资料编码：2304LC1200-1

LC1200控制器
编程手册

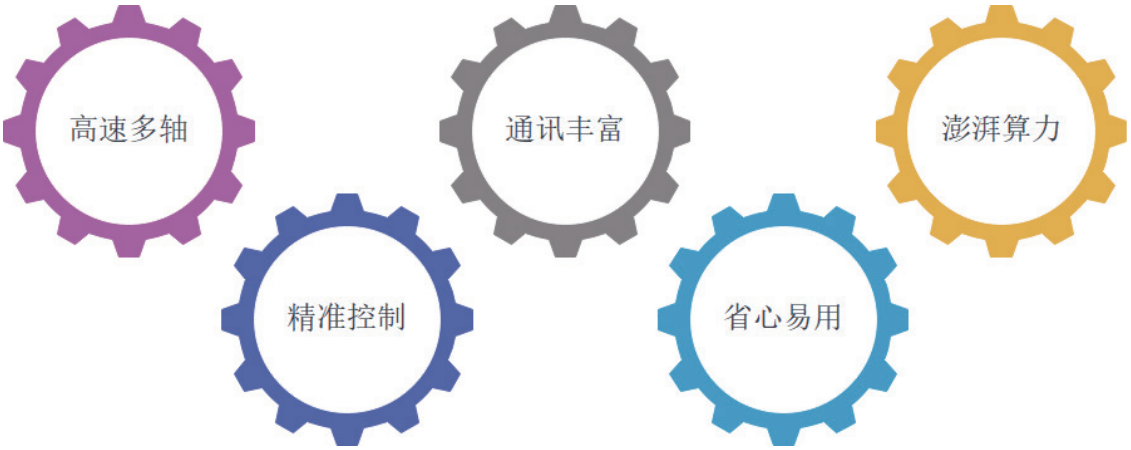
苏州市凌臣采集计算机技术有限公司
www.szpcbase.com

公司简介

苏州市凌臣采集计算机有限公司成立于2006年，是一家本着与客户共赢、为客户创造方案价值的经营理念的企业。为客户提供各种测试测量、运动控制、机器视觉、机器人等自动化设备的核心零部件和系统解决方案。凌臣科技于2017年开始创立凌臣采集LCT品牌，研发了包括工控机、PLC控制器、EtherCAT步进驱动器、PCI/PCIe运动控制卡、远程IO模块、气动阀岛等产品，为我司的客户提供了更具性价比的方案解决。

凌臣科技十分重视研发的投入，目前员工总数270人其中研发技术人员占比超50%。公司同时于2014年成立了基于PC-Base的“授人渔“自动化培训机构,为自动化行业培养了大量专业技术人才。
公司和ADLINK、ABB、ACS、TRIO、 TOYO等知名企业保持着长期紧密的合作关系。

LC1200系列
通用型运动控制器



凌臣全新一代LC1200系列运动控制器，结合CODESYS工业自动化软件平台，将运动控制技术和信息化技术融合于一身，更好的服务于非标自动化，3C,光伏，物流，锂电。。。

版权声明

前　言

版本更新说明

更新时间	版本号	更新内容
2023年01月	V_1.0	初版发行
2023年03月	V_1.1	添加运动指令详解内容
2023年04月	V_1.1.1	优化运行指令详解内容

目录

第 1 章 概述

1.1 PLCopen简介

PLCopen是一个独立的世界性组织，根据用户的需求提供工业自动化的效率。它成立于1992年，总部设在荷兰，在美国、日本和中国设有支持办事处。PLCopen 遵循市场需求的要求，其主要重点是通过定义通用标准来提高自动化效率。为了降低工业工程的成本，PLCopen及其成员一直专注于围绕IEC 61131-3标准的技术规范。PLCopen组织为协调原则性的技术挑战提供了一个坚实的基础，并为成员提供了一个工作平台。

PLCopen 中国组织作为国际组织世界范围内的第三支区域性的推广机构（PC5）承担着该项标准在中国区域的推广工作，旨在搭建工业控制领域交流平台，成为技术标准与行业发展趋势的信息纽带，在供应商与最终用户之间搭建互通桥梁，现已有 30 余家国内外知名企业、高校成为了组织的骨干。

1.2 IEC 61131-3标准简介

IEC 61131-3标准是由国际电工委员会（International Electrotechnical Commission，缩写为IEC）于1993年12月所制定IEC 61131标准的第3部分，是用于规范可编程逻辑控制器（PLC），DCS，IPC，CNC和SCADA的编程系统的标准。在工业控制领域，IEC 61131-3标准目前已经成为应用趋势。

IEC 61131-3 规范的语法提出一套可跨不同目标平台的可编程控制器实现机制。规范中透过模组化的规划与设计，将控制动作分为逻辑运算与硬件动作两个部分：其一，逻辑部分以共同的描述格式来统一 IEC 61131-3 所定义的各语法并加以实现；其二，硬件动作则针对各硬件设计专属之固件函数库，使得控制逻辑可以在各目标平台上使用硬件资源，这样的设计使不同的控制芯片皆可执行以 IEC 61131-3 语法所设计的控制动作，而设计人员只需学会 IEC 61131-3 语法，便可借由符合各项标准的语言架构，进而能建立任何人皆可了解的程序，也就可使用所支持的控制芯片进行可编程控制器设计。

此外，由于所设计的程式码可以在不同的目标平台间重复使用。因此，透过自行建立的函数库及利用重复使用的特性，更可缩短自动化流程的开发时程。

1.3 IEC 61131-3编程语言

IEC 61131-3 标准一共定义了 5 种标准编程语言：

- 指令表（Instruction List，IL）
- 结构化文字（Structured Text Language，ST/STL）
- 梯形逻辑图（Ladder Diagram，LD）
- 功能块图（Function Block Diagram，FBD）
- 顺序功能流程图（Sequential Function Chart，SFC）

其中使用IL、LAD、SFC编制的程序都可以相互转换后进行显示、编辑。

Lingchen-LC1200系列运动控制器选用的是 CODESYS 编程平台，该平台完整支持PLCopen 规范，用户可以引用许多标准的功能函数库；除完整支持IEC 61131-3标准的5种编程语言外，还额外支持顺序功能块（Continuous Function Chart，CFC）的编程语言，共计6种，更易于 PLC 厂家和用户开发自己专有的功能块和指令库；也可借用已有的类似控制程序，形成行业特点的“工艺包”，可显著提高用户编程效率。

第 2 章 Lingchen-LC1200系列控制器概述

2.1 Lingchen-LC1200系列（基础款）中型PLC控制器

2.1.1 产品概述

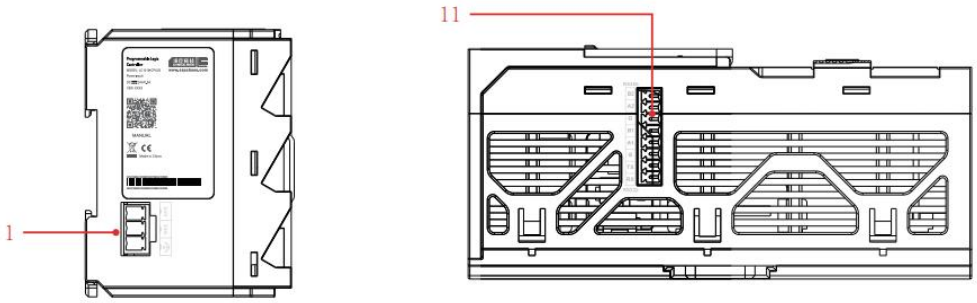
Lingchen-LC1200系列（以下简称LC1200）控制器是一款采用模块化结构设计的可编程逻辑控制器，为用户提供智能自动化解决方案。

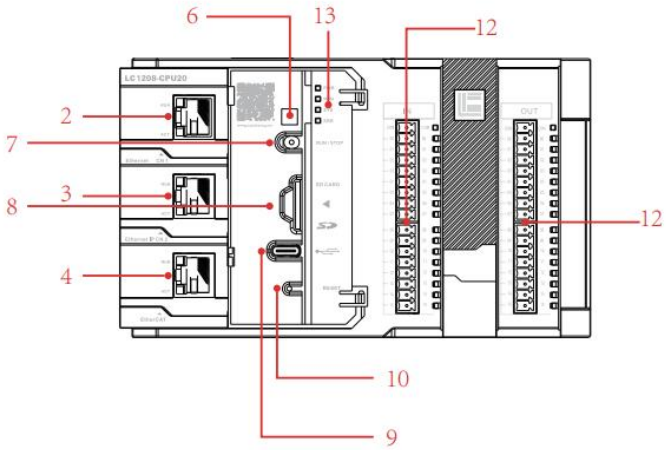
LC1200控制器采用 IEC61131-3 编程语言体系，支持 6 种编程语言。可以通过EtherCAT 总线进行机架的远程扩展。通过EtherCAT 总线可实现高性能运动控制功能；具有单轴加减速控制功能、电子齿轮功能、电子凸轮功能、CNC功能等；同时支持 RS485、以太网、USB 等通信功能， 满足用户多样化的应用需求。

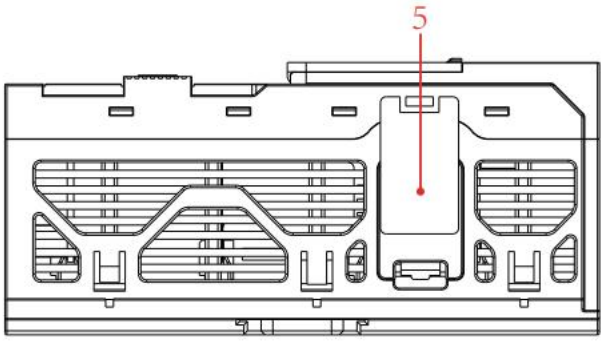
LC1200控制器具有以下功能特点：

- 支持 IEC61131-3 编程环境，符合 PLCopen 规范，提供了 PLC 逻辑控制、SoftMotoin、CNC等多轴逻辑控制功能；
- 控制器内置 16 入 16 出的 I/O 端口；
- 可通过 EtherCAT 总线扩展机架，支持更大的 I/O 点数；
- 更快的指令执行速度；
- 更大的程序容量和数据存储区；
- 支持更多的 EtherCAT 总线、Modbus 通信、USB通信等；
- 更易用的软件，更符合国内用户应用习惯；
- 支持在线侦错模式、离线仿真调试；
- 支持在线编辑下载用户程序。

2.1.2 产品硬件及模块说明



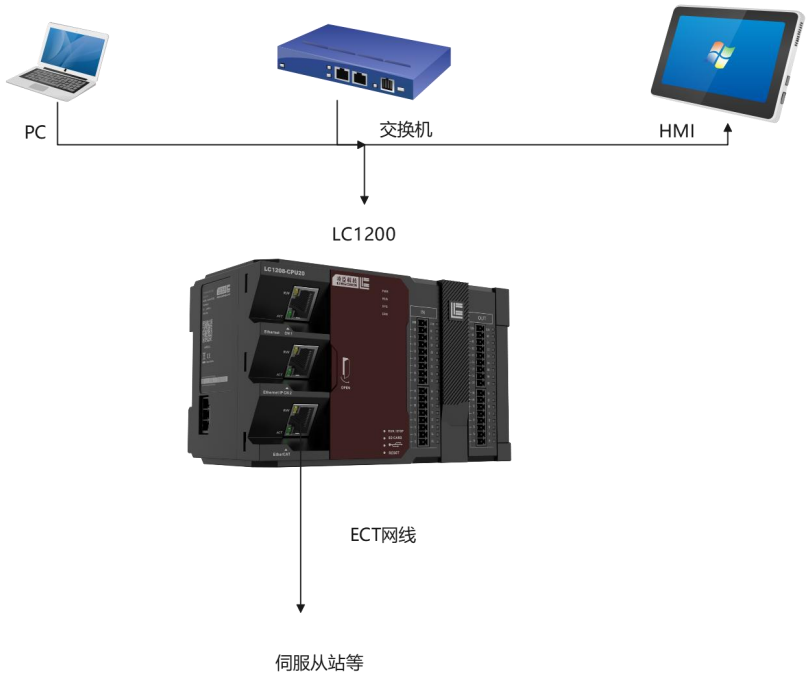




标号	接口名称	功能描述
1	24V 电源输入端子	直流 24V 1A 电压输入
2	网口（eth0）	默认ip地址eth0:192.168.0.99 1. MODBUS TCP 协议 2. 标准以太网功能 3. 系统程序调试 4. 用户程序下载与调试（只支持 IPv4）
3	网口（eth1）	默认ip地址eth1:192.168.2.99 1. MODBUS TCP 协议

		2. Ethernet IP 协议 3. 标准以太网功能
4	网口（eth2）	默认ip地址eth2:192.168.8.99 EtherCAT协议
5	电池	RTC电池供电
6	终端电阻	出厂默认置OFF
7	拨码开关	RUN/STOP拨码开关
8	SD卡接口	用于导入/导出程序
9	USB（Type-C）	程序下载及调试，导入/导出程序
10	RESET按键	复位按键
11	通讯串口	2路RS485接口，1路RS232接口，均支持MODBUS协议
12	16路IO	16路输入&16路输出
13	LED指示灯	电源等状态指示灯

LC1200控制器架构集成示意图如下图所示：



LED指示灯状态说明：

指示灯名称	状态含义			颜色
	常亮	常灭	闪烁	
PWR	有供电	无供电	\	绿色
RUN	run	stop	\	绿色
SYS	启动中	未启动/死机	PLC进程运行中	绿色
ERR	运行错误	运行无错	总线错误	红色

RESET按键说明：

RESET按键可以用于清除PLC内已下载的程序或将网口IP地址复原。具体操作如下：

1） 长按RESET按键2~6s，PLC恢复出厂值，擦除已下载的用户工程；

2） 长按6s以上，PLC的三个网口恢复默认IP地址。

第 3 章 运动控制应用系统组成与运动控制程序组成

3.1 运动控制应用系统组成

LC1200是一个具有 SoftMotion 运动控制功能 (CAM/CNC/ROBOT) 的通用型可编程控制器，通过 EtherCAT总线控制多个运动轴，典型的控制总线网络如下图。其中伺服采用总线控制型伺服，IO 扩展机架也是通过 EtherCAT 总线与LC1200控制器的 CPU 模块连接。

如下图为典型的运动控制网络，其中 LC1200为控制主站，伺服轴、远程 IO 等为从站。EtherCAT 总线为实时总线，其第一个从站的时钟将作为整个网络的参考同步时钟，因此伺服应安装于 EtherCAT 总线网络的前端，即网络的 1# 从站必需为伺服；而 EtherCAT 远程模块（即 RTU-ETC）内部没有时钟单元，在需要运动控制的网络中，一般安装于网络的中端或后端。

【图3.1 运动控制网络】

MC 运动控制（Motion Control）的特点是控制器通过软件计算、以数字命令通过 EtherCAT 实时总线来控制伺服运行，利用 EtherCAT 总线的高速（100Mbps）、高频度（最快可以 1ms 通信一次）来进行交互，相比于传统的脉冲控制方式，运动控制能更及时准确的进行。由此带来的一些编程方法也与以往梯形图逻辑控制也不相同，需要使用包含更多底层功能的“功能块”来编程。

3.2 运动控制程序组成

LC1200控制器是基于多任务操作系统开发的控制器，系统运行的多个功能模块以多任务执行的方式，对于用户程序， 也可分为多个任务组成，根据用户设定的任务优先级，分别执行。

编写用户程序时，用户可根据应用系统中不同处理的任务类型和紧急程度，分成若干个程序组织单元组成，可将每个任务指定不同的执行触发条件，或相应的执行时间间隔（也称执行周期），这样可以让应用系统的控制响应达到最佳状

态。

3.2.1 LC-1200系列用户程序组成

控制器可采用多任务的执行模式，即可以“同时”执行几个任务，每个任务可以有若干个用户程序组织单元（简称POU），典型的构成举例如下图：

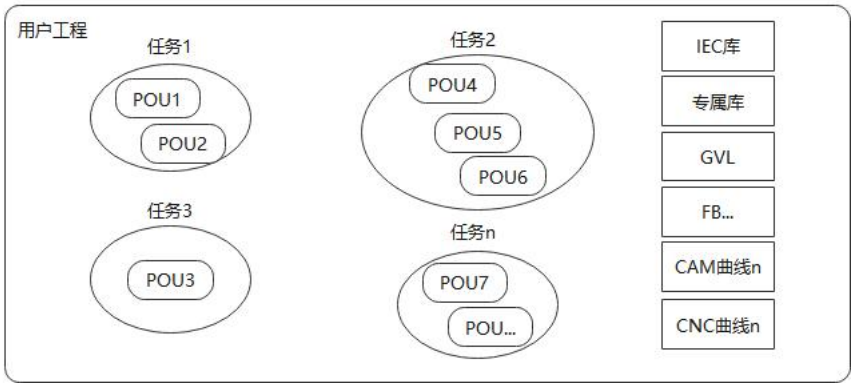


图3.2 用户工程构成举例

用户工程是由若干个 POU 组成，按照 POU 执行特性要求，分为若干个任务组，配置其执行特性，没有列入任务配置的 POU 将不会被执行的。

另外用户工程中，还有一些支撑用户程序的对象，如库函数、全局变量 GVL、功能块 FB、凸轮定义 CAM 曲线、多轴插补轨迹定义 CNC 曲线等等，作为用户程序的组成部分。

3.2.2 控制器中的任务类型

任务配置是将用户程序按执行要求，分成若干个任务组，每个任务组可以设置不同的执行触发条件、执行时间间隔、优先级等。LC1200控制器常见的任务有：EtherCAT 任务、主循环任务等，其中，运动控制相关的用户程序主体，是安排在 EtherCAT 任务下执行。

EtherCAT 任务在 LC1200 控制器中是一个最为重要的任务，运动控制功能的实时处理，都是在这个任务中完成的，它是一个执行时间间隔短、优先级最高的一个时钟中断型的任务，一旦满足时间条件，它可以无条件地中断其他的任务，开始执行 EtherCAT 任务，直到该任务配置下所有的 POU 执行完毕，才会退出。

每个任务中，都可以指定执行多个用户程序单元（即POU），这些POU会被

逐个依次序被执行，该次序就是在任务配置中的顺序，如下图：

图3.3 POU执行顺序

图中，三个 POU 的执行顺序依次是 PLC_PRG、POU1、POU2，有全局变量更新操作和判断的情况时， 需要注意安排合适的顺序。

图中还有 EtherCAT_Task 的任务，添加 EtherCAT_Master_SoftMotion 设备时会自动添加此任务，根据优先级执行，可以理解为进入 EtherCAT 任务时系统进行的总线通信任务默认处理，包括主站与所有从站的 PDO 发送接收、各伺服轴数据结构的更新处理等。

3.2.3 用户程序由多个POU组成的好处

不同执行周期的处理程序，应放在不同的 POU 中编写。比如按 EtherCAT 周期执行的POU、外部中断程序 POU、按 20ms 时间处理的程序 POU，就必需分成独立的 POU 来编写。

为提高程序的可读性，按不同控制工艺段、不同的操作对象、不同的物理结构部件等，分别用不同的 POU 来处理，每个 POU 分别命名为易于理解的名称；如同 C 语言编程， 将一个反复调用的处理程序做成一个独立的 POU，方便本工程调用；甚至还方便其他工程沿用。

多人合作编程时，每个编程人员各自编写调试自己负责的工艺段的 POU，最后合成为一个用户程序项目；CODESYS 编程软件支持 6 种编程语言，根据所需的处理逻辑类型不同，某种的语言可能更方便，而一般情况下，每个 POU 只能用一种编程语言来进行编写，一个项目中若需要同时使用多种编程语言的话，分为多个 POU 来编写也是一个比较好的对策。

3.2.4 用户程序如何同时做到逻辑控制与运动控制

应用系统的同步控制、轨迹控制，往往都有高实时性要求，而逻辑控制的及时性要求则相对较低，在 LC1200 控制器 用户程序中，可以将运动控制（MC）的 POU 放在 EtherCAT 任务周期中执行，而逻辑控制 POU 就放在普通任务周期中执行即可。若将特定的程序变量声明为全局变量，就可以在运动控制中，实现与逻辑控制的协调动作。

对于主要以伺服驱动器+马达为控制对象的单轴 MC 控制，需要有伺服的使能、原点回归、定位控制、速度控制、力矩控制、停止与复位；对于多轴同步 MC 控制的应用，如凸轮控制、轨迹插补控制等，控制器提供有对应的 MC 功能块来完成这些操作。因此，功能块是运动控制编程中常用的控制命令，就如建筑中的采用预制件代替砂石水泥，以提高施工效率一样。

用户程序可以根据应用系统的控制逻辑进行，控制功能块的运行触发、终止执行等，同 时可对功能块的执行状态、是否出错等进行判断；在 PLCopen 规范中，还引入了轴状态数 据结构，控制器系统为用户已经配置的每一个伺服轴建立了一个对应的数据结构，并自动在每个 EtherCAT 周期中及时对其状态进行更新，用户程序可以通过访问该数据结构的变量， 就可对伺服轴的运行状态进行监控，将状态变量作为逻辑控制的依据，这样就使得逻辑控制 与运动控制在一个用户程序中得以轻松实现。

3.3 编写用户程序的典型步骤【改到第5章首】

编写一个完整的具有MC运动控制功能的用户程序，一般需要经过以下几个步骤：

(1) 配置应用系统的硬件：根据所使用的主控制器、扩展模块、网络类型、伺服从站等，进行网络配置；

(2) 编写用户程序：根据所需实现的控制功能，将运动控制逻辑用一个POU编写，将普通逻辑控制用另一个POU编写（编程用户程序数据的储存宽度、使用范围，来自由定义变量，可以与硬件配置无关）；

(3) 关联变量：将系统组成中的各硬件端口对应的输入端口变量（I）、输出端口变量（Q）与永无程序中的变量进行关联；

(4) 配置伺服驱动器参数：根据硬件件配置中的伺服名称，伺服的运行模式，来配置 SDO、PDO 的对象，保证用户程序的 MC 功能块与伺服之间所需的通讯对象都填在配置表中；

(5) 配置伺服电机参数：要准确填写伺服电机的编码器分辨率、机械结构的传动比、轴运动范围特点等，使得控制对象的位移指令与实际位移的准确对应；

(6) 任务等级、种类等安排设置：配置网络通信的同步周期，根据各任务的实时性要求，配置用户程序单元的触发条件、执行周期与优先级等（例如：将运动控制功能的POU放在EtherCAT任务

中执行，周期可设为 2ms，优先级为 0；将普通逻辑控制的POU放在普通任务下，周期可设为10ms，优先级为 16）；

(7) 在CODESYS编程环境下，登陆到PLC控制器，下载用户程序，仿真调试修改，最终准确无误运行。

注：用户程序编程详细示例请参照【第5章 一个简单的用户程序】。

3.3.1 用户系统的配置

运行CODESYS编程软件（推荐使用V3.5 SP18版本或更高级别版本），新建一个用户工程。选定标准工程新建后，会弹出向导，选择要使用的设备和编程语言。

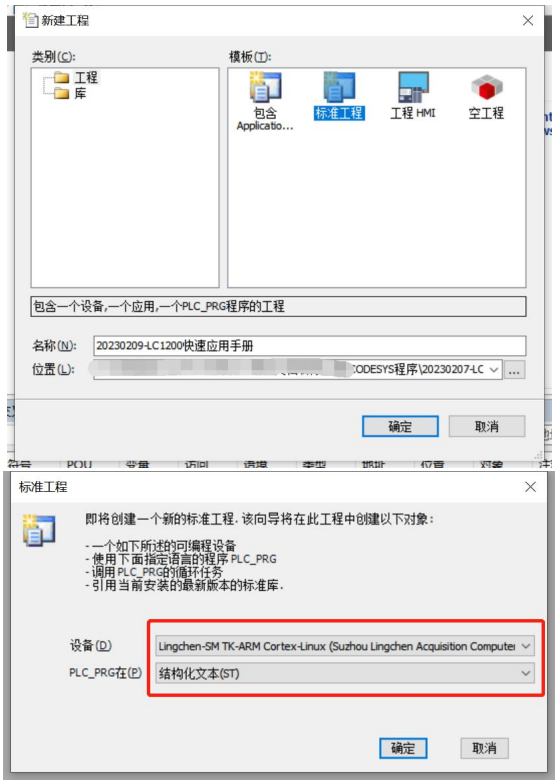


图5-1 设备选择与编程语言选择

倘若选定后，想更换设备，也可以在CODESYS软件的主画面，通过右键左侧树形窗口的“Device”项，即进入设备添加界面，根据实际应用系统使用的模块型号、安装顺序，依次从设备库的界面中，双击选中，或者拖拽摆放到“Device”下，若要删除某个模块，选中该模块后，按Del键可以删除。

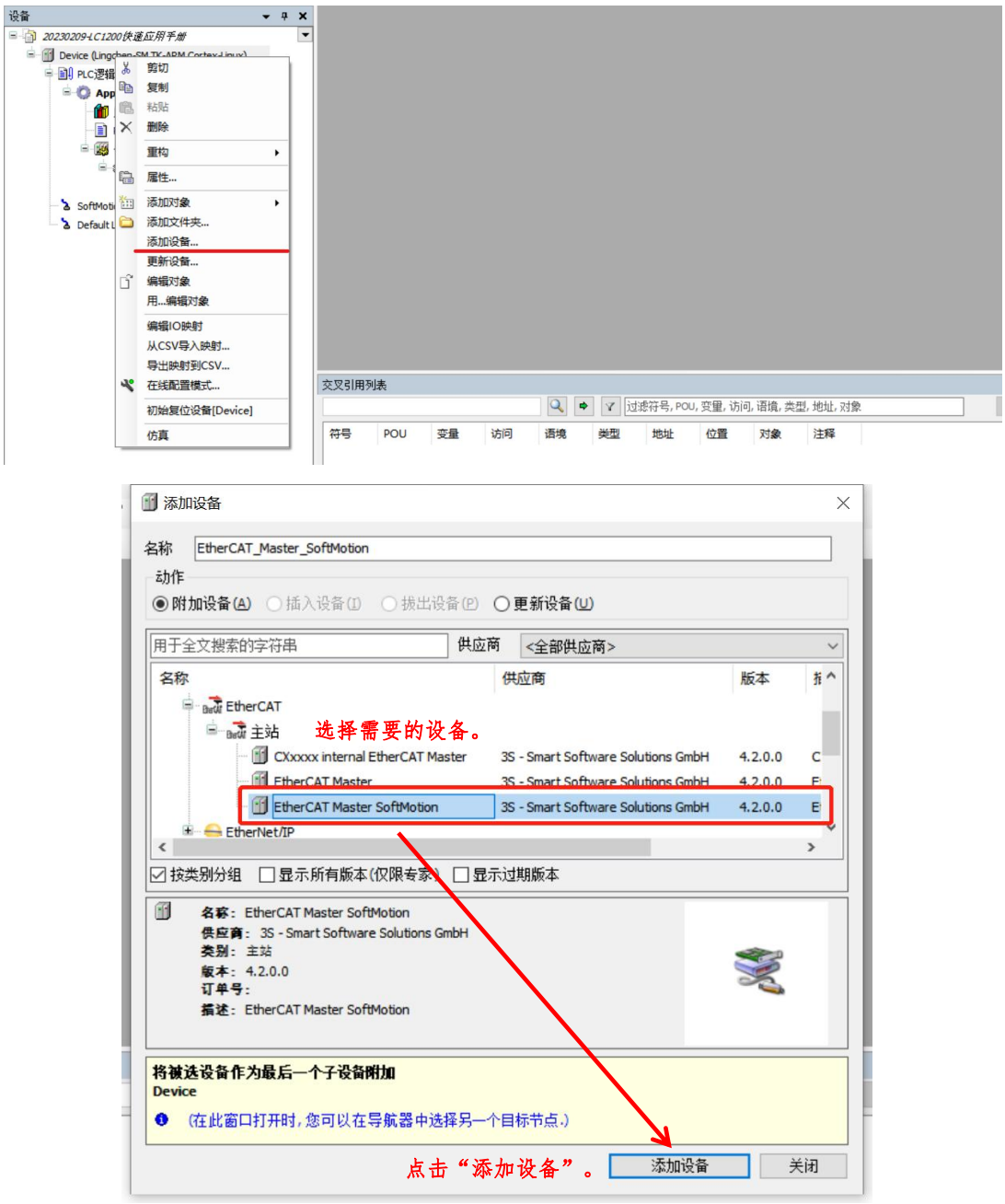
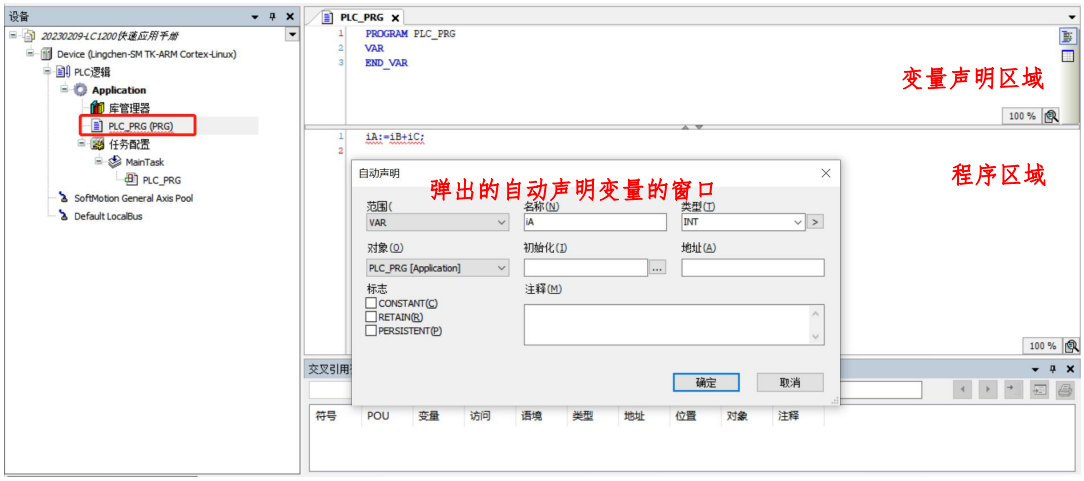


图5-2 配置画面

3.3.2 用户程序的编写

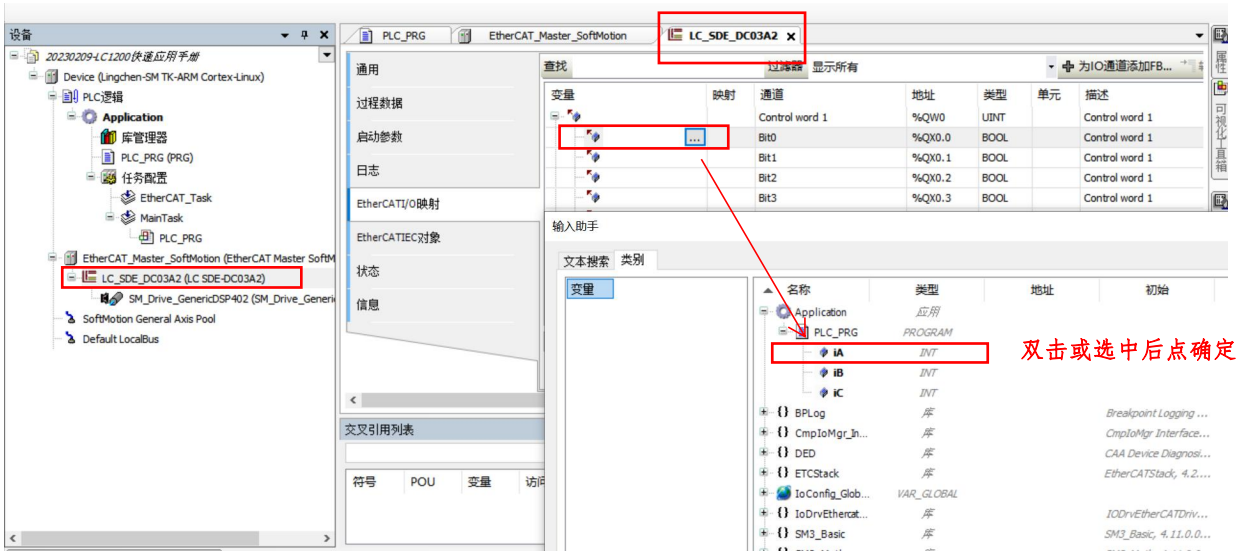
双击左侧树形窗口的“PLC_PRG(PRG)”项，即可打开用户编程界面，编程语言为 ST（新建工程时已选择），如下图所示。与 C 语言编程相似，每个变量需要声明后才能使用，如果先直接写程序语句，回车时，编程环境会自动弹出声明框，让用户填写，一旦点击“确定”，变量声明窗口会自动增加该变量的声明语句，简化了编程。



5-3 程序编写与变量声明

3.3.3 用户程序变量与端口关联

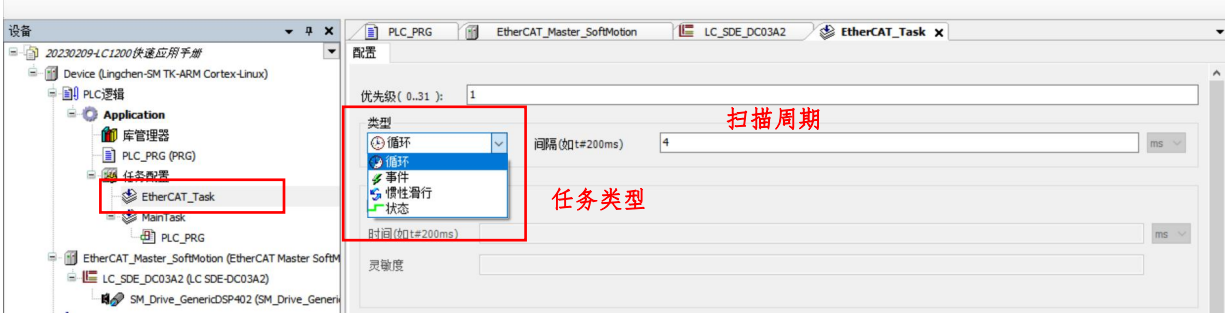
在本地总线配置页面，将所需要关联的硬件端口，与用户程序中的变量进行关联，如下图，将“iA”的变量值，在步进驱动器模块的EtherCAT I/O映射的控制字Bit0位输出，配置步骤如下：



5-4 变量与端口声明

3.3.4 用户程序的执行方式和运行周期

程序完成后，需要将程序添加到任务中，并且对任务进行配置，默认为 4ms 执行一次， 如果要改为其他的执行方式，如反复执行，定时执行、执行周期等等，可以分别设置，如下图：

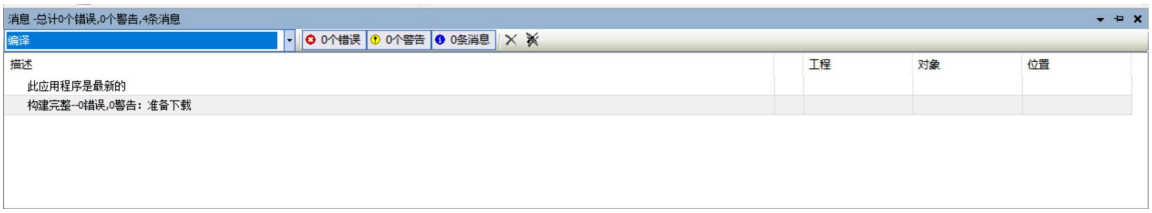


5-5 程序执行方式与运行周期配置

3.3.5 程序编译及登陆下载

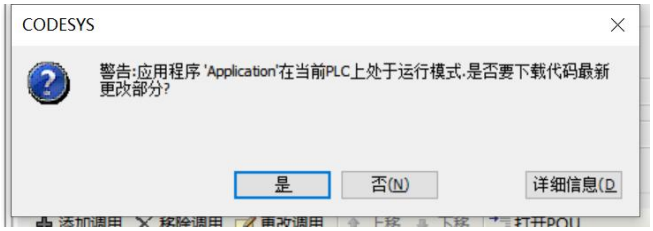
程序编写完成后，点击编译，生成用户应用程序，查看是否有错，若有错，点击错误信息行，可定位到用户程序的报错点，方便修改，直到错误全部排除。

相关编译信息会在如下的编译信息框中显示：



5-6 程序编译信息框

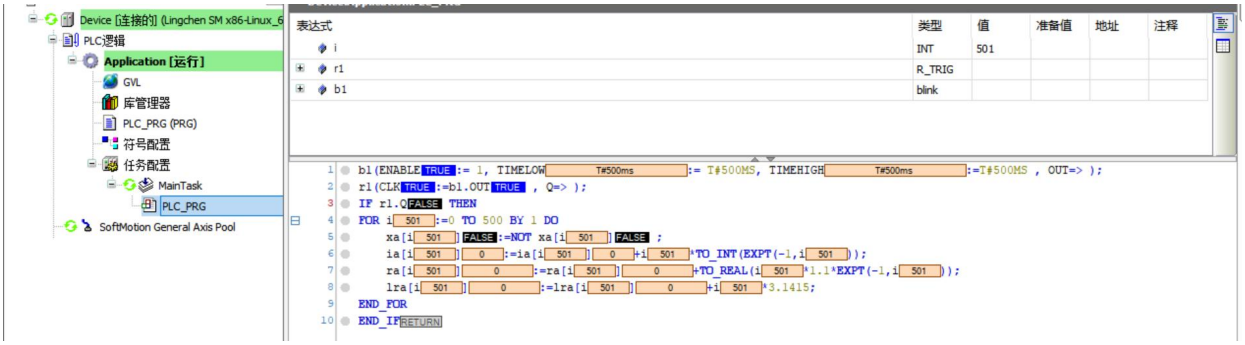
编译无误后，点击“登录”，或者使用快捷键“Alt+F8”，弹出如下对话框，



5-7 登录下载

选择“是”。下载完成后，点击“运行”，运行并且调试程序。

下图为正在运行的用户程序监控画面：



5-8 程序监控画面

第 4 章 LC1200运动控制程序执行机制

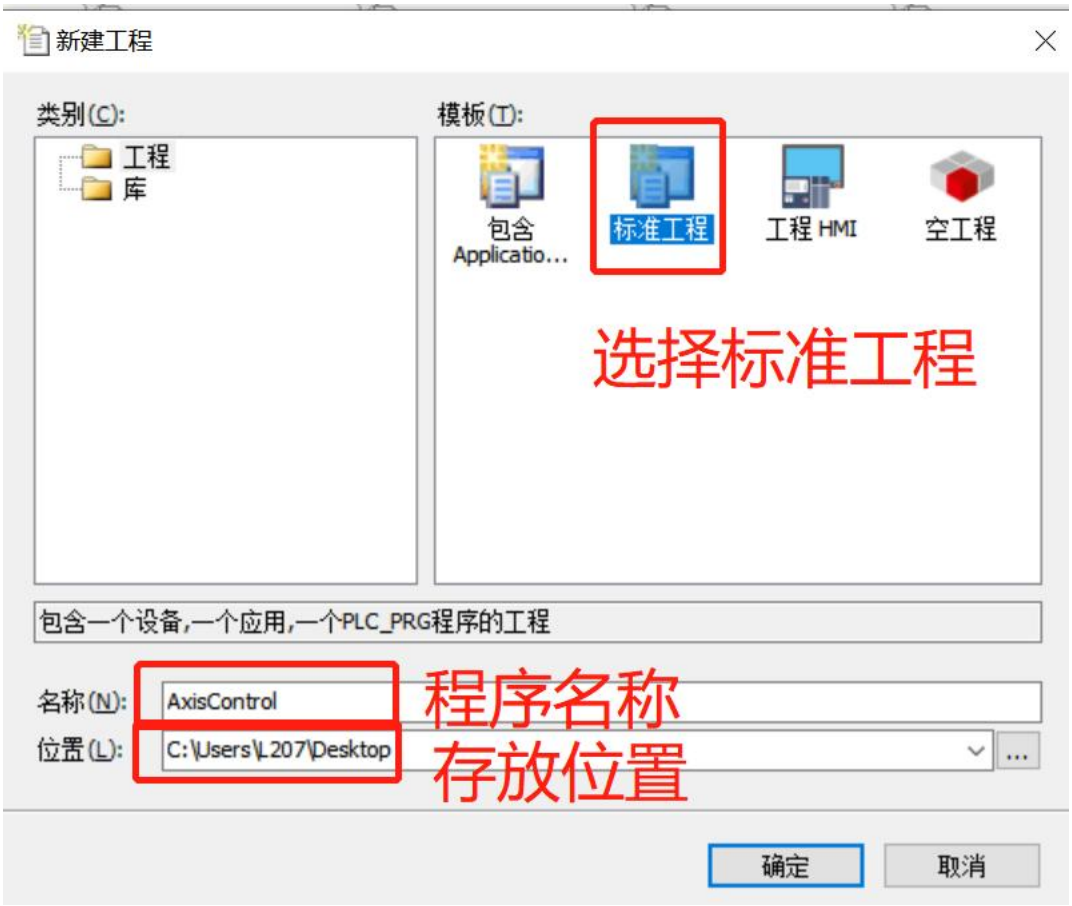
为了让用户更快熟悉软件并且动手编程，本章节将会示范，如何用 CODESYS 建立一个简单的 EtherCAT 总线工程，使用凌臣运动控制器，通过 EtherCAT 总线，控制 总线伺服，完成使能、位置模式运行以及停止等动作。

注意：工程非标准模板，内容仅供参考。

4.1 建立一个工程并且下载调试

4.1.1 新建标准工程

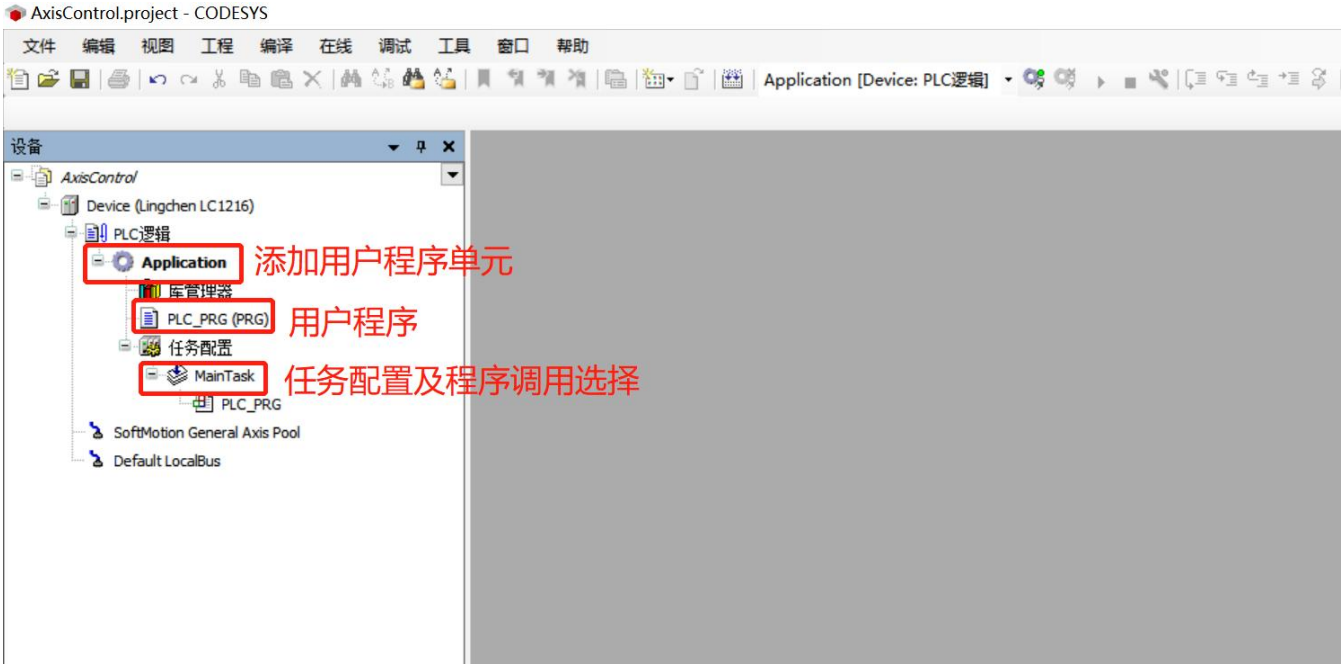
1）软件打开后，点击文件-新建工程，弹出对话框，选择工程，工程类型选择标准工程，工程路径自行选择，工程命名，最后点击“OK”



2）进入标准工程界面，用户可以选择该工程的设备类型和编程语言。如下图：

设备：选择主模块的机型，选择Lingchen LC1216.(嵌入式平台控制器，需要先安装 XML 描述文件： LC1216.devdesc.xml， 安装方法请参照安装设备描述文件)， 或 CODESYS Softmotion RTE V3 x64（实时控制的软平台控制器）

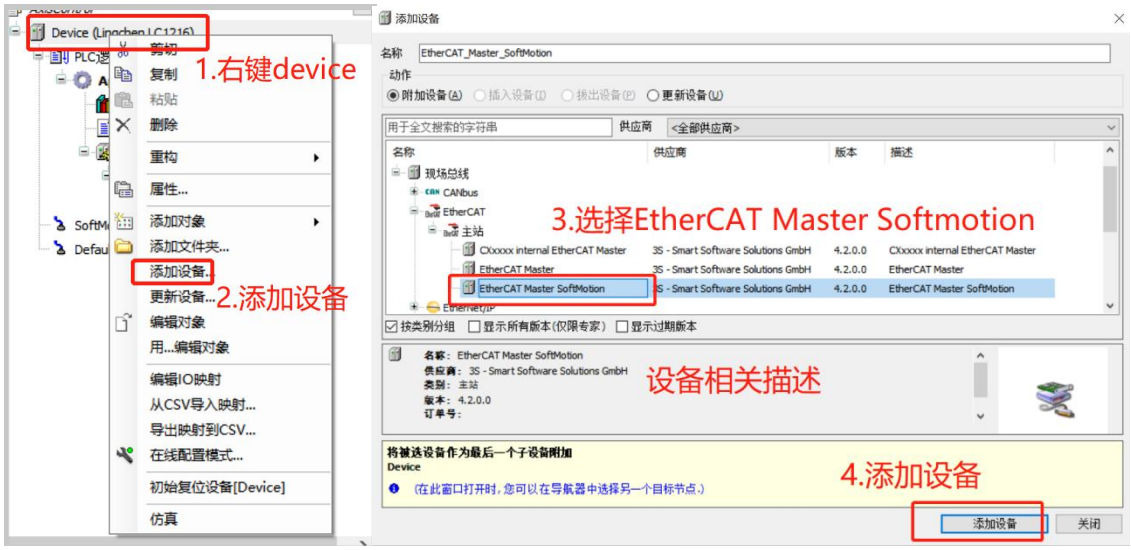
编程语言：ST，用户也可选用其他编程语言。选定后进入工程后仍可以修改。 点击“OK”，建立完成后如图



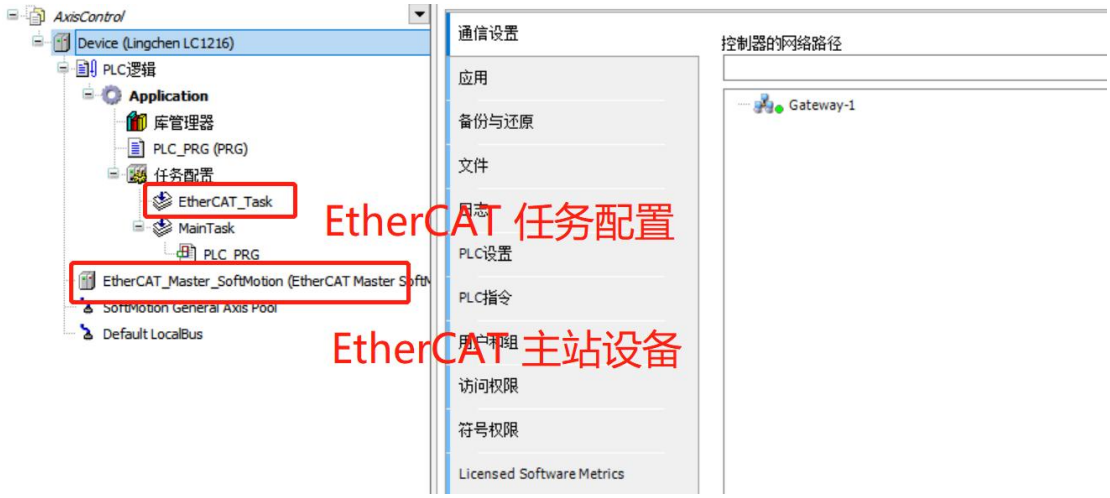
4.1.2 系统配置与参数设定

添加 EtherCAT_Master_Softmotion

EtherCAT Master Softmotion 是带实时运动控制的 EtherCAT 主站模块。具体添加方法：右键“Device→添加设备→线程总线→EtherCAT→EtherCAT Master Softmotion→OK”， 添加 EtherCAT 主站



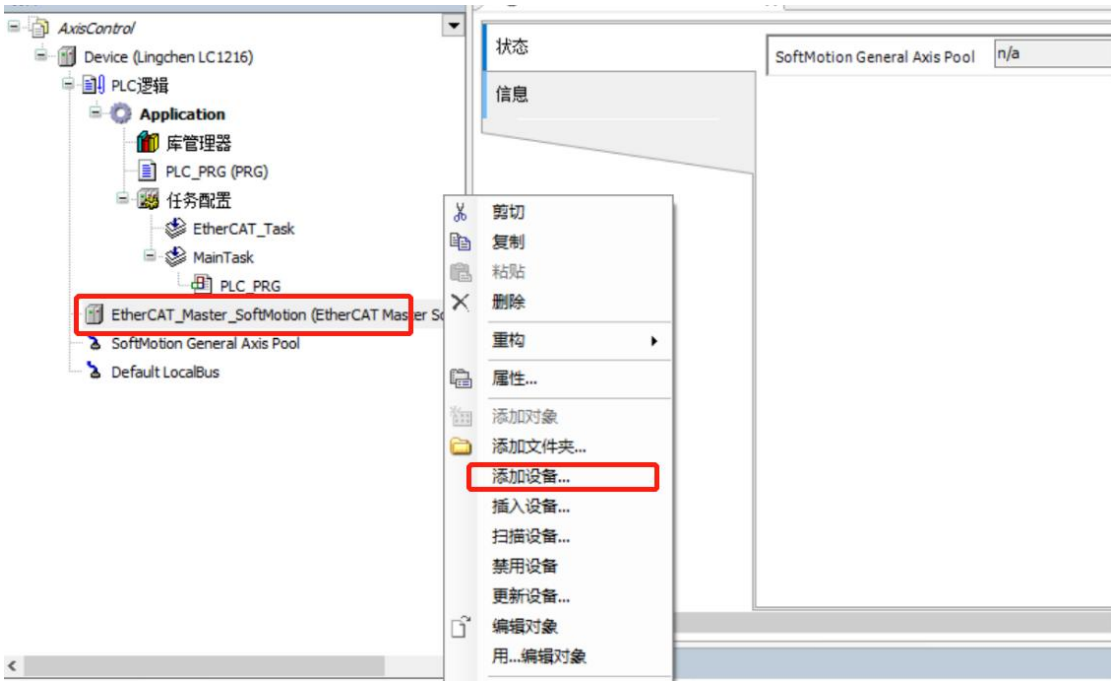
添加后如图，系统会同时分配一个 EtherCAT_Task，可配置 EtherCAT 任务相关参数。



添加总线伺服

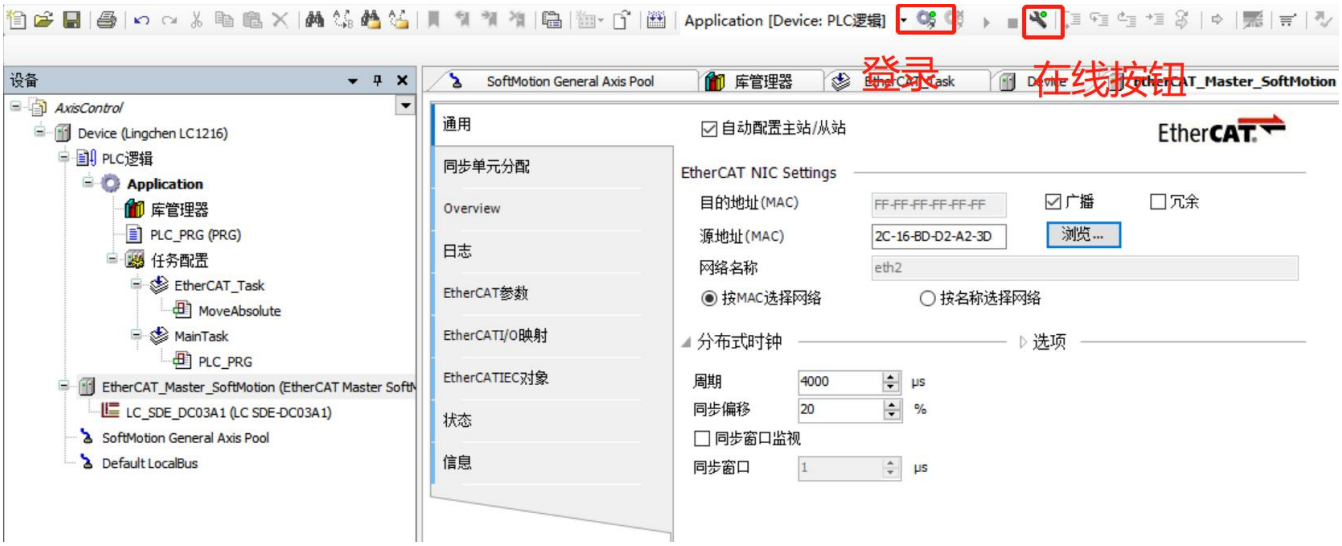
添加了主站设备后，在该主站下方添加从站设备，此处添加设备为“LC_SDE_DC03A1（凌臣总线步进）。添加前必须先安装设备描述文件（.XML）安装流程参考 2.2.5蠹设备描述文件。具体添加方法如下：

1）若是在未连接主站的离线状态下，可通过右键“EtherCAT_Master_Softmotion→添加设备”，在设备弹窗中找到对应厂商以及设备型号：LC_SDE_DC03A1，点击确认，添加设备。

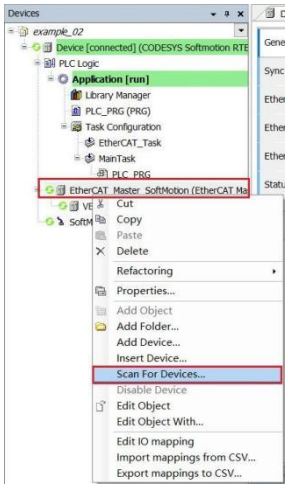


如果是在登录主站的在线状态下，首先选择Ethercat口网口，固定为***eth2***;

点击在线或登录按钮，连接设备；



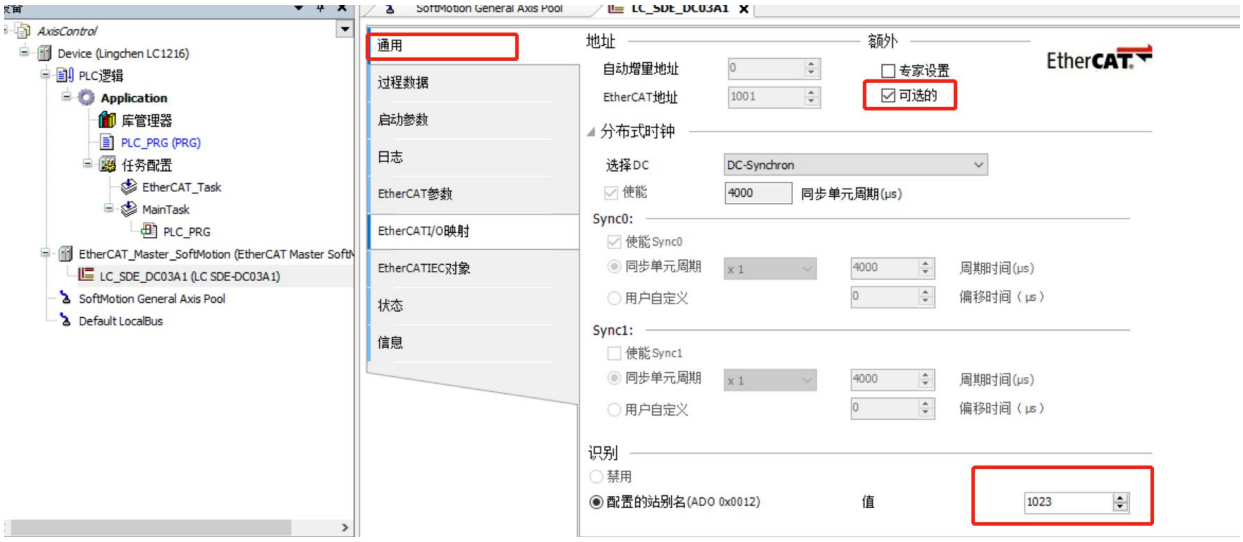
右键“EtherCAT_Master_Softmotion→扫描设备”，通过扫描添加。



注意：设备的节点地址，默认是自动分配，即离主机距离从近到远来分配节点地址，本例不做修改，按默认设置。

若需要手动分配节点地址，可参考如下方法，以LC_SDE_DC03A1举例：

a） 按照下图方法分配地址：从站勾选附加处的“可选的”，然后配置的站别名出填上站号（1-65535）

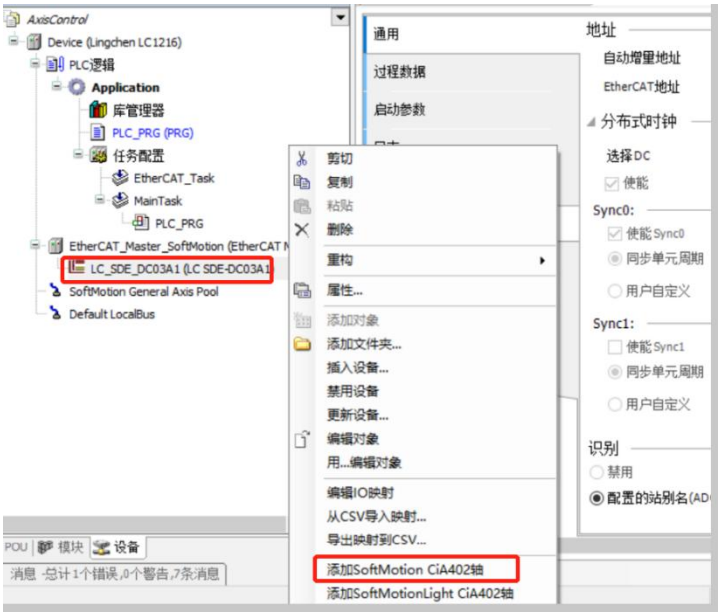


b) 设置伺服站号为 1023，复位伺服或重新上电，然后登录设备下载程序，如果第一次没有连接成功，复位重新运行即可。

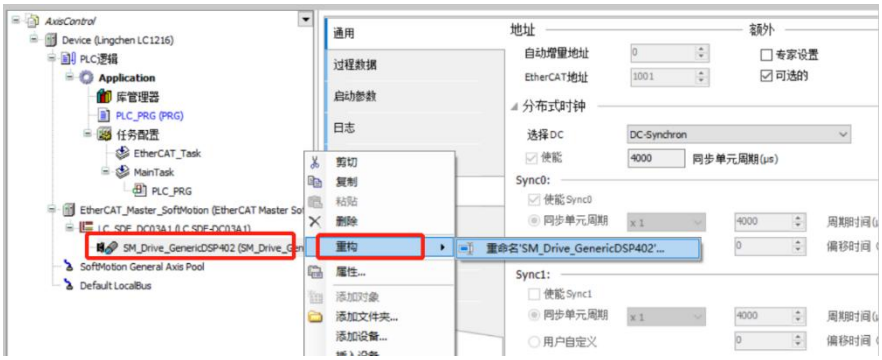
c）按照上面设置后只要从站的别名与后台工程配置的别名一样，不管顺序如何都能运行。

添加 CiA402 Axis

1）设备与轴关联运行，添加方法：右键“LC_SDE_DC03A1→添加SoftMotion CiA402 轴”，添加运动控制轴，如下图。



添加后如下图，为了方便编程，重命名轴为“Axis1”



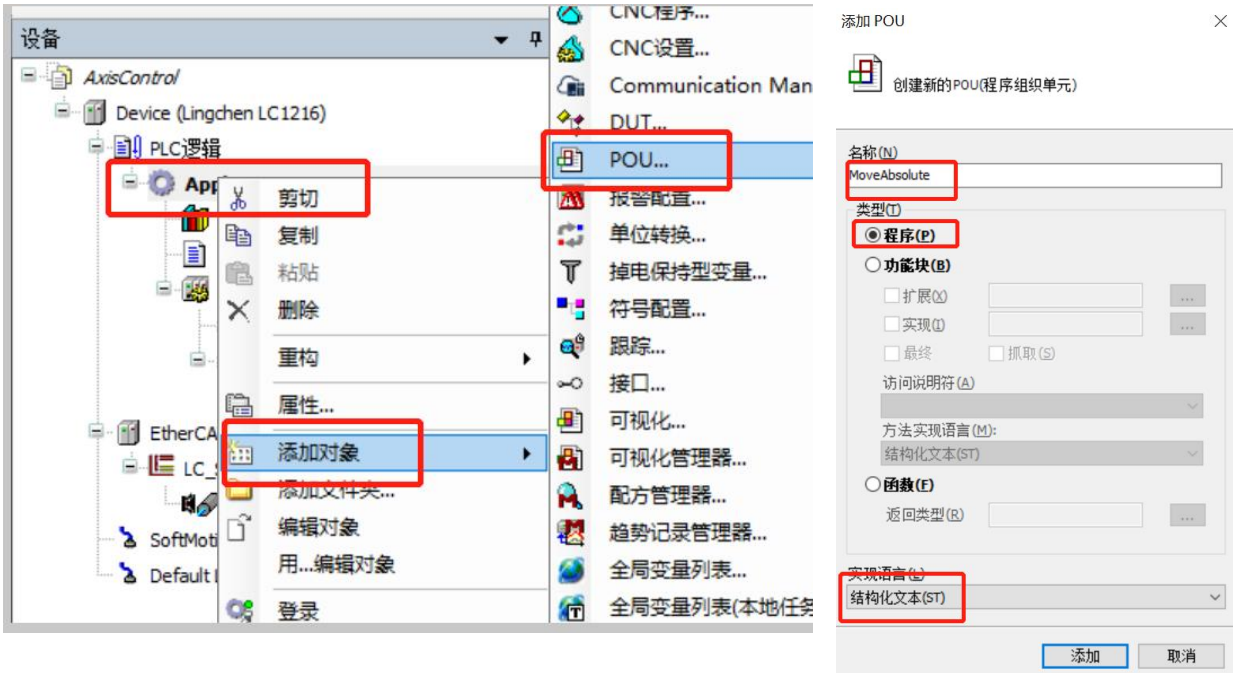
2）设置控制相关参数，双击 Axis1，打开参数配置页面，设置齿轮比

4.1.3 用户控制程序编写

此处编写一个程序，使控制器控制伺服电机执行绝对位置指令，做往返运动。

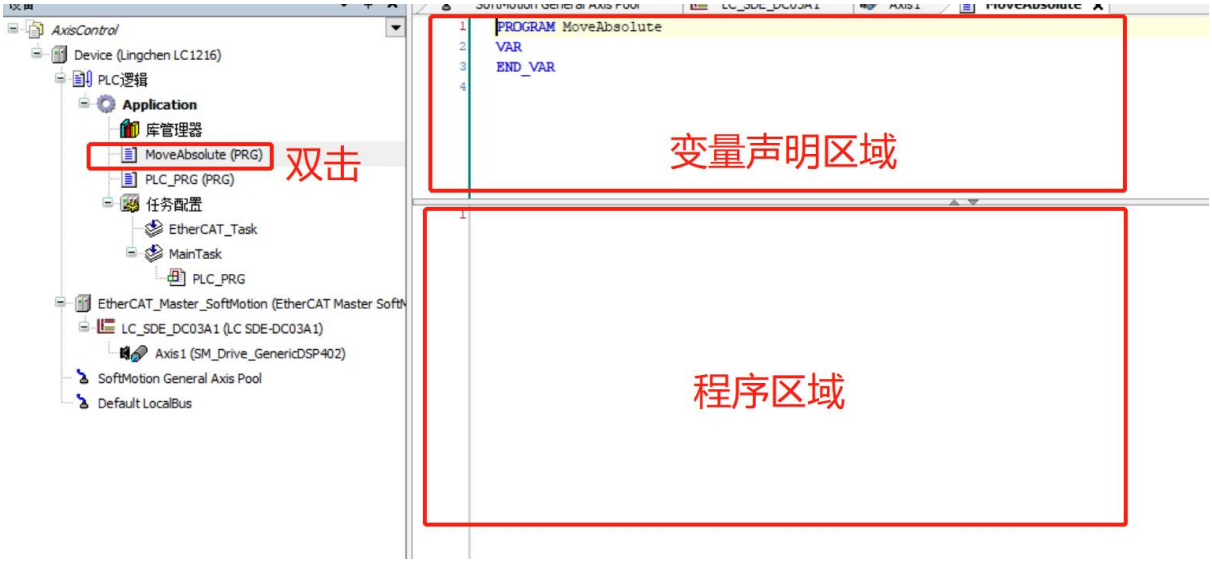
建立对象

如下图，右键鼠标“Application→添加对象→POU”，在弹窗中为新建的 POU 命名为“MoveAbsolute”，类型选择“程序”，编程语言选择“结构化文本（ST）”，点击“OK”，完成添加。



打开编程环境

双击打开“MoveAbsolute”，如下图，编程界面包括变量声明区域和编程区域。

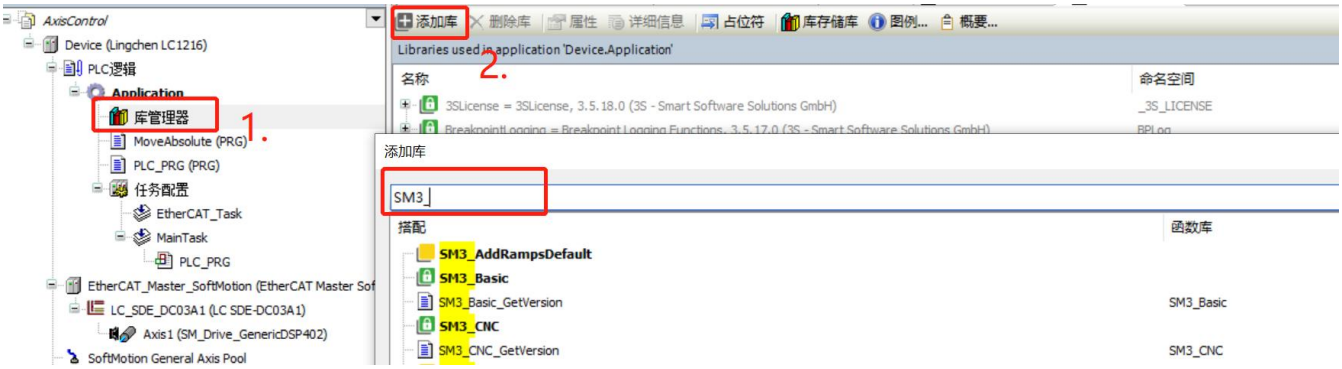


定义变量

在变量声明区域添加变量，变量声明代码如下。

```
PROGRAM MoveAbsolute
VAR
iStatus:INT;
Power:MC_Power;    //使能模块
ActPos:LREAL; //实际位置值
MoveAbsolute:MC_MoveAbsolute; //绝对位移模块
p:REAL:=1; //位移值
END_VAR
:
```

此处需要注意，库管理器库中是否添加了库“SM3_Basic”，一般都是默认添加，如若未添加则需要手动通过双击“库管理器→添加库”，找到库“SM3_Basic”然后选择添加，也可以通过此方法添加更多的库。



程序编写

在编程区域添加程序如下。（程序功能：程序执行时，立即使能伺服，等伺服使能成功后，控制电机在位置 P 与起点 0 之间做往返运动。）

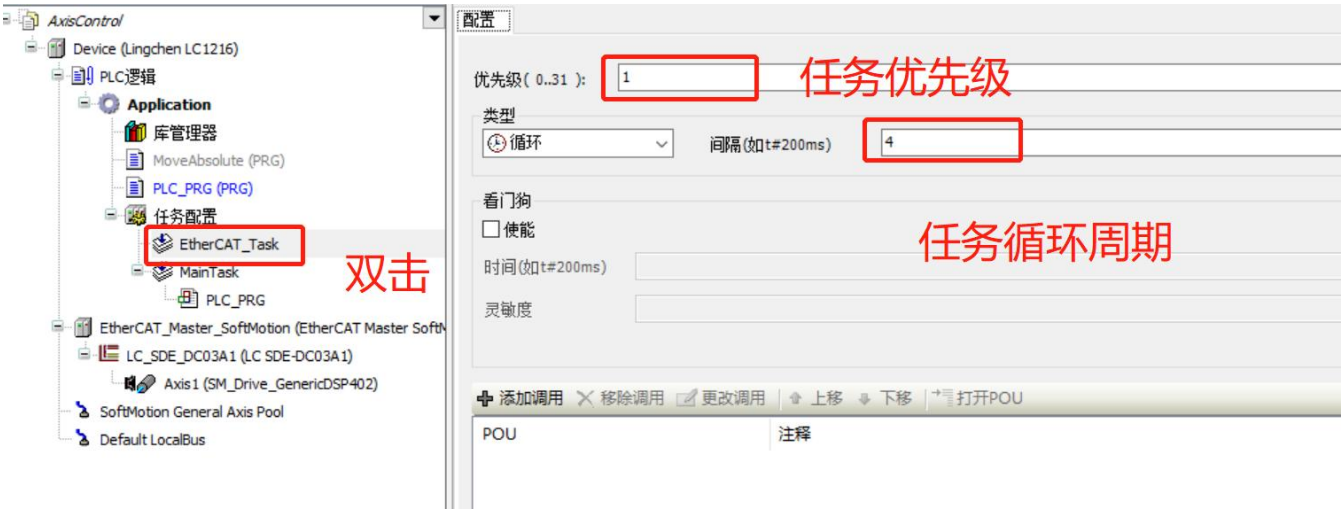
```
CASE iStatus OF
0:
Power(Axis:=Axis1, Enable:=TRUE , bRegulatorOn:=TRUE, bDriveStart:=TRUE );
IF Power.Status THEN iStatus:=iStatus+1;
END_IF
1:      //走绝对位移，运行到P 处
MoveAbsolute(Axis:=Axis1, Execute:=TRUE, Position:= 0, Velocity:=100 , Acceleration:= 100, Deceleration:=100 );
IF MoveAbsolute.Done THEN MoveAbsolute(Axis:=Axis1, Execute:= FALSE); iStatus:=iStatus+1;
END_IF
2://走绝对位移，运行回到 0 处
MoveAbsolute(Axis:=Axis1, Execute:=TRUE, Position:= p, Velocity:=100 , Acceleration:= 100, Deceleration:=100 );
IF MoveAbsolute.Done THEN
MoveAbsolute(Axis:=Axis1, Execute:= FALSE);
iStatus:=1; END_IF
END_CASE

ActPos:= Axis1.fActPosition;
```

程序编写完成后，点击编译  ，确认编写无误。

4.1.4 总线与任务周期

总线任务周期

在添加 EtherCAT Master SoftMotion 的时候，工程会自动添加总线任务 EtherCAT_Task，设置总线执行方式和循环周期以及任务优先级（0~31，0 为最高级）此处 EtherCAT_Task优先级设置为 1，其它任务，例如 Main_Task 优先级设置为 1~31。

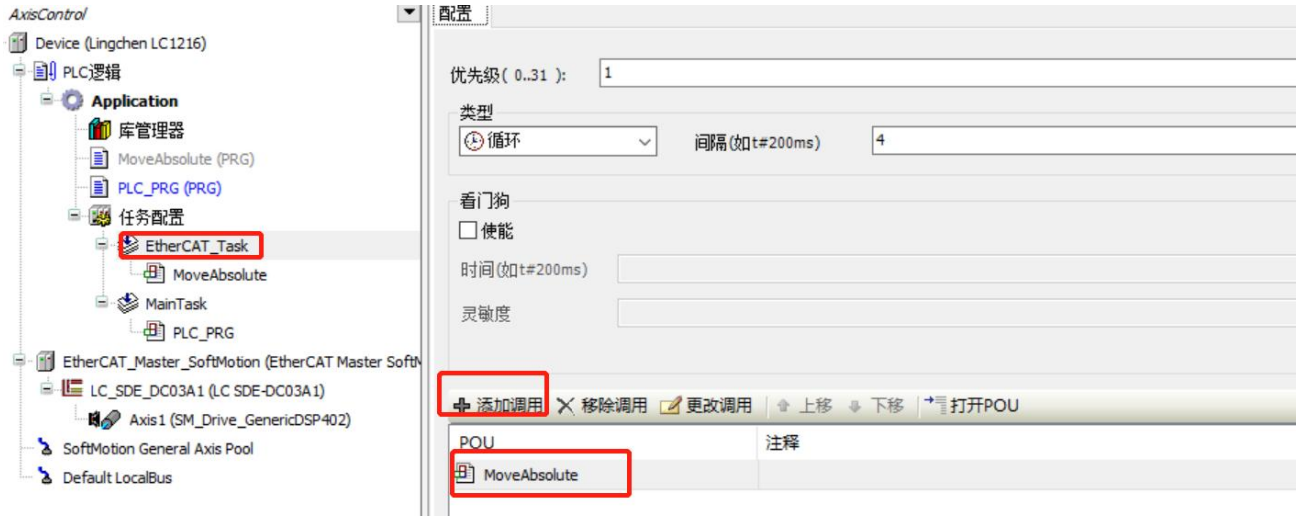


程序任务周期

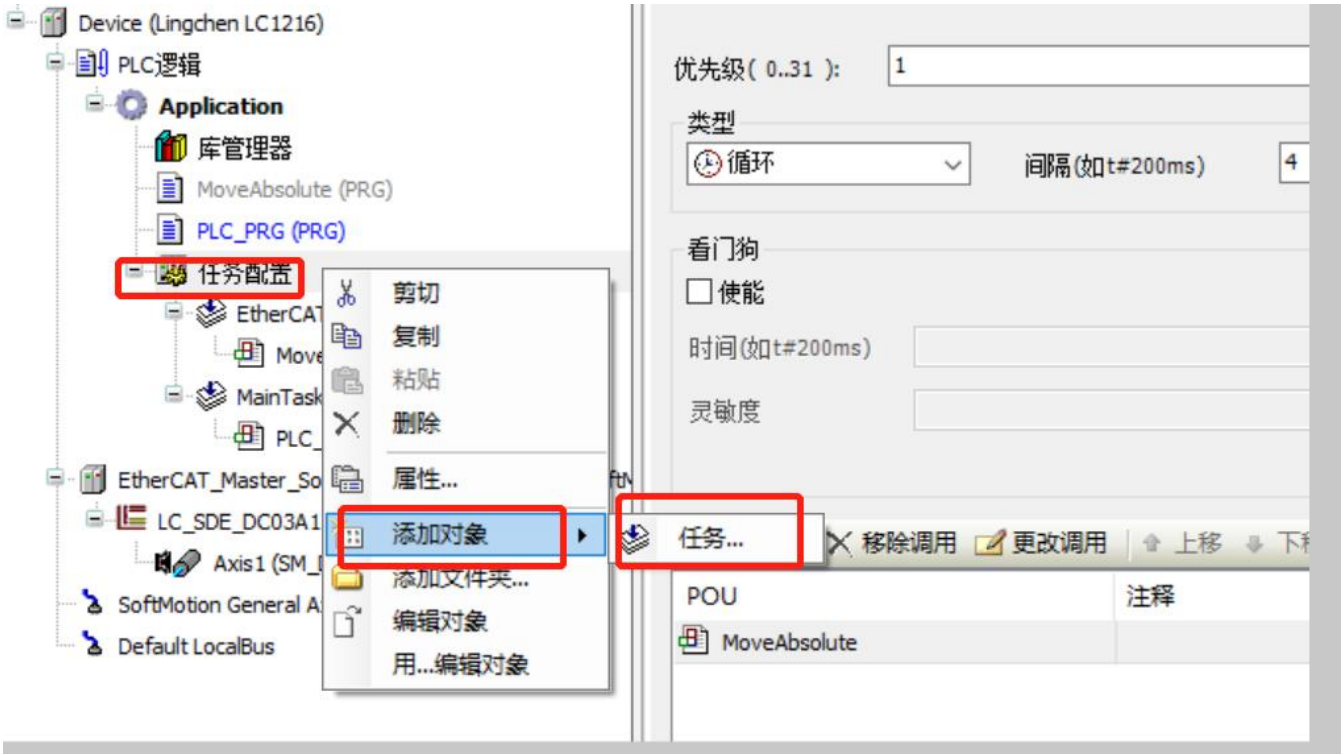
程序编写完成后，需要将程序添加到任务中，并且对任务进行配置。

与运动相关的 POU 建议添加到 EtherCAT_Task 中，与逻辑或计算相关的 POU 则建议添加到其他 Task 中（例如 Main_Task，建立工程时已经默认建立了一个程序“PLC PRG”和一个任务“Main Task”，并且已经把“PLC PRG”添加到“Main Task”中。）

新建的 POU 对象“MoveAbsolute”需要手动程序添加到 EtherCAT_Task 任务中，添加方法如下，双击“EtherCAT_Task→添加调用”，选择“MoveAbsolute”，点击“OK”。



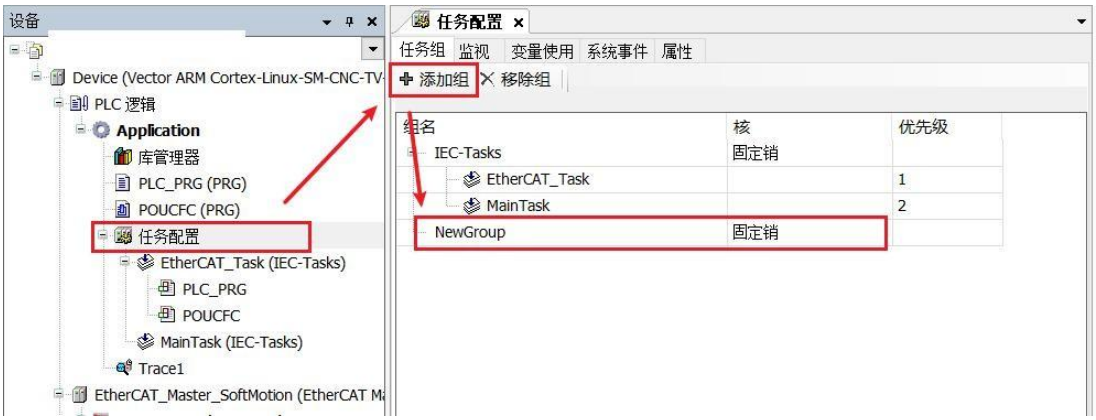
除了默认任务外，也可自行添加新任务，添加方法如下，鼠标右键“Task Configuration”， 选择“添加任务→Task”，即可添加新任务，双击任务即可对任务进行配置。



4.1.5 任务分核【删掉】

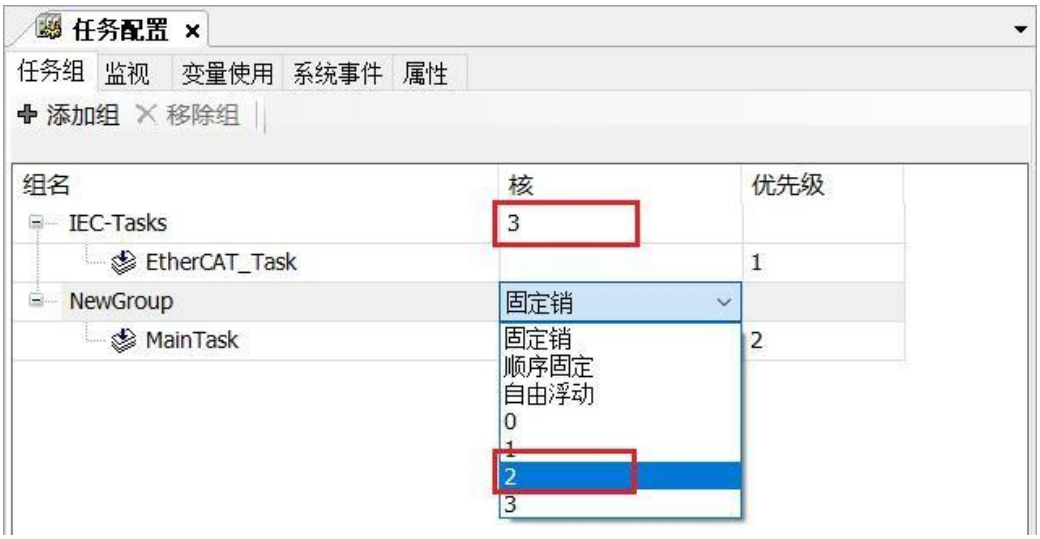
LC1200控制器采用四核芯设计，为了总线更平稳运行，可以对总线任务进行分核。步骤如下

（1） 打开“任务配置”，点击添加组，添加“NewGroup”



(2) 将其他任务添加到新建组中，EtherCAT_Task 单独一组

(3) EtherCAT_Task 分配到第 3 核，其他任务分配到第 2 核，保证 EtherCAT 任务稳定运行



注意：

Modbus 设备设置任务时，不能添加到 EtherCAT_Task 中，需要添加到其他任务中。

4.1.6 登录设备

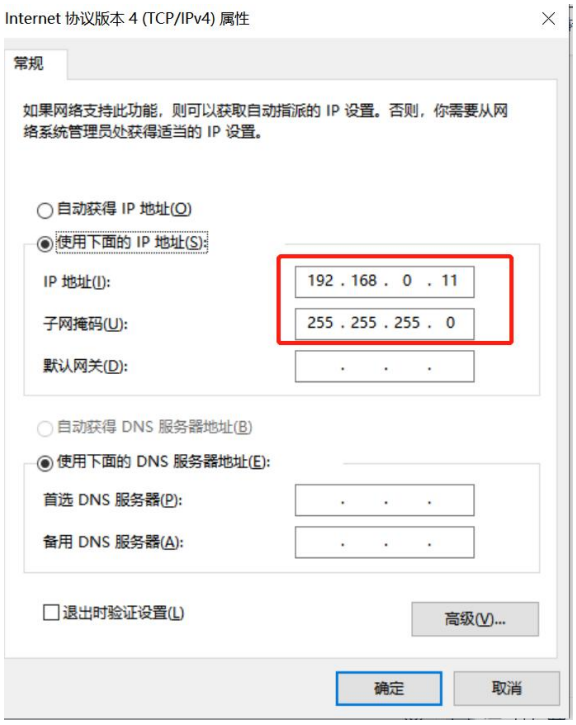
连接控制器

在 PC 上运行 CODESYS 的环境，与控制器建立通讯联系，可以进行用户程序的下

载、启停和监控用户程序的运行、进行参数的查看或修改操作，等等。

目前可通过 LAN 局域网络登录控制器，PC 电脑与控制器之间可以通过网线进行 1 对 1 的直接连接；也可以通过路由器、集线器进行联机，这种情况下，可以一台 PC 与多台 LC1200控制器联机，也可以多台 PC 访问同一个LC1200控制器。

PC 电脑与控制器两者的 IP 地址默认必需是同一个网段，才能登录控制器，否则 CODESYS 中将无法扫描到控制器。控制器的出厂默认 IP 地址为 192.168.0.99，若 PC 机的 IP 地址为 192.168.0.xxx,(这里 xxx 表示 1~254 范围，但不要与控制器 的 IP 末尾地址相同）那么 CODESYS 就可以扫描到控制器，并可以与之交互数据，进行用户程序下载、运行监控等。若控制器的 IP 被人为修改过，其地址不在 PC 所在的 IP 地址网段，PC中无法访问，可以将控制器的 IP 地址恢复为出厂默认 IP 地址 192.168.0.99，再将 PC 本机的地址修改为 192.168.0xxx，与之建立 1 对 1 联机后，可将 LC1200控制器的地址修改为希望的 IP 网段地址。



扫描网络

双击工程树中的“设备”，弹出以下界面。在该画面上，鼠标点击“Gateway-1→扫描网络”按钮，弹出如下界面，扫描到了控制器，在窗口左侧点击其名称，在窗口右边可以看到其简介信息，点击OK，即可连接设备：

在登录后可以根据自己的需要修改设备名，改为一个便于辨别的设备名，可以方便的识 别，

在有多台控制器的应用场合，很有帮助。

设置总线控制网口

双击“EtherCAT_Master_SoftMotion”，设置 EtherCAT 网卡，如图，点击“浏览”，在弹窗中选 EtherCAT 网卡名称（连接伺服端网口网口固定为eth2。

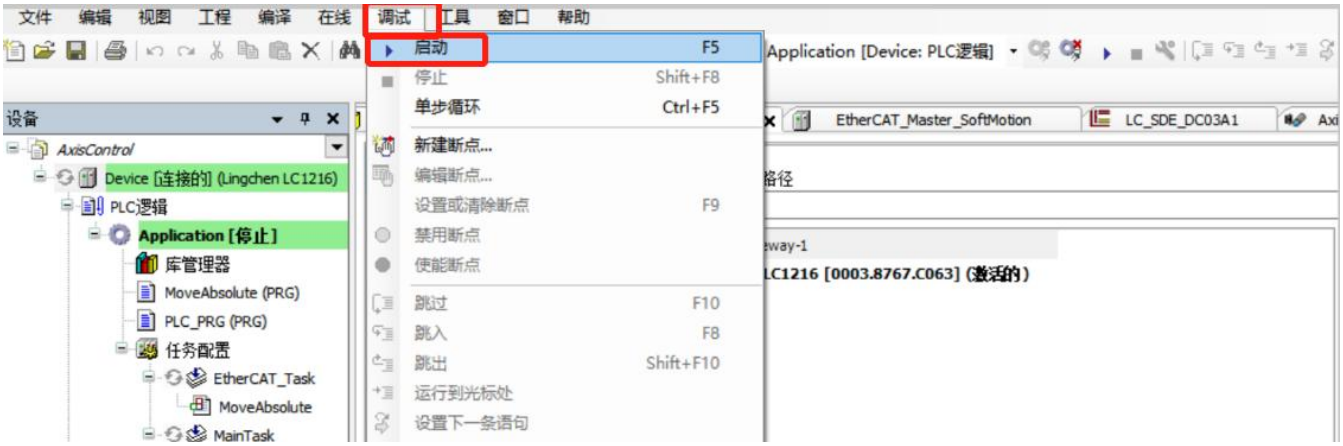
登录设备

点击“编译”，编译排错无误后，再点击“登录”

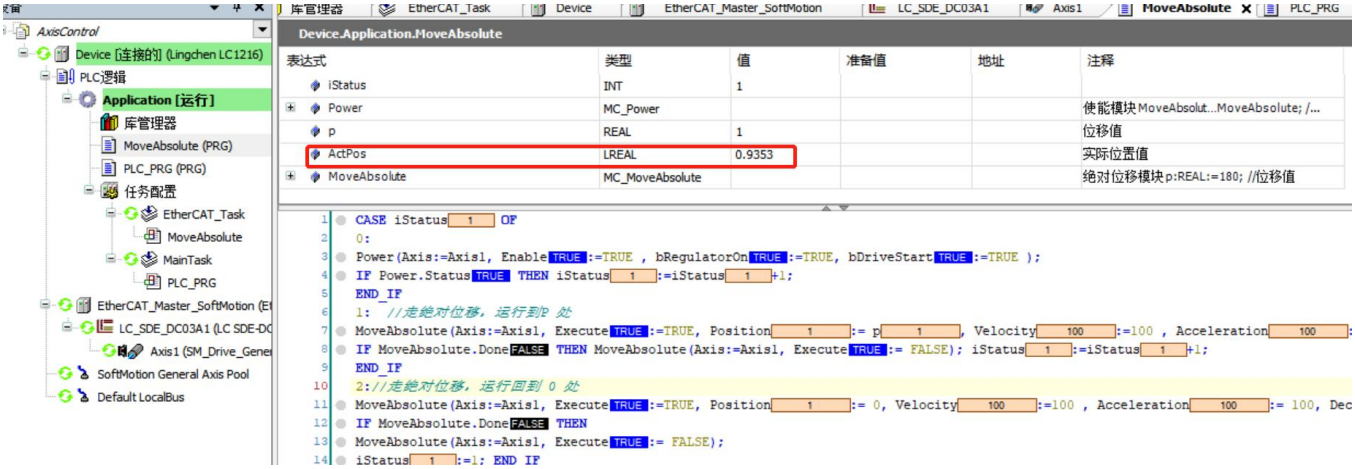
弹出对话框，选择“确定”，下载程序到控制器中。

4.1.7 启动调试

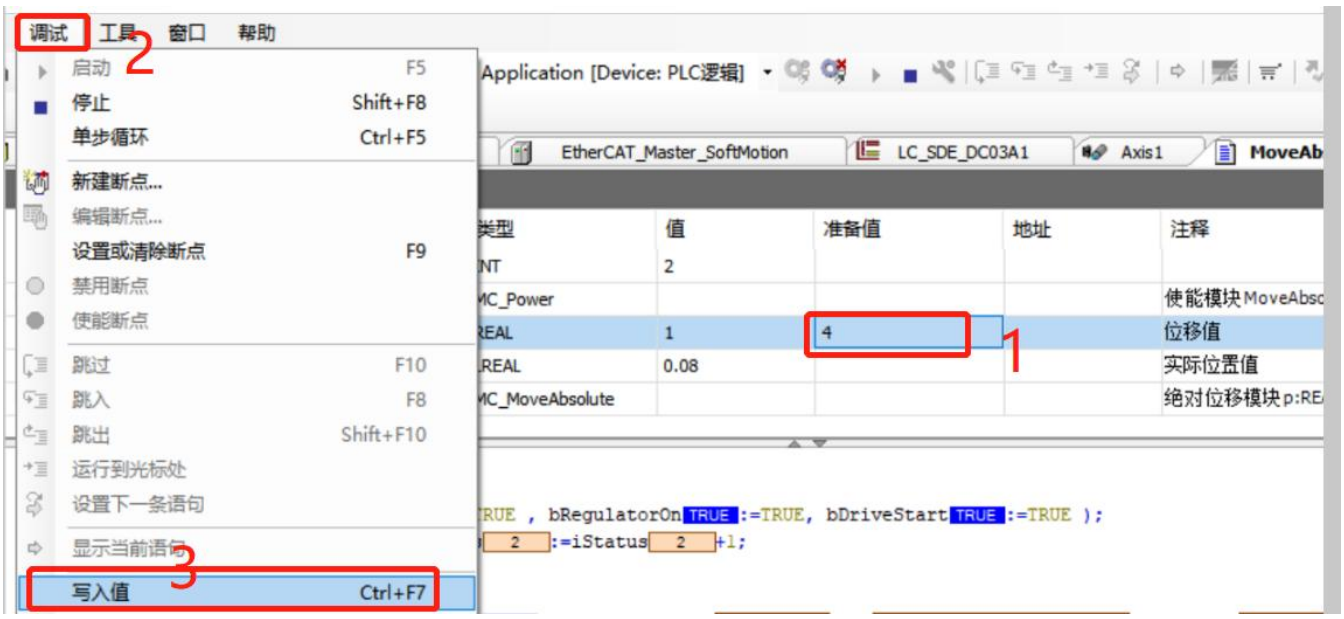
登录成功后，选择“Debug→Start”，控制器启动运行。



打开“MoveAbsolute”，程序运行如下图所示，程序执行伺服使能后，可以电机在位置 p 与起点位置0之间做往返运动。



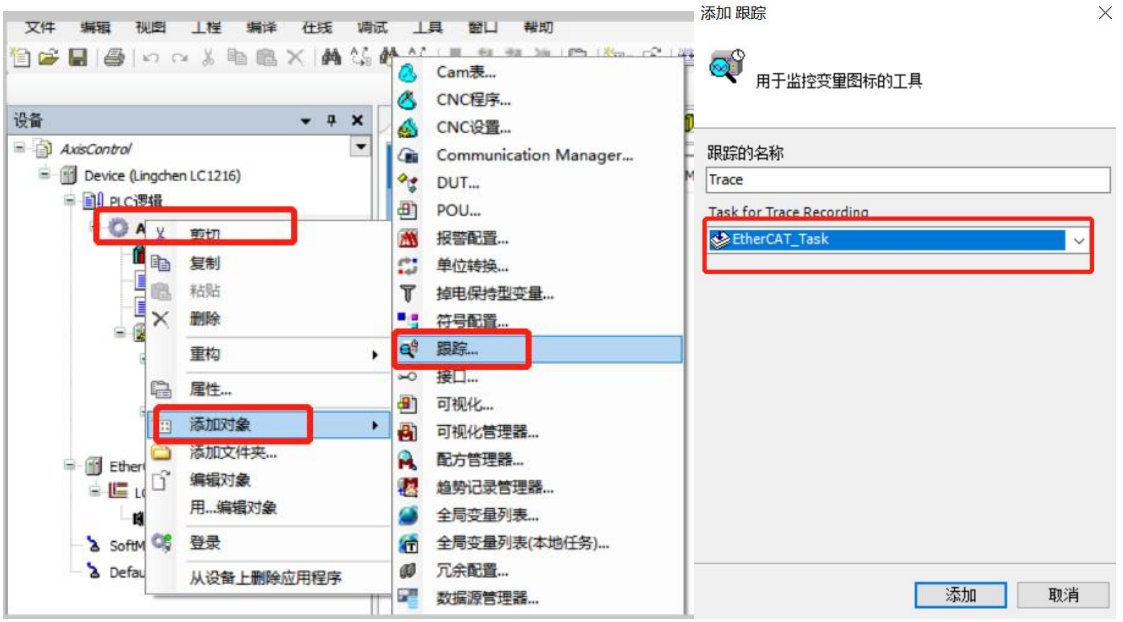
在线修改位置 P 的值：单击变量“P”的预设值“准备值”，使其为然后输入数值“10”，然后选择“调试→写入值”或者快捷键“Ctrl+F7”将值写入到“值”中，即可在线修改变量“P”的值。



4.1.8 添加Trace 跟踪

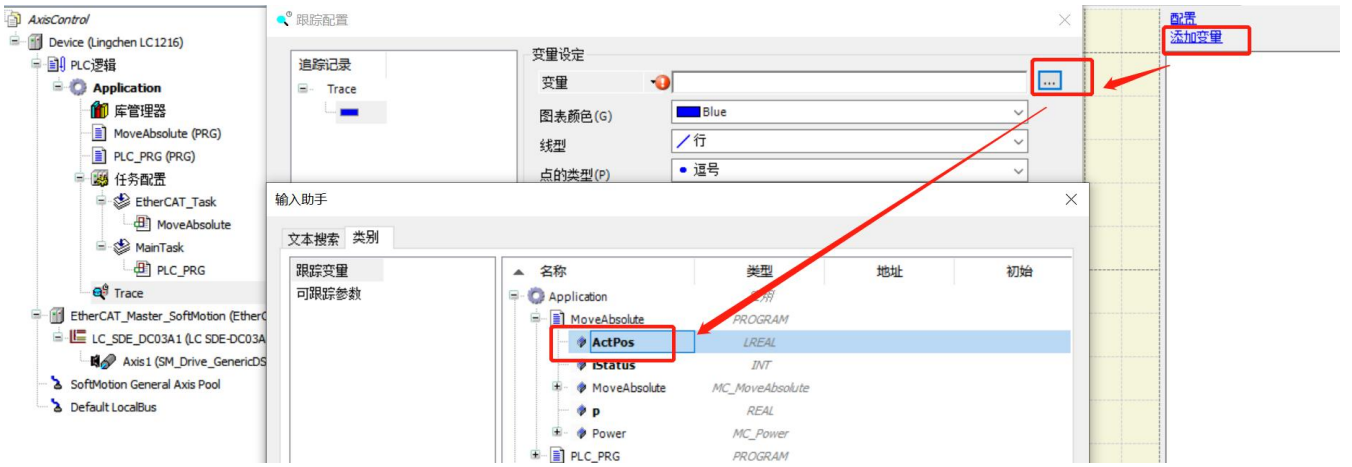
添加 Trace

为了更加直观地观察伺服轴位置的变化，这里添加一个示波器，用于记录运动曲线。 右键“应用”，选择“添加对象→跟踪”，弹出对话框，命名然后点击“添加”，添加 Trace跟踪，选择需要监控变量所在的任务周期如下图。

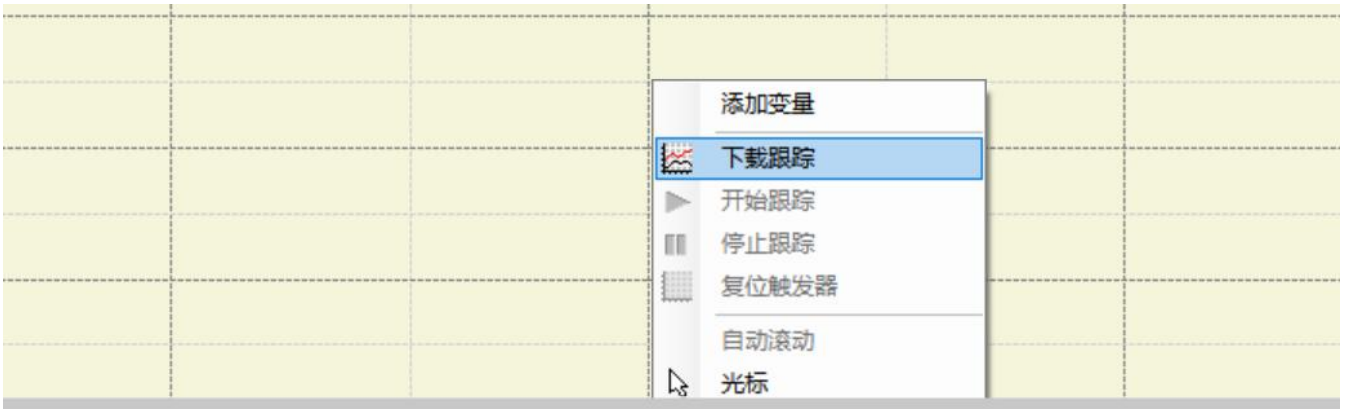


配置 Trace

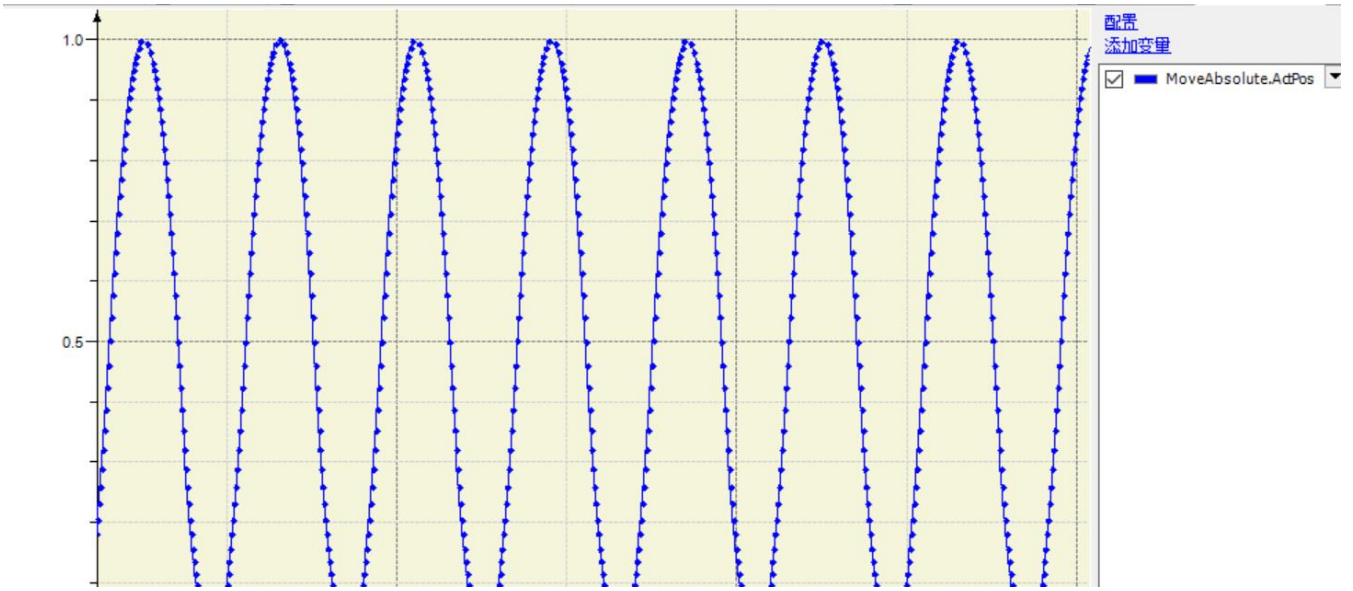
1) 点击（添加变量),选择按钮[...]，在变量弹出窗口中找到变量 “ActPos”，点击“OK”，添加到示波器中。



2) 右键单击示波器空白界面，选择“Download Trace”，下载追踪；、



3) Axis1 的实际位置曲线打印如下。



4.1.9 停止调试

调试完成后，可点击“调试→停止 ”，停止执行程序

4.2 设备常用配置说明

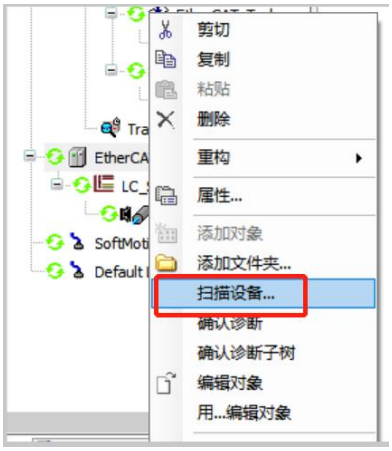
4.2.1 设备树和设备编辑器

设备树

在设备视图（也称为设备树）中，可以根据目标设备来组织应用程序。在此视图中，可 以查看 PLC 硬件和现场总线系统，配置硬件通信以及分配应用程序。

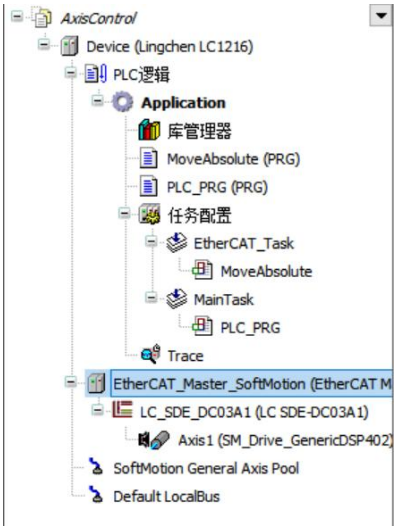
设备树的根节点是一个符号节点条目：<项目名称>。如下

在一个或多个 PLC 的该节点下插入设备对象，也称为目标系统。每个设备对象代表一个特定的硬件组件，例如控制器，现场总线，总线耦合器，驱动器，I/O 模块或监视器等。如果已经连接到控制器网络，则可以扫描硬件以查找可用的设备，并将其保存到当前配置的设备树中，如图。



每个设备由设备描述文件定义，并且必须安装在本地系统上才能插入到设备树中。设备　　描述文件为可配置性，可编程性以及与其他设备的可能连接定义了设备属性。

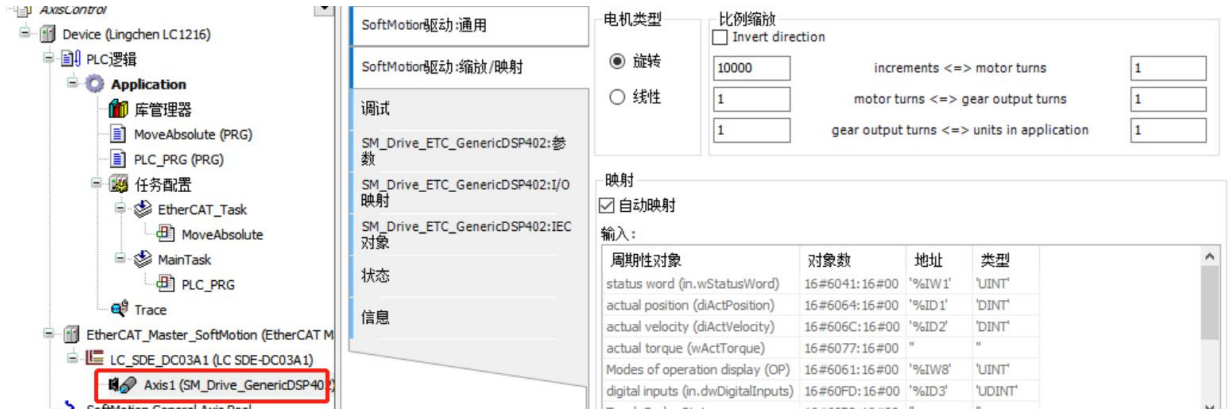
设备树示例：



设备树中的设备条目由设备符号、设备名称以及设备类型组成，例如：

。

可以在设备编辑器对话框中配置设备通信，参数和 IO 映射。双击设备对象打开此编辑器。



在线模式下的设备树

当 CODESYS 处于在线模式下时，设备条目前的符号表示设备状态:

: PLC 已连接，应用程序正在运行，设备正在运行，并且正在交换数据。
: PLC 已连接并且处于 STOP 状态。
: PLC 已连接，应用程序正在运行。诊断信息可用。
: 设备处于操作前模式，尚未运行。诊断信息可用。
: 设备未交换数据，总线错误，无法进入配置或仿真模式。
: 设备在演示模式下运行 30 分钟。经过此时间后，演示模式将终止，现场总线将结束数据交换。
: 设备已配置，但不能完全运行。没有数据交换。
: 冗余模式处于活动状态。现场总线主站未发送任何数据，因为另一个主站处于活动状态。
: 在设备存储库中找不到设备描述。
: 设备本身正在运行，但是子设备未运行。由于设备树已折叠，子设备不可见。 所

有连接的设备和应用程序的名称以绿色突出显示

在模拟模式下运行的设备的名称以斜体显示:

其他诊断信息位于相应设备编辑器的“ 状态”选项卡上。

4.2.2 Device 设备

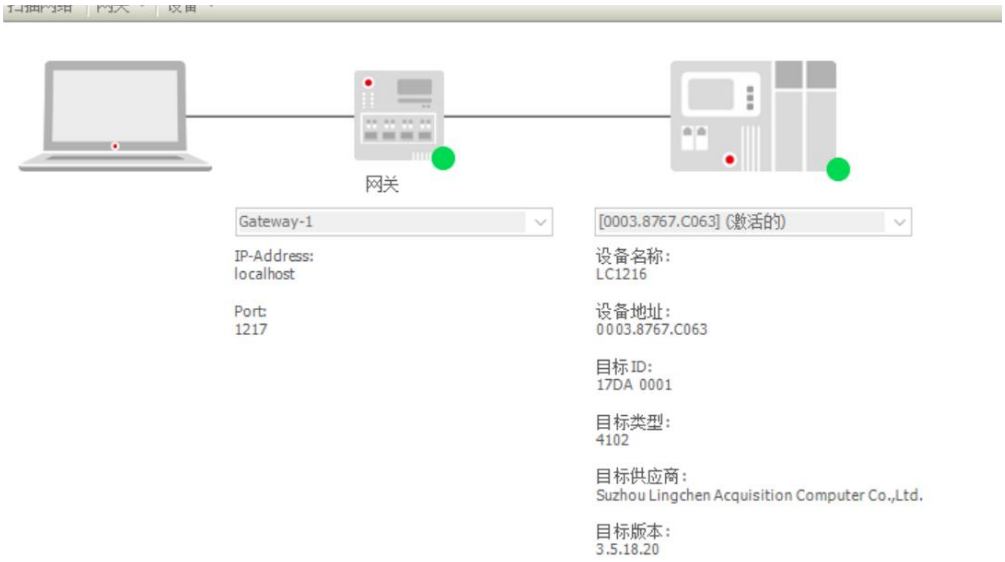
通讯设置

在通用设备编辑器的此选项卡中，您定义 CODESYS 与应用程序应在其中运行的设备之间的连接。

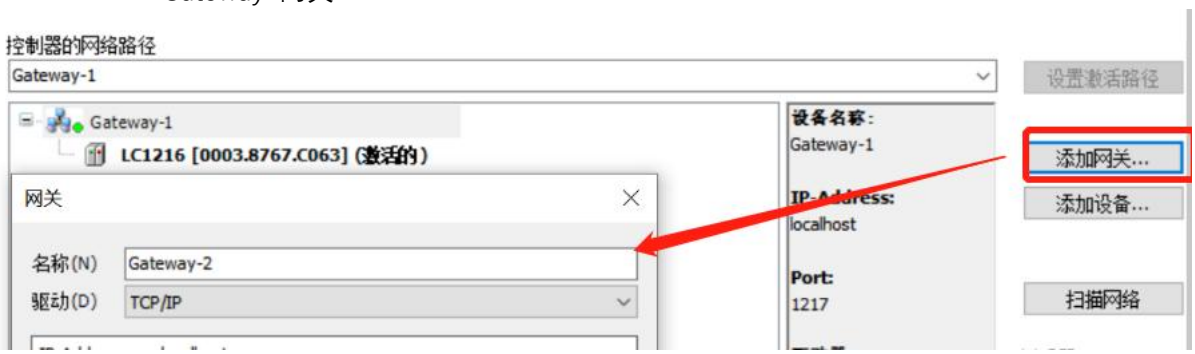
Scan network：扫描网络

扫描网络步骤如下，点击“Gateway-1”（网关）点击“Scan network”扫描网络，点击扫描出来的设备名称，再点击 OK

当设备状态灯显示绿灯并且状态描述为 Active 时，代表 PC 连上了设备

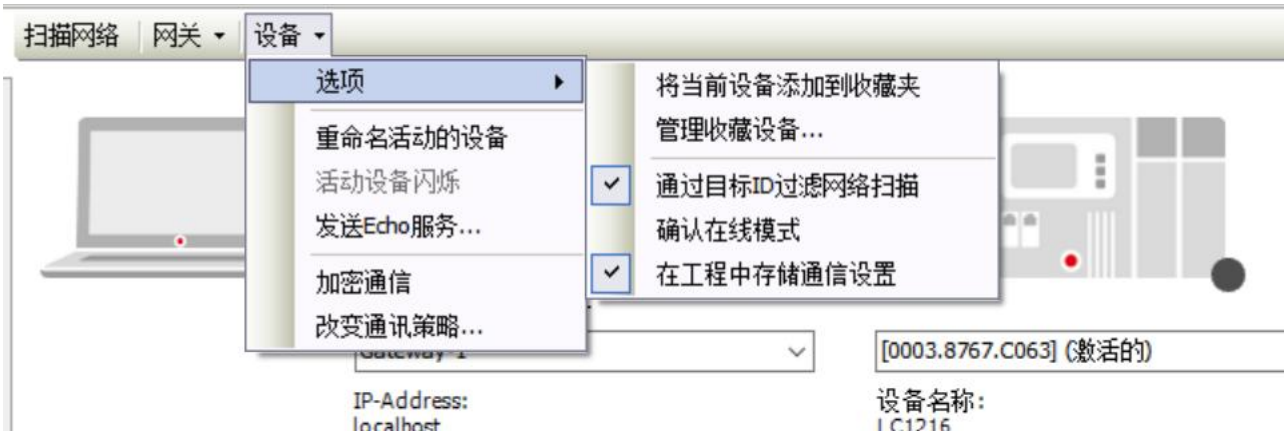


Gateway 网关：



可添加、管理网关，或配置本地网关

Device：



通过设备 ID 过滤网络扫描（取消勾选）
在工程中存储通信设置（勾选）

Applications 应用

在通用设备编辑器的此选项卡上，可以查看设备上存在哪些应用程序。根据系统的不同，可以从设备中删除应用程序或检索有关该应用程序的详细信息。


通信设置
应用
备份与还原
文件
日志
PLC设置
PLC上的应用(C)
Application

备份还原

从设备读取备份信息：该命令从 PLC 的“PlcLogic” 目录中搜索特定于应用程序的文件，并在选项卡式页面下部的表中列出它们。

创建备份文件并保存到磁盘：从设备读取备份信息命令用于确定与备份相关的文件。此命令将设置为“活动”的表中的文件和 meta.info 信息文件压缩为备份 zip 文件。 文件扩展名是 tbf（“目标备份文件”）。

保存备份文件到磁盘：此命令将备份文件保存到 PLC 的TBF 目录。

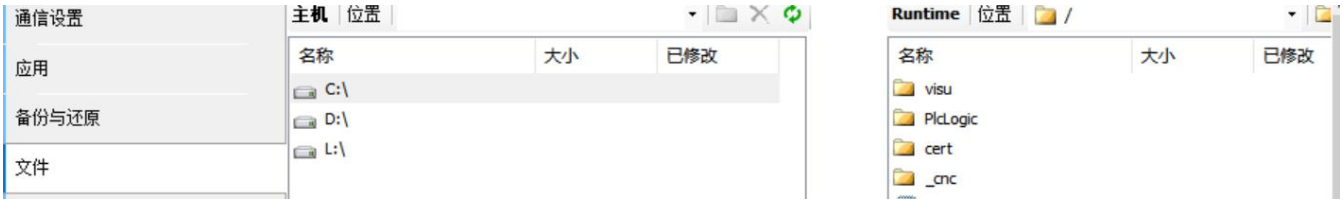
从磁盘加载备份文件：此命令将打开对话框，以在文件系统中导航已保存的备份文件。

从设备加载备份文件：此命令生成在 PLC 上找到的所有备份文件的列表。选择这些文件之一，可在选项卡式页面上的表中查看其内容。

将备份恢复到设备：如果选项卡式页面中当前加载的备份文件的至少一个组件设置为活动，则此命令可用，它提示恢复设备上的应用程序的状态。

文件

在 Device 设备编辑器的此选项卡中，可以在 CODESYS（PC 主机）和 PLC 之间传输文件。 如果通信设置正确并且 PLC 处于在线状态，则 CODESYS 在文件传输期间自动建立与PLC 的连接。



Log 日志

查看 PLC 日志。 它列出了在目标系统上记录的事件。 这涉及:

- 系统启动和关闭期间的事件（加载的组件，带版本号）
- 下载并加载启动应用程序
- 自定义条目
- 来自 I/O 驱动程序的日志
- 来自数据源的日志

PLC Settings PLC 设置

进行 PLC 组态的基本设置，例如输入和输出的处理以及总线循环任务。

停止时更新IO：勾选时，即使 PLC 处于停止状态，CODESYS 也会刷新输入和输出通道的值。 如果看门狗检测到故障，则将输出设置为预定义的默认值。不勾选时， 则当 PLC 处于停止状态时，CODESYS 不会刷新输入和输出通道的值。

停止时输出的行为：控制器进入停止状态时输出通道的处理

- 保留当前值。
- 所有输出均为默认值，默认值根据 I/O 映射分配。
- 执行程序，通过项目中包含的程序控制输出值的处理，CODESYS 在“ STOP”处执行该程序。 在右侧的字段中输入程序的名称。

总是更新变量：定义 CODESYS 是否更新总线周期任务中的 I/O 变量。 仅当从站和模块的更新设置中定义为“禁用”时，此设置才对从站和模块的 I/O 变量有效。

- 失能（仅在任务中使用时更新），

CODESYS 仅在任务中使用 I/O 变量时才更新。

- 激活 1（如果未在其他任务中使用，则使用总线循环任务），如果未在其他任务中使用它们，则 CODESYS 更新总线循环任务中的 I/O 变量。
- 激活 2（始终在总线循环任务中）CODESYS更新总线循环任务的每个循环中的所有变量，无论是否使用它们以及将它们映射到输入通道还是输出通道。

总线周期任务：控制总线周期的任务。默认情况下，输入由设备描述定义的任务，将应用上级总线设备的总线周期设置（上级总线的周期使用设置）即向上扫描设备树以查找一个有效的总线周期任务定义。

用户和组

在通用设备编辑器的此选项卡上，可以编辑控制器的设备用户管理。根据设备支持的方 式，可以定义用户帐户和用户组。 结合访问权限选项卡上的配置，可以在运行时控制对控件对象和文件的访问。

Synchronization：打开和关闭设备上编辑器和用户管理之间的同步，如果未按下该按钮，则编辑器为空白。 如果按下该按钮，则 CODESYS 会将编辑器中的显示与所连接设备上的当前用户管理连续同步。

Import from disk：用于从硬盘中选择和导入用户管理配置。

Users

Add：打开“添加用户”对话框以创建新的用户帐户

Import：打开对话框导入用户。

Groups

Add：打开对话框添加组。 定义一个新的组名，从定义的用户列表中，选择属于该组的用户。

Import：打开对话框导入用户。

访问权限

在设备编辑器的此选项卡上，定义设备用户对控制器上的对象的设备访问权限。与项目用户管理一样，用户必须是至少一个用户组的成员，并且只能向用户组授予某些访问权限。

符号权限

在通用设备编辑器的此选项卡中，定义了不同用户组（客户端）对控制器上可用的各个 符号集的访问权限。

要求：必须在 PLC 上设置用户管理。 已将一个应用程序下载到控制器，该控制器在 CODESYS 项目中为其定义了符号集。 他们具有用于登录控制器的访问数据。

在“符号集”视图中，所有符号集都列在“应用程序”节点下，该节点的定义与应用程序一起下载到控制器。在“权限”视图中，在表的列表中列出了在控制器的用户管理中定义的用户组。 选择符号集后，将看到相应用户组对该符号集的访问权限。 : 授予访问权限; : 未授予访问权限。 可以通过双击符号来更改访问权限。

单击 按钮将当前访问配置保存到 XML 文件。文件类型为设备符号管理文件(*.dsm)。单击 按钮以从硬盘驱动器读取这样的文件。

任务部署

设备编辑器的子对话框显示输入和输出表以及它们对已定义任务的分配。

仅在为应用程序生成代码后，该信息才变得可见。它用于故障排除，因为它显示了在具 有不同优先级的多个任务中使用输入或输出的位置。

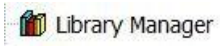
状态

通用设备编辑器的此选项卡显示状态信息，例如“正在运行”或“已停止”，以及来自各个设备的特定诊断消息，还包括有关内部总线系统的信息。

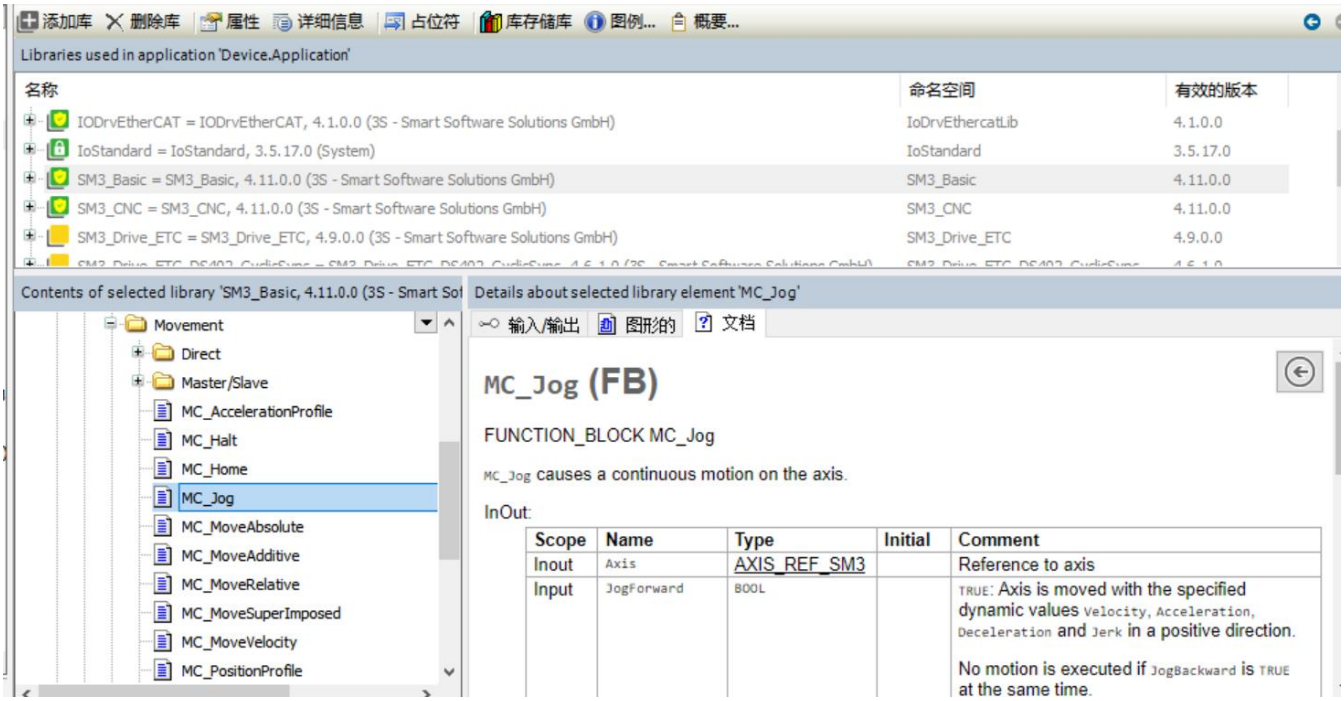
信息

通用设备编辑器的此选项卡显示源自设备描述文件的常规信息：名称，供应商，类别， 版本，订单号，描述等等。

4.2.3 库管理器



库管理器列出了项目中集成的所有库，用于创建应用程序。它提供有关库类型、属性和 内容的信息，可以展开或折叠集成库的列表。



项目中集成的所有库的列表。如果一个库依赖于其他库，则这些引用的库将自动集成。 库管理器包含三个视图：

- 上视图：集成库列表
- 左下方视图：树形结构，在上方视图中选择了库的所有模块
- 右下方视图：在树中选择的模块的相关文档

4.3 EtherCAT 总线常用配置

4.3.1 EtherCAT_Master 主站

General（常规）

自动配置主站从站：默认情况下，自动配置模式处于激活状态，适用于标准应用程序。如果未激活该模式，则必须手动进行主机和从机的所有配置设置，这需要专业知识。勾选后，根据设备描述文件和隐式计算，大部分主从配置是自动完成的。默认勾选。

EtherCAT NIC setting（EtherCAT NIC 设置）

目的地址：EtherCAT 网络中要接收电报的设备的 MAC 地址。

广播 ：无需指定目的地址（MAC）默认勾选。

冗余：如果总线以环形拓扑构造并且要支持冗余，则激活该功能。 使用此功能，即使在电缆断开的情况下，EtherCAT 网络也可以正常工作。 如果激活此功能， 则必须在“冗余 EtherCAT NIC 设置”区域中定义参数。默认不勾。

源地址：控制器（目标系统）的 MAC 地址或网卡名称（即 PLC（目标系统）点击 浏览 选择。

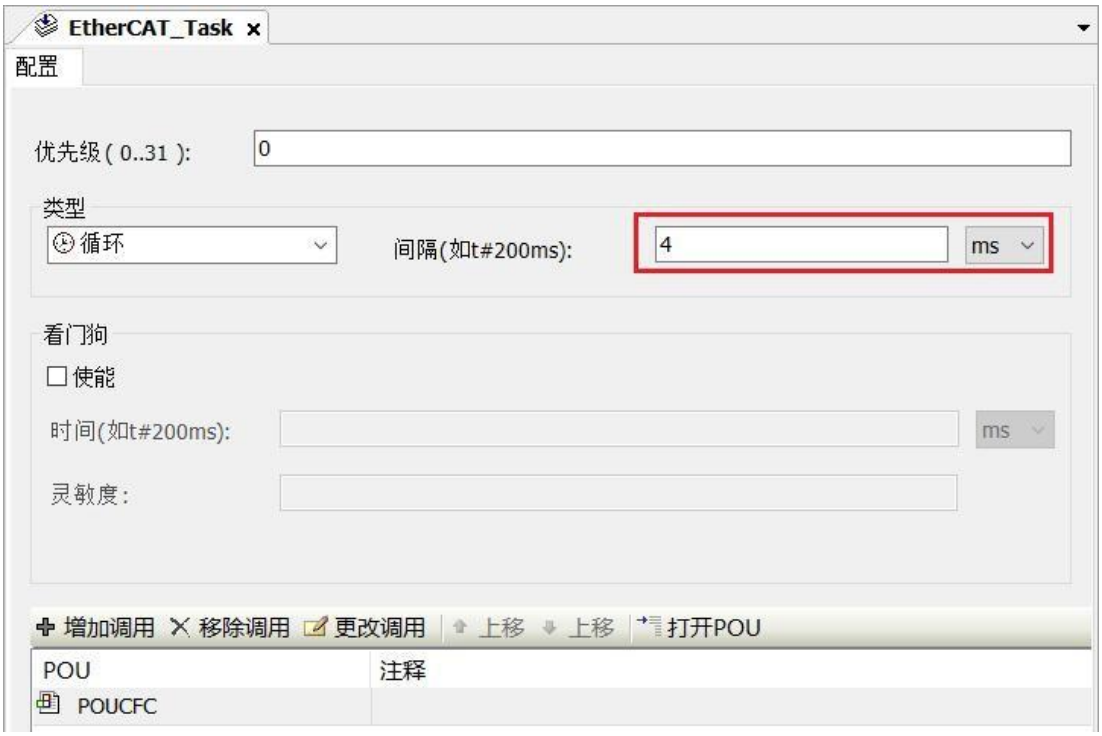
网络名称：与 Source address 取决于激活了以下哪个选项。

浏览：扫描网络以查找当前可用的目标设备的 MAC-ID 或名称。、

分布式时钟

周期 (µs)：在总线上分派新数据报文的时间间隔。 如果在从站中激活了分布式时钟功能，则此处指定的主站周期时间将传输到从站时钟。 以这种方式可以实现数据交换

的精确同步，这在空间分布的过程需要同时动作的情况下尤其重要。 例如，同时动作是多个轴必须同时执行协调运动的应用。 通过这种方式，可以实现非常精确的全网时基，其抖动基本上小于 1 微秒。
注意：分布式时钟时间设置默认与 EtherCAT_Task 时间设置一致，如若修改分布式时钟时间或 EtherCAT_Task 任务时间，需两者同时修改一致，否则编译出错，无法运行。



同步偏移：允许将 EtherCAT 从站的同步中断的时间延迟调整为 PLC 的循环时间。 通

常，PLC 周期比从站的同步中断晚 20%开始。 这意味着 PLC 循环可能会延迟 80%的循环时间而不会丢失任何消息。

同步窗口监视：可以监视从站的同步。

同步窗口：同步窗口监视的时间。 如果所有从站的同步都在此时间窗口内，则变量 xSyncInWindow（ioDrvEthercat）设置为 TRUE，否则设置为 FALSE。

选项

使用 LRW而非 LWR/LRD：从站到从站的直接通讯是可能的。使用组合的读/写命令（LRW）代替单独的读（LRD）和写命令（LWR）。

每个任务的消息：读写命令，即输入和输出消息的处理，可以通过各种任务来控制。

自动重启从站：如果通讯中断，主站立即尝试重启从站。

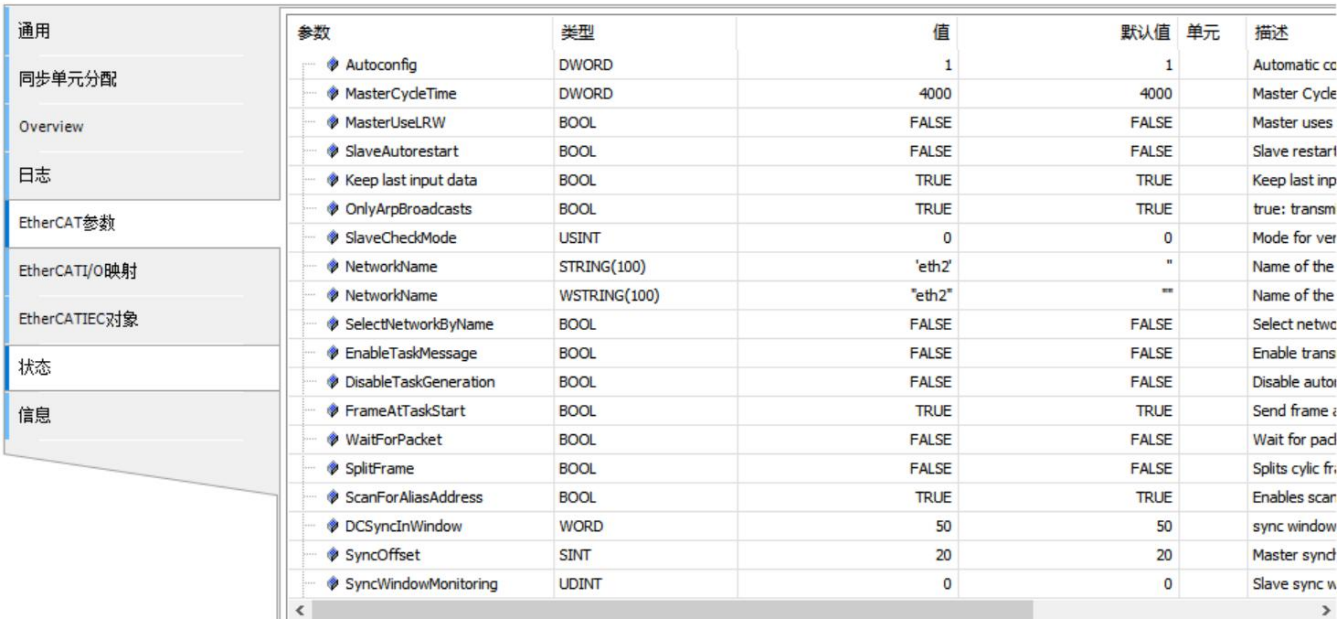
同步单元分配

此选项卡显示插入到特定主站下方的所有从站，并分配了同步单元。

使用 EtherCAT 同步单元，可以将多个从站配置为组，然后细分为较小的单元。 对于每个组，可以监视工作计数器以改进和更精确地进行错误检测。一旦同步单元组中缺少一个从站，该组中的其他从站也将显示为丢失。 由于连续检查工作计数器，因此在下一个总线周期中立即进行检测。 通过设备诊断，可以尽快纠正丢失的组。

未受影响的群体仍可正常运作，不受任何干扰。

Ethercat参数



参数	类型	值	默认值	单元	描述
Autoconfig	DWORD	1	1		Automatic cc
MasterCycleTime	DWORD	4000	4000		Master Cycle
MasterUseLRW	BOOL	FALSE	FALSE		Master uses
SlaveAutorestart	BOOL	FALSE	FALSE		Slave restart
Keep last input data	BOOL	TRUE	TRUE		Keep last inp
OnlyArpBroadcasts	BOOL	TRUE	TRUE		true: transm
SlaveCheckMode	USINT	0	0		Mode for ver
NetworkName	STRING(100)	'eth2'	''		Name of the
NetworkName	WSTRING(100)	"eth2"	""		Name of the
SelectNetworkByName	BOOL	FALSE	FALSE		Select netwc
EnableTaskMessage	BOOL	FALSE	FALSE		Enable trans
DisableTaskGeneration	BOOL	FALSE	FALSE		Disable autor
FrameAtTaskStart	BOOL	TRUE	TRUE		Send frame a
WaitForPacket	BOOL	FALSE	FALSE		Wait for pacl
SplitFrame	BOOL	FALSE	FALSE		Splits cylic fr
ScanForAliasAddress	BOOL	TRUE	TRUE		Enables scar
DCSyncInWindow	WORD	50	50		sync window
SyncOffset	SINT	20	20		Master synch
SyncWindowMonitoring	UDINT	0	0		Slave sync w

该选项卡包含在设备描述文件中定义的主参数。

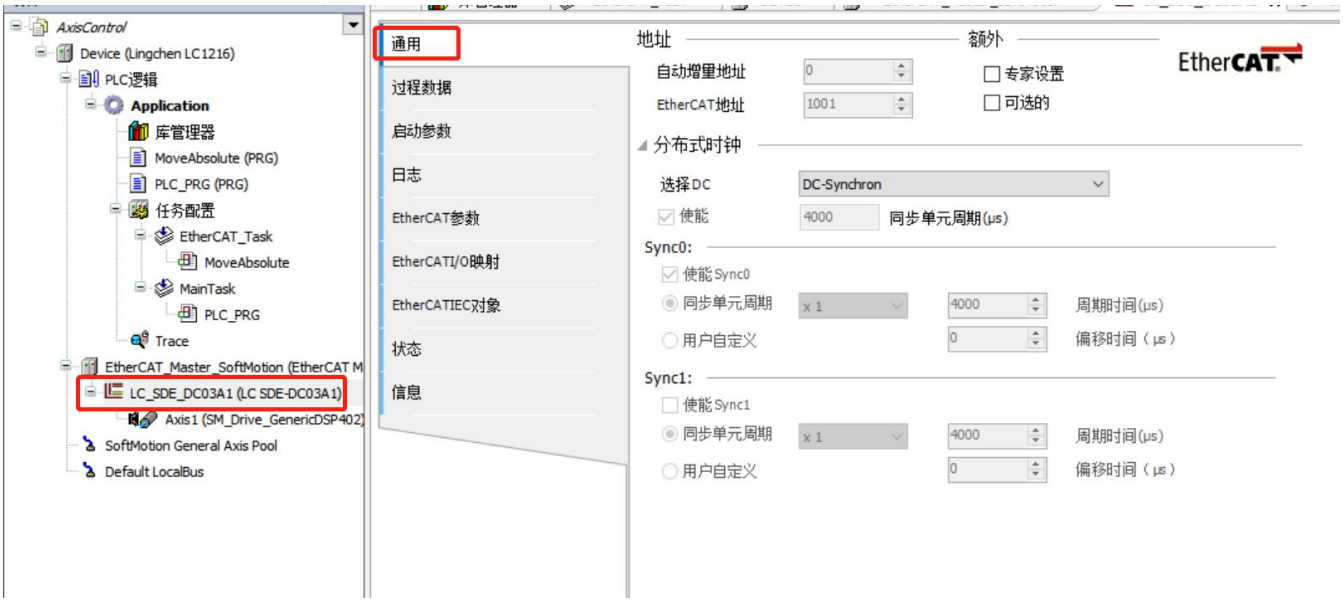
如果在“主”对话框中激活了自动配置模式，则此处将根据设备描述文件和网络拓扑中的规范自动设置参数。 通用编辑器中不应更改任何内容，因为可以在此处设置无效的配置。

4.3.2 EtherCAT_Slave 从站

对象:EtherCAT 从站

EtherCAT 从站的基本设定在该选项中配置。设备描述文件已预设为基本设定。

通用



地址

仅当禁用 EtherCAT 主站的自动配置模式时，才可以编辑字段。

自动增量地址：自增地址(16 位)由网络中从站的位置产生。只有主站把 EtherCAT 地址分配给它的从站时，地址才在系统启动过程中使用。当第一条消息为此经过从站时,各个从站的 AutoInc 地址自加 1。

EtherCAT地址：主站在启动中分配给从站的最终地址，地址独立于从站在网络中的位置。

额外）

专家设置：附加的设定在启动检查和时间监控时可用(见下文)。勾选后专家过程数据选项卡在设备编辑器中可用，然而，专家设置不是标准应用所必需的。

可选的：从站定义为可选项，当总线系统中缺少设备时不会生成错误信息。注意:如果将从站定义为"可选",那么该从站必须有一个独特的标识。可以通过更改 Identification 部分的三种可能设定对此进行更改。只有当主/从站自动配置选项在 EtherCAT 主站中激活且EtherCAT 从站支持该功能时才可用。

分布式时钟

选择DC：包含设备描述文件中分布式时钟的所有设置的下拉列表。

使能：显示在同步单元周期(µs)，用于数据交换的周期时间,由主站的周期时间确定，因此主站时间能在网络中同步数据交换。

下面描述的 Sync0 和 Sync1 设置是从属依赖的:

Sync0

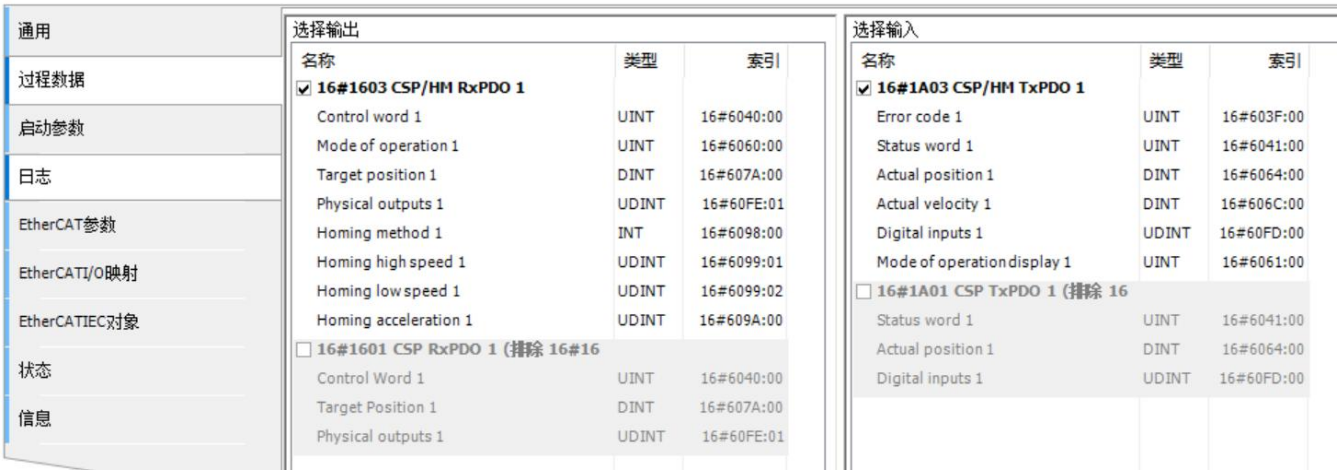
激活 Sync0	☑:使用同步单元 Sync0。同步单元描述一串同步交换的过程数据。
同步单元循环	:主站周期时间(乘以从下拉列表中选择的因子)用作从站的同步周期时间，周 期时间(µs)显示当前设定的周期时间。
用户自定义	:用户自定义周期时间(毫秒级)可以在周期时间(µs)字段指定。

Sync1

激活 Sync1	☑:使用同步单元 Sync1.同步单元描述一串同步交换的过程数据。
同步单元循环	:主站周期时间(乘以从下拉列表中选择的因子)用作从站的同步周期时间，周 期时间(µs)显示当前设定的周期时间。
用户自定义	:用户自定义周期时间(毫秒级)可以在周期时间(µs)字段指定。

过程数据

此 EtherCAT 配置器选项显示用于从站输入输出的过程数据，数据源自设备描述文件。



选择输出：表格显示从站输出名称,、类型,和索引地址。如果此处的设备输出(用于写)激活，这些输出可以被分配到 EtherCAT I/O 映射对话框的项目列表。

选择输入：表格显示从站输入名称,、类型,和索引地址。如果此处的设备输入(用于读)激活，这些输入可以被分配到 EtherCAT I/O 映射对话框的项目列表。

专家过程数据

设置该选项，需要先勾选从站中专家设置的复选框 ☐专家设置 ，勾选后

会出现新的选项卡，如下图，该选项提供一个不同且更详细的过程数据视图。

同步管理器：带有数据大小和 PDO 类型的 sync 管理器列表

PDO 分配：分配到已选同步管理器的 PDO 列表，如果选中复选框,则激活 PDO 并创建 I/O 通道。

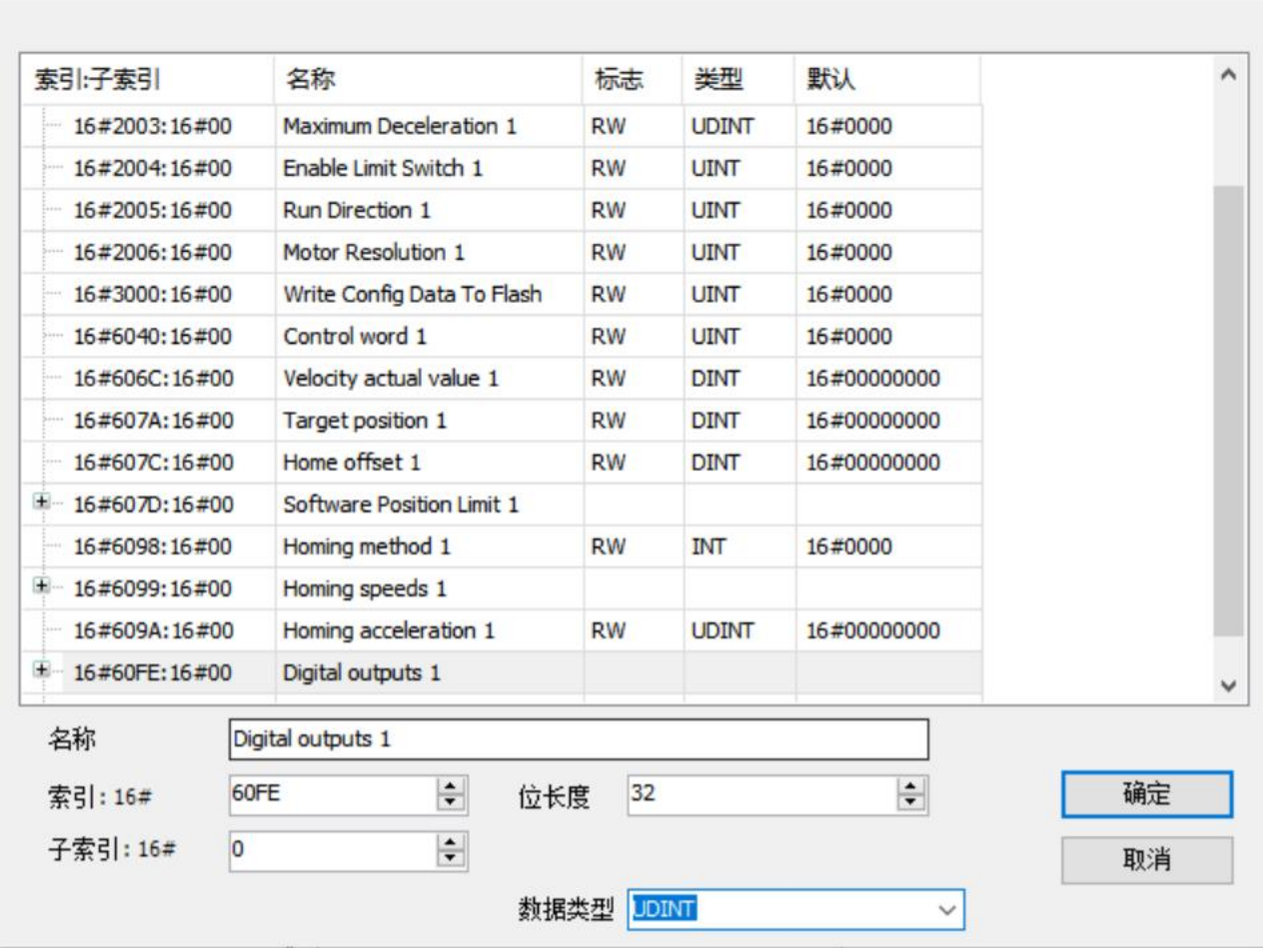
PDO List：分配到已选同步管理器的 PDO 列表，可以通过执行命令栏或快捷菜单中的不同命令(添加，删除，编辑)来添加新的 PDO 或者编辑和删除现有的 PDO。

PDO 内容：显示 PDO 列表中已选的 PDO 内容。可以通过执行命令栏或快捷菜单中的不同命令(Insert 添加, Delete 删除, Edit 编辑)来添加新条目或编辑和删除现有条目。可以通过点击 上移 和 下移 移动来改变 PDO 顺序。

注意：

当项目需要自定义PDO 时，伺服提供两组 PDO 可供用户自定义：16#1601和 16#1A01。添加方法：在 PDO List 选中 16#1601 或 16#1A01，然后点击 插入 添加，弹出选择对话框，此对话框中包含了伺服所有的对象，用户可根据项目需要进行选择，然后点击 确定 即可插入，自定义添加 PDO。

从对象目录中选择条目



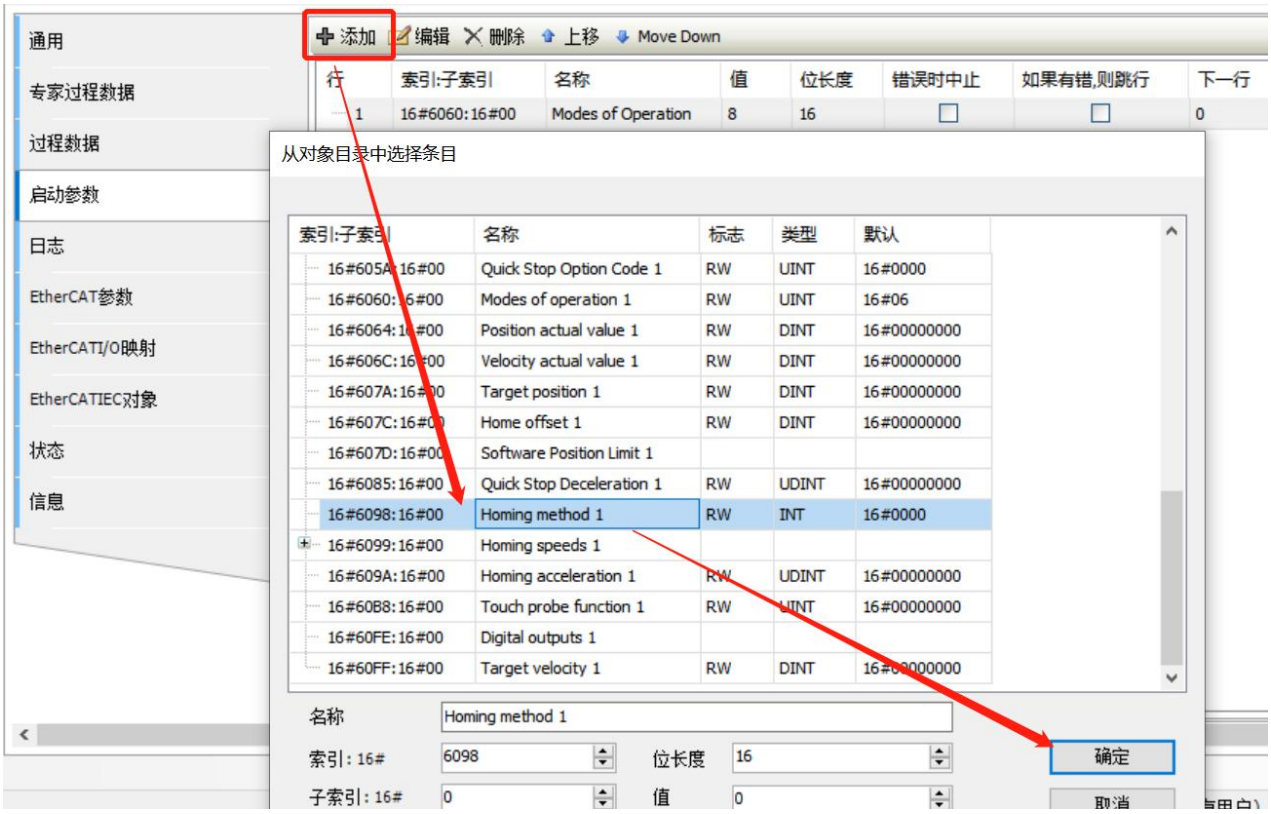
下载:

PDO Assignment	:生成用于初始化 `0x1cxx`对象的特定 CoE 命令并写入从站。
PDO 配置	:生成用于 `0x16xx` or `0x1axx`的 CoE 命令,加载 PDO 映射到从站。作为一项规则,默认值来自 ESI 文件,且设备必须支持此功能。
从设备加载 PDO 信息	从从站读取当前 PDO 配置并进入配置.然后删除右上角和右下角的列表并用读取的数据填充。此项在 ESI 文件不完整和配置只在从站可用时尤其有效。

启动参数

在此选项中为当前从站定义系统启动时传输指定参数到设备的 SDO(服务数据对象)或在 XML 文件中引用的 EDS 文件中描述。

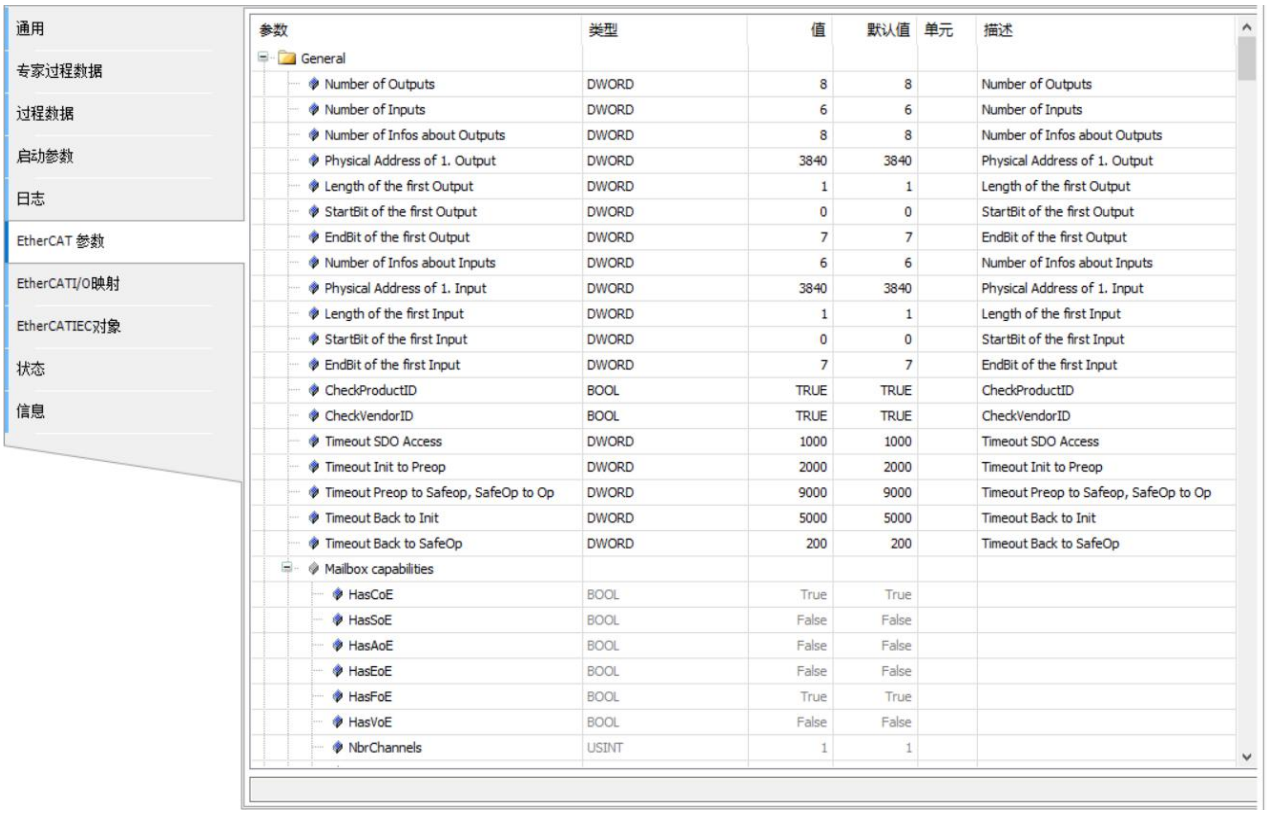
要求:设备支持 CAN over EtherCAT 或 Servodrive over EtherCAT



注意：某些插入到从站下面的模块有自己的启动参数.然后这些参数在该列表中显示,但不能被修改。这些参数可以在相关模块的编辑器中改变。

SDO 表中的顺序(从上到下)指定 SDO 传输到模块的顺序.	
行	行号
索引：子索引	////////////////SDO 的索引号和子索引号
位长度	SDO 的位长度
错误时中止	:在出现错误状态的情况下,传输中断.
如果有错，则跳行	:为防止错误,在指定行恢复 SDO 传递.
下一行	:在下一行使用 SDO 恢复传输.
注释	用于注释的输入字段
上移	将选定行向上移动一行
Move Down	将选定行向下移动一行
添加	打开对话框从对象目录中选择条目，在该对话框中可以在 SDO 被添加到配置之前改变 SDO 参数。通过指定新的索引/子索引条目,可以添加新的对象到还没有在 EDS 文件中描述的 SDO 中。
删除	移除已选的条目。
编辑	打开对话框从对象目录中选择条目来表格中已选 SDOs 参数。

EtherCAT参数



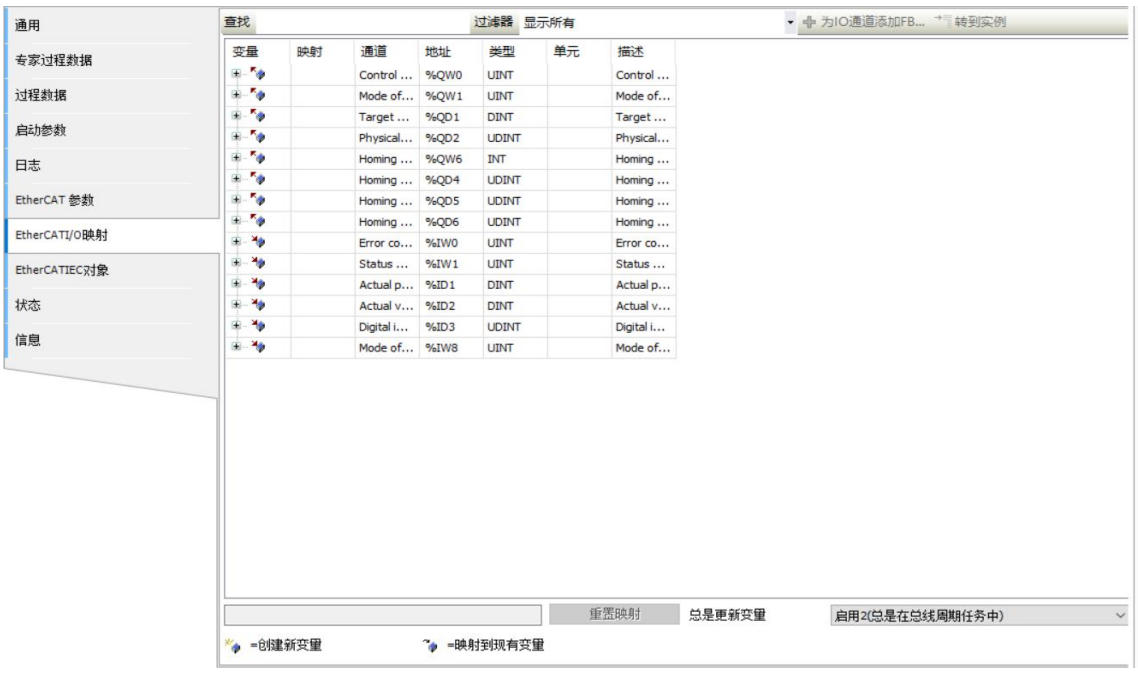
参数	类型	值	默认值	单元	描述
General					
Number of Outputs	DWORD	8	8		Number of Outputs
Number of Inputs	DWORD	6	6		Number of Inputs
Number of Infos about Outputs	DWORD	8	8		Number of Infos about Outputs
Physical Address of 1. Output	DWORD	3840	3840		Physical Address of 1. Output
Length of the first Output	DWORD	1	1		Length of the first Output
StartBit of the first Output	DWORD	0	0		StartBit of the first Output
EndBit of the first Output	DWORD	7	7		EndBit of the first Output
Number of Infos about Inputs	DWORD	6	6		Number of Infos about Inputs
Physical Address of 1. Input	DWORD	3840	3840		Physical Address of 1. Input
Length of the first Input	DWORD	1	1		Length of the first Input
StartBit of the first Input	DWORD	0	0		StartBit of the first Input
EndBit of the first Input	DWORD	7	7		EndBit of the first Input
CheckProductID	BOOL	TRUE	TRUE		CheckProductID
CheckVendorID	BOOL	TRUE	TRUE		CheckVendorID
Timeout SDO Access	DWORD	1000	1000		Timeout SDO Access
Timeout Init to Preop	DWORD	2000	2000		Timeout Init to Preop
Timeout Preop to Safeop, SafeOp to Op	DWORD	9000	9000		Timeout Preop to Safeop, SafeOp to Op
Timeout Back to Init	DWORD	5000	5000		Timeout Back to Init
Timeout Back to SafeOp	DWORD	200	200		Timeout Back to SafeOp
Mailbox capabilities					
HasCoE	BOOL	True	True		
HasSoE	BOOL	False	False		
HasAoE	BOOL	False	False		
HasEoE	BOOL	False	False		
HasFoE	BOOL	True	True		
HasVoE	BOOL	False	False		
NbrChannels	USINT	1	1		

此选项包含在设备描述文件中定义的从站参数。

如果主站的自动配置模式被激活,那么这些参数将根据设备描述文件和网络拓扑结构的规范自动在此设置。对于标准应用一般来说不用修改它们。

EtherCAT 输入输出映射

Process Data（过程数据）选项中选择的从站的输入、输出会列在此处，它显示了可用的通道，并允许将控制器的输入，输出和内存地址映射到应用程序的变量或整个功能块。通 过这种方式，可以创建所谓的“I/O 映射”。



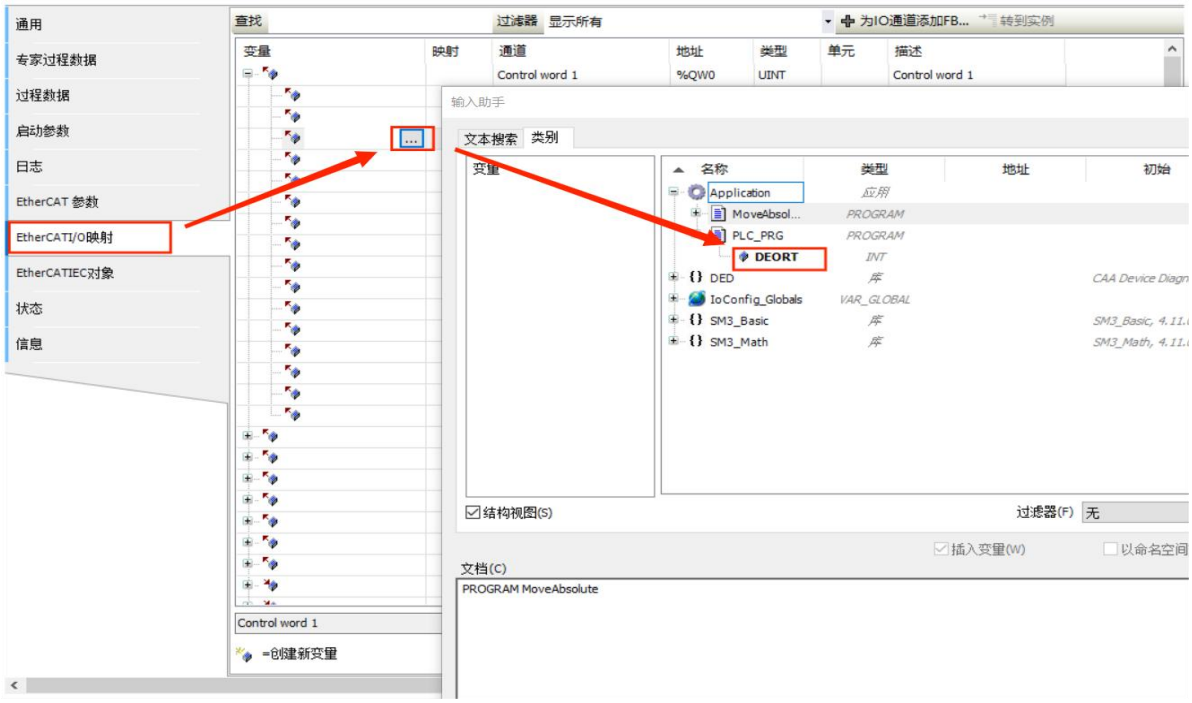
查找	映射表搜索输入字段的字符串。
过滤器	映射表中列出的用于过滤 I/O 映射的下拉列表: 显示所有 只显示输出 只显示输入 只显示未映射的变量 只显示映射变量 只显示映射到现有变量 只显示映射到新变量
为IO通道添加FB...	取决于设备，如果在映射表中选择了通道条目，则可用。打开“ 选择功能块”对话框，以选择应直接链接到通道的功能块。
转到实例	如果在映射表中选择了该条目，则可用。跳转到<设备名称> IEC Objects 选项卡中的相应条目。
变量	根据设备的不同，设备的输入和输出将显示为节点，并在节点下方显示缩进的关联通道，或者根据设备，仅显示隐式创建的设备实例。符号表示通道类型: : 输入 : 输出 双击单元格将打开一个输入字段。 选项 1: 该变量已存在；指定完整路径: <应用程序名称>。<模块名称>。<变量名称>; 例如: app1.plc_prg.ivar; 通过输入帮助 ... 。 选项 2: 该变量尚不存在；输入一个简单的名字；在内部自动创建为全局变量 根据设备的不同，输入或输出可以直接链接到功能块。在这种情况下，将激活 “ 为 I/O 通道添加 FB ”按钮。
映射	映射类型: : 变量已存在 : 新变量 FB: 映射到功能块实例
通道	通道的符号名称。
地址	通道地址，例如 %IW0。 地址删除线: 表示您不应为该地址分配任何其他变量。原因: 尽管此处指定的变量（作为已经存在的变量）在不同的存储位置进行管理，但是在写入值的过程中可能会产生歧义，尤其是对于输出。 : 表示此地址已被编辑和修复。如果设备树中设备对象的排列发生更改，则 CODESYS 不会自动调整该地址。

类型	通道的数据类型，例如 BOOL。 仅当设备描述中定义的结构或位字段属于 IEC 标准并在设备描述中被标识为 IEC 数据类型时，才会显示它们。否则，表格单元格保持为空。 映射结构变量时，编辑器会阻止同时输入结构变量（例如）%QB0 和单个结构元素（例如：%QB0.1 和 QB0.2）。因此，如果在映射表中存在一个带有位通道条目的子树的主条目，则适用以下条件：可以在主条目的行中或子元素（位通道）的行中输入一个变量 ，但不能同时使用。
Default Value	适用于通道的参数的默认值：仅在“ PLC Setting”中针对停止时的输出行为激活了“ Set all output to defalt ”选项时，才会显示。
单元	参数值的单位，例如 ms 毫秒。
描述	参数的简要说明。
Current Value	应用于通道的参数的实际值；仅在在线模式下显示。

重置映射	CODESYS 将映射设置重置为设备描述文件定义的默认值。
总是更新变量	有关更新 I/O 变量的设备对象的定义。默认值在设备描述中定义： 使父设备设置：根据上级设备的设置进行更新。 使能1（如果未在任何任务中使用总线周期任务，则使用它）如果未在任何其他任务中使用它们，则 CODESYS 更新总线周期任务中的 I/O 变量。 启用2（始终在总线循环任务中）CODESYS 在总线循环任务的每个循环中更新所有变量，无论是否使用它们以及将它们映射到输入通道还是输出通道。

程序关联 I/O 映射的方法有两种：

（1） 映射中选择变量



（2） 程序中赋地址

```
PROGRAM PLC_PRG
VAR
    w100 AT %IW5: REAL;
END_VAR
```

在线

登录设备在线后，会出现 Online（在线）选项卡，通过 EtherCAT，可以使用从站状态信息和用于将文件传输到从站的功能。

状态机

Init	初始化，用于调试目的
Boot Strap	从站切换到 *Bootstrap 模式* 如果要与从设备传输固件文件，则为必需
Pre-Op	预操作模式，用于调试目的
Safe-Op	安全启动模式，用于调试目的
Op	开，用于调试目的
当前状态	当前状态
请求状态	请求状态

通过 EtherCAT 进行文件访问

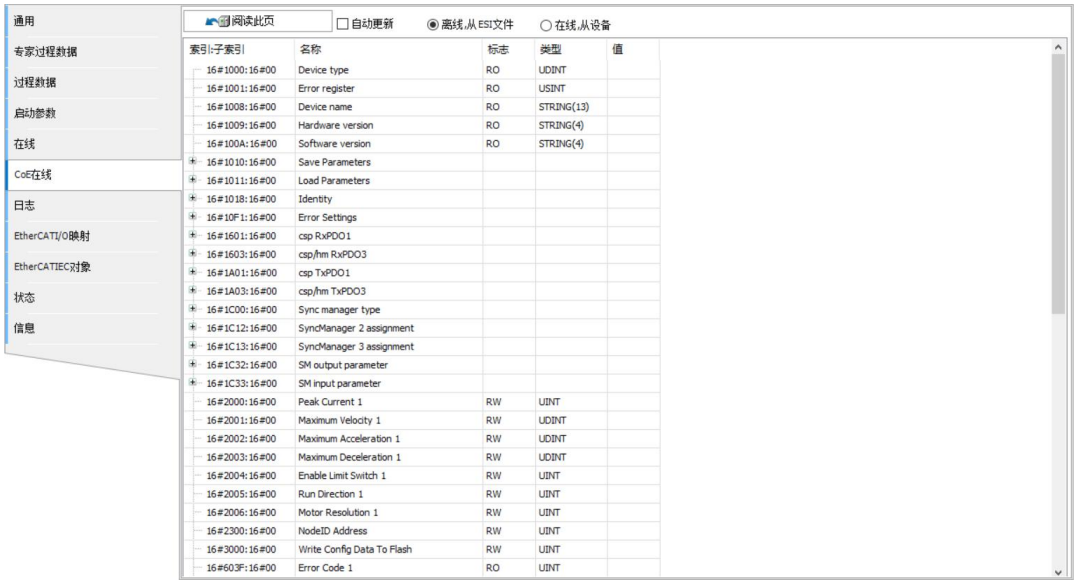
下载	下载固件文件 出现一个对话框，用于存储固件文件。在此对话框中，必须输入字符串和密码才能执行文件传输。该信息将从从站的数据表中获取。
上传	上载固件文件 出现一个对话框，用于打开固件文件。在此对话框中，必须输入字符串和密码才能执行文件传输。该信息将从从站的数据表中获取。

E²PROM 访问

写入²PROM	将从站的配置写入 E²PROM。
读取 E²PROM	从 E²PROM 读取从站的配置。上载固件文件
写入 E²PROM XML	将从属配置直接从 XML 文件写入设备。 仅当 XML 文件中存在配置数据（<ConfigData>部分）时才能执行。

CoE 在线

设置该选项，需要从站支持 CoE Online 模式，先勾选从站中专家设置的复选框 ☑ Enable expert settings ，并且登录设备在线后，会出现新的选项卡 CoE 在线，如下图，此选项显示 ESI 或从站的对象索引。



阅读此页	对象索引被读取一次
自动更新	对象被周期性读取，自动更新
离线，从 ESI 文件	此对话框显示设备描述中对象索引的内容
在线从设备	显示设备中对象索引的 SDOInfo 必须在 ESI 文件中使能
标志	RO:值受写保护 RW:值可以更改
类型	参数的数据类型
值	值可以通过双击进入文本区来编辑

4.3.3 SM_Drive_GenericDSP402 轴配置

SoftMotion驱动：通用

轴类型和限位

虚拟模式：该驱动器被类似于虚拟驱动器单元的模拟所替代。如果有耦合驱动器，则对现场总线设备没有任何影响，它们照常运行，而无需向物理设备发送消息或从物理设备接收消息。

注意：还可以通过 SMC3_ReinitDrive 功能块以 IEC 代码设置和重置驱动器的虚拟模式。

模数：驱动器无限旋转，而不限制运行范围（例如皮带驱动器）。

模数值[u]: 一个周期（模周期）的值。该值保存在功能块 AXIS_REF_SM3 的 fPositionPeriod参数中。

注意：如果选择 Modulo 驱动器类型，则乘积必须为整数。

fPositionPeriod * dwRatioTechUnitsDenom

有限：驱动器具有固定的工作区域（例如一个线性驱动器）。

激活：软件限位开关已激活 ☑：位置值受下限负值和上限正值限制。

- 负[u]：负极限值的输入字段
- 正[u]：正极限值的输入字段

软件错误反应

减速度 [u/s²]：到达限位开关时的减速度值。

最大距离 [u]：可选的，发生错误后，驱动器必须在此距离内达到停止状态。

动态限制

速度 [u/s]：速度的极限值

加速度 [u/s²]：加速度的极限值

减速度 [u/s²]：减速度的极限值

加加速度 [u/s³]：加速度变化率的极限值

速度斜坡类型

梯形：梯形速度曲线（在每个段中具有恒定的加速度）

Sin²：由 sin²函数定义的速度曲线（具有恒定的加速度曲线）

二次：梯形形式的加速度曲线，有加速度限制

二次(平滑)：与 Quadratic 相似，但生成的跳动曲线没有跳跃

标识

ID：整数标识符。每个驱动器应该唯一。例如，此标识符在 PLC 日志中使用，以便在发生错误时识别驱动器。

位置滞后监视

去使能：已停用。没有响应或牵引错误监视被禁用。

禁用驱动：禁用驱动器。bRegulatorOn 位被强制设置为 FALSE（MC_Power 输入）这首先会强制驱动器减速，然后使驱动器停用（取决于驱动器实现）。

快速停止：快速停止。bDriveStart 位被强制设置为 FALSE（与 MC_Power 输入比较）这将强制驱动器执行快速停止。

保持使能：保持启用状态。驱动器保持打开状态，但是所有运行中的动作突然停止。

滞后限制：滞后极限。拖动控制器中的错误监视。

SoftMotion驱动：缩放/映射

电机类型

旋转：旋转式。缩放比例中的设置适用于旋转电机。

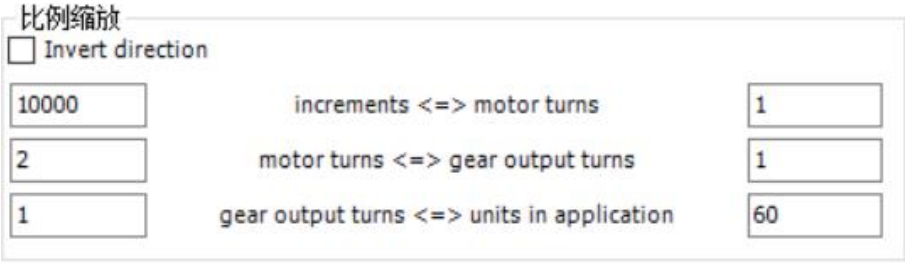
线性：线性式。放比例中的设置适用于线性电动机。(无齿轮和电机转弯的简化配置)

Scaling

Invert direction：反转电机方向。电机获得带有相反符号的指定值。

increments <=> motor turns：脉冲增量和电机圈数。

motor turns <=> gear output turns：电机圈数和给定齿轮输出圈数。**gear output turns <=> units in application**：齿轮输出圈数和应用的单元。例：



如图，电机转 1 圈对应的脉冲增量为 10000，减速比为 2:1，齿轮输出 1 圈对应的终端行进 60 个单元。

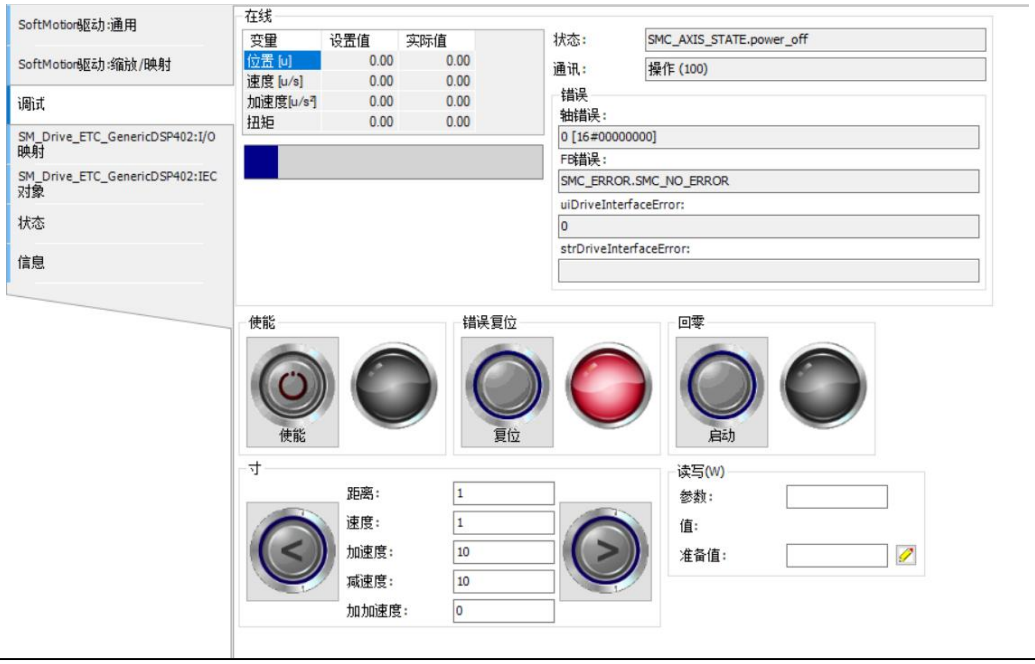
映射

自动映射：默认勾选，影响驱动器的 IEC 参数自动映射到设备的相应输入和输出。取消激活该选项后，可以手动编辑映射。

调试

调试物理驱动器时，此选项卡用于测试目的。仅在激活“在线配置模式 ”时可用。在这种模式下，开发系统已连接到设备。但是，不必下载应用程序。

在线



要求：PLC 处于在线模式	
变量表	具有变量名称，设置值和实际值的驱动器变量列表
状态	显示 SoftMotion 驱动器的当前状态
通讯	显示当前通讯状态
错误	轴错误: FB 错误: uiDriveInterfaceError: strDriveInterfaceError:

Operating elements

使能：驱动器使能（对比 MC_Power）

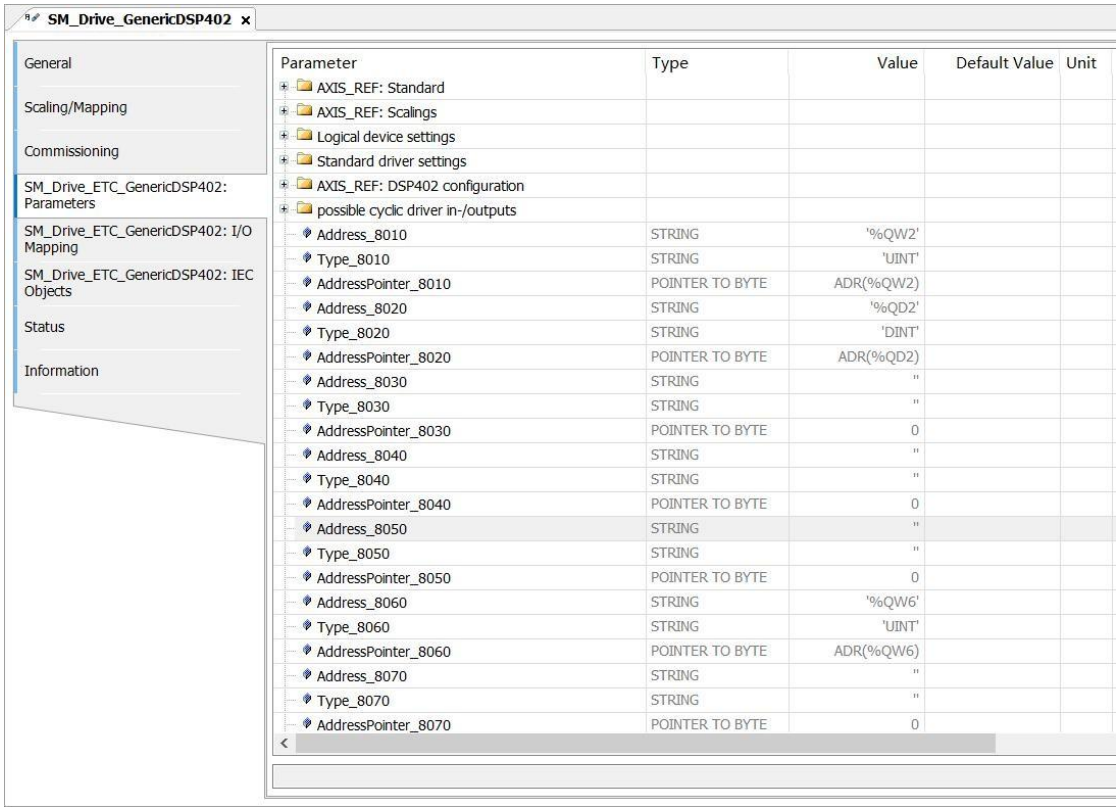
错误复位：出现错误后重置驱动器（对比 MC_Reset）

回零：使用驱动器中的设置参数执行原点回归（对比 MC_Home）

寸：通过“<”、“>”开关，驱动器可以根据 距离，速度，加速度，减速度和加加速的指定值向前和向后移动。

读写：对于指定的驱动器参数，将从 PLC 读取当前值并显示。在“ 准备值”中， 可以指定一个新值并通过小按钮（对比 MC_ReadParameter，MC_WriteParameter）写入驱动器中的参数。

Parameters（参数）



SM_Drive_ETC_GenericDSP402：I/O 映射

总线周期任务：有关更新总线的设备对象的定义。默认值在设备描述中定义：

- Use Parent bus cycle setting（使用上级设备设置）根据上级设备的设置进行更新。
- EtherCAT Task：使用 EtherCAT Task 更新总线的设备对象。
- Main Task：使用 Main Task 更新总线的设备对象。

SM_Drive_ETC_GenericDSP402：IEC 对象

在通用设备编辑器的此选项卡中，列出了“对象”，这些对象允许从 IEC 应用程序访问设备。在线模式下，它用作监视视图。

状态

在线模式下，显示轴状态。

信息

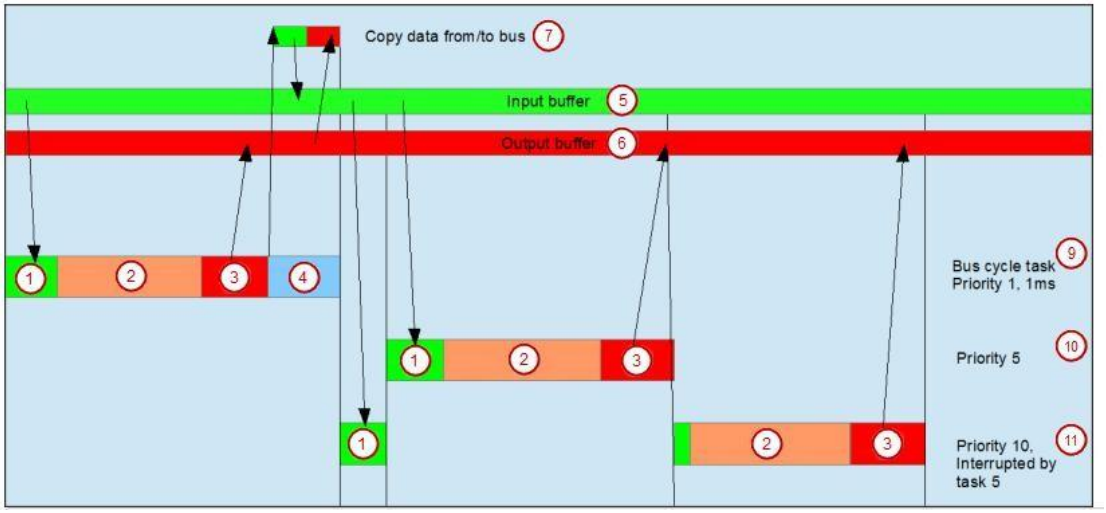
显示轴信息。

4.3.4 EtherCAT 总线周期行为

通常，对于每个 IEC 任务，在每个任务(1)的开始处读取使用的输入数据，并且在任务结束时将输出数据传送到 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。

可以为 PLC 设置中的所有现场总线定义 PLC 的总线循环任务。但是，对于某些现场总线，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(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) 复制数据到/从总线　　(9) 总线周期任务，优先级 1, 1 ms
(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

4.3.5 Ether CAT 具体变量

如果主站设备被插入到设备树中，则 EtherCAT_Master 任务被插入到当前应用的任务配置中。如往常，POU 调用可以添加到任务配置中。EtherCAT 具体布尔变量可以在 POU 中设定来影响应用上下文中 EtherCAT 配置行为：

1）支持可选设备

应用中 EtherCAT 设备的丢失在启动总线时造成防止堆栈加载的错误，在第一个 PLC 周期开始设置变量

```
<instance name of EtherCAT master>.StartConfigWithLessDevice := TRUE;
```

后，丢失的设备被当作不会影响正常堆栈启动过程的可选设备处理。2）

抑制额外的消息调度

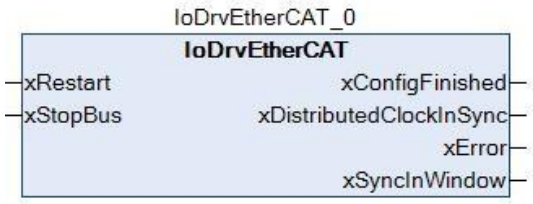
为了尽快刷新输出，EtherCAT 主站给每个单独的任务发送自己的消息。但是，如果附属驱动单元与实时输出数据同步，那么总线循环任务应该是唯一允许设置输出的任务。附加 消息会扰乱同步。为了抑制附加任务消息，

```
<instance name of EtherCAT master>.EnableTaskOutputMessage := FALSE;
```

必须在第一个 PLC 周期中设置一次。

4.3.6 EtherCAT 库

主站实例



为每个插入到设备树的 EtherCAT 主站创建类型为 IoDrvEtherCAT 的实例。实例的名称对应于设备树中主服务器的名称。实例的可用性显示在设备编辑器的 IEC 对象选项卡中。

输入

名称	数据类型	描述
xRestart	BOOL	上升沿重新启动主服务器，重新加载所有配置参数。
xStopBus	BOOL	TRUE：通讯停止。不会再发送 EtherCAT 报文。之后，大多数设备都需要重新启动，因为它们已切换到错误状态。

输出

名称	数据类型	描述
xConfigFinished	BOOL	TRUE:完成所有配置参数的传输而没有错误，通信正在总线上 运行。
xDistributed ClockInSync	BOOL	如果使用分布式时钟， 则 PLC 与第一个激活 DC 选项的 EtherCAT 从站同步。 只要同步成功完成，输出就变为 TRUE。然后可以使用该信号例如，仅当 PLC 处于同步模式时才激活 SoftMotion 功能块， 否则可能发生位置跳变。 当启动 PLC 时，输出为 FALSE，并在几秒后变为 TRUE。如果同步因一些错误而丢失，则输出被复位为 FALSE。
xError	BOOL	输出变为 TRUE 如果 • EtherCAT 堆栈启动期间发生错误. • 与从站的通信中断,因为无法接收其他消息(例如由于电缆断裂)。
xSyncInWindow	BOOL	如果同步窗口监视选项被激活并且所有从站的同步都在Sync Window 内，则输出变为 TRUE。

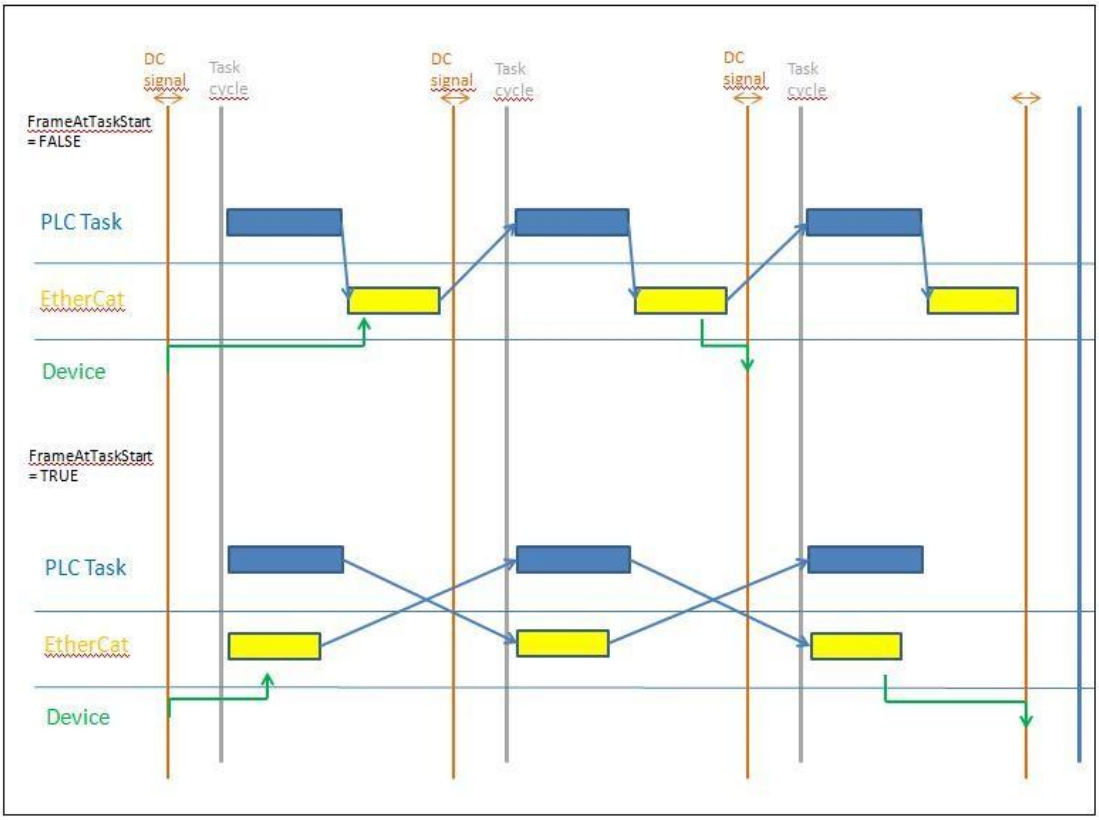
例如

- 通过 xRestart 变量启动主服务器的重启:
  ```
  EtherCAT_Master();
  EtherCAT_Master.xRestart := xRestart;
  ```
- 通过 xStop 变量停止总线上的通信:
  ```
  EtherCAT_Master.xStopBus := xStop;
  ```
- 调用主站以获取有关下载配置参数成功的信息:
  ```
  EtherCAT_Master();
  xFinish := EtherCAT_Master.xConfigFinished;
  ```

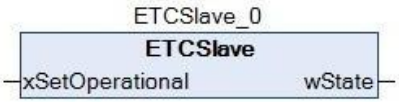
主站的属性:

AutoSetOperational	TRUE:如果通信中断,主站立即尝试重启从站 默认:FALSE
ConfigRead	TRUE:读取配置已完成，用户可以编辑设置，例如为了添加自定 义 SDO。
DCInSyncWindow	XDistributedClockInSync 的时间窗口。抖动必须在窗口内，以便 XDistributedClockInSync 输出信号一直保持为 TRUE。 默认:50 毫秒(当使用 CODESYS SoftMotion 时为 200 微秒)
DCClockReferenceTime	使用"分布式时钟"返回第一个从站的当前时间。此时间是所有其他从站和 PLC 本身的参考时间。
DCIntegralDivider	分布式时钟控制环路的积分因子 默认:20
DCPropFactor	分布式时钟控制环路的比例因子 默认:25
DCSyncToMaster	主站上的分布式时钟同步。如果设定为 TRUE，则所有从站都同

	步到主站而不是同步到 PLC 的第一个从站。 默认:FALSE
DCSyncToMasterWith SysTime	主站上的分布式时钟同步。 TRUE:所有从站都与主站的系统时间同步，从 SysTimeRtcHighResGet 读取的时间被用于把PLC 的系统时间分置到所有 EtherCAT 从站。 默认:FALSE
EnableTaskOutput Message	EtherCAT 信号通常由总线周期任务调度，另外还从使用从属输出的每个任务调度。在总线循环任务中，所有输出均已写入，所有 输入均已读取。在其他任务中，输出又传输一次，以便立即将其 写入相应的从站。以此方式尝试保持停滞时间直到尽可能短地写 入。这与分布式时钟一起会在某些设备中引起问题，例如，如果 伺服控制器未与 Sync 中断同步，而是使用写入时间进行内部同步。在这种情况下，一个周期内可能发生多次写访问。如果EnableTaskOutputMessage 设置为 FALSE，则仅使用总线循环任务。进一步的任务将不会导致进一步的消息。 默认： TRUE
FirstSlave	指向主站下方第一个从站的指针
FrameAtTaskStart	TRUE：从站的帧在任务开始时(在 IEC 任务之前)传输，这确保了最小的抖动。该命令用于实现伺服驱动器的无冲击运动。如果此 标志设为 TRUE，输出缓存器的框架在下个周期被写入(见下方图表)。 默认：FALSE (使用 CODESYS SoftMotion 时为 TRUE)
LastInstance	指向已连接主站列表的指针->之前的主站。
LastMessage	此属性返回一个字符串，其中包含 EtherCAT 堆栈的最后一条消息。如果启动成功完成，返回为 All slaves done。使用与在线模式下 EtherCAT 主设备编辑器窗口中显示的诊断消息相同的字符串。
NextInstance	指向已连接主站的列表的指针->下一个主站。
NumberActiveSlaves	此属性返回实际连接的从站数。 StartConfigWithLessDevice :=TRUE:检测到物理设备的编号。
OpenTimeout	打开网络适配器超时。默认值为 4 秒。
SplitFrame	用于将框架分成两部分。第一部分包含过程数据，第二部分包含异步邮箱通信和状态标志。由于分离，过程数据被更早地接收， 因此 PLC 的抖动对从设备的影响较小。
StartConfigWithLess Device	用于影响堆栈的启动行为。例如，如果在项目中配置了五个伺服 控制器，但仅连接了三个，则 EtherCAT 堆栈通常会停止。但是如果在第一个周期中，StartConfigWithLessDevice := TRUE，那么堆栈仍将尝试启动。这样，例如可以执行 10 个伺服控制器的通用配置，但保持实际连接的数量可变。请注意，无论如何都要检 查每个从站的供应商 ID 和产品 ID。如果发现差异，则堆栈停止



从站实例



为每个插入到设备树的 EtherCAT 从站生成 ETCSlave 数据类型的实例。实例的名称对应于设备树中从属的名称，实例的可用性显示在设备编辑器中的 IEC 对象选项卡。

从属实例在应用程序中使用，以便在运行时查询或更改从属的状态。

输入

名称	数据类型	描述
xSetOperational	BOOL	上升沿：尝试切换到 ETC_SLAVE_ OPERATIONAL 模式.

输出

名称	数据类型	描述
wState	ETC_SLAVE_STATE	从站的当前状态: • 0:ETC_SLAVE_BOOT • 1:ETC_SLAVE_Init • 2:ETC_SLAVE_PREOPERATIONAL • 4:ETC_SLAVE_SAVEOPERATIONAL • 8:ETC_SLAVE_OPERATIONAL 配置已成功完成.

输出

名称	数据类型	描述
		如果在配置期间发生错误,则从站可以回退到较早的状态.

从站的属性:

VendorID	EtherCAT 堆栈启动后,此属性将返回从设备读取的供应商 ID
ConfigVendorID	从配置中读取供应商 ID
ProductID	EtherCAT 堆栈启动后,此属性将返回从设备读取的产品 ID
ConfigProductID	从配置中读取产品 ID
SerialID	EtherCAT 堆栈启动后，此属性包含设备的序列号。
LastEmergency	如果收到消息，则此信息存储在从属服务器中，并且可以使用此属性从应 用程序中查询，还添加了一条日志消息。

注意： 如果在专家设置中激活厂商或产品 ID 的检查，则只要在 VendorID 和 ConfigVendorID 或 ProductID 和 ConfigProductID 之间发现一个差异，停止堆栈的启动。

检查所有从站的链式列表

为了监视程序中的各个从站，将调用实例，并通过确定状态 `wState`。为简化起见，可以通过链表确定所有主机和从机，并且可以通过简单的 WHILE 循环检查所有从机。主机和从机都存在该属性 `NextInstance`和 `LastInstance`。这些属性指向下一个或上一个从属。对于主服务器，还有一个附加属性 `FirstSlave`，它提供一个指向第一个从服务器的指针。根据以下示例，可以检查所有从站。

示例:

```
声明:
pSlave:POINTER TO ETCSlave;

程序:
pSlave := EtherCAT_Master.FirstSlave;
WHILE pSlave <> 0 DO
pSlave^();
IF pSlave^.wState = ETC_SLAVE_STATE.ETC_SLAVE_OPERATIONAL THEN
;
END_IF
pSlave := pSlave^.NextInstance;
END_WHILE
```

最开始，第一个从站通过 EtherCAT_Master.FirstSlave 被提取到主站。在 WHILE 循环中每一个主站被各自调用，wState 也因此指定，然后可以检查状态，指向下一个从站的指针通过 pSlave^.NextInstance 提取。如果列表完成，则指针为零，循环结束。

4.3.7 IODrvEtherCAT

如果支持和执行 EtherCAT 配置，则库自动集成到工程中。它包含用于读取和写入设备参数的功能块。因此能够检查甚至在 runtime 改变个体参数。有几个功能块可以同时处于活跃状态。循环中的各个请求在内部进行管理并连续处理。

ETC_CO_SdoRead

库:IODrvEtherCAT

该功能块用于读取 EtherCAT 从站参数。不像 ETC_CO_SdoRead4，也支持长于 4 字节的参数。要读取的参数用 Index 和 Subindex 指定，如对象目录中所使用的那样。

输入

名称	数据类型	描述
xExecute	BOOL	上升沿：开始读取从站参数。 为了在之后再次释放内部通道，实例必须被 xExecute:调用至少一次= FALSE。
xAbort	BOOL	TRUE：当前的读取过程中止。
usiCom	USINT	EtherCAT 主站的号码:如果仅使用一个 EtherCAT 主站则 usiCom 总是为 1。如果使用多个主站，1 指定第一个，2 指定第二个， 以此类推。
uiDevice	UInt	从站的物理地址。 如果在主站中取消激活自动配置模式，则可以为从站提供自己的 地址。必须在此处指定此地址。 如果自动配置模式被激活，第一个从站的地址为 1001。当前从站地址可以在 EtherCAT 地址区设备编辑器里的 Slave 对话框检验。
usiChannel	USINT	保留用于将来的扩展
wIndex	WORD	对象目录中参数的索引。
bySubindex	BYTE	对象目录中参数的子索引。
udiTimeOut	UDINT	监视看门狗时间的定义，以毫秒为单位。 如果在此时间到期时参数的读取尚未完成，则输出错误消息。
pBuffer	CAA_PVOID	指向数据缓冲区的指针，数据缓冲区在成功传输参数后存储数据
szSize	CAA_SIZE	数据缓冲区(pBuffer)的尺寸，以字节为单位。

输出

名称	数据类型	描述
xDone	BOOL	TRUE:完成参数读取没有错误。
xBusy	BOOL	TRUE:阅读尚未完成。
xError	BOOL	TRUE:读取期间发生错误。
eError	ETC_CO_ERROR	有关 xError 显示的错误原因的信息，例如超时时的 ETC_CO_TIMEOUT

名称	数据类型	描述
udiSdoAbort	UDINT	如果设备中发生错误，此输出将提供有关它的更多信息。
szDataRead	CAA_SIZE	读取的字节数;最大 szSize(输入)。

ENUM ETC_CO_ERROR

ETC_CO_NO_ERROR	0	没有错误
ETC_CO_FIRST_ERROR	5750	错误原因存储在输出 udiSdoAbort
ETC_CO_OTHER_ERROR	5751	找不到主站
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited 和尺寸> 4
ETC_CO_TIME_OUT	5753	超出时限
ETC_CO_FIRST_MF	5770	未使用
ETC_CO_LAST_ERROR	5799	未使用

ETC_CO_SdoRead4

库:IODrvEtherCAT

该功能块用于读取 EtherCAT 从站参数。不像 ETC_CO_SdoRead，只支持不长于 4 字节的参数。要读取的参数使用 Index 和 Subindex 指定，如对象目录中所使用的那样。

输入

名称	数据类型	描述
xExecute	BOOL	上升沿：开始读取从站参数。 为了在之后再次释放内部通道，实例必须被 xExecute:调用至少一次 = FALSE。
xAbort	BOOL	TRUE：当前的读取过程中止
usiCom	USINT	EtherCAT 主站的编号:如果一个 EtherCAT 主站被使用，usiCom 总是为 1。如果使用多个主站，1 指定第一个，2 指定第二个，以此类推。
uiDevice	UInt	从站的物理地址。 如果在主站中取消激活自动配置模式，则可以为从站提供自己的地址。必须在此处指定此地址。 如果自动配置模式被激活，第一个从站的地址为 1001。当前从站地址可以在 EtherCAT 设备编辑器里的从站对话框检验。
usiChannel	USINT	保留用于将来的扩展
wIndex	WORD	对象目录中参数的索引。
bySubindex	BYTE	对象目录中参数的子索引。
udiTimeOut	UDINT	监视时间的定义，以毫秒为单位。 如果在此时间到期时参数的读取尚未完成，则输出错误消息。

名称	数据类型	描述
xDone	BOOL	TRUE:完成参数读取没有错误。
xBusy	BOOL	TRUE:阅读尚未完成。
xError	BOOL	TRUE:读取期间发生错误。
eError	ETC_CO_ERROR	有关 xError 显示的错误原因的信息，例如超时时的 ETC_CO_TIMEOUT
udiSdoAbort	UDINT	如果设备发生错误，此输出将提供有关它的更多信息
abyData	ARRAY [1..4] OFBYTE	读取参数数据被复制到的 4 个字节的数组
usiDataLength	USINT	读取的字节数(1，2，4)。

ENUM ETC_CO_ERROR

ETC_CO_NO_ERROR	0	没有错误
ETC_CO_FIRST_ERROR	5750	错误原因存储在输出 udiSdoAbort 中
ETC_CO_OTHER_ERROR	5751	未发现主站
ETC_CO_DATA_OVERFLOW	5752	ETC_CO_Expedited 和尺寸> 4
ETC_CO_TIME_OUT	5753	超出时限
ETC_CO_FIRST_MF	5770	未使用
ETC_CO_LAST_ERROR	5799	未使用

ETC_CO_SdoRreadDWord

库:IODrvEtherCAT

类似 ETC_CO_SdoRead4，此功能块用于读取 EtherCAT 从站参数。然而，要读取的数据以DWORD(dwData)传输而不是数组。如果需要字节交换，则自动执行，因此读取的数据可以直接重复使用。

ETC_CO_SdoWrite

库:IODrvEtherCAT

此函数块用于编写 EtherCAT 从属参数。不像 ETC_CO_SdoWrite4，不长于 4 字节的参数可以被支持。要写入的参数由索引和子索引指定，如在对象目录中使用的那样。

输入

名称	数据类型	描述
xExecute	BOOL	上升 4 沿:启动从参数的读取。 为了在之后再次释放内部通道，实例必须被 xExecute:调用至少一次= FALSE。

输入

名称	数据类型	描述
xAbort	BOOL	TRUE:当前写入过程中止。
USICOM	USINT	EtherCAT 主站的编号：如果只使用一个 EtherCAT 主站， usiCom 总是为 1。如果使用多个主站，1 指定第一个，2 指定第二个，以此类推。
UIDevice	UINT	从站的物理地址。 如果在主站中取消激活自动配置模式，则可以为从站提供自己的地址。必须在此处指定此地址。 如果自动配置模式被激活，第一个从站的地址为 1001。当前从站的地址总是位于 EtherCAT 地址区内从站的选项卡。
usiChannel	USINT	保留用于将来的扩展
wIndex	字	对象目录中参数的索引。
bySubindex	字节	对象目录中参数的子索引。
udiTimeOut	UDINT	监视时间的定义，以毫秒为单位。 如果在此时间到期时参数的写入尚未完成，则输出错误消息
pBuffer	CAA_PVOID	指向包含要写入的数据的数据缓冲区的指针。
szSize	CAA_SIZE	数据缓存区(pBuffer)的大小，以字节为单位
eMode 输出 ENUM ETC_CO_MODE	ETC_CO_MODE	要写入的字节数。可能的输入: • ETC_CO_AUTO • ETC_CO_EXPEDITED • ETC_CO_SEGMENTED 通常设置 AUTO 模式，因此自动使用适合长度的模式。

名称	数据类型	描述
xDone	BOOL	TRUE:参数的写入完成且没有错误。
xBusy	BOOL	TRUE:写入尚未完成。
xError	BOOL	TRUE:写入期间发生错误。
eError	ETC_CO_ERROR	有关 xError 显示的错误原因的信息， 例如超时时的 ETC_CO_TIMEOUT
udiSdoAbort	UDINT	如果设备中发生错误，此输出将提供有关它的更多信息
szDataWritten	CAA_SIZE	写入的字节数;最大 szSize(输入)。

AUTO	0	客户端自动选择模式
EXPEDITED	1	客户端使用加速协议
SEGMENTED	2	客户端使用分段协议

ETC_CO_SdoWrite4

库:IODrvEtherCAT

该功能块用于写入 EtherCAT 从站参数。不像 ETC_CO_SdoWrite，只能支持不长于 4 字节的参数。要写入的参数由 Index 和 Subindex 指定，在对象目录中使用。

输入

名称	数据类型	描述
xExecute	BOOL	上升沿:开始读取从站参数。
xAbort	BOOL	TRUE:当前写入过程中止。
usiCom	USINT	EtherCAT 主站的编号:如果只使用一个 EtherCAT 主站，usiCom 总是为 1。如果使用多个母版，则"1"表示第一个，"2"表示第二个，依此类推。
uiDevice	UInt	从站的物理地址。 如果在主站中取消激活自动配置模式，则可以为从站 提供自己的地址。必须在此处指定此地址 如果自动配置模式被激活，第一个从站的地址总是为1001。当前从站的地址总是位于 EtherCAT地址 内从站的从站选项卡。
usiChannel	USINT	保留用于将来的扩展
wIndex	WORD	对象目录中参数的索引。
bySubindex	字节	对象目录中参数的子索引。
udiTimeOut	UDINT	监视时间的定义，以毫秒为单位。 如果在此时间到期时参数的写入尚未完成，则输出错 误消息。
abyData	ARRAY [1..4] BYBYTE	包含要写入的数据。 数据必须以英特尔字节顺序保存。
usiDataLength	USINT	要写入的字节数(1,2,4)。

输出

名称	数据类型	描述
xDone	BOOL	TRUE:参数的写入完成且没有错误。
xBusy	BOOL	TRUE:写入尚未完成。
xError	BOOL	TRUE:写入期间发生错误。
eError	ETC_CO_ERROR	有关 xError 显示的错误原因的信息， 例如超时时的 ETC_CO_TIMEOUT。
udiSdoAbort	UDINT	如果设备中发生错误，此输出将提供有关它的更多信息

ENUM ETC_CO_MODE

AUTO	0	客户端自动选择模式
EXPEDITED	1	客户端使用加速协议
SEGMENTED	2	客户端使用分段协议

ETC_CO_SdoWriteDWord

库:IODrvEtherCAT

就像 ETC_CO_SdoWrite4，此功能块用于写入 EtherCAT 从站参数。但是，要写入的数据不是作为数组传输，而是传递给 DWORD(dwData)。如果需要字节交换，则自动执行。因此可以直接指定要写入的值。

ReadMemory

库:IODrvEtherCAT

该功能块用于读取 EtherCAT 从站的存储器.

输入

名称	数据类型	描述
xExecute	BOOL	上升沿:开始读. 下降沿:复位输出. 如果在功能块完成命令之前出现下降沿,则输出继续正常工作.只有当命令未完全执行,中断(xAbort)或发生错误时他们才会被复位.在这种情况下相应的输出值(xDone, xError, iError) 恰好在输出中出现一个周期.
xAbort	BOOL	TRUE:命令立即中止,所有输出都设置为初始值.
USICOM	USINT	EtherCAT 主站的索引号(第一个主站为 1…)
wSlaveAddress	WORD	自动增加设备的地址或物理地址.
xAutoIncAdr	BOOL	地址解释标志
xBroadcast	BOOL	指示是否要使用广播读数的标志. TRUE:不使用 wSlaveAddress 和 bAutoIncAdr
uiMemOffset	UINT	EtherCAT 从存储器映像中的存储器偏移量
iSize	INT	要读取的字节数.
pDest	BYTE 指针	用于存储数据的缓冲区
udiTimeOut	IDINT	命令的看门狗时间,以 ms 为单位

输出

名称	数据类型	描述
xDone	BOOL	TRUE:读取完成没有错误.
xBusy	BOOL	TRUE:读取尚未完成.
xError	BOOL	TRUE:阅读时发生错误;功能块中止命令.
xAborted	BOOL	TRUE:命令被用户中止.

示例:读取寄存器 0x130(当前状态)

```
PROGRAM PLC_PRG
VAR
    etcreadmemory :ReadMemory;
    wStatus :WORD;
    xRead :BOOL;
END_VAR

etcreadmemory(xExecute := xRead, usiCom:=1, wSlaveAddress := 1002,
              xAutoIncAdr := FALSE, xBroadcast := FALSE, uiMemOffset := 16#130,
              iSize := 2, pDest := ADR(wStatus), udiTimeout := 500);
```

WriteMemory

库:IODrvEtherCAT

该功能块用于写 EtherCAT 从站的存储器.

输入

名称	数据类型	描述
xExecute	BOOL	上升沿:开始写 下降沿:重置输出. 如果在功能块完成命令之前出现下降沿,则输出继续正 常工作.只有当命令没有完全执行,中断(xAbort)或有错 误发生时,它们才会被复位.在这种情况下相应的输出值(xDone, xError, iError)恰好在输出中出现一个周期.
xAbort	BOOL	TRUE:命令立即中止,所有输出都设置为初始值.
usiCom	USINT	EtherCAT 主站的索引号(第一个主站为 1...)
wSlaveAddress	WORD	自动增加设备的地址或物理地址.
xAutoIncAdr	BOOL	地址解释标志
xBroadcast	BOOL	指示是否要使用广播读数的标志. TRUE:不使用 wSlaveAddress 和 bAutoIncAdr
uiMemOffset	UINT	EtherCAT 从站映像中的存储器偏移量
iSize	INT	要写入的字节数

输入

名称	数据类型	描述
pDest	POINTER OF BYTE	用于存储数据的缓冲区
udiTimeOut	IDINT	命令的看门狗时间,以 ms 为单位

输出

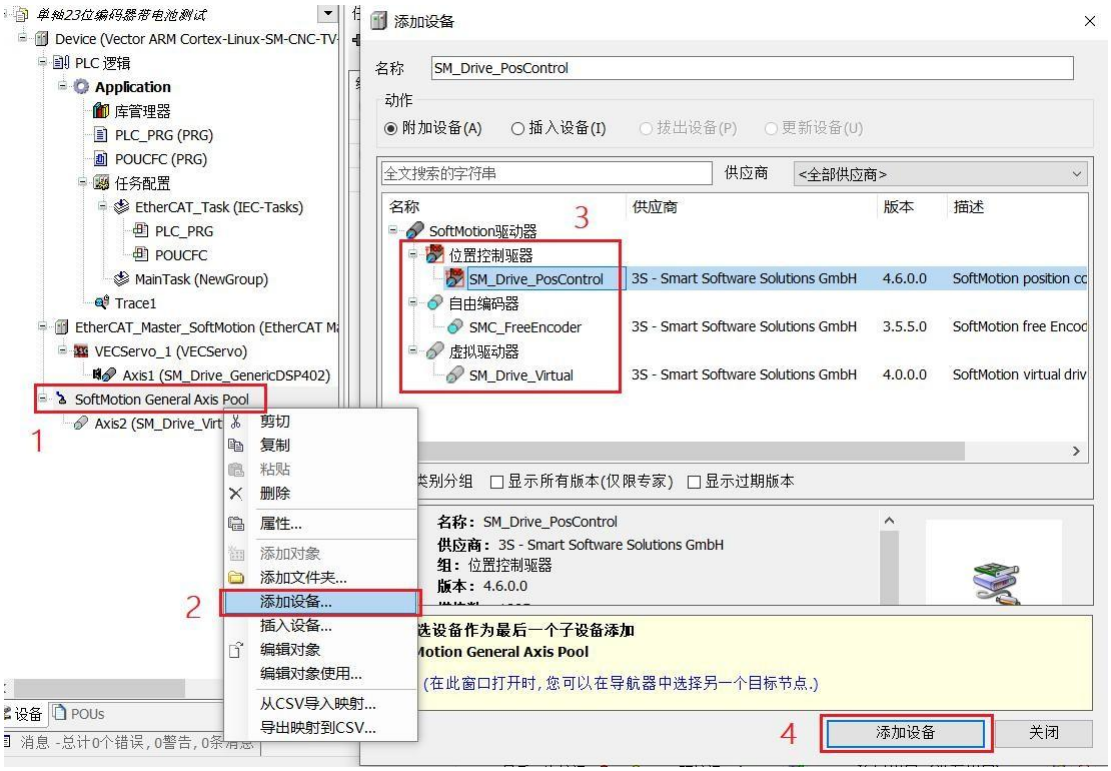
名称	数据类型	描述
xDone	BOOL	TRUE:写作完成没有错误.
xBusy	BOOL	TRUE:读取尚未完成.
xError	BOOL	TRUE:读取时发生错误;功能块中止命令.
xAborted	BOOL	TRUE:命令被用户中止.

4.3.8 SoftMotion General Axis Pool

如果使用 SoftMotion PLC（例如 CODESYS SoftMotion Win V3）则会在库管理器中自动链接基础库。这些类型的控制器提供了 SoftMotion General Axis Pool（通用轴池），可以在此处插入 SoftMotion 自由驱动器单元。

SoftMotion 驱动器接口是一个标准接口，可用于在 IEC 程序中链接，配置和寻址驱动器硬件。通过将不同的硬件映射到一个接口，可以轻松交换驱动器并重复使用 IEC 程序。该接口将驱动器耦合到 I / O 映射，并负责更新所需的运动数据并将其传输到驱动器控件。

添加 SoftMotion 自由驱动器方法如下图所示：



位置控制驱动器

使用 SM_Drive_PosControl驱动器控制，可以运行 CODESYS 轴的位置控制。要求是由设置的速度控制并返回其当前位置的设备。例如，它可以是具有位置反馈的速度控制设备（变频器）。

自由编码器

使用 SMC_FreeEncoder集成未永久耦合到 I/O 或硬件的编码器。

将编码器的输入值分配给变量<FREE_ENCODER_AXIS>.diEncoderPosition。这可以作为 IEC 代码或通过映射输入数据的存储器来实现。

虚拟驱动器

虚拟驱动器 SM_Drive_Virtual是软件中的模拟驱动器。可以在没有连接硬件的情况下测试程序或实现扩展功能。这些类型的功能包括，例如，轴运动的控制。

第 5 章 编程语言与参考

5.1 数据类型

CODESYS支持所有 IEC 61131-3 数据类型、对 IEC 61131-3 扩展的类型以及用户定义的数据类型。

IEC 61131-3 数据类型	对 IEC 61131-3 扩展的类型	用户定义的数据类型
BOOL	UNION	ARRAY
Integer	BIT	Structure
REAL / LREAL	UXINT and XWORD	Enumerations
STRING	Reference	Reference
WSTRING	Pointers	Pointers
Time		Subrange Types
LTIME		Identifiers

5.1.1 BOOL 布尔类型

数据类型	值	内存大小
BOOL	TRUE (1), FALSE (0)	8 bit

5.1.2 Integer 整型

CODESYS软件提供以下整型数据类型。

数据类型	下限	上限	内存大小
BYTE	0	255	8 bit
WORD	0	65535	16 bit
DWORD	0	4294967295	32 bit
LWORD	0	2^{64}-1	64 bit
SINT	-128	127	8 bit
USINT	0	255	8 bit
INT	-32768	32767	16 bit
UINT	0	65535	16 bit
DINT	-2147483648	2147483647	32 bit
UDINT	0	4294967295	32 bit
LINT	-2^{63}	2^{63}-1	64 bit
ULINT	0	2^{64}-1	64 bit

5.1.3 REAL/LREAL 浮点型

数据类型	下限	上限	内存大小
REAL	-3.402823E+38	3.402823E+38	32 bit

LREAL	-1.7976931348623157E+308	1.7976931348623157E+308	64 bit

5.1.4 STRING 字符串类型

STRING 数据类型的变量可以包含任何字符串。在声明期间分配的内存量涉及字符，并在括号或方括号中显示。如果未定义大小，则 CODESYS 默认情况下分配 80 个字符。通常，CODESYS 不会限制字符串的长度。但是，字符串函数只能处理 1 到 255 之间的长度。如果变量使用对于数据类型而言太长的字符串初始化，则 CODESYS 会从右侧相应地截断该字符串。STRING 变量所需的内存始终是每个字符一个字节加上一个附加字节（例如，对于 STRING[80] 声明，为 81 个字节）。

例：35 个字符的字符串声明示例:

```
1 str : STRING(35):= 'This is a String';
```

5.1.5 WSTRING

与 STRING（ASCII）数据类型相反，WSTRING 数据类型以 Unicode 格式解释。作为此编码的结果，WSTRING 的显示字符数取决于字符。WSTRING 的长度为 10 意味着 WSTRING 的长度最多可以包含 10 个字。但是，对于 Unicode 中的某些字符，编码一个字符需要多个WORD，这样字符的数量不必与 WSTRING 的长度相对应（在这种情况下为 10）。数据类型每个字符需要 1 WORD 的内存，再加上 1 WORD 的额外内存。每个 STRING 只需要 1 个字节。数据类型 WSTRING 以终止 0。

例：

```
1 wstr : WSTRING := "This is a WString";
```

5.1.6 TIME 时间类型

时间数据类型在内部被视为 DWORD 。TIME 和 TIME_OF_DAY 以毫秒为单位，DATE_AND_TIME 解析，以秒为单位解析。

Data type	Lower limit	Upper limit	Storage space	Resolution
TIME	T#0d0h0m0s0ms	T#49d17h2m47s295ms	32 bit	ms
TIME_OF_DAY	TOD#0:0:0.000	TOD#23:59:59.999	32 bit	ms
TOD	00:00:00.000	23:59:59.999		
DATE	D#1970-1-1 01/01/70	DATE#2106-2-7 February 07, 2106	32 bit	Day
DATE_AND_TIME DT	DT#1979-1-1-00:00:00 01/01/1970 00:00:00	DT#2106-2-7-6:28:15 February 07, 2106 6:28:15	32 bit	Seconds

5.1.7 LTIME

LTIME 用作高分辨率计时器的时基。高分辨率计时器的分辨率以纳秒为单位。

Data type	Lower limit	Upper limit	Storage space
LTIME	0	213503d23h34m33s709ms551us615ns	64 bit

使用方法：`LTIME#<time declaration>`

时间声明可以包括适用于 TIME 常量的时间单位，以及

- “us”: microseconds，微秒
- “ns”: nanoseconds，纳秒

例:

```
1 LTIME1 := LTIME#1000d15h23m12s34ms2us44ns;
```

5.1.8 UNION 联合声明

UNION 是通常包含不同数据类型的数据结构。

在一个联合中，所有组件都具有相同的偏移量，因此具有相同的内存量。在下面的联合声明示例中，对的赋值 name.a 也会影响 name.b。

例:

```
1 TYPE name;
2 UNION
3    a:LREAL;
4    b:LINT;
5 END_UNION
6 END_TYPE
```

5.1.9 BIT 位

只能将 BIT 数据类型用于结构或功能块中的单个变量。可能的值为 TRUE (1)和 FALSE (0)。一个 BIT 元素需要 1 位内存。因此，您可以按名称引用结构的单个位。BIT 连续声明的元素以字节为单位捆绑在一起。通过这种方式，您可以优化内存使用方式，而不是使用BOOL类型（每个类型保留 8 位）。另一方面，位访问明显更耗时。因此，BIT 仅在需要以预定义格式定义数据时，才应使用数据类型。

5.1.10 _UXIN 和 _XWORD 伪数据类型

CODESYS 支持具有 32 位和 64 位宽度的地址寄存器的系统。为了使 IEC 代码尽可能独立于目标系统，请使用伪数据类型 _UXINT 和 _XWORD 。编译器检查哪些目标系统类型是最新的，然后将这些数据类型转换为适当的标准数据类型。

_UXINT 在 64 位平台上转换为 ULINT， 在 32 位平台上转换为 UDINT。

_XWORD 在 64 位平台上转换为 LWORD，在 32 位平台上转换为 DWORD。

5.1.11 POINTERS 指针

指针的语法声明

指针在应用程序运行时存储变量，程序，功能块，方法和函数的地址。指针指向所提到的对象之一或任何数据类型的变量。指针声明的语法：

```
<identifier>: POINTER TO <data type | function block | program | method | function>;
```

当取消引用指针时，将确定指针指向的地址的值。为了取消引用指针，请将内容运算符附加到指针标识符（请参见 pt^下面的示例）。

使用地址运算符 ADR，您可以将变量的地址分配给指针。

```
VAR
pt: POINT TO INT; (* 指针 pt 的声明 *) var_int1: INT: = 5; (*变量 var_int1 和 var_int2 的声明*) var_int2: INT;
END_VAR

pt: = ADR (var_int1) ; (*指针 pt 被分配给 var_int1 的地址*)
var_int2: = pt ^; (*通过取消指针 pt 的引用，将 var_int1 的值 5 分配给变量 var_int2 *)
```

注意：

如果使用了指向设备输入的指针， 则该访问（ 例如“`pTest := ADR(input);...invalid assignment target`”）被视为写访问。当生成代码时，这会导致编译器警告。如果需要这种构造，则必须首先将输入值（input）复制到具有写访问权的变量。在在线模式下，您可以通过单击“ 转到参考”命令从指针跳转到参考变量的声明位置。

指向外部函数的函数指针

CODESYS 支持替换 INDEXOF 运算符的函数指针，可以将这些指针传递给外部库。但是，CODESYS 不提供任何从开发系统中的应用程序内部调用函数指针的方法。用于注册回调函数（系统库函数）的运行时系统函数需要函数指针。根据注册了哪个回调，运行时系统会隐式调用相关函数（例如在 STOP 的情况下）。为了使这样的系统调用（运行时系统）成为可能，必须在 Build 选项卡中设置相应的对象属性。

可以将 ADR 运算符用于函数，程序，功能块和方法。CODESYS 输出指向函数的指针的地址，而不是函数的地址，因为函数的值可以在在线更改后更改。只要该功能存在于目标系统上，此地址就有效。

索引访问指针

在 CODESYS，索引访问“[ ]”输入 POINTER，STRING 和 WSTRING 变量是允许的。pint[i] 返回基本数据类型

- 对指针的索引访问是通过算术方式完成的：如果对 POINTER TO 变量使用索引访问，则 CODESYS 将偏移量计算 pint[i] = (pint +i * SIZEOF(base type))^。索引访问还导致指针的隐式取消引用。结果数据类型是指针的基本数据类型。注意 pint[7] != (pint + 7)^
- 当对类型为的变量使用索引访问时，将 STRING 在索引表达式的偏移量处获得字符。结果是一个 BYTE 类型。str [i]以 SINT（ASCII）返回字符串的第 i 个字符。
- 当对类型为的变量使用索引访问时，将 WSTRING 在索引表达式的偏移量处获得字符。结果是一个 WORD 类型。wstr [i]以 INT（Unicode）返回字符串的第 i 个字符。

注意：

1. DWORD 当指针是64 位指针时，即使在64 位平台上，两个指针之间的差异结果也是type。
2. 与指针相比，可以使用直接控制值的引用。
3. 可以通过隐式监视功能 CheckPointer 在运行时监视指针的内存访问。

5.1.12 REFERENCE 参考

REFERENCE 也是一个指针，但是它比 POINTER 具有一些优点：

- 易于使用：不必显式取消引用的引用（使用^）即可访问引用对象的内容。
- 用于传递值的更好语法：如果输入是 REFERENCE TO，则 a(refInput := value)不必显式编写 ADR(value)。
- 类型安全：与指针不同，对于引用，编译器在分配两个引用时会检查基本类型是否相同

可以按照以下语法声明引用：

```
<identifier> : REFERENCE TO <data type>
A : REFERENCE TO DUT;
B : DUT;
C : DUT;

A REF= B; // 对应于 A := ADR(B);
A := C; // 对应于 A^ := C;
```

检查有效参考，可以使用运算符 ISVALIDREF 来检查参考是否指向有效值，即不等于 0 的值。语法：

```
<Boolean variable> :=ISVALIDREF(<with REFERENCE TO <data type> declared ident ifier);
```

当引用指向有效值时，<布尔变量>为 TRUE；否则为 FALSE。例：

```
ivar:INT;
ref_int:REFERENCE TO INT;
ref_int0 : REFERENCE TO INT;
testref : BOOL := FALSE;
```

实现方式：

```
ivar: = ivar + 1;
ref_int REF = ivar;
ref_int0 REF = 0;
testref: = ISVALIDREF (ref_int) ; (*为 TRUE，因为 ref_int 指向不为零的 ivar *)
testref: = ISVALIDREF (ref_int0) ; (*为 FALSE，因为 ref_int0 设置为 0 *)
```

5.1.13 ARRAY 数组

数组是相同数据类型的数据元素的集合。CODESYS 支持固定长度或可变长度的一维和多维数组。数组类型分为：固定长度数组、数组数组以及可变长度的数组，可以在 POU 的声明部分或全局变量列表中定义数组。

固定长度数组

一维数组声明的语法：

```
<variable name> : ARRAY[ <dimension> ] OF <data type> ( := <initialization> )? ;
<dimension> : <lower index bound>..<upper index bound>
<data type> : elementary data types | user defined data types | function block types
// (...)? : Optional
```

二维数组声明的语法：

```
<variable name> : ARRAY[ <1st dimension> ( , <next dimension> )+ ] OF <data type> ( :=
<initialization> )? ;

<1st dimension> : <1st lower index bound>..<1st upper index bound>
<next dimension> : <next lower index bound>..<next upper index bound>
<data type> : elementary data types | user defined data types | function block types
// (...)+ : One or more further dimensions
// (...)? : Optional
```

索引限制是整数；数据类型的最大值 DINT。

数据访问语法：

```
<variable name>[ <index of 1st dimension> ( , <index of next dimension> )* ]
// (...)* : 0, one or more further dimensions
```

例一：

10 个整数元素的一维数组定义：

```
VAR
```

```
    aiCounter: ARRAY[0..9] OF INT;  //索引下限: 0 , 索引上限: 9
END_VAR
```

程序:

```
aiCounter: ARRAY[0..9]: = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]; //初始化:
iLocalVariable: = aiCounter [2]; //数据存取
```

数组中的值 20 被分配给局部变量 iLocalVariable。

例二:

二维数组定义:

```
VAR
    aiCardGame: ARRAY [1..2, 3..4] OF INT; //  第1维: 1到2, 第2维: 3到4
END_VAR
```

初始化程序:

```
aiCardGame: ARRAY [1..2, 3..4] OF INT: = [2 (10) , 2 (20) ];    // [10, 10, 20, 20]的简写
```

	1	2
3	10	10
4	20	20

数据存取程序:

```
iLocal_1: = aiCardGame [1, 3];        //分配10
iLocal_2: = aiCardGame [2, 4];        //分配20
```

例三:

三维数组定义:

```
VAR
    aiCardGame: ARRAY [1..2, 3..4, 5..6] OF INT;
END_VAR
```

第 1 维: 1 到 2

第 2 维: 3 到 4

第 3 维: 5 到 6

一共有: 2 * 2 * 2 = 8 个数组元素

初始化 1:

```
aiCardGame: ARRAY [1..2, 3..4, 5..6] OF INT: = [10, 20, 30, 40, 50, 60, 70, 80];
```

数据存取 1:

```
iLocal_1: = aiCardGame [1, 3, 5];     //分配10
iLocal_2: = aiCardGame [2, 3, 5];         //分配20
iLocal_3: = aiCardGame [1, 4, 5];         //分配30
```

```
iLocal_4: = aiCardGame [2, 4, 5];          //分配40
iLocal_5: = aiCardGame [1, 3, 6];          //分配50
iLocal_6: = aiCardGame [2, 3, 6];          //分配60
iLocal_7: = aiCardGame [1, 4, 6];          //分配70
iLocal_8: = aiCardGame [2, 4, 6];          //分配80
```

初始化 2:

```
aiCardGame: ARRAY [1..2, 3..4, 5..6] OF INT: = [2 (10) , 2 (20) , 2 (30) , 2 (40) ];
// [2 (10) , 2 (20) , 2 (30) , 2 (40) ]是 [10, 10, 20, 20, 30, 30, 40, 40]的简略写法;
```

数据存取 2:

```
iLocal_1: = aiCardGame [1, 3, 5];          //分配10
iLocal_2: = aiCardGame [2, 3, 5];          //分配10
iLocal_3: = aiCardGame [1, 4, 5];          //分配20
iLocal_4: = aiCardGame [2, 4, 5];      //分配20
iLocal_5: = aiCardGame [1, 3, 6];          //分配30
iLocal_6: = aiCardGame [2, 3, 6];          //分配30
iLocal_7: = aiCardGame [1, 4, 6];      //分配40
iLocal_8: = aiCardGame [2, 4, 6];          //分配40
```

例四：

用户定义结构的 3 维数组：

```
TYPE DATA_A STRUCT
iA_1 : INT; iA_2 : INT;
dwA_3 : DWORD; END_STRUCT END_TYPE

PROGRAM PLC_PRG VAR
aData_A : ARRAY[1..3, 1..3, 1..10] OF DATA_A;
END_VAR
```

该数组 aData_A 由总共 3 * 3 * 10 = 90 个数据类型的数组元素组成 DATA_A。

部分初始化：

```
aData_A : ARRAY[1..3, 1..3, 1..10] OF DATA_A := [(iA_1 := 1, iA_2 := 10, dwA_3 :=
16#00FF),(iA_1 := 2,iA_2 := 20, dwA_3 := 16#FF00),(iA_1 := 3, iA_2 := 30, dwA_3 := 16#FFFF)];
```

在该示例中，仅前三个元素被显式初始化。未明确分配初始化值的元素在内部使用基本数据类型的默认值进行初始化。这将以元素 aData_A[2, 1, 1]为开头的初始化 0 结构组件。

资料存取：

```
iLocal_1: = aData_A [1,1,1] .iA_1;         //分配1
dwLocal_2: = aData_A [3,1,1] .dwA_3;  //分配16＃FFFF
```

“array of arrays” 数组数组

声明“array of arrays”是多维数组的另一种语法。嵌套元素的集合而不是标注元素的尺寸。嵌套深度是无限的。

数组数组声明语法：

```
<variable name> : ARRAY[<first>] ( OF ARRAY[<next>] )+ OF <data type> ( := <initialization> )? ;

<first> : <first lower index bound>..<first upper index bound>
<next> : <lower index bound>..<upper index bound> // one or more arrays
<data type> : elementary data types | user defined data types | function block types
// (...)+ : One or more further arrays
// (...)? : Optional
```

数据访问语法：

```
<variable name>[<index of first array>] ( [<index of next array>] )+ ;
// (...)* : 0, one or more further arrays
```

例1：

```
PROGRAM PLC_PRG VAR
    aiPoints : ARRAY[1..2,1..3] OF INT := [1,2,3,4,5,6];
    ai2Boxes : ARRAY[1..2] OF ARRAY[1..3] OF INT := [ [1, 2, 3], [ 4, 5, 6]];
    ai3Boxes : ARRAY[1..2] OF ARRAY[1..3] OF ARRAY[1..4] OF INT := [ [ [1, 2, 3, 4], [5, 6, 7, 8 ], [9, 10,11, 12] ], [ [13, 14, 15, 16], [ 17, 18, 19, 20], [21, 22, 23, 24] ] ];
    ai4Boxes : ARRAY[1..2] OF ARRAY[1..3] OF ARRAY[1..4] OF ARRAY[1..5] OF INT;
END_VAR
```

```
aiPoints[1, 2] := 1200;
ai2Boxes[1][2] := 1200;
```

变量 aiPoints 和 ai2Boxes 收集相同的数据元素，但是声明的语法与数据访问的语法不同。

aiPoints	ARRAY [1..2, 1..3] OF INT	
aiPoints[1, 1]	INT	1
aiPoints[1, 2]	INT	1200
aiPoints[1, 3]	INT	3
aiPoints[2, 1]	INT	4
aiPoints[2, 2]	INT	5
aiPoints[2, 3]	INT	6

ai2Boxes	ARRAY [1..2] OF ARRAY [1..3] OF INT	
ai2Boxes[1]	ARRAY [1..3] OF INT	
ai2Boxes[1][1]	INT	1
ai2Boxes[1][2]	INT	1200
ai2Boxes[1][3]	INT	3
ai2Boxes[2]	ARRAY [1..3] OF INT	
ai2Boxes[2][1]	INT	4
ai2Boxes[2][2]	INT	5
ai2Boxes[2][3]	INT	6

可变长度的数组

在功能块，函数或方法中，可以在 VAR_IN_OUT 声明部分声明长度可变的数组。LOWER_BOUND和 UPPER_BOUND运营商提供了用于确定在运行时实际使用的数组的索引范围。

可变长度一维数组声明的语法:

```
<variable name> : ARRAY[*] OF <data type> ( := <initialization> )? ;

<data type> : elementary data types | user defined data types | function
block types
// (...)? : Optional
```

可变长度的多维数组声明的语法:

```
<variable name> : ARRAY[* ( , * )+ ] OF <data type> ( := <initialization> )? ;

<data type> : elementary data types | user defined data types | function
block types
// (...)+ : One or more further dimensions
// (...)? : Optional
```

计算极限指数的运算符语法:

```
LOWER_BOUND( <variable name> , <dimension number> )
UPPER_BOUND( <variable name> , <dimension number> )
```

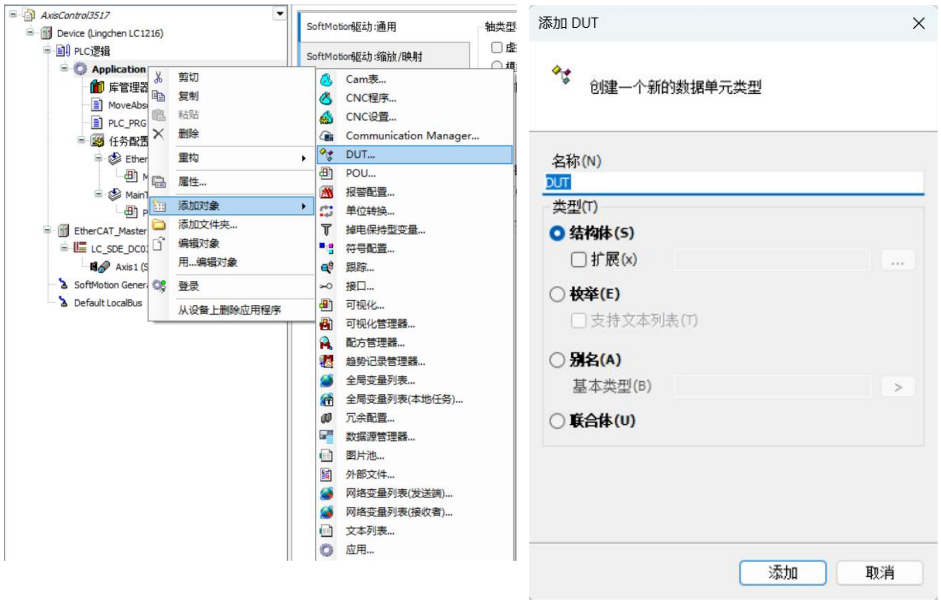
例 1：

该 SUM 函数将数组元素的整数值相加，并返回计算得出的总和作为结果。在运行时可用的所有数组元素中计算总和。由于仅在运行时才知道数组元素的实际数量，因此将局部变 量声明为可变长度的一维数组。

```
FUNCTION SUM: INT; VAR_IN_OUT
aiData : ARRAY[*] OF INT;
END_VAR
VAR
diCounter, diResult : DINT;
END_VAR

diResult := 0;
FOR
diCounter := LOWER_BOUND(aiData, 1) TO UPPER_BOUND(aiData, 1) DO //
Calculatesth length of the current array diResult := diResult + A[i];
END_FOR;
SUM := diResult;
```

5.1.14 Structure 结构体

通过单击“添加对象”，在具有 DUT 对象的项目中创建结构体。



结构声明开头的关键字 TYPE 和 STRUCT，并与关键字两端 END_STRUCT 和 END_TYPE。

结构声明的语法：

```
TYPE <structure name>: STRUCT
<variable declaration 1>
    ...
<variable declaration n> END_STRUCT
END_TYPE
```

<structure name>是 CODESYS 可以识别整个项目的类型，可以将其用作标准数据类型。也

可以使用嵌套结构。唯一的限制是不允许您将地址分配给变量（因为不允许 AT 声明）。

例 1:

```
TYPE polygonline:
STRUCT
    start:ARRAY [1..2] OF INT;
    point1:ARRAY [1..2] OF INT;
    point2:ARRAY [1..2] OF INT;
    point3:ARRAY [1..2] OF INT;
    point4:ARRAY [1..2] OF INT;
end:ARRAY [1..2] OF INT;
END STRUCT
```

初始化结构

例:

```
pPoly_1 : polygonline := ( start:=[3,3], point1:=[5,2], point2 := [7,3],
point3 := [8,5], point4 := [5,7], end :=[3,5]);
```

不允许使用带有变量的初始化。

访问结构体成员

可以使用以下语法访问结构成员:

```
<structure name>.<component name>
```

因此，可以在以上示例中使用 poly 1.start访问结构的 start组件 polygonline。

```
TYPE <structure name>: STRUCT
    <bit name bit1> : BIT;
    <bit name bit2> : BIT;
    <bit name bit3> : BIT;
    ...
    <bit name bitn> : BIT;
END_STRUCT END_TYPE
```

可以使用以下语法访问 BIT 结构成员:

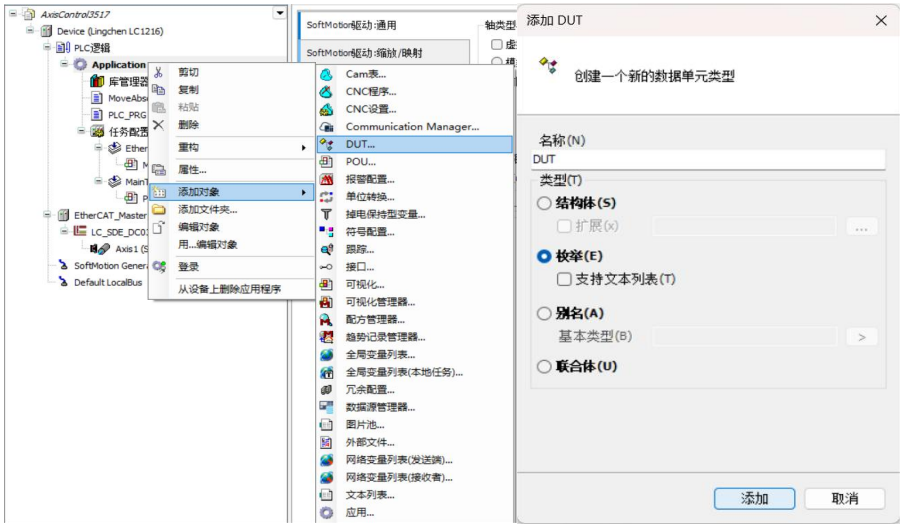
```
<structure name>.<bit name>
```

5.1.15 Enumerations 枚举

枚举是由用户定义的数据类型，由用于声明用户定义变量的一系列逗号分隔的组件（枚举值）组成。此外，可以使用枚举组件，例如常量，其标识符在项目中被全局识别。

```
<enumeration name>.<component name>
```

可以在 DUT 对象中声明一个枚举，通过单击“添加”在项目中创建该枚举。



声明语法：

```
( {attribute 'strict'} )? // Pragma optional but recommended
TYPE
    <enumeration name> :
    (<first component declaration>, ( <component declaration> ,)+<last component declaration>)( <basic data type> )? ( := <default variable initialization> )? ;
END_TYPE
( ... )? : Optional
<component declaration> : <component name> ( := <component initialization> )?
<basic data type> : INT | UINT | SINT | USINT | DINT | UDINT | LINT | ULINT | BYTE | WORD | DWORD LWORD
<variable initialization> : <one of the component names>
```

在枚举声明中，通常至少声明两个组件。但是，您可以声明任意多个。每个组件都可以分配自己的初始化。枚举自动具有基本数据类型 INT，但是您可以指定其他基本数据类型。此外，您可以在声明中指定一个组件，然后使用该组件初始化枚举变量。

语用{attribute 'strict'}引起如下所述的严格类型测试。

例 1:

```
{attribute 'qualified_only'}
{attribute 'strict'}
TYPE COLOR_BASIC :
    (yellow, green, blue, black);// 基本数据类型为INT，所有 COLOR_BASIC 变量的默认初始化为黄色
END_TYPE
```

5.1.16 Subrange Types 子范围类型

子范围类型是一种数据类型，其值范围是基本类型的子集。

声明的语法：

```
<name> : <int type> (<lowerlimit>..<upper limit>);
```

<name>	有效的 IEC 标识符
<int type>	子范围的数据类型（SINT，USINT，INT，UINT，DINT，UDINT，BYTE，WORD，DWORD LINT，ULINT，LWORD）
<lower limit>	范围的下限：必须与基本数据类型兼容的常量。下限也包括在该范围内。
<upper limit>	范围的上限：必须与基本数据类型兼容的常数。上限也包括在该范围内。

例1：

```
VAR
    i: INT (-4095..4095) ;
    ui: UINT (0..10000) ;
END_VAR
```

如果为声明或实现部分中的子范围类型分配的值不在该范围内（例如：i：= 5000）则 CODESYS 会发出错误消息。在运行时模式下，可以使用隐式监视功能 CheckRangeSigned 和监视子范围类型的范围限制 CheckRangeUnsigned。

5.2 变量

5.2.1 局部变量-VAR

局部变量在关键字之间 VAR和 END_VAR编程对象的声明部分中声明。可以使用实例路径对局部变量进行只读访问，也可以使用 attribute 关键

```
VAR
    iVar1: INT;
END_VAR
```

5.2.2 输入变量-VAR_INPUT

输入变量用于功能块的输入。

VAR_INPUT变量在关键字 VAR_INPUT和 END_VAR之间编程对象的声明

```
VAR_INPUT
    iIn1: INT; (*第一个输入变量*)
END_VAR
```

5.2.3 输出变量-VAR_OUTPUT

输出变量用于功能块的输出。

VAR_OUTPUT变量在关键字 VAR_OUTPUT和 END_VAR之间编程对象的声明部分中声明。

CODESYS 将此变量的值返回到调用 POU。可以在那里检索值并继续使用它们，也可以使用attribute关键字扩展输出变量。

例:

```
VAR_OUPUT
    iOut1: INT; (*第一个输出变量*)
END_VAR
```

函数和方法中的输出变量:

根据 IEC 61131-3 标准，功能和方法具有附加输出。调用函数时，必须分配这些其他输出，如下所示。

例:

```
fun (iIn1: = 1, iIn2: = 2, iOut1 => iLoc1, iOut2 => iLoc2) ;
```

5.2.4输入输出变量-VAR_IN_OUT

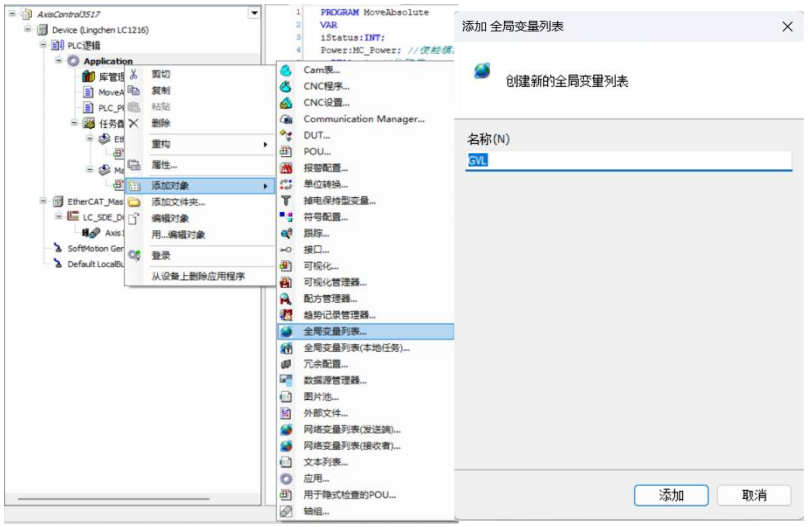
VAR_IN_OUT 变量是输入/输出变量，其是 POU 接口的一部分，并作为一个正式的引用

```
<keyword> <POU name>
 VAR_IN_OUT
    <variable name> : <data type> ( := <initialization value> )? ;
END_VAR
<keyword> : FUNCTION | FUNCTION_BLOCK | METHOD | PRG
```

5.2.5全局变量-VAR_GLOBAL

全局变量是在整个项目中可以识别的普通变量，常量，外部或剩余变量。

可以在全局变量列表中或在关键字 VAR_GLOBAL和 END_VAR之间的编程对象的声明部分中声明全局变量。



在变量名称前加一个点（例如.iGlobVar1）时，系统会识别一个全局变量。

注意:

如果在块中声明的局部变量与全局变量具有相同的名称，则它在块中具有优先级。CODESYS 始终在局部 POU 变量之前初始化全局变量

```
VAR_GLOBAL
    iVarGlob1: INT;
END_VAR
```

5.2.6 临时变量-VAR_TEMP

临时变量功能是 IEC 61131-3 标准的扩展。

可以在关键字 VAR_TEMP和 END_VAR之间在本地声明临时变量。VAR_TEMP声明仅在程序块和功能块中可用。每次调用该块时，CODESYS 都会初始化临时变量。应用程序只能在

```
VAR_TEMP
    iVarTmp1：INT;(*第一个临时变量*)
END_VAR
```

5.2.7静态变量-VAR_STAT

此功能是 IEC 61131-3 标准的扩展。

可以在关键字 VAR_STAT和 END_VAR之间在本地声明静态变量。首次调用每个块时，CODESYS 初始化静态变量。只能从声明了变量的名称空间中访问静态变量。但是当应用程序离开该块时，静态变量将保留其值。例如，您可以将静态变量用作函数调用的计数器。可 以使用

```
VAR_STAT
    iVarStat1：INT;
END_VAR
```

5.2.8 外部变量-VAR_EXTERNAL

外部变量是导入到块中的全局变量。

您可以在关键字 VAR_EXTERNAL和之间声明这些变量 END_VAR。如果全局变量不存在，则会显示一条错误消息。

注意：

```
<POU keyword> <POU name>
VAR_EXTERNAL
    <variable name> : <data type>;
END_VAR
```

该变量不允许初始化。

```
FUNCTION_BLOCK FB_DoSomething
VAR_EXTERNAL
        iVarExt1: INT;(*第一个外部变量*)
END_VAR
```

5.2.9 实例变量-VAR_INST

CODESYS 不会将 VAR_INST方法变量保存在方法堆栈中，而是保存在功能块实例的堆栈中。这意味着 VAR_INST变量的功能类似于功能块实例的其他变量，并且不会在每次调用该方法时重新初始化。

VAR_INST仅在方法中允许使用变量，并且只能在方法内访问这些变量。在方法的声明部分监视实例变量的变量值。

可以使用 attribute 关键字扩展实例变量。

例 1:

```
方法 meth_last: INT
VAR_INPUT
    iVar: INT;
END_VAR
VAR_INST
    iLast: INT: = 0;
END_VAR
meth_last: = iLast; iLast: =iVar;
```

5.2.10 配置变量-VAR_CONFIG

使用配置变量将完整地址分配给在功能块中声明的具有不完整地址的变量，这些变量将映射到设备 I/O 上。在 VAR_CONFIG和 END_VAR之间的全局变量列表中声明变量。全局变量列表称为“变量配置”，可以在其中键入具有完整实例路径和正确地址的配置变量。

```
FUNCTION_BLOCK locio
VAR
    xLocIn AT%I *: BOOL: = TRUE;
END_VAR
```

该 locio功能块在 PLC_PRG程序中使用:

```
PROGRAM   PLC_PRG
VAR
    locioVar1: locio;
END_VAR
```

全局变量列表中正确的变量配置如下：

```
VAR_CONFIG
  PLC_PRG.locioVar1.xLocIn AT%IX1.0: BOOL;
END_VAR
```

5.2.11 常数变数-VAR CONSTANT

常量变量在全局变量列表或编程对象的声明部分中声明。在实现中，可以通过实例路径以只读方式访问常量变量。

语法：

```
<scope> CONSTANT
<identifier> : <data type> := <initialization> ;
 END_VAR
<scope> : VAR | VAR_INPUT | VAR_STAT | VAR_GLOBAL
```

声明常数变量时，始终分配一个初始化值。这样就不能再写常量了。

例1:　　声明

```
VAR CONSTANT
    c_rTAXFACTOR : REAL := 1.19;
END_VAR
```

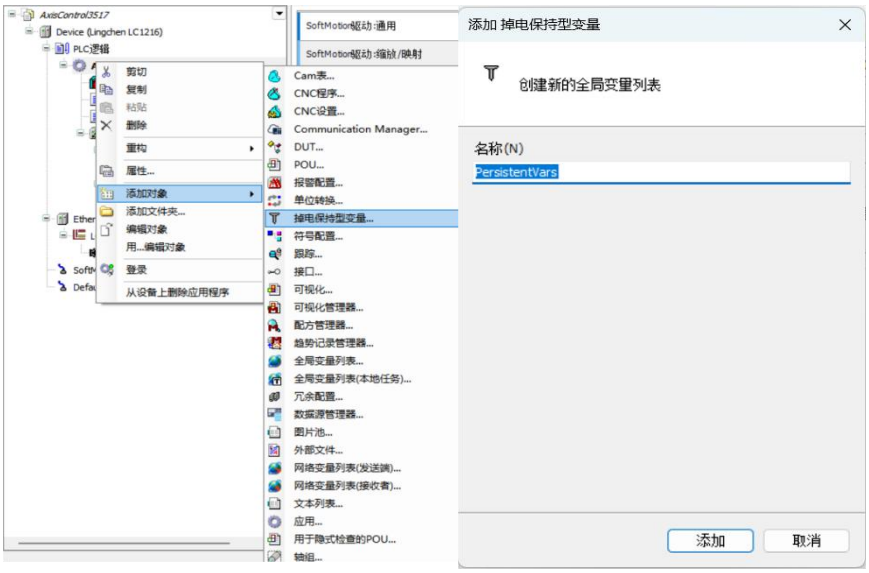
调用

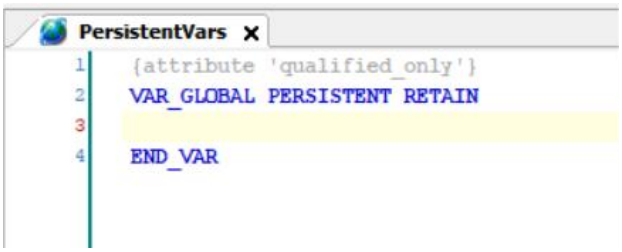
```
rPrice := rValue * c_rTAXFACTOR;
```

在实现中，可以仅以只读方式访问常量变量。常量变量位于赋值运算符的右侧。

5.2.12 持久变量-PERSISTENT

持久变量在持久全局变量列表的 VAR_GLOBAL RETAIN PERSISTENT声明部分中声明。对于在持久性编辑器外部用关键字标记的变量，实例路径将添加到其中。





一个变量声明PERSISTENT RETAIN与RETAIN PERSISTENT或 PERSISTENT具有相同效果。

全局持久变量列表中声明的语法 PersistentVars:

```
VAR_GLOBAL PERSISTENT RETAIN
    <identifier>: <data type> (:= <initialization>)?;
 <instance path to POU
variable> END_VAR
```

POU 中声明的语法:

```
<scope> PERSISTENT RETAIN
<identifier>: <datatype>( :=<initialization> )?;  // ( ... )? : Optional
 END_VAR
<scope> : VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | VAR_STAT | VAR_GLOBAL
```

注意:

1、永远不要在持久变量列表中使用 POINTER TO数据类型。如果再次下载该应用程序，其地址可能会更改。

2、不允许使用 AT关键字分配输入、输出或存储器地址。

3、如果您经常更改剩余变量的名称或数据类型，则最好 RETAIN仅使用关键字将它们声明为保留变量。

4、声明有两种方法:

a)直接在持久变量列表中声明变量，并避免插入实例路径。

b)在程序或功能块中的本地声明，并且在持久变量列表中添加实例路径（方法如下）。

这两种方法，后者占用了两倍的内存，并且还增加了循环时间。

直接在永久全局变量列表中声明	该变量是持久变量，位于受保护的内存区域中。
在程序中的本地声明，在持久变量列表中具有实例路径 在功能块中的本地声明，在持久变量列表中具有实例路径	该变量是持久变量，位于受保护的内存区域和内存中（双重分配）。
仅在程序本地 仅在功能块中本地	此变量不是持久性的。消息窗口中显示警告。 提示："单击声明▸添加所有实例路径"以将变量导入到持久变量列表中。
局部功能	该声明没有任何作用。此变量不是持久性的。

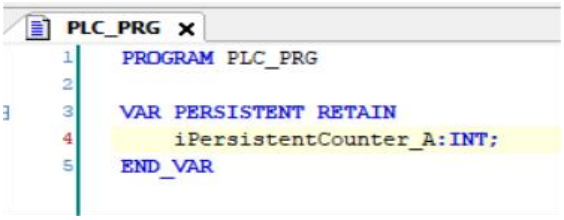
方法 a:

在持久变量列表中的声明 PersistentVars:

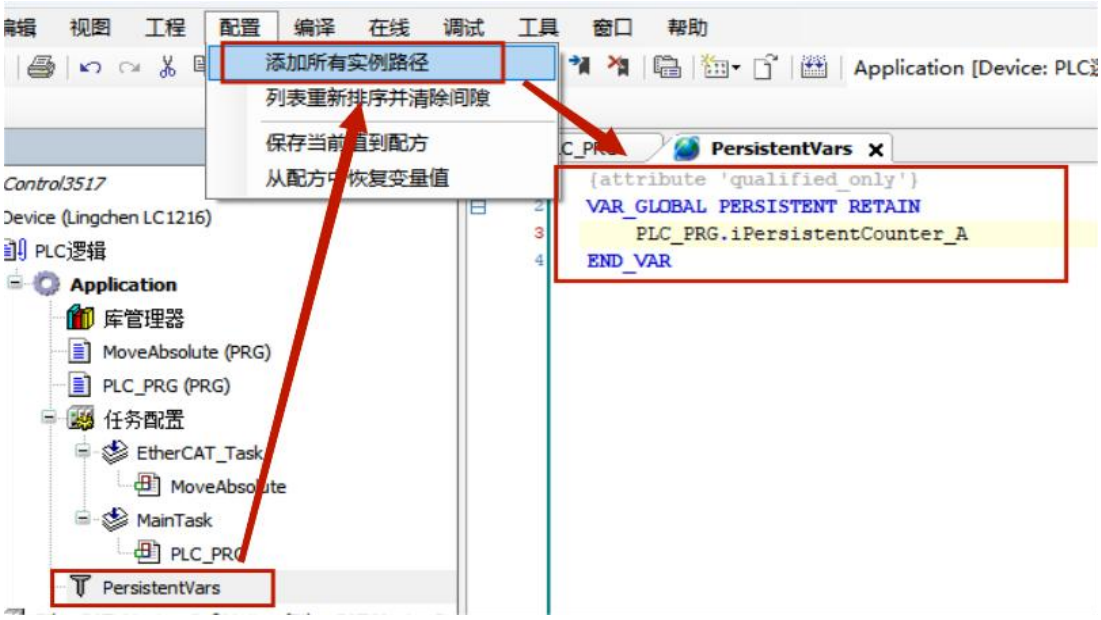
```
{attribute'qualified_only'}
VAR_GLOBAL
PERSISTENT RETAIN g_iCounter: INT;// 生 成 的 持 久 变 量
PLC_PRG.fb_A.iPersistentCounter_A 的实例路径: INT;
END_VAR
```

方法 b:

在本地或功能块中声明:



在持久变量列表中添加实例路径



尽可能避免使用标记在功能块中声明的变量 PERSISTENT。这是因为功能块实例会完全存储在剩余内存中，而不仅仅是标记的变量。

5.2.13 保留变量- RETAIN

保留变量的声明用关键字是RETAIN，声明范围 VAR、VAR_INPUT、VAR_OUTPUT、VAR_IN_OUT，VAR_STAT，或 VAR_GLOBAL。

```
<scope> RETAIN
   <identifier>: <data type> ( := <initialization> )? // ( ... )? : Optional
END_VAR
<scope> : VAR | VAR_INPUT | VAR_OUTPUT | VAR_IN_OUT | VAR_STAT | VAR_GLOBAL
```

注意：

不允许使用 AT关键字分配输入，输出或存储器地址。

声明的区域：

在程序本地	只有变量位于保留存储区域内。
全局在全局变量列表中	只有变量位于保留存储区域内。
在功能块中本地	功能块的整个实例及其所有数据位于保留存储区中。仅声明的保留变量受保护。
局部功能	即使变量也不位于保留存储区域内。该声明没有任何作用。
局部和持久地运行	即使变量也不位于保留存储区域内。该声明没有任何作用。

尽可能避免使用 RETAIN 标记功能块的变量。

5.2.14 特殊变量-SUPERA

SUPER 是一个特殊变量，用于面向对象的编程。

SUPER是功能块的指针，该指针指向从中生成功能块的基本功能块实例。因此，该指针还允许访问基本功能块（基本类）的方法的实现。SUPER指针是自动可用于每个功能块。SUPER只能在方法和相关的功能块实现中使用。

指针的解除引用： SUPER^

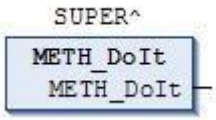
SUPER指针的使用：借助关键字 SUPER，可以调用在基本类或父类实例中有效的方法。

例 ：

ST：

```
SUPER^.METH_DoIt();
```

FBD / CFC / LD：



5.3 运算符

CODESYS 支持所有 IEC-61131-3 运算符。这些运算符在整个项目中被隐式识别。除这些 IEC 运算符外，CODESYS 还支持某些非 IEC 61131-3 标准的运算符。

运算符用于功能块等块中。包括算术运算符、位串运算符、位移位运算符、选择运算符、 比较运算符、地址运算符、呼叫运算符、类型转换运算符、数值运算符、命名空间运算符、多核运算符、其他运算符等。

算术运算符

“ ADD”

“ SUB”

“ MUL”

“ DIV”

'MOD'

“ MOVE”

“ INDEXOF”

“ SIZEOF”

位串运算符

“ AND”

“ OR”

“ XOR”

“ NOT”

“ AND_THEN”

“ OR_ELSE”

移位运算符

“ SHL”

“ SHR”

“ ROL”

“ ROR”

选择运算符

“ SEL”

“ MAX”

“ MIN”

“ LIMIT” “ MUX”

比较运算符

'GT'

“ LT”

“ LE”

'GE'

“ EQ”

'NE'

地址运算符

“ ADR”

“内容运算符”

“ BITADR”

呼叫运算符

“ CAL”

类型转换运算符

从较大的类型到较小的类型的隐式转换是不可能的（例如，从 INT to BYTE 或从DINT to WORD）必须使用特殊的类型转换才能将较大的类型转换为较小的类型。通常，您可以将任何基本类型转换为任何其他基本类型。

类型转换： `<elementary type1>_TO_<elementary type2>`

溢出转换： `TO_<elementary type2>`

数值运算符号

“ ABS”

“ SQRT”

“ LN”

“ LOG”

“ EXP”

“ EXPT”

“ SIN”

“ ASIN”

'COS'

“ TAN”

'ACOS'

“ ATAN”

5.3.1 算术运算符

“ADD”加法运算

该 IEC 运算符用于变量相加。

允许的数据类型：BYTE，WORD，DWORD，LWORD，SINT，USINT，INT，UINT，DINT，UDINT，LINT，ULINT，REAL，LREAL，TIME，TIME_OF_DAY，(TOD)DATEDATE_AND_TIME(DT)

TIME 数据类型的可能组合：TIME+TIME= TIME，TOD+TIME= TOD，DT+TIME=DT

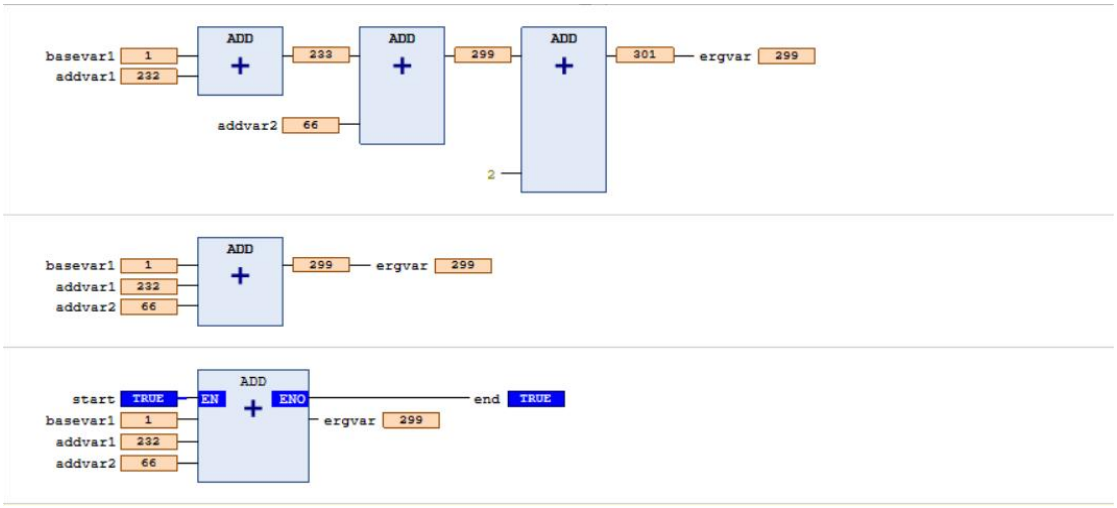
FBD / LD 编辑器中的功能：可以将 ADD 扩展到功能块输入。附加功能块输入的数量受到限制。

例：

ST：

```
var1: = 7 + 2 + 4 + 7;
```

FBD：



“MUL”乘法运算

该 IEC 运算符用于将变量相乘。

允许的数据类型：BYTE，WORD，DWORD，LWORD，SINT，USINT，INT，UINT，DINT，UDINT，LINT，ULINT，REAL，LREAL，TIME，

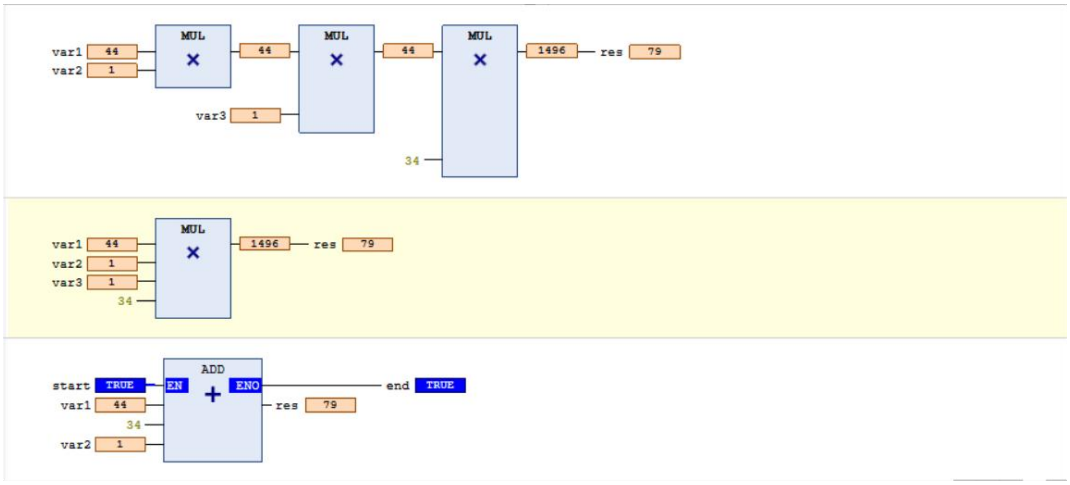
FBD / LD 编辑器中的功能：可以将 MUL 运算符扩展到其他功能块输入。附加功能块输入的数量受到限制。

例：

ST：

```
var1: = 7 * 2 * 4 * 7;
```

FBD：



“SUB”减法运算

该 IEC 运算符用于减去变量。

允许的数据类型：BYTE，WORD，DWORD，LWORD，SINT，USINT，INT，UINT，DINT，UDINT，LINT，ULINT，REAL，LREAL，TIME，TIME_OF_DAY，(TOD)DATEDATE_AND_TIME (DT)，

TIME 数据类型的可能组合：TIME-TIME= TIME，DATE-DATE= TIME，TOD-TIME= TOD，TOD-TOD= TIME，DT-TIME= DT，DT-DT=TIME

注意：

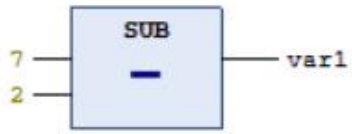
负 TIME 值未定义。

例：

ST：

```
var1: = 7-2;
```

FBD：



“DIV”除法运算

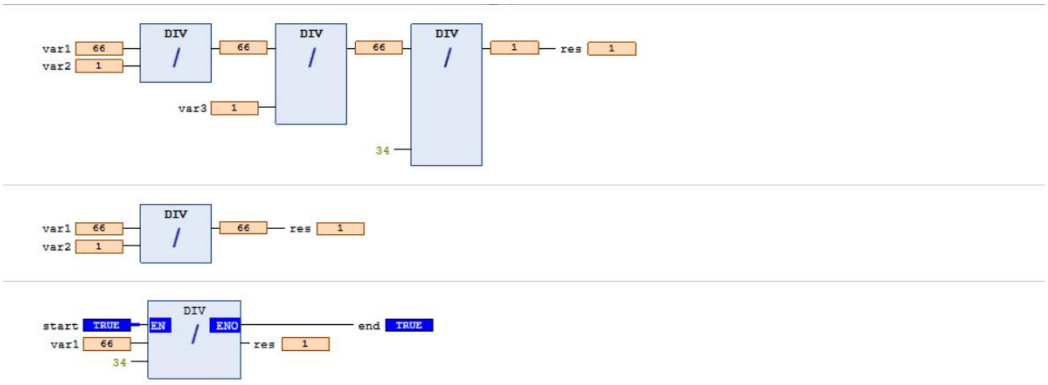
该 IEC 运算符用于划分变量。允许的数据类型：WORD，DWORD，LWORD，SINT，USINT，INT，UINT，DINT，UDINT，LINT，ULINT，REAL，除以零的结果可能会因目标系统而异。

例：

ST：

```
var1: = 8/2;
```

FBD:



“MOD”取余运算

该 IEC 运算符用于模除。

函数的结果是除法的整数余数。

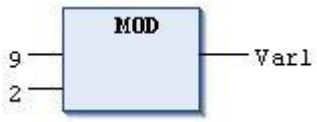
允许的数据类型：BYTE，WORD，DWORD，LWORD，SINT，USINT，INT，UINT，DINT，UDINT，LINT，ULINT。除以零的结果可能会因目标系统而异。

例:

ST:

```
var1: = 9 MOD 2; //结果: 1
```

FBD:



“MOVE”赋值运算

该 IEC 运算符用于将变量分配给相应类型的另一个变量。

由于该 MOVE块在 CFC，FBD 和 LD 编辑器中可用，因此还可以使用 EN / ENO 功能进行变量分配。

具有 EN / ENO 功能的 CFC：

CODESYS 仅在“ en_i”为 TRUE时才分配 var1的值到 var2。

ST:

```
ivar2 := MOVE(ivar1);
```

等效于:

```
ivar2 := ivar1;
```

“SIZEOF“字节运算

该运算符是 IEC 61131-3 标准的扩展。该运算符用于定义变量所需的字节数 x。

SIZEOF运算符总是产生一个无符号值。返回变量的类型适应于检测到的变量大小 x。

SIZEOF(x)的返回值	常数数据类型 (CODESYS 使用隐式检测大小)
0 <= size of x < 256	USINT
256 <= size of x < 65536	UINT
65536 <= size of x < 4294967296	UDINT
4294967296 <= size of x	ULINT

例:

ST:

```
arr1 : ARRAY[0..4] OF INT;
Var1 : INT;
var1 := SIZEOF(arr1);
(* var1 := USINT#10; *)
```

5.3.2 位串运算符

“AND”与

该 IEC 运算符用于按 AND 位操作数。

当输入位全都为 1 时，输出位为 1；否则，输出为 0。允许的数据类型：BOOL，BYTE，WORD，DWORD，LWORD

例子:

ST:

```
var1: = 2#1001_0011 AND 2#1000_1010;   //结果 var1 是 2#1000_0010
```

FBD:

“OR”或

该 IEC 运算符用于按 OR 位操作数。

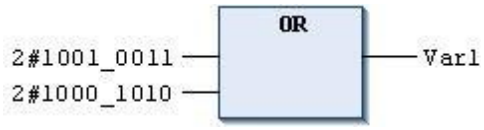
当至少一个输入位产生 1 时，输出位也产生 1；否则，输出位为 0。允许的数据类型：BOOL，BYTE，WORD，DWORD，LWORD

例子

ST：

```
Var1：= 2#1001_0011 OR 2#1000_1010;       //结果Var1 是 2#1001_1011
```

FBD：



“NOT”非

该 IEC 运算符用于 NOT 位操作数的按位。

当相应的输入位产生 0 时，输出位产生 1，反之亦然允许的数据类型：BOOL，BYTE，WORD，DWORD，LWORD

例：

ST：

```
var1：= NOT 2#1001_0011;          //结果var1：2#0110_1100
```

FBD：

“XOR”异或

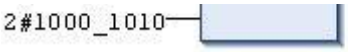
当且仅当两个输入位之一为 1 时，输出位也产生 1。当两个输入都为 1 或都为 0 时，输出产生 0。允许的数据类型：BOOL，BYTE，WORD，DWORD，LWORD

例：

ST：

```
var1：= 2#1001_0011 XOR 2#1000_1010;      //结果var1：2#0001_1001
```

FBD：



“AND_THEN”

当所有操作数都产生时 TRUE，操作数的结果也产生 TRUE；否则 FALSE。

“OR_ELSE”

当至少一个操作数产生时 TRUE，运算的结果也产生 TRUE；否则 FALSE。

5.3.3 移位运算符

“SHL”左移位

该 IEC 运算符用于将操作数向左移。

`erg := SHL (in, n)`

in：向左移动的操作数，n：in 向左移位的位数。此操作的位数 n，由输入变量 in 的数据类型定义。

例：

ST：

```
erg_word :=SHL(in_word,n)
```

```
PROGRAM shl_st
VAR
    in_byte : BYTE := 16#45;  (* 2#01000101 )
    in_word : WORD := 16#0045; (* 2#0000000001000101 )
    erg_byte : BYTE; erg_word : WORD; n: BYTE := 2;
END_VAR
erg_byte := SHL(in_byte,n);  (* Result is 16#14, 2#00010100 *)
erg_word := SHL(in_word,n);  (* Result is 16#0114, 2#0000000100010100 *)
```

FBD:

“SHR”右移位

该 IEC 运算符用于将操作数向右移。

`erg:=SHR(in,n)`

in：向右移动的操作数，n：in 向右移动的位数。

例：

ST：

```
PROGRAM shr_st
VAR
    in_byte : BYTE:=16#45;  (* 2#01000101 )
    in_word : WORD:=16#0045;  (* 2#0000000001000101 )
    erg_byte : BYTE; erg_word : WORD;
    n: BYTE :=2;
END_VAR
erg_byte := SHR(in_byte,n); (* Result is 16#11, 2#00010001 *)
erg_word := SHR(in_word,n); (* Result is 16#0011, 2#0000000000010001 *)
```

FBD:

“ROL”循环左移

该 IEC 运算符用于将操作数向左循环移动。允许的数据类型：BYTE，WORD，DWORD，LWORD

`erg := ROL (in, n)`

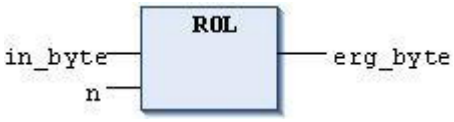
将 in次向左移动 n 位，然后将该位从右侧添加到最左侧的位置。由输入变量的数据类型定义 in。如果这是一个常数，则使用最小的数据类型。输出变量的数据类型仍然不会影响此操作。

例：

ST:

```
PROGRAM rol_st
VAR
    in_byte : BYTE := 16#45;
    in_word : WORD := 16#45;
    erg_byte : BYTE; erg_word : WORD;
    n: BYTE := 2;
END_VAR
erg_byte := ROL(in_byte,n); (* Result: 16#15 *)
erg_word := ROL(in_word,n); (* Result: 16#0114 *)
```

FBD:



“ROR”循环右移

该 IEC 运算符用于将操作数向右循环移动。允许的数据类型：BYTE，WORD，DWORD，LWORD

`erg := ROR(in,n)`

将 in 次向右移动 n 位，然后将该位从左侧添加到最右边的位置。由输入变量的数据类型定义 in。如果这是一个常数，则使用最小的数据类型。输出变量的数据类型仍然不会影响此操作。

例：

ST:

```
PROGRAM ror_st
VAR
    in_byte : BYTE := 16#45;
    in_word : WORD := 16#45;
    erg_byte : BYTE;
    erg_word : WORD;
    n: BYTE := 2;
END_VAR
erg_byte := ROR(in_byte,n); (* Result: 16#51 *)
erg_word := ROR(in_word,n); (* Result: 16#4011 *)
```

FBD:

5.3.4 选择运算符

“SEL”选择

IEC 运算符用于按位选择。

`OUT := SEL(G, IN0, IN1)`

等效于:

`OUT := IN0; if G = FALSE`

`OUT := IN1; if G = TRUE`

IN0, ... INn 和 OUT 的数据类型: 任何相同的数据类型, G: BOOL。

例:

ST:

```
Var1: = SEL (TRUE, 3,4) ;   (*结果: 4 *)
```

FBD:

“MAX”最大值

该 IEC 运算符用于最大功能。它产生两个值中的最大值。

`OUT := MAX(IN0, IN1)`

允许的数据类型: 全部

例:

ST:

```
Var1: = MAX (40, MAX (90,30) ;      结果: 90
```

FBD:

“MIN”最小值

该 IEC 操作符用于最小功能。它产生两个值中的最小值。

`OUT := MIN(IN0,IN1)`

允许的数据类型：全部

例：

ST：

```
Var1: = MIN (MIN (90,30) ,40) ;    “ 结果，30”
```

FBD：

“LIMIT”限制值

该 IEC 选择运算符用于限制。

OUT := LIMIT(Min, IN, Max)

相当于： OUT := MIN (MAX (IN, Min), Max)，Max 是结果的上限，Min 是结果的下限。如果该 IN 值高于 Max 上限，则 LIMIT 结果为 Max。如果的值 IN 低于最小 Min 下限，则结果为 Min。

允许的数据类型 IN 和 OUT：全部

例：

ST：

```
Var1: = LIMIT (30,90,80) ;    //结果Var1 是 80
```

“MUX”复用

该 IEC 运算符用作多路复用器。

`OUT := MUX(K, IN0,...,INn)`，相当于： `OUT = IN_K`

MUX 从一组值中选择第 K 个值。第一值是 K = 0。如果 K 大于其他输入的数量(n)，则传递最后一个值（INn）

允许的数据类型 K： BYTE，WORD，DWORD，LWORD，SINT，USINT，INT，UINT，DINT，LINT，ULINT，UDINT。

例：

ST：

```
Var1: = MUX (0,30,40,50,60,70,80) ;          //结果Var1为30。
```

5.3.5 比较运算符

“GT”大于

该 IEC 运算符用于“大于”功能。如果第一个操作数大于第二个操作数，则运算符产生结果 TRUE；否则 FALSE。

允许数据类型：任何基本数据类型。

例：

ST：

```
VAR1：= 20> 30；结果：FALSE
```

FBD：

“LT”小于

该 IEC 运算符用于“小于”功能。如果第一个操作数小于第二个操作数，则运算符产生结果 TRUE；否则 FALSE。

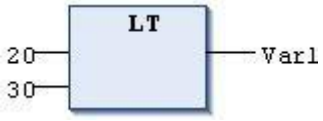
允许数据类型：任何基本数据类型。

例：

ST：

```
Var1：= 20 <30;   //结果：   TRUE
```

FBD：



“LE”小于或等于

该 IEC 运算符用于“小于或等于”功能。如果第一个操作数小于或等于第二个操作数，则运算符产生结果 TRUE；否则 FALSE。

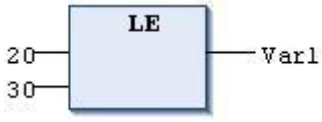
允许数据类型：任何基本数据类型。

例：

ST：

```
Var1：= 20 <= 30；结果 Var1：TRUE
```

FBD：



“GE”大于或等于

该 IEC 运算符用于“大于或等于”功能。如果第一个操作数大于或等于第二个操作数，则运算符产生结果 TRUE；否则 FALSE。

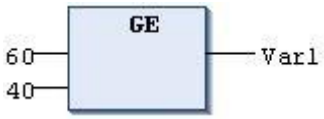
操作数的允许数据类型：任何基本数据类型。

例：

ST：

```
VAR1: = 60> = 40;     结果：  TRUE
```

FBD：



“EQ”等于

该 IEC 运算符用于“等于”功能。如果输入数相等，则运算符产生结果 TRUE，否则 FALSE。

允许数据类型：任何基本数据类型。

例：

结果：TRUE

ST：

```
VAR1: = 40 = 40;
```

FBD：

“NE”不等于

该 IEC 运算符用于“不相等”功能。如果操作数不相等，则运算符产生结果 TRUE；否则 FALSE。

允许数据类型：任何基本数据类型。

例：

ST：

```
Var1: = 40 <> 40;  结果 Var1是 FALSE
```

FBD：

5.3.6 地址运算符

“ADR”

该运算符是 IEC 61131-3 标准的扩展。ADR 会在中产生其参数的地址 DWORD。可以将此地址传递给制造商功能，也可以将它们分配给项目中的指针。

注意：

可以将 ADR 运算符与函数名称，程序名称，功能块名称和方法名称一起使用。因此，ADR 代替 INDEXOF 运算符。

ST：

```
FUNCTION_BLOCK FB_Address
VAR
    piAddress1: POINTER TO INT;
    iNumber1: INT := 5;
    lwAddress2: POINTER TO INT;
    iNumber2: INT := 10;
END_VAR

piAddress1 := ADR(iNumber1); // piNumber is assigned to address of iNumber1 lwAddress2 :=
ADR(iNumber2); // 64 bit runtime system
```

例：

```
VAR
    <address name> : DWORD | LWORD | POINTER TO < basis data type>
END_VAR
<address name> := ADR( <variable name> );
```

“Content Operator”

该运算符是 IEC 61131-3 标准的扩展。通过将运算符附加^到指针标识符，可以使用此运算符取消引用指针。使用指向地址的指针时，请注意，应用在线更改可能会移位地址内容。

例：

ST：

```
pt: POINT TO INT;
var_int1: INT;
var_int2: INT;
pt: = ADR (var_int1) ;
var_int2: = pt ^;
```

“BITADR”

该运算符是 IEC 61131-3 标准的扩展。BITADR 产生的段中的位偏移量 DWORD。偏移量取决于在目标系统设置中是否选择了字节寻址复选框。最高的半字节（4 位）DWORD 定义了存储范围：

`Flag: 16x40000000 Input: 16x80000000`

`Output: 16xC0000000`

使用指向地址的指针时，请注意，应用在线更改可能会移位地址内容。

例:

ST:

```
WHERE
    Var1 AT%IX2.3: BOOL;
    bitoffset: DWORD;
END_VAR
bitoffset: = BITADR (var1) ;
(*字节寻址= TRUE 时: 16x80000013, 字节寻址= FALSE 时: 16x80000023 *)
```

5.3.7 调用运算符

“CAL”调用

该 IEC 运算符用于调用功能块。在 IL 中，CAL 调用功能块的实例。

`CAL <function block> (<input variable1> := <value>,  <input variableN> := <value>)`

例:

Inst 用输入变量 Par1、Par2 以及分配的 0 或 TRUE 调用功能块的实例。

```
CAL Inst(Par1 := 0, Par2 := TRUE);
```

5.3.8 数值运算符

“ABS”绝对值

该 IEC 运算符得出数字的绝对值。

允许数据类型：任何数字基本数据类型

例：

ST：

```
i: = ABS (-2) ;   //结果 i 为 2
```

FBD：

“SQRT”开平方

该 IEC 当然会产生数字的平方根。

输入变量的允许数据类型：任何数字基本数据类型 输出变量允许的数据类型：REAL 或 LREAL

例：

ST：

```
q: = SQRT (16) ;      //结果 q: 4
```

FBD：

“LN”自然对数

该 IEC 运算符得出数字的自然对数。

输入变量的允许数据类型：任何数字基本数据类型。输出变量允许的数据类型：REAL 和 LREAL。

例：

ST：

```
Q = LN (45) ;        //结果:  3.80666
```

FBD:

“LOG”常数对数

该 IEC 运算符得出一个以 10 为底的对数。

输入变量的允许数据类型: 任何数字基本数据类型。输出变量允许的数据类型: REAL 和 LREAL。

例:

ST:

```
q: = LOG (314.5) ;      //结果 q: 2.49762
```

FBD:

“EXP”自然数 e 的指数

该 IEC 运算符产生指数函数。

输入变量的允许数据类型: 任何数字基本数据类型 输出变量允许的数据类型: REAL 和 LREAL

例:

ST:

```
q: = EXP (2) ;      //结果 q: 7.389056099
```

FBD:

“EXPT”幂（X 的 Y 次方）

该 IEC 运算符用于计算幂函数, power = base exponent。语法:

EXPT(<base>,<exponent>)

输入值数据类型的: SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, BYTE, WORD, DWORD, LWORD

返回值的数据类型: 浮点数类型（REAL 和 LREAL）

例 1:

带文字的幂函数

```
Var1 := EXPT(7,2);
```

FBD:

结果: Var1 = 49

例 2:

带变量的幂函数

```
PROGRAM PLC_PRG
VAR
    lrPow : LREAL;
    iBase : INT := 2;
    iExponent : INT := 7;
END_VAR

lrPow := EXPT(iBase, iExponent);  // 结果: lrPow = 128
```

“SIN”正弦函数

该 IEC 运算符得出数字的正弦值。

用于测量以弧度为单位的角度的输入变量, 允许数据类型: 任何数字基本数据类型 输出变量允许的数据类型: REAL 和 LREAL

例:

ST:

```
q: = SIN (0.5)        //结果q: 0.479426
```

FBD:

“COS”余弦函数

该 IEC 运算符得出数字的余弦值。

用于测量以弧度为单位的角度的输入变量, 允许数据类型: 任何数字基本数据类型 输出变量允许的数据类型: REAL 和 LREAL

例:

ST:

```
q: = COS (0.5) ;      //结果q: 0.877583
```

FBD:

“TAN”正切函数

该 IEC 运算符得出数字的正切值。

用于测量以弧度为单位的角度的输入变量，允许数据类型：任何数字基本数据类型　　输出变量允许的数据类型：REAL 和 LREAL

例：

ST：

```
q: = TAN (0.5)  //结果 q: 0.546302
```

FBD：

“ASIN”反正弦函数

该 IEC 运算符得出数字的反正弦值。

输入变量的允许数据类型：任何数字基本数据类型 输出变量允许的数据类型：REAL 和 LREAL

例：

ST：

```
q: = ASIN (0,5) ;     //结果为q:  0.523599
```

FBD：

“ACOS”反余弦函数

该 IEC 运算符得出数字的反余弦值。该值以弧度计算。输入变量的允许数据类型：任何数字基本数据类型

输出变量允许的数据类型：REAL 和 LREAL

例：

ST：

```
q: = ACOS (0.5) ;     //结果: q=1.0472
```

FBD：

“ATAN”反正切函数

该 IEC 运算符得出数字的反正切值。该值以弧度计算。

用于测量以弧度为单位的角度的输入变量，允许数据类型：任何数字基本数据类型　　输出变量允许的数据类型：REAL 和 LREAL

例：

ST：

```
q: =与 (0.5)     //结果 q: 0.463648
```

FBD：

5.3.9 类型转换运算符

“BOOL_TO”

IEC 运算符用于将 BOOL 数据类型转换为另一种数据类型。语法：

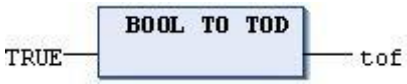
BOOL_TO_<data type>

当为数字类型数据类型时，布尔值为时 TRUE，结果为 1，为 FALSE 时，结果为 0。当为 STRING 数据类型，结果为 TRUE 或 FALSE。

例：

ST code	Result
i := BOOL_TO_INT(TRUE);	1
str := BOOL_TO_STRING(TRUE);	TRUE
t := BOOL_TO_TIME(TRUE);	T#1ms
tof := BOOL_TO_TOD(TRUE);	TOD#00:00:00.001
dat := BOOL_TO_DATE(FALSE);	D#1970
dandt := BOOL_TO_DT(TRUE);	DT#1970-01-01-00:00:01
FBD code	**Result**

FBD code	Result
	1

	
	TRUE
	T#1ms
	TOD#00:00:00.001
	D#1970-01-01
	DT#1970-01-01-00:00:01

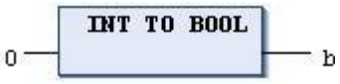
“TO_BOOL”

IEC 运算符用于将其他变量类型转换为 BOOL 变量。语法:

<data type>_TO_BOOL

当操作数不等于 0 时产生结果 TRUE。当操作数等于 0 时，产生结果 FALSE。对于 STRING 类型，如果操作数为“TRUE”则结果是 TRUE；否则是 FALSE。

例:

ST code	Result
b := BYTE_TO_BOOL(2#11010101);	TRUE
b := INT_TO_BOOL(0);	FALSE
b := TIME_TO_BOOL(T#5ms);	TRUE
b := STRING_TO_BOOL('TRUE');	TRUE
FBD code	**Result**
	TRUE
	FALSE
	TRUE
FBD code	**Result**
	TRUE

“TO_ <xxx>”

IEC 运算符用于将一种数据类型的变量转换为另一种数据类型。

`TO_<data type>`

从较大的数据类型转换为较小的数据类型时，信息可能会丢失。如果要转换的数值超出范围限制，则 CODESYS 将忽略该数值的前几个字节。例如，从 LREAL 转换到 DINT 输入值为负时，就是这种情况。

例:

ST 实现语言:

```
VAR
   iVar : INT; bVar : BOOL;
   sVar : STRING; rVar : REAL;
END_VAR
iVar := TO_INT(4.22);      (* Result:    4 *)
bVar := TO_BOOL(1);        (* Result: TRUE *)
sVar := TO_STRING(342);    (* Result: '342'  *)
rVar := TO_WORD('123');    (* Result: 123 *)
```

“<INT Type>_TO_<INT Type>”

将一种整数数据类型转换为另一种整数数据类型。

`<INT data type>_TO_<INT data type>`

注意，从较大的数据类型转换为较小的数据类型时，信息可能会丢失。如果要转换的数值超出范围限制，则 CODESYS 将忽略该数值的前几个字节。

例：

ST:

```
si := INT_TO_SINT(4223);  //结果: Result in si: 127
```

如果将整数 4223（十六进制表示为 16＃107f）另存为 SINT变量，则会为该变量分配值 127（十六进制表示为 16＃7f）。

FBD:

“REAL_TO- / LREAL_TO”

IEC 运算符用于将 REAL 和 LREAL 数据类型转换为另一种数据类型。

`REAL_TO_<data type> LREAL_TO_<data type>`

将操作数的实值向上或向下舍入为整数值，然后将其转换为相应的类型。(STRING, BOOL, REAL，和 LREAL 数据类型例外）。

Examples

ST code	Result
`i := REAL_TO_INT(1.5);`	2

j := REAL_TO_INT(1.4);	1
i := REAL_TO_INT(-1.5);	-2
j := REAL_TO_INT(-1.4);	-1
FBD code	**Result**
	2

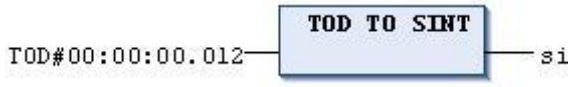
“TIME_TO / TIME_OF_DAY_TO”

IEC 运算符用于将 TIME 和 TIME_OF_DAY 数据类型转换为另一种数据类型。

<TIME data type>_TO_<data type>

在内部，CODESYS 将时间（以毫秒为单位）保存为 DWORD（TIME_OF_DAY 从 00:00 开始）。

CODESYS 转换此值。对于 STRING 数据类型，结果是时间常数。例：

ST code	**Result**
str := TIME_TO_STRING(T#12ms);	T#12ms
dw := TIME_TO_DWORD(T#5m);	300000
si := TOD_TO_SINT(TOD#00:00:00.012);	12
FBD code	**Result**
	T#12ms
	30000
	12

“DATE_TO / DT_TO”

IEC 运算符用于将 DATE 和 DATE_AND_TIME 数据类型转换为另一种数据类型。

<DATE data type>_TO_<data type>

在内部，CODESYS 将日期保存为 DWORD（自 1970 年 1 月 1 日起，以秒为单位）。CODESYS转换此值。对于 STRING 数据类型，结果是日期常量。

例：

ST code	Result
b := DATE_TO_BOOL(D#1970-01-01);	FALSE
i := DATE_TO_INT(D#1970-01-15);	29952
i := DT_TO_BYTE(DT#1970-01-15-05:05:05);	129
str := DT_TO_STRING(DT#1998-02-13-14:20);	DT#1998-02-13-14:20
FBD code	**Result**
	FALSE
	29952
	129
	DT#1998-02-13-14:20

“STRING_TO”

IEC 运算符用于将 STRING 数据类型转换为另一种数据类型。

STRING_TO_<data type>

该值必须是目标类型的有效常数（数字），允许在数字后加上其他字符（例如 23xy）。不允许在数字前添加其他字符。

例：ST code	Result
b := STRING_TO_BOOL('TRUE');	TRUE
w := STRING_TO_WORD('abc34');	0
w := STRING_TO_WORD('34abc');	34
t := STRING_TO_TIME('T#127ms');	T#127ms
r := STRING_TO_REAL('1.234');	1.234
bv := STRING_TO_BYTE('500');	244
FBD code	**Result**
	TRUE

“TRUNC”

IEC 运算符用于将 REAL 数据类型转换为 DINT 数据类型。CODESYS 仅采用数字的整数部分。

例:

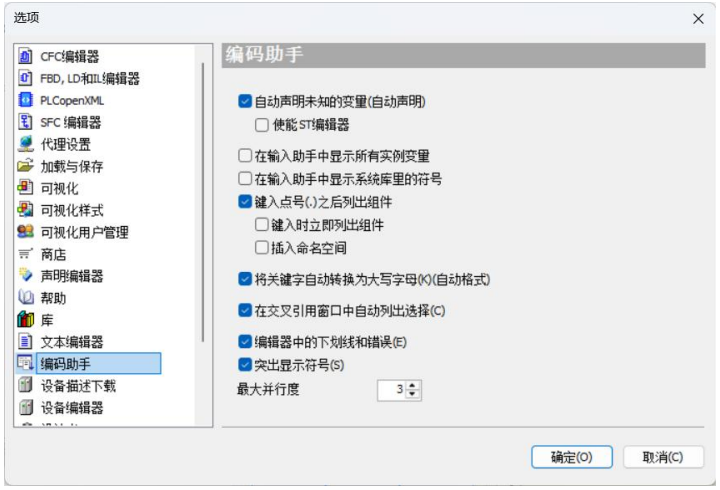
```
diVar := TRUNC(1.9);  (* Result: 1 *)
diVar := TRUNC(-1.4);  (* Result: -1 *)
```

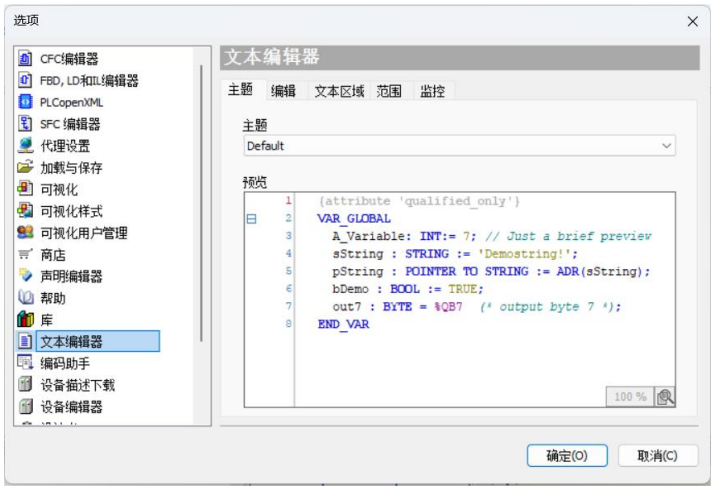
5.4 结构化文本（ST）

5.4.1 ST 编辑器

ST 编辑器是用于在结构化文本（ST）和扩展结构化文本（ExST）中实现代码的文本编辑器。

行号显示在编辑器的左侧。输入编程元素时，“List components 列表组件”功能（在 CODESYS 选项的 SmartCoding 类别中激活）和输入助手 F2 也很有帮助。当光标放在变量上时，CODESYS 将显示一个工具提示，其中包含用于声明变量的信息。





编辑器的行为（例如括号，鼠标操作，选项卡）和外观在 CODESYS 选项的“文本编辑器”类别中进行配置。

5.4.2 ST 表达式

表达式是一种构造，在其求值后返回一个值。

表达式由运算符和操作数组成。在扩展结构化文本（ExST）中，您还可以将赋值用作表达式。操作数可以是常量，变量，函数调用或其他表达式。

例:

2014	(* Constant *)
ivar	(* Variable *)
fct(a,b)	(* Function call *)
(x*y)/z	(* Expression *)
real_var2 := int.var;	(* in ExST: Assignment *) *)

通过根据某些连接规则处理运算符，可以对表达式进行优先级评估。CODESYS 首先处理具有最强连接的运算符。具有相同连接强度的运算符从左到右进行处理。

操作符	符号	连接强度
Parenthesize	(Expression)	Strongest binding
Function Call	Function name (parameter list) all operators with syntax: <operator> ()	
Exponentiate	EXPT	
Negate Complementation	- NOT	
Multiplication Division Modulo	* / MOD	
Addition	+	
Subtraction	-	
Comparison	<,>,<=,>=	
Equality Inequality	= <>	
Bool AND	AND AND_THEN	
Bool XOR	XOR	
Bool OR	OR OR_ELSE	Weakest binding

5.4.3 ST 赋值方法

赋值表达式

语法：<operand> := <expression>
该赋值运算符执行与该 MOVE 运算符相同的功能。

输出的 ST 赋值运算符

赋值运算符 => 将函数、功能块或方法的输出赋给变量。
语法：<output> => <variable>
例：

```
FBcomp_Output1 => bVar1;
FBcomp_Output2 =>;
```

FBcom_Output1 和 FB_Output2 是功能块的输出的值，FBcom_Output1 已分配给变量bVar1。

扩展 ST 赋值符“S=”、“R=”

“S=“相当于 PLC 中的“SET“，语法为：
<variable name> S= <operand name> ;
变量和操作数的数据类型为 BOOL，当操作数 Operand 从切换 FALSE 到 TRUE 时，TRUE 将分配给变量 Variable。但是，即使操作数 Operand 继续更改其状态，变量也将保持状态TRUE。
例：

```
PROGRAM PLC_PRG
VAR
    xOperand: BOOL := FALSE;
    xSetVariable: BOOL :=FALSE;
END_VAR
xSetVariable S= xOperand;
```

“R=“相当于 PLC 中的“RST”，语法为：
<variable name> R= <operand name> ;
变量和操作数的数据类型为 BOOL，当操作数 Operand 从 FALSE 切换到 TRUE 时，FALSE将分配给变量 Variable。但是，即使操作数继续更改其状态，变量也将保持状态 FALSE。

5.4.4 ST 语法

IF 语句

```
IF <condition> THEN
    <statements>
ELSIF <condition> THEN
    <statements>
… ELSE
    <statements>
END_IF;
```

ELSIF 分支和 ELSE 分支是可选的。

例:

```
PROGRAM PLC_PRG
VAR
    iTemp: INT;
    xHeatingOn: BOOL;
    xOpenWindow: BOOL;
END_VAR

IF iTemp < 17 THEN
    xHeatingOn := TRUE;
ELSIF iTemp > 25 THEN
    xOpenWindow := TRUE;
ELSE
    xHeatingOn := FALSE;
END_IF;
```

FOR 语句

FOR 循环用于执行具有一定重复次数的指令。

语法:

```
FOR
    <counter> := <start value> TO <end value> {BY <increment> } DO <instructions> END_FOR;
```

括号{}内的部分是可选的。只要<counter>的值不大于<end value>并且不小于<startvalue>，则执行<instructions>，每次<instructions>执行指令后，计数器<counter>都会自动递增<increment>。增量<increment>可以是任何整数值。如果未指定增量，则标准增量为 1。

例:

```
FOR
    iCounter := 1 TO 5 BY 1 DO iVar1 := iVar1*2;
END_FOR;
Erg := iVar1;
```

当 iVar1 初始值为 1 时，FOR 循环结束后，iVar 的值为 32。

CASE 语句

使用此对话框可将包含相同条件变量的多个条件指令合并到构造中。

语法:

```
CASE <Var1> OF
    <value1>:<instruction1>
    <value2>:<instruction2>
    <value3, value4, value5>:<instruction3>
    <value6 ... value10>:<instruction4>
    ...
    <value n>:<instruction n>
    {ELSE <ELSE-instruction>}
END_CASE;
```

大括号{}内的部分是可选的。

CASE 指令的处理方案:

如果变量的<Var1>值为<value i>，则执行指令<instruction i>。

如果变量<Var1>没有给定的值，则执行<ELSE-instruction>。

如果对变量的多个值执行相同的指令，则可以按顺序写入值，并用逗号分隔。

例:

```
CASE iVar OF
    1, 5: bVar1 :=TRUE;
    bVar3 := FALSE;
    2: bVar2 := FALSE;
    bVar3 := TRUE;
    10..20:bVar1 :=TRUE;
    bVar3= TRUE;
ELSE
    bVar1 := NOT bVar1;
    bVar2 := bVar1 ORbVar2;
END_CASE;
```

WHILE 语句

WHILE 循环类似于 FOR 循环，用于满足循环条件时循环执行指令。WHILE 循环的中止条件是布尔表达式。

语法:

```
WHILE <boolean expression> DO
    <instructions>
END_WHILE;
```

当 boolean expression 为 TRUE 时执行表达式<instructions>，为 FALSE 时则不执行。若boolean expression 一直为 TRUE，则指令将无休止地重复，结果可能会导致运行时错误。

例:

```
WHILE iCounter <> 0 DO Var1 := Var1*2
    iCounter := iCounter-1;
END_WHILE;
```

从某种意义上说，WHILE 和 REPEAT 循环比 FOR 循环更强大，因为在执行循环之前不需要知道循环的执行次数。因此，在某些情况下，只能使用这两种循环。但是，如果明确执行循环的次数，则最好使用 FOR 循环，以避免无限循环。

作为对 IEC 61131-3 标准的扩展，可以在 WHILE 循环内使用 CONTINUE 指令。

REPEAT 语句

该 REPEAT 循环的用法与 WHILE 循环类似，但是区别在于仅在执行循环后才检查中止条件。此行为的结果是 REPEAT，无论中止条件如何，循环至少执行一次。

语法：

```
REPAEAT
    <instructions>
UNTIL <boolean expression>
END_REPEAT;
```

执行<instructions>直到<boolean expression>为 TRUE 返回。

如果 <boolean expression> 在第一次求值时已经返回 TRUE，则只执行一次<instructions>指令。如果布尔表达式永不为 TRUE，则指令将无休止地重复，结果可能导致运行时错误。

例：

```
REPEAT
    Var1 := Var1*2;
    iCounter := iCounter-1;
UNTIL iCounter = 0
END_REPEAT;
```

RETURN 返回

使用该 RETURN 语句以退出功能块。

例：

```
IF xIsDone = TRUE THEN
    RETURN;
END_IF;

iCounter := iCounter + 1;
```

如果 xIsDone 的值等于 TRUE，则功能块将立即退出，并且不执行 iCounter := iCounter + 1 语句。

JMP 跳转

JMP 指令用于无条件跳转到以跳转标签标记的程序行。

语法：

```
<label>: <instructions>
JMP <label>;
```

跳转标签<label>是放置在任何程序行开头的唯一标识符。到达 JMP 指令后，将返回带代码的程序行<label>。

```
iVar1 := 0;
_label1: iVar1 := iVar1+1; (*instructions*)

IF (iVar1 < 10) THEN
   JMP _label1;
END_IF;
```

EXIT

该 EXIT指令是在使用 FOR，WHILE或 REPEAT循环，不管其中止条件，立刻结束循环。

CONTINUE

CONTINUE 是扩展结构化文本（ExST）的指令。该指令在 FOR，WHILE 和 REPEAT 循环内使用，以跳转到下一次执行循环的开始。

例：

```
FOR Counter:=1 TO 5 BY 1 DO
INT1:=INT1 / 2;
 IF INT1=0 THEN
   CONTINUE; (* to provide a division by zero *)
 END_IF
Var1:=Var1/INT1; (* executed, if INT1 is not 0 *)
END_FOR;


Erg:=Var1;
```

ST Function Block Call 函数块调用

ST 函数块调用的语法：

```
<FB-instance>(<FB input variable>:=<value or address>|, <other FB input variables>);
```

例:

定时器功能块 TON 已在 TMR:TON 其中实例化，并使用分配的参数 IN 和 PT 进行调用。

```
TMR:TON;
TMR (IN:=%OX5, PT:=T#300ms);
varA:=TMR.Q;
```

输出用 Q 寻址 TMR.Q 并分配给变量 varA。

ST COMMENTS 注释

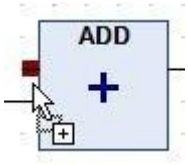
注释	描述	例
单行	//开头，直到行末尾	// This is a comment
多行	(* 开头， *)结尾	(* This is a multi-line comment *)
嵌套	(*开头， *)结尾，注释内部可能还有注释 (*.....*)	(* a:=inst.out; (* 1st comment *) b:=b+1; (* 2nd comment *) *)

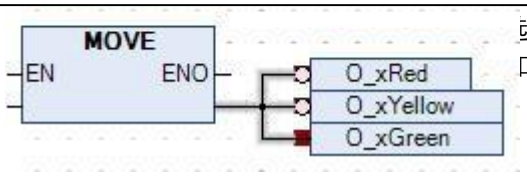
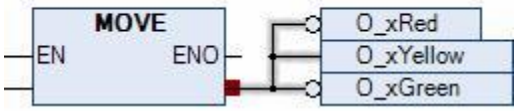
5.5 连续功能图（CFC）

连续功能图（CFC）语言是 IEC 61131-3 的标准语言延伸的图形编程语言。可以通过 CFC 中的 POU 以图形方式对系统进行编程。可以插入元素并自由放置它们，插入连接并将元素连接到网络，以便创建结构良好的功能框图。

功能框图的执行顺序基于数据流。而且，POU 可以处理多个数据流。这样数据流就没有任何通用数据。在编辑器中，多个网络之间没有任何连接。

5.5.1 CFC 编辑器

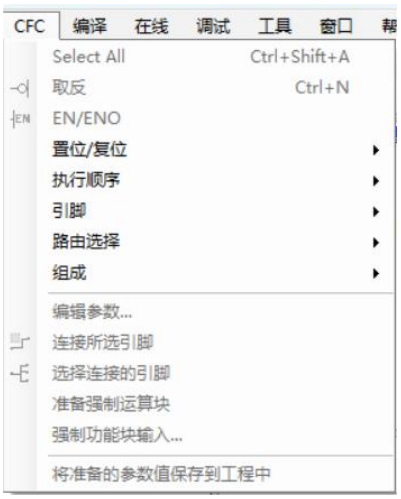
光标符号：	要求：在工具箱视图中选择了指针 。 符号表示您可以在编辑器中进行编辑。选择元素或连接以移动它们或执行命令。
光标符号：	要求：在工具箱视图中选择了任意一个元素。 在编辑器中单击将插入选定的元素。您也可以将元素拖到编辑器。
从编辑器的声明中拖动功能块实例	要求：在 CFC 的声明中选择了一行。 实例将作为具有名称，类型和所有引脚的 POU 插入。
将变量从声明拖动到编辑器中的 POU 引脚	该变量作为输入或输出插入，并与焦点所在的 POU 引脚连接。 提示：光标指示您的焦点位置何时对变量有效。 

Ctrl +单击编程区域	要求：在工具箱视图中选择了一个元素。 只要按住 Ctrl键，每次在编程区域中每次单击都会创建一个选定的元素。
Ctrl+Right Arrow	 择用于一个元素。移 果有多个引脚则全部
Ctrl+Left Arrow	 了一个输入引脚。移 有多个引脚 则全部

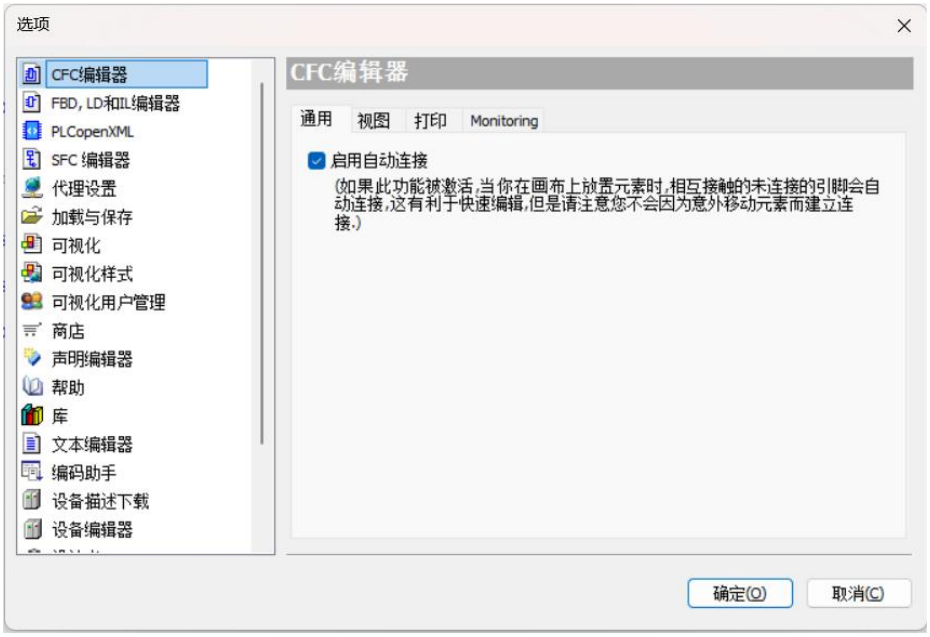
5.5.2 CFC 数据流执行顺序

在 CFC 编辑器中，可以自由放置元素，因此执行顺序最初不是唯一的。因此，软件通过数据流确定执行顺序，在有多个网络的情况下，通过元素的拓扑位置确定执行顺序：元素 从上到下，从左到右排序。

添加 CFC 语言对象 POU 后，在该 POU 界面，菜单栏会出现 CFC 选项，如下

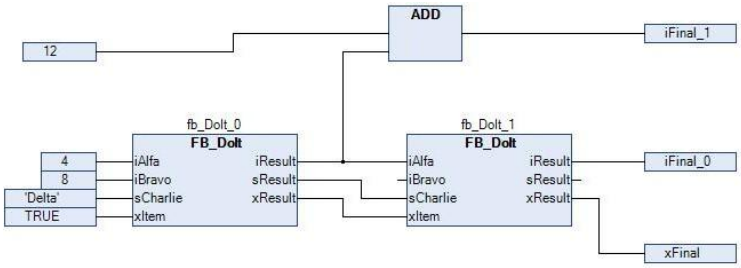


打开“工具——选项——CFC编辑器”，打开 CFC 编辑器设置，可对编辑器内容自定义。

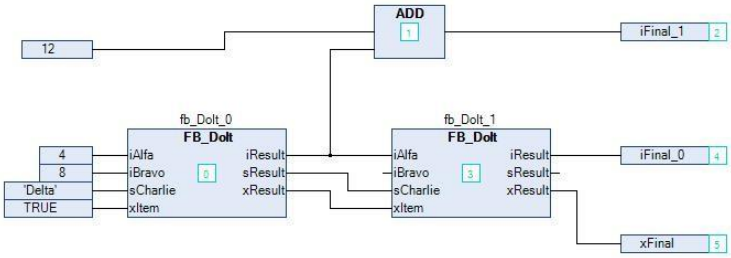


默认情况下，将自动确定 CFC 对象的执行顺序。为此勾选了“Display Execution Order”属性。可以在 CFC 编辑器中勾选临时显示自动确定的执行顺序。

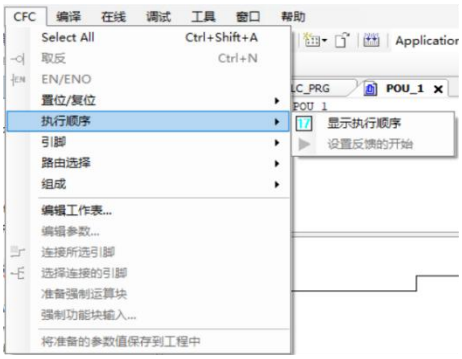
例：添加程序如下



点击“CFC ‣ 执行顺序 ‣ 展示执行顺序”，显示对象的执行顺序。框和输入已相应编号，并反映了时间顺序。再次在 CFC 编辑器中单击时，该编号将被隐藏。



右键功能块，或点击“CFC ‣ 执行顺序”，可以改变执行顺序。



5.5.3 CFC 元素

Page 页面

符号：

元素将新页面插入编辑器。仅在面向页面的 CFC 编辑器中可用。页面编号根据其位置自动分配。可以在橙色标题中输入页面的名称和描述。使用“ 编辑页面大小”命令可以调整页面大小。

Control Point 控制点

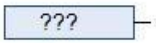
符号：

在调整线路路由之 前，可使用 Control Point 控制点来固定连接点。这样，将元素拖动到连接线上的所需位置，具有控制点的连接线不再自动布设。

Input 输入

符号：

默认情况下会插入带有文本“ ???”的输入元素。可以通过单击该字段并输入常量值或变量名来直接编辑该字段。或者，可以打开输入助手以通过单击...来选择变量。



Output 输出

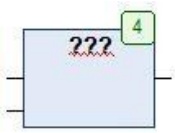
符号：

默认情况下会插入带有文本“ ???”的输出元素。可以通过单击该字段并输入常量值或变量名来直接编辑该字段。或者，可以打开输入助手以通过单击...来选择变量。

Box 运算块

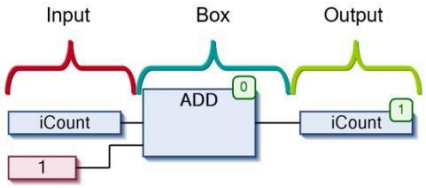
符号：

可以使用该元素来插入运算符，函数，功能块或程序。默认情况下，插入名称为“???”的元素。可以通过单击该字段并输入功能块名称来直接编辑该字段。或者，可以打开输入助手并通过



单击选择功能块 。

对于功能块，CODESYS 在功能块符号上方还显示一个输入字段“???”。必须用功能块实例的名称替换该名称。替换现有的框，只需将当前插入的标识符替换为所需的新名称。进行此操作时，请注意，软件根据 POU 的定义调整输入和输出引脚的数量，因此可能会删除现有的分配。CFC 的 Input、Output 以及 Box 结构如下：



Jump 跳转

符号：

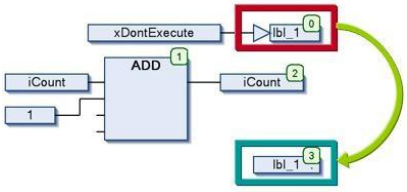
可以使用元素来定义程序继续执行的位置。必须通过标签定义此目标位置。要做到这一点，请在输入栏标记的名字“???”。如果已经插入了相应的标签，也可以通过输入助手（ ） 选择它。

Label 标签

符号：

标签定义一个位置，程序执行借助 jump 元素跳转到该位置。

例：Jump 和 Label 用法



Return 返回

符号：

使用元素可以退出功能块 POU。

Composer 合成器

符号：

合成器元素用于处理结构体类型的 Box 的输入。 合成器将显示结构组件，从而使程序员可以在 CFC 中对其进行访问，可以将结构的各个组件用作输入。为此，必须像相关结构一样命名合成器元素（替换???）。

合成器元素与选择器元素相对应。

Selector 选择器

符号：

选择器用于处理结构体类型的 Box 的输出。 选择器将显示结构组件，从而使程序员可以在 CFC 中对其进行访问，可以将结构的各个组件用作输出。为此，必须像相关结构一样命名选择器元素（替换???）。

选择器元素与合成器元素相对应。

Comment 注释

符号：

使用此元素，您可以在 CFC 编辑器中输入注释。用注释文本替换元素中的占位符文本。可以使用快捷键 Ctrl+ Enter插入换行符。

Connection Mark - Source/Sink 连接标记-发送/接收

符号： ,

可以使用连接标记代替元素之间的连接线。这可以帮助您更清晰地显示复杂的图。

对于有效的连接，必须将元素连接标记-源与元素的输出连接，并将元素连接标记-接收 器与另一个元素的输入连接。两个标记必须具有相同的名称，名称不区分大小写。

Input Pin 输入引脚

符号: 

根据功能块的类型，可以将更多输入添加到插入的功能块元素中。为此，必须选择功能 块元素并将功能块输入元素拖到功能块的主体上。

Output Pin 输出引脚

符号: 

根据功能块的类型，可以将更多输出添加到插入的功能块元素中。为此，必须选择功能 块元素并将功能块输出元素拖到功能块的主体上。

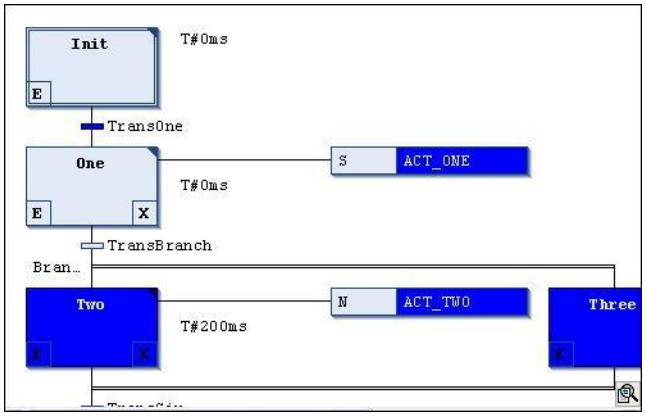
5.6 顺序功能图（SFC）

5.6.1 SFC 编辑器

SFC 编辑器是图形编辑器。新的 SFC POU 包含一个 Init 步骤和随后的过渡。

在 SFC 编辑器中，可以通过 SFC 菜单（打开 SFC 语言的 POU 后自动出现）上下文菜单的命令将单个元素插入到图中。

也可以将 SFC 元素从“工具箱”视图拖到图中。在编辑器上拖动元素时，软件用灰色框标记所有可能的插入点。如果将鼠标移到灰色框上，则框的颜色将变为绿色。释放鼠标按钮时，该对象将插入到该位置。



在在线模式下，CODESYS 以蓝色显示活动步骤。

5.6.2 SFC 的处理顺序

1 、 重置IEC动作CODESYS 重置动作限定符（N，R，S，L，D，P，SD，DS，SL）的内部动作控制标志，这些标志控制 IEC 动作。

2、执行退出动作

软件验证所有步骤是否满足每个步骤执行退出动作的条件。验证顺序遵循 SFC 图中的布局，从上到下，从左到右。禁用步骤后，软件将执行退出动作（在前一个循环中执行了任何输入和步骤动作，且随后步骤的条件产生 TRUE 后）。

3、执行输入动作

软件验证所有步骤是否满足每个步骤执行输入动作的条件。验证顺序遵循 SFC 图中的布局，从上到下，从左到右。如果满足条件，则软件执行输入操作。一旦上一步的转换已完成并产生 TRUE，软件将立即执行输入动作，从而表明该步骤已被激活。

4、时间检查/执行步骤动作

软件按照 SFC 布局的顺序对每个步骤执行以下检查：

- 软件将活动步的经过时间复制到相应的隐式步变量。`<step name>.t`
- 如果发生超时，则 CODESYS 设置各自的错误标志。
- 对于非 IEC 步骤：CODESYS 执行步骤动作。

5 、 执 行IEC行 动

CODESYS 按字母顺序执行 IEC 动作，两次通过动作列表。在第一遍中，软件对在上一个循环中禁用的每个步骤执行 IEC 动作。在第二遍中，针对每个活动步骤执行 IEC 动作。

6、过渡检查/激活下一步

转换按如下方式通过：如果某个步骤在当前循环中处于活动状态，并且随后的转换产生，TRUE 并且该步骤定义的最短时间已过，则激活后续步骤。

5.6.3 SFC 动作条件

可以将限定词分配给 IEC 步骤。限定词描述了如何处理步进动作。

限定符由“SFCActionControl”库中的功能块处理 “IecSfc.library”。该库通过 SFC 插件自动集成到项目中。

N	非储存	只要该步骤处于活动状态，该操作就处于活动状态。
R0	覆盖复位	该动作被禁用。
S0	设置（存储）	该步骤激活后，软件将立即执行此操作。即使已取消激活该步骤，该 操作将一直执行到收到复位为止。
L	限时	该步骤激活后，软件将立即执行此操作。该操作将一直执行到步骤被 禁用或给定的时间间隔过去为止。
D	时间延迟	仅在步骤激活后经过给定的延迟时间并且该步骤仍处于活动状态后， 软件才开始执行操作。执行该操作，直到禁用该步骤。
P	脉冲	软件精确地执行该动作两次：一次激活该步骤，一次激活该步骤。
SD	存储和时间延迟	仅在步骤激活后经过给定的延迟时间后，软件才开始执行操作。该动 作将执行到收到复位为止。
DS	延迟和存储	仅在步骤激活后经过给定的延迟时间并且该步骤仍处于活动状态后， 软件才开始执行操作。该动作将执行到收到复位为止。
SL	储存时间有限	该步骤激活后，软件将立即执行此操作。该操作将一直执行到给定时 间过去或收到重置为止。

5.6.4 SFC 隐式变量和标志

SFC 隐式变量

每个 SFC 对象都提供隐式变量，监视在运行时步骤和 IEC 动作的状态。CODESYS 针对每个步骤和每个 IEC 动作自动声明这些隐式变量。

隐式变量是步骤类型 SFCStepType 和动作类型 SFCActionType 的结构实例。变量与其元素具有相同的名称，例如“ step1”步骤名称为“ step1”变量名称。结构成员描述步骤或动作的状态或活动步骤中当前经过的时间。

隐式变量声明的语法：

```
<step name>:SFCStepType;
_<action name>:SFCActionType;
```

以下隐式变量可用于步骤或 IEC 动作状态：

Step 步	
<step name>.x	显示当前周期的激活状态。 当<step name>.x = TRUE ，表明软件正在处理当前周期的步骤。
<step name>._x	显示下一个周期的激活状态。 当<step name>._x = TRUE 且<step name>.x = FALSE，表明软件正在下一个循环中处理该步骤
<step name>.t	自激活该步骤以来的当前经过时间。这仅适用于步骤，而不管是否在步骤属性中定义了最短时间。
<step name>._t	仅限内部使用
IEC 行动	
_<action name>.x	TRUE ： 动作正在被执行动作。
_<action name>._x	TRUE ： 动作处于激活状态。

可以使用上述变量将特定的状态值强制到步骤（激活步骤）。

访问隐式变量语法：

a、直接在 POU 中分配隐式变量：

```
<variable name>:=<step name>.<implicit variable>或
<variable name>:=_<action name>.<implicit variable>
```

例：

```
status:=step1._x;
```

b、从另一个 POU，其名称为：

```
<variable name>:=<POU name>.<step name>.<implicit variable>或
```

```
<variable name>:=<POU name>._<action name>.<implicit variable>
```

例:

```
status:=SFC_prog.step1._x;
```

SFC 标志

SFC 标志是具有预定义名称的隐式生成的变量，用于控制 SFC 图的处理。例如，可以使用这些标志来显示超时或重置步骤链。此外，可以专门激活提示模式以激活过渡。必须声明 并激活这些变量才能访问它们。

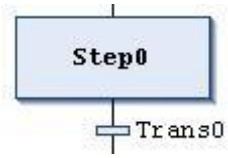
名称	数据类型	描述
SFCInit	BOOL	TRUE：软件将序列重置为初始步骤。其他 SFC 标志也会被复位（初始化）。 当变量为 TRUE时，初始步骤保持设置（活动）状态，但不执行其操作。当 SFCInit重新给 FALSE时，其块处理才会继续往下
SFCReset	BOOL	此功能类似于 SFCInit。但是，软件在初始化初始步骤后继续处理。
SFCError	BOOL	TRUE：如果 SFC 图中发生超时。 如果程序中发生第二次超时， 则除非您先前重置了变量 SFCError，否则它不会被置 FALSE。其他用于控制时间顺序标志变量函数（SFCErrorStep，SFCErrorPOU，SFCQuitError）的 使用，需要先声明 SFCError。
SFCEnableLimit	BOOL	用于激活（TRUE）和取消激活（FALSE）超时控制 SFCError。必须将其设置 TRUE为 SFCError才能起作用。如果不这样做，则忽略超时。
SFCErrorStep	String	存储导致超时的步骤的名称。前提是声明了 SFCError。
SFCErrorPOU	String	存储发生超时的块的名称。 前提是声明了 SFCError。
SFCQuitError	BOOL	TRUE：软件会暂停 SFC 图形的处理，并且该变量中的任何超时 SFCError都将被重置。如果将变量重置为 FALSE，则将重置活动步骤中所有先前的时间。前提是声明了 SFCError。
SFCPause	BOOL	TRUE：软件暂停 SFC 的处理。
SFCTrans	BOOL	如果过渡 Transidition 处于活动状态，则为 TRUE。
SFCCurrentStep	String	显示活动步骤的名称，与时间监视无关。 在并行分支中，始终存储最右边分支线的步骤名称。

5.6.5 SFC 元素

Step and Transition 步和过渡

步符号 ; 过渡符号

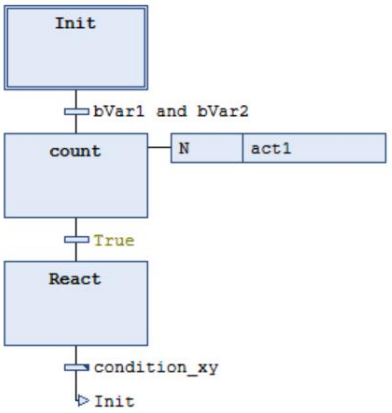
通常，软件将步和过渡作为组合插入。插入没有过渡的步或没有步的过渡，会导致编译 时出错。可以通过双击名称进行修改，并且步名称必须唯一，不能重复。

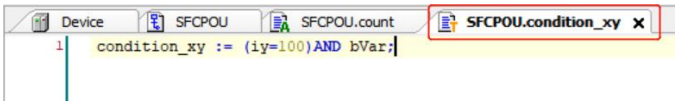


所有步骤均由步骤属性定义，可以根据设置的选项在“ 属性”视图中显示和编辑它们。必须将那些活动添加到该步骤中，以便在该步骤处于活动状态时执行。

过渡条件必须产生 TRUE 或 FALSE。可以通过以下两种方式之一进行定义：

- ①内联条件（直接）可以将默认转换名称替换为布尔变量的名称，布尔地址，布尔常数或带有布尔结果的语句，例如：(i<100) AND b 。但不能在此处指定程序，功能块或分配。
- ②多次使用条件（引用过渡或属性过渡）：将默认的过渡名称替换为引用或属性对象的名称（ , ），通过单击“项目 Project 添加对象”创建这些对象。这样可以多次使用过渡，例如下图中的“ condition_xy”。像内联条件一样，对象可以包含布尔变量，布尔地址，布尔常数或带有布尔结果的语句。此外，它还可以包含带有任何代码的多个语句。





引用过渡或属性对象的过渡在过渡框的右上角标记有一个小三角形。

CODESYS 将转换条件视为方法调用。该条目具有以下语法：

```
<transition name>:=<transition condition>
```

例：

```
trans1:= a=100
```

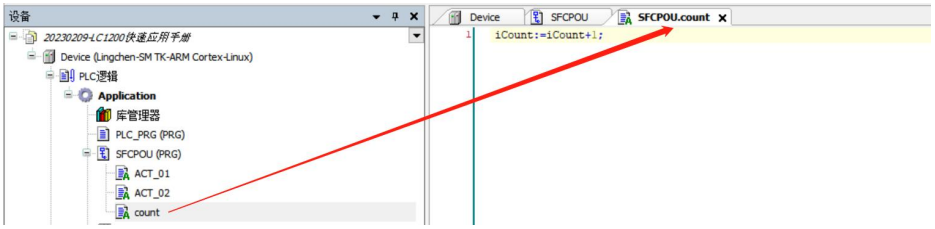
或仅写

```
<transition condition>                    //（例如a=100）
```

Action 动作

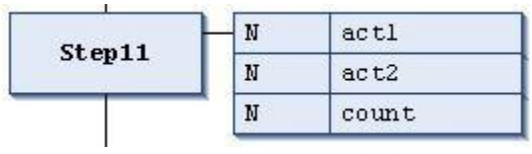
符号： 

动作包括使用一种有效实现语言的一个或多个语句。可以将操作分配给步骤，在 SFC 步骤中使用的动作必须在项目中创建为 POU。



1、IEC 动作

IEC动作符合IEC1131-3标准。它们根据其限定符执行。IEC 动作执行两次：第一次在激活该步骤时，第二次在禁用该步骤时。如果将多个动作分配给一个步骤，则动作列表从上到下进行处理。每个动作框的第一栏中包括限定符（N 等），第二栏中包括动作名称（act1 等），两者均可直接进行编辑。



与步动作相比，可以对 IEC 动作使用不同的限定符。此外，每个 IEC 动作都带有一个控制标志，这将指示 CODESYS 在任何时候仅一次执行一个动作，即使该动作同时被另一个步骤调用也是如此。对于这个，步动作不能保证。

通过单击 SFC‣ 插入动作关联，将 IEC 动作分配给步骤。

2、步动作

可以使用这些步骤操作来扩展 IEC 标准。

- 进入动作：

在激活步之后且在执行主操作之前，软件执行此操作。

可以通过输入动作元素属性②，从一个步骤中引用一个新动作或在 SFC 对象下右键“Add Entry Action”创建进入动作。也可以通过“添加条目操作命令向该步骤添加新步。输入操作在步骤框的左下角标记为 E。

- 主要动作：

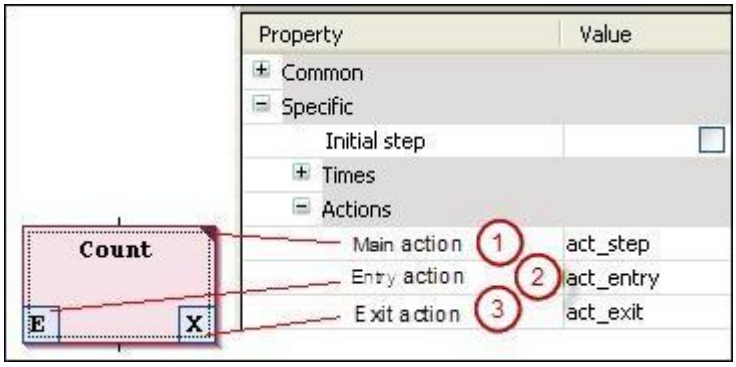
当步处于活动状态并且任何输入动作均已处理时，软件会执行此动作。但是，与 IEC 动作相反（请参见上文），这些步骤动作在禁用该步骤后不会再次执行。

可以通过 Main action 元素属性④，将现有操作添加到步骤，也可以通过单击 step 元素来创建和添加新操作。一个主要动作在步骤框的右上角用实心三角形标记。

- 退出动作：

跳出该步后，软件将一次执行此操作。但是注意，退出动作不是在同一周期内执行，而是在下一个周期开始时执行。

可以通过退出动作元素属性③，从一个步骤中引用一个新动作或在 SFC 对象下创建的动作。也可以通过“ 添加退出操作”命令将新操作添加到步骤中。退出动作在步骤框的右下角标记为 X。



3、IEC 动作与步进动作之间的区别：

步骤动作和带有限定符 N 的 IEC 动作之间的主要区别在于，IEC 动作始终执行两次：激活步骤和取消激活步骤。对于 IEC 动作，可以指定布尔变量而不是动作对象。这对于步进操作是不可能的。

Branch 分支

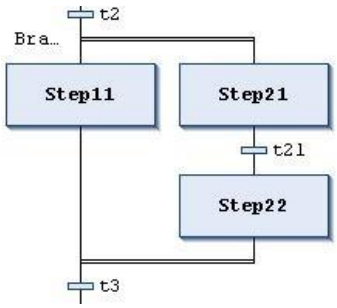
符号 

使用分支对顺序功能图中的并行或替代顺序进行编程。

对于替代分支，软件一次仅处理分支线之一，具体取决于前面的转换条件；对于并行分　支，则同时处理。

1、平行分支

对于并行分支，分支线必须以步骤开头和结尾。平行分支线可以包含其他分支。分支之　前和之后的水平线是双线。



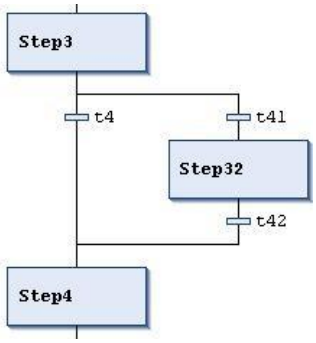
在线模式下的处理：如果前面的转换（本例中为 t2）产生 TRUE，则所有并行分支线中的第一步均处于活动状态（Step11 和 Step21）。软件同时处理各个分支线，然后传递后续转换（t3）。

“Branch <n>”跳转标记会自动添加到指示分支开始的水平线，可以将此标记定义为跳转目的地。

2、替代分支

分支之前和之后的水平线是一条直线。

在替代分支中，分支线必须以过渡开始和结束，分支线可以包含其他分支。



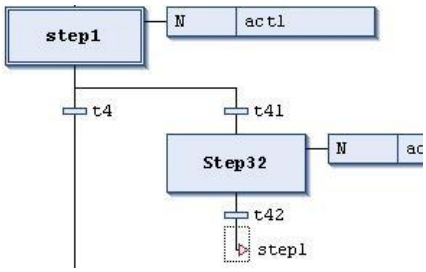
如果分支之前的步骤处于活动状态，则软件从左到右传递每个替代分支线的第一个过渡。　对于产生的第一个过渡，TRUE关联的分支线打开，从而激活过渡后的步骤。

Jump 跳转

符号

使用跳转来定义一旦跳转之前的转换为 TRUE，则下一步应在步骤中执行哪些动作。由于执行路径不能交叉或向上引导，因此可能需要跳转。除了在图的末尾所需的跳转外，通常 只能在分支的末尾插入跳转。

跳转的目的地由添加的文本字符串定义，可以直接对其进行编辑。跳转目标可以是步骤 名称或并行分支的标记。



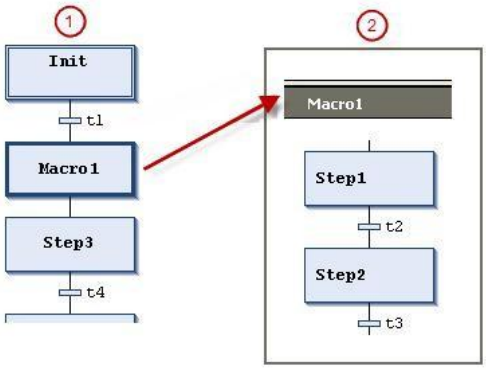
Macro 宏

符号

宏包含 SFC 图的一部分，但在编辑器的主视图中未详细显示。

使用宏不会影响处理流程,宏用于隐藏图的特定部分，例如，以提高整体清晰度。

可以通过双击宏框或单击“SFC Zoom into Macro”来打开宏编辑器。可以像在 SFC 编辑器的主视图中一样在此处进行编程。要关闭宏编辑器，请单击“SFC Zoom out of Marco”。



①SFC 编辑器中的主视图

②Macro1 的宏编辑器视图

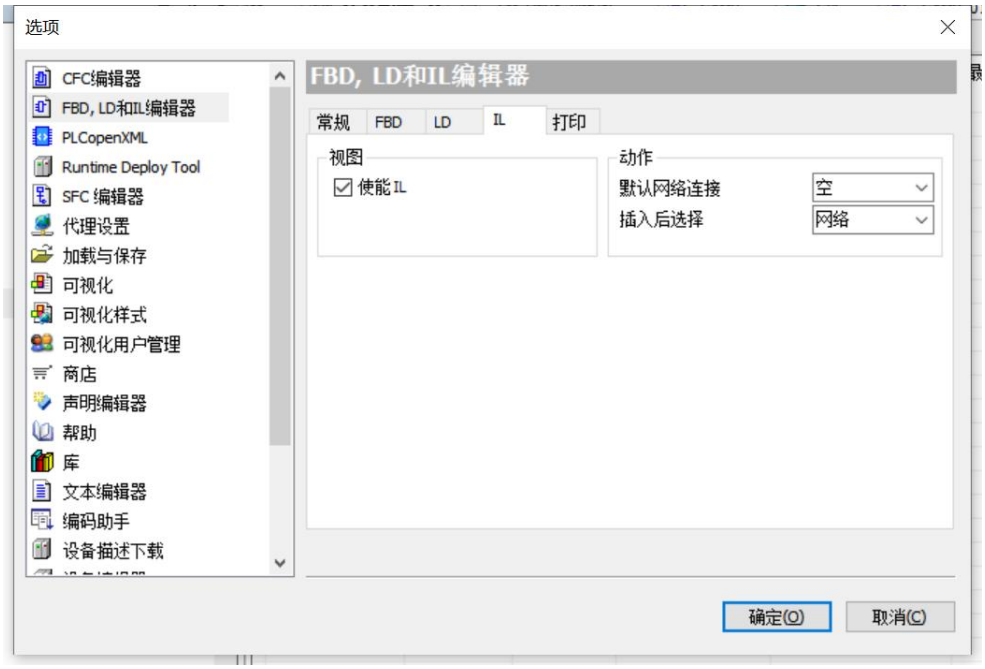
宏还可以包括其他宏。宏编辑器的标题始终显示图中打开的宏的路径，例如:

5.7 功能框图/梯形图/指令表（CFC/LD/IL）

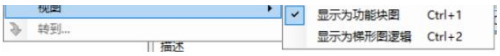
5.7.1 FBD / LD / IL 编辑器

FBD / LD / IL 编辑器是 FBD，LD 和 IL 编程语言的组合编辑器。

如有必要，可以在软件选项“工具-选项-FBD，LD和IL编辑器”中的IL界面，选择“使能IL”，激活 。



在内部自动将 3 种编程语言相互转换。在网络的帮助下，实现部分中的代码以所有三种语言构造。在 FBD / LD / IL 菜单提供了在编辑器中工作的命令。在离线和在线模式下，可以随时通过使用 View 中的菜单命令来切换编辑器。



5.7.2 FBD/LD/IL 元素

Network 网络

符号

网络是 FBD 或 LD 程序的基本单元。在 FBD / LD / IL 编辑器中，网络以列表形式排列。每个网络在左侧都有一个顺序的网络号。可以包括: 逻辑和算术表达式, 程序/功能/功能块 调用，跳转或返回语句。

一个 IL 程序包含至少一个网络，该网络可以包括程序的所有 IL 语句。可以为每个网络提供标题，评论或标签。在软件选项“Tools Options FBD,LD,IL Editor”中，可以定义是否在编辑器中显示各个网络之间的网络标题，注释和分隔符。单击网络的第一行以输入网络标题。单击网络的第二行以输入网络注释。

Box 运算块

符号:

一个运算块及其调用可以代表附带功能，例如 IEC 功能块，IEC 功能，库功能块等。一个运算块可以有任意数量的输入和输出。

如果该运算块中还提供了图像文件，则该运算块内将显示块图标。要求是在软件选项“Tools Options FBD,LD,IL Editor”中激活“ Show box iCon 显示框符号 ”选项。如果更改了Box 接口，则可以使用命令”FBD / LD / IL Update Parameters”更新 Box 参数，而不必重新插入 Box。

Box with EN/ENO 带 EN/ENO 的运算块

符号： 

该元素仅在 FBD 和 LD 编辑器中可用。该运算块通常对应于 FBD / LD / IL 元素“Box”；但是，此运算块还包含一个 EN 输入和一个 ENO 输出。EN 和 ENO 数据类型为 BOOL。

所述的功能的 EN输入和 ENO输出：如果在运算块执行时，输入 EN具有值 FALSE，则在运算块中所定义的操作没有被执行。如果 EN值为 TRUE，则执行运算块定义的操作。ENO 输出具有与 EN 输入相同的值。

Assignment 分配

符号：

FBD 编辑器将新插入的作业显示为一行，其后带有 3 个问号。LD 编辑器将新插入的分配显示为线圈，上方有 3 个问号。插入后，可以将占位符???替换为变量的名称，来自左侧的信号将被分配给该变量。

Input 输入

符号： 

输入的最大数值取决于运算块定义的接口的类型。

新添加的输入首先用???标记。您可以将字符串替换???为变量或常量。

Label 标签

符号：

标签是 FBD 和 LD 中网络的可选标识符，可以将其指定为跳转的目的地。

如果在网络中插入跳转标签，则会将其作为可编辑字段“ Label：”添加到网络中。

Jump 跳转

符号 

在 FBD 或 LD 中，根据当前光标位置，直接在输入之前，输出之后或网络末端插入跳转。在跳转元素后面直接输入一个跳转标签作为跳转目的地。

在 IL 中，可以根据指令编写跳转 JMP。

Return 返回

符号：

如果 RETURN 元素执行 TRUE，则该元素立即中断运算块的执行。在 FBD 或 LD 网络中，可以将 Return 指令与前面的元素平行或放置在后面。

Branch 分支

符号： 

该元素在 LD 和 FBD 编辑器中可用，代表一个开放式分支。一个行分支将处理行从当前光标位置向前分成两个子网，从上到下依次执行。可以进一步分支每个子网，结果是在网络 中创建了多个分支。

每个子网在分支点处都有一个标记符号（矩形）您可以选择该标记符号以执行更多命令。复制，剪切和粘贴命令不适用于子网。

为了删除子网，您必须首先删除网络的所有元素，然后删除子网的标记符号。

Excute 执行

符号： 

该元素是一个框，使可以直接在 FBD 和LD 编辑器中输入ST 代码。可以用鼠标将 Execute 元素从“ 工具”视图拖动到 POU 的实现部分。如果单击此处输入 ST 代码...，则会打开一个输入字段，可以在其中输入多行 ST 代码。

Contact 触点

符号：  ，在编辑器中 

该元素仅在 LD 编辑器中可用。

触点从左到右传递信号 TRUE（ON）或 FALSE（OFF）直到信号最终到达网络右侧的线圈。为此，将包含信号的布尔变量分配给触点。为此，将触点上方的???占位符替换为 BOOL 变量的名称。

可以串联和并联布置多个触点。对于两个并联触点，只有一个需要获取该值 TRUE，以便将 ON 传递到右侧。如果触点是串联的，则所有触点都必须获得值 TRUE，以便 ON 可以通过串联中的最后一个触点传递到右侧。

如果变量值为 FALSE，则负接点会  转发信号。将鼠标指针放在触点上，同时按下鼠标左键并选择了网络，则网络中将显示“ 转换为线圈 ”按钮。如果现在将鼠标指针移到该按钮上，同时仍然按下鼠标按钮，然后在该按钮上释放鼠标按钮，则 CODESYS 会将触点转换为线圈。

Coil 线圈

符号：  ，在编辑器中 

该元素仅在 LD 编辑器中可用。

线圈采用从左侧提供的值，并将其保存在分配给该线圈的布尔变量中。其输入的值可以为 TRUE（ON）或 FALSE（OFF）。网络中的多个线圈只能并联排列。在取反的线圈  中，输入信号的取反值存储在分配给线圈的布尔变量中。

设定线圈，复位线圈

符号：  ，  在编辑器中：  ，

设定线圈：如果值 TRUE到达设定线圈，则线圈将保留该值 TRUE。只要应用程序正在运行， 该值就不再可以在此处覆盖。

重置线圈：如果值 TRUE到达重置线圈，则线圈将保留该值 FALSE。只要应用程序正在运行， 该值就不再可以在此处覆盖。

Branch Start/End 分支开始/结束

符号: 

元素用于封闭的分支。

第 6 章 常用运动控制指令详解

6.1 单轴 MC 指令的运动控制编程

6.1.1 MC 指令编程要点

LC1200与伺服轴配合的运动控制是基于EtherCAT总线网络来实现的，与以往的硬件输出的脉冲方式不同，完全是采用软件方式实现，具体来说是在每个很短的EtherCAT总线周期中进行一次计算、发布一次控制命令，来实现对伺服的控制。因此，需要注意如下要点：

用户MC控制程序，是以EtherCAT任务周期执行的，MC相关的POU应配置在EtherCAT任务下执行，多数MC功能块若放在低优先级的Main任务的POU中，无法正常运行；

MC功能块的执行，需要通信中的通信数据对象来传递，因此PDO配置表中要有必需的配置项；

若遗漏了配置相关的数据对象，伺服可能无法正常运行，且不会有出错报警；

控制器可通过SDO的配置对伺服的功能码进行初始化设定操作，使得伺服的运行模式（一般为CSP模式）、伺服电机编码器模式、电子齿轮比等，以确保控制命令与物理运行位置的对应；

对伺服的初始化，还可提高设备的调试效率、部件更换后不会出错；

对于伺服轴的控制，要遵循轴状态转移的规则和逻辑，根据轴当前的状态和希望的运动，使用合适的MC功能块进行控制；

用户程序中使用的是MC功能块的实例，一个MC实例，只能用于一个伺服轴的控制，若同时用于几个伺服轴的控制，会导致控制混乱；

一个正在运转的伺服轴，必需有一个MC功能块对其运行进行监控，哪怕是MC_Stop也是一种监控，避免因程序逻辑的跳转导致无MC功能块监控的状态，系统会停机报错，这样的错误不容易检查；

注意调试的安全处理。若伺服系统采用增量式编码器，正常运行之前需要有归零操作，伺服驱动器的DI信号输入端口，可接入原点位置信号，对于在有限范围内运动（如丝杆），调试前应有极限与安全保护信号。

6.1.2 单轴控制常用的MC 功能块

MC功能块（Function Block，简称FB）也称为MC指令，准确地讲，用户程序中使用的是MC功能块的对象实例，伺服轴通过MC对象实例来进行控制。单轴的控制，一般用于定位的控制，即伺服电机拖动外部机构运动到指定的位置；有时也需要伺服以指定的速度或力矩运行等，在单轴控制中，常用到如下的MC功能块：

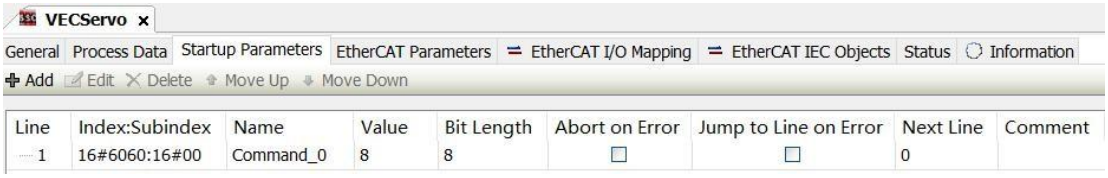
控制操作	需要使用的MC指令	说明
伺服使能	MC_Power	运行该指令，使伺服轴使能，才能进行后续的运行控制
伺服点动运行	MC_Jog	伺服电机的点动运行，常用于低速试车，用于检验设备或调整伺服电机位置
相对定位	MC_MoveRelative	以当前位置为参考，运行指定距离
相对叠加定位	MC_MoveAdditive	在伺服当前运行指令的基础上，再相对运行指定距离
绝对定位	MC_MoveAbsolute	命令伺服运行到指定的坐标点
速度控制	MC_MoveVelocity	命令伺服以指定的速度运行
转矩控制	MC_MoveTorque	命令伺服以指定的力矩运行
伺服暂停	MC_Halt	命令伺服暂停运行，若MC_Movexxx再次触发，伺服可以再运行
紧急停机	MC_Stop	命令伺服紧急停机，只有stop命令复位后，触发MC_Movexxx，伺服才可以再运行
报警复位	MC_Reset	当伺服出现告警停机后，运行该指令进行复位
改变运行模式	MC_ControlMode	使用该指令可以让伺服选择“位置”、“速度”或“力矩”模式
伺服原点回归	MC_Home	命令伺服开始原点回归操作，应用系统的原点信号、两侧极限信号等都接在伺服的DI端口
控制器原点回归	MC_Homing	控制系统开始原点回归操作，应用系统的原点信号、两侧极限信号等都接在控制器的DI端口

6.1.3 MC 指令与PDO/SDO 配置

LC1200控制器在执行用户程序的伺服轴MC控制命令时，需要将执行MC指令时与伺服交互所需的信息项目，添加在通信PDO/SDO配置表中，以便完成所需的控制功能。

MC指令	所需的TPDO对象	所需的RPDO对象
MC_Power MC_Halt MC_Stop MC_ResetMC_Home MC_Homing	ControlWord（控制字）	StatusWord（状态字） Errorcode（错误码）
MC_Jog	TargetPosition（目标位置）	Position actual value （当前轴位置） Following error actual value （当前跟踪误差）
MC_MoveRelative		
MC_MoveAdditive		
MC_MoveAbsolute		
MC_MoveVelocity	Target velocity（目标速度） Max profile velocity （最大轮廓速度）	
SMC_SetTorque	Targettorque（目标转矩）	Torque actual value （当前转矩）
SMC_SetControllerMode	Modesofoperation（运行模式）	1:周期同步转矩模式CST 2:周期同步速度模式CSV 3:周期同步位置CSP

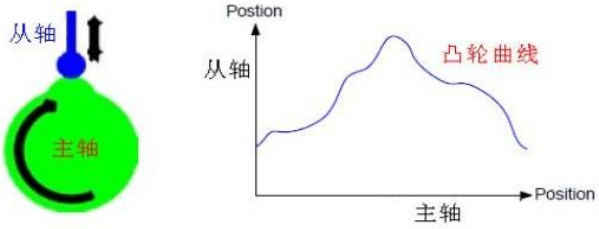
上述TPDO、RPDO是进行单轴控制所需的基本配置项。在MC控制中，伺服多数情况下为位置模式，尤其是在基于EtherCAT总线的应用系统中，为“周期同步位置模式”，因此编程时在SDO配置中，一般将伺服初始化设置为该运行模式。【此图需替换，替换用步进？】



6.2 多轴 CAM 凸轮同步的运动控制编程

凸轮运动是借用了机械凸轮与挺杆相对运动特性概念，指控制器按特定的相对位置非线性关系，令伺服从轴跟随主轴，进行的连续同步运动，以满足设备所需的运动特性的一种运动，在定长切割、追剪控制、飞剪控制、多色套印等有同步要求的应用中，被大量使用。

电子凸轮曲线的主从轴位置关系如下图，水平轴为主轴位置，而纵轴为从轴位置：



LC1200控制器采用的是软件方式，实现的凸轮运动控制特性，也就是利用软件数字方式的“凸轮表”来代替机械凸轮，因此也称电子凸轮控制。相比于机械凸轮，有如下特点：

- 凸轮形状制作容易：采用凸轮表、凸轮曲线或数组描述凸轮；
- 凸轮形状容易多样：支持多个凸轮表选择、运行中可动态切换；
- 凸轮形状修改容易：允许凸轮表关键点运行中动态修改；
- 多个凸轮从轴：允许有多个凸轮从轴；
- 凸轮挺杆：允许有多个凸轮挺杆、多个设置区间；
- 凸轮离合器：凸轮运行中，用户程序可使之进入与退出凸轮运行；

电子凸轮特有功能：支持虚拟主轴，相位偏移，输出叠加。

LC1200控制器的凸轮运算采用纯软件方式执行，若在CAM运行状态，每次进入EtherCAT任务，都会计算一次从轴的下一目标点，因此相比硬件凸轮运算，具有更好的功能灵活性。

电子凸轮的控制有三个要素：

① 主轴：被用于同步控制的参考轴；

② 从轴：根据主轴位置，按照所需非线性特性进行跟随运动的伺服轴；

③ 凸轮表：描述主轴—从轴相对位置与范围、周期性等的数据表或凸轮曲线。

用户编写程序需要设计凸轮表，指定主轴与从轴，运行中在合适时刻触发凸轮运行，就可以使得从轴进入凸轮运行了。

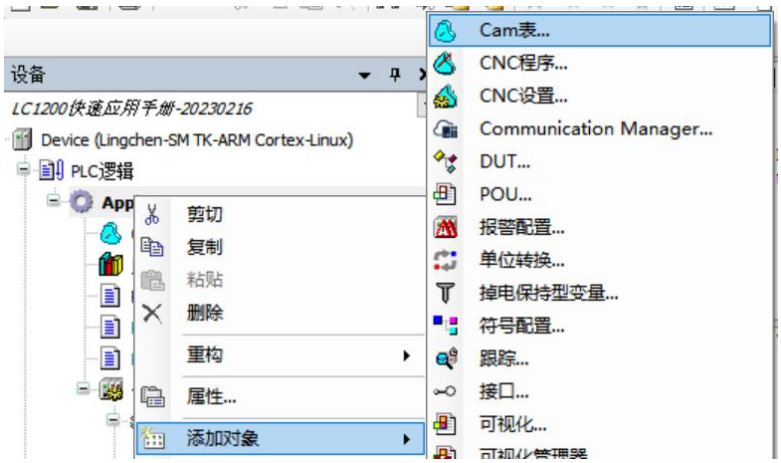
电子凸轮控制的基本指令功能块：

控制操作	需要使用的 MC 指令	说明
凸轮表选择	MC_CamTableSelect	运行该指令，关联主轴、从轴凸轮表三者关系
进入凸轮运行	MC_CamIn	让从轴进入凸轮运行
退出凸轮运行	MC_CamOut	让从轴退出凸轮运行
修正凸轮相位	MC_Phasing	主轴相位修改

6.2.1 凸轮表的特点

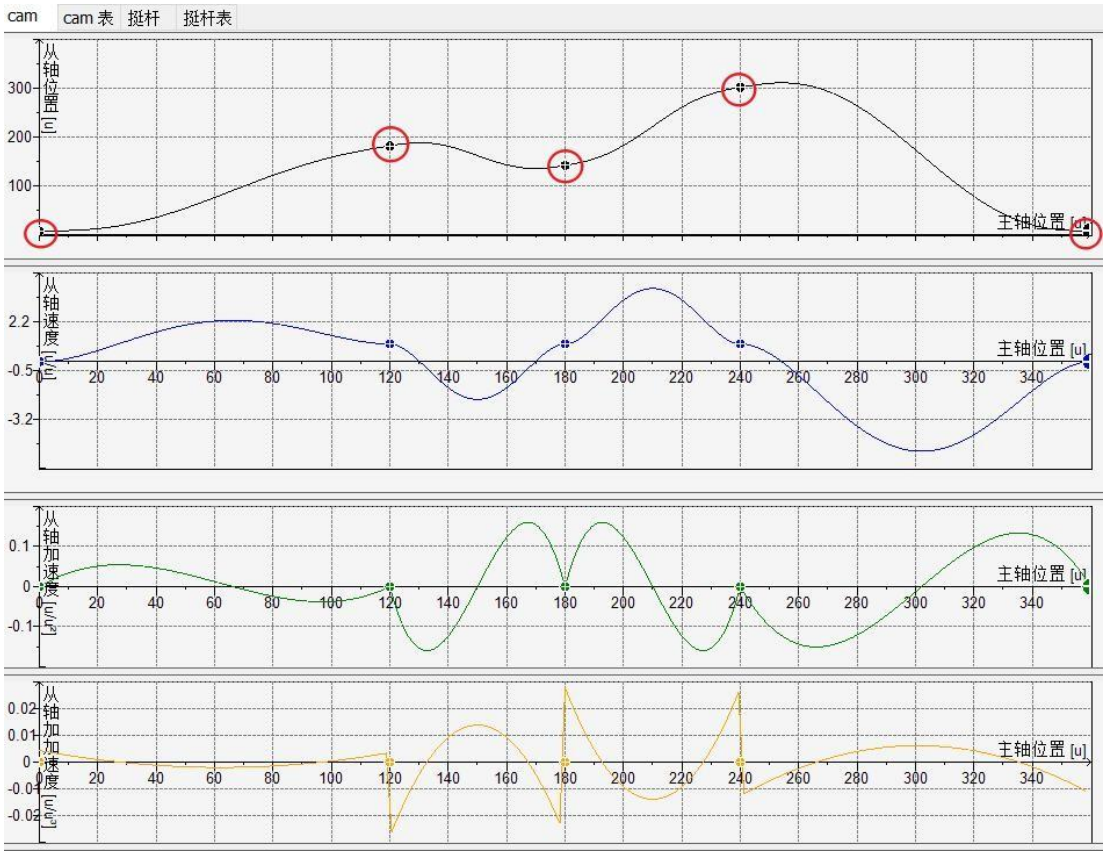
新建凸轮表方法如下：

“右键Application—添加对象—Cam表...”



在编写凸轮运行的用户程序时，凸轮表是其中的一个编写项，它决定了凸轮运行的特性，可以采用图形和表格的方式输入。

如下图为图形方式的 CAM 凸轮表，横坐标为主轴位置轴，轴的长度为凸轮运行的行程， 共有 4 条坐标曲线，纵轴分别为从轴位置、从轴的速度、从轴



的加速度、从轴的加加速度曲线。编程调试时，往往更关注位置曲线、速度曲线，调试平稳性时还会关注加速度曲线。

凸轮曲线有如下特点：

在主从位置曲线坐标中，垂直轴即为从轴可能运动的范围；

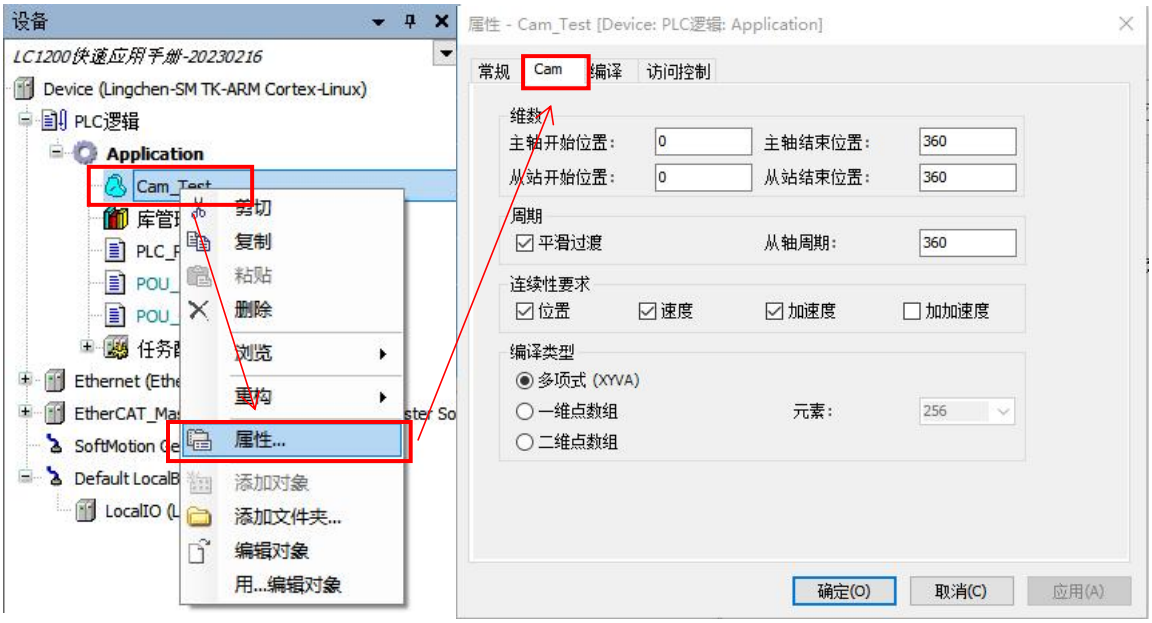
其他 3 个曲线的垂直轴则是从站与主轴的速度之比、从轴与主站的加速度之比；

凸轮曲线在垂直方向是一个单调曲线，即主轴的每一个坐标，只能对应从轴唯一的坐标值；

凸轮执行时，主轴坐标是按由小到大的方向运动；

凸轮曲线可以有若干个关键点，两个关键点之间的线型可以分别设定直线或 5 次曲线， 系统会为每条 5 次曲线作最佳优化，以尽量减小速度和加速度的突变；

水平轴（主轴）的起止坐标默认从 0 开始，360 结束，用户可以根据实际的物理行程进行修改。



6.2.2 凸轮表输入

① 当新建一个凸轮表时，系统会自动设置一个最简单的凸轮曲线，用户在此基础上进行修改，形成自己所需的 CAM 曲线表；

② 用户可以增减凸轮曲线的关键点个数、修改关键点的坐标；

③ 用户可以修改任意两个相邻关键点之间的线型，或 5 次曲线，或直线；

④ 凸轮曲线中的关键点之间，系统会默认按 5 次曲线进行联接，这样可以确保运行时速度的连续性，减小机械冲击。

凸轮曲线中的关键点，往往与控制对象的机械运动要求相关，例如：

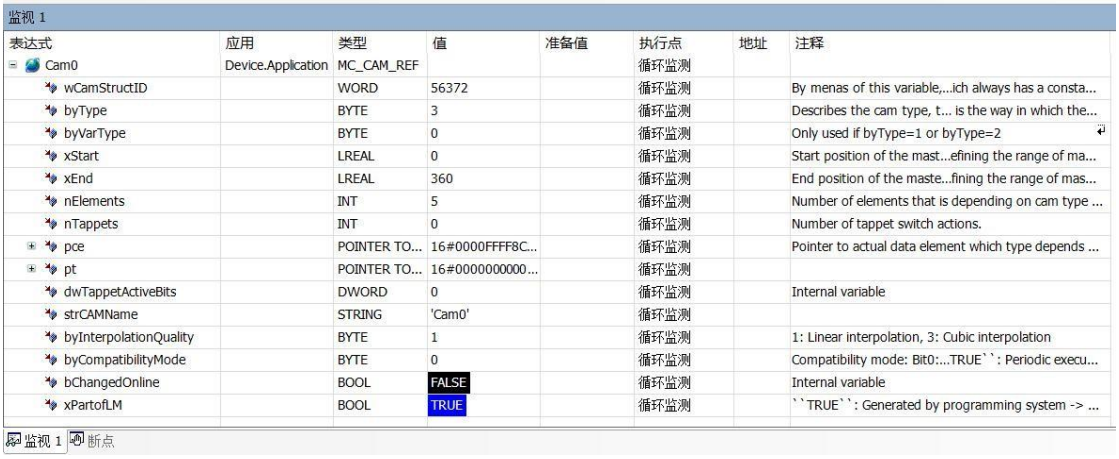
① 对于追剪应用，主轴的坐标范围建议对应运行区间的物理行程，便于分析；

② 从轴往返行程起止点、同步运行区间的起始位置点、脱离同步位置点等，就是重要的关键点；

③ 比例同步区间，凸轮曲线的线段应该为直线，其他区间则为 5 次曲线。

6.2.3 CAM 凸轮表的内部数据结构与数组

在 CODESYS 中，对每一个 CAM 表，都有描述该 CAM 表的数据结构，描述该 CAM 表的特征数据。下图为“CAM0”凸轮表的描述数据结构，请注意其结构各变量名称：



在内部有一个数据结构来描述 CAM 凸轮表的特征：如果我们手动编写一个 CAM 表也是可以的，如下：

虽然我们不需要手动编写 CAM 表，但我们可以通过数据结构的访问操作，对所需的CAM 特性数据进行修改。

【注意：我们在声明 CAM_Test 凸轮表时，系统自动默认声明了全局变量类型的 CAM_Test 数据结构，同时声明了 CAM_Test_A[i] 数组。】

例如在用户程序中，修改 CAM_Test 凸轮表关键点个数或坐标：

```
1  CAM_Test.nElements:=20; // 将关键点个数改为20个
2  CAM_Test.xEnd:=500; // 将主轴的结束点改为500
3  // 例如在用户程序中，修改其中 2 个关键点的坐标：
4  CAM_Test_A[3].dx:=30;
5  CAM_Test_A[3].dy:=45;
6  CAM_Test_A[3].dv:=1;
7  CAM_Test_A[3].da:=0;
8  CAM_Test_A[4].dx:=60;
9  CAM_Test_A[4].dy:=75;
10 CAM_Test_A[4].dv:=1;
11 CAM_Test_A[4].da:=0;
```

在线修改 CAM 凸轮表的方法

所谓“在线修改CAM曲线”，是指用户编写的程序在执行过程中，根据控

制特性的需要，对CAM曲线的关键点坐标，进行的修改。修改的内容，一般是修改关键点坐标，但也可以修改关键点的个数、修改主轴的距离范围等。

提醒：在进入凸轮运行之前，修改凸轮表，不宜运行中修改，避免出现预想不到的运动结果需要修改CAM凸轮表应用场合：

①一般情况下，OEM客户使用调试验证成功的凸轮表；

②若有几种加工对象或模式，可以考虑预设多个凸轮表，根据用户的需要，自动进行切换；

③有些设备，要求其适应范围更宽，例如包装设备，若要求其适用的包装长度在10cm~25cm 范围，且对应运行速度作自动适应改变时，可能必需在线修改 CAM 凸轮表。

6.2.4 CAM 凸轮表的引用与动态切换

CAM 凸轮表在控制器内部是用一个数组来保存，可以通过特定的 MC_CAM_REF 变量类型来指向，例如声明：

凸轮表 p: MC_CAM_REF

可以给该变量赋值，也可认为是将其指向某具体的凸轮表：

```
凸轮表 p:= Cam0; // 指向所需的凸轮表
凸轮表 p: MC_CAM_REF; // 凸轮表指针；
TableID: uint; // 凸轮表选择命令，可由 HMI 设置；
Case TableID of
0: 凸轮表 p := 凸轮表 A;
1: 凸轮表 p := 凸轮表 B;
2: 凸轮表 p := 凸轮表 C;
End_case
MC_CamTableSelect_0( // 凸轮关系
Master:= 凸轮主轴 ,
Slave:= 凸轮从轴 ,
CamTable:= 凸 轮 表 p,
Execute:= ReSelect, // 上升沿触发凸轮表选择
Periodic:= TRUE,
MasterAbsolute:=FALSE,
SlaveAbsolute:= FALSE);
```

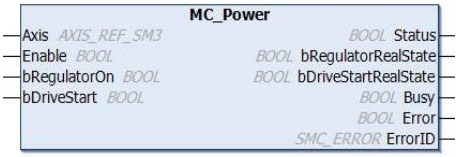
上面的例程， 利用该 MC_CAM_REF 变量的赋值运算，就可以实现多个

凸轮表的切换操作了。

6.3 单轴指令

6.3.1 MC_Power

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_Power	轴使能指令		MC_Power(Axis:= , Enable:= , bRegulatorOn:= , bDriveStart:= , Status=> , bRegulatorRealState=> , bDriveStartRealState=> , Busy=> , Error=> , ErrorID=>);

(2)相关变量

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即AXIS_REF_SM3的一个实例
Enable	输入有效	BOOL	TRUE, FALSE	FALSE	设置为 TRUE 则功能块开始处理
bRegulatorOn	使能状态	BOOL	TRUE, FALSE	FALSE	设置为 TRUE 则设置轴为使能状态
bDriveStart	允许驱动	BOOL	TRUE, FALSE	FALSE	设置为 TRUE 以关闭功能块的紧急停止处理

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Status	可运行状态	BOOL	TRUE, FALSE	FALSE	如果轴已经准备好运动，置为 TRUE
bRegulatorRealState	轴使能信号状态	BOOL	TRUE, FALSE	FALSE	当轴使能处于有效状态时，置为 TRUE
bDriveStartRealState	允许驱动状态	BOOL	TRUE, FALSE	FALSE	如果轴没有被快速停止机制中断，置为 TRUE
Busy	执行中	BOOL	TRUE, FALSE	FALSE	如果功能块的处理没有完成，置为 TRUE

Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
ErrorID	错误代码	SMC_ERROR	参阅SMC_ERROR	0	异常发生时，输出错误代码

(3)功能说明

只有在输入Enable为TRUE的时候，其它的输入才会被功能块处理。

如果功能块MC_Power已经被调用，并且bRegulatorOn=FALSE，那么功能块将设置相关轴的轴状态（nAxisState）为power_off状态，表明驱动器还没有做好运动准备。

如果功能块MC_Power已经被调用，并且bRegulatorOn=TRUE，如果此时轴没有错误发生，那么功能块将设置相关轴的轴状态（nAxisState）为standstill状态；如果有错误发生，将输出相应的错误状态。

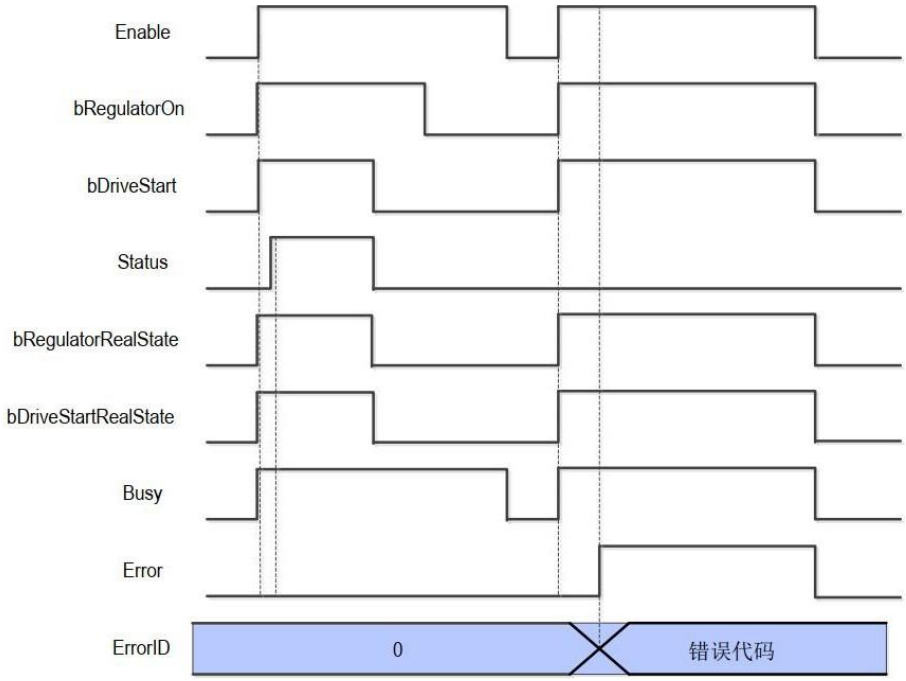
如果Enable，bRegulatorOn以及bDriveStart都为TRUE，但是输出Status在一定时间后仍为FALSE，那么输出Error将会被置位。当在使能状态情况下产生一个硬件问题，可能会发生这种情况。

如果使能信号丢失（通常在操作模式下），相关轴的nAxisState将会被置位ErrorStop状态。

使用时要注意Enable和bRegulatorOn的操作次序。Enable可保持高电平状态，通过控制bRegulatorOn，控制伺服使能开断。不要同时开断Enable和bRegulatorOn。因为在Enable无效后，功能块不再执行，此时改变bRegulatorOn也无法生效，就会导致“明明已经复位bRegulatorOn，伺服仍然处于使能状态”的现象。

◆ 时序图

将Enable设为TRUE，bRegulatorOn设为TRUE，bDriveStart设为TRUE，表示指令正在受理的Busy变为TRUE，然后轴进入使能ON状态，Status状态变为TRUE。



(4)错误说明

请勿在正在执行MC_Power指令的轴中，编写用于启动其它实例的MC_Power指令的程序。原则上1个轴只能设1个MC_Power指令。如果在正在执行MC_Power指令的轴中，启动其它实例的MC_Power指令，则优先执行后执行的MC_Power指令。

【注】请阅读“附录C错误代码说明”以了解相关错误代码说明。

6.3.2 MC_Stop

MC_Stop将轴置于停止状态。功能块实例的当前正在运行的运动被中止。

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_Stop	轴停止命令		MC_Stop(Axis:= , Execute:= , Deceleration:= , Jerk:= , Done=> , Busy=> , Error=> , ErrorID=>);

(2)相关变量

◆ 输入输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Deceleration	减速度	LREAL	“正数”、“0”	0	功能块的减速度 (u/S^2)
Jerk	速度变化率	LREAL	“正数”、”0”	0	速度变化率（u /S^3）

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

(3)功能说明

本功能块为在正常运行的情况下停止一个轴的运动，当轴处于 stopping 状

态对本轴的任何指令都是无效的。

本功能块在轴为运行状态 (Motion) 才能运行，其他状态无法运行。启动指令为Execute 的上升沿启动。在 MC_Stop 有效执行的过程 Busy 有效时，再一次启动 MC_Stop 会令指令系统变成 Errorstop 状态。

◆ 时序图

轴必须处于运行状态 (Motion)，MC_Stop 指令才能运行；

功能块的 Execute 必须有上升沿的条件；

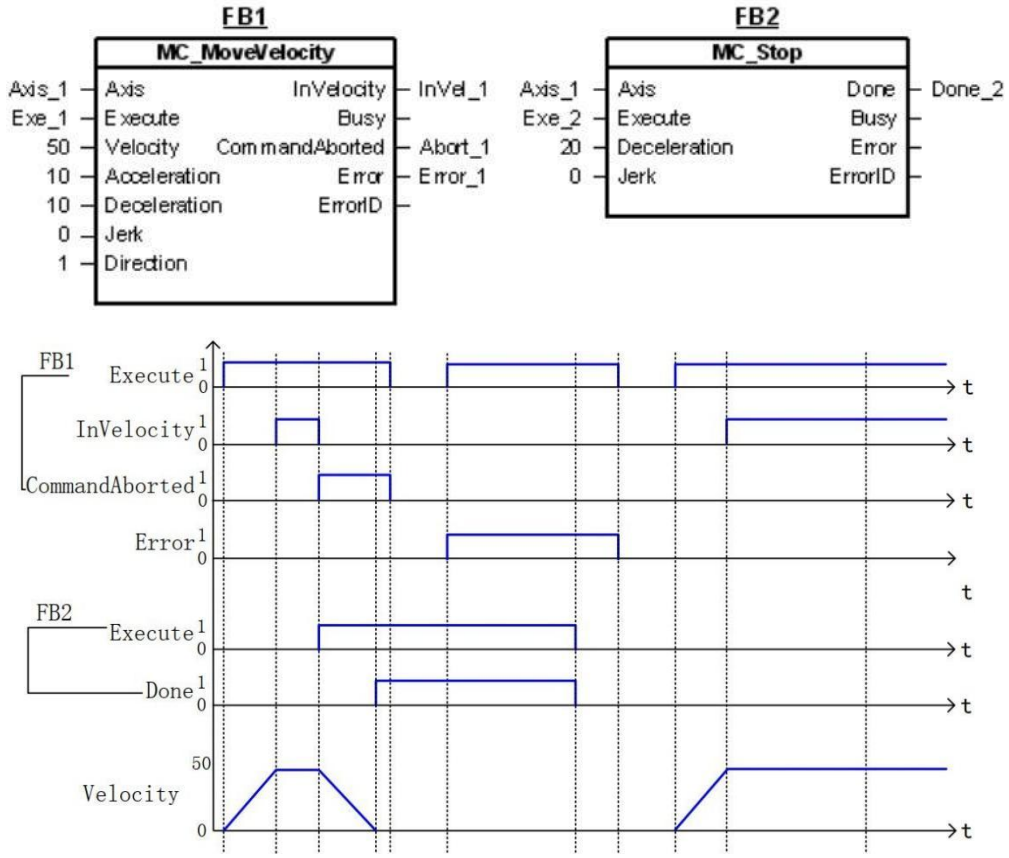
功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；

功能块的 CommandAborted 表示指令被其他运动控制指令中断，此时标志位为 TRUE；

例程：在执行 MC_MoveVelocity 指令和 MC_Stop 指令在不同的时序操作中对应的 标志位的变化；

对 CommandAborted 的处理描述如下图的时序描述。



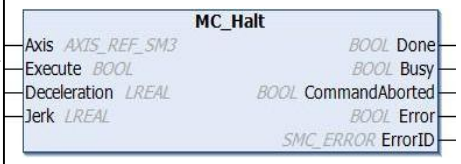
(4)错误说明

MC_Stop有重复性的指令运行时,错误标志Error为True， ErrorID为SMC_MS_AXI错误。

【注意】：请阅读“附录C错误代码说明”以了解相关错误代码说明。

6.3.3 MC_Halt

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_Halt	轴正常暂停 命令		MC_Halt(Axis:= , Execute:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

(2)相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE, FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Deceleration	减速度	LREAL	“正数”“ 0”	0	功能块的减速度 (u/S^2)
Jerk	跃度	LREAL	“正数”“ 0”	0	指定跃度［指令单位 /S^3］

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中， 置为 TRUE
Command Aborted	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参 阅 SMC_ ERROR	0	异常发生时，输出错误代码

(3)功能说明

该功能块以受控方式停止参考轴。如果此时正在运行其他功能块的动作，则这些动作将中止。轴进入离散运动状态，直到速度达到0。如果设置了MC_Halt的“完成”输出，则轴的状态将变为静止。只要MC_Halt处于活动状态，就可以

发出新的运动命令来中断MC_Halt的执行，这点与MC_Stop不同，MC_Halt是可以被中断。

本功能块只有在运行状态(Motion)才能运行,其他状态无法运行。

启动指令为Execute的上升沿启动；指令运行中的状态为DiscreteMotion，运行完成后状态为Standstill。

(4)时序图

轴必须处于运行状态(Motion)指令才能运行；

功能块的Execute必须有上升沿的条件；

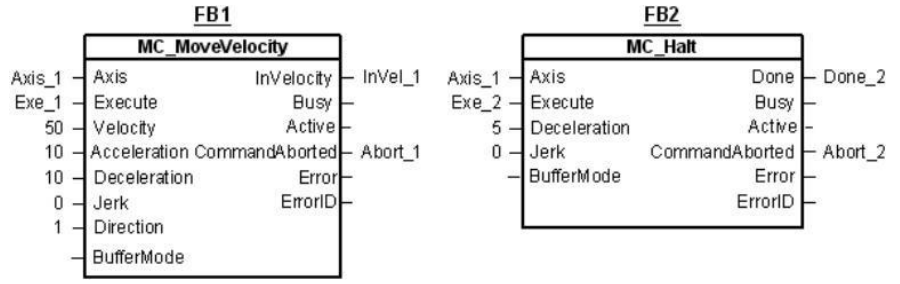
功能块的Done表示指令正常执行完成；

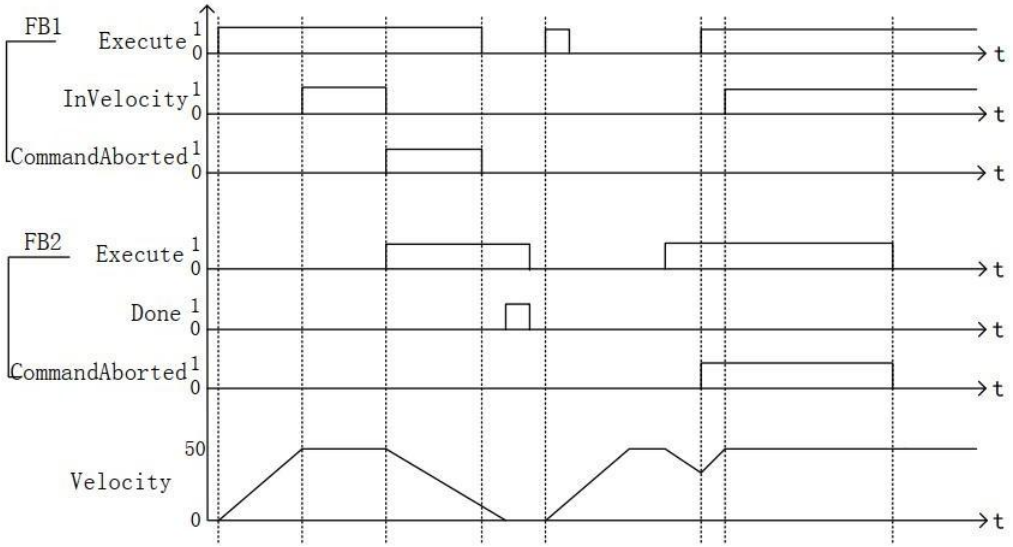
功能块的Busy表示当前功能块正在执行中；

功能块的CommandAborted表示指令被其他运动控制指令中断，此时标志位为TRUE；

例程：在执行MC_MoveVelocity指令和MC_Halt指令在不同的时序操作中对应的标志位的变化；

对CommandAborted的处理描述如下图的时序描述。





(5)错误说明

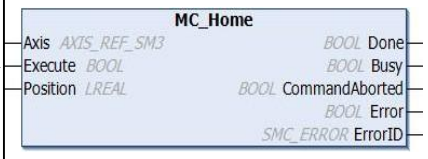
错误的出现为轴状态不是在Standstill中启动指令或指令系统中的参数错误，出现轴错误只能清除错误后才开始运行。

【注意】：请阅读“附录C错误代码说明”以了解相关错误代码说明。

6.3.4 MC_Home

它的执行将使轴执行“搜索原点”序列。该序列的详细信息取决于制造商，可以通过轴参数进行设置。当检测到参考信号时，位置输入用于设置绝对位置。功能块停止并终止。

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_Home	轴回零指令		MC_Home(Axis:= Axis, Execute:= , Position:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

(2)相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Position	轴到达位置	LREAL	数据范围	0	代表轴位置的回零位置

◆ 输出变量

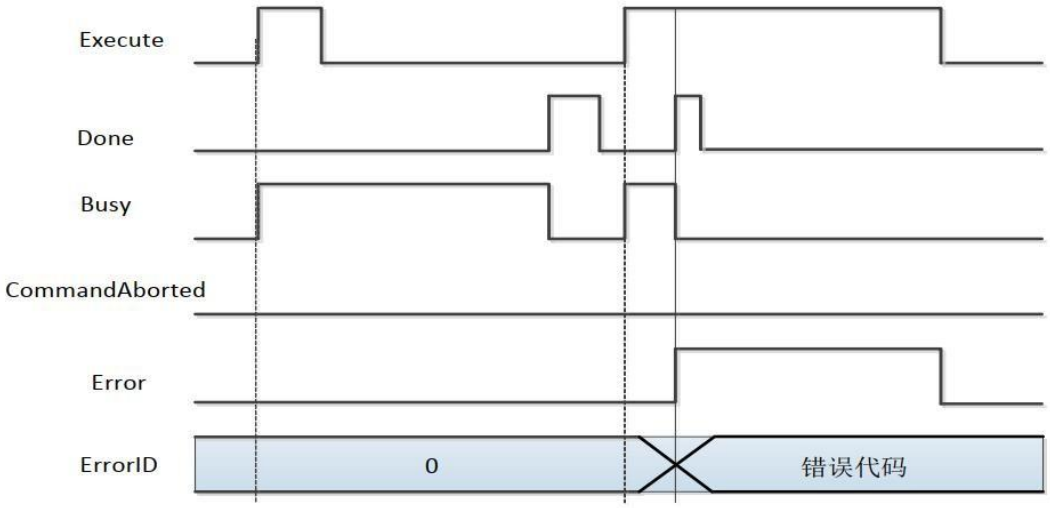
输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Command Abort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参 阅 SMC_ ERROR	0	异常发生时，输出错误代码

(3)功能说明

本功能块为回零操作，Position数据为轴的零点位置。

本功能块运行状态为 Standstill 中，指令运行时的状态为 homing，其他状态无法运行。启动指令为 Execute 的上升沿启动指令。

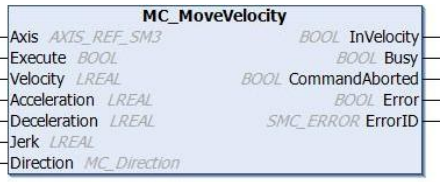
(4)时序图



6.3.5 MC_MoveVelocity

使用驱动器位置控制模式下的进行模拟速度控制，在轴使能并指令有效的情况下，对Velocity的赋值能控制驱动器的速度。

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_MoveVelocity	速度控制指令		MC_MoveVelocity(Axis:= , Execute:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Direction:= , InVelocity=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

(2)相关变量

◆ 输入输出变量

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	—	—	映射到轴，即AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Velocity	速度	LREAL	数据范围	0	此数据为本指令的速度运行值
Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值
Jerk	跃度	LREAL	数据范围	0	曲线加减速的斜率变化值
Direction	运行方向	MC_Direction	1：positive(正向) -1：negative(反向) 2：current(当前，模轴)	current	为运行方向的指令操作

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InVelocity	达到设定速度的标志	BOOL	TRUE,FALSE	FALSE	设定的运行速度达到，置为 TRUE

Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断， 置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参 阅 SMC_ ERROR	0	异常发生时，输出错误代码

（3）功能说明

更改 Velocity 参数，对驱动器的模拟速度控制。

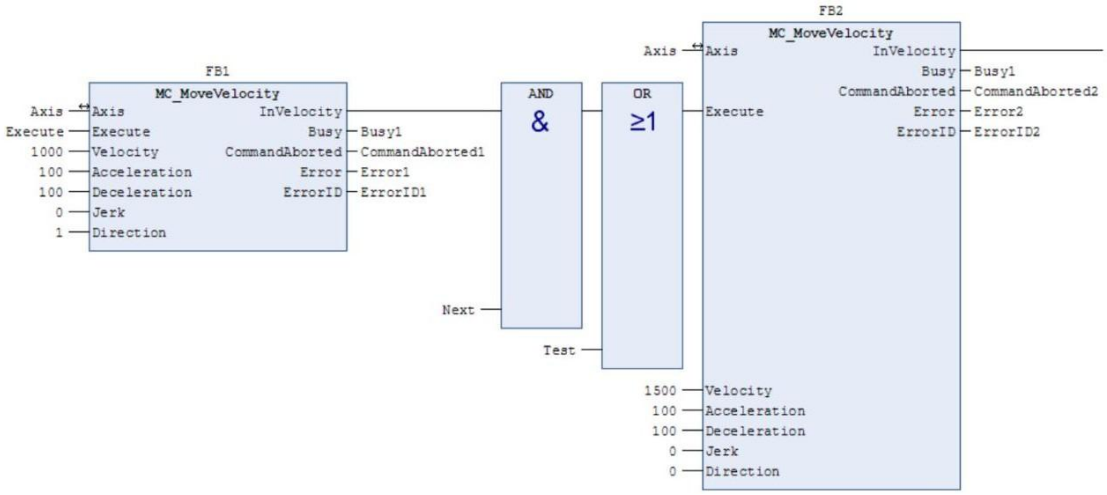
◆ 时序图

功能块的 Execute 必须有上升沿的条件；

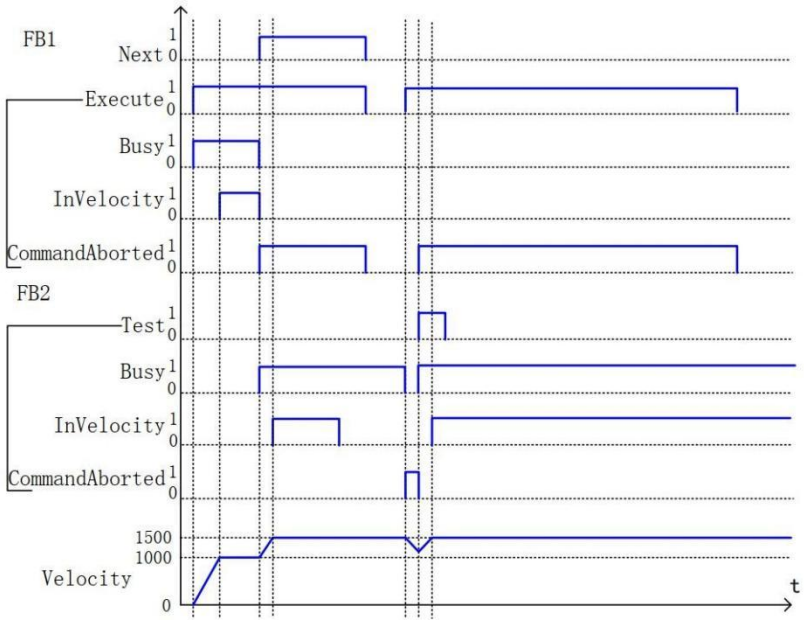
功能块的 InVelocity 表示指令的运行速度达到设定值；

功能块的 Busy 表示当前功能块正在执行中。

◆ 举例说明



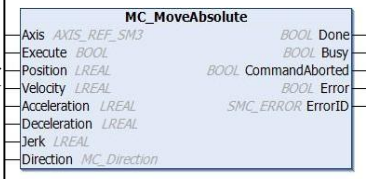
时序操作说明 ：


FB1
Next
Execute
Busy
InVelocity
CommandAborted
FB2
Test
Busy
InVelocity
CommandAborted
1500
1000
Velocity
0
t

6.3.6 MC_MoveAbsolute

轴按绝对位置运行(单位按轴设置),绝对位置由Position指定；本指令运行前设置好相关的参数,加速度(Acceleration)、减速度(Deceleration)、运行速度(Velocity)和加减速模式的跃度(Jerk)；对加速度(Acceleration)或减速度(Deceleration)的赋值为0时指令运行错误。

（1）指令格式

指令	名称	图形表现	ST 表 现
MC_MoveAbsolute	轴绝对位置控制指令		MC_MoveAbsolute(Axis:= , Execute:= , Position:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Direction:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

（2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块
Position	轴到达位置	LREAL	数据范围	0	此位置为轴的绝对位置数据
Velocity	运行速度	LREAL	数据范围	0	轴运行到目标位置的最大速度
Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值
Jerk	跃度	LREAL	数据范围	0	曲线加减速的斜率变化值
Direction	指令方向	MC_DIRECTION	Negative,shortestPositive,current,fastest	shortest	Negative:反向移动; Shortest:根据最短路径选择方向; Positive:正向移动; Current:按当前方向移动; Fastest:自动选择最快到达的方向移动

					(此功能轴在旋转模式下有效)

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE
Command Abort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码

(3)功能说明

◆ 本功能块为轴绝对定位指令,Position数据为轴的绝对位置。

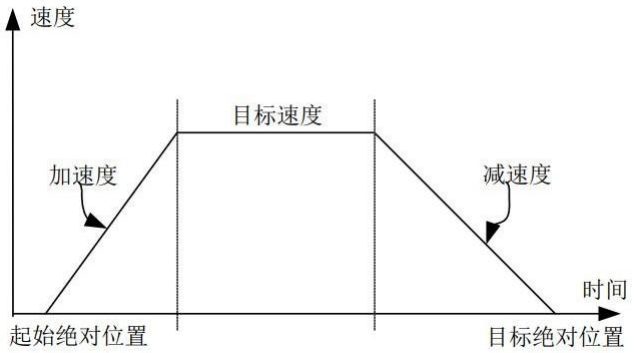
◆ 本功能块运行状态为Standstill中,指令运行时的状态为DiscreteMotion,一个完整的运行过程一定要控制轴的不同运动状态。

◆ 启动指令为Execute的上升沿启动,本指令在DiscreteMotion可以重复上升沿有效,每次都可以刷新最新的Position位置。

◆ Acceleration或Deceleration为零，指令运行都为异常状态，但轴的状态为DiscreteMotion;

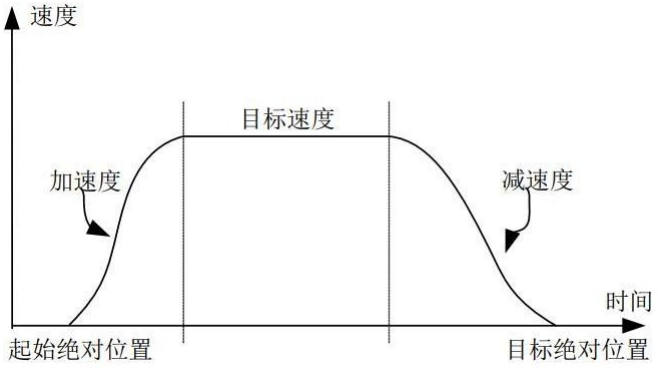
◆ 梯形加减速动作

Velocity、Acceleration和Deceleration有数据；而Jerk为0;



◆ S

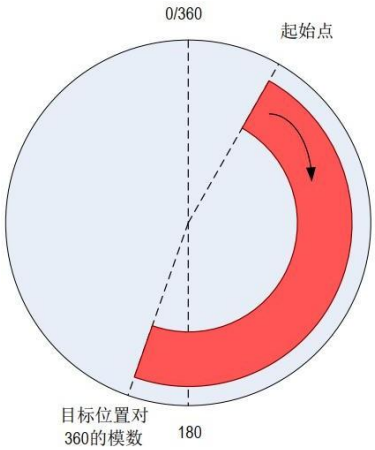
曲线加减速动作Velocity、Acceleration、Deceleration和Jerk都有数据;



◆ 轴在周期模式下绝对定位

①轴旋转周期设置为360、Direction设置为Positive。

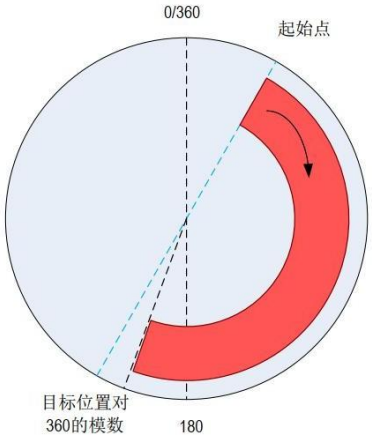
当Position对360的模值（Position/360取余，比如Position为380则对360的模值为20，Position为350，则对360模值为350）>起始绝对位置时，此时轴朝正向运行(Position对360的模值-起始绝对位置)的距离。



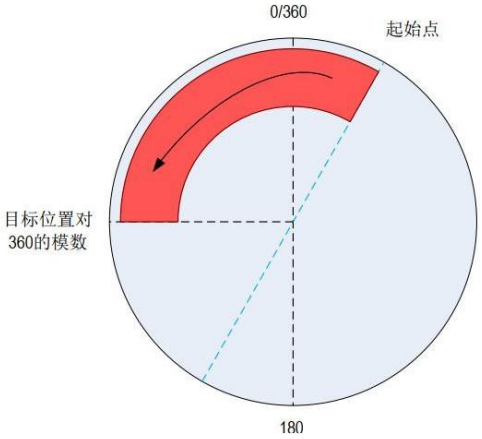
当Position对360的模值（Position/360取余，比如Position为380则对360的模值为20）<起始绝对位置时，此时轴朝正向运行(360-起始绝对位置+Position对360的模值)的距离。

②轴旋转周期设置为360、Direction设置为shortest或者fastest。Position对360的模值为XPosition

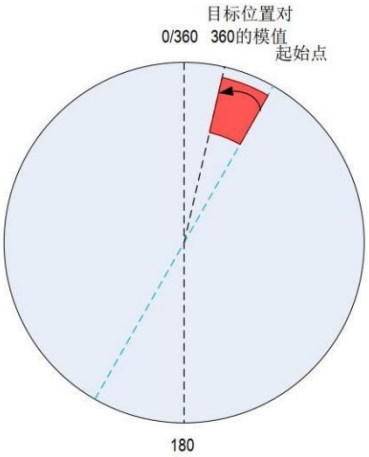
当0<XPosition－起始绝对位置<180时,轴朝正向运行（XPosition－起始绝对位置）的距离。



当180<XPosition-起始绝对位置时,轴朝反向运行360-XPosition+起始绝对位置的距离。



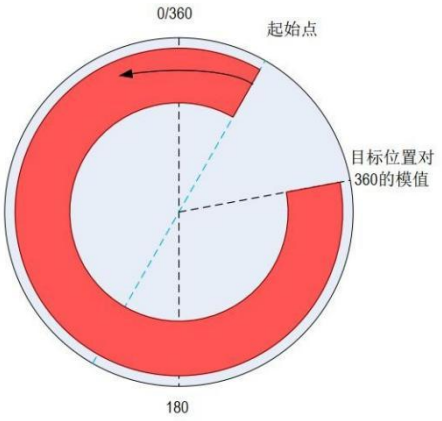
当 XPosition< 起始绝对位置时 ，轴朝反向运行 起始绝对位置－XPosition 的距离。



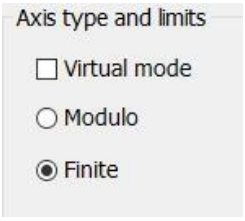
③轴旋转周期设置为360，Direction设置为shortest或者Negative。Position对360的

模值为XPosition

轴朝反向运行 起始绝对位置+360-XPosition 的距离。



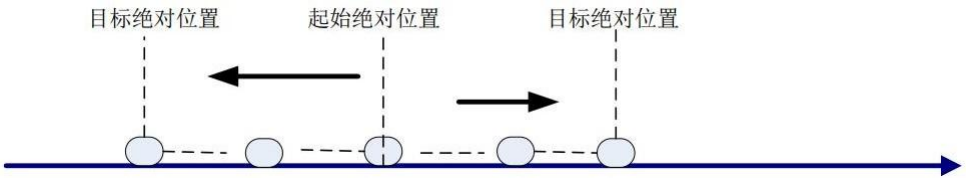
◆ 轴在线性模式下绝对定位



当目标绝对位置 > 起始位置，则正向移动目标（绝对位置 - 起始位置距离）

当目标位置 < 起始位置时，则反向移动（起始位置 - 目标位置的距离）。

线性模式下设定的运行方向不决定轴运行方向，即 Direction 无效。



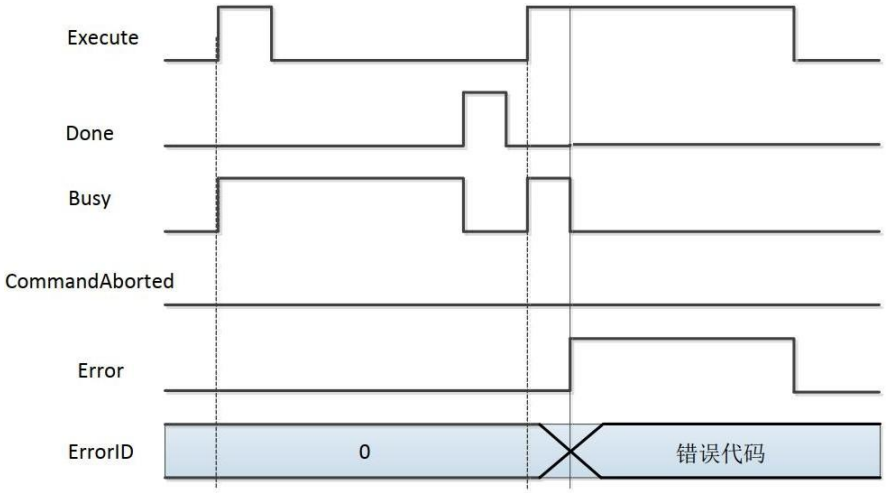
◆ 时序图

轴必须处于 Standstill 状态指令才能运行；

功能块的 Execute 必须有上升沿的条件；

功能块的 Done 表示指令正常执行完成；

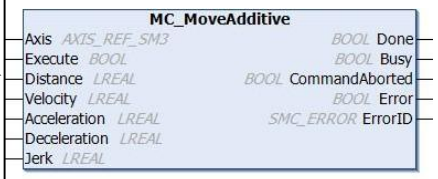
功能块的 Busy 表示当前功能块正在执行中。


Execute
Done
Busy
CommandAborted
Error
ErrorID
0
错误代码

6.3.7 MC_MoveAdditive

轴在原来指令位置上再叠加Distance指定的数据，用于运动轴控制过程的在线叠加位置；本指令运行前设置好相关的参数,加速度(Acceleration)、减速度(Deceleration)、运行速度(Velocity)；对加速度(Acceleration)或减速度(Deceleration)的若赋值为0则指令运行错误；本指令在DiscreteMotion状态下可以在任何时刻添加MC_MoveAddtive的相关执行过程；在ContinuousMotion中只能在指令执行的某段中；MC_MoveAddtive在standstill状态下执行相当于MC_MoveRelative指令。

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_Move Additive	叠加绝对运动指令		MC_MoveAdditive(Axis:= , Execute:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

(2)相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Distance	轴到达位置	LREAL	数据范围	0	此数据为叠加位置数据
Velocity	运行速度	LREAL	数据范围	0	轴运行到目标位置的最大速度
Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值
Jerk	跃度	LREAL	数据范围	0	曲线加减速的斜率变化值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为TRUE

Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE
Command Abort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
ErrorID	错误代码	SMC_ ERROR	参阅SMC_ ERROR	0	异常发生时，输出错误代码

（3）功能说明

本功能块为叠加位置指令,Distance数据为轴的叠加数据；

本功能块运行状态如果为DiscreteMotion,使用情况会把其他指令的CommandAbort置位；

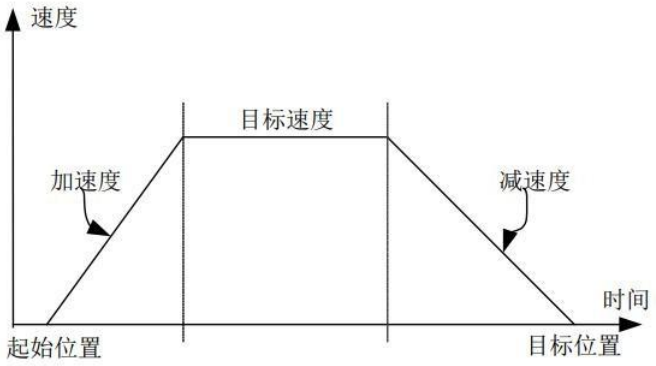
在standstill状态，本指令能独立运行，实现相对定位需求；

Acceleration或Deceleration为零，指令运行都为异常状态，但轴的状态为DiscreteMotion；

启动指令为Execute的上升沿启动。

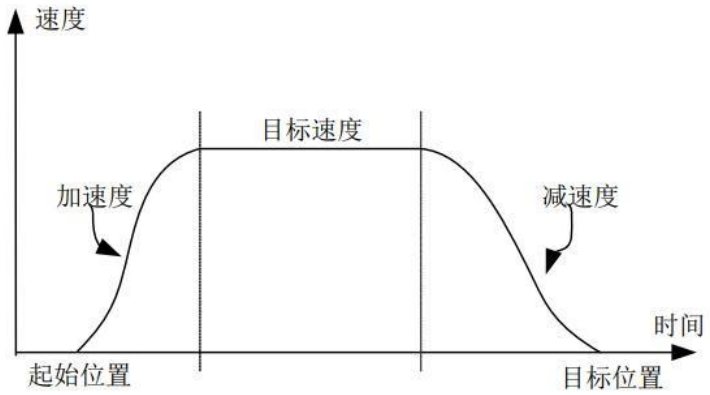
◆ 梯形加减速动作

Velocity、Acceleration和Deceleration有数据；而Jerk为0；



◆ S

曲线加减速动作Velocity、Acceleration、Deceleration和Jerk都有数据。



(4)时序图

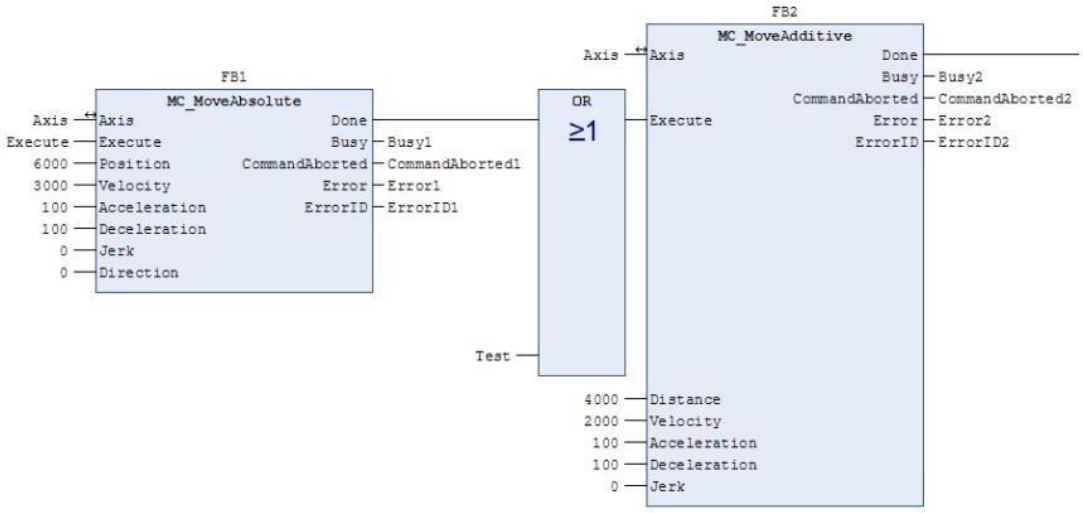
轴必须处于Standstill状态指令才能运行；

功能块的Execute必须有上升沿的条件；

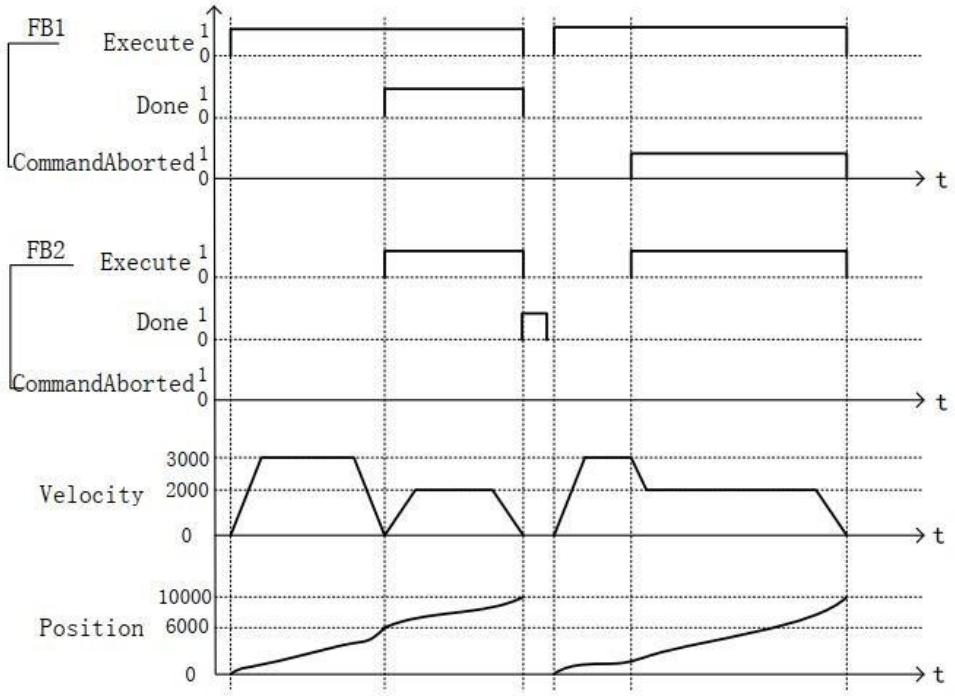
功能块的Done表示指令正常执行完成；

功能块的Busy表示当前功能块正在执行中；

◆ 举例说明：



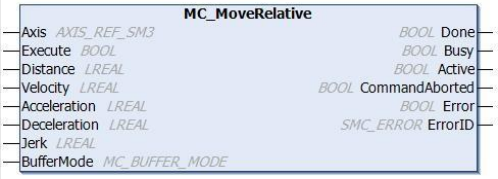
◆ 时序操作说明 ：


FB1
Execute
Done
CommandAborted
FB2
Execute
Done
CommandAborted
Velocity
3000
2000
0
Position
10000
6000
0
t

6.3.8 MC_MoveRelative

轴按相对位置运行。相对位置由Distance指定(单位按轴设置)；本指令运行前设置好相关的参数,加速度(Acceleration)、减速度(Deceleration)、运行速度(Velocity)、加减速模式的跃度(Jerk)和缓冲模式(BufferMode)。

（1）指令格式

指令	名称	图形表现	ST 表 现
MC_ MoveRelative	轴相对定位指令		MC_MoveRelative(Axis:= , Execute:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , BufferMode:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

（2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Distance	运动相对位置	LREAL	数据范围	0	此数据为运动的相对位置
Velocity	运行速度	LREAL	数据范围	0	轴运行到目标位置的最大速度
Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值
Jerk	跃度	LREAL	数据范围	0	曲线加减速的斜率变化值
BufferMode	缓冲模式	MC_BUFFER_MODE	Aborting；Buffered;BlendingLow;BlendingPrevious;BlendingNext;BlendingHigh;	Aborting	定义此FB相对于前一个功能块的时间顺序。如果功能块是Busy，那么只允许使用BufferMode=Aborting。

BufferMode（缓冲模式）	介绍
（注：CODESYS SoftMotion 版本位4.8.0.0 才有此项）	
Aborting	立即启动 FB（默认模式）
Buffered	在当前队列中的最后一个动作结束后启动 FB
BlendingLow	路径与两个 FB 提供的最小动态限制相融合
BlendingPrevious	路径与第一个 FB 的动态限制融合在一起
BlendingNext	路径与第二个 FB 的动态极限混合
BlendingHigh	路径与两个 FB 提供的最大动态限制相融合

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断， 置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参 阅 SMC_ ERROR	0	异常发生时，输出错误代码

(3)功能说明

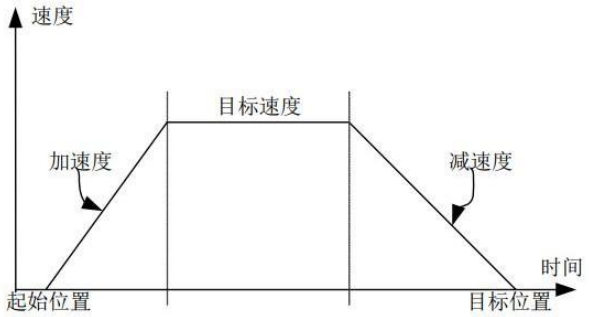
本功能块运行状态为 Standstill 中， 指令运行时的状态为 Discrete Motion，在指令执行中关注本轴的运行状态，避免打断本轴的其他指令或被其他指令打断本轴的执行。

启动指令为 Execute 的上升沿启动 ，本指令在 Discrete Motion 可以重复上升沿有效 ，每次都可以刷新最新的 Position 位置。

Acceleration 或 Deceleration 为零，指令运行都为异常状态，但轴的状态为 Discrete Motion。

◆ **梯形加减速动作**

Velocity、Acceleration和Deceleration有数据；而Jerk为0；



◆ **S**

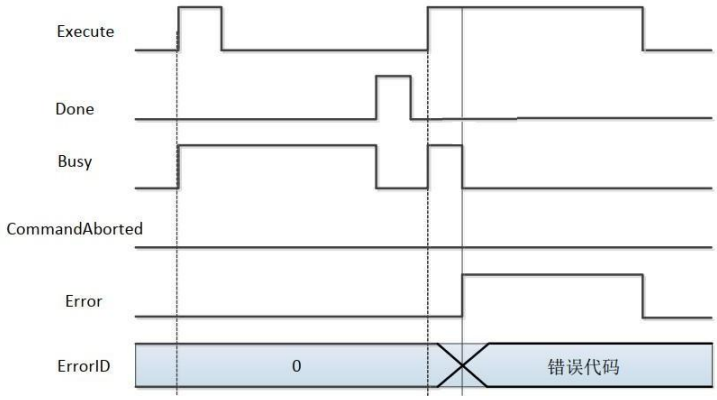
Velocity、Acceleration和Deceleration和Jerk都有数据。

(4)时序图

功能块的 Execute 必须有上升沿的条件；

功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中。



6.3.9 MC_MoveSuperImposed

轴在原来指令速度和位置的基础上叠加速度和位置的数据在运行的指令上，对整个原来的指令执行时间模型上没有变化；通过此指令能解决我们实际运行中一些类似于由皮带和齿轮间隙误差补偿，能保证运动的一致性；

指令运行时需要设置参数叠加位置(Distance)、速度(VelocityDiff)、加速度(Acceleration)、减速度(Deceleration)、运行速度(Velocity)；对加速度(Acceleration)或减速度(Deceleration)的赋值为0指令运行错误；MC_MoveSuperImposed在standstill状态下相当于MC_MoveRelative指令。

（1）指令格式

指令	名称	图形表现	ST 表 现
MC_ MoveSuperImposed	叠加相对运动指令		MC_MoveSuperImposed(Axis:= , Execute:= , Distance:= , VelocityDiff:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

（2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即AXIS_REF_SM3的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Distance	轴到达位置	LREAL	数据范围	0	此数据为叠加位置数据
VelocityDiff	叠加速度	LREAL	数据范围	0	轴运行叠加速度
Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值
Jerk	跃度	LREAL	数据范围	0	曲线加减速的斜率变化值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成， 置为TRUE

Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
ErrorID	错误代码	SMC_ERROR	参阅SMC_ERROR	0	异常发生时，输出错误代码

（3）功能说明

本功能块为叠加位置和速度指令，VelocityDiff 和 Distance 分别为叠加在其他指令上的速度和位置；

在运动模式下 MC_MoveSuperImposed 可以叠加在任何其他指令；

MC_MoveSuperImposed 也可以被 MC_MoveSuperImposed 中止；

在状态 StandStill 下，功能块 MC_MoveSuperimposed 的动作类似于 MC_MoveRelative； 启动指令为 Execute 的上升沿启动。

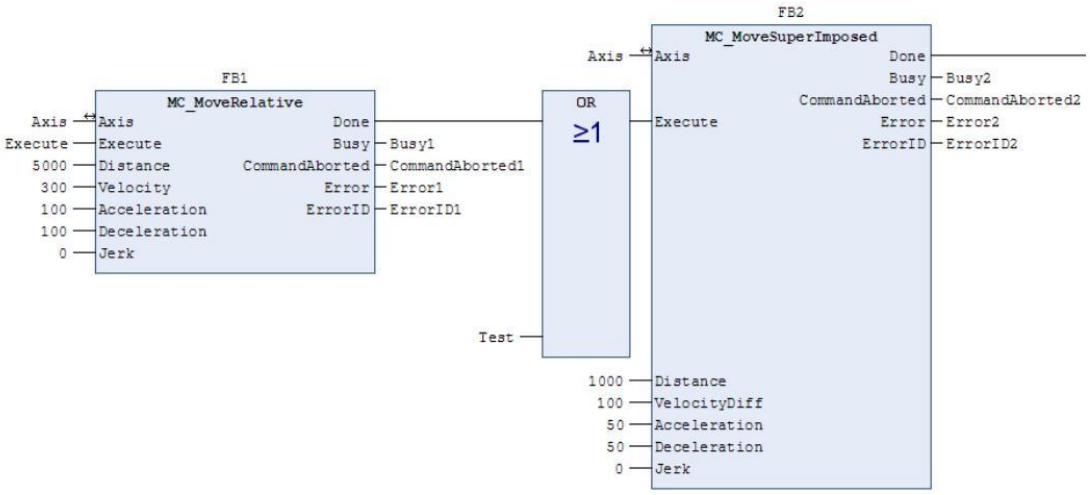
◆ 时序图

功能块的 Execute 必须有上升沿的条件；

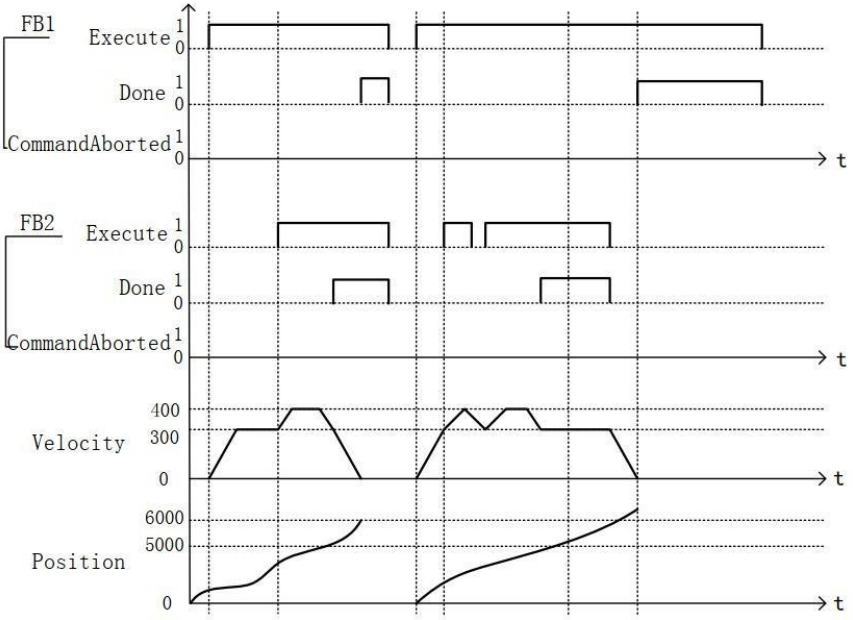
功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中。

(a)举例说明


FB1
MC_MoveRelative
Axis
Execute
5000
300
100
100
0
Axis
Execute
Distance
Velocity
Acceleration
Deceleration
Jerk
Done
Busy
CommandAborted
Error
ErrorID
Busy1
CommandAborted1
Error1
ErrorID1
OR
≥1
Test
FB2
MC_MoveSuperImposed
Axis
Execute
1000
100
50
50
0
Distance
VelocityDiff
Acceleration
Deceleration
Jerk
Done
Busy
CommandAborted
Error
ErrorID
Busy2
CommandAborted2
Error2
ErrorID2

(b)时序操作说明 ：


FB1
Execute
Done
CommandAborted
FB2
Execute
Done
CommandAborted
1
0
t
Velocity
400
300
0
Position
6000
5000
0

6.3.10 MC_PositionProfile

该功能块设计用于命令时间位置锁定的运动曲线。

(1) 指令格式

指令	名称	图形表现	ST表现
MC_PositionProfile	位置轮廓指令		MC_PositionProfile(Axis:= , TimePosition:= , Execute:= , ArraySize:= , PositionScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

(2)相关变量

◆ 输入输出变量

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	—	—	映射到轴，即AXIS_REF_SM3的一个实例
TimePosition	轴位置运行时间和位置描述	MC_TP_REF			轴位置运行时间和位置数据描述，数据由多组数据组成

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
ArraySize	动态数组	INT	数据范围	0	运行轮廓中使用的数组个数
PositionScale	综合因子	LREAL	“正数”+“0”	1	MC_TP_REF中位置的比例因子
Offset	偏移	LREAL		0	位置的总体偏移值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为

					TRUE
ErrorID	错误代码	SMC_ERROR	参阅SMC_ERROR	0	异常发生时，输出错误代码

(3)功能说明

本功能块为时间段和位置的轮廓运动模型，运行模式为DiscreteMotion，按用户在TimePosition变量中设定的数据运行。本功能块运行状态为Standstill中，指令运行时的状态为DiscreteMotion，其他状态无法运行。启动指令为Execute的上升沿启动，本指令在DiscreteMotion重复运行。TimePosition为MC_TP_REF数据类型。

MC_TP_REF具体描述如下:

成员	类型	初始值	描述
Number_of_pairs	INT	0	轮廓路径的段数
IsAbsolute	BOOL	TRUE	绝对运动 (TRUE) 和相对运动选择
MC_TP_Array	ARRAY[1..N] OF SMC_TP		时间和位置的数组

SMC_TP 具体描述如下 ：

成员	类型	初始值	描述
delta_time	TIME	TIME#0ms	位置段的时间
position	LREAL	0	当前的位置值

【注 ：按设置的位置数据对应的速度有变化时都按 S 曲线进行相关调整。】

◆ 时序图

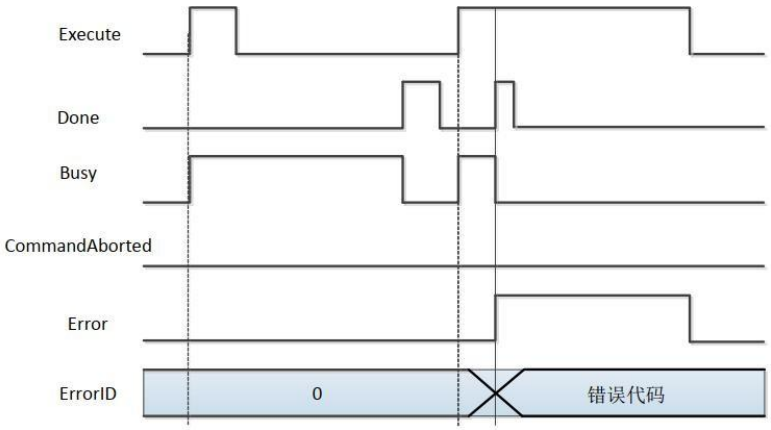
条件 MC_TP_Array 通过其他方式已经设置才能运行位置轮廓曲线指令；

轴必须处于 Standstill 状态指令才能运行；

功能块的 Execute 必须有上升沿的条件；

功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中。



(4)错误说明

错误的出现为轴状态不是在Standstill中启动指令或指令系统中的参数错误，出现轴错误只能清除错误后才开始运行。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.3.11 MC_Reset

通过复位所有与内部轴相关的错误，该功能块设计用于从状态错误停止到停止。 这不会影响功能块实例的输出。

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_Reset	轴错误状态复位指令		MC_Reset(Axis:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

(2)相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

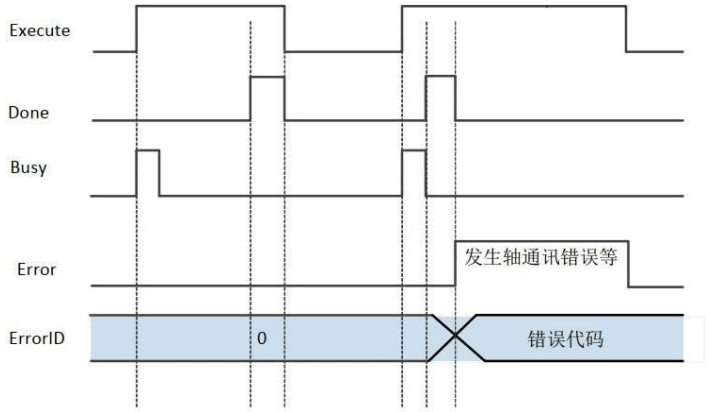
(3)功能说明

本功能块在轴通讯正常的情况下，把轴状态处于 errorstop 变为 Standstill，把轴的异常状态变为正常可运行的状态；

当出现轴 errorstop 无法复位，Axis.bCommunication 为 FLASE 状态 ， 必须要重新建立主站和从站轴的通讯；

在指令中的 Busy 标志位接通的时间非常短 ，使用时请注意。

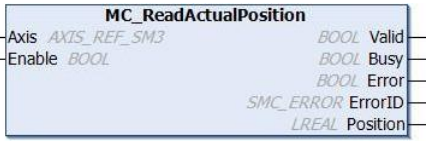
◆ 时序图



6.3.12 MC_ReadActualPosition

指令读取驱动器运行的实际位置，保存在自己定义的变量单元中。

(1) 指令格式

指令	名称	图形表现	ST 表 现
MC_ ReadActualP osition	实际位置读取指令		MC_ReadActualPosition(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Position=>);

（2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE,FALSE	FALSE	为 TRUE 状态读取伺服的当前位置

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	位置数据可获取标志	BOOL	TRUE,FALSE	FALSE	能正确的获取驱动器的位置，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参阅SMC_ ERROR	0	异常发生时，输出错误代码
Position	获取到的轴位置	LREAL	轴位置	0	指令读出来的轴位置数据

（3）功能说明

通过本指令读取驱动器中的实际位置指令，指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

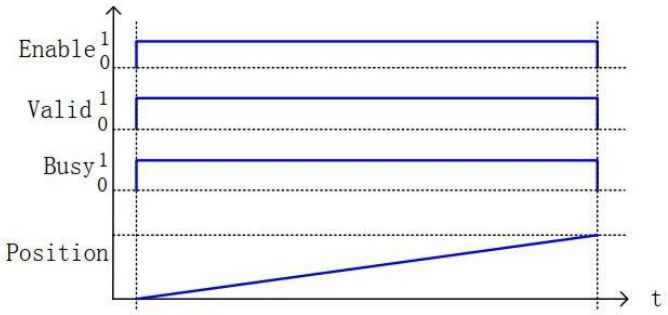
◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的 Position 为有效的数据值；

功能块的 Busy 表示当前功能块正在执行中。

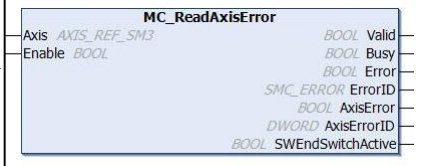
时序操作说明 ：



6.3.13 MC_ReadAxisError

指令读取轴的错误情况，保存在自己定义的变量单元中。

（1）指令格式

指令	名称	图形表现	ST 表 现
MC_ ReadAxisError	读轴的错误状态		MC_ReadAxisError(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , AxisError=> , AxisErrorID=> , SWEndSwitchActive=>);

（2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE FALSE	FALSE	为TRUE状态读取伺服的当前位置

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	错误数据可获取标志	BOOL	TRUE,FALSE	FALSE	能获取轴的错误数据，置为TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码
AxisError	轴错误标示	BOOL	TRUE,FALSE	FALSE	读出轴为错误，对应的标示位置
AxisErrorID	轴错误码	DWORD	0		读出轴为错误码
SWEnd SwitchActive	软限位开关有效	BOOL	TRUE,FALSE	FALSE	在指令读取中，检查软限位开关状态

（3）功能说明

通过 MC_ReadAxisError 读取驱动器中的错误码，指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的 AxisError 和 AxisErrorID 为有效的数据值；

功能块的 Busy 表示当前功能块正在执行中。

6.3.14 MC_ReadBoolParameter

指令读取驱动轴的位参数，保存在自己定义的变量单元中。

（1）指令格式

指令	名称	图形表现	ST 表 现
MC_ ReadBoolParamet er	读取轴的位参数		MC_ReadBoolParameter(Axis:= , Enable:= , ParameterNumber:= , Valid=> , Busy=> , Error=> , ErrorID=> , Value=>);

（2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE FALSE	FALSE	为 TRUE 状态读取伺服的当前位置
ParameterNumb er	轴参数的序号	DINT		0	访问轴参数的索引和子索引和序号

【注:ParameterNumber(DINT)=

-DWORD_TO_DINT(SHL(USINT_TO_DWORD(usiDataLength),24)(对象字典中的数据长度)

+SHL(UINT_TO_DWORD(uiIndex),8)(对象字典中的索引-16BIT)

+usisubIndex)(对象字典中的子索引-8BIT))usiDataLength:按字节数填写；1字节为16#01；2字节为16#02；4字节为16#04等。】

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	位置数据可获取标志	BOOL	TRUE,FALSE	FALSE	能正确的获取驱动器的位置，置为TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参阅SMC_ ERROR	0	异常发生时，输出错误代码
Value	获取到的轴位状态	BOOL	TRUE,FALSE	FALSE	指令读出来的轴位状态

（3）功能说明

通过 MC_ReadBoolParam 读取驱动器中的位数据状态，指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

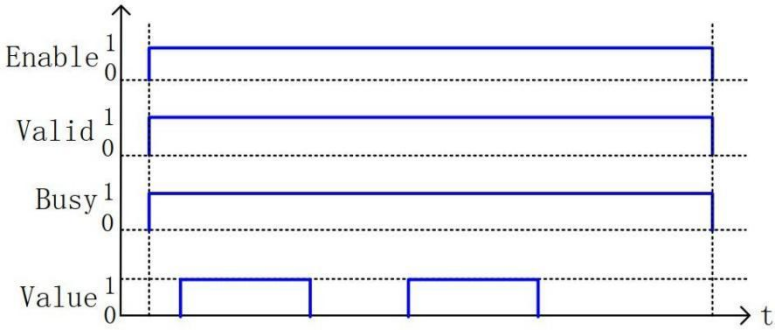
◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的 Valid 为有效的位状态数据；

功能块的 Busy 表示当前功能块正在执行中。

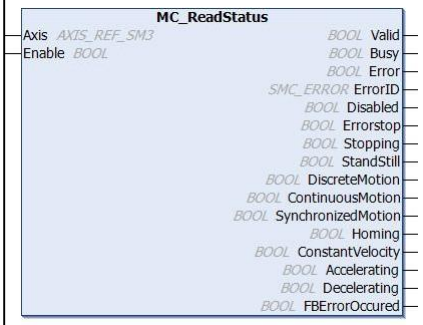
时序操作说明：



6.3.15 MC_ReadStatus

指令读取轴的状态数据，保存在自己定义的变量单元中。

（1） 指令格式

指令	名称	图形表现	ST 表 现
MC_ ReadStatus	读轴的状态		MC_ReadStatus(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Disabled=> , Errorstop=> , Stopping=> , StandStill=> , DiscreteMotion=> , ContinuousMotion=> , SynchronizedMotion=> , Homing=> , ConstantVelocity=> , Accelerating=> , Decelerating=> , FBErrorOccured=>);

（2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE,FALSE	FALSE	为TRUE状态读取伺服的当前位置

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	轴状态可获取标志	BOOL	TRUE,FALSE	FALSE	为 TRUE 时，代表轴状态可获取
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参 阅 SMC_ ERROR	0	异常发生时，输出错误代码
Disabled	轴没使能状态	BOOL	TRUE,FALSE	FALSE	轴在没使能状态为 TRUE；
Errorstop	轴错误状态	BOOL	TRUE,FALSE	FALSE	轴在错误运行状态为 TRUE；
Stoping	轴停止过程状态	BOOL	TRUE,FALSE	FALSE	轴在停止过程中为 TRUE
StandStill	轴标准状态	BOOL	TRUE,FALSE	FALSE	轴在标准（能运行）状态中为TRUE

Discrete Motion	轴离散运动状态	BOOL	TRUE,FALSE	FALSE	轴在离散运动状态中为 TRUE
Continuous Motion	轴连续运动状态	BOOL	TRUE,FALSE	FALSE	轴在连续运动状态中为 TRUE
Synchronized Motion	轴同步运行状态	BOOL	TRUE,FALSE	FALSE	轴在同步运动状态中为 TRUE
Homing	轴回原点状态	BOOL	TRUE,FALSE	FALSE	轴在回原点状态中为 TRUE
Constant Velocity	轴运行速度到达	BOOL	TRUE,FALSE	FALSE	轴达到运行速度中为 TRUE
Accelerating	轴加速过程状态	BOOL	TRUE,FALSE	FALSE	轴加速过程状态为 TRUE
Dccelerating	轴减速过程状态	BOOL	TRUE,FALSE	FALSE	轴减速过程状态为 TRUE
FBError Occured	轴功能块错误出现标志	BOOL	TRUE,FALSE	FALSE	轴功能块错误标志为 TRUE

（3）功能说明

通过 MC_ReadStatus 对应轴的各种状态，指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的后面个状态标志的各种数据；

功能块的 Busy 表示当前功能块正在执行中。

6.3.16 MC_ReadParameter

指令读取驱动轴的参数，保存在自己定义的变量单元中。

（1）指令格式

指令	名称	图形表现	ST 表 现
MC_ ReadParameter	读取轴的参数		MC_ReadParameter(Axis:= , Enable:= , ParameterNumber:= , Valid=> , Busy=> , Error=> , ErrorID=> , Value=>);

（2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE,FALSE	FALSE	为 TRUE 状态读取伺服的当前位置
Parameter Number	轴参数的序号	DINT		0	访问轴参数的索引和子索引和序号

【注:ParameterNumber(DINT)=

-DWORD_TO_DINT(SHL(USINT_TO_DWORD(usiDataLength),24)(对象字典中的数据长度)

+SHL(UINT_TO_DWORD(uiIndex),8)(对象字典中的索引-16BIT)

+usisubIndex)(对象字典中的子索引-8BIT)

usiDataLength：按字节数填写；1字节为16#01；2字节为16#02；4字节为16#04等。】

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	位置数据可获取标志	BOOL	TRUE,FALSE	FALSE	能正确的获取驱动器的位置，置为TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中， 置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码

Value	获取到的轴参数	LREAL		0	指令读出来的轴参数

（2）功能说明

通过 MC_ReadParam 读取驱动器中的位数据状态，指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

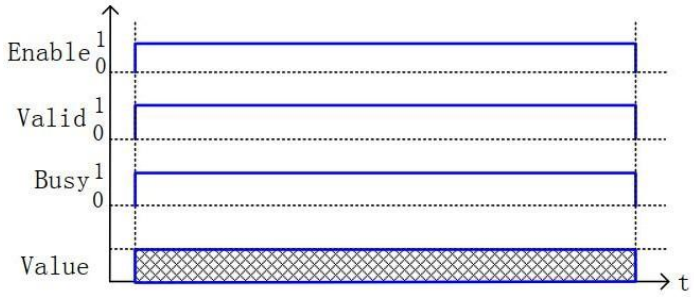
◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的 Valid 为有效的位状态数据；

功能块的 Busy 表示当前功能块正在执行中。

时序操作说明：



6.3.17 MC_AccelerationProfile

（1）指令格式

指令	名称	图形表现	ST 表 现
MC_ AccelerationProf ile	加速度轮廓指令		MC_AccelerationProfile(Axis:= , TimeAcceleration:= , Execute:= , ArraySize:= , AccelerationScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

（2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即AXIS_REF_SM3的一个实例
TimeAcceleration	轴加速时间和加速度描述	MC_TA_ REF			轴加速时间和加速度数据描述，加速数据由多组数据组成

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
ArraySize	动态数组	INT	数据范围	0	运行轮廓中使用的数组个数
AccelerationScale	综合因子	LREAL	“正数”“ 0”	1	MC_TA_REF 中加速度或减速度的比例因子
Offset	偏移	LREAL		0	加减速度的总体偏移值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE
Command Abort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
ErrorID	错误代码	SMC_ ERROR	参 阅 SMC_ ERROR	0	异常发生时，输出错误代码

（3）功能说明

本功能块为时间段和加减速度的轮廓运动模型，运行模式为DiscreteMotion，按用户在TimeAcceleration变量中设定的数据运行。本功能块运行状态为Standstill

中，指令运行时的状态为DiscreteMotion，其他状态无法运行。启动指令为Execute的上升沿启动，本指令在DiscreteMotion重复运行速度都是在上一次的叠加，容易引起系统故障。

TimeAcceleration为MC_TA_REF数据类型。

MC_TA_REF 具体描述如下：

成员	类型	初始值	描述
Number_of_pairs	INT	0	轮廓路径的段数
IsAbsolute	BOOL	TRUE	绝对运动 (TRUE) 和相对运动选择
MC_TA_Array	ARRAY[1..N] OF SMC_TA		时间和加速值的数组

SMC_TA 具体描述如下：

成员	类型	初始值	描述
delta_time	TIME	TIME#0ms	加速度段的时间
acceleration	LREAL	0	当前的加速度值

注:设置的加速度体现在速度的变化上,所有的加速度变化按S曲线的方式变化,从最终的结果变化为[起始加速度为A，终止加速度为B](A+B)/2的加速度数据体现在最终的速度上;

◆ 时序图

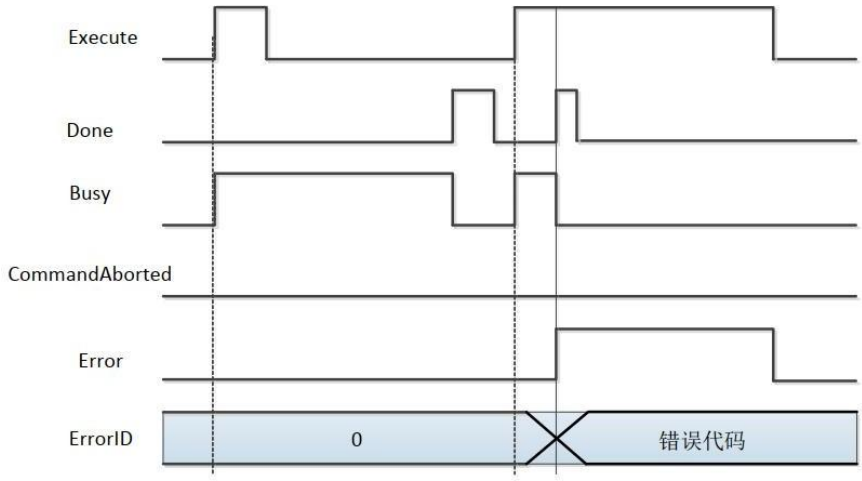
条件 MC_TA_Array 通过其他方式已经设置;

轴必须处于 Standstill 状态指令才能运行;

功能块的 Execute 必须有上升沿的条件;

功能块的 Done 表示指令正常执行完成;

功能块的 Busy 表示当前功能块正在执行中。



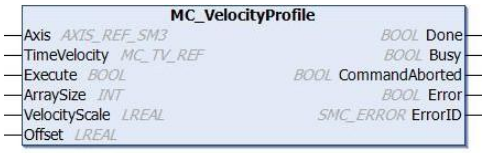
（4）错误说明

错误的出现为轴状态不是在 Standstill 中启动指令或指令系统中的参数错误，出现轴错误只能清除错误后才开始运行。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.3.18 MC_VelocityProfile

（1）指令格式

指令	名称	图形表现	ST 表 现
MC_ VelocityProfile	速度轮廓指令		MC_VelocityProfile(Axis:= , TimeVelocity:= , Execute:= , ArraySize:= , VelocityScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

（2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即AXIS_REF_SM3的一个实例
TimeVelocity	轴速度运行时间和速度描述	MC_TV_ REF			轴速度运行时间和速度数据描述，由多组数据组成

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
ArraySize	动态数组	INT	数据范围	0	运行轮廓中使用的数组个数
VelocityScale	速度因子	LREAL	“正数”，”0”	1	速度的比例因子
Offset	偏移	LREAL		0	速度值的总体偏移值

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Command Abort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参 阅 SMC_ ERROR	0	异常发生时，输出错误代码

(3)功能说明

本功能块为时间段和速度的轮廓运动模型，运行模式为 Continuous Motion，按用户在TimeVelocity 变量中设定的数据运行。

本功能块运行状态为 Standstill 中，指令运行时的状态为 Discrete Motion，其他状态无法运行。

启动指令为 Execute 的上升沿启动，本指令在 Discrete Motion 重复运行。

TimeVelocity为MC_TV_REF数据类型；

MC_TV_REF 具体描述如下：

成员	类型	初始值	描述
Number_of_pairs	INT	0	轮廓路径的段数
IsAbsolute	BOOL	TRUE	绝对运动 (TRUE) 和相对运动选择
MC_TV_Array	ARRAY[1..N] OF SMC_TV		时间和速度的数组

SMC_TV 具体描述如下：

成员	类型	初始值	描述
delta_time	TIME	TIME#0ms	速度值段的时间
Velocity	LREAL	0	当前记录的速度值

注：整个速度的过程为S曲线加减速的方式，每段轮廓都速度都为叠加的计算方式；在指令重复运行时，速度也为叠加方式，在指令使用时避免速度超限的出现；重复运行一定把本轴的状态重回到Standstill状态。

◆ 时序图

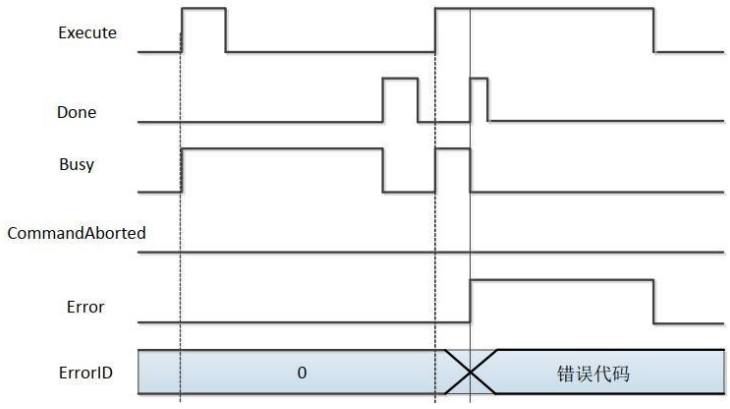
条件 MC_TV_Array 通过其他方式已经设置才能运行位置轮廓曲线指令；

轴必须处于 Standstill 状态指令才能运行；

功能块的 Execute 必须有上升沿的条件；

功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；



（4）错误说明

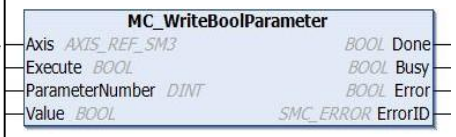
错误的出现为轴状态不是在 Standstill 中启动指令或指令系统中的参数错误，出现轴错误只能清除错误后才开始运行。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.3.19 MC_WriteBoolParameter

指令设置驱动轴的位参数。

1） 指令格式

指令	名称	图形表现	ST 表 现
MC_ WriteBoolParameter	设置轴的位参数		MC_WriteBoolParameter(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);

2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	为上升沿操作驱动一次设置操作
Parameter Number	轴参数的序号	DINT		0	访问轴参数的索引和子索引和序号
Value	设定值	BOOL	TRUE,FALSE	FALSE	设置位参数值

注：ParameterNumber (DINT) =

-DWORD_TO_DINT(SHL(USINT_TO_DWORD(usiDataLength), 24)(对象字典中的数据长度)

+ SHL(UINT_TO_DWORD(uiIndex), 8) (对象字典中的索引 -16BIT)

+ usisubIndex)(对象字典中的子索引-8BIT)

usiDataLength：按字节数填写；1字节为16#01；2字节为16#02；4字节为16#04等。

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

Done	设置操作成功	BOOL	TRUE,FALSE	FALSE	设置操作成功置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参 阅 SMC_ ERROR	0	异常发生时，输出错误代码

3）功能说明

通过 MC_ WriteBoolParameter 设置轴的位参数，指令为 Execute 上升沿触发。指令可以重复多次使用，互不影响。

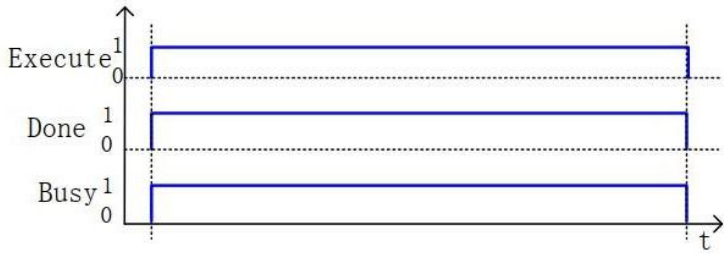
◆ 时序图

功能块的 Execute 必须为上升沿触发条件；

功能块的 Done 表示设置操作成功；

功能块的 Busy 表示当前功能块正在执行中。

◆ 时序操作说明：



6.3.20 MC_WriteParameter

指令写入驱动轴的参数，保存在自己定义的变量单元中。

1） 指令格式

指令	名称	图形表现	ST 表 现
MC_ WriteParameter	设置轴参数		MC_WriteParameter(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);

2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	为上升沿操作驱动一次设置操作
Parameter Number	轴参数的序号	DINT		0	访问轴参数的索引和子索引和序号
Value	设定值	LREAL			设置位参数值

注：ParameterNumber (DINT) =

-DWORD_TO_DINT(SHL(USINT_TO_DWORD(usiDataLength), 24)(对象字典中的数据长度)

+ SHL(UINT_TO_DWORD(uiIndex), 8) (对象字典中的索引 -16BIT)

+ usisubIndex)(对象字典中的子索引 -8BIT)

usiDataLength:按字节数填写；1字节为16#01；2字节为16#02；4字节为16#04等。

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	设置操作成功	BOOL	TRUE,FALSE	FALSE	设置操作成功置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE

ErrorID	错误代码	SMC_ ERROR	参 阅 SMC_ ERROR	0	异常发生时，输出错误代码

3）功能说明

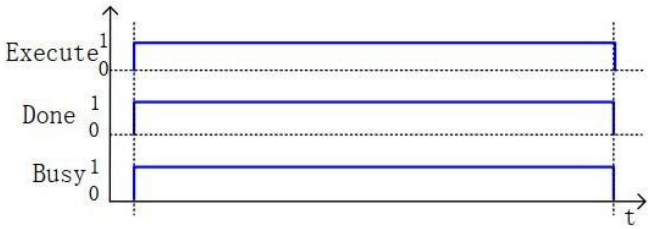
通过 MC_ WriteParameter 设置轴的位参数，指令为 Execute 上升沿触发。指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Execute 必须为上升沿触发条件；

功能块的 Done 表示设置操作成功；

功能块的 Busy 表示当前功能块正在执行中。



6.3.21 MC_AbortTrigger

功能块终止输入锁存相关事件的关联特性，和 MC_Touchprobe 配套使用。

1） 指令格式

指令	名称	图形表现	ST 表 现
MC_AbortTrigger	功能块终止事件关联		MC_AbortTrigger(Axis:= , TriggerInput:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例
TruggerInput	触发信号	TRIIGGER_REF	—	—	触发信号、触发属性等描述

◆ TRIIGGER_REF 说明：

结构体	元素	数据类型	初始值	描述
TRIIGGER_REF	iTrigger Number	INT	-1	在驱动器模式下，锁定功能中的哪一个。 0：探针 1 上升沿锁存 1：探针 1 下降沿锁存 2：探针 2 上升沿锁存 3：探针 2 下降沿锁存
	bFastLatching	BOOL	TRUE	指定锁存触发的式： TRUE：驱动器模式FALSE：控制器模式
	bInput	BOOL		bFastLatching=FLASE 时，由控制器 Input 信号触发
	bActive	BOOL		触发的有效信号

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	为上升沿操作驱动一次设置操作

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	设置操作成功	BOOL	TRUE,FALSE	FALSE	设置操作成功置为 TRUE

Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3）功能说明

通过 MC_AbortTrigger 功能块是把触发信号或属性和相关的触发指令终止其关联操作。

功能块的 Execute 必须为上升沿触发条件；

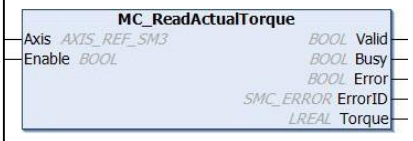
功能块的 Done 表示设置操作成功；

功能块的 Busy 表示当前功能块正在执行中。

6.3.22 MC_ReadActualTorque

指令读取驱动器运行的当前力矩值，读取的当前力矩值保存在自己定义的变量单元中。

1）指令格式

指令	名称	图形表现	ST 表 现
MC_ ReadActualTorque	当前力矩值读取指令		MC_ReadActualTorque0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Torque=>);

2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE, FALSE	FALSE	为 TRUE 状态读取伺服的当前位置

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	当前力矩值可获取标志	BOOL	TRUE,FALSE	FALSE	能正确的获取驱动器的力矩值，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参 阅 SMC_ERROR	0	异常发生时，输出错误代码
Torque	获取的当前力矩值	LREAL	力矩值（‰）	0	指令读出来的当前力矩数据

3）功能说明

通过 MC_ReadActualTorque 读取驱动器中的当前力矩值指令，指令为 Enable

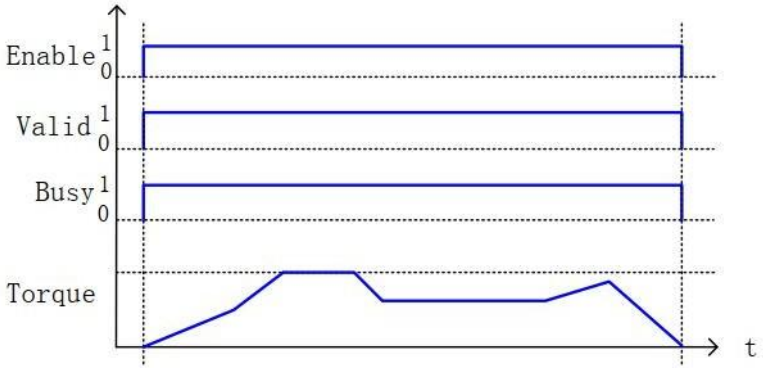
电平使能效应。指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的 Torque 为有效的数据值；

功能块的 Busy 表示当前功能块正在执行中。



6.3.23 MC_ReadActualVelocity

指令读取驱动器运行的当前速度值，读取的当前速度值保存在自己定义的变量单元中。

1） 指令格式

指令	名称	图形表现	ST 表 现
MC_ ReadActualVelocity	当前速度读取指令		MC_ReadActualVelocity0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Velocity=>);

2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行条件	BOOL	TRUE,FALSE	FALSE	为TRUE状态读取当前轴速度

◆ 输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	当前速度值可获取标志	BOOL	TRUE,FALSE	FALSE	能正确的获取驱动器的速度值，置为TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码
Velocity	获取的当前速度值	LREAL	速度值	0	指令读出来的当前速度数据

3）功能说明

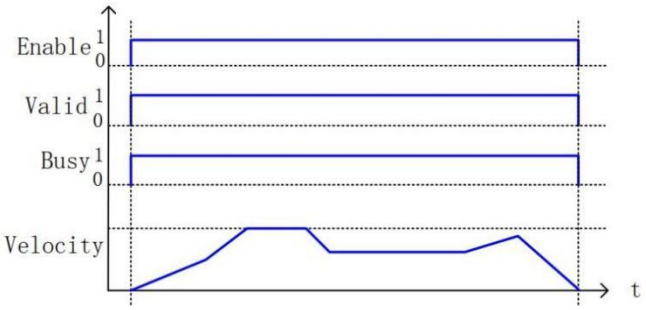
通过 MC_ReadActualVelocity 读取驱动器中的当前速度值指令，指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

◆ 时序图

功能块的 Enable 必须为 TRUE 的条件；

功能块的 Valid 表示读出的 Velocity 为有效的数据值；

功能块的 Busy 表示当前功能块正在执行中。



6.3.24 MC_SetPosition

将指令中的位置数据设为当前轴的位置数据，对设置位置数据操作不会产生任何位移移动，用于产生坐标系的位移。

1）指令格式

指令	名称	图形表现	ST 表 现
MC_SetPosition	读取轴的参数		MC_SetPosition0(Axis:= , Execute:= , Position:= , Mode:= , Done=> , Busy=> , Error=> , ErrorID=>);

2）相关变量

◆ 输入输出变量

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	—	—	映射到轴，即AXIS_REF_SM3的一个实例

◆ 输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	为上升沿操作驱动一次设置操作
Position	轴位置数据	LREAL		0	位置数据
Mode	设定值	BOOL	TRUE,FALSE	FALSE	位置模式；TRUE：相对位置(RELATIVE)；FALSE：绝对位置(ABSOLUTE)；

输出变量

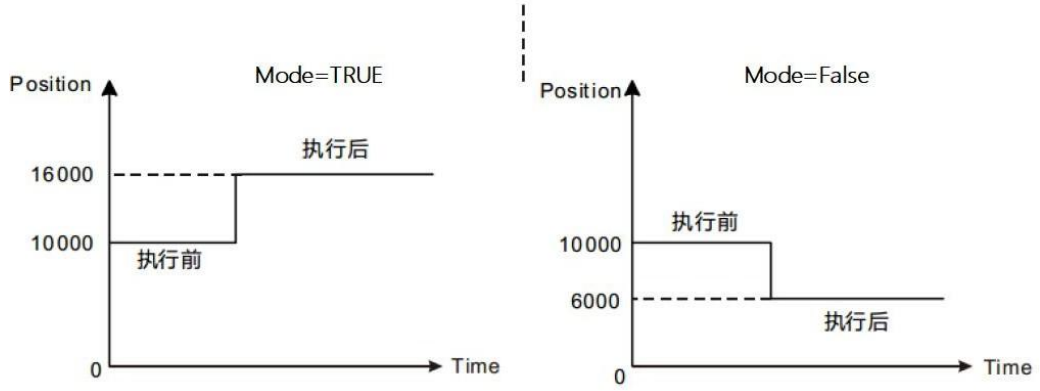
输出变量	名称	数据类型	有效范围	初始值	描述

Done	设置操作成功	BOOL	TRUE,FALSE	FALSE	设置操作成功置为TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
ErrorID	错误代码	SMC_ERROR	参阅SMC_ERROR	0	异常发生时，输出错误代码

功能说明

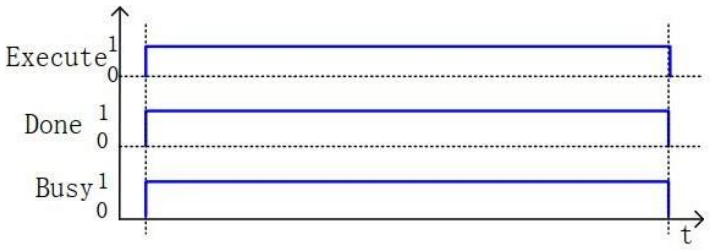
a、通过 MC_ SetPosition 设置轴的位置参数，不产生任何位移，但形成了坐标偏移； 指令为 Execute 上升沿触发；指令可以重复多次使用，互不影响。

b、与参考位置之间的关系。当 Mode=TRUE 时，Position 与参考位置为相对关系，位置设置值=参考位置+Position；当 Mode=FALSE 时，Position 与参考位置为绝对关系，位置设置值=Position。如下图所示，指令执行时的参考位置为 10000，输入参数 Position 的值为6000，当输入参数 Relative 为不同值时，对应的执行效果分别为左下图和右下图。



时序图

功能块的 Execute 必须为上升沿触发条件； 功能块的 Done 表示设置操作成功；功能块的 Busy 表示当前功能块正在执行中；



6.3.25 MC_TouchProbe

1）指令通过外部信号触发，保存当前轴的位置数据。 指令格式

指令	名称	图形表现	ST 表 现
MC_TouchProbe	启 用 外部锁定		MC_TouchProbe(Axis:= , TriggerInput:= , Execute:= , WindowOnly:= , FirstPosition:= , LastPosition:= , Done=> , Busy=> , Error=> , ErrorID=> , RecordedPosition=> , CommandAborted=>);

相关变量

输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例
TruggerInput	触发信号	TRIIGGER_REF	—	—	触发信号或触发属性等关联属性

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	为上升沿操作驱动一次设置操作
WindowOnl	触发窗口	BOOL	TRUE,FAL	FALSE	

y			SE		
FirstPosition	触发开始位置	LREAL	—	0	指定接收触发的开始位置
LastPosition	触发结束位置	LREAL	—	0	指定接收触发的结束位置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	设置操作成功	BOOL	TRUE,FALSE	FALSE	设置操作成功置为TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
ErrorID	错误代码	SMC_ERROR	参阅SMC_ERROR	0	异常发生时，输出错误代码
RecordedPosition	触发记录位置	LREAL	—	0	触发发生时当前的位置
CommandAborted	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE

功能说明探针功能说明

探针功能为了实现以传感器输入等发生触发信号为基点进行位置控制，发生触发信号时记录(锁定)轴位置。通常可同时为每个轴设定2点触发信号。

可使用MC_TouchProbe(启用外部锁定)指令，对需锁定的轴指定“触发输入条件”、“启用窗口”。触发信号除可指定连接伺服驱动器的信号以外，还可指定用户程序可使用的变量。需终止锁定功能时，应使用MC_AbortTrigger(禁用外部锁定)指令。

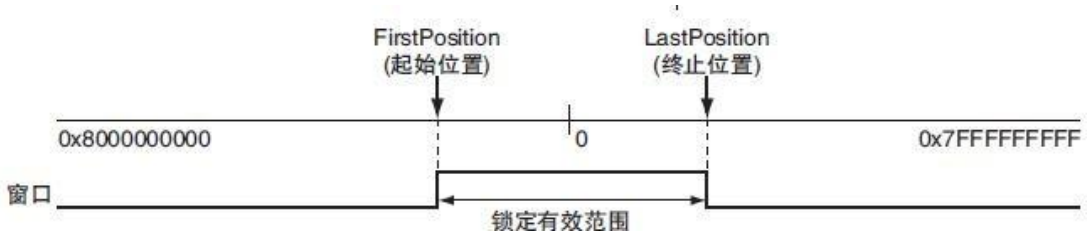
锁定功能可用于支持探针功能的伺服、编码器等。

使用WindowOnly(启用窗口)后，仅在起点和终点的范围内检测触发信号。不同计数模式的范围如下所示。

◆ 线性模式

仅 FirstPosition(起始位置) ≤ LastPosition(终止位置) 时可检测。

指定 FirstPosition(起始位置) ＞ LastPosition(终止位置) 时，指令会发生异常。



超过线性模式的位置范围进行指定时，指令会发生异常。

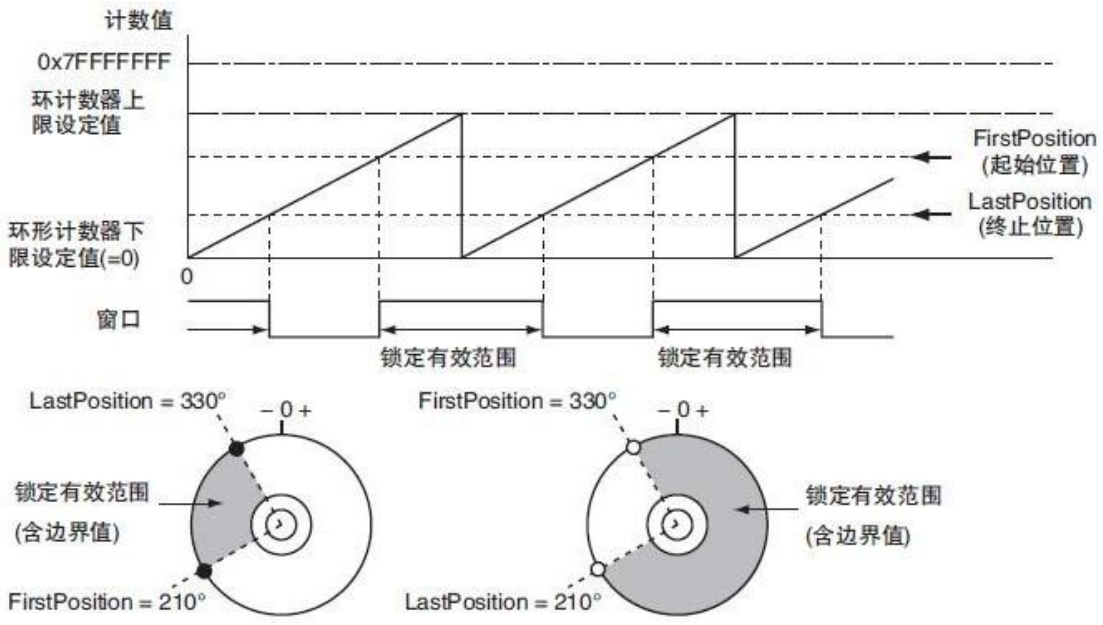
◆ 旋转模式

FirstPosition(起始位置)≤LastPosition(终止位置)和FirstPosition(起始位置)＞LastPosition(终止位置)两者均可指定。指定后者时，设定为跨越环形计数器下限设定值。

超过环形计数器上下限范围进行指定时，指令会发生异常。

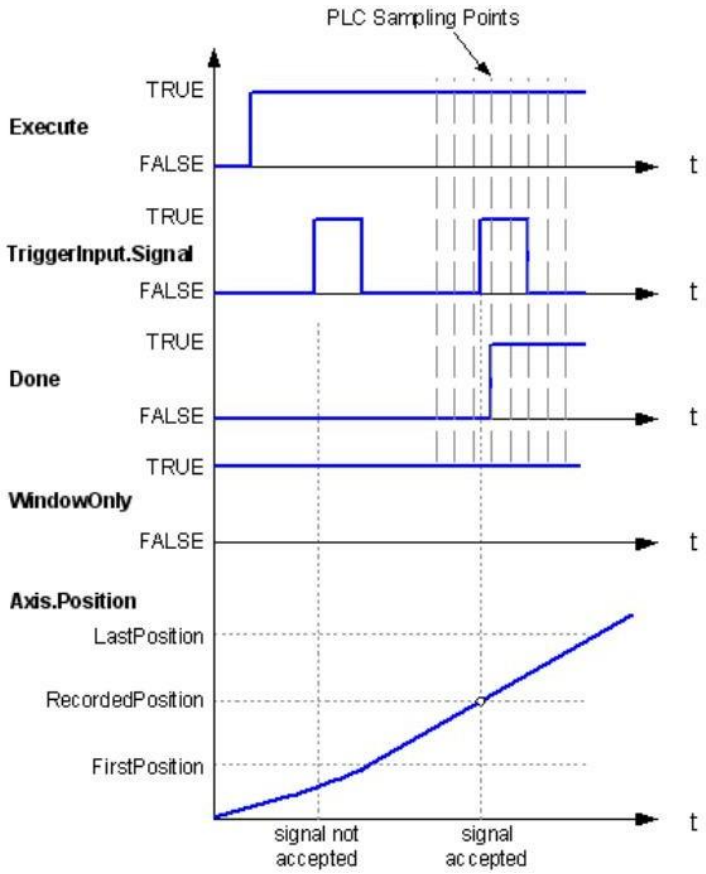
获取锁存位置有两种方法，下面分别介绍：

	First Position ≤ Last Position	First Position > Last Position
有效范围	FirstPosition ~ LastPosition	LastPosition ~ FirstPosition



- MC_TuochProbe 指令获取

通过 MC_TouchProbe 功能块 TruggerInput 的信号触发时记录运行轴的当前位置。Execute 上升沿执行执行，驱动器锁存时：驱动器采集到锁存信号会在记录位置，然后传送到控制器。



结构体数据类型 TRIIGGER_REF，描述了探针输入所使用的轴驱动器，决定了哪个探针编号与哪个硬件探针相对应。

成员名称	数据类型	初始值	描述
iTriggerNumber	INT	-1	触发通道； 由驱动程序定义（仅在 bFastLatching``=``TRUE"时使用）.
bFastLatching	BOOL	TRUE	TRUE：锁存在驱动器中完成，使用伺服轴 60B8 定义的探针功能（精确）。 FALSE：在运动任务循环中，bInput 锁存（不

			精确）。
bInput	BOOL		内部锁存信号，当 bFastLatching = FALSE 时触发信号有效。
bActive	BOOL	FALSE	探针状态，为 TRUE 时表示探针激活

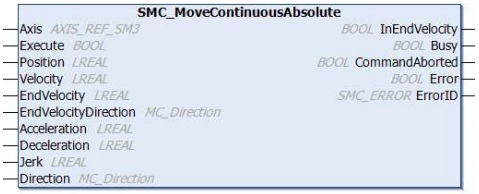
当bFastLatching:=TRUE时，以威科达总线伺服 VECServo 为例，iTriggerNumber 编号与伺服探针的关系如下：

iTriggerNumber	伺服探针的硬件 DI 以及边沿
0	伺服 DI9 上升沿锁存
1	伺服 DI9 下降沿锁存
2	伺服 DI10 上升沿锁存
3	伺服 DI10 下降沿锁存

6.3.26 SMC_MoveContinuousAbsolute

轴按绝对位置连续运行（单位按轴设置），绝对位置由 Position 指定，最后的运行速度EndVelocity 运行； 本指令运行前设置好相关的参数，加速度(Acceleration)、减速度(Deceleration)和运行速度 (Velocity)；对加速度(Acceleration)或减速度 (Deceleration)的赋值为 0 指令运行错误；在运行过程中，一定要关注本指令运行的完整过程，从用户程序的设计角度避免其他指令中断此指令。

(1)指令格式

指令	名称	图形表现	ST 表现
MC_MoveContinousAbsolute	轴绝对位置连续控制指令		

(2)相关变量

输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理

Position	运动绝对位置	LREAL		0	此数据为运动的绝对位置
Velocity	运行速度	LREAL		0	轴运行到目标位置的最大速度
EndVelocity	运行结束速度	LREAL		0	指令执行完成后的运行速度
EndVelocity Direction	结束速度的方向	MC_Direction	positive, negative, current;	Current	可以使用：positive, negative, current; 不可使用 ：shortest, fastest
Acceleration	加速度	LREAL		0	速度变大时加速度值
Deceleration	减速度	LREAL		0	速度变小时减速度值
Jerk	加速度变化率	LREAL		0	加加速度
Direction	运行方向	shortest		shortest	对于线性 / 直线轴：positive, negative; 对于旋转 / 圆周轴：positive, negative,current, shortest, fastest

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InEndVelocity	指令位置到达	BOOL	TRUE,FALSE	FALSE	轴指令执行位置到达，置为TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE

CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅SMC_ERROR	0	异常发生时,输出错误代码

(3)功能说明

本功能块为轴绝对定位指令 ，Distance 数据为轴的绝对位置。

本功能块运行状态为 Standstill 中 ，指令运行时的状态为 Discrete Motion，一个完整的运 行过程一定要控制轴的不同运动状态。

启动指令为 Execute 的上升沿启动 ，本指令在 Discrete Motion 可以重复上升沿有效 ，每 次都可以刷新最新的 Position 位置。

Acceleration 或 Deceleration 为零，指令运行都为异常状态，但轴的状态为 Discrete Motion;

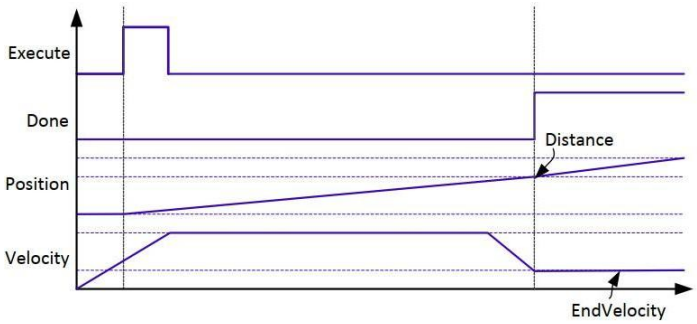
时序图

轴必须处于 Standstill 状态指令才能运行;

功能块的 Execute 必须有上升沿的条件;

功能块的 Done 表示指令正常执行完成;

功能块的 Busy 表示当前功能块正在执行中;



6.3.27 SMC_MoveContinuousRelative

轴按相对位置连续运行（单位按轴设置），相对位置由 Distance 指定，最后的运行速度EndVelocity 运行；本指令运行前设置好相关的参数，加速度(Acceleration)、减速度(Deceleration)和运行速度(Velocity)；对加速度(Acceleration)或减速度(Deceleration)的赋值为 0 指令运行错误；在运行过程中，一定要关注本指令运行的完整过程，从用户程序的设计角度避免其他指令中断此指令。

(1)指令格式

指令	名称	图形表现	ST 表现
MC_Move ContinuousRelative	轴相对定位指令		

(2)相关变量

输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行条件	BOOL	TRUE,FALSE	FALSE	输入的一个上升沿将启动功能块的处理
Distance	运动相对位置	LREAL	数据范围	0	此数据为运动的相对位置

Velocity	运行速度	LREAL	数据范围	0	轴运行到目标位置的最大速度
EndVelocity	运行结束速度	LREAL	数据范围	0	指令执行完成后的运行速度
EndVelocity Direction	结束速度的方向	MC_Direction	positive, negative, current;	Current	可以使用： positive, negative, current; 不可使用 ：shortest, fastest
Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InEndVelocity	指令位置到达	BOOL	TRUE,FALSE	FALSE	轴指令执行位置到达，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断， 置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

(3)功能说明

本功能块运行状态为 Standstill 中，指令运行时的状态为 Discrete Motion，在指令执行中关注本轴的运行状态，避免打断本轴的其他指令或被其他指令打断本轴的执行。

启动指令为 Execute 的上升沿启动，本指令在 Discrete Motion 可以重复上升沿有效，每次都可以刷新最新的 Position 位置。

Acceleration 或 Deceleration 为零，指令运行都为异常状态，但轴的状态为 Discrete Motion；

时序图

功能块的 Execute 必须有上升沿的条件；

功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；

6.3.28 MC_Jog

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_Jog	轴点动指令		MC_Jog(Axis:= , JogForward:= , JogBackward:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Busy=> , CommandAborted=> , Error=> , ErrorId=>);

(2)相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
JogForward	正向有效	BOOL	TRUE,FALSE	FALSE	设置为 TRUE 则开始正向移动；设置为 FALSE 则停止正向移动
JogBackward	负向有效	BOOL	TRUE,FALSE	FALSE	设置为 TRUE 则开始反向移动；设置为 FALSE 则停止反向移动
Velocity	目标速度	LREAL	正数或“0”	0	指定目标速度。单位：[指令单位 /s]
Acceleration	加速度	LREAL	正数或“0”	0	指定加速度。单位：[指令单位 /s2]
Deceleratio	减速度	LREAL	正数或“0”	0	指定减速度。单位：[指令单

n					位 /s2]
Jerk	加加速度	LREAL	正数或“0”	0	指令加速度变化率。单位：[指令单位 /s3]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行中	BOOL	TRUE,FALSE	FALSE	接收指令后，置为 TRUE
CommandAborted	执行中断	BOOL	TRUE,FALSE	FALSE	指令中止时，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

输入输出变量

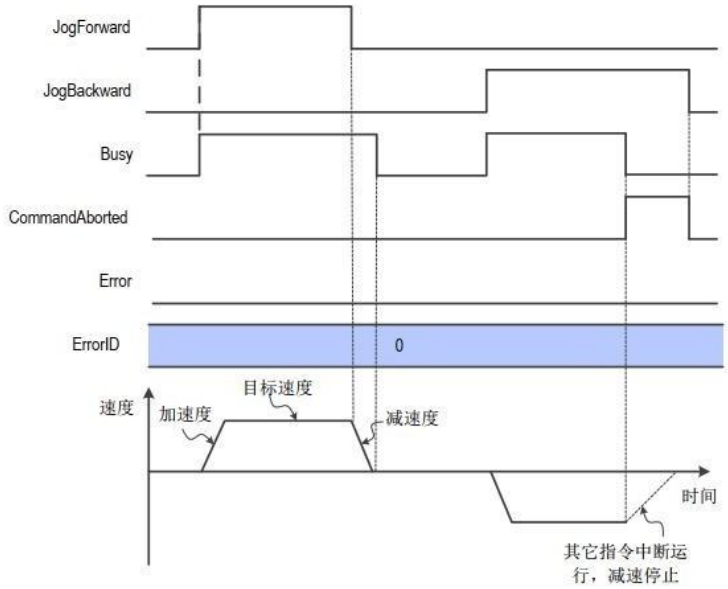
输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

(3)功能说明

根据指定 Velocity（目标速度）执行点动运行。

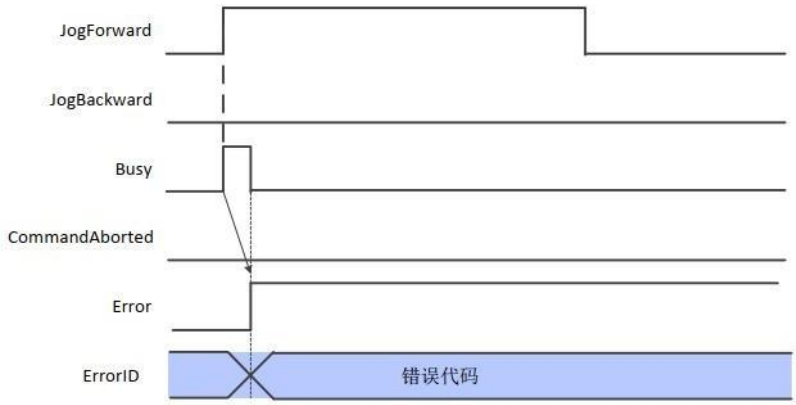
需要正向运行时，将 JogForward（正向运行有效）置为 TRUE；需要反向运行时，将JogBackward（负向运行有效）置为 TRUE。

同时将 JogForward(正向运行有效)和 JogBackward(负向运行有效)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。



(1)错误说明

在执行本指令中发生异常时， Error（错误）变成 TRUE，轴停止动作。可查看 ErrorID（错误代码）的输出值，了解发生异常的原因。



发生异常时的时序图

关于指令发生的异常，请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.3.29 SMC_Inch

轴单步运动控制，通过程序能实现一步接一步的单步控制。 (1)指令格式

指令	名称	图形表现	ST 表 现
SMC_Inch	轴相对定位指令		SMC_Inch0(Axis:= , InchForward:= , InchBackward:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Busy=> , CommandAborted=> , Error=> , ErrorId=>);

(2)相关变量

输入输出变量

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	—	—	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

InchForward	正向执行	BOOL	TRUE, FALSE	FALSE	如果 InchForward 为 TRUE，轴将按照给定的速度运动(Velocity, Acceleration,Deceleration) 以正向直到到达距离。输入必须被指定为 FALSE 然后再为 TRUE 再次启动运动。 如果 InchForward 在到达位置之前被设置为FALSE，那么轴将立即减速到 0 并且 Busy 将会被设置为 FALSE。 如果输入 InchBackward 在仿真情况下被设置 为 TRUE，那么将不会有运动产生。
InchBackward	反向执行	BOOL	TRUE, FALSE	FALSE	如果 InchBackward 为 TRUE，轴将会按照给定的速度值进行运动(Velocity, Acceleration, Deceleration) 以反向运动到设定位置。然后输入必须被设置为 FALSE，然后再设置为 TRUE 启动另一个运动。 如果输入信号 InchForward 同时被设置为 TRUE，那么将不会有轴运动。
Distance	移动的距离	LREAL	数据范围	0	此数据为运动的距离
Velocity	运行速度	LREAL	数据范围	0	轴运行到目标位置的最大速度

Acceleration	加速度	LREAL	数据范围	0	速度变大时加速度值
Deceleration	减速度	LREAL	数据范围	0	速度变小时减速度值

输出变量

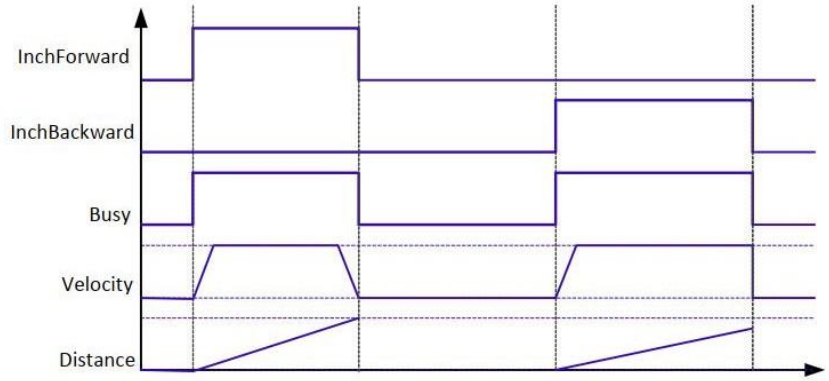
输出变量	名称	数据类型	有效范围	初始值	描述
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为TRUE
CommandAborted	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
ErrorID	错误代码	SMC_ERROR	参阅SMC_ERROR	0	异常发生时，输出错误代码

(3)功能说明

本功能块运行状态为 Standstill 中， 指令运行时的状态为 Discrete Motion，在指令执行中关注本轴的运行状态，避免打断本轴的其他指令或被其他指令打断本轴的执行。Acceleration 或 Deceleration 为零，指令运行都为异常状态，但轴的状态为 Discrete Motion；

时序图

功能块的 InchForward/ InchBackward 必须有 TRUE/FALSE 的条件；功能块的 Busy 表示当前功能块正在执行中；



6.3.30 SMC3_PersistPosition

该指令用来保持纪录实轴绝对值编码器的位置（断电重启控制器后，恢复断电前位置记录值）。如果伺服电机使用的是绝对值编码器，使用该功能块配合使用。

(1)指令格式

指令	名称	图形表现	ST 表 现
SMC3_ PersistPosition	轴 位 置 保持		SMC3_PersistPosition0(Axis:= , PersistentData:= , bEnable:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

(2)相关变量

输入输出变量

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	-	-	映 射 到 轴 ， 即 AXIS_REF_SM3 的一个实 例
PersistentData	保持数据	SMC3_PersistPosition_ Data			存储位置信息的断电保持 型数据结构

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	执行	BOOL	TRUE,FALSE	FALSE	True 功能块执行，false 不执行功能块，若要在 初始化期间还原上次存储的位置，则必须从应用程序启动时将该值置为 true

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bPositionRestored	位置恢复	BOOL	TRUE,FALSE	FALSE	TRUE，轴重启后位置恢复完成
bPositionStored	位置保存	BOOL	TRUE,FALSE	FALSE	TRUE，调用功能块后保存位置完成
bBusy	FB 执行中	BOOL	TRUE,FALSE	FALSE	TRUE，功能块没有执行完成
bError	错误	BOOL	TRUE,FALSE	FALSE	TRUE，异常发生
eErrorID	错误代码	SMC_ERROR	SMC_NO_ERROR	异常发生时，输出错误代码	
eRestoringDiag	恢复诊断	SMC3_PersistPositionDiag		SMC3_PersistPositionDiag.SMC3_PPD_RESTORING_OK	位置恢复中的诊断信息： SMC3_PPD_RESTORING_OK：位置成功恢复 SMC3_PPD_AXIS_PROP_CHANGED：轴参数有更改，无法恢复位置 SMC3_PPD_DATA_STORED_DURING_WRITING：功能块从轴参数数据结构复制

					数据，而不是从PersistentData数据中复制。可能原因：非同步性持续变量、控制器崩溃死机

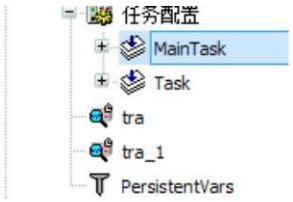
(3)功能说明

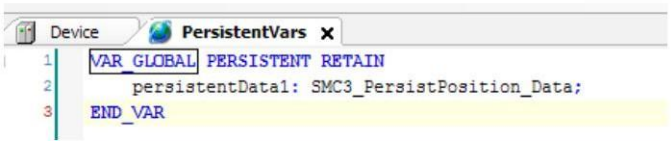
PLC 重启 bEnable 信号为 TRUE，则 bPositionRestroed 输出为 TRUE。不支持虚轴跟逻辑轴。

在此声明： VE 控制器 里面的轴实际位置为：偏移量 + 编码反馈位置（指令单位 Plus） * 标尺，绝对值编码器断电所记录的位置为指令单位值。所以 PLC 重启后要恢复断电前“实际位置”需使用该功能块，并且为了记录断电前轴 “实际位置”需将 SMC3_PersistPosition_Data 配成持续型变量”。

使用方法（实轴编码器为多圈绝对值时）：

④ PersistentVars 中声明 SMC3_PersistPosition_Data 型数据





⑤ PLC 主任务（EthCat 任务）中调用

申明部分： VAR

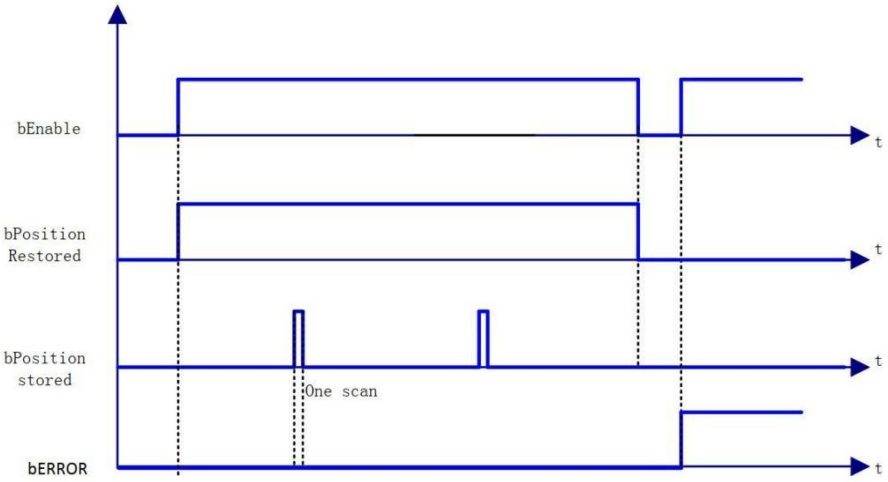
SMC3_PersistPosition_1:SMC3_PersistPosition;

END_VAR

```
//绝对值位置保存
SMC3_PersistPosition_1(Axis:=X_Axis , PersistentData:=persistentData1 ,bEnable:=TRUE );
```

程序部分：

时序图



(4)错误说明

输入轴为虚拟轴或者逻辑轴会导致错误输出； 轴有错误时会导致错误输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.3.31 SMC3_PersistPositionSingleturn

该指令用来保持纪录实轴绝对值编码器（单圈绝对值）的位置（断电重启控制器后，恢复断电前位置记录值）。

如果伺服电机使用的是单圈绝对值编码器，使用该功能块配合使用。 (1)指令格式

指令	名称	图形表现	ST 表现
SMC3_ PersistPosition Singleturn	轴位置保持		

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例
PersistentData	保持数据	SMC3_ PersistPosition Singletrun_Data			存储位置信息的断电保持型数据结构

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	执行	BOOL	TRUE,FALSE	FALSE	True 功能块执行， false 不执行功能块 若要在初始化期间还原上次存储的位置，则必须从应用程序启动时将

					该值置为 true
usiNumberofAbsoluteBitesusiNumberofAbsoluteBites	位数	UINT		16	多少位的绝对值编码器（如 20 位，24 位编码器等等）

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bPositionRestored	位置恢复	BOOL	TRUE,FALSE	FALSE	TRUE，轴重启后位置恢复完成
bPositionStored	位置保存	BOOL	TRUE,FALSE	FALSE	TRUE，调用功能快后保存位置完成

bBusy	FB 执行中	BOOL	TRUE,FALSE	FALSE	TRUE，功能块没有执行完成
bError	错误	BOOL	TRUE,FALSE	FALSE	TRUE，异常发生
eErrorID	错误代码	SMC_ERROR	SMC_NO_ERROR	异常发生时，输出错误代码	
eRestoringDi ag	恢复诊断	SMC3_PersistPositionDiag		SMC3_PersistPositionDiag.SMC3_PPD_RESTORING_OK	位置恢复中的诊断信息 SMC3_PPD_RESTORING_OK: 位置成功恢复 SMC3_PPD_AXIS_PROP_CHANGED: 轴参数有更改，无法恢复位置 SMC3_PPD_DATA_STORED_DURING_WRITING: 功能块从轴参数数据结构复制数据，而不是从PersistentData 数据中复制。可能原因：非同步性持续变量 、控制器崩溃死机

(3)功能说明

PLC 重启 bEnable 信号为 TRUE，则 bPositionRestroed 输出为 TRUE。不支

持虚轴跟逻辑轴。

PLC 重启后要恢复断电前“实际位置”需使用该功能块，并且为了记录断电前轴 “实际位置”需将 SMC3_PersistPositionSingleTurn_Data 配成持续型变量”。

使用方法（实轴编码器为多圈绝对值时）：

① PersistentVars 中声明 SMC3_PersistPositionSingleTurn_Data 型数据


```
VAR_GLOBAL PERSISTENT RETAIN
    persistentData2: SMC3_PersistPositionSingleTurn_Data;
END_VAR
```

② PLC 主任务（EthCat 任务）中调用

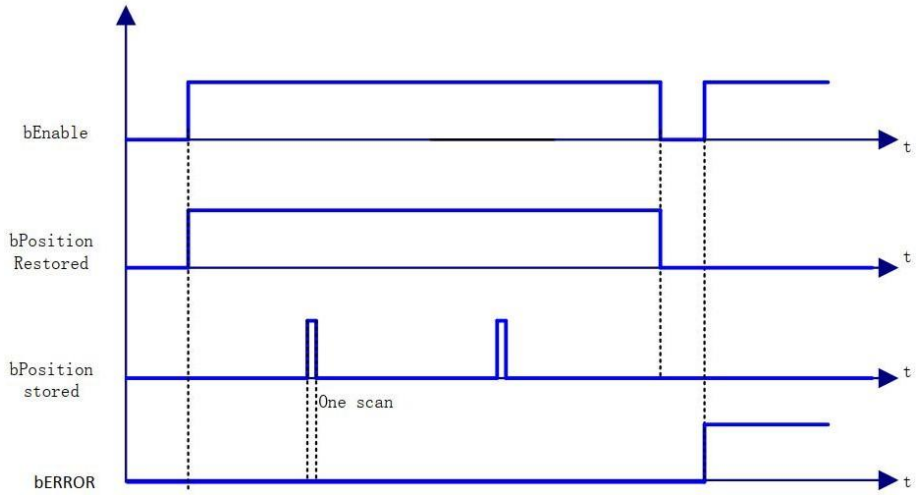
申明部分： VAR

SMC3_PersistPosition_2: SMC3_PersistPositionSingleTurn_Data; END_VAR

```
//绝对值位置保存
SMC3_PersistPosition_2(Axis:=Y_Axis , PersistentData:=persistentData2 ,bEnable:=TRUE );
```

程序部分：

(4)时序图



(5)错误说明

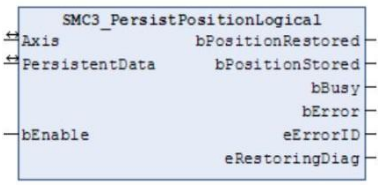
输入轴为虚拟轴或者逻辑轴会导致错误输出；轴有错误时会导致错误输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.3.32 SMC3_PersistPositionLogical

该指令用来保持记录逻辑轴（在实轴或者虚轴右键“添加设备”来选择添加逻辑轴）的位置（断电重启控制器后，恢复断电前位置记录值）。

(1)指令格式

指令	名称	图形表现	ST 表现
SMC3_PersistPositionLogical	轴位置保持		SMC3_PersistPositionLogical0(Axis:= , PersistentData:= , bEnable:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

（2）相关变量

输入输出变量

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_LOGICAL_SM3	-	-	映射到轴，即AXIS_REF_SM3的一个实例
PersistentData	保持数据	SMC3_PersistPositionLogical_Data			存储位置信息的断电保持型数据结构

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	执行	BOOL	TRUE,FALS	FALSE	True 功能块执行， false 不执行功能块

			E		若要在初始化期间还原上次存储的位置，则必须从应用程序启动时将该值置为 true

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bPositionRestored	位置恢复	BOOL	TRUE, FALSE	FALSE	TRUE，轴重启后位置恢复完成
bPositionStored	位置保存	BOOL	TRUE, FALSE	FALSE	TRUE，调用功能快后保存位置完成
bBusy	FB 执行中	BOOL	TRUE, FALSE	FALSE	TRUE，功能块没有执行完成
bError	错误	BOOL	TRUE, FALSE	FALSE	TRUE，异常发生
eErrorID	错误代码	SMC_ERROR	SMC_NO_	异常发生时，输出	

			ERROR	错误代码	
eRestoring Diag	恢复诊断	SMC3_PersistPosition Diag		SMC3_PersistPosition Diag.SMC3_PPD_RESTORING_OK	位置恢复中的诊断信息 SMC3_PPD_RESTORING_OK: 位置成功恢复 SMC3_PPD_AXIS_PROP_CHANGED: 轴参数有更改，无法恢复位置 SMC3_PPD_DATA_STORED_DURING_WRITING: 功能块从轴参数数据结构复制数据，而不是从PersistentData数据中复制。 可能原因：非同步性持续变量、控制器崩溃死机

（3）功能说明

PLC 重启 bEnable 信号为 TRUE，则 bPositionRestroed 输出为 TRUE。不支持虚轴跟实轴。

PLC 重启后要恢复断电前“位置”需使用该功能块，并且为了记录断电前轴 “实际位置”需将 SMC3_PersistPositionLogical_Data 配成持续型变量”。

使用方法（实轴编码器为多圈绝对值时）：

PersistentVars 中声明 SMC3_PersistPositionLogical_Data 型数据


```
VAR_GLOBAL PERSISTENT RETAIN
    persistentData3: SMC3_PersistPositionLogical_Data;
END_VAR
```

PLC 主任务（EthCat 任务）中调用

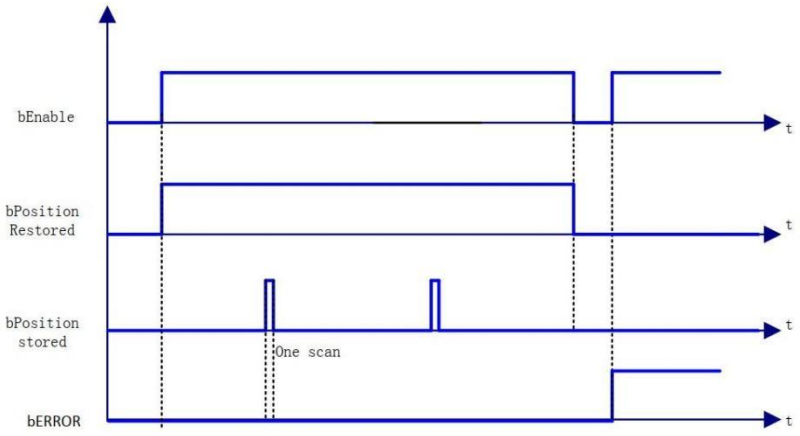
申明部分： VAR

SMC3_PersistPosition_3:SMC3_PersistPositionLogical; END_VAR

```
//绝对值位置保存
SMC3_PersistPosition_1(Axis:=X_Axis , PersistentData:=persistentData1 ,bEnable:=TRUE );
```

程序部分

时序图



(4)错误说明

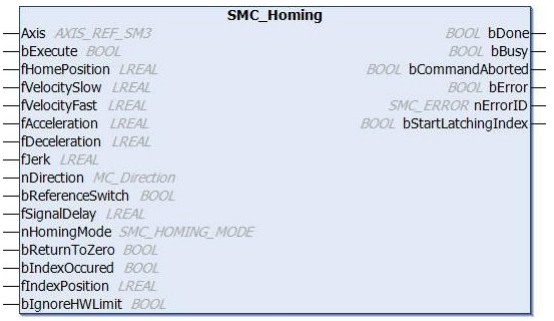
输入轴为虚拟轴或者实轴会导致错误输出；轴有错误时会导致错误输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.3.33 SMC_Homing

轴回零指令，与 MC_Homing 有区别， MC_Homing 在轴配置处设置回零方式，该指令为控制器控制的回零方式。

(1)指令格式

指令	名称	图形表现	ST 表 现
SMC_Homing	轴 回零		SMC_Homing0(Axis:= , bExecute:= , fHomePosition:= , fVelocitySlow:= , fVelocityFast:= , fAcceleration:= , fDeceleration:= , fJerk:= , nDirection:= , bReferenceSwitch:= , fSignalDelay:= , nHomingMode:= , bReturnToZero:= , bIndexOccured:= , fIndexPosition:= , bIgnoreHWLimit:= , bDone=> , bBusy=> , bCommandAborted=> , bError=> , nErrorID=> , bStartLatchingIndex=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE, FALSE	FALSE	True 功能块执行，false 不执行功能

					块
fHomePosition	原点设定位置	LREAL		0	回零后原点设定位置，单位为用户标定后的单位
fVelocitySlow	慢速	LREAL		0	离开参考开关后慢速设定速度
fVelocityFast	快速	LREAL		0	离开参考开关置位时，快速设定速度
fAcceleration	加速度	LREAL		0	加速度设定值
fDeceleration	减速度	LREAL		0	减速度设定值
fJerk	加速度导数	LREAL		0	Jerk in [u/s3]
nDirection	回零方向	MC_DIRECTION		negative	回零开始方向，参考MC_DIRECTION

bReference Switch	参考开关	BOOL	TRUE, FALSE	FALSE	连接参考开关，TRUE: 参考开关触发，FALSE: 参考开关闭合
fSignalDelay	延迟	LREAL		0	参考开关的传输时间，用来补偿死区时间。单位为秒。
nHomingMod	回零模式	SMC_HOMING_MODE			参考 SMC_HOMING_MODE
bReturnTozero	返回零位	BOOL	TRUE, FALSE	FALSE	TRUE: 回零完成后轴运行到位置零（注意：如果 fHomePosition=10，则回零完成后轴位置变为 10，bReturnTozero 为 ture 则回零完成后轴反向走 10 个单位到 0 位）
bIndexOccured		BOOL	TRUE, FALSE	FALSE	标志脉冲记录，回零模式为 FAST_BSLOW_I_S_STOP, FAST_SLOW_I_S_STOP 时生效
fIndexPosition		LREAL		0	标志脉冲时记录的位置
bIgnoreHWLimit	忽略硬限位	BOOL	TRUE, FALSE	FALSE	TRUE, 设定硬件限位开关使能为false，如果相同的物理开关用于硬件限位开关和参考开关，那么硬件控

					制将被设置为假

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone		BOOL	TRUE,FALSE	FALSE	True，回零完成
bBusy		BOOL	TRUE,FALSE	FALSE	True，功能块正在生效中
bCommandAborted		BOOL	TRUE,FALSE	FALSE	True，功能块被其他动作指令中断
Error		BOOL	TRUE,FALSE	FALSE	True，错误发生
ErrorID		SMC_ERROR		0	错误代码， 枚举型变量， 查看帮助 smc_error 查看具体报警代码
bStartLatching Index		BOOL	TRUE,FALSE	FALSE	由 “ bIndexOccured ” 和 “fIndexPosition”共同作用产生

回零模式（SMC_HOMING_MODE）

枚举名称	类型	初始值	描述
FAST_BSLOW_S_STOP	SMC_HOMING_MODE	0	按照设定方向以快速度走向原点开关，碰到原点开关后反向以慢速离开原点开关，离开后先执行 MC_setPosition 将当前位置设为 fHomePosition 设定值，然后执行 MC_stop

FAST_BSLOW_STOP_S	SMC_HOMING_MOD	1	按照设定方向以快速走向原点开关，碰到原点开关后反向以慢速离开原点开关，离开后先执行 MC_stop 将轴停止，然后执行 MC_setPosition，将当前位置设为 fHomePosition 设定值

FAST_BSLOW_I_S_STOP	SMC_HOMING_MOD	2	按照设定方向以快速走向原点开关,碰到原点开关后反向以慢速离开原点开关,bIndexOccured 信号到时先执行 MC_setPosition 再执行 MC_stop
FAST_SLOW_S_STOP	SMC_HOMING_ MOD	4	按照设定方向以快速走向原点开关,碰到原点开关后以慢速离开原点开关,离开后先执行 MC_setPosition 将当前位置设为 fHomePosition 设定值,然后执行 MC_stop
FAST_SLOW_STOP_S	SMC_HOMING_ MOD	5	按照设定方向以快速走向原点开关,碰到原点开关后以慢速离开原点开关, 离开后先执行 MC_stop , 然后执行MC_setPosition 将当前位置设为 fHomePosition 设定值。
FAST_SLOW_I_S_STOP	SMC_HOMING_ MOD	6	按照设定方向以快速走向原点开关,碰到原点开关后反向以慢速离开原点开关,bIndexOccured 信号到时先执行 MC_setPosition 再执行 MC_stop

(3)功能说明

SMC_HOMING 通过 bExecute 的上升沿启动之后，轴将会按照速度 fVelocityFast 并以 nDirection 定义的方向开始运动，直到 bReferenceSwitch = FALSE。然后轴将会缓慢停止并按照相反的方向以速度 fVelocitySlow 离开参考开关。 bReferenceSwitch = TRUE 后回零完成，使能回零指令后 bReferenceSwitch 的状态为 ON->OFF->ON，在 OFF->ON 的上升沿回零完成，设置参考位置。

参考位置 =fHomePosition+ （（ fSignalDelay*1000+1 个 DC 时钟周期）/1000 ）

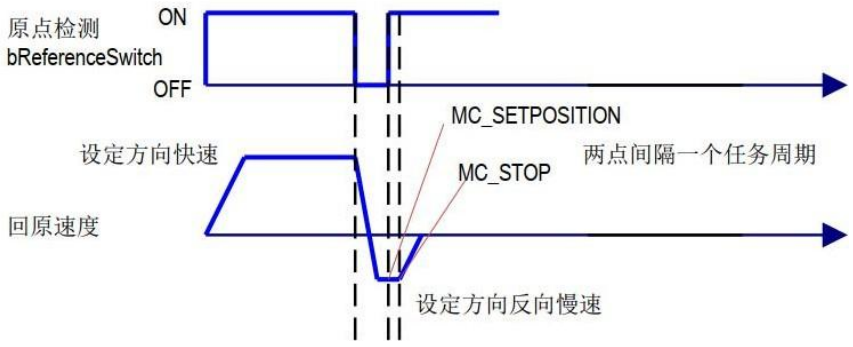
*fVelocitySlow 实际就是补偿了设置的 bReferenceSwitch 采样延迟和一个通讯周期位移延迟。

如果 bReturnToZero=TRUE， bReferenceSwitch 的状态在 OFF->ON 的上升沿

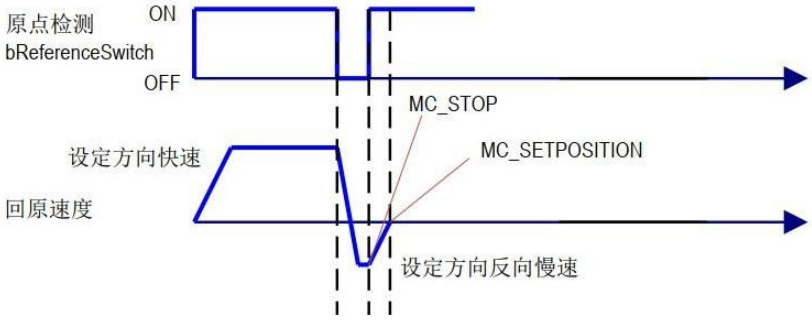
将参考位置设置为 fHomePosition+（（ fSignalDelay*1000+1 个 DC 时钟周期）/1000 ）*fVelocitySlow，然后按照速度fVelocityFast 运行到 0 位置。

注意： Done 完成信号后， 轴位置设定为： fHomePosition 。设定的时机跟 nHomingMode 有关（详情参考 SMC_HOMING_MODE）。下图为几种回零模式：

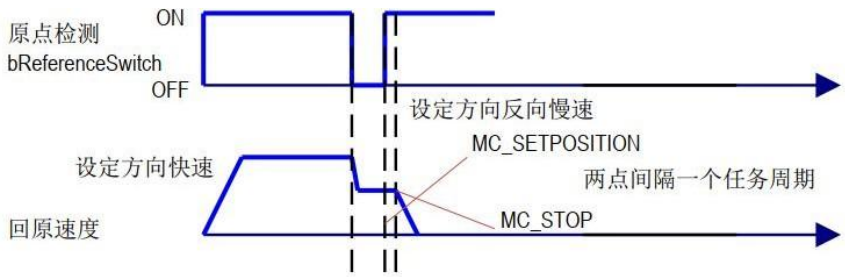
① 回零模式“0”时



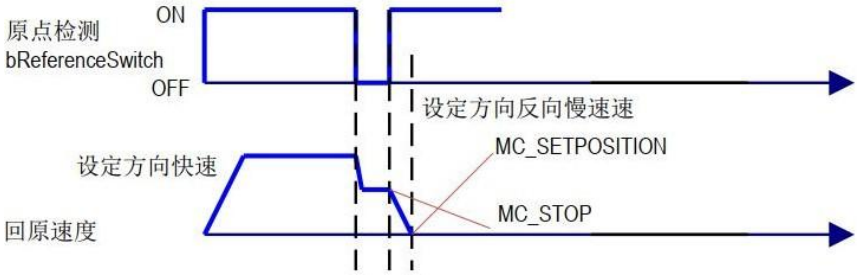
② 回零模式“1”时



③ 回零模式“4”时

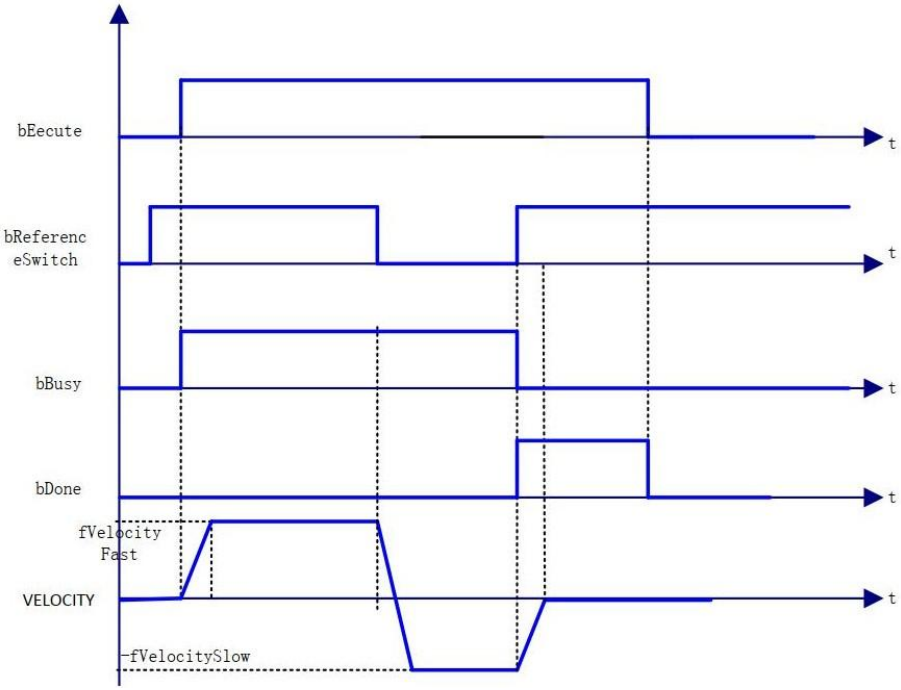


④ 回零模式“5”时

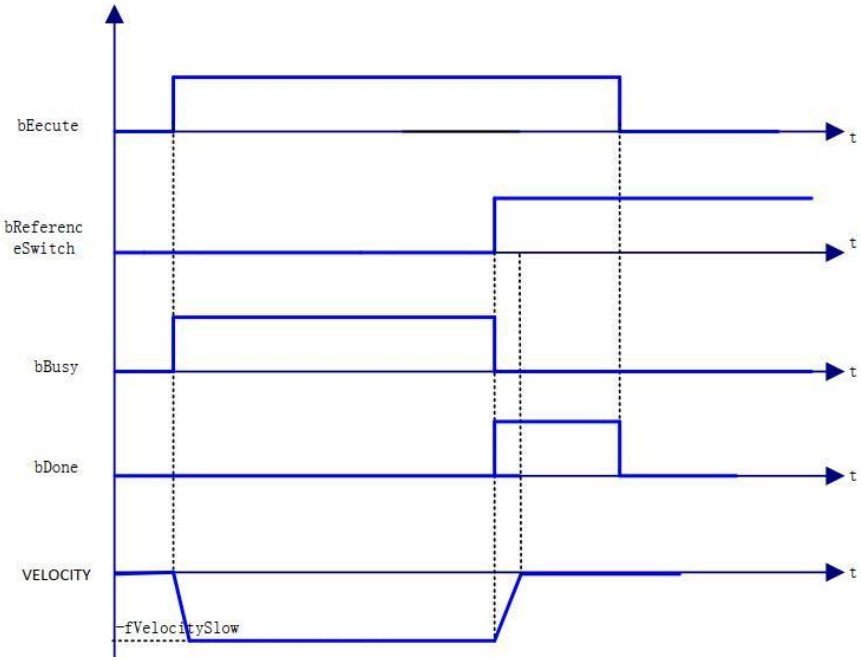


时序图

① 指令执行时 bReferenceSwitch TRUE 时



② 指令执行时 bReferenceSwitch FALSE 时



(4)错误说明

输入轴类型出错。

轴有错误。

轴没有使能

速度或加速度无效。

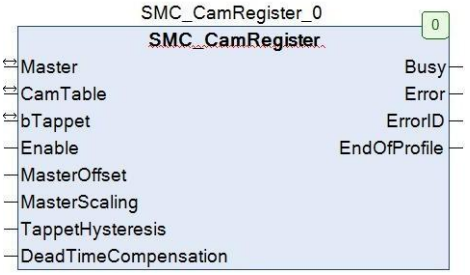
【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4 轴组指令（主/从轴指令）

6.4.1 SMC_CamRegister

实现凸轮挺杆控制（凸轮开关）。凸轮编辑的时候可以不编辑主从轴曲线，只需配置挺杆表就可通过该功能块实现挺杆控制。

(1)指令格式

指令	名称	图形表现	ST 表 现
SMC_CamRegister	凸轮挺杆控制		SMC_CamRegister0(Master:= , CamTable:= , bTappet:= , Enable:= , MasterOffset:=0 , MasterScaling:= 1, TappetHysteresis:= , DeadTimeCompensation:= , Busy=> , Error=> , ErrorID=> , EndOfProfile=>);

(2)相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例
CamTable	凸轮表	MC_CAM_REF			映射到一个电子凸轮，即一个电子凸轮实例

bTappet	挺杆输出	ARRAY [1..MAX_NUM_TAPPETS] OF BOOL			挺杆点的输出

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行	BOOL	TRUE,FALSE	FALSE	True 功能块执行， false 不执行功能块
Masteroffset	主轴偏移	LREAL		0	主轴偏移量
MasterScaling	主轴标尺	LREAL		1	主轴线性缩放因子
TappetHysteresis	挺杆阻尼	LREAL		0	挺杆控制阻尼系数
DeadTime Compensation	死区时间补偿	LREAL		0	死区补偿时间单位为 S，根据主轴当前速度线性补偿挺杆输出，可为正值，可为负值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行中	BOOL	TRUE,FALSE	FALSE	TRUE，功能块执行中
Error	错误	BOOL	TRUE,FALSE	FALSE	TRUE，异常发生
ErrorID	错误代码	SMC_ERROR		SMC_NO_ER	异常发生时，输出错误代

				ROR	码
EndofProfile	曲线周期完成	BOOL	TRUE,FALSE	FALSE	True，主轴位置大于等于设定周期

(3)功能说明

Enable 信号为 TRUE，如果没有错误输出则 Busy 输出为 TRUE，执行挺杆控制。

该控制功能块跟电子凸轮中的从轴没有关系，只需配置主轴周期与挺杆表。

“bTappet”为一维布尔型结构体（MAX_NUM_TAPPETS=512），且 bTappet[i] 对应第 i 个挺杆点的输出。

DeadTimeCompensation 单位为 S/ 秒，设置为正值时则超前输出挺杆信号，设置为负值时则滞后输出挺杆信号。比如说设置为 0.02 秒 Ethcat 任务周期设置为 4ms 则根据主轴线性速度 v，挺杆输出位置为 P，则挺杆在主轴设定位置=P-V*0.02 处输出挺杆值。反之如果设置为 -0.02 秒则主轴设定位置大于等于 P 后滞后五个周期挺杆信号输出

该功能块使用样例： 变量声明：

```
VAR
TPP:ARRAY[1..MAX_NUM_TAPPETS] OF BOOL;
SMC_CamRegister0: SMC_CamRegister;
END_VAR
```

程序部分：

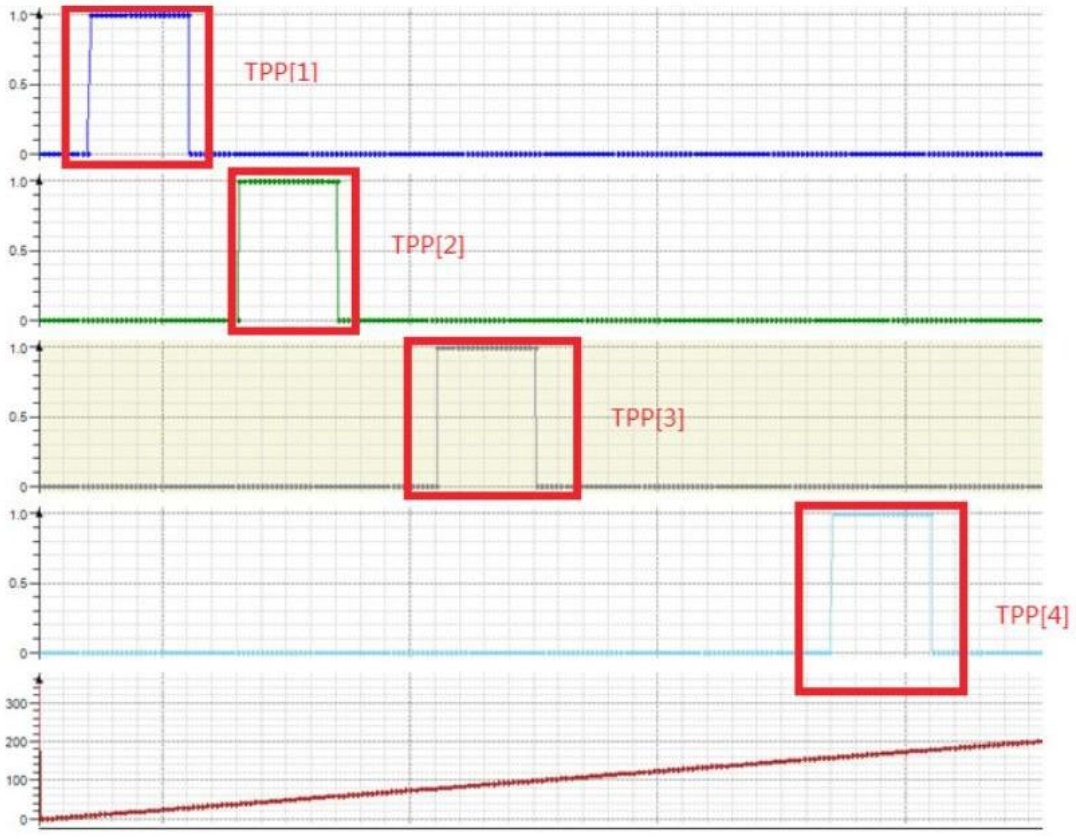
```
SMC_CamRegister0( Master:=Virtual_X , CamTable:=Cam, bTappet:=TPP ,
Enable:=TRUE , MasterOffset:=0 , MasterScaling:= 1,
TappetHysteresis:= 0, DeadTimeCompensation:=0 , Busy=> ,
Error=> , ErrorID=> , EndOfProfile=> );
```

Cam 编辑如下图：

cam | cam 表 | 挺杆 | 挺杆表

	Track ID	X	positive pass	negative pass
⊕	1			
🗑		10	打开	无
🗑		30	关闭	无
⊕	2			
🗑		40	打开	无
🗑		60	关闭	无
⊕	3			
🗑		80	打开	无
🗑		100	关闭	无
⊕	4			
🗑		160	打开	无
🗑		180	关闭	无
⊕				

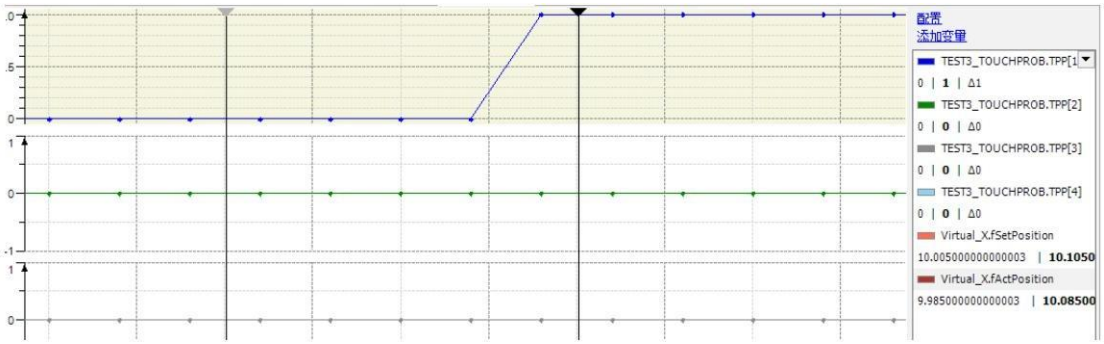
启动 Virtual_X 轴： 监控曲线如下图：



当死区补偿时间设置为 -0.02 秒时SMC_CamRegister0(Master:=Virtual_X , CamTable:=Cam, bTappet:=TPP , Enable:=TRUE , MasterOffset:=0 ,MasterScaling:= 1,

TappetHysteresis:= 0, DeadTimeCompensation:=-0.02 , Busy=> ,

Error=> , ErrorID=> , EndOfProfile=>);



挺杆输出滞后五个任务周期（任务周期为 4ms），如下图所示：

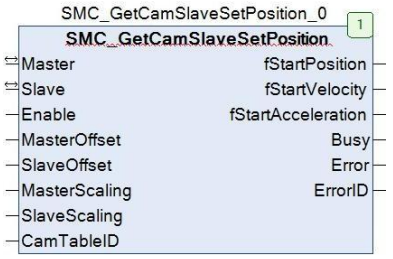
(4)错误说明

轴有错误、轴没有使能、偏移值或者标尺值设置超过主轴范围。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.2 SMC_GetCamSlaveSetPosition

读取凸轮表从轴位置、速度、加速度信息。(1)指令格式

指令	名称	图形表现	ST 表 现
SMC_GetCamSlaveSetPosition	获取凸轮从轴位置		SMC_GetCamSlaveSetPosition0(Master:= , Slave:= , Enable:= , MasterOffset:= , SlaveOffset:= , MasterScaling:= , SlaveScaling:= , CamTableID:= fStartPosition=> , fStartVelocity=> , fStartAcceleration=> , Busy=> , Error=> , ErrorID=>);

(2)相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF	-	-	映射到一个轴
Slave	从轴	AXIS_REF	-	-	映射到一个轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行	BOOL	TRUE,FALSE	FALSE	True 功能块执行，false 不执行功能块
Masteroffset	主轴偏移	LREAL		0	凸轮表主轴偏移
Slaveoffset	从轴偏移	LREAL		0	凸轮表从轴偏移
MasterScaling	主轴缩	LREAL		1	凸轮表主轴缩放因子

	放				
SlaveScaling	从轴缩放	LREAL		1	凸轮表从轴缩放因子
CamTableID	凸轮表 ID	MC_CAM_ID			凸轮表 ID

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
fStartPosition	从轴位置	LREAL		0	根据凸轮表跟当前主轴信息所获得的从轴位置
fStartVelocity	从轴速度	LREAL		0	根据凸轮表跟当前主轴信息所获得的从轴速度
fStart Acceleration	从轴加速度	LREAL		0	根据凸轮表跟当前主轴信息所获得的从轴加速度
busy	执行中	BOOL	TRUE, FALSE	FALSE	TRUE，表示功能块正在执行中

Error	错误	BOOL	TRUE, FALSE	FALSE	TRUE，异常发生
ErrorID	错误代码	SMC_ERROR		SMC_NO_ERROR	异常发生时，输出错误代码

(3)功能说明

该指令计算的输出值为： Y=（cam（(凸轮起始主轴查表位置 +Masteroffset)* MasterScaling）

+slaveoffset）* SlaveScaling，

Cam 为凸轮表函数。例如：凸轮起始主轴位置为 0，主从轴缩放为 1，Masteroffset 为100， slaveoffset 为 0，则功能块输出为凸轮表在 100 所对应的从轴位置。

该功能块读取从轴位置只需要凸轮表构建成功，对于主从轴是否运转并没有要求，示例。 声明：

SMC_GetCamSlaveSetPosition0: SMC_GetCamSlaveSetPosition; ENABLE: BOOL;

MC_CamTableSelect0: MC_CamTableSelect; 程序：

MC_CamTableSelect0(Master:=Virtual_X , Slave:=Virtual_Y , CamTable:=Cam , Execute:= , Periodic:=TRUE , MasterAbsolute:=0 , SlaveAbsolute:=0 , Done=> , Busy=> , Error=> , ErrorID=> , CamTableID=>);

SMC_GetCamSlaveSetPosition0(Master:= Virtual_X,

Slave:= Virtual_Y, Enable:=ENABLE , MasterOffset:= 100, SlaveOffset:=0 , MasterScaling:=1 , SlaveScaling:= 1,

CamTableID:=MC_CamTableSelect0.CamTableID, fStartPosition=> , fStartVelocity=> , fStartAcceleration=> , Busy=> , Error=> , ErrorID=>);

(4)错误说明

Error 输出为 True，则指令错误输出；

参考 ErrorID,SMC_ERROR 确定错误原因。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.3 SMC_GetTappetValue

与 MC_CamIn 指令配合使用，获取当前挺杆输出值。(1) 指令格式

指令	名称	图形表现	ST 表现
SMC_GetTappetValue	获取挺杆输出值		SMC_GetTappetValue0(Tappets:= iID:=, bInitValue:= , bSetInitValueAtReset:= , bTappet=>);

(2)相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Tappets	挺杆	SMC_TappetData	-	-	映射到一个挺杆

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
iID	挺杆组号	INT		0	挺杆的组 ID
bInitValue	初始值	BOOL			功能块第一次调用时挺杆初始化值

bSetInitValue AtReset	BOOL				TRUE,MC_CamIn 功能块重启时挺杆输出值将被初始化为bInitValue 设定值 FALSE, 挺杆输出值将会保持当 MC_CamIn 功能块重启时。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bTappet	挺杆输出	BOOL		FALSE	挺杆值

(3)功能说明

该功能块需要与 MC_CamIn 指令配合使用。

该功能块与 SMC_CamRegister 功能一样都是读取挺杆输出，但是两者存在冲突，所以在同一个凸轮挺杆表中只使用一种。使 用 示 例 ： MC_CamIn0(

Master:=Virtual_X , Slave:= Virtual_Y, Execute:= , MasterOffset:= 0,

SlaveOffset:= 0, MasterScaling:=1 , SlaveScaling:= 1,

StartMode:= 1,

CamTableID:= MC_CamTableSelect0.CamTableID, VelocityDiff:= ,

Acceleration:= , Deceleration:= , Jerk:= , TappetHysteresis:= , InSync=> ,

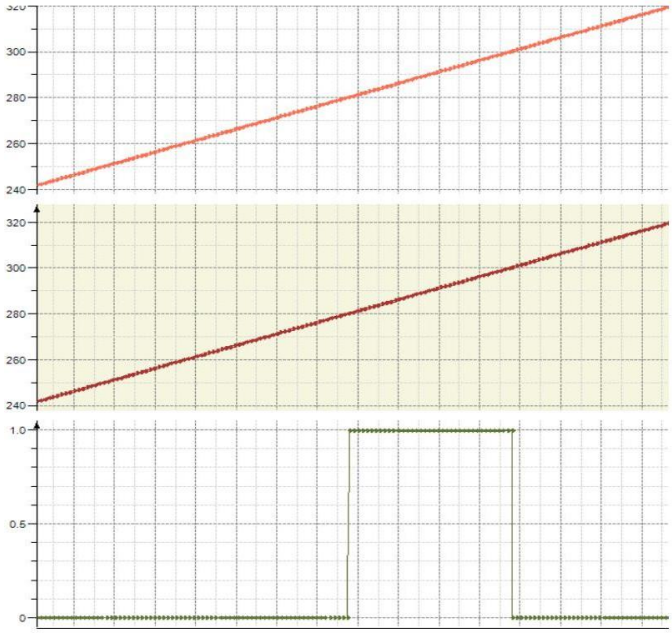
Busy=> , CommandAborted=> , Error=> ,

ErrorID=> , EndOfProfile=> , Tappets=>);

SMC_GetTappetValue0(

Tappets:= MC_CamIn0.Tappets, iID:=2,

bInitValue:= false, bSetInitValueAtReset:=true , bTappet=>);



(4) 错误说明

轴 有 错 误 ；

轴没有使能 ；

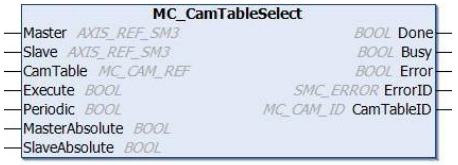
CamTable ID 没有指向。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.4 MC_CamTableSelect

凸轮表选择 MC_CamTableSelect 功能块，指定凸轮表，与 MC_CamIn 指令配合使用。该功能块用于关联主轴、从轴、凸轮表三者的关系，并设定凸轮运行的周期、主轴从轴的位置模式（绝对位置或相对位置）等，该指令为管理型指令，即触发该指令，只执行一次之后，　　相关主从轴就可以按该特性一直运行下去了；若需要更换凸轮表，或改变主从轴，就需要再触发执行该功能块一次。

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_ CamTableSelect	凸 轮 表 指 定		MC_CamTableSelect0(Master:= , Slave:= , CamTable:=), Execute:= , Periodic:= , MasterAbsolute:= , SlaveAbsolute:= , Done=> , Busy=> , Error=> , ErrorID=> , CamTableID=>);

(2)相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF	-	-	映 射 到 主 轴 ， 即 AXIS_REF_SM3 的一个实例
Slave	从轴	AXIS_REF	-	-	映 射 到 从 轴 ， 即 AXIS_REF_SM3 的一个实例
CamTable	选 择 表	MC_CAM_REF	-	-	映射 到 CAM 表格 描述 ， 即 MC_CAM_REF 的一个实例

使用注意事项：

主轴和从轴不能指定为同一轴，否则会有报错输出， CamTable 所对应的凸轮表编辑需要正确无误，否则也会导致指令报错。主轴、从轴可以为实轴也可以是虚轴。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行	BOOL	TRUE,FALSE	FALSE	上升沿信号，执行指令
Periodic	重复模式	BOOL	TRUE,FALSE	FALSE	指定是反复执行指定的凸轮表还是只执行一次 TRUE: 重 复 False: 不重复
MasterAbsolute	主轴绝对模式	BOOL	TRUE,FALSE	FALSE	指定主轴跟踪距离坐标系是按绝对位置还是相对位置1：绝对位置， 0:相对位置
SlaveAbsolute	从轴绝对模式	BOOL	TRUE,FALSE	FALSE	与 MC_CamIn 指 令中 的StartMode 综合指定从轴当前指令位置为凸轮表输出的绝对（当前主轴位置对应的凸轮表输出值)还是相对（凸轮表输出值叠加指令开始时从轴位置）1：绝对位置， 0: 相对位置

使用注意事项：

MasterAbsolute、 SlaveAbsolute 选择不当可能会导致电子凸轮输出跳变，所以设定前请确定设定凸轮曲线工作方式。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE,FALSE	FALSE	选择完成时为 TRUE,
Busy	执行中	BOOL	TRUE,FALSE	FALSE	选择中没有完成时为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码
CamTableID	生效CAMID	MC_CAM_ID	-	-	选择生效的 Cam_ID ，与MC_CamIn 指令中的 CamTableID 配合使用

使用注意事项:

Error 发生时请对照 ErrorID 查看帮助里面 SMC_ERROR

(3)功能说明

本指令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）。

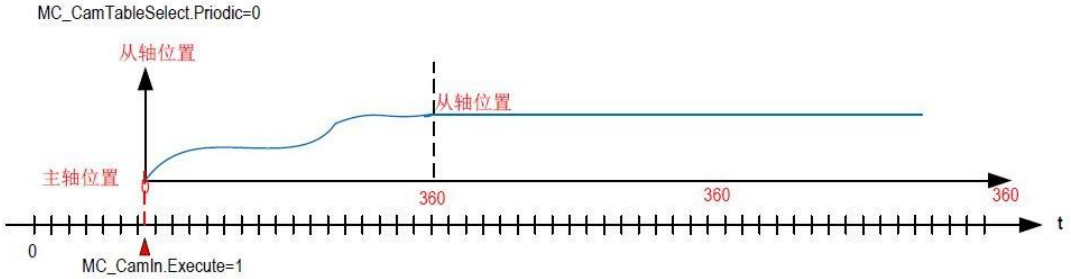
Excute 上升沿，执行指定凸轮表，亦可凸轮表更新后刷新指定凸轮表。

Done 信号输出为 TRUE 时，则输出变量“CamTableID”生成并且生效。

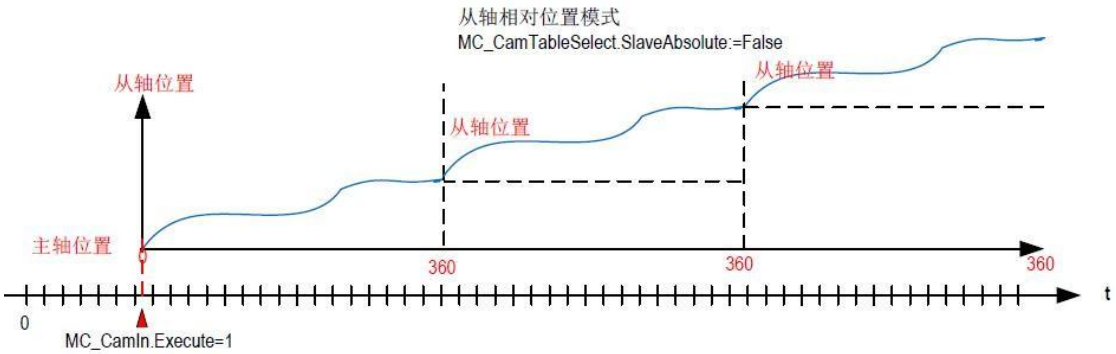
指令执行中，Busy 信号输出 TRUE，Done 信号 TRUE、Busy 信号为 FALSE。

Periodic 参数

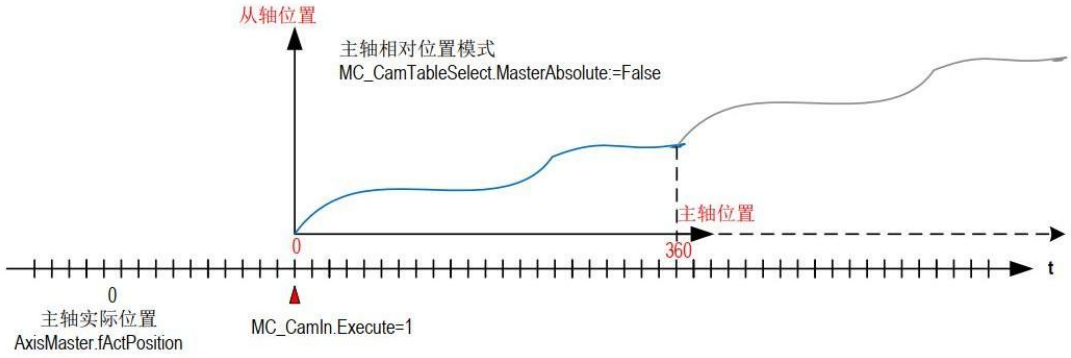
下图为单周期凸轮运行的效果。当凸轮表选择为单周期模式（Periodic:=0），在运行完一个凸轮表周期后，从轴即脱离凸轮运行状态。



当凸轮表选择为周期模式（Periodic:=1），在运行完一个凸轮表周期后，从轴又开始下一凸轮周期的运行，直到有用户程序命令其退出凸轮运行状态，如下图：



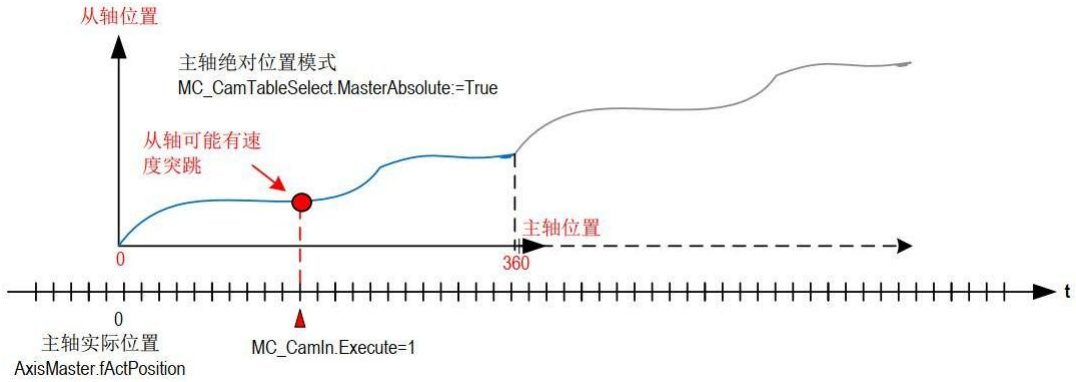
主轴和从轴都为相对位置模式时的运行特点



当主轴为相对位置模式，在进入 CAM 时，凸轮运算模块会以当前位置作为主轴的起点 X=0，进行运算；

当从轴为相对位置模式，在进入 CAM 时，凸轮运算模块会以当前位置作为从轴的起点 Y0 进行运算，此后的 CAM 输出结果在此基础上进行叠加。

主轴为绝对位置模式，从轴为相对位置模式

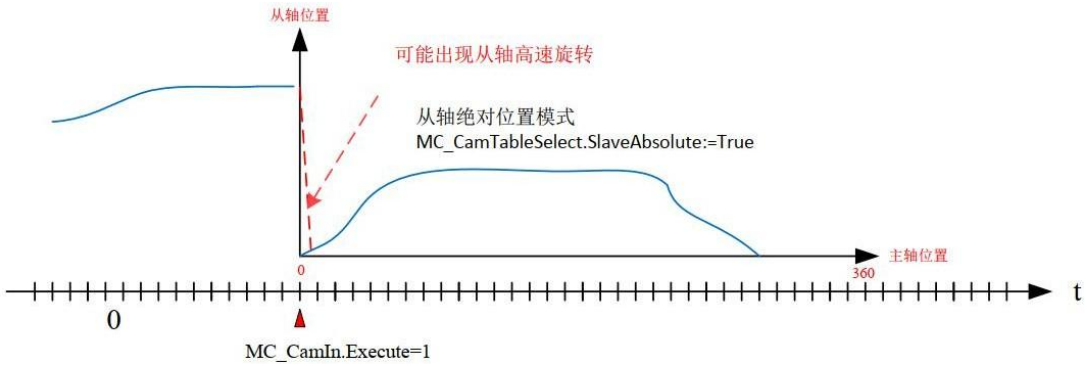


当主轴为绝对位置模式，在进入 CAM 时，凸轮运算模块会以当前主轴位置运算得到从轴位置，因此：

进入 CAM 运行时从轴位置调整时的高速转动，导致设备冲击或损坏； 若当前位置超出 CAM 表有效范围，从轴不会运动，并会告警；

若 CAM 表为周期模式，当本周期执行完毕，即开始下一 CAM 周期的连续运行。

主轴为相对对位置模式， 从轴为绝对位置模式



当从轴为绝对位置模式，进入 CAM 运行时会调整到 CAM 要求的位置，若偏差比较大， 会出现高速运动的自动调整。

根据应用特点采取对策：

对于必需对准操作的设备，如定长切割的旋转刀作为凸轮从轴必需采用绝对位置，编程时注意在旋转刀第一次旋切动作之前，要进行旋转刀的归零位操作；

合理设置凸轮表的主轴位置范围，在下一周期开始时，避免凸轮出现反方向的位置调整；运行 SMC_GetCamSlaveSetPosition，将凸轮进入点的从轴位置设为从轴的当前坐标。对于可以采用相对位置模式的应用，尽量采用相对位置模式：MC_CamTableSelect.SlaveAbsolute:=False; 或设 MC_CamIn.StartMode:=1;（相对模式） 注意：

当从轴设为“有限长”的绝对模式时，在作回零调整时，若可以左转回零，也可以右转回零，控制器会选择一个比较近的方向回零。设计凸轮表的范围时，尤其要注意不要让凸轮表的范围超过了实际需要运行的范围，否则就可以出现伺服从轴的瞬间高速回转调整，形成机械冲击。

(4)错误说明

主轴和从轴不能指定为同一轴，否则会有报错输出。

CamTable 所对应的凸轮表编辑需要正确无误否则会错误输出。

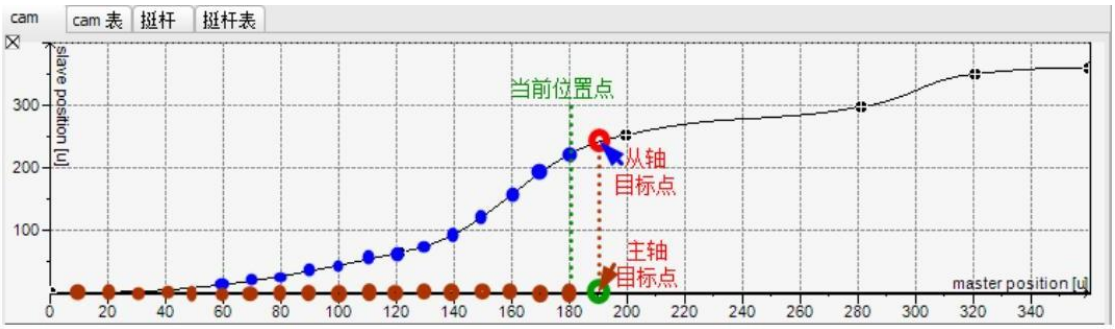
【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.5 MC_CamIn

使凸轮从轴进入与凸轮主轴同步运行状态，根据主轴当前位置、凸轮表的位置关系，控制凸轮从轴调整到对应的目标点，该指令的执行对主轴没有任何影响。可根据应用需求指定主从轴偏置值、缩放比和工作模式。

一旦 MC_CamIn 被触发，从轴按照凸轮表的位置对应关系，跟随主轴的位置运动，注意是位置对应，而不是速度对应。

进入凸轮运行后，系统每次 EtherCAT 中断都会解析 CAM 凸轮表，根据主轴



当前位置计算从轴的下一个目标点，然后将下一目标位置发送给从轴，令其运行。

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_CamIn	凸 轮 动 作开始	MC_CamIn Master AXIS_REF_SM3 — BOOL InSync Slave AXIS_REF_SM3 — BOOL Busy Execute BOOL — BOOL CommandAborted MasterOffset LREAL — BOOL Error SlaveOffset LREAL — SMC_ERROR ErrorID MasterScaling LREAL — BOOL EndOfProfile SlaveScaling LREAL — SMC_TappetData Tappets StartMode MC_StartMode CamTableID MC_CAM_ID VelocityDiff LREAL Acceleration LREAL Deceleration LREAL Jerk LREAL TappetHysteresis LREAL	MC_CamIn0(Master:= , Slave:= , Execute:= , MasterOffset:= , SlaveOffset:= , MasterScaling:= , SlaveScaling:= , StartMode:= , CamTableID:= , VelocityDiff:= , Acceleration:= , Deceleration:= , Jerk:= , TappetHysteresis:= , InSync=> , Busy=> , CommandAborted=> , Error=> , ErrorID=> , EndOfProfile=> , Tappets=>);

(2)相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例
Slave	从轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

使用注意事项：

主轴和从轴不能指定为同一轴，否则会有报错输出。

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行凸轮功进入能块	BOOL	TRUE, FALSE	FALSE	上升沿，执行电子凸轮

MasterOffset	主轴偏置	LREAL	负数，正数，0	0	以指定的偏移值移动主轴的相位
SlaveOffset	从轴偏置	LREAL	负数，正数，0	0	以指定的偏移值移动从轴的相位
MasterScaling	主轴预编移比例	LREAL	>0.0	1	以指定的比例放大/缩小主轴的相位
SlaveScaling	从轴预编移比例	LREAL	>0.0	1	以指定的比例放大/缩小从轴轴的相位
StartMode	从轴相对凸轮输出模式	MC_StartMode		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	表格编号	MC_CAM_ID			定义 cam 表格的使用，与 MC_CamTableSelect 的输出点 CamTableID 配合使用
VelocityDiff		LREAL			与 ramp_in 不同的最大速度
Acceleration		LREAL			ramp_in 时加速度
Deceleration		LREAL			ramp_in 时减速度
Jerk		LREAL			ramp_in 的加加速度

Tappet Hysteresis		LREAL			挺杆的阻尼系数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InSync	凸轮生效	BOOL	TRUE, FALSE	FALSE	主轴和从轴建立凸轮关系后，InSync 被置位，当指令的执行条件 OFF 时，InSync 被复位。
Busy	同步运行中	BOOL	TRUE, FALSE	FALSE	Execute 输入上升沿时，置位 TRUE， TRUE 时表示凸轮关系耦合中，需用 Cam_out 指令复位，指令执行条件复位不能复位该状态。
Command Aborted	指令中断	BOOL	TRUE, FALSE	FALSE	从轴被其他控制指令中断输出为 TRUE
Error	错误	BOOL	TRUE, FALSE	FALSE	如果检测到有错误， Error 位被置位；当指令的执行条件 OFF 时， Error 位被复位。
ErrorID	错误代码	SMC_ ERROR	参阅 SMC_ ERROR	0	异常发生时，输出错误代码
EndOf Profile	曲线完成	BOOL		FALSE	如果 MC_CamTableSelect 指令执行时Periodic 参数为 0（非周期），当凸轮曲线执行完一次后，EndOfProfile 位被置位，当指令的执行条件 OFF 时，EndOfProfile 位被复位。

Tappets	SMC_TappetData				关联一个凸轮挺杆，可被MC_GetTappetValue指令读出

(3)功能说明

Execute 上升沿、轴没报错、选择凸轮表正确则本指令启动；

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 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 CommandAborted 输出为TRUE。

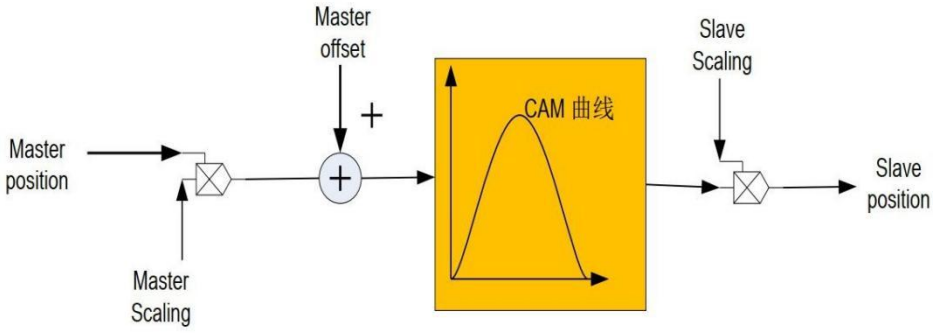
(4)指令详情

下面对指令详细说明：

指令启动条件

在主轴停止中、位置控制中、速度控制中、同步控制中任意状态下都可以启动本指令 注意：凸轮从轴位置设定值要在软件限位值以内，否则会导致错误输出指令。

凸轮曲线中齿合点的计算



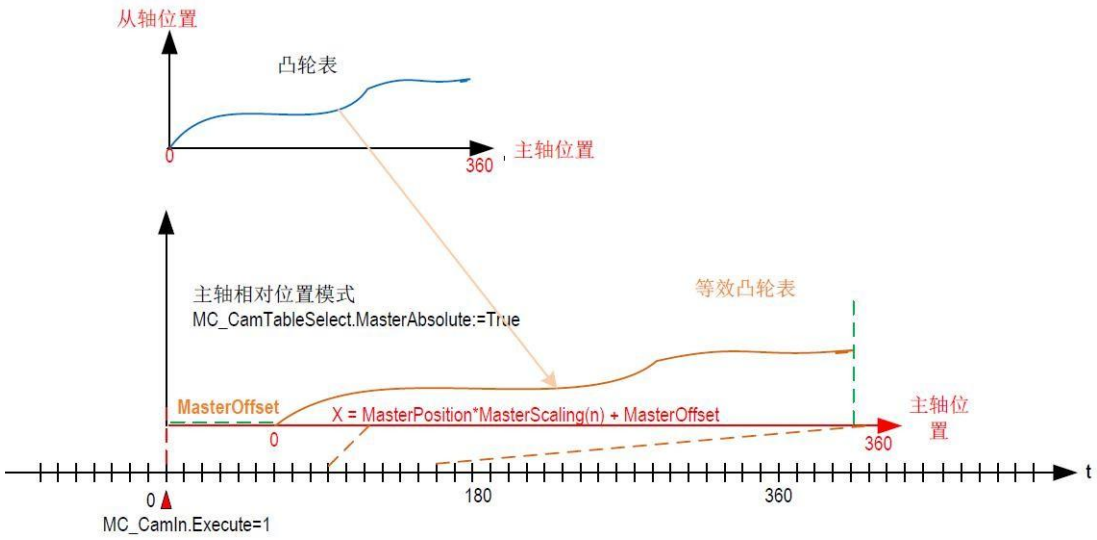
由上图得出计算方法如下:

Position_Slave = SlaveScaling*CAM(MasterScaling*MasterPosition + MasterOffset) +SlaveOffset

该公式中的主轴位置、从轴位置并不是代表实际物理轴的位置，而是凸轮函数曲线相关的主从轴位置。主从轴位置跟主从实轴位置之间的关系有详细描述。

凸轮主轴 MasterScaling 计算

默认情况下，系统对 MasterScaling=1，若用户程序修改了该变量，则：



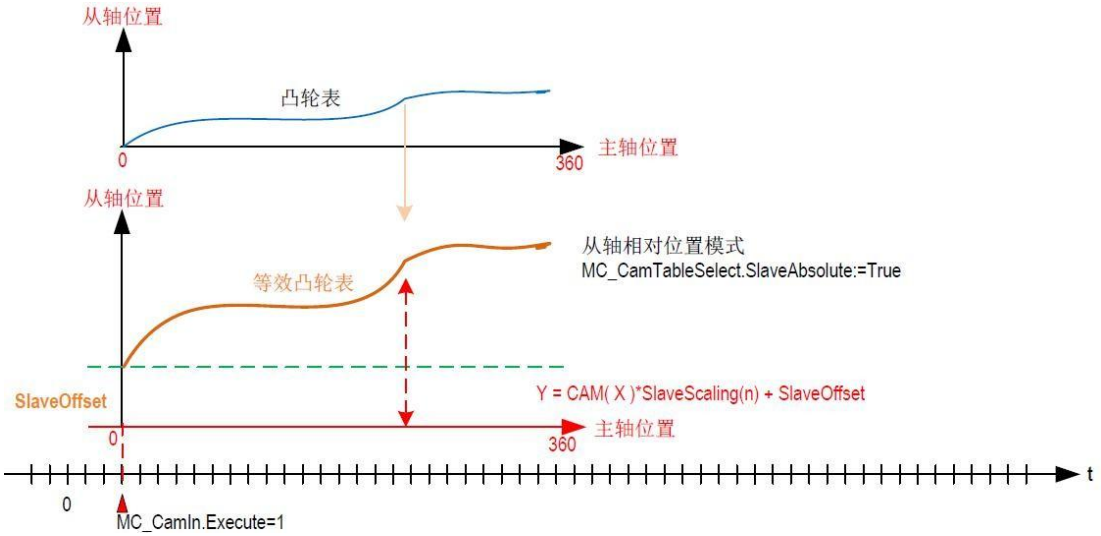
给凸轮主轴设置比例 SCALE 值，可以对主轴的位置进行线性缩放，使之与凸轮表的对应位置关系符合所期望的要求。

若考虑了主轴的偏移设定值，在凸轮表中主轴（X）的计算位置将是：

X = MasterPosition*MasterScaling(n) + MasterOffset

该参数可用于设备加工工件的尺寸微调。

凸轮从轴 SlaveScaling 计算



默认情况下，系统对 SlaveScaling=1，若用户程序修改了该变量，则：

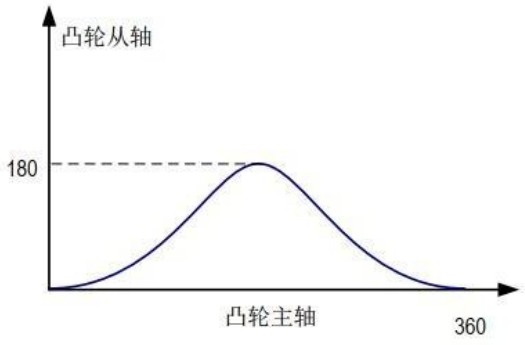
给凸轮从轴设置比例 SlaveScaling 值，可以对从轴的位置进行线性缩放，使凸轮控制的输出符合所期望的从轴运动位置要求。

若考虑了从轴的偏移设定值，凸轮从轴（Y）的输出位置是：

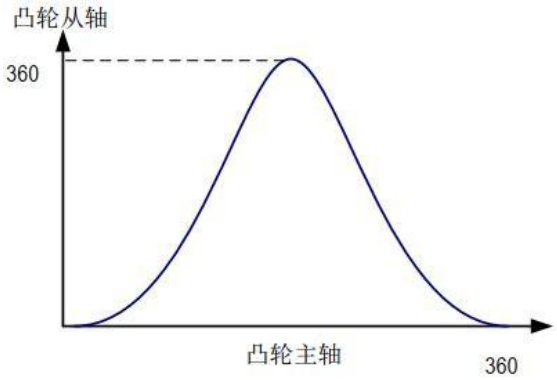
Y = CAM(X)*SlaveScaling(n) + SlaveOffset

该参数可用于设备加工工件的尺寸微调。用法举例：

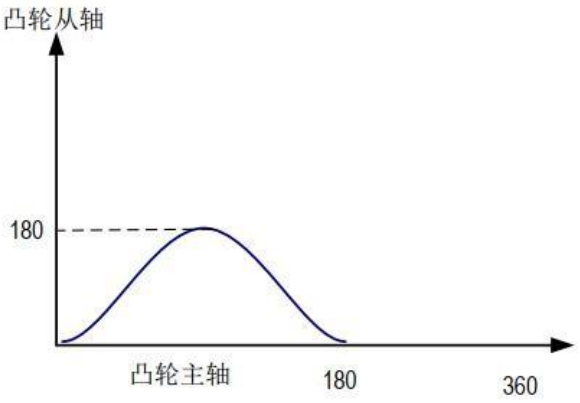
当 MasterScaling=1.0、 SlaveScaling=1.0、 MasterOffset=0、 SlaveOffset=0，此时凸轮曲线为规划的凸轮曲线如下图所示：



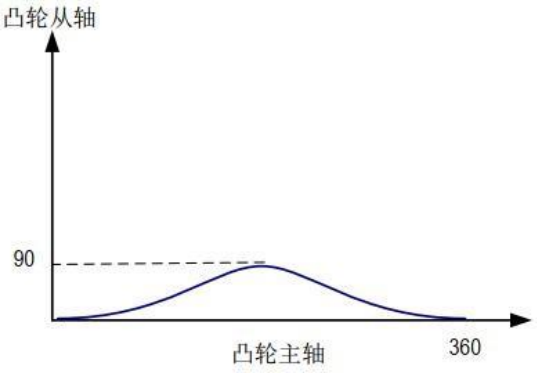
当 MasterScaling=1.0、 SlaveScaling=2.0、 MasterOffset=0、 SlaveOffset=0，此时凸轮曲线如下图所示：



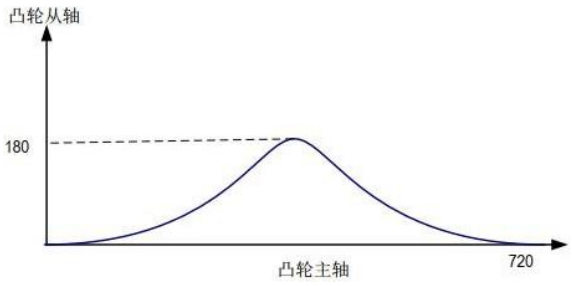
当 MasterScaling=2.0、 SlaveScaling=1.0、 MasterOffset=0、 SlaveOffset=0，此时凸轮曲线如下图所示：



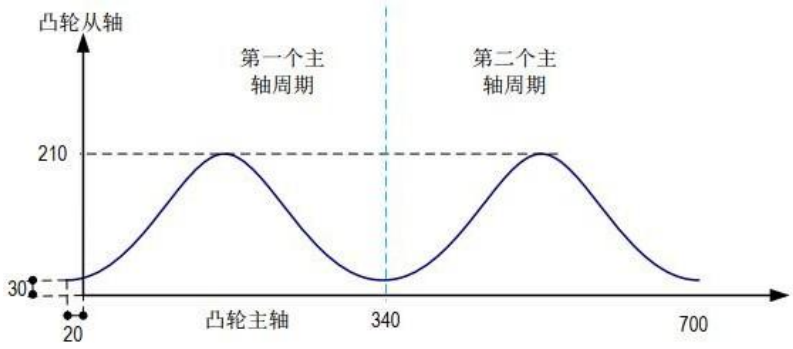
当 MasterScaling=1.0、 SlaveScaling=0.5、 MasterOffset=0、 SlaveOffset=0，此时凸轮曲线如下图所示：



当 MasterScaling=0.5、 SlaveScaling=1、 MasterOffset=0、 SlaveOffset=0，此时凸轮曲线如下图所示：



当 MasterScaling=1、 SlaveScaling=1、 MasterOffset=20、 SlaveOffset=30，此时凸轮曲线如下图所示：



凸轮运行中的 Offset、Scale 使用特性与注意事项：

① 主轴位置模式、从站位置模式，除应用系统特别要求外，建议尽量采用相对模式，这样编程简单，出现机械系统冲击的可能性比较小；

② 凸轮表主轴起止范围、Offset、Scale 等设定值，是可以弥补 CAM 表的设计偏差，建议尽量参考默认设定，这样便于调试和维护，运行出错的机会也可以减少；

③ 当 CAM 凸轮表周期执行完毕/ 或退出/ 或切换 CAM 表后，再次执行 MC_CamIn 重新进入时，系统会清除内存中的 Offset、Scale 等设定值，恢复为默认值，需要留意。

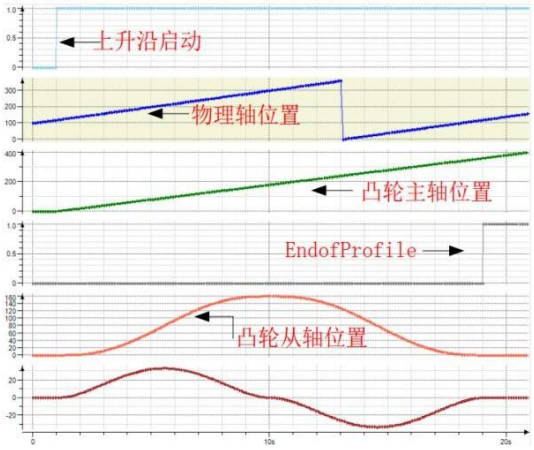
周期模式与 EndOfProfile 的关系

周期模式非周期模式决定了主轴到终止位置后电子凸轮是否要再次进行。

① 非周期模式： MC_CamTableSelect 指令 Periodic 选择 False

非周期模式时， 凸轮完成 EndofProfile 信号输出为 True ， 执行输入 FALSE 则

EndofProfile 输出 FALSE。此时凸轮只执行一个主轴周期。



注意：主轴周期指的是电子凸轮主轴位置从起始位置到终止位置的范围。

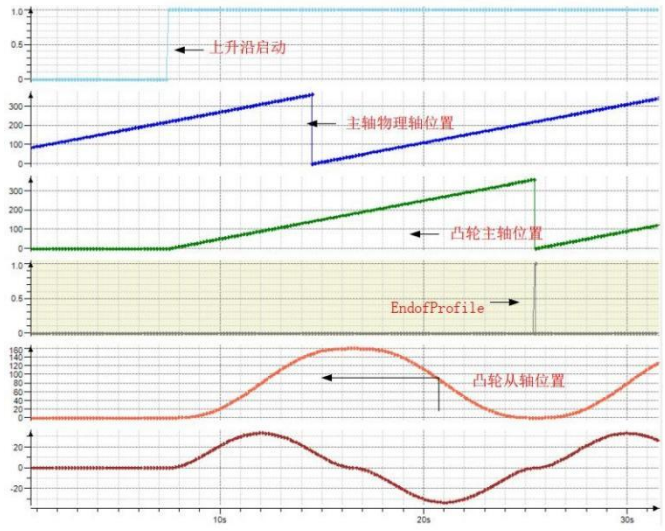
② 周期模式： MC_CamTableSelect 指令 Periodic 选择 TRUE

此时凸轮一个主轴周期完成后会接着执行下一个周期，且 EndofProfile 信号 TRUE 输出只维持一个任务周期。

注意：

当凸轮主轴位置大于等于凸轮终止位置时 EndofProfile 信号输出为 TRUE，并且凸轮主轴位置更新为：凸轮起始位置 + 大于终止位置部分。

例如：电子凸轮主轴起始位置为 0、终止位置为 360、主从轴缩放设置为 1、主从轴偏移值设为 0、任务周期 2ms，主轴速度为 100。当某个任务周期凸轮主轴位置为 359.99， 那么下个周期 EndofProfile 输出为 True 且主轴位置变为 359.99+100*0.002-360=0.19。周期模式下设计的凸轮曲线起始位置跟终止位置最好能保持平滑过渡，否则会产生跳变。比如说起始速度为 0，终止速度不为 0，会导致主轴在周期完成和新周期开始处产生跳变。


上升沿启动
主轴物理轴位置
凸轮主轴位置
EndofProfile
凸轮从轴位置
10s
20s
30s

StartMode 与 MC_CamTableSlect 中主从轴绝对相对模式的关系

绝对模式：在新的电子凸轮循环开始时，电子凸轮的计算与当前从轴的位置无关。如果从轴相对于主轴起始位置不同于从轴相对于主轴的终止位置，其将造成一个跳变。

相对模式：新的电子凸轮会根据当前从轴的位置进行改变；也就是说，从轴在上一个电子凸轮循环结束的位置，会被现在的电子凸轮运动作为“从轴偏移”进行位置相加的计算。但是， 如果在电子凸轮定义中，与主轴起始位置对应的从轴位置不是 0，其将造成一个跳变。

斜坡输入：通过增加一个补偿运动（根据极限值 VelocityDiff、加速度、减速度得出的运动） 来防止电子凸轮在开始时的潜在跳变。因此，只要从轴是旋转的方式，正向斜坡输入选项则只进行正向补偿，而反向斜坡输入则只进行反向补偿。对于线性运动的从轴，补偿方向可以自动实现，也就是说，正向斜坡输入和反向斜坡输入可以被用斜坡输入的方式进行解释。)

关系表如下表所示：

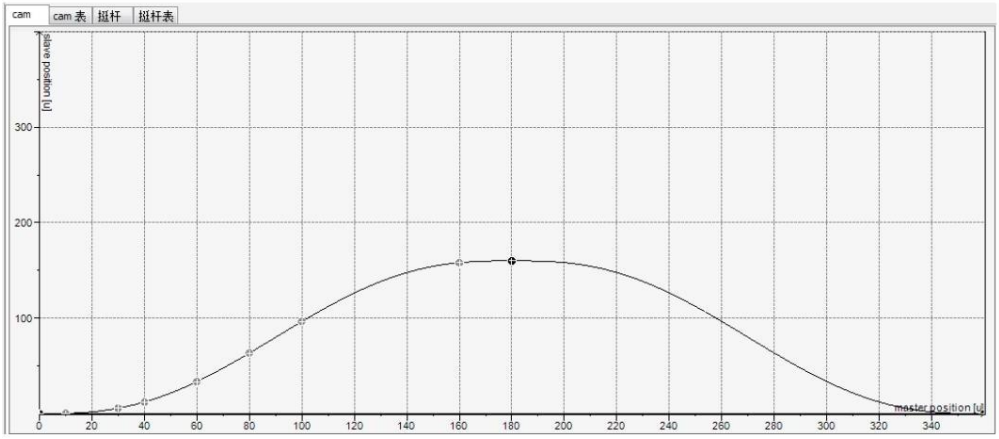
MC_CamTableSelect.MasterAbsolute	主轴模式
absolute	绝对模式
relative	相对模式

MC_CamIn.StartMode	MC_CamTableSelect.SlaveAbsolute	从轴模式
absolute	TRUE	绝对模式
absolute	FALSE	相对模式
relative	TRUE	相对模式
relative	FALSE	相对模式
ramp_in	TRUE	斜坡切入绝对模式

ramp_in	FALSE	斜坡切入相对模式
ramp_in_pos	TRUE	正向斜坡切入绝对模式
ramp_in_pos	FALSE	正向斜坡切入相对模式
ramp_in_neg	TRUE	反向斜坡切入绝对模式
ramp_in_neg	FALSE	反向斜坡切入相对模式

详细关系描述如下：

凸轮主轴范围（0-360）、凸轮从轴范围为（0-180）、周期模式、主从轴偏移值



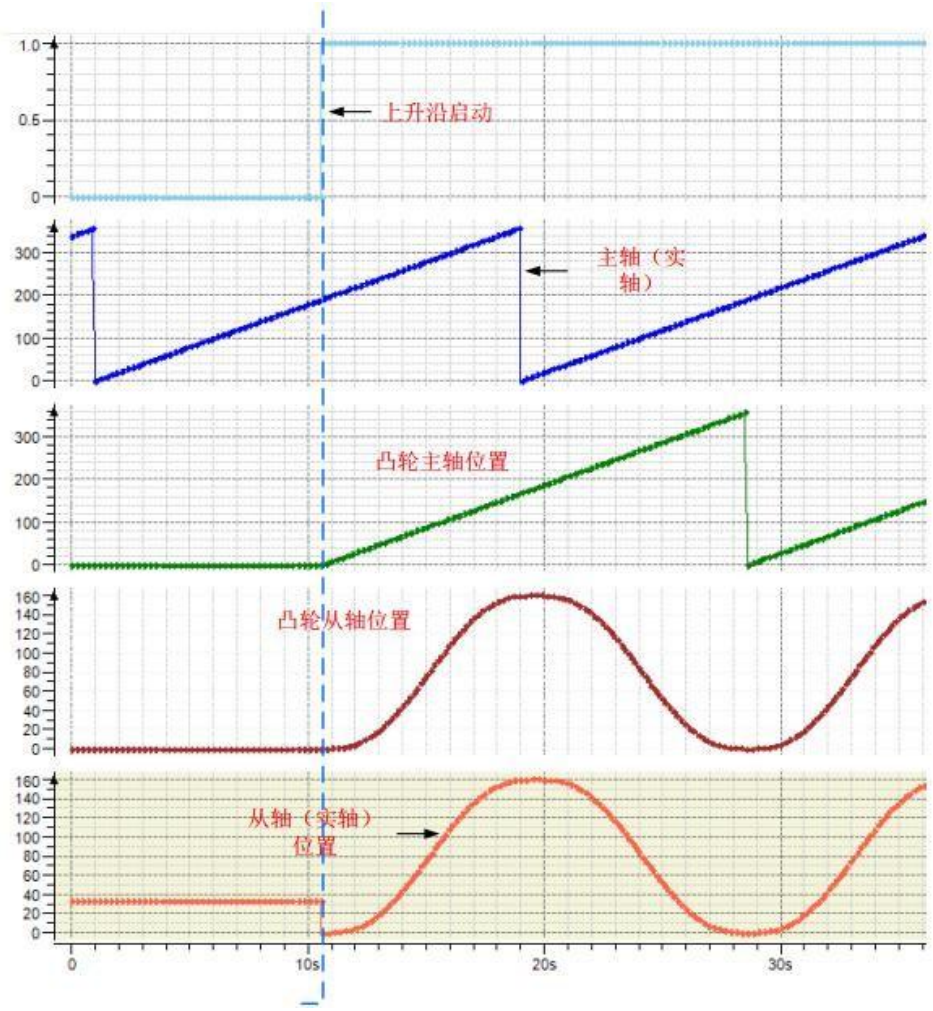
0、主从轴缩放比 1。设计的凸轮表如下图所示：

StartMode 为 0（绝对模式）

①当 MC_CamTableSlect 指令 MasterAbsolute 设置为 FALSE， SlaveAbsolute 设置为TRUE 时 ：

此时主轴工作在相对模式，从轴工作在绝对模式。当 Excute 上升沿 ，凸轮启动时，凸轮主轴从凸轮表“起始位置”（0）开始，凸轮从轴按照上述的“凸轮表齿合公式”计算输出，从轴实轴指令位置等于齿合计算输出值。比如说凸轮从轴起始位置为 0，凸轮启动时从轴实轴位置为 20，则启动开始从轴实轴的位置指令为 0 产生跳变。

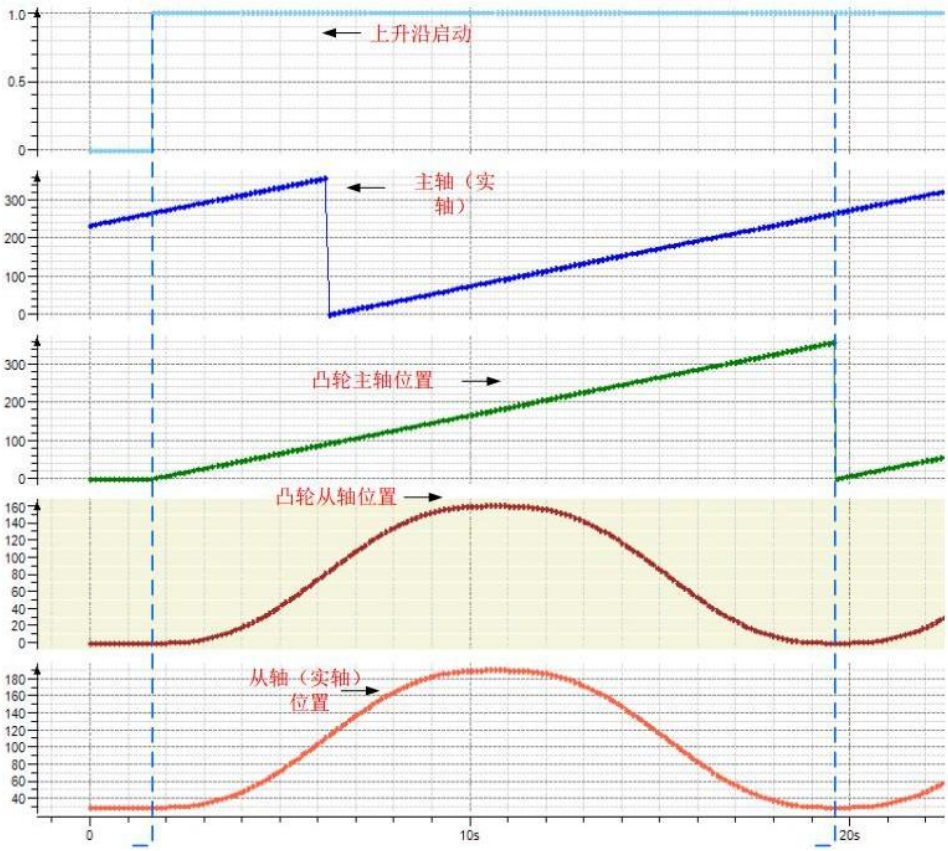
注意：该情况下从轴（实轴）开始位置不在凸轮从轴开始位置则会产生跳变。



② 当 MC_CamTableSlect 指令 MasterAbsolute 设置为 FALSE，SlaveAbsolute 设置为FALSE 时：

此时主轴工作在相对模式，从轴工作在相对模式。当 Excute 上升沿 ，凸轮启动时，凸轮主轴从凸轮表“起始位置”（0）开始，凸轮从轴按照上述的“凸轮表齿合公式”计算输出，从轴实轴指令位置等于齿合计算输出值（凸轮从轴位置）+ 启动时从轴实轴位置。

比如说凸轮启动时从轴实轴位置为 20，凸轮表从轴起始位置为 0，则启动凸轮时从轴实轴位置指令为 20，后续为 20+ 凸轮表计算值，最高峰值为 20+ 凸轮表计算最大值（此处 为 180） =200 。



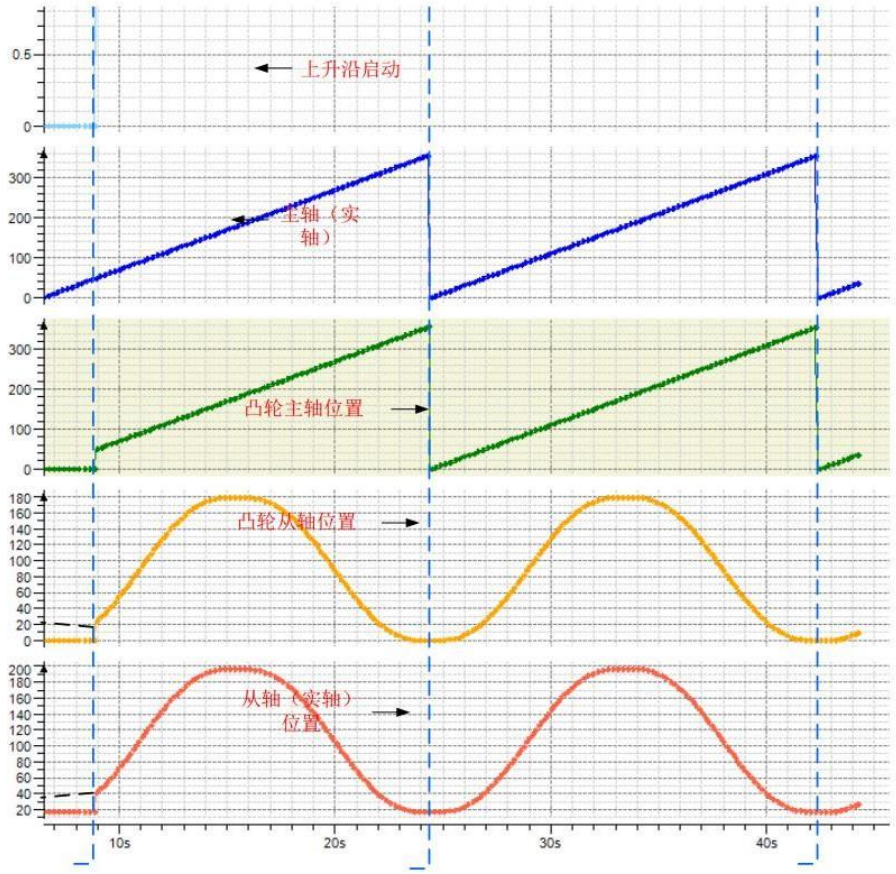
③ 当 MC_CamTableSlect 指令 MasterAbsolute 设置为 TRUE， SlaveAbsolute 设置为FALSE 时：

此时主轴工作在绝对模式，从轴工作在相对模式。当 Excute 上升沿 ，凸轮启动时，凸轮主轴从当前“主轴实轴位置开始”，从轴实轴位置指令 = 凸轮表齿合计算值（凸轮从轴位置） + 启动时从轴位置。

注意：

如果该情况下主轴（实轴）开始位置不在凸轮主轴开始位置则会产生跳变；

主轴位置应在凸轮主轴位置范围内。



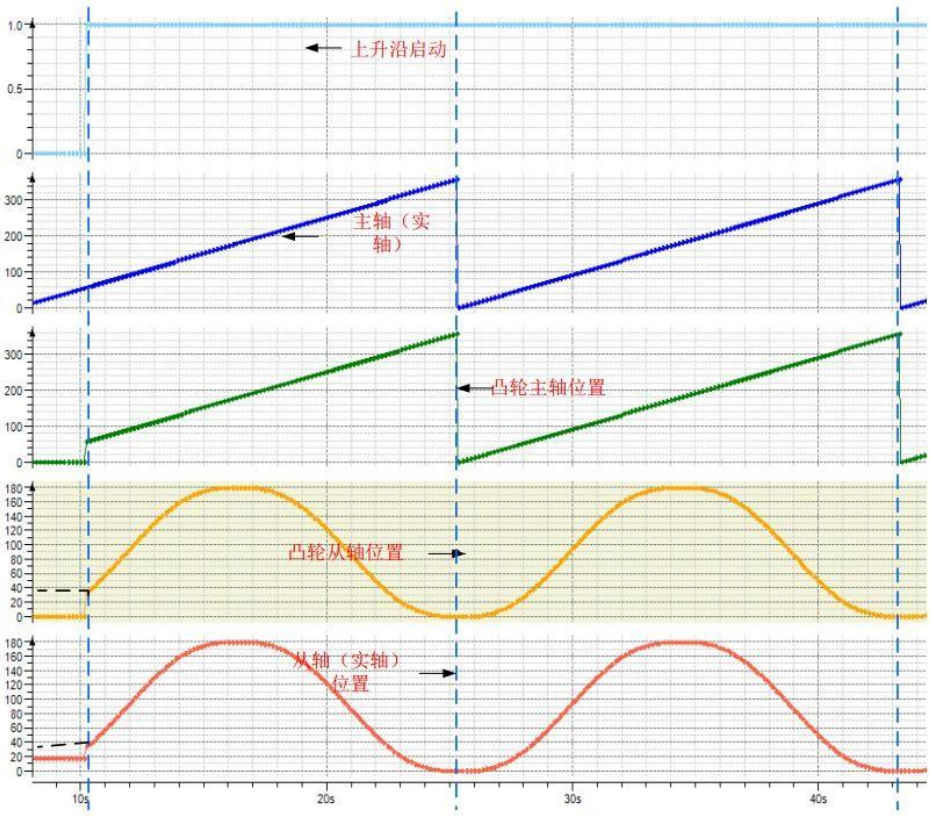
④ 当 MC_CamTableSlect 指令 MasterAbsolute 设置为 TRUE， SlaveAbsolute 设置为TRUE 时：

此时主轴工作在绝对模式，从轴工作在绝对模式。当 Excute 上升沿 ，凸轮启动时，凸轮主轴从当前“主轴实轴位置开始”，从轴实轴位置指令 = 凸轮表齿合计算值（凸轮从轴位置）。

注意：

如果该情况下主轴（实轴）开始位置不在凸轮主轴开始位置、从轴位置不在凸轮从轴开始位置则会产生跳变；

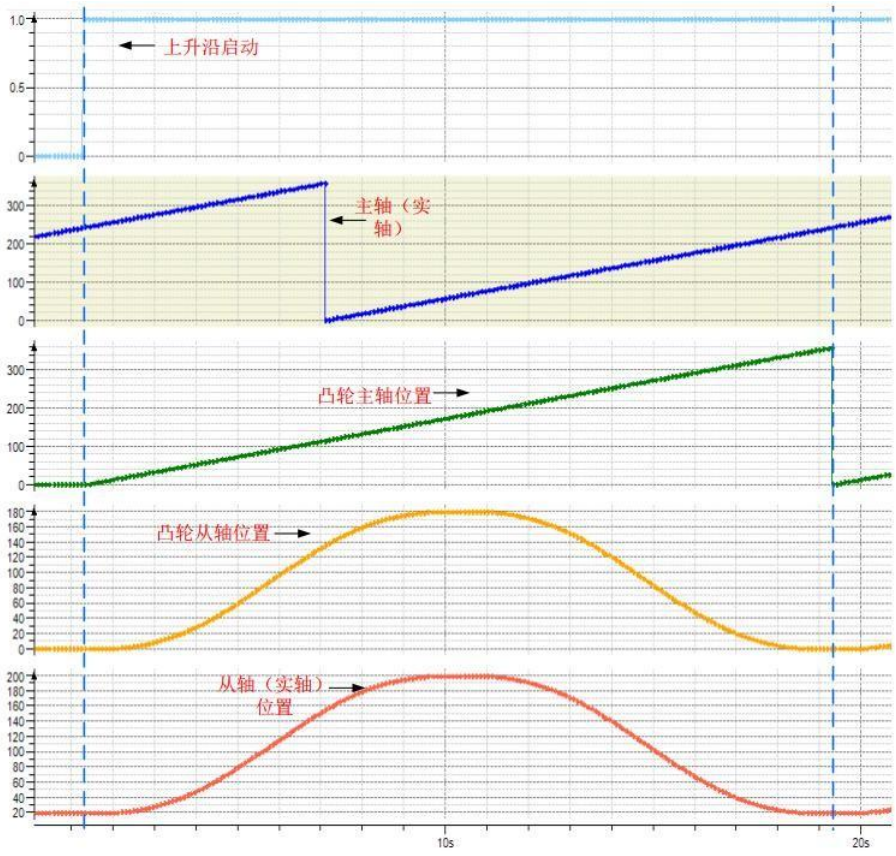
主轴位置应在凸轮主轴位置范围内。



StartMode 为 1（相对模式）

①当 MC_CamTableSlect 指令 MasterAbsolute 设置为 FALSE， SlaveAbsolute 设置为TRUE 或 者 为 False 时

此时主轴工作在相对对模式，从轴工作在相对模式。当 Excute 上升沿 ，凸轮启动时， 凸轮主轴从 “凸轮表起始位置开始”，从轴实轴位置指令 = 凸轮表齿合计算值 + 凸轮表齿合计算值（凸轮从轴位置）。



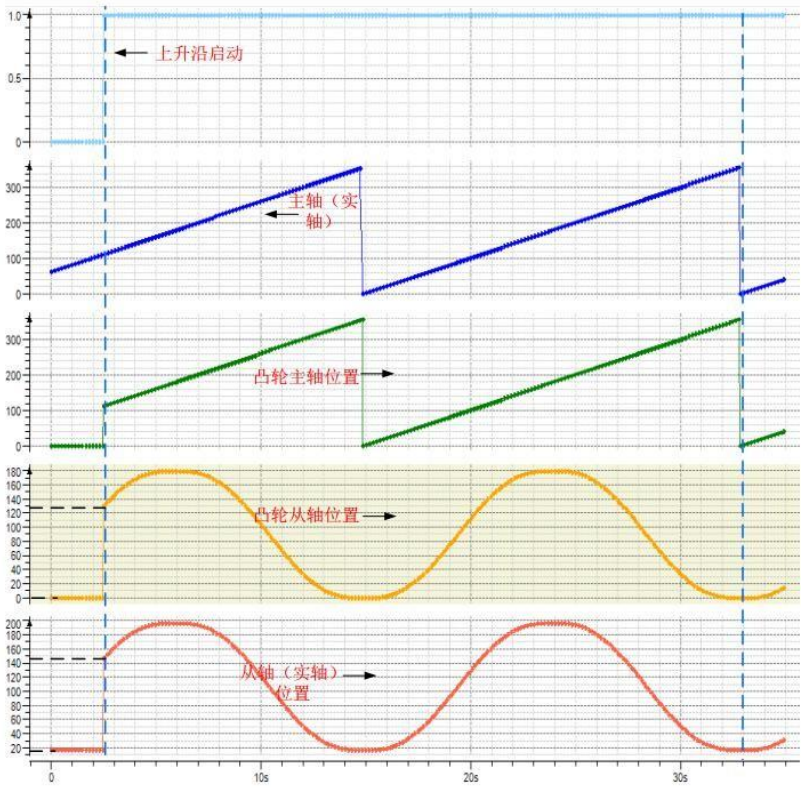
②当 MC_CamTableSlect 指令 MasterAbsolute 设置为 TRUE， SlaveAbsolute 设置为TRUE 或 者 为 False 时：

此时主轴工作在绝对模式，从轴工作在相对模式。当 Excute 上升沿 ，凸轮启动时，凸轮主轴从 “主轴当前位置开始”，从轴实轴位置指令 = 启动时从轴位置 + 凸轮表齿合计算值（凸轮从轴位置）。

注意：

如果该情况下主轴（实轴）开始位置不在凸轮主轴开始位置则会产生跳变。

主轴位置应在凸轮主轴位置范围内

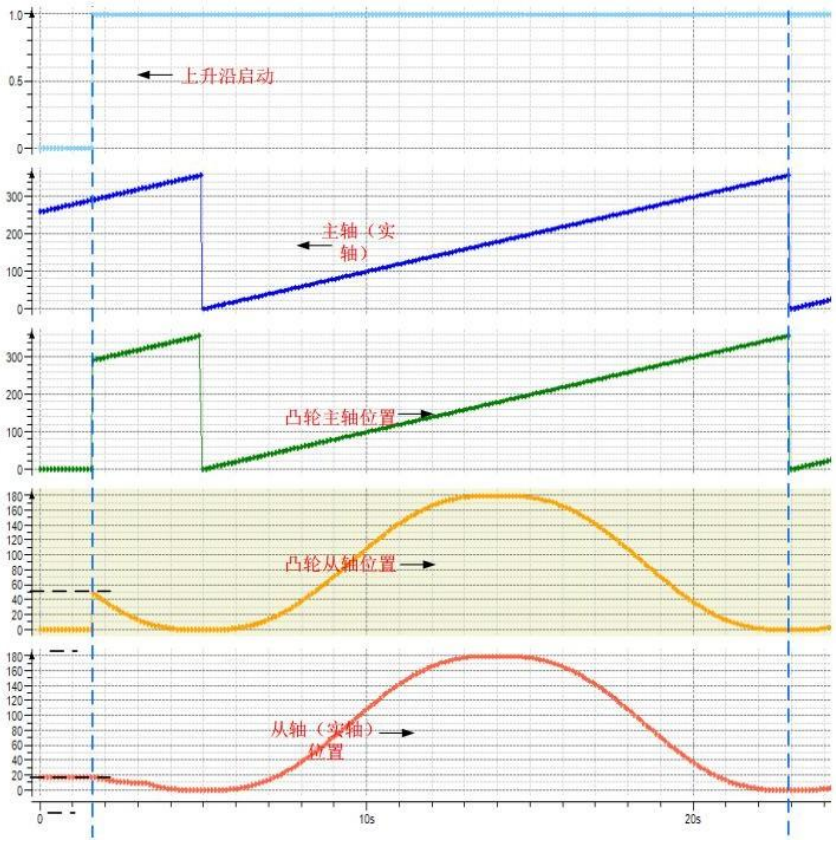


StartMode 为 2（rampin 斜坡切入模式）

① 当 MC_CamTableSlect 指令 MasterAbsolute 设置为 TRUE， SlaveAbsolute 设置为TRUE 时：

此时主轴工作在绝对模式，从轴工作在绝对模式。当 Excute 上升沿 ，凸轮启动时，凸轮主轴从 “主轴当前位置开始”，从轴通过设定的 VelocityDiff、Acceleration、 Deceleration 增加一个补偿运动避免切入时的潜在跳变。

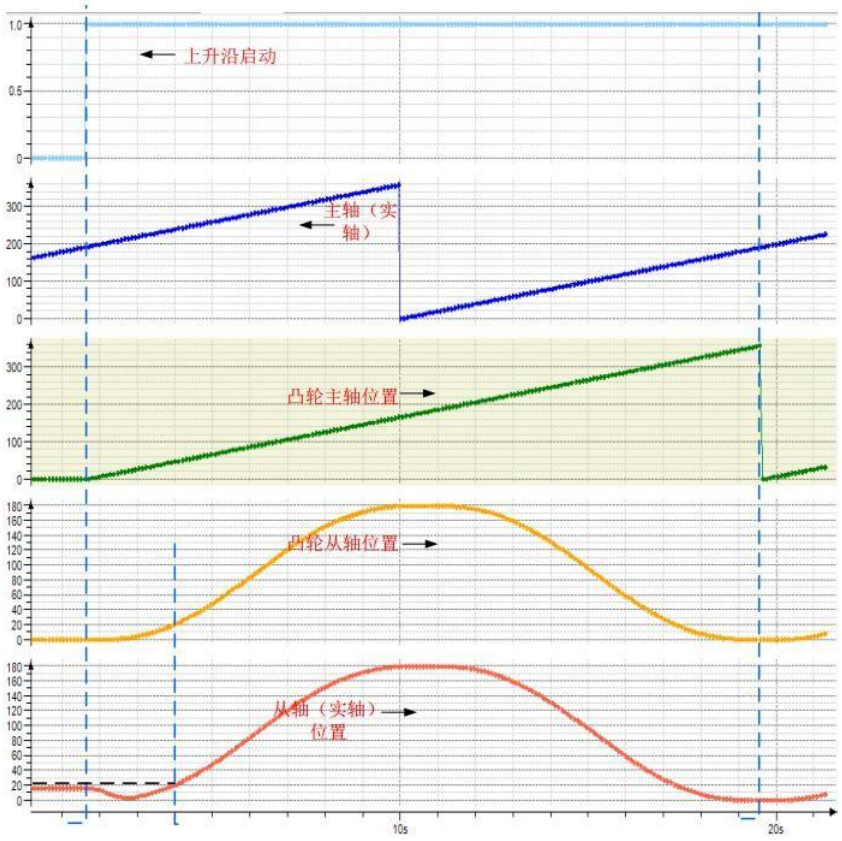
从轴实轴位置指令 = 凸轮表齿合计算值（凸轮从轴位置）+f(VelocityDiff,Acceleration,Deceleration)。



② 当 MC_CamTableSlect 指令 MasterAbsolute 设置为 FALSE，SlaveAbsolute 设置为TRUE 时：

此时主轴工作在相对模式，从轴工作在绝对模式。当 Excute 上升沿 ，凸轮启动时，凸轮主轴从“凸轮主轴起始位置开始”，从轴通过设定的 VelocityDiff、Acceleration、Deceleration增加一个补偿运动避免切入时的潜在跳变。

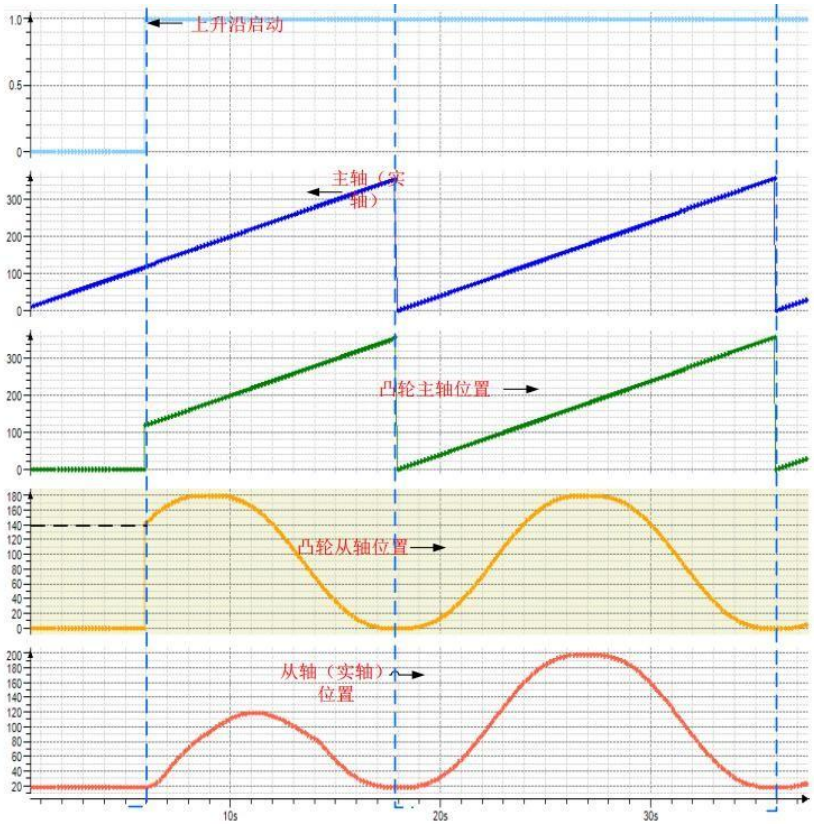
从轴实轴位置指令 = 凸轮表齿合计算值（凸轮从轴位置）+f(VelocityDiff,Acceleration,Deceleration)。



③ 当 MC_CamTableSlect 指令 MasterAbsolute 设置为 TRUE， SlaveAbsolute 设置为FALSE 时，此时主轴工作在绝对模式，从轴工作在相对模式。当 Excute 上升沿 ，凸轮启动时凸轮主轴从 “主轴当前位置开始”，从轴通过设定的 VelocityDiff、 Acceleration、Deceleration 增加一个补偿运动避免切入时的潜在跳变。

从轴实轴位置指令 = 从轴当前位置 + 凸轮表齿合计算值（凸轮从轴位置）+f(VelocityDiff,Acceleration,Deceleration)。

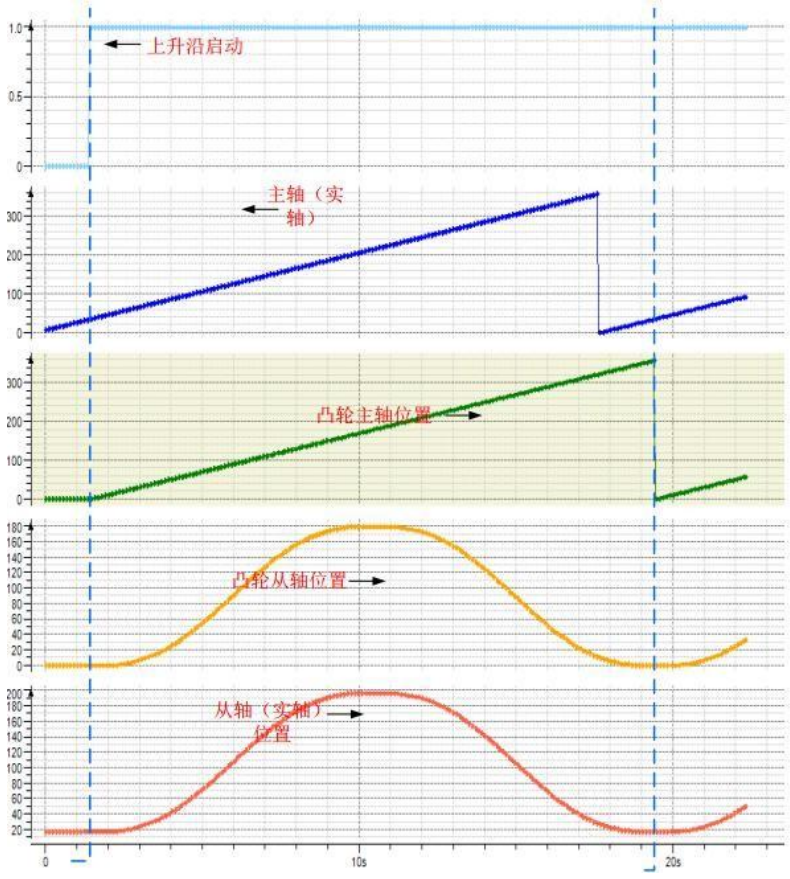
注意：该方式下第一个主轴周期内凸轮曲线与设计曲线可能存在较大变化



④ 当 MC_CamTableSlect 指令 MasterAbsolute 设置为 FALSE，SlaveAbsolute 设置为FALSE 时，此时主轴工作在相对模式，从轴工作在相对模式。当 Excute 上升沿 ，凸轮启动时，凸轮主轴从 “凸轮主轴起始位置开始”，从轴通过设定的 VelocityDiff、 Acceleration、Deceleration 增加一个补偿运动避免切入时的潜在跳变。

从轴实轴位置指令 = 从轴当前位置 + 凸轮表齿合计算值（凸轮从轴位置）+f(VelocityDiff,Acceleration,Deceleration)。

注意：该方式下第一个主轴周期内凸轮曲线与设计曲线可能存在较大变化



StartMode 为 3、 4（正向斜坡切入 ramp_in_pos、反向斜坡切入 ramp_in_neg）当从轴工作为“旋转模式” ramp_in_pos 只会沿着轴正向运动方向补偿，ramp_in_neg 只朝轴反向运动方向补偿，当轴工作为线性模式 ramp_in_pos、ramp_in_neg、 ramp_in 三种模式都是为补偿方向自动调整，也就是说如果轴设置为线性模式工作则 ramp_in_pos、ramp_in_neg、 ramp_in 三种启动模式工作情况是一样的。

电子凸轮重启

基本上，两个电子凸轮在任何时候都可以切换，但是需要考虑一些情况：在电子凸轮编 辑器中，从轴的位置定义为电子凸轮函数的计算输出，电子凸轮函数是以在主轴范围内的一 个主轴位置为计算条件，由此可以用下述简单的公式来进行表达说明：

SlavePosition = CAM(MasterPosition)

因为主轴驱动的实际周期一般与电子凸轮定义的主轴范围是不同的，所以为满足电子凸 轮函数的正确输入，主轴位置必须被按比例调整到函数定义域内：

SlavePosition = CAM(MasterScale*MasterPosition + MasterOffset)

类似的情况，如果一个电子凸轮在绝对值模式下启动，产生了一个向上的跳变，函数输 出（也就是虚拟从轴位置）也会按比例被修正：

SlavePosition = SlaveScale*CAM(MasterPosition)+ SlaveOffset

最坏情况下，这两种比例修正都必须被采用，所以，事实上从轴位置（SlavePosition）是由更为复杂的公式计算得出的：

Slaveposition =SlaveScale*CAM(MasterScale*Masterposition + MasterOffset) + SlaveOffset 在每个电子凸轮周期结束时，比例和偏移可以改变以获得更为合适的参数。遗憾的是，电

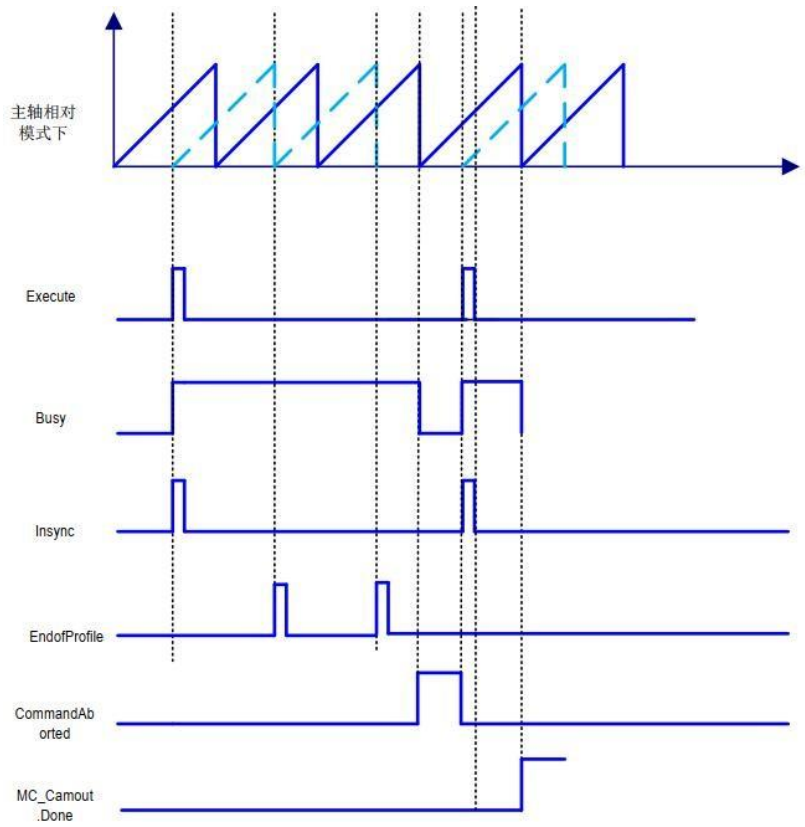
子凸轮的 MC_CamIn 模块的重新启动，将会删除它的内存并且包括比例值和偏移值。因此，

所定义的电子凸轮函数会适应于一般各种不同的从轴数值。基于这个原因，建议只有在需要　　处理另外一个不同的电子凸轮时，才去重启 MC_CamIn-FB。

(1)时序图：

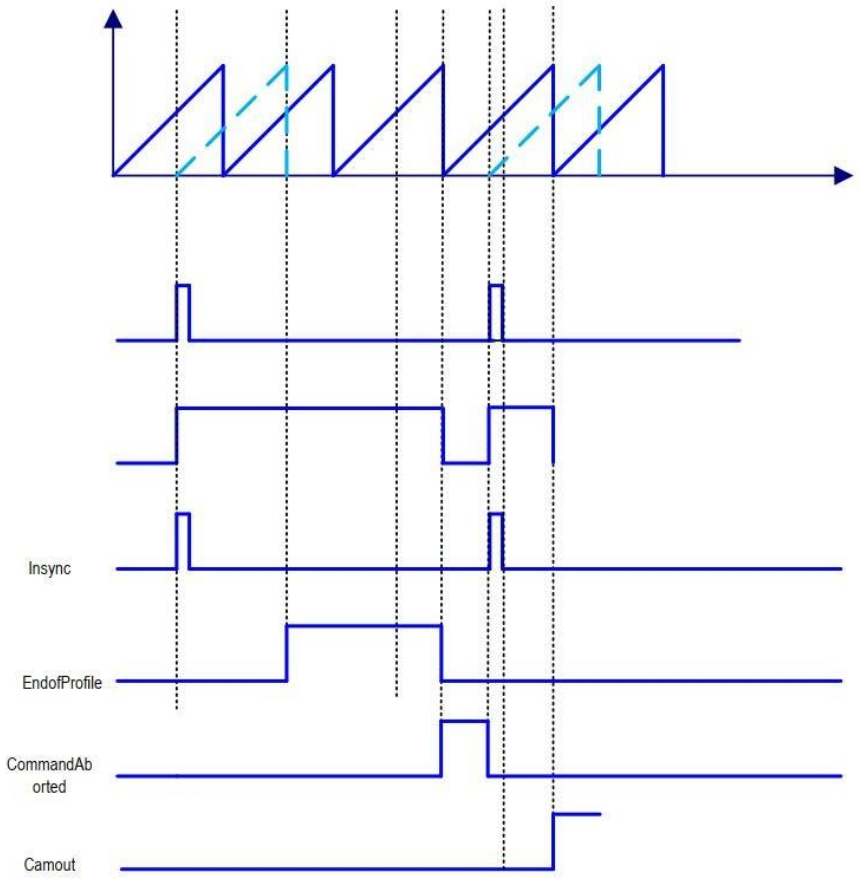
周期模式（MC_CamTableSelect.Periodic 设为 TRUE）如下图：

注意： MC_Camout 指令只切断主从轴的凸轮耦合关系，如果断开时候从轴速



度不为 0， 则从轴不会自动减速为 0，需配合使用 MC_STOP 指令 .

非周期模式（MC_CamTableSelect.Periodic 设为 FALSE）如下图：



(2)错误说明

指令设置信息与 Camslect 指令设置信息不相符合。轴没有使能情况。

启动本指令检测到异常时，Error(错误) 变为 TRUE。

可查看 ErrorID(错误代码) 的输出值，阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.6 MC_CamOut

断开从轴的凸轮耦合关系。当从轴正在凸轮运行，触发执行该功能块可使得 Slave 从轴退出凸轮运行状态，进入连续运行状态（Continuios_Motion，即 Axis.nAxisState=5），该指令的执行对主轴没有任何影响。注意：执行该指令后从轴会按照分离前的速度继续运行，所以需要跟 MC_Stop 等指令配合使用。

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_CamOut	断开凸轮耦合		MC_CamOut(Slave:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

(2)相关变量

输入输出

输入输出变量	名称	数据类型	有效范围	初始值	描述
Slave	从轴	AXIS_REF	-	-	映射到轴,即 AXIS_REF_SM3 的一个实例

输入

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行指令	BOOL	-	-	上升沿信号执行指令

输出

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE,FALSE	- FALSE	完成与主轴的凸轮耦合断开
Busy	执行中	BOOL	TRUE,FALSE	FALSE	指令执行中

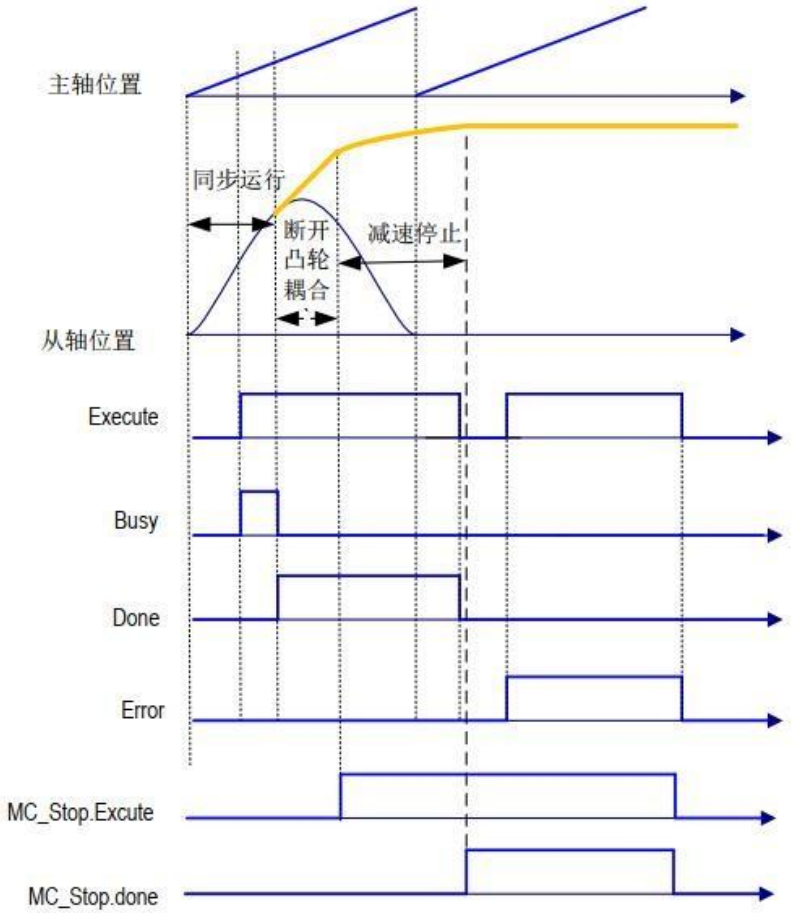
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
ErrorID	错误代码	SMC_ERROR	参阅SMC_ERROR	0	异常发生时，输出错误代码

(3)功能说明

执行本指令解除从轴的凸轮耦合关系，Excute 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；

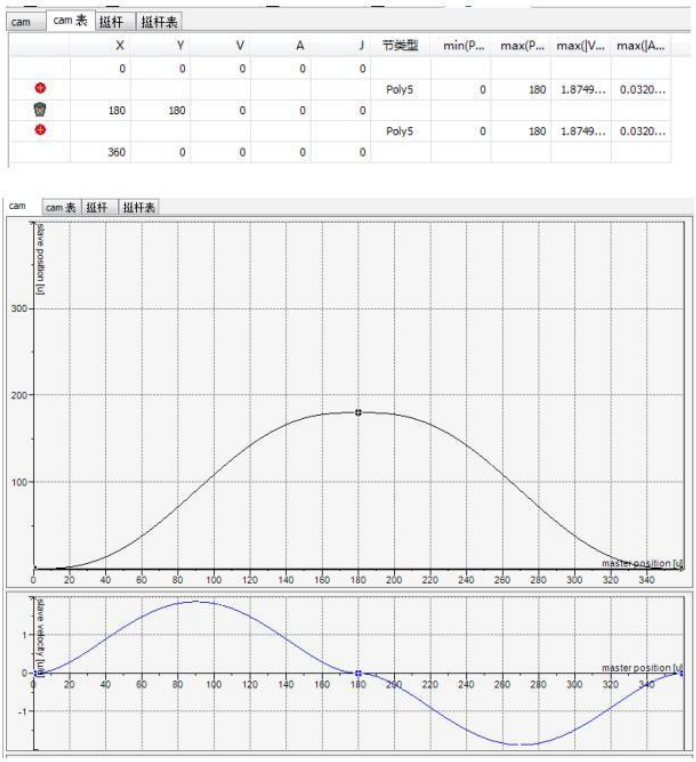
如果从轴在执行该指令前速度不为 0，则指令 DONE 信号完成后凸轮耦合关系断开但是从轴任然按照切出去前速度运行；

从轴没有凸轮耦合关系执行该执行，则 ERROR 输出。(4) 时序图

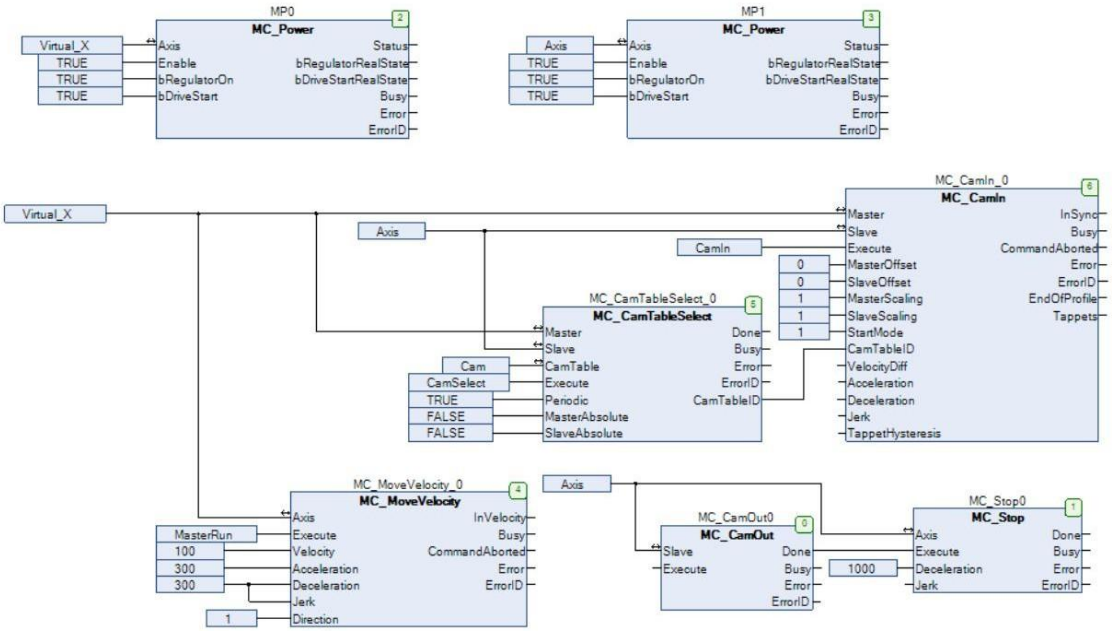


(5)使用范例

本范例应用凸轮相关指令，介绍凸轮关系的建立、运行及脱离时轴的相关运动状态凸轮编辑器建立如下凸轮表（cam）：



程序：



上电后主从轴自动使能， MasterRun 置为 TRUE 则主轴以 100 速度运行 CamSelect 置为 True，选择凸轮表，然后 CamIn 置为 True 启动电子凸轮。需要断开电子凸轮时将 MC_CamOut0.Execute 置为 True。

(5)错误说明

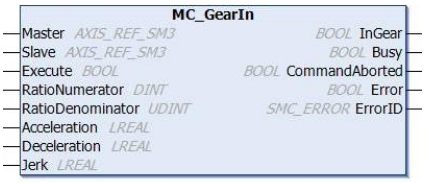
启动本指令发生异常时， Error 输出为 True。

可查看 ERRORID，参考“附录 C 错误代码说明” 以了解 SMC_ERROR 错误代码。

6.4.7 MC_GearIn

设定从轴与主轴间的齿轮比，进行电子齿轮动作。

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_GearIn	电子齿轮功能块		MC_GearIn0(Master:= , Slave:= , Execute:= , RatioNumerator:= , RatioDenominator:= , Acceleration:= , Deceleration:= , Jerk:= , InGear=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

(2)相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例
Slave	从轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行	BOOL	TRUE-FAKSE	FALSE	上升沿，开始执行指令
RatioNumerator	齿轮比分子	DINT	正数、负数	1	齿轮比分子
RatioDenomina	齿轮比分母	UDINT	正数	1	齿轮比分

tor					母
Acceleration	加速度	LREAL	正数或 0		指定加速度
Deceleration	减速度	LREAL	正数或 0		指定减速度
Jerk	越度	LREAL	正数或 0		加加速度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InGear	齿轮比到达	BOOL	TRUE-FAKSE	- FALSE	True，从轴达到目标速度
Busy	执行中	BOOL	-TRUE,FALSE	FALSE	True，正在执行指令
Command Aborted	中断	BOOL	TRUE,FALSE	FALSE	True，被其他控制指令中断
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

(3)功 能 说 明

Execute 上升沿， 开始电子齿轮动作。

执行电子齿轮后要解除耦合必须通过 GearOut 指令。

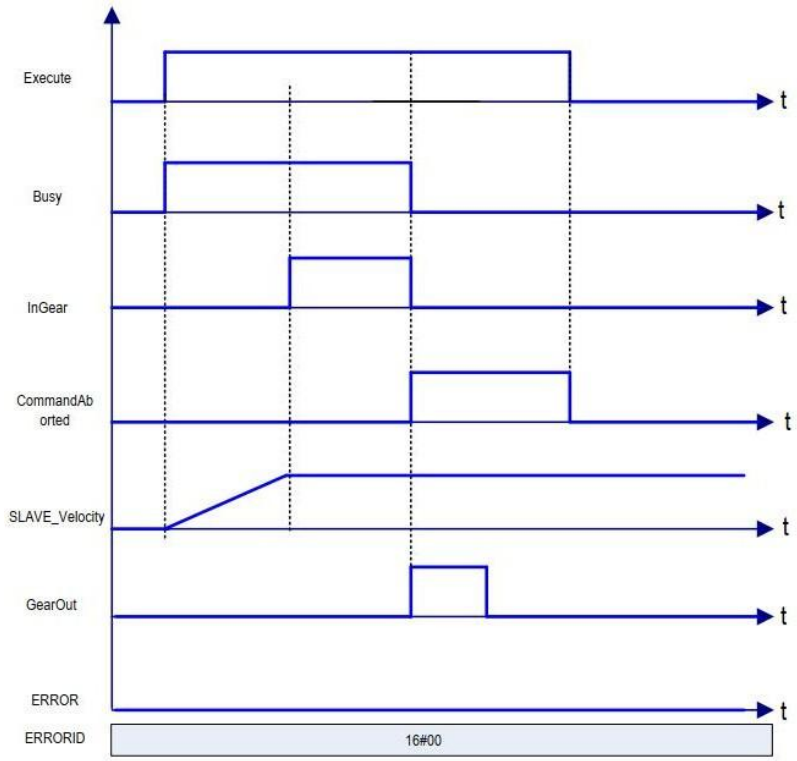
该指令为速度电子齿轮功能，加速过程中造成的同步距离丢失不会自动补偿。

指令执行中 Busy 信号为 TRUE 时，如果从轴目标速度没有达到此时 Execute 新的上升沿不会影响它。

指令执行中 Busy 信号为 TRUE 时，如果从轴目标速度达到此时 Execute 新的上升沿不会影响它。到达目标速度， InGear 为 TRUE， 此后从轴移动量 = 主轴移动量 * RatioNumerator/ RatioDenominator。

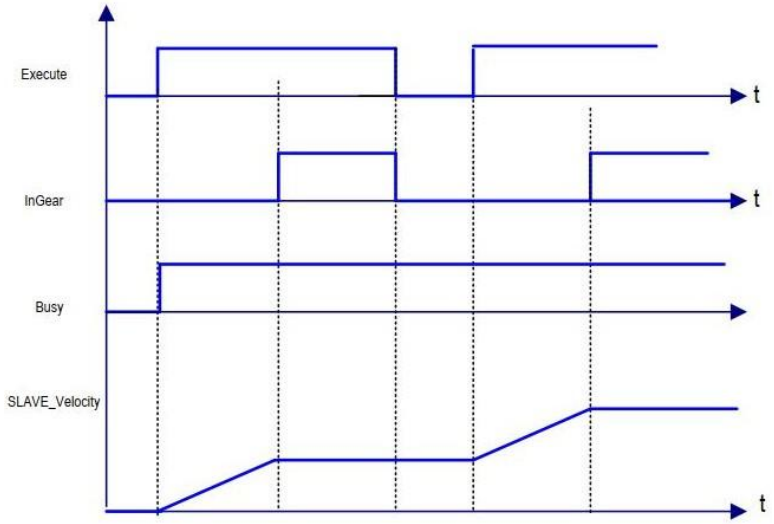
如果主轴速度实时变化的，情况下请注意慎重使用该指令。

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。



时序图：

变更齿轮比参数后重启指令时序图如下：



(4)错误说明

启动指令， ERROR 为 TRUE, 则有错误输出。

相关错误根据 ERRORID 参考 SMC_ERROR。请阅读“附录 C 错误代码说明”以了解相关错误代码说明。

6.4.8 MC_GearOut

终止执行中的 MC_GearIn,MC_GearInPos 指令。

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_GearOut	电子齿轮耦合断开		MC_GearOut0(Slave:= Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);

(2)相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Slave	从轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	执行	BOOL	TRUE- FAKSE	FAKSE	上升沿，开始执行指令

输出变量

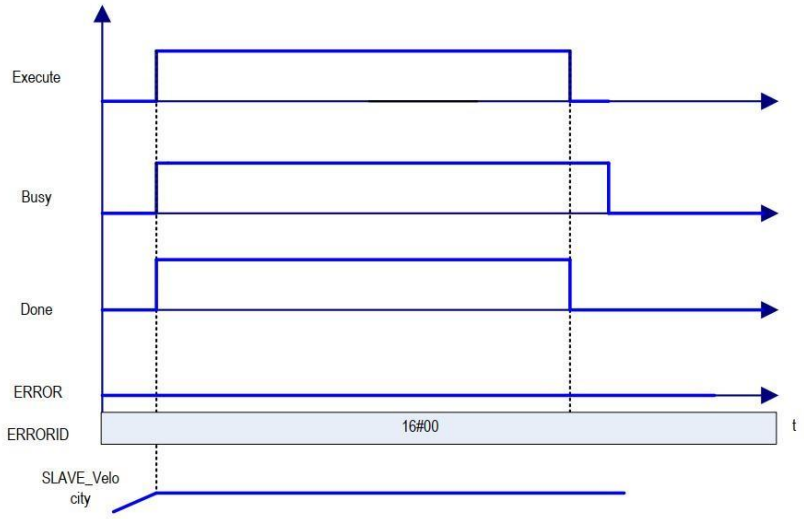
输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FAKSE	FALSE	True，从轴与主轴电子齿轮耦合断开
Busy	执行中	BOOL	TRUE,FALSE	FALSE	True，指令正在执行中

Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

(3)功能说明Execute上升沿,执行切出电子齿轮动作。Excute为TRUE，ERROR为FALSE则Busy输出为TRUE，Done输出为TRUE。

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。

Execute下降沿则，Done为FALSEMC_Stop指令执行复位Busy信号



(4)错误说明

相关参数设置有错，会导致指令报警。轴没有使能会导致指令报警。

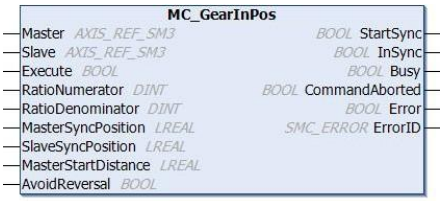
【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.9 MC_GearInPos

设定主轴与从轴之间的电子齿轮比，进行电子齿轮动作。

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。

(1)指令格式

指令	名称	图形表现	ST 表 现
MC_GearInPos	指定位置切入电子齿轮耦合		MC_GearInPos0(Master:= Slave:= Execute:= , RatioNumerator:= , RatioDenominator:= , MasterSyncPosition:= , SlaveSyncPosition:= , MasterStartDistance:= , AvoidReversal:= , StartSync=> , InSync=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

(2)相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例
Slave	从轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	指令执行	BOOL	TRUE FAKSE	-FALSE	上升沿，开始执行指令
Ratio Numerator	齿轮比分	DINT	-	1-	主从轴速度比的分子

	子				
Ratio Denominator	齿轮比分母	DINT		1	主从轴速度比的分母
Master SyncPosition	主轴同步位置	LREAL			主从轴齿轮比耦合时主轴位置
Slave SyncPosition	从轴同步位置	LREAL			主从轴齿轮比耦合时从轴位置
Master StartDistance	执行同步主轴位置	LREAL			从轴按照该位置值和MasterSyncPosition 以及SlaveSyncPosition 值计算一条平滑曲线使从轴在SlaveSyncPosition 时跟主轴齿轮同步，曲线主轴范围为[MasterStartDistance，MasterSyncPosition]
AvoidReversal	禁止反转	BOOL	TRUE FAKSE	FALSE	设置为 FALSE ，如果从轴物理位置超前的情况下进行反转。设置为 TRUE 如果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。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
StartSync	开始耦合处理	BOOL	TRUE- FALSE	ALSE	True，开始电子齿轮耦合处理
InSync	耦合中	BOOL	TRUE- FALSE	FALSE	True，电子齿轮耦合处理完成，主从轴齿轮比耦合中
Busy	指令处理中	BOOL	TRUE,FALSE	FALSE	True，指令正在处理中
Command Aborted	指令中断	BOOL	TRUE,FALSE	FALSE	被其它控制指令中断
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
ErrorID	错误代码	SMC_ERROR	参阅SMC_ERROR	0	异常发生时，输出错误代码

4）功能说明

Execute 上升沿信号，开始执行指令。开始动作后， Slave（从轴）以 Master（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工作在周期模式。例如：

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 >MasterSyncPositionMasterStartDistance，或者从轴位置 >

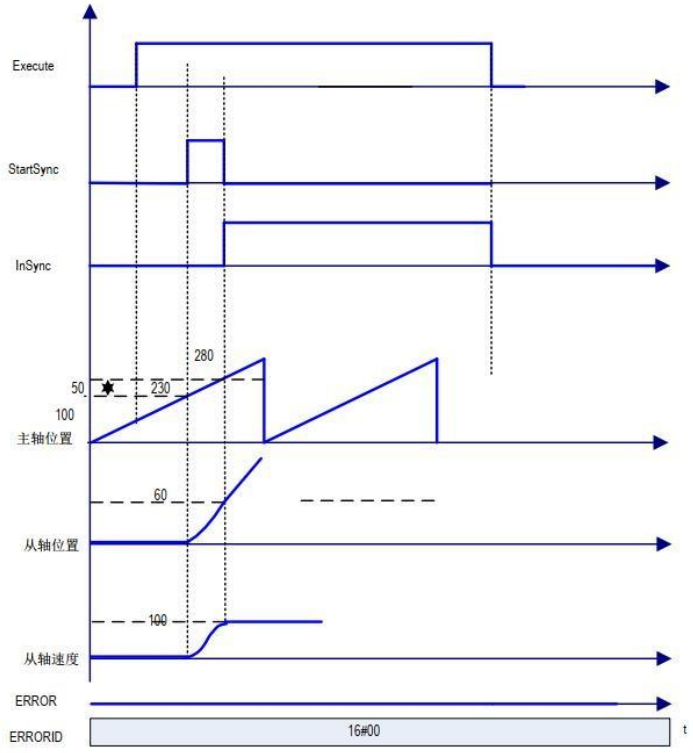
SlaveSyncPosition 则无法切入电子齿轮动作。

下面给出了几种不同参数下的样例时序图：

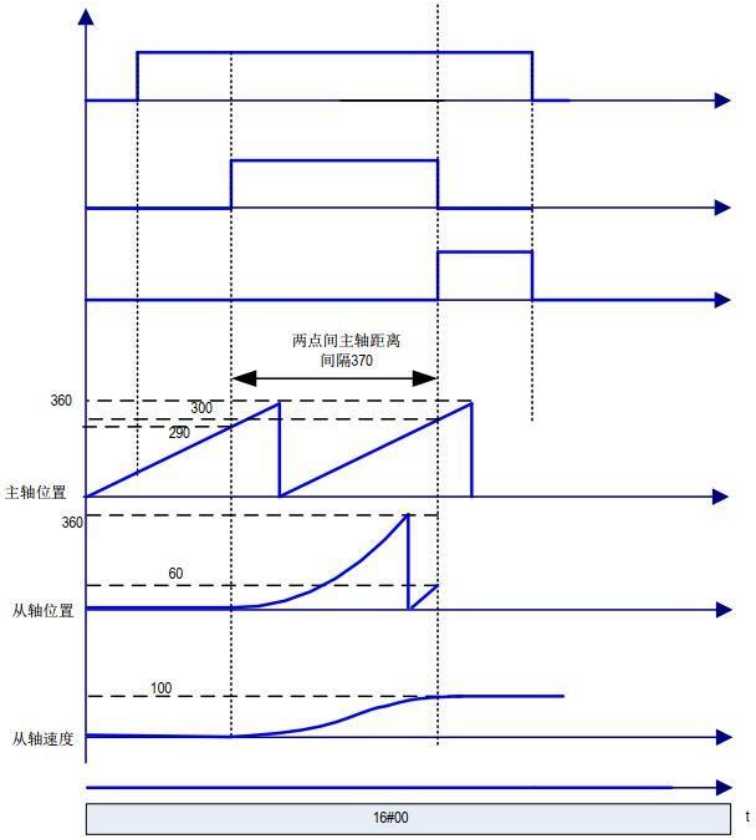
当主轴工作在周期模式（360 循环）、从轴工作在周期模式（360 循环）：

① MasterSyncPosition=280、 MasterStartDistance=50、 SlaveSyncPosition=60，主轴速度为 50 、

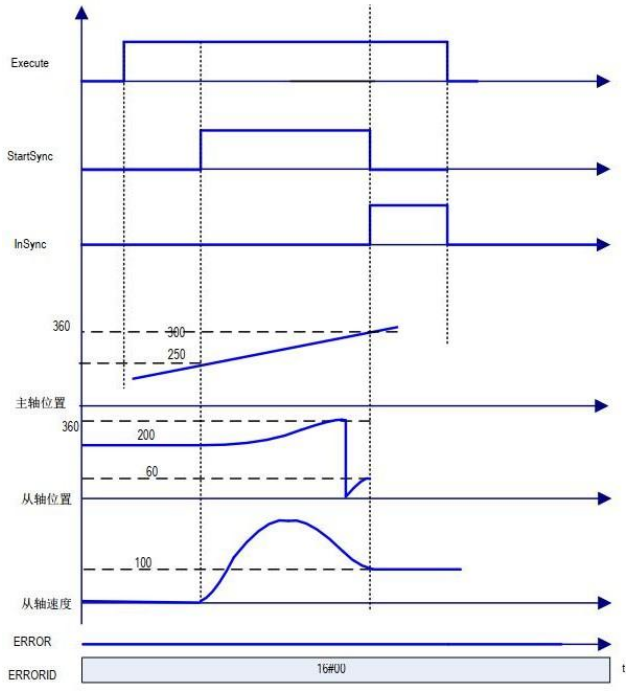
AvoidReversal=FALSE



② MasterSyncPosition=300、 MasterStartDistance=370、 SlaveSyncPosition=60，主轴速度为 50、AvoidReversal=FALSE

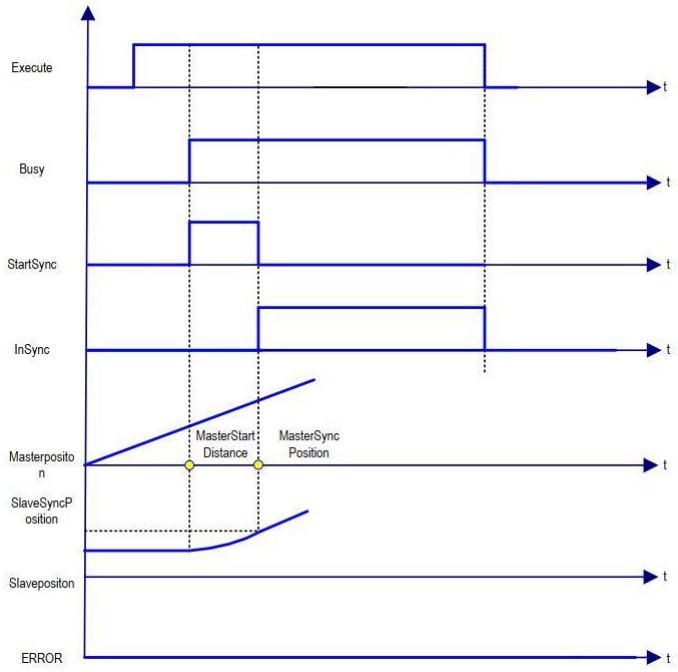


③ MasterSyncPosition=300、 MasterStartDistance=50、 SlaveSyncPosition=60，主轴速度为 50、AvoidReversal=FALSE、从轴起始位置大于 60



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。

(5)时序图



(6)错误说明

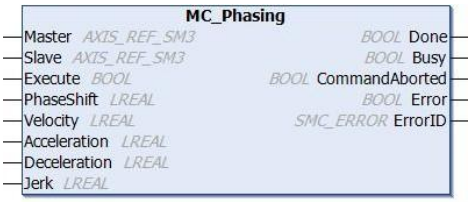
相关参数设置有错，会导致指令报警。

轴没有使能会导致指令报警。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.10 MC_Phasing

指定主从轴之间的相位偏差。 （1）指令格式

指令	名称	图形表现	ST 表 现
MC_Phasing	主从轴相位偏移		MC_Phasing0(Master:= Slave:= Execute:= , PhaseShift:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

（2）相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例
Slave	从轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入相关变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	指令执行	BOOL	TRUE-FAKSE	FALSE	上升沿，开始执行指令
PhaseShift	主从轴相位偏差值	LREAL		0	主从轴相位偏差值，正数代表从轴滞后。
Velocity	速度	LREAL		0	执行相位偏移时最大速度值
Acceleratio	加速度	LREAL		0	执行相位偏移时最大加

n					速度值
Deceleration	减速度	LREAL		0	执行相位偏移时最大减速度值
Jerk	速度二次导数	LREAL		0	执行相位偏移时最大 Jerk 值

输出相关变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE- FALSE	FALSE	True，如果相位偏移完成
Busy	指令处理中	BOOL	TRUE,FALSE	FALSE	True，指令正在处理中
Command Aborted	指令中断	BOOL	TRUE,FALSE	FALSE	被其它控制指令中断
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

（3）功能说明

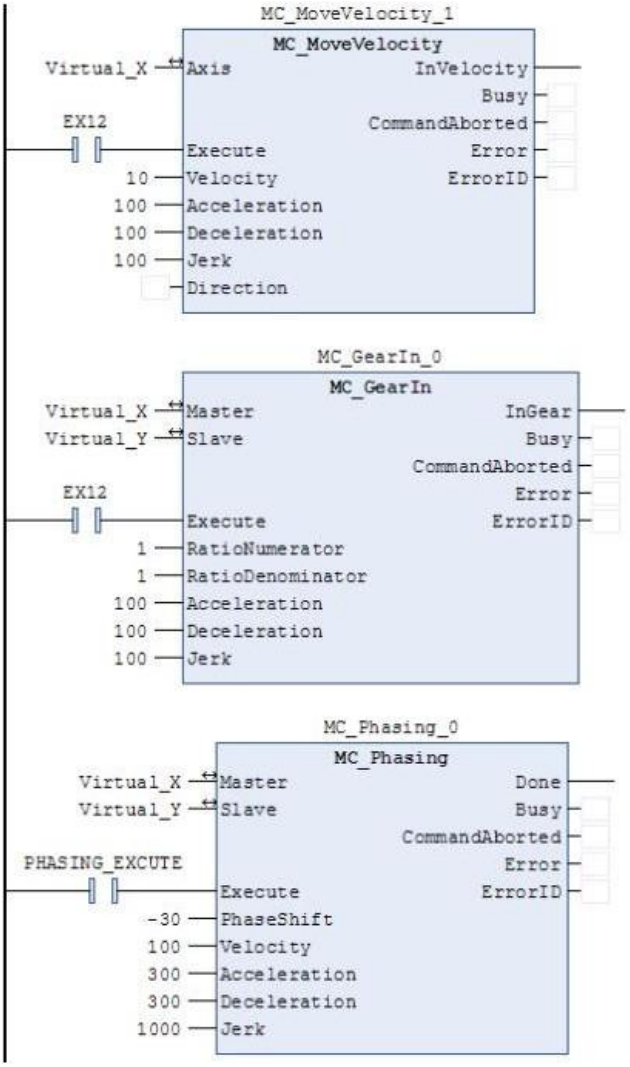
Execute 上升沿执行相位偏移，从轴自动计算一条平滑曲线，完成从轴对主轴的相位偏移，主从轴相位差为输入信号的 PhaseShift 值，正值为从轴滞后于主轴。完成偏移后 Done 信号输出为 True。

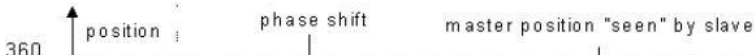
根据设定的 PhaseShift、Velocity、Acceleration、Deceleration 对主从轴相位差进行补偿。主从轴相位差达到 PhaseShift 时，Done 信号输出。

执行指令时主轴指令位置与反馈位置不变，从轴进行调整，完成后，从轴跟主轴之间相位差为 PhaseShift 。

该指令最终结果为轴给定值之间的相位偏移，所以实轴的实际反馈值与最终的偏移可能不一致。

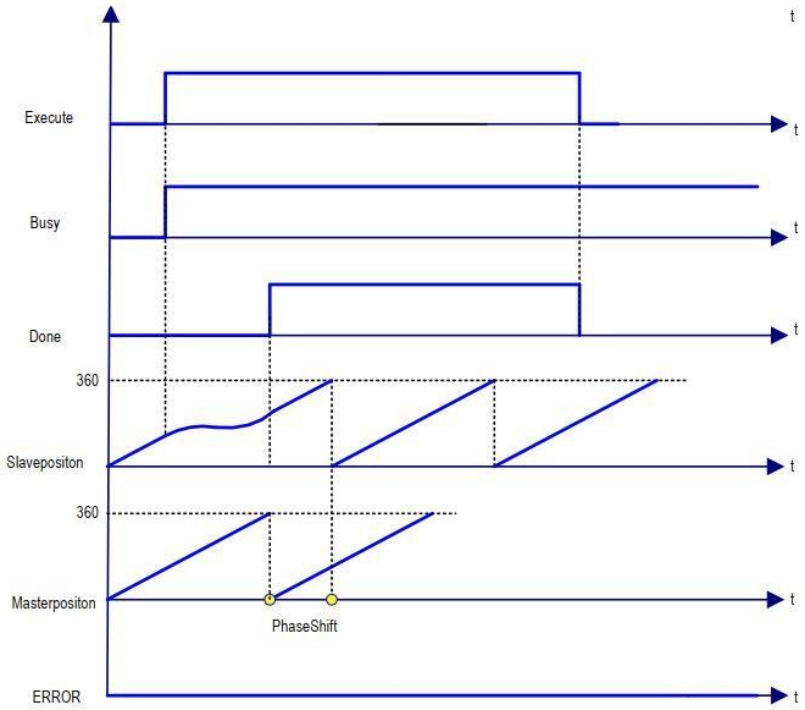
该指令跟MC_GearIn 指令配合使用，如下：主轴为Virtual_x，从轴为Virtual_y,EX12 上升沿执行主轴速度控制以及主从轴电子齿轮动作，然后执行相位偏移。此外可以跟电子凸轮配合使用，此时从轴作为“电子凸轮主轴”以达到电子凸轮主轴相位偏移的效果。





（4）时序图

以主从轴都按 360 周期运动， Execute 信号上升沿执行调整，调整完成后从轴与主轴之间相位偏差为PhaseShift 设定的值。



（5）错误说明

启动指令时， Error 输出为 TRUE，则有错误发生。

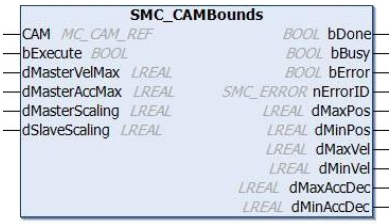
查看 ErrorID,查看帮助中 SMC_ERROR 确定报警信息，请阅读“附录 C 错误代码说明” 了解相关错误代码说明。

6.4.11 SMC_CAMBounds

当从轴与主轴凸轮耦合后可以通过该功能块来计算出从轴最大位置，速度，加速度。

主轴在输入最大速度、加减速限制下运动。该指令在设计凸轮表时候可以检查曲线是否正确，前提知道主轴最大加减速，速度等。

（1）指令格式

指令	名称	图形表现	ST 表 现
SMC_CAMBounds	凸轮上下限		MC_Phasing0(Master:= Slave:= Execute:= , PhaseShift:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

（2）相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
CAM	凸轮	MC_CAM_REF	-	-	映射到凸轮，即MC_CAM_REF的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	指令执行	BOOL	TRUE-FAKSE	FALSE	上升沿，开始执行指令
dMasterVelMax	最大速度	LREAL	-	1	绝对模式下主轴最大速度。
dMasterAccMax	最大加速度	LREAL	-	0	绝对模式下主轴最大加速度

dMasterScaling	标尺因子	LREAL	-	1	主轴凸轮应用中标尺因子
dSlaveScaling	标尺因子	LREAL	-	1	从轴凸轮应用中标尺因子

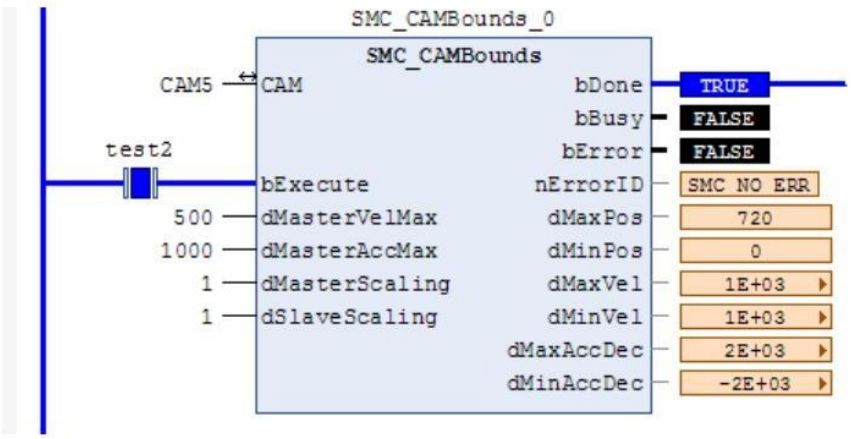
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	完成	BOOL	TRUE- FALSE	FALSE	True，如果计算完成
bBusy	指令处理中	BOOL	TRUE,FALSE	FALSE	True，指令正在处理中
bError	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
nErrorID	错误代码	SMC_ERROR	参阅SMC_ERROR	0	异常发生时，输出错误代码
dMaxPos	最大位置	LREAL		0	根据凸轮表计算出从轴最大位置
dMinPos	最小位置	LREAL		0	根据凸轮表计算出从轴最小位置
dMaxVel	最大速度	LREAL		0	计算出最大速度
dMinVel	最小速度	LREAL		0	计算出最小速度
dMaxAccDec	最大加速度	LREAL		0	计算出最大加速度
dMinAccDec	最小加速度	LREAL		0	计算出最小加速度

（4）功能说明

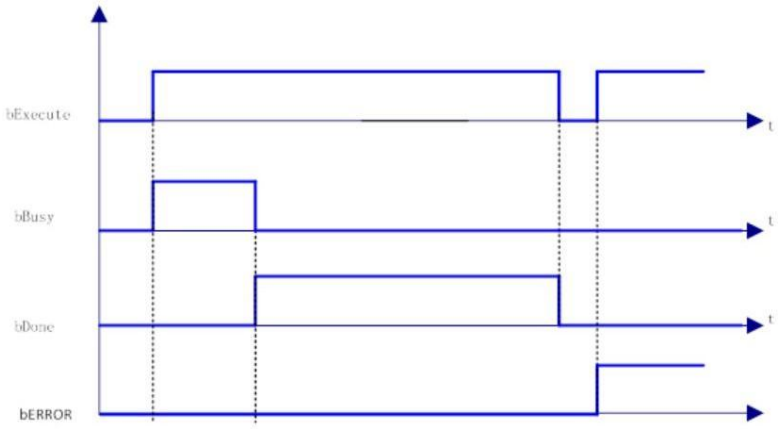
bExecute 上升沿来综合输入变量的“dMasterVelMax”，“dMasterAccMax”，“dMasterScaling”，

“dSlaveScaling”值与凸轮表数据计算出从轴“最大位置”最小位置等值。例如：主轴周期 360， 凸轮表为一条斜率为 2 的直线，计算出的结果如下图所示：



主轴以绝对模式运行或者主轴设置为周期模式，模值设置为主轴周期都可使用该指令计算。凸轮表为 XYVA（多项式模式时有效），一维数组、二维数组等无效

（5）时序图



（6）错误说明

凸轮表格式不是多项式模式。

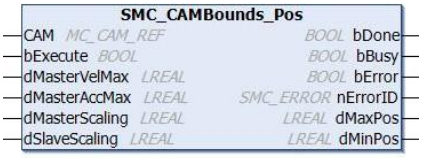
凸轮表 MC_CAM_REF 设定值与实际凸轮表不匹配。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.12 SMC_CAMBounds_Pos

当从轴与主轴凸轮耦合后可以通过该功能块来计算出从轴最大位置，与最小位置。该功能块与SMC_ CAMBounds 相比少了最大加速度等计算 ，其他功都一致。

（1）指令格式

指令	名称	图形表现	ST 表 现
SMC_ CAMBounds_Pos	凸 轮 位 置 上 下 限		SMC_CAMBounds_Pos0(CAM:= , bExecute:= , dMasterVelMax:= , dMasterAccMax:= , dMasterScaling:= , dSlaveScaling:= , bDone=> , bBusy=> , bError=> , nErrorID=> , dMaxPos=> , dMinPos=>);

（2）相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
CAM	凸轮	MC_CAM_ REF	-	-	映射到凸轮，即MC_CAM_REF 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	指令执行	BOOL	TRUE-FAKSE	FALSE	上升沿，开始执行指令
dMasterVelMax	最大速度	LREAL		1	绝对模式下主轴最大速度。
dMasterAccMax	最大加速度	LREAL		0	绝对模式下主轴最大加速度
dMasterScalin	标尺因子	LREAL		1	主轴凸轮应用中标尺因

g					子
dSlaveScaling	标尺因子	LREAL		1	从轴凸轮应用中标尺因子

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	完成	BOOL	TRUE-FALSE	FALSE	True，如果计算完成
bBusy	指令处理中	BOOL	TRUE,FALSE	FALSE	True，指令正在处理中
bError	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
nErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码
dMaxPos	最大位置	LREAL		0	根据凸轮表计算出从轴最大位置
dMinPos	最小位置	LREAL		0	根据凸轮表计算出从轴最小位置

（3）功能说明

bExecute 上升沿来综合输入变量的“dMasterVelMax”，“dMasterAccMax”，“dMasterScaling”，“dSlaveScaling”值与凸轮表数据计算出从轴“最大位置”最小位置。

主轴以绝对模式运行或者主轴设置为周期模式，模值设置为主轴周期都可使用该指令计算。凸轮表为 XYVA（多项式模式时有效），一维数组、二维数组等无效

（4）错误说明

凸轮表格式不是多项式模式；凸轮表 MC_CAM_REF 设定值与实际凸轮表不匹配。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.13 SMC_WriteCAM

程序运行时用于将编辑的凸轮表存储为一个文件。使之可以被 MC_CamIn 等指令使用。生成的文件包含内容参考“Cam Format”。

该指令可以用来跟 SMC_ReadCAM 配合使用。

（1）指令格式

指令	名称	图形表现	ST 表 现
SMC_WriteCAM	凸 轮 上 下限		

（2）相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初 始值	描述
CAM	凸轮	MC_CAM_REF	-	-	映射到凸轮，即 MC_CAM_REF 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	指 令 执行	BOOL	TRUE,FAKSE	FALSE	上升沿，开始执行指令
sFileName	文件名	STRING	‘’		包含凸轮描述的以 ASCII 格式定义的文件名， 可以通过帮助里面” Cam Format”来查看具体描述。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

bDone	完成	BOOL	TRUE- FALSE	FALSE	True，如果凸轮写进文件完成
bBusy	指令处理中	BOOL	TRUE,FALSE	FALSE	True，指令执行没有完成
bError	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为TRUE
nErrorID	错误代码	SMC_ERROR	参阅SMC_ERROR	0	异常发生时，输出错误代

（3）功能说明

bExecute 上升沿，指令执行 - 将“CAM”连接的凸轮信息存储到文件名为“sFileName”连接的文件中。

存储成功并完成 bDone 信号输出为 true。

存储的凸轮表信息受硬件内存限制。

注意：该功能在程序运行中的执行，凸轮表信息也可以手动存储到离线信息

（4）错误说明

该指令只能完成 XYVA 多项式模式的凸轮表，一维，二维等会造成错误输出；

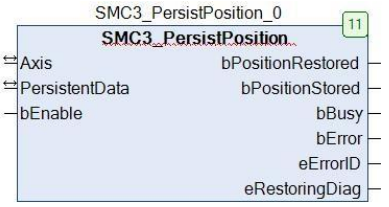
sFileName 连接的文件名不存在或者信息错误。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明

6.4.14 SMC3_PersistPosition

该指令用来保持记录实轴绝对值编码器的位置（断电重启控制器后，恢复断电前位置记录值）。 如果伺服电机使用的是绝对值编码器，使用该功能块配合使用。

（1）指令格式

指令	名称	图形表现	ST 表 现
SMC3_ PersistPositi on	轴 位置保持		SMC3_PersistPosition0(Axis:= , PersistentData:= , bEnable:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

（2）相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例
PersistentData	保持数据	SMC3_Per sist Position_D ata			存储位置信息的断电保持型数据结构

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	执行	BOOL	TRUE,FALSE	FALSE	True 功能块执行， false 不执行功能块，若要在初始化期间还原上次存储的位置，则必须从应用程序启动时将该值置

					为 true

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bPosition Restored	位置恢复	BOOL	TRUE,FALSE	FALSE	TRUE，轴重启后位置恢复完成
bPosition Stored	位置保存	BOOL	TRUE,FALSE	FALSE	TRUE，调用功能快后保存位置完成
bBusy	FB 执行中	BOOL	TRUE,FALSE	FALSE	TRUE，功能块没有执行完成
bError	错误	BOOL	TRUE,FALSE	FALSE	TRUE，异常发生
eErrorID	错误代码	SMC_ ERROR	SMC_NO_ERROR		异常发生时，输出错误代码
eRestoringDiag	恢复诊断	SMC3_Persistl PositionD iag	SMC3_ PersistPosition Diag. SMC3_PPD_ RESTORING_ OK		位置恢复中的诊断信息 SMC3_PPD_RESTORING_OK：位置成功恢复 SMC3_PPD_AXIS_PROP_CHANGED: 轴参数有更改，无法恢复位

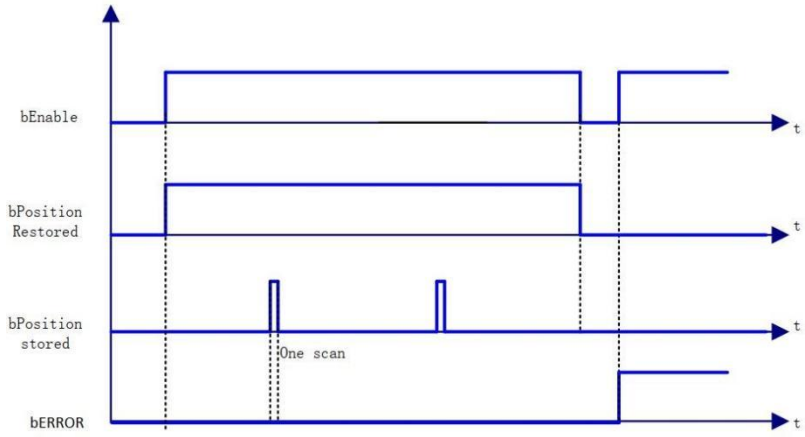
					SMC3_PPD_DATA_STORED_DURING_WRITING: 功能块从轴参数数据结构复制数据，而不是从 PersistentData 数据中复制。可能原因：非同步性持续变量、控制器崩溃死机

（3）功能说明

PLC 重启 bEnable 信号为 TRUE，则 bPositionRestroed 输出为 TRUE。

不支持虚轴跟逻辑轴。

时序图



（4）错误说明

输入轴为虚拟轴或者逻辑轴会导致错误输出。

轴有错误。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.15 SMC_FollowVelocity

不做任何检查直接给轴设定速度。该指令与 MC_MoveVelocity 有所不同，执行上升沿信号后，每个任务周期都会给轴速度指令（MC_MoveVelocity 指令速度更改后必须刷新执行才能生效）。

（1）指令格式

指令	名称	图形表现	ST 表现
SMC_FollowVelocity	轴速度给定		SMC_FollowVelocity_0(Axis:= , bExecute:= , fSetVelocity:= , bBusy=> , bCommandAborted=> , bError=> , iErrorID=>);

（2）相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE,FALSE	FALSE	上升沿执行功能块
fSetVelocity	设定速度	LREAL		0	轴设定的速度

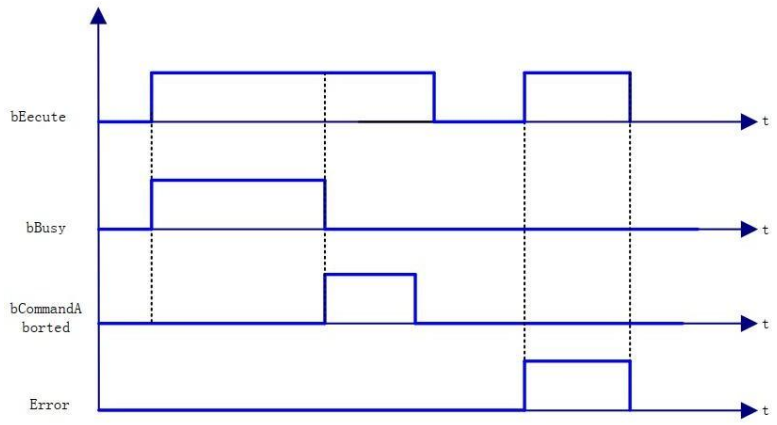
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True：指令执行中，（此时轴处于同步状态，与凸轮 MC_CamIn 指令运行时轴状态一样），可以用

					MC_Camout 指令清除 bBusy 状态
bCommand Aborted	指令被中断	BOOL	TRUE,FALSE	FALSE	True：轴被其他控制命令打断（bExecute 为 True 时）
bError	错误	BOOL	TRUE,FALSE	FALSE	True，异常产生
iErrorID	错误代码	SMC_ERROR			参考 SMC_Error

（3）功能说明SMC_FollowVelocity 通过 bExecute 的上升沿启动之后，轴会每个任务周期给轴发送速度指令。bBusy 信号来时会轴的状态为同步运行与 MC_CamIn 指令生效时从轴状态一样，可以用 MC_CamOut指令清除。bExecute 信号为 TRUE 时，当有其他控制命令中断该指令则 bBusy 由 TRUE 变为 FALSE。

时序图



（4）错误说明

bExecute

上升沿时：Axis 变量连接的为非 AXIS_REF_SM3 类型结构变量，Error 输出；轴没使能，Error 输出；

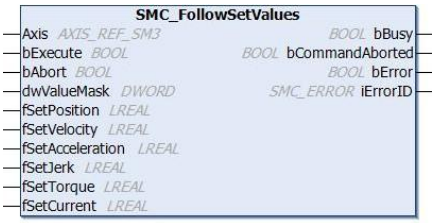
指令运行中，轴出错，Error 输出。

【注意】：请阅读“附录 C 错误代码说明”以了解相关错误代码说明。

6.4.16 SMC_FollowSetValues

跟其他的 SMC_Follow 功能一样都是直接给轴指令。但是该指令不仅包含其他 SMC_Follow 指令功能，还包括加速度、电流、转矩等控制信号，可认为是个综合版。通过 DwValueMask 值来选择所需指令。

（1）指令格式

指令	名称	图形表现	ST 表现
SMC_FollowSetValues	轴相关指令给定		SMC_FollowSetValues_0(Axis:= , bExecute:= , dwValueMask:= , fSetPosition:= , fSetVelocity:= , fSetAcceleration:= , fSetJerk:= , fSetTorque:= , fSetCurrent:= , bBusy=> , bCommandAborted=> , bError=> , iErrorID=>);

（2）相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE,FALSE	FALSE	上升沿执行功能块
DwValueMask	控制管理	DWORD		0	Bite0:TRUE:fSetPosition 激活 FALSE: 忽略 Bite1:TRUE: fSetVelocity 激活 FALSE: 忽略 Bite2:TRUE: fSetAcceleration 激活

					FALSE: 忽略 Bite3:TRUE: fSetJerk 激活 FALSE: 忽略 Bite4:TRUE: fSetTorque 激活 FALSE: 忽略 Bite5:TRUE: fSetCurrent 激活 FALSE: 忽略
bAbort	中断	BOOL	TRUE,FALSE	FALSE	上升沿中断功能块
fSetPosition	设定位置	LREAL		0	轴设定的位置（标定好的单位）
fSetVelocity	设定速度	LREAL		0	轴设定的速度（标定好的单位/s）
fSetAcceleration	设定加速度	LREAL		0	轴设定的加速度（标定好的单位 /s2）
fSetJerk	设定跃度值	LREAL		0	轴设定的跃度值（标定好的单位 /s3）
fSetTorque	设定转矩	LREAL		0	轴设定的转矩值（NM/N）
fSetCurrent	设定电流	LREAL		0	轴设定的电流值（A）

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True- 指令执行中，（此时轴处于同步状态，与凸轮MC_CamIn指令运行时轴状态一样），可以用MC_Camout 指令清除bBusy 状态
bCommand Aborted	指令被中断	BOOL	TRUE,FALSE	FALSE	True- 轴被其他控制命令打断
bError	错误	BOOL	TRUE,FALSE	FALSE	True, 异常产生
iErrorID	错误代码	SMC_ERROR			参考 SMC_Error

（3）功能说明

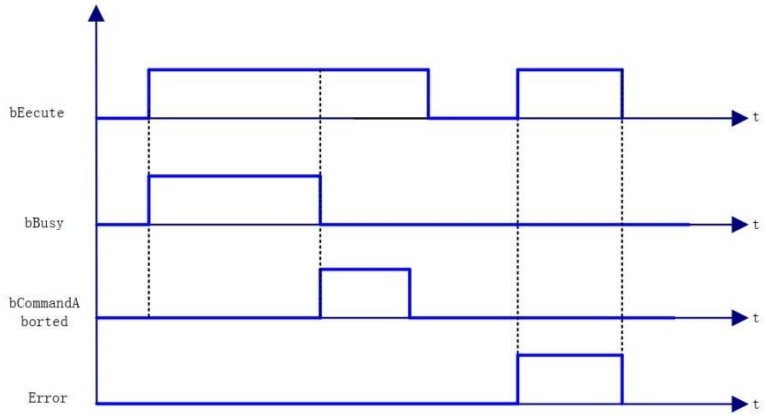
SMC_FollowSetValues 通过 bExecute 的上升沿启动之后，轴会每个任务周期给轴发送选好的参数指令。

bBusy 信号来时会轴的状态为同步运行与 MC_CamIn 指令生效时从轴状态一样，可以用MC_CamOut 指令清除。

bExecute 信号为 TRUE 时，当有其他控制命令中断该指令则 bBusy 由 TRUE 变为FALSE。

通过 DwValueMask 值来选择控制参数，比如 DwValueMask 为 1，则为每个任务周期发送位置与 SMC_FollowPosition 指令功能一样。 DwValueMask 为 2 则为单独速度指令输出。 DwValueMask 为 3，则为位置速度指令输出。DwValueMask 为 7，则为位置、速度、加速度指令输出，等等。

时序图



（4）错误说明

bExecute 上升沿时：

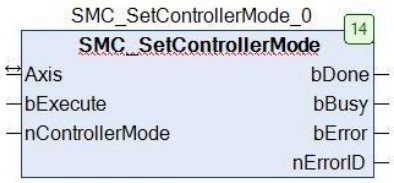
Axis 变量连接的为非 AXIS_REF_SM3 类型结构变量， Error 输出轴没使能，Error 输出。

指令运行中，轴出错， Error 输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.17 SMC_SetControllerMode

设定伺服当前运行模式，默认为同步周期位置控制，控制模式相关设置请参考伺服控制模式。

指令	名称	图形表现	ST 表 现
SMC_SetControllerMode	设定轴控制模式		SMC_SetControllerMode0(Axis:= , bExecute:= , nControllerMode:= , bDone=> , bBusy=> , bError=> , nErrorID=>);

（1）指令格式

相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

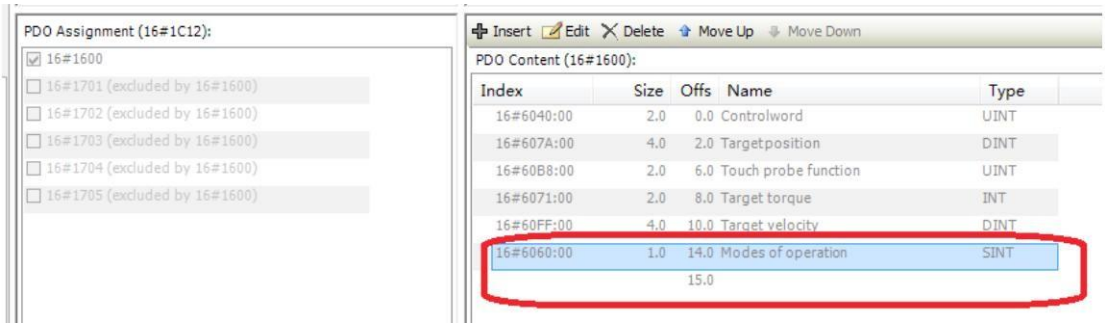
输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE,FALSE	FALSE	上升沿执行功能块
nController Mode	控制模式	SMC_CONTROLLER_MODE		SMC_Position	轴控制模式 1：转矩控制模式，SMC_torque 2：速度控制模式，SMC_Velocity 3：位置控制模式，SMC_Position 4: 电流控制模式，SMC_Current

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	模式设定完成	BOOL	TRUE,FALSE	FALSE	True, 模式设定完成
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True- 指令执行中，
bError	错误	BOOL	TRUE,FALSE	FALSE	True, 异常产生
iErrorID	错误代码	SMC_ERROR			参考 SMC_Error

（3）功能说明

SMC_SetControllerMode，通过 bExecute 的上升沿启动之后，给伺服驱动器控制模式指令，也可通过轴配置后 Axis.out.byModesofOpreation 值来设定控制模式（需在过程数据中添加对象字典 6060h）。



功能块使用需满足条件：

1：轴必须满足这些控制条件，比如虚轴不能使用该功能块。

2：各模式支持的同步周期必须保证一致（参考威科达《EtherCAT 总线伺服》说明书手册）

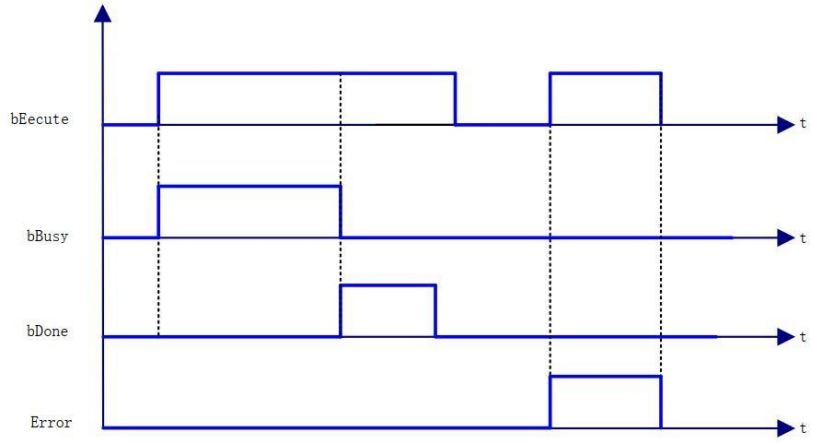
3：执行指令时轴必须在非“errorstop”，“stopping”，“homing”状态，否则会产生错误。

如果指令执行 1000 个任务周期后，轴仍然没有变为设定控制方式，则指令报错，bError 由 false 变为 true 。

当轴的控制方式由低级向高级转变时(torque -> velocity, torque->position, velocity->position)，功能块会计算高级方式的设定值，例如当有转矩模式变为位置模式时，功能块会根据当前轴实际位置叠加一个预期位置距离（通过当前实际的速度和任务周期内的时间偏移来计算），来补偿实际和设定值之间的时间滞后。

指令执行后，当轴的实际控制方式变为设定控制方式 ,bDone 信号触发，在指令触发与bDone 信号触发之间的时间内轴仍然会运行，并且在这段时间内功能块会按照设定控制方式计算合适的设定值，但是如果一旦 bDone 信号触发而没有其他的控制指令继续给轴设定值， 则轴会立即停止并报错， 因此需使用 bDone 信号上升沿来触发 MC_Halt, MC_MoveVelocity or MC_MoveAbsolute 等指令来平滑控制轴。

时序图



(5)错误说明bExecute 上升沿时： 轴无效

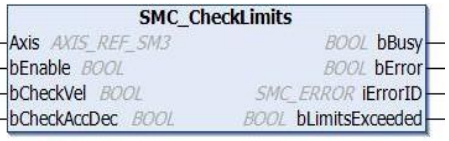
轴状态无效。

轴不满足控制方式。 轴报错， Error 输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.18 SMC_CheckLimits

该指令功能为：检查当前驱动器设置值是否超出控制器配置的最大值。 (1)指令格式

指令	名称	图形表现	ST 表 现
SMC_ CheckLimits	轴限制检查		SMC_CheckLimits0(Axis:= , bEnable:= , bCheckVel:= , bCheckAccDec:= , bBusy=> , bError=> , iErrorID=> , bLimitsExceeded=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	执行	BOOL	TRUE, FALSE	FALSE	TRUE: 执行检查中
bCheckVel	速度检查	BOOL	TRUE, FALSE	FALSE	TRUE: 执行速度检查，false: 不执行速度检查
bCheckAccDec	加减速检查	BOOL	TRUE, FALSE	FALSE	TRUE: 执行加减速检查，false: 不执行加减速检查

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True- 执行轴检查， False: 不执行轴检查
bError	错误	BOOL	TRUE,FALSE	FALSE	True, 异常产生
iErrorID	错误代码	SMC_ERROR			参考 SMC_Error
bLimits Exceeded	检查限制输出	BOOL	TRUE,FALSE	FALSE	TRUE: 当前设定速度，或加减速超过 Axis.fSWMaxVelocity， Axis.fSWMaxAcceleration Axis.fSWMaxDeceleration

(3)功能说明

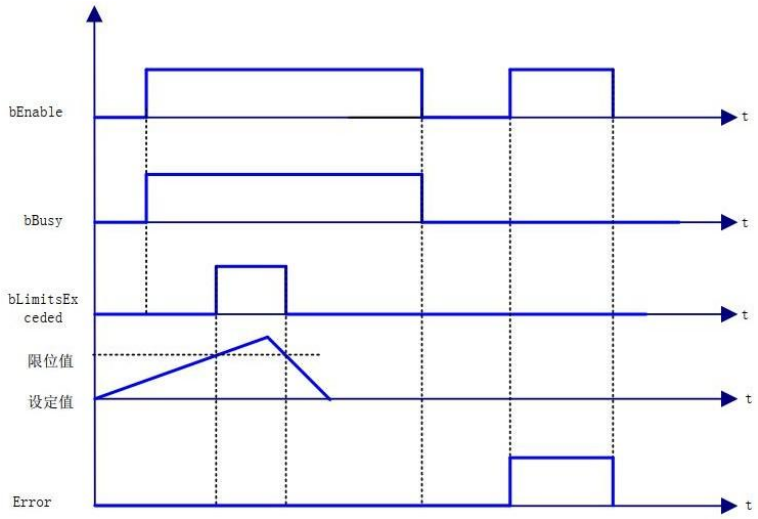
bEnable 为 TRUE，bBusy 输出 TRUE。

执行轴速度、加速度检查。

当前轴的设定速度或者加减速度超过 Axis.fSWMaxVelocity 、Axis.fSWMaxAcceleration、Axis.fSWMaxDeceleration 设定值， bLimitsExceeded 信号输出为 TRUE

注意:该功能只是检查当前的指令速度或加减速超过设置的限值,并不能停止轴。

(3)时序图



(4) 错误说明bExecute 上升沿时： 轴报错， Error 输出。

无效的轴输入， Error 输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.19 SMC_GetMaxSetAccDec

该指令功能为：读取轴最大加减速度。(1) 指令格式

指令	名称	图形表现	ST表现
SMC_ GetMaxSetAccDec	轴 度 最大 加 减速		SMC_GetMaxSetAccDec_0(Axis:= , bEnable:= , dwTimeStamp:= , bValid=> , bBusy=> , fMaxAcceleration=> , dwTimeAtMax=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	执行	BOOL	TRUE,FALSE	FALSE	TRUE: 执行读取
dwTimeStamp		Dword			可选的时间戳输入；可以用来查找最大值时发生的情况

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
			TRUE,		

bValid	有效	BOOL	FALSE	FALSE	True，指令执行有效
bBusy	执行中	BOOL	TRUE，FALSE	FALSE	True，正在读取数据
fMaxAcceleration	最大加减速值	LREAL		0	最大的加减速值（正为加速，负为减速，加减速绝对值最大值为最终值）
dwTimeAtMax	最大值对应时间戳	Dword		0	最大加减速时对应的dwTimeStamp值（例如加速度持续增加时，该值更随dwTimeStamp，fMaxAcceleration值也更新，一旦加速度达到最大值，则fMaxAcceleration记录最大值，同时最大值对应的dwTimeStamp也被记录）

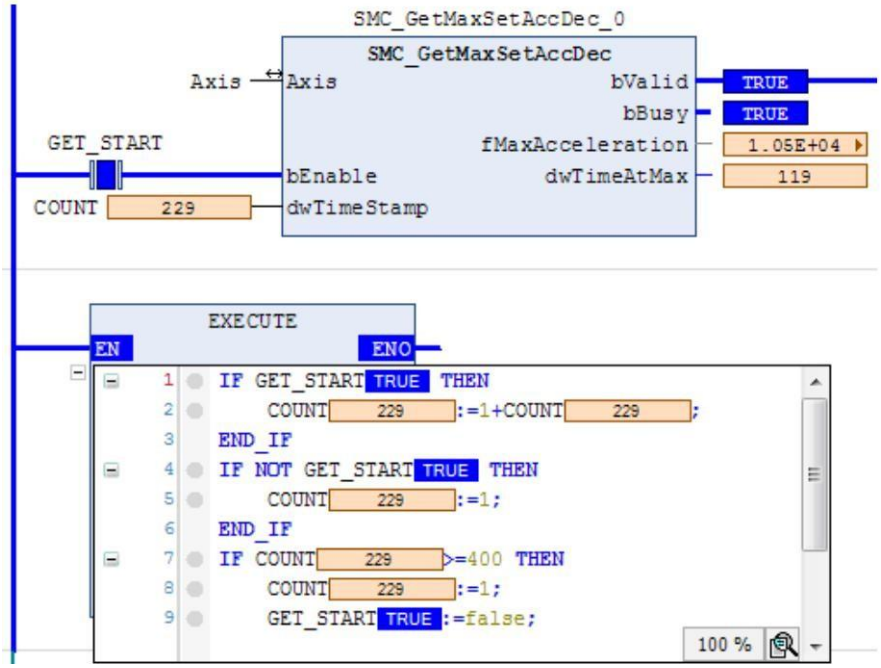
(3)功能说明

bEnable为TRUE，无错误，bValid输出TRUE。执行轴最大加减速检测。

当加减速绝对值大于前面记录值时，fMaxAcceleration跟dwTimeAtMax会刷新。

dwTimeAtMax值为最大加减速时对应dwTimeStamp值，所以dwTimeStamp应设置为可变的，比如设置为随着任务周期或者固定时间周期的计数值。（见样例程序）

样列程序


SMC_GetMaxSetAccDec_0
SMC_GetMaxSetAccDec
Axis
Axis
bValid
TRUE
bBusy
TRUE
GET_START
fMaxAcceleration
1.05E+04
bEnable
dwTimeAtMax
119
COUNT
229
dwTimeStamp
EXECUTE
EN
ENO
1 IF GET_START TRUE THEN
2 COUNT 229 :=1+COUNT 229 ;
3 END_IF
4 IF NOT GET_START TRUE THEN
5 COUNT 229 :=1;
6 END_IF
7 IF COUNT 229 >=400 THEN
8 COUNT 229 :=1;
9 GET_START TRUE :=false;
100 %

6.4.20 SMC_GetMaxSetVelocity

该指令功能为：读取轴最大速度。(1) 指令格式

指令	名称	图形表现	ST 表现
SMC_GetMaxSetVelocity	轴度最大加减速		SMC_GetMaxSetVelocity_0(Axis:= , bEnable:= , dwTimeStamp:= , bValid=> , bBusy=> , fMaxVelocity=> , dwTimeAtMax=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	执行	BOOL	TRUE,FALSE	FALSE	TRUE: 执行读取
dwTimeStamp		Dword			可选的时间戳输入；可以用来查找最大值时发生的情况。

输出变量

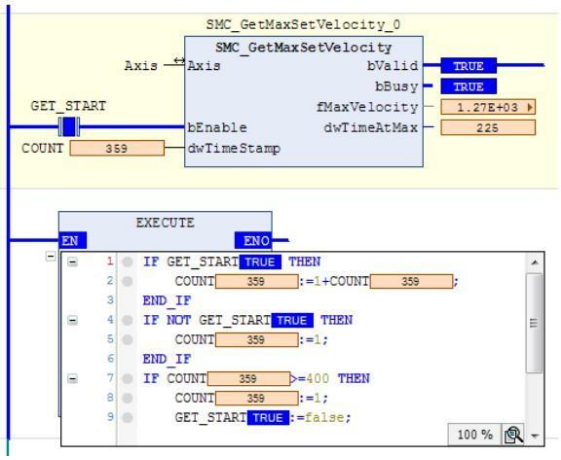
输出变量	名称	数据类型	有效范围	初始值	描述
bValid	有效	BOOL	TRUE,FALSE	FALSE	True，指令执行有效
bBusy	执行中	BOOL	TRUE,FA	FALSE	True，正在读取数据

			LSE		
fMaxVelocity	最大速度值	LREAL		0	最大的速度值（正为正向，负为反向，绝对值最大值为最终值）
dwTimeAtMax	最大值对应时间戳	Dword		0	最大速度时对应的dwTimeStamp 值（例如速度持续增加时，该值更随 dwTimeStamp，fMaxVelocity 值也更新，一旦速度达到最大值，则fMaxVelocity 记录最大值，同时最大值对应的dwTimeStamp 也被记录）

(2)功 能 说 明

bEnable 为 TRUE，无错误， bValid 输出 TRUE。执行轴最大加减速检测。当速度绝对值大于前面记录值时， fMaxVelocity 跟 dwTimeAtMax 会刷新。dwTimeAtMax 值为最大速度时对应 dwTimeStamp 值，所以 dwTimeStamp 应设置为可变的，比如设置为随着任务周期或者固定时间周期的计数值。（见样例程序）

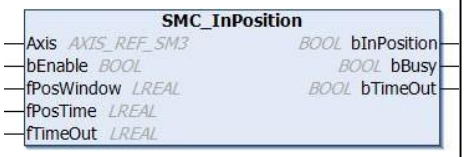
样列程序



6.4.21 SMC_InPosition

该指令功能为：监控当前轴设定位置值跟实际值之间的偏差，通过设定的偏差窗口来确定轴是否在要求的偏差范围内。

(1)指令格式

指令	名称	图形表现	ST 表现
SMC_InPosition	轴偏差监控		SMC_InPosition0(Axis:=Axis , bEnable:= , fPosWindow:= , fPosTime:= , fTimeOut:= , bInPosition=> , bBusy=> , bTimeOut=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	执行	BOOL	TRUE, FALSE	FALSE	TRUE: 执行读取
fPosWindow	偏差窗口	LREAL		0	设定偏差监控的窗口，fPosWindow>Distance（指令位置跟反馈位置间偏差），则根据 fPosTime 时间输出 bInPosition 为 TRUE
	触发				偏差在窗口范围内时间，用来触发

fPosTime	时间	LREAL		0	bInPosition 单位为 S（秒）
fPosTiOut	超时时间	LREAL		0	偏差超时单位为 S（秒）

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bInPosition	偏差正常	BOOL	TRUE,FALSE	FALSE	True，偏差在设置窗口范围内
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True，执行中
bTimeOut	超时	LREAL	TRUE,FALSE	FALSE	跟 byDeaTimeCycles 值相关的当前偏差检测

(3)功 能 说 明

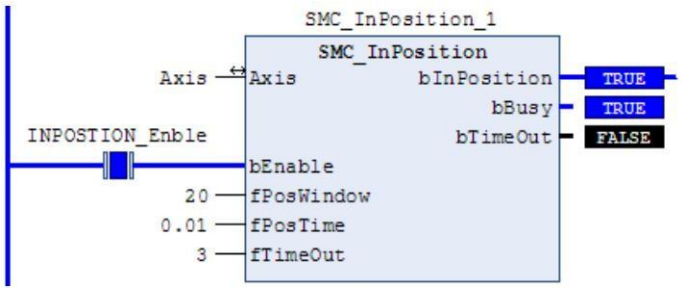
bEnable 为 TRUE，一旦检测到的偏差小于设置的窗口 fPosWindow 持续 fPosTime 秒则bInPosition 触发为 TRUE。一旦检测到的偏差大于设置的窗口，bInPosition 立马输出为FALSE。

注意： fPosTime 时间设定需合理否则会造成 bTimeOut 触发（比如一个凸轮周期为 2 秒的凸轮，连续偏差没有超过设定窗口的时间为 1.5 秒， fPosTime 如果设定的大于 1.5 秒则 会 造 成bInPosition不 会 触 发 ）。 bEnable 为 TRUE， bBusy 输 出 为 true 。

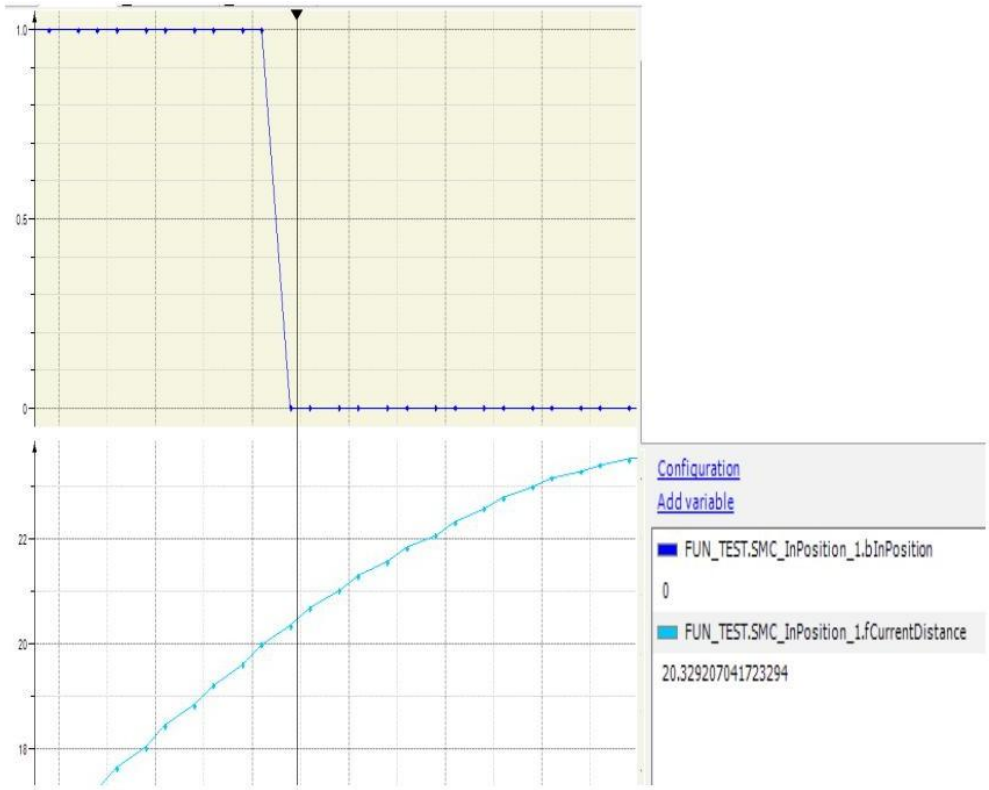
偏差值可监控 SMC_InPosition 结构体中的数据 fCurrentDistance。

bEnable 为 TRUE ， 超过 fPosTime 设定时间 bInPosition 仍然没有触发为 TRUE, 则bTimeOut 触发为TRUE。

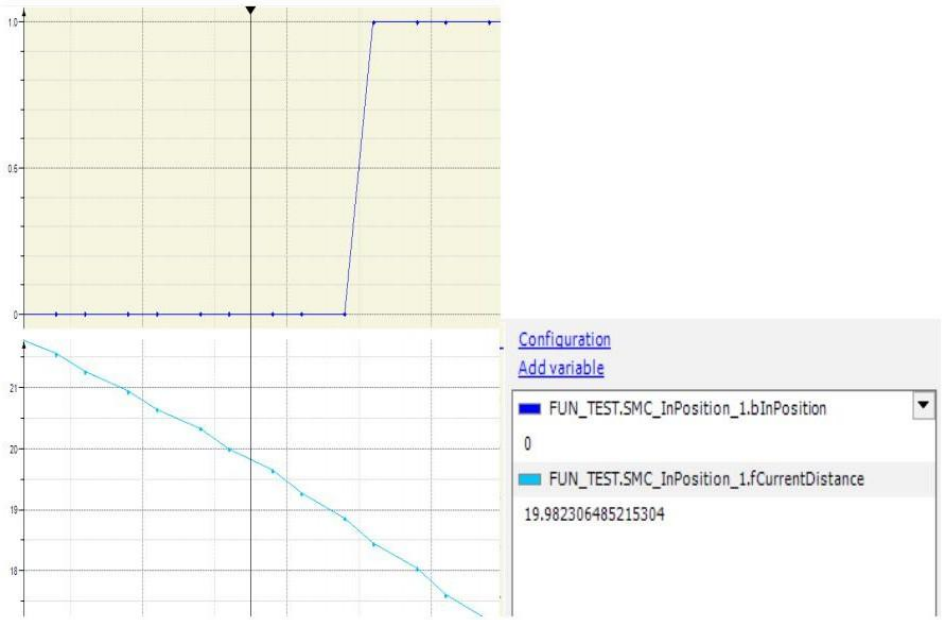
时序图样列程序



样例程序

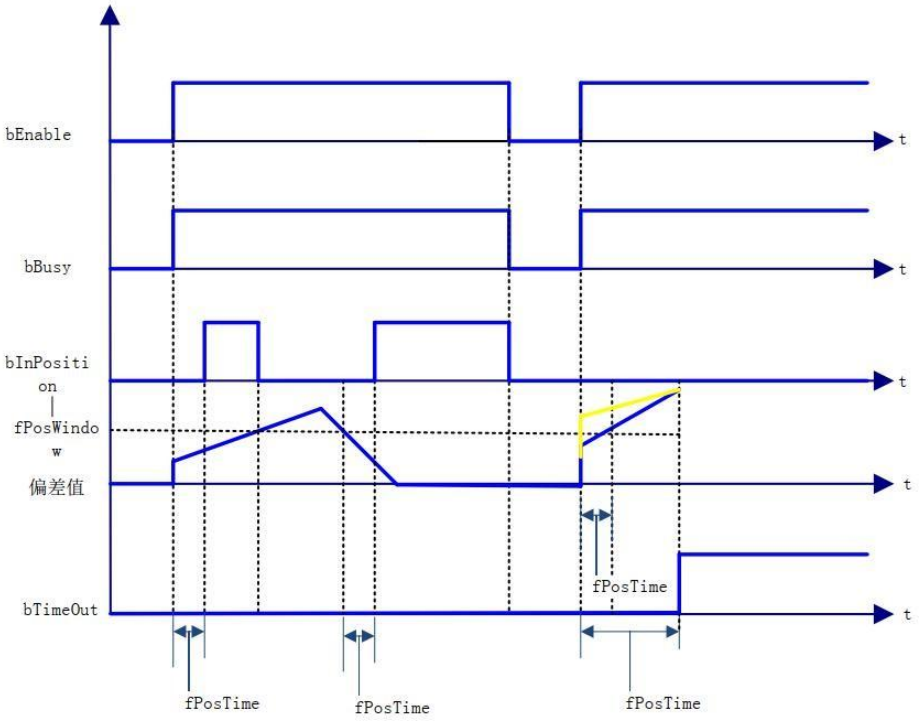


大于窗口设定 bInPosition 立即由 true 变为 FALSE



在设定窗口以内 4 个任务周期（2.5ms）后 bInPosition 变为 TRUE，跟程序设置 0.01S 相符

时序图



6.4.22 SMC_ReadSetPosition

该指令功能为：读取轴的指令位置（转换过后的用户单位）。 (1)指令格式

指令	名称	图形表现	ST 表现
SMC_ ReadSetPosition	读轴指令位置		SMC_ReadSetPosition0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Position=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	执行	BOOL	TRUE,FALSE	FALSE	TRUE: 执行读取

输出变量

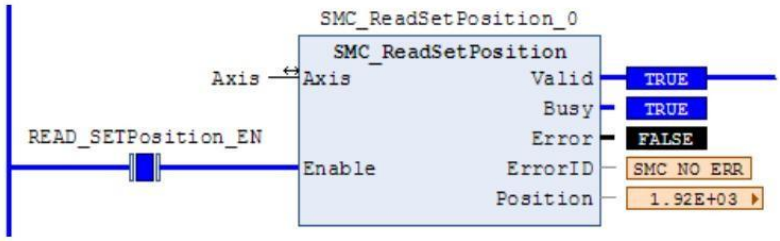
输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效	BOOL	TRUE,FALSE	FALSE	True，读取有效
Busy	执行中	BOOL	TRUE,FALSE	FALSE	True，执行中
Error	错误	BOOL	TRUE,FALSE	FALSE	True，异常产生
ErrorID	错误代码	SMC_ERROR			参考 SMC_Error
Position	指令位置	LREAL		0	当前任务周期的指令位置

(3)功 能 说 明

Enable 为 TRUE，无错误则 Valid， Busy 输出为 TURE。

Position输 出 的 值 为Axis.fSetPosition的值 。Enable 变 为 FALSE， ，则 Valid， Busy 输 出 为 FALSE 。 Position 停 留 在 FALSE 之 前 的值。

时序图样列程序



(4)错 误 说 明

bExecute 上升沿时：轴报错， Error 输出；无效的轴输入， Error 输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.23 SMC_SetTorque

该指令功能为：设定轴转矩（转矩控制模式时有效）。 (1)指令格式

指令	名称	图形表现	ST 表 现
SMC_SetTorque	力矩设定		SMC_SetTorque0(Axis:= , bEnable:= , fTorque:= , bBusy=> , bError=> , nErrorID=>);

(2)相关变量输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	执行	BOOL	TRUE,FALSE	FALSE	上升沿，设定轴力矩
fTorque	设定的力矩	LREAL		0	单位为 0.1‰

输出变量

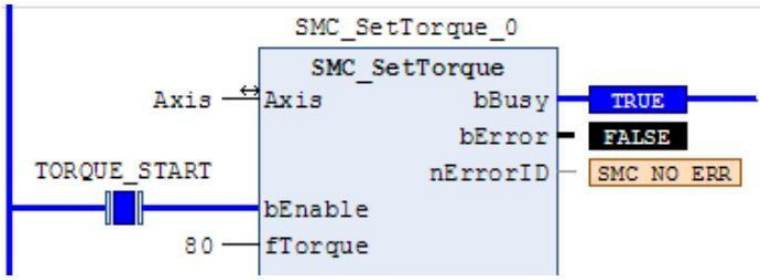
输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行中	BOOL	TRUE,FALSE	FALSE	True，执行中
Error	错误	BOOL	TRUE,FALSE	FALSE	True，异常产生
ErrorID	错误代码	SMC_ERROR			参考 SMC_Error

(3)功 能 说 明

bEnable 上升沿，无错误则 bBusy 输出为 TURE。

该指令只是给轴设定转矩值用并不是转矩控制功能，轴控制模式在转矩控制模式下有效，即需要先使用 SMC_SetControllerMode 指令，将伺服设定为转矩模式，再执行该指令。

时序图样列程序



(4)错误说明

bExecute 上升沿时：

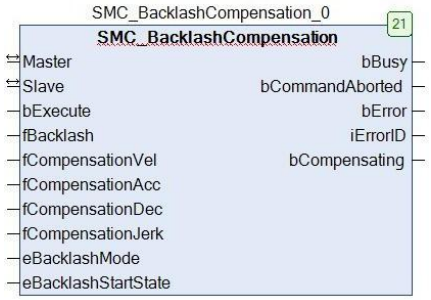
轴报错， Error 输出；无效的轴输入， Error 输出。轴控制模式错误， Error 输出，错误代码SMC_ST_WRONG_CONTROLLER_MODE

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.24 SMC_BacklashCompensation

该指令功能为：用来补偿主从轴间隙，比如说皮带传送中虚轴为主轴，从轴为虚轴同步镜像，由于外部原因导致从轴位置跟主轴存在间隙，可用该指令来补偿这种间隙。

该指令功能与相位偏移指令（MC_Phasing）类似，其相位取决于主轴运行的方向。(1) 指令格式

指令	名称	图形表现	ST 表现
SMC_BacklashCompensation	间隙补偿		SMC_BacklashCompensation0(Master:= , Slave:= , bExecute:= , fBacklash:= , fCompensationVel:= , fCompensationAcc:= , fCompensationDec:= , eBacklashMode:= , eBacklashStartState:= , bBusy=> , bCommandAborted=> , bError=> , iErrorID=> , bCompensating=>);

(3)相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例
Slave	从轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE, FALSE	FALSE	上升沿，设定偏移

fBacklash	LREAL	0			补偿间隙
fCompensationVel	LREAL	0			补偿时速度
fCompensationAcc	LREAL	0			补偿时加速度
fCompensationDec	LREAL	0			补偿时减速度
eBacklashMode	SMC_BACKLASH_MODE	SMC_BL_AUTO			补偿模式：SMC_BL_AUTO: 主轴运行方向决定补偿方向 SMC_BL_POSITIVE: 正向补偿，独立于于主轴运行方向 SMC_BL_NEGATIVE: 反向补偿，独

					立于主轴运行方向 SMC_BL_OFF: 不补偿
eBacklash StartState	SMC_ BACKLASH_ STARTSTATE	SMC_BL_ START_ NEGATIVE			描述该指令工作时轴的工作状态。 SMC_BL_START_NEGATIVE: 从轴在负方向牵引下运动，在负方向运动下不需要补偿，一旦正向运动以两倍 fBacklash 建立补偿 SMC_BL_START_POSITIVE: 从轴在正方向牵引下运动，在正方向运动下不需要补偿，一旦反向运动需要以两倍 fBacklash 建立补偿 SMC_BL_START_NONE: 在正的或反的方向运动会产生 fBacklash 值的距离补偿。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True，执行中

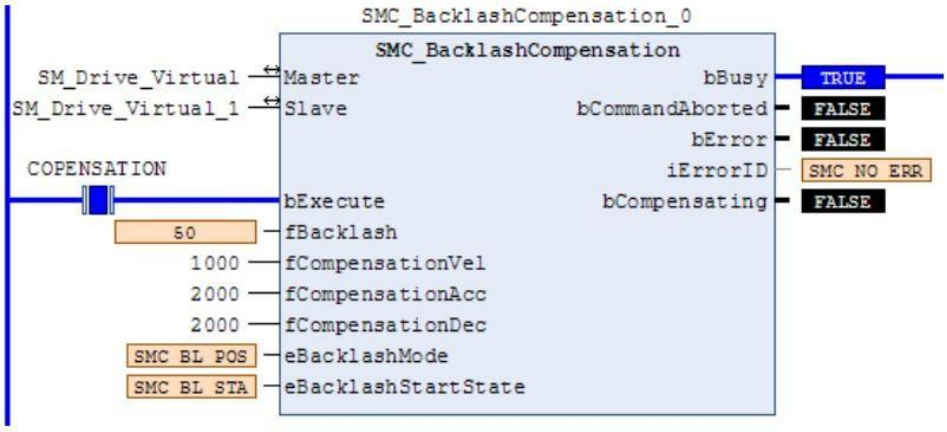
bCommandAborted	指令被中断	BOOL	TRUE,FALSE	FALSE	True- 被其他控制命令打断
bError	错误	BOOL	TRUE,FALSE	FALSE	True，异常产生
iErrorID	错误代码	SMC_ERROR	参考SMC_Error		
bCompsating	补偿中	BOOL	TRUE,FALSE	FALSE	

(3)功 能 说 明

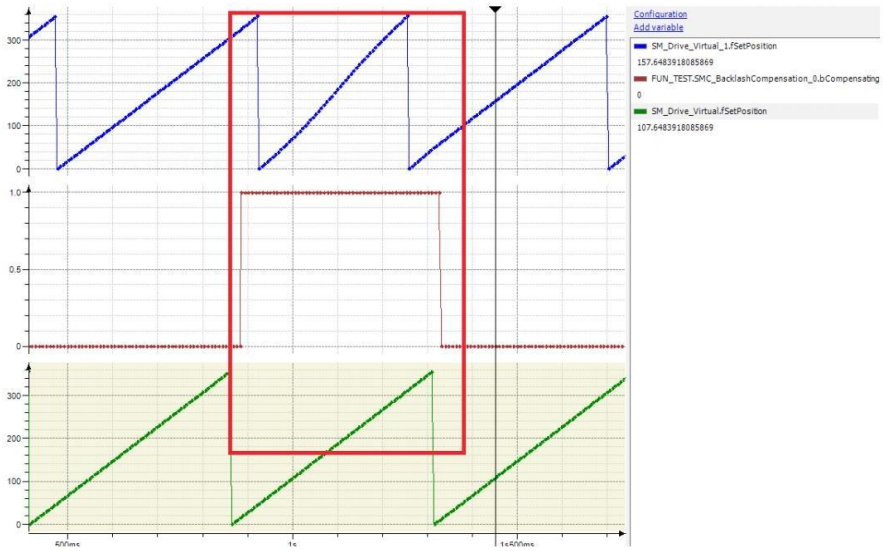
bEecute 上升沿，无错误则， bBusy 输出为 TURE， bCompsating 输出为 true，补偿完成后 bCompsating 输出为 false。

工作方式为：eBacklashMode- 补偿方向为“正”，eBacklashStartState 为“正”，fBacklash 为正值。在 bBusy 信号没来之前，最好主从轴位置一致否则 bEecute 上升沿来后，从轴会调整到主轴相位同步， bBusy 信号已经有的情况再刷新 bEecute 上升沿请遵守：

时序图样列程序



样例程序



(4)错误说明bExecute 上升沿时：

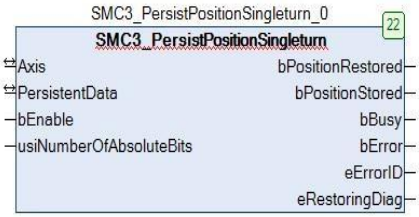
轴报错， Error 输出；无效的轴输入， Error 输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.25 SMC3_PersistPositionSingleturn

该指令用来保持记录实轴单圈绝对值编码器的位置（断电重启控制器后，恢复断电前位置记　　录值）。

(1)指令格式

指令	名称	图形表现	ST 表 现
SMC3_ PersistPositi on Singleturn	轴 位 置 保持		SMC3_PersistPositionSingleturn_0(Axis:= , PersistentData:= , bEnable:= , usiNumberOfAbsoluteBits:= , bPositionRestored=> , bPositionStored=> , bBusy=> , bError=> , eErrorID=> , eRestoringDiag=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例
PersistentDat a	保 持 数据	SMC3_ PersistPositio n Singletrun_D ata			映射到记录位置结构，为 SMC3_ PersistPosition_Data 的一个映射（该结构变量必须为断电保持型）

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述

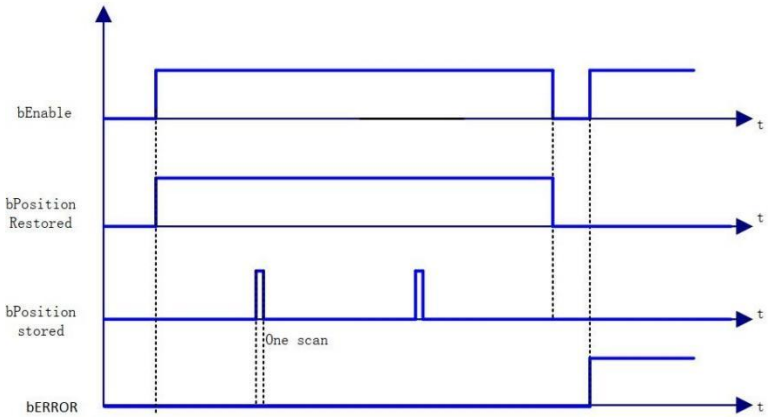
bEnable	执行	BOOL	TRUE, FALSE	FALSE	True 功能块执行，false 不执行功能块 PLC 重启后需要为 true 才能恢复重启前存储的位置。
usiNumberof AbsoluteBites	位数	UINT		16	多少位的绝对值编码器（如 20 位， 24 位 编码器等等）

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bPositionRestored	位置恢复	BOOL	TRUE, FALSE	FALSE	TRUE，轴重启后位置恢 复完成
bPositionStored	位置保存	BOOL	TRUE, FALSE	FALSE	TRUE，调用功能快后保 存位置完成
bBusy	FB 执行中	BOOL	TRUE, FALSE	FALSE	TRUE，功能块没有执行 完成
bError	错误	BOOL	TRUE, FALSE	FALSE	TRUE，异常发生
eErrorID	错误代码	SMC_ERROR		SMC_NO_	异常发生时，输出错误代

				ERROR	码
eRestoringDiag	恢复诊断	SMC3_PersistPositionDiag		SMC3_PersistPositionDiag.SMC3_PPD_RESTORING_OK	位置恢复中的诊断信息

功能说明PLC 重启 bEnable 信号为 TRUE，则 bPositionRestroed 输出为 TRUE。不支持虚轴跟逻辑轴。



时序图

错误说明

输入轴为虚拟轴或者逻辑轴会导致错误输出。 轴有错误。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.26 SMC_CheckAxisCommunication

该指令功能为：检查当前驱动器通讯状态。(1) 指令格式

指令	名称	图形表现	ST 表 现
SMC_ CheckLimits	轴 限 制 检查		eErrorID=> , bOperational=> , eComState=> , wComState=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	执行	BOOL	TRUE,FALSE	FALSE	TRUE: 执行检查中

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bValid	执 行 中	BOOL	TRUE, FALSE	FALSE	True, 指令执行有效
bError	错误	BOOL	TRUE, FALSE	FALSE	True, 异常产生
eErrorID	错 误 代 码	SMC_ERRO R			参 考 SMC_Error

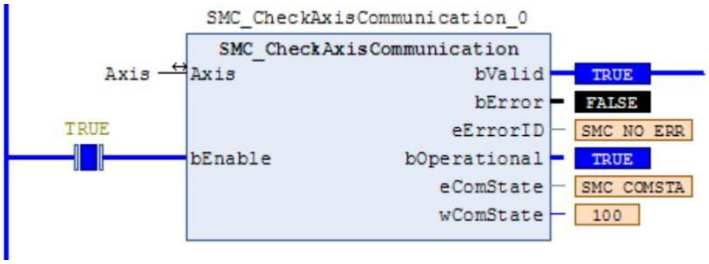
bOperational	通讯正常	BOOL	TRUE, FALSE	FALSE	True，通讯正常（代码为 100）可操作 False，通讯不正常，不可对轴操作
eComState	通讯状态	SMC_COMMUNICATIONSTATE			包含： SMC_COMSTATE_NOT_STARTED，通讯没有启动 SMC_COMSTATE_VARIABLE_INITIALIZATION，通讯变量初始化 SMC_COMSTATE_BASE_COM_INITIALIZATION，基本端口初始化 SMC_COMSTATE_DRIVE_INITIALIZATION，通讯驱动初始化 SMC_COMSTATE_DRIVE_WAITING_FOR_SYNC，同步警告 SMC_COMSTATE_INITIALIZATION_DONE，初始化完成

					SMC_COMSTATE_OPERATIONAL, 通讯可正常使用 SMC_COMSTATE_REINITIALIZATION, 通讯重新初始化 SMC_COMSTATE_ERROR, 通讯错误 SMC_COMSTATE_UNKNOWN 通讯状态不知
wComState	通讯代码	WORD			与输入输出轴结构体变量中：Axis.wCommunicationState 值相同，表示当前通讯状态的代码，参考 AXIS_REF_SM3 参数 1013

(3)功能说明

bEnable为TRUE，无错误，bValid输出TRUE。执行轴通讯状态检查。bValid输出TRUE时检查轴通讯状态，当eComState输出为SMC_COMSTATE_OPERATIONAL时，bOperational输出为TRUE。

样例程序



(4)错误说明

bExecute 上升沿时：轴报错， Error 输出； 无效的轴输入， Error 输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.27 SMC_FollowPosition

指令功能为不做任何检查直接给轴设定位置。该指令与 MC_MoveAbsolute 有所不同，执行上升沿型号来后，每个任务周期都会给轴位置指令而不管轴的状态。（用户可用该指令写凸轮功能，而不使用 MC_CamIn 等指令）。

(1)指令格式

指令	名称	图形表现	ST 表 现
SMC_FollowPosition	轴 位置给定		SMC_FollowPosition_0(Axis:= , bExecute:= , fSetPosition:=SET_POSITION , bBusy=> , bCommandAborted=> , bError=> , iErrorID=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE,FALSE	FALSE	上升沿执行功能块
fSetPosition	设定位置	LREAL		0	轴设定的位置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

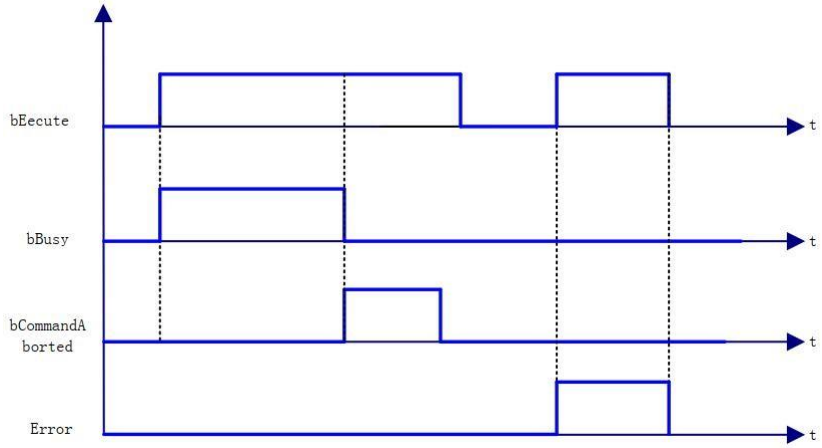
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True- 指令执行中，此时轴处于同步状态，与凸轮 MC_CamIn 指令运行时轴状态一样，可以用 MC_Camout 指令清除 bBusy 状态
bCommand Aborted	指令被中断	BOOL	TRUE,FALSE	FALSE	True- 轴被其他控制命令打断
bError	错误	BOOL	TRUE,FALSE	FALSE	True, 异常产生
iErrorID	错误代码	SMC_ERROR			参考 SMC_Error

(3)功能说明

SMC_FollowPosition 通过 bExecute 的上升沿启动之后，轴会每个任务周期给轴发送位置指令。bBusy 信号来时会轴的状态为同步运行与 MC_CamIn 指令生效时从轴状态一样， 可以用 MC_CamOut 指令清除。

轴的的速度 - 由轴两个任务周期相差的位置增量自己计算出来，速度： ΔL/ Δt, ΔL 本任务周期fSetVelocity 跟上个任务周期 fSetVelocity 差值、Δt 为扫描时间。bExecute 信号为 TRUE 时，当有其他控制命令中断该指令则 bBusy 由 TRUE 变为 FALSE。

时序图



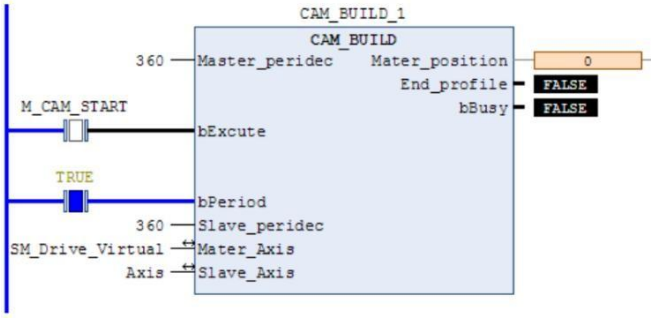
(4)错 误 说 明

bExecute 上 升 沿 时 ： Axis 变量连接的为非AXIS_REF_SM3 类型结构变量， Error 输出； 轴没使能， Error 输出。

指令运行中，轴出错， Error 输出

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

(5)样例说明



使用 SMC_FollowPosition 实现电子凸轮功能。

功 能 块 变 量 定 义 部 分 ： FUNCTION_BLOCK CAM_BUILD

VAR_INPUT// 输入变量定义Master_peridec:REAL; // 主轴周期bExcute:BOOL; // 指 令 执 行 bPeriod:BOOL; // 凸 轮 周 期 执 行 ， false 单 周 期 执 行 Slave_peridec:REAL; // 从轴周期END_VAR

VAR_OUTPUT// 输出变量定义

Mater_position:LREAL;// 主 轴 位 置 （ 指 令 执 行 开 始 后 计 算 的 主 轴 位 置 ）

End_profile:BOOL; // 曲线完成输出标志位

bBusy:BOOL; // 执行中END_VAR

VAR// 功能块中间变量定义

SMC_FollowPosition_0: SMC_FollowPosition; SET_POSITION: LREAL;

SET_POSITIONOLD: LREAL;

Mater_positionOLD:LREAL; bExcute_old:BOOL; INC:LREAL;

Y:LREAL; X5:LREAL; X4:LREAL; X3:LREAL; X2:LREAL; X1:LREAL;

MC_Stop0: MC_Stop; STOP:BOOL; COUNTNUM:DINT; SET_INC:LREAL;

YOLD:LREAL;

SMC_FollowPositionVelocity_0: SMC_FollowPositionVelocity; K:REAL;

K_OUT:REAL;

MC_CamOut_0: MC_CamOut; END_VAR

VAR_IN_OUT// 输入输出变量定义 Mater_Axis:AXIS_REF_SM3;

Slave_Axis:AXIS_REF_SM3;

END_VAR

程序部分：

IF bExcute AND NOT bExcute_old THEN // 上升沿初始化参数Mater_position:=0;

Mater_positionOLD:=Mater_Axis.fActPosition; End_profile:=FALSE;

SET_POSITION:=Slave_Axis.fActPosition;

SET_POSITIONOLD:=Slave_Axis.fActPosition; COUNTNUM:=0;

YOLD:=0; K:=0; ELSE

IF bExcute_old THEN

INC:=Mater_Axis.fActPosition-Mater_positionOLD;// 主轴任务周期增量

IF INC<0 THEN // 主轴编码位置过零点（轴设置为 modulo- 模数模式时）

INC:=Mater_Axis.fActPosition-Mater_positionOLD+Mater_Axis.fPositionPeriod;

END_IF

Mater_position:=INC+Mater_position;// 当前主轴位置

```
Mater_positionOLD:=Mater_Axis.fActPosition;
//******** 判断曲线完成 *********//
IF Mater_position>=Master_peridec THEN End_profile:=TRUE;
ELSE
End_profile:=FALSE; END_IF
IF bPeriod THEN
IF Mater_position>=Master_peridec THEN
Mater_position:=Mater_position-Master_peridec; END_IF
END_IF END_IF END_IF
IF bExcute_old THEN X1:=(Mater_position/Master_peridec); X2:=X1*X1;
X3:=X2*X1; X4:=X3*X1; X5:=X4*X1;
Y:=(6*X5-15*X4+10*X3)*Slave_peridec;// 从 轴 位 置 , 曲 线
K:=(30*X4-60*X3+30*X2)*Slave_peridec/Master_peridec;// 曲 线 斜 率
SET_INC:=Y-YOLD;
IF SET_INC<0 THEN
SET_INC:=Slave_peridec-YOLD+Y; END_IF
YOLD:=Y;
IF bPeriod THEN SET_POSITION:=SET_POSITION+SET_INC; ELSE
IF End_profile THEN SET_POSITION:=SET_POSITIONOLD+Slave_peridec;
ELSE
SET_POSITION:=SET_POSITION+SET_INC; END_IF
END_IF
IF SET_POSITION>=Slave_Axis.fPositionPeriod THEN
SET_POSITION:=SET_POSITION-Slave_Axis.fPositionPeriod; END_IF
END_IF
SMC_FollowPosition_0( Axis:=Slave_Axis, bExecute:=bExcute,
fSetPosition:=SET_POSITION ,
```

```
bBusy=>bBusy , bCommandAborted=> , bError=> ,
iErrorID=> );
IF NOT bExcute AND bExcute_old THEN STOP:=TRUE;
END_IF
MC_CamOut_0( Slave:=Slave_Axis, Execute:= STOP, Done=> , Busy=> ,
Error=> , ErrorID=> ); MC_Stop0( Axis:=Slave_Axis,
Execute:= MC_CamOut_0.Done OR MC_CamOut_0.Error , Deceleration:=20000 ,
Jerk:= 20000,
Done=> , Busy=> , Error=> , ErrorID=> );
IF MC_Stop0.Done OR MC_Stop0.Error THEN STOP:=FALSE;
END_IF
IF NOTbExcute_old THEN End_profile:=FALSE; END_IF
bExcute_old:=bExcute;
```

6.4.28 SMC_FollowPositionVelocity

该指令功能跟 SMC_FollowPosition 使用跟功能都一样，但是增加了速度设定。

注意：速度设定要满足位置设定变化即：速度设定 = 相隔任务周期位置设定差值对时间的一次导数。比如：

两个相隔周期位置设定一致，则速度应该设为 0，否则会造成电机剧烈振动。

(1)指令格式

指令	名称	图形表现	ST 表 现
SMC_ FollowPositionVelocity	轴 位 置、速 度 给 定		SMC_FollowPositionVelocity_0(Axis:= , bExecute:= , fSetPosition:= , fSetVelocity:= , bBusy=> bBusy, bCommandAborted=> , bError=> , iErrorID=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE,FALSE	FALSE	上升沿执行功能块
fSetPosition	设定位置	LREAL		0	轴设定的位置
fSetVelocity	设定速度	LREAL		0	轴设定的位置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True- 指令执行中，（此时轴处于同步状态，与凸轮 MC_CamIn 指令运行时轴状态一样），可以用MC_Camout 指令清除bBusy 状态
bCommandAborted	指令被中断	BOOL	TRUE,FALSE	FALSE	True- 轴被其他控制命令打断
bError	错误	BOOL	TRUE,FALSE	FALSE	True, 异常产生
iErrorID	错误代码	SMC_ERROR			参考 SMC_Error

(3)功能说明

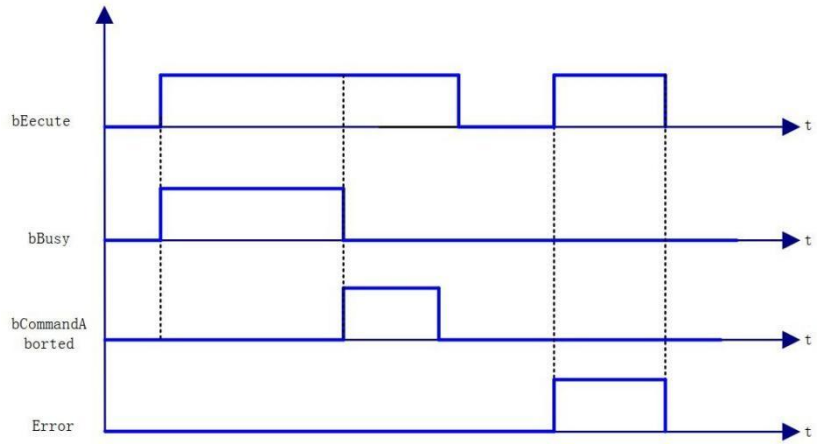
SMC_FollowPositionVelocity 通过 bExecute 的上升沿启动之后，轴会每个任务周期给轴发送设定位置、设定速度指令。

bBusy 信号来时会轴的状态为同步运行与 MC_CamIn 指令生效时从轴状态一样，可以用

MC_CamOut 指令清除。

轴的设定速度要与设定位置变化一致： fSetVelocity= ΔL/ Δt, ΔL 本任务周期 fSetVelocity 跟上个任务周期 fSetVelocity 差值、Δt 为扫描时间。bExecute 信号为 TRUE 时，当有其他控制命令中断该指令则 bBusy 由 TRUE 变为 FALSE。

时序图



(3)错 误 说 明

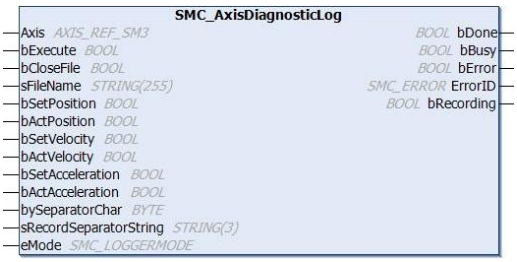
bExecute上 升 沿 时 ： Axis 变量连接的为非 AXIS_REF_SM3 类型结构变量，Error 输出； 轴没使能， Error 输出；

指令运行中，轴出错， Error 输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.29 SMC_AxisDiagnosticLog

该指令功能为：周期性的将轴的一个参数写入文件。 (1)指令格式

指令	名称	图形表现	ST 表 现
SMC_ AxisDiagnostic Log	轴 参数 写入 文件		

相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE,FALSE	FALSE	上升沿，执行功能块
bClosefile	关闭文件	BOOL	TRUE,FALSE	FALSE	TRUE，指令立马关闭文件
sFileName	文件名	STRING(80)		‘’	存储的文件名（路径之前
bSetPosition	记 录 设定 位置	BOOL	TRUE,FALSE	FALSE	TRUE，执行指令时记录设定 位置

bActPosition	记录实际位置	BOOL	TRUE,FALSE	FALSE	TRUE，执行指令时记录实际位置
bSetVelocity	记录设定速度	BOOL	TRUE,FALSE	FALSE	TRUE，执行指令时记录设定速度
bActVelocity	几轮实际速度	BOOL	TRUE,FALSE	FALSE	TRUE，执行指令时记录实际速度
bSetAcceleration	记录设定加速度	BOOL	TRUE,FALSE	FALSE	TRUE，执行指令时记录设定加速度
bActAcceleration	记录实际加速度	BOOL	TRUE,FALSE	FALSE	TRUE，执行指令时记录实际加速度
bySeparatorChar		BYTE		9	ASCII 代码值，写在两个不同值之间

sRecord SeparatorString				‘RN’	日期结束时写的字符串
eMode		SMC_ LOGGERMODE	LOG_ CONTINUOUS		log_continuous：连续记录到文件 log_at_close：连续记录到缓冲区（10kbyte）。当bclosefile为 true 将缓冲区的数据写入文件

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	完成	BOOL	TRUE,FALSE	FALSE	True，保存完成
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True，执行中
bError	错误	BOOL	TRUE,FALSE	FALSE	True，异常产生
ErrorID	错误代码	SMC_ERROR			参考 SMC_Error
bRecording	记录中	BOOL	TRUE,FALSE	FALSE	True，参数正在保存记录中

（4）功能说明

此功能块用于将属于一个轴的一组参数值循环写入文件。此输出文件非常适合用于诊断　　目的。由于在数据媒体上写入数据通常需要一段时间，此模块将收集到的数据存储在一个10kbyte 大小的缓冲区中，在调用模块动作 WriteToFile 之前，数据不会被写入。为了防止实际的动作任务和动作本身受到干扰，这个动作调用应该放在一个较慢（约 50 毫秒）的低优先级任务中。一旦超过缓冲区，模块将创建一个错误输出。

4） 错误说明

bExecute 上升沿时：轴报错， Error 输出； 无效的轴输入， Error 输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.30 SMC_ChangeGearingRatio

该指令功能为：用来改变用户设定电子齿轮比（脉冲转用户使用单位比）和驱动类型。注意：执行了功能块后轴需要通过 SMC3_ReinitDrive 来重启来保证能够正确初始化设置变量 （1）指令格式

指令	名称	图形表现	ST 表 现
SMC_ ChangeGearingRatio	改变齿轮比		SMC_ChangeGearingRatio0(Axis:= , bExecute:= , dwRatioTechUnitsDenom:= , iRatioTechUnitsNum:= , fPositionPeriod:= , iMovementType:= , bDone=> , bBusy=> , bError=> , nErrorID=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例 ．齿轮比将被改的轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE,FALSE	FALSE	上升沿，执行功能块
dwRatioTechUnitsDenom		DWORD		0	脉冲单位转换为应用单位（eg:mm）
iRatioTechUnitsNum		DINT		0	dwRatioTechUnitsDenom 值对应所需的应用单位

fPositionPeriod		LREAL			位置循环周期（模数值），只对旋转电机有效
iMovementType		INT			0: modulo axis（模数轴），1: finite axis（有限长轴）.

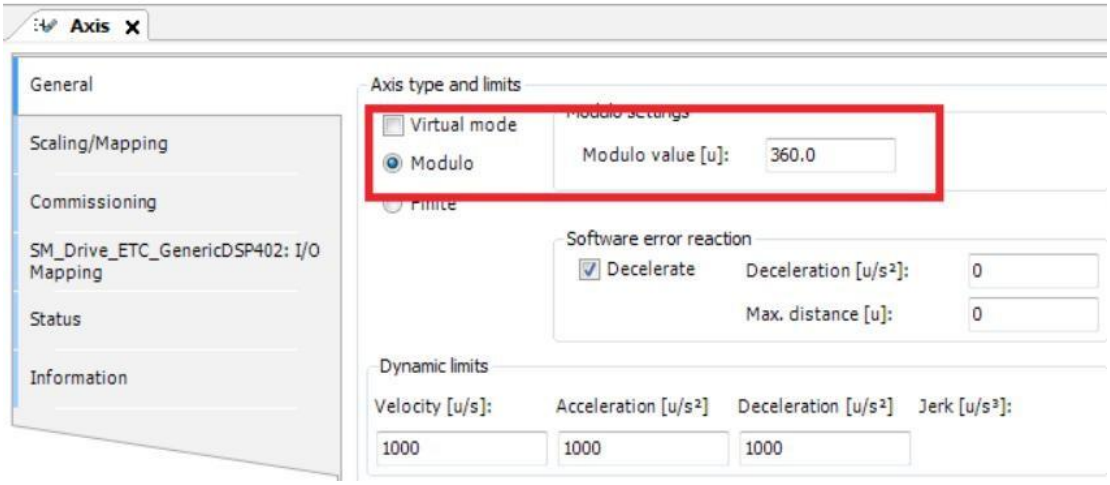
输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bDone	完成	BOOL	TRUE,FALSE	FALSE	True, 执行设定完成
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True, 执行中
bError	错误	BOOL	TRUE,FALSE	FALSE	True, 异常产生
nErrorID	错误代码	SMC_ERROR			参考 SMC_Error

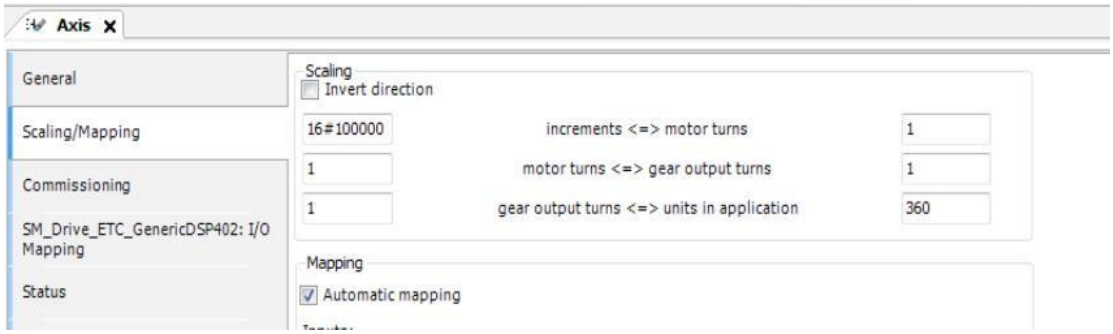
(3)功能说明

bEecute 上升沿,无错误则，bBusy 输出为 TURE,完成 bDone 输出为 true，bBusy 输出为 false。

比如 20 位编码器伺服电机加 10:1 减速比，驱动丝杠（10mm 节距），则电机转 10 圈，丝杆动距离 10mm，设置 dwRatioTechUnitsDenom 1048576*10， iRatioTechUnitsNum 为10。



该功能块的作用为程序动态修改下图所示的部分：



(4)错误说明

bExecute 上升沿时：

轴报错， Error 输出。

输入值无效， Error 输出 ，错误代码 SMC_CGR_ZERO_VALUES

轴在指令控制运行中， Error 输出 ，错误代码 SMC_CGR_DRIVE_POWERED

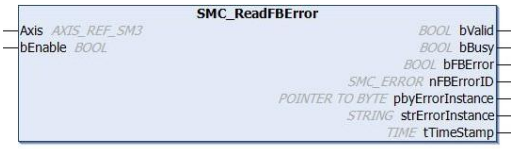
输入的模数值无效（eg： <0），Error 输出 ，错误代码 SMC_CGR_INVALID_POSPERIOD

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.31 SMC_ReadFBError

该指令功能为： MC,SMC 功能块错误 .

(1)指令格式

指令	名称	图形表现	ST 表 现
SMC_ ReadFBErr or	读功能块错误		SMC_ReadFBError(Axis:= , bEnable:= , bValid=> , bBusy=> , bFBError=> , nFBErrorID=> , pbyErrorInstance=> , strErrorInstance=> , tTimeStamp=>);

(2)相关变量

输入输出变量

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	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	执行	BOOL	TRUE,FALSE	FALSE	TRUE: 执行读取

输出变量

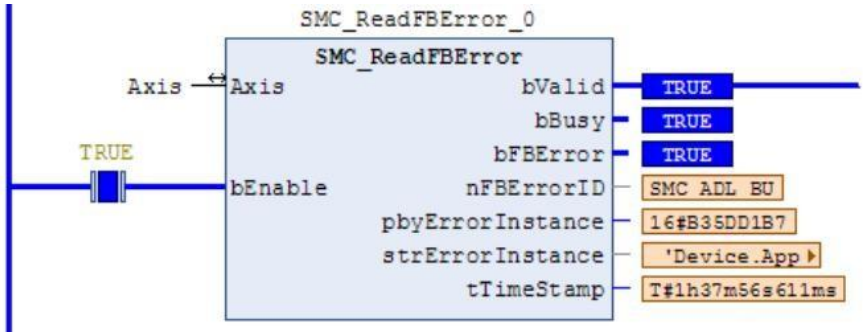
输出变量	名称	数据类型	有效范围	初始值	描述
bValid	有效	BOOL	TRUE,FALSE	FALSE	True, 读取有效
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True, 执行中
bFBError	错误	BOOL	TRUE,FALSE	FALSE	True, 有 FB 错误产生

nFBErrorID	错误代码	SMC_ERROR			参考 SMC_Error
pbyErrorInstance		POINTER TO BYTE			输出点的功能块报错
strErrorInstance		STRING			指向错误功能块（程序，子程序，功能块）
tTimeStamp		TIME			错误发生时的时间戳

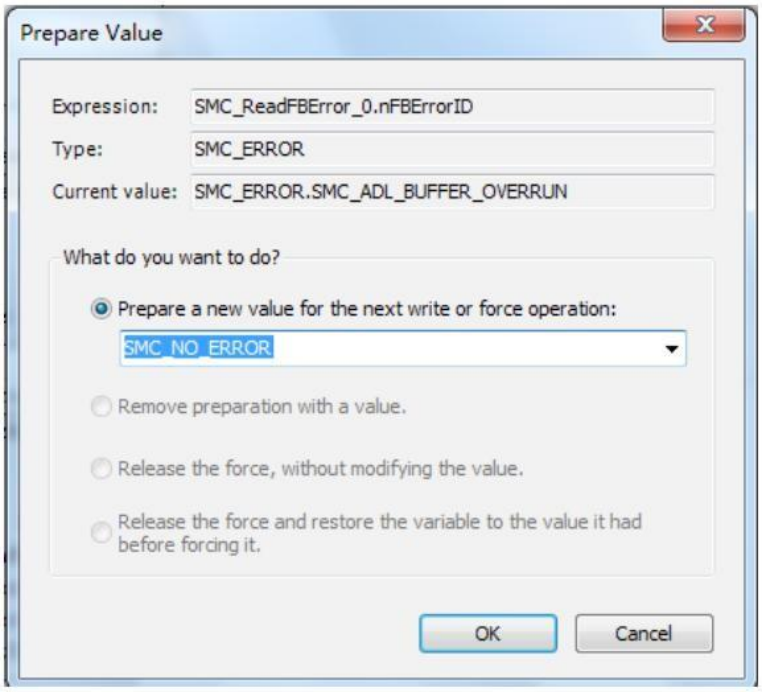
(3)功能说明

Enable 为 TRUE，无错误则 Valid， Busy 输出为 TURE。有功能块报警则，bFBError输出为 true。Enable 变为 FALSE， ，则 Valid， Busy 输出为 FALSE 。

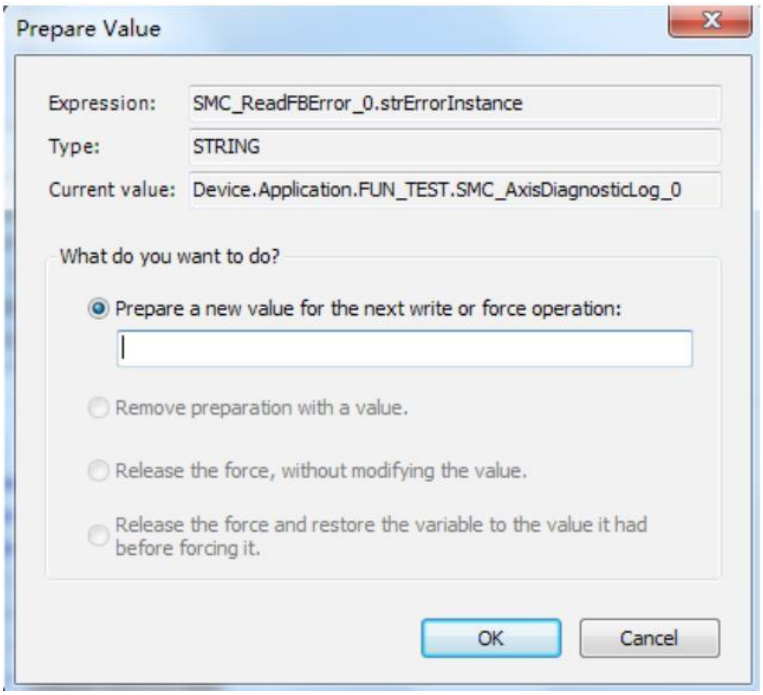
时序图样列程序

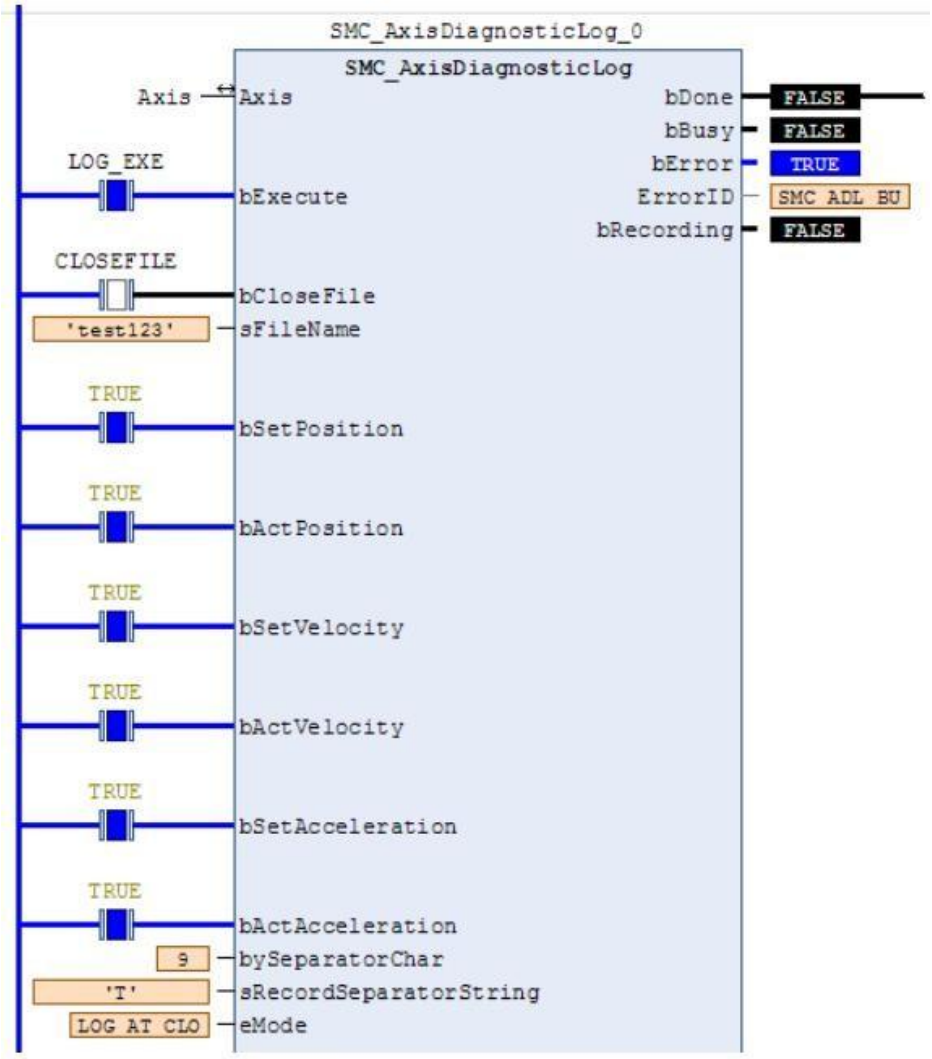


样例程序


Prepare Value
Expression: SMC_ReadFBError_0.nFBErrorID
Type: SMC_ERROR
Current value: SMC_ERROR.SMC_ADL_BUFFER_OVERRUN
What do you want to do?
Prepare a new value for the next write or force operation:
SMC_NO_ERROR
Remove preparation with a value.
Release the force, without modifying the value.
Release the force and restore the variable to the value it had before forcing it.
OK
Cancel

错 误 ID


Prepare Value
Expression: SMC_ReadFBError_0.strErrorInstance
Type: STRING
Current value: Device.Application.FUN_TEST.SMC_AxisDiagnosticLog_0
What do you want to do?
Prepare a new value for the next write or force operation:
Remove preparation with a value.
Release the force, without modifying the value.
Release the force and restore the variable to the value it had before forcing it.
OK
Cancel



发生错误的功能块

错 误 说 明

bExecute 上升沿时：

轴报错， Error 输出；

无效的轴输入， Error 输出。

【注意】：请阅读“附录 C 错误代码说明” 以了解相关错误代码说明。

6.4.32 SMC_ClearFBError

该指令功能为：清除功能块的 FB 错误。

1） 指令格式

指令	名称	图形表现	ST 表 现
SMC_ ClearFBErr or	清 除 功 能 块 错 误		TEST:=SMC_ ClearFBError(pDrive:=A DR(Axis));

2） 相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
pDrive	轴	AXIS_REF	-	-	映射到轴，即 AXIS_REF_SM3 的一个实例

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
SMC_ClearFB Error	清除错误	BOOL	TRUE,FALS E	FALSE	True，清 除

附录A_LC1200支持的原点回归模式

回零模式设置流程

注意：如果是绝对值编码器，且采用 Z 点作为编码器零点，请先预先设置好P03.79-绝对值编码器每周输出多少脉冲。

（1） 先 设 置 6060=6

（2）设置回零偏置 607Ch，其单位为用户位置单位。

（3）设置回零方式 6098h

（4）设置寻找原点开关的速度 6099h_01,其单位是 rpm

（5）设置寻找 Z 点的速度 6099h_02,其单位是 rpm

（6）设置回零加减速度 609Ah，该值单位为用户单位/s/s

（7） 设置控制字 6040h 依次为 6->7->15->31,执行回零

（8） 读取状态字 6041h

回零模式相关对象

回零方式 6098h

索引	6098h
名称	回零方式
对象类型	变量
数据类型	有符号 8 位
PDO 映射	可映射
读写属性	可读可写
默认值	0
设置范围	0-35
详细描述	设置回零方式

回零速度 6099h

索引	6099h
名称	回零速度
对象类型	数组对象

数据类型	无符号 32 位
PDO 映射	可映射
读写属性	可读可写

索引_子索引	6099h_00
名称	6099h 有效子索引个数
数据类型	无符号 32 位
PDO 映射	不可映射
读写属性	只读
默认值	2

索引_子索引	6099h_01
名称	寻找原点开关的速度 rpm

数据类型	无符号 32 位
PDO 映射	可映射
读写属性	可读可写
默认值	P03.53

索引_子索引	6099h_02
名称	寻找 Z 点的速度 rpm
数据类型	无符号 32 位
PDO 映射	可映射
读写属性	可读可写
默认值	P03.54

回零加速度 609Ah

索引	609Ah
名称	回零加速度
对象类型	变量
数据类型	无符号 32 位
PDO 映射	可映射
读写属性	可读可写
默认值	500000
设置范围	0~4294967295
详细描述	回零加速度，单位 用户单位/s/s

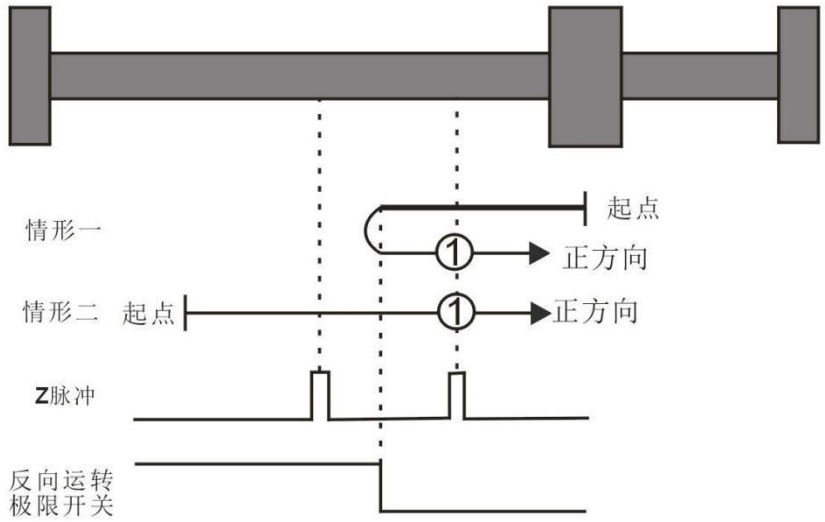
回零模式

回零就是标定一个机械零点，标记后，所有绝对位置都以该零点作为参考点进行运动。 VEC 总线型伺服有多种回零方式，根据回零方式的 6098h 的设置，执行相应的回零动作。用户可根据现场条件及工艺要求选择合适的原点回零模式。

原点回零模式 1：取决于反向运转极限开关和 Z 脉冲的原点回归

情形一：用户触发执行回零时，若反向运转极限开关状态处于低位，那么轴开始以第一段速　　反向运动，当遇到反向运转极限开关处于高位时，运动方向改变且以第二段速开始运动；在反向运转极限开关状态处于低位时遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若反向运转极限开关状态处于高位，那么直接以第二段速开　　始正向运动，在反向运转极限开关状态处于低位时遇到第一个 Z 脉冲的位置就是原点。

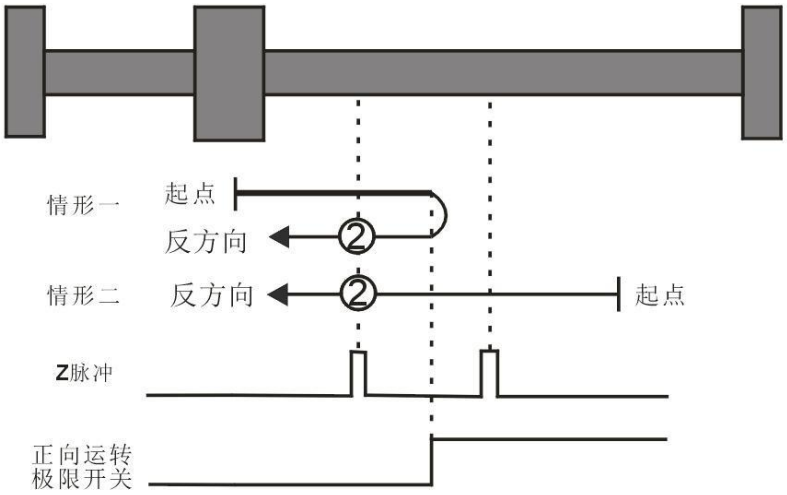


取决于反向运转极限开关和Z脉冲的原点回零，上图中的①表示原点回零模式1

原点回零模式 2：取决于正向运转极限开关和 Z 脉冲的原点回归

情形一：用户触发执行回零时，若正向运转极限开关状态处于低位，那么轴开始以第一段速　　正向运动，当遇到正向运转极限开关处于高位时，运动方向改变且以第二段速开始运动，在正向运转极限开关状态处于低位时遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若正向运转极限开关状态处于高位，那么轴直接以第二段速　　开始反向运动，在正向运转极限开关状态处于低位时遇到第一个



取决于正向运转极限开关和Z脉冲的原点回零，上图中的②表示原点回零模式2

Z 脉冲的位置就是原点位置。

模式 3~模式 6 取决于原点开关和 Z 脉冲的原点回零

原点回零模式 3

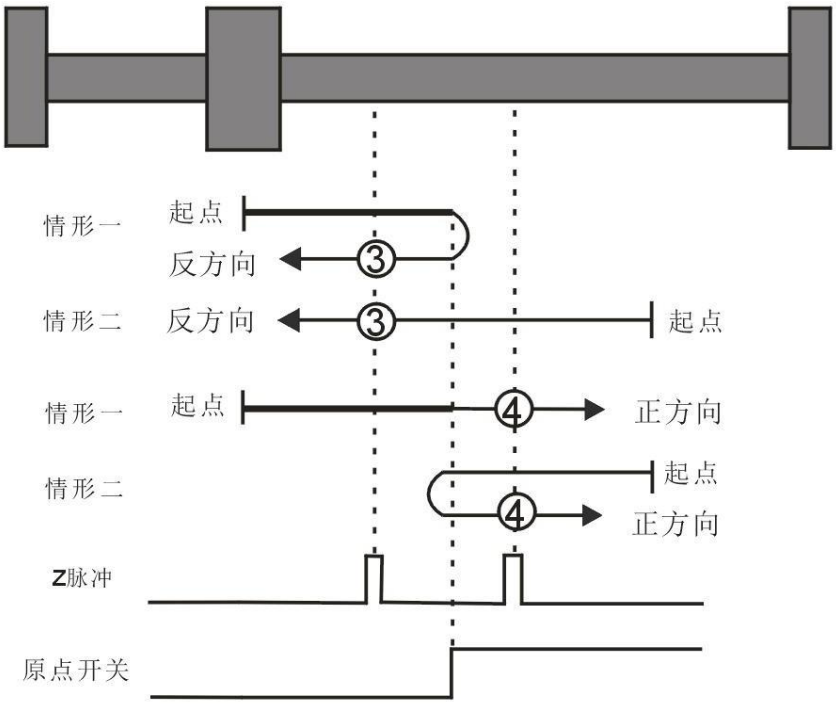
情形一：用户触发执行回零时，若原点开关状态处于低位，轴开始以第一段速正向运动，当 遇到原点开关处于高位时，运动方向改变且以第二段速开始运动，在原点开关状态处于低位 时遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始反向运动，在原点开关状态处于低位时遇到第一个 Z 脉冲的位置就是原点位置。

原点回零模式 4

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时，以第二段速正向运动，遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始反向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。



取决于原点开关和Z脉冲的原点回零，上图中的③、④表示原点回零模式3、4

原点回零模式 5

情形一：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始正向运动，在原点开关状态处于低位时遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时，运动方向改变且以第二段速开始运动，在原点开关状态处于低位时遇到第一个 Z 脉冲的位置就是原点位置。

原点回零模式 6

情形一：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始正向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时，以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

取决于原点开关和Z脉冲的原点回零，上图中的⑤、⑥表示原点回零模式5、6

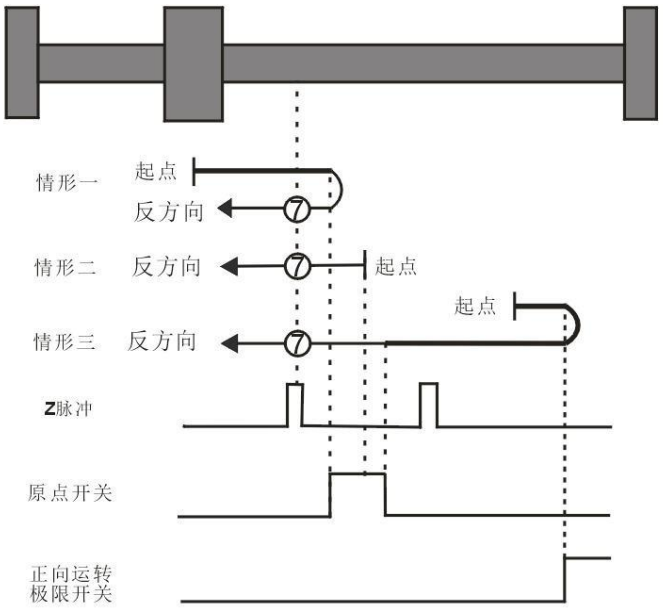
模式 7～模式 10 取决于原点开关、正向运转极限和 Z 脉冲的原点回零

原点回零模式 7

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时，运动方向改变且以第二段速开始运动，在原点开关状态处于低位时遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始反向运动，在原点开关状态处于低位时遇到第一个 Z 脉冲的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方



取决于原点开关、正向运转极限开关和Z脉冲的原点回零，上图中的⑦表示原点回零模式7

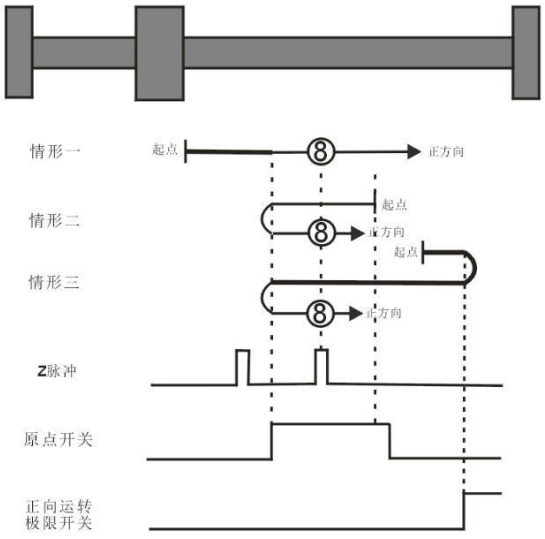
向改变且以第一段速开始运动，当遇到原点开关处于高位时，以第二段速开始运动，在原点开关状态处于低位时遇到第一个 Z 脉冲的位置就是原点位置。

原点回零模式 8

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时，以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始反向 运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时，仍以第一段速运动，



取决于原点开关、正向运转极限开关和Z脉冲的原点回零，上图中的⑧表示原点回零模式8

在原点开关状态处于低位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于高位时，以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

原点回零模式 9

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时，以第二段速开始运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴开始以第二段速正向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时，以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

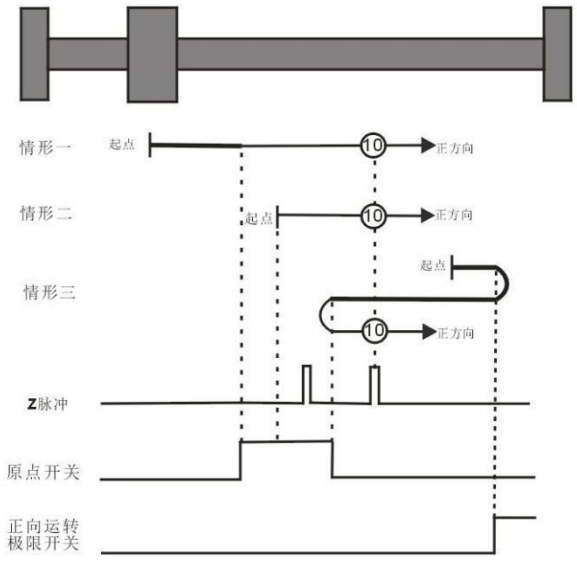
取决于原点开关、正向运转极限开关和Z脉冲的原点回零，上图中的⑨表示原点回零模式9

原点回零模式 10

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时，以第二段速开始运动，当遇到原点开关处于低位时，遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴开始以第二段速正向运动，当遇到原点开关处于低位时，遇到第一个 Z 脉冲的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方



取决于原点开关、正向运转极限开关和Z脉冲的原点回零，上图中的⑩表示原点回零模式10

向改变且以第一段速开始运动，当遇到原点开关处于高位时，运动方向再次改变且以第二段速开始运动，当原点开关处于低位时，遇到第一个 Z 脉冲的位置就是原点位置。

模式 11～模式 14 取决于原点开关、反向运转极限和 Z 脉冲的原点回零

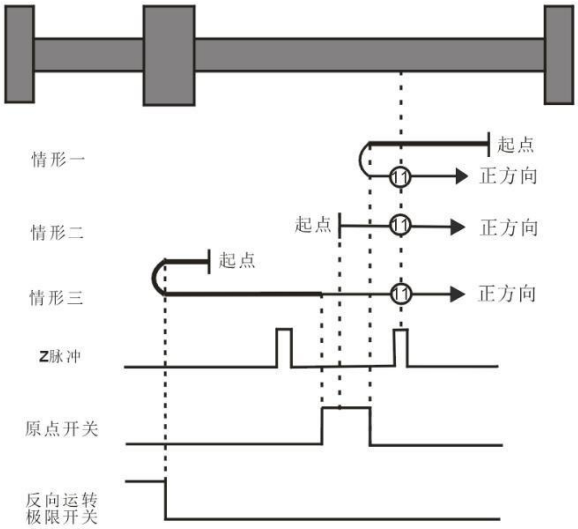
原点回零模式 11

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时，运动方向改变且以第二段速开始运动，在原点开关状态处于低

位时遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始正向 运动,在原点开关状态处于低位时遇到第一个 Z 脉冲的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方



取决于原点开关、反向运转极限开关和Z脉冲的原点回零，上图中的表示⓫原点回零模式11

向改变且以第一段速开始运动，当遇到原点开关处于高位时，以第二段速开始运动，在原点开关状态处于低位时遇到第一个 Z 脉冲的位置就是原点位置。

原点回零模式 12

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时，以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始正向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方向改变且以第一段速开始运动,当遇到原点开关处于高位时,仍以第一段速运动，在原点开关状态处于低位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时，以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位

置。

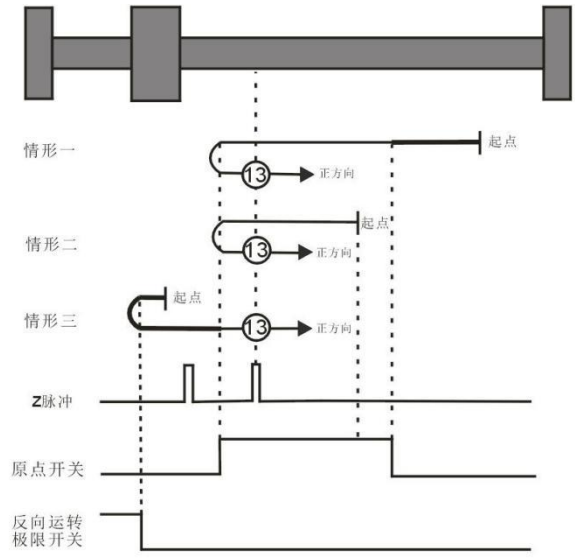
取决于原点开关、反向运转极限开关和Z脉冲的原点回零，上图中的表示⑫原点回零模式12

原点回零模式 13

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时，以第二段速开始运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速反向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时，以第二段速开始运动，遇到第一个 Z 脉冲的位置就是原点位置。



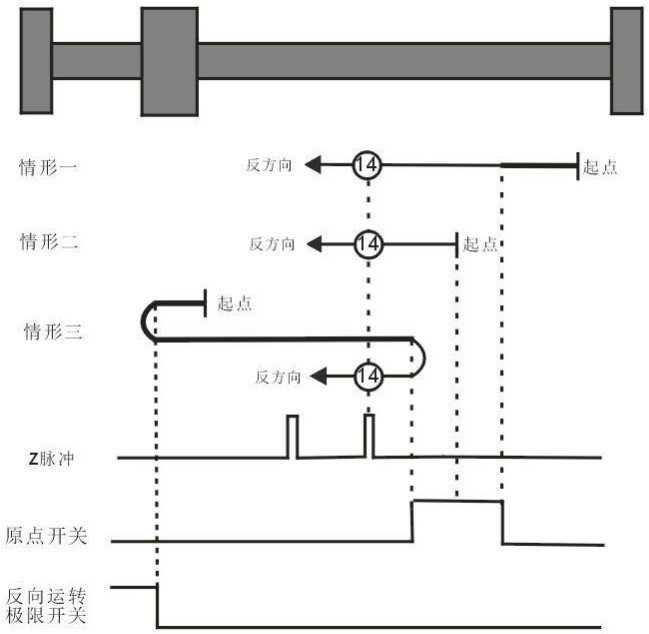
取决于原点开关、反向运转极限开关和Z脉冲的原点回零，上图中的表示⑬原点回零模式13

原点回零模式 14

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时，以第二段速开始运动，当遇到原点开关处于低位时，遇到第一个 Z 脉冲的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴开始以第二段速反向运动，当遇到原点开关处于低位时，遇到第一个 Z 脉冲的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方



取决于原点开关、反向运转极限开关和Z脉冲的原点回零，上图中的表示⑭原点回零模式14

向改变且以第一段速开始运动，当遇到原点开关处于高位时，运动方向再次改变且以第二段速开始运动，当遇到原点开关处于低位时，遇到第一个 Z 脉冲的位置就是原点位置。

模式 15～模式 16 保留

模式 15 和模式 16 被保留，作为以后发展的原点回归模式。模式 17～模式 30 需要 Z 脉冲的原点回归

模式 17~模式 30 分别和前面所讲的模式 1~模 14 相似，只是它们的原点回归位置的定位不再需要 Z 脉冲，而是仅仅根据相关原点开关和极限开关的状态改变来实现。模式 17 与模式1 相似，模式 18 与模式 2 相似，模式 19 和模式 20

同前面的模式 3 相似，模式 21 和模式22 同前面的模式 5 相似，模式 23 和模式 24 同前面的模式 7 相似，模式 25 和模式 26 同前面的模式 9 相似。模式 27 和模式 28 同前面的模式 11 相似，模式 29 和模式 30 同前面的模式 13 相似。

原点回零模式 17：取决于反向运转极限开关的原点回零

情形一：用户触发执行回零时，若反向运转极限开关状态处于低位，那么轴开始以第一段速反向运动，当遇到反向运转极限开关处于高位时，运动方向改变且以第二段速开始运动；在反向运转极限开关状态处于低位时的位置就是原点位置。

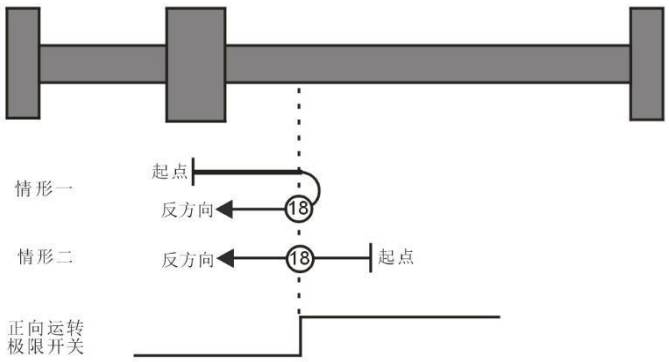
情形二：用户触发执行回零时，若反向运转极限开关状态处于高位，那么轴直接以第二段速开始正向运动，在反向运转极限开关状态处于低位时的位置就是原点位置。

取决于反向运转极限开关的原点回零，上图中的表示⑰原点回零模式17

原点回零模式 18：取决于正向运转极限开关的原点回归

情形一：用户触发执行回零时，若正向运转极限开关状态处于低位，那么轴开始以第一段速　　正向运动，当遇到正向运转极限开关处于高位时，运动方向改变且以第二段速开始运动，在正向运转极限开关状态处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若正向运转极限开关状态处于高位，那么轴直接



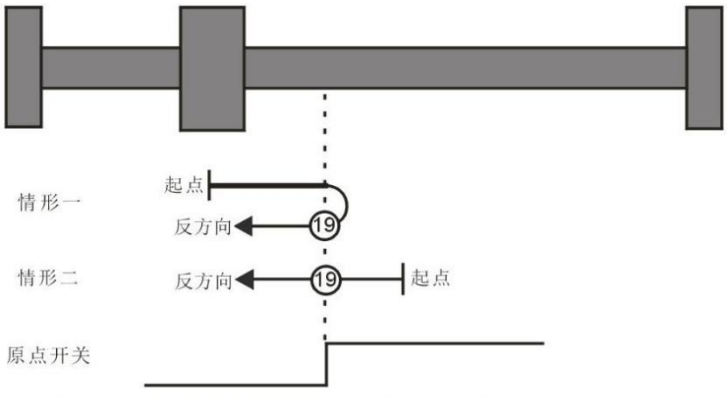
取决于正向运转极限开关的原点回零，上图中的表示⑱原点回零模式18

以第二段速 开始反向运动，在正向运转极限开关状态处于低位时的位置就是原点位置。

原点回零模式 19

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段



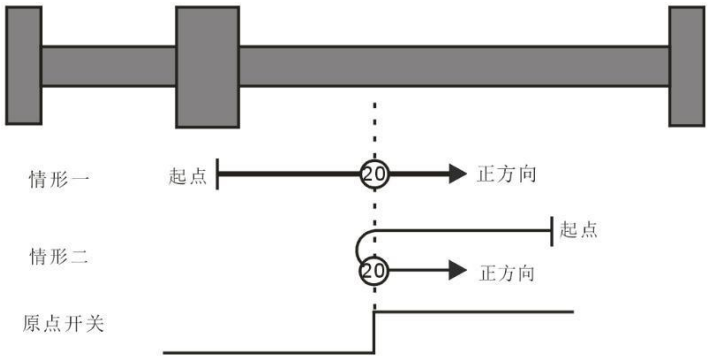
取决于原点开关的原点回零，上图中的表示⑲原点回零模式19

速开始反向运动，当遇到原点开关处于低位时的位置就是原点位置。

原点回零模式 20

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始反向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于高位时的位置就是原点位置。

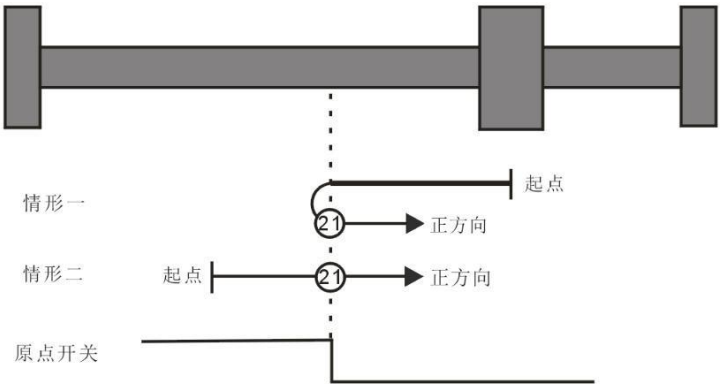


取决于原点开关的原点回零，上图中的表示20原点回零模式20

原点回零模式 21

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始正向 运动，当遇到原点开关处于低位时的位置就是原点位置。



取决于原点开关的原点回零，上图中的表示21原点回零模式21

原点回零模式 22

情形一：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始正向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时的位置就是原点位置。

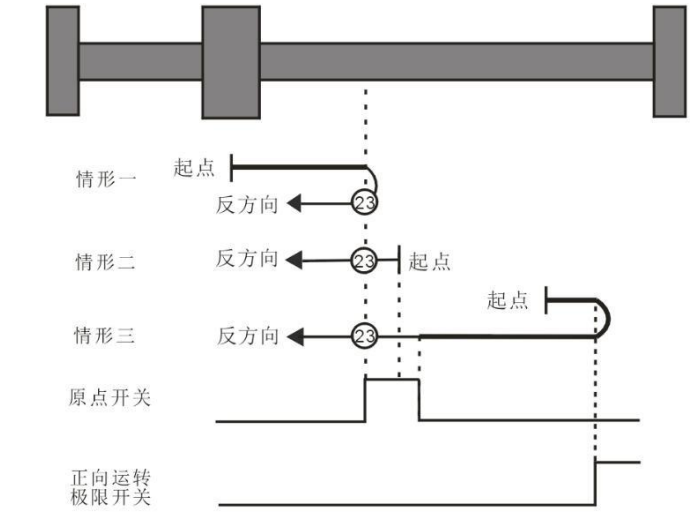
取决于原点开关的原点回零，上图中的表示㉒原点回零模式22

原点回零模式 23

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时，运动方向改变且以第二段速开始运动，在原点开关状态处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始反向运动，在原点开关状态处于低位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时，以第二段速开始运动，在原点开关状态处于低位时的位置就是原点位置。



取决于原点开关、正向运转极限开关的原点回零，上图中的表示㉓原点回零模式23

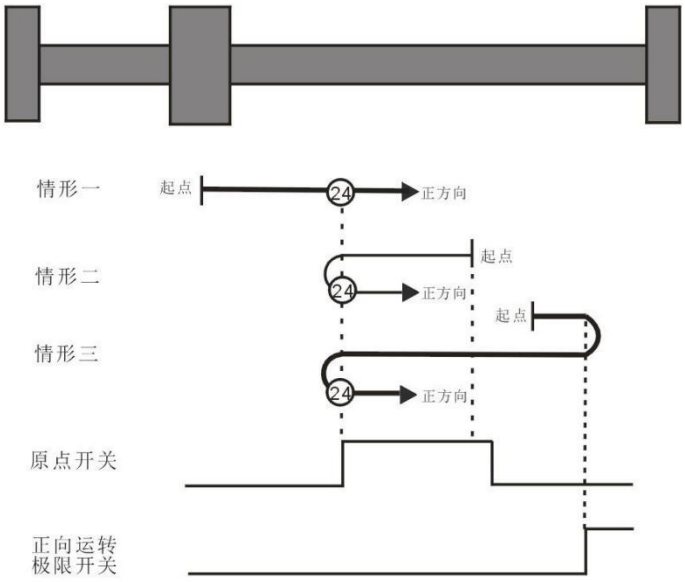
原点回零模式 24

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始反向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，遇到原点开关处于高位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段

速正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时，仍以第一段速运动，在原点开关状态处于低位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时的位置就是原点位置。



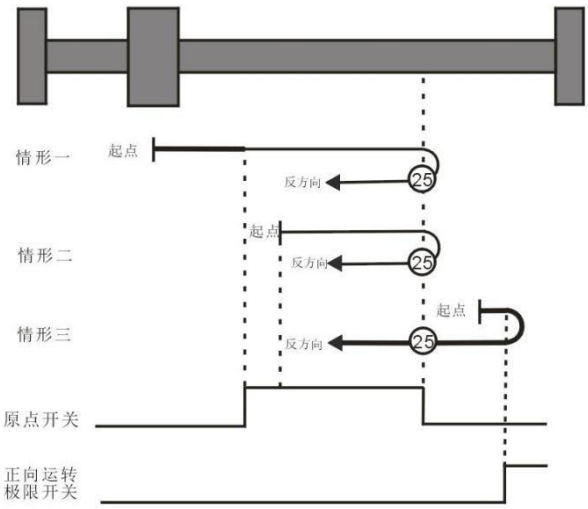
取决于原点开关、正向运转极限开关的原点回零，上图中的表示24原点回零模式24

原点回零模式 25

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时，以第二段速开始运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴开始以第二段速正向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于高位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时的位置就是原点位置。



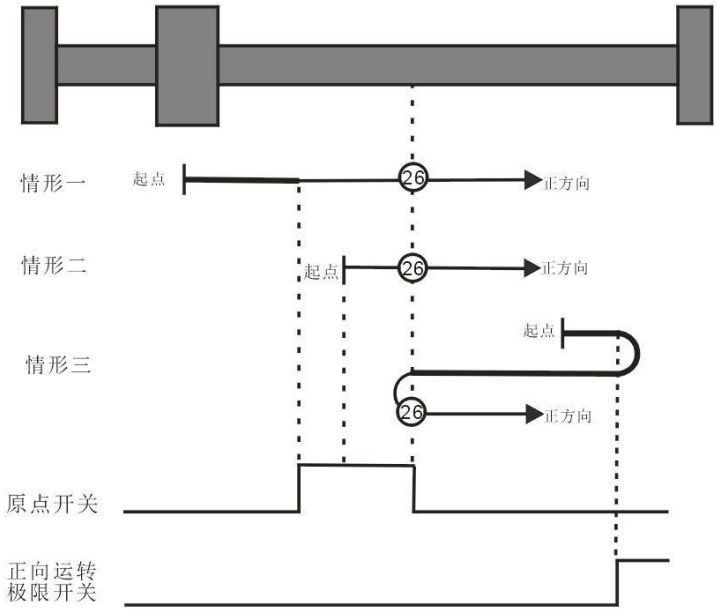
取决于原点开关、正向运转极限开关的原点回零，上图中的表示25原点回零模式25

原点回零模式 26

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时，以第二段速开始运动，当遇到原点开关处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴开始以第二段速正向运动，当遇到原点开关处于低位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时，运动方向再次改变且以第二段速开始运动，当原点开关处于低位时的位置就是原点位置。



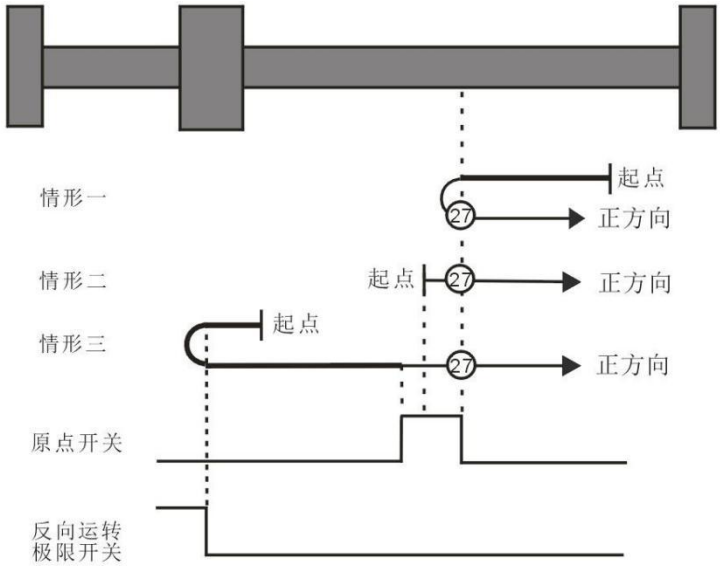
取决于原点开关、正向运转极限开关的原点回零，上图中的表示26原点回零模式26

原点回零模式 27

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时，运动方向改变且以第二段速开始运动，在原点开关状态处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始正向运动，在原点开关状态处于低位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时，以第二段速开始运动，在原点开关状态处于低位时的位置就是原点位置。



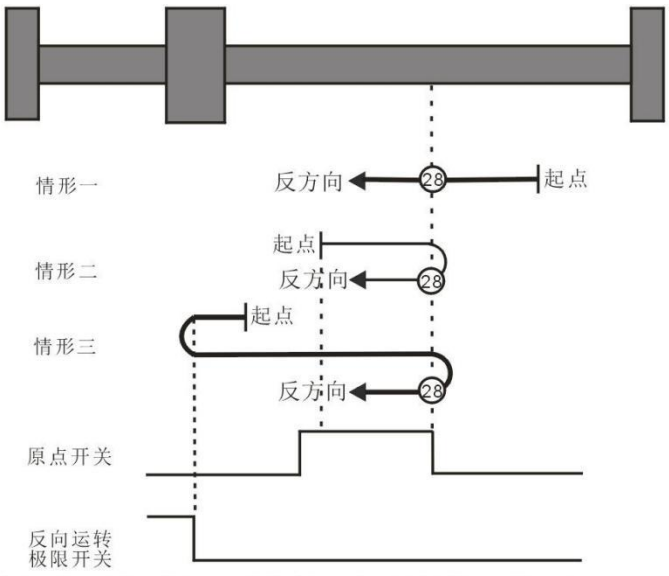
取决于原点开关、反向运转极限开关的原点回零，上图中的表示27原点回零模式27

原点回零模式 28

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始正向　运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于高位时的的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方



取决于原点开关、反向运转极限开关的原点回零，上图中的表示28原点回零模式28

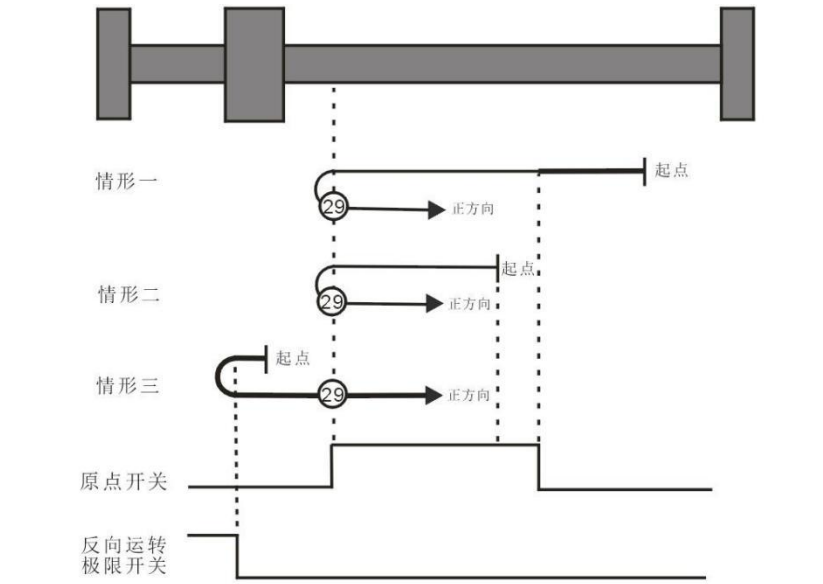
向改变且以第一段速开始运动，当遇到原点开关处于高位时，仍以第一段速运动，在原点开关状态处于低位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时的位置就是原点位置。

原点回零模式 29

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时，以第二段速开始运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速反向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于高位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时的位置就是原点位置。



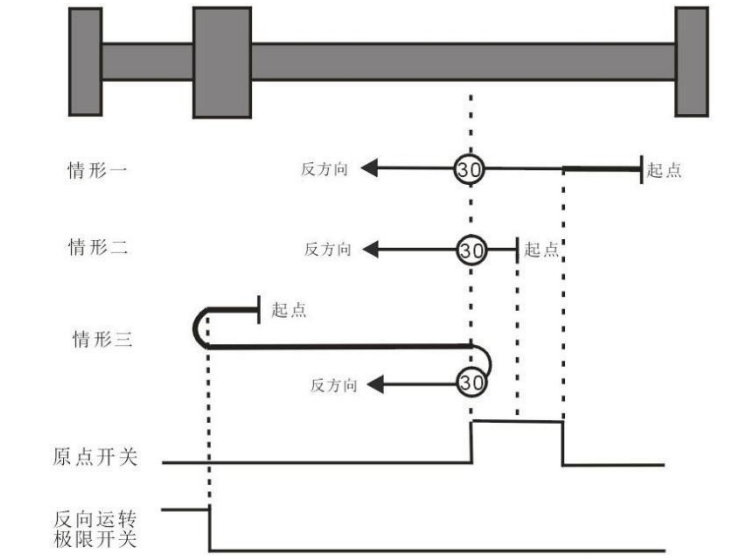
取决于原点开关、反向运转极限开关的原点回零，上图中的表示29原点回零模式29

原点回零模式 30

情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当遇到原点开关处于高位时，以第二段速开始运动，当遇到原点开关处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴开始以第二段速反向运动，当遇到原点开关处于低位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方向改变且以第一段速开始运动，当遇到原点开关处于高位时，运动方向再次改变且以第二段速开始运动，当遇到原点开关处于低位时的位置就是原点位置。



取决于原点开关、反向运转极限开关的原点回零，上图中的表示30原点回零模式30

模式 31 和模式 32 保留

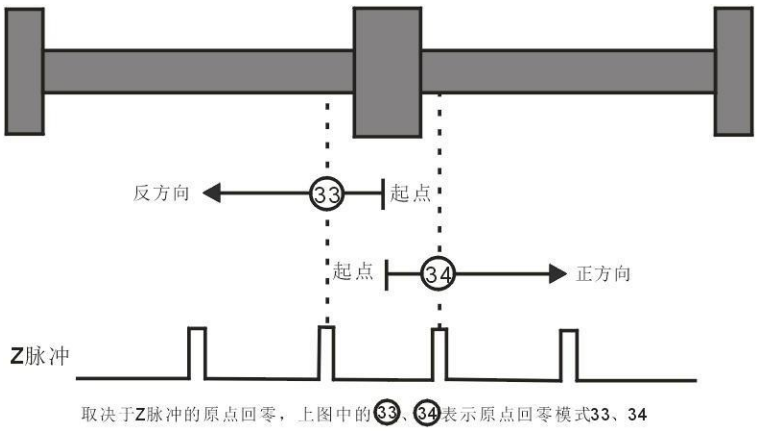
模式 31 和模式 32 被保留，作为以后发展的原点回归模式。模式 33～ 模式 34 取决于 Z 脉冲的原点回归

原点回零模式 33

在模式 33 下，用户触发执行回零时，轴开始以第二段速反向运动，当遇到第一个 Z 脉冲的位置就是原点位置。

原点回零模式 34

在模式 34 下，用户触发执行回零时，轴开始以第二段速正向运动，当遇到第一个 Z 脉冲的位置就是原点位置。



取决于Z脉冲的原点回零，上图中的33、34表示原点回零模式33、34

原点回零模式 35:取决于当前位置的原点回归

在模式 35 下，用户触发执行回零时，轴不运动，轴的当前位置被认为是原点回归的位置。

附录B_LC1200支持的CiA402常用数据对象速查表

索引(hex)	子索引(hex)	名称	访问	大小	单位	设置范围	默认值	PDO映射
603F	00	错误码	RO	UINT16	-	TPDO		
此对象给出驱动器最近发生的故障码或警告码，对应低 12 位给出故障码其定义可查看总线伺服说明书。查看故障记录可以通过 200B:22 与 23 来查看最多 10 条最新的故障记录码。								
6040	00	控制字	RW	UINT16	-	0~65535	0	RPDO
伺服上电后状态引导，各伺服模式下指令控制								
6041	00	状态字	RO	UINT16	-	TPDO		
反应伺服驱动器运行状态。								
6050	00	慢速停机时间	RW	UDINT32	ms	0-U32MAX	5000	
伺服慢速停机时间设置								
6051	00	快速停机时间	RW	UDINT32	ms	0-U32MAX	50	
伺服快速停机时间设置								
605A	00	快速停机方式选择	RW	INT8	0~7	2	-	
0~7：选择驱动器快速停机方式								
605D	00	暂停停机方式选择	RW	INT8	1~3	1	-	
选择驱动器暂停方式								
605E	00	故障响应选项	RW	INT16	-	0-4	0	
0-4 直接断使能 快速停机断使能 慢速停机断使能 快速停机保持使能 慢速停机保持使能								
6060	00	伺服模式选择	RW	INT8	-	0~10	0	RPDO
1- 轮廓位置模式 (pp) 轮廓速度模式 (pv) 轮廓转矩模式 (pt) 6- 原点回零模式 (hm) 周期同步位置模式 (csp) 周期同步速度模式 (csv) 周期同步转矩模式 (cst)								
6061	00	伺服运行模式显示	RO	INT8	-	TPDO		
实际运行模式								
6062	00	位置指令	RO	INT32	指令单位	TPDO		
每个位置环周期时间内的位置指令值，指令单位								
6063	00	位置反馈	RO	INT32	编码器单位	TPDO		

电机编码器反馈的电机当前位置。								
6064	00	位置反馈	RO	INT32	指令单位	TPDO		
经齿轮比逆运算后的位置反馈值。 6063=6064× 齿轮比								

索引(hex)	子索引(hex)	名称	访问	大小	单位	设置范围	默认值	PDO映射
6065	00	位置偏差过大阀值	RW	UINT32	指令单位	0~232^{-1}	3145728	RPDO
当位置偏差 60F4 大于 ±6065 时驱动器报位置偏差过大 (Er.B00) 故障。同时轮廓位置模式下， 6041 的 bit13=1，此故障可复位。								
6067	00	位置到达阈值	RW	UINT32	指令单位	0~65535	7	RPDO
当位置偏差 60F4 小于此值，且时间达到 6068 时，定位完成的 DO 信号有效，同时 6041 的 bit10=1。不满足两者之中任一条件，位置到达无效。								
6068	00	位置到达窗口时间	RW	UINT16	ms	0-65535	0	RPDO
当位置偏差 60F4 小于此值，且时间达到 6068 时，定位完成的 DO 信号有效，同时 6041 的 bit10=1。不满足两者之中任一条件，位置到达无效。								
606B	00	速度指令值	RO	DINT32	0.1RPM	TPDO	0	
伺服速度指令值								
606C	00	实际速度	RO	INT32	s	TPDO		
此对象显示每秒位置反馈 (指令单位)								
606D	00	速度到达阈值	RW	UINT32	rpm	0~65535	10	RPDO
当电机速度反馈与速度指令的差值在 ±606D 以内，且时间达到 606E 时，速度到达的 DO 信号有效，同时 6041 的 bit10=1。不满足两者之中任一条件，速度到达无效。								
606E	00	速度到达窗口时间	RW	UINT16	ms	0-65535	0	RPDO
当速度反馈与速度指令的差值在 ±606D 以内，且时间达到 606E 时，速度到达的 DO 信号有效，同时 6041 的 bit10=1。不满足两者之中任一条件，速度到达无效。								
606F	00	零速阈值	RW	UINT16	0.1rpm	0-65535	50	RPDO
伺服的零速阈值								
6071	00	目标转矩	RW	INT16	0.1	-5000~5000	0	RPDO
转矩模式下，目标转矩设定								
6072	00	最大转矩指令	RW	UINT16	0.1	0~5000	0	RPDO
最大转矩限制值。								
6074	00	转矩指令	RO	INT16	0.1%	-5000~5000	0	TPDO

驱动器内部计算后的转矩输出指令								
6075	00	电机额定电流	RO	UDINT32	mA	-	-	TPDO
电机额定电流								
6077	00	实际转矩	RO	INT16	0.1%	-5000~5000	0	TPDO
驱动器获取的反馈转矩值								
6078	00	驱动器实际转矩百分比	RO	INT16	0.1%	-	-	TPDO
驱动器实际转矩百分比								
607A	00	目标位置	RW	INT32	指令单位	$-231^{-(231-1)}$	0	RPDO
上位机给定的目标位置，根据位置因子即控制字，伺服电机行驶响应的位移增量。								
607C	00	原点偏移量	RW	INT32	指令单位	$-231^{-(231-1)}$	0	RPDO
机械原点偏移机械零点的位置								
607D	软件绝对位置限制							
	子索引	子索引个数	RO	UINT8	-	2	2	-
	01	最小位置限制	RW	INT32	用户位置单位	$-231^{-(231-1)}$	-231	RPDO
	02	最大位置限制	RW	INT32	用户位置单位	$-231^{-(231-1)}$	231^{-1}	RPDO
原点回零完成后，通过与 607C 结合，设置允许运行的最小与最大位置限制值。超过该值的位置指令将在到达限位后止。								
607E	00	指令极性	RW	UINT8	-	0-255	0	RPDO
BIT7- 位置指令极性 :0- 保持原极性， 1- 极性反转 BIT6- 速度指令极性 :0- 保持原极性， 1- 极性反转 BIT5- 转矩指令极性 :0- 保持原极性， 1- 极性反转								
607F	00	最大速度	RW	UDINT32	指令单位 /s	$0\sim232^{-1}$	104857600	RPDO
允许的最大速度限定值。设定方法：								

607F = 允许电机最大转速 (rpm)* 编码器分辨率 /60								
6080	00	电机最高转速	RW	UDINT32	0.1RPM	-	-	RPDO
电机最高转速								
6081	00	轮廓运行速度	RW	UINT32	用户单位	0~232-1	0	RPDO
轮廓位置模式下，电机该段位移内的匀速运行速度设定								
6083	00	轮廓加速度	RW	UINT32	指令单位 /s2	1~232-1	1747626667	RPDO
pp，csv，pv 模式下加速度。默认值 1747626667 指令单位 /s2 表示：从 0rpm 加速到 1000rpm 用时 10ms。								
6084	00	轮廓减速度	RW	UINT32	指令单位 /s2	1~232-1	1747626667	RPDO
pp，csv，pv 模式下减速度。默认值 1747626667 指令单位 /s2 表示：从 0rpm 加速到 1000rpm 用时 10ms。								
6085	00	快速停机减速度	RW	UINT32	用户加速度单位	1~232-1	1747626667	RPDO
上位机发出快速停机指令 (6040 的 bit2=0) 时， 605A=2 时的减速段加速度。 默认值 1747626667 指令单位 /s2 表示：从 0rpm 加速到 1000rpm 用时 10ms。								
6086	00	运行曲线选择	RW	INT16	-	0	0	RPDO
设置轮廓位置模式下的电机运行曲线。								

目前仅支持直线运动。								
6087	00	转矩斜坡	RW	UINT32	0.1%/s	0	0xFFFFFFFF	RPDO
设置轮廓转矩模式下，每秒转矩指令增量								
6091	齿轮比							
	00	子索引个数	RO	UINT8	2	2		
	01	电机分辨率	RW	UINT32	-	0~232-1	1	RPDO
	02	负载轴分辨率	RW	UINT32	-	1-232-1	1	RPDO
建立编码器单位与指令单位间的比例关系。								
6098	00	原点复归方法	RW	INT8	-	0-35	0	RPDO
支持 DS402 协议规定的 35 种回零方式								
6099	01	高速搜索减速点	RW	UINT32	指令单位/s	0~232-1	1747626	RPDO
	02	搜索原点低速	RW	UINT32	指令单位/s	0~232-1	174762	RPDO
609A	00	回零加速度	RW	UINT32	指令单位/s2	1~232-1	1747	RPDO
原点回零模式下，变速段的加速度。默认值 1747 指令单位 /s2 表示：从 0rpm 加速到 1000rpm 用时 10ms。								
60B0h	00	位置偏置	RW	INT32	指令单位	-231-(231-1)	0	RPDO
60B1h	00	速度偏置	RW	INT32	指令单位/s	-231-(231-1)	0	RPDO
60B2h	00	转矩偏置	RW	INT32	0.1%	-5000-5000	0	RPDO
60B8h	00	探针模式	RW	UINT16	-	0-65535	0	RPDO
60B9h	00	探针状态	RW	UINT16	-	0-65535	0	RPDO
60BAh	00	探针 1 上升沿位置值	RW	INT32	指令单位	-231-(231-1)	0	RPDO
60BBh	00	探针 1 下降沿位置值	RW	INT32	指令单位	-231-(231-1)	0	RPDO

60BCh	00	探针 2 上升沿位置值	RW	INT32	指令单位	-231^(-(231-1))	0	RPDO
60BDh	00	探针 2 下降沿位置值	RW	INT32	指令单位	-231^(-(231-1))	0	RPDO
60E0h	00	正向转矩限制	RW	UINT16	0.1%	0-5000	2000	RPDO
60E1h	00	反向转矩限制	RW	UINT16	0.1%	0-5000	2000	RPDO
60E3h	00	支持的回零方式	RW	UINT16	-	-	-	-
60E6h	00	位置计算方式	RW	UINT16	-	0-1	0	-
60F4h	00	位置偏差	RO	INT32	指令单位	-231^(-(231-1))	0	TPDO
位置偏差，指令单位								
60FDh	00	DI 状态	RO	UINT32	-	0~232^-1	0	RPDO
60FEh	00	DO 状态	RO	UINT32	-	0~232^-1	0	RPDO
60FFh	00	目标速度	RW	INT32	指令单位 /s	-231^(-(231-1))	0	RPDO
同步周期速度模式下，设定的速度指令								
6502	00	支持驱动模式	RO	UINT32	0000 03ADhex	TPDO		
显示驱动器支持的相关模式。								

附录C_错误代码说明

SMC_ERROR：记录了SoftMotion功能块可能返回的错误代码。

错误代码	产生源	变量名称	错误原因描述
0	所有	SMC_NO_ERROR	没有错误
1	驱动器接口	SMC_DI_GENERAL_COMMUNICATION_ERROR	通讯错误（例如 Sercos 环断裂）
2	驱动器接口	SMC_DI_AXIS_ERROR	轴错误
10	驱动器接口	SMC_DI_SWLIMITS_EXCEEDED	软限位被激活（bSWLimitEnable 使能后，轴的当前位置不在 fSWLimitPositiLC1200 和 fSWLimitNegatiLC1200 范围）
11	驱动器接口	SMC_DI_HWLIMITS_EXCEEDED	硬件限位开关被激活
13	驱动器接口	SMC_DI_HALT_OR_QUICKSTOP_NOT_SUPPORTED	驱动器状态停止或者不支持快速停止
14	驱动器接口	SMC_DI_VOLTAGE_DISABLED	驱动器没有使能
15	驱动器接口	SMC_DI_IRREGULAR_ACTPOSITION	驱动器当前给予的位置格式不正确。检查通讯。
16	驱动器接口	SMC_DI_POSITIONLAGERROR	位置滞后错误。在设置和当前位置超过限制值
20	所有运动控制创建的模块	SMC_REGULATOR_OR_START_NOT_SET	控制器没有使能或者抱闸没有打开
21	轴在错误的控制模式下	SMC_WRONG_CONTROLLER_MODE	轴不是一个正确的控制方式
30	驱动器接口	SMC_FB_WASNT_CALLED_DURING_MOTION	运动控制创建的模块在运动结束之前没有被调用
31	所有模块	SMC_AXIS_IS_NO_AXIS_REF	给出的 AXIS_REF 变量不是 AXIS_REF 类型
32	轴位于错误控制模式下	SMC_AXIS_REF_CHANGED_DURING_OPERATION	AXIS_REF- 变量的返回值在模块激活前被处理
33	驱动器接口	SMC_FB_ACTILC1200_AXIS_DIABLED	轴在移动时没有被激活(MC_Power.bRegulatorOn)
34	所有运动控制创建的模块	SMC_AXIS_NOT_READY_FOR_MOTION	在当前状态下轴不能处理当前命令
40	虚拟驱动器	SMC_VD_MAX_LC1200LOCITY_EXCEEDED	达到最大速度(fMaxLC1200locity)
41	虚拟驱动器	SMC_VD_MAX_ACCELERATION_EXCEEDED	达到最大加速度(fMaxAcceleration)
42	虚拟驱动器	SMC_VD_MAX_DECELERATION_EXC	达到最大减速度

错误代码	产生源	变量名称	错误原因描述
		EEDED	(fMaxDeceleration)
50	SMC_Homing	SMC_3SH_INVALID_LC1200LACC_VALUES	无效的速度或者加速度值
51	SMC_Homing	SMC_3SH_MODE_NEEDS_HWLIMIT	模块需要使用结束限位开关（安全用途）
70	SMC_SetControllerMode	SMC_SCM_NOT_SUPPORTED	模式不支持
71	SMC_SetControllerMode	SMC_SCM_AXIS_IN_WRONG_STATE	在当前模式下使用的控制模式不支持
75	SMC_SetTorque	SMC_ST_WRONG_CONTROLLER_MODE	轴不是一个正确的控制模式，需要在转矩模式下使能此功能块
80	SMC_ResetAxisGroup	SMC_RAG_ERROR_DURING_STARTUP	在轴组启动时发生错误
90	SMC_ChangeGearingRatio	SMC_CGR_ZERO_VALUES	不正确的变量
91	SMC_ChangeGearingRatio	SMC_CGR_DRILC1200_POWERED	驱动器控制模式下不能更改传动比
92	SMC_ChangeGearingRatio	SMC_CGR_INVALID_POSPERIOD	不合适的位置周期 (<=0)
110	MC_Power	SMC_P_FTASKCYCLE_EMPTY	轴在扫描周期内不包含任何信息 (fTaskCycle = 0)
120	MC_Reset	SMC_R_NO_ERROR_TO_RESET	轴没有错误复位
121	MC_Reset	SMC_R_DRILC1200_DOESNT_ANSWER	轴没有执行错误复位
122	MC_Reset	SMC_R_ERROR_NOT_RESETTABLE	错误不能被复位
123	MC_Reset	SMC_R_DRILC1200_DOESNT_ANSWER_IN_TIME	与轴之间的通讯没有回应
130	MC_ReadParameter, MC_ReadBoolParameter	SMC_RP_PARAM_UNKNOWN	参数序号位置
131	MC_ReadParameter, MC_ReadBoolParameter	SMC_RP_REQUESTING_ERROR	在将参数传送到驱动器过程中发生错误。参阅 功能块实例 ReadDriLC1200Parameter 的错误 (SM_DriLC1200Basic.lib)
140	MC_WriteParameter, MC_WriteBoolParameter	SMC_WP_PARAM_INVALID	参数序号位置或者不允许进行写操作
141	MC_WriteParameter, MC_WriteBoolParameter	SMC_WP_SENDING_ERROR	参阅模块实例 WriteDriLC1200Parameter 的错误 (DriLC1200_Basic.lib)

错误代码	产生源	变量名称	错误原因描述
170	MC_Home	SMC_H_AXIS_WASNT_STANDSTILL	轴不是标准状态
171	MC_Home	SMC_H_AXIS_DIDNT_START_HOMING	在执行回零时发生错误
172	MC_Home	SMC_H_AXIS_DIDNT_ANSWER	通讯错误
173	MC_Home	SMC_H_ERROR_WHEN_STOPPING	执行回零错误停止。查阅是否设置减速度。
180	MC_Stop	SMC_MS_UNKNOWN_STOPPING_ERROR	停止时发生完未知错误
181	MC_Stop	SMC_MS_INVALID_ACCDEC_VALUES	不合适的速度或者加速度值
182	MC_Stop	SMC_MS_DIRECTION_NOT_APPLICABLE	Direction=shortest 不可用
183	MC_Stop	SMC_MS_AXIS_IN_ERRORSTOP	轴位于错误停止状态。停止不能被处理。
184	MC_Stop	SMC_BLOCKING_MC_STOP_WASNT_CALLED	一个 MC_Stop 的实例，锁定轴(Execute=TRUE)，不能进行调用。请调用 MC_Stop(Execute=FALSE).
201	MC_MoLC1200Absolute	SMC_MA_INVALID_LC1200LACC_VALUES	不合适的速度或者加速度值
202	MC_MoLC1200Absolute	SMC_MA_INVALID_DIRECTION	方向错误
226	MC_MoLC1200RelatiLC1200	SMC_MR_INVALID_LC1200LACC_VALUES	不合适的速度或者加速度值
227	MC_MoLC1200RelatiLC1200	SMC_MR_INVALID_DIRECTION	方向错误
251	MC_MoLC1200AdditiLC1200	SMC_MAD_INVALID_LC1200LACC_VALUES	不合适的速度或者加速度值
252	MC_MoLC1200AdditiLC1200	SMC_MAD_INVALID_DIRECTION	方向错误
276	MC_MoLC1200SuperImposed	SMC_MSI_INVALID_LC1200LACC_VALUES	不合适的速度或者加速度值
277	MC_MoLC1200SuperImposed	SMC_MSI_INVALID_DIRECTION	方向错误
301	MC_MoLC1200LC1200locity	SMC_MV_INVALID_ACCDEC_VALUES	不合适的速度或者加速度值
302	MC_MoLC1200LC1200locity	SMC_MV_DIRECTION_NOT_APPLICABLE	Direction=shortest/fastest 不支持

错误代码	产生源	变量名称	错误原因描述
325	MC_PositionProfile	SMC_PP_ARRAYSIZE	错误排列尺寸
326	MC_PositionProfile	SMC_PP_STEP0MS	步时间 ＝t#0s
350	MC_LC1200locityProfile	SMC_VP_ARRAYSIZE	错误排列尺寸
351	MC_LC1200locityProfile	SMC_VP_STEP0MS	步时间 ＝t#0s
375	MC_AccelerationProfile	SMC_AP_ARRAYSIZE	错误排列尺寸
376	MC_AccelerationProfile	SMC_AP_STEP0MS	步时间 ＝t#0s
400	MC_TouchProbe	SMC_TP_TRIGGEROCCUPIED	触发条件已经被激活
401	MC_TouchProbe	SMC_TP_COULDNT_SET_WINDOW	驱动器接口不支持窗口功能
402	MC_TouchProbe	SMC_TP_COMM_ERROR	通讯错误
410	MC_AbortTrigger	SMC_AT_TRIGGERNOTOCCUPIED	触发条件已经被终止
426	SMC_MoLC1200ContinuousRelatiLC1200	SMC_MCR_INVALID_LC1200LACC_VALUES	不合适的速度或者加速度值
427	SMC_MoLC1200ContinuousRelatiLC1200	SMC_MCR_INVALID_DIRECTION	方向错误
451	SMC_MoLC1200ContinuousAbsolute	SMC_MCA_INVALID_LC1200LACC_VALUES	不合适的速度或者加速度值
452	SMC_MoLC1200ContinuousAbsolute	SMC_MCA_INVALID_DIRECTION	方向错误
453	SMC_MoLC1200ContinuousAbsolute	SMC_MCA_DIRECTION_NOT_APPLICABLE	Direction= fastest 不可用
600	SMC_CamRegister	SMC_CR_NO_TAPPETS_IN_CAM	CAM 中不包含任何挺杆
601	SMC_CamRegister	SMC_CR_TOO_MANY_TAPPETS	挺杆组 ID 达到 MAX_NUM_TAPPETS
602	SMC_CamRegister	SMC_CR_MORE_THAN_32_ACCESSES	在一个 CAM_REF 中多于 32 个接口
625	MC_CamIN	SMC_CI_NO_CAM_SELECTED	没有 CAM 被选中
62	MC_CamIN	SMC_CI_MASTER_OUT_OF_SCALE	主轴超出范围

错误代码	产生源	变量名称	错误原因描述
6			
627	MC_CamIN	SMC_CI_RAMPIN_NEEDS_LC1200LACC_VALUES	针对 ramp_in 功能块速度和加速度必须被精确指定
628	MC_CamIN	SMC_CI_SCALING_INCORRECT	比例变量 fEditor/TableMasterMin/Max 不正确
640	SMC_CAMBounds, SMC_CamBounds_Pos	SMC_CB_NOT_IMPLEMENTED	给予的 CAM 格式的功能块不支持
675	MC_GearIn	SMC_GI_RATIO_DENOM	RatioDenominator = 0
676	MC_GearIn	SMC_GI_INVALID_ACC	加速度不合适
677	MC_GearIn	SMC_GI_INVALID_DEC	加速度不合适
725	MC_Phase	SMC_PH_INVALID_LC1200LACCDEC	速度，加速度，减速度不合适
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	给予的 CAM 不是类型 MC_CAM_REF
751	MC_CamTableSelect	SMC_CAM_TABLE_DOES_NOT_COLC1200R_MASTER_SCALE	如果从 CamTable 中获取的数据不是通过数据转化得到的主轴区域 (xStart and xEnd)
775	MC_GearInPos	SMC_GIP_MASTER_DIRECTION_CHANGE	在从轴耦合过程中主轴改变旋转方向
800	SMC_BacklashCompensation	SMC_BC_BL_TOO_BIG	齿轮返回比 (fBacklash) 太大 (>position periode/2)
1000	CNC 需要授权的功能块	SMC_NO_LICENSE	目标没有进行 CNC 的授权。
1001	SMC_Interpolator	SMC_INT_LC1200L_ZERO	路径不能被处理因为速度 = 0.
1002	SMC_Interpolator	SMC_INT_NO_STOP_AT_END	上一个路径对象 LC1200l_End > 0.
1003	SMC_Interpolator	SMC_INT_DATA_UNDERRUN	警告： GEOINFO- 列表在 DataIn 进行处理，但是列表最后没有被设置。理由：忘记在 DataIn 中设置 EndOfList 或者 SMC_Interpolator 比路径编

错误代码	产生源	变量名称	错误原因描述
			译模块处理速度快。
1004	SMC_Interpolator	SMC_INT_LC1200L_NONZERO_AT_STOP	停止速度 ＞0.
1005	SMC_Interpolator	SMC_INT_TOO_MANY_RECURSIONS	使用太多 SMC_Interpolator 调用 SoftMotion- 错误。
1006	SMC_Interpolator	SMC_INT_NO_CHECKLC1200LOCITIES	Input-OutQueue DataIn 没有作为 SMC_CHeckLC1200locities 的最后处理模块
1007	SMC_Interpolator	SMC_INT_PATH_EXCEEDED	内部 / 数值错误错误
1008	SMC_Interpolator	SMC_INT_LC1200L_ACC_DEC_ZERO	速度，加速度或者减速度为空或者太低。
1009	SMC_Interpolator	SMC_INT_DWIPOTIME_ZERO	FB 调用 dwIpoTime = 0
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	路径没有完全包含在队列中
1100	SMC_CheckLC1200locities	SMC_CV_ACC_DEC_LC1200L_NONPOSITILC1200	速度，减速度或者加速度值不为正向
1120	SMC_Controlaxisbypos	SMC_CA_INVALID_ACCDEC_VALUES	变量 of fGapLC1200locity / fGapAcceleration /fGapDeceleration 不是正值
1200	SMC_NCDecoder	SMC_DEC_ACC_TOO_LITTLE	加速度值不允许
1201	SMC_NCDecoder	SMC_DEC_RET_TOO_LITTLE	减速度值不允许
1202	SMC_NCDecoder	SMC_DEC_OUTQUEUE_RAN_EMPTY	低于 Queue 的数据被读取并且为空。
1203	SMC_NCDecoder	SMC_DEC_JUMP_TO_UNKNOWN_LINE	因为行号未知所以跳转的行号不能执行
1204	SMC_NCDecoder	SMC_DEC_INVALID_SYNTAX	语法错误
1205	SMC_NCDecoder	SMC_DEC_3DMODE_OBJECT_NOT_SUPPORTED	这些对象不支持 3D 模式
1300	SMC_GCodeViewer	SMC_GCV_BUFFER_TOO_SMALL	缓冲区太小
13	SMC_GCodeViewer	SMC_GCV_BUFFER_WRONG_TYPE	缓冲区元素类型错误

错误代码	产生源	变量名称	错误原因描述
01			
1302	SMC_GCodeViewer	SMC_GCV_UNKNOWN_IPO_LINE	当前插补行不能被找到
1500	使用SMC_CNC_REF的所有功能块	SMC_NO_CNC_REF_TYPE	给定的 CNC 程序不是类型 SMC_CNC_REF
1501	所有使用 SMC_OUTQUEUE的功能块	SMC_NO_OUTQUEUE_TYPE	给定的 OutQueue 不是类型SMC_OUTQUEUE
1600	CNC 功能块	SMC_3D_MODE_NOT_SUPPORTED	这个功能块只在2D路径中可用
2000	SMC_ReadNCFile	SMC_RNCF_FILE_DOESNT_EXIST	文件不存在
2001	SMC_ReadNCFile	SMC_RNCF_NO_BUFFER	没有缓冲分配
2002	SMC_ReadNCFile	SMC_RNCF_BUFFER_TOO_SMALL	缓冲区太小
2003	SMC_ReadNCFile	SMC_RNCF_DATA_UNDERRUN	换种区中低缓冲数据被读取，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	输入的 pvl 不能指向 SMC_VARLIST 对象
2050	SMC_ReadNCQueue	SMC_RNCQ_FILE_DOESNT_EXIST	文件不能打开
2051	SMC_ReadNCQueue	SMC_RNCQ_NO_BUFFER	没有缓冲区定义
2052	SMC_ReadNCQueue	SMC_RNCQ_BUFFER_TOO_SMALL	缓冲区太小
2053	SMC_ReadNCQueue	SMC_RNCQ_UNEXPECTED_EOF	未知文件结尾
2100	SMC_AxisDiagnosticLog	SMC_ADL_FILE_CANNOT_BE_OPENED	文件不能被打开
2101	SMC_AxisDiagnosticLog	SMC_ADL_BUFFER_OLC1200RRUN	超过范围的缓冲；WriteToFile 必须更经常的调用
2200	SMC_ReadCAM	SMC_RCAM_FILE_DOESNT_EXIST	文件不能打开
2201	SMC_ReadCAM	SMC_RCAM_TOO_MUCH_DATA	保存到 CAM 数据太多

错误代码	产生源	变量名称	错误原因描述
2202	SMC_ReadCAM	SMC_RCAM_WRONG_COMPILE_TYPE	错误编译模式
2203	SMC_ReadCAM	SMC_RCAM_WRONG_LC1200RSION	文件版本错误
2204	SMC_ReadCAM	SMC_RCAM_UNEXPECTED_EOF	未知的文件结尾
3001	SMC_WriteDriLC1200ParamsToFile	SMC_WDPF_CHANNEL_OCCUPIED	SMC_WDPF_TIMEOUT_PREPARING_LIST
3002	SMC_WriteDriLC1200ParamsToFile	SMC_WDPF_CANNOT_CREATE_FILE	文件不能被创建
3003	SMC_WriteDriLC1200ParamsToFile	SMC_WDPF_ERROR_WHEN_READING_PARAMS	读取文件参数的时候错误
3004	SMC_WriteDriLC1200ParamsToFile	SMC_WDPF_TIMEOUT_PREPARING_LIST	准备参数列表时时间错误
5000	SMC_Encoder	SMC_ENC_DENOM_ZERO	译码器参考的转换因子(dwRatioTechUnitsDenom)为0。
5001	SMC_Encoder	SMC_ENC_AXISUSEDBYOTHERFB	其他模块正在处理译码轴。
5002	驱动器接口	SMC_ENC_FILTER_DEPTH_INVALID	过滤器选择不合适