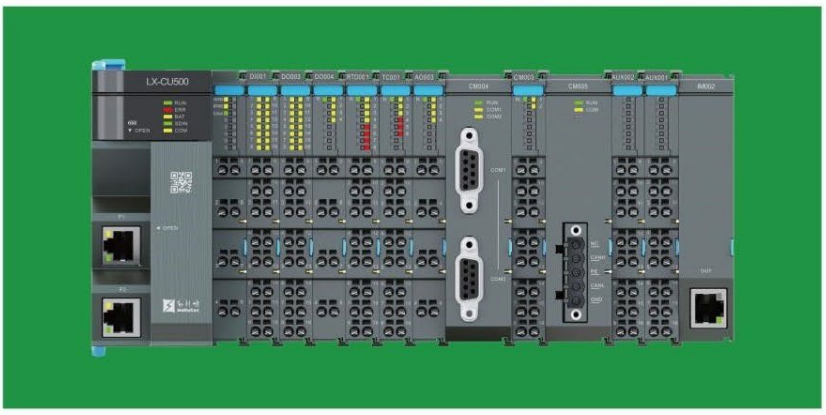


LX系列可编程控制器
PLCopen指令手册

版本：2024.09

版权声明

地址：北京经济技术开发区地盛中路 2 号院

邮编：100176

商务电话：010-5898 1588

产品咨询热线：400-811-1999

技术支持电话：010-58981514

传真：010-5898 1558

网址：http://www.hollysys.com/

Email：PLC@hollysys.com

新浪微博：http://weibo.com/hollysysplc

目录

第1章　前言

1.1　本文档用途

本文档将介绍 LX 系统中 PLCopen 运动指令的使用，包含参数、功能、示例、使用注意事项、参见等内容。请结合其他配套产品资料一起使用，帮助您更全面的了解指令使用方法。

1.2　阅读对象

本手册供具有自动化及控制系统专业知识的工程师或相关专业技术人员使用。

1.3　对象产品

本手册适用的产品对象如下：

LX 系列 PLC 产品

1.4　使用声明

本手册内容都按规定经过测试，图表与所述软件和硬件产品相符，但误差不可避免，并不能保证完全一致。如果您有关于改进或更正本手册内容的任何建议，请及时告知，我们将在下个版本进行修正。

因产品迭代更新，本手册中的相关参数和信息有可能会发生变更，恕不事先通知。

由于本手册中所描述的设备有多种使用方法，用户以及设备使用责任人必须保证每种方法的可容许性。对由使用或错误使用这些设备造成的任何直接或间接损失，和利时公司将不负法律责任。

由于实际应用时的不确定因素，和利时公司不承担直接使用本手册中提供的数据的责任。

1.5 安全注意事项

1.5.1 安全声明

使用本产品前，请先阅读并遵守本安全注意事项。

为保障人身和设备安全，在使用本产品时，请严格遵守产品上标识信息及手册中的安全注意事项。

手册中的“注意”、“小心”、“警告”和“危险”事项，并不代表所应遵守的所有安全事项，只作 为所有安全注意事项的补充。

本产品应在符合设计规格要求的环境下使用，否则可能造成故障，因未遵守相关规定引发的设备损坏不在产品质量保证范围之内。

对于任何未按本手册规定操作或不符合要求的人员操作导致的人身安全事故和财产损失，和利时公司概不负责。

1.5.2 警告提示

为了您的人身安全以及避免财产损失，必须注意本手册中的警告提示。以下警告提示根据危险等级由高到低表示。当存在多个危险等级时，仅提示最高等级，如果最高等级为导致人身伤害的警告提示，则同时可能存在导致财产损失的警告。

- 人身安全提示：用一个警告三角  表示。
- 财产损失提示：不带警告三角。

 危险
表示如果不按规定操作，将会导致死亡或者严重的人身伤害。

 警告
表示如果不按规定操作，可能导致死亡或者严重的人身伤害。

 小心
表示如果不按规定操作，可能导致轻微的人身伤害。

注意
表示如果不按规定操作，可能导致财产损失。

第2章　文档指南

2.1　相关手册

本产品的手册种类如下表所示，请根据使用目的选读。

手册名称	用途	内容
LX 系列可编程控制器模块手册	了解 LX 系统硬件模块相关信息	对所有硬件模块进行说明，内容包含： 模块功能 硬件接口 接线 指示灯 技术参数
LX 系列可编程控制器通信手册	指导用户搭建 LX 系统通信网络	对 LX 系统所支持的通信网络进行说明，内容包含： 通讯协议介绍以及功能使用
LX 系列可编程控制器编程手册	了解 FA-AutoThink 编程软件的使用	对 FA-AutoThink 编程软件的界面、功能及相关操作进行说明，内容包含： 软件界面介绍 工程创建 编程 在线调试功能
LX 系列可编程控制器基本指令手册	了解 LX 系统基本控制指令的使用	对基本控制指令的使用进行说明，内容包含： 指令功能 参数 示例
LX 系列可编程控制器运动指令手册	了解 LX 系统运动控制指令的使用	对运动控制指令的使用进行说明，内容包含： 指令功能 参数 示例

2.2　使用约定

在本手册中使用以下符号：

- 提示：该符号表示有用的技巧或建议，以便更高效地使用产品。

2.3 关于手册获取

如需获得电子版 PDF 手册文件，可通过以下方式获取：

- 登陆和利时公司官网（https://cn.hollysys.com/other/download.html）下载。

第3章　手册修订记录

手册版本	修订日期	修订内容
V1.0	2024 年 9 月	创建

第4章　指令概述

4.1　指令列表

指令库	指令	名称	功能简介
单轴运动指令	SMC_AxisInit	轴初始化配置	用于初始化轴配置，通过该功能块可以设置轴运行所需的基本配置，如轴类型、用户单位换算、限位开关通道等
	MC_Power	轴使能	控制伺服使能状态指令
	MC_Home	原点回归	单轴原点回归指令
	MC_Stop	轴停止	控制轴按照指定减速度停止，中止该轴上任何正在执行的运动控制指令，并撤销该轴上所有运动缓存指令
	MC_MoveJog	Jog 点动	单轴进行正向/负向的点动指令
	MC_MoveAbsolute	绝对定位	单轴绝对定位指令，控制轴运动至指定的绝对位置
	MC_MoveRelative	相对定位	单轴相对定位指令，控制轴以增量方式运动至指定位置
	MC_MoveVelocity	速度控制	单轴速度控制指令，在位置控制模式下模拟速度控制
	SMC_SyncMoveVelocity	周期同步速度控制	周期同步速度模式下的速度控制
	MC_TorqueControl	力矩控制	单轴力矩控制指令，在力矩模式下对轴进行带斜坡的扭矩控制
	SMC_SetTorqueLimit	轴力矩限制值设置	设定伺服正向/负向扭矩的限制值
	MC_SetPosition	设定当前位置	以绝对或相对的方式重新定义指定轴的当前位置
	MC_SetOverride	运动倍率设置（超驰）	该指令设定速度、加减速、加加速度比例因子，可按设定比例放大或缩小轴运行的速度、加减速、加加速度
	SMC_SetDynamicLimits	运动参数限制值设置	该指令设定轴运动参数的限制值
	MC_ReadDigitalInput	读取轴数字量输入	读取伺服设备的数字量输入
	MC_ReadActualPosition	读取轴实际位置	该指令读取轴实际位置
	MC_ReadActualVelocity	读取轴实际速度	读取轴的实际速度
	MC_ReadActualTorque	读取轴实际力矩	读取轴的实际力矩
	MC_ReadStatus	读取轴状态	读取轴 PLCopen 状态机的状态

	MC_ReadAxisInfo	读取轴信息	用于读取指定轴的相关信息
	MC_ReadAxisError	读取轴错误	读取轴错误信息（非功能块错误）
	SMC_ReadAxisLimitStatus	读取轴限位状态	用于读取指定轴的正负硬限位状态和正负软限位状态
	MC_Reset	复位轴错误	复位轴错误
	MC_TouchProbe	轴位置锁存	该指令用于检测到指定的探针事件触发时，锁存指定轴和锁存通道的位置
	MC_AbortTrigger	停止位置锁存	该指令用停止 MC_TouchProbe 指令发起的位置锁存操作
同步运动指令	SMC_CamIn	电子凸轮	该指令实现电子凸轮功能
	MC_GearIn	电子齿轮	本指令实现电子齿轮功能
	SMC_MoveLink	追剪/飞剪	该指令实现追剪工艺动作
轴组运动指令	MC_GroupEnable	轴组使能	轴组使能
	MC_GroupDisable	轴组断使能	关闭轴组功能
	MC_GroupStop	轴组停止	控制轴组减速停止，中止轴组上任何正在执行的运动控制指令，并撤销该轴组合轴上所有运动缓存指令
	SMC_MoveLinearAbsolute2D	2 轴绝对位置直线插补	该指令实现 2 轴相对位置直线插补功能
	SMC_MoveLinearRelative2D	2 轴相对位置直线插补	该指令实现 2 轴绝对位置直线插补功能
	SMC_MoveCircularRelative2D	2 轴相对位置圆弧插补	该指令实现 2 轴相对位置圆弧插补功能
	MC_ GroupSetOverride	轴组运动倍率设置（超驰）	该指令设定轴组中合轴速度、加减速、加加速度比例因子，可按设定比例放大或缩小合轴运行的速度、加减速、加加速度
	SMC_GroupSetDynamicLimits	轴组运动参数限制值设置	该指令设定轴组运动参数的限制值
	SMC_GroupCornerSpeedLimits	轴组转角限速设置	该指令设定轴组运动过程中的转角限速参数，减小非光滑转角衔接处的运动冲击
	SMC_GroupCircularSpeedLimits	轴组圆弧限速设置	该指令设定轴组运动过程中的圆弧限速参数，减小圆弧转弯处的运动冲击

4.2 指令使用说明

使用指令前，请先了解指令内容中涉及的相关信息，便于更好地理解和使用指令。

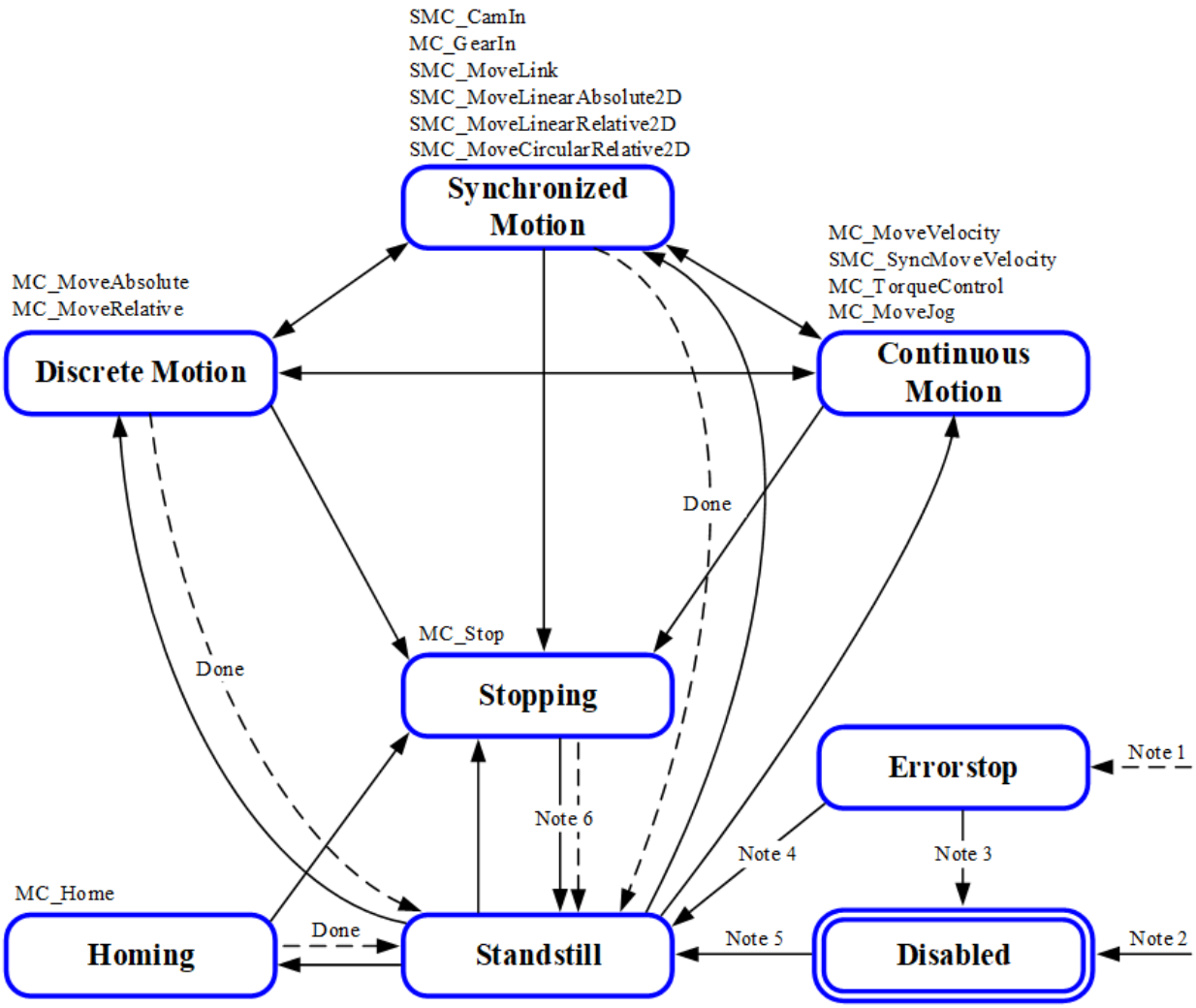
信息项	说明
指令/功能块	表示指令名称，例如：ADD
名称	表示指令的中文名称，例如：加法指令

FB/FUN	表示指令是功能块（FB）还是函数（FUN） 功能块型指令：可以被程序或 FB 调用。 函数型指令：可以被程序、FB、FUN 的任意一个调用。
图形示例	LD 梯形图程序语言编写的指令应用示例
ST 示例	ST 程序语言编写的指令应用示例
参数	对指令的输入、输出以及中间变量等参数信息进行说明。
功能	对指令功能进行说明
示例	指令的应用示例，通过在 LD 和 ST 中的编程示例进行说明。
使用注意事项	说明指令使用时的注意事项，也包含特殊场景下的应用以及异常情况的发生。
参见	使用该指令需要参考的其他内容信息

第5章　单轴指令

5.1　PLCopen 状态机

PLCopen 单轴轴状态机如下：



PLCopen 单轴轴状态机

状态机详细描述如下：

轴状态定义

轴状态	状态功能描述
Disabled	轴未使能状态（初始状态）
ErrorStop	故障停机状态
Standstill	使能状态
Homing	原点回归状态
Stopping	停止状态
Discrete Motion	离散运动
Continuous Motion	连续运动状态
Synchronized Motion	同步运动状态

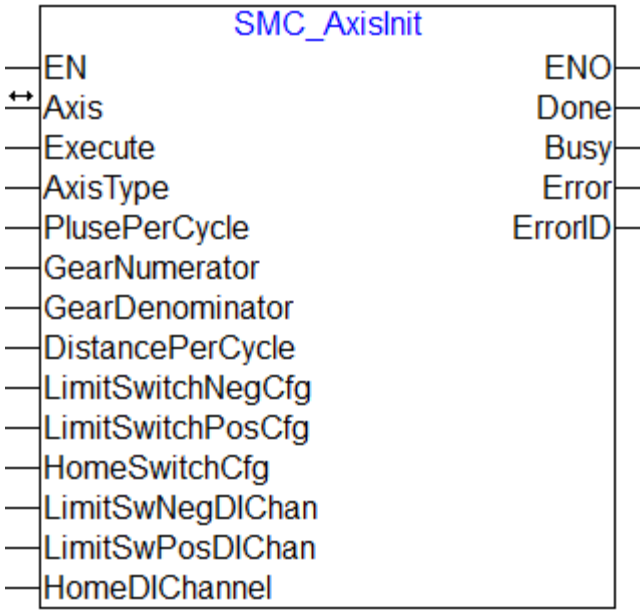
轴状态转换条件

转换	轴状态转换条件
Note 1	当轴的故障检测逻辑检测到故障时立即进入该状态
Note 2	当轴无故障且 MC_Power.Enable=FALSE 时
Note 3	当调用 MC_Reset 复位轴故障且 MC_Power.Status=FALSE 时
Note 4	当调用 MC_Reset 复位轴故障且 MC_Power.Enable=TRUE 且 MC_Power.Status=TRUE 时
Note 5	当 MC_Power.Enable=TRUE 且 MC_Power.Status=TRUE 时
Note 6	当 MC_Stop.Done=TRUE 且 MC_Stop.Execute=FALSE 时

- 可通过轴参 AxisState 查看轴 PLCopen 状态，也可通过指令 MC_ReadStatus 获取轴 PLCopen 状态。

5.2 SMC_AxisInit（轴初始化配置）

用于初始化轴配置，通过该功能块可以设置轴运行所需的基本配置，如轴类型、用户单位换算、限位开关通道等。



SMC_AxisInit 功能块

5.2.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	AxisType	轴类型	否	-	-1：Unused 0：Virtual 50：EtherCAT_Position 51：EtherCAT_Speed 52：EtherCAT_Torque 53：EtherCAT_Encoder	EMAXISTYPE
	PlusePerCycle	电机/编码器旋转一圈的脉冲数	否	-	正值	LREAL
	GearNumerator	减速比分子 （电机输入轴转数）	是	1	正值	UDINT
	GearDenominator	减速比分母 （减速机输出轴转数）	是	1	正值	UDINT
	DistancePerCycle	减速比为 1:1 时，表示电机/编码器旋转一圈工作台的移动量	否	-	正值	LREAL

		减速比不为 1:1 时，表示减速机输出轴旋转一圈工作台的移动量				
	LimitSwitchNegCfg	负限位开关属性配置	是	-1	-1：不使用 0：使用驱动器本体 DI（常开） 1：使用驱动器本体 DI（常闭） 10：使用通用 DI（常开） 20：使用通用 DI（常闭）	INT
	LimitSwitchPosCfg	正向限位开关属性配置	是	-1	-1：不使用 0：使用驱动器本体 DI（常开） 1：使用驱动器本体 DI（常闭） 10：使用通用 DI（常开） 20：使用通用 DI（常闭）	INT
	HomeSwitchCfg	原点开关属性配置	是	0	-1：不使用 0：使用驱动器本体 DI（常开） 10：使用通用 DI（常开）	INT
	LimitSwNegDIChan	负限位开关通道号 使用驱动器本体 DI 时，通道号为驱动器输入 PIN 脚在 Digital inputs 对象字典（0x60FD）的第几个 bit 位，且 PDO 中必须包含 0x60FD； 使用通用 DI 时，通道号为 DI 点在 I 区映射的位偏移。例如%IX100.1 在 I 区的 bit 位偏移为 100×8 + 1。	是	0	非负值	UDINT
	LimitSwPosDIChan	正限位开关通道号 计算方法同上	是	1	非负值	UDINT
	HomeDIChannel	原点开关通道号 计算方法同上	是	2	非负值	UDINT
OUTPUT	Done	完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL

	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.2.2 功能

设置轴运行所需的基本参数，当 Execute 上升沿时触发功能块进行轴相关参数的初始化配置操作，输出引脚 Done 为 TRUE 时表示调用该功能块配置轴相关参数成功，如果配置过程中出错，则 Error 为 TRUE，ErrorID 指示错误原因。参数分类介绍如下：

- 轴类型设置

通过参数 AxisType 设置指定轴的轴类型，可设置轴类型为控制器型号所支持的轴类型，LX 系列控制器支持的轴类型如下：

  - Unused（-1）：该轴未使用
  - Virtual（0）：虚轴
  - EtherCAT_Position（50）：EtherCAT 总线连接轴，位置控制
  - EtherCAT_Speed（51）：EtherCAT 总线连接轴，速度控制
  - EtherCAT_Torque（52）：EtherCAT 总线连接轴，转矩控制
  - EtherCAT_Encoder（53）：EtherCAT 总线连接轴，编码器计数

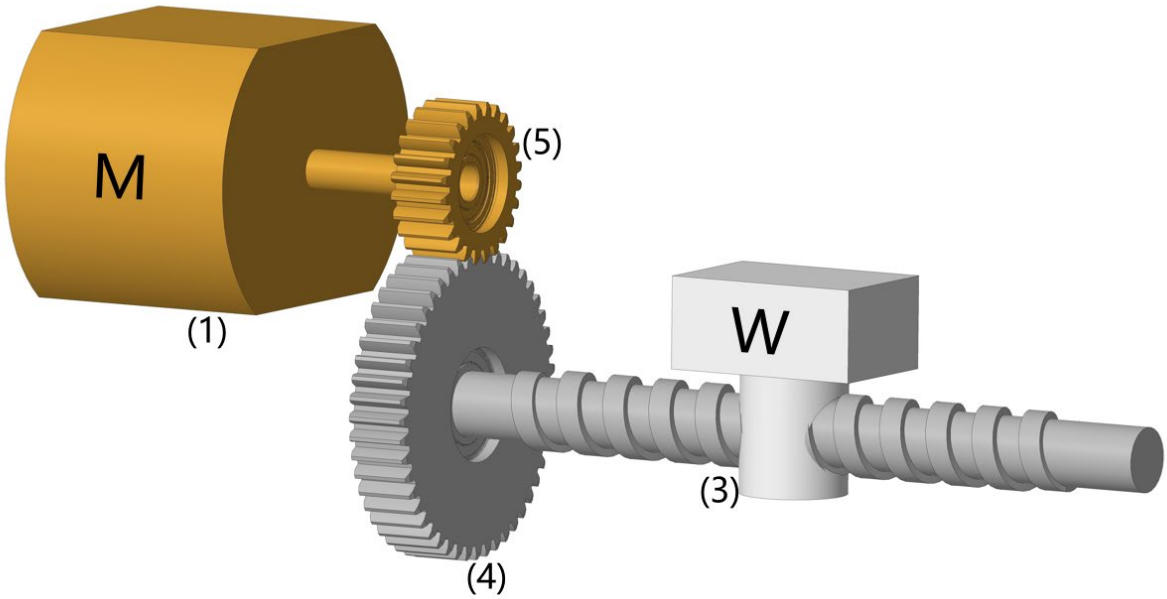
- 不同的轴类型需要注意：
  - EtherCAT_Encoder 类型的轴不接受运动控制指令，仅跟踪编码器反馈值。
  - 当该轴的轴指令缓存中有指令正在运行时，设置轴类型不生效。
  - 轴类型设置成功后，系统会自动根据不同的轴类型关联修改轴参数 RepMode。当用户设置轴类型为虚轴时，系统将设置 RepMode=2；当用户设置轴类型为其他轴类型时，系统将设置 RepMode=0。

- 单位转换因子设置
  - 通过入参 PlusePerCycle 设置电机/编码器旋转一圈的脉冲数(1)，单位：pulse/rev；

- 通过入参 GearNumerator 设置减速比分子，即下图中 (4)的齿数；
- 通过入参 GearDenominator 设置减速比分母，即下图中(5)的齿数；
- 不使用变速箱时，入参 DistancePerCycle 表示电机、编码器旋转一圈工作台的移动量(2)，单位为用户单位/rev；使用变速箱时，通过入参 DistancePerCycle 设置减速机输出轴旋转一圈工作台的移动量(3)，单位为用户单位/rev。

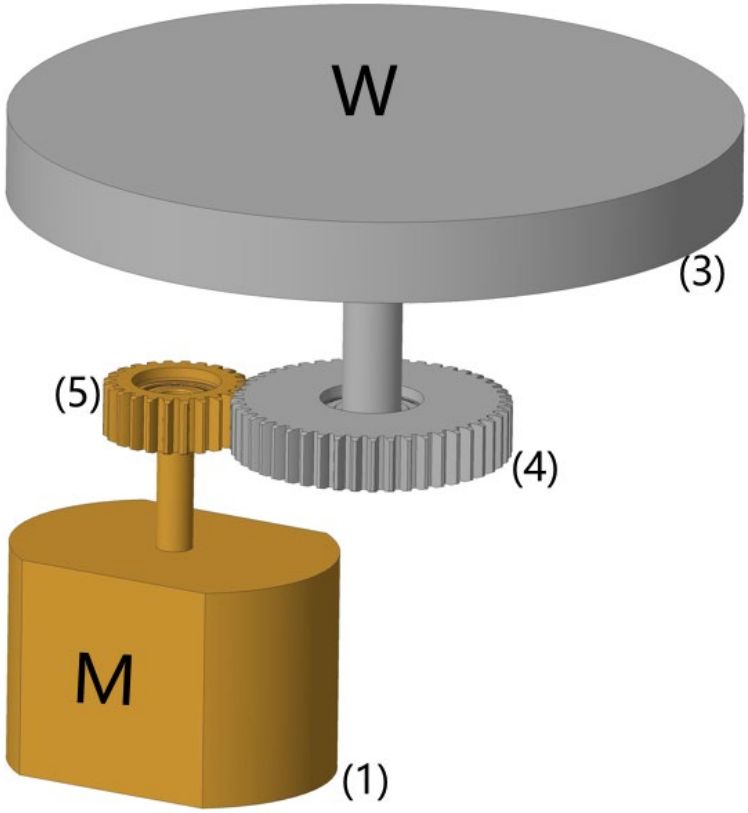
  其中用户单位可根据用户需求自行定义，可能是脉冲数、毫米、度等。

- 输出为直线位移时，示意图如下：



M:电机/编码器，W:工作台

- 输出为旋转位移时，示意图如下：



M:电机/编码器，W:工作台

该指令根据上述设置参数计算工作台移动量与电机指令脉冲数之间的转换关系，并设置轴单位转换因子 Units 参数。最终计算出的单位转换因子可通过轴参数 Units 查看，转换关系如下：

$$\text{Units} = \frac{\text{PlusePerCycle} * \text{GearNumerator}}{\text{DistancePerCycle} * \text{GearDenominator}}$$

电机/编码器与工作台之间不使用变速装置时，设置减速比分母 GearDenominator 等于 1，减速比分子 GearNumerator 等于 1，转换关系如下：

$$\text{脉冲数}[pulse] = \frac{(1)\text{PlusePerCycle}[\text{LREAL}]}{(2)\text{DistancePerCycle}[\text{LREAL}]} * \text{工作台移动量}[\textit{用户单位}]$$

电机/编码器与工作台之间使用变速装置时，根据实际情况修改 GearNumerator 和 GearDenominator，转换关系如下：

$$脉冲数[pulse] = \frac{(1)\mathrm{PlusePerCycle[LREAL]} * (4)\mathrm{GearNumerator[UDINT]}}{(3)\mathrm{DistancePerCycle[LREAL]} * (5)\mathrm{GearDenominator[UDINT]}} * 工作台移动量[用户单位]$$

- 正负限位开关设置

通过参数 LimitSwitchPosCfg/LimitSwitchNegCfg 设置正负限位开关属性，默认值为不使用，表示轴对应的硬限位功能无效。

参数 LimitSwNegDIChan/LimitSwPosDIChan 用来设置正负限位的通道号，通道号的含义与正负限位开关的属性设置相关。

设置属性为“使用驱动器本体 DI”时，表示轴对应的限位通道使用驱动器本体的 DI 通道，对应的通道号为对象字典 Digital inputs（0x60FD）的第几个 bit 位，驱动器 PDO 映射中必须包含对象字典 0x60FD，否则会报错，例如：使用驱动器本体 DI 时，设置通道号为 1，表示使用 0x60FD 的 bit 1 作为限位开关通道。

设置属性为“使用通用 DI”时，表示使用控制器通用 DI 通道作为限位通道，对应的通道号为 DI 点在 I 区映射的位偏移，例如：使用控制器通用 DI 时，所用 DI 的映射地址为%IX4.1，其在 I 区的位偏移为 4*8+1=33，即通道号设置为应为 33。

正负限位开关属性设置为常开表示限位通道输入逻辑为 1 时有效，设置为常闭表示限位通道输入逻辑为 0 时有效。

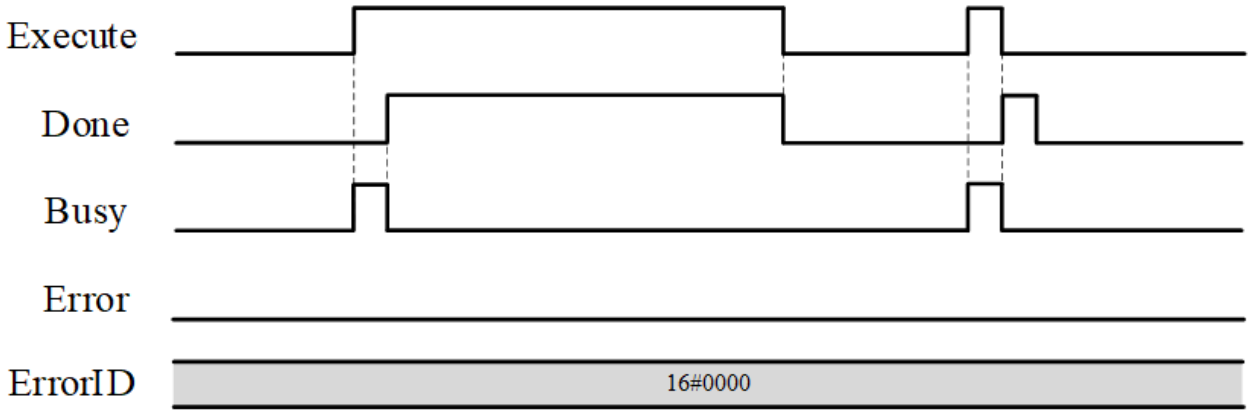
如果正负限位的属性值设置同为驱动器本体 DI 或同为通用 DI，则通道号输入值不能设置一样，否则功能块报错。另外，如果正负限位的属性值一个设置为驱动器本体 DI，另一个设置为通用 DI。如果设置为通用 DI 属性的通道号对应到了驱动器 Digtial Inputs 在 I 区的映射地址范围，且和设置为驱动器本体 DI 属性的通道号对应到了 Digtial Inputs 的同一个 bit 位，则二者也会发生冲突并报错。

- 原点开关设置

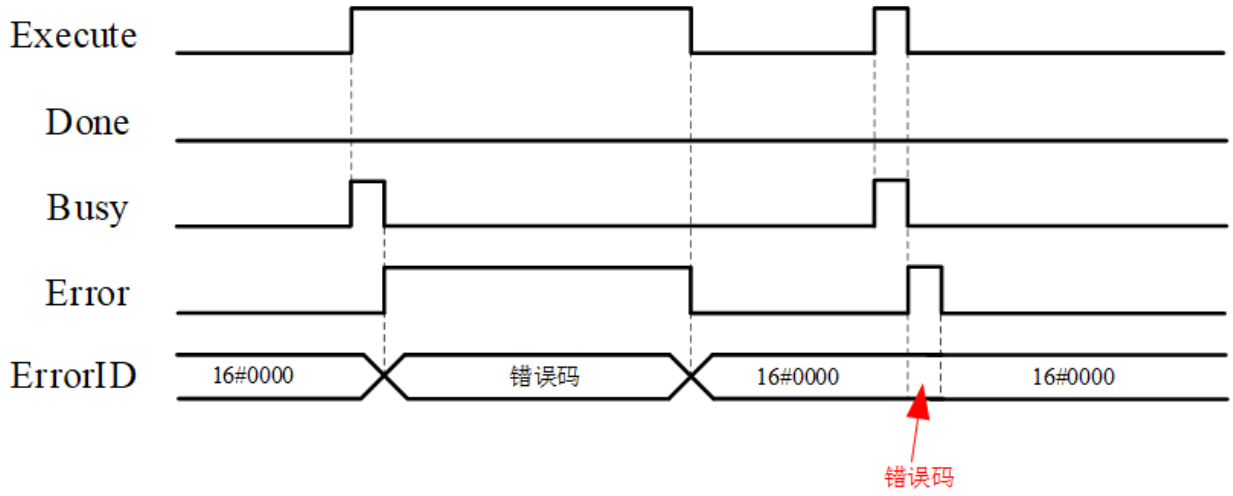
可使用 HomeSwitchCfg 和 HomeDIChannel 设置原点开关属性和原点开关通道值，参数含义同正负限位开关设置，只是原点开关通道属性不能设置为常闭，即通道输入逻辑不能反转。

- 功能块时序图

（1） 功能块正常执行且无错误，Execute 保持高电平的时序图。如果 Execute 只有一个周期的高电平，则 Done 信号也只保持一个周期。



（2） 功能块执行过程中报错，Execute 保持高电平的时序图。如果 Execute 只有一个周期的高电平，则 Error 信号及 ErrorID 也只保持一个周期。



5.2.3 示例

该示例使用 SMC_AxisInit 功能块初始化指定轴的轴类型、用户单位相关参数、限位开关和原点开关逻辑以及初始化限位开关和原点开关通道号。步骤如下：

（1） 设置轴类型为 EtherCAT_Position；

（2） 设置单位转换因子，使得电机转一圈（8388608 个脉冲）带动工作台移动 10000 个用户单位。

（3） 设置正负限位开关属性和原点开关属性配置为 10-使用通用 DI（常开）；

（4） 设置负限位开关通道号为 32，通道号对应 I 区地址为%IX4.0(4*8=32)，正限位开关通道号为

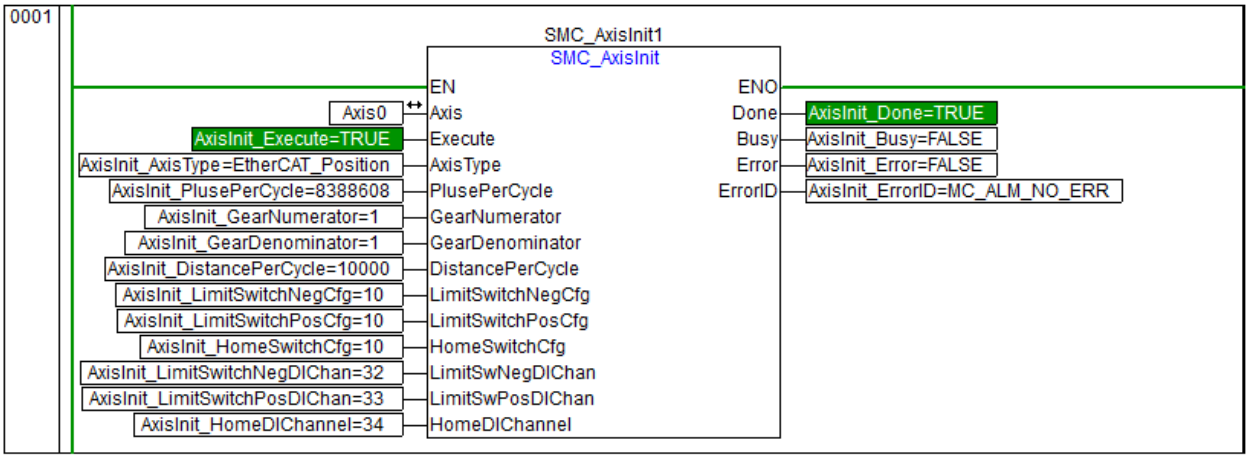
33(4)，通道号对应 I 区地址为%IX4.1，原点开关通道号为 34，通道号对应 I 区地址为%IX4.2。

使用 SMC_AxisInit 功能块初始化完成后，再进行轴使能和其它的指令控制。

■ 变量

序号	变量名	变量类型	初始值	变量说明
1	SMC_AxisInit1	SMC_AxisInit	-	指令实例定义
2	AxisInit_Execute	BOOL	FALSE	上升沿使能
3	AxisInit_AxisType	EMAXISTYPE	EtherCAT_Position	轴类型
4	AxisInit_PlusePerCycle	LREAL	8388608	伺服每周期的脉冲数
5	AxisInit_GearNumerator	UDINT	1	齿轮比分子
6	AxisInit_GearDenominator	UDINT	1	齿轮比分母
7	AxisInit_DistancePerCycle	LREAL	10000	每周期的距离
8	AxisInit_LimitSwitchNegCfg	INT	10	负向限位开关配置
9	AxisInit_LimitSwitchPosCfg	INT	10	正向限位开关配置
10	AxisInit_HomeSwitchCfg	INT	10	原点开关配置
11	AxisInit_LimitSwitchNegDIChan	UDINT	32	负向限位开关 DI 通道
12	AxisInit_LimitSwitchPosDIChan	UDINT	33	正向限位开关 DI 通道
13	AxisInit_HomeDIChannel	UDINT	34	原点 DI 通道
14	AxisInit_Done	BOOL	FALSE	初始化完成标志
15	AxisInit_Busy	BOOL	FALSE	初始化忙标志
16	AxisInit_Error	BOOL	FALSE	初始化错误标志
17	AxisInit_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	初始化错误 ID

■ LD 实现程序



- 初始化完成之后方可进行其它的指令操作，如 MC_Power 等。

■ ST 实现程序

```
SMC_AxisInit1(

Execute := AxisInit_Execute,

AxisType := AxisInit_AxisType,

PlusePerCycle := AxisInit_PlusePerCycle,

GearNumerator := AxisInit_GearNumerator,

GearDenominator := AxisInit_GearDenominator,

DistancePerCycle := AxisInit_DistancePerCycle,

LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,

LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,

HomeSwitchCfg := AxisInit_HomeSwitchCfg,

LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,

LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,

HomeDIChannel := AxisInit_HomeDIChannel,

Axis := Axis0,

Done=>AxisInit_Done,

Busy=>AxisInit_Busy,

Error=>AxisInit_Error,
```

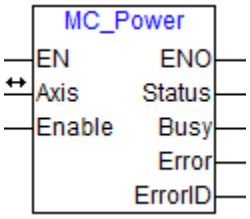
```
ErrorID=>AxisInit_ErrorID);
```

5.2.4 使用注意事项

- 该功能块在设置参数的时候内部配置顺序是先配置 PlusePerCycle、GearNumerator、GearDenominator、DistancePerCycle，再配置 LimitSwitchNegCfg、LimitSwNegDIChan，LimitSwitchPosCfg、LimitSwPosDIChan，HomeSwitchCfg、HomeDIChannel，最后配置 AxisType，如果前面的某一个参数设置出错，则后面的参数也不会进行配置，已配置成功的参数不受影响。
- 同一个轴可以重复使用本指令，不同任务重复使用本指令时，可能会存在交叉设置导致出错的情况，请避免这样使用。
- 在使用其它指令之前，需要先调用 SMC_AxisInit 指令进行轴初始化，并判断 SMC_AxisInit 指令 Done 引脚为 TRUE 后再进行其它操作，否则，可能会导致错误。

5.3 MC_Power（轴使能）

控制伺服使能状态指令。



5.3.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Enable	使能	否	FALSE	TRUE/FALSE	BOOL
OUTPUT	Status	轴使能标志	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL

	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.3.2 功能

本指令对轴进行使能设置，对该指令的详细介绍如下：

（1） 本指令高电平有效，将 Enable(使能)设为 TRUE 时，指定的轴可以进入使能状态，Status(使能中)变为 TRUE，可以实现轴的运动控制。

（2） 将 Enable(使能)设为 FALSE 时，可以对轴进行断使能，断使能后轴不会接受动作指令，无法实现轴的控制。并且，在断掉使能之后对轴执行动作指令轴也会表现异常。

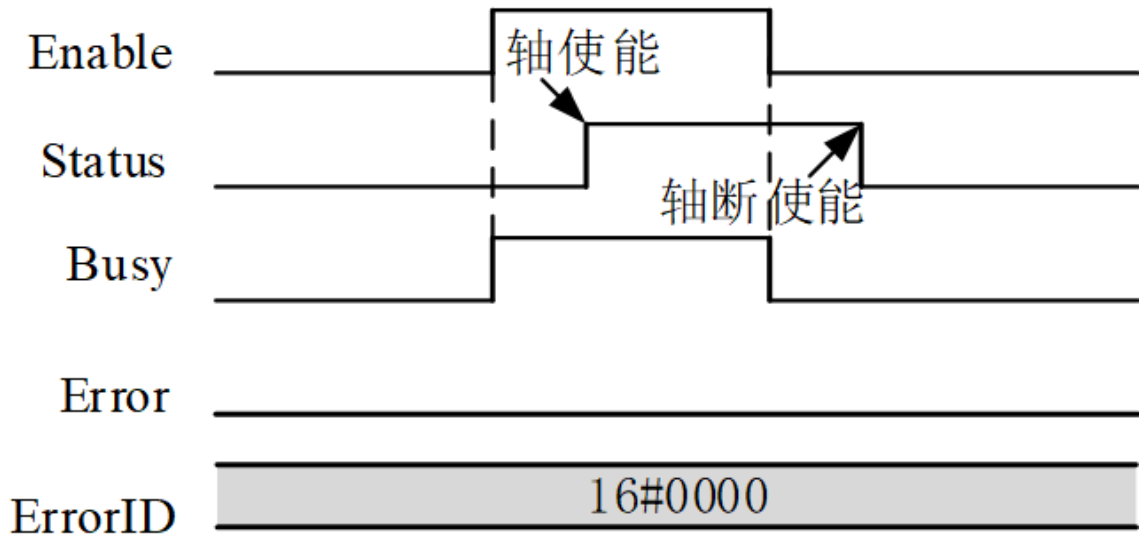
（3） 本指令输入为“Enable 型”指令时，不会重启运动指令，原则上一个轴只能设置一个 MC_Power（使能）模块。

（4） 将 Enable(使能)设为 FALSE，Busy(忙标志)变为 FALSE，Status(使能中)在断掉使能时变为 FALSE，Status(使能中)均表现为轴的状态。

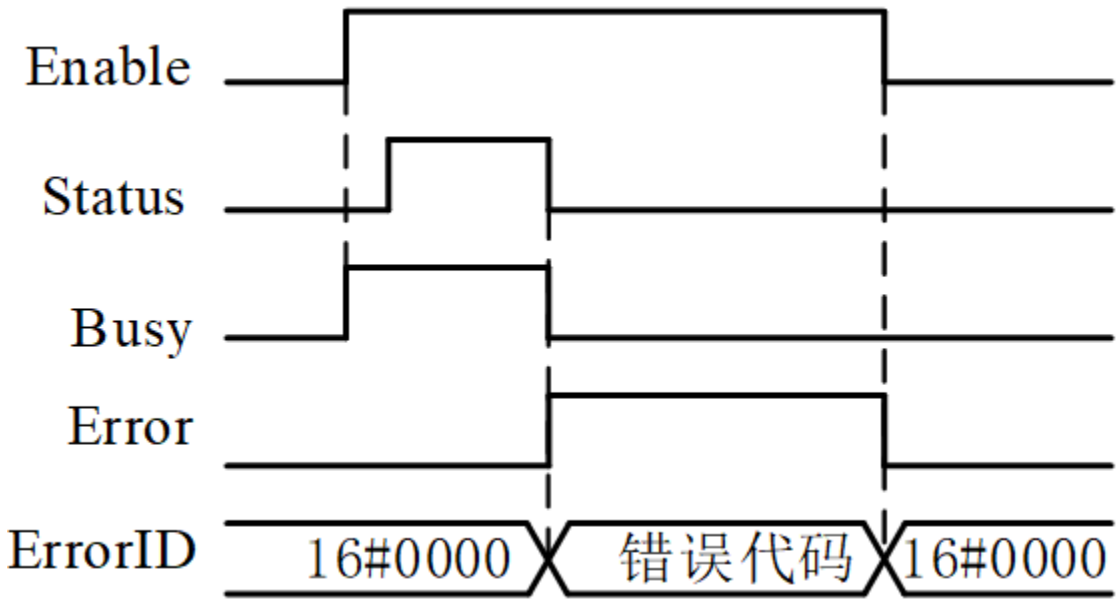
■ 功能块时序图

将 Enable(使能)设为 TRUE 时，指令的 Busy(执行中)变为 TRUE。

（1） 指令执行时的正确时序如下：



（2） 发生异常时候时序图如下：



5.3.3 示例

组建 MC_Power 示例程序进行轴使能设置。

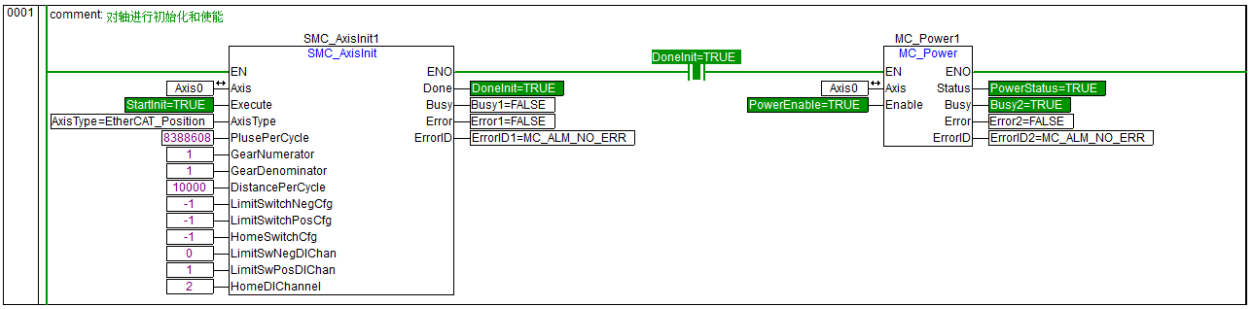
主要变量定义

名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
PowerEnable	BOOL	FALSE	开始使能时变为 TRUE
PowerStatus	BOOL	FALSE	使能成功时变为 TRUE
StartReset	BOOL	FALSE	开始复位时变为 TRUE

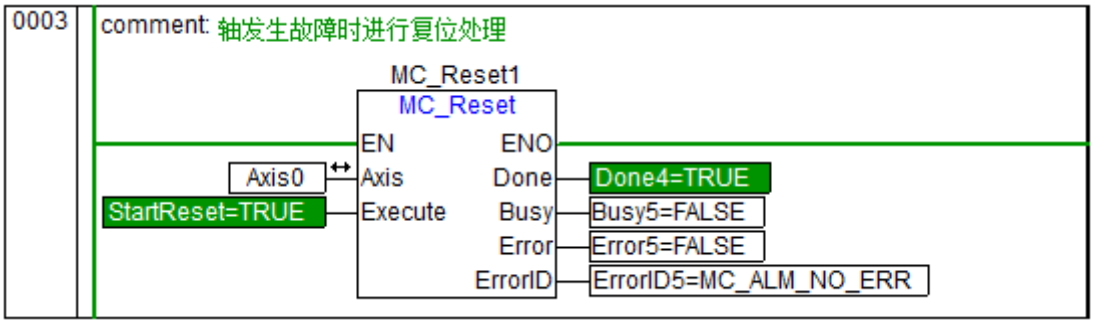
■ 梯形图程序

利用梯形图组建程序，示例程序如下：

（1） 首先对轴进行初始化，对轴的类型和各种参数进行初始化设置，待轴初始化完成之后，对轴进行使能，进入伺服使能的 ON 状态时，轴可以实现运动控制。



（2） 当轴出现错误时，可以调用 MC_Reset 功能块进行复位处理。



■ ST 语言程序

（1） 对轴进行初始化设置。

```
(*轴进行初始化*)

SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
```

```
ErrorID=>ErrorID1);
```

（2） 对轴进行初始化设置。

```
(*对轴进行使能*)

IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

（3） 出错时可以进行复位处理。

```
MC_Reset1(
Execute := StartReset,
Axis := Axis0,
Done=>Done4,
Busy=>Busy5,
Error=>Error5,
ErrorID=>ErrorID5);
```

5.3.4 注意事项

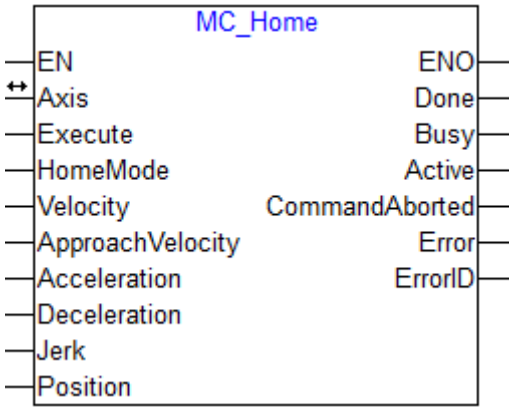
- 因为无论该功能块的 Enable(使能)为 TRUE 还是 FALSE 时功能块都会起作用，因此在调用 Mc_Power 时，应该在调用之前添加一个触发模块。
- 在垂直轴等一些工作环境中，断掉使能可能会导致轴位置发生急剧变化，因此调用此功能块时需要谨慎注意。

5.3.5 参见

- 若有错误发生，参见功能块错误码。

5.4 MC_Home（原点回归）

单轴原点回归指令。



5.4.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	HomeMode	原点回归模式	否	-	支持 CIA402 规约中如下回零模式： 1/2，3/5，7/11，17/18，19/21，23/27，33/34，35	SMC_HOME_MODE
	Velocity	高速查找速度	否	-	正值	LREAL
	ApproachVelocity	低速寻零速度	否	-	正值	LREAL
	Acceleration	加速度	否	-	正值	LREAL
	Deceleration	减速度	否	-	正值	LREAL
	Jerk	0：梯形加减速 正值：S 形加减速加加速度	是	0	0/正值	LREAL
	Position	原点回归完成后，零点信号位置被定义为 Position	是	0	正值/负值/0	LREAL
OUTPUT	Done	原点回归完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为 TRUE	是	FALSE	TRUE/FALSE	BOOL

	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.4.2 功能

该指令用于单轴的原点回归功能，通过检测原点开关信号或伺服电机 Z 相信号确定指定轴原点，原点回归模式由参数 HomeMode 指定，原点回归模式遵循 CIA402 协议中的模式 1/2、3/5、7/11、17/18、19/21、23/27、33/34、35 共 15 种回归模式。

在使用该指令之前需要先调用 SMC_AxisInit 指令设置必要的正负限位开关和原点开关参数，原点开关信号可设置为伺服驱动器本体 DI 或控制器通用 DI，具体配置方式参见 SMC_AxisInit 章节。

该指令只有使用 MC_Power 指令将指定轴切换到使能状态并且设置了支持的轴类型后才能正常使用，该指令支持虚拟轴的原点回归功能，但设置为虚拟轴时不支持 Z 信号相关的原点回归模式。

该指令上升沿有效，在指令 Execute 上升沿时，锁定轴名、原点回归模式、速度、加减速等入参，在 Execute=TRUE 或 Execute=FALSE 保持期间修改入参无效。MC_Home 指令在运行时，轴状态处于 Homing 状态。指令入参 HomeMode 用于设置原点回归方式，Velocity 用于设置高速查找零点时的速度，ApproachVelocity 用于设置低速寻找零点的速度，ApproachVelocity 设置值要小于 Velocity 设置值，Acceleration、Deceleration 用于设置回零过程中的加速度和减速度，Jerk 用于配置是否启用 S 形加减速功能，Position 用于在原点回归完成后，定义用户原点位置。该功能块成功执行后，Busy=TRUE，Home 指令投送成功并激活后 Active=TRUE，原点回归正常执行完成后 Done=TRUE。该指令有效时轴状态处于 Homing 状态，能让轴处于 Stopping 状态的 MC_Stop 指令可以打断 MC_Home 指令，指令被打断后 CommandAboarted=TRUE。原点回归找寻原点完成后，可通过 MC_ReadAxisInfo 指令的 IsHomed 引脚查看原点回归完成状态。

重复使能同一个 MC_Home 指令功能块时，新 Home 指令将被缓存，指令功能块将失去对上一条指令的监控，转而监控新投送的指令执行情况。

该指令在执行时，控制器软限位功能无效，与原点回归模式相关的硬限位功能无效，指定的原点回归模式没有用到的硬限位功能有效。

- 原点回归模式说明

原点回归模式简要说明

模式	回零方向	回零使用到的信号	原点最终位置	原点回归过程简要说明
1	负向	N-OT Z	Z	负向回零，减速点为负向限位开关，原点为电机 Z 信号，遇到 Z 信号前必须先遇到负向限位下降沿
2	正向	P-OT Z	Z	正向回零，减速点为正向限位开关，原点为电机 Z 信号，遇到 Z 信号前必须先遇到正向限位下降沿
3	正向	HMSW Z	Z	正向回零，减速点为原点开关，原点为电机 Z 信号，遇到 Z 信号前必须先遇到原点开关同一侧下降沿
5	负向	HMSW Z	Z	负向回零，减速点为原点开关，原点为电机 Z 信号，遇到 Z 信号前必须先遇到原点开关同一侧下降沿
7	正向	HMSW P-OT Z	Z	正向回零，减速点为原点开关，原点为电机 Z 信号，遇到 Z 信号前必须先遇到原点开关同一侧下降沿（错过原点开关，遇到正向限位，高速负向）
11	负向	HMSW N-OT Z	Z	负向回零，减速点为原点开关，原点为电机 Z 信号，遇到 Z 信号前必须先遇到原点开关同一侧下降沿（错过原点开关，遇到负向限位，高速负向）
17	负向	N-OT	N-OT↓	负向回零，减速点为负向限位开关，原点为负向限位下降沿
18	正向	P-OT	P-OT↓	正向回零，减速点为正向限位开关，原点为正向限位下降沿
19	正向	HMSW	HMSW↓	正向回零，减速点为原点开关，原点为原点开关下降沿
21	负向	HMSW	HMSW↓	负向回零，减速点为原点开关，原点为原点开关下降沿
23	正向	HMSW P-OT	HMSW↓	正向回零，减速点为原点开关，原点为原点开关下降沿（错过原点开关，遇到正向限位，高速负向）
27	负向	HMSW N-OT	HMSW↓	负向回零，减速点为原点开关，原点为原点开关下降沿（错过原点开关，遇到负向限位，高速负向）
33	负向	Z	Z	负向回零，原点为电机 Z 信号
34	正向	Z	Z	正向回零，原点为电机 Z 信号
35	无	无	当前位置	以当前位置为原点

- 注：Z 表示伺服电机 Z 脉冲信号，N-OT 表示反向限位开关信号，P-OT 表示正限位开关信号，HMSW 表示原点开关信号，↓表示开关信号下降沿。

（1） 模式 1，寻找负限位开关和 Z 信号

原点：电机 Z 信号

减速点：负限位开关

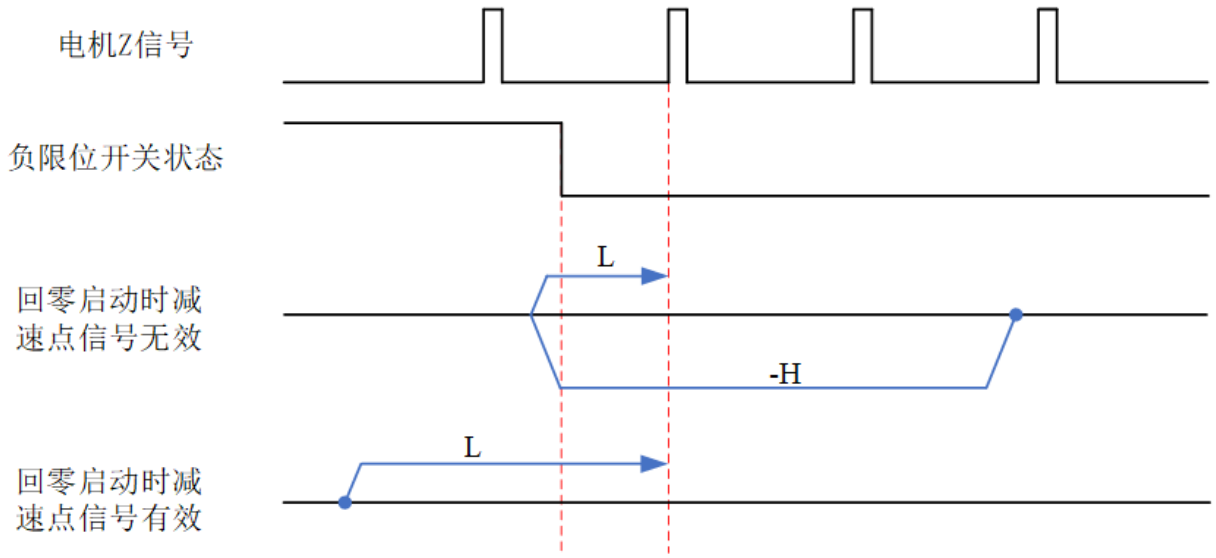
运动轨迹 1：回零启动时如果负限位开关状态无效，则以高速朝负向运动，遇到负限位开关的上升

沿之后减速停止，然后换低速朝正向运动。在低速朝正向运动时遇到负限位开关的下降沿之后，继续以低速朝正向找最近的 Z 信号位置作为原点

运动轨迹 2：回零启动时如果负限位开关状态有效，则以低速朝正向运动。在朝正向遇到负限位开关的下降沿之后，继续以低速朝正向找最近的 Z 信号位置作为原点。

这种模式下，遇到正限位开关的有效状态，会中止回原点流程，即该指令被打断。

模式 1 回归过程如下图所示：



- 注：图中“H”代表高速查找速度 Velocity，“L”代表低速寻零速度 ApproachVelocity，“-”代表反向运动，下同。

(2) 模式 2：寻找正限位开关和 Z 信号。

原点：电机 Z 信号

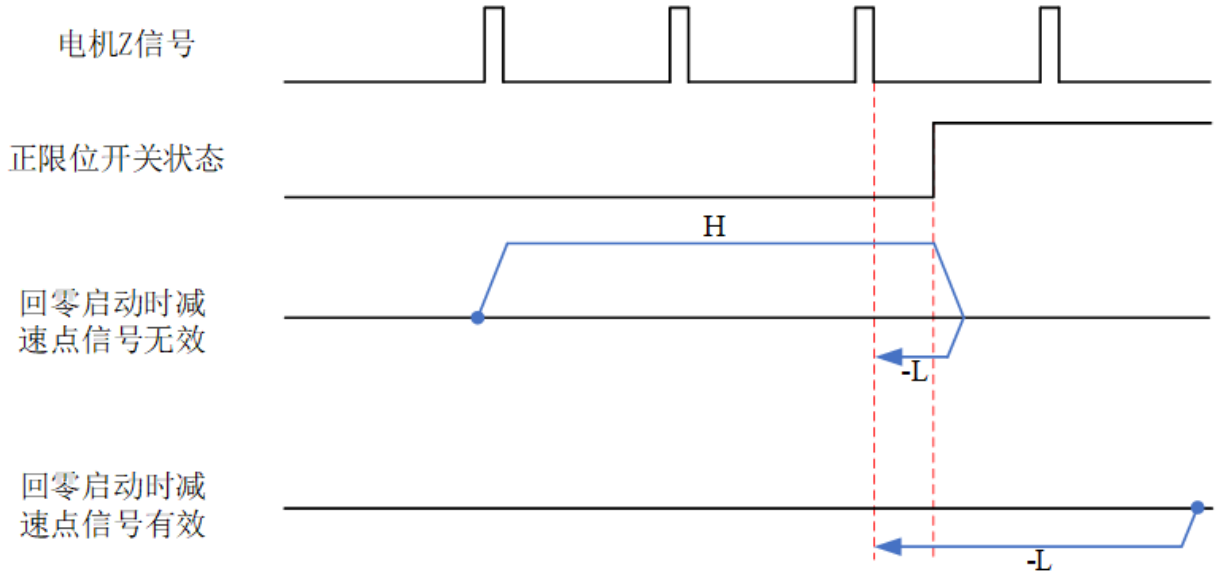
减速点：正限位开关

运动轨迹 1:回零启动时如果正限位开关状态无效，则以高速朝正向运动，遇到正限位开关的上升沿之后减速停止，然后换低速朝负向运动。在低速朝负向运动时遇到正限位开关的下降沿之后，继续以低速朝负向找最近的 Z 信号位置作为原点。

运动轨迹 2:回零启动时如果正限位开关状态有效，则以低速朝负向运动。在朝负向遇到正限位开关的下降沿之后，继续以低速朝负向找最近的 Z 信号位置作为原点。

这种模式下，遇到负限位开关的有效状态，会中止回原点流程，即该指令被打断。

模式 2 回归过程如下图所示：



(3) 模式 3：寻找朝负向运动时的原点开关下降沿位置和 Z 信号。

原点：电机 Z 信号

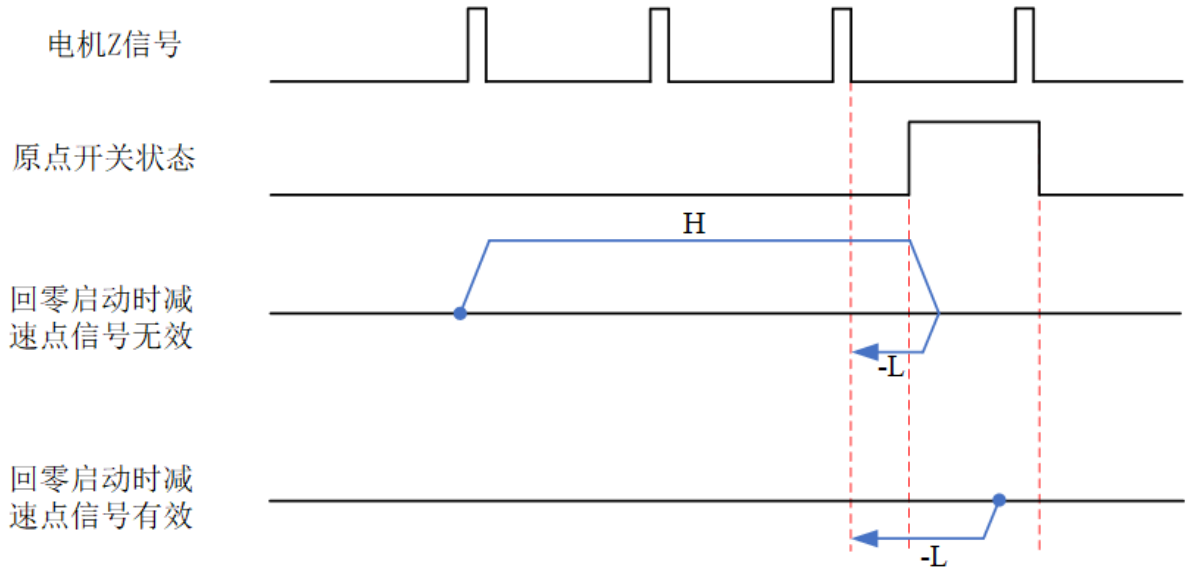
减速点：原点开关

运动轨迹 1：回零启动时原点开关无效，则以高速朝正向运动。在正向运动时遇到原点开关的上升沿之后减速停止，然后换低速朝负向运动。在低速负向运动时遇到原点开关的下降沿之后，继续以低速朝负向找最近的 Z 信号位置作为原点。

运动轨迹 2：回零启动时原点开关有效，则以低速朝负向运动。在负向运动时遇到原点开关的下降沿之后，继续以低速朝负向运行，找最近的 Z 信号位置作为原点。

这种模式下，无论遇到正限位开关还是负限位开关的有效状态，都会中止回原点流程，即该指令被打断。

模式 3 回归过程如下图所示：



（4） 模式 5，寻找朝正向运动时的原点开关下降沿位置和 Z 信号。

原点：电机 Z 信号

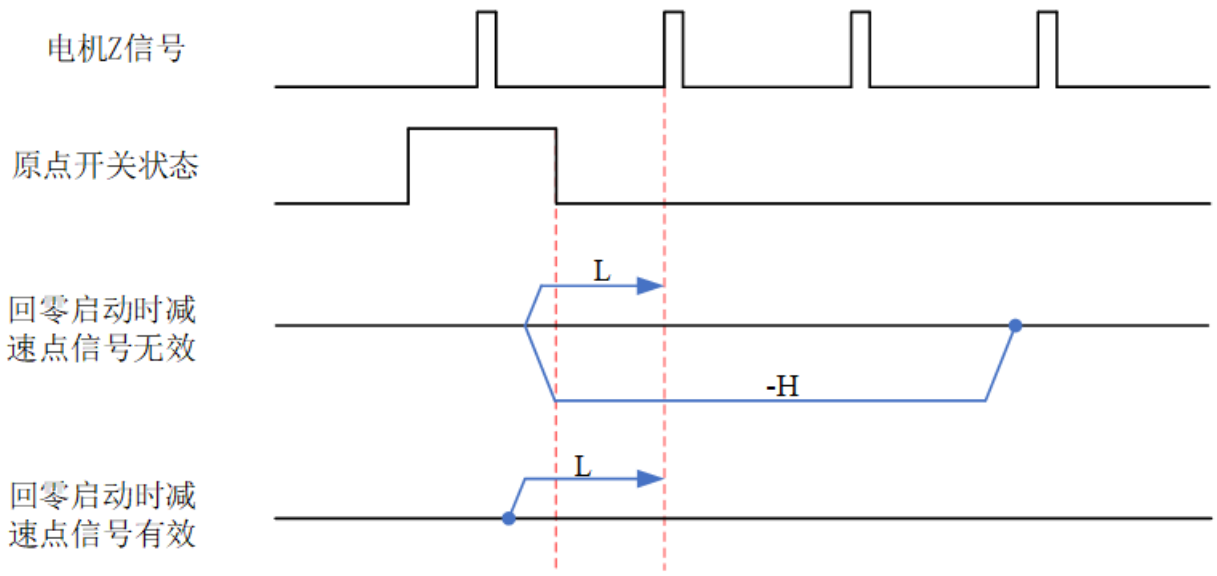
减速点：原点开关

运动轨迹 1：回零启动时原点开关无效，则以高速朝负向运动。在负向运动时遇到原点开关的上升沿之后减速停止，然后换低速朝正向运动。在低速正向运动时遇到原点开关的下降沿之后，继续以低速朝正向找最近的 Z 信号位置作为原点。

运动轨迹 2：回零启动时原点开关有效，则以低速朝正向运动。在正向运动时遇到原点开关的下降沿之后，继续以低速朝正向运行，找最近的 Z 信号位置作为原点。

这种模式下，无论遇到正限位开关还是负限位开关的有效状态，都会中止回原点流程，即该指令被打断。

模式 5 回归过程如下图所示：



（5） 模式 7，寻找朝负向运动时的原点开关下降沿位置和 Z 信号，遇正限位自动反向。

原点：电机 Z 信号

减速点：原点开关

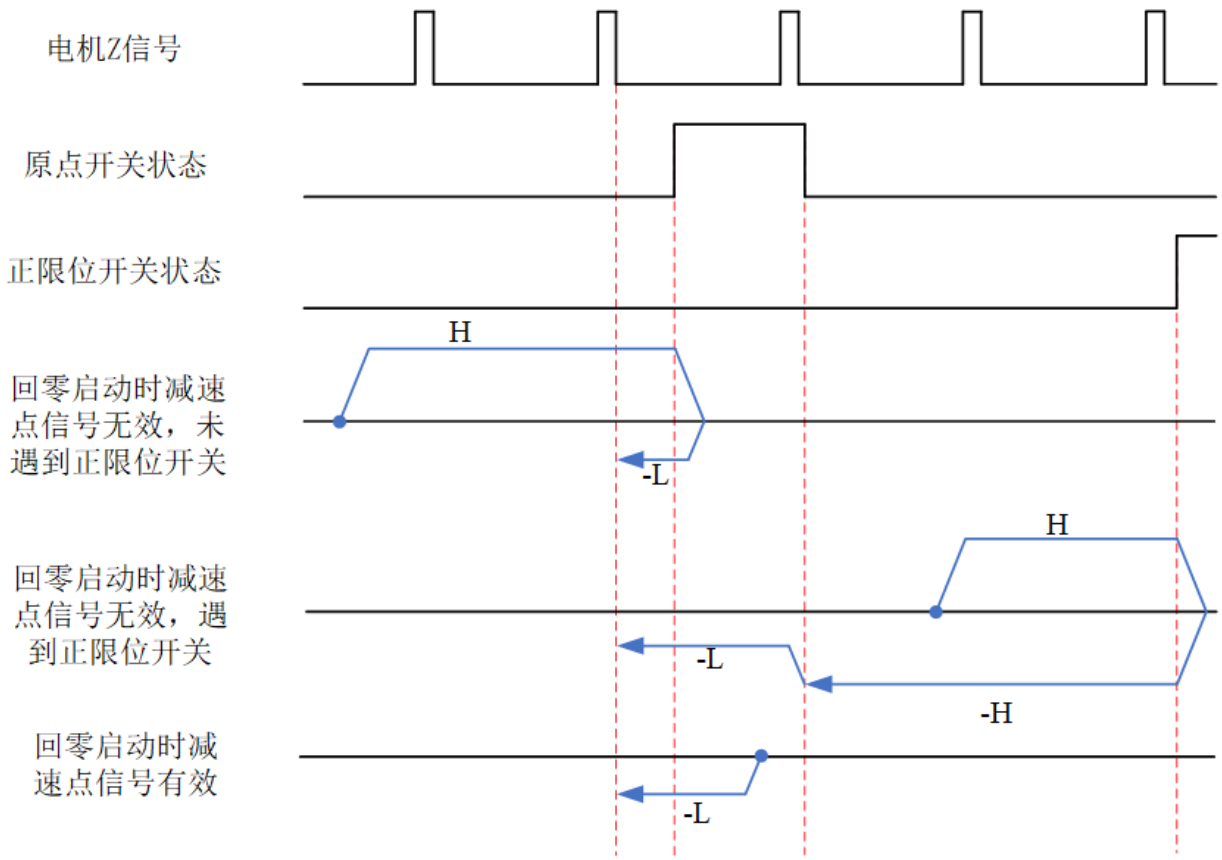
运动轨迹 1：回零启动时原点开关状态无效，则以高速朝正向运动，在正向运动时遇到原点开关的上升沿之后减速停止，然后换低速朝负向运动。低速负向运动时遇到原点开关的下降沿之后，继续以低速朝负向找最近的 Z 信号位置作为原点。

运动轨迹 2：回零启动时原点开关状态无效，则以高速朝正向运动，在正向运动时遇到正向限位开关的有效状态时减速停止，然后以高速朝负向运动。在负向运动时遇到原点开关的上升沿后开始减速至低速，然后以低速继续朝负向运动。在低速负向运动时遇到原点开关的下降沿之后，继续以低速朝负向找最近的 Z 信号位置作为原点。

运动轨迹 3：回零启动时原点开关状态有效，则以低速朝负向运动。在负向运动时遇到原点开关的下降沿之后，继续以低速朝负向找最近的 Z 信号位置作为原点。

这种模式下，朝正向运动首先遇到正向限位开关的有效状态时自动反向；遇到负向限位开关的有效状态时，则中止回原点流程，即该指令被打断。

模式 7 回归过程如下图所示：



(6) 模式 11，寻找朝正向运动时的原点开关下降沿位置和 Z 信号，遇负限位自动反向。

原点：电机 Z 信号

减速点：原点开关

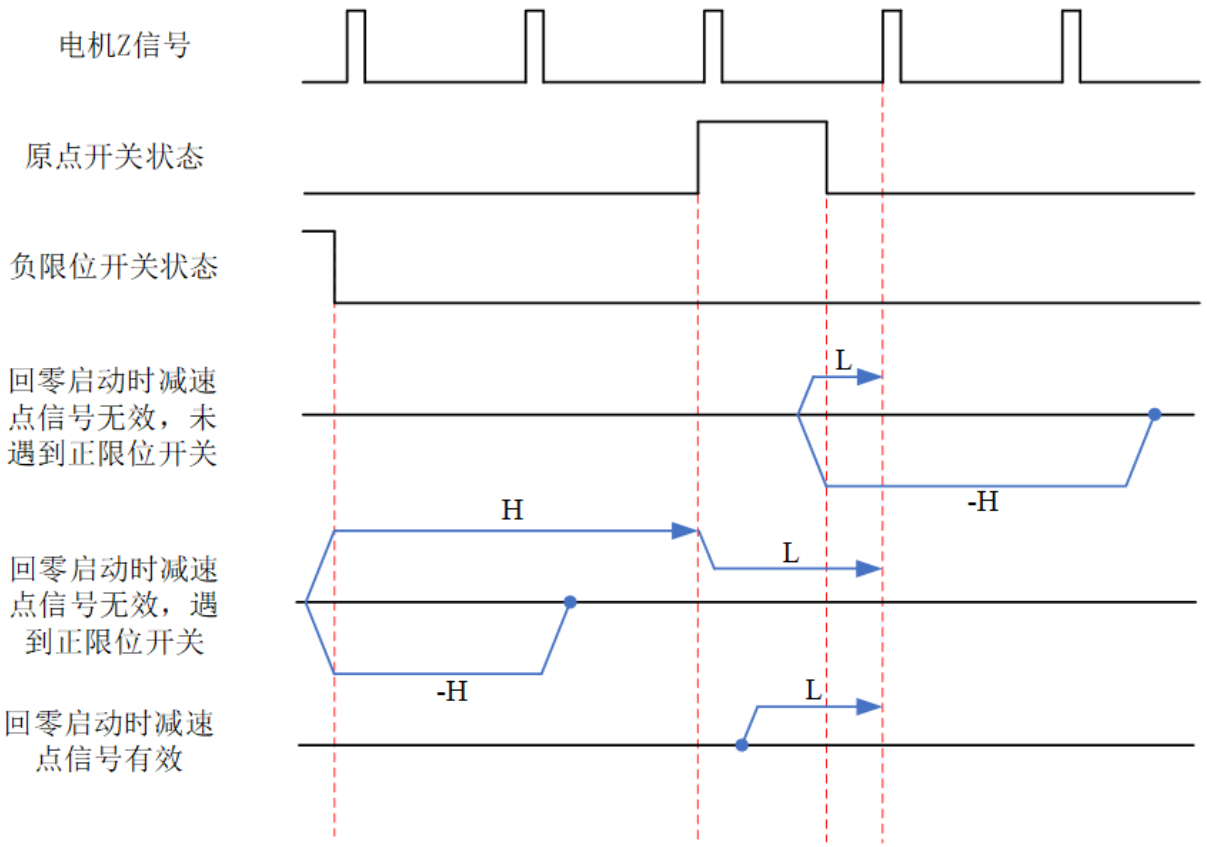
运动轨迹 1：回零启动时原点开关状态无效，则以高速朝负向运动，在负向运动时遇到原点开关的上升沿之后减速停止，然后换低速朝正向运动。低速正向运动时遇到原点开关的下降沿之后，继续以低速朝正向找最近的 Z 信号位置作为原点。

运动轨迹 2：回零启动时原点开关状态无效，则以高速朝负向运动，在负向运动时遇到负向限位开关的有效状态时减速停止，然后以高速朝正向运动。在正向运动时遇到原点开关的上升沿后开始减速至低速，然后以低速继续朝正向运动。在低速正向运动时遇到原点开关的下降沿之后，继续以低速朝正向找最近的 Z 信号位置作为原点。

运动轨迹 3：回零启动时原点开关状态有效，则以低速朝正向运动。在正向运动时遇到原点开关的下降沿之后，继续以低速朝正向找最近的 Z 信号位置作为原点。

这种模式下，朝负向运动首先遇到负向限位开关的有效状态时自动反向;遇到正向限位开关的有效状态时，则中止回原点流程，即该指令被打断。

模式 11 回归过程如下图所示：



（7） 模式 17，寻找负限位开关。

原点：负限位开关

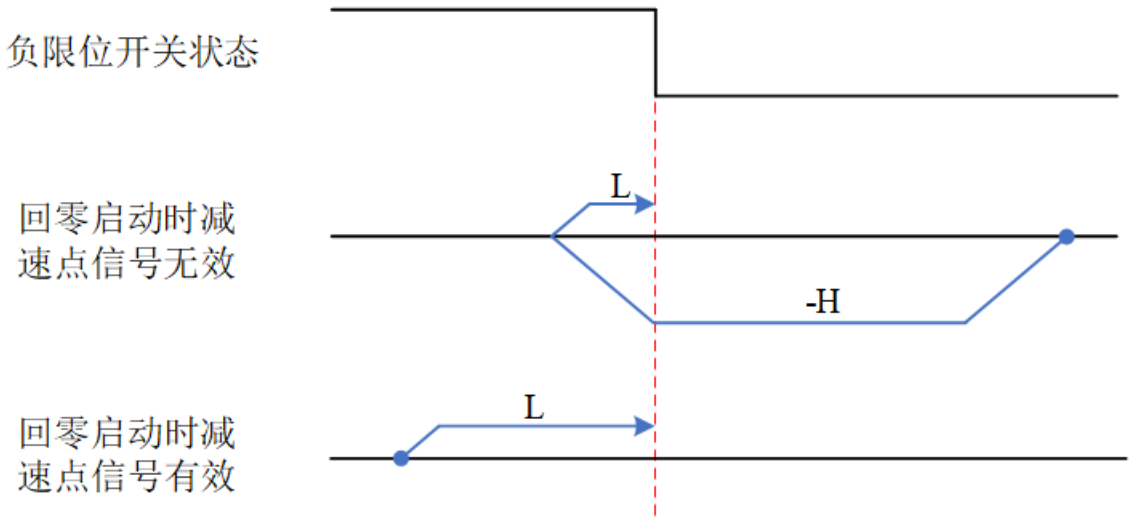
减速点：负限位开关

运动轨迹 1：回零启动时如果负限位开关状态无效，则以高速朝负向运动，遇到负限位开关的上升沿之后减速停止，然后换低速朝正向运动。在低速朝正向运动过程中，找负限位开关的下降沿所在的位置作为原点。

运动轨迹 2：回零启动时如果负限位开关状态有效，则以低速朝正向运动。在低速朝正向运动过程中，找负限位开关的下降沿所在的位置作为原点。

这种模式下，遇到正限位开关的有效状态，会中止回原点流程，即该指令被打断。

模式 17 回归过程如下图所示：



(8) 模式 18，寻找正限位开关。

原点：正限位开关

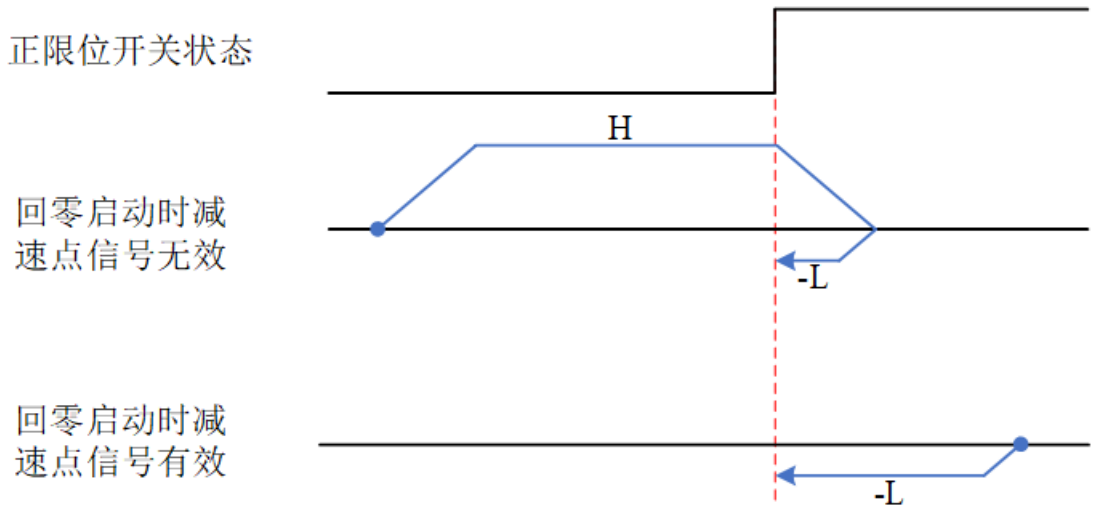
减速点：正限位开关

运动轨迹 1：回零启动时如果正限位开关状态无效，则以高速朝正向运动，遇到正限位开关的上升沿之后减速停止，然后换低速朝负向运动。在低速朝负向运动过程中，找正限位开关的下降沿所在的位置作为原点。

运动轨迹 2：回零启动时如果正限位开关状态有效，则以低速朝负向运动。在低速朝负向运动过程中，找正限位开关的下降沿所在的位置作为原点。

这种模式下，遇到负限位开关的有效状态，会中止回原点流程，即该指令被打断。

模式 18 回归过程如下图所示：



(9) 模式 19，寻找朝负向运动时原点开关的下降沿位置。

原点：原点开关

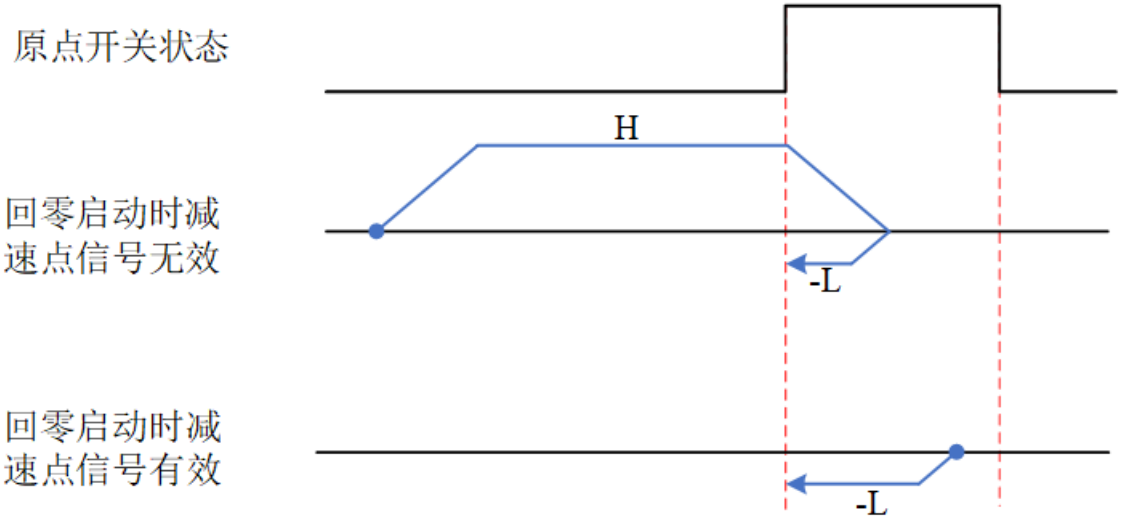
减速点：原点开关

运动轨迹 1：回零启动时原点开关无效，则以高速朝正向运动。在正向运动时遇到原点开关的上升沿之后减速停止，然后换低速朝负向运动。在低速朝负向运动过程中，寻找原点开关的下降沿所在的位置作为原点。

运动轨迹 2：回零启动时原点开关有效，则以低速朝负向运动。在低速朝负向运动过程中，寻找原点开关的下降沿所在的位置作为原点。

这种模式下，无论遇到正限位开关还是负限位开关的有效状态，都会中止回原点流程，即该指令被打断。

模式 19 回归过程如下图所示：



（10） 模式 21，寻找朝正向运动时原点开关的下降沿位置。

原点：原点开关

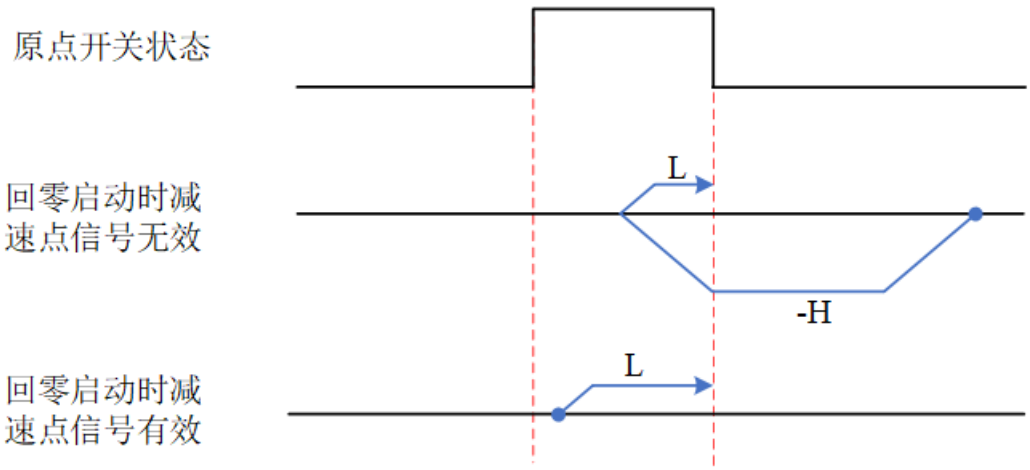
减速点：原点开关

运动轨迹 1：回零启动时原点开关无效，则以高速朝负向运动。在负向运动时遇到原点开关的上升沿之后减速停止，然后换低速朝正向运动。在低速朝正向运动过程中，寻找原点开关的下降沿所在的位置作为原点。

运动轨迹 2：回零启动时原点开关有效，则以低速朝正向运动。在低速朝正向运动过程中，找原点开关的下降沿所在的位置作为原点。

这种模式下，无论遇到正限位开关还是负限位开关的有效状态，都会中止回原点流程，即该指令被打断。

模式 21 回归过程如下图所示：



（11） 模式 23，寻找朝负向运动时的原点开关下降沿位置，遇正限位自动反向。

原点：原点开关

减速点：原点开关

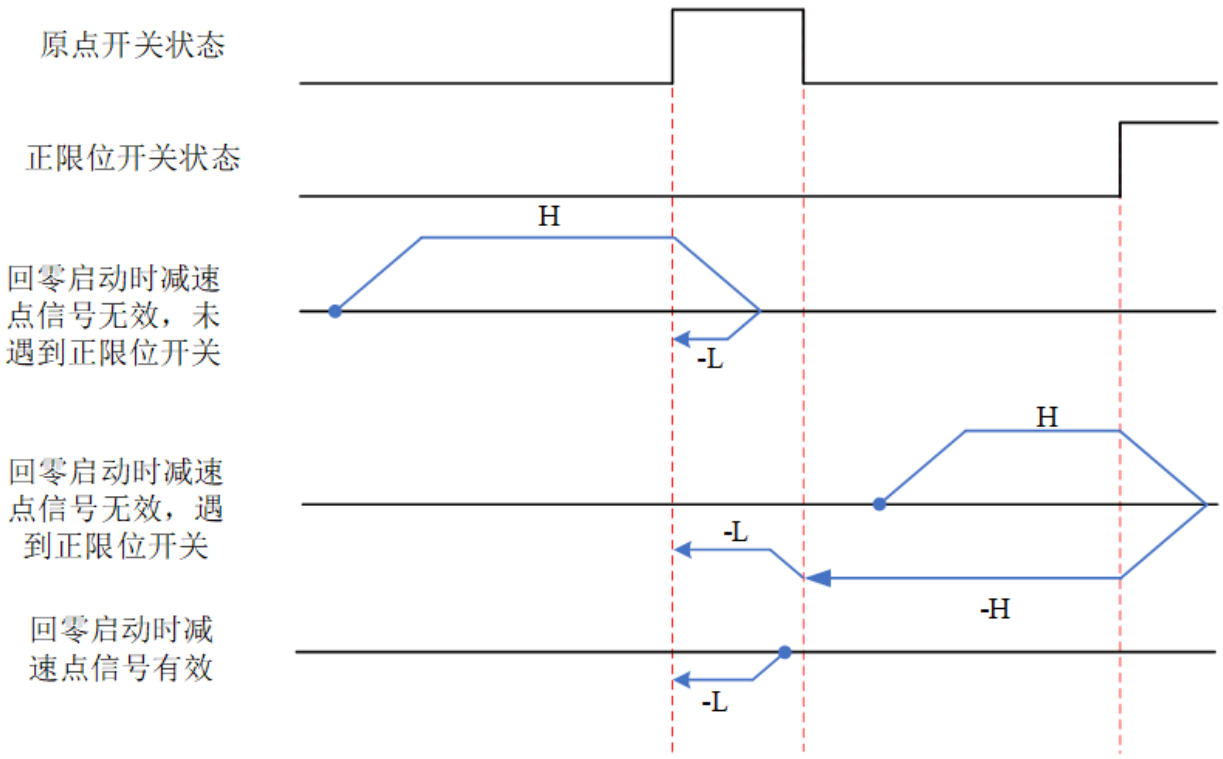
运动轨迹 1：回零启动时原点开关状态无效，则以高速朝正向运动，在正向运动时遇到原点开关的上升沿之后减速停止，然后换低速朝负向运动。在低速朝负向运动过程中，寻找原点开关的下降沿所在的位置作为原点。

运动轨迹 2：回零启动时原点开关状态无效，则以高速朝正向运动，在正向运动时遇到正向限位开关的有效状态时减速停止，然后以高速朝负向运动。在负向运动时遇到原点开关的上升沿后开始减速至低速，然后以低速继续朝负向运动。在低速朝负向运动过程中，寻找原点开关的下降沿所在的位置作为原点。

运动轨迹 3：回零启动时原点开关状态有效，则以低速朝负向运动。在低速朝负向运动过程中，寻找原点开关的下降沿所在的位置作为原点。

这种模式下，朝正向运动首先遇到正向限位开关的有效状态时自动反向；遇到负向限位开关的有效状态时，则中止回原点流程，即该指令被打断。

模式 23 回归过程如下图所示：



（12） 模式 27，寻找朝正向运动时的原点开关下降沿位置，遇负限位自动反向。

原点：原点开关

减速点：原点开关

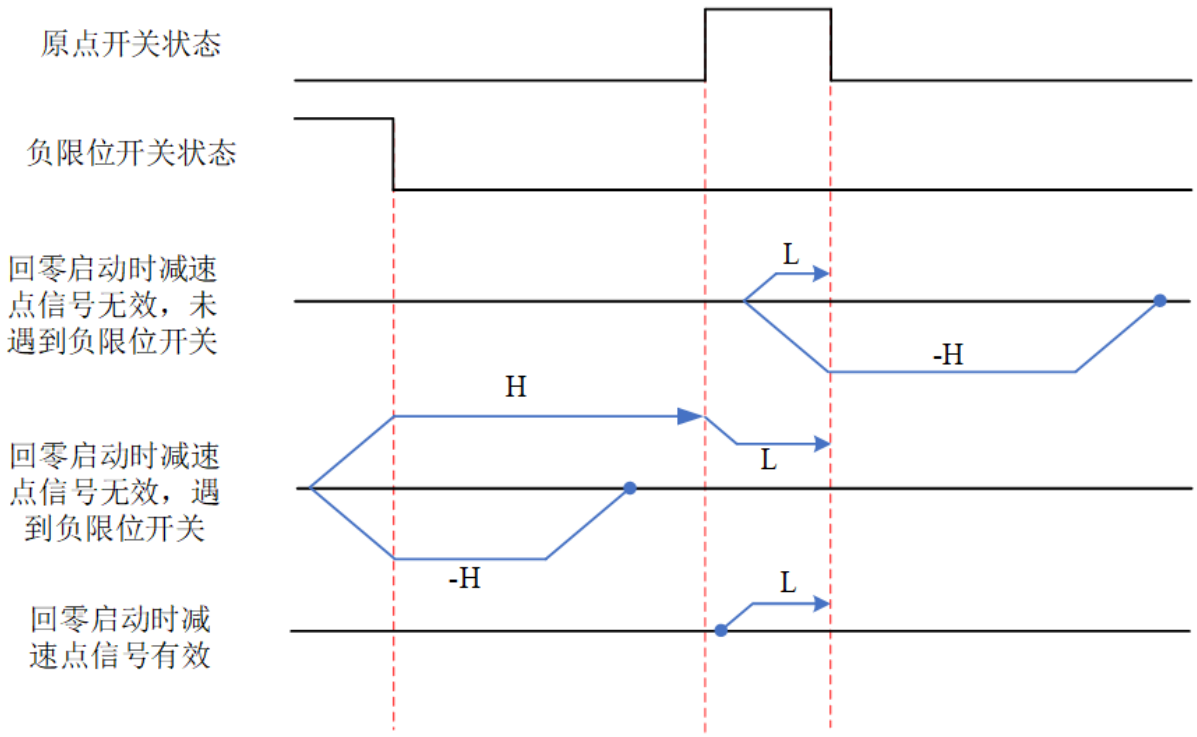
运动轨迹 1：回零启动时原点开关状态无效，则以高速朝负向运动，在负向运动时遇到原点开关的上升沿之后减速停止，然后换低速朝正向运动。在低速朝正向运动过程中，寻找原点开关的下降沿所在的位置作为原点。

运动轨迹 2：回零启动时原点开关状态无效，则以高速朝负向运动，在负向运动时遇到负向限位开关的有效状态时减速停止，然后以高速朝正向运动。在正向运动时遇到原点开关的上升沿后开始减速至低速，然后以低速继续朝正向运动。在低速朝正向运动过程中，寻找原点开关的下降沿所在的位置作为原点。

运动轨迹 3：回零启动时原点开关状态有效，则以低速朝正向运动。在低速朝正向运动过程中，寻找原点开关的下降沿所在的位置作为原点。

这种模式下，朝负向运动首先遇到负向限位开关的有效状态时自动反向；遇到正向限位开关的有效状态时，则中止回原点流程，即该指令被打断。

模式 27 回归过程如下图所示：



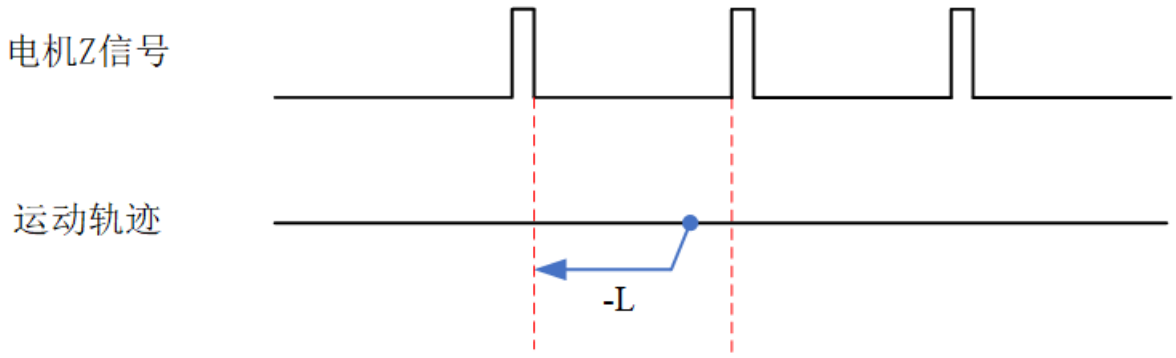
（13） 模式 33，寻找朝负向运动时最近的 Z 信号。

原点：电机 Z 信号

运动轨迹：回零启动后以低速朝负向运动，找最近的 Z 信号位置作为原点。

这种模式下，无论遇到正限位开关还是负限位开关的有效状态，都会中止回原点流程，即该指令被打断。

模式 33 回归过程如下图所示：



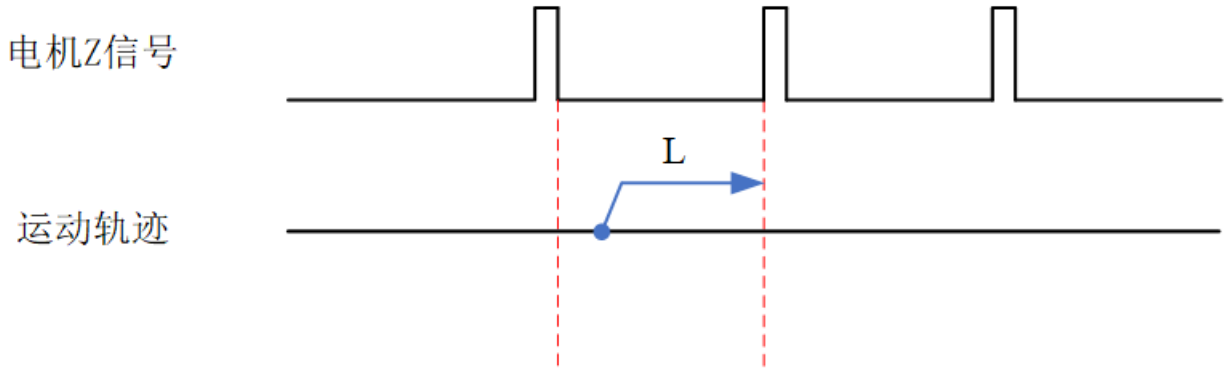
（14） 模式 34，寻找朝正向运动时最近的 Z 信号。

原点：电机 Z 信号

运动轨迹：回零启动后以低速朝正向运动，找最近的 Z 信号位置作为原点。

这种模式下，无论遇到正限位开关还是负限位开关的有效状态，都会中止回原点流程，即该指令被打断。

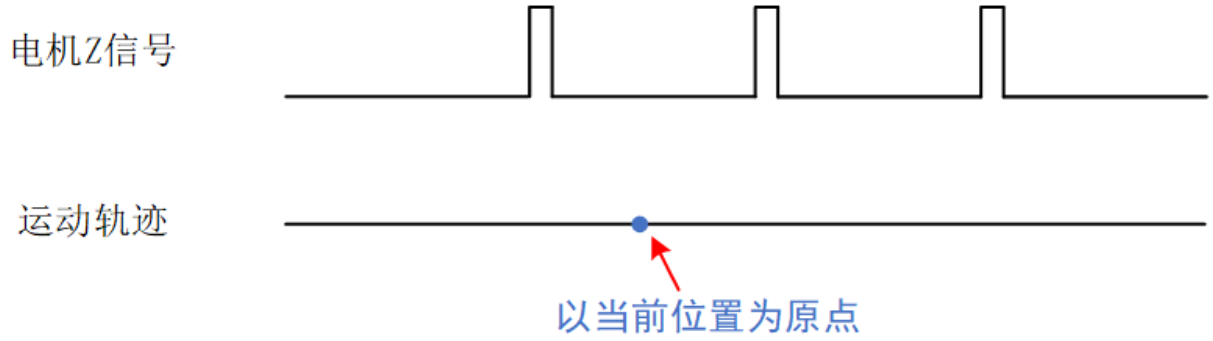
模式 34 回归过程如下图所示：



（15） 模式 35，以当前位置为原点。

运动轨迹：以回零启动时的位置为原点，进行原点回零。

模式 35 回归过程如下图所示：

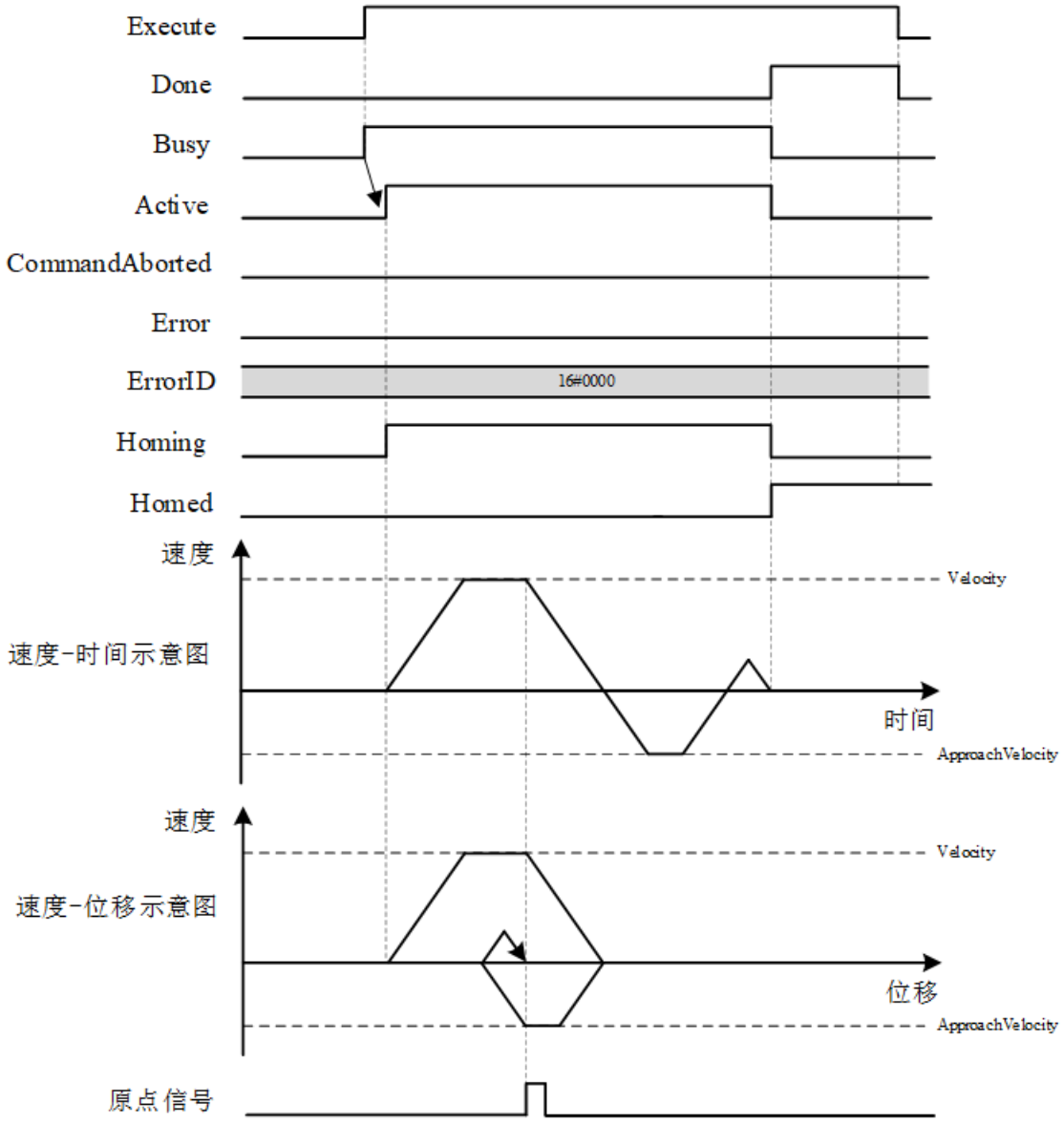


■ 功能块时序图

（1） 启动原点回归，正常执行原点回归动作的时序图。

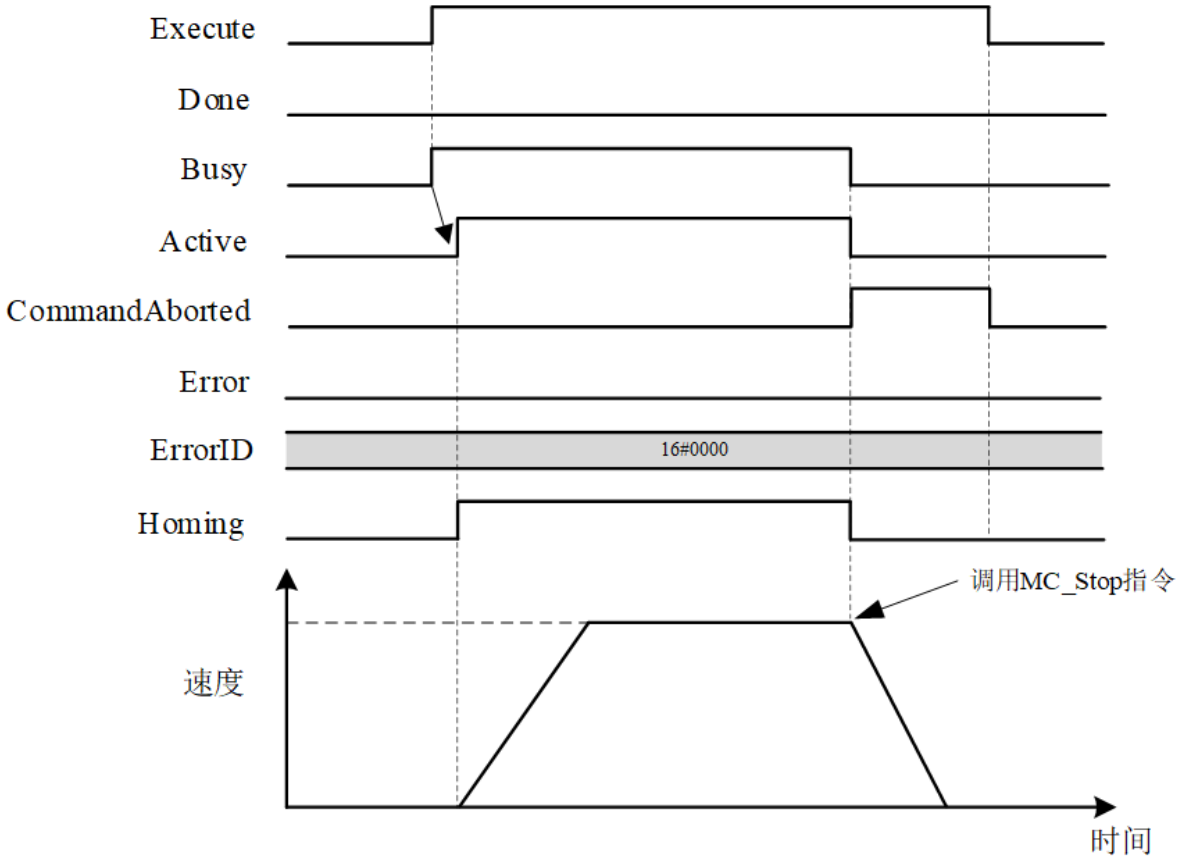
原点回归过程中轴状态为 Homing 状态。

下图以原点回归模式 19 为例说明回原过程。

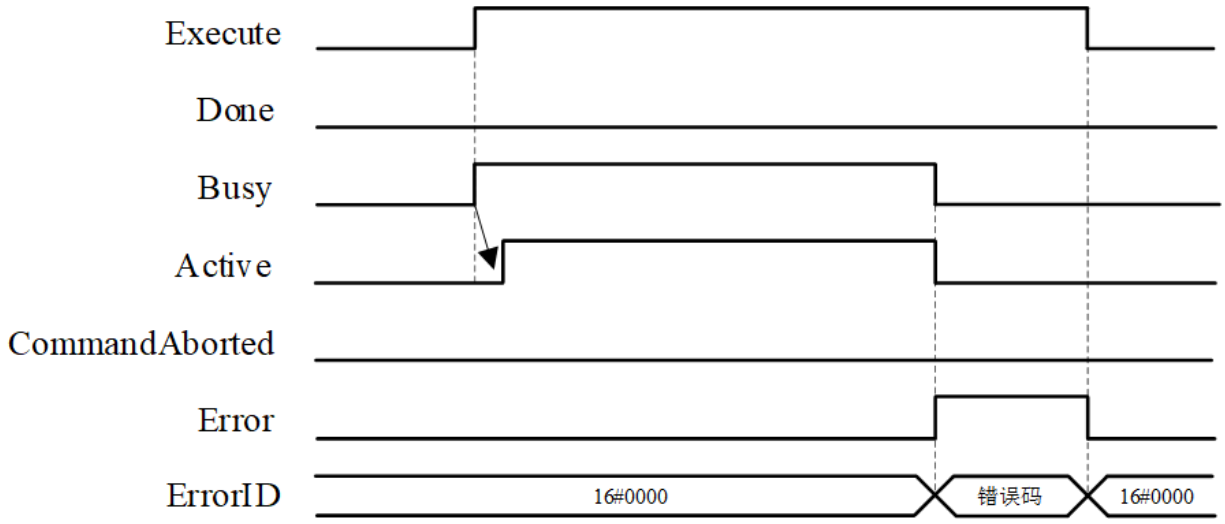


（2） 原点回归过程中调用 MC_Stop 指令打断原点回归动作后的时序图。

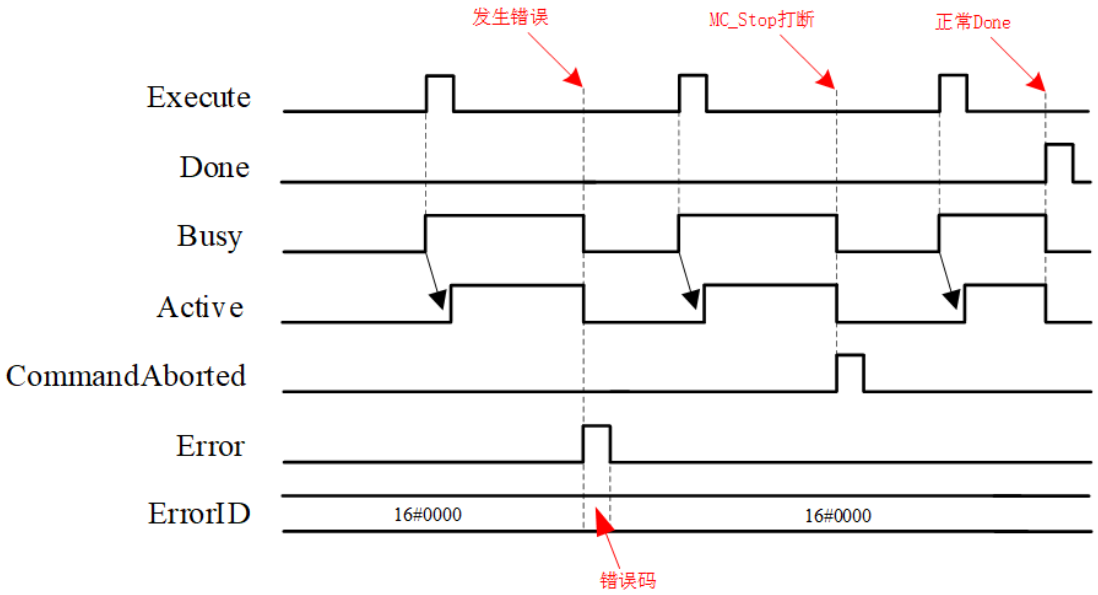
调用 MC_Stop 打断的时候，减速度使用 MC_Stop 入参设置的减速度。



（3） 原点回归过程中出现故障的时序图。

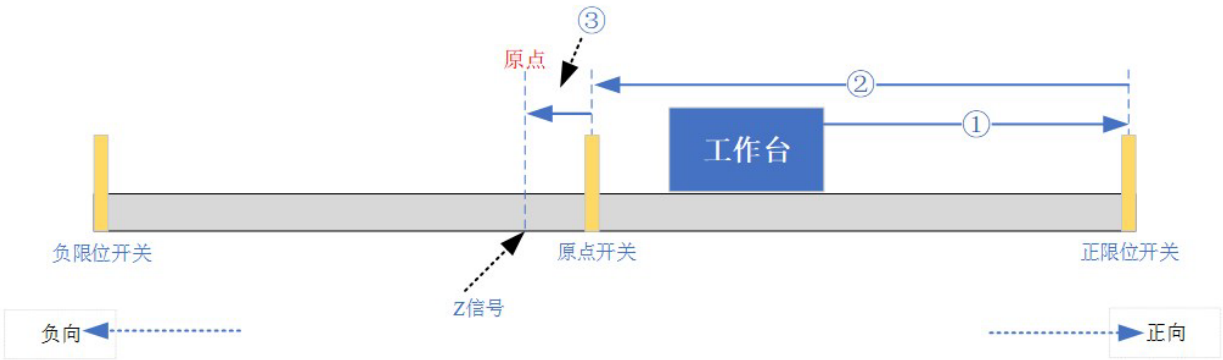


（4） Execute 保持一个周期时，功能块运行过程中发生错误、被 MC_Stop 打断和正常执行完成的时序图。



5.4.3 示例

本示例工艺要求工作台工作前先进行寻找原点回归的步骤，原点位置为原点开关负向的第一个 Z 信号，成功找到原点后，才能执行其它的工艺流程。



基于以上要求，使用 MC_Home 指令进行原点回归操作，原点回归模式应选择 7。以工作台目前停止的位置为例，使用原点回归模式 7 时，工作台先正向以速度 Velocity 高速运动，遇到正限位开关后，减速，并以速度 Velocity 高速反向寻找原点开关，找到原点开关后，再以速度 ApproachVelocity 低速寻找最近的 Z 信号作为原点。

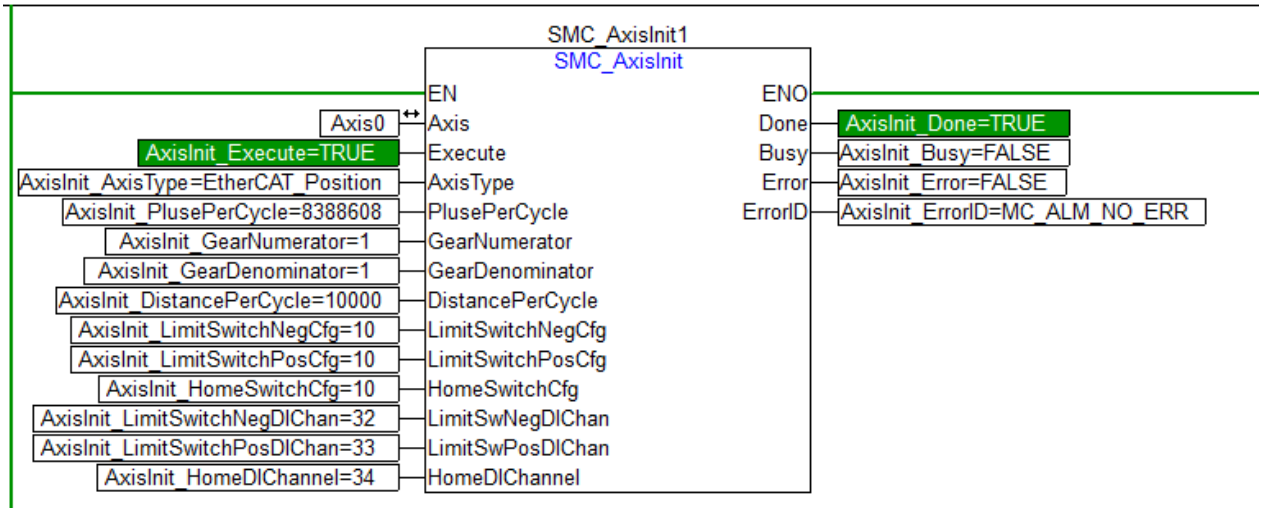
■ 变量

序号	变量名	变量类型	初始值	变量说明

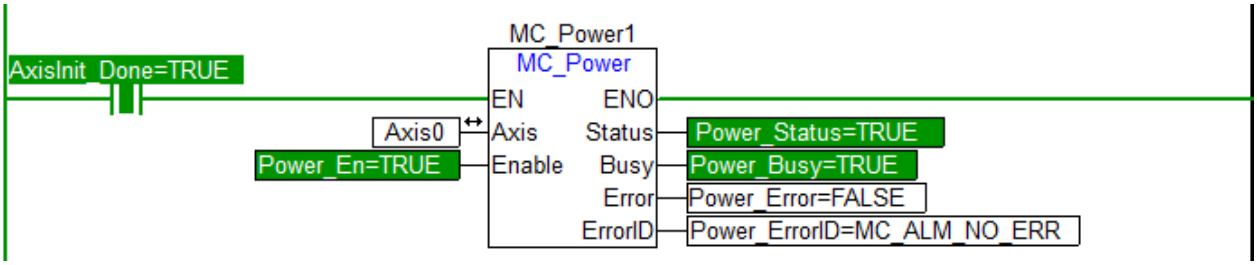
1	MC_Home	MC_HOME	-	原点回归指令
2	Home_En	BOOL	FALSE	原点回归使能标志
3	Home_HomeMode	SMC_HOME_MODE	MC_HM1_NEG_LIMIT_Z	原点回归模式
4	Home_Vel	LREAL	10000	原点回归高速找减速点速度
5	Home_Vel2	LREAL	5000	原点回归低速找原点速度
6	Home_Acc	LREAL	10000	原点回归加速度
7	Home_Dec	LREAL	10000	原点回归减速度
8	Home_Jerk	LREAL	0	原点回归加加速度
9	Home_Pos	LREAL	0	原点回归完成后零点位置
10	Home_Done	BOOL	FALSE	原点回归完成标志
11	Home_Busy	BOOL	FALSE	原点回归忙状态标志
12	Home_Active	BOOL	FALSE	原点回归指令运行标志
13	Home_Abort	BOOL	FALSE	原点回归中止标志
14	Home_Error	BOOL	FALSE	原点回归错误标志
15	Home_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	原点回归错误码
16	MC_MoveAbsolute	MC_MOVEABSOLUTE	-	绝对位置定位指令
17	MoveAbs_En	BOOL	FALSE	绝对位置定位使能标志
18	MoveAbs_Pos	LREAL	20000	绝对位置定位目标位置
19	MoveAbs_Vel	LREAL	10000	绝对位置定位速度
20	MoveAbs_Acc	LREAL	100000000	绝对位置定位加速度
21	MoveAbs_Dec	LREAL	100000000	绝对位置定位减速度
22	MoveAbs_Dir	MC_DIRECTION	mcPositiveDirection	绝对位置定位方向
23	MoveAbs_Jerk	LREAL	0	绝对位置定位加加速度
24	MoveAbs_Buff	MC_BUFFER_MODE	Aborting	绝对位置定位缓冲模式
25	MoveAbs_Done	BOOL	FALSE	绝对位置定位完成标志
26	MoveAbs_Busy	BOOL	FALSE	绝对位置定位忙状态标志
27	MoveAbs_Active	BOOL	FALSE	绝对位置定位指令激活标志
28	MoveAbs_Abort	BOOL	FALSE	绝对位置定位中止标志
29	MoveAbs_Error	BOOL	FALSE	绝对位置定位错误标志
30	MoveAbs_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	绝对位置定位错误代码

■ LD 程序实现

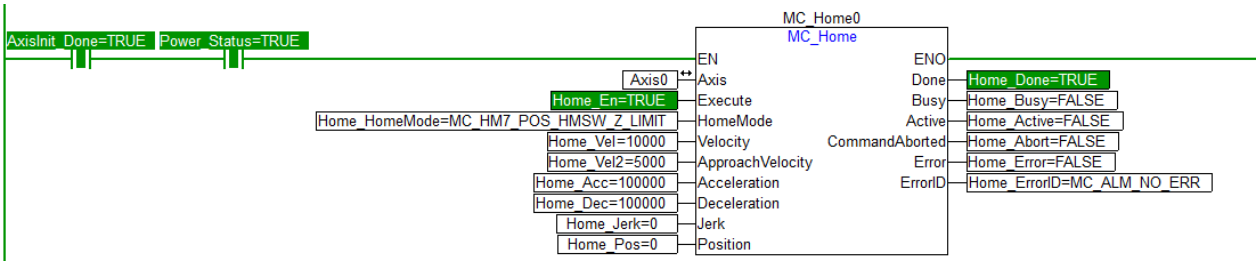
（1） 首先需要调用 SMC_AxisInit 指令初始化轴参数，将轴类型设置为 EtherCAT_Poaition，并设置用户单位，原点开关信号以及正负限位开关信号。指令如下：



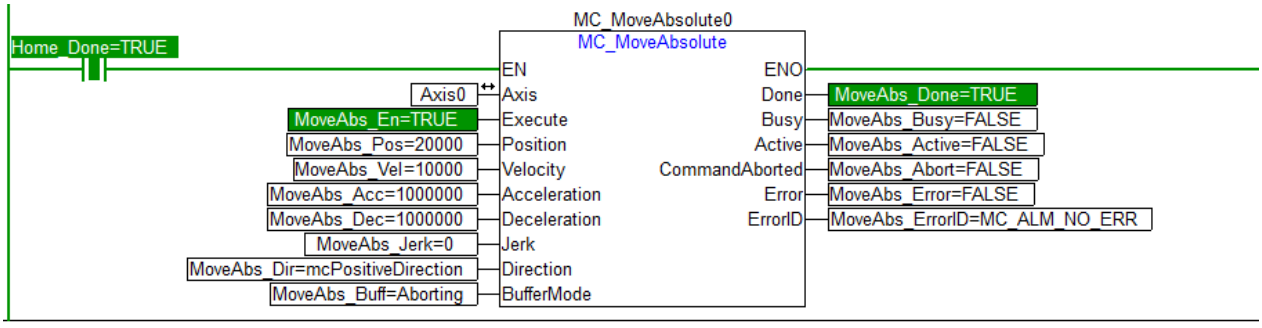
（2） 在调用 MC_Home 指令之前先要调用 MC_Power 指令使能对应轴。



（3） 调用 MC_Home 指令进行原点回归操作，原点回归模式设置为 7。



（4） 原点回归完成后，再执行其它工艺流程，如以原点为起点进行绝对位置定位操作。



■ ST 程序实现

（1） 首先需要调用 SMC_AxisInit 指令初始化轴参数，将轴类型设置为 EtherCAT_Poaition，并设置用户单位，原点开关信号以及正负限位开关信号。指令如下：

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

（2） 再调用 MC_Home 指令之前先要调用 MC_Power 指令使能对应轴。

```
IF TRUE = AxisInit_Done THEN
   Power_En := TRUE;
ELSE
   Power_En := FALSE;
END_IF;
MC_Power1(
Enable := Power_En,
Axis := Axis0,
Status=>Power_Status,
Busy=>Power_Busy,
Error=>Power_Error,
ErrorID=>Power_ErrorID);
```

（3） 调用 MC_Home 指令进行原点回归操作，原点回归模式设置为 7。

```
IF TRUE = Power_Status THEN
   MC_Home0(
```

```
    Execute := Home_En,
    HomeMode := Home_HomeMode,
    Velocity := Home_Vel,
    ApproachVelocity := Home_Vel2,
    Acceleration := Home_Acc,
    Deceleration := Home_Dec,
    Jerk := Home_Jerk,
    Position := Home_Pos,
    Axis := Axis0,
    Done=>Home_Done,
    Busy=>Home_Busy,
    Active=>Home_Active,
    CommandAborted=>Home_Abort,
    Error=>Home_Error,
    ErrorID=>Home_ErrorID);
END_IF
```

（4） 原点回归完成后，再执行其它工艺流程，如以原点为起点进行绝对位置定位操作。

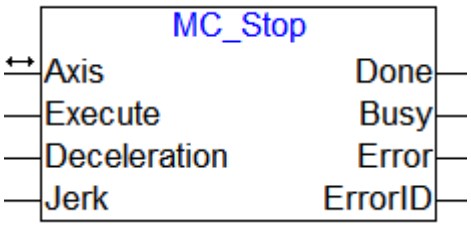
```
IF TRUE=Home_Done THEN
    MC_MoveAbsolute0(
    Execute := MoveAbs_En,
    Position := MoveAbs_Pos,
    Velocity := MoveAbs_Vel,
    Acceleration := MoveAbs_Acc,
    Deceleration := MoveAbs_Dec,
    Jerk := MoveAbs_Jerk,
    Direction := MoveAbs_Dir,
    BufferMode := MoveAbs_Buff,
    Axis := Axis0,
    Done=>MoveAbs_Done,
    Busy=>MoveAbs_Busy,
    Active=>MoveAbs_Active,
    CommandAborted=>MoveAbs_Abort,
    Error=>MoveAbs_Error,
    ErrorID=>MoveAbs_ErrorID);
END_IF
```

5.4.4 使用注意事项

该指令在设置为 Z 信号相关的原点回归模式时，若探针捕获通道 1 已经被占用，则该指令报错，MC_TouchProbe 和 HMC_GantryInit 两个指令会占用探针捕获通道，在使用时，错峰使用探针捕获通道 1 进行 Z 信号捕获。

5.5 MC_Stop（轴停止）

控制轴按照指定减速度停止，中止该轴上任何正在执行的运动控制指令，并撤销该轴上所有运动缓存指令。



5.5.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Deceleration	减速度	否	-	正值	LREAL
	Jerk	加加速度 0：梯形加减速 正值：S 形加减速	是	0	0/正值	LREAL
OUTPUT	Done	停止完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.5.2 功能

本指令控制轴按照指定减速度，由当前速度减速停止，中止该轴上任何正在执行的运动控制指令。

■ 指令功能详情

（1） 减速参数设定

减速停止时的减速度、加加速度分别由输入变量的 Deceleration(减速度)、Jerk(加加速度)指定。

（2） 运行本指令时的轴状态

指令上升沿触发引脚持续高电平时，轴一直处于 Stopping 状态时，轴无法投送运动指令。

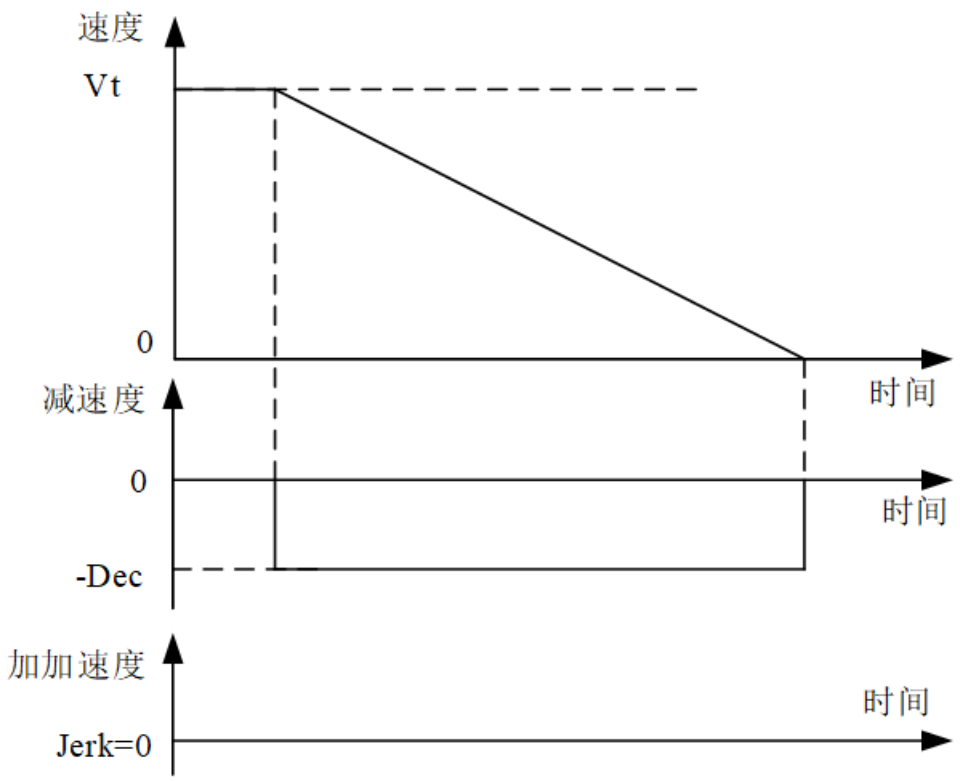
（3） 轴锁存

本指令上升沿触发，触发时将输入输出参数 Axis 的轴 ID、输入参数 Deceleration 和 Jerk 值锁存，指令执行过程中，修改指令输入的参数无效，直到重复触发本指令。

（4） 加减速曲线

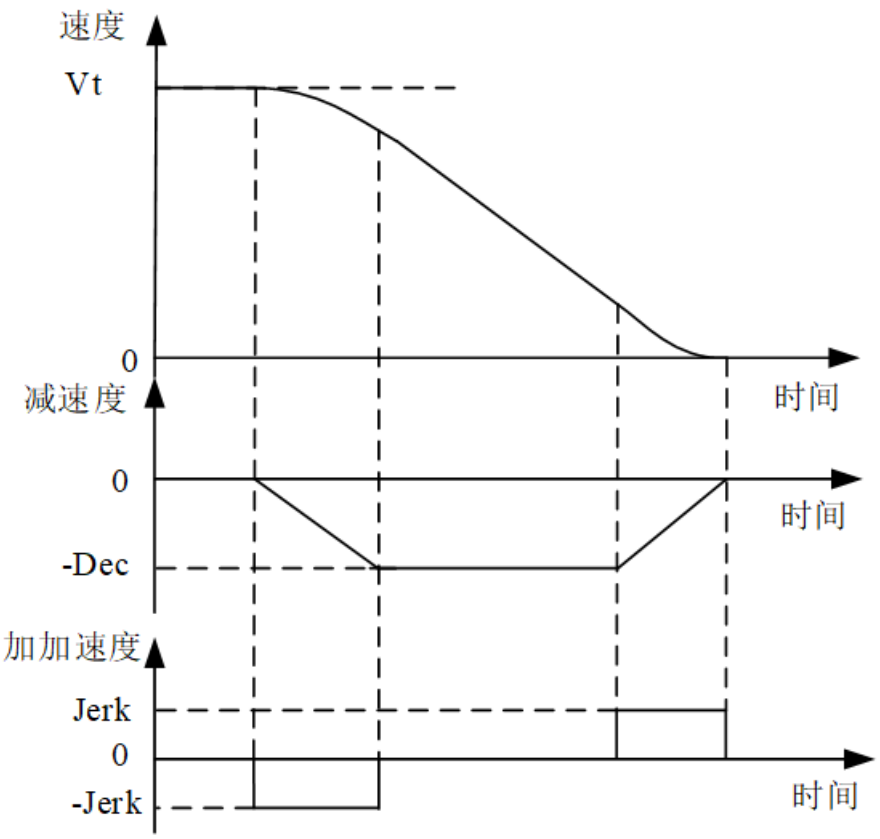
若输入参数 Jerk=0 时，则减速曲线为梯形加减速曲线；若输入参数 Jerk>0 时，则减速曲线为 S 形加减速曲线。

□ Jerk=0 时，减速曲线为梯形加减速曲线，如下所示：



Vt:减速启动时的速度，Dec：减速度的设定值，Jerk：加加速度的设定值

□ Jerk>0 时，以减速度生成速度的指令值。减速曲线为 S 形加减速曲线，如下所示：



Vt:减速启动时的速度，Dec：减速度的设定值，Jerk：加加速度的设定值

- 本指令支持的轴类型：

  Virtual（0）：虚轴

  EtherCAT_Position（50）：EtherCAT 总线连接轴，位置控制

  EtherCAT_Speed（51）：EtherCAT 总线连接轴，速度控制

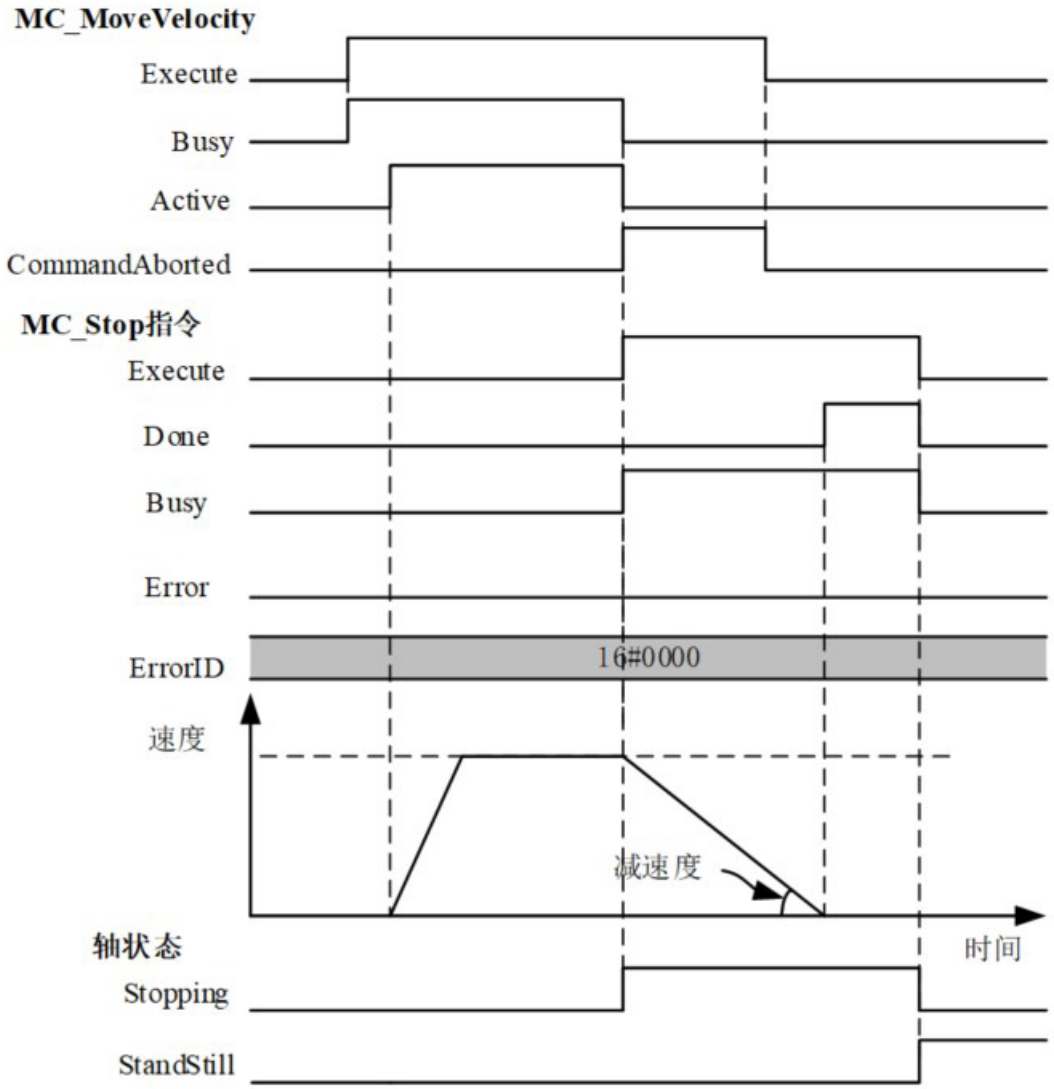
  EtherCAT_Torque（52）：EtherCAT 总线连接轴，转矩控制

- 时序图

  （1） 正常时序图

  在 Execute 输入的上升沿，若无异常情况，Busy 引脚置为 TRUE，指令执行时，轴处于 Stopping 状态，轴运动减速停止。减速完成后 Done 置为 TRUE。若 Execute 保持高电平，则 Busy、Done 引脚保持 TRUE，轴持续处于 Stopping 状态；若 Execute 置低电平，则 Busy、Done 引脚置为 FALSE，轴切换为 StandStill 状态。

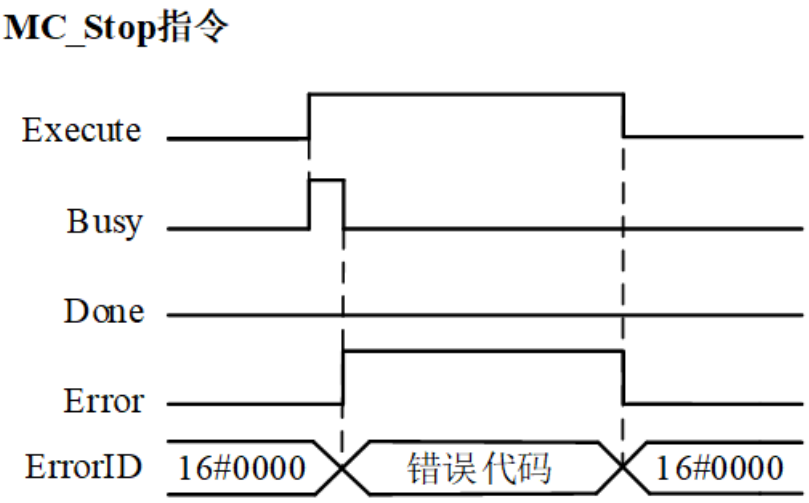
指令执行时的正确时序如下：



（2） 异常错误发生时引脚时序

执行本指令时，指令输入参数非法、轴处于 Disabled 状态或者 ErrorStop 状态等异常时，调用本指令失败。此时，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，可以了解错误发生原因。

指令输入引脚 Execute 为低电平时，所有输出引脚恢复为默认值。

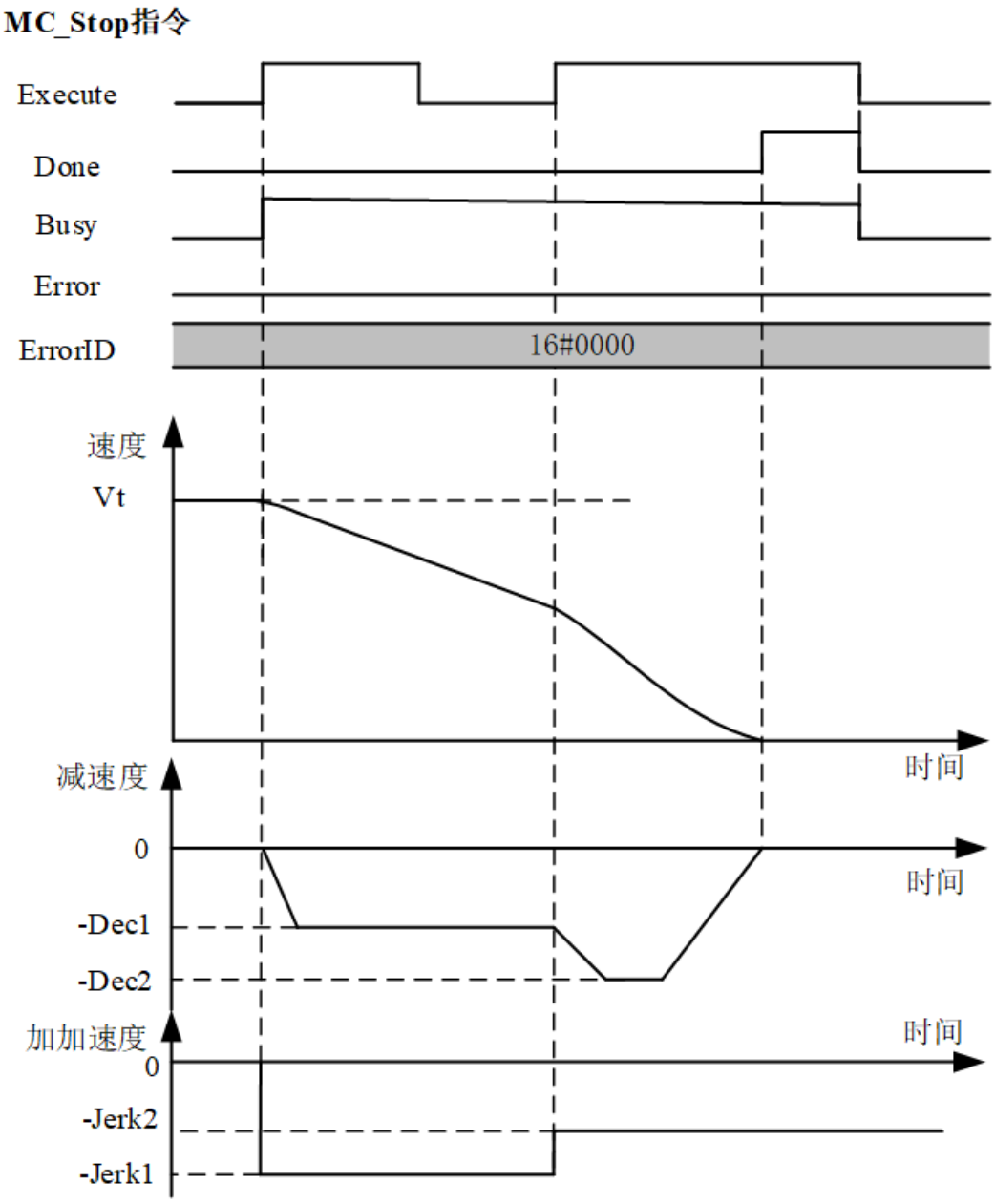


当指令的异常情况解决后，需要重新上升沿触发 Execute，启动 MC_Stop 功能块，执行控制轴按照指定减速度停止动作。

（3） 重复投送指令

多个 MC_Stop 功能块重复触发时，若 Deceleration（减速度）和 Jerk（加加速度）有变更时，以最后一次投送指令的减速度和加加速度进行减速停止控制，指令对减速度和加加速度进行变更。

下图为重复投送指令的时序图。



Vt：减速开始时的速度，Dec1：第一次投送指令时的减速度，Dec2：第二次投送时的减速度，Jerk1：第一次投送指令时的加加速度，Dec2：第二次投送时的加加速度

5.5.3 示例

本示例中，设定 Axis0 为指令轴，变量类型为 AXIS_REF。利用 MC_Stop 指令停止 MC_MoveVelocity 运动指令，MC_ReadStatus 指令用于读取当前轴状态。

■ 梯形图

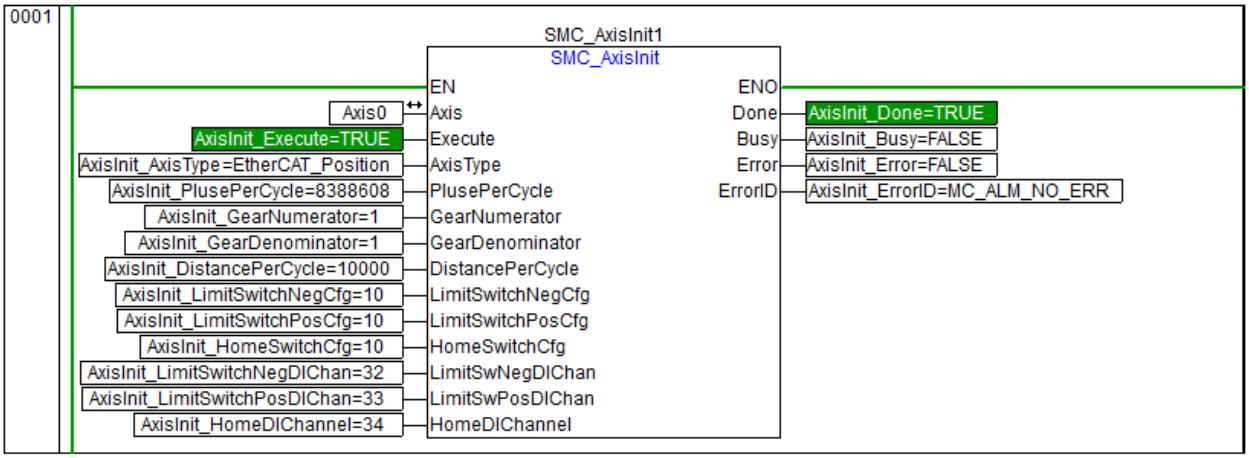
利用梯形图组建程序，输入参数 Deceleration 为 10000，Jerk 为 0，梯形加减速。

主要变量说明

变量名称	变量类型	初始值	注释
Axis0	AXIS_REF	-	轴 0 的轴变量
Power1_Status	BOOL	FALSE	轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
MV1_Active	BOOL	FALSE	轴的速度控制指令动作时的变量。轴控制运动时，该变量变为 TRUE。
MV1_Comd	BOOL	FALSE	速度控制指令被中断的变量。该变量变为 TRUE，速度指令被中断。
Stop1_Exe	BOOL	FALSE	轴停止指令的上升沿触发变量。
Stop1_Done	BOOL	FALSE	轴停止成功时的变量。Done=TRUE，轴停止完成。
Switch1	BOOL	FALSE	轴准备就绪的状态变量，轴初始化完成后， Switch 为 TRUE，轴准备就绪。

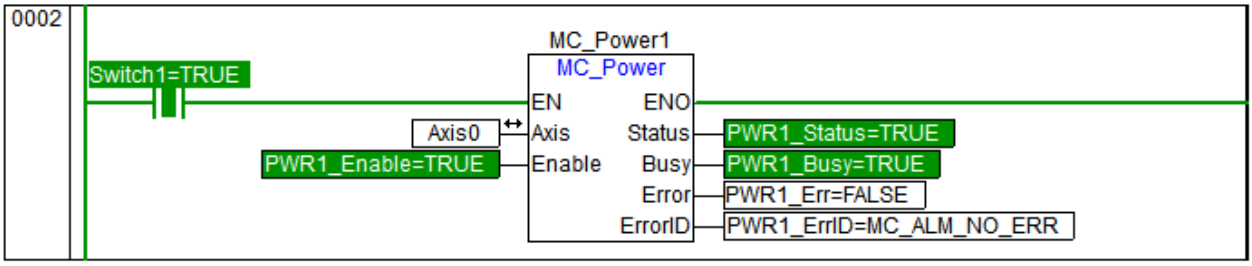
（1） 轴初始化

调用 SMC_AxisInit 指令完成 Axis0 的初始化配置，轴类型设为 50：EtherCAT_Position。SMC_AxisInit 指令使用方法参见 SMC_AxisInit 的指令说明章节。



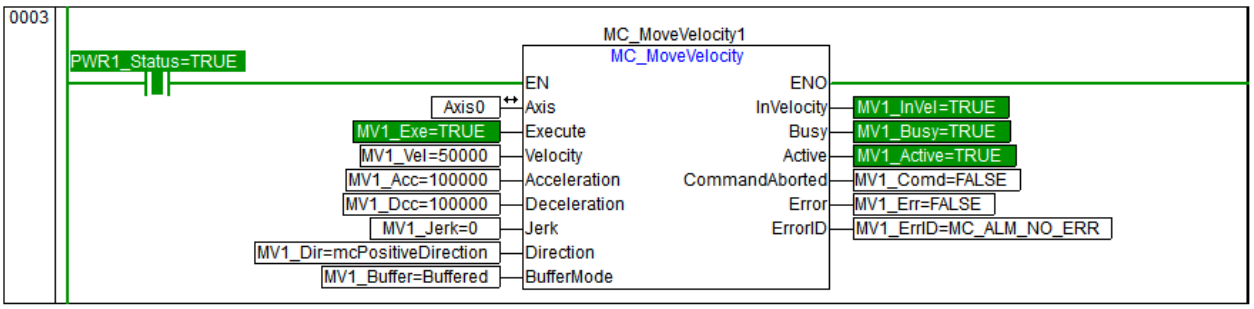
（2） 轴使能

0002 节中，轴初始化完成，轴准备就绪的状态变量 Switch1 变为 TRUE，调用 MC_Power 对轴使能。



（3） 轴投送运动指令

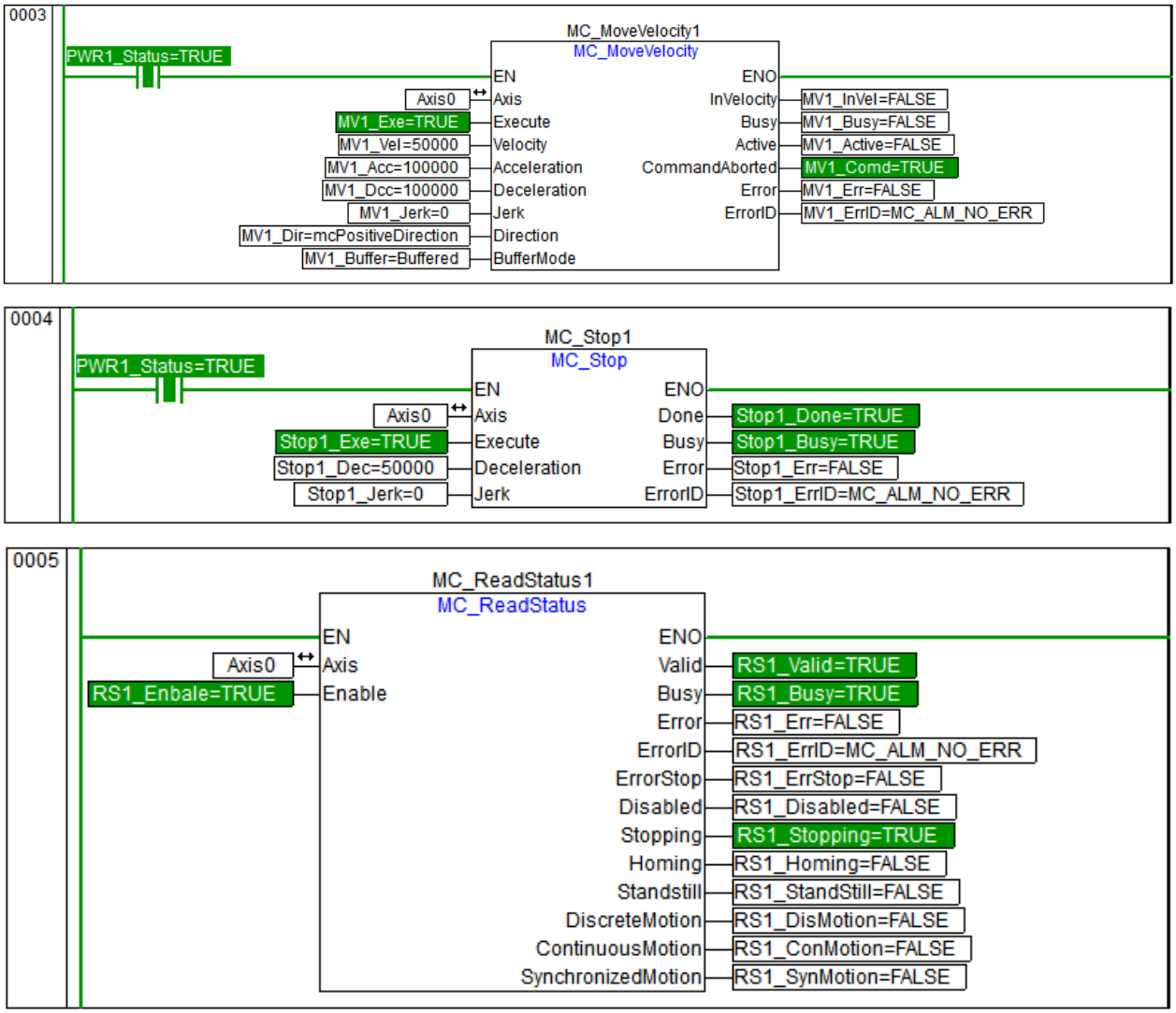
0003 节中，轴使能成功后，调用 MC_MoveVelocity 控制轴运动。



（4） 轴投送减速停止指令

0004 节中，调用 MC_Stop 中断 MC_MoveLocity 运动，控制轴减速停止。MC_MoveLocity 的 CommandAborted 引脚变为 TRUE。

MC_Stop 的 Execute 引脚保持高电平时，0005 节中读取到轴的状态持续为 Stopping 状态。



■ ST 语言

利用 ST 语言组建程序，输入参数 Deceleration 为 10000，Jerk 为 0，梯形加减速。

主要变量说明

变量名称	变量类型	初始值	注释
Axis0	AXIS_REF	-	轴 0 的轴变量
Power1_Status	BOOL	FALSE	轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
MV1_Active	BOOL	FALSE	轴的速度控制指令动作时的变量。轴控制运动时，该变量变为 TRUE。
Stop1_Exe	BOOL	FALSE	轴停止指令的上升沿触发变量。
Stop1_Done	BOOL	FALSE	轴停止成功时的变量。Done=TRUE，轴停止完成。
Switch1	BOOL	FALSE	轴准备就绪的状态变量，轴初始化完成后， Switch 为 TRUE，轴准备就绪。
InitFlag	BOOL	FALSE	如果该变量为 TRUE，则初始化参数设置完成。

（1） 轴初始化

Axis0 的初始化配置，轴类型设为 50：EtherCAT_Position。SMC_AxisInit 指令使用方法参见其指令说明章节。

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

（2） 停止指令输入参数初始。

```
IF FALSE=InitFlag THEN
Stop1_Dec:=10000;(*设定 MC_Stop 指令的减速度参数*)

Stop1_Jerk:= 0; (*设定 MC_Stop 指令的 Jerk 参数*)

InitFlag:=TRUE;
END_IF
```

（3） 轴使能

轴初始化完成，轴准备就绪状态 Switch1 为 TRUE 后，调用 MC_Power 完成轴使能。

```
IF Switch1 THEN
 MC_Power1(
 Enable := PWR1_Enable,
 Axis := Axis0,
 Status=> PWR1_Status,
```

```
 Busy=> PWR1_Busy,
 Error=> PWR1_Err,
 ErrorID=> PWR1_ErrID);
END_IF
```

（4） 投送运动指令

调用 MC_MoveVelocity，给 Axis0 投送运动指令。

```
IF  PWR1_Status THEN
   MC_MoveVelocity1(
   Execute := MV1_ Exe,
   Velocity := MV1_ Vel,
   Acceleration := MV1_ Acc,
   Deceleration := MV1_ Dcc,
   Jerk := MV1_ Jerk,
   Direction := MV1_ Dir,
   BufferMode := MV1_ Buffer,
   Axis := Axis0,
   InVelocity=> MV1_ InVel,
   Busy=> MV1_ Busy,
   Active=> MV1_ Active,
   CommandAborted=> MV1_ Comd,
   Error=> MV1_ Err,
   ErrorID=> MV1_ Error);
END_IF
```

（5） 运行 MC_Stop 指令

调用 MC_Stop 指令控制轴减速停止 MC_MoveVelocity 运动。

```
IF  PWR1_Status THEN
   MC_Stop1(
   Execute := Stop1_Exe,
   Deceleration := Stop1_Dec,
   Jerk := Stop1_Jerk,
   Axis := Axis0,
   Done=> Stop1_Done,
   Busy=> Stop1_Busy,
   Error=> Stop1_Err,
   ErrorID=> Stop1_ErrID);
```

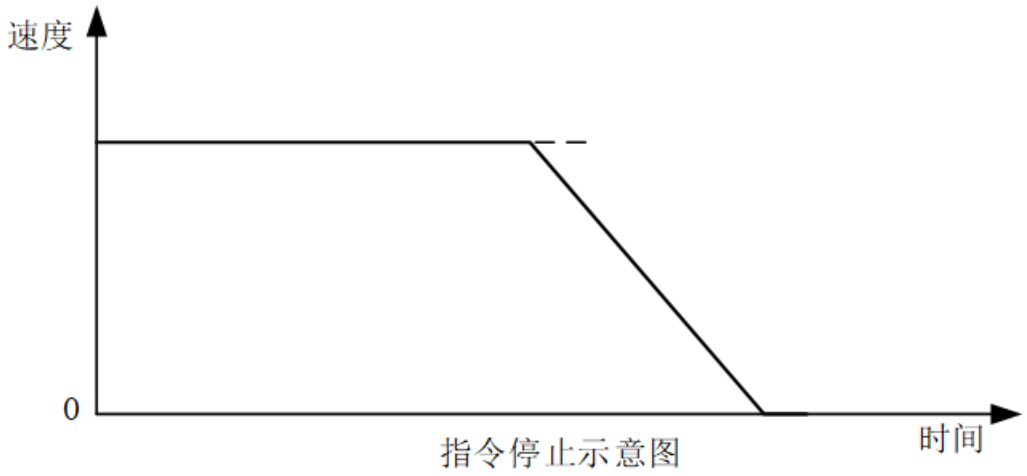
```
END_IF
```

（6） 运行 MC_ReadStatus

调用 MC_ReadStatus 查看轴状态。

```
MC_ReadStatus1(
Enable := RS1_Enbale,
Axis := Axis0,
Valid=> RS1_Valid,
Busy=> RS1_Busy,
Error=> RS1_Err,
ErrorID=> RS1_ErrID,
ErrorStop=> RS1_ErrStop,
Disabled=> RS1_Disabled,
Stopping=> RS1_Stopping,
Homing=> RS1_Homing,
Standstill=> RS1_StandStill,
DiscreteMotion=> RS1_DisMotion,
ContinuousMotion=> RS1_ConMotion,
SynchronizedMotion=> RS1_SynMotion);
```

MC_MoveVelocity 运动指令减速停止示意图如下所示：



指令停止示意图

5.5.4 使用注意事项

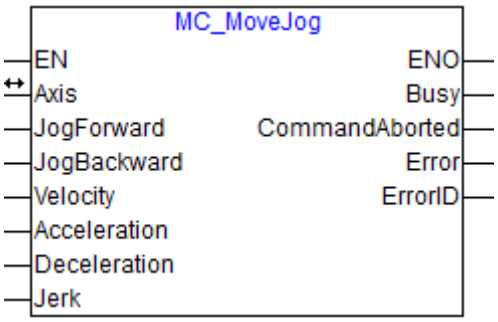
- MC_Stop 指令执行成功，若输入参数 Execute 保持高电平，则轴持续处于 Stopping 状态，禁止投送运

动指令。

- 输入参数 Execute 为低电平时，停止功能被释放，轴切换到 StandStill 状态，可以正常投送运动指令。

5.6 MC_MoveJog（JOG 点动）

单轴进行正向/负向的点动指令



5.6.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	JogForward	正向使能	否	-	TRUE/FALSE	BOOL
	JogBackward	负向使能	否	-	TRUE/FALSE	BOOL
	Velocity	目标速度	否	-	非负	LREAL
	Acceleration	加速度	否	-	正值	LREAL
	Deceleration	减速度	否	-	正值	LREAL
	Jerk	0：梯形加减速 正值：S 形加减速加加速度	是	0	0/正值	LREAL
OUTPUT	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.6.2 功能

本指令可以按照设置的速度，加减速度，加加速度等参数进行正负点动运动，对该指令的详细介绍如

下：

（1） 本指令高电平触发有效，按照设定的参数进行移动，执行过程中，修改锁存的参数无效，直到再次触发高电平生效，，将 JogForward(正向使能)设为 TRUE 时轴进行正向移动，将 JogBackward(负向使能)设为 TRUE 时轴进行负向点移动。

（2） 当 JogForward(正向使能)和 JogBackward(负向使能)同时为 TRUE 时，轴进行正向移动。

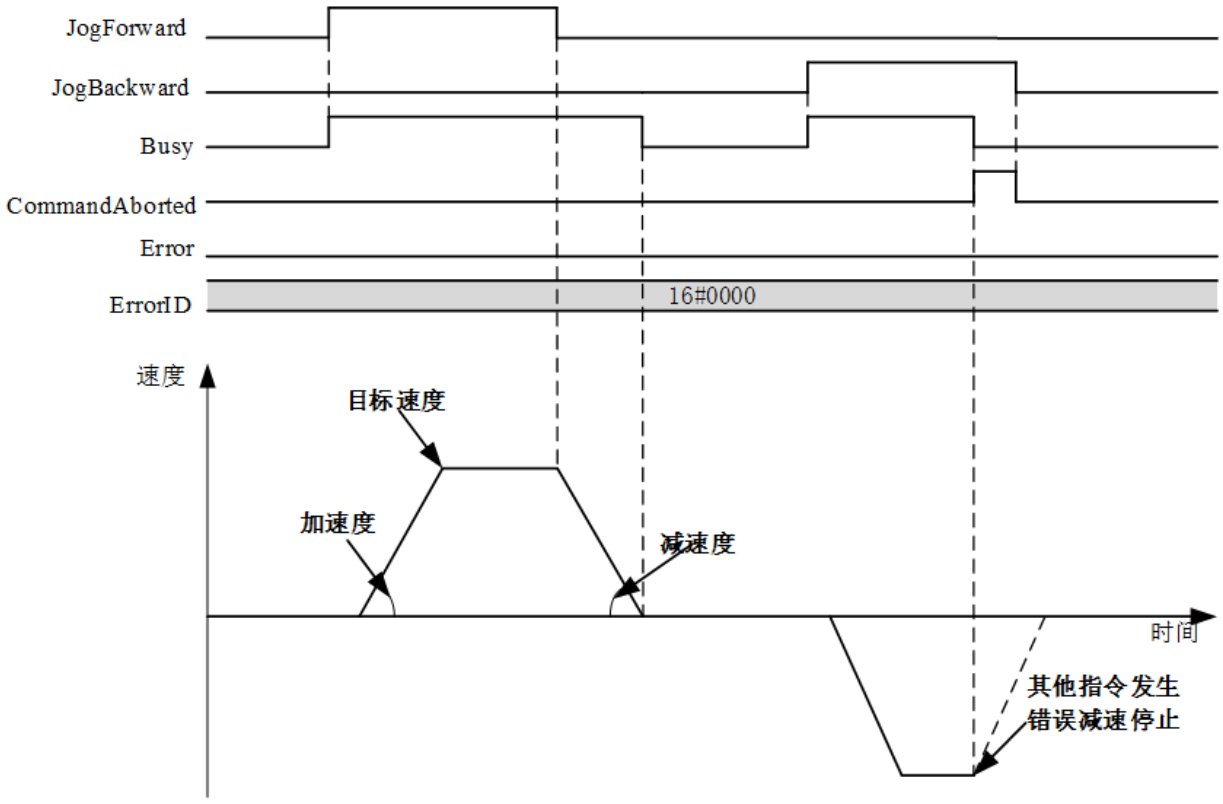
（3） 将 JogForward 从 TRUE 变为 FALSE 时，轴按照设定的减速度减速停止，在减速过程中由于轴未停止，Busy(忙标志)不会变为 FALSE，负向时也是一样的情况。

- 重启和多重启动运动指令
  - 同向重启时，将 JogForward(正向使能)或 JogBackward(负向使能)设为 FALSE 并减速的过程中，再次将 JogForward(正向使能)或 JogBackward(负向使能)设为 TRUE 时，轴从减速途中进行加速，重新以新的运动参数进行移动。
  - 不同方向重启指令时，在将 JogForward(正向使能)设为 TRUE，正方向进行运动时，如果将 JogBackward(负向使能)设为 TRUE，则此时方向反转，进行反向运动。此时，功能块以 JogBackward(负向使能)设为 TRUE 时的输入变量进行运动。同理，在将 JogBackward(负向使能)设为 TRUE，负向转动时，如果将 JogForward(正向使能)设为 TRUE，也会发生同样的运动。
  - JogForward(正向使能)和 JogBackward(负向使能)引脚说明：在将 JogForward(正向使能)设为 TRUE，之后将 JogBackward(负向使能)设为 TRUE 进行负向转动时，将 JogBackward(负向使能)设为 FALSE，此时轴减速停止，并不会因为 JogForward(正向使能)为 TRUE 就进行正向运动，需要进行正转时，重新进行高电平触发即可。
- 功能块时序图

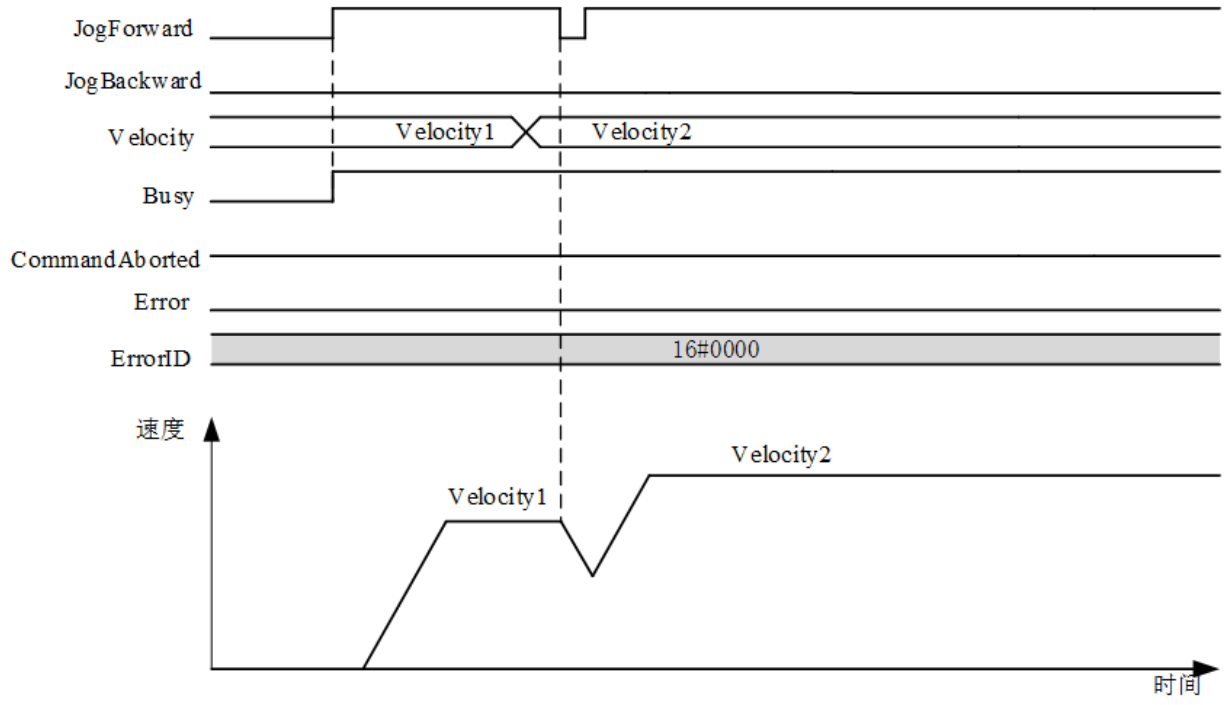
在功能块启动 JogForward(正向使能)或 JogBackward(负向使能)为 TRUE 时，Busy(忙标志)变为 TRUE。

在 JogForward(正向使能)或 JogBackward(负向使能)的下降沿开始减速并停止轴的同时，Busy(忙标志)变为 FALSE，利用其他指令中止本指令时，CommandAborted(中止执行)变为 TRUE，Busy(忙标志)变为 FALSE。

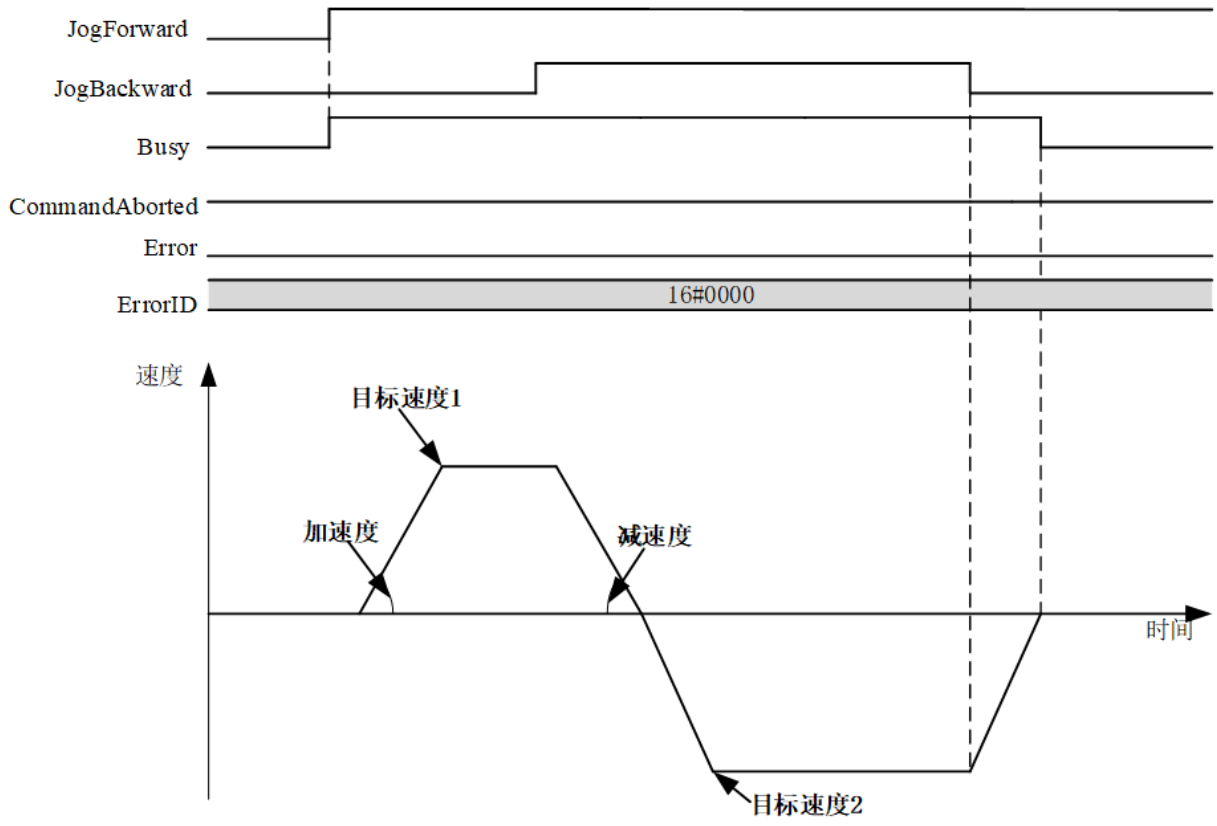
（1） 正常引脚时序



(2) 同向重启时序图



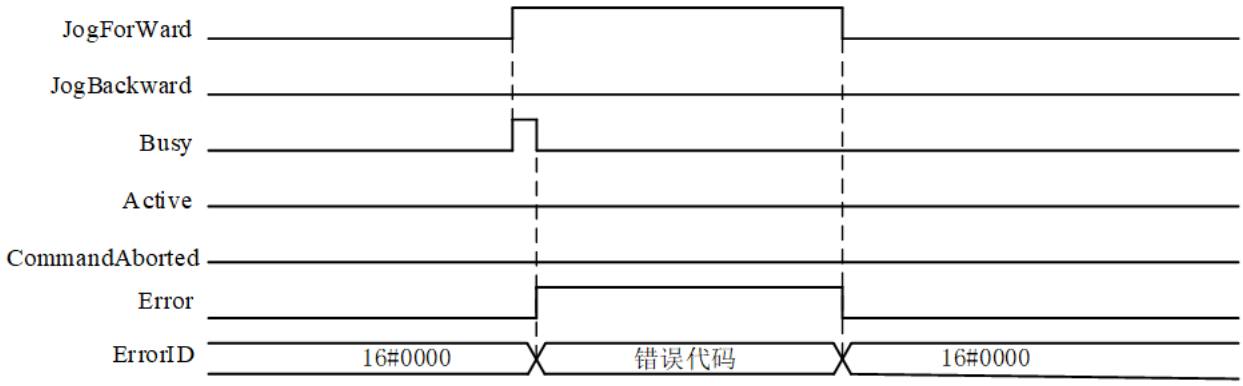
(3) 不同方向重启时序图



（4） 异常情况时序图

在执行指令发生异常时，Error(错误)变为 TRUE，轴停止运动。

可以查看 ErrorID(错误码)的值，了解错误发生的具体原因。



5.6.3 示例

组建 MC_MoveJog 示例程序进行模拟速度运动指令运行。

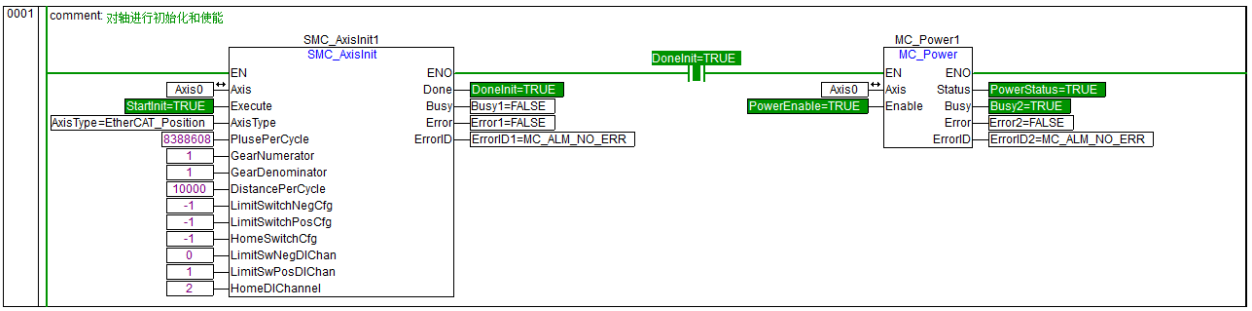
主要变量定义

名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
PowerEnable	BOOL	FALSE	开始使能时变为 TRUE
PowerStatus	BOOL	FALSE	使能成功时变为 TRUE
FwMoveJog	BOOL	FALSE	启动正向指令时为 TRUE
BwMoveJog	BOOL	FALSE	启动负向指令时为 TRUE
BusyMoveJog	BOOL	FALSE	指令投送完成变为 TRUE
ComMoveJog	BOOL	FALSE	指令被打断时变为 TRUE
ErrorMoveJog	BOOL	FALSE	指令出错时变为 TRUE
StartReset	BOOL	FALSE	开始复位时变为 TRUE

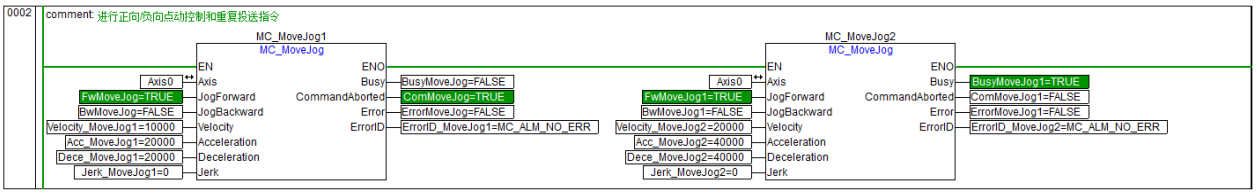
■ 梯形图程序

利用梯形图组建程序，输入参数 Velocity 为 10000，Acceleration 和 Deceleration 为 20000，Jerk 为 0，正向上升沿触发控制轴进行点动运动，示例程序如下：

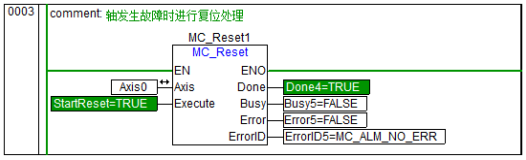
（1） 设置轴类型为 EtherCAT_Position。



（2） 待轴使能完成之后可以在功能块的输入变量中输入指定的运动参数，进行速度运动指令的投送和重复指令的投送，可以看出当重复投送时，后一个功能块会打断前一个功能块。



（3） 当轴出现错误时，可以调用 MC_Reset 功能块进行复位处理。



■ ST 语言程序

（1） 对轴进行初始化设置和使能

```
(*轴进行初始化*)

SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);

(*对轴进行使能*)

IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

（2） 高电平触发启动正负点动运动，控制轴进行运动。

```
(*启动速度控制指令*)

MC_MoveJog1(
JogForward := FwMoveJog,
JogBackward := BwMoveJog,
Velocity := 10000,
Acceleration := 20000,
Deceleration := 20000,
Jerk := 0,
Axis := Axis0,
Busy=>BusyMoveJog,
CommandAborted=>ComMoveJog,
Error=>ErrorMoveJog,
ErrorID=>ErrorID_MoveJog1);
```

（3） 重复触发高电平，控制轴进行运动。

```
(*重复启动速度控制指令*)

MC_MoveJog2(
JogForward := FwMoveJog1,
JogBackward := BwMoveJog1,
Velocity := 20000,
Acceleration := 40000,
Deceleration := 40000,
Jerk := 0,
Axis := Axis0,
Busy=>BusyMoveJog1,
CommandAborted=>ComMoveJog1,
Error=>ErrorMoveJog1,
ErrorID=>ErrorID_MoveJog2);
```

（4） 出错时可以进行复位处理。

```
MC_Reset1(
Execute := Exe5,
Axis := Axis0,
Done=>Done4,
Busy=>Busy5,
Error=>Error5,
```

```
ErrorID=>ErrorID5);
```

5.6.4 使用注意事项

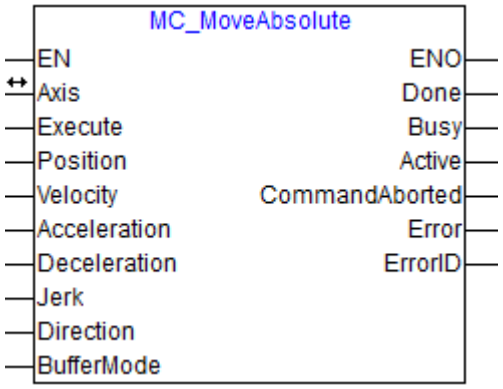
- 输入变量中加速度和减速度不允许输入 0 值。
- 当正向使能和负向使能同时为 TRUE 时，正向使能优先，因此在使用之前需要确定好运动方向。

5.6.5 参见

- 若有错误发生，参见附录功能块错误码。
- 有关初始化和使能，参见初始化和使能章节。

5.7 MC_MoveAbsolute（绝对定位）

单轴绝对定位指令，控制轴运动至指定的绝对位置。



5.7.1 参数

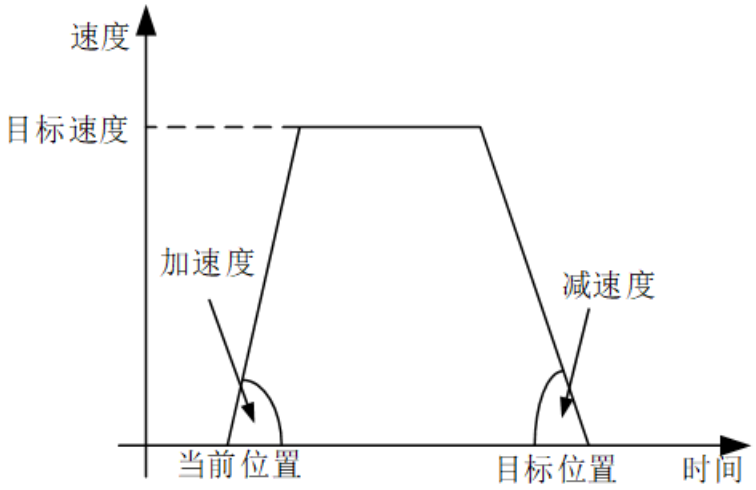
参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Position	目标位置（绝对）	否	-	正值/负值/0	LREAL
	Velocity	目标速度	否	-	非负	LREAL
	Acceleration	加速度	否	-	正值	LREAL

	Deceleration	减速度	否	-	正值	LREAL
	Jerk	0：梯形加减速 正值：S 形加减速加加速度	是	0	0/正值	LREAL
	Direction	运动方向，仅循环模式下有效。 0: 正向 1: 最短距离 2: 负向 3: 当前运动方向 4: 未指定方向，在当前循环行程范围内自动计算方向	是	4	0: mcPositiveDirection 1: mcShortestWay 2: mcNegativeDirection 3: mcCurrentDirection 4: mcNoDirection	MC_DIRECTION
	BufferMode	缓冲模式 0：打断 1：缓存 2：以相邻指令间的低速融合 3：以上一条指令的速度融合 4：以下一条指令的速度融合 5：以相邻指令间的高速融合	是	0	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE
OUTPUT	Done	定位运动完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为 TRUE	是	FALSE	TRUE/FALSE	BOOL
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.7.2 功能

本指令可以按照设置的速度，加减速度，加加速度等参数进行绝对位置运动，对该指令的详细介绍如下：

- 本指令上升沿触发有效，在 Execute（启动）输入的上升沿时，输入变量中指定 Position(绝对位置)，Velocity（目标速度）、Acceleration（加速度）、Deceleration（减速度）和 Jerk（跃度），开始绝对定位动作，执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令，绝对定位的动作示例如下图所示。



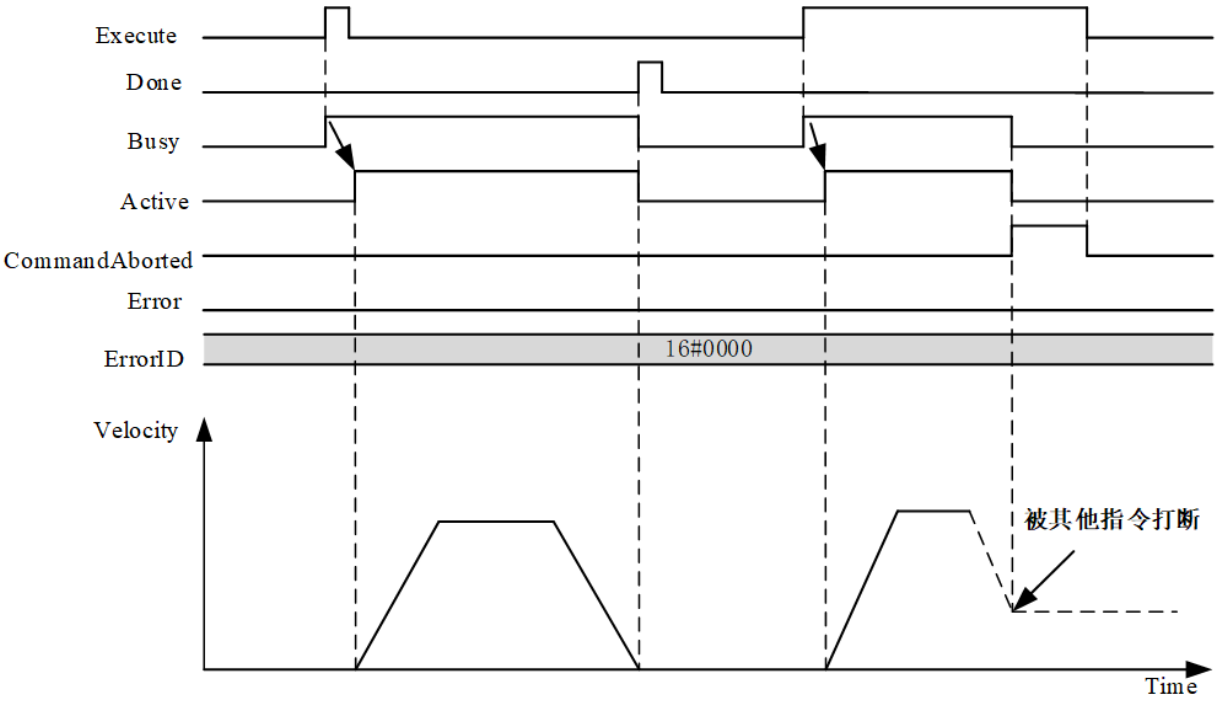
- Direction（方向选择）介绍
  - Direction（方向选择）为 mcPositiveDirection（正向）时，运动方向为正向。
  - Direction（方向选择）为 mcShortestWay（最短距离）时，运动方向会根据算法计算的正向运动到目标位置的距离和负向运动到目标位置的距离来计算最短路程，若是相同时则方向为正向。
  - Direction（方向选择）为 mcNegativeDirection（负向）时，运动方向为负向。
  - Direction（方向选择）设为 mcCurrentDirection（当前运动方向）时，指令会沿着前一动作的指令方向进行动作，若前一动作的方向为正向则当前运动方向为正向，否则为负向。
  - Direction（方向选择）设为 mcNoDirection（未指定运动方向）时，指令会根据当前位置和目标位置的值来确定运动方向。当前位置小于目标位置则运动方向为正向，当前位置大于目标位置则运动方向为负向。
- BufferMode（缓存模式）

绝对定位指令支持 BufferMode (缓冲模式选择)，BufferMode 支持 0：Aborting，1：Buffered，2：BlendingLow、3:BlendingPrevious、4:BlendingNext 以及 5:BlendingHigh。

投送本指令时，若 BufferMode 为 0：Aborting，则打断当前正在执行的指令。若 BufferMode 为 1：Buffered，从当前执行中的指令正常结束的周期起，已缓存的本指令自动启动。若 BufferMode 为 2：BlendingLow，则以相邻指令间的低速融合。若 BufferMode 为 3: BlendingPrevious，则以上一条指令的速度融合。若 BufferMode 为 4: BlendingNext，则以下一条指令的速度融合。若 BufferMode 为 5: BlendingHigh，则以相邻指令间的高速融合。

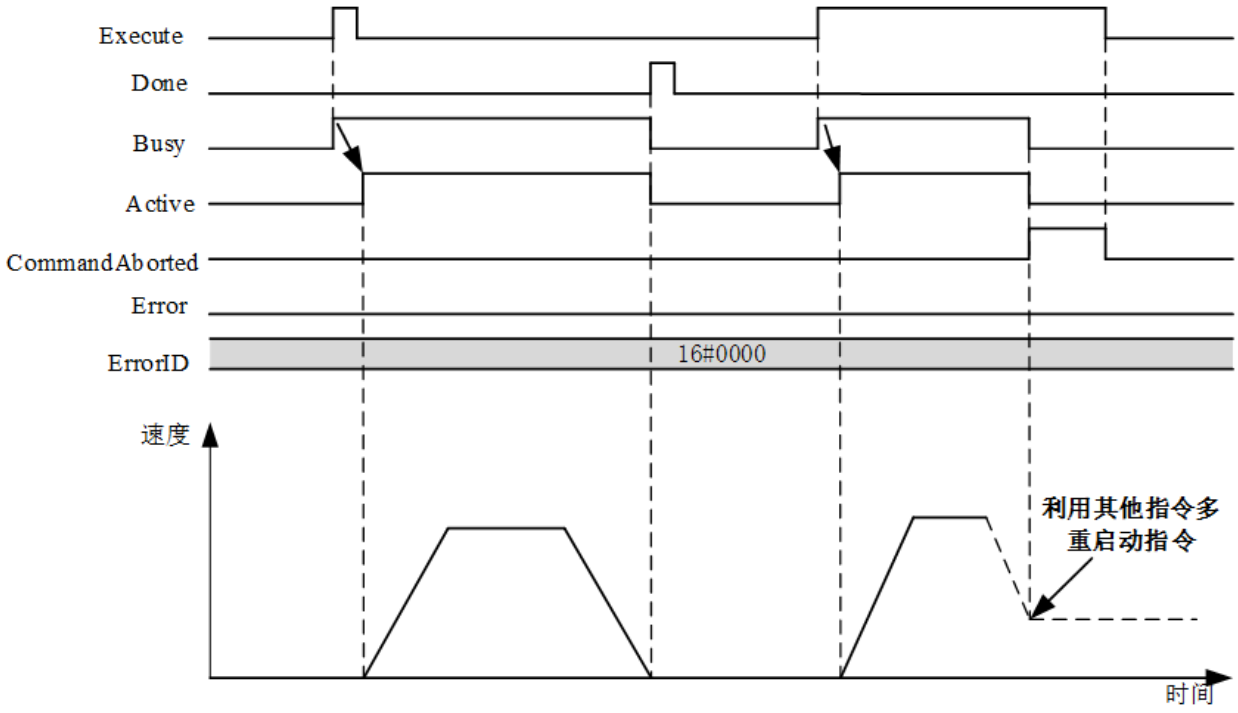
缓冲模式 BufferMode 详情请参考附录章节的缓冲模式说明。

- 重复启动运动指令
  - 本指令重复启动时与上条指令的衔接方式由 BufferMode（缓存模式选择）指定，可选择打断、缓存和融合。
  - 将 BufferMode（缓存模式选择）设为 Aborting(打断)时，再次将 Execute 设为 TRUE 重启指令，即可变更本指令的动作。可变更指令的 Position（目标位置）、Velocity（目标速度）、Acceleration（加速度）、Deceleration（减速度）、Jerk（加加速度）。示例动作如下图所示。



- 功能块时序图

（1） 指令执行时的正确时序如下：



（2） 发生异常时的时序图

在执行本指令中发生异常时，Error（错误）变为 FALSE，轴停止动作。

可以通过查看 ErrorID（错误代码）的输出值，了解发生异常的原因。

5.7.3 示例

组建 MC_MoveAbsolute 示例程序进行绝对定位指令运行。

主要变量定义

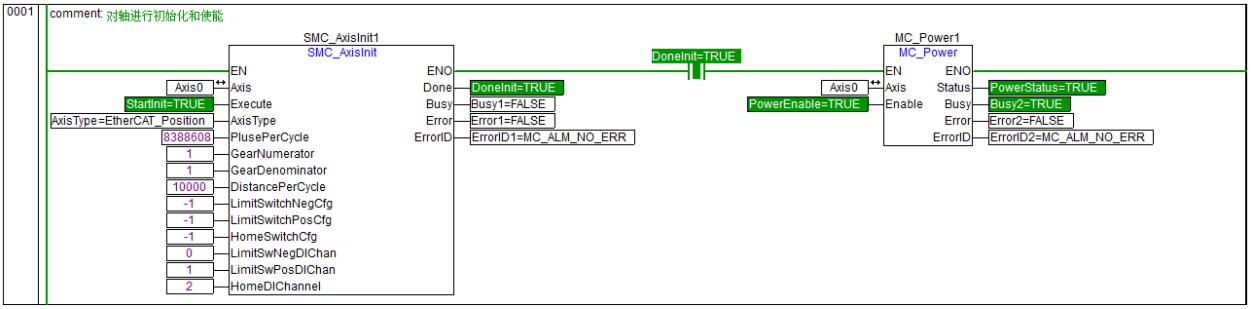
名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
PowerEnable	BOOL	FALSE	开始使能时变为 TRUE
PowerStatus	BOOL	FALSE	使能成功时变为 TRUE
StartMoveAbs	BOOL	FALSE	启动绝对定位指令时为 TRUE
BusyMoveAbs	BOOL	FALSE	绝对定位指令投送完成变为 TRUE
ActiveMoveAbs	BOOL	FALSE	指令运行时变为 TRUE
DoneMoveAbs	BOOL	FALSE	绝对定位指令完成时变为 TRUE

ComMoveAbs	BOOL	FALSE	指令被打断时变为 TRUE
ErrorMoveAbs	BOOL	FALSE	指令出错时变为 TRUE
StartReset	BOOL	FALSE	开始复位时变为 TRUE

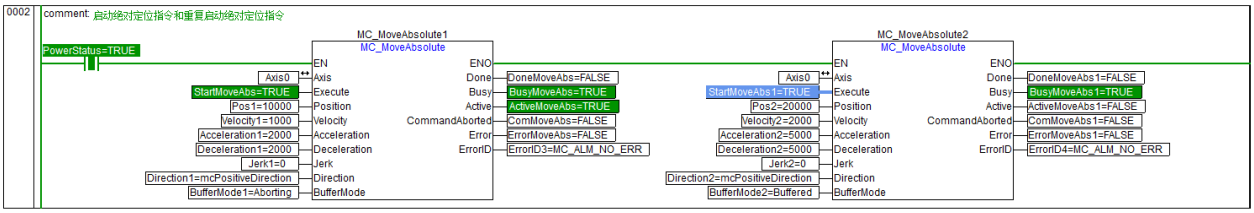
■ 梯形图程序

利用梯形图组建程序，输入参数 Position 为 10000，Velocity 为 1000，Acceleration 和 Deceleration 为 2000，Jerk 为 0，Direction 为正向运行，BufferMode 参数为 Aborting(打断)，上升沿触发控制轴进行运动，示例程序如下：

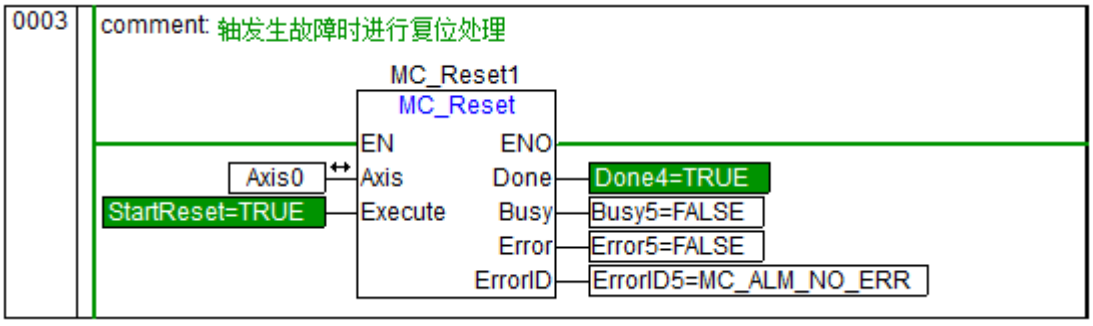
（1） 首先对轴进行初始化，对轴的类型和各种参数进行初始化设置，待轴初始化完成之后，对轴进行使能，进入伺服使能的 ON 状态时，轴可以实现运动控制。



（2） 待轴使能完成之后可以在功能块的输入变量中输入指定的运动参数，进行绝对定位运动指令的投送和重复指令的投送。



（3） 当轴出现错误时，可以调用 MC_Reset 功能块进行复位处理。



■ ST 语言程序

（1） 对轴进行初始化设置和使能

```
(*轴进行初始化*)
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=>Done1,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
(*对轴进行使能*)
IF DoneInit THEN
  MC_Power1(
  Enable := PowerEnable,
  Axis := Axis0,
  Status=> PowerStatus,
  Busy=>Busy2,
  Error=>Error2,
  ErrorID=>ErrorID2);
END_IF
```

（2） 上升沿触发 MC_MoveAbsolute 指令或重复执行本指令，控制轴运动。

```
IF PowerStatus THEN
(*启动绝对定位指令*)
MC_MoveAbsolute1(
Execute := StartMoveAbs,
```

```
Position := 10000,
Velocity := 1000,
Acceleration := 2000,
Deceleration := 2000,
Jerk := 0,
Direction := 4,
BufferMode := 0,
Axis := Axis0,
Done=> DoneMoveAbs,
Busy=> BusyMoveAbs,
Active=> ActiveMoveAbs,
CommandAborted=> ComMoveAbs,
Error=> ErrorMoveAbs,
ErrorID=>ErrorID3);
(*重复启动绝对定位指令*)
MC_MoveAbsolute2(
Execute := StartMoveAbs1,
Position := 50000,
Velocity := 2000,
Acceleration := 5000,
Deceleration := 5000,
Jerk := 0,
Direction := 4,
BufferMode :=0,
Axis := Axis0,
Done=> DoneMoveAbs1,
Busy=> BusyMoveAbs1,
Active=> ActiveMoveAbs1,
CommandAborted=> ComMoveAbs1,
Error=> ErrorMoveAbs1,
ErrorID=>ErrorID4);
END_IF
```

（3） 出现错误时，调用 MC_Reset 功能块对轴进行复位处理。

```
IF ErrorMoveAbs THEN
 (*出错时可以进行复位处理*)
 MC_Reset1(
```

```
  Execute := StartReset,
  Axis := Axis0,
  Done=>Done4,
  Busy=>Busy5,
  Error=>Error5,
  ErrorID=>ErrorID5);
END_IF
```

5.7.4 使用注意事项

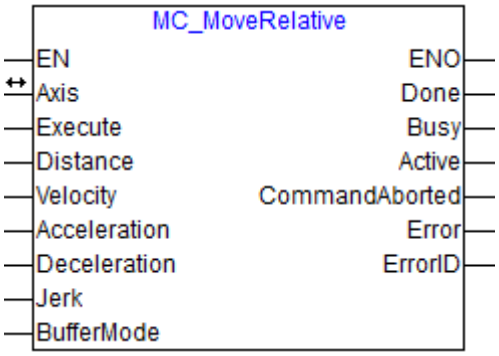
- 在使用绝对定位指令时，目标位置的输入应该在正限位和负限位之间。
- Direction（方向选择）设为 mcCurrentDirection（当前运动方向）时，指令会沿着前一动作的指令方向进行动作，因此应该在使用之前根据上一条指令确认好运动方向。

5.7.5 参见

- 若发生故障参见附录功能块的错误代码说明。
- 关于初始化和使能参见初始化和使能章节介绍。
- BufferMode (缓冲模式选择)参见附录缓冲模式说明。

5.8 MC_MoveRelative（相对定位）

单轴相对定位指令，控制轴以增量方式运动至指定位置。



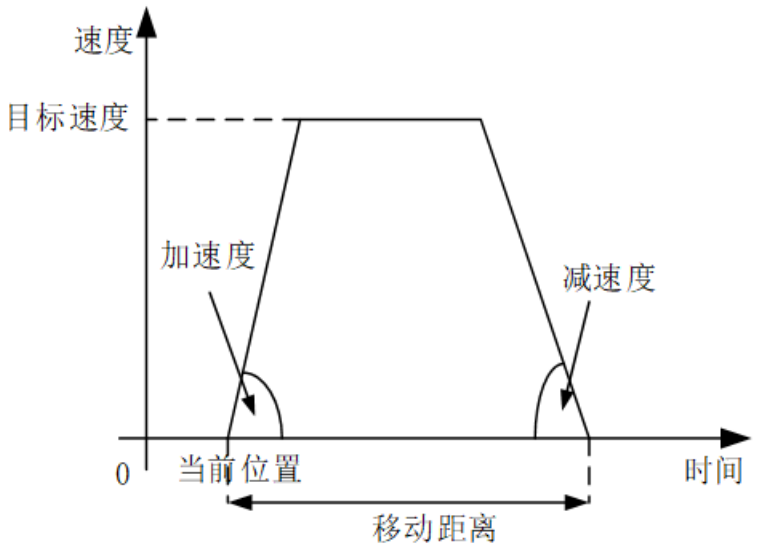
5.8.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Distance	目标位置（增量）	否	-	非 0 值	LREAL
	Velocity	目标速度	否	-	非负	LREAL
	Acceleration	加速度	否	-	正值	LREAL
	Deceleration	减速度	否	-	正值	LREAL
	Jerk	0：梯形加减速 正值：S 形加减速加加速度	是	0	0/正值	LREAL
	BufferMode	缓冲模式 0：打断 1：缓存 2：以相邻指令间的低速融合 3：以上一条指令的速度融合 4：以下一条指令的速度融合 5：以相邻指令间的高速融合	是	0	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE
OUTPUT	Done	定位运动完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为 TRUE	是	FALSE	TRUE/FALSE	BOOL
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.8.2 功能

本指令可以按照设置的速度，加减速度，加加速度等参数进行相对定位运动，控制轴以增量方式运动至指定位置，对该指令的详细介绍如下：

- 本指令上升沿触发有效，在 Execute（启动）输入的上升沿时，输入变量中指定 Distance（移动距离），Velocity（目标速度）、Acceleration（加速度）、Deceleration（减速度）和 Jerk（跃度），开始绝对定位动作，执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令，绝对定位的动作示例如下图所示。



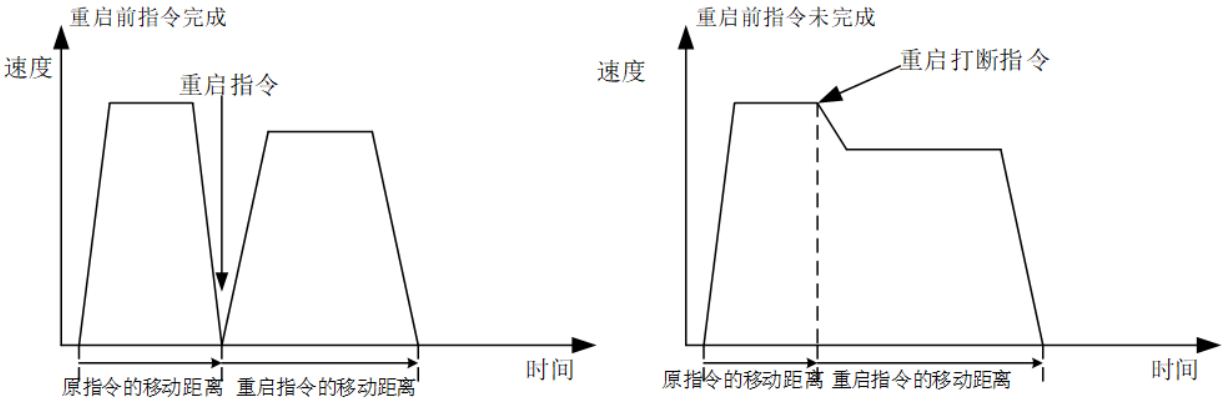
- 本指令中 Distance（移动距离）设为正值时指令就正向转动，设为负指时指令就负向转动。
- BufferMode（缓存模式）

绝对定位指令支持 BufferMode（缓冲模式选择），BufferMode 支持 0：Aborting ，1：Buffered，2：BlendingLow、3: BlendingPrevious、4: BlendingNext 以及 5: BlendingHigh。

投送本指令时，若 BufferMode 为 0：Aborting，则打断当前正在执行的指令。若 BufferMode 为 1：Buffered，从当前执行中的指令正常结束的周期起，已缓存的本指令自动启动。若 BufferMode 为 2：BlendingLow，则以相邻指令间的低速融合。若 BufferMode 为 3: BlendingPrevious，则以上一条指令的速度融合。若 BufferMode 为 4: BlendingNext，则以下一条指令的速度融合。若 BufferMode 为 5: BlendingHigh，则以相邻指令间的高速融合。

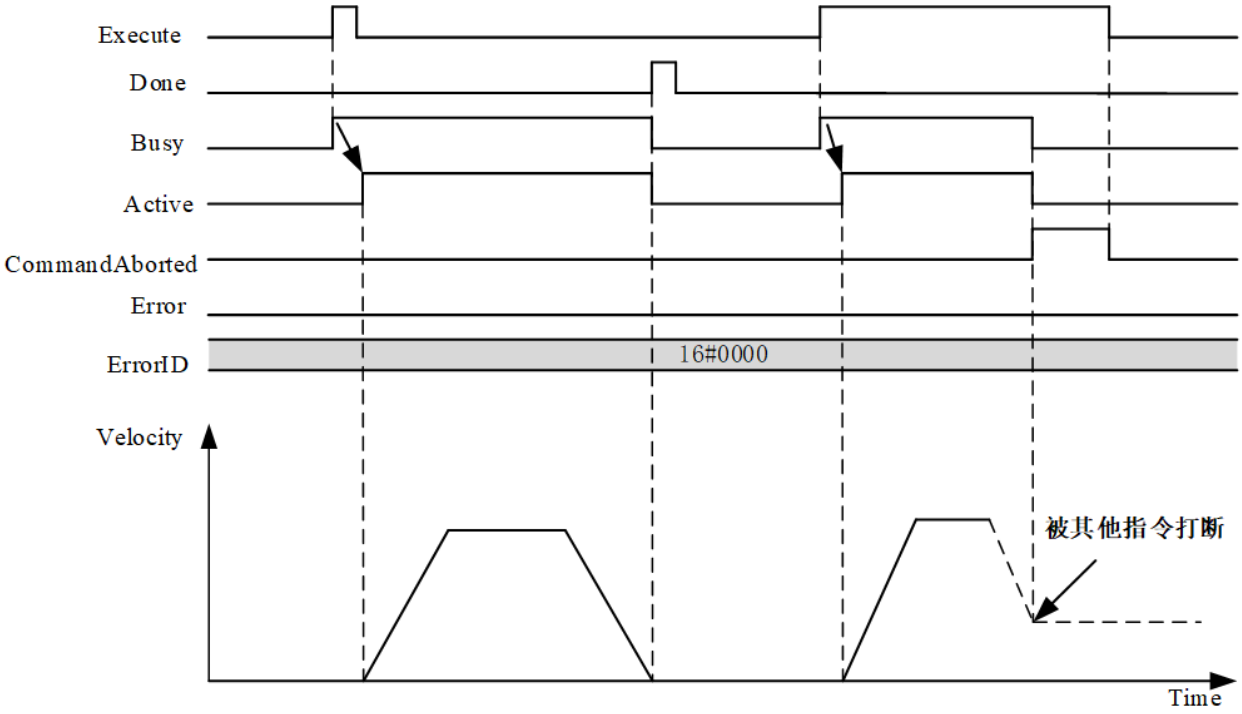
缓冲模式 BufferMode 详情请参考附录章节的缓冲模式说明。

- 重复启动运动指令
  - 本指令重复启动时令与上条指令的衔接方式由 BufferMode（缓存模式选择）指定，可选择打断、缓存和融合。
  - 将 BufferMode（缓存模式选择）设为 Aborting(打断)时，再次将 Execute 设为 TRUE 重启指令，即可变更本指令的动作。可变更指令的 Distance（目标位置）、Velocity（目标速度）、Acceleration（加速度）、Deceleration（减速度）、Jerk（加加速度）。示例动作如下图所示。



- 功能块时序图

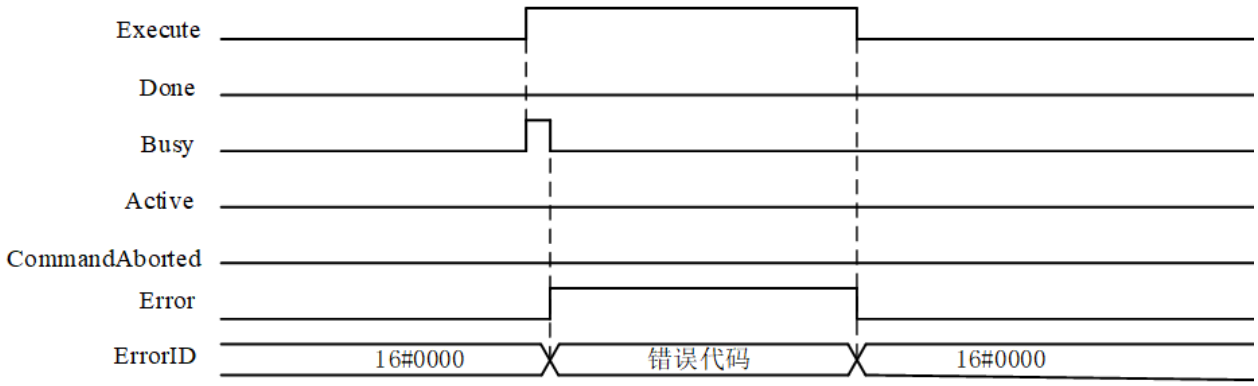
（1） 指令执行时的正确时序如下：



（2） 发生异常时的时序图

在执行本指令中发生异常时，Error（错误）变为 FALSE，轴停止动作。

可以通过查看 ErrorID（错误代码）的输出值，了解发生异常的原因。



5.8.3 示例

组建 MC_MoveRelative 示例程序进行相对定位运行。

主要变量定义

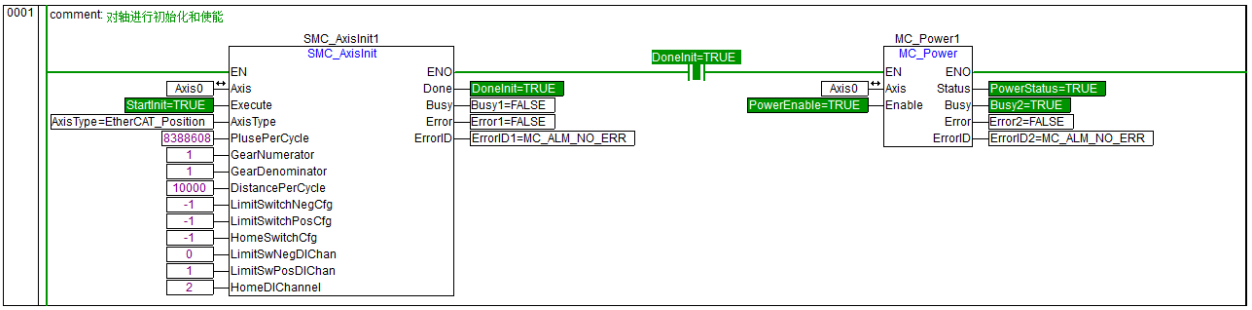
名称	数据类型	初始值	注释说明

Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
PowerEnable	BOOL	FALSE	开始使能时变为 TRUE
PowerStatus	BOOL	FALSE	使能成功时变为 TRUE
StartMoveRel	BOOL	FALSE	启动相对定位指令时为 TRUE
BusyMoveRel	BOOL	FALSE	相对定位指令投送完成变为 TRUE
ActiveMoveRel	BOOL	FALSE	指令运行时变为 TRUE
DoneMoveRel	BOOL	FALSE	相对定位指令完成时变为 TRUE
ComMoveRel	BOOL	FALSE	指令被打断时变为 TRUE
ErrorMoveRel	BOOL	FALSE	指令出错时变为 TRUE
StartReset	BOOL	FALSE	开始复位时变为 TRUE

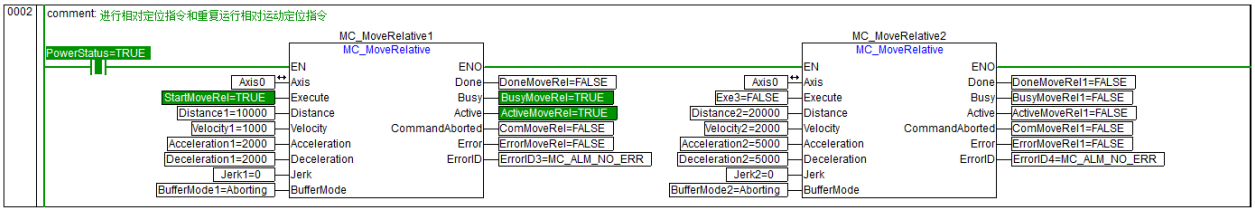
■ 梯形图程序

利用梯形图组建程序，输入参数 Distance 为 10000，Velocity 为 1000，Acceleration 和 Deceleration 为 2000，Jerk 为 0，Direction 为正向运行，BufferMode 参数为 Aborting(打断)，上升沿触发控制轴进行运动，示例程序如下：

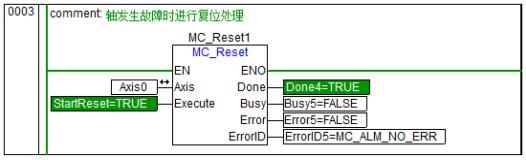
（1） 首先对轴进行初始化，对轴的类型和各种参数进行初始化设置，待轴初始化完成之后，对轴进行使能，进入伺服使能的 ON 状态时，轴可以实现运动控制。



（2） 待轴使能完成之后可以在功能块的输入变量中输入指定的运动参数，进行相对运动指令的投送和重复指令的投送。



（3） 当轴出现错误时，可以调用 MC_Reset 功能块进行复位处理。



■ ST 语言程序

（1） 对轴进行初始化设置和使能

```
(*轴进行初始化*)

SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);

(*对轴进行使能*)

IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

（2） 上升沿触发相对运动指令

```
(*启动相对运动指令*)
IF PowerStatus THEN
 MC_MoveRelative1(
 Execute := StartMoveRel,
 Distance := 10000,
 Velocity := 1000,
 Acceleration := 2000,
 Deceleration := 2000,
 Jerk := 0,
 BufferMode := 0,
 Axis := Axis0,
 Done=> DoneMoveRel,
 Busy=> BusyMoveRel,
 Active=> ActiveMoveRel,
 CommandAborted=> ComMoveRel,
 Error=> ErrorMoveRel,
 ErrorID=>ErrorID3);
```

（3） 重复启动相对运动指令

```
(*重复启动运动指令*)
 MC_MoveRelative2(
 Execute := StartMoveRel1,
 Distance := 20000,
 Velocity := 2000,
 Acceleration := 5000,
 Deceleration := 5000,
 Jerk := 0,
 BufferMode := 0,
 Axis := Axis0,
 Done=> DoneMoveRel1,
 Busy=> BusyMoveRel1,
 Active=> ActiveMoveRel1,
 CommandAborted=> ComMoveRel1,
 Error=> ErrorMoveRel1,
 ErrorID=>ErrorID4);
END_IF
```

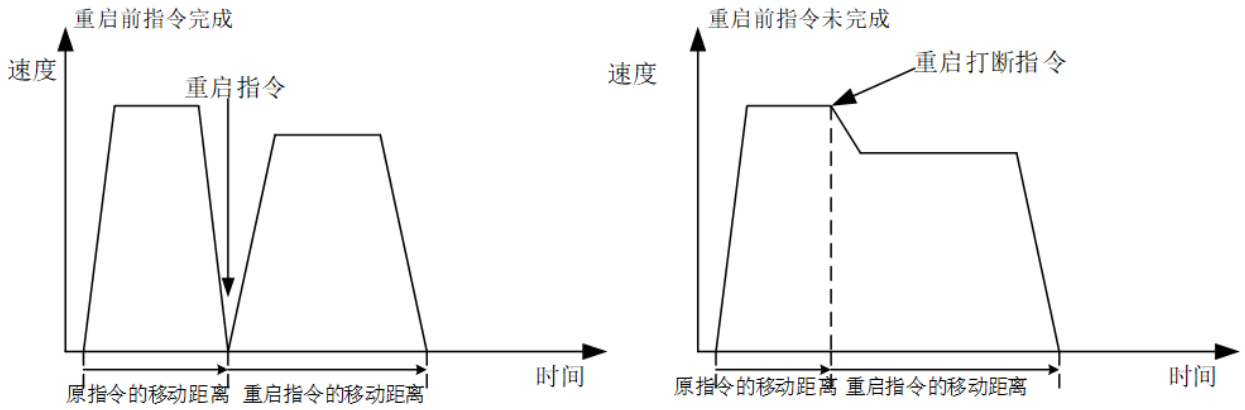
（4） 出现错误时进行复位处理

```
(*出错时可以进行复位处理*)
MC_Reset1(
Execute := StartReset,
Axis := Axis0,
```

```
Done=>Done4,
Busy=>Busy5,
Error=>Error5,
ErrorID=>ErrorID5);
```

5.8.4 使用注意事项

- 在输入参数 Distance（运动距离）时，不允许输入 0 值，不然就有报错产生。
- 如果在功能块模块重启之前定位完成，重启时的位置为基准的相对位置进行定位。
- 如果在功能块模块重启之前定位未完成，重启功能块进行打断重启时，则以指令运行时刻的位置为基准。

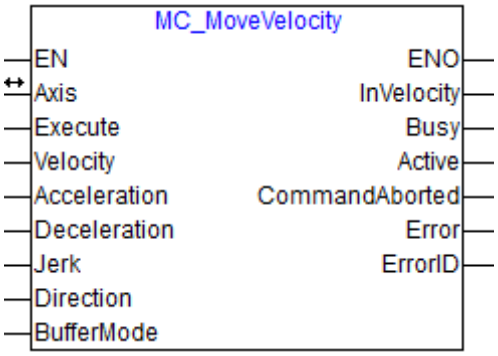


5.8.5 参见

- 若有错误发生，参见附录功能块错误码。
- 有关使用可参见 MC_MoveAbsolute（绝对定位）功能块。
- BufferMode (缓冲模式选择)参见附录缓冲模式说明。

5.9 MC_MoveVelocity（速度控制）

单轴速度控制指令，在位置控制模式下模拟速度控制。



5.9.1 参数

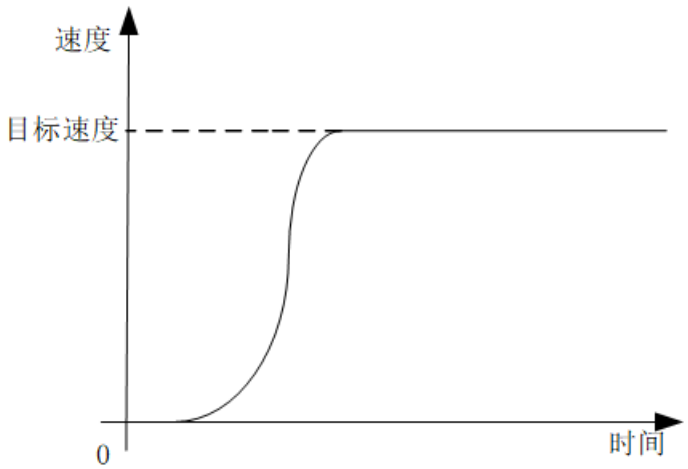
参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Velocity	目标速度	否	-	非负	LREAL
	Acceleration	加速度	否	-	正值	LREAL
	Deceleration	减速度	否	-	正值	LREAL
	Jerk	0：梯形加减速 正值：S 形加减速加加速度	是	0	0/正值	LREAL
	Direction	运动方向，仅循环模式下有效。 0: 正向 2: 负向 3: 当前运动方向	是	0	0: mcPositiveDirection 2: mcNegativeDirection 3: mcCurrentDirection	MC_DIRECTION
	BufferMode	缓冲模式 0：打断 1：缓存 2：以相邻指令间的低速融合 3：以上一条指令的速度融合 4：以下一条指令的速度融合 5：以相邻指令间的高速融合	是	0	0: Aborting 1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE
OUTPUT	InVelocity	目标速度到达	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为	是	FALSE	TRUE/FALSE	BOOL

		TRUE				
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.9.2 功能

本指令可以按照设置的速度，加减速度，加加速度等参数进行模拟速度控制运动，对该指令的详细介绍如下：

- 本指令上升沿触发有效，在 Execute（启动）输入的上升沿时，输入变量中指定 Velocity（目标速度）、Acceleration（加速度）、Deceleration（减速度）和 Jerk（跃度），开始绝对定位动作，执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令，指令动作示例如下。



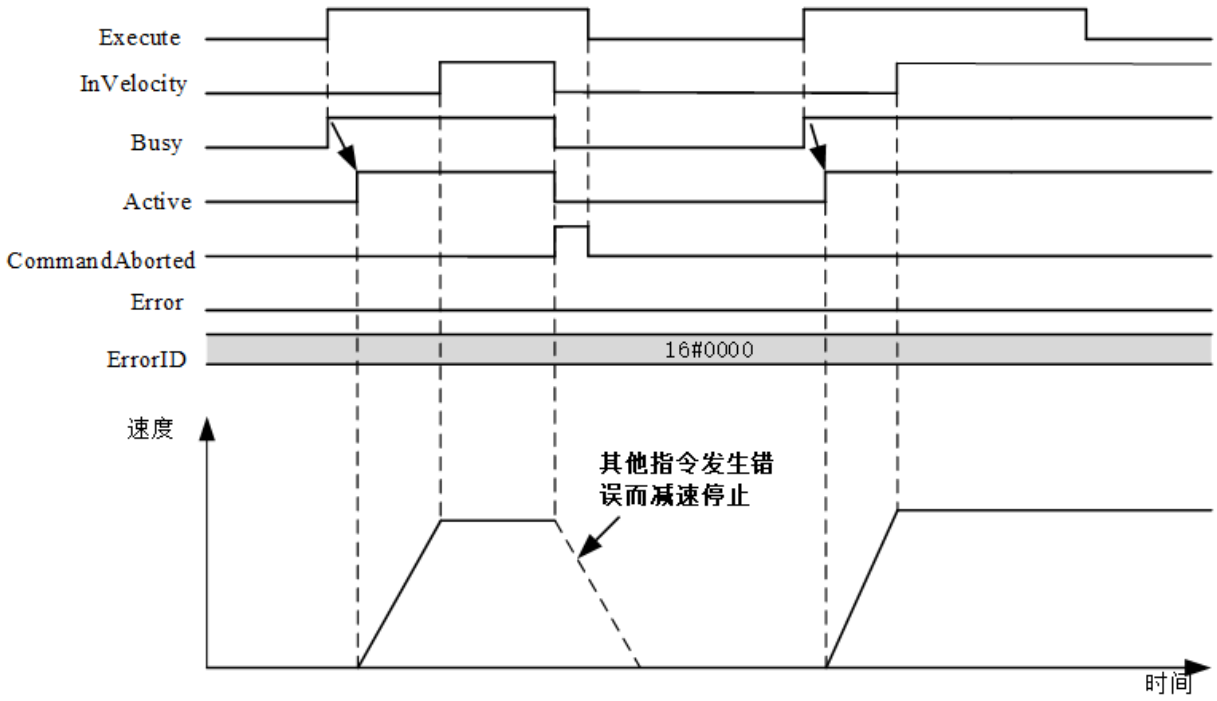
- Direction（方向选择）介绍
  - Direction（方向选择）为 mcPositiveDirection（正向）时，运动方向为正向。
  - Direction（方向选择）为 mcNegativeDirection（负向）时，运动方向为负向。
  - Direction（方向选择）设为 mcCurrentDirection（当前运动方向）时，指令会沿着前一动作的指令方向进行动作，若前一动作的方向为正向则当前运动方向为正向，否则为负向。
- 重复启动运动指令
  - 在定位动作中变更输入变量，将 BufferMode（缓存模式选择）设为 Aborting(打断)时，再次将 Execute 设为 TRUE，即可变更本指令的动作。

- 重复启动指令时，将 BufferMode（缓存模式选择）设为 Aborting(打断)时，可变更指令的 Velocity（目标速度）、Acceleration（加速度）、Deceleration（减速度）和方向等参数。
- 重复启动时的本指令的动作由 BufferMode（缓存模式选择）指定，本指令的重复启动可以选择打断、缓存和融合。

- InVelocity(达到目标速度)是表示对于启动本指令和重启运动指令达到等速的输出。因此，InVelocity(达到目标速度)变为 TRUE 后，即使利用 MC_SetOverride 来变更速度，InVelocity(达到目标速度)也不会变为 FALSE。并且，在 InVelocity(达到目标速度)变为 TRUE 之前已使用 MC_SetOverride 变更速度时，如果已达到变更目标速度，则 InVelocity(达到目标速度)变为 TRUE。

- 功能块时序图

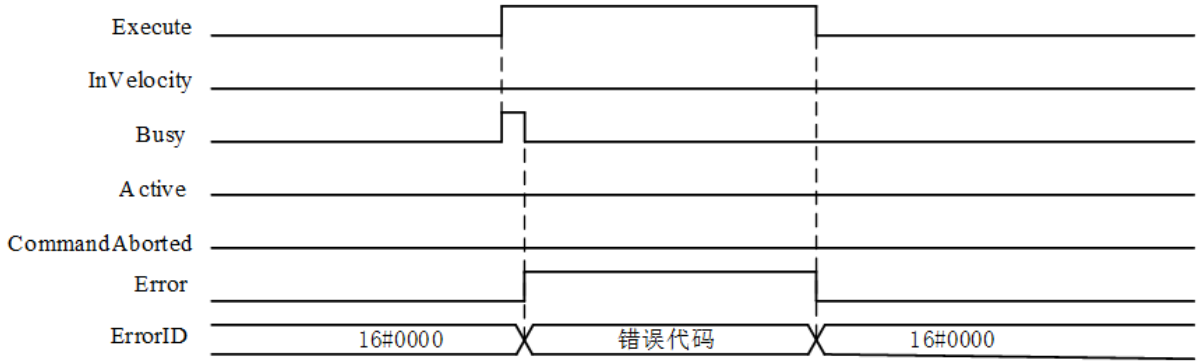
(1) 指令执行时的正确时序如下：



(2) 发生异常时候时序图

在执行本指令中发生异常时，Error（错误）变为 TRUE，轴停止动作。

可查看 ErrorID(错误代码)的输出值，了解发生异常的原因。



5.9.3 示例

组建 MC_MoveVelocity 示例程序进行模拟速度运动指令运行。

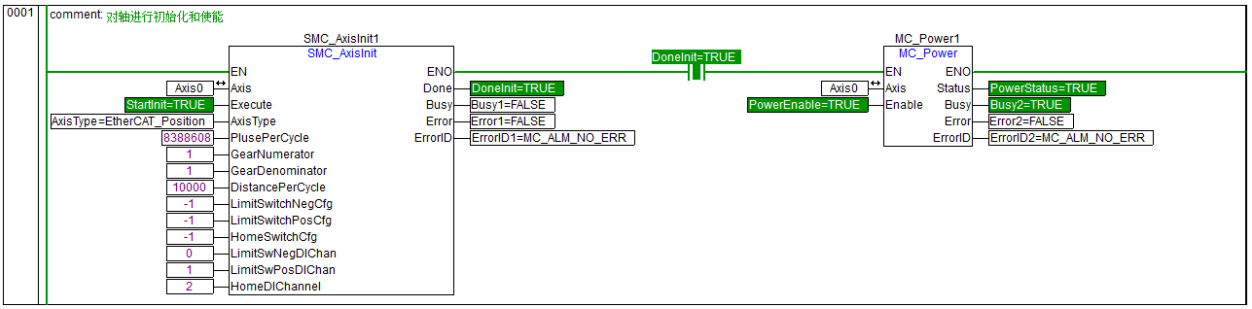
主要变量定义

名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
PowerEnable	BOOL	FALSE	开始使能时变为 TRUE
PowerStatus	BOOL	FALSE	使能成功时变为 TRUE
StartMoveVel	BOOL	FALSE	启动指令时为 TRUE
BusyMoveVel	BOOL	FALSE	指令投送完成变为 TRUE
ActiveMoveVel	BOOL	FALSE	指令运行时变为 TRUE
InVelMoveVel	BOOL	FALSE	指令速度达到时变为 TRUE
ComMoveVel	BOOL	FALSE	指令被打断时变为 TRUE
ErrorMoveVel	BOOL	FALSE	指令出错时变为 TRUE
StartReset	BOOL	FALSE	开始复位时变为 TRUE

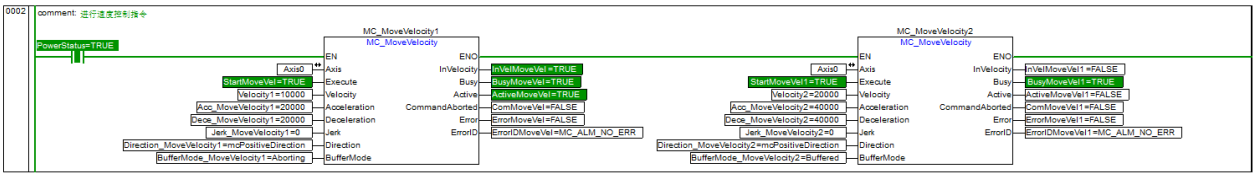
■ 梯形图程序

利用梯形图组建程序，输入参数 Velocity 为 10000，Acceleration 和 Deceleration 为 20000，Jerk 为 0，Direction 为正向运行，BufferMode 参数为 Aborting(打断)，上升沿触发控制轴进行速度运动，示例程序如下：

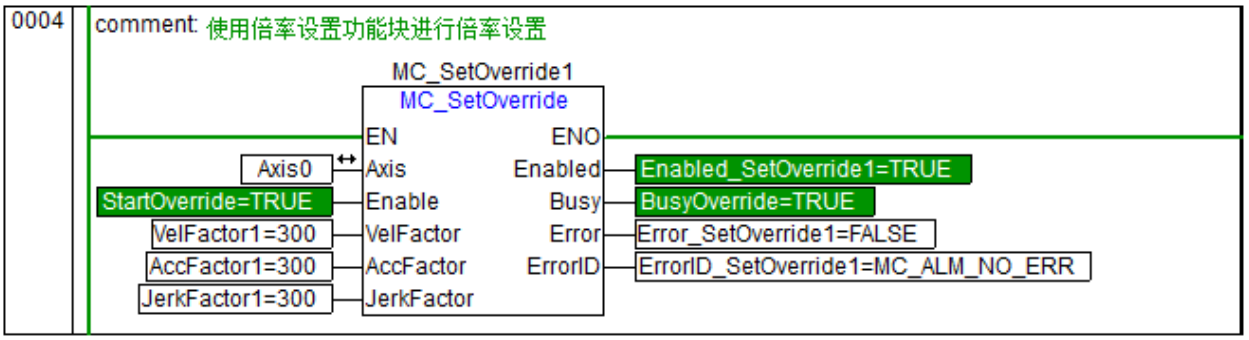
（1） 首先对轴进行初始化，对轴的类型和各种参数进行初始化设置，待轴初始化完成之后，对轴进行使能，进入伺服使能的 ON 状态时，轴可以实现运动控制。



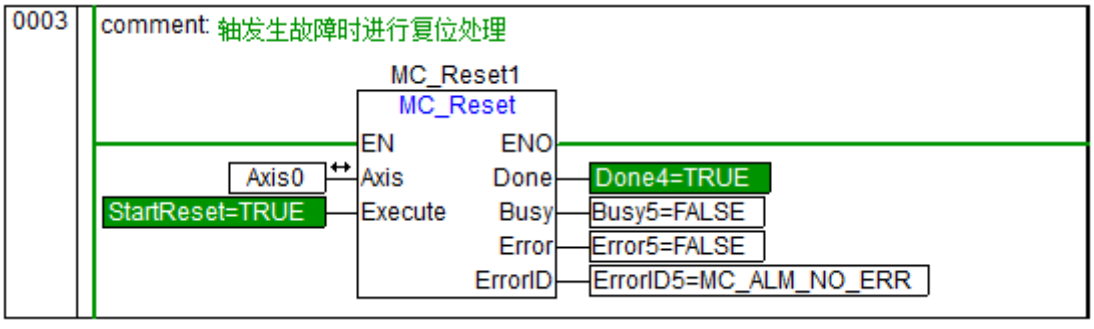
（2） 待轴使能完成之后可以在功能块的输入变量中输入指定的运动参数，进行速度运动指令的投送和重复指令的投送。



（3） 使用 MC_SetOverride 指令进行超调设置，查看 InVelocity 引脚状态变化。



（4） 当轴出现错误时，可以调用 MC_Reset 功能块进行复位处理。



■ ST 语言程序

（1） 对轴进行初始化设置和使能

```
(*轴进行初始化*)
```

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
(*对轴进行使能*)
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

（2） 上升沿触发 MC_MoveAbsolute 指令或重复执行本指令，控制轴运动。

```
IF PowerStatus THEN
(*启动绝对定位指令*)
MC_MoveAbsolute1(
Execute := StartMoveAbs,
Position := 10000,
Velocity := 1000,
Acceleration := 2000,
```

```
Deceleration := 2000,
Jerk := 0,
Direction := 4,
BufferMode := 0,
Axis := Axis0,
Done=> DoneMoveAbs,
Busy=> BusyMoveAbs,
Active=> ActiveMoveAbs,
CommandAborted=> ComMoveAbs,
Error=> ErrorMoveAbs,
ErrorID=>ErrorID3);
(*重复启动绝对定位指令*)
MC_MoveAbsolute2(
Execute := StartMoveAbs1,
Position := 50000,
Velocity := 2000,
Acceleration := 5000,
Deceleration := 5000,
Jerk := 0,
Direction := 4,
BufferMode :=0,
Axis := Axis0,
Done=> DoneMoveAbs1,
Busy=> BusyMoveAbs1,
Active=> ActiveMoveAbs1,
CommandAborted=> ComMoveAbs1,
Error=> ErrorMoveAbs1,
ErrorID=>ErrorID4);
END_IF
```

（3） 出现错误时，调用 MC_Reset 功能块对轴进行复位处理。

```
IF ErrorMoveAbs THEN
 (*出错时可以进行复位处理*)
 MC_Reset1(
 Execute := StartReset,
 Axis := Axis0,
 Done=>Done4,
```

```
  Busy=>Busy5,
  Error=>Error5,
  ErrorID=>ErrorID5);
END_IF
```

5.9.4 使用注意事项

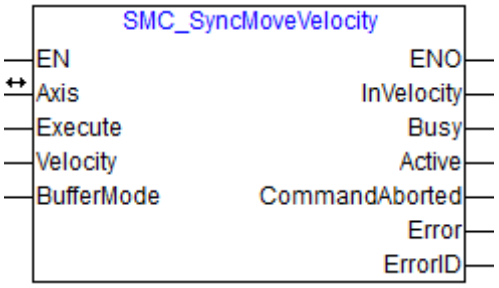
- 在使用绝对定位指令时，目标位置的输入应该在正限位和负限位之间。
- Direction（方向选择）设为 mcCurrentDirection（当前运动方向）时，指令会沿着前一动作的指令方向进行动作，因此应该在使用之前根据上一条指令确认好运动方向。

5.9.5 参见

- 若发生故障参见附录功能块的错误代码说明。
- 关于初始化和使能参见初始化和使能章节介绍。
- BufferMode (缓冲模式选择)参见附录缓冲模式说明。

5.10 SMC_SyncMoveVelocity（周期同步速度控制）

周期同步速度模式下的速度控制



5.10.1 参数

参数说明

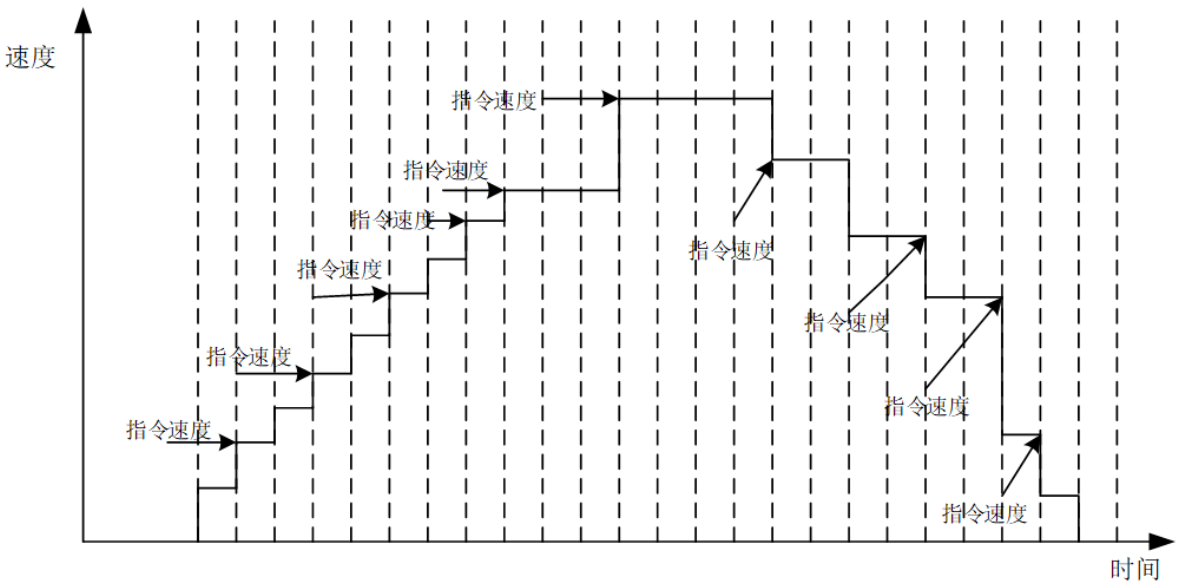
参数类型	名称	描述	能否空	默认值	范围	数据类型

IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Velocity	目标速度	否	-	正值/负值/0	LREAL
	BufferMode	缓冲模式 0：打断 1：缓存	是	0	0: Aborting 1: Buffered	MC_BUFFER_MODE
OUTPUT	InVelocity	目标速度到达	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为 TRUE	是	FALSE	TRUE/FALSE	BOOL
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.10.2 功能

本指令按照任务周期，以周期同步速度模式将用户程序给定的目标速度输出到伺服驱动器，该指令详情介绍如下：

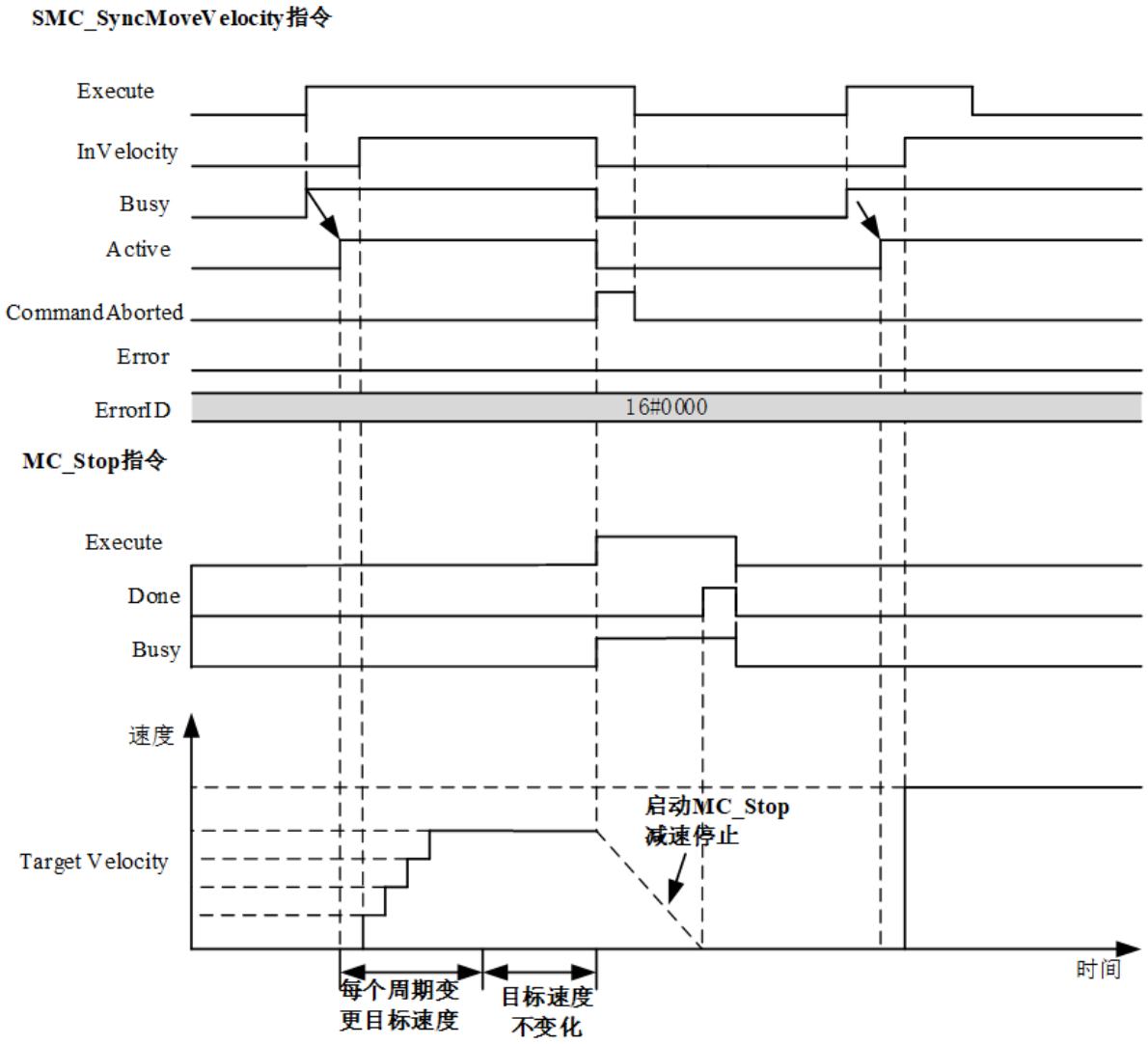
- 在 Execute(上升沿触发)为 TRUE 时，指令触发切换伺服驱动器为速度控制模式，当任务中有本指令时，无论任务的优先级，都在下个周期达到目标速度，指令动作示意图如下所示。



- 使用本指令时，需要在轴的基本设置中映射如下对象数据：控制字(6040 HEX)、状态字(6041 HEX)、控制模式(6060 HEX)、控制模式显示(6061 HEX)、目标速度(60FF HEX)、实际速度(606C HEX)。

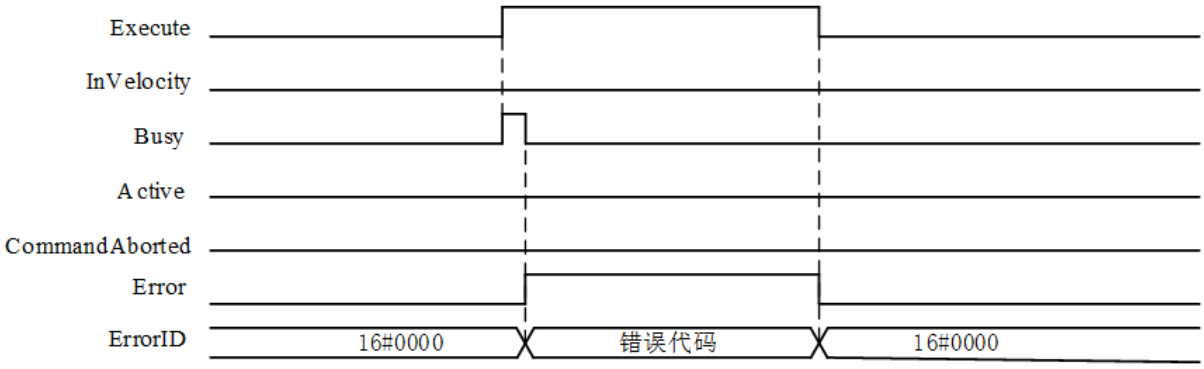
- 输入的目标速度为正值时，沿正方向移动，输入负数时，沿负方向移动。输入速度 0 时，功能块不会运动。
- 本指令支持实时更改目标速度，每个周期用户都可以通过功能块程序修改设定的目标速度，输入的目标速度与上个周期的目标速度不同时，才用新的目标速度，与上一周期的目标速度相同时，以上一周期的目标速度进行运动。
- 本指令支持 BufferMode(缓存模式选择)中的打断和缓存，可以打断其他运动指令或者是被其他运动指令打断，重启本指令时可以选择缓存或者打断，要结束本指令时，使用 MC_Stop 指令进行停止。
- 功能块时序图

  (1) 正常情况时序图



(2) 异常时序图

在执行本指令中发生错误时，Error(错误)变为 TRUE，其他引脚变为 FALSE，轴停止动作，ErrorID(错误代码)输出相关的错误码，发生异常时候的时序图如下所示。



5.10.3 示例

组建 SMC_SyncMoveVelocity 示例程序进行周期同步速度指令运行。

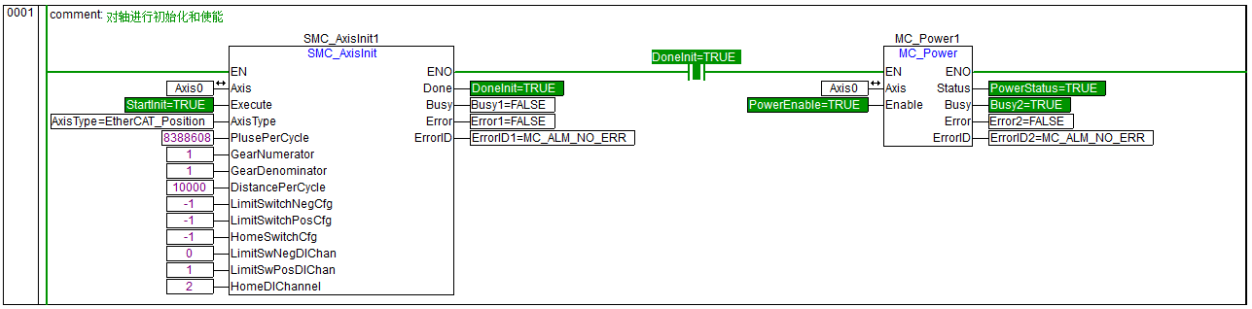
主要变量定义

名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
PowerEnable	BOOL	FALSE	开始使能时变为 TRUE
PowerStatus	BOOL	FALSE	使能成功时变为 TRUE
StartSyncMove	BOOL	FALSE	启动周期同步速度指令时为 TRUE
BusySyncMove	BOOL	FALSE	指令投送完成变为 TRUE
ActiveSyncMove	BOOL	FALSE	指令运行时变为 TRUE
InVelocitySyncMove	BOOL	FALSE	速度达到时变为 TRUE
ComSyncMove	BOOL	FALSE	指令被打断时变为 TRUE
ErrorSyncMove	BOOL	FALSE	指令出错时变为 TRUE
StartReset	BOOL	FALSE	开始复位时变为 TRUE

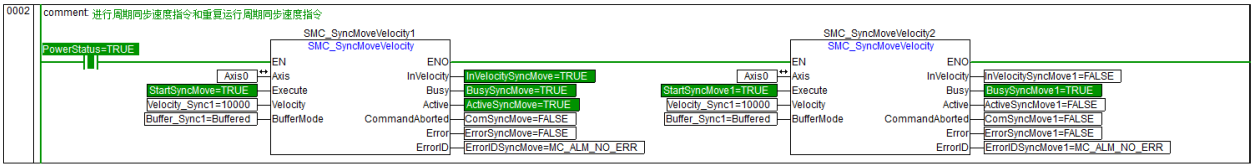
■ 梯形图程序

利用梯形图组建程序，输入参数 Velocity 为 10000，BufferMode 参数为 Aborting(打断)，上升沿触发控制轴进行周期同步速度运动，示例程序如下：

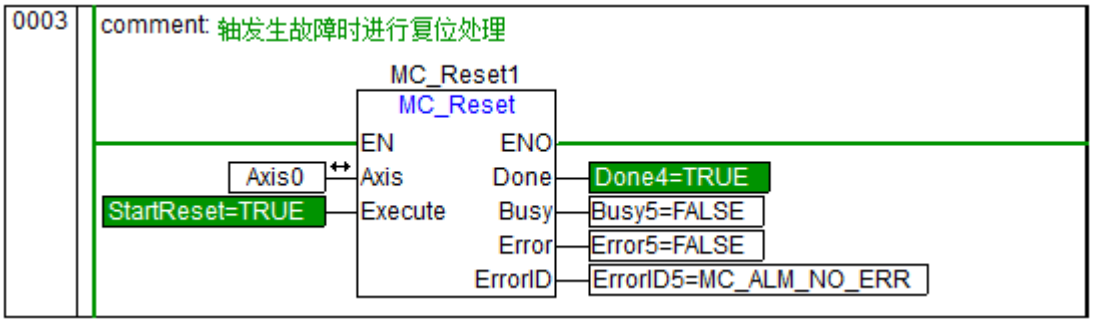
（1） 确保对象参数映射完成之后，首先对轴进行初始化，对轴的类型和各种参数进行初始化设置，待轴初始化完成之后，对轴进行使能，进入伺服使能的 ON 状态时，轴可以实现运动控制。



（2） 待轴使能完成之后可以在功能块的输入变量中输入指定的运动参数，进行同步速度指令的投送和重复指令的投送。



（3） 当轴出现错误时，可以调用 MC_Reset 功能块进行复位处理。



■ ST 语言程序

（1） 对轴进行初始化设置

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
```

```
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

（2） 对轴进行使能

```
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

（3） 启动周期同步速度指令

```
IF PowerStatus THEN
SMC_SyncMoveVelocity1(
Execute := StartSyncMove,
Velocity := 10000,
BufferMode := Buffer_Sync1,
Axis := Axis0,
InVelocity=>InVelocitySyncMove,
Busy=>BusySyncMove,
Active=>ActiveSyncMove,
CommandAborted=>ComSyncMove,
Error=>ErrorSyncMove,
ErrorID=>ErrorIDSyncMove);
```

（4） 重复启动周期同步速度指令

```
   SMC_SyncMoveVelocity2(
   Execute := StartSyncMove1,
   Velocity := 10000,
   BufferMode := Buffer_Sync1,
   Axis := Axis0,
   InVelocity=>InVelocitySyncMove1,
```

```
    Busy=>BusySyncMove1,
    Active=>ActiveSyncMove1,
    CommandAborted=>ComSyncMove1,
    Error=>ErrorSyncMove1,
    ErrorID=>ErrorIDSyncMove1);
END_IF
```

（5） 出错时可以进行复位处理

```
MC_Reset1(
Execute := StartReset,
Axis := Axis0,
Done=>Done4,
Busy=>Busy5,
Error=>Error5,
ErrorID=>ErrorID5);
```

5.10.4 使用注意事项

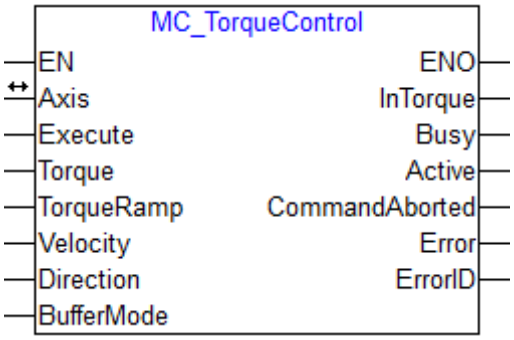
- 本指令不支持虚拟轴和编码器轴，轴类型需是 EtherCAT 总线轴类型。
- 超驰 MC_SetOverride 指令对本指令无效。
- 本指令是速度跳动比较大，因此不同场景使用时需要注意。

5.10.5 参见

- 若有错误发生，参见附录功能块错误码。
- 有关初始化和使能，参见初始化和使能章节。
- BufferMode (缓冲模式选择)参见附录缓冲模式说明。

5.11 MC_TorqueControl（力矩控制）

单轴力矩控制指令，在力矩模式下对轴进行带斜坡的扭矩控制。



5.11.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Torque	目标力矩 单位：1%额定力矩	是	300	0 ~ 1000	LREAL
	TorqueRamp	力矩变化率 单位：1%额定力矩/秒	否	-	正值	LREAL
	Velocity	最大速度	否	-	非负	LREAL
	Direction	运动方向 0: 正向 2: 负向	是	0	0: mcPositiveDirection 2: mcNegativeDirection	MC_DIRECTION
	BufferMode	缓冲模式 0：打断 1：缓存	是	0	0: Aborting 1: Buffered	MC_BUFFER_MODE
OUTPUT	InTorque	目标力矩到达 当设置力矩达到目标力矩且反馈力矩和目标力矩的差值的绝对值在 5%以内时输出有效	是	OFF	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为 TRUE	是	FALSE	TRUE/FALSE	BOOL
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.11.2 功能

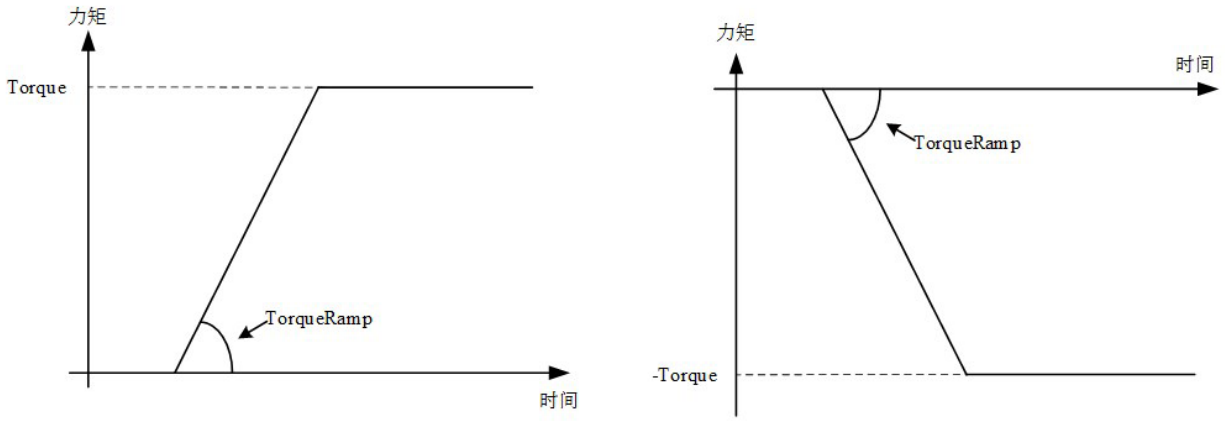
该指令用于实现力矩控制功能，仅支持 EtherCAT 总线伺服轴，不支持虚拟轴。该指令使用伺服驱动器同步力矩模式（CST）实现力矩控制功能，指令运行时将伺服控制模式设置为 CST 模式。

该指令上升沿使能有效，在 Execute 上升沿时，锁存 Torque、TorqueRamp、 Velocity 和 Direction 输入参数，在 Execute 为 TRUE 期间，输入参数修改无效。指令正常运行时，轴处于 Continuous Motion 状态并进行力矩运动，指令输出参数 Busy、Active 为 TRUE，当设置力矩达到目标力矩且反馈力矩和目标力矩的差值的绝对值在 5%以内时 InTorque 引脚输出有效，指令执行时，可使用 MC_Stop 打断，被打断后 CommandAboarted 引脚输出有效。

■ 输入参数说明

（1） **Torque**：目标力矩，单位是 1%额定力矩，设置值小数点后超过第 2 位时，第二位四舍五入，后面的直接舍弃，如设置 Torque 值为 10.26435，实际生效的值是 10.3。驱动器实际力矩受正负力矩限幅值限制，在使用时，注意设置合理的最大正负力矩限制值。

（2） **TorqueRamp**：力矩斜坡，单位是 1%额定扭矩/秒，表示目标扭矩的变化率，指令执行时，指定从当前转矩到输出目标转矩的倾斜度。



TorqueRamp 示意图

（3） **Velocity**：速度限幅参数，用于限制驱动器力矩控制时的最大速度，单位为脉冲单位/s，驱动器速度限幅对象字典为 0x607F。如果 PDO 参数中映射了 0x607F，则在 Execute 上升沿时，指令将参数 Velocity 设置值通过 PDO 写入 0x607F 中；如果 PDO 参数中未映射 0x607F，则在 Execute 上升沿时，指令将参数 Velocity 设置值通过 SDO 写入 0x607F 中。多轴场景对象字典

0x607F 需加偏移 0x800，具体请查阅驱动器相关手册。

在使用该指令进行力矩控制时，需满足以下两种情况才能将 Velocity 用作最大速度限制：

- 伺服可以通过对象字典 0x607F 限制电机的最大速度；
- 伺服对象字典 0x607F 的单位为脉冲单位，而不是转速单位。

力矩控制时的加速度取决于伺服内部控制。

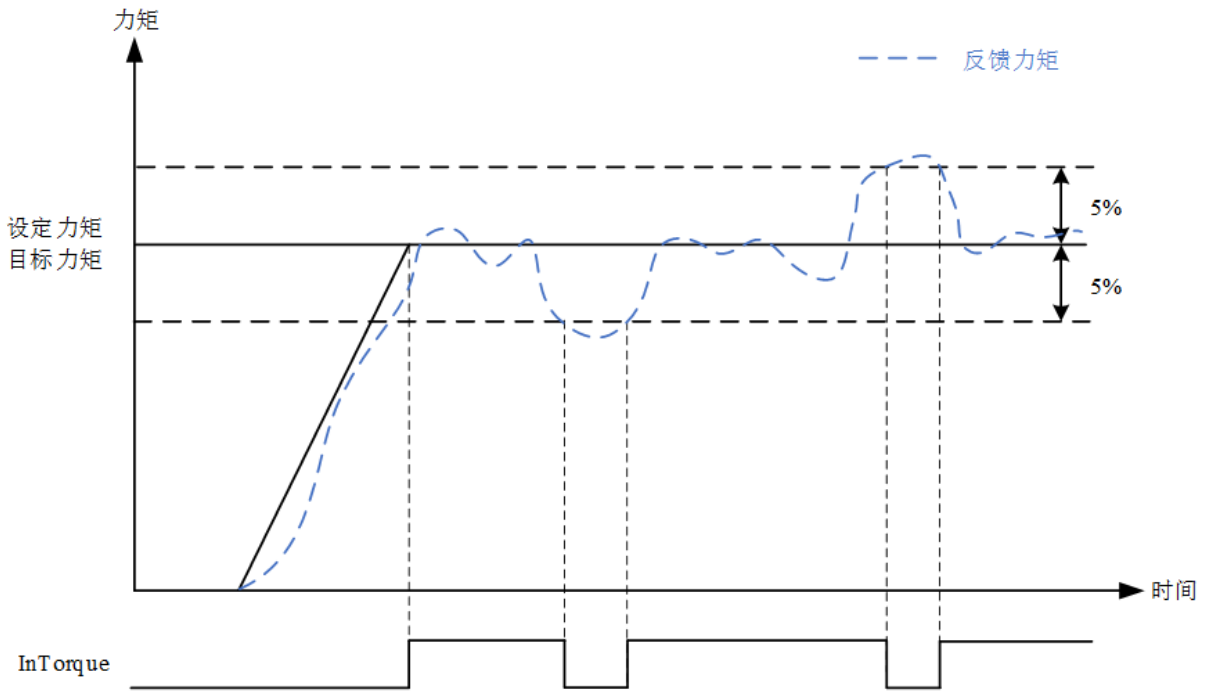
(4) **Direction**：指定目标力矩的输出方向。要沿轴的正方向输出时，指定正方向；要沿轴的负方向输出时，指定负方向。

(5) **BufferMode**：指定前一个轴动作和本次动作的衔接方式。本指令支持以下两种模式：

- 打断：立即中止当前正在执行的指令，切换为本指令。
- 缓存：当前正在执行的指令完成后，已缓存的本指令才能执行。

■ 输出参数说明

InTorque：目标力矩到达标志，当设置力矩达到目标力矩且反馈力矩和目标力矩的差值的绝对值在 5%以内时输出有效。示意图如下：



InTorque 示意图

- 对象字典说明

  力矩控制需要驱动器映射以下对象字典（多轴场景对象字典需加偏移 0x800）：

  - 控制字 Controlword (6040 hex)
  - 状态字 Statusword (6041 hex)
  - 控制模式 Mode of Operation (6060 hex)
  - 控制模式显示 Mode of Operation Display (6061 hex)
  - 目标力矩 Target Torque(6071 hex)
  - 实际力矩 Torque Actual Value (6077 hex)
  - 实际速度 Velocity Actual Value (606C hex)

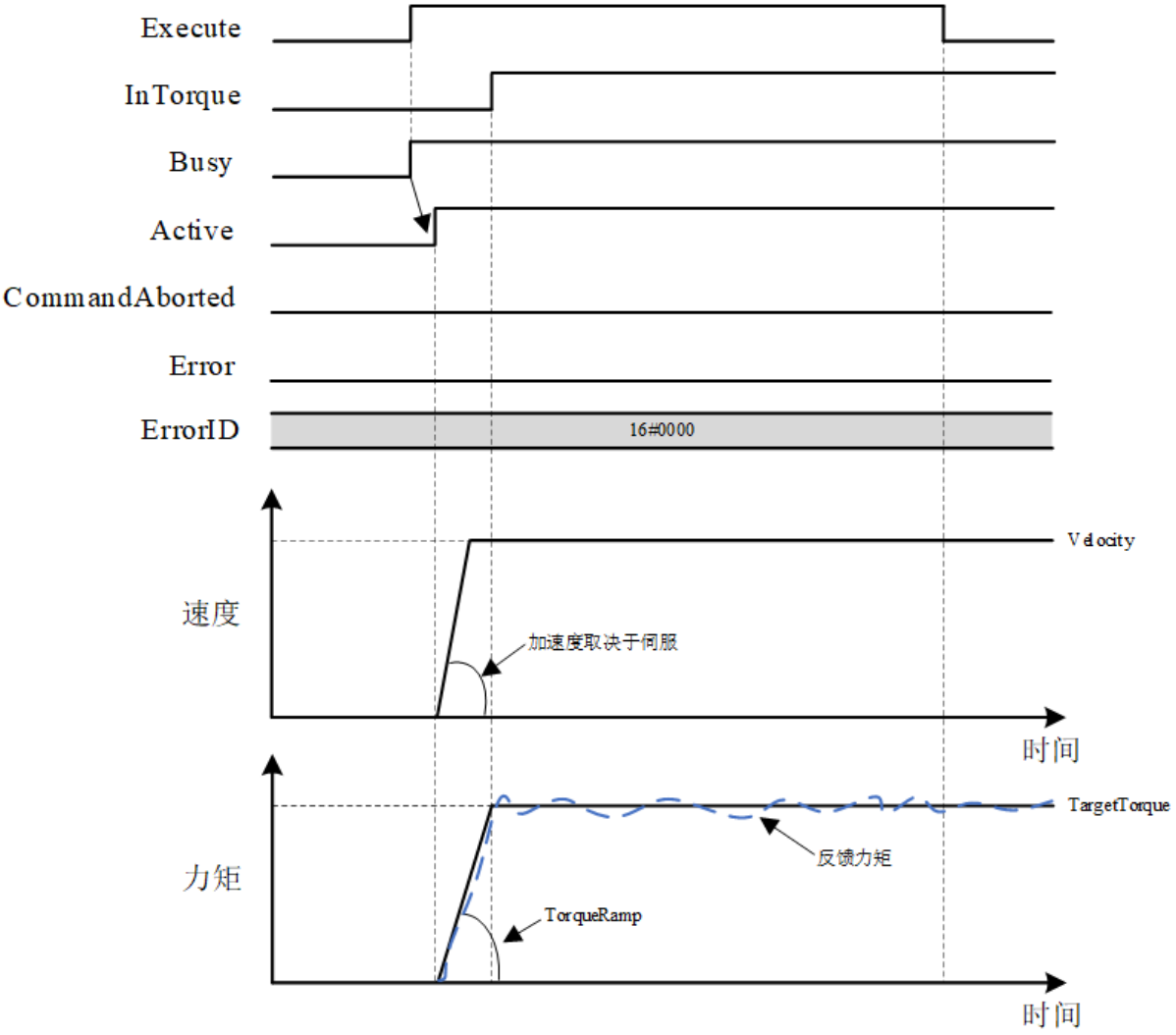
- 力矩控制模式下的停止控制

在该指令运行过程中，可以调用 MC_Stop 指令执行停止力矩运动的操作，停止时，该指令将目标力矩按照参数 TorqueRamp 的设定值从当前实际力矩减速至 0，然后撤销该力矩指令。停止完成后，将驱动器控制模式切换为力矩指令运动前的控制模式。

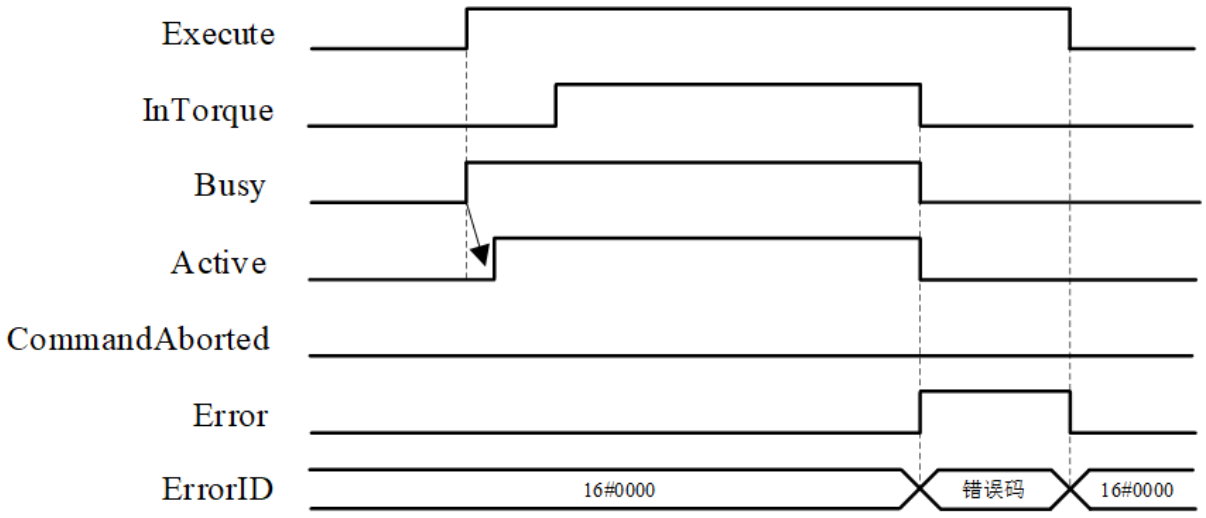
调用 MC_Stop 指令打断 MC_TorqueControl 指令时，轴状态从 ContinuousMotion 切换为 Stopping 状态，MC_TorqueControl 指令被打断时，其输出引脚 CommandAbort 有效。

- 功能块时序图

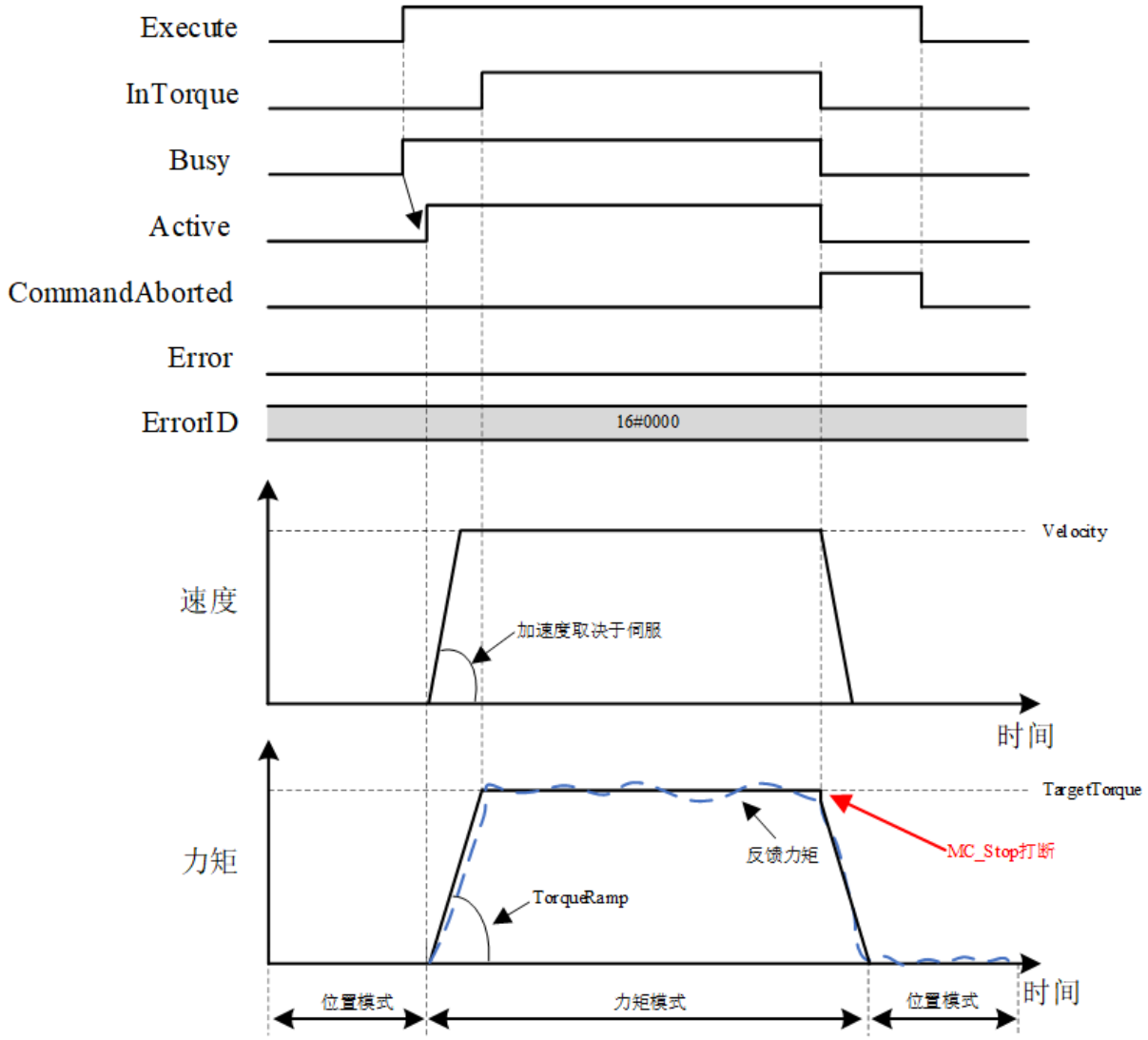
（1） 功能块正常执行且无错误的时序图（实际力矩可以达到给定力矩）。



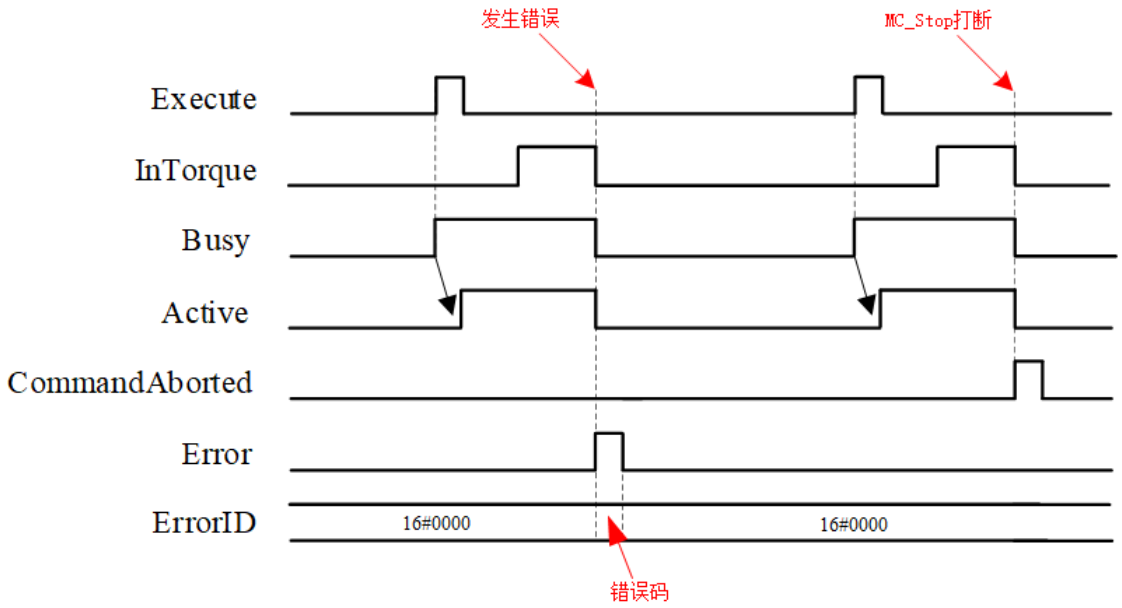
（2） 功能块执行过程中驱动器报错的时序图。



（3） 功能块运行过程中 MC_Stop 打断指令的时序图（假设力矩运动控制之前是位置控制模式）。



（4） Execute 保持一个周期时，功能块运行过程中发生错误和被 MC_Stop 指令打断的时序图。



5.11.3 示例

该示例使用 MC_TorqueControl 功能块进行力矩运动控制，并读取实际力矩值，最后调用 MC_Stop 指令停止力矩运动指令。

力矩运动控制时伺服需要映射所需的 PDO，否则会报错。

在进行力矩运动控制前，需要先使用 SMC_AxisInit 进行轴相关参数初始化，以及调用 MC_Power 使能轴。

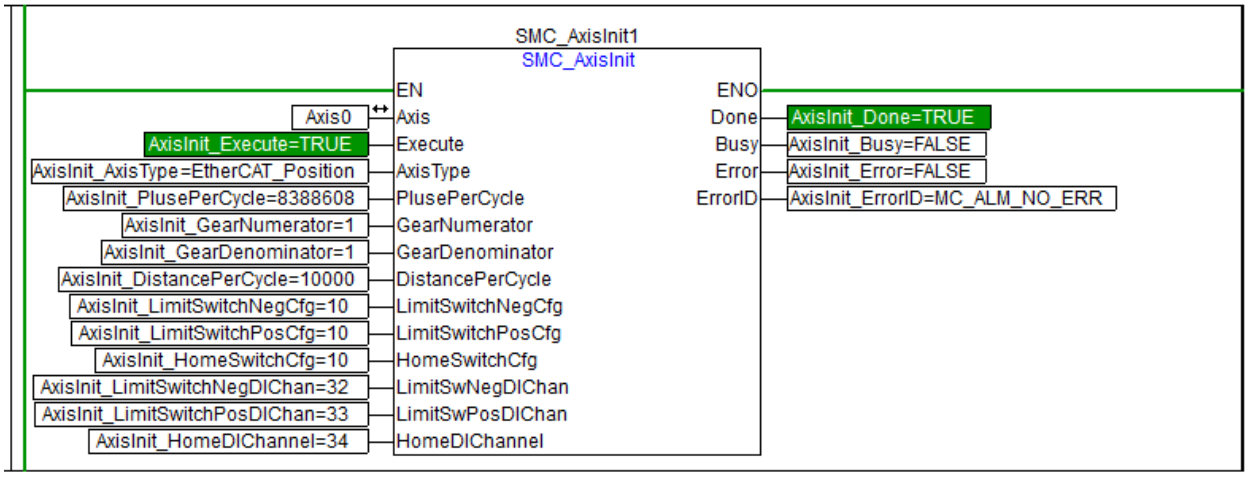
■ 变量

序号	变量名	变量类型	初始化值	变量说明
1	MC_TorqueControl0	MC_TORQUECONTROL	-	扭矩控制功能块实例定义
2	TorqueControl_En	BOOL	FALSE	扭矩控制使能信号
3	TorqueControl_Torque	LREAL	0	目标扭矩值
4	TorqueControl_TorqueRamp	LREAL	0	扭矩斜坡
5	TorqueControl_Vel	LRE 停止功能块 AL	0	扭矩控制最大速度限制
6	TorqueControl_Dir	MC_DIRECTION	moPositiveDirection	运动方向
7	TorqueControl_Buffer	MC_BUFFER_MODE	Aborting	缓冲模式
8	TorqueControl_InTorque	BOOL	FALSE	扭矩到达标志位
9	TorqueControl_Busy	BOOL	FALSE	扭矩控制执行中标志位

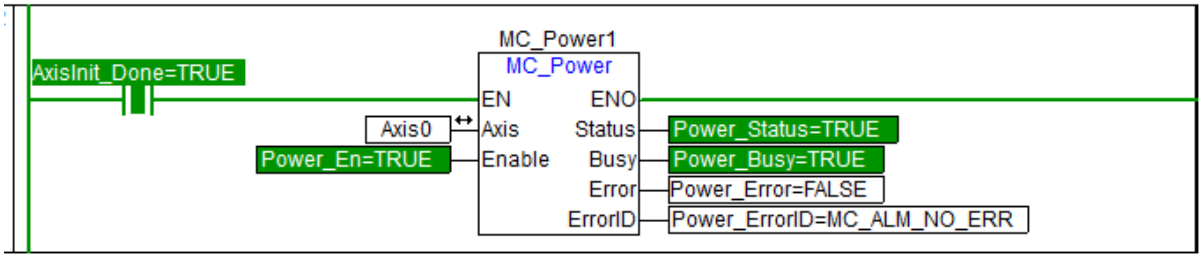
10	TorqueControl_Active	BOOL	FALSE	扭矩控制激活标志位
11	TorqueControl_Abort	BOOL	FALSE	扭矩控制中止标志位
12	TorqueControl_Error	BOOL	FALSE	扭矩控制错误标志位
13	TorqueControl_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	错误代码
14	MC_ReadActualTorque1	MC_READACTUALTORQUE	-	实际扭矩读取功能块实例定义
15	ReadTorque_En	BOOL	FALSE	实际扭矩读取使能信号
16	ReadTorque_Valid	BOOL	FALSE	实际扭矩数据有效标志位
17	ReadTorque_Busy	BOOL	FALSE	正在执行实际扭矩读取标志位
18	ReadTorque_Error	BOOL	FALSE	实际扭矩读取错误标志位
19	ReadTorque_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	实际扭矩读取错误代码
20	ReadTorque_Torque	LREAL	0	读取的实际扭矩值
21	MC_Stop1	MC_STOP	-	停止功能块实例定义
22	Stop_En	BOOL	FALSE	停止功能块使能信号
23	Stop_Dec	LREAL	0	停止功能块减速度
24	Stop_Jerk	LREAL	0	停止功能块加加速度
25	Stop_Done	BOOL	FALSE	停止完成标志位
26	Stop_Busy	BOOL	FALSE	正在停止标志位
27	Stop_Error	BOOL	FALSE	错误标志位
28	Stop_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	错误代码

■ LD 程序实现

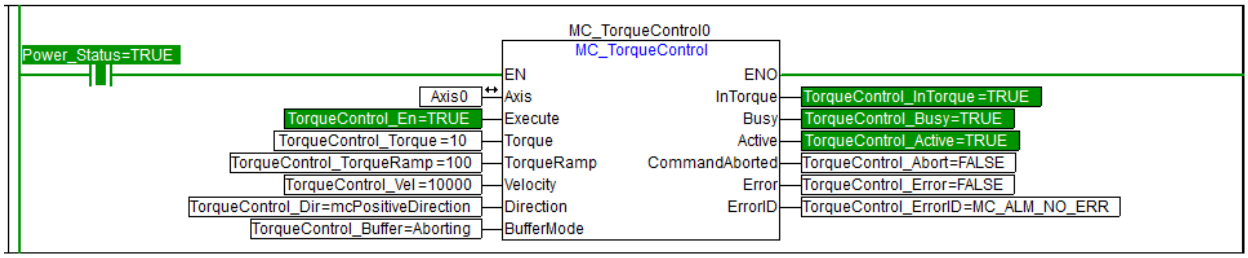
（1） 先调用 SMC_AxisInit 指令初始化轴参数，轴参需根据实际伺服情况进行设置。

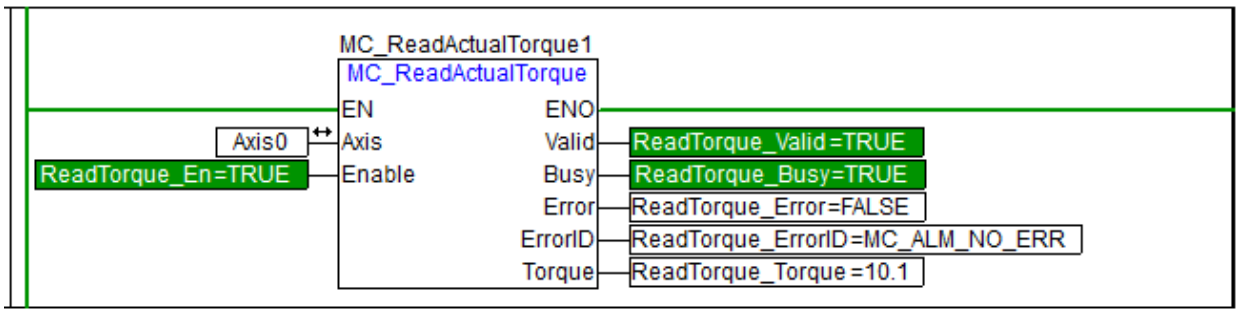


（2） 轴初始化完成后调用 MC_Power 指令进行轴使能操作。

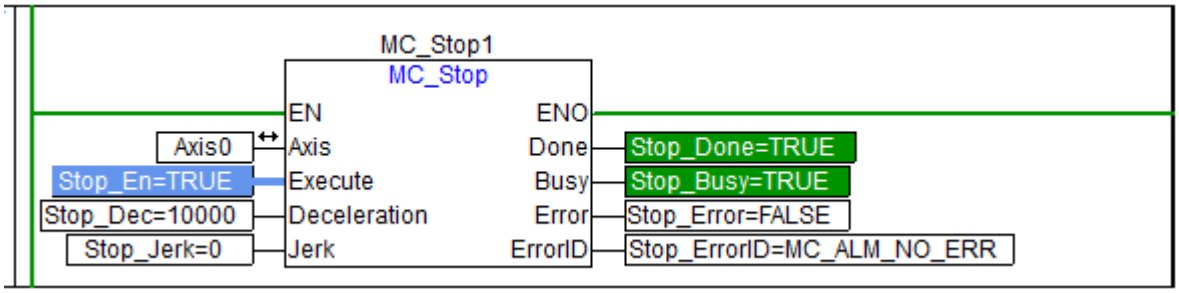


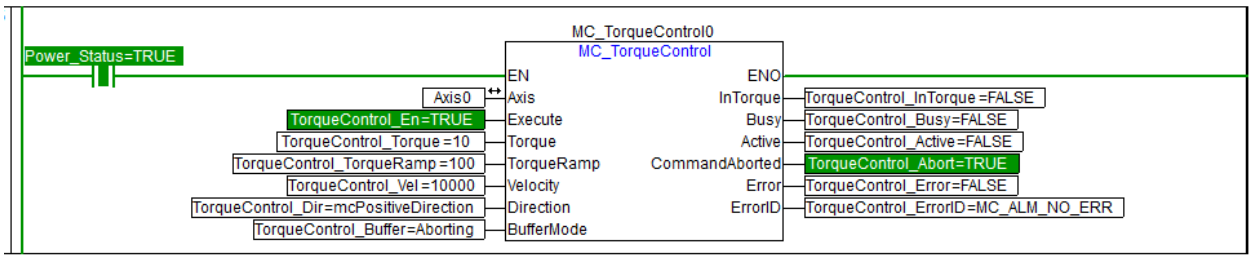
（3） 轴使能成功后调用 MC_TorqueControl 指令对轴进行力矩运动控制，可通过 MC_ReadActualTorque 获取伺服电机当前力矩值。





（4） 在需要停止力矩运动控制时，调用 MC_Stop 即可打断力矩控制指令，打断后，MC_TorqueControl 指令 CommandAborted 指令有效。





■ ST 程序实现

（1） 先调用 SMC_AxisInit 指令初始化轴参数，轴参需根据实际伺服情况进行设置。

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

（2） 轴初始化完成后调用 MC_Power 指令进行轴使能操作。

```
IF TRUE = AxisInit_Done THEN
   Power_En := TRUE;
```

```
ELSE
    Power_En := FALSE;
END_IF;
MC_Power1(
Enable := Power_En,
Axis := Axis0,
Status=>Power_Status,
Busy=>Power_Busy,
Error=>Power_Error,
ErrorID=>Power_ErrorID);
```

（3） 轴使能成功后调用 MC_TorqueControl 指令对轴进行力矩运动控制，可通过 MC_ReadActualTorque 获取伺服电机当前力矩值。

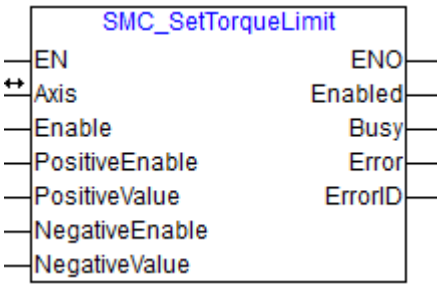
```
IF TRUE = Power_Status THEN
    MC_TorqueControl0(
    Execute := TorqueControl_En,
    Torque := TorqueControl_Torque,
    TorqueRamp := TorqueControl_TorqueRamp,
    Velocity := TorqueControl_Vel,
    Direction := TorqueControl_Dir,
    BufferMode := TorqueControl_Buffer,
    Axis := Axis0,
    InTorque=>TorqueControl_InTorque,
    Busy=>TorqueControl_Busy,
    Active=>TorqueControl_Active,
    CommandAborted=>TorqueControl_Abort,
    Error=>TorqueControl_Error,
    ErrorID=>TorqueControl_ErrorID);
END_IF
MC_ReadActualTorque1(
Enable := ReadTorque_En,
Axis := Axis0,
Valid=>ReadTorque_Valid,
Busy=>ReadTorque_Busy,
Error=>ReadTorque_Error,
ErrorID=>ReadTorque_ErrorID,
Torque=>ReadTorque_Torque);
```

（4） 在需要停止力矩运动控制时，调用 MC_Stop 即可打断力矩控制指令，打断后，MC_TorqueControl 指令 CommandAborted 指令有效。

```
MC_Stop1(
Execute := Stop_En,
Deceleration := Stop_Dec,
Jerk := Stop_Jerk,
Axis := Axis0,
Done=>Stop_Done,
Busy=>Stop_Busy,
Error=>Stop_Error,
ErrorID=>Stop_ErrorID);
```

5.12 SMC_SetTorqueLimit（轴力距限制值设置）

设定伺服正向/负向扭矩的限制值。



5.12.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Enable	使能	否	-	TRUE/FALSE	BOOL
	PositiveEnable	正向限幅使能	否	-	TRUE/FALSE	BOOL
	PositiveValue	正向最大扭矩 单位：1%额定扭矩	是	300	0 ~ 1000	LREAL
	NegativeEnable	负向限幅使能	否	-	TRUE/FALSE	BOOL
	NegativeValue	负向最大扭矩 单位：1%额定扭矩	是	300	0 ~ 1000	LREAL
OUTPUT	Enabled	设定成功	否	-	TRUE/FALSE	BOOL

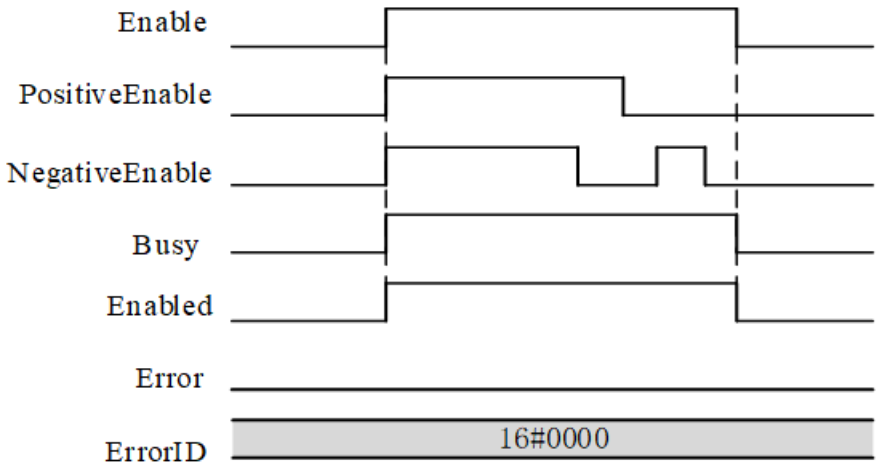
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.12.2 功能

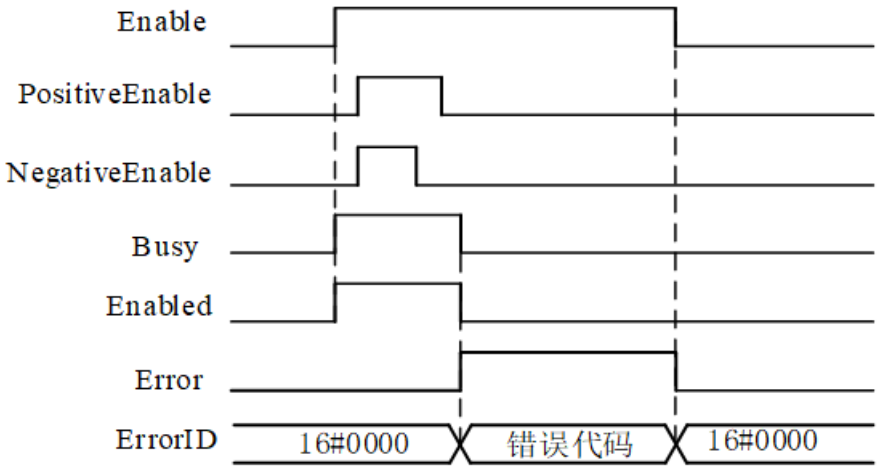
本指令可按照 CIA402 协议对伺服驱动器进行正负转矩限制值的设定，该指令详情介绍如下。

- Enable(使能)为 TRUE 时，如果将 PositiveEnable(正向线幅使能)设为 TRUE，则以 PositiveValue(正向最大扭矩)进行限制正向最大扭矩值；如果将 NegativeEnable(负向线幅使能)设为 TRUE，则以 NegativeValue(负向最大扭矩)进行限制负向最大扭矩值。
- Enable(使能)为 FALSE 时，则不进行限制值的设定，使用之前设定的限制值进行转矩的限制，将 PositiveEnable(正向限幅使能)或者 NegativeEnable(负向限幅使能)设为 FALSE 时，则不进行正负限制值的设定，保持之前设定的值进行限制。
- 设定值的单位是 1%的额定扭矩，设定值小数点后超过第二位时，第二位进行四舍五入的计算。
- 本指令的输入为 Enable 型，Enable 为 TRUE 时功能块一直运行，不存在指令的重启触发。
- 对于正负限制值的非法性检查，只有在 PositiveEnable(正向限幅使能)或 NegativeEnable(负向限幅使能)设为 TRUE 时才会进行检查。
- 功能块时序图

  (1) 正常情况时序图



（2） 异常情况时序图



5.12.3 示例

组建 SMC_SetTorqueLimit 示例程序设定伺服正向/负向扭矩的限制值。

■ 主要变量

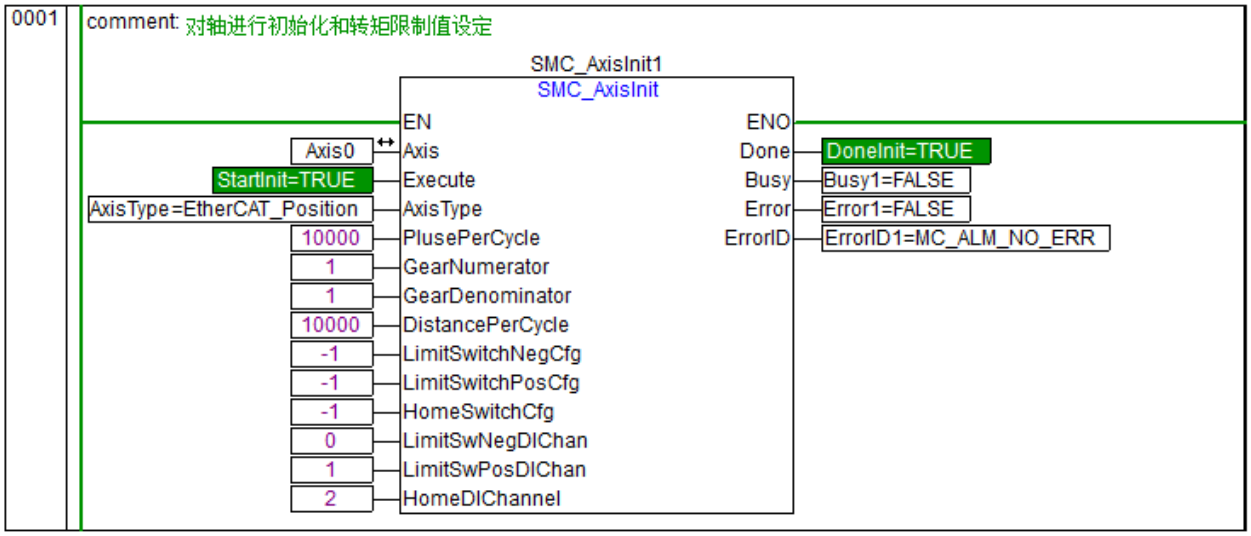
名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
SetTorqEnable	BOOL	FALSE	开始使能时变为 TRUE
PosSetTorqEnable	BOOL	FALSE	设置正向使能时变为 TRUE
NegSetTorqEnable	BOOL	FALSE	设置负向使能时为 TRUE

EnabledSetTor	BOOL	FALSE	设定成功时变为 TRUE
BusySetTor	BOOL	FALSE	开始设定时变为 TRUE
ErrorSetTor	BOOL	FALSE	出错时变为 TRUE

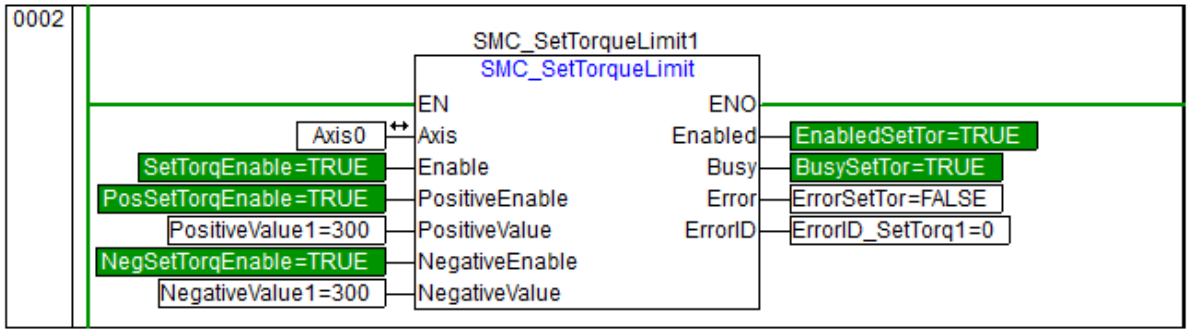
■ 梯形图程序

利用梯形图组建程序，输入参数 PositiveValue 和 NegativeValue 均为 300。将 SMC_SetTorqueLimit 的 Enable(使能)置为 TRUE，示例程序如下：

（1） 在进行轴的转矩设置之前，首先要对轴进行初始化。



（2） 按照设定的值设定伺服正向/负向扭矩的限制值



■ ST 语言程序

（1） 初始化轴号和轴类型设置。

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
```

```
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

（2） 对轴进行转矩限制设置

```
SMC_SetTorqueLimit1(
Enable := SetTorqEnable,
PositiveEnable := PosSetTorqEnable,
PositiveValue := 300,
NegativeEnable := NegSetTorqEnable,
NegativeValue := 300,
Axis := Axis0,
Enabled=>EnabledSetTor,
Busy=>BusySetTor,
Error=>ErrorSetTor,
ErrorID=>ErrorID_SetTorq1);
```

5.12.4 使用注意事项

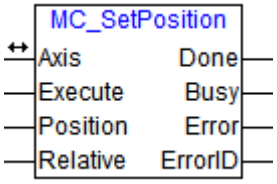
- 该功能块不支持虚拟轴和编码器轴。
- 由于伺服电机存在差别，有些伺服内部转矩限制默认值为 350 或者 300，当设置值超过这个值就会有报错产生，伺服内部的最大值为其默认值。

5.12.5 参见

- 若有错误发生时，参见附录功能块错误码。
- 设置完限制值使用时参见 MC_TorqueControl 指令章节。

5.13 MC_SetPosition（设定当前位置）

以绝对或相对的方式重新定义指定轴的当前位置。



5.13.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Position	位置	否	-	正值/负值/0	LREAL
	Relative	模式选择 FALSE：绝对模式，将 Position 定义为当前位置 TRUE：相对模式，在当前位置的基础上增加 Position 对应的值	是	FALSE	TRUE/FALSE	BOOL
OUTPUT	Done	完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.13.2 功能

本指令以绝对或相对的方式重新定义指定轴的当前位置。

- 指令功能详情

（1） 入参锁存

本指令上升沿触发有效。在 Execute 输入的上升沿时，将输入输出参数 Axis 的轴 ID、输入参数 Position 和 Relative 值锁存，指令执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令。

（2） 位置设定模式

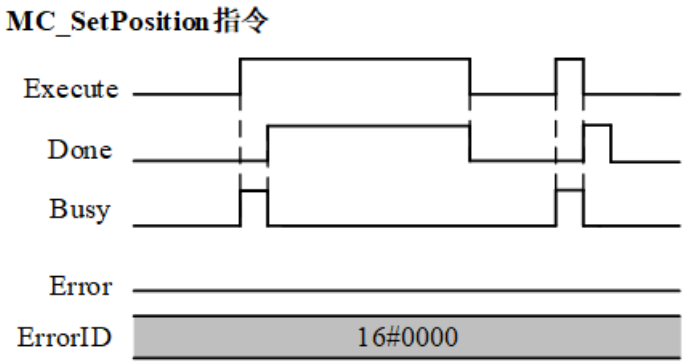
输入参数 Relative=FALSE 时，本指令以绝对的方式重新定义指定轴的当前位置，执行本指令后，将 Position 定义为轴的当前位置；

输入参数 Relative=TRUE 时，本指令以相对的方式重新定义指定轴的当前位置， 执行本指令后，在轴的当前位置上加上 Position 作为轴的当前位置。

■ 时序图

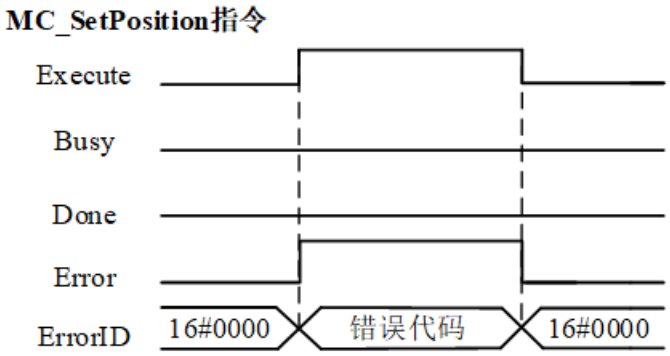
（1） 正常时序图

在 Execute 输入的上升沿，若无异常情况，Busy 引脚置为 TRUE，轴位置设置成功后，Done 引脚置为 TRUE，此时 Busy 引脚置为 FALSE，Done 引脚至少保持一个周期。



（2） 异常错误发生时引脚时序

执行本指令时，参数非法、轴类型非法（轴类型为 Unused）时，调用本指令失败。此时，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Execute 为低电平时，所有输出引脚恢复为默认值。



- 当指令的异常情况解决后，需要重新上升沿触发 Execute，启动 MC_SetPosition 功能块，完成轴的当前位置设定。

5.13.3 示例

组建 MC_SetPosition 示例程序设置轴的当前位置。

■ 梯形图

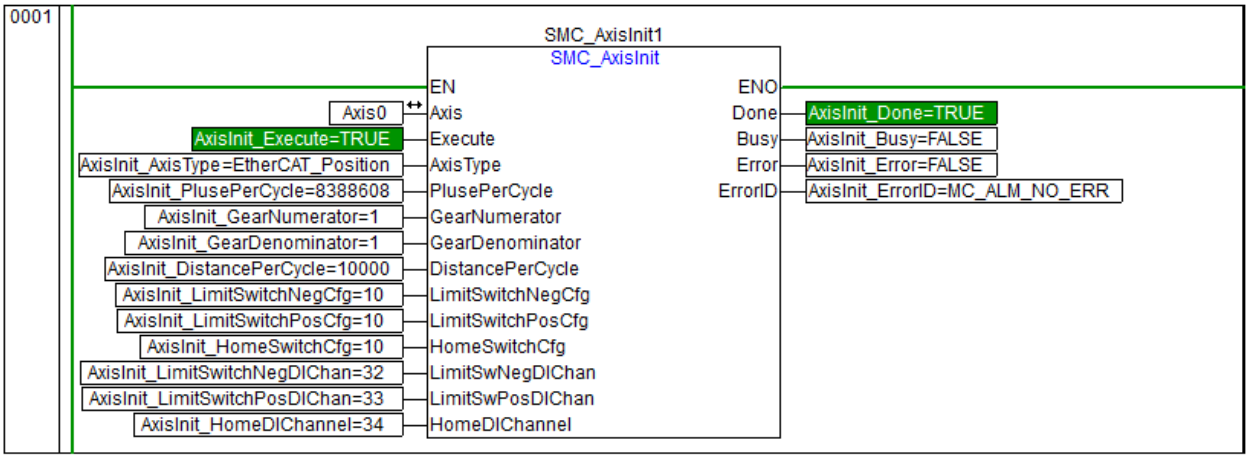
利用梯形图组建程序，输入参数 Position 为 0，Relative 为 FALSE，以绝对方式定义当前轴位置为 0。

主要变量说明

变量名称	变量类型	初始值	注释
Axis0	AXIS_REF	-	轴 0 的轴变量
SetPos_Done	BOOL	FALSE	轴当前位置设置成功时的变量。设置轴的当前位置成功时，该变量变为 TRUE。
Switch1	BOOL	FALSE	轴准备就绪的状态变量，轴初始化完成后， Switch 为 TRUE，轴准备就绪。

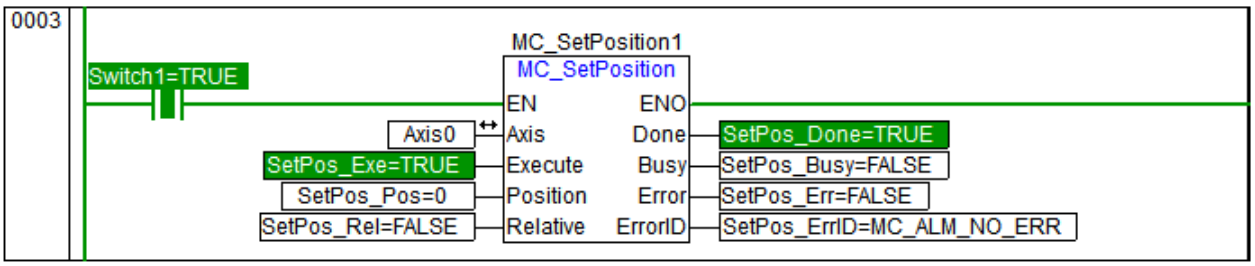
（1） 轴初始化

设置 Axis0.AxisID = 0，轴类型设置为 50：EtherCAT_Position，调用轴初始化 SMC_AxisInit 指令完成轴初始化配置。SMC_AxisInit 指令使用方法参见其指令说明章节。



（2） 执行 MC_SetPosition 指令

调用 MC_SetPosition 完成轴的当前位置的设置。当轴 Axis0 的初始化配置完成后，待轴准备就绪（Switch1=TRUE），设置轴当前位置。



■ ST 语言

利用 ST 语言组建程序，输入参数 Position 为 10000，Relative 为 TRUE，以相对方式定义当前轴位置，轴在当前位置的基础上增加 10000 作为当前轴位置。

（1） 主要变量

ST 语言程序主要变量同 LD 语言程序变量。

（2） 轴初始化

设置 Axis0.AxisID = 0，轴类型设置为 50：EtherCAT_Position，调用轴初始化 SMC_AxisInit 指令完成轴初始化配置。SMC_AxisInit 指令的使用方法参见其指令介绍章节。

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
```

```
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

（3） 执行 MC_SetPosition 指令

绝对方式重新定义指定轴的当前位置。

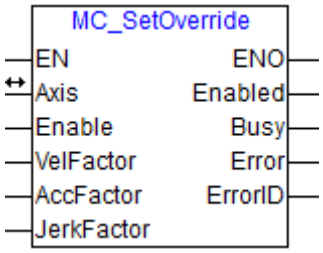
```
  Relative:=TRUE;(*模式选择参数设定*)

  Position:=10000; (*位置参数设定*)

IF  Switch1 THEN
  MC_SetPosition1(
  Execute := Exe_SetPos,
  Position := Position,
  Relative := Relative,
  Axis := Axis0,
  Done=>Done_Pos,
  Busy=>Busy_Pos,
  Error=>Error_Pos,
  ErrorID=>ErrorID_Pos);
END_IF
```

5.14 MC_SetOverride（运动倍率设置（超驰））

该指令设定速度、加减速、加加速度比例因子，可按设定比例放大或缩小轴运行的速度、加减速、加加速度。



5.14.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Enable	使能	否	-	TRUE/FALSE	BOOL
	VelFactor	速度比例因子，单位为 1%	是	100	0 ~ 500	LREAL
	AccFactor	加减速度比例因子，单位为 1%	是	100	0 ~ 500（非零）	LREAL
	JerkFactor	加加速度比例因子，单位为 1%	是	100	0 ~ 500（非零）	LREAL
OUTPUT	Enabled	设定成功	否	-	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.14.2 功能

本指令可按照设定的比例因子放大或缩小轴运行的速度、加减速、加加速度，该指令详情介绍如下：

- 该指令对轴的目标速度进行超调，可以变更轴运动过程中的目标速度，可以变更的指令如下所示：
  - MC_MoveJog
  - MC_MoveRelative
  - MC_MoveAbsolute
  - MC_MoveVelocity
- 超调比例因子的单位是[%]。

  变更后的目标速度＝ 当前执行的速度×速度比例因子；

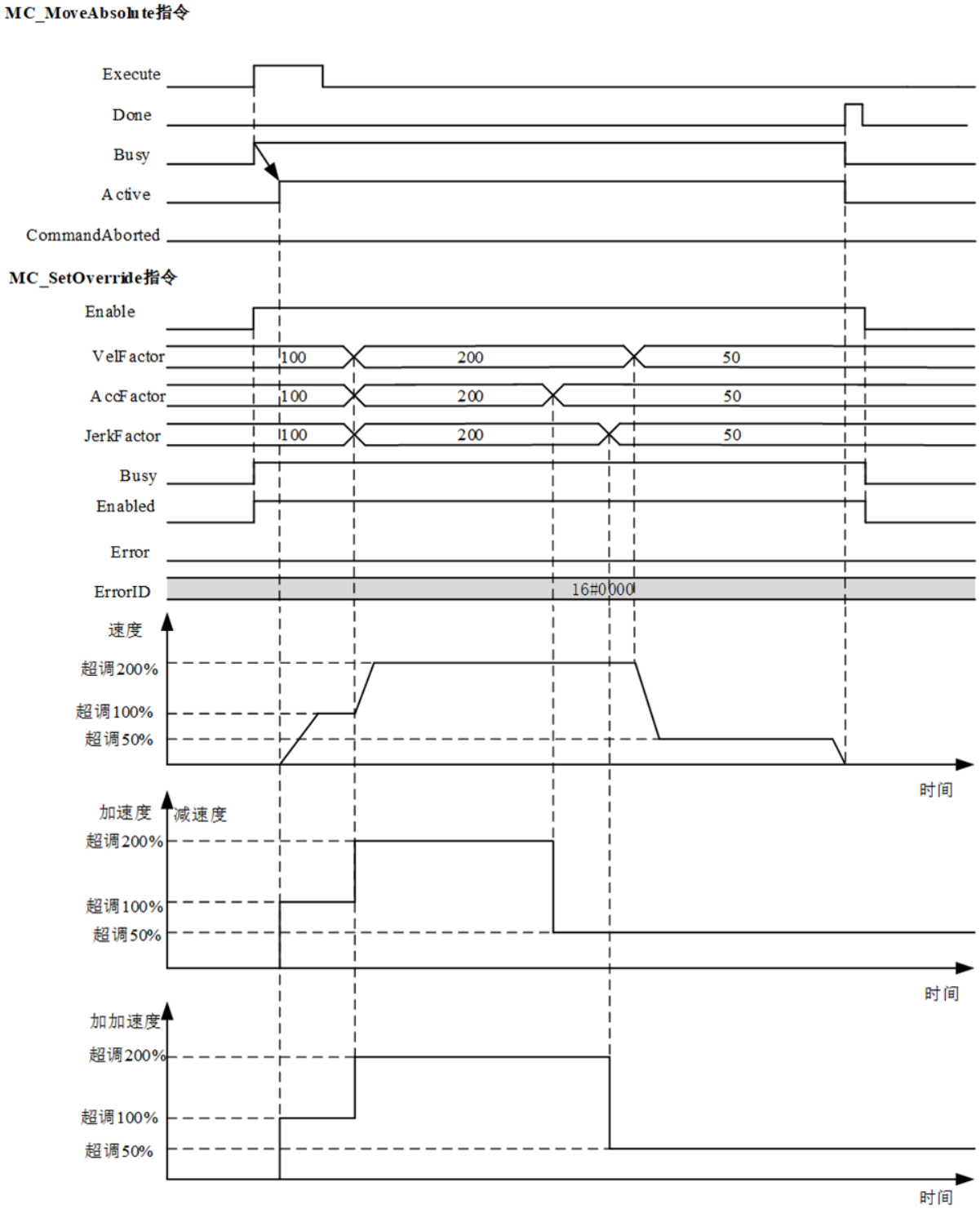
  变更后的加减速度 = 当前的加减速度×加减速度比例因子；

  变更后的加加速度 = 当前的加加速度 ×加加速度比例因子。

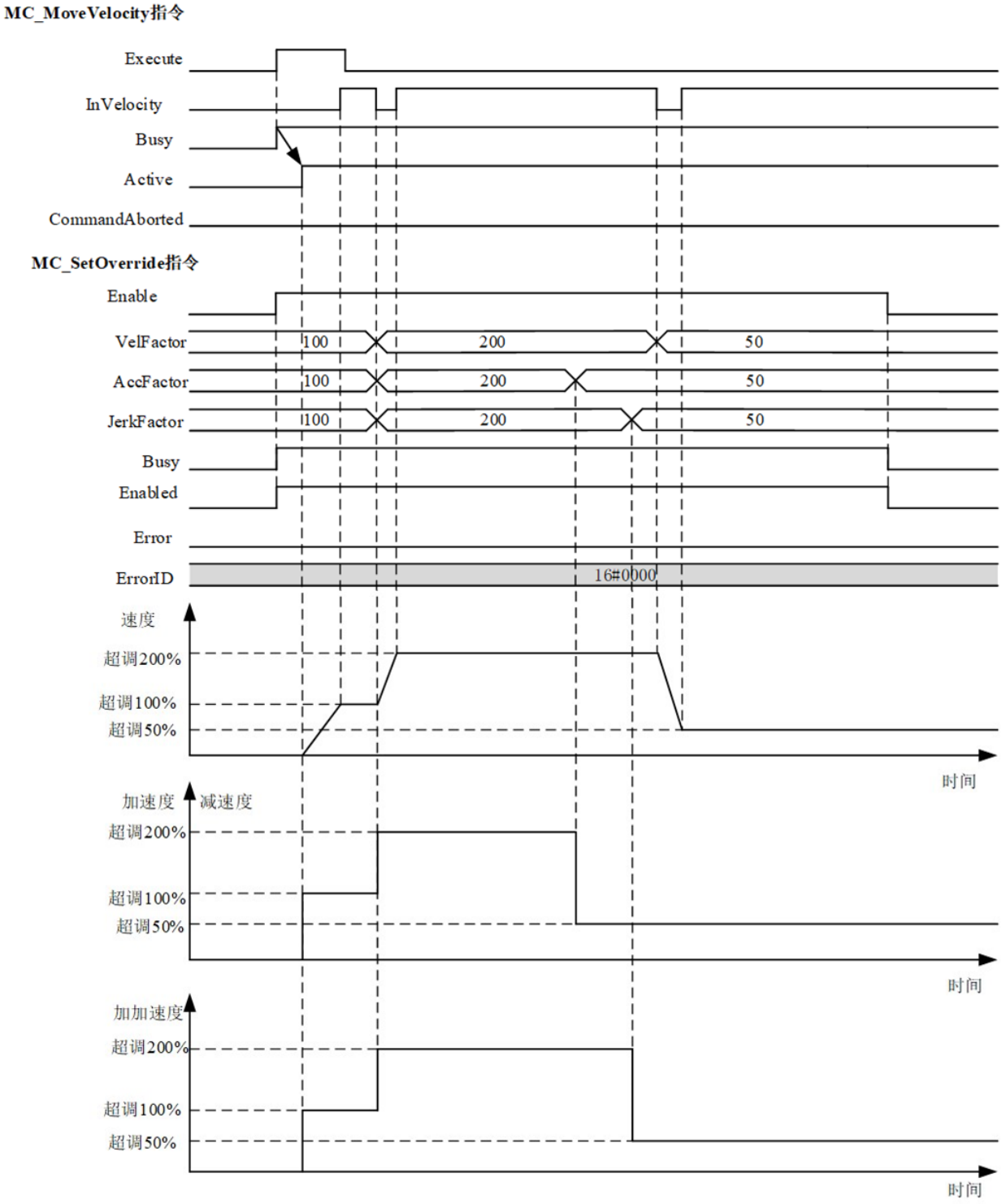
- 当超调后的目标速度大于最大速度时，轴运行的速度为最大速度，加减速度和加加速度也是一致的表现。
- 在 Enable(使能)为 TRUE 时，功能块持续进行比例因子超调设置，当 Enable(使能)为 FALSE 时，轴保持上一次设置的超调比例因子值进行运动。
- 功能块时序图

（1） 对 MC_MoveAbsolute(绝对定位指令)进行超调设置

在 Enable(使能)为 TRUE 时，Busy(忙标志)和 Enabled(设定成功)立即为 TRUE，Enable(使能)为 FALSE 时，Busy(忙标志)和 Enabled(设定成功)立即为 FALSE，在 MC_MoveAbsolute(绝对定位指令)中使用超调指令时的时序图如下所示。

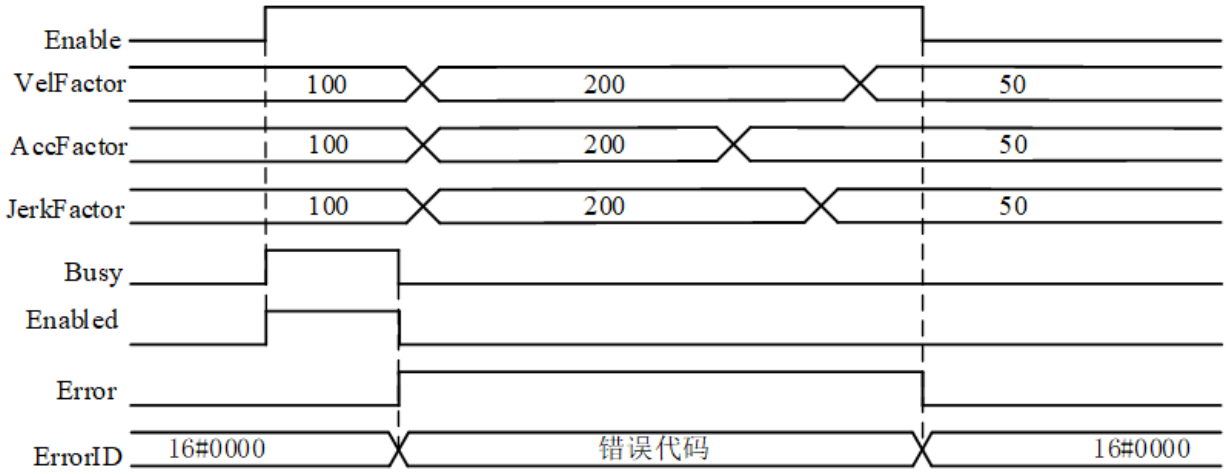


（2） 对 MC_MoveVelocity(速度控制)指令执行超调，当 InVelocity(达到目标速度)变为 TRUE 后，使用超调变更目标速度后，InVelocity(达到目标速度)会变为 FALSE，当达到新的目标速度之后，InVelocity(达到目标速度)变为 TRUE，将 MC_SetOverride 的 Enable(使能)设为 FALSE 时，轴保持上一次设置的超调比例因子值进行运动，指令时序图如下所示。



（3） 异常时的时序图

在执行本指令发生异常时，Error(错误)变为 TRUE，Enabled(设定成功)和 Busy(忙标志)变为 FALSE，当错误清除后，Error(错误)变为 FALSE，Enabled(设定成功)和 Busy(忙标志)变为 TRUE，时序图如下所示。



■ 重启和多重启动指令

本指令的输入为 Enable 型，Enable(使能)为 TRUE 时功能块一直运行，当存在正在执行的 MC_SetOverride 指令时，启动其他的 MC_SetOverride 指令时，后执行的指令被优先处理。

5.14.3 示例

组建 MC_SetOverride 示例程序设定轴运动参数的超调值。

■ 主要变量

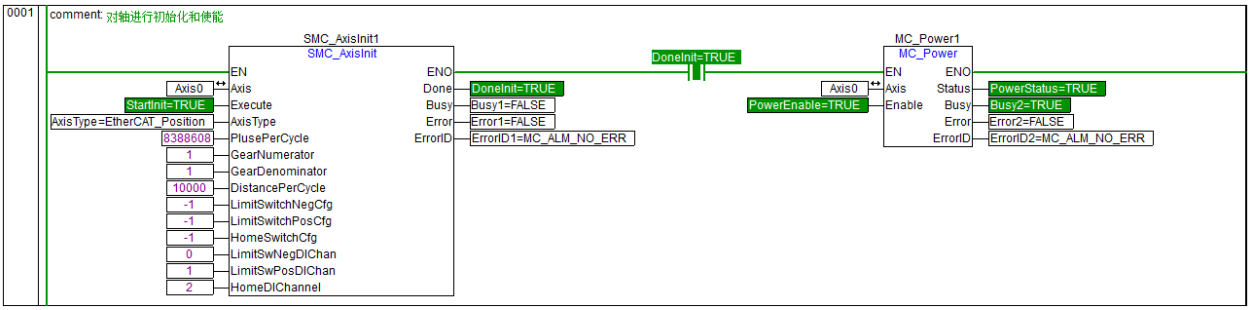
名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
PowerEnable	BOOL	FALSE	开始使能时变为 TRUE
PowerStatus	BOOL	FALSE	使能成功时变为 TRUE
StartMoveVel	BOOL	FALSE	启动运动指令时为 TRUE
InVelMoveVel	BOOL	FALSE	指令速度达到时变为 TRUE
StartOverride	BOOL	FALSE	开始执行超调时变为 TRUE
BusyOverride	BOOL	FALSE	超调时变为 TRUE
EnabledOverride	BOOL	FALSE	超调成功时变为 TRUE
ErrorOverride	BOOL	FALSE	指令出错时变为 TRUE

■ 梯形图程序

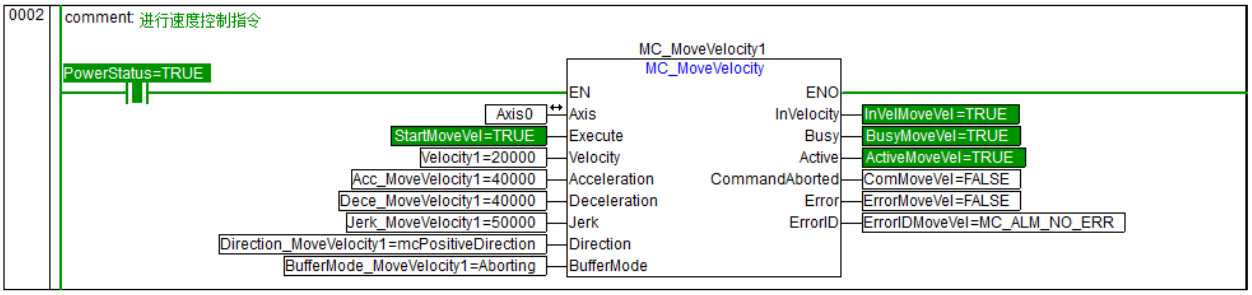
利用梯形图组建程序，输入参数 VelFactor、AccFactor 和 JerkFactor 均为 300。执行

MC_MoveVelocity 指令，设置 MC_MoveVelocity 速度为 20000，控制轴进行运动。将 MC_SetOverride 的 Enable(使能)置为 TRUE，轴的速度，加减速度和加加速度均表现为超调之后的值，示例程序如下：

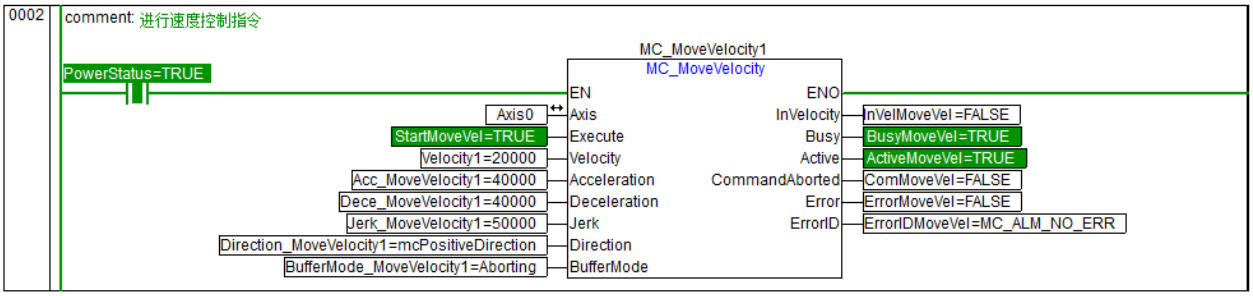
（1） 对轴进行初始化以及使能，轴 ID 设为 0，轴类型设为 EtherCAT_Position 类型，待轴初始化完成之后，对轴进行使能。

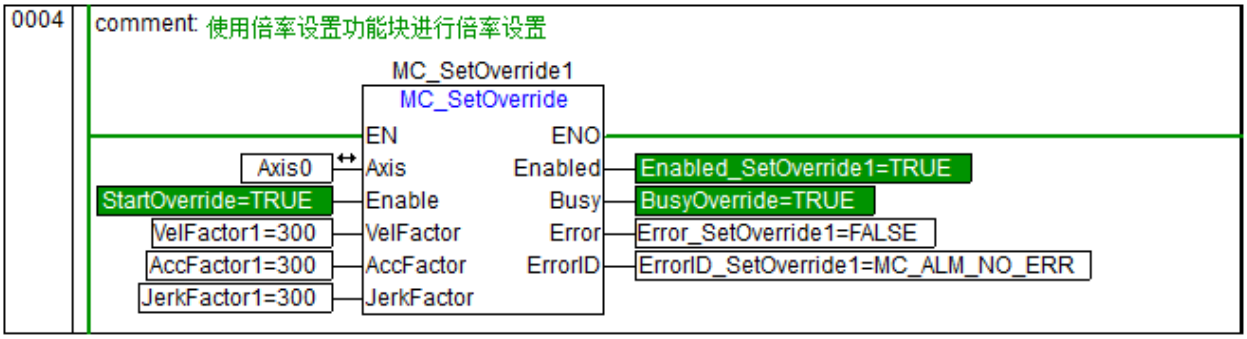


（2） 调用 MC_MoveVelocity 实例输出引脚变量的状态，InVelocity 引脚为 TRUE，MC_MoveVelocity 指令的实际速度达到设定的指令速度 20000。



（3） 保持 MC_SetOverride 的 Enable 引脚为高电平，进行超调参数的设置，可以看见在设置参数之后 MC_MoveVelocity 实例输出引脚变量 InVelocity 引脚变为 FALSE 一段时间，当达到速度之后 InVelocity 引脚又变为 TRUE。





■ ST 语言程序

（1） 初始化轴号和轴类型设置，完成初始化之后对轴进行使能。

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
(*对轴进行使能*)

IF DoneInit THEN
```

```
    MC_Power1(
    Enable := PowerEnable,
    Axis := Axis0,
    Status=> PowerStatus,
    Busy=>Busy2,
    Error=>Error2,
    ErrorID=>ErrorID2);
END_IF
```

（2） 上升沿触发 MC_MoveVelocity 指令，控制轴进行运动。

```
MC_MoveVelocity1(
Execute := StartMoveVel,
Velocity := 20000,
Acceleration := 40000,
Deceleration := 40000,
Jerk := 50000,
Direction := 0,
BufferMode := 0,
Axis := Axis0,
InVelocity=> InVelMoveVel,
Busy=> BusyMoveVel,
Active=> ActiveMoveVel,
CommandAborted=> ComMoveVel,
Error=> ErrorMoveVel,
ErrorID=> ErrorIDMoveVel);
```

（3） 保持 MC_SetOverride 为高电平，进行运动参数的超调设置。

```
MC_SetOverride1(
Enable := StartOverride,
VelFactor := 300,
AccFactor := 300,
JerkFactor := 300,
Axis := Axis0,
Enabled=>EnabledOverride,
Busy=>BusyOverride,
Error=>ErrorOverride,
ErrorID=>ErrorID);
```

5.14.4 使用注意事项

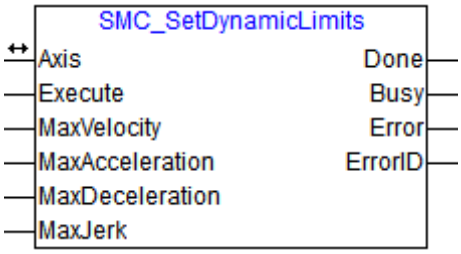
- 使用该指令进行超调设置时，若超过轴的最大速度，则以最大速度为运行速度
- 该指令对编码器轴不生效。

5.14.5 参见

- 关于轴初始化和使能，参见 SMC_AxisInit（轴初始化配置）和 MC_Power（轴使能）的指令章节。
- 关于错误码可参加附录功能块错误码。

5.15 SMC_SetDynamicLimits（运动参数限制值设置）

该指令设定轴运动参数的限制值。



5.15.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	MaxVelocity	最大速度	是	1E100	正值/0	LREAL
	MaxAcceleration	最大加速度	是	1E100	正值	LREAL
	MaxDeceleration	最大减速度	是	1E100	正值	LREAL
	MaxJerk	最大加加速度	是	1E100	正值	LREAL
OUTPUT	Done	完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL

	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.15.2 参数

本指令设定轴运动参数的限制值。

■ 指令功能详情

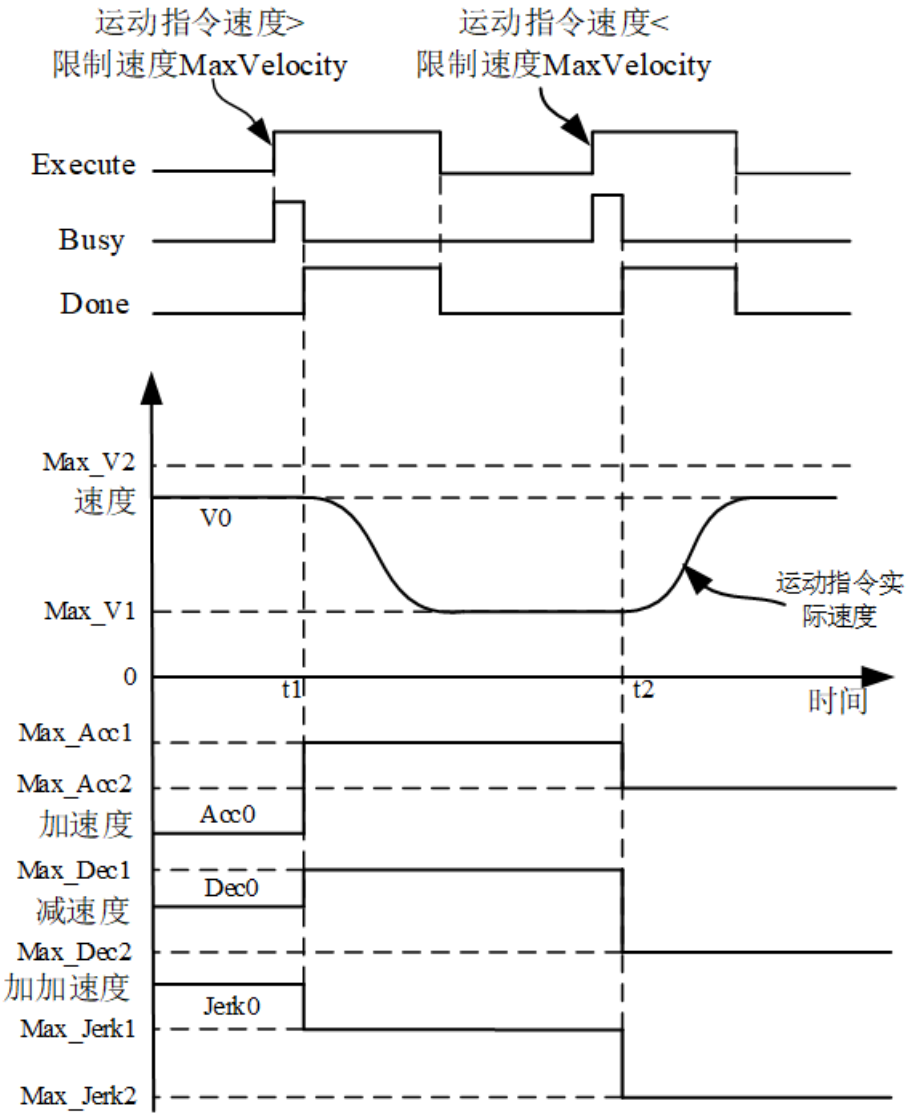
（1） 入参锁存

本指令上升沿触发有效。在 Execute 输入的上升沿时，将输入输出参数 Axis 的轴 ID、输入参数 MaxVelocity、MaxAcceleration、MaxDeceleration 和 MaxJerk 值锁存，指令执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令。

（2） 功能详情

设置轴运动参数的限制值后，轴运动指令的运动参数取自身运动参数和设置的限制参数的最小值。如下图，执行两次该指令，每次执行时设置不同的最大速度 MaxVelocity、最大加速度 MaxAcceleration、最大减速度 MaxDeceleration 和最大加加速度 MaxJerk 值。

设置最大运动参数如下图所示：



上图中，V0 为运动指令的指令速度、Acc0 为运动指令的指令加速度、Dec0 为运动指令的指令减速度、Jerk0 为运动指令的指令加加速度。

在 t1 时刻，最大限制速度 MaxVelocity（图中的 Max_V1 值）设置成功， 最大限制速度 MaxVelocity 小于轴运动指令速度 V0，则轴运动指令实际运行的最大速度为限制值 MaxVelocity，轴运动指令降速。同时，Max_Acc1、Max_Dec1、Max_Jerk1 分别为设定的其余运动参数限制值。

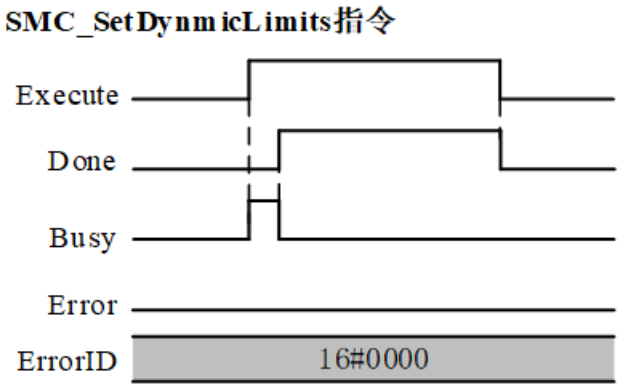
在 t2 时刻，再次成功设置最大限制速度 MaxVelocity， MaxVelocity= Max_V2，MaxVelocity 大于轴运动指令的速度 V0，轴运动指令以自身设置的指令速度作为实际运行速度，轴运动指令升速。同时，Max_Acc2、Max_Dec2、Max_Jerk2 分别为设定的其余运动参数限制值。

■ 时序图

（1） 正确时序图

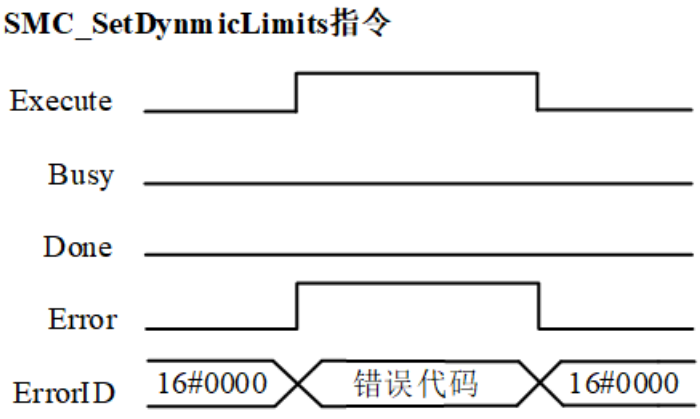
在 Execute 输入的上升沿，若无异常情况，Busy 引脚置为 TRUE，参数设置完成后，Done 置为 TRUE，Busy 置为 FALSE，若 Execute 保持高电平，则 Done 引脚保持 TRUE，若 Execute 置低电平，则 Done 引脚置为 FALSE。

指令执行时的正确时序如下：



（2） 异常错误发生时引脚时序

执行本指令时，参数非法、轴类型非法（轴类型为 Unused）时，调用本指令失败，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Execute 变为低电平时，所有输出引脚恢复为默认值。



- 当指令的异常情况解决后，需要重新上升沿触发 Execute，启动 SMC_SetDynamicLimits 功能块，完成轴的运动参数限制设定。

5.15.3 示例

组建 SMC_SetDynamicLimits 示例程序设定轴运动参数的限制值。

SMC_SetDynamicLimits 指令输入参数 MaxVelocity 为 20000，MaxAcceleration、MaxDeceleration、MaxJerk 均为 50000。执行指令 MC_Movelocity，指令 MC_Movelocity 设置速度为 50000，加减速度、加加速度均为 100000，控制轴运动。执行 SMC_SetDynamicLimits 指令，轴运动受到速度限制，实际运行速度为 20000，加速度和减速度均为限制值 50000。示例程序如下：

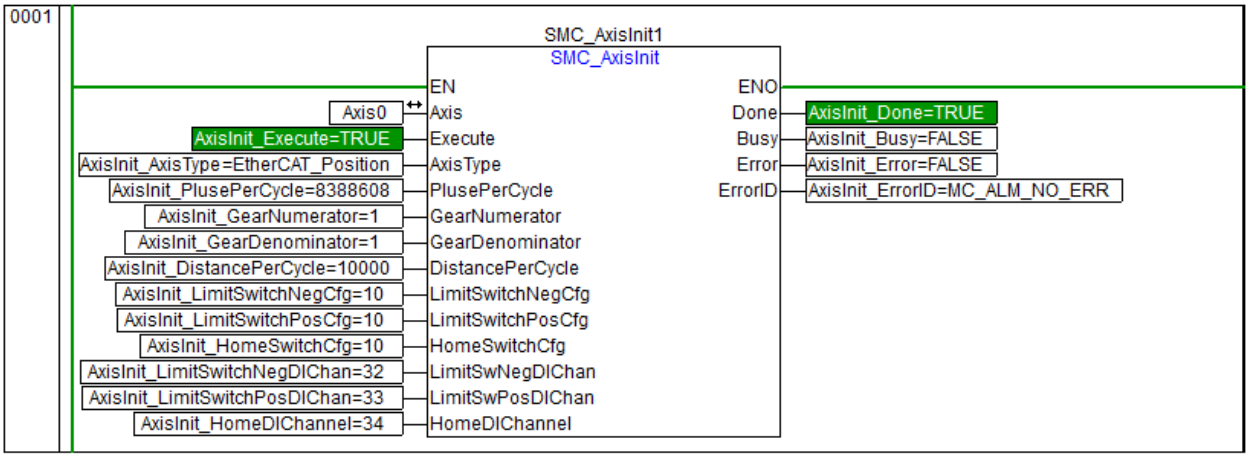
■ 梯形图

主要变量说明

变量名称	变量类型	初始值	注释
Axis0	AXIS_REF	-	轴 0 的轴变量
Power1_Status	BOOL	FALSE	轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
MV1_Vel	LREAL	-	轴速度控制指令的的输入 Velocity 的变量。控制轴运动的指令速度。
MV1_Active	BOOL	FALSE	轴速度控制指令动作时的变量。轴控制运动时，该变量变为 TRUE。
MV1_InVel	BOOL	FALSE	轴速度控制指令达到目标速度的标识变量。该变量变为 TRUE，速度指令的速度达到目标速度。
Switch1	BOOL	FALSE	轴准备就绪的状态变量，轴初始化完成后， Switch 为 TRUE，轴准备就绪。

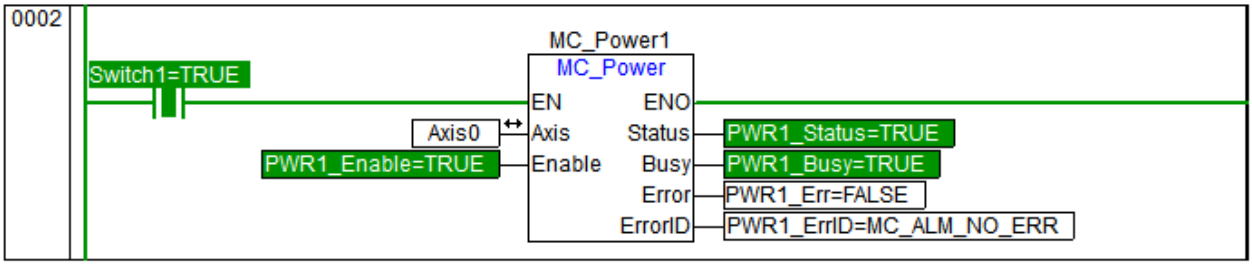
（1） 轴初始化

Axis0 的 AxisID 设为 0，轴类型设置为 50：EtherCAT_Position，调用轴初始化 SMC_AxisInit 指令完成 Axis0 的初始化配置，SMC_AxisInit 示的指令使用方法参见其介绍章节。



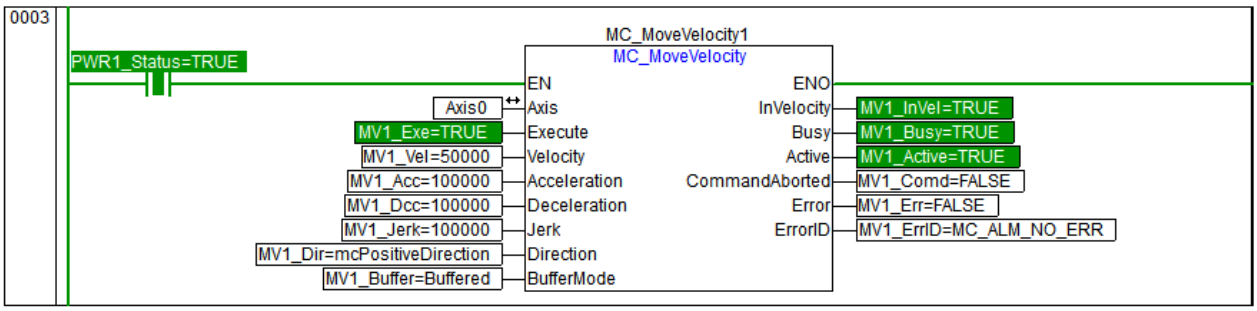
（2） 轴使能

轴初始化完成，轴准备就绪状态 Switch1 为 TRUE 后，调用 MC_Power 完成轴使能。



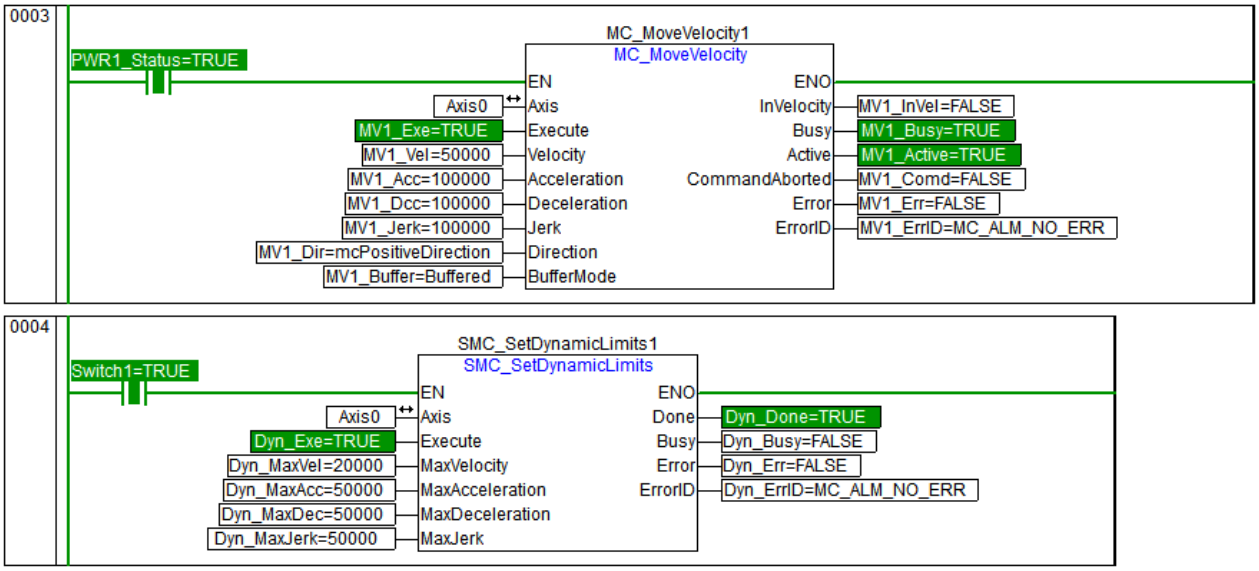
（3） 轴投送运动指令

0003 节中，轴使能成功后，上升沿触发 MC_MoveVelocity 运动指令，控制轴运动，通过 MC_MoveVelocity 实例输出引脚变量的状态，InVelocity 引脚为 TRUE， MC_MoveVelocity 指令的实际速度达到设定的指令速度 MV1_Vel= 50000。



（4） 设置运动参数限制

0004 节中，触发 SMC_SetDynamicLimits1 的 Execute 引脚，完成运动参数限制的设置。0003 节中， MC_MoveVelocity 实例输出引脚变量 InVelocity 引脚为 FALSE，可以直观得到 MC_MoveVelocity 指令的速度受限，该指令的实际速度未达到设定的指令速度。



■ ST 语言

主要变量说明

变量名称	变量类型	初始值	注释
Axis0	AXIS_REF	-	轴 0 的轴变量
Power1_Status	BOOL	FALSE	轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
MV1_Vel	LREAL	-	轴速度控制指令的的输入 Velocity 的变量。控制轴运动的指令速度。
MV1_Active	BOOL	FALSE	轴的速度控制指令动作时的变量。轴控制运动时，该变量变为 TRUE。
MV1_InVel	BOOL	FALSE	轴速度控制指令达到目标速度的标识变量。该变量变为 TRUE，速度指令的速度达到目标速度。
Switch1	BOOL	FALSE	轴准备就绪的状态变量，轴初始化完成后， Switch 为 TRUE，轴准备就绪。
InitFlag	BOOL	FALSE	如果该变量为 TRUE，则初始化参数设置完成。

（1） 轴初始化

调用 SMC_AxisInit 指令完成 Axis0 的初始化配置。设定轴类型为 50：EtherCAT_Position。SMC_AxisInit 的指令使用方法参见其介绍章节。

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
```

```
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

（2） 初始化运动限制参数

```
IF  FALSE = InitFlag THEN
   MaxVel:=20000;(*速度限制*)

   MaxAcc:=50000; (*加速度限制*)

   MaxDec:=50000; (*减速度限制*)

   MaxJerk:=50000; (*加加速度限制*)

   MV1_Vel:=50000; (*运动指令速度*)

   MV1_Acc:=100000; (*运动指令速度*)

   MV1_Dec:=100000; (*运动指令速度*)

   MV1_Jerk:=100000; (*运动指令速度*)

   InitFlag:=TRUE;
END_IF
```

（3） 轴使能

轴初始化完成，轴准备就绪状态 Switch1 为 TRUE 后，调用 MC_Power 完成轴使能。

```
IF  Switch1 THEN
   MC_Power1(
   Enable := PWR1_Enable,
   Axis := Axis0,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
   Error=> PWR1_Err
```

```
    ErrorID=> PWR1_ErrID);
END_IF
```

（4） 投送运动指令

轴使能成功后，上升沿触发 MC_MoveVelocity 运动指令，控制轴运动。

```
IF  PWR1_Status THEN
    MC_MoveVelocity1(
    Execute := MV1_ Exe,
    Velocity := MV1_ Vel,
    Acceleration := MV1_ Acc,
    Deceleration := MV1_ Dcc,
    Jerk := MV1_ Jerk,
    Direction := MV1_ Dir,
    BufferMode := MV1_ Buffer,
    Axis := Axis0,
    InVelocity=> MV1_ InVel,
    Busy=> MV1_ Busy,
    Active=> MV1_ Active,
    CommandAborted=> MV1_ Comd,
    Error=> MV1_ Err,
    ErrorID=> MV1_ Error);
END_IF
```

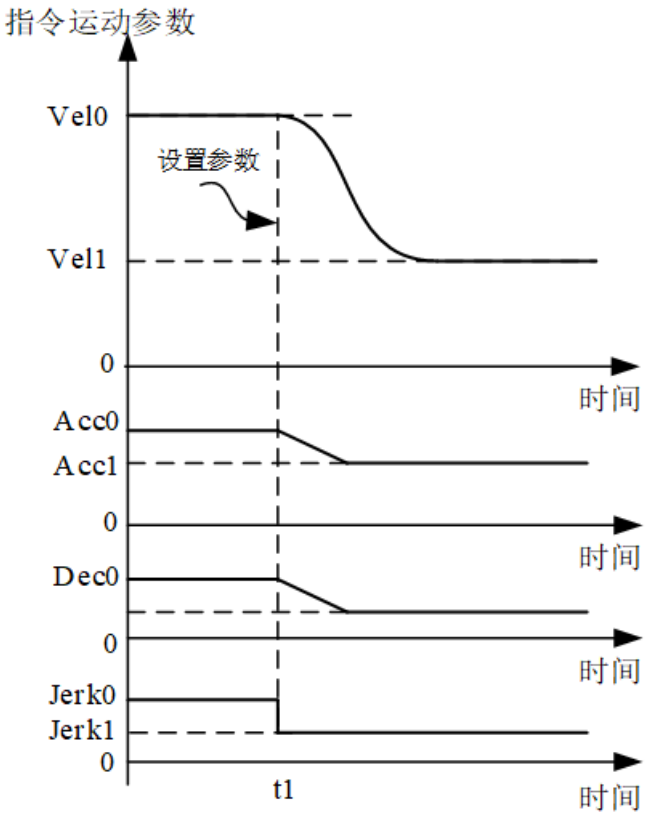
（5） 运动参数限制设置

上升沿触发 SMC_SetDynamicLimits 指令，完成运动参数限制设置。

```
IF  Switch1 THEN
    SMC_SetDynamicLimits1(
    Execute := Dyn_Exe,
    MaxVelocity := Dyn_MaxVel,
    MaxAcceleration := Dyn_MaxAcc,
    MaxDeceleration := Dyn_ MaxDec,
    MaxJerk := Dyn_MaxJerk,
    Axis := Axis0,
    Done=> Dyn_Done,
    Busy=> Dyn_Busy,
    Error=> Dyn_Err,
    ErrorID=> Dyn_ErrID);
```

```
END_IF
```

观察 MC_MoveVelocity 运动指令的运动参数曲线，在 0~t1 时间段，指令运动参数未受限，观察运动指令的速度曲线，运动指令的实际运行速度达到 MC_MoveVelocity 设置的指令速度；在 t1 时间后，SMC_SetDynamicLimits 指令设置的最大速度小于速度 MC_MoveVelocity 运动指令的指令速度，轴运动参数受限制示意图如下所示：



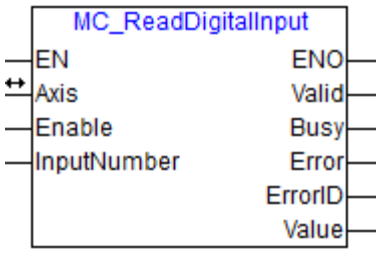
- Vel0、Acc0、Dec0、Jerk0 分别为运动指令的运动速度，加速度，减速度，加加速度
- Vel1、Acc1、Dec1、Jerk1 分别为设置限制参数后的运动速度，加速度，减速度，加加速度

5.15.4 使用注意事项

- 调用该指令时，仅 MaxVelocity 可设为 0，其余输入运动限制参数需为正数。
- 该指令设置最大参数成功后，下降沿不能取消最大参数限制。用户需重新调用该指令设置参数限制值。

5.16 MC_ReadDigitalInput（读取轴数字量输入）

读取伺服设备的数字量输入。



5.16.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Enable	使能	否	-	TRUE/FALSE	BOOL
	InputNumber	Input 编号，表示在 Digtal inputs 对象字典的第几个 bit 位	否	-	0 ~ 31	INT
OUTPUT	Valid	输出有效	否	-	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR
	Value	读取结果	否	-	TRUE/FALSE	BOOL

5.16.2 功能

本指令遵循 CIA402 标准协议，对伺服设备的数字量输入进行读取，该指令的详细介绍如下：

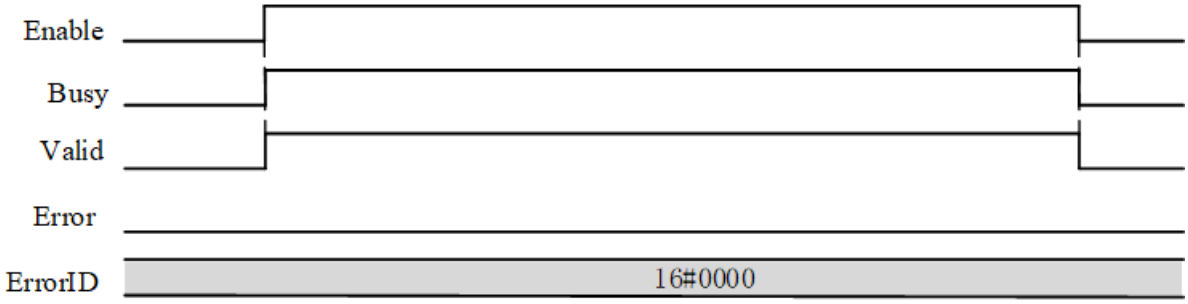
（1） 本指令高电平有效，轴初始化完成之后，输入功能块的 Input 编号，进行数字量输入的读取。

（2） 在 Enable(使能)为 TRUE 时，功能块持续读取数字量输入，当 Enable(使能)为 FALSE 时，功能块引脚恢复默认值。

■ 功能块时序图

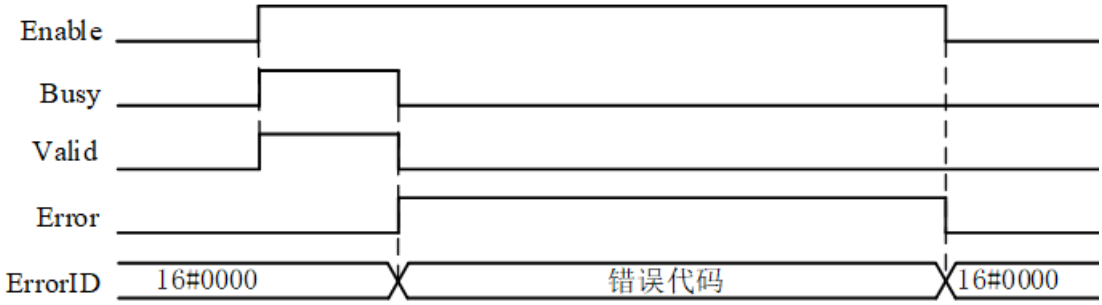
在 Enable(使能)为 TRUE 时，Busy(忙标志)和 Valid(输出有效)立即为 TRUE，若是数字量输入有效则 Value(读取结果)为 TRUE，Enable(使能)为 FALSE 时，功能块引脚恢复默认值。

（1） 正常状态下时序图



（2） 异常状态下时序图

发生异常时，Error(错误)为 TRUE，其他引脚为 FALSE。



5.16.3 示例

组建 MC_ReadDigitalInput 示例程序读取伺服设备的数字量输入值。

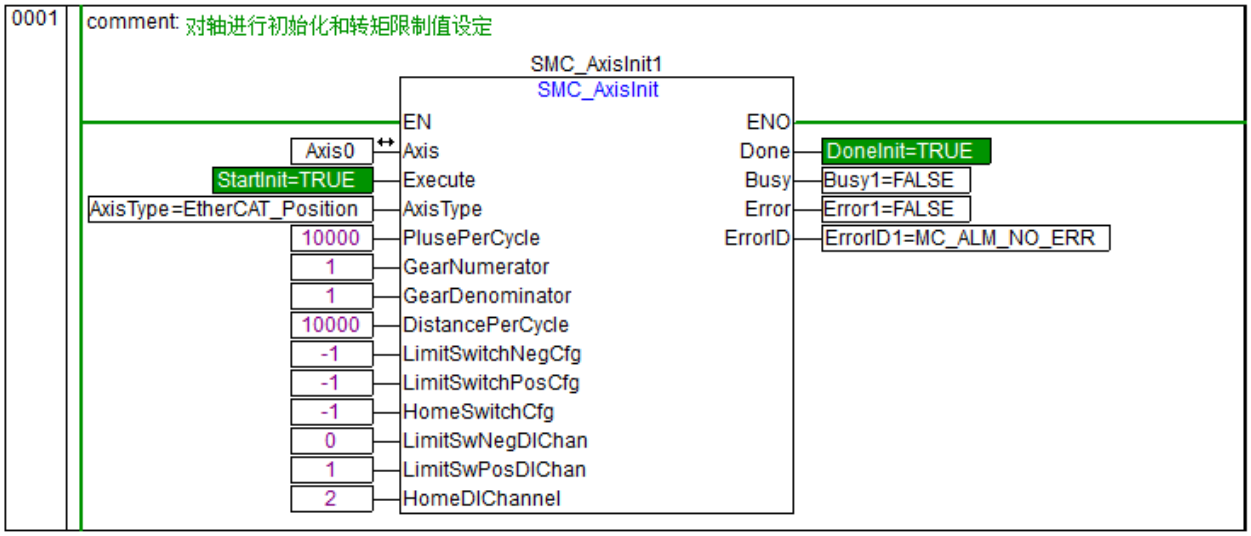
主要变量说明

名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
ReadInputEnable	BOOL	FALSE	开始读取时变为 TRUE
ValidReadInput	BOOL	FALSE	读取成功时变为 TRUE
BusyReadInput	BOOL	FALSE	读取时变为 TRUE
ValueReadInput	BOOL	FALSE	读取到值后变为 TRUE
ErrorReadInput	BOOL	FALSE	出错时变为 TRUE

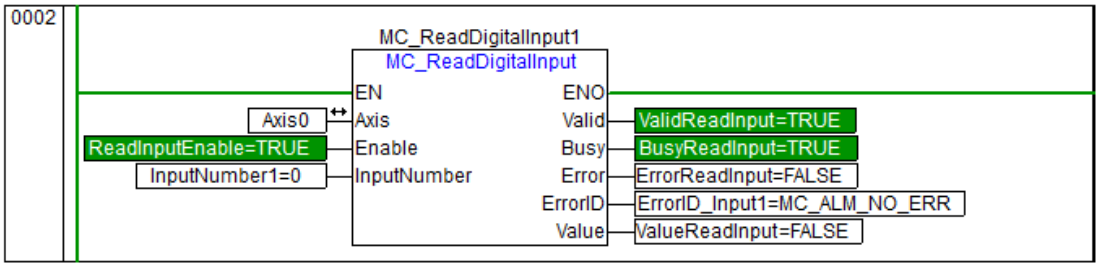
■ 梯形图程序

利用梯形图组建程序，输入对应的对象字典编号。将 MC_ReadDigitalInput 的 Enable(使能)置为 TRUE，示例程序如下：

（1） 在进行读取轴的数字量输入之前，首先要对轴进行初始化。



（2） 初始化完成之后，输入对应的对象字典位数，进行数字量输入读取。



■ ST 语言程序

（1） 初始化轴号和轴类型设置

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
```

```
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

（2） 读取轴的数字量输入

```
MC_ReadDigitalInput1(
Enable := ReadInputEnable,
InputNumber := 0,
Axis := Axis0,
Valid=>ValidReadInput,
Busy=>BusyReadInput,
Error=>ErrorReadInput,
ErrorID=>ErrorID_Input1,
Value=>ValueReadInput);
```

5.16.4 使用注意事项

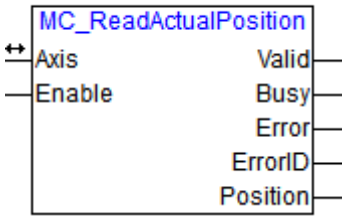
- 本指令不支持虚拟轴，支持 EtherCAT 总线轴。
- 使用本指令之前，需要映射 Digtial Inputs 对应的 PDO（0x60FD）。

5.16.5 参见

- 初始化模块参见初始化章节介绍。
- 若是有错误产生，参见附录功能块错误码介绍。

5.17 MC_ReadActualPosition（读取轴实际位置）

该指令读取轴实际位置。



5.17.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Enable	使能	否	-	TRUE/FALSE	BOOL
OUTPUT	Valid	输出有效	否	-	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR
	Position	读取结果	否	-	正值/负值/0	LREAL

5.17.2 功能

本指令读取轴的实际位置。

■ 指令功能详情

（1） 本指令高电平有效。输入参数合法，轴无异常时，给定 Enable 为高电平 TRUE 时，输出参数 Valid 置高电平，有效，指令持续读取轴的实际位置。

（2） 本指令支持的轴类型有：

□ Virtual（0）：虚轴

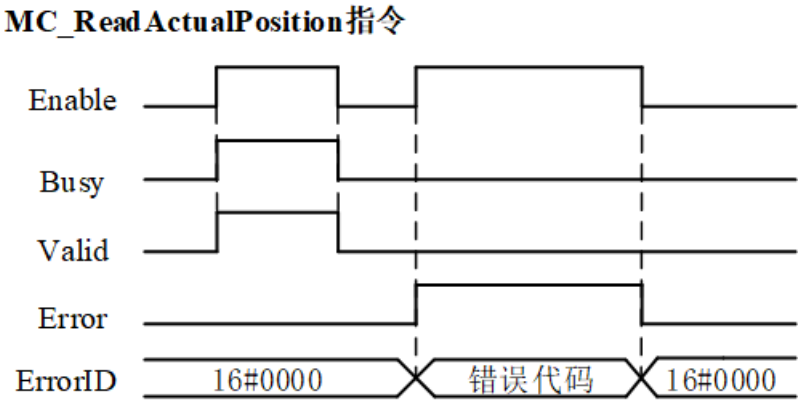
□ EtherCAT_Position（50）：EtherCAT 总线连接轴，位置控制

□ EtherCAT_Speed（51）：EtherCAT 总线连接轴，速度控制

□ EtherCAT_Torque（52）：EtherCAT 总线连接轴，转矩控制

■ 时序图

下述为指令执行时的正常、异常时序图。



- 指令使能引脚高电平，无异常错误时，Busy 和 Valid 引脚立即置 TRUE，Position 读出轴的实际位置。
- 指令执行过程中，异常错误发生时，指令输出引脚 Busy 和 Valid 置为 FLASE，输出结果 Position 保持当前值，Error 置为 True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Enable 变为低电平时，所以输出引脚恢复为默认值。

5.17.3 示例

运行指令 MC_Movelocity，控制轴持续运动，调用 MC_ReadActualPosition 读取轴实际位置。

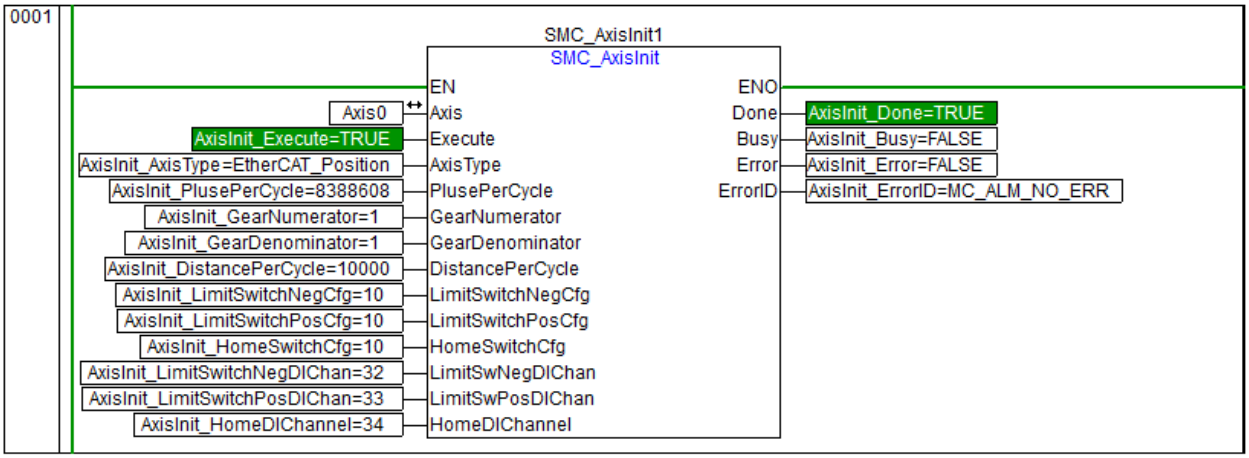
■ 梯形图程序

主要变量说明

变量名称	变量类型	初始值	注释
Axis0	AXIS_REF	-	轴 0 的轴变量
Power1_Status	BOOL	FALSE	轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
ReadActPos_Valid	BOOL	FALSE	轴读取实际位置有效时的变量。读取轴位置成功时，该变量变为 TRUE。
ReadActPos_Pos	LREAL	-	轴读取实际位置的变量。
Switch1	BOOL	FALSE	轴准备就绪的状态变量，若 Switch 为 TRUE 即轴准备就绪。

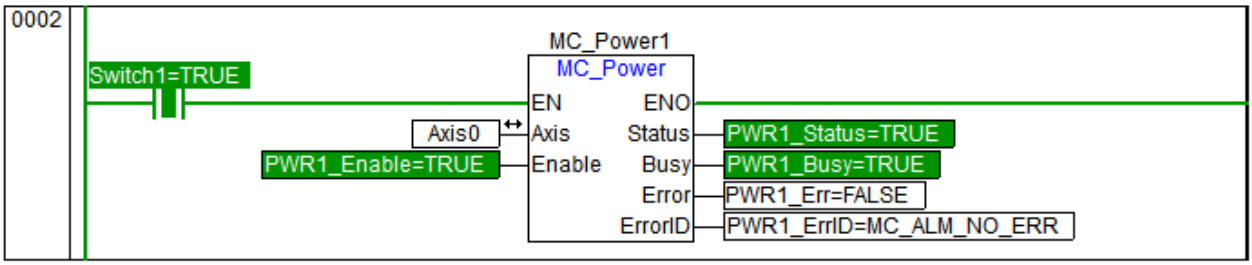
（1） 轴初始化

Axis0 的 AxisID 设为 0，轴类型设置为 50：EtherCAT_Position，调用轴初始化 SMC_AxisInit 指令完成 Axis0 的初始化配置，SMC_AxisInit 的指令使用方法参见其介绍章节。

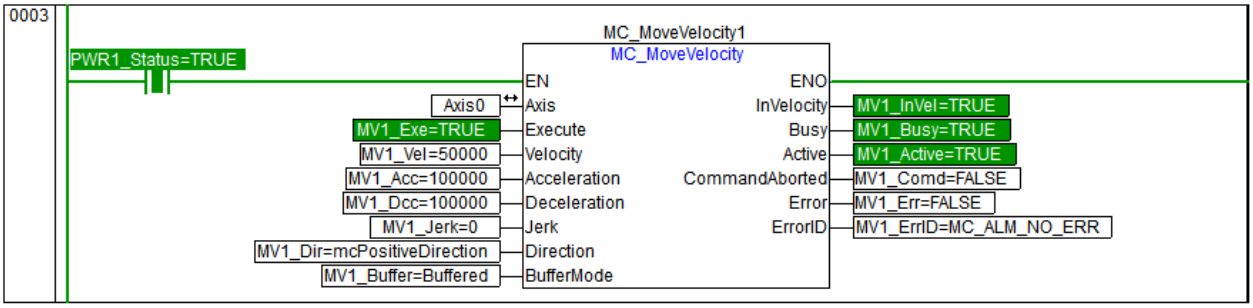


（2） 轴使能

轴初始化完成，轴准备就绪状态 Switch1 为 TRUE 后，调用 MC_Power 完成轴使能。

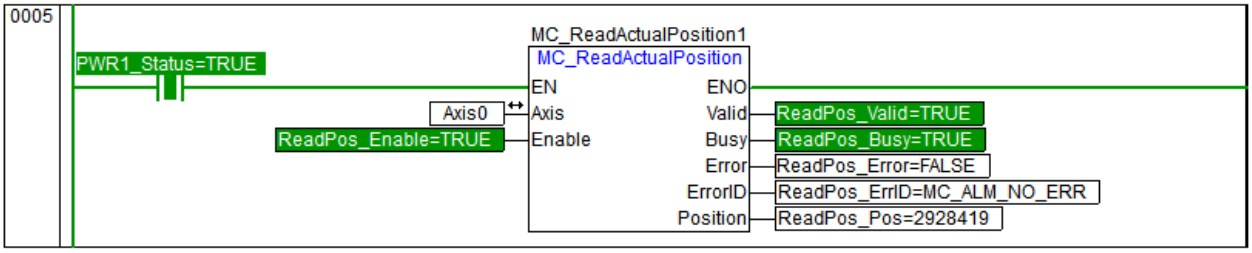


（3） 0003 节中，轴使能成功后，上升沿触发 MC_MoveVelocity 运动指令，控制轴运动。



（4） 执行读取轴位置指令

0005 节中，执行 MC_ReadActualPosition 指令，开始读取轴的实际位置，ReadPos_Valid 为 TRUE，读取位置有效，ReadPos_Pos 为读取的轴实际位置。



■ ST 语言程序

（1） ST 语言程序的主要变量同 LD 语言程序的主要变量。

（2） 设定 Axis0 的轴号

```
Axis0.AxisID:=0;
```

（3） 轴初始化

调用 SMC_AxisInit 指令完成 Axis0 的初始化配置，设定轴类型为为 50：EtherCAT_Position。SMC_AxisInit 的指令使用方法参见其说明章节。

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

（4） 轴使能

轴初始化完成，轴准备就绪状态 Switch1 为 TRUE 后，调用 MC_Power 完成轴使能。

```
IF  Switch1 THEN
```

```
  MC_Power1(
  Enable := PWR1_Enable,
  Axis := Axis0,
  Status=> PWR1_Status,
  Busy=> PWR1_Busy,
  Error=> PWR1_Err
  ErrorID=> PWR1_ErrID);
END_IF
```

（5） 轴使能成功后，执行 MC_MoveVelocity 运动指令，控制轴运动。

```
IF  PWR1_Status THEN
  MC_MoveVelocity1(
  Execute := MV1_ Exe,
  Velocity := MV1_ Vel,
  Acceleration := MV1_ Acc,
  Deceleration := MV1_ Dcc,
  Jerk := MV1_ Jerk,
  Direction := MV1_ Dir,
  BufferMode := MV1_ Buffer,
  Axis := Axis0,
  InVelocity=> MV1_ InVel,
  Busy=> MV1_ Busy,
  Active=> MV1_ Active,
  CommandAborted=> MV1_ Comd,
  Error=> MV1_ Err,
  ErrorID=> MV1_ Error);
END_IF
```

（6） 执行 MC_ReadActualPosition 指令，读取轴实际位置。

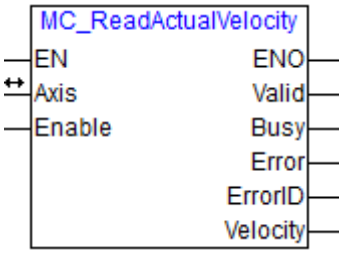
```
IF  PWR1_Status THEN
  MC_ReadActualPosition1(
  Enable := ReadPos_Enable,
  Axis := Axis0,
  Valid=> ReadPos_Valid,
  Busy=> ReadPos_Busy,
  Error=> ReadPos_Err,
  ErrorID=> ReadPos_ErrID,
  Position=> ReadPos_Pos);
END_IF
```

5.17.4 使用注意事项

- 指令执行过程中，发生异常错误时，输出结果 Position 保持当前值。

5.18 MC_ReadActualVelocity（读取轴实际速度）

读取轴的实际速度。



5.18.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Enable	使能	否	-	TRUE/FALSE	BOOL
OUTPUT	Valid	输出有效	否	-	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR
	Velocity	读取结果	否	-	正值/负值/0	LREAL

5.18.2 功能

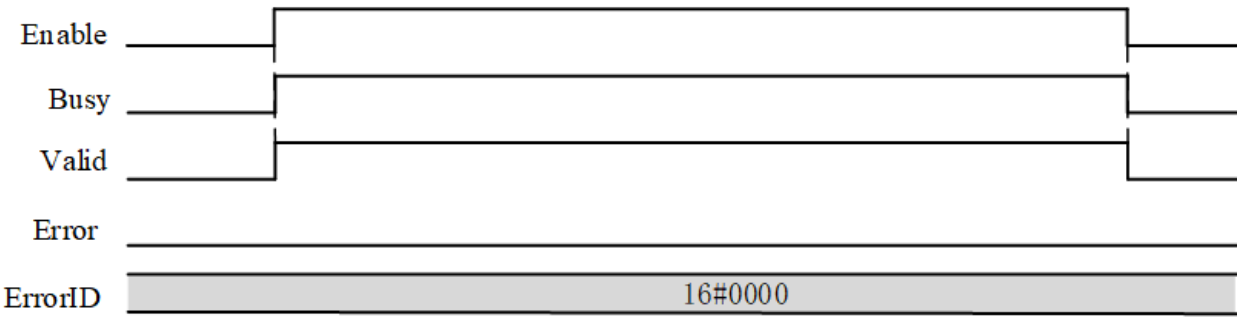
本指令读取轴的实际速度，该指令的详细介绍如下：

（1） 本指令高电平有效，轴初始化完成之后，可以直接进行实际速度的读取。

（2） 在 Enable(使能)为 TRUE 时，功能块持续读取实际速度，当 Enable(使能)为 FALSE 时，功能块引脚恢复默认值。

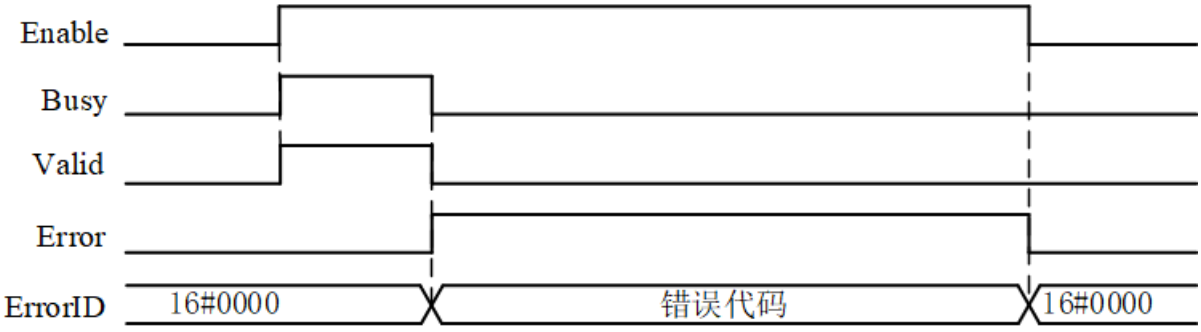
- 功能块时序图

（1） 正常状态下时序图



（2） 异常状态下时序图

发生异常时，Error(错误)为 TRUE，其他引脚为 FALSE。



5.18.3 示例

组建 MC_ReadActualVelocity 示例程序读取轴的实际速度。

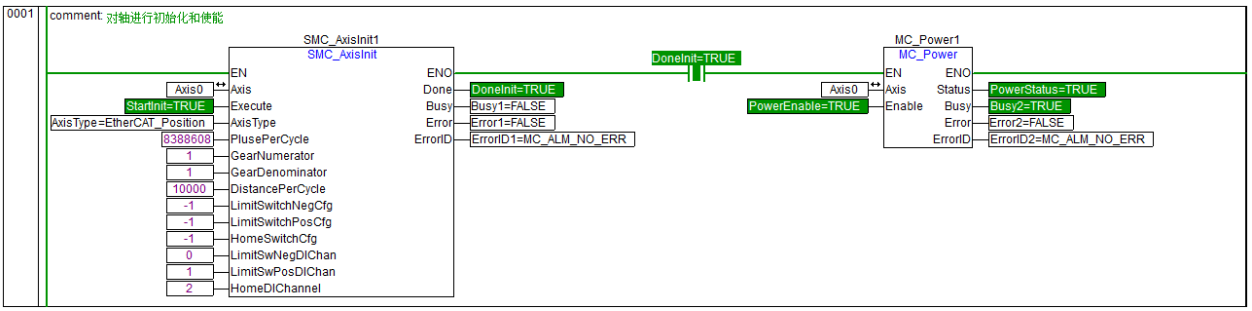
主要变量说明

名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
ReadVelEnable	BOOL	FALSE	开始读取时变为 TRUE
ValidReadVel	BOOL	FALSE	读取成功时变为 TRUE
BusyReadVel	BOOL	FALSE	读取时变为 TRUE
ValueReadVel	LREAL		读取到的实际速度
ErrorReadVel	BOOL	FALSE	出错时变为 TRUE

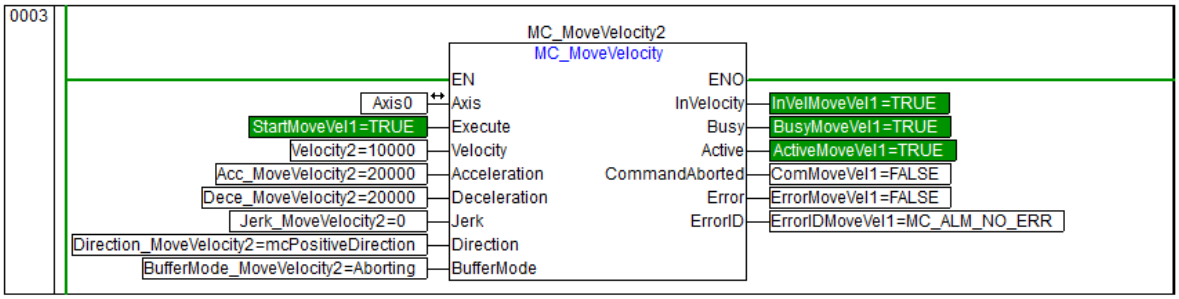
■ 梯形图程序

利用梯形图组建程序，输入对应的对象字典变化。将 MC_ReadActualVelocity 的 Enable(使能)置为 TRUE，示例程序如下：

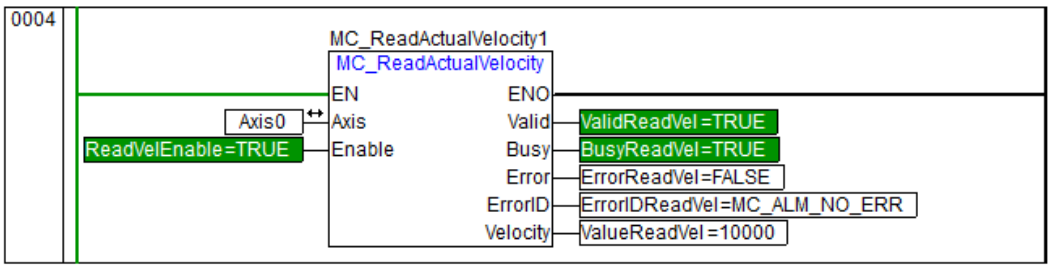
（1） 首先对轴进行初始化，对轴的类型和各种参数进行初始化设置，待轴初始化完成之后，对轴进行使能，进入伺服使能的 ON 状态时，轴可以实现运动控制。



（2） 运行 MC_MoveVelocity 指令，输入 Velocity 为 10000，控制轴进行运动。



（3） 调用 MC_ReadActualVelocity 指令读取其速度，可以看见读取的实际速度值。



■ ST 语言程序

（1） 初始化轴号和轴类型设置

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
```

```
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
(*对轴进行使能*)
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
   Busy=>Busy2,
   Error=>Error2,
   ErrorID=>ErrorID2);
END_IF
```

（2） 输入运动相关参数，启动速度控制指令

```
(*启动速度控制指令*)
MC_MoveVelocity1(
Execute := StartMoveVel,
Velocity := 10000,
Acceleration := 20000,
Deceleration := 20000,
Jerk := 0,
Direction := 0,
BufferMode := 0,
Axis := Axis0,
```

```
InVelocity=>InVelMoveVel,
Busy=>BusyMoveVel,
Active=>ActiveMoveVel,
CommandAborted=>ComMoveVel,
Error=>ErrorMoveVel,
ErrorID=>ErrorIDMoveVel);
```

（3） 读取轴的实际速度

```
MC_ReadActualVelocity1(
Enable := ReadVelEnable,
Axis := Axis0,
Valid=>ValidReadVel,
Busy=>BusyReadVel,
Error=>ErrorReadVel,
ErrorID=>ErrorIDReadVel,
Velocity=>ValueReadVel);
```

5.18.4 使用注意事项

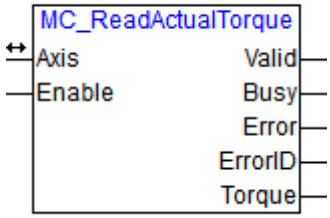
- 当轴号超限时会有报错产生。
- 该指令支持虚拟轴和 Ethercat 总线轴。

5.18.5 参见

- 关于 MC_MoveVelocity 等相关运动指令，可以参见 MC_MoveVelocity 等运动指令章节。
- 关于 ErrorID 相关详细错误，参见附录功能块错误码。

5.19 MC_ReadActualTorque（读取轴实际力矩）

该指令读取遵循 CIA402 标准的伺服设备的扭矩值。



5.19.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Enable	使能	否	-	TRUE/FALSE	BOOL
OUTPUT	Valid	输出有效	否	-	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR
	Torque	读取结果 单位：1%额定扭矩	否	-	正值/负值/0	LREAL

5.19.2 功能

本指令读取遵循 CIA402 标准的伺服设备的扭矩值。

■ 指令功能详情

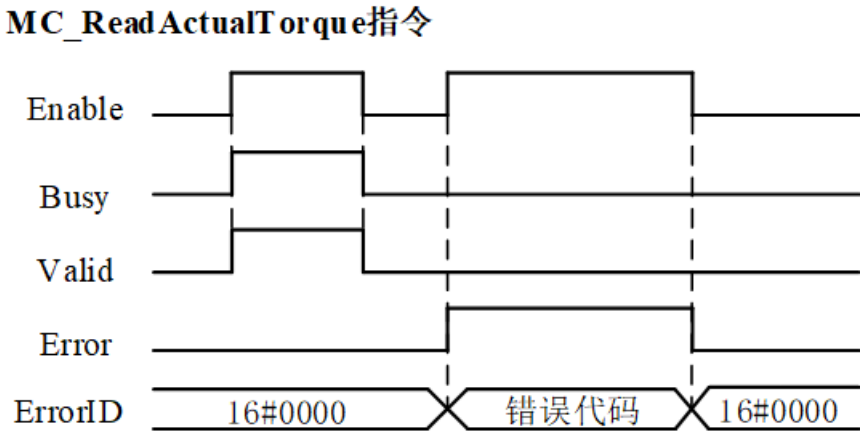
（1） 本指令高电平有效。输入参数合法，轴无异常时，给定 Enable 为高电平 TRUE 时，输出参数 Valid 置高电平，有效，指令持续读取伺服设备的扭矩，指令读取的结果为额定力矩的 1%。

（2） 本指令支持的轴类型有：

- □ EtherCAT_Position（50）：EtherCAT 总线连接轴，位置控制
- □ EtherCAT_Speed（51）：EtherCAT 总线连接轴，速度控制
- □ EtherCAT_Torque（52）：EtherCAT 总线连接轴，转矩控制

■ 时序图

下述为指令执行时的正常、异常时序图。



- 指令使能引脚高电平，无异常错误时，Busy 和 Valid 引脚立即置 TRUE，Torque 读出伺服设备的扭矩值。
- 指令执行过程中，异常错误发生时，指令输出引脚 Busy 和 Valid 置为 FLASE，输出结果 Torque 为 0，Error 置为 True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Enable 变为低电平时，所以输出引脚恢复为默认值。

5.19.3 示例

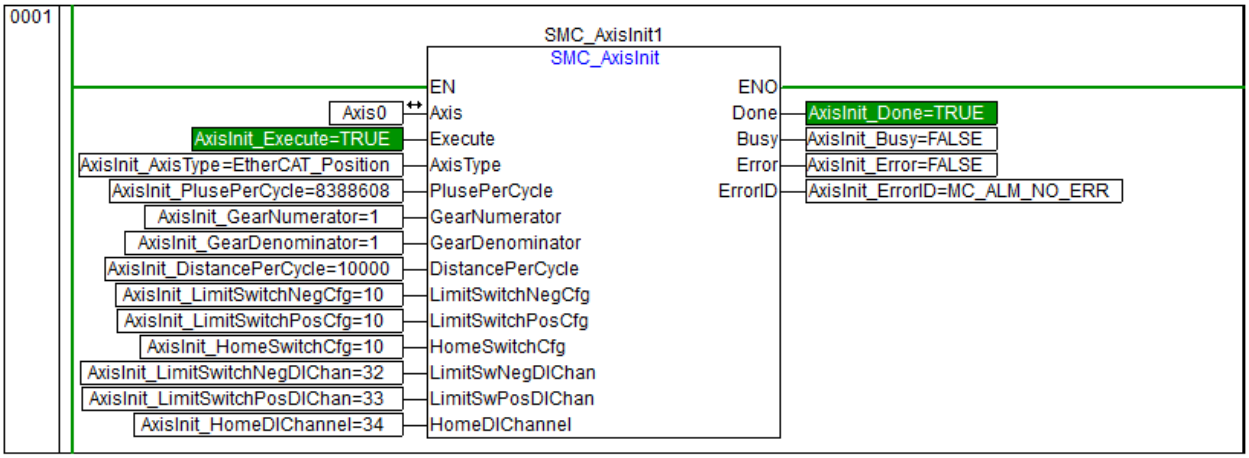
■ 梯形图程序

主要变量说明

变量名称	变量类型	初始值	注释
Axis0	AXIS_REF	-	轴 0 的轴变量
Power1_Status	BOOL	FALSE	轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
ReadTor_Valid	BOOL	FALSE	轴读取实际力矩有效时的变量。读取轴实际力矩成功时，该变量变为 TRUE。
ReadTor_Tor	LREAL	-	轴读取实际力矩的变量。输出为读取轴的实际力矩，单位：1%额定扭矩。
Switch1	BOOL	FALSE	轴准备就绪的状态变量，若 Switch 为 TRUE 即轴准备就绪。

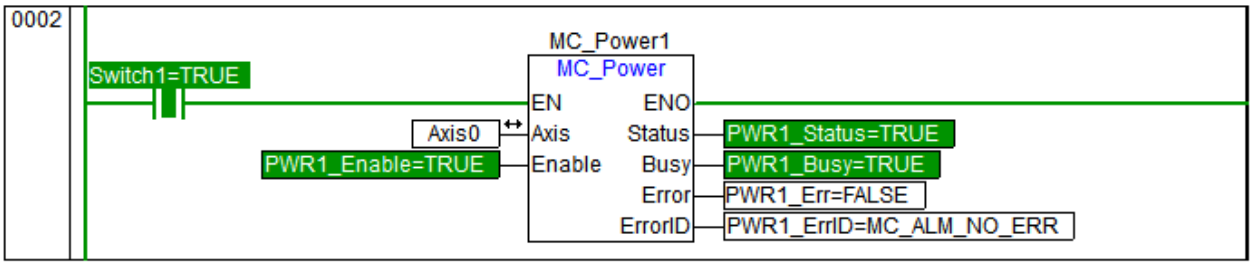
（1） 轴初始化

Axis0 的 AxisID 设为 0，轴类型设置为 50：EtherCAT_Position，调用轴初始化 SMC_AxisInit 指令完成 Axis0 的初始化配置，SMC_AxisInit 的指令使用方法参见其介绍章节。

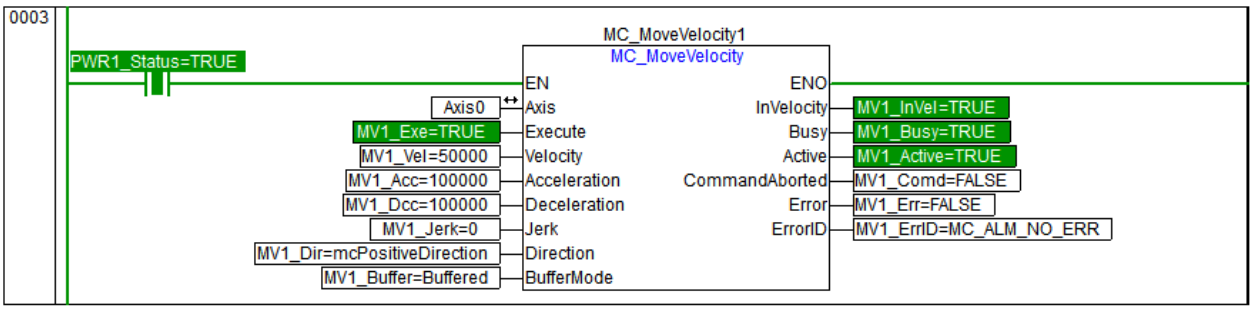


（2） 轴使能

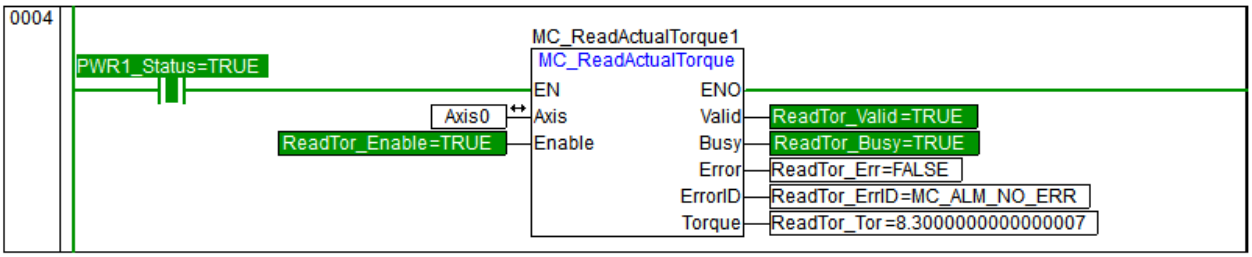
轴初始化完成，轴准备就绪状态 Switch1 为 TRUE 后，调用 MC_Power 完成轴使能。



（3） 0003 节中，轴使能成功后，上升沿触发 MC_MoveVelocity 运动指令，控制轴运动。



（4） 0004 节中，高电平触发 MC_ReadActualTorque 的 Enable 引脚，开始读取轴的实际力矩。ReadTor_Valid 为 TRUE，读取力矩有效， ReadPos_Tor 为额定力矩的 1%。



■ ST 语言程序

ST 语言程序的主要变量同 LD 语言程序的主要变量。

（1） 设定 Axis0 的轴号

```
Axis0.AxisID:=0;
```

（2） 轴初始化

调用 SMC_AxisInit 指令完成 Axis0 的初始化配置，设定轴类型为为 50：EtherCAT_Position。SMC_AxisInit 的指令使用方法参见其介绍章节。

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

（3） 轴使能

轴初始化完成，轴准备就绪状态 Switch1 为 TRUE 后，调用 MC_Power 完成轴使能。

```
IF  Switch1 THEN
    MC_Power1(
    Enable := PWR1_Enable,
    Axis := Axis0,
    Status=> PWR1_Status,
    Busy=> PWR1_Busy,
    Error=> PWR1_Err
    ErrorID=> PWR1_ErrID);
END_IF
```

（4） 轴使能成功后，上升沿触发 MC_MoveVelocity 运动指令，控制轴运动。

```
IF  PWR1_Status THEN
    MC_MoveVelocity1(
    Execute := MV1_ Exe,
    Velocity := MV1_ Vel,
    Acceleration := MV1_ Acc,
    Deceleration := MV1_ Dcc,
    Jerk := MV1_ Jerk,
    Direction := MV1_ Dir,
    BufferMode := MV1_ Buffer,
    Axis := Axis0,
    InVelocity=> MV1_ InVel,
    Busy=> MV1_ Busy,
    Active=> MV1_ Active,
    CommandAborted=> MV1_ Comd,
    Error=> MV1_ Err,
    ErrorID=> MV1_ Error);
END_IF
```

（5） 高电平触发 MC_ReadActualTorque 指令，读取轴实际力矩。

```
IF  MC_Power1. Status THEN
    MC_ReadActualTorque1(
    Enable := ReadTor_Enable,
    Axis := Axis0,
    Valid=> ReadTor_Valid,
    Busy=> ReadTor_Busy,
    Error=> ReadTor_Err,
    ErrorID=> ReadTor_ErrID,
    Torque=> ReadTor);
```

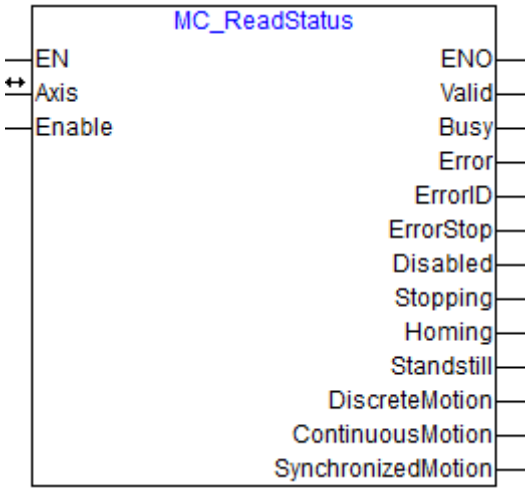
```
END_IF
```

5.19.4 使用注意事项

- 调用指令的伺服设备必须为遵循 CIA402 标准的伺服设备。
- 指令读取伺服设备反馈扭矩（16#6077）对应的 PDO（0x60FD），故伺服设备从站需配置相关 PDO。

5.20 MC_ReadStatus（读取轴状态）

读取轴 PLCopen 状态机的状态。



5.20.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Enable	使能	否	-	TRUE/FALSE	BOOL
OUTPUT	Valid	输出有效	否	-	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR
	ErrorStop	为 TRUE 时表示轴处于 ErrorStop 状态	是	FALSE	TRUE/FALSE	BOOL
	Disabled	为 TRUE 时表示轴处于 Disabled 状态	是	FALSE	TRUE/FALSE	BOOL

	Stopping	为 TRUE 时表示轴处于 Stopping 状态	是	FALSE	TRUE/FALSE	BOOL
	Homing	为 TRUE 时表示轴处于 Homing 状态	是	FALSE	TRUE/FALSE	BOOL
	Standstill	为 TRUE 时表示轴处于 Standstill 状态	是	FALSE	TRUE/FALSE	BOOL
	DiscreteMotion	为 TRUE 时表示轴处于 DiscreteMotion 状态	是	FALSE	TRUE/FALSE	BOOL
	ContinuousMotion	为 TRUE 时表示轴处于 ContinuousMotion 状态	是	FALSE	TRUE/FALSE	BOOL
	SynchronizedMotion	为 TRUE 时表示轴处于 SynchronizedMotion 状态	是	FALSE	TRUE/FALSE	BOOL

5.20.2 功能

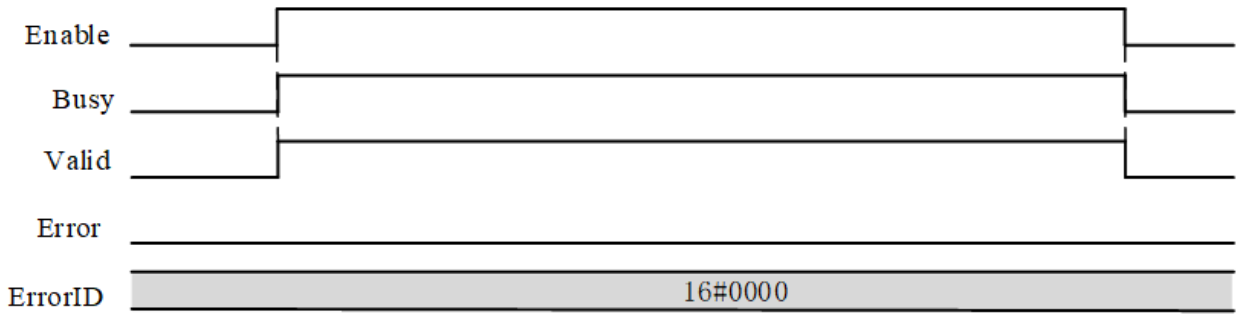
本指令读取轴 PLCopen 状态机的状态，该指令的详细介绍如下：

（1） 本指令高电平有效，轴初始化完成之后，可以直接进行轴 PLCopen 状态机的状态读取。

（2） 在 Enable(使能)为 TRUE 时，功能块持续读取轴 PLCopen 状态机的状态，当 Enable(使能)为 FALSE 时，功能块引脚恢复默认值。

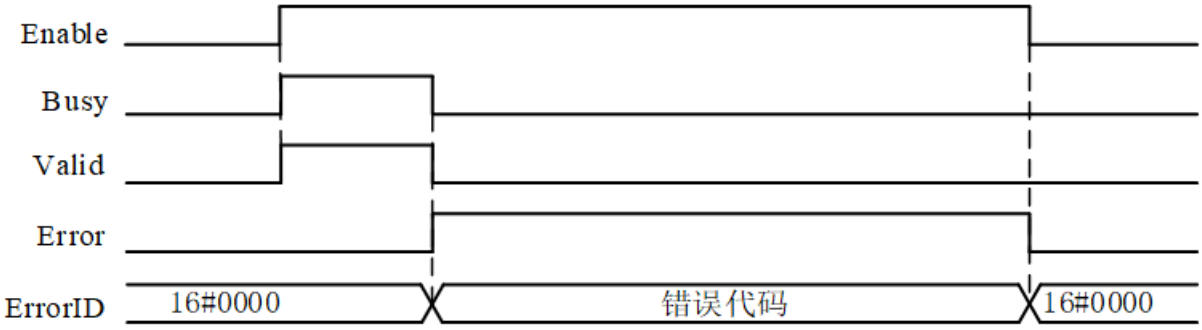
■ 功能块时序图

（1） 正常状态下时序图



（2） 异常状态下时序图

发生异常时，Error(错误)为 TRUE，其他引脚为 FALSE。



5.20.3 示例

组建 MC_ReadStatus 示例程序读取轴 PLCopen 状态机的状态。

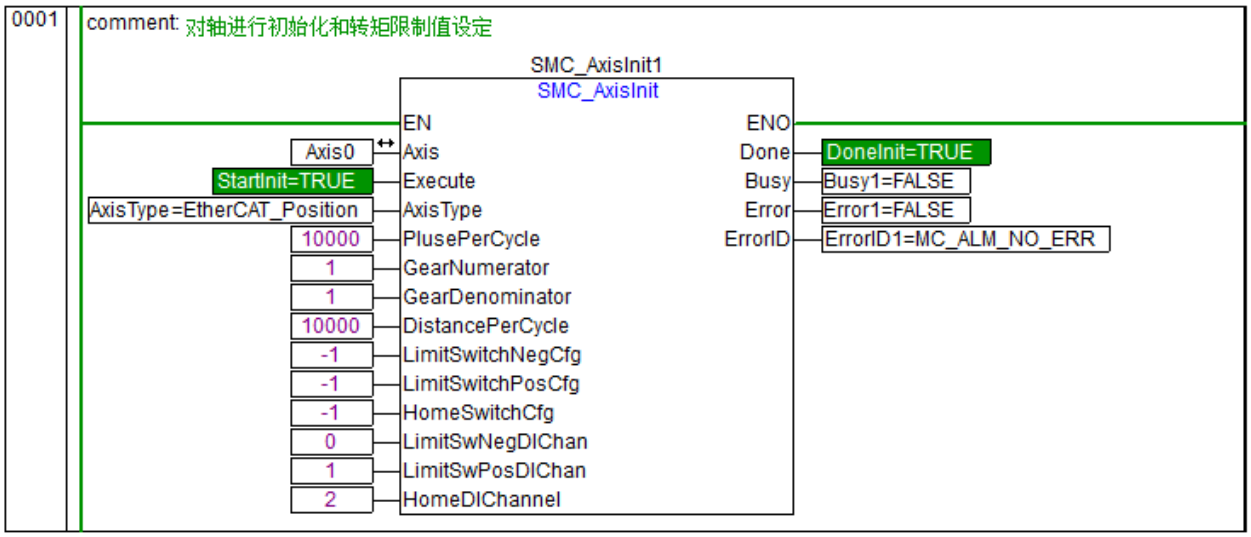
■ 主要变量

名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
ReadStatusEnable	BOOL	FALSE	开始读取时变为 TRUE
ValidReadStatus	BOOL	FALSE	读取成功时变为 TRUE
BusyReadStatus	BOOL	FALSE	读取时变为 TRUE
ErrorReadStatus	BOOL	FALSE	出错时变为 TRUE

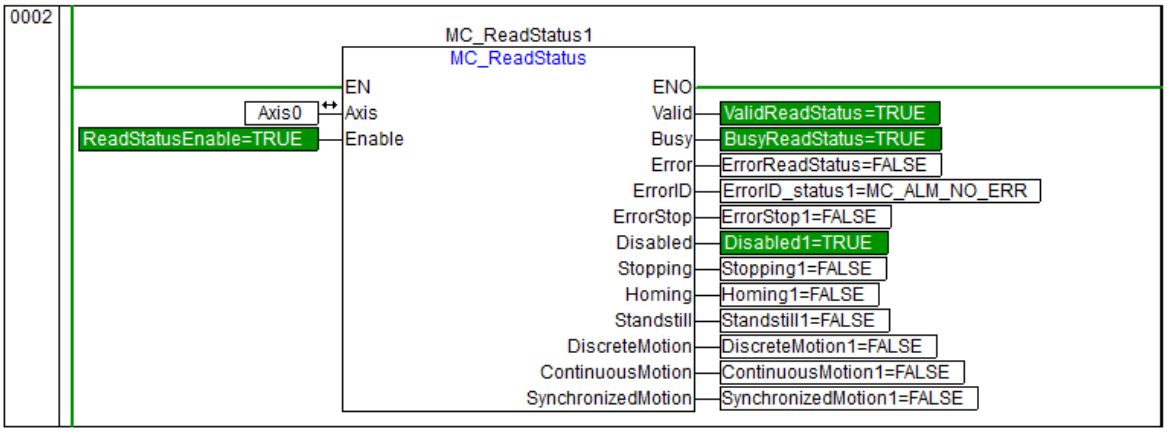
■ 梯形图程序

利用梯形图组建程序。将 MC_ ReadStatus 的 Enable(使能)置为 TRUE，示例程序如下：

（1） 在进行轴的 PLCopen 状态机的状态读取之前，首先要对轴进行初始化。



（2） 初始化完成之后，可以直接进行轴 PLCopen 状态机的状态的读取。



■ ST 语言程序

（1） 初始化轴号和轴类型设置

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
```

```
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

（2） 进行轴 PLCopen 状态机状态的读取

```
MC_ReadStatus1(
Enable := ReadStatusEnable,
Axis := Axis0,
Valid=>ValidReadStatus,
Busy=>BusyReadStatus,
Error=>ErrorReadStatus,
ErrorID=>ErrorID_status1,
ErrorStop=>ErrorStop1,
Disabled=>Disabled1,
Stopping=>Stopping1,
Homing=>Homing1,
Standstill=>Standstill1,
DiscreteMotion=>DiscreteMotion1,
ContinuousMotion=>ContinuousMotion1,
SynchronizedMotion=>SynchronizedMotion1);
```

5.20.4 使用注意事项

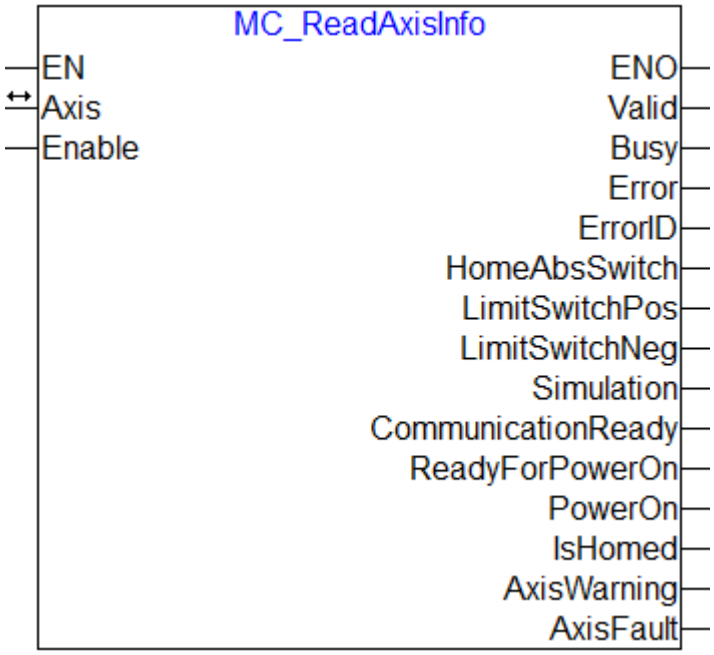
该指令为 Enable 型，Enable 为高电平时功能块一直读取轴状态，不存在重启和多重启动情况。

5.20.5 参见

若是有错误发生，可以参见附录功能块错误码。

5.21 MC_ReadAxisInfo（读取轴信息）

用于读取指定轴的相关信息。



5.21.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Enable	使能	否	-	TRUE/FALSE	BOOL
OUTPUT	Valid	输出有效	否	-	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR
	HomeAbsSwitch	原点开关状态	是	FALSE	TRUE/FALSE	BOOL
	LimitSwitchPos	正向限位开关状态	是	FALSE	TRUE/FALSE	BOOL
	LimitSwitchNeg	负向限位开关状态	是	FALSE	TRUE/FALSE	BOOL
	Simulation	轴处于仿真状态（保留参数，FALSE）	是	FALSE	TRUE/FALSE	BOOL
	CommunicationReady	轴通信状态正常	是	FALSE	TRUE/FALSE	BOOL
	ReadyForPowerOn	轴准备好使能	是	FALSE	TRUE/FALSE	BOOL
	PowerOn	轴使能中	是	FALSE	TRUE/FALSE	BOOL
	IsHomed	是否完成原点回归	是	FALSE	TRUE/FALSE	BOOL
	AxisWarning	轴发生警告	是	FALSE	TRUE/FALSE	BOOL
	AxisFault	轴发生故障	是	FALSE	TRUE/FALSE	BOOL

5.21.2 功能

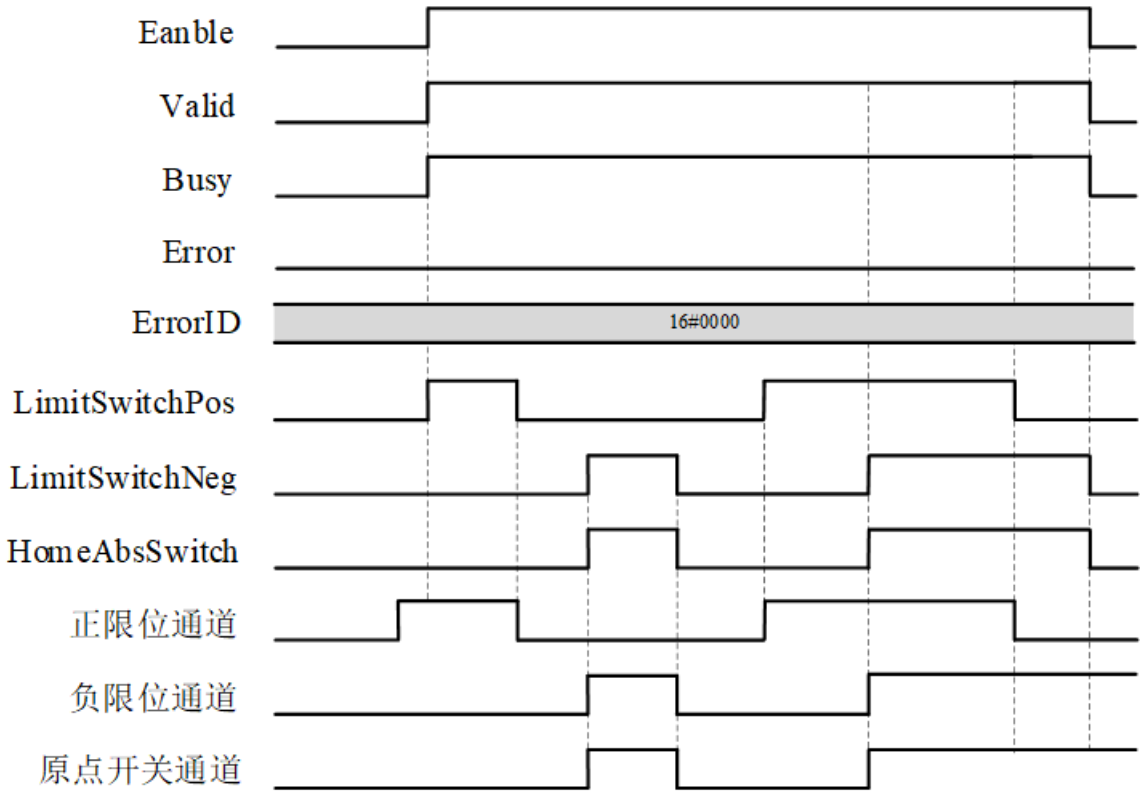
该指令在 Enable 为 TRUE 时读取轴相关信息，输出引脚 Valid 为 TRUE 时，读取的轴信息状态为有效值，否则为无效值。读取过程中若发生错误，输出引脚 Error 置 TRUE，ErrorID 显示错误原因。当 Enable 为 FALSE 时，功能块输出引脚全部无效。

- 输出轴信息说明
  - 输出引脚 HomeAbsSwitch 指示原点回归开关通道的状态，为 TRUE 表示有效，FALSE 表示无效；
  - 输出引脚 LimitSwitchPos、LimitSwitchNeg 指示正负硬限位开关通道的状态，为 TRUE 表示有效，FALSE 表示无效；
  - 输出引脚 Simulation 指示当前轴是否处于仿真模式，为 TRUE 表示轴处于仿真模式，为 FALSE 表示非仿真模式，仿真功能暂不支持，故引脚 Simulation 始终为 FALSE；
  - 输出引脚 CommunicationReady 指示当前轴通信状态是否正常，如果当前轴是 EtherCAT 总线轴并且通信正常，则 CommunicationReady 输出为 TRUE，否则为 FALSE；
  - 输出引脚 ReadyForPowerOn 指示当前轴是否准备好使能，当前轴没有错误并且轴未使能时 ReadyForPowerOn 输出为 TRUE，否则为 FALSE；
  - 输出引脚 PowerOn 指示当前轴是否正在使能中，当前轴没有错误并且轴处于使能中时 PowerOn 输出为 TRUE，否则为 FALSE；
  - 输出引脚 IsHomed 指示当前轴是否完成原点回归，为 TRUE 表示当前轴已完成原点回归，为 FALSE 表示当前轴未完成原点回归；
  - 输出引脚 AxisWarning 指示当前轴是否发生警告，如果当前轴是 EtherCAT 轴并且报警告，则 AxisWarning 输出为 TRUE，否则为 FALSE；
  - 输出引脚 AxisFault 指示当前轴是否发生故障，为 TRUE 表示当前轴发生故障，为 FALSE 表示当前轴无故障。

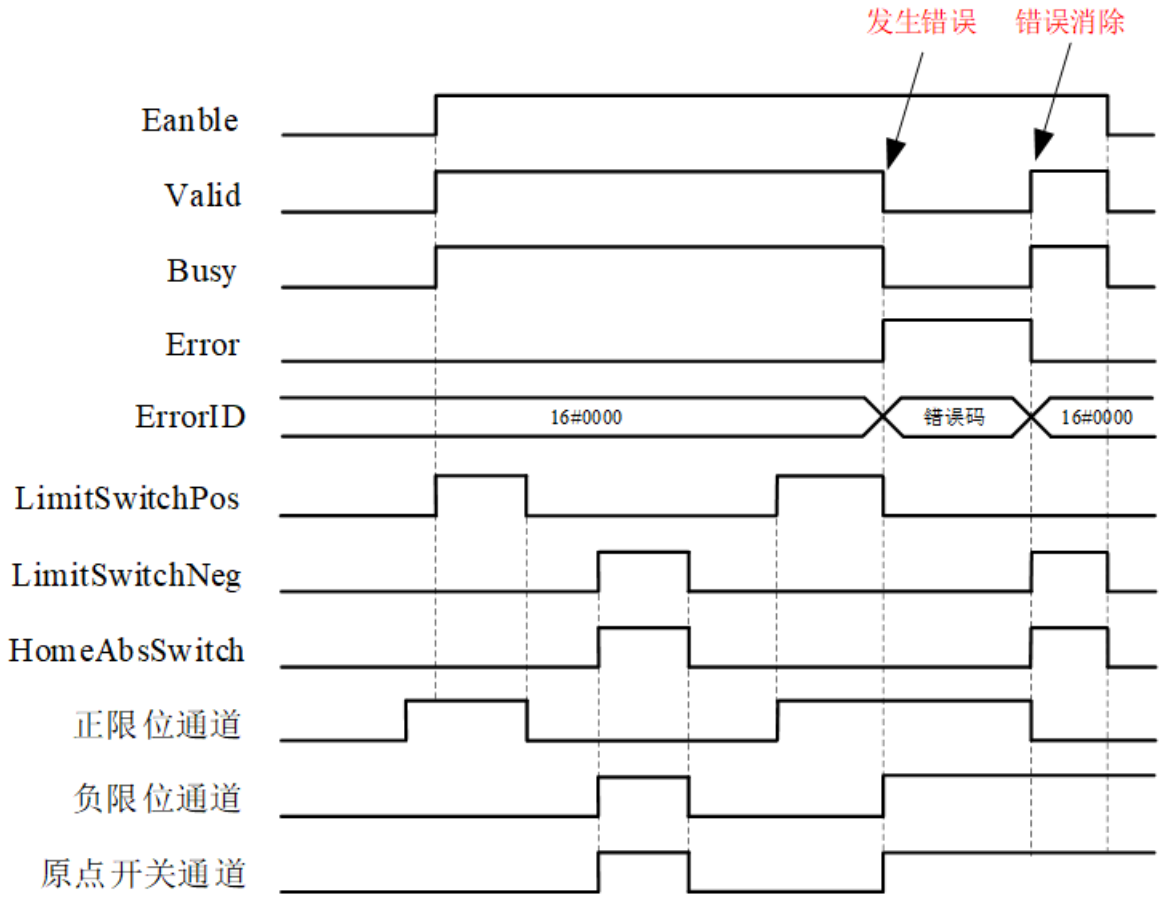
  以上输出引脚不存在强制互斥，输出状态根据轴实际状态信息显示。

- 功能块时序图

（1） 正负硬限位状态、原点开关状态时序图，其它的轴信息状态时序图与此类似。



（2） 获取正负硬限位状态、原点开关状态时发生错误后又恢复错误的时序图。



5.21.3 示例

该示例使用功能块 MC_ReadAxisInfo 获取 Axis0 的轴信息状态。示例通过判断 Axis0 是否有错误以及 Axis0 是否准备好 PowerOn，来调用 MC_Power 使能 Axis0。

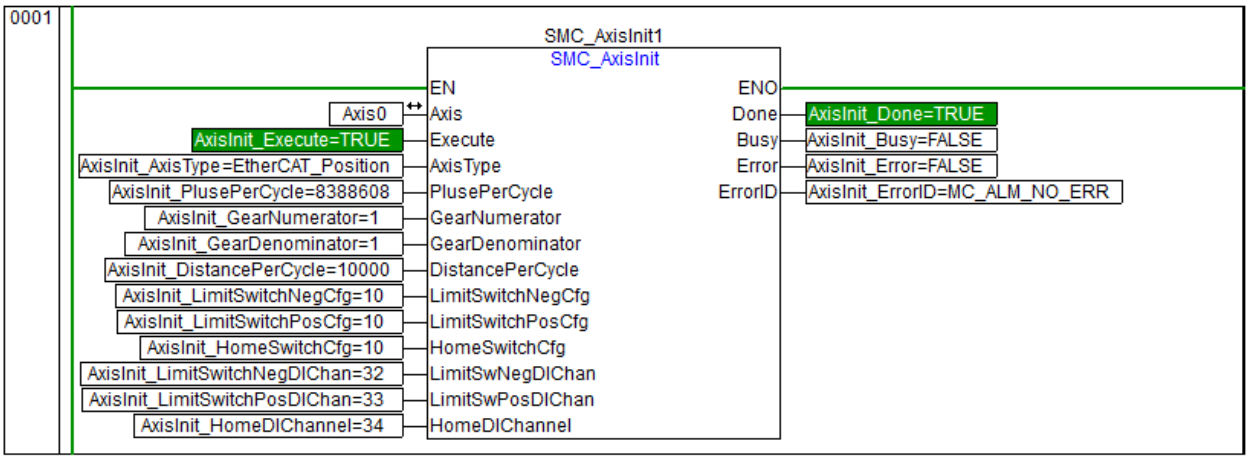
■ 变量

序号	变量名	变量类型	初始值	变量说明
1	MC_ReadAxisInfo	MC_READAXISINFO		读取轴信息指令
2	ReadAxisInfo_En	BOOL	FALSE	使能，上升沿有效
3	ReadAxisInfo_Valid	BOOL	FALSE	读取轴信息有效标志
4	ReadAxisInfo_Busy	BOOL	FALSE	读取轴信息忙标志
5	ReadAxisInfo_Error	BOOL	FALSE	读取轴信息错误标志
6	ReadAxisInfo_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	读取轴信息错误 ID
7	ReadAxisInfo_HomeAbsSwitch	BOOL	FALSE	绝对原点开关状态

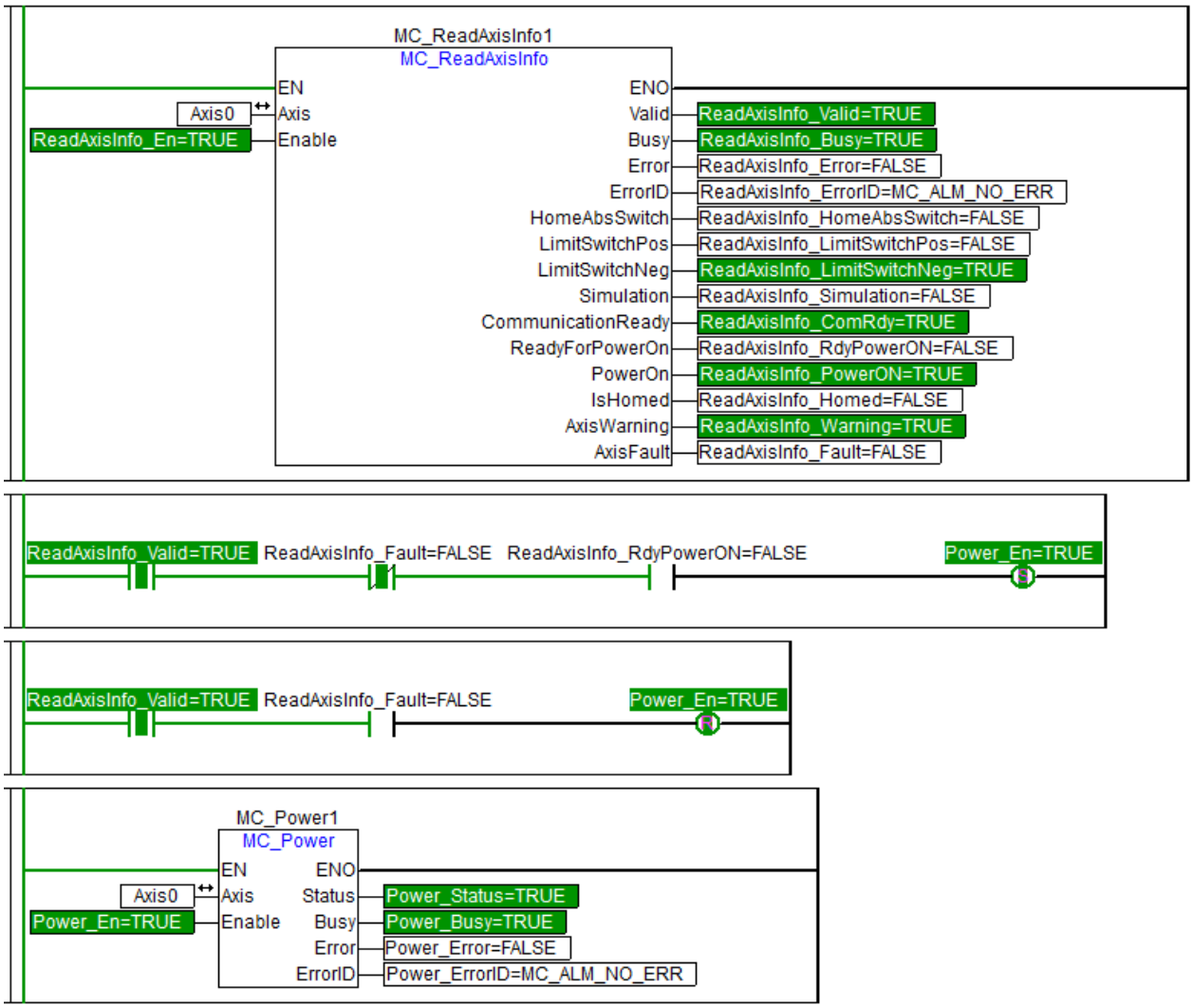
8	ReadAxisInfo_LimitSwitchPos	BOOL	FALSE	正向限位开关状态
9	ReadAxisInfo_LimitSwitchNeg	BOOL	FALSE	负向限位开关状态
10	ReadAxisInfo_Simulation	BOOL	FALSE	仿真模式标志
11	ReadAxisInfo_ComRdy	BOOL	FALSE	通信就绪标志
12	ReadAxisInfo_RdyPowerON	BOOL	FALSE	轴使能就绪标志
13	ReadAxisInfo_PowerON	BOOL	FALSE	轴使能状态标志
14	ReadAxisInfo_Homed	BOOL	FALSE	轴原点回归完成标志
15	ReadAxisInfo_Warning	BOOL	FALSE	读取轴信息警告标志
16	ReadAxisInfo_Fault	BOOL	FALSE	读取轴信息故障标志
17	MC_Power1	MC_POWER		轴使能指令
18	Power_En	BOOL	FALSE	轴使能，电平有效
19	Power_Status	BOOL	FALSE	轴使能状态
20	Power_Busy	BOOL	FALSE	轴使能指令忙标志
21	Power_Error	BOOL	FALSE	轴使能指令错误标志
22	Power_ErrorID	SMC_ERROR	MC_ALM_NO_ERR	轴使能指令错误 ID

■ LD 程序实现

（1） 首先调用 SMC_AxisInit 指令初始化轴相关信息。



（2） 然后调用 MC_ReadAxisInfo 指令获取轴信息，待轴无错误且准备好使能之后，再调用 MC_Power 指令进行轴使能。



■ ST 程序实现

（1） 首先调用 MC_AxisInit 指令初始化轴相关信息。

```
MC_ReadAxisInfo1(
Enable := ReadAxisInfo_En,
Axis := Axis0,
Valid=>ReadAxisInfo_Valid,
Busy=>ReadAxisInfo_Busy,
Error=>ReadAxisInfo_Error,
ErrorID=>ReadAxisInfo_ErrorID,
HomeAbsSwitch=>ReadAxisInfo_HomeAbsSwitch,
LimitSwitchPos=>ReadAxisInfo_LimitSwitchPos,
LimitSwitchNeg=>ReadAxisInfo_LimitSwitchNeg,
Simulation=>ReadAxisInfo_Simulation,
CommunicationReady=>ReadAxisInfo_ComRdy,
ReadyForPowerOn=>ReadAxisInfo_RdyPowerON,
PowerOn=>ReadAxisInfo_PowerON,
```

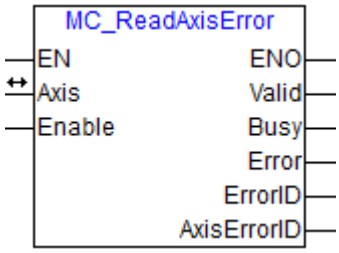
```
IsHomed=>ReadAxisInfo_Homed,
AxisWarning=>ReadAxisInfo_Warning,
AxisFault=>ReadAxisInfo_Fault);
```

（2） 然后调用 MC_ReadAxisInfo 指令获取轴信息，待轴无错误且准备好使能之后，再调用 MC_Power 指令进行轴使能。

```
IF TRUE = ReadAxisInfo_Valid AND FALSE = ReadAxisInfo_Fault AND
TRUE = ReadAxisInfo_RdyPowerON THEN
   Power_En := TRUE;
ELSIF TRUE = ReadAxisInfo_Valid AND TRUE =  ReadAxisInfo_Fault THEN
   Power_En := FALSE;
END_IF;
MC_Power1(
Enable := Power_En,
Axis := Axis0,
Status=>Power_Status,
Busy=>Power_Busy,
Error=>Power_Error,
ErrorID=>Power_ErrorID);
```

5.22 MC_ReadAxisError（读取轴错误）

读取轴错误信息（非功能块错误）。



5.22.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Enable	使能	否	-	TRUE/FALSE	BOOL
OUTPUT	Valid	输出有效	否	-	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL

	ErrorID	指令错误码	是	0	-	SMC_ERROR
	AxisErrorID	轴错误码	是	0	-	DWORD

5.22.2 功能

本指令读取轴错误信息，该指令的详细介绍如下：

（1） 本指令高电平有效，轴初始化完成之后，可以直接进行轴错误信息的读取，如果读取的轴没有错误，则 Valid 为 TRUE，AxisErrorID 为 0，若是轴发生错误则 AxisErrorID 输出对应的轴错误码。

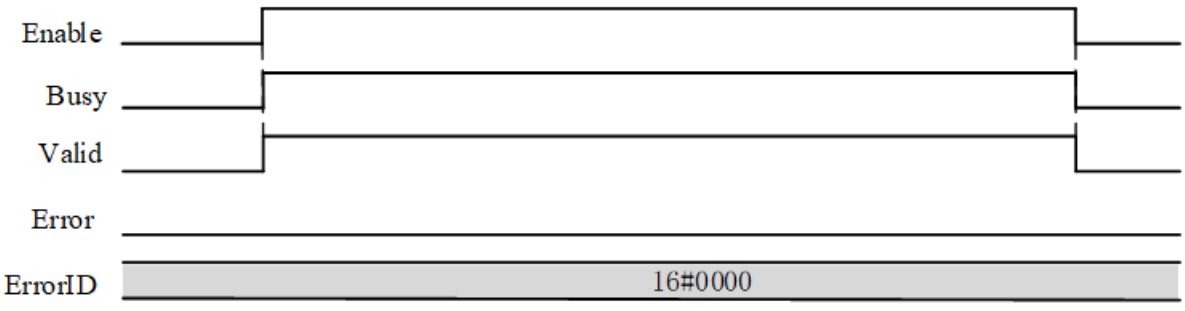
（2） 在 Enable(使能)为 TRUE 时，功能块持续读取轴错误信息，当 Enable(使能)为 FALSE 时，功能块引脚恢复默认值。

（3） 轴错误说明：

错误码	错误类型	错误说明
100001	轴错误	轴有错误
100002	从站离线	从站离线错误
100003	读取失败	读取错误码失败
200000+	系统错误	系统错误
0-65535	轴错误码	参见轴手册，读取的为轴 603F 对象字典错误码

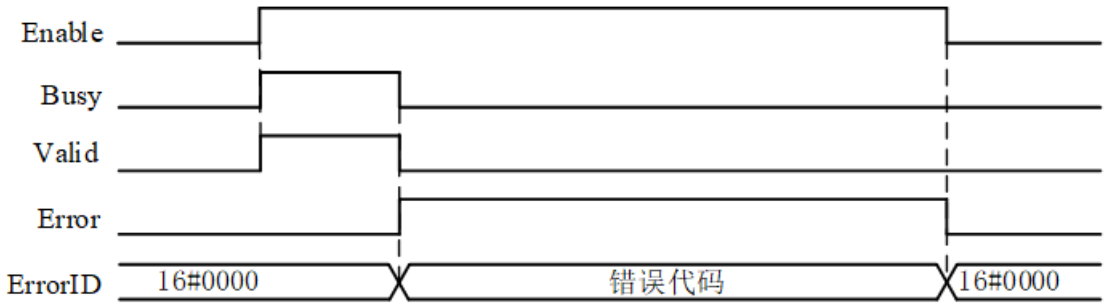
■ 功能块时序图

（1） 正常状态下时序图



（2） 异常状态下时序图

发生异常时，Error(错误)为 TRUE，其他引脚为 FALSE。



5.22.3 示例

组建 MC_ ReadAxisError 示例程序读取轴错误信息。

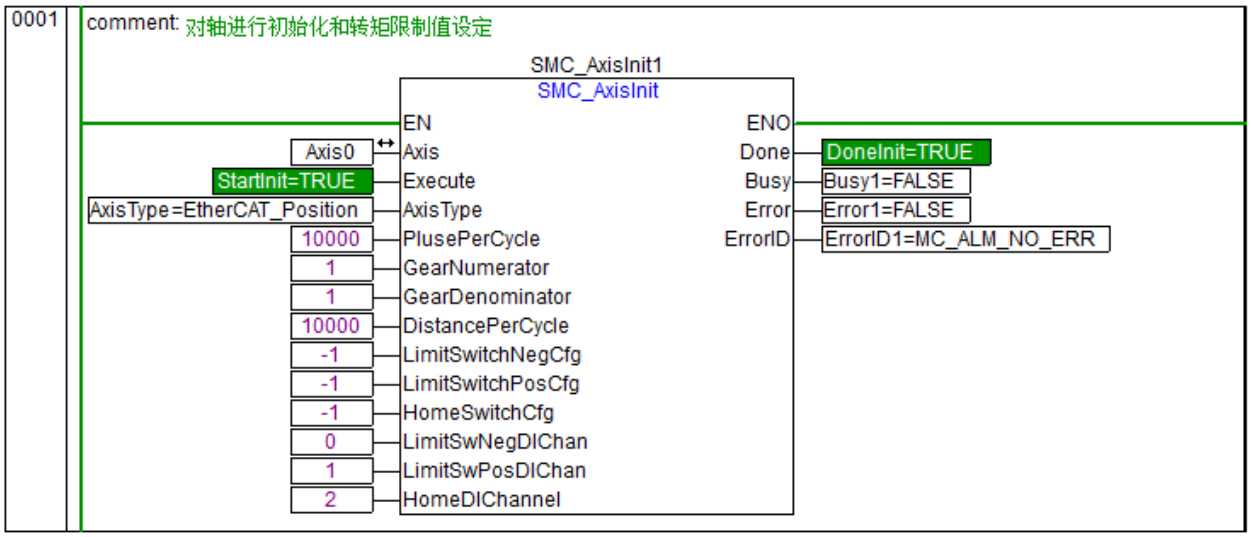
■ 主要变量

名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
ReadAxisErrEnable	BOOL	FALSE	开始读取时变为 TRUE
ValidReadAxisErr	BOOL	FALSE	读取成功时变为 TRUE
BusyReadAxisErr	BOOL	FALSE	读取时变为 TRUE
ErrorReadAxisErr	BOOL	FALSE	出错时变为 TRUE
AxisErrorId	DWORD	0	读取到的结果

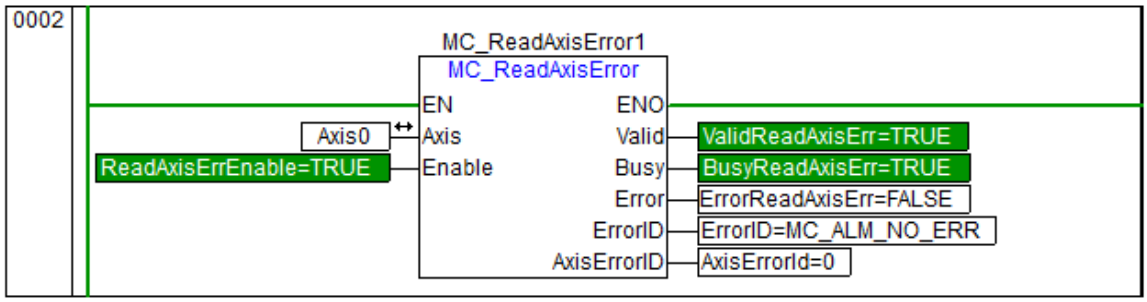
■ 梯形图程序

利用梯形图组建程序。将 MC_ ReadAxisError 的 Enable(使能)置为 TRUE，示例程序如下：

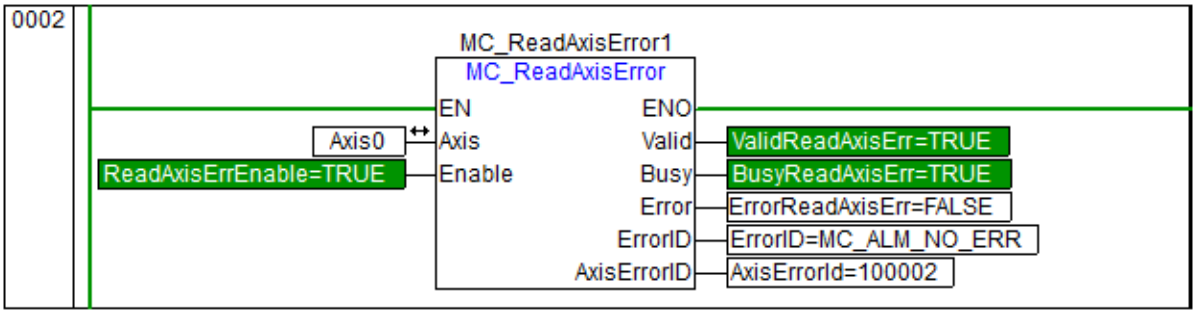
（1） 在进行轴错误信息读取之前，首先要对轴进行初始化。



（2） 初始化完成之后，可以直接进行轴错误信息的读取。



（3） 制造从站离线错误进行错误读取。



■ ST 语言程序

（1） 初始化轴号和轴类型设置

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
```

```
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

（2） 进行轴错误的读取

```
MC_ReadAxisError1(
Enable := ReadAxisErrEnable,
Axis := Axis0,
Valid=>ValidReadAxisErr,
Busy=>BusyReadAxisErr,
Error=>ErrorReadAxisErr ,
ErrorID=>ErrorID,
AxisErrorID=>AxisErrorId);
```

5.22.4 使用注意事项

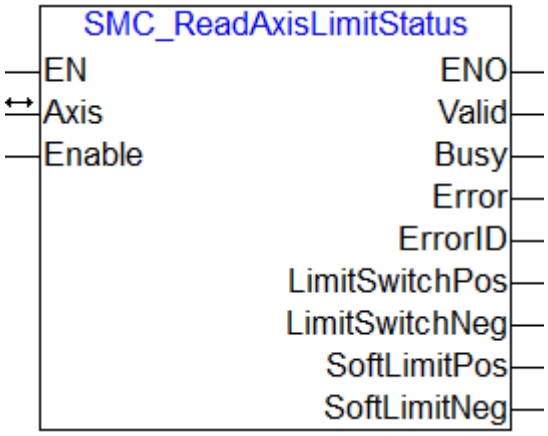
- 该指令为 Enable 型，不存在重启和多重启动

5.22.5 参见

- 功能块的错误信息参见附录功能块错误码说明。

5.23 SMC_ReadAxisLimitStatus（读取轴限位状态）

用于读取指定轴的正负硬限位状态和正负软限位状态。



5.23.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Enable	使能	否	-	TRUE/FALSE	BOOL
OUTPUT	Valid	输出有效	否	-	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR
	LimitSwitchPos	正向硬限位生效	是	FALSE	TRUE/FALSE	BOOL
	LimitSwitchNeg	负向硬限位生效	是	FALSE	TRUE/FALSE	BOOL
	SoftLimitPos	正向软限位生效	是	FALSE	TRUE/FALSE	BOOL
	SoftLimitNeg	负向软限位生效	是	FALSE	TRUE/FALSE	BOOL

5.23.2 功能

在 Enable=TRUE 时，本指令将读取轴当前的硬件限位状态和软件限位状态，包括正负硬限位状态和正负软限位状态。在输出引脚 Valid 有效的时候，读取的限位状态才是有效状态。当 Enable 为 FALSE 时，功能块输出引脚全部无效。

如果设置的硬限位属性配置是常开，则指定的限位通道为 TRUE 时，对应的限位状态为有效状态，否则为无效状态；如果设置的硬限位属性配置是常闭，则指定的限位通道为 FALSE 时，对应的限位状态为有效状态，否则为无效状态。

软限位状态与轴参 FSLimit 和 RSLimit 有关，当实际位置 MPos 大于等于 FSLimit 限制值时，正向软限位状态为 TRUE，否则为 FALSE；当实际位置 MPos 小于等于 RSLimit 限制值时，负向软限位状态为 TRUE，否则为 FALSE。

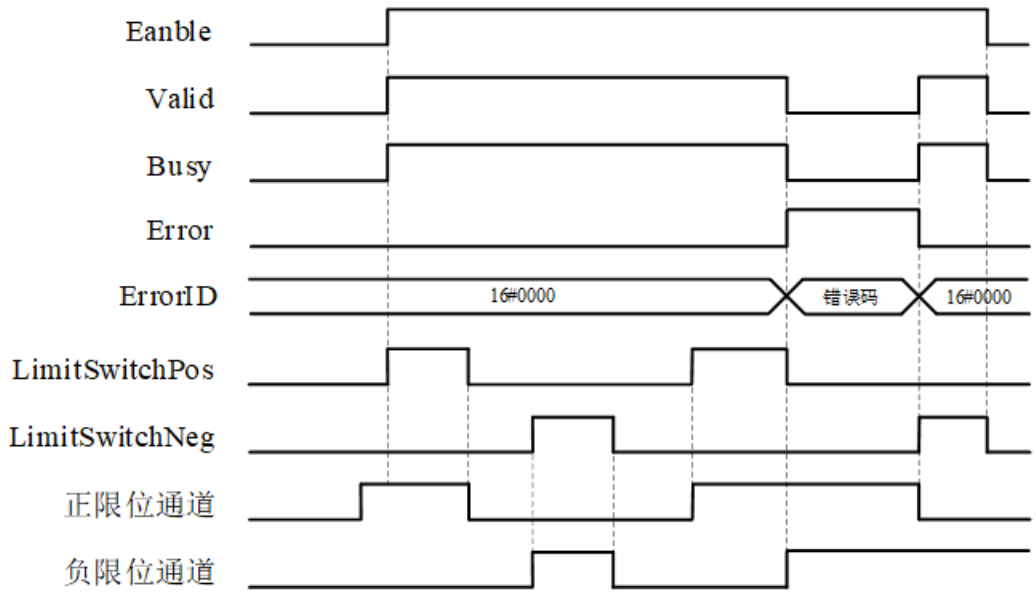
如果正负硬限位输入状态同时有效，则该指令报错。

同一个轴本指令可以多条同时运行，每一个指令读取到的状态结果一致。

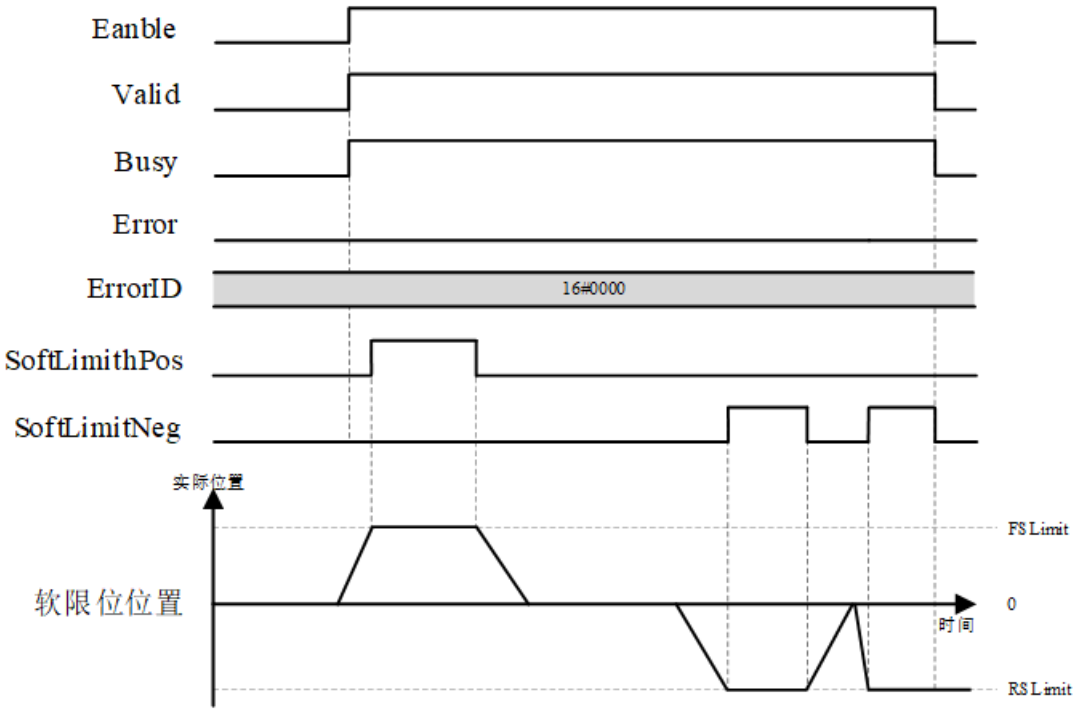
■ 功能块时序图

（1） 硬件正限位及负限位状态时序图。

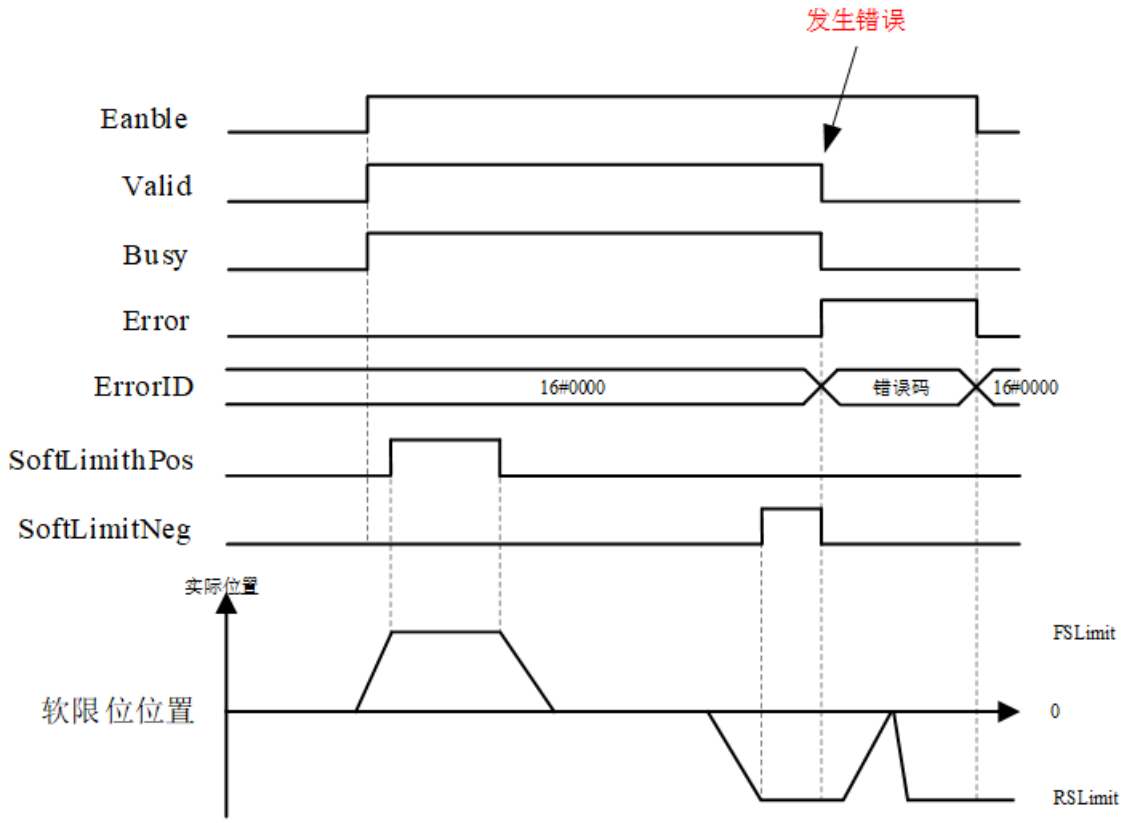
当正负硬限位开关同时有效时，该指令报错。



（2） 软件正限位及负限位状态时序图。



（3） 获取软件正限位及负限位状态时发生错误的时序图。



5.23.3 示例

该示例使用 SMC_AxisInit 功能块示例初始化后的正负硬限位属性配置及通道配置结果。

使用功能块 SMC_ReadAxisLimitStatus 获取 Axis0 的正负硬限位通道状态及正负软限位状态。

当负限位开关通道（%IX4.0）逻辑有效的时候，功能块 SMC_ReadAxisLimitStatus 输出引脚 LimitSwitchNeg 为 TRUE。

■ 变量

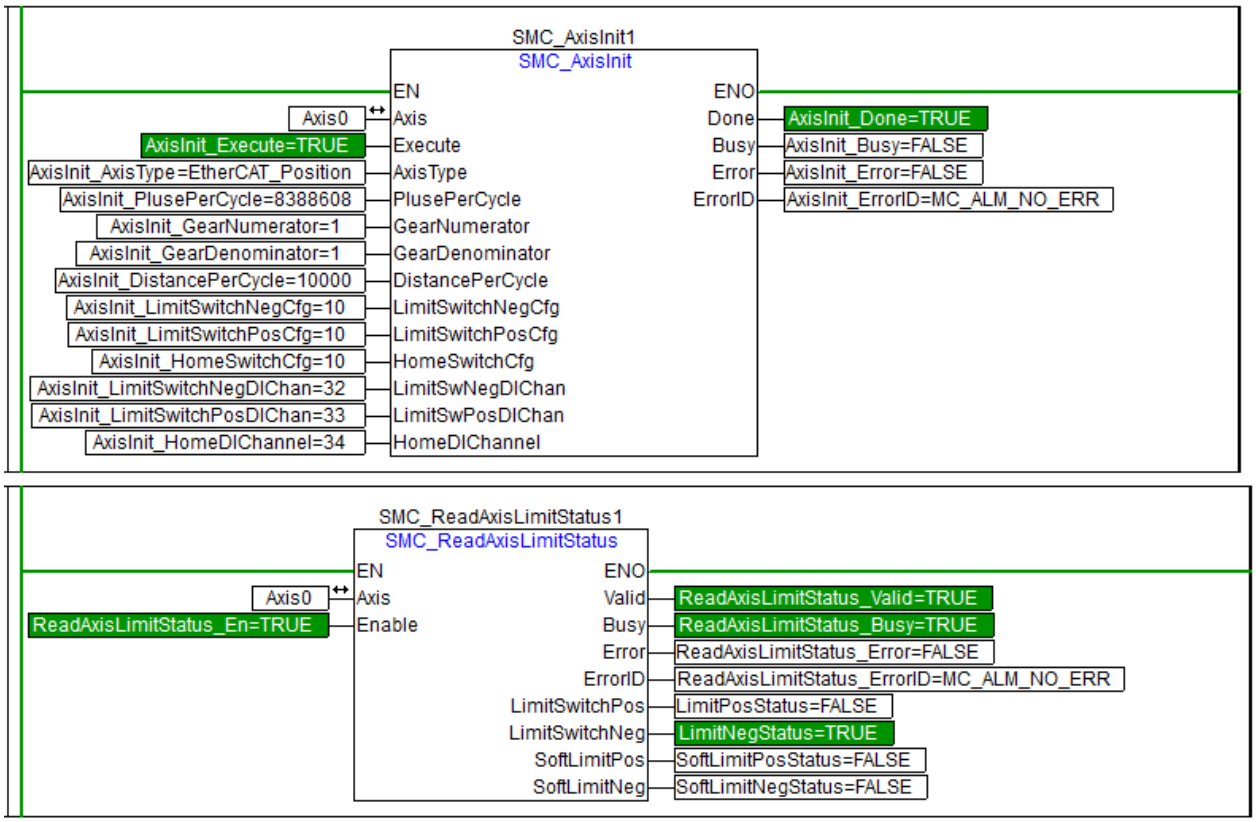
序号	变量名	变量类型	初始值	变量说明
1	SMC_ReadAxisLimitStatus1	SMC_ReadAxisLimitStatus		读取轴限制状态指令
2	ReadAxisLimitStatus_En	BOOL	FALSE	启用读取轴限制指令使能
3	ReadAxisLimitStatus_Valid	BOOL	FALSE	读取轴限制状态有效标志
4	ReadAxisLimitStatus_Busy	BOOL	FALSE	读取轴限制状态忙标志
5	ReadAxisLimitStatus_Error	BOOL	FALSE	读取轴限制状态错误标志
6	ReadAxisLimitStatus_ErrorID	SMC_ERROR	MC_ALM_NO_ERROR	读取轴限制状态错误 ID

7	LimitPosStatus	BOOL	FALSE	正向限位状态
8	LimitNegStatus	BOOL	FALSE	负向限位状态
9	SoftLimitPosStatus	BOOL	FALSE	正向软限位状态
10	SoftLimitNegStatus	BOOL	FALSE	负向软限位状态

■ LD 程序实现

先调用 SMC_AxisInit 指令初始化轴相关信息，然后调用指令 SMC_ReadAxisLimitStatus 获取限位状态信息。

触发负硬限位状态为有效，查看负限位状态为 TRUE。



■ ST 程序实现

（1） 先调用 SMC_AxisInit 指令初始化轴相关信息。

```
SMC_AxisInit1(
Execute := AxisInit_Execute,
AxisType := AxisInit_AxisType,
PlusePerCycle := AxisInit_PlusePerCycle,
GearNumerator := AxisInit_GearNumerator,
GearDenominator := AxisInit_GearDenominator,
DistancePerCycle := AxisInit_DistancePerCycle,
```

```
LimitSwitchNegCfg := AxisInit_LimitSwitchNegCfg,
LimitSwitchPosCfg := AxisInit_LimitSwitchPosCfg,
HomeSwitchCfg := AxisInit_HomeSwitchCfg,
LimitSwNegDIChan := AxisInit_LimitSwitchNegDIChan,
LimitSwPosDIChan := AxisInit_LimitSwitchPosDIChan,
HomeDIChannel := AxisInit_HomeDIChannel,
Axis := Axis0,
Done=>AxisInit_Done,
Busy=>AxisInit_Busy,
Error=>AxisInit_Error,
ErrorID=>AxisInit_ErrorID);
```

（2） 然后调用指令 SMC_ReadAxisLimitStatus 获取限位状态信息。

```
SMC_ReadAxisLimitStatus1(
Enable := ReadAxisLimitStatus_En,
Axis := Axis0,
Valid=>ReadAxisLimitStatus_Valid,
Busy=>ReadAxisLimitStatus_Busy,
Error=>ReadAxisLimitStatus_Error,
ErrorID=>ReadAxisLimitStatus_ErrorID,
LimitSwitchPos=>LimitPosStatus,
LimitSwitchNeg=>LimitNegStatus,
SoftLimitPos=>SoftLimitPosStatus,
SoftLimitNeg=>SoftLimitNegStatus);
```

5.23.4 使用注意事项

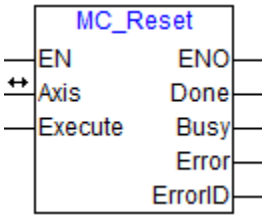
- 如果正负硬限位状态同时有效、正向软限位设置值小于等于负向软限位设置值或软限位和硬限位状态同时有效时，功能块报错。
- 当形程模式设置为循环行程（RepMode =1 或 2）时，限位状态无效且无实际意义。

5.23.5 参见

- 指令 SMC_AxisInit

5.24 MC_Reset（复位轴错误）

复位轴错误。



5.24.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
OUTPUT	Done	完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.24.2 功能

本指令可以复位轴错误，对该指令的详细介绍如下：

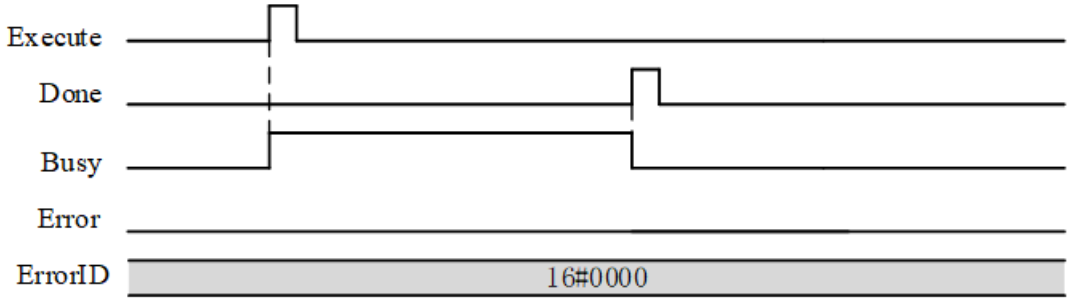
（1） 本指令可以复位 Ethercat 总线轴和虚拟轴的故障。

（2） 本指令在 Execute 上升沿触发有效，复位过程中 Busy(忙标志)为 TRUE，复位完成后 Done(完成)引脚为 TRUE，其他引脚为 FALSE，复位失败后 Error(错误)为 TRUE，ErrorID 输出对应的错误码。

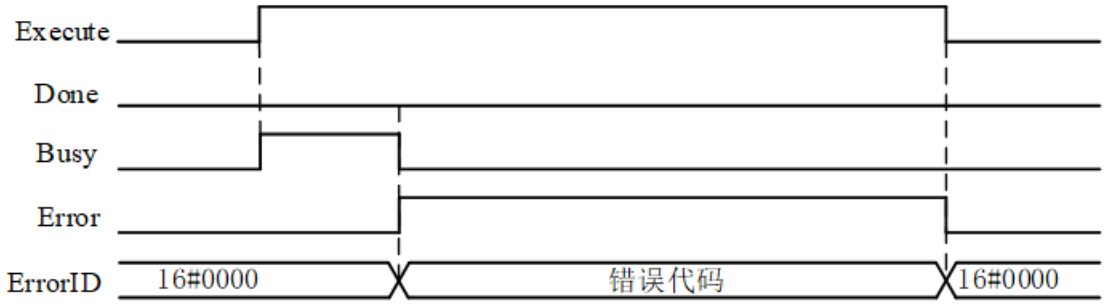
（3） 复位成功后，如果驱动器处于使能状态则轴的 PLCopen 状态机进入 StandStill 状态，否则进入 Disabled 状态。

■ 功能块时序图

（1） 轴发生故障且复位成功的时序图。



(2) 轴发生不可复位的故障时的时序图



5.24.3 示例

组建 MC_ Reset 示例程序进行轴错误的复位处理。

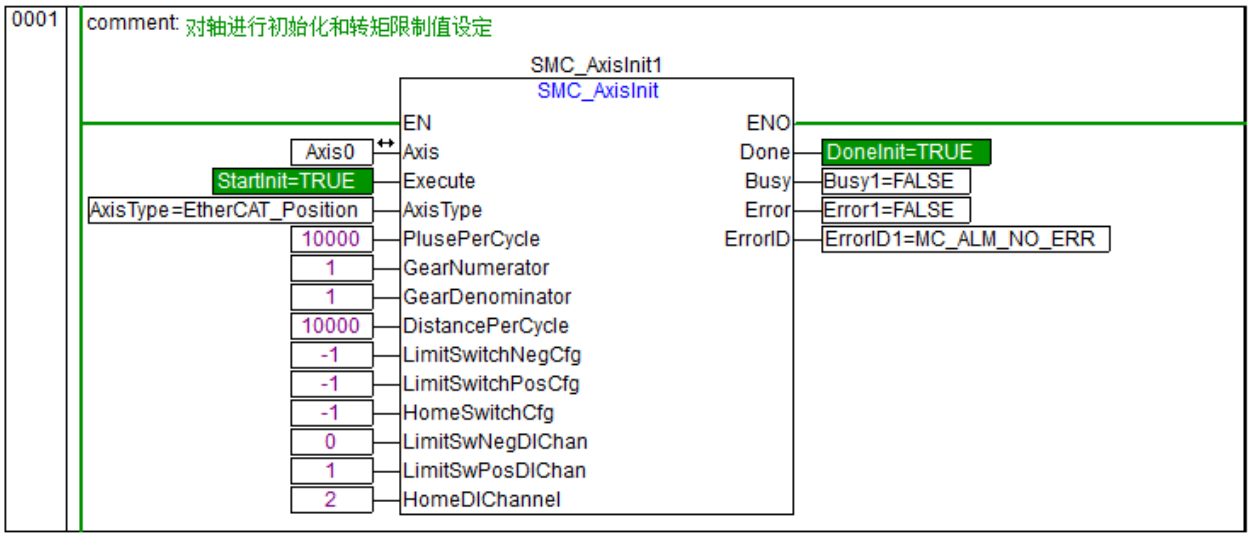
■ 主要变量

名称	数据类型	初始值	注释说明
Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
ResetExe	BOOL	FALSE	开始复位时变为 TRUE
DoneReset	BOOL	FALSE	复位成功时变为 TRUE
BusyReset	BOOL	FALSE	复位时变为 TRUE
ErrorReset	BOOL	FALSE	复位失败时变为 TRUE

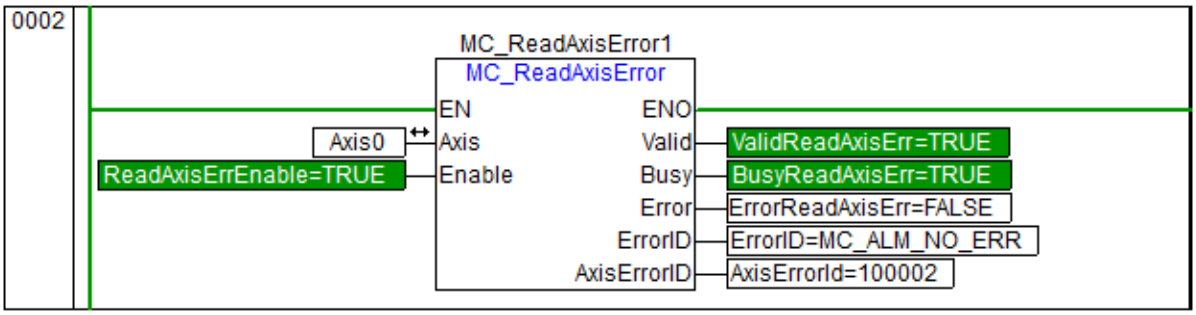
■ 梯形图程序

利用梯形图组建程序。将 MC_ Reset 的 Execute(上升沿触发)置为 TRUE，示例程序如下：

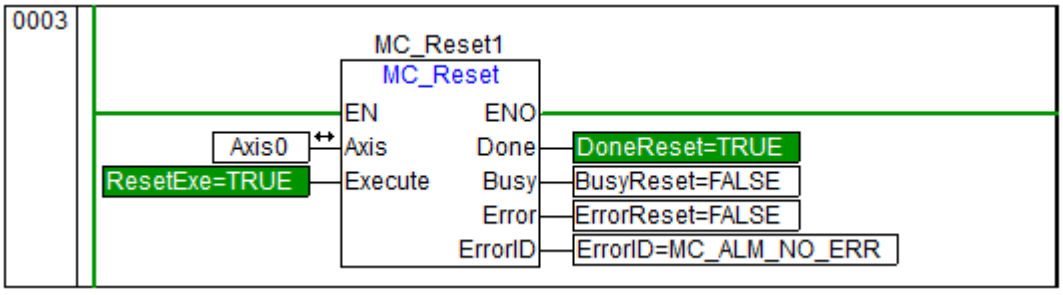
(1) 首先要对轴进行初始化



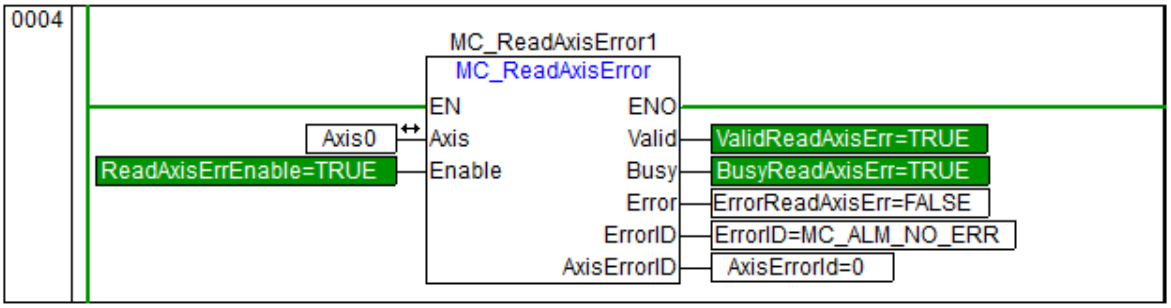
（2） 制造伺服离线错误



（3） 使用 MC_Reset 进行复位



（4） 读取轴错误可以看见错误已经被复位成功



■ ST 语言程序

（1） 初始化轴号和轴类型设置

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

（2） 调用 MC_Reset 指令进行复位

```
MC_Reset1(
Execute := ResetExe,
Axis := Axis0,
Done=>DoneReset,
Busy=>BusyReset,
Error=>ErrorReset,
ErrorID=>ErrorID);
```

5.24.4 使用注意事项

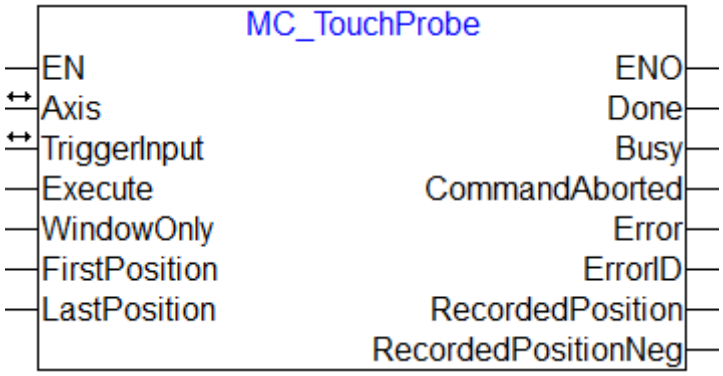
■ 当存在不可复位的错误时，调用该功能块无法复位轴的错误。

5.24.5 参见

■ 相关复位错误信息，参见附录功能块错误码。

5.25 MC_TouchProbe（轴位置锁存）

该指令用于检测到指定的探针事件触发时，锁存指定轴和锁存通道的位置。



5.25.1 参数

参数说明

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
	TriggerInput	锁存配置参数	否	-	参见数据结构 MC_TRIGGER_REF	MC_TRIGGER_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	WindowOnly	FALSE：禁止窗口功能。 TRUE：使能窗口功能，只有当前位置在[FirstPosition, LastPosition]之间时才检测探针信号。	是	FALSE	TRUE/FALSE	BOOL
	FirstPosition	探针窗口开始位置	是	0	正值/负值/0	LREAL
	LastPosition	探针窗口结束位置	是	0	正值/负值/0	LREAL
OUTPUT	Done	锁存完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL

	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR
	RecordedPosition	上升沿锁存位置	是	-	正值/负值/0	LREAL
	RecordedPositionNeg	下降沿锁存位置	是	-	正值/负值/0	LREAL

MC_TRIGGER_REF 数据结构

成员	含义	默认值	范围	数据类型
LatchMode	锁存模式： 0：驱动器模式	0	0	USINT
LatchID	驱动器模式下的锁存通道： 0：Latch 1 1：Latch 2	0	0/1	USINT
DriveInput	驱动器模式下的触发信号： 0：驱动器 IO 端子 1：编码器 Z 相	0	0/1	UDINT
TriggerEdge	触发边沿类型： 0：上升沿 1：下降沿 2：双沿	0	0/1/2	USINT
ContinuousTrigger	是否连续触发	FALSE	TRUE/FALSE	BOOL

5.25.2 功能

该指令用于实现符合 CIA402 标准的 EtherCAT 伺服轴或编码器轴（ECI002）的探针捕获功能。

该指令不支持虚轴模式。指令在运行过程中，如果对应轴的状态跳转到 ErrorStop 状态，则该功能块报错，并释放指令已占用的锁存通道。

该指令支持的功能：

- 支持两个锁存通道的探针捕获；
- 支持驱动器 DI 端子触发和 Z 相信号为触发源，需伺服驱动器支持；
- 支持上升沿、下降沿和双沿触发类型，需伺服驱动器支持；
- 支持连续触发模式；

- 支持窗口位置捕获模式。

使用该指令，需要映射以下对象字典到 PDO：

- Touch probe function (60B8 hex，必需)；
- Touch probe status (60B9 hex，必需)；
- Touch probe pos1 pos value (60BA hex，探针 1 上升沿模式需要)；
- Touch probe pos1 neg value (60BB hex，探针 1 下降沿模式需要)；
- Touch probe pos2 pos value (60BC hex，探针 2 上升沿模式需要)；
- Touch probe pos2 neg value (60BD hex，探针 2 下降沿模式需要)。

如果是多轴伺服驱动器，需要根据实际情况映射偏移后的对象字典。

该指令在 Excute 上升沿时，检查 TriggerInput 参数和窗口参数是否在有效范围内，是则将 TriggerInput 参数和窗口参数记录锁存，Excute 在其它状态时无法更改参数，如果检查出错则功能块报错，输出引脚 Error 置 TRUE。入参检查通过后，指令进入锁存状态，输出引脚 Busy 置 TRUE，指令在检测到 LatchID 指定的探针输入有效且满足探针检测条件时，功能块将锁存轴的当前位置，并将锁存的位置按需刷新到输出引脚 RecordedPosition 或 RecordedPositionNeg，同时功能块输出引脚 Done 置 TRUE。其中输出引脚 Busy、Done 和 Error 强制互斥。

该指令可独立检测探针信号的上升沿、下降沿或者同时检查上升沿和下降沿。在只检测上升沿（下降沿）时，指令将上升沿（下降沿）检测到的位置值写入 RecordedPosition（RecordedPositionNeg）参数中，此时完成一个检测周期并将 Done 引脚置位。如果同时检测上升沿和下降沿，则在指令 Excute 上升沿有效后，当指令检测到探针上升沿后立即将位置写入 RecordedPosition 引脚，检测到下降沿后立即将位置写入 RecordedPositionNeg 引脚，此时才算作一个完整的检测周期并输出 Done 信号，此时对上升沿和下降沿的输入先后顺序没有要求。

该指令可以配置为单次触发或者连续触发。当配置为单次触发时，Done 信号输出有效时表示单次位置捕获完成，指令执行结束；当配置为连续触发模式时，位置捕获完成后，Done 输出有效信号，一个 IEC 扫描周期后重新复位，指令自动开始检测新的探针输入信号。在连续触发模式下如果探针信号的输入频率大于 IEC 扫描周期的频率，该指令将丢失部分探针信号。

- 窗口模式说明

输入参数 WindowOnly 为 FALSE 时，窗口检测功能无效。此时只要探针输入信号有效，就可以锁存对应的轴的位置。

输入参数 WindowOnly 为 TRUE 时，窗口检测功能有效。此时只有轴的当前位置在窗口范围内时，探针捕获指令才检查探针信号，并锁存位置。

（1） 有限行程

窗口有效范围：FirstPosition ≤ 窗口范围 ≤ LastPosition，入参 FirstPosition ＞ LastPosition 时，功能块报错。

（2） 循环行程

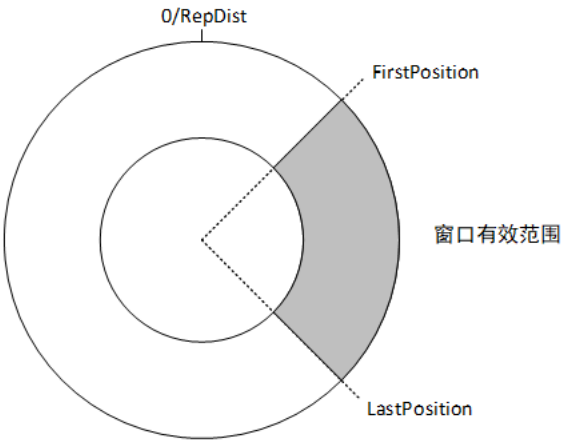
循环行程时 FirstPosition ≤ LastPosition)和 FirstPosition ＞ LastPosition 两者均可根据实际使用情况配置。

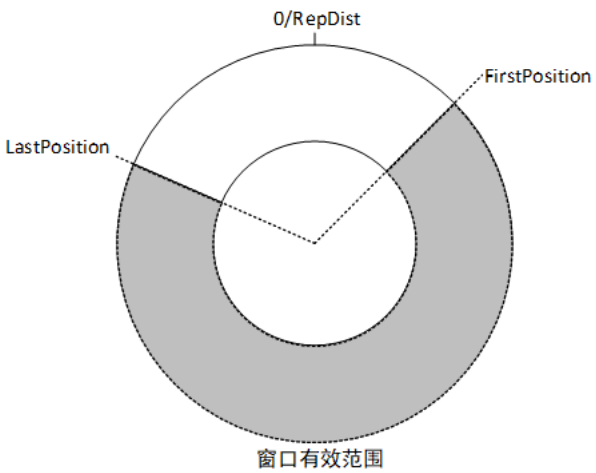
FirstPosition ＞ LastPosition 时，设定值将跨越循环行程的上下限位置。

FirstPosition、LastPosition 设置值超过循环行程上下限范围时，功能块报错。

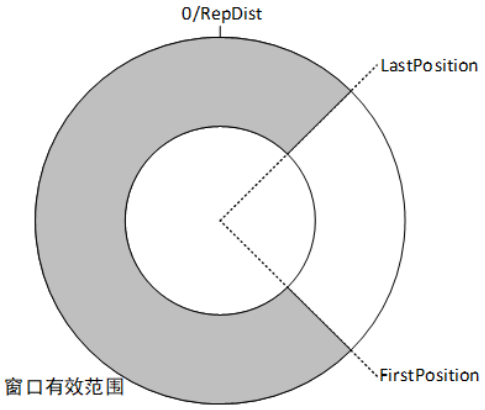
□ 循环行程模式 1 时有效范围如下图，循环行程范围为[0,RepDist]：

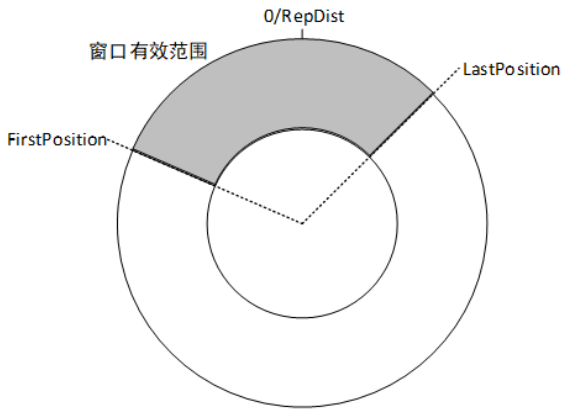
当 FirstPosition ≤ LastPosition 时，如下图所示：





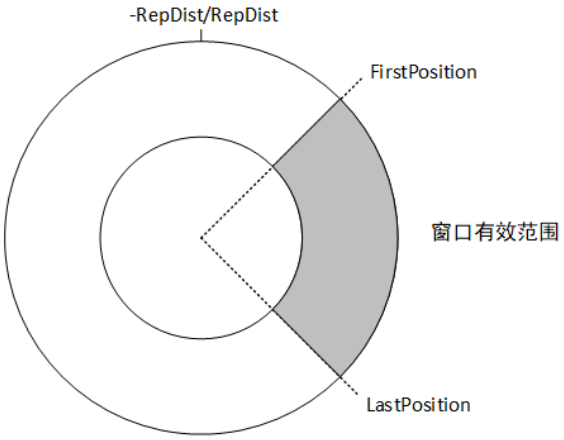
当 FirstPosition > LastPosition 时，如下图所示：

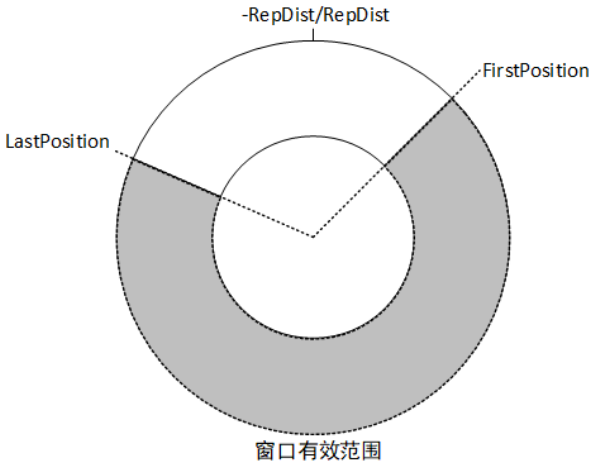




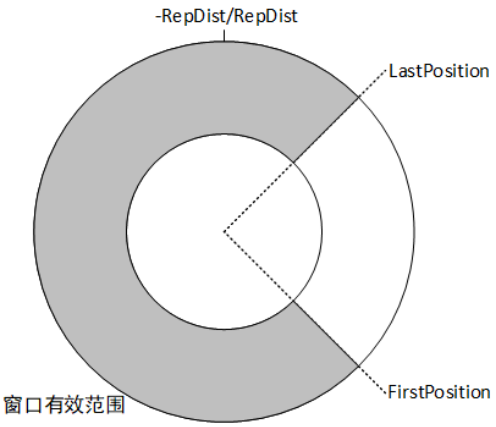
□ 循环行程模 2 时有效范围如下图，循环行程范围为[-RepDist,RepDist]：

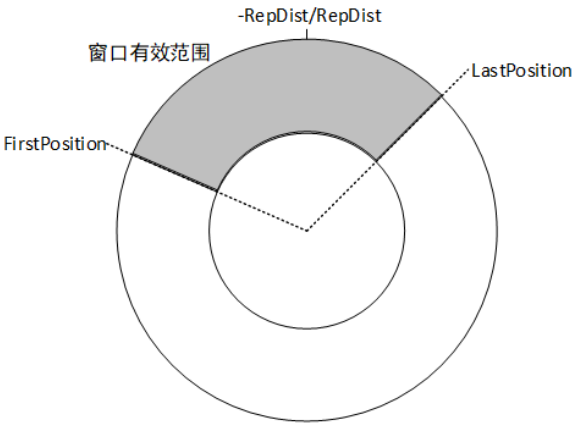
当 FirstPosition ≤ LastPosition 时，如下图所示：





当 FirstPosition > LastPosition 时，如下图所示：





■ 打断说明

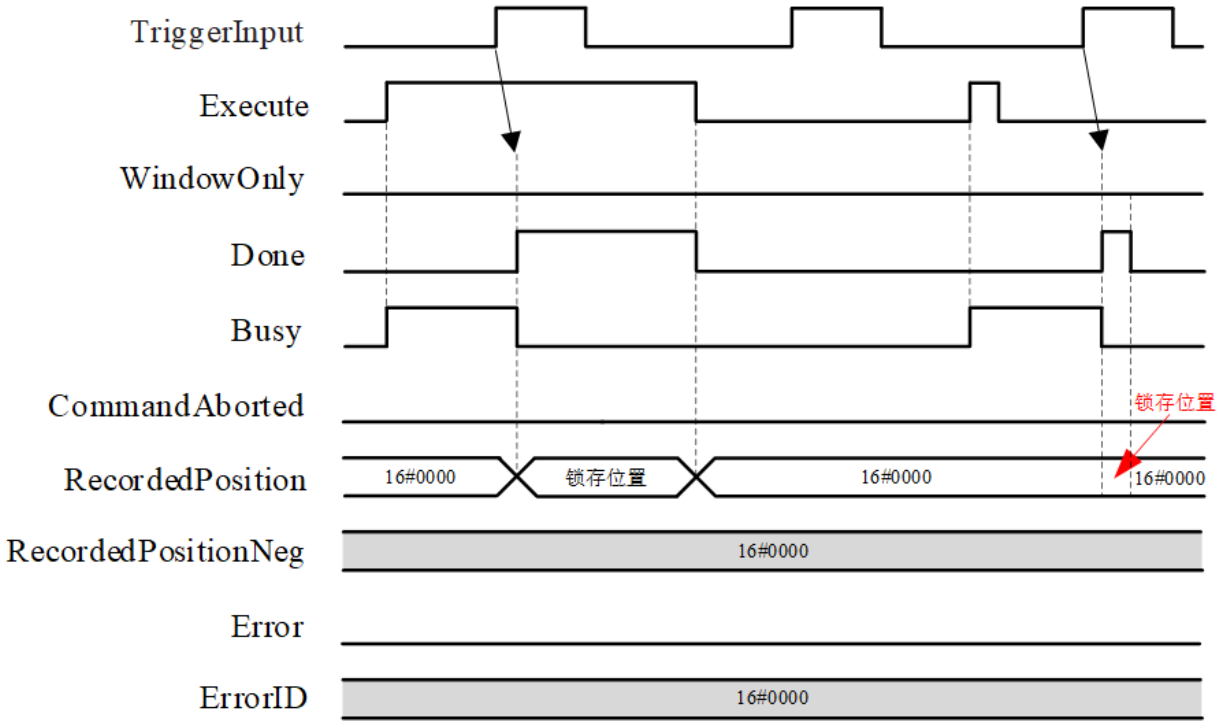
该指令支持同时检测同一个轴的探针 1 和探针 2，如果程序中定义两条探针指令，两条指令的探针 ID 选择不同，则两条探针指令独立工作，互不影响。如果两条指令的探针 ID 相同，则后执行的探针指令报错。如果程序中只定义了一条探针指令，当指令在运行过程中，再次上升沿触发该指令，则功能块报错并中止正在运行的探针指令，释放指令占用的探针 ID。如果一个探针指令被重复调用，使能后功能块直接报错，无法使用指令，实际情况不允许这样使用，重复调用同一个功能块实例在组态编译时会报警告。

该指令在正常运行时，可通过 MC_AbortTrigger 指令中止 MC_TouchProbe 指令，并释放已占用的探针。当 MC_AbortTrigge 输出引脚 Done 为 TRUE 时，表示打断成功，此时 MC_TouchProb 指令输出引脚 CommandAborted 置位。

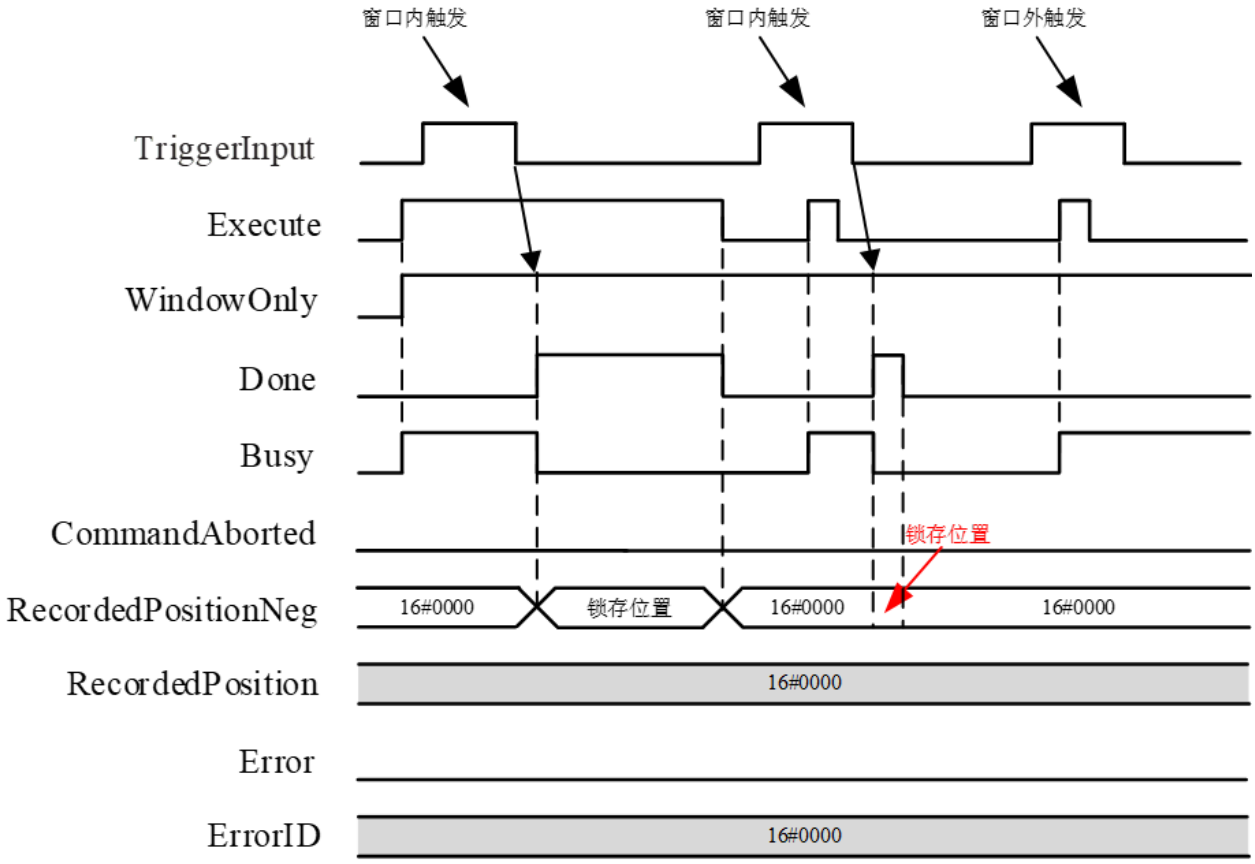
MC_Home 和 HMC_GantryInit 指令在使用时，如果将原点回归模式配置为 Z 相信号相关，则会使用探针 1 进行 Z 相信号捕获，此时将占用该轴的探针 1 通道，其它指令无法使用，且不能被打断。在使用 MC_Home 和 HMC_GantryInit 指令进行 Z 相信号相关的原点回归时，若该轴的探针 1 通道已经被占用，则 MC_Home 和 HMC_GantryInit 指令报错。

■ 功能块时序图

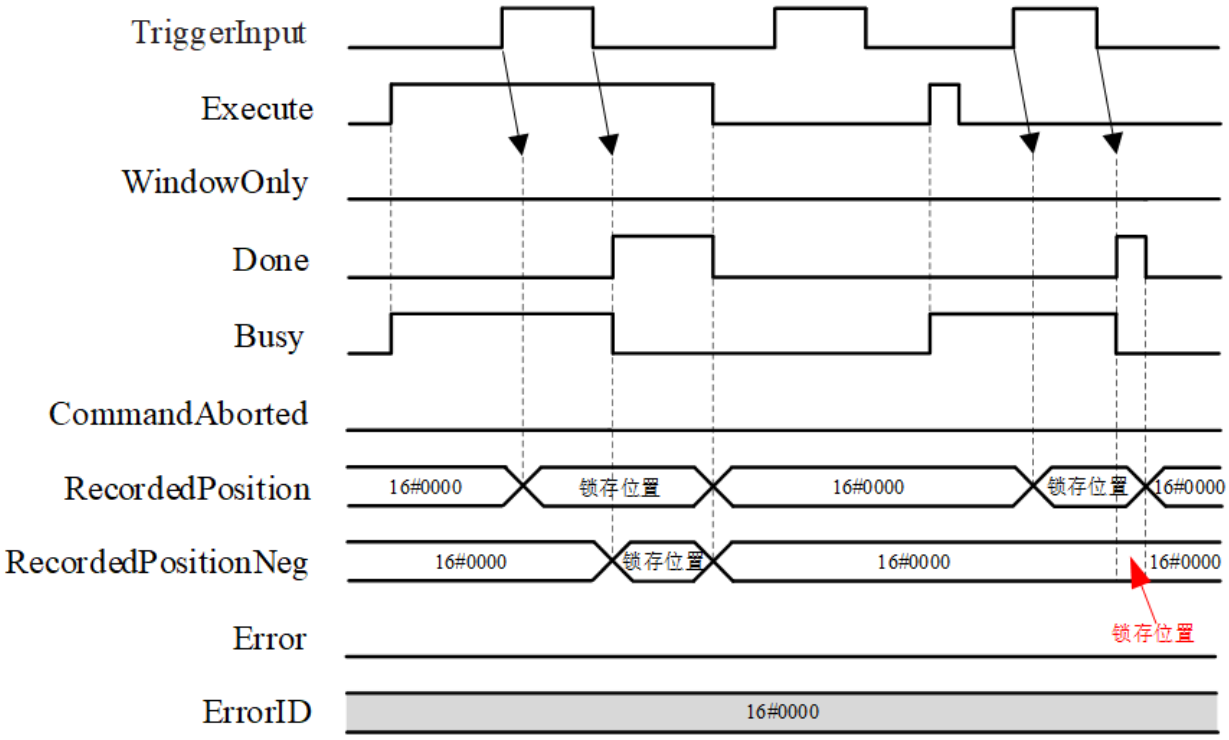
（1） 探针 1 上升沿有效，DI 端子触发，单次触发模式，窗口功能无效。



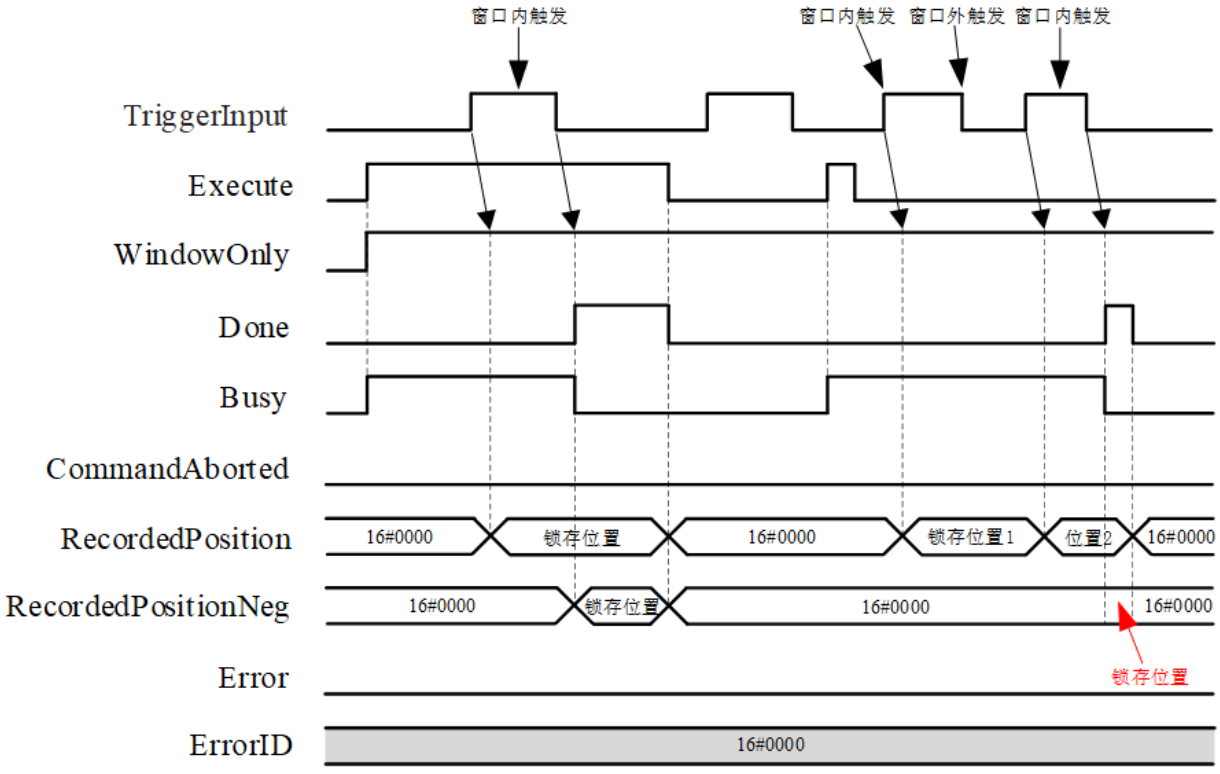
(2) 探针 1 下降沿有效，DI 端子触发，单次触发模式，窗口功能有效。



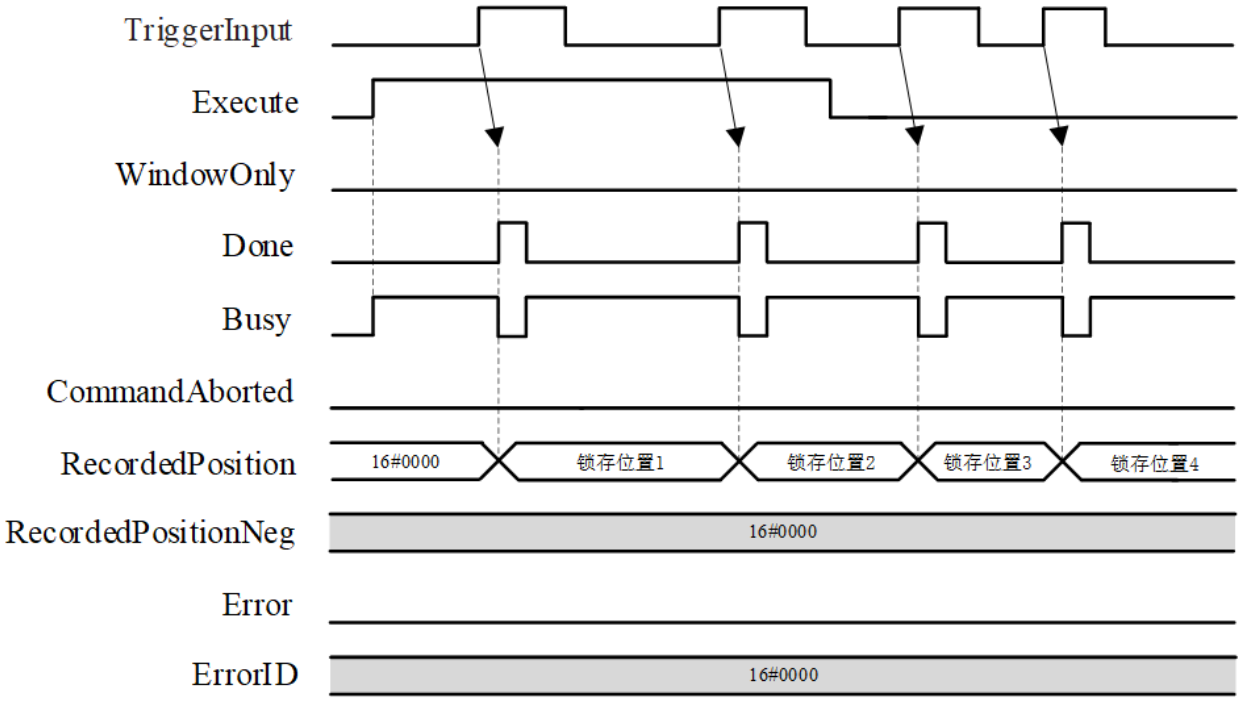
（3） 探针 1 上升沿和下降沿有效，DI 端子触发，单次触发模式，窗口功能无效。



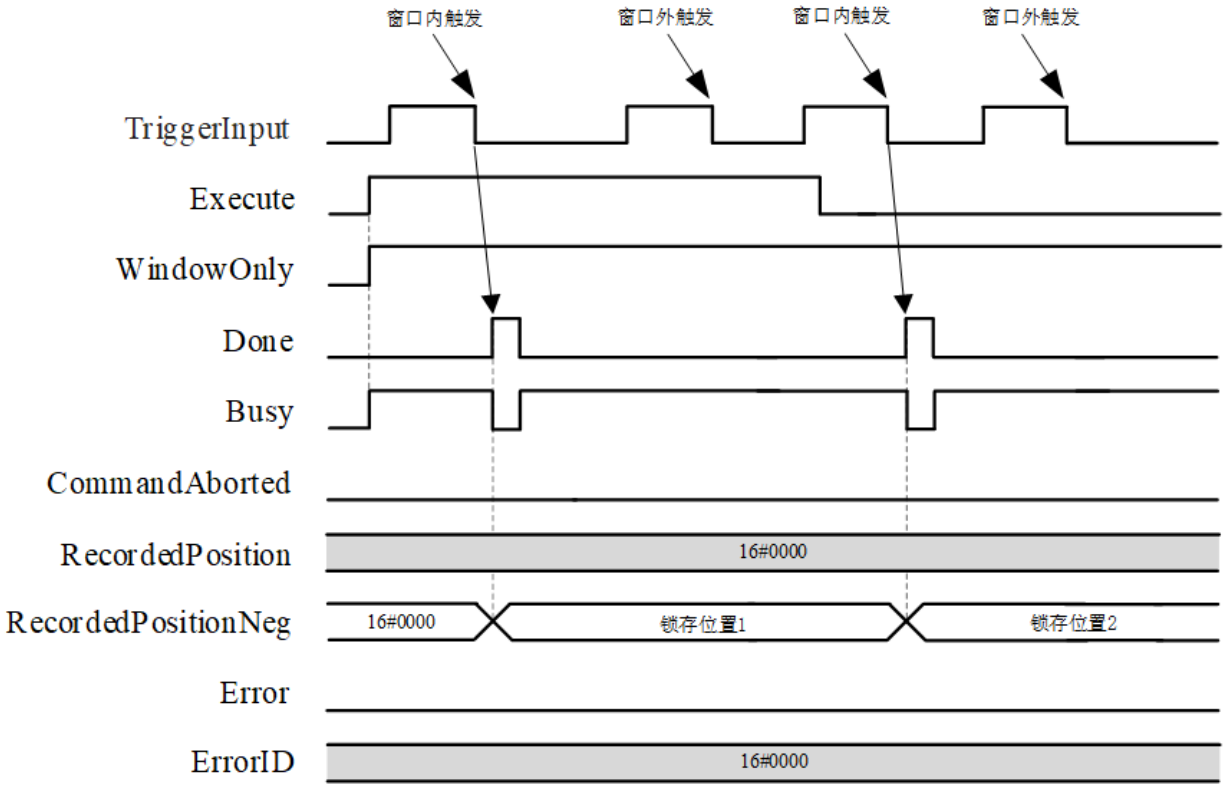
（4） 探针 1 上升沿和下降沿有效，DI 端子触发，单次触发模式，窗口功能有效。



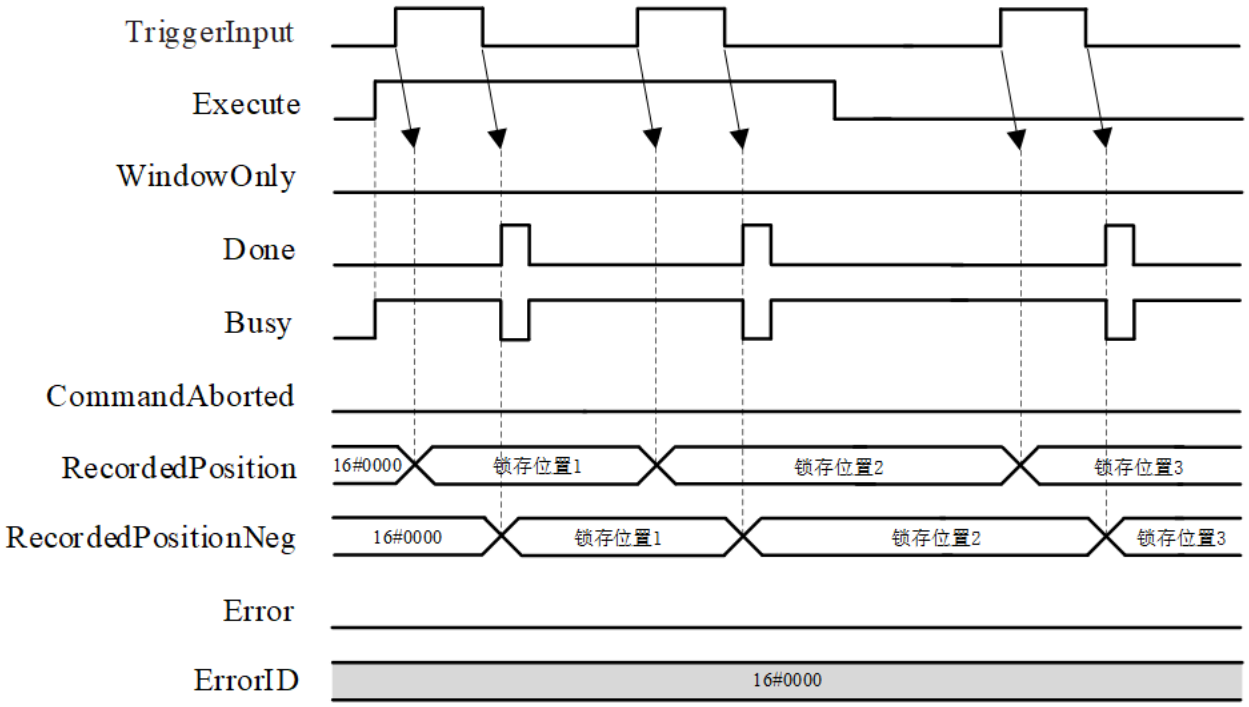
（5） 探针 1 上升沿有效，DI 端子触发，连续触发模式，窗口功能无效。



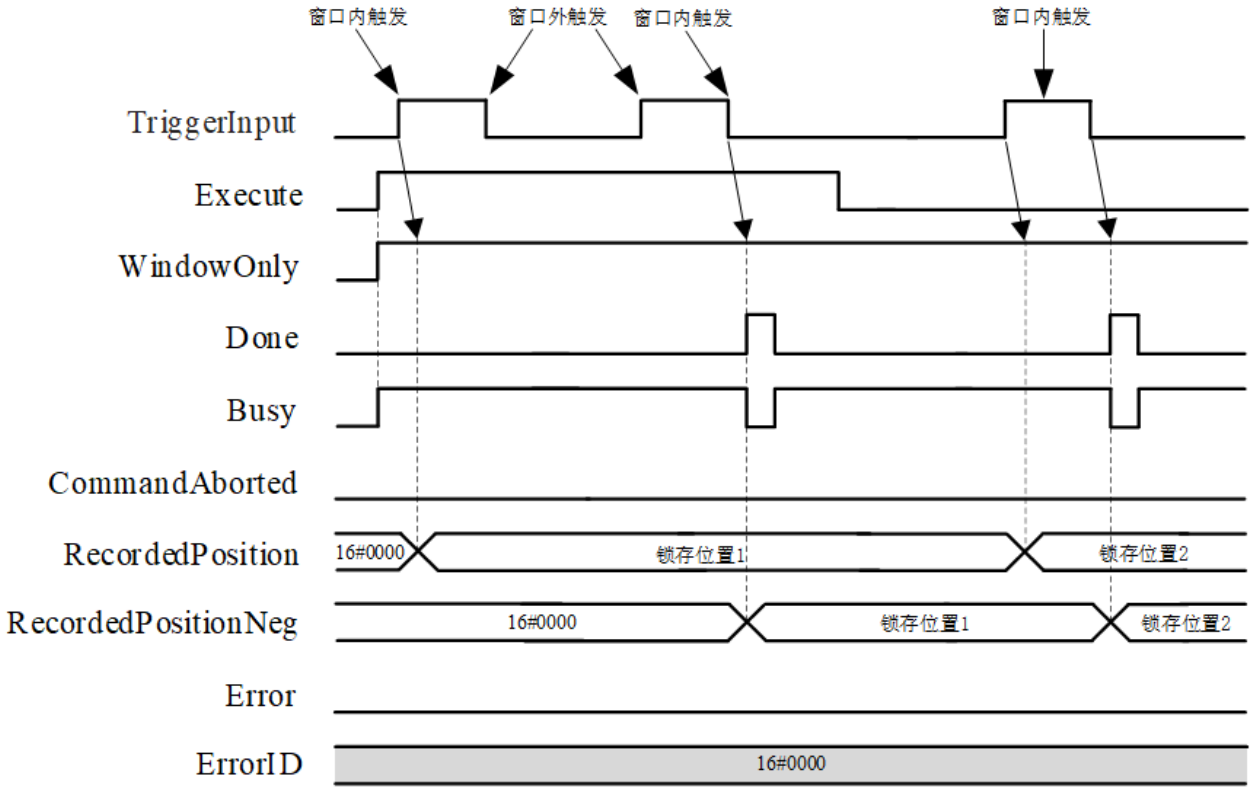
（6） 探针 1 下降沿有效，DI 端子触发，连续触发模式，窗口功能有效。



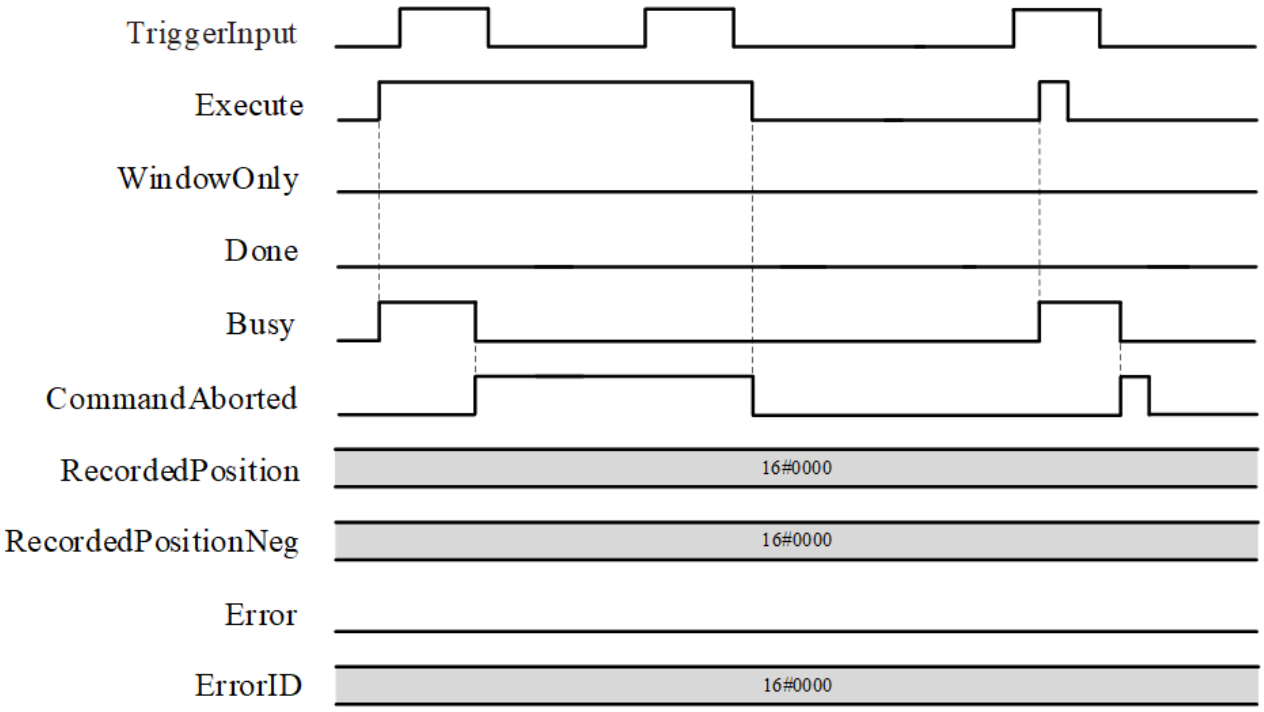
（7） 探针 1 上升沿和下降沿有效，DI 端子触发，连续触发模式，窗口功能无效。



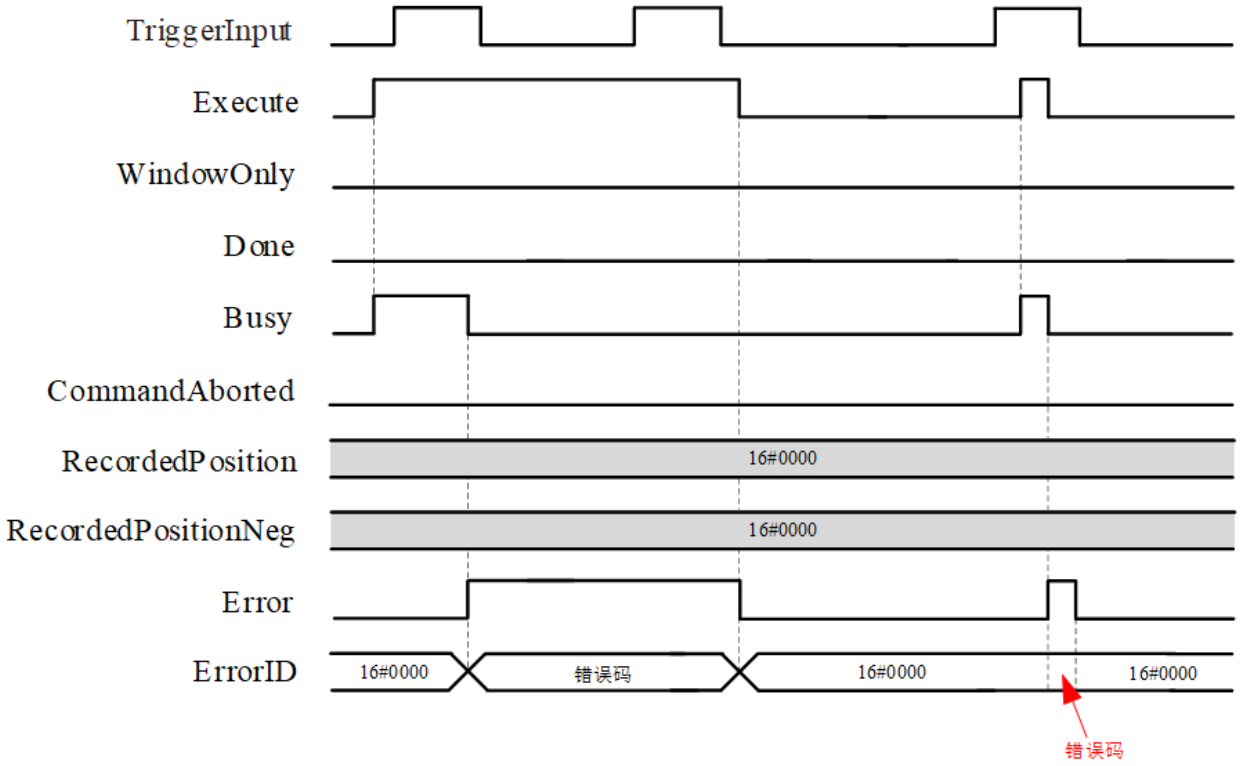
（8） 探针 1 上升沿和下降沿有效，DI 端子触发，连续触发模式，窗口功能有效。



（9） 探针 1 上升沿有效，DI 端子触发，窗口功能无效，被 MC_AbortTrigger 指令打断。



（10） 探针 1 上升沿有效，DI 端子触发，窗口功能无效，指令发生故障。



5.25.3 示例

该示例使用功能块 MC_TouchProbe 抓取 EtherCAT 伺服探针信号并记录位置。

触发条件设置如下：

- 使用探针 1；
- 上升沿触发；
- 触发信号为驱动器 IO 端子；
- 连续触发；
- 不使用窗口触发。

在使用探针捕获功能之前需要使用 SMC_AxisInit 指令进行轴类型设置和用户单位配置，还需要配置好相关的 PDO 映射。

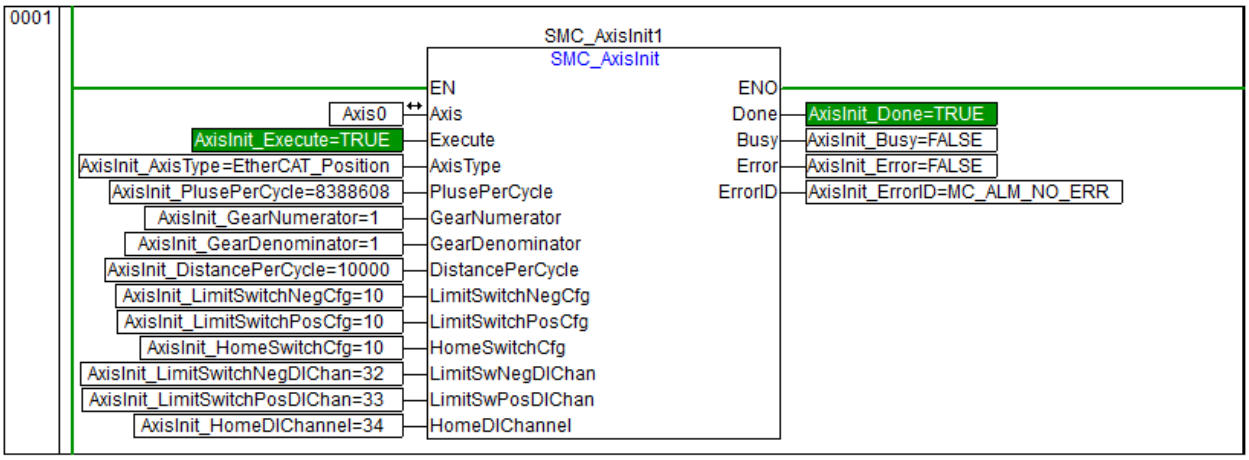
伺服驱动器侧需要配置 DI 功能为探针 1 触发信号，具体查看伺服相关手册。

- 变量

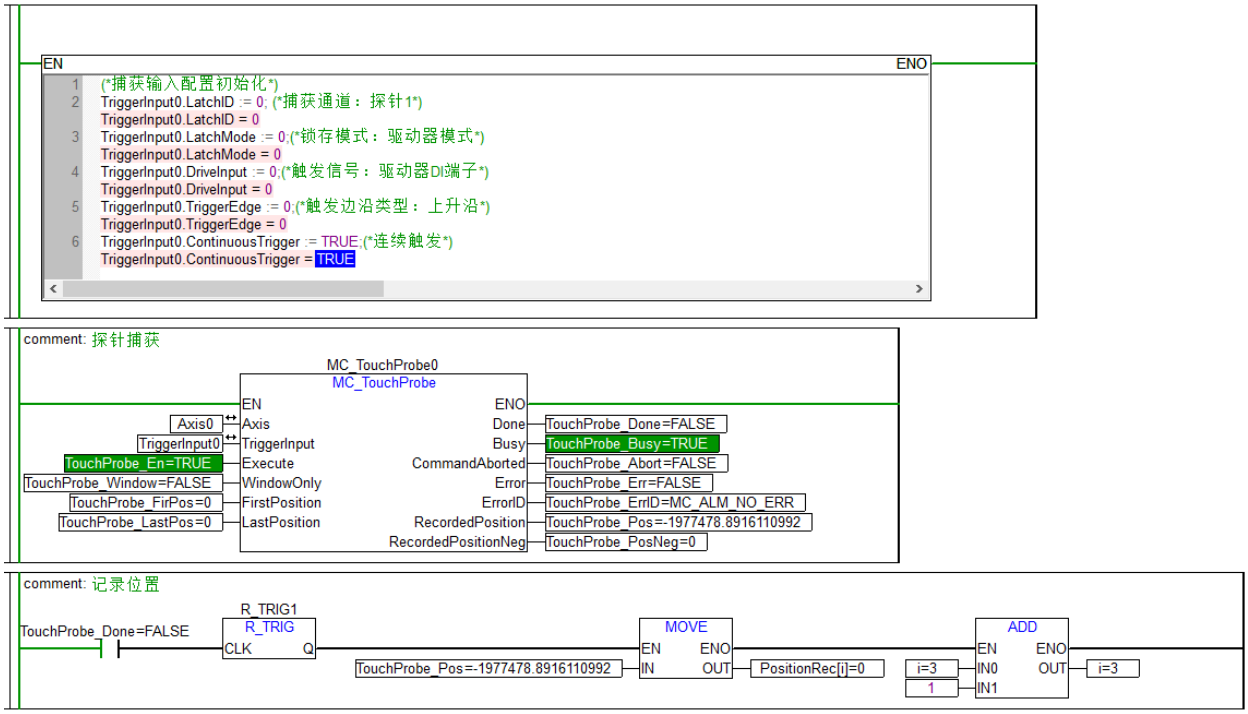
序号	变量名	变量类型	初始值	变量说明
1	MC_TouchProbe0	MC_TOUCHPROBE	FALSE	探针捕获指令
2	TriggerInput0	MC_TRIGGER_REF	FALSE	探针触发输入定义
3	TouchProbe_En	BOOL	FALSE	使能探针捕获，上升沿有效
4	TouchProbe_Window	BOOL	FALSE	窗口捕获使能
5	TouchProbe_FirPos	LREAL	0	窗口模式开始位置
6	TouchProbe_LastPos	LREAL	0	窗口模式结束位置
7	TouchProbe_Done	BOOL	FALSE	探针捕获指令完成标志
8	TouchProbe_Busy	BOOL	FALSE	探针捕获指令忙标志
9	TouchProbe_Abort	BOOL	FALSE	探针捕获指令被打断标志
10	TouchProbe_Err	BOOL	FALSE	探针捕获指令错误标志
11	TouchProbe_ErrID	SMC_ERROR	MC_ALM.	探针捕获指令错误 ID
12	TouchProbe_Pos	LREAL	0	探针捕获上升沿位置
13	TouchProbe_PosNeg	LREAL	0	探针捕获指令下降沿位置
14	R_TRIG1	R_TRIG	FALSE	上升沿触发
15	PositionRec	ARRAY[0..10] OF LREAL		位置记录数组
16	i	INT	0	变量

■ LD 程序实现

（1） 先调用 SMC_AxisInit 指令初始化轴相关信息。



（2） 对 TriggerInput0 结构体进行初始化后，调用 MC_TouchProbe 进行探针位置捕获，并记录捕获位置。



■ ST 程序实现

（1） 首先调用 MC_AxisInit 指令初始化轴相关信息

```
MC_ReadAxisInfo1(
Enable := ReadAxisInfo_En,
Axis := Axis0,
Valid=>ReadAxisInfo_Valid,
Busy=>ReadAxisInfo_Busy,
Error=>ReadAxisInfo_Error,
ErrorID=>ReadAxisInfo_ErrorID,
HomeAbsSwitch=>ReadAxisInfo_HomeAbsSwitch,
LimitSwitchPos=>ReadAxisInfo_LimitSwitchPos,
LimitSwitchNeg=>ReadAxisInfo_LimitSwitchNeg,
Simulation=>ReadAxisInfo_Simulation,
CommunicationReady=>ReadAxisInfo_ComRdy,
ReadyForPowerOn=>ReadAxisInfo_RdyPowerON,
PowerOn=>ReadAxisInfo_PowerON,
IsHomed=>ReadAxisInfo_Homed,
AxisWarning=>ReadAxisInfo_Warning,
AxisFault=>ReadAxisInfo_Fault);
```

（2） 捕获输入配置初始化

```
TriggerInput0.LatchID := 0; (*捕获通道：探针 1*)

TriggerInput0.LatchMode := 0;(*锁存模式：驱动器模式*)
```

```
TriggerInput0.DriveInput := 0;(*触发信号：驱动器 DI 端子*)
TriggerInput0.TriggerEdge := 0;(*触发边沿类型：上升沿*)
TriggerInput0.ContinuousTrigger := TRUE;(*连续触发*)
```

（3） 调用 MC_TouchProbe 进行探针位置捕获

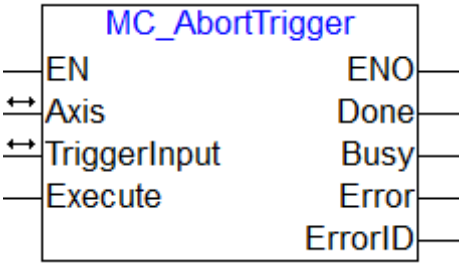
```
(*MC_TouchProbe 指令*)
MC_TouchProbe0(
   Execute := TouchProbe_En,
   WindowOnly := TouchProbe_Window,
   FirstPosition := TouchProbe_FirPos,
   LastPosition := TouchProbe_LastPos,
   Axis := Axis0,
   TriggerInput := TriggerInput0,
   Done=>TouchProbe_Done,
   Busy=>TouchProbe_Busy,
   CommandAborted=>TouchProbe_Abort,
   Error=>TouchProbe_Err,
   ErrorID=>TouchProbe_ErrID,
   RecordedPosition=>TouchProbe_Pos,
   RecordedPositionNeg=>TouchProbe_PosNeg);
```

（4） 记录捕获位置。

```
R_TRIG1(CLK:=TouchProbe_Done);
(*记录位置*)
IF R_TRIG1.Q = TRUE THEN
   PositionRec[i] := TouchProbe_Pos;
   i:=i+1;
END_IF;
```

5.26 MC_AbortTrigger（停止位置锁存）

该指令用停止 MC_TouchProbe 指令发起的位置锁存操作。



5.26.1 参数

参数说明

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Axis	轴名	否	-	-	AXIS_REF
	TriggerInput	锁存配置参数	否	-	参见数据结构 MC_TRIGGER_REF	MC_TRIGGER_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
OUTPUT	Done	完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

5.26.2 功能

该指令用于中止 MC_TouchProbe 发起的位置锁存操作，不支持非 EtherCAT 轴。

该指令在执行时，只匹配指定 Axis(轴)和 LatchID(锁存通道)对应的位置锁存操作，若其它配置参数与该指令入参不相同，也不影响该指令中止位置锁存的功能。

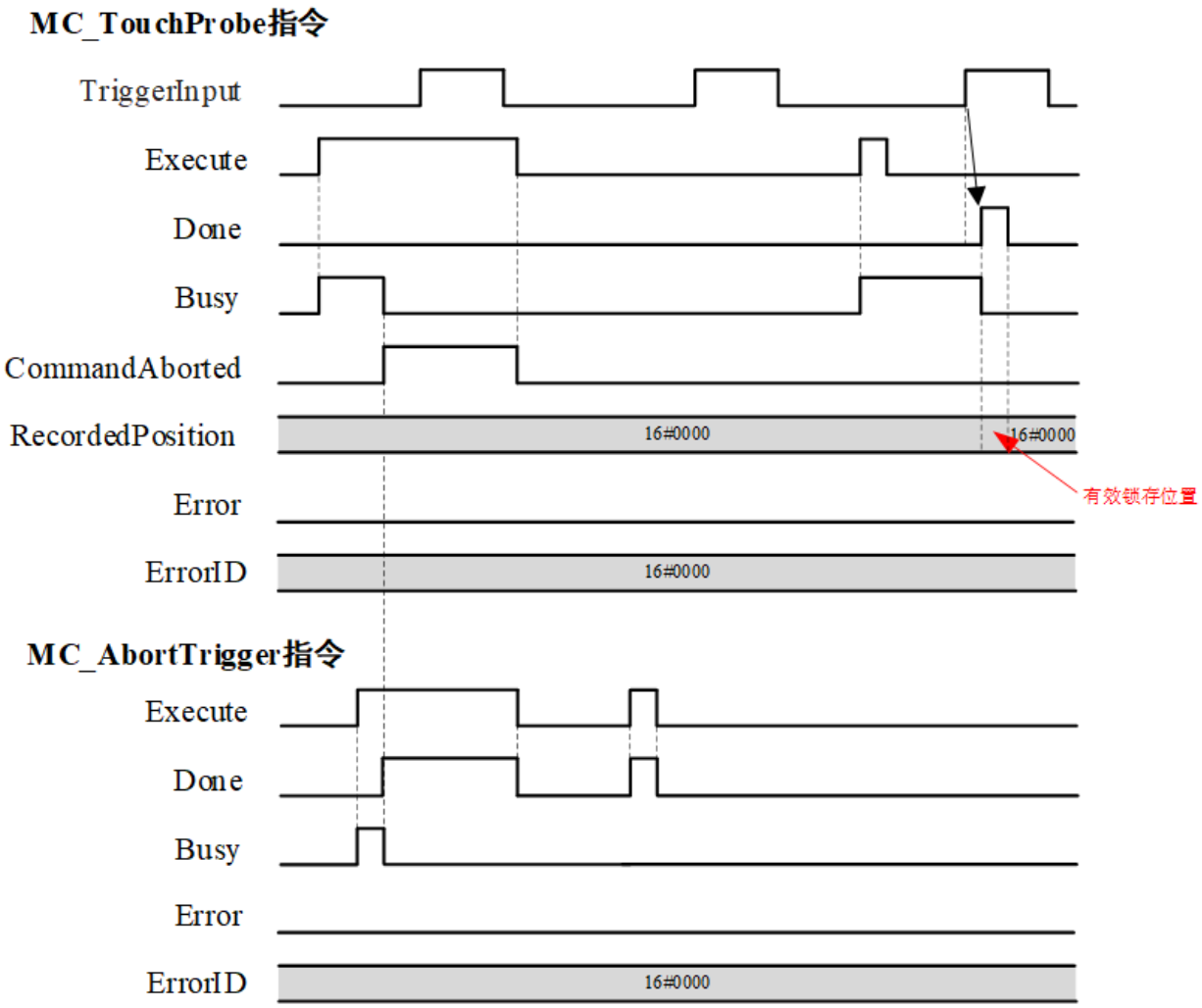
该指令只能中止 MC_TouchProbe 发起的位置锁存操作，不能中止 MC_Home 和 HMC_GantryInit 指令发起的内部位置锁存操作，该指令报错。

如果该指令在执行时，指令指定锁存通道的位置锁存已经完成，或者指定锁存通道没有执行位置锁存操作，则该指令直接正常完成。

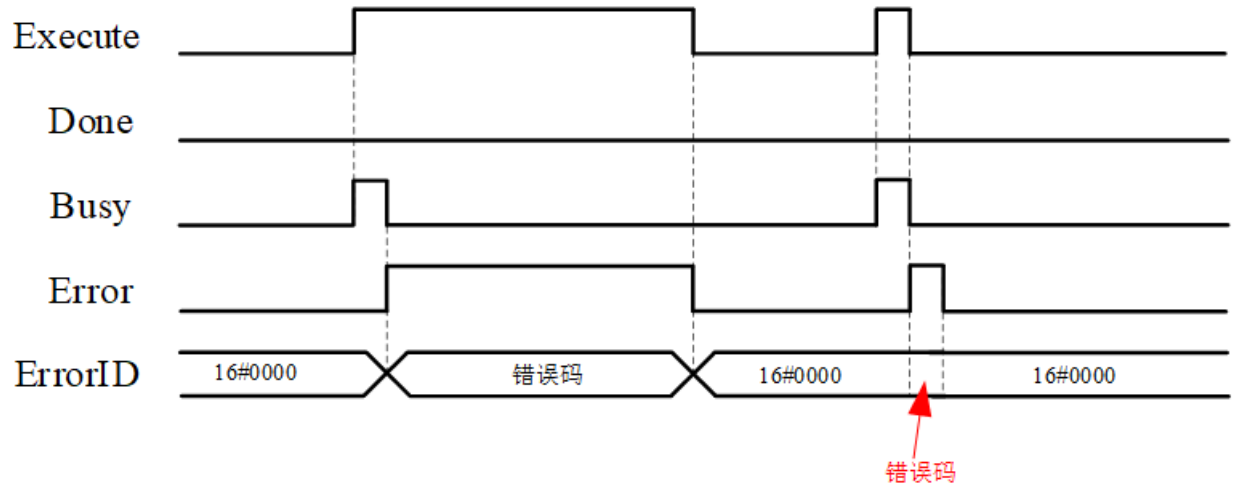
■ 功能块时序图

（1） 探针 1 上升沿有效，DI 端子触发，单次触发模式，窗口功能无效，使用 MC_AbortTrigger 中止探

针 1 位置捕获。此时序图为两个指令在同一任务中调用。



（2） MC_AbortTrigger 指令执行时发生故障的时序图。



5.26.3 示例

该示例使用 MC_AbortTrigger 指令中止 MC_TouchProbe 指令发起的位置捕获操作，打断后 MC_TouchProbe 指令 CommandAborted 引脚置 TRUE，MC_AbortTrigger 指令 Done 引脚置 TRUE。

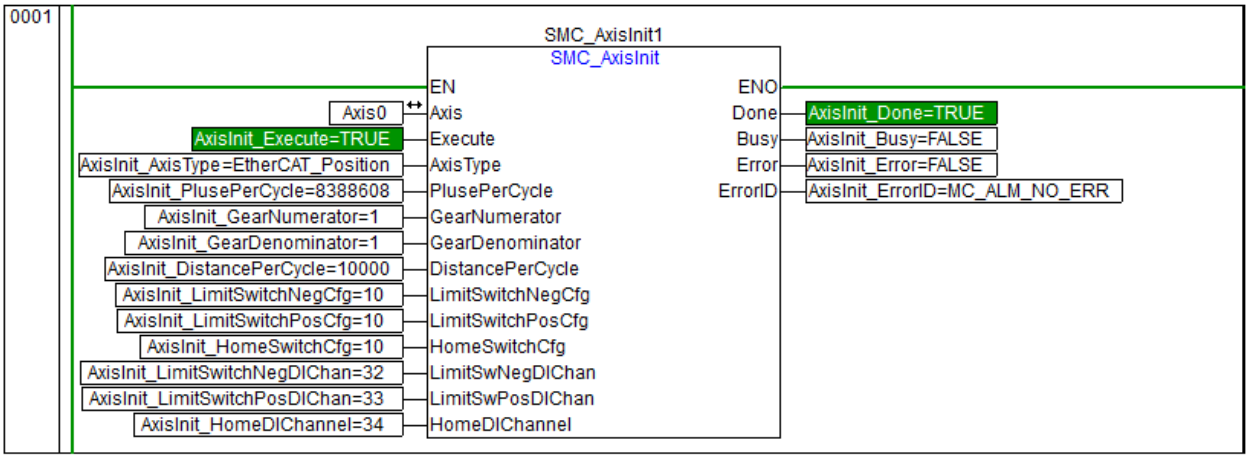
MC_TouchProbe 相关示例说明与 MC_TouchProbe 指令章节一致，本章节不再赘述。

■ 变量

序号	变量名	变量类型	初始值	变量说明
1	MC_AbortTrigger0	MC_ABORTTRIGGER		中止探针触发指令
2	AbortTrigger_En	BOOL	FALSE	中止触发指令的使能标志
3	AbortTrigger_Done	BOOL	FALSE	中止触发指令的完成标志
4	AbortTrigger_Busy	BOOL	FALSE	中止触发指令的忙标志
5	AbortTrigger_Error	BOOL	FALSE	中止触发指令的错误标志
6	AbortTrigger_Error	MC_ERROR	MC_ALM_NO_ERROR	中止触发指令的错误 ID

■ LD 程序实现

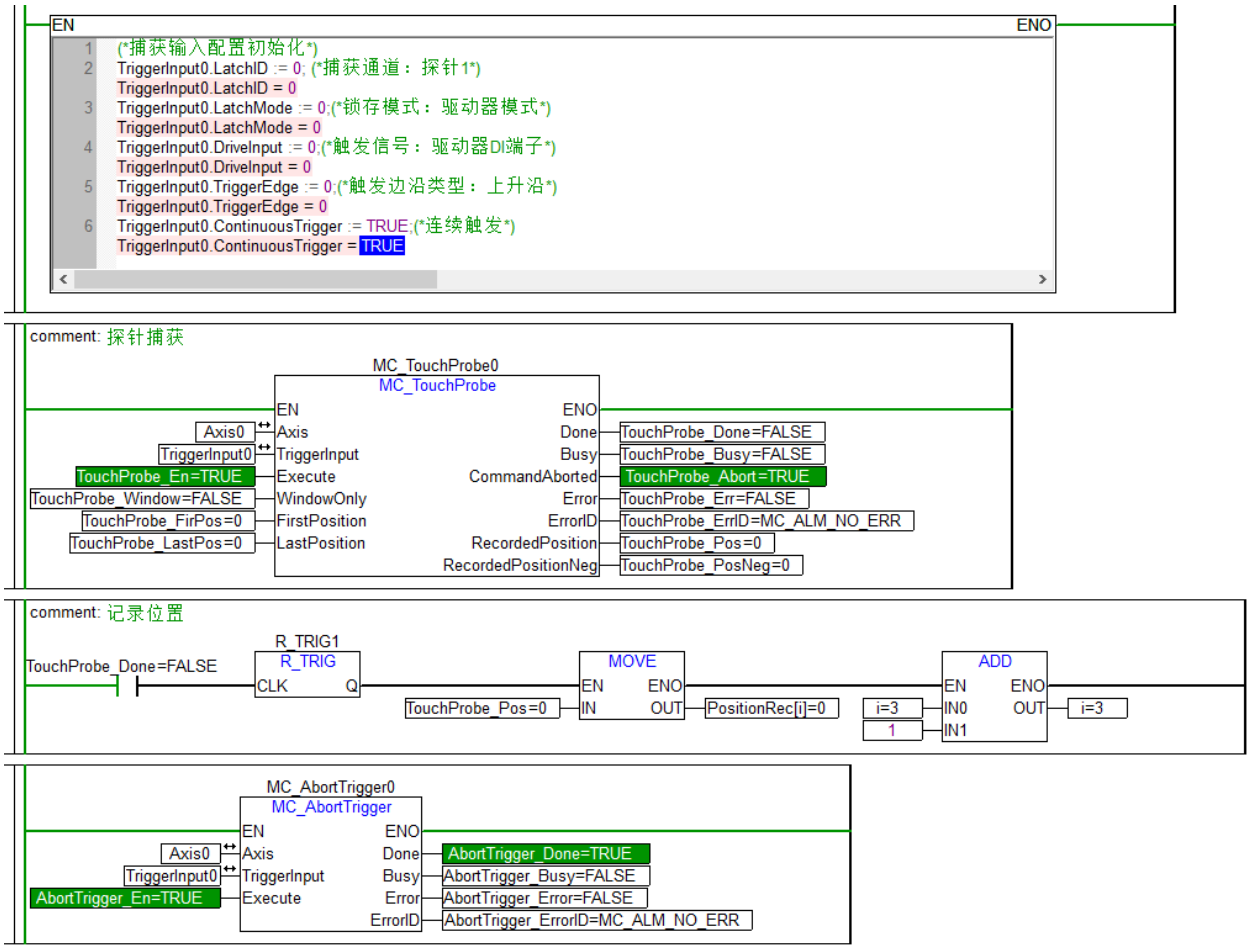
（1） 首先调用 SMC_AxisInit 指令初始化轴相关信息。



（2） 对 TriggerInput0 结构体进行初始化后，调用 MC_TouchProbe 进行探针位置捕获。

在 MC_TouchProbe 指令处于 Busy 状态时，调用 MC_AbortTrigger 指令打断探针捕获。

打断后 MC_AbortTrigger 指令 Done 为 TRUE，MC_TouchProbe 指令 CommandAborted 为 TRUE。



■ ST 程序实现

（1） 调用 MC_AxisInit 指令初始化轴相关信息。

```
MC_ReadAxisInfo1(
Enable := ReadAxisInfo_En,
Axis := Axis0,
Valid=>ReadAxisInfo_Valid,
Busy=>ReadAxisInfo_Busy,
Error=>ReadAxisInfo_Error,
ErrorID=>ReadAxisInfo_ErrorID,
HomeAbsSwitch=>ReadAxisInfo_HomeAbsSwitch,
LimitSwitchPos=>ReadAxisInfo_LimitSwitchPos,
LimitSwitchNeg=>ReadAxisInfo_LimitSwitchNeg,
Simulation=>ReadAxisInfo_Simulation,
CommunicationReady=>ReadAxisInfo_ComRdy,
ReadyForPowerOn=>ReadAxisInfo_RdyPowerON,
PowerOn=>ReadAxisInfo_PowerON,
IsHomed=>ReadAxisInfo_Homed,
AxisWarning=>ReadAxisInfo_Warning,
AxisFault=>ReadAxisInfo_Fault);
```

（2） 捕获输入配置初始化。

```
TriggerInput0.LatchID := 0; (*捕获通道：探针 1*)
TriggerInput0.LatchMode := 0;(*锁存模式：驱动器模式*)
TriggerInput0.DriveInput := 0;(*触发信号：驱动器 DI 端子*)
TriggerInput0.TriggerEdge := 0;(*触发边沿类型：上升沿*)
TriggerInput0.ContinuousTrigger := TRUE;(*连续触发*)
```

（3） 调用 MC_TouchProbe 进行探针位置捕获。

```
(*MC_TouchProbe 指令*)
MC_TouchProbe0(
   Execute := TouchProbe_En,
   WindowOnly := TouchProbe_Window,
   FirstPosition := TouchProbe_FirPos,
   LastPosition := TouchProbe_LastPos,
   Axis := Axis0,
   TriggerInput := TriggerInput0,
   Done=>TouchProbe_Done,
   Busy=>TouchProbe_Busy,
   CommandAborted=>TouchProbe_Abort,
   Error=>TouchProbe_Err,
   ErrorID=>TouchProbe_ErrID,
   RecordedPosition=>TouchProbe_Pos,
   RecordedPositionNeg=>TouchProbe_PosNeg);
```

（4） 记录捕获位置。

```
R_TRIG1(CLK:=TouchProbe_Done);
(*记录位置*)
IF R_TRIG1.Q = TRUE THEN
   PositionRec[i] := TouchProbe_Pos;
   i:=i+1;
END_IF;
```

（5） 在 MC_TouchProbe 指令处于 Busy 状态时，调用 MC_AbortTrigger 指令打断探针捕获。

打断后 MC_AbortTrigger 指令 Done 为 TRUE，MC_TouchProbe 指令 CommandAborted 为 TRUE。

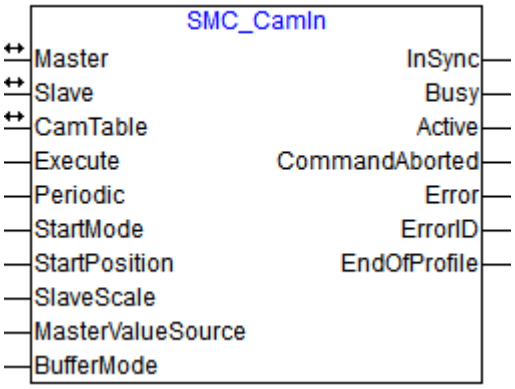
```
(*MC_AbortTrigger 指令*)
MC_AbortTrigger0(
   Execute := AbortTrigger_En,
   Axis := Axis0,
```

```
TriggerInput := TriggerInput0,
Done=>AbortTrigger_Done,
Busy=>AbortTrigger_Busy,
Error=>AbortTrigger_Error,
ErrorID=>AbortTrigger_ErrorID);
```

第6章　同步运动指令

6.1　SMC_CamIn（电子凸轮）

该指令实现电子凸轮功能。



6.1.1 参数

指令参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Master	主轴名	否	-	-	AXIS_REF
	Slave	从轴名	否	-	-	AXIS_REF
	CamTable	凸轮表	否	-	参见数据结构 SMC_CAM_TABLE	SMC_CAM_TABLE
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Periodic	FALSE: 单次模式 TRUE: 循环模式	是	FALSE	TRUE/FALSE	BOOL
	StartMode	启动模式 0：绝对位置启动（正向经过绝对位置处触发） 2：立即启动 3：DI 触发启动	是	0	0：mcAbsolute 10：mcImmediately 11：mcDITrigger	MC_CAMIN_STARTMODE
	StartPosition	绝对位置启动时，为启	是	0	绝对位置启动：	LREAL

		动位置； DI 触发启动时，为 DI 点在 I 区映射的位偏移。例如%IX100.1 在 I 区的 bit 位偏移为 100×8 + 1。			正值/负值/0; DI 触发启动： 0 ~ （I 区 byte 容量 * 8）- 1	
	SlaveScale	凸轮从轴位置数据比例因子	是	1	正值	LREAL
	MasterValueSource	主轴位置源 1: 与主轴反馈位置同步	是	1	1: mcActualValue	MC_SOURCE
	BufferMode	缓冲模式 0：打断 1：缓存	是	1	0: Aborting 1: Buffered	MC_BUFFER_MODE
OUTPUT	InSync	同步中	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为 TRUE	是	FALSE	TRUE/FALSE	BOOL
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR
	EndOfProfile	凸轮周期完成	是	FALSE	TRUE/FALSE	BOOL

SMC_CAM_TABLE 数据结构

成员	含义	默认值	范围	数据类型
PointerToPos	指向凸轮位置数组的指针（位置数组为二维数组，第一维为主轴位置，第二维为从轴位置）	0	-	POINTER TO ARRAY OF LREAL
PointsNum	位置个数	0	大于等于 3	UDINT
CurveType	凸轮曲线插值方式 1：直线 3：3 次曲线	1	1：smcStraightLine 3：smcPolynomic3	USINT

对于 SMC_CAM_TABLE 数据结构有：

（1） 凸轮位置数组为二维数组 ArrayPos[2][PointsNum]。通过选取主从轴目标运动轨迹（二维）上的离散坐标点，完成凸轮位置数组。PointerToPos 为指向该二维数组的指针。

（2） SMC_CAM_TABLE 数据结构个数需大于等于 3，并且该位置个数需与凸轮位置数组长度严格对应。

（3） 凸轮曲线插值方式支持直线插补和 3 次曲线插补两种方式。

- 注：单次模式时，凸轮启动后：对于立即启动和 DI 触发启动，指令左边界在凸轮启动时刻的位置；对于绝对位置启动，指令左边界在设定的绝对位置处；指令右边界=左边界+凸轮行程区间长度。主轴负向运动跨越左边界或正向跨越或到达右边界时，指令撤销。

6.1.2 功能

本指令实现电子凸轮功能。

■ 指令功能说明

（1） 本指令可使主轴和从轴按照凸轮位置数组进行同步凸轮动作。

（2） 本指令上升沿触发有效。在 Execute 输入的上升沿时，输入参数合法，轴无异常时，输入参数的值被锁存，指令执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令。

（3） SMC_CAM_TABLE 数据结构中的凸轮位置数组需要在运行本指令前完成编制。

（4） 根据选取主、从轴目标运动轨迹（二维）上的离散坐标点编制凸轮位置数组，凸轮通过主从轴联动的方式逼近目标运动轨迹。

（5） 主轴位置源。主轴位置源 MasterValueSource 仅可设为 1，从轴指令位置与主轴的反馈位置同步。

（6） 凸轮从轴支持以下轴类型：

- □ **Virtual（0）**：虚轴
- □ **EtherCAT_Position（50）**：EtherCAT 总线连接轴，位置控制
- □ **EtherCAT_Speed（51）**：EtherCAT 总线连接轴，速度控制
- □ **EtherCAT_Torque（52）**：EtherCAT 总线连接轴，转矩控制

■ 凸轮位置数组说明

（1） 凸轮位置数组主、从轴位置关系：

凸轮主、从轴位置预先存储在主从轴位置数组中。二维凸轮位置数组第一维为主轴位置，第二维

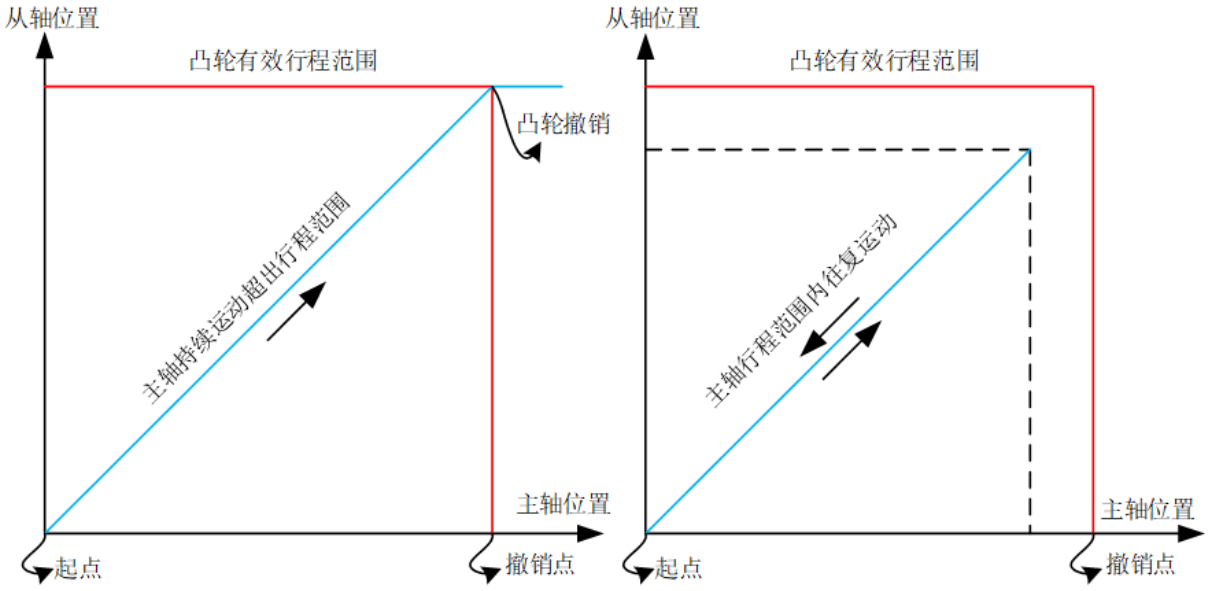
为从轴位置，二者为一一对应关系。

（2） 凸轮行程

凸轮主轴的一个行程长度为凸轮主轴位置数组的最后一个元素值减去第一个元素值。

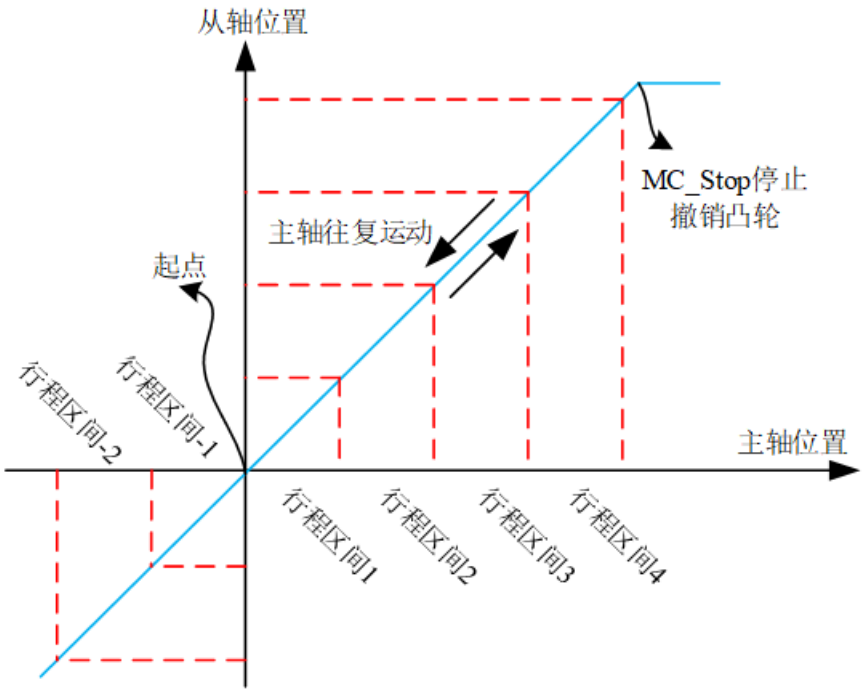
单次模式时，凸轮启动后：对于立即启动和 DI 触发启动，指令左边界在凸轮启动时刻的位置；对于绝对位置启动，指令左边界在设定的绝对位置处；指令右边界 = 左边界+凸轮行程区间长度。主轴负向运动跨越左边界或正向跨越或到达右边界时，指令撤销。

单次凸轮主轴在有效行程范围内时，主轴可进行单向、往复运动，当主轴在有效行程范围内往复运动时，从轴也会进行往复性的联动动作。如下图所示单次凸轮立即启动的主从轴运动轨迹。



单次立即启动凸轮主从轴联动运动轨迹

对于循环凸轮，当凸轮主轴位置越过一个凸轮主轴行程长度时，将会进入下一个凸轮行程，此时从轴将会以前一个凸轮行程结束时的从轴位置为起点，在此基础上作增量位移计算，重复上一个凸轮行程的联动动作，周而复始。凸轮主轴也可做单向、往复运动。如果需要撤消循环凸轮，可执行 MC_Stop 指令停止和撤销凸轮动作，详细使用方法请参考 MC_Stop 指令说明。如下图所示循环凸轮立即启动的主从轴运动轨迹。



循环立即启动凸轮主从轴联动运动轨迹

(3) 凸轮启动位置

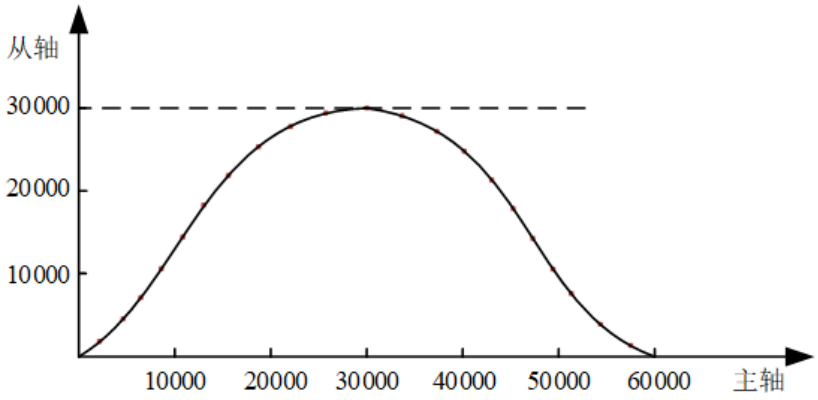
启动指令后，等待主轴到达 StartPosition。

主轴通过 StartPosition 时，执行凸轮位置数组的起点，凸轮指令输出变量 InSync （凸轮同步中）变为 TRUE。

凸轮位置数组的主轴位置和从轴位置均以起点的相对量指定。根据设定的凸轮曲线插值方式对两个凸轮位置数据之间进行插补，计算出主轴位置对应的从轴位置，如下图所示。

凸轮表

主轴	从轴
0	0
10000	12000
20000	21000
30000	30000
40000	25000
50000	11000
60000	0



(4) 凸轮位置数组异常

控制器中不存在指定凸轮位置数组时，触发本指令检测到异常。

(5) 凸轮位置数组的使用

多个轴可指定同一个凸轮位置数组。注意：本指令执行过程中，用户改写凸轮位置数组主轴的位置为非递增的数据时，运行本指令会发生错误异常。

(6) 凸轮启动

在 StandStill 状态、位置控制中、速度控制中、同步控制中的任一状态下都可以启动本指令。

凸轮启动模式有 0：固对位置启动，10：立即启动，11：DI 触发启动。

□ 固对位置启动

凸轮启动模式为 0：固对位置启动。启动指令后，等待凸轮主轴 Mpos 从负向跨过凸轮启动位置 StartPosition 处，执行凸轮表的起点。输出变量 InSync(同步中)变为 TRUE。

凸轮位置数组以相对量指定主轴和从轴位置，即是以凸轮位置数组起点处各轴的绝对位置为起点的相对值。例如，主轴的运动行程为[0～360°)的循环行程模式时，StartPosition 为 100 时。如下所示，主轴的绝对位置是凸轮位置数组的主轴位置加上 StartPosition 的值，从轴的绝对位置是凸轮位置数组的从轴位移加上凸轮位置数组起点的从轴绝对位置的值。

凸轮位置数组

主轴	从轴
0	0
60	1000
120	1500
180	2000
240	1500
300	1000
360	0

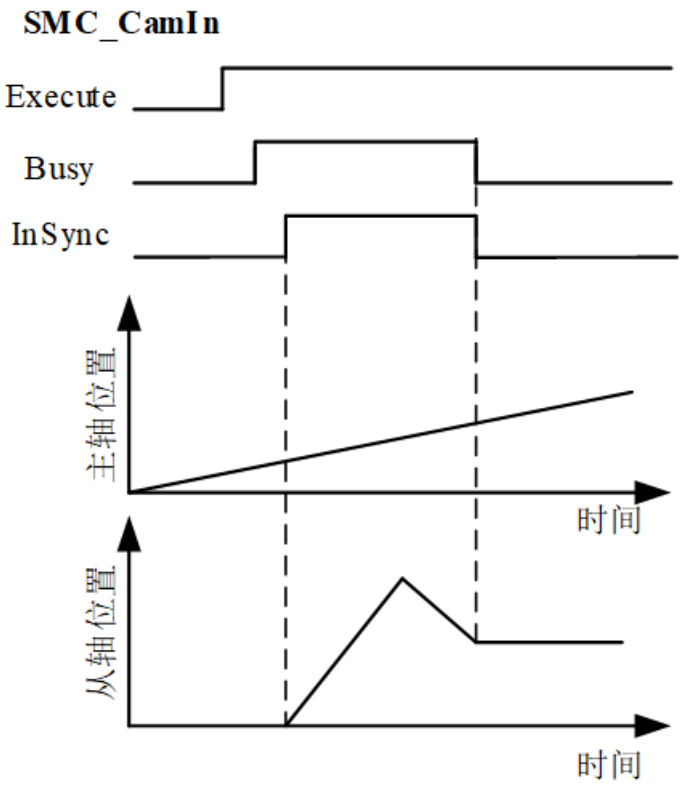
StartPosition=100

主轴、从轴的绝对位置

主轴	从轴
100	0+从轴在凸轮位置数组起点处的绝对位置
160	1000+从轴在凸轮位置数组起点处的绝对位置
220	1500+从轴在凸轮位置数组起点处的绝对位置
280	2000+从轴在凸轮位置数组起点处的绝对位置
340	1500+从轴在凸轮位置数组起点处的绝对位置
40	1000+从轴在凸轮位置数组起点处的绝对位置
100	0+从轴在凸轮位置数组起点处的绝对位置

□ 立即启动

凸轮启动模式为 10：立即启动。启动指令后，立即开始执行凸轮动作。此时 StartPosition 无意义。



□ DI 触发启动

凸轮启动模式为 11：DI 触发启动。凸轮指令投放后，当设定的某 DI 信号为 1 时启动凸轮，启动位置为程序检测到 DI 为 1 时主轴 Mpos 位置。StartPosition 为 DI 点在 I 区映射的位偏移。例如%IX100.1 在 I 区的 bit 位偏移为 100×8 + 1，即 StartPosition=801。

DI 触发启动时，启动位置 StartPosition 的值需小于运行控制器 I 区的容量。

（7） 周期模式

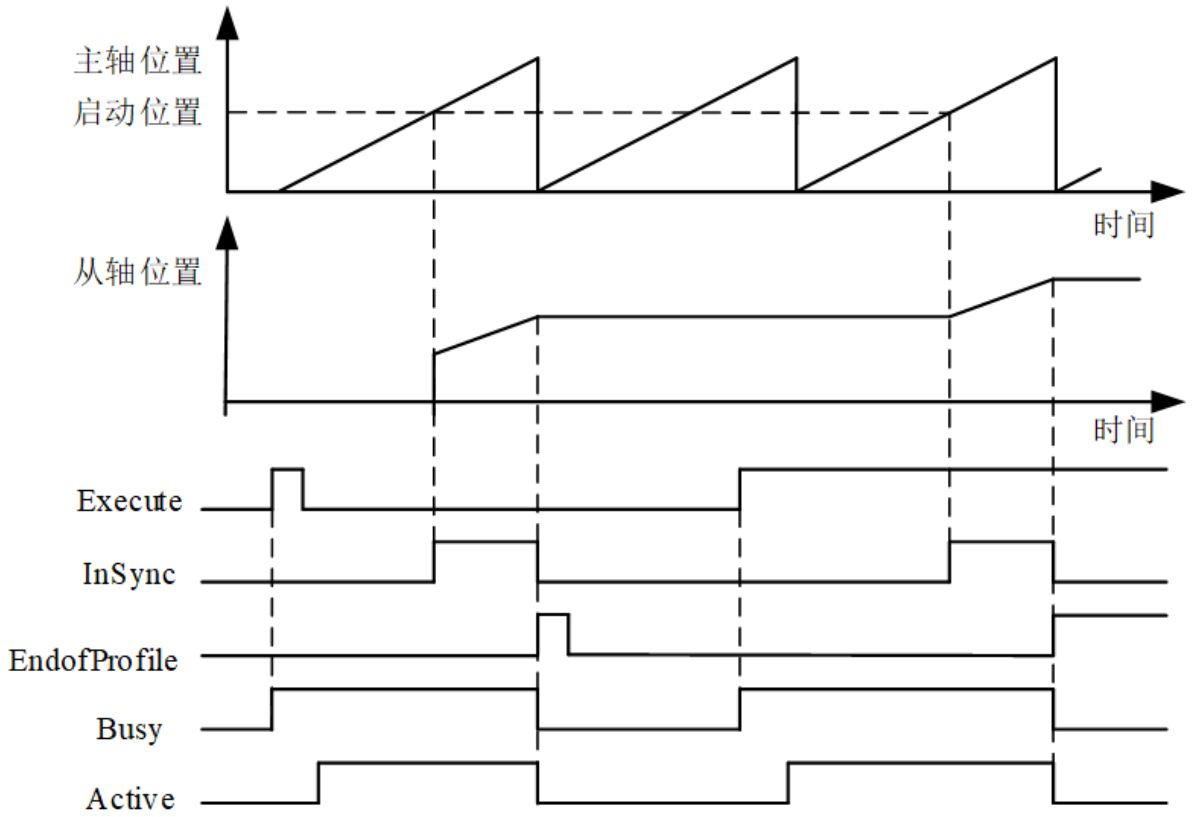
按主轴位置的边界范围划分，凸轮分为单次凸轮和循环凸轮，由输入参数 Periodic 确定。单次模式，Periodic=FALSE，主轴负向运动跨越左边界或正向跨越右边界后，指令撤销。

循环凸轮，Periodic=TRUE，当凸轮主轴位置越过一个凸轮主轴行程长度时，将会进入下一个凸轮行程，周而复始，持续运行。

□ 单次凸轮

凸轮启动后，若凸轮主轴位置在有效行程范围内，则凸轮可保持正常联动运行，主轴超过有效行程范围时凸轮指令将自动撤销。凸轮启动后，满足凸轮触发条件后，凸轮指令 InSync 引脚置 TRUE，从轴开始运动，单次凸轮指令运行完成后，EndofProfile 引脚置 TRUE 且至少保持一个周期，Busy 和 Active 引脚变为 FALSE，遇到 Execute 下降沿后，所有输出引脚恢复为默认值。

下图为单次模式，固定位置启动时的引脚时序：



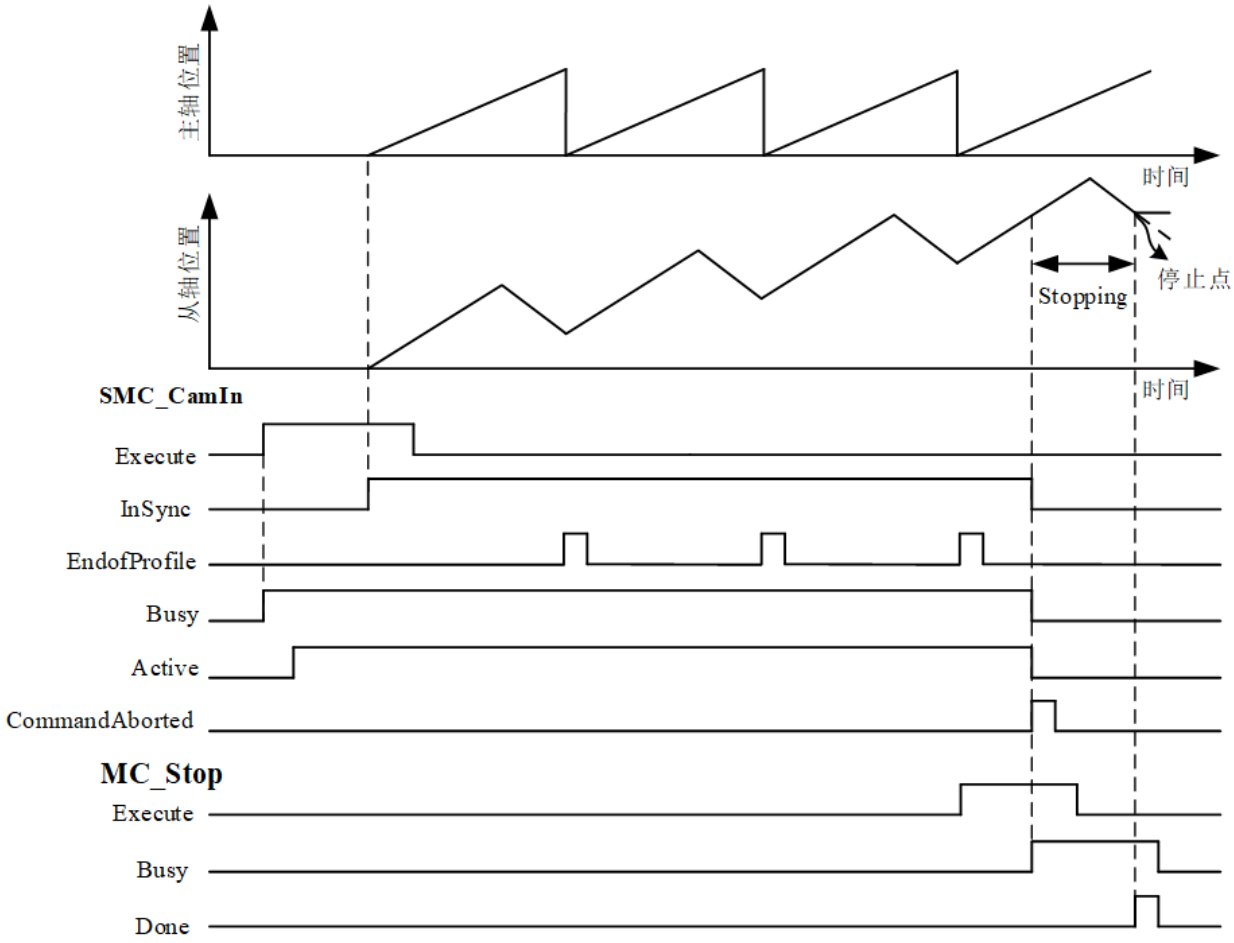
□ 循环凸轮

循环凸轮主轴位置没有边界范围限制，启动后若不执行停止撤销指令，凸轮功能将一直有效。

凸轮从轴跟随主轴作相应的联动动作，当主轴一个凸轮行程结束时，进入下一个凸轮行程，从轴以上一个凸轮行程的终点为起点作增量计算，执行周期性的联动动作。每一个凸轮行程结束时，输出引脚 EndofProfile 置一个周期的高电平。

利用 MC_Stop 指令撤销停止循环凸轮指令时，此时凸轮输出引脚 CommandAborted 置 TRUE，其余输出引脚复位。从轴沿着凸轮运动轨迹减速停止。

下图为循环模式，立即启动模式的凸轮指令引脚时序，利用 MC_Stop 指令撤销停止循环凸轮指令。



（8） 功能块重复触发的处理

□ 同一凸轮指令功能块重复触发

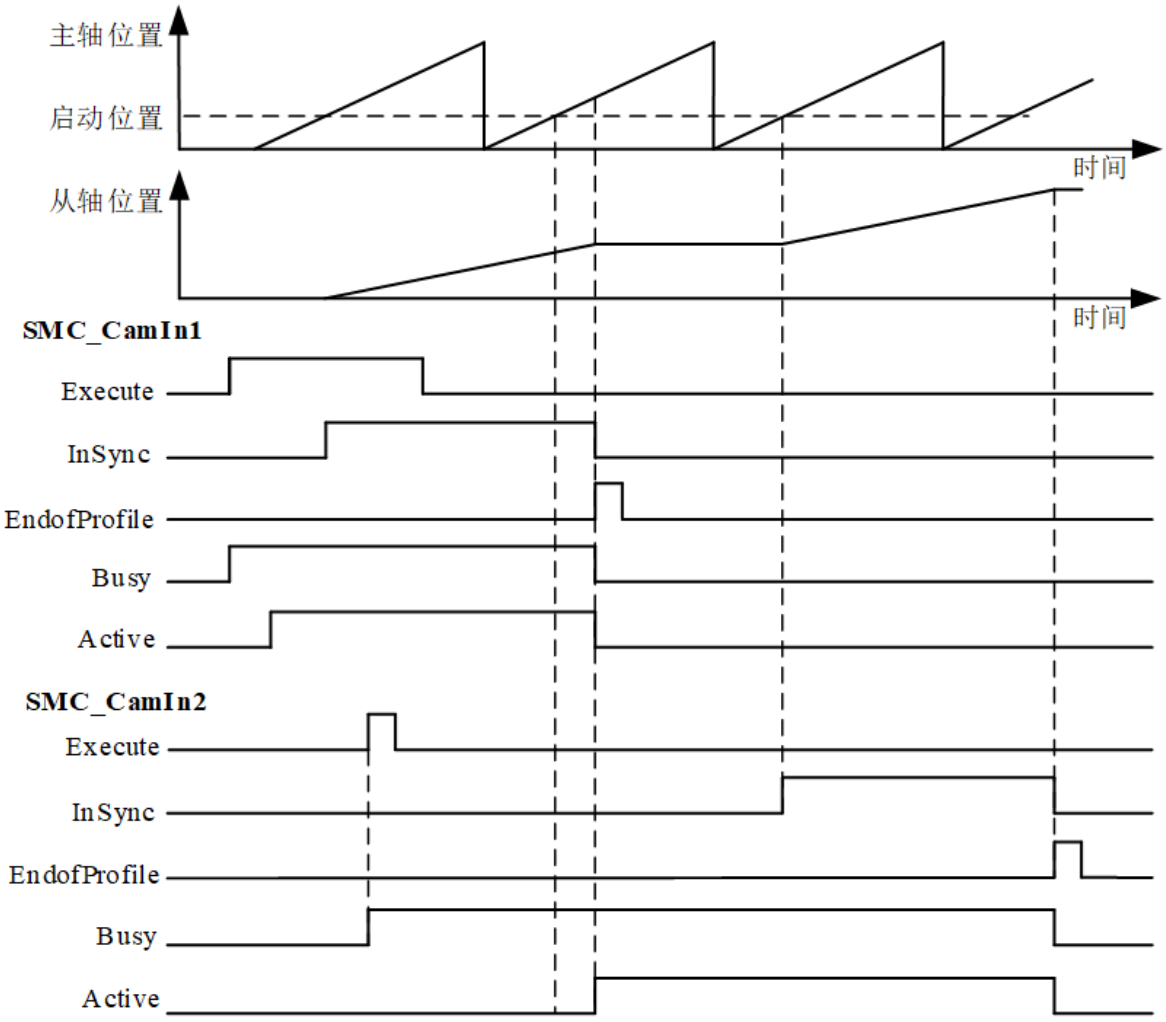
对于运行中的同一凸轮指令功能块，如果重复触发执行，则有：

（a） 若指令的 BufferMode 为 0：Aborting，则打断当前正在执行的指令；

（b） 若指令的 BufferMode 为 1：Buffered，则指令被放入运动缓存中，之后功能块监控的状态为后一条指令的运行状态，前一条指令的运行状态则失去监控。

□ 缓存凸轮指令等待运行

在前一个凸轮指令执行期间，后一个凸轮指令等待前一个凸轮指令结束后，开始执行。如下图示例，在执行 SMC_CamIn1（单次凸轮，固定位置启动）指令期间，重复启动 SMC_CamIn2 指令。此时，在 SMC_CamIn1 的 EndOfProfile 引脚变为 TRUE、SMC_CamIn2 的 Active 引脚变为 TRUE 的下一 StartPosition(凸轮启动位置)，InSync(同步中)变为 TRUE，开始凸轮动作。



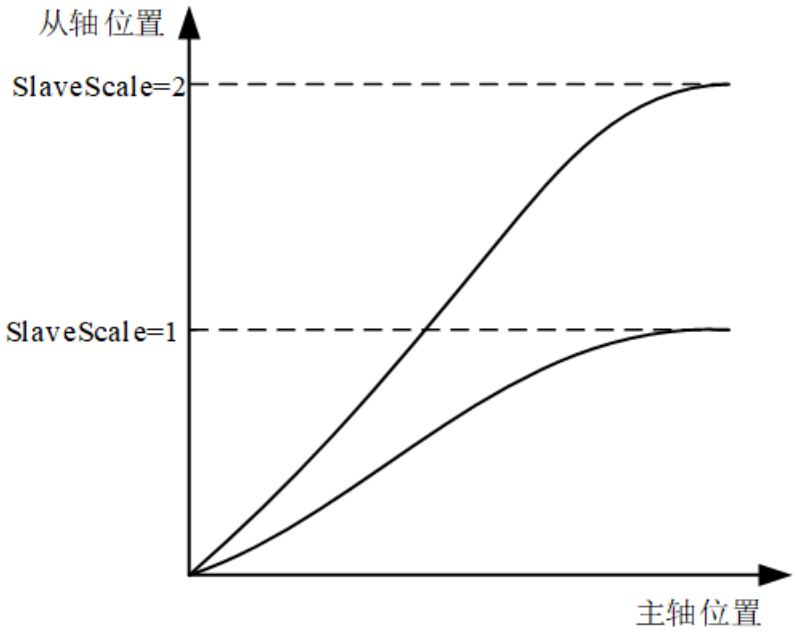
(9) 缓存模式选择

指定前一个轴动作和本次动作的连接方式。

本指令 BufferMode 仅支持 0：Aborting 和 1：Buffered。投送本指令时，若 BufferMode 为 0：Aborting，立即中止当前正在执行的指令，切换为本指令。若 BufferMode 为 1：Buffered，从当前执行中的指令正常结束的周期起，已缓存的本指令自动启动。

(10) 凸轮从轴位置数据比例因子（缩放系数）

针对指定凸轮数组的主轴相位、从轴位移关系，修改凸轮从轴位置数据比例因子 SlaveScale 的值，可以使从轴的位移以指定的比例（SlaveScale）放大或者缩小。



凸轮从轴位置数据比例因子 SlaveScale

（11） EndofProfile（凸轮周期完成）

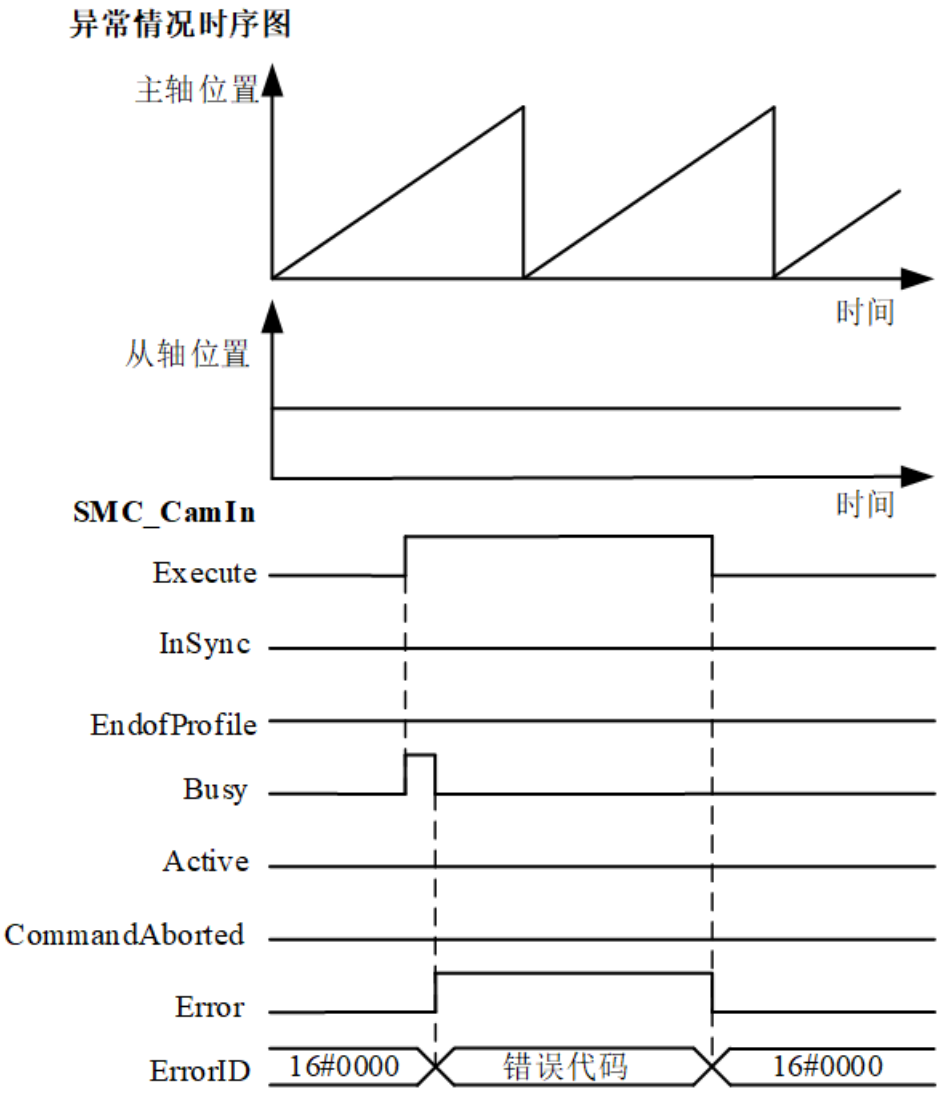
凸轮指令为单次模式时，凸轮周期完成，指令输出引脚 EndofProfile 置 TRUE 保持，直到输入引脚 Execute 变为 FALSE 时，引脚 EndofProfile 变为 FALSE。

凸轮指令为周期模式时，第一个凸轮行程完成，指令输出引脚 EndofProfile 置 TRUE 一个周期，指令进入下一个凸轮行程，EndofProfile 变为 FALSE，直到该凸轮行程完成，引脚 EndofProfile 置 TRUE 一个周期，EndofProfile 随着周期性凸轮持续变化更新。可利用 MC_Stop 指令撤销停止循环凸轮指令。

EndofProfile 引脚时序详情见上述时序图。

（12） 异常

启动凸轮指令时，若发生异常错误，则 Error 引脚置为 True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Execute 变为低电平时，所以输出引脚恢复为默认值。



6.1.3 示例

本示例中，定义 MasterAxis 为主轴，SlaveAxis 为从轴， 变量类型为 AXIS_REF。

主轴执行 MC_MoveVelocity 指令、从轴执行 MC_CamIn 指令。使用 MC_Stop 指令停止撤销凸轮指令。

■ 梯形图程序

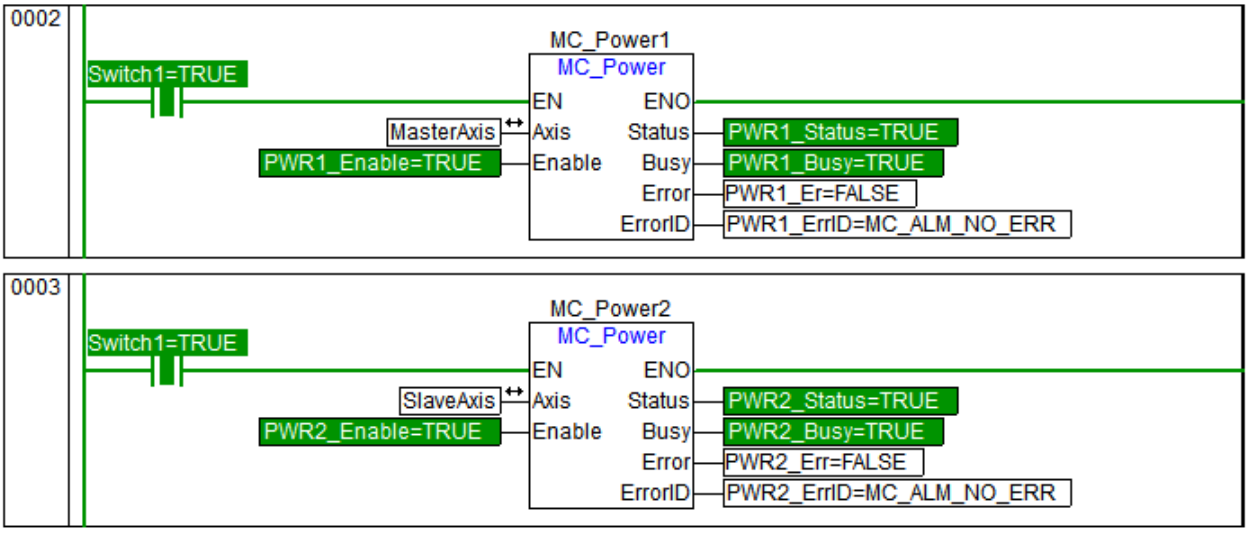
本示例组建周期模式，立即启动的凸轮指令。将 Periodic(重复模式)设为 TRUE，执行重复动作。

（1） 示例程序主要变量

变量名称	变量类型	初始值	注释
MasterAxis	AXIS_REF	-	主轴的轴变量
SlaveAxis	AXIS_REF	-	从轴的轴变量
Power1_Status	BOOL	FALSE	主轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	从轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
MV_Exe	BOOL	FALSE	轴的速度控制指令的上升沿触发变量。
MV_InVel	BOOL	FALSE	轴速度控制指令达到目标速度的标识变量。该变量变为 TRUE，速度指令的速度达到目标速度。
CamTable	SMC_CAM_TABLE	-	凸轮表变量
ArrTablePos	ARRAY OF LREAL	-	凸轮位置数组。
PointerToPos	POINTER TO ARRAY OF LREAL	-	指向凸轮位置数组的指针变量，数组位置 Pos[2][PosNum]。PosNum 为位置个数。
Cam_InSync	BOOL	FALSE	投送的从轴的凸轮指令的同步变量，凸轮指令从轴同步时，该变量变为 TRUE。
Cam_EndProFile	BOOL	FALSE	投送的从轴的凸轮指令的凸轮周期完成的变量，凸轮指令凸轮周期完成时，该变量变为 TRUE。
Switch1	BOOL	FALSE	轴准备就绪的状态变量，轴初始化完成后， Switch 为 TRUE，轴准备就绪。

（2） 轴初始化。设置主轴 MasterAxis 的 AxisID 为 0，从轴 SlaveAxis 的 AxisID 为 1，主、从轴的轴类型均为 50：EtherCAT_Position，调用 SMC_AxisInit 指令完成主、从轴的初始化配置。SMC_AxisInit 指令使用方法参见其介绍章节。

（3） 轴使能。0002 节中，主、从轴初始化完成，轴准备就绪状态 Switch1 为 TRUE 后，执行轴使能指令，完成轴使能。

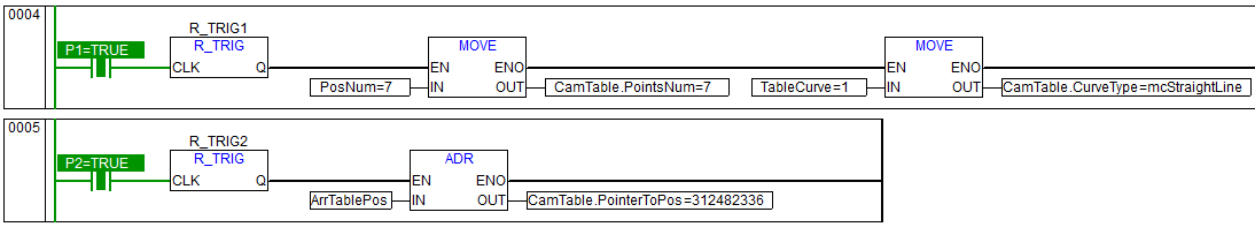


（4） 初始化 SMC_CAM_TABLE 数据结构。0004 节设置凸轮位置数组位置个数，以及凸轮插补曲线方式，0005 节中对凸轮位置数组取地址。完成凸轮数据结构变量 CamTable 的初始化。

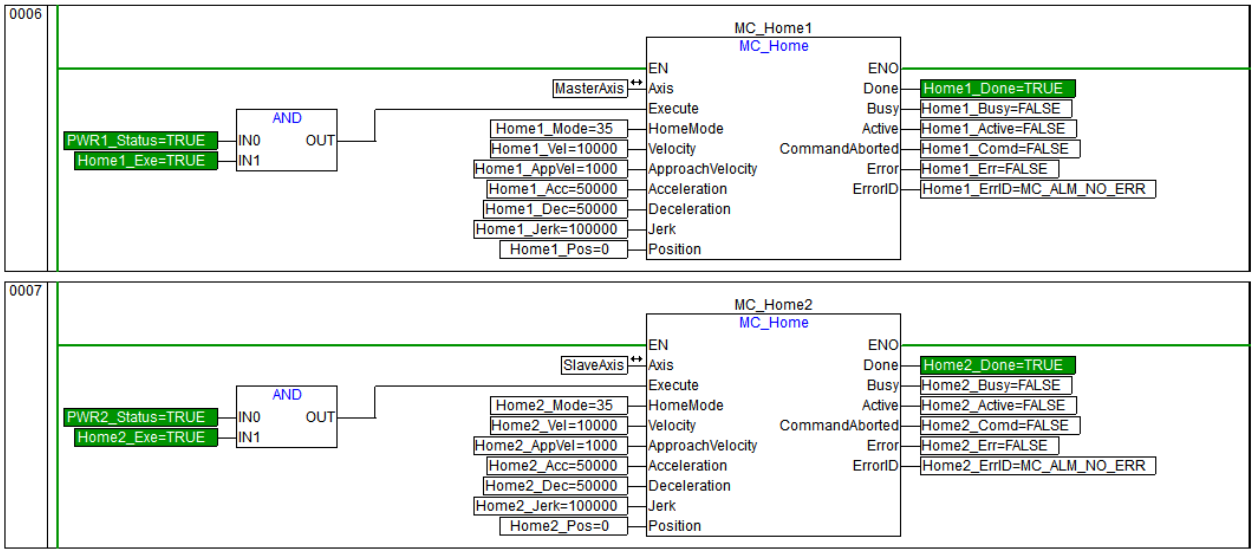
ArrTablePos [0][PosNum] ={0,60,120,180,240,300,360};

ArrTablePos [1][PosNum] ={0,1000, 2000,3000,2500,2000,1500};

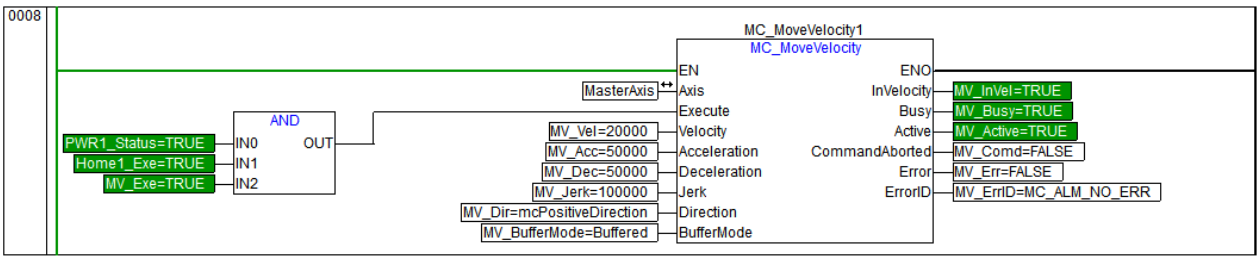
PosNum=7;



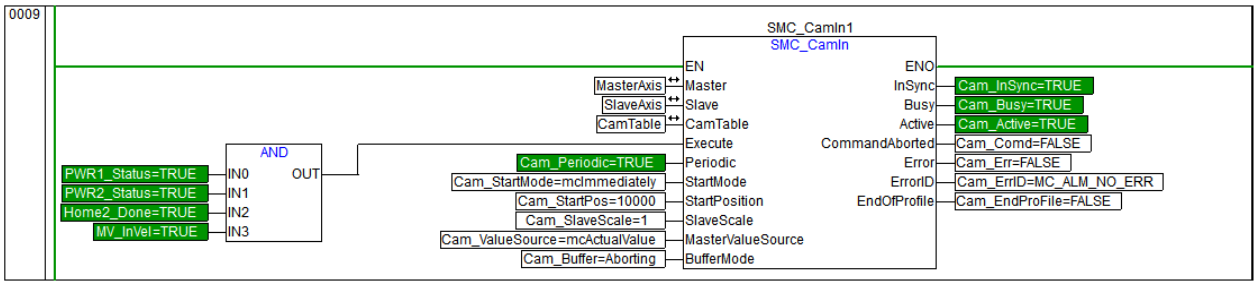
（5） 主、从轴使能成功，若原点未确定，则执行主从轴的原点回归指令，确定原点，原点回零见 0006 和 0007 节，MC_Home 指令使用方法见其介绍章节。



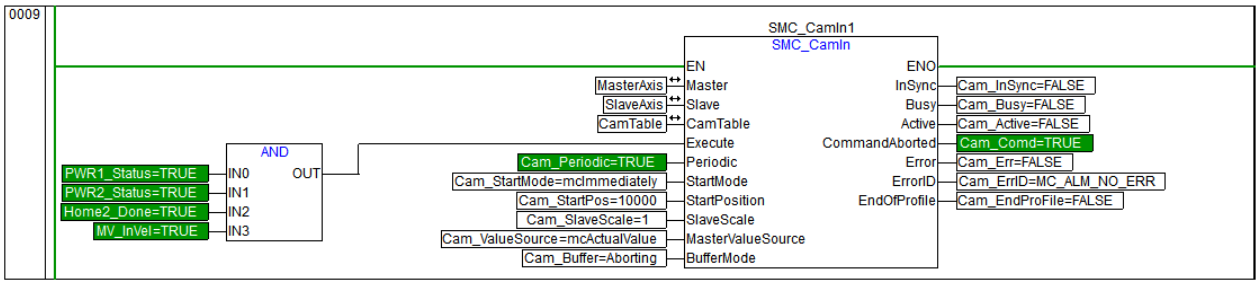
（6） 见 0008 节，原点回归完成后，执行 MC_MoveVelocity(速度控制)指令。主轴开始运动。

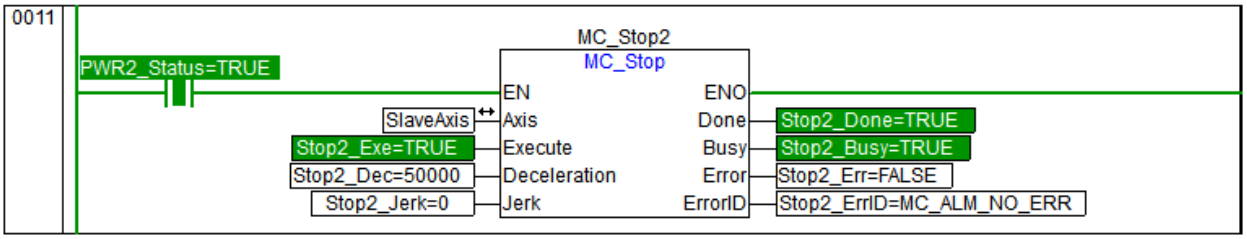


（7） 见 0009 节，当主轴速度指令 MC_MoveVelocity 的 MV_InVel 引脚变为 TRUE 后，启动从轴的 SMC_CamIn1 指令。凸轮指令为循环立即启动，Cam_InSync 引脚为 TRUE。

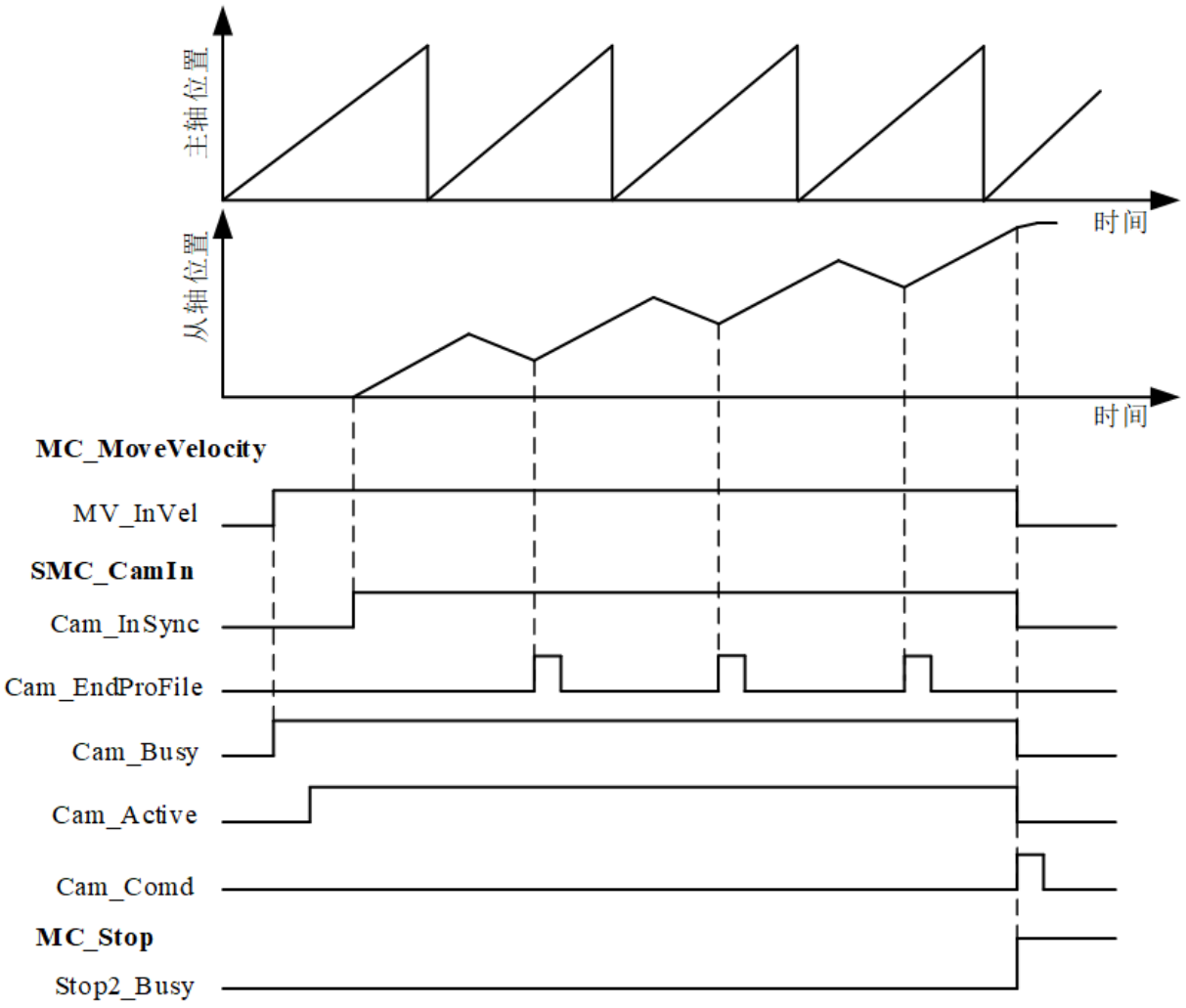


（8） 对于循环凸轮，可以使用 MC_Stop 撤销并停止从轴的凸轮运动指令。如下 0011 节中，上升沿触发 MC_Stop2 指令，从轴的凸轮指令被中断，0009 节中凸轮的 Cam_Comd 引脚变为 TRUE。





本示例的部分变量时序图如下：



■ ST 语言程序

指令设为单次模式，绝对位置启动，凸轮启动固定位置为 10000。主轴位置从负向跨过凸轮启动位置 10000 处，开始凸轮动作。

（1） 主要变量

变量名称	变量类型	初始值	注释
MasterAxis	AXIS_REF	-	主轴的轴变量
SlaveAxis	AXIS_REF	-	从轴的轴变量
Power1_Status	BOOL	FALSE	主轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	从轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
MV_Exe	BOOL	FALSE	轴速度控制指令的上升沿触发变量。
MV_InVel	BOOL	FALSE	轴速度控制指令达到目标速度的标识变量。该变量变为 TRUE，速度指令的速度达到目标速度。
CamTable	SMC_CAM_TABLE	-	凸轮表变量
ArrTablePos	ARRAY OF LREAL	-	凸轮位置数组。
PointerToPos	POINTER TO ARRAY OF LREAL	-	指向凸轮位置数组的指针变量，数组位置 Pos[2][PosNum]。PosNum 为位置个数。
Cam_InSync	BOOL	FALSE	凸轮指令的同步变量，凸轮指令从轴同步时，该变量变为 TRUE。
Cam_EndProFile	BOOL	FALSE	凸轮指令的凸轮周期完成的变量，凸轮指令凸轮周期完成时，该变量变为 TRUE。
Switch1	BOOL	FALSE	轴准备就绪的状态变量，轴初始化完成后， Switch 为 TRUE，轴准备就绪。
InitFlag	BOOL	FALSE	如果该变量为 TRUE，则初始化参数设置完成。

（2） 轴初始化

设定主轴 MasterAxis 的 AxisID 为 0，从轴 SlaveAxis 的 AxisID 为 1。调用 SMC_AxisInit 指令完成主从轴的初始化配置，主、从轴类型均为 50：EtherCAT_Position，调用 SMC_AxisInit 指令完成主、从轴的初始化配置。

（3） 轴使能

主、从轴初始化完成，轴准备就绪状态 Switch1 为 TRUE 后，高电平触发轴使能指令，完成轴使能。

```
IF  Switch1 THEN
   MC_Power1(
   Enable := PWR1_Enable,
   Axis := MasterAxis,
```

```
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
Error=> PWR1_Err,
ErrorID=> PWR1_ErrID);
MC_Power2(
   Enable := PWR2_Enable,
   Axis := SlaveAxis,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
END_IF
```

（4） 初始化 SMC_CAM_TABLE 数据结构

设置凸轮位置数组位置个数，以及凸轮插补曲线方式，对凸轮位置数组取地址，完成凸轮数据结构变量 CamTable 的初始化。

```
IF FALSE=InitFlag THEN
CamTable.PointsNum := PointsNum ;                    (*位置数组个数*)
ArrTablePos[0,0]:=0; (*初始化位置凸轮数组*)
ArrTablePos[0,1]:=10000; (*初始化位置凸轮数组*)
…
…
ArrTablePos[0, PointsNum-1]:=60000; (*初始化位置凸轮数组*)
ArrTablePos[1,0]:=0; (*初始化位置凸轮数组*)
ArrTablePos[1,1]:=5000; (*初始化位置凸轮数组*)
…
…
ArrTablePos[1, PointsNum-1]:=10000; (*初始化位置凸轮数组*)
   CamTable.PointsNum := CurveType ;                  (*凸轮曲线插值方式*)
CamTable.PointerToPos := ADR(ArrTablePos);       (*凸轮位置数组取地址*)
InitFlag:=TRUE;
END_IF
```

（5） 执行主从轴的原点回归指令，确定原点，MC_Home 指令使用方法见其介绍章节。

```
IF  PWR1_Status THEN
   MC_Home1(
   Execute := Home1_Exe,
   HomeMode := Home1_Mode,
   Velocity := Home1_Vel,
```

```
    ApproachVelocity := Home1_AppVel,
    Acceleration := Home1_Acc,
    Deceleration := Home1_Dec,
    Jerk := Home1_Jerk,
    Position := Home1_Pos,
    Axis := MasterAxis,
    Done=> Home1_Done,
    Busy=> Home1_Busy,
    Active=> Home1_Active,
    CommandAborted=> Home1_Comd,
    Error=> Home1_Err,
    ErrorID=> Home1_ErrID);
    MC_Home2(
    Execute := Home2_Exe,
    HomeMode := Home2_Mode,
    Velocity := Home2_Vel,
    ApproachVelocity := Home2_AppVel,
    Acceleration := Home2_Acc,
    Deceleration := Home2_Dec,
    Jerk := Home2_Jerk,
    Position := Home2_Pos,
    Axis := SlaveAxis,
    Done=> Home2_Done,
    Busy=> Home2_Busy,
    Active=> Home2_Active,
    CommandAborted=> Home2_Comd,
    Error=> Home2_Err,
    ErrorID=> Home2_ErrID);
END_IF
```

（6） 从轴投送凸轮指令，单次固定位置启动，主轴从固定位置的负向到正向时，凸轮动作。

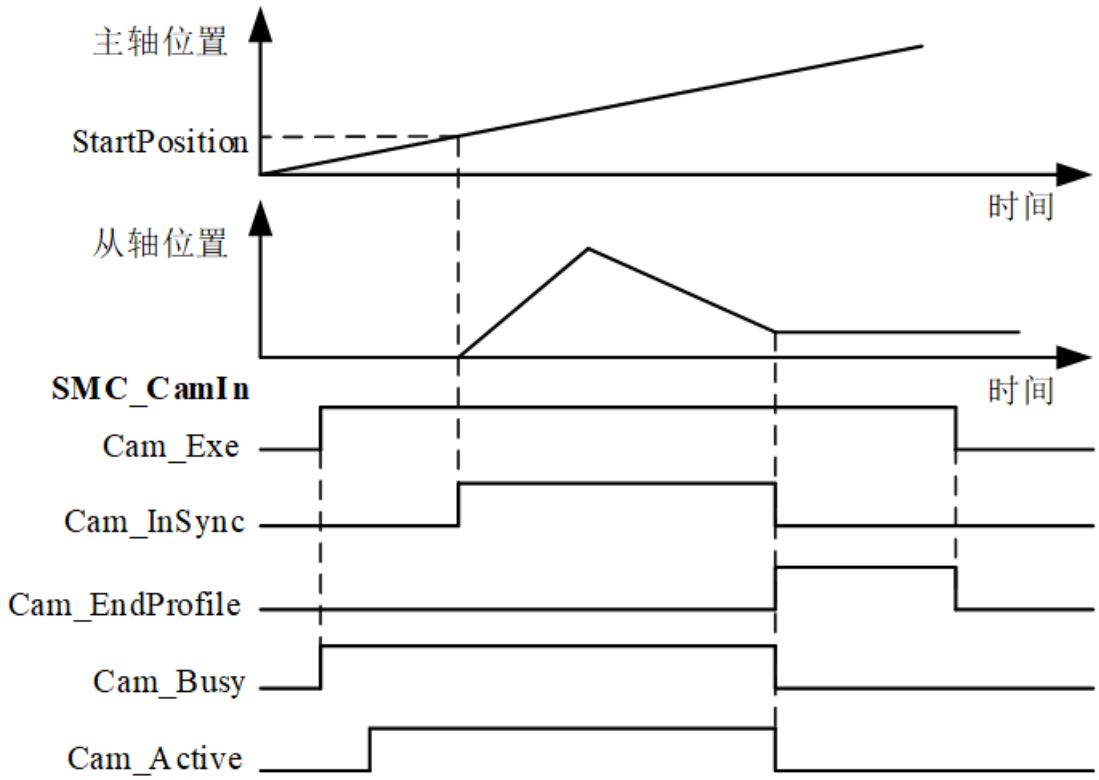
```
IF  PWR1_Status AND PWR2_Status AND Home1_Done THEN
    SMC_CamIn1(
    Execute := Cam_Exe,
    Periodic := Cam_Periodic,
    StartMode := Cam_StartMode,
    StartPosition := Cam_StartPos,
    SlaveScale := Cam_SlaveScale,
    MasterValueSource := Cam_ValueSource,
    BufferMode := Cam_Buffer,
    Master := MasterAxis,
    Slave := SlaveAxis,
    CamTable := CamTable,
    InSync=> Cam_InSync,
    Busy=> Cam_Busy,
```

```
    Active=> Cam_Act,
    CommandAborted=> Cam_Comd,
    Error=> Cam_Err,
    ErrorID=> Cam_ErrID,
    EndOfProfile=> Cam_EndPro);
END_IF
```

（7） 投送主轴运动指令

```
IF  PWR1_Status AND Home1_Done THEN
    MC_MoveVelocity1(
    Execute := MV_Exe,
    Velocity := MV_Vel,
    Acceleration := MV_Acc,
    Deceleration := MV_Dcc,
    Jerk := MV_Jerk,
    Direction := MV_Dir,
    BufferMode := MV_Buffer,
    Axis := Axis0,
    InVelocity=> MV_InVel,
    Busy=> MV_Busy,
    Active=> MV_Active,
    CommandAborted=> MV_Comm,
    Error=> MV_Err,
    ErrorID=> MV_Error);
END_IF
```

下图为单次模式，立即启动凸轮的位置曲线和引脚时序。



6.1.4 使用注意事项

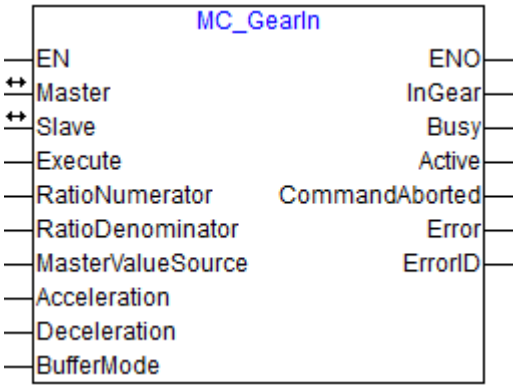
- 主轴位置数组中的元素必须是单调递增的（不可有相同数），数据正负均可。
- 本指令执行过程中，用户改写凸轮位置数组主轴的位置为非递增的数据时，运行本指令会发生错误异常。
- 凸轮位置数组位置个数大于等于 3。凸轮位置数组位置个数需与凸轮位置数组长度严格对应。
- 凸轮为绝对位置启动时，主轴正向经过绝对位置处触发凸轮指令。
- 凸轮为 DI 触发启动时，输入参数 StartPosition 必须为正数， StartPosition 的值需小于运行控制器 I 区的容量。
- 凸轮启动时的参考启动点、运行时用于计算的动态坐标点、撤消时的参考坐标点均以凸轮主轴的 Mpos 值为参考基准。当主轴为位置闭环控制时，由于主轴 Mpos 来自于外部检测装置，为避免在凸轮有效行程范围的边界处由于 Mpos 的小扰动所造成的凸轮意外撤消，撤消条件也会兼顾主轴 Dpos 值。
- 对于单次凸轮，当 Mpos 来自外部检测装置时，为避免小扰动所引起的凸轮异常撤消， 应确保凸轮正

常工作区间距离凸轮边界位置有一定余量（保证 Mpos 的扰动不会超出凸轮边界位置）。

- 在凸轮动作中从轴超过软限位时，凸轮被中断，从轴立即停止运行。
- 对于单次凸轮，在行程边界内，从轴跟随主轴运动，主轴负向运动跨越左边界或正向跨越右边界后，凸轮指令被撤销。

6.2 MC_GearIn（电子齿轮）

本指令实现电子齿轮功能。



6.2.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Master	主轴名	否	-	-	AXIS_REF
	Slave	从轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	RatioNumerator	齿轮比分子	是	1	正值/负值/0	DINT
	RatioDenominator	齿轮比分母	是	1	正值	UDINT
	MasterValueSource	主轴位置源 0: 与主轴指令位置同步 1: 与主轴反馈位置同步	是	0	0: mcSetValue 1: mcActualValue	MC_SOURCE
	Acceleration	加速度	否	-	正值	LREAL
	Deceleration	减速度	否	-	正值	LREAL
	BufferMode	缓冲模式 0：打断	是	0	0: Aborting 1: Buffered	MC_BUFFER_MODE

		1：缓存				
OUTPUT	InGear	同步中	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为 TRUE	是	FALSE	TRUE/FALSE	BOOL
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

6.2.2 功能

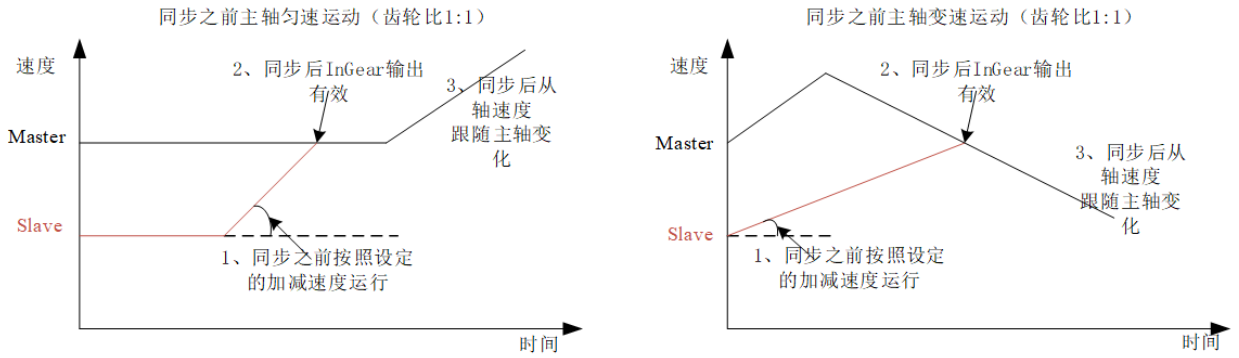
本指令可设定主轴和从轴之间的齿轮比，进行电子齿轮动作。指定从轴为运动轴，可以按照设置的齿轮比分子，齿轮比分母，加速度，减速度等参数进行电子齿轮动作，对该指令的详细介绍如下：

（1） 本指令上升沿触发有效，设定对象为从轴，根据主轴的设定位置和反馈位置进行电子齿轮动作。

（2） 开始齿轮动作之后，从轴的目标速度是主轴的速度乘以齿轮比，从轴的移动距离是主轴的移动距离乘以齿轮比。

（3） 齿轮比为正数时，从轴沿主轴同方向运动，齿轮比为负数时，从轴沿主轴反向运动。

（4） 到达目标位置之前称为追赶中，到达后称为同步中，在达到同步之前，从轴按照设定的加减速度运动，达到同步之后，从轴完全跟随主轴变化，示意图如下所示。

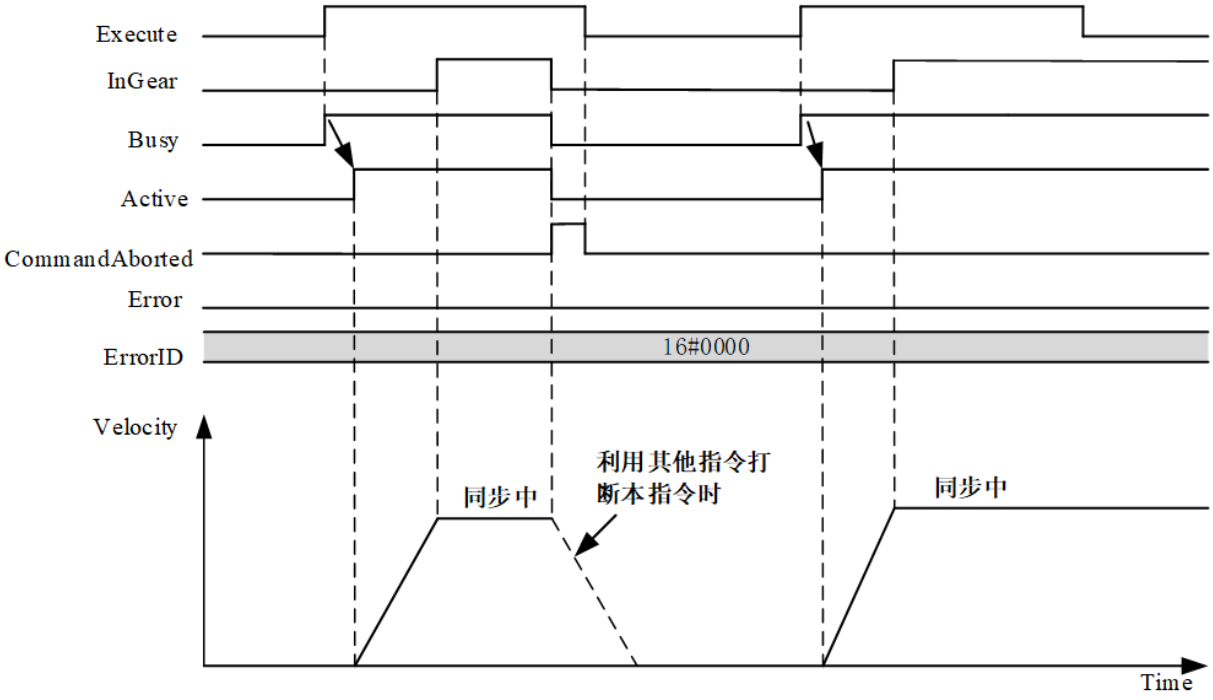


（5） 重启和多重启动时的本指令的动作由 BufferMode（缓冲模式选择）指定，本指令的多重启动可以选择打断和缓存，选择中断时从轴将根据齿轮比和主轴速度重新计算跟随动作，并根据计算结果确定 InGear 标志位是否置位。

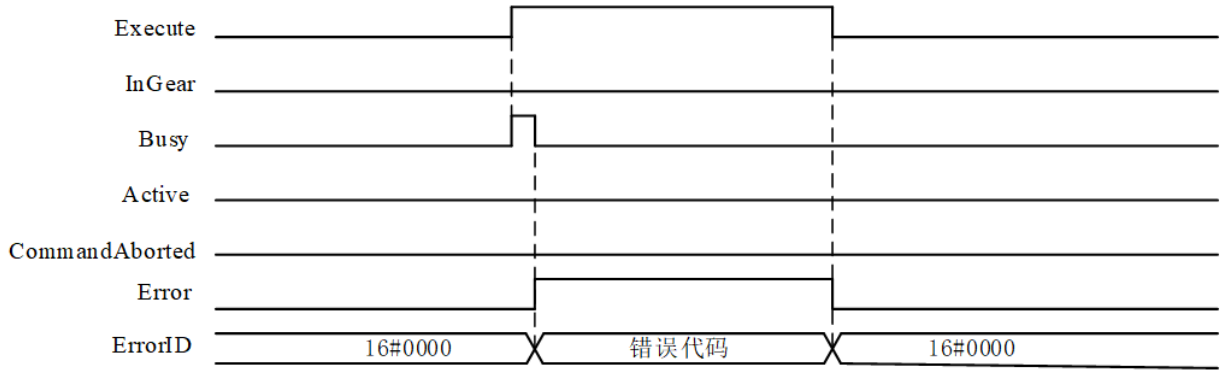
（6） 想要中途结束电子齿轮动作时，可以使用其他指令进行打断或使用 MC_Stop 指令。

■ 时序图

（1） 正常情况时序图



（2） 异常时序图



6.2.3 示例

组建 MC_GearIn 示例程序进行电子齿轮动作指令运行。

■ 变量

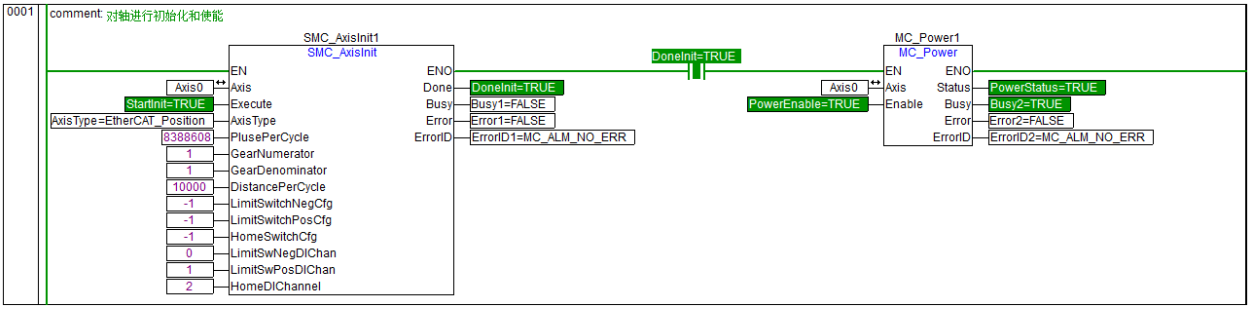
名称	数据类型	初始值	注释说明

Axis0	AXIS_REF		轴 0 的轴变量
StartInit	BOOL	FALSE	开始初始化时变为 TRUE
DoneInit	BOOL	FALSE	完成初始化时变为 TRUE
PowerEnable	BOOL	FALSE	开始使能时变为 TRUE
PowerStatus	BOOL	FALSE	使能成功时变为 TRUE
StartGear	BOOL	FALSE	启动指令时为 TRUE
RatioNumerator	DINT	1	齿轮比分子
RatioDenominator	UDINT	1	齿轮比分母
MasterValueSource	MC_SOURCE	0	主轴位置源选择
InGear	BOOL	FALSE	指令同步时变为 TRUE
BusyGear	BOOL	FALSE	指令投送成功时变为 TRUE
ActiveGear	BOOL	FALSE	指令运行中变为 TRUE
ComGear	BOOL	FALSE	指令被打断时变为 TRUE
ErrorGear	BOOL	FALSE	指令出错时变为 TRUE

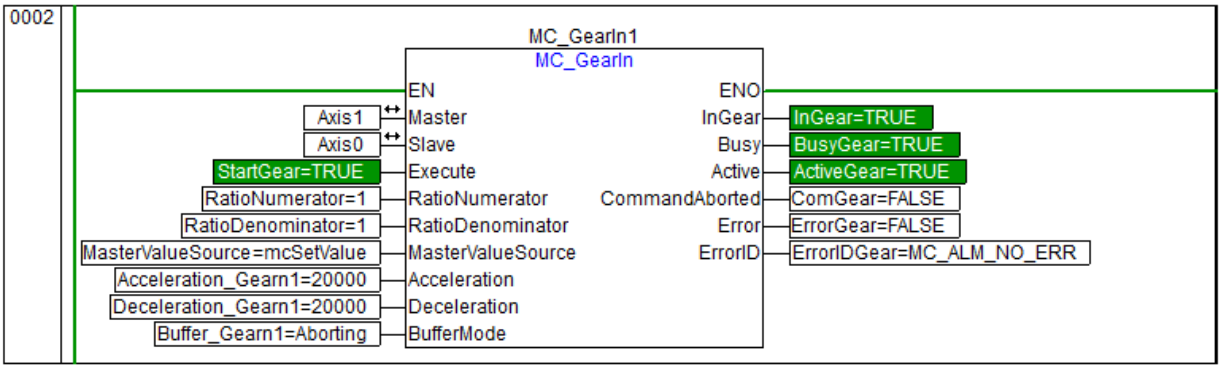
■ 梯形图程序

利用梯形图组建程序，确定主轴为 1 轴，从轴为 0 轴，对主轴投送 MC_MoveVelocity 指令，输入主轴速度为 10000，输入齿轮比为 1:1，位置源选择与主轴指令位置同步，上升沿触发控制从轴进行电子齿轮动作，示例程序如下：

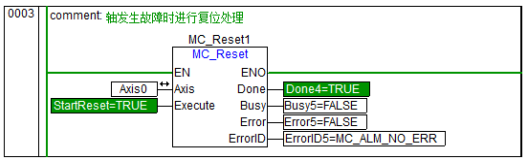
（1） 对从轴进行初始化，对轴的类型和各种参数进行初始化设置，待轴初始化完成之后，对轴进行使能，进入伺服使能的 ON 状态时，轴可以实现运动控制。



（2） 先启动主轴开始速度运动，然后输入从轴相关运动参数和齿轮比，上升沿触发控制电子齿轮进行运动。



（3） 当轴出现错误时，可以调用 MC_Reset 功能块进行复位处理。



■ ST 语言程序

（1） 对轴进行初始化设置

```
SMC_AxisInit1(
Execute := StartInit,
AxisType := 50,
PlusePerCycle := 10000,
GearNumerator := 1,
GearDenominator := 1,
DistancePerCycle := 10000,
LimitSwitchNegCfg := -1,
LimitSwitchPosCfg := -1,
HomeSwitchCfg := -1,
LimitSwNegDIChan := 0,
LimitSwPosDIChan := 1,
HomeDIChannel := 2,
Axis := Axis0,
Done=> DoneInit,
Busy=>Busy1,
Error=>Error1,
ErrorID=>ErrorID1);
```

（2） 对轴进行使能

```
IF DoneInit THEN
   MC_Power1(
   Enable := PowerEnable,
   Axis := Axis0,
   Status=> PowerStatus,
```

```
    Busy=>Busy2,
    Error=>Error2,
    ErrorID=>ErrorID2);
END_IF
```

（3） 投送电子齿轮指令，进行电子齿轮动作

```
MC_GearIn1(
Execute := StartGear,
RatioNumerator := RatioNumerator,
RatioDenominator := RatioDenominator,
MasterValueSource := MasterValueSource,
Acceleration := Acceleration_Gearn1,
Deceleration := Deceleration_Gearn1,
BufferMode := Buffer_Gearn1,
Master := Axis1,
Slave := Axis0,
InGear=>InGear,
Busy=>BusyGear,
Active=>ActiveGear,
CommandAborted=>ComGear,
Error=>ErrorGear,
ErrorID=>ErrorIDGear);
```

（4） 发生错误时进行复位

```
MC_Reset1(
Execute := StartReset,
Axis := Axis0,
Done=>Done4,
Busy=>Busy5,
Error=>Error5,
ErrorID=>ErrorID5);
```

6.2.4 使用注意事项

- 电子齿轮动作是对从轴进行生效，电子齿轮中的加速度和减速度是从轴在追赶中的加减速度，若是处于同步中则从轴完全跟随主轴运动，加速度和减速度不生效。
- 主轴和从轴不能设为同一轴，齿轮比分母不能为 0。

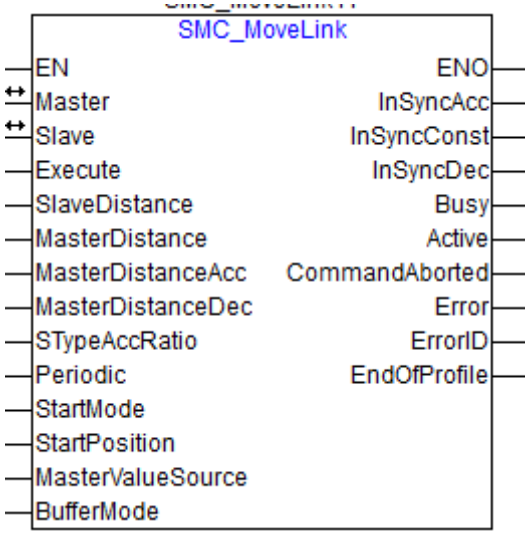
6.2.5 参见

- 对于主轴的使用，参见其他单轴运动章节。

■ 有错误发生时，参见附录功能块错误码。

6.3 SMC_MoveLink（追剪/飞剪）

该指令实现追剪工艺动作。



6.3.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	Master	主轴名	否	-	-	AXIS_REF
	Slave	从轴名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	SlaveDistance	从轴运动距离	否	-	正值/负值/0	LREAL
	MasterDistance	主轴运动距离	否	-	正值	LREAL
	MasterDistanceAcc	从轴加速阶段主轴移动的距离	否	-	正值/0	LREAL
	MasterDistanceDec	从轴减速阶段主轴移动的距离	否	-	正值/0	LREAL
	STypeAccRatio	S 形加减速占比，单位为 1%	是	0	0 ~ 100	LREAL
	Periodic	FALSE: 单次模式 TRUE: 循环模式	是	FALSE	TRUE/FALSE	BOOL
	StartMode	启动模式 0：绝对位置启动（正向经	是	0	0：mcAbsolute 10：	MC_CAMIN_STARTMODE

		过绝对位置点时触发） 10：立即启动 11：DI 触发启动			mcImmediately 11：mcDITrigger	
	StartPosition	绝对位置启动时，为启动位置； DI 触发启动时，为 DI 点在 I 区映射的位偏移。例如%IX100.1 在 I 区的 bit 位偏移为 100×8 + 1。	是	0	绝对位置启动： 正值/负值/0; DI 触发启动： 0 ~ （I 区 byte 容量 * 8）- 1	LREAL
	MasterValueSource	主轴位置源 1: 与主轴反馈位置同步	是	1	1: mcActualValue	MC_SOURCE
	BufferMode	缓冲模式 0：打断 1：缓存	是	1	0: Aborting 1: Buffered	MC_BUFFER_MODE
OUTPUT	InSyncAcc	从轴处于加速同步段	是	FALSE	TRUE/FALSE	BOOL
	InSyncConst	从轴处于匀速同步段	是	FALSE	TRUE/FALSE	BOOL
	InSyncDec	从轴处于减速同步段	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为 TRUE	是	FALSE	TRUE/FALSE	BOOL
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR
	EndOfProfile	周期完成	是	FALSE	TRUE/FALSE	BOOL

6.3.2 功能

本指令实现追剪工艺动作。

■ 指令功能说明

（1） 本指令实现主轴从轴同步追剪动作，属于电子凸轮的一种。

（2） 上升沿触发有效。在 Execute 输入的上升沿，输入参数合法，轴无异常时，输入参数的值被锁存，指令执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令。

（3） 要停止通过本指令处于动作中的轴，请使用 MC_Stop 指令。

（4） 追剪运动行程。追剪运动主轴的一个行程长度为主轴运动距离 MasterDistance。设追剪运动主轴

起点为 Mpos0，则单次追剪的有效行程范围为[Mpos0，Mpos0+ MasterDistance)。

- 单次模式。MoveLink 启动后：对于立即启动和 DI 触发启动，指令左边界在凸轮启动时刻的位置；对于绝对位置启动，指令左边界在设定的绝对位置处；指令右边界 = 左边界+主轴行程长度。主轴负向运动跨越左边界或正向跨越或到达右边界时，指令撤销。
- 对于单次追剪运动，主轴在有效行程范围内时，主轴可进行单向、往复运动，当主轴在有效行程范围内往复运动时，从轴也会进行往复性的联动动作。
- 循环模式。当主轴位置越过一个主轴行程长度时，将会进入下一个追剪行程，此时从轴将会以前一个追剪行程结束时的从轴位置为起点，在此基础上作增量位移计算，重复上一个追剪行程的联动动作，周而复始。如果需要撤消循环追剪运动，可执行 MC_Stop 指令停止和撤销追剪动作。

追剪运动主从轴的联动运动轨迹和凸轮一致，运动轨迹图具体情况参见凸轮指令说明章节的主从轴的联动运动轨迹图（见 SMC_CamIn 章节中的凸轮行程介绍部分）。

（5） 从轴运动距离为 0

从轴运动距离为 0 时，MoveLink 启动后：在从轴加速阶段，InSyncAcc 输出 TRUE；在从轴减速阶段，InSyncDec 输出 TRUE；其余阶段，InSyncConst 输出 TRUE。

（6） 追剪指令从轴支持以下轴类型：

- **Virtual（0）**：虚轴
- **EtherCAT_Position（50）**：EtherCAT 总线连接轴，位置控制
- **EtherCAT_Speed（51）**：EtherCAT 总线连接轴，速度控制
- **EtherCAT_Torque（52）**：EtherCAT 总线连接轴，转矩控制

■ 追剪启动方式

在 StandStill 状态、位置控制中、速度控制中、同步控制中的任一状态下都可以启动本指令。

追剪运动启动时，根据不同的启动方式设定主轴起点位置。根据启动模式 StartMode 设定值，追剪启动模式分为 0：固定位置启动、 10：立即启动、11：DI 启动。

■ 固对位置启动

指令启动模式为 0：固对位置启动。启动指令后，等待追剪指令主轴 Mpos 从负向跨过凸轮启动位置 StartPosition 处，指令触发，追剪运动开始。

- 立即启动

指令启动模式为 10：立即启动。启动指令后，立即开始执行追剪动作。此时 StartPosition 无意义。

- DI 触发启动

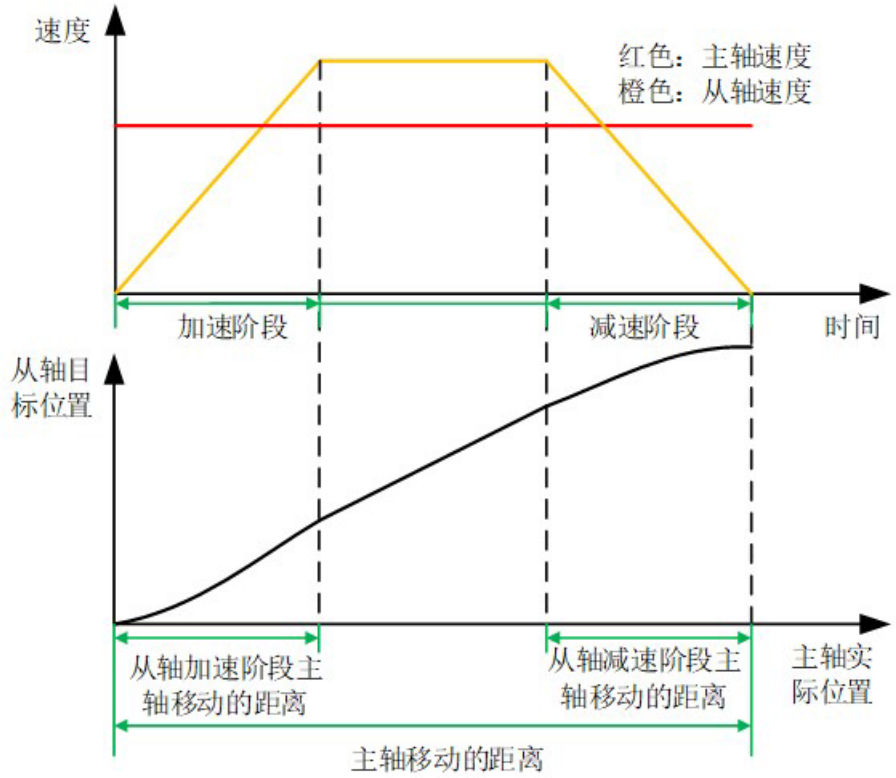
指令启动模式为 11：DI 触发启动。追剪指令投放后，当设定的某 DI 信号为 1 时启动指令，启动位置为程序检测到 DI 为 1 时主轴 Mpos 位置。StartPosition 为 DI 点在 I 区映射的位偏移。例如%IX100.1 在 I 区的 bit 位偏移为 100×8 + 1，即 StartPosition=801。DI 触发启动时，启动位置 StartPosition 的值需小于运行控制器 I 区的容量。

- 加减速阶段规划方式

加减速阶段规划方式。指令支持加减速阶段按梯形或 S 形曲线加减速两种方式。根据设定好的加减速阶段距离、加减速方式等，可选择梯形、S 形加减速。

- 梯形加减速(STypeAccRatio=0)

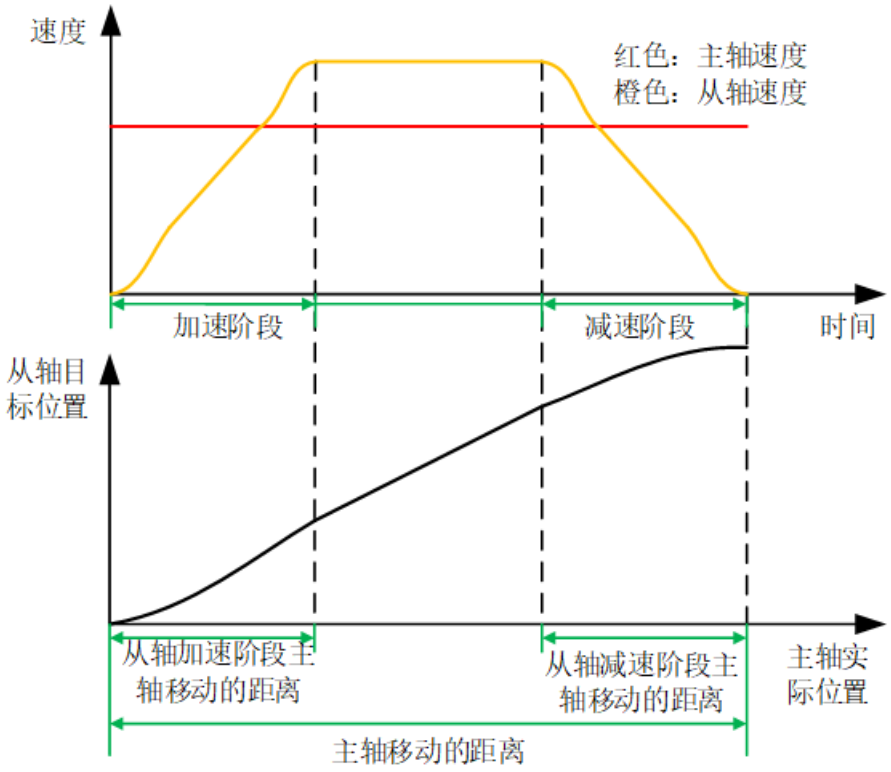
当 STypeAccRatio= 0 时，从轴的速度按照梯形曲线加减速规划，根据梯形模型、以及主轴实际位置实时计算本周期从轴的目标位置。梯形加减速时，以主轴匀速运动为例，主、从轴的速度、位移速度曲线示意图如下：



■ S 型曲线加减速 (STypeAccRatio 在(0,100]范围)

当 STypeAccRatio 在(0,100]范围时，从轴的速度按照 S 形曲线加减速规划，根据 S 型模型、主轴实际位置实时计算本周期从轴的目标位置。STypeAccRatio 表示 S 形加减速在整个加速/减速过程所占比例，以加速阶段为例，加加速阶段所用时间占整个加速阶段时间的比例为 STypeAccRatio/100/2，匀加速阶段所用时间占整个加速阶段时间的比例为 1- STypeAccRatio/100，减加速阶段所用时间占整个加速阶段时间的比例为 STypeAccRatio/100/2。

以 STypeAccRatio = 50/100，主轴匀速运动为例，主、从轴的速度、位移曲线示意图如下：



当设置 STypeAccRatio= 100 时，加减速阶段就没有匀加速/匀减速阶段，加速阶段的加加速和减加速比例分别为 0.5，减速阶段类似。

■ Periodic（循环模式）

按主轴位置的边界范围划分，追剪运动可分为单次和循环追剪运动。追剪循环模式由输入参数 Periodic 决定，当 Periodic=False 时，为单次追剪，当 Periodic=TRUE 时，为循环追剪。

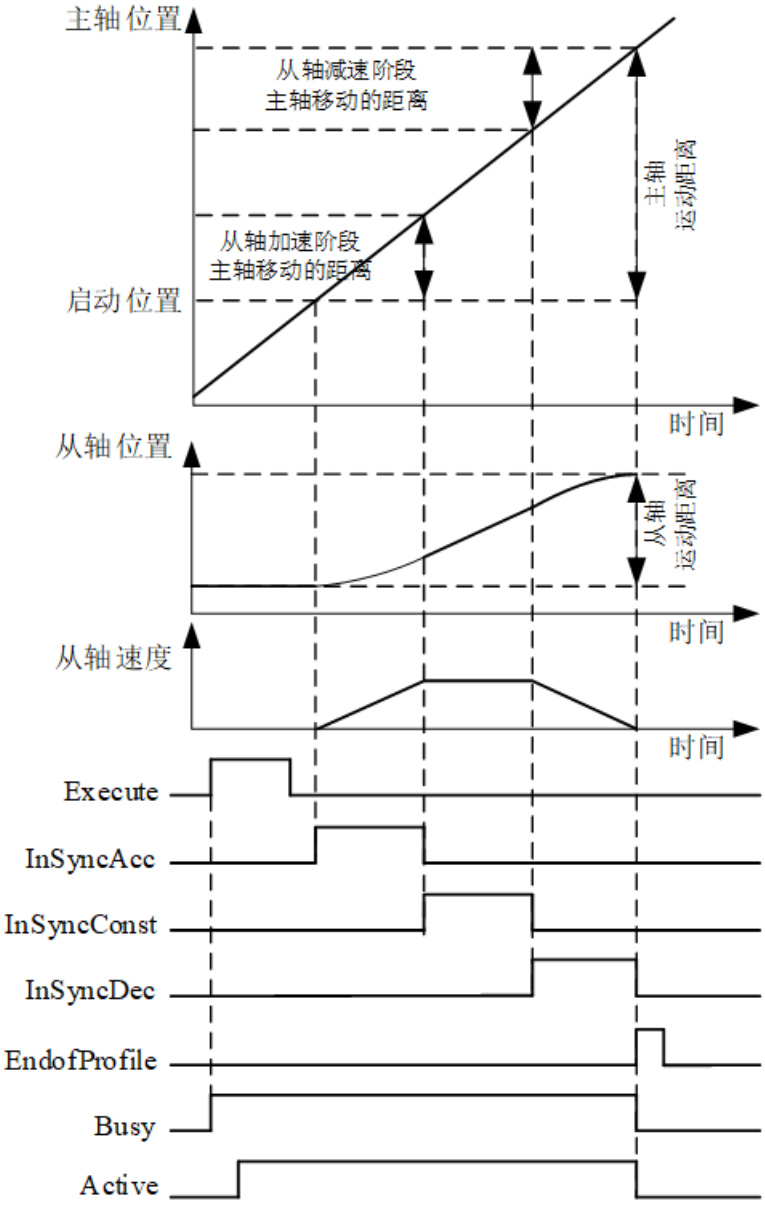
（1） 单次追剪

追剪运动启动后，若主轴位置在有效行程范围内，则追剪运动可保持正常联动运行，主轴超过有效行程范围时指令将自动撤销。

追剪启动后，满足指令触发条件后，指令 InSyncAcc 引脚置 TRUE，从轴跟随主轴同步运动；当从轴完成加速同步段后进入匀速同步段，InSyncConst 引脚置 TRUE，InSyncAcc 置 FALSE，当从轴进入减速同步段时，InSyncDec 引脚置 TRUE，InSyncConst、InSyncAcc 置 FALSE。

单次追剪指令运行完成后，EndofProfile 引脚置 TRUE 且至少保持一个周期，Busy 和 Active 引脚变为 FALSE，遇到 Execute 下降沿后，所有输出引脚恢复为默认值。

下图为单次模式，固定位置启动时的引脚时序：

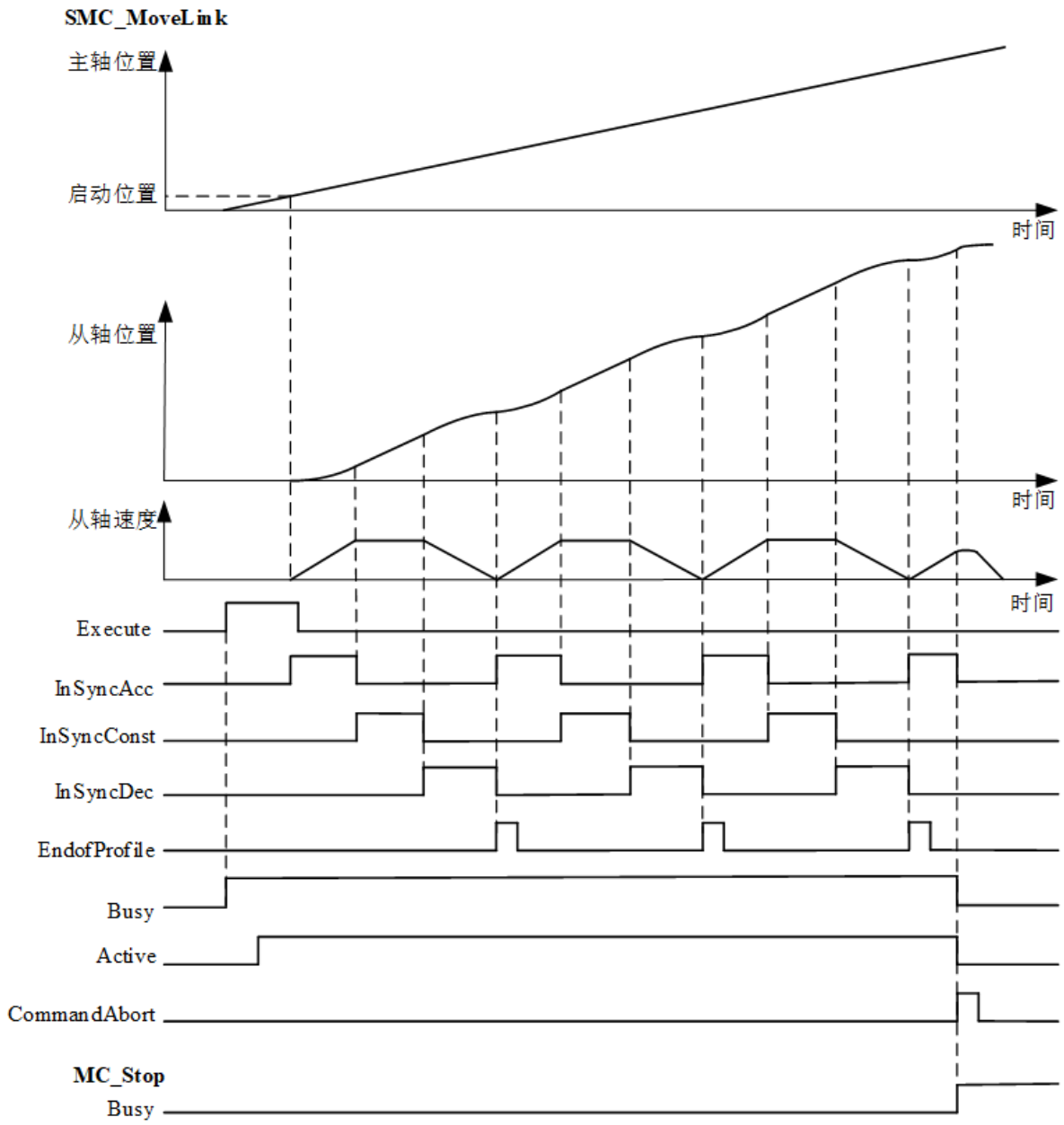


（2） 循环追剪

循环追剪运动，主轴位置没有边界范围限制，启动后若不执行相应的停止撤销指令，追剪功能将一直有效。

下图为循环模式，固定位置启动模式下的追剪指令引脚时序。从轴跟随主轴作相应的联动动作，当主轴一个追剪行程结束时，进入下一个追剪行程，从轴以上一个追剪行程的终点为起点作增量计算，执行周期性的联动动作。每一个追剪行程结束时，输出引脚 EndofProfile 置一个周期的高电平。

利用 MC_Stop 指令撤销停止循环追剪指令时，追剪指令输出引脚 CommandAborted 置 TRUE，其余输出引脚复位。



■ BufferMode （缓存模式选择）

指定前一个轴动作和本次动作的连接方式。

本指令 BufferMode 仅支持 0：Aborting 和 1：Buffered。

投送本指令时，若 BufferMode 为 0：Aborting，立即中止当前正在执行的指令，切换为本指令。若 BufferMode 为 1：Buffered，从当前执行中的指令正常结束的周期起，已缓存的本指令自动启动。

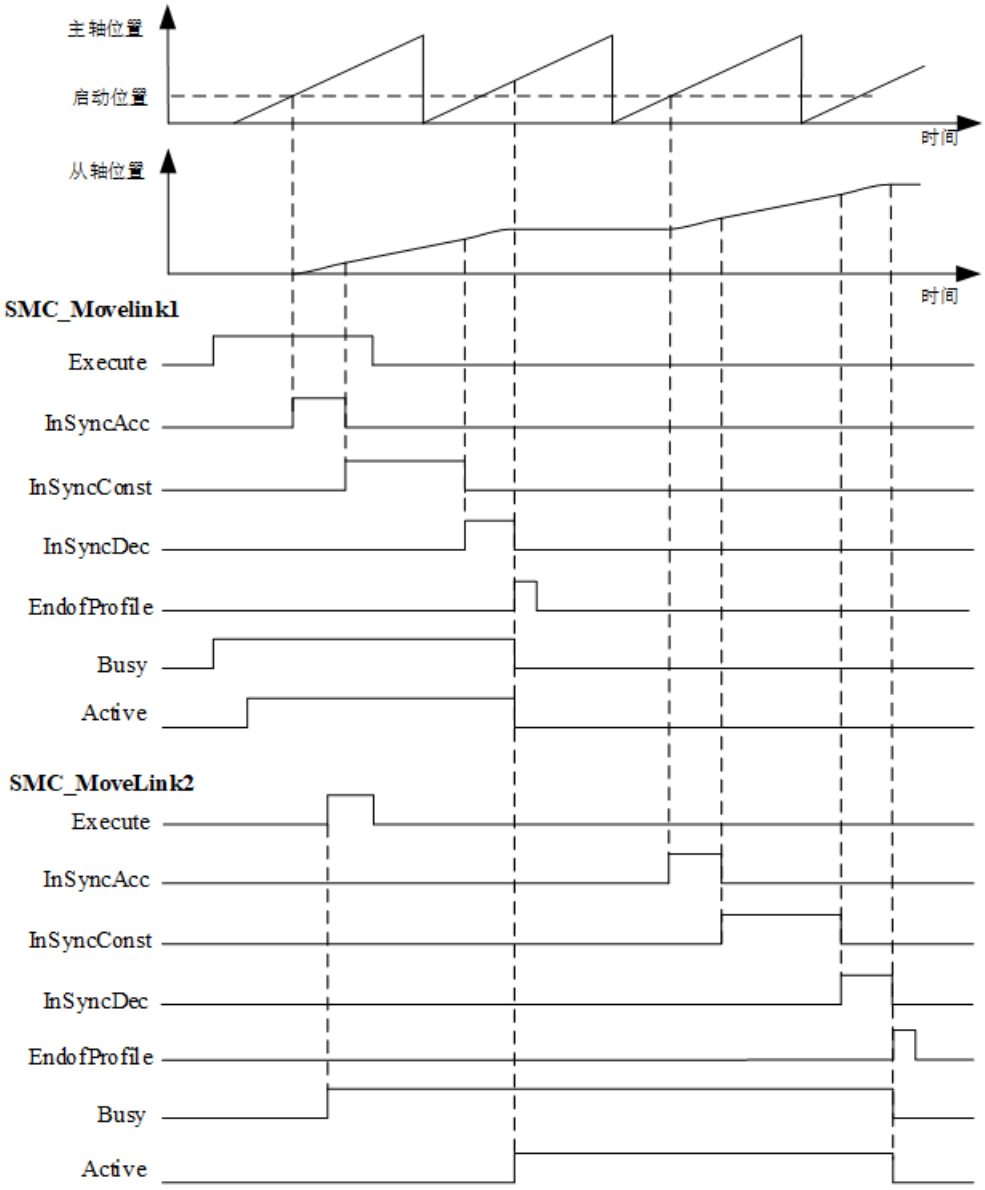
- 功能块重复触发的处理
  - 对于运行中的同一追剪运动指令功能块，如果重复触发执行，则有：

    (a) **若指令的 BufferMode 为 0：**Aborting，则打断当前正在执行的指令；

    (b) **若指令的 BufferMode 为 1：**Buffered，则指令被放入运动缓存中，之后功能块监控的状态为后一条指令的运行状态，前一条指令的运行状态则失去监控。

  - 功能块指令等待重复触发时，在前一个追剪指令执行期间，后一个追剪指令等待前一个追剪指令结束后，开始执行。

如下图示例，在执行 SMC_MoveLink1（单次模式，固定位置启动）指令期间，指定等待后重复启动 SMC_MoveLink2（单次模式，固定位置启动）指令。在 SMC_MoveLink1 的 EndOfProfile 变为 TRUE 以及 SMC_MoveLink2 的 Active 变为 TRUE 后，待主轴到达 SMC_MoveLink2 指令的 StartPosition（启动位置）位置处，InSyncAcc(加速同步中)变为 TRUE，开始追剪动作。



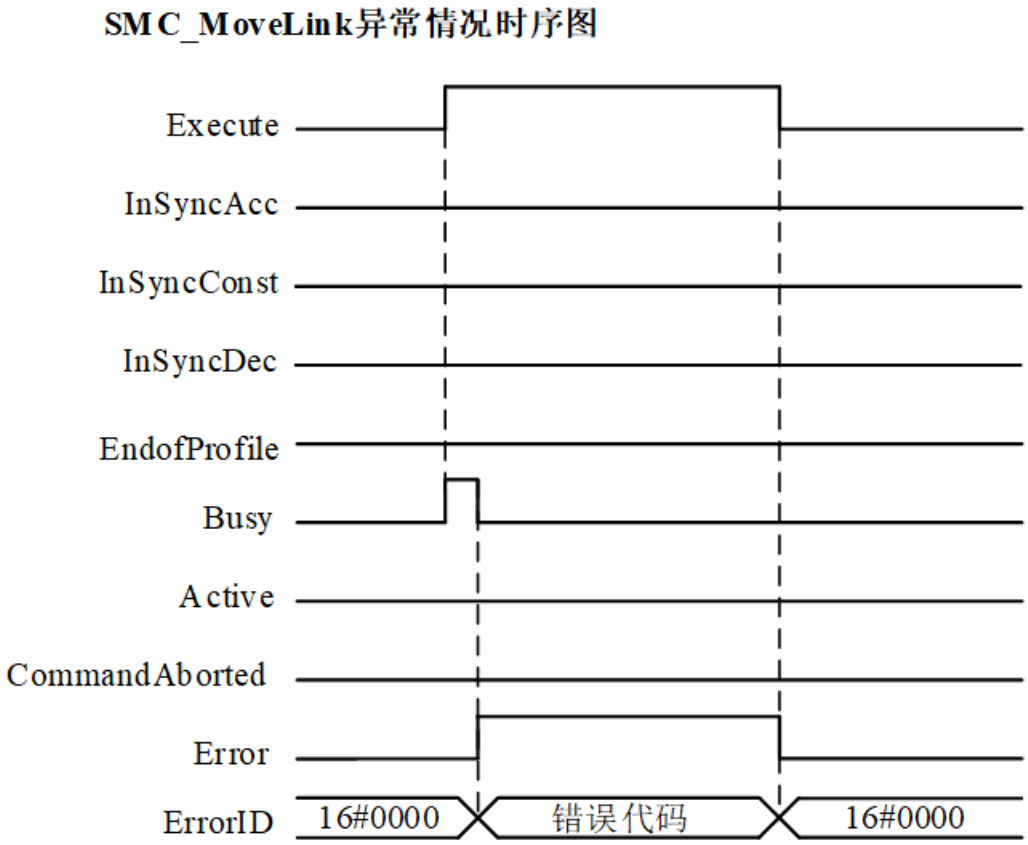
■ EndofProfile（追剪周期完成）

追剪指令为单次模式时，指令周期完成，指令输出引脚 EndofProfile 置 TRUE 保持，直到输入引脚 Execute 变为 FALSE 时，引脚 EndofProfile 变为 FALSE。

追剪指令为周期模式时，第一个追剪行程完成，指令输出引脚 EndofProfile 置 TRUE 仅一个周期，指令进入下一个追剪行程，EndofProfile 变为 FALSE，直到该行程完成，引脚 EndofProfile 置 TRUE 一个周期，EndofProfile 随着周期性追剪动作持续变化更新。

■ 异常

启动追剪运动指令时，若发生异常错误，则 Error 引脚置为 True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Execute 变为低电平时，所以输出引脚恢复为默认值。



6.3.3 示例

本示例中，定义 MasterAxis 为主轴，SlaveAxis 为从轴，变量类型为 AXIS_REF。

主轴执行 MC_MoveVelocity 指令、从轴执行 MC_MoveLink 指令。使用 MC_Stop 指令停止撤销追剪运动指令。

■ 梯形图程序

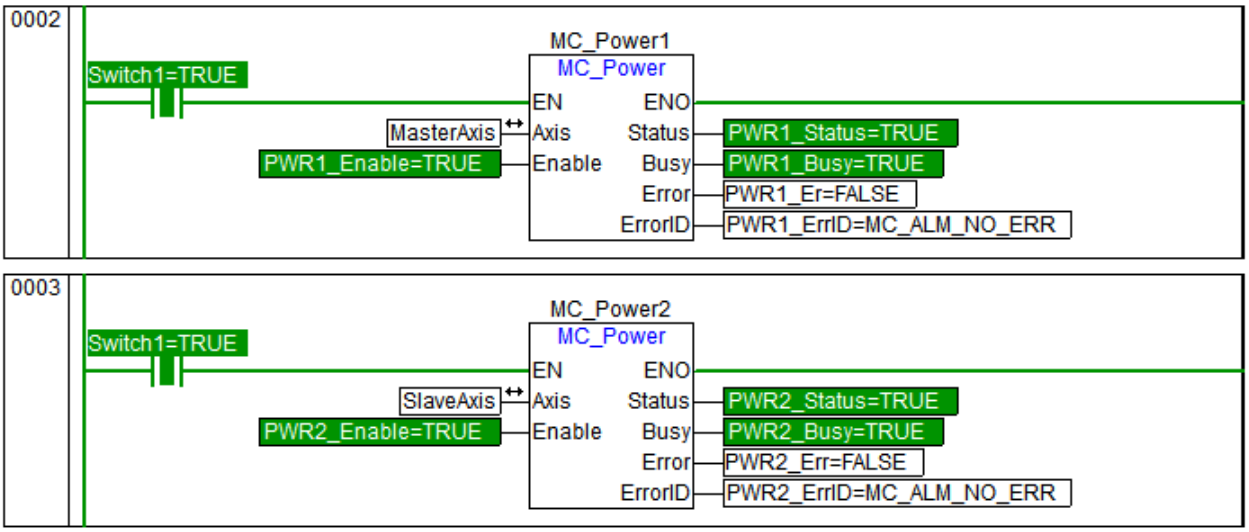
本示例组建周期模式，立即启动的追剪指令。

示例程序主要变量

变量名称	变量类型	初始值	注释
MasterAxis	AXIS_REF	-	主轴的轴变量
SlaveAxis	AXIS_REF	-	从轴的轴变量
Power1_Status	BOOL	FALSE	主轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	从轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
Switch1	BOOL	FALSE	轴准备就绪的状态变量， Switch 为 TRUE，轴准备就绪。
MV_Exe	BOOL	FALSE	轴的速度控制指令的上升沿触发变量。
MV_InVel	BOOL	FALSE	轴速度控制指令达到目标速度的标识变量。该变量变为 TRUE，速度指令的速度达到目标速度。
ML_SlaveDis	LREAL	-	从轴投送追剪指令的从轴运动距离。
ML_MasterDis	LREAL	-	从轴投送追剪指令的主轴运动距离。
ML_DisAcc	LREAL	-	从轴投送追剪指令的从轴加速段主轴运动距离。
ML_DisDec	LREAL	-	从轴投送追剪指令的从轴减速段主轴运动距离。

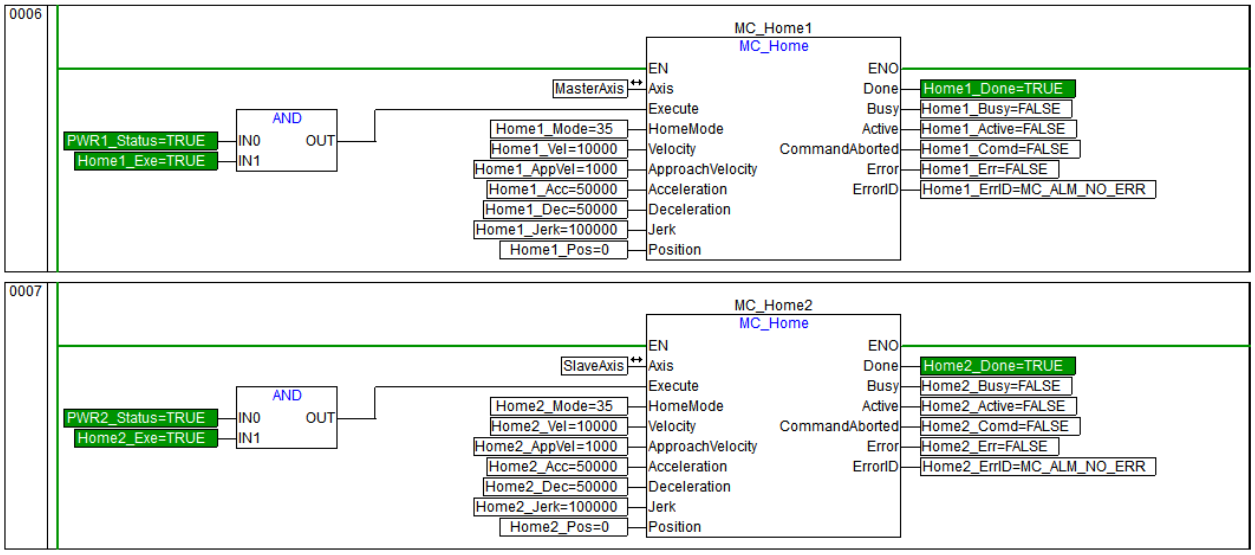
（1） 轴初始化。设置主轴 MasterAxis 的 AxisID 为 0，从轴 SlaveAxis 的 AxisID 为 1，主、从轴的轴类型均为 50，调用 SMC_AxisInit 指令完成主、从轴的初始化配置。SMC_AxisInit 指令使用方法参见其介绍章节。

（2） 轴使能。0002 节中，主、从轴初始化完成，轴准备就绪状态 Switch1 为 TRUE 后，执行轴使能指令，完成轴使能。

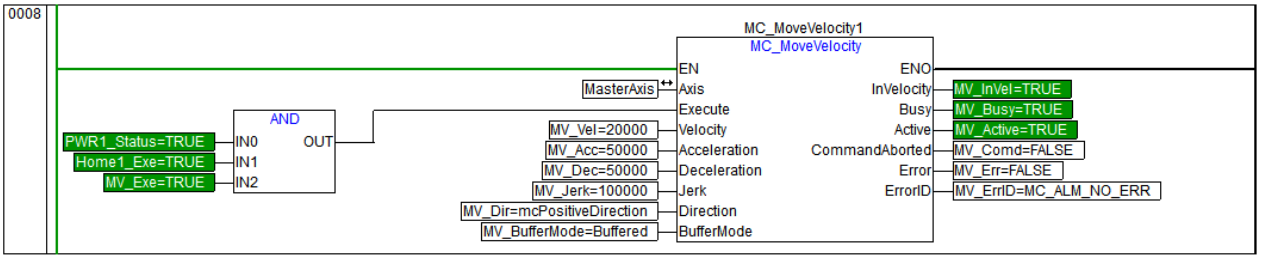


（3） 主、从轴使能成功，若原点未确定，则执行主从轴的原点回归指令，确定原点，原点回零见 0006

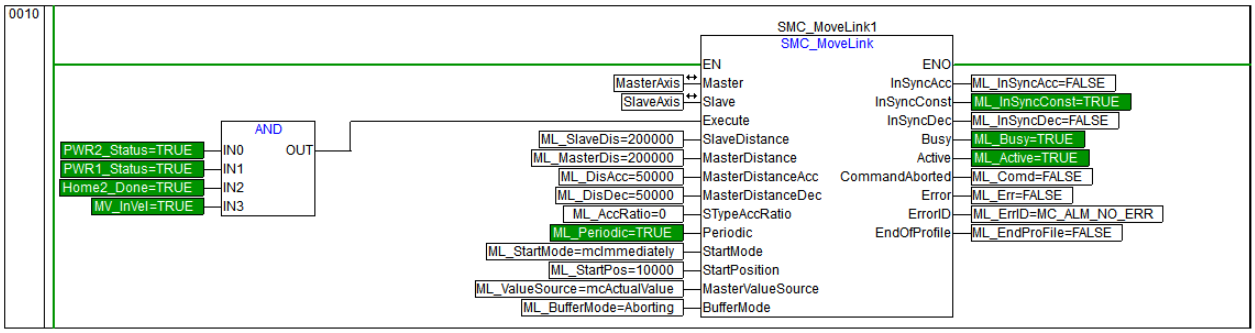
和 0007 节，MC_Home 指令使用方法见其介绍章节。



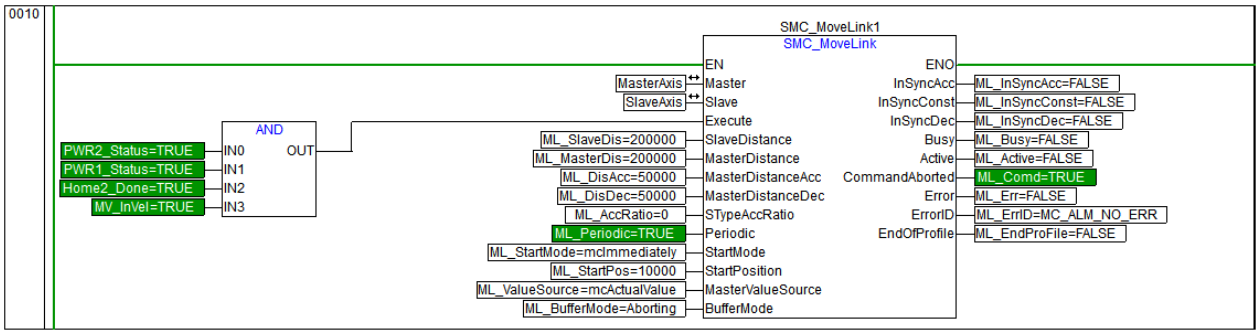
（4） 见 0008 节，原点回归完成后，执行 MC_MoveVelocity(速度控制)指令。主轴开始运动。

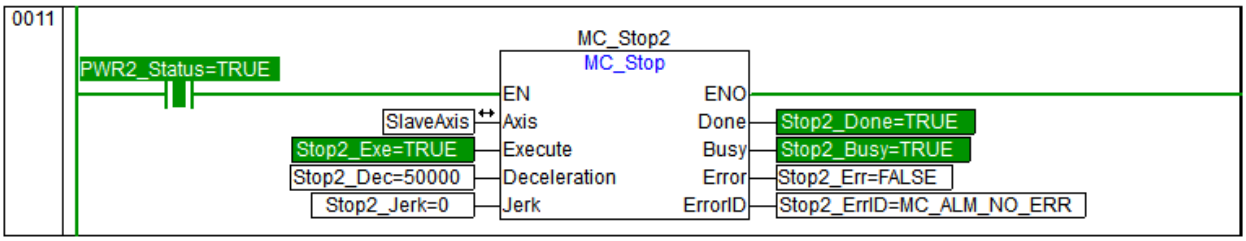


（5） 见 0009 节，当主轴速度指令 MC_MoveVelocity 的 MV_InVel 引脚变为 TRUE 后，启动从轴的 SMC_MoveLink1 指令。指令为循环立即启动，ML_InSyncAcc 引脚为 TRUE，从轴进入追剪运动加速同步段（ML_InSyncAcc=TRUE），从轴跟随主轴开始运动，然后依次进入追剪运动匀速同步段（ML_InSyncConst =TRUE）、追剪运动减速同步段（ML_InSyncDec =TRUE）。

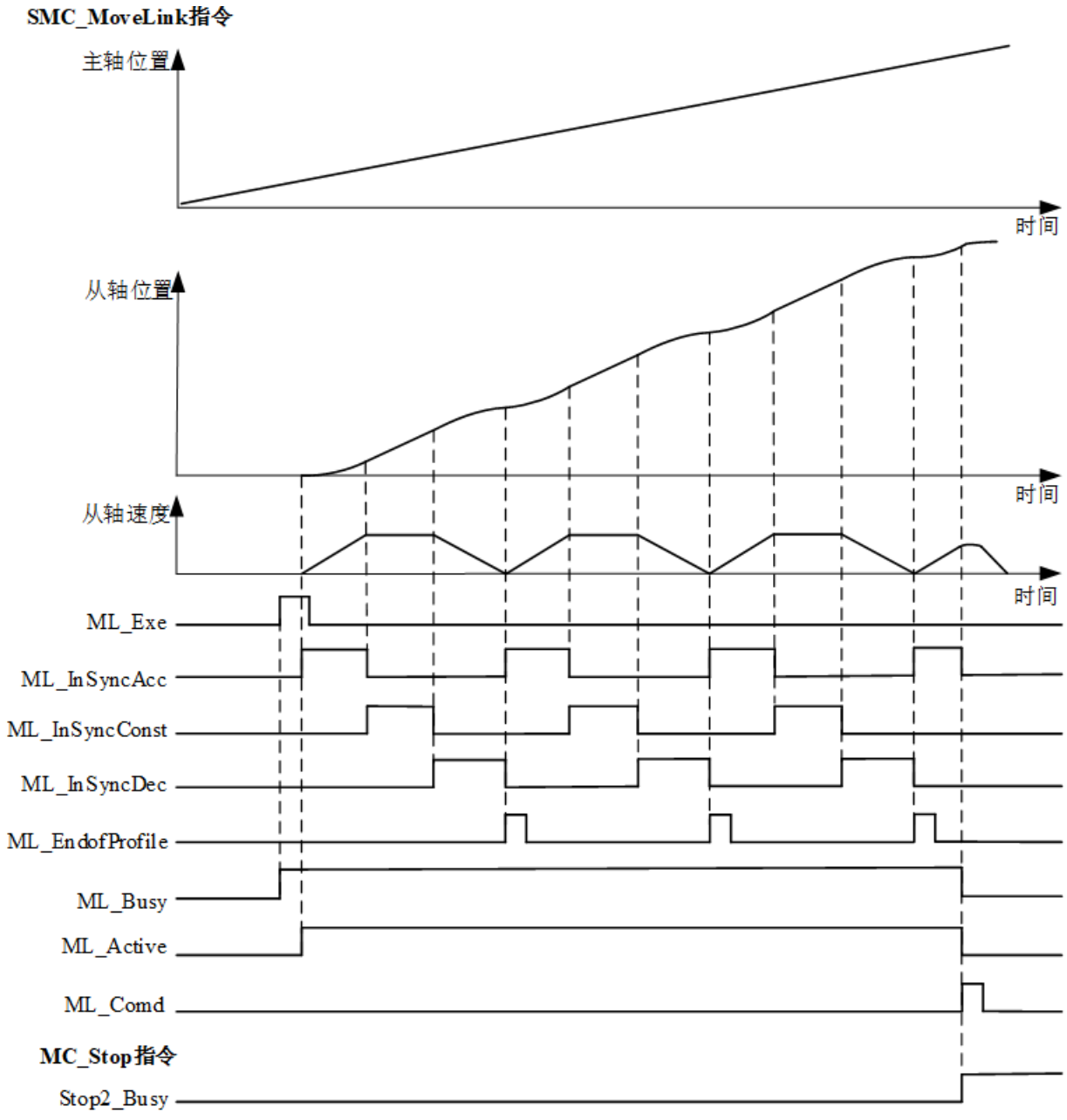


（6） 对于循环追剪指令，可以使用 MC_Stop 撤销并停止从轴的追剪运动指令。如下 0011 节中，执行 MC_Stop2 指令，从轴的追剪指令被中断，追剪指令的 ML_Comd 变为 TRUE。


0010
SMC_MoveLink1
SMC_MoveLink
EN
ENO
MasterAxis
Master
InSyncAcc
ML_InSyncAcc=FALSE
SlaveAxis
Slave
InSyncConst
ML_InSyncConst=FALSE
Execute
InSyncDec
ML_InSyncDec=FALSE
AND
PWR2_Status=TRUE
IN0
OUT
ML_SlaveDis=200000
SlaveDistance
Busy
ML_Busy=FALSE
PWR1_Status=TRUE
IN1
ML_MasterDis=200000
MasterDistance
Active
ML_Active=FALSE
Home2_Done=TRUE
IN2
ML_DisAcc=50000
MasterDistanceAcc
CommandAborted
ML_Comd=TRUE
MV_InVel=TRUE
IN3
ML_DisDec=50000
MasterDistanceDec
Error
ML_Err=FALSE
ML_AccRatio=0
STypeAccRatio
ErrorID
ML_ErrID=MC_ALM_NO_ERR
ML_Periodic=TRUE
Periodic
EndOfProfile
ML_EndProFile=FALSE
ML_StartMode=mcImmediately
StartMode
ML_StartPos=10000
StartPosition
ML_ValueSource=mcActualValue
MasterValueSource
ML_BufferMode=Aborting
BufferMode


0011
MC_Stop2
MC_Stop
PWR2_Status=TRUE
EN
ENO
SlaveAxis
Axis
Done
Stop2_Done=TRUE
Stop2_Exe=TRUE
Execute
Busy
Stop2_Busy=TRUE
Stop2_Dec=50000
Deceleration
Error
Stop2_Err=FALSE
Stop2_Jerk=0
Jerk
ErrorID
Stop2_ErrID=MC_ALM_NO_ERR

本示例的时序图如下：



■ ST 语言程序。

指令设为单次模式，绝对位置启动，凸轮启动固定位置为 100。主轴位置从负向跨过凸轮启动位置 100 处，开始凸轮动作。

主要变量

变量名称	变量类型	初始值	注释
MasterAxis	AXIS_REF	-	主轴的轴变量
SlaveAxis	AXIS_REF	-	从轴的轴变量

Power1_Status	BOOL	FALSE	主轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	从轴使能成功的变量。轴使能成功时，该变量变为 TRUE。
MV_Exe	BOOL	FALSE	轴的速度控制指令的上升沿触发变量。
Switch1	BOOL	FALSE	轴准备就绪的状态变量， Switch 为 TRUE，轴准备就绪。
InitFlag	BOOL	FALSE	如果该变量为 TRUE，则初始化参数设置完成。
ML_SlaveDis	LREAL	-	从轴投送追剪指令的从轴运动距离。
ML_MasterDis	LREAL	-	从轴投送追剪指令的主轴运动距离。
ML_DisAcc	LREAL	-	从轴投送追剪指令的从轴加速段主轴运动距离。
ML_DisDec	LREAL	-	从轴投送追剪指令的从轴减速段主轴运动距离。

（1） 轴初始化

设定主轴 MasterAxis 的 AxisID 为 0，从轴 SlaveAxis 的 AxisID 为 1。调用 SMC_AxisInit 指令完成主从轴的初始化配置，主、从轴类型均为 50，调用 SMC_AxisInit 指令完成主、从轴的初始化配置。SMC_AxisInit 指令使用方法参见其介绍章节。

（2） 追剪指令参数初始化

```
IF FALSE = InitFlag THEN
  ML_SlaveDis:=200;
  ML_MasterDis:=200;
  ML_DisAcc:=10;
  ML_ DisDec:=10;
  ML_Periodic:=FALSE;
  ML_STartMode:= mcAbsolute;
  ML_StartPosition:=100;
  ML_Buffer:=Buffered;
  InitFlag:=TRUE;
END_IF
```

（3） 轴使能

主、从轴初始化完成，轴准备就绪状态 Switch1 为 TRUE 后，执行轴使能指令，完成轴使能。

```
IF  Switch1 THEN
    MC_Power1(
    Enable := PWR1_Enable,
    Axis := MasterAxis,
    Status=> PWR1_Status,
    Busy=> PWR1_Busy,
Error=> PWR1_Err
ErrorID=> PWR1_ErrID)
MC_Power2(
    Enable := PWR2_Enable,
```

```
  Axis := SlaveAxis,
  Status=> PWR2_Status,
  Busy=> PWR2_Busy,
  Error=> PWR2_Err
  ErrorID=> PWR2_ErrID);
END_IF
```

（4） 执行主从轴的原点回归指令，确定原点，MC_Home 指令使用方法见其介绍章节。

```
IF  PWR1_Status THEN
 MC_Home1(
 Execute := Home1_Exe,
 HomeMode := Home1_Mode,
 Velocity := Home1_Vel,
 ApproachVelocity := Home1_AppVel,
 Acceleration := Home1_Acc,
 Deceleration := Home1_Dec,
 Jerk := Home1_Jerk,
 Position := Home1_Pos,
 Axis := MasterAxis,
 Done=> Home1_Done,
 Busy=> Home1_Busy,
 Active=> Home1_Active,
 CommandAborted=> Home1_Comd,
 Error=> Home1_Err,
 ErrorID=> Home1_ErrID);
 MC_Home2(
 Execute := Home2_Exe,
 HomeMode := Home2_Mode,
 Velocity := Home2_Vel,
 ApproachVelocity := Home2_AppVel,
 Acceleration := Home2_Acc,
 Deceleration := Home2_Dec,
 Jerk := Home2_Jerk,
 Position := Home2_Pos,
 Axis := SlaveAxis,
 Done=> Home2_Done,
 Busy=> Home2_Busy,
 Active=> Home2_Active,
 CommandAborted=> Home2_Comd,
 Error=> Home2_Err,
 ErrorID=> Home2_ErrID);
END_IF
```

（5） 从轴投送追剪指令，单次固定位置启动，主轴从固定位置的负向到正向时，追剪动作启动。

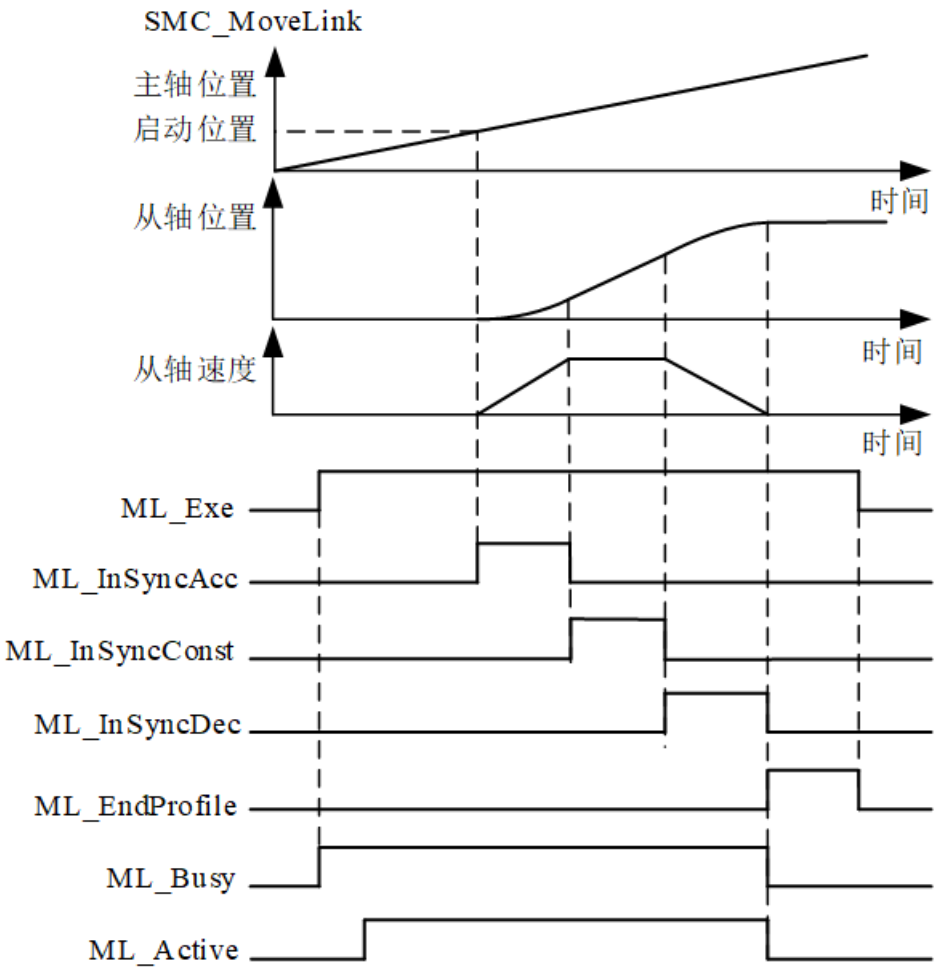
```
IF  PWR1_Status AND PWR2_Status AND Home1_Done THEN
```

```
 SMC_MoveLink1(
 Execute := ML_Exe,
 SlaveDistance := ML_SlaveDis,
 MasterDistance := ML_MasterDis,
 MasterDistanceAcc := ML_DisAcc,
 MasterDistanceDec := ML_ DisDec,
 STypeAccRatio := ML_AccRatio,
 Periodic := ML_Periodic,
 StartMode := ML_STartMode,
 StartPosition := ML_StartPosition,
 MasterValueSource := ML_ValueSource,
 BufferMode := ML_Buffer,
 Master := MasterAxis,
 Slave := SlaveAxis,
 InSyncAcc=> ML_InSyncAcc,
 InSyncConst=> ML_InSyncConst,
 InSyncDec=> ML_InSyncDec,
 Busy=> ML_Busy,
 Active=> ML_Act,
 CommandAborted=> ML_Comd,
 Error=> ML_Err,
 ErrorID=> ML_ErrID,
 EndOfProfile=> ML_EndPro);
END_IF
```

（6） 投送主轴运动指令

```
IF  PWR1_Status AND Home1_Done THEN
 MC_MoveVelocity1(
 Execute := MV_Exe,
 Velocity := MV_Vel,
 Acceleration := MV_Acc,
 Deceleration := MV_Dcc,
 Jerk := MV_Jerk,
 Direction := MV_Dir,
 BufferMode := MV_Buffer,
 Axis := Axis0,
 InVelocity=> MV_InVel,
 Busy=> MV_Busy,
 Active=> MV_Active,
 CommandAborted=> MV_Comm,
 Error=> MV_Err,
 ErrorID=> MV_Error);
END_IF
```

（7） 示例的主从轴位置曲线和指令引脚时序如下图所示：



6.3.4 使用注意事项

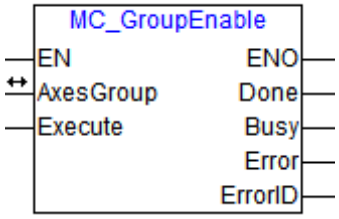
- 主轴的运动距离需为正值。
- 对于单次追剪运动，在行程边界内，从轴跟随主轴运动，主轴负向运动跨越左边界或正向跨越右边界后，指令被撤销。
- 追剪为绝对位置启动时，主轴正向经过绝对位置处触发凸轮指令。
- 追剪为 DI 触发启动时，输入参数 StartPosition 必须为正数。StartPosition 的值需小于运行控制器 I 区的容量
- SlaveDistance 为 0 时，MoveLink 启动后：在 MasterDistanceAcc 阶段，InSyncAcc 输出 TRUE；在 MasterDistanceDec 阶段，InSyncDec 输出 TRUE；其余阶段，InSyncConst 输出 TRUE。

- 主轴和从轴设定为同一轴时，指令报出异常错误。
- 在追剪动作中从轴超过软限位时，指令被中断，从轴立即停止运行。
- 主轴的计数器模式为旋转模式时，请将 MasterDistance(主轴移动距离)指定为主轴的环计数器 1 周以内的数值。

第7章　轴组运动指令

7.1　MC_GroupEnable（轴组使能）

轴组使能。



7.1.1 参数

■　引脚参数

引脚参数说明

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	AxesGroup	轴组名	否	-	-	AXES_GROUP_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
OUTPUT	Done	完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

■　轴组结构参数

目前只支持两轴插补。

AXES_GROUP_REF 数据结构

成员	含义	默认值	范围	数据类型
AxisInterp_ID	插补合运动轴 ID	255	0~63	USINT

AxisX_ID	插补 X 轴 ID	255	0~63	USINT
AxisY_ID	插补 Y 轴 ID	255	0~63	UDINT
ReservedPara1	保留参数	255	-	USINT
ReservedPara2	保留参数	255	-	USINT
ReservedPara3	保留参数	255	-	USINT
ReservedPara4	保留参数	255	-	USINT
ReservedPara5	保留参数	255	-	USINT
AxesGroupDisabled	轴组处于 Disabled 状态	TRUE	TRUE/FALSE	BOOL

- 用户需在 AT 工程中手动添加轴组变量，输入合法的合轴 ID、X 轴 ID、Y 轴 ID。

7.1.2 功能

本指令实现轴组使能。

■ 指令功能说明

（1） MC_GroupEnable 指令将通过 AxesGroup（轴组）指定的轴组变为 Enable 状态。

（2） 定义轴组时，需要自定义三个轴，对应与轴组中的插补合轴和两个分轴。自定义的三个轴 ID 与轴组中的三个插补轴 ID 对应。

（3） 轴组变量 AxesGroup 类型为 AXES_GROUP_REF，设定轴组后，调用本指令进行轴组使能，当 AxesGroup.AxesGroupDisabled=FALSE，轴组为使能状态，轴组便可以执行所有多轴协调指令。

（4） Execute 上升沿该指令，轴参数检查正常，MC_GroupEnable 指令将指定的轴组中的 AxesGroupDisabled 变量置为 FALSE，完成轴组使能。AxesGroupDisabled 变量仅通过 MC_GroupEnable 指令和 MC_GroupDisable 指令更改变量值，严禁用户手动更改。

（5） 轴组合轴的轴类型为未使用轴时，无需调用 MC_Power 进行合轴的轴使能，轴组使能指令自行完成合轴的使能，因此，只需完成两个分轴使能即可。

（6） 若设置合轴为虚拟轴时，同分轴一样，需要调用 MC_Power 完成轴使能。

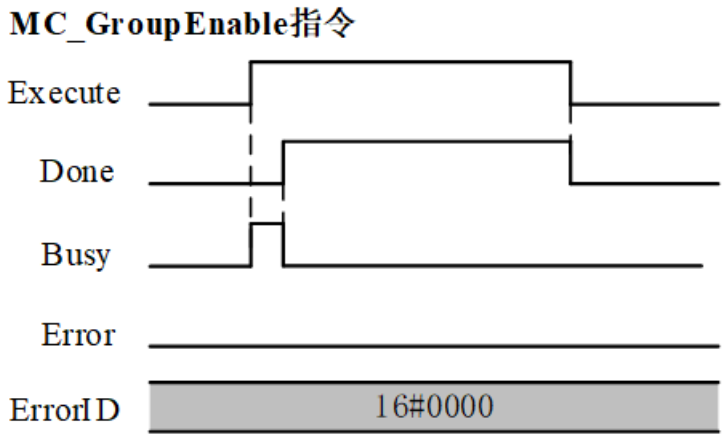
（7） 使轴组无效的条件有执行 MC_GroupDisable（关闭轴组功能）指令，该指令执行成功后，AxesGroupDisabled 变量置为 TRUE，关闭轴组功能。

（8） 支持的轴类型。调用该指令时，指定到轴组中的两个分轴的轴类型为 EtherCat 总线伺服轴和 Virtual：虚拟轴，若指定了其他轴种类，将出现异常。合轴类型为虚拟轴。

■ 时序图

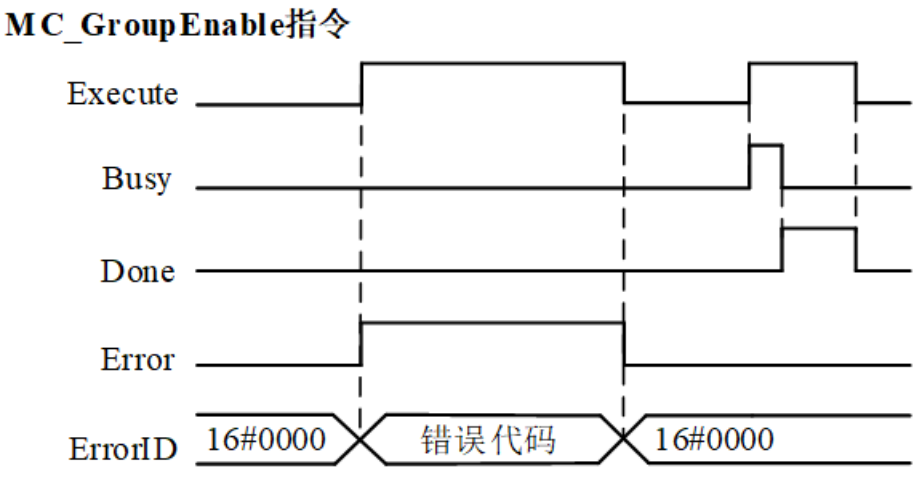
下述为 MC_GroupEnable 指令执行时的正常、异常时序图。

（1） 正常时序



（2） 异常错误发生时引脚时序

执行本指令时，轴组存在轴 ID 非法、轴组中轴类型非法等异常情况，调用本指令失败。此时，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，可以了解错误发生原因。指令输入引脚 Execute 为低电平时，所有输出引脚恢复为默认值。



- 当指令的异常情况解决后，重新执行 MC_GroupEnable 功能块，完成轴组使能。

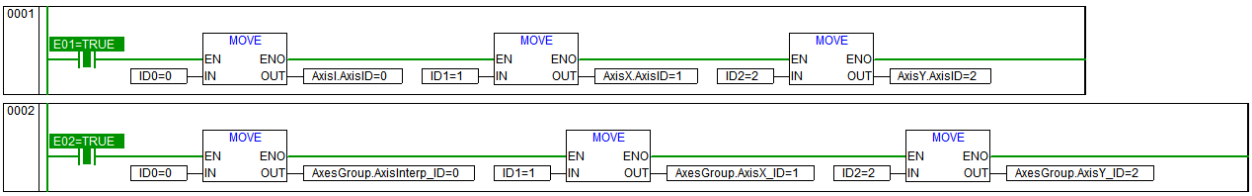
7.1.3 示例

■ 梯形图程序

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power0_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
Power1_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
GrpEnable1_Exe	BOOL	FALSE	轴组使能上升沿触发变量。

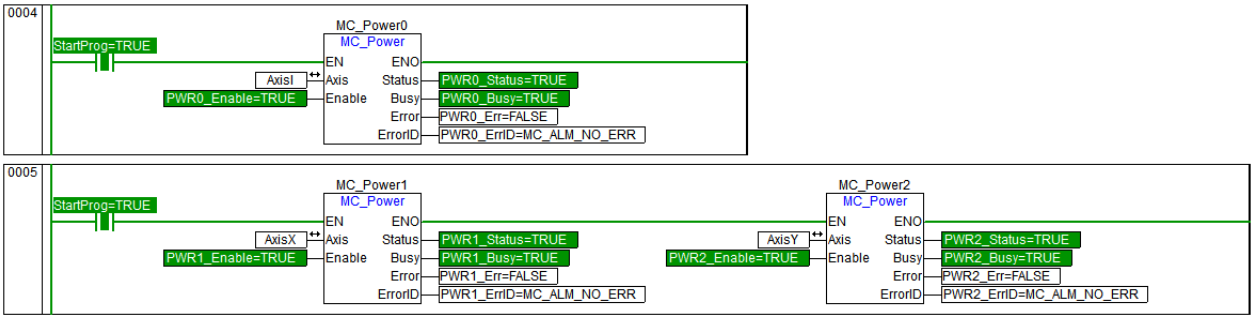
（1） 设定轴组中合轴和分轴的轴 ID。定义轴组时，首先需要定义轴组中的三个轴，将定义的三个轴的轴号 ID 与轴组中合轴和分轴的 ID 对应即可。见下述 0001 节和 0002 节。



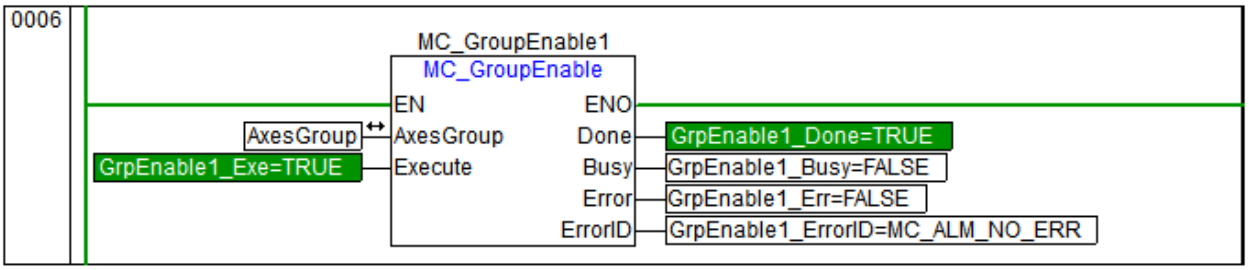
（2） 轴初始化。调用 SMC_AxisInit 指令分别完成轴组中三个轴的轴初始化配置，设置合轴为 0：Virtual，分轴类型为 50: EtherCAT_Position。SMC_AxisInit 指令使用方法参见各自指令章节介绍。

- 注意：轴组合轴的轴类型可以不设置，为默认的未使用轴时，MC_GroupEnable 指令自动设置合轴的轴类型为虚拟轴，并自行完成合轴使能。本实例中设定合轴为虚拟轴，需调用 MC_Power 完成合轴使能。

（3） 轴使能。轴组中各轴初始化完成，轴准备就绪状态 StartProg 为 TRUE 后，执行轴使能指令，完成轴使能。见 0004 ~ 0005 节。



（4） 轴组使能。调用 MC_GroupEnable 对轴组使能。轴组变量中的 AxesGroupDisabled 为 FALSE，轴组使能成功。见 0006 节。



■ ST 语言组建程序

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power0_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
Power1_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
GrpEnable1_Exe	BOOL	FALSE	轴组使能上升沿触发变量。

（1） 设定轴组中合轴和分轴的轴 ID。设定所定义的合轴 AxisI、分轴 AxisX 和 AxisY 的轴 ID。

```
IF Switch1 THEN
  AxisI.AxisID:=0;
  AxisX.AxisID:=1;
  AxisY.AxisID:=2;
  AxesGroup.AxisInterp_ID:=AxisI.AxisID;
  AxesGroup.AxisX_ID:=AxisX.AxisID;
```

```
  AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

（2） 调用 SMC_AxisInit 指令完成轴组中三个轴的轴初始化配置。轴组合轴轴类型设置为 0：Virtual；轴组分轴 AxisX/AxisY 轴类型均设置为 50：EtherCAT_Position。SMC_AxisInit 指令使用方法参见其指令章节介绍。

（3） 轴组中各轴初始化完成，轴准备就绪状态 StartProg 为 TRUE 后，执行轴使能指令，完成轴使能。

```
IF StartPro THEN
   MC_Power0(                          (*合轴使能*)
   Enable := PWR0_Enable,
   Axis := AxisI,
   Status=> PWR0_Status,
   Busy=> PWR0_Busy,
Error=> PWR0_Err,
ErrorID=> PWR0_ErrID);
MC_Power1(                          (*分轴 AxisX 使能*)
   Enable := PWR1_Enable,
   Axis := AxisX,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
Error=> PWR1_Err,
ErrorID=> PWR1_ErrID);
MC_Power2(                          (*分轴 AxisY 使能*)
   Enable := PWR2_Enable,
   Axis := AxisY,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
END_IF
```

（4） 执行轴组使能

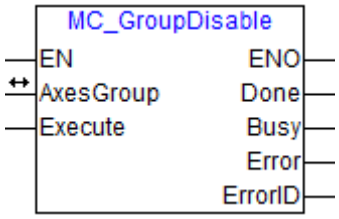
```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
```

7.1.4 使用注意事项

- 轴组使能后，禁止手动再次修改轴 ID；
- 轴组的 AxesGroupDisabled 状态变量通过 MC_GroupEnable/MC_GroupDisable 指令控制，不允许手动修改；
- 轴组使能后，禁止手动再把轴类型切换为轴组不支持的轴类型。

7.2 MC_GroupDisable（轴组断使能）

关闭轴组功能。



7.2.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	AxesGroup	轴组名	否	-	-	AXES_GROUP_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
OUTPUT	Done	完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

7.2.2 功能

本指令关闭轴组功能。

- 指令功能说明

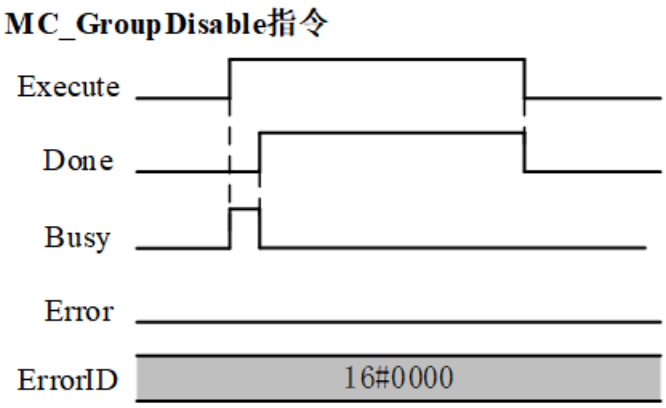
（1） Execute 上升沿该指令，轴参数检查正常，MC_GroupDisable 指令将指定的轴组中的

AxesGroupDisabled 变量置为 TRUE，关闭轴组功能。轴组处于 Disable 状态。

（2） 该指令执行成功后，将停止轴组的动作（不影响单轴和同步指令运行）。

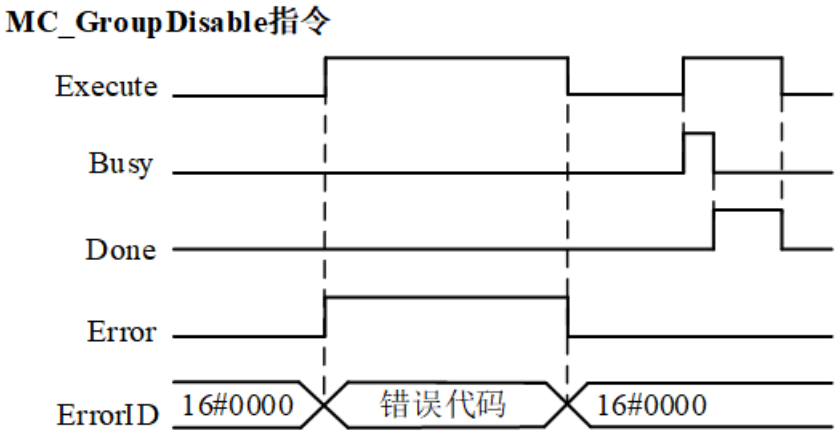
■ 时序图

（1） 正常时序图



（2） 异常错误发生时引脚时序

执行本指令时，轴组存在轴 ID 非法、轴组中轴类型非法等异常情况，调用本指令失败。此时，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，可以了解错误发生原因。指令输入引脚 Execute 为低电平时，所有输出引脚恢复为默认值。



- 当指令的异常情况解决后，重新执行 MC_GroupDisable 功能块，关闭轴组使能。

7.2.3 示例

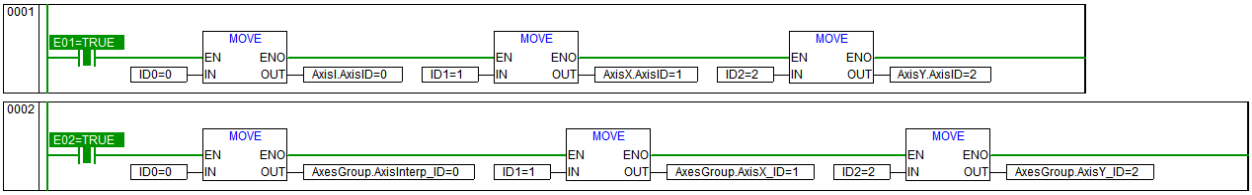
■ 变量

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power0_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
Power1_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
GrpDisable1_Exe	BOOL	FALSE	轴组断使能上升沿触发变量。

■ 梯形图程序

（1） 设定轴组中合轴和分轴的轴 ID。定义轴组时，首先需要定义轴组中的三个轴，将定义的三个轴的轴号 ID 与轴组中合轴和分轴的 ID 对应即可。见下述 0001 节和 0002 节。

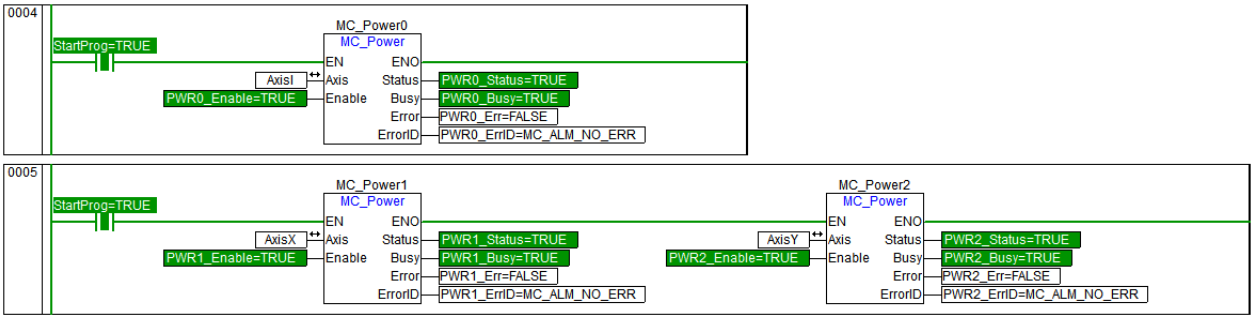


（2） 轴初始化。调用 SMC_AxisInit 指令分别完成轴组中三个轴的轴初始化配置，设置合轴为 0：Virtual，分轴类型为 50: EtherCAT_Position。SMC_AxisInit 指令使用方法参见各自指令章节介绍。

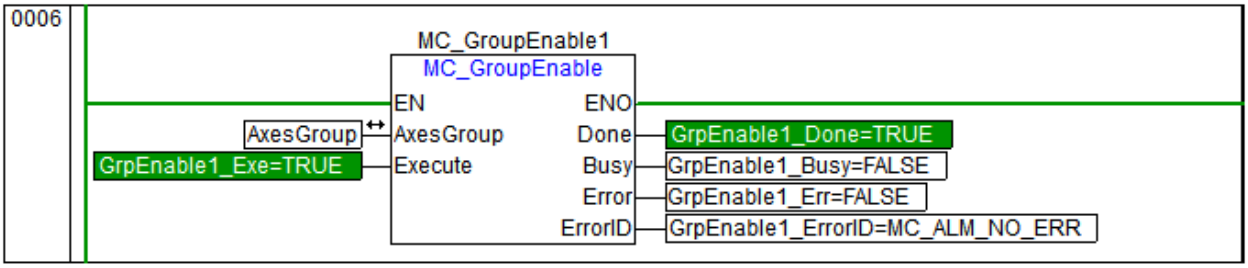
- 注意：轴组合轴的轴类型不设置，为默认的未使用轴时，MC_GroupEnable 指令自动设置合轴的轴类型为虚拟轴，并自行完成合轴使能。本实例中设定合轴为虚拟轴，需调用 MC_Power 完成合轴使能。

（3） 轴使能。轴组中各轴初始化完成，轴准备就绪状态 StartProg 为 TRUE 后，执行轴使能指令，完

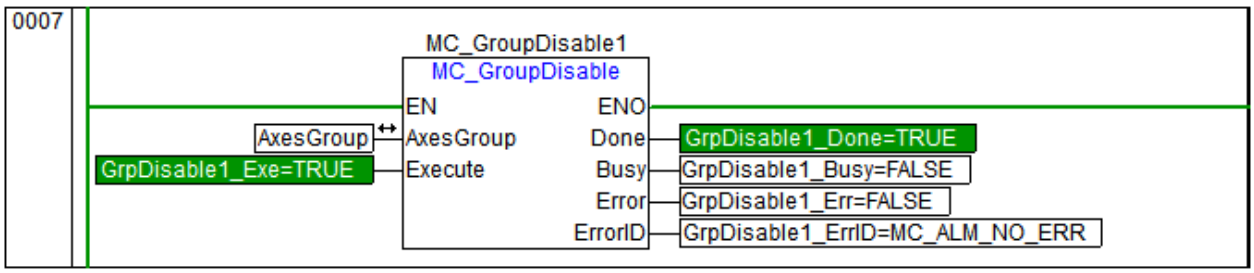
成轴使能。见 0004~0005 节。



（4） 轴组使能。调用 MC_GroupEnable 对轴组使能。轴组变量中的 AxesGroup. AxesGroupDisabled 为 FALSE，完成轴组使能。见 0006 节。



（5） 关闭轴组使能。调用 MC_GroupDisable 关闭轴组使能。轴组变量中的 AxesGroupDisabled 变为 TRUE，完成轴组使能。见 0007 节。



■ ST 语言组建程序

（1） 设定轴组中合轴和分轴的轴 ID。设定所定义的合轴 AxisI、分轴 AxisX 和 AxisY 的轴 ID。

```
IF Switch1 THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

（2） 调用 SMC_AxisInit 指令完成轴组中三个轴的轴初始化配置。轴组合轴轴类型设置为 0：Virtual；轴组分轴 AxisX/AxisY 轴类型均设置为 50：EtherCAT_Position。SMC_AxisInit 指令使用方法参见其指令章节介绍。

（3） 轴组中各轴初始化完成，轴准备就绪状态 StartPro 为 TRUE 后，执行轴使能指令，完成轴使能。

```
IF StartPro THEN
   MC_Power0(                          (*合轴使能*)
   Enable := PWR0_Enable,
   Axis := AxisI,
   Status=> PWR0_Status,
   Busy=> PWR0_Busy,
   Error=> PWR0_Err,
   ErrorID=> PWR0_ErrID);
   MC_Power1(                          (*分轴 AxisX 使能*)
   Enable := PWR1_Enable,
   Axis := AxisX,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
   Error=> PWR1_Err,
   ErrorID=> PWR1_ErrID);
   MC_Power2(                          (*分轴 AxisY 使能*)
   Enable := PWR2_Enable,
   Axis := AxisY,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
END_IF
```

（4） 执行轴组使能

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
```

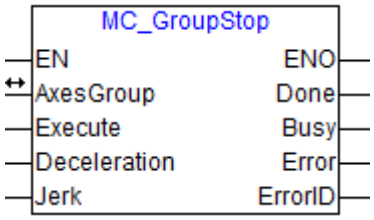
（5） 执行 MC_GroupDisable，关闭轴组功能

```
MC_GroupDisable1(
Execute := GrpDisable_Exe,
AxesGroup := AxesGroup,
```

```
Done=> GrpDisable_Done,
Busy=> GrpDisable_Busy,
Error=> GrpDisable_Err,
ErrorID=> GrpDisable_ErrorID);
```

7.3 MC_GroupStop（轴组停止）

控制轴组减速停止，中止轴组上任何正在执行的运动控制指令，并撤销该轴组合轴上所有运动缓存指令。



7.3.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	AxesGroup	轴组名	否	-	-	AXES_GROUP_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Deceleration	减速度	否	-	正值	LREAL
	Jerk	加加速度 0：梯形加减速 正值：S 形加减速	是	0	0/正值	LREAL
OUTPUT	Done	停止完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

7.3.2 功能

本指令控制轴组按照指定减速度，由当前速度减速停止，中止该轴组中合轴上任何正在执行的运动控制指令。

■ 指令功能详情

（1） 减速参数设定

减速停止时的减速度、加加速度由输入变量的 Deceleration(减速度)、Jerk(加加速度)指定。

（2） 运行本指令时的轴状态

指令上升沿触发引脚持续高电平时，轴组合轴一直处于 Stopping 状态时，轴组无法投送插补运动指令。调用 MC_GroupStop 时，仅轴组合轴处于 Stopping 状态。

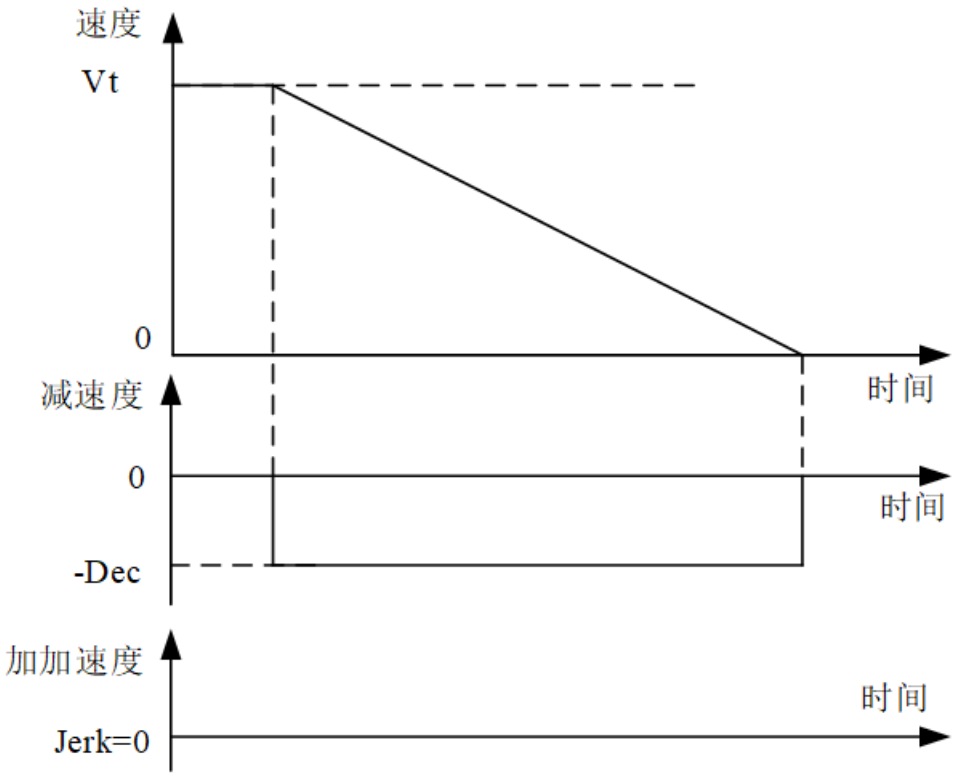
（3） 入参锁存

本指令上升沿触发时，将输入输出参数轴组的各轴 ID、输入参数 Deceleration 和 Jerk 值锁存，指令执行过程中，修改锁存的参数无效，直到重复触发本指令。另外，轴组使能后，禁止修改 AxesGroup 的合轴 ID、分轴 ID，以及轴类型等参数。

（4） 加减速曲线

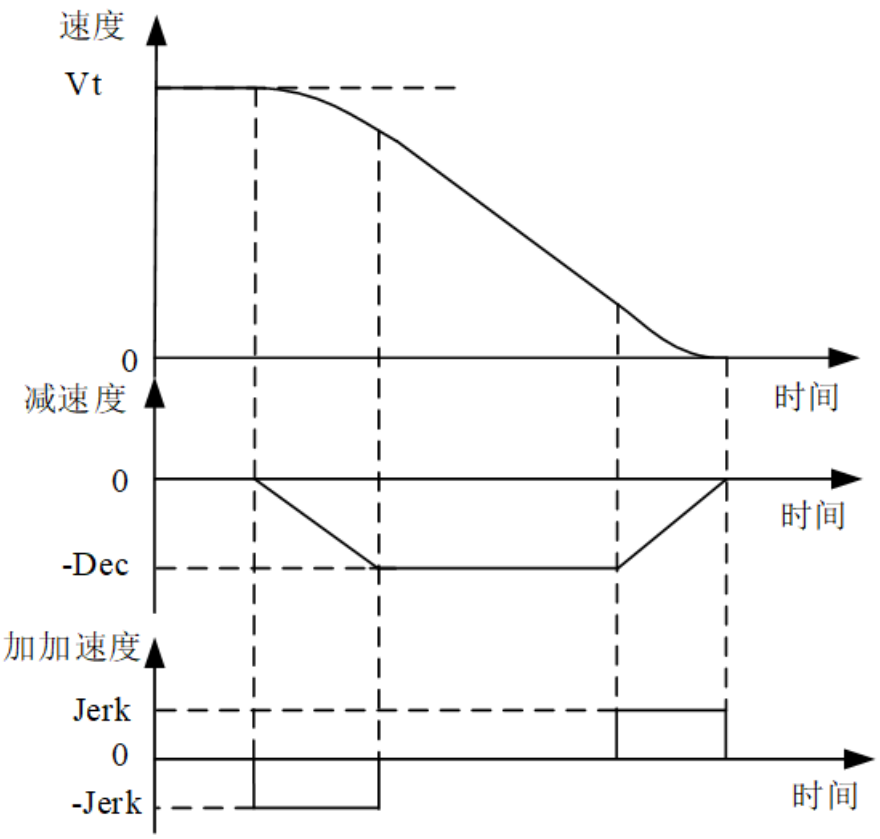
若输入参数 Jerk=0 时，则减速曲线为梯形加减速曲线；若输入参数 Jerk>0 时，则减速曲线为 S 形加减速曲线。

- Jerk=0 时，轴组运动减速曲线为梯形加减速曲线，如下所示：



Vt:减速启动时的速度，Dec：减速度的设定值，Jerk：加加速度的设定值

□ Jerk>0 时，以减速度生成速度的指令值。轴组运动减速曲线为 S 形加减速曲线，如下所示：



Vt:减速启动时的速度，Dec：减速度的设定值，Jerk：加加速度的设定值

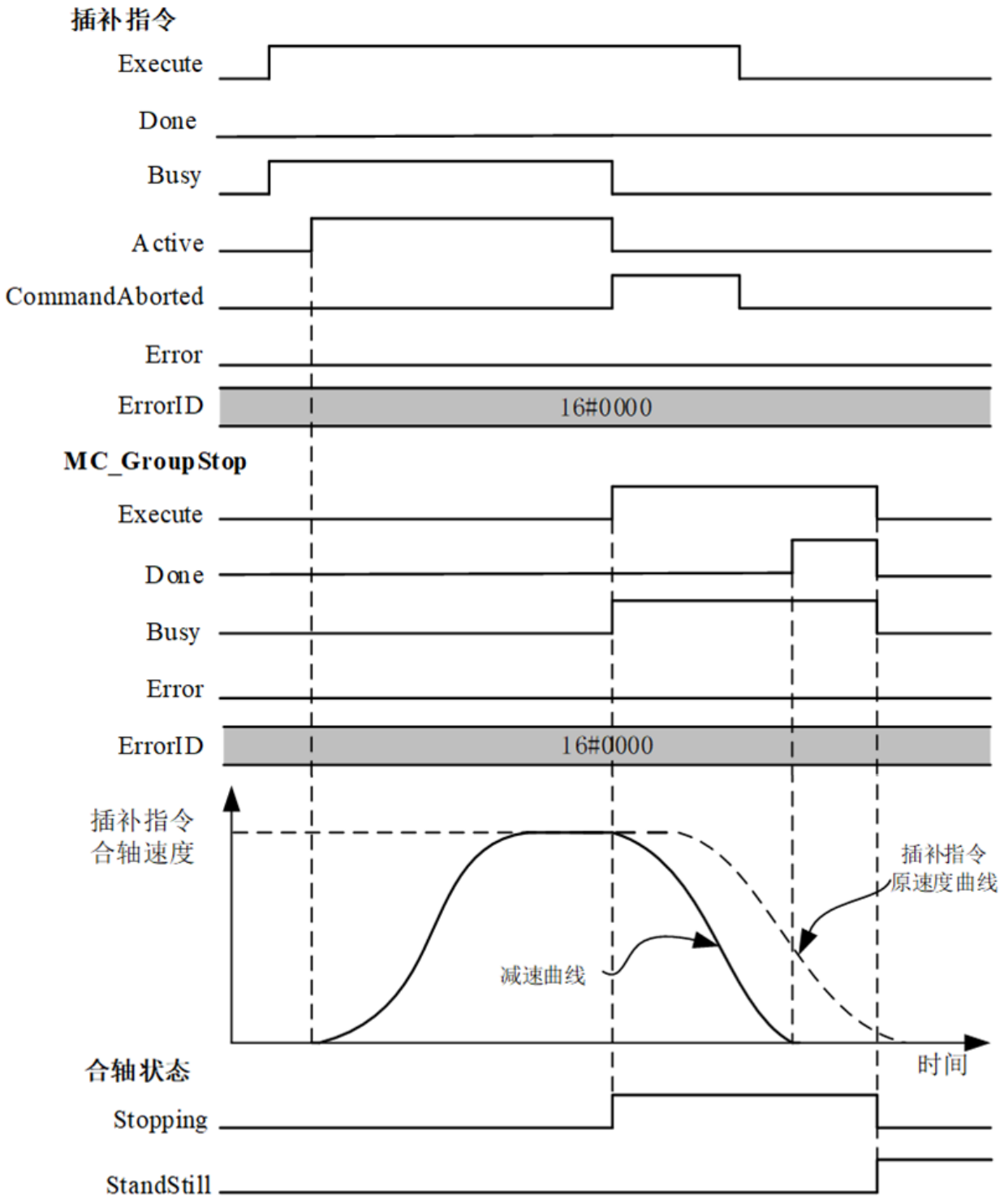
执行 MC_GroupStop 指令时，可以以设定的减速参数减速停止 SMC_MoveLinearAbsolute(2 轴绝对值直线插补)指令、SMC_MoveLinearRelative(2 轴相对值直线插补)指令、SMC_MoveCircular2D(2 轴圆弧插补)指令。

若轴组在执行插补指令过程中执行 MC_GroupStop 指令，则会在直线插补或圆弧插补的轨迹上减速停止。轴组中的插补指令被中断。

■ 时序图

（1） 正常时序图

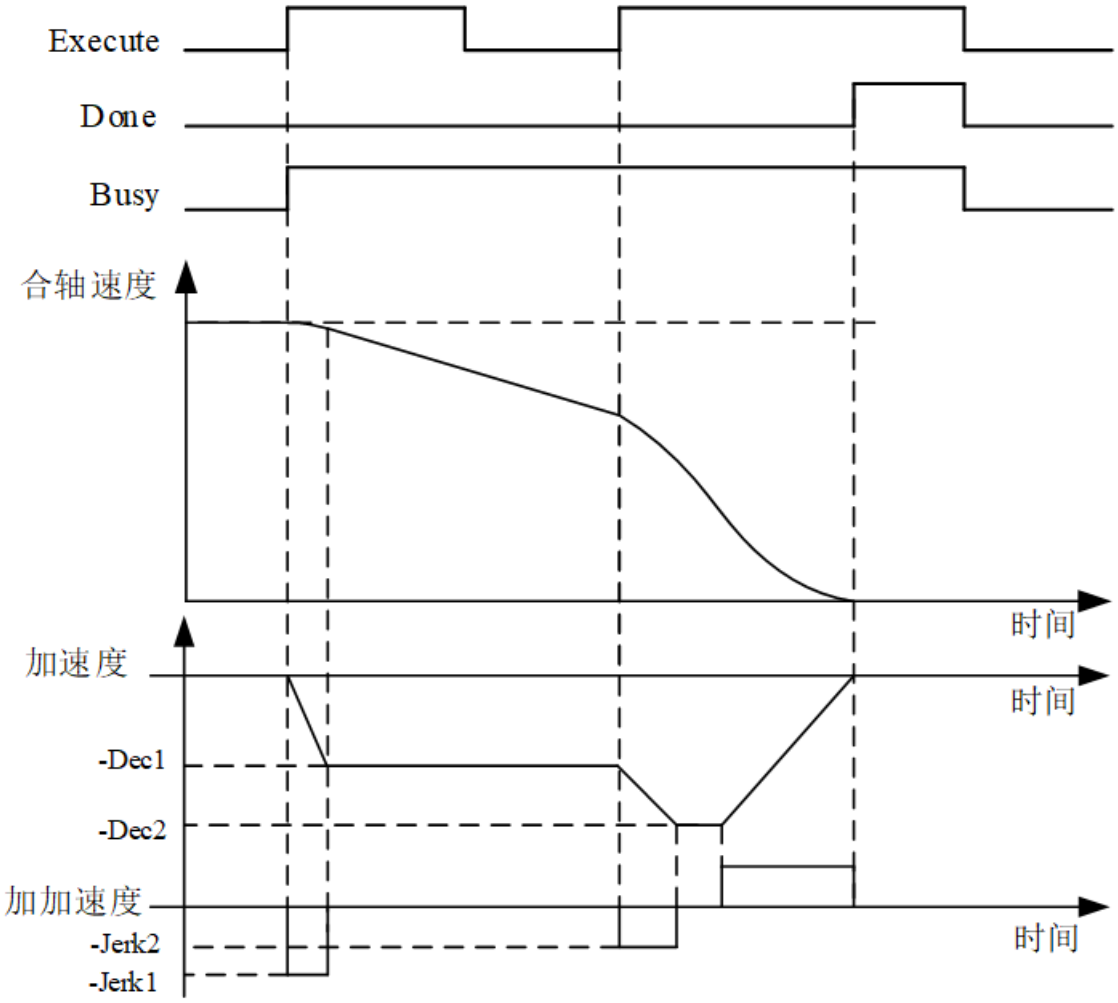
本指令上升沿触发有效。在 Execute 输入的上升沿后，若无异常情况，Busy 引脚置为 TRUE，指令执行时，轴组中合轴处于 Stopping 状态，轴组运动减速停止。轴组运动减速完成后 Done 置为 TRUE，若 Execute 保持高电平，则 Busy、Done 引脚保持 TRUE，轴组合轴持续处于 Stopping 状态；若 Execute 置低电平，则 Busy、Done 引脚置为 FALSE，轴组合轴切换为 StandStill 状态。



(2) 重复投送指令引脚时序

多个 MC_GroupStop 功能块重复触发时，以最后一次投送指令的 Deceleration（减速度）进行减速停止控制，指令对减速度和加加速度进行变更。

MC_GroupStop指令

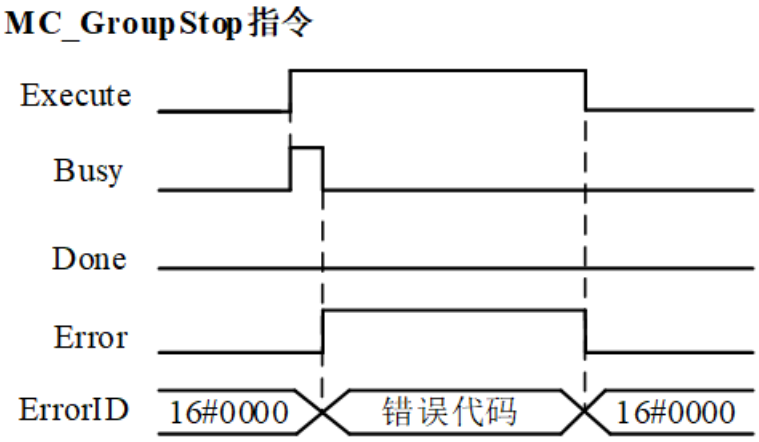


Dec1：第一次投送指令的减速度，Dec2：第二次投送时减速度

Jerk1：第一次投送指令的 JERK，Jerk2：第二次投送时的 Jerk

（3） 异常错误发生时引脚时序

执行本指令时，指令输入参数非法、轴组处于 Disabled 状态或者 ErrorStop 状态等异常时，调用本指令失败。此时，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，可以了解错误发生原因。指令输入引脚 Execute 为低电平时，所有输出引脚恢复为默认值。



- 当指令的异常情况解决后，需要重新上升沿触发 Execute，启动 MC_GroupStop 功能块，执行控制轴组按照指定减速度停止动作。

7.3.3 示例

本示例中，利用 MC_GroupStop 指令停止 2 轴相对位置直线插补运动指令，利用 MC_ReadStatus 指令读取轴组轴状态。

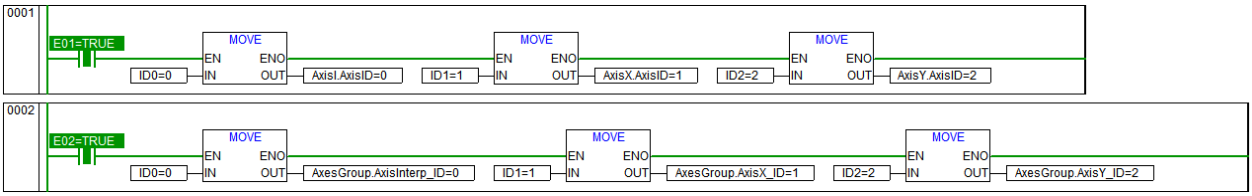
■ 梯形图程序

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power0_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
Power1_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
GrpEnable1_Exe	BOOL	FALSE	轴组使能上升沿触发变量。
Rel2D1_Exe	BOOL	FALSE	轴组投送的相对插补指令的上升沿触发变量。
Rel2D1_	BOOL	FALSE	轴组投送的相对插补指令的中断引脚变量。

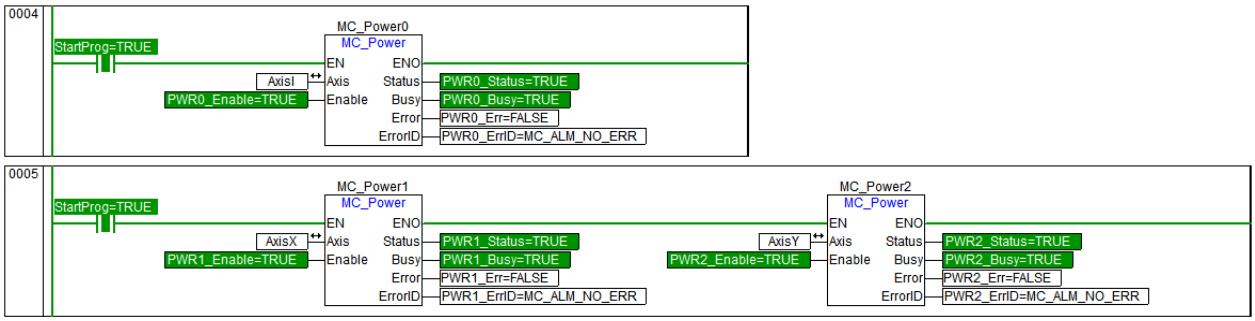
Comdabort			
GrpStop1_Busy	BOOL	FALSE	轴组投送的停止指令的 Busy 变量。
GrpStop1_Don3	BOOL	FALSE	轴组投送的停止指令的 Done 变量。合轴停止完成时，该变量变为 TRUE。
Stopping	BOOL	FALSE	轴组合轴的轴状态读取指令 Stopping 引脚变量。合轴开始减速停止时，该变量变为 TRUE。

（1） 设定轴组中合轴和分轴的轴 ID。定义轴组时，首先需要定义轴组中的三个轴，将定义的三个轴的轴号 ID 与轴组中合轴和分轴的 ID 对应即可。见下述 0001 节和 0002 节。

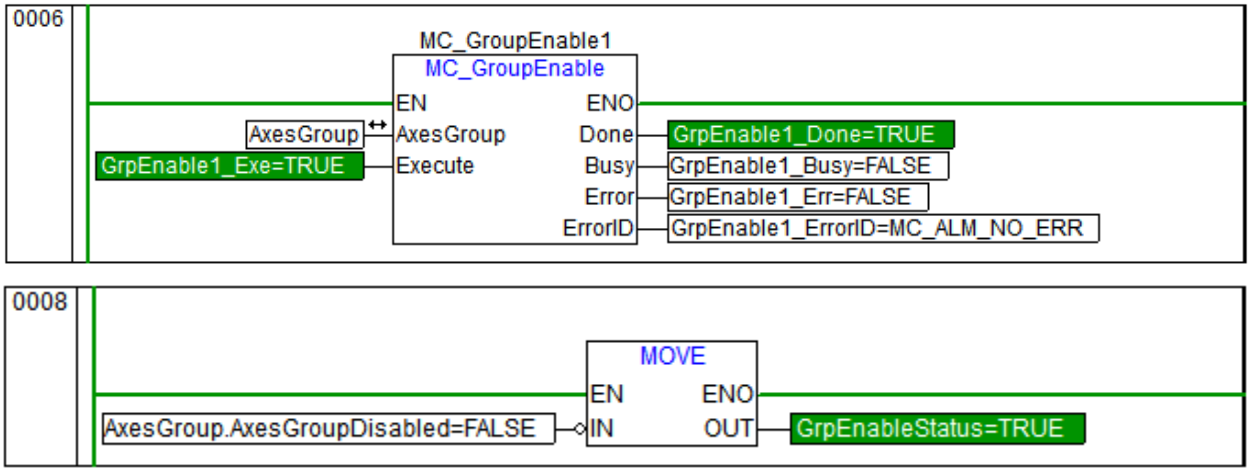


（2） 轴初始化。调用 SMC_AxisInit 指令分别完成轴组中三个轴的轴初始化配置，设置合轴为 0：Virtual，分轴类型为 50: EtherCAT_Position。SMC_AxisInit 指令使用方法参见各自指令章节介绍。

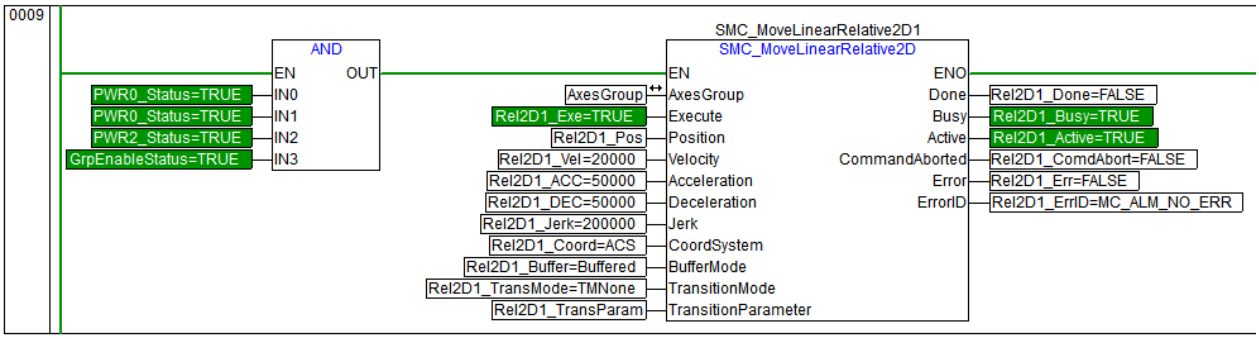
（3） 轴使能。轴组中各轴准备就绪（StartProg =TRUE）后，调用 MC_Power 对轴组的合轴和两个分轴进行轴使能。见 0004~0005 节。



（4） 轴组使能。调用 MC_GroupEnable 对轴组使能。轴组变量中的 AxesGroupDisabled 为 FALSE，轴组使能成功。见 0006 节。轴组使能成功后，将 AxesGroupDisabled 取反赋给 GrpEnableStatus 变量。

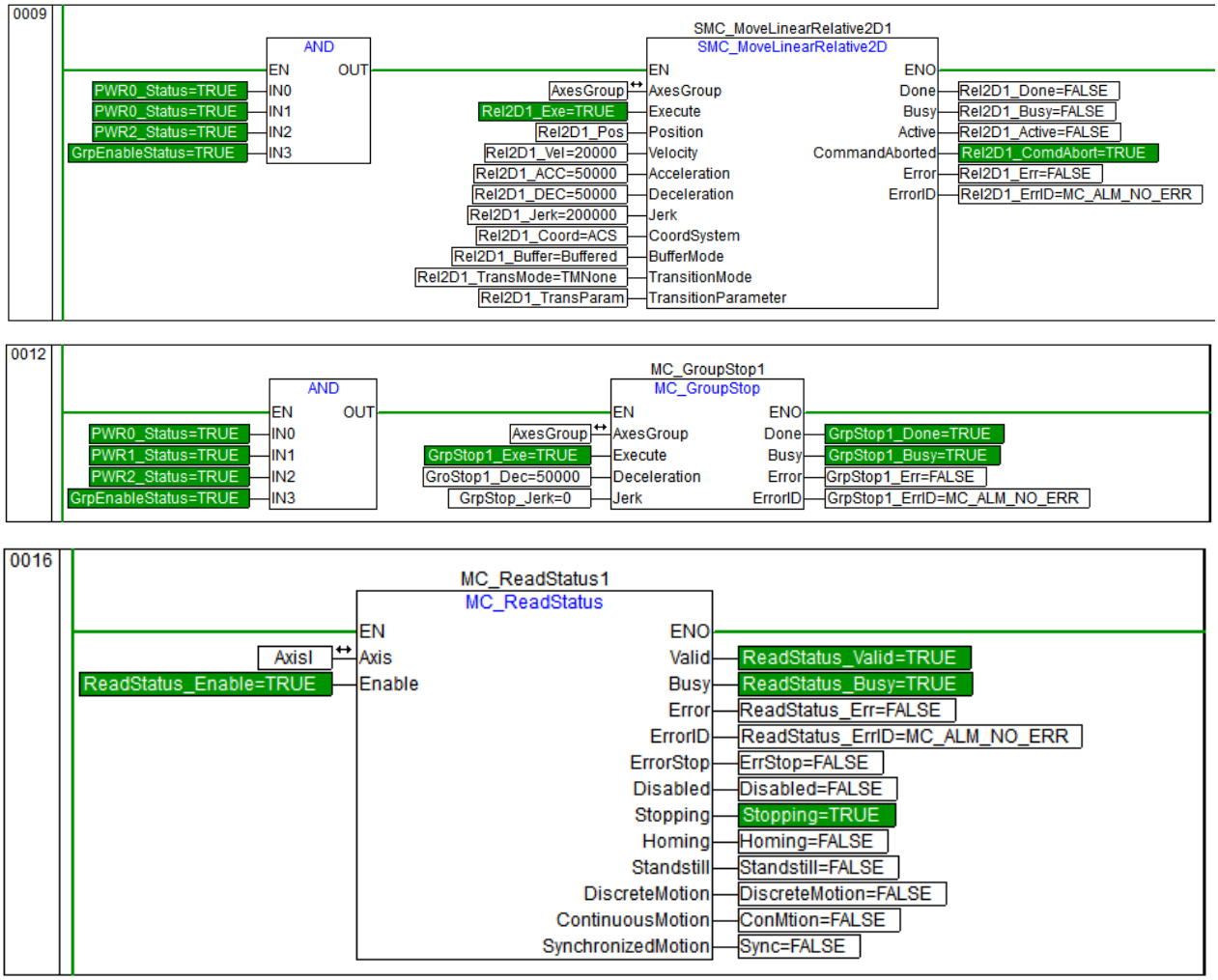


（5） 0009 节中，完成轴组中各轴的使能以及轴组使能后，触发相对位置插补运动指令，控制轴组运动。SMC_MoveLinearRelative2D 指令的使用参考其说明文档。



（6） 0012 节调用 MC_GroupStop 指令停止轴组插补运动，减速度设置为 50000，Jerk 为 0，采用梯形曲线减速。

当 MC_GroupStop 指令控制轴组减速停止时，插补运动指令 SMC_MoveLinearRelative2D1 的输出引脚 CommandAborted 置 TRUE，插补指令被中断见下述 0009 节。此时，查看 0016 节中的 MC_ReadStatus1 的输出引脚，确定轴组合轴的状态。



■ ST 语言

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
InitFlag	BOOL	FALSE	如果该变量为 TRUE，初始化完成。
Power0_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
Power1_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。

GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
Rel2D1_Exe	BOOL	FALSE	轴组投送的插补指令的上升沿触发变量。
GrpStop_Exe	BOOL	FALSE	轴组投送的停止指令的上升沿触发变量。

（1） 设定轴组中合轴和分轴的轴 ID。设定所定义的合轴 AxisI、分轴 AxisX 和 AxisY 的轴 ID。

```
IF FALSE=InitFlag THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
   GrpStop_Dec:=50000;
   GrpStop_Jerk:=0;
   InitFlag:=TRUE;
END_IF
```

（2） 调用 SMC_AxisInit 指令完成轴组中三个轴的轴初始化配置。轴组合轴轴类型设置为 0：Virtual；轴组分轴 AxisX/AxisY 轴类型均设置为 50：EtherCAT_Position。SMC_AxisInit 指令使用方法参见其指令章节介绍。

（3） 执行 MC_Power 指令完成轴组中三个轴的轴使能。

```
IF StartPro THEN
   MC_Power0(                          (*合轴使能*)
   Enable := PWR0_Enable,
   Axis := AxisI,
   Status=> PWR0_Status,
   Busy=> PWR0_Busy,
   Error=> PWR0_Err,
   ErrorID=> PWR0_ErrID);
   MC_Power1(                          (*分轴 AxisX 使能*)
   Enable := PWR1_Enable,
   Axis := AxisX,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
   Error=> PWR1_Err,
   ErrorID=> PWR1_ErrID);
   MC_Power2(                          (*分轴 AxisY 使能*)
   Enable := PWR2_Enable,
   Axis := AxisY,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
```

```
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
END_IF
```

（4） 执行轴组使能

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

（5） 轴以及轴组使能功能后，上升沿触发轴组插补指令，SMC_MoveLinearRelative2D 指令使用方法参见其指令章节介绍。

```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus THEN
   SMC_MoveLinearRelative2D1(
   Execute := Rel2D1_Exe,
   Position := Rel2D1_Pos,
   Velocity := Rel2D1_Vel,
   Acceleration := Rel2D1_ACC,
   Deceleration := Rel2D1_DEC,
   Jerk := Rel2D1_Jerk,
   CoordSystem := Rel2D1_Coord,
   BufferMode := Rel2D1_Buffer,
   TransitionMode := Rel2D1_Trans,
   TransitionParameter := Rel2D1_Param,
   AxesGroup := AxesGroup,
   Done=> Rel2D1_Done,
   Busy=> Rel2D1_Busy,
   Active=> Rel2D1_Active,
   CommandAborted=> Rel2D1_ComdAbort,
   Error=> Rel2D1_Err,
   ErrorID=> Rel2D1_ErrID);
END_IF
```

（6） 调用 MC_GroupStop 指令停止轴组插补运动，减速度设置为 50000，Jerk 为 0，梯形曲线减速。当 MC_GroupStop 指令控制轴组减速停止时，插补运动指令 SMC_MoveLinearRelative2D1 的输出引脚 CommandAborted 置 TRUE，插补指令被中断。此时， MC_ReadStatus1 的输出引脚，确定轴组合轴的状态。

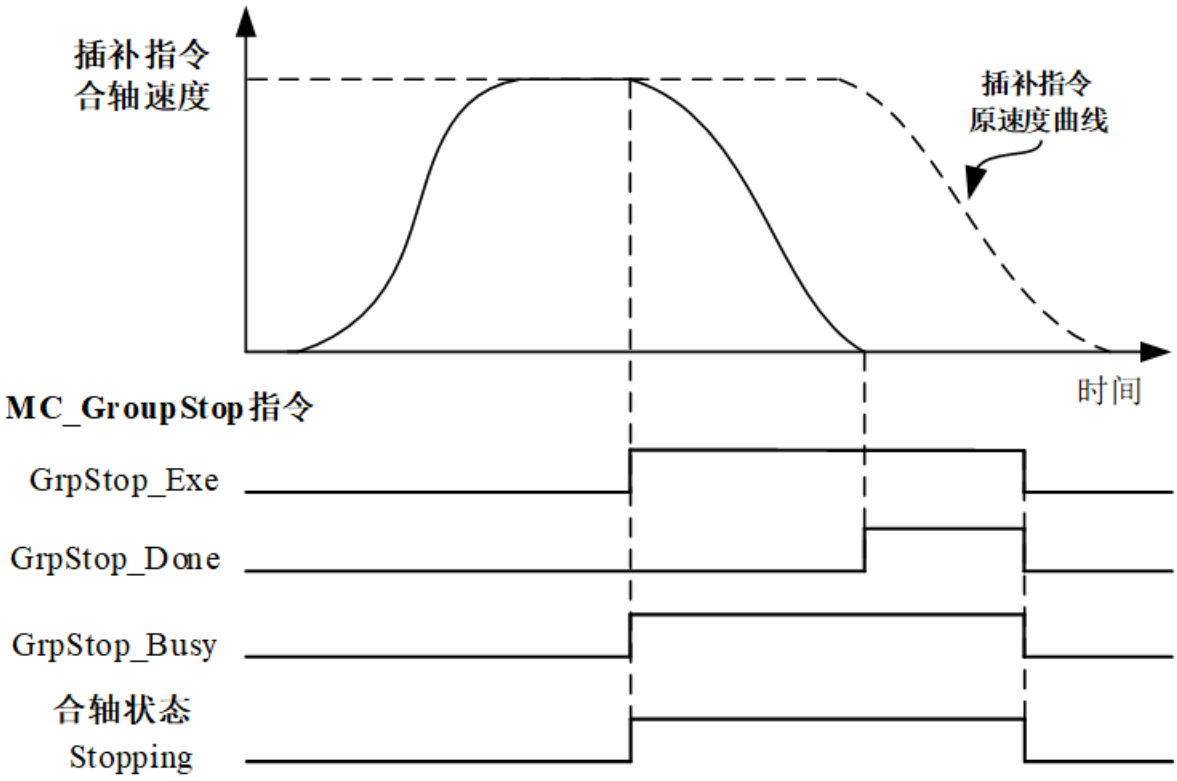
```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus THEN
   MC_GroupStop1(
```

```
    Execute := GrpStop_Exe,
    Deceleration := GrpStop_Dec,
    Jerk := GrpStop_Jerk,
    AxesGroup := AxesGroup,
    Done=> GrpStop_Done,
    Busy=> GrpStop_Busy,
    Error=> GrpStop_Err,
    ErrorID=> GrpStop_ErrID);
END_IF
```

（7） 读取合轴的轴状态

```
MC_ReadStatus1(
Enable := ReadStatus Enable,
Axis := AxisI,
Valid=> ReadStatus_Valid,
Busy=> ReadStatus_Busy,
Error=> ReadStatus_Err,
ErrorID=> ReadStatus ErrID,
ErrorStop=>ErrStop,
Disabled=>Disabled,
Stopping=>Stopping,
Homing=>Homing,
Standstill=>Standstill,
DiscreteMotion=>DiscreteMotion,
ContinuousMotion=>ConMtion,
SynchronizedMotion=>Sync);
```

下图为轴组插补指令被 MC_GroupStop 中断停止示意图。

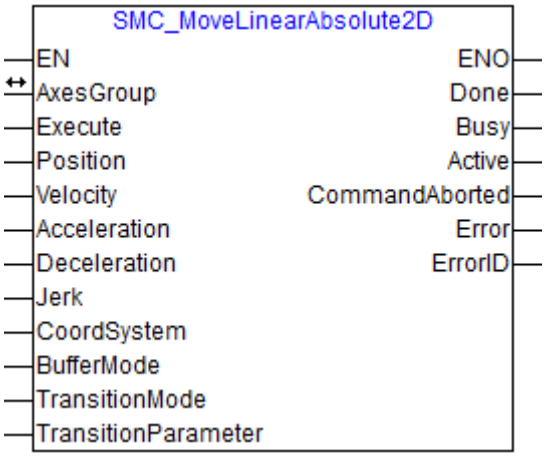


7.3.4 使用注意事项

- 调用 MC_GroupStop 时，轴组需要完成使能。
- 调用 MC_GroupStop 时，使轴组运动成功停止后，若 MC_GroupStop 的输入引脚 Execute 为高电平时，则轴组合轴保持 Stopping 状态，此时，投送插补指令失败。当输入引脚 Execute 为低电平时，轴组合轴切换到 StandStill 状态，可以正常投送插补指令。

7.4 SMC_MoveLinearAbsolute2D（2 轴绝对位置直线插补）

该指令实现 2 轴绝对位置直线插补功能。



7.4.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	AxesGroup	轴组名	否	-	-	AXES_GROUP_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Position	目标位置（绝对）	否	-	正值/负值/0	ARRAY[0..1] OF LREAL
	Velocity	目标速度	否	-	非负	LREAL
	Acceleration	加速度	否	-	正值	LREAL
	Deceleration	减速度	否	-	正值	LREAL
	Jerk	加加速度 0：梯形加减速 正值：S 形加减速	是	0	0/正值	LREAL
	CoordSystem	坐标系 0：轴坐标系	是	0	0: ACS	MC_COORD_SYSTEM
	BufferMode	缓冲模式 1：缓存 2：以相邻指令间的低速融合 3：以上一条指令的速度融合 4：以下一条指令的速度融合 5：以相邻指令间的高速融合	是	1	1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE
	TransitionMode	过渡曲线选择	是	0	0：TMNone	MC_TRANSITION_MODE

		0：不插入过渡曲线				
	TransitionParameter	保留参数	是	0	-	ARRAY[0..1] OF LREAL
OUTPUT	Done	插补运动完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为TRUE	是	FALSE	TRUE/FALSE	BOOL
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

7.4.2 功能

该指令实现 2 轴绝对位置直线插补功能。

■ 功能说明

（1） 本指令实现 2 轴的直线插补，目标位置以绝对位置指令。

（2） 目标位置

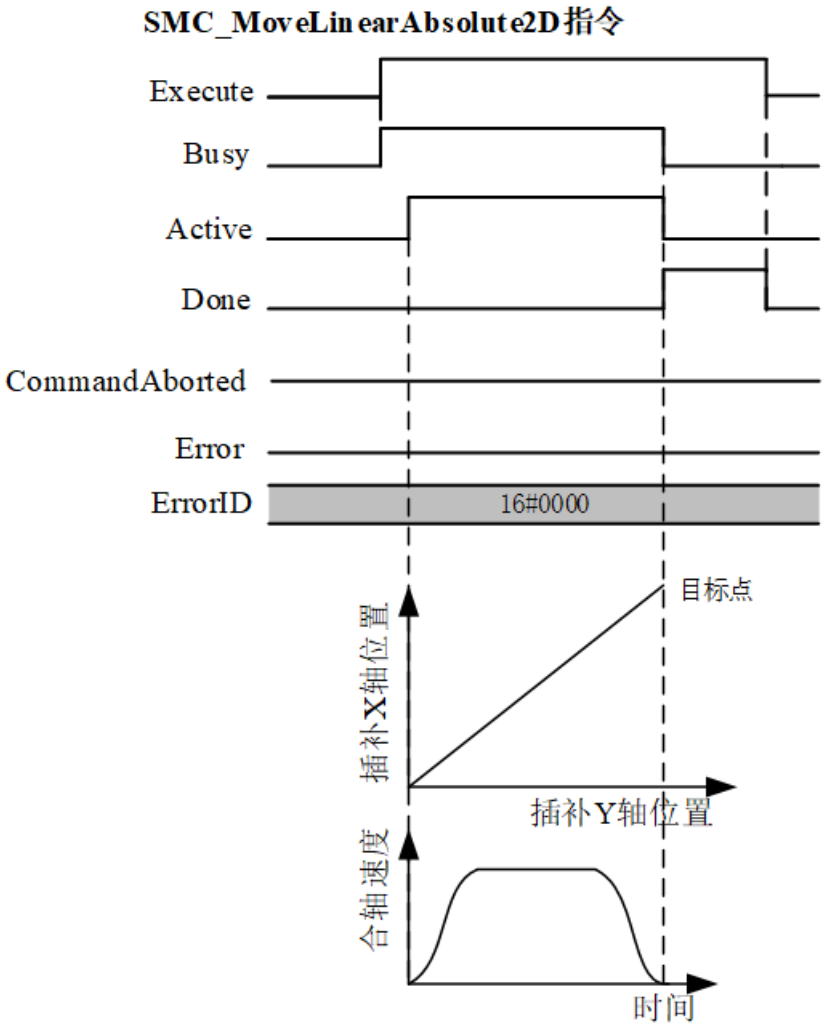
此指令的目标位置都是绝对位置。以本指令开始执行时刻为起点，以指令输入参数设置的两分轴绝对位置为终点，起点和终点的连线即为该指令的运动轨迹。

（3） 其他功能与 SMC_MoveLinearRelative2D(相对位置直线插补)指令功能相同。

■ 时序图

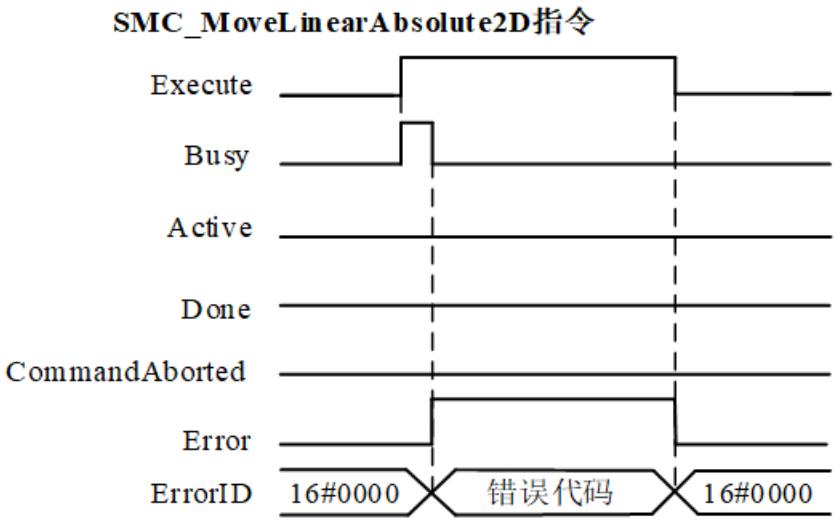
下述为绝对位置插补指令执行时的正常、异常时序图。

（1） 正常时序图



(2) 异常错误发生时引脚时序

执行本指令时，若发生参数非法、轴组中有轴为轴类型为 0：Unused（未使用轴），轴组中各轴 ID 非法等异常，调用本指令失败，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Execute 变为低电平时，所有输出引脚恢复为默认值。



- 当指令的异常情况解决后，需要重新上升沿触发 Execute，投送轴组插补指令，进行轴组绝对位置插补运动。

7.4.3 示例

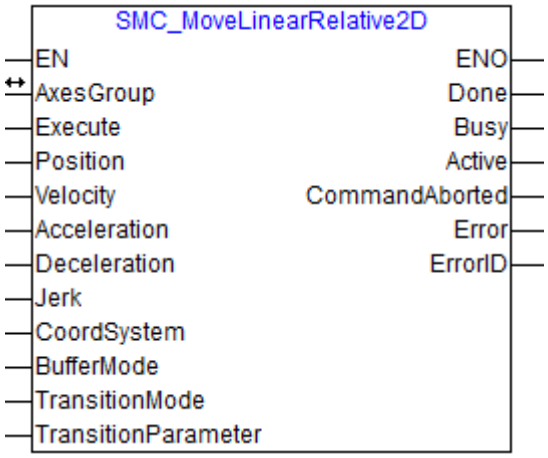
SMC_MoveLinearAbsolute2D 示例参考 SMC_MoveLinearRelative2D 指令。两种指令唯一区别在于目标位置是绝对位置或者相对位置。

7.4.4 使用注意事项

同 SMC_MoveLinearRelative2D 指令注意事项。

7.5 SMC_MoveLinearRelative2D（2 轴相对位置直线插补）

该指令实现 2 轴相对位置直线插补功能。



7.5.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	AxesGroup	轴组名	否	-	-	AXES_GROUP_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Position	目标位置（相对）	否	-	正值/负值/0	ARRAY[0..1] OF LREAL
	Velocity	目标速度	否	-	非负	LREAL
	Acceleration	加速度	否	-	正值	LREAL
	Deceleration	减速度	否	-	正值	LREAL
	Jerk	加加速度 0：梯形加减速 正值：S 形加减速	是	0	0/正值	LREAL
	CoordSystem	坐标系 0：轴坐标系	是	0	0: ACS	MC_COORD_SYSTEM
	BufferMode	缓冲模式 1：缓存 2：以相邻指令间的低速融合 3：以上一条指令的速度融合 4：以下一条指令的速度融合 5：以相邻指令间的高速融合	是	1	1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE

	TransitionMode	过渡曲线选择 0：不插入过渡曲线	是	0	0：TMNone	MC_TRANSITION_MODE
	TransitionParameter	保留参数	是	0	-	ARRAY[0..1] OF LREAL
OUTPUT	Done	插补运动完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为TRUE	是	FALSE	TRUE/FALSE	BOOL
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

7.5.2 功能

该指令实现 2 轴相对位置直线插补功能。

- 插补步骤

（1） 确定插补的轴组。轴组 AxesGroup 变量类型为 AXES_GROUP_REF。

（2） 指定轴组变量中的轴号，对应于合轴和分轴的轴号。

（3） 完成各轴的使能以及轴组使能。

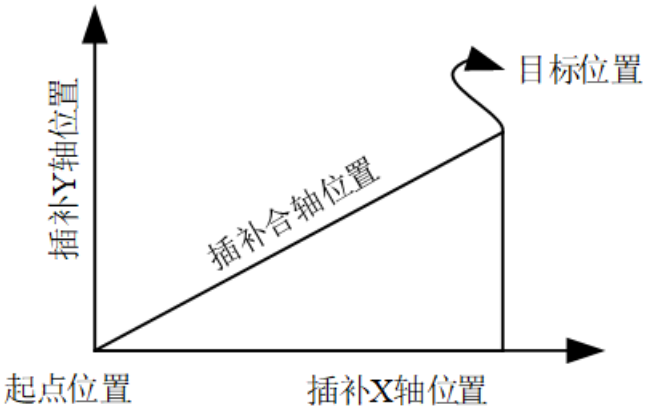
（4） 调用相对位置直线插补指令执行直线插补动作。

- 基本功能

  - 目标位置

以本指令开始执行时刻为起点，以指令输入参数设置的两分轴相对位置为终点，起点和终点的连线即为该指令的运动轨迹。

本指令的目标位置都是根据上一次运动的终点位置做增量运动，即每一步都是增量模式运动。插补合轴的目标位置在指令开始执行时刻位置基础上，增加以两分轴增量位置为直角边的直角三角形的斜边长度。



- 入参锁存

本指令上升沿触发有效。在 Execute 输入的上升沿时，将输入输出参数 AxesGroup 的合轴以及两个分轴的 ID、输入参数 Position、Velocity、Acceleration、Deceleration 和 Jerk 等参数值锁存，指令执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令。

- 运动参数设定

插补运动合轴运动速度由本指令输入参数 Velocity 设定，加减速分别由入参 Acceleration、Deceleration 设定，加加速度由入参 Jerk 设定。

分轴运动速度由合轴指令速度实时关联计算。

- 限位

插补分轴遇到限位时，选取合轴分解计算的分轴减速度和自身快减速度两者的最大值进行最优保护停止。

- 功能块重复触发的处理

功能块指令等待重复触发时，重复触发的指令被放入运动缓存中，功能块监控的状态为后一条指令的运行状态，前一条指令的运行状态则失去监控。在前一个插补指令执行期间，后一个插补指令等待前一个插补指令结束后，开始执行。

- BufferMode (缓冲模式选择)

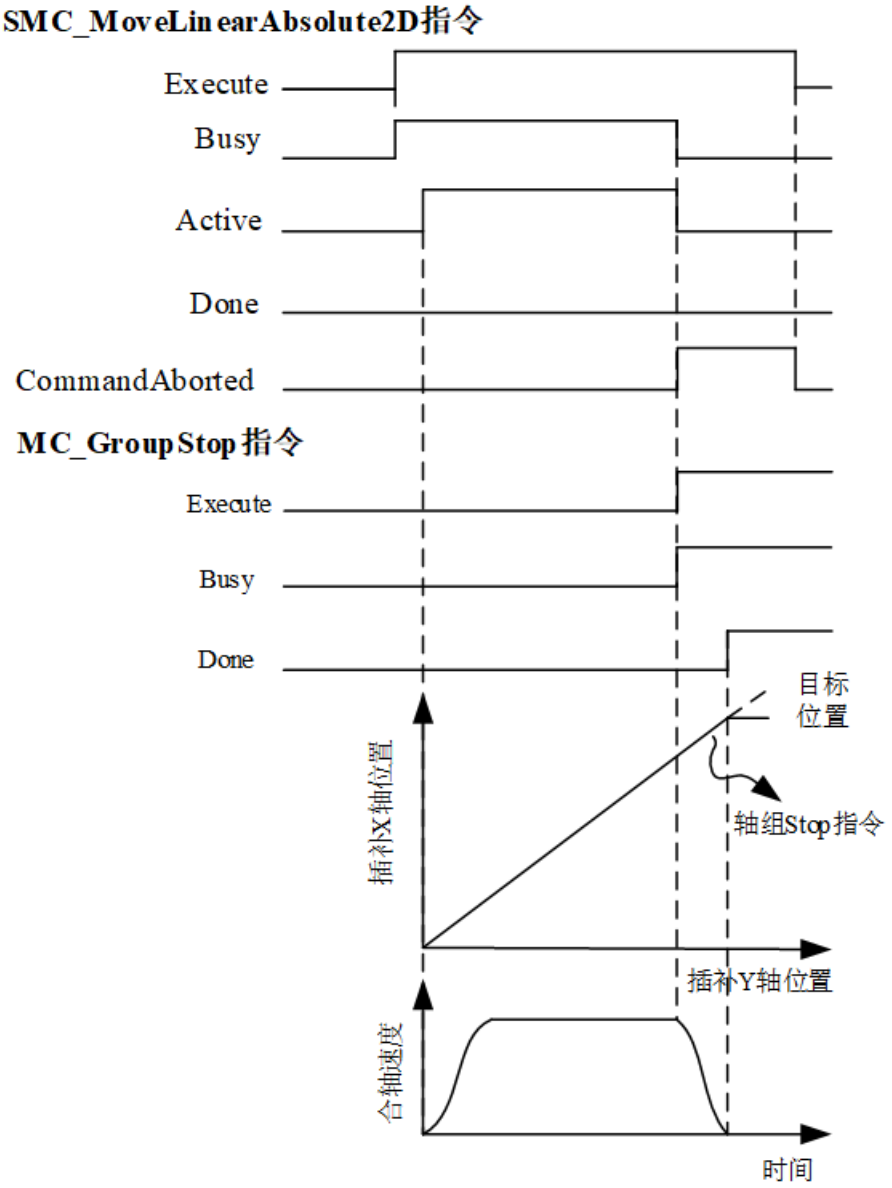
相对位置直线插补指令支持 BufferMode (缓冲模式选择)，BufferMode 支持 1：Buffered，2：BlendingLow、3: BlendingPrevious、4: BlendingNext 以及 5: BlendingHigh。

投送本指令时，若 BufferMode 为 1：Buffered，从当前执行中的指令正常结束的周期起，已缓存的本指令自动启动。若 BufferMode 为 2：BlendingLow，则以相邻指令间的低速融合。若 BufferMode 为 3: BlendingPrevious，则以上一条指令的速度融合。若 BufferMode 为 4: BlendingNext，则以下一条指令的速度融合。若 BufferMode 为 5: BlendingHigh，则以相邻指令间的高速融合。

插补指令的缓冲模式 BufferMode 详情请参考附录章节的缓冲模式说明。

- 指令中断

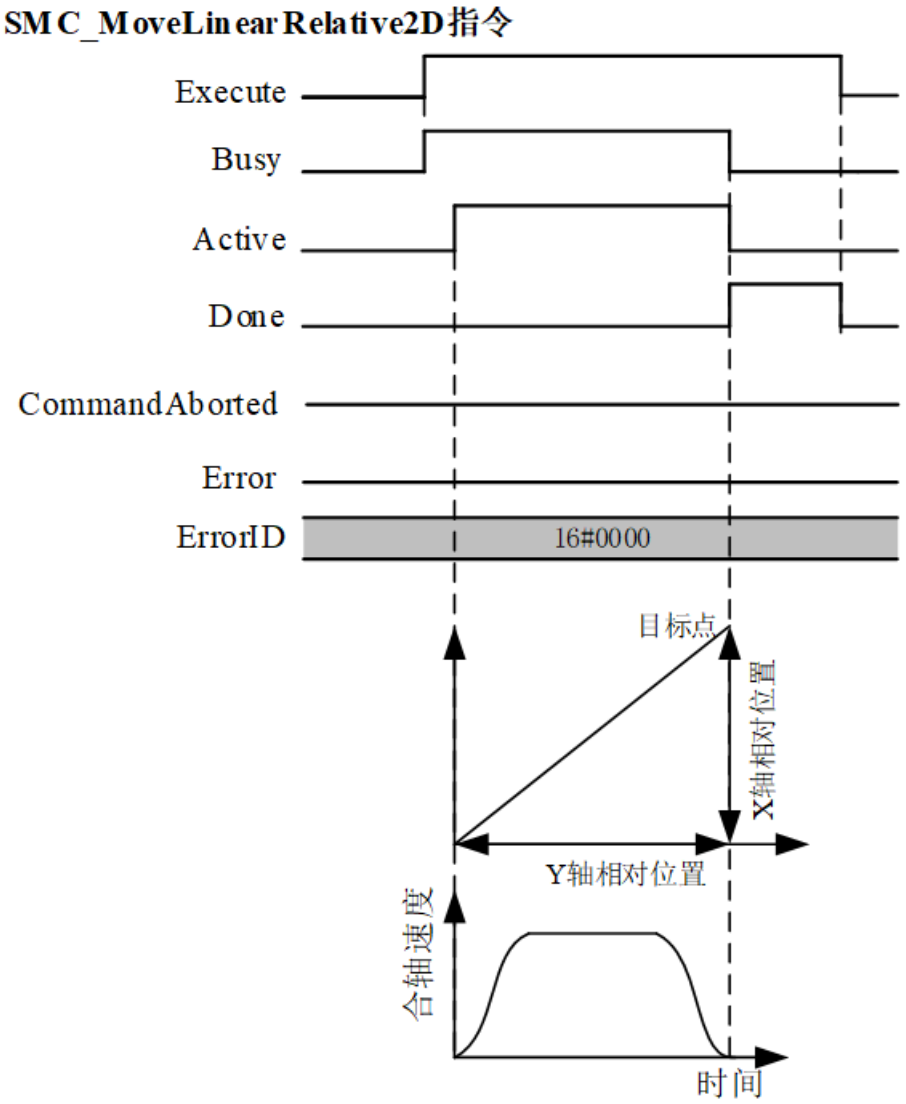
执行轴组插补指令过程中，可以使用 MC_GroupStop 指令中断插补指令。轴组各轴沿着插补运动轨迹减速停止。



- CoordSystem(坐标系)
  - 指定进行相对位置直线插补的坐标系。
  - 使用轴坐标系(ACS)。
- 时序图

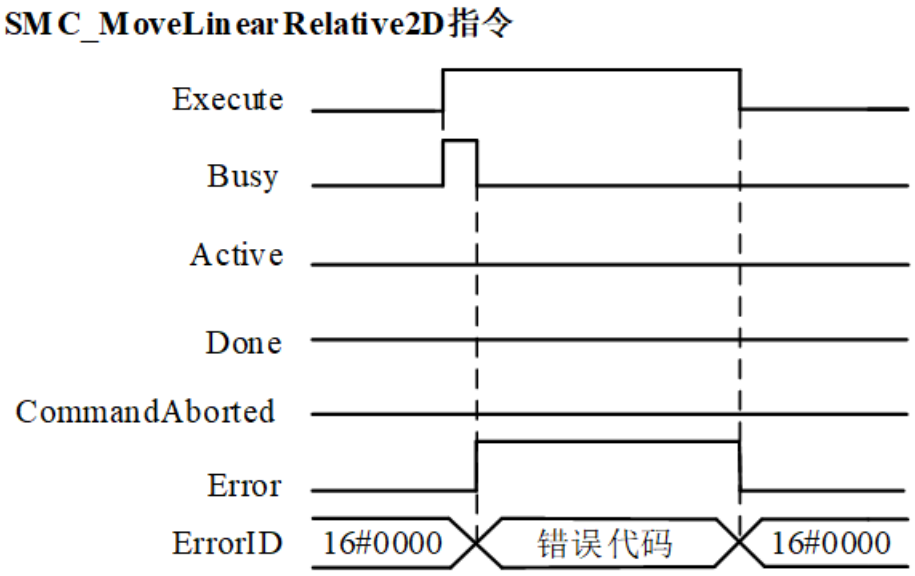
下述为相对对位置插补指令执行时的正常、异常时序图。

（1） 正常时序图



(2) 异常错误发生时引脚时序

执行本指令时，若发生参数非法、轴组中有轴为轴类型为 0：Unused（未使用轴），轴组中各轴 ID 非法等异常，调用本指令失败，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Execute 变为低电平时，所有输出引脚恢复为默认值。



- 当指令的异常情况解决后，需要重新上升沿触发 Execute，投送轴组插补指令，进行轴组相对位置插补运动。

7.5.3 示例

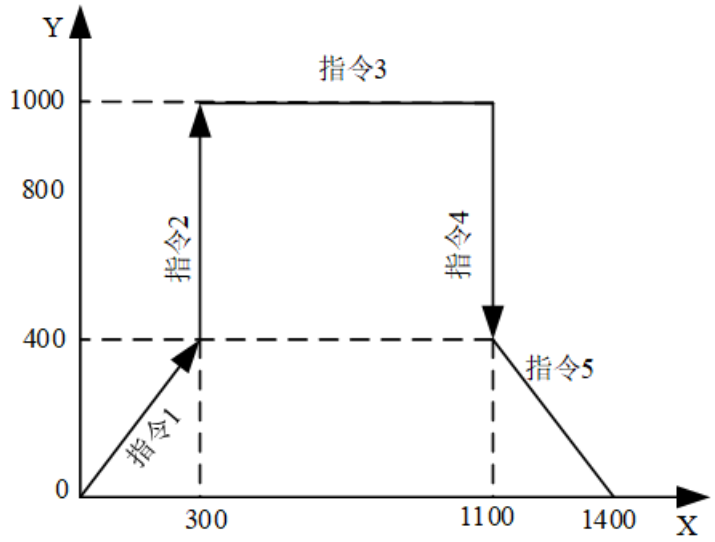
- 动作示例
  - 动作介绍

    以下示例执行 5 条相对位置直线插补指令，执行完插补指令后将移动到目标位置。

    设定插补指令的缓存模式 BufferMode 选择指定为缓存 Buffered，缓存多个运动指令。

    在本例中，直线插补指令 1 的输出 Active 变为 TRUE 时，执行其余插补指令 2-5。

  - 动作示意图



依次执行直线插补指令 1 到达（300,400）位置处，执行直线插补指令 2 到达（300,1000）位置处，执行直线插补指令 3 到达（1100,1000）位置处，执行直线插补指令 4 到达（1100,400）位置处，执行直线插补指令 3 到达（1400, 0）位置处。

■ 梯形图程序

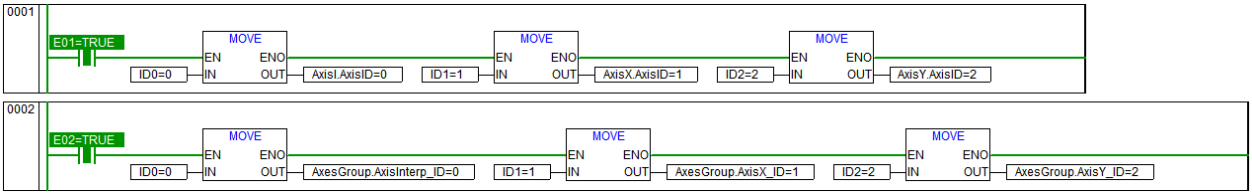
设定各插补指令的速度为 100，加减速度均为 200、加加速度为 5000，缓存模式为 Buffered。

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power0_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
Power1_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
GrpEnable1_Exe	BOOL	FALSE	轴组使能上升沿触发变量。
AxisHomed	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴回原点完成。
Rel2D1_Exe	BOOL	FALSE	轴组投送的相对插补指令的上升沿触发变量。
Rel2D1_Active	BOOL	FALSE	轴组投送的相对插补指令的 Active 引脚的变量，该变量变为 TRUE 时，

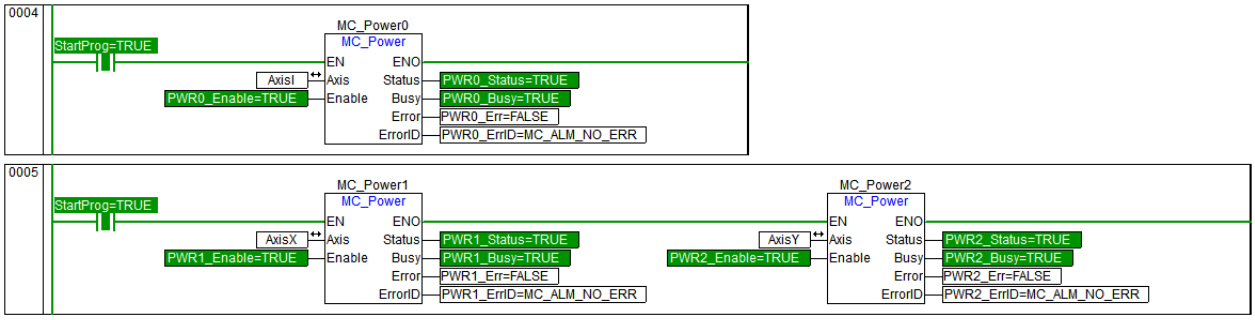
			轴组执行插补指令。

（1） 设定轴组中合轴和分轴的轴 ID。定义轴组时，首先需要定义轴组中的三个轴，将定义的三个轴的轴号 ID 与轴组中合轴和分轴的 ID 对应即可。见下述 0001 节和 0002 节。

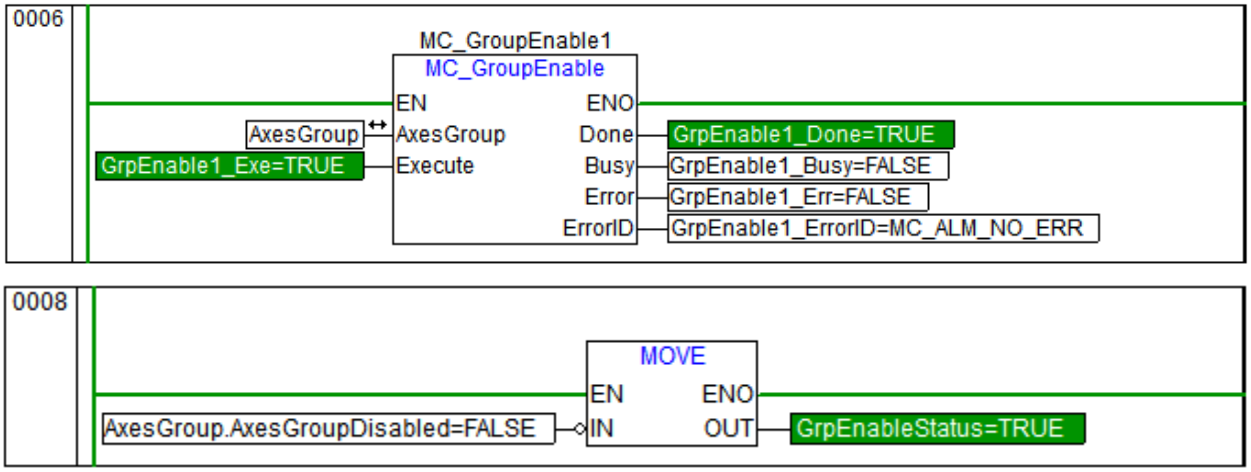


（2） 轴初始化。调用 SMC_AxisInit 指令分别完成轴组中三个轴的轴初始化配置，设置合轴为 0：Virtual，分轴类型为 50: EtherCAT_Position。SMC_AxisInit 指令使用方法参见各其指令章节介绍。

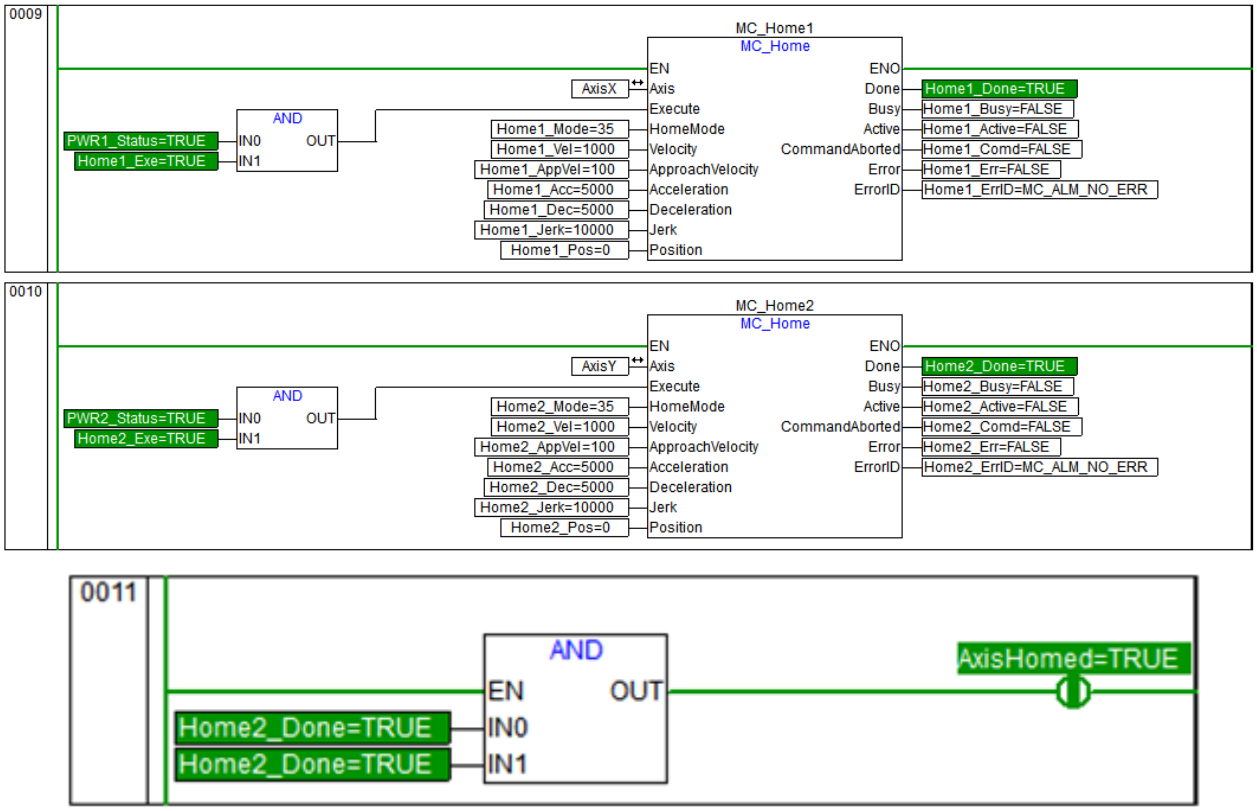
（3） 轴使能。轴组中各轴准备就绪（StartProg =TRUE）后，调用 MC_Power 对轴组的合轴和两个分轴进行轴使能。见 0004 ~ 0005 节。



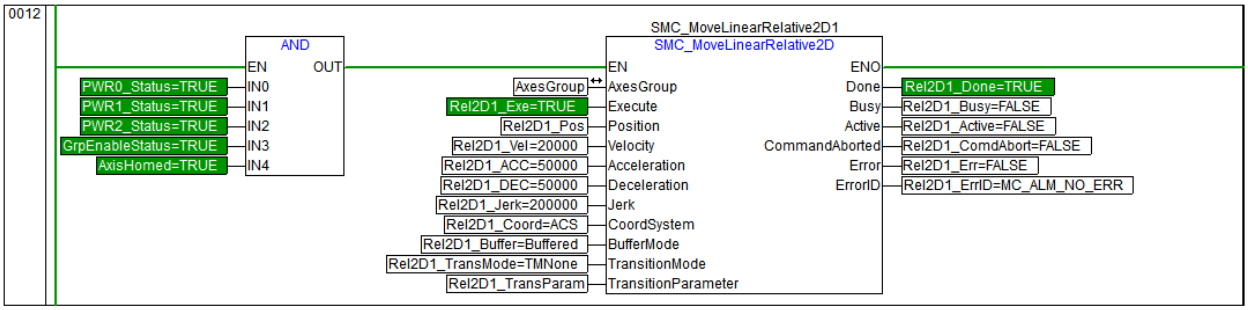
（4） 轴组使能。调用 MC_GroupEnable 对轴组使能。轴组变量中的 AxesGroupDisabled 为 FALSE，轴组使能成功。见 0006 节。轴组使能成功后，将 AxesGroupDisabled 取反赋给 GrpEnableStatus 变量。

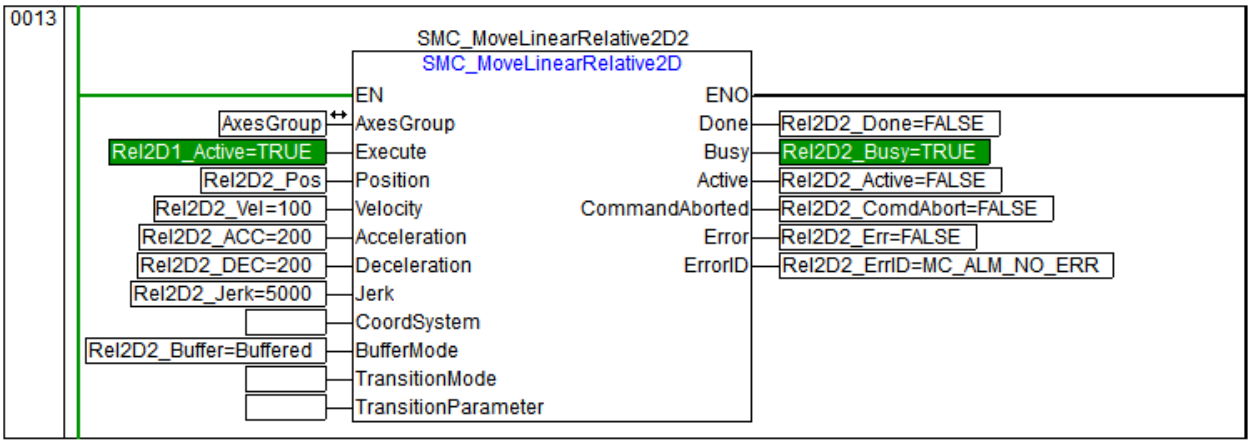


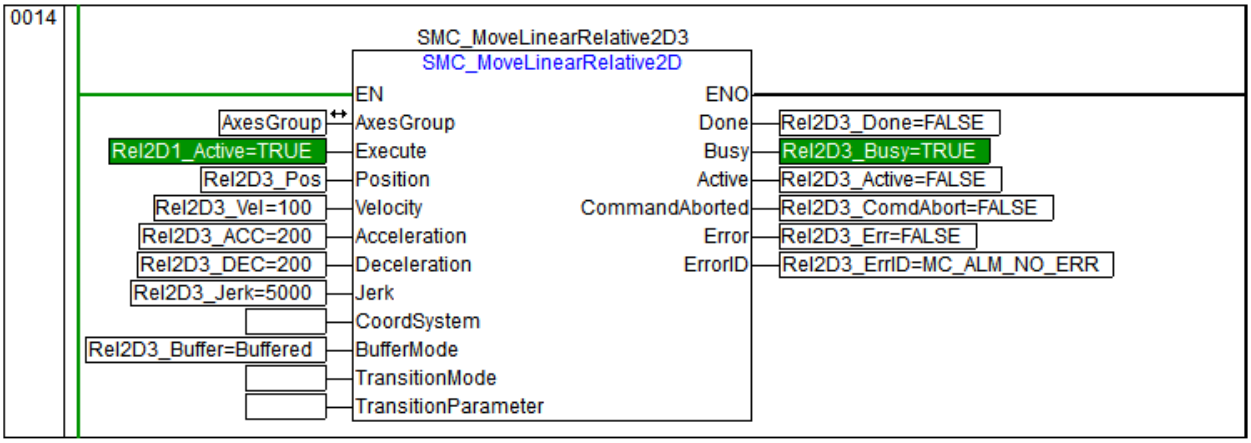
（5） 轴组中两个分轴使能完成后，若原点未确定，这执行回零操作。回零完成后，回零完成变量 AxisHomed 置 TRUE。见 0009-0011 节。

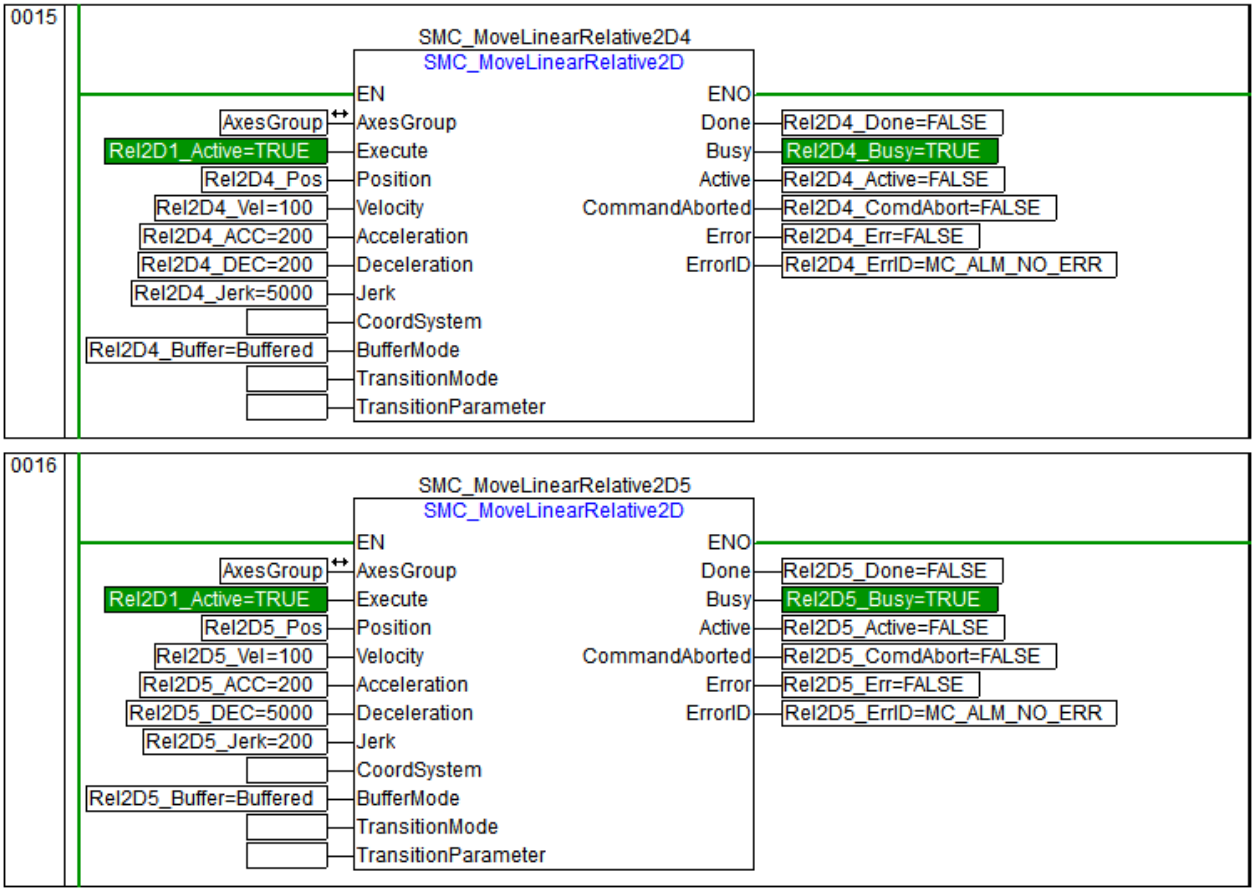


（6） 0011 节中，轴回原点完成后，开始执行插补运动指令。执行指令 1，当指令 1 的 Rel2D1_Active 为 TRUE 时，依次投送指令 2-5。见下述 0012-0016 节。


0012
AND
EN
OUT
PWR0_Status=TRUE
PWR1_Status=TRUE
PWR2_Status=TRUE
GrpEnableStatus=TRUE
AxisHomed=TRUE
IN0
IN1
IN2
IN3
IN4
SMC_MoveLinearRelative2D1
SMC_MoveLinearRelative2D
AxesGroup
Rel2D1_Exe=TRUE
Rel2D1_Pos
Rel2D1_Vel=20000
Rel2D1_ACC=50000
Rel2D1_DEC=50000
Rel2D1_Jerk=200000
Rel2D1_Coord=ACS
Rel2D1_Buffer=Buffered
Rel2D1_TransMode=TMNone
Rel2D1_TransParam
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
Rel2D1_Done=TRUE
Rel2D1_Busy=FALSE
Rel2D1_Active=FALSE
Rel2D1_ComdAbort=FALSE
Rel2D1_Err=FALSE
Rel2D1_ErrID=MC_ALM_NO_ERR


0013
SMC_MoveLinearRelative2D2
SMC_MoveLinearRelative2D
AxesGroup
Rel2D1_Active=TRUE
Rel2D2_Pos
Rel2D2_Vel=100
Rel2D2_ACC=200
Rel2D2_DEC=200
Rel2D2_Jerk=5000
Rel2D2_Buffer=Buffered
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
Rel2D2_Done=FALSE
Rel2D2_Busy=TRUE
Rel2D2_Active=FALSE
Rel2D2_ComdAbort=FALSE
Rel2D2_Err=FALSE
Rel2D2_ErrID=MC_ALM_NO_ERR


0014
SMC_MoveLinearRelative2D3
SMC_MoveLinearRelative2D
AxesGroup
Rel2D1_Active=TRUE
Rel2D3_Pos
Rel2D3_Vel=100
Rel2D3_ACC=200
Rel2D3_DEC=200
Rel2D3_Jerk=5000
Rel2D3_Buffer=Buffered
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
Rel2D3_Done=FALSE
Rel2D3_Busy=TRUE
Rel2D3_Active=FALSE
Rel2D3_ComdAbort=FALSE
Rel2D3_Err=FALSE
Rel2D3_ErrID=MC_ALM_NO_ERR



■ ST 语言程序

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power0_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
Power1_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
GrpEnable1_Exe	BOOL	FALSE	轴组使能上升沿触发变量。
AxisHomed	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴回原点完成。
InitFlag	BOOL	FALSE	如果该变量为 TRUE，则初始化参数设置完成。
Rel2D1_Exe	BOOL	FALSE	轴组投送的相对插补指令的上升沿触发变量。

Rel2D1_Active	BOOL	FALSE	轴组投送的相对插补指令的 Active 引脚的变量，该变量变为 TRUE 时，轴组执行插补指令。

（1） 设定轴组中合轴和分轴的轴 ID。设定所定义的合轴 AxisI、分轴 AxisX 和 AxisY 的轴 ID。

```
IF Switch1 THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

（2） 设置指令运动参数

```
IF FALSE = InitFlag THEN
   (*相对位置直线插补指令 1 的参数设置。未设置的输入参数采用默认值。

   Rel2D1_Pos[0]:=300;
   Rel2D1_Pos[1]:=400;
   Rel2D1_Vel:=100;
   Rel2D1_ACC:=200;
   Rel2D1_DEC:=200;
   Rel2D1_Jerk:=5000;
   Rel2D1_Buffer:=Buffered;
   (*相对位置直线插补指令 2 的参数设置。未设置的输入参数采用默认值。

   Rel2D2_Pos[0]:= 0;
   Rel2D2_Pos[1]:=400;
   Rel2D2_Vel:=100;
   Rel2D2_ACC:=200;
   Rel2D2_DEC:=200;
   Rel2D2_Jerk:=5000;
   Rel2D2_Buffer:=Buffered;
   (*相对位置直线插补指令 3 的参数设置。未设置的输入参数采用默认值。

   Rel2D3_Pos[0]:=300;
   Rel2D3_Pos[1]:=400;
   Rel2D3_Vel:=100;
   Rel2D3_ACC:=200;
   Rel2D3_DEC:=200;
```

```
    Rel2D3_Jerk:=5000;
    Rel2D3_Buffer:=Buffered;
    (*相对位置直线插补指令 4 的参数设置。未设置的输入参数采用默认值。
    Rel2D4_Pos[0]:= 0;
    Rel2D4_Pos[1]:=400;
    Rel2D4_Vel:=100;
    Rel2D4_ACC:=200;
    Rel2D4_DEC:=200;
    Rel2D4_Jerk:=5000;
    Rel2D4_Buffer:=Buffered;
    (*相对位置直线插补指令 5 的参数设置。未设置的输入参数采用默认值。
    Rel2D5_Pos[0]:=300;
    Rel2D5_Pos[1]:=400;
    Rel2D5_Vel:=100;
    Rel2D5_ACC:=200;
    Rel2D5_DEC:=200;
    Rel2D5_Jerk:=5000;
    Rel2D5_Buffer:=Buffered;
    InitFlag:=TRUE;
END_IF
```

（3） 调用 SMC_AxisInit 指令完成轴组中三个轴的轴初始化配置。轴组合轴轴类型设置为 0：Virtual；轴组分轴 AxisX/AxisY 轴类型均设置为 50：EtherCAT_Position。SMC_AxisInit 指令使用方法参见其指令章节介绍。

（4） 执行 MC_Power 指令完成轴组中三个轴的轴使能。

```
IF StartPro THEN
    MC_Power0(                          (*合轴使能*)
    Enable := PWR0_Enable,
    Axis := AxisI,
    Status=> PWR0_Status,
    Busy=> PWR0_Busy,
Error=> PWR0_Err,
ErrorID=> PWR0_ErrID);
MC_Power1(                          (*分轴 AxisX 使能*)
    Enable := PWR1_Enable,
```

```
  Axis := AxisX,
  Status=> PWR1_Status,
  Busy=> PWR1_Busy,
Error=> PWR1_Err,
ErrorID=> PWR1_ErrID);
MC_Power2(                              (*分轴 AxisY 使能*)
  Enable := PWR2_Enable,
  Axis := AxisY,
  Status=> PWR2_Status,
  Busy=> PWR2_Busy,
  Error=> PWR2_Err,
  ErrorID=> PWR2_ErrID);
END_IF
```

（5） 执行轴组使能。

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

（6） 执行轴组中分轴的原点回归指令，确定原点，MC_Home 指令使用方法见其介绍章节。

```
IF  PWR1_Status THEN
  MC_Home1(
  Execute := Home1_Exe,
  HomeMode := Home1_Mode,
  Velocity := Home1_Vel,
  ApproachVelocity := Home1_AppVel,
  Acceleration := Home1_Acc,
  Deceleration := Home1_Dec,
  Jerk := Home1_Jerk,
  Position := Home1_Pos,
  Axis := AxisX,
  Done=> Home1_Done,
  Busy=> Home1_Busy,
```

```
    Active=> Home1_Active,
    CommandAborted=> Home1_Comd,
    Error=> Home1_Err,
    ErrorID=> Home1_ErrID);
    MC_Home2(
    Execute := Home2_Exe,
    HomeMode := Home2_Mode,
    Velocity := Home2_Vel,
    ApproachVelocity := Home2_AppVel,
    Acceleration := Home2_Acc,
    Deceleration := Home2_Dec,
    Jerk := Home2_Jerk,
    Position := Home2_Pos,
    Axis := AxisY,
    Done=> Home2_Done,
    Busy=> Home2_Busy,
    Active=> Home2_Active,
    CommandAborted=> Home2_Comd,
    Error=> Home2_Err,
    ErrorID=> Home2_ErrID);
END_IF
IF  Home1_Done  AND Home2_Done THEN
    AxisHomed:=TRUE;
END_IF
```

（7） 轴回原点完成后，开始执行插补运动指令。执行指令 1，当指令 1 的 Rel2D1_Active 为 TRUE 时，依次投送指令 2-5。

```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus AND AxisHomed THEN
    SMC_MoveLinearRelative2D1(          (*插补指令 1*)
    Execute := Rel2D1_Exe,
    Position := Rel2D1_Pos,
    Velocity := Rel2D1_Vel,
    Acceleration := Rel2D1_ACC,
    Deceleration := Rel2D1_DEC,
    Jerk := Rel2D1_Jerk,
    CoordSystem := Rel2D1_Coord,
    BufferMode := Rel2D1_Buffer,
```

```
TransitionMode := Rel2D1_Trans,
TransitionParameter := Rel2D1_Param,
AxesGroup := AxesGroup,
Done=> Rel2D1_Done,
Busy=> Rel2D1_Busy,
Active=> Rel2D1_Active,
CommandAborted=> Rel2D1_ComdAbort,
Error=> Rel2D1_Err,
ErrorID=> Rel2D1_ErrID);
SMC_MoveLinearRelative2D2(                (*插补指令 2*)
Execute := Rel2D1_Active,
Position := Rel2D2_Pos,
Velocity := Rel2D2_Vel,
Acceleration := Rel2D2_ACC,
Deceleration := Rel2D2_DEC,
Jerk := Rel2D2_Jerk,
CoordSystem := Rel2D2_Coord,
BufferMode := Rel2D2_Buffer,
TransitionMode := Rel2D2_Trans,
TransitionParameter := Rel2D2_Param,
AxesGroup := AxesGroup,
Done=> Rel2D2_Done,
Busy=> Rel2D2_Busy,
Active=> Rel2D2_Active,
CommandAborted=> Rel2D2_ComdAbort,
Error=> Rel2D2_Err,
ErrorID=> Rel2D2_ErrID);
SMC_MoveLinearRelative2D3(            (*插补指令 3*)
Execute := Rel2D1_Active,
Position := Rel2D3_Pos,
Velocity := Rel2D3_Vel,
Acceleration := Rel2D3_ACC,
Deceleration := Rel2D3_DEC,
Jerk := Rel2D3_Jerk,
CoordSystem := Rel2D3_Coord,
BufferMode := Rel2D3_Buffer,
TransitionMode := Rel2D3_Trans,
```

```
TransitionParameter := Rel2D3_Param,
AxesGroup := AxesGroup,
Done=> Rel2D3_Done,
Busy=> Rel2D3_Busy,
Active=> Rel2D3_Active,
CommandAborted=> Rel2D3_ComdAbort,
Error=> Rel2D3_Err,
ErrorID=> Rel2D3_ErrID);
SMC_MoveLinearRelative2D4(                (*插补指令 4*)
Execute := Rel2D1_Active,
Position := Rel2D4_Pos,
Velocity := Rel2D4_Vel,
Acceleration := Rel2D4_ACC,
Deceleration := Rel2D4_DEC,
Jerk := Rel2D4_Jerk,
CoordSystem := Rel2D4_Coord,
BufferMode := Rel2D4_Buffer,
TransitionMode := Rel2D4_Trans,
TransitionParameter := Rel2D4_Param,
AxesGroup := AxesGroup,
Done=> Rel2D4_Done,
Busy=> Rel2D4_Busy,
Active=> Rel2D4_Active,
CommandAborted=> Rel2D4_ComdAbort,
Error=> Rel2D4_Err,
ErrorID=> Rel2D4_ErrID);
SMC_MoveLinearRelative2D5(                (*插补指令 5*)
Execute := Rel2D1_Active,
Position := Rel2D5_Pos,
Velocity := Rel2D5_Vel,
Acceleration := Rel2D5_ACC,
Deceleration := Rel2D5_DEC,
Jerk := Rel2D5_Jerk,
CoordSystem := Rel2D5_Coord,
BufferMode := Rel2D5_Buffer,
TransitionMode := Rel2D5_Trans,
TransitionParameter := Rel2D5_Param,
```

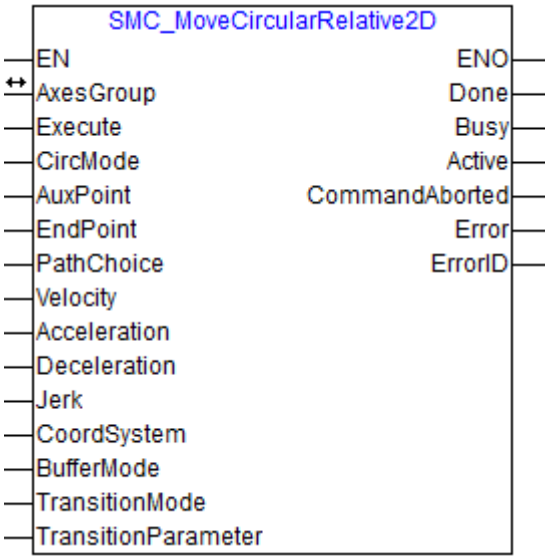
```
    AxesGroup := AxesGroup,
    Done=> Rel2D5_Done,
    Busy=> Rel2D5_Busy,
    Active=> Rel2D5_Active,
    CommandAborted=> Rel2D5_ComdAbort,
    Error=> Rel2D5_Err,
    ErrorID=> Rel2D5_ErrID);
END_IF
```

7.5.4 使用注意事项

- 调用该指令时，输入参数 Positon 不可设为 0。
- 本指令 BufferMode (缓存模式选择)不支持 0：Aborting。

7.6 SMC_MoveCircularRelative2D（2 轴相对位置圆弧插补）

该指令实现 2 轴相对位置圆弧插补功能。



7.6.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	AxesGroup	轴组名	否	-	-	AXES_GROUP_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	CircMode	模式选择 0：三点 1：圆心/终点	是	0	0：BORDER 1：CENTER	MC_CIRC_MODE
	AuxPoint	CircMode 为 BORDER 时，AuxPoint 为中间点位置； CircMode 为 CENTER 时，AuxPoint 为圆心点位置。	否	-	正值/负值/0	ARRAY[0..1] OF LREAL
	EndPoint	终点位置	否	-	正值/负值/0	ARRAY[0..1] OF LREAL
	PathChoice	运动方向（CircMode 为 CENTER 时有效） 0：顺时针 1：逆时针	是	0	0：CLOCKWISE 1：COUNTERCLOCKWISE	MC_CIRC_MODE
	Velocity	目标速度	否	-	非负	LREAL
	Acceleration	加速度	否	-	正值	LREAL

	Deceleration	减速度	否	-	正值	LREAL
	Jerk	加加速度 0：梯形加减速 正值：S 形加减速	是	0	0/正值	LREAL
	CoordSystem	坐标系 0：轴坐标系	是	0	0: ACS	MC_COORD_SYSTEM
	BufferMode	缓冲模式 1：缓存 2：以相邻指令间的低速融合 3：以上一条指令的速度融合 4：以下一条指令的速度融合 5：以相邻指令间的高速融合	是	1	1: Buffered 2: BlendingLow 3: BlendingPrevious 4: BlendingNext 5: BlendingHigh	MC_BUFFER_MODE
	TransitionMode	过渡曲线选择 0：不插入过渡曲线	是	0	0：TMNone	MC_TRANSITION_MODE
	TransitionParameter	保留参数	是	0	-	ARRAY[0..1] OF LREAL
OUTPUT	Done	插补运动完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Active	控制中，指令被执行时为 TRUE	是	FALSE	TRUE/FALSE	BOOL
	CommandAborted	终止执行	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

7.6.2 功能

该指令实现 2 轴相对位置圆弧插补功能。

■ 插补步骤

（1） 确定插补的轴组。轴组 AxesGroup 变量类型为 AXES_GROUP_REF。

（2） 指定轴组变量中的轴号，对应于合轴和分轴的轴号。

（3） 完成各轴的使能以及轴组使能。才可使用圆弧插补指令。

- 基本功能

  - 圆弧指令位置

    以 2 维平面执行 2 轴的圆弧插补，插补合轴的目标位置在指令开始执行时刻位置基础上，增加指令轨迹圆弧长度。

    圆弧插补指令指定的位置以“相对值定位”的指定，即圆弧插补指令的指令位置 AuxPoint 和 EndPoint 以指令开始执行时刻位置为原点，进行增量计算。

  - 入参锁存

    本指令上升沿触发有效。在 Execute 输入的上升沿时，将输入输出参数 AxesGroup 的合轴以及两个分轴的 ID、输入参数 Position、Velocity、Acceleration、Deceleration、BufferMode 等参数值锁存，指令执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令。

  - 运动参数设定

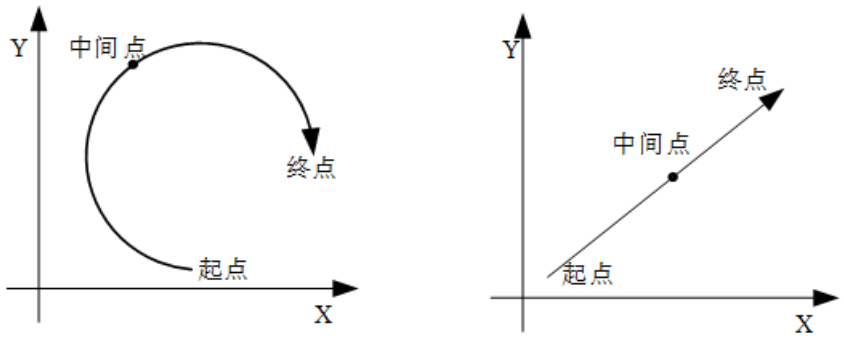
    圆弧插补的插补速度、加速度、减速度、加加速度通过指令输入参数 Velocity、Acceleration、Deceleration、Jerk 指定。

  - 圆弧插补模式（CircMode）

    圆弧插补方式有三点模式和圆心/终点模式 2 种，通过 CircMode(圆弧插补模式)指定。当 CircMode 为 BORDER 时，AuxPoint 为中间点位置；CircMode 为 CENTER 时，AuxPoint 为圆心点位置。

    （a） 三点模式（CircMode=BORDER）

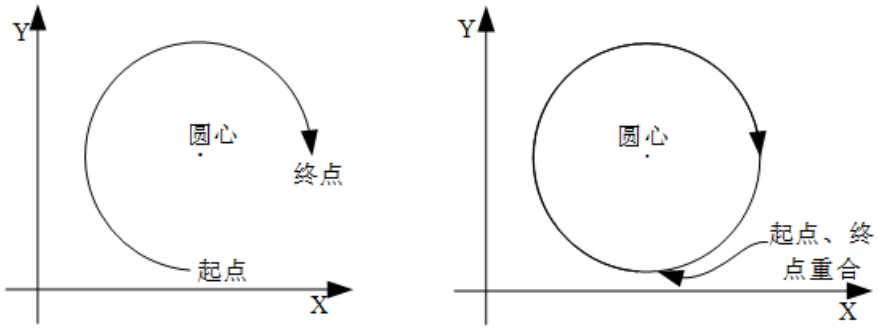
    以当前位置为起点，执行连接中间点位置 AuxPoint(X, Y)和终点 EndPoint(X, Y)的圆弧插补。起点、中间点位置与终点在同一条直线上、或者中间点位置与终点为同一点、以及起点和中间点位置为同一点时，执行从起点到终点的直线插补。



（b） 圆心/终点模式（CircMode=CENTER）

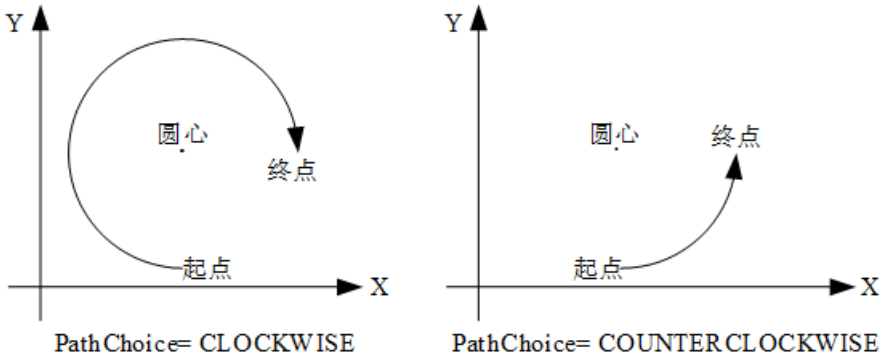
以当前位置为起点，执行以通过圆心位置 AuxPoint(X, Y) 指定的圆弧连接终点 EndPoint(X, Y) 的圆弧插补。圆弧的旋转方向通过 PathChoice(路径选择)指定。起点、终点为同一点时，绘制整圆。

指定的中心位置到起点的半径与到终点的半径不同时，使用指定的中心位置到起点的半径进行圆弧插补。



■ 圆弧插补运动方向（PathChoice）

该功能在 CircMode 为 CENTER 时有效，运动方向 PathChoice 为 0：顺时针（CLOCKWISE）和 1：逆时针（COUNTERCLOCKWISE）。如下图为两条插补圆弧的 AuxPoint 和 EndPoint 一致，圆弧插补运动方向相反的插补轨迹。



- 功能块重复触发的处理

功能块指令等待重复触发时，重复触发的指令被放入运动缓存中，功能块监控的状态为后一条指令的运行状态，前一条指令的运行状态则失去监控。在前一个插补指令执行期间，后一个插补指令等待前一个插补指令结束后，开始执行。

- BufferMode (缓冲模式选择)

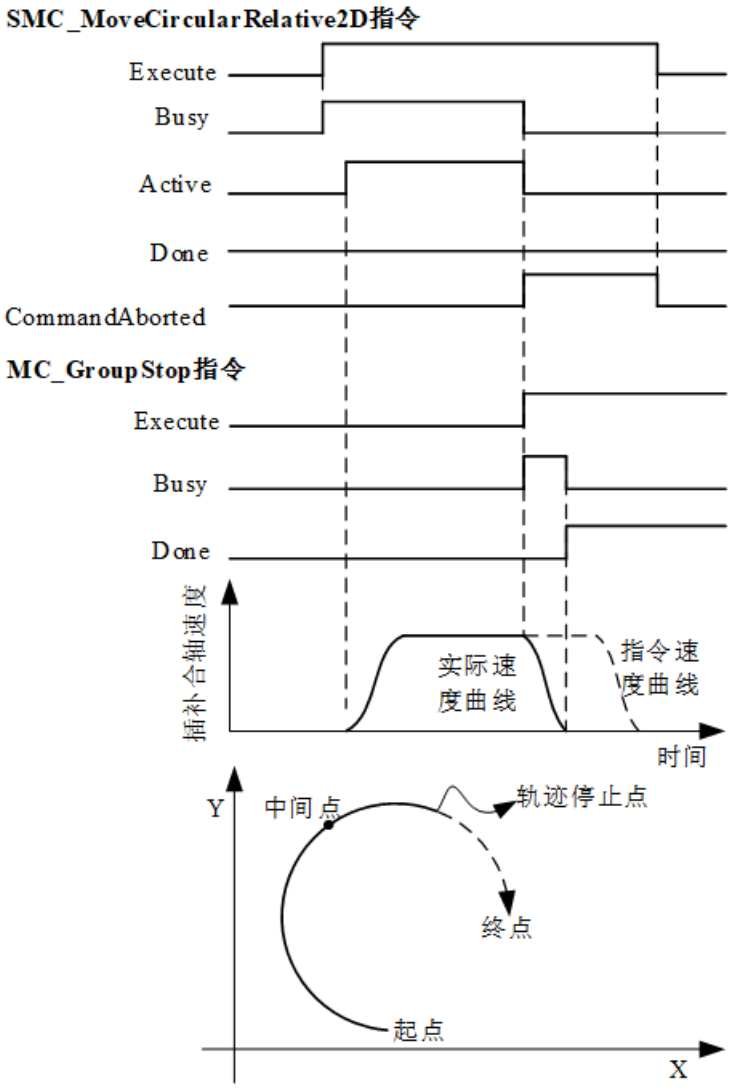
2 轴圆弧插补指令支持 BufferMode 支持 1：Buffered，2：BlendingLow、3: BlendingPrevious、4: BlendingNext 以及 5: BlendingHigh。

投送本指令时，若 BufferMode 为 1：Buffered，从当前执行中的指令正常结束的周期起，已缓存的本指令自动启动。若 BufferMode 为 2：BlendingLow，则以相邻指令间的低速融合。若 BufferMode 为 3: BlendingPrevious，则以上一条指令的速度融合。若 BufferMode 为 4: BlendingNext，则以下一条指令的速度融合。若 BufferMode 为 5: BlendingHigh，则以相邻指令间的高速融合。

插补指令的缓冲模式 BufferMode 详情请参考附录章节的缓冲模式说明。

- 指令中断

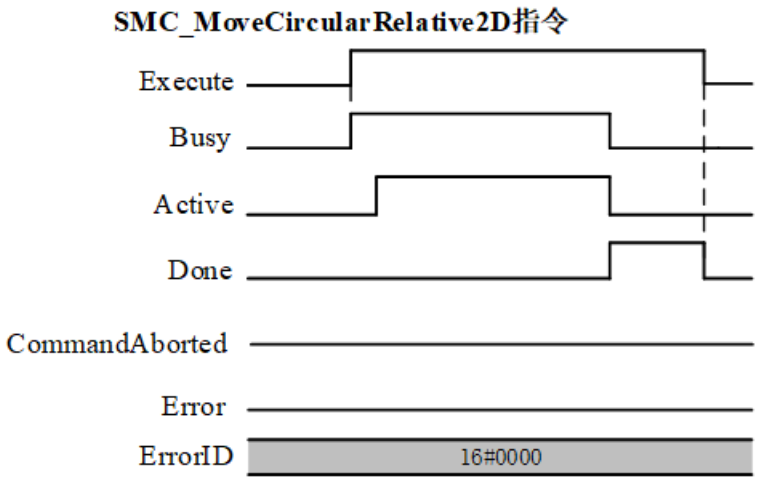
执行轴组圆弧插补指令过程中，可以使用 MC_GroupStop 指令中断插补指令。轴组各轴沿着圆弧插补运动轨迹减速停止。



- CoordSystem(坐标系)
  - 指定进行圆弧插补的坐标系。
  - 使用轴坐标系(ACS)。
- 时序图

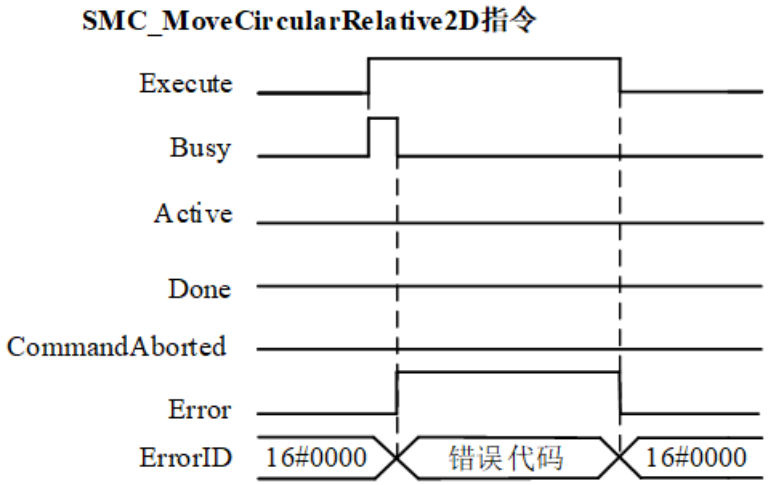
下述为圆弧插补指令执行时的正常、异常时序图。

(1) 正常时序图



(2) 异常错误发生时引脚时序

执行本指令时，若发生参数非法、轴组中有轴为轴类型为 0：Unused（未使用轴），轴组中各轴 ID 非法等异常，调用本指令失败，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Execute 变为低电平时，所有输出引脚恢复为默认值。



- 当指令的异常情况解决后，需要重新上升沿触发 Execute，投送轴组插补指令，进行轴组绝对位置插补运动。

7.6.3 示例

■ 动作示例

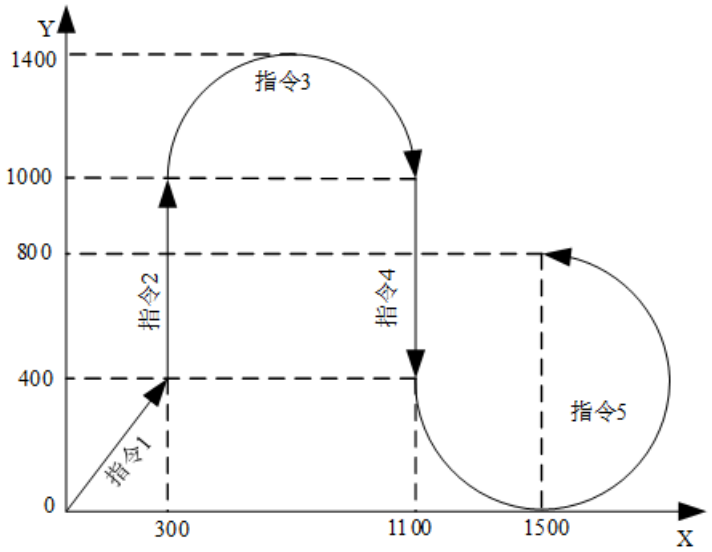
■ 动作介绍

以下示例包含直线插补指令和圆弧插补指令，执行完插补指令后将移动到目标位置。

设定插补指令的缓存模式 BufferMode 选择指定为缓存 Buffered，缓存多个运动指令。

在本例中，直线插补指令 1 的输出 Active 变为 TRUE 时，执行其余插补指令 2-5。

■ 动作示意图



依次执行直线插补指令 1 到达（300,400）位置处，执行直线插补指令 2 到达（300,1000）位置处，执行圆弧插补指令 3 到达（1100,1000）位置处，执行直线插补指令 4 到达（1100,400）位置处，执行圆弧插补指令 5 到达（1500,800）位置处。

■ 梯形图程序

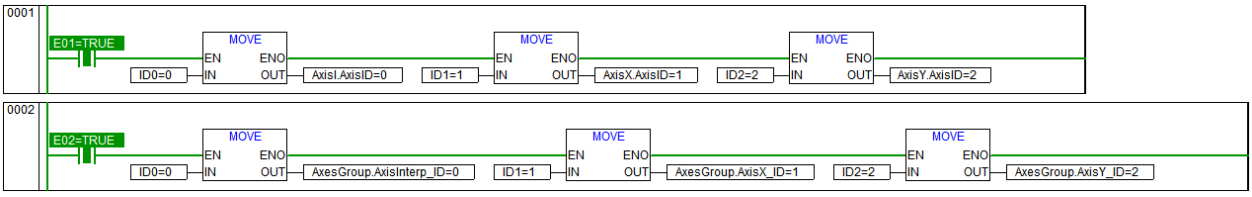
设定各插补指令速度为 100，加减速度均为 200、加加速度为 5000，缓存模式为 Buffered。

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。

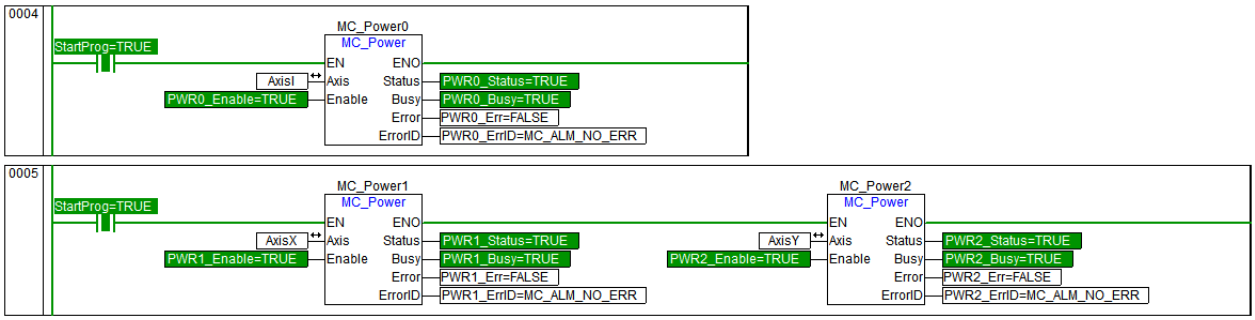
Power1_Status	BOOL	FALSE	轴组中插补分轴 X 轴使能成功的变量。X 轴使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	轴组中插补分轴 Y 轴使能成功的变量。Y 轴使能成功时，该变量变为 TRUE。
Power0_Status	BOOL	FALSE	轴组中插补合轴使能成功的变量。插补合轴使能成功时，该变量变为 TRUE。
GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
GrpEnable1_Exe	BOOL	FALSE	轴组使能上升沿触发变量。
AxisHomed	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴回原点完成。
Rel2D1_Exe	BOOL	FALSE	轴组投送的圆弧插补指令的上升沿触发变量。
Rel2D1_Active	BOOL	FALSE	轴组投送的圆弧插补指令的 Active 引脚的变量，该变量变为 TRUE 时，轴组执行插补指令。

（1） 设定轴组中合轴和分轴的轴 ID。定义轴组时，首先需要定义轴组中的三个轴，将定义的三个轴的轴号 ID 与轴组中合轴和分轴的 ID 对应即可。见下述 0001 节和 0002 节。

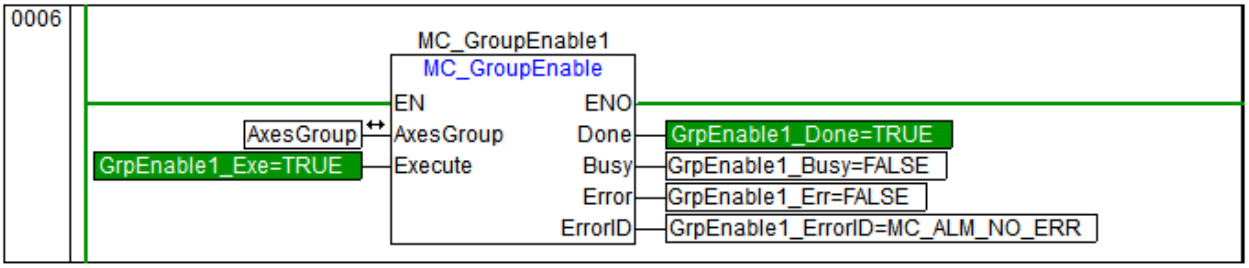


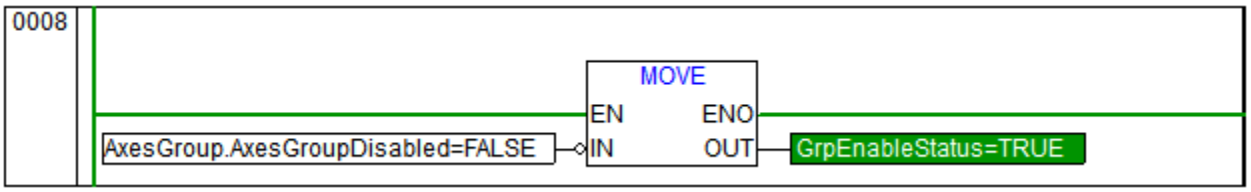
（2） 轴初始化。调用 SMC_AxisInit 指令分别完成轴组中三个轴的轴初始化配置，设置合轴为 0：Virtual，分轴类型为 50: EtherCAT_Position。SMC_AxisInit 指令使用方法参见相关章节介绍。

（3） 轴使能。轴组中各轴准备就绪（StartProg =TRUE）后，调用 MC_Power 对轴组的合轴和两个分轴进行轴使能。见 0004 ~ 0005 节。

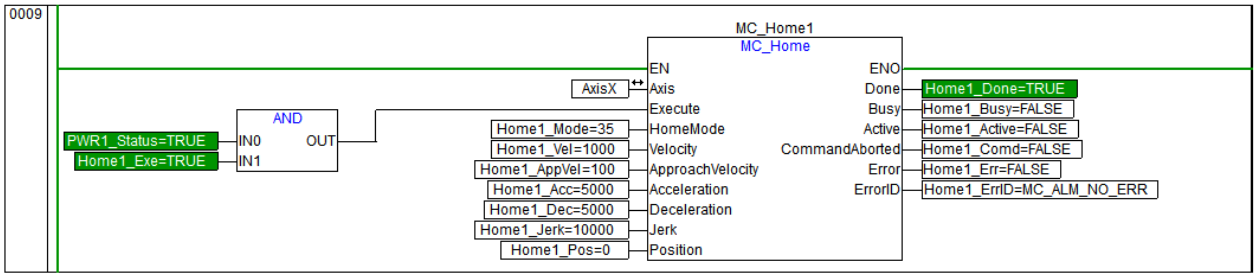


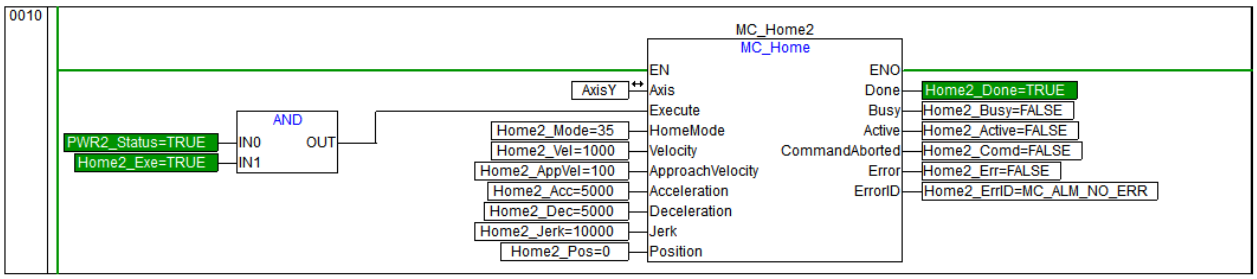
（4） 轴组使能。调用 MC_GroupEnable 对轴组使能。轴组变量中的 AxesGroupDisabled 为 FALSE，轴组使能成功，见 0006 节。轴组使能成功后，将 AxesGroupDisabled 取反赋给 GrpEnableStatus 变量，见 0008 节。

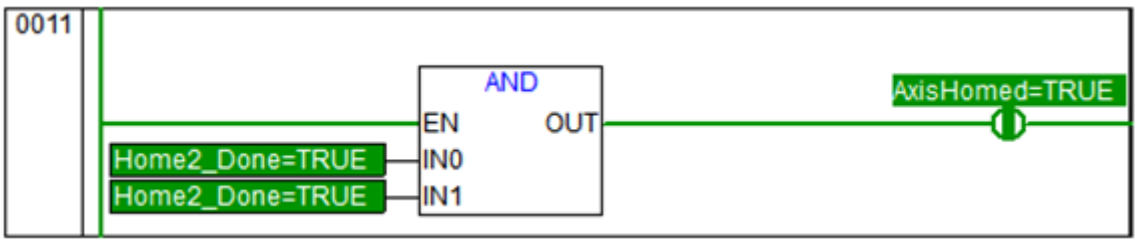




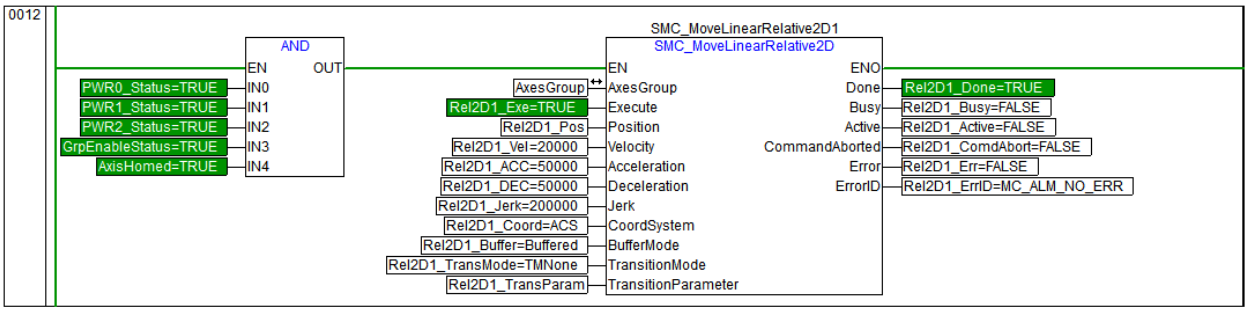
（5） 轴组中两个分轴使能完成后，若原点未确定，这执行回零操作。回零完成后，回零完成变量 AxisHomed 置 TRUE。见 0009-0011 节。

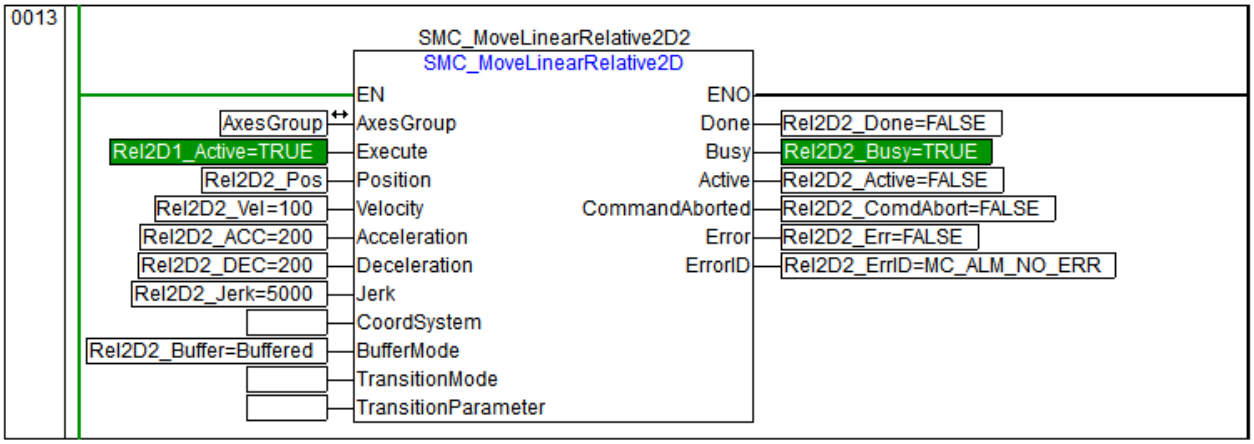


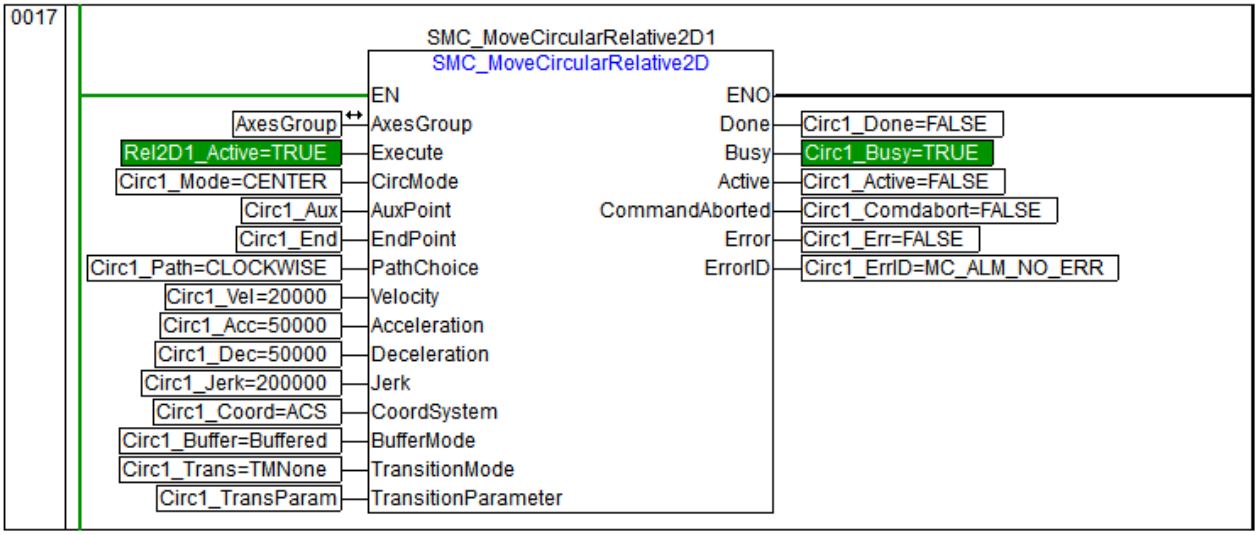


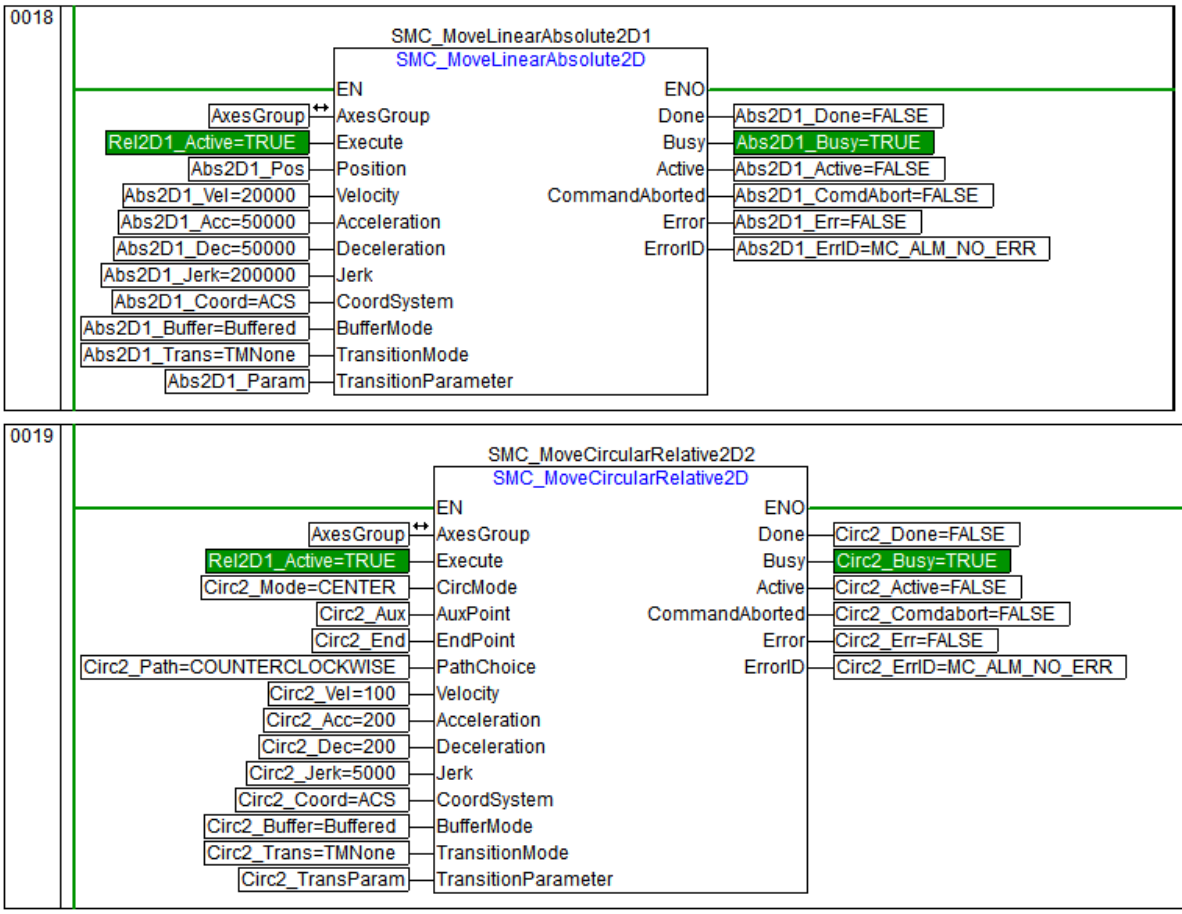


（6） 0011 节中，轴回原点完成后，开始执行插补运动指令。执行指令 1，当指令 1 的 Rel2D1_Active 为 TRUE 时，依次投送指令 2-5。见下述 0012-0013/0017-0019 节。


0012
AND
EN
OUT
PWR0_Status=TRUE
PWR1_Status=TRUE
PWR2_Status=TRUE
GrpEnableStatus=TRUE
AxisHomed=TRUE
IN0
IN1
IN2
IN3
IN4
SMC_MoveLinearRelative2D1
SMC_MoveLinearRelative2D
AxesGroup
Rel2D1_Exe=TRUE
Rel2D1_Pos
Rel2D1_Vel=20000
Rel2D1_ACC=50000
Rel2D1_DEC=50000
Rel2D1_Jerk=200000
Rel2D1_Coord=ACS
Rel2D1_Buffer=Buffered
Rel2D1_TransMode=TMNone
Rel2D1_TransParam
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
Rel2D1_Done=TRUE
Rel2D1_Busy=FALSE
Rel2D1_Active=FALSE
Rel2D1_ComdAbort=FALSE
Rel2D1_Err=FALSE
Rel2D1_ErrID=MC_ALM_NO_ERR


0013
SMC_MoveLinearRelative2D2
SMC_MoveLinearRelative2D
AxesGroup
Rel2D1_Active=TRUE
Rel2D2_Pos
Rel2D2_Vel=100
Rel2D2_ACC=200
Rel2D2_DEC=200
Rel2D2_Jerk=5000
Rel2D2_Buffer=Buffered
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
Rel2D2_Done=FALSE
Rel2D2_Busy=TRUE
Rel2D2_Active=FALSE
Rel2D2_ComdAbort=FALSE
Rel2D2_Err=FALSE
Rel2D2_ErrID=MC_ALM_NO_ERR


0017
SMC_MoveCircularRelative2D1
SMC_MoveCircularRelative2D
AxesGroup
Rel2D1_Active=TRUE
Circ1_Mode=CENTER
Circ1_Aux
Circ1_End
Circ1_Path=CLOCKWISE
Circ1_Vel=20000
Circ1_Acc=50000
Circ1_Dec=50000
Circ1_Jerk=200000
Circ1_Coord=ACS
Circ1_Buffer=Buffered
Circ1_Trans=TMNone
Circ1_TransParam
EN
AxesGroup
Execute
CircMode
AuxPoint
EndPoint
PathChoice
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
Circ1_Done=FALSE
Circ1_Busy=TRUE
Circ1_Active=FALSE
Circ1_Comdabort=FALSE
Circ1_Err=FALSE
Circ1_ErrID=MC_ALM_NO_ERR



■ ST 语言程序

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power1_Status	BOOL	FALSE	轴组中插补分轴 X 轴使能成功的变量。X 轴使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	轴组中插补分轴 Y 轴使能成功的变量。Y 轴使能成功时，该变量变为 TRUE。
Power0_Status	BOOL	FALSE	轴组中插补合轴使能成功的变量。插补合轴使能成功时，该变量变为 TRUE。

GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
GrpEnable1_Exe	BOOL	FALSE	轴组使能上升沿触发变量。
AxisHomed	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴回原点完成。
InitFlag	BOOL	FALSE	如果该变量为 TRUE，则初始化参数设置完成。
Rel2D1_Exe	BOOL	FALSE	轴组投送的插补指令 1 的上升沿触发变量。
Rel2D1_Active	BOOL	FALSE	轴组投送的插补指令 1 的 Active 引脚的变量，该变量变为 TRUE 时，轴组执行插补指令。

（1） 设定轴组中合轴和分轴的轴 ID。设定所定义的合轴 AxisI、分轴 AxisX 和 AxisY 的轴 ID。

```
IF Switch1 THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

（2） 设置指令运动参数

```
IF FALSE = InitFlag THEN
   (*相对位置直线插补指令 1 的参数设置。未设置的输入参数采用默认值*)

   Rel2D1_Pos[0]:=300;
   Rel2D1_Pos[1]:=400;
   Rel2D1_Vel:=100;
   Rel2D1_ACC:=200;
   Rel2D1_DEC:=200;
   Rel2D1_Jerk:=5000;
   Rel2D1_Buffer:=Buffered;
   (*相对位置直线插补指令 2 的参数设置。未设置的输入参数采用默认值*)

   Rel2D2_Pos[0]:= 0;
   Rel2D2_Pos[1]:=400;
   Rel2D2_Vel:=100;
   Rel2D2_ACC:=200;
   Rel2D2_DEC:=200;
   Rel2D2_Jerk:=5000;
   Rel2D2_Buffer:=Buffered;
   (*圆弧插补指令 3 的参数设置。未设置的输入参数采用默认值*)
```

```
    Circ1_Mode:= BORDER;
    Circ1_Aux [0] :=400;
    Circ1_Aux [1] :=400;
    Circ1_End [0] :=800;
    Circ1_End [1] :=0;
    Circ1_Path:= CLOCKWISE;
    Circ1_Vel:=100;
    Circ1_ACC:=200;
    Circ1_DEC:=200;
    Circ1_Jerk:=5000;
    Circ1_Buffer:=Buffered;
    (*绝对位置直线插补指令 4 的参数设置。未设置的输入参数采用默认值*)
    Abs2D1_Pos[0]:= 1100;
    Abs2D1_Pos[1]:=400;
    Abs2D1_Vel:=100;
    Abs2D1_ACC:=200;
    Abs2D1_DEC:=200;
    Abs2D1_Jerk:=5000;
    AbsD1_Buffer:=Buffered;
    (*圆弧插补指令 5 的参数设置。未设置的输入参数采用默认值*)
    Circ1_Mode:= CENTER;
    Circ1_Aux [0] :=400;
    Circ1_Aux [1] := 0;
    Circ1_End [0] :=400;
    Circ1_End [1] :=400;
    Circ1_Path:= COUNTERCLOCKWISE;
    Circ1_Vel:=100;
    Circ1_ACC:=200;
    Circ1_DEC:=200;
    Circ1_Jerk:=5000;
    Circ1_Buffer:=Buffered;
    InitFlag:=TRUE;
END_IF
```

（3） 调用 SMC_AxisInit 指令完成轴组中三个轴的轴初始化配置。轴组合轴轴类型设置为 0：Virtual；轴组分轴 AxisX/AxisY 轴类型均设置为 50：EtherCAT_Position。SMC_AxisInit 指令使用方法参

见其指令章节介绍。

（4） 执行 MC_Power 指令完成轴组中三个轴的轴使能。

```
IF StartPro THEN
   MC_Power0(                          (*合轴使能*)
   Enable := PWR0_Enable,
   Axis := AxisI,
   Status=> PWR0_Status,
   Busy=> PWR0_Busy,
   Error=> PWR0_Err,
   ErrorID=> PWR0_ErrID);
   MC_Power1(                          (*分轴 AxisX 使能*)
   Enable := PWR1_Enable,
   Axis := AxisX,
   Status=> PWR1_Status,
   Busy=> PWR1_Busy,
   Error=> PWR1_Err,
   ErrorID=> PWR1_ErrID);
   MC_Power2(                          (*分轴 AxisY 使能*)
   Enable := PWR2_Enable,
   Axis := AxisY,
   Status=> PWR2_Status,
   Busy=> PWR2_Busy,
   Error=> PWR2_Err,
   ErrorID=> PWR2_ErrID);
END_IF
```

（5） 执行轴组使能

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

（6） 执行轴组中分轴的原点回归指令，确定原点，MC_Home 指令使用方法见其介绍章节

```
IF  PWR1_Status THEN
   MC_Home1(
   Execute := Home1_Exe,
   HomeMode := Home1_Mode,
   Velocity := Home1_Vel,
   ApproachVelocity := Home1_AppVel,
   Acceleration := Home1_Acc,
   Deceleration := Home1_Dec,
   Jerk := Home1_Jerk,
   Position := Home1_Pos,
   Axis := AxisX,
   Done=> Home1_Done,
   Busy=> Home1_Busy,
   Active=> Home1_Active,
   CommandAborted=> Home1_Comd,
   Error=> Home1_Err,
   ErrorID=> Home1_ErrID);
   MC_Home2(
   Execute := Home2_Exe,
   HomeMode := Home2_Mode,
   Velocity := Home2_Vel,
   ApproachVelocity := Home2_AppVel,
   Acceleration := Home2_Acc,
   Deceleration := Home2_Dec,
   Jerk := Home2_Jerk,
   Position := Home2_Pos,
   Axis := AxisY,
   Done=> Home2_Done,
   Busy=> Home2_Busy,
   Active=> Home2_Active,
   CommandAborted=> Home2_Comd,
   Error=> Home2_Err,
   ErrorID=> Home2_ErrID);
END_IF
IF  Home1_Done  AND Home2_Done THEN
   AxisHomed:=TRUE;
END_IF
```

（7） 轴回原点完成后，开始执行插补运动指令。执行指令 1，当指令 1 的 Rel2D1_Active 为 TRUE 时，依次投送指令 2-5。

```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus AND AxisHomed THEN
  SMC_MoveLinearRelative2D1(            (*插补指令 1*)
  Execute := Rel2D1_Exe,
  Position := Rel2D1_Pos,
  Velocity := Rel2D1_Vel,
  Acceleration := Rel2D1_ACC,
  Deceleration := Rel2D1_DEC,
  Jerk := Rel2D1_Jerk,
  CoordSystem := Rel2D1_Coord,
  BufferMode := Rel2D1_Buffer,
  TransitionMode := Rel2D1_Trans,
  TransitionParameter := Rel2D1_Param,
  AxesGroup := AxesGroup,
  Done=> Rel2D1_Done,
  Busy=> Rel2D1_Busy,
  Active=> Rel2D1_Active,
  CommandAborted=> Rel2D1_ComdAbort,
  Error=> Rel2D1_Err,
  ErrorID=> Rel2D1_ErrID);
  SMC_MoveLinearRelative2D2(              (*插补指令 2*)
  Execute := Rel2D1_Active,
  Position := Rel2D2_Pos,
  Velocity := Rel2D2_Vel,
  Acceleration := Rel2D2_ACC,
  Deceleration := Rel2D2_DEC,
  Jerk := Rel2D2_Jerk,
  CoordSystem := Rel2D2_Coord,
  BufferMode := Rel2D2_Buffer,
  TransitionMode := Rel2D2_Trans,
  TransitionParameter := Rel2D2_Param,
  AxesGroup := AxesGroup,
  Done=> Rel2D2_Done,
  Busy=> Rel2D2_Busy,
  Active=> Rel2D2_Active,
```

```
CommandAborted=> Rel2D2_ComdAbort,
Error=> Rel2D2_Err,
ErrorID=> Rel2D2_ErrID);
SMC_MoveCircularRelative2D1(                (*插补指令 3*)
Execute := Rel2D1_Active,
CircMode := Circ1_Mode ,
AuxPoint := Circ1_Aux,
EndPoint := Circ1_End,
PathChoice:= Circ1_Path,
Velocity := Circ1_Vel,
Acceleration := Circ1_ACC,
Deceleration := Circ1_DEC,
Jerk := Circ1_Jerk,
CoordSystem := Circ1_Coord,
BufferMode := Circ1_Buffer,
TransitionMode := Circ1_ Trans,
TransitionParameter := Circ1_Param,
AxesGroup := AxesGroup,
Done=> Circ1_Done,
Busy=> Circ1_Busy,
Active=> Circ1_Active,
CommandAborted=> Circ1_Abort,
Error=> Circ1_Err,
ErrorID=> Circ1_ErrID);
SMC_MoveLinearAbsolute2D1(          (*插补指令 4*)
Execute := Rel2D1_Active,
Position := Abs2D1_Pos,
Velocity := Abs2D1_Vel,
Acceleration := Abs2D1_ACC,
Deceleration := Abs2D1_DEC,
Jerk := Abs2D1_Jerk,
CoordSystem := Abs2D1_Coord,
BufferMode := Abs2D1_Buffer,
TransitionMode := Abs2D1_Trans,
TransitionParameter := Abs2D1_Param,
AxesGroup := AxesGroup,
Done=> Abs2D1_Done,
```

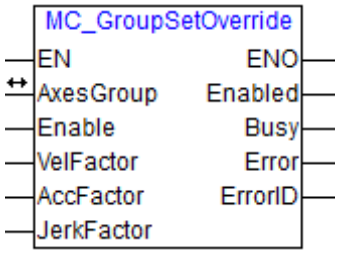
```
  Busy=> Abs2D1_Busy,
  Active=> Abs2D1_Active,
  CommandAborted=> Abs2D1_ComdAbort,
  Error=> Abs2D1_Err,
  ErrorID=> Abs2D1_ErrID);
  SMC_MoveCircularRelative2D2(              (*插补指令 5*)
  Execute := Rel2D1_Active,
  CircMode := Circ2_Mode ,
  AuxPoint := Circ2_Aux,
  EndPoint := Circ2_End,
  PathChoice:= Circ2_Path,
  Velocity := Circ2_Vel,
  Acceleration := Circ2_ACC,
  Deceleration := Circ2_DEC,
  Jerk := Circ2_Jerk,
  CoordSystem := Circ2_Coord,
  BufferMode := Circ2_Buffer,
  TransitionMode := Circ2_ Trans,
  TransitionParameter := Circ2_Param,
  AxesGroup := AxesGroup,
  Done=> Circ2_Done,
  Busy=> Circ2_Busy,
  Active=> Circ2_Active,
  CommandAborted=> Circ2_Abort,
  Error=> Circ2_Err,
  ErrorID=> Circ2_ErrID);
END_IF
```

7.6.4 使用注意事项

- 起点和终点为同一点时，会发生圆弧插补起点终点相同的异常。
- 本指令 BufferMode (缓存模式选择)不支持 0：Aborting。

7.7 MC_GroupSetOverride（轴组运动倍率设置（超驰））

该指令设定轴组中合轴速度、加减速、加加速度比例因子，可按设定比例放大或缩小合轴运行的速度、加减速、加加速度。



7.7.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	AxesGroup	轴组名	否	-	-	AXES_GROUP_REF
INPUT	Enable	使能	否	-	TRUE/FALSE	BOOL
	VelFactor	速度比例因子，单位为 1%	是	100	0 ~ 500	LREAL
	AccFactor	加减速度比例因子，单位为 1%	是	100	0 ~ 500（非零）	LREAL
	JerkFactor	加加速度比例因子，单位为 1%	是	100	0 ~ 500（非零）	LREAL
OUTPUT	Enabled	设定成功	否	-	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

7.7.2 功能

本指令可按照设定的比例因子放大或缩小轴组运行的速度、加减速、加加速度，该指令详情介绍如下：

（1） 该指令对轴组的目标速度进行超调，可以变更轴组运动过程中的目标速度，可以变更的指令如下所示：

- SMC_MoveLinearAbsolute2D
- SMC_MoveCircularRelative2D

- SMC_MoveLinearRelative2D

（2） 超调比例因子的单位是[%]。

- 变更后的目标速度＝ 当前执行的速度×速度比例因子；
- 变更后的加减速度 = 当前的加减速度×加减速度比例因子；
- 变更后的加加速度 = 当前的加加速度 ×加加速度比例因子。

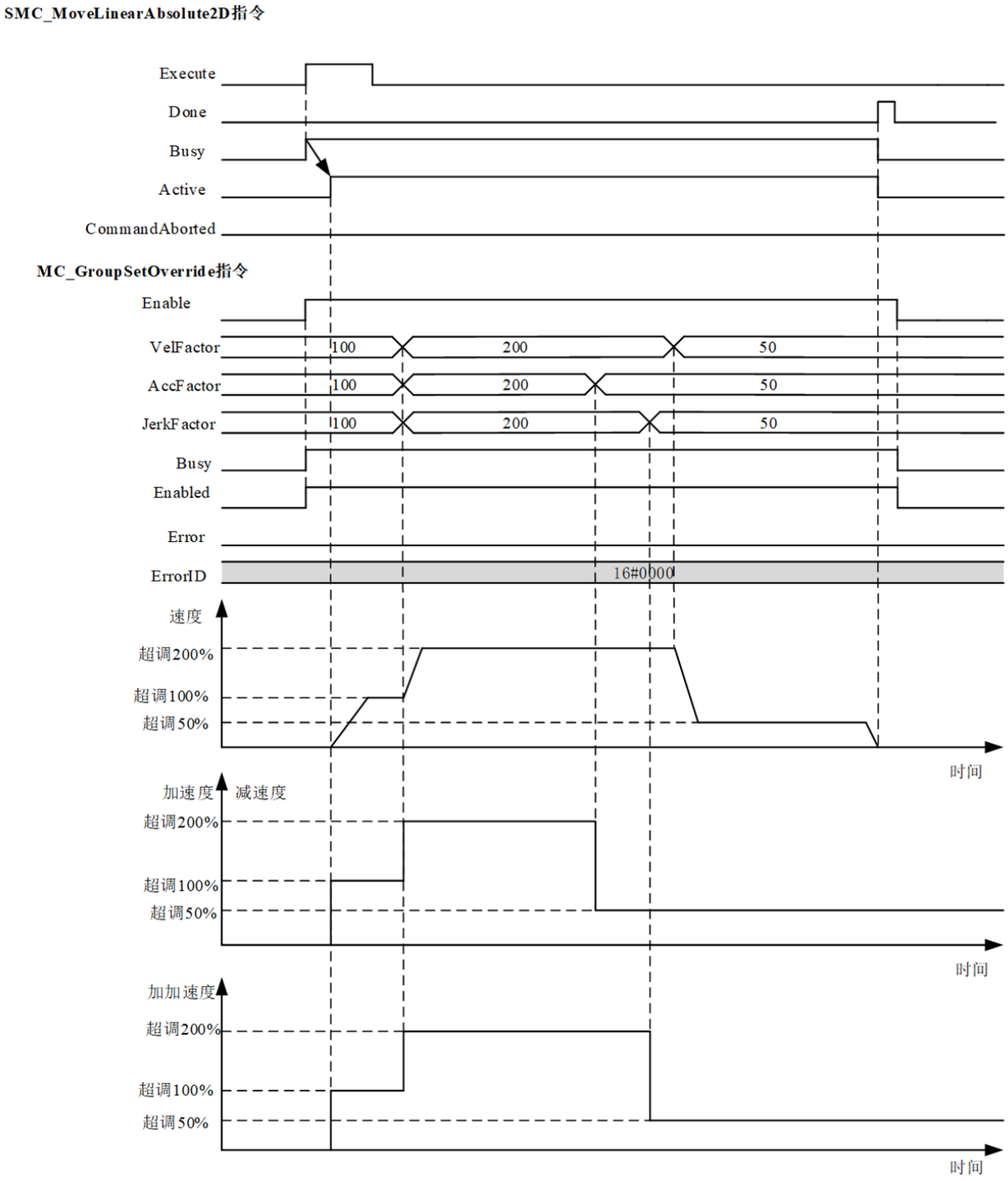
（3） 当超调后的目标速度大于最大速度时，轴组运行的速度为最大速度，加减速度和加加速度也是同理。

（4） 在 Enable(使能)为 TRUE 时，功能块持续进行比例因子超调设置，当 Enable(使能)为 FALSE 时，轴组保持上一次设置的超调比例因子值进行运动。

■ 功能块时序图

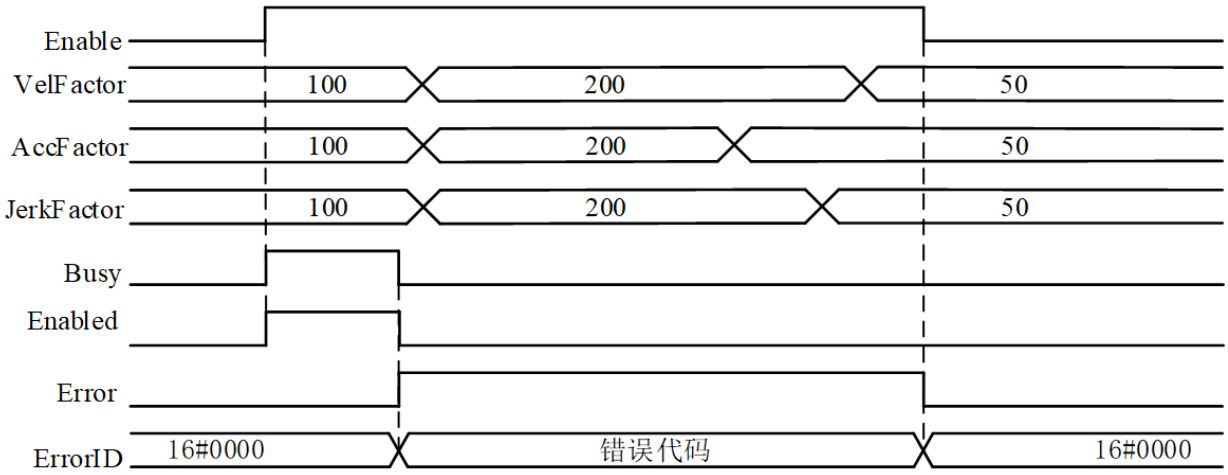
（1） 对 SMC_ MoveLinearAbsolute2D (2 轴绝对位置插补指令)进行超调设置

在 Enable(使能)为 TRUE 时，Busy(忙标志)和 Enabled(设定成功)立即为 TRUE，Enable(使能)为 FALSE 时，Busy(忙标志)和 Enabled(设定成功)立即为 FALSE，在 SMC_MoveLinearAbsolute2D (2 轴绝对位置插补指令)中使用超调指令时的时序图如下所示。



(2) 异常时的时序图

在执行本指令发生异常时，Error(错误)变为 TRUE，Enabled(设定成功)和 Busy(忙标志)变为 FALSE，当错误清除后，Error(错误)变为 FALSE，Enabled(设定成功)和 Busy(忙标志)变为 TRUE，时序图如下所示。



(3) 重启和多重启动指令

本指令的输入为 Enable 型，Enable(使能)为 TRUE 时功能块一直运行，当存在正在执行的 MC_GroupSetOverride 指令时，启动其他的 MC_GroupSetOverride 指令时，后执行的指令被优先处理。

7.7.3 示例

组建 MC_GroupSetOverride 示例程序设定轴组运动参数的超调值。

主要变量说明

名称	数据类型	初始值	注释说明
AxisGrp	AXES_GROUP_REF		轴组变量名称
StartOverride	BOOL	FALSE	开始执行超调时变为 TRUE
BusyOverride	BOOL	FALSE	超调时变为 TRUE
EnabledOverride	BOOL	FALSE	超调成功时变为 TRUE
ErrorOverride	BOOL	FALSE	指令出错时变为 TRUE

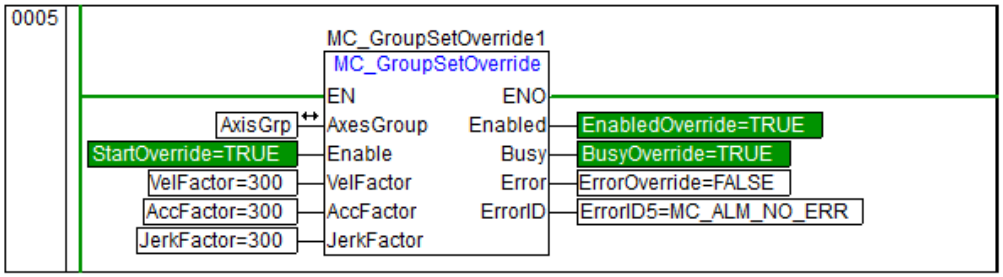
■ 梯形图程序

利用梯形图组建程序，输入参数 VelFactor、AccFactor 和 JerkFactor 均为 300。执行 SMC_MoveLinearAbsolute2D 指令，控制轴组进行运动。将 MC_GroupSetOverride 的 Enable(使能)置为 TRUE，轴组的速度，加减速度和加加速度均表现为超调之后的值，示例程序如下：

(1) 对轴组进行初始化以及使能，分别设置 X、Y 和合轴的 ID 和轴类型，合轴类型设为虚拟轴，然后对轴组进行使能，待使能完成之后对轴组投送 SMC_MoveLinearAbsolute2D 指令，上升沿触发

控制轴组进行 2 轴绝对位置插补指令运行，具体用法可参考 SMC_MoveLinearAbsolute2D 指令章节指令章节。

（2） 保持 MC_GroupSetOverride 的 Enable 引脚为高电平，进行超调参数的设置。



- ST 语言程序

（1） 对轴组进行初始化以及使能，分别设置 X、Y 和合轴的 ID 和轴类型，合轴类型设为虚拟轴，然后对轴组进行使能，待使能完成之后对轴组投送 SMC_MoveLinearAbsolute2D 指令，上升沿触发控制轴组进行 2 轴绝对位置插补指令运行，具体用法可参考 SMC_MoveLinearAbsolute2D 指令章节。

（2） 保持 MC_GroupSetOverride 为高电平，进行运动参数的超调设置。

```
MC_GroupSetOverride1(
Enable := StartOverride,
VelFactor := 300,
AccFactor := 300,
JerkFactor := 300,
Axis := Axis0,
Enabled=>EnabledOverride,
Busy=>BusyOverride,
Error=>ErrorOverride,
ErrorID=>ErrorID);
```

7.7.4 使用注意事项

- 使用该指令进行超调设置时，若超过轴组的最大速度，则以最大速度为运行速度
- 该指令对编码器轴不生效。
- 对于本指令无效的指令，即使将指令的 Enabled 设为 TRUE 时，Enabled 也会保持为 TRUE。

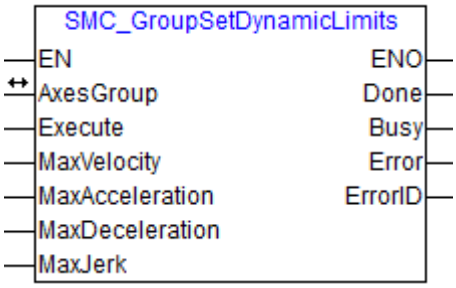
- 该指令仅对于轴组中合轴生效。

7.7.5 参见

- 关于轴组运动指令，参见 SMC_MoveLinearAbsolute2D 指令章节。
- 关于错误码可参见附录功能块错误码。

7.8 SMC_GroupSetDynamicLimits（轴组运动参数限制值设置）

该指令设定轴组运动参数的限制值。



7.8.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	AxesGroup	轴组名	否	-	-	AXIS_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	MaxVelocity	最大速度	是	1E100	正值/0	LREAL
	MaxAcceleration	最大加速度	是	1E100	正值	LREAL
	MaxDeceleration	最大减速度	是	1E100	正值	LREAL
	MaxJerk	最大加加速度	是	1E100	正值	LREAL
OUTPUT	Done	完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

7.8.2 功能

本指令设定轴组运动参数的限制值。

- 指令功能详情

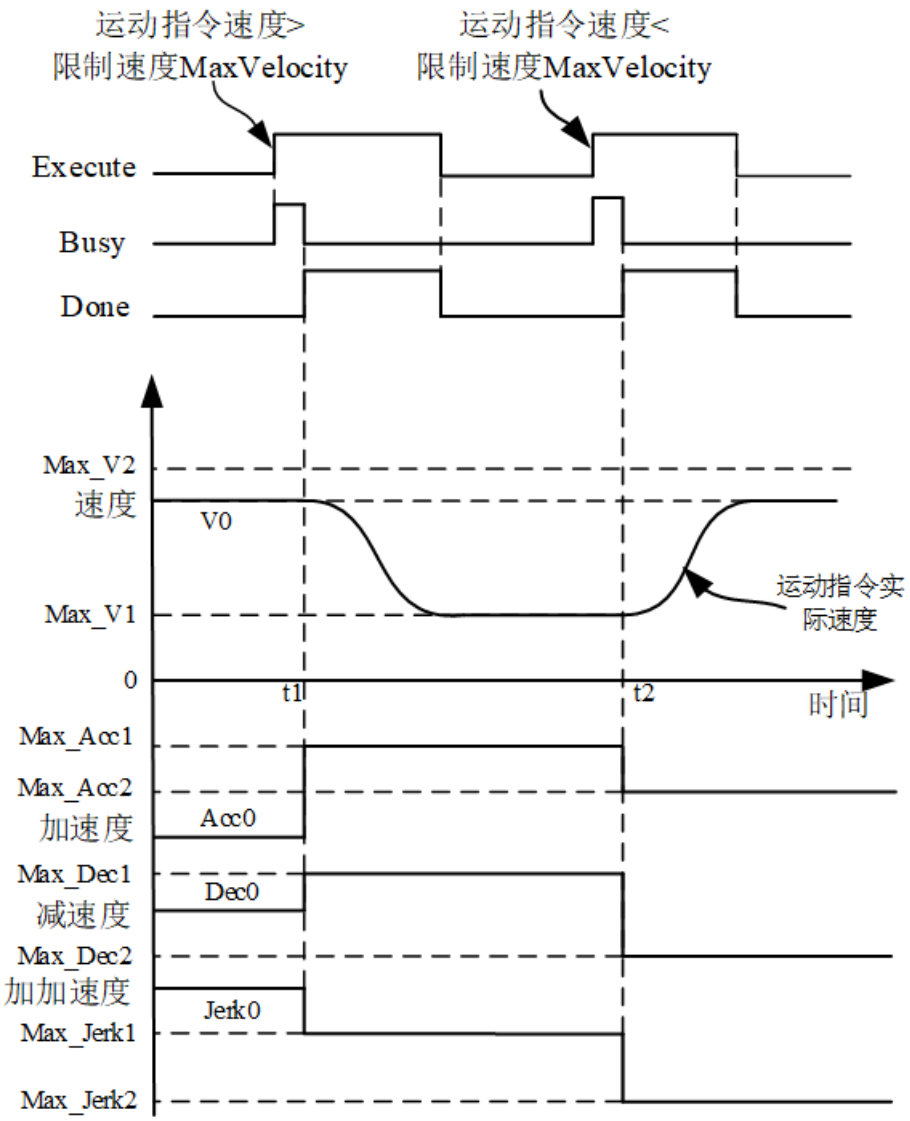
（1） 入参锁存

本指令上升沿触发有效。在 Execute 输入的上升沿时，将输入输出参数 AxesGroup 的合轴以及两个分轴的 ID、输入参数 MaxVelocity、MaxAcceleration、MaxDeceleration 和 MaxJerk 值锁存，指令执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令。

（2） 功能详情

设置轴组运动参数的限制值后，轴组运动指令的运动参数取自身运动参数和设置的限制参数的最小值。

上升沿触发两次该指令，每次触发时设置不同的最大速度 MaxVelocity、最大加速度 MaxAcceleration、最大减速度 MaxDeceleration 和最大加加速度 MaxJerk 值。设置最大运动参数如下图所示：



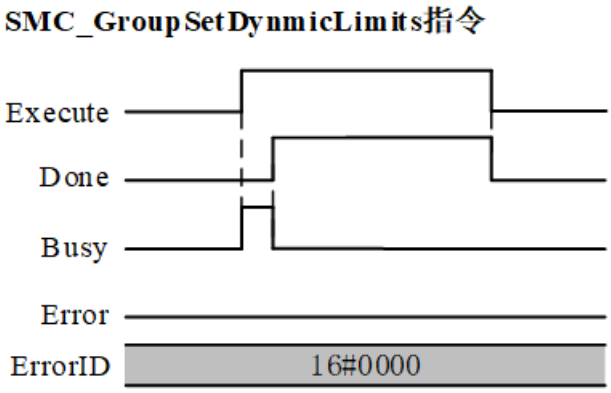
上图中，V0 为运动指令的指令速度、Acc0 为运动指令的指令加速度、Dec0 为运动指令的指令减速度、Jerk0 为运动指令的指令加加速度。

在 t1 时刻，最大速度 MaxVelocity 设置成功，MaxVelocity= Max_V1， MaxVelocity 小于轴组运动指令的合轴指令速度 V0，轴组运动指令运行过程中的最大速度为 MaxVelocity，轴组的运动指令降速。同时，Max_Acc1、Max_Dec1、Max_Jerk1 分别为设定的其余运动参数限制值。

在 t2 时刻，再次成功设置最大速度 MaxVelocity， MaxVelocity= Max_V2，MaxVelocity 大于轴组运动指令的速度 V0，轴组运动指令以自身设置的指令速度作为实际运行速度，轴组运动指令升速。同时，Max_Acc2、Max_Dec2、Max_Jerk2 分别为设定的其余运动参数限制值。

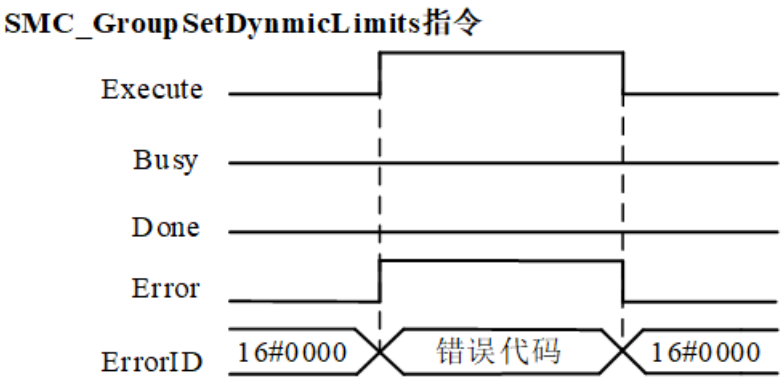
■ 时序图

（1） 正常时序图



（2） 异常错误发生时引脚时序

执行本指令时，若发生参数非法、轴组中各轴参数非法等异常，调用本指令失败，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Execute 变为低电平时，所有输出引脚恢复为默认值。



- 当指令的异常情况解决后，需要重新上升沿触发 Execute，启动 SMC_SetGroupDynamicLimits 功能块，完成轴的运动参数限制设定。

7.8.3 示例

组建 SMC_GroupSetDynamicLimits 示例程序设定轴组运动参数的限制值。

■ 动作示例

（1） 执行插补运动指令 SMC_MoveLinearRelative2D，该插补指令速度设置为 20000，加减速度均为 20000，加加速度为 500000，控制轴组运动。

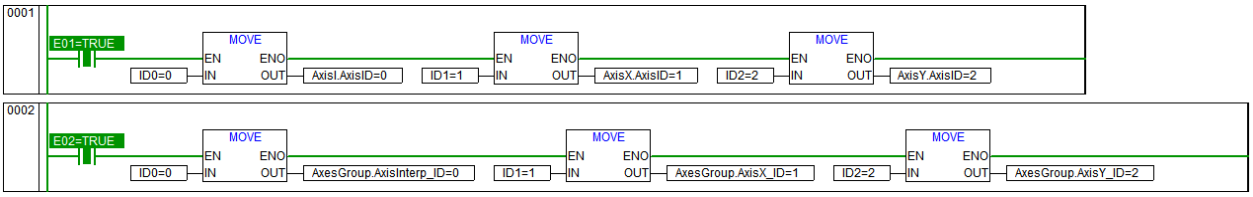
（2） 执行 SMC_GroupSetDynamicLimits 指令，设置运动参数限制 MaxVelocity 为 10000，MaxAcceleration、MaxDeceleration 均为 20000、MaxJerk 为 50000。

■ 梯形图程序

主要变量示例

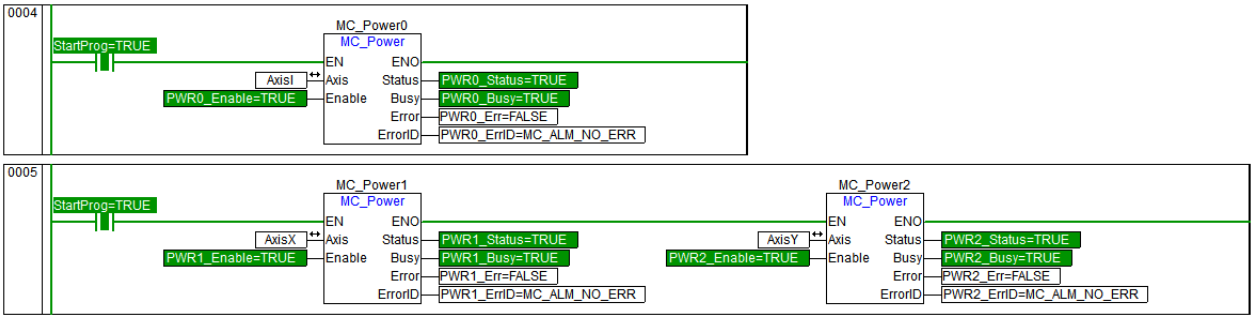
变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power1_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
Power0_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
GrpEnable1_Exe	BOOL	FALSE	轴组使能上升沿触发变量。
AxisHomed	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴回原点完成。
Rel2D1_Exe	BOOL	FALSE	投送的相对插补指令的上升沿触发变量。
Rel2D1_Vel	BOOL	FALSE	投送的相对插补指令的输入速度对应的变量。
Dyn_Exe	BOOL	FALSE	轴组运动参数限制指令的上升沿触发变量。

（1） 设定轴组中合轴和分轴的轴 ID。定义轴组时，首先需要定义轴组中的三个轴，将定义的三个轴的轴号 ID 与轴组中合轴和分轴的 ID 对应即可。见下述 0001 节和 0002 节。

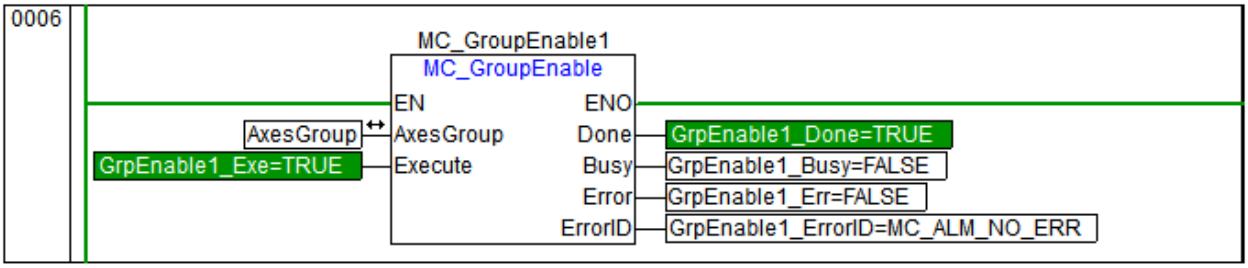


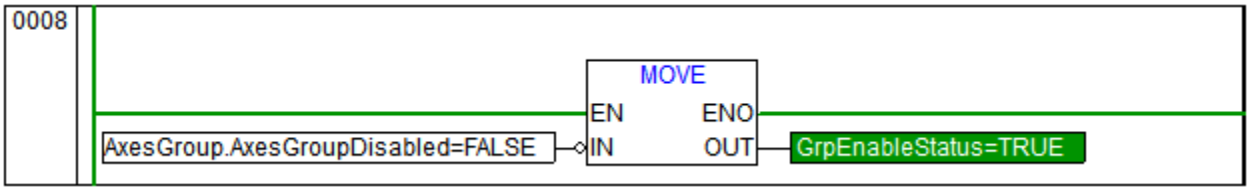
（2） 轴初始化。调用 SMC_AxisInit 指令分别完成轴组中三个轴的轴初始化配置，设置合轴为 0：Virtual，分轴类型为 50: EtherCAT_Position。SMC_AxisInit 指令使用方法参见相关章节介绍。

（3） 轴使能。轴组中各轴准备就绪（StartProg =TRUE）后，调用 MC_Power 对轴组的合轴和两个分轴进行轴使能。见 0004 ~ 0005 节。

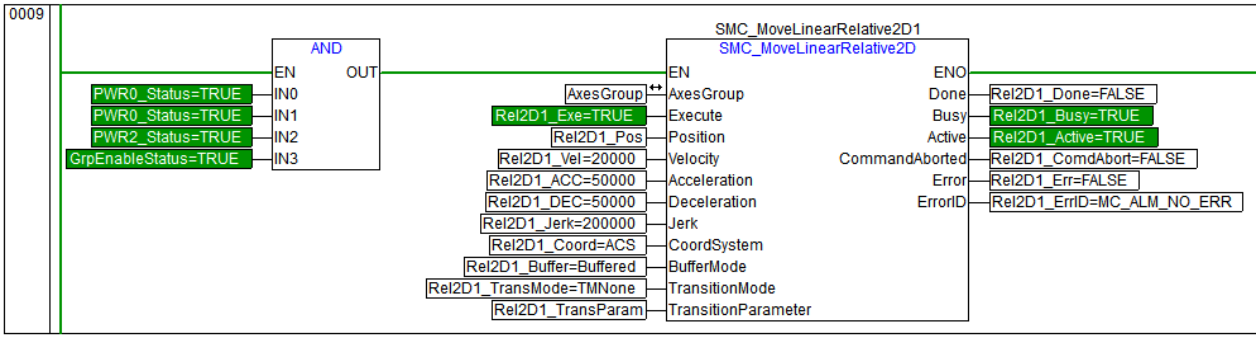


（4） 轴组使能。调用 MC_GroupEnable 对轴组使能。轴组变量中的 AxesGroupDisabled 为 FALSE，轴组使能成功。见 0006 节。轴组使能成功后，将 AxesGroupDisabled 取反赋给 GrpEnableStatus 变量。

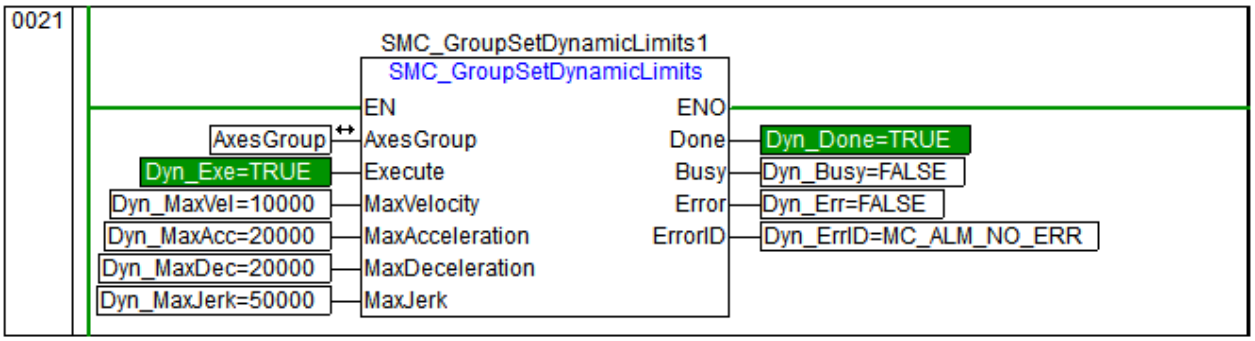




（5） 0009 节中，完成轴组中各轴的使能以及轴组使能后，触发相对位置插补运动指令，控制轴组运动。SMC_MoveLinearRelative2D 指令使用方法参见相关章节。



（6） 0021 节中，执行 SMC_GroupSetDynamicLimits 指令，设置轴组运动参数的限制值。



■ ST 语言程序

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power0_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power1_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
GrpEnable1_Exe	BOOL	FALSE	轴组使能上升沿触发变量。
AxisHomed	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴回原点完成。
Rel2D1_Exe	BOOL	FALSE	投送的相对插补指令的上升沿触发变量。
Rel2D1_Vel	BOOL	FALSE	投送的相对插补指令的输入速度对应的变量。。
Dyn_Exe	BOOL	FALSE	运动参数限制指令的上升沿触发变量。
InitFlag	BOOL	FALSE	如果该变量为 TRUE，则初始化参数设置完成。

（1） 设定轴组中合轴和分轴的轴 ID。设定所定义的合轴 AxisI、分轴 AxisX 和 AxisY 的轴 ID。

```
IF FALSE=InitFlag THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
```

```
    Dyn_MaxVel:=10000;          (*速度限制*)
    Dyn_MaxAcc:=20000;          (*加速度限制*)
    Dyn_MaxDec:=20000;          (*减速度限制*)
    Dyn_MaxJerk:=50000;         (*加加速度限制*)
    InitFlag:=TRUE;
END_IF
```

（2） 调用 SMC_AxisInit 指令完成轴组中三个轴的轴初始化配置。轴组合轴轴类型设置为 0：Virtual；轴组分轴 AxisX/AxisY 轴类型均设置为 50：EtherCAT_Position。SMC_AxisInit 指令使用方法参见其指令章节介绍。

（3） 执行 MC_Power 指令完成轴组中三个轴的轴使能。

```
IF StartPro THEN
    MC_Power0(                          (*合轴使能*)
    Enable := PWR0_Enable,
    Axis := AxisI,
    Status=> PWR0_Status,
    Busy=> PWR0_Busy,
    Error=> PWR0_Err,
    ErrorID=> PWR0_ErrID);
    MC_Power1(                          (*分轴 AxisX 使能*)
    Enable := PWR1_Enable,
    Axis := AxisX,
    Status=> PWR1_Status,
    Busy=> PWR1_Busy,
    Error=> PWR1_Err,
    ErrorID=> PWR1_ErrID);
    MC_Power2(                          (*分轴 AxisY 使能*)
    Enable := PWR2_Enable,
    Axis := AxisY,
    Status=> PWR2_Status,
    Busy=> PWR2_Busy,
    Error=> PWR2_Err,
    ErrorID=> PWR2_ErrID);
```

```
END_IF
```

（4） 执行轴组使能。

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

（5） 轴以及轴组使能功能后，执行轴组插补指令。

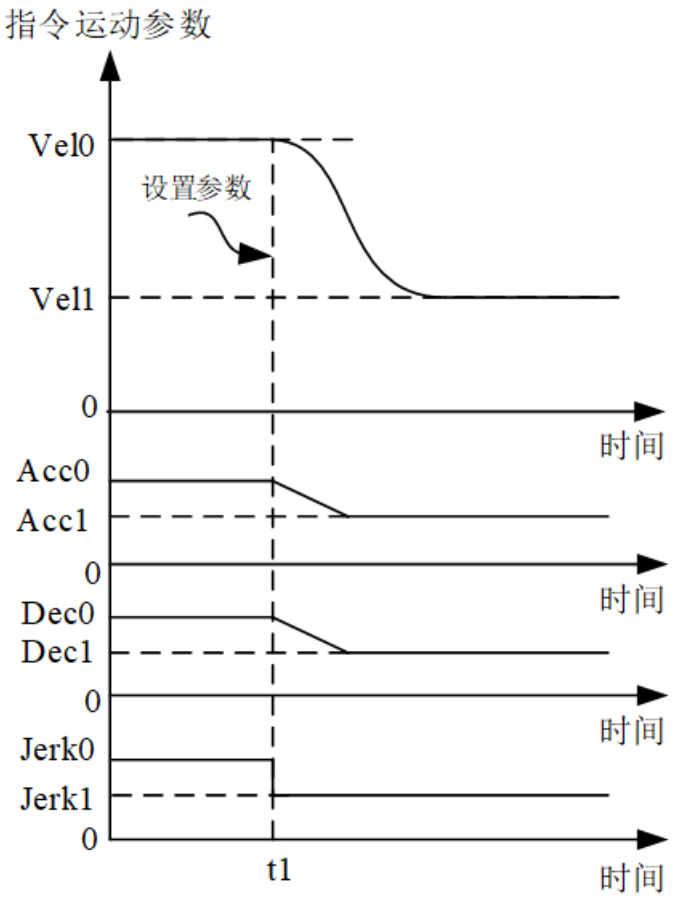
```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus THEN
   SMC_MoveLinearRelative2D1(
   Execute := Rel2D1_Exe,
   Position := Rel2D1_Pos,
   Velocity := Rel2D1_Vel,
   Acceleration := Rel2D1_ACC,
   Deceleration := Rel2D1_DEC,
   Jerk := Rel2D1_Jerk,
   CoordSystem := Rel2D1_Coord,
   BufferMode := Rel2D1_Buffer,
   TransitionMode := Rel2D1_Trans,
   TransitionParameter := Rel2D1_Param,
   AxesGroup := AxesGroup,
   Done=> Rel2D1_Done,
   Busy=> Rel2D1_Busy,
   Active=> Rel2D1_Active,
   CommandAborted=> Rel2D1_ComdAbort,
   Error=> Rel2D1_Err,
   ErrorID=> Rel2D1_ErrID);
END_IF
```

（6） 执行 SMC_GroupSetDynamicLimits 指令，完成运动参数限制设置。

```
SMC_SetGroupDynamicLimits1(
Execute := Dyn_Exe,
MaxVelocity := Dyn_MaxVel,
MaxAcceleration := Dyn_MaxAcc,
```

```
MaxDeceleration := Dyn_MaxDec,
MaxJerk := Dyn_MaxJerk,
AxesGroup := AxesGroup,
Done=> Dyn_Done,
Busy=> Dyn_Busy,
Error=> Dyn_Err,
ErrorID=> Dyn_ErrID);
```

观察插补运动指令的运动参数曲线，在 0~t1 时间段，插补指令运动参数未受限，插补指令的实际运行速度达到设置的指令速度；在 t1 时间后，SMC_GroupSetDynamicLimits 指令设置的运动参数限制生效，插补指令速度减速到速度限制值。插补合轴运动参数受限制示意图如下所示：



Vel0、Acc0、Dec0、Jerk0 分别为插补指令的指令速度、加速度、减速度、加加速度

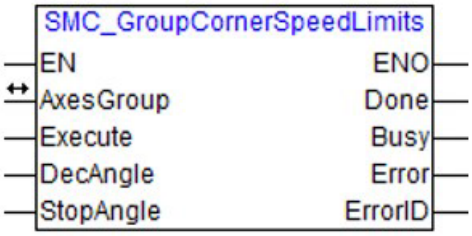
Vel1、Acc1、Dec1、Jerk1 分别为插补指令的限制参数设置后的速度、加速度、减速度、加加速度

7.8.4 使用注意事项

- 调用该指令时，仅 MaxVelocity 可设为 0，其余输入运动限制参数需为正数。
- 该指令设置最大参数成功后，下降沿不能取消最大参数限制。用户需重新调用该指令设置参数限制值。

7.9 SMC_GroupCornerSpeedLimits（轴组转角限速设置）

该指令设定轴组运动过程中的转角限速参数，减小非光滑转角衔接处的运动冲击。



7.9.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	AxesGroup	轴组名	否	-	-	AXES_GROUP_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	DecAngle	临界减速角度，单位为度	是	30	0 ~ 30 （≤StopAngle）	LREAL
	StopAngle	临界停止角度，单位为度	是	90	0 ~ 90 （不可为 0）	LREAL
OUTPUT	Done	完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

7.9.2 功能

本指令设定轴组运动过程中的转角限速参数，减小非光滑转角衔接处的运动冲击。

■ 指令功能详情

（1） 入参锁存

本指令上升沿触发有效。在 Execute 输入的上升沿时，将输入输出参数 AxesGroup 的合轴以及两个分轴的 ID、输入参数 DecAngle 和 StopAngle 值锁存，指令执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令。

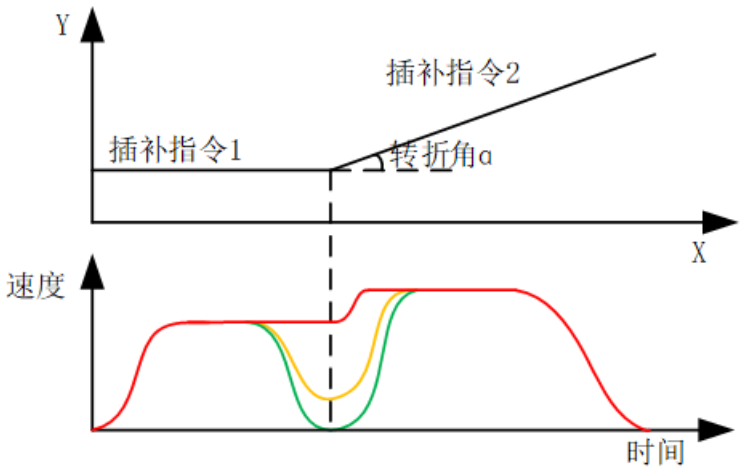
（2） 转接限速参数

临界减速角度 DecAngle 可以限制轴组运动指令衔接处的速度，临界停止角度 DecAngle 限制轴组运动指令衔接处的速度为 0。临界减速角度 DecAngle 小于等于临界停止角度 DecAngle。

（3） 功能说明

设置轴组运动过程中的转角限速参数，轴组插补运动指令衔接处的速度受到限制，减小非光滑转角衔接处的运动冲击。

如下图，两条插补指令衔接处速度曲线，以第一条插补指令的速度为两条指令的衔接处的速度。



上图中，设定插补运动指令 2 以插补指令 1 的速度融合，即设置第二条插补指令的起始点速度为插补指令 1 的指令速度。

若设定的临界减速角度和临界停止角度大于两条指令之间的转折角 α，则两条指令之间衔接速度不受限，速度曲线见红色速度曲线；

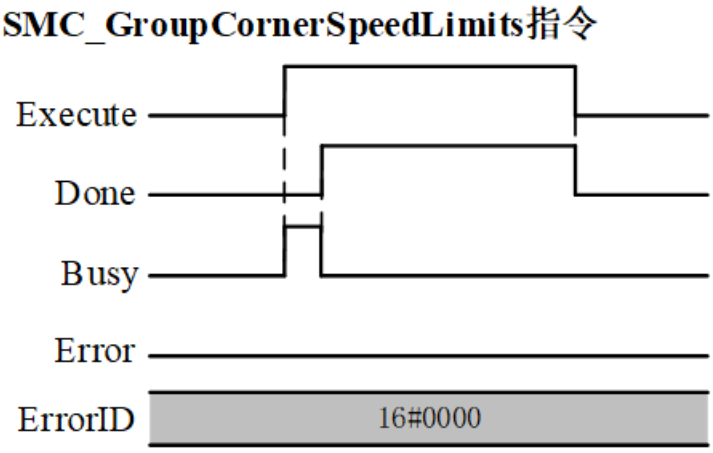
若设定的临界减速角度小于两条指令之间的转折角 α，且临界停止角度大于两条指令之间的转折角

α，则两条指令之间衔接速度受限，速度曲线见橙色速度曲线；

若设定的临界减速角度和临界停止角度均小于两条指令之间的转折角 α，则两条指令之间衔接速度受限，且衔接处速度姜维 0，速度曲线见绿色速度曲线。

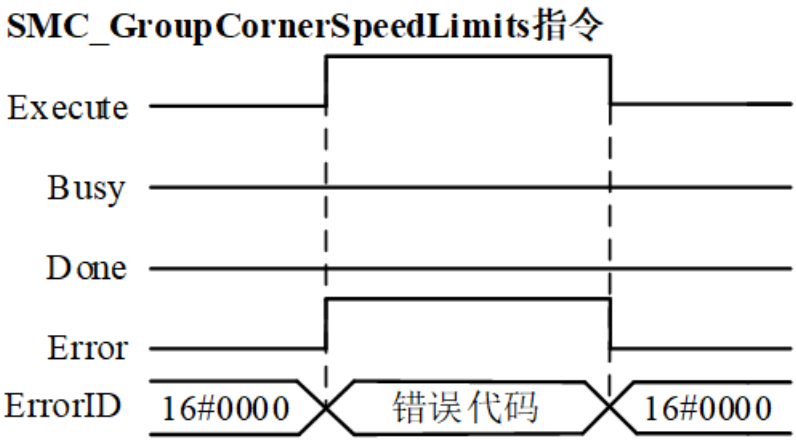
■ 时序图

（1） 正常时序图



（2） 异常错误发生时引脚时序

执行本指令时，若发生参数非法、轴组中各轴参数非法等异常，调用本指令失败，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Execute 变为低电平时，所有输出引脚恢复为默认值。



- 当指令的异常情况解决后，需要重新上升沿触发 Execute，启动 SMC_SetGroupCornerSpeedLimits 功能块，完成轴组的转角限速参数设定。

7.9.3 示例

- 动作示例

（1） 动作介绍

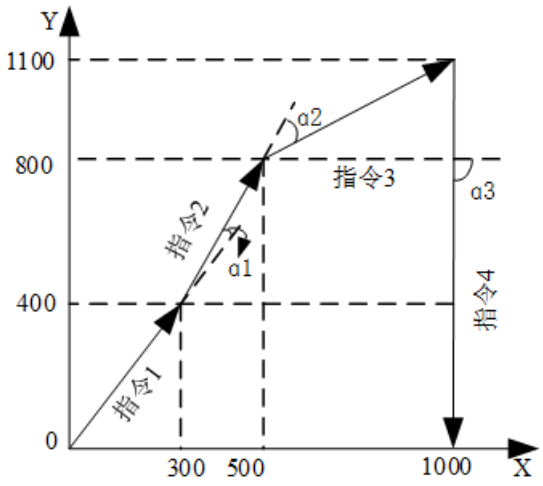
执行 SMC_GroupCornerSpeedLimits 指令设定轴组的转角限速参数。

以下示例执行 4 条直线插补指令。

在本例中，直线插补指令 1 的缓存模式 BufferMode 设定为 Buffered。其余 3 条插补指令的缓存模式 BufferMode 分别为 3：以上一条指令的速度融合、4：以下一条指令的速度融合、5：以相邻指令间的高速融合。

在本例中，直线插补指令 1 的输出 Active 变为 TRUE 时，执行其余插补指令 2-4。

- 动作示意图



依次执行直线插补指令 1 到达（300,400）位置处，执行直线插补指令 2 到达（500,800）位置处，执行直线插补指令 3 到达（1000,1100）位置处，执行直线插补指令 4 到达（1000, 0）位置处。

插补指令 1 和指令 2 的转折角 α1 约为 10°、插补指令 2 和指令 3 的转折角为 32°、插补指令 3 和指令 4 的转折角为 121°。

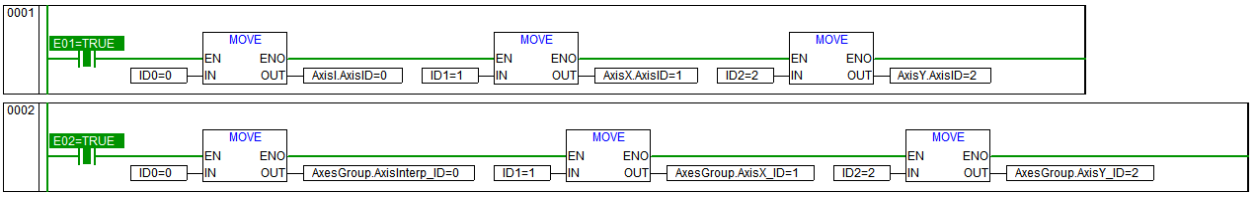
执行 SMC_GroupCornerSpeedLimits 设定轴组的转角限速参数， 减速角 DecAngle 设置为 20°，停止角 StopAngle 设置为 60°。

■ 梯形图程序

主要变量说明

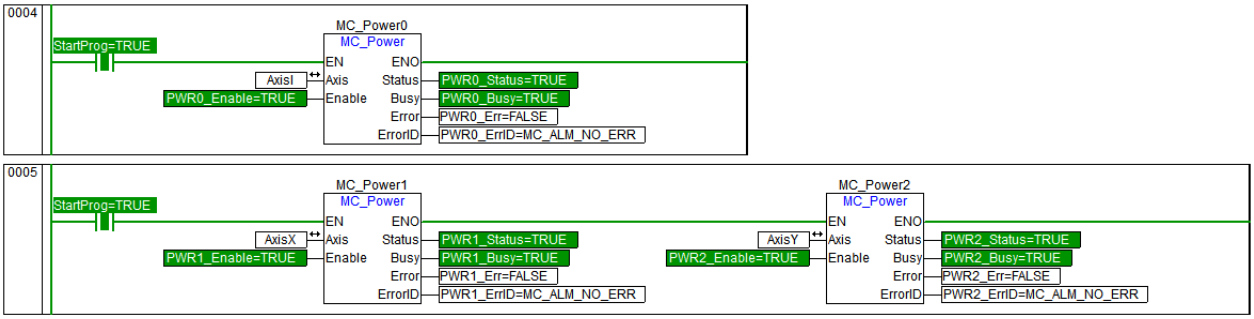
变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power1_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
Power0_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
AxisHomed	BOOL	FALSE	轴回零完成变量。如果该变量为 TRUE，则轴组中的各轴回原点完成。
Rel2D1_Active	BOOL	FALSE	轴组投送的相对插补指令 1 的 Active 引脚的变量，该变量变为 TRUE 时，轴组执行插补指令。

（1） 设定轴组中合轴和分轴的轴 ID。定义轴组时，首先需要定义轴组中的三个轴，将定义的三个轴的轴号 ID 与轴组中合轴和分轴的 ID 对应即可。见下述 0001 节和 0002 节。

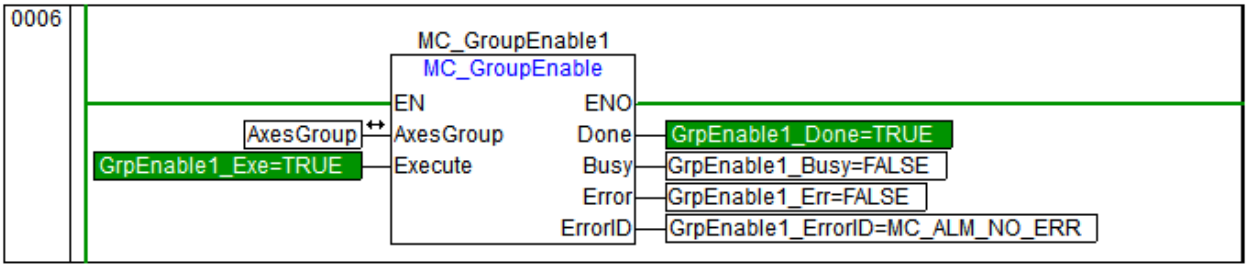


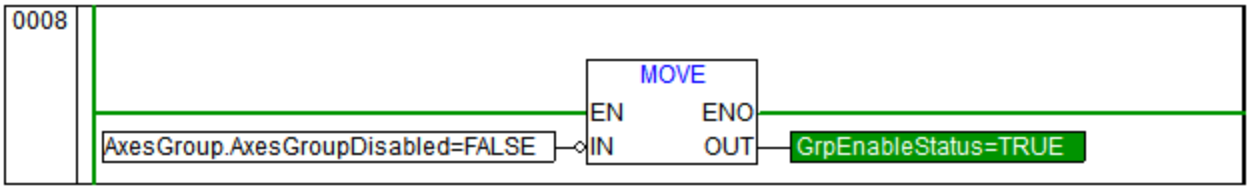
（2） 轴初始化。调用 SMC_AxisInit 指令分别完成轴组中三个轴的轴初始化配置，设置合轴为 0：Virtual，分轴类型为 50: EtherCAT_Position。SMC_AxisInit 指令使用方法参见其章节绍。

（3） 轴使能。轴组中各轴准备就绪（StartProg =TRUE）后，调用 MC_Power 对轴组的合轴和两个分轴进行轴使能。见 0004 ~ 0005 节。

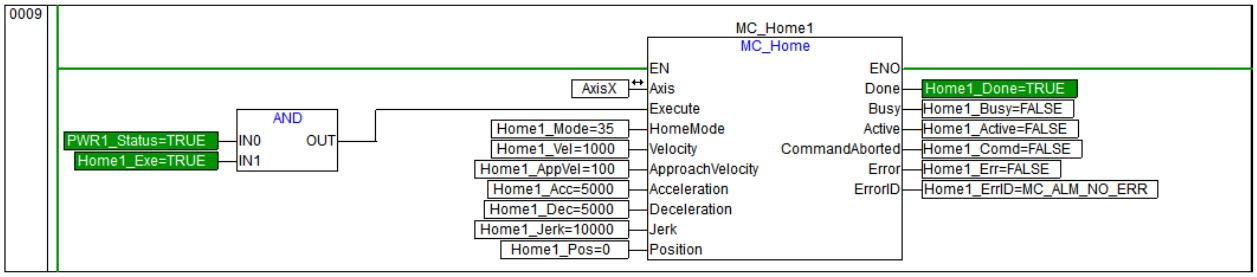


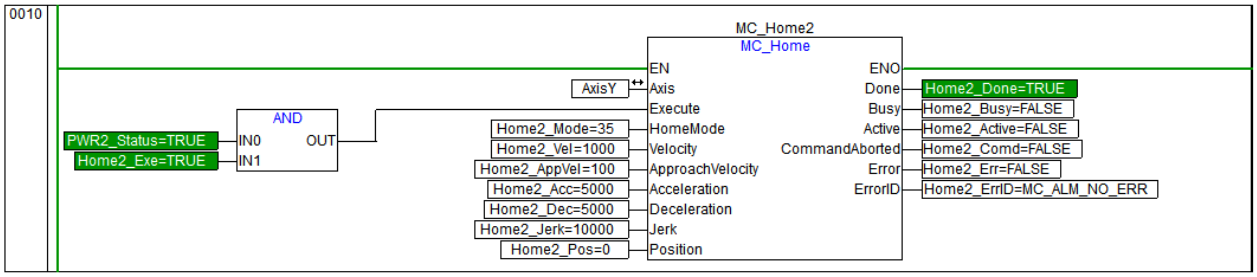
（4） 轴组使能。调用 MC_GroupEnable 对轴组使能。轴组变量中的 AxesGroupDisabled 为 FALSE，轴组使能成功，见 0006 节。轴组使能成功后，将 AxesGroupDisabled 取反赋给 GrpEnableStatus 变量，见 0008 节。

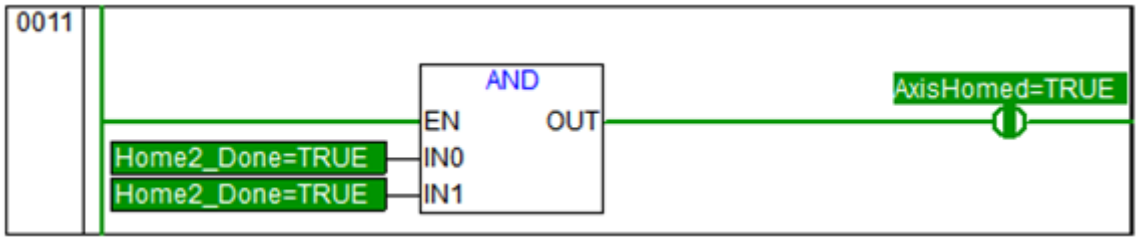




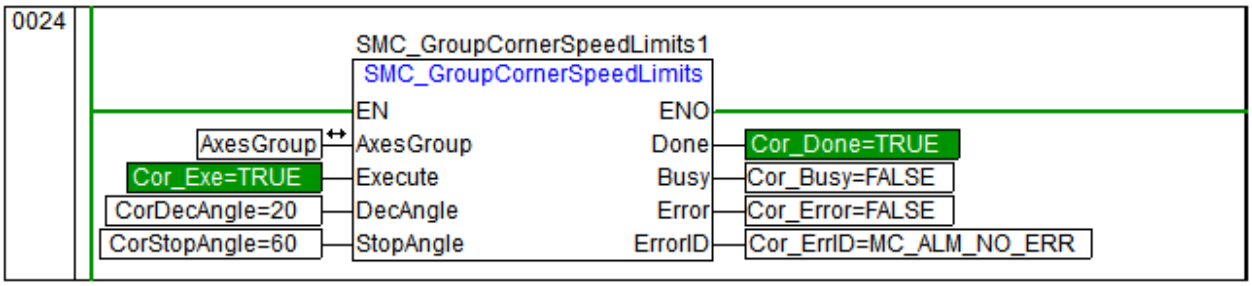
（5） 轴组中两个分轴使能完成后，若原点未确定，这执行回零操作。回零完成后，回零完成变量 AxisHomed 置 TRUE。见 0009-0011 节。



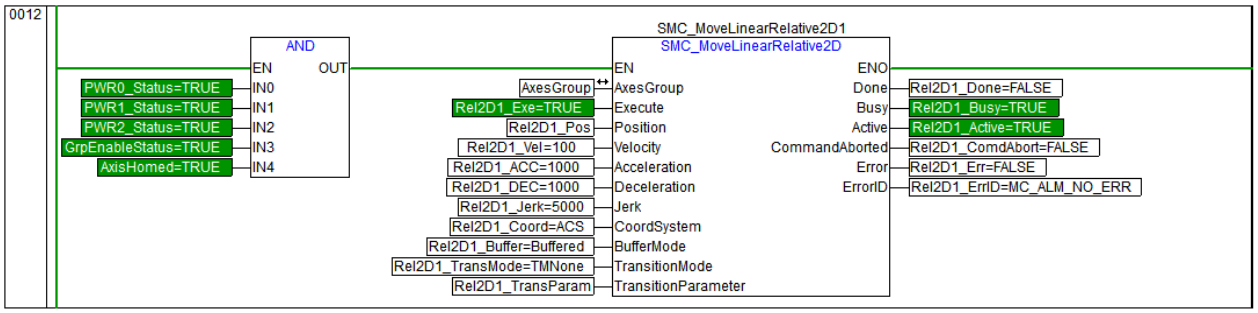


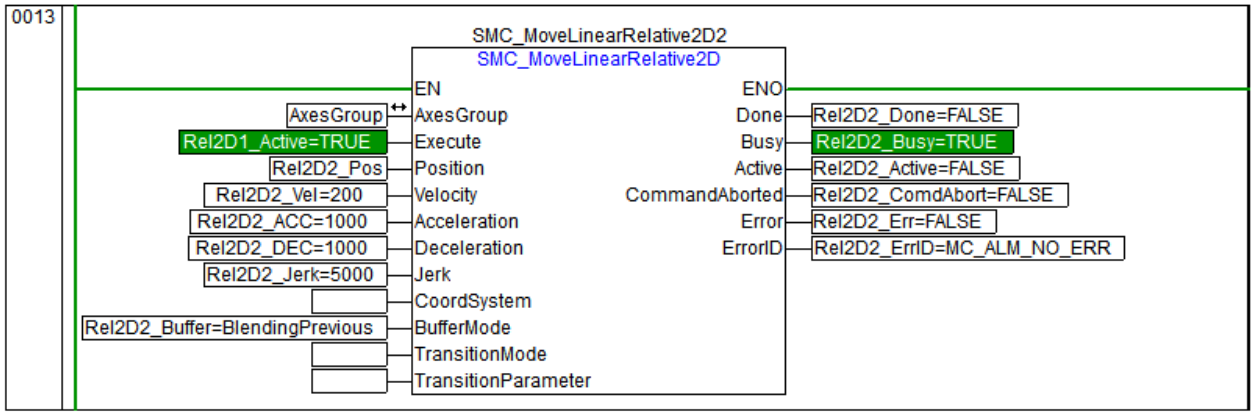


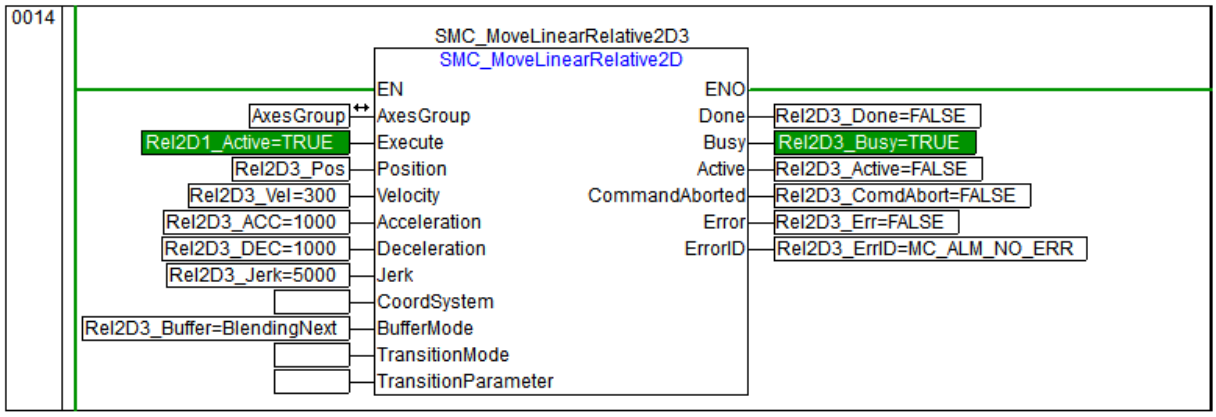
（6） 0024 节中，执行 SMC_GroupCornerSpeedLimits 指令，设置转角限速参数。

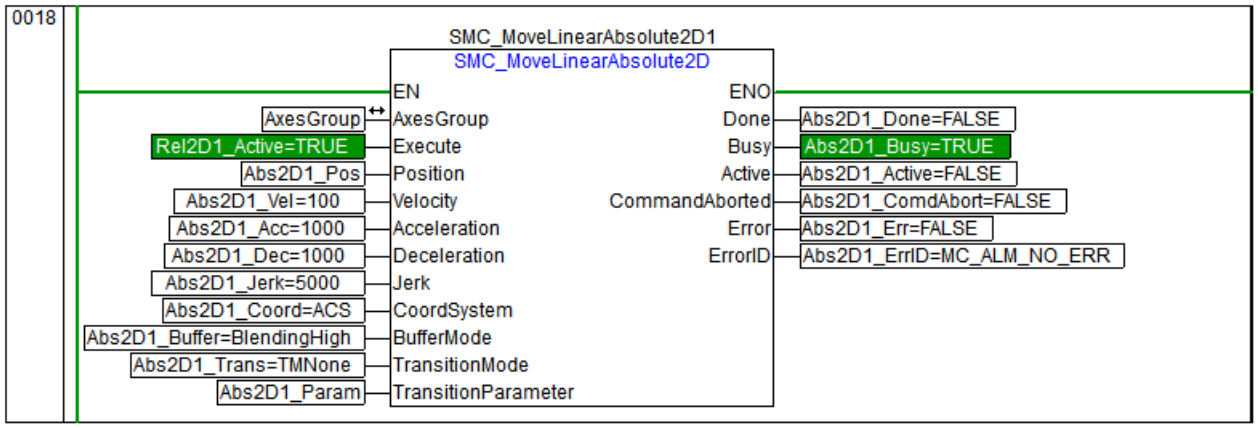


（7） 0011 节中，轴回原点完成后，开始执行插补运动指令。执行指令 1，当指令 1 的 Rel2D1_Active 为 TRUE 时，依次投送指令 2-4。见下述 0012-0014 节、0018 节。


0012
AND
EN
OUT
IN0
IN1
IN2
IN3
IN4
PWR0_Status=TRUE
PWR1_Status=TRUE
PWR2_Status=TRUE
GrpEnableStatus=TRUE
AxisHomed=TRUE
SMC_MoveLinearRelative2D1
SMC_MoveLinearRelative2D
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
AxesGroup
Rel2D1_Exe=TRUE
Rel2D1_Pos
Rel2D1_Vel=100
Rel2D1_ACC=1000
Rel2D1_DEC=1000
Rel2D1_Jerk=5000
Rel2D1_Coord=ACS
Rel2D1_Buffer=Buffered
Rel2D1_TransMode=TMNone
Rel2D1_TransParam
Rel2D1_Done=FALSE
Rel2D1_Busy=TRUE
Rel2D1_Active=TRUE
Rel2D1_ComdAbort=FALSE
Rel2D1_Err=FALSE
Rel2D1_ErrID=MC_ALM_NO_ERR


0013
SMC_MoveLinearRelative2D2
SMC_MoveLinearRelative2D
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
AxesGroup
Rel2D1_Active=TRUE
Rel2D2_Pos
Rel2D2_Vel=200
Rel2D2_ACC=1000
Rel2D2_DEC=1000
Rel2D2_Jerk=5000
Rel2D2_Buffer=BlendingPrevious
Rel2D2_Done=FALSE
Rel2D2_Busy=TRUE
Rel2D2_Active=FALSE
Rel2D2_ComdAbort=FALSE
Rel2D2_Err=FALSE
Rel2D2_ErrID=MC_ALM_NO_ERR


0014
SMC_MoveLinearRelative2D3
SMC_MoveLinearRelative2D
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
AxesGroup
Rel2D1_Active=TRUE
Rel2D3_Pos
Rel2D3_Vel=300
Rel2D3_ACC=1000
Rel2D3_DEC=1000
Rel2D3_Jerk=5000
Rel2D3_Buffer=BlendingNext
Rel2D3_Done=FALSE
Rel2D3_Busy=TRUE
Rel2D3_Active=FALSE
Rel2D3_ComdAbort=FALSE
Rel2D3_Err=FALSE
Rel2D3_ErrID=MC_ALM_NO_ERR


0018
SMC_MoveLinearAbsolute2D1
SMC_MoveLinearAbsolute2D
EN
AxesGroup
Execute
Position
Velocity
Acceleration
Deceleration
Jerk
CoordSystem
BufferMode
TransitionMode
TransitionParameter
ENO
Done
Busy
Active
CommandAborted
Error
ErrorID
AxesGroup
Rel2D1_Active=TRUE
Abs2D1_Pos
Abs2D1_Vel=100
Abs2D1_Acc=1000
Abs2D1_Dec=1000
Abs2D1_Jerk=5000
Abs2D1_Coord=ACS
Abs2D1_Buffer=BlendingHigh
Abs2D1_Trans=TMNone
Abs2D1_Param
Abs2D1_Done=FALSE
Abs2D1_Busy=TRUE
Abs2D1_Active=FALSE
Abs2D1_ComdAbort=FALSE
Abs2D1_Err=FALSE
Abs2D1_ErrID=MC_ALM_NO_ERR

■ ST 语言程序

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power0_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
Power1_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
AxisHomed	BOOL	FALSE	轴回零完成变量。如果该变量为 TRUE，则轴组中的各轴回原点完成。
InitFlag	BOOL	FALSE	如果该变量为 TRUE，则初始化参数设置完成。
Rel2D1_Exe	BOOL	FALSE	投送的相对插补指令 1 的上升沿触发变量。
Rel2D1_Active	BOOL	FALSE	投送的相对插补指令 1 的 Active 引脚的变量，该变量变为 TRUE 时，轴组执行插补指令。

（1） 设定轴组中合轴和分轴的轴 ID。设定所定义的合轴 AxisI、分轴 AxisX 和 AxisY 的轴 ID。

```
IF Switch1 THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

（2） 设置指令运动参数

```
IF FALSE = InitFlag THEN
 (*相对位置直线插补指令 1 的参数设置。未设置的输入参数采用默认值*)
   Rel2D1_DEC:=1000;,
   Rel2D1_Jerk:=5000;,
   Rel2D1_Buffer:=Buffered;
   (*相对位置直线插补指令 2 的参数设置。未设置的输入参数采用默认值*)
   Rel2D2_Pos[0]:= 200;
```

```
  Rel2D2_Pos[1]:=400;
  Rel2D2_Vel:=200;
  Rel2D2_ACC:=1000;,
  Rel2D2_DEC:=1000;,
  Rel2D2_Jerk:=5000;,
  Rel2D2_Buffer:= BlendingPrevious;
  (*相对位置直线插补指令 3 的参数设置。未设置的输入参数采用默认值*)
  Rel2D2_Pos[0]:= 500;
  Rel2D2_Pos[1]:=300;
  Rel2D2_Vel:=300;
  Rel2D2_ACC:=1000;,
  Rel2D2_DEC:=1000;,
  Rel2D2_Jerk:=5000;,
  Rel2D2_Buffer:= BlendingNext;
  (*绝对位置直线插补指令 4 的参数设置。未设置的输入参数采用默认值*)
  Abs2D1_Pos[0]:= 1000;
  Abs2D1_Pos[1]:= 0;
  Abs2D1_Vel:=100;
  Abs2D1_ACC:=1000;,
  Abs2D1_DEC:=1000;,
  Abs2D1_Jerk:=5000;,
  AbsD1_Buffer:= BlendingHigh;
  (*转角限速参数设置*)
  CorDecAngle:=20;
  CorStopAngle:=60;
  InitFlag:=TRUE;        (*参数初始化标志*)
END_IF
```

（3） 调用 SMC_AxisInit 指令完成轴组中三个轴的轴初始化配置。轴组合轴轴类型设置为 0：Virtual；轴组分轴 AxisX/AxisY 轴类型均设置为 50：EtherCAT_Position。SMC_AxisInit 指令使用方法参见其指令章节介绍。

（4） 执行 MC_Power 指令完成轴组中三个轴的轴使能。

```
IF StartPro THEN
  MC_Power0(                          (*合轴使能*)
```

```
  Enable := PWR0_Enable,
  Axis := AxisI,
  Status=> PWR0_Status,
  Busy=> PWR0_Busy,
  Error=> PWR0_Err,
  ErrorID=> PWR0_ErrID);
  MC_Power1(                          (*分轴 AxisX 使能*)
  Enable := PWR1_Enable,
  Axis := AxisX,
  Status=> PWR1_Status,
  Busy=> PWR1_Busy,
  Error=> PWR1_Err,
  ErrorID=> PWR1_ErrID);
  MC_Power2(                          (*分轴 AxisY 使能*)
  Enable := PWR2_Enable,
  Axis := AxisY,
  Status=> PWR2_Status,
  Busy=> PWR2_Busy,
  Error=> PWR2_Err,
  ErrorID=> PWR2_ErrID);
END_IF
```

（5） 执行轴组使能。

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

（6） 执行轴组中分轴的原点回归指令，确定原点，MC_Home 指令使用方法见其介绍章节。

```
IF  PWR1_Status THEN
  MC_Home1(
  Execute := Home1_Exe,
  HomeMode := Home1_Mode,
```

```
    Velocity := Home1_Vel,
    ApproachVelocity := Home1_AppVel,
    Acceleration := Home1_Acc,
    Deceleration := Home1_Dec,
    Jerk := Home1_Jerk,
    Position := Home1_Pos,
    Axis := AxisX,
    Done=> Home1_Done,
    Busy=> Home1_Busy,
    Active=> Home1_Active,
    CommandAborted=> Home1_Comd,
    Error=> Home1_Err,
    ErrorID=> Home1_ErrID);
    MC_Home2(
    Execute := Home2_Exe,
    HomeMode := Home2_Mode,
    Velocity := Home2_Vel,
    ApproachVelocity := Home2_AppVel,
    Acceleration := Home2_Acc,
    Deceleration := Home2_Dec,
    Jerk := Home2_Jerk,
    Position := Home2_Pos,
    Axis := AxisY,
    Done=> Home2_Done,
    Busy=> Home2_Busy,
    Active=> Home2_Active,
    CommandAborted=> Home2_Comd,
    Error=> Home2_Err,
    ErrorID=> Home2_ErrID);
END_IF
IF  Home1_Done  AND Home2_Done THEN
    AxisHomed:=TRUE;
END_IF
```

（7） 执行转角限速设参指令。

```
SMC_GroupCornerSpeedLimits1(
Execute := Cor_Exe,
DecAngle := Cor_DecAngle,
```

```
StopAngle := Cor_StopAngle,
AxesGroup := AxesGroup,
Done=> Cor_Done,
Busy=> Cor_Busy,
Error=> Cor_Err,
ErrorID=> Cor_ErrID);
```

（8） 轴回原点完成后，执行插补运动指令。执行指令 1，当指令 1 的 Rel2D1_Active 为 TRUE 时，依次投送指令 2-4。

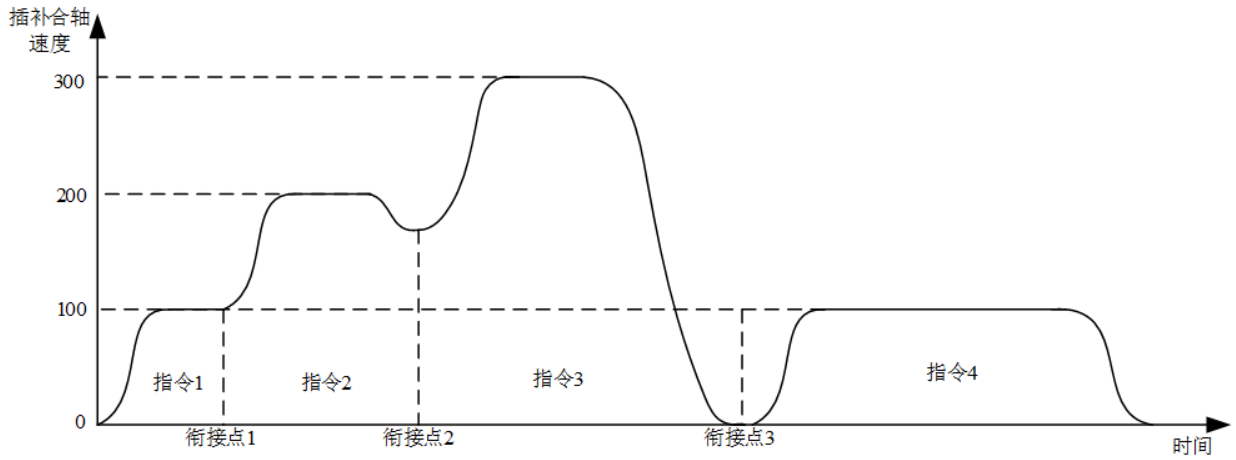
```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus AND AxisHomed THEN
   SMC_MoveLinearRelative2D1(            (*插补指令 1*)
   Execute := Rel2D1_Exe,
   Position := Rel2D1_Pos,
   Velocity := Rel2D1_Vel,
   Acceleration := Rel2D1_ACC,
   Deceleration := Rel2D1_DEC,
   Jerk := Rel2D1_Jerk,
   CoordSystem := Rel2D1_Coord,
   BufferMode := Rel2D1_Buffer,
   TransitionMode := Rel2D1_Trans,
   TransitionParameter := Rel2D1_Param,
   AxesGroup := AxesGroup,
   Done=> Rel2D1_Done,
   Busy=> Rel2D1_Busy,
   Active=> Rel2D1_Active,
   CommandAborted=> Rel2D1_ComdAbort,
   Error=> Rel2D1_Err,
   ErrorID=> Rel2D1_ErrID);
   SMC_MoveLinearRelative2D2(              (*插补指令 2*)
   Execute := Rel2D1_Active,
   Position := Rel2D2_Pos,
   Velocity := Rel2D2_Vel,
   Acceleration := Rel2D2_ACC,
   Deceleration := Rel2D2_DEC,
   Jerk := Rel2D2_Jerk,
   CoordSystem := Rel2D2_Coord,
   BufferMode := Rel2D2_Buffer,
```

```
TransitionMode := Rel2D2_Trans,
TransitionParameter := Rel2D2_Param,
AxesGroup := AxesGroup,
Done=> Rel2D2_Done,
Busy=> Rel2D2_Busy,
Active=> Rel2D2_Active,
CommandAborted=> Rel2D2_ComdAbort,
Error=> Rel2D2_Err,
ErrorID=> Rel2D2_ErrID);
SMC_MoveLinearRelative2D3(                (*插补指令 3*)
Execute := Rel2D1_Active,
Position := Rel2D3_Pos,
Velocity := Rel2D3_Vel,
Acceleration := Rel2D3_ACC,
Deceleration := Rel2D3_DEC,
Jerk := Rel2D3_Jerk,
CoordSystem := Rel2D3_Coord,
BufferMode := Rel2D3_Buffer,
TransitionMode := Rel2D3_Trans,
TransitionParameter := Rel2D3_Param,
AxesGroup := AxesGroup,
Done=> Rel2D3_Done,
Busy=> Rel2D3_Busy,
Active=> Rel2D3_Active,
CommandAborted=> Rel2D3_ComdAbort,
Error=> Rel2D3_Err,
ErrorID=> Rel2D3_ErrID);
SMC_MoveLinearAbsolute2D1(            (*插补指令 4*)
Execute := Rel2D1_Active,
Position := Abs2D1_Pos,
Velocity := Abs2D1_Vel,
Acceleration := Abs2D1_ACC,
Deceleration := Abs2D1_DEC,
Jerk := Abs2D1_Jerk,
CoordSystem := Abs2D1_Coord,
BufferMode := Abs2D1_Buffer,
TransitionMode := Abs2D1_Trans,
```

```
    TransitionParameter := Abs2D1_Param,
    AxesGroup := AxesGroup,
    Done=> Abs2D1_Done,
    Busy=> Abs2D1_Busy,
    Active=> Abs2D1_Active,
    CommandAborted=> Abs2D1_ComdAbort,
    Error=> Abs2D1_Err,
    ErrorID=> Abs2D1_ErrID);
END_IF
```

设定的转角限速减速角为 20°，停止角度为 60°。插补指令 1 和指令 2 的转角小于减速角，两条指令转角衔接处（见下图衔接点 1）速度未受到限制；插补指令 2 和指令 3 的转角大于减速角，小于停止角，两条指令转角衔接处（见下图衔接点 2）速度出现减速；插补指令 3 和指令 4 的转角大于停止角，两条指令转角衔接处（见下图衔接点 3）速度降为 0；

插补运动指令的合轴速度曲线如下所示：

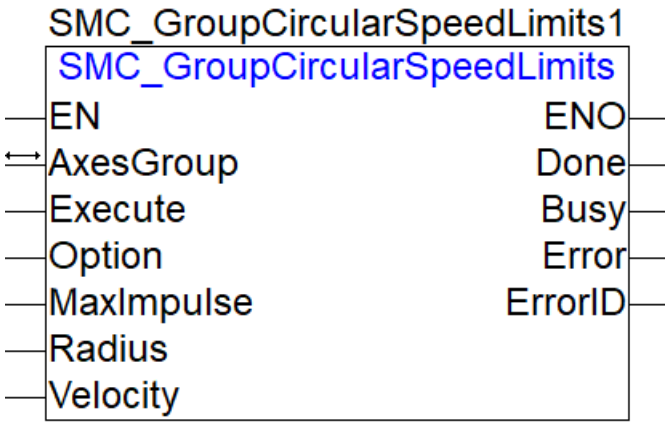


7.9.4 使用注意事项

- 调用该指令时，减速角需小于等于停止角。
- 该指令设置转角限速参数成功后，下降沿不能取消参数限制。用户需重新调用该指令设置参数限制。

7.10 SMC_GroupCircularSpeedLimits（轴组圆弧限速设置）

该指令设定轴组运动过程中的圆弧限速参数，减小圆弧转弯处的运动冲击。



7.10.1 参数

参数类型	名称	描述	能否空	默认值	范围	数据类型
IN_OUT	AxesGroup	轴组名	否	-	-	AXES_GROUP_REF
INPUT	Execute	上升沿触发	否	-	TRUE/FALSE	BOOL
	Option	参数设置方式： 0：通过设定圆弧转弯处最大冲击值的方式限速： 1：通过设定指定圆弧半径的最大速度的方式限速。	是	0	0/1	USINT
	MaxImpulse	Option 为 0 时生效： 最大冲击值，单位同加加速度	是	1E100	正值	LREAL
	Radius	Option 为 1 时生效： 指定的圆弧半径	是	1	正值	LREAL
	Velocity	Option 为 1 时生效： 指定圆弧半径的最大插补速度	是	1E30	正值	LREAL
OUTPUT	Done	完成	是	FALSE	TRUE/FALSE	BOOL
	Busy	忙标志	是	FALSE	TRUE/FALSE	BOOL
	Error	指令错误标志	是	FALSE	TRUE/FALSE	BOOL
	ErrorID	指令错误码	是	0	-	SMC_ERROR

7.10.2 功能

设定轴组运动过程中的圆弧限速参数，减小圆弧转弯处的运动冲击。

■ 指令功能详情

（1） 入参锁存

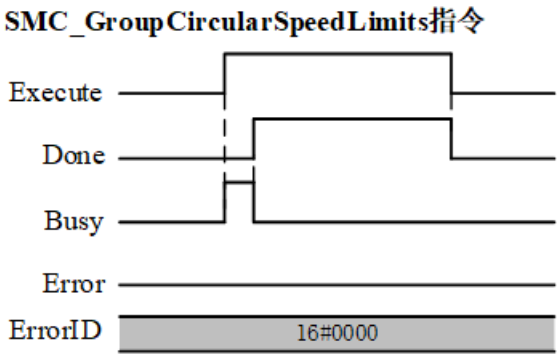
本指令上升沿触发有效。在 Execute 输入的上升沿时，将输入输出参数 AxesGroup 的合轴以及两个分轴的 ID、输入参数 Option、MaxImpulse、Radius 和 Velocity 值锁存，指令执行过程中，修改锁存的参数无效，直到再次上升沿触发本指令。

（2） 参数说明

- Option：圆弧限速参数设置方式，本指令支持 2 种参数设置方式。Option 为 0 时，通过设定圆弧转弯处最大冲击值的方式限速；Option 为 1 时，通过设定指定圆弧半径的最大速度的方式限速。
- 当参数设置方式 Option 为 0 时，MaxImpulse 参数有意义，为设置的最大冲击值。指令圆弧半径 Radius 和指令圆弧半径的最大插补速度 Velocity 无意义。
- 当参数设置方式 Option 为 1 时，指令圆弧半径 Radius 和指令圆弧半径的最大插补速度 Velocity 有意义，最大冲击值 MaxImpulse 参数无意义。
- MaxImpulse：最大冲击值，单位同加加速度，该参数在 Option 为 0 时生效。
- 当参数设置方式 Option 为 1 时，根据入参指定圆弧半径的最大插补速度 Velocity 和曲率 Curvature 重新计算 MaxImpulse。MaxImpulse =Curvature² * dSpeed³。Curvature=1/Radius。
- 输入的限速参数均为正值。

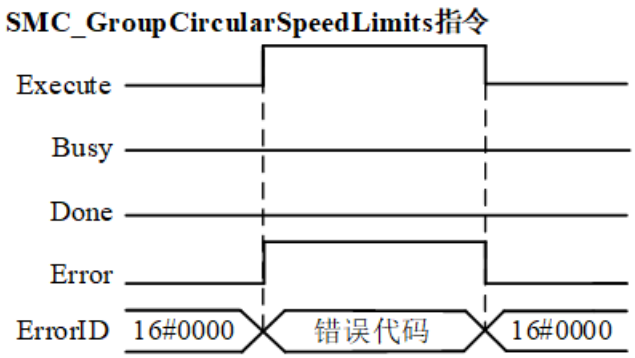
■ 时序图

（1） 正常时序图



（2） 异常错误发生时引脚时序

执行本指令时，若发生参数非法、分轴为轴类型为 0：Unused（未使用轴），轴组中各轴 ID 非法等异常，调用本指令失败，指令输出引脚 Error=True，错误代码 ErrorID 报出异常的错误值，查阅附录错误代码说明，获取错误发生原因。指令输入引脚 Execute 变为低电平时，所有输出引脚恢复为默认值。



- 当指令的异常情况解决后，需要重新上升沿触发 Execute，启动 SMC_GroupCircularLimits 功能块，完成圆弧限速参数限制设定。

7.10.3 示例

■ 动作示例

（1） 执行圆弧插补指令 SMC_MoveCircularRelative2D，该插补指令速度设置为 5000，加减速度均为 50000，加加速度为 500000，控制轴组运动。

（2） 执行 SMC_GroupCircularSpeedLimits 指令，设置圆弧限速参数。

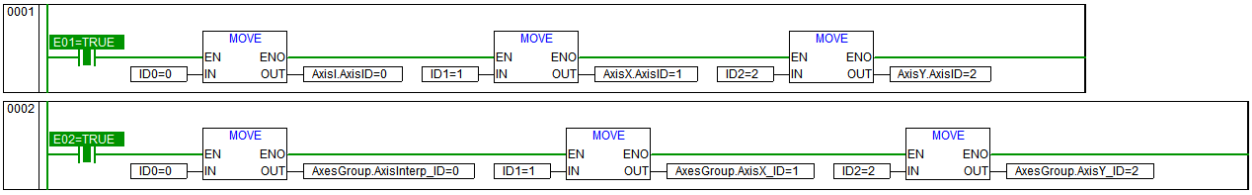
■ 梯形图

设定圆弧限速指令输入参数 Option 为 0，考虑到圆弧转弯处的运动冲击对设备安全的影响，MaxImpulse 为 10000。此时轴组中圆弧插补指令的加加速度受到限制，加加速度最大值为 10000。

主要变量说明

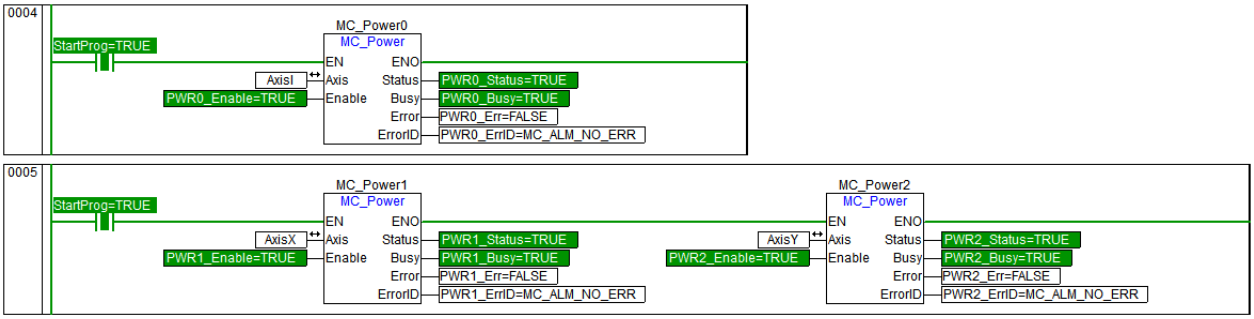
变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power1_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
Power0_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
GrpEnable1_Exe	BOOL	FALSE	轴组使能上升沿触发变量。
Circ2D1_Exe	BOOL	FALSE	圆弧插补指令的上升沿触发变量。
CirSpeed_Exe	BOOL	FALSE	圆弧限速参数指令的上升沿触发变量。

（1） 设定轴组中合轴和分轴的轴 ID。定义轴组时，首先需要定义轴组中的三个轴，将定义的三个轴的轴号 ID 与轴组中合轴和分轴的 ID 对应即可。见下述 0001 节和 0002 节。

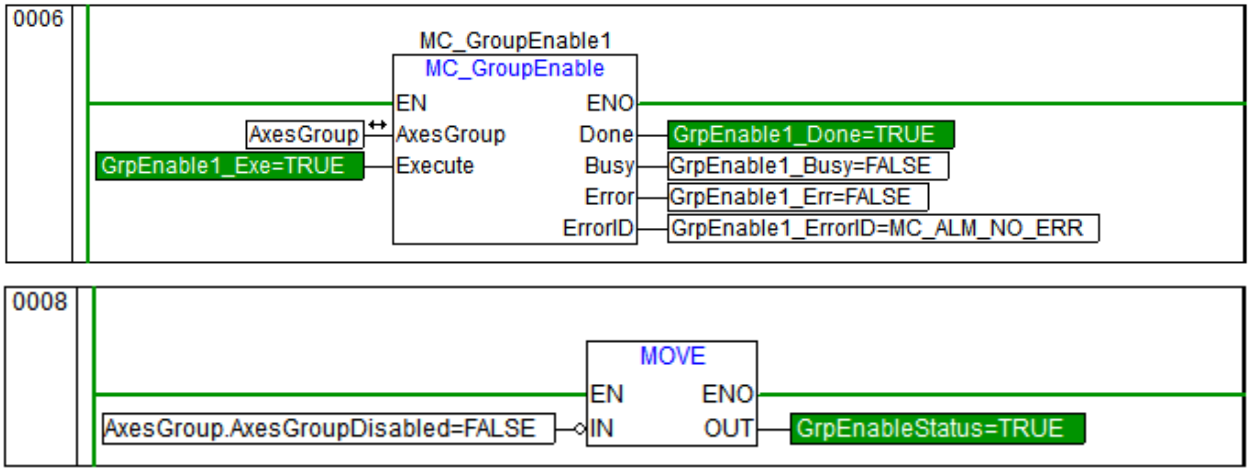


（2） 轴初始化。调用 SMC_AxisInit 指令分别完成轴组中三个轴的轴初始化配置，设置合轴为 0：Virtual，分轴类型为 50: EtherCAT_Position。SMC_AxisInit 指令使用方法参见其章节介绍。

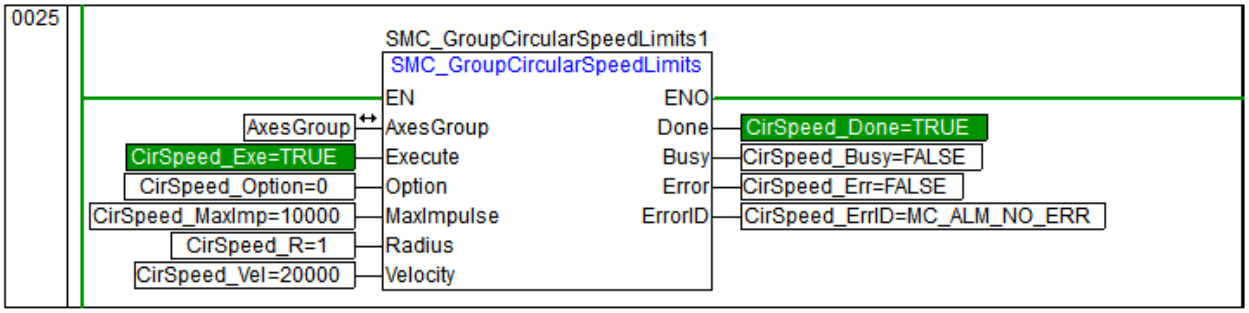
（3） 轴使能。轴组中各轴准备就绪（StartProg =TRUE）后，调用 MC_Power 对轴组的合轴和两个分轴进行轴使能。见 0004 ~ 0005 节。



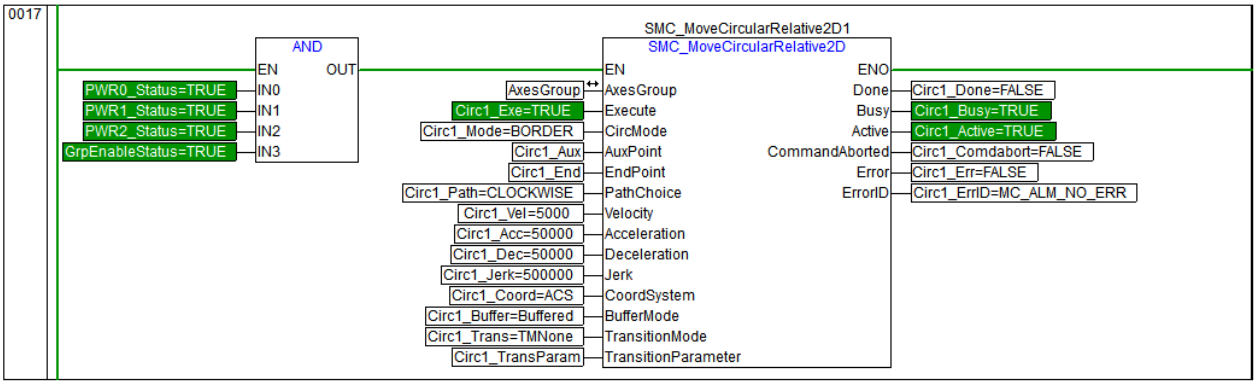
（4） 调用 MC_GroupEnable 对轴组使能。轴组变量中的 AxesGroupDisabled 为 FALSE，轴组使能成功，见 0006、0008 节。轴组使能成功后，将 AxesGroupDisabled 取反赋给 GrpEnableStatus 变量。



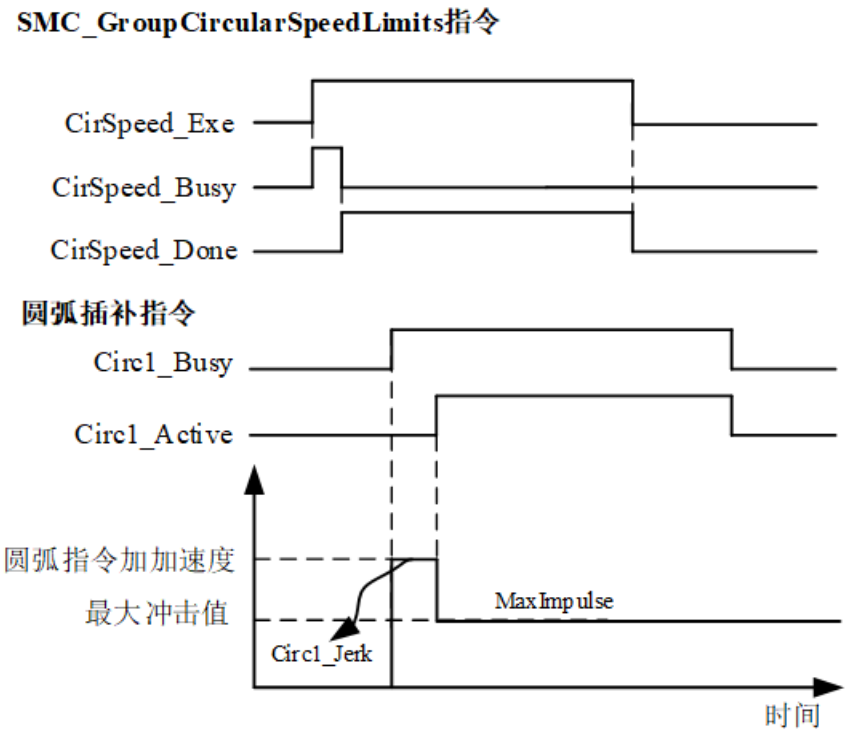
（5） 0025 节中，执行 SMC_GroupCircularSpeedLimits 指令，设置轴组圆弧限速参数的限制值，圆弧插补指令的实际加加速度 Jerk 为该设定的 MaxImpulse（10000）



（6） 0017 节中，执行圆弧插补指令。



参数设定后，圆弧插补指令加加速度参数受到限制情况如下：



■ ST 语言

执行 SMC_GroupCircularSpeedLimits 指令，设定输入参数 Option 为 1，圆弧半径为 1000，圆弧最大插补最大速度 Velocity 设定为 1000。计算 MaxImpulse =（1/1000）² * 1000³ = 1000。

执行圆弧插补指令 SMC_MoveCircularRelative2D，该圆弧插补指令的圆弧半径为 1000，指令速度设置为 5000，加减速度均为 50000，加加速度为 500000。轴组插补运动的速度和加加速度受到限制，轴组合轴的速度为 1000，加加速度为限制值 1000（MaxImpulse）。

主要变量说明

变量名称	变量类型	初始值	注释
AxesGroup	AXES_GROUP_REF	-	轴组名
AxisI	AXIS_REF	-	轴组合轴的轴变量
AxisX	AXIS_REF	-	轴组分轴 X 轴的轴变量
AxisY	AXIS_REF	-	轴组分轴 Y 轴的轴变量
StartProg	BOOL	FALSE	如果该变量为 TRUE，则轴组中的各轴以处于准备就绪状态，可以执行轴使能工作。
Power0_Status	BOOL	FALSE	分轴 X 使能成功的变量。分轴 X 使能成功时，该变量变为 TRUE。
Power1_Status	BOOL	FALSE	分轴 Y 使能成功的变量。分轴 Y 使能成功时，该变量变为 TRUE。
Power2_Status	BOOL	FALSE	合轴使能成功的变量。合轴使能成功时，该变量变为 TRUE。
InitFlag	BOOL	FALSE	如果该变量为 TRUE，则初始化参数设置完成。
GrpEnableStatus	BOOL	TRUE	该变量为 TRUE 时，轴组使能。
GrpEnable1_Exe	BOOL	FALSE	轴组使能上升沿触发变量。
Circ2D1_Exe	BOOL	FALSE	圆弧插补指令的上升沿触发变量。
CirSpeed_Exe	BOOL	FALSE	圆弧限速参数指令的上升沿触发变量。

（1） 设定轴组中合轴和分轴的轴 ID。设定所定义的合轴 AxisI、分轴 AxisX 和 AxisY 的轴 ID。

```
IF Switch1 THEN
   AxisI.AxisID:=0;
   AxisX.AxisID:=1;
   AxisY.AxisID:=2;
   AxesGroup.AxisInterp_ID:=AxisI.AxisID;
   AxesGroup.AxisX_ID:=AxisX.AxisID;
   AxesGroup.AxisY_ID:=AxisY.AxisID;
END_IF
```

（2） 设置指令运动参数

```
IF FALSE = InitFlag THEN
   (*圆弧插补指令 3 的参数设置。未设置的输入参数采用默认值*)
   Circ1_Mode:= BORDER;
   Circ1_Aux [0] :=1000;
   Circ1_Aux [1] :=1000;
   Circ1_End [0] :=2000;
   Circ1_End [1] :=0;
   Circ1_Path:= CLOCKWISE;
   Circ1_Vel:=2000;
```

```
    Circ1_ACC:=5000;
    Circ1_DEC:=5000;
    Circ1_Jerk:=50000;
    Circ1_Buffer:=Buffered;
    CirSpeed_Option:=1;
    CirSpeed_R:=1000;
    CirSpeed_Vel:=1000;
    InitFlag:=TRUE;
END_IF
```

（3） 调用 SMC_AxisInit 指令完成轴组中三个轴的轴初始化配置。轴组合轴轴类型设置为 0：Virtual；轴组分轴 AxisX/AxisY 轴类型均设置为 50：EtherCAT_Position。SMC_AxisInit 指令使用方法参见其指令章节介绍。

（4） 执行 MC_Power 指令完成轴组中三个轴的轴使能。

```
IF StartPro THEN
    MC_Power0(                          (*合轴使能*)
    Enable := PWR0_Enable,
    Axis := AxisI,
    Status=> PWR0_Status,
    Busy=> PWR0_Busy,
    Error=> PWR0_Err,
    ErrorID=> PWR0_ErrID);
    MC_Power1(                          (*分轴 AxisX 使能*)
    Enable := PWR1_Enable,
    Axis := AxisX,
    Status=> PWR1_Status,
    Busy=> PWR1_Busy,
    Error=> PWR1_Err,
    ErrorID=> PWR1_ErrID);
    MC_Power2(                          (*分轴 AxisY 使能*)
    Enable := PWR2_Enable,
    Axis := AxisY,
    Status=> PWR2_Status,
    Busy=> PWR2_Busy,
    Error=> PWR2_Err,
```

```
   ErrorID=> PWR2_ErrID);
END_IF
```

（5） 执行轴组使能

```
MC_GroupEnable1(
Execute := GrpEnable_Exe,
AxesGroup := AxesGroup,
Done=> GrpEnable_Done,
Busy=> GrpEnable_Busy,
Error=> GrpEnable_Err,
ErrorID=> GrpEnable_ErrorID);
GrpEnableStatus:= AxesGroup. AxesGroupDisabled;
```

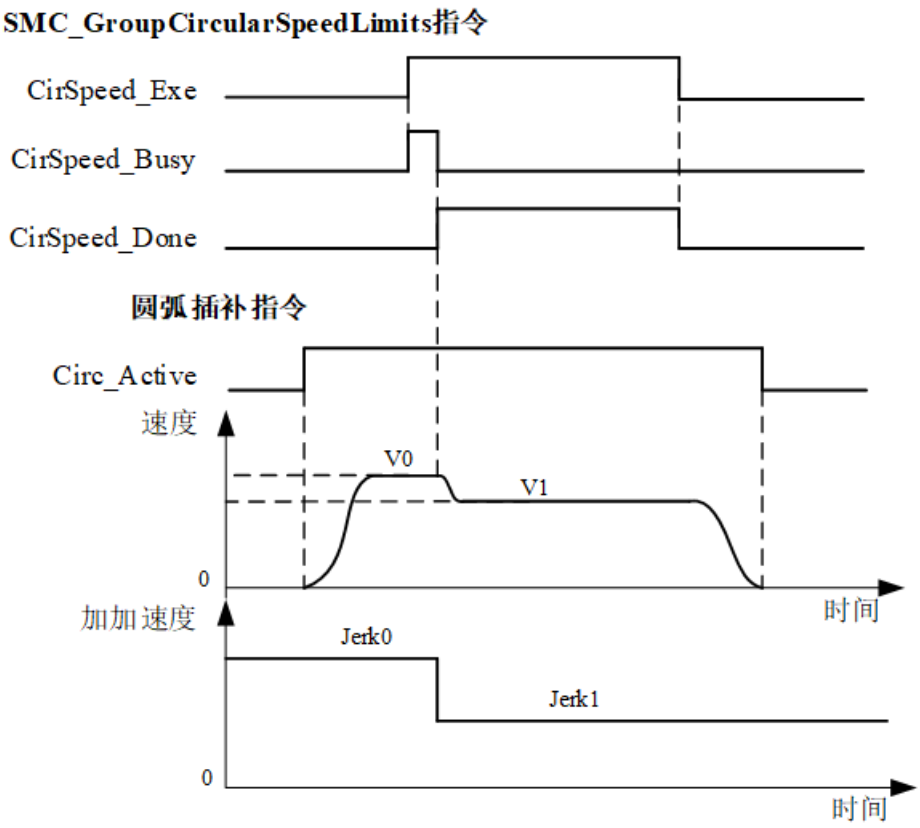
（6） 执行插补运动指令

```
IF PWR0_Status  AND PWR1_Status  AND PWR2_Status  AND GrpEnableStatus THEN
   SMC_MoveCircularRelative2D1(              (*插补指令 3*)
   Execute := Rel2D1_Active,
   CircMode := Circ1_Mode ,
   AuxPoint := Circ1_Aux,
   EndPoint := Circ1_End,
   PathChoice:= Circ1_Path,
   Velocity := Circ1_Vel,
   Acceleration := Circ1_ACC,
   Deceleration := Circ1_DEC,
   Jerk := Circ1_Jerk,
   CoordSystem := Circ1_Coord,
   BufferMode := Circ1_Buffer,
   TransitionMode := Circ1_ Trans,
   TransitionParameter := Circ1_Param,
   AxesGroup := AxesGroup,
   Done=> Circ1_Done,
   Busy=> Circ1_Busy,
   Active=> Circ1_Active,
   CommandAborted=> Circ1_Abort,
   Error=> Circ1_Err,
   ErrorID=> Circ1_ErrID);
```

（7） 执行 SMC_GroupCircularSpeedLimits 指令，设置轴组圆弧限速参数的限制值。

```
SMC_GroupCircularSpeedLimits1(
Execute := CirSpeed_Exe,
Option := CirSpeed_Option,
MaxImpulse := CirSpeed_MaxImp,
Radius := CirSpeed_R,
Velocity := CirSpeed_Vel,
AxesGroup := AxesGroup,
Done=> CirSpeed_Done
Busy=> CirSpeed_Busy,
Error=> CirSpeed_Err,
ErrorID=> CirSpeed_ErrID);
```

（8） 圆弧限速参数设定后，插补指令运动参数受到限制。插补指令速度和加加速度曲线如下所示：



7.10.4 使用注意事项

- 调用该指令时，输入参数 MaxImpulse、Radius 和 Velocity 为正数。
- 该指令设置最大参数成功后，下降沿不能取消参数限制。用户需重新调用该指令设置参数限制值。

第8章　附录

8.1　错误代码说明

功能块错误代码说明

错误码	错误名称	错误说明
0	MC_ALM_NO_ERR	无错误
2000	MC_ERR_AXIS_TYPE_UNUSED	轴类型为未使用轴/轴组中有轴类型为未使用轴
2001	MC_ERR_AXISID_INVALID	轴 ID 非法/轴组中有轴 ID 非法
2002	MC_ERR_AXISID_IS_REPEATED	有轴号相同/轴组中有轴轴号相同
2003	MC_ERR_AXIS_TYPE_NOT_SUPPORTED	不支持的轴类型/轴组中有轴为不支持的轴类型
2004	MC_ERR_SLAVEID_INVALID	从站 ID 非法/轴组中存在从站 ID 非法
2005	MC_ERR_SLAVE_OFFLINE	从站离线/轴组中存在从站 ID 离线
2006	MC_ERR_SLAVE_FAULT	从站故障/轴组中存在从站 ID 故障
2007	MC_ERR_SYSTEM_ERR	发生系统错误
2008	MC_ERR_AXIS_ERR	轴有错误/轴组中轴有错误
2009	MC_ERR_AXIS_MOTION_IS_REJECTED_IN_STOPPING	轴 Stopping 状态下拒绝运动指令
2010	MC_ERR_AXIS_STATE_ERROR_STOP	轴错误停止/轴组中有轴发生错误停止
2011	MC_ERR_AXIS_DISABLE	轴未使能/轴组中有轴未使能
2012	MC_ERR_PDO_SETTING_MISSING	PDO 未配置
2013	MC_ERR_AXIS_MOTION_BUFFER_FULL	轴运动缓存满
2014	MC_ERR_VELOCITY_INVALID	目标速度参数非法
2015	MC_ERR_ACCELERATION_INVALID	加速度参数非法
2016	MC_ERR_DECELERATION_INVALID	减速度参数非法
2017	MC_ERR_JERK_INVALIDE	加加速度参数非法
2018	MC_ERR_BUFFERMODE_INVALID	BufferMode 参数非法
2019	MC_ERR_DIRECTION_INVALID	方向参数非法
2020	MC_ERR_SDO_READ_ERR	EtherCAT SDO 读取错误
2021	MC_ERR_SDO_WRITE_ERR	EtherCAT SDO 写入错误
2022	MC_ERR_AXIS_NOT_ASSOCIATED_WITH_PHYSICAL_DEVICE	该轴未关联实际的物理设备
2200	MC_ERR_AXES_GROUP_DISABLED	轴组未使能
2201	MC_ERR_RELATIVE_INTERPOLATION_DIST0	相对插补指令运动距离为 0 参数检查
2202	MC_ERR_COORD_SYSTEM_INVALID	坐标系非法
2203	MC_ERR_TRANSITION_MODE_INVALID	过渡曲线选择参数非法

2204	MC_ERR_AXES_GROUP_ENABLED_FAILURE	轴组使能失败
2950	MC_ERR_PDO_UNMAPPED_FOR_STATUS_WORD	总线驱动器 PDO 未映射 Statusword
2951	MC_ERR_PDO_UNMAPPED_FOR_ACTUAL_POS	总线驱动器 PDO 未映射 Position actual value
2952	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_STATUS	总线驱动器 PDO 未映射 Touch probe status
2953	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_POS_POS1	总线驱动器 PDO 未映射 Touch probe pos1 pos value
2954	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_NEG_POS1	总线驱动器 PDO 未映射 Touch probe pos1 neg value
2955	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_POS_POS2	总线驱动器 PDO 未映射 Touch probe pos2 pos value
2956	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_NEG_POS2	总线驱动器 PDO 未映射 Touch probe pos2 neg value
2957	MC_ERR_PDO_UNMAPPED_FOR_DIGITAL_INPUTS	总线驱动器 PDO 未映射 Digital inputs
2958	MC_ERR_PDO_UNMAPPED_FOR_OPERATE_MODE_DISPLAY	总线驱动器 PDO 未映射 Modes of operation display
2959	MC_ERR_PDO_UNMAPPED_FOR_ACTUAL_VELOCITY	总线驱动器 PDO 未映射 Velocity actual value
2960	MC_ERR_PDO_UNMAPPED_FOR_ACTUAL_TORQUE	总线驱动器 PDO 未映射 Torque actual value
2961	MC_ERR_PDO_UNMAPPED_FOR_ERROR_CODE	总线驱动器 PDO 未映射 Error code
2962	MC_ERR_PDO_UNMAPPED_FOR_CONTROL_WORD	总线驱动器 PDO 未映射 Controlword
2963	MC_ERR_PDO_UNMAPPED_FOR_OPERATE_MODE	总线驱动器 PDO 未映射 Modes of operation
2964	MC_ERR_PDO_UNMAPPED_FOR_TARGET_POS	总线驱动器 PDO 未映射 Target position
2965	MC_ERR_PDO_UNMAPPED_FOR_TARGET_VELOCITY	总线驱动器 PDO 未映射 Target velocity
2966	MC_ERR_PDO_UNMAPPED_FOR_TARGET_TORQUE	总线驱动器 PDO 未映射 Target torque
2967	MC_ERR_PDO_UNMAPPED_FOR_TOUCH_PROBE_FUNC	总线驱动器 PDO 未映射 Touch probe function
2968	MC_ERR_PDO_UNMAPPED_FOR_DIGITAL_OUTPUTS	总线驱动器 PDO 未映射 Digital outputs
2969	MC_ERR_PDO_UNMAPPED_FOR_MAX_PROFILE_VELOCITY	总线驱动器 PDO 未映射 Max profile velocity
2970	MC_ERR_PDO_UNMAPPED_FOR_MAX_MOTOR_SPEED	总线驱动器 PDO 未映射 Max motor speed
2971	MC_ERR_PDO_UNMAPPED_FOR_MAX_TORQUE	总线驱动器 PDO 未映射 Max torque
2972	MC_ERR_PDO_UNMAPPED_FOR_POS_TORQUE_LIMIT	总线驱动器 PDO 未映射 Positive torque limit value
2973	MC_ERR_PDO_UNMAPPED_FOR_NEG_TORQUE_LIMIT	总线驱动器 PDO 未映射 Negative torque limit value
3000	MC_ERR_AXIS_INIT_UNITS_PARA_CONFIG_ERR	用户单位相关参数设置错误
3001	MC_ERR_AXIS_INIT_POS_LIMIT_SWITCH_CFG_ERR	正向限位开关入参配置错误
3002	MC_ERR_AXIS_INIT_NEG_LIMIT_SWITCH_CFG_ERR	负向限位开关入参配置错误
3003	MC_ERR_AXIS_INIT_HOME_IN_SWITCH_CFG_ERR	原点回归开关入参配置错误

3004	MC_ERR_AXIS_INIT_HMSW_SWITCH_CHANNEL_NUM_ERR	硬件开关通道号配置错误
3005	MC_ERR_AXIS_INIT_HMSW_SWITCH_CHANNEL_TYPE_ERR	硬件限位开关类型错误
3006	MC_ERR_AXIS_INIT_POS_LIMIT_SWITCH_NUM_ERR	正向限位开关通道号错误
3007	MC_ERR_AXIS_INIT_NEG_LIMIT_SWITCH_NUM_ERR	负向限位开关通道号错误
3008	MC_ERR_AXIS_INIT_HOME_IN_SWITCH_NUM_ERR	原点回归开关通道号错误
3009	MC_ERR_AXIS_INIT_SET_USER_UNITS_ERR	设置用户单位出错
3010	MC_ERR_AXIS_INIT_AXIS_TYPE_CFG_TIMEOUT	轴类型配置超时
3011	MC_ERR_AXIS_INIT_SET_AXIS_TYPE_REPEAT	重复设置相同轴类型
3012	MC_ERR_AXIS_INIT_SET_AXIS_TYPE_TIMING_ERR	设置轴类型时该轴已经有运动控制指令
3013	MC_ERR_AXIS_INIT_SET_OPERATION_MODE_ERR	设置轴操作模式失败
3014	MC_ERR_AXIS_INIT_NOT_ECAT_ENCODER_AXIS_TYPE	当前轴不是 EtherCAT_Encoder 轴类型
3015	MC_ERR_NEG_LIM_CHAN_AND_POS_LIM_CHAN_REUSED	正负硬限位通道号配置重复
3016	MC_ERR_DRIVER_CHAN_CONFLICT_WITH_COMM_DI_CHAN	驱动器本体 DI 通道号转换为实际的地址与物理 DI 通道地址冲突
3017	MC_ERR_NEG_LIM_CHAN_CONFLICT_WITH_POS_LIM	设置的正限位通道号与负限位通道号冲突
3018	MC_ERR_POS_LIM_CHAN_CONFLICT_WITH_NEG_LIM	设置的负限位通道号与正限位通道号冲突
3030	MC_ERR_AXIS_POWER_MOD_DISP_TIMEOUT	轴类型配置超时
3040	MC_ERR_HOME_APPROACH_VELOCITY_INVALID	原点回归 APPROACH VELOCITY 设置非法
3041	MC_ERR_HOME_CAPTURE_CHL_INVALID	原点回归捕获通道号非法
3042	MC_ERR_HOME_CAPTURE_CHL_BUSY	原点回归捕获通道被占用
3043	MC_ERR_HOME_MODE_Z_NOT_SUPPORTED	原点回归不支持 Z 相模式
3044	MC_ERR_HOME_HOME_MODE_INVALID	入参原点回归模式无效
3045	MC_ERR_HOME_AXIS_TYPE_FOR_Z_ERR	配置了 Z 相捕获的模式，但不是 EtherCAT 伺服轴
3046	MC_ERR_HOME_HOME_CHL_CONFIG_ERR	原点开关通道配置错误
3047	MC_ERR_HOME_POT_CHL_CONFIG_ERR	正限位开关通道配置错误
3048	MC_ERR_HOME_NOT_CHL_CONFIG_ERR	负限位开关通道配置错误
3049	MC_ERR_HOME_ALL_DI_CONFIG_ERR	高级原点回归 DI 通道配置错误
3090	MC_ERR_MOVE_RELATIVE_DIST_INVALID	单轴相对运动距离非法
3100	MC_ERR_MCMD_NOT_SYNC_MOVE_VELOCITY	当前指令非 SyncMoveVelocity 指令
3101	MC_ERR_SYNC_MOVE_VELOCITY_CMD_DATA_NULL	SyncMoveVelocity 指令数据为空
3120	MC_ERR_SET_JERKM_MODE_INVALID	加减速类型设置错误
3130	MC_ERR_TORQUE_CONTROL_TORQUE_RAMP_INVALID	扭矩控制功能块入参扭矩变化率检查无效
3131	MC_ERR_TORQUE_CONTROL_TORQUE_INVALID	扭矩控制功能块入参目标扭矩检查无效
3132	MC_ERR_TORQUE_CONTROL_VELOCITY_INVALID	扭矩控制功能块入参速度检查无效

3133	MC_ERR_TORQUE_CONTROL_MAX_VELOCITY_CONFIG_ERR	扭矩控制功能块最大速度配置错误
3160	MC_ERR_SET_OVERRIDE_VEL_FACTOR_INVALID	速度比例因子参数错误
3161	MC_ERR_SET_OVERRIDE_ACC_FACTOR_INVALID	加减速度比例因子参数错误
3162	MC_ERR_SET_OVERRIDE_JERK_FACTOR_INVALID	加加速度比例因子参数错误
3170	MC_ERR_MAX_VELOCITY_INVALID	最大限制速度参数非法
3171	MC_ERR_MAX_ACCELERATION_INVALID	最大限制加速度参数非法
3172	MC_ERR_MAX_DECELERATION_INVALID	最大限制减速度参数非法
3173	MC_ERR_MAX_JERK_INVALID	最大限制加加速度参数非法
3180	MC_ERR_READ_DIGITAL_INPUT_NUM_INVALID	读取数字量输入编号参数非法
3190	MC_ERR_READ_LIMIT_STATUS_SOFT_LIMIT_PARA_ERR	软限位相关参数错误
3191	MC_ERR_READ_LIMIT_STATUS_EACH_STATUS_CONFLICT	限位状态冲突错误
3210	MC_ERR_RESET_CMD_BACK_INVALID	复位函数返回错误
3211	MC_ERR_RESET_AXIS_TIME_OUT_ERR	复位轴错误超时失败
3212	MC_ERR_RESET_AXIS_WAIT_TIME_OUT_ERR	复位等待超时
3220	MC_ERR_SET_TORQUE_LIMIT_POS_VALUE_INVALID	正向限制值非法
3221	MC_ERR_SET_TORQUE_LIMIT_NEG_VALUE_INVALID	负向限制值非法
3222	MC_ERR_SET_TORQUE_LIMIT_SLAVE_INVALID	从站号非法
3223	MC_ERR_SDO_WRITE_POS_TORQUE_LIMITERR	写入正向限制值 SDO 出错
3224	MC_ERR_SDO_WRITE_NEG_TORQUE_LIMITERR	写入负向限制值 SDO 出错
3225	MC_ERR_SDO_WRITE_POS_TORQUE_LIMITOVERRUN	写入正向限制值超限
3226	MC_ERR_SDO_WRITE_NEG_TORQUE_LIMITOVERRUN	写入负向限制值超限
3230	MC_ERR_TOUCH_PROBE_LATCH_MODE_INVALID	位置锁存功能块入参锁存模式配置无效
3231	MC_ERR_TOUCH_PROBE_LATCH_ID_INVALID	位置锁存功能块入参锁存 ID 配置无效
3232	MC_ERR_TOUCH_PROBE_TRIGGER_SIGNAL_INVALID	位置锁存功能块入参锁存触发信号配置无效
3233	MC_ERR_TOUCH_PROBE_TRIGGER_EDGE_INVALID	位置锁存功能块入参锁存触发边沿配置无效
3234	MC_ERR_TOUCH_PROBE_WINDOW_CONFIG_ERR_1	窗口配置错误 1，有限行程时，First 位置设置大于 Last 位置
3235	MC_ERR_TOUCH_PROBE_WINDOW_CONFIG_ERR_2	窗口配置错误 2，循环行程 1 时，窗口位置设置不合理
3236	MC_ERR_TOUCH_PROBE_WINDOW_CONFIG_ERR_3	窗口配置错误 3，循环行程 2 时，窗口位置设置不合理
3237	MC_ERR_TOUCH_PROBE_IS_REUSED	锁存通道重复使用错误
3238	MC_ERR_TOUCH_PROBE_SET_USED_ERR	设置通道正在使用时发生错误
3239	MC_ERR_TOUCH_PROBE_SYSTEM_ERR	位置捕获发生系统内部错误
3240	MC_ERR_TOUCH_PROBE_PARA_COMBINE_ERR	配置参数组合出错

3241	MC_ERR_TOUCH_PROBE_CHANNEL_INNER_USED	该通道被内部使用或通道使用状态未知
3242	MC_ERR_TOUCH_PROBE_FB_IS_USING	该功能块正在被调用
3243	MC_ERR_TOUCH_PROBE_PDO_CONFIG_CHECK_ERR	位置锁存配置时 PDO 映射检查发生错误
5000	MC_ERR_GEAR_RATIO_DENOMINATOR_INVALID	齿轮比分母为 0
5001	MC_ERR_GEAR_MASTER_SOURCE_INVALID	主轴位置源参数非法
5040	MC_ERR_CAM_POINTER_TO_ARRARYPOS_ISNULL	凸轮位置数组的指针为空
5041	MC_ERR_CAM_TABLE_POSNUM_INVALID	凸轮位置数组的位置个数(小于 3)非法
5042	MC_ERR_CAM_MASTER_SOURCE_INVALID	电子凸轮主轴位置源参数非法
5043	MC_ERR_CAM_SLAVESCALE_INVALID	电子凸轮从轴位置数据比例因子非法
5044	MC_ERR_CAM_DI_STARTPOSITION_INVALID	电子凸轮 DI 触发启动位置非法
5045	MC_ERR_CAM_START_MODE_INVALID	电子凸轮启动模式错误
5046	MC_ERR_CAM_SLAVE_STANDIN_POSITION_LIMIT1	电子凸轮开始时从轴处于限位状态
5047	MC_ERR_CAM_ARRARYPOS_ISNOT_MONOINCREASE	凸轮数组不是单调递增数组
5048	MC_ERR_CAM_MASTER_POS_MONODECREASE	主轴位置数据变为递减
5049	MC_ERR_CAM_STARTMODE_INVALID	凸轮启动模式错误
5050	MC_ERR_CAM_CAPTURE_CHL_OVERLIMIT	凸轮捕捉通道超过上限
5051	MC_ERR_CAM_DI_CHL_OVERLIMIT	凸轮 DI 通道超过上限
5052	MC_ERR_CAM_CAMTABLE_CURVETYPE_INVALID	凸轮曲线插值方式非法
5053	MC_ERR_CAM_TABLE_CHANGED_ERR_IN_RUNNING	凸轮表运行过程中被篡改为非法值
5100	MC_ERR_MOVELINK_MASTER_DIST_INVALID	追剪主轴运动距离参数非法
5101	MC_ERR_MOVELINK_MASTER_ACCDIST_INVALID	追剪从轴加速阶段主轴移动的距离参数非法
5102	MC_ERR_MOVELINK_MASTER_DECDIST_INVALID	追剪从轴减速阶段主轴移动的距离参数非法
5103	MC_ERR_MOVELINK_S_RATIO_INVALID	追剪 S 形加减速占比参数非法
5104	MC_ERR_MOVELINK_START_MODE_INVALID	追剪启动模式错误
5105	MC_ERR_MOVELINK_MASTER_SOURCE_INVALID	追剪主轴位置源参数非法
5106	MC_ERR_MOVELINK_DI_STARTPOPITION_INVALID	追剪 DI 触发启动位置非法
5107	MC_ERR_MOVELINK_SLAVE_STANDIN_POS_LIMIT	追剪运动开始时从轴处于限位状态
5108	MC_ERR_MOVELINK_START_MODE_INVALID1	追剪运动启动模式错误（内部）
5109	MC_ERR_MOVELINK_CAPTURE_CHL_OVERLIMIT	追剪捕捉通道超过上限
5110	MC_ERR_MOVELINK_DI_CHL_OVERLIMIT	追剪 DI 通道超过上限
5111	MC_ERR_MOVELINK_S_CURVE_DIST_INVALID	追剪 S 形曲线加减速距离非法
5112	MC_ERR_MOVELINK_DIST_PARAM_INVALID	追剪距离参数设置非法
5113	MC_ERR_MOVELINK_MASTER_ADD_DIST_INVALID	飞剪叠加运动主轴完成和剩余距离非法
5150	MC_ERR_GANTRY_AXIS_NUM_ERR	龙门同步轴组个数应大于等于 2 且小于等于 8

5151	MC_ERR_GANTRY_AXISARRY_OR_DICHLARRT_IS_NULL	龙门同步相关函数中入参轴号数组或 DI 通道数组为空
5152	MC_ERR_GANTRY_AXIS_TYPES_ARE_DIFFERENT	龙门同步轴组内轴类型不相同
5153	MC_ERR_GANTRY_AXIS_TYPE_SET_ERR	龙门同步轴类型错误
5154	MC_ERR_GANTRY_INIT_INPUT_DICHL_REPEATED	龙门同步初始化中使用的原点开关通道重复
5155	MC_ERR_GANTRY_INIT_HOME_MODE_SET_ERR	龙门同步初始化原点回归模式设置错误，不支持回归模式 33、34、35
5156	MC_ERR_GANTRY_INIT_AXIS_POS_FE_OVERRIDE	龙门同步初始化回原点完成后轴间偏差过大
5157	MC_ERR_GANTRY_INIT_INPUT_PARA_POSERR_SET_ERR	龙门同步初始化输入参数 dPosErr 小于 0
5158	MC_ERR_GANTRY_AXIS_NOT_REPMODE_0	龙门同步轴行程模式不是有限行程
5159	MC_ERR_GANTRY_INIT_NOT_ECAT_AXIS_FOR_Z	龙门同步初始化中用到 Z 相捕获但轴不是 EtherCAT 总线轴
6010	MC_ERR_CIRC_INTERP_MODE_INVALID	不支持的圆弧插补模式
6011	MC_ERR_CIRC_MODE_INVALID	圆弧模式非法
6012	MC_ERR_CIRC_PATHCHOICE_INVALID	圆弧运动方向非法
6020	MC_ERR_CIRC_SPEEDLIMITS_OPTION_INVALID	圆弧限速参数设置方式值非法
6021	MC_ERR_CIRC_SPEEDLIMITS_MAXIMPULSE_INVALID	圆弧限速最大冲击值非法
6022	MC_ERR_CIRC_SPEEDLIMITS_RADIUS_INVALID	圆弧限速指定圆弧半径值非法
6023	MC_ERR_CIRC_SPEEDLIMITS_VELOCITY_INVALID	圆弧限速指定圆弧半径的最大插补速度非法
6030	MC_ERR_CORNER_SPEEDLIMITS_DECANGLE_INVALID	转角限速减速角参数非法
6031	MC_ERR_CORNER_SPEEDLIMITS_STOPANGLE_INVALID	转角限速停止角参数非法

8.2 缓冲模式说明

8.2.1 缓冲模式

定义缓存模式变量类型为 MC_BUFFER_MODE。

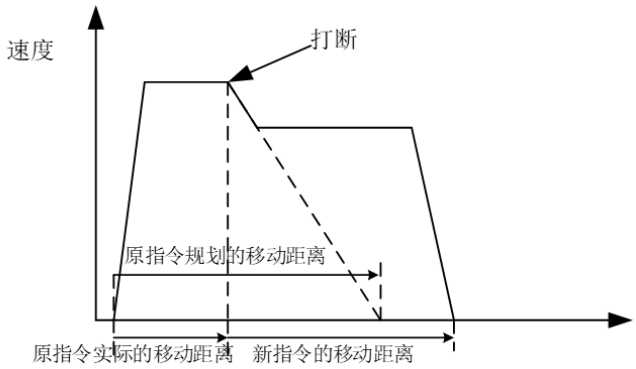
枚举值	MC_BUFFER_MODE	说明
0	Aborting	打断。打断当前正在运动的指令，立即执行本运动指令。
1	Buffered	缓存。当前的运动指令执行完成后，执行本运动指令。
2	BlendingLow	以相邻指令间的低速融合。
3	BlendingPrevious	以上一条指令的速度融合。

4	BlendingNext	以下一条指令的速度融合。
5	BlendingHigh	以相邻指令间的高速融合。

8.2.1.1 打断

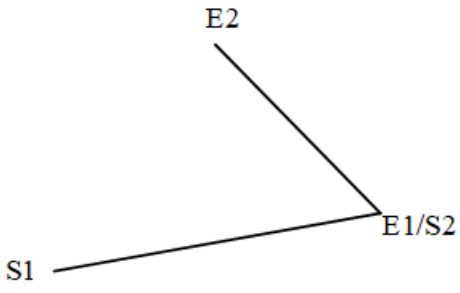
仅单轴运动指令支持打断功能，对于插补指令，不支持打断功能。

若触发的单轴运动指令的缓存模式为 Aborting(打断)，则该指令可变更正在执行的指令的动作，正在执行的运动指令被打断。此时，轴执行触发的运动指令，Position（目标位置）、Velocity（目标速度）、Acceleration（加速度）、Deceleration（减速度）、Jerk（加加速度）等参数被变更。示例动作如下图所示。



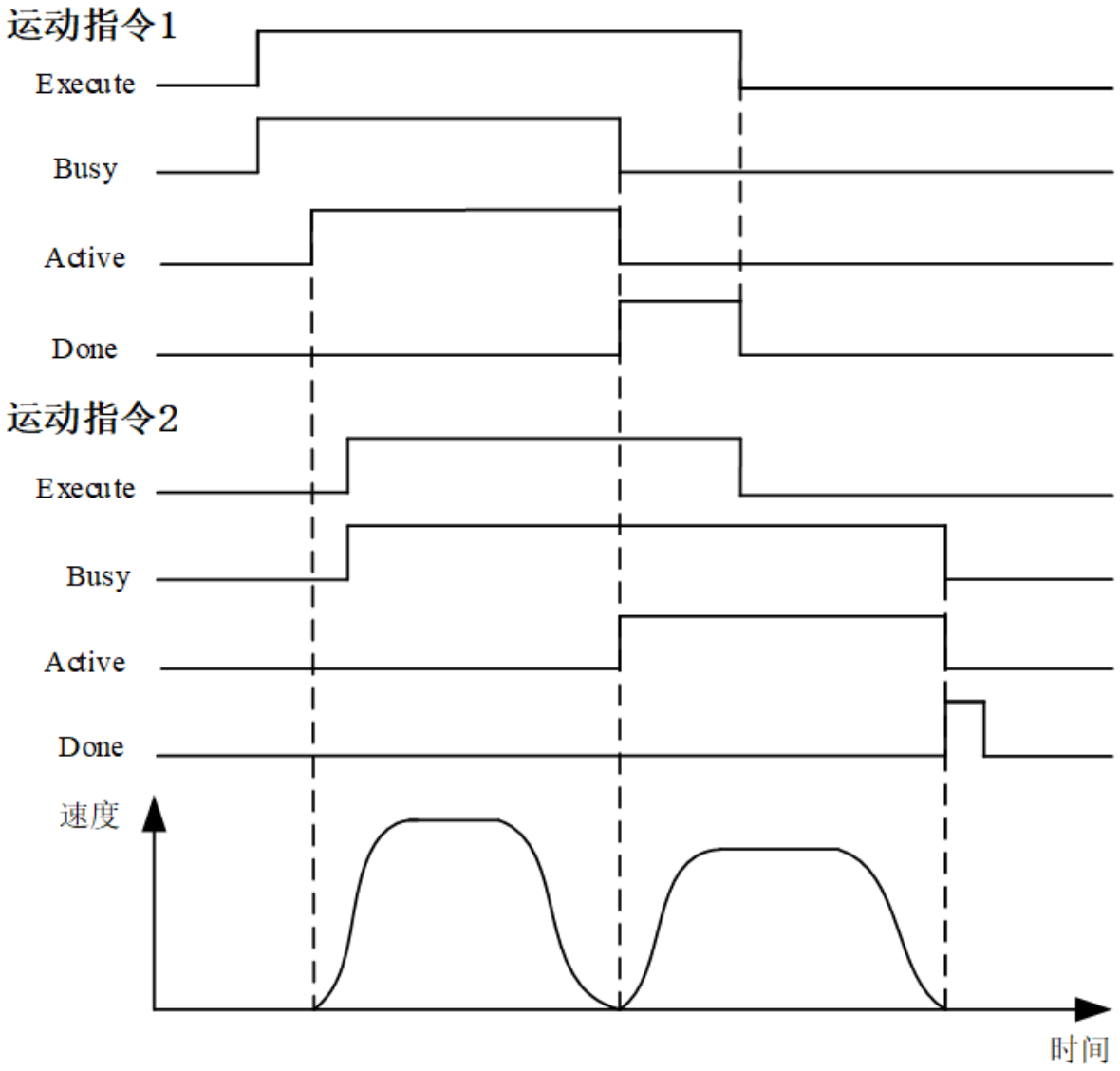
8.2.1.2 缓存

当前正在运动的指令执行完成后，执行下一条指令。下述图分别为缓存模式的运动示意图：



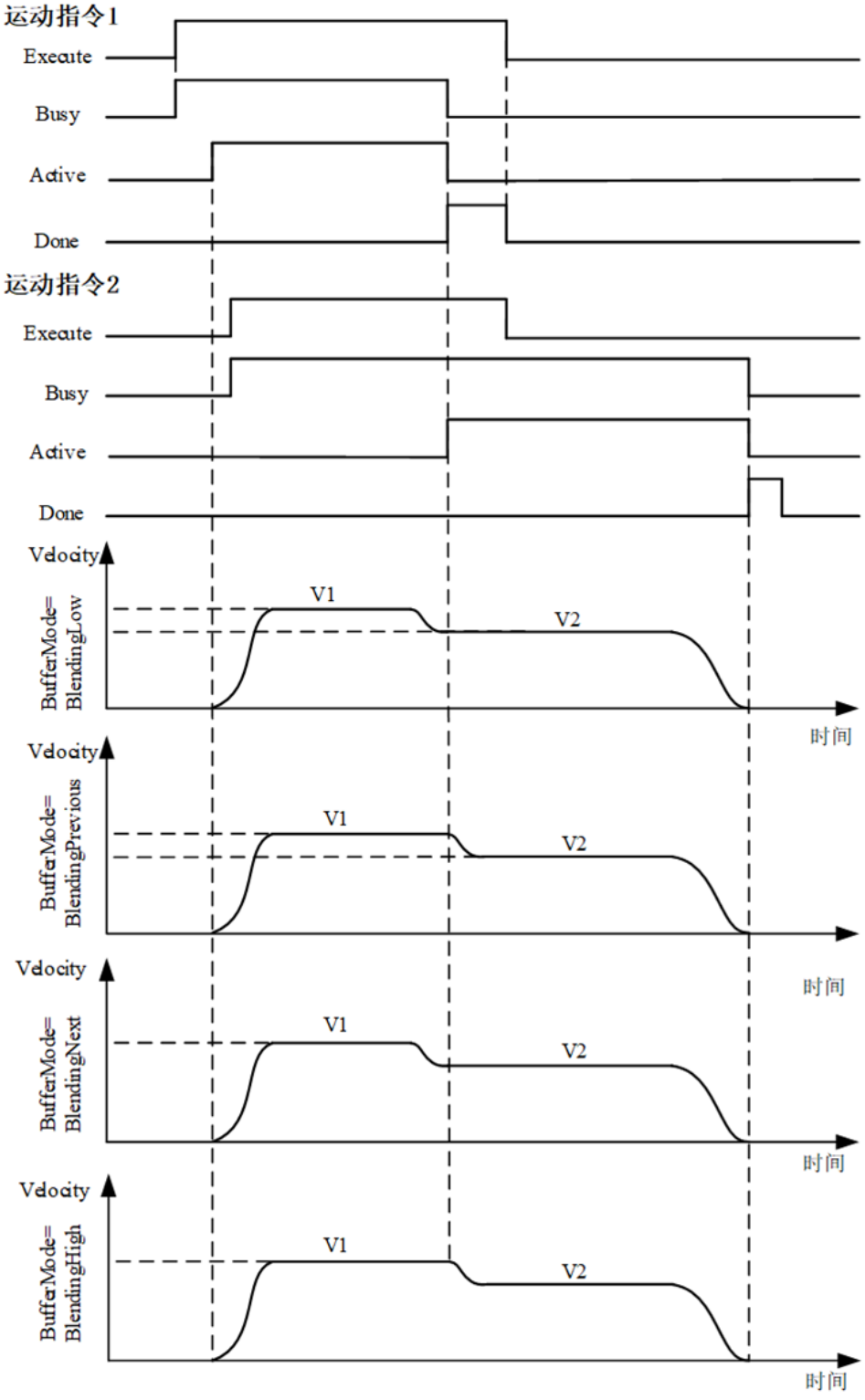
- S1:前一条运动指令的起点位置
- E1 : 前一条运动指令的目标位置

- S2:后一条运动指令的起点位置
- E2 : 后一条运动指令的目标位置



8.2.1.3 融合

前后两条指令以设定的融合模式进行速度融合，具体的速度融合方式由融合模式决定。


运动指令1
Execute
Busy
Active
Done
运动指令2
Execute
Busy
Active
Done
Velocity
BufferMode=
BlendingLow
V1
V2
时间
Velocity
BufferMode=
BlendingPrevious
V1
V2
时间
Velocity
BufferMode=
BlendingNext
V1
V2
时间
Velocity
BufferMode=
BlendingHigh
V1
V2
时间

北京和利时智能技术有限公司

Beijing HollySys Intelligent Technologies Co., Ltd..

地址：北京经济技术开发区地盛中路2号院

邮编：100176

电话：010-5898 1588

热线：400-811-1999

传真：010-5898 1558

http://www.hollysys.com