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

PMC600 中型 PLC 用户手册（四）

运动指令篇

2021 年 6 月

版权说明

目 录

关于版本的说明

PMC600 系列 IEC 控制器使用的软件系统、库版本、设备版本与 SMC604/604A/606 等 IEC 控制器所使用的有所区分，如下表 II 所示。

表 I　配套版本介绍

控制器	SMC604/604A/606	PMC600
软件版本	IEC STUDIO V3.5	Codesys V3.5.15.40
主站设备版本	V3.5.9.50	V3.5.15.40
编译库版本	V3.5.9.50	V3.5.15.40
运动库版本	V4.1.0	V4.8.0
总线主站与配套库版本	V3.5.9.50	V3.5.9.50
ETC402 轴与配套库版本	4.0.0	4.6.0

关于库名称的说明

如上所述，SMC604/604A/606 控制器与 PMC600 控制器所使用的库版本是有差异的，但功能本身差异不大。下面具体将控制器的库文件名称对应关系列出来，比如 SMC606 控制器中的 Ipolib 库，对应了 PMC600 控制器的 SMC_Ipolib 库；比如 SMC606 控制器中的 SMC606 库，对应了 PMC600 控制器的 SMC_Controller 库等等，具体请看表III：

表 II　库名称差异

控制器	SMC604/604A/606	PMC600
基础库	SMC606、SysDateTime	SMC_Controller
通讯功能	HMI、RS485、TCPIP、SMC_Communication	SMC_Communication
文件功能	FileManage	SMC_FileManage
插补模块	IpoLibModule	SMC_BasicModule
单轴运动	EtherCATVirtualAxis、HomeLib、SingleAxisLib	SMC_SingleAxisLib
插补运动	IpoLib	SMC_IpoLib
机器人	IpoRobot	SMC_Robot
CANOPEN	CAN_CQ、CANBusAPI、CANopenLib	—

第 1 章　运动控制指令概要

1.1. 运动控制指令的分类

使用 PMC 控制器进行运动控制时，需要创建基于 PMC 设备（device）的工程（project），并在工程的用户程序（POU）之中，调用运动控制指令，以执行运动控制功能。

运动控制指令本身是实现运动功能的功能块（FB），所有的运动控制指令以 PLCopen 运动控制用功能块的技术规格为基础。

PMC 控制器支持两种运动控制指令，第一种是 PLCOpen 的运动控制指令，第二种是雷赛的专用运动控制指令。

本章将分别对 PLCOpen 运动控制指令、雷赛专用运动指令，以及运动控制指令的基础知识做概要说明。

PLCOpen

PLCopen 是独立于生产商和产品的全球性协会，其宗旨是成为一个领导协会来解决该领域中有关控制编程的问题，支持使用国际标准。

PLCopen 的一项主要活动是致力于 IEC61131-3（PLC 编程的国际标准规格），它是工业控制编程唯一的全球标准。它使编程接口标准化从而协调了人们设计和从事工业控制的方式。标准的编程接口允许不同背景和技能的人们在软件生命周期的不同阶段创造成不同元素的程序：技术规范、设计、实现、测试、安装和维护，它们都遵守一个共同的结构并且和谐地一起工作。该标准定义了用于构造程序内部结构的 SFC（顺序功能图）语言和四个互操作编程语言：IL（指令表）、LD（梯形图）、FBD（功能块图）和 ST（结构文本）。通过分解成逻辑元素、模块化以及现代软件技术来组成每个程序，从而提高了其重复使用性，减少了错误，提高了编程和用户的效率。

雷赛专用运动指令

雷赛指令是按 PLCopen 标准开发的，使用方法与 PLCopen 指令一致。雷赛 PMC600 系列产品是基于德国 3S 公司 CoDeSys 编程平台而开发的，该平台完整支持 PLCopen 的运动控制指令，用户可以引用许多标准的功能函数库。另外，为了贴合用户的实际需求，简化运动控制指令的用法，雷赛基于 PLCopen 运动控制指令的标准，封装了一些专用于 PMC600 系列产品的运动控制指令，使用方式与 PLCopen 指令完全一致。这让 PMC600 控制器的运动控制功能更加丰富，用户有了更多的功能可以选择。

在使用 PMC600 的运动控制指令时，一般以 PLCopen 的指令为主，在适当的场合再辅以雷赛开发的专用运动控制指令。

运动控制指令类别

PMC600 系列总线控制器的运动控制指令主要分为五大类：单轴指令、同步指令、轴组指令、插补指令以及雷赛专用特殊 IO 指令，如表 1.1 所示。

表 1.1　PMC600 系列总线控制运动控制指令分类

种类	功能项目	简要说明
单轴	单轴参数设置	设置单轴运动参数的指令
	单轴状态监控	读取单轴运动状态的指令
	单轴运动动作	使轴执行单轴动作的指令
同步	电子齿轮	电子齿轮运动的设置、动作执行指令
	电子凸轮	电子凸轮运动的设置、动作执行指令
轴组	轴组参数设置	设置轴组运动的参数的指令
	轴组状态监控	读取轴组的运动状态的指令
	轴组坐标系	对各种坐标系进行转换的指令
	轴组运动动作	使轴组执行运动动作的指令
插补	多轴插补指令	执行直线、圆弧等单段插补的指令，速度不连续
	连续插补指令	执行直线、圆弧等单段插补的指令，速度连续
	G 代码插补指令	使用 G 代码进行插补运动的指令
雷赛专用特殊 IO	高速比较	与控制器高速输出口绑定的运动控制指令
	高速位置锁存	与控制器高速输入口绑定的运动控制指令

1.2. 运动控制指令基础说明

下面将对本手册内所有的运动控制指令的基本规格和限制事项做出说明，以帮助用户更好更快地使用运动控制指令。

指令名称

本手册所描述的运动控制指令，都是以 PLCopen 为基础的指令。其中名称以 MC 和 SMC 开头的指令均是源于 codesys 提供的 PLCopen 相关库以及 SM3_Basic、SM3_CNC 和 SM3_Robotics 等库；而以 LS 开头的指令，则来自于雷赛的专用运动库，如 PMC_Ipolib、PMC_SingleAxisLib、PMC_Controller 等库。高速比较和高速锁存所使用的运动控制指令，来自于雷赛提供的 PMC_Controller 库。

编程语言

本手册所涉及的 PLCopen 和雷赛专用运动控制指令，支持以下的编程方式：

- 图形方式（LD、FBD、CFC）
- 结构文本方式（ST）

图形编程（LD、FBD、CFC）：

以 MC_MoveRelative 指令为例，运动控制指令实例配置到梯形图上的表达方式如图 1.1 所示，图中标记出各个参数的名称。

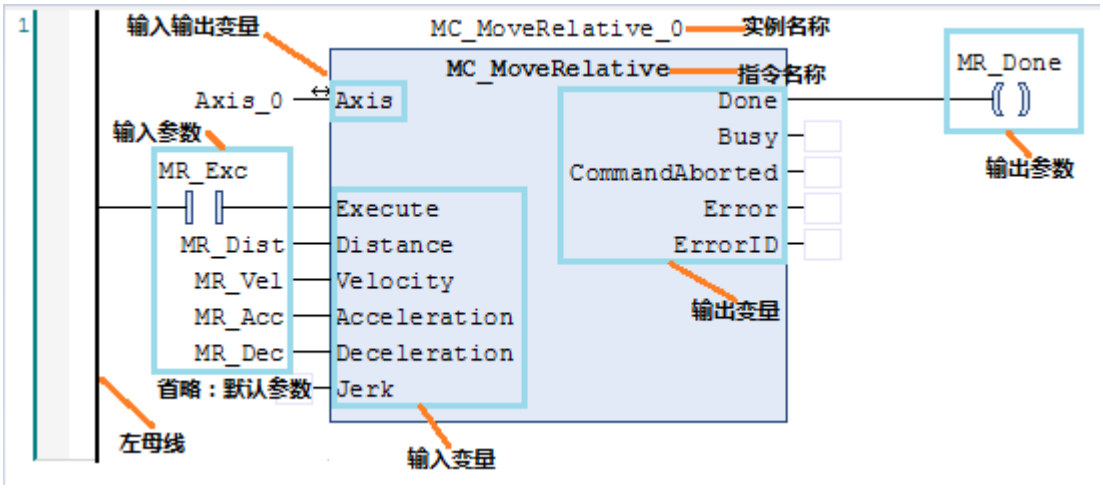


图 1.1　梯形图格式

- 向输入输出变量 Axis 中输入指定的运动轴的变量名称。
- 通过输入参数向输入变量指定目标位置、速度、加速度等运动参数。
- 输出变量向各输出参数输出指令的状态以及报错等信息。
- 省略输出参数时，指令在执行时将使用默认初始值参数。

结构文本（ST）：

仍以 MC_MoveRelative 指令为例，运动控制指令实例配置到结构文本的表达方式如下。

```
MC_MoveRelative_0(
    Axis：= Axis_0，
    Execute：= MR_Exc，
    Distance：= MR_Dist，
    Velocity：= MR_Vel，
    Acceleration：= MR_Acc，
    Deceleration：= MR_Dec，
    Jerk：= ，
    Done=> ，
    Busy=> ，
```

```
CommandAborted=> ,
Error=> ,
ErrorID=>
);
```

运动指令的任务配置

在使用运动控制指令的时候，需要特别注意运动控制指令的任务配置方法。运动控制指令所在的 POU，其任务优先级需要配置为 0，即实时任务。需要这么去配置的原因，是因为在实际运动场合，运动控制需要确保极高的稳定性，以确保每个周期运动轴所走的位置都能按照规划，稳定地执行。因此，在控制端对运动控制指令的实时性要求较高，配置优先级为 0 的实时任务，以让运动控制按照规划去执行。

运动指令在程序中的引用

在调用运动控制执行指令的时候，应避免将运动指令放在逻辑条件之中，否则可能会因为逻辑问题，导致指令没执行完成，便不再调用，从而发生“SMC_FB_WASNT_CALLED_DURING_MOTION”报错，如图 1.2 所示。

正确的用法，应该是将运动控制执行指令放到逻辑条件外面，在逻辑条件之中调用运动控制执行指令的执行参数，如图 1.3 中的常用的“Execute”参数，从而实现运动控制指令的调用和执行。

```
34      IF COUNT>33 THEN
35          MC_MoveRelative_0(
36              Axis:= Axis_0,
37              Execute:= MR_Exc,
38              Distance:= MR_Dist,
39              Velocity:= MR_Vel,
40              Acceleration:= MR_Acc,
41              Deceleration:= MR_Dec,
42              Jerk:= ,
43              Done=> ,
44              Busy=> ,
45              CommandAborted=> ,
46              Error=> ,
47              ErrorID=>
48              );
49      END_IF
50
```

图 1.2　错误调用运动控制执行指令的方法

```
54      MC_MoveRelative_0(
55          Axis:= Axis_0,
56          Execute:= MR_Exc,
57          Distance:= MR_Dist,
58          Velocity:= MR_Vel,
59          Acceleration:= MR_Acc,
60          Deceleration:= MR_Dec,
61          Jerk:= ,
62          Done=> ,
63          Busy=> ,
64          CommandAborted=> ,
65          Error=> ,
66          ErrorID=>
67          );
```

图 1.3　正确调用运动控制执行指令的方法

运动控制执行指令同时启动

以下对在同一个任务周期内对同一个轴启动多个运动控制执行指令的情况进行说明。由于 PLC 遵循由上到下的程序扫描方式，因此在同时启动一个轴的多个运动控制执行指令时，总是执行最后一个指令，可参考下面的两种情况。

情况 1：在同一个任务周期内执行同一个轴的 2 个相同运动控制指令，只有速度不同，如图 1.4 所示。最终的执行结果是轴走到了 2234 的位置。

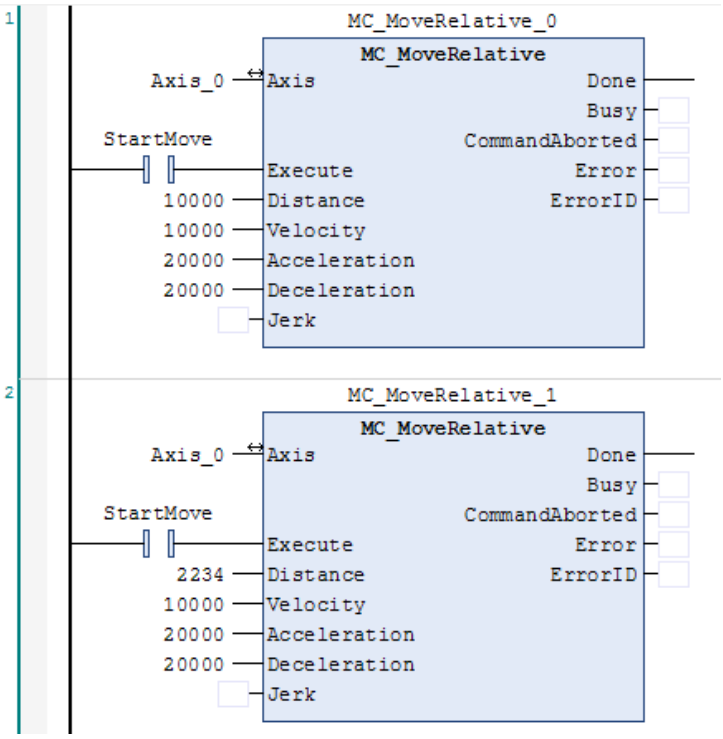


图 1.4　情况 1

情况 2：在同一个任务周期内执行同一个轴的 2 个不同运动控制指令，如图 1.5 所示。最终的执行结果是轴走到了-5000 的位置。

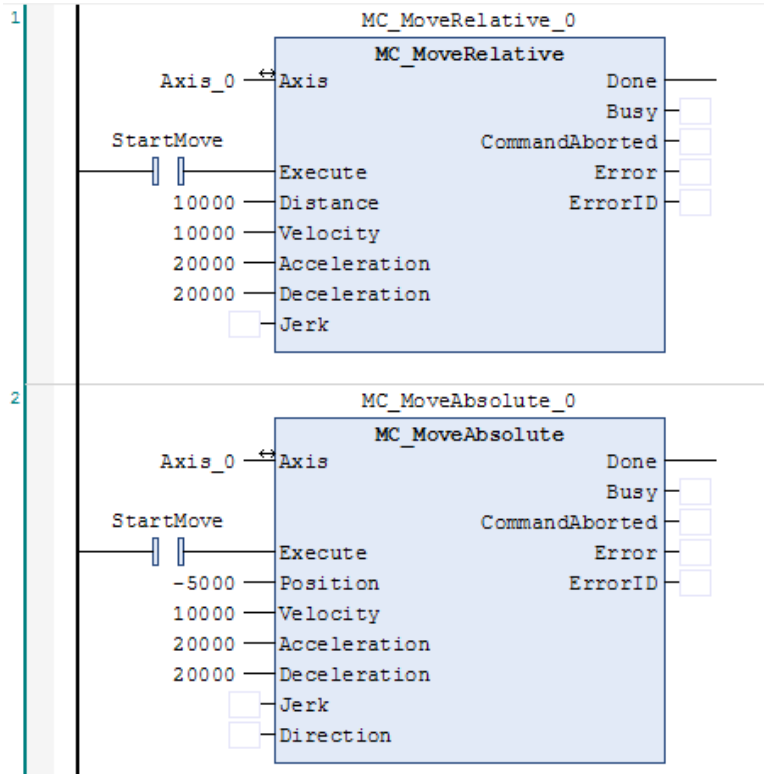


图 1.5　情况 2

第 2 章　运动控制系统概要

2.1　EtherCAT 总线概要

EtherCAT（以太网控制自动化技术）是一个开放架构，以以太网为基础的现场总线系统。各节点高速传输 Ethernet 帧，可实现较短的通信周期时间。此外，利用时钟信息共享机制，可实现通信抖动低、精度高的同步控制。

PMC600 控制器使用标准的 EtherCAT 总线协议，配套雷赛、松下等 EtherCAT 总线伺服构成的总线控制系统，所有控制信息均可通过高速数据通信进行交换。利用数据通信传达各种控制指令，不受编码器反馈脉冲的响应频率等接口规格的制约，可最大限度地发挥伺服电机的性能。此外，伺服驱动器的各种控制参数或监视信息可在上位控制器上处理，因此可实现系统信息管理的统一化。

EtherCAT 网络配置和使用方法，请参考《PMC600 中型 PLC 用户手册（二）　编程软件篇》。

2.2　运动控制系统结构

脉冲伺服控制系统一般以半闭环方式控制电机的动作。是通过电机上安装的编码器，检测与指令值对应的电机旋转量，然后作为机械移动量反馈。计算指令值和电机实际旋转量的偏差，使两者差值为“0”的控制方式。仅在伺服与电机构成的系统内部形成闭环，并不反馈实际位置给控制端，如图 2.1 所示。

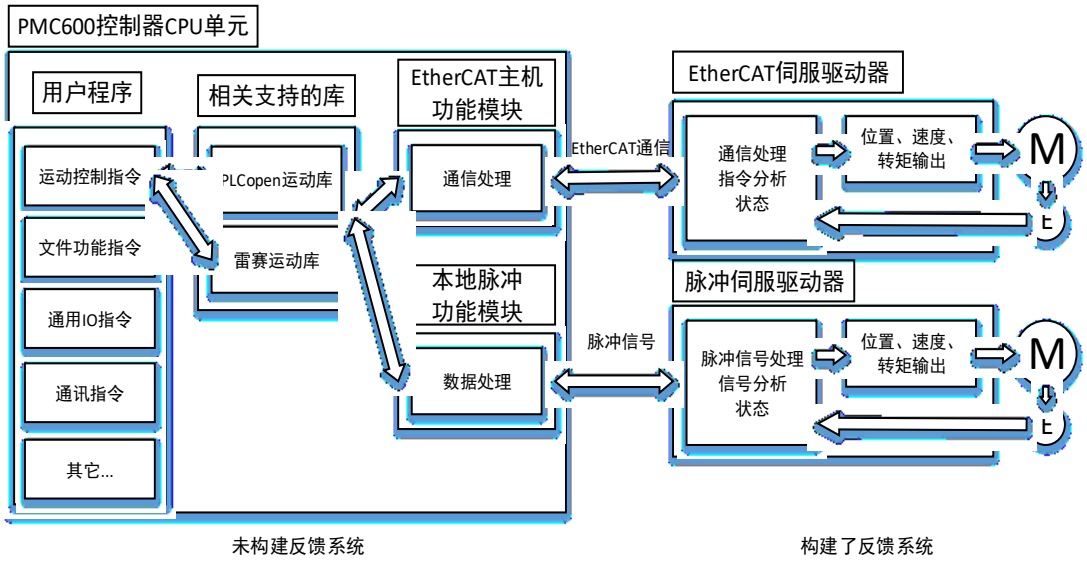


图 2.1　伺服控制系统的结构

- 在用户程序中启动运动控制指令，将在相应的 PLCopen 运动库和雷赛运动库中对运动算法进行分析和支持。
- 运动算法的分析结果，将按照循环周期的方式执行运动计算，生成的运动信息发送至总线伺服驱动器的指令值。生成了包括目标位置、目标速度、目标转矩。脉冲轴则是将指令值发送给 FPGA 芯片。
- 生成的指令值，将按照 EtherCAT 通信的过程数据通信周期通过 PDO 通信进行发送。脉冲轴则通过内部总线进行发送。
- 伺服驱动器根据接收到的周期发送的指令值，执行对电机的位置、速度、转矩控制。
- EtherCAT 总线方式下，编码器的当前值或伺服驱动器的状态安装 EtherCAT 通信的过程数据通信周期发送至 CPU 单元。

2.3 CPU 单元与任务

任务类型：

任务是指用户程序的处理执行条件或顺序属性。

PLC 的 CPU 支持多种任务类型：循环、惯性滑行、事件和状态 4 种。

任务优先级：

任务有执行优先级，会优先执行优先级较高的任务。正在执行某一任务时，若有执行优先级更高的任务满足执行条件，将优先执行优先级更高的任务。支持的优先级等级设置有（0 - 31）。

任务的分配：

一个任务内可以包含多个 POU 执行程序，多个 POU 将均以相同的时间间隔执行，执行时按该任务中 POU 添加顺序执行。

轴、轴组以及轴的 I/O 设备，需要分配到优先级为“0”的主循环任务之中。

在程序中调用运动指令时，运动指令本体也需要放到优先级为“0”的主循环任务之中。指令的触发则可以放到其他的处理逻辑顺序的任务之中。

一般 EtherCAT 任务的周期设置为 1ms、2ms、4ms 或 8ms，而普通循环任务（如通信任务、逻辑任务等）的周期为 20ms，更低优先级（如文件执行任务）的循环周期为 100ms 以上，等等。

注意：EtherCAT 周期一般不要设为 3ms、6ms、7ms、9ms 等，容易因非整数倍的关系带来一些问题。

任务的周期：

在多个循环任务的工程中，需要先确定优先级为“0”的主任务的周期，其它的任务再围绕着优先级最高的主任务周期去进行配置，需要与主任务的周期同步执行，且这些任务的任务周期需要设定为主周期的整数倍。

例如，设定任务 1 为主周期（优先级 0）1ms、任务 2（优先级 5）的任务周期 2ms、任务 3（优先级 16）的任务周期 4ms 时，则以每两个主周期一次为间隔，任务 1 和任务 2 的周期起点重叠。同样，任务 3 和任务 1 以四个周期一次为间隔，周期起点重叠。

2.4 EtherCAT 通信

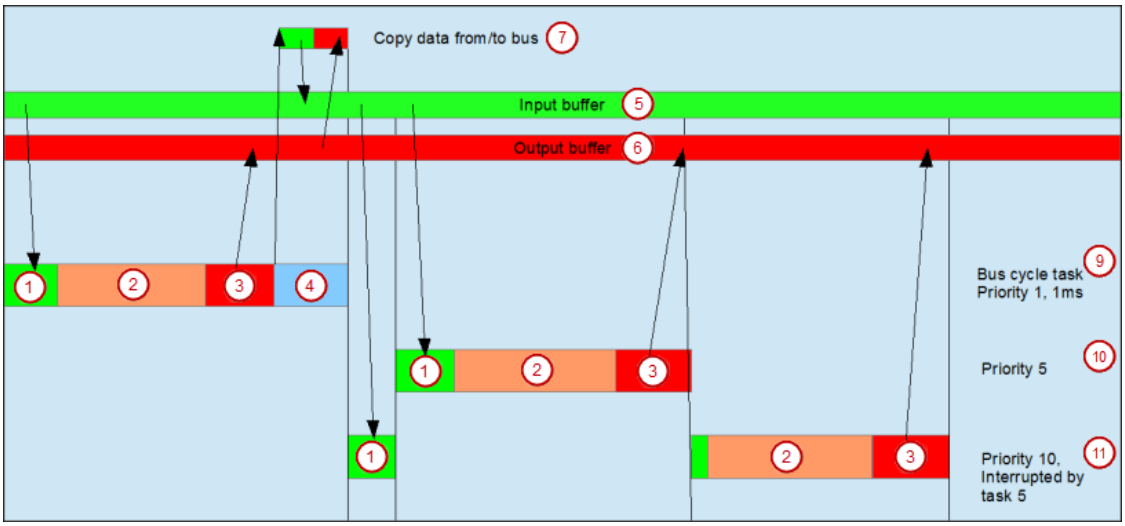
总线周期任务

一般信息

图 2.2 对普通的任务用图示做了说明。对于每个 IEC 任务，通常在每个任务的开始处（1）读取使用的输入数据，并且在任务结束时将写入的输出数据传送到 I/O 驱动器(3)。I/O 驱动程序中的实现对于进一步传输 I/O 数据起决定性作用。因此，该实现负责实际传输在相应总线系统上发生的时间帧和特定时间。

可以为 PLC 设置中的所有现场总线全局定义 PLC 的总线循环任务。但是，对于某些现场总线，您可以独立于全局设置更改此设置。具有最短循环时间的任务用作总线循环任务(设置：在 PLC 设置中未指定)。在此任务中，消息通常在总线上传输。

其他任务仅复制内部缓冲区中的 I / O 数据，该内部缓冲区仅与总线循环任务中的物理硬件交换。



(1) 从输入缓存区读输入　(2) IEC 任务　(3) 写输出到输出缓存区
(4) 总线周期 (5) 输入缓存区　(6) 输出缓存区　(7) 与总线互相复制数据
(9) 总线周期任务，优先级 1，1ms　(10) 总线周期任务，优先级 5
(11) 总线周期任务，优先级 10，被任务 5 中断

图 2.2　总线周期任务

注意：

如果输出是在各种任务中进行写入的，则状态是未定义的，因为在每种情况下都可以覆盖它。在各种任务中使用相同的输入时，输入可能会在处理任务时发生变化，如果任务

被具有更高优先级的任务中断并导致再次读取流程图，则会发生这种情况。解决方案：在 IEC 任务开始时，将输入变量复制到变量，然后仅使用其余代码中的局部变量。

结论：在几个任务中使用相同的输入和输出没有任何意义，并且在某些情况下可能导致意外的反应。

EtherCAT 总线任务周期行为

在复制 IEC 输入之前，读取最后一个周期的待处理网络消息。如图 2.3 所示。

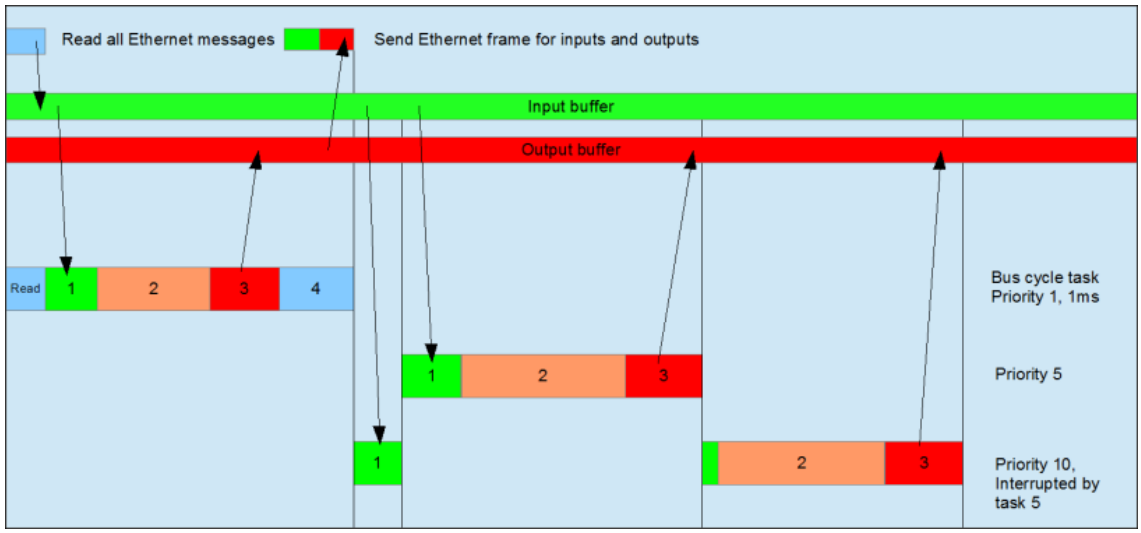


图 2.3　EtherCAT 总线任务周期行为

如果在 EtherCAT 主站的设置中激活了按任务发送/接收选项，则会将额外的报文传输到每个任务所使用的设备以及所使用的输入或输出。在慢速任务中使用通道的传输也不太频繁。因此可以减少总线负载。如图 2.4 所示。

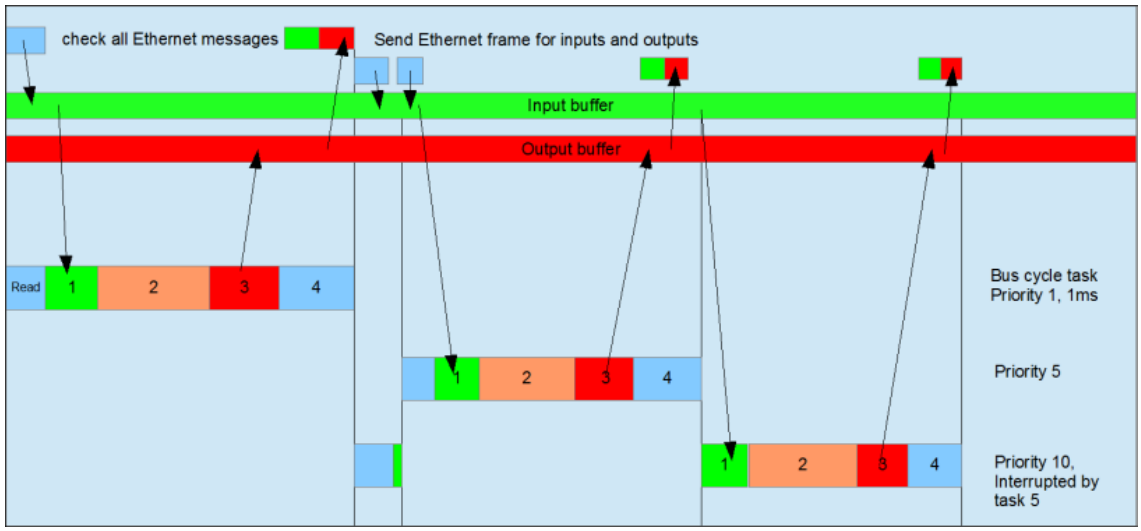


图 2.4　EtherCAT 总线任务周期行为

运动控制周期与总线周期的关系

PLC 基本功能模块通过启动用户程序中的运动控制指令，向运动控制功能模块发行运动控制指令。运动控制功能模块将根据该指令执行运动计算，并将计算结果发送到 EtherCAT 上的伺服驱动器，或本地脉冲轴上。如图 2.5 所示，“主周期”=“运动控制周期”=“EtherCAT 通信的过程数据通信周期”。

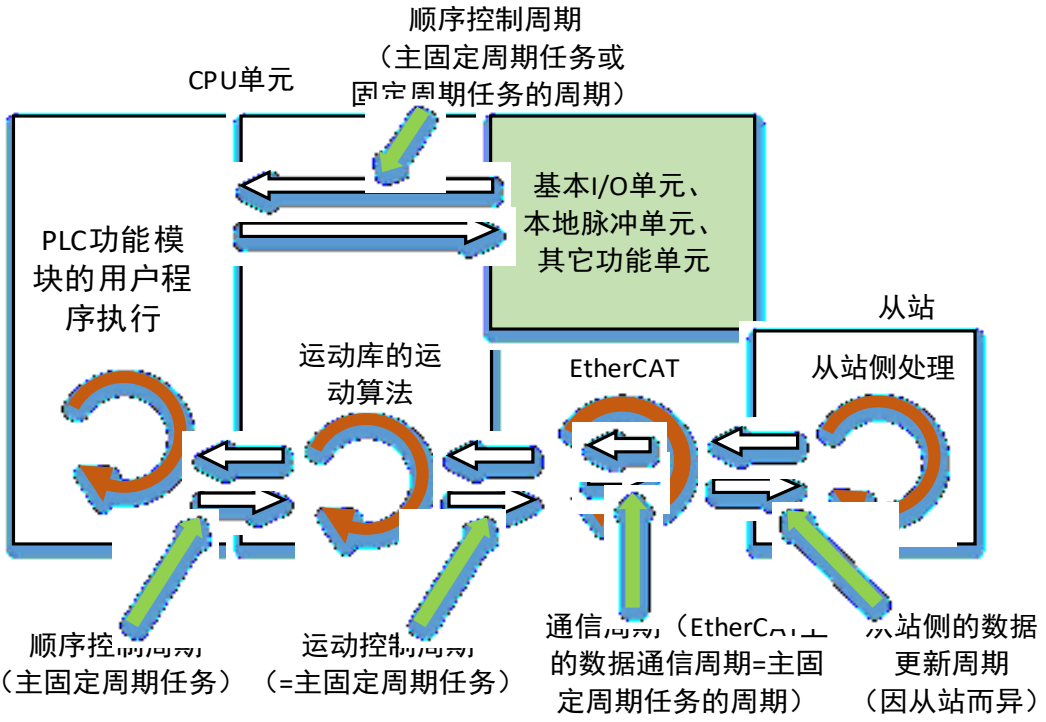


图 2.5　运动控制周期与总线周期的关系

同步时钟概述

当需要在 EtherCAT 总线上连接多个总线轴时，这些总线轴作为从站，往往需要同时开始运行或停止运行，但由于每个从站本身的绝对时间的不同，因此需要使用分布式时钟（Distributed Clock，简称 DC）机制，来让每个智能从站（如伺服驱动器、智能高速扩展模块），保持同一个时间，在这里叫做时钟，每个从站按设定的同步出发时间，将主站写入的数据输出给执行单元，达到同时运行的目的，如图 2.6 所示。

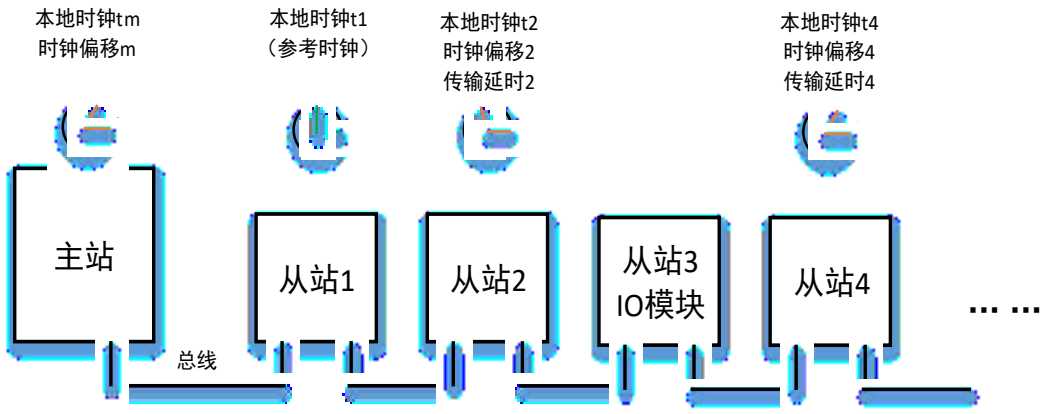


图 2.6　同步时钟

如图 2.6 中的网络所示，在 EtherCAT 总线上的每个智能从站内部都有一个高分辨率的内部本地时钟 TLocal(n)，在 EtherCAT 总线初始化阶段，主站会读取每一个从站的当前时间，将第一个从站的本地时间作为网络的“参考时钟”，即可算得每个从站相对于参考时钟的“时钟偏移 Toffset(n)”，将每个从站的时钟偏移写给对应的从站，以便其修正时钟，消除静态误差；另外在通信数据帧传输的过程中，还会有因硬件网络产生的传输延迟时间，主站会通过发送特定的广播帧，让每个从站记录数据到达时刻，主站再读取每个从站所记录的时刻值，同时测量数据返回数据帧的总延迟，可以准确计算出每个从站的“传输延迟 Tdelay(n)”，之后主站将每个从站的传输延迟时间写入每个从站的内存，从站在有了这些时钟修正值后，通过计算 TLocal(n) - Toffset(n) -Tdelay(n)，就得到与参考时钟 t1 相同的时钟了。

EtherCAT 网络中，对 DC 时钟并不敏感的 IO 从站，可以不设置 DC 处理，EtherCAT 主站在 DC 校准时，会忽略其时钟的校对。

每个从站 ESC 芯片内部还有同步脉宽寄存器，一旦同步单元被激活，即可定时产生 SYNC 同步信号，将当前收到的数据生效，对于伺服驱动器，就是以所收到的位置指令为目标点，开始执行。

上面所述的 EtherCAT 网络从站的 DC 时钟的初始化校准操作，由 EtherCAT 主站自动完成，无需用户干预，当 EtherCAT 总线就绪时，表明 DC 初始化工作也已完成。用户需要注意的是，将具有内部时钟功能的从站，尽量安排在网络的前端。

PDO 和 SDO

如图 2.7 所示，PMC 控制器使用 CoE 协议与 EtherCAT 总线上的从站进行信息交换。CoE 中，各种从站拥有的参数、控制信息由名为对象字典 (OD) 的数据规格规定。为了在控制器 / 通信主机和从站之间传达这些数据，可使用以固定周期进行实时信息交换的过程数据对象 (Process Data Objects： PDO) 和在任意时间传输信息的服务数据对象 (Service Data Objects： SDO)。

在运动控制中，一般需要对伺服电机的位置控制等以固定控制周期更新输入输出数据的指令，使用 PDO 通信。而对于参数传送等以指定时间读取或写入数据的指令（如设置参数），则使用 SDO 通信。

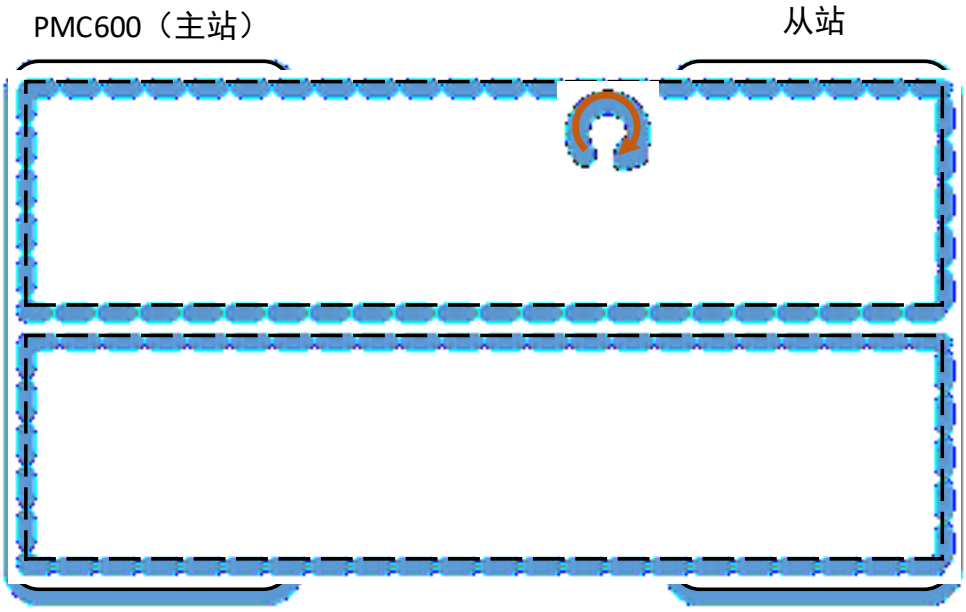


图 2.7　主站与从站的通讯

TPDO 和 RPDO

在软件中，PDO 的数据通过“PDO 配置表”进行配置，“PDO 配置表”中存放的是主站需要发送和读取的控制参数（对象）的“索引号”（主索引号 ： 子索引号），分为两个配置表：TPDO 和 RPDO，如表 2.1 所示。

表 2.1　TPDO 和 RPDO 的配置表

	用途说明
TPDO 配置表	编程时，根据从站间的循环发送的内容，来进行这个对象与属性的清单表的配置；该表在网络初始化阶段控制器会自动发送给从站 ESC；主站将根据该表安排发送缓冲的大小，运行时将待发送的命令数据存入发送缓冲；运行时从站根据此表解析所收到的数据帧；每个从站可以有不同的 TPDO 配置表。
RPDO 配置表	编程时，根据从站自动应答的对象内容，来进行这个对象与属性的清单表的配置；该表在网络初始化阶段会自动发送给从站 ESC；运行时从站根据此表制备数据，在主站访问本从站时及时将数据插入 EtherCAT 数据帧的时隙，然后返回给主站；运行时主站根据此表解析返回数据帧从站应答数据；每个从站可以有不同的 RPDO 配置表。

在网络初始化阶段，主站会将这份包含有 TPDO、RPDO、每个对象的数据类型与宽度等信息的“PDO 配置表”发送给从站，作为从站解析数据帧的依据。

TPDO 配置表中依次存放每个对象的索引号及数据宽度信息，各对象的在表中的顺序，将是系统将运动控制指令所要发送数据放入发送缓存单元的依据。

从站根据 RPDO 配置表，将每个对象的索引号及顺序，依次将伺服的运行状态数据，存放于应答缓存单元，当主站通信帧访问本从站时，ESC 自动将该缓存单元的数据插入到数据帧的合适时隙，返回给主站；RPDO 表也将是主站解析从站应答数据的依据。

CiA402 与伺服常用对象字典

PMC600 在 EtherCAT 总线通信应用层上采用的是 CANOpen DS402 协议，也叫作 CIA402 协议。CiA402 协议的核心包括以下几部分：

（1）定义了“对象字典 OD”及其功能属性，统一通信数据解析方法；

（2）周期性的过程通信数据，先发送过程对象配置，在完成配置后，则将数据包以帧的格式，周期性地发送到从站上；

（3）其偶发性的数据，也可以采用附加通信字段的方式，与对应的从站进行问答式的通信；

（4）将伺服的主要设置参数、控制参数、状态参数等项目，归纳为具有固定编号（索引号+子索引号）的“对象”，完整的对象定义表就是“对象字典”。在 CiA402 的定义里面，索引号的范围和作用如下：

①0x0000-0x1FFF 用来存放协议类型、制造商信息、配置表描述信息等；

②0x2000-0x5FFF 制造商定义的对象，可能用来设置功能码参数，这类静态参数的设置；

③0x6000-0x9FFF 作为主站和伺服进行控制的通信交互数据；

④0xA000-0xFFFF 保留

表 2.2 列出了在总线运动控制中，伺服、步进轴常用到的地址以及作用。

表 2.2　总线运动控制中常用的地址及其作用

RxPDO [261th 接收 PDO 映射] (1704Hex)	控制字 (6040Hex)、目标位置(607AHex)、 目标速度(60FFHex)、目标转矩(6071Hex)、操作模式(6060Hex)、锁定功能(60B8Hex)、最大曲线速度 (607FHex)、正方向转矩限制值(60E0Hex)、负方向转矩限制值(60E1Hex)
TxPDO [259th 发送 PDO 映射] (1B02Hex)	错误代码(603FHex)、状态字(6041Hex)、当前位置(6064Hex)、当前转矩(6077Hex)、操作模式显示(6061Hex)、锁定状态(60B9Hex)、锁定位置 1(60BAHex)、锁定位置 2(60BCHex)、数字输入(60FDHex)

用户如果需要了解到每个地址的具体定义和更加详细的功能描述，请参考具体的驱动厂家的手册。

第 3 章　总线运动控制工程

一般一个总线运动控制工程，为了实现 CPU 资源效率的最大化，可以利用控制器本身多线程执行的特点去进行配置。将不同循环周期，不同优先级的功能配置到多个任务中，并将这些任务按优先级和循环周期的需要，分别配置不同的任务优先级和周期时间。

比如在实际的应用中，用户的工程除了使用运动模块，可能会同时用到触屏、文件、IO、逻辑控制等功能，需要分别将各类功能分配到不同的 POU 中，再将这些 POU 放到不同的任务中，并配置不同优先级和周期时间。

运动工程构成

如图 3.1 所示，以一个普通的工程为例，其由若干部分组成，包括：

（1）用户程序：多个任务，多个 POU，和不同的优先级；

（2）文件库：codesys 基础库、PLCopen 库、雷赛专用库，以及其它专用库等；

（3）设备：EtherCAT 总线及从站、高速 IO 模块等。

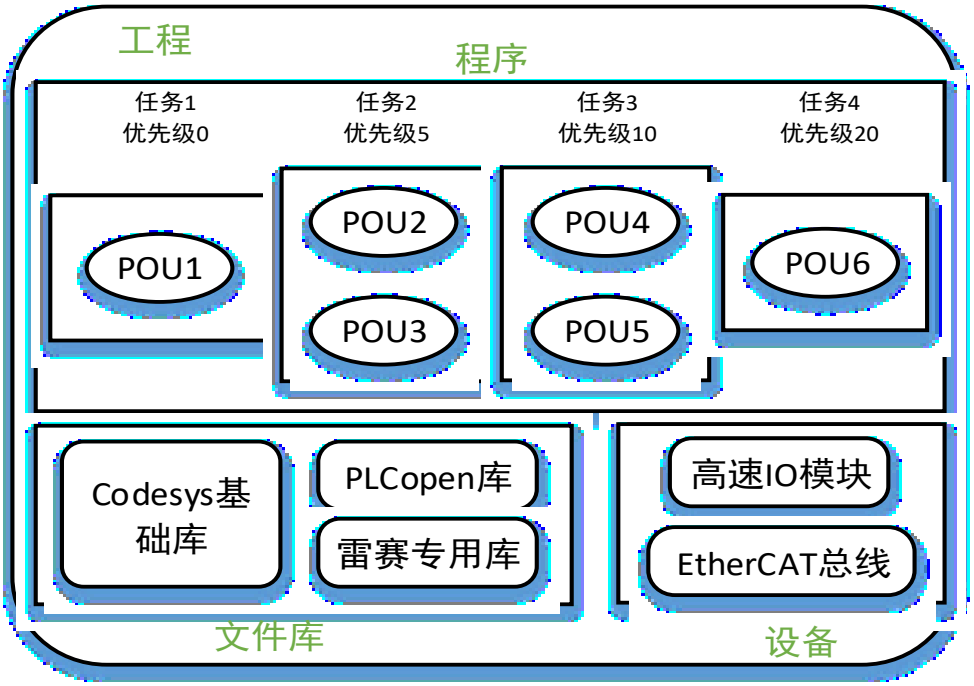


图 3.1　用户工程的组成

在一个工程中，程序中所用到的指令，都需要来自文件库的支持。而每一个 POU，如果不在某个任务中调用的话，便不会执行。用户可以选择直接配置到某一个任务之中去执行，或者选择在另外一个已经在任务中的 POU 对该为配置任务的 POU 进行调用。

POU 中执行的程序，如果需要和外部的 IO 或总线进行交互，则需要在程序中另外配置相应的高速 IO 模块或 EtherCAT 总线及从站设备。

多个 POU 的用法

在编写应用程序时，不同执行周期的程序功能，应放到不同的 POU 中进行编写，并配置到不同优先级和周期时间的任务之中，这样做，会带来很多好处。

- 合理分配 CPU 资源，按每个功能需要的周期时间去进行周期的分配；
- 程序结构清晰，各功能区分清楚，不混在一块，且名称从工程栏便可以清晰地一目了然；
- 调试方便，在调试时，可以很方便地对某部分需要屏蔽的功能进行屏蔽；
- 可以实现在不同工程之间，对 POU 进行直接的引用，直接将 POU 从工程 1 复制到工程 2；
- 将程序规划清楚后，可以分到多个人进行编程开发，提高编程的效率
- 可以在不同的 POU 使用不同的编程语言，只要接口清晰，里面是用什么编程语言编写的就没那么重要了。

运动功能的调用方式

如上述，在一个工程中，为了更合理地分配 CPU 的资源，编程时会将不同周期的程序放到不同的 POU 和任务之中。

运动功能需要最高优先级的任务，而逻辑功能一般不需要那么高优先级的任务配置，因此，在实际工程中，通常这两块是放在两个不同的 POU 与任务之中的。那么，要怎么实现即使运动功能与逻辑功能分开，依然能够在逻辑功能之中去控制一个运动功能的执行？

一般都是在运动功能中定义一些输入变量和输出变量，给其它功能进行调用，如在逻辑 POU 中，需要调运动功能，就往运动 POU 的输入变量写入控制数据，运动 POU 将运动状态放到输出变量中，给到逻辑 POU，用以判断运动状态，判断程序逻辑的执行。该过程如图 3.2 所示。

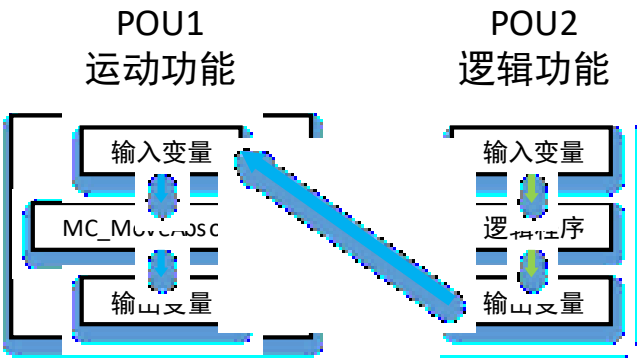


图 3.2　运动功能的调用

第 4 章　变量和指令

4.1 变量

轴变量　AXIS_REF_SM3（FB）

此驱动接口是一个功能块（FB），它包含了驱动轴工作需要的相关参数。每一个 SoftMotion 轴都是 AXIS_REF_SM3 的一个实例。此接口内部参数一般用户不需要使用。

表 4.1　轴变量相关参数

编号	名称	数据类型	默认值	注释
1000	nAxisState	SMC_AXIS_STATE （INT）	standstill	根据 PLCopen 状态图，轴的状态： 0：关机 1：错误停止 2：正在停止 3：待机 4：离散运动 5：连续运动 6：同步运动 7：回零运动
1012	bCommunication	BOOL	FALSE	TRUE 通讯正常
1013	wCommunicationState	WORD	16#FFFF	
1014	uiDriveInterfaceError	UINT	0	驱动接口错误号
1036	bDisableErrorLogging	BOOL	FALSE	不将错误记录到 fbeFBError
1040	bVirtual	BOOL	FALSE	TRUE：虚拟驱动；应用 AXIS_REF_VIRTUAL_SM3 方法
1112，9	fMaxVelocity	LREAL	100	速度最大值，以“u/s”表示
1113	fSWMaxVelocity	LREAL	100	动作的默认速度最大值，以“u/s”表示
1122，13	fMaxAcceleration	LREAL	100	加速度最大值，以“u/s^2”表示
1123	fSWMaxAcceleration	LREAL	100	默认加速度最大值，以“u/s^2”表示
1132，15	fMaxDeceleration	LREAL	100	减速度最大值，以“u/s^2”表示
1133	fSWMaxDeceleration	LREAL	100	默认减速度最大值，以“u/s^2”表示
1142，16	fMaxJerk	LREAL	100000	跃度最大值，以“u/s^3”表示
1143	fSWMaxJerk	LREAL	100000	默认跃度最大值，以“u/s^3”表示
1152	fMaxCurrent	LREAL	100	当前的最大值
1153	fSWMaxCurrent	LREAL	0	用户定义的当前的最大值

1162	fMaxTorque	LREAL	0	最大转矩值，以“Nm”或“N（直线运动）”表示
1200，2	fSWLimitPositive	LREAL	0	在正方向上的位置限制，以“u”表示
1201，3	fSWLimitNegative	LREAL	0	在负方向上的位置限制，以“u”表示
1203	fSWLimitDeceleration	LREAL	0	减速度[u/s^2]限制，如果有软件限制开关，系统将以该值刹车。
1205	bSWLimitEnable	BOOL	FALSE	激活软件限制开关
1250	fSWErrorMaxDistance	LREAL	0	在一个错误产生之后驱动器达到 standstill 状态的距离

常用变量

MC_DIRECTION （ENUM）

此全局变量的作用是指定运动方向，是很多功能块的输入参数。

注意：方向模式共五种，并不是所有的方向模式都可以应用到所有的功能块。

表 4.2 运动方向变量

枚举值	类型	初始值	描述
fastest	MC_DIRECTION	3	自动选择最快到达目的地的方向（仅模数轴）
current	MC_DIRECTION	2	保持当前方向（仅模数轴）
positive	MC_DIRECTION	1	在正方向上移动
shortest	MC_DIRECTION	0	按最短距离选择方向（仅模数轴）
negative	MC_DIRECTION	-1	在负方向上移动

SMC_CONTROLLER_MODE （ENUM）

此变量描述驱动器的控制模式。

表 4.3 控制模式变量

枚举值	类型	初始值	描述
SMC_torque	SMC_CONTROLLER_MODE	1	力矩
SMC_velocity	SMC_CONTROLLER_MODE	2	速度
SMC_position	SMC_CONTROLLER_MODE	3	位置
SMC_current	SMC_CONTROLLER_MODE	4	当前模式

SMC_HOMING_MODE （ENUM）

此全局变量定义了回零模式，共有 6 种回零模式，将在功能块 SMC_Homing 中使用。

表 4.4 回零模式变量

枚举值	类型	初始值	描述
FAST_BSLOW_S_STOP	SMC_HOMING_MODE	0	快速移动到原点开关；反向并慢速离开原点开关；执行“设置位置”；停止
FAST_BSLOW_STOP_S	SMC_HOMING_MODE	1	快速移动到原点开关；反向并慢速离开原点开关；停止；执行“设置位置”
FAST_BSLOW_I_S_STOP	SMC_HOMING_MODE	2	快速移动到原点开关；反向并慢速离开原点开关；等待 index 脉冲；执行“设置位置”；停止
FAST_SLOW_S_STOP	SMC_HOMING_MODE	4	快速移动到原点开关；慢速离开原点开关；执行“设置位置”；停止
FAST_SLOW_STOP_S	SMC_HOMING_MODE	5	快速移动到原点开关；慢速离开原点开关；停止；执行“设置位置”
FAST_SLOW_I_S_STOP	SMC_HOMING_MODE	6	快速移动到原点开关；慢速离开原点开关；等待 index 脉冲；执行“设置位置”；停止

SMC_INT_STATUS （ENUM）

此枚举变量描述了插补器的各种状态。

表 4.5 插补器状态变量

枚举值	类型	初始值	描述
IPO_UNKNOWN	SMC_INT_STATUS	0	内部状态；
IPO_INIT	SMC_INT_STATUS	1	初始状态；运动尚未开始
IPO_ACCEL	SMC_INT_STATUS	2	插补进行：加速状态
IPO_CONSTANT	SMC_INT_STATUS	3	插补进行：匀速状态
IPO_DECEL	SMC_INT_STATUS	4	插补进行：减速状态
IPO_FINISHED	SMC_INT_STATUS	5	插补完成
IPO_WAIT	SMC_INT_STATUS	6	插补进行：等待状态（例如等待一个停止信号或一个 M 指令）
IPO_INCREASING_ACCEL	SMC_INT_STATUS	7	插补进行：加速状态（加速度值增加）
IPO_DECREASING_ACCEL	SMC_INT_STATUS	8	插补进行：加速状态（加速度值减小）
IPO_INCREASING_DECEL	SMC_INT_STATUS	9	插补进行：减速状态（减速度值增加）
IPO_DECREASING_DECEL	SMC_INT_STATUS	10	插补进行：减速状态（减速度值减小）

MC_TRACK_REF （STRUCT）

表 4.6 MC_TRACK_REF

组件	类型	初始值	描述
OnCompensation	LREAL	0	设置切换开关的提前后延迟，时间为单位：秒
OffCompensation	LREAL	0	设置切换开关的提前后延迟，时间为单位：秒
Hysteresis	LREAL	0	设置滞后时间，用于避开输出开关的输出滞后。这种情况是可能发生的，例如伺服驱动器器刚好在位置切换点，会产生轻微的摆动，导致位置滞后。

MC_OUTPUT_REF （STRUCT）

这个数据类型是库“SM3_Basic.library”的一部分。它定义了 32 个布尔数组形式的输出。

```
TYPE MC_OUTPUT_REF ：
    ARRAY [0..31] OF BOOL;
END_TYPE
```

MC_TA_REF （STRUCT）

它描述了一个 MC_AccelerationProfile 执行的轨迹。

表 4.7 MC_AccelerationProfile 执行轨迹变量

组件	类型	初始值	描述
Number_of_pairs	INT	0	分布点编号
IsAbsolute	BOOL	TRUE	加速度值是绝对或相对
MC_TA_Array	ARRAY[1..100] OF SMC_TA		时间/加速度点

MC_TP_REF （STRUCT）

它描述了一个 MC_PositionProfile 执行的轨迹。

表 4.8 MC_PositionProfile 执行轨迹变量

组件	类型	初始值	描述
Number_of_pairs	INT	0	分布点编号
IsAbsolute	BOOL	TRUE	位置值是绝对或相对
MC_TP_Array	ARRAY[1..100] OF SMC_TP		时间/位置点

MC_TV_REF （STRUCT）

它描述了一个 MC_VelocityProfile 执行的轨迹。

表 4.9　MC_VelocityProfile 执行轨迹变量

组件	类型	初始值	描述
Number_of_pairs	INT	0	分布速度点的编号。
IsAbsolute	BOOL	TRUE	速度值是绝对或相对
MC_TV_Array	ARRAY[1..100] OF SMC_TV		时间/速度点。

SMC_TA （STRUCT）

它定义了轨迹上的一个点，该点由 MC_TA_REF 定义并由一对时间/加速度值组成。

表 4.10　SMC_TA 结构体

组件	类型	初始值	描述
delta_time	TIME	TIME#0ms	到达上一个和当前点之间的时间周期。
acceleration	LREAL	0	在这点上的（绝对-/相对-）加速度，单位：pulse/s2

SMC_TP （STRUCT）

它定义了轨迹上的一个点，该点由 MC_TP_REF 定义并由一对时间/位置值组成。

表 4.11　SMC_TP 结构体

组件	类型	初始值	描述
delta_time	TIME	TIME#0ms	到达上一个点和当前点的时间周期。
position	LREAL	0	在这点上的（绝对-/相对-）位置，单位：pulse

SMC_TV （STRUCT）

它定义了轨迹上的一个点，该点由 MC_TP_REF 定义并由一对时间/速度值组成。

表 4.12　SMC_TV 结构体

组件	类型	初始值	描述
delta_time	TIME	TIME#0ms	到达上一个点和当前点的时间周期。
velocity	LREAL	0	在这点上的（绝对-/相对-）速度，单位：pulse/s

MoveSequence （STRUCT）

连续插补缓存队列结构体。

表 4.13　MoveSequence 结构体

组件	类型	初始值	描述
LineNum	LREAL	0	数据行号
DataType	LREAL	0	数据类型：1：直线运动；2：三点圆弧
X_EndPos	LREAL	0	X 终点坐标，绝对坐标，脉冲数
Y_EndPos	LREAL	0	Y 终点坐标，绝对坐标，脉冲数
Z_EndPos	LREAL	0	Z 终点坐标，绝对坐标，脉冲数
U_EndPos	LREAL	0	U 终点坐标，绝对坐标，脉冲数
Vel	LREAL	100	合成速度，PULL/S
Acc	LREAL	1000	合成加速度，PULL/S2
Aux_Pos	ARRAY[0..1] of LREAL	[0，0]	辅助位置

4.2 指令一览表

单轴状态监控

表 4.14　单轴状态监控

指令	指令名称	功能说明
MC_Power	轴使能	使轴进行可运行状态（轴在可运行状态才能进行控制）
MC_Reset	复位轴	复位轴内部相关错误
MC_ReadStatus	读轴状态	获取轴目前的状态（可能的状态有：未使能、出错、停止过程中、停止、点位运动中、连续运动中、同步运动中、回零中、加速中、减速中、匀速中等等）
MC_ReadAxisError	读轴错误	获取错误编码
MC_ReadParameter	读参数	根据参数 ID 获取参数值
MC_ReadBoolParameter	读布尔参数	根据参数序号获取参数值
MC_WriteParameter	写参数	修改用户指定的特殊参数
MC_WriteBoolParameter	写布尔参数	修改用户指定的特定的布尔变量的参数值
MC_ReadActualPosition	读实际位置	读取当前相关轴的当前位置
MC_ReadActualVelocity	读实际速度	读取当前相关轴的当前速度
MC_ReadActualTorque	读实际转矩	读取当前相关轴的当前力矩
MC_SetPosition	设置位置	用于移动一个轴的坐标系而不引起任何运动
SMC_ReadSetPosition	读取设置位置	读取当前轴的设置位置
SMC_ReadFBError	读取历史错误信息	读取功能块的历史错误信息
SMC_ClearFBError	清除历史错误信息	清除功能块的历史错误信息
SMC_ErrorString	错误码到错误信息	读取该错误码对应的错误描述信息

单轴运动控制

表 4.15　单轴运动控制

指令	指令名称	功能说明
MC_Home	回零	控制总线驱动的回零运动。
LS_EtherCATHomeMove	回零	控制总线驱动的回零运动
MC_MoveAbsolute	绝对运动	实现一个控制轴达到指定绝对位置。
MC_MoveRelative	相对运动	从当前轴的位置将轴移动一个相对位置。
MC_MoveVelocity	定速运动	控制指定轴以一个指定速度持续运行下去，直到被其它指令终止。
MC_Stop	停止	停止控制器运动并将轴的状态设置为“Stopping”状态，轴减速到 0 并停止之后，轴的状态将会转化为状态 “StandStill”。
MC_Halt	暂停	暂停进行中的功能块的执行，暂停运动可被后面的指令终止。
SMC_Inch	步进运动	手动控制轴一段一段的朝指定方向运动。
MC_MoveAdditive	附加运动	控制终端执行机构按给定的速度，加速度移动一段附加的距离。
MC_MoveSuperImposed	叠加运动	前一个运动基础上，叠加速度、加速度运行一段附加的距离。
MC_PositionProfile	位置规划	根据时间—位置规划执行运动。
MC_VelocityProfile	速度规划	根据时间—速度规划执行运动。
MC_AccelerationProfile	加速度规划	根据时间—加速度规划执行运动。
Encoder_SetAxisMove	编码器转轴运动	通过编码器值的变化，转换成从动轴的运动，类似手轮运动。
LS_MoveAbsChangePosVel	在线变速变位置	单轴绝对运动在线变速、变位置。
LS_MoveChangeVel	在线变速	单轴恒速运动，在线变速。
LS_MoveAbs	单轴起跳速度绝对运动	控制轴达到指定绝对位置，有起跳速度。
LS_MoveRel	单轴起跳速度相对运动	控制轴运动指定距离，有起跳速度。
LS_MoveVel	单轴起跳速度恒速运动	控制轴以指定速度连续运动，有起跳速度。
LS_ResetVirtualAxis	单轴复位功能	解除错误和报错信息，恢复轴状态为 StandStill。

同步运动功能

表 4.16　同步运动功能

指令	指令名称	功能说明
MC_GearIn	电子齿轮耦合	设置主从轴齿轮比并启动电子齿轮
MC_GearInPos	电子齿轮平滑耦合	设置主从轴同步距离和齿轮比并启动电子齿轮
MC_GearOut	电子齿轮脱离	断开主从轴电子齿轮
MC_CamTableSelect	凸轮挺杆关联	将选择的 cam 表连接到实际的凸轮表上
MC_CamIn	电子凸轮关联	配置主从轴电子凸轮参数并启动
MC_CamOut	电子凸轮脱离	断开主从轴电子凸轮
SMC_GetTappetValue	读取挺杆状态	读取当前的挺杆状态

轴组运动功能

表 4.17　轴组运动功能

指令	指令名称	功能说明
MC_AddAxisToGroup	添加轴到轴组	将单独的轴添加到轴组之中
MC_RemoveAxisFromGroup	从轴组中移除轴	将轴从轴组中移除
MC_GroupEnable	启用轴组	用于启用指定的轴组
MC_GroupDisable	使轴组无效	指定的轴组变成无效状态，让其它轴组指令无法调用该轴组进行运动控制
MC_GroupReset	轴组复位	用于解除轴组及轴的异常状态
MC_GroupSetPosition	设置轴组指令位置	设置轴组中各轴的指令位置
MC_UngroupAllAxes	解除轴组	这个指令的作用是将某个轴组所包含的轴全部移除，解散该轴组
SMC_GroupPower	轴组使能	使能轴组下的所有轴，等价于轴组下的所有轴调用 MC_Power 指令
SMC_GroupSaveContinueData	保存轴组继续运动数据	保存轴组运动中暂停后的位置数据信息
SMC_GroupReadSetPosition	读取轴组指令位置	读取轴组在指定坐标系下的当前指令位置
SMC_GroupReadSetVelocity	读取轴组的设置速度	读取轴组在指定坐标系下的设置速度
SMC_GroupReadSetAcceleration	读取轴组的设置加速度	读取轴组在指定坐标系下的设置加速度
MC_GroupReadActualPosition	读取轴组反馈位置	读取轴组在指定坐标系下的反馈位置
MC_GroupReadActualVelocity	读取轴组反馈速度	读取轴组在指定坐标系下的反馈速度
MC_GroupReadActualAcceleration	读取轴组反馈加速度	读取轴组在指定坐标系下的反馈加速度
SMC_GroupTargetPosition	读取轴组目标位置	读取轴组在指定坐标系下的目标位置
SMC_GroupConvertPosition	轴组位置转换	轴组中在输入坐标系下的位置信息转换为输出坐标系下的位置
SMC_GroupGetContinuePosition	读取连续运动位置	读取轴组运动中断时的位置
MC_GroupReadConfiguration	读取轴组配置参数	读取轴组包含的轴、数量等配置参数

MC_GroupReadStatus	读取轴组的当前运动状态	用于获取轴组的当前运动状态
MC_GroupReadError	读取轴组错误	获取轴组的错误信息
MC_SetCoordinateTransform	坐标系转换	用来转换不同参考坐标系的指令坐标
MC_SetDynCoordTransform	动态坐标系转换	指定的坐标系相对于 WCS 移动时，需要调用该指令实现坐标系转换
MC_GroupSetOverride	设置轴组超调值	轴组在 Moving 状态下，改变轴组的运动速度
SMC_GroupSetAncillaryAxisLimits	辅助轴限制	限制轴组的运动参数等
SMC_GroupSetAncillaryPathLimits	辅助路径限制	设置轴组的辅助动态路径限制
SMC_SetDynamicLimitFactors	动态限制因子设置	对轴组中所有轴的速度等运动参数做动态限制
MC_SetKinTransform	运动学坐标变换	设置轴组的运动学转换，从 ACS 坐标系到 MCS 坐标系
SMC_SetKinConfiguration	运动学坐标参数配置	配置机构的逆运动学参数
MC_GroupInterrupt	轴组中断	中断当前正在运动的轴组，可通过 MC_GroupContinue 指令继续执行未执行完的运动指令
MC_GroupContinue	轴组继续运行	解除轴组的中断状态，继续执行未完成的指令
MC_GroupHalt	轴组暂停	用于暂停当前的轴组运动
MC_GroupStop	轴组停止	停止轴组的运动
MC_MoveCircularAbsolute	绝对圆弧插补	控制轴组执行绝对位置模式下的圆弧插补运动
MC_MoveCircularRelative	相对圆弧插补	控制轴组执行相对位置模式的圆弧插补运动
MC_MoveDirectAbsolute	绝对位置快速定位	控制轴组内所有轴各自以指定速度运行到绝对位置终点
MC_MoveDirectRelative	相对位置快速定位	控制轴组内所有轴各自以指定速度运行到相对位置的终点
MC_MoveLinearAbsolute	绝对位置直线插补	控制轴组在指定坐标系下的绝对位置模式的直线插补运动
MC_MoveLinearRelative	相对位置直线插补	控制轴组在指定坐标系下的相对位置模式的直线插补运动
SMC_GroupEnableResumeAfterError	错误复位后重启轴组运动	恢复因报错而被中断的轴组状态
SMC_GroupJog	轴组点动	控制轴组在指定坐标系下进行 Jog 运动
SMC_GroupWait	轴组运动延时	设定轴组的延时等待

雷赛专用插补

表 4.18　雷赛专用插补

指令	指令名称	功能说明
LS_2AxisLineAbs	两轴绝对直线插补	绝对坐标两轴直线插补运动
LS_2AxisLineAbs	两轴相对直线插补	相对坐标两轴直线插补运动
LS_3AxisLineAbs	三轴绝对直线插补	绝对坐标三轴直线插补运动
LS_3AxisLineRel	三轴相对直线插补	相对坐标三轴直线插补运动
LS_4AxisLineAbs	四轴绝对直线插补	绝对坐标三轴直线插补、一轴跟随运动

LS_4AxisLineRel	四轴相对直线插补	相对坐标三轴直线插补、一轴跟随运动
LS_5AxisLineAbs	五轴绝对直线插补	绝对坐标三轴直线插补、两轴跟随运动
LS_5AxisLineRel	五轴相对直线插补	相对坐标三轴直线插补、两轴跟随运动
LS_6AxisLineAbs	六轴绝对直线插补	绝对坐标三轴直线插补、三轴跟随运动
LS_6AxisLineRel	六轴相对直线插补	相对坐标三轴直线插补、三轴跟随运动
LS_2AxisLineAbs_Ratio	两轴可调速绝对直线插补	绝对坐标两轴可调速直线插补
LS_2AxisLineRel_Ratio	两轴可调速相对直线插补	相对坐标两轴可调速直线插补
LS_LineFollowAbs	绝对跟随	一轴点位，另一轴跟随绝对位置运动。
LS_LineFollowRel	相对跟随	一轴点位，另一轴跟随相对位置运动。
LS_2AxisCircleAbs	两轴绝对圆弧插补	绝对坐标两轴圆弧插补运动
LS_2AxisCircleRel	两轴相对圆弧插补	相对坐标两轴圆弧插补运动
LS_3AxisCircleAbs	三轴绝对圆弧插补	绝对坐标三轴圆弧插补运动
LS_3AxisCircleRel	三轴相对圆弧插补	相对坐标三轴圆弧插补运动
LS_2AxisEllipsesAbs	两轴绝对椭圆插补	绝对坐标两轴椭圆插补
LS_2AxisEllipsesRel	两轴相对椭圆插补	相对坐标两轴椭圆插补
LS_2AxisCircleAbs_Helical	绝对螺旋插补	绝对坐标三轴圆弧螺旋线插补
LS_2AxisCircleRel_Helical	相对螺旋插补	绝对坐标三轴圆弧螺旋线插补
LS_3AxisGCode	三轴 G 代码插补	三轴 G 代码连续插补（G 代码形式），支持坐标轴 X、Y、Z
LS_3AxisGCode_File	三轴 G 代码插补-文件	三轴 G 代码连续插补（G 代码文件形式），支持坐标轴 X、Y、Z
LS_4AxisGCode	四轴 G 代码插补	四轴 G 代码连续插补（G 代码形式），支持坐标轴 X、Y、Z、U
LS_4AxisGCode_File	四轴 G 代码插补-文件	四轴 G 代码连续插补（G 代码文件形式），支持坐标轴 X、Y、Z、U
LS_4AxisGCodeAxisP	四轴 G 代码插补-P	四轴 G 代码连续插补（G 代码形式），支持坐标轴 X、Y、Z、P
LS_4AxisGCodeAxisP_File	四轴 G 代码插补-文件-P	四轴 G 代码连续插补（G 代码文件形式），支持坐标轴 X、Y、Z、P
LS_6AxisGCodeAxisUVW_file	六轴 G 代码插补-文件	六轴 G 代码连续插补（G 代码文件形式），支持坐标轴 X、Y、Z、U、V、W
LS_6Axis_ZeroOffset	6 轴零点偏移	零点坐标偏移（相当于 G54）

雷赛专用特殊 IO 功能

表 4.19　雷赛专用特殊 IO 功能

指令	指令名称	功能说明	类别
Axis_SetSpecialIOState	设置轴特殊 IO 状态	设置每个轴的特殊 IO 信号（正限位，负限位，原点，报警）的状态。	轴 IO
Axis_SetHardLimit	硬限位设置	设置是否使用本地轴的硬限位	轴 IO
Axis_ELP	正限位信号	读取本地轴正限位信号	轴 IO
Axis_ELN	负限位信号	读取本地轴负限位信号	轴 IO
Axis_n.bSWLimitEnable	软限位设置	设置本地轴（0-AxisNum）是否使用软限位	软限位
Axis_n.fSWLimitPositive	正软限位设置	设置本地轴（0-AxisNum）正软限位值	软限位
Axis_n.fSWLimitNegative	负软限位设置	设置本地轴（0-AxisNum）负软限位值	软限位
Axis_SetServOn	设置伺服轴使能	设置轴的伺服使能信号	伺服专用 IO
Axis_SetSpecialIOState	设置特殊 IO 信号	设置每个轴的特殊 IO 信号（包含 ALRM 信号）	伺服专用 IO
Axis_Alrm	读取轴报警信号	读取报警信号	伺服专用 IO
Axis_ClearWarning	清楚轴报警	设置轴清除伺服报警信号	伺服专用 IO
Encoder_EzClearRead	读取 Ez 清零参数	读取编码器 Z 相清零参数配置	EZ 清零
Encoder_EzClearSet	设置 Ez 清零参数	设置编码器遇到 Z 相清零的参数	EZ 清零
Encoder_SetAxisMove	编码器到轴运动转换	通过编码器的位置值,转换到轴的运动	编码器功能
Encoder_SetVal	设置编码器数值	设置编码器的值，将 SetEncoderVal 值写入编码器	编码器功能
Encoder_SetWorkMode	设置编码器模式	设置编码器的工作模式	编码器功能
CMP_ClearPara	清除比较器参数	清除比较器的参数，清除掉所有状态值，包括比较位置，FIFO 数据等	高速比较
CMP_GetCurState	获取比较器当前状态	获取比较器当前状态	高速比较
CMP_GetLinearPara	获取 Linear 模式下参数	获取 Linear 模式下参数	高速比较
CMP_GetWorkPara	获取比较器的参数	获取比较器的参数	高速比较
CMP_SetCmpPos	设置比较器位置	设置/更新比较器比较位置，在线性模式下，更新起始比较位置；在其他模式下，更新比较位置。	高速比较
CMP_SetFIFOPos	设置 FIFO 时间模式的比较位置	设置 FIFO 模式下比较位置，用于在 FIFO 模式下，同时写入多个位置点，	高速比较

		最多可以输入 128 个比较点;写入数据完成后，bDone 状态为 True。	
CMP_SetFIFOPos_Type2	设置 FIFO 电平模式的比较位置	设置比较位置。用于在 FIFO 模式下，写入多个位置点每个位置点,可以单独配置比较输出状态写入数据完成后，bDone 状态为 True	高速比较
CMP_SetLinearPara	设置 Linear 模式的参数	设置 Linear 模式下参数	高速比较
CMP_SetWorkPara	设置比较器的参数	设置比较器的参数	高速比较
ZeroLatch_SetPara	设置原点锁存参数	设置原点锁存参数	原点锁存
ZeroLatch_GetState	读取原点锁存状态	读取原点锁存状态	原点锁存
EzLatch_SetPara	设置 EZ 锁存参数	设置 EZ 锁存参数	EZ 锁存
EzLatch_GetState	读取 EZ 锁存状态	读取 EZ 锁存状态	EZ 锁存
HighSpeedLatch_SetWorkMode	设置高速锁存参数	设置高速锁存参数	高速锁存
HighSpeedLatch_SetSource	设置高速锁存数据源	设置高速锁存数据源	高速锁存
HighSpeedLatch_ReadState	读取高速锁存状态	读取高速锁存状态	高速锁存
HighSpeedLatch_ReadVal	读取当前锁存值	读取当前锁存值	高速锁存
HighSpeedLatch_FIFOReadVal	读取 FIFO 锁存值	读取当前锁存值，多周期指令	高速锁存
HighSpeedLatch_FIFOReadNumber	读取 FIFO 锁存个数	读取当前锁存的数据个数	高速锁存
PWM_SetVal	设置 PWM 参数	设置 PWM 的参数	PWM

第 5 章　单轴指令

5.1 单轴状态监控和参数设置指令

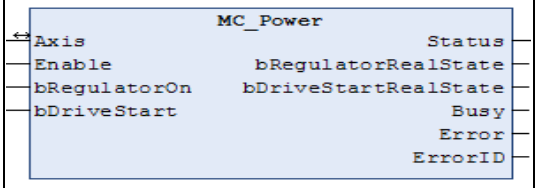
表 5.1　单轴状态监控和参数设置指令

指令名	功能说明
MC_Power	使轴进行可运行状态（轴在可运行状态才能进行控制）
MC_Reset	复位轴内部相关错误
MC_ReadStatus	获取轴目前的状态（可能的状态有：未使能、出错、停止过程中、停止、点位运动中、连续运动中、同步运动中、回零中、加速中、减速中、匀速中等等）
MC_ReadAxisError	获取错误编码
MC_ReadParameter	根据参数 ID 获取参数值
MC_ReadBoolParameter	根据参数序号获取参数值
MC_WriteParameter	修改用户指定的特殊参数
MC_WriteBoolParameter	修改用户指定的特定的布尔变量的参数值
MC_ReadActualPosition	读取当前相关轴的当前位置
MC_ReadActualVelocity	读取当前相关轴的当前速度
MC_ReadActualTorque	读取当前相关轴的当前力矩
MC_SetPosition	用于移动一个轴的坐标系而不引起任何运动
SMC_ReadSetPosition	读取当前轴的设置位置
SMC_ReadFBError	读取功能块的历史错误信息
SMC_ClearFBError	清除功能块的历史错误信息
SMC_ErrorString	读取该错误码对应的错误描述信息

轴使能 MC_Power

用于使能指定轴，使轴进入可运行状态或退出可运行状态，也叫轴使能。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_Power	FB		MC_Power(Axis：= （参数）， Enable：= （参数）， bRegulatorOn：= （参数）， bDriveStart：=（参数）， Status=> （参数）， bRegulatorRealState=> （参数）， bDriveStartRealState=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Enable	有效	BOOL	TRUE FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
bRegulatorOn	使能	BOOL	TRUE FALSE	FALSE	必须设置为 TRUE，以激活功能块的使能状态。
bDriveStart	驱动启用	BOOL	TRUE FALSE	FALSE	必须设置为 TRUE，以关闭功能块的紧急停止处理。
VAR_OUTPUT					
Status	可运行	BOOL	TRUE FALSE	FALSE	轴准备好则为 TRUE。
bRegulatorRealState	使能有效	BOOL	TRUE FALSE	FALSE	轴使能的有效状态。
bDriveStartRealState	驱动可使用	BOOL	TRUE FALSE	FALSE	驱动器没有被快速停止机制中断为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	功能块的处理没有完成为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	发生异常时为 TRUE
ErrorID	错误代码	SMC_ERROR	-	0	正常时数值为 0，发生异常时，输出报错代码。

功能说明：

- 将 Enable 设为 TRUE 时，该功能块进入可运行状态；将 Enable 设为 FALSE，功能块不运行，但是可以执行 MC_Power 指令、MC_Reset（轴错误复位）指令。
- 将 bRegulatorOn 设为 TRUE，如果轴没有错误则轴状态为 standstill。如果轴有错误，轴状态为 errorstop。
- 将 bRegulatorOn 设为 FALSE，轴状态为 Disabled，表明轴没有做好运动准备。
- 将 bDriveStart 设为 FALSE，轴急停。

时序图：

将 Enable、bRegulatorOn 和 bDriveStart 分别置为 TRUE，功能块的 Busy 状态会变成 TRUE，然后 Status 信号也变为 TRUE，该轴变成使能状态。如图 5.1 所示。

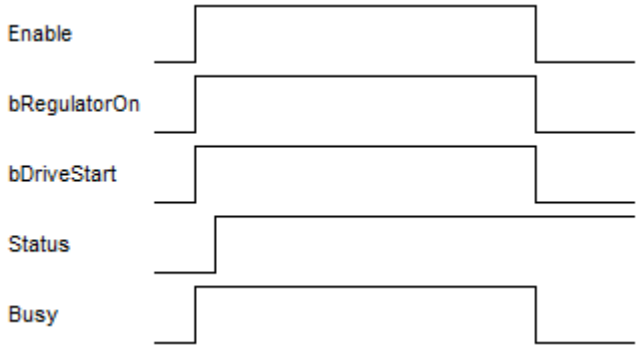


图 5.1　MC_Power 的时序图

轴复位 MC_Reset

用于复位（清除）轴的错误。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_Reset	FB	MC_Reset Axis　Done Execute　Busy Error ErrorID	MC_Reset(Axis：= （参数）， Execute：= （参数）， Done=>（参数）， Busy=>（参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有限范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	输入值的上升沿将启动该功能块的执行。
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	如果复位被执行则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

功能说明：

轴在报错后，将不能继续执行运动指令，只有调用 MC_Reset 指令清除错误之后才能操作轴。MC_Reset 便是用于将轴的状态从 ErrorStop 状态转换成 StandStill 状态，消除轴报错状态，变成可执行状态。

时序图：

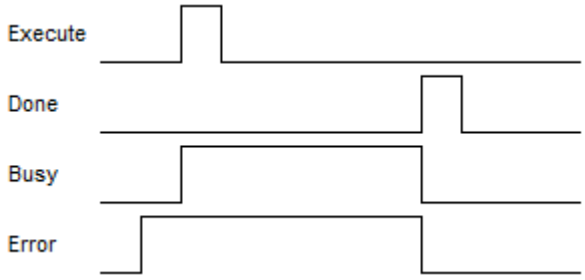


图 5.2　MC_Reset 的时序图

单轴复位功能　LS_ResetVirtualAxis

读取轴状态为错误停止 ErrorStop 和轴结构体错误 FBErrorOccured 时，可调用此功能解除错误和报错信息，恢复轴状态为 StandStill。PMC_SingleAxisLib 库支持，如需添加库文件请参考 STUDIO 编辑器手册。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_ResetVirtualAxis	FB	LS_ResetVirtualAxis Axis　Done Busy Error Execute　ErrorID	LS_ResetVirtualAxis(Axis:= （参数）, Execute:= （参数）, Done=> （参数）, Busy=> （参数）, Error=> （参数）, ErrorID=> （参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始化	注释
Axis	轴	AXIS_REF_SM3	—	—	指定轴
Execute	启动	BOOL	TRUE FALSE	FALSE	启动，上升沿有效
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	运动完成状态
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误； True-输入参数越界
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

例程：

此例使用软限位触发 ErrorStop 和 FBErrorOccured，延时 1S 执行此功能块，恢复 StandStill。具体代码见例程“RESETAXIS1”，运行结果如图 5.3、5.4 所示。

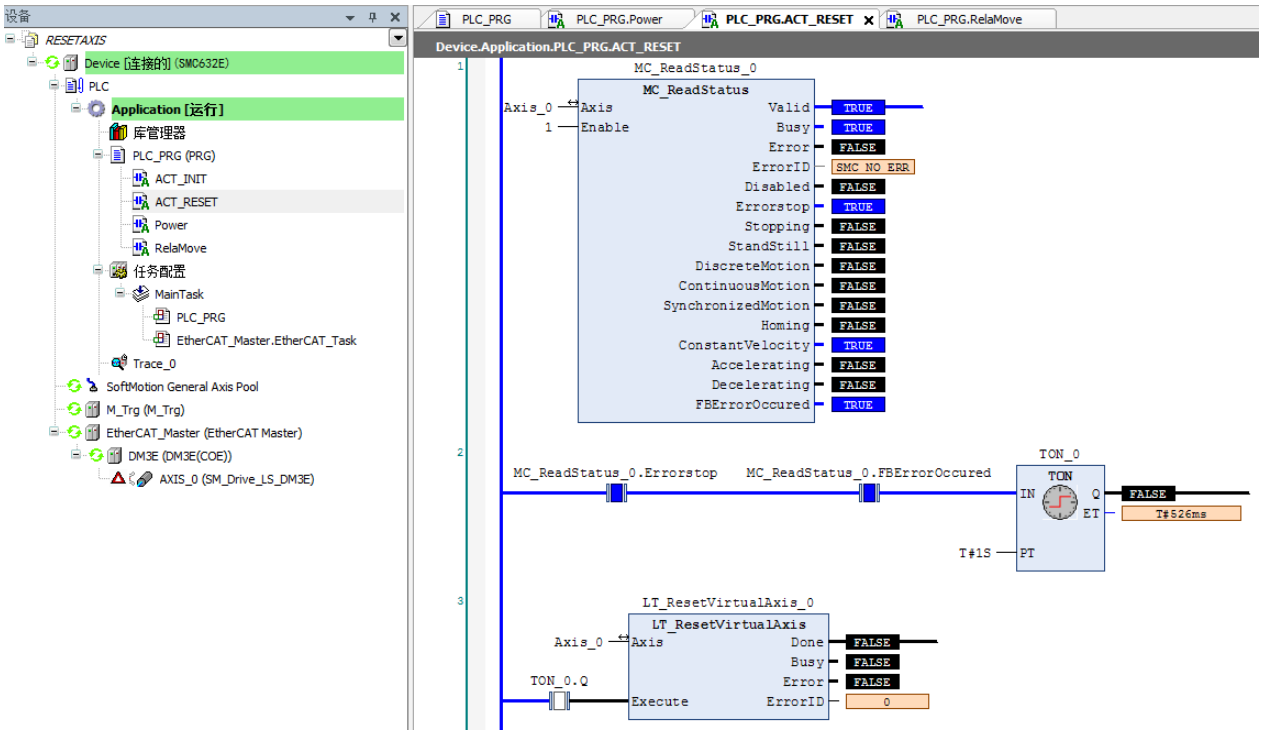


图 5.3　程序界面

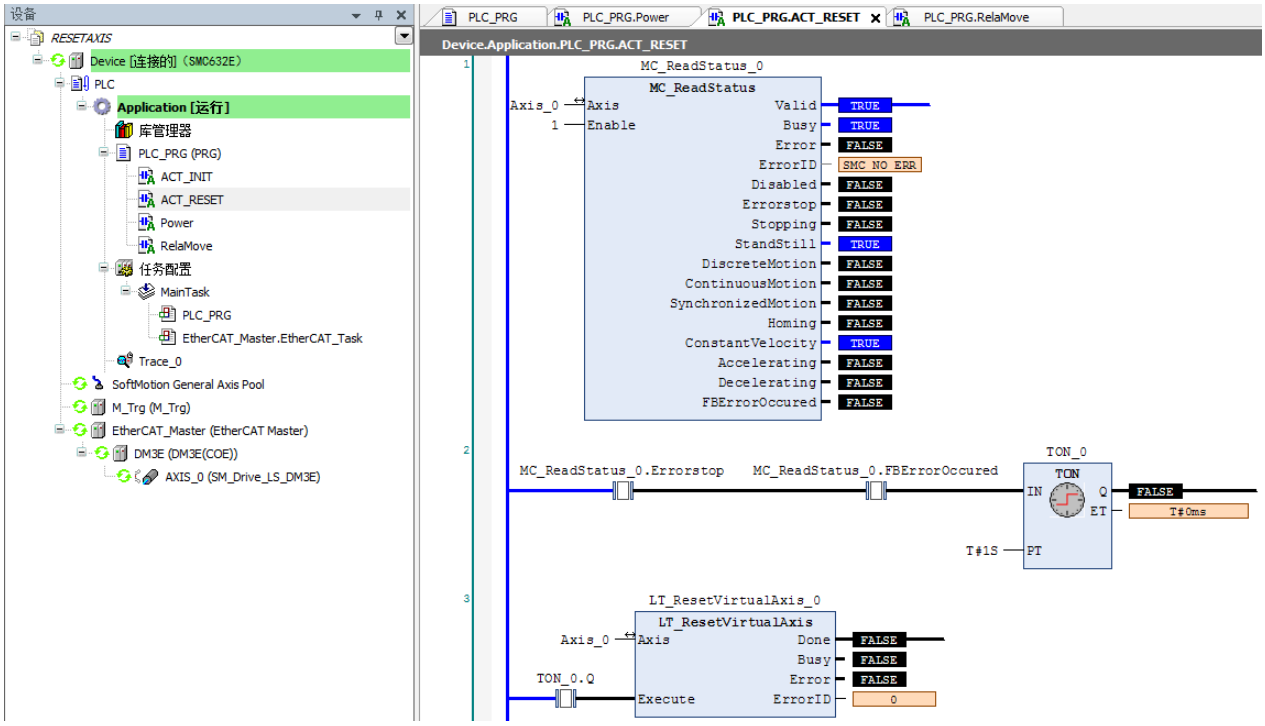


图 5.4　运行结果

读轴状态 MC_ReadStatus

用于读取轴的详细状态。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_ReadStatus	FB	MC_ReadStatus Axis — Valid Enable — Busy Error ErrorID Disabled Errorstop Stopping StandStill DiscreteMotion ContinuousMotion SynchronizedMotion Homing ConstantVelocity Accelerating Decelerating FBErrorOccured	MC_ReadStatus(Axis：= （参数）， Enable：= （参数）， Valid=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）， Disabled=> （参数）， Errorstop=> （参数）， Stopping=> （参数）， StandStill=> （参数）， DiscreteMotion=> （参数）， ContinuousMotion=> （参数）， SynchronizedMotion=> （参数）， Homing=> （参数）， ConstantVelocity=> （参数）， Accelerating=> （参数）， Decelerating=> （参数）， FBErrorOccured=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Enable	有效	BOOL	TRUE FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
VAR_OUTPUT					
Valid	有效	BOOL	TRUE FALSE	FALSE	如果轴准备好则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
Disabled	未使能	BOOL	TRUE FALSE	FALSE	如果轴状态为 Disabled 则为 TRUE
Errorstop	报错	BOOL	TRUE FALSE	FALSE	如果轴状态为 Errorstop 则为 TRUE
Stopping	停止中	BOOL	TRUE FALSE	FALSE	如果轴状态为 Stopping 则为 TRUE
StandStill	准备好	BOOL	TRUE FALSE	FALSE	如果轴状态为 Standstill 则为 TRUE
DiscreteMotion	点位运动	BOOL	TRUE FALSE	FALSE	如果轴状态为 DiscreteMotion 则为 TRUE
ContinuousMotion	连续运动	BOOL	TRUE FALSE	FALSE	如果轴状态为 ContinuousMotion 则为 TRUE
SynchronizedMotion	同步运动	BOOL	TRUE FALSE	FALSE	如果轴状态为 SynchronizedMotion 则为 TRUE
Homing	回零中	BOOL	TRUE FALSE	FALSE	如果轴状态为 Homing 则为 TRUE
ConstantVelocity	恒速运动	BOOL	TRUE FALSE	FALSE	如果电机以恒定速度移动则为 TRUE
Accelerating	加速中	BOOL	TRUE FALSE	FALSE	如果电机正在加速则为 TRUE
Decelerating	减速中	BOOL	TRUE FALSE	FALSE	如果电机正在减速则为 TRUE
FBErrorOccured	FB 报错	BOOL	TRUE FALSE	FALSE	如果功能块错误被探测到并且还没有由 SMC_ClearFBError 清除，则为 TRUE

说明：

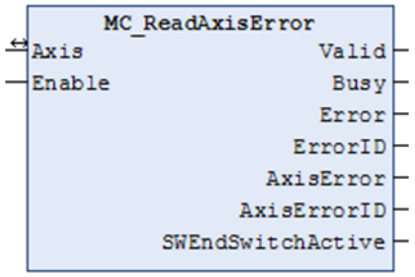
可以用来读取轴的状态，并将状态引用到对应的程序步骤之中。

Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的状态。

读轴错误 MC_ReadAxisError

用于读取轴的错误。

指令外观：

指令	FB/ FUN	图形模块	结构文本
MC_ReadAxisError	FB		MC_ReadAxisError(Axis：= （参数）， Enable：= （参数）， Valid=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）， AxisError=> （参数）， AxisErrorID=> （参数）， SWEndSwitchActive=>);

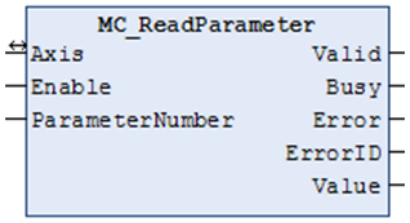
变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Enable	有效	BOOL	TRUE FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
VAR_OUTPUT					
Valid	有效	BOOL	TRUE FALSE	FALSE	如果轴准备好则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
AxisError	轴报错	BOOL	TRUE FALSE	FALSE	轴错误的标志
AxisErrorID	轴错误值	DWORD	0 正数	0	轴错误值
SWEndSwitchActive	软限位	BOOL	TRUE FALSE	FALSE	如果超过软限位则为 TRUE。

读参数 MC_ReadParameter

用于读取指定的参数值。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_ReadParameter	FB		MC_ReadParameter(Axis：= （参数）， Enable：= （参数）， ParameterNumber:=（参数）， Valid=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）， Value=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Enable	启动	BOOL	TRUE，FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
ParameterNumber	轴参数号	DINT	0 正数	0	访问轴参数的索引和子索引和序号
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE，FALSE	FALSE	如果参数已读取则为 TRUE。
Busy	执行中	BOOL	TRUE，FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
Value	参数值	LREAL	负数，0，正数	0	读取参数的值

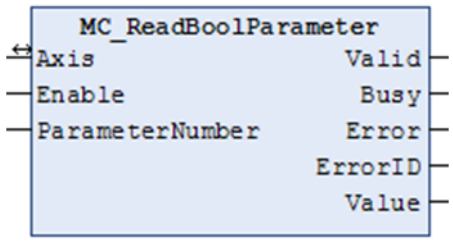
功能说明：

- 可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到 ParameterNumber 参数具体的参数序号所代表的参数用途，或参考 4.1.1 节。
- Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的参数。

读布尔参数 MC_ReadBoolParameter

用于读取指定的 BOOL 型变量的值。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_ReadBoolParameter	FB		MC_ReadBoolParameter(Axis：= （参数）， Enable：= （参数）， ParameterNumber：= （参数）， Valid=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）， Value=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Enable	启动	BOOL	TRUE，FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
ParameterNumber	轴参数号	DINT	0 正数	0	参数编号。
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE，FALSE	FALSE	如果参数值已获取则为 TRUE。
Busy	执行中	BOOL	TRUE，FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，SMC_Error。
Value	参数值	BOOL	TRUE FALSE	FALSE	读取参数的值

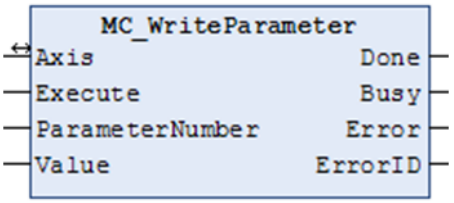
功能说明：

- 可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到 ParameterNumber 参数具体的参数序号所代表的参数用途，或参考 4.1.1 节。
- Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的参数。

写参数 MC_WriteParameter

用于写入指定的参数值。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_WriteParameter	FB		MC_WriteParameter(Axis：= （参数）， Execute：= （参数）， ParameterNumber：= （参数）， Value：= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	输入值的上升沿将启动该功能块的执行。
ParameterNumber	参数号	DINT	0 正数	0	参数 ID
Value	参数值	LREAL	负数，0，正数	0	需要写入的值。
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	如果参数被成功写入则为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

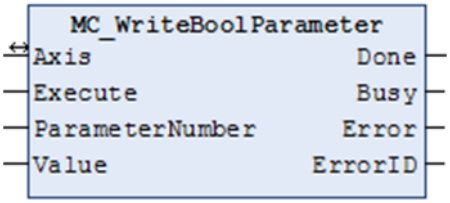
功能说明：

可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到 ParameterNumber 参数具体的参数序号所代表的参数用途，或参考 4.1.1 节。

写布尔参数 MC_WriteBoolParameter

用于写入 BOOL 类型的参数值。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_WriteBoolParameter	FB		MC_WriteBoolParameter(Axis：= （参数）， Execute：= （参数）， ParameterNumber：= （参数）， Value：= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	输入值的上升沿将启动该功能块的执行。
ParameterNumber	参数号	DINT	0 正数	0	参数 ID
Value	参数值	LREAL	负数，0，正数	0	需要写入的值。
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	如果参数值已被写入成功则为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

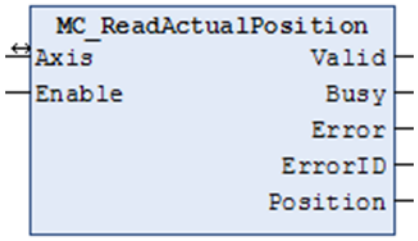
功能说明：

可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到 ParameterNumber 参数具体的参数序号所代表的参数用途，或参考 4.1.1 节。

读实际位置 MC_ReadActualPosition

用于读取轴当前实际位置值。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_ReadActualPosition	FB		MC_ReadActualPosition (Axis：= （参数）， Enable：= （参数）， Valid=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）， Position=> （参数）);

变量

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Enable	有效	BOOL	TRUE FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE FALSE	FALSE	如果输出值有效，则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
Position	位置	LREAL	负数，0，正数	0	新的绝对位置（以轴的单位表达[u]）。

功能说明：

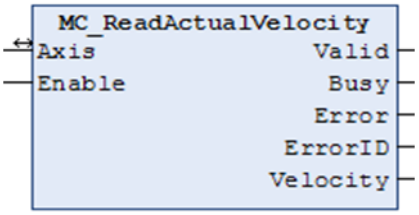
这个指令用来读取驱动器中轴的实际位置，即 fActPosition。

Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的状态。

读实际速度 MC_ReadActualVelocity

用于读取轴的实际速度值。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_ReadActualVelocity	FB		MC_ReadActualVelocity(Axis: = （参数）, Enable: = （参数）, Valid=> （参数）, Busy=> （参数）, Error=> （参数）, ErrorID=> （参数）, Velocity=> （参数）);

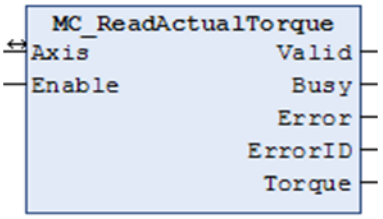
变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Enable	有效	BOOL	TRUE FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理。
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE FALSE	FALSE	如果输出值有效，则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
Velocity	速度	LREAL	负数，0，正数	0	实际速度的值（以[计数单位/秒]表达）。

读实际力矩 MC_ReadActualTorque

用于读取轴实际转矩值。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_ReadActualTorque	FB		MC_ReadActualTorque(Axis：= （参数）， Enable：= （参数）， Valid=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）， Torque=> （参数）)；

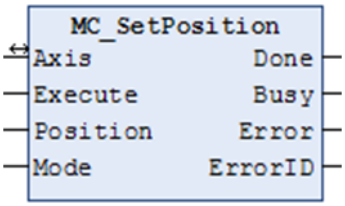
变量：

VAR_IN_OUT	类型	初始值	描述
Axis	AXIS_REF		
VAR_INPUT			
Enable	BOOL	FALSE	必须设置为 TRUE，以激活功能块的处理。
VAR_OUTPUT			
Valid	BOOL	FALSE	如果有效输出可用则为 TRUE。
Busy	BOOL	FALSE	当功能块执行还没结束时为 TRUE。
Error	BOOL	FALSE	功能块执行错误
ErrorID	SMC_ERROR	0	错误指示，见 SMC_Error。
Torque	LREAL	0	实际转矩或力的值（以计数单位表达）。

设置位置 MC_SetPosition

用于设置轴当前的位置（不会引起轴的运动）。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_SetPosition	FB		MC_SetPosition(Axis：= （参数）， Execute：= （参数）， Position：= （参数）， Mode：= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）)；

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	输入值的上升沿将启动该功能块的执行。
Position	位置	LREAL	负数，0，正数	0	位置单位[u]（表示“距离”如果 Mode = RELATIVE）
Mode	模式	BOOL	TRUE FALSE	FALSE	TRUE = 相对， FALSE = 绝对
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	功能块执行完成为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

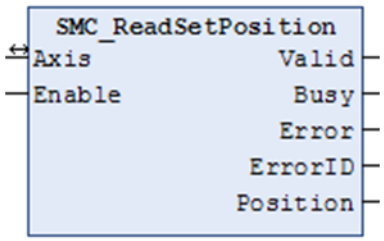
功能说明：

Execute 的上升沿将触发这个指令的执行。

读取设置位置 SMC_ReadSetPosition

用于读取当前轴的设置位置。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_ReadSetPosition	FB		SMC_ReadSetPosition(Axis: = （参数）, Enable: = （参数）, Valid=> （参数）, Busy=> （参数）, Error=> （参数）, ErrorID=> （参数）, Position=> （参数）);

变量：

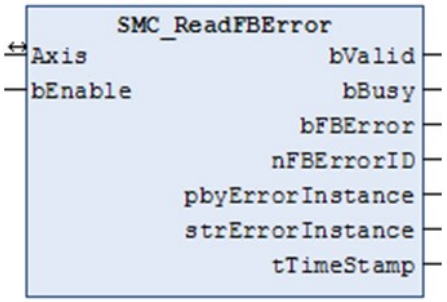
VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Enable	有效	BOOL	TRUE FALSE	FALSE	必须被设为 TRUE 以使功能块能够执行。
VAR_OUTPUT					

Valid	获取标志	BOOL	TRUE FALSE	FALSE	如果参数有效则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
Position	位置	LREAL	负数，0，正数	0	设置的位置值

读功能块错误信息 SMC_ReadFBError

用于读取轴功能块的错误信息。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_ReadFBError	FB		SMC_ReadFBError(Axis:= （参数）， bEnable:= （参数）， bValid=> （参数）， bBusy=> （参数）， bFBError=> （参数）， nFBErrorID=> （参数）， pbyErrorInstance=> （参数）， strErrorInstance=> （参数）， tTimeStamp=> （参数）);

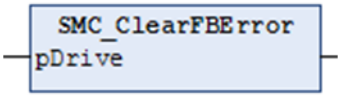
变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
bEnable	有效	BOOL	TRUE FALSE	FALSE	必须被设为 TRUE 以使功能块能够执行。
VAR_OUTPUT					
bValid	获取标志	BOOL	TRUE FALSE	FALSE	如果参数有效则为 TRUE。
bBusy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
bFBError	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
nFBErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。
pbyErrorInstance	错误指针	POINTER TO BYTE	-	-	指针指向报错的 FB
strErrorInstance		STRING	-	-	
tTimeStamp	时间戳	TIME	-	0	错误发生的时间戳

清除功能块错误 SMC_ClearFBError

清除功能块的历史错误信息。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_ClearFBError	FUN		SMC_ClearFBError(pDrive: = （参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
pDrive	轴指针	POINTER TO AXIS_REF_SM3	-	-	该输入为指向轴结构体的地址指针。

功能说明：

当轴出现错误，调用复位功能块将轴复位后，需要调用该功能块清除轴的历史错误状态。Execute 的上升沿将触发这个指令的执行。

5.2 单轴运动指令

单轴运动指令主要包含的指令如表 5.2 所示。

表 5.2　单轴运动指令

指令名	功能说明
MC_Home	控制总线驱动的回零运动。
LS_EtherCATHomeMove	控制总线驱动的回零运动
MC_MoveAbsolute	实现一个控制轴达到指定绝对位置。
MC_MoveRelative	从当前轴的位置将轴移动一个相对位置。
MC_MoveVelocity	控制指定轴以一个指定速度持续运行下去，直到被其它指令终止。
MC_Stop	停止控制器运动并将轴的状态设置为 “Stopping”状态，轴减速到 0 并停止之后，轴的状态将会转化为状态 “StandStill”。
MC_Halt	暂停进行中的功能块的执行，暂停运动可被后面的指令终止。
SMC_Inch	手动控制轴一段一段的朝指定方向运动。
MC_MoveAdditive	控制终端执行机构按给定的速度，加速度移动一段附加的距离。
MC_MoveSuperImposed	前一个运动基础上，叠加速度、加速度运行一段附加的距离。
MC_PositionProfile	根据时间—位置规划执行运动。
MC_VelocityProfile	根据时间—速度规划执行运动。
MC_AccelerationProfile	根据时间—加速度规划执行运动。

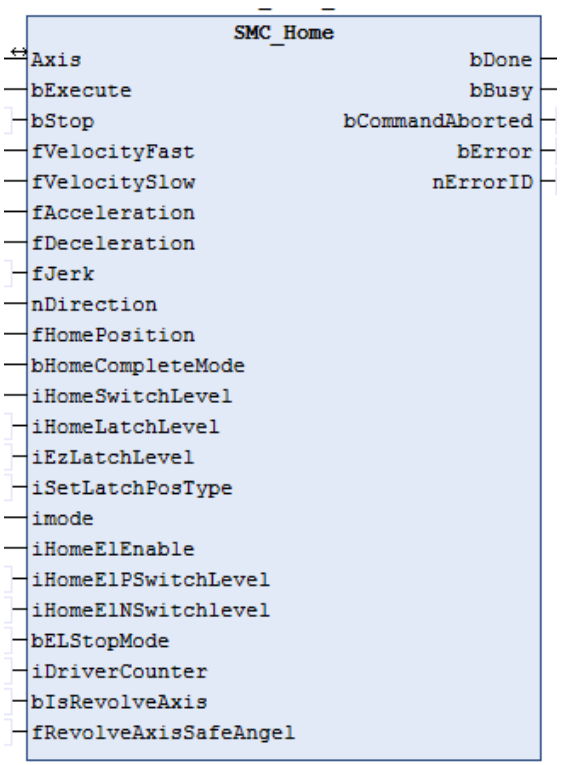
Encoder_SetAxisMove	通过编码器值的变化，转换成从动轴的运动，类似手轮运动。
LS_MoveAbsChangePosVel	单轴绝对运动在线变速、变位置。
LS_MoveChangeVel	单轴恒速运动，在线变速。
LS_MoveAbs	控制轴达到指定绝对位置，有起跳速度。
LS_MoveRel	控制轴运动指定距离，有起跳速度。
LS_MoveVel	控制轴以指定速度连续运动，有起跳速度。
LS_ResetVirtualAxis	解除错误和报错信息，恢复轴状态为 StandStill。

5.2.1 回零

脉冲电机回零 SMC_HOME

此指令执行脉冲电机回零动作，具体回零过程由总线驱动设计的回零模式决定。

指令外观：

指令	FB/ FUN	图形模块	结构文本
SMC_Home	FB		SMC_Home(Axis: = , bExecute: = , bStop: = , fVelocityFast: = , fVelocitySlow: = , fAcceleration: = , fDeceleration: = , fJerk: = , nDirection: = , fHomePosition: = , bHomeCompleteMode: = , iHomeSwitchLevel: = , iHomeLatchLevel: = , iEzLatchLevel: = , iSetLatchPosType: = , imode: = , iHomeElEnable: = , iHomeElPSwitchLevel: = , iHomeElNSwitchlevel: = , bELStopMode: = , iDriverCounter: = , bIsRevolveAxis: = , fRevolveAxisSafeAngel: = , bDone=> , bBusy=> , bCommandAborted=> , bError=> , nErrorID=>);

变量：

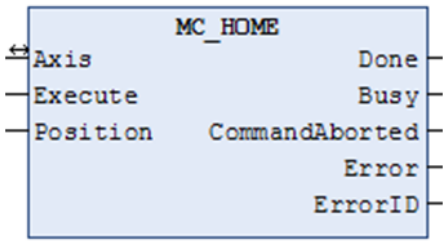
VAR_IN_OUT	名称	类型	初始值	描述
Axis	轴	AXIS_REF_VIRTUAL_SM3	—	执行回零的轴
VAR_INPUT				
bExecute	启动	BOOL	FALSE	启动回零，回零过程中该信号需要保持，回零完成再清除
bStop	停止	BOOL	FALSE	
fVelocityFast	高速	LREAL	1000	
fVelocitySlow	低速	LREAL	100	
fAcceleration	加速度	LREAL	1000	
fDeceleration	减速度	LREAL	1000	
fJerk	加加速度	LREAL	90000000	
nDirection	回零方向	MC_Direction	MC_Direction.positive	1 表示正向， -1 表示负向
fHomePosition	回零位置偏移	LREAL	0	
bHomeCompleteMode	回零完成模式	INT	0	0：回零结束，位置不做处理 1：回零结束，在当前位置清零 2：回零结束，先运动到 fHomePosition 位置，然后清零。
iHomeSwitchLevel	原点信号有效电平	UINT	0	
iHomeLatchLevel	原点锁存信号有效电平	UINT	0	
iEzlatchLevel	Ez 锁存信号有效电平	UINT	0	
iSetLatchPosType	锁存位置源类型	UINT	0	
Imode	回零模式	INT	0	0：一次回零 1：一次回零加反找 2：二次回零 6：原点锁存 7：原点锁存+同向 EZ 锁存 8：单独 EZ 回零 9：原点锁存+反向 EZ 锁存 10：两次原点锁存回零
iHomeElEnable	限位信号有效使能	BOOL	True	
iHomeElPSwitchLevel	正限位有效电平	BOOL	True	
iHomeElNSwitchLevel	负限位有效电平	BOOL	True	
bELStopMode	限位停止模式	BOOL	False	False：急停 True：减速停

iDriverCounter	停止周期数	DINT	200	驱动器停止周期数
bIsRevolveAxis	保留参数			
fRevolveAxisSafeAngel	保留参数			
VAR_OUTPUT				
bDone	完成	BOOL	FALSE	如果回零已完成则为 TRUE。
bBusy	执行中	BOOL	FALSE	回零运动执行中。
bCommandAborted	命令中断	BOOL	FALSE	回零运动终止。当 bstop 输入为 True 后，运动结束，该位被置为 True。
bError	错误	BOOL	FALSE	回零运动出错
nErrorID	错误代码	DINT	0	错误码

总线电机回零　MC_HOME

此指令执行总线电机回零动作，具体回零过程由总线驱动设计的回零模式决定。

指令外观:

指令	FB/FUN	图形模块	结构文本
MC_Home	FB		MC_Home(Axis:= （参数）, Execute:= （参数）, Position:= （参数）, Done=> （参数）, Busy=> （参数）, CommandAborted=> （参数）, Error=> （参数）, ErrorID=> （参数）);

变量:

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	执行回零的轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运行
Position	位置	LREAL	负数，0，正数	0	驱动器的回零偏置 16#607C
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	如果回零已完成则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	命令中断	BOOL	TRUE FALSE	FALSE	如果该命令已被其他命令终止则为 TRUE。

Error	错误	BOOL	TRUE FALSE	FALSE	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

说明:

- 这个指令由“SM3_Basic”库实现。
- 这是总线驱动的回零运动指令，当所控制的运动轴是一个总线轴时，需要调用这个指令，以实现运动轴的回零。
- 此回零指令的回零模式由连接在总线上的从站决定，控制端只是将需要的回零参数发给驱动器，具体的回零动作，在驱动器端完成。
- 一般驱动器的回零偏移地址为 16#607C，各驱动器的处理不同，具体请参考该驱动器的回零流程及回零参数说明
- 在执行回零之前，需要配置总线驱动的回零参数，如回零模式、速度、加速度等。总线驱动回零需要配置哪些参数，请参考所使用驱动的手册。
- 一般的总线驱动回零，需要设置索引和子索引的数据如表 5.3 所示。

表 5.3　总线回零设置

参数	索引	子索引	描述
回零方式	0x6098		各家驱动器回零方式有所差异，需要根据具体的驱动厂家手册，选择相应的回零方式
回零高速	0x6099	0x01	开始回零到找到零点过程的速度，数值较高 ， 以减少回零时间
回零低速	0x6099	0x02	找到零点到回零完成过程的速度，数值较低，以提高精度
回零加减速	0x609A		在原点回归时的加减速变化

- 回零模式的三种方式：

1） 控制器通过 SDO 服务向驱动器写入回零参数。具体的程序及用法，请参考例程“LS_EtherCATAxis_HomeMove”；

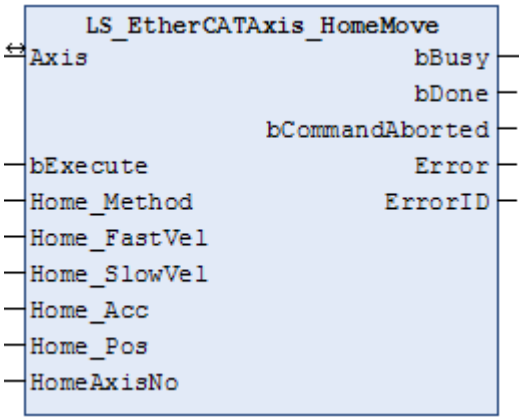
2） 通过配置 PDO 参数并映射相关回零参数进行回零设置。具体的程序及用法，请参考例程“MC_Home By PDO”；

3） 通过配置启动参数向驱动器写入回零参数。具体的程序及用法，请参考例程“MC_Home By StartPara”。

SDO 回零 LS_EtherCATAxis_HomeMove

EtherCAT 总线驱动回零模块，通过 SDO 方式进行回零。

指令外观：

指令	FB/ FUN	图形模块	结构文本
LS_EtherCATAxis_HomeMove	FB		LS_EtherCATAxis_HomeMove(Axis: = , bExecute: = , Home_Method: = , Home_FastVel: = , Home_SlowVel: = , Home_Acc: = , Home_Pos: = , HomeAxisNo: = , bBusy=> , bDone=> , bCommandAborted=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	执行回零的轴
VAR_INPUT					
bExecute	启动	BOOL	TRUE FALSE	FALSE	启动回零，回零过程中该信号需要保持，回零完成再清除
Home_Method	方式	DWORD	-	21	回零方式，请参考各驱动支持的回零方式（回零方式对象字典索引为 0x6098，子索引 0x00）
Home_FastVel	快速	DWORD	0 正数	500	回零快速，对象字典索引为 0x6099，子索引 0x01
Home_SlowVel	慢速	DWORD	0 正数	100	回零慢速，对象字典索引为 0x6099，子索引 0x02
Home_Acc	加速度	DWORD	0 正数	1000	回零加减速，对象字典索引为 0x609A，子索引 0x00
Home_Pos	偏移值	DWORD	负数，0，正数	0	回零后偏移值，对象字典索引为 0x607C，子索引 0x00
HomeAxisNo	驱动下轴号	DWORD	0，1	0	轴编号。0—驱动或模块下 0 轴；1—驱动或模块下 1 轴
VAR_OUTPUT					
bBusy	执行中	BOOL	TRUE FALSE	FALSE	回零过程中
bDone	完成	BOOL	TRUE FALSE	FALSE	回零完成
bCommandAborted	命令中断	BOOL	TRUE FALSE	FALSE	回零取消
Error	错误	BOOL	TRUE FALSE	FALSE	回零出错
ErrorID	错误代码	SMC_ERROR	-	0	回零错误码

说明：

- 这个指令由“PMC_SingleAxisLib”库实现。
- 调用该指令时，需要正确配置对象字典：回零模式 6098、回零速度 6099、回零加减速 609A、回零偏移值 607C，具体的参数说明需要参考所使用的总线驱动。
- 此回零指令的回零模式由连接在总线上的从站决定，控制端只是将需要的回零参数发给驱动器，具体的回零动作，在驱动器端完成。

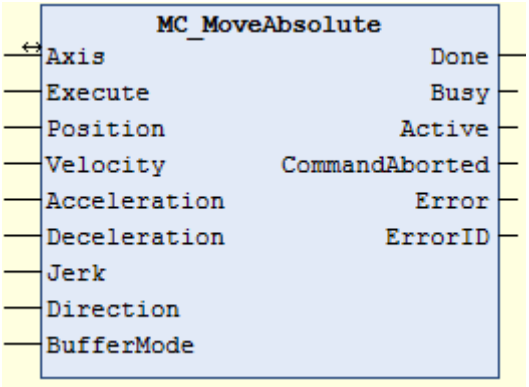
5.2.2 基础运动指令

基础指令是指单轴基础运动控制指令包括：绝对位置指令、相对位置指令、恒速运动指令、停止运动指令、暂停运动指令、点动运动指令、寸动运动指令、位置叠加指令、位置速度叠加指令。

绝对位置 MC_MoveAbsolute

该指令用于实现将控制轴按照设定参数，运动到指定的绝对位置。

指令外观：

指令	FB/ FUN	图形模块	结构文本
MC_MoveAbsolute	FB		MC_MoveAbsolute(Axis：= （参数）， Execute：= （参数）， Position：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Direction：= （参数）， BufferMode：= （参数）， Done=> （参数）， Busy=> （参数）， Active=> （参数）， CommandAborted=>（参数）， Error=> （参数）， ErrorID=> （参数） ）；

变量：

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
输入变量	**名称**	**数据类型**	**有效范围**	**初始值**	**描述**
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Position	位置	LREAL	负数，0，正数	0	运动的目标位置 [u]（负或正）。
Velocity	目标速度	LREAL	负数，0，正数	0	速度最大值（总是为正） [u/s]
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值（总是为正）[u/s2]
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值（总是为正）[u/s2]
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值（总是为正）[u/s3]
Direction	方向	MC_DIRECTION	-1,1	Positive	1，Positive，正； -1，Negative，负
BufferMode	缓存模式	MC_BUFFER_MODE	0-5	0	指定多重启动运动指令时的动作。 0： mcAborting，中断； 1： mcBuffered，等待； 2： mcBlendingLow，以低速合并； 3： mcBlendingPrevious，以前一个速度合并； 4： mcBlendingNext，以后一个速度合并； 5： mcBlendingHigh，以高速合并
输出变量	**名称**	**数据类型**	**有效范围**	**初始值**	**描述**
Done	执行完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	功能块的处理没有完成为 TRUE
Active		BOOL			该轴在工作
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	指令执行被中断时为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	发生异常时为 TRUE
ErrorID	错误代码	SMC_ERROR	-	0	正常时数值为 0，发生异常时，输出报错代码

功能说明：

- 这个指令由“SM3_Basic”库实现。
- 如果是线性轴的绝对点位运动，方向值将被忽略。
- 当速度曲线是梯形曲线时，指令执行时的“速度-时间”曲线如图 5.5 所示。

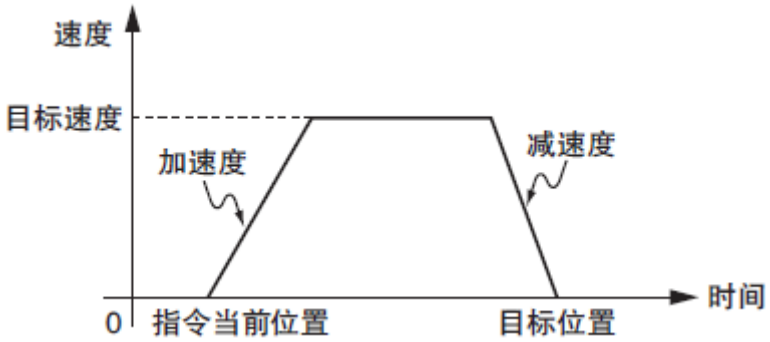


图 5.5　梯形速度曲线

- 如果距离过短，可能达不到最大速度。
- 将加减速度设定为 0 后，将不做加减速而直接达到目标速度。
- 具体的程序及用法，请参考例程“MC_MoveAbsolute”。

例程 MC_MoveAbsolute

实现功能：

图 5.6 是两个 MC_MoveAbsolute 功能块的实例“First”和“Second”连接的例程：

时序图的前段画出了“Second”在“First”之后调用的情况。当“First”达到要求的位置 6000（速度为 0）时，输出 Done 将引发“Second”向位置 10000 移动。

时序图的后段画出了当“First”还在执行时“Second”被启动的情况。这时，“First”的运动在速度恒定期间被 Test 信号终止。“Second”功能块将直接移向位置 10000，虽然并没有到达位置 6000。

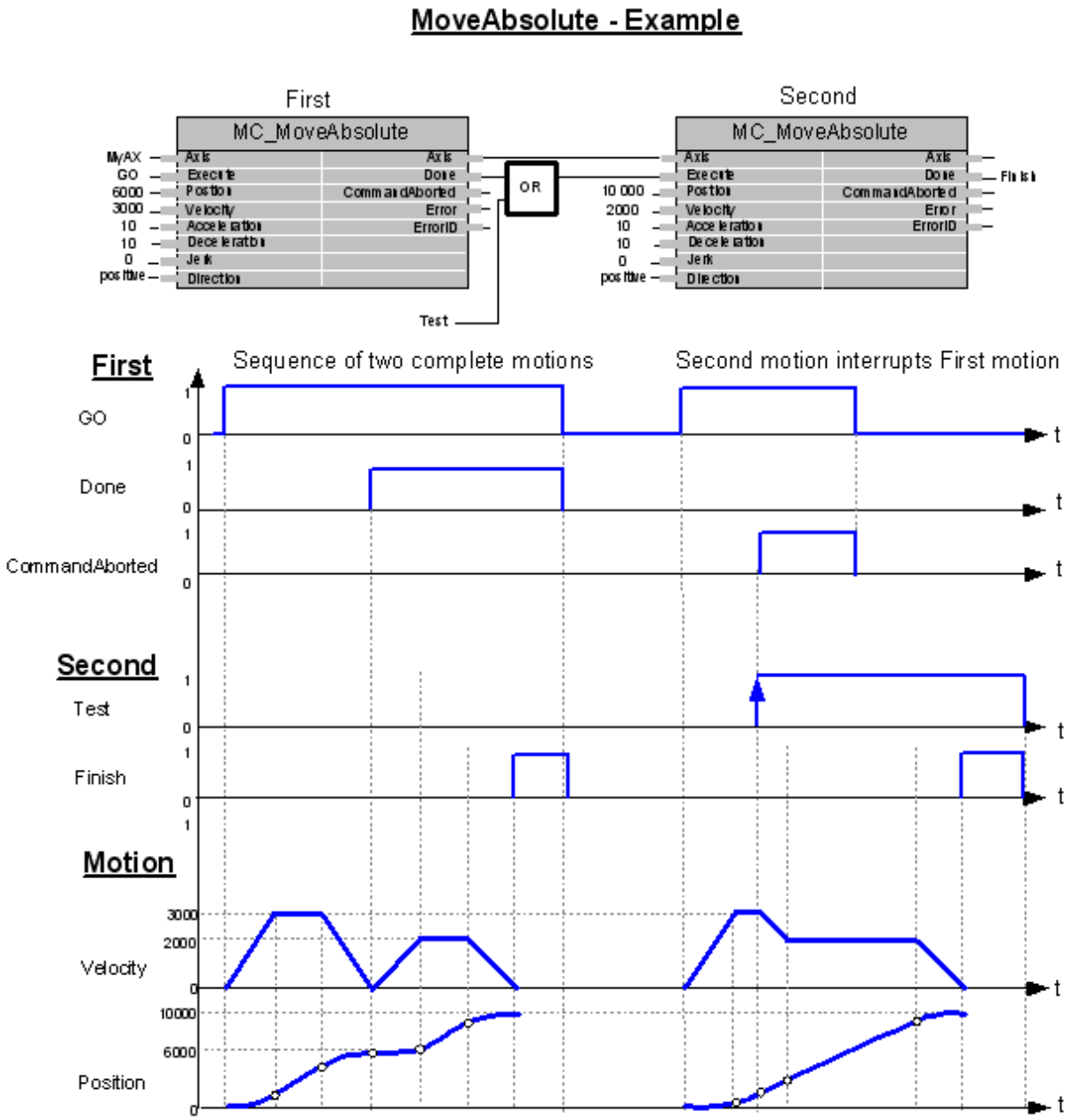
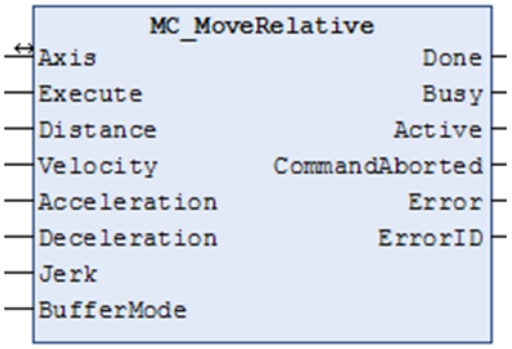


图 5.6　绝对位置运动

相对位置 MC_MoveRelative

控制轴按照设定参数运动一段相对距离。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_MoveRelative	FB		MC_MoveRelative(Axis：= （参数）， Execute：= （参数）， Distance：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， BufferMode：= （参数）， Done=> （参数）， Busy=> （参数）， Active=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Distance	距离	LREAL	负数，0，正数	0	目标的与当前位置的相对距离
Velocity	目标速度	LREAL	负数，0，正数	0	目标速度
Acceleration	目标加速度	LREAL	负数，0，正数	0	启动的加速度
Deceleration	目标减速度	LREAL	负数，0，正数	0	停止的减速度值
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度
BufferMode	缓存模式	MC_BUFFER_MODE			
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Active		BOOL			该轴在工作
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	运动异常终止
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

功能说明：

- 这个指令由“SM3_Basic”库实现。

- 以当前位置为原点做相对运动，终点坐标即为起点到终点的距离。
- 当速度曲线是梯形曲线时，指令执行时的“速度-时间”如图 5.7 所示。
- 具体的程序及用法，请参考例程“MC_MoveRelative”。

例程 MC_MoveRelative：

图 5.7 是两个“MC_MoveRelative”功能块的实例“First”和“Second”连接的例程。

1）时序图的前段为“Second”在“First”之后调用的情况。当“First”达到要求的位置 6000（速度为 0）时，输出 Done 将引发“Second”向位置 10000 移动。

2）时序图的后段为“First”还在执行时“Second”被启动的情况。这时，“First”在速度恒定期间被 Test 信号终止。将距离 4000 加上实际位置 3250 后，“Second”功能块将把轴移向得到的新位置 7250。

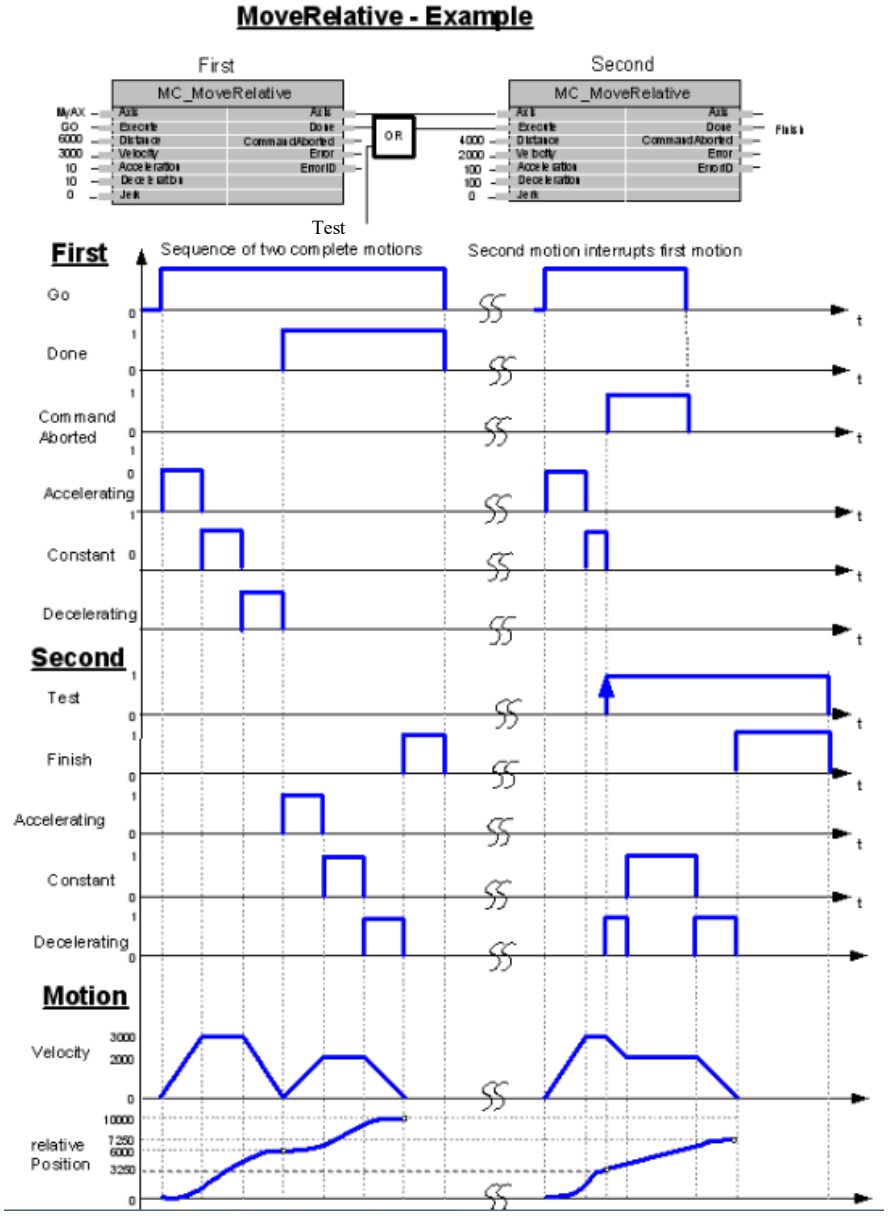
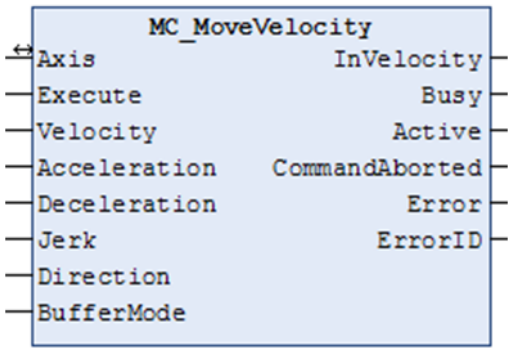


图 5.7　相对位置运动

恒速运动 MC_MoveVelocity

指定驱动轴，保持恒速运动控制。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_MoveVelocity	FB		MC_MoveVelocity(Axis：= （参数）， Execute：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Direction：= （参数）， BufferMode：= （参数）， InVelocity=> （参数）， Busy=> （参数）， Active=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Velocity	目标速度	LREAL	负数，0，正数	0	速度值
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值
Direction	方向	MC_DIRECTION	-1，0，1，2，3	0	运动方向：支持正方向、负方向和当前方向等三种方向值，见 4.1 节的 MC_Direction 说明
BufferMode	缓存模式	MC_BUFFER_MODE			
VAR_OUTPUT					
InVelocity	速度到	BOOL	TRUE	FALSE	设定速度到达之后则为 TRUE

	达		FALSE		
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Active		BOOL			该轴在工作
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

说明

- 这个指令由“SM3_Basic”库实现。
- Execute 的上升沿启动恒速运动，而后，只能由其他的命令来中止恒速运动。
- 当恒速运动到达设定速度后，InVelocity 信号将变为 TRUE。
- 当恒速运动被其它运动中断之后，InVelocity 信号将被重置为 FALSE。
- 负速度×负方向=正速度。
- 控制电机以指定的加速度加速到最大速度，然后以该最大速度一直运行，直到调用停止指令或者其他的中断指令中断该指令

例程：

图 5.8 以梯形图的形式，展示了“MC_MoveVelocity”功能块的两个实例“First”和“Second”连接的例子：

1）时间图的左部画出了“Second”在“First”之后调用的情况。当“First”达到设置速度 3000 时，第一个指令输出了 InVelocity 信号，与下一个指令进行“与”运算，触发了第二个指令以 2000 的速度运动。

2）时间图的右部画出了当“First”还在执行时“Second”被启动的情况。这时，“First”的运动被中断，并被在“First”速度恒定期间传递的 Test 信号终止。虽然第一个指令仍在向着 3000 的速度加速过程中，但由于中途执行了第二个实例，第一个的加速被中断并中止。然后第二个实例将会减速到 2000，之后第二个实例的 InVelocity 置 TRUE。

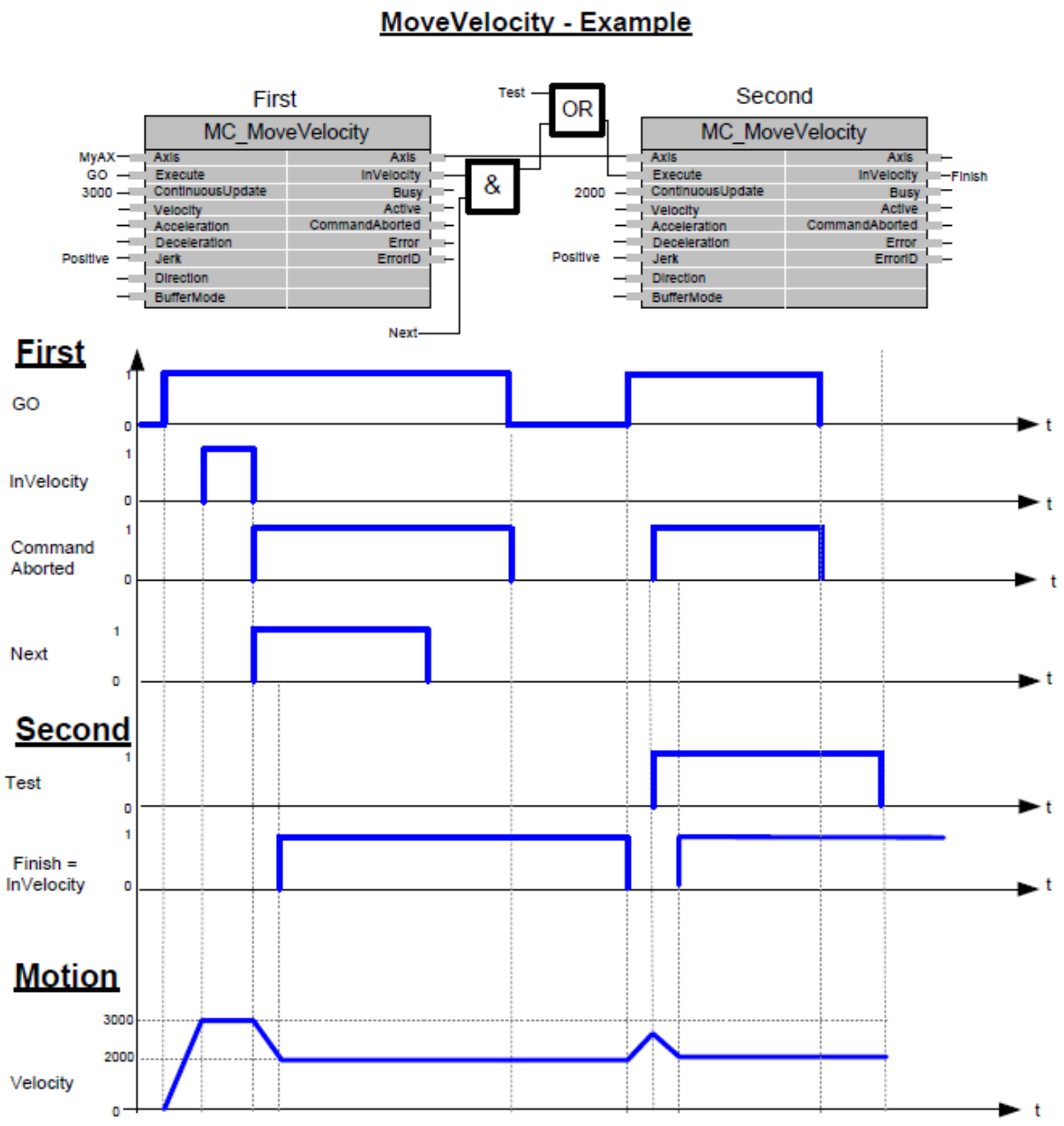
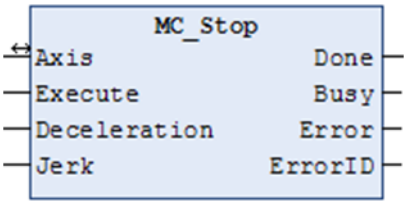

MoveVelocity - Example
First
MC_MoveVelocity
MyAX
GO
3000
Positive
Axis
Execute
ContinuousUpdate
Velocity
Acceleration
Deceleration
Jerk
Direction
BufferMode
Axis
InVelocity
Busy
Active
CommandAborted
Error
ErrorID
Test
OR
&
Next
Second
MC_MoveVelocity
2000
Positive
Finish
First
GO
InVelocity
Command Aborted
Next
Second
Test
Finish = InVelocity
Motion
3000
2000
Velocity
t

图 5.8　恒速运动

停止运动 MC_Stop

中断轴正在进行的运动，对轴进行减速停止。

指令外观：

指令	FB/ FUN	图形模块	结构文本
MC_Stop	FB		MC_Stop(Axis：= （参数）， Execute：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Deceleration	目标减速度	LREAL	负数，0，正数	0	停止的减速度值
Jerk	目标减减速度	LREAL	负数，0，正数	0	停止的减减速度值
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error

说明：

- 这个指令由“SM3_Basic”库实现。
- Execute 的上升沿处开始停止运动。
- 此指令运行时，不能被其它任何指令终止。
- 控制轴减速，由当前速度变为“0”。
- 因启动 MC_Stop(强制停止)指令而处于运行中的指令执行 CommandAborted(执行中断)

例程：

图 5.9 为 FB2（MC_Stop）实例与 FB1（MC_MoveVelocity）实例相结合时的使用方法。

- 由图中时序图可以看到，在 FB1 的运动过程中，启动了 FB2，轴的速度会呈现倾斜向下的趋势，直到为 0。
- 只要 FB2 的 Execute 信号为 TRUE，对应的轴就不会执行任何运动命令。
- 在 FB2 启动后，FB1 的 Error 信号置为 TRUE，只是当前轴处于停止状态。

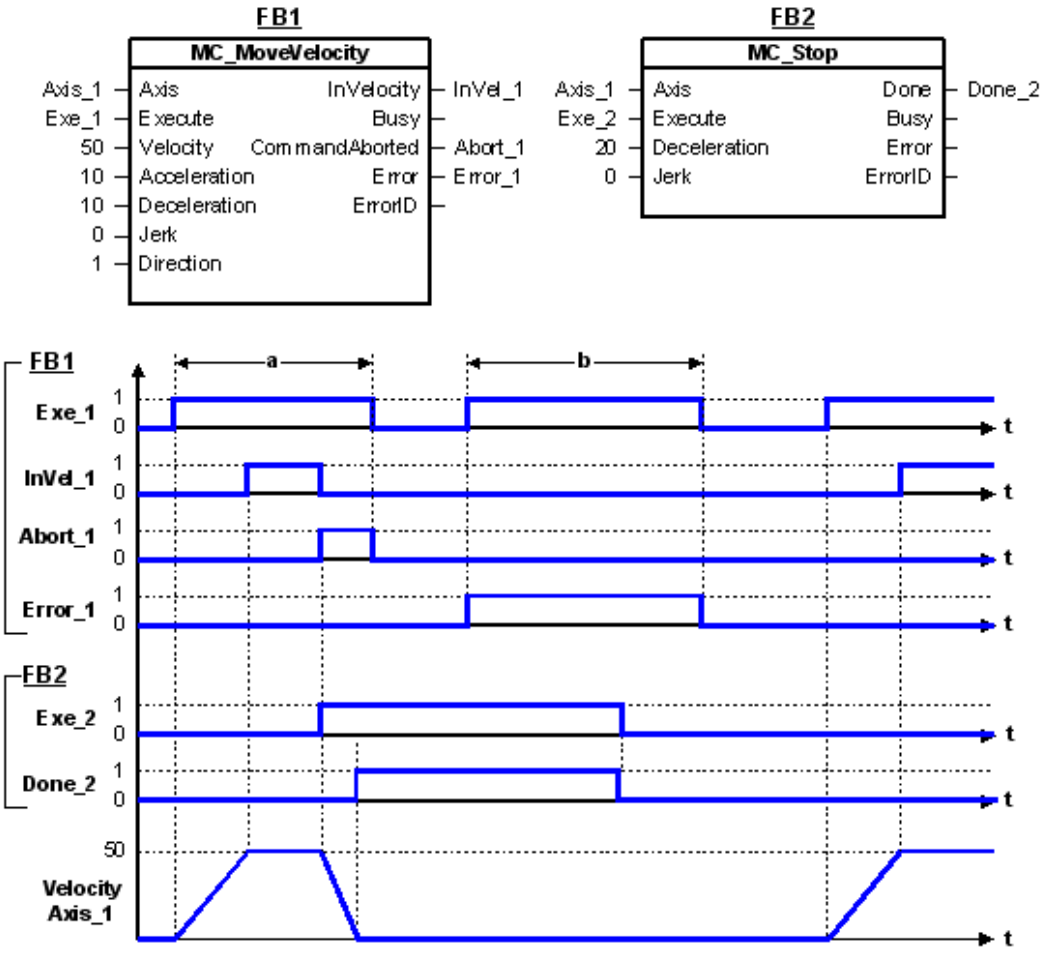
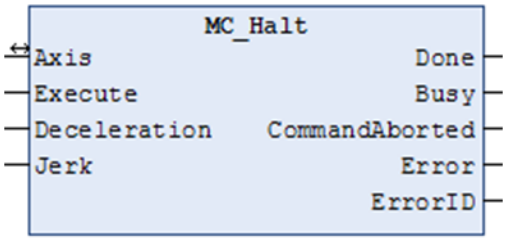


图 5.9　停止运动

暂停运动 MC_Halt

减速停止轴正在执行的运动，停止的运动可恢复执行未完成部分。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_Halt	FB		MC_Halt(Axis：= （参数）， Execute：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型		初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Deceleration	目标减速度	LREAL	负数，0，正数	0	停止的减速度值
Jerk	目标减减速度	LREAL	负数，0，正数	0	停止的减减速度值
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
bCommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	如果该命令已被其他命令终止则为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

功能说明：

- 这个指令由“SM3_Basic”库实现。
- 对于设定轴，在 Execute 的上升沿处开始暂停运动。
- 重新执行原来运动指令可以恢复原运动的执行。

例程：

- 图 5.10 显示了 MC_MoveVelocity 指令正在执行运动的过程中，被 MC_Halt 指令中断的行为。
- 与 MC_Stop 相比，MC_Halt 只是暂停运动，运动在后续还能够继续执行直到指令完成。
- 在被 MC_Halt 指令暂停后，轴可以在速度没有为 0 时，重新调用运动指令，进行再次加速。

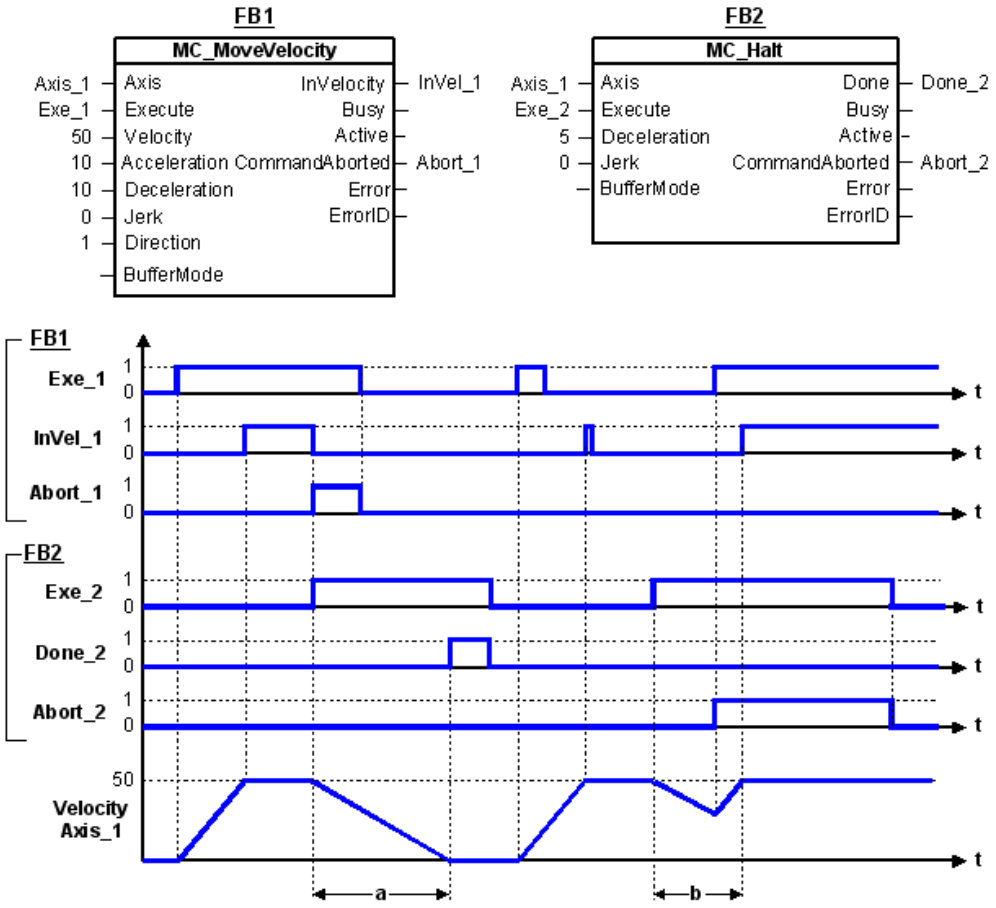


图 5.10　暂停运动

点动运动 MC_Jog

该指令用于手动控制轴朝指定方向运动。

指令外观：

指令	FB/ FUN	图形模块	结构文本
MC_Jog	FB	MC_Jog Axis　Busy JogForward　CommandAborted JogBackward　Error Velocity　ErrorId Acceleration Deceleration Jerk	MC_Jog(Axis：= （参数）， JogForward：= （参数）， JogBackward：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Busy=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorId=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
JogForward	向前点动	BOOL	TRUE FALSE	FALSE	如果 JogForward 为 TRUE，轴将按给定参数（Velocity、Acceleration、Deceleration）朝正向运动，如果 JogBackward 同时为 TRUE，轴不动。
JogBackward	向后点动	BOOL	TRUE FALSE	FALSE	如果 JogBackward 为 TRUE，轴将按给定参数（Velocity、Acceleration、Deceleration）朝负向运动，如果 JogForward 同时为 TRUE，轴不动。
Velocity	目标速度	LREAL	负数，0，正数	0	速度最大值 [u/s]
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值[u/s^2]
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值[u/s^2]
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值
VAR_OUTPUT					
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	如果该命令已被其他命令终止则为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

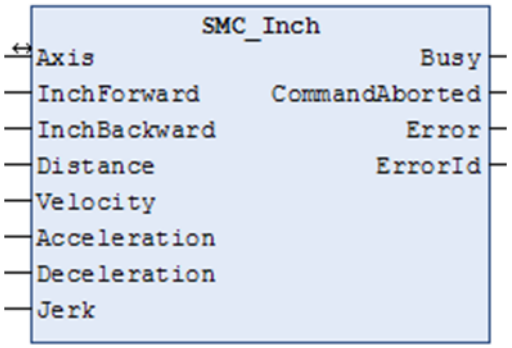
功能说明：

- 这个指令由“SM3_Basic”库实现。
- 当“JogForward”或“JogBackward”为 TRUE 时，指定轴分别往正方向或负方向执行恒速运动
- 同时将 JogForward 和 JogBackward 置为 TRUE，将不会有运动发生。
- 如果 MC_Jog 指令的指令速度设置值超过轴参数中的点动最高速度，则以点动最高速度执行。

寸动指令 SMC_Inch

该指令用于手动控制轴按指定的距离单位，一段一段的朝指定方向运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_Inch	FB		SMC_Inch(Axis：= （参数）， InchForward：= （参数）， InchBackward：= （参数）， Distance：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Busy=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorId=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
InchForward	向前寸动	BOOL	TRUE FALSE	FALSE	如果 InchForward 为 TRUE，轴朝正向运动设定的距离，如果 InchForward 再次设定 FALSE 到 TRUE，则轴再移动一段设定的距离。如果轴还没有运动到设定的距离，InchForward 就变为 FALSE，轴立即减速到 0，Busy 输出为 FALSE。如果 InchBackward 同时为真，轴不动。
InchBackward	向后寸动	BOOL	TRUE FALSE	FALSE	如果 InchBackward 为 TRUE，轴朝反向运动设定的距离，如果 InchBackward 再次设定 FALSE 到 TRUE，则轴再移动一段设定的距离。如果轴还没有运动到设定的距离，InchBackward 就变为 FALSE，轴立即减速到 0，Busy 输出为 FALSE。如果 InchForward 同时为真，轴不动。
Distance	寸动距离	LREAL	负数，0，正数	0	定义一次输入轴要运动的距离
Velocity	目标速度	LREAL	负数，0，正数	0	速度最大值 [u/s]
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值[u/s^2]
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值[u/s^2]
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值
VAR_OUTPUT					
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE

CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error

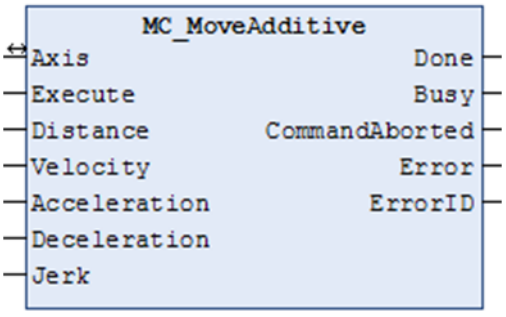
功能说明：

- 这个指令由“SM3_Basic”库实现。
- 一次运动的最大距离是固定的，由参数 Distance 定义。
- 如果需要再次运行 Distance 的距离，需要重置输入（InchForward OR InchBackward）。
- 如果距离 Distance 还没有到达，输入（InchForward OR InchBackward）就重置为 FALSE，运动立即减速停止。
- 输入 InchForward 和 InchBackward 都为 TRUE 时，轴不会运动。此时，当其中一个信号变成 FALSE，则轴将执行还为 TRUE 的信号所指定的运动。
- 每次正向或反向运动的速度、加减速度等参数由 Velocity、Acceleration、Deceleration 和 Jerk 决定。

位置叠加 MC_MoveAdditive

用于控制终端执行机构在运动过程中做位置叠加相运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_MoveAdditive	FB		MC_MoveAdditive(Axis：= （参数）， Execute：= （参数）， Distance：= （参数）， Velocity：ion：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Distance		LREAL		0	运动的相对距离
Velocity	目标速度	LREAL	负数，0，正数	0	最大速度
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

功能说明：

- 这个指令由“SM3_Basic”库实现。
- 此指令用于控制终端执行机构按给定的速度、加速度移动一段附加的距离。
- 终端执行机构的最终位置为前一个位移指令和此指令给定的距离总和。
- 打断其它指令的运动时，轴的速度为此功能块的速度。
- 当前一个指令为速度指令时，此指令执行时会终止速度指令执行，并按给定的速度，加减速移动给定的距离后停止。

例程：

图 5.11 展示了第一个实例“First”（MC_MoveAbsolute）和第二个实例“Second”（MC_MoveAdditive）组合的运动。

- 在图的前段，Second 实例在 First 实例之后被调用。当 First 实例运动到指定的位置 6000，则速度为 0 并且 Done 信号为 True，开始执行实例 Second，使轴移动到位置 10000。
- 在图的后段，在实例 First 执行的过程中，启动 Second 实例，First 实例的运动此刻正处在 4000 的位置，此时 First 的运动被打断，开始减速，当到达位置 6000 时，以 Second 指定的速度，将轴移动到最终的位置 10000。

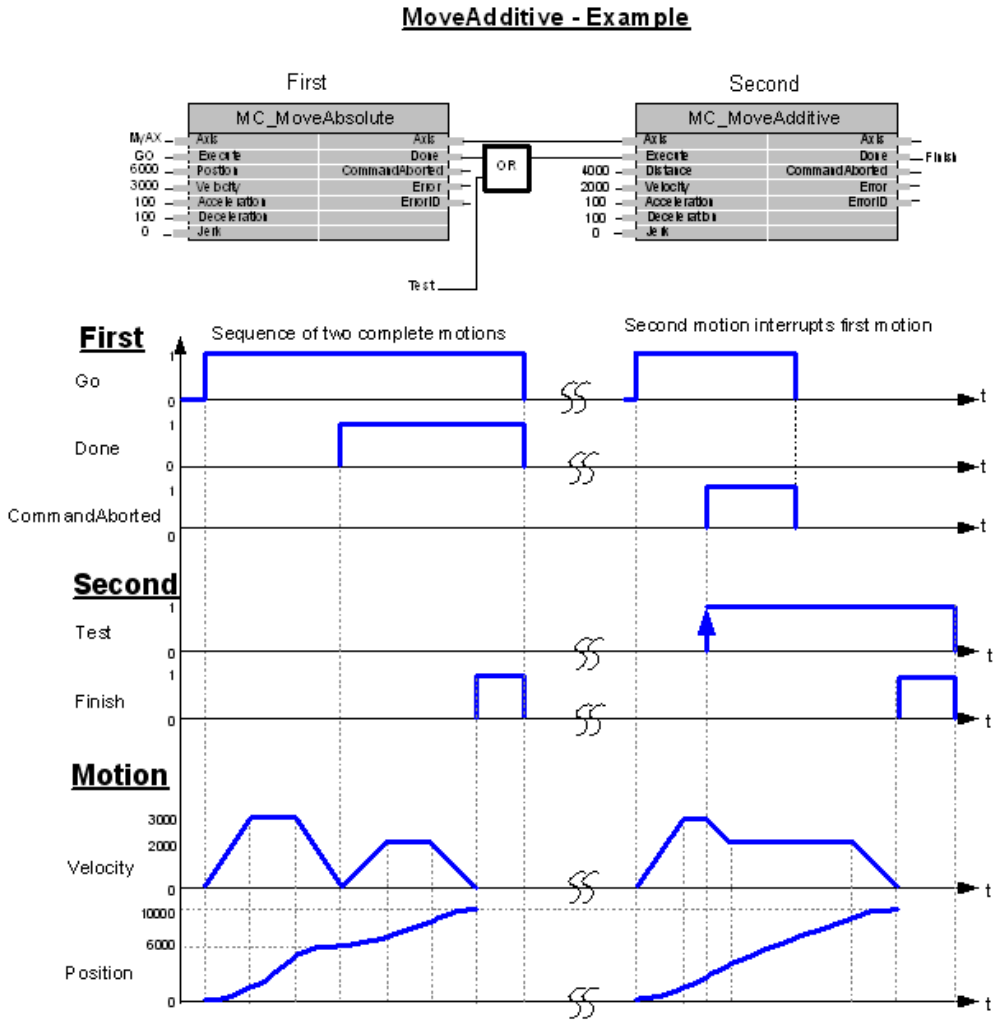


图 5.11　位置叠加运动

位置速度叠加　MC_MoveSuperImposed

用于控制终端执行机构在运动过程中叠加运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_MoveSuperImposed	FB	MC_MoveSuperImposed Axis　Done Execute　Busy Abort　CommandAborted Distance　Error VelocityDiff　ErrorID Acceleration Deceleration Jerk	MC_MoveSuperImposed(Axis: = （参数）， Execute: = （参数）， Abort: = （参数）， Distance: = （参数）， VelocityDiff: = （参数）， Acceleration: = （参数）， Deceleration: = （参数）， Jerk: = （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Abort	终止	BOOL			终止运动，输出参数复位
Distance		LREAL		0	运动的相对距离
VelocityDiff	叠加速度	LREAL	负数，0，正数	0	叠加的速度
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

功能说明：

- 这个指令由“SM3_Basic”库实现。
- 此指令用于控制终端执行机构按给定的速度、加速度移动一段叠加的距离。
- 终端执行机构的最终位置为前一个位移指令和此指令给定的距离总和。
- 第一段运动未完成时，速度为两段运动之和；第一段运动完成时，速度为此功能块的速度。
- 当前一个指令为速度指令时，此指令执行时会终止速度指令执行，并按给定的速度，加减速移动给定的距离后停止。

例程：

图 5.12 展示了 MC_MoveRelative 功能块的实例“First”和 MC_MoveSuperImposed 功能块的实例“Second”连接的例子：

可以看到 CommandAborted 信号的曲线一直都是零，这是因为“Second”实例所作用的，是和“First”命令在同一个实例上的。

最终运动结束的位置是在 7000-8000 之间，具体取决于“Second”实例的实际启停时间。

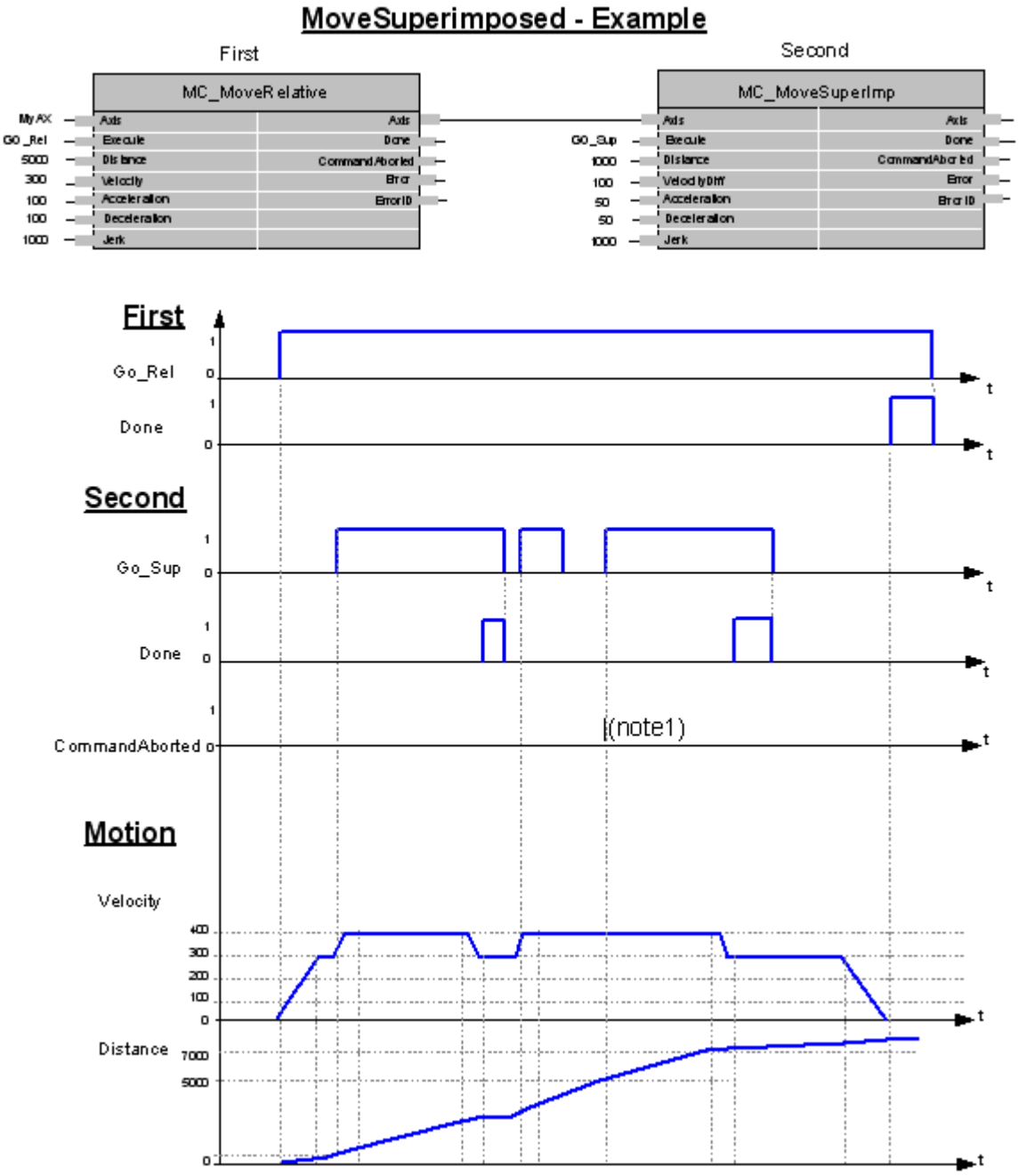


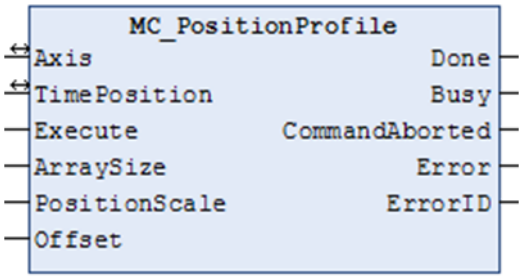
图 5.12　位置速度叠加运动

5.2.3　PVT 运动

时间-位置规划　MC_PositionProfile

用户可以自己规划“时间-位置”数据表，控制器将按照规划的数据完成运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_PositionProfile	FB		MC_PositionProfile(Axis：= （参数）， TimePosition：= （参数）， Execute：= （参数）， ArraySize：= （参数）， PositionScale：= （参数）， Offset：= （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
TimePosition	数据表	MC_TP_REF	—	—	用户规划的时间-位置数据表
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
ArraySize	数据点数	INT	0 正数	0	数据表大小
PositionScale	比例	LREAL	负数，0，正数	1	整个数据表位置的比例系数
Offset	偏移	LREAL	负数，0，正数	0	位置偏置
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

功能说明：

- 这个指令由“SM3_Basic”库实现。
- PVT 运动是指通过定义各个时刻点轴达到的位置、速度、加速度等参数来规划一个运动。控制器支持的 PVT 运动有三种，如表 5.4 所示。

表 5.4　PVT 运动指令

指令名	功能说明
MC_PositionProfile	通过定义时间-位置表，规划轴的运动
MC_VelocityProfile	通过定义时间-速度表，规划轴的运动
MC_AccelerationProfile	通过定义时间-加速度表，规划轴的运动

- MC_PositionProfile 运动所用到的数据结构定义如下：

（1）“时间-位置”数据表的结构体：

```
TYPE MC_TP_TABLE
    STRUCT
        Number_of_pairs ： INT；        //位置-时间对的个数
        IsAbsolute ： BOOL；            //位置值绝对相对选择
        MC_TP_Array ： ARRAY [1..N] of MC_TP；      //位置-时间数据
    END_STRUCT
END_TYPE
```

（2）“时间-位置”数据的结构体：

```
TYPE MC_TP
    STRUCT
        delta_time ： TIME；      //位置-时间数据之时间
        position ： REAL：        //位置-时间数据之位置（绝对值或相对值）
    END_STRUCT
END_TYPE
```

例程：

MC_TP_REF 支持一个特殊的数据类型。图 5.13 中的代码给出的是这个数据结构的一个示例。

一个时间/位置的内容可以被表述为 DeltaTime/Pos，这里的 DeltaTime 是指连续两点间的时间差：

```
//PT运动参数的赋值
    arPT.IsAbsolute:=FALSE ;    //相对位置
    arPT.MC_TP_Array:=arNUM ;   //PT点的数据
    arPT.Number_of_pairs:=4 ;   //PT点的个数

//PT点的数据赋值
    arNUM[1].delta_time:=T#1S;  //第一个PT点的时间数据
    arNUM[1].position:=10000;   //第一个PT点的位置数据
    arNUM[2].delta_time:=T#2S;
    arNUM[2].position:=40000;
    arNUM[3].delta_time:=T#1S;
    arNUM[3].position:=-10000;
    arNUM[4].delta_time:=T#0.5S;
    arNUM[4].position:=-5000;
```

图 5.13　程序 PT 的数据

本例实现了 AXIS_0 的 PT 运动，分 4 个阶段，各阶段之间运动连续：

第一阶段：1 秒，运动 10000

第二阶段：2 秒，运动 40000

第三阶段：1 秒，运动-10000

第四阶段：0.5 秒，运动-5000

例程“PT”的运行结果如图 5.14 所示。

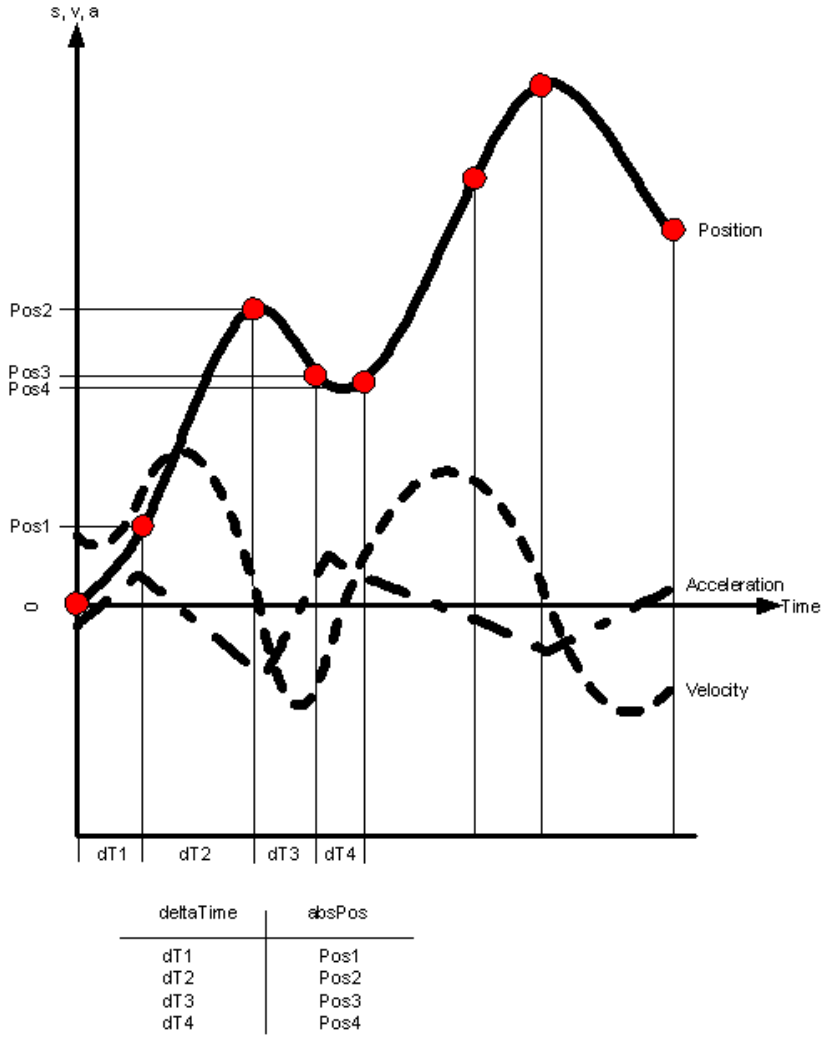
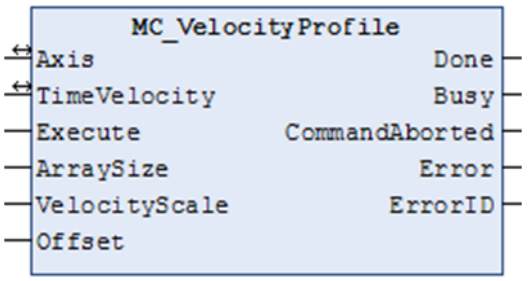


图 5.14　PT 运行结果

时间-速度规划　MC_VelocityProfile

与 MC_PositionProfile 指令相似，MC_VelocityProfile 通过定义“时间-速度”数据来规划运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_VelocityProfile	FB		MC_VelocityProfile(Axis：= （参数）， TimeVelocity：= （参数）， Execute：= （参数）， ArraySize：= （参数）， VelocityScale：= （参数）， Offset：= （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
TimeVelocity		MC_TV_REF			用户规划的时间-速度数据表
VAR_INPUT					
Execute	启动	BOOL	TRUE、FALSE	FALSE	上升沿触发指令运动。
ArraySize	数据点数	INT	0 正数	0	数据表大小
VelocityScale	比例	LREAL	负数，0，正数	1	比例系数
Offset	偏移	LREAL	负数，0，正数	0	速度偏置
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE、FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

功能说明：

- 这个指令由“SM3_Basic”库实现。
- MC_VelocityProfile 功能块为时间和速度的轮廓运动模型 ， 按照用户在“时间-速度”表变量中设定的数据执行运动。
- VT 运动所用到的数据表：

（1）“时间-速度”数据表的结构体如下：

TYPE MC_TV_TABLE

```
    STRUCT
        Number_of_pairs ： INT;        //速度-时间对的个数
        IsAbsolute ： BOOL;            //速度值绝对相对选择
        MC_TV_Array ： ARRAY [1..N] of MC_TV;      //速度-时间数据
    END_STRUCT
END_TYPE
```

（2）“时间-速度”数据的类型

```
TYPE MC_TV
    STRUCT
        delta_time ： TIME;    //速度-时间数据之时间
        velocity ： REAL;      //速度-时间数据之速度（绝对值或相对值）
    END_STRUCT
END_TYPE
```

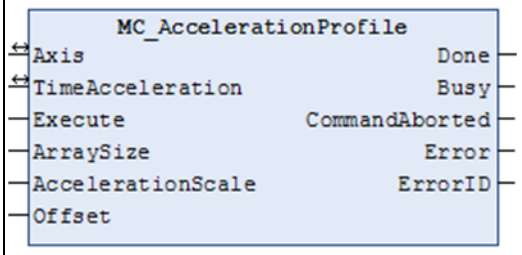
例程：

例程同 PT 例程，只需把 MC_TP_REF 改成 MC_TV_REF 即可，详见例程“VT”。

时间-加速度规划 MC_AccelerationProfile

与 MC_PositionProfile 指令相似，MC_AccelerationProfile 通过定义“时间-加速度”数据来规划运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_AccelerationProfile	FB		MC_AccelerationProfile(Axis：= （参数）， TimeAcceleration：= （参数）， Execute：= （参数）， ArraySize：= （参数）， AccelerationScale：= （参数）， Offset：= （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
TimeAcceleration		MC_TA_REF			用户规划的时间-加速度数据表
VAR_INPUT					

Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
ArraySize	数据点数	INT	0 正数	0	数据表大小
AccelerationScale	比例	LREAL	负数，0，正数	1	比例系数
Offset	偏移	LREAL	负数，0，正数	0	加速度偏置
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

功能说明：

- 这个指令由“SM3_Basic”库实现。
- MC_AccelerationProfile 功能块为时间和加速度的轮廓运动模型，按照用户在“时间-加速度”表变量中设定的数据执行运动。
- AT 运动所用到的数据表：

（1）“时间-加速度”数据表的结构体如下：

```
TYPE MC_TA_TABLE
    STRUCT
        Number_of_pairs ： INT;        //加速度-时间对的个数
        IsAbsolute ： BOOL;            //加速度值绝对相对选择
        MC_TA_Array ： ARRAY [1..N] of MC_TA;      //加速度-时间数据
    END_STRUCT
END_TYPE
```

（2）“时间-加速度”数据的类型

```
TYPE MC_TA
    STRUCT
        delta_time ： TIME;     //加速度-时间数据之时间
        acceleration： REAL;        //加速度-时间数据之加速度（绝对值或相对值）
    END_STRUCT
END_TYPE
```

例程：

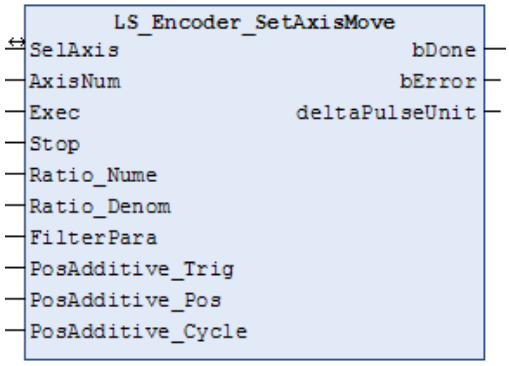
例程同 PT 例程，只需把 MC_TP_REF 改成 MC_TA_REF 即可，详见例程“AT”。

5.2.4 雷赛指令

编码器值转换轴运动 LS_Encoder_SetAxisMove

将编码器读到的数值转换成轴的运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_Encoder_SetAxisMove	FB		LS_Encoder_SetAxisMove(SelAxis:= （参数）， Exec:= （参数）， Stop:= （参数）， EncodeNum:= （参数）， Ratio_Nume:= （参数）， Ratio_Denom:= （参数）， FilterPara:= （参数）， PosAdditive_Trig:= （参数）， PosAdditive_Pos:= （参数）， PosAdditive_Cycle:= （参数）， bDone=> （参数）， bError=> （参数）， deltaPulseUnit=> （参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始化	注释
SelAxis	轴	AXIS_REF_VIRTUAL_SM3	-	-	配置需要运动的轴
Exec	启动	BOOL	TRUE FALSE	FALSE	启动，上升沿有效
Stop	停止	BOOL	TRUE FALSE	FALSE	停止，上升沿有效
EncodeNum	编码器号	SINT	0-2	0	编码器号，范围 0-2
Ratio_Nume	分子	INT	1-10000	1	配置放大倍率，放大倍率分子，范围：1-10000
Ratio_Denom	分母	INT	1-10000	1	配置放大倍率，放大倍率分母，范围：1-10000
FilterPara	平滑系数	UINT	1-100	2	平滑系数，参数越大，数据平滑增强，范围：1-100
PosAdditive_Trig	触发信号	BOOL	TRUE FALSE	FALSE	位置叠加触发信号，上升沿有效。
PosAdditive_Pos	叠加位置	LREAL	负数，0，正数	0	叠加的位置，叠加的位置不参与倍率运算
PosAdditive_Cycle	周期	UDINT	正数	1	叠加的位置在几个周期内完成
VAR_OUTPUT					
dDone	完成	BOOL	TRUE FALSE	FALSE	运动完成状态
bError	报错	BOOL	TRUE	FALSE	FALSE-没有错误； True-输入参

			FALSE		数越界
deltaPulseUnit	脉冲单位	LREAL	0，正数	0	每周期输出的脉冲单位

功能说明：

- 这个指令由“PMC_Controller”库实现。
- 指令的功能，是将编码器值的变化，转换成从动轴的运动，类似手轮运动。
- Ratio_Nume 是倍率的分子，用来计算倍率。
- Ratio_Denom 是倍率的分母，用来计算倍率。
- PosAdditive_Pos 是从动轴目标位置的附加值，用来调整误差。
- 从动轴的目标位置=主动轴编码器值×Ratio_Nume/Ratio_Denom+PosAdditive_Pos
- 注意：运动轴编码器的相位顺序影响主动轴和从动轴的运动方向相反或相同。

例程：

本例用轴 1 的编码器值控制轴 0 运动，放大比例为 1：1。具体的例程，请参考“MoveENC”。运行结果如图 5.15 所示。

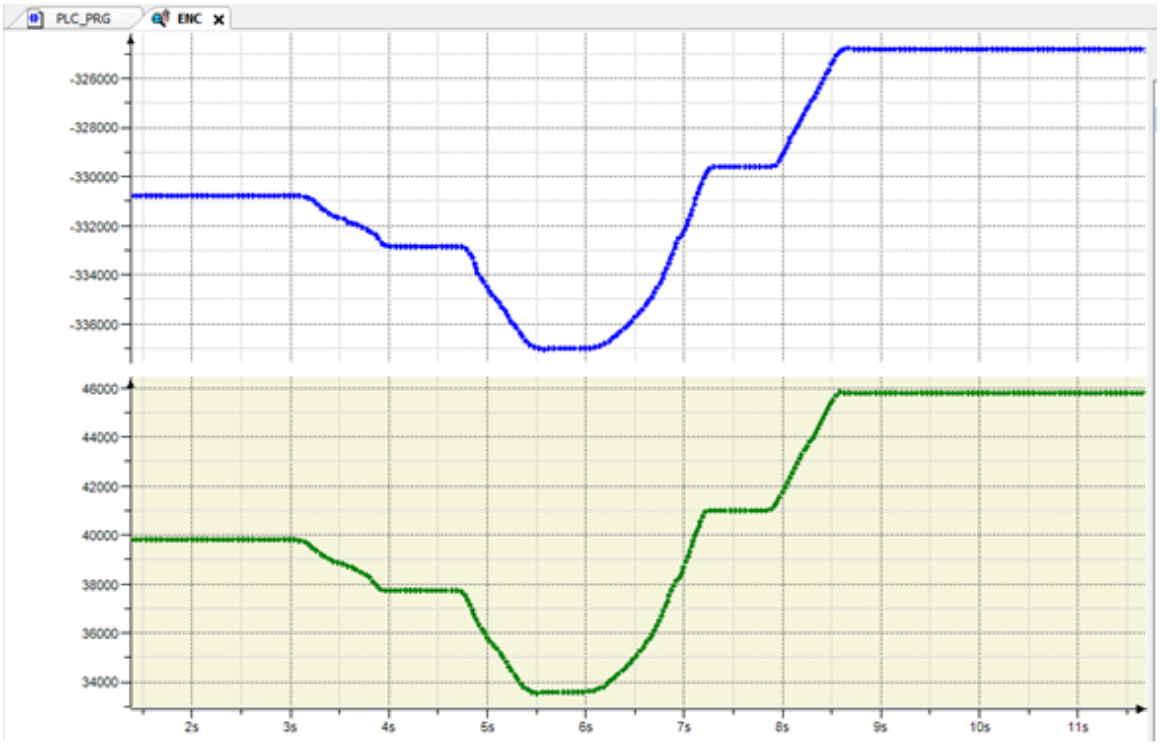


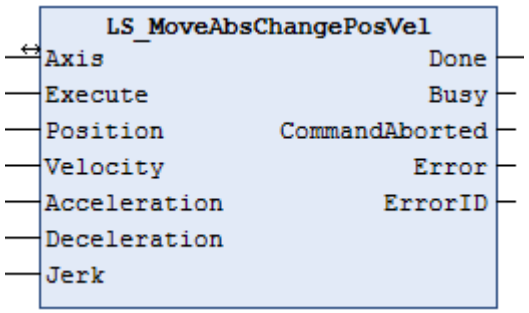
图 5.15　编码器值转换轴运动

如图 5.15 所示，蓝色线为从动轴 AXIS_0 的规划位置，绿色线为主动轴 AXIS_1 的编码器位置。

在线变速变位置 LS_MoveAbsChangePosVel

单轴绝对运动，可以在线变速、变位置（绝对位置）。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_MoveAbsChangePosVel	FB		LS_MoveAbsChangePosVel(Axis：= （参数）， Execute：= （参数）， Position：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Position		LREAL		0	绝对目标位置
Velocity	目标速度	LREAL	负数，0，正数	0	最大速度
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

功能说明：

- 这个指令由“PMC_SingleAxisLib”库实现。
- 当 Execute 为上升沿时，启动模块执行，且该信号需要保持。在 Execute 为 True 状态，改变 Position 或 Velocity，数据立即生效，按照新设置的值执行；如果 Execute 为 False，改变的值不起作用。
- 如果运动完成后，即 Done 为 True，再改变 Position 或 Velocity，新的运动不会执行。
- 轴运动的停止，需要单独调用 MC_Stop 功能块。

例程：

本例可实现 AXIS_0 先启动目标 50000，速度 10000，加速度 100000 的绝对运动，2 秒后在线变速变位置为目标 100000，速度 20000，加速度 200000 的绝对运动。运行结果如图 5.16 所示。

具体例程见“MoveAbsChangePosVel”

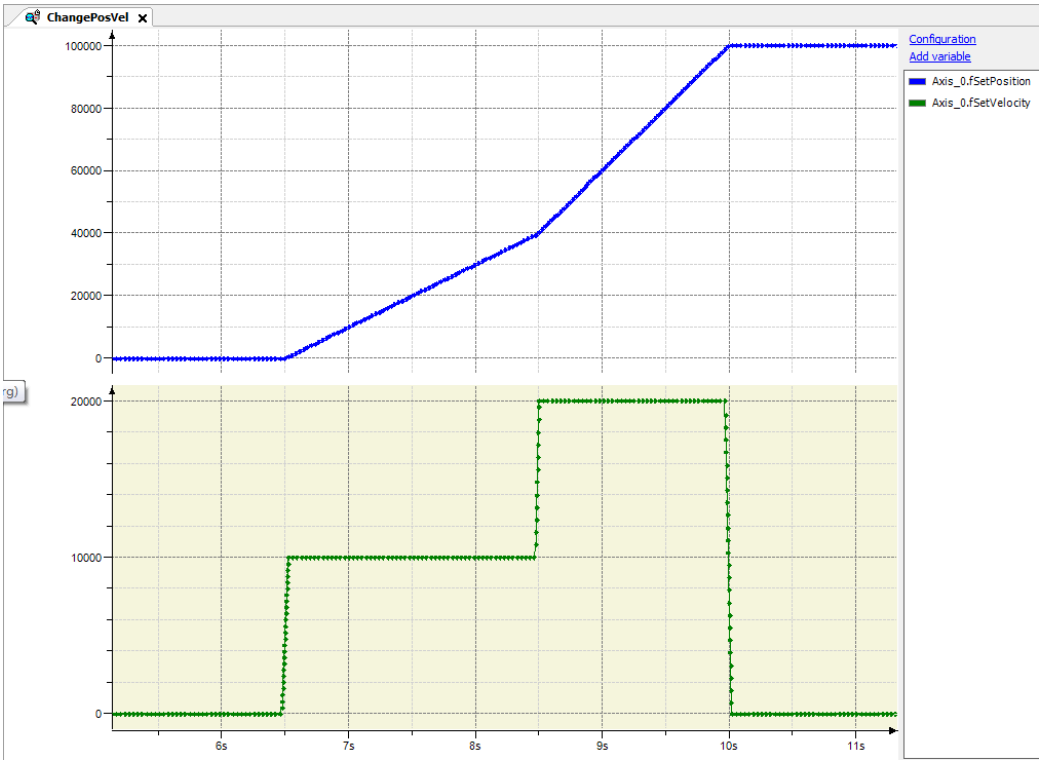
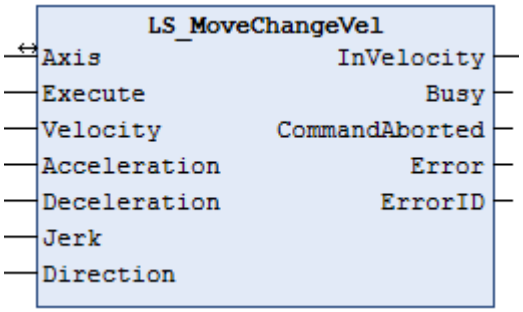


图 5.16　在线变速变位置运动

在线变速 LS_MoveChangeVel

单轴恒速运动，可以在线变速。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_MoveChangeVel	FB		LS_MoveChangeVel(Axis:= （参数）， Execute:= （参数）， Velocity:= （参数）， Acceleration:= （参数）， Deceleration:= （参数）， Jerk:= （参数）， Direction:= （参数）， InVelocity=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Velocity	目标速度	LREAL	负数，0，正数	0	修改的速度
Acceleration	目标加速度	LREAL	负数，0，正数	0	加速度值
Deceleration	目标减速度	LREAL	负数，0，正数	0	减速度值
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度值
Direction	方向	INT	-1，1	1	运动方向。 1：正方向；-1：负方向
VAR_OUTPUT					
InVelocity	到达设定速度	BOOL	TRUE FALSE	FALSE	设定速度到达之后则为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

功能说明：

- 这个指令由“PMC_SingleAxisLib”库实现。
- 当 Execute 为上升沿时，启动模块执行，该信号需要保持。
- 该指令的主要功能，是在运动过程中，修改运动方向、速度、加速度、减速度、Jerk，立即生效。
- 轴运动的停止，需要单独调用 MC_Stop 功能块。

例程：

本例可实现 AXIS_0 先启动速度 10000，加速度 100000 的正向恒速运动，2 秒后再启动在线变速为 20000 的负向恒速运动。运行结果如图 5.17 所示。

具体例程见“MoveChangeVel”。

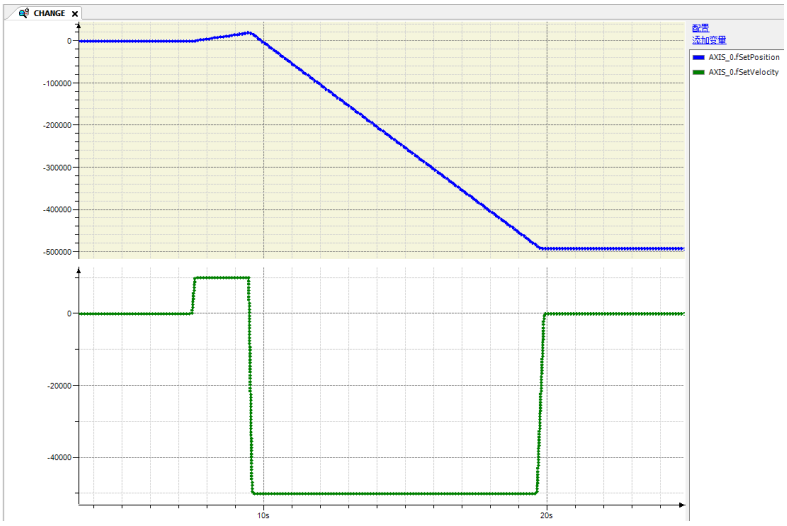
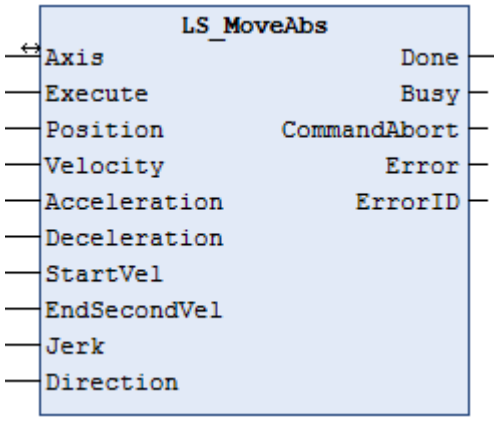


图 5.17　在线变速运动

单轴起跳速度绝对运动 LS_MoveAbs

控制轴按照设定参数运动到指定的绝对位置，只支持 T 型加减速。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_MoveAbs	FB		LS_MoveAbs(Axis：= （参数）， Execute：= （参数）， Position：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， StartVel：= （参数）， EndSecondVel：= （参数）， Jerk：= （参数）， Direction：= （参数）， Done=> （参数）， Busy=> （参数）， CommandAbort=> （参数）， Error=> （参数）， ErrorID=> （参数）)；

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Position	位置	LREAL	负数，0，正数	0	运动的目标位置 [u]（负或正）。
Velocity	目标速度	LREAL	0 正数	0	速度最大值（总是为正）[u/s]
Acceleration	目标加速度	LREAL	0 正数	0	加速度值（总是为正）[u/s2]
Deceleration	目标减速度	LREAL	0 正数	0	减速度值（总是为正）[u/s2]
StartVel	起跳速度	LREAL	>0.001	20	起跳速度，必须为正值，最小值为 0.001
EndSecondVel	停止速度	LREAL	0 正数	20	停止前缓冲速度，最小值为 0.001
Jerk	加加速度	LREAL	0 正数	0	加加速度值（总是为正）[u/s3]
Direction	方向	MC_DIRECTION	-1，1	shortest	1：正；-1：负
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	如果达到最终位置为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	如果该命令已被其他命令终止则为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

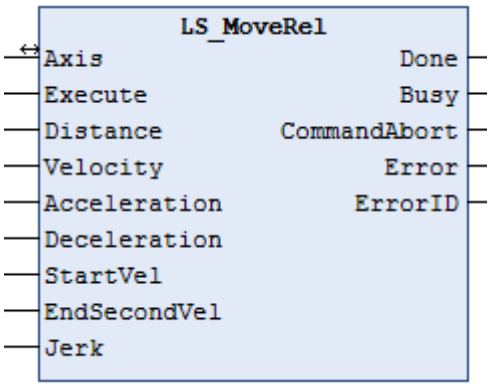
说明：

- 这个指令由“PMC_SingleAxisLib”库实现。
- 指定绝对坐标的目标位置，进行定位。
- 如果是线性轴的绝对点位运动，方向值将被忽略。
- 起跳速度和停止缓冲速度不能为负。
- 如果距离过短，可能达不到最大速度
- 将加减速度设定为 0 后，将不做加减速而直接达到目标速度。

单轴起跳速度相对运动 LS_MoveRel

控制轴按照设定参数运动一段相对距离，只支持梯形加减速。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_MoveRel	FB		LS_MoveRel(Axis:= （参数）, Execute:= （参数）, Distance:= （参数）, Velocity:= （参数）, Acceleration:= （参数）, Deceleration:= （参数）, StartVel:= （参数）, EndSecondVel:= （参数）, Jerk:= （参数）, Done=> （参数）, Busy=> （参数）, CommandAbort=> （参数）, Error=> （参数）, ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Distance	距离	LREAL	负数，0，正数	0	目标位置
Velocity	速度	LREAL	负数，0，正数	0	目标速度
StartVel	起跳速度	LREAL	>0.001	20	起跳速度，必须为正值，最小值为 0.001
EndSecondVel	停止速度	LREAL	>0.001	20	停止前缓冲速度，最小值为 0.001
Acceleration	加速度	LREAL	0	0	启动的加速度

			正数		
Deceleration	减速度	LREAL	0 正数	0	停止的减速度值
Jerk	加加速度	LREAL	0 正数	0	加加速度
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	运动异常终止
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

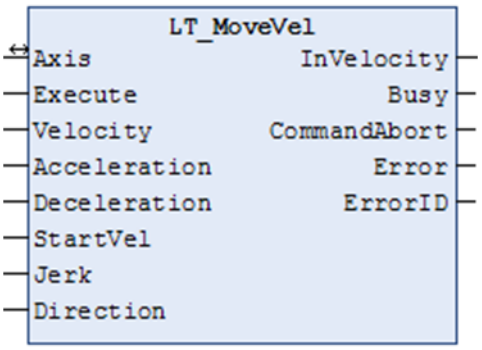
说明：

- 这个指令由“PMC_SingleAxisLib”库实现。
- 此功能块不支持在线变速变位置的功能。

单轴起跳速度恒速运动 LS_MoveVel

控制电机以指定的起跳速度和加速度加速到最大速度，然后以该最大速度一直运行。直到调用停止指令或者其他的中断指令中断该指令，只支持梯形加减速。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_MoveVel	FB		LS_MoveVel(Axis:= （参数）， Execute:= （参数）， Velocity:= （参数）， Acceleration:= （参数）， Deceleration:= （参数）， StartVel:= （参数）， Jerk:= （参数）， Direction:= （参数）， InVelocity=> （参数）， Busy=> （参数）， CommandAbort=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
Velocity	目标速度	LREAL	0 正数	0	速度值
Acceleration	目标加速度	LREAL	0 正数	0	加速度值
Deceleration	目标减速度	LREAL	0 正数	0	减速度值
StartVel	起跳速度	LREAL	>0.001	20	起跳速度，必须为正值，最小值为 0.001
Jerk	加加速度	LREAL	0 正数	0	加加速度值
Direction	方向	MC_DIRECTION	-1，1	0	运动方向：支持正方向、负方向和当前方向等三种方向值，其它方向不支持
VAR_OUTPUT					
InVelocity	速度到达	BOOL	TRUE FALSE	FALSE	设定速度到达之后则为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	当功能块执行还没结束时为 TRUE。
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	当功能块执行被中断时为 TRUE。
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误指示，见 SMC_Error。

说明：

- 这个指令由“PMC_SingleAxisLib”库实现。
- 负速度×负方向=正速度。
- 起跳速度必须为正。

例程：

本例可实现轴 0 起跳绝对运动，轴 1 起跳相对运动，轴 2 起跳恒速运动。运动参数中位置均为 50000，速度均为 10000，加减速均为 100000，起跳速度 1000。当轴 0 和轴 1 完成时，停止所有轴。运行结果如图 5.18 所示。具体代码见例程“MoveStartVel”。

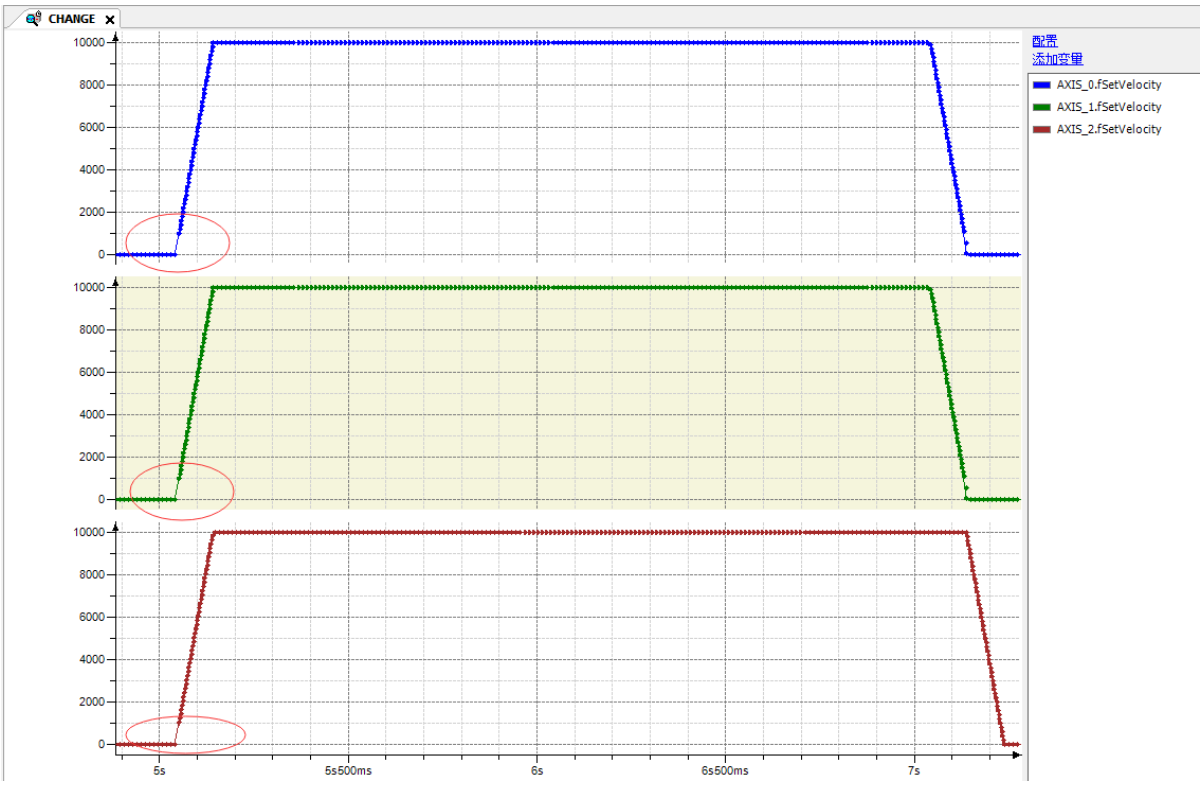


图 5.18　单轴起跳速度恒速运动

定点定速 LS_PositionToVelocity

单轴运动，实现定点定速功能。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_PositionToVelocity	FB	LS_PositionToVelocity Axis → Vel Exec → Done Halt → Busy Stop → CommandAborted IpoCycle → CommandHalt MoveMode → Error EndPos → ErrorID VelocityMode Velocity Acceleration Deceleration Jerk	LS_PositionToVelocity(Axis:= , Exec:= , Halt:= , Stop:= , IpoCycle:= , MoveMode:= , EndPos:= , VelocityMode:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Vel=> , Done=> , Busy=> , CommandAborted=> , CommandHalt=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Axis	轴	AXIS_REF_VIRTUAL_SM3	-	-	轴号
VAR_INPUT					
Exec	启动	BOOL	TRUE FALSE	FALSE	启动，上升沿有效
Halt	暂停	BOOL	TRUE FALSE	FALSE	暂停，上升沿有效
Stop	停止	BOOL	TRUE FALSE	FALSE	停止，上升沿有效
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us（与调用该功能块的任务周期一致）
MoveMode	移动模式	BOOL	TRUE FALSE	FALSE	运动模式选择，False 表示绝对模式，True 表示相对模式
EndPos	目标位置	ARRAY [0..1] OF LREAL	负数，0，正数	0	轴的目标位置，EndPos[0]表示第一段的位置，EndPos[1]表示第二段的位置
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	ARRAY [0..1] OF LREAL	负数，0，正数	[10,10]	插补速度 Pulse/s，Velocity[0]表示第一段的运行速度，Velocity[1]表示第二段的运行速度
Acceleration	加速度	ARRAY [0..1] OF LREAL	0 正数	[10,10]	插补加速度 Pulse/s2，Acceleration[0]表示第一段的加速度，Acceleration[1]表示第二段的加速度
Deceleration	减速度	ARRAY [0..1] OF LREAL	0 正数	[10,10]	插补加速度 Pulse/s2，Deceleration[0]表示第一段的减速度，Deceleration[1]表示第二段的减速度
Jerk	加加速度	ARRAY [0..1] OF LREAL	0 正数	[90000000, 90000000]	加加速度 Pulse/s3，速度模式设置为 3 时，需要设置该参数，该参数值不能为 0。Jerk[0]表示第一段的加加速度，Jerk[1]表示第二段的加加速度
VAR_OUTPUT					
Vel	当前速度	LREAL	负数，0，正数	0	当前运动速度
Done	执行完成	BOOL	TRUE FALSE	FALSE	运动结束状态
Busy	执行中	BOOL	TRUE FALSE	FALSE	运动进行中
CommandAborted	指令中断	BOOL	TRUE FALSE	FALSE	运动终止状态
CommandHalt	指令暂停	BOOL	TRUE FALSE	FALSE	运动暂停状态

Error	错误	BOOL	TRUE FALSE	FALSE	错误状态
ErrorID	错误码	SMC_ERROR	-	0	错误码

功能说明:

- 这个指令由“PMC_Ipolib”库实现。
- 定点定速运动属于单轴运动，实现指定轴定点定速功能：分两段运动到达目标位置，需要指定第一段的目标位置、最大速度和第二段的目标位置、最大速度。两段运动之间速度连续，不会降为零。

例程:

实现轴 0 分两段运动到达指定位置 50000Pulse，第一段运动速度为 5000Pulse/s，加减速度 10000Pulse/s^2，目标位置为 10000Pulse；第二段运动速度为 20000Pulse/s，加减速度 20000Pulse/s^2，目标位置为 50000Pulse。

新建工程后，要添加插补库 PMC_IpoLib 和插补辅助库 PMC_BasicModule；

运动模块 ACT_PositionToVelocity 以及主程序，如图 5.19 和 5.20 所示。

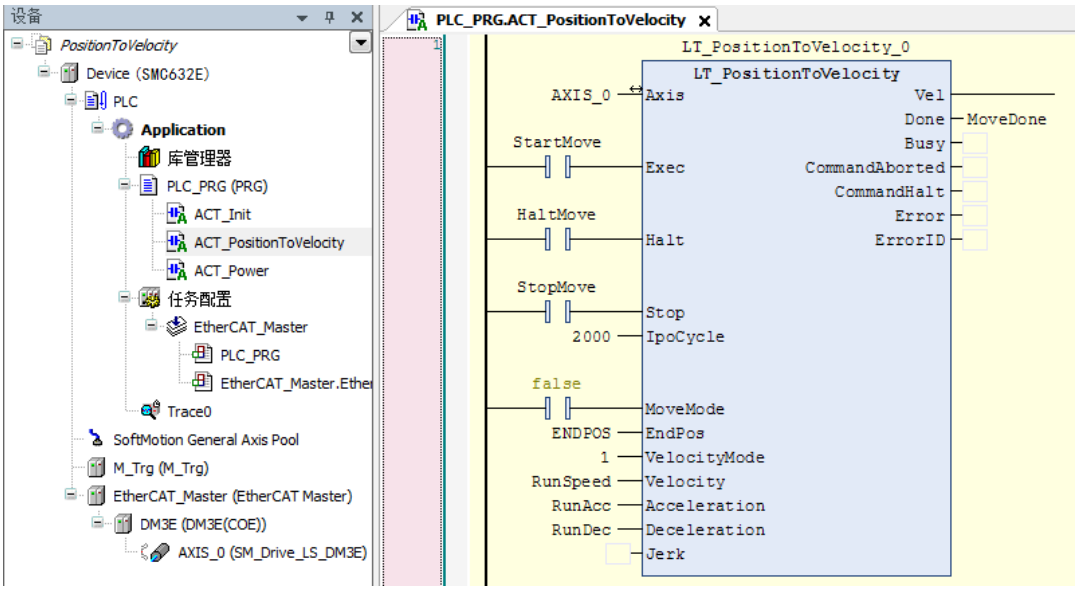


图 5.19　运动模块

图 5.21 为运行结果：轴 0 第一段运动以 5000Pulse/s 的速度运动到 10000Pulse 后，加速到 20000Pulse/s 并运动到 50000Pulse 的位置；中间段到达 10000Pulse 位置时速度并没有降为 0 再加速到 20000Pulse/s，而是从 10000Pulse/s 加速到 20000Pulse/s。

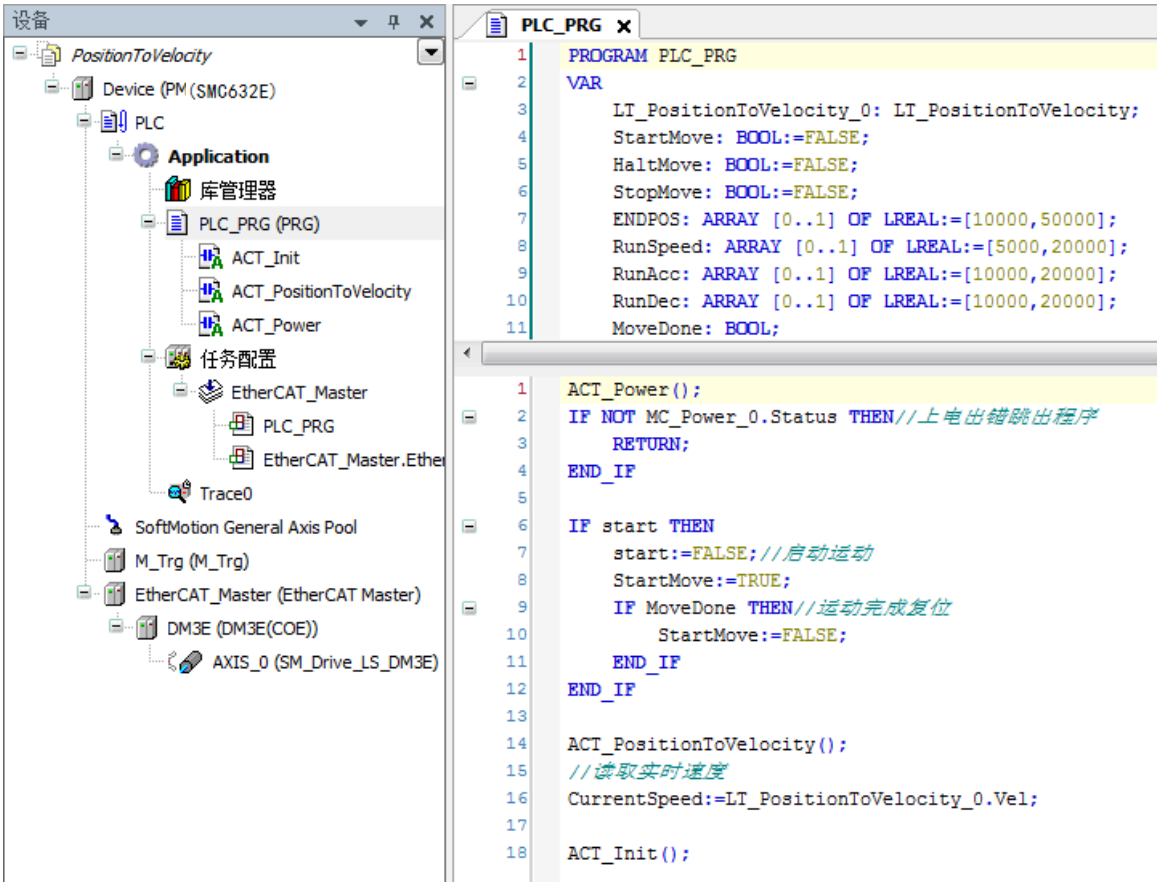


图 5.20　主程序

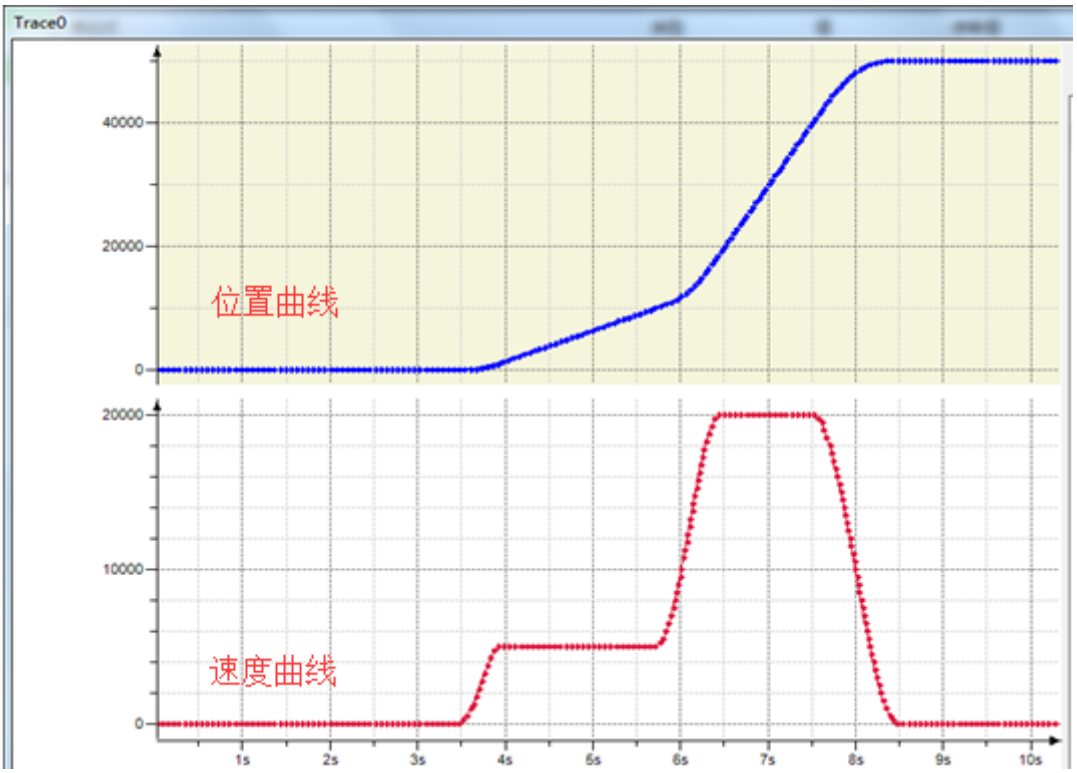


图 5.21　LS_PositionToVelocity 指令的速度位置曲线

第 6 章　同步指令

PMC600 提供灵活的、高性能的同步运动功能，同步功能包含在 SM3_Basic 和 PMC_Ipolib 这两个库之中。

其中，调用 SM3_Basic 库，用户可以创建自定义电子齿轮、电子凸轮运动用来同步多个轴的运动，实现复杂的同步需求。

调用 PMC_Ipolib 库，用户可以使用绝对位置跟随和相对位置跟随功能。

在实际应用中，同步切割过程、动态传输过程、灵活的长度分割、包装切割和口罩机等应用都可使用这类功能。

表 6.1　同步运动功能类型

功能类型	指令名	功能说明
电子齿轮	MC_GearIn	设置主从轴齿轮比并启动电子齿轮
	MC_GearInPos	设置主从轴同步距离和齿轮比并启动电子齿轮
	MC_GearOut	断开主从轴电子齿轮
电子凸轮	MC_CamIn	配置主从轴电子凸轮参数并启动
	MC_CamTableSelect	将选择的 cam 表连接到实际的凸轮表上
	MC_CamOut	断开主从轴电子凸轮
	SMC_GetTappetValue	读取当前的挺杆状态
跟随运动	LS_LineFollowAbs	一轴点位，另一轴跟随绝对位置运动
	LS_LineFollowRel	一轴点位，另一轴跟随相对位置运动

6.1　电子齿轮指令

电子齿轮用来建立主轴和从轴的特定比率（齿轮比）的线性关系，该功能一般用于简单的传送带，例如产品必须从一个传送带传到另一个，主从传送带必须定时定点交接产品，主从传送带的速度必须同步。

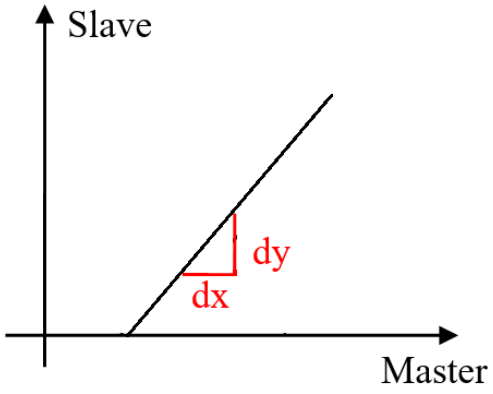


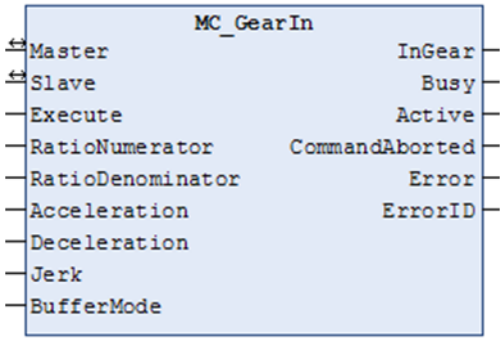
图 6.1　电子齿轮示意图

如图 6.1 所示，水平轴显示连接的主轴位置，垂直轴显示连接的从轴位置。当主轴信号有规律的运动（匀速运动），从轴的速度按照指定也是匀速的（固定的位置改变）。线性曲线的斜率决定电子齿轮比率。

齿轮耦合 MC_GearIn

设置主从轴齿轮比并启动电子齿轮。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GearIn	FB		MC_GearIn(Master：= （参数）， Slave：= （参数）， Execute：= （参数）， RatioNumerator：= （参数）， RatioDenominator：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， BufferMode：= （参数）， InGear=> （参数）， Busy=> （参数）， Active=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

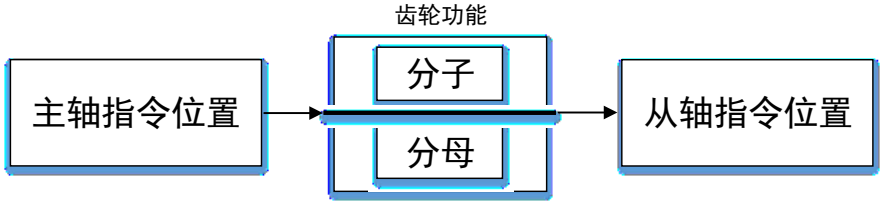
VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Master	主轴	AXIS_REF	-	-	映射到主轴，参阅“AXIS_REF_SM3”
Slave	从轴	AXIS_REF	-	-	映射到主轴，参阅“AXIS_REF_SM3”
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	此输入的一个上升沿将启动功能块的处理
RatioNumerator	分子	INT	正数	1	齿轮比分子
RatioDenominator	分母	UINT	正数	1	齿轮比分母
Acceleration	目标加速度	LREAL	负数，0，正数	0	电子齿轮比加速度
Deceleration	目标减速度	LREAL	负数，0，正数	0	电子齿轮比减速度
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度
BufferMode	缓存模式	MC_BUFFER_MODE			
VAR_OUTPUT					
InGear	齿轮啮合	BOOL	TRUE FALSE	FALSE	TRUE 表示齿轮比的处理已经完成

Busy	执行中	BOOL	TRUE FALSE	FALSE	TRUE 表示功能块的处理没有完成
Active		BOOL			该轴在工作
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	TRUE 表示命令被另一个命令中断
Error	错误	BOOL	TRUE FALSE	FALSE	功能块内部发生错误信号
ErrorID	错误码	SMC_ERROR	-	0	错误 ID，参阅“SMC_ERROR”

功能说明：

- 这个指令由“SM3_Basic”库实现。
- 执行电子齿轮后，从轴将处于同步状态，要解除耦合，必须调用 GearOut 指令。
- 指令属于速度同步类型，当启动指令后，从轴速度将上升至给定的齿轮比。在完成这个过程时，从轴耦合完成。在耦合期间造成的同步距离丢失不会得到补偿。
- 在 MC_GearIn 指令处于运行状态时，可以通过重新触发 MC_GearIn 指令，去改变主从轴的电子齿轮比，且中间不需要先调用 MC_GearOut 指令去暂停原来的电子齿轮运动。
- 到达目标速度，InGear 信号为 TRUE，此后，

  从轴移动量 = 主轴移动量×RatioNumerator / RatioDenominator



- 该指令一般适用于主轴速度稳定的情况，如果主轴速度实时变化的情况下，请注意慎重使用该指令。
- 注意：执行指令过程中请不要使用 MC_SetPosition 指令以免引起电机急速运转产生事故。

例程：

如果比率为 1：2，如图 6.2 所示，则从轴的移动距离正好是主轴距离的一半。如果使用速度控制模式，则使用速度值执行耦合；在每个循环中，从属对象将以主速度的一半移动。

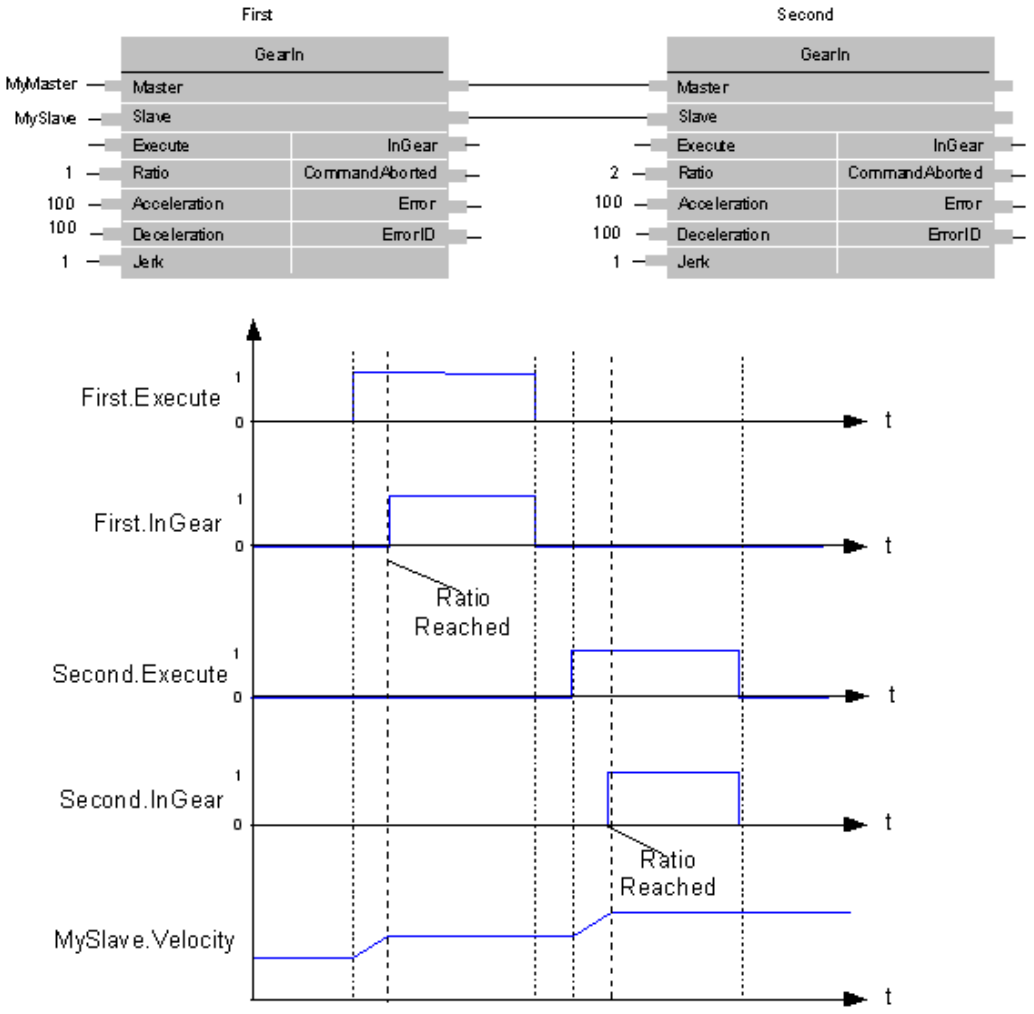
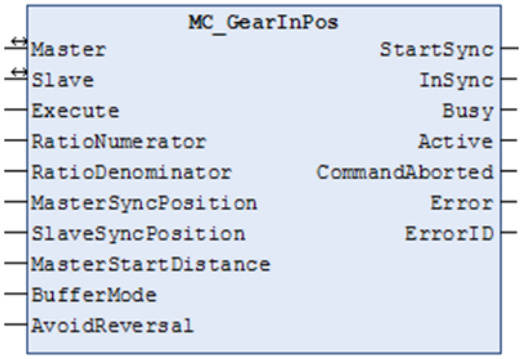


图 6.2　电子齿轮

齿轮平滑耦合　MC_GearInPos

设定主从轴电子齿轮比，执行电子齿轮运动。与 MC_GearIn 指令不同的，是该指令需要指定开始同步的主轴位置、从轴位置、主轴开始同步距离，并以此来完成切入电子齿轮动作。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GearInPos	FB		MC_GearInPos(Master：= （参数）， Slave：= （参数）， Execute：= （参数）， RatioNumerator：= （参数）， RatioDenominator：= （参数）， MasterSyncPosition：= （参数）， SlaveSyncPosition：= （参数）， MasterStartDistance：= （参数）， BufferMode：= （参数）， AvoidReversal：= （参数）， StartSync=> （参数）， InSync=> （参数）， Busy=> （参数）， Active=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Master	主轴	AXIS_REF	-	-	映射到主轴，参阅“AXIS_REF_SM3”
Slave	从轴	AXIS_REF	-	-	映射到主轴，参阅“AXIS_REF_SM3”
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	此输入的一个上升沿将启动功能块的处理
RatioNumerator	分子	INT	正数	1	齿轮比分子
RatioDenominator	分母	UINT	正数	1	齿轮比分母
MasterSyncPosition	主轴同步位置	REAL	负数，0，正数	0	主从轴达到同步时主轴的位置
SlaveSyncPosition	从轴同步位置	REAL	负数，0，正数	0	主从轴达到同步时从轴的位置
MasterStartDistance	啮合段主轴运动距离	REAL	正数	0	从轴启动到和主轴同步时主轴运动的距离
BufferMode	缓存模式	MC_BUFFER_MODE			
AvoidReversal	禁止反转	BOOL	TRUE FALSE	FALSE	设置为 FALSE 表示从轴物理位置超前的情况下进行反转；设置为 TRUE 表示从轴物理上不能实现反转或者导致危险，只在模态轴下适用。如果反转不能被避免，那么轴将错误停止。
VAR_OUTPUT					
StartSync	开始同	BOOL	TRUE	FALSE	TRUE 表示电子齿轮开始进行

	步		FALSE		处理
InSync	到达同步	BOOL	TRUE FALSE	FALSE	TRUE 表示电子齿轮命令完成
Busy	执行中	BOOL	TRUE FALSE	FALSE	TRUE 表示功能块的处理没有完成
Active		BOOL			该轴在工作
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	TRUE 表示命令被另一个命令中断
Error	错误	BOOL	TRUE FALSE	FALSE	功能块内部发生错误信号
ErrorID	错误代码	SMC_ERROR	-	0	错误 ID，参阅“SMC_ERROR”

功能说明：

- 这个指令由“SM3_Basic”库实现。
- 开始动作后，从轴以主轴速度乘以齿轮比得到的速度为目标速度，进行加减速动作。
- 该功能块同步开始到同步结束的过程，本质为同步区间内从轴跟随主轴的一个电子凸轮、此时根据主轴范围（MasterSyncPosition- MasterStartDistance，MasterSyncPosition），从轴范围（当前位置，SlaveSyncPosition），指令会根据设置齿轮比以及上述三个参数自动设计一条凸轮曲线，执行同步时从轴跟随主轴完成凸轮动作。
- 注意，如果主从轴工作在线性模式，需保证需保证上述几个参数设置合理否则齿轮动作无法正确进行，故而建议使用该指令时主从轴为模数轴模式。比如主从轴线性工作模式都向正向运动，如果执行指令时主轴位置 MasterSyncPosition-MasterStartDistance，或者从轴位置 > SlaveSyncPosition 则无法切入电子齿轮动作。
- 同步完成（InSync 为 TRUE）的同时达到目标速度，

  此后从轴移动量 = 主轴移动量 × RatioNumerator / RatioDenominator
- 对于 AvoidReversal：如果从轴是模态轴并且主轴速度（齿轮比的倍数关系）不是相对于从轴的速度关系，那么 MC_GearInPos 会尝试着避免从轴的反转。它试图通过增加 5 个从轴周期“拉伸”从轴的运动。如果这个“拉伸”无效，那么会有错误出现并且从轴错误停止。如果从轴速度关联主轴速度（是齿轮比的倍数），那么会有错误出现，并且轴错误停止。如果从轴是线性轴模式轴，那么一个错误会产生在 Execute 输入上升沿处理时。

时序图：

图 6.3 的时序图画出了在 Execute 信号置 True 后，MC_GearInPos 指令的各个主要信号的时序。

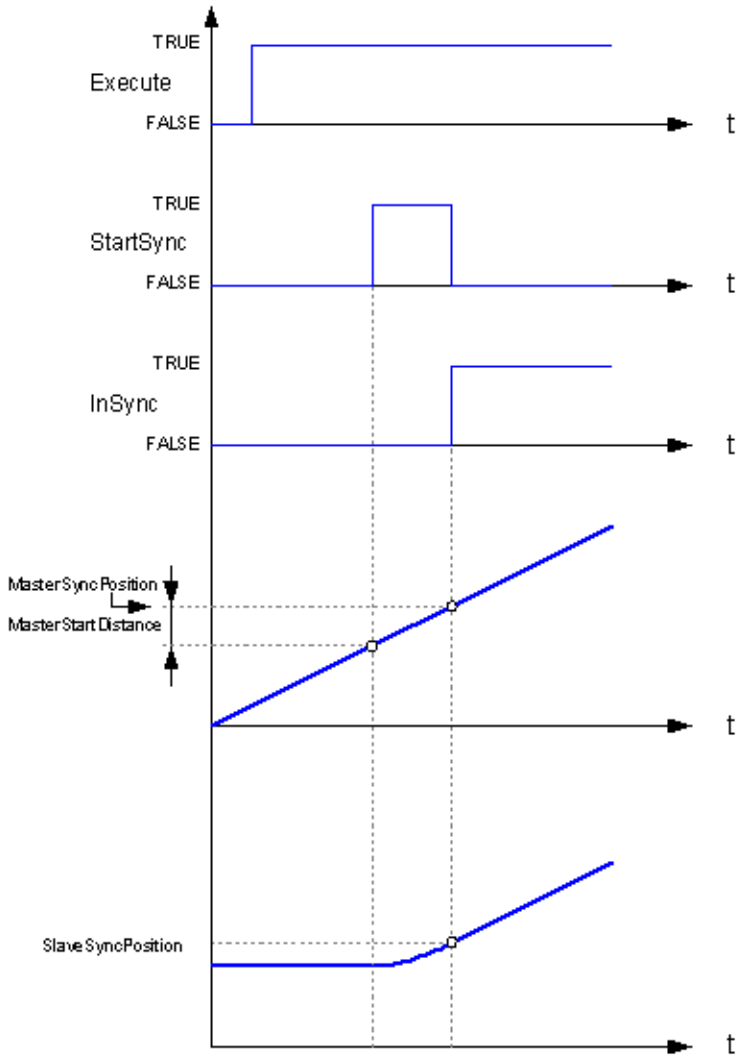


图 6.3　时序图

齿轮脱离 MC_GearOut

将从轴与主轴的电子齿轮耦合断开。

指令外观：

指令	FB/ FUN	图形模块	结构文本
MC_GearOut	FB	MC_GearOut Slave　Done Execute　Busy Error ErrorID	MC_GearOut(Slave：= （参数）， Execute：= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Slave	从轴	AXIS_REF	-	-	映射到主轴，参阅“AXIS_REF_SM3”
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	此输入的一个上升沿将启动功能块的处理
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	TRUE 表示电子凸轮断开
Busy	执行中	BOOL	TRUE FALSE	FALSE	TRUE 表示功能块的处理没有完成
Error	错误	BOOL	TRUE FALSE	FALSE	功能块内部发生错误信号
ErrorID	错误代码	SMC_ERROR	-	0	错误 ID，参阅“SMC_ERROR”

功能说明：

- 这个指令由“SM3_Basic”库实现。
- 切出电子齿轮动完成后此时从轴的速度为切出前的速度，所以需配合以 MC_Stop 指令停止从轴。
- 通过 Slave(从轴)指定动作对象轴，指定 Deceleration(减速度)，中止执行中的 MC_GearIn(齿轮动作开始)指令、MC_GearInPos(位置指定齿轮动作)指令。
- 本指令对 MC_GearIn(齿轮动作开始)指令、MC_GearInPos(位置指定齿轮动作)指令的主轴动作没有影响。

电子齿轮使用方法及例程

电子齿轮使用步骤如下：

步骤一：配置电子齿轮参数：调用 MC_GearIn 指令或者 MC_GearInPos 指令配置电子齿轮工作模式以及参数；

步骤二：调用运动指令，设置对应的运动参数，使得主轴按照一定的规律运动（如恒速运动或者定长运动）；

步骤三：配置成功后从轴也会按照一定的线性关系运动，如果需要停止轴的运动，需要先调用 MC_GearOut 指令断开电子齿轮，再调用 MC_Stop 具体停止某个轴的运动。

注意，电子齿轮使用有两种模式：

1） MC_GearIn 为主从轴同时启停模式，具体见例程 6.1；

2） MC_GearInPos 为主从轴非同时启动但同时停止模式，具体见例程 6.2。

例程 6.1：利用电子齿轮实现主轴（Axis_0）循环正反转（正反转距离为 20000Pulse，运动速度 10000Pulse/s），从轴 Axis_1、Axis_2 分别按照-2 和 0.5 的齿轮比运动。

1) 新建工程，命名 Gear，并添加 PMC_Controller 库到工程中；

2) 在主程序 PLC_PRG 下添加上电使能模块 ACT_Power、配置电子齿轮模块 ACT_Gear、脉冲模块 ACT_Init 和主轴运动模块 ACT_Move，如图 6.4~6.6 所示。

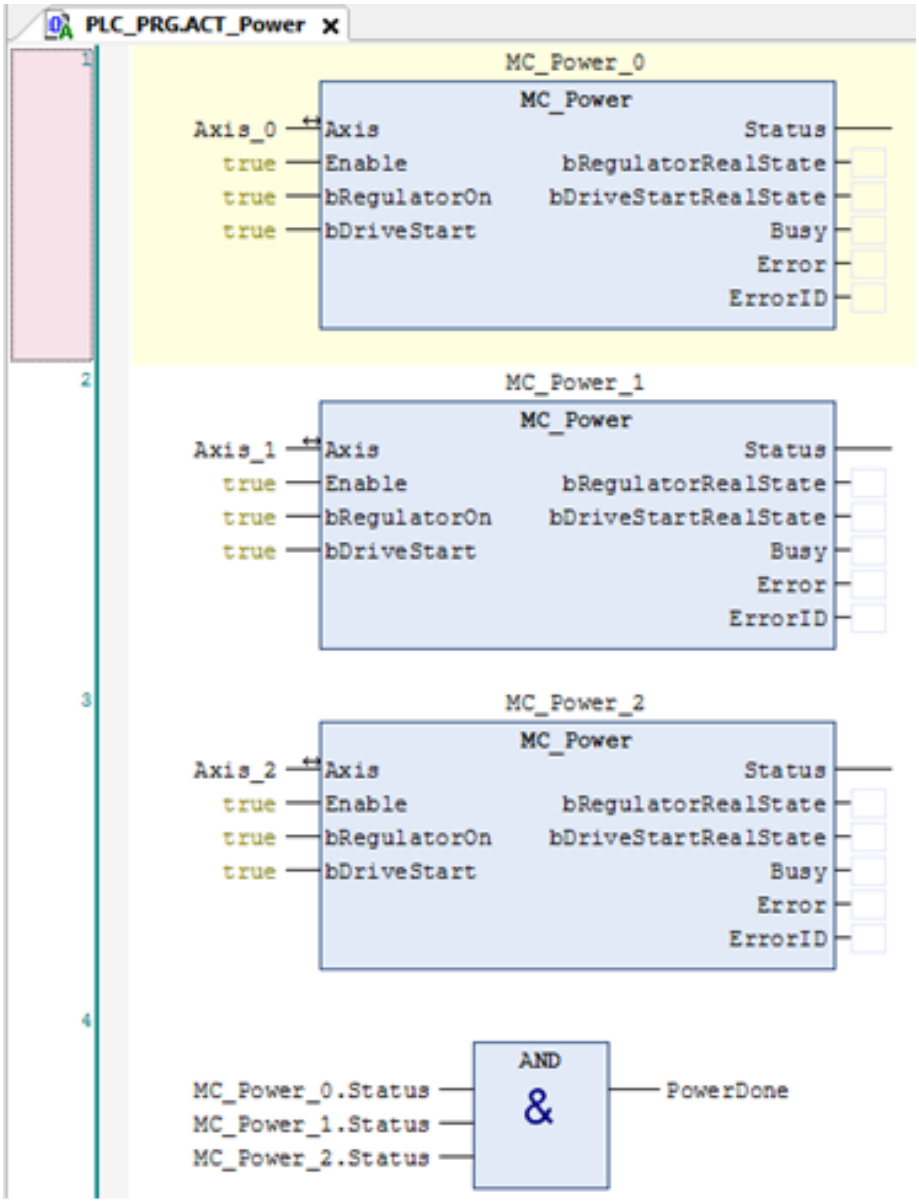


图 6.4 上电初始化模块

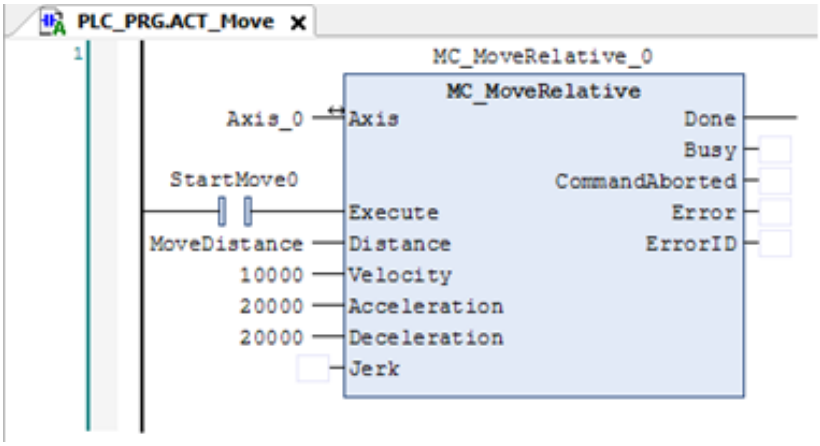


图 6.5　主轴运动模块

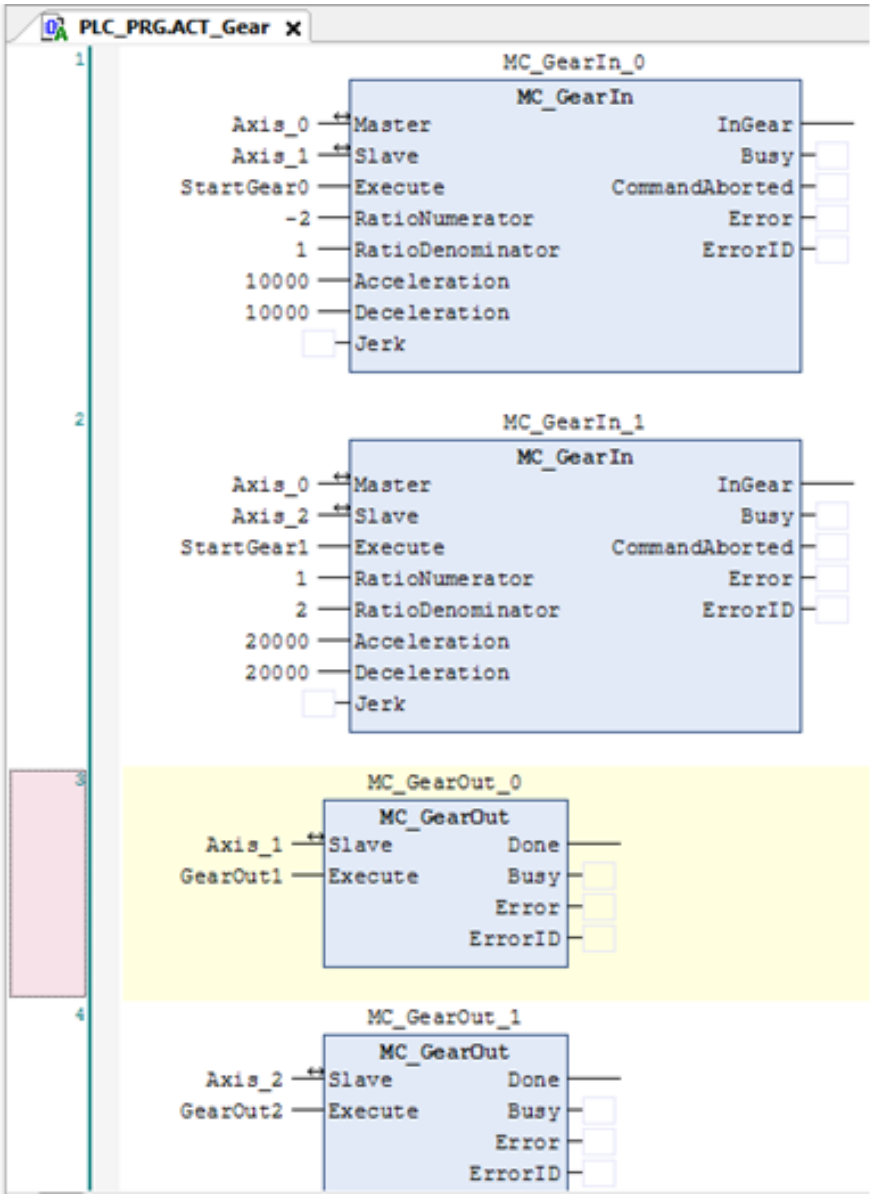


图 6.6　电子齿轮模块

3) 在主程序中分别调用上述模块，并根据功能需求加入设置主轴循环正反转的功能程序，如图 6.7 所示；

4) 程序完成编程后，编译、下载到控制器中执行，并将 iState 的值强制为 1 启动整个电子齿轮运动，程序运行结果如图 6.7、6.8 所示。

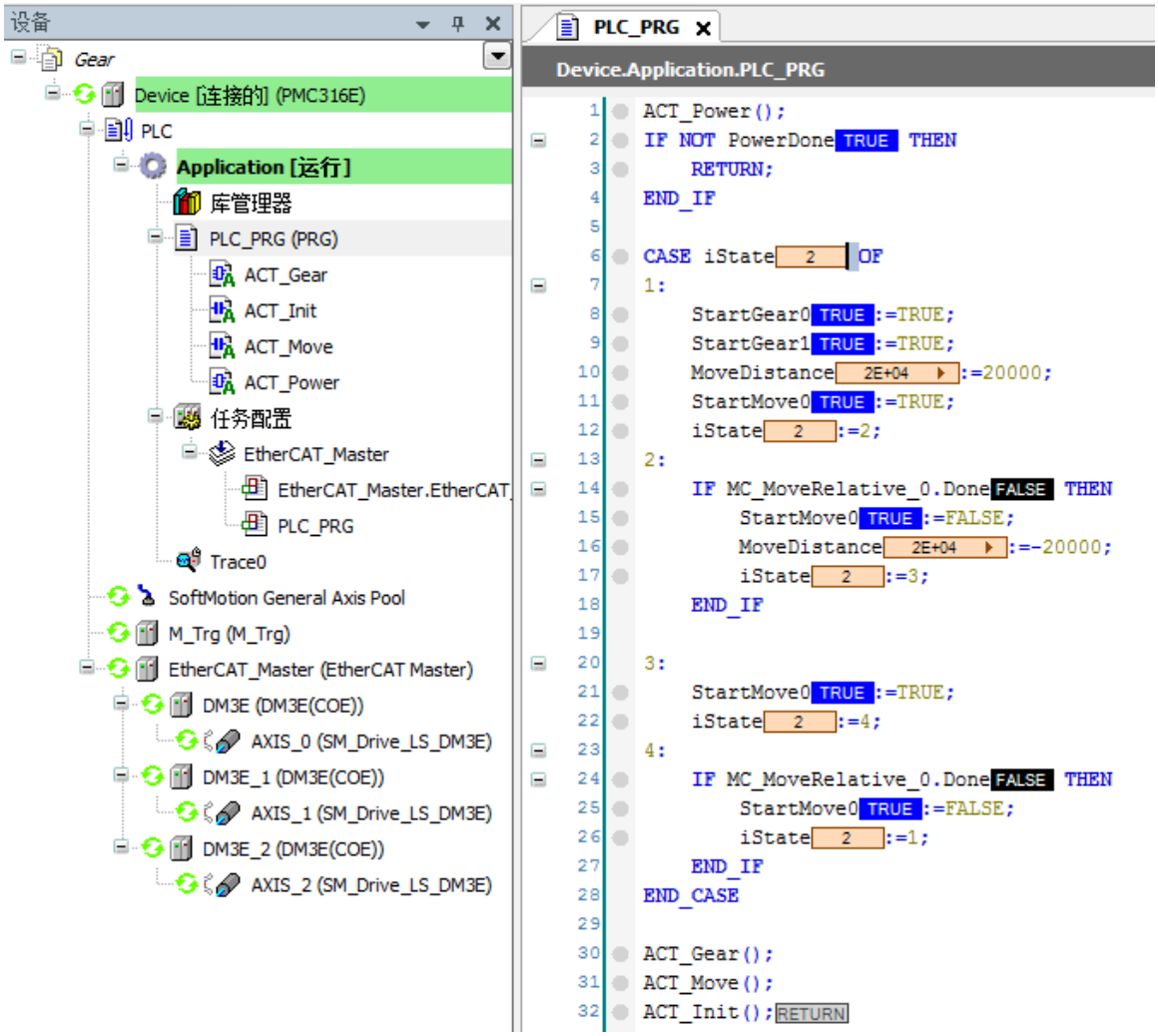


图 6.7　电子齿轮主程序

如图 6.6 所示，设置从轴 Axis_1、Axis_2 和主轴 Axis_0 的齿轮比设置为-2 和 0.5，它们对应的运动参数和主轴的运动参数也是这样的线性关系，如主轴加速度为 10000Pulse/s^2，Axis_1 的加速度为 20000Pulse/s^2，Axis_2 的加速度为 5000Pulse/s^2，位置参数也是一样；主从轴同时启停，同时加减速；此外设置 Axis_1 和主轴的齿轮比为负数，则从轴 Axis_1 的位置和速度的方向和主轴是相反的，如图 6.8 所示。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**同步运动-电子齿轮-Gear**”。

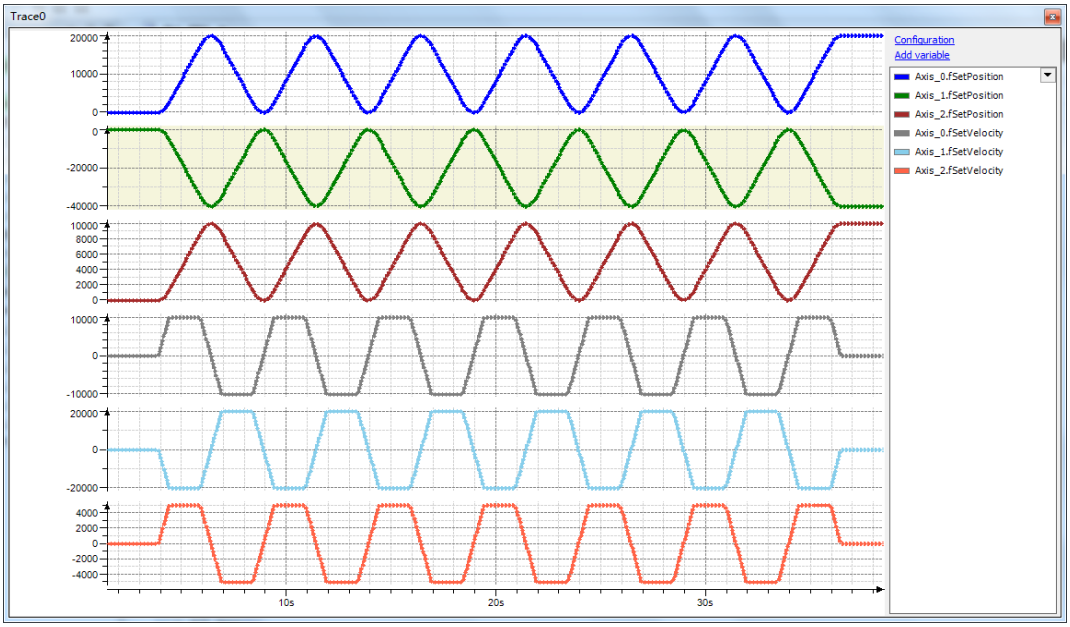


图 6.8　各轴位置和速度曲线

例程 6.2：利用电子齿轮实现主轴（Axis_0）定长运动，运动距离为 10000Pulse，运动速度 10000Pulse/s），从轴 Axis_1 设置为从轴启动到主从轴到达啮合同步这这段时间主轴运动的距离为 4000Pulse、从轴运动的距离为 2000Pulse、主轴当前的位置为 5000Pulse。也就是当主轴运动到 1000Pulse（5000-4000），从轴开始运动直至啮合同步继续以齿轮比为 0.5 的运动。

MC_GearInPos 模式和 MC_GearIn 模式在使用上是类似的，现在在例程 6.1 的基础上将电子齿轮模块的 MC_GearIn 指令改为 MC_GearInPos 指令，主程序实现主轴定长运动而不是正反转运动，其他的同例程 6.1。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**同步运动-电子齿轮-GearInPos**”。

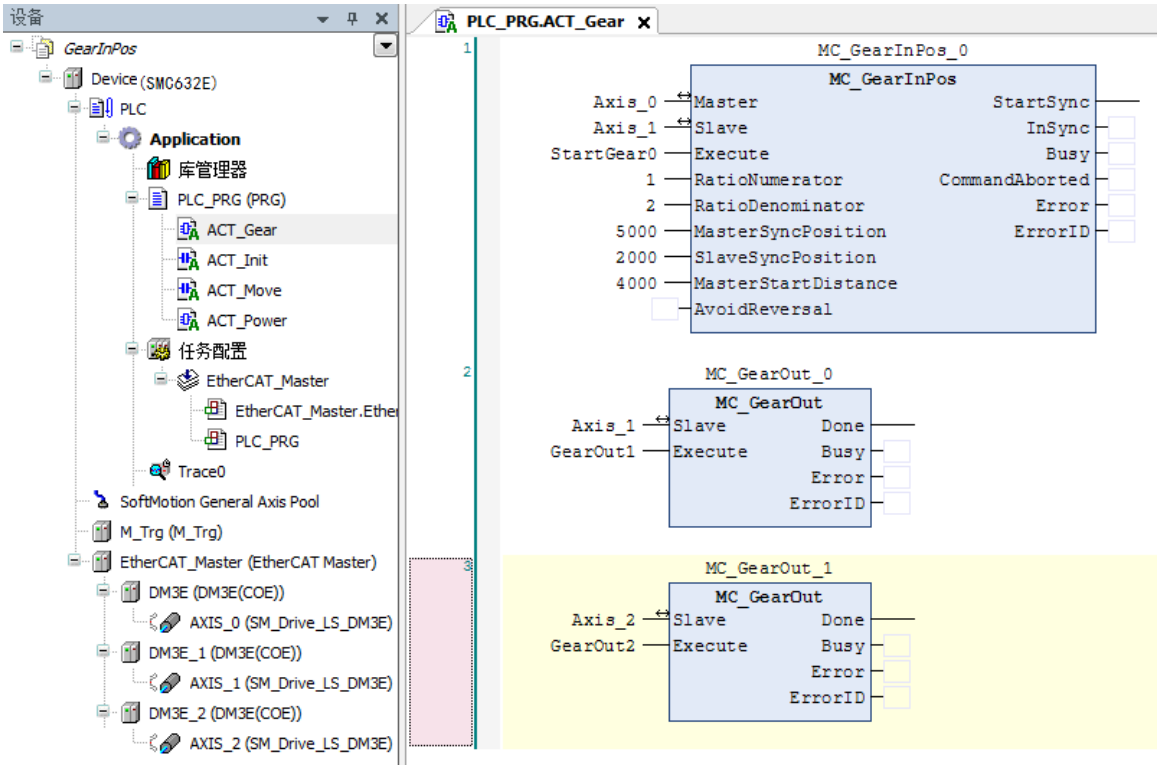


图 6.9　电子齿轮模式二模块

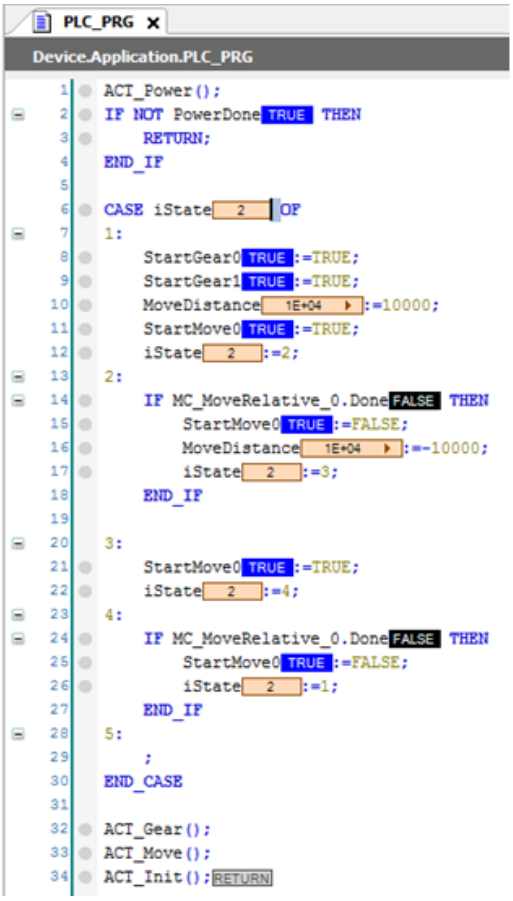


图 6.10　电子齿轮模式二主程序

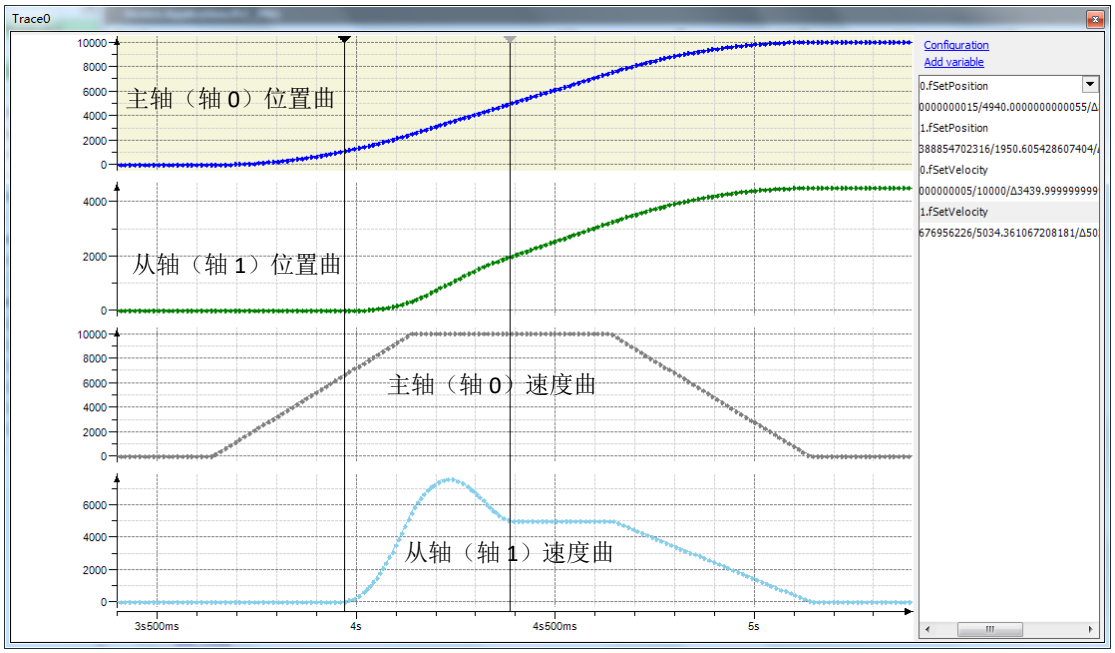


图 6.11　电子齿轮模式二主从轴位置和速度曲线

如图 6.11 所示，当从轴 Axis_1 开始运动到主从轴啮合达到同步时，主轴运动的距离为两光标间的距离（5000-1000）Pulse，从轴运动的距离为 2000Pulse，主轴运动总距离为 5000Pulse；达到啮合同步后从轴以 10000×0.5 的速度运动直至同主轴同时停止。如果此时将主轴设置为 10000Pulse 循环正反转（在例程 6.2 中主程序 CASE 语句为 2 的功能块中将 iState：=5 改为 iState：=3 即可），后面的运动将会一直同步下去，如图 6.12 所示。

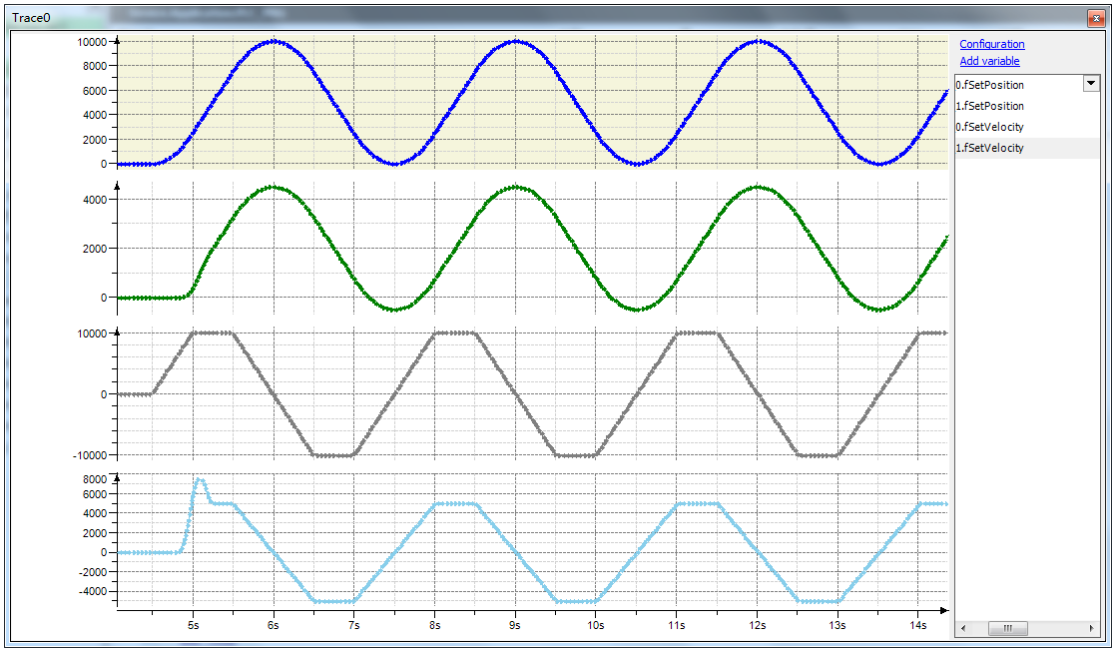


图 6.12　电子齿轮模式二主从轴位置和速度曲线（主轴循环正反转）

需要注意的是电子齿轮 MC_GearInPos 指令中，参数 MasterSyncPosition 为主从轴达到同步时主轴的位置，SlaveSyncPosition 为主从轴达到同步时从轴的位置，MasterStartDistance 为从从轴开始运动到主从轴达到同步这段间隔主轴运动的距离。

6.2 电子凸轮指令

PMC600 提供电子凸轮的功能，以实现主从轴动态、非线性的运动。电子凸轮可以有多种使用方式，比如电子凸轮用于弹簧机效率相当高，不同的轴用来控制各自的送料、弯曲和倾斜，这样可以制造任意需要的外形（斜面、圆锥等）；当然也可以用于追剪系统中，主轴送料系统按一定速度运动送料，从轴切刀机构按照设定的凸轮表参数匹配主轴运动，达到同步后切刀开始切料，完成后切刀机构高速返回进行下一次的循环运动，当然这个需要用户不断实验来匹配出最佳的凸轮参数。

在图 6.13 中，我们可以看到水平轴显示主轴位置值，垂直轴显示从轴位置值。凸轮在特定的区域内（凸轮控制周期）针对每个主轴有一个独立的从轴位置值，当凸轮驱动连接激活时，从轴必须遵循此轮廓。

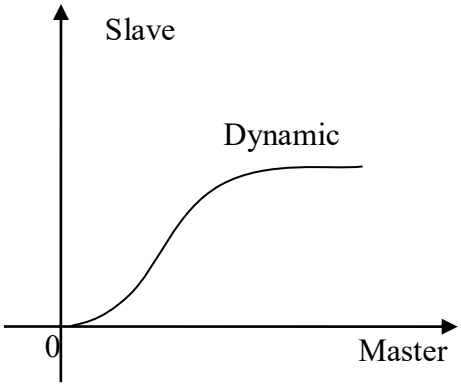


图 6.13　电子凸轮示意图

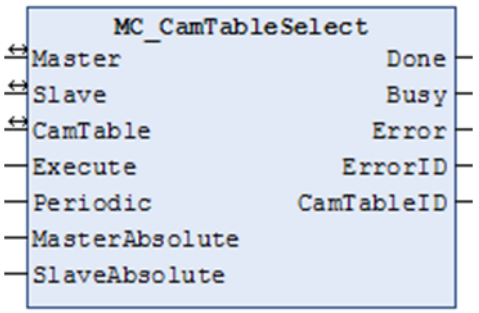
表 6.2　电子凸轮相关指令

名称	功能
MC_CamIn	配置主从轴电子凸轮参数并启动
MC_CamTableSelect	将选择的 cam 表连接到实际的凸轮表上
MC_CamOut	断开主从轴电子凸轮
SMC_GetTappetValue	读取当前的挺杆状态

凸轮挺杆关联　MC_CamTableSelect

用于选择需要执行的 cam 表，需要和 MC_CamIn 指令配合使用。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_CamTableSelect	FB		MC_CamTableSelect(Master：= （参数）， Slave：= （参数）， CamTable：= （参数）， Execute：= （参数）， Periodic：= （参数）， MasterAbsolute：= （参数）， SlaveAbsolute：= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）， CamTableID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Master	主轴	AXIS_REF	-	-	映射到主轴，参阅“AXIS_REF_SM3”
Slave	从轴	AXIS_REF	-	-	映射到主轴，参阅“AXIS_REF_SM3”
CamTable	凸轮表	MC_CAM_REF	-	-	映射为 cam 表格描述，参阅 MC_CAM_REF
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	此变量的一个上升沿将会启动功能块的处理
Periodic	重复模式	BOOL	TRUE FALSE	TRUE	True：周期、重复地执行所指定的凸轮表 FALSE：仅执行一次凸轮表
MasterAbsolute	主轴绝对模式	BOOL	TRUE FALSE	TRUE	TRUE 表示绝对坐标，FALSE 表示相对坐标
SlaveAbsolute	从轴绝对模式	BOOL	TRUE FALSE	TRUE	TRUE 表示绝对坐标，FALSE 表示相对坐标
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	TRUE 表示 cam 表格选择结束
Busy	执行中	BOOL	TRUE FALSE	FALSE	TRUE 表示功能块的处理没有完成
Error	错误	BOOL	TRUE FALSE	FALSE	功能块内部发生错误信号
ErrorID	错误代码	SMC_ERROR	-	0	错误 ID，参阅 SMC_Error
CamTableID	挺杆 ID	MC_CAM_ID	-	-	cam 表格的定义，用于功能块 MC_CamID

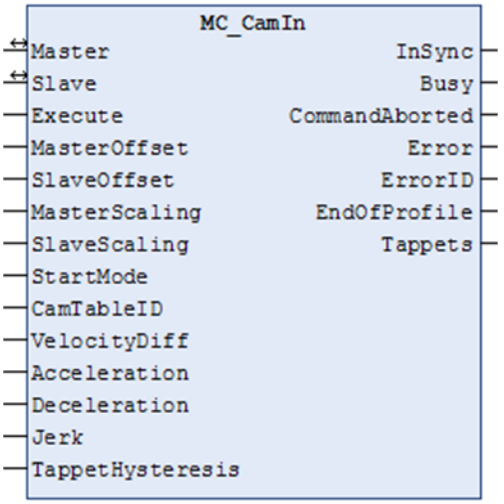
功能说明：

- 这个指令由“SM3_Basic”库实现。
- 这个指令用于指定电子凸轮运行所需凸轮表，所以在使用本指令之前先要将凸轮表编辑好（凸轮编辑器编辑或者在线编辑好）。
- Execute 上升沿，执行指定凸轮表，亦可凸轮表更新后刷新指定的凸轮表。
- Done 信号输出为 TRUE 时，则输出变量“CamTableID”生成并且生效。
- 主轴和从轴不能指定为同一轴，否则会有报错输出。

电子凸轮关联　MC_CamIn

使用指定的凸轮表开始执行电子凸轮动作。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_CamIn	FB		MC_CamIn(Master：= （参数）， Slave：= （参数）， Execute：= （参数）， MasterOffset：= （参数）， SlaveOffset：= （参数）， MasterScaling：= （参数）， SlaveScaling：= （参数）， StartMode：= （参数）， CamTableID：= （参数）， VelocityDiff：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， TappetHysteresis：= （参数）， InSync=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， Error=> （参数）， ErrorID=> （参数）， EndOfProfile=> （参数）， Tappets=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Master	主轴	AXIS_REF	-	-	映射到主轴，参阅“AXIS_REF_SM3”
Slave	从轴	AXIS_REF	-	-	映射到主轴，参阅“AXIS_REF_SM3”

VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	一个上升沿将启动功能块的处理
MasterOffset	主轴偏移	LREAL	负数，0，正数	0	主轴表格的偏移
SlaveOffset	从轴偏移	LREAL	负数，0，正数	0	从轴列表的偏移
MasterScaling	主轴比例	LREAL	正数	1	主轴缩放比例
SlaveScaling	从轴比例	LREAL	正数	1	从轴缩放比例
StartMode	从轴啮合方式	MC_StartMode	0-4	0：absolute	0：absolute 绝对位置 ：1：relative 相对位置 ：2：ramp_in (斜坡切入) 3：ramp_in_pos（正向斜坡切入）4：ramp_in_neg 反向斜坡切入
CamTableID	凸轮表	MC_CAM_ID	-	-	定义 cam 表格的使用，是 MC_CamID 的输出
VelocityDiff	速度	LREAL	负数，0 正数	0	与 ramp_in 不同的最大速度
Acceleration	加速度	LREAL	0，正数	0	对 ramp_in 的最大加速度
Deceleration	减速度	LREAL	0，正数	0	对 ramp_in 的最大减速度
Jerk	目标加加速度	LREAL	负数，0，正数	0	加加速度
TappetHysteresis	挺杆阻尼	LREAL	0 正数	0	挺杆滞后的尺寸大小
VAR_OUTPUT					
InSync	同步中	BOOL	TRUE FALSE	FALSE	TRUE 表示从轴根据 cam 表格与主轴同步
Busy	执行中	BOOL	TRUE FALSE	FALSE	TRUE 表示功能块的处理没有完成
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	TRUE 表示命令被另一个命令中断
Error	错误	BOOL	TRUE FALSE	FALSE	功能块内部发生错误
ErrorID	错误代码	SMC_ERROR	-	0	错误 ID，参阅 SMC_Error
EndOfProfile	曲线完成	BOOL	TRUE FALSE	FALSE	在 cam 编译周期结束输出的脉冲信号
Tappets	挺杆表	SMC_TappetData	-	-	用于 SMC_GetTappetValue 处理的挺杆信号

功能说明：

- 这个指令由“SM3_Basic”库实现。
- 本指令可使相位(主轴)和位移(从轴)按照凸轮表进行同步凸轮动作。
- 由本指令指定的凸轮表有两种制作方式，

1） 使用凸轮编辑器编制；

2） 通过编程自建凸轮表数据结构。

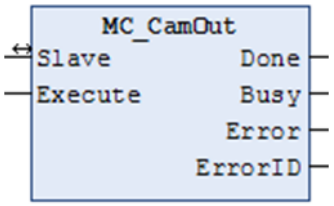
- 在一个凸轮系统中，要调用一条凸轮曲线，先调用 MC_CamTableSelect 指令选择相应的凸轮表，再执行 MC_CamIn；如要更换凸轮曲线，则再调用 MC_CamTableSelect 指令重新选择凸轮表。

- 需使用 MC_CamOut 指令断开主从轴的凸轮耦合关系。
- 该指令执行时 ， 该指令的从轴再执行其它运动指令时，从轴和主轴之间的凸轮关系会解除，并且 MC_CamIn 的 Command-Aborted 变量输出为 TRUE。

电子凸轮脱离 MC_CamOut

解除指定的从轴与其对应主轴的凸轮耦合关系。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_CamOut	FB		MC_CamOut(Slave：= （参数）， Execute：= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Slave	从轴	AXIS_REF	-	-	映射到主轴，参阅“AXIS_REF_SM3”
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	此变量的一个上升沿将会启动功能块的处理
VAR_OUTPUT					
Done	执行完成	BOOL	TRUE FALSE	FALSE	TRUE，如果 cam 已经被分离
Busy	执行中	BOOL	TRUE FALSE	FALSE	TRUE，如果功能块的处理没有完成
Error	错误	BOOL	TRUE FALSE	FALSE	功能块内有错误信号发生
ErrorID	错误码	SMC_ERROR	-	0	错误 ID，参阅 SMC_Error

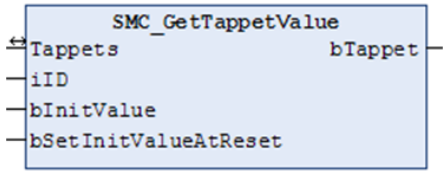
功能说明：

- 这个指令由“SM3_Basic”库实现。
- 本指令用于解除指定的从轴与其对应主轴的凸轮耦合关系。
- Execute 上升沿时执行从轴的凸轮耦合关系断开
- 凸轮关系断开后，从轴并不一定是停止的。如果从轴在执行该指令前速度不为 0，则指令 DONE 信号完成后，凸轮耦合关系断开，但是从轴任然按照切出去前速度运行
- 从轴没有凸轮耦合关系执行该指令，会有 ERROR 报错发生。

读取挺杆状态　SMC_GetTappetValue

读取当前的挺杆状态，需要和 MC_CamIn 指令配合使用。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GetTappetValue	FB		SMC_GetTappetValue(Tappets:= （参数）， iID:= （参数）， bInitValue:= （参数）， bSetInitValueAtReset:= （参数）， bTappet=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Tappets	挺杆	SMC_TappetData	-	-	通过 MC_GetTappetValue 评估的挺杆信号，参阅 SMC_TappetData
VAR_INPUT					
iID	挺杆组号	INT	0，正数	0	要评估的挺杆组 ID
bInitValue	初始值	BOOL	TRUE FALSE	FALSE	第一次调用时分配的挺杆初始值
bSetInitValueAtReset	挺杆复位	BOOL	TRUE FALSE	FALSE	如果为 TRUE，在 CamIn 功能块重启的时候挺杆的输出值将会被设置为初始值。如果为 FALSE，挺杆的值在 CamIn 功能块重启的时候将会被保持。
VAR_OUTPUT					
bTappet	挺杆值	BOOL	TRUE FALSE	FALSE	挺杆的值

功能说明：

- 这个指令由“SM3_Basic”库实现。
- 在机械凸轮结构中，凸轮的旋转运动靠挺杆转换为垂直方向的运动。电子凸轮的挺杆，与机械挺杆类似，相当于电子化的机械挺杆。电子凸轮的挺杆就是凸轮主轴在规定的位置，给出个高电平 或者低电平，作为开关去使用。
- 该功能块需要和 MC_CamIn 指令配合使用，用来获取凸轮表管理的挺杆的数值。

表格形式凸轮使用说明及例程

电子凸轮使用步骤如下：

1） 新建凸轮表：右击工程设备栏中的“Application”，选择“添加对象”-“cam 表”，命名并打开，添加相应的凸轮表参数；

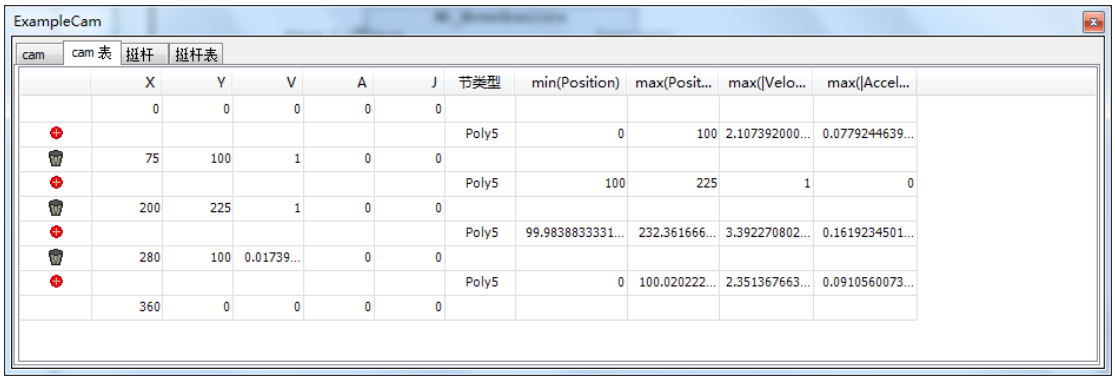
2） 调用 MC_CamTableSelect 指令将选择的 cam 表连接到实际的凸轮表，并获取 cam 表 ID 号供后续指令使用；

3） 在获取 cam 表 ID 号后调用 MC_CamIn 指令配置电子凸轮参数并启动电子凸轮；

4） 根据功能需求调用 SMC_GetTappetValue 指令读取当前电子凸轮挺杆值；

5）需要断开电子凸轮时调用 MC_CamOut 断开主从轴间的联系即可，再调用 MC_Stop 来停止具体某个轴的运动。

例程 4.3：利用电子凸轮实现主轴做定长运动（速度为 200Pulse/s，运动距离为 1080Pulse），从轴按照图 6.14 示的凸轮表运动，设置主从比为 1，即主从轴同步的时候速度相同。


ExampleCam

cam | cam 表 | 挺杆 | 挺杆表

	X	Y	V	A	J	节类型	min(Position)	max(Posit...	max(|Velo...	max(|Accel...
	0	0	0	0	0					
						Poly5	0	100	2.107392000...	0.0779244639...
	75	100	1	0	0					
						Poly5	100	225	1	0
	200	225	1	0	0					
						Poly5	99.9838833331...	232.361666...	3.392270802...	0.1619234501...
	280	100	0.01739...	0	0					
						Poly5	0	100.020222...	2.351367663...	0.0910560073...
	360	0	0	0	0					

图 6.14　凸轮表数据配置

1) 新建工程，命名 CAM，并添加 PMC_Controller 库到工程中；

2) 右击工程设备栏中的“Application”，选择“添加对象”-“cam 表”，命名为 ExampleCam 并打开，添加如图 6.14 相应的凸轮表参数，如图 6.15 所示；

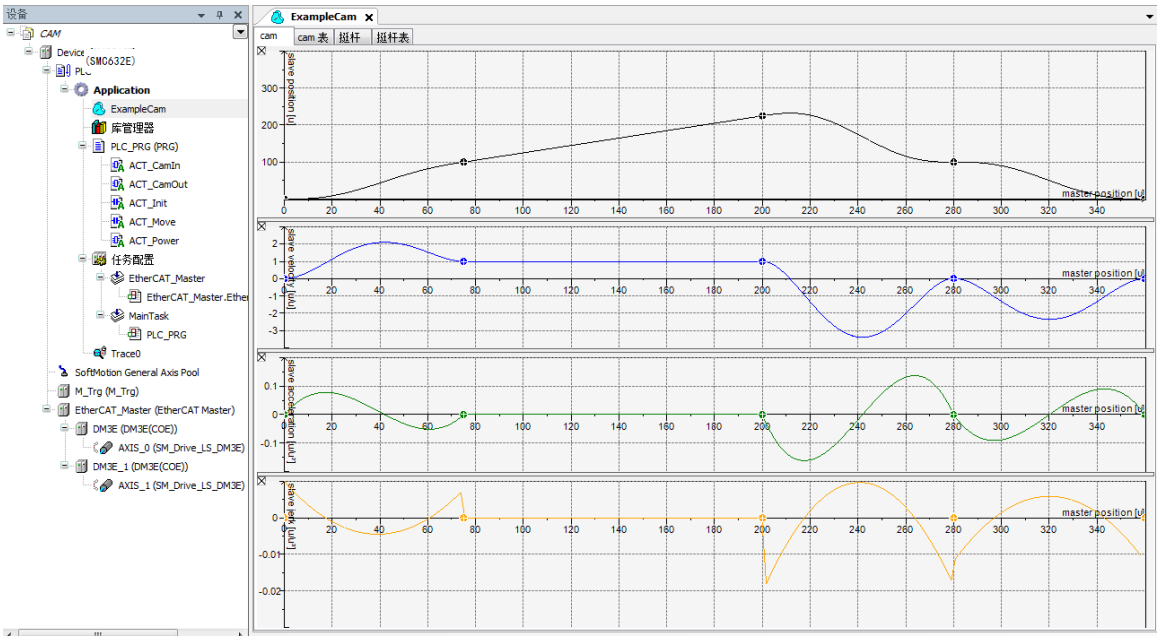


图 6.15　配置凸轮表数据

3) 在主程序 PLC_PRG 下添加上电使能模块 ACT_Power、配置电子凸轮模块 ACT_CamIn、脉冲模块 ACT_Init 和主轴运动模块 ACT_Move 和断开电子凸轮模块 ACT_CamOut，如图 6.16~6.20 所示。

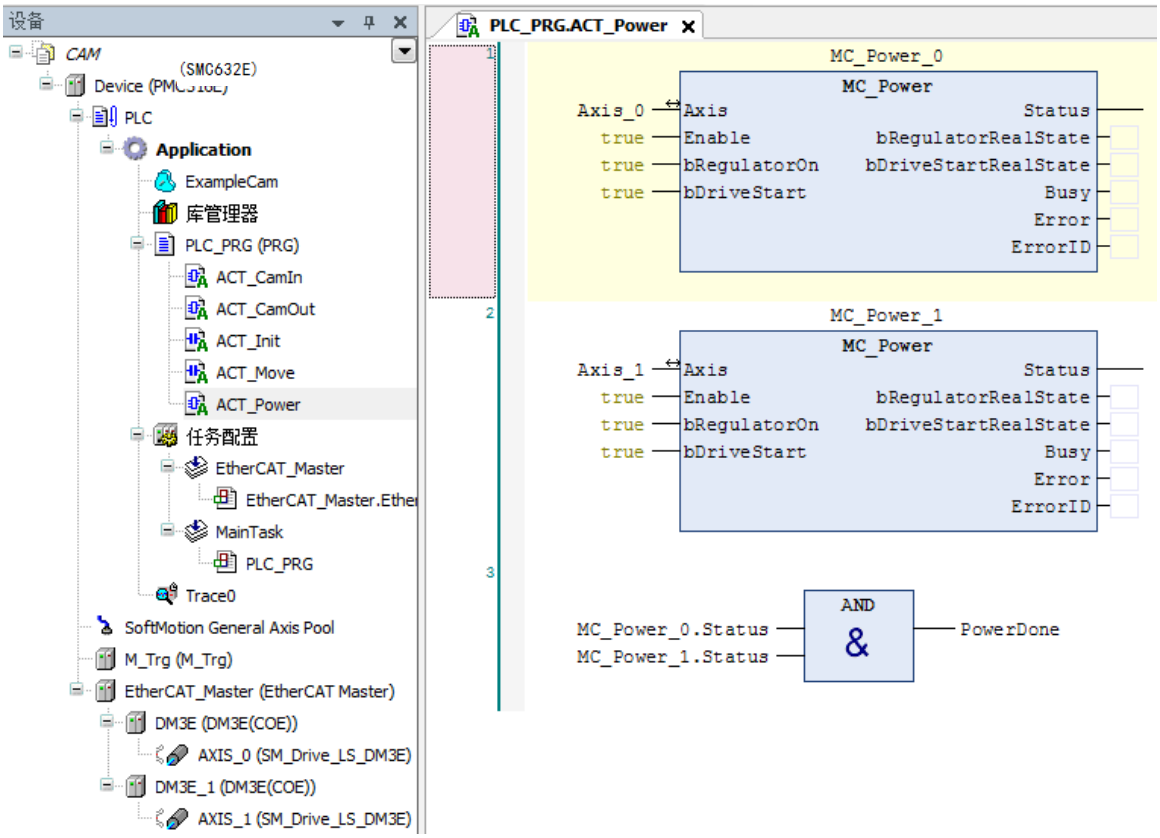


图 6.16　上电模块

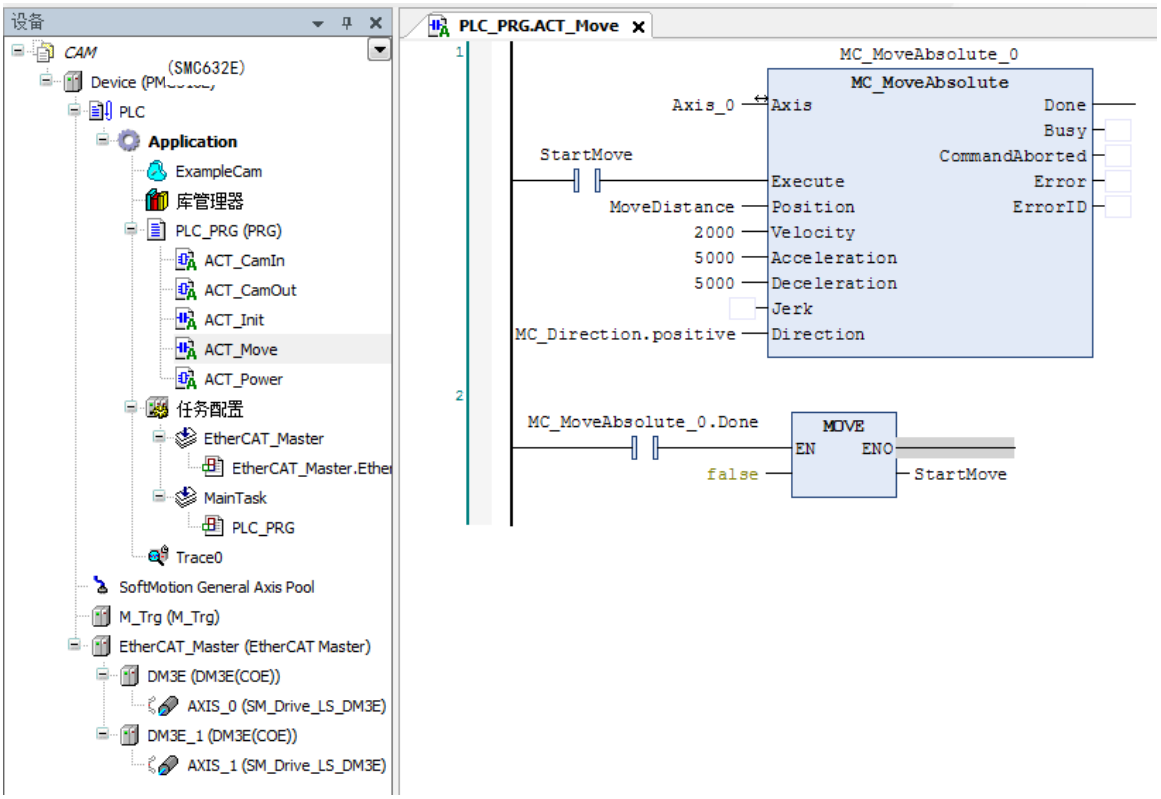


图 6.17　主轴运动模块

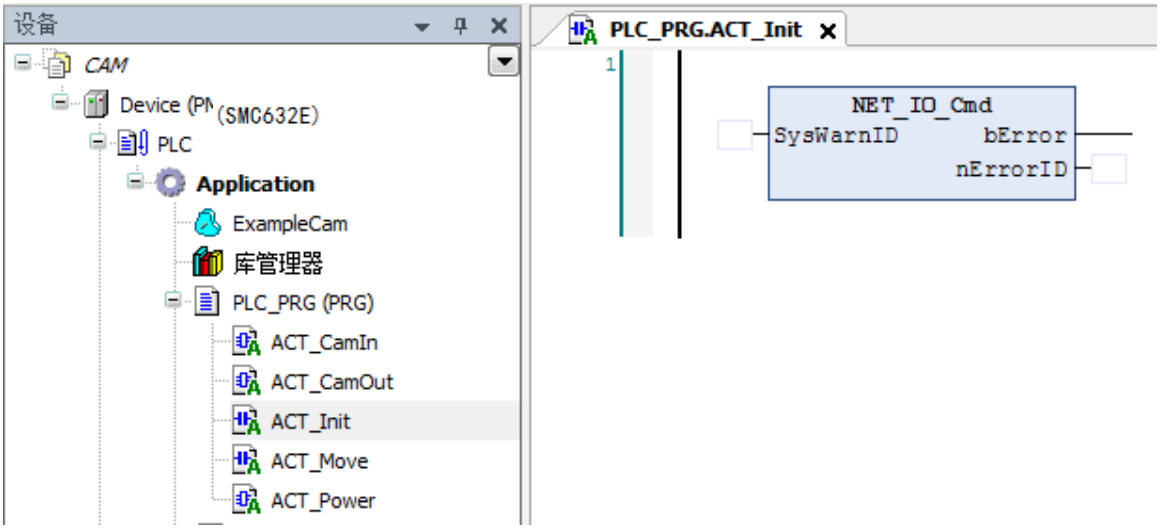


图 6.18　脉冲模块

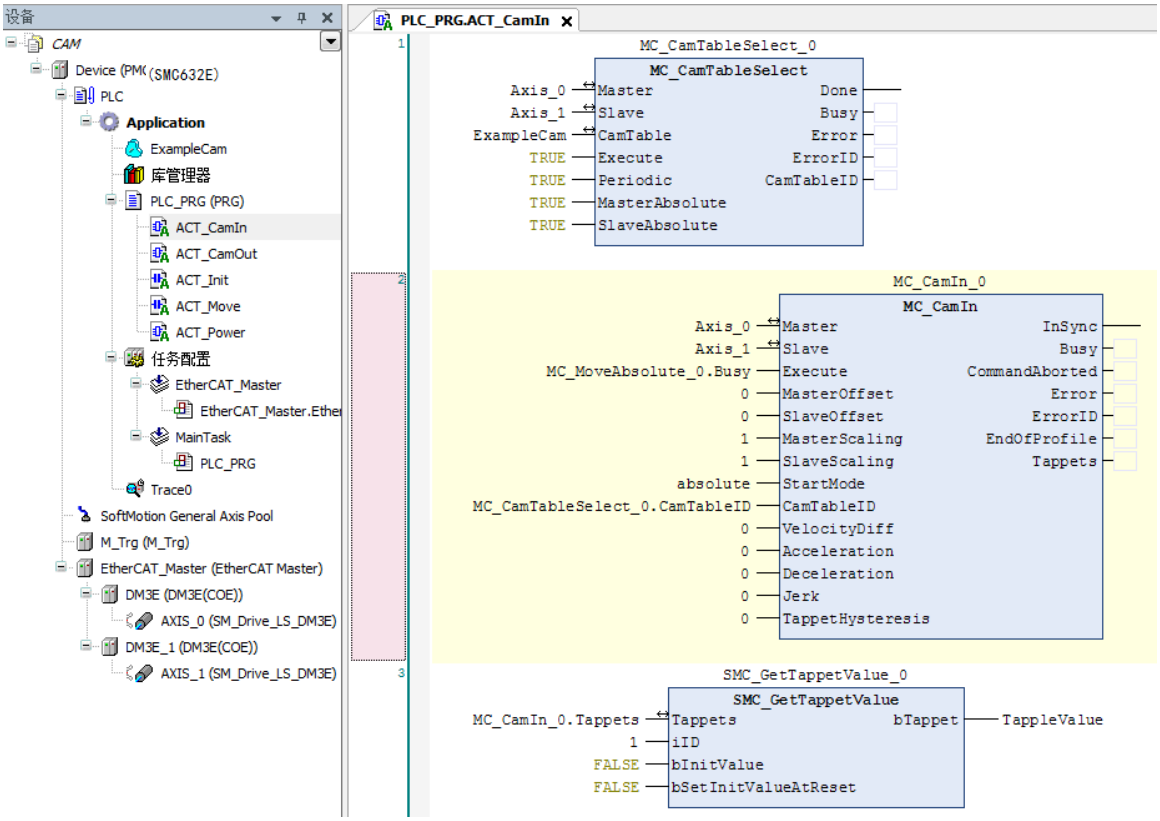


图 6.19　配置电子凸轮模块

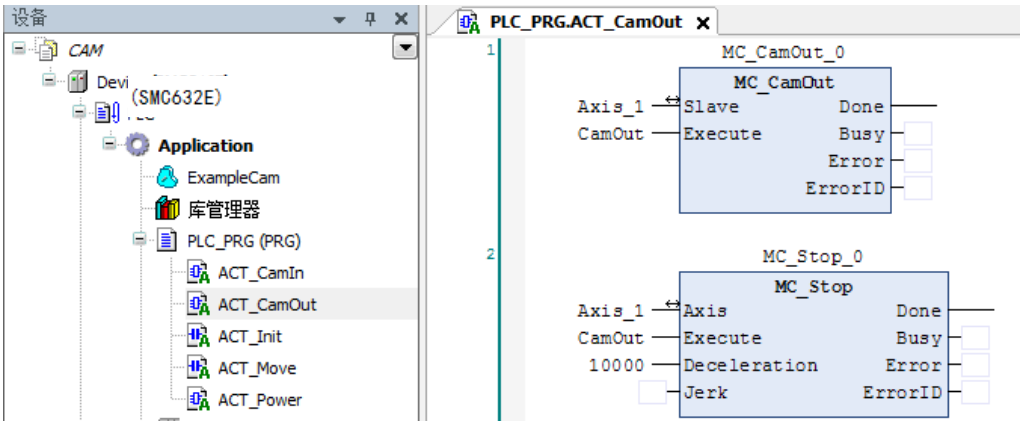


图 6.20　断开电子凸轮

4) 在主程序中分别调用上述模块，并根据功能需求加入设置断开电子齿轮使能条件的功能程序；

5) 程序完成编程后，编译、下载到控制器中执行，并将主轴运动模块的使能变量 StartMove 的值强制为 TRUE 启动整个电子凸轮运动，程序运行结果如图 6.21 和图 6.22 所示。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**同步运动-电子凸轮-CAM**”。

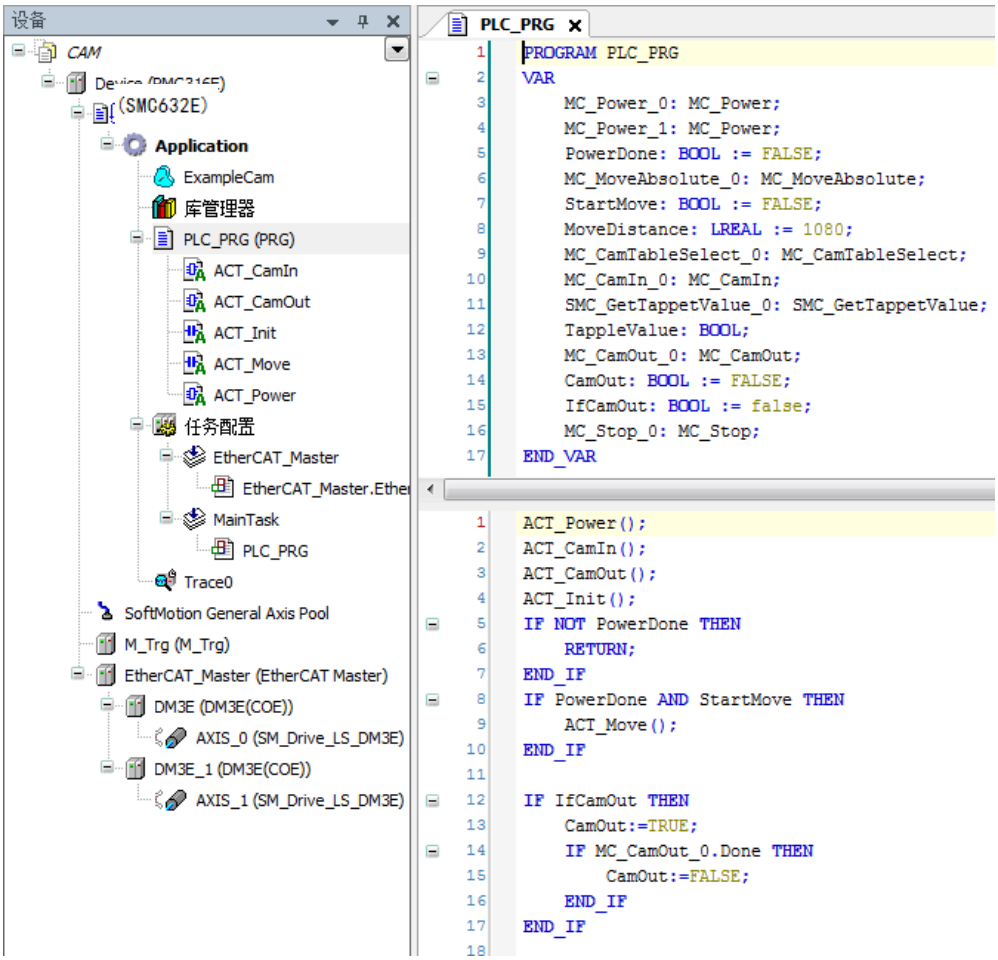


图 6.21　电子凸轮运动主程序

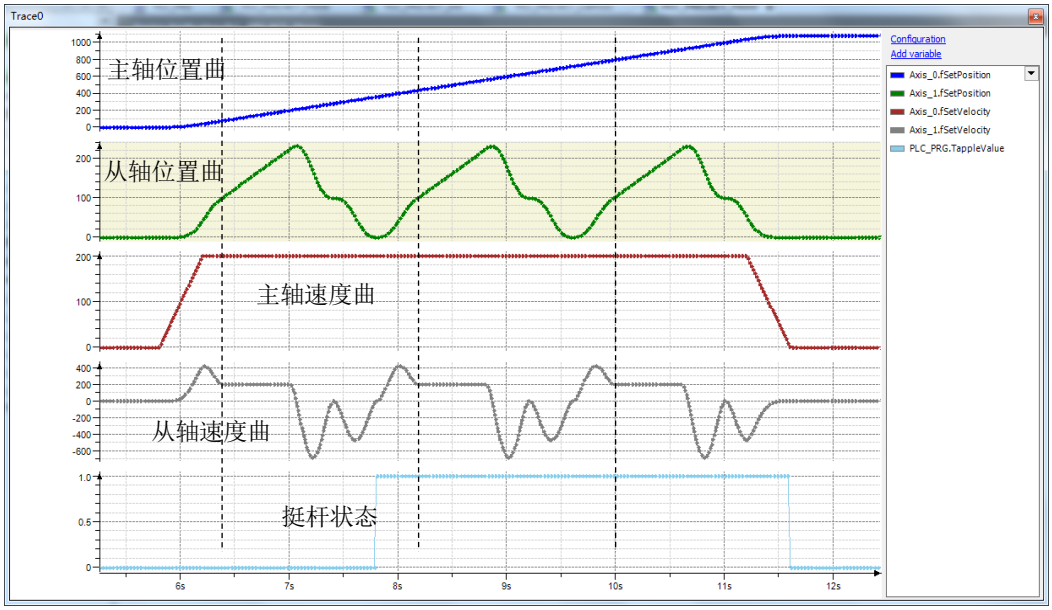


图 6.22　电子凸轮运动主从轴位置速度曲线

从轴在主轴运动到 75Pulse 位置时两者达到同步状态，如图 6.22 虚线所示。由于设置的主从比为 1，从轴同步的速度和主轴一样均为 200Pulse，在主轴运动到 200Pulse 位置，

主从轴脱离同步区域，从轴按照凸轮表设置的参数继续运动。进入到下一个周期，主从轴依然按照这样的关系运动直到主轴停止运动。如果设置的主从比为其他值，则从轴对应的运动参数则是凸轮表中从轴的数据和主从比的乘积。

编程形式凸轮使用说明及例程

电子凸轮使用步骤：

1） 配置凸轮表参数：主轴范围、从轴范围、起始标志点、过程标志点、结束标志点等等；

2） 调用 MC_CamTableSelect 指令将配置的凸轮表连接到实际凸轮表，并获取 cam 表 ID 号供后续指令使用；

3） 在获取 cam 表 ID 号后调用 MC_CamIn 指令配置电子凸轮参数并启动电子凸轮；

4） 启动主轴运动。

5） 运动完成断开电子凸轮时调用 MC_CamOut 断开主从轴间的联系即可，再调用 MC_Stop 来停止具体某个轴的运动。

例程 4.4：利用电子凸轮实现主轴做定长运动（速度为 10000Pulse/s，运动距离为 10000Pulse），从轴按照表 6.3 所示的凸轮表运动，设置主从比为 1，即主从轴同步的时候速度相同。

表 6.3 从轴凸轮表

标志点	主轴	从轴
1	0	0
2	5000	5000
3	10000	7000

1) 新建工程，命名 CAMOrder，并添加 PMC_Controller 库到工程中；

2) 右击工程设备栏中的“PLC_PRG”，选择“添加对象”-“动作”，命名为 ACT_PRM 并打开，添加表 6.3 相应的凸轮表参数，如图 6.23 所示；

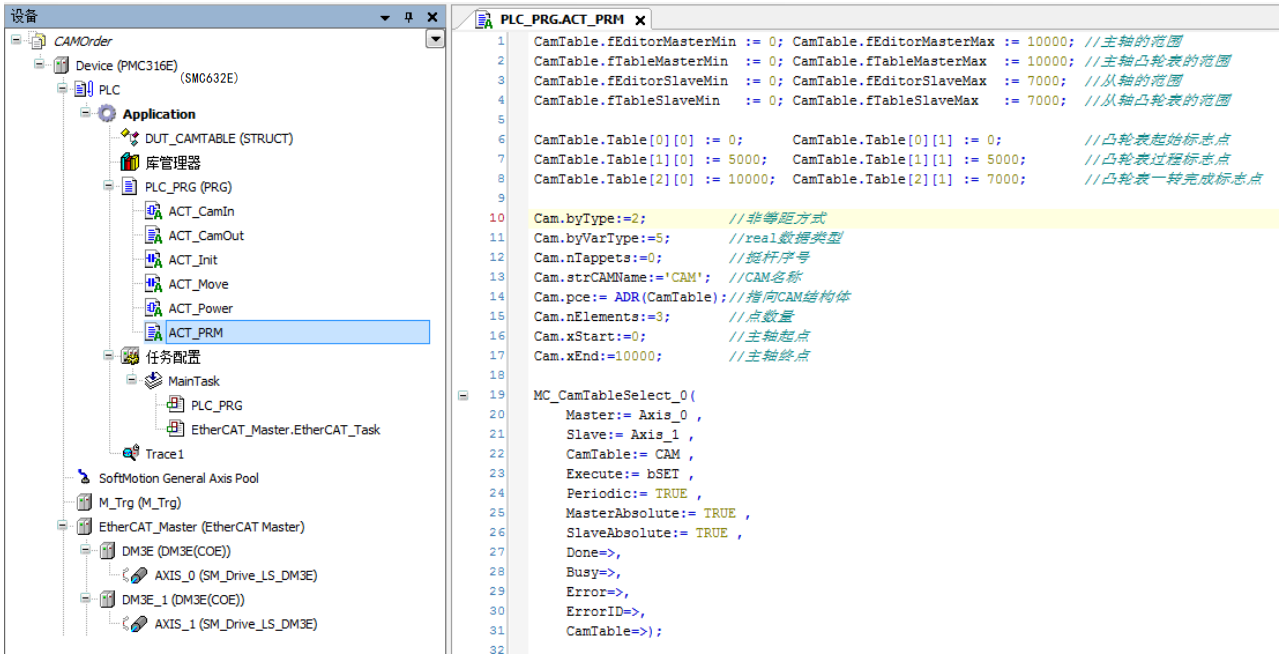


图 6.23　配置凸轮表数据

3)　在主程序 PLC_PRG 下添加上电使能模块 ACT_Power、脉冲模块 ACT_Init、主轴运动模块 ACT_Move、配置电子凸轮模块 ACT_CamIn 和断开电子凸轮模块 ACT_CamOut，如图 6.24~6.28 所示。

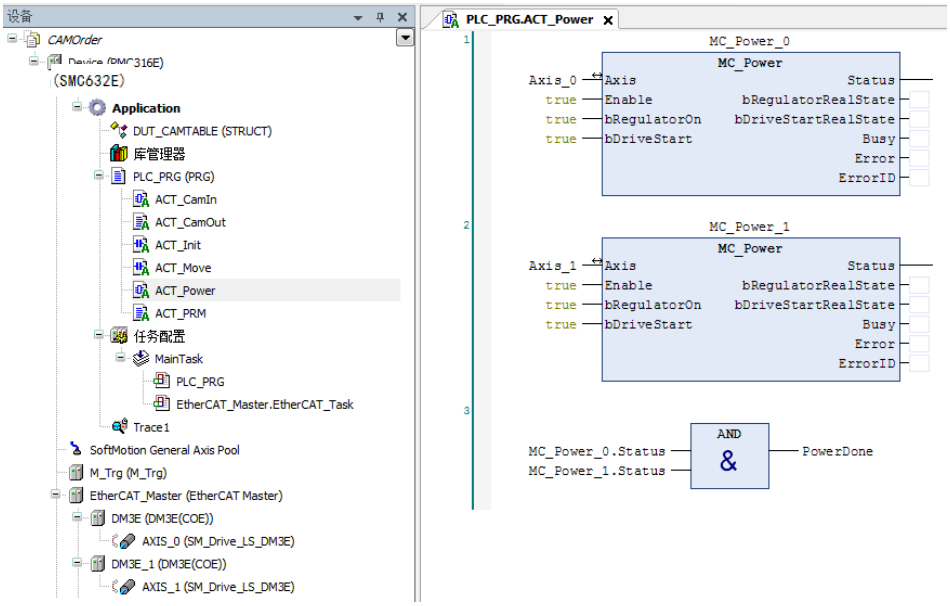


图 6.24　上电模块

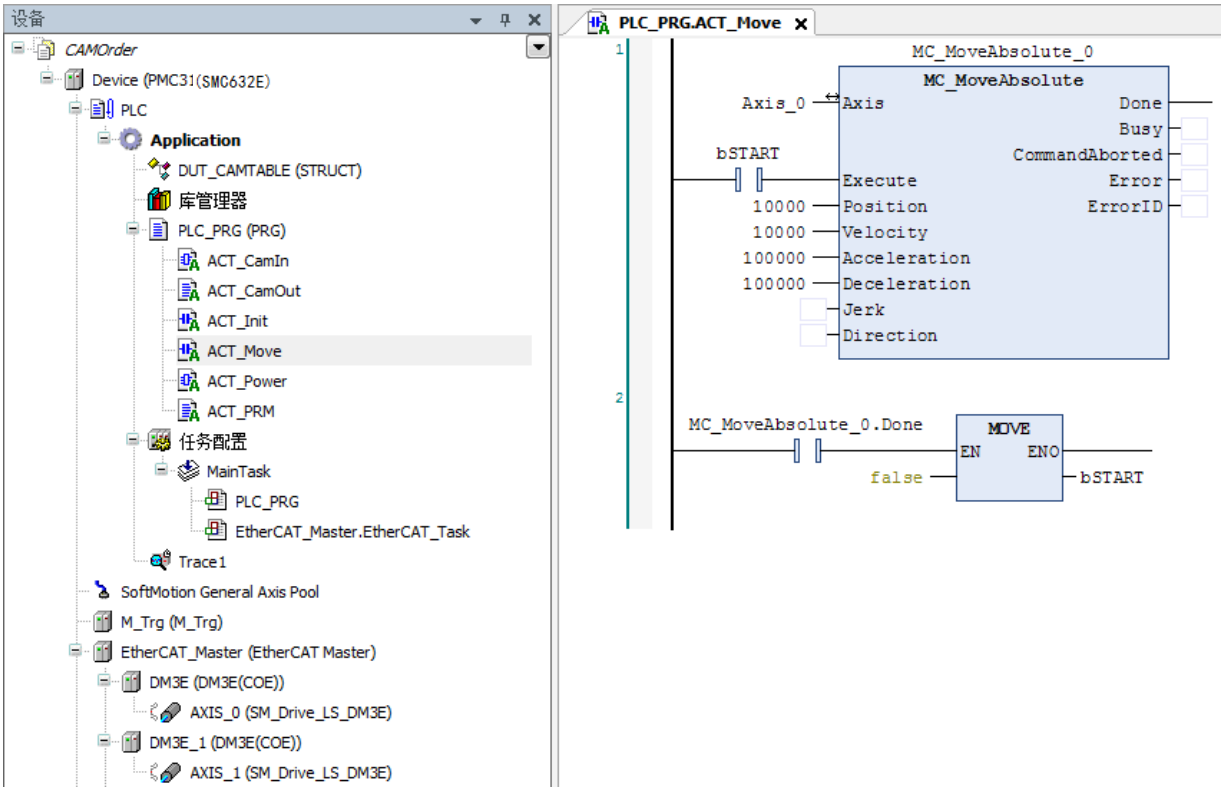


图 6.25 主轴运动模块

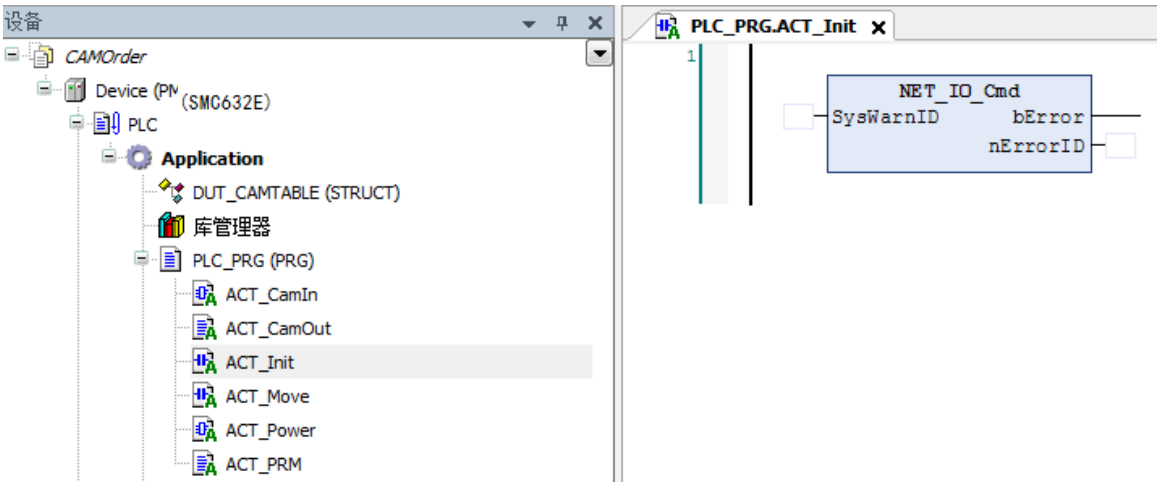


图 6.26　脉冲模块

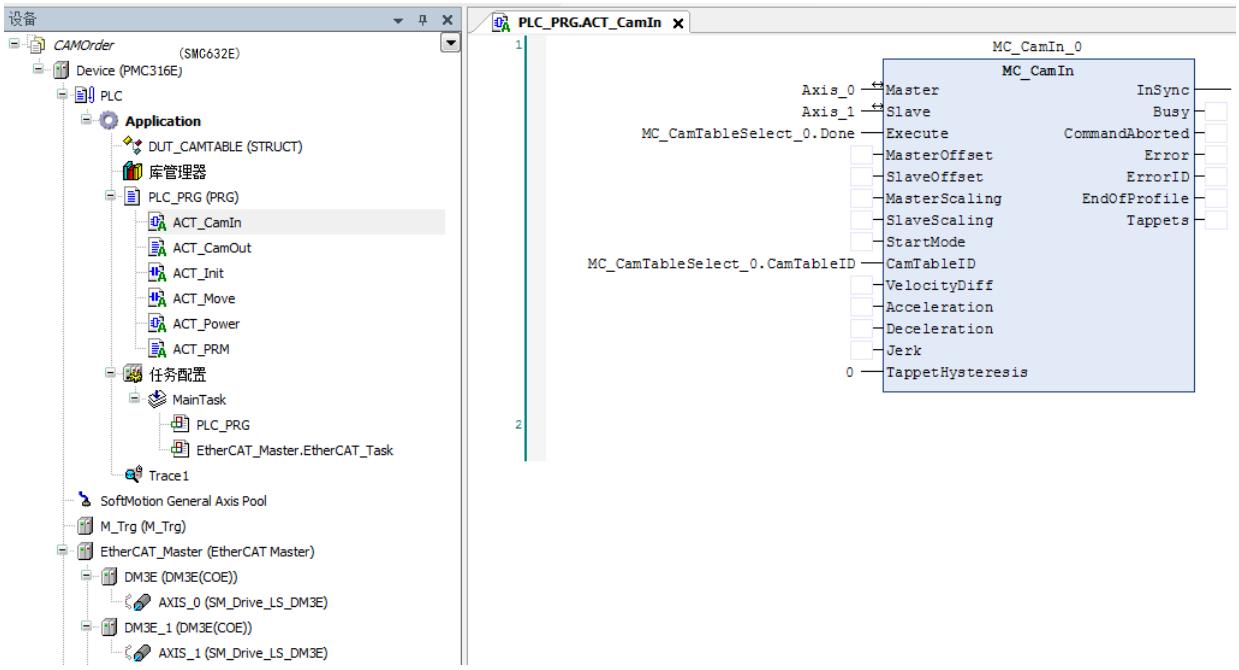


图 6.27　配置电子凸轮模块

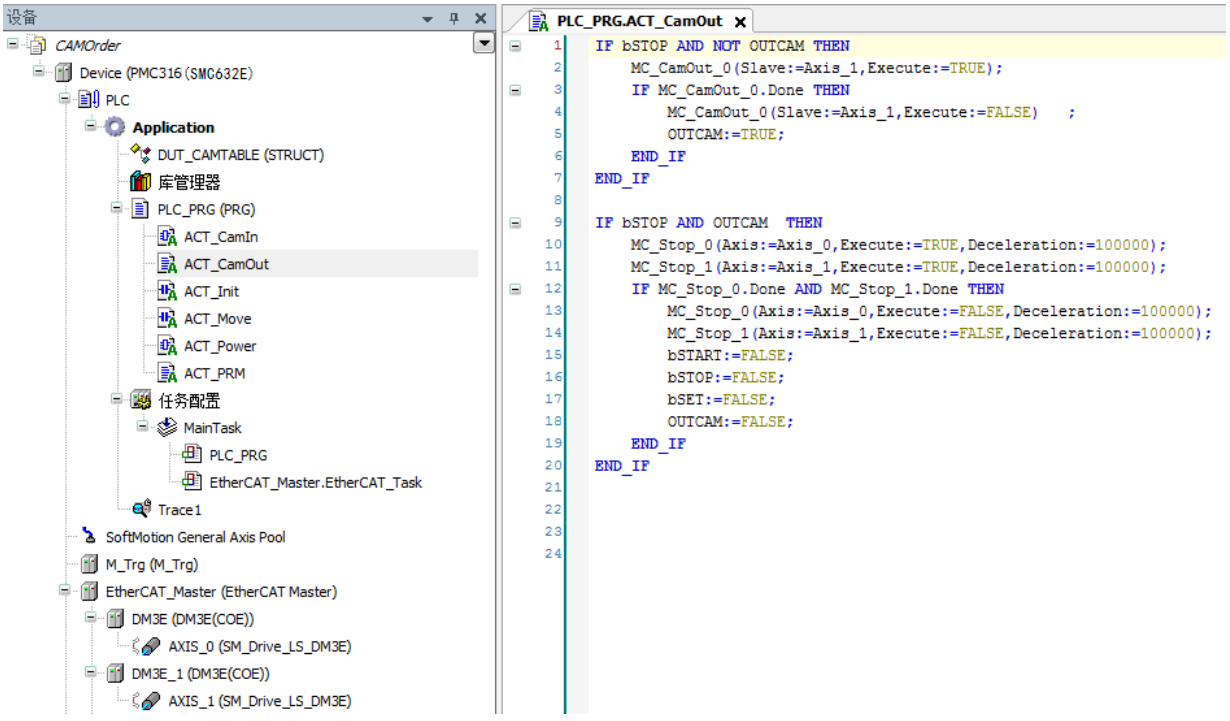


图 6.28　断开电子凸轮

4) 程序完成编程后，编译、下载到控制器中执行，并将 CASE 流程变量 i 的值强制为 1 启动整个电子凸轮运动，程序运行结果如图 6.29 所示。

1） 蓝色为主轴位置，绿色为从轴位置，红色为主轴速度，灰色为从轴速度。

2） 从轴在主轴运动到 5000Pulse 位置时也运动到 5000Pulse，由于设置的主从比为 1，从轴同步的速度和主轴一样均为 10000；

3） 主轴 5000Pulse 运动到 10000Pulse 位置，从轴 5000Pulse 运动到 7000Pulse，主轴速度不变，从轴速度降低为 4000。

4） 完成后主从轴脱离同步。

5） 如果设置的主从比为其他值，则从轴对应的运动参数则是凸轮表中从轴的数据和主从比的乘积。

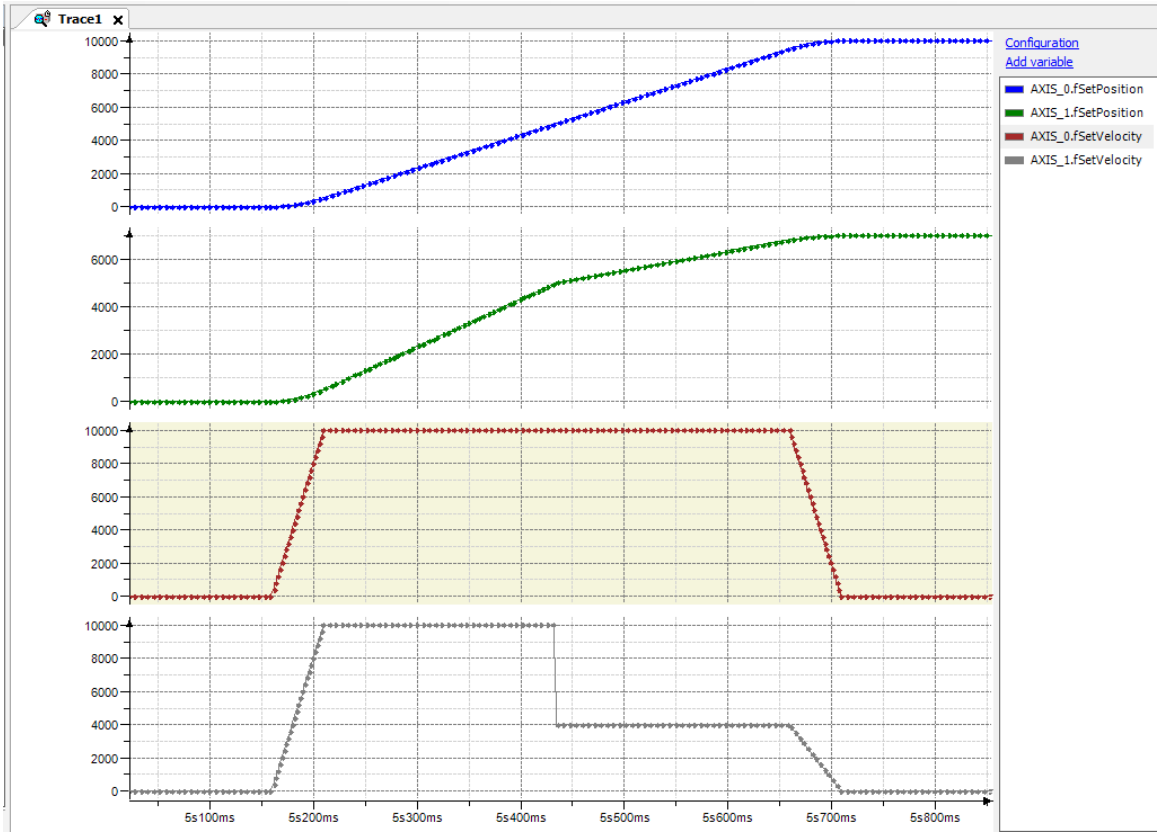


图 6.29　电子凸轮运动主从轴位置速度曲线

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**同步运动-电子凸轮-CAMOrder**”。

6.3　跟随运动指令

PMC600 支持雷赛专用的两轴相对坐标和绝对坐标跟随运动，主轴做点位运动，另一轴跟随运动；从轴和主轴同时启停，速度曲线类似，根据从轴与主轴的位置来分配从轴的速度。

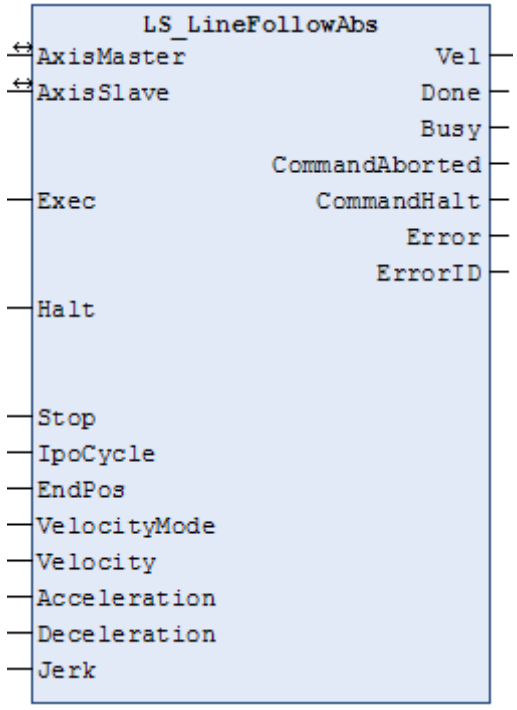
表 6.4　跟随运动指令

指令名	功能说明
LS_LineFollowAbs	一轴点位，另一轴跟随绝对位置运动
LS_LineFollowRel	一轴点位，另一轴跟随相对位置运动

绝对位置跟随　LS_LineFollowAbs

一轴点位，另一轴跟随运动指令，绝对位置模式。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_LineFollowAbs	FB		LS_LineFollowAbs(AxisMaster：= （参数）， AxisSlave：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， EndPos：= （参数）， VelocityMode：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_INPUTP_OUT	名称	类型	有效范围	初始化	注释
AxisMaster	主轴	AXIS_REF_VIRTUAL_SM3	-	-	主轴
AxisSlave	从轴	AXIS_REF_VIRTUAL_SM3	-	-	跟随轴
VAR_INPUT					
Exec	启动	BOOL	TRUE FALSE	FALSE	启动
Halt	暂停	BOOL	TRUE FALSE	FALSE	暂停
Stop	停止	BOOL	TRUE FALSE	FALSE	停止
IpoCycle	周期	DWORD	-	2000	周期，单位：us
EndPos	目标位置	ARRAY [0..1] OF LREAL	负数，0 正数	0	轴的目标位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）

Velocity	速度	LREAL	负数，0 正数	10	主轴速度 Pulse/s
Acceleration	加速度	LREAL	0，正数	10	主轴加速度 Pulse/s2
Deceleration	减速度	LREAL	0，正数	10	主轴减速度 Pulse/s2
Jerk	加加速度	LREAL	0，正数	90000000	加加速度 Pulse/s3，速度模式设置为 3 时需要设置该参数，该参数值不能为 0
VAR_OUTPUT					
Vel	速度值	LREAL	0，正数	0	主轴当前速度
Done	执行完成	BOOL	TRUE FALSE	FALSE	运动结束
Busy	执行中	BOOL	TRUE FALSE	FALSE	运动进行中
CommandAborted	指令中断	BOOL	TRUE FALSE	FALSE	运动终止
CommandHalt	指令暂停	BOOL	TRUE FALSE	FALSE	运动暂停
Error	错误	BOOL	TRUE FALSE	FALSE	运动出现错误
ErrorID	错误代码	SMC_ERROR	-	0	运动错误码

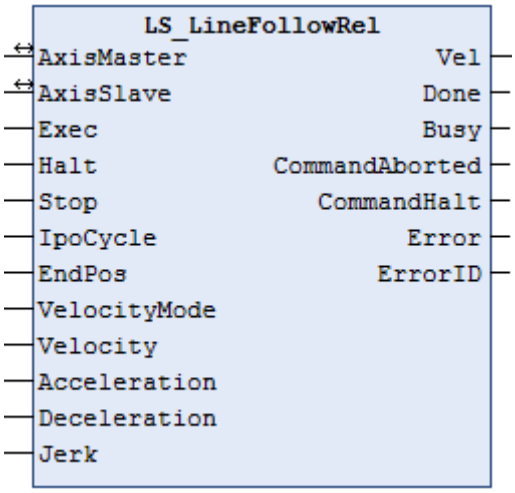
功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- 从轴跟随主轴做绝对位置模式的跟随运动。
- 注意，IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现 Error 报错。
- 参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。
- 跟随运动指令仅在指令启动到结束这段时间有效，对主从轴有跟随控制效果，到达指令位置后，主从轴不再同步，都恢复 standstill 状态。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。

相对位置跟随　LS_LineFollowRel

一轴点位，另一轴跟随运动指令，相对位置模式。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_LineFollowRel	FB		LS_LineFollowRel(AxisMaster：= （参数）， AxisSlave：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， EndPos：= （参数）， VelocityMode：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_INPUTP_OUT	名称	类型	有效范围	初始化	注释
AxisMaster	主轴	AXIS_REF_VIRTUAL_SM3	-	-	主轴
AxisSlave	从轴	AXIS_REF_VIRTUAL_SM3	-	-	跟随轴
VAR_INPUT					
Exec	启动	BOOL	TRUE FALSE	FALSE	启动
Halt	暂停	BOOL	TRUE FALSE	FALSE	暂停
Stop	停止	BOOL	TRUE FALSE	FALSE	停止
IpoCycle	周期	DWORD	-	2000	周期，单位：us
EndPos	目标位置	ARRAY [0..1] OF LREAL	负数，0 正数		轴的目标位置，单位：pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	负数，0 正数	10	主轴速度 Pulse/s
Acceleration	加速度	LREAL	0，正数	10	主轴加速度 Pulse/s2
Deceleration	减速度	LREAL	0，正数	10	主轴减速度 Pulse/s2

Jerk	加加速度	LREAL	0，正数	90000000	加加速度 Pulse/s3，速度模式设置为 3 时需要设置该参数，该参数值不能为 0
VAR_OUTPUT					
Vel	速度值	LREAL	0，正数	0	主轴当前速度
Done	执行完成	BOOL	TRUE FALSE	FALSE	运动结束
Busy	执行中	BOOL	TRUE FALSE	FALSE	运动进行中
CommandAborted	指令中断	BOOL	TRUE FALSE	FALSE	运动终止
CommandHalt	指令暂停	BOOL	TRUE FALSE	FALSE	运动暂停
Error	错误	BOOL	TRUE FALSE	FALSE	运动出现错误
ErrorID	错误代码	SMC_ERROR	-	0	运动错误码

功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- 从轴跟随主轴做相对位置模式的跟随运动。
- 注意，IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现 Error 报错。
- 参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。
- 跟随运动指令仅在指令启动到结束这段时间有效，对主从轴有跟随控制效果，到达指令位置后，主从轴不再同步，都恢复 standstill 状态。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。

跟随运动使用说明及例程：

PMC600 支持两轴相对坐标和绝对坐标跟随运动，主轴做点位运动，另一轴跟随运动；从轴和主轴同时启停，速度曲线类似，根据从轴与主轴的位置来分配从轴的速度。

该指令在程序中可直接调用执行，其中指令中的参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。

例程 6.7：编写程序，实现 Axis_0 做速度为 2000Pulse/s、加减速度为 4000Pulse/s^2、运动距离为 4000Pulse 的点位运动，从轴 Axis_1 跟随运动，运动距离为 2000Pulse。

跟随运动和直线插补运动指令用法类似，程序在例程 6.1 的基础上修改，将运动模块换成两轴相对坐标跟随运动 ACT_Move，主程序再稍作修改，设置主从轴的运动距离，并实时读取主轴的速度，程序如图 6.30、6.31 所示。

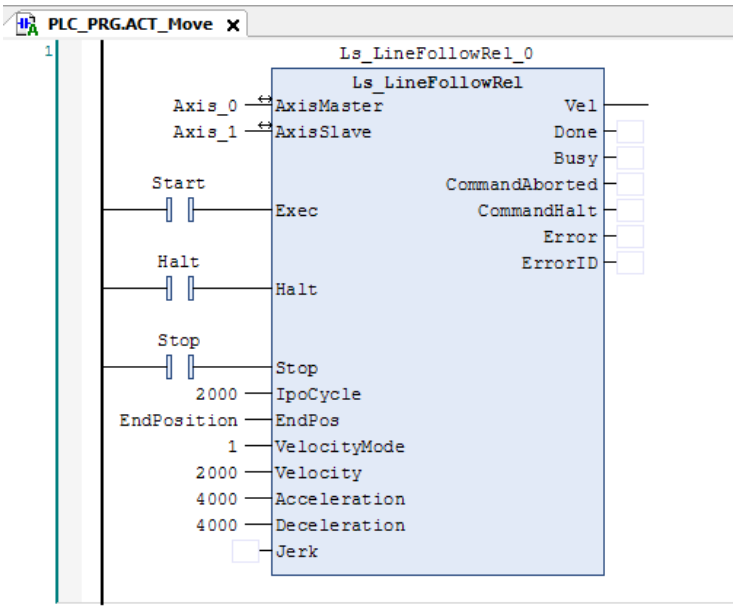


图 6.30　跟随运动模块

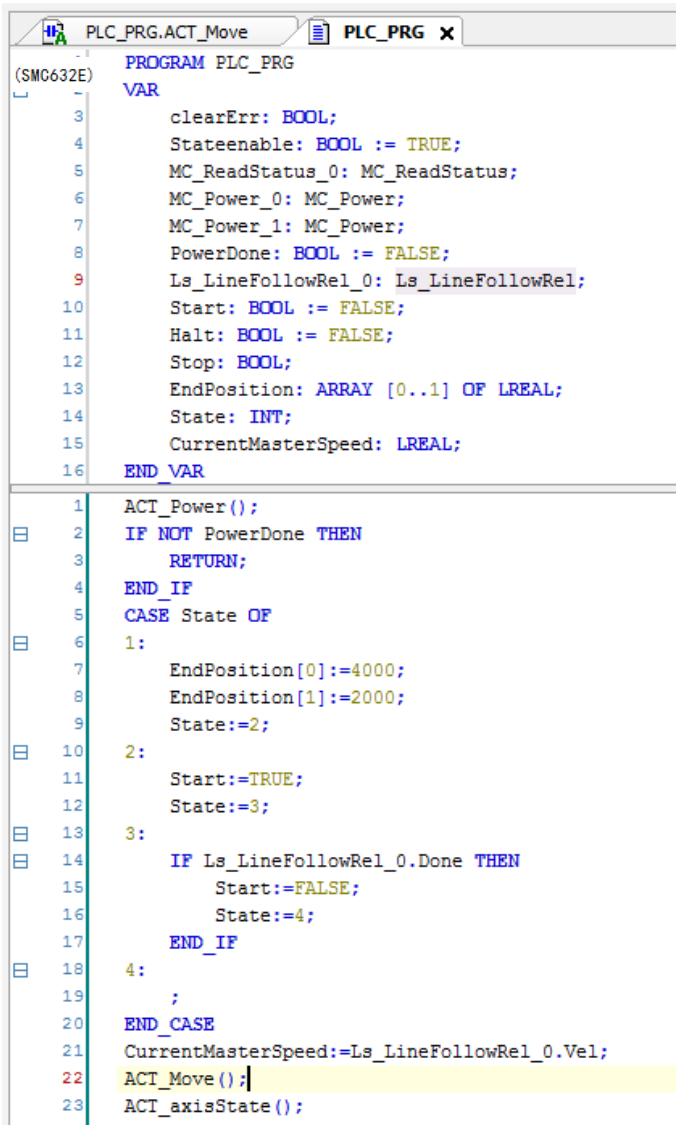


图 6.31　跟随运动主程序

将程序编译下载到控制器中执行，并设置 State 强制为 TRUE，启动运动，运行结果如图 6.31 所示。启动跟随运动后，主轴 Axis_0 轴做点位运动，从轴 Axis_1 轴按照设置的位移与主轴的位移比来匹配主轴的速度，主从轴同时启停，速度曲线模式都是设置的 S 型。

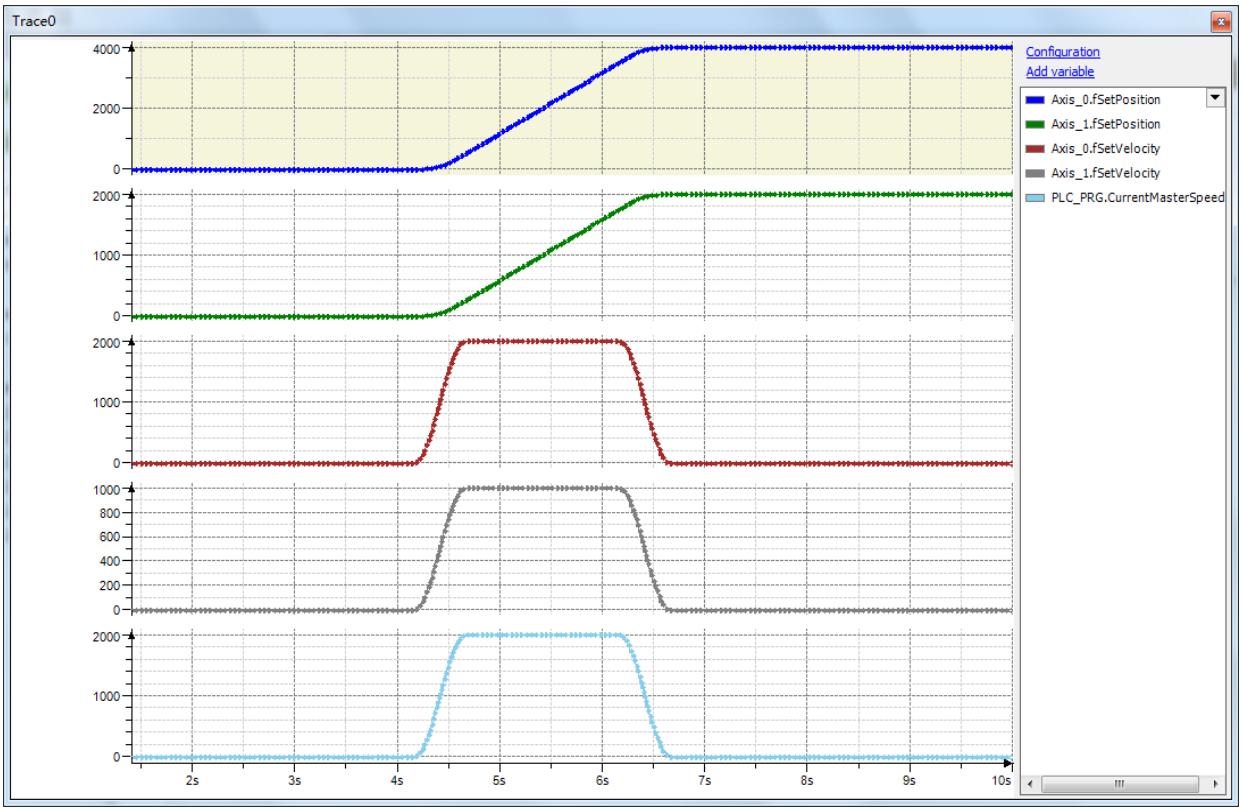


图 6.32　位置速度曲线

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**插补运动-跟随运动-2Axis_FollowMoveRel**”。

第 7 章　轴组指令

本节的轴组指令遵循国际通用的 IEC61131-3 PART4 标准，并且所有轴组指令包含在 SM3_Robotics 运动库之中，要使用轴组功能，必须先在工程中添加 SM3_Robotics 运动库。

轴组是由多个运动轴所组成的轴的集合，轴组能够被轴组相关指令所调用，以实现多轴的单段插补、多轴的连续插补功能。当轴组与机器人模型配套使用的时候，能够实现对机械手的控制功能。

本节主要分为轴组指令介绍和轴组功能应用两部分，其中轴组指令包括了基础轴组指令、轴组状态监控指令、坐标系指令、动力学指令、运动学指令和轴组运动指令等 6 部分，讲解了各个轴组指令的使用说明。轴组功能应用包括了连续插补运动、SCARA 机器人、Delta 机器人、六自由度关节机器人和 5 轴运动建模等。

在一些对运动效果要求比较高的场合，比如点胶设备，需要指令之间的速度连续，则可以选用连续插补运动功能。如果机械结构本身是基于机器人运动模型时，则选用相应的机器人模型进行运动控制。

7.1 轴组参数说明

以下将对轴组功能中重要的一些数据类型做些说明，以便在后续使用过程中，用户能够更快更好地编程。

Blending

轴组运动的拐角过渡功能。

通过配置功能块 MC_MoveDirectAbsolute，MC_MoveDirectRelative，MC_MoveLinearAbsolute，MC_MoveLinearRelative，MC_MoveCircularAbsolute 和 MC_MoveCircularRelative 中的 TransitionMode 和 BufferMode 参数，进行参数的组合使用，从而达到不同的拐角过渡效果。

MC_BUFFER_MODE（ENUM）

轴组运动速度交接模式。

对于同一个轴组，当轴组在运动过程之中，可以调用其它轴组运动指令，改变轴组的运动状态。当轴组在前后两个运动指令之间进行切换的时候，通过设置新调用指令的 BufferMode 参数，可以选择 6 种速度交接模式，表 7.1 列出了每种模式的说明。

表 7.1　速度交接模式及说明

交接模式	说明
0：mcAborting（打断）	立即中断当前的运动并执行新的运动指令
1：mcBuffered（等待）	等待当前运动正常执行结束后，立即执行下一个指令的运动
2：mcBlendingLow（低速滤波处理）	等待当前运动到目标位置后，以两个指令之中较低的速度进行动作的切换
3：mcBlendingPrevious（当前速度滤波处理）	等待当前运动到目标位置后，以当前运动指令的速度进行动作的切换
4：mcBlendingNext（下个速度滤波处理）	等待当前运动到目标位置后，以下一个运动指令的速度进行动作的切换
5：mcBlendingHigh（高速滤波处理）	等待当前运动到目标位置后，以两个指令之中较高的速度进行动作的切换

MC_TRANSITION_MODE (ENUM)

拐角过渡模式，可以选择 3 种模式，参数说明分别如下：

0：TMNone

不设置拐角过渡，Blending 模式不支持配置这个参数

1：TMStartVelocity

基于速度的拐角过渡模式

1）基于速度参数的拐角模式，拐角路径由图 7.1 中的“A”点和“B”点组成。

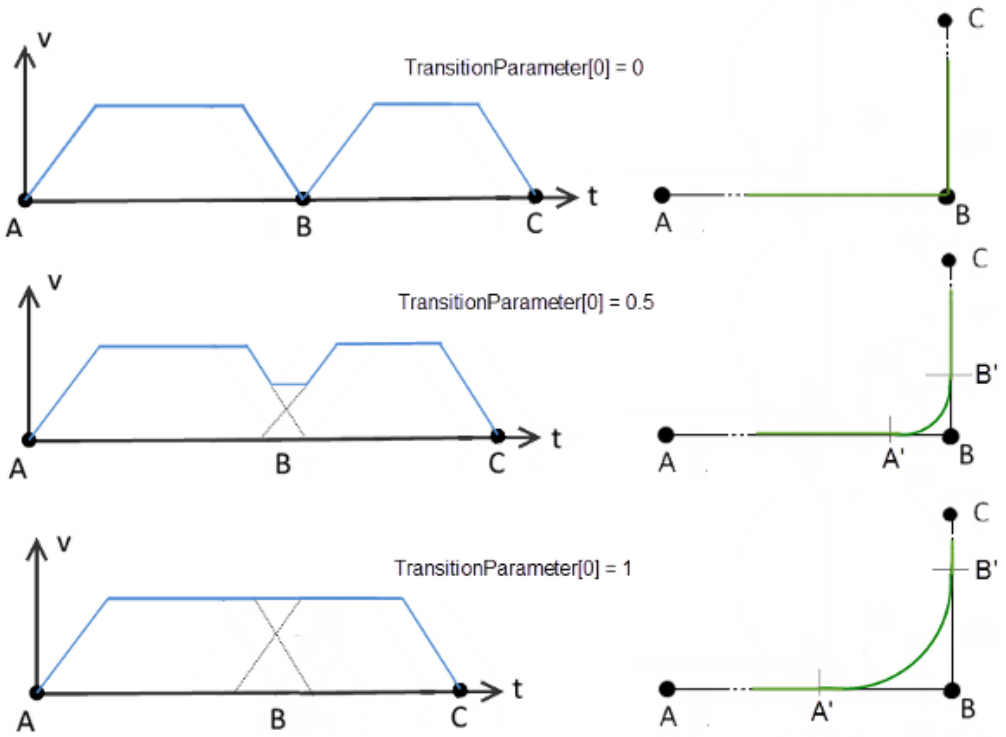


图 7.1　拐角过渡模式

2）在 PTP 运动模式下，运动拐角过渡点是根据先前计算的两个运动的速度曲线来确定的。在插补运动中有多个轴，如图 7.2 所示，点 A’由第一个轴减速到拐角点 B 的时间决定。点 B’是最后一个轴从拐角点 B 开始到其加速运动完成的时间来决定的。

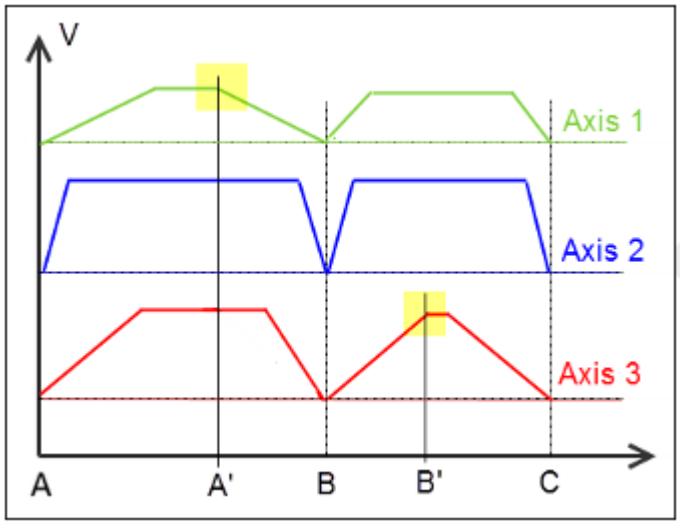


图 7.2　拐角过渡模式

3）TransitionParameter[0]是路径拐点的持续时间的一个因子。数值为 1 表示路径在点 A’和 B’处进行过渡，数值<1 将减小过渡区域，值 0 对应于 TMNone。也可以使用大于 1 的值，过渡区域将会相应的增加。

4）在 CP 模式下，过渡点的计算基于一个理想的速度曲线，该曲线可能偏离实际速度曲线。其在原始路径上模拟减速斜坡，该路径朝向过渡点和加速斜坡，不经过实际的拐点。由编程路径速度和由轴极限得到的估计最大路径速度的最小值作为目标速度。

另外，当直线之间进行过渡的时候，要考虑直线之间的夹角。过渡元素的最小曲率半径是由期望的路径速度和估计的动力学极限产生的。点 A’和 B’从该半径和直线之间的角度依次产生，如图 7.3 所示。

5）TransitionParameter[0]在这里作为一个关键因子，但不是在时间维度上，而是与路径长度有关。值 1 表示减速斜坡的开始和加速斜坡的结束。值 0.5 表示正好介于两者之间。 TransitionParameter[0]在考虑了所有约束条件（如一半元素长度、前瞻规划和切点距离之间的最大比率）后起作用。因此，小于 1 的值会导致距离 A’B 和 BB’减小。由于上面所描述的限制，大于 1 的值不一定会规划出更长的距离。

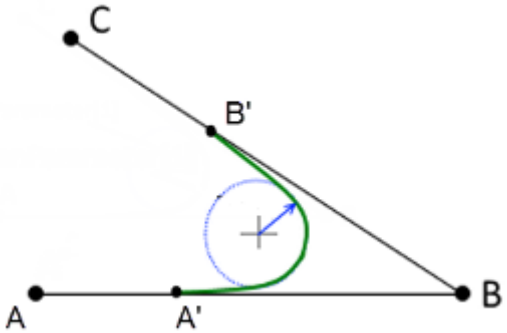


图 7.3　拐角过渡模式

2：TMCornerDistance

基于距离的拐角过渡模式

1）基于位置的拐角规划模式。在这个模式下，TransitionParameter[0]是实际交汇点的半径（过渡前的轨迹终点和过度后的轨迹起点），如图 7.4 所示。

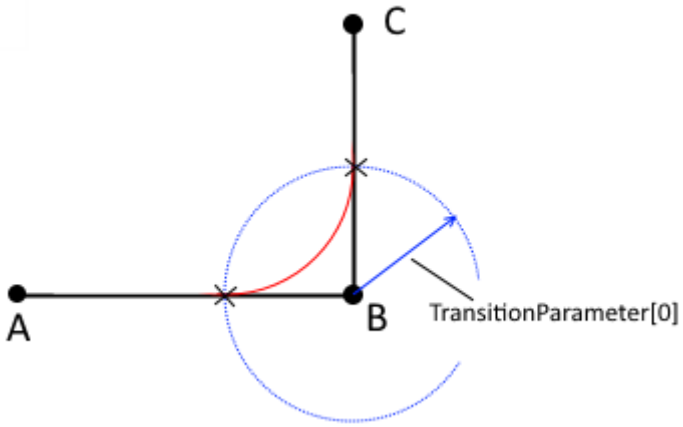


图 7.4　拐角过渡模式

2）过渡时，在点 A 和点 B 之间，与点 B 和点 C 之间开始规划出相切的路径。CP 模式在缓存模式（非 Aborting）下的运动：比较长的相切段为较短相切段长度的 5 倍，如果相切段的长度相差太大，有可能会产生很高的曲率，从而导致拐角半径较小。

3）TransitionParameter[1]仅在 CP 模式下有效，用来作为圆弧过渡的最小过渡圆弧半径，如图 7.5 所示。另外，指令本身也会计算最小的过渡圆弧半径。规划的轨迹，其实际过渡圆弧半径不得小于上面的两个数值，否则圆弧过渡将无法执行，运动将无法执行，导致指令中断。

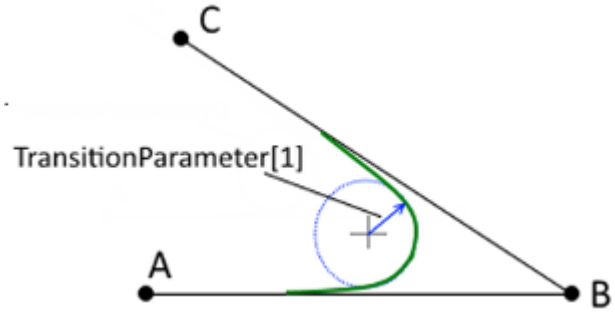


图 7.5　拐角过渡模式

备注：如果两个元素具有相同的切线和曲率，则他们之间不需要设置过渡参数。

例如两个半圆运动的情况，且这两个半圆运动是由 MC_MoveCircularAbsolute 控制的，以获得一个完整的圆。另一个例子是单位向量相同的两条直线连接在一起。

SMC_COORD_SYSTEM (ENUM)

用来表示轴组位置关系的坐标系，具体的坐标系名称以及说明见表 7.2。

表 7.2　坐标系名称及说明

名称	说明
ACS	轴坐标系：不属于笛卡尔坐标系，坐标系的维度与轴的数量相等，每个坐标对应轴组中的一个轴
MCS	机器坐标系，坐标系与轴组所对应的运动学模型相关，通过运动学变换定义了 MCS 的位置和方向。
WCS	世界坐标系，是所有轴组的通用坐标系。
PCS_1	产品坐标系 1，依据产品来确定，可能是静态的，也可能是动态的。
PCS_2	产品坐标系 2，依据产品来确定，可能是静态的，也可能是动态的。
TCS	工具坐标系，当使用了 MoveDirect-， MoveLinear- 或 MoveCircular-等 FB 模块时，位置数据与轴组在开始移动后的位置和方向都有关系，因此没有绝对位置和相对位置的区别。当调用的是其它的指令时（如 SMC_GroupConvertPosition），TCS 表示轴组在当前循环周期启动后的位置。

备注：轴组功能的坐标系数量较多，在使用过程中，需要特别注意，设置坐标系之间正确的位置转换关系。

SMC_AXIS_GROUP_STATE (ENUM)

轴组状态类型，遵循了 PLCOpen Part4 的标准，如图 7.6 所示，共 6 种状态。

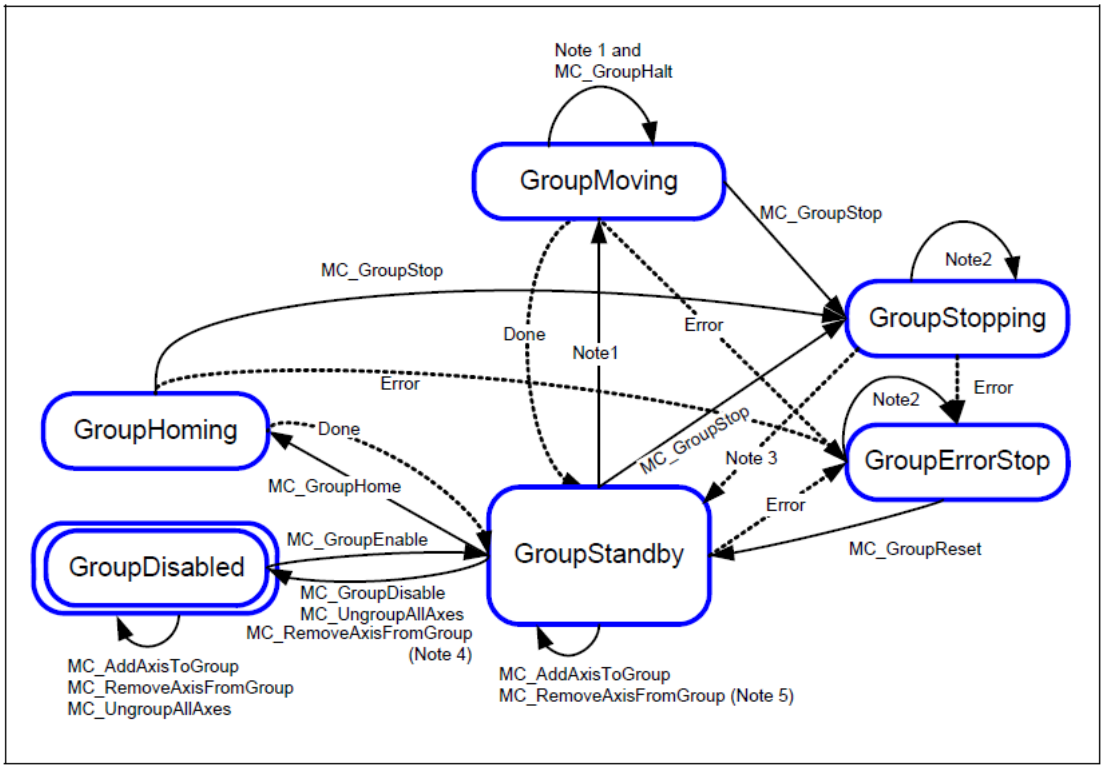


图 7.6　PLCOpen Part4 标准

- Disabled（轴组无效）
- Standby（准备好）
- Moving（运动中）
- Homing（回零中）
- Stopping（停止中）
- ErrorStop（报错停止）

SMC_POS_REF（UNION）

表示 TCP（工件坐标系）的位置。

位置可以用笛卡尔坐标（X，Y，Z，A，B，C）或轴坐标（A0...A5）进行表示，如表 7.3 所示。

表 7.3　TCP 的位置

名称	类型	注释
a	TRAFO.AXISPOS_REF	轴坐标系
c	MC_COORD_REF	笛卡尔位置
v	ARRAY [0..(SMC_RCNST.MAX_AXES - 1)] OF LREAL	数组数值，根据最终使用的坐标系来确定

SMC_PTP_MOVEMENT_TYPE（ENUM）

参数主要用来设置 MC_MoveDirectAbsolute 和 MC_MoveDirectRelative 指令的轨迹运行方式。具体的参数说明，见表 7.4。

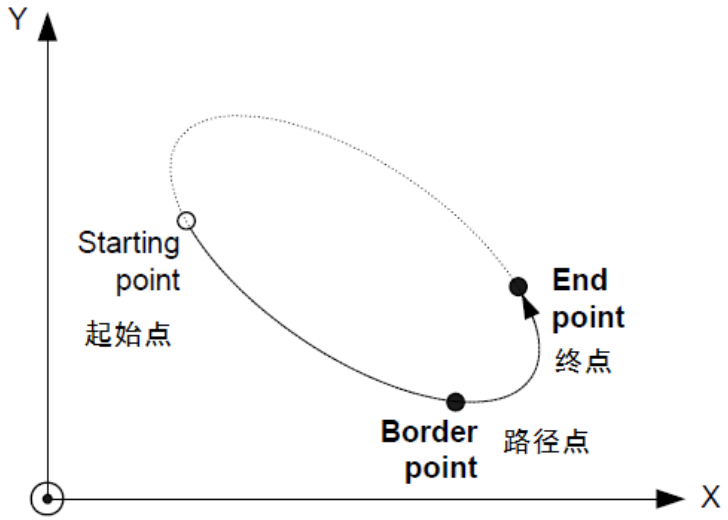
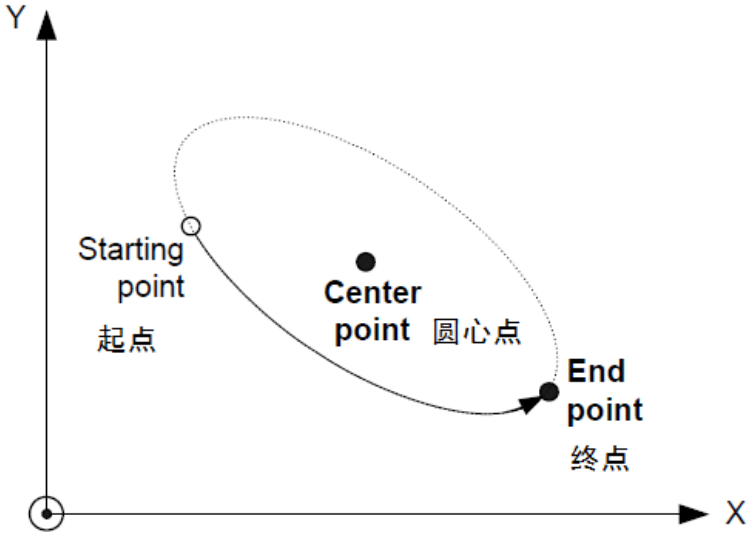
表 7.4　设置轨迹运行方式

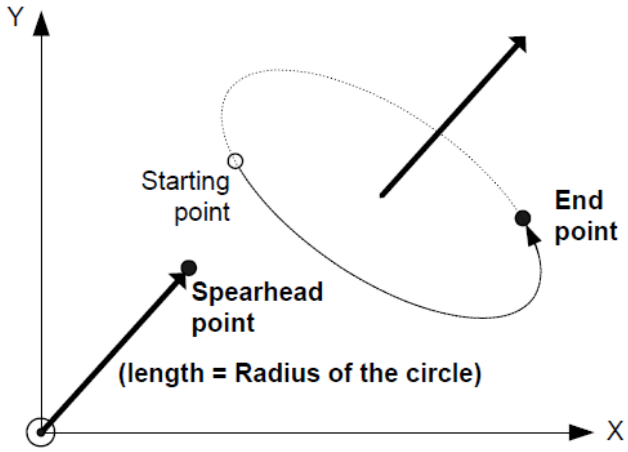
名称	注释
Fast（最快）	基于时间最优，去进行规划的 PTP 运动
Path_Invariant（轨迹确定）	在这种模式下，运动轨迹有以下特点： 1.会在规划的轨迹内运行（除了在 TransitionMode TMStartVelocity 模式下），且不受轴的运动参数限制（速度、加速度、减速度和加加速度的辅助或全局限制）； 2.在 TransitionMode TMCornerDistance 模式下不受 blending(BlendingHigh / Low / Previous / Next)的影响； 3.暂停或停止后（MC_GroupHalt / MC_GroupStop），不会离开既定轨迹； 4.中断和继续后（MC_GroupInterrupt / MC_GroupContinue），不会离开既定轨迹。

SMC_CIRC_MODE (ENUM)

圆弧插补模式，如表 7.5 所示，定义使用何种方式进行画圆弧插补，用于 MC_MoveCircularAbsolute 和 MC_MoveCircularRelative 指令中。

表 7.5　圆弧插补模式

名称	说明
BORDER	三点圆弧：用户输入圆弧轨迹能经过的三点坐标去规划圆弧的起点、终点和路径点（辅助点）。 （Border 模式不考虑输入路径选择） 三个输入点一定是在同一个平面上，且三点不共一条直线。 这种方式的优点：输入点通常是圆弧轨迹能够达到的点，方便采集。 缺点：三点之间的角度一定是小于 2π。 
CENTER	中心圆弧：需要输入圆弧的起点、终点和圆心点等参数，从而确定具体的圆弧轨迹。 

名称	说明
RADIUS	终点半径模式：需要输入起点、终点和圆弧半径。 

SMC_ORIENTATION_MODE（ENUM）

方向模式。用来设置在 CP 模式下，插补运动的方向，详见表 7.6。

表 7.6　CP 模式下设置插补运动的方向

交接模式	说明
GreatCircle	沿着最短的路径从起始位置向目标位置运动，在这种模式下，即使起始位置和终点位置都在指定区域内，但实现的路径也可能会离开这个区域。
Axis	定位轴从起始位置到终点位置都在指定的区域内运动，不是所有的运动学变换都支持该种模式。

7.2　轴组指令说明

7.2.1　基础轴组指令

表 7.7　基础轴组指令

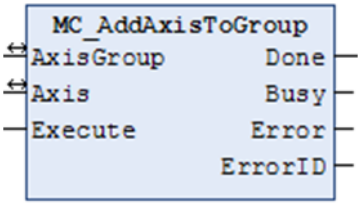
指令名	功能说明
MC_AddAxisToGroup	将单独的轴添加到轴组之中
MC_RemoveAxisFromGroup	将轴从轴组中移除
MC_GroupEnable	用于启用指定的轴组
MC_GroupDisable	指定的轴组变成无效状态，让其它轴组指令无法调用该轴组进行运动控制
MC_GroupReset	用于解除轴组及轴的异常状态
MC_GroupSetPosition	设置轴组中各轴的指令位置
MC_UngroupAllAxes	这个指令的作用是将某个轴组所包含的轴全部移除，解散该轴组

SMC_GroupPower	使能轴组下的所有轴，等价于轴组下的所有轴调用 MC_Power 指令
SMC_GroupSaveContinueData	保存轴组运动中暂停后的位置数据信息

添加轴到轴组　MC_AddAxisToGroup

使用指令方式将某个轴添加到轴组之中。

指令外观：

指令	FB/ FUN	图形模块	结构文本
MC_AddAxisToGroup	FB		MC_AddAxisToGroup(AxisGroup:= （参数）， Axis:= （参数）， Execute:= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
Axis	轴	AXIS_REF_SM3	-	-	轴号
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	报错	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

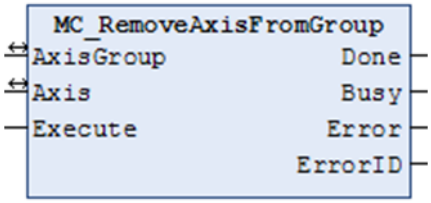
功能说明：

- 指令用于往指定的轴组之中添加轴，以轴组的关系绑定起来。
- 当指令的 Done 变量变成 TRUE 时，表示该轴成功添加到轴组之中。请注意，将 Execute 设置为 FALSE 并不能将轴从轴组中删除，如果需要将该轴从轴组中删除，需要使用 MC_RemoveAxisFromGroup 指令。
- 只有在轴组处于 GroupDisabled 的状态，才能够执行这条指令，如果在轴组使能之后再执行这条指令则会报错。

从轴组中移除轴 MC_RemoveAxisFromGroup

这条指令的作用是将轴从轴组中移除。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_RemoveAxisFromGroup	FB		MC_RemoveAxisFromGroup(AxisGroup：= （参数）， Axis：= （参数）， Execute：= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
Axis	轴	AXIS_REF_SM3	-	-	轴号
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	报错	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

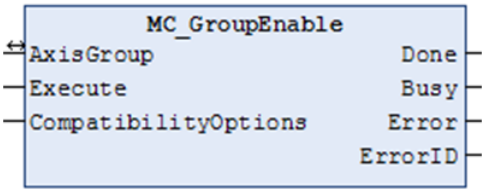
功能说明：

- 只有在轴组未使能的状态，才能够执行这条指令，如果在轴组使能之后再执行这条指令则会报错。
- 当指令的 Done 变量变成 TRUE 时，表示该轴成功地从轴组之中移除。

启用轴组 MC_GroupEnable

启用指定的轴组，调用运动控制指令操作轴组之前必须调用该指令将轴组状态切换到 Standby。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupEnable	FB		MC_GroupEnable(AxisGroup：= （参数）， Execute：= （参数）， CompatibilityOptions：= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	-	FALSE	执行当前指令
CompatibilityOptions	兼容性选项	SMC_AXIS_GROUP_COMPATIBILITY_OPTIONS	-	-	为了兼容以前的版本而存在的参数
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	报错	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

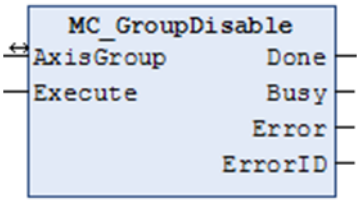
功能说明：

- 在调用轴组的运动控制等指令之前，需要先调用 SMC_GroupPower 或者 MC_Power 使能轴组。
- 当指令的 Done 变量变成 TRUE 时，表示该轴组成功切换成为 Standby 状态。
- 可指定到轴组中的轴的种类只能是“伺服轴”和“虚拟伺服轴”，若指定了其他轴种类，将出现异常。
- 在执行该指令时，该轴组下的所有轴必须处于停止状态。
- 如果存在已经属于其他轴组、且该轴组已启用的轴，则不能执行 MC_GroupEnable(启用轴组)指令，会发生异常报错。
- 使轴组无效的条件有：执行 MC_GroupDisable(不启用轴组)指令、切换到程序模式使运行停止、以及开始 MC 试运行。

使轴组无效 MC_GroupDisable

轴组切换为 Disable 状态，该状态下不可以进行轴组的运动控制。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupDisable	FB		MC_GroupDisable(AxisGroup：= （参数）， Execute：= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	报错	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

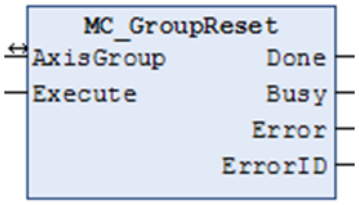
功能说明：

- 该指令将指定的轴组切换到 GroupDisable 状态。
- 轴组状态变为 GroupDisable(不启用轴组)时，指定的 AxesGroup(轴组)的缓存指令会被清除。

轴组复位 MC_GroupReset

解除轴组及轴的异常状态。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupReset	FB		MC_GroupReset(AxisGroup:= （参数）， Execute:= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	报错	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

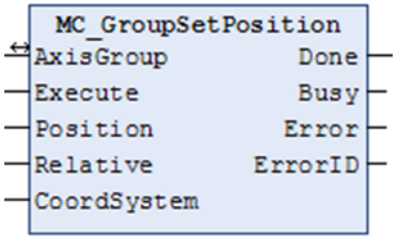
功能说明：

- 在 Execute(启动)的上升沿，对 GroupEnable(启用轴组)状态的 AxesGroup(轴组)指定的轴组的异常及轴组所属轴的异常进行解除处理。可进行解除的异常有：轴及轴组发生的 “轻度故障”、“监视信息”的异常、以及驱动器错误复位。
- 无论轴是伺服 ON 状态还是伺服 OFF 状态，均可执行异常解除处理。
- 对于发生驱动器错误的轴，应先执行驱动器错误复位处理，然后再执行异常解除处理。
- 驱动器错误复位处理可选择清除驱动器错误，还是在轴参数[驱动器错误复位监视时间]内保持不变。驱动器错误复位对属于轴组的所有轴同时进行。
- 可解除的异常对象为 Execute(启动)上升沿时发生的异常。不能对在异常解除过程中发生的异常执行异常解除。
- 若在轴组的错误减速停止中执行指令，则该指令无法执行，这是因为在轴停止之前无法进行异常解除。此外，轴组中的轴本身发生异常报错，也无法通过本指令解除异常。

设置轴组指令位置 MC_GroupSetPosition

用于设置轴组中各轴的指令位置。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupSetPosition	FB		MC_GroupSetPosition(AxisGroup:= （参数）， Execute:= （参数）， Position:= （参数）， Relative:= （参数）， CoordSystem:= （参数）， Done=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
Position	位置	SMC_POS_REF	负数，0 正数	0	轴的目标位置
Relative	位置模式	BOOL	TRUE FALSE	FALSE	相对位置模式= True，绝对位置模式= False (默认)
CoordSystem	应用坐标系	SMC_COORD_SYSTEM	-	-	应用坐标系
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	报错	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

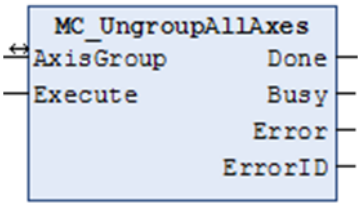
功能说明：

- 设置轴组中指定坐标系下的坐标位置；
- 指令在轴组处于 GroupStandby 状态下执行，不能在动态坐标系下执行，也不能与 MC_GroupContinue 指令同时执行。

解除轴组 MC_UngroupAllAxes

将某个轴组所包含的轴全部移除，解散该轴组。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_UngroupAllAxes	FB		MC_UngroupAllAxes(AxisGroup:= （参数）, Execute:= （参数）, Done=> （参数）, Busy=> （参数）, Error=> （参数）, ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	报错	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

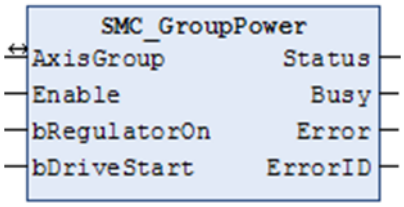
功能说明：

指令需要在轴组处于 Disable 状态下使用。

轴组使能 SMC_GroupPower

使能轴组下的所有轴，等价于轴组下的所有轴调用 MC_Power 指令。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupPower	FB		SMC_GroupPower(AxisGroup:= （参数）， Enable:= （参数）， bRegulatorOn:= （参数）， bDriveStart:= （参数）， Status=> （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Enable	有效	BOOL	TRUE FALSE	FALSE	输入 TRUE 让模块开始运行
bRegulatorOn	使能	BOOL	TRUE FALSE	FALSE	设置 TRUE 让轴组进入使能状态
bDriveStart	驱动启用	BOOL	TRUE FALSE	FALSE	设置为 TRUE 让功能块关闭紧急停止处理
VAR_OUTPUT					
Status	完成	BOOL	TRUE FALSE	FALSE	输出 TRUE 则轴组能够开始运动
Busy	执行中	BOOL	TRUE FALSE	FALSE	当模块被调用时为 TRUE
Error	报错	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

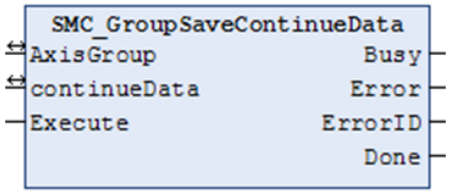
功能说明：

- 指令用于实现轴组中所有的轴切换为使能状态。
- 该指令不能切换轴组的状态机，如：当轴组处于 GroupDisabled（轴组禁用）时，在调用 SMC_GroupPower 之后，轴组的状态依然是 GroupDisabled。

保存轴组继续运动数据 SMC_GroupSaveContinueData

保存轴组运动中暂停后的位置数据信息。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupSaveContinueData	FB		SMC_GroupSaveContinueData(AxisGroup：= （参数）， continueData：= （参数）， Execute：= （参数）， Busy=> （参数）， Error=> （参数）， ErrorID=> （参数）， Done=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
continueData	暂停数据	SMC_AXIS_GROUP_CONTINUE_DATA	-	-	与 MC_GroupContinue 组合用来存储轴组运动暂停的位置数据，以便继续运动进行调用
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
VAR_OUTPUT					
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Error	报错	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
ErrorID	错误码	SMC_ERROR	-	0	功能块执行错误
Done	完成	BOOL	TRUE FALSE	FALSE	错误代码指示，见 SMC_Error

功能说明：

- 该指令保存轴组运动中暂停后的位置数据信息。注意：数据的保存需要多个周期才能完成，当 Done 信号为 True 后，可调用 MC_GroupContinue 指令来完成后续的运动。
- 当轴组的状态为 Disabled 或 ErrorStop 时，调用该指令会报错。
- 为了确保在被 MC_GroupHalt 或 MC_GroupStop（或其它可能导致运动中断的情况）暂停后，能够继续连续插补运动，需要确保以下两点：
  - SMC_GroupSaveContinueData 需要和 MC_GroupHalt 或 MC_GroupStop 在同一个任务周期内调用
  - SMC_GroupSaveContinueData 指令需要在 MC_GroupHalt 或 MC_GroupStop 之前被调用，否则继续运动位置可能会被清除。
- 注意，一个工程中只能实例化一个该指令，同一个工程中只能调用一次 SMC_GroupSaveContinueData 指令。

7.2.2 轴组状态监控指令

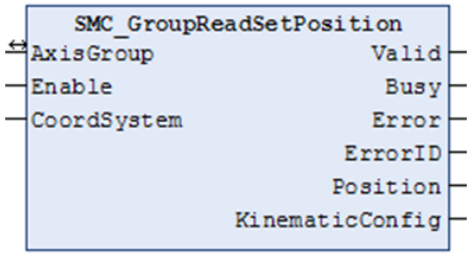
表 7.8 轴组状态监控指令表

指令名	功能说明
SMC_GroupReadSetPosition	读取轴组在指定坐标系下的当前指令位置
SMC_GroupReadSetVelocity	读取轴组在指定坐标系下的设置速度
SMC_GroupReadSetAcceleration	读取轴组在指定坐标系下的设置加速度
MC_GroupReadActualPosition	读取轴组在指定坐标系下的反馈位置
MC_GroupReadActualVelocity	读取轴组在指定坐标系下的反馈速度
MC_GroupReadActualAcceleration	读取轴组在指定坐标系下的反馈加速度
SMC_GroupTargetPosition	读取轴组在指定坐标系下的目标位置
SMC_GroupConvertPosition	轴组中在输入坐标系下的位置信息转换为输出坐标系下的位置
SMC_GroupGetContinuePosition	读取轴组运动中断时的位置
MC_GroupReadConfiguration	读取轴组包含的轴、数量等配置参数
MC_GroupReadStatus	用于获取轴组的当前运动状态
MC_GroupReadError	获取轴组的错误信息

读取轴组指令位置 SMC_GroupReadSetPosition

读取轴组在指定坐标系下的当前指令位置。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupReadSetPosition	FB		SMC_GroupReadSetPosition(AxisGroup:= （参数）, Enable:= （参数）, CoordSystem:= （参数）, Valid=> （参数）, Busy=> （参数）, Error=> （参数）, ErrorID=> （参数）, Position=> （参数）, KinematicConfig=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	为 True 时连续获取轴组在参考坐标系中的指令位置
CoordSystem	坐标系	SMC_COORD_SYSTEM	-	-	参考坐标系
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE FALSE	FALSE	当获取有效数值时为 True
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
Position	位置	SMC_POS_REF	负数，0，正数	0	轴组的当前指令位置
KinematicConfig	运动配置	TRAFO.CONFIGDATA	-	-	当参考坐标系为笛卡尔坐标系（非 ACS）时，运动学换算后的当前轴组设置位置

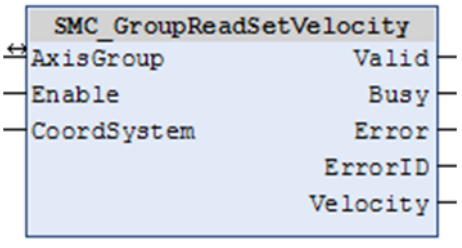
功能说明：

- 该指令用来读取各个轴在指定坐标系下的当前指令位置。
- 选择的坐标系，可以参考 7.1 节的“SMC_COORD_SYSTEM”说明。

读取轴组的设置速度 SMC_GroupReadSetVelocity

读取轴组在指定坐标系下的设置速度。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupReadSetVelocity	FB		SMC_GroupReadSetVelocity(AxisGroup: = , Enable: = , CoordSystem: = , Valid=> , Busy=> , Error=> , ErrorID=> , Velocity=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	为 True 时连续获取轴组在参考坐标系中的指令速度
CoordSystem	坐标系	SMC_COORD_SYSTEM	-	-	参考坐标系
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE FALSE	FALSE	当获取有效数值时为 True
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
Velocity	速度	SMC_POS_REF	-	-	轴组的当前指令速度。如果选择的是笛卡尔坐标 Velocity.c 包含笛卡尔速度：(X，Y，Z)是速度矢量，(A，B，C)分别是 x，y，z 轴周围的角速度。

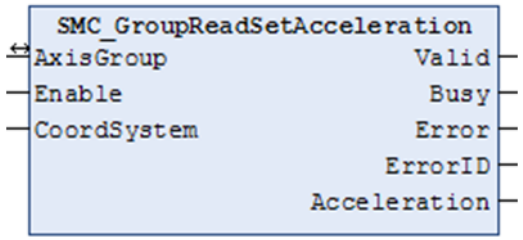
功能说明：

- 该指令读取各轴在指定坐标系下的当前指令速度。
- 选择的坐标系，可以参考 7.1 节的“SMC_COORD_SYSTEM”说明。

读取轴组的设置加速度 SMC_GroupReadSetAcceleration

读取轴组在指定坐标系下的设置加速度。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupReadSetAcceleration	FB		SMC_GroupReadSetAcceleration(AxisGroup: = , Enable: = , CoordSystem: = , Valid=> , Busy=> , Error=> , ErrorID=> , Acceleration=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	为 True 时连续获取轴组在参考坐标系中的指令加速度
CoordSystem	坐标系	SMC_COORD_SYSTEM	-	-	参考坐标系
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE FALSE	FALSE	当获取有效数值时为 True
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
Acceleration	加速度	SMC_POS_REF	-	-	轴组的当前指令加速度。如果选择的是笛卡尔坐标系，Velocity.c 包含笛卡尔加速度：(X，Y，Z)是加速度矢量，(A，B，C)分别是 x，y，z 轴周围的角加速度。

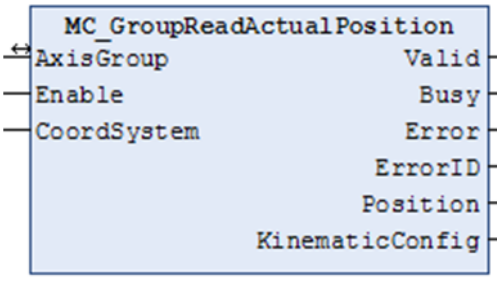
功能说明：

- 读取轴组在指定坐标系下的设置加速度。
- 选择的坐标系，可以参考 7.1 节的“SMC_COORD_SYSTEM”说明。

读取轴组反馈位置 MC_GroupReadActualPosition

读取轴组在指定坐标系下的反馈位置。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupReadActualPosition	FB		MC_GroupReadActualPosition(AxisGroup: = , Enable: = , CoordSystem: = , Valid=> , Busy=> , Error=> , ErrorID=> , Position=> , KinematicConfig=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	为 True 时连续获取轴组在参考坐标系中的实际位置
CoordSystem	坐标系	SMC_COORD_SYSTEM	-	-	指定的坐标系
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE FALSE	FALSE	
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
Position	位置	SMC_POS_REF	-	-	轴组的实际位置
KinematicConfig	运动配置	TRAFO.CONFIGDATA	-	-	当参考坐标系为笛卡尔坐标系（非 ACS）时，运动学换算后的当前轴组反馈位置

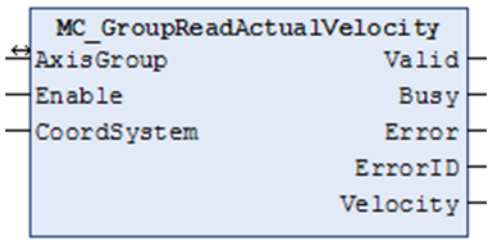
功能说明：

- 读取轴组在指定坐标系下的反馈位置。
- 选择的坐标系，可以参考 7.1 节的“SMC_COORD_SYSTEM”说明。

读取轴组反馈速度 MC_GroupReadActualVelocity

读取轴组在指定坐标系下的反馈速度。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupReadActualVelocity	FB		MC_GroupReadActualVelocity(AxisGroup: = , Enable: = , CoordSystem: = , Valid=> , Busy=> , Error=> , ErrorID=> , Velocity=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	为 True 时连续获取轴组在参考坐标系中的实际速度
CoordSystem	坐标系	SMC_COORD_SYSTEM	-	-	参考坐标系
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE FALSE	FALSE	当获取有效数值时为 True
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
Velocity	速度	SMC_POS_REF	-	-	轴组的当前反馈速度。如果选择的是笛卡尔坐标系；Velocity.c 包含笛卡尔速度：(X，Y，Z)是速度矢量，(A，B，C)分别是 x,y,z 轴周围的角速度。

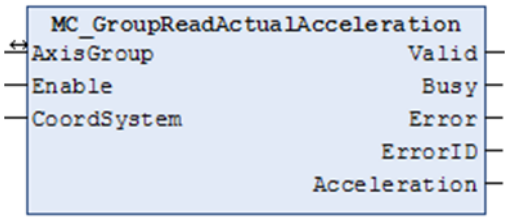
功能说明：

- 读取轴组在指定坐标系下的反馈速度。
- 选择的坐标系，可以参考 7.1 节的“SMC_COORD_SYSTEM”说明。

读取轴组反馈加速度 MC_GroupReadActualAcceleration

读取轴组在指定坐标系下的反馈加速度。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupReadActualAcceleration	FB		MC_GroupReadActualAcceleration(AxisGroup: = , Enable: = , CoordSystem: = , Valid=> , Busy=> , Error=> , ErrorID=> , Acceleration=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	为 True 时连续获取轴组在参考坐标系中的实际加速度
CoordSystem	坐标系	SMC_COORD_SYSTEM	-	-	参考坐标系
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE FALSE	FALSE	当获取有效数值时为 True
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
Acceleration	加速度	SMC_POS_REF	-	-	轴组的当前反馈加速度。如果选择的是笛卡尔坐标系，Velocity.c 包含笛卡尔加速度：(X，Y，Z)是加速度矢量，(A，B，C)分别是 x，y，z 轴周围的角加速度。

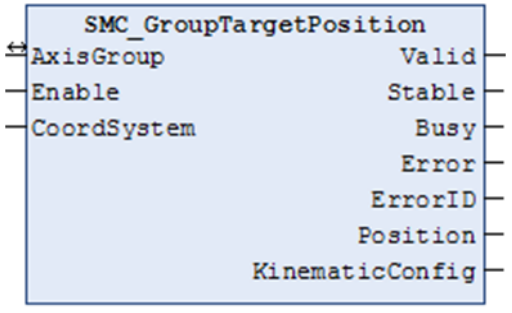
功能说明：

- 读取轴组在指定坐标系下的反馈加速度。
- 选择的坐标系，可以参考 7.1 节的“SMC_COORD_SYSTEM”说明。

读取轴组目标位置 SMC_GroupTargetPosition

读取轴组在指定坐标系下的目标位置。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupTargetPosition	FB		SMC_GroupTargetPosition(AxisGroup: = , Enable: = , CoordSystem: = , Valid=> , Stable=> , Busy=> , Error=> , ErrorID=> , Position=> , KinematicConfig=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	为 True 时连续获取轴组在参考坐标系中的指令目标位置
CoordSystem	坐标系	SMC_COORD_SYSTEM	-	-	参考坐标系
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE FALSE	FALSE	当获取有效数值时为 True
Stable	获取位置标志	BOOL	TRUE FALSE	FALSE	在接受到新的移动位置之前，状态一直保持为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
Position	位置	SMC_POS_REF	-	-	轴组的目标位置
KinematicConfig	运动配置	TRAFO.CONFIGDATA	-	-	当参考坐标系为笛卡尔坐标系（非 ACS）时，运动学换算后的当前轴组目标位置

功能说明：

- 读取轴组在指定坐标系下的目标位置，在以下几种情况下，Valid 为 FALSE：

1）最后获取的是一个相对位置指令，同时开始位置也是未知的（例如这是一个跟踪运动）；

2）最后的运动是一个暂停/停止命令，且当前未执行；

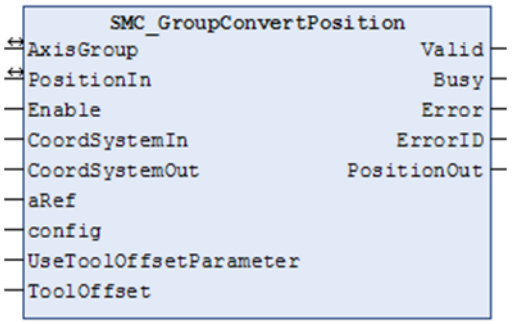
3）尚未执行的运动指令和当前的参考坐标系不一致。

- 当起始位置已知的情况下，指令在每个周期能获得一次数据。
- Valid 为 True 和 Stable 为 FALSE 时，说明轴组正在参考坐标系下运动。
- 选择的坐标系，可以参考 7.1 节的“SMC_COORD_SYSTEM”说明。

轴组位置转换 SMC_GroupConvertPosition

轴组中在输入坐标系下的位置信息转换为输出坐标系下的位置。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupConvertPosition	FB		SMC_GroupConvertPosition(AxisGroup: = , PositionIn: = , Enable: = , CoordSystemIn: = , CoordSystemOut: = , aRef: = , config: = , UseToolOffsetParameter: = , ToolOffset: = , Valid=> , Busy=> , Error=> , ErrorID=> , PositionOut=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
PositionIn	位置	SMC_POS_REF	-	-	待转换的位置
VAR_INPUT					
Enable	启动转换	BOOL	TRUE FALSE	FALSE	开始进行位置转换
CoordSystemIn	带转换坐标系	SMC_COORD_SYSTEM	-	-	待转换坐标系
CoordSystemOut	目标坐标系	SMC_COORD_SYSTEM	-	-	目标坐标系
aRef	轴组参考位置	TRAFO.AXISPOS_REF	-	-	轴组的参考位置，只适用于： CoordSystemOut = ACS 且 CoordSystemIn <> ACS
config	配置	TRAFO.CONFIGDATA	-	-	运动学变换的配置，只适用于： CoordSystemOut = ACS 且 CoordSystemIn <> ACS
UseToolOffsetParameter	工件偏移参数	BOOL	TRUE FALSE	FALSE	无论是否使用 ToolOffset（工具偏移参数），只要这个参数为 FALSE，总会调用 SMC_GroupSetTool 对最后的一个偏移量参数进行设置

ToolOffset	工件偏移	MC_COORD_REF	-	-	用于转换位置的偏移量工具
VAR_OUTPUT					
Valid	获取标志	BOOL	TRUE FALSE	FALSE	当获取有效数值时为 True
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
PositionOut	位置	SMC_POS_REF	-	-	转换成目标坐标系后，输出的位置

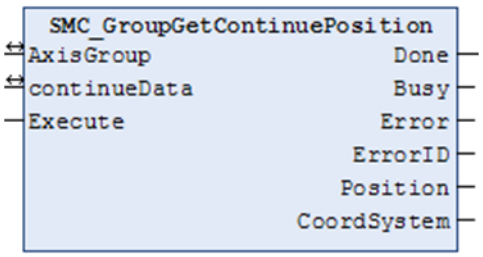
功能说明：

- 轴组中在输入坐标系下的位置信息转换为输出坐标系下的位置；
- 输入的坐标系信息可不为当前正在进行的运动所在的坐标系，也可以是将要进行的运动所在的坐标系；
- 选择的坐标系，可以参考 7.1 节的“SMC_COORD_SYSTEM”说明。

读取连续运动位置 SMC_GroupGetContinuePosition

读取轴组运动中断时的位置。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupGetContinuePosition	FB		SMC_GroupGetContinuePosition(AxisGroup: = , continueData: = , Execute: = , Done=> , Busy=> , Error=> , ErrorID=> , Position=> , CoordSystem=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
continueData	继续运动数据	SMC_AXIS_GROUP_CONTINUE_DATA	-	-	继续运动数据

VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令，上升沿有效
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误为 True
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
Position	中断时位置	SMC_POS_REF	-	-	中断时的运动位置
CoordSystem	坐标系	SMC_COORD_SYSTEM	-	-	存储位置的参考坐标系，一般是 ACS，如果跟踪运动已经停止，则是 PCS

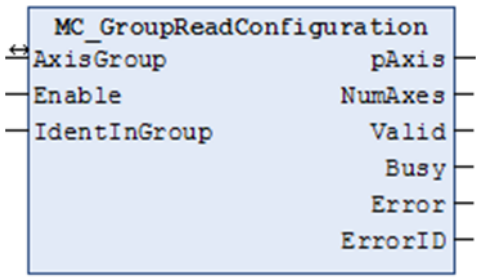
功能说明：

- 如果该位置被判断能够调用 MC_GroupContinue 指令继续运动，将从 ContinueData 之中读取继续运动的位置数据。
- 选择的坐标系，可以参考 7.1 节的“SMC_COORD_SYSTEM”说明。

读取轴组配置参数 MC_GroupReadConfiguration

读取轴组包含的轴、数量等配置参数。

指令外观：

指令	FB/ FUN	图形模块	结构文本
MC_GroupRead Configuration	FB		MC_GroupReadConfiguration(AxisGroup: = , Enable: = , IdentInGroup: = , pAxis=> , NumAxes=> , Valid=> , Busy=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组

VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	输入 True 时获取轴组信息
IdentInGroup	轴编号	IDENT_IN_GROUP_REF_SM3	0，正数	0	输入轴组中相应轴的编号
VAR_OUTPUT					
pAxis	参考轴	POINTER TO AXIS_REF_SM3	-	-	选定的参考轴
NumAxes	轴数量	UDINT	0，正数	0	轴组中轴的数量
Valid	获取有效	BOOL	TRUE FALSE	FALSE	当获取有效数值时为 True
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

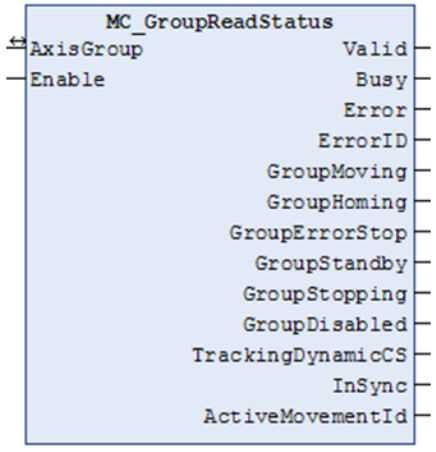
功能说明：

- 读取轴组包含的轴、数量等配置参数。
- 选择的坐标系，可以参考 7.1 节的“SMC_COORD_SYSTEM”说明。

读取轴组的当前运动状态 MC_GroupReadStatus

用于获取轴组的当前运动状态。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupReadStatus	FB		MC_GroupReadStatus(AxisGroup: = , Enable: = , Valid=> , Busy=> , Error=> , ErrorID=> , GroupMoving=> , GroupHoming=> , GroupErrorStop=> , GroupStandby=> , GroupStopping=> , GroupDisabled=> , TrackingDynamicCS=> , InSync=> , ActiveMovementId=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	输入为 True 时，持续获取轴组状态信息
VAR_OUTPUT					
Valid	有效	BOOL	TRUE FALSE	FALSE	当获取有效数值时为 True
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
GroupMoving	运动中	BOOL	TRUE FALSE	FALSE	轴组正在运动中为 True
GroupHoming	回零中	BOOL	TRUE FALSE	FALSE	轴组正在回零中为 True
GroupErrorStop	错误停止	BOOL	TRUE FALSE	FALSE	轴组报错停止为 True
GroupStandby	准备运动	BOOL	TRUE FALSE	FALSE	轴组运动准备状态为 True
GroupStopping	停止中	BOOL	TRUE FALSE	FALSE	轴组运动停止中为 True
GroupDisabled	未启用轴组	BOOL	TRUE FALSE	FALSE	轴组无效状态为 True
TrackingDynamicCS	当前是动态坐标系	BOOL	TRUE FALSE	FALSE	当前使用的是动态坐标系时为 True
InSync	在路径上或已到位	BOOL	TRUE FALSE	FALSE	在连续插补运动中，当获取位置属于指定路径时为 True；在普通点位、插补运动中，当前位置等于目标位置时为 True
ActiveMovementId	运动段号	SMC_Movement_Id	0，正数	0	运动标识符，从 0 开始，每进行一次运动加 1

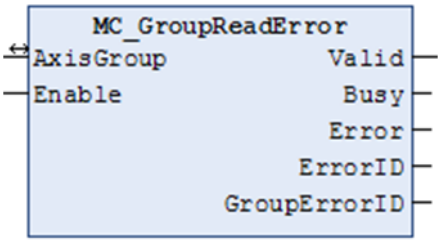
功能说明：

调用该指令能够持续获取轴组的运动状态、是否到位，以及运动段号等信息。

读取轴组错误 MC_GroupReadError

获取轴组的错误信息。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupReadError	FB		MC_GroupReadError(AxisGroup: = , Enable: = , Valid=> , Busy=> , Error=> , ErrorID=> , GroupErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	为 True 时持续获取轴组报错信息
VAR_OUTPUT					
Valid	有效	BOOL	TRUE FALSE	FALSE	当获取有效数值时为 True
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	单轴错误代码指示，见 SMC_Error
GroupErrorID	错误代码	SMC_ERROR	-	0	轴组错误代码指示，见 SMC_Error

功能说明：

获取单轴和轴组的报错信息，比如读取硬限位（或软限位）或单轴报错。

7.2.3 坐标系指令

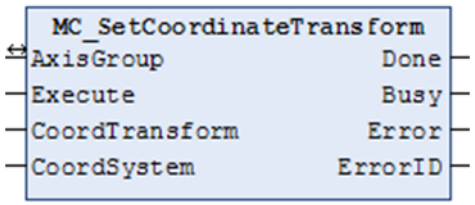
表 7.9 坐标系指令

	功能说明
MC_SetCoordinateTransform	用来转换不同参考坐标系的指令坐标
MC_SetDynCoordTransform	指定的坐标系相对于 WCS 移动时，需要调用该指令实现坐标系转换

坐标系转换 MC_SetCoordinateTransform

用来转换不同参考坐标系的指令坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_SetCoordinate Transform	FB		MC_SetCoordinateTransform(AxisGroup: = , Execute: = , CoordTransform: = , CoordSystem: = , Done=> , Busy=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令的执行
CoordTransform	待转换坐标系	MC_COORD_REF	-	-	待转换坐标系，比如在世界坐标系（WCS)中表示产品坐标系（PCS_1 或 PCS_2)或机械坐标系（MCS)
CoordSystem	坐标系	SMC_COORD_SYSTEM	-	-	目标坐标系，允许转换成 PCS_1， PCS_2， 和 MCS.
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

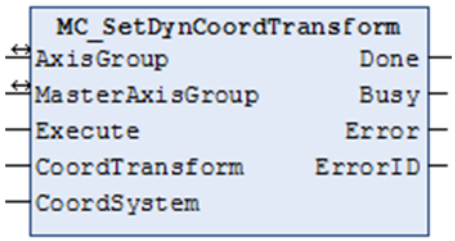
功能说明：

- 设置轴组中世界坐标系（WCS）与 产品/机器坐标系（PCS_*/MCS）进行坐标系的转换。
- 当 PCS 是动态的坐标系（PCS 相对于 WCS 移动），需要使用 MC_SetDynCoordTransform 指令。
- 该指令只进行坐标系的转换，与运动控制指令无关。

动态坐标系转换 MC_SetDynCoordTransform

指定的坐标系相对于 WCS 移动时，需要调用该指令实现坐标系转换。

指令外观：

指令	FB/ FUN	图形模块	结构文本
MC_SetDynCoord Transform	FB		MC_SetDynCoordTransform (AxisGroup: = , MasterAxisGroup: = , Execute: = , CoordTransform: = , CoordSystem: = , Done=> , Busy=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
MasterAxisGroup	主轴组	AXIS_GROUP_REF_SM3	-	-	指定的主轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令的执行
CoordTransform	待转换坐标系	MC_COORD_REF	-	-	主轴组的工具坐标系相对与 PCS 的坐标和方向
CoordSystem	坐标系	SMC_COORD_SYSTEM	-	-	将要转换的 PCS 坐标系(PCS_1 或 PCS_2)
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE

Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

功能说明：

- 通常在用到动态 PCS 时使用，比如一般会和 MC_TrackConveyorBelt 或 MC_TrackRotaryTable 一起被调用。
- 指令 SMC_SetDynCoordTransformEX 提供更加通用的接口。

7.2.4 动力学指令

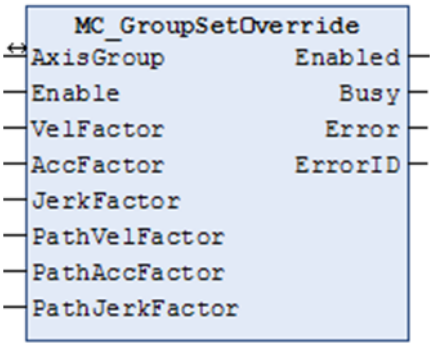
表 7.10 动力学指令

指令名	功能说明
MC_GroupSetOverride	轴组在 Moving 状态下，改变轴组的运动速度
SMC_GroupSetAncillaryAxisLimits	限制轴组的运动参数等
SMC_GroupSetAncillaryPathLimits	设置轴组的辅助动态路径限制
SMC_SetDynamicLimitFactors	对轴组中所有轴的速度等运动参数做动态限制

设置轴组超调值 MC_GroupSetOverride

轴组在 Moving 状态下，改变轴组的运动速度。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupSetOverride	FB		MC_GroupSetOverride(AxisGroup: = , Enable: = , VelFactor: = , AccFactor: = , JerkFactor: = , PathVelFactor: = , PathAccFactor: = , PathJerkFactor: = , Enabled=> , Busy=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	为 TRUE 时，输入参数的改变实时生效； 为 FALSE 时，则保持最后一个周期的数值
VelFactor	速度因子	LREAL	0-1	1	速度因子，每个轴的最大速度乘以这个速度因子，数值在[0，1]之间
AccFactor	加速度因子	LREAL	0-1	1	加速度因子，每个轴的最大加速度乘以这个加速度因子，数值在[0，1]之间
JerkFactor	加加速度因子	LREAL	0-1	1	加加速度因子，每个轴的最大加加速度乘以这个加加速度因子，数值在[0，1]之间
PathVelFactor	合速度因子	LREAL	0-1	1	合速度因子，整个轴组运动轨迹的最大合速度乘以这个速度因子，数值在[0，1]之间
PathAccFactor	合加速度因子	LREAL	0-1	1	合加速度因子，整个轴组运动轨迹的最大合加速度乘以这个合加速度因子，数值在[0，1]之间
PathJerkFactor	合加加速度因子	LREAL	0-1	1	合加加速度因子，整个轴组运动轨迹的最大合加加速度乘以这个合加加速度因子，数值在[0，1]之间
VAR_OUTPUT					
Enabled	设置完成	BOOL	TRUE FALSE	FALSE	当超调因子设置完成为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

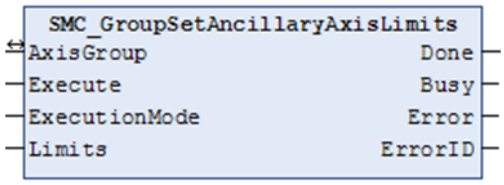
功能说明：

- 轴组在 Moving 状态下，通过执行该指令能够改变运动的速度、加速度和加加速度。
- 给定的超调因子在设定新的数值之前持续有效。比如处于 CP 运动之中（即使是跟随运动）的轴组，将 VelFactor 或 PathVelFactor 设置成 0 将导致运动轨迹的突然停止。如果 MC_GroupStop 当前处于 Active 的状态，则会返回报错。
- MC_GroupStop 指令不会受到超调因子的影响。
- 重新启用轴组，超调因子依然有效。
- 超调因子的默认数值为 1。
- 减少 AccFactor 或 JerkFactor 会导致位置的超调，可能会给设备造成损坏。

辅助轴限制 SMC_GroupSetAncillaryAxisLimits

限制轴组的运动参数等。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupSetAncillaryAxisLimits	FB		SMC_GroupSetAncillaryAxisLimits(AxisGroup: = , Execute: = , ExecutionMode: = , Limits: = , Done=> , Busy=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT		类型		初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令的执行
ExecutionMode	执行模式	MC_EXECUTION_MODE	-	-	执行模式： Immediately：立即生效 Queued：暂缓执行
Limits	动态限制	ARRAY [0..(SMC_RCNST.MAX_AXES - 1)] OF SMC_DYN_LIMITS	-	-	辅助动态限制数组
VAR_OUTPUT					

Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

功能说明：

- 设置轴组运动参数的最大值限制，即每个轴的速度、加速度、减速度的限制。
- 能够通过编程对的不同操作模式的机器人进行辅助的限制，比如因为安全问题，对机器人的手动模式进行缓慢移动的限制。
- 运动的有效动态限制是由以下输入构建起来的：

  G：存储在轴配置之中的全局轴限制

  M：在运动函数块（输入的 VelFactor/AccFactor…等数值）上给出的因子

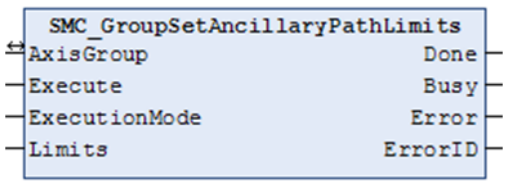
  O：当前的超调因子

  A：辅助限制
- 从参数中计算出有效轴限 L 的计算公式为：$L = O \times \mathrm{Min}(A, M \times G)$
- 如果在 GroupDisabled 或 GroupErrorStop 状态下，使用立即执行模式去调用该指令，可能导致 MC_GroupStop 报错。
- 轴的极限值不能为 0 或负数，否则将会报错。
- MC_GroupStop 指令不影响辅助限制。
- 使用立即执行模式去减少加速度或加加速度，会导致位置的超调，可能会给设备造成损坏。

辅助路径限制 SMC_GroupSetAncillaryPathLimits

设置轴组的辅助动态路径限制。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupSetAncillaryPathLimits	FB		SMC_GroupSetAncillaryPathLimits(AxisGroup: = , Execute: = , ExecutionMode: = , Limits: = , Done=> , Busy=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令的执行
ExecutionMode	执行模式	MC_EXECUTION_MODE	-	-	执行模式： Immediately：立即生效 Queued：暂缓执行
Limits	动态限制	ARRAY [0..(SMC_RCNST.MAX_AXES - 1)] OF SMC_DYN_LIMITS	-	-	辅助动态限制
VAR_OUTPUT					
Done	指令完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

功能说明：

- 运动的有效动态限制是由以下输入构建起来的：

  C： 对路劲中的运动进行限制（比如速度、加速度等）

  O：当前路劲的超调因子

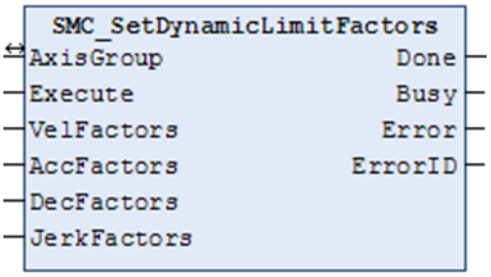
  A：辅助路劲限制

- 从参数中计算出路径限制 L 的计算公式为：L= O × Min(A，C)
- 如果在 GroupDisabled 或 GroupErrorStop 状态下，使用立即执行模式去调用该指令，可能导致 MC_GroupStop 报错。
- 轴的极限值不能为 0 或负数，否则将会报错。
- MC_GroupStop 指令不影响辅助限制。
- 使用立即执行模式去减少加速度或加加速度，会导致位置的超调，可能会给设备造成损坏。

动态限制因子设置 SMC_SetDynamicLimitFactors

对轴组中所有轴的速度等运动参数做动态限制。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_SetDynamicLimitFactors	FB		SMC_SetDynamicLimitFactors(AxisGroup: = , Execute: = , VelFactors: = , AccFactors: = , DecFactors: = , JerkFactors: = , Done=> , Busy=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令的执行
VelFactors	速度因子	ARRAY [0..(SMC_RCNST.MAX_AXES - 1)] OF LREAL	0-1	1	轴速度因子数组，每个轴的最大速度乘以这个速度因子，数值在[0，1]之间
AccFactors	加速度因子	ARRAY [0..(SMC_RCNST.MAX_AXES - 1)] OF LREAL	0-1	1	轴加速度因子数组，每个轴的最大加速度乘以这个加速度因子，数值在[0，1]之间
DecFactors	减速度因子	ARRAY [0..(SMC_RCNST.MAX_AXES - 1)] OF LREAL	0-1	1	轴减速度因子数组，每个轴的最大减速度乘以这个减速度因子，数值在[0，1]之间
JerkFactors	加加速度因子	ARRAY [0..(SMC_RCNST.MAX_AXES - 1)] OF LREAL	0-1	1	加加速度因子数组，每个轴的最大加加速度乘以这个加加速度因子，数值在[0，1]之间
VAR_OUTPUT					
Done	指令完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为TRUE

Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

功能说明：

- 在 MC_GroupEnable 之后，初始化所有的设置因子都是 1。
- 在 GroupDisabled 或 GroupErrorStop 状态下禁止调用该指令。
- 辅助极限设置为给出的动态因子和轴的全局动态极限的乘积，详细见 SMC_GroupSetAncillaryAxisLimits.

7.2.5 运动学指令

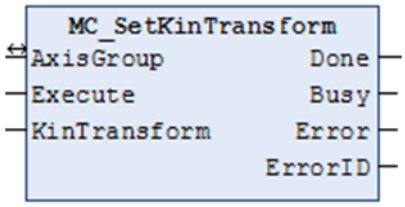
表 7.11 运动学指令

指令名	功能说明
MC_SetKinTransform	设置轴组的运动学转换，从 ACS 坐标系到 MCS 坐标系
SMC_SetKinConfiguration	配置机构的逆运动学参数

运动学坐标变换 MC_SetKinTransform

设置轴组的运动学转换，从 ACS 坐标系到 MCS 坐标系。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_SetKinTransform	FB		MC_SetKinTransform(AxisGroup: = , Execute: = , KinTransform: = , Done=> , Busy=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令的执行
KinTransform	运动学变换值	TRAFO.MC_KIN_REF_SM3	-	-	运动学变换

VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

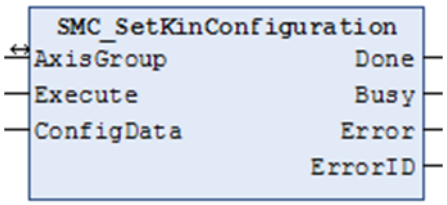
功能说明：

- 设置轴坐标系(ACS)和机器坐标系(MCS)之间进行运动学变换。
- 执行此功能块后，将重置工具偏移量(见 SMC_GroupSetTool)。

运动学坐标参数配置 SMC_SetKinConfiguration

配置机构的运动学逆变换参数。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_SetKinConfiguration	FB		SMC_SetKinConfiguration(AxisGroup: = , Execute: = , ConfigData: = , Done=> , Busy=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令的执行
ConfigData	配置数据	TRAFO.CONFIGDATA	-	-	运动学参数配置
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE

Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

7.2.6 轴组运动指令

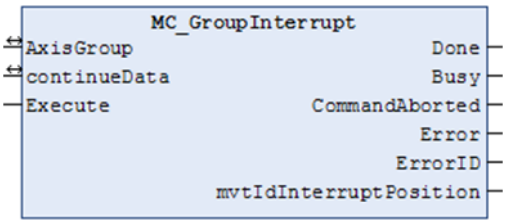
表 7.12 轴组运动指令

指令名	功能说明
MC_GroupInterrupt	中断当前正在运动的轴组，可通过 MC_GroupContinue 指令继续执行未执行完的运动指令
MC_GroupContinue	解除轴组的中断状态，继续执行未完成的指令
MC_GroupHalt	用于暂停当前的轴组运动
MC_GroupStop	停止轴组的运动
MC_MoveCircularAbsolute	控制轴组执行绝对位置模式下的圆弧插补运动
MC_MoveCircularRelative	控制轴组执行相对位置模式的圆弧插补运动
MC_MoveDirectAbsolute	控制轴组内所有轴各自以指定速度运行到绝对位置终点
MC_MoveDirectRelative	控制轴组内所有轴各自以指定速度运行到相对位置的终点
MC_MoveLinearAbsolute	控制轴组在指定坐标系下的绝对位置模式的直线插补运动
MC_MoveLinearRelative	控制轴组在指定坐标系下的相对位置模式的直线插补运动
SMC_GroupEnableResumeAfterError	恢复因报错而被中断的轴组状态
SMC_GroupJog	控制轴组在指定坐标系下进行 Jog 运动
SMC_GroupWait	设定轴组的延时等待

轴组中断 MC_GroupInterrupt

中断当前正在运动的轴组，可通过 MC_GroupContinue 指令继续执行未执行完的运动指令。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupInterrupt	FB		MC_GroupInterrupt(AxisGroup: = , continueData: = , Execute: = , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=> , mvtIdInterruptPosition=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
ContinueData	继续运动数据	SMC_AXIS_GROUP_CONTINUE_DATA	-	-	轴组运动中断时的运动信息
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	指令执行被中断为 True
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
mvtIdInterruptPosition		SMC_Movement_Id			与中断位置对应的 I

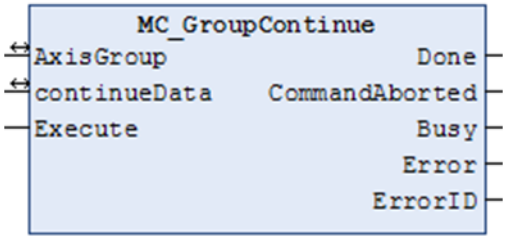
功能说明：

- 轴组在执行 MC_MoveDirectAbsolute、MC_MoveDirectRelative、MC_MoveLinearAbsolute、MC_MoveLinearRelative 与 MC_MoveCircularAbsolute 时，可调用 MC_GroupInterrupt 指令中断当前的运动，不支持其它类型运动指令。
- 执行 MC_GroupInterrupt 指令后，需要使用 MC_GroupContinue 指令恢复被中断的轴组运动。
- 多个 MC_GroupInterrupt 指令同时执行，它们的执行效果一致。

轴组继续运行 MC_GroupContinue

解除轴组的中断状态，继续执行未完成的指令。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupContinue	FB		MC_GroupContinue(AxisGroup: = , continueData: = , Execute: = , Done=> , CommandAborted=> , Busy=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
continueData	继续运动数据	SMC_AXIS_GROUP_CONTINUE_DATA	-	-	运动中断时的轴组位置
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
VAR_OUTPUT					
Done	完成	BOOL	TRUE FALSE	FALSE	指令执行完成时为 TRUE
CommandAborted	指令被中断	BOOL	TRUE FALSE	FALSE	模块执行被中断为 True
Busy	执行中	BOOL	TRUE FALSE	FALSE	指令正在执行中为 TRUE
Error	错误	BOOL	TRUE FALSE	FALSE	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

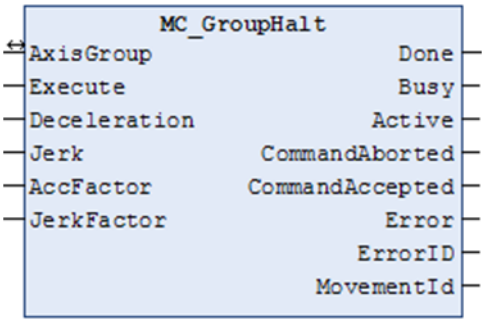
功能说明：

- 继续执行被“MC_GroupInterrupt”中断的轴组运动。仅当轴组状态为“Interrupted”时，“MC_GroupContinue”指令才能生效。
- 只有在轴组执行 MC_MoveDirectAbsolute、MC_MoveDirectRelative、MC_MoveLinearAbsolute、MC_MoveLinearRelative 与 MC_MoveCircularAbsolute 指令被 MC_GroupInterrupt 暂停时，才能够执行 MC_GroupContinue 指令来继续执行轴组运动。
- 只有当前轴组状态正常，且已启用轴组功能的轴组，才能够调用 MC_GroupContinue 指令恢复轴组运动。

轴组暂停 MC_GroupHalt

用于暂停当前的轴组运动

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupHalt	FB		MC_GroupHalt(AxisGroup: = , Execute: = , Deceleration: = , Jerk: = , AccFactor: = , JerkFactor: = , Done=> , Busy=> , Active=> , CommandAborted=> , CommandAccepted=> , Error=> , ErrorID=> , MovementId=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发轴组运动暂停
Deceleration	减速度	LREAL	0，正数	0	最大合减速度 [u/s²]，只能输入正数
Jerk	加加速度	LREAL	0，正数	0	最大合加加速度 [u/s²]，只能输入正数
AccFactor	加速度因子	LREAL	0-1	1	加速度因子，每个轴的最大速度乘以这个加速度因子，数值在[0，1]之间
JerkFactor	加加速度因子	LREAL	0-1	1	加加速度因子，每个轴的最大速度乘以这个加速度因子，数值在[0，1]之间
VAR_OUTPUT					
Done	完成	BOOL	TRUE，FALSE	FALSE	指令执行完成时为 TRUE
Busy	运动中	BOOL	TRUE，FALSE	FALSE	指令正在执行中为 TRUE
Active	启动中	BOOL	TRUE，FALSE	FALSE	当轴正在被这个模块调用时为 TRUE
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	模块执行被中断为 True
CommandAccepted	运动接收	BOOL	TRUE，FALSE	FALSE	当模块成功调用轴组时为 True

Error	报错	BOOL	TRUE，FALSE	FALSE	功能块执行错误
ErrorID	错误码 ID	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
MovementId	运动标志	SMC_Movement_Id	TRUE，FALSE	FALSE	当运动正在执行或完成，则为 True

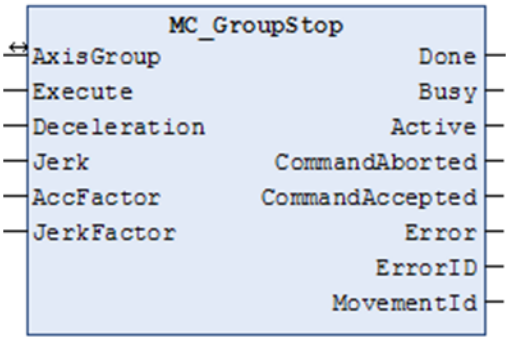
功能说明：

- 这个指令用于暂停轴组运动，能够暂停除 MC_GroupStop 之外的其它运动模块。在调用了指令后，轴组的运动从 GroupMoving 转换成为 GroupStandby 状态。
- 如果 Deceleration 和 Jerk 的数值太小，可能导致实际减速由运动指令的 Dec、Jerk、ACC 因子和 Jerk 因子共同决定，运动效果可能与规划的不一致。为了避免不必要的错误发生，请确保 Deceleration 和 Jerk 的数值至少能达到暂停运动目的
- 对于停止点对点运动，每个轴的速度/加速度/减速/抖动是每个轴的属性，而不是在这个函数块中指定的
- 在调用 MC_GroupHalt 前，需要先调用 SMC_GroupSaveContinueData 指令，用来保存当前位置、状态和运动指令，以便在需要时，能够从暂停的位置继续运动。

轴组停止 MC_GroupStop

停止轴组的运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_GroupStop	FB		MC_GroupStop(AxisGroup: = , Execute: = , Deceleration: = , Jerk: = , AccFactor: = , JerkFactor: = , Done=> , Busy=> , Active=> , CommandAborted=> , CommandAccepted=> , Error=> , ErrorID=> , MovementId=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					

Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
Deceleration	减速度	LREAL	0，正数	0	最大合减速度 [u/s²]，只能输入正数
Jerk	加加速度	LREAL	0，正数	0	最大合加加速度 [u/s²]，只能输入正数
AccFactor	加速度因子	LREAL	0-1	1	加速度因子，每个轴的最大速度乘以这个加速度因子，数值在[0，1]之间
JerkFactor	加加速度因子	LREAL	0-1	1	加加速度因子，每个轴的最大速度乘以这个加速度因子，数值在[0，1]之间
VAR_OUTPUT					
Done	完成	BOOL	TRUE，FALSE	FALSE	指令执行完成时为 TRUE
Busy	运动中	BOOL	TRUE，FALSE	FALSE	指令正在执行中为 TRUE
Active	启动中	BOOL	TRUE，FALSE	FALSE	当轴正在被这个模块调用时为 TRUE
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	模块执行被中断为 True
CommandAccepted	运动接收	BOOL	TRUE，FALSE	FALSE	当模块成功调用轴组时为 True
Error	报错	BOOL	TRUE，FALSE	FALSE	功能块执行错误
ErrorID	错误码 ID	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
MovementId	运动标志	SMC_Movement_Id	TRUE，FALSE	FALSE	当运动正在执行或完成，则为 True

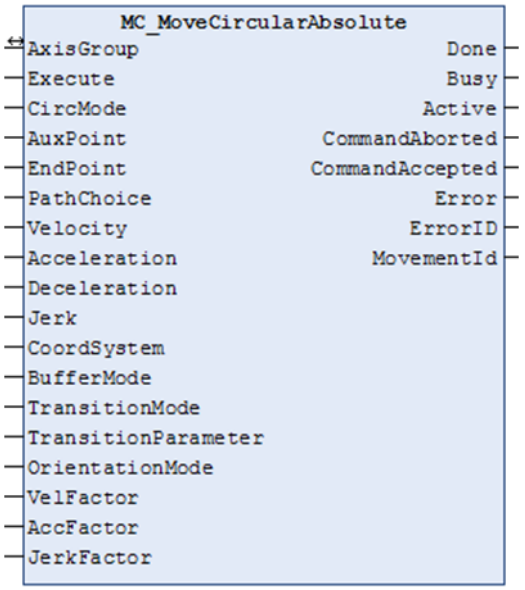
功能说明：

- 停止正在运动的 AxesGroup(轴组)，让轴组中的所有轴停止动作，并使轴组状态切换为 GroupStopping。
- 停止轴动作时，MC_MoveLinear(直线插补)指令、MC_MoveLinearAbsolute(绝对值直线插补)指令、MC_MoveLinearRelative(相对值直线插补)指令、MC_MoveCircular2D(2 轴圆弧插补)指令会以 Deceleration(减速度)减速停止；MC_GroupSyncMoveAbsolute(轴组周期同步绝对位置控制)指令则执行立即停止。
- 若在执行插补指令时，调用该指令，将在直线插补或圆弧插补的轨迹上减速停止。

绝对圆弧插补 MC_MoveCircularAbsolute

控制轴组执行绝对位置模式下的圆弧插补运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_MoveCircularAbsolute	FB		MC_MoveCircularAbsolute (AxisGroup: = , Execute: = , CircMode: = , AuxPoint: = , EndPoint: = , PathChoice: = , Velocity: = , Acceleration: = , Deceleration: = , Jerk: = , CoordSystem: = , BufferMode: = , TransitionMode: = , TransitionParameter: = , OrientationMode: = , VelFactor: = , AccFactor: = , JerkFactor: = , Done=> , Busy=> , Active=> , CommandAborted=> , CommandAccepted=> , Error=> , ErrorID=> , MovementId=>);

变量：

VAR_IN_OUT	名称	数据类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT	**名称**	**数据类型**	**有效范围**	**初始值**	**描述**
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
CircMode	圆弧模式	SMC_CIRC_MODE	0-2	0	指定圆弧插补的方法，见“SMC_CIRC_MODE”说明。 0： _mcBorder，三点圆弧； 1： _mcCenter，中心圆弧； 2： _mcRadius，半径圆弧
AuxPoint	辅助点	SMC_POS_REF	负数，0 正数	0	辅助点位置，见“SMC_CIRC_MODE”说明
EndPoint	终点	SMC_POS_REF	负数，0 正数	0	终点位置，见“SMC_CIRC_MODE”说明
PathChoice	方向	MC_CIRC_PATHCHOICE	0，1	0	运动方向，0：顺时针；1：逆时针
Velocity	速度	LREAL	0，正数	0	最大合速度[u/s]，只能输入正数
Acceleration	加速度	LREAL	0，正数	0	最大合加速度 [u/s²]，只能输入正数
Deceleration	减速度	LREAL	0，正数	0	最大合减速度 [u/s²]，只能输入

					正数
Jerk	加加速度	LREAL	0，正数	0	最大合加加速度 [u/s²]，只能输入正数
CoordSystem	参考坐标系	SMC_COORD_SYSTEM	SMC_COORD_SYSTEM	-	参考坐标系
BufferMode	缓存模式	MC_BUFFER_MODE	0-5	0	指定多重启动运动指令时的动作。 0： mcAborting，中断； 1： mcBuffered，等待； 2： mcBlendingLow，以低速合并； 3： mcBlendingPrevious，以前一个速度合并； 4： mcBlendingNext，以后一个速度合并； 5： mcBlendingHigh，以高速合并
TransitionMode	拐角过渡模式	MC_TRANSITION_MODE	TMNone，TMStartVelocity，TMCornerDistance	0	TMNone：不设置拐角过渡 TMStartVelocity：基于速度的拐角过渡模式 TMCornerDistance：基于距离的拐角过渡模式
TransitionParameter	拐角过渡参数	ARRAY [0..(SMC_RCNST.MAX_TRANS_PARAMS - 1)] OF LREAL	0，正数	0	拐角过渡参数
OrientationMode	插补定位模式	SMC_ORIENTATION_MODE	GreatCircle，Axis	0	GreatCircle：沿着最短的路径从起始位置向目标位置运动，在这种模式下，即使起始位置和终点位置都在指定区域内，但实现的路径也可能会离开这个区域。 Axis：定位轴从起始位置到终点位置都在指定的区域内运动，不是所有的运动学变换都支持该种模式。
VelFactor	速度因子	LREAL	0-1	1	速度因子，每个轴的最大速度乘以这个速度因子，数值在[0，1]之间
AccFactor	加速度因子	LREAL	0-1	1	加速度因子，每个轴的最大加速度乘以这个加速度因子，数值在[0，1]之间
JerkFactor	加加速度因子	LREAL	0-1	1	加加速度因子，每个轴的最大加加速度乘以这个加加速度因子，数值在[0，1]之间
VAR_OUTPUT	**名称**	**数据类型**	**有效范围**	**初始值**	**描述**
Done	完成	BOOL	TRUE，FALSE	FALSE	指令执行完成时为 TRUE
Busy	运动中	BOOL	TRUE，FALSE	FALSE	指令正在执行中为 TRUE
Active	启动中	BOOL	TRUE，FALSE	FALSE	当轴正在被这个模块调用时为 TRUE
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	模块执行被中断为 True

CommandAccepted	运动接收	BOOL	TRUE，FALSE	FALSE	当模块成功调用轴组时为 True
Error	报错	BOOL	TRUE，FALSE	FALSE	功能块执行错误为 True
ErrorID	错误码 ID	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
MovementId	运动标志	SMC_Movement_Id	TRUE，FALSE	FALSE	CommandAccepted 或 Done 为 TRUE 时有效

功能说明：

- Execute 的上升沿触发指令运动，下降沿对指令运动没有影响。
- 当使用圆心圆弧模式时，输入参数 AuxPoint[1]、AuxPoint[2]的数值为圆心与圆弧起点的距离；而当使用半径圆弧模式时，AuxPoint[1]表示半径数值，AuxPoint[2]无效。
- EndPoint[1]~EndPoint[8]的数值表示各轴的终点坐标。

时序图：

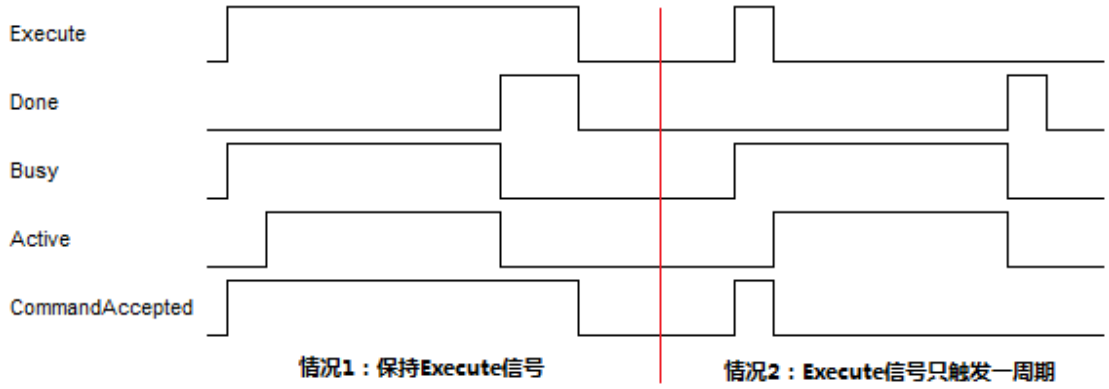


图 7.7　MC_MoveDirectAbsolute 主要状态变量的时序图

情况 1：

当 Execute 由 FALSE 变为 TRUE ，Busy 和 CommandAccepted 变为 TRUE，且 1 个周期后，Active 变为 TRUE；当轴组到达终点位置时，Done 变为 TRUE，同时 Busy 和 Active 变为 FALSE；当 Execute 也变成 FALSE 时，Done 和 CommandAccepted 也变成 FALSE。

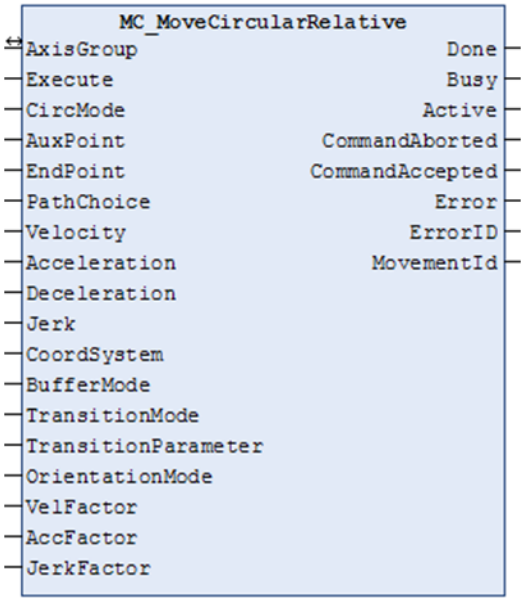
情况 2：

当 Execute 由 FALSE 变为 TRUE 时，Busy 和 CommandAccepted 变为 TRUE，且 1 个周期后，Active 变为 TRUE；Execute 信号只触发一个周期，CommandAccepted 也跟着只触发一个周期；当轴组到达终点位置时，Done 变为 TRUE，同时 Busy 和 Active 变为 FALSE；且一个周期后 Done 变成 FALSE。

相对圆弧插补 MC_MoveCircularRelative

控制轴组执行相对位置模式的圆弧插补运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_MoveCircularRelative	FB		MC_MoveCircularRelative(AxisGroup: = , Execute: = , CircMode: = , AuxPoint: = , EndPoint: = , PathChoice: = , Velocity: = , Acceleration: = , Deceleration: = , Jerk: = , CoordSystem: = , BufferMode: = , TransitionMode: = , TransitionParameter: = , OrientationMode: = , VelFactor: = , AccFactor: = , JerkFactor: = , Done=> , Busy=> , Active=> , CommandAborted=> , CommandAccepted=> , Error=> , ErrorID=> , MovementId=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
CircMode	圆弧模式	SMC_CIRC_MODE	0-2	0	指定圆弧插补的方法，见“SMC_CIRC_MODE”说明。 0： _mcBorder，三点圆弧； 1： _mcCenter，中心圆弧； 2： _mcRadius，半径圆弧
AuxPoint	辅助点	SMC_POS_REF	负数，0 正数	0	辅助点位置，见“SMC_CIRC_MODE”说明
EndPoint	终点	SMC_POS_REF	负数，0 正数	0	终点位置，见“SMC_CIRC_MODE”说明
PathChoice	方向	MC_CIRC_PATHCHOICE	0，1	0	运动方向，0：顺时针；1：逆时针

Velocity	速度	LREAL	0，正数	0	最大合速度[u/s]，只能输入正数
Acceleration	加速度	LREAL	0，正数	0	最大合加速度 [u/s²]，只能输入正数
Deceleration	减速度	LREAL	0，正数	0	最大合减速度 [u/s²]，只能输入正数
Jerk	加加速度	LREAL	0，正数	0	最大合加加速度 [u/s²]，只能输入正数
CoordSystem	参考坐标系	SMC_COORD_SYSTEM	SMC_COORD_SYSTEM	-	参考坐标系
BufferMode	缓存模式	MC_BUFFER_MODE	0-5	0	速度交接模式，请参考 7.1 节说明
TransitionMode	拐角过渡模式	MC_TRANSITION_MODE	TMNone，TMStartVelocity，TMCornerDistance	0	拐角过渡模式，请参考 7.1 节说明
TransitionParameter	拐角过渡参数	ARRAY [0..(SMC_RCNST.MAX_TRANS_PARAMS - 1)] OF LREAL	0，正数	0	拐角过渡参数
OrientationMode	插补定位模式	SMC_ORIENTATION_MODE	GreatCircle，Axis	0	插补定位模式
VelFactor	速度因子	LREAL	0-1	1	速度因子，每个轴的最大速度乘以这个速度因子，数值在[0，1]之间
AccFactor	加速度因子	LREAL	0-1	1	加速度因子，每个轴的最大加速度乘以这个加速度因子，数值在[0，1]之间
JerkFactor	加加速度因子	LREAL	0-1	1	加加速度因子，每个轴的最大加加速度乘以这个加加速度因子，数值在[0，1]之间
VAR_OUTPUT					
Done	完成	BOOL	TRUE，FALSE	FALSE	指令执行完成时为 TRUE
Busy	运动中	BOOL	TRUE，FALSE	FALSE	指令正在执行中为 TRUE
Active	启动中	BOOL	TRUE，FALSE	FALSE	当轴正在被这个模块调用时为 TRUE
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	模块执行被中断为 True
CommandAccepted	运动接收	BOOL	TRUE，FALSE	FALSE	当模块成功调用轴组时为 True
Error	报错	BOOL	TRUE，FALSE	FALSE	功能块执行错误为 True
ErrorID	错误码 ID	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
MovementId	运动标志	SMC_Movement_Id	TRUE，FALSE	FALSE	当运动正在执行或完成，则为 True

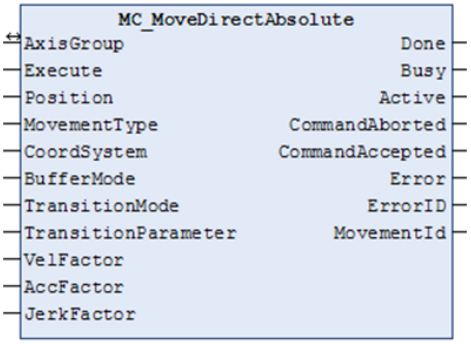
功能说明：

- Execute 的上升沿触发指令运动，下降沿对指令运动没有影响。
- 当使用圆心圆弧模式时，输入参数 AuxPoint[1]、AuxPoint[2]的数值为圆心与圆弧起点的距离；而当使用半径圆弧模式时，AuxPoint[1]表示半径数值，AuxPoint[2]无效。
- EndPoint[1]~EndPoint[8]的数值表示各轴的终点坐标。
- 时序图和 MC_MoveDirectAbsolute 指令相同。

绝对位置快速定位 MC_MoveDirectAbsolute

控制轴组内所有轴各自以指定速度运行到绝对位置终点。

指令外观：

指令	FB/ FUN	图形模块	结构文本
MC_MoveDirectAbsolute	FB		MC_MoveDirectAbsolute (AxisGroup: = , Execute: = , Position: = , MovementType: = , CoordSystem: = , BufferMode: = , TransitionMode: = , TransitionParameter: = , VelFactor: = , AccFactor: = , JerkFactor: = , Done=> , Busy=> , Active=> , CommandAborted=> , CommandAccepted=> , Error=> , ErrorID=> , MovementId=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
Position	位置	SMC_POS_REF	-	-	在指定参考坐标系中的绝对目标位置

MovementType	PTP 运动模式	SMC_PTP_MOVEMENT_TYPE	-	-	Fast（0）：时间优先的 PTP 运动模式 Path_Invariant：路径固定的 PTP 运动
CoordSystem	参考坐标系	SMC_COORD_SYSTEM	-	-	参考坐标系
BufferMode	缓存模式	MC_BUFFER_MODE	0-5	0	速度交接模式，请参考 7.1 节相关说明
TransitionMode	拐角过渡模式	MC_TRANSITION_MODE	TMNone，TMStartVelocity，TMCornerDistance	0	拐角过渡模式，请参考 7.1 节相关说明
TransitionParameter	拐角过渡参数	ARRAY [0..(SMC_RCNST.MAX_TRANS_PARAMS - 1)] OF LREAL	0，正数	0	拐角过渡参数
VelFactor	速度因子	LREAL	0-1	1	速度因子，每个轴的最大速度乘以这个速度因子，数值在[0，1]之间
AccFactor	加速度因子	LREAL	0-1	1	加速度因子，每个轴的最大加速度乘以这个加速度因子，数值在[0，1]之间
JerkFactor	加加速度因子	LREAL	0-1	1	加加速度因子，每个轴的最大加加速度乘以这个加加速度因子，数值在[0，1]之间
VAR_OUTPUT					
Done	完成	BOOL	TRUE，FALSE	FALSE	指令执行完成时为 TRUE
Busy	运动中	BOOL	TRUE，FALSE	FALSE	指令正在执行中为 TRUE
Active	启动中	BOOL	TRUE，FALSE	FALSE	当轴正在被这个模块调用时为 TRUE
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	模块执行被中断为 True
CommandAccepted	运动接收	BOOL	TRUE，FALSE	FALSE	当模块成功调用轴组时为 True
Error	报错	BOOL	TRUE，FALSE	FALSE	功能块执行错误为 True
ErrorID	错误码 ID	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
MovementId	运动标志	SMC_Movement_Id	TRUE，FALSE	FALSE	当运动正在执行或完成，则为 True

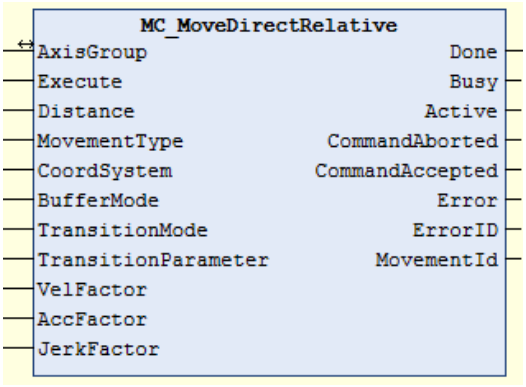
功能说明：

- 运动中各轴是相对独立的绝对位置运动，其运动轨迹不确定。
- 时序图和 MC_MoveDirectAbsolute 指令相同。

相对位置快速定位 MC_MoveDirectRelative

控制轴组内所有轴各自以指定速度运行到相对位置的终点。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_MoveDirectRelative	FB		MC_MoveDirectRelative(AxisGroup:= , Execute:= , Distance:= , MovementType:= , CoordSystem:= , BufferMode:= , TransitionMode:= , TransitionParameter:= , VelFactor:= , AccFactor:= , JerkFactor:= , Done=> , Busy=> , Active=> , CommandAborted=> , CommandAccepted=> , Error=> , ErrorID=> , MovementId=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
Distance	相对位置	SMC_POS_REF	-	-	在指定参考坐标系中的相对目标位置
MovementType	PTP 运动模式	SMC_PTP_MOVEMENT_TYPE	-	-	Fast（0）：时间优先的 PTP 运动模式 Path_Invariant：路径固定的 PTP 运动
CoordSystem	参考坐标系	SMC_COORD_SYSTEM	-	-	参考坐标系
BufferMode	缓存模式	MC_BUFFER_MODE	0-5	0	速度交接模式，请参考 5.1.6.5 节相关说明
TransitionMode	拐角过渡模式	MC_TRANSITION_MODE	TMNone，TMStartVelocity，TMCornerDistance	0	拐角过渡模式，请参考 5.1.6.5 节相关说明

TransitionParameter	拐角过渡参数	ARRAY [0..(SMC_RCNST.MAX_TRANS_PARAMS - 1)] OF LREAL	0，正数，	0	拐角过渡参数
VelFactor	速度因子	LREAL	0-1	1	速度因子，每个轴的最大速度乘以这个速度因子，数值在[0，1]之间
AccFactor	加速度因子	LREAL	0-1	1	加速度因子，每个轴的最大加速度乘以这个加速度因子，数值在[0，1]之间
JerkFactor	加加速度因子	LREAL	0-1	1	加加速度因子，每个轴的最大加加速度乘以这个加加速度因子，数值在[0，1]之间
VAR_OUTPUT					
Done	完成	BOOL	TRUE，FALSE	FALSE	指令执行完成时为 TRUE
Busy	运动中	BOOL	TRUE，FALSE	FALSE	指令正在执行中为 TRUE
Active	启动中	BOOL	TRUE，FALSE	FALSE	当轴正在被这个模块调用时为 TRUE
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	模块执行被中断为 True
CommandAccepted	运动接收	BOOL	TRUE，FALSE	FALSE	当模块成功调用轴组时为 True
Error	报错	BOOL	TRUE，FALSE	FALSE	功能块执行错误为 True
ErrorID	错误码 ID	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
MovementId	运动标志	SMC_Movement_Id	TRUE，FALSE	FALSE	当运动正在执行或完成，则为 True

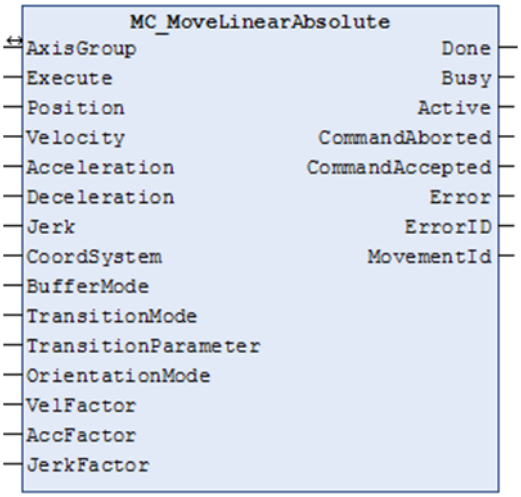
功能说明：

- 运动中各轴是独立的相对位置运动，其运动轨迹不确定。
- 时序图和 MC_MoveDirectAbsolute 指令相同。

绝对位置直线插补 MC_MoveLinearAbsolute

控制轴组在指定坐标系下的绝对位置模式的直线插补运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_MoveLinearAbsolute	FB		MC_MoveLinearAbsolute(AxisGroup: = , Execute: = , Position: = , Velocity: = , Acceleration: = , Deceleration: = , Jerk: = , CoordSystem: = , BufferMode: = , TransitionMode: = , TransitionParameter: = , OrientationMode: = , VelFactor: = , AccFactor: = , JerkFactor: = , Done=> , Busy=> , Active=> , CommandAborted=> , CommandAccepted=> , Error=> , ErrorID=> , MovementId=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
Position	位置	SMC_POS_REF	负数，0，正数	0	在指定参考坐标系中的绝对目标位置
Velocity	速度	LREAL	0，正数	0	最大合速度[u/s]，只能输入正数
Acceleration	加速度	LREAL	0，正数	0	最大合加速度 [u/s²]，只能输入正数
Deceleration	减速度	LREAL	0，正数	0	最大合减速度 [u/s²]，只能输入正数
Jerk	加加速度	LREAL	负数，0 正数	0	最大合加加速度 [u/s²]，只能输入正数
CoordSystem	参考坐标系	SMC_COORD_SYSTEM	SMC_COORD_SYSTEM	-	参考坐标系
BufferMode	缓存模式	MC_BUFFER_MODE	0-5	0	速度交接模式，请参考 5.1.6.5 节相关说明

TransitionMode	拐角过渡模式	MC_TRANSITION_MODE	TMNone，TMStartVelocity，TMCornerDistance	0	拐角过渡模式，请参考 5.1.6.5 节相关说明
TransitionParameter	拐角过渡参数	ARRAY [0..(SMC_RCNST.MAX_TRANS_PARAMS - 1)] OF LREAL	0，正数	0	拐角过渡参数
OrientationMode	插补定位模式	SMC_ORIENTATION_MODE	GreatCircle，Axis	0	插补定位模式，请参考 5.1.6.5 节相关说明
VelFactor	速度因子	LREAL	0-1	1	速度因子，每个轴的最大速度乘以这个速度因子，数值在[0，1]之间
AccFactor	加速度因子	LREAL	0-1	1	加速度因子，每个轴的最大加速度乘以这个加速度因子，数值在[0，1]之间
JerkFactor	加加速度因子	LREAL	0-1	1	加加速度因子，每个轴的最大加加速度乘以这个加加速度因子，数值在[0，1]之间
VAR_OUTPUT					
Done	完成	BOOL	TRUE，FALSE	FALSE	指令执行完成时为 TRUE
Busy	运动中	BOOL	TRUE，FALSE	FALSE	指令正在执行中为 TRUE
Active	启动中	BOOL	TRUE，FALSE	FALSE	当轴正在被这个模块调用时为 TRUE
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	模块执行被中断为 True
CommandAccepted	运动接收	BOOL	TRUE，FALSE	FALSE	当模块成功调用轴组时为 True
Error	报错	BOOL	TRUE，FALSE	FALSE	功能块执行错误为 True
ErrorID	错误码 ID	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
MovementId	运动标志	SMC_Movement_Id	TRUE，FALSE	FALSE	当运动正在执行或完成，则为 True

功能说明：

- 控制轴组在指定坐标系下的绝对位置模式的直线插补运动。
- 参数 BufferMode、TransitionMode 和 TransitionParameter 之间的关系说明如下：

1）当 BufferMode 选择 mcBuffered 模式时，TransitionMode 仅支持 mcTMNone 模式；

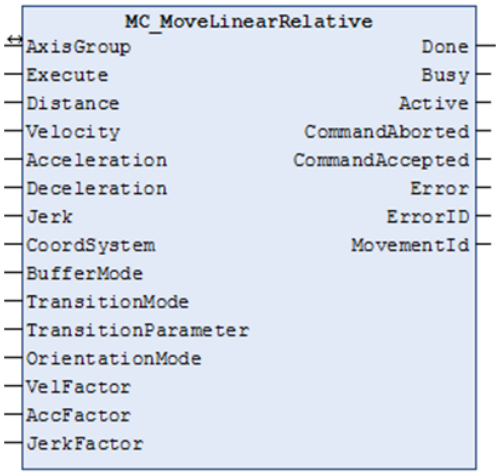
2）当 BufferMode 选择 mcBlendingPrevious 模式时，TransitionMode 可选择 mcTMConstantVelocity 和 mcTMCornerDistance 模式。

- 时序图和 MC_MoveDirectAbsolute 指令相同。

相对位置直线插补 MC_MoveLinearRelative

控制轴组在指定坐标系下的相对位置模式的直线插补运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
MC_MoveLinearRelative	FB		MC_MoveLinearRelative (AxisGroup: = , Execute: = , Distance: = , Velocity: = , Acceleration: = , Deceleration: = , Jerk: = , CoordSystem: = , BufferMode: = , TransitionMode: = , TransitionParameter: = , OrientationMode: = , VelFactor: = , AccFactor: = , JerkFactor: = , Done=> , Busy=> , Active=> , CommandAborted=> , CommandAccepted=> , Error=> , ErrorID=> , MovementId=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
Position	位置	SMC_POS_REF	负数，0，正数	0	在指定参考坐标系中的绝对目标位置
Velocity	速度	LREAL	0，正数	0	最大合速度[u/s]，只能输入正数
Acceleration	加速度	LREAL	0，正数	0	最大合加速度 [u/s²]，只能输入正数
Deceleration	减速度	LREAL	0，正数	0	最大合减速度 [u/s²]，只能输入正数
Jerk	加加速度	LREAL	负数，0 正数	0	最大合加加速度 [u/s²]，只能输入正数

CoordSystem	参考坐标系	SMC_COORD_SYSTEM	SMC_COORD_SYSTEM	-	参考坐标系
BufferMode	缓存模式	MC_BUFFER_MODE	0-5	0	速度交接模式，请参考 5.1.6.5 节相关说明
TransitionMode	拐角过渡模式	MC_TRANSITION_MODE	TMNone，TMStartVelocity，TMCornerDistance	0	拐角过渡模式，请参考 5.1.6.5 节相关说明
TransitionParameter	拐角过渡参数	ARRAY [0..(SMC_RCNST.MAX_TRANS_PARAMS - 1)] OF LREAL	0，正数	0	拐角过渡参数
OrientationMode	插补定位模式	SMC_ORIENTATION_MODE	GreatCircle，Axis	0	插补定位模式，请参考 5.1.6.5 节相关说明
VelFactor	速度因子	LREAL	0-1	1	速度因子，每个轴的最大速度乘以这个速度因子，数值在[0，1]之间
AccFactor	加速度因子	LREAL	0-1	1	加速度因子，每个轴的最大加速度乘以这个加速度因子，数值在[0，1]之间
JerkFactor	加加速度因子	LREAL	0-1	1	加加速度因子，每个轴的最大加加速度乘以这个加加速度因子，数值在[0，1]之间
VAR_OUTPUT					
Done	完成	BOOL	TRUE，FALSE	FALSE	指令执行完成时为 TRUE
Busy	运动中	BOOL	TRUE，FALSE	FALSE	指令正在执行中为 TRUE
Active	启动中	BOOL	TRUE，FALSE	FALSE	当轴正在被这个模块调用时为 TRUE
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	模块执行被中断为 True
CommandAccepted	运动接收	BOOL	TRUE，FALSE	FALSE	当模块成功调用轴组时为 True
Error	报错	BOOL	TRUE，FALSE	FALSE	功能块执行错误为 True
ErrorID	错误码 ID	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
MovementId	运动标志	SMC_Movement_Id	TRUE，FALSE	FALSE	当运动正在执行或完成，则为 True

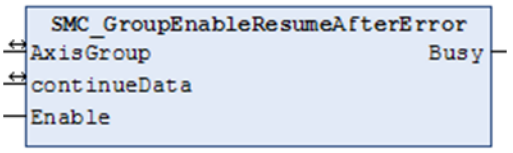
功能说明:

- 只有当 TransitionMode 为 mcTMCornerDistance 时，参数 TransitionParameter 才有效。
- 时序图和 MC_MoveDirectAbsolute 指令相同。

错误复位后重启 SMC_GroupEnableResumeAfterError

恢复因报错而被中断的轴组状态。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupEnableResumeAfterError	FUN		SMC_GroupEnableResumeAfterError(AxisGroup: = , continueData: = , Enable: = , Busy=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	解释
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
ContinueData	继续运动数据	SMC_AXIS_GROUP_CONTINUE_DATA	-	-	运动中断时的轴组位置
VAR_INPUT					
Enable	启用	BOOL	TRUE FALSE	FALSE	启用指令功能
VAR_OUTPUT					
Busy	调用中	BOOL	TRUE FALSE	FALSE	指令正在被调用时为 True

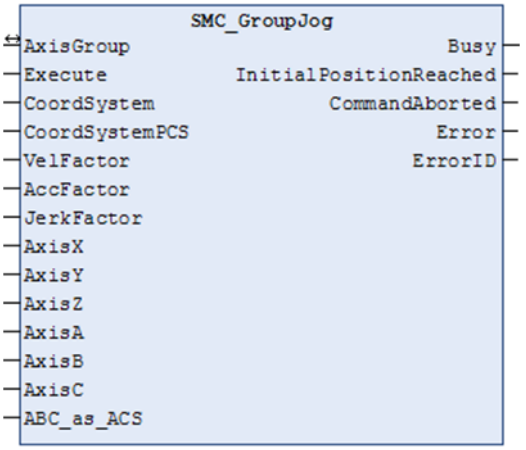
功能说明：

- 恢复因为轴报错而被打断的轴组运动，参数 ContinueData 用来存储指令中断时轴组的运动信息。
- 首先用 MC_GroupReset 指令将轴组报错进行复位；再次用 SMC_GroupGetContinuePosition 获取轴组运动中断时的位置并移动到该位置；最后，调用 MC_GroupContinue 指令恢复被中断的轴组运动。

轴组点动 SMC_GroupJog

控制轴组在指定坐标系下进行 Jog 运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupJog	FB		SMC_GroupJog(AxisGroup: = , Execute: = , CoordSystem: = , CoordSystemPCS: = , VelFactor: = , AccFactor: = , JerkFactor: = , AxisX: = , AxisY: = , AxisZ: = , AxisA: = , AxisB: = , AxisC: = , ABC_as_ACS: = , Busy=> , InitialPositionReached=> , CommandAborted=> , Error=> , ErrorID=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
CoordSystem	参考坐标系	SMC_COORD_SYSTEM	-	-	参考坐标系
CoordSystemPCS	PCS 坐标系	SMC_COORD_SYSTEM	-	-	内部用于点动的坐标系为 PCS，需要时使用 SMC_SetDynCoordTransformEx 进行更改
VelFactor	速度因子	LREAL	0-1	1	速度因子，每个轴的最大速度乘以这个速度因子，数值在[0，1]之间
AccFactor	加速度因子	LREAL	0-1	1	加速度因子，每个轴的最大加速度乘以这个加速度因子，数值在[0，1]之间
JerkFactor	加加速度因子	LREAL	0-1	1	加加速度因子，每个轴的最大加加速度乘以这个加加速度因子，数值在[0，1]之间
AxisX	轴 X	IAxisRef	-	0	坐标系里的 X 轴，如果没有用到该轴，则设置为 0
AxisY	轴 Y	IAxisRef	-	0	坐标系里的 Y 轴，如果没有用到该轴，则设置为 0

AxisZ	轴 Z	IAxisRef	-	0	坐标系里的 Z 轴，如果没有用到该轴，则设置为 0
AxisA	轴 A	IAxisRef	-	0	坐标系里的 A 轴（X 轴的旋转轴），或者第一个跟随轴，如果没有用到该轴，则设置为 0
AxisB	轴 B	IAxisRef	-	0	坐标系里的 B 轴（Y 轴的旋转轴），或者第二个跟随轴，如果没有用到该轴，则设置为 0
AxisC	轴 C	IAxisRef	-	0	坐标系里的 C 轴（Z 轴的旋转轴），或者第三个跟随轴，如果没有用到该轴，则设置为 0
ABC_as_ACS	启动坐标转化	BOOL	TRUE FALSE	FALSE	输入 True，则轴 A、B、C 的位置将转化成为工具运动学的轴目标位置，否则将表示 ZYZ 的方向(在 ACS 下不会进行转换)。另外轴组的运动学变换必须是 Kin_Coupled，且不支持 SMC_ORIENTATION_MODE.Axis
VAR_OUTPUT					
Busy	执行中	BOOL	TRUE，FALSE	FALSE	指令正在执行中为 TRUE
InitialPositionReached	已接收到位置	BOOL	TRUE，FALSE	FALSE	当接收到目标位置时为 True
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	模块执行被中断为 True
Error	报错	BOOL	TRUE，FALSE	FALSE	功能块执行错误为 True
ErrorID	错误码 ID	SMC_ERROR	-	0	错误代码指示，见 SMC_Error

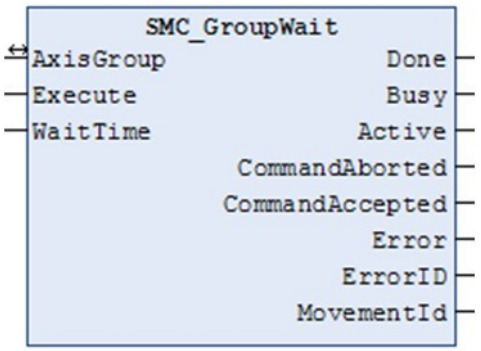
功能说明：

- 轴组进行 JOG 运动（点动）时，将根据配置轴给出的位置和方向进行运动。
- 指令会将输入坐标系转化成目标坐标系进行输出，比如在使用 MCS 的机器坐标系中的 JOG 运动。
- 设置了 ABC_as_ACS 参数即是使用了参考轴的混合转化，其中 X/Y/Z 作为笛卡尔坐标系的位置，A/B/C 作为工具坐标系的位置。
- 产品坐标系被用在位置实时变化的情况下，能通过配置输入坐标系 CoordSystemPCS 实现，且需要标记轴组在坐标系中的实时变化。
- 在重新启动这个 FB 而不改变 CoordSystemPCS 的情况下，需要添加一个其他的移动指令，比如 MC_GroupHalt。如果不遵循这种使用原则，将会返回 SMC_AXIS_GROUP_PCS_STILL_IN_USE 的报错。

轴组运动延时 SMC_GroupWait

设定轴组的延时等待。

指令外观：

指令	FB/FUN	图形模块	结构文本
SMC_GroupWait	FB		SMC_GroupWait(AxisGroup: = , Execute: = , WaitTime: = , Done=> , Busy=> , Active=> , CommandAborted=> , CommandAccepted=> , Error=> , ErrorID=> , MovementId=>);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
AxisGroup	轴组	AXIS_GROUP_REF_SM3	-	-	指定的轴组
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	执行当前指令
WaitTime	等待时间	LREAL	0，正数	0	轴组运动的等待时间，单位秒（s），≥0
VAR_OUTPUT					
Done	完成	BOOL	TRUE，FALSE	FALSE	指令执行完成时为 TRUE
Busy	运动中	BOOL	TRUE，FALSE	FALSE	指令正在执行中为 TRUE
Active	启动中	BOOL	TRUE，FALSE	FALSE	当轴正在被这个模块调用时为 TRUE
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	模块执行被中断为 True
CommandAccepted	运动接收	BOOL	TRUE，FALSE	FALSE	当模块成功调用轴组时为 True
Error	报错	BOOL	TRUE，FALSE	FALSE	功能块执行错误为 True
ErrorID	错误码 ID	SMC_ERROR	-	0	错误代码指示，见 SMC_Error
MovementId	运动标志	SMC_Movement_Id	TRUE，FALSE	FALSE	当运动正在执行或完成，则为 True

功能说明：

- 如果两个相邻指令之间的等待时间和任务配置的时间相同，则两个运动之间的实际等待时间可通过 SMC_GroupWait 指令变得更少，比如：一般下个指令的执行通常是从任务周期的下个循环开始的，但如果中间有个 SMC_GroupWait 指令，则在 SMC_GroupWait 指令延时结束后会立即开始下个运动指令，而不需要等待任务周期的下一次循环的开始。
- 如果跟踪移动后有等待命令，则轴组将在指定的时间内跟踪上一个移动的终点。
- 如果各个轴不处于 Standstill 状态，但不受轴组控制，等待被调用，轴组将报错 SMC_AXIS_GROUP_IDLE_WAIT_AXES_MOVING。
- 跟任务周期每循环更新一次不同，SMC_GroupWait 指令的时间如果与总线周期时间的倍数不同，也能够在等待时间后立即下个指令，让后续移动更加平滑的启动。
- 由于技术原因，在以下情况下，等待时间最多可以增加一个周期： - non-tracking -> waiting -> tracking - tracking -> waiting -> non-tracking - tracking -> waiting -> PTP-tracking。

7.3 轴组例程

本节通过几个例程讲述轴组运动的具体应用方法。

第一部分例程是基于笛卡尔三轴坐标系的运动模型，介绍编程软件中的“空间类型轴组 Kin_Gantry3”模型如何建模和配置参数、如何调用指令及指令的执行顺序，从而实现连续插补运动。

第二部分例程则分别是基于 SCARA 机器人、Delta 机器人、六自由度机器人和五轴机器人模型，介绍编程软件如何建模和配置参数、如何调用指令及指令的执行顺序，从而实现连续插补运动。

5 个例程的主要差别是模型不同，因此建模与模型参数配置不同。建模完成后，其调用的运动指令、运动指令的执行顺序、使用方式，基本上一致。

7.3.1 连续插补运动例程

连续插补运动是机械坐标系下执行末端（TCP）的多段运动轨迹，通过在线段间增加过渡曲线等方式实现速度曲线的连续。与采用单段插补指令实现的轨迹相比，可提高轨迹的速度，让两段插补曲线过渡得更加平滑。

使用轴组功能实现连续插补运动，可以总结为以下三个步骤：

第一步：建立连续插补轴组，配置运动学模型

连续插补的轴组根据轴数确定具体的轴组类型，一般分为两种：

（1）平面类型的连续插补轴组，选择 TRAFO.Kin_Gantry2 运动学模型；

（2）空间类型的连续插补轴组，选择 TRAFO.Kin_Gantry3 运动学模型。

具体方法如下：

使用轴组的配置工具，建立执行连续插补运动的轴组（以空间类型轴组 Kin_Gantry3 为例）。首先在工程中，右键单击工程“设备窗口”中的“Application”，选择“添加对象”，然后点击添加“轴组”，并将轴组命名为 AxisGroup。通过“运动学模型”界面，选择“TRAFO.Kin_Gantry3 ”运动学模型，如图 7.8 所示。

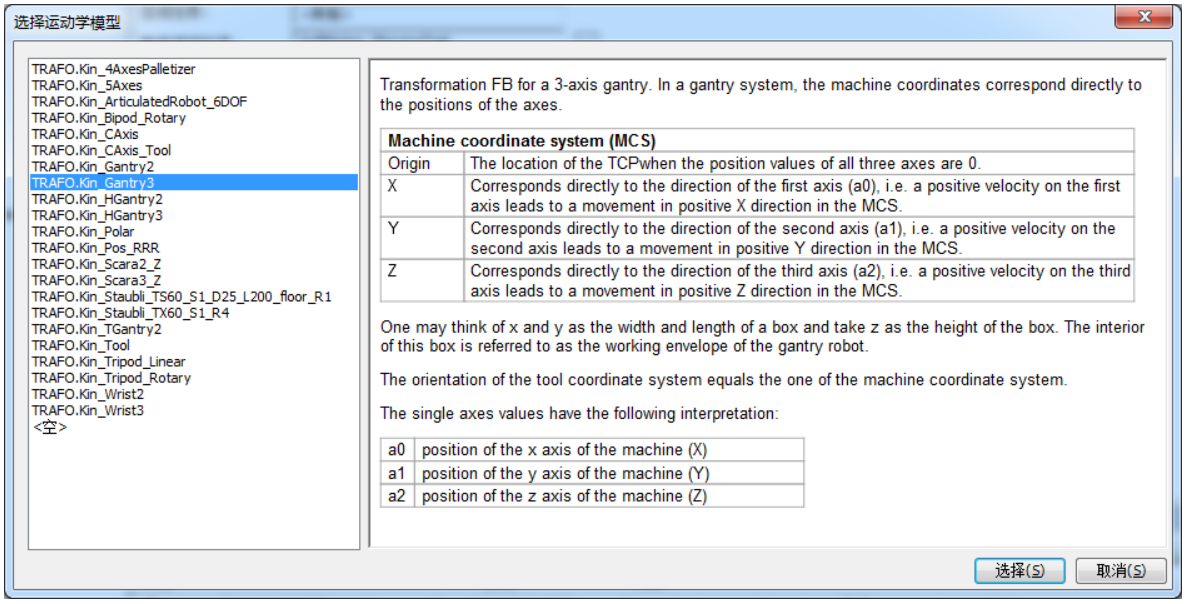


图 7.8　运动学模型界面

然后将“TRAFO.Kin_Gantry3 ”运动学模型的 XYZ 3 轴分别映射为实际轴接口的 X、Y、Z 三轴，这三个轴可以是脉冲轴，也可以是总线轴。最后再将轴组规划任务设置为“Softmotion_PlanningTask”，如图 7.9 所示。

图 7.9　轴组配置

第二步：初始化轴组

轴组模块的初始化包括三个部分，分别为：

（1）启用轴组，如图 7.10 所示。

（2）坐标系转换，如图 7.11 所示，将轴组所在的机械坐标系（MCS），转化为所选择的机械结构的机械坐标系（MCS），这样，才能够计算出转化后坐标系内各个方向轴的位置和速度等参数。由于这里所选择的连续插补 3 轴机械坐标系与轴组的机械坐标系是一样的，因此，在这里也可以不需要进行坐标系的转换，可以直接引用原来 XYZ 三轴的位置和

速度等参数。如果是在两个坐标系不一样的情况下，则需要通过坐标系的转换，才能获取转换后坐标系的运动参数。

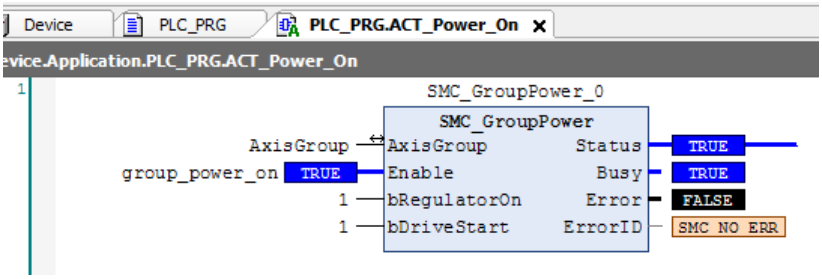


图 7.10　启动轴组

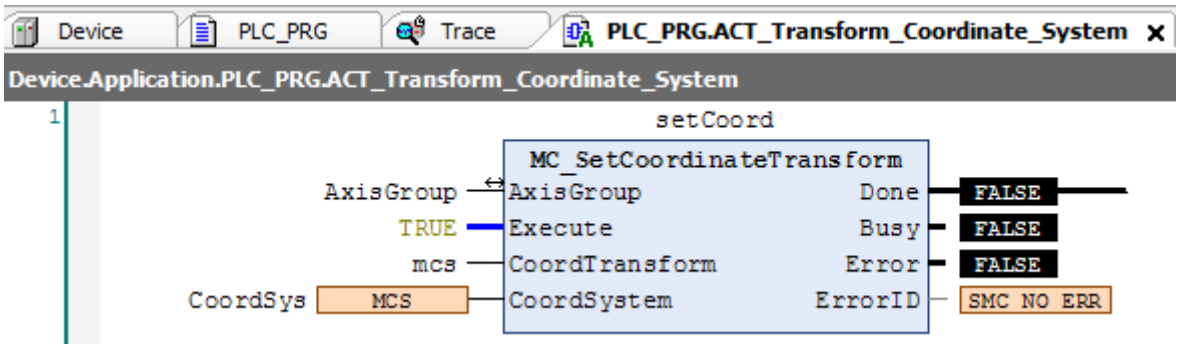


图 7.11　设置坐标系

（3）使能轴组，如图 7.12 所示。

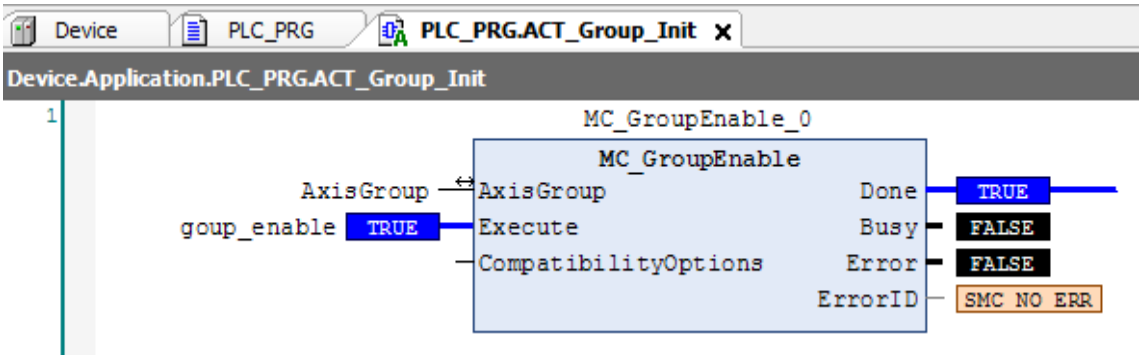


图 7.12　使能轴组

调用两条该指令，将轴组状态机切换为 StandBy，如图 7.13 所示。

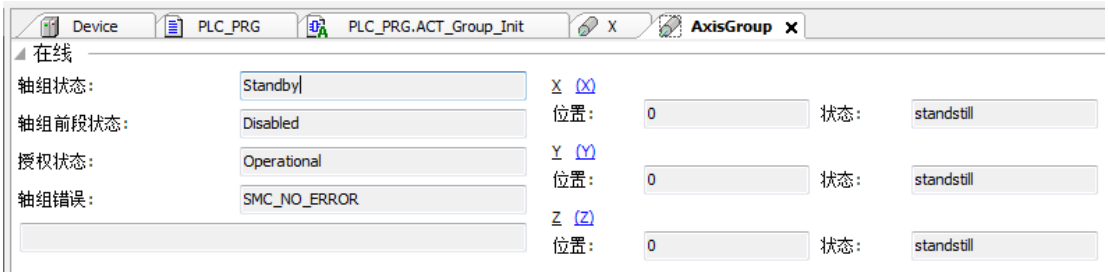


图 7.13 轴组状态机切换

第三步：调用插补指令，配置轨迹过渡指令参数

根据执行末端（TCP）的运动轨迹，调用轴组的直线、圆弧插补指令组成该轨迹，并配置合理的拐角参数。其具体参数意义说明，请参考本手册 7.2.6 轴组运动指令。在程序中，将 action_move 信号设置为 TRUE，即可实现轴组的连续插补运动。

图 7.14 为 Trace 界面上显示的运动轨迹。浅蓝色线为 Y 轴的速度曲线，显然在两条线段的交点 P，Y 轴实现了速度的连续过渡。

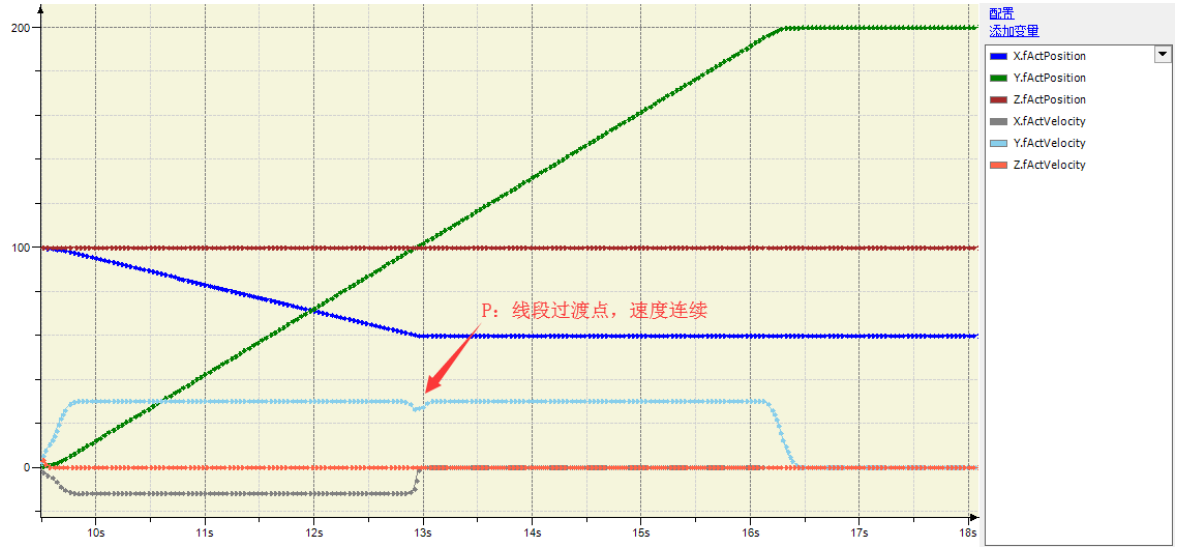


图 7.14　Trace 运动轨迹

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“连续插补”。

7.3.2　机器人运动例程

1. SCARA 机器人运动建模及例程

1）两个节点的 SCARA 机器人结构

（1）机械结构分析

该类机械结构类似人类的手臂，包括了机座、大臂、小臂、以及执行末端；其中大臂和小臂被限制在 X-Y 平面。而 Z 轴的运动由执行末端完成，与大小臂运动无关，因此其运动学模型如图 7.15 所示。

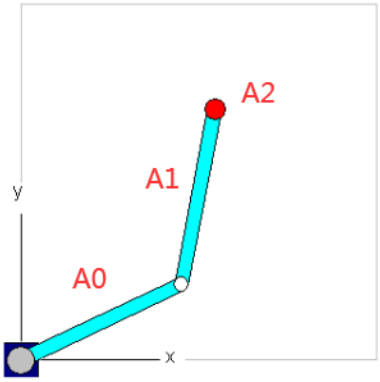


图 7.15　两节点 SCARA 机械人运动学模型

该类机械结构核心组件：

Ⅰ：大臂旋转轴 A0，围绕 Z 轴旋转；

Ⅱ：小臂旋转轴 A1，跟随大臂旋转同时自身旋转；

Ⅲ：执行末端的直线轴 A2，确定在 Z 轴坐标位置。

（2）建立坐标系

首先，建立基于 SCARA 机械手的机械坐标系（MCS）并确定原点。原点位置为大臂 A0 轴转动中心及 X、Y 轴的交点，其中 X 轴为大臂 A0 轴旋转时的 0 度位置；Y 轴为大臂 A0 轴旋转时的 90 度位置；Z 轴为执行末端直线运动时，其垂直 X-Y 平面的方向。

其次建立工具坐标系（TCS），相对于机械坐标系的偏移量及绕 Z 轴的旋转角度。

（3）配置运动控制参数

首先：配置机构参数：

大臂（第一个节点）长度：dArmLenth1（单位 unit）

小臂（第二个节点）长度：dArmLength2（单位 unit）

大臂与小臂间姿态设置：dElbowRight

其次：配置坐标系参数

A1 轴的偏移量：dOffsetA1（单位度）

A2 轴的偏移量：dOffsetA2（单位度）

Z 轴的位置偏移量：dOffsetZ（单位 unit）

2） 三个节点的 SCARA 机器人结构

（1）机械结构分析

该类机械结构类似人类的手臂，包括了机座、大臂、小臂、第三杆、以及执行末端；其中大臂和小臂及第三杆被限制在 X-Y 平面。而 Z 轴的运动由执行末端完成，与大小臂及第三杆运动无关，因此其运动学模型如图 7.16 所示。

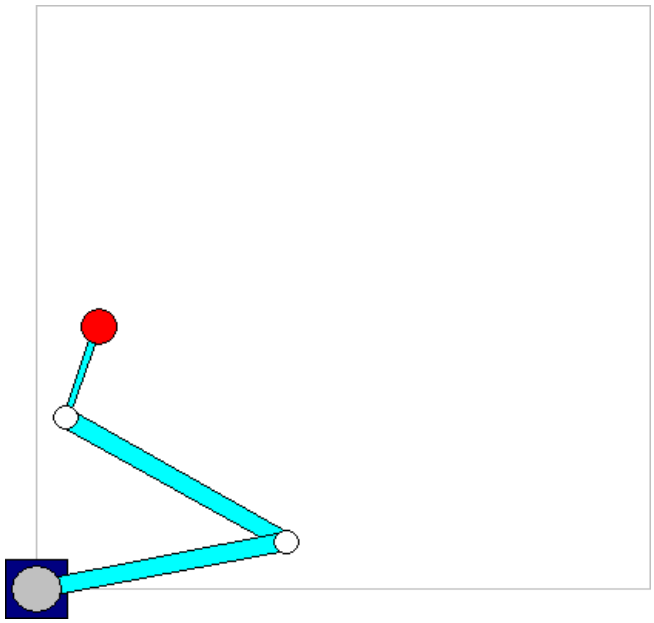


图 7.16　三节点 SCARA 机械人运动学模型

该类机械结构核心组成：

Ⅰ：大臂旋转轴 A0，围绕 Z 轴旋转；

Ⅱ：小臂旋转轴 A1，跟随大臂旋转同时自身旋转；

Ⅲ：第三杆旋转轴 A2，跟随小臂旋转同时自身旋转；

Ⅳ：执行末端的直线轴 A3，确定在 Z 轴坐标位置。

（2）建立坐标系

如图 7.16 所示，机械坐标系（MCS）及原点与两节点 SCARA 机械人运动学模型相同。

建立工具坐标系（TCS）相对于机械坐标系的偏移量及绕 Z 轴的旋转角度。

（3）配置运动控制参数

机构参数：

大臂（第一个节点）长度：dArmLenth1（单位 unit）

小臂（第二个节点）长度：dArmLength2（单位 unit）

第三杆（第三个节点）长度：dArmLength3（单位 unit）

大臂与小臂间姿态设置：dElbowRight

小臂与第三杆间姿态设置：dElbowRight1

坐标系参数：

A1 轴的偏移量：dOffsetA1（单位度）

A2 轴的偏移量：dOffsetA2（单位度）

A3 轴的偏移量：dOffsetA3（单位度）

Z 轴的位置偏移量：dOffsetZ（单位 unit）

3） 例程

本例程介绍使用系统仿真实现两个节点的 SCARA 系统的运动控制。编程可分为以下两部分：

（1）建模及配置参数：

第一步：添加控制系统需要的控制轴数；

第二步：使用编程软件提供的建模工具，建立运动学模型；

第三步：配置运动学参数。

（2）运动控制编程：

第四步：初始化轴组模块

第五步：根据运动轨迹，编写运动学指令，配置运动参数；

详细过程如下：

第一步：添加控制系统需要的控制轴数

两个节点的 SCARA 系统为两个旋转轴和一个直线轴，执行末端没有绕 Z 的旋转运动。因此需要添加三个虚拟轴，如图 7.17 所示。最后设置虚拟轴名称分别为 Axis0、Axis1 以及 Axis2。

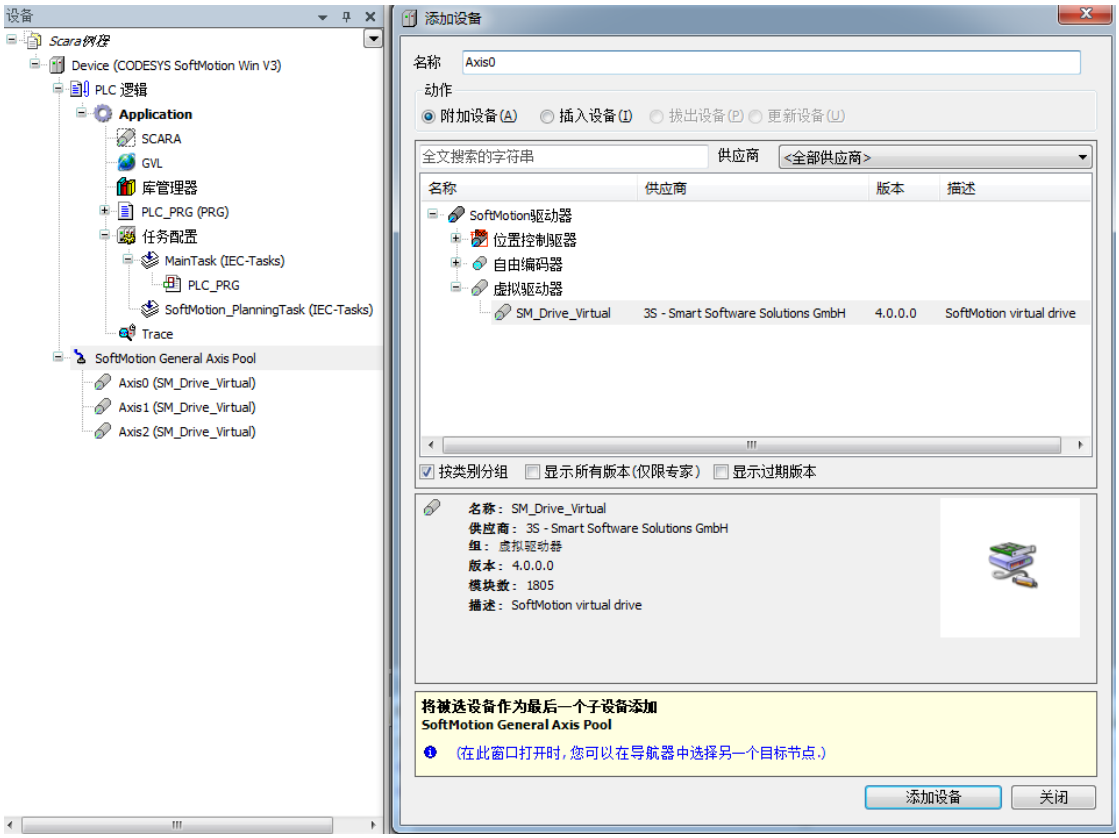


图 7.17　添加虚拟轴

第二步：使用编程软件提供的建模工具，建立运动学模型

右键单击工程中的“Application”，通过“添加对象”，选择添加“轴组”，并将轴组命名为 SCARA。通过“运动学模型”界面选择 SCARA 运动学模型，如图 7.18 所示。

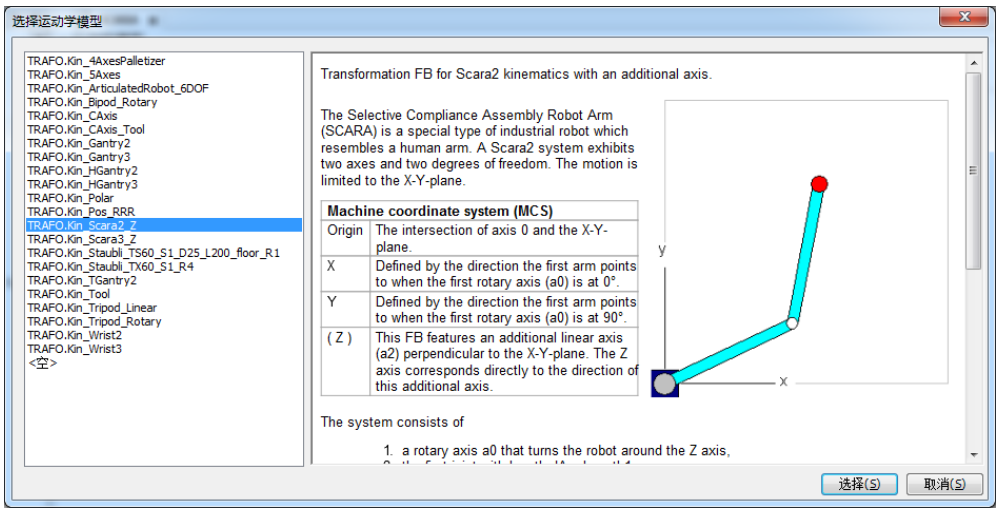


图 7.18　建立运动学模型

第三步：配置运动学参数

结构参数配置：大小臂长度都为 1000，角度偏移量全为零。

轴参数配置：在映射轴位置，配置 A1 映射为 Axis0，A2 映射为 Axis1，Z 轴映射为 Axis2；同时根据实际的转动比，配置轴参数。

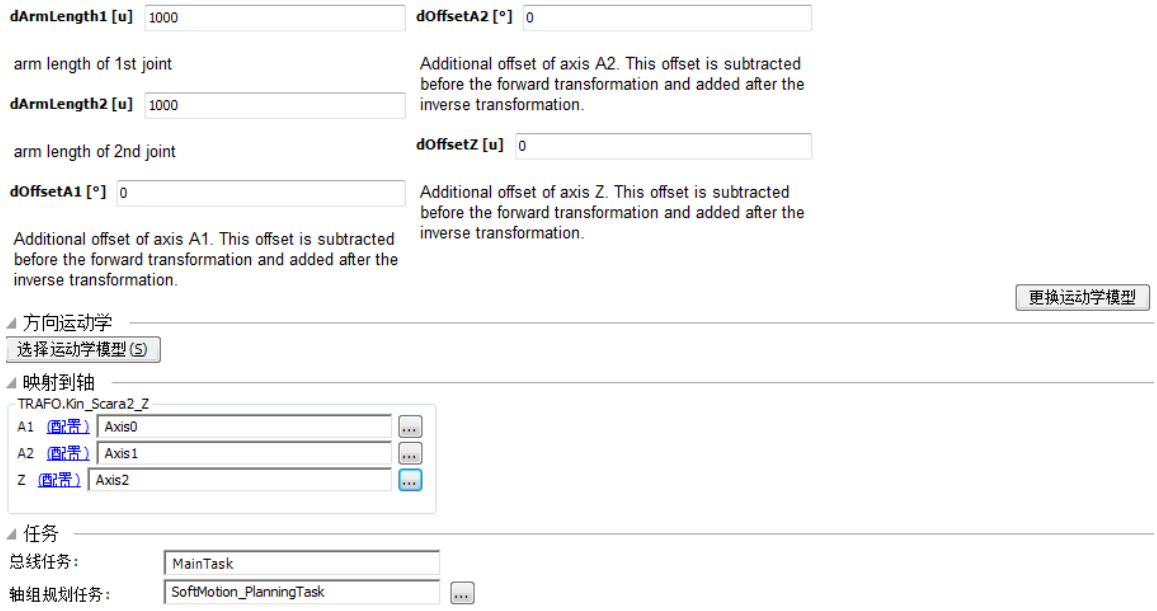


图 7.19　配置运动学参数

第四步：初始化轴组模块

轴组模块的初始化包括三个部分：

（1）使能轴组（指令：SMC_GroupPower）；

（2）坐标系转换（指令：MC_SetCoordinateTransform）；

（3）启用轴组（指令：MC_GroupEnable）。

初始化完成后，实现轴组状态机切换到 StandBy

第五步：根据运动轨迹，编写运动学指令

根据设计的运动轨迹调用运动控制指令。首先：快速定位指令，运动到坐标系零点（指令：MC_MoveDirectAbsolute）；其次：调用多条直线插补指令，实现连续插补运动（指令：MC_MoveLinearAbsolute）；再次：调试插补指令的轨迹段过渡参数，得到最好的运动参数。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“SCARA 机械手”。

2. Delta 机器人运动建模及例程

三脚架式的 Delta 机器人的机械结构包括两种类型：

第一种：直线插补运动类型三脚架 Delta 机器人；

第二种：旋转运动类型三脚架 Delta 机器人。

1）直线插补运动类型三脚架 Delta 机器人

（1）机械结构分析

该类机器人的机械结构如图 7.20 所示，三个固定连杆、三组活动连杆及工作台；所有活动连杆的运动均为直线运动。

图 7.20　三脚架 Delta 机器人机械结构示意图

机械结构核心：

Ⅰ：固定连杆导轨：提供活动连杆运动轴的运动导轨，该固定连杆长度相等，而且长度固定。该段为固定段，不可运动。

Ⅱ：活动连杆手臂：每个连接工作台到固定连杆导轨的手臂长度相等，而且长度固定。

Ⅲ：每组活动连杆手臂包含两个连杆，其每组活动连杆手臂的间隔长度相等，其三组连杆分别定义为 A0，A1 及 A2。

（2）建立坐标系

首先建立机械坐标系（MCS）及确定零点。

每个导轨延长点 A 组成的等边三角形的中点为坐标系零点；从零点位置指向 A0 轴与 X-Y 平面的交点 A 的方向为 X 轴正向；Y 方向是基于已定义的 X 及 Z 轴的方向，通过右手定则，确定 Y 的正向；在坐标系零点位置，垂直工作台，背离工作台位置的方向为 Z 的正向，如图 7.21 所示。

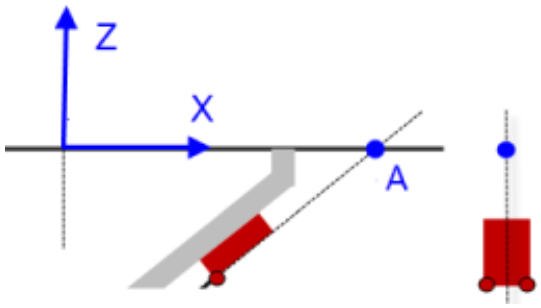


图 7.21　三脚架 Delta 机器人坐标系

然后建立工具坐标系（TCS）相对于机械坐标系的偏移量及绕 Z 轴的旋转角度。

（3）配置机构参数

最大外部半径：由导轨延长线相交形成的三角形的边长，dOuterRadius（单位 unit）；

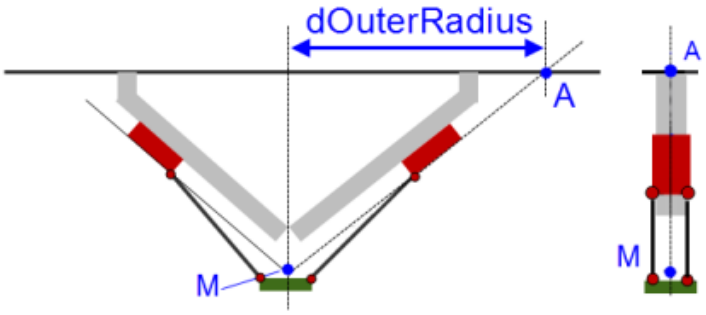


图 7.22　最大外部半径

工作台上连杆节点形成的半径：dInnerRadius（单位 unit），如图 7.23 所示。

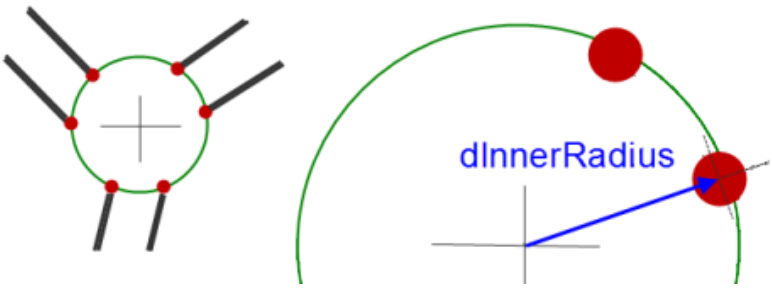


图 7.23　工作台上连杆节点形成的半径

活动连杆长度：dLength（单位 unit）

每组连杆中两个连杆的间隔：dDistance（单位 unit）

导轨与 Z 轴之间的角度：dAxisAngle（单位度）

2）旋转运动类型三脚架 Delta 机器人

（1）机械结构分析

该类机器人的机械结构如图 7.24 所示，有三个固定连杆、三组活动连杆及工作台。所有活动连杆的运动都为旋转运动。

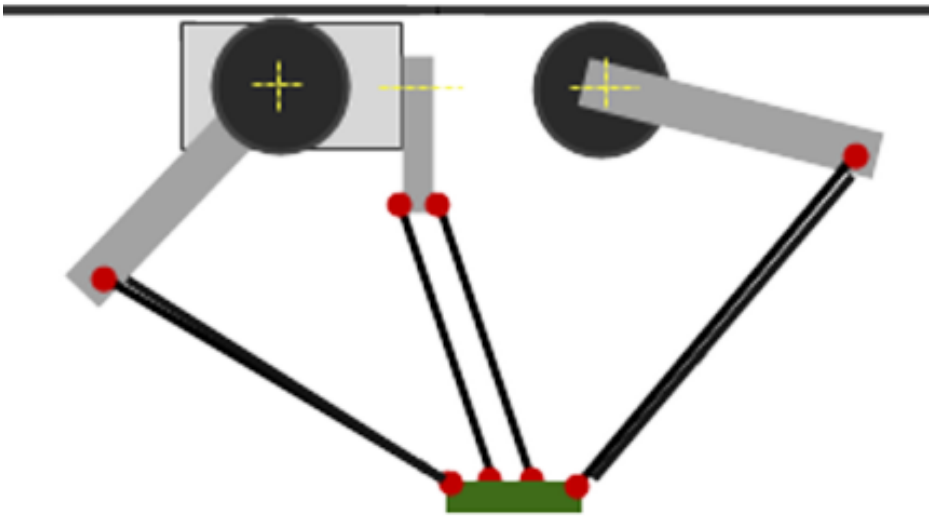


图 7.24　旋转运动类型三脚架 Delta 机器人机械结构示意图

机械结构核心：

Ⅰ：固定连杆：从机座到每个活动连杆的连接处的长度相等，而且长度固定。该段为固定段，不可移动，只能转动；

Ⅱ：活动连杆手臂：每个连接工作台到固定连杆的手臂长度相等，而且长度固定。该手臂拥有三个自由度，即可绕 X、Y、Z 轴做旋转运动；

Ⅲ：每组活动连杆手臂包含两个连杆，其每组活动连杆手臂的间隔长度相等，其三组连杆分别定义为 A0，A1 及 A2。

（2）建立坐标系

首先建立机械坐标系（MCS）及确定零点。

三个活动连杆手臂水平时，其工作台的中心位置为坐标系零点；在坐标系零点位置，A0 手臂背向连杆的方向为 X 轴的正向；Y 方向基于已定义的 X 及 Z 轴的方向，通过右手定则，确定 Y 的正向；在坐标系零点位置，垂直工作台，背离机械机构的方向为 Z 的正向；

然后建立工具坐标系（TCS）相对于机械坐标系的偏移量及绕 Z 轴的旋转角度。

（3）配置机构参数

固定连杆长度：dArmLength1（单位 unit）

活动连杆长度：dArmLength2（单位 unit），如图 7.25 所示。

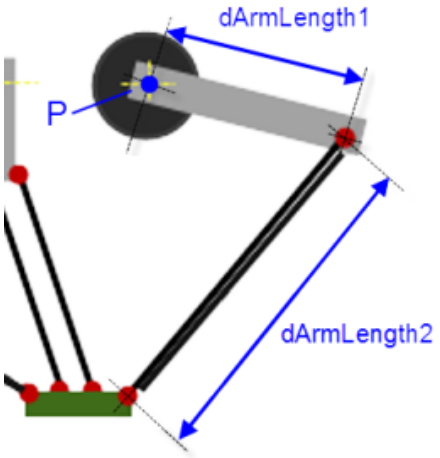


图 7.25　旋转运动类型三脚架 Delta 机器人连杆长度

由三个固定手臂与基座的连接节点组成圆弧半径：dArm1Radius（单位 unit）；
由六个活动手臂与工作台的连接点组成的圆弧半径：dStewartRadius（单位 unit）；
每组活动手臂中的两个连杆的间距：dDistance（单位 unit）；
每个节点的最大旋转角度：dMaxAngleBallJoint（单位度）。
每个节点的最大旋转角度：dMaxAngleBallJoint（单位度）。

3）例程

本例程使用系统仿真实现旋转运动类型三脚架 Delta 机器人系统的运动控制。详细过程描述如下：

第一步：添加控制系统需要的控制轴数

旋转类型的 Delta 系统为三个运动轴，执行末端没有绕 Z 的旋转，因此添加三个虚拟轴，并设置虚拟轴名称为 Axis0、Axis1 及 Axis2，如图 7.26 所示。

第二步：使用编程软件提供的建模工具，建立运动学模型

右键单击工程中的“Application”，通过“添加对象”，选择添加“轴组”，并将轴组命名为 Tripod。通过“运动学模型”界面选择“TRAFO.Kin_Tripod_Rotary”运动学模型，如图 7.27 所示。

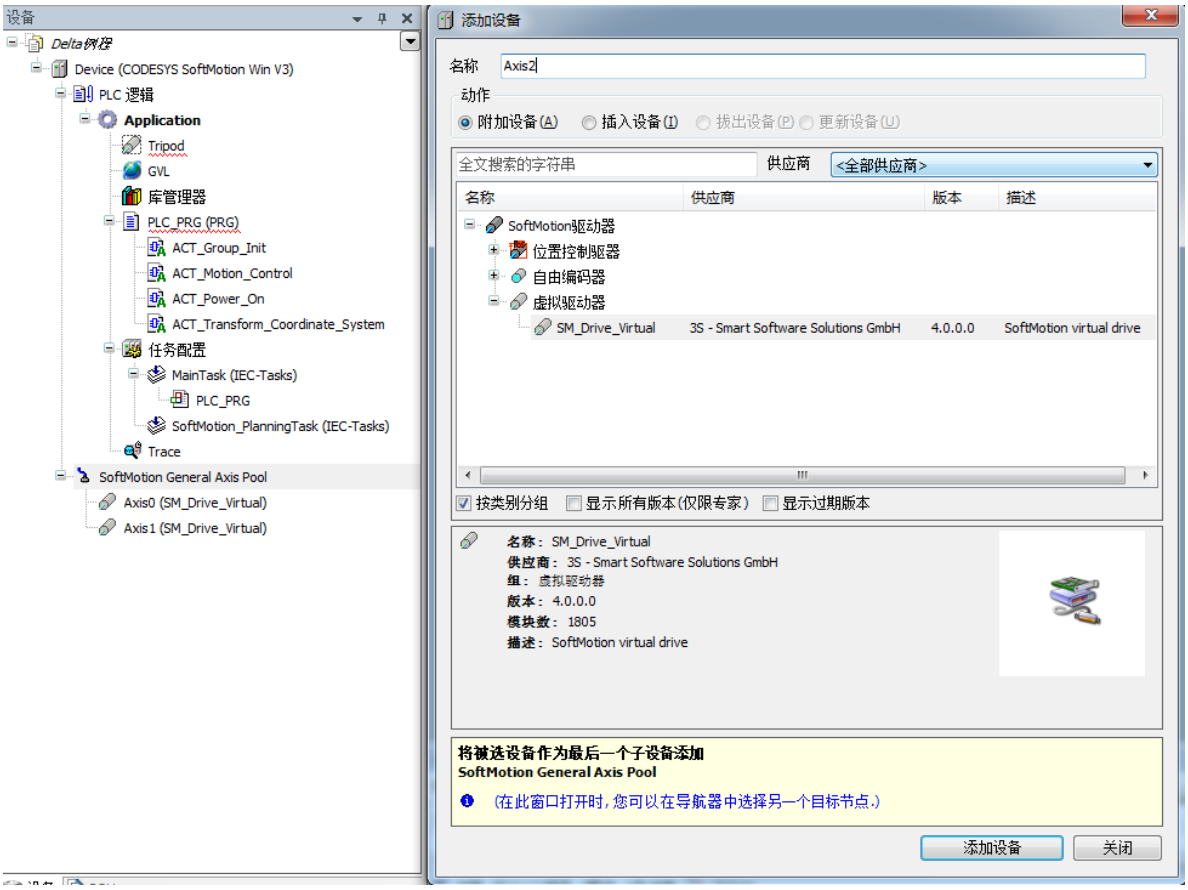


图 7.26　添加虚拟轴

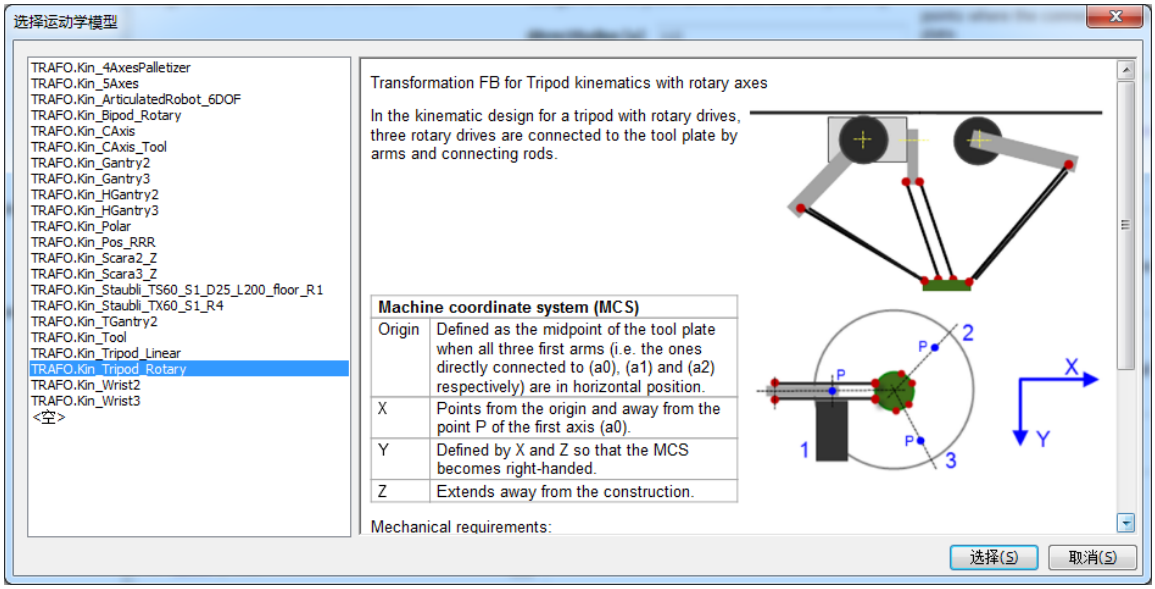


图 7.27　添加运动学模型界面

第三步：配置运动学参数

结构参数配置：

dArmLength1: 200; dArmLenth2:300;dArm1Radius:112;

dStewartRadius:53;dDistance:41;dMaxAangleBallJoint:41;

轴参数配置：在映射轴位置，配置 A1 映射为 Axis0，A2 映射为 Axis1，Z 轴映射为 Axis2；同时根据实际的转动比，配置轴参数。

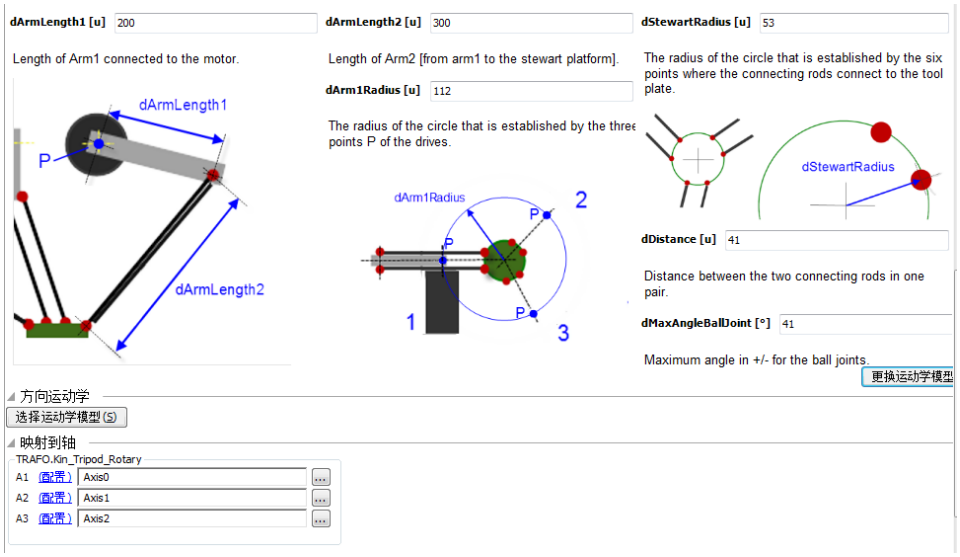


图 7.28　运动学参数配置

第四步：初始化轴组模块

轴组模块的初始化包括三个部分：

（1）使能轴组（指令：SMC_GroupPower）；

（2）坐标系转换（指令：MC_SetCoordinateTransform）；

（3）启用轴组（指令：MC_GroupEnable）。

初始化完成后，实现轴组状态机切换到 StandBy。

第五步：根据运动轨迹，编写运动学指令

根据设计的运动轨迹调用运动控制指令。首先调用快速定位指令，运动到坐标系零点（指令：MC_MoveDirectAbsolute）；其次调用多条直线插补指令，实现连续插补运动（指令：MC_MoveLinearAbsolute）；最后调试插补指令的轨迹段过渡参数，得到最好的运动参数。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“Delta 机械手”。

3. 六自由度（6D）关节机器人运动建模及例程

1）六自由度（6D）关节机器人运动建模

（1）机械结构分析

该类机械结构由 6 个旋转轴组成。机械结构如图 7.29 所示。

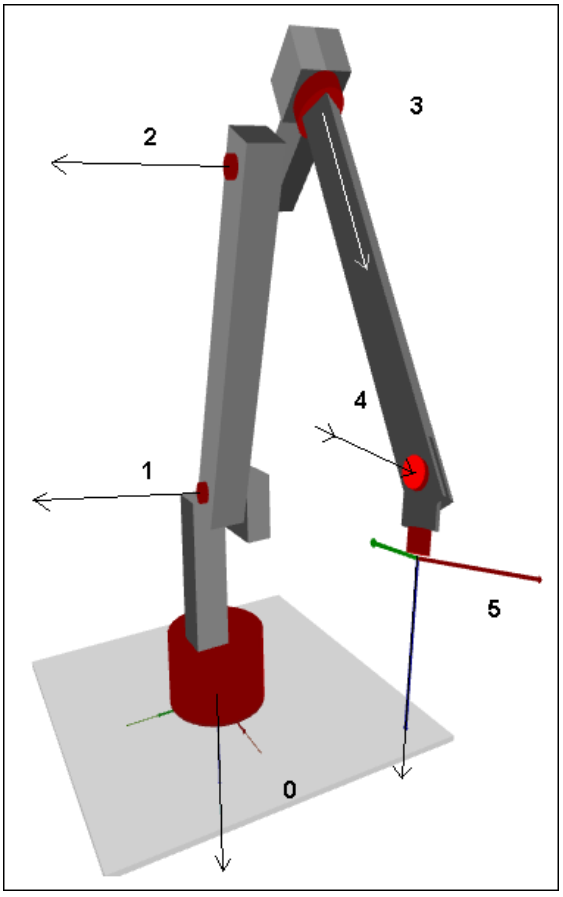


图 7.29　六自由度（6D）关节机器人机械结构示意图

机械结构的核心：

Ⅰ：底座：A0 旋转轴，绕垂直机座方向旋转；

Ⅱ：三个平行于水平面轴：该三个轴在平行于水平面的方向做旋转运动，如图 7.29 中黑色箭头所指轴 A1，A2 及 A4 轴；

Ⅲ：其它两个运动轴：其它两个轴为绕垂直连杆的方向做旋转运动。

（2）建立坐标系

首先建立机械坐标系（MCS）及确定零点。A0 轴与底座相交的中点为机械零点；水平面上，指向机械零点的方向为 X 正向；基于已定义的 X 及 Z 轴的方向，通过右手定则，确定 Y 的正向；垂直水平面，向上的方向为 Z 轴正向。

然后建立工具坐标系（TCS）相对于机械坐标系的偏移量及绕 Z 轴的旋转角度。

（3）配置机构参数

图 7.30 中标记了各个节点间的参数。

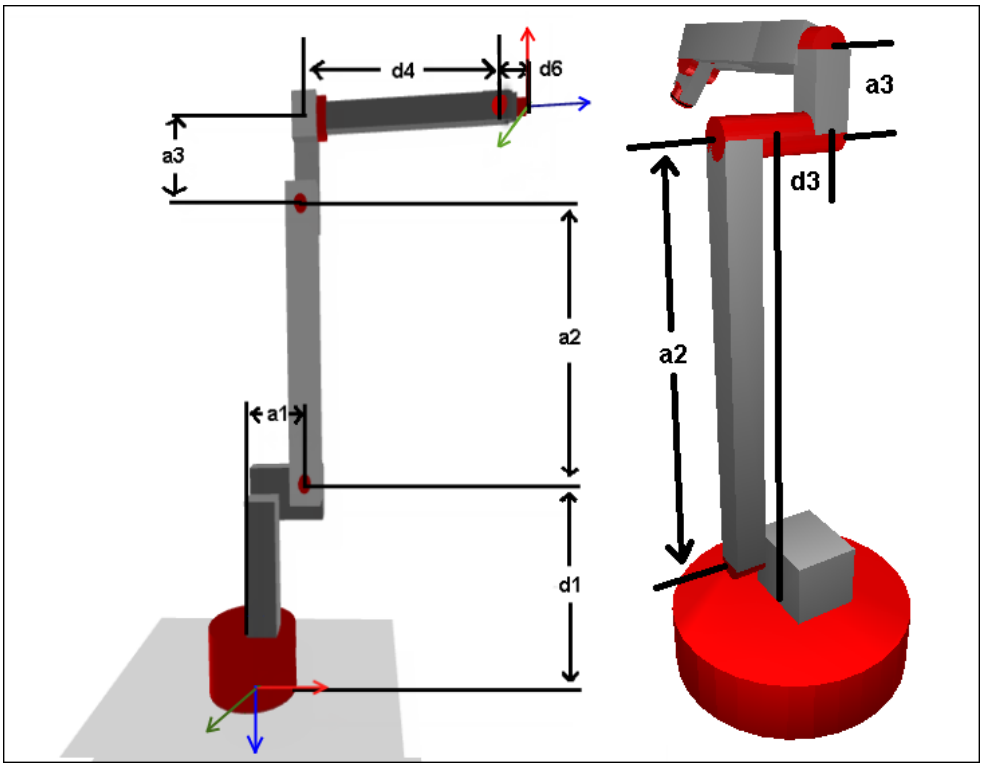


图 7.30　六自由度（6D）关节机器人机构参数示意图

根据 D-H 算法得到对应的数据如表 7.13 所示。

表 7.13　六自由度（6D）关节机器人参数表

Joint number	Joint offset (度)	Link offset d_i(unit)	Link length a_i(unit)	Link twist(度)
0	0	d1	a1	90
1	90	0	a2	0
2	0	d3	a3	90
3	0	d4	0	90
4	0	0	0	-90
5	0	d6	0	0

配置运动参数：

旋转轴 A0 及 A3：　没有运动角度现在，默认为[-180，180]；

旋转轴 A1 及 A4：运动范围[-180，180]；

旋转轴 A2：运动范围[-90，180]；

旋转轴 A5：没有运动限制，默认为[0，360]。

2）例程

本例程介绍使用系统仿真实现六自由度（6D）关节机器人的运动控制。

第一步：添加控制系统需要的控制轴数

六自由度（6D）关节机器人系统共六个旋转轴，执行末端没有其它运动轴，因此添加 6 个虚拟轴，并设置虚拟轴名称分别为 Axis0、Axis1、Axis2、Axis3、Axis4 及 Axis5，如图 7.31 所示。

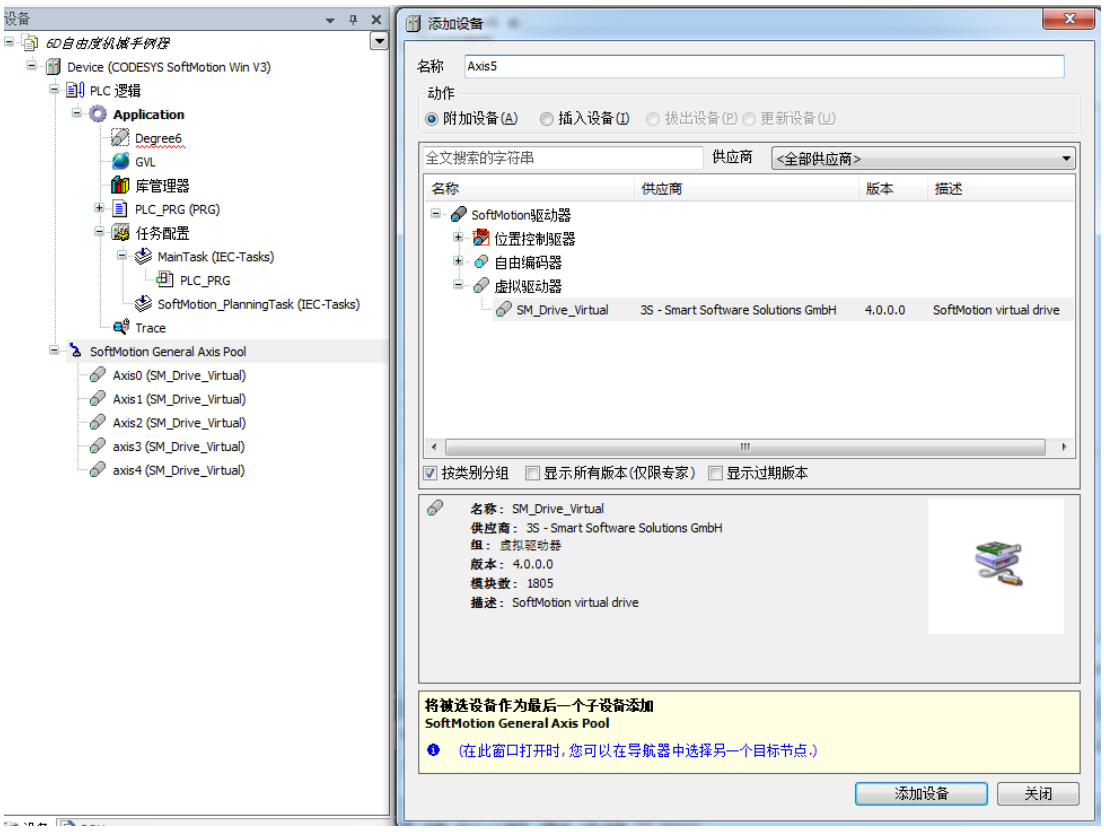


图 7.31　虚拟轴设置界面

第二步：使用编程软件提供的建模工具，建立运动学模型

右键单击工程中的“Application”，通过“添加对象”，选择添加“轴组”，并将轴组命名为 Degree6。通过“运动学模型”界面选择“TRAFO.Kin_RaticulatedRobot_6DOF”运动学模型，如图 7.32 所示。

第三步：配置运动学参数

结构参数配置：

d1：0；a1：0；a2：350；a3：10；d3：20；a4：350；d6：129

轴参数配置：

在映射轴位置，配置 A0 映射为 Axis0，依次对应到 A5 映射为 Axis5；同时根据实际的转动比，配置轴参数如图 7.33 所示。

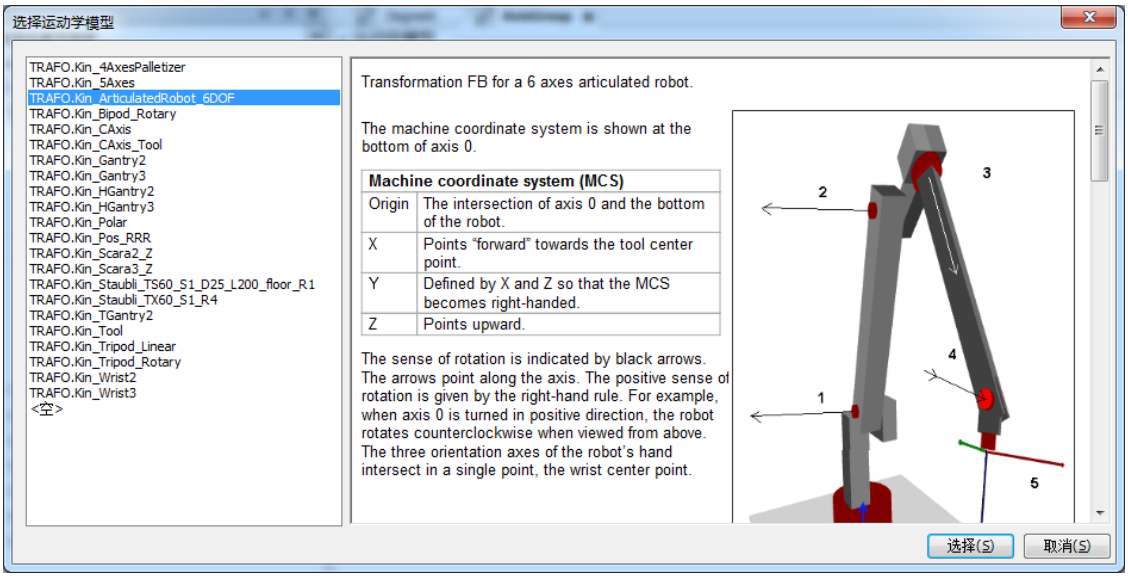


图 7.32　运动学模型界面

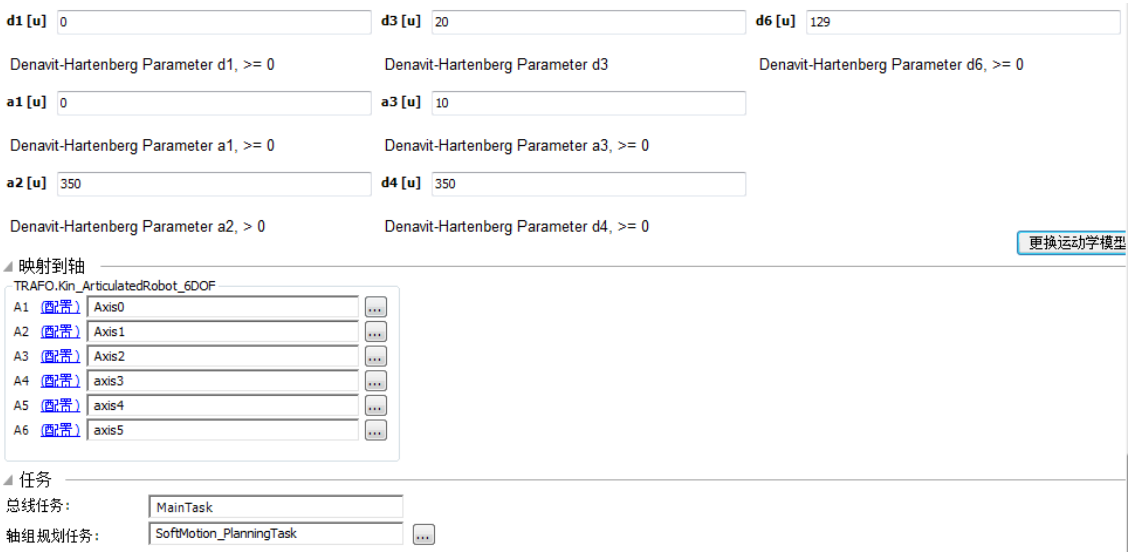


图 7.33　轴参数配置界面

第四步：初始化轴组模块

轴组模块的初始化包括三个部分：

（1）使能轴组（指令：SMC_GroupPower）；

（2）坐标系转换（指令：MC_SetCoordinateTransform）；

（3）启用轴组（指令：MC_GroupEnable）。

初始化完成后，轴组状态机切换到 StandBy。

第五步：根据运动轨迹，编写运动学指令

根据设计的运动轨迹调用运动控制指令。首先调用快速定位指令，运动到坐标系零点（指令：MC_MoveDirectAbsolute）；其次调用多条直线插补指令，实现连续插补运动（指

令：MC_MoveLinearAbsolute）；最后调试插补指令的轨迹段过渡参数，得到最好的运动参数。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“6D 自由度机械手”。

4. 五轴运动建模及例程

（1）机械结构分析

该类机械结构主要由 5 个运动轴组成，其中包括三个直线轴，2 个旋转轴。机械结构图如 7.34 所示。

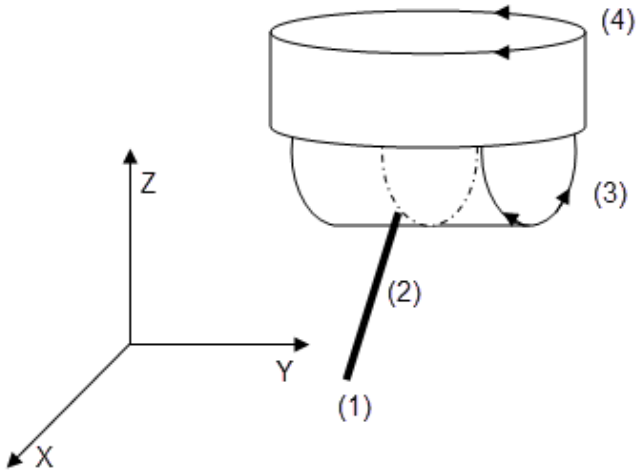


图 7.34　五轴运动控制结构示意图

机械结构核心：

Ⅰ：机械坐标系轴：机械坐标系对应的 X、Y、Z 轴为直线运动轴，映射轴号分别为 A0、A1、A2；

Ⅱ：其它两个运动轴：分别为绕 Y 轴及 Z 轴的旋转运动，映射轴号分别为 A3 及 A4。

（2）建立坐标系

机械坐标系（MCS）与工具坐标系（TCS）相重合；

零点：执行末端（TCP）的前三个轴位置全为零的点，如图 7.35 所示；

X 方向：A0 轴运动的正向即为 X 的正向；

Y 方向：A1 轴运动的正向即为 Y 轴的正向；

Z 方向：A2 轴运动的正向即为 Z 轴的正向；

倾斜角（inclination）：执行末端（TCP）绕 Y 的运行，刀尖指向 X 轴正向的旋转为正向。

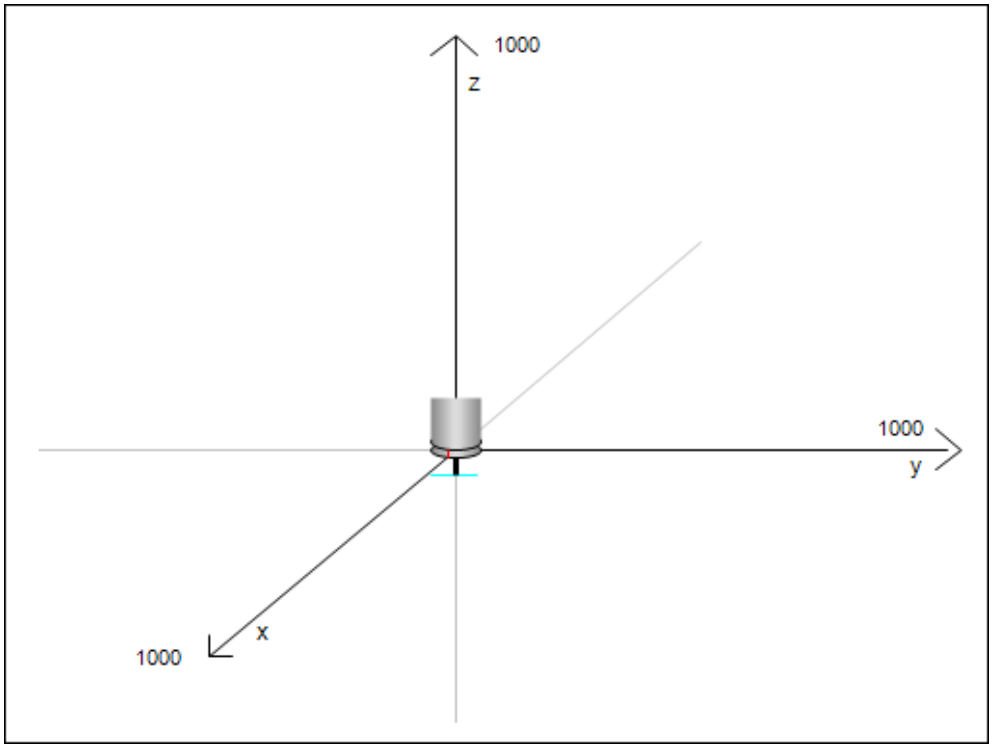


图 7.35　五轴运动控制零点位置示意图

（3）配置机构参数

执行末端（TCP）长度：坐标系零点位置到刀尖的位置长度，dToolLength（单位 unit）。

建立工程、参数配置的方法与上述几个例程类似，可参考 PMC600 软件资料中的“例程”文件夹中的“Kiin_axis5 例程”。

第 8 章　雷赛专用插补指令

PMC600 运动控制器还提供了一套雷赛专用插补指令，包括：直线插补运动、平面和空间圆弧插补运动、两轴椭圆插补、三轴螺旋线插补运动、多轴 G 代码插补指令等。其中通用多轴插补运动指令包含在 PMC_IpoLib 库（需要插补辅助库 PMC_BasicModule）之中，程序若用到插补相关运动功能，须在工程中分别添加 PMC_IpoLib 库和 PMC_BasicModule 库，如图 8.1 所示。

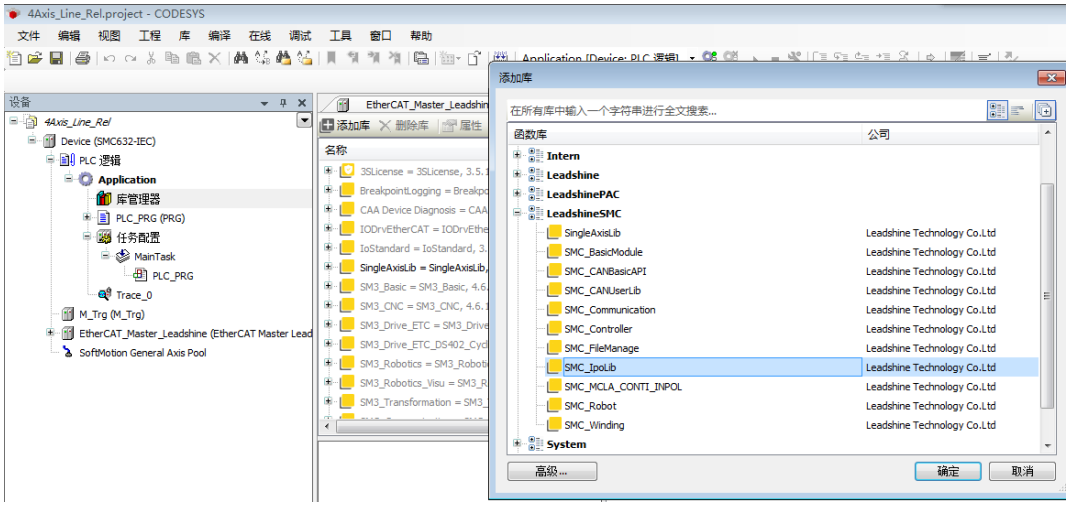


图 8.1　添加 PMC_IpoLib 库

8.1　插补运动指令

8.1.1　直线插补指令

PMC600 支持 2~6 轴直线插补运动，包括相对坐标模式和绝对坐标模式；其中 4~6 轴直线插补运动中前三轴做空间直线插补运动，剩余对应的轴做跟随运动，和参与插补运动的轴同时启停，相关指令如表 8.1 所示。

直线插补指令在程序中可直接调用执行，但其中的参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。

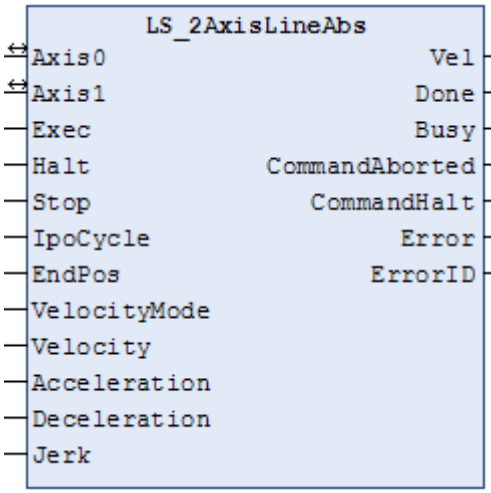
表 8.1　雷赛直线插补指令

指令名	功能说明
LS_2AxisLineAbs	绝对坐标两轴直线插补运动
LS_2AxisLineRel	相对坐标两轴直线插补运动
LS_3AxisLineAbs	绝对坐标三轴直线插补运动
LS_3AxisLineRel	相对坐标三轴直线插补运动
LS_4AxisLineAbs	绝对坐标三轴直线插补、一轴跟随运动
LS_4AxisLineRel	相对坐标三轴直线插补、一轴跟随运动
LS_5AxisLineAbs	绝对坐标三轴直线插补、两轴跟随运动
LS_5AxisLineRel	相对坐标三轴直线插补、两轴跟随运动
LS_6AxisLineAbs	绝对坐标三轴直线插补、三轴跟随运动
LS_6AxisLineRel	相对坐标三轴直线插补、三轴跟随运动

两轴绝对直线插补 LS_2AxisLineAbs

两轴直线插补指令，采用绝对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_2AxisLineAbs	FB		LS_2AxisLineAbs(Axis0: = （参数）， Axis1: = （参数）， Exec: = （参数）， Halt: = （参数）， Stop: = （参数）， IpoCycle: = （参数）， EndPos: = （参数）， VelocityMode: = （参数）， Velocity: = （参数）， Acceleration: = （参数）， Deceleration: = （参数）， Jerk: = （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，	False	直线插补模块启动命令，上升沿

			FALSE		有效
Halt	暂停	BOOL	TRUE，FALSE	False	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us，必须和所在任务周期一致
Endpos	目标位置	ARRAY [0..1] OF REAL	负数，0，正数	0	轴的目标位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VEL MODE	0-3	SIGMOID	速度模式：梯形-0（TRAPEZOID）；S 形-1（SIGMOID）；四次方-3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度，Pulse/s^2
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度，Pulse/s^2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度，Pulse/s^2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时候需要设置此参数，该参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	True 表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	True 表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	False	错误状态
ErrorID	错误码	DINT	0，正数	0	错误码

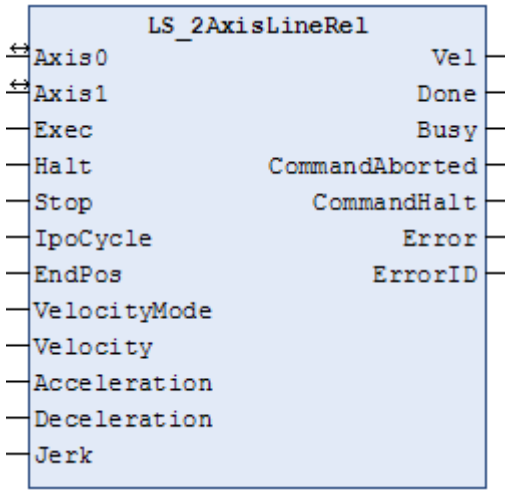
功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- 指令指定的两个轴做直线插补运动，位置坐标为绝对坐标。
- 注意：IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现 Error 报错。
- 参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。

两轴相对直线插补 LS_2AxisLineRel

两轴直线插补指令，采用相对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_2AxisLineRel	FB		LS_2AxisLineRel(Axis0：= （参数）， Axis1：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， EndPos：= （参数）， VelocityMode：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	直线插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	暂停命令，当该输入量为 True 时，当前插补运动暂停，当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
Endpos	目标位置	ARRAY [0..1] OF REAL	负数，0，正数	0	轴需要移动的距离，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度
Jerk	加加	LREAL	负数，0	90000000	加加速度，速度模式为 3 时候

	速度		正数		需要设置此参数，此参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
Busy	运动中	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补过程中出现错误。
Error	报错	BOOL	TRUE，FALSE	False	错误状态
ErrorID	错误码	DINT	0，正数	0	错误码

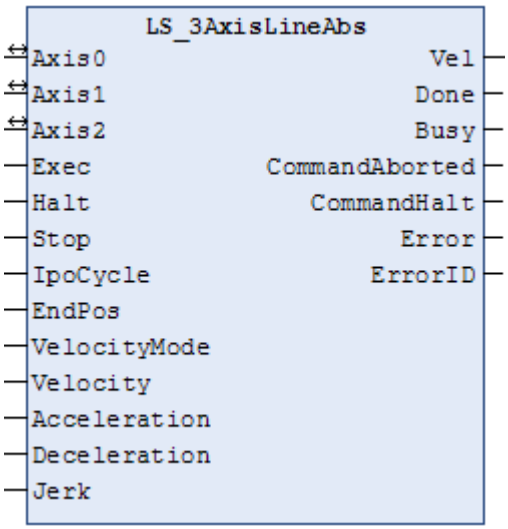
功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- 指令指定的两个轴做直线插补运动，位置坐标为相对坐标。
- 注意，IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现 Error 报错。
- 参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。

三轴绝对直线插补 LS_3AxisLineAbs

三轴直线插补指令，采用绝对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_3AxisLineAbs	FB		LS_3AxisLineAbs(Axis0：= （参数）， Axis1：= （参数）， Axis2：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， EndPos：= （参数）， VelocityMode：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
Axis2	轴 2	AXIS_REF_VIRTUAL_SM3	-	-	插补 2 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	直线插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
Endpos	目标位置	ARRAY [0..2] OF REAL	负数，0，正数	0	轴的目标位置，单位：pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度

Acceleration	加速度	LREAL	0，正数	0	插补合成加速度
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时候需要设置此参数，此参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	False	错误状态
ErrorID	错误码	DINT	0，正数	0	错误码

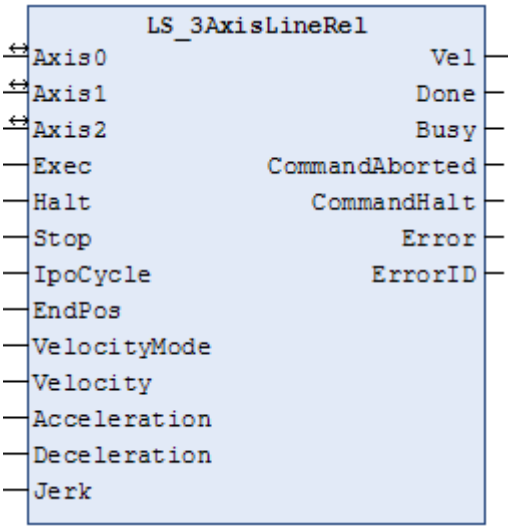
功能说明：

与两轴直线插补指令说明相同。

三轴相对直线插补 LS_3AxisLineRel

三轴直线插补指令，采用相对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_3AxisLineRel	FB		LS_3AxisLineRel(Axis0: = （参数）， Axis1: = （参数）， Axis2: = （参数）， Exec: = （参数）， Halt: = （参数）， Stop: = （参数）， IpoCycle: = （参数）， EndPos: = （参数）， VelocityMode: = （参数）， Velocity: = （参数）， Acceleration: = （参数）， Deceleration: = （参数）， Jerk: = （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
Axis2	轴 2	AXIS_REF_VIRTUAL_SM3	-	-	插补 2 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	直线插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	暂停命令，当该输入量为 True 时，当前插补运动暂停，当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
Endpos	目标位置	ARRAY [0..2] OF REAL	负数，0，正数	0	轴需要移动的距离，单位：pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时候需要设置此参数，此参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	False	错误状态
ErrorID	错误码	DINT	0，正数	0	错误码

功能说明： 与两轴直线插补指令说明相同。

四轴绝对直线插补 LS_4AxisLineAbs

请参考 LS_3AxisLineAbs 指令。

四轴相对直线插补 LS_4AxisLineRel

请参考 LS_3AxisLineRel 指令。

五轴绝对直线插补 LS_5AxisLineAbs

请参考 LS_3AxisLineAbs 指令。

五轴相对直线插补 LS_5AxisLineRel

请参考 LS_3AxisLineRel 指令。

六轴绝对直线插补 LS_6AxisLineAbs

请参考 LS_3AxisLineAbs 指令。

六轴相对直线插补 LS_6AxisLineRel

请参考 LS_3AxisLineRel 指令。

8.1.2 直线插补（可调速）指令

PMC600 支持 2 轴直线插补运动在运动过程中改变插补速度，相关指令如表 8.2 所示。

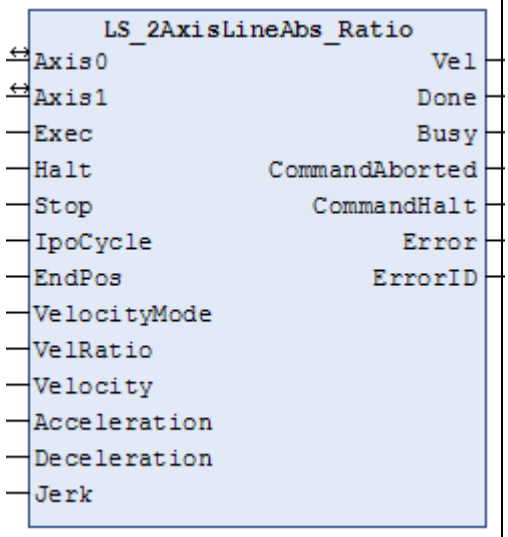
表 8.2 直线插补可调速指令表

指令名	功能说明
LS_2AxisLineAbs_Ratio	绝对坐标两轴可调速直线插补
LS_2AxisLineRel_Ratio	相对坐标两轴可调速直线插补

两轴绝对直线插补可调速 LS_2AxisLineAbs_Ratio

两轴直线插补可调速指令，采用绝对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_2AxisLineAbs_Ratio	FB		LS_2AxisLineAbs_Ratio(Axis0：= （参数）， Axis1：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， EndPos：= （参数）， VelocityMode：= （参数）， VelRatio：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数） ）；

变量：

VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	启动，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	暂停，上升沿有效
Stop	停止	BOOL	TRUE，FALSE	False	停止，上升沿有效
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
EndPos	目标位置	ARRAY [0..1] OF REAL	负数，0，正数	0	轴的目标位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值 0.01，最大值 2

Velocity	速度	LREAL	0，正数	0	插补速度 Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补加速度 Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补减速度 Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度 Pulse/s3，速度模式设置为 3 时需要设置该参数，该参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合速度
Done	完成	BOOL	TRUE，FALSE	FALSE	插补结束状态
Busy	运动中	BOOL	TRUE，FALSE	FALSE	是否插补进行中
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	插补终止状态
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	插补暂停状态
Error	报错	BOOL	TRUE，FALSE	False	错误状态
ErrorID	错误码	DINT	0，正数	0	错误码

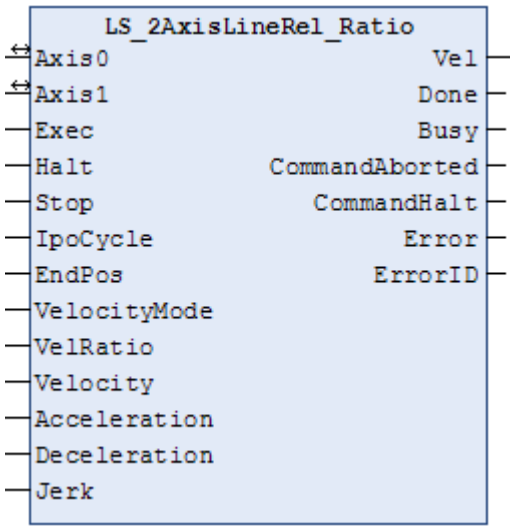
功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- 与 LS_2AxisLineAbs 相比，增加了可调速功能，即在插补过程中可通过改变 VelRatio 的值来在线改变插补速度。
- 注意，IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现 Error 报错。
- 参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。

两轴相对直线插补可调速 LS_2AxisLineRel_Ratio

两轴直线插补可调速指令，采用相对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_2AxisLineRel_Ratio	FB		LS_2AxisLineRel_Ratio (Axis0:= （参数）， Axis1:= （参数）， Exec:= （参数）， Halt:= （参数）， Stop:= （参数）， IpoCycle:= （参数）， EndPos:= （参数）， VelocityMode:= （参数）， VelRatio:= （参数）， Velocity:= （参数）， Acceleration:= （参数）， Deceleration:= （参数）， Jerk:= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	启动，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	暂停，上升沿有效
Stop	停止	BOOL	TRUE，FALSE	False	停止，上升沿有效
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
EndPos	目标位置	ARRAY [0..1] OF REAL	负数，0，正数	0	轴的目标位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值 0.01，最大值 2
Velocity	速度	LREAL	0，正数	0	插补速度 Pulse/s

Acceleration	加速度	LREAL	0，正数	0	插补加速度 Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补减速度 Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，Pulse/s3 速度模式设置为 3 时需要设置该参数，该参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合速度
Done	完成	BOOL	TRUE，FALSE	FALSE	插补结束状态
Busy	运动中	BOOL	TRUE，FALSE	FALSE	是否插补进行中
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	插补终止状态
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	插补暂停状态
Error	报错	BOOL	TRUE，FALSE	False	错误状态
ErrorID	错误码	DINT	0，正数	0	错误码

功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- 与 LS_2AxisLineRel 相比，增加了可调速功能，即在插补过程中可通过改变 VelRatio 的值来在线改变插补速度。
- 注意，IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现 Error 报错。
- 参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。

8.1.3 圆弧插补运动指令

PMC600 运动控制器的雷赛插补指令支持平面圆弧插补（两轴圆弧插补）和空间圆弧插补（三轴圆弧插补）如表 8.3 所示。

平面圆弧插补有 4 种模式：

模式 0：三点圆弧模式，辅助点输入圆弧经过的点坐标；

模式 1：圆心与终点模式，辅助点输入圆心的坐标；

模式 2：终点半径模式，需要输入圆弧的半径；

模式 3：目标角度模式，停止位置为相对于起点的位置，需要输入圆弧的方向和圆心坐标。

空间圆弧插补只支持模式 0-三点圆弧模式，具体参照运动指令说明。

圆弧插补指令可在程序中直接调用执行，其中参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0；各个模式中没有用到的参数就不要填。

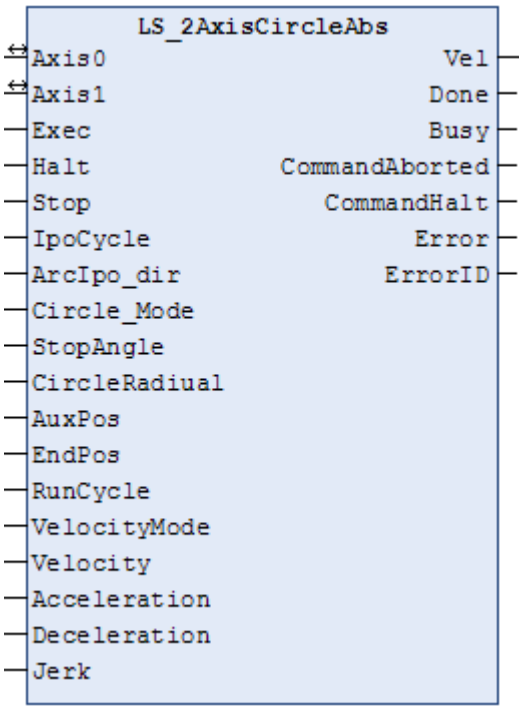
表 8.3 圆弧插补运动指令

指令名	功能说明
LS_2AxisCircleAbs	绝对坐标两轴圆弧插补运动
LS_2AxisCircleRel	相对坐标两轴圆弧插补运动
LS_3AxisCircleAbs	绝对坐标三轴圆弧插补运动
LS_3AxisCircleRel	相对坐标三轴圆弧插补运动

两轴绝对圆弧插补 LS_2AxisCircleAbs

两轴圆弧插补指令，采用绝对坐标

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_2AxisCircleAbs	FB		LS_2AxisCircleAbs(Axis0：= （参数）， Axis1：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， ArcIpo_dir：= （参数）， Circle_Mode：= （参数）， StopAngle：= （参数）， CircleRadiual：= （参数）， AuxPos：= （参数）， EndPos：= （参数）， RunCycle：= （参数）， VelocityMode：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数） ）；

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	圆弧插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	圆弧插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
ArcIpo_dir	插补方向	INT	0，1	0	圆弧插补的方向，0 表示逆时针方向，1 表示顺时针方向。圆弧模式 1、2、3 需要设置该值
Circle_Mode	插补模式	INT	0-3	1	圆弧模式：0 表示三点圆弧模式；1 表示圆心位置与终点确定圆弧；2 表示终点半径模式；3 表示目标角度模式
StopAngle	停止角度	REAL	0-360	0	停止的目标角度，相对于起点的角度，单位为度，取值范围为[0，360.0]；圆弧模式 3 需要设置该值
CircleRadiual	圆弧半径	REAL	正数	0	圆弧半径，圆弧模式 2 需要设置该值。该值为负表示圆弧角小于 180 度，半径为正表示圆弧角大于 180 度
AuxPos	辅助位置	ARRAY [0..1] OF REAL	负数，0 正数	0	辅助点，单位：Pulse
Endpos	停止位置	ARRAY [0..1] OF REAL	负数，0 正数	0	终点位置，单位：Pulse
RunCycle	循环次数	UINT	0，正数	0	圆弧运动的循环次数（即圆弧旋转的圈数）
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时候需要设置此参数，该参数值不能为 0
VAR_OUTPUT					

Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	插补结束状态，为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	插补进行中状态，为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	插补终止状态，当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	False	错误状态
ErrorID	错误码	DINT	0，正数	0	错误码

功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- “Circle_Mode”参数用来设置圆弧插补的模式，分别如下：

1）Circle_Mode=0，表示三点圆弧模式，如图 8.2 所示。

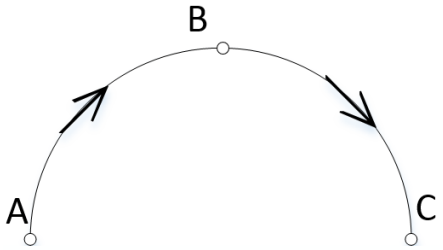


图 8.2　三点圆弧插补

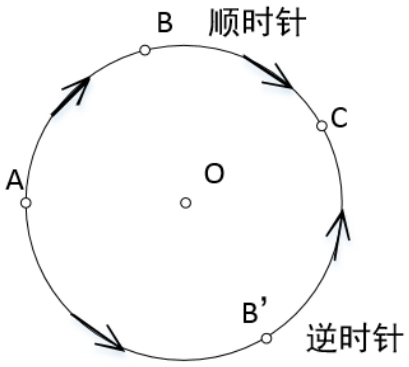


图 8.3　圆心终点圆弧插补模式

在使用三点圆弧模式时，需要在圆弧轨迹上确定 3 个不重合的点，如图 8.2 中的 A、B、C 三点。在指令中，A 代表圆弧插补轴 0 和 1 的当前位置，不需要设置；B 代表圆弧经过的点，需要设置为参数 AuxPos；C 代表圆弧的终点位置，需要设置为参数 Endpos。

2）Circle_Mode=1，表示圆心与终点模式的圆弧，如图 8.3 所示。

在使用圆心终点圆弧模式时，需要设置圆心 O、终点 C 和运动方向等参数。在指令中 O 代表圆心坐标，需要设置为参数 AuxPos；C 代表圆弧的终点坐标，需要设置为参数 Endpos；运动方向需要设置参数 ArcIpo_dir。

3）Circle_Mode=2，表示终点半径模式，如图 8.4 所示。

在使用终点半径模式时，需要设置终点 C，半径 R 和运动方向等参数。在指令中 C 代表圆弧的终点位置，需要设置为参数 Endpos；半径 R 需要设置为参数 CircleRadiual；运动方向需要设置为参数 ArcIpo_dir。

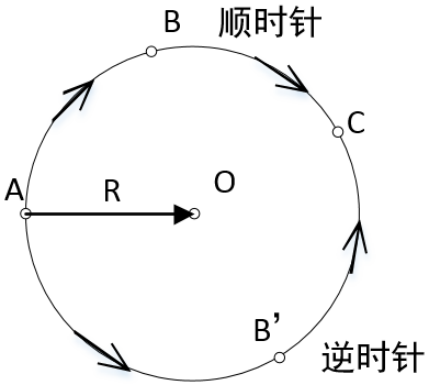


图 8.4　终点半径圆弧插补模式

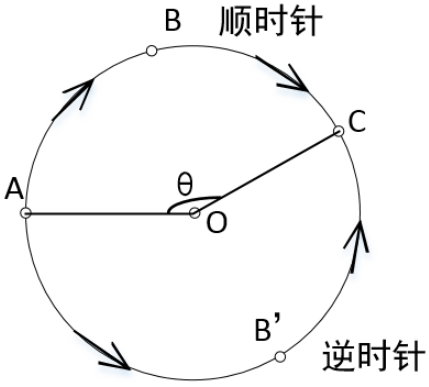


图 8.5　目标角度圆弧插补模式

4）Circle_Mode=3，表示目标角度模式，如图 8.5 所示。

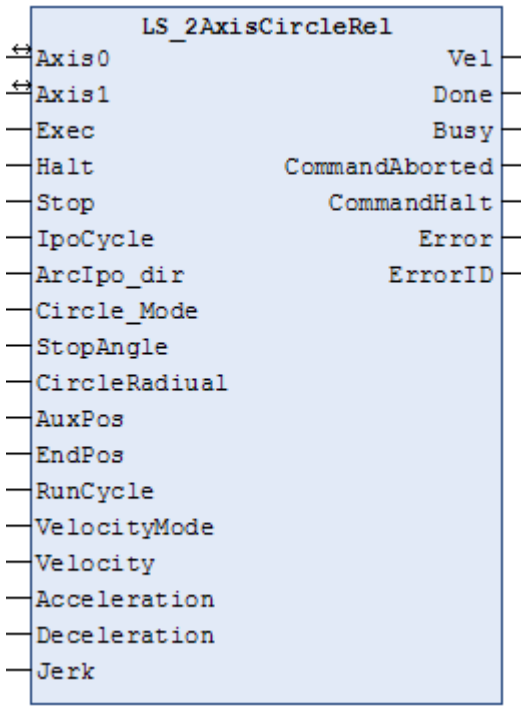
在使用目标角度模式时，需要设置圆心 O，角度 θ 和运动方向等参数。在指令中 θ 代表圆心位置，需要设置为参数 Auxpos；目标角度需要设置为参数 StopAngle；运动方向需要设置为参数 ArcIpo_dir。

- 注意：IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现 Error 报错。
- 参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。
- 在指令运行期间，运动轴都不能再被其它运动指令所调用。

两轴相对圆弧插补 LS_2AxisCircleRel

两轴圆弧插补指令，采用相对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_2AxisCircleRel	FB		LS_2AxisCircleRel(Axis0：= （参数）， Axis1：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， ArcIpo_dir：= （参数）， Circle_Mode：= （参数）， StopAngle：= （参数）， CircleRadiual：= （参数）， AuxPos：= （参数）， EndPos：= （参数）， RunCycle：= （参数）， VelocityMode：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	圆弧插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	圆弧插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
ArcIpo_dir	插补方向	INT	0，1	0	圆弧插补的方向，0 表示逆时针方向，1 表示顺时针方向。圆弧模式 1、2、3 需要设置该值
Circle_Mode	插补模式	INT	0-3	1	圆弧模式：0 表示三点圆弧模式；1 表示圆心位置与终

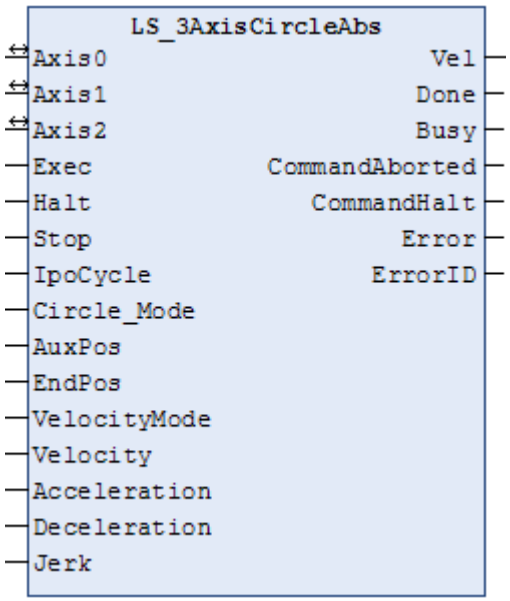
					点确定圆弧；2 表示终点半径模式；3 表示目标角度模式
StopAngle	停止角度	REAL	0-360	0	停止的目标角度，相对于起点的角度，单位为度，取值范围为[0，360.0]。圆弧模式 3 需要设置该值
CircleRadiual	圆弧半径	REAL	正数	0	圆弧半径，圆弧模式 2 需要设置该值。该值半径为负表示大圆弧，半径为正表示小圆弧
AuxPos	辅助位置	ARRAY [0..1] OF REAL	负数，0 正数	0	辅助点，增量坐标，单位：Pulse
Endpos	停止位置	ARRAY [0..1] OF REAL	负数，0 正数	0	终点位置，增量坐标，单位：Pulse
RunCycle	循环次数	UINT	0，正数	0	圆弧运动的循环次数（即圆弧旋转的圈数）
VelocityMode	速度模式	SMC_INT_VEL MODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度 Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度 Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度 Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时候需要设置此参数，此参数值不能位 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	插补结束状态，为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	插补进行中状态，为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	插补终止状态，当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	False	错误状态
ErrorID	错误码	DINT	0，正数	0	错误码

功能说明：除相对坐标外，其他参数设置与绝对坐标两轴圆弧插补指令相同。

三轴绝对圆弧插补 LS_3AxisCircleAbs

空间圆弧插补运动，采用绝对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_3AxisCircleAbs	FB		LS_3AxisCircleAbs(Axis0：= （参数）， Axis1：= （参数）， Axis2：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， Circle_Mode：= （参数）， AuxPos：= （参数）， EndPos：= （参数）， VelocityMode：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
Axis2	轴 2	AXIS_REF_VIRTUAL_SM3	-	-	插补 2 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	启动，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	暂停，上升沿有效
Stop	停止	BOOL	TRUE，FALSE	False	停止，上升沿有效
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
Circle_Mode	插补模	INT	0-3	1	圆弧模式:0 表示三点圆弧模

	式				式;1 表示圆心位置与终点确定圆弧；2 表示终点半径模式；3 表示目标角度模式
AuxPos	辅助位置	ARRAY [0..1] OF REAL	负数，0 正数	0	辅助点，单位：Pulse
EndPos	停止位置	ARRAY [0..1] OF REAL	负数，0 正数	0	终点位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式 梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度 Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度 Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度 Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度 Pulse/s3，速度模式设置为 3 时需要设置该参数，该参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合速度
Done	完成	BOOL	TRUE，FALSE	FALSE	插补结束状态
Busy	运动中	BOOL	TRUE，FALSE	FALSE	插补进行中状态
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	插补终止状态
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	插补暂停状态
Error	报错	BOOL	TRUE，FALSE	False	插补错误状态
ErrorID	错误码	DINT	0，正数	0	错误码

功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- “Circle_Mode”参数用来设置三轴圆弧的圆弧模式，默认模式为三点圆弧，对应参数值为 0，且仅支持这种圆弧模式，圆弧轨迹如图 8.6 所示，A 点为轴 0、1、2 的当前位置；B 点位圆弧经过的点，C 点为圆弧的终点。

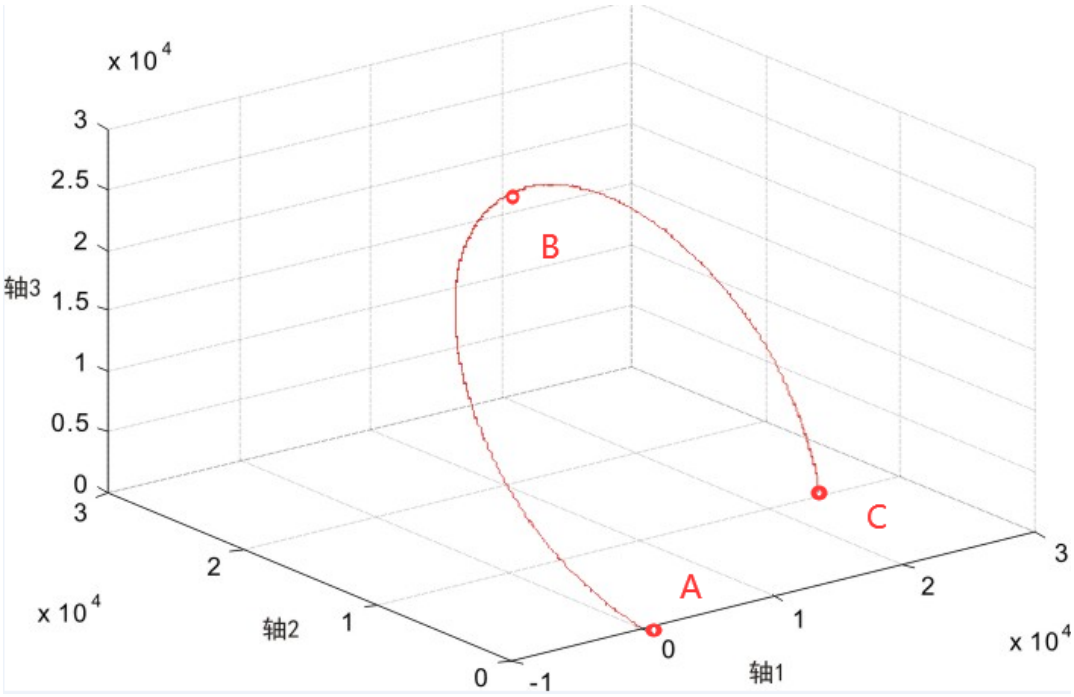


图 8.6　绝对位置模式空间三轴圆弧插补轨迹

三轴相对圆弧插补 LS_3AxisCircleRel

空间圆弧插补运动，采用相对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_3AxisCircleRel	FB	LS_3AxisCircleRel Axis0 → Vel Axis1 → Done Axis2 → Busy Exec → CommandAborted Halt → CommandHalt Stop → Error IpoCycle → ErrorID Circle_Mode AuxPos EndPos VelocityMode Velocity Acceleration Deceleration Jerk	LS_3AxisCircleRel(Axis0：= （参数）， Axis1：= （参数）， Axis2：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， Circle_Mode：= （参数）， AuxPos：= （参数）， EndPos：= （参数）， VelocityMode：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始化	注释
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
Axis2	轴 2	AXIS_REF_VIRTUAL_SM3	-	-	插补 2 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	启动，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	暂停，上升沿有效
Stop	停止	BOOL	TRUE，FALSE	False	停止，上升沿有效
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
Circle_Mode	插补模式	INT	0-3	1	圆弧模式：0 表示三点圆弧模式；1 表示圆心位置与终点确定圆弧；2 表示终点半径模式；3 表示目标角度模式
AuxPos	辅助位置	ARRAY [0..1] OF REAL	负数，0 正数	0	辅助点，单位：Pulse
EndPos	停止位置	ARRAY [0..1] OF REAL	负数，0 正数	0	终点位置，单位：Pulse
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式， 梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度 Pulse/s
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度 Pulse/s2
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度 Pulse/s2
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度 Pulse/s3 速度模式设置为 3 时需要设置该参数，该参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合速度
Done	完成	BOOL	TRUE，FALSE	FALSE	插补结束状态
Busy	运动中	BOOL	TRUE，FALSE	FALSE	插补进行中状态
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	插补终止状态
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	插补暂停状态
Error	报错	BOOL	TRUE，FALSE	False	插补错误状态
ErrorID	错误码	DINT	0，正数	0	错误码

功能说明：除相对坐标外，其他与 LS_3AxisCircleAbs 指令相同。

8.1.4 椭圆插补运动指令

PMC600 运动控制器目前仅支持两轴椭圆插补（即平面椭圆插补），相关指令如表 8.4 所示。两轴椭圆插补指令可在程序中直接调用执行，其中参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。

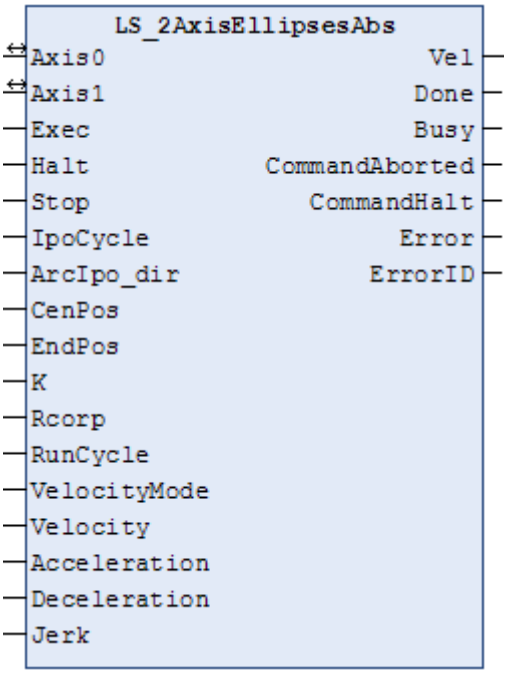
表 8.4 椭圆插补运动指令

指令名	功能说明
LS_2AxisEllipsesAbs	绝对坐标两轴椭圆插补
LS_2AxisEllipsesRel	相对坐标两轴椭圆插补

两轴绝对椭圆插补 LS_2AxisEllipsesAbs

两轴椭圆插补，采用绝对坐标。

指令外观 ：

指令	FB/FUN	图形模块	结构文本
LS_2AxisEllipsesAbs	FB		LS_2AxisEllipsesAbs(Axis0: = （参数）， Axis1: = （参数）， Exec: = （参数）， Halt: = （参数）， Stop: = （参数）， IpoCycle: = （参数）， ArcIpo_dir: = （参数）， CenPos: = （参数）， EndPos: = （参数）， K: = （参数）， Rcorp: = （参数）， RunCycle: = （参数）， VelocityMode: = （参数）， Velocity: = （参数）， Acceleration: = （参数）， Deceleration: = （参数）， Jerk: = （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	椭圆插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	椭圆插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
ArcIpo_dir	插补方向	INT	0，1	0	椭圆插补的方向：0 表示逆时针方向；1 表示顺时针方向
CenPos	中心位置	ARRAY [0..1] OF REAL	负数，0 正数	0	椭圆的中心位置，绝对坐标，单位：Pulse
Endpos	终点位置	ARRAY [0..1] OF REAL	负数，0 正数	0	终点位置，绝对坐标，单位：Pulse
K	主轴角度	REAl	0-360	0	主轴的角度，主轴相对于 X 轴的角度；长轴与 X 轴的夹角，用来确定主轴的方向。
Rcorp	短轴长轴比	REAL	0，1	0	短轴与长轴的比，取值范围为[0，1]
RunCycle	循环次数	UINT	0，正数	0	椭圆运动的循环次数（即椭圆的圈数）
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时候需要设置此参数，此参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。

CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	False	为 True 时，表示插补过程中出现错误。
ErrorID	错误码	DINT	0，正数	0	当 Error 为 True 时，该变量输出错误码

功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- 指令指定的两个轴做椭圆插补运动，位置坐标为绝对坐标。
- 椭圆插补的轨迹，如图 8.7 所示。

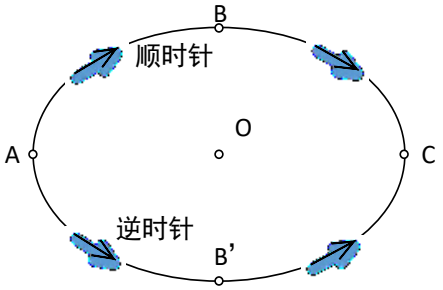


图 8.7　椭圆插补

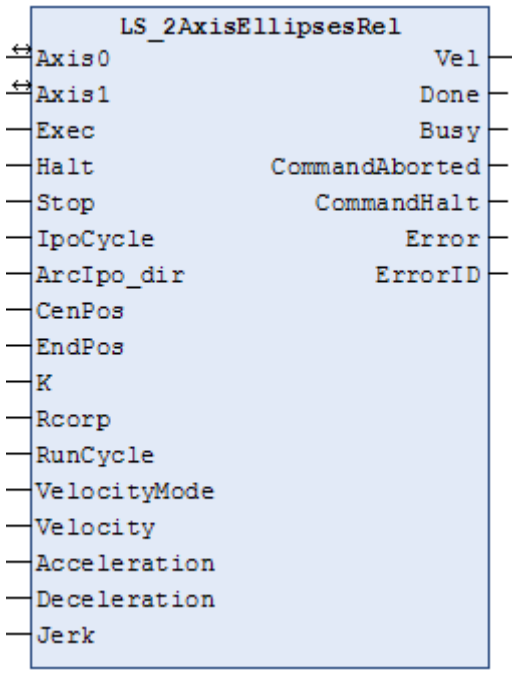
在使用椭圆插补运动时，需要设置圆心 O、终点 C 和运动方向等参数。在指令中，A 代表圆弧插补轴 0 和 1 的当前位置，不需要设置；O 代表圆心位置，需要设置为参数 CenPos；C 代表圆弧的终点位置，需要设置为参数 Endpos；运动方向需要设置为参数 ArcIpo_dir；椭圆轨迹执行圈数，则需要设置为参数 RunCycle。

- 注意，IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现 Error 报错。
- 参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。
- 在指令运行期间，运动轴都不能再被其它运动指令所调用。

两轴相对椭圆插补 LS_2AxisEllipsesRel

两轴椭圆插补，采用相对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_2AxisEllipsesRel	FB		LS_2AxisEllipsesRel(Axis0：= （参数）， Axis1：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， ArcIpo_dir：= （参数）， CenPos：= （参数）， EndPos：= （参数）， K：= （参数）， Rcorp：= （参数）， RunCycle：= （参数）， VelocityMode：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数） ）；

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	椭圆插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	椭圆插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
ArcIpo_dir	插补方向	INT	0，1	0	椭圆插补的方向：0 表示逆时针方向；1 表示顺时针方向
CenPos	中心位置	ARRAY [0..1] OF REAL	负数，0 正数	0	椭圆的中心位置，相对坐标，单位：Pulse
Endpos	终点位置	ARRAY [0..1] OF REAL	负数，0 正数	0	终点位置，相对坐标，单位：Puls
K	主轴角度	REAl	0-360	0	主轴的角度，主轴相对于

					X 轴的角度；长轴与 X 轴的夹角，用来确定主轴的方向。
Rcorp	短轴长轴比	REAL	0，1	0	短轴与长轴的比，取值范围为[0，1]
RunCycle	循环次数	UINT	0，正数	0	椭圆运动的循环次数（即椭圆的圈数）
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时候需要设置此参数，此参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	False	为 True 时，表示插补过程中出现错误。
ErrorID	错误码	DINT	0，正数	0	当 Error 为 True 时，该变量输出错误码

功能说明：除相对坐标外，其他与 LS_2AxisEllipsesAbs 指令相同。

8.1.5 螺旋插补运动指令

PMC600 运动控制器目前支持相对模式和绝对模式三轴圆弧螺旋线插补，该指令可理解为其中两轴做圆弧插补，另外一轴在垂直方向做跟随运动。相关指令如表 8.5 所示。

其中圆弧有 4 种模式：

模式 0：三点圆弧模式，辅助点输入圆弧经过的点坐标；

模式 1：圆心与终点模式，辅助点输入圆心的坐标；

模式 2：终点半径模式，需要输入圆弧的半径；

模式 3：目标角度模式，停止位置为相对于起点的位置，需要输入圆弧的方向和圆心坐标。

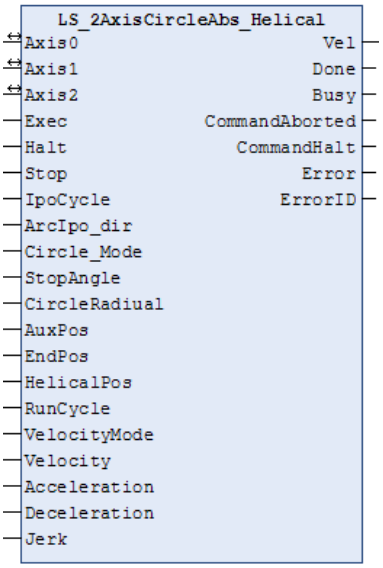
表 8.5　螺旋插补运动指令

指令名	功能说明
LS_2AxisCircleAbs_Helical	绝对坐标三轴圆弧螺旋线插补
LS_2AxisCircleRel_Helical	绝对坐标三轴圆弧螺旋线插补

两轴绝对螺旋插补 LS_2AxisCircleAbs_Helical

三轴圆弧螺旋线插补指令，采用绝对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_2AxisCircleAbs_Helical	FB		LS_2AxisCircleAbs_Helical(Axis0:= （参数）, Axis1:= （参数）, Axis2:= （参数）, Exec:= （参数）, Halt:= （参数）, Stop:= （参数）, IpoCycle:= （参数）, ArcIpo_dir:= （参数）, Circle_Mode:= （参数）, StopAngle:= （参数）, CircleRadiual:= （参数）, AuxPos:= （参数）, EndPos:= （参数）, HelicalPos:= （参数）, RunCycle:= （参数）, VelocityMode:= （参数）, Velocity:= （参数）, Acceleration:= （参数）, Deceleration:= （参数）, Jerk:= （参数）, Vel=> （参数）, Done=> （参数）, Busy=> （参数）, CommandAborted=> （参数）, CommandHalt=> （参数）, Error=> （参数）, ErrorID=> （参数）);

变量：

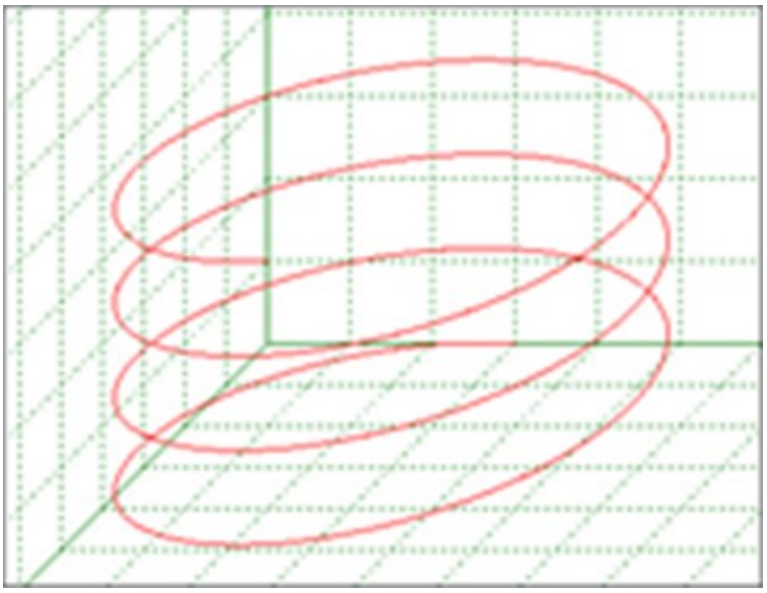
VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
Axis2	轴 2	AXIS_REF_VIRTUAL_SM3	-	-	插补 2 轴
VAR_INPUT					

Exec	启动	BOOL	TRUE，FALSE	False	螺旋线插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
ArcIpo_dir	插补方向	INT	0，1	0	圆弧插补的方向，0 表示逆时针方向，1 表示顺时针方向。圆弧模式 1、2、3 需要设置该值
Circle_Mode	插补模式	INT	0-3	1	圆弧模式：0 表示三点圆弧模式；1 表示圆心位置与终点确定圆弧；2 表示终点半径模式；3 表示目标角度模式
StopAngle	停止角度	REAL	0-360	0	停止的目标角度，相对于起点的角度，单位为度，取值范围为[0，360.0]。圆弧模式 3 需要设置该值
CircleRadiual	圆弧半径	REAL	正数	0	圆弧半径，圆弧模式 2 需要设置该值。该值半径为负表示大圆弧，半径为正表示小圆弧
AuxPos	辅助位置	ARRAY [0..1] OF REAL	负数，0 正数	0	辅助点，单位：Pulse
Endpos	停止位置	ARRAY [0..1] OF REAL	负数，0 正数	0	终点位置，单位：Pulse
HelicalPos	螺旋轴位置	REAl	负数，0 正数	0	螺旋轴位置，单位：Pulse
RunCycle	循环次数	UINT	0，正数	0	圆弧运动的循环次数（即圆弧旋转的圈数）
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时候需要设置此参数，此参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。

CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	False	为 True 时，表示插补过程中出现错误。
ErrorID	错误码	DINT	0，正数	0	当 Error 为 True 时，该变量输出错误码

功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- 指令指定的三个轴做空间螺旋插补运动，位置坐标为绝对坐标。
- 螺旋插补的轨迹，如图 8.8 所示。



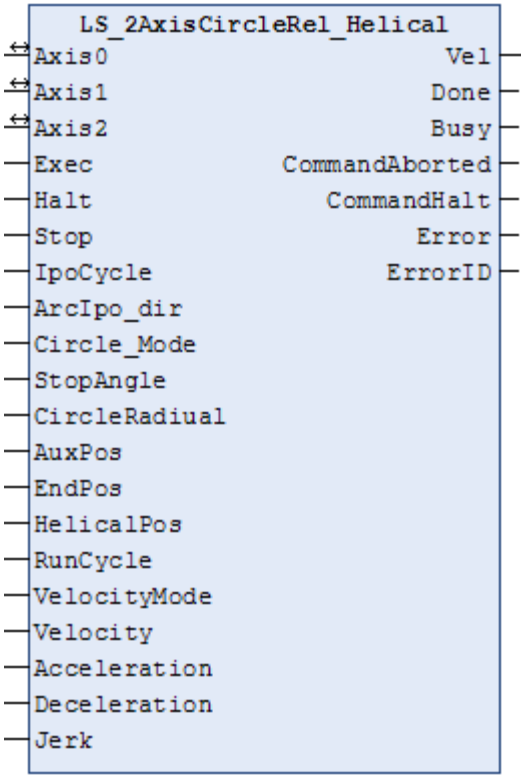
8.8　螺旋插补轨迹

- 在螺旋插补运动之中，轴 0 和轴 1 用于实现圆弧运动，“Circle_Mode”参数用来设置轴 0、轴 1 的圆弧插补的模式，定义与两轴圆弧插补指令相同。

两轴相对螺旋插补 LS_2AxisCircleRel_Helical

三轴圆弧螺旋线插补指令，采用相对坐标。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_2AxisCircleRel_Helical	FB		LS_2AxisCircleRel_Helical(Axis0：= （参数）， Axis1：= （参数）， Axis2：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， ArcIpo_dir：= （参数）， Circle_Mode：= （参数）， StopAngle：= （参数）， CircleRadiual：= （参数）， AuxPos：= （参数）， EndPos：= （参数）， HelicalPos：= （参数）， RunCycle：= （参数）， VelocityMode：= （参数）， Velocity：= （参数）， Acceleration：= （参数）， Deceleration：= （参数）， Jerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 0 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 1 轴
Axis2	轴 2	AXIS_REF_VIRTUAL_SM3	-	-	插补 2 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	螺旋线插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周	DWORD	-	2000	插补周期，单位：us

	期				
ArcIpo_dir	插补方向	INT	0，1	0	圆弧插补的方向，0 表示逆时针方向，1 表示顺时针方向。圆弧模式 1、2、3 需要设置该值
Circle_Mode	插补模式	INT	0-3	1	圆弧模式：0 表示三点圆弧模式；1 表示圆心位置与终点确定圆弧；2 表示终点半径模式；3 表示目标角度模式
StopAngle	停止角度	REAL	0-360	0	停止的目标角度，相对于起点的角度，单位为度，取值范围为[0，360.0]。圆弧模式 3 需要设置该值
CircleRadiual	圆弧半径	REAL	正数	0	圆弧半径，圆弧模式 2 需要设置该值。该值半径为负表示大圆弧，半径为正表示小圆弧
AuxPos	辅助位置	ARRAY [0..1] OF REAL	负数，0 正数	0	辅助点，增量坐标，单位：Pulse
Endpos	停止位置	ARRAY [0..1] OF REAL	负数，0 正数	0	终点位置，增量坐标，单位：Pulse
HelicalPos	螺旋轴位置	REAl	负数，0 正数	0	螺旋轴位置，增量坐标，单位：Pulse
RunCycle	循环次数	UINT	0，正数	0	圆弧运动的循环次数（即圆弧旋转的圈数）
VelocityMode	速度模式	SMC_INT_VELMODE	0-3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Velocity	速度	LREAL	0，正数	0	插补合成速度
Acceleration	加速度	LREAL	0，正数	0	插补合成加速度
Deceleration	减速度	LREAL	0，正数	0	插补合成减速度
Jerk	加加速度	LREAL	负数，0 正数	90000000	加加速度，速度模式为 3 时候需要设置此参数，此参数值不能为 0
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	False	为 True 时，表示插补过程中出现错误。
ErrorID	错误码	DINT	0，正数	0	当 Error 为 True 时，该变量输出错误码

功能说明：除相对坐标外，其他与 LS_2AxisCircleAbs_Helical 指令相同。

8.1.6 连续插补指令

前面所列举的雷赛插补指令，都是单段插补指令，使用时，每走一个位置，都需要调用一次指令。每一段插补运动在距离足够长的情况下，都是一个完整的，包含了加速、匀速、减速过程的运动。与前后衔接的运动指令在速度上相互独立。这类插补指令，适用于一般的运动场合。

而连续插补指令，则采用了缓存方式存放各段插补运动的坐标点。运行指令时，需要先将各线段坐标点压入缓存区，规划出运动轨迹，再进行连续插补运动。与单段插补指令不同的，是连续插补指令只要设置的插补点位置合适，走出来的整段轨迹的合速度将会是连续的。

PMC600 运动控制器支持三轴连续插补和四轴连续插补功能。四轴连续插补其实是三轴连续插补加另外一轴做跟随运动，但速度不连续。

连续插补指令可在程序中直接调用执行，参数 IpoCycle、Jerk、LimitMaxAcc、LimitMaxAccJerk、RoundDiameter 不可以设置为 0。

相关指令如表 8.6 所示。

表 8.6 雷赛连续插补指令

名称	功能
LS_3AxisMoveSequence	三轴连续插补运动
LS_4AxisMoveSequence	四轴连续插补运动

Ipolib 库中定义了连续插补中一次性最多可输入点的个数的全局变量 MoveSequence_MAX 以及连续插补缓存队列结构体 MoveSequence，目前最多支持 500 个点，缓存队列结构体 MoveSequence 具体参数如表 8.7 所示：

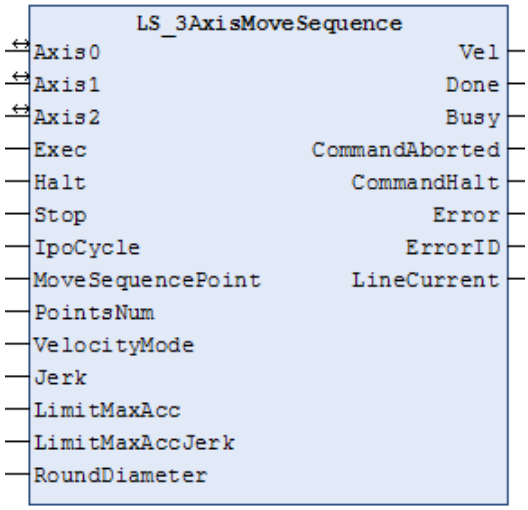
表 8.7 结构体 MoveSequence 具体参数

名称	类型	初始化	注释
LineNum	LREAL	0	数据行号
DataType	LREAL	0	数据类型 1：直线运动；
X_EndPos	LREAL	0	X 终点坐标，绝对坐标，脉冲数
Y_EndPos	LREAL	0	Y 终点坐标，绝对坐标，脉冲数
Z_EndPos	LREAL	0	Z 终点坐标，绝对坐标，脉冲数
U_EndPos	LREAL	0	U 终点坐标，绝对坐标，脉冲数
Vel	LREAL	100	合成速度，Pulse/s
Acc	LREAL	1000	合成加速度，Pulse/s^2
Aux_Pos	ARRAY [0..1] OF LREAL	[0，0]	辅助点坐标

三轴连续插补 LS_3AxisMoveSequence

三轴连续插补，速度连续。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_3AxisMoveSequence	FB		LS_3AxisMoveSequence(Axis0:= （参数）， Axis1:= （参数）， Axis2:= （参数）， Exec:= （参数）， Halt:= （参数）， Stop:= （参数）， IpoCycle:= （参数）， MoveSequencePoint:=（参数）， PointsNum:= （参数）， VelocityMode:= （参数）， Jerk:= （参数）， LimitMaxAcc:= （参数）， LimitMaxAccJerk:= （参数）， RoundDiameter:= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）， LineCurrent=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 X 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 Y 轴
Axis2	轴 2	AXIS_REF_VIRTUAL_SM3	-	-	插补 Z 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	连续插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位 us
MoveSequencePoint	插补数据表	ARRAY [0..MoveSequence_MAX] OF	-	-	输入的数据点

		MoveSequence			
PointsNum	点数	UINT	正数	0	输入的数据点数
VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	1	速度模式，0： 梯形（TRAPEZOID）；1：S 型曲线（SIGMOID）；3：四次方（QUADRATIC）
Jerk	加加速度	REAL	正数	90000000	加加速度，速度模式为 3 时候需要设置此参数
LimitMaxAcc	最大加速度	REAL	正数	9000000	拐角过渡圆弧最大加速度限制
LimitMaxAccJerk	最大减速度	REAL	正数	90000000	拐角过渡圆弧最大加加速度限制
RoundDiameter	拐角半径	REAL	正数	100	拐角过渡圆弧半径，单位：Pulse
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态
Error	报错	BOOL	TRUE，FALSE	False	错误码，True 表示插补过程中出现错误
ErrorID	错误码	DINT	0，正数	0	当 Error 为 True 时，该变量输出错误码
LineCurent		UINT		0	当前执行的行号

功能说明：

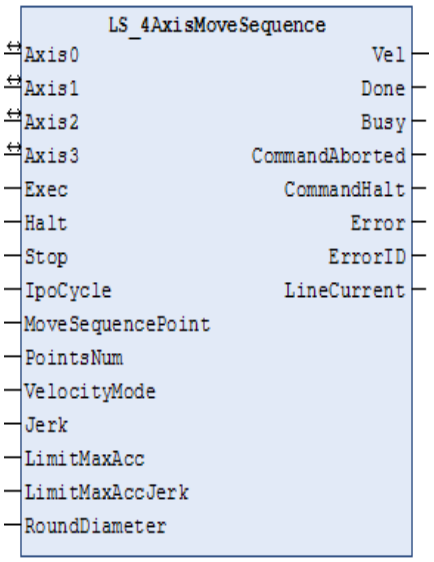
- 这个指令由“PMC_Ipolib”库实现。
- 指定 3 个轴做连续插补运动，位置坐标为绝对坐标，一次最多可以输入 200 个插补运动点。
- 在连续插补运动中，每个运动点位，都是以 MoveSequence 结构体的形式存在的，具体的参数设置说明，“MoveSequence”结构体参数如表 8.7 所示。
- 当选择的插补类型为直线插补时，如果相邻两段插补运动的轨迹相切或拐角时，规划出来的合速度便会是连续的，如果拐角太大，则速度不连续；当选择的插补类型为圆弧插补时，相邻两段插补运动的轨迹在交接点处如果是相切的，则速度连续，如果不相切时，则速度不连续。
- 将指令的 Exec 信号置为 TRUE，便能启动连续插补指令。
- 启动连续插补指令后，指令内部需要先进行压缓存的操作，插补点数越多，压缓存的时间会越长。

- 注意，IpoCycle 参数需要与运动指令所在的任务周期时间保持一致，否则指令执行时可能会出现 Error 报错。
- 参数 IpoCycle、Jerk、Velocity、Acceleration、Deceleration 不可以设置为 0。
- 在指令运行期间，主从轴都不能再被其它运动指令所调用。

四轴连续插补 LS_4AxisMoveSequence

四轴连续插补，速度连续。

指令外观：

指令	FB/ FUN	图形模块	结构文本
LS_4AxisMove Sequence	FB		LS_4AxisMoveSequence(Axis0:= （参数）， Axis1:= （参数）， Axis2:= （参数）， Axis3:= （参数）， Exec:= （参数）， Halt:= （参数）， Stop:= （参数）， IpoCycle:= （参数）， MoveSequencePoint:= （参数）， PointsNum:= （参数）， VelocityMode:= （参数）， Jerk:= （参数）， LimitMaxAcc:= （参数）， LimitMaxAccJerk:= （参数）， RoundDiameter:= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）， LineCurrent=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
Axis0	轴 0	AXIS_REF_VIRTUAL_SM3	-	-	插补 X 轴
Axis1	轴 1	AXIS_REF_VIRTUAL_SM3	-	-	插补 Y 轴
Axis2	轴 2	AXIS_REF_VIRTUAL_SM3	-	-	插补 Z 轴
Axis3	轴 3	AXIS_REF_VIRTUAL_SM3	-	-	插补 U 轴，跟随轴，速度不连续
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	连续插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	暂停命令，当该输入量为 True 时，当前插补运动暂

					停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位 us
MoveSequencePoint	插补数据表	ARRAY [0..MoveSequence_MAX] OF MoveSequence	-	-	输入的数据点
PointsNum	点数	UINT	正数	0	输入的数据点数
VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	1	速度模式，0： 梯形（TRAPEZOID）；1： S 型曲线（SIGMOID）；3：四次方（QUADRATIC）
Jerk	加加速度	REAL	正数	90000000	加加速度，Pulse/s2，速度模式为 3 时候需要设置此参数，此参数值不能为 0
LimitMaxAcc	最大加速度	REAL	正数	9000000	拐角过渡圆弧最大加速度限制
LimitMaxAccJerk	最大减速度	REAL	正数	90000000	拐角过渡圆弧最大加加速度限制
RoundDiameter	拐角半径	REAL	正数	100	拐角过渡圆弧半径，单位：Pulse
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	False	错误码，True 表示插补过程中出现错误
ErrorID	错误码	DINT	0，正数	0	当 Error 为 True 时，该变量输出错误码
LineCurent		UINT		0	当前执行的行号

功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- 指定 Axis0-Axis2 做连续插补运动，Axis3 做跟随运动，位置坐标为绝对坐标，一次最多可以输入 200 个插补运动点。
- 其他参数的设置和 LS_3AxisMoveSequence 指令相同。

8.1.7 G 代码插补运动指令

PMC600 运动控制器支持 G 代码进行插补运动。运动指令包括三轴 G 代码连续插补、四轴 G 代码连续插补和六轴 G 代码连续插补。G 代码能以文件方式导入，也可以在程序内部直接编写 G 代码。相关指令如表 8.8 所示。

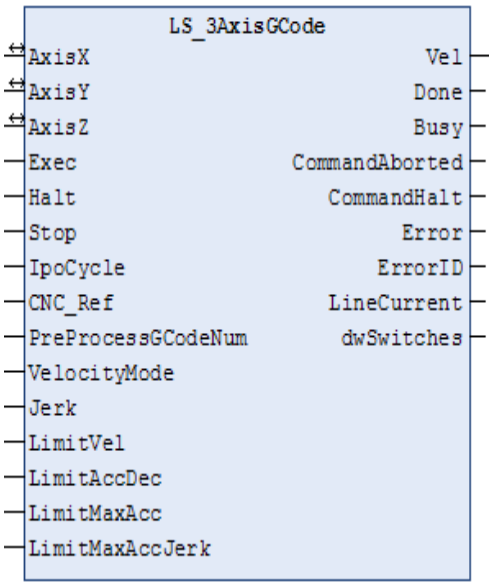
表 8.8 G 代码插补运动指令

指令名	功能说明
LS_3AxisGCode	三轴 G 代码连续插补（G 代码形式），支持坐标轴 X、Y、Z
LS_3AxisGCode_File	三轴 G 代码连续插补（G 代码文件形式），支持坐标轴 X、Y、Z
LS_4AxisGCode	四轴 G 代码连续插补（G 代码形式），支持坐标轴 X、Y、Z、U
LS_4AxisGCode_File	四轴 G 代码连续插补（G 代码文件形式），支持坐标轴 X、Y、Z、U
LS_4AxisGCodeAxisP	四轴 G 代码连续插补（G 代码形式），支持坐标轴 X、Y、Z、P
LS_4AxisGCodeAxisP_File	四轴 G 代码连续插补（G 代码文件形式），支持坐标轴 X、Y、Z、P
LS_6AxisGCodeAxisUVW_File	六轴 G 代码连续插补（G 代码文件形式），支持坐标轴 X、Y、Z、U、V、W
LS_6Axis_ZeroOffset	零点坐标偏移（相当于 G54）

三轴 G 代码插补 LS_3AxisGCode

G 代码三轴连续插补，支持坐标轴 X、Y、Z。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_3AxisGCode	FB		LS_3AxisGCode(AxisX：= （参数）， AxisY：= （参数）， AxisZ：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， CNC_Ref：= （参数）， PreProcessGCodeNum:=（参数）， VelocityMode：= （参数）， Jerk：= （参数）， LimitVel：= （参数）， LimitAccDec：= （参数）， LimitMaxAcc：= （参数）， LimitMaxAccJerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）， LineCurrent=> （参数）， dwSwitches=> （参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
AxisX	G 代码 X 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 X 轴
AxisY	G 代码 Y 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Y 轴
AxisZ	G 代码 Z 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Z 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
CNC_Ref	G 文件指针	POINTER TO SMC_CNC_REF	-	-	G 代码文件指针
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数

VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Jerk	加加速度	LREAL	正数	90000000	加加速度，Pulse/s3，速度模式设置为 3 时需要设置该参数，此参数值不能为 0
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
LimitAccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
LimitMaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态
Error	报错	BOOL	TRUE，FALSE	False	错误码，True 表示插补过程中出现错误
ErrorID	错误码	DINT	0，正数	0	当 Error 为 True 时，该变量输出错误码
LineCurrent		INT		0	当前的执行的行
dwSwitches		DWORD		0	当前 H 代码对应的开关状态

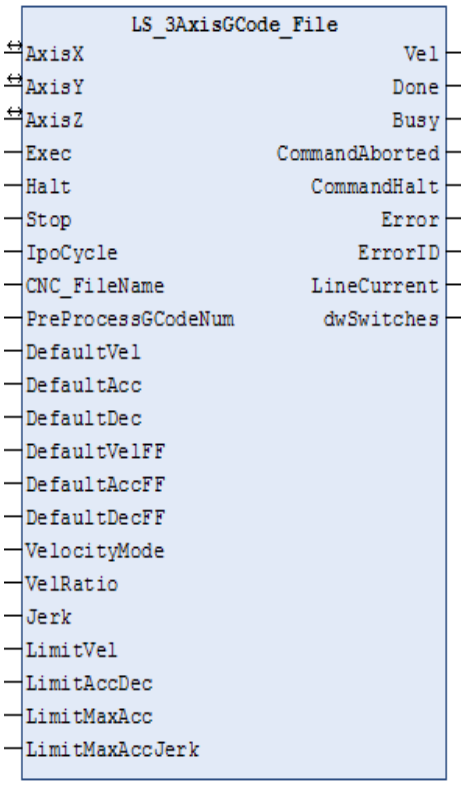
功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- 参数 IpoCycle、Jerk、PreProcessGCodeNum、LimitVel、LimitAccDec、LimitMaxAcc、LimitMaxAccJerk 不可以设置为 0。

三轴 G 代码文件插补 LS_3AxisGCode_File

G 代码插补指令，三轴连续插补，支持坐标轴 X、Y、Z。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_3AxisGCode_File	FB		LS_3AxisGCode_File(AxisX：= （参数）， AxisY：= （参数）， AxisZ：= （参数）， Exec：= （参数）， Halt：= （参数）， Stop：= （参数）， IpoCycle：= （参数）， CNC_FileName：= （参数）， PreProcessGCodeNum：= （参数）， DefaultVel：= （参数）， DefaultAcc：= （参数）， DefaultDec：= （参数）， DefaultVelFF：= （参数）， DefaultAccFF：= （参数）， DefaultDecFF：= （参数）， VelocityMode：= （参数）， VelRatio：= （参数）， Jerk：= （参数）， LimitVel：= （参数）， LimitAccDec：= （参数）， LimitMaxAcc：= （参数）， LimitMaxAccJerk：= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）， LineCurrent=> （参数）， dwSwitches=> （参数）);

变量：

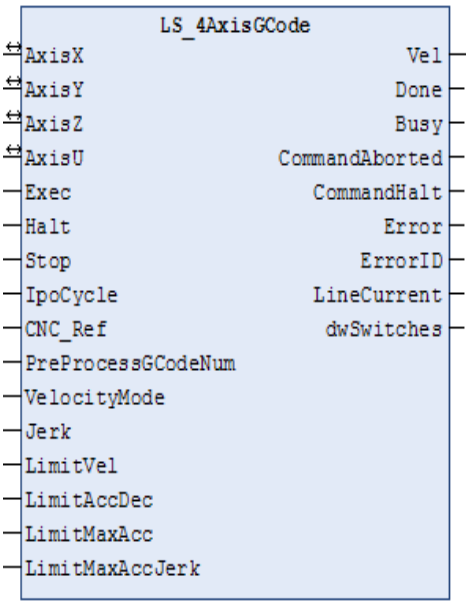
VAR_IN_OUT	名称	类型	有效范围	初始化	注释
AxisX	G 代码 X 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 X 轴
AxisY	G 代码 Y 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Y 轴
AxisZ	G 代码 Z 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Z 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	使能
Halt	暂停	BOOL	TRUE，FALSE	False	暂停
Stop	停止	BOOL	TRUE，FALSE	False	停止

IpoCycle	插补周期	DWORD	-	2000	插补时间，单位 us
CNC_FileName	G 代码文件名	STRING	-	-	G 代码文件名称
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
DefaultVel	默认速度	LREAL	正数	10	在 CNC 文件中，没有指定 F 的情况下，默认的 F 值
DefaultAcc	默认加速度	LREAL	正数	10	在 CNC 文件中，没有指定加速度的情况下，默认的加速度值
DefaultDec	默认减速度	LREAL	正数	10	在 CNC 文件中，没有指定减速度的情况下，默认的减速度值
DefaultVelFF	默认 G01 速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的 F 情况下，默认的 F 值
DefaultAccFF	默认 G01 加速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的加速度的情况下，默认的加速度值
DefaultDecFF	默认 G01 减速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的减速度的情况下，默认的减速度值
VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值 0.01，最大值 2
Jerk	加加速度	LREAL	正数	90000000	加加速度，Pulse/s3，速度模式设置为 3 时需要设置该参数，该参数值不能为 0
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
LimitAccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
LimitMaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合速度
Done	完成	BOOL	TRUE，FALSE	FALSE	表示插补结束
Busy	运动中	BOOL	TRUE，FALSE	FALSE	插补进行中
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	插补终止
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	插补暂停
Error	报错	BOOL	TRUE，	False	插补出错

			FALSE		
ErrorID	错误码	DINT	0，正数	0	插补错误码
LineCurrent	当前行号	INT	0，正数	0	当前的执行的行
dwSwitches	开关状态	DWORD	0，正数	0	当前 H 代码对应的开关状态

四轴 G 代码插补 LS_4AxisGCode

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_4AxisGCode	FB		LS_4AxisGCode(AxisX:= （参数）， AxisY:= （参数）， AxisZ:= （参数）， AxisU:= （参数）， Exec:= （参数）， Halt:= （参数）， Stop:= （参数）， IpoCycle:= （参数）， CNC_Ref:= （参数）， PreProcessGCodeNum:= （参数）， VelocityMode:= （参数）， Jerk:= （参数）， LimitVel:= （参数）， LimitAccDec:= （参数）， LimitMaxAcc:= （参数）， LimitMaxAccJerk:= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）， LineCurrent=> （参数）， dwSwitches=> （参数）);

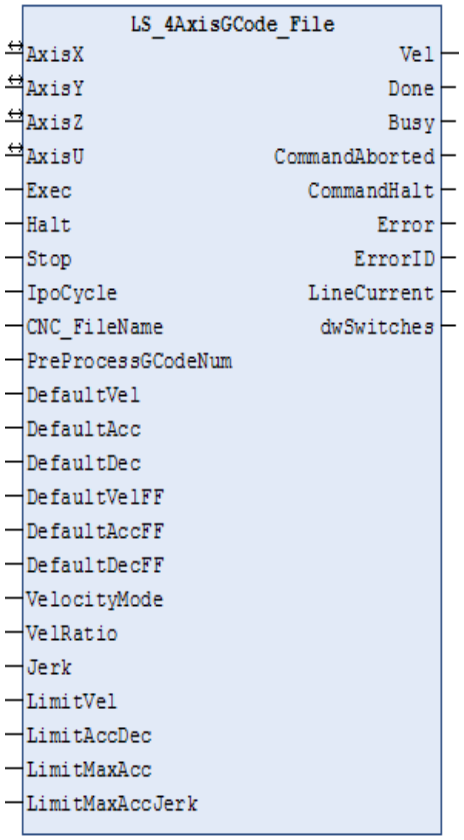
变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	功能描述
AxisX	G 代码 X 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 X 轴
AxisY	G 代码 Y 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Y 轴
AxisZ	G 代码 Z 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Z 轴
AxisU	G 代码 U 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 U 轴

VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	插补模块启动命令，上升沿有效
Halt	暂停	BOOL	TRUE，FALSE	False	插补暂停命令，当该输入量为 True 时，当前插补运动暂停；当状态再次切换到 False 时，插补模块继续之前的未完成的插补任务。
Stop	停止	BOOL	TRUE，FALSE	False	停止命令，该命令为 True 时，插补模块停止。
IpoCycle	插补周期	DWORD	-	2000	插补周期，单位：us
CNC_Ref	G 文件指针	POINTER TO SMC_CNC_REF	-	-	G 代码文件指针
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Jerk	加加速度	LREAL	正数	90000000	加加速度，Pulse/s3，速度模式设置为 3 时需要设置该参数，此参数值不能为 0
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
LimitAccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
LimitMaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合成速度
Done	完成	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补模块已完成插补任务
Busy	运动中	BOOL	TRUE，FALSE	FALSE	为 True 时，表示插补运动正在进行之中。
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	当 Stop 有效时，该变量为 True，表示插补模块非正常中止。
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	当 Halt 为 True 时，该变量为 True，表示插补模块处于暂停状态。
Error	报错	BOOL	TRUE，FALSE	False	错误码，True 表示插补过程中出现错误。
ErrorID	错误码	DINT	0，正数	0	当 Error 为 True 时，该变量输出错误码
LineCurrent	当前行号	INT	0，正数	0	当前的执行的行
dwSwitches	开关状态	DWORD	0，正数	0	当前 H 代码对应的开关状态

四轴 G 代码文件插补 LS_4AxisGCode_File

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_4AxisGCode_File	FB		LS_4AxisGCode_File(AxisX:= （参数）， AxisY:= （参数）， AxisZ:= （参数）， AxisU:= （参数）， Exec:= （参数）， Halt:= （参数）， Stop:= （参数）， IpoCycle:= （参数）， CNC_FileName:= （参数）， PreProcessGCodeNum:= （参数）， DefaultVel:= （参数）， DefaultAcc:= （参数）， DefaultDec:= （参数）， DefaultVelFF:= （参数）， DefaultAccFF:= （参数）， DefaultDecFF:= （参数）， VelocityMode:= （参数）， VelRatio:= （参数）， Jerk:= （参数）， LimitVel:= （参数）， LimitAccDec:= （参数）， LimitMaxAcc:= （参数）， LimitMaxAccJerk:= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）， LineCurrent=> （参数）， dwSwitches=> （参数）);

变量：

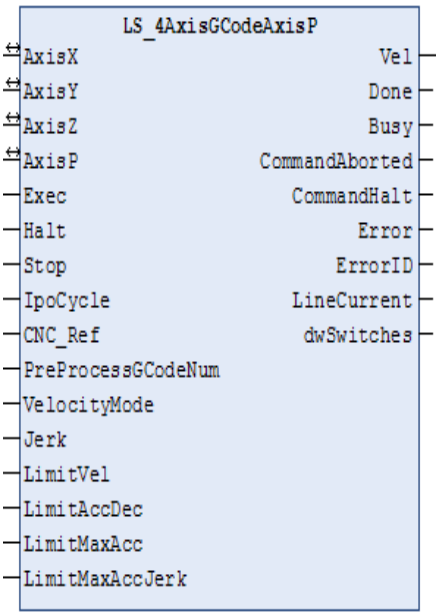
VAR_IN_OUT	名称	类型	有效范围	初始化	注释
AxisX	G 代码 X 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 X 轴
AxisY	G 代码 Y 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Y 轴
AxisZ	G 代码 Z 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Z 轴
AxisU	G 代码 U 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 U 轴

VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	使能
Halt	暂停	BOOL	TRUE，FALSE	False	暂停
Stop	停止	BOOL	TRUE，FALSE	False	停止
IpoCycle	插补周期	DWORD	-	2000	插补时间，单位 us
CNC_FileName	G 代码文件名	STRING	-	-	G 代码文件
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
DefaultVel	默认速度	LREAL	正数	10	在 CNC 文件中，没有指定 F 的情况下，默认的 F 值
DefaultAcc	默认加速度	LREAL	正数	10	在 CNC 文件中，没有指定加速度的情况下，默认的加速度值
DefaultDec	默认减速度	LREAL	正数	10	在 CNC 文件中，没有指定减速度的情况下，默认的减速度值
DefaultVelFF	默认 G01 速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的 F 情况下，默认的 F 值
DefaultAccFF	默认 G01 加速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的加速度的情况下，默认的加速度值
DefaultDecFF	默认 G01 减速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的减速度的情况下，默认的减速度值
VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值 0.01，最大值 2
Jerk	加加速度	LREAL	正数	90000000	加加速度，Pulse/s3，速度模式设置为 3 时需要设置该参数，该参数值不能为 0
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
LimitAccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
LimitMaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合速度
Done	完成	BOOL	TRUE，FALSE	FALSE	表示插补结束
Busy	运动中	BOOL	TRUE，FALSE	FALSE	插补进行中

CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	插补终止
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	插补暂停
Error	报错	BOOL	TRUE，FALSE	False	插补出错
ErrorID	错误码	DINT	0，正数	0	插补错误码
LineCurrent	当前行号	INT	0，正数	0	当前的执行的行
dwSwitches	开关状态	DWORD	0，正数	0	当前 H 代码对应的开关状态

四轴 G 代码插补 LS_4AxisGCodeAxisP

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_4AxisGCode AxisP	FB		LS_4AxisGCodeAxisP(AxisX:= （参数）， AxisY:= （参数）， AxisZ:= （参数）， AxisP:= （参数）， Exec:= （参数）， Halt:= （参数）， Stop:= （参数）， IpoCycle:= （参数）， CNC_Ref:= （参数）， PreProcessGCodeNum:= （参数）， VelocityMode:= （参数）， Jerk:= （参数）， LimitVel:= （参数）， LimitAccDec:= （参数）， LimitMaxAcc:= （参数）， LimitMaxAccJerk:= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）， LineCurrent=> （参数）， dwSwitches=> （参数）);

变量：

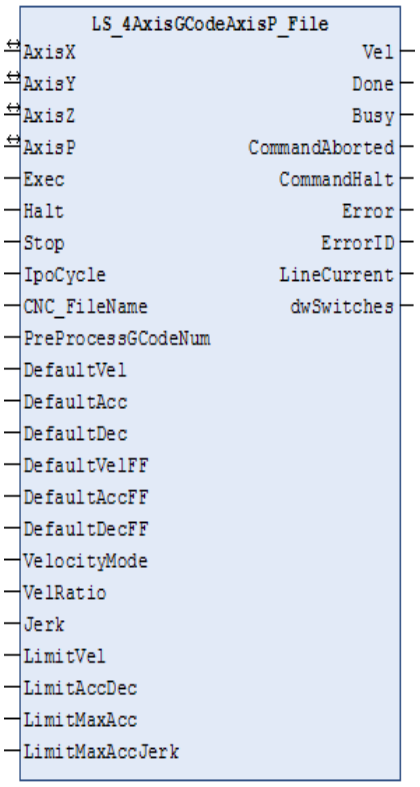
VAR_IN_OUT	名称	类型	有效范围	初始化	注释
AxisX	G 代码 X 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 X 轴
AxisY	G 代码 Y 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Y 轴
AxisZ	G 代码 Z 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Z 轴
AxisP	G 代码	AXIS_REF_VI	-	-	对应 G 代码 P 轴

	P 轴	RTUAL_SM3			
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	使能
Halt	暂停	BOOL	TRUE，FALSE	False	暂停
Stop	停止	BOOL	TRUE，FALSE	False	停止
IpoCycle	插补周期	DWORD	-	2000	插补时间，单位 us
CNC_Ref	G 文件指针	POINTER TO SMC_CNC_REF	-	-	G 代码文件指针
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
Jerk	加加速度	LREAL	正数	90000000	加加速度，Pulse/s3，速度模式设置为 3 时需要设置该参数，该参数值不能为 0
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
LimitAccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
LimitMaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合速度
Done	完成	BOOL	TRUE，FALSE	FALSE	表示插补结束
Busy	运动中	BOOL	TRUE，FALSE	FALSE	插补进行中
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	插补终止
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	插补暂停
Error	报错	BOOL	TRUE，FALSE	False	插补出错
ErrorID	错误码	DINT	0，正数	0	插补错误码
LineCurrent	当前行号	INT	0，正数	0	当前的执行的行

dwSwitches	开关状态	DWORD	0，正数	0	当前 H 代码对应的开关状态

四轴 G 代码文件插补 LS_4AxisGCodeAxisP_File

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_4AxisGCodeAxisP_File	FB		LS_4AxisGCodeAxisP_File(AxisX:= （参数）， AxisY:= （参数）， AxisZ:= （参数）， AxisP:= （参数）， Exec:= （参数）， Halt:= （参数）， Stop:= （参数）， IpoCycle:= （参数）， CNC_FileName:= （参数）， PreProcessGCodeNum:= （参数）， DefaultVel:= （参数）， DefaultAcc:= （参数）， DefaultDec:= （参数）， DefaultVelFF:= （参数）， DefaultAccFF:= （参数）， DefaultDecFF:= （参数）， VelocityMode:= （参数）， VelRatio:= （参数）， Jerk:= （参数）， LimitVel:= （参数）， LimitAccDec:= （参数）， LimitMaxAcc:= （参数）， LimitMaxAccJerk:= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）， LineCurrent=> （参数）， dwSwitches=> （参数）);

变量：

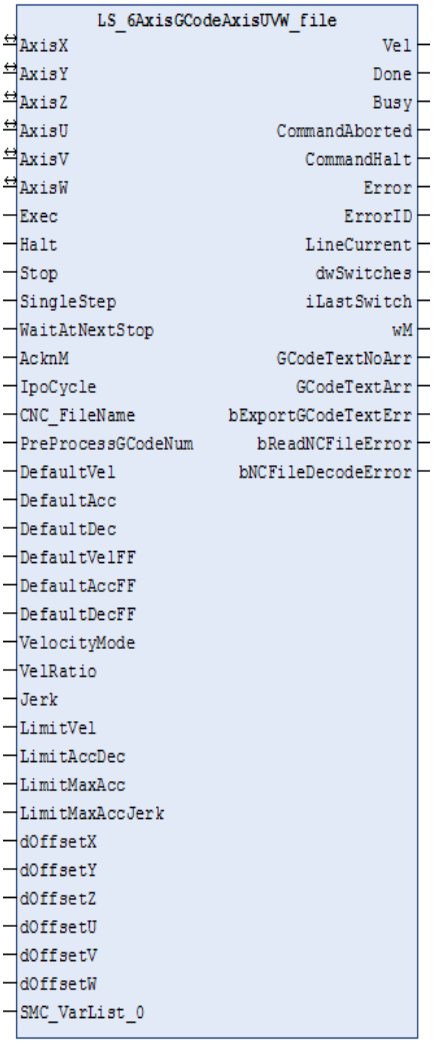
VAR_IN_OUT	名称	类型	有效范围	初始化	注释
AxisX	G 代码 X 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 X 轴
AxisY	G 代码 Y 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Y 轴
AxisZ	G 代码 Z 轴	AXIS_REF_VIRTUAL_S	-	-	对应 G 代码 Z 轴

		M3			
AxisP	G 代码 P 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 P 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	使能
Halt	暂停	BOOL	TRUE，FALSE	False	暂停
Stop	停止	BOOL	TRUE，FALSE	False	停止
IpoCycle	插补周期	DWORD	-	2000	插补时间，单位：us
CNC_FileName	G 代码文件名	STRING	-	-	G 代码文件
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
DefaultVel	默认速度	LREAL	正数	10	在 CNC 文件中，没有指定 F 的情况下，默认的 F 值
DefaultAcc	默认加速度	LREAL	正数	10	在 CNC 文件中，没有指定加速度的情况下，默认的加速度值
DefaultDec	默认减速度	LREAL	正数	10	在 CNC 文件中，没有指定减速度的情况下，默认的减速度值
DefaultVelFF	默认 G01 速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的 F 情况下，默认的 F 值
DefaultAccFF	默认 G01 加速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的加速度的情况下，默认的加速度值
DefaultDecFF	默认 G01 减速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的减速度的情况下，默认的减速度值
VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值 0.01，最大值 2
Jerk	加加速度	LREAL	正数	90000000	加加速度，Pulse/s3，速度模式设置为 3 时需要设置该参数，该参数值不能为 0
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
LimitAccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
LimitMaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合速度
Done	完成	BOOL	TRUE，FALSE	FALSE	表示插补结束

Busy	运动中	BOOL	TRUE，FALSE	FALSE	插补进行中
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	插补终止
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	插补暂停
Error	报错	BOOL	TRUE，FALSE	False	插补出错
ErrorID	错误码	DINT	0，正数	0	插补错误码
LineCurrent	当前行号	INT	0，正数	0	当前的执行的行
dwSwitches	开关状态	DWORD	0，正数	0	当前 H 代码对应的开关状态

六轴 G 代码文件插补 LS_6AxisGCodeAxisUVW_file

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_6AxisGCodeAxisUVW_File	FB		LS_6AxisGCodeAxisUVW_File(AxisX:= （参数）， AxisY:= （参数）， AxisZ:= （参数）， AxisU:= （参数）， AxisV:= （参数）， AxisW:= （参数）， Exec:= （参数）， Halt:= （参数）， Stop:= （参数）， SingleStep:= （参数）， WaitAtNextStop:= （参数）， AcknM:= （参数）， IpoCycle:= （参数）， CNC_FileName:= （参数）， PreProcessGCodeNum:=（参数）， DefaultVel:= （参数）， DefaultAcc:= （参数）， DefaultDec:= （参数）， DefaultVelFF:= （参数）， DefaultAccFF:= （参数）， DefaultDecFF:= （参数）， VelocityMode:= （参数）， VelRatio:= （参数）， Jerk:= （参数）， LimitVel:= （参数）， LimitAccDec:= （参数）， LimitMaxAcc:= （参数）， LimitMaxAccJerk:= （参数）， dOffsetX:= （参数）， dOffsetY:= （参数）， dOffsetZ:= （参数）， dOffsetU:= （参数）， dOffsetV:= （参数）， dOffsetW:= （参数）， SMC_VarList_0:= （参数）， Vel=> （参数）， Done=> （参数）， Busy=> （参数）， CommandAborted=> （参数）， CommandHalt=> （参数）， Error=> （参数）， ErrorID=> （参数）， LineCurrent=> （参数）， dwSwitches=> （参数）， iLastSwitch=> （参数）， wM=> （参数）， GCodeTextNoArr=> （参数）， GCodeTextArr=> （参数）， bExportGCodeTextErr=> （参数）， bReadNCFileError=> （参数）， bNCFileDecodeError=> （参数）);

变量：

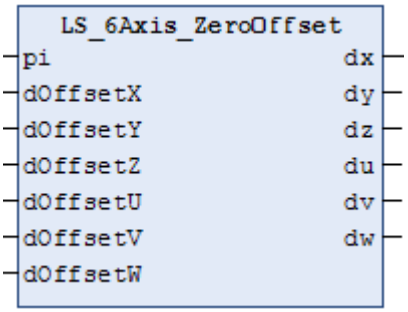
VAR_IN_OUT	名称	类型	有效范围	初始化	注释
AxisX	G 代码 X 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 X 轴
AxisY	G 代码 Y 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Y 轴
AxisZ	G 代码 Z 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 Z 轴
AxisU	G 代码 U 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 U 轴
AxisV	G 代码 V 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 V 轴
AxisW	G 代码 W 轴	AXIS_REF_VIRTUAL_SM3	-	-	对应 G 代码 W 轴
VAR_INPUT					
Exec	启动	BOOL	TRUE，FALSE	False	使能
Halt	暂停	BOOL	TRUE，FALSE	False	暂停
Stop	停止	BOOL	TRUE，FALSE	False	停止
SingleStep	单步标志	BOOL	TRUE，FALSE	False	单步标志
WaitAtNextStop	单步启动	BOOL	TRUE，FALSE	False	单步启动
AcknM	M code 处理标志	BOOL	TRUE，FALSE	False	M code 处理标志
IpoCycle	插补周期	DWORD	-	2000	插补时间 单位 us
CNC_FileName	G 代码文件名	STRING	-	-	G 代码文件
PreProcessGCodeNum	每周期段数	INT	正数	1	每周期读取 G 代码段数
DefaultVel	默认速度	LREAL	正数	10	在 CNC 文件中，没有指定 F 的情况下，默认的 F 值
DefaultAcc	默认加速度	LREAL	正数	10	在 CNC 文件中，没有指定加速度的情况下，默认的加速度值
DefaultDec	默认减速度	LREAL	正数	10	在 CNC 文件中，没有指定减速度的情况下，默认的

					减速度值
DefaultVelFF	默认 G01 速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的 F 情况下，默认的 F 值
DefaultAccFF	默认 G01 加速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的加速度的情况下，默认的加速度值
DefaultDecFF	默认 G01 减速度	LREAL	正数	20	在 CNC 文件中，没有指定 G01 的减速度的情况下，默认的减速度值
VelocityMode	速度模式	SMC_INT_VELMODE	0，1，3	SIGMOID	速度模式，梯形：0（TRAPEZOID）；S 形：1（SIGMOID）；四次方：3（QUADRATIC）
VelRatio	速度倍率	LREAL	0.01-2	1	速度倍率，最小值 0.01，最大值 2
Jerk	加加速度	LREAL	正数	90000000	加加速度，Pulse/s3，速度模式设置为 3 时需要设置该参数，该参数值不能为 0
LimitVel	最大速度	LREAL	正数	100	各轴允许的最大速度
LimitAccDec	最大加速度	LREAL	正数	1000	各轴允许的最大加减速度
LimitMaxAcc	最大圆弧加速度	LREAL	正数	9000000	过渡圆弧允许的最大加速度
LimitMaxAccJerk	最大圆弧加加速度	LREAL	正数	90000000	过渡圆弧允许的最大加加速度
dOffsetX	X 轴偏移值	LREAL	负数，0 正数	0	X 轴零点坐标偏移值
dOffsetY	Y 轴偏移值	LREAL	负数，0 正数	0	Y 轴零点坐标偏移值
dOffsetZ	Z 轴偏移值	LREAL	负数，0 正数	0	Z 轴零点坐标偏移值
dOffsetU	U 轴偏移值	LREAL	负数，0 正数	0	U 轴零点坐标偏移值
dOffsetV	V 轴偏移值	LREAL	负数，0 正数	0	V 轴零点坐标偏移值
dOffsetW	W 轴偏移值	LREAL	负数，0 正数	0	W 轴零点坐标偏移值
SMC_VarList_0	-	SMC_VarList	-	-	DIN 66025 系统变量
VAR_OUTPUT					
Vel	合速度	LREAL	0，正数	0	当前运动合速度
Done	完成	BOOL	TRUE，FALSE	FALSE	表示插补结束
Busy	运动中	BOOL	TRUE，FALSE	FALSE	插补进行中
CommandAborted	运动中断	BOOL	TRUE，FALSE	FALSE	插补终止
CommandHalt	运动暂停	BOOL	TRUE，FALSE	FALSE	插补暂停
Error	报错	BOOL	TRUE，FALSE	False	插补出错
ErrorID	错误码	DINT	0，正数	0	插补错误码

LineCurrent	当前行号	INT	0，正数	0	当前的执行的行
dwSwitches	开关状态	DWORD	0，正数	0	当前 H 代码对应的开关状态
iLastSwitch	-	INT	-	0	
wM	当前 M 指令值	WORD	-	0	当前 M 指令值
GCodeTextNoArr	暂存译码行号	ARRAY [0..2] OF DINT	-	[0，1，2]	暂存译码行号
GCodeTextArr	暂存译码内容	ARRAY [0..2] OF SMC_GCODE_TEXT	-	-	暂存译码内容
bExportGCodeTextErr	G 文本报错	BOOL		FALSTRUE，FALSEE	输出 G 代码文本出错，不影响系统运行，但会造成输出的近 2
bReadNCFileError	变量名称不存在	BOOL	TRUE，FALSE	FALSE	不能取代 DIN66025 格式 G 代码文件中的变量，SMC_VarList 中变量名称不存在
bNCFileDecodeError	G 文件不存在	BOOL	TRUE，FALSE	FALSE	加载的 G 代码文件不存在

六轴零点坐标偏移 LS_6Axis_ZeroOffset

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_6Axis_ZeroOffset	FB		LS_6Axis_ZeroOffset(pi:= （参数）， dOffsetX:= （参数）， dOffsetY:= （参数）， dOffsetZ:= （参数）， dOffsetU:= （参数）， dOffsetV:= （参数）， dOffsetW:= （参数）， dx=> （参数）， dy=> （参数）， dz=> （参数）， du=> （参数）， dv=> （参数）， dw=> （参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	功能描述
pi	插补输出位置	SMC_POSINFO	-	-	插补模块输出的位置
dOffsetX	X 轴偏移值	LREAL	负数，0 正数	0	X 轴零点坐标偏移值
dOffsetY	Y 轴偏移值	LREAL	负数，0 正数	0	Y 轴零点坐标偏移值
dOffsetZ	Z 轴偏移值	LREAL	负数，0 正数	0	Z 轴零点坐标偏移值
dOffsetU	U 轴偏移值	LREAL	负数，0 正数	0	U 轴零点坐标偏移值
dOffsetV	V 轴偏移值	LREAL	负数，0 正数	0	V 轴零点坐标偏移值
dOffsetW	W 轴偏移值	LREAL	负数，0 正数	0	W 轴零点坐标偏移值
VAR_OUTPUT					
dx	偏移后 X 位置	LREAL	负数，0 正数	0	偏移后 X 轴位置
dy	偏移后 Y 位置	LREAL	负数，0 正数	0	偏移后 Y 轴位置
dz	偏移后 Z 位置	LREAL	负数，0 正数	0	偏移后 Z 轴位置
du	偏移后 U 位置	LREAL	负数，0 正数	0	偏移后 U 轴位置
dv	偏移后 V 位置	LREAL	负数，0 正数	0	偏移后 V 轴位置
dw	偏移后 W 位置	LREAL	负数，0 正数	0	偏移后 W 轴位置

功能说明：

- 这个指令由“PMC_Ipolib”库实现。
- 零点坐标偏移函数相当于 G54。

8.2 插补运动例程

为了便于用户理解和使用雷赛专用插补指令，本节将结合例程为用户介绍插补指令的具体用法。例程主要包括：直线插补运动、圆弧插补运动、椭圆插补运动、螺旋线插补运动、连续插补运动、G 代码插补运动。

8.2.1 直线插补运动例程

不可调速直线插补运动

由于各种直线插补指令类似，下面以**相对坐标模式四轴直线插补运动**为例介绍插补运动指令在程序中的用法。

例程 8.1：编写程序实现 XYZ 轴做空间直线插补运动，合速度为 1000Pulse/s、加减速度为 1000Pulse/s^2；U 轴跟随直线插补运动做点位运动；运动完后改变目标位置再次进行相对坐标四轴直线插补运动。

1) 新建工程，命名 4Axis_Line_Rel，并添加插补库 PMC_IpoLib 和插补辅助库 PMC_BasicModule；

2) 在主程序 PLC_PRG 下添加上电模块 ACT_Power 和运动模块 ACT_Move，并配置对应的参数，如图 8.9、8.10 所示。

3) 如图 8.11 所示，在主程序中分别调用这些模块，根据条件分别设置两次运动各轴的位置。

4) 程序完成后，编译下载到控制器中执行，并将变量 State 强制改为 1，启动插补运动。用 Trace 采集到的程序运行结果如图 8.12 所示。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**插补运动-直线插补运动-4Axis_Line_Rel**”。

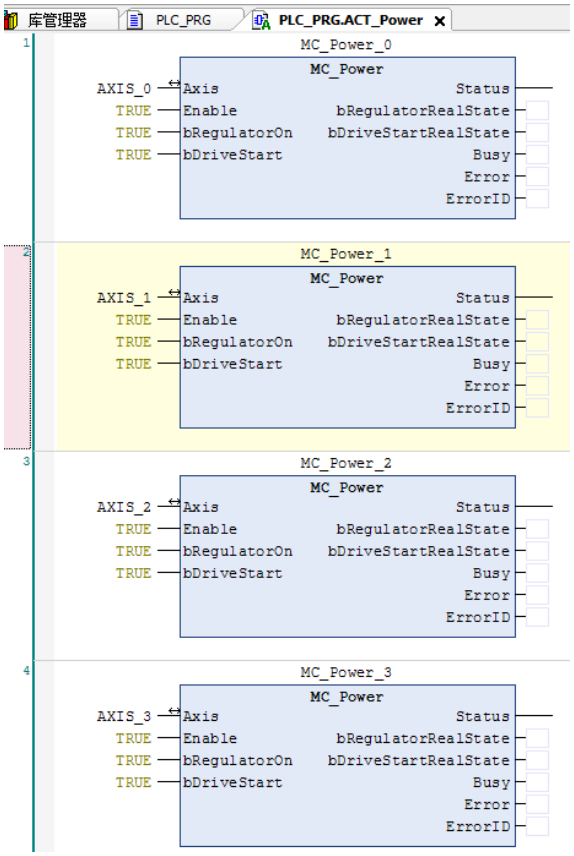


图 8.9　上电模块

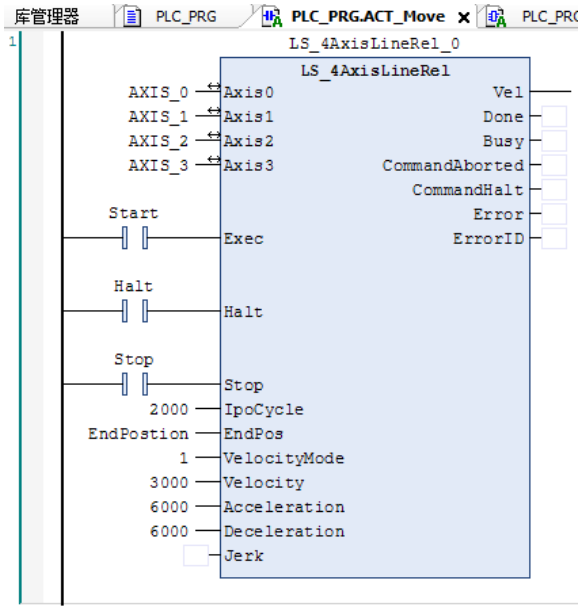


图 8.10　直线插补运动模块

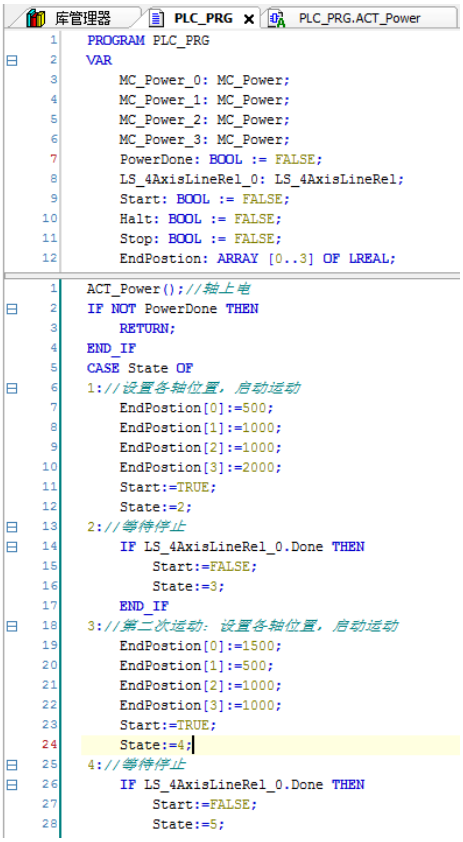


图 8.11　主程序

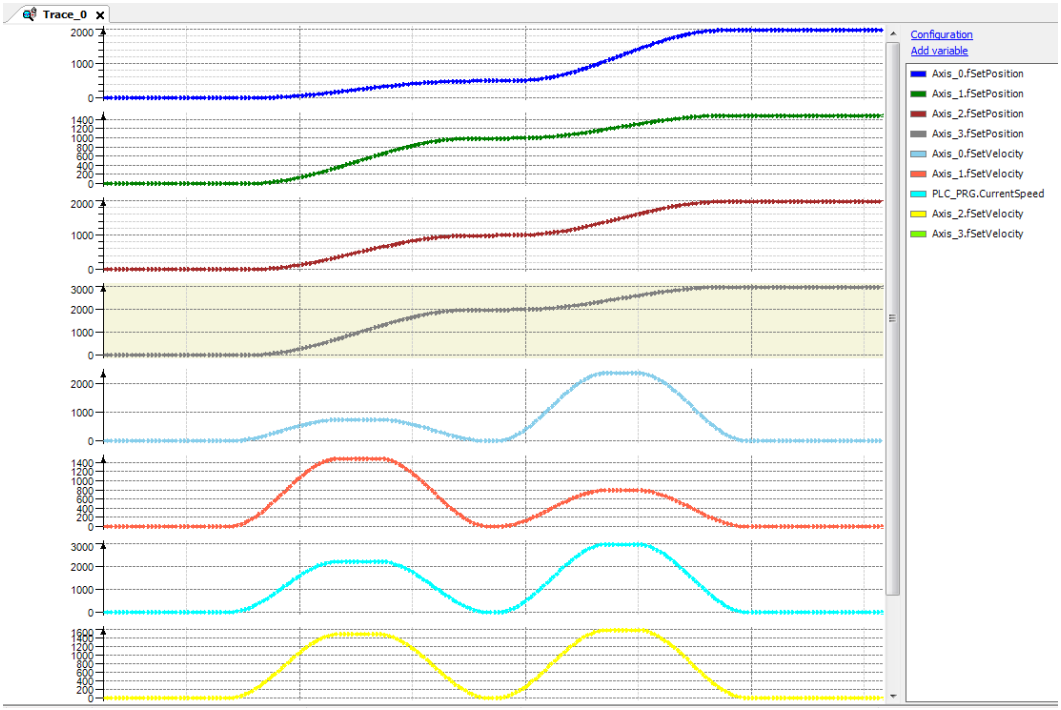


图 8.12　Trace 采集的参数曲线

可调速直线插补运动例程

例程 8.2：编写程序实现 XY 轴以三个不同速度倍率执行直线插补运动，基准速度为 3000Pulse/s，加减速度为 10000Pulse/s^2，三个速度倍率按顺序分别为 0.5、1.5、1，也就是三段速度分别为 1500Pulse/s、4500Pulse/s、3000Pulse/s，并实时监控直线插补运动的合速度。

两轴可调速直线插补指令的用法和一般直线插补用法相同，可在上一个直线插补例程的基础上修改运动模块，将直线插补指令改成直线插补可调速指令，如图 8.13 所示。

在主程序中根据条件改变速度倍率可实现在线改变两轴直线插补的速度，如图 8.14 所示。

程序完成后，编译下载到控制器中执行，并将变量 State 强制改为 1 启动可调速直线插补，用 Trace 采集到的插补运动 XY 轴位置曲线和速度曲线，如图 8.15 所示。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**插补运动-可调速直线插补-2Axis_Line_Rel_ChangeSpeed**”。

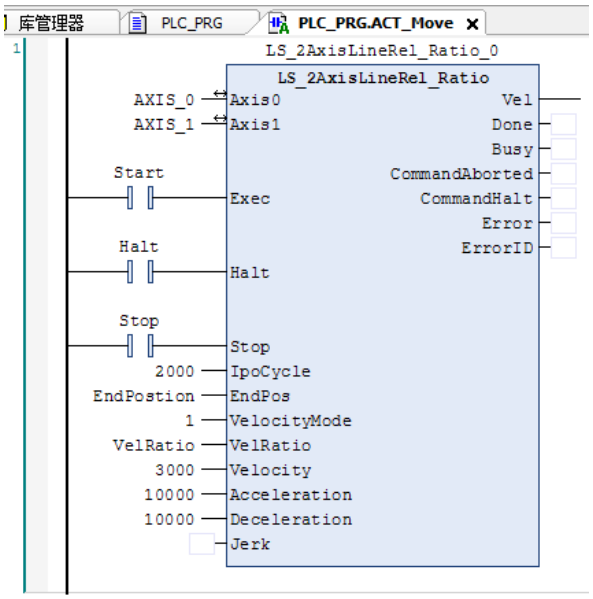


图 8.13　两轴可调速直线插补指令

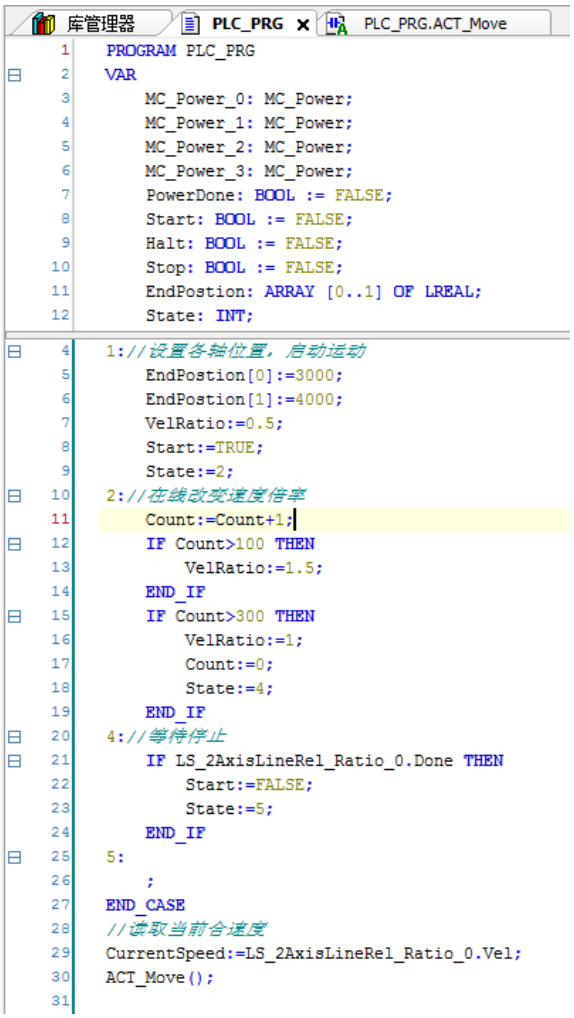


图 8.14　两轴可调速直线插补主程序

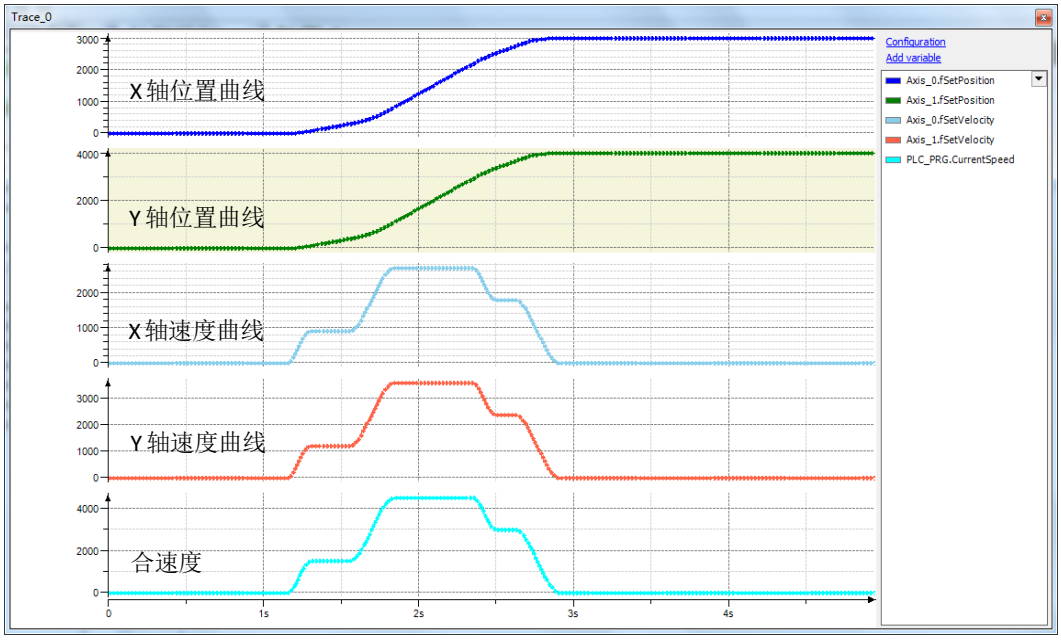


图 8.15　Trace 采集的参数曲线

8.2.2 圆弧插补运动例程

例程 8.3：编写程序实现 XY 轴圆弧插补运动，如图 8.16 所示。采用模式 2-终点半径模式，第一段轨迹顺时针运动到（10000，0），半径 5000，第二段轨迹逆时针运动到（30000，0），半径 10000，单位为 Pulse；运行速度 10000Pulse/s，加减速度 20000Pulse/s^2。

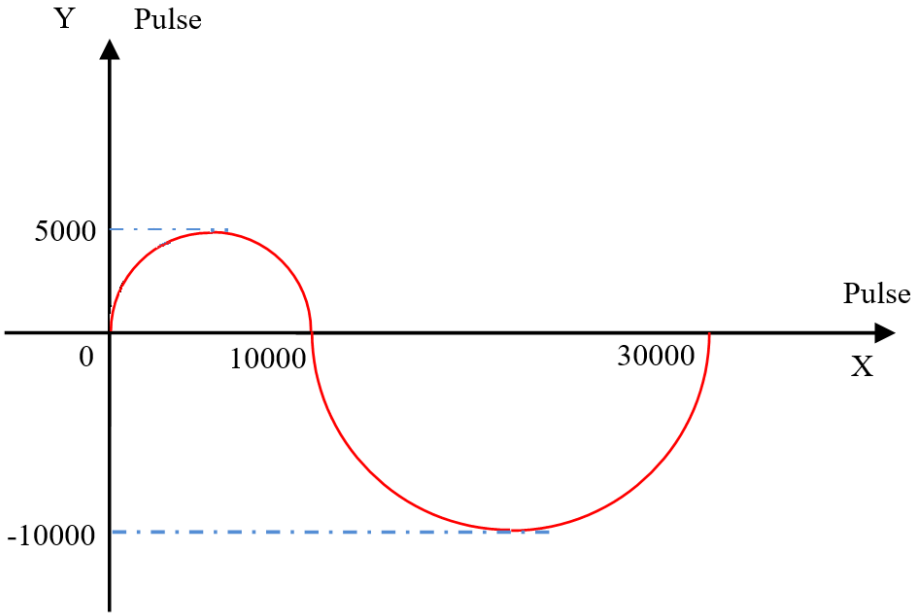


图 8.16　XY 轴的圆弧插补轨迹

1）新建工程，命名 2Axis_Circle_Abs，并添加插补库 PMC_IpoLib、插补辅助库 Module 和 PMC_Controller 库；

2）在主程序 PLC_PRG 下新建上电模块 ACT_Power 和运动模块 ACT_Move，并填写对应的参数和变量，如图 8.17 所示；

3）如图 8.18 所示，在主程序中分别调用这些模块，根据条件分别设置两段圆弧插补的位置参数；

4）程序完成后，编译下载到控制器中执行，并将变量 State 强制改为 1 启动圆弧插补，Trace 采集到的程序运行结果如图 8.19 所示。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**插补运动-圆弧插补-2Axis_Circle_Abs**”。

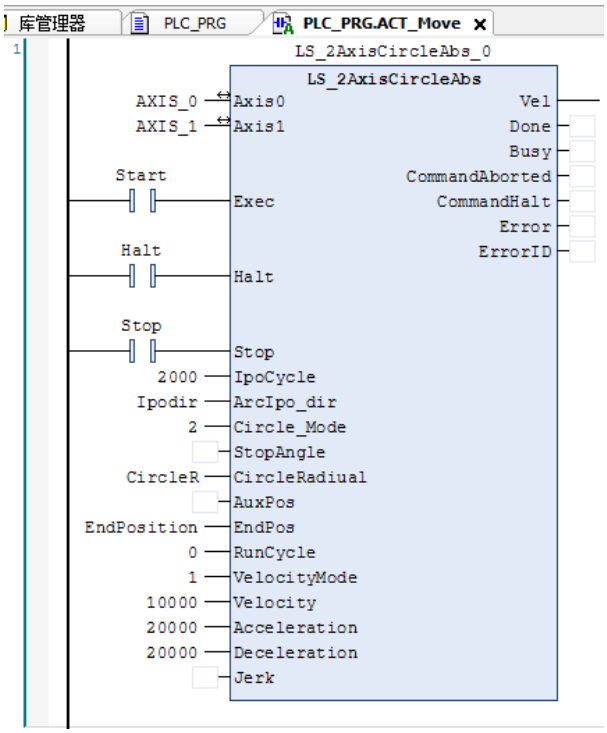


图 8.17　圆弧插补运动模块

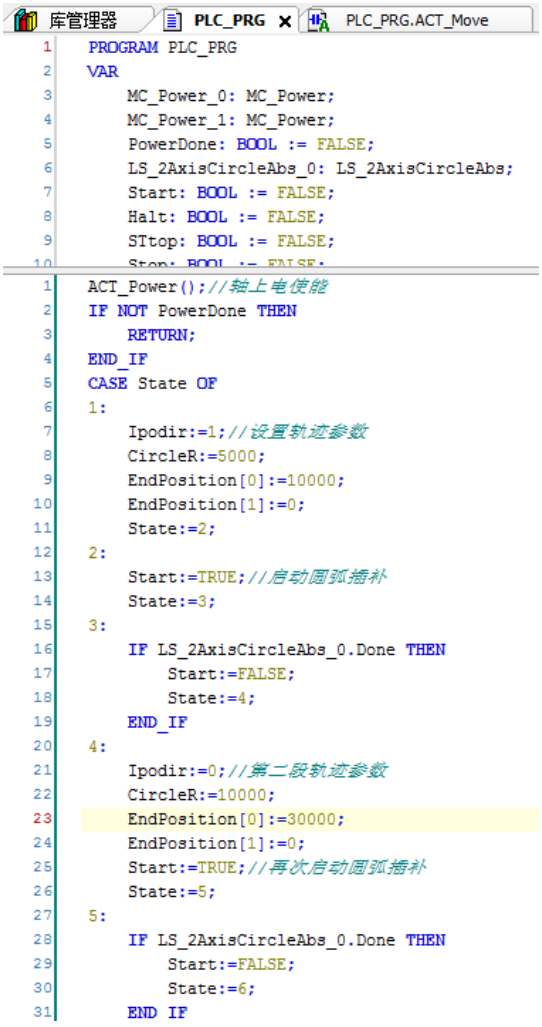


图 8.18　主程序

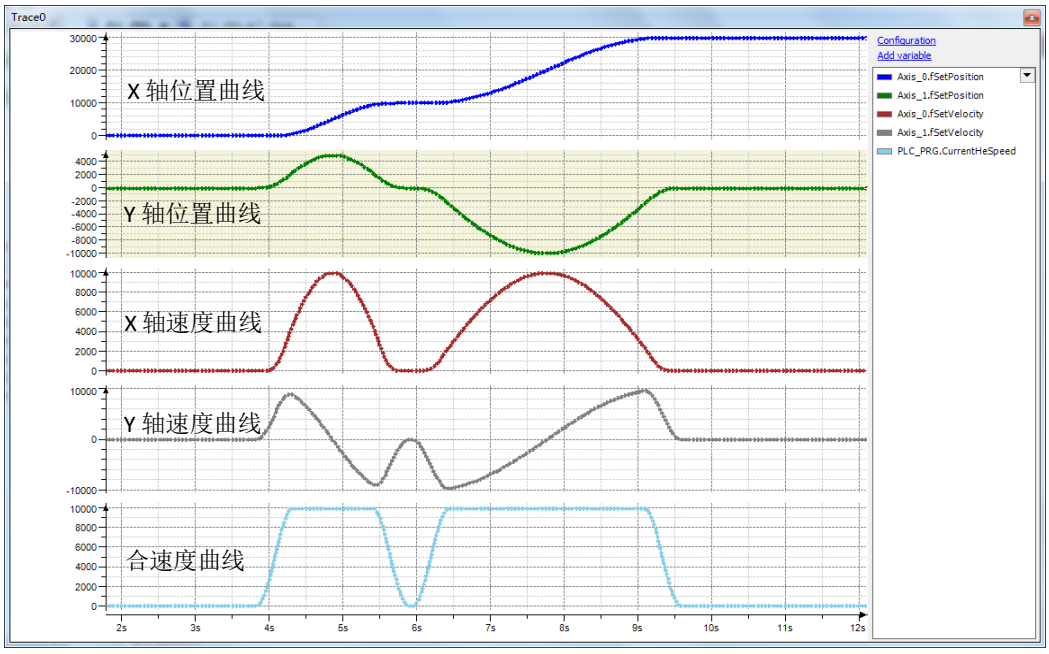


图 8.19　Trace 采集的参数曲线

8.2.3 椭圆插补运动例程

例程 8.4：编写程序实现 XY 轴椭圆插补运动。第一段轨迹顺时针运动到（10000，0），X 轴为长轴，长 10000，Y 为短轴，长 8000，中心点（5000，0）；第二段轨迹逆时针运动到（10000，10000），Y 轴为长轴，长 10000，X 为短轴，长 8000，中心点（10000，5000），单位为 Pulse，运行速度 10000Pulse/s，加减速度 20000Pulse/s^2。

椭圆插补的用法和圆弧插补的用法相同，指令参数有所区别。可在圆弧插补例程的基础上将圆弧插补模块改成椭圆插补模块，修改后运动模块如图 8.20 所示。

如图 8.21 所示，主程序中设置对应的参数，使得程序跑两段半椭圆轨迹。

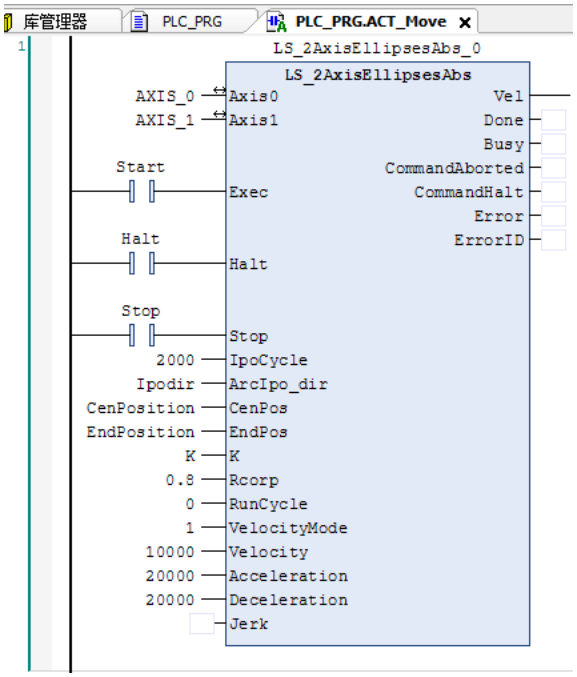


图 8.20　椭圆插补运动模块

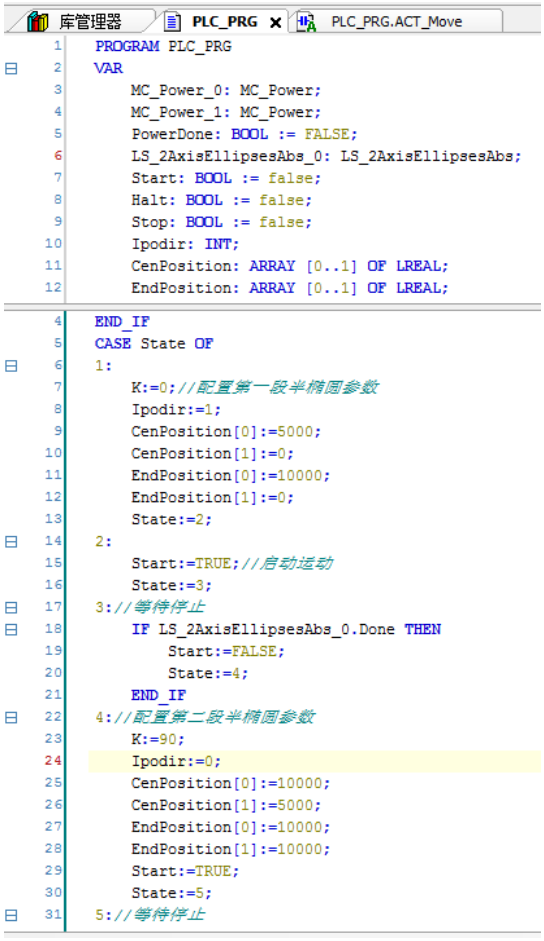


图 8.21　椭圆插补主程序

程序完成后，编译下载到控制器中执行，并将变量 State 强制改为 1 启动椭圆插补，在 trace 中采集到两段椭圆插补运动 XY 轴位置曲线和速度曲线如图 8.22 所示。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**插补运动-椭圆插补-2Axis_Elipses_Abs**”。

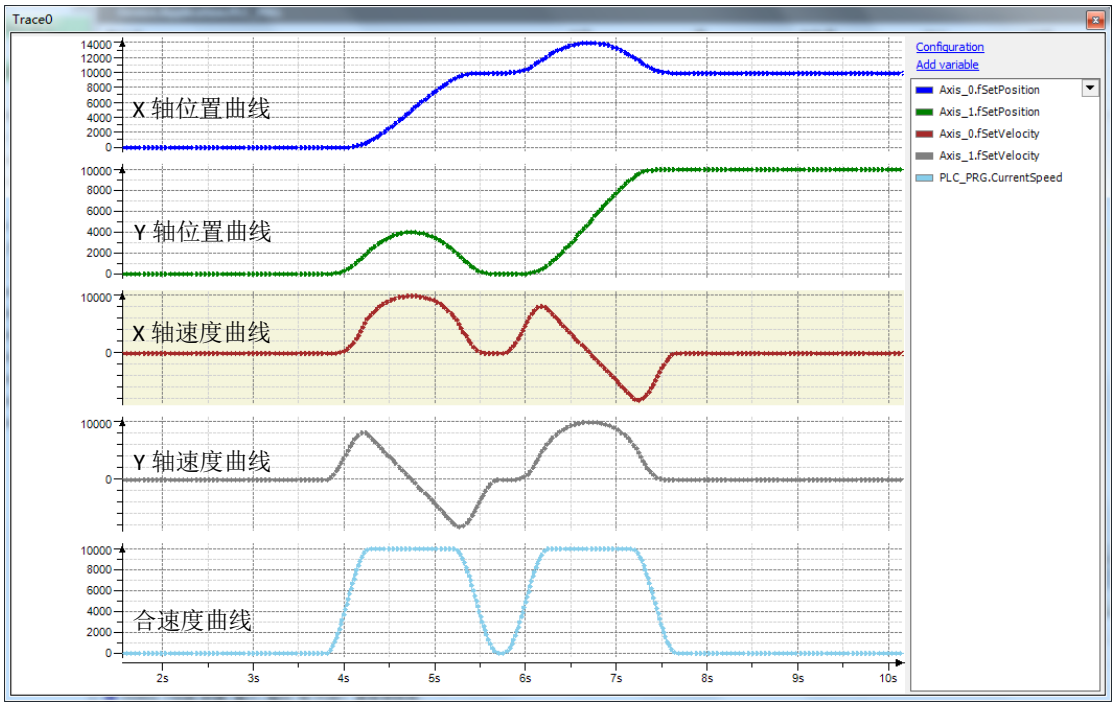


图 8.22　Trace 采集的参数曲线

8.2.4　螺旋线插补运动例程

例程 8.5：编写程序实现 XYZ 轴做三轴圆弧螺旋线插补运动。其中 XY 轴做圆弧插补，采用模式 2 — 终点半径模式，半径 5000，终点（10000，0），Z 轴做跟随运动，距离为 10000，单位为 Pulse，圆弧循环次数为 3 次，设置运行速度 10000Pulse/s，加减速度 20000Pulse/s^2。

三轴圆弧螺旋线插补的用法和圆弧插补的用法相似，指令参数有差别。可在圆弧插补例程的基础上将圆弧插补模块改成三轴圆弧螺旋线插补模块，如图 8.23 所示。

在主程序中设置对应的参数实现 3 圈螺旋线轨迹，如图 8.24 所示。

程序完成后，编译下载到控制器中执行，并将变量 State 强制改为 1 启动圆弧螺旋线插补，在 trace 中采集到的插补运动 XYZ 轴位置曲线和速度曲线如图 8.25 所示。

注意：三轴圆弧螺旋线插补指令中设置的圈数为 3，XY 轴圆弧插补终点为（10000，0），即半圈，故实际上 XY 圆弧最后走了 3.5 圈，如图 8.8 所示。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**插补运动-圆弧螺旋线插补-3Axis_Circle_Helical_Abs**”。

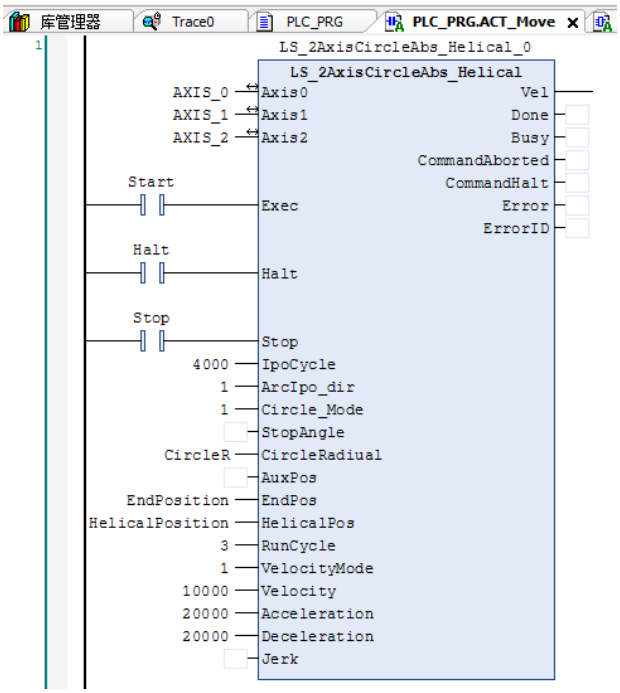


图 8.23　三轴圆弧螺旋线插补模块

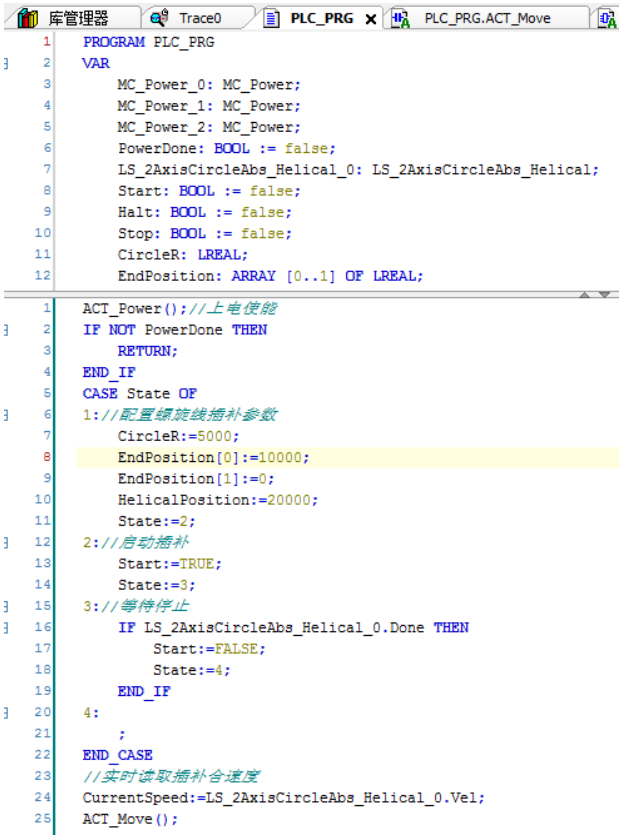


图 8.24　螺旋线插补主程序

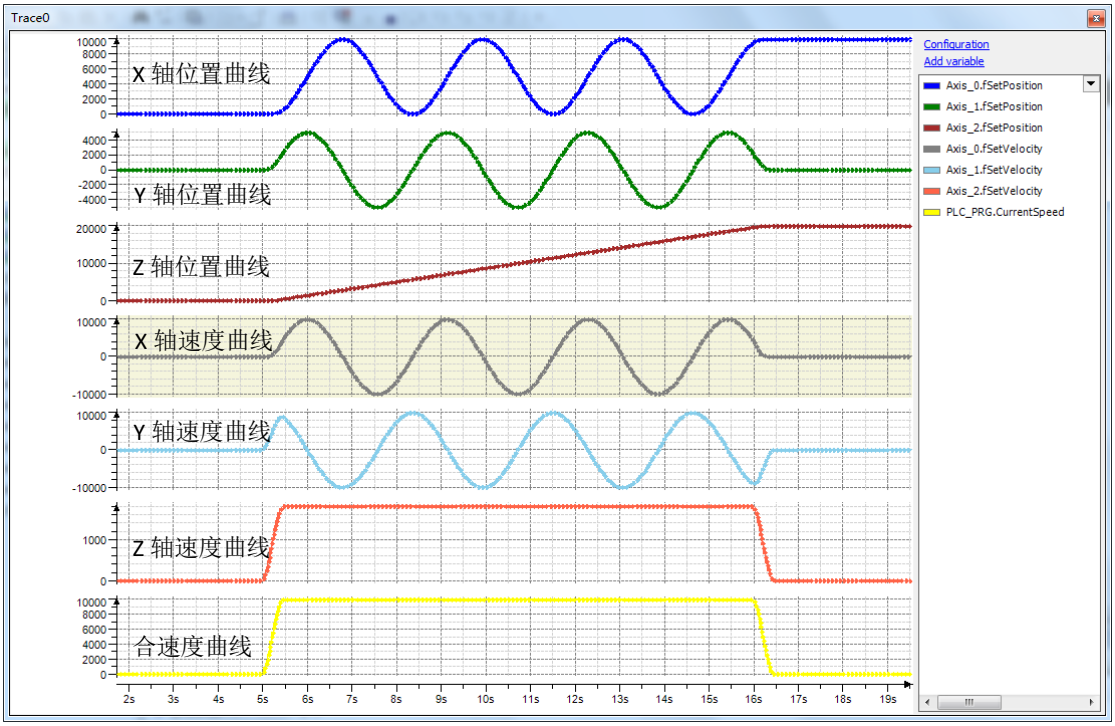


图 8.25　Trace 采集的参数曲线

8.2.5 连续插补运动例程

例程 8.6：编写程序用连续插补指令实现 XY 轴进行三段直线连续插补，速度均为 1000Pulse/s，加减速度为 2000Pulse/s^2，第一段终点坐标为（1000，1000），第二段终点坐标为（3000，2000），第三段终点坐标为（4000，0），如图 8.26 所示。

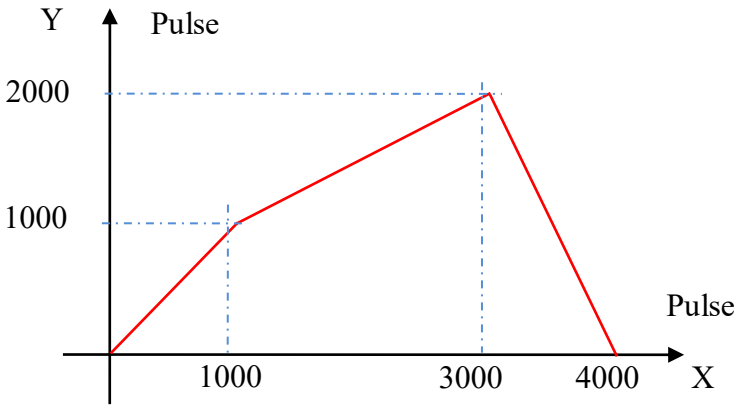


图 8.26　三段直线的连续插补轨迹

连续插补指令的使用方法和前面讲述的几种插补指令都类似，可在直线插补例程的基础上修改，连线插补的运动模块如图 8.27 所示。

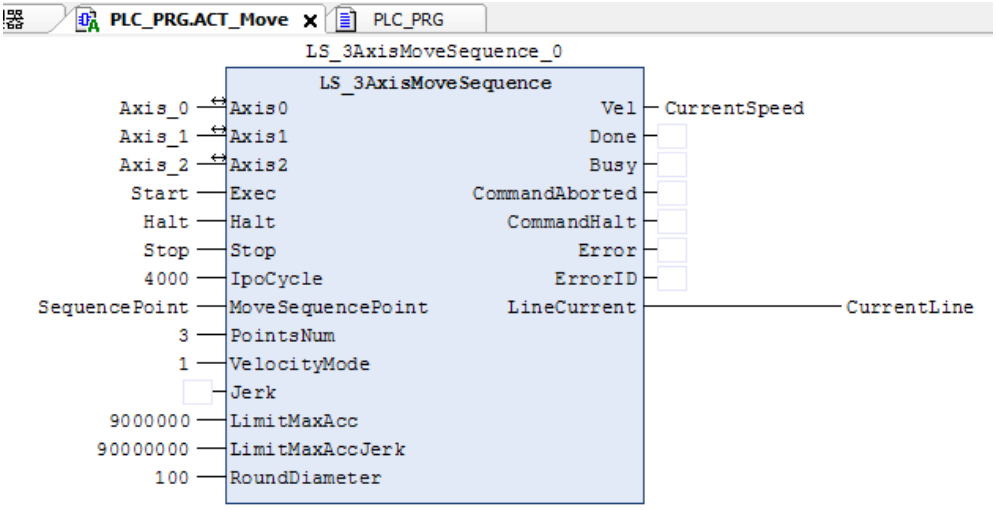


图 8.27　三轴连续插补模块

在主程序中添加数据点的相关参数，如图 8.28 所示。程序完成后，编译下载到控制器中执行，并将变量 State 强制改为 1 启动连续插补，三段连续插补运动 XY 轴位置曲线和速度曲线，如图 8.29 所示。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**插补运动-连续插补-3Axis_Sequence**”。

```
PROGRAM PLC_PRG
VAR
    clearErr: BOOL;
    Stateenable: BOOL := TRUE;
    MC_ReadStatus_0: MC_ReadStatus;
    MC_Power_0: MC_Power;
    MC_Power_1: MC_Power;
    MC_Power_2: MC_Power;
    PowerDone: BOOL := FALSE;
    LS_3AxisMoveSequence_0: LS_3AxisMoveSequence;
    Start: BOOL := FALSE;
    Halt: BOOL := FALSE;
END_IF
CASE State OF
1://配置连续插补数据点参数
    FOR i:=0 TO 2 DO
        SequencePoint[i].Acc:=20000;
        SequencePoint[i].DataType:=1;
        SequencePoint[i].LineNum:=i;
        SequencePoint[i].U_EndPos:=0;
        SequencePoint[i].Vel:=10000;
        SequencePoint[i].Z_EndPos:=0;
    END_FOR
    SequencePoint[0].X_EndPos:=1000;
    SequencePoint[0].Y_EndPos:=1000;
    SequencePoint[1].X_EndPos:=3000;
    SequencePoint[1].Y_EndPos:=2000;
    SequencePoint[2].X_EndPos:=4000;
    SequencePoint[2].Y_EndPos:=0;
    State:=2;
2://启动运动
    Start:=TRUE;
    State:=3;
3://等待运动停止
    IF LS_3AxisMoveSequence_0.Done THEN
        Start:=FALSE;
        State:=4;
    END_IF
```

图 8.28　监控连续插补合速度程序

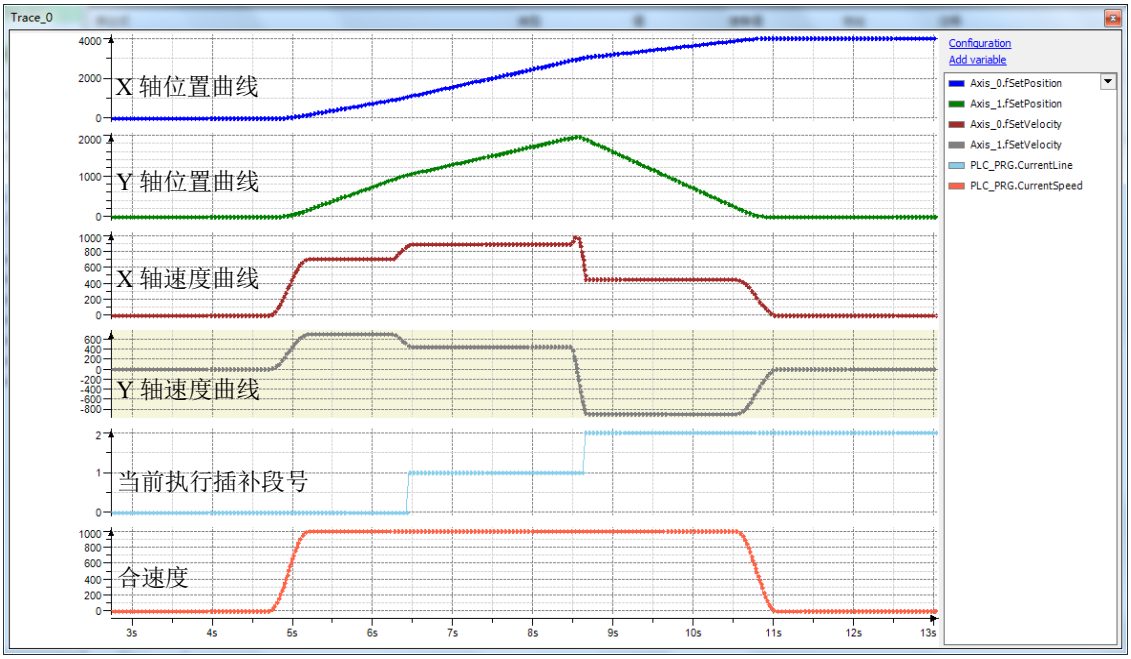


图 8.29　Trace 采集的各参数曲线

8.2.6 采用界面的形式执行 G 代码

例程 8.8：编写程序实现 XYZ 轴 G 代码连续插补运动。轨迹分三段，第一段 XY 轴执行 G01 直线插补，运动到点（1000，1000）；第二段 XY 轴执行 G02 顺时针圆弧插补，运动到点（1000，0）；第三段 XY 轴执行 G03 逆时针圆弧插补，运动到点（0，0）。插补运行速度为 1000Pulse/s，加、减速度为 2000Pulse/s^2。

1) 新建工程，命名 3Axis_G_Code，并添加插补库 PMC_IpoLib、插补辅助库 PMC_BasicModule 和 PMC_Controller 库；

2) 在设备栏“Application”下新建 CNC 程序。方法：右击 Application-添加对象-CNC 程序，并命名为 CNC_1。“执行选项”选择默认参数，“编译模式”选择 SMC_CNC_REF，并在 G 代码程序编辑区编写 G 代码，如图 8.30 所示。

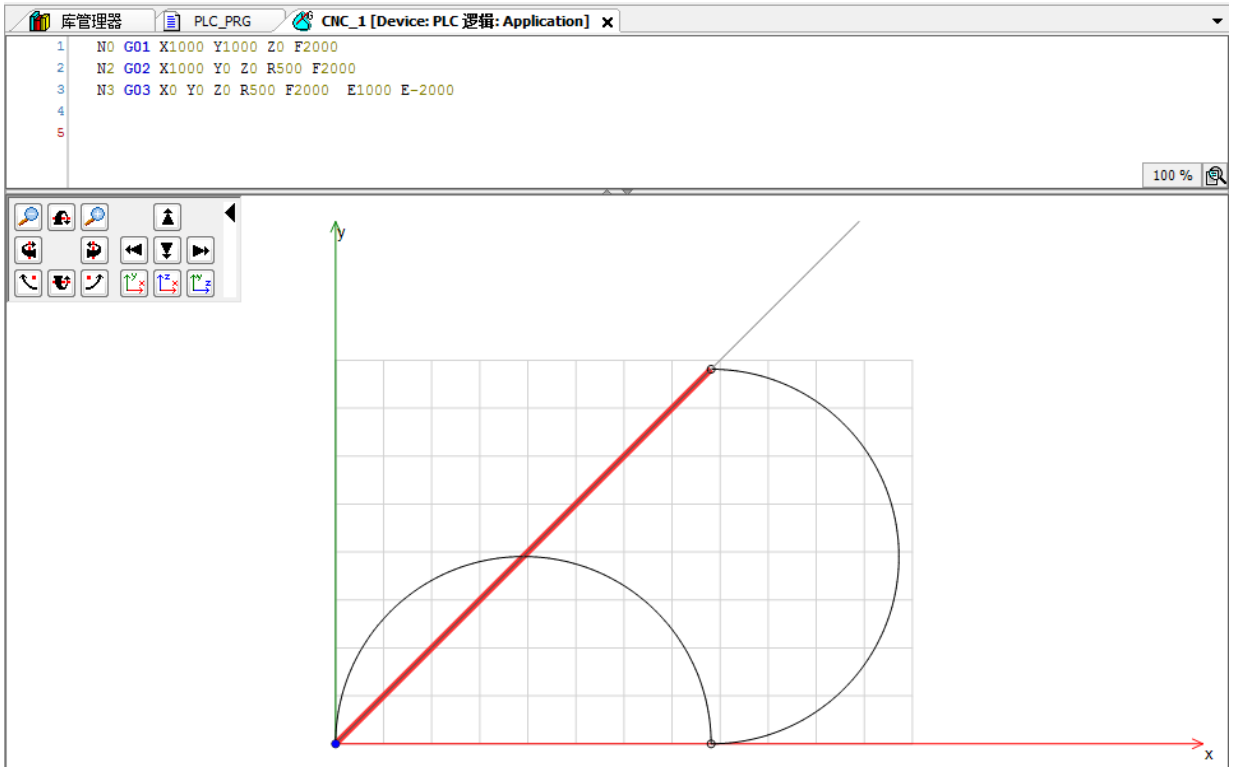


图 8.30 G 代码程序编辑

3) 添加模块：在主程序 PLC_PRG 下新建上电模块 ACT_Power、运动模块 ACT_Move，并填写相应的参数和变量，实时读取插补运动的当前速度和当前执行的 G 代码行号，如图 8.31 所示。

4) 编辑主程序，在主程序中分别调用上述模块，根据条件启动和停止三轴 G 代码连续插补，如图 8.32 所示。

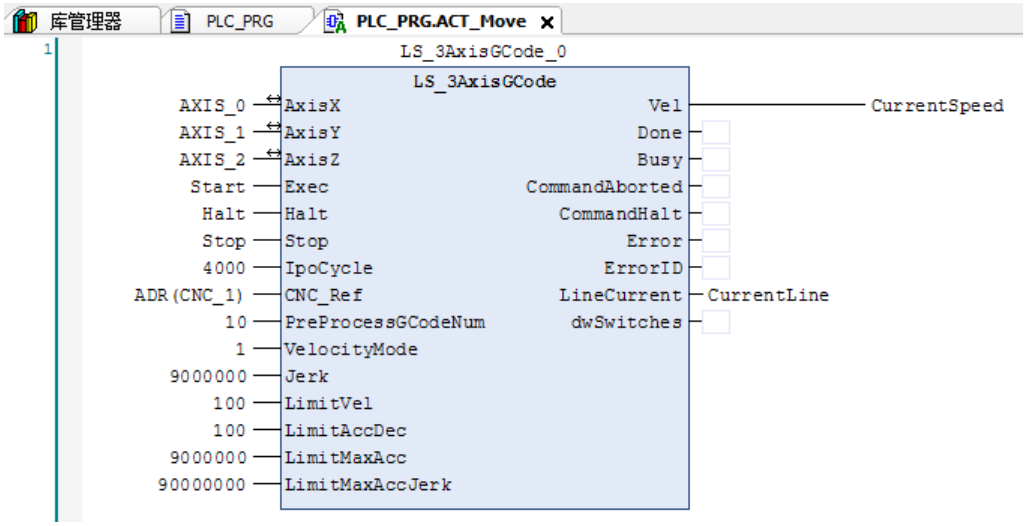


图 8.31　G 代码插补模块

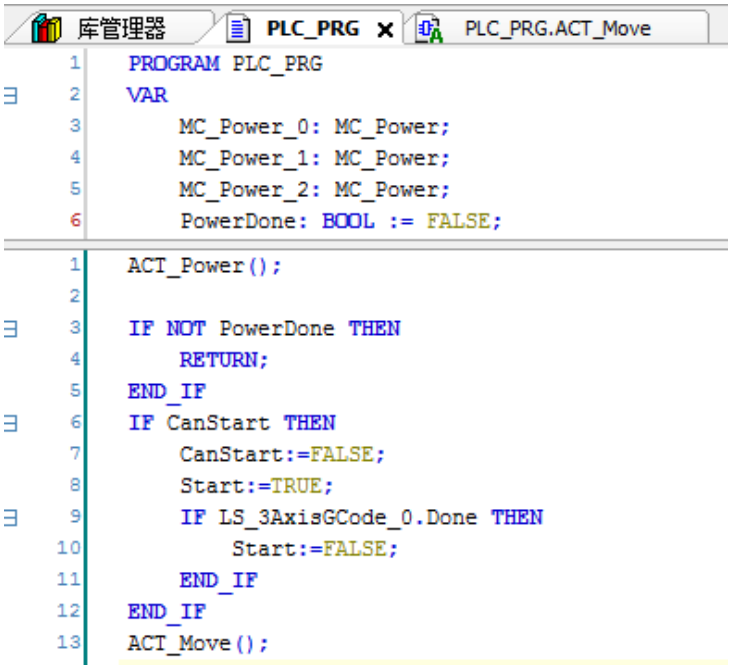


图 8.32　调用 G 代码插补模块程序

5) 程序完成后，编译下载到控制器中执行，并将变量 CanStart 强制改为 TRUE，启动三轴 G 代码连续插补，程序运行结果参见图 8.33 和图 8.34。

需要注意：本例中使用三轴 G 代码连续插补指令 LS_3AxisGCode 中的参数 CNC_Ref 为指向 G 代码程序的指针，程序中新建的 G 代码程序文件名为 CNC_1，直接引用 ADR（CNC_1）即可；如果是 G 代码文件的形式，直接将 G 代码读取出来放在字符串变量中，再调用对应的 G 代码连续插补指令。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**插补运动-G 代码连续插补-3Axis_G_Code**”。

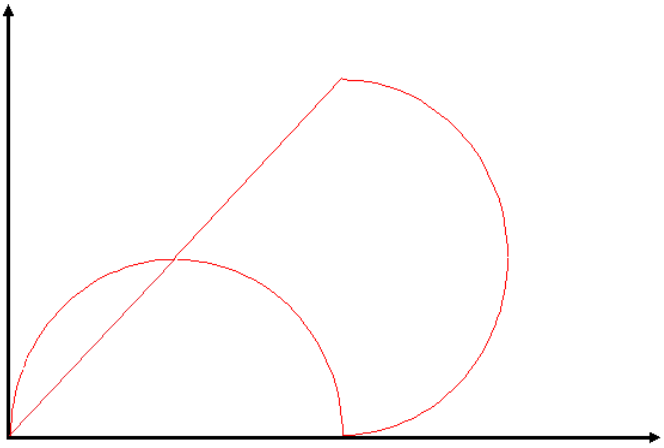


图 8.33　G 代码插实际位置轨迹

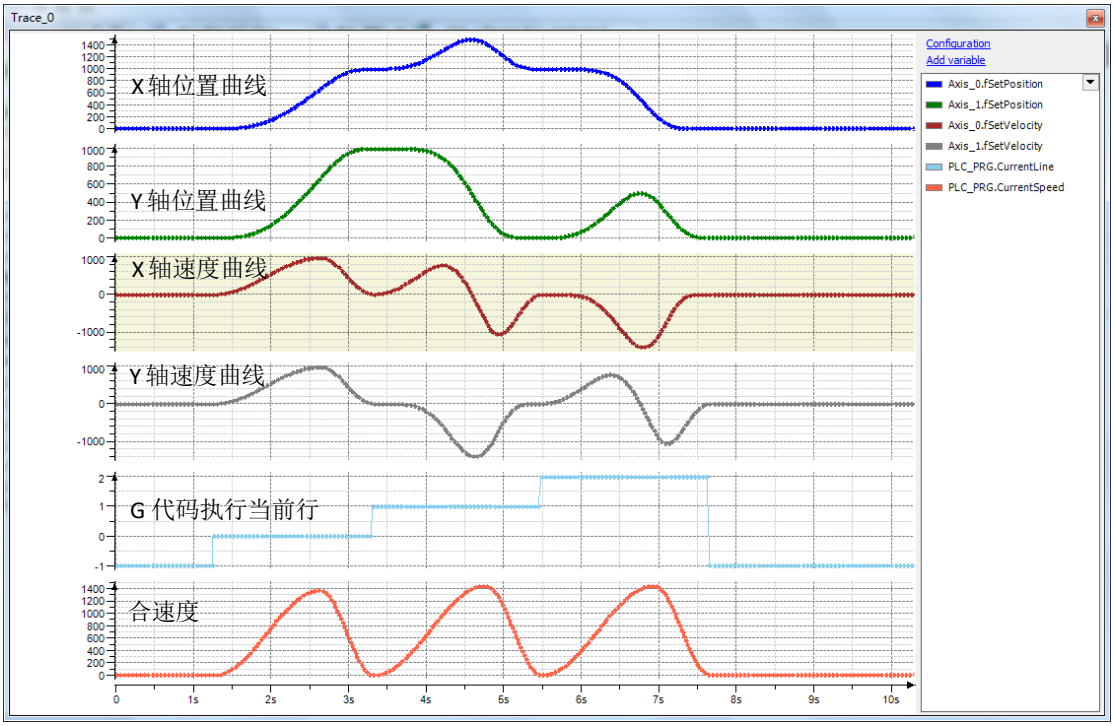


图 8.34　Trace 采集的各参数曲线

8.2.7　采用文件的形式执行 G 代码

例程 8.9：编写辑程序实现 XYZ 轴执行 G 代码。轨迹分二段：第一段 XY 轴执行 G00 快速定位，运动到点（50000，50000），运行速度为 50000Pulse/s，加、减速度为 500000Pulse/s^2。第二段 XY 轴执行 G01 直线插补，运动到点（0，0），运行速度为 30000Pulse/s，加、减速度为 300000Pulse/s^2。

1）新建工程，命名 3Axis_G_File，并添加插补库 PMC_IpoLib、插补辅助库 PMC_BasicModule 和 PMC_Controller 库；

2）下载 G 代码文件到控制器。方法参考《PMC600 中型 PLC 用户手册 3—编程语言篇》；

3）添加模块：在主程序 PLC_PRG 下新建上电模块 ACT_Power、运动模块 ACT_Move，并填写相应的参数和变量，并实时读取插补运动的当前速度和当前执行的 G 代码行号，如图 8.35 所示。其中的 CNC 文件“XYZ.g”文件，通过文件系统传输到控制器的 FLASH 中，G 代码文件的内容如图 8.36 所示。

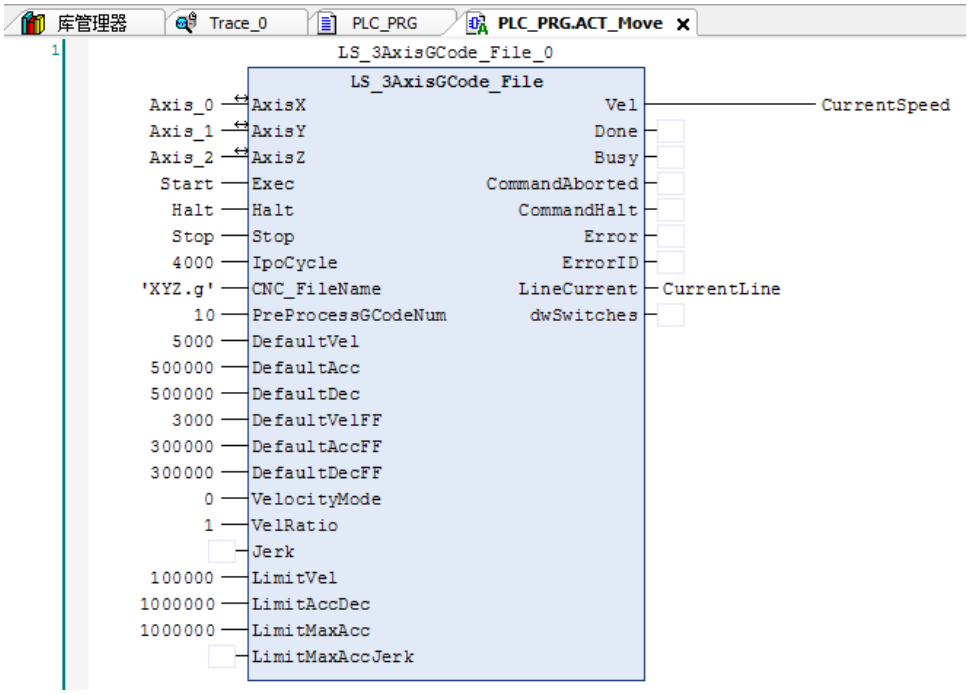


图 8.35　G 代码插补模块

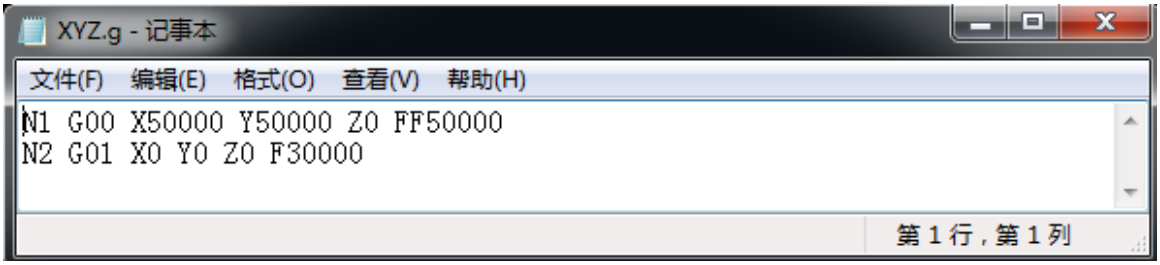


图 8.36　G 代码文件

4）编辑主程序，在主程序中分别调用上述模块，根据条件启动三轴 G 代码文件插补，如图 8.37 所示。

5）程序完成后，编译下载到控制器中执行，并将变量 CanStart 强制改为 TRUE 启动三轴 G 代码连续插补，程序运行结果如图 8.38 所示。

注意：本例中使用三轴 G 代码连续插补指令 LS_3AxisGCode_File，参数中 CNC_FileName 为无路径 G 代码文件名，包含文件类型。G 代码文件格式为 DIN66025 的标准 G 代码格式，必须包含行号 N。G0 速度、加速度和减速度可以用代码 FF、EF，也可

以用模块 VAR_INPUT。G1 速度、加速度和减速度可以用代码 F、E，也可以用模块的 VAR_INPUT。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**插补运动-G 代码连续插补-3Axis_G_File**”。

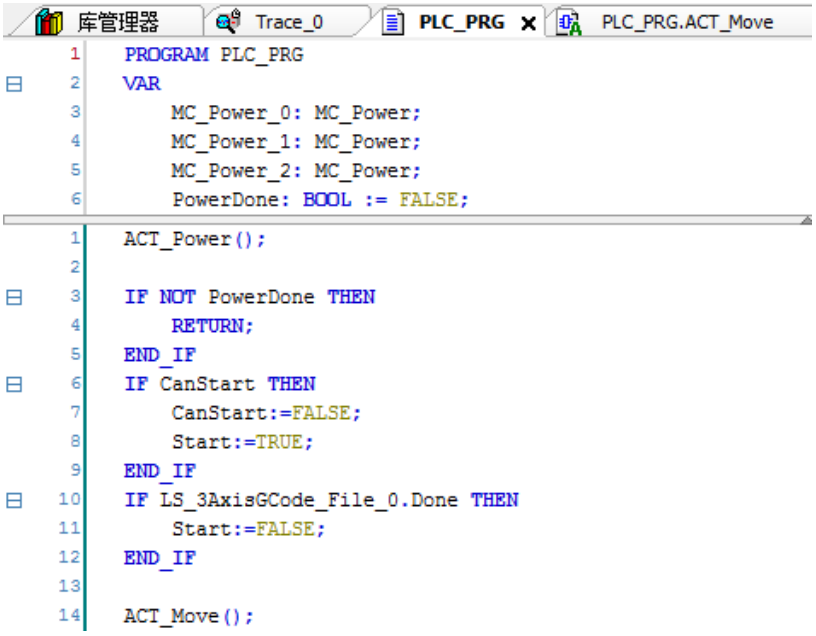


图 8.37　执行 G 代码插补程序

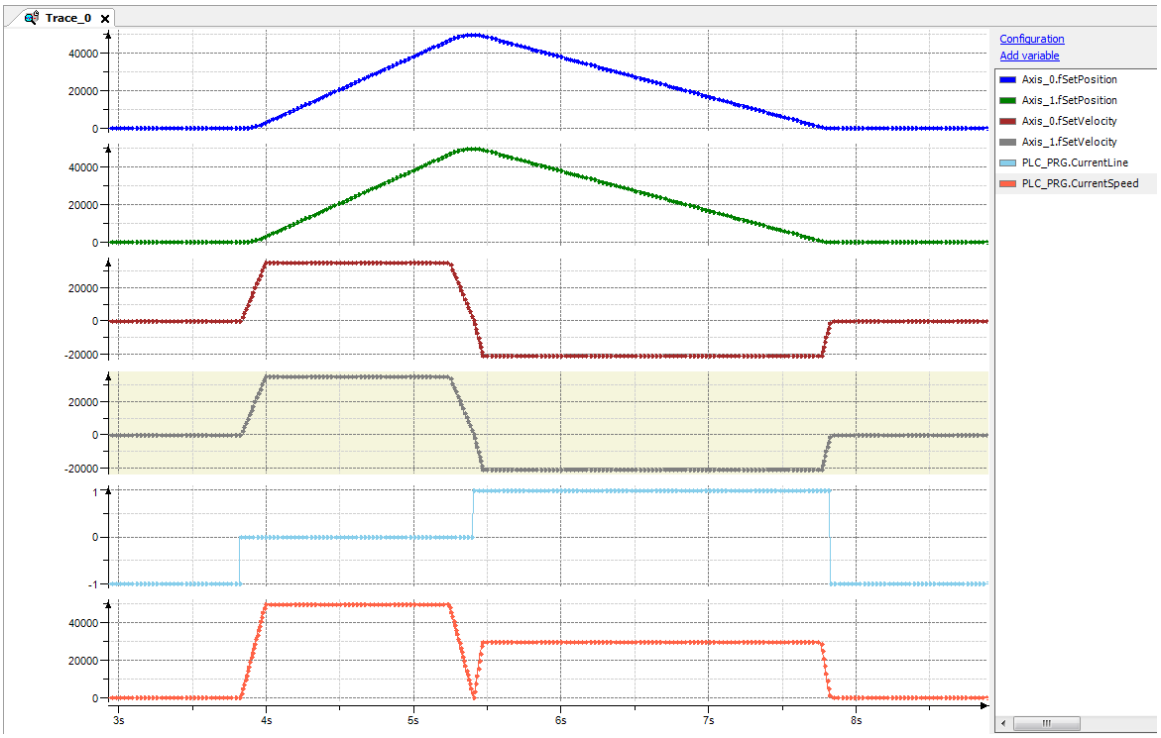


图 8.38　Trace 采集的各参数曲线

第 9 章　雷赛专用特殊 IO 功能

PMC600 运动控制器支持限位、编码器、高速比较、位置锁存和伺服专用功能等特殊 IO 功能。相关指令包含在 PMC_Controller 库中。程序中若想使用这些功能，必须在工程中添加 PMC_Controller 库，库中包含的相关指令如表 9.1 所示。

表 9.1　PMC_Controller 库中包含的指令

功能类型	指令名	功能说明
脉冲轴控制	LS_PulseAxis_MachineState	轴脉冲输出控制模块
	LS_PulseAxis_SetPulseMode	设置轴脉冲模式
	LS_PulseAxis_GetPulseMode	读取轴脉冲模式
	LS_PulseAxis_SetSpecialIOLogicalState	设置轴特殊 IO 信号
	LS_PulseAxis_GetSpecialIOLogicalState	读取轴特殊 IO 信号
	LS_PulseAxis_GetSpecialIOState	获取轴特殊 IO 状态
	LS_PulseAxis_SetRatio	设置轴电子齿轮比
	LS_PulseAxis_ReadRatio	读取轴电子齿轮比
编码器	LS_Encoder_EzClearSet	EZ 清零参数设置
	LS_Encoder_EzClearRead	EZ 清零配置读取
	LS_Encoder_EzSetFilterTime	EZ 滤波参数设置
	LS_Encoder_EzGetFilterTime	EZ 滤波参数读取
	LS_Encoder_SetVal	设置编码器数值
	LS_Encoder_GetVal	读取编码器数值
	LS_Encoder_SetAxisMove	编码器数值转换轴运动
	LS_Encoder_SetWorkMode	设置编码器工作模式
高速比较	LS_HighSpeedCmp_ClearPara	清除比较器
	LS_HighSpeedCmp_SetLinearPara	设置线性模式参数
	LS_HighSpeedCmp_GetLinearPara	读取线性模式参数
	LS_HighSpeedCmp_SetWorkPara	设置比较器参数
	LS_HighSpeedCmp_GetWorkPara	读取比较器参数
	LS_HighSpeedCmp_SetCmpPos	设置比较位置
	LS_HighSpeedCmp_GetCurState	获取比较器当前状态
	LS_HighSpeedCmp_SetFIFOPos	设置 FIFO 模式比较位置
	LS_HighSpeedCmp_SetFIFOPosType2	设置 FIFO 模式 2 比较位置
锁存功能	LS_ZeroLatch_SetPara	设置原点锁存参数
	LS_ZeroLatch_GetState	读取原点锁存状态
	EzLatch_SetPara	设置 EZ 锁存参数
	EzLatch_GetState	读取 EZ 锁存状态
	LS_HighSpeedLatch_Clear	清除高速比较锁存
	LS_HighSpeedLatch_SetSource	设置高速锁存数据源
	LS_HighSpeedLatch_SetWorkMode	设置高速锁存模式

	LS_HighSpeedLatch_ReadState	读取高速锁存状态
	LS_HighSpeedLatch_ReadVal	读取当前锁存值
	LS_HighSpeedLatch_FIFOReadVal	读取 FIFO 模式锁存值
	LS_HighSpeedLatch_FIFOReadNumber	读取 FIFO 模式锁存个数
PWM 输出	LS_PWM_SetVal	设置 PWM 的参数
	LS_PWM_GetVal	获取 PWM 数值

9.1 脉冲轴控制

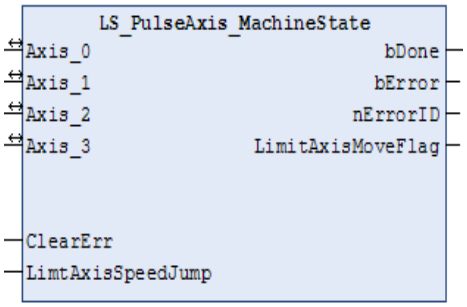
表 9.2　脉冲轴控制指令

指令名	功能说明
LS_PulseAxis_MachineState	轴脉冲输出控制模块
LS_PulseAxis_SetPulseMode	设置轴脉冲模式
LS_PulseAxis_GetPulseMode	读取轴脉冲模式
LS_PulseAxis_SetSpecialIOLogicalState	设置轴特殊 IO 信号
LS_PulseAxis_GetSpecialIOLogicalState	读取轴特殊 IO 信号
LS_PulseAxis_GetSpecialIOState	获取轴特殊 IO 状态
LS_PulseAxis_SetRatio	设置轴电子齿轮比
LS_PulseAxis_ReadRatio	读取轴电子齿轮比

轴脉冲输出控制模块　LS_PulseAxis_MachineState

控制 PMC600 主机脉冲轴的脉冲输出。在使用脉冲轴前必须调用，否则没有脉冲输出。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_PulseAxis_MachineState	FUN		LS_PulseAxis_MachineState(Axis_0: = （参数）， Axis_1: = （参数）， Axis_2: = （参数）， Axis_3: = （参数）， ClearErr: = （参数）， LimtAxisSpeedJump: = （参数）， bDone=>（参数）， bError=>（参数）， nErrorID=>（参数）， LimitAxisMoveFlag=>（参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis_0	轴 0	AXIS_REF_SM3	—	—	指定轴
Axis_1	轴 1	AXIS_REF_SM3	—	—	指定轴
Axis_2	轴 2	AXIS_REF_SM3	—	—	指定轴
Axis_3	轴 3	AXIS_REF_SM3	—	—	指定轴
VAR_INPUT					
ClearErr	清除报错	BOOL	TRUE FALSE	FALSE	清除错误，上升沿有效。(该参数仅用于清除模块内的错误标记和错误码)
LimtAxisSpeedJump	速度跳变限制	LREAL	正数	50000	限制轴实际输出时单周期速度跳变，当速度跳变大于该值时，模块报错并限制轴脉冲输出
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行
bError	执行中	BOOL	TRUE FALSE	FALSE	错误状态。False：没有错误；True：出现错误或其他意外情况
nErrorID	错误	DINT	0，正数	0	错误码
LimitAxisMoveFlag	限制运动标记	BOOL	TRUE FALSE	FALSE	限制轴运动标记

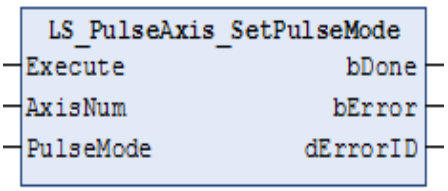
说明：

- 这个指令由“PMC_Controller”库实现。
- 注意在使用该模块的时候，需要将模块放置到主任务之中。

设置轴脉冲模式 LS_PulseAxis_SetPulseMode

设置脉冲轴的脉冲输出模式。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_PulseAxis_SetPulseMode	FUN		LS_PulseAxis_SetPulseMode(Execute：= （参数）， AxisNum：= （参数）， PulseMode：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行
AxisNum	轴号	UINT	0-3	0	轴号：0—3
PulseMode	脉冲输出模式	UINT	0-6	0	设置轴的脉冲输出模式。0-3：脉冲方向模式；4-5：双脉冲模式；6：AB 相模式
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行
bError	报错	BOOL	TRUE FALSE	FALSE	FALSE-没有错误；True-输入参数越界，或者模式不正确，或者固件版本太低不支持该功能块
dErrorID	错误码	DINT	0，正数	0	错误码

说明：

- 这个指令由“PMC_Controller”库实现。
- 脉冲输出模式总共有 6 种：0~3：脉冲方向模式；4~5：双脉冲模式；6：AB 相模式。详见表 9.3。

表 9.3　脉冲输出模式

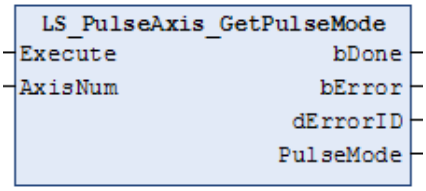
脉冲输出模式	正方向脉冲		负方向脉冲	
	PULSE 输出	DIR 输出端	PULSE 输出	DIR 输出端
0		高电平		低电平
1		高电平		低电平
2		低电平		高电平
3		低电平		高电平
4		高电平	高电平	
5		低电平	低电平	
6	A 相  B 相 		B 相  A 相 	

注意：脉冲模式默认是模式 0，如果实际应用不是使用这种模式，则在调用运动函数输出脉冲之前，一定要根据驱动器接收脉冲的模式，调用脉冲模式设置函数设置控制器脉冲输出模式。

读取轴脉冲模式 LS_PulseAxis_GetPulseMode

获取脉冲轴的脉冲输出模式。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_PulseAxis_GetPulseMode	FUN		LS_PulseAxis_GetPulseMode(Execute:= （参数）, AxisNum:= （参数）, bDone=>（参数） , bError=>（参数） , dErrorID=>（参数） PulseMode=>（参数） ,);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行
AxisNum	轴号	UINT	0-3	0	轴号：0—3
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误； True-输入参数越界，或者模式不正确，或者固件版本太低不支持该功能块
dErrorID	错误码	DINT	0，正数	0	错误码
PulseMode	脉冲输出模式	UINT	0-6	0	轴的脉冲输出模式。0-3：脉冲方向模式；4-5：双脉冲模式；6：AB 相模式

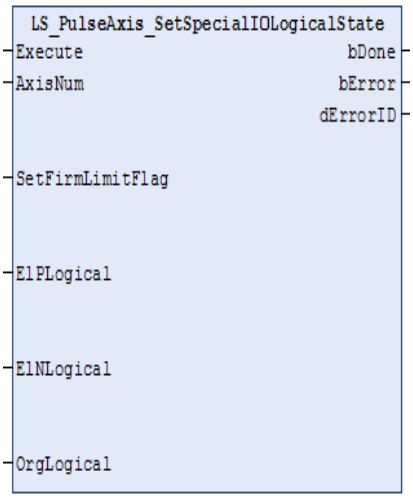
说明：

这个指令由“PMC_Controller”库实现。与指令“LS_PulseAxis_SetPulseMode”配合使用。

设置轴特殊 IO 信号 LS_PulseAxis_SetSpecialIOLogicalState

设置脉冲轴的特殊 IO 信号的启用与逻辑电平。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_PulseAxis_SetSpecialIOLogicalState	FUN		LS_PulseAxis_SetSpecialIOLogicalState(Execute:= （参数）， AxisNum:= （参数）， SetFirmLimitFlag:=（参数）， ElPLogical:= （参数）， ElNLogical:= （参数）， OrgLogical:= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
AxisNum	轴号	UINT	0-3	0	轴号：0—3
SetFirmLimitFlag	硬件限位有效标记	BOOL	TRUE FALSE	FALSE	设置硬件限位是否有效，false 表示不处理硬件限位，true 表示处理硬件限位
ElPLogical	正限位逻辑电平	BOOL	TRUE FALSE	FALSE	设置正限位端口逻辑电平
ElNLogical	负限位逻辑电平	BOOL	TRUE FALSE	FALSE	设置负限位端口逻辑电平
OrgLogical	原点逻辑电平	BOOL	TRUE FALSE	FALSE	设置原点端口逻辑电平
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行
bError	错误码	BOOL	TRUE FALSE	FALSE	FALSE-没有错误；True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码

说明：

- 这个指令由“PMC_Controller”库实现。
- PMC600 为用户提供了设置硬件限位功能，用户必须根据设备的限位开关硬件接线，控制器内部限位开关的有效电平为低电平。
- 通过指令去设置每个轴的特殊 IO 信号（正限位，负限位，原点）的内部读取状态。例如，设置 SetElP=0，表示正限位端口在高电平的状态下内部读取的状态值为 FALSE，正限位端口在低电平状态下，读取的状态值为 TRUE；设置 SetElP 为非 0，表示正限位端口在高电平的状态下，内部读取的状态值为 TRUE。

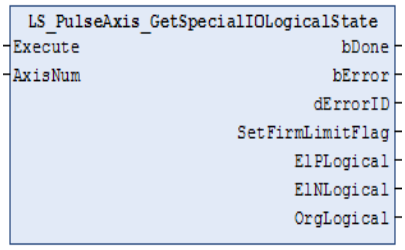
• 需要注意的是：指令只能设置读取物理电平对应的状态，并不影响物理电平本身。如果设置 SetELP=0，表示正限位在高电平的状态下内部读取的状态值为 FALSE，正限位在低电平状态下，读取的状态值为 TRUE；设置 SetELP 为非 0，表示正限位在高电平的状态下，内部读取的状态值为 TRUE，正限位在低电平状态下，读取的状态值为 FALSE。

• 当设置某个轴的硬件限位有效时，正限位只对正向运动起作用，也就是说轴在正向运动时负限位不起作用，同理负向运动正限位也不会起作用。

读取轴特殊 IO 信号 LS_PulseAxis_GetSpecialIOLogicalState

读取每个轴的特殊 IO 信号。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_PulseAxis_GetSpecialIOLogicalState	FUN		LS_PulseAxis_GetSpecialIOLogicalState(Execute:= （参数）， AxisNum:= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）， SetFirmLimitFlag=>（参数）， ElPLogical=>（参数）， ElNLogical=>（参数）， OrgLogical=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
AxisNum	轴号	UINT	0-3	0	轴号：0—3
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误；True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码
SetFirmLimitFlag	硬件限位有效标记	BOOL	TRUE FALSE	FALSE	系统设置的硬件限位是否有效，false 表示不处理硬件限位，true 表示处理硬件限位
ElPLogical	正限位逻辑电平	BOOL	TRUE FALSE	FALSE	系统设置的正限位端口逻辑电平
ElNLogical	负限位逻辑电平	BOOL	TRUE FALSE	FALSE	系统设置的负限位端口逻辑电平
OrgLogical	原点逻辑电平	BOOL	TRUE FALSE	FALSE	系统设置的原点端口逻辑电平

说明：

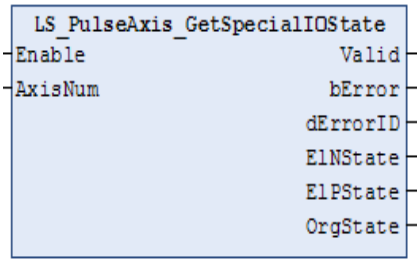
- 这个指令由“PMC_Controller”库实现。
- 读取每个轴的限位是否有效，以及特殊 IO 信号（正限位，负限位，原点）的状态。

例如，设置 SetElP=0，表示正限位端口在高电平的状态下内部读取的状态值为 FALSE，正限位端口在低电平状态下，读取的状态值为 TRUE；设置 SetElP 为非 0，表示正限位端口在高电平的状态下，内部读取的状态值为 TRUE。

获取轴特殊 IO 状态 LS_PulseAxis_GetSpecialIOState

获取脉冲轴的硬件限位、原点信号的状态。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_PulseAxis_GetSpecialIOState	FUN		LS_PulseAxis_GetSpecialIOState(Enable:= （参数）， AxisNum:= （参数）， Valid=>（参数） ， bError=>（参数） ， dErrorID=>（参数） ElNState=>（参数） ， ElPState=>（参数） ， OrgState=>（参数） ，);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Enable	执行指令	BOOL	TRUE FALSE	FALSE	TRUE 表示读取使能
AxisNum	轴号	UINT	0-3	0	轴号：0—3
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中；FALSE：指令未执行
bError	错误代码	BOOL	TRUE FALSE	FALSE	FALSE-没有错误；True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码
ElNState	负限位信号	BOOL	TRUE FALSE	FALSE	负限位有效状态，false 表示无效，true 表示有效
ElPState	正限位信号	BOOL	TRUE FALSE	FALSE	正限位有效状态，false 表示无效，true 表示有效
OrgState	原点信号	BOOL	TRUE FALSE	FALSE	原点有效状态，false 表示无效，true 表示有效

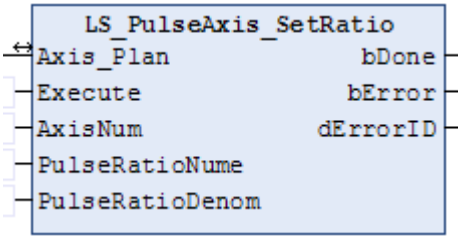
说明：

- 这个指令由“PMC_Controller”库实现。
- 获取脉冲轴的硬件限位、原点信号的状态值。

设置轴电子齿轮比 LS_PulseAxis_SetRatio

设置脉冲轴的脉冲当量。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_PulseAxis_SetRatio	FUN		LS_PulseAxis_SetRatio(Axis_Plan:= （参数）, Execute:= （参数）, AxisNum:= （参数）, PulseRatioNume:= （参数）, PulseRatioDenom:= （参数）, bDone=>（参数）, bError=>（参数）, dErrorID=>（参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
Axis_Plan	脉冲规划轴	AXIS_REF_VIRTUAL_SM3	—	—	脉冲规划轴
VAR_INPUT					
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
AxisNum	轴号	UINT	0-3	0	设置规划轴号：0—3
PulseRatioNume	轴传动比分子	UDINT	正数	1	设置轴传动比分子
PulseRatioDenom	轴传动比分母	UDINT	正数	1	设置轴传动比分母
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE:表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误； TRUE-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码

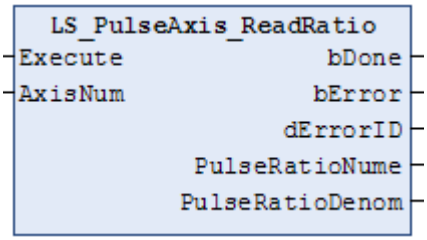
说明：

- 这个指令由“PMC_Controller”库实现。
- 用于设置脉冲轴的脉冲当量，与总线轴的传动比类似，即：
  伺服轴传动比=传动比的分子/传动比的分母
  如轴 0 的传动比可表示为：Axis_Ratio_Nume[0]/Axis_Ratio_Denom[0]
- 所有轴的轴传动比默认值是 1。

读取轴电子齿轮比　LS_PulseAxis_ReadRatio

获取脉冲轴的脉冲当量值。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_PulseAxis_ReadRatio	FUN		LS_PulseAxis_ReadRatio(Execute：= （参数）， AxisNum：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数） PulseRatioNume=>（参数）， PulseRatioDenom=>（参数），);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
AxisNum	轴号	UINT	0-3	0	轴号：0—3
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误； True-输入参数越界，或者模式不正确，或者固件版本太低不支持该功能块
dErrorID	错误码	DINT	0，正数	0	错误码
PulseRatioNume	轴传动比分子	UDINT	正数	1	轴传动比分子
PulseRatioDenom	轴传动比分母	UDINT	正数	1	轴传动比分母

9.2 编码器应用

表 9.4　编码器相关指令表

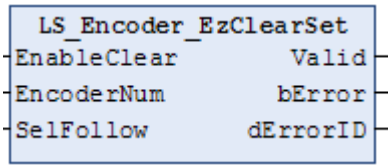
指令名	功能说明
LS_Encoder_EzClearSet	EZ 清零参数设置
LS_Encoder_EzClearRead	EZ 清零配置读取
LS_Encoder_EzSetFilterTime	EZ 滤波参数设置
LS_Encoder_EzGetFilterTime	EZ 滤波参数读取
LS_Encoder_SetVal	设置编码器数值
LS_Encoder_GetVal	读取编码器数值
LS_Encoder_SetWorkMode	设置编码器工作模式
LS_Encoder_SetAxisMove	编码器数值转换轴运动

9.2.1　编码器指令

EZ 清零参数设置　LS_Encoder_EzClearSet

配置编码器值遇 Z 信号清零功能的参数。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_Encoder_EzClearSet	FUN		LS_Encoder_EzClearSet(EnableClear：= （参数）， EncoderNum：= （参数）， SelFollow：= （参数）， Valid=>（参数）， bError=>（参数）， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
EnableClear	启动清零	BOOL	TRUE FALSE	FALSE	启动清零功能：True-清零使能（启用）；FALSE-清零关闭。该信号需要保持。
EncoderNum	编码器号	UINT	0-2	0	编码器号：0—2
SelFollow	触发沿	UINT	0，1	0	选择清零触发沿：0-下降沿清零；1-上升沿清零。编码器 Z 相信号状态从无到有为上升沿
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行

bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误

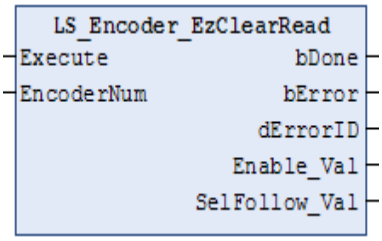
说明：

- 这个指令由“PMC_Controller”库实现。
- 编码器 Z 相清零，是当编码器在走到 Z 相信号有效的位置时，编码器数值会被清零。可选择上升沿或下降沿进行数值清零。
- 当指令的“EnableClear”信号持续为 TRUE 时，执行编码器 Z 相清零。

EZ 清零配置读取 LS_Encoder_EzClearRead

编码器值遇 Z 信号清零，读取配置参数。。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_Encoder_EzClearRead	FUN		LS_Encoder_EzClearRead(Execute:= （参数）， EncoderNum:= （参数）， bDone=>（参数） ， bError=>（参数） ， dErrorID=>（参数）， Enable_Val=>（参数） ， SelFollow_Val=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
EncoderNum	编码器号	UINT	0-2	0	编码器号：0—2
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误
Enable_Val	使能状态	BOOL	TRUE FALSE	FALSE	读取清零使能信号
SelFollow_Val	触发沿	UINT	0，1	0	读取配置的启动 Ez 清零沿

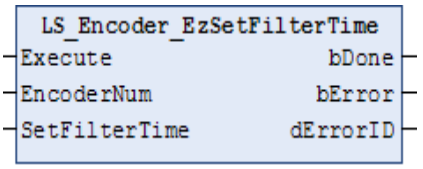
说明：

- 这个指令由“PMC_Controller”库实现。
- 指令用来读取编码器 Z 相清零功能的设置参数，与“LS_Encoder_EzClearSet”指令配合使用。

EZ 滤波参数设置　LS_Encoder_EzSetFilterTime

设置编码器 Z 相信号的滤波参数。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_Encoder_EzSetFilterTime	FUN		LS_Encoder_EzSetFilterTime(Execute：= （参数）， EncoderNum：= （参数）， SetFilterTime：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）);

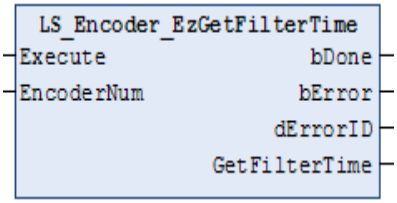
变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
EncoderNum	编码器号	UINT	0-2	0	编码器号：0—2
SetFilterTime	滤波时间	UDINT	正数	400	设置滤波时间，单位 us。建议值大于 400us 以上
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误

EZ 滤波参数读取　LS_Encoder_EzGetFilterTime

获取编码器 Z 信号滤波时间参数。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_Encoder_EzGetFilterTime	FUN		LS_Encoder_EzGetFilterTime(Execute：= （参数）， EncoderNum：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）， GetFilterTime=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
EncoderNum	编码器号	UINT	0-2	0	编码器号：0—2
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误
GetFilterTime	执行中	UDINT	TRUE FALSE	0	读取滤波时间，单位 us

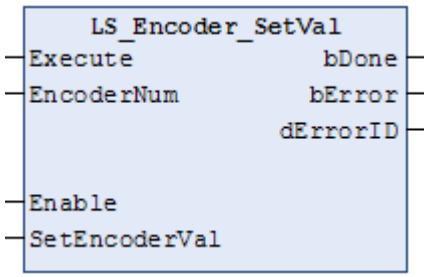
说明：

- 这个指令由“PMC_Controller”库实现。
- 用于获取编码器端口的滤波时间参数。
- 与指令“LS_Encoder_EzSetFilterTime”配合使用。

设置编码器数值 LS_Encoder_SetVal

设置编码器的数值。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_Encoder_SetVal	FUN		LS_Encoder_SetVal(Execute：= （参数）， EncoderNum：= （参数）， Enable：= （参数）， SetEncoderVal：= （参数）， bDone=>（参数） ， bError=>（参数）， dErrorID=>（参数）);

变量：

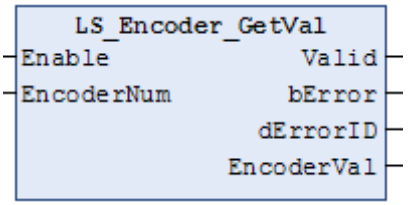
VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
EncoderNum	编码器号	UINT	0-2	0	编码器号：0—2
Enable	执行指令	LREAL	负数，0，正数	0	TRUE：允许写入； FALSE：不允许写入
SetEncoderVal	编码器数值	DINT	负数，0，正数	0	当前需要写入编码器的值，有效范围[-2147483648，2147483648]
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE	FALSE	TRUE：表示指令执行完成；

			FALSE		FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误

读取编码器数值 LS_Encoder_GetVal

获取编码器数值。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_Encoder_GetVal	FUN		LS_Encoder_GetVal(Enable：= （参数）， EncoderNum：= （参数）， Valid=>（参数） ， bError=>（参数） ， dErrorID=>（参数）， EncoderVal=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Enable	执行指令	BOOL	TRUE FALSE	FALSE	TRUE：持续执行指令 FALSE：不执行指令
EncoderNum	编码器号	UINT	0-2	0	编码器号：0—2
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误
EncoderVal	编码器数值	DINT	负数，0，正数	0	当前需要写入编码器的值，有效范围[-2147483648，2147483648]

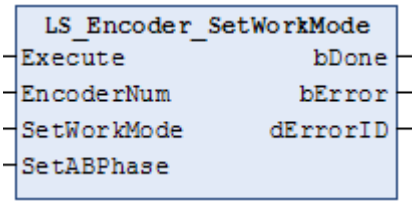
说明：

- 这个指令由“PMC_Controller”库实现。
- 与指令“LS_Encoder_SetVal”配合进行使用。

设置编码器工作模式 LS_Encoder_SetWorkMode

设置编码器的工作模式与相位参数。

指令外观：

指令	FB/ FUN	图形模块	结构文本
LS_Encoder_Set WorkMode	FUN		LS_Encoder_SetWorkMode(Execute：= （参数）， EncoderNum：= （参数）， SetWorkMode：= （参数）， SetABPhase：= （参数）， bDone=>（参数）， bError=>（参数）， bErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
EncoderNum	编码器号	UINT	0-2	0	编码器号：0—2
SetWorkMode	工作模式	SINT	0-2	0	编码器的工作模式：0-四倍频模式（AB 相模式）;1-脉冲方向模式; 2-表示单向计数
SetABPhase	相位	SINT	0，1	0	设置编码器反馈的相位：0-在正方向 A 相比 B 相超前 90 度；1-在正方向 B 相比 A 相超前 90 度
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误

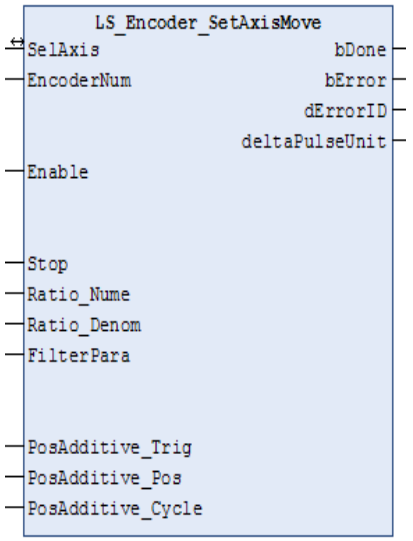
说明：

- 这个指令由“PMC_Controller”库实现。
- 指令用于设置编码器的工作模式与相位参数，以适应与不同的脉冲型伺服、步 进驱动器之间的配合使用。

编码器数值转换轴运动 LS_Encoder_SetAxisMove

将编码器接收到的位置值，转化成为指定轴的运动。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_Encoder_SetAxisMove	FB		LS_Encoder_SetAxisMove(SelAxis:= （参数）， EncoderNum:= （参数）， Enable:= （参数）， Stop:= （参数）， Ratio_Nume:= （参数）， Ratio_Denom:= （参数）， FilterPara:= （参数）， PosAdditive_Trig:= （参数）， PosAdditive_Pos:= （参数）， PosAdditive_Cycle:= （参数）， bDone=>（参数）， bError=>（参数）， bErrorID=>（参数）， deltaPulseUnit=>（参数）);

变量：

VAR_IN_OUT	名称	类型	有效范围	初始值	描述
SelAxis	轴	AXIS_REF_VIRTUAL_SM3	—	—	配置需要运动的轴，需要执行运动的轴
VAR_INPUT					
EncoderNum	编码器号	UINT	0-2	0	编码器号：0-2
Enable	执行运动	BOOL	TRUE FALSE	FALSE	TRUE：持续将编码器信号转换为轴运动；FALSE：不执行指令
Stop	停止	BOOL	TRUE FALSE	FALSE	上升沿触发指令的停止执行
Ratio_Nume	放大倍率分子	INT	1-10000	1	配置放大倍率，放大倍率分子，范围 1-10000
Ratio_Denom	放大倍率分母	INT	1-10000	1	配置放大倍率，放大倍率分母，范围 1-10000
FilterPara	平滑系数	UINT	1-100	5	平滑系数，参数越大，数据平滑增强，范围 1-100
PosAdditive_Trig	位置叠加触发	BOOL	TRUE FALSE	FALSE	位置叠加触发信号，上升沿有效。
PosAdditive_Pos	叠加位置	LREAL	负数，0 正数	0	叠加的位置。叠加的位置不参与倍率运算
PosAdditive_Cycle	叠加周期	UDINT	正数	1	叠加的位置在几个周期内完成
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误
deltaPulseUnit	每周期输出的脉冲	LREAL	0，正数	0	每周期输出的脉冲单位

说明：

- 这个指令由“PMC_Controller”库实现。
- 可以通过调整“Ratio_Nume”和“Ratio_Denom”参数，来配置编码器位置值和轴运动的位置值的比例。
- 可以设置“FilterPara”过滤器参数，参数值越大，数据平滑增强。
- 有位置叠加功能，通过触发“PosAdditive_Trig”信号，让跟随轴在参数“PosAdditive_Cycle”指定的周期内，进行位置值为“PosAdditive_Pos”的位置叠加。

9.2.2 编码器应用与例程

编码器的模式可以通过 Encoder_SetAxisMove 函数进行设置，分为两种模式：将参数 SetWorkMode 设为 0 是 AB 相模式，设为 1 是脉冲方向模式。

通过 SMC606 库的变量 Axis_Encoder 可以实时读取编码器 0-5 的数值。

另外，通过 Encoder_SetVal 可以对编码器数值进行设置。

例程 9.1：让轴 0 执行速度为 5000Pulse/s、加减速度为 10000Pulse/s^2、运动距离为 100000Pulse 的相对运动，并实时读取增量编码器反馈的位置。

1) 新建工程并命名 Encoder，编程语言为结构化文本。

2) 添加脉冲方向和硬件 IO 处理模块：右击 PLC_PRG，选择“添加对象”-“动作”，命名 ACT_PD606_IO_Cmd，编程语言为梯形图 LD，双击进入程序编辑区，添加 PD606_IO_Cmd 模块并在指令中填上对应的轴号。如图 9.1 所示。

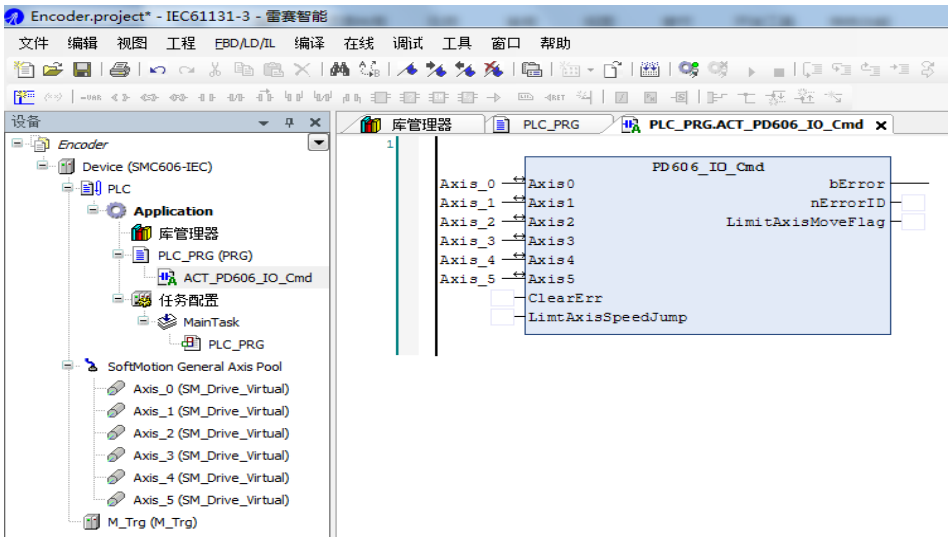


图 9.1 初始化模块

3) 按照步骤 2 的方法依次添加上电模块 Power、相对运动模块 ACT_RelaMove、编码器设置模块 ACT_EncoderSet，并填写相应的参数，如图 9.2~9.4 所示。

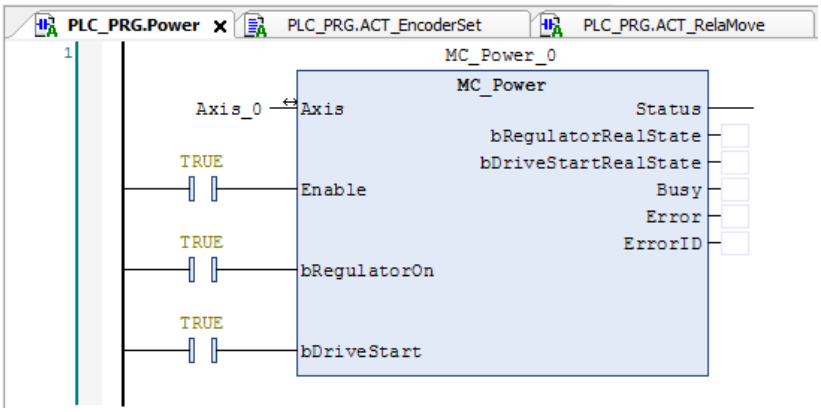


图 9.2　使能模块

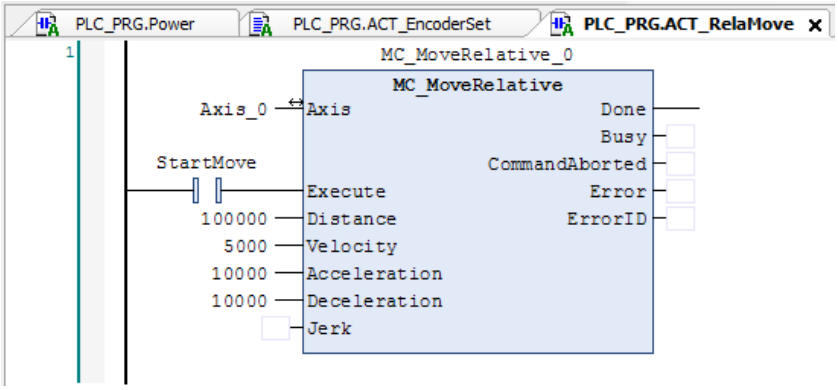


图 9.3　相对运动模块

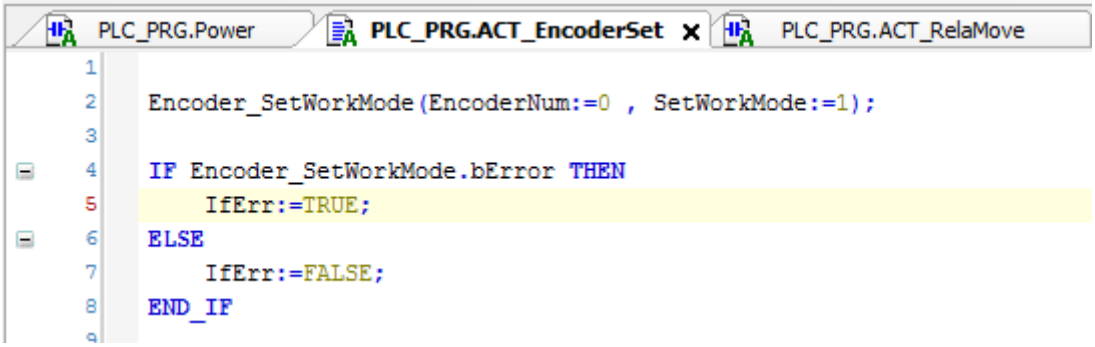


图 9.4　编码器设置模块

4) 在主程序中读取轴 0 的编码器值并调用这些模块，如图 9.5 所示。编译、下载程序并执行。

5) 运行结果：实时读取到轴 0 编码器反馈的值并将其赋值给变量 Encoder_0；如果变量 ClearEncoder 为 TRUE 则将编码器计数值清零，结果如图 9.5 所示。

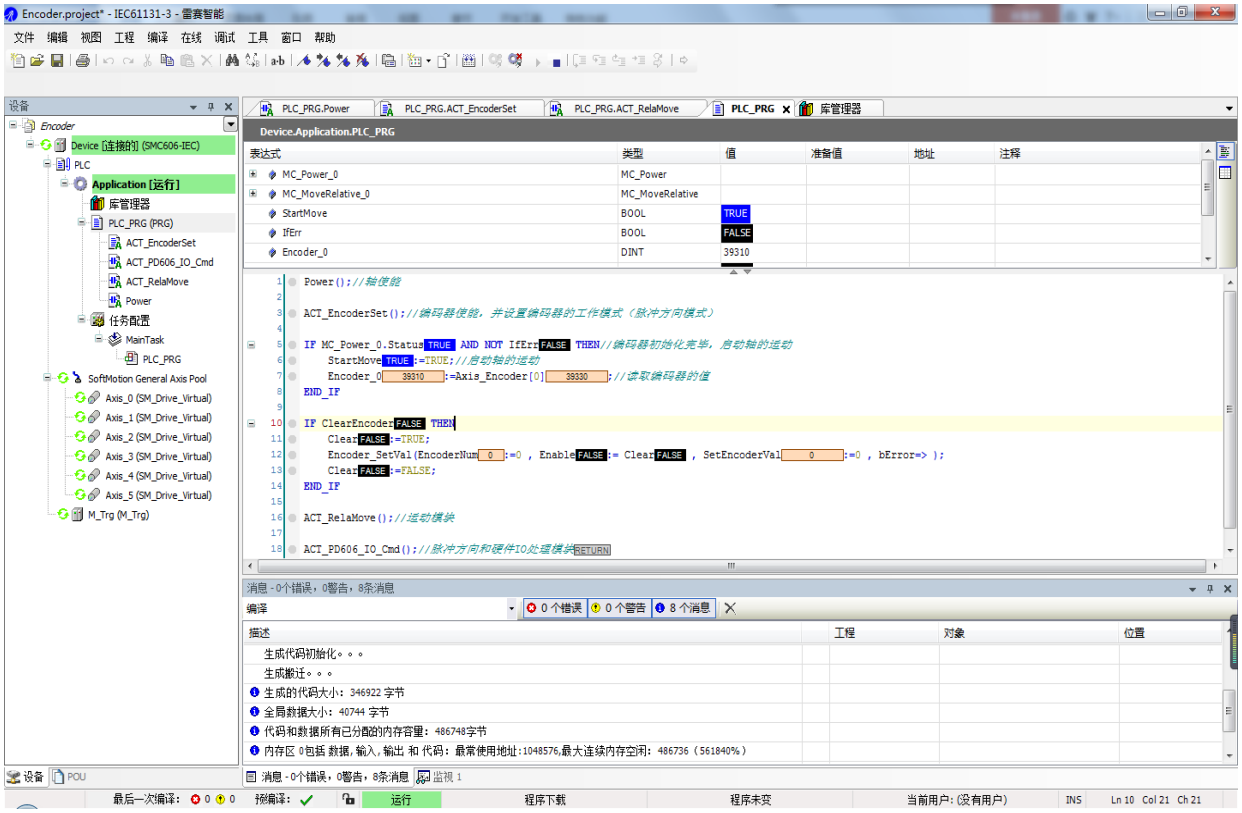


图 9.5　程序运行结果

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**例程编码器-Encoder**”。

9.2.3　EZ 清零的使用及例程

编码器 EZ 清零功能，能够让编码器值在遇到 Z 相信号时会自动清零，适用于需要用到编码器单圈内数值的场合，如编码器一圈为 10000Pulse，那无论往哪个方向转，只要碰到 Z 相信号，编码器数值就会被置零。无论往同个方向转多少圈，数值变化范围不超过 10000。

例程 9.2：设置轴 0 执行速度为 1000Pulse/s、加减速度为 10000Pulse/s^2，运动方式为正方向匀速运动，当轴 0 的位置超过 30000Pulse 的时候，开启 EZ 清零功能。

1) 新建工程并命名 EZ 编码器清零，编程语言为结构化文本 ST。

2) 添加脉冲方向和硬件 IO 处理模块：右击 PLC_PRG，选择“添加对象”-“动作”，命名 ACT_PD606_IO_Cmd，编程语言为梯形图 LD，双击进入程序编辑区，添加 PD606_IO_Cmd 模块并在指令中填上对应的轴号，如图 9.6 所示。

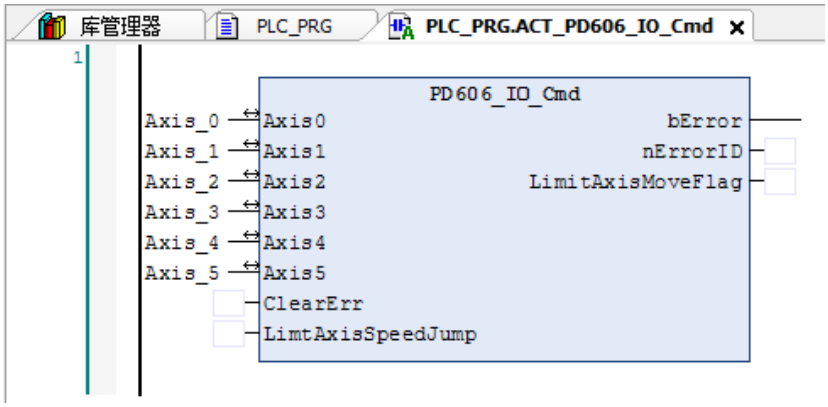


图 9.6　初始化模块

3) 按照步骤 2 的方法依次添加上电模块 Power、相对运动模块 ACT_Move、编码器设置模块 ACT_EZClear，并填写相应的参数，如图 9.7~9.9 所示。

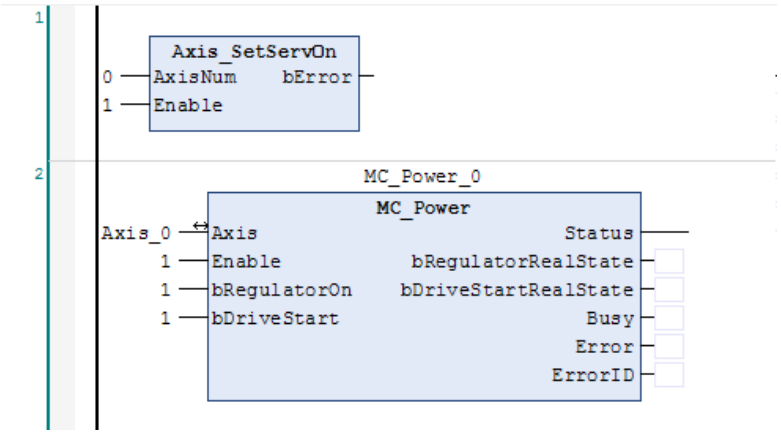


图 9.7　使能模块

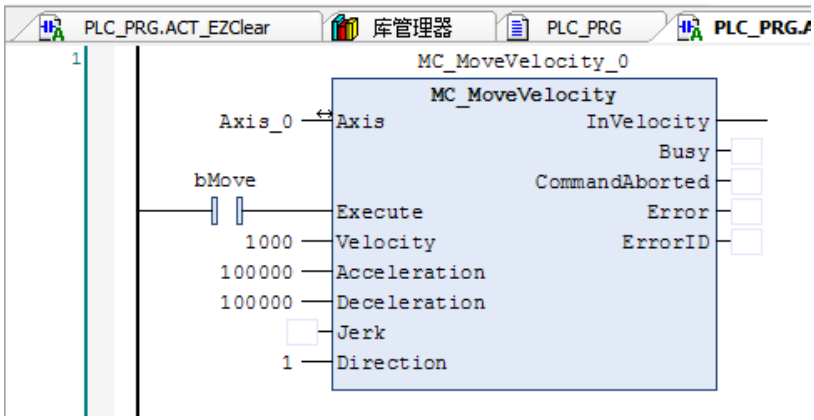


图 9.8　运动模块

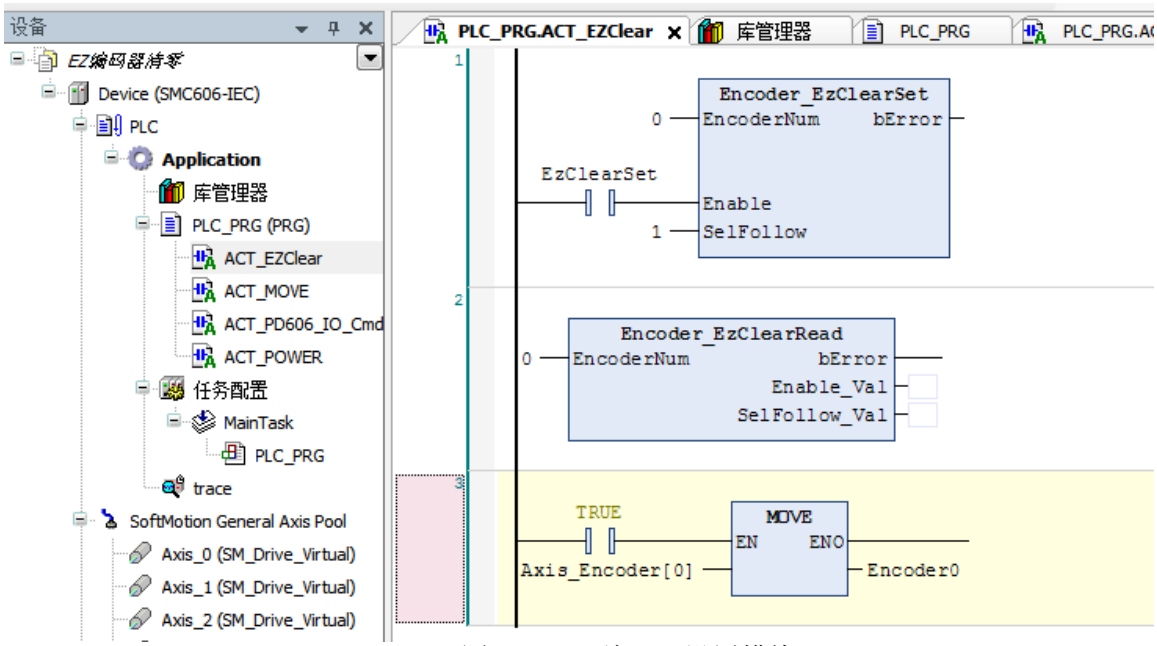


图 9.9　EZ 编码器设置模块

4) 在主程序中读取轴 0 的编码器值并调用这些模块。编译、下载程序到控制器执行。然后将 istep 置为 1，程序就能自动执行，结果如图 9.10 所示。

图 9.10　位移、编码器值、EZ 清零信号的曲线

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**编码器-EZ 编码器清零**”。

9.3 高速位置比较

PMC610 有 4 个高速位置比较器通道，每个高速比较输出端口都可以与任意轴关联，支持多种比较模式。4 个锁存通道对应的高速口分别是 OUT16-OUT19，电路中使用的光耦响应频率可达 200KHz。

高速位置比较器通道可以通过调用“PMC_Controller”库的指令进行设置，也可以在编程软件“IEC STUDIO”上进行配置。具体的配置方法可以参考《PMC600 中型 PLC 用户手册 2—软件编程篇》。

高速比较功能支持 7 种比较模式：0--不工作；1--计数器的值等于比较器；2--计数器的值小于比较器；3--计数器的值大于比较器；4--FIFO 模式 1（设置输出时间）；5--Linear 模式；6--FIFO 模式 2（设置输出电平的高低）。

注意：

1） 每个比较器的位置比较都是独立进行的。

2） 在两种 FIFO 及线性 Linear 比较模式中，执行位置比较时，每个比较点的触发是按照添加的比较点顺序执行的，即如果有一个比较点没有被触发比较动作，那么后面的比较点是不会被触发的。

3） 高速比较输出的最大误差为扫描周期与运动速度的乘积。

表 9.5 高速比较指令

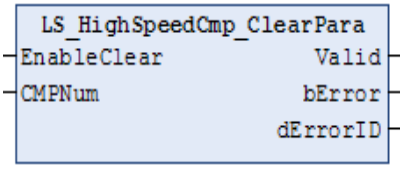
指令名	功能说明
LS_HighSpeedCmp_ClearPara	清除比较器
LS_HighSpeedCmp_SetWorkPara	设置比较器参数
LS_HighSpeedCmp_GetWorkPara	读取比较器参数
LS_HighSpeedCmp_SetCmpPos	设置比较位置
LS_HighSpeedCmp_SetLinearPara	设置线性模式参数
LS_HighSpeedCmp_GetLinearPara	读取线性模式参数
LS_HighSpeedCmp_SetFIFOPos	设置 FIFO 模式比较位置
LS_HighSpeedCmp_SetFIFOPosType2	设置 FIFO 模式 2 比较位置
LS_HighSpeedCmp_GetCurState	获取比较器当前状态

9.3.1 高速位置比较指令

清除比较器 LS_HighSpeedCmp_ClearPara

清除高速比较器的参数和所有状态值。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedCmp_ClearPara	FUN		LS_HighSpeedCmp_ClearPara(EnableClear:= （参数）, CMPNum:= （参数）, Valid=>（参数） , bError=>（参数） , dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
EnableClear	比较器使能	BOOL	TRUE FALSE	FALSE	TRUE：持续清除比较器 FALSE：不执行指令
CMPNum	比较器通道	SINT	0-3	0	比较器通道：0-3
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误

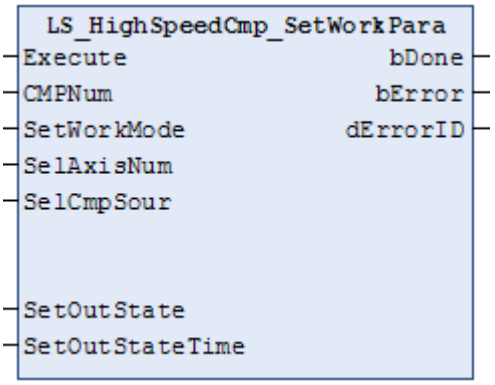
说明：

- 这个指令由“PMC_Controller”库实现。
- 清除高速比较器的参数和所有状态值，包括比较位置，FIFO 数据等。

设置比较器参数 LS_HighSpeedCmp_SetWorkPara

设置一维高速比较器的参数。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedCmp_SetWorkPara	FUN		LS_HighSpeedCmp_SetWorkPara(Execute:= （参数）, CMPNum:= （参数）, SetWorkMode:= （参数）, SelAxisNum:= （参数）, SelCmpSour:= （参数）, SetOutState:= （参数）, SetOutStateTime:= （参数）, bDone=>（参数） , bError=>（参数） , dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0-3	0	比较器通道：0-3
SetWorkMode	比较模式	UINT	0-6	0	设置比较器工作模式：0-不工作；1-计数器的值等于比较器；2-计数器的值小于比较器；3-计数器的值大于比较器;4-FIFO 模式;5-Linear 模式；6-FIFO 模式 2。
SelAxisNum	比较轴号	UINT	0-3	0	选择需要比较的轴号：0-3
SelCmpSour	比较源	UINT	0，1	0	选择需要比较的数据源：0-指令位置；1-反馈位置
SetOutState	输出电平逻辑	BOOL	TRUE FALSE	FALSE	设置输出端口的有效状态：False-条件成立输出 false，指示灯灭；True-条件成立输出 True,指示灯亮
SetOutStateTime	输出时间	UINT	0-20000000	0	在 FIFO 模式和 Linear 模式下才使用。用于控制输出端口保持有效状态时间，单位 us：范围 0-20000000
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误

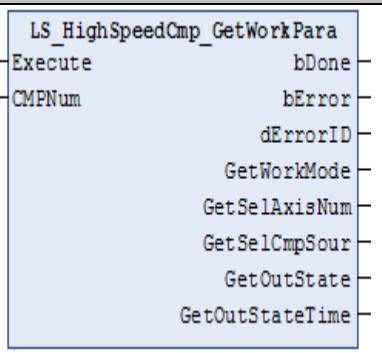
说明：

- 这个指令由“PMC_Controller”库实现。
- 在模式 1-3 中，当比较器满足设置的位置“等于”、“小于”和|“大于”的参数条件后，设定的高速比较输出口会持续保持输出状态。
- 在 FIFO 模式 1 和 Linear 模式下，比较器满足条件后，SetOutStateTime 控制输出端口保持有效状态的时间。
- Linear 模式的比较器增量和点数，在指令“LS_HighSpeedCmp_SetLinearPara”中进行设置。
- FIFO 模式 1 的比较器点数和位置，在指令“LS_HighSpeedCmp_SetFIFOPos”中进行设置。
- 在 FIFO 模式 2 中，比较器满足条件后，输出状态一直保持，直到遇到下一比较点。比较器的位置和输出状态，在指令“LS_HighSpeedCmp_SetFIFOPosType2”中进行设置。

读取比较器参数 LS_HighSpeedCmp_GetWorkPara

获取一维高速比较器的参数。

指令外观：

指令	FB/ FUN	图形模块	结构文本
LS_HighSpeedCmp_ GetWorkPara	FUN		LS_HighSpeedCmp_GetWorkPara(Execute：= （参数）， CMPNum：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）， GetWorkMode=>（参数）， GetSelAxisNum=>（参数）， GetSelCmpSour=>（参数）， GetOutState=>（参数）， GetOutStateTime=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0-3	0	比较器通道：0-3
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误
GetWorkMode	比较模式	UINT	0-6	0	比较器工作模式：0-5
GetSelAxisNum	比较轴号	UINT	0-3	0	选择需要比较的轴号：0-3
GetSelCmpSour	比较源	UINT	0，1	0	选择需要比较的数据源：0-指令位置；1-反馈位置
GetOutState	输出电平逻辑	BOOL	TRUE FALSE	FALSE	设置输出端口的有效状态：False-条件成立输出 false，指示灯灭；True-条件成立输出 True，指示灯亮
GetOutStateTime	输出时间	UINT	0-20000000	0	在 FIFO 模式和 Linear 模式下才使用，用于控制输出端口保持有效状态时间，单位 us

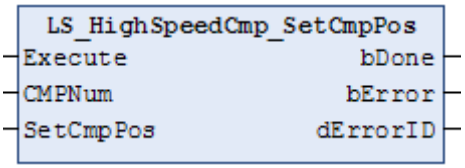
说明：

- 这个指令由“PMC_Controller”库实现。
- 与“LS_HighSpeedCmp_SetWorkPara”指令配合进行使用。

设置比较位置　LS_HighSpeedCmp_SetCmpPos

设置除 FIFO 模式外的一维高速比较器的比较位置。

指令外观：

指令	FB/ FUN	图形模块	结构文本
LS_HighSpeedCmp_ SetCmpPos	FUN		LS_HighSpeedCmp_SetCmpPos(Execute：= （参数）， CMPNum：= （参数）， SetCmpPos：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0-3	0	比较器通道：0-3
SetCmpPos	比较位置	DINT	负数，0，正数	0	设置比较位置
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误

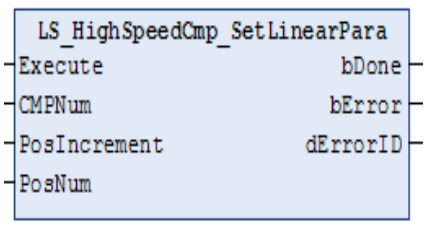
说明：

- 这个指令由“PMC_Controller”库实现。
- 设置或更新比较器的比较位置，仅在模式 1/2/3/5 有效。
- 其中，在模式 5 线性模式下，用来更新起始比较位置；在其他模式下，更新比较位置。

设置线性模式参数 LS_HighSpeedCmp_SetLinearPara

设置一维高速比较器模式 5Linear 模式的参数。

指令外观：

指令	FB/ FUN	图形模块	结构文本
LS_HighSpeedCmp_ SetLinearPara	FUN		LS_HighSpeedCmp_SetLinearPara(Execute：= （参数）， CMPNum：= （参数）， PosIncrement：= （参数）， PosNum：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0-3	0	比较器通道：0-3
PosIncrement	位置增量	DINT	负数，0 正数	0	设置 Linear（线性）模式下比较点位置增量
PosNum	比较点数	SINT	0-128	0	设置 Linear（线性）模式下比较点个数，范围：0-128
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误

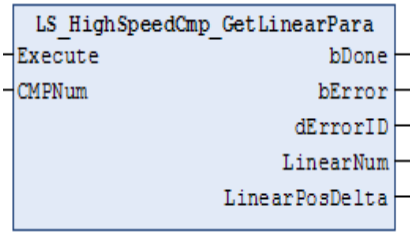
说明：

- 这个指令由“PMC_Controller”库实现。
- 设置一维高速比较器模式 5Linear 模式的位置增量和比较点数。
- 当比较点设置为 0 或 1 时，都会有且只有输出一个比较点，对应指令“LS_HighSpeedCmp_SetCmpPos”所设置的比较位置。
- 当设置的比较点个数大于 1，且位置增量也不等于 0，则输出的高速比较点将从“LS_HighSpeedCmp_SetCmpPos”指令所设置的比较位置开始输出第 1 个比较点，之后如果运动方向保持，将以增量“PosIncrement”，输出剩下的“PosNum-1”个比较点。
- 注意，如果 2 次高速比较输入相距很近，则自动重合为一个脉冲。

读取线性模式参数 LS_HighSpeedCmp_GetLinearPara

获取一维高速比较器 Linear 模式下参数。

指令外观：

指令	FB/ FUN	图形模块	结构文本
LS_HighSpeedCmp_ GetLinearPara	FUN		LS_HighSpeedCmp_GetLinearPara(Execute：= （参数）， CMPNum：= （参数）， bDone=>（参数） ， bError=>（参数） ， dErrorID=>（参数）， LinearNum=>（参数） ， LinearPosDelta=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0-3	0	比较器通道：0-3
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码
LinearNum	比较点数	SINT	0-128	0	读取 Linear（线性）模式下比较点个数，范围：0-128
LinearPosDelta	位置增量	DINT	负数，0 正数	0	读取 Linear（线性）模式下比较点位置增量

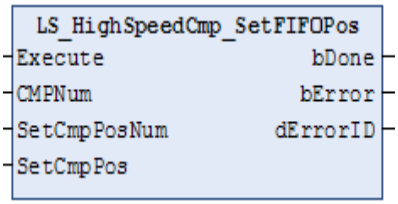
说明：

- 这个指令由“PMC_Controller”库实现。
- 用来获取一维高速比较 Linear 模式的比较点和位置增量参数。
- 与“LS_HighSpeedCmp_SetLinearPara”指令配合进行使用。

设置 FIFO 模式比较位置　　LS_HighSpeedCmp_SetFIFOPos

设置一维高速比较器 FIFO 模式的比较位置。

指令外观：

指令	FB/ FUN	图形模块	结构文本
LS_HighSpeedCmp_ SetCmpPos	FUN		LS_HighSpeedCmp_SetCmpPos(Execute：= （参数）， CMPNum：= （参数）， SetCmpPosNum：= （参数）， SetCmpPos：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0-3	0	比较器通道：0-3
SetCmpPosNum	比较点数	SINT	0-128	0	设置高速比较的点数
SetCmpPos	比较位置	ARRAY[0..127] OF DINT	负数，0 正数	0	设置高速比较位置数组
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行

bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误

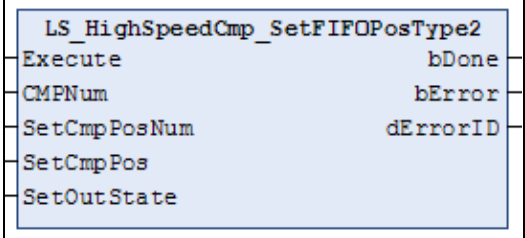
说明：

- 这个指令由“PMC_Controller”库实现。
- 设置一维高速比较器 FIFO 模式的比较位置。用于在模式 4 FIFO（设置输出时间）下，写入多个位置点，最多可以输入 128 个比较点。
- 注意：如果 2 次高速比较输入相距很近，则自动重合为一个脉冲。

设置 FIFO 模式 2 比较位置LS_HighSpeedCmp_SetFIFOPosType2

设置一维高速比较器比较位置。

指令外观：

指令	FB/ FUN	图形模块	结构文本
LS_HighSpeedCmp_ SetFIFOPosType2	FUN		LS_HighSpeedCmp_SetFIFOPosType2(Execute:= （参数）， CMPNum:= （参数）， SetCmpPosNum:= （参数）， SetCmpPos:= （参数）， SetOutState:= （参数）， bDone=>（参数） ， bError=>（参数） ， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0-3	0	比较器通道：0-3
SetCmpPosNum	比较点数	SINT	0，正数	0	设置高速比较的点数
SetCmpPos	比较位置	ARRAY[0..127] OF DINT	负数，0 正数	0	设置高速比较位置数组
SetOutState	输出口状态	ARRAY[0..127] OF BOOL	TRUE FALSE	0	设置高速输出端口的有效状态：False-条件成立输出 false，指示灯灭；True-条件成立输出 True，指示灯亮
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误

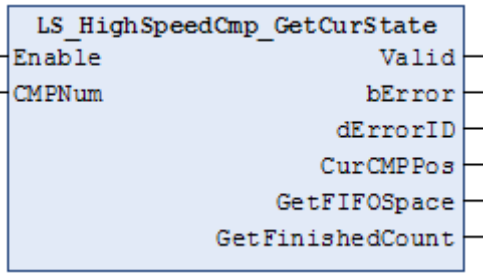
说明：

- 这个指令由“PMC_Controller”库实现。
- 设置比较位置。用于在模式 6FIFO 模式（设置输出电平的高低）下，写入多个位置点。每个位置点，可以单独配置比较输出状态。

获取比较器当前状态　LS_HighSpeedCmp_GetCurState

获取一维高速比较器当前状态。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedCmp_GetCurState	FUN		LS_HighSpeedCmp_GetCurState(Enable:= （参数）, CMPNum:= （参数）, Valid=>（参数） , bError=>（参数） , dErrorID=>（参数）, CurCMPPos=>（参数） , GetFIFOSpace=>（参数） , GetFinishedCount=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE FALSE	FALSE	TRUE：持续获取状态； FALSE：不执行指令
CMPNum	比较器通道	SINT	0-3	0	比较器通道：0-3
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误
CurCMPPos	当前比较位置	DINT	TRUE FALSE	0	获取比较器当前比较点
GetFIFOSpace	FIFO 剩余比较点	UDINT	TRUE FALSE	0	获取 FIFO 剩余比较点
GetFinishedCount	完成点数	UDINT	TRUE FALSE	0	获取已经完成的比较点个数

说明：

- 这个指令由“PMC_Controller”库实现。
- 指令可以获取一维高速比较器当前状态，包括当前目标比较点的位置、剩余比较点数、已完成比较点数等参数。

9.3.2 高速位置比较器的使用与例程

高速比较输出口的设置步骤：

1） 清除比较器的参数：用 CMP_ClearPara 指令清除比较器所有状态值，包括比较位置，FIFO 数据；

2） 配置比较器的参数：用 CMP_SetWorkPara 指令配置比较器参数，包括工作模式、关联轴号、需要比较的数据源、满足条件后比较器输出的状态、FIFO 队列和 Linear 线性模式下输出状态保持的时间；

3） 设置比较器比较位置：Linear 线性模式下，CMP_SetCmpPos 指令设置起始比较位置，CMP_SetLinearPara 指令设置比较点个数及位置增量；FIFO 模式下，CMP_SetFIFOPos 设置比较点个数及比较位置；其他模式则用 CMP_SetCmpPos 指令设置比较位置；

4） 获取比较器当前状态：如果比较器工作在线性 Linear 模式或者队列 FIFO 模式下，则需要最后调用 CMP_GetCurState 指令来获取比较器当前状态，如比较器当前比较点、已完成比较点个数、FIFO 剩余空间等，检测到已完成比较点数为设定值时再进行下一步的动作。

1. 模式 3 的例程

例程 9.3：模式 3-大于模式，关联 0 轴，当轴 0 相对运动计数值大于 10000 时高速比较器 0（OUT16）端口输出低电平并保持。

1) 新建工程并命名大于模式，编程语言为结构化文本 ST，如图 9.11 所示；

2) 右击 PLC_PRG 主程序，选择“添加对象”-“动作”，命名 ACT_PD606_CMD，负责脉冲方向模块和硬件 IO 处理模块，编程语言为梯形图 LD，在设备栏双击“ACT_PD606_CMD”进入程序编辑区，添加 PD606_IO_Cmd 模块并在指令中填上对应的轴号，如图 9.12 所示；

3) 按照步骤 2 的方法依次添加上电模块 Power、相对运动模块 RelativeMove、清除比较器状态模块 ClearCMP、大于模式模块 More，并填写相应的参数，如图 9.13~16 所示；

4) 在主程序中分别调用这些模块，编译、下载程序到控制器执行。

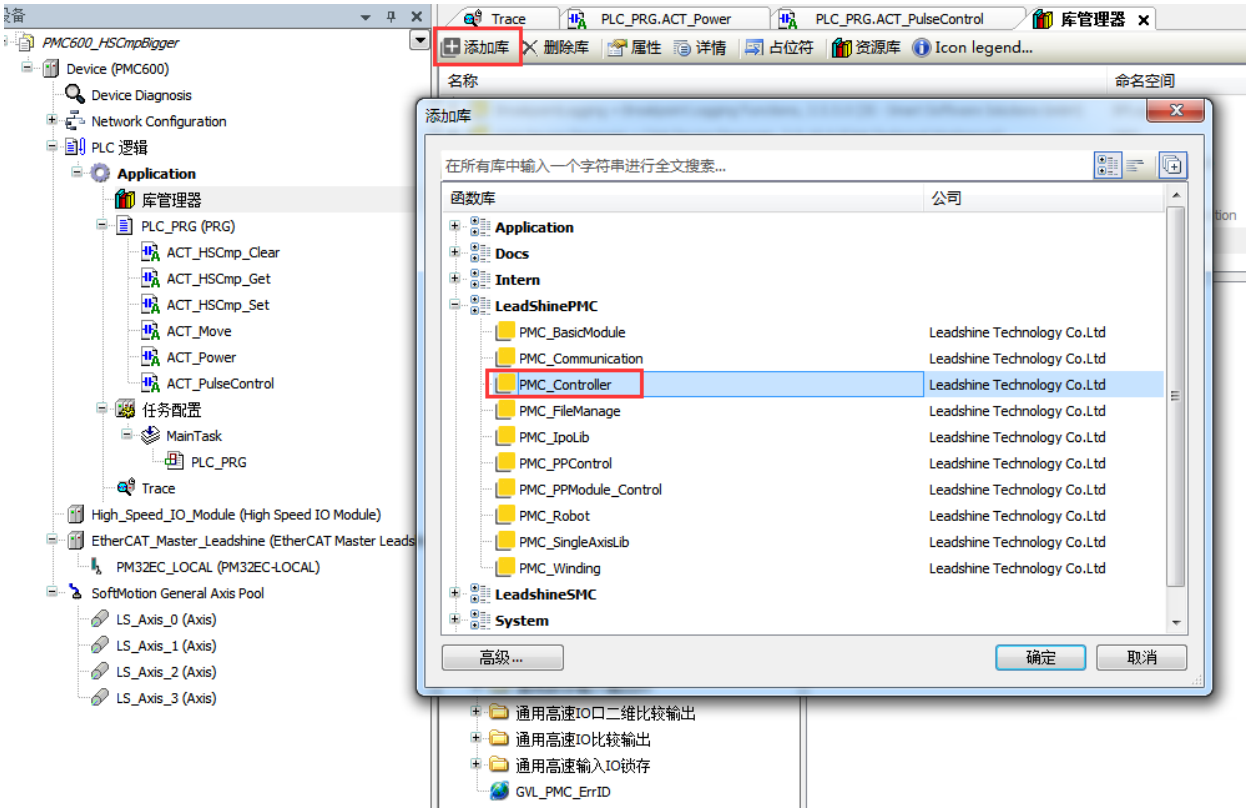


图 9.11　添加 SMC606 库

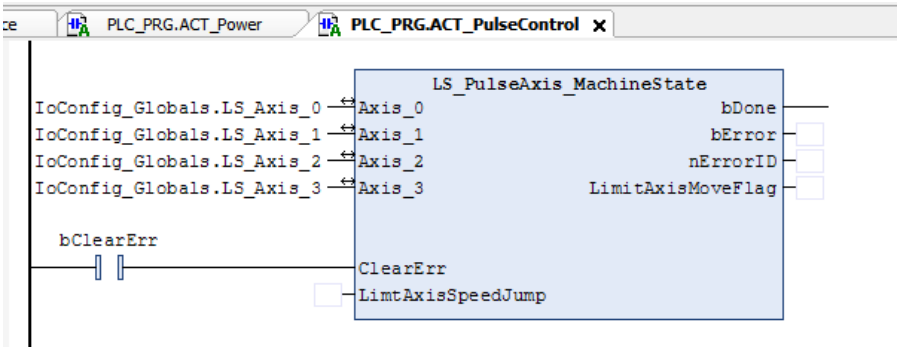


图 9.12　脉冲模块

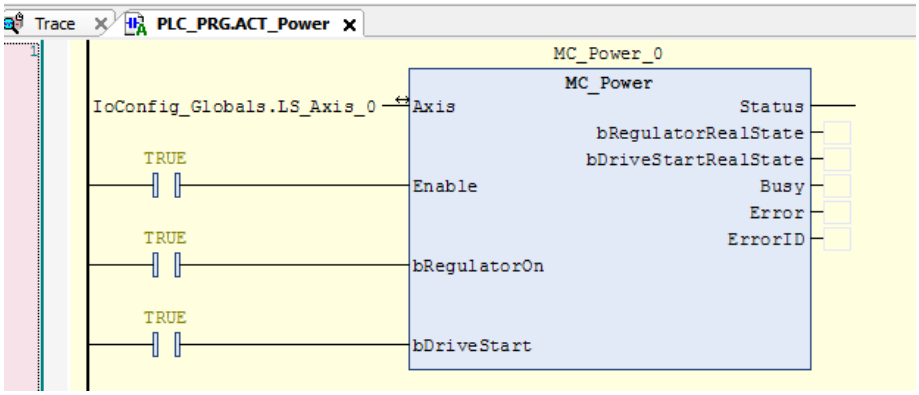


图 9.13　上电模块

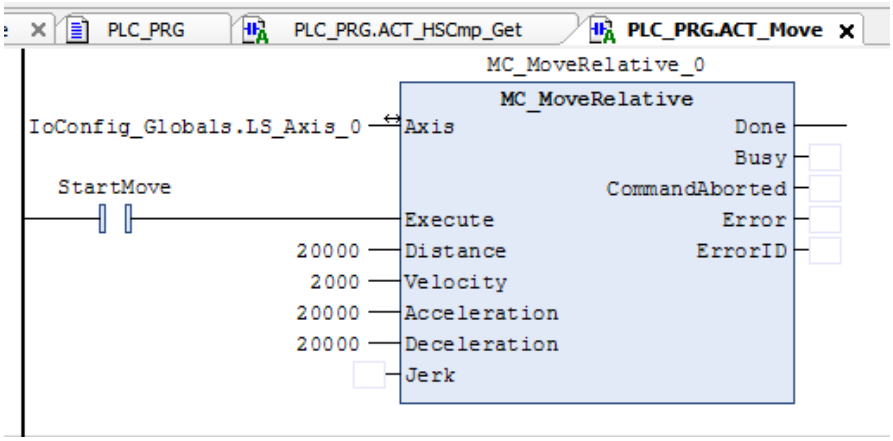


图 9.14　相对运动模块

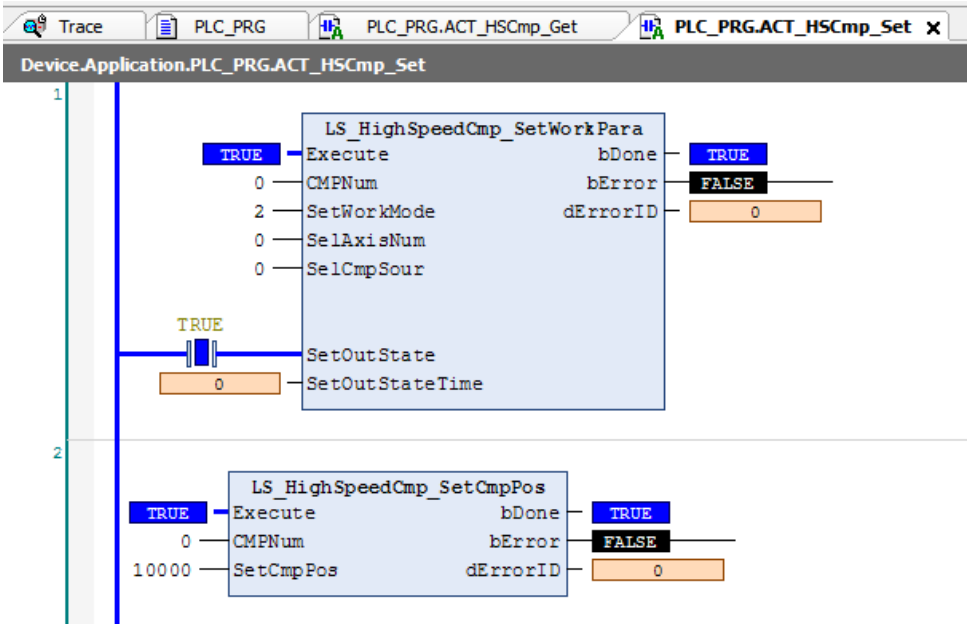


图 9.15　大于模式模块

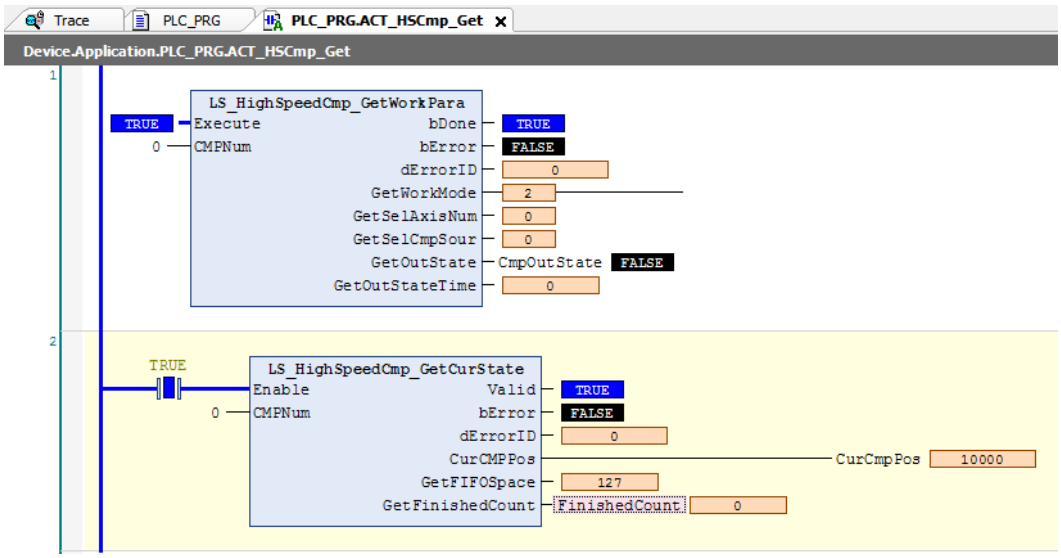


图 9.16　清除比较器模块

运行结果：当 0 号轴运动经过位置 10000Pulse 时，CMP0 端口输出低电平，并一直保持，如图 9.17、9.18 所示。

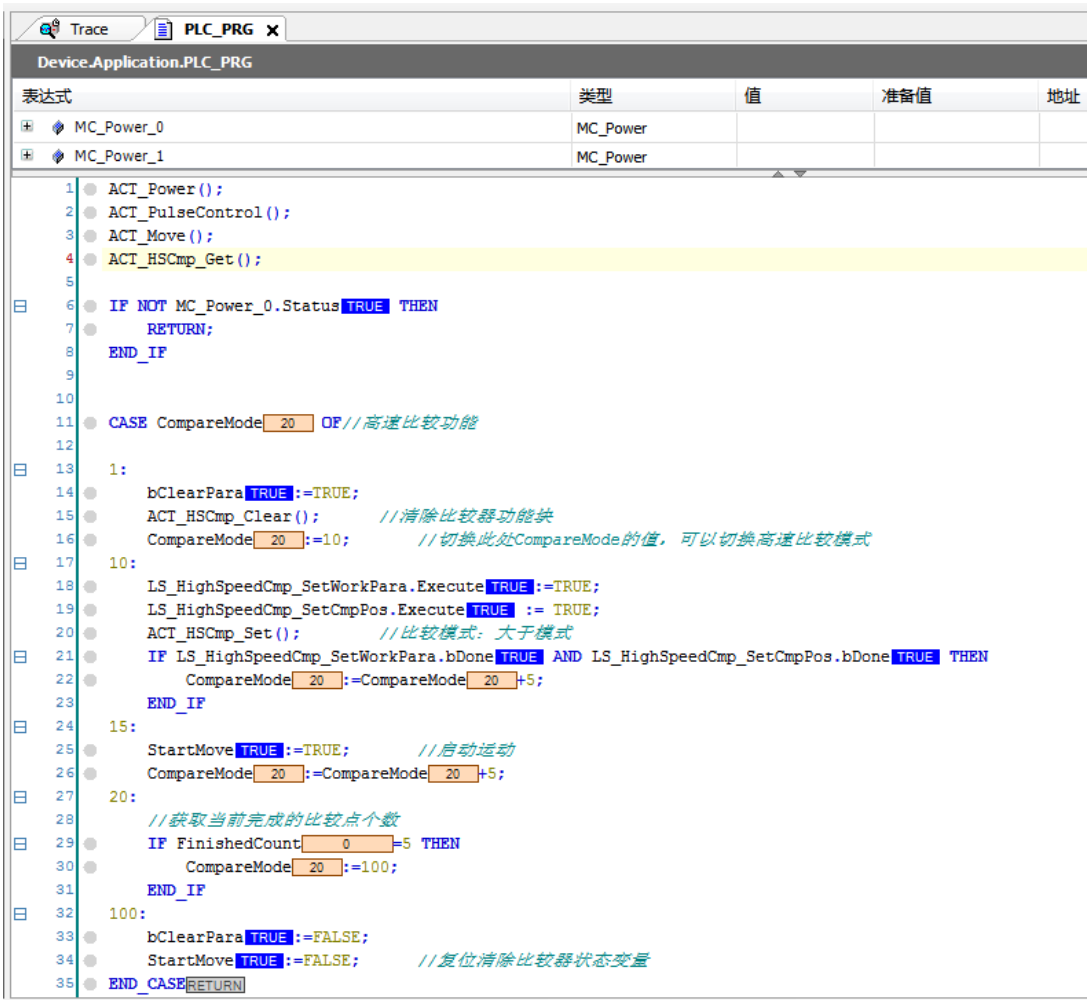


图 9.17　程序运行结果

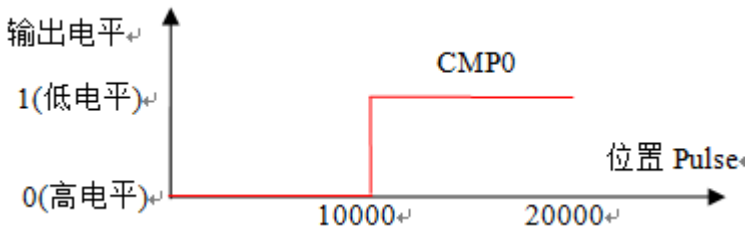


图 9.18　输出电平示意图

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**高速比较-大于模式-大于模式**”。

2. 模式 5 的例程

例程 9.4：模式 5-线性比较模式，关联 0 轴，起始比较点为 2000，增量 2000，比较点数 5 个，当 0 号轴运动经过位置 2000、4000、6000、8000、10000Pulse 时，高速比较器 0（OUT16）端口输出有效电平并维持状态时间 0.5s。

线性比较模式功能程序编写步骤同例程 9.3。在例程 9.3 的基础上，将大于模式模块 More 删除，增加线性模式模块 Line 以及读取比较器状态模块 Get_CMPState，并在该模块填写相应的参数，如图 9.19、9.20 所示。然后在主程序中分别调用这些模块，编译、下载程序到控制器中运行，如图 9.21 所示。

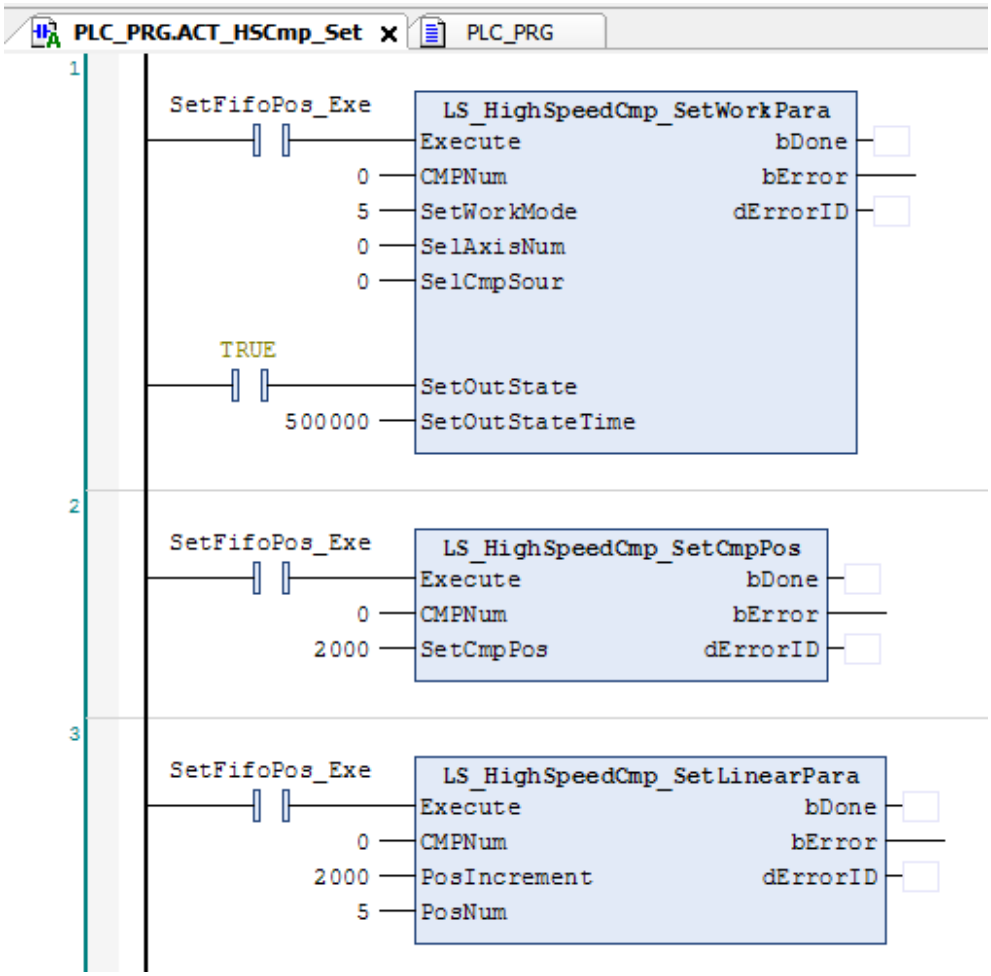


图 9.19　线性模式模块

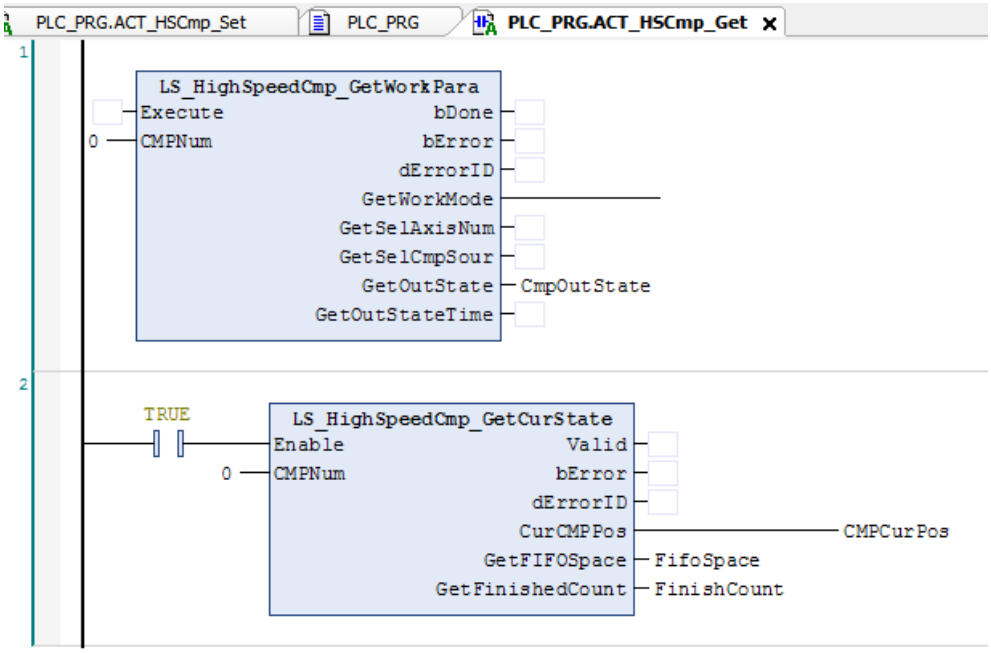


图 9.20　读取比较器状态模块

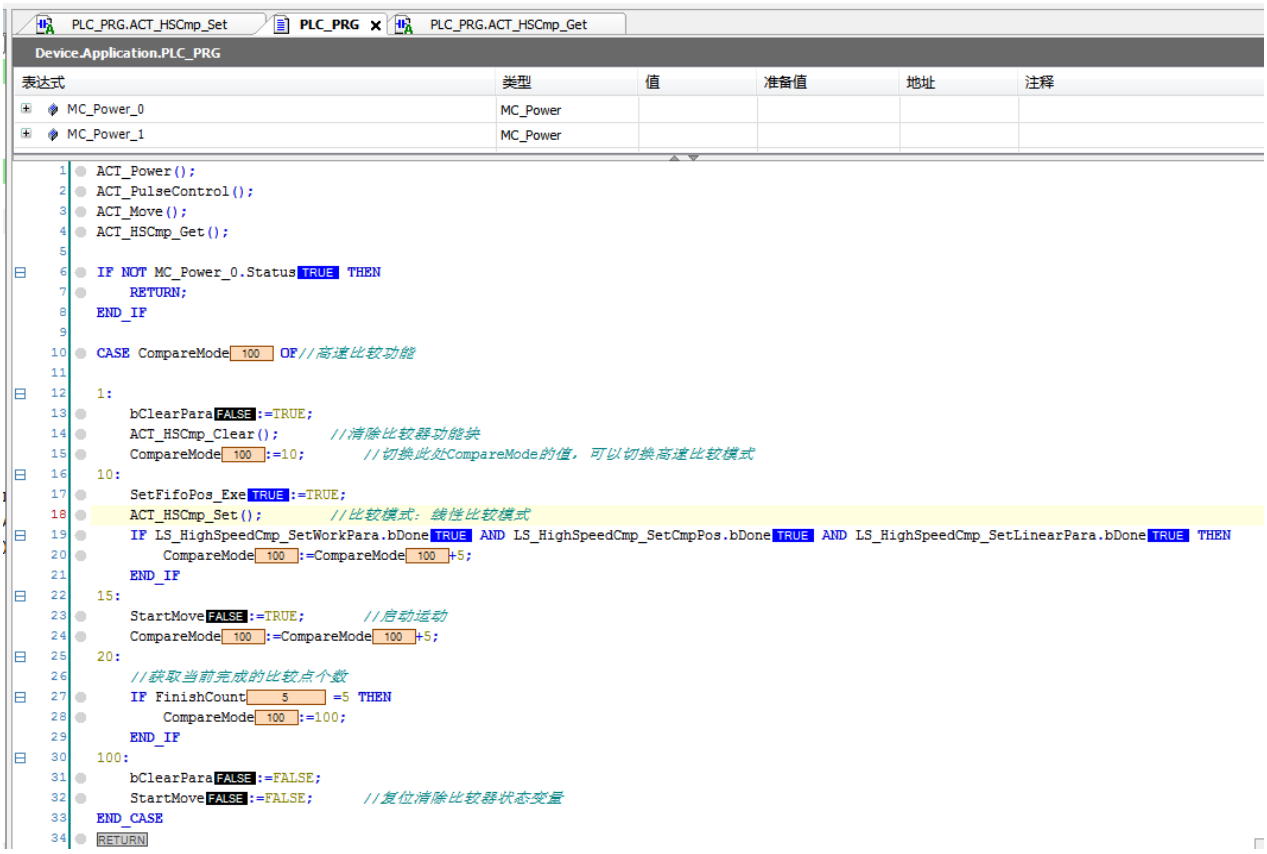


图 9.21　主程序运行结果

运行结果：当 0 号轴运动经过位置 2000、4000、6000、8000、10000Pulse 时，CMP0 端口（OUT16）输出低电平，持续 500ms，如图 9.22、9.23 所示。

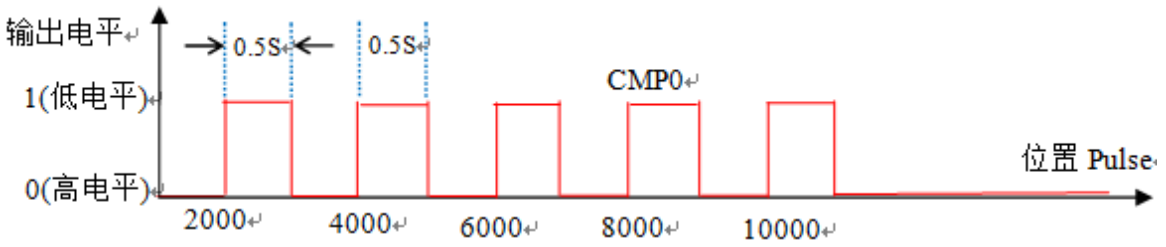


图 9.22　输出电平示意图

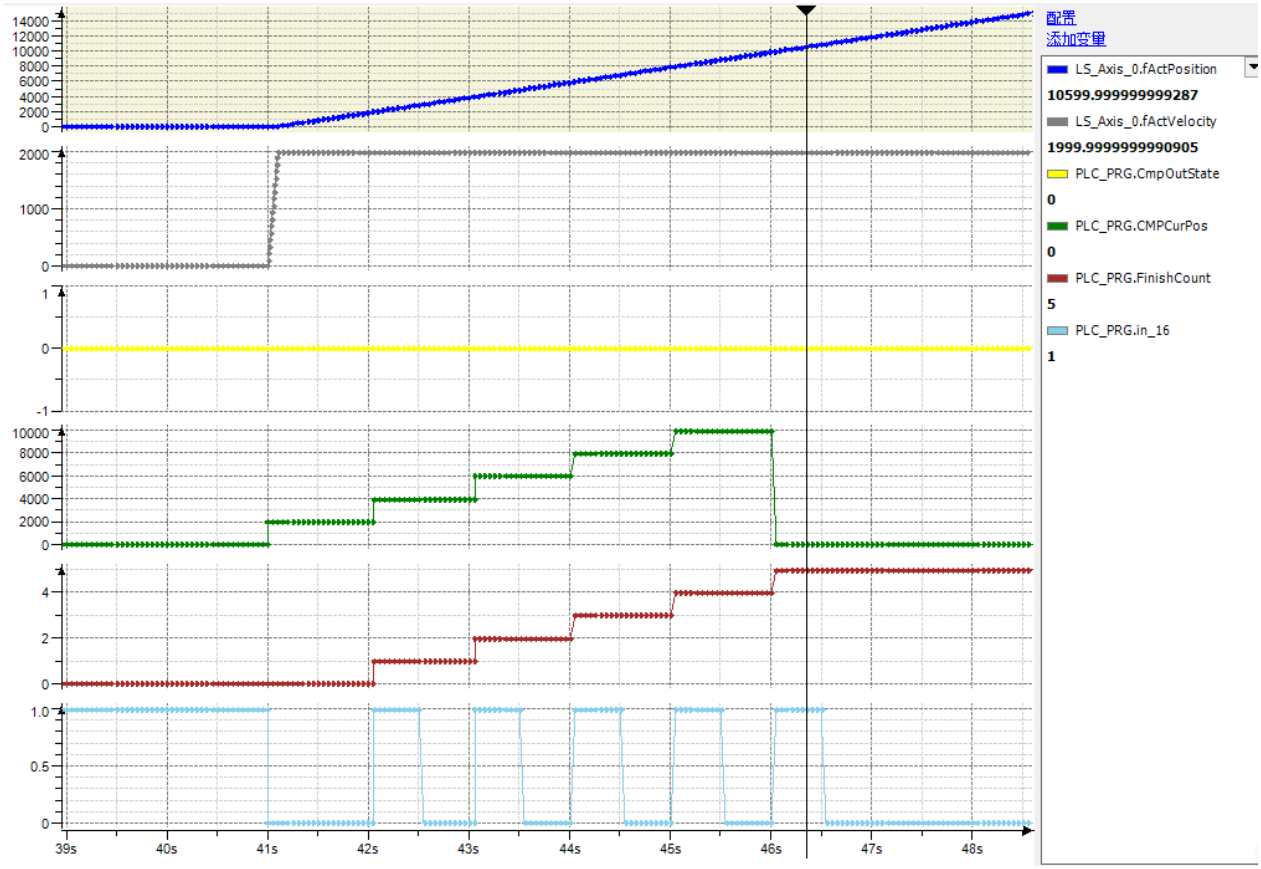


图 9.23　Trace 采集的参数曲线

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**高速比较-线性模式-线性模式**”。

3.模式 4 的例程

例程 9.5：模式 4-队列比较模式 FIFO-时间方式，关联 0 轴，比较点为 1000、5000、7000、10000、15000，比较点数 5 个，当 0 号轴运动经过位置 1000、5000、6000、10000、15000Pulse 时，高速比较器 0（OUT16）端口输出有效电平维持时间 0.5s：

队列比较模式功能程序编写步骤同例程 9.4。在例程 9.4 的基础上，将大于模式模块 More 删除，增加队列模式模块 FIFO 以及读取比较器状态模块 Get_CMPState，并在该模块填写相应的参数如图 9.24、9.25 所示，然后在主程序中分别调用这些模块，编译、下载程序到控制器中运行，如图 9.26 所示。

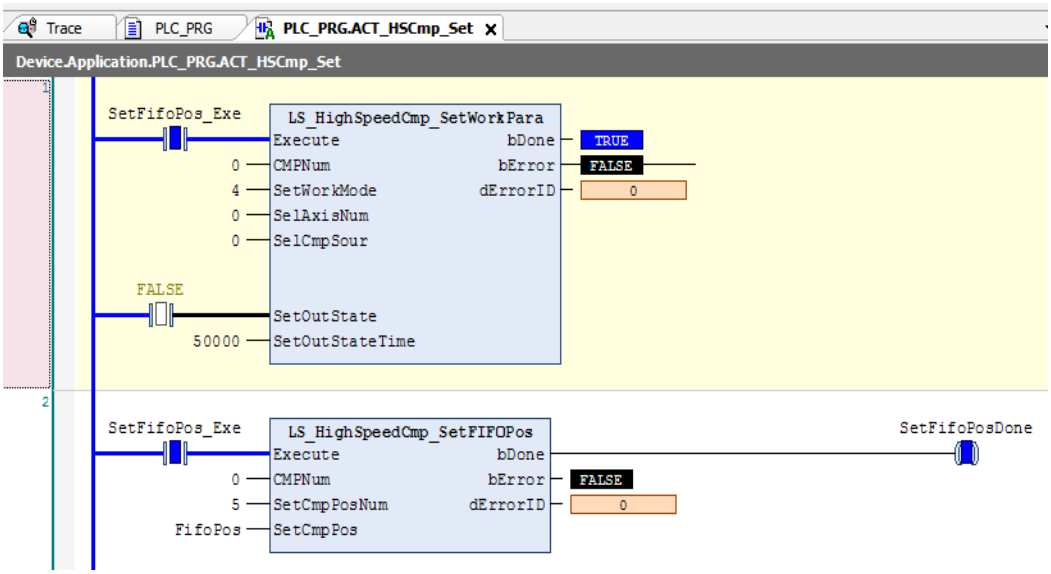


图 9.24　队列模式结果

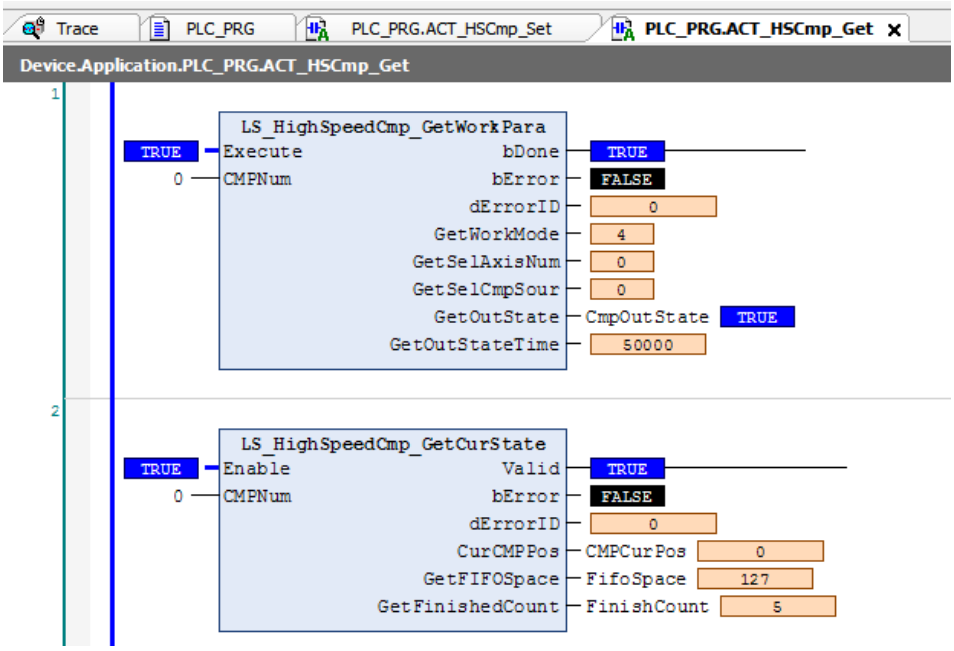


图 9.25 读取比较器状态模块

Trace | PLC_PRG

Device.Application.PLC_PRG

表达式	类型	值	准备
MC_Power_0	MC_Power		
MC_Power_1	MC_Power		

```
ACT_Power();
ACT_PulseControl();
ACT_Move();
ACT_HSCmp_Get();

IF NOT MC_Power_0.Status TRUE THEN
    RETURN;
END_IF

CASE CompareMode 100 OF//高速比较功能

1:
    bClearPara FALSE :=TRUE;
    ACT_HSCmp_Clear();      //清除比较器功能块
    CompareMode 100 :=10;       //切换此处CompareMode的值，可以切换高速比较模式
10:
    SetFifoPos_Exe TRUE :=TRUE;
    ACT_HSCmp_Set();        //比较模式：线性比较模式
    IF LS_HighSpeedCmp_SetWorkPara.bDone TRUE AND SetFifoPosDone TRUE THEN
        CompareMode 100 :=CompareMode 100 +5;
    END_IF
15:
    StartMove FALSE :=TRUE;       //启动运动
    CompareMode 100 :=CompareMode 100 +5;
20:
    //获取当前完成的比较点个数
    IF FinishCount 5 =5 THEN
        CompareMode 100 :=100;
    END_IF
100:
    bClearPara FALSE :=FALSE;
    StartMove FALSE :=FALSE;      //复位清除比较器状态变量
END_CASE RETURN
```

图 9.26　程序执行结果

运行结果：当 0 号轴运动经过位置 1000、5000、7000、10000、15000Pulse 时，CMP0 端口输出低电平，持续 500ms，如图 9.27、9.28 所示。

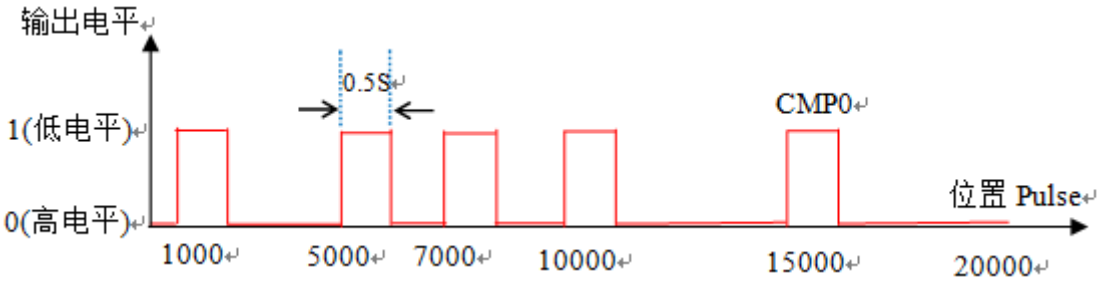


图 9.27　输出电平示意图

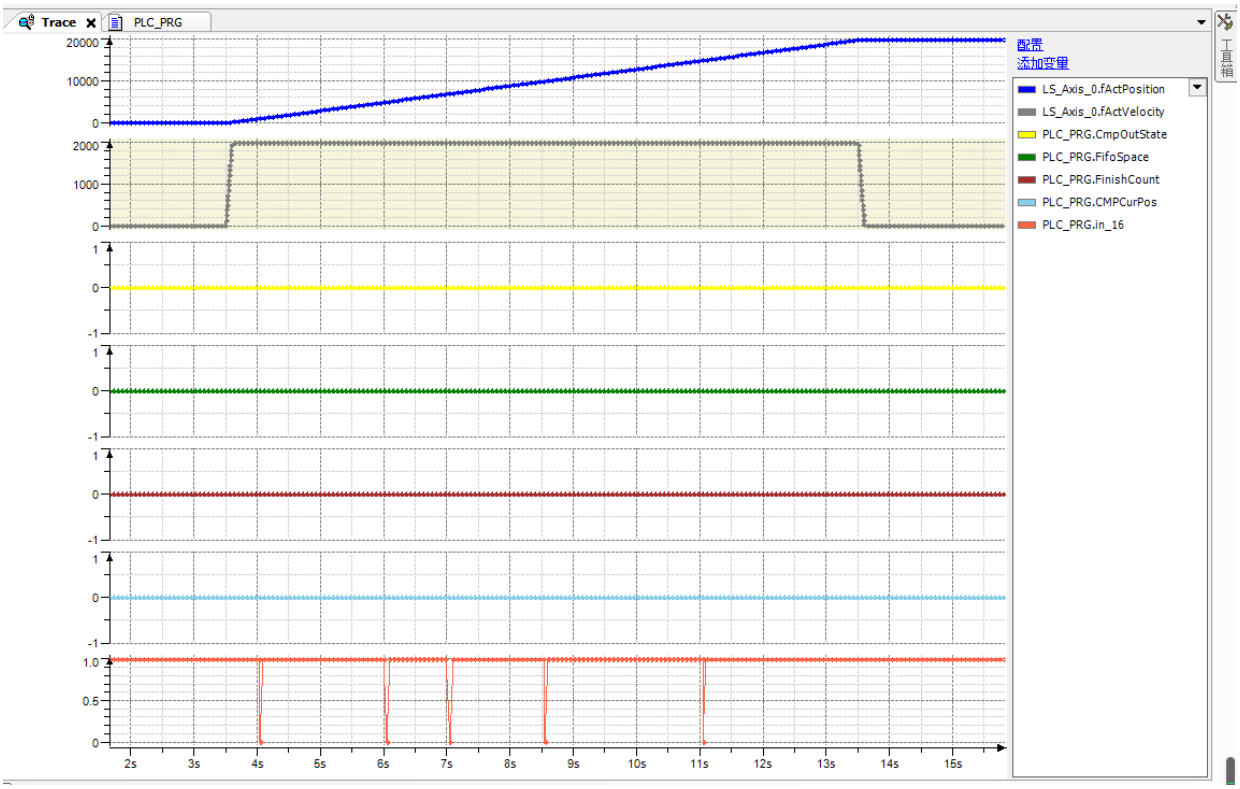


图 9.28　Trace 采集的各参数曲线

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**高速比较-队列模式-队列模式**”。

4. 模式 6 的例程

例程 9.6：模式 6-队列比较模式 FIFO-电平方式，关联 0 轴，设置 256 个比较点，并且通过设置比较点的电平参数，让相邻比较点的电平输出重复高低的变化，即每当经过设置的比较点，高速比较器 0（OUT16）端口电平重复高低高低的循环。

队列比较模式 FIFO-电平方式相对于**队列比较模式 FIFO-时间方式**的区别，在于一个是设置输出电平的高低，一个是让输出电平保持一定的时间后自动关闭。

队列比较模式-电平方式功能程序编写步骤同例程 9.5。在例程 9.5 的基础上，将大于模式模块 More 删除，增加队列模式模块 FIFO、读取比较器状态模块 Get_CMPState，并在该模块填写相应的参数，如图 9.29、9.30 所示。

然后在主程序中分别调用这些模块，编译、下载程序到控制器中运行，如图 9.31 所示。

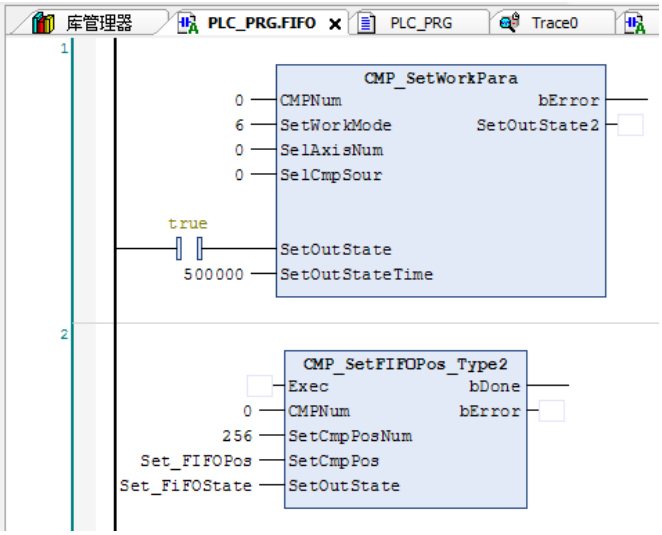


图 9.29　设置 FIFO 电平模式模块

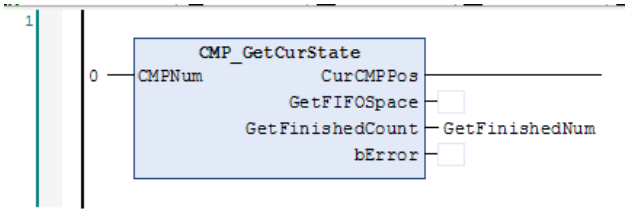


图 9.30　读取比较器状态模块

库管理器 | PLC_PRG.FIFO | PLC_PRG | Trace0 | PLC_PRG.ClearCMP | PLC_PRG.Get_CMPState

```
PROGRAM PLC_PRG
VAR
    MC_MoveRelative_0: MC_MoveRelative;
    StartMove: BOOL := false;
    MC_Power_0: MC_Power;

    IfClear: BOOL := FALSE;
    CompareMode: UINT:=0;//高速比较选择
    GetFinishedNum: UDINT;//获取比较点数
    Set_FIFOPos: ARRAY [0..256] OF DINT:=[2000,5000,6000,10000,15000,18000];//设置FIFO电平比较模式的比较点
```

```
Power();
readOut16 := OUT[16];

IF MC_Power_0.Status=TRUE THEN
    ;
ELSE
    RETURN;
END_IF

CASE CompareMode OF//高速比较功能
1:
    IfClear:=TRUE;
    ClearCMP();             //清除比较器功能块
    CompareMode:=2;
2:
    IfClear:=FALSE;
    ClearCMP();             //复位清除功能块
    Set_FIFOPos[0] := 1000; //设置第0个比较点位置
    CompareMode := 10;
10:
    FOR i := 1 TO 130 BY 1 DO //存储第1-130比较点的位置，相隔1000
        Set_FIFOPos[i] := Set_FIFOPos[i-1] + 1000;
        Set_FiFOState[i] := i MOD 2;
    END_FOR
    IF i = 131 THEN
        CompareMode := 20;
    END_IF
20:
    FOR j := 131 TO 256 BY 1 DO     //存储第131-256比较点的位置，相隔1000
        Set_FIFOPos[j] := Set_FIFOPos[j-1] - 1000;
```

```
    FOR j := 131 TO 256 BY 1 DO     //存储第131-256比较点的位置，相隔1000
        Set_FIFOPos[j] := Set_FIFOPos[j-1] - 1000;
        Set_FiFOState[J] := J MOD 2;
    END_FOR
    IF j = 257 THEN
        CompareMode := 30;
    END_IF
30:
    CMP_SetFIFOPos_Type2.Exec := TRUE; //设置比较位置
    IF CMP_SetFIFOPos_Type2.bDone THEN
        CMP_SetFIFOPos_Type2.Exec:=FALSE;
        CompareMode := 40;
    END_IF
    FIFO();                 //设置比较模式：FIFO电平比较模式
40:
    MC_MoveRelative_0.Distance := 132000;
    StartMove:=TRUE;        //启动运动
    IF GetFinishedNum =131 AND MC_MoveRelative_0.Done THEN
        CompareMode:=50;
    END_IF
50:
    StartMove:=FALSE;   //复位运动状态变量
    CompareMode:=60;
60:
    MC_MoveRelative_0.Distance := -132000;
    StartMove:=TRUE;        //启动运动
    IF GetFinishedNum =256 AND MC_MoveRelative_0.Done THEN
        CompareMode:=100;
    END_IF
100:
    StartMove:=FALSE;   //复位运动状态变量
END_CASE

RelativeMove();//单轴运动功能块
Get_CMPState();         //获取当前完成的比较点个数
ACT_PD606_CMD();
```

```
    IF i = 131 THEN
        CompareMode := 20;
    END_IF
20:
    FOR j := 131 TO 256 BY 1 DO     //存储第131-256比较点的位置，相隔1000
        Set_FIFOPos[j] := Set_FIFOPos[j-1] - 1000;
```

图 9.31　主程序

运行结果：当 0 号轴运动经过设置的 256 个比较点时，CMP0 端口电平呈现高低循环的变化，如图 9.32 所示，可以看到最终采集到了 256 个高速比较点。

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**高速比较输出 FIFO 设置电平模式**”。

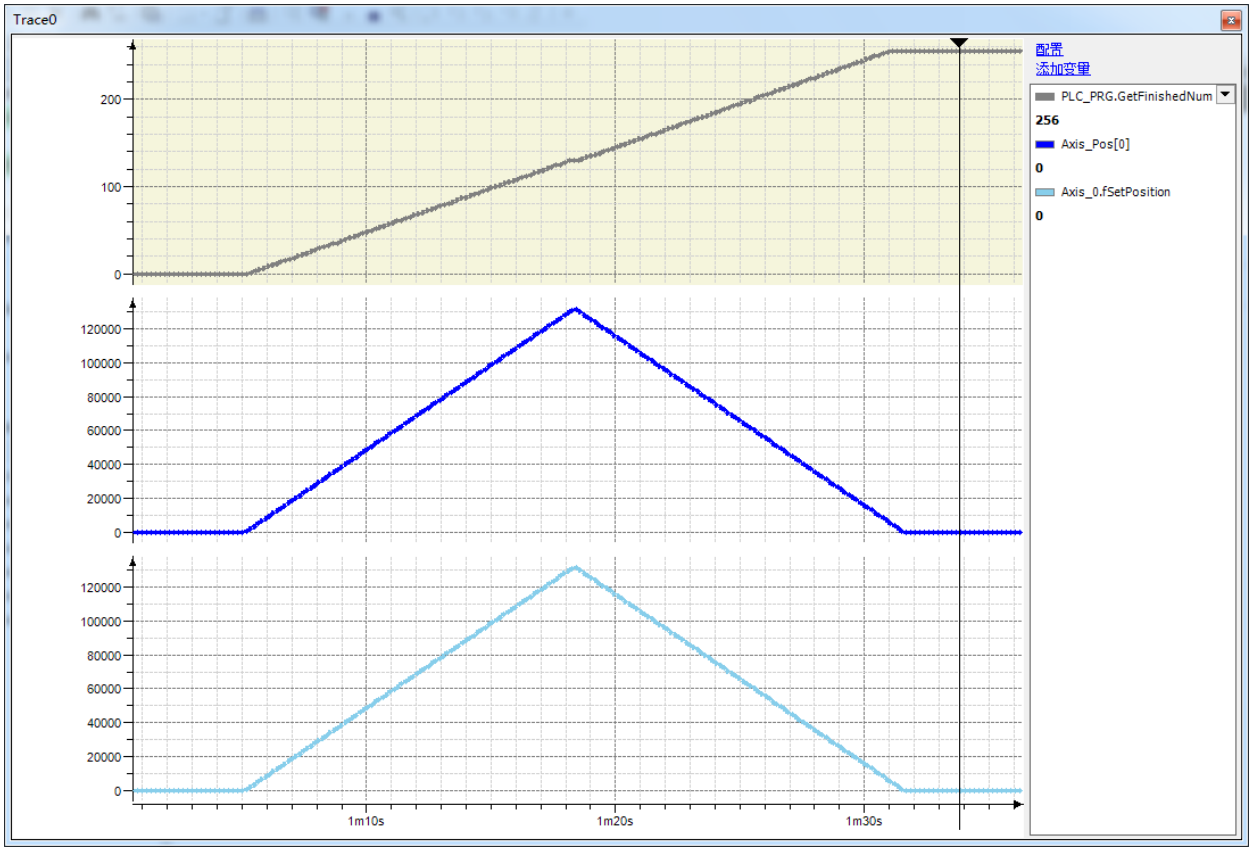

Trace0
配置
添加变量
PLC_PRG.GetFinishedNum
256
Axis_Pos[0]
0
Axis_0.fSetPosition
0
200
100
0
120000
100000
80000
60000
40000
20000
0
1m10s
1m20s
1m30s

图 9.32　程序执行结果

9.4 高速二维比较

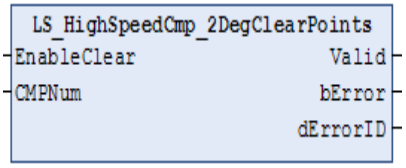
表 9.6　高速二维比较相关指令

指令名	功能说明
LS_HighSpeedCmp_2DegClearPoints	清除二维比较点
LS_HighSpeedCmp_2DegSetWorkPara	设置二维比较参数
LS_HighSpeedCmp_2DegGetWorkPara	获取二维比较参数
LS_HighSpeedCmp_2DegAddPoint	添加二维比较点
LS_HighSpeedCmp_2DegSetEnable	二维比较器使能
LS_HighSpeedCmp_2DegGetState	获取二维比较器状态
LS_HighSpeedCmp_2DegSetPwm	设置 PWM 二维比较参数
LS_HighSpeedCmp_2DegGetPwm	获取 PWM 二维比较参数

清除高速二维比较点　LS_HighSpeedCmp_2DegClearPoints

清除高速二维比较器的参数和所有状态值。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedCmp_2DegClearPoints	FB		LS_HighSpeedCmp_2DegClearPoints(EnableClear：= （参数）， CMPNum：= （参数）， Valid=>（参数）， bError=>（参数）， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
EnableClear	启动	BOOL	TRUE FALSE	FALSE	TRUE：持续清除比较器 FALSE：不执行指令
CMPNum	比较器通道	SINT	0	0	比较器通道：0
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回，0：表示无错误

说明：

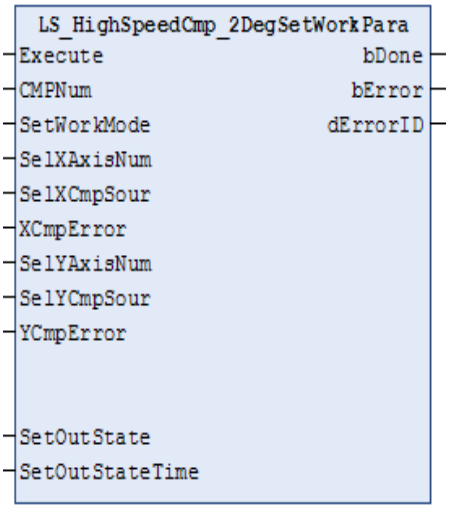
- 这个指令由“PMC_Controller”库实现。

• 设置高速二维比较参数和使用功能前，必须先调用这条指令，清除高速二维比较器的参数和所有状态值，才能够设置高速二维比较器的参数，与启动高速二维比较器。

设置高速二维比较参数 LS_HighSpeedCmp_2DegSetWorkPara

设置高速二维比较器参数。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedCmp_2DegSetWorkPara	FB		LS_HighSpeedCmp_2DegSetWorkPara(Execute：= （参数）， CMPNum：= （参数）， SetWorkMode：= （参数）， SelXAxisNum：= （参数）， SelXCmpSour：= （参数）， XCmpError：= （参数）， SelYAxisNum：= （参数）， SelYCmpSour：= （参数）， YCmpError：= （参数）， SetOutState：= （参数）， SetOutStateTime：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0	0	比较器通道：0
SetWorkMode	工作模式	UINT	0，1	0	设置比较器工作模式：0：进入误差带后触发；1：进入误差带单轴等于后再触发
SelXAxisNum	比较器 X 轴	UINT	0-3	0	选择需要比较的 X 轴号
SelXCmpSour	X 比较源	UINT	0，1	0	选择需要比较的数据源：0-指令位置；不支持 1-反馈位置
XCmpError	X 运行误差	LREAL	0，正数	0	X 轴的允许误差
SelYAxisNum	比较器 Y 轴	UINT	0-3	0	选择需要比较的 Y 轴号
SelYCmpSour	Y 比较源	UINT	0，1	0	选择需要比较的数据源：0-指令位置；不支持 1-反馈位置
YCmpError	Y 运行误差	LREAL	0，正数	0	Y 轴的允许误差
SetOutState	输出口状态	BOOL	TRUE FALSE	FALSE	设置输出端口的有效状态：False-条件成立输出 false，指示灯灭；True-条件成立输出 True，指示灯亮
SetOutStateTime	输出时间	UDINT	0-20000000	0	用于控制输出端口保持有效状态时间，单位 us：范围 0-20000000
VAR_OUTPUT					

bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	执行完成	BOOL	TRUE FALSE	FALSE	FALSE-没有错误； True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码

说明：

- 这个指令由“PMC_Controller”库实现。
- 高速二维比较的工作模式有两种：

模式 0：进入误差带后触发，即只要进入比较点（X，Y）的误差区域，便会触发比较输出，如图 9.32 所示的阴影区域；

模式 1：进入误差带单轴等于后再触发，即需要同时满足进入误差区域，以及其中一个轴等于比较点（X，Y）的位置，如图 9.32 所示的红色垂线和水平线。

- 高速二维比较的“XCmpError”和“YCmpError”两个参数，用来构建高速二维比较区域，分别代表了以 X 轴和 Y 轴为中心的矩形误差带，如图 9.33 中的浅黄色和浅绿色矩形，分别是比较点（X，Y）的 X 轴误差带和 Y 轴误差带。

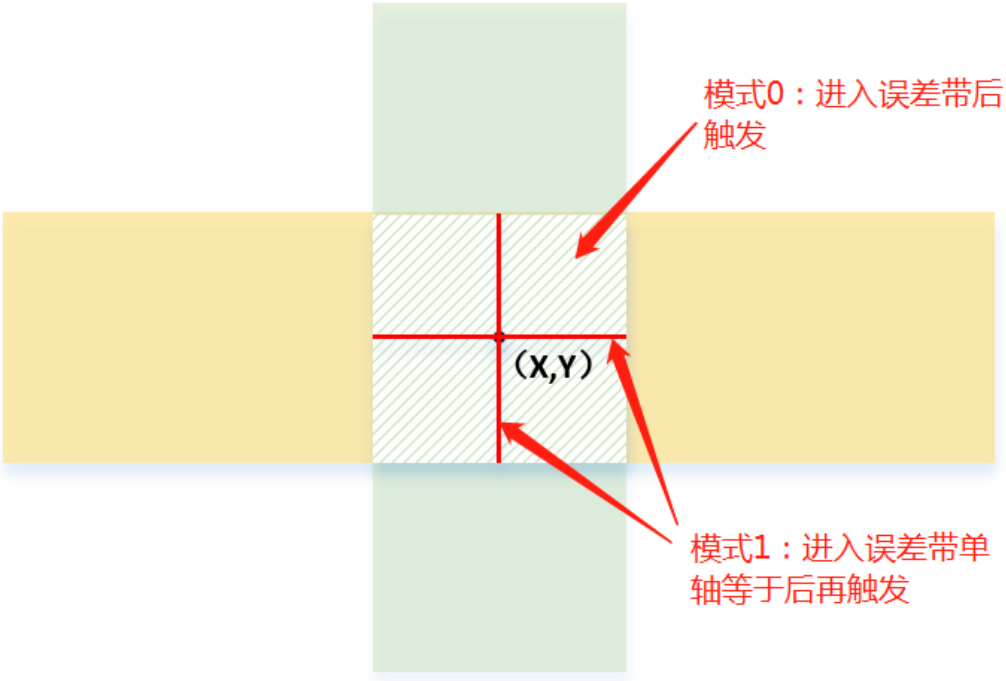


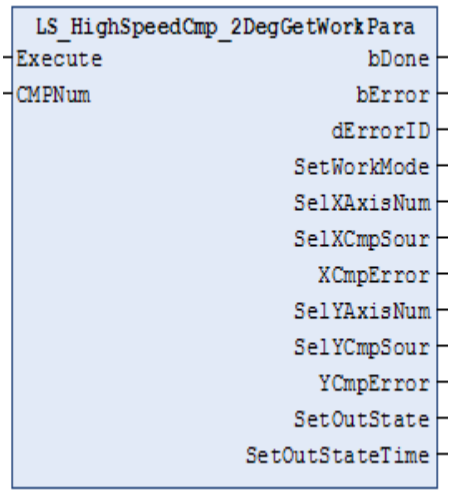
图 9.33　高速二维比较区域

- 注意：在使用高速二维比较功能时，如果 XY 轴不经过设置的高速比较区域，则不会触发高速比较输出。

获取高速二维比较参数 LS_HighSpeedCmp_2DegGetWorkPara

获取高速二维比较器参数。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedCmp_2DegGetWorkPara	FB		LS_HighSpeedCmp_2DegGetWorkPara(Execute：= （参数）， CMPNum：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）， SetWorkMode=>（参数）， SelXAxisNum=>（参数）， SelXCmpSour=>（参数）， XCmpError=>（参数）， SelYAxisNum=>（参数）， SelYCmpSour=>（参数）， YCmpError=>（参数）， SetOutState=>（参数）， SetOutStateTime=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0	0	比较器通道：0
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码
SetWorkMode	工作模式	UINT	0，1	0	设置比较器工作模式：0：进入误差带后触发；1：进入误差带单轴等于后再触发
SelXAxisNum	比较器 X 轴	UINT	0-3	0	选择需要比较的 X 轴号
SelXCmpSour	X 比较源	UINT	0，1	0	选择需要比较的数据源：0-指令位置；不支持 1-反馈位置
XCmpError	X 运行误差	LREAL	0，正数	0	X 轴的允许误差
SelYAxisNum	比较器 Y 轴	UINT	0-3	0	选择需要比较的 Y 轴号
SelYCmpSour	Y 比较源	UINT	0，1	0	选择需要比较的数据源：0-指令位置；不支持 1-反馈位置
YCmpError	Y 运行误差	LREAL	0，正数	0	Y 轴的允许误差
SetOutState	输出口状态	BOOL	TRUE FALSE	FALSE	设置输出端口的有效状态： False-条件成立输出 false，指示灯灭；True-条件成立输出 True，指示灯亮
SetOutStateTime	输出时间	UDINT	0-20000000	0	用于控制输出端口保持有效状态时间，单位 us：范围 0-20000000

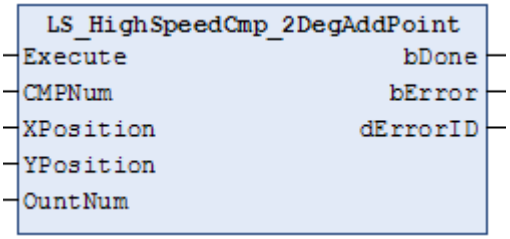
说明：

- 这个指令由“PMC_Controller”库实现。
- 指令用来获取高速二维比较器的比较模式、比较轴号、比较源、比较误差、输出口状态和时间等参数。
- 与“LS_HighSpeedCmp_2DegSetWorkPara”指令配合使用。

添加高速二维比较点　LS_HighSpeedCmp_2DegAddPoint

添加高速二维比较点。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedCmp_2DegAddPoint	FB		LS_HighSpeedCmp_2DegAddPoint(Execute:= （参数）， CMPNum:= （参数）， XPosition:= （参数）， YPosition:= （参数）， OuntNum:= （参数）， bDone=>（参数） ， bError=>（参数） ， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0	0	比较器通道：0
XPosition	目标减速度	LREAL	负数，0，正数	0	停止的减速度值
YPosition	目标减减速度	LREAL	负数，0，正数	0	停止的减减速度值
OuntNum	输出口	UINT	16-19	0	比较输出口，范围：16-19
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为 TRUE。
dErrorID	错误码	DINT	0，正数	0	错误码

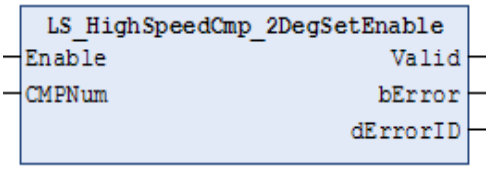
说明：

- 这个指令由“PMC_Controller”库实现。
- 指令用来添加二维高速比较点，“Execute”信号的每个上升沿添加一个比较点。

高速二维比较器使能　LS_HighSpeedCmp_2DegSetEnable

使能高速二维比较器，让高速二维比较器进入工作状态。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedCmp_2DegSetEnable	FB		LS_HighSpeedCmp_2DegSetEnable(Enable：= （参数）， CMPNum：= （参数）， Valid：= （参数）， bError=>（参数）， dErrorID=>（参数）);

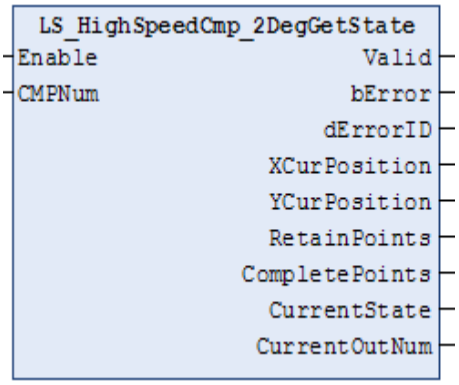
变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE FALSE	FALSE	TRUE：持续执行指令 FALSE：不执行指令
CMPNum	比较器通道	SINT	0	0	比较器通道：0
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	执行完成	BOOL	TRUE FALSE	FALSE	如果指令执行完成则为TRUE。
dErrorID	错误码	DINT	0，正数	0	错误码

获取高速二维比较器状态　LS_HighSpeedCmp_2DegGetState

获取高速二维比较器的比较状态。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedCmp_2DegGetState	FB		LS_HighSpeedCmp_2DegGetState(Enable：= （参数）， CMPNum：= （参数）， Valid=>（参数）， bError=>（参数）， dErrorID=>（参数）， XCurPosition=>（参数）， YCurPosition=>（参数）， RetainPoints=>（参数）， CompletePoints=>（参数）， CurrentState=>（参数）， CurrentOutNum=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE FALSE	FALSE	TRUE：持续执行指令 FALSE：不执行指令
CMPNum	比较器通道	SINT	0	0	比较器通道：0
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	执行完成	BOOL	TRUE FALSE	FALSE	FALSE-没有错误； True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误
XCurPosition	X 轴当前比较位置	LREAL	负数，0，正数	0	X 轴当前比较位置
YCurPosition	Y 轴当前比较位置	LREAL	负数，0，正数	0	Y 轴当前比较位置
RetainPoints	剩余比较个数	DINT	0，正数	0	剩余比较个数
CompletePoints	已比较点数	DINT	0，正数	0	已比较点数
CurrentState	当前比较状态	SINT		0	当前比较状态
CurrentOutNum	当前比较输出口	INT		0	当前比较输出口

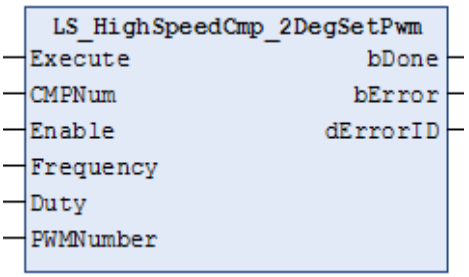
说明：

- 这个指令由“PMC_Controller”库实现。
- 指令可以获取当前高速二维比较器的 X 轴、Y 轴当前比较位置、剩余比较个数、已比较个数、当前比较状态和比较熟出口编号。
- 将指令的“Enable”信号置为 TRUE，将持续获取二维高速比较器的状态。

设置 PWM 二维比较参数 LS_HighSpeedCmp_2DegSetPwm

中断轴正在进行的运动，对轴进行减速停止。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedCmp_2DegSetPwm	FB		LS_HighSpeedCmp_2DegSetPwm(Execute:= （参数）， CMPNum:= （参数）， Enable:= （参数）， Frequency:= （参数）， Duty:= （参数）， PWMNumber:= （参数）， bDone=>（参数） ， bError=>（参数） ， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0	0	比较器通道：0
Enable	PWM 使能	BOOL	TRUE FALSE	FALSE	TRUE：持续执行指令 FALSE：不执行指令
Frequency	PWM 脉冲频率	LREAL	0，正数	0	输出 PWM 脉冲频率
Duty	占空比	LREAL	0-1	0	输出占空比
PWMNumber	脉冲个数	DINT	正数	0	输出脉冲个数
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	报错	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码

说明：

- 这个指令由“PMC_Controller”库实现。
- 配置高速二维比较器参数：

  cmp_mode： 0--进入误差带后触发，1--进入误差带单轴等于后再触发

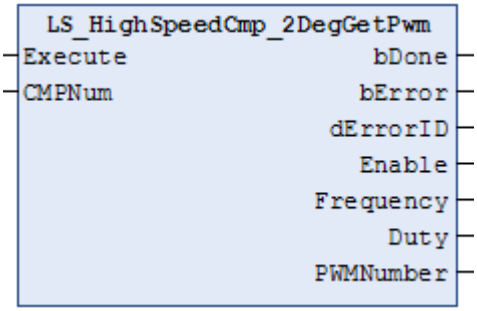
  cmp_logic： 0--有效电平为低电平，1--有效电平为高电平

  time： 脉冲宽度，单位：us，取值范围：1us~20s

获取 PWM 二维比较参数 LS_HighSpeedCmp_2DegGetPwm

中断轴正在进行的运动，对轴进行减速停止。

指令外观

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedCmp_2DegGetPwm	FB		LS_HighSpeedCmp_2DegGetPwm(Execute:= （参数）， CMPNum:= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）， Enable=>（参数）， Frequency=>（参数）， Duty=>（参数）， PWMNumber=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
CMPNum	比较器通道	SINT	0	0	比较器通道：0
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	执行完成	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码
Enable	PWM 使能	BOOL	TRUE FALSE	FALSE	TRUE：持续执行指令 FALSE：不执行指令
Frequency	PWM 脉冲频率	LREAL	0，正数	0	输出 PWM 脉冲频率
Duty	占空比	LREAL	0-1	0	输出占空比
PWMNumber	脉冲个数	DINT	正数	0	输出脉冲个数

说明：与 LS_HighSpeedCmp_2DegSetPwm 相同。

9.5 位置锁存

9.5.1 原点位置锁存指令

SMC600 系列运动控制器支持原点锁存功能，该功能可以实现设备在触发原点信号时将当前位置锁存，精确回原点运动。相关指令如表 9.7 所示。

表 9.7 位置锁存相关指令

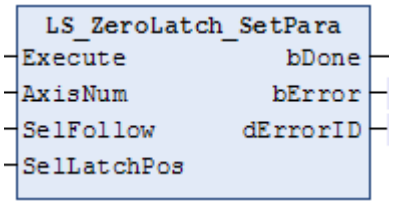
指令名	功能说明
LS_ZeroLatch_SetPara	设置原点锁存参数
LS_ZeroLatch_GetState	读取原点锁存状态
LS_HighSpeedLatch_Clear	清除高速比较锁存
LS_HighSpeedLatch_SetSource	设置高速锁存数据源
EzLatch_SetPara	设置 EZ 锁存参数
EzLatch_GetState	读取 EZ 锁存状态

需要注意的是，每个轴由各自的原点信号触发；锁存完成后，锁存的位置保存于 SMC606 库中 GVL_mPC_M606_User 全局变量列表的 ZeroLatch_Axis_Pos 数组中，如 ZeroLatch_Axis_Pos[0]表示轴 0 的原点锁存位置，其他轴以此类推。

设置原点锁存参数 LS_ZeroLatch_SetPara

设置原点锁存参数。

指令外观：

指令	FB/ FUN	图形模块	结构文本
LS_ZeroLatch_SetPara	FUN		LS_ZeroLatch_SetPara(Execute：= （参数）， AxisNum：= （参数）， SelFollow：= （参数）， SelLatchPos：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
AxisNum	轴号	UINT	0-3	0	轴号：0—3
SelFollow	触发沿	UINT	0，1	0	选择高速锁存触发沿：0-下降沿锁存；1-上升沿锁存。原点输入端口状态从 FALSE 到 TRUE 为上升沿
SelLatchPos	锁存位置类型	UINT	0，1	0	选择高速锁存位置类型：0-锁存指令位置；1-锁存反馈位置
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码

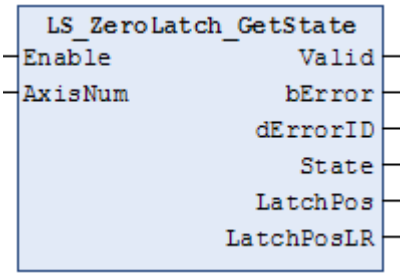
说明：

- 这个指令由“PMC_Controller”库实现。
- 原点锁存功能，用于在轴运动过程中，遇到原点信号，锁存此刻的轴位置。该功能可以实现设备在触发原点信号时将当前位置锁存，精确回原点运动。
- 指令用于设置原点锁存参数，设定轴号、触发沿和锁存位置类型。
- 锁存参数设置完成后，触发指定轴号的原点信号，便能触发该轴的原点位置锁存。锁存完成信号与锁存的数据，可以在指令“LS_ZeroLatch_GetState”中获取。

读取原点锁存状态　LS_ZeroLatch_GetState

获取原点锁存状态以及数值。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_ZeroLatch_GetState	FUN		LS_ZeroLatch_GetState(Enable：= （参数）， AxisNum：= （参数）， Valid=>（参数）， bError=>（参数）， dErrorID=>（参数）， State=>（参数）， LatchPos=>（参数）， LatchPosLR=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE FALSE	FALSE	TRUE：持续执行指令 FALSE：不执行指令
AxisNum	轴号	UINT	0-3	0	轴号：0—3
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码
State	锁存状态	BOOL	TRUE FALSE	FALSE	FALSE-锁存没有完成 True-锁存已经完成
LatchPos	锁存位置	DINT	负数，0 正数	0	获取锁存位置，当锁存完成标志位 true 时，该位置有效
LatchPosLR	unit 锁存位置	LREAL	负数，0 正数	0	包含脉冲当量的位置信息，以 unit 为单位

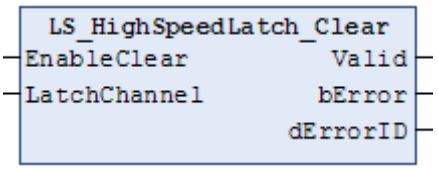
说明：

- 这个指令由“PMC_Controller”库实现。
- 用于获取原点锁存状态、锁存位置。
- 当 Enable 信号持续为 TRUE，则持续获取原点锁存信息。
- 与“LS_ZeroLatch_SetPara”指令配合进行使用。

清除高速比较锁存　LS_HighSpeedLatch_Clear

清除发生过的锁存及状态值。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedLatch_Clear	FUN		LS_HighSpeedLatch_Clear(EnableClear：= （参数）， LatchChannel：= （参数）， Valid=>（参数）， bError=>（参数）， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
EnableClear	清除锁存	BOOL	TRUE FALSE	FALSE	True-清除锁存 FALSE-保持当前状态
LatchChannel	通道号	UINT	0-3	0	通道号：0-3
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码

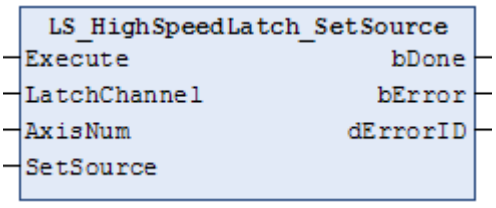
说明：

- 这个指令由“PMC_Controller”库实现。
- 清除已经发生过的锁存及状态标记，使其恢复到初始状态。
- 调用高速锁存，必须先调用该指令，以避免锁存器的一些残留数据。

设置高速锁存数据源　LS_HighSpeedLatch_SetSource

设置高速锁存数据源。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedLatch_SetSource	FUN		LS_HighSpeedLatch_SetSource(Execute：= （参数）， LatchChannel：= （参数）， AxisNum：= （参数）， SetSource：= （参数）， bDone=>（参数）， bError=>（参数）， dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
LatchChannel	通道号	UINT	0-3	0	通道号：0-3
AxisNum	轴号	UINT	0-3	0	轴号：0—3
SetSource	锁存源	UINT	0，1	0	设置锁存源：0-指令位置，1-编码器反馈位置。编码器轴设置为 1
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成；FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码

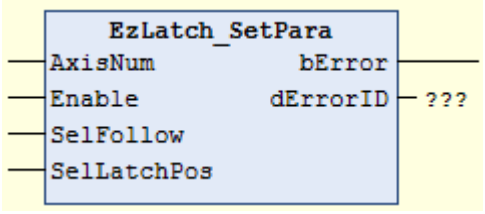
说明：

- 这个指令由“PMC_Controller”库实现。
- 设置高速锁存数据源，当 SetSource 为 0 时，则锁存源是指令位置，即当锁存触发时，锁存器里面记录的是指令位置；当 SetSource 为 1 时，则锁存源是编码器反馈位置，即当锁存触发时，锁存器里面记录的是编码器反馈位置。
- 指令只需要调用一个周期，触发参数的执行。

设置 EZ 锁存参数　EzLatch_SetPara

设置 EZ 锁存参数。

指令外观：

指令	FB/FUN	图形模块	结构文本
EzLatch_SetPara	FUN		EzLatch_SetPara(AxisNum:= , Enable:= , SelFollow:= , SelLatchPos:= , bError=> , dErrorID=>);

变量：

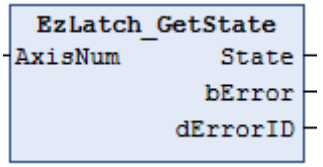
VAR_INPUT	名称	类型	有效范围	初始值	描述
AxisNum	轴号	UINT			轴号：0~7
Enable	使能	BOOT	0-3	0	通道号：0-3
SelFollow		UINT			选择高速锁存触发沿
SelLatchPos		UINT			选择高速锁存位置类型

VAR_OUTPUT					
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码

读取 EZ 锁存状态　EzLatch_GetState

读取 EZ 锁存状态。

指令外观：

指令	FB/FUN	图形模块	结构文本
EzLatch_GetPara	FUN		EzLatch_GetState(AxisNum:= , State=> , bError=> , dErrorID=>);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
AxisNum	轴号	UINT			轴号：0~7
VAR_OUTPUT					
State		BOOL		FALSE	True-锁存已经完成
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码

9.5.2　高速位置锁存指令

雷赛 PMC600 系列运动控制器支持高速位置单次锁存和连续功能，可实现精确定位功能。高速锁存的四个通道，分别由高速输入端口 In20、In21、In22、In23 触发，可自由配置通道和轴号的关联。

在锁存信号有效后可锁存一次，再次锁存需复位锁存标志。

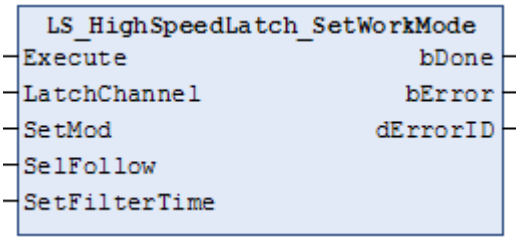
表 9.8　单次高速锁存相关指令：

名称	功能
HighSpeedLatch_SetWorkMode	设置高速锁存参数
HighSpeedLatch_SetSource	设置高速锁存数据源
HighSpeedLatch_ReadState	读取高速锁存状态
HighSpeedLatch_ReadVal	读取当前锁存值
HighSpeedLatch_FIFOReadVal	读取当前锁存值，多周期
HighSpeedLatch_FIFOReadNu	读取当前锁存的数据个

设置高速锁存模式 LS_HighSpeedLatch_SetWorkMode

设置高速锁存参数值。

指令外观

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedLatch_SetWorkMode	FUN		LS_HighSpeedLatch_SetWorkMode(Execute:= （参数）, LatchChannel:= （参数）, SetMod:= （参数）, SelFollow:= （参数）, SetFilterTime:= （参数）, bDone=>（参数） , bError=>（参数） , dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
LatchChannel	通道号	UINT	0-3	0	通道号：0-3
SetMod	锁存模式	UINT	0，1	0	设置锁存模式：0-单次锁存； 1-连续锁存（系统底层 FIFO 数量为 256）。参数值需要保持
SelFollow	锁存 触发沿	UINT	0-2	0	选择高速锁存触发沿：0-下降沿锁存；1-上升沿锁存；2-任意沿锁存（上升沿或下降沿）。输入端口状态从 FALSE 到 TRUE 为上升沿
SetFilterTime	-	-	-	-	参数保留
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码

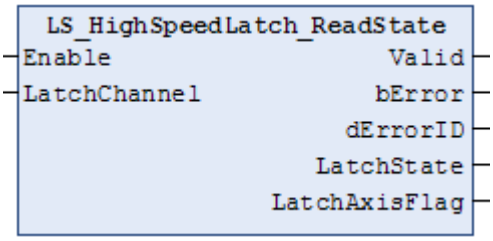
说明：

- 这个指令由“PMC_Controller”库实现。
- 设置高速锁存的模式、触发沿等参数。
- 指令只需要调用一个周期，触发参数的执行。
- 有两种锁存模式，SetMod 为 0 是单次锁存，即设置完锁存器之后，锁存器只能够触发一次，且第一次有效，触发多次是不会产生更多的锁存数值的；SetMod 为 1 时是连续锁存，即设置完锁存器之后，锁存器可以连续被触发，且最多为 256 个。

读取高速锁存状态 LS_HighSpeedLatch_ReadState

读取单次锁存的锁存状态。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedLatch_ReadState	FUN		LS_HighSpeedLatch_ReadState(Enable:= （参数）， LatchChannel:= （参数）， Valid=>（参数） ， bError=>（参数） ， dErrorID=>（参数）， LatchState=>（参数） ， LatchAxisFlag=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE FALSE	FALSE	TRUE：持续执行指令 FALSE：不执行指令
LatchChannel	通道号	UINT	0-3	0	通道号：0-3
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码
LatchState	锁存完成状态	BOOL	TRUE FALSE	FALSE	FALSE-锁存没有完成 True-锁存已经完成
LatchAxisFlag	锁存轴	UDINT	0-3	0	已经锁存有数据的轴标记

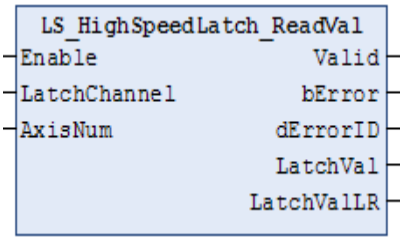
说明：

- 这个指令由“PMC_Controller”库实现。
- 用来读取单次锁存的锁存状态，不适用于连续锁存。
- 锁存完成后，锁存状态 LatchState 将变成 TRUE，且参数 LatchAxisFlag 也将显示当前锁存器锁存的轴号。

读取当前锁存值 LS_HighSpeedLatch_ReadVal

读取单次锁存的当前锁存值。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedLatch_ReadVal	FUN		LS_HighSpeedLatch_ReadVal(Enable：= （参数）， LatchChannel：= （参数）， AxisNum：= （参数）， Valid=>（参数）， bError=>（参数）， dErrorID=>（参数）， LatchVal=>（参数）， LatchValLR=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE FALSE	FALSE	TRUE：持续执行指令 FALSE：不执行指令
LatchChannel	通道号	UINT	0-3	0	通道号：0-3
AxisNum	轴号	UINT	0-3	0	轴号：0—3
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码
LatchVal	锁存值	DINT	负数，0 正数	0	返回锁存值（脉冲数）
LatchValLR	锁存值	LREAL	负数，0 正数	0	返回锁存值（除以脉冲当量后的值）

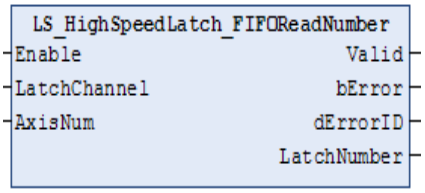
说明：

- 这个指令由“PMC_Controller”库实现。
- 读取单次锁存的当前锁存值，不适用于连续锁存。
- 锁存完成后，将输出获取的锁存数值。

读取 FIFO 模式锁存个数　LS_HighSpeedLatch_FIFOReadNumber

读取连续锁存当前的数据个数。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedLatch_FIFOReadNumber	FUN		LS_HighSpeedLatch_FIFOReadNumber(Enable:= （参数）， LatchChannel:= （参数）， AxisNum:= （参数）， Valid=>（参数） ， bError=>（参数） ， dErrorID=>（参数）， LatchNumber=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE FALSE	FALSE	TRUE：持续执行指令 FALSE：不执行指令
LatchChannel	通道号	UINT	0-3	0	通道号：0-3
AxisNum	轴号	UINT	0-3	0	轴号：0—3
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码
LatchNumber	锁存个数	UDINT	负数，0 正数	0	返回当前已经锁存的数据个数

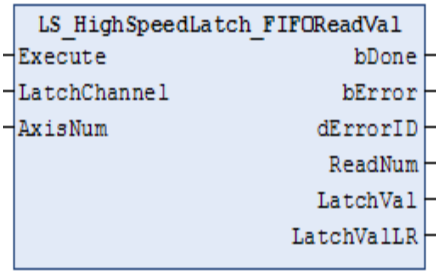
说明：

- 这个指令由“PMC_Controller”库实现。
- 读取连续锁存当前的数据个数，可以通过指令实时获取当前连续锁存所运行的进度。
- 只要指令的 Enable 为 TRUE，将实时获取当前的锁存状态，一般在使用连续锁存时，可以持续调用指令获取数据。

读取 FIFO 模式锁存值 LS_HighSpeedLatch_FIFOReadVal

读取连续锁存当前的锁存值。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_HighSpeedLatch_FIFOReadVal	FUN		LS_HighSpeedLatch_FIFOReadVal(Execute:= （参数）， LatchChannel:= （参数）， AxisNum:= （参数）， bDone=>（参数） ， bError=>（参数） ， dErrorID=>（参数）， ReadNum=>（参数） ， LatchVal=>（参数） ， LatchValLR=>（参数）);

变量

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
LatchChannel	通道号	UINT	0-3	0	通道号：0-3
AxisNum	轴号	UINT	0-3	0	轴号：0—3
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误； True-输入参数越界，或者模式不正确，或者固件版本太低不支持该功能块
dErrorID	错误码	DINT	0，正数	0	错误码
ReadNum	执行中	UDINT	0，正数	0	返回本次读取的锁存值个数
LatchVal	锁存值	ARRAY[0..31] OF DINT	负数，0 正数	0	返回锁存值
LatchValLR	锁存值	ARRAY[0..31] OF LREAL	负数，0 正数	0	返回锁存值（除以脉冲当量后的值）

说明：

- 这个指令由“PMC_Controller”库实现。
- 读取连续锁存当前的锁存值。
- 当锁存缓存里有超过 32 个数值的时候，每触发一次 Execute 的上升沿，将从缓存里读取 32 个锁存值到 LatchVal 和 LatchValLR 之中，且保持先进先出的队列原则，即先锁存的数值将优先获取。
- 如果缓存里的数值已经取完，则继续触发锁存值了，将获取到重复的数值。

9.5.3 原点锁存精确回零方法及例程

原点锁存精确回零使用方法：

1） 清除原点锁存状态：用 LS_ZeroLatch_SetPara 指令清除发生过的锁存及状态值（不启用使能）；

2） 配置原点锁存的参数：用 LS_ZeroLatch_SetPara 指令配置高速锁存参数，包括锁存触发沿、锁存位置类型等；

3） 启动运动：调用 MC_MoveVelocity 启动恒速运动；

4） 获取原点锁存的当前状态：用 LS_ZeroLatch_GetState 指令读取原点锁存的当前状态，包括出错状态、是否锁存完成等，锁存完成后停止运动并用 HighSpeedLatch_Axis_Pos 读取对应轴的原点锁存位置；

5） 恒速运动停止完成后使用绝对模式的定长运动运动到锁存的原点位置；

6） 回到原点锁存位置后，指令脉冲计数器清零。

例程 9.7：设置轴 0 执行速度为 1000Pulse/s、加减速度为 2000Pulse/s^2 恒速运动，原点信号有效时锁存当前位置值并停止运动，完全停止后执行绝对模式的定长运动回到锁存的位置，最后将指令计数器清零。

1) 新建工程并命名 ZeroLatch，编程语言为结构化文本 ST；

2) 右击 PLC_PRG 主程序，选择“添加对象”-“动作”，命名 PD606_IO，负责脉冲方向模块和硬件 IO 处理模块，编程语言为梯形图 LD，在设备栏双击 PD606_IO 进入程序编辑区，添加 PD606_IO_Cmd 模块并在指令中填上对应的轴号，如图 9.34 所示；

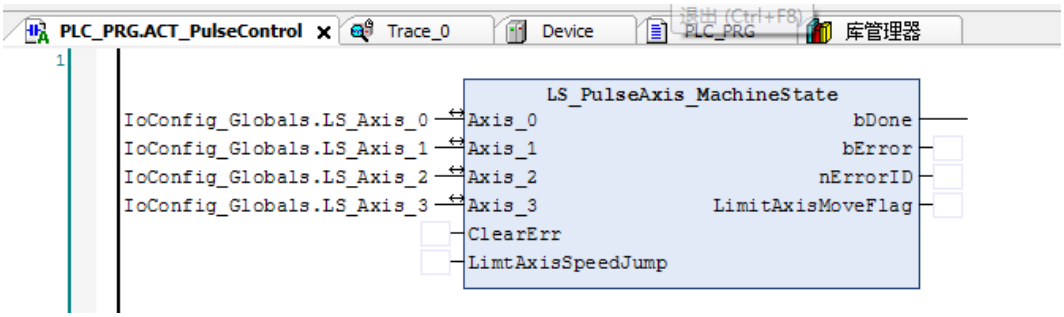


图 9.34 脉冲模块 PD606_IO

3) 按照步骤 2 的方法依次添加上电模块 Power、恒速运动模块 ACT_Vmove、原点锁存模块 ACT_ZeroLatch，并填写相应的参数，如图 9.35~9.37 所示。

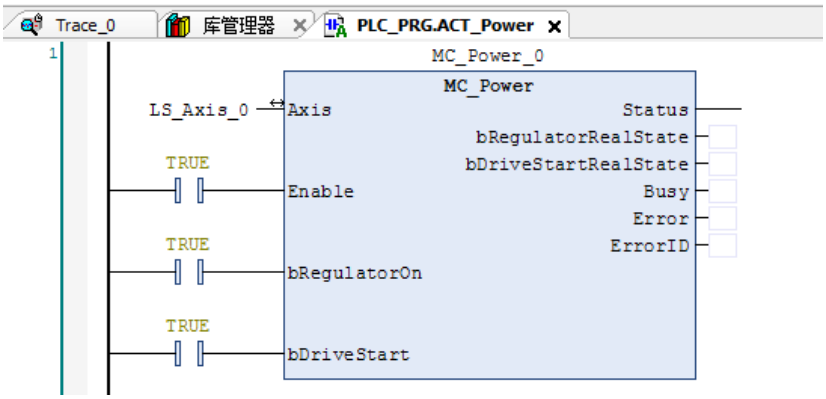


图 9.35　上电模块 Power

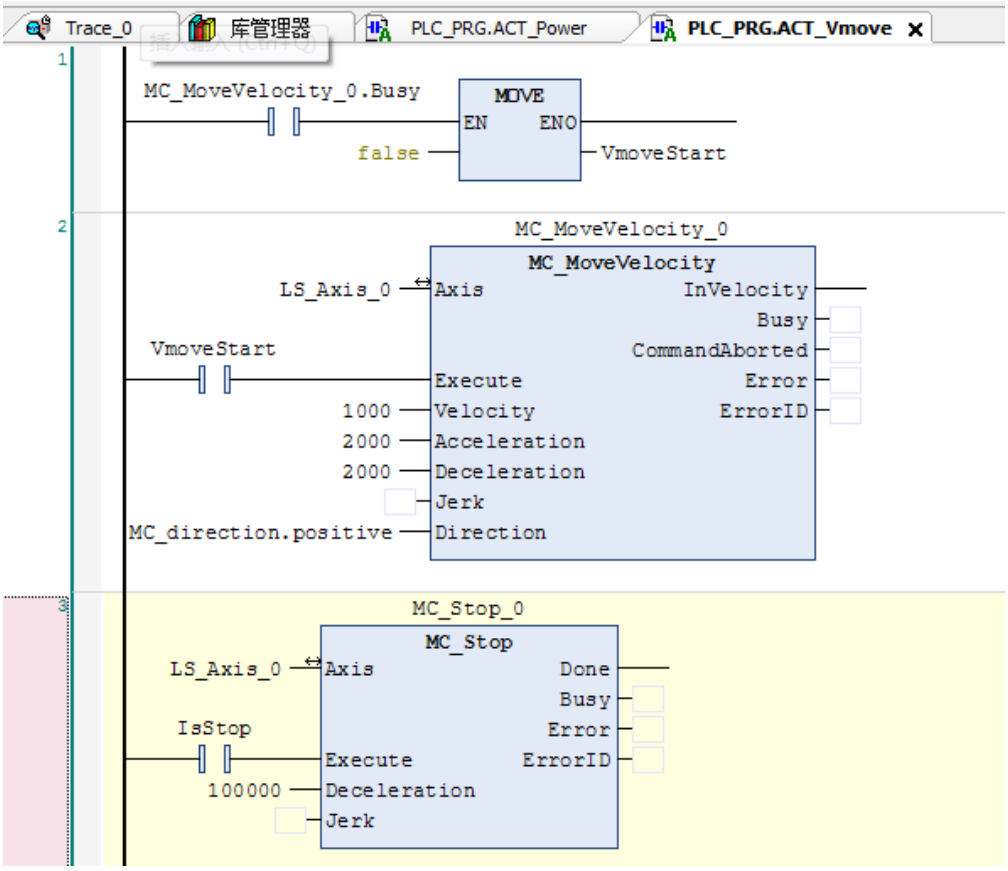


图 9.36　恒速运动模块 ACT_Vmove

注意：在原点锁存模块中当锁存完成时停止轴运动，在复位停止使能时须再次调用停止模块，再执行绝对模式的定长运动，到位后将指令位置计数器清零。

4) 在主程序中分别调用这些模块，编译、下载程序到控制器执行，如图 9.38 所示。

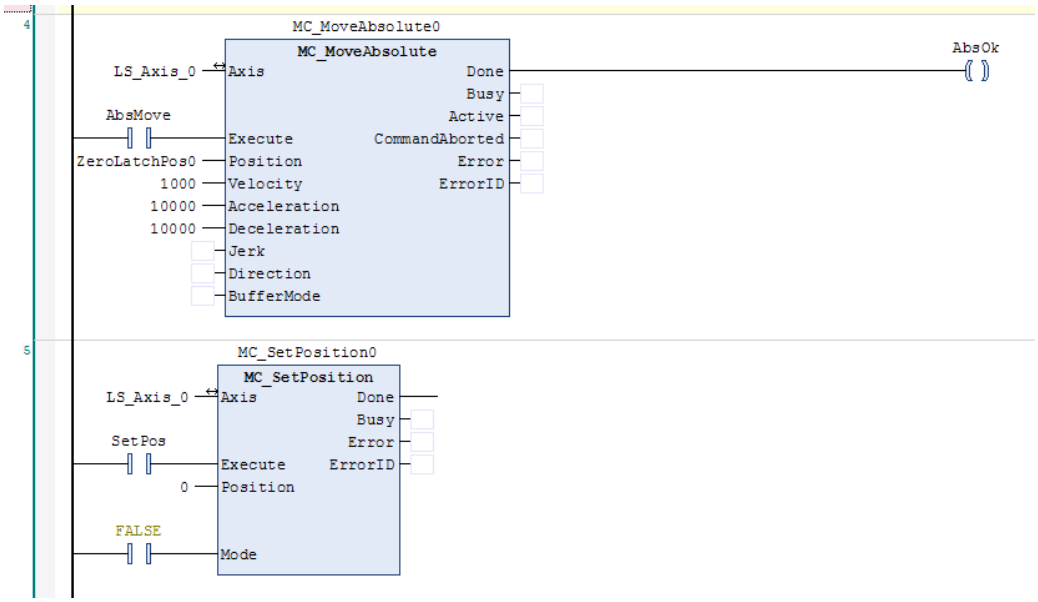


图 9.37 原点锁存模块 ACT_ZeroLatch

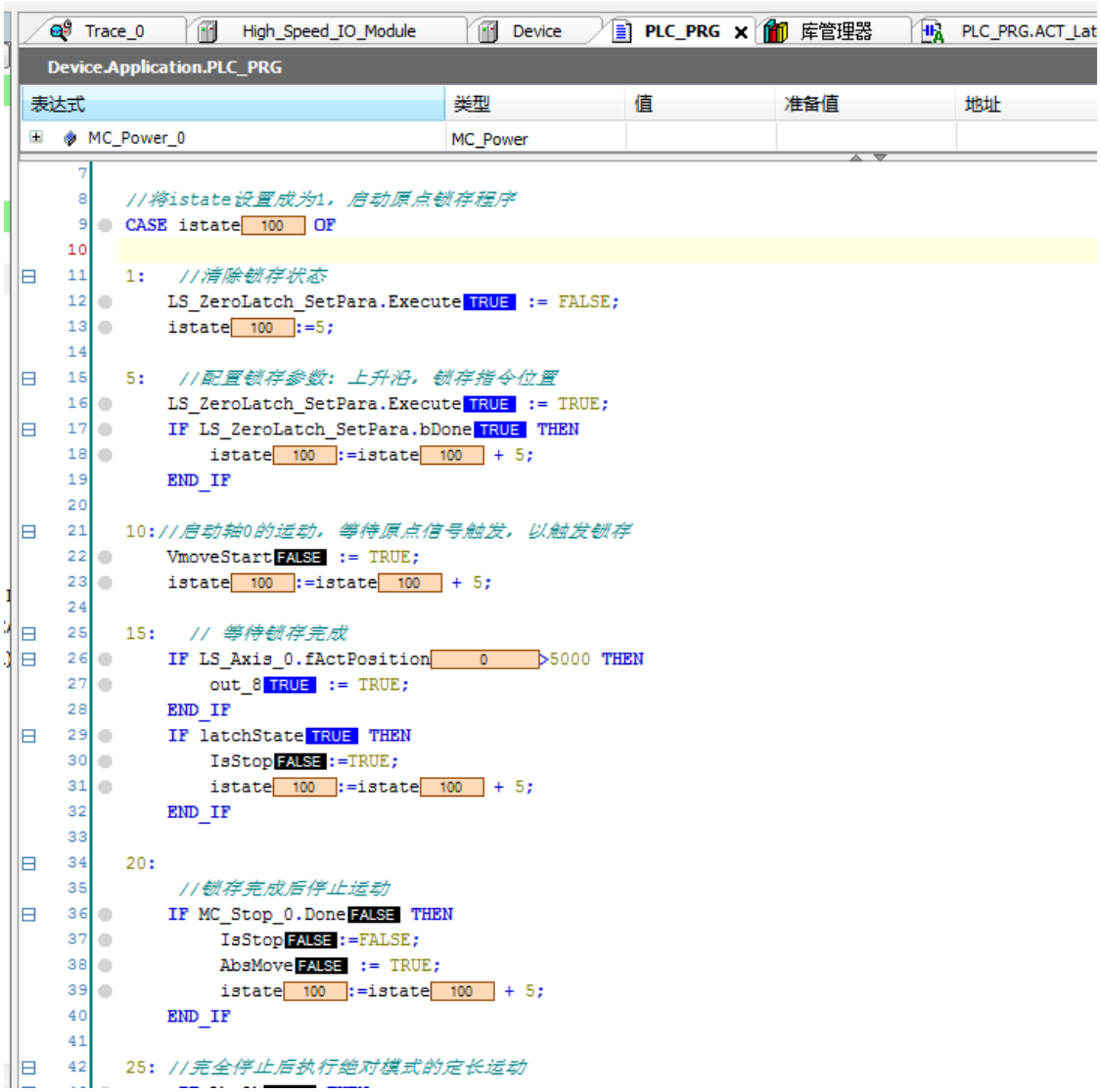


图 9.38 原点锁存主程序及运行结果

运行结果：当轴 0 的原点信号有效时，程序锁存轴 0 的当前位置并赋值给变量 ZeroLatchPos0，停止运动后执行绝对模式的定长运动到锁存位置并清零指令计数器，结果如图 9.39 所示。

从图 9.38 中可看出：当轴 0 原点信号有效，速度开始降低也就是减速停止；待完全停止后再反向运动到锁存位置，到达锁存位置后将指令计数器清零。

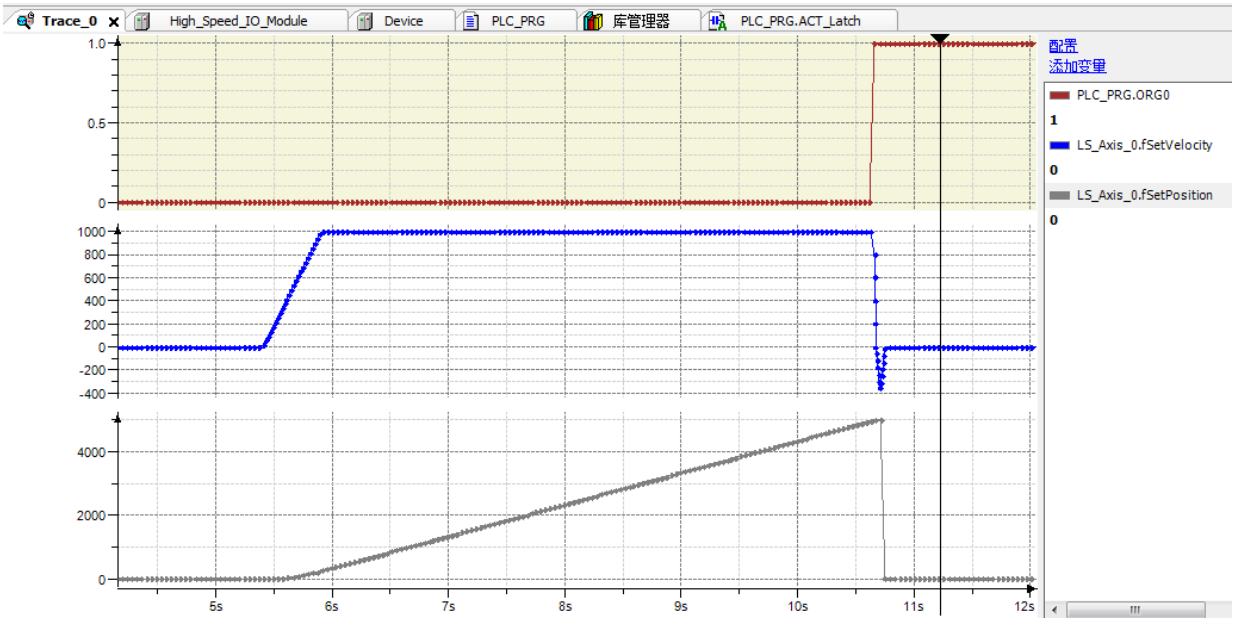


图 9.39　轴 0 位置和速度曲线

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**锁存功能-ZeroLatch**”。

9.5.4　EZ 锁存精确回零方法及例程

雷赛 SMC600 系列运动控制器支持 EZ 锁存功能，该功能可以在设备回原点过程中如果编码器 EZ 信号有效则将当前位置锁存，实现精确回原点运动。

需要注意的是，每个轴由各自的 EZ 信号触发；锁存完成后，锁存的位置保存于 SMC606 库中 GVL_mPC_M606_User 全局变量列表的 EzLatch_Axis_Pos 数组中，如 EzLatch_Axis_Pos[0]表示轴 0 的 EZ 锁存位置，其他轴以此类推。另外在用到此功能时必须外接驱动器 EZ 信号，未连接外部 EZ 信号时切不可在程序中读取 EZ 信号，它处于一个不稳定未知的状态。

EZ 锁存精确回零方法：

1） 清除 EZ 锁存状态：用 EzLatch_SetPara 指令清除发生过的锁存及状态值（不启用使能）；

2） 配置 EZ 锁存的参数：用 EzLatch_SetPara 指令配置 EZ 锁存参数，包括锁存触发沿、锁存位置类型等；

3） 启动运动：调用 MC_MoveVelocity 启动恒速运动；

4） 获取 EZ 锁存的当前状态：用 EzLatch_GetState 指令读取 EZ 锁存的当前状态，包括出错状态、是否锁存完成等，锁存完成后停止运动并用 EzLatch_Axis_Pos 读取对应轴的 EZ 锁存位置；

5） 恒速运动停止完成后使用绝对模式的定长运动运动到锁存的 EZ 位置；

6） 回到原点锁存位置后，指令脉冲计数器清零。

由此可见，EZ 锁存回零和原点锁存回零是类似的，只是换了个指令而已，所用方法是一样的，具体可参见原点锁存精确回零的方式，下面仅介绍如何实现 EZ 锁存。

例程 9.10：设置轴 0 执行速度为 1000Pulse/s、加减速度为 2000Pulse/s^2 恒速运动，原点信号有效时锁存当前位置值并停止运动，完全停止后执行绝对模式的定长运动回到锁存的位置，最后将指令计数器清零：

1) 新建工程并命名 EZLatch，编程语言为结构化文本 ST。

2) 右击 PLC_PRG 主程序，选择“添加对象”-“动作”，命名 ACT_PD606_IO_Cmd，负责脉冲方向模块和硬件 IO 处理模块，编程语言为梯形图 LD，在设备栏双击 ACT_PD606_IO_Cmd 进入程序编辑区，添加 PD606_IO_Cmd 模块并在指令中填上对应的轴号。

3) 按照步骤 2 的方法依次添加上电模块 Power、相对运动模块 ACT_RelativeMove、EZ 锁存模块 ACT_EzLatch，并填写相应的参数，如图 9.40~9.42 所示。

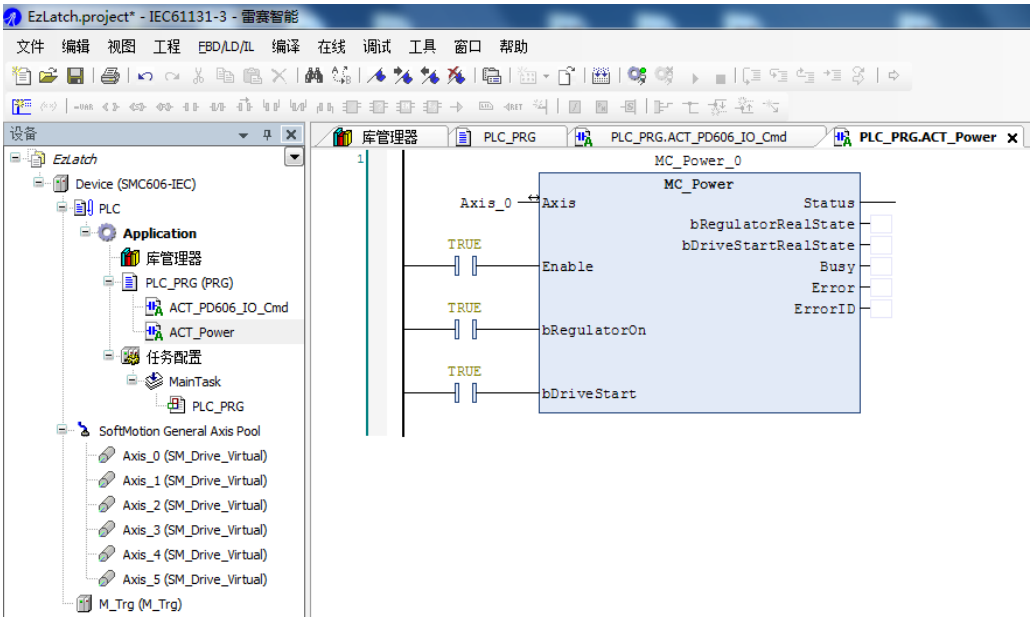


图 9.40 上电模块 Power

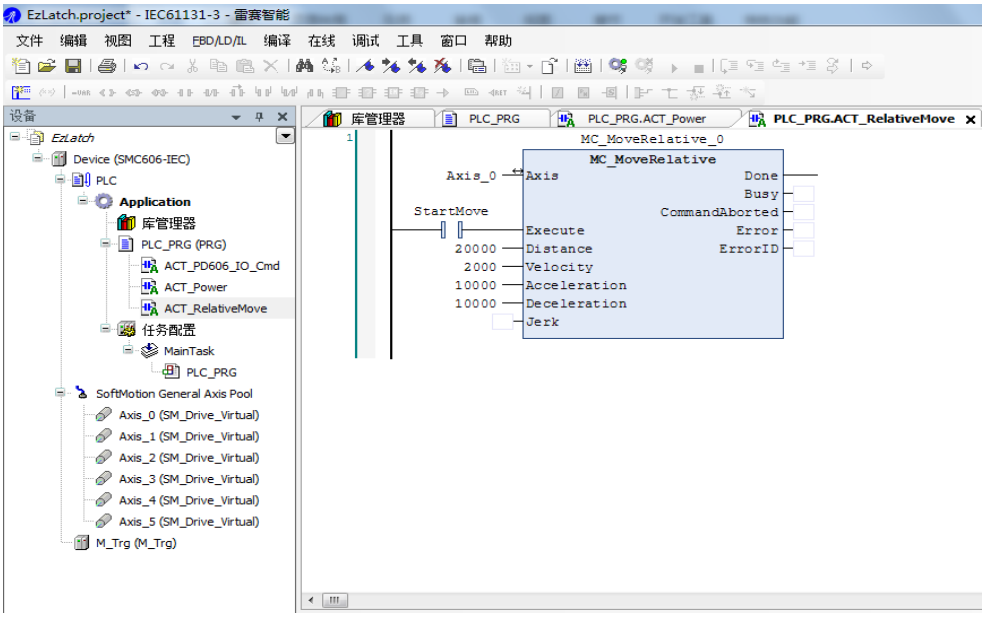


图 9.41　相对运动模块 ACT_RelativeMove

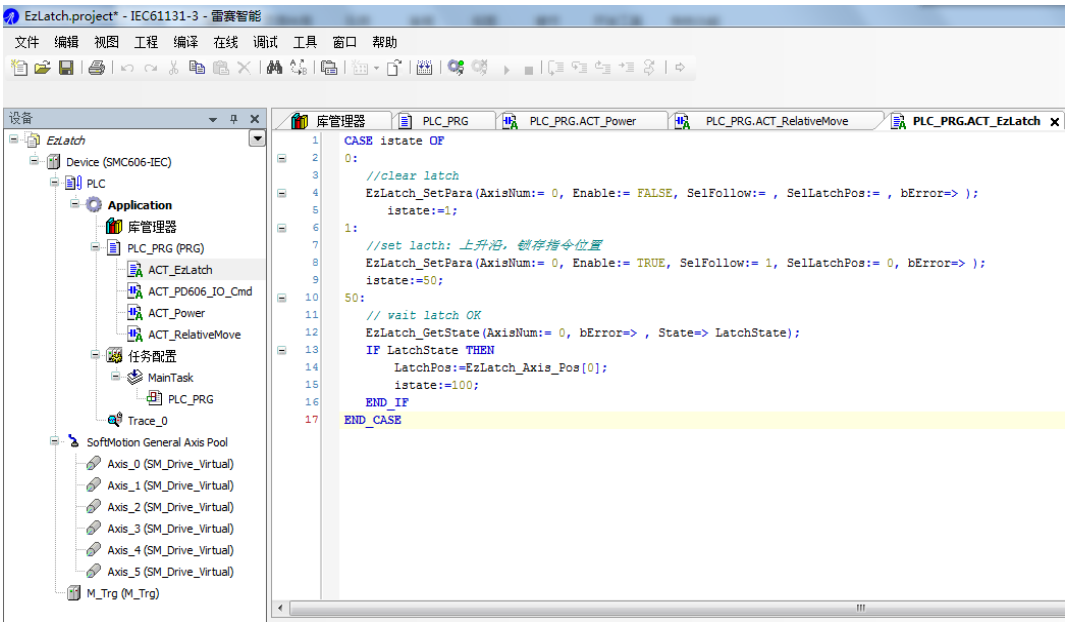


图 9.42　EZ 锁存模块 ACT_EzLatch

运行结果：当 EZ 信号有效时，程序锁存轴 0 的当前位置并赋值给变量 LatchPos，如图 9.43 所示。

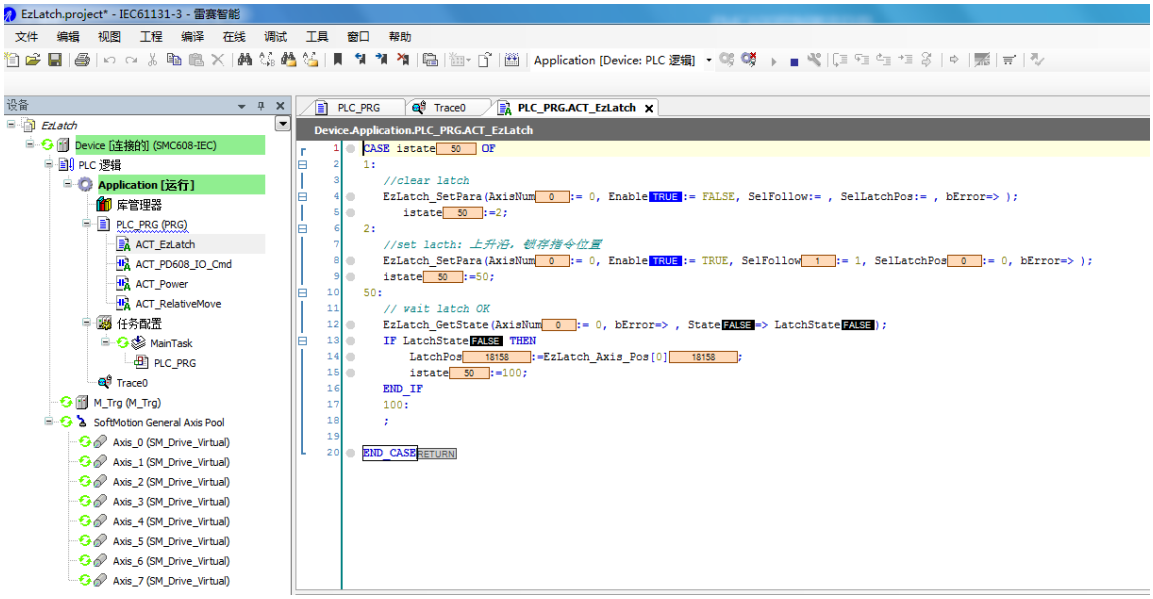


图 9.43　EZ 锁存程序运行结果

9.5.5　单次高速位置锁存方法及例程

使用方法：

1）清除高速锁存状态及标记：用 HighSpeedLatch_SetWorkMode 指令清除发生过的锁存及状态值；

2）配置高速锁存的参数：用 HighSpeedLatch_SetWorkMode 指令配置高速锁存参数，包括锁存触发沿、锁存模式等；

3）配置高速锁存的数据源：用 HighSpeedLatch_ SetSource 指令配置高速锁存的通道、轴号及其数据源；

4）获取高速锁存的当前状态：用 HighSpeedLatch_ ReadState 指令读取高速锁存的当前状态，包括出错状态、是否锁存完成和被锁存的轴号；

5）读取高速锁存位置：判断锁存完成后用 HighSpeedLatch_ReadVal 读取对应轴的高速锁存位置。

例程 9.11：设置轴 0 执行速度为 5000Pulse/s、加减速度为 100000Pulse/s^2、运动距离为 100000Pulse 的相对运动，高速单次锁存轴 0 的指令位置：

1）新建工程并命名 HighSpeedLatch，编程语言为结构化文本 ST。

2）右击 PLC_PRG 主程序，选择“添加对象”-“动作”，命名 ACT_PD606_Cmd，负责脉冲方向模块和硬件 IO 处理模块，编程语言为梯形图 LD，在设备栏双击 ACT_PD606_Cmd 进入程序编辑区，添加 PD606_IO_Cmd 模块并在指令中填上对应的轴号（注意轴号名字必须和设备栏中 SoftMotion General Axis Pool 下的一致）。

3） 按照步骤 2 的方法依次添加上电模块 Power、相对运动模块 RelativeMove、高速锁存模块 ACT_HighSpeedLatch，并填写相应的参数，如图 9.44~9.48。

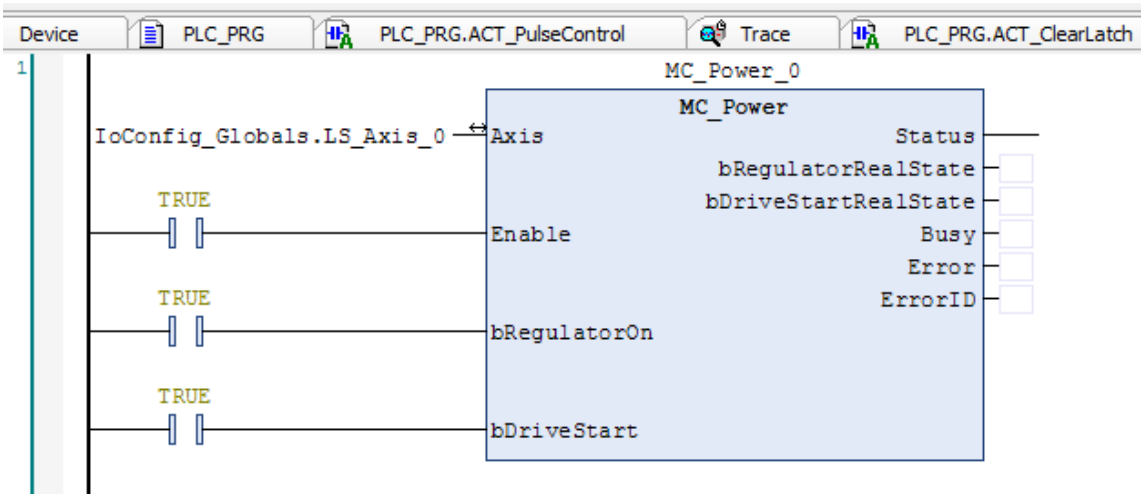


图 9.44　上电模块 Power

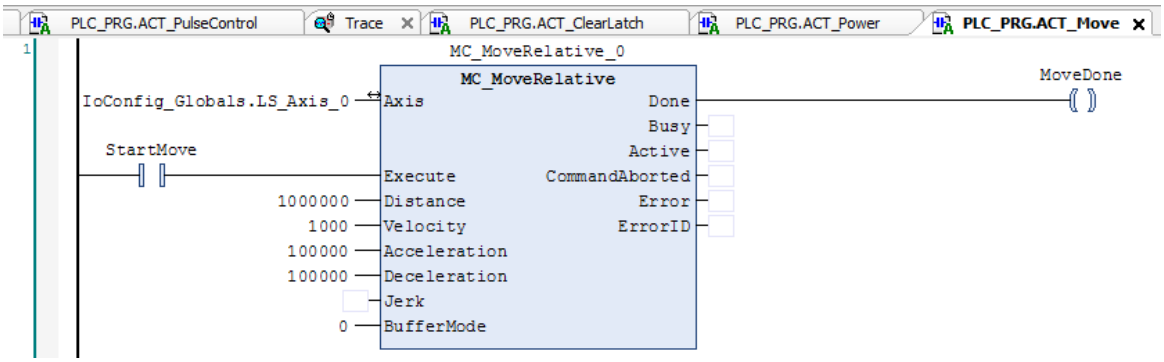


图 9.45　相对运动模块 RelativeMove

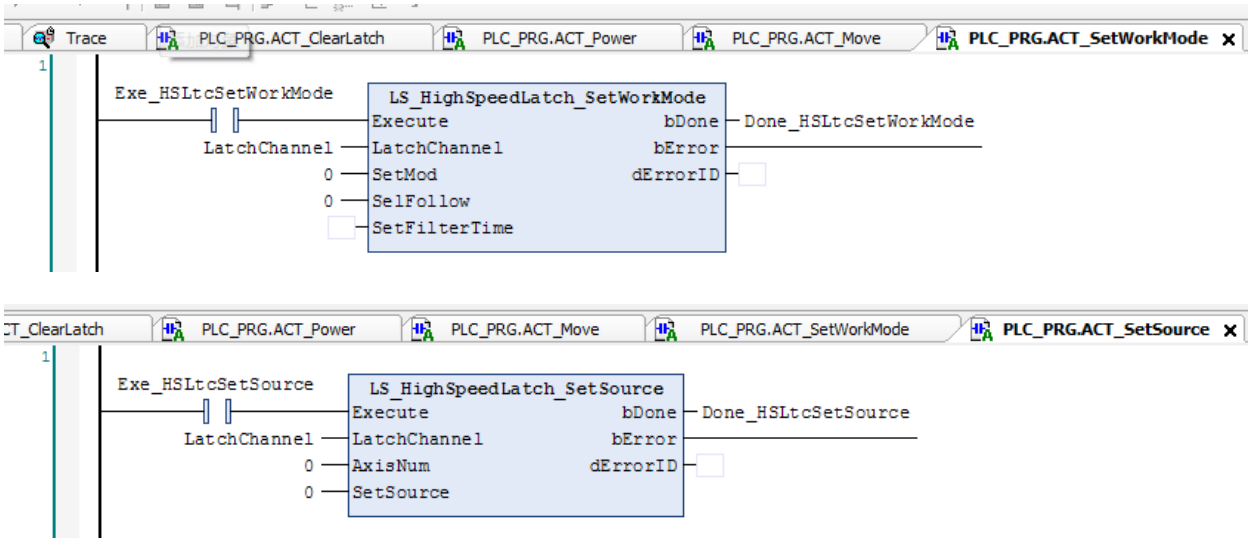


图 9.46 高速锁存模块 ACT_HighSpeedLatch

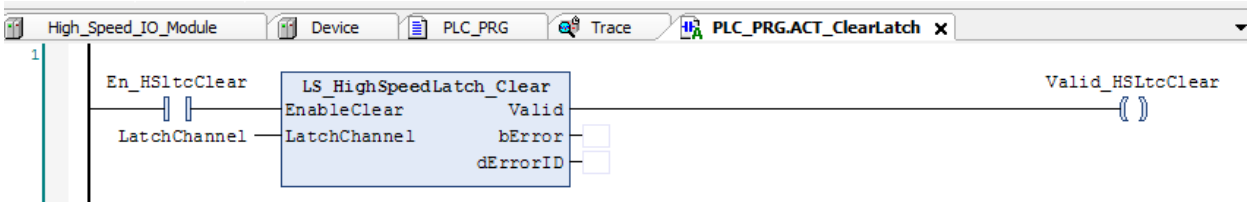


图 9.47　高速锁存 Clear 模块

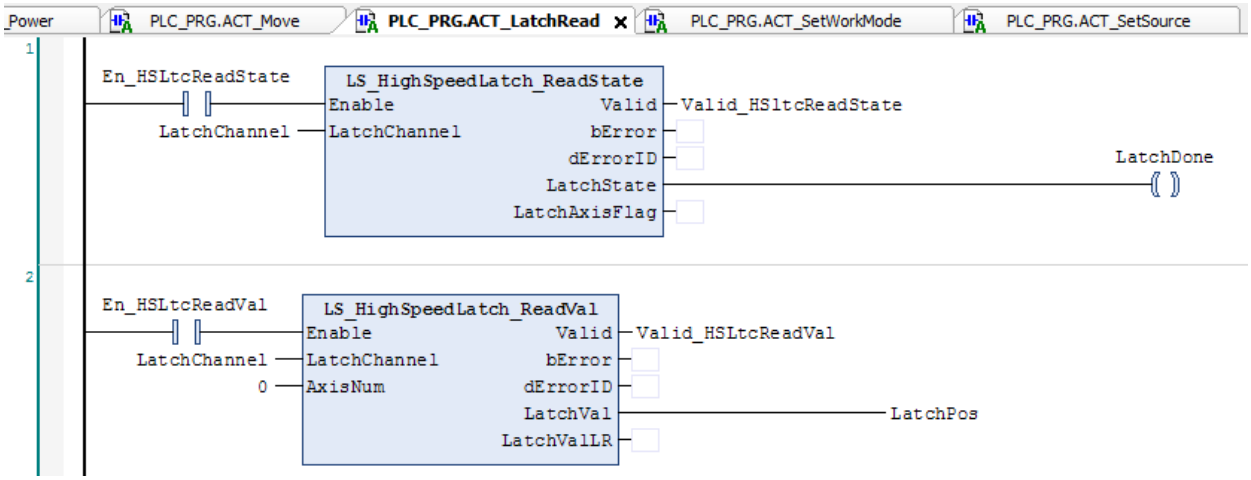


图 9.48　高速锁存 Read 模块

4） 在主程序中分别调用这些模块，编译、下载程序到控制器执行。

运行结果：当输入 IN[20]有效时，程序锁存轴 0 的当前位置并赋值给变量 latchPos，如图 9.49、9.50 所示。

图 9.49　高速锁存程序执行结果

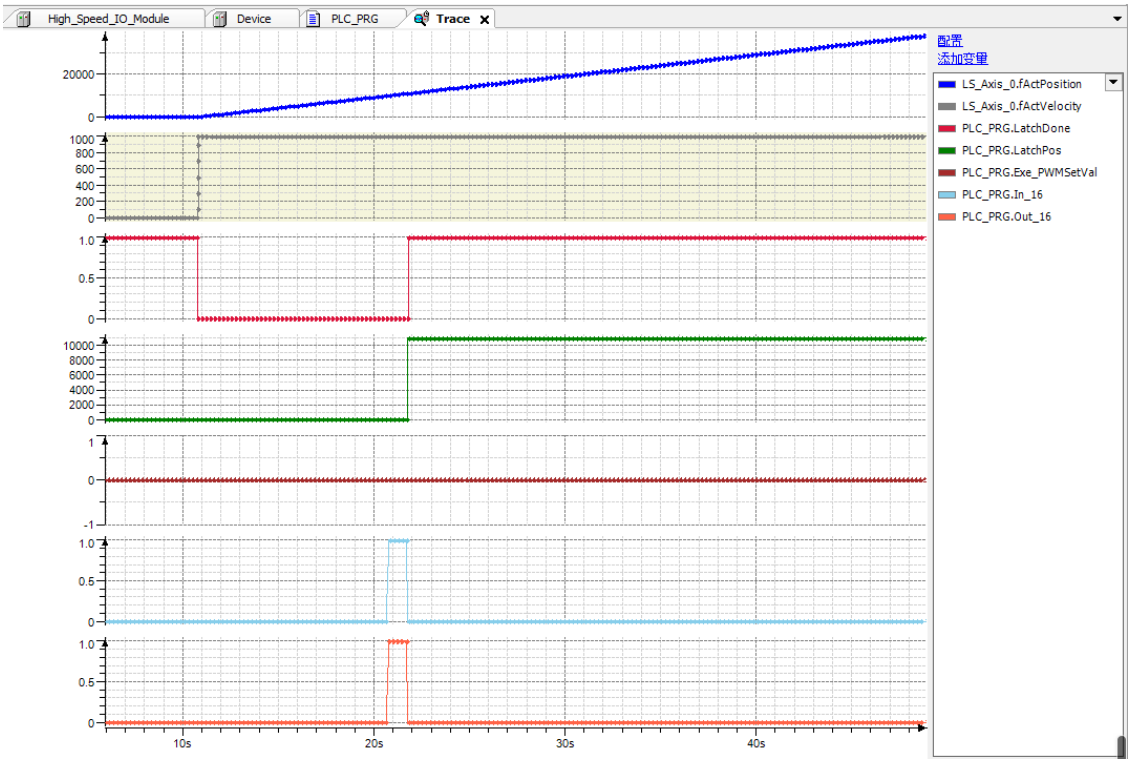


图 9.50　Trace 采集的各参数曲线

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**锁存功能-HighSpeedLatch**”。

9.5.6　连续高速位置锁存方法及例程

连续高速位置锁存可以锁存 256 个位置值，但每次只能读取 32 个位置值，若超过 32 个，则：

1）第一次读取编号 0~31 的位置值，存入锁存寄存器中；

2）第二次读取编号 32~63 的位置值，存入锁存寄存器中；

3）第三次依次类推，直至读取完毕。

注意：读取寄存器是 32 个元素的数组，每次读取都会覆盖上一次的数据，使用时必须在下一次读取前转移寄存器的值。

使用方法：

1） 清除高速锁存状态及标记：用 HighSpeedLatch_SetWorkMode 指令清除发生过的锁存及状态值；

2） 配置高速锁存的参数：用 HighSpeedLatch_SetWorkMode 指令配置高速锁存参数，包括锁存触发沿、锁存模式等；

3） 配置高速锁存的数据源：用 HighSpeedLatch_ SetSource 指令配置高速锁存的通道、轴号及其数据源；

4）获取连续高速锁存的数据个数：用 HighSpeedLatch_FIFOReadNumber 指令读取连续高速锁存的数据个数；

5）读取高速锁存位置：判断读取锁存数据个数完成后用 HighSpeedLatch_FIFOReadVal 读取对应轴的高速锁存位置，超过 32 个则继续读取高速锁存位置。

例程 9.12：设置轴 0 执行速度为 5000Pulse/s、加减速度为 100000Pulse/s^2、运动距离为 100000Pulse 的相对运动，高速单次锁存轴 0 的指令位置：

（1）新建工程并命名 HighSpeedLatch，编程语言为结构化文本 ST。

（2）右击 PLC_PRG 主程序，选择“添加对象”-“动作”，命名 ACT_PD606_Cmd，负责脉冲方向模块和硬件 IO 处理模块，编程语言为梯形图 LD，在设备栏双击 ACT_PD606_Cmd 进入程序编辑区，添加 PD606_IO_Cmd 模块并在指令中填上对应的轴号。

（3）按照步骤 2 的方法依次添加上电模块 Power、相对运动模块 RelativeMove、高速锁存模块 ACT_HighSpeedLatch，并填写相应的参数，如图 9.51~9.53 所示。

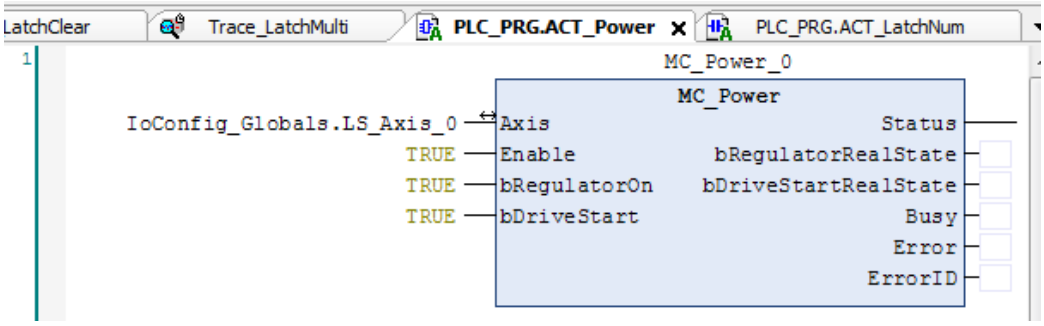


图 9.51　上电模块 Power

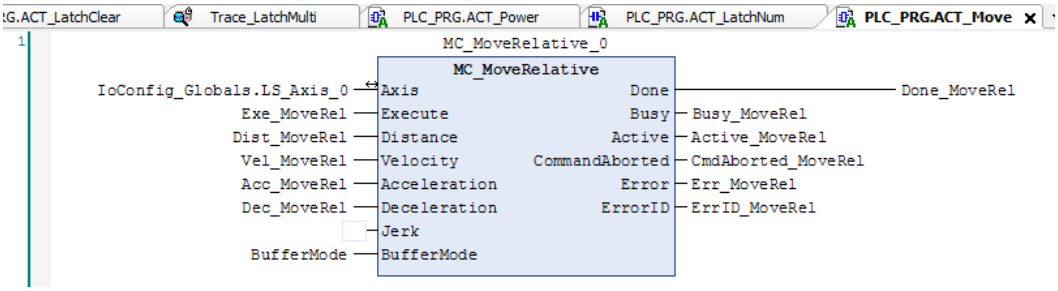


图 9.52　相对运动模块 RelativeMove

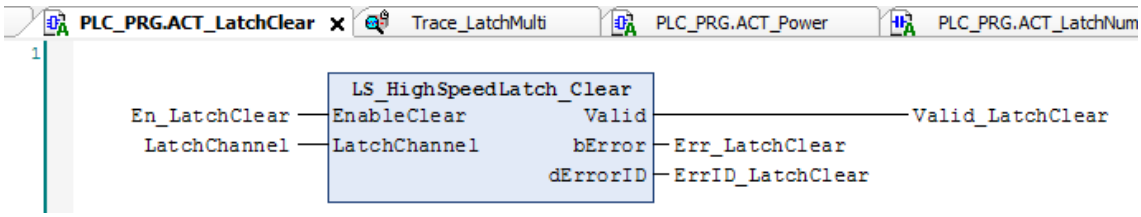


图 9.53　高速锁存模块 ACT_HighSpeedLatch

PLC_PRG | PLC_PRG.ACT_LatchRead | Device | PLC_PRG.AC

```
    Valid_HSLtcClear:BOOL;
    En_LatchFifoRead: BOOL;
    LocalAxis: UINT:=0;
    Valid_LatchFifoReadNum: BOOL;
    LatchNumber: UDINT;
    Valid_LatchFifoReadVal: BOOL;
```

```
IF NOT AllPowerDone THEN
    RETURN;
END_IF

TON_PWM(IN:= , PT:=T#6S , Q=> , ET=> );
TON_Latch(IN:= , PT:=T#1S , Q=> , ET=> );

CASE LatchSwitch OF
    1:
    //清除锁存器
    En_LatchClear:=TRUE;
    IF Valid_LatchClear THEN
        En_LatchClear:=FALSE;
        LatchSwitch:=2;
    END_IF
    ACT_LatchClear();
    2:
    //设置锁存模式
    Exe_LatchSetMode:=TRUE;
    IF Done_LatchSetMode THEN
        Exe_LatchSetMode:=FALSE;
        LatchSwitch:=3;
    END_IF
    ACT_LatchSet();
    3:
    //设置锁存源
    Exe_LatchSetSource:=TRUE;
    IF Done_LatchSetSource THEN
        Exe_LatchSetSource:=FALSE;
        LatchSwitch:=4;
    END_IF
    ACT_LatchSet();
    4:
    Exe_MoveRel:=TRUE;//启动运动
    TON_Latch.IN:=TRUE;
    IF TON_Latch.Q THEN
        StartPWM:=TRUE;//输出PWM触发锁存信号
        LatchSwitch:=5;
    END_IF

    5:
    //运动完成后，从FIFO缓存区中获取锁存点
    IF Done_MoveRel THEN
        //将更新的锁存值转移至数组
```

图 9.54　高速锁存主程序

4）在主程序中分别调用这些模块，编译、下载程序到控制器执行，如图 9.54 所示。

运行结果：当输入 IN[20]有效时，程序锁存轴 0 的当前位置并赋值给变量 latchPos，如图 9.55 所示。

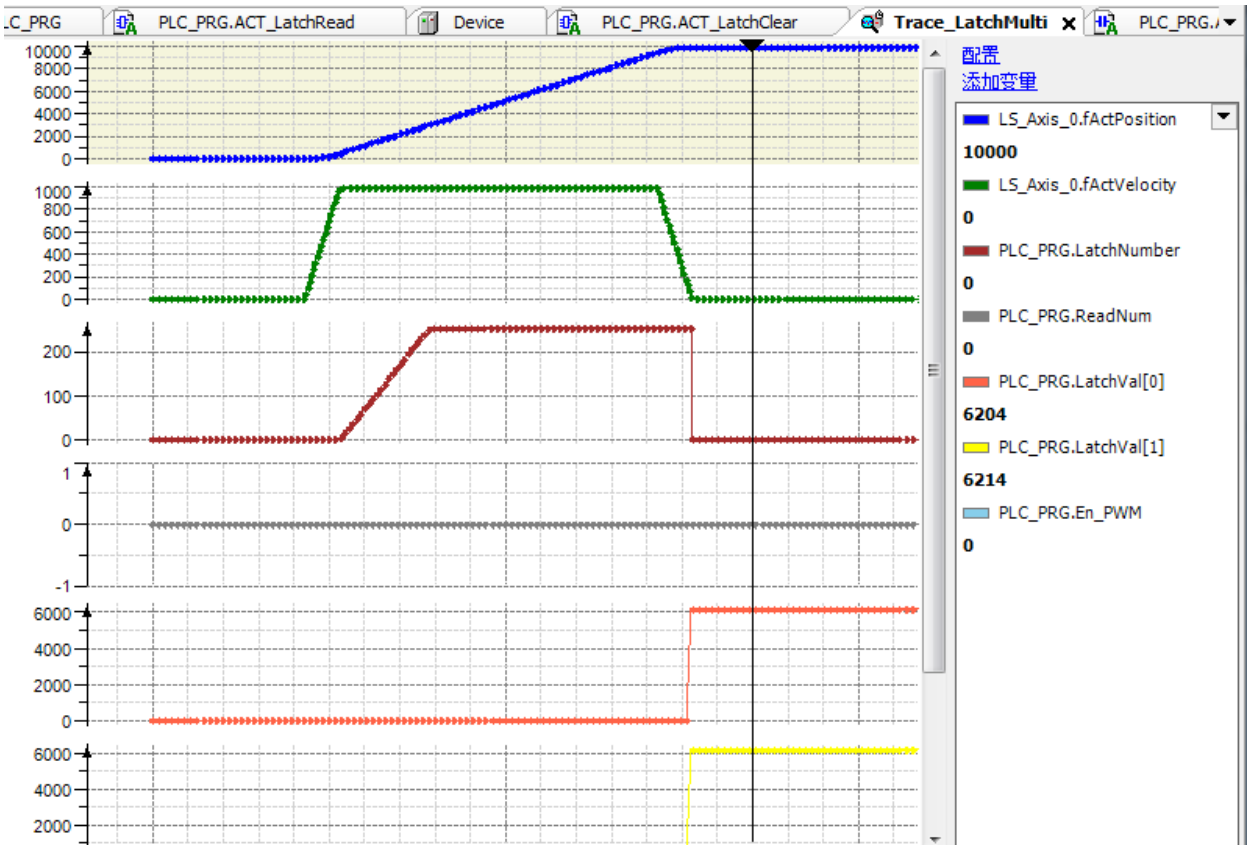


图 9.55　Trace 采集的各参数曲线

9.6 PWM 输出

PMC610 运动控制器提供了 4 路 PWM（脉冲宽度调制）输出信号，与 OUT16~19 复用。PWM 输出均有光电隔离电路，频率和占空比可调，输出频率范围：1HZ~500KHZ，其输出端口详见硬件手册。

PWM 的输出波形如图 9.56 所示，波形的周期为 t2（频率即为 1/t2），占空比为 t1/t2。

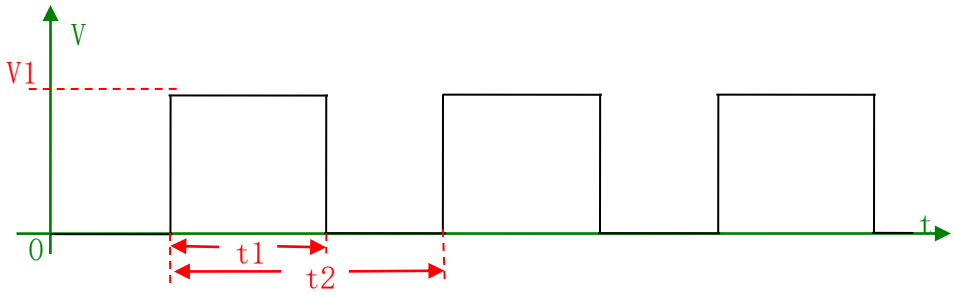


图 9.56　PWM 输出波形

9.6.1 PWM 指令

表 9.9　PWM 输出相关函数

指令名	功能说明
LS_PWM_SetVal	设置 PWM 的参数
LS_PWM_GetVal	获取 PWM 数值

设置 PWM 的参数　LS_PWM_SetVal

设置 PWM 输出参数，启动 PWM 输出功能。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_PWM_SetVal	FUN	LS_PWM_SetVal Execute　bDone PWMNum　bError dErrorID PWMEnable Frequency Ratio	LS_PWM_SetVal(Execute:= （参数）, PWMNum:= （参数）, PWMEnable:= （参数）, Frequency:= （参数）, Ratio:= （参数）, bDone=>（参数）, bError=>（参数）, dErrorID=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令执行。
PWMNum	输出通道	DINT	0 - 3	0	PWM 的输出通道
PWMEnable	使能输出	BOOL	TRUE FALSE	FALSE	TRUE：允许 PWM 输出 FALSE：不允许 PWM 输出
Frequency	频率	DINT	1 - 500000	0	PWM 的频率 1-500000Hz
Ratio	占空比	REAL	0 - 1	0	PWM 的占空比 0-1
VAR_OUTPUT					
bDone	执行完成	BOOL	TRUE FALSE	FALSE	TRUE：表示指令执行完成； FALSE：表示指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误

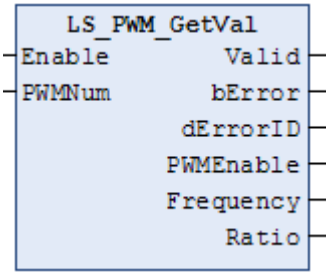
说明：

- 这个指令由“PMC_Controller”库实现。
- 指令用来设置 PWM 的参数，包括输出通道、频率和占空比等参数。
- 共有四路 PWM 通道，分别是 OUT16-OUT19。
- 这是一个单周期的指令，Execute 的上升沿有效。
- 当 PWMEnable 信号为 TRUE 时，触发 Execute 信号，开始 PWM 输出；当 PWMEnable 信号为 FALSE 时，触发 Execute 信号，停止 PWM 输出。

获取 PWM 数值 LS_PWM_GetVal

中断轴正在进行的运动，对轴进行减速停止。

指令外观：

指令	FB/FUN	图形模块	结构文本
LS_PWM_GetVal	FUN		LS_PWM_GetVal(Enable:= （参数）， PWMNum:= （参数）， Valid=>（参数） ， bError=>（参数） ， dErrorID=>（参数）， PWMEnable=>（参数） Frequency=>（参数） ， Ratio=>（参数）);

变量：

VAR_INPUT	名称	类型	有效范围	初始值	描述
Enable	启动	BOOL	TRUE FALSE	FALSE	TRUE：持续执行指令 FALSE：不执行指令
PWMNum	启动	BOOL	TRUE FALSE	FALSE	上升沿触发指令运动。
VAR_OUTPUT					
Valid	完成	BOOL	TRUE FALSE	FALSE	TRUE：指令执行生效中； FALSE：指令未执行
bError	错误	BOOL	TRUE FALSE	FALSE	FALSE-没有错误 True-输入参数越界
dErrorID	错误码	DINT	0，正数	0	错误码返回， 0： 表示无错误
PWMEnable	使能输出	BOOL	TRUE FALSE	FALSE	TRUE：允许 PWM 输出 FALSE：不允许 PWM 输出
Frequency	频率	DINT	1 - 500000	0	PWM 的频率 1-500000Hz
Ratio	占空比	REAL	0 - 1	0	PWM 的占空比 0-1

说明：

- 这个指令由“PMC_Controller”库实现。
- 获取 PWM 参数设置，与“LS_PWM_SetVal”指令配合进行使用。
- 当 Enable 为 TRUE 时，能够获取 PWM 口的使能状态、频率和占空比等参数。

9.6.2 PWM 例程

设置 PWM0 输出频率 10HZ，占空比为 30%的信号，PWM1 输出频率 100HZ，占空比为 50%的信号。步骤如下：

（1）新建工程并命名 PWM，编程语言为结构化文本 ST。

（2）右击 PLC_PRG 主程序，选择“添加对象”-“动作”，命名 ACT_PD606_IO_Cmd，负责硬件 IO 处理模块，编程语言为梯形图 LD，在设备栏双击 ACT_PD606_IO_Cmd 进入程序编辑区，添加 PD606_IO_Cmd 模块并在指令中填上对应的轴号（注意轴号名字必须和设备栏中 SoftMotion General Axis Pool 下的一致）。

（3）按照步骤 2 的方法依次添加 PWM0、PWM1 并填写相应的参数，如图 9.57 所示。

（4）在主程序中调用上述模块完成编程，编译、下载程序到控制器执行。

运行结果：当程序下载到控制器中执行时，PWM0 通道及 PWM1 通道输出设置的 PWM 信号。从图 9.58 可看到 PWM 通道 1 输出 100HZ 频率，占空比为 50%的 PWM 信号。

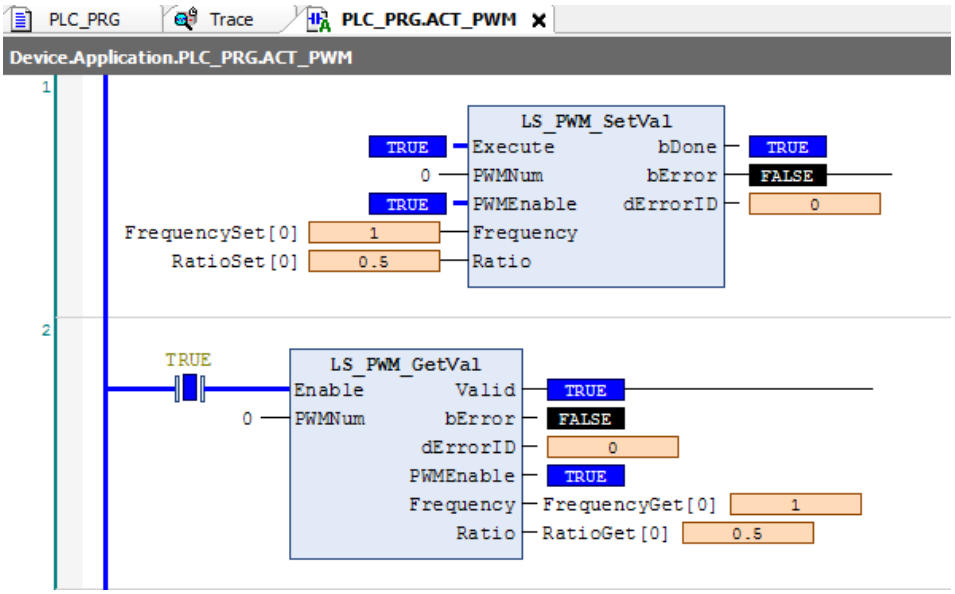


图 9.57　PWM 模块

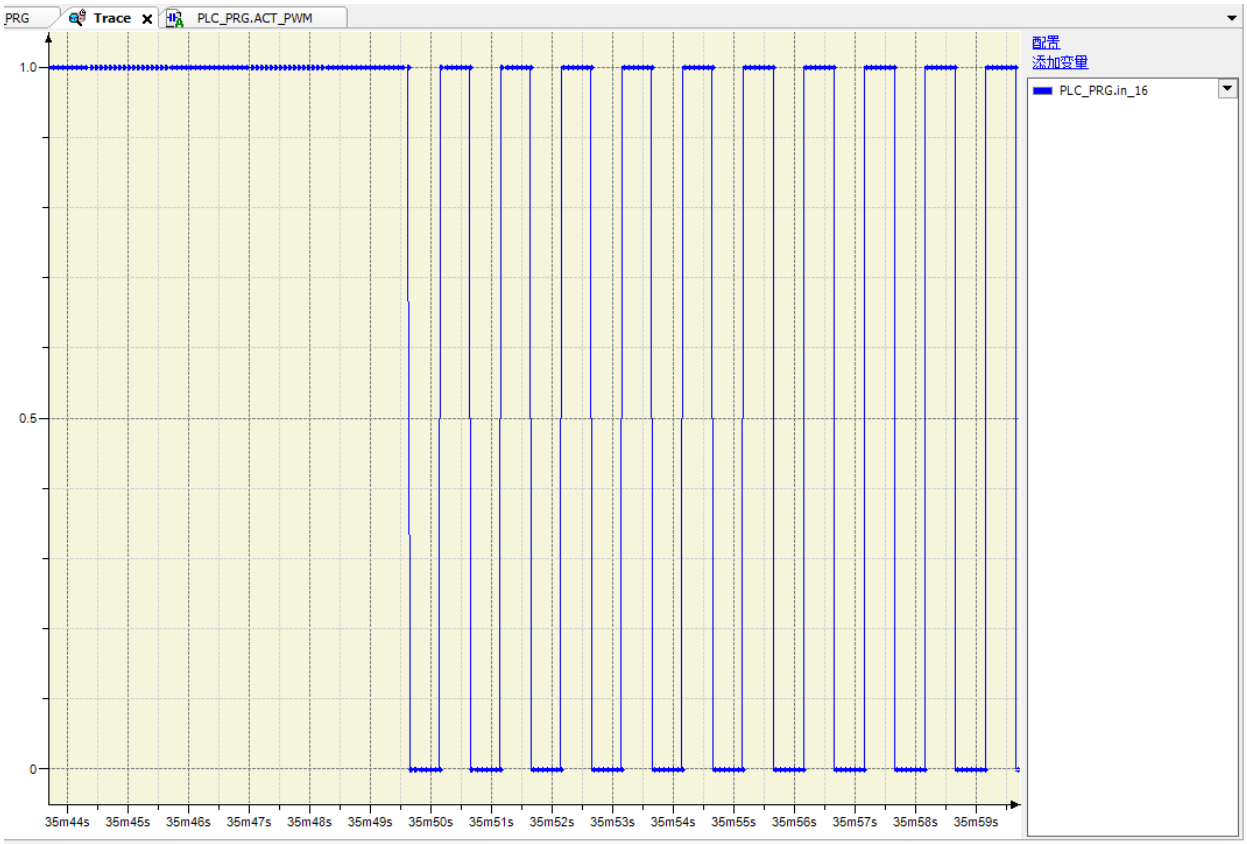


图 9.58　主程序运行结果

程序源码见例程 PWM：

本例程原代码参见 PMC600 软件资料中的“例程”文件夹中的“**PWM**”。

附 录

A-1 错误码 SMC_ERROR（Enumeration）

这个数据结构包括了 SoftMotion 功能块可能返回的错误编号，可以通过错误编号了解发生了什么错误，方便及时处理。调用 SMC_ErrorString 功能块可以根据错误号生成错误字符串输出量。

附表 1　错误码及其描述

错误编号	模块	枚举量	描述
0	所有	SMC_NO_ERROR	无错误
1	DriveInterface	SMC_DI_GENERAL_COMMUNICATION_ERROR	通讯错误（例如 e.g. Sercos 环断开）
2	DriveInterface	SMC_DI_AXIS_ERROR	轴错误
3	DriveInterface	SMC_DI_FIELDBUS_LOST_SYNCHRONITY	总线周期同步失败
10	DriveInterface	SMC_DI_SWLIMITS_EXCEEDED	超出了软件限位
11	DriveInterface	SMC_DI_HWLIMITS_EXCEEDED	超出了硬件限位
12	DriveInterface	SMC_DI_LINEAR_AXIS_OUTOFRANGE	超过轴的位置限制
13	DriveInterface	SMC_DI_HALS_OR_QUICKSTOP_NOT_SUPPORTED	不支持暂停和快速停止
14	DriveInterface	SMC_DI_VOLTAGE_DISABLED	轴未使能
15	DriveInterface	SMC_DI_IRREGULAR_ACTPOSITION	从驱动获取的实际位置为非法值，请检查通信。
16	DriveInterface	SMC_DI_POSITIONLAGERROR	跟随误差超出限制错误。设置位置和实际位置之间的误差超过了给定的限制。
17	DriveInterface	SMC_DI_HOMING_ERROR	回零过程错误
18	DriveInterface	SMC_DI_LICENSING_ERROR	软件授权失败
20	all motion generating modules	SMC_REGULATOR_OR_START_NOT_SET	控制器不能执行或者正在刹车
21	All motion generating modules	SMC_WRONG_CONTROLLER_MODE	轴处于错误的控制模式。
25		SMC_INVALID_ACTION_FOR_LOGICAL	无效的逻辑动作
30	DriveInterface	SMC_FB_WASNT_CALLED_DURING_MOTION	在运动没有结束前，运动模块没有被调用。

31	All modules	SMC_AXIS_IS_NO_AXIS_REF	所给的 AXIS_REF 变量不是 AXIS_REF 类型的。
32	All motion generating modules	SMC_AXIS_REF_CHANGED_DURING_OPERATION	在运动过程中，输入的轴 AXIS_REF 变量被更换。
33	DriveInterface	SMC_FB_ACTIVE_AXIS_DIABLED	在运动过程中，轴失效（MC_Power.bRegulatorOn）
34	All motion generating modules	SMC_AXIS_NOT_READY_FOR_MOTION	轴没有处于准备好的状态，不能执行运动
35	All motion generating function blocks	SMC_AXIS_ERROR_DURING_MOTION	轴在运动过程中报错
40	VirtualDrive	SMC_VD_MAX_VELOCITY_EXCEEDED	超过最大速度（fMaxVelocity）
41	VirtualDrive	SMC_VD_MAX_ACCELERATION_EXCEEDED	超过最大加速度（fMaxAcceleration）
42	VirtualDrive	SMC_VD_MAX_DECELERATION_EXCEEDED	超过最大减速度（fMaxDeceleration）
50	SMC_Homing	SMC_3SH_INVALID_VELACC_VALUES	无效的速度或加速度值
51	SMC_Homing	SMC_3SH_MODE_NEEDS_HW LIMIT	模式要求（可能基于安全原因）关闭限位开关
60		SMC_FRC_NO_FREE_HANDLE	无空闲句柄用于打开文件
70	SMC_SetControllerMode	SMC_SCM_NOT_SUPPORTED	模式不支持
71	SMC_SetControllerMode	SMC_SCM_AXIS_IN_WRONG_STATE	在当前模式下，不能改变控制器模式。
72	SMC_SetControllerMode	SMC_SCM_INTERRUPTED	设置控制器模式被 MC_Stop 指令或者 errorstop 打断
75	SMC_SetTorque	SMC_ST_WRONG_CONTROLLER_MODE	轴不在正确的模式下
90	SMC_ChangeGearingRatio	SMC_CGR_ZERO_VALUES	无效值
91	SMC_ChangeGearingRatio	SMC_CGR_DRIVE_POWERED	电子齿轮参数在轴受控时不可被修改
92	SMC_ChangeGearingRatio	SMC_CGR_INVALID_POSPERIOD	旋转轴无效的模值（<=0）
93	SMC_ChangeGearingRatio	SMC_CGR_POSPERIOD_NOT_INTEGRAL	参数值非整数
110	MC_Reset	SMC_P_FTASKCYCLE_EMPTY	设置的轴运动循环周期错误（fTaskCycle = 0）
120	MC_Reset	SMC_R_NO_ERROR_TO_RESET	轴无错误
121	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER	轴不执行“错误复位”
122	MC_Reset	SMC_R_ERROR_NOT_RESETTABLE	错误不能被复位
123	MC_Reset	SMC_R_DRIVE_DOESNT_ANSWER_IN_TIME	轴的通讯状态不响应
130	MC_ReadParameter，MC_ReadBoolParameter	SMC_RP_PARAM_UNKNOWN	参数个数不明确
131	MC_ReadParameter，MC_ReadBoolParameter	SMC_RP_REQUESTING_ERROR	向驱动设备传递时出错；可通过见功能块 ReadDriveParameter 获取错误号。

132	MC_ReadParameter，MC_ReadBoolParameter	SMC_RP_DRIVE_PARAMETER_NOT_MAPPED	驱动参数没有映射
133	MC_ReadParameter，MC_ReadBoolParameter	SMC_RP_PARAM_CONVERSION_ERROR	参数值转换失败
140	MC_WriteParameter，MC_WriteBoolParameter	SMC_WP_PARAM_INVALID	不可识别的参数编号，或写错误
141	MC_WriteParameter，MC_WriteBoolParameter	SMC_WP_SENDING_ERROR	可通过功能块 WriteDriveParameter 获取错误号。
142	MC_WriteParameter，MC_WriteBoolParameter	SMC_WP_DRIVE_PARAMETER_NOT_MAPPED	驱动参数没有映射
143	MC_WriteParameter，MC_WriteBoolParameter	SMC_WP_PARAM_CONVERSION_ERROR	参数值转换失败
170	MC_Home	SMC_H_AXIS_WASNT_STANDSTILL	轴没有处于 standstill 状态
171	MC_Home	SMC_H_AXIS_DIDNT_START_HOMING	启动回零运动时出错。
172	MC_Home	SMC_H_AXIS_DIDNT_ANSWER	通讯错误。
173	MC_Home	SMC_H_ERROR_WHEN_STOPPING	回零运动错误（没有设置减速）
174	MC_Home	SMC_H_AXIS_IN_ERRORSTOP	回零运动无法执行（轴处于错误停止状态）
180	MC_Stop	SMC_MS_UNKNOWN_STOPPING_ERROR	停止时出现未知错误
181	MC_Stop	SMC_MS_INVALID_ACCDEC_VALUES	无效速度或加速度值
182	MC_Stop	SMC_MS_DIRECTION_NOT_APPLICABLE	方向信号不能使用 shortest
183	MC_Stop	SMC_MS_AXIS_IN_ERRORSTOP	停止无法执行（轴处于错误停止状态）
184	MC_Stop	SMC_BLOCKING_MC_STOP_WASNT_CALLED	MC_Stop 功能块未调用
185		SMC_MS_AXIS_ALREADY_STOPPING	正在执行停止运动，功能块无法中断
200		SMC_UNKNOWN_TASK_INTERVAL	不确定的总线任务时间
201	MC_MoveAbsolute	SMC_MA_INVALID_VELACC_VALUES	速度或加速度值无效
202	MC_MoveAbsolute	SMC_MA_INVALID_DIRECTION	方向错误
226	MC_MoveRelative	SMC_MR_INVALID_VELACC_VALUES	速度或加速度值无效
227	MC_MoveRelative	SMC_MR_INVALID_DIRECTION	方向错误
251	MC_MoveAdditive	SMC_MAD_INVALID_VELACC_VALUES	速度或加速度值无效

252	MC_MoveAdditive	SMC_MAD_INVALID_DIRECTION	方向错误
276	MC_MoveSuperImposed	SMC_MSI_INVALID_VELACC_VALUES	速度或加速度值无效
277	MC_MoveSuperImposed	SMC_MSI_INVALID_DIRECTION	方向错误
301	MC_MoveVelocity	SMC_MV_INVALID_ACCDEC_VALUES	速度或加速度值无效
302	MC_MoveVelocity	SMC_MV_DIRECTION_NOT_APPLICABLE	方向错误，不能使用 shortest/fastest
325	MC_PositionProfile	SMC_PP_ARRAYSIZE	数组大小错误
326	MC_PositionProfile	SMC_PP_STEP0MS	每段间隔时间= t#0s
350	MC_VelocityProfile	SMC_VP_ARRAYSIZE	数组大小错误
351	MC_VelocityProfile	SMC_VP_STEP0MS	每段间隔时间= t#0s
375	MC_AccelerationProfile	SMC_AP_ARRAYSIZE	数组大小错误
376	MC_AccelerationProfile	SMC_AP_STEP0MS	每段间隔时间= t#0s
400	MC_TouchProbe	SMC_TP_TRIGGEROCCUPIED	触发已经激活
401	MC_TouchProbe	SMC_TP_COULDNT_SET_WINDOW	DriveInterface 不支持窗口函数
402	MC_TouchProbe	SMC_TP_COMM_ERROR	通讯错误
410	MC_AbortTrigger	SMC_AT_TRIGGERNOTOCCUPIED	触发已重新分配
426	SMC_MoveContinuousRelative	SMC_MCR_INVALID_VELACC_VALUES	速度或加速度值无效
427	SMC_MoveContinuousRelative	SMC_MCR_INVALID_DIRECTION	无效的方向
451	SMC_MoveContinuousAbsolute	SMC_MCA_INVALID_VELACC_VALUES	速度或加速度值无效
452	SMC_MoveContinuousAbsolute	SMC_MCA_INVALID_DIRECTION	无效的方向
453	SMC_MoveContinuousAbsolute	SMC_MCA_DIRECTION_NOT_APPLICABLE	Direction= fastest 不能使用
475	SMC_ChangeDynamicLimits	SMC_SDL_INVALID_AXIS_STATE	轴状态不正确，应该为 standshill 或 power_off 才能调用该模块
476	SMC_ChangeDynamicLimits	SMC_SDL_INVALID_VELACC_VALUES	无效的速度，加减速度，加加速度
600	SMC_CamRegister	SMC_CR_NO_TAPPETS_IN_CAM	CAM 不包含任何 tappets
601	SMC_CamRegister	SMC_CR_TOO_MANY_TAPPETS	Tappet 组 ID 超过 MAX_NUM_TAPPETS
602	SMC_CamRegister	SMC_CR_MORE_THAN_32_ACCESSES	在一个 CAM_REF 上有超过 32 个访问连接
625	MC_CamIN	SMC_CI_NO_CAM_SELECTED	没有 CAM 被选择
626	MC_CamIN	SMC_CI_MASTER_OUT_OF_SCALE	主轴在超出有效值范围
627	MC_CamIN	SMC_CI_RAMPIN_NEEDS_VELACC_VALUES	对 ramp_in 函数，速度和加速度值必须被指定。

628	MC_CamIN	SMC_CI_SCALING_INCORRECT	缩放变量 fEditor/TableMasterMin/Max 不正确
629	MC_CamIn	SMC_CI_TOO_MANY_TAPPETS_PER_CYCLE	在同一周期内有太多的 Tappet 被激活使用
640	SMC_CAMBounds，SMC_CamBounds_Pos	SMC_CB_NOT_IMPLEMENTED	调用 CAM 数据表的功能块没有执行
675	MC_GearIn	SMC_GI_RATIO_DENOM	分母为 0
676	MC_GearIn	SMC_GI_INVALID_ACC	加速度无效
677	MC_GearIn	SMC_GI_INVALID_DEC	减加速度无效
678	MC_GearIn，MC_CamIn	SMC_GI_MASTER_REGULATOR_CHANGED	在不允许的情形下，主轴的 Enable/Disable 状态切换
679	MC_GearIn	SMC_GI_INVALID_JERK	无效的加加速度
725	MC_Phase	SMC_PH_INVALID_VELACCDEC	速度、减加速度或加速度无效
726	MC_Phase	SMC_PH_ROTARYAXIS_PERIOD0	旋转轴 fPositionPeriod = 0
750	All modules using MC_CAM_REF as input	SMC_NO_CAM_REF_TYPE	所给的 CAM 不是 MC_CAM_REF 类型的
751	MC_CamTableSelect	SMC_CAM_TABLE_DOES_NOT_COVER_MASTER_SCALE	CamTable 的数据没有覆盖 Master area 从 xStart 到 xEnd 的范围
752	MC_CamTableSelect	SMC_CAM_TABLE_EMPTY_MASTER_RANGE	凸轮表的主轴范围为空
753	MC_CamTableSelect	SMC_CAM_TABLE_INVALID_MASTER_MINMAX	凸轮表的主轴范围最大最小值无效
754	MC_CamTableSelect	SMC_CAM_TABLE_INVALID_SLAVE_MINMAX	凸轮表的从轴范围最大最小值无效
775	MC_GearInPos	SMC_GIP_MASTER_DIRECTION_CHANGE	在与从轴连接期间，主轴改变了旋转方向
776	MC_GearInPos	SMC_GIP_SLAVE_REVERSAL_CANNOT_BE_AVOIDED	VAR_INPUT “AvoidReversal”被设置，但是从轴反转无法避免
777	MC_GearInPos	SMC_GIP_AVOID_REVERSAL_FOR_FINITE_AXIS	如果是直线轴，VAR_INPUT “AvoidReversal”不需要设置
800	SMC_BacklashCompensation	SMC_BC_BL_TOO_BIG	齿轮间隙（fBacklash）太大（>位置回合/2）
825	All motion generating function blocks	SMC_QPROF_DIVERGES	内部计算错误
826	All motion generating function blocks	SMC_QPROF_DISTANCE_TOO_SHORT	内部计算错误
827	All motion generating function blocks	SMC_QPROF_NO_RESULT	内部计算错误
728	All motion generating function blocks	SMC_QPROF_INVALID_NEW_LBD	内部轨迹二次方计算错误
729	All motion generating function blocks	SMC_QPROF_BAD_NEGOTIATION	内部轨迹二次方计算错误

730	All motion generating function blocks	SMC_QPROF_INVALID_INTERVAL	内部轨迹二次方计算错误
731	All motion generating function blocks	SMC_QPROF_NOT_ENOUGH_PHASES	内部轨迹二次方计算错误
850	SMC_SetRampType	SMC_SRT_NOT_STANDSTILL_OR_POWEROFF	只允许轴在 STANDSTILL 和 POWER_OFF 状态下设置
851	SMC_SetRampType	SMC_SRT_INVALID_RAMPTYPE	无效的类型
852	SMC_SetRampType	SMC_SMT_NOT_STANDSTILL_OR_POWEROFF	只允许轴在 STANDSTILL 和 POWER_OFF 状态下设置
853	SMC_SetRampType	SMC_SMT_INVALID_MOVEMENTTYPE_OR_POSITIONPERIOD	无效的运动类型或位置周期
854	SMC_SetRampType	SMC_SMT_AXIS_NOT_VIRTUAL	只允许在虚轴上使用的功能块
1000	CNC function blocks which are supervising the licensing	SMC_NO_LICENSE	无 CNC 许可
1001	SMC_Interpolator	SMC_INT_VEL_ZERO	轨迹未被处理，因为设置速度为 0
1002	SMC_Interpolator	SMC_INT_NO_STOP_AT_END	上一个轨迹对象最终的速度 Vel_End > 0。
1003	SMC_Interpolator	SMC_INT_DATA_UNDERRUN	警告：在 DataIn 中处理了 GEOINFO 表，但是列表最后速度未到达。 原因：忘记设置在 DataIn 中的队列的 EndOfList，或者 SMC_Interpolator 速度比轨迹预处理模块要快。
1004	SMC_Interpolator	SMC_INT_VEL_NONZERO_AT_STOP	停止时速度>0。
1005	SMC_Interpolator	SMC_INT_TOO_MANY_RECURSIONS	多个 SMC_Interpolator 使用同一个轴
1006	SMC_Interpolator	SMC_INT_NO_CHECKVELOCITIES	在数据预处理阶段，SMC_CHeckVelocities 不是最后一个模块。
1007	SMC_Interpolator	SMC_INT_PATH_EXCEEDED	内部/数字错误。
1008	SMC_Interpolator	SMC_INT_VEL_ACC_DEC_ZERO	速度、加速度或减加速度值为空或太小。
1009	SMC_Interpolator	SMC_INT_DWIPOTIME_ZERO	dwIpoTime 为 0
1010	SMC_Interpolator	SMC_INT_JERK_NONPOSITIVE	加加速度必须是正值
1011	SMC_Interpolator	SMC_INT_QPROF_DIVERGES	内部计算错误
1012	SMC_Interpolator	SMC_INT_INVLALID_VELOCITY_MODE	无效的速度模式
1013	SMC_Interpolator	SMC_INT_TOO_MANY_AXES_INTERPOLATED	太多轴插补
1050	SMC_Interpolator2Dir	SMC_INT2DIR_BUFFER_TOO_SMALL	数据缓冲区太小
1051	SMC_Interpolator2Dir	SMC_INT2DIR_PATH_FITS_NOT_IN_QUEUE	轨迹在队列中没有运行完。
1070		SMC_XINT_INVALID_DIRECTION	无效的方向输入值

1071		SMC_XINT_NOINTERSECTION	根据给定的 CNC 路径中的 X 轴位置，无法确定终止位置
1080	SMC_Interpolator	SMC_WAR_INT_OUTQUEUE_TOO_SMALL	OutQueueDataIn 太小
1081	SMC_Interpolator	SMC_WAR_END_VELOCITIES_INCORRECT	终止速度不一致
1100	SMC_CheckVelocities	SMC_CV_ACC_DEC_VEL_NONPOSITIVE	速度、加速度或减速度值不为正值
1120	SMC_Controlaxisbypos	SMC_CA_INVALID_ACCDEC_VALUES	fGapVelocity / fGapAcceleration / fGapDeceleration 不为正值
1200	SMC_NCDecoder	SMC_DEC_ACC_TOO_LITTLE	不允许的加速度值
1201	SMC_NCDecoder	SMC_DEC_RET_TOO_LITTLE	不允许的减速度值
1202	SMC_NCDecoder	SMC_DEC_OUTQUEUE_RAN_EMPTY	数据队列为空
1203	SMC_NCDecoder	SMC_DEC_JUMP_TO_UNKNOWN_LINE	由于行号未知，跳转指令不能执行
1204	SMC_NCDecoder	SMC_DEC_INVALID_SYNTAX	语法无效
1205	SMC_NCDecoder	SMC_DEC_3DMODE_OBJECT_NOT_SUPPORTED	对象不支持 3D 模式。
1206	SMC_NCDecoder	SMC_DEC_NEGATIVE_PERIOD	辅助轴周期为负值不正确
1207	SMC_NCDecoder	SMC_DEC_DIMENSIONS_EXCLUSIVE_AU	插补中，轴 A 与轴 U 不能同时执行
1208	SMC_NCDecoder	SMC_DEC_DIMENSIONS_EXCLUSIVE_BV	插补中，轴 B 与轴 V 不能同时执行
1209	SMC_NCDecoder	SMC_DEC_DIMENSIONS_EXCLUSIVE_CW	插补中，轴 C 与轴 W 不能同时执行
1300	SMC_GCodeViewer	SMC_GCV_BUFFER_TOO_SMALL	缓冲区过小
1301	SMC_GCodeViewer	SMC_GCV_BUFFER_WRONG_TYPE	缓冲区元素类型错误
1302	SMC_GCodeViewer	SMC_GCV_UNKNOWN_IPO_LINE	当前行不能从插补器中找到
1500	All function blocks using SMC_CNC_REF	SMC_NO_CNC_REF_TYPE	给出的 CNC 程序不是 SMC_CNC_REF 类型的
1501	All function blocks using SMC_OUTQUEUE	SMC_NO_OUTQUEUE_TYPE	给出的 OutQueue 不是 SMC_OUTQUEUE 类型的
1502	All function blocks using pbyBuffer	SMC_GEOINFO_BUFFER_MISALIGNED	在 pbyBuffer 中，没有使用 4 字节对齐
1600	CNC function blocks	SMC_3D_MODE_NOT_SUPPORTED	功能块只支持 2D 模式
1700	SMC_SmoothAddAxes	SMC_SAA_SMOOTHAREA_TOO_LARGE	平滑范围过大
1701	SMC_SmoothAddAxes	SMC_SAA_SP_INVALID_INPUT	dSmoothingPart 输入参数无效，有效范围[0..1]
1800	SMC_SegmentAnalyzer	SMC_SA_QUEUE_NOT_IN_BUFFER	功能块检测到 OutQueue 缓存已经满，但是数据没有结束
1801	SMC_SegmentAnalyzer	SMC_SA_QUEUE_CHANGED_DURING_OP	当功能块操作 OutQueue 时，OutQueue 缓存发生改变
1820		SMC_OS_INVALID_PARAMETER	无效参数

1830	SMC_BlockSearchSavePos	SMC_BSSP_IPO_NOT_ACTIVE	位置不能保存，插补器是无效的
1831	SMC_BlockSearch	SMC_BS_SAVEDPOS_NOT_REACHED	保存的位置没有找到，可能是一个不同的路径
1832	SMC_BlockSearch	SMC_BS_NO_POS_STORED	SMC_BlockSearchSavePos 不能执行或者为一个错误的状态
1900		SMC_INVALID_FEATURE_FLAG	特征标记有效值范围为 [1..31]
1901		SMC_SMB_HFUN_NOT_SUPPORTED	功能块不支持 H 代码
1902		SMC_SMB_ONLY_3DMODE	功能块只能工作在 3D 模式
1903		SMC_SMB_ERROR_COMPUTING_SPLINE	计算样条错误
1910		SMC_SMM_INVALID_PARAM_NUMBER	辅助参数值太大
2000	SMC_ReadNCFile	SMC_RNCF_FILE_DOESNT_EXIST	文件不存在
2001	SMC_ReadNCFile	SMC_RNCF_NO_BUFFER	无缓冲区被分配
2002	SMC_ReadNCFile	SMC_RNCF_BUFFER_TOO_SMALL	缓冲区太小
2003	SMC_ReadNCFile	SMC_RNCF_DATA_UNDERRUN	缓冲区为空。
2004	SMC_ReadNCFile	SMC_RNCF_VAR_COULDNT_BE_REPLACED	占位符变量不能被替换。
2005	SMC_ReadNCFile	SMC_RNCF_NOT_VARLIST	输入 pvl 不指向 SMC_VARLIST 对象。
2006	SMC_ReadNCFile	SMC_RNCF_NO_STRINGBUFFER	读 NC 文件没有定义缓存
2007	SMC_ReadNCFile	SMC_RNCF_STRINGBUFFER_OVERRUN	读 NC 文件的缓存溢出
2050	SMC_ReadNCQueue	SMC_RNCQ_FILE_DOESNT_EXIST	文件不存在。
2051	SMC_ReadNCQueue	SMC_RNCQ_NO_BUFFER	无缓冲区被定义
2052	SMC_ReadNCQueue	SMC_RNCQ_BUFFER_TOO_SMALL	缓冲区太小
2053	SMC_ReadNCQueue	SMC_RNCQ_UNEXPECTED_EOF	文件异常结束
2100	SMC_AxisDiagnosticLog	SMC_ADL_FILE_CANNOT_BE_OPENED	文件不能打开
2101	SMC_AxisDiagnosticLog	SMC_ADL_BUFFER_OVERRUN	缓冲区过载；WriteToFile 一定是被过于频繁地调用
2200	SMC_ReadCAM	SMC_RCAM_FILE_DOESNT_EXIST	文件不能打开
2201	SMC_ReadCAM	SMC_RCAM_TOO_MUCH_DATA	要存储地 CAM 过大
2202	SMC_ReadCAM	SMC_RCAM_WRONG_COMPILE_TYPE	错误地编译模式
2203	SMC_ReadCAM	SMC_RCAM_WRONG_VERSION	文件版本错误
2204	SMC_ReadCAM	SMC_RCAM_UNEXPECTED_EOF	文件异常结束
3001	SMC_WriteDriveParamsToFile	SMC_WDPF_CHANNEL_OCCUPIED	SMC_WDPF_TIMEOUT_PREPARING_LIST

3002	SMC_WriteDriveParams ToFile	SMC_WDPF_CANNOT_CREATE_FILE	文件不能被创建
3003	SMC_WriteDriveParams ToFile	SMC_WDPF_ERROR_WHEN_READING_PARAMS	读取参数出错
3004	SMC_WriteDriveParams ToFile	SMC_WDPF_TIMEOUT_PREPARING_LIST	准备参数表时超时
5000	SMC_Encoder	SMC_ENC_DENOM_ZERO	编码参考的转变因数（dwRatioTechUnitsDenom）分母为 0。
5001	SMC_Encoder	SMC_ENC_AXISUSEDBYOTHERFB	其他模块尝试在编码器轴上处理运动。
5002	DriveInterface	SMC_ENC_FILTER_DEPTH_INVALID	无效的过滤器深度。

A-2 错误码 CAA Net ERROR

这个数据结构包括了调用“CAA_NetBaseServices.lib”库返回的错误编号。可通过错误编号找到对应的报错信息。

附表 2　报错名及其描述

报错名	报错代码	注释
NO_ERROR	0	没报错
FIRST_ERROR	6000	
TIME_OUT	6001	超时
INVALID_ADDR	6002	IP 地址无效
INVALID_HANDLE	6003	句柄无效
INVALID_DATAPOINTER	6004	数据指针无效
INVALID_DATASIZE	6005	数据尺寸无效
UDP_RECEIVE_ERROR	6006	接收不到 UDP 通讯数据
UDP_SEND_ERROR	6007	无法发送 UDP 通讯数据
UDP_SEND_NOT_COMPLETE	6008	UPD 通讯数据发送未完成
UDP_OPEN_ERROR	6009	创建 UPD 端口失败
UDP_CLOSE_ERROR	6010	关闭 UPD 端口失败
TCP_SEND_ERROR	6011	无法发送 TCP 通讯数据
TCP_RECEIVE_ERROR	6012	接收不到 TCP 通讯数据
TCP_OPEN_ERROR	6013	创建 TCP 端口失败
TCP_CONNECT_ERROR	6014	无法建立 TCP 连接
TCP_CLOSE_ERROR	6015	无法关闭 TCP 连接
TCP_SERVER_ERROR	6016	TCP 服务报错
WRONG_PARAMETER	6017	有一个无效的参数
ERROR_UNKNOWN	6018	未知错误
TCP_NO_CONNECTION	6019	没有 TCP 连接
IOCTL_ERROR	6020	内部错误 (例如不支持 IOCTL)
FIRST_MF	6050	
LAST_ERROR	6099	

A-3 错误码 PMC_BasicModule GVL_Err

这个数据结构包括了调用“PMC_BasicModule.lib”库所返回的错误编号，可通过错误编号找到对应的报错信息。

这里报错信息主要是在插补运动中发生的错误。

附表 3　错误信息及其描述

错误信息	代码	说明
IpoErrID_ComformFail	16#FFFFFFF	不可识别的控制器
IpoErrID_ParaErr	16#FFFFFFE	参数错误：IpoCycle、Jerk、PreProcessGCodeNum、LimitVel、LimitAccDec、LimitMaxAcc、LimitMaxAccJerk、RoundDiameter 参数不可设置值为 0
IpoErrID_CirAxisList	16#FFFFFFD	平面圆弧插补轴列表 CircleAxisList 错误，该值只能为 3，5，6
IpoErrID_CirDir	16#FFFFFFC	平面圆弧插补的方向输入错误
IpoErrID_CirMode	16#FFFFFFB	平面圆弧插补的模式错误
IpoErrID_DataType	16#FFFFFFA	插补的数据类型 DataType 错误
IpoErrID_Vel	16#FFFFFF9	插补的速度错误
IpoErrID_Acc	16#FFFFFF8	插补的加速度错误
IpoErrID_Dec	16#FFFFFF7	插补的减速度错误
IpoErrID_CirParaErr	16#FFFFFF6	平面圆参数错误，输入的参数不能组成圆
IpoErrID_BufferOver	16#FFFFFF5	缓冲区满
IpoErrID_CirRadErr	16#FFFFFF4	平面圆弧半径小于 0.000001
IpoErrID_CirAngelOrg	16#FFFFFF3	平面圆弧目标角度圆心参数错误
IpoErrID_CirThreePoint	16#FFFFFF2	平面圆弧三点参数错误，三点不能连接成圆弧
IpoErrID_ConfigAxisErr	16#FFFFFF1	轴列表中，配置轴号错误，轴号重复
IpoErrID_SpaceCircleParaErr	16#FFFFFF0	空间圆弧参数错误，有两个点重合或三点共线
IpoErrID_AxisNotStandStill	16#FFFFF0	轴启动时不是 standstill 状态
Erroroffset	16#FFFFFE0	偏移
ErrorPosInput	16#FFFFFEF	输入的点错误
ErrorCalJoint	16#FFFFFEE	计算的相交点错误
ErrorPunchJog	16#FFFFFED	Jog
ErrorPunchJogInput	16#FFFFFFE	Jog
ErrWeldingoffset	16#FFFFFC0	焊接偏移
ErrorCalCen	16#FFFFFC1	计算圆心错误
ErrorRotateUnite	16#FFFFFC2	输入脉冲当量错误

圆弧错误信息		
ErrorCheckCircleOffset	16#FFFFFB0	圆弧错误检测
ErrorCircleRatioError	16#FFFFFB1	圆弧输入比例错误
ErrorCirclePreciseError	16#FFFFFB2	圆弧插补输入精度错误
ErrorThreeCircleInputError	16#FFFFFB3	圆弧插补输入点位置错误
ErrorCenEndInputError	16#FFFFFB4	圆弧插补输入终点半径点
机器人正逆解转换错误码		
ErrorRobotOffset	16#FFFFF90	机器人错误码偏移
ErrorRobotJ3BeyondPos	16#FFFFF91	J3 轴出现了位置跳变
ErrorSCARAElbow	16#FFFFF92	scara 机械手的 elbow 输出错误
ErrorSCARARangeDegree	16#FFFFF93	scara 轨迹解析时输入的角度范围错误
ErrorSCARAArmLength	16#FFFFF94	scara 轨迹解析时输入的 arm 长度错误
ErrorSCARAPosOutRange	16#FFFFF95	scara 轨迹解析时输入的位置点超过了范围
ErrorSCARAStartPosOutRange	16#FFFFF95	scara 轨迹解析时输入的起点位置点超过了范围
ErrorSCARAEndPosOutRange	16#FFFFF95	scara 轨迹解析时输入的终点位置点超过了范围
ErrorSCARAElbowRange	16#FFFFF96	scara 轨迹解析时输入的 elbow 与运动的范围不匹配

A-4 错误码 GVL_PMC_ErrID

这个数据结构包括了调用“PMC_Controller.lib”库所返回的错误编号，可通过错误编号找到对应的报错信息。

这里的报错信息主要是在调用 PMC 本身的硬件接口功能时出现的错误。

附表 4　错误信息及其描述

错误信息	报错代码	说明
PMWErrorID_OverMaxNum	9700	PWM 设置参数错误码
PMWErrorID_ParaOverLimit	9701	PWM 超过参数限制
AxisNum_Input_BeyondMax	9800	输入轴参数超过最大限制
HighSpeedCmp2Deg_WorkMode_BeyondMax	9810	高速二维比较输出比较模式超出范围
HighSpeedCmp2Deg_CMPNum_Beyond	9811	高速二维比较器通道超出范围
HighSpeedCmp2Deg_OutNum_Beyond	9812	高速二维比较输出口超出范围

AxisErrID_Axis0ELP	9900	Axis0 正限位
AxisErrID_Axis0ELN	9901	Axis0 负限位
AxisErrID_Axis1ELP	9902	Axis1 正限位
AxisErrID_Axis1ELN	9903	Axis1 负限位
AxisErrID_Axis2ELP	9904	Axis2 正限位
AxisErrID_Axis2ELN	9905	Axis2 负限位
AxisErrID_Axis3ELP	9906	Axis3 正限位
AxisErrID_Axis3ELN	9907	Axis3 负限位
CounterErrorID_Para	9920	设置参数不正确
CounterErrorID_Dnt_Enable	9921	计数器对应 IO 功能未配置
CounterErrorID_Channel_Dnt_Exist	9922	设置通道不存在
AxisErrID_Axis0RatioChange	9930	不允许在运动中改变 Axis0 传动比
AxisErrID_Axis1RatioChange	9931	不允许在运动中改变 Axis1 传动比
AxisErrID_Axis2RatioChange	9932	不允许在运动中改变 Axis2 传动比
AxisErrID_Axis3RatioChange	9933	不允许在运动中改变 Axis3 传动比
AxisErrID_PulseMode	9940	设置的脉冲模式不存在
AxisErrID_Axis0SpeedOverflow	9960	Axis0 速度超出硬件允许的最大速度
AxisErrID_Axis1SpeedOverflow	9961	Axis1 速度超出硬件允许的最大速度
AxisErrID_Axis2SpeedOverflow	9962	Axis2 速度超出硬件允许的最大速度
AxisErrID_Axis3SpeedOverflow	9963	Axis3 速度超出硬件允许的最大速度
AxisErrID_Axis0SpeedJump	9980	Axis0 速度变化超出范围
AxisErrID_Axis1SpeedJump	9981	Axis1 速度变化超出范围
AxisErrID_Axis2SpeedJump	9982	Axis2 速度变化超出范围
AxisErrID_Axis3SpeedJump	9983	Axis3 速度变化超出范围

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

深圳市雷赛控制技术有限公司
地　址：深圳市深圳市南山区学苑大道 1001 号南山智园 A3 栋 9 楼
邮　编：518052
电　话：0755-26415968
传　真：0755-26417609
Email：info@szleadtech.com.cn
网　址：http：//www.szleadtech.com.cn