AT_SOFT_ELP

12：负软限位减速停止，DEC_STOP_AT_SOFT_ELN

13：命令立即停止，IMD_STOP_AT_CMD

14：命令减速停止，DEC_STOP_AT_CMD

15：其他原因，　IMD_STOP_AT_OTHER

16：其他原因减速停止，DEC_STOP_AT_OTHER

17：未知原因立即停止，IMD_STOP_AT_UNKOWN

18：未知原因减速停止，DEC_STOP_AT_UNKOWN

19：减速阶段减速停止，DEC_STOP_AT_DEC

20:保留

21：在全程内未找到原点信号停止，原点不在两个限位之间，HOME_STOP_NO_HOME_SIGNAL

22：回零方向和当前有效限位端相反（正限位有效时，回零方向为负方向；负限位有效时，回零方向为正方向），HOME_STOP_HOME_DIR

23：正负限位同时有效，不能启动回零，HOME_STOP_HOME_EL

24：全程未找到EZ信号停止，HOME_STOP_NO_EZ_SIGNAL

25：回零位置溢出停止HOME_STOP_EXCEED_POS

26：运动加速度过大，异常停止，IMD_STOP_AT_SPEED_JIT

27：双原点停止，可能由干扰引起，HOME_STOP_DOUBLE_HOME

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCClearStopReason

语　法：short SMCClearStopReason(WORD axis)

描　述：清除当前轴错误原因

参　数：axis　　指定轴号，取值范围：0~控制器最大轴数-1

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

4.27 轴 IO 映射

SMCSetAxisIoMap

语　法：short SMCSetAxisIoMap(WORD Axis,WORD IoType,WORD MapIoType,WORD MapIoIndex,double filter_time)

描　述：设置轴 IO 映射关系

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

IoType　　指定轴需映射的功能：

0：正限位信号，AxisIoInMsg_PEL

1：负限位信号，AxisIoInMsg_NEL

2：原点信号，AxisIoInMsg_ORG

3：急停信号，AxisIoInMsg_EMG

4：减速停止信号，AxisIoInMsg_DSTP（保留）

5：伺服报警信号，AxisIoInMsg_ALM

6：伺服准备信号，AxisIoInMsg_RDY（保留）

7：伺服到位信号，AxisIoInMsg_INP

MapIoType　轴 IO 映射端口：

0：正限位输入端口，AxisIoInPort_PEL

1：负限位输入端口，AxisIoInPort_NEL

2：原点输入端口，AxisIoInPort_ORG

3：伺服报警输入端口，AxisIoInPort_ALM

4：伺服准备输入端口，AxisIoInPort_RDY

5：伺服到位输入端口，AxisIoInPort_INP

6：通用输入端口，AxisIoInPort_IO

7：急停输入端口，xisIoInMsg_EMG（SMC306A)

MapIoIndex　轴 IO 映射索引号：

1）当轴 IO 映射类型设置为 6 时，此参数可设置为 0~23 整数，表示该映射对应的具体通用输入端口号

2）当轴 IO 映射类型设置为 0-5 时，此参数可设置 0-5 整数，表示该映射所对应的具体轴号

filter_time 轴 IO 信号滤波时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：该函数可以实现对专用 IO 信号的硬件输入接口进行任意配置，不支持控制器 SMC104。

例程 1：

```
'IO 输入口 1 电平变化后将会触发轴 0 的急停信号，运动将停止
SMCSetAxisIoMap(0,3,6,1,0)
```

例程 2：

```
'当轴 3 的正限位输入端有效后，轴 0 的回原点信号将触发
SMCSetAxisIoMap(0,2,0,3,0)
```

SMCGetAxisIoMap

语　法：short SMCGetAxisIoMap(WORD Axis,WORD IoType,WORD* MapIoType,WORD* MapIoIndex,double* filter_time)

描　述：读取轴 IO 映射关系；

参　数：axis　　指定轴号，取值范围：0-控制器最大轴数目-1

IoType　　指定轴的 IO 信号类型：

0：正限位信号，AxisIoInMsg_PEL

1：负限位信号，AxisIoInMsg_NEL

2：原点信号，AxisIoInMsg_ORG

3：急停信号，AxisIoInMsg_EMG

4：减速停止信号，AxisIoInMsg_DSTP（保留）

5：伺服报警信号，AxisIoInMsg_ALM

6：伺服准备信号，AxisIoInMsg_RDY（保留）

7：伺服到位信号，AxisIoInMsg_INP

MapIoType　　返回轴 IO 映射类型：

0：正限位输入端口，AxisIoInPort_PEL

1：负限位输入端口，AxisIoInPort_NEL

2：原点输入端口，AxisIoInPort_ORG

3：伺服报警输入端口，AxisIoInPort_ALM

4：伺服准备输入端口，AxisIoInPort_RDY

5：伺服到位输入端口，AxisIoInPort_INP

6：通用输入端口，AxisIoInPort_IO

MapIoIndex　　返回轴 IO 映射索引号：

1）当轴 IO 映射类型设置为 6 时，此参数范围可为 0-各控制器最大输入 IO 数-1，表示该映射对应的具体通用输入端口号

2）当轴 IO 映射类型设置为 0-5 时，此参数可设置 0-5 整数，表示

该映射所对应的具体轴号

filter_time　　返回轴 IO 信号滤波时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

4.28 虚拟轴 IO 映射

SMCSetIoMapVirtual

语　法：short SMCSetIoMapVirtual(WORD bitno,WORD MapIoType,WORDMapIoIndex,
double filter_time)

描　述：设置虚拟 IO 映射关系

参　数：bitno　　虚拟 IO 口号

MapIoType　　虚拟 IO 映射类型：

0：正限位输入端口，AxisIoInPort_PEL

1：负限位输入端口，AxisIoInPort_NEL

2：原点输入端口，AxisIoInPort_ORG

3：伺服报警输入端口，AxisIoInPort_ALM

4：伺服准备输入端口，AxisIoInPort_RDY

5：伺服到位输入端口，AxisIoInPort_INP

6：通用输入端口，AxisIoInPort_IO

MapIoIndex　　虚拟 IO 映射索引号：

1）当虚拟 IO 映射类型设置为 6 时，此参数可取值范围：0~控制器最大输入 IO 口数-1，表示该映射对应的具体通用输入端口号

2）当虚拟 IO 映射类型设置为 0-5 时，此参数可设置 0-5 整数，表示该映射所对应的具体轴号

filter_time　　虚拟 IO 信号滤波时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：该函数可以实现专用通用 IO 输入接口的滤波功能

SMCGetIoMapVirtual

语　法：short SMCSetIoMapVirtual(WORD bitno,WORD* MapIoType,WORD* MapIoIndex,double* filter_time)

描　述：读取虚拟 IO 映射关系；

参　数：bitno　　虚拟 IO 口号

MapIoType　　返回虚拟 IO 映射类型：

0：正限位输入端口，AxisIoInPort_PEL

1：负限位输入端口，AxisIoInPort_NEL

2：原点输入端口，AxisIoInPort_ORG

3：伺服报警输入端口，AxisIoInPort_ALM

4：伺服准备输入端口，AxisIoInPort_RDY

5：伺服到位输入端口，AxisIoInPort_INP

6：通用输入端口，AxisIoInPort_IO

MapIoIndex　　返回虚拟 IO 映射索引号：

1）当虚拟 IO 映射类型设置为 6 时，此参数可取值范围：0~控制器最大输入 IO 口数-1，表示该映射对应的具体通用输入端口号

2）当虚拟 IO 映射类型设置为 0-5 时，此参数可设置 0-5 整数，表示该映射所对应的具体轴号

filter_time　　返回虚拟 IO 信号滤波时间，单位：s

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCReadInbitVirtual

语　法：short SMCReadInbitVirtual(WORD bitno)

描　述：读取虚拟电平状态；

参　数：bitno　　虚拟 IO 口号，取值范围：各控制器最大输入 IO 数-1

适用范围：SMC300、SMC600 系列控制器

4.29 密码管理指令

SMCEnterPassword

语　法：void SMCEnterPassword(const char* password)

描　述：输入密码，主要有 BASIC 上传、恢复默认参数值

参　数：password 密码值，默认值为 123456

返回值：无

适用范围：全系列控制器

例　程：SMCEnterPassword(“123456”)　'输入密码

⚠注意：一般在BASIC上传、恢复默认参数值时需输入密码

SMCModifyPassword

语　法：void SMCModifyPassword(const char* password)

描　述：修改密码

参　数：password 新密码

返回值：无

适用范围：全系列控制器

⚠注意：修改密码后，若需再次修改需重新启动控制器

4.30 寄存器操作指令

雷赛BASIC系列控制器在内存中定义了2个与MODBUS协议外设通讯的数据区，MODBUS位寄存器BIT和MODBUS字寄存器REG。位寄存器BIT有10000位，寄存器REG有10000个字。它们的地址分配如下：

BIT00000～BIT09999：客户自定义区

BIT10000～BIT10299：数字输入端口电平。从0号端口开始按顺序排列,0-断开（高电平），1-导通（低电平）

BIT10300～BIT10399：负向限位输入信号状态。0-正常，1-报警

BIT10400～BIT10499：正向限位输入信号状态。0-正常，1-报警

BIT10500～BIT10599：HOME输入信号状态。0-正常，1-报警

BIT10600～BIT10699：ALARM输入信号状态。0-正常，1-报警

BIT11000～BIT11399：数字输出端口电平。从0号端口开始按顺序排列,0-断开（高电平），1-导通（低电平）

REG00000～REG09999：客户自定义区

REG10000～REG10001：0 号轴当前位置，单位：pulse，类型：int32

REG10002～REG10003：1 号轴当前位置，单位：pulse，类型：int32

REG10004～REG10005：2 号轴当前位置，单位：pulse，类型：int32

REG10006～REG10007：3 号轴当前位置，单位：pulse，类型：int32

REG10008～REG10009：4 号轴当前位置，单位：pulse，类型：int32

REG10010～REG10011：5 号轴当前位置，单位：pulse，类型：int32

REG10100～REG10101：0 号轴当前位置，单位：unit，类型：float

REG10102～REG10103：1 号轴当前位置，单位：unit，类型：float

REG10104～REG10105：2 号轴当前位置，单位：unit，类型：float

REG10106～REG10107：3 号轴当前位置，单位：unit，类型：float

REG10108～REG10109：4 号轴当前位置，单位：unit，类型：float

REG10110～REG10111：5 号轴当前位置，单位：unit，类型：float

REG10200～REG10201：0 号轴当前速度，单位：pulse/s，类型：int32

REG10202～REG10203：1 号轴当前速度，单位：pulse/s，类型：int32

REG10204～REG10205：2 号轴当前速度，单位：pulse/s，类型：int32

REG10206～REG10207：3 号轴当前速度，单位：pulse/s，类型：int32

REG10208～REG10209：4 号轴当前速度，单位：pulse/s，类型：int32

REG10210～REG10211：5 号轴当前速度，单位：pulse/s，类型：int32

REG10300～REG10301：0 号轴当前速度，单位：unit/s，类型：float

REG10302～REG10303：1 号轴当前速度，单位：unit/s，类型：float

REG10304～REG10305：2 号轴当前速度，单位：unit/s，类型：float

REG10306～REG10307：3 号轴当前速度，单位：unit/s，类型：float

REG10308～REG10309：4 号轴当前速度，单位：unit/s，类型：float

REG10310～REG10311：5 号轴当前速度，单位：unit/s，类型：float

REG10400～REG10401：0 号编码器当前位置，单位：pulse，类型：int32

REG10402～REG10403：1 号编码器当前位置，单位：pulse，类型：int32

REG10404～REG10405：2 号编码器当前位置，单位：pulse，类型：int32

REG10406～REG10407：3 号编码器当前位置，单位：pulse，类型：int32

REG10408～REG10409：4 号编码器当前位置，单位：pulse，类型：int32

REG10410～REG10411：5 号编码器当前位置，单位：pulse，类型：int32

REG10500～REG10501：0 号编码器当前位置，单位：unit，类型：float

REG10502～REG10503：1 号编码器当前位置，单位：unit，类型：float

REG10504～REG10505：2 号编码器当前位置，单位：unit，类型：float

REG10506～REG10507：3 号编码器当前位置，单位：unit，类型：float

REG10508～REG10509：4 号编码器当前位置，单位：unit，类型：float

REG10510～REG10511：5 号编码器当前位置，单位：unit，类型：float

MODBUS_BIT

命令语法：MODBUS_BIT(first,[last]) = value

函数语法：MODBUS_BIT(first,[last])

描　述：修改或者读取MODBUS BIT位寄存器

参　数：first　　位寄存器编号（SMC104范围0-2047，其他范围0-9999）

Last　　位寄存器编号（SMC104范围first-2047，其他范围first-9999）

返回值：寄存器值

适用范围：全系列控制器

MODBUS_REG

命令语法：MODBUS_REG(regnum) = value

函数语法：MODBUS_REG(regnum)

描　述：修改或者读取MODBUSREG字寄存器

参　数：regnum　　寄存器编号（SMC104范围0-2047，其他范围0-9999）

返回值：寄存器值

适用范围：全系列控制器

MODBUS_LONG

命令语法：MODBUS_LONG(regnum) = value

函数语法：MODBUS_LONG(regnum)

描　述：修改或者读取MODBUS双字寄存器

参　数：regnum　　寄存器编号（SMC104范围0-2046，其他范围0-9998）

value　　整数

返回值：寄存器值

适用范围：全系列控制器

MODBUS_IEEE

命令语法：MODBUS_IEEE(regnum) = value

函数语法：MODBUS_IEEE(regnum)

描　述：修改或者读取双字寄存器，浮点数方式。只能保证8位有效数字

参　数：regnum　　寄存器编号（SMC104范围0-2046，其他范围0-9998）

value　　浮点数

返回值：寄存器值

适用范围：全系列控制器

MODBUS_STRING

命令语法：MODBUS_STRING(regnum，chares) = “string”

函数语法：MODBUS_STRING(regnum)

描　述：修改或者读取多个寄存器，字符串方式，chares为字节数，可以用于文件名等

参　数：regnum　　寄存器起始编号（SMC104范围0-2046，其他范围0-9998）

chares　　字符个数

string　　字符串，或者字符串存储的变量

返回值：把字符串存储到变量里面

适用范围：全系列控制器

⚠注意：读取时最多只能8个字节

例程:将字符写入寄存器，然后打印出来

```
dim state
state = string("X_speed")
modbus_string(1,7) = state
state = modbus_string(1,6)    ' 打印寄存器 1 中的前 6 个字符
Print str(state)              ' 打印字符串变量 state 值
```

运行结果：　X_spee

SMCModbusSetString

语　法：short SMCModbusSetString(uint32 startaddr,uint32 num,char* buf)

描　述：设置寄存器中的数值

参　数：startaddr 寄存器起始地址（SMC104 范围 0-2047，其他范围 0-9999）

num 寄存器位数

buf 寄存器数值

返回值：错误代码

适用范围：全系列控制器

SMCModbusGetString

语　法：short SMCModbusGetString(uint32 startaddr,uint32 num,char* buf)

描　述：读取寄存器中的数值

参　数：startaddr 寄存器起始地址（SMC104 范围 0-2047，其他范围 0-9999）

num 寄存器位数

buf 读取寄存器数值

返回值：错误代码

适用范围：全系列控制器

例程 1：发送和读取寄存器值

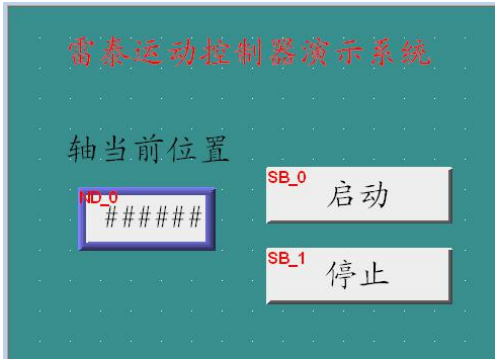
```
dim a(2),buf1(2)
a(0)=asc("3")
a(1)=asc("2")
Print SMCModbusSetString(0,2,a(0))
Print SMCModbusGetString(0,2,buf1(0))
Print *buf1
```

例程2:Modbus寄存器的应用。用触摸屏控制一个电机轴的启动、停止，并显示电机位置

触摸屏界面如下图所示，上面的“启动”、“停止”按钮的寄存器地址分别对应于运动控制器的BIT00000和BIT00001。轴当前位置显示窗口的寄存器地址对应于控制器的REG10000、REG10001，即0号轴的位置。不需要对REG10000、REG10001编程，控制器系统软件自动刷新电机位置。

```
SMCSetProfileUnit(0,0,1000,0.1,0.1,0)      ' 设置 0 号电机速度参数
While1
        If modbus_bit(0) = 1then'  如果触摸屏“启动”按钮按下
                modbus_bit(0) = 0'  将寄存器 BIT00000 清零，等待下次触摸屏的命令
```

```
            SMCVMove(0,1)    ' 0 号电机以速度控制方式运动
        endif
        If modbus_bit(1) = 1then'如果触摸屏“停止”按钮按下
            modbus_bit(1) = 0'将寄存器 BIT00001 清零，等待下次触摸屏的命令
            SMCStop(0,1)                    ' 让 0 号电机停止运动
            Wait idle
        endif
wend
end
```



触摸屏界面

运行结果：轴当前位置将一直显示当前位置值；当启动按键按下时，运动启动；停止按键按下时，运动停止。

4.31 变量存储指令

雷赛运动控制器具有 FLASH 存储器，且支持 U 盘存储方式。

4.31.1 U 盘存储

U_WRITE

语　法：U_WRITE filename，[varname]，[arrayname]

描　述：存储变量或者数组至U盘中

参　数：filename　　文件名，可以为数字或者字符串。字符串长度不可大于100

varname　　变量名

arrayname　　数组名

⚠注意：1、只能存储变量值或整个数组，需要保存字符串时请定义字符串数组保存；

2、文件存储在U盘根目录。

适用范围：除SMC104控制器

U_READ

语　法：U_READ filename，[varname]，[arrayname]

描　述：从U盘里面读取变量或者数组

参　数：filename　　文件名，可以为数字或者字符串。字符串长度不可大于100

varname　　变量名

arrayname　　数组名

⚠注意：变量或数组必须已经定义，文件必须存储在U盘根目录

适用范围：除SMC104控制器

U_STATE

语　法：U_STATE

描　述：检查U盘是否插入

返回值：TRUE：U盘已插入，FALSE：U盘未插入

适用范围：除SMC104控制器

U_CHECKFILE

语　法：U_CHECKFILE(filename)

描　述：检查U盘存储文件是否存在。不用后缀名，系统默认为.sav

参　数：filename　　文件名，可以为数值或者字符串

返回值：TRUE：存在，FALSE：不存在

适用范围：除SMC104控制器

⚠注意：U_CHECKFILE只能检测到变量存储文件，后缀名为.sav

4.31.2 FLASH 按块存储

FLASH_REG

语　法：FLASH_REG(nsect，regnum)

描　述：直接读取FLASH的存储变量

参　数：nsect　　FLASH块编号,SMC104:0-9,其他:0-99

regnum　寄存器编号

返回值：寄存器值

适用范围：全系列控制器

FLASH_READ

语　法：FLASH_READ nsect，[varname]，[arrayname]

描　述：从内部FLASH里面读取变量或者数组

参　数：nsect　　FLASH块编号,SMC104:0-9,其他:0-99

varname　变量名

arrayname　数组名

适用范围：全系列控制器

⚠注意：1）变量或者数组必须已经定义

2）内部FLASH采用顺序存储的方式，读取的顺序必须与存储时的顺序一致

FLASH_WRITE

语　法：FLASH_WRITE nsect，[varname]，[arrayname]

描　述：存储变量或者数组至内部FLASH中

参　数：nsect　　FLASH块编号,SMC104:0-9,其他:0-99

varname　变量名

arrayname　数组名

适用范围：全系列控制器

⚠注意：1）内部FLASH采用顺序存储的方式，读取的顺序必须与存储时的顺序一致

2）内部FLASH有存储次数限制，不要频繁操作

FLASH_SECTSIZE

语　法：FLASH_SECTSIZE

描　述：读取内部FLASH一个块可以存储的变量数

返回值：FLASH一个块的变量数

适用范围：全系列控制器

FLASH_SECTES

语　法：FLASH_SECTES

描　述：读取内部FLASH的总块数

返回值：FLASH的总块数。

适用范围：全系列控制器

4.31.3 FLASH 按文件名存储

NAND_WRITE

语 法： NAND_WRITE filename， [varname]， [arrayname]

描 述：存储变量或者数组到NAND FLASH里面。注意只能存储变量，或者整个数组

参 数： filename 文件名，可以为数字或者字符串；字符串长度不可大于100

varname 变量名

arrayname 数组名

适用范围：除SMC104控制器

NAND_READ

语 法： NAND_READ filename, [varname], [arrayname]

描 述：从NAND FLASH里面读取变量或者数组。变量或数组必须已经定义

参 数： filename 文件名，可以为数字或者字符串；字符串长度不可大于100

varname 变量名

arrayname 数组名

适用范围：除SMC104控制器

NAND_CHECKFILE

语　法： NAND_CHECKFILE(filename)

描　述：检查NAND存储文件是否存在，不用后缀名

参　数： filename 文件名，可以为数值或者字符串

返回值： TRUE： 存在， FALSE： 不存在

适用范围：除SMC104控制器

例　程：检查文件是否存在

```
dim ret
ret=nand_checkfile("filename")
Print ret
```

NAND_DELETEFILE

语　法：NAND_DELETEFILE("FILENAME")

描　述：删除 NAND 存储文件

参　数：　filename 文件名，可以为数值或者字符串

适用范围：除 SMC100 系列控制器

4.32 U 盘文件管理

SMCUdiskGetFirstFile

语　法：short SMCUdiskGetFirstFile(char* filename,int32* filesize,int32* fileid,
uint8 filetype)

描　述：读取 U 盘第一个文件

参　数：filename　文件名
filesize　文件大小
fileid　　文件序号
filetype　文件类型，0-BASIC,1-gcode,2-set 参数设置值

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCUdiskGetNextFile

语　法：short SMCUdiskGetNextFile(char* filename,int32* filesize,int32* fileid,uint8 filetype)

描　述：读取 U 盘下一个文件

参　数：filename　文件名
filesize　文件大小
fileid　　文件序号
filetype　文件类型，0-BASIC,1-gcode,2-set 参数设置值

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCUdiskCheckFile

语　法：SMCUdiskCheckFile(char* filename,int32* filesize，uint8 filetype)

描　述：检测 U 盘中此文件是否存在

参　数：filename　文件名

filesize　文件大小，若为-1 则表示文件不存在

filetype　文件类型，0-BASIC,1-gcode,2-set 参数设置值

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCUdiskCopyFile

语　法：short SMCUdiskCopyFile(const char* SrcFileName,const char* DstFileName,uint8 filetype,uint8 mode)

描　述：复制文件到另一文件

参　数：SrcFileName　源文件名

DstFileName　目标文件

Filetype　文件类型，0-BASIC,1-gcode,2-set 参数设置值

Mode　0-U 盘复制到 Flash,1-Flash 复制到 U 盘,2-U 盘复制到 U 盘

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

⚠注意：控制器内部 Flash 中 Basic 程序文件名固定为 auto.bas ,参数文件名固定为 motion.cfg,U 盘中可为任意名。更新 Basic 程序文件后需重新上电，才能有效。

4.33 G 代码操作指令

4.33.1 G 代码文件执行指令

SMCGcodeStart

语　法：short SMCGcodeStart(void)

描　述：启动

参　数：无

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodePause

语　法：short SMCGcodePause(void)

描　述：暂停

参　数：无

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeStop

语　法：short SMCGcodeStop(void)

描　述：停止

参　数：无

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeState

语　法：short SMCGcodeState(WORD* state)

描　述：读取状态

参　数：state 状态：1-运行，2-暂停，3-停止，100-异常

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeSetStepState

语　法：short SMCGcodeSetStepState(WORD state)

描　述：设置单步运行状态

参　数：state 单步运行状态：0-禁止，1-允许

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeGetStepState

语　法：short SMCGcodeGetStepState(WORD* state)

描　述：读取单步运行状态

参　数：state 单步运行状态：0-禁止，1-允许

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeStopReason

语　法：short SMCGcodeStopReason(WORD* stop_reason)

描　述：读取停止原因

参　数：stop_reason 停止原因

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

GCode_LinerNO

语　法：GCode_LinerNO

描　述：设置G代码插补坐标系

GCode_LinerNO=0'坐标系 0

GCode_LocateSpeed

语　法：GCode_LocateSpeed(axis)

描　述：设置G指令定位运行速度

参　数：　axis　　指定轴号，取值范围：0-控制器最大轴数目-1

GCode_LocateSpeed(0)=40'0 轴定位速度 40

GCode_LocateSpeed(1)=40'1 轴定位速度 40

适用范围：SMC300、SMC600 系列控制器

GCode_LocateTacc

语　法：GCode_LocateTacc(axis)

描　述：设置G指令定位加减速时间

参　数：　axis　　指定轴号，取值范围：0-控制器最大轴数目-1

GCode_LocateTacc(0)=0.1'0 轴定位加减速时间 0.1s

GCode_LocateTacc(1)=0.1'1 轴定位加减速时间 0.1s

适用范围：SMC300、SMC600 系列控制器

GCode_PathSpeed

语　法：GCode_PathSpeed(card)

描　述：设置G指令插补运行速度

参　数：　card　　坐标系号，取值范围：0-1

GCode_PathSpeed(0)=50'插补速度 50

适用范围：SMC300、SMC600 系列控制器

GCode_PathTacc

语　法：GCode_PathTacc(card)

描　述：设置G指令插补加减速时间

参　数：　card　　坐标系号，取值范围：0-1

```
GCode_PathTacc(0)=0.1'插补加减速时间 0.1s
```

适用范围：SMC300、SMC600 系列控制器

4.33.2 G 代码文件编辑指令

SMCGcodeCurrentLine

语　法：short SMCGcodeCurrentLine(uint32* line,uint32* totalline,char* strline)

描　述：读取当前行状态

参　数：line 当前执行行号

totalline 当前文件总行数

strline 当前行字符数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeGetLine

语　法：short SMCGcodeGetLine(uint32 line,char* strLine)

描　述：读取行字符数组

参　数：line 读取行号

strline 读取行字符数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeGetLineArray

语　法：short SMCGcodeGetLineArray(uint32 line,uint32* arraysize,double* linearray)

描　述：读取行数据数组

参　数：line 读取行号

arraysize 读取行数据数组大小

linearray 读取行数据数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeDeleteLine

语　法：short SMCGcodeDeleteLine(uint32 line)

描　述：删除一行 G 代码

参　数：line 删除行号

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeAddLine

语　法：short SMCGcodeAddLine(const char* strLine)

描　述：添加一行 G 代码字符串或字符数字

参　数：strLine 添加行直接字符串(“”)或者字符数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeAddLineArray

语　法：short SMCGcodeAddLineArray(uint32 arraysize,const double* linearray)

描　述：添加一行 G 代码数据数组

参　数：arraysize 添加行数据数组大小

linearray 添加行数据数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeModifyLine

语　法：short SMCGcodeModifyLine(uint32 line,const char* strLine)

描　述：修改一行 G 代码字符串或字符数组

参　数：line 修改行号

strLine 修改行直接字符串(“”)或者字符数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeModifyLineArray

语　法：short SMCGcodeModifyLineArray(uint32 line,uint32 arraysize,double* linearray)

描　述：修改一行 G 代码数据数组

参　数：line 修改行号

arraysize 修改行数据数组大小

linearray 修改行数据数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeInsertLine

语　法：short SMCGcodeInsertLine(uint32 line,const char* strLine)

描　述：插入 G 代码字符

参　数：line 插入行号

strLine 插入行直接字符串(“”)或者字符数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeInsertLineArray

描　述：short SMCGcodeInsertLineArray(uint32 line,uint32 arraysize,double* linearray)

功　能：插入 G 代码数组

参　数：line 插入行号

arraysize 插入行数据数组大小

linearray 插入行数据数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

4.33.3 G 代码文件管理指令

SMCGcodeCheckFile

语　法：short SMCGcodeCheckFile(const char* pFileName,int32* filesize)

描　述：检查文件状态

参　数：pFileName 文件名直接字符串(“”)或者字符数组

filesize 文件大小，文件不存在时返回-1，否则返回文件大小

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeSetCurrentFile/SMCGcodeSetCurFile

语　法：short SMCGcodeSetCurrentFile(const char* pFileName)

描　述：设置 G 代码运行文件

参　数：pFileName 文件名直接字符串(“”)或者字符数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeSaveFile

语　法：short SMCGcodeSaveFile(const char* pFileName)

描　述：保存 G 代码文件

参　数：pFileName 文件名直接字符串(“”)或者字符数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeCreateFile

语　法：short SMCGcodeCreatFile(const char* pFileName)

描　述：创建 G 代码文件

参　数：pFileName 文件名直接字符串(“”)或者字符数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeDeleteFile

语　法：short SMCGcodeDeleteFile(const char* pFileName)

描　述：删除 G 代码文件

参　数：pFileName 文件名直接字符串(“”)或者字符数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeDeleteFileId

语　法：short SMCGcodeDeleteFileId(int fileid)

描　述：按文件号删除 G 代码文件

参　数：fileid 文件号

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeCopyFile

语　法：short SMCGcodeCopyFile(const char* pFileName,const char* pNewFileName)

描　述：复制 G 代码文件

参　数：pFileName 文件名直接字符串(“”)或者字符数组

pNewFileName 文件名直接字符串(“”)或者字符数组

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeGetCurrentFile

语　法：short SMCGcodeGetCurrentFile(char* pFileName,int32* filesize,uint32* fileid,uint32* totalfilenum)

描　述：读取当前 G 代码文件信息

参　数：pFileName 文件名字符数组

filesize 文件大小

fileid 文件号

totalfilenum 文件总数

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeGetFirstFile

语　法：short SMCGcodeGetFirstFile(char* pFileName,int32* filesize)

描　述：读取第一个 G 代码文件

输　出：pFileName 文件名字符数组

filesize 文件大小

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeGetNextFile

语　法：short SMCGcodeGetNextFile(char* pFileName,int32* filesize)

描　述：读取下一个 G 代码文件

参　数：pFileName 文件名字符数组

filesize 文件大小

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCGcodeGetFile

语　法：short SMCGcodeGetFile(uint32 fileid,char* pFileName,uint32* filesize)

描　述：读取 G 代码文件属性

参　数：fileid 文件号

pFileName 文件名字符数组

filesize 文件大小

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

4.34 CAN 指令

雷赛控制器 CAN 模块指令与控制器定义方法相差不大，区别在于 CAN 模块指令需连接 Can 相对应连接口。

4.34.1 CAN 连接基本设置

SETCAN

语　法： SETCAN(Baudrate，CanId)

描　述：设置CAN口的通讯参数

参　数：Baudrate　　波特率，单位: Kbps/s

CanId　　主机 CAN 连接 ID 号

⚠注意：可设波特率值，80,100,125,200,250,400,500,1000。单位KHZ

适用范围：全系列控制器

SMCSetPulseOutmodeCan

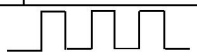
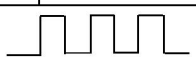
语　法：void SMCSetPulseOutModeCan(WORD canid,WORD axis,WORD outmode,short* result)

描　述：设置指定轴的脉冲输出模式

参　数：Canid　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

Outmode 脉冲输出方式选择，脉冲+方向（0、1、2、3）双脉冲（4、5）

脉冲输出模式	正方向脉冲		负方向脉冲	
	PULSE 输出端	DIR 输出端	PULSE 输出端	DIR 输出端
0		高电平		低电平
1		高电平		低电平
2		低电平		高电平
3		低电平		高电平
4		高电平	高电平	
5		低电平	低电平	
6	PULSE 输出端  DIR 输出端 		PULSE 输出端  DIR 输出端 	

适用范围：脉冲型全系列控制器

⚠注意：请按照驱动器能接收的脉冲模式设置！

SMCGetPulseOutmodeCan

语　法：void SMCGetPulseOutModeCan(WORD canid,WORD axis,WORD* outmode,short* result)

描　述：读取指定轴的脉冲输出模式

参　数：Canid　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

Outmode　返回脉冲输出方式选择，脉冲+方向（0、1、2、3）双脉冲（4、5）

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCSetEquivCan

语　法：void SMCSetEquivCan(WORD canid,WORD axis,double equiv,short* result)

描　述：设置脉冲当量值

参　数：Canid ：　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

equiv 脉冲当量，单位：pulse/unit

result：返回值见错误代码，可省略不写

适用范围：脉冲型全系列控制器

⚠注意：运动关联此参数，必须在运动前设置！

SMCGetEquivCan

语　法：void SMCGetEquivCan(WORD canid,WORD axis,double* equiv,short* result)

描　述：读取脉冲当量值

参　数：Canid ：　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

equiv 返回脉冲当量，单位：pulse/unit

result：返回值见错误代码，可省略不写

适用范围：脉冲型全系列控制器

4.34.2 CAN 运动指令

4.34.2.1 CAN 点位运动指令

SMCSetProfileCan

语　法：void SMCSetProfileCan(WORD canid,WORD axis,double Min_Vel,double Max_Vel,
double Tacc,double Tdec,double Stop_Vel,short* result)

描　述：设置单轴运动速度曲线参数

参　数：Canid ：　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

Min_Vel 起始速度，单位：unit/s, 取值范围：0-2000000

Max_Vel 最大速度，单位：unit/s, 取值范围：0-2000000

Tacc 加速时间，单位：s，取值范围：大于等于 0.001

Tdec 减速时间，单位：s，取值范围：大于等于 0.001

Stop_Vel 停止速度，单位：unit/s，取值范围：0-2000000

result：返回值，见错误代码，可省略不写

适用范围：脉冲型全系列控制器

⚠注意：该函数需运动指令前设置才有效，可设置起始、停止速度，加减速时间决定加速段

例程 1：无起始、停止速度

```
Dim result
SMCSetProfileCan (3,1,0,1000,0.1,0.2,0,result)
'设置 canid 为 3，加速时间为 0.1s，减速时间 0.2s，最大运行速度 1000
```

例程 2：设置起始、停止速度

```
SMCSetProfileCan (3,0,1000,2000,0.1,0.1,2000,result)
'设置 canid 为 3 轴 0 起始速度 100、停止速度为 200，加减速时间为 0.1s，最大运行速度 1000
```

SMCGetProfileCan

语　法：void SMCGetProfileCan(WORD canid,WORD axis,double* Min_Vel,double* Max_Vel, double* Tacc,double* Tdec,double* Stop_Vel,short* result)

描　述：读取单轴运动速度曲线参数

参　数：Canid ：　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

Min_Vel 返回起始速度，单位：unit/s, 取值范围：0-2000000

Max_Vel 返回最大速度，单位：unit/s, 取值范围：0-2000000

Tacc 返回加速时间，单位：s，取值范围：大于等于 0.001

Tdec　返回减速时间，单位：s，取值范围：大于等于 0.001

Stop_Vel 返回停止速度，单位：unit/s，取值范围：0-2000000

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCSetSprofileCan

语　法：void SMCSetSprofileCan(WORD canid,WORD axis,WORD s_mode,double s_para,short* result)

描　述：设置单轴速度曲线 S 段参数值

参　数：Canid　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

s_mode 保留参数，固定值为 0

s_para　S 段时间，单位：s；范围：0~1

result 返回值，见错误代码，可省略不写

适用范围：脉冲型全系列控制器

⚠注意：单轴速度曲线 S 平滑时间，若值为 0 则为 T 形曲线，不为 0 则为 S 平滑曲线。

例　程：

```
SMCSetSprofile (1,0, 0.1)   '设置轴 1 平滑系数 s 为 0.1s，返回值可省略
SMCSetSprofile (0,0, 0)     '设置轴 0，速度曲线为 T 形曲线
```

SMCGetSprofileCan

语　法：void SMCGetSprofileCan(WORD canid,WORD axis,WORD s_mode,double* s_para,short* result)

描　述：读取单轴速度曲线S段参数值

参　数：Canid　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

s_mode 保留参数，固定值为 0

s_para　返回 S 段时间，单位：s；范围：0~1

result 返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCPmoveCan

语　法：void SMCPmoveCan(WORD canid,WORD axis,double Dist,WORD posi_mode,short* result)

描　述：启动定长运动

参　数：Canid　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

Dist 目标位置，单位：unit

posi_mode　运动模式，0：相对坐标模式，1：绝对坐标模式

result　返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

⚠注意：需在运动指令执行前设置速度、加速度等相关参数才有效，返回值可省略

例　程：

```
SMCPmoveCan(3,1,10000, 1)   '轴 1 绝对运动 10000 个脉冲，返回值可省略。
SMCPmoveCan(3,0,10000, 0)   '轴 0 相对运动 10000 个脉冲，返回值可省略。
```

4.34.2.2 CAN 恒速运动指令

SMCVmoveCan

语　法：void SMCVMoveCan(WORD canid,WORD axis,WORD dir,short* result)

描　述：指定轴恒速运动

参　数：Canid：　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

dir　　运动方向，0：负方向，1：正方向

result　　返回值，见错误代码，可省略不写

适用范围：脉冲型全系列控制器

⚠注意：执行该函数时，需设定好速度等参数，当 dir 不为 0 或 1 时，其值大于 1 时为正向运动，小于 0 时为负向运动

例　程：　负向、正向恒速运动

```
dim result
SMCvmoveCan(3,1,0,result)                    '轴 1 负方向速度运动
SMCvmoveCan (3,1,1,result)                   '轴 1 正方向速度运动
```

4.34.2.3 CAN 回原点运动指令

SMCSetHomePinLogicCan

语　法：void SMCSetHomePinLogicCan(WORD canid,WORD axis,WORD org_logic,double filter,short* result)

描　述：设置 ORG 原点信号参数

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围：0~定位模块最大轴数-1

org_logic　ORG 信号有效电平，0：低有效，1：高有效

filter　　保留参数，固定值为 0

result　　返回值，见错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCGetHomePinLogicCan

语　法：void SMCGetHomePinLogicCan(WORD canid,WORD axis,WORD* org_logic,double* filter,short* result)

描　述：读取 ORG 原点信号参数

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围：0~定位模块最大轴数-1

org_logic　　返回 ORG 信号有效电平，0：低有效，1：高有效

filter　　保留参数，固定值为 0

result　　返回值，见错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCSetHomeProfileCan

语　法：short SMCSetHomeProfileCan(WORD canid,WORD axis,double Low_Vel,double High_Vel,double Tacc,double Tdec, short* result)

描　述：设置回原点速度参数

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围：0-控制器最大轴数-1

Low_Vel　设置回原点起始速度

High_Vel　设置回原点运行速度

Tacc　　设置回原点加速、减速时间，单位：s

Tdec　　保留值 0

result　　返回值，见错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCGetHomeProfileCan

语　法：short SMCGetHomeProfileCan(WORD canid,WORD axis,double* Low_Vel,double *High_Vel,double* Tacc,double* Tdec, short* result)

描　述：回读回原点速度参数

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围：0-控制器最大轴数-1

Low_Vel　回读设置回原点起始速度

High_Vel　回读设置回原点运行速度

Tacc　　回读设置回原点加速、减速时间，单位：s

Tdec　　保留值 0

result　　返回值，见错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCSetHomemodeCan

语　法：void SMCSetHomeModeCan(WORD canid,WORD axis,WORD home_dir,double speed_mode,WORD mode,WORD EZ_count,short* result)

描　述：设置回原点模式

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

home_dir　回原点方向，0：负向，1：正向

vel_mode　回原点速度模式：默认值 1，高速回原点

mode　　回原点模式：

1：一次回原点，即平台高速向原点传感器方向运动，当原点传感器触发，电机立即停止。

2：一次回原点加反找，即平台高速向原点传感器方向运动，当原点传感器触发后，电机低速反转；退出原点传感器触发区域后，电机立即停止。

3：二次回原点，即平台高速向原点传感器方向运动，当原点传感器触发后，电机低速反转；退出原点传感器触发区域后，再次以低速向原点传感器方向运动；当原点传感器触发，电机立即停止。

EZ_count 保留参数，固定值为 0

result　返回错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCGetHomemodeCan

语　法:void SMCGetHomeModeCan(WORD canid,WORD axis,WORD* home_dir,double* speed_mode,WORD* mode,WORD* EZ_count,short* result)

描　述：读取回原点模式参数

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

home_dir　返回回原点方向，0：负向，1：正向

vel_mode　返回回原点速度模式：默认值 1，高速回原点

mode　返回回原点模式：

1：一次回原点，即平台高速向原点传感器方向运动，当原点传感器触发，电机立即停止。

2：一次回原点加反找，即平台高速向原点传感器方向运动，当原点传感器触发后，电机低速反转；退出原点传感器触发区域后，电机立即停止。

3：二次回原点，即平台高速向原点传感器方向运动，当原点传感器触发后，电机低速反转；退出原点传感器触发区域后，再次以低速向原点传感器方向运动；当原点传感器触发，电机立即停止。

EZ_count 保留参数，固定值为 0

result　返回错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCHomeMoveCan

语　法：void SMCHomeMoveCan(WORD canid,WORD axis,short* result)

描　述：回原点运动

参　数：Canid　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

result　返回值，见错误代码，可省略不写

适用范围：脉冲型全系列控制器

4.34.3 CAN IO 指令

4.34.3.1 CAN 通用 IO 指令

SMCReadInbitCan

语　法：void SMCReadInbitCan(WORD canid,WORD bitno,WORD* state,short* result)

描　述：读取某个输入端口的电平状态

参　数：Canid　从站 ID 号，范围：1-最大节点数

Bitno　　输入端口号，取值范围：0~定位模块最大输入 IO 端口数-1

state　　读取指定的输入端口电平：0：低电平，1：高电平

result　　返回值，见错误代码，可省略不写

适用范围：全系列控制器

SMCWriteOutbitCan

语　法：void SMCWriteOutbitCan(WORD canid,WORD bitno,WORD state,short* result)

描　述：设置指定控制器的某个输出端口的电平状态

参　数：Canid　　从站 ID 号，范围：1-最大节点数

Bitno　输出端口号，取值范围：0~定位模块最大输出 IO 端口数-1

state　　输出电平，0：低电平，1：高电平

result　返回值，见错误代码，可省略不写

适用范围：全系列控制器

SMCReadOutbitCan

语　法：void SMCReadOutbitCan(WORD canid,WORD bitno,WORD* state,short* result)

描　述：读取某个输出端口的电平

参　数：Canid　　从站 ID 号，范围：1-最大节点数

Bitno　　输出端口号，取值范围：0~定位模块最大输出 IO 端口数-1

state　　返回指定输出端口的电平，0：低电平，1：高电平

result　返回值，见错误代码，可省略不写

适用范围：全系列控制器

SMCReadInportCan

语　法：void SMCReadInportCan(WORD canid,WORD portno,DWORD* state,short* result)

描　述：读取全部输入端口的电平

参　数：Canid　　从站 ID 号，范围：1-最大节点数

portno　　IO 组号默认值为 0，当 IN 输出大于 32 位时，则 IO 组号值为 1

state　　返回输入端口值，为 10 进制数

result　返回值，见错误代码，可省略不写

适用范围：全系列控制器

SMCWriteOutportCan

语　法：void SMCWriteOutportCan(WORD canid,WORD portno,DWORD state,short* result)

描　述：设置指定控制器的全部输出口的电平

参　数：Canid　　从站 ID 号，范围：1-最大节点数

portno　　IO 组号默认值为 0，当 out 输出大于 32 位时，则 IO 组号值为 1

state　　设置 IO 输出值

result　　返回值错误代码，可省略不写

⚠注意：输出端口电平值为 10 进制数，将其转换为 2 进制后各 bit 值为对应 OUT 口的状态值

适用范围：全系列控制器

SMCReadOutportCan

语　法：void SMCReadOutportCan(WORD canid,WORD portno,DWORD* state,short* result)

描　述：设置指定控制器的全部输出口的电平

参　数：Canid　　从站 ID 号，范围：1-最大节点数

portno　　IO 组号默认值为 0，当 out 输出大于 32 位时，则 IO 组号值为 1

state　　返回 IO 输出值,为 10 进制数

result　　返回值错误代码，可省略不写

⚠注意：返回值为 10 进制数，将其转换为 2 进制后各 bit 值为对应 OUT 口的状态值

适用范围：全系列控制器

4.34.3.2 CAN 专用 IO 指令

SMCSetAlmModeCan

语　法：void SMCSetAlmModeCan(WORD canid,WORD axis,WORD enable,WORD alm_logic,WORD alm_action,short* result)

描　述：设置指定轴的ALM信号参数

参　数：Canid 从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

enableALM 信号使能，0：禁止，1：允许

alm_logicALM 信号的有效电平，0：低有效，1：高有效

alm_actionALM 信号的制动方式，0：立即停止（只支持该方式）

result：返回值，见错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCGetAlmModeCan

语　法：void SMCGetAlmModeCan(WORD canid,WORD axis,WORD* enable,WORD* alm_logic,WORD* alm_action,short* result)

描　述：读取指定轴的ALM信号参数

参　数：Canid　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

enable 返回 ALM 信号使能值，0：禁止，1：允许

alm_logic　返回 ALM 信号的有效电平，0：低有效，1：高有效

alm_action 返回 ALM 信号的制动方式，0：立即停止（只支持该方式）

result：返回错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCSetInpModeCan

语　法：void SMCSetInpModeCan(WORD canid,WORD axis,WORD enable,WORD inp_logic,short* result)

描　述：设置指定轴的 INP 信号

参　数：Canid　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

enable　INP 信号使能，0：禁止，1：允许

inp_logic　INP 信号的有效电平，0：低有效，1：高有效

result：返回值，见错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCGetInpModeCan

语　法：void SMCGetInpModeCan(WORD canid,WORD axis,WORD* enable,WORD* inp_logic,short* result)

描　述：读取指定轴的 INP 信号

参　数：Canid　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

enable 返回 INP 信号使能，0：禁止，1：允许

inp_logic 返回 INP 信号的有效电平，0：低有效，1：高有效

result：返回错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCSetEmgModeCan

语　法：void SMCSetEmgModeCan(WORD canid,WORD axis,WORD enable,WORD emg_logic,short* result)

描　述：设置 EMG 急停信号

参　数：Canid ：　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

enable 允许/禁止信号功能，0：禁止，1：允许

emg_logicEMG 信号有效电平，0：低有效，1：高有效

result：返回值，见错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCGetEmgModeCan

语　法：void SMCGetEmgModeCan(WORD canid,WORD axis,WORD* enable,WORD* emg_logic,short* result)

描　述：读取 EMG 急停信号

参　数：Canid ：　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

enable 返回允许/禁止信号功能，0：禁止，1：允许

emg_logic 返回 EMG 信号有效电平，0：低有效，1：高有效

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

4.34.4 CAN 软硬件限位指令

SMCSetElModeCan

语　法：void SMCSetElModeCan(WORD canid,WORD axis,WORD el_enable,WORD el_logic,WORD el_mode,short* result)

描　述：设置 EL 限位信号参数

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

el_enable EL 信号的使能状态：

0：正负限位禁止

1：正负限位允许

2：正限位禁止、负限位允许

3：正限位允许、负限位禁止

el_logic EL 信号的有效电平：

0：正负限位低电平有效

1：正负限位高电平有效

2：正限位高有效，负限位低有效

3：正限位低有效，负限位高有效

el_modeEL 制动方式：

0：正负限位立即停止

1：正负限位减速停止

2：正限位立即停止，负限位减速停止

3：正限位减速停止，负限位立即停止

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCGetElModeCan

语　法：void SMCGetElModeCan(WORD canid,WORD axis,WORD* el_enable,WORD* el_logic,WORD* el_mode,short* result)

描　述：读取 EL 限位信号参数

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

el_enable 返回 EL 信号的使能状态：

0：正负限位禁止

1：正负限位允许

2：正限位禁止、负限位允许

3：正限位允许、负限位禁止

el_logic　返回 EL 信号的有效电平：

0：正负限位低电平有效

1：正负限位高电平有效

2：正限位低有效，负限位高有效

3：正限位高有效，负限位低有效

el_mode　返回 EL 制动方式：

0：正负限位立即停止

1：正负限位减速停止

2：正限位立即停止，负限位减速停止

3：正限位减速停止，负限位立即停止

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCSetSoftlimitCan

语　　法：void SMCSetSoftlimitCan (WORD canid,WORD axis,WORD enable, WORD source_sel,WORD SL_action, double N_limit,double P_limit,short* result)

描　述：设置软限位参数

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围：0~定位模块最大轴数-1

enable　使能状态，0：禁止，1：允许

source_sel 计数器选择，0：指令位置计数器，1：编码器计数器

SL_action　限位停止方式，0：立即停止，1：　减速停止

N_limit　　负限位位置，单位：unit

P_limit　　正限位位置，单位：unit

result：　返回值错误代码，可省略不写

⚠注意：正、负限位位置可为正数也可为负数，但正限位位置应大于负限位位置

适用范围：脉冲型全系列控制器

SMCGetSoftlimitCan

语　法：void SMCGetSoftLimitCan(WORD canid,WORD axis,WORD* enable, WORD* source_sel,WORD* SL_action, double* N_limit,double* P_limit,short* result)

描　述：读取软限位参数

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围：0~定位模块最大轴数-1

Enable　　返回使能状态，0：禁止，1：允许

source_sel　返回计数器选择，0：指令位置计数器，1：编码器计数器

SL_action　返回限位停止方式，0：立即停止，1：　减速停止

N_limit　　返回负限位位置，单位：unit

P_limit　　返回正限位位置，单位：unit

result：　返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

⚠注意：正、负限位位置可为正数也可为负数，但正限位位置应大于负限位位置

4.34.5 CAN 寄存器指令

SMCSetModbus0xCan

语　法：void SMCSetModbus0xCan (WORD canid, WORD start, WORD inum, char* pdata, short* result)

描　述：设置 CAN MODBUS位寄存器

参　数：Canid　　从站ID号，范围：1-最大节点数（BAC116C、SMC-E316无此参数）

Start　　位寄存器起始编号

Inum　　位个数

Pdata　　位数据

result：返回值错误代码，可省略不写

适用范围：全系列控制器

⚠注意：pdata，一个字节为 8 个寄存器。如 pdata=5,则对应寄存器 0、1、2 位值分别为 1、0、1。当设定寄存器个数大于 8 个时，如设定 9、10 号寄存器位值为 1、1，则设定 Pdata[1]=3

SMCGetModbus0xCan

语　法：void SMCGetModbus0xCan(WORD canid, WORD start, WORD inum, char* pdata,short*

result)

描　述：读取MODBUS位寄存器（BAC116C、SMC-E316无此参数）

参　数：Canid　　从站ID号，范围：1-最大节点数

Start　　位寄存器起始编号

Inum　　位个数

Pdata　　返回位数据

result：返回值错误代码，可省略不写

适用范围：全系列控制器

SMCSetModbus4xCan

语　法：void SMCSetModbus4xCan(WORD canid, WORD start, WORD inum, WORD* pdata,short* result)

描　述：设置寄存器值

参　数：Canid　　从站ID号，范围：1-最大节点数（BAC116C、SMC-E316无此参数）

Start　　寄存器起始编号

Inum　　位个数

Pdata　　字符串，或者字符串存储的变量

result：返回值错误代码，可省略不写

适用范围：全系列控制器

SMCGetModbus4xCan

语　法：void SMCGetModbus4xCan(WORD canid, WORD start, WORD inum, WORD* pdata,short* result)

描　述：读取寄存器值

参　数：Canid　　从站ID号，范围：1-最大节点数（BAC116C、SMC-E316无此参数）

Start　　寄存器起始编号

Inum　　位个数

Pdata　　返回字符串，或者字符串存储的变量

result：返回值错误代码，可省略不写

适用范围：全系列控制器

4.34.6 CAN 状态指令

SMCSetPositionCan

语　法：void SMCSetPositionCan(WORD canid,WORD axis,double position,short* result)

描　述：设置当前指令位置计数器值

参　数：Canid　　从站 ID 号，范围：1-最大节点数

Axis　　指定轴号，取值范围：0~定位模块最大轴数-1

Position　位置值，单位：unit

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCGetPositionCan

语　法：void SMCGetPositionCan(WORD canid,WORD axis,double* position,short* result)

描　述：读取当前指令位置值

参　数：Canid　　从站 ID 号，范围：1-最大节点数

Axis　　指定轴号，取值范围：0~定位模块最大轴数-1

Position　返回位置值，单位：unit

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCAxisIoStatusCan

语　法：void SMCAxisIoStatusCan(WORD canid,WORD axis,DWORD* state,short* result)

描　述：读取指定轴有关运动信号的状态

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围：0~定位模块最大轴数-1

state　返回状态值，具体见下表

位号	信号名称	描述
0	ALM	1：表示伺服报警信号 ALM 为 ON； 0：OFF
1	EL+	1：表示正硬限位信号 +EL 为 ON； 0：OFF
2	EL-	1：表示负硬限位信号–EL 为 ON； 0：OFF
3	EMG	1：表示急停信号 EMG 为 ON； 0：OFF
4	ORG	1：表示原点信号 ORG 为 ON； 0：OFF

位号	信号名称	描述
6	SL+	1：表示正软限位信号+SL 为 ON； 0：OFF
7	SL-	1：表示负软件限位信号-SL 为 ON； 0：OFF
8	INP	1：表示伺服到位信号 INP 为 ON； 0：OFF
9	EZ	1：表示 EZ 信号为 ON； 0：OFF
10	RDY 保留	1：表示伺服准备信号 RDY 为 ON； 0：OFF
11	DSTP	1：表示减速停止信号 DSTP 为 ON； 0：OFF
其他位	保留	

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCSetDecStopTimeCan

语　法：void SMCSetDecStopTimeCan(WORD canid,WORD axis,double stop_time, short* result)

描　述：设置减速停止时间；

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围： 0-控制器最大轴数-1

stop_time　减速时间，单位：s

result　　错误代码，可省略不写

适用范围：脉冲型全系列控制器

⚠注意：当发生异常停止时，如：调用 SMCStopCan 函数、限位信号（软硬件）被触发、减速停止信号(DSTP)被触发等进行减速停止时，减速停止时间都为 SMCSetDecStopTimeCan 函数里设置的减速时间

SMCGetDecStopTimeCan

语　法：void SMCGetDecStopTimeCan(WORD canid,WORD axis,double *stop_time, short* result)

描　述：读取减速停止时间设置；

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围： 0-控制器最大轴数-1

stop_time　返回设置的减速时间，单位：s

result　　错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCStopCan

语　法：void SMCStopCan(WORD canid,WORD axis,WORD stopmode,short* result)

描　述：指定轴停止运动

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　指定轴号，取值范围：0~定位模块最大轴数-1

stop_mode 制动方式，0：减速停止，1：紧急停止

result：返回值错误代码，可省略不写

适用范围：全系列控制器

SMCEmgStopCan

语　法：void SMCEmgStopCan(WORD canid,short* result)

描　述：紧急停止所有轴

参　数：Canid　　从站 ID 号，范围：1-最大节点数

result：返回值错误代码，可省略不写

适用范围：全系列控制器

SMCGetTotalAxesCan

语　法：void SMCGetTotalAxesCan(WORD canid,WORD* totalaxes,short* result)

描　述：读取控制器使用的最多轴数

参　数：Canid　　从站 ID 号，范围：1-最大节点数

Totalaxes　返回控制器能使用的最多轴数

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCCheckDoneCan

语　法：short　SMCCheckDoneCan(canid, axis, checkstate, result)

描　述：检测指定轴的运动状态

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围：0~定位模块最大轴数-1

checkstate 轴状态返回值，0：指定轴正在运行，1：指定轴已停止

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCReadCurrentSpeedCan

语　法：void SMCReadCurrentSpeedCan(WORD canid,WORD axis,double* curspeed,short* result)

描　述：读取当前轴运动速度值

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围：0~定位模块最大轴数-1

curspeed　返回当前轴运动速度值

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCGetStopReasonCan

语　法：void SMCGetStopReasonCan(WORD canid,WORD axis,long* stop_reason,short* result)

描　述：轴异常停止原因

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围：0~定位模块最大轴数-1

stop_reason：返回当前轴异常停止原因,具体见以下：

0：正常停止

1：ALM 立即停止，IMD_STOP_AT_ALM

2：ALM 减速停止，DEC_STOP_AT_ALM

3：LTC 外部触发立即停止，IMD_STOP_AT_LTC

4：EMG 立即停止，IMD_STOP_AT_EMG

5：正硬限位立即停止，IMD_STOP_AT_ELP

6：负硬限位立即停止，IMD_STOP_AT_ELN

7：正硬限位减速停止，DEC_STOP_AT_ELP

8：负硬限位减速停止，DEC_STOP_AT_ELN

9：正软限位立即停止，IMD_STOP_AT_SOFT_ELP

10：负软限位立即停止，IMD_STOP_AT_SOFT_ELN

11：正软限位减速停止，DEC_STOP_AT_SOFT_ELP

12：负软限位减速停止，DEC_STOP_AT_SOFT_ELN

13：命令立即停止，IMD_STOP_AT_CMD

14：命令减速停止，DEC_STOP_AT_CMD

15：其他原因， IMD_STOP_AT_OTHER

16：其他原因减速停止，DEC_STOP_AT_OTHER

17：未知原因立即停止，IMD_STOP_AT_UNKOWN

18：未知原因减速停止，DEC_STOP_AT_UNKOWN

19：减速阶段减速停止，DEC_STOP_AT_DEC

20:保留

21：在全程内未找到原点信号停止，原点不在两个限位之间，HOME_STOP_NO_HOME_SIGNAL

22：回零方向和当前有效限位端相反（正限位有效时，回零方向为负方向；负限位有效时，回零方向为正方向），HOME_STOP_HOME_DIR

23：正负限位同时有效，不能启动回零，HOME_STOP_HOME_EL

24：全程未找到EZ信号停止，HOME_STOP_NO_EZ_SIGNAL

25：回零位置溢出停止HOME_STOP_EXCEED_POS

26：运动加速度过大，异常停止，IMD_STOP_AT_SPEED_JIT

27：双原点停止，可能由干扰引起，HOME_STOP_DOUBLE_HOME

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

SMCClearStopReasonCan

语　法：void SMCClearStopReasonCan(WORD canid,WORD axis,short* result)

描　述：清除当前轴错误原因

参　数：Canid　　从站 ID 号，范围：1-最大节点数

axis　　指定轴号，取值范围：0~定位模块最大轴数-1

result：返回值错误代码，可省略不写

适用范围：脉冲型全系列控制器

4.34.7 CAN 模拟量指令

SMCSetAinModeCan

语　法：void SMCSetAinModeCan(WORD canid,WORD channel,WORD enable,WORD workmode,WORD filter,short* result)

描　述：设置输入口工作参数；

参　数：canid：　设置从站 ID 号（不可以设置为零）；

channel：通道设置，设置范围为 0~3；

enable：　工作使能；0 表示禁止该功能，1 表示允许该功能；

workmode：工作模式选择；0 表示电压输入模式，1 表示电流输入模式；

filter：滤波参数设置；设置范围 0~65535；

result：函数返回值；0 表示函数设置成功。

适用范围：脉冲型全系列控制器

SMCGetAinModeCan

语　法：void SMCGetAinModeCan(WORD canid,WORD channel,WORD* enable,WORD* workmode,WORD* filter,short* result)

描　述：回读设置的输入接口参数

参　数：canid：　设置从站 ID 号（不可以设置为零）；

channel：通道设置，设置范围为 0~3；

enable：　工作使能；0 表示禁止该功能，1 表示允许该功能；

workmode：工作模式选择；0 表示电压输入模式，1 表示电流输入模式；

filter：滤波参数设置；设置范围 0~65535；

result：函数返回值；0 表示函数设置成功。

适用范围：脉冲型全系列控制器

SMCGetAinCan

语　法：void SMCGetAinCan(WORD canid,WORD channel,double* fvalue,short* result)

描　述：获取当前输入电压值

参　数：canid：　设置从站 ID 号（不可以设置为零）

channel：通道设置，设置范围为 0~3；

fvalue：　输入接口的当前值。

result：函数返回值；0 表示函数设置成功。

适用范围：脉冲型全系列控制器

SMCSetAoutModeCan

语　法：void SMCSetAoutModeCan(WORD canid,WORD channel,WORD enable,WORD workmode,short* result)

描　述：设置输出口工作参数；

参　数：canid：　设置从站 ID 号（不可以设置为零）；

channel：通道设置，设置范围为 0~1；

enable：　工作使能；0 表示禁止该功能，1 表示允许该功能；

workmode：工作模式选择；0 表示电压输入模式，1 表示电流输入模式；

result：函数返回值；0 表示函数设置成功。

适用范围：脉冲型全系列控制器

SMCGetAoutModeCan

语　法：void SMCGetAoutModeCan(WORD canid,WORD channel,WORD* enable,WORD* workmode,short* result)

描　述：获取输出口工作模式；

参　数：canid：　设置从站 ID 号（不可以设置为零）；

channel：通道设置，设置范围为 0~1；

enable：　工作使能；0 表示禁止该功能，1 表示允许该功能；

workmode：工作模式选择；0 表示电压输入模式，1 表示电流输入模式；

result：函数返回值；0 表示函数设置成功。

适用范围：脉冲型全系列控制器

SMCSetAoutCan

语　法：void SMCSetAoutCan(WORD canid,WORD channel,double fvalue,short* result)

描　述：设置输出口值；

参　数：canid：　设置从站 ID 号（不可以设置为零）；

channel：通道设置，设置范围为 0~1；

fvalue：工作电压或者电流值；

result：函数返回值；0 表示函数设置成功。

适用范围：脉冲型全系列控制器

SMCGetAoutCan

语　法：void SMCGetAoutCan(WORD canid,WORD channel,double* fvalue,short* result)

描　述：获取当前输出接口值

参　数：canid：　设置从站 ID 号（不可以设置为零）；

channel：通道设置，设置范围为 0~1；

fvalue： 输出口的当前值。

result：函数返回值；0 表示函数设置成功。

适用范围：脉冲型全系列控制器

4.35 自由协议指令

4.35.1 串口自由协议指令

SMCComSetMode

语　法：void SMCComSetMode(WORD com, WORD mode)

描　述：设置串口通讯方式

参　数：com　　通讯端口：1-RS232，2-RS485

　　mode　通讯方式：0-Modbus,1-直接读写

返回值：错误代码

适用范围:全系列控制器

⚠注意：通讯模式为 1 时可执行串口的读写操作指令 SMCComWrite、SMCComRead。

SMCComWrite

语　法：short SMCComWrite (WORD com,WORD sendsize, const char * buf)

描　述：发送串口字节数据

参　数：com　　通讯端口：1-RS232，2-RS485

　　sendsize　发送字节数，不大于字节数组长度

　　buf　　字节数组

返回值：错误代码

适用范围:全系列控制器

Print #

语　法：short Print #Port, char* string1

描　述：发送串口字节数据

参　数：Port　通讯端口：1-RS232，2-RS485

　　string1　发送字符串

返回值：错误代码

适用范围:全系列控制器

SMCComRead

语　法：WORD SMCComRead(WORD com,WORD recvsize, byte * buf)

描　述：读取串口字节数据

参　数：com　　通讯端口：1-RS232，2-RS485

recvsize 最大读取字节数，不大于字节数组长度

buf　　字节数组缓冲区

返回值：读取到的字节数

适用范围:全系列控制器

举例说明：串口数据读写。通过串口助手，可检测数据收发是否正确。

```
dim index,mode,bytes,recvsize,data(5)'定义变量
index=1'通讯端口：1-RS232，2-RS485
mode=1'通讯方式：0-Modbus,1-直接读写
bytes=5'最大接收字节数，不大于缓冲区大小
setcom(115200,8,2,2,1)    '设置串口通讯参数
Print SMCComSetMode(index,mode)
strcpy(data(0),"abc")
Print SMCComWrite(index,3,data(0))'发送数据，链接终端会接收到"abc"三个字符
print #index,”aa”+”bcd”"发送字符串，链接终端会接收到"aabcd"三个字符
While 1
    recvsize=SMCComRead(index,bytes,data(0))    '读取数据
    If   recvsize>0   then
        For   i=0 to recvsize-1
                print data(i)          '输出显示接收到的数据
        Next i
    endif
wend
```

4.35.2 网口自由协议指令

Inet_Addr

语　法：int Inet_Addr(const char* IpAddr)

描　述：字符串 IP 地址转换为内部数值 IP 地址

参　数：IpAddr 字符串 IP 地址，如”192.168.5.11”

返回值：数值 IP 地址

适用范围：SMC300、SMC600 系列控制器

SMCEthSetMode

语　法：

short SMCEthSetMode(WORD index,WORD enable,WORD port[,DWORD IpAddrValue[,double TimeOut]])

描　述：设置网络模式

参　数：index　网络通道号:0-2　为服务器端,3-5 为客户端

enable　网络使能:0-禁止,1-使能

port　网络端口号:缺省为 5000

IpAddrValue 控制器做客户端，链接服务器的 IP 地址值，用以下方式赋值：inet_addr("192.168.5.11")

TimeOut　控制器做客户端，链接超时时间，单位 s

返回值：错误代码

SMCEthGetState

语　法：short SMCEthGetState(WORD index,WORD *state)

描　述：读取网络状态

参　数：index　网络通道号:0-2 为服务器端,3-5 为客户端

state　返回网络状态:0-未连接,1-连接

返回值：错误代码

适用范围：SMC300、SMC600 系列控制器

SMCEthRead

语　法：WORD SMCEthRead(WORD index,WORD recvsize,char* buf)

描　述：读取网络数据

参　数：index　网络通道号:0-2 为服务器端,3-5 为客户端

recvsize　接收长度,不得超过接收缓冲区长度

buf 接收缓冲区

返回值：接收长度

适用范围：SMC300、SMC600 系列控制器

SMCEthWrite

语 法：WORD SMCEthWrite(WORD index,WORD sendsize,const char* buf)

描 述：写入网络数据

参 数：index 网络通道号:0-2 为服务器端,3-5 为客户端

sendsize 发送长度,不得超过发送缓冲区长度

buf 发送缓冲区

返回值:实际发送长度

适用范围：SMC300、SMC600 系列控制器

例 1：服务端处理

```
ServerProcess:
dim index,enable,port,state '定义端口索引号、使能、端口号及链接状态变量
dim bufsize,buf(100)          '定义接收缓冲区
dim recvsize                  '定义接收长度变量
index=0'选择 0 号端口索引
enable=1'使能 0 号端口
port=5000'端口号设置为 5000，502 端口号为系统默认，不允许使用
bufsize=100'最大接收 100 个字节
print SMCEthSetMode(index,0,port)          '先禁用端口
print SMCEthSetMode(index,enable,port)   '配置并使能端口
While 1
    SMCEthGetState(index,state) '读取链接状态：0-未链接，1-链接中
    If  state=1 then
        recvsize = SMCEthRead(index,bufsize,buf(0))'读取数据
        if  recvsize>0  then
            print SMCEthWrite(index,recvsize,buf(0))'回发接收到的数据
        endif
    endif
```

```
wend
End
```

例 2：客户端处理

```
ClientProcess:
dim index,enable,port,state '定义端口索引号、使能、端口号及链接状态变量
dim bufsize,buf(100)            '定义接收缓冲区
dim recvsize                    '定义接收长度变量
index=3'选择 3 号端口索引
enable=1'使能 3 号端口
port=5001'端口号设置为 5001，502 端口号为系统默认，不允许使用
bufsize=100'最大接收 100 个字节
print SMCEthSetMode(index,0,port)          '先禁用端口
print SMCEthSetMode(index,enable,port,Inet_Addr("192.168.5.11"),0.01)   '配置并使能端口,10ms 超时
While 1
    SMCEthGetState(index,state) '读取链接状态：0-未链接，1-链接中
    If state=1 then
        recvsize = SMCEthRead(index,bufsize,buf(0))'读取数据
        If  recvsize>0  then
            print SMCEthWrite(index,recvsize,buf(0))'回发接收到的数据
        endif
    endif
wend
end
```

4.35.3 CAN 口自由协议指令

SMCCanSetMode

语　法：void SMCCanSetMode(WORD canid, WORD mode)

描　述：设置串口通讯方式

参　数：canid 目标节点号：1-16

mode　通讯方式：0-内部协议,1-自由协议

返回值：错误代码

适用范围:全系列控制器

⚠注意：通讯模式为 1 时可执行 CAN 口的读写操作指令 SMCCanWrite、SMCCanRead。

SMCCanWrite

语　法：short SMCCanWrite (WORD canid,WORD sendsize, const char * buf)

描　述：发送 CAN 口字节数据

参　数：canid　目标节点号：1-16

sendsize　发送字节数，不大于字节数组长度

buf　　字节数组

返回值：错误代码

适用范围:全系列控制器

SMCCanRead

语　法：WORD SMCCanRead(WORD canid,WORD recvsize, byte * buf)

描　述：读取 CAN 口字节数据

参　数：canid 目标节点号：1-16

recvsize 最大读取字节数，不大于字节数组长度

buf　　字节数组缓冲区

返回值：读取到的字节数

适用范围:全系列控制器

举例说明：CAN 口数据读写。

```
dim canid,mode,bytes,recvsize,data(5)'定义变量
canid=2'目标节点号为 2
mode=1'通讯方式：0-内部协议,1-自由协议
bytes=5'最大接收字节数，不大于缓冲区大小
setcan(1000,1)    '设置 CAN 通讯参数,1MHz 波特率，本机节点号为 1
print SMCCanSetMode(canid,mode)
strcpy(data(0),"abc")
print SMCCanWrite(canid,3,data(0))'发送数据，链接终端会接收到"abc"三个字符
```

```
While 1
    recvsize=SMCCanRead(canid,bytes,data(0))    '读取数据
  If recvsize>0then
       For i=0 to recvsize-1
           print data(i)          '输出显示接收到的数据
       Next i
  endif
wend
```

⚠注意：如果不是雷赛 BASIC 版本控制器之间级联操作读写时需注意以下内部节点号设计规则：节点号采用 11 位，其中最高位保留，低 5 位为目标机节点号，中间 5 位为本机节点号。

4.36 总线相关指令

4.36.1 总线配置指令

NMCSResetCANopen

语　法：short NMCSResetCANopen()
描　述：主站复位
返回值：错误代码
适用范围:总线系列控制器

NMCSStopETC(WORD* ETCState)

语　法：short NMCSStopETC()
描　述：停止 EtherCAT 总线
参　数：ETCState　　EtherCAT 总线状态
返回值：错误代码
适用范围:总线系列控制器

NMCSGetEmergeneyMessege_Nodes

语　法：　NMCSGetEmergeneyMessege_Nodes(WORD PortNo,DWORD* NodeMsg,WORD* MsgNum)

功　能：获取紧急报文报文信息

参　数：PortNo　　端口号 0-3

NodeMsg　　返回节点信息列表(4 字节：1B 寄存器+1B 节点号+2B 错误代码)

MsgNum　　节点信息数

返回值：错误代码

NMCSSetNodeOd

语　法：　short NMCSSetNodeOd(WORD PortNo,WORD NodeNum, WORD Index,WORD SubIndex,WORD ValLength,DWORD* Value)

描　述：设置从站对象字典

参　数：PortNo　　端口号 0-3

NodeNum　　节点号

Index　　索引

SubIndex　　子索引

ValLength　　值长度

Value　　主站值

返回值：错误代码

适用范围:全系列控制器

NMCSGetNodeOd

语　法：　short NMCSGetNodeOd(WORD PortNo,WORD NodeNum, WORD Index,WORD SubIndex,WORD ValLength,DWORD* Value)

描　述：获取从站对象字典

参　数：PortNo　　端口号 0-3

NodeNum　　节点号

Index　　索引

SubIndex　　子索引

ValLength 值长度

Value 主站值

返回值：错误代码

适用范围:全系列控制器

NMCSSendNmtCommand

语 法：short NMCSSendNmtCommand(WORD PortNo,WORD NodeID,WORD NmtCommand)

描 述：发送 NMT 管理报文

参 数：PortNo 端口号 0-3

NodeID 节点号

NmtCommand NMT 命令，该参数有五个值：

0x01-启动远程节点

0x02-停止远程节点

0x80-远程节点进入预操作模式

0x81-复位远程节点

0x82-远程节点复位通讯

返回值：错误代码

适用范围:全系列控制器

NMCSGetCycleTime

语 法：short NMCSGetCycleTime(WORD PortNum,DWORD* CycleTime)

功 能：读取 EtherCAT 总线循环周期

参 数：PortNum EtherCAT 端口号，固定为 2(其中 0、1 为 CANopen 端口)

CycleTime EtherCAT 总线循环周期，单位：us

返回值：错误码

NMCSWriteRxpdoExtra

语法：NMCSWriteExtra(WORD PortNo,WORD address,WORD DataLen,DWORD Value)

功能：写扩展 rxpdo

参数： PortNum：端口号，0,1 表示 CANOpen，2，3 表示 EtherCAT 端口

Address：扩展 PDO 的首地址

DataLen：数据长度，按 16bit 计算，最大值为 2（表示 32bit 数据）

Value：数据值

返回值：错误码

NMCSReadRxpdoExtra

语法：Short nmcs_read_rxpdo_extra(WORD PortNo,WORD address,WORD DataLen,DWORD* Value)

功能：读扩展 rxpdo

参数： ConnectNo 链接号：0-7 号，默认值 0

PortNum：端口号，0,1 表示 CANOpen，2，3 表示 EtherCAT 端口

address：扩展 PDO 的首地址

DataLen：数据长度，按 16bit 计算，最大值为 2（表示 32bit 数据）

Value：数据值

返回值：错误码

NMCSReadTxpdoExtra

NMCSReadTxpdoExtra(WORD PortNo,WORD address,WORDDataLen,DWORD* Value)

功能：读扩展 txpdo

参数： ConnectNo 链接号：0-7 号，默认值 0

PortNum：端口号，0,1 表示 CANOpen，2，3 表示 EtherCAT 端口

address：扩展 PDO 的首地址

DataLen：数据长度，按 16bit 计算，最大值为 2（表示 32bit 数据）

Value：数据值

返回值:错误码

4.36.2 总线 IO 及轴控制指令

NMCSGetAxisType

语　法：short NMCSGetAxisType(WORD axis,WORD* Axis_Type)

功　能：读取轴类型

参　数：axis　　轴号

Axis_Type　　轴的类型，0：虚拟轴，1：EtherCAT 轴，2：CANopen 轴，3：脉冲轴，4：未知类型轴

返回值：错误码

NMCSSetAxisEnable

语　法：short NMCSSetAxisEnable(WORD axis)

功　能：设置 EtherCAT 总线轴使能

参　数：axis　　EtherCAT 总线轴轴号

返回值：错误码

NMCSSetAxisDisable

语　法：short NMCSSetAxisEnable(WORD axis)

功　能：设置 EtherCAT 总线轴禁止使能

参　数：axis　　EtherCAT 总线轴轴号

返回值：错误码

NMCSSynMoveUnit

语　法：short NMCSSynMoveUnit(WORD AxisNum,WORD* AxisList,double* Position,WORD* PosiMode)

描　述：同步运动

参　数：AxisNum　　轴数(值范围 1-32，最大 32 个轴)

AxisList　　轴列表

Position　　目标位置列表

PosiMode　　位置模式列表：

0-相对模式

1-绝对模式

返回值：错误代码

适用范围:全系列控制器

⚠注意：该运动的运动速度由 SMCSetProfileUnit 设置

NMCSGetAxisStateMachine

语　法：short NMCSGetAxisStateMachine(WORD axis, WORD* Axis_StateMachine)

功　能：读取 EtherCAT 总线轴状态机

参　数：axis　　EtherCAT 总线轴轴号

Axis_StateMachine　　EtherCAT 总线轴状态机

0：未启动状态(NOT_READY_SWITCH_ON)

1：启动禁止状态(SWITCH_ON_DISABLE)

2：准备启动状态(READY_TO_SWITCH_ON)

3：启动状态(SWITCH_ON)

4：操作使能状态(OP_ENABLE)

5：停止状态(QUICK_STOP)

6：错误触发状态(FAULT_ACTIVE)

7：错误状态(FAULT)

返回值：错误码

NMCSGetAxisControlMode

语　法：short NMCSGetAxisControlMode(WORD axis,long* controlmode)

功　能：读取 EtherCAT 总线轴控制模式

参　数：axis　　EtherCAT 总线轴轴号

controlmode　　EtherCAT 总线轴控制模式

6：回零模式

8：CSP 模式

返回值：错误码

NMCSGetTotalAxes

语　法：short NMCSGetTotalAxes(DWORD *TotalAxis)

描　述：获取控制器轴数

参　数：TotalAxis　　控制器轴数

返回值：错误代码

适用范围:全系列控制器

NMCSGetTotalIonum

语　法：short NMCSGetTotalIonum(WORD *TotalIn,WORD *TotalOut)

描　述：获取控制器通用 I/O 点数

参　数：TotalIn　　返回输入口点数

TotalOut　　返回输出口点数

返回值：错误代码

适用范围:全系列控制器

NMCSGetTotalAdcnum

语　法：short NMCSGetTotalAdcnum(WORD *TotalIn,WORD *TotalOut)

描　述：获取总线卡模拟量 I/O 点数

参　数：TotalIn　　返回模拟量输入口点数

TotalOut　　返回模拟量输出口点数

返回值：错误代码

适用范围:全系列控制器

NMCSGetTotalIONum

语　法：short NMCSGetTotalIONum(WORD *TotalIn,WORD *TotalOut)

功　能：读取 EtherCAT 总线 IO 输入输出口数

参　数：TotalIn　　EtherCAT 总线输入总通道数

TotalOut　　EtherCAT 总线输出总通道数

返回值：错误码

NMCSGetTotalSlaves

语　法：short NMCSGetTotalSlaves(WORD PortNum,WORD* TotalSlaves)

功　能：获取 EtherCAT 从站总数

参　数：PortNum　　EtherCAT 端口号，固定为 2(其中 0、1 为 CANopen 端口)

　　TotalSlaves　　EtherCAT 从站总数

返回值：错误代码

NMCSGetAxisIOOut

语　法：short NMCSGetAxisIOOut(WORD axis,DWORD* iostate)

功　能：获取 EtherCAT 轴数字量输出 IO 状态

参　数：axis　　EtherCAT 轴号

　　iostate　　EtherCAT 轴数字量输出 IO 状态

返回值：错误代码

NMCSSetAxisIOOut

语　法：short NMCSSetAxisIOOut(WORD axis,DWORD iostate)

功　能：设置 EtherCAT 轴数字量输出 IO 状态

参　数：axis　　EtherCAT 轴号

　　iostate　　EtherCAT 轴数字量输出 IO 状态

返回值：错误代码

NMCSGetAxisIOIn

语　法：short NMCSGetAxisIOOut(WORD axis,DWORD* iostate)

功　能：获取 EtherCAT 轴数字量输入 IO 状态

参　数：axis　　EtherCAT 轴号

　　iostate　　EtherCAT 轴数字量输入 IO 状态

返回值：错误代码

NMCSWriteOutbit

语　法：short NMCSWriteOutbit(WORD PortNo,WORD NodeID,WORD BitNo,WORD IoValue)

描　述：设置指定控制器或扩展模块的某个输出端口的电平

参　数：PortNo　　端口号 0-3

NodeID　　节点号

IoBit　　输出端口号

IoValue　　输出电平，　0：　低电平，　1：　高电平

返回值：错误代码

适用范围:全系列控制器

NMCSReadOutbit

语　法：short NMCSReadOutbit(WORD PortNo,WORD NodeID,WORD BitNo,WORD *IoValue)

描　述：读取指定控制器或扩展模块的某个输出端口的电平

参　数：PortNo　　端口号 0-3

NodeID　　节点号

IoBit　　输出端口号

IoValue　　返回输出电平，　0：　低电平，　1：　高电平

返回值：错误代码

适用范围:全系列控制器

NMCSReadInbit

语　法：short NMCSReadInbit(WORD PortNo,WORD NodeID,WORD BitNo,WORD *IoValue)

描　述：读取指定控制器或扩展模块的某个输入端口的电平

参　数：PortNo　　端口号 0-3

NodeID　　节点号

BitNo　　输入口号

IoValue　　输入端口电平，　0：　低电平，　1：　高电平

返回值：错误代码

适用范围:全系列控制器

NMCSReadInport

语 法：short NMCSReadInport(WORD PortNo,WORD NodeID,WORD PortNo,DWORD *IoValue)

描 述：读取指定 IO 组号的全部输入口的电平

参 数：PortNo 端口号 0-3

NodeID 节点号

PortNo IO 组号，最小值 0

IoValue 输入端口电平， 0： 低电平， 1： 高电平

返回值：错误代码

适用范围:全系列控制器

⚠注意：IO 组号最小值为 0，每 32 点 IO 一组；IO 数满 32，组号加 1

NMCSReadOutport

语 法 ： short NMCSReadOutport(WORD PortNo,WORD NodeID,WORD PortNo,DWORD *IoValue)

描 述：读取指定 IO 组号的全部输出口的电平

参 数：PortNo 端口号 0-3

NodeID 节点号

PortNo IO 组号，最小值 0

IoValue 输入端口电平， 0： 低电平， 1： 高电平

返回值：错误代码

适用范围:全系列控制器

4.36.3 总线错误代码指令

NMCSSetAlarmClear

语法：NMCSSetAlarmClear(WORD PortNo,WORD NodeNo)

功能：清除报警信号,支持 CANopen、EtherCAT

参数: ConnectNo 指定链接号：0-7,默认值 0

PortNo EtherCAT 总线端口号，固定为 2(其中 0、1 为 CANopen 端口)

NodeNo 默认为 0

返回值：错误代码

NMCSGetErrcode

语　法：NMCSGetErrcode(WORD PortNum,WORD *errcode)

功　能：获取 EtherCAT 总线状态

参　数：PortNum　　EtherCAT 端口号，固定为 2(其中 0、1 为 CANopen 端口)

　　errcode　　EtherCAT 总线状态，0 表示正常

返回值：错误代码

NMCSGetCardErrcode

语　法：short NMCSGetCardErrcode(WORD *Errcode)

描　述：获取总线错误代码

参　数：Errcode　　总线错误代码

返回值：错误代码

适用范围:全系列控制器

NMCSClearCardErrcode

语　法：short NMCSClearCardErrcode()

描　述：清除指定连接号总线错误代码

参　数：无

返回值：错误代码

适用范围:全系列控制器

NMCSGetAxisErrcode

语　法：short NMCSGetAxisErrcode(WORD axis,WORD *Errcode)

描　述：获取轴错误代码

参　数：axis　　指定轴号，0~控制器最大轴数-1

　　Errcode　　轴错误代码

返回值：错误代码

适用范围:全系列控制器

NMCSClearAxisErrcode

语　法：short NMCSClearAxisErrcode(WORD iaxis)

描　述：清除轴错误代码

参　数：axis　　指定轴号，0~控制器最大轴数-1

返回值：错误代码

适用范围:全系列控制器

第 5 章 BASIC 指令索引

表格 1：BASIC 指令一览表

BASIC 指令最新版本，增加了许多新功能的指令。在线输出指在 SMC BASIC Studio 软件中在线输出指令，在线输出指令需对所属参数赋值，否则可能报错。

功能	名称	功能说明	在线输出
基本运算指令	+、-、*、/、\、Mod	加、减、乘、除、整除、取余	√
	=、<>、<、>、<=、>=	等于、不等于、小于、大于、小于等于、大于等于	√
	Not、And、Or 、Xor、Eqv	逻辑非、逻辑与、逻辑或、逻辑异或、逻辑同或	√
	ABS	绝对值	√
	SIN、COS、ACOS、TAN、ATAN	正弦、余弦、反余弦、正切、反正切函数	√
	SQR、LN、LOG	平方根、自然对数、对数	√
	CLEAR_BIT	将操作数的二进制的第 bit 位清 0	√
	READ_BIT	读取操作数的二进制的第 bit 位的值	√
	SET_BIT	将操作数的二进制的第 bit 位置 1	√
	FRAC	返回表达式的小数部分	√

	INT	返回表达式的整数部分	√
	SGN	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	√
	CHR	返回表达式的值对应的 ASCII 码	√
	STR	打印变量中存储的字符串	√
	STRING	把字符串存储在一个变量中	√
	ASC	获取字符串首字符的 ASCII 十进制值	√
系统默认指令	PI	圆周率，等于1.14159	√
	TRUE	true 为常数：-1	√
	FALSE	false 为常数：0	√
	ON	on 为常数：1	√
	OFF	off 为常数：0	√
流程控制指令	DIM	定义变量，数组	X
	UNDIM	删除变量，数组	X
	CONST	定义常数符号名和值	X
	DMINS	数组的链表操作，插入	X
	DMDEL	数组的链表操作，删除	X
	IF THEN ELSEIF ENDIF	该指令为条件判断语句。	X
	FORTOSTEPNEXT	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	X
	WHILE WEND	当 condition 条件成立时，执行循环体内的指令块；否则，结束循环	X
	REPEAT UNTIL	循环语句，循环执行 commands 指令块，当 condition 为真时退出循环	X
	WAIT UNTIL	原地等待，直到条件满足	X

	GOTO	强制跳转，也称为无条件跳转	X
	ON GOTO	条件跳转，当 expression 条件为真时，跳转	X
	DELAY	延时指令	X
子程序多任务指令	GOSUB (CALL)	调用过程（子程序）	X
	RETURN(End Sub)	用户过程返回	X
	SUB	定义过程（子程序）	X
	ON GOSUB	expression 条件为真时，调用过程 label	X
	RUN	启动多任务程序	X
	END	结束当前任务或程序	X
	STOP	任务强制停止	X
	AUTO	主程序标号。如果有 auto 标号，开机自动运行该程序	X
	PAUSE	暂停任务	
	HALT	停止全部任务	X
定时器指令	TIMER_START	启动定时器	X
	TIMER_STOP	强制停止定时器	X
	TIMER_IFEND	查询定时器是否停止	X
	TICKS	读取系统开机后时钟的计时时间。	√
中断控制指令	ONTIMER0～控制器最大定时器-1	定时器计数结束中断入口	X
	OnInOn0～最大输入端口数-1	通用数字输入信号由 0 变为 1 时，产生中断的程序入口	X
	OnInOff0～最大输入端口数-1	通用数字输入信号由 1 变为 0 时，产生中断的程序入口	X
	OnSoftEl0～最大电机轴数-1	软件限位信号的中断处理程序入口	X
	OnEl0～最大电机轴数-1	硬件限位信号的中断处理程序入口	X
	OnAlarm0～最大电机轴数-1	伺服电机的报警信号 ALM 的中断处理程序入口	X
信息输出	PRINT	在终端打印参数。通常用于输出一些信息	√

指令	TRACE	显示字符串	√
	VERSION	读取运动控制器软件版本	√
	ERROR WORN TRACE	与 PRINT 指令类似，定义用户特殊输出	√
	ERRSWITCH	出错输出开关。通过开关可以屏蔽 ERROR 等的输出	√
断电保存指令	STATICREG/NVRAM_INT	读写断电保护寄存器（铁电），整数	√
	NVRAM/NVRAM_FLOAT	读写断电保护寄存器（铁电），浮点数	√
	NVRAM_BYTE	读写断电保护寄存器（铁电），8 位整数	√
	NVRAM_SHORT	读写断电保护寄存器（铁电），16 位整数	√
FLASH 存储指令	BURNSET	保存所有参数的当前值至 FLASH，下次开机时会自动从 FLASH 读取保存的参数	√
	FLASH_REG	直接读取 FLASH 的存储变量	X
	FLASH_READ	从内部 FLASH 里面读取变量或者数组	X
	FLASH_WRITE	存储变量或者数组至内部 FLASH 中	X
	FLASH_SECTSIZE	读取内部 FLASH 一个块可以存储的变量数	√
	FLASH_SECTES	读取内部 FLASH 的总块数	√
	NAND_WRITE	存储变量或者数组到 NAND FLASH 里面	X
	NAND_READ	从 NAND FLASH 里面读取变量或者数组	X
	NAND_CHECKFILE	检查 NAND 存储文件是否存在	√
	NAND_DELETEFILE	删除 NAND 存储文件	√
U 盘存储	U_WRITE	存储变量或者数组至U盘中	X
	U_READ	从 U 盘里面读取变量或者数组	X
	U_STATE	检查 U 盘是否插入	√
	U_CHECKFILE	检查 U 盘存储文件是否存在	√
U 盘管理	SMCUdiskGetFirstFile	读取 U 盘第一个文件	X
	SMCUdiskGetNextFile	读取 U 盘下一个文件	X
	SMCUdiskCheckFile	检测 U 盘中此文件是否存在	√
	SMCUdiskCopyFile	复制文件内容到另一文件	√
字符、整	strcmp	字符串间比较	X

数等类型转换	strcpy	字符串间的复制	X
	strtof	字符串转换为浮点数	X
	strtol	字符串转换为整数	X
	sprintf_l	字符串格式化命令，把整形数据按格式写入字符串	X
	sprintf_f	字符串格式化命令，把浮点数据按格式写入字符串	X
	strcat	连接字符串	X
	strlen	计算给定字符串的长度	X
	memset	将 s 数组中的前 n 个数用 ch 替换	X
实时时钟	RTC_TIMES	读取 RTC 的运行秒数；掉电后不会丢失。	√
	RTC_TIME	读取 设置 RTC 的当前时间，掉电后不会丢失。	√
	RTC_DATE	读取设置 RTC 的当前日期；掉电后不会丢失	√
参数操作	SMCDefaultSet	恢复默认参数设置	√
	BURNSET	保存所有参数的当前值至 FLASH，下次开机时会自动从 FLASH 读取保存的参数	√
通讯连接指令	IPADDR	查询 IP 地址	√
	SETIP	设置 IP 地址	√
	SETCOM	设置串口的通讯参数	√
	COM	查询 com 口参数	√
脉冲模式指令	SMCSetPulseOutmode	设置指定轴的脉冲输出模式	√
	SMCGetPulseOutmode	读取指定轴的脉冲输出模式	X
脉冲当量指令	SMCSetEquiv	设置脉冲当量值	√
	SMCGetEquiv	读取脉冲当量值	X
反向间隙	SMCSetBacklashUnit	设置反向间隙值	√
	SMCGetBacklashUnit	读取反向间隙设定值	X
状态监控	SMCStop	指定轴停止运动	√

指令	SMCStopMulticoor	停止坐标系内所有轴的运动	√
	SMCEmgStop	紧急停止所有轴	√
	SMCCheckDone	检测指定轴的运动状态	√
	SMCCheckDoneMulticoor	检测坐标系的运动状态	√
	SMCAxisIoStatus	读取指定轴有关运动信号的状态	√
	SMCSetPositionUnit	设置当前指令位置计数器值	√
	SMCGetPositionUnit	读取当前指令位置值	√
	SMCGetTargetPositionUnit	读取当前目标位置	√
	SMCSetWorkPosUnit	设置当前工件原点	√
	SMCGetWorkPosUnit	读取当前工件原点坐标	√
	SMCReadCurrentSpeedUnit	读取当前轴运动速度	√
点位运动指令	SMCSetProfileUnit	设置单轴运动速度参数	√
	SMCGetProfileUnit	读取单轴运动速度参数	X
	SMCSetSprofile	设置单轴速度曲线 S 段参数值	√
	SMCGetSprofile	读取单轴速度曲线 S 段参数值	X
	SMCPmoveUnit	定长运动，加/减速度恒定	√
	SMCVmove	指定轴恒速运动	√
在线变速变位	SMCResetTargetPositionUnit	在线改变指定轴的当前目标位置，位置值为运动的绝对位置点	√
	SMCUpdateTargetPositionUnit	强制改变指定轴的当前目标位置	√
	SMCChangeSpeedUnit	在线改变指定轴的当前运动速度	√
回原点运动	SMCSetHomemode	设置回原点模式参数	√
	SMCGetHomemode	读取回原点模式参数	X
	SMCSetHomePinLogic	设置 HOME 信号的有效电平	√
	SMCGetHomePinLogic	读取 HOME 信号的有效电平	X
	SMCSetEZCount	设置回原点 EZ 个数	√
	SMCGetEZCount	回读回原点 EZ 个数	X
	SMCSetHomePositionUnit	设置回原点完成后偏移位置值	√
	SMCGetHomePositionUnit	读取回原点完成后偏移位置值	X

	SMCSetHomeProfileUnit	设置回原点速度参数值	√
	SMCGetHomeProfileUnit	读取回原点速度参数值	X
	SMCHomeMove	回原点运动	√
	SMCGetHomeResult	读取回原点运动状态	X
PVT 运动指令	SMCPttTableUnit	向指定数据表传送数据，采用 PTT 模式	X
	SMCPtsTableUnit	向指定数据表传送数据，采用 PTS 模式	X
	SMCPvtTableUnit	向指定数据表传送数据，采用 PVT 模式	X
	SMCPvtsTableUnit	向指定数据表传送数据，采用 PVTS 模式	X
	SMCPvtMove	启动 PVT 运动	X
插补运动参数	SMCSetVectorProfileUnit	设置插补运动的速度、加减速时间等参数	√
	SMCGetVectorProfileUnit	设置插补运动的速度、加减速时间等参数	X
	SMCSetVectorSprofile	设置平滑 S 段时间值	√
	SMCGetVectorSprofile	读取平滑 S 段时间值	X
插补运动指令	SMCLineUnit	启动直线插补运动	X
	SMCArcMoveCenterUnit	圆心+圆弧终点模式扩展的螺旋线插补运动	X
	SMCArcMoveRadiusUnit	半径+圆弧终点模式扩展的螺旋线插补运动	X
	SMCArcMove3pointsUnit	三点圆弧模式扩展的螺旋线插补运动	X
连续插补设参数置指令	SMCContiOpenList	打开连续插补缓冲区	X
	SMCContiStartList	开始连续插补	√
	SMCContiCloseList	关闭连续插补缓冲区	√
	SMCContiPauseList	暂停连续插补	√
	SMCContiDelay	连续插补中暂停延时指令	√
	SMCContiStopList	停止连续插补	√
	SMCContiSetLookAheadMode	设置小线段前瞻模式及参数	√
	SMCContiGetLookAheadMode	读取小线段前瞻模式及参数	√
	SMCContiSetBlend	设置 Blend 是否使能	√
	SMCContiGetBlend	读取插补是否使用了 Blend 功能	√
	SMCContiChangeSpeedRatio	动态调整连续插补速度比例	√
连续插补	SMCContiRemainSpace	读取插补缓存去剩余空间	√

运动状态	SMCContiReadCurrentMark	读取连续插补缓冲区当前插补段号	√
	SMCContiGetRunState	读取连续插补运动状态	√
连续插补 IO 指令	SMCContiSetPauseOutput	设置连续插补暂停及异常停止时 IO 输出状态	X
	SMCContiGetPauseOutput	读取连续插补暂停及异常停止时 IO 输出状态参数	X
	SMCContiWaitInput	连续插补等待 IO 输入	√
	SMCContiDelayOutbitToStart	连续插补中相对于轨迹段起点 IO 滞后输出（段内执行）	√
	SMCContiDelayOutbitToStop	连续插补中相对于轨迹段终点 IO 滞后输出	√
	SMCContiAheadOutbitToStop	连续插补中相对于轨迹段终点 IO 提前输出（段内执行）	√
	SMCContiWriteOutbit	连续插补中缓冲区立即 IO 输出	√
	SMCContiClearIoAction	清除段内未执行完的 IO 动作	√
PWM 立即输出	SMCSetPwmOutput	设置 PWM 输出信号的占空比、频率参数	√
	SMCGetPwmOutput	读取 PWM 输出信号的占空比、频率参数	X
通用 IO 指令	SMCReadInbit	读取某个输入端口的电平	√
	SMCWriteOutbit	设置指定控制器的某个输出端口的电平	√
	SMCReadOutbit	读取某个输出端口的电平	√
	SMCReadInport	读取全部输入端口的电平	√
	SMCWriteOutport	设置指定控制器的全部输出口的电平	√
	SMCReadOutport	读取指定控制器的全部输出口的电平	√
	SMCReverseOutbit	IO 输出延时翻转	√
	SMCSetIoCountMode	设置 IO 计数模式	√
	SMCGetIoCountMode	读取 IO 计数模式设置值	X
	SMCSetIoCountValue	设置 IO 计数值	√
	SMCGetIoCountValue	读取 IO 计数值	X
专用 IO 指令	SMCSetAlmMode	设置指定轴的 ALM 信号参数	√
	SMCGetAlmMode	读取指定轴的 ALM 信号参数	X

	SMCSetInpMode	设置指定轴的 INP 信号	√
	SMCGetInpMode	读取指定轴的 INP 信号	X
	SMCSetEzMode	设置指定轴的 EZ 信号	√
	SMCGetEzMode	读取指定轴的 EZ 信号	X
	SMCWriteSevonPin	设置指定轴的伺服使能端口的输出	√
	SMCReadSevonPin	读取指定轴的伺服使能端口的电平	√
	SMCWriteErcPin	设置指定轴的 ERC 信号输出	√
	SMCReadErcPin	读取指定轴的 ERC 信号输出	√
	SMCReadAlarmPin	读取指定轴的 ALARM 端口电平	√
	SMCReadInpPin	读取指定轴的 INP 端口电平	√
	SMCReadOrgPin	读取指定轴的 ORG 端口电平	√
	SMCReadElpPin	读取指定轴的 ELP 端口电平	√
	SMCReadElnPin	读取指定轴的 ELN 端口电平	√
	SMCReadEzPin	读取指定轴的 EZ 端口电平	√
	SMCReadEmgPin	读取指定轴的 EMG 端口电平	√
手轮指令	SMCHandwheelSetAxislist	设置同一轴选档位下具体运动轴列表	X
	SMCHandwheelGetAxislist	读取同一轴选档位下具体运动轴列表	X
	SMCHandwheelSetRatiolist	设置同一轴选下手轮倍率档位	X
	SMCHandwheelGetRatiolist	读取同一轴选下手轮倍率档位	X
	SMCHandwheelSetMode	设置手轮运动模式，硬件、还是软件模式下运动	√
	SMCHandwheelGetMode	读取手轮运动模式，硬件、还是软件模式下运动	X
	SMCHandwheelSetIndex	选择或更换手轮运动轴选、倍率档位	√
	SMCHandwheelGetIndex	读取选择或更换手轮运动轴选、倍率档位	X
	SMCHandwheelMove	启动手轮运动	√
	SMCHandwheelStop	停止手轮运动	√
编码器	SMCSetCounterInmode	设置编码器信号参数	√
	SMCGetCounterInmode	读取编码器信号参数	X

	SMCSetEncoderUnit	设置编码器计数值	√
	SMCGetEncoderUnit	读取编码器计数值	X
	SMCSetCounterReverse	设置 AB 相反向计数模式	√
	SMCGetCounterReverse	读取 AB 相反向计数模式	X
高速位置锁存指令	SMCSetLtcMode	设置指定轴的 LTC 信号参数	√
	SMCGetLtcMode	读取指定轴的 LTC 信号参数	X
	SMCSetLatchMode	设置锁存方式	√
	SMCGetLatchMode	读取锁存方式参数	X
	SMCGetLatchValueUnit	从控制器内读取锁存器的值	√
	SMCGetLatchFlag	从控制器内读取指定轴的锁存次数	√
	SMCResetLatchFlag	复位锁存器的标志位	√
原点位置锁存指令	SMCSetHomelatchMode	设置原点锁存模式	√
	SMCGetHomelatchMode	读取原点锁存模式参数	X
	SMCResetHomelatchFlag	清除原点锁存标志	√
	SMCGetHomelatchFlag	读取原点锁存标志	√
	SMCGetHomelatchValueUnit	读取原点锁存值	√
EZ 位置锁存指令	SMCSetEzlatchMode	设置 EZ 锁存模式	√
	SMCGetEzlatchMode	读取 EZ 锁存模式参数	X
	SMCResetEzlatchFlag	清除 EZ 锁存标志	√
	SMCGetEzlatchFlag	读取 EZ 锁存标志	√
	SMCGetEzlatchValueUnit	读取 EZ 锁存值	√
低速一维位置比较	SMCCompareSetConfig	设置一维位置比较器	√
	SMCCompareGetConfig	读取一维位置比较器参数	X
	SMCCompareClearPoints	清除已添加的所有一维位置比较点	√
	SMCCompareAddPointUnit	添加一维位置比较点	√
	SMCCompareGetCurState	读取比较状态	X
高速一维位置比较	SMCHcmpSetMode	设置高速比较模式	√
	SMCHcmpGetMode	读取高速比较模式参数	X
	SMCHcmpSetConfig	配置高速比较器	√

	SMCHcmpGetConfig	读取高速比较器参数	X
	SMCHcmpClearPoints	清除已添加的所有高速位置比较点	√
	SMCHcmpAddPointUnit	添加/更新高速比较位置	√
	SMCHcmpSetLinerUnit	设置高速比较线性模式参数	√
	SMCHcmpGetLinerUnit	读取高速比较线性模式参数	X
	SMCHcmpGetCurState	读取高速比较状态	X
软件硬件限位指令	SMCSetElMode	设置 EL 限位信号参数	√
	SMCGetElMode	读取 EL 限位信号设置参数	√
	SMCSetSoftlimitUnit	设置软限位参数	√
	SMCGetSoftlimitUnit	读取软限位参数	√
运动异常停止	SMCSetEmgMode	设置 EMG 急停信号参数	√
	SMCGetEmgMode	读取 EMG 急停信号参数	X
	SMCSetIoDstpMode	设置 IO 触发减速停止电平信号	√
	SMCGetIoDstpMode	读取 IO 触发减速停止电平信号	X
	SMCGetDecStopTime	设置减速停止时间值	√
	SMCGetDecStopTime	读取减速停止设置时间值	X
	SMCGetStopReason	轴异常停止原因	X
	SMCClearStopReason	清除轴停止原因	√
轴 IO 映射	SMCSetAxisIoMap	设置轴 IO 映射关系	√
	SMCGetAxisIoMap	读取轴 IO 映射关系参数	X
虚拟轴 IO 映射	SMCSetIoMapVirtual	设置虚拟 IO 映射关系	√
	SMCGetIoMapVirtual	读取虚拟 IO 映射关系	X
	SMCReadInbitVirtual	读取虚拟 IO 电平状态	X
密码管理指令	SMCEnterPassword	输入密码	√
	SMCModifyPassword	修改密码	√
寄存器操作	MODBUS_BIT	修改或者读取 MODBUS BIT 位寄存器	√
	MODBUS_REG	修改或者读取 MODBUSREG 字寄存器	√
	MODBUS_LONG	修改或者读取 MODBUS 双字寄存器	√
	MODBUS_IEEE	修改或者读取双字寄存器，浮点数方式	√

	MODBUS_STRING	修改或者读取多个寄存器，字符串方式	√
	SMCModbusSetString	设置寄存器中的数值	X
	SMCModbusGetString	读取寄存器中的数值	X
G 代码操作指令	SMCGcodeSetCurrentFile	设置为当前加工文件	√
	SMCGcodeStart	启动程序	√
	SMCGcodePause	暂停程序	√
	SMCGcodeStop	停止程序	√
	SMCGcodeState	读取运动状态	√
	SMCGcodeSetStepState	设置单步运行状态	√
	SMCGcodeGetStepState	读取单步运行状态	X
	SMCGcodeStopReason	读取停止原因	X
	GCode_LinerNO	设置 G 插补坐标系	√
	GCode_LocateSpeed	设置 G 指令定位运行速度	√
	GCode_LocateTacc	设置 G 指令定位加减速时间	√
	GCode_PathSpeed	设置 G 指令插补运行速度	√
	GCode_PathTacc	设置 G 指令插补加减速时间	√
	SMCGcodeCurrentLine	读取当前行状态	√
	SMCGcodeGetLine	读取行字符数组	X
	SMCGcodeGetLineArray	读取行数据数组	X
	SMCGcodeDeleteLine	删除一行 G 代码	√
	SMCGcodeAddLine	添加一行 G 代码字符串或字符数字	√
	SMCGcodeAddLineArray	添加一行 G 代码数据数组	X
	SMCGcodeModifyLine	修改一行 G 代码字符串或字符数组	√
	SMCGcodeModifyLineArray	修改一行 G 代码数据数组	X
	SMCGcodeInsertLine	插入 G 代码字符	√
	SMCGcodeInsertLineArray	插入 G 代码数组	X
	SMCGcodeCheckFile	检查 G 文件状态	X
	SMCGcodeSetCurFile	设置 G 代码运行文件	√
	SMCGcodeSaveFile	保存 G 代码文件	√

	SMCGcodeCreatFile	创建 G 代码文件	√
	SMCGcodeDeleteFile	删除 G 代码文件	√
	SMCGcodeDeleteFileId	按文件号删除 G 代码文件	√
	SMCGcodeCopyFile	复制 G 代码文件	√
	SMCGcodeGetCurrentFile	读取当前 G 代码文件信息	X
	SMCGcodeGetFirstFile	读取第一个 G 代码文件	X
	SMCGcodeGetNextFile	读取下一个 G 代码文件	X
	SMCGcodeGetFile	设置 G 代码运行文件	X
CAN 指令	SETCAN	设置 CAN 口的通讯参数	X
	SMCSetPulseOutmodeCan	设置指定轴的脉冲输出模式	√
	SMCGetPulseOutmodeCan	读取指定轴的脉冲输出模式	X
	SMCSetEquivCan	设置脉冲当量值	√
	SMCGetEquivCan	读取脉冲当量值	X
	SMCSetProfileCan	设置单轴运动速度曲线参数	√
	SMCGetProfileCan	读取单轴运动速度曲线参数	X
	SMCSetSprofileCan	设置单轴速度曲线 S 段参数值	√
	SMCGetSprofileCan	读取单轴速度曲线 S 段参数值	X
	SMCPmoveCan	启动定长运动	√
	SMCVmoveCan	指定轴恒速运动	√
	SMCSetHomePinLogicCan	设置 ORG 原点信号参数	√
	SMCGetHomePinLogicCan	读取 ORG 原点信号参数	X
	SMCSetHomeProfileCan	设置 ORG 原点速度参数	√
	SMCGetHomeProfileCan	读取 ORG 原点速度参数	X
	SMCSetHomemodeCan	设置回原点模式	√
	SMCGetHomemodeCan	读取回原点模式参数	X
	SMCHomeMoveCan	回原点运动	√
	SMCReadInbitCan	读取某个输入端口的电平状态	X
	SMCWriteOutbitCan	设置指定控制器的某个输出端口的电平状态	√

	SMCReadOutbitCan	读取某个输出端口的电平	X
	SMCReadInportCan	读取全部输入端口的电平	X
	SMCWriteOutportCan	设置指定控制器的全部输出口的电平	√
	SMCReadOutportCan	设置指定控制器的全部输出口的电平	X
	SMCSetAlmModeCan	设置指定轴的 ALM 信号参数	√
	SMCGetAlmModeCan	读取指定轴的 ALM 信号参数	X
	SMCSetInpModeCan	设置指定轴的 INP 信号	√
	SMCGetInpModeCan	读取指定轴的 INP 信号	X
	SMCSetEmgModeCan	设置 EMG 急停信号	√
	SMCGetEmgModeCan	读取 EMG 急停信号	X
	SMCSetElModeCan	设置 EL 限位信号参数	√
	SMCGetElModeCan	读取 EL 限位信号参数	X
	SMCSetSoftlimitCan	设置软限位参数	√
	SMCGetSoftlimitCan	读取软限位参数	X
	SMCSetModbus0xCan	设置 MODBUS 位寄存器	√
	SMCGetModbus0xCan	读取 MODBUS 位寄存器	X
	SMCSetModbus4xCan	设置寄存器值	X
	SMCGetModbus4xCan	读取寄存器值	X
	SMCSetPositionCan	设置当前指令位置计数器值	√
	SMCGetPositionCan	读取当前指令位置值	X
	SMCAxisIoStatusCan	读取指定轴有关运动信号的状态	X
	SMCStopCan	指定轴停止运动	√
	SMCSetDecStopTimeCan	设置异常停止时减速时间	√
	SMCGetDecStopTimeCan	读取异常停止时减速时间值	X
	SMCEmgStopCan	紧急停止所有轴	√
	SMCGetTotalAxesCan	读取控制器使用的最多轴数	X
	SMCCheckDoneCan	检测指定轴的运动状态	X
	SMCReadCurrentSpeedCan	读取当前轴运动速度值	X
	SMCGetStopReasonCan	轴异常停止原因	X

	SMCClearStopReasonCan	清除当前轴错误原因	√
自由协议指令	SMCComSetMode	设置串口通讯方式	√
	SMCComWrite	发送串口字节数据	X
	SMCComRead	读取串口字节数据	X
	Inet_Addr	字符串 IP 地址转换为内部数值 IP 地址	√
	SMCEthSetMode	设置网络模式	√
	SMCEthGetState	读取网络状态	√
	SMCEthRead	读取网络数据	√
	SMCEthWrite	写入网络数据	√
	SMCCanSetMode	设置串口通讯方式	√
	SMCCanWrite	发送 VAN 口字节数据	√
	SMCCanRead	读取 CAN 口字节数据	√
总线相关指令	NMCSResetCANopen	CANopen总线主站复位	√
	NMCSStopETC	停止EtherCAT总线	√
	NMCSGetEmergeneyMessege_Nodes	获取紧急报文报文信息	√
	NMCSSetNodeOd	设置从站对象字典	√
	NMCSGetNodeOd	获取从站对象字典	√
	NMCSSendNmtCommand	发送NMT管理报文	√
	NMCSGetCycleTime	读取EtherCAT总线循环周期	√
	NMCSWriteRxpdoExtra	写扩展rxpdo	√
	NMCSReadRxpdoExtra	读扩展rxpdo	√
	NMCSReadTxpdoExtra	读扩展txpdo	√
	NMCSGetAxisType	读取轴类型	√
	NMCSSynMoveUnit	同步运动	√
	NMCSGetAxisStateMachine	获取EtherCAT总线轴状态机	√
	NMCSGetAxisControlMode	读取EtherCAT总线轴控制模式	√
	NMCSGetTotalAxes	获取控制器轴数	√
	NMCSGetTotalIonum	获取控制器通用I/O点数	√

EtnerCAT 相关指令	NMCSGetTotalAdcnum	获取总线卡模拟量I/O点数	√
	NMCSGetTotalIONum	读取EtherCAT总线IO输入输出口数	√
	NMCSGetTotalSlaves	获取EtherCAT从站总数	√
	NMCSGetAxisIOOut	设置EtherCAT轴数字量输出IO状态	√
	NMCSGetAxisIOIn	获取EtherCAT轴数字量输入IO状态	√
	NMCSWriteOutbit	设置指定控制器或扩展模块的某个输出端口的电平	√
	NMCSReadOutbit	读取指定控制器或扩展模块的某个输出端口的电平	√
	NMCSReadInbit	读取指定控制器或扩展模块的某个输入端口的电平	√
	NMCSReadInport	读取指定IO组号的全部输入口的电平	√
	NMCSReadOutport	读取指定 IO 组号的全部输出口的电平	√
	NMCSSetAlarmClear	清除报警信号	√
	NMCSGetErrcode	获取 EtherCAT 总线状态	√
	NMCSGetCardErrcode	获取总线错误代码	√
	NMCSClearCardErrcode	清除指定连接号总线错误代码	√
	NMCSGetAxisErrcode	获取轴错误代码	√
	NMCSClearAxisErrcode	清除轴错误代码	√

表格 2：G 代码指令一览表

G指令	功能	G指令	功能
G00	快速定位	M30	程序结束并循环
G01	直线插补	M80	输出口开
G02	顺圆插补	M81	输出口关
G03	圆弧插补 逆圆	M82	等待输入口有效
G04	延时	M83	等待输入口无效
G05	三点圆弧插补	M90	局部循环结束

G06	三点螺旋插补	M91	局部循环开始
G26	回原点	M84	恒速运动
G28	回工件零点	M85	关闭恒速运动
G53	机械坐标	M98	子程序调用
G54	还原为工件坐标	M99	子程序返回
G90	绝对坐标	M02	程序结束
G91	相对坐标	M11	输出口2 开
G92	重定义坐标	M12	输出口2 关
F	F指令，速度设置	M86	变量加一个数
M00	程序暂停	M87	变量赋值
M07	输出口0 开	M92	强制修改工件坐标
M08	输出口0 关	M94	根据条件跳转
M09	输出口1 开	M95	强制跳转
M10	输出口1 关	M96	根据条件调用子程序

表格 3：老版本 BASIC 指令一览表

雷赛控制器可同时兼容老版本指令，建议使用最新指令。

老版本 BASIC 指令具体描述请参考雷赛老一代控制器 SMC6490 等的 BASIC 编程手册《BASIC 语言编程手册 V2.1》

功能	名称	功能说明
点位运动指令及参数	BASE	选择要参与运动的轴号
	PULSET	设置电机脉冲信号输出的模式
	MPOS	修改或者读取轴的当前坐标
	UNITS	设置或读取轴运动的脉冲当量参数
	STARTSPEED	单轴运动起始速度参数
	STOPSPEED	单轴运动停止速度参数
	SPEED	单轴运动最大速度参数
	SCURVESET	设置S形速度曲线的时间
	CURSPEED	读取轴当前速度
	ACC	单轴运动加速度参数
	DEC	单轴运动减速度参数
	TACC	单轴运动加速时间参数
	TDEC	单轴运动减速时间参数
	PMOVEABS	绝对坐标下运动到指定的坐标
	PMOVE	相对坐标下运动到指定距离
	PMOVE_MODIFY	修改上次PMOVE或PMOVEABS指令的终点坐标
	AXISSTOP	BASE轴列表减速停止
	RAPIDSTOP	BASE轴列表立即停止
	IDLE	读取轴的运动是否停止
	WAITIDLE	等待BASE指令列表中的轴运动完成

	DIS_LEFT	读取当指定轴的PMOVE运动指令剩余的距离
	BACKLASH	设置各轴的反向间隙补偿量
	MAX_AXISES	读取系统最多轴数
恒速运动指令	FORWARD	BASE指令中所列的轴以speed参数设定的速度，正向恒速运动
	REVERSE	BASE指令中所列的轴以speed指令设定的速度，反向恒速运动
	VMOVE	轴往一个方向恒速运动
插补运动指令	VMODE	插补模式：0-非前瞻,1-前瞻
	VSTARTSPEED	插补运动的起跳速度参数
	VSPEED	插补运动的速度参数
	VSTOPSPEED	插补运动的停止速度参数
	VSCURVESET	插补运动的S段时间参数
	CURVSPEED	读取当前插补系的当前插补速度
	VTACC	插补运动的加速时间参数
	VACC	插补运动的加速度参数
	VTDEC	插补运动的减速时间参数
	VDEC	插补运动的减速度参数
	COOR	改变当前插补坐标系
	MOVE	相对坐标下，多轴进行直线插补运动
	MOVEABS	2～6轴进行直线插补运动；采用绝对坐标
	MOVECIRC	相对坐标下，圆心+终点模式下的圆弧插补
	MOVECIRCABS	绝对坐标下，圆心+终点模式下的圆弧插补
	MOVECIRC2	相对坐标下，三点模式下圆弧插补
	MOVECIRC2ABS	绝对坐标下，三点模式下圆弧插补
	MOVECIRC3	相对坐标下，半径+终点模式下圆弧插补
	MOVECIRC3ABS	绝对坐标下，半径+终点模式下圆弧插补
连续插	CONTI_ENTER	当前插补坐标系进入连续插补状态

补运动指令	CONTI_END	当前插补坐标系退出连续插补状态
	CONTI_LIMIT	在当前的插补目标位置加一个速度限制
	CONTI_MARK	插补段的编号
	CONTI_CURMARK	读取当前运行的段标识
	MOVE_PAUSE	暂停当前插补坐标系
	MOVE_START	启动当前插补坐标系
	MOVE_STOP	停止当前插补坐标系
	CONTI_BUFF	读取剩余插补缓冲个数
	CONTI_IFEND	读取插补结束状态
	CONTI_TOTAL	读取已加入插补缓冲区总的矢量距离
	CONTI_DIS	读取插补已运行矢量距离
	CONTI_LEFT	读取插补剩余矢量距离
输入输出IO	IN	读取一个或多个通用输入端口状态
	OUT	设置或者读取输出端口状态
	IFALM	是否使用伺服电机的报警信号ALARM
	IN_ALARM	读取某一个轴的伺服电机报警信号ALARM的输入状态
	ALM_INVERT	设置伺服电机报警信号ALARM的有效电平
	IFINP	是否使用伺服电机到位信号INP
	IN_INP	读取某一个轴的伺服电机到位信号INP的输入状态
	INP_INVERT	设置伺服电机到位信号INP的有效电平
	IN_EZ	读取某一个编码器的零点信号EZ的状态
	INVERT_IN	翻转输入信号的电平状态
	INVERT_OUT	翻转输入信号的电平状态
回原点指令	HOMEMOVE	启动回原点运动
	HOMESPEEDMODE	设置回原点起始速度模式
	HOMESPEED	设置回原点速度
	HOMEDIR	设置回原点方向

	HOMEMODE	设置回原点模式
	IN_HOME	读取某一个轴的HOME传感器信号的输入状态
	HOME_INVERT	设置HOME信号的有效电平
限位指令	IFEL	是否使用限位信号EL
	EL_INVERT	限位信号EL的有效电平
	IN_ELFWD	读取某一个轴的正向限位传感器的输入状态
	IN_ELREV	读取某一个轴的负向限位传感器的输入状态
	IFEL_DEC	限位信号触发后是否为减速停止
	IFSOFTEL	是否使用软限位
	SOFTEL_FWD	设置正向软限位坐标
	SOFTEL_REV	设置反向软限位坐标
	IN_SOFTFWD	读取某一个轴的正向软限位状态
	IN_SOFTREV	读取某一个轴的反向软限位状态
	IFSOFT_ENCODE	是否用编码器来进行软限位
	IFSOFTEL_DEC	软限位信号触发后是否为减速停止
编码器	EPOS	修改或者读取当前编码器坐标
	ENCODESET	设置编码器信号参数
	LPOS	读取当前编码器坐标的锁存值
	ENCODE_LATCHSTATE	读取和清除编码器的锁存状态

附录 1：BASIC 指令运行错误一览表

当 BASIC 程序出现语法错误或者参数范围错误等情况时，BASIC 程序会被停止运行，同时会有错误的位置和信息输出。详见下表。

错误代码	错误号	可能错误原因
ERR_AXIS_SEL_ERR	30	手轮脉冲的轴档位选择超出范围（软件控制模式）
ERR_HAND_AXIS_NUM_ERR	31	手轮脉冲的轴映射数量超出范围
ERR_AXIS_RATIO_ERR	32	手轮脉冲的倍率档位选择超出范围（软件控制模式）
ERR_HANDWH_START	33	已进入手轮脉冲模式，不能切换软硬件控制模式
ERR_AXIS_BUSY_STATE	34	轴已在运动，不能切换到手轮模式
ERR_LIST_NUM_ERR	50	LIST 号超出范围
ERR_LIST_NOT_OEPN	51	LIST 没有初始化
ERR_PARA_NOT_VALID	52	参数不在有效范围
ERR_LIST_HAS_OPEN	53	LIST 已经打开
ERR_MAIN_LIST_NOT_OPEN	54	LIST 没有初始化
ERR_AXIS_NUM_ERR	55	轴数不在有效范围
ERR_AXIS_MAP_ARRAY_ERR	56	轴映射表为空
ERR_MAP_AXIS_ERR	57	映射轴错误
ERR_MAP_AXIS_BUSY	58	映射轴忙
ERR_PARA_SET_FORBIT	59	运动中不允许更改参数
ERR_FIFO_FULL	60	缓冲区已满
ERR_RADIUS_ERR	61	半径为 0 或小于两点的距离的一半
ERR_MAINLIST_HAS_START	62	LIST 已经启动
ERR_ACC_TIME_ZERO	63	加减速时间为 0
ERR_MAINLIST_NOT_START	64	主要 LIST 没有启动
ERR_POINT_SAME_ON_RADIUS	67	圆弧插补在半径模式下起点和终点不能重合
MCVP_SMOOTH_TIME_SET_ERROR	80	s 时间设置错误(小于等于 0)

MCVP_START_VEL_SET_ERROR	81	起始速度绝对值设置错误(小于 0)
MCVP_STEADY_VEL_SET_ERROR	82	最大速度绝对值设置错误(小于等于 0)
MCVP_END_VEL_SET_ERROR	83	停止速度绝对值设置错误(小于 0)
MCVP_TOTAL_LENGTH_SET_ERROR	84	运动距离为 0，无法运动
ERR_AXIS_INDEX	101	所选轴超出最大值
ERR_SET_WHILE_MOVING	102	轴正在运动，不能设置参数
ERR_ENTER_WHILE_MOVING	103	轴正在运动，不能进入该模式
ERR_PEL_STATE	104	轴处于正限位，不能正向运动
ERR_NEL_STATE	105	轴处于负限位，不能负向运动
ERR_SOFT_PEL_STATE	106	轴处于软正限位，不能正向运动
ERR_SOFT_NEL_STATE	107	轴处于软负限位，不能负向运动
ERR_FORCE_IN_OTHER_MODE	108	轴处于非点位模式，不能强制变位
ERR_MAX_VEL_ZERO	109	设置最大速度错误，不能为 0
ERR_EQU_ZERO	110	设置轴当量错误，不能为 0
ERR_BACKLASH_NEG	111	设置反向间隙错误，不能为负值
ERR_MAX_PULSE	112	设置位置错误，已超出允许范围
ERR_CMP_EXCEED_MAX_AXISES	121	所选比较轴超出范围
ERR_CMP_EXCEED_MAX_INDEX	122	比较点数已满，不能继续添加
ERR_CMP_EXCEED_MAX_IO	123	进行比较的 IO 超出范围
ERR_CMP_EXCEED_MAX_CHAN	124	选择的高速比较 IO 超出范围
ERR_MAP_AXISIO_MAX_AXISES	130	映射的轴超出范围
PVT_ERROR_AXISES_OVER_RANGE	140	所选轴超出范围
PVT_ERROR_INDEX_OVER_RANGE	141	控制点已满，不能继续添加
PVT_ERROR_INDEX_EXCEED	142	控制点已满，不能继续添加
PVT_ERROR_TIME_EROOR	143	插入段时间为 0 或者负数
HOME_ERROR_AXISES_OVER_RANGE	200	所选轴超出最大值
HOME_ERROR_MAX_VEL	202	设置的最大速度为 0

HOME_ERROR_MAX_ACC	203	设置的加速度小于等于 0
HOME_ERROR_BOTH_LIMIT	207	同时处于正、负限位，无法启动回 0 运动
ERROR_ZSHELL_PARAERR	1000	参数错误
ERROR_ZSHELL_STOP_USER	1040	用户手动停止
ERROR_ZSHELL_STOP_OTHERTASK	1041	其他任务牵连停止
ERROR_ZSHELL_PARA_CANNOT_MODIFIY	1042	少数参数不让修改, SET 扩展返回
ERROR_ZSHELL_ARRAY_OVER	1043	数组越界
ERROR_ZSHELL_VAR_TOOMUCH	1044	变量个数超过
ERROR_ZSHELL_ARRAY_TOOMUCH	1045	数组个数超过
ERROR_ZSHELL_ARRAY_NOSPACE	1046	数组没有空间
ERROR_ZSHELL_SUB_TOOMUCH	1047	SUB 过多
ERROR_ZSHELL_LABEL_NAMEERR	1048	标识符 命名错误
ERROR_ZSHELL_LABEL_TOOMANYCHARES	1049	标识符 命名过长
ERROR_ZSHELL_NO_RIGHTBRACKET	1050	没有右括号
ERROR_ZSHELL_UNKOWNCHAR	1051	不认识的字符
ERROR_ZSHELL_UNKOWN_LABEL	1052	不认识的名称, 表达式中碰到
ERROR_ZSHELL_STOP_INVALIDCMD	1053	不认识的命令
ERROR_ZSHELL_STOP_OVERSTACK	1054	超过堆栈层数
ERROR_ZSHELL_OVER_RECURSION	1055	递归过多
ERROR_ZSHELL_NO_QUOTEEND	1056	引号没有结束
ERROR_ZSHELL_CMD_CANNOTREAD	1057	不能读取, 不能在表达式中使用
ERROR_ZSHELL_CMD_CANNOTRUN	1058	函数之类不能出现在行首, 只能在表达式中使用
ERROR_ZSHELL_LINE_MUST_END	1059	期望行结束, 一些特殊指令需要
ERROR_ZSHELL_ARRAY_NEEDINDEX	1060	数组需要编号， 参数也使用
ERROR_ZSHELL_NOTBRACKET_AFTERVAR	1061	变量后不需要编号
ERROR_ZSHELL_DIM_CONFLICT	1062	数组重新定义冲突, 长度不一致
ERROR_ZSHELL_DIM_ARRAYLENGTHERR	1063	数组长度错误

ERROR_ZSHELL_DIM_LABEL_SUB	1064	定义，名称 SUB
ERROR_ZSHELL_DIM_LABEL_PARA	1065	定义，名称 PARA
ERROR_ZSHELL_DIM_LABEL_RESERVE	1066	定义，名称 保留
ERROR_ZSHELL_UNWANT_CHAR	1067	不能出现的字符
ERROR_ZSHELL_STACK_NOPUSH	1068	POP 时没有压栈
ERROR_ZSHELL_FORMAT_ERR	1070	格式错误
ERROR_ZSHELL_SET_OVER	1071	参数溢出, para(10) 10 过大
ERROR_ZSHELL_PARA_RANGEERR	1072	一些函数和命令的参数范围错误
ERROR_ZSHELL_PARA_TOOMANY	1073	一些函数和命令的参数 过多
ERROR_ZSHELL_PARA_TOOFEW	1074	一些函数和命令的参数 过少
ERROR_ZSHELL_NO_EXPR	1075	读取不到表达式
ERROR_ZSHELL_OPERNOPARA	1076	操作符没有参数
ERROR_ZSHELL_NOPARA_BEFOREOPER	1077	操作符前面没有参数
ERROR_ZSHELL_SIGNAL_CANNOTINEXPR	1078	符号不能在表达式中使用
ERROR_ZSHELL_NEED_OPER	1079	需要双目操作符
ERROR_ZSHELL_SUB_NOTSUB	1080	CALL 的不是 SUB
ERROR_ZSHELL_NO_AUTO	1081	没有 AUTO 所以不启动
ERROR_ZSHELL_EQ_WANTED	1082	赋值语句没有等号，变量或者参数等只能为赋值语句
ERROR_ZSHELL_FILE_VAIN	1083	程序文件空
ERROR_ZSHELL_SUB_RENAME	1084	SUB 重名，包括与其他的名称重了
ERROR_ZSHELL_TASK_RUNNING	1085	任务已经运行中
ERROR_ZSHELL_NEED_COMMA_BETWEEN_PARA	1086	操作数之间需要逗号
ERROR_ZSHELL_NO_LEFTBRACKET	1087	没有右括号
ERROR_ZSHELL_TOOMANY_IFNESTED	1088	IF 嵌套太多
ERROR_ZSHELL_TOOMANY_LOOPNESTED	1089	LOOP 嵌套太多
ERROR_ZSHELL_MOVEAXISES_FEW	1090	插补轴太少

ERROR_ZSHELL_CONST_VAR	1091	变量不能修改
ERROR_ZSHELL_NOT_ONLINECMD	1092	不能作为在线命令
ERROR_ZSHELL_AXIS_OVER	1093	轴号超出
ERROR_ZSHELL_CRD_OVER	1094	插补系超出
ERROR_ZSHELL_STOP_UNKNOWN	1099	未知错误，不可能出现的，一般是内部错误引起
ERROR_ZSHELL_DIVISION_BY_ZERO	1200	除零错误，请排查除数是否为 0

附录 2：运动参数总表

控制器在设置参数等数据前，我们会提供一些默认值。默认值可通过在线输入调试指令 PRINT *SET 打印出系统的默认初始值。见下表 1 控制器 SMC606 参数值。

我们也可通过在线输入调试指令 Print Basic_Max_Inf 打印出控制器配置的最大参数值，如最大轴数、最大定时器数等。见下表 2 控制器 SMC606 参数的参数值。

表 1： Print *Set 详细说明

参数名称	中文解释	出厂设置	参数范围	适用范围
PULSET	脉冲模式	0	[0,6]	
UNITS	脉冲当量	1	>=1	
BACKLASH	反向间隙	0	>0	
ENCODESET	编码器计数方式	1	[0,3]	
ENCODEREVERSE	编码器计数反向	0	[0,1]	SMC104 除外
STARTSPEED	点位起始速度	0		
SPEED	点位运行速度	10000		
STOPSPEED	点位停止速度	0		
ACC	点位加速度	4000000		
DEC	点位减速度	4000000		
TACC	点位加速时间	0.001		
TDEC	点位减速时间	0.001		
SCURVESET	点位 S 段时间	0		
PLANMODE	点位规划模式	0	[0,1]	SMC104 除外
HOMESPEED_LOW	回零低速值	0		
HOMESPEED	回零高速值	0		
HOMETACC	回零加速时间	0.001		
HOMEMODE	回零模式	0		
HOMEDIR	回零方向	0	[0,1]	

HOME_INVERT	原点有效电平	0	[0,1]	
HOMEFINISHMODE	回零完成处理方式	1	[0,2]	SMC104 除外
HOMEOFFSETPOS	回零完成后偏移位置	0		SMC104 除外
HOMINGLATCHSRC	回零锁存位置源	0	[0,1]	SMC104 除外
EZ_INVERT	EZ 有效电平	0	[0,1]	
IFSOFTEL	软限位是否使能	0	[0,1]	
IFSOFT_ENCODE	软限位是否参考编码器位置	0	[0,1]	
IFSOFTEL_DEC	软限位是否减速停止	1	[0,1]	
SOFTEL_FWD	软限位正向值	1000		
SOFTEL_REV	软限位负向值	0		
IFEL	硬限位是否使能	1	[0,1]	
IFEL_DEC	硬限位是否减速停止	0	[0,1]	
EL_INVERT	硬限位有效电平	0	[0,1]	
IFEMG	紧急停止是否使能	0	[0,1]	
EMG_INVERT	紧急停止有效电平	0	[0,1]	
IFALM	伺服报警是否使能	1	[0,1]	SMC104 除外
ALM_INVERT	伺服报警有效电平	0	[0,1]	SMC104 除外
IFINP	伺服到位是否使能	0	[0,1]	SMC104 除外
INP_INVERT	伺服到位有效电平	0	[0,1]	SMC104 除外
IFDSTP	异常减速停止是否使能	0	[0,1]	SMC104 除外
DSTP_INVERT	异常减速停止有效电平	0	[0,1]	SMC104 除外
DSTP_TIME	异常减速停止时间	0		SMC104 除外
PEL_MAPTYPE	正限位映射输入接口类型	0		SMC104 除外
PEL_MAPINDEX	正限位映射输入接口序号	1		SMC104 除外
PEL_FILTER	正限位信号滤波时间	0		SMC104 除外
NEL_MAPTYPE	负限位映射输入接口类型	1		SMC104 除外
NEL_MAPINDEX	负限位映射输入接口序号	1		SMC104 除外

NEL_FILTER	负限位信号滤波时间	0		SMC104 除外
ORG_MAPTYPE	原点映射输入接口类型	2		SMC104 除外
ORG_MAPINDEX	原点映射输入接口序号	1		SMC104 除外
ORG_FILTER	原点信号滤波时间	0		SMC104 除外
EMG_MAPTYPE	急停映射输入接口类型	0		SMC104 除外
EMG_MAPINDEX	急停映射输入接口序号	1		SMC104 除外
EMG_FILTER	急停信号滤波时间	0		SMC104 除外
DSTP_MAPTYPE	异常减速停止输入接口类型	6		SMC104 除外
DSTP_FILTER	异常减速停止滤波时间	0		SMC104 除外
ALM_MAPTYPE	伺服报警映射输入接口类型	3		SMC104 除外
ALM_MAPINDEX	伺服报警映射输入接口序号	1		SMC104 除外
ALM_FILTER	伺服报警信号滤波时间	0		SMC104 除外
INP_MAPTYPE	伺服到位映射输入接口类型	5		SMC104 除外
INP_MAPINDEX	伺服到位射输入接口序号	1		SMC104 除外
INP_FILTER	伺服到位信号滤波时间	0		SMC104 除外
VMODE	插补模式	0	[0,2]	SMC104 除外
VSTARTSPEED	插补起始速度	0		
VSPEED	插补运行速度	10000		
VSTOPSPEED	插补停止速度	0		
VACC	插补加速度	10000		
VDEC	插补减速度	10000		
VTACC	插补加速时间	0.01		
VTDEC	插补减速时间	0.01		
VSCURVESET	插补 S 段时间	0		SMC104 除外
VAHEADSEGMENT	插补前瞻段数	4900	[2,5000]	SMC600 支持
VAHEADACC	插补前瞻拐角加速度	90000		SMC600 支持
VAHEADPATHERROR	插补前瞻拐角轨迹误差	10		SMC600 支持

VIFARCLIMIT	插补圆弧是否限速	0		SMC600 支持
GCODE_SYSTYPE	G 代码系统类型	0	[0,1]	SMC104 除外
GLUE_IFCONTRLIO	点胶过程是否操作 IO	0	[0,1]	SMC104 除外
GCODE_IFWORKZERO	G 代码是否启用工件坐标系	0	[0,1]	SMC104 除外
GCODE_WORKZERO	G 代码工件原点	0		SMC104 除外
GCODE_DEFAULTFILEID	G 代码默认文件号	0	[0,999]	SMC104 除外
GCODE_IFRELATIVE	G 代码是否启用相对坐标	0	[0,1]	SMC104 除外
GCODE_IFGOOUSELINE	G 代码中 G00 是否用直线执行	0	[0,1]	SMC104 除外
GCODE_LINERNO	G 代码启用坐标系号	0	[0,3]	SMC104 除外
GCODE_IFCONTINUE	G 代码是否采用连续插补	1	[0,1]	SMC104 除外
GCODE_AXISNUM	G 代码执行轴数	3		SMC104 除外
GCODE_AXISLIST	G 代码轴映射列表	0,1,2		SMC104 除外
GCODE_LOCATESTARTSPEED	G 代码单轴定位起始速度	0		SMC104 除外
GCODE_LOCATESPEED	G 代码单轴定位速度	10		SMC104 除外
GCODE_LOCATETACC	G 代码单轴定位加减速时间	0.1		SMC104 除外
GCODE_PATHSTARTSPEED	G 代码轨迹起始速度	0		SMC104 除外
GCODE_PATHSPEED	G 代码轨迹速度	10		SMC104 除外
GCODE_PATHTACC	G 代码轨迹加减速时间	0.1		SMC104 除外
GLUE_IFOPENGLUE	点胶是否开胶	0	[0,1]	SMC104 除外
GLUE_AHEAD_OPEN	提前开胶距离	0		SMC104 除外
GLUE_OPEN_DELAY	开胶延时时间	0		SMC104 除外
GLUE_AHEAD_CLOSE	提前关胶距离	0		SMC104 除外
GLUE_CLOSE_DELAY	关胶延时时间	0		SMC104 除外
GLUE_UP_HIGH	上抬高度	0		SMC104 除外
GLUE_UP_STARTSPEED	上抬起始速度	0		SMC104 除外
GLUE_UP_SPEED	上抬速度	10		SMC104 除外
GLUE_UP_TACC	上抬加减速时间	0.1		SMC104 除外

GLUE_DRAW_HIGH	拉丝高度	0		SMC104 除外
GLUE_DRAW_DIST	拉丝距离	0		SMC104 除外
GLUE_DRAW_STARTSPEED	拉丝起始速度	0		SMC104 除外
GLUE_DRAW_SPEED	拉丝速度	10		SMC104 除外
GLUE_DRAW_TACC	拉丝加减速时间	0.1		SMC104 除外
GLUE_IO_DEFAULTNO	默认胶头	0	[0,3]	SMC104 除外
GLUE_IO_IFDISPOFFSET	是否修正胶枪偏移位置	0	[0,1]	SMC104 除外
GLUE_IO_NO	胶头输出口选择	0		SMC104 除外
GLUE_IO_INVERT	开胶有效电平	0		SMC104 除外
GLUE_IO_XOFFSET	X 方向偏移值	0		SMC104 除外
GLUE_IO_YOFFSET	Y 方向偏移值	0		SMC104 除外
GLUE_IO_ZOFFSET	Z 方向偏移值	0		SMC104 除外

表 2： print Basic_Max_Inf 详细说明

指令名称	名称	默认值
MaxTaskNum	最大任务数	6
MaxTimerNum	最大定时器数	5
MaxVarNum	最大变量数	20480
MaxArrayNum	最大数组数	1024
MaxArraySpaceNum	最大数组空间	200000
MaxSubNum	最大子程序个数	512
MaxStackNum	最大堆栈数	100
MaxExpressionLayerNum	最大表达式递归层数	100
MaxLabelCharNum	最大字符串字符个数	54
MaxModbusBitNum	位寄存器个数	10000
MaxModbusRegNum	字寄存器个数	10000

深圳市雷赛控制技术有限公司
SHENZHEN LEADSHINE CONTROL TECHNOLOGY CO.,LTD

深圳市雷赛智能控制股份有限公司

地　址：深圳市深圳市南山区学苑大道 1001 号南山智园 A3 栋 9 楼

邮　编：518052

电　话：0755-26415968

传　真：0755-26417609

Email：info@szleadtech.com.cn

网　址：http://www.szleadtech.com.cn