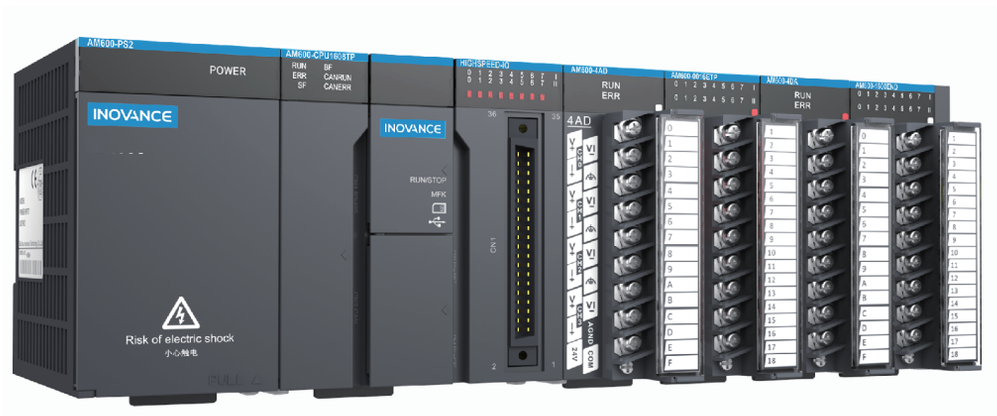
INOVANCE



AM600系列可编程逻辑控制器

编程手册（运动控制篇）

综合手册

A02

资料编码 19010539

前 言

首先感谢您购买使用汇川技术开发生产的 AM600 系列可编程逻辑控制器及扩展模块，并使用 InoProShop 编程软件！

● 面向的用户

本手册面向对 AM600 系列 PLC 进行运动控制相关功能配置、编程和调试的技术人员。读者需要具备一定的自动化和 PLC 知识。

● 主要内容

第 1 章 简介 PLCopen 规范

第 2 章 简单介绍 AM600 运动控制应用系统组成

第 3 章 介绍 AM600 运动控制程序的组成

第 4 章 介绍 AM600 运动控制程序执行机制

第 5 章 介绍 AM600 用户程序的应用编程

第 6 章 详细解析 AM600 常用 MC 指令详解

第 7 章 介绍仿真与调试相关操作

第 8 章 附录内容，含 IS620N 支持的原点回归模式、IS620N 支持的 CiA402 常用数据对象速查表、和 AM600 错误代码说明

在使用本软件之前，应仔细阅读本手册以及本手册，同时在充分注意安全的前提下正确操作。

● 术语和缩略语

术语 / 缩略语	说明
InoProShop	AM600 系列 PLC 的编程软件
Gateway	本手册中代表 AM600 系列 PLC 的专用通讯服务
PLC	Programmable Logic Controller（可编程控制器的简称）

● 版本更新记录

变更时间	版本号	变更说明
2017 年 5 月	A00	第一版发行
2019 年 3 月	A01	切换 Logo
2020 年 11 月	A02	细小勘误

目 录

第1章 PLCopen规范简介

一、PLCopen 规范简介

IEC61131 是一部针对通用可编程控制器 PLC 的国际标准，当时是由欧洲几家 PLC 技术领先的公司首先发起拟定的一个行业标准，其中的 IEC 61131-3 是针对 PLC 编程的国际标准规格，先后定义了 6 种编程语言标准。

PLCopen 是总部在欧洲的 IEC 61131-3 推广团体，是一个全球性的会员组织，该组织中的若干知名 PLC 产品厂家对其中的一些技术细节作了完善，目的是为了使得不同 PLC 厂家的产品应用编程通用化，消除产品技术差异和壁垒，在用户选用用不同品牌 PLC 时，不必另外学习相应的编程方法。

在中国，与其对应的国家推荐标准 GBT15969.3 先后于 1995、2005 年发布和升级，作为 PLC 设备厂家的设计推荐标准。在中国也有相应的 PLCopen 推广组织。

PLCopen 规范除了对通用逻辑控制指令、程序结构、各种语言所含的关键字进行了规范建议外，还对运动控制功能块 MC 的技术规格进行了约定，包括 MC 功能模块的命名、具体功能、输入输出变量定义、相关时序逻辑等，最大限度地保证用户编程技术的互通。

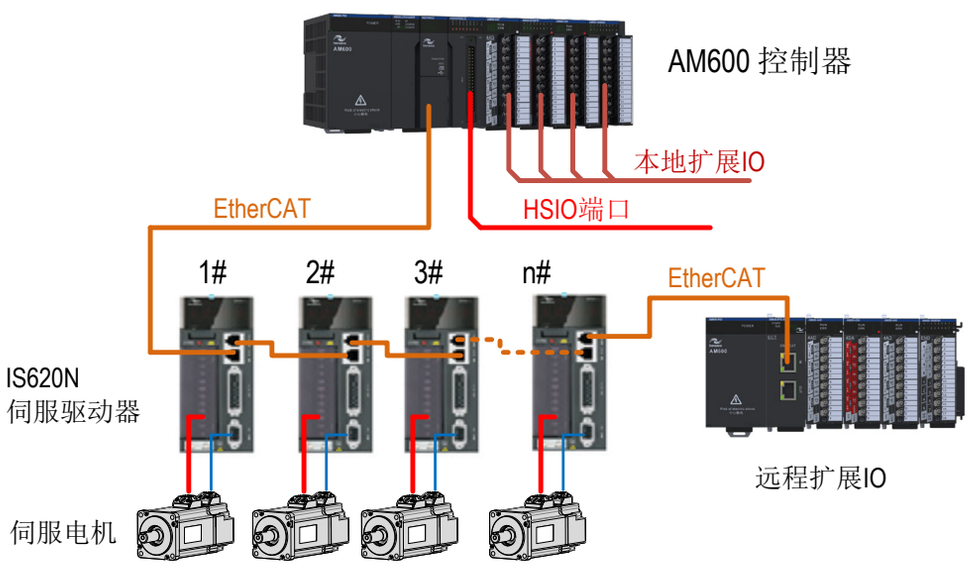
AM600 控制器选用的是德国 3S 公司的 CoDeSys 编程平台，该平台完整支持 PLCopen 规范，用户可以引用许多标准的功能函数库；高级语言的编程方式，易于 PLC 厂家和用户开发自己专有的功能块和指令库，借用已有的类似控制程序，形成行业特点的“工艺包”，可显著提高用户编程效率。

第2章 AM600运动控制应用系统组成

二、AM600 运动控制应用系统组成

AM600 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 IS620N 总线控制型伺服，IO 扩展机架也是通过 EtherCAT 总线与 AM600 控制器的 CPU 模块连接。

如下图为典型的运动控制网络，其中 AM600 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。



MC 是运动控制（Motion Control）的特点是控制器通过软件计算、以数字命令通过 EtherCAT 实时总线命令伺服运行，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。

第3章 AM600运动控制程序的组成

三、AM600 运动控制程序的组成

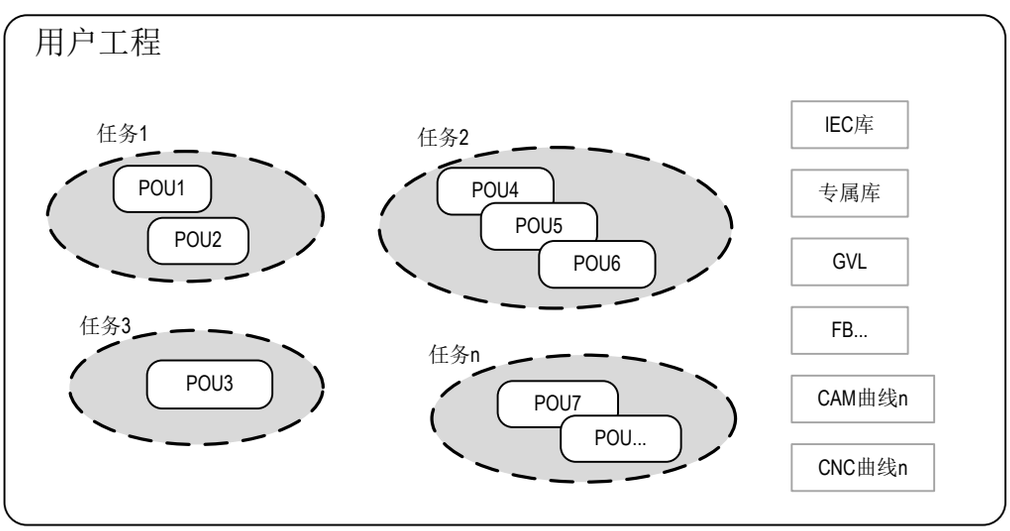
3.1 AM600 的用户程序结构

AM600 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 AM600 的用户程序时，用户可根据应用系统中不同处理的业务类型和紧急程度，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。

3.1.1 AM600 的用户程序组成

正如前面介绍，AM600 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：



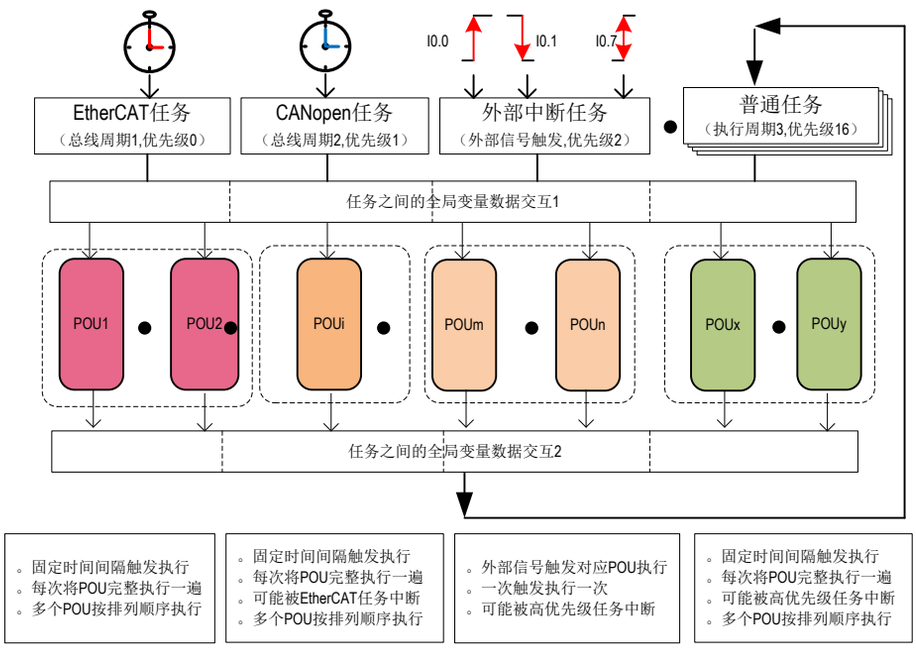
用户工程是由若干个 POU 组成，按照 POU 执行特性要求，分为若干个任务组，配置其执行特性，没有列入任务配置的 POU 将不会被执行的。

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。

3.1.2 AM600 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。

AM600 常见的任务有：EtherCAT 任务、CANopen 任务、HSIO 高速中断任务、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。



其中的 EtherCAT 任务，在 AM600 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：

上图中，三个 POU 的执行顺序依次是 PLC_PRG、POU_ipo、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.EtherCAT_Task 的“POU”项目，默认就是首先被执行，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。

3.1.3 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；

InoProShop 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；

3.1.4 用户程序如何同时做到逻辑控制与运动控制

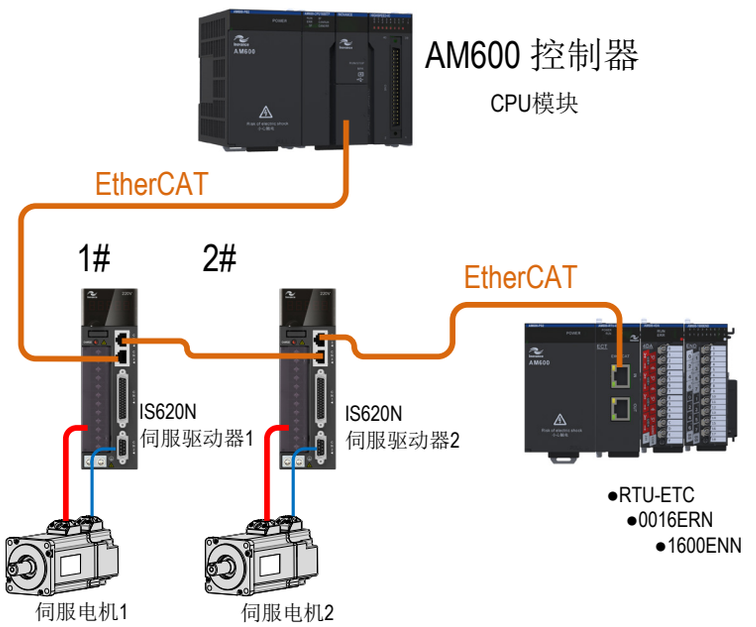
应用系统的同步控制、轨迹控制，往往都有高实时性要求，而逻辑控制的及时性要求则相对较低，在 AM600 用户程序中，可以将运动控制（MC）的 POU 放在 EtherCAT 任务周期中执行，而逻辑控制 POU 就放在普通任务周期中执行即可。若将特定的程序变量声明为全局变量，就可以在运动控制中，实现与逻辑控制的协调动作。

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。

3.2 一个简单的用户程序的编写调试过程

在说明编程系统原理与运控程序编写方法之前，先举例介绍一个基本的伺服控制程序，以便对编程过程有一个初步的了解。例如下面的应用系统中，由 CPU 模块、IS620N 伺服系统、RTU-ETC 及 0016ERN 扩展模块组成:



要求编写一个简单的程序，让 AM600CPU 控制器实现如下功能：

让伺服马达 1 能够点动运行；

每触发一次命令标志，伺服马达 2 运转 2 圈后停止，用于测试系统是否运行正常；

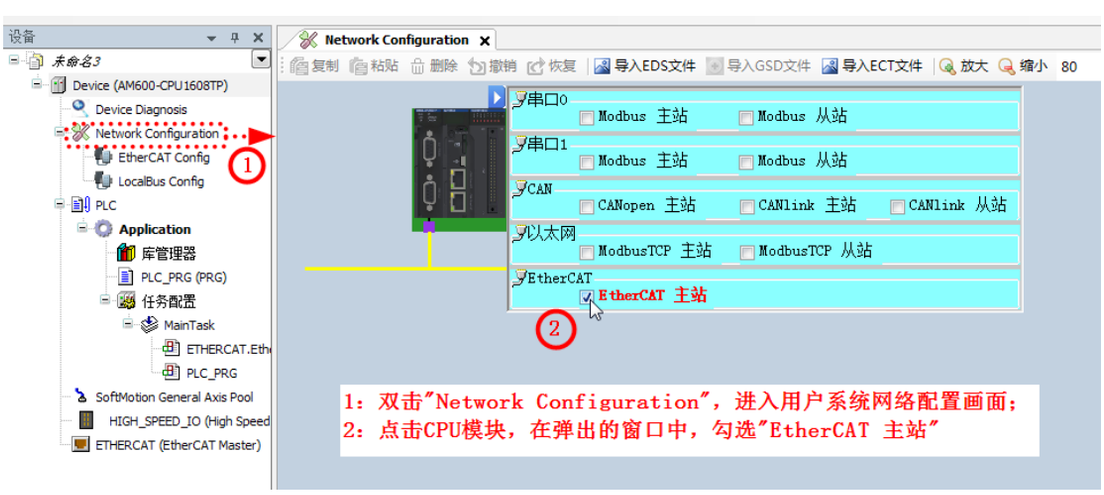
让扩展机架上的 IO 输出端口，进行跑马灯输出，每 0.5 秒由低向高移动 1 位，在 16bit 范围内循环移动。

例程的编程方法与步骤如下：

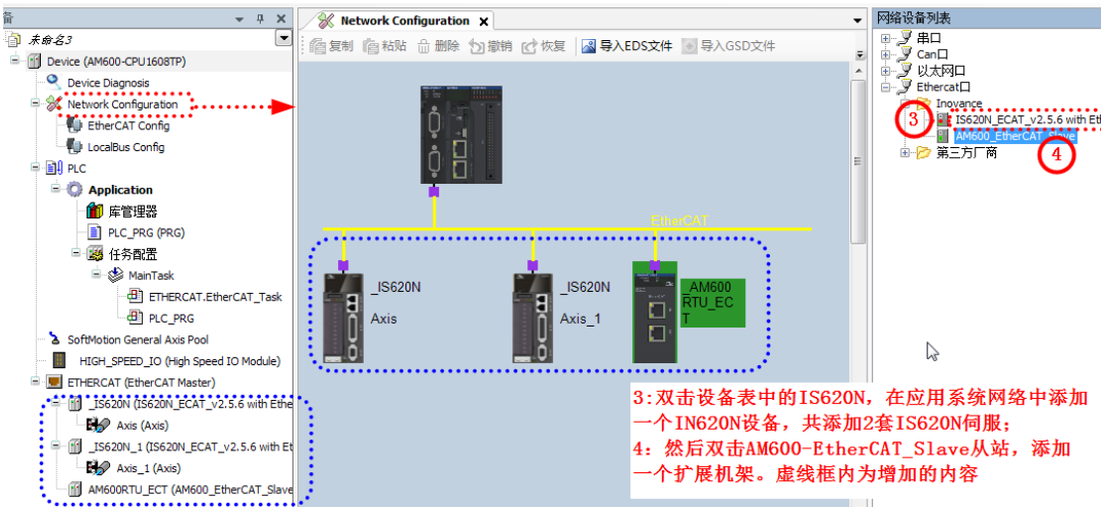
伺服的运动控制，需要放在高实时 EtherCAT 任务周期中处理；跑马灯的控制实时性要求不高，放在 20ms 的任务循环中处理；

3.2.1 新建工程

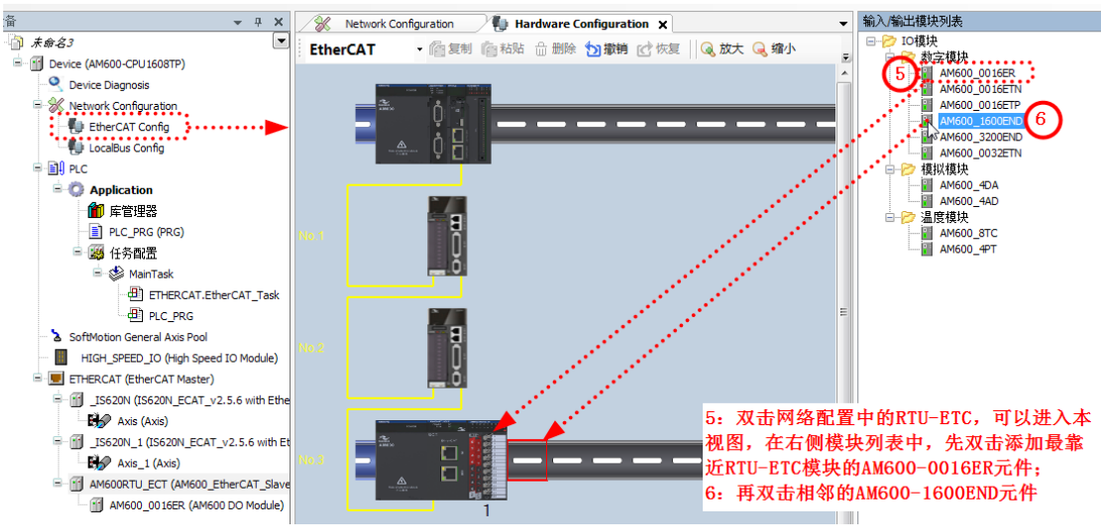
运行 AM600 编程软件 InoProShop，新建一个用户工程，在如下图的画面中，双击左侧 Network Configuration，添加 EtherCAT 网络总线：



根据实际系统的设备接线顺序，在网络中依次添加 2 个 IS620N 伺服、1 个 RTU-ETC 远程模块（扩展机架），如下图：



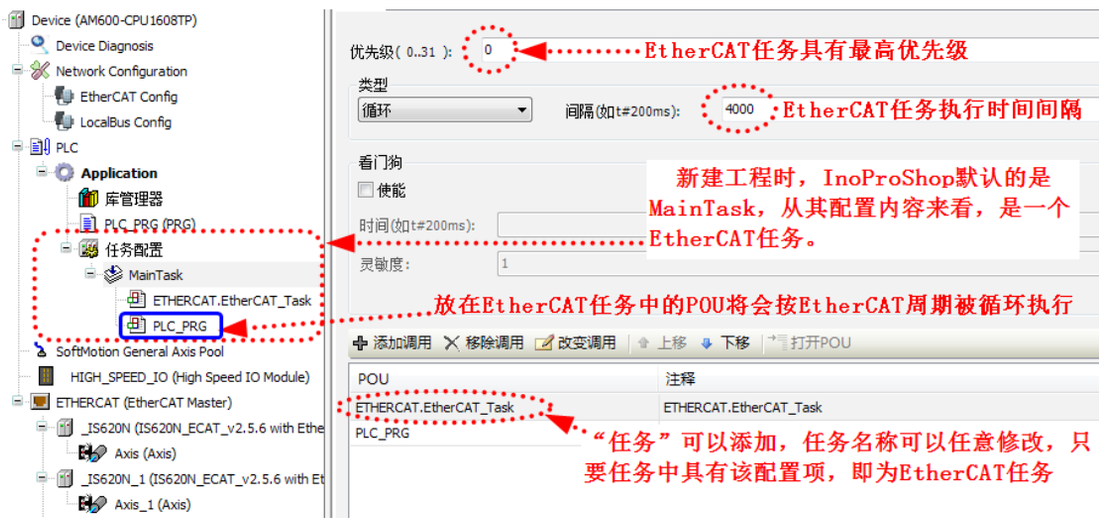
双击上图中的 RTU-ECT 模块，进入扩展机架的配置画面，在该画面中，按实际接线顺序，依次添加扩展 IO 模块，如下图：



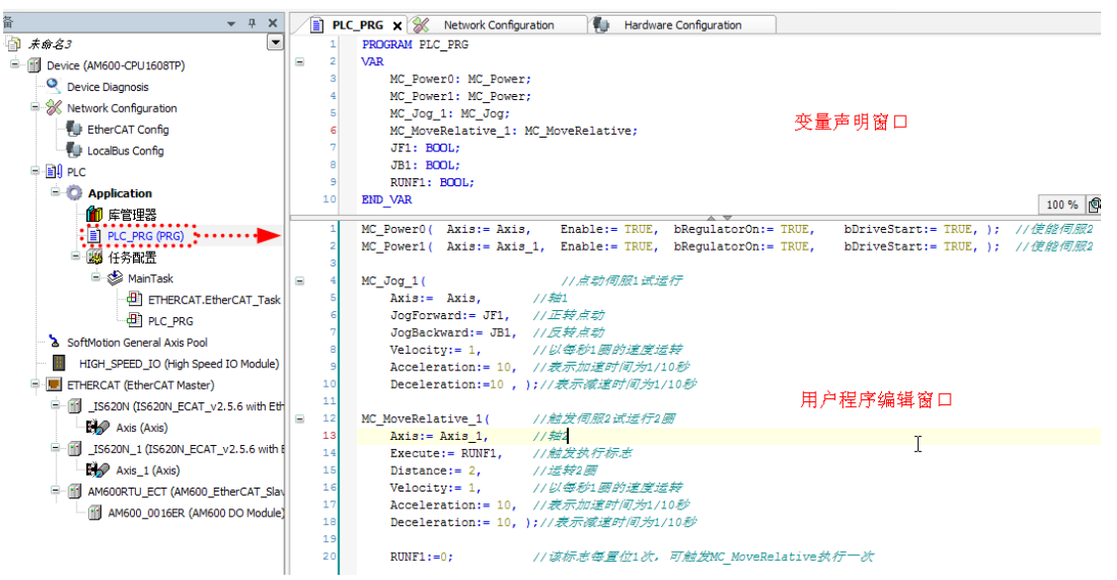
至此，在用户工程中与 AM600 实际应用的接线相同的硬件配置完成。

3.2.2 编写功能处理的 POU

先看一下 InoProShop 编程环境中的默认任务配置，默认有一个 MainTask 任务，点击查看其包含 EHERCAT. EtherCAT_Task 项目，因此这是一个 EtherCAT 任务，该任务下还有一个名称为 PLC_PRG 的 POU，在新建工程的同时创建了，我们希望的伺服控制程序代码就可以在 PLC_PRG 中来编写。

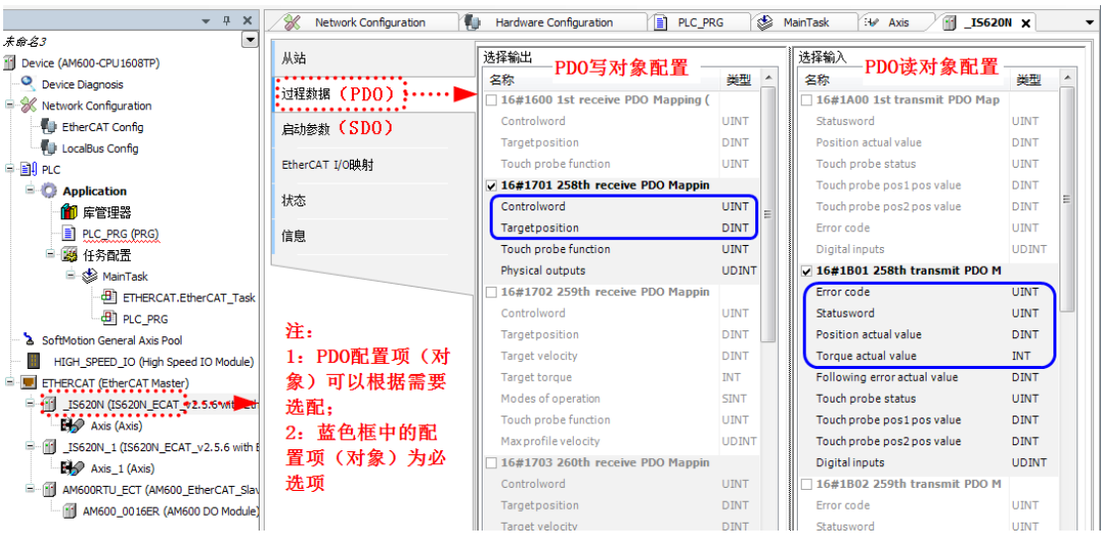


双击上图左侧的 PLC_PRG，进入该 POU 的编辑画面，输入如下图的内容，为了简练，MC 功能块实例中没有用到的变量没有显示：

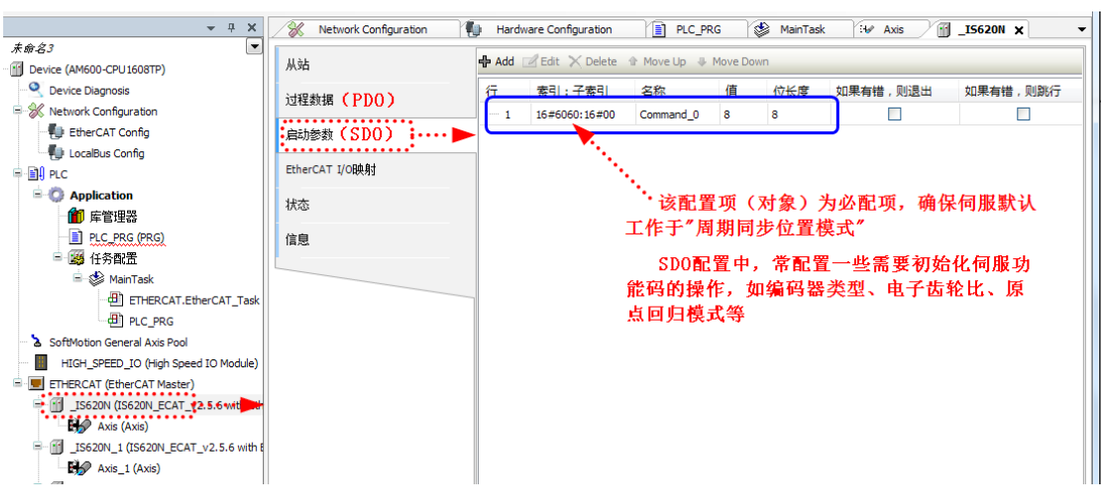


上面的就是要编写的 2 个伺服试运行代码，可令伺服 1 点动运行，每次置位 RUNF1 标志，可令伺服 2 旋转 2 圈。

为了达到这个目标，还需根据伺服驱动器作相应的配置，对 EtherCAT 主站通信 PDO 进行配置，如下图中被勾选的配置项，将来是每次 EtherCAT 通信中，都要进行交互的数据内容；



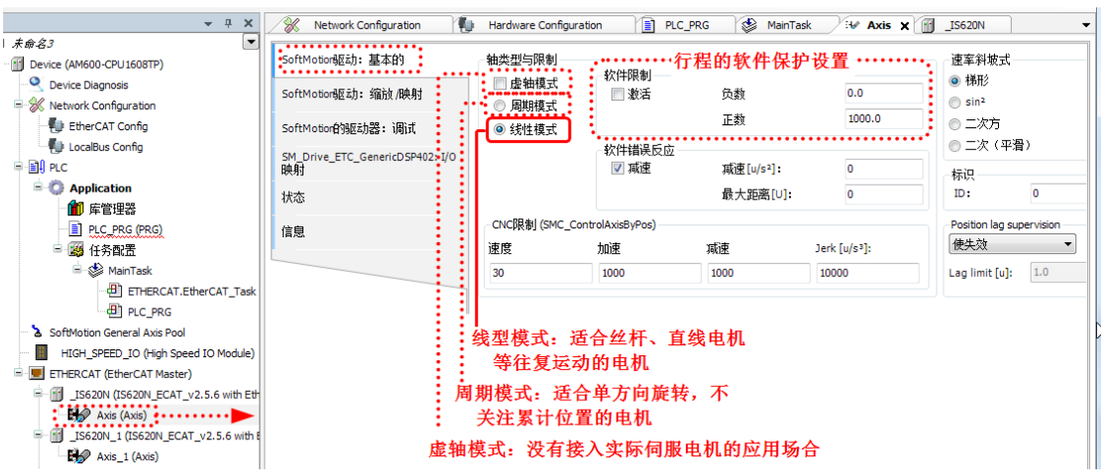
还有一些只需要 AM600 对伺服功能码进行改写操作的内容，就放在 SDO 配置中，如电子齿轮比、原点回归模式等，这些通信操作的是伺服的功能码，且只在上电后进行一次改写操作：



上图中，是将伺服的运行模式设置为“周期同步位置模式”，简单的讲，就是 AM600 控制器，在每次 EtherCAT 任务执行中，计算下一个周期点要求到达的位置（TargetPosition），发送给伺服驱动器，而伺服将根据这个距离 / 时间命令，完成到达下一个到达目标点的运动 .

3.2.3 电机参数的设置

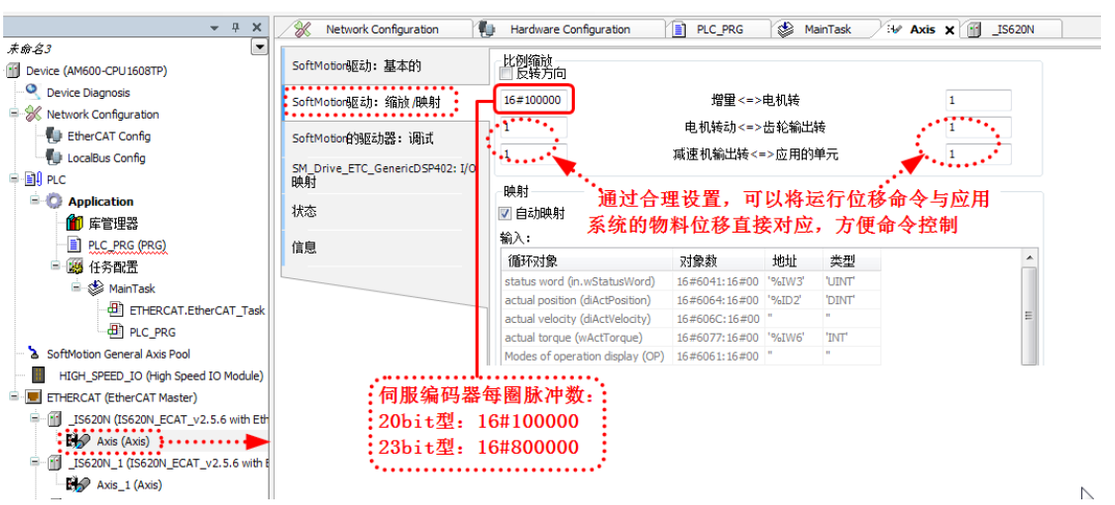
为了精确地控制运动位置，控制器必需准确计算伺服电机的位置，根据应用系统的运转特性、行程特点，选择“轴类型与限制”，以便控制器内部对读电机编码器反馈信息进行计算，得到准确位置，避免编码器脉冲数累积溢出造成的错误，如下图：



对于丝杆类型的往复运行机构，其行程是有限的，我们往往需要知道其在丝杆行程范围内的绝对位置，此时选择“线性模式”比较好；

若是单方向运转类型的转轴，采用线性模式容易出现位置计数溢出，导致位置计算错误，则选择“周期模式”比较好；

电机的编码器参数（如分辨率），应用系统的机械减速比可能各不相同，在编程时也需要根据实际情况进行设定，如下图：



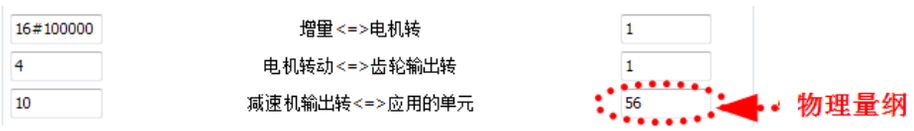
IS620N 伺服配套的电机有两种典型分辨率，普通增量式编码器为 20bit 分辨率，即每圈有 1048576 个脉冲数；而绝对编码器为 23bit 分辨率，每圈 8388608 个脉冲数。实际运行时，控制器以 EtherCAT 通信方式向伺服驱动器发送所需要运行的脉冲数，来控制伺服运行，因此编码器分辨率，需要根据实际情况准确设定，如上图。

例如上图中为 20bit 编码器，没有减速机的情况，当命令伺服运行 1 个单位时，伺服将会选择 1 圈（轴运动 360°）。

如果将上图中圆圈内“应用的单元”参数输入栏填写为 360，当命令伺服运行 1 个单位时，伺服将会选择 1/360 圈（轴运动 1°）。依此类推，按照实际机械结构的设定对应参数（俗称电子齿轮比）之后，就可以按照应用系统的运动距离物理单位输入 distance 命令了，使控制参数直观易懂。：

另外还需注意，上图中圆圈内的参数输入栏中只能输入整形数，因左右两边对应行中的参数之比为有效比例值，可以通过在左右两边对应行输入合适的整数值。例如伺服电机经过变比为 4:1 机械减速机构后，驱动导程为

5.6mm 的丝杆（即丝杆转动 1 圈，丝杆滑块运动 5.6mm）运动，设定如下图：



图中圆圈栏中参数的量纲，后续就可以作为 MC 控制命令中 distance 的参数量纲了。

上面说明的伺服驱动器、电机设置内容，在 Axis、Axis_1 两个轴的对应项中均需设置和核实，否则不会按所希望的特性运转。

举例：

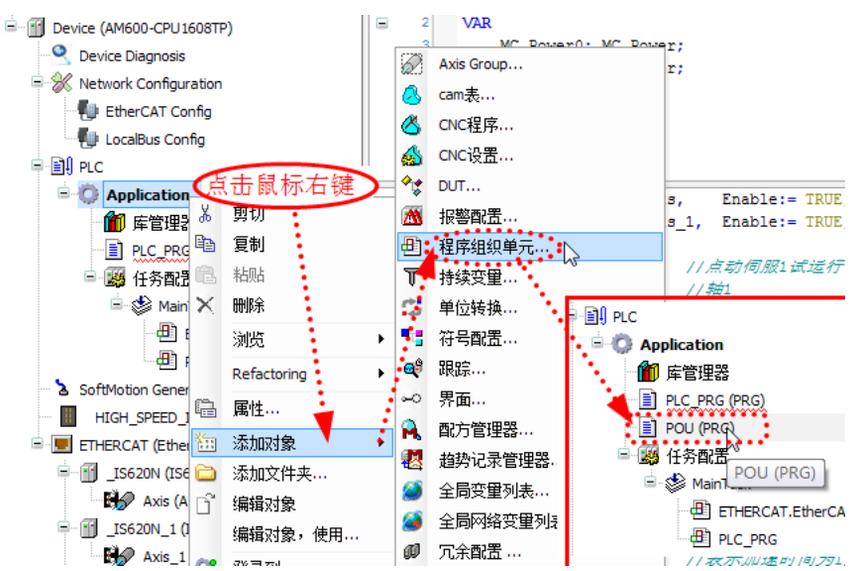
若设置好电机齿轮比之后，用户程序中的语句

MC_MoveAbsolute(Axis1:=1, Distance:=80.00);// 命令运行到 80.00mm 坐标处

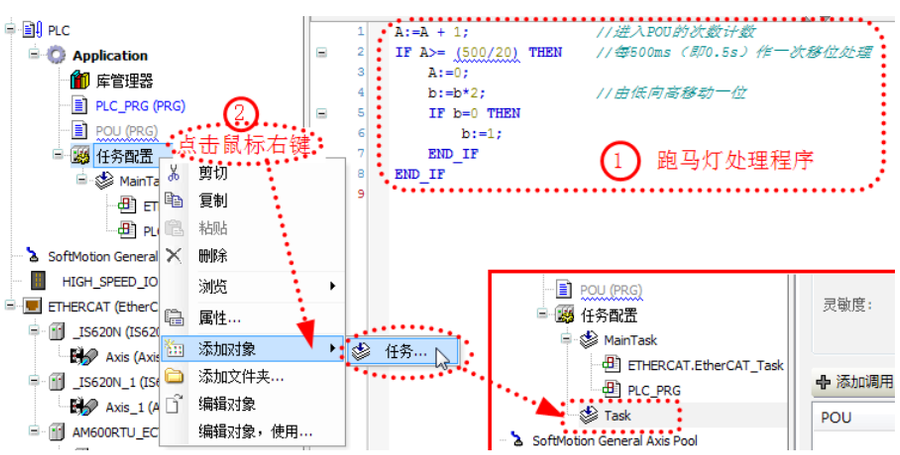
就可使得工件运动到 80.00mm 坐标处，位置命令单位就是设备的物理坐标单位，这可方便用户调试操作。

3.2.4 跑马灯控制逻辑的编写

相对于伺服轴的运动控制，跑马灯的逻辑控制程序执行需要的实时性就可以低许多，只要按每秒 2 次的 DO 端口变化即可，可以另外设置一个普通任务，每 20ms 执行一次对应的 POU，来进行移位刷新处理。先增加一个 POU，如下：



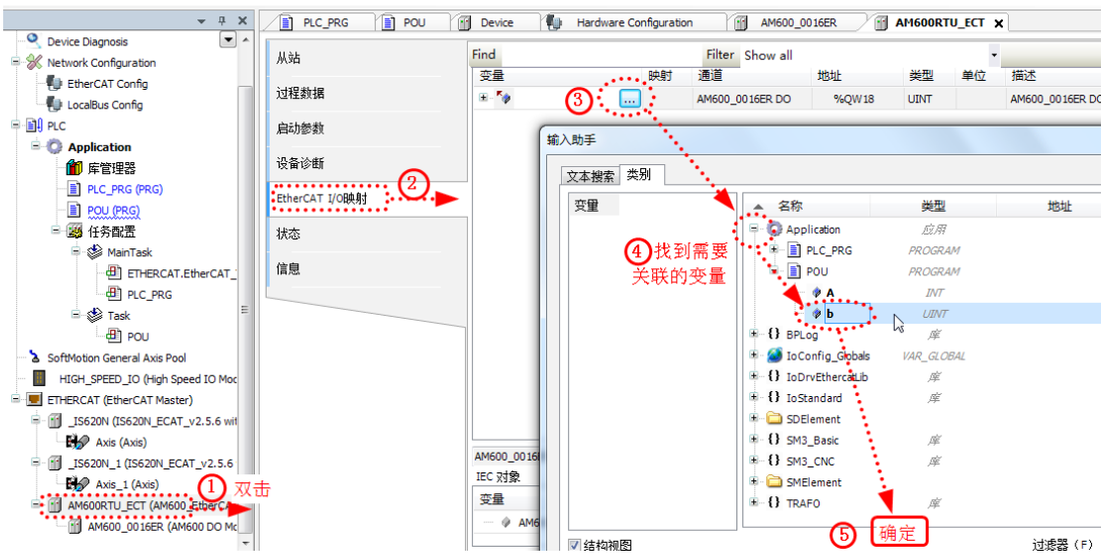
在 POU 中编写一个如下的程序，另外再增加一个任务：



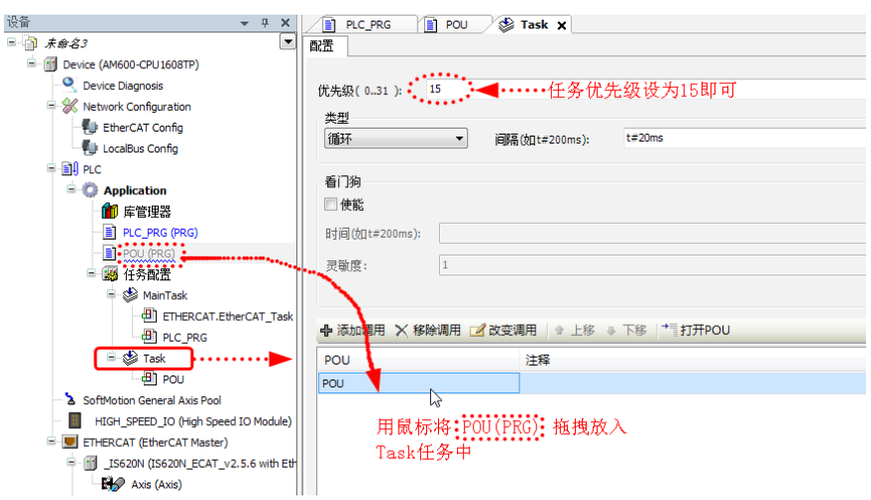
上图的程序采用 ST 语言编写，因 POU 每 20ms 执行一次，用变量 A 为计算执行的次数，每执行 25 次（即程序中的 500/20），把变量 b 扩大 2 倍，即将其二进制数由低向高移位 1 次，我们将变量 b 送给跑马灯输出端口，就可以实现跑马灯效果了。

3.2.5 将变量与硬件输出端口关联

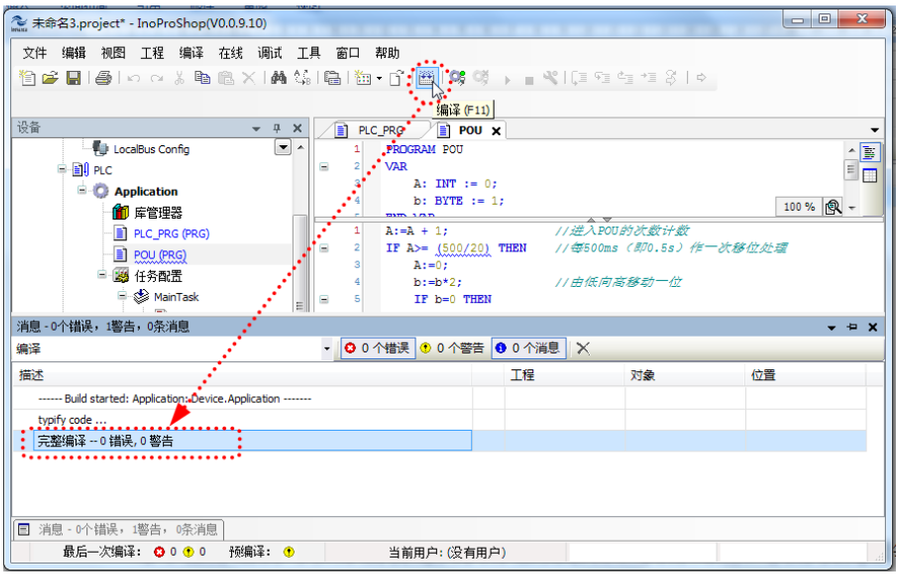
根据前面的要求，将刚才编写的 POU 程序中变量 b 与扩展 RTU-ETC 机架中 IO 模块端口关联，操作方法是先选择应用系统网络中 IO 的模块，再选择 IO 端口，在其 I/O 映射中，指明 POU 程序的变量。用户编写的程序变量都是在 Application\ 下面对应 POU 中选择的，如下图：



然后将跑马灯 POU 分配给新任务 Task 中，同时配置 Task 的任务执行，将其优先级设为普通优先级，比如 15；执行时间间隔设为 20ms：

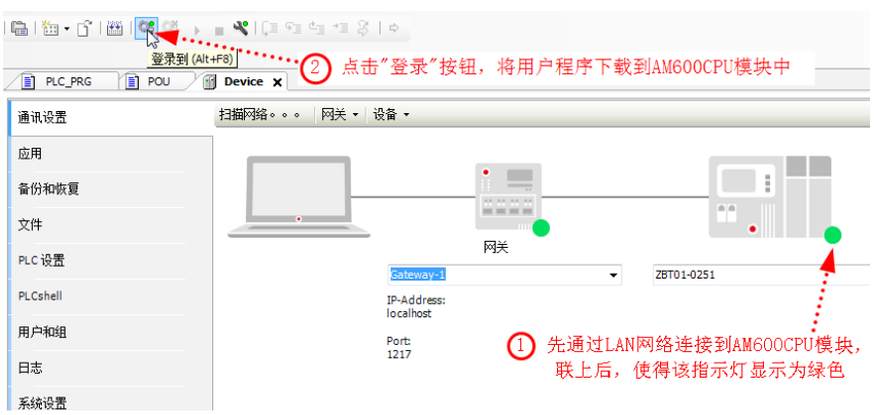


3.2.6 用户程序编译排错

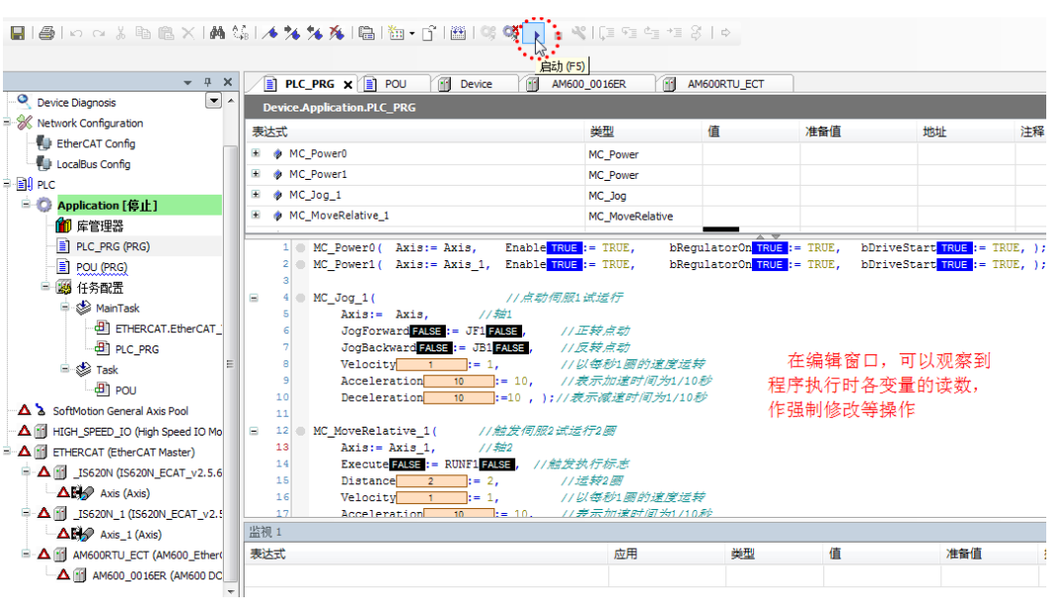


若有编写错误，上图中会列出错误类型与原因，双击其中的错误描述，光标会跳转到对应的程序编辑窗口，便于修订；逐一处理后，再进行编译，直到所有编译问题排除。

最后将用户程序下载到 AM600CPU 模块中，

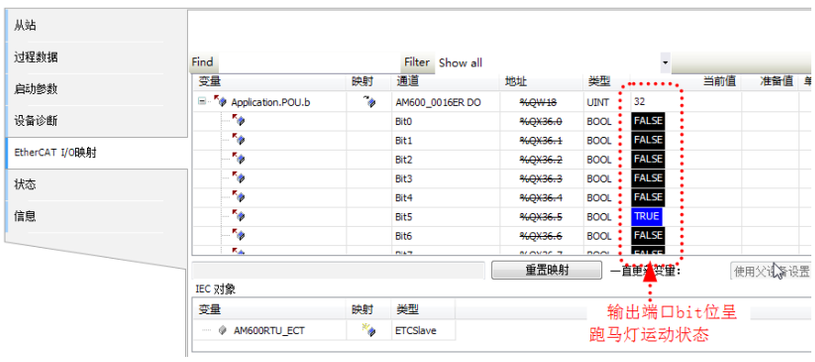


3.2.7 监控用户程序的运行



在监控画面，可以直观看到程序在执行情况，比如将变量 JF1 置 1，使得轴 1 点动运行，将之清 0 则轴 1 停止；每次强制变量 RUNF1 置 1，轴 2 就会运转 2 圈。

切换到 RTU-ETC 扩展模块画面，就可以看到所要求的 IO 输出端口呈跑马灯切换状态：



至此，编程所需的伺服点动、触发运行 2 圈的功能都已实现，一个简单的编程过程就完成了。

3.2.8 编写一个 AM600 运动控制工程的典型步骤总结

从上面的举例来看，编写具有 MC 运动控制功能的用户程序，一般需要有如下的内容处理，

应用系统硬件配置：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；

用户程序编写：根据所需实现的控制功能，将运动控制用一个 POU 编写（如 POU1），将普通逻辑控制用 POU 编写（如 POU2）；

伺服驱动器参数配置：根据硬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；

伺服电机参数配置：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；

任务安排：按控制的实时性要求，将运动控制功能 POU1 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 POU2 放在普通任务下，周期可设为 20ms，优先级为 16；

在线调试：将 AM600 控制器通过 LAN 网络与 PC 相连接，接线无误后上电，下载调试用户程序，排除用户程序 Bug；若有条件，可将伺服驱动系统接入 AM600 控制器，再进行调试。若手头没有伺服系统，可以将伺服设为虚轴；若手头没有 AM600 控制器，还可以在 PC 上仿真调试用户程序，先排除用户程序中可能的错误，直达满意为止。

Memo NO. _____

Date / /

第4章 AM600运动控制程序执行机制

四、AM600 运动控制程序执行机制

4.1 AM600 用户工程中的任务与配置

由上例中看到，每个任务组的可以分别设定其执行触发条件、执行周期、执行优先级等，AM600 支持的任务类型如下：

任务执行类型	执行特点	任务举例
循环	在设定的每个时间间隔内执行对应的 POU 一次	EtherCAT 总线任务 CANopen 总线任务 普通任务循环
外部事件	可以配置为当 HSIO 状态变化，或高速计数器的读数匹配时，引起对应的 POU 执行一次	HSIO 端口状态中断响应任务 HSIO 端口计数器中断响应任务
惯性滑行（飞轮）	一旦开始执行，即反复循环执行，没有停歇间隔	普通任务循环
事件	在设定的 Bool 型变量状态 0　1 的情况下触发执行一次，而其他的状态组合 0　0,1　1,1　0 则不会触发执行	软中断处理 POU
状态	在设定的 Bool 型变量的状态为 1 的情况下，反复循环执行	条件执行任务 POU

1）　任务配置的要点

设为“循环”类型，“任务周期”是指执行该类任务的时间间隔，一般的逻辑控制，普通 IO 端口变量状态变化比较慢，任务周期设定周期可以比较大一些，比如任务周期可设为 20ms；若需要及时处理的任务，任务周期设定周期则可以小一些；

EtherCAT 总线通信的任务配置，是一个比较特殊的“循环”类型的任务，具有最高的优先级，其任务周期的设定值，也将是 EtherCAT 总线的通信周期，一般为设为 1ms~4ms；设定值越小，运动控制的精度越高；当所需控制的轴数越多，则需要设定周期越大，否则容易导致 CPU 的计算负荷过载；

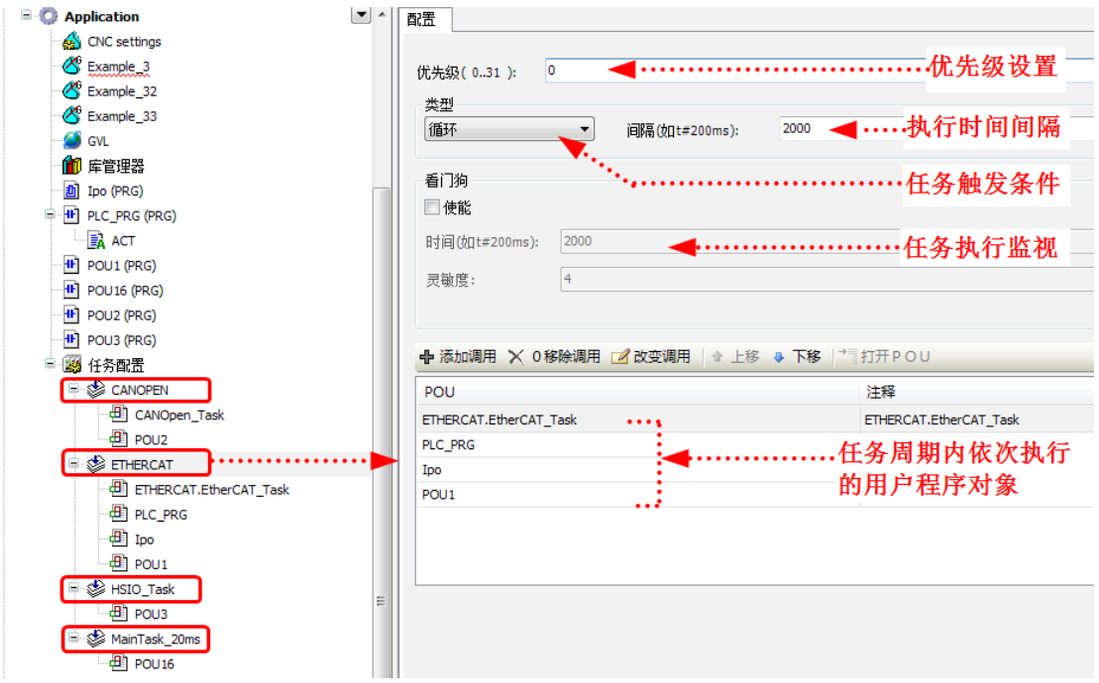
同样的原理，CANopen 总线通信的任务配置，是另一个比较特殊的“循环”类型的任务，具有第二优先级，其任务周期的设定值，也将是 CANopen 总线的通信周期，一般为设为 4ms~8ms；设定值越小，运动控制的精度越高；当所需控制的轴数越多，通信所需的时间更长，则需要设定周期越大。

还有一些任务则在满足某些状态的情况下，才开始执行，例如由高速 IO 端口（HSIO）的状态变化，也称 HSIO 中断信号，引发任务执行，不是按时间间隔来执行；

一个任务配置，只能设为一种执行类型、时间间隔及优先级，若要得到不同的执行特性，可以通过添加多个任务配置；

一个任务配置内，允许包含有多个 POU，该多个 POU 将均以相同的时间间隔执行，执行时按该任务中 POU 添加顺序执行。

如下图中所示，在 InoProShop 编程环境的“任务配置”下有 4 个任务。双击已经存在的任务，右侧窗口可以看到任务的设置参数：



鼠标选中“任务配置”，点击右键，可以增加新的任务。

请注意，上图中具有“ETHERCAT.EtherCAT_Task”项目的任务，将会是 EtherCAT 总线任务，其任务的优先级应设为 0.

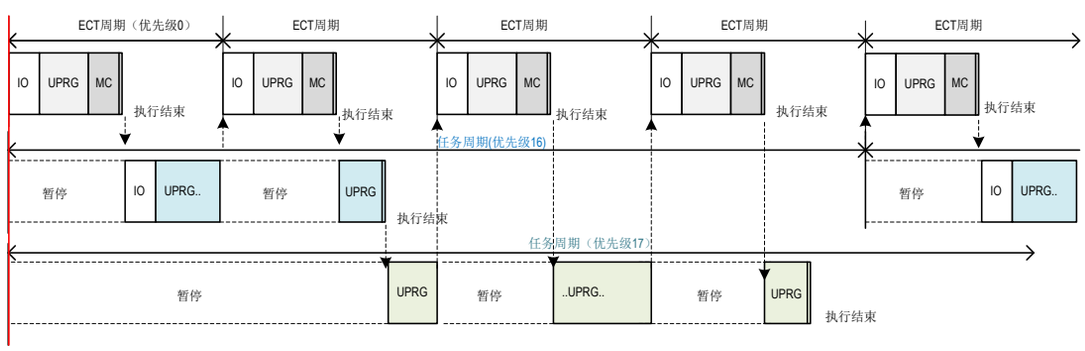
2） AM600 任务的优先级安排

AM600 控制器的系统软件，默认对不同类型的任务配置，安排了不同的优先级，确保对运动控制等重要任务优先执行，在一些需要高性能运动控制（MC）的应用中，可让控制器的性能得到合理利用。

任务优先级顺序如下，（用户不要强行修改优先级顺序）：

默认优先级	任务类型	描述
0	EtherCAT 总线任务	优先级最高，只允许有一个 EtherCAT 任务
1	CANopen 中断任务	优先级次之，只允许有一个 CANopen 任务
2	HSIO 中断任务	优先级再次之，每个 HSIO 输入端口事件允许有一个 HSIO 任务
3	ModbusTCP	
4	ModbusRTU	
16	MainPOU	优先级最低，最多可允许 4 个周期不同的任务

优先级设定值越小，优先级越高；高优先级的 POU 可以打断低优先级 POU 的执行，如下图，其中 ECT 代表 EtherCAT。



从上图可知：

控制器执行任务时，有一个用户观察不到的时间对准点，如上图左侧，在这个时间点开始，按最高优先级次高优先级 .. 最低优先级的顺序开始执行；

低优先的任务在执行时，可能被高优先级的任务打断，待高优先级任务执行完成后，返回被打断的任务，继续执行该低优先级任务；

EtherCAT 任务为最高优先级任务，按 EtherCAT 周期进入该任务，完整执行一遍该任务内的所有 POU 后，才返回较低优先级任务；

3） 任务配置中的执行周期设定的要求

AM600 系统软件采用多任务方式来执行用户程序的“任务”，而每个“任务”分配了不同的执行周期，有些全局变量可能要在不同的 POU 之间被访问和修改， 于是需要对全局变量进行交互同步，也是在任务的“时间对准点”进行的，在设置循环类型任务的周期时，呈整数倍数的关系。

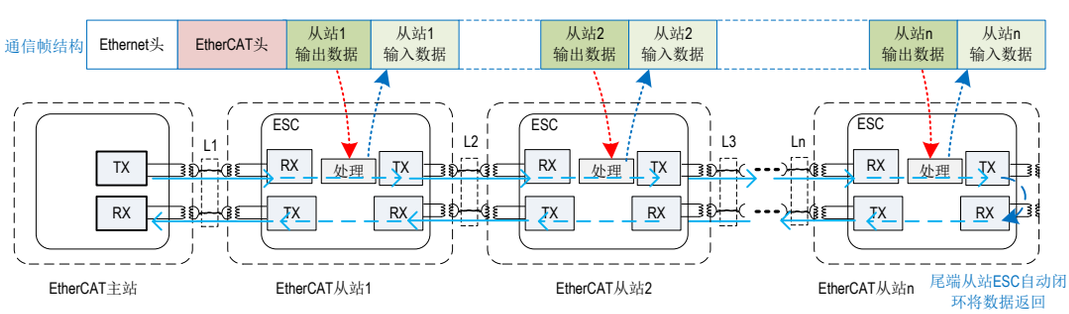
比如，设 EtherCAT 周期为 1ms、2ms 或 4ms，而普通循环任务的周期为 20ms，更低优先级的循环任务周期为 100ms 等等。EtherCAT 周期不要设为 3ms、6ms、7ms、9ms 等，容易造成非整数倍的关系。

4.2 EtherCAT 总线网络的数据流分析

1） EtherCAT 总线的网络简介

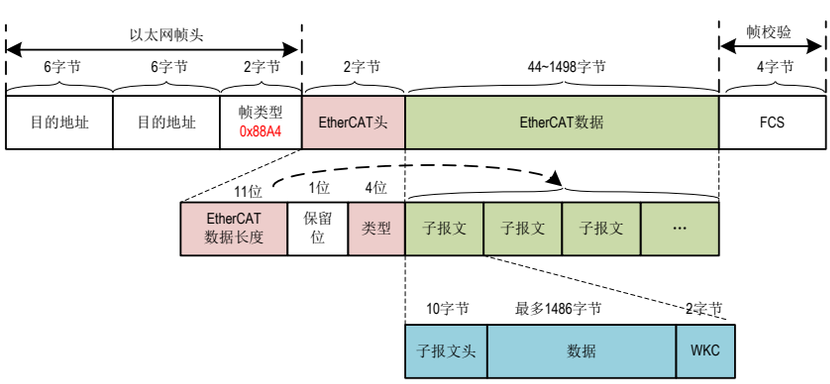
EtherCAT 总线采用常见的连接形式是采用 RJ45 插座、多芯以太网电缆连接，推荐采用超 5 类电缆，可以提高网络的抗干扰能力。

与通用的以太网络相似，网络通信速率为 100Mbps，每个相邻从站的链接电缆长度可达 100 米等。 网络内部的等效连接关系与通信数据的流动如下图。



EtherCAT 网络有与普通以太网明显不同的方面，在一个网络里只有一个 EtherCAT 主站，另外是从站内部的网络专用控制芯片 ESC（EtherCAT Slave Controller），能在一个通信帧中实时收取发给本站通信数据、将本站的应答数据插入该数据帧，使得主站在一个通信帧中即可完成多个从站的访问，大大提高了通信效率。

EtherCAT 总线中的通信数据帧沿用了以太网数据 UDP/IP 帧结构，帧结构类型 0x88A4，只是中间的数据字段需要按 EtherCAT 通信协议进行制备和解析，如下图所示。



其中的 EtherCAT 数据段可以进一步按“EtherCAT 帧”，以某种协议来进行定义和解析，只要主站和从站都遵守这个协议，就可以实现数据通信了。一般采用的协议有 CANopen Over EtherCAT（CoE），还有 Sercos Over EtherCAT（SoE）等，正如在 TCP/IP 网络上传输 Modbus 协议帧数据（ModbusTCP）一样。

AM600 控制器采用的是 CoE 协议，也就是应用层协议为 CANopen 协议的 DS402 行规（也称 CiA402），是一种用于伺服运动控制类的专用协议，该协议最重要的几个特点是：

① 为了提高了通信效率，主站 - 从站不是采用问答的方式来访问，而是在总线网络初始化阶段，主站事先把需要发送的数据项目清单给从站，例如“过程数据 PDO”，告知其主站将会发送的数据项目及顺序（TPDO）、要求从站发送的数据项目及顺序（RPDO），这样从站在接收到主站数据帧后知道如何解析，也可以事先准备好所需要的应答数据；

当主站的数据帧到来时，每个从站的网络控制芯片（ESC）就可以拿取发给本站的数据段，供本站的处理器按照配置表进行解析，并在 EtherCAT 通信帧的合适阶段及时插入本站的应答数据块，返回给主站；

② 将用户所需通信的数据按照实时性要求，分为“过程数据 PDO”和“服务数据 SDO”，前者 PDO 安排为高频度周期性循环收发，后者 SDO 则只在有需要的时候才进行通信；

③ 伺服驱动器的控制命令参数、运行状态参数、功能码设置参数，最多者达到数百个，每个品牌的伺服参数的命名方法各不相同，为了保证不同品牌的主站、从站都具有互换性，CiA402 协议中制定了一个“对象字典 OD”，将伺服驱动器中所要用到的所有功能码、运行命令及其设定值意义、运行状态参数及量纲等，都作了具体定义，形成了一个专业的技术规范，不同品牌的设备供应商只要按照这个 CiA402 协议规范开发的产品，能保证通用性和互换性，就可以与 AM600 配合运行；

④ 主从站之间的通信对象配置，是保证运控功能块功能成功执行的条件。控制器在执行用户程序中 MC 功能块时，内部需要使用具体的“通信数据对象”来向伺服从站发送命令、读取从站轴状态，编程人员需要在 TPDO、RPDO 中填入所需要用到的数据对象，主控器才能对伺服从站进行控制，不要有遗漏；

⑤ 从站设备或许并不支持“对象字典 OD”中的所有项目定义，设备厂家有针对该设备的“设备描述文件 EDS”进行了定义，编程用户在对该设备配置之前，需要在 InoProShop 中导入该从站设备的设备描述文件 EDS，就可以看到所支持的对象内容了；

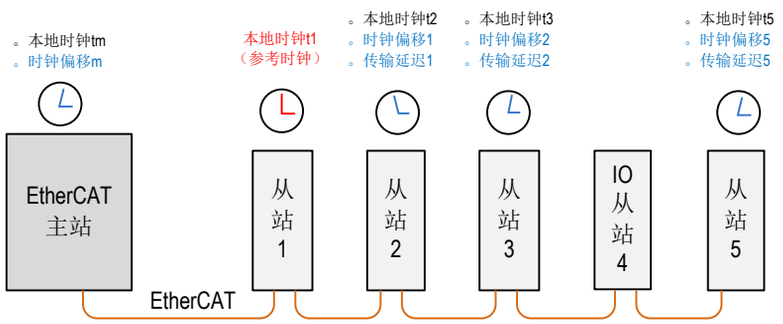
⑥ 用户在编写用户工程时，按控制需要，选择和配置 TPDO、RPDO 数据对象表，运行时主站会自动以通信方式转发给对应的从站；尽量只选择需要用到的配置项，减少无关数据对象的配置项，可以减轻 EtherCAT 通讯的负荷，利于提高通信效率；

⑦ 服务数据对象 SDO 配置项目一般用于系统运行之初对从站设备的功能码初始化，运行过程中也可以通过 MC_SDOread 之类的功能块进行参数的访问，其通信及时性会比较低，而且会占用额外的 EtherCAT 通信开销，对于总线负载率比较高的应用中，还会引起同步超时故障，使用要慎重；

我们理解了 CiA402 对象字典 OD、常用 MC 功能块所需要用到的从站参数对象，就可以合理配置 PDO、SDO 表了。

2） EtherCAT 总线的同步时钟概念

作为多轴运动控制的网络，往往需要让多个从站同时开始或停止运动，EtherCAT 网络就具有分布式时钟（Distributed Clock，简称 DC）机制，使得每个智能从站（如伺服驱动器、智能高速扩展模块），有一致的时钟，每个从站按设定的同步触发周期，将主站写入的数据输出给执行单元，达到同时运行的目的。



如上图的网络，每个智能从站内部都有一个高分辨率的内部本地时钟 TLocal，在 EtherCAT 总线初始化阶段，主站会读取每一个从站的当前时间，将第一个从站的本地时间作为网络的“参考时钟”，即可算得每个从站相对于参考时钟的“时钟偏移 Toffset”，将每个从站的时钟偏移写给对应的从站，以便其修正时钟，消除静态误差；

另外在通信数据帧传输的过程中，还会有因硬件网络产生的传输延迟时间，主站会通过发送特定的广播帧，让每个从站记录数据到达时刻，主站再读取每个从站所记录的时刻值，同时测量数据返回数据帧的总延迟，可以准确计算出每个从站的“传输延迟 Tdelay”，之后主站将每个从站的传输延迟时间写入每个从站的内存，从站在有了这些时钟修正值后，通过计算 TLocal-Toffset-Tdelay，就得到与参考时钟 t1 相同的时钟了。

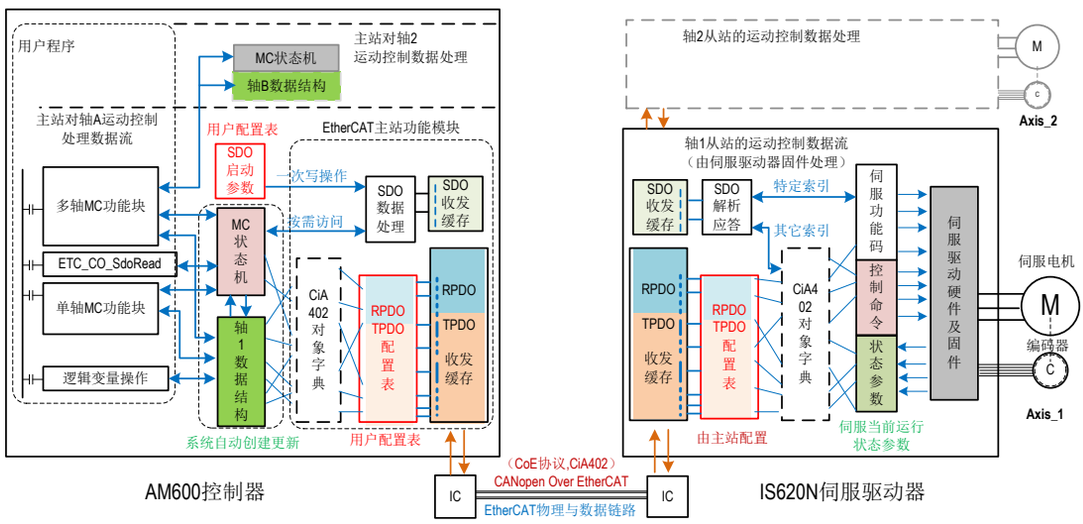
EtherCAT 网络中，对 DC 时钟并不敏感的 IO 从站，可以不设置 DC 处理，EtherCAT 主站在 DC 校准时，会忽略其时钟的校对。

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

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

4.3 AM600 与伺服从站的通信数据流程

正如前面提到的，AM600 的 EtherCAT 通信中，应用层采用 CoE，控制器在执行 Motion Control（MC）用户程序时，控制器系统软件与伺服的通信数据经过了多级功能单元的处理，流程如下图所示：



上图左侧为 AM600 控制器，右侧为 IS620N 系列伺服驱动器及其配套伺服电机，控制器和驱动器的网络接口和协议 PLCopen 规范，两者均可以与第三方设备兼容，并能互换使用，因此内部的通信数据流程也适用于采用 CiA402 协议的第三方设备，我们理解了每个功能块的功能特点和用法，就掌握了基于 EtherCAT 网络的运动控制的原理及方法。

运动控制指令的源头是用户程序中的 MC 功能块，控制操作的对象是伺服轴，运控轴的状态以 “轴数据结构”的形式存放与主控器内容中，供用户程序访问。

4.3.1 控制信息流程步骤说明

第一步：执行用户程序 MC 功能块，处理需要发送的命令数据

控制器在执行用户程序时，执行了 MC 运动控制功能块实例，例如 MC_MoveRelative（Axis_1），控制器根据内存中的从站（Axis_1）状态机和数据结构：

检查从轴当前状态，若从轴未使能、或正以力矩模式运行、或以同步方式运行、或正在 Homing 运行中、或正在告警等，就报 MC 执行错误；

若从轴为停止状态、位置模式的非同步模式运行中，就发送使从轴运行命令 ControlWord；

分析从轴的当前运行速度 fActPosition，运行速度 fActVelocity，以及目标位置、允许最大速度、加速度、减速度等约束条件，计算得到下一个运行周期所需要的运动位置指令 TargetPosition；

控制器还需等待下一通信周期从站返回来的数据，分析判断本条 MC 功能块指令的执行情况，用户才能知道执行中（Busy）、已执行完成（Done）、出错（Error）、还是被其他 MC 指令中断了（Aborted），等待…

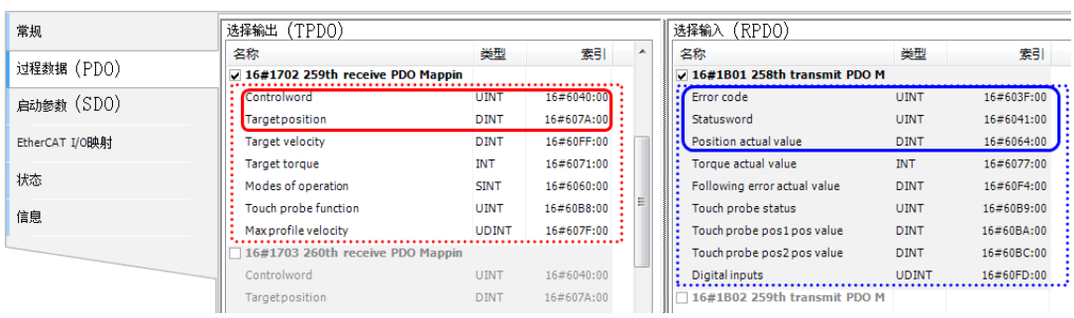
第二步：将需要发送的控制命令数据放置到 EtherCAT 发送缓存单元

将需要发送给从站 Axis_1 轴的命令数据 ControlWord、TargetPosition 存放到 PDO 发送缓存单元，完成这个操作需要的前提条件 PDO 配置表中已有这两个参数（在 CiA402 中称为“对象”）选项；

“PDO 配置表”中存放的是需要主站需要发送和读取的控制参数（对象）的“索引号”（主索引号 : 子索引号），

	用途说明
TPDO 配置表	该表需要用户编程时，根据对从站控制的需要循环发送的内容，进行配置的对象与属性清单表； 该表在网络初始化阶段控制器会自动发送给从站 ESC； 控制器主站将根据该表安排发送缓冲的大小，运行时将待发送的命令数据存入发送缓冲； 运行时从站根据此表解析所收到的数据帧； 每个从站可以有不同的 TPDO 配置表；
RPDO 配置表	该表需要用户编程时，根据需要从站自动应答的对象内容，进行配置的对象与属性清单表； 该表在网络初始化阶段会自动发送给从站 ESC； 运行时从站根据此表制备数据，在主站访问本从站时及时将数据插入 EtherCAT 数据帧的时隙，返回给主站； 运行时主站根据此表解析返回数据帧从站应答数据； 每个从站可以有不同的 RPDO 配置表；

下图所示为 InoProShop 中 PDO 配置表的画面，其中的每种控制参数的索引号、数据类型，均由 CiA402 行规规定，通过“索引号”我们可以在“对象字典 OD”中反查得知是什么参数、参数的宽度类型了。



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

TPDO 配置表中依次存放每个对象的索引号及数据宽度信息，各对象的在表中的顺序，将是系统将 MC 指令所要发送数据的放入发送缓存单元的依据。如上图将 ControlWord 放在第一个发送单元，将 TargetPosition 放在第二个单元，依此类推；

从站根据 RPDO 配置表（如上图中的 9 个“对象”），将每个对象的索引号及顺序，依次将伺服的运行状态数据，存放于应答缓存单元，当主站通信帧访问本从站时，ESC 自动将该缓存单元的数据插入到数据帧的合适时隙，返回给主站；

RPDO 表也将是主站解析从站应答数据的依据。

第三步：主站控制芯片将发送缓存单元单元的数据定时发送给从站 ESC，从站同时将应答数据

控制器作为主站，按照用户设置的 EtherCAT 时钟周期，定时产生 EtherCAT 中断，进入 EtherCAT 中断后，即发起 EtherCAT 通信，将 PDO 发送缓存单元的数据，通过一帧或若干帧，发送给多个从站，顺便将每个从站的应答数据在同一个通信帧里取回来。

从时间顺序来看，控制器发送缓存单元的数据是由前一次 EtherCAT 中断 POU 执行所产生命令数据；

从站的应答数据，并非是主站询问后的回答，而是根据 RPDO 配置中要求的循环回复“对象”的当前值；

第四步：从站接收和解析主站发送给本站的数据

进入网络正常运行状态后，从站 ESC 会定时收到主站发送的通讯数据帧，并自动将通讯帧中发给本站的数据存入到本地缓存，

		ControlWord	TargetPosition	TargetVelocity	TargetTorque	Modes of operation	Touchprobe function	Maxprofile velocity		Error code	Status Word
主站TPDO配置表	对象索引号	6040：00	607A: 00	60FF: 00	6071：00	6060：00	60B8：00	607F: 00	RPDO配置表	603F：00	6041：00
	类型	UINT	DINT	DINT	INT	SINT	UINT	UDINT		UINT	UINT
	数据宽度	16bit	32bit	32bit	16bit	8bit	16bit	32bit		16bit	16bit
从站1对接收数据串解析		F1A4	5B32 05C7	01F2 3201	01E2	08	04A0	1008 FA11	(应答数据)	4C87	5DF1

从站的处理器在接收到了一串 PDO 数据后，就按 TPDO 表格，将所接收到的数据串按指定对象数据类型（宽度）依次截取，按“对象索引”号所代表的参数属性，存放到相应的控制命令单元，用于伺服的运行控制；

从站的处理器会根据 RPDO 的配置表所要求的对象属性和顺序，将本伺服轴的当前运行状态和参数，循环刷新本地 ESC 中应答缓存单元，在合适的通信帧时隙，ESC 以高速硬件操作将该缓存的数据插入 EtherCAT 通信帧中，“发送”给主站。

第五步：主站接收和解析从站应答的数据，更新轴状态参数，判断是否执行完成

控制器作为 EtherCAT 网络主站，在发送数据帧的同时，会同时接收由从站网络闭环反送回来的通信帧，从中可以提取各从站应答的数据串，还可以判断网络的通信状态，分析通信操作成功与否。

根据从站应答的数据，如 Error code、Status Word、Position Actual Value…等，控制器系统就可以判断是否已经到达 MC 功能块实例所要求的运行位置，刷新该 MC 功能块实例的输出变量状态；

另外，控制器系统软件会及时更新该轴状态参数的数据结构，供用户程序访问，这是 AM600 系统软件最强大的内在功能之一。

以上是 AM600 系列控制器的 EtherCAT 数据组包发送、接收解析的原理流程的说明，便于用户理解其内在机制，其中许多环节是系统自动完成，并不需要用户干预，用户只需理解 CiA402 对象概念，掌握伺服轴常用的 “对象”类型，进行 TPDO、RPDO 配置表的对象选择就可以了。

4.3.2 CiA402 数据对象字典与伺服驱动器常用对象

AM600 的 EtherCAT 总线上的协议应用层采用的是 CoE 协议，即 CANopen Over EtherCAT，因此有必要先理解 CANopen 协议及其应用方法。

CANopen 是一个通用的协议标准，为了适应不同类型设备的通信控制，分别制定了不同的“行规”子集，例如:

CiA401 I/O 模块行规

CiA402 伺服与运动控制

CiA403 人机接口行规

CiA404 测量设备与闭环控制行规

CiA406 编码器行规

CiA408 比例液压行规

AM600 的 EtherCAT 总线通信应用层采用的是 CANopen DS402 协议，也称 CiA402 协议，它属于 CANopen 协议的“伺服与运动控制”行规，该协议行规广泛应用于基于 CAN 总线、EtherCAT 总线网络的运动控制。不同设备厂家按照这个协议规范开发的控制器、伺服驱动器（从站设备）均可配合运行，或可替换使用，给用户增加了选择机会，符合 PLCopen 规范的宗旨。

CiA402 的核心包括如下几个部分：

定义“对象字典 OD”及其“对象”的功能属性，统一通信数据解析方法；

周期性过程通信数据，采用先发过程对象配置，后续则按配置帧结构，周期性地发送对象参数的通信方式；

偶发性的数据通信，采用附加通信字段的方式，进行问答式的通信；

网络通信分为若干个运行状态，便于主从部分进行通信的初始化操作，诊断通信异常的原因，恢复网络通信等;

CiA402 协议将伺服运动控制应用中，具有代表性的设置参数、控制参数、状态参数等项目，归纳整理为具有固定编号（索引号 + 子索引号）的“对象”，完整的对象定义表就是“对象字典”。

类似的定义方法在其他设备也有使用，如 PLC 控制器模块的 BFM 区地址定义方法，如电机驱动器的功能码定义方法等，都是将不同功能参数给与不同的序号来定义，不难理解。

CiA402 对象类型按属性分成索引号段，如下表：

主索引号范围	含义	说明
0x0000~0x1FFF	协议类型描述、制造商信息、行规类型描述、配置表描述信息等	有设备制造商进行信息初始化，配置信息由系统软件自动完成，无需控制器用户干预。
0x2000~0x5FFF	制造商定义对象， 由设备商自行定义对象的功能属性	设备商可以将主索引号设计成伺服驱动器的功能码，用设置功能码参数，用于静态参数的设置
0x6000~0x9FFF	行规定义数据对象，用于设备的控制与监视	控制器与伺服进行控制的通信交互的数据
0xA000~0xFFFF	保留	

由上表可以看出，运动控制所需的对象，主要在（0x6000~0x9FFF）索引号段部分，如果是希望 SDO 配置修改伺服功能码，需要关注（0x2000~0x5FFF）索引号段部分。

我们把对象字典看作是一套可以由 EtherCAT 总线通信访问的伺服驱动器功能码定义，这样更好理解。

运控应用中常用的数据对象：

AM600 控制器控制伺服的运行，大致上有这样的几种命令类型：

控制伺服的运行状态，如使能、原点回归、启停运行、告警复位等；

设定伺服的运行模式，如设为位置模式、速度模式、力矩模式；

设定伺服运行的目标位置、运行速度、输出力矩；

读取伺服系统的运行状态，如运行状态、运行模式、位置、当前速度、输出力矩等；

设定或修改伺服系统的功能码参数，运行约束参数等。

完成这些控制操作，有几个常用的数据对象，编程时必需在 PDO 或 SDO 配置表中设定，还有一些数据对象，则是根据用户程序中使用到的功能，才酌情添加。

本节介绍的数据对象的取值，用于解释该对象的功能定义，实际运行时，控制器会根据所需的控制操作，自动发送对应的取值：

对于 PDO 配置表，用户只需添加控制器运行时需要用到的数据对象，不需要填写具体的参数值或变量名，InoProShop 软件编译时，会自动将 MC 功能块中的变量与 PDO 的对象进行关联；

对于 SDO 配置表，一般用于控制器初始化伺服功能码的操作（写操作），其中的写入值为确定的常数量，因此填写的常数必需符合 DS402 规范定义，有些则是按伺服驱动器内部特有的功能码定义值常数。

1） 控制字 6040H(Control word)

该对象是主控器单元控制伺服的运行状态的命令字，用于控制伺服的使能、启停、告警复位等运行状态，这是最基本的控制命令字，因此，6040H（Control word）是 PDO 配置表的一个必选项。

索引号	6040H
对象名称	Control word
数据结构	VAR
可访问性	RW
数据类型	UNSIGNED16
数据范围	0~65535
默认值	0
可访问性	RW
PDO 可映射性	YES
相关模式	ALL

该对象是主控器单元控制伺服的运行状态的命令字，其设定值有明确的定义：

bit	名称	描述
0	伺服准备好	1- 有效，0- 无效
1	接通主回路电	1- 有效，0- 无效
2	快速停机	1- 无效，0- 有效
3	伺服运行	1- 有效，0- 无效
4~6		与各伺服运行模式相关。
7	故障复位	对于可复位故障和警告，执行故障复位功能 bit7 上升沿有效； bit7 保持为 1，其他控制指令均无效。
8	暂停	各模式下的暂停方式请查询对象字典 605Dh。
9~10	NA	预留
11~15	厂家自定义	预留，未定义

注意：

控制字的每一个 bit 位单独赋值无意义，必须与其他位共同构成某一控制指令。

bit0~bit3 和 bit7 在各伺服模式下意义相同，必须按顺序发送命令，才可将伺服驱动器按照 CiA402 状态机切换流程引导入预计的状态，每一命令对应一确定的状态。

bit4~bit6 与各伺服模式相关 (请查看不同模式下的控制指令)

2） 目标位置 607AH（Target Position）

由主控器发送的伺服运行的目标位置命令。伺服多数情况下以位置模式运行，在 AM600 的 MC 应用中，多以周期同步位置模式（CSP），控制器命令伺服在下一 EtherCAT 周期运行的目标位置，其量纲为用户设置的物理量纲。在对应轴数据结构变量 Axis.fSetPosition 可监控得到。

因此，目标位置对象 607AH（Target Position）是 PDO 配置表的一个必选项。

索引	607AH
名称	目标位置 Target Position
数据结构	VAR
数据类型	INTER32
可访问性	RW
PDO 可映射性	YES
数据范围	0x80000000~0x7FFFFFFF
默认值	0
单位	指令单位
相关模式	PP CSP
描述 Comment	

设置轮廓位置模式与周期同步位置模式下的伺服目标位置：

6040h 的 bit6	描述
0	607A 是当前段的目标绝对位置； 当前段定位完成后，位置反馈 6064 = 607A
1	607A 是当前段的目标增量位移； 当前段定位完成后，位置反馈增量 = 607A

3） 伺服预运行模式 6060h（Modes of operation）

主控制器可通过对象 6060h 对伺服运行状态进行设置，选择运行模式模式选择

索引	6060H
名称	模式选择
数据结构	VAR
可访问性	RW
PDO 可映射性	YES
数据类型	INTEGER8
数据范围	0x00~0x0A
默认值	0
相关模式	ALL

选择伺服运行模式：

设定值	伺服模式	AM600 支持
0x00	NA	-
0x01	PP(轮廓位置模式)	-
0x02	NA	-
0x03	PV(轮廓速度模式)	-

设定值	伺服模式	AM600 支持
0x04	PT(轮廓转矩模式)	支持
0x05	NA	-
0x06	HM(回零模式)	支持
0x07	IP(插补模式)，不支持	-
0x08	CSP(周期同步位置模式)	支持，默认运行模式
0x09	CSV(周期同步速度模式)	-
0x0A	CST(周期同步转矩模式)	-

伺服运行模式切换是的注意事项：

伺服驱动器处于任何状态下，从轮廓位置模式或周期同步位置模式切入其他模式后，未执行的位置指令将被抛弃。

伺服驱动器处于任何状态下，从轮廓速度模式、轮廓转矩模式、周期同步速度模式、周期同步转矩模式切入其他模式后，首先执行斜坡停机，停机完成后，可切入其他模式。

伺服处于回零模式，且正在运行时，不可切入其他模式；回零完成或被中断 (故障或使能无效) 时，可切入其他模式

伺服运行状态，从其他模式切换到周期同步模式下运行时，请间隔至少 1ms 再发送指令，否则将发生指令丢失或错误。

4） 目标速度 60FFh（profile velocity）

当伺服以速度模式运行时（轮廓速度模式与周期同步速度模式），需要设置该速度指令。

索引 INDEX	60FFH
名称	目标速度 profile velocity
数据结构	VAR
数据类型	INTER32
可访问性	RW
PDO 可映射	YES
数据范围	0x80000000~0x7FFFFFFF
默认值	0x64
单位	指令单位 /s
相关模式	PV CSV

5） 目标转矩 6071h（Target Torque）

当伺服以转矩模式运行时（轮廓转矩模式与周期同步转矩），需要设置该转矩指令，量纲为电机额定转矩的百分比（0.1%）。

索引	6071H
名称	目标转矩 Target Torque
数据结构	VAR
数据类型	INTER16
可访问性	RW
PDO 可映射性	YES
数据范围	0xEC78~0x1388
默认值	0x0000

索引	6071H
单位	0.1%
相关模式	PT CST

设置伺服目标转矩时，100%（读数为 1000）对应于 1 倍的电机额定转矩。

6） 最大轮廓速度 607Fh

该对象用于设置伺服的最大运行速度，在轮廓速度模式（PV）下，作为最大速度的限制；

在同步力矩模式（ST）、轮廓力矩模式（PT）时，为避免电机转速过高，造成机械冲击，使用该设定值作为最大速度的限制；

在周期同步位置模式下无效。

索引	607FH
名称	最大轮廓速度 Max profile velocity
数据结构	VAR
数据类型	UNSIGNED32
可访问性	RW
PDO 可映射性	YES
数据范围	0x00000000~0xFFFFFFFF
默认值	0x06400000
单位	指令单位 /s
相关模式	ALL

设置伺服最大运行速度（轮廓速度模式、同步力矩模式、轮廓力矩模式时有效）。用于限制伺服电机的最大运行速度。

例如，当伺服运行于力矩模式，当电机的实际的负荷转矩小于给定转矩时，运行转速会越来越快，当运行转速达到最大轮廓速度 Max profile velocity（对象 607FH）后，速度会被限制在该速度值。

7） 探针功能 60B8h

该对象用于主控器设定伺服的探针功能方式及启停。在运动控制系统中，通过伺服的探针功能，检测特定 DI 信号变化时的伺服位置信号，伺服以中断方式记录 DI 信号变动时的伺服位置，供上位机读取，这样可以提高系统控制的精度。

索引	60B8H
名称	探针功能 Touch probe function
数据结构	VAR
数据类型	UINTER16
可访问性	RW
可映射性 PDO	YES
数据范围	0x0000~0xFFFF
默认值	0x0000
单位	-
相关模式	ALL

设置探针 1 和探针 2 的功能。

Bit 位	描述
0	探针 1 使能： 0：探针 1 不使能 1：探针 1 使能
1	探针 1 触发模式 0：单次触发，只在触发信号第一次有效时触发 1：连续触发
2	探针 1 触发信号选择 0：DI8 输入信号 1：Z 信号
3	NA
4	探针 1 上升沿使能 0：上升沿不锁存 1：上升沿锁存
5	探针 1 下降沿使能 0：下降沿不锁存 1：下降沿锁存
6	NA
7	NA
8	探针 2 使能： 0：探针 2 不使能 1：探针 2 使能
9	探针 2 触发模式 0：单次触发，只在触发信号第一次有效时触发 1：连续触发
10	探针 2 触发信号选择 0：DI9 输入信号；1：Z 信号
11	NA
12	探针 2 上升沿使能 0：上升沿不锁存 1：上升沿锁存
13	探针 2 下降沿使能 0：下降沿不锁存 1：下降沿锁存
14	NA
15	NA

IS620N 驱动器只支持 Z 信号下降沿。

对于绝对值编码器，Z 信号是指每个单圈的 0 位置。

8） 伺服运行状态字 6041h(Status word)

该对象用于读取伺服驱动器的运行状态，伺服根据其当前运行状态，该对象为 PDO 配置表的必选项目之一。

索引	6041H
名称	status word
数据结构	VAR
数据类型	UNSIGNED16
数据范围	0~65535
默认值	-
可访问性	RO
PDO 可映射性	TPDO
相关模式	ALL

伺服从站以不同 bit 位状态反馈给主站：

bit	名称	描述
0	伺服无故障	
1	等待打开伺服使能	
2	伺服运行	
3	故障	
4	接通主回路电	
5	快速停机	
6	伺服准备好	
7	警告	
8	厂家自定义	预留，未定义
9	远程控制	0- 非远程控制模式，IS620N 系列产品仅支持远程控制模式；1：远程控制模式
10	目标到达	0：- 目标位置或速度未到达 1：- 目标位置或速度到达
11	软件内部位置超限	0- 位置指令或反馈未达到软件内部位置限制 1：- 位置指令达到软件内部位置限制，软件绝对位置限制生效后 (参考对象字典 607Dh 与 200A-02h)，伺服将以位置限制值为目标位置运行，到达限位值处停止
12~13		与各伺服模式相关
14	NA	预留
15	原点回零完成	0- 原点回零未进行或未完成 1- 已完成原点回零，参考点已找到

注意：

状态字的每一个 bit 位单独读取无意义，必须与其他位共同组成，反馈伺服当前状态

bit0~bit9 在各伺服模式下意义相同，控制字 6040h 按顺序发送命令后，伺服反馈一确定的状态。

bit12~bit13 与各伺服模式相关 (请查看不同模式下的控制指令)

bit10 bit11 bit15 在各伺服模式下意义相同，反馈伺服执行某伺服模式后的状态。

9） 位置反馈 6064h（Position actual value）

伺服多数情况下以位置模式运行，控制器必需实时监测伺服的当前位置，主控器通过该对象读取伺服当前的实际位置。该对象为 PDO 配置表的必选项目之一。

索引	6064H
名称	位置反馈 Position actual value
数据结构	VAR
数据类型	INTER32
可访问性	RO
PDO 可映射性	TPDO
数据范围	-
默认值	-
单位	指令单位
相关模式	ALL

反映实时用户绝对位置反馈。

位置反馈 6064h（指令单位）× 位置因子 6093h（齿轮比） = 实际位置 6063h（脉冲单位）

10）伺服告警码 603Fh（Error code）

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 IS620N 说明书。主控器通过查看伺服是最新故障记录码。该对象为 PDO 配置表的必选项目之一。

索引	603FH
名称	Error code
数据结构	VAR
数据类型	UINT16
数据范围	0~65535
默认值	-
可访问性	RO
PDO 可映射性	TPDO
相关模式	ALL

11）转矩反馈 6077h（Torque ActualValue）

显示伺服内部转矩反馈，量纲为伺服电机额定值的百分比（0.1%）。该对象为 PDO 配置表的常选项目之一。

索引	6077H
名称	转矩反馈 Torque ActualValue
数据结构	VAR
数据类型	INTER16
可访问性	RO
PDO 可映射性	TPDO
数据范围	-
默认值	-
单位	0.1%

索引	6077H
相关模式	ALL
描述 Comment	

显示伺服内部转矩反馈。

100%（即读数值为 1000）对应于 1 倍的电机额定转矩。

12）位置偏差反馈 60F4h（Following error actual value）

该对象用于指示当前位置与目标位置的偏差，量纲为指令单位。用于判断位置到达情况。

索引 INDEX	60F4H
名称 Name	位置偏差 Following error actual value
数据结构 Object Code	VAR
数据类型 Data Type	INTER32
可访问性 Access	RO
可映射性 PDO Mapping	TPDO
数据范围 Value Range	0x80000000~0x7FFFFFFF
默认值 Default Value	-
单位 Unit	指令单位
相关模式 Related Mode	PP HM CSP
描述 Comment	

显示位置偏差 (指令单位)。位置偏差 60F4 = 位置偏差 200B-36h

13）探针状态 60B9h（Touch probe status）

显示伺服探针触发端口的设置状态与触发状态，便于主控器读取判断探针位置记录数据有效性。

索引	60B9H
名称	探针状态 Touch probe status
数据结构	VAR
数据类型	UINTER16
可访问性	RO
PDO 可映射性	TPDO
数据范围	-
默认值	-
单位 t	-
相关模式	ALL

读取探针 1 和探针 2 的状态。

Bit 位	描述
0	探针 1 使能： 0：探针 1 不使能；1：探针 1 使能
1	探针 1 上升沿锁存执行 0：上升沿锁存未执行；1：上升沿锁存已执行
2	探针 1 下降沿锁存执行 0：下降沿锁存未执行；1：下降沿锁存已执行

Bit 位	描述
3	NA
4	NA
5	NA
6	探针 1 触发信号选择 0：DI8 输入信号；1—Z 信号
7	探针 1 触发信号监控 0—DI8 为低电平；1—DI8 为高电平
8	探针 2 使能： 0：探针 2 不使能；1：探针 2 使能
9	探针 2 上升沿锁存执行 0：上升沿锁存未执行；1：上升沿锁存已执行
10	探针 2 下降沿锁存执行 0：下降沿锁存未执行 1：下降沿锁存已执行
11	NA
12	NA
13	NA
14	探针 2 触发信号选择 0—DI9 输入信号；1—Z 信号
15	探针 2 触发信号监控 0—DI9 为低电平；1—DI9 为高电平

14）探针 1 位置反馈 60BAh、60BBh（Touch Probe Pos1 Value）

索引	60BAH	60BBH
名称	探针 1 上升沿位置反馈 Touch Probe Pos1 Pos Value	探针 1 下降沿位置反馈 Touch Probe Pos1 Neg Value
数据结构	VAR	VAR
数据类型	INTER32	INTER32
可访问性	RO	RO
PDO 可映射性	TPDO	TPDO
数据范围	-	-
默认值	-	-
单位	指令单位	指令单位
相关模式	ALL	ALL
描述 Comment	显示探针 1 信号的上升沿时刻，位置反馈 (指令单位)。	显示探针 1 信号的下降沿时刻，位置反馈 (指令单位)。

15）探针 2 位置反馈 60BCH、60BDH（Touch Probe Pos2 Value）

索引	60BCH	60BDH
名称	探针 2 上升沿位置反馈 Touch Probe Pos2 Pos Value	探针 2 下降沿位置反馈 Touch Probe Pos2Neg Value
数据结构	VAR	VAR
数据类型	INTER32	INTER32
可访问性	RO	RO
PDO 可映射性	TPDO	TPDO
数据范围	-	-
默认值	-	-
单位	指令单位	指令单位
相关模式	ALL	ALL
描述 Comment	显示探针 2 信号的上升沿时刻，位置反馈 (指令单位)。	显示探针 2 信号的下降沿时刻，位置反馈 (指令单位)。

16）查看伺服运行模式 6061H（Modes of operation display）

索引	6061H
名称	Modes of operation display
数据结构	VAR
数据类型	INTEGER8
可访问性	RO
PDO 可映射性	TPDO
数据范围	-
默认值	-
相关模式	ALL

通过对象 6061H 可查看伺服当前运行模式，伺服根据其当前运行模式，以对应的数值来指示：

值	描述
0x00	NA
0x01	PP(轮廓位置模式)
0x02	NA
0x03	PV(轮廓速度模式)
0x04	PT(轮廓转矩模式)
0x05	NA
0x06	HM(回零模式)
0x07	IP(插补模式)
0x08	CSP(周期同步位置模式)
0x09	CSV(周期同步速度模式)
0x0A	CST(周期同步转矩模式)

17）回零方式 6098H

对于采用相对定位方式的应用中，需要首先进行回零操作，便于伺服驱动器、运动控制器确定位置的参考零点。

索引	6098H
名称	回零方式 Homing method
数据结构	VAR
数据类型	INTER8
可访问性	RW
PDO 可映射性	YES
数据范围	0x00~0x23
默认值	0x00
单位	-
相关模式	HM
描述 Comment	

在通过 EtherCAT 通信让伺服回零操作前，需要设定伺服的回零方式，主站通过该数据对象设定伺服的回零方式，请参见附录中的归零方式详细说明。

注意：当选用 IS620N 的绝对位置编码器方式，可采用第 35 种回零方式，回零操作的结果是以当前位置作为零点，电机并不转动。

18）回零速度 6099H

在伺服驱动器在寻找原点信号时的运行速度可以通过该对象进行设定，该对象有 2 个子索引：

索引	6099H
名称	Homing speeds
数据结构	ARR
数据类型	UNSIGNED 32
可访问性	RW
可映射性	YES
数据范围	OD 数据范围
默认值	OD 默认值
相关模式	HM

设置回零模式下 2 个速度值：搜索减速点信号速度、搜索原点信号速度。

子索引	0	1	2
名称	回零速度的子索引个数	搜索减速点信号速度 speed during search for switch	搜索原点信号速度 speed during search for zero
数据类型	UNSIGNED8	UNSIGNED 32	INTER32
可访问性	RO	RW	RW
PDO 可映射性	NO	YES	YES
数据范围	2	0x00000000~ 0xFFFFFFFF	0x00000000~ 0xFFFFFFFF
默认值	2	0x001AAAAB	0x0002AAAB
单位		指令单位 /s	指令单位 /s

说明：

设置搜索减速点信号速度，此速度可以设置为较高数值，防止回零时间过长，发生回零超时故障 Er.601；

从站找到减速点后，将减速运行，减速过程中，从站屏蔽原点信号的变化，为避免在减速过程中即碰到原点信号，应合理设置减速点信号的开关位置，留出足够的减速距离，或增大回零加速度以缩短减速时间；

设置搜索原点信号速度，此速度应设置为较低速度，防止伺服高速停车时产生过冲，导致停止位置与设定机械原点有较大偏差。

19）回零加速度 609Ah

在伺服驱动器在寻找原点信号时的运行加速度可以通过该对象进行设定：

索引	609AH
名称	Homing acceleration
数据结构	VAR
数据类型	UNSIGNED32
可访问性	RW
PDO 可映射性	YES
数据范围	0x00000000~0xFFFFFFFF
默认值	0x682AAAAB
单位	指令单位 /s2
相关模式	HOME

设置原点回零模式下的加速度，原点回零启动后，设定值生效。

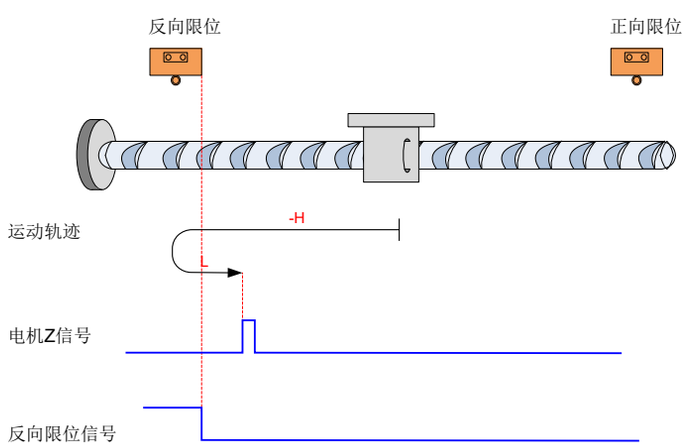
HM 模式下，暂停方式 605Dh=2 时，也将以 609Ah 设定减速停车。

该对象字典的意义为每秒位置指令 (指令单位) 增量，参数值设为 0 将被强制转换为 1

回零模式举例：

6098h=1

适用于如下机械结构的应用，在滑块行程两端有限位开关，没有零点开关信号，如下图：



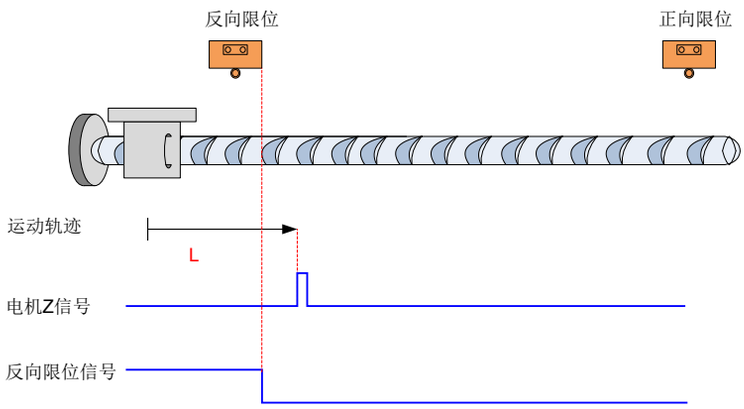
机械原点采用电机 Z 信号，减速点为反向超程开关。

◆ 回零启动时减速点信号无效

注：图中“H”代表高速 6099-1h，“L”代表低速 6099-2h

开始回零时 N-OT=0，以反向高速开始回零，遇到 N-OT 上升沿后，减速，反向，正向低速运行，遇到 N-OT 下降沿后的第一个 Z 停机；

◆ 回零启动时减速点信号有效



回零启动时 N-OT=1，直接正向低速开始回零，遇到 N-OT 下降沿后的第一个 Z 停机。

从上面对象的功能来看，把对象字典看作是一套可以由 EtherCAT 总线通信访问的伺服驱动器功能码定义，这样更好理解。

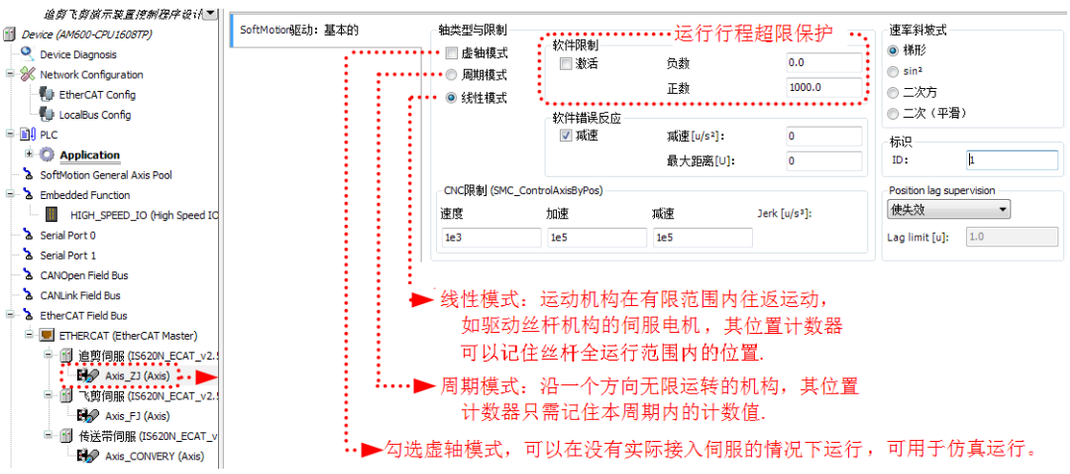
4.3.3 伺服轴电机参数的配置

运动控制的动作，最终是通过伺服电机的运转来实现的，若要伺服电机按照我们所希望的，控制器需要知道伺服电机参数、应用系统机械传动机构特性参数，以及用户希望的运转特性，才能发送合适的运行位置指令，这需编程时对控制器设置这些特性参数。

设置伺服电机参数的方法如下图说明，双击伺服驱动器下伺服电机，在右侧的窗口即可对电机相关的参数进行设置了。

① 在基本参数标签下，主要设置轴位置计数器模值。若伺服电机是往返运行的特点，如丝杆的往复运转，可以选择“线性模式”，（也称多圈模式、有限长模式），方便在伺服电机旋转多圈数的情况下，能进行绝对位置模式的定位；

② 若伺服电机是单方向无限运转，例如飞剪轧辊的运转，就可以“周期模式”，其位置计数器在每个运转周期内，从 0 开始计数，不会产生位置计数器溢出的情况；



📖 注意，上述设置规则，既适用于增量编码器伺服电机，也适合绝对编码器的伺服电机，上述设定值并没有送给伺服驱动器，而电机的当前位置，是由 AM600 根据电机返馈的位置信号，以及上述设定参数，自动进行位置的累计和量程计算，因此，若需对伺服位置有掉电保持特性，需要用户程序中对轴的当前位置备份到掉电保持的变量中，上电后再及时恢复给相关参数。

上图中的“软件限制”，是指通过 AM600 的软件，对伺服电机的行程超限进行保护，让 AM600 不会发出超限的定位指令，这在绝对位置编码器、绝对定位指令的 MC 指令的应用系统中非常有用。图中还有加减速特性曲线的选择项，在调试中可酌情选择，以使机械系统运转更为平稳。

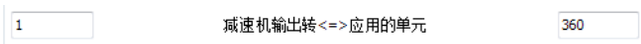
③ 因控制器总是以令伺服运转多少脉冲数来作为运行位置命令的，因此，控制器必需知道伺服电机的编码器每圈脉冲值，同时还需知道运转机构的减速比、丝杆导程、皮带轮周长等机械参数，在电机参数的“缩放 / 映射”标签下可输入这些参数，如下图：

上图中第 1 项，用于设定每圈脉冲数，可以十六进制输入；

上图中第 2 项，用于设定减速机的减速比，图中 5:1 表示伺服电机轴每转 5 圈，减速机输出轴旋转 1 圈；若没有使用减速机，则减速比为 1:1；

上图中第 3 项，用于设定输出轴每旋转 1 圈，工件的运行物理距离。例如，

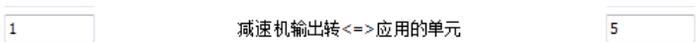
- 若使用的是飞剪轧辊，我们只关注其旋转角度，若这样填写：



指令 MC_MoveRelative(,distance:=1,) 执行的结果是机构旋转 1°；

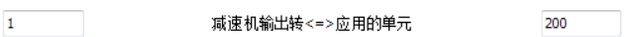
指令 MC_MoveRelative(,distance:=360,) 执行的结果是机构旋转 360°；

- 若使用的是导程 5mm 的丝杆，即丝杆每转 1 圈，丝杆上的滑块移动距离为 5mm，这样填写：



指令 MC_MoveRelative(,distance:=1,) 执行的结果是滑块机构行进 1mm；

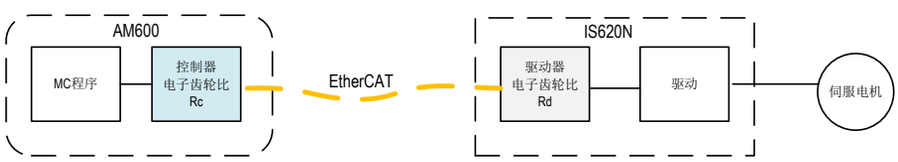
- 若使用的是直径为 63.7mm 的同步齿轮，则每转 1 圈，同步皮带的移动距离为 63.7mm×3.14=200mm，若这样填写：



指令 MC_MoveRelative(,distance:=100,) 执行的结果是皮带机构行进 100mm；

由此可知，通过对 1/2/3 项的准确设置，就可以实现应用系统物理单位与 MC 运行指令单位一致，使得用户程序的指令直观明了，方便变量设置，不易发生错误。

📖 注意，设置电机参数，是用于 AM600 在发送最终的（脉冲数）位置指令时，进行电子齿轮比的换算，并没有下载到伺服驱动器中；而伺服中功能码设置的电子齿轮比同样会对运行指令进行衰减处理，因此，如下图中，对伺服电机实际产生影响的是 Rc*Rd：



因此，若要保证用户程序在每个应用设备中的运行表现相同，必需通过 SDO 操作，将伺服的电子齿轮比功能码初始化到指定参数值，否则因为伺服功能码设置的不同，会导致运行响应的差异。

> 📖 注意，我们在 MC 编程时，通过设置电机参数，是应用系统物理单位与 MC 运行指令单位一致，往往是调试的第一个步骤，否则编程人员无法正确判断运行效果是否正确，还可能因运行位置超出应用系统的允许范围，极易导致设备损坏或人身伤害，需要特别注意。

4.3.4 EtherCAT 网络状态初始化与管理

1） EtherCAT 网络的初始化与状态判别

每次 AM600 控制器上电后，会自行启动，大约 10 秒内完成操作系统、用户程序的加载。若用户程序中没有使用到 EtherCAT 总线，控制器会根据用户程序对其他总线的初始化操作后，开始用户程序的执行；

若用户程序中使用了 EtherCAT 通信网络，AM600 作为 EtherCAT 主站，会对 EtherCAT 网络总线进行初始化操作，具体包括：

① 根据用户的 EtherCAT 配置，对主站进行配置，约需 3 秒钟；

② 发送网络初始命令，让所有从站 ESC 控制芯片开始初始化操作，依次读取 EtherCAT 网络的各从站的信息，并与用户程序中的 EtherCAT 网络配置进行比对，若有从站数量和顺序差异，就会报错；

③ 若网络配置正常，就会将 SDO、PDO 依次下发给各从站 ESC 芯片；

④ 让网络先进入 Pro-OP、Safe-OP、再到 OP 运行；

上述操作过程，是 AM600 自动完成的，并不需要用户干预，对每个从站的配置时间只需约 2 秒钟，从站数量越多，网络初始化时间相应增加；

用户程序判断应用系统的网络状态是否启动正常，最简单可靠的方法是检测每个伺服轴的 MC_Power.status 状态是否为 true，因为该状态可以表明网络正常、伺服正常，具备了开始运行的条件。

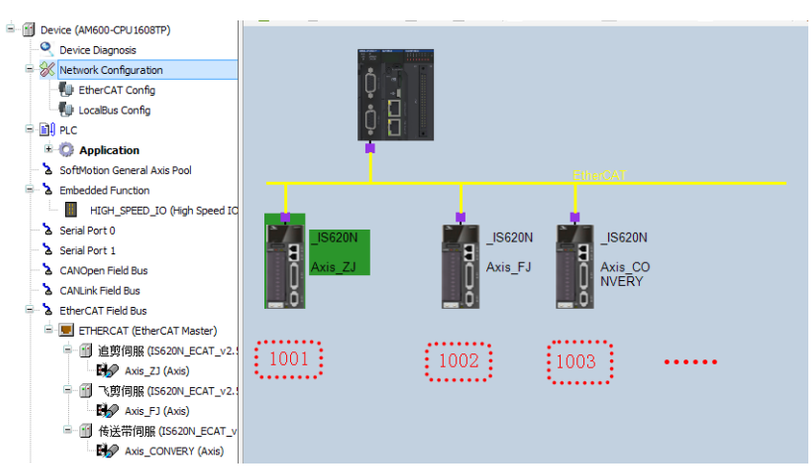
2） 通信掉站与通信恢复

我们知道，EtherCAT 从站能与主站正常通信的前提，是从站 ESC 经过主站配置后，先后进入网络的 Pro-OP、Safe-OP、再到 OP 运行状态，ESC 内部典型的配置内容包含 PDO 配置表，这些信息只有在主站给网络配置的时候，从站 ESC 才能得到，而主站网络一旦进入 OP 运行状态，就不能再发生配置信息，因此，当 EtherCAT 网络主站进入运行状态后，从站才上电，或从站在运行中掉电后再上电，都将无法进入网络运行状态。

目前 EtherCAT 从站掉电后要恢复网络运行，只有让主站重新启动开始运行，如拨动 RUN/STOP 开关，让主站重新开始运行，但这样会影响其他从站的运行。

3） 从站地址的编址与设定

在编写用户程序是，默认情况下，AM600 主控器是按照 EtherCAT 从站的网络电缆链接顺序，自动进行编址和寻址的，这种寻址方式的优点是，用户无需关心设备的命名与重名问题，只需按照用户程序中的总线网络配置，便于主控制器检查网络配置，发现硬件连接错误。AM600 对用户程序中添加的从站自动命名规则如下图:



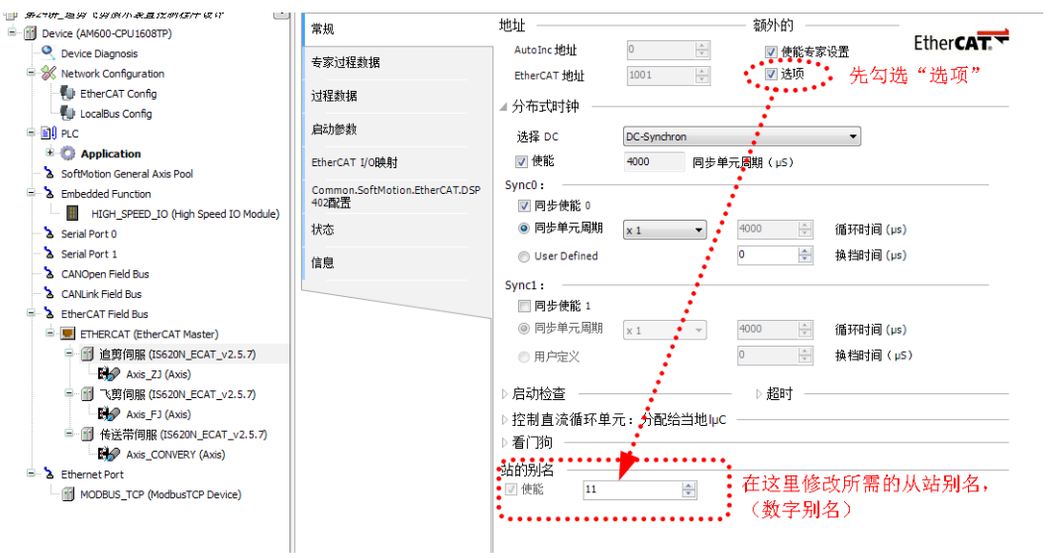
```
MC_P1：MC_Power；// 声明 MC_Power 的实例 MC_P1

MC_P1（Axis:= Axis1,
        Enable:= 1,
        bRegulatorOn:=1 ,
        bDriveStart:=1 ,
        Status=> ,）// 执行实例 MC_P1，使能伺服轴 Axis1
```

其中的从站序号从 1001 开始，根据添加的顺序，依次 +1，运行时，根据其中伺服的网络电缆的链接顺序，将与 AM600 直接相连的伺服命名为 1001，依次给伺服命名，将用户程序中，对某轴的控制功能，赋予对应序号的伺服。这种寻找方式的要点是，EtherCAT 网络电缆的链接顺序，必需按照用户程序中的网络配置顺序来进行。

但是，有些应用设备中，其中的部分轴的功能已有明确定义的情况下，并有一个固定的命名，要求 AM600 主控器的用户程序能按照这个固定的名称来寻址，这就要求用户编程时，将网络从站的寻址方式设为按“从站别名”寻址，另外在伺服中设置好对应的 “从站别名”。

① 将从站设为按“从站别名”寻址的方法如下图：



② 在伺服从站中设置设置从站别名。

例如对于 IS620N，我们可以将其“从站别名”功能码 H0C.05 功能码设为 11 即可。

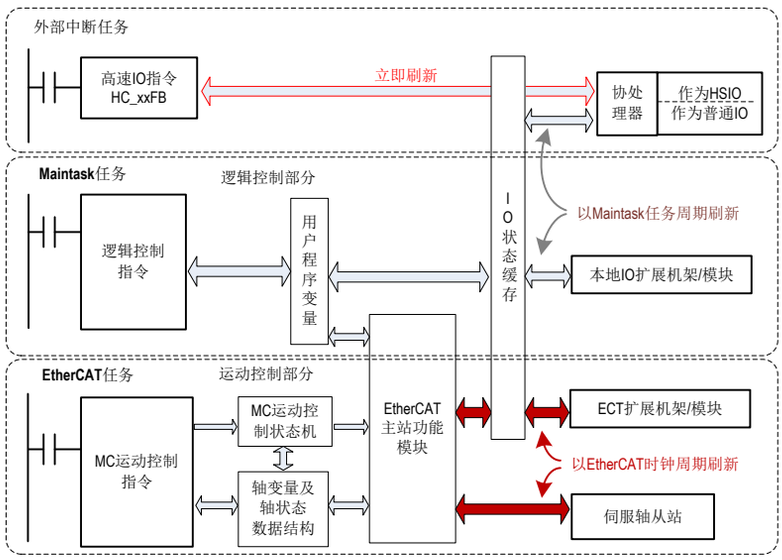
用户程序这样配置后，无论别名为“11”伺服的接入位置顺序如何，就都可以寻找到该伺服，将用户程序中对该伺服轴的操作功能特性赋予该轴。

📖 注意：

- 用户程序中，可以一个或多个轴按此方式进行命名，但是不要出现重名的情况；
- ECT 远程扩展模块目前没有设置从站的功能，不能按“从站别名”方式寻址；
- 若应用系统中还有部分伺服轴为自动命名，则系统会首先确定有“别名”的从站，其余从站仍按自动命名处理。

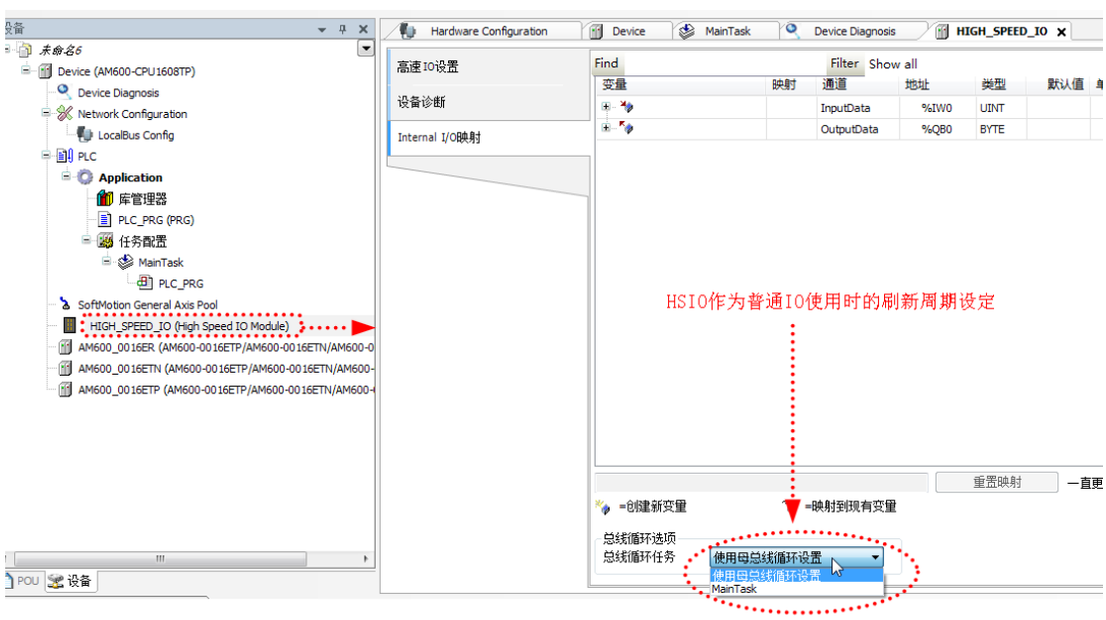
4.3.5 AM600 与伺服轴及 IO 端口控制数据刷新

AM600 的 IO 分为三种类型：主模块内置的高速 HSIO、主机架扩展模块 IO、扩展机架模块 IO。扩展机架与伺服轴一样，作为 EtherCAT 从站与 AM600 主模块连接。这些外设的访问刷新时间特点如下图：

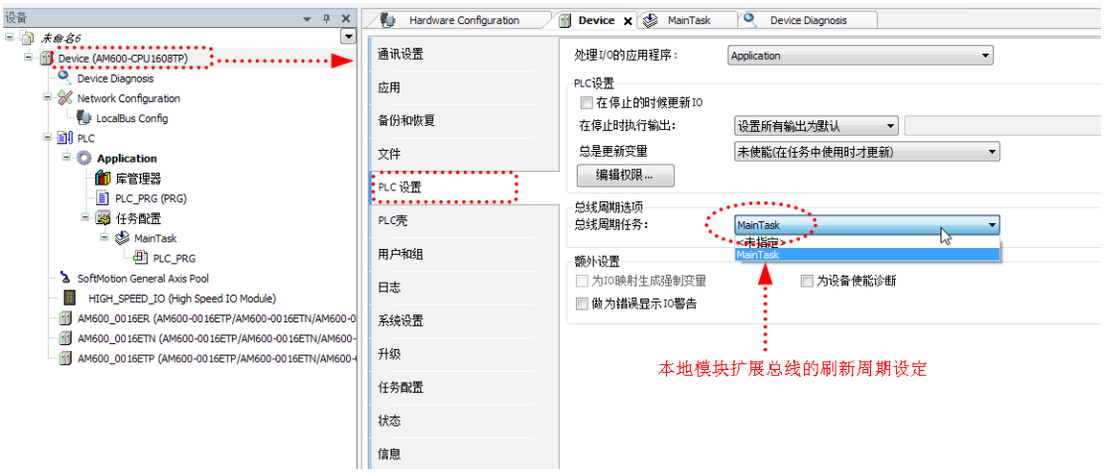


AM600 控制器内置有 16 路输入、8 路输出的 HSIO，内部配有一个协处理器，用于处理高速应用，比如中断信号、脉冲计数、脉冲特性测量等输入信号，还可以进行 PTO、PWM、脉冲定位等具有实时性要求控制输出，在执行汇川专有的高速处理功能块时，会立即触发协处理器的执行，及时更新输出。

当 HSIO 端口配置为普通端口时，其输出刷新端口的刷新周期可以设置为按普通任务周期进行，如下图设置：



AM600 主 CPU 模块与架上的 IO 扩展模块之间，通过所谓“本地总线”相连接，IO 状态的刷新由 CPU 模块控制，其刷新的时间间隔与普通任务周期相同，用户可在 InoProShop 中设定：



AM600 控制器基于 EtherCAT 总线扩展的机架远程 IO，其 IO 通信数据是与伺服轴的通信在同一通信帧中进行传送的，比如每 1ms、2ms、4ms 就传送一遍，但是逻辑控制 POU 一般是在普通任务中执行的，IO 状态的实际更新周期为该任务周期，比如每 20ms 更新一次。

4.4 MC 运动控制数据的传送时序

AM600 按照用户设定的 EtherCAT 周期，定时进入 EtherCAT 中断，完整执行一次 EtherCAT 任务，首先执行的是主控器与各 EtherCAT 从站的通信操作，然后执行该任务下用户配置的所有 POU，执行的顺序按任务配置表中的 POU 顺序进行。

主控器与各 EtherCAT 从站的通信操作的内容：

① 启动 EtherCAT 总线发送操作，将上一个 EtherCAT 周期系统准备好的 TPDO 发送缓冲区数据，发送给各从站的数据依次发送，通信帧按照 RPDO 配置，预留了从站应答数据需要占用的若干字节空隙，以便将各从站的数据取回；

TPDO 发送缓冲区中的数据，按照从站接续的顺序；发送的数据包含普通 I/O 的数据和 MC 运动控制轴的控制数据；

当从站数量比较多，超过一个通信帧允许的数据长度，会使用若干通信帧来进行；

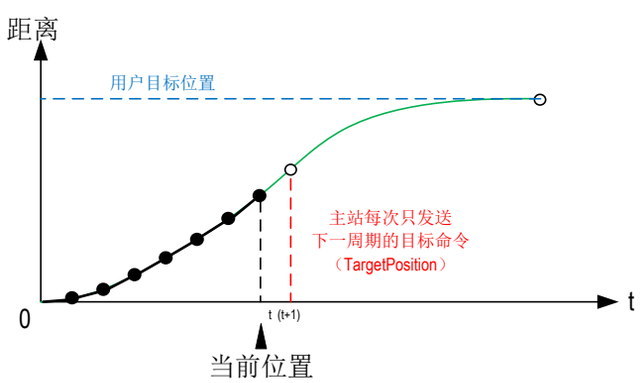
若用户程序执行了 SDO 读写操作，最后会发送 SDO 发送请求；

② 主控器对通信返回帧进行解析，取出各从站的应答数据，对于 MC 从站轴的应答数据进行分析，更新轴状态和位置、速度、力矩等数据结构，确定和更新 MC 功能块的执行状态指示，供用户程序读取。每次进入 EtherCAT 中断后，用户程序读取的轴参数就是在这个环节系统已经自动处理和更新后的数据；

4.5 AM600 执行 MC 功能块的处理机制：

4.5.1 AM600 对伺服的运动命令，采取周期同步位置的控制模式

所谓“周期同步位置模式”，就是 AM600 控制器，根据用户希望从站轴到达的位置、允许的运行速度、加速度、EtherCAT 总线周期等条件，在每次 EtherCAT 任务执行中，由相关的 MC 功能块计算下一个周期点要求到达的位置（TargetPosition），发送给伺服驱动器，而伺服将根据这个距离 / 时间命令，运动到达下一个到达目标点。在这种运行模式，AM600 全程负责伺服的每一个时间点运行位置与速度的规划计算，而伺服只知道下一个 EtherCAT 时刻需要到达的目标点和运行速度。



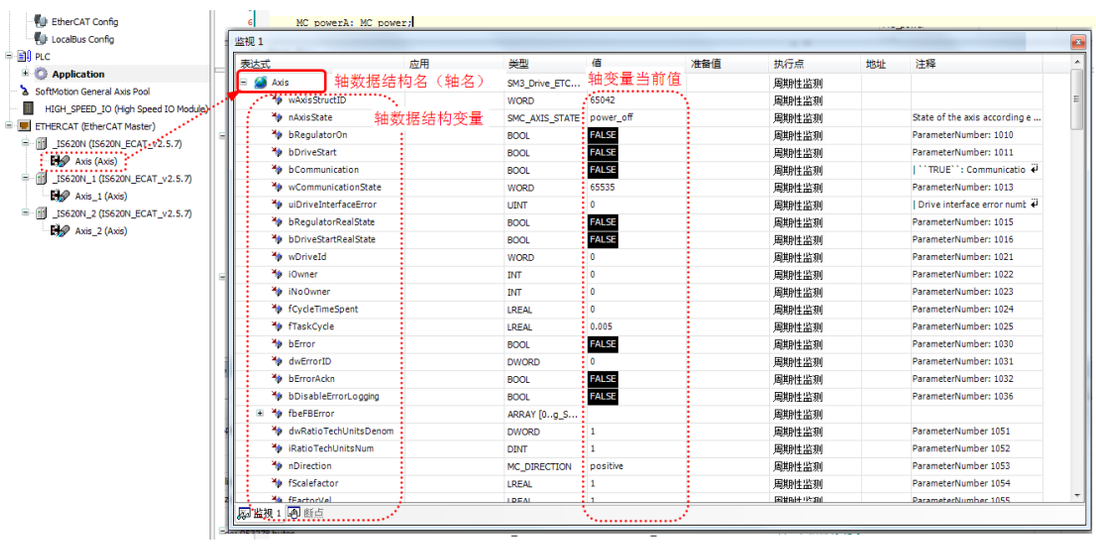
值得注意的是，伺服在以 “位置模式”、“速度模式”运行时，AM600 都是以“周期同步位置模式”来命令伺服运行的。

反过来讲，一个正在运转的伺服轴，必需有一个被有效触发的 MC 功能块持续监控该伺服轴的运行，如果因用户程序的逻辑跳转，导致没有对应该轴的 MC 块在运行，这种状态持续若干个 EtherCAT 周期后，伺服将停机，而控制器也会出错报警。

4.5.2 伺服轴的数据结构

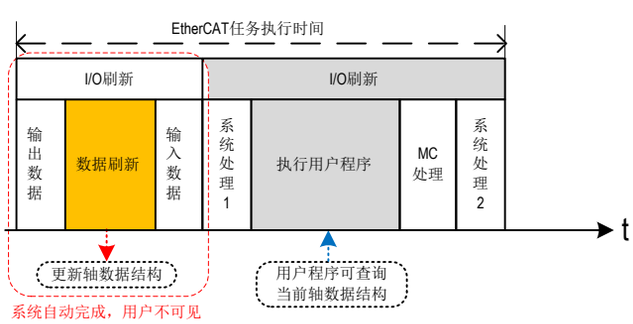
在 AM600 中，将伺服从站作为特殊的“轴”来进行管理的，轴是一个重要的对象。AM600 的系统软件中，对用户配置的每个伺服轴，系统都会同时自动声明一个对应于该轴的数据结构，并在每个 EtherCAT 中断执行时，自动实时更新维护；用户程序可以通过访问该数据结构，来了解伺服轴当前的命令值、运行状态、运行位置、速度、加速度、力矩等等参数信息，共有 100 余个数据结构变量，对轴的状态信息有一个全方位的描述。

下图为用户程序中一个轴 Axis 的监控窗口举例，其信息就是来自该轴的数据结构。



关于轴数据结构，用户需要理解和注意如下特性：

- 用户在进行应用网络的伺服轴配置时，系统同时自动声明了该数据结构，数据结构名即该轴的名字相同，而数据结构中的变量名及数据类型则是由系统定义的。
- 如上图的用户工程中，有 3 个伺服轴（Axis、Axis_1、Axis_2），每个轴都有其对应的数据结构。
- 用户程序中若用到虚轴，包括编码器轴，系统同样会为之声明和维护一个对应的轴数据结构，只是部分结构变量是否变化。
- 轴数据结构变量为全局变量类型，即在用户工程的所有 POU 中，都可以直接访问。
- 只要控制器运算能力满足应用系统的要求，系统允许的轴数没有明确的限制，有对应数量的轴数据结构。
- 一旦控制器开始运行，在每个 EtherCAT 任务运行阶段，控制器取回从站通信应答数据后，即自动将伺服的反馈值更新到该数据结构中，在执行用户 POU 时，即可访问该数据结构的变量。



- 轴数据变量的指定方法是：“数据结构名 . 结构变量名”，如下所示的数据结构中，我们经常用到如下的参数：

  Axis.nAxisState：轴当前运行状态，伺服反馈给控制器的状态参数；

  Axis.fSetPostion：轴设定位置，控制器发送给伺服轴的参数；

  Axis.fActPostion：轴当前实际位置，伺服反馈给控制器的状态参数，量纲与用户程序设定的命令单位相同；

Axis.fActVelocity: 轴当前实际速度，伺服反馈给控制器的状态参数量纲与用户程序设定的命令单位相同；

在用户程序中，可以将这些变量作为运动控制计算和判断的依据；轴结构中部分变量为控制器发送给伺服轴的命令数据，在用户程序中，也可以通过直接赋值给这些变量的方式，来对伺服轴进行控制。例如如下 ST 语句:

Axis.fSetPostion:=500;// 该参数的单位与指令单位相同

4.5.3 伺服轴的状态及转移规则

AM600 符合 PLCopen 规范，在运动控制系统中，将轴的运行状态分成若干个逻辑状态，而每个逻辑状态直接的转移，需要特定的条件、或指定的 MC 运控指令。这样划分处理的好处是便于轴按运动模式分类控制，轴在一个时候只能处于一种逻辑状态，而逻辑状态的转移需要按规则进行，不会因不同 MC 的误触发而带来运行的混乱。

轴数据结构变量 (Axis.nAxisState) 来指示轴的当前运行状态，该变量 Axis.nAxisState 为枚举型变量，共有如下 8 种可能状态：

0：Power_off（Disabled）：轴未上电使能，或未使能，需执行 MC_Power 指令

1：Errorstop;---------- 先执行 MC_Reset/MC_Power 指令

2：Stopping;---------- 等待停机操作完成

3：Standstill;---------- 轴已停止运行，脱离同步

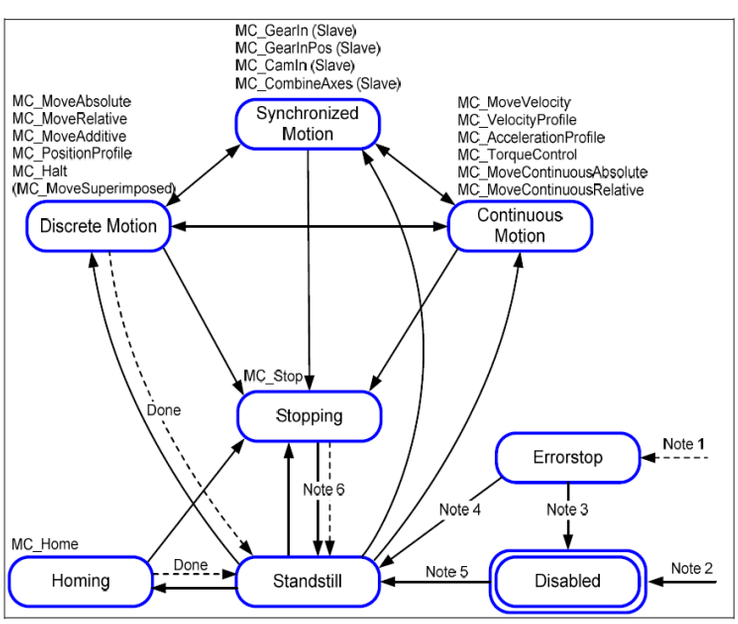
4：Discrete_Motion;--------- 轴处于离散运行状态

5：Continuous_Motion;----- 轴处于连续运行中

6：Synchronized_Motion;--- 轴处于同步运行中

7：Homing;---------- 轴处于回零运行中，等待归零操作执行完成

轴状态转移图如下，从一个状态转移到另一个状态，需要运行对应的条件，如运行 MC 指令，或外部出现了故障，用户无法对其状态进行强制，编程时一定要按照逻辑要求，运行相关的指令：



Note 1:	From any state. An error in the axis occurred.
Note 2:	From any state. MC_Power.Enable = FALSE and there is no error in the axis.
Note 3:	MC_Reset AND MC_Power.Status = FALSE
Note 4:	MC_Reset AND MC_Power.Status = TRUE AND MC_Power.Enable = TRUE
Note 5:	MC_Power.Enable = TRUE AND MC_Power.Status = TRUE
Note 6:	MC_Stop.Done = TRUE AND MC_Stop.Execute = FALSE

图中的 MC 功能块可以使轴状态转移到指定的状态，由图中可以看到：

- 在轴停止状态（Standstill，即 Axis.nAxisState=3）可以转移到各种运行状态；

- 可以由多种状态转移到停止状态（Standstill，即 Axis.nAxisState=3），，
- 若伺服轴出现告警（Errorstop，即 Axis.nAxisState=1），就必需先运行 MC_Reset 指令、MC_Power 指令使轴进入 Standstill 状态，才能让轴再次运行；
- 若不按上述转移图方式使用 MC 指令命令轴运动，就不会使轴响应，反而得到 MC 功能块的错误告警信息；

用户程序中，有时需要根据轴的状态，启动后续的控制逻辑，此时依据 Axis.nAxisState 的判断，相比于对 MC 功能块的 done 信号判断，更为准确可靠。

熟悉上述轴状态图的转移条件，并在编程时，注意 MC 指令的使用逻辑和顺序，才能编写出稳定可靠的应用程序。

4.5.4 MC 功能块的执行逻辑：

与伺服从站有关的轴控制命令，多以 MC 功能块（也称指令）的形式出现，因为 MC 功能块需要短间隔的持续执行，需要及时监视伺服的运行响应，所以与轴运动有关的 MC 功能块只能在 EtherCAT 任务中被调用执行。系统内部会进行如下处理：

① 在执行 MC 功能块时，被有效触发 MC 块才执行，对于同一个 MC 功能块的多个实例（针对同一个轴对象），后触发者优先的原则；

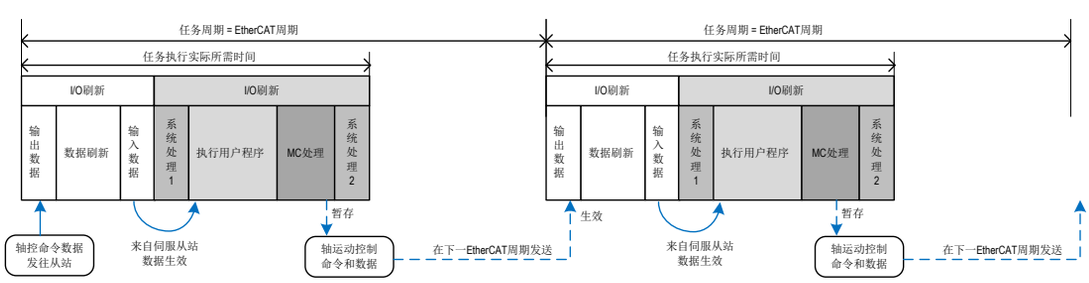
② 对于伺服轴的 MC 控制命令，按照轴状态转移规范检查操作的合法性，再进行处理，包括轴状态转移、轴的目标参数更新，最后制备轴控制命令数据；

③ EtherCAT 总线控制部分的系统软件，会根据用户设定的每个伺服从站轴的 TPDO 配置表、对象字典，将轴控制命令参数参数，制备成 PDO 发送缓存区数据；

④ EtherCAT 总线控制部分的系统软件，会根据用户设定的每个伺服从站轴的 RPDO 配置表、对象字典，将主站需要读取的该轴状态参数，在 EtherCAT 帧接收区段预留了从站应答数据需要占用的若干字节空隙，最后“组装”到 EtherCAT 帧发送缓存区，等待下一个 EtherCAT 周期开始后，发送给从站；

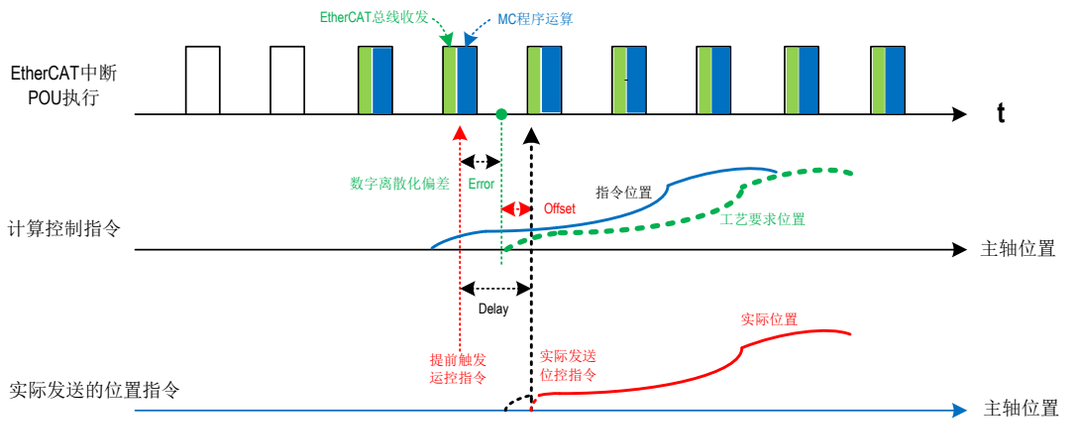
⑤ 关于 EtherCAT 远程 IO 的操作结果，按照从站机架的连接顺序，存储于缓存区，与伺服从站一起被发送。只不过在普通任务周期（任务优先级为 15 或更低，周期如 20ms）完成后，才更新对于发送缓存区数据状态而已；

⑥ 主控制器执行用户程序、通过 EtherCAT 收发数据发送的时序如下图：



⑦ 用户运控程序中，若伺服系统处于运行状态，必需要有一个被触发执行的 MC 功能块在监管该伺服轴，避免因程序逻辑跳转导致一个正在运转的伺服轴，呈没有 MC 监控的状态，用 MC_Stop 使之停止，也是一种监控。

由此可见，EtherCAT 任务中涉及轴控制的操作命令，并非是在当前 POU 执行周期内发送出去的，而是有一个 EtherCAT 周期的延迟，这在一些需要精确的位置与长度控制的应用中，如多轴同步控制的 MC_CamIn 的触发等，需要考虑这个延迟带来的误差，如下图：



🕮 注意：

针对上述原因产生的延迟误差，编程时注意如下处理：

提前 1 个 EtherCAT 周期触发运控指令；

控制工艺要求的运控启动不一定正好处于 EtherCAT 周期的起始时刻，而是处于周期中间的某个时刻，这个离散化偏差的消除，编程时应该考虑在内，利用 MC 运动控制功能块提供的 Offset 参数的进行补偿；

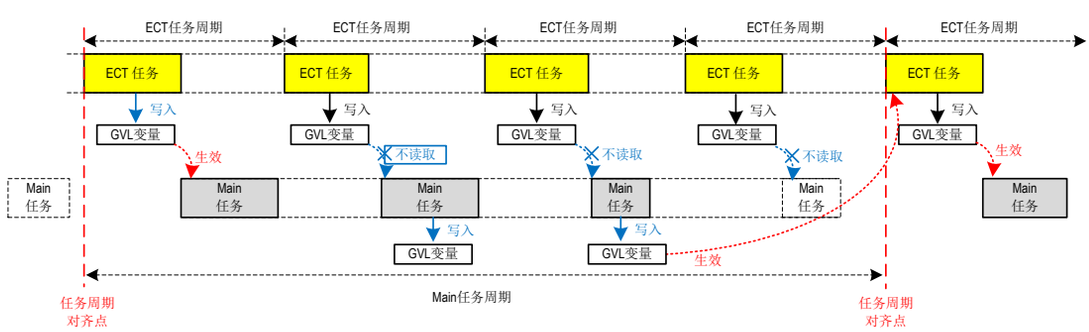
⑧ 消除离散化偏差，基于当前对象的运行位置、速度、加速度等参数来估算，这种通信机制导致的误差。适当减小 EtherCAT 总线周期的设定，有利于减少不可控误差。

4.5.5 不同优先级任务 POU 之间的数据交互

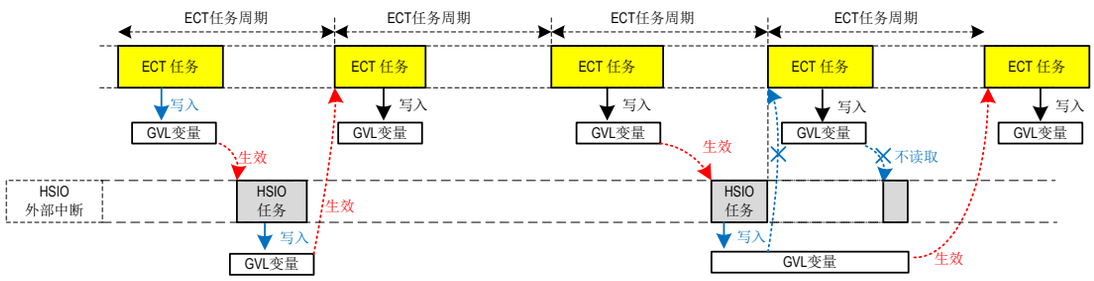
多个 POU 之间若要有变量的访问交互，就需要使用全局变量，即在 GVL 全局变量列表中声明；但若 POU 处于不同优先级任务中，数据并非实时交互，数据的更新结果与任务优先级及任务周期的设置、变量访问类型等有关。我们需要注意如下机制：

用户程序执行时，对于不同优先级、不同循环周期的任务，AM600 的系统内部采用了起始时间对齐的规则，即有一个共有的任务周期起始时间计算点，若两个任务的周期呈整数倍的关系，那么他们将有一个定期重合的时间点（对齐点），这个时间点常作为 GVL 数据交互点；

- 只有在任务执行完毕后，其 POU 对于变量的修改才会写入到 GVL 缓存区；低优先级任务对 GVL 参数的修改，只有在其任务周期结束时，才会生效；
- 高优先级的 POU，对 GVL 的改写操作，会立即生效；
- 低优先级任务在对齐点开始的第一个任务执行前，会从 GVL 缓存区复制 GVL 值一次，作为本任务 POU 执行期间使用的依据，在此任务执行过程中不会再读取 GVL 缓存区变量；



- 伺服轴数据结构为系统自动定义的全局变量，每次执行 ECT 任务时，系统会自动刷新该数据结构，若 Main 任务 POU 中要读取该数据结构的变量，其读数也是每个“任务周期对齐点”之后的第一个 ECT 任务所更新后的数据；同样的原理，若 Main 任务 POU 中要修改该数据结构的变量，是在下一个“任务周期对齐点”之后的第一个 ECT 任务中发送给伺服轴，会有约一个 Main 任务周期的滞后；
- 事件触发执行的任务，如 HSIO 引发的中断任务 POU 执行，会使用 GVL 中的最新变量值，如下图所示为 EtherCAT 任务与 HSIO 中断任务之间的 GVL 更新值可以及时得到交互，当优先级较低的 HSIO 任务执行被 EtherCAT 任务打断，其修改后的 GVL 值只有在执行完毕后，才能被下一次 EtherCAT 任务执行时才能生效，如下图：



📖 注意：

由此可见，在设定用户程序的 EtherCAT 任务与普通主循环任务的周期，应保持两者呈整数倍关系，（如 EtherCAT 任务为 2ms 或 4ms，主循环任务为 20ms），这样可以避免 GVL 参数交互时出现异常的情况；

不同优先级任务中，若对同一个 GVL 变量有修改操作，可能有彼此覆盖的情况，编程时，建议一个全局变量只让一个 POU 进行改写或置位操作，其他 POU 只是读取和引用或复位操作，以避免预料之外的错误。

Memo NO. _____

Date / /

第5章 AM600用户程序的应用编程

五、AM600 用户程序的应用编程

5.1 单轴 MC 定位的运动控制编程

5.1.1 AM600 运控应用的编程要点提醒

AM600 控制器与伺服轴（如 IS620N）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。

5.1.2 单轴控制常用的 MC 功能块

MC 功能块（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，例如：

MC_Power1：MC_Power；// 声明实例 MC_Power1

MC_Power1（Axis=Axis1，）

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

控制操作	需要使用的 MC 指令	说明
速度控制	MC_MoveVelocity	命令伺服以指定的速度运行
力矩控制	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
告警复位	MC_Reset	当伺服出现告警停机后，运行该指令进行复位，
改变伺服运行模式	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

5.1.3 MC 指令与 PDO/SDO 配置

AM600 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_Reset MC_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）	
MC_MoveTorque	Target torque（目标力矩）	Torque actual value（当前力矩）
MC_ControlMode	Modes of operation（运行模式）	16#6060=8: 周期同步位置 CSP 16#6060=9: 速度模式 16#6060=10: 力矩模式

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，一般将伺服设置为该运行模式。以 IS620N 为例，一般在 SDO 中初始化如下项目：

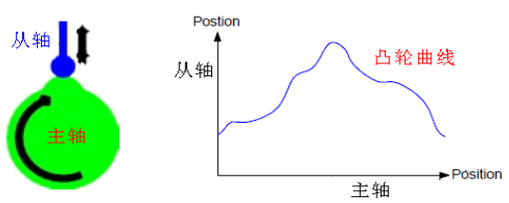
对伺服的初始化操作	需要使用 SDO 对象	说明
设为周期同步位置模式	Modes of operation（16#6060）	设为 8
设电子齿轮比	16#6091-1: 16#6091-2:	建议设为 1:1 （不推荐采用功能码设定）

对伺服的初始化操作	需要使用 SDO 对象	说明
设马达编码器类型	16#0201（IS620N 功能码）	0: 增量；1: 绝对有限长； 2: 绝对无限长
设允许最大转速	Max profile velocity（16#607F）	适用于速度模式、力矩模式
设允许最大力矩	Max torque（16#6072）	适用于速度模式、力矩模式
设回零模式	Homing method（16#6098）	
设回零速度	Homing speeds（16#6099）	
探针功能	Touchprobe function（16#60B8）	

5.2 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：



AM600 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

AM600 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。

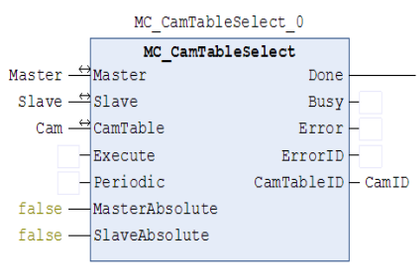
电子凸轮控制的基本指令功能块：

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

5.2.1 凸轮运行的主要功能块说明

1） 凸轮表选择 MC_CamTableSelect 功能块

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。




```
MC_CamTableSelect(
        Master:= , // 凸轮主轴
        Slave:= , // 凸轮从轴
        CamTable:= ,// 凸轮表
        Execute:= , // 指令触发变量，上升沿有效
        Periodic:= , // 设定凸轮周期性
        MasterAbsolute:= ,// 主轴位置模式
        SlaveAbsolute:= , // 从轴位置模式
        Done=> , //
        Busy=> ,
        Error=> ,
        ErrorID=> ,
        CamTableID=> );
```

```
MC_CamIn(
        Master:= , // 凸轮主轴
        Slave:= , // 凸轮从轴
        Execute:= ,// 执行触发，上升沿有效
        MasterOffset:= ,// 主轴位置偏置
        SlaveOffset:= , // 从轴位置偏置
        MasterScaling:= ,// 主轴放大比率
        SlaveScaling:= , // 从轴放大比率
        StartMode:= , // 从轴触发位置模式
        CamTableID:= , // 凸轮表指针
        VelocityDiff:= , // 速度偏差
        Acceleration:= , // 加速度
        Deceleration:= , // 减速度
        Jerk:= , // 加加速度
        TappetHysteresis:= , // 挺杆回差
        InSync=> , // 同步指示
        Busy=> , // 正在运行
        CommandAborted=> , // 指令被中断
        Error=> , // 出错
        ErrorID=> ,// 错误号
        EndOfProfile=> , // 已执行到凸轮尾
        Tappets=> ,// 挺杆有效
);
```

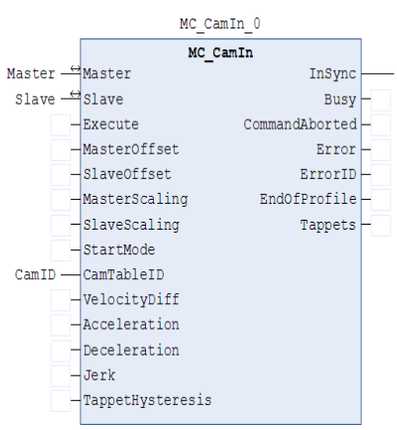
该功能块的输出输出变量及其说明如下：

MC_CamTableSelect

	类型	初始值	描述	
VAR_IN_OUT				
Master	AXIS_REF		映射到主轴	主轴名
Slave	AXIS_REF		映射到从轴	从站名
CamTable	MC_CAM_REF		映射为cam表格描述	待选择的CAM曲线表名称
VAR_INPUT				
Execute	BOOL	FALSE	此变量的一个上升沿将会启动功能块的处理	
Periodic	BOOL	TRUE	TRUE = 周期， FALSE = 非周期	
MasterAbsolute	BOOL	TRUE	TRUE = 绝对坐标， FALSE = 相对坐标	
SlaveAbsolute	BOOL	TRUE	TRUE = 绝对坐标， FALSE = 相对坐标	
VAR_OUTPUT				
Done	BOOL	FALSE	TRUE，如果cam表格选择结束	
Busy	BOOL	FALSE	TRUE，如果功能块的处理没有完成	
Error	BOOL	FALSE	功能块内部发生错误信号	
ErrorID	SMC_ERROR	0	错误ID	
CamTableID	MC_CAM_ID		cam表格的定义	选择并解析后的CAM表ID

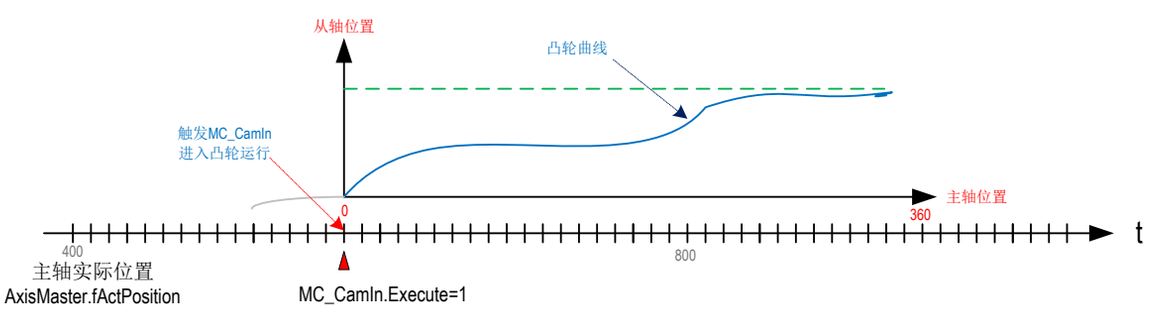
2） 凸轮运行的 MC_CamIn 功能块

当运行了 MC_CamTableSelect 功能块后，再该功能块可使得 Slave 从轴进行凸轮运行状态（Synchronized_Motion，即 Axis.nAxisState=6），并在每次进入 EtherCAT 任务后，系统均执行一次该功能块，根据主轴当前位置和凸轮表，计算从轴的下一个目标点。若事先没有运行 MC_CamTableSelect 功能块，触发该功能块就会报错。



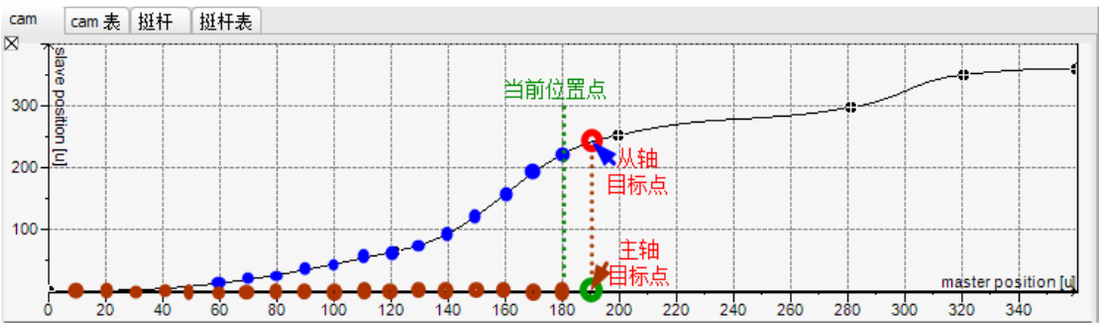

```
MC_CamOut(
    Slave:= , // 凸轮从轴
    Execute:= ,// 触发变量，上升沿有效
    Done=> , // 执行完成
    Busy=> , // 正在执行中
    Error=> , // 出错
    ErrorID=> // 错误号
);
```

本指令的功能是，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。

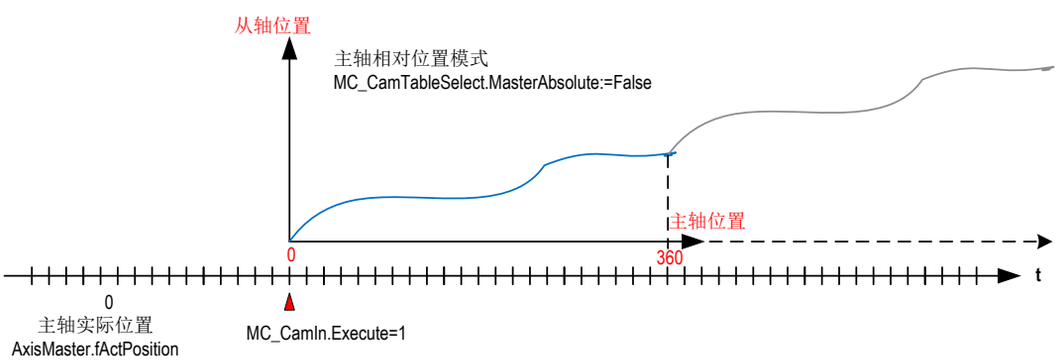


一旦 MC_CamIn 被触发，从轴按照凸轮表的位置对应关系，跟随主轴的位置运动，注意是位置对应，而不是速度对应。

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：



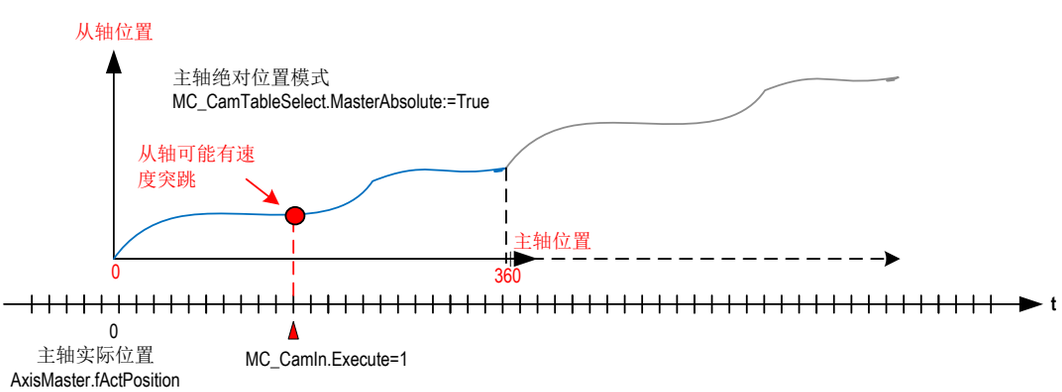
5.2.2 当主轴和从轴都为相对位置模式时的运行特点



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。

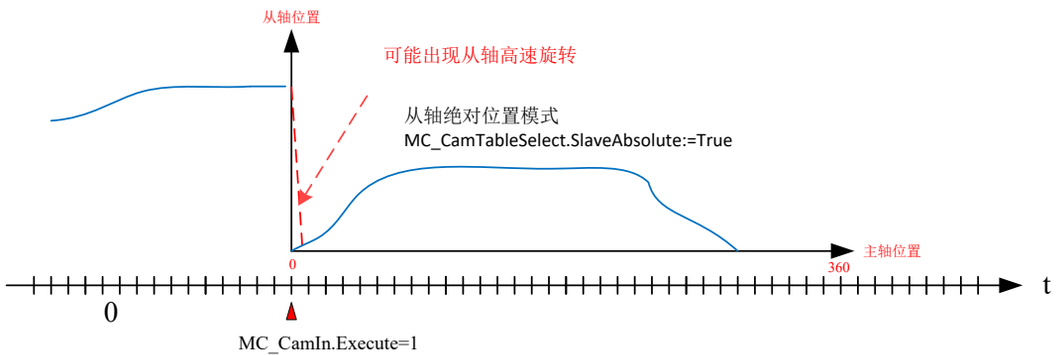
5.2.3 当主轴为绝对位置模式，从轴为相对位置模式



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。注意：

5.2.4 当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：

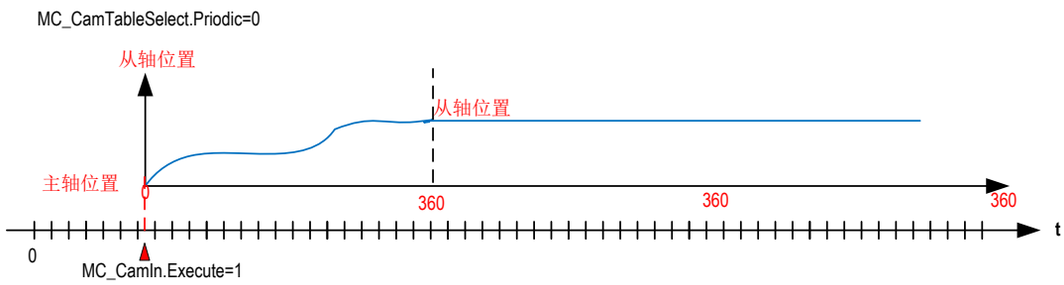
  MC_CamTableSelect.SlaveAbsolute:=False; 或设 MC_CamIn.StartMode:=1;（相对模式）

  注意：

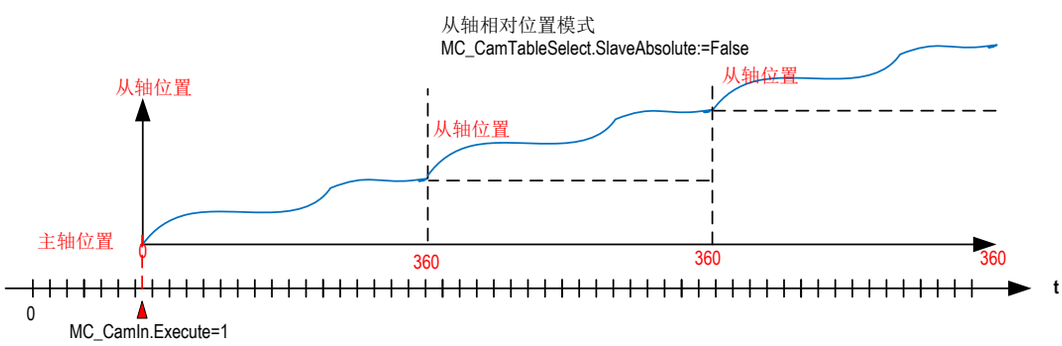
  当从轴设为“有限长”的绝对模式时，在作回零调整时，若可以左转回零，也可以右转回零，控制器会选择一个比较近的方向回零。设计凸轮表的范围时，尤其要注意不要让凸轮表的范围超过了实际需要运行的范围，否则就可以出现伺服从轴的瞬间高速回转调整，形成机械冲击。

5.3 凸轮表的周期模式特点

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。

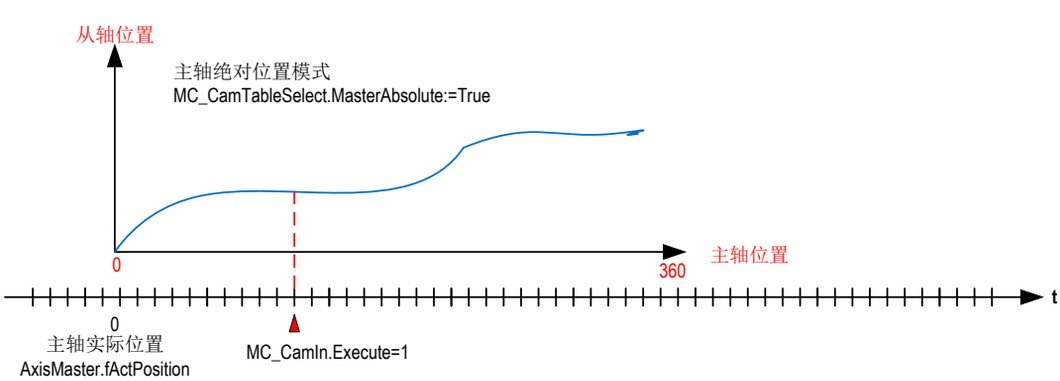


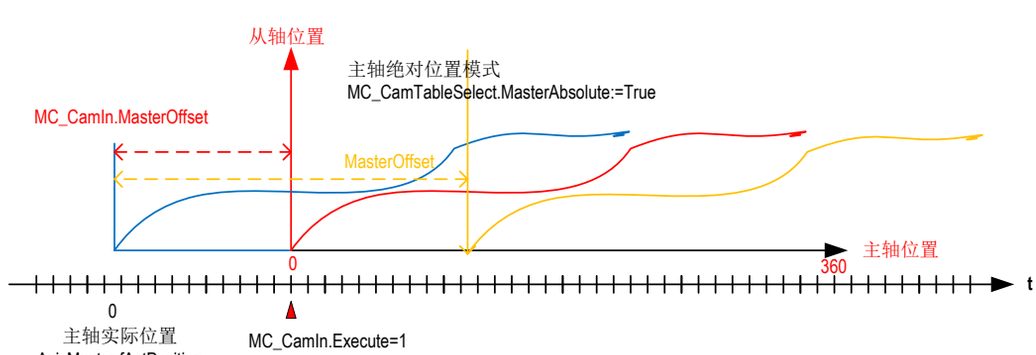
当凸轮表选择为周期模式（Periodic:=1），在运行完一个凸轮表周期后，从轴又开始下一凸轮周期的运行，直到有用户程序命令其退出凸轮运行状态，如下图：



上图为周期性凸轮运行的效果，一旦完成凸轮表中主轴位置范围，即自动开始下一凸轮周期的运动。

5.3.1 CamIn 运行的 Offset 功能



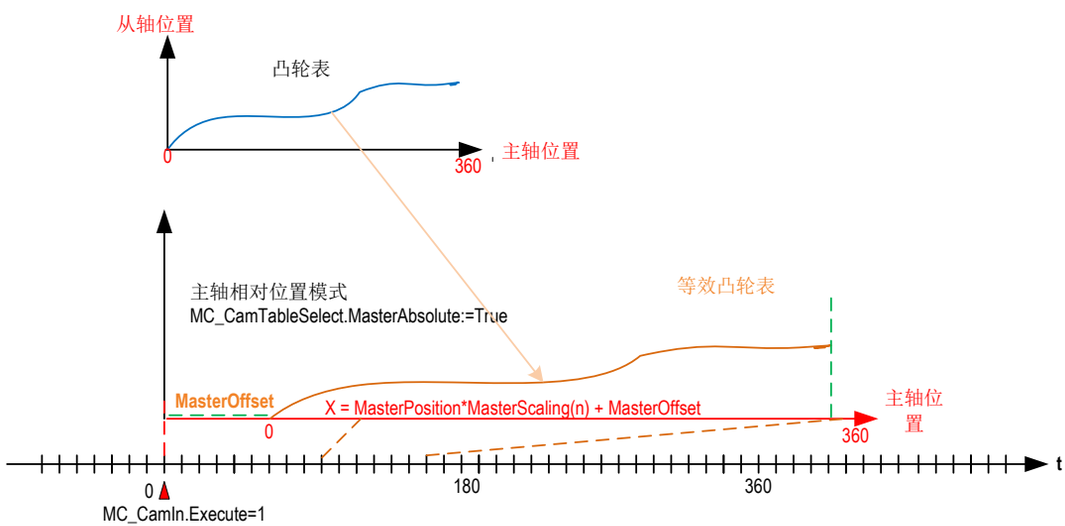


通过给凸轮主轴设置合适的偏移 Offset 值，可以修改凸轮从轴的起始点。根据主轴当前的实际位置，来计算偏移值，可以使其从凸轮表的 0 点开始：

MasterOffset := 0 – AxisMaster.fActPosition

5.3.2 凸轮运行中的主轴 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。

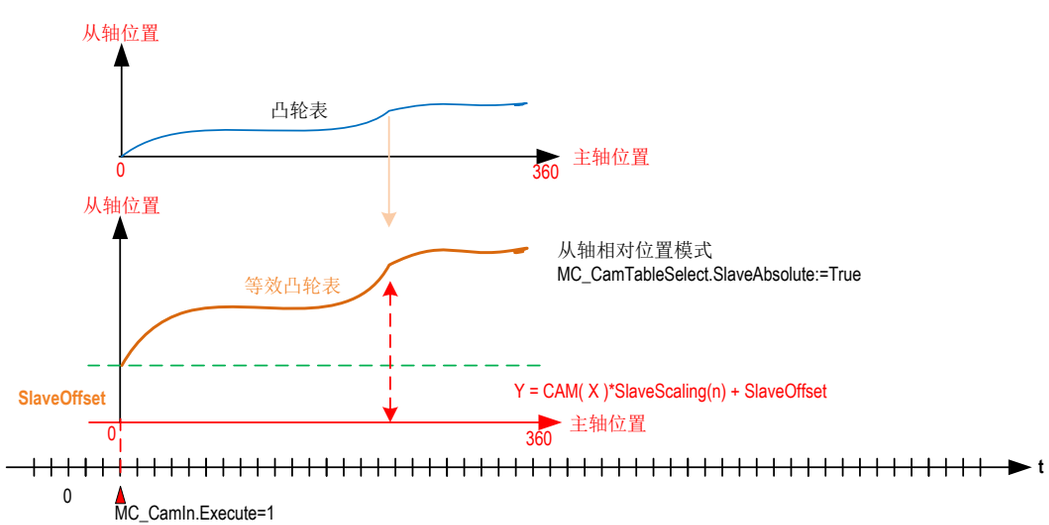
若考虑了主轴的偏移设定值，在凸轮表中主轴（X）的计算位置将是：

X = MasterPosition*MasterScaling(n) + MasterOffset

该参数可用于设备加工工件的尺寸微调。

5.3.3 凸轮运行中的从轴 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。

5.3.4 凸轮运行中的 Offset、Scale 使用特性与注意事项

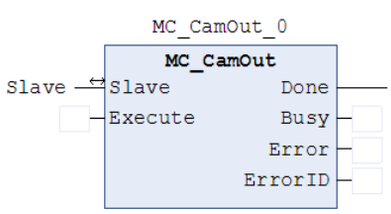
主轴位置模式、从站位置模式，除应用系统特别要求外，建议尽量采用相对模式，这样编程简单，出现机械系统冲击的可能性比较小；

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；

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。

5.3.5 凸轮运行状态的退出 MC_CamOut 功能块

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。



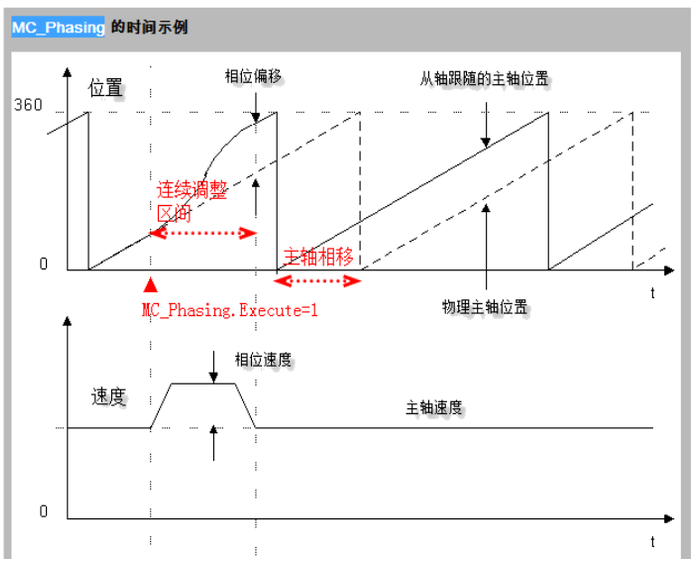

```
VAR i: INT;
CAM: MC_CAM_REF := (
    byType:=2, (* non-equidistant *)
    byVarType:=2, (* UINT *)
    nElements:=128,
    xStart:=0,
    xEnd:=360);
Table: SMC_CAMTable_UINT_128_2 := (
     fEditorMasterMin := 0, fEditorMasterMax := 360,
     fTableMasterMin := 0, fTableMasterMax := 6000,
     fEditorSlaveMin := 0, fEditorSlaveMax := 360,
     fTableSlaveMin := 0, fTableSlaveMax := 6000);
END_VAR
```

注意，执行该指令，从轴脱离凸轮运行状态，但其运行速度将仍以退出凸轮状态瞬间的速度继续运行，就如机械离合器断开之后，设备从动轮仍会靠惯性继续运转一样。此时必须有其他的 MC 功能块接替对从轴进行运动控制，比如 MC_Movexxx、MC_Halt、MC_Stop 等等都可以。

5.4 凸轮主轴相位调整 MC_Phasing 功能块

在有些设备的 CAM 同步运行中，有时会需要对凸轮从轴与主轴的相对相位需要作修正处理，可以采用 MC_Phasing 指令来实现。

该指令执行时，修改的是 CAM 从轴位置的运算结果，可以设置因相位调整产生的速度、加速度约束；调整时从轴的运行速度和位置保持连续；调整完成后，保持该相位差持续运行



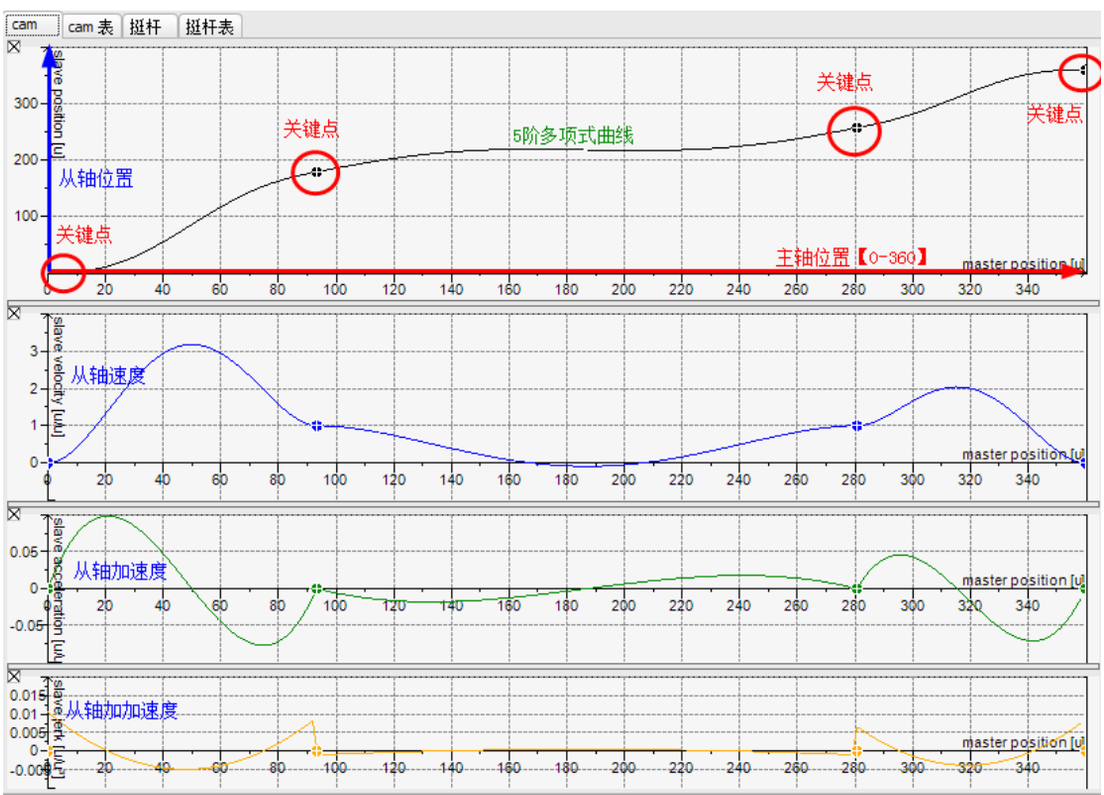
该指令在一些具有色标的同步控制中，需要调整色标在被剪切工件段中的位置，就需要用到该指令。

5.5 凸轮表设计与其数据结构

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。

5.5.1 凸轮表的特点

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。

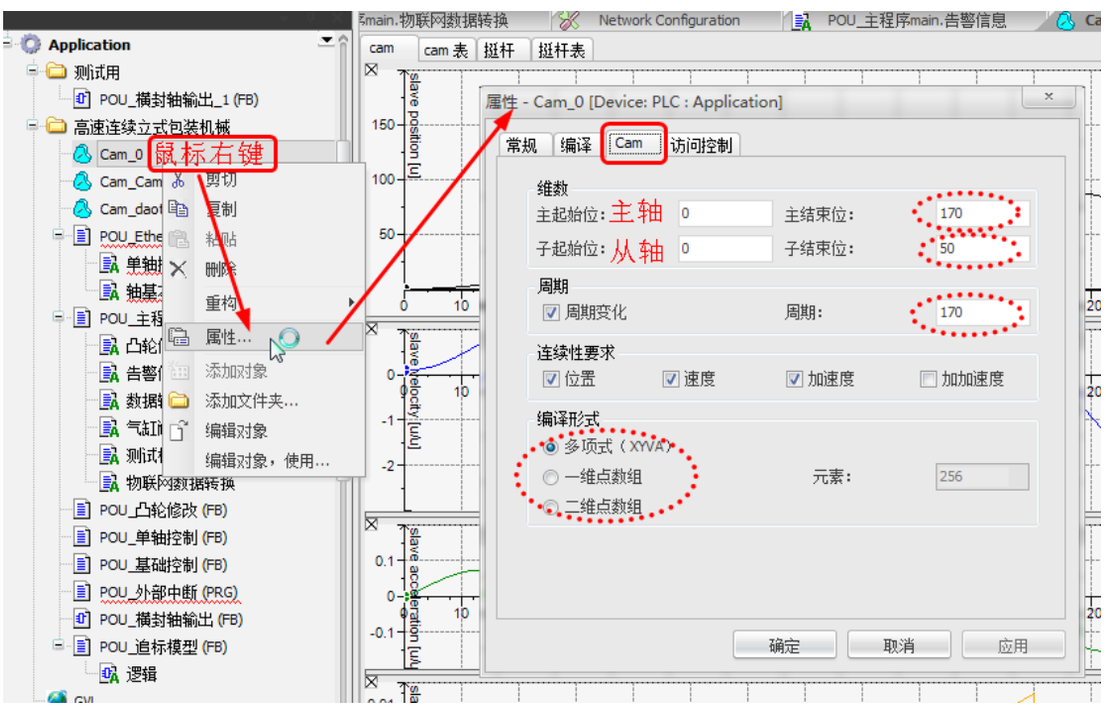
5.5.2 CAM 凸轮表的输入方法

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；

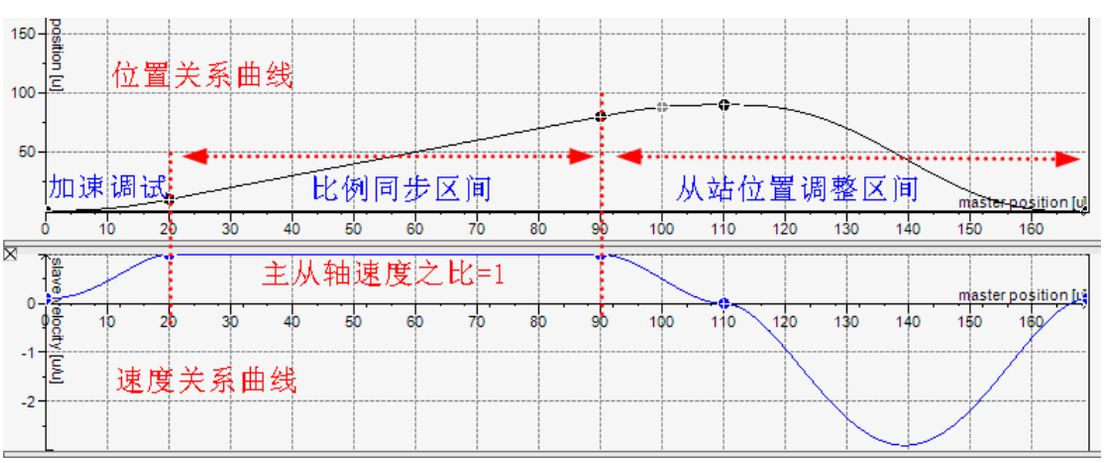


凸轮曲线中的关键点，往往与控制对象的机械运动要求相关，例如：

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；

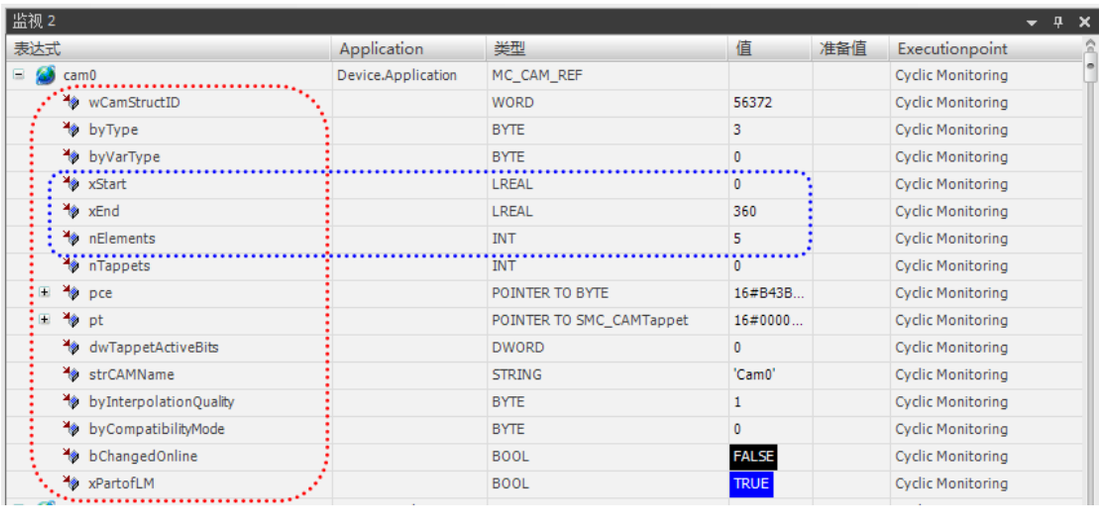
② 从轴往返行程起止点、同步运行区间的起始位置点、脱离同步位置点等，就是重要的关键点；

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



5.5.3 CAM 凸轮表的内部数据结构与数组

在 InoPro 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：



InoPro 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 CAM0 凸轮表时，系统自动默认声明了全局变量类型的 CAM0 数据结构，同时声明了 CAM0_A[i] 数组

例如在用户程序中，修改 CAM0 凸轮表关键点个数或坐标：

```
CAM0. nElements:=20; // 将关键点个数改为 20 个
  CAM0. xEnd:=500;     // 将主轴的结束点改为 500.            // 例如在用户程序中，修改其中 2 个关键点的坐标：
   CAM0_A[3].dx:=30;
   CAM0_A[3].dy:=45;
   CAM0_A[3].dv:=1;
   CAM0_A[3].da:=0;

   CAM0_A[4].dx:=60;
   CAM0_A[4].dy:=75;
   CAM0_A[4].dv:=1;
   CAM0_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。

5.5.4 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:=  虚主轴 ,
        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。

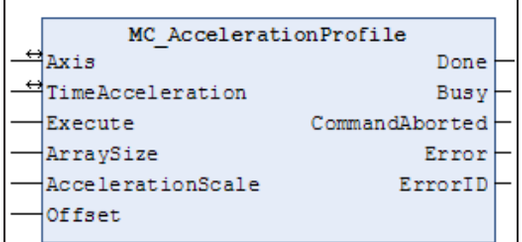
第6章 常用MC指令详解

六、常用 MC 指令详解

6.1 单轴指令

MC_AccelerationProfile

1） 指令格式

指令	名称	图形表现	ST 表现
MC_AccelerationProfile	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 w

◆ 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
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加减速度的轮廓运动模型 , 运行模式为 Discrete Motion, 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 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 ;

4） 时序图

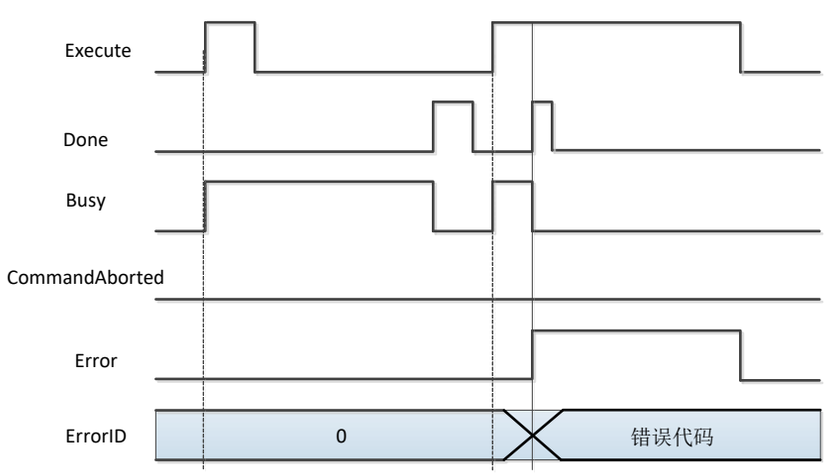
条件 MC_TA_Array 通过其他方式已经设置;

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;



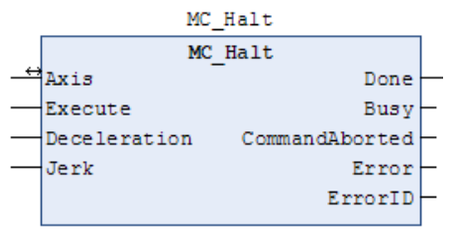
5） 错误说明

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。

MC_Halt

1） 指令格式

指令	名称	图形表现	ST 表现
MC_Halt	轴正常停止命令		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
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

本功能块为在正常运行的情况下停止一个轴的运动，而其它一个运动轴指令再运行时可以终止本指令的运行。

本功能块运行状态为运行状态 (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 ;

指令运行中的状态为 Discrete Motion, 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;

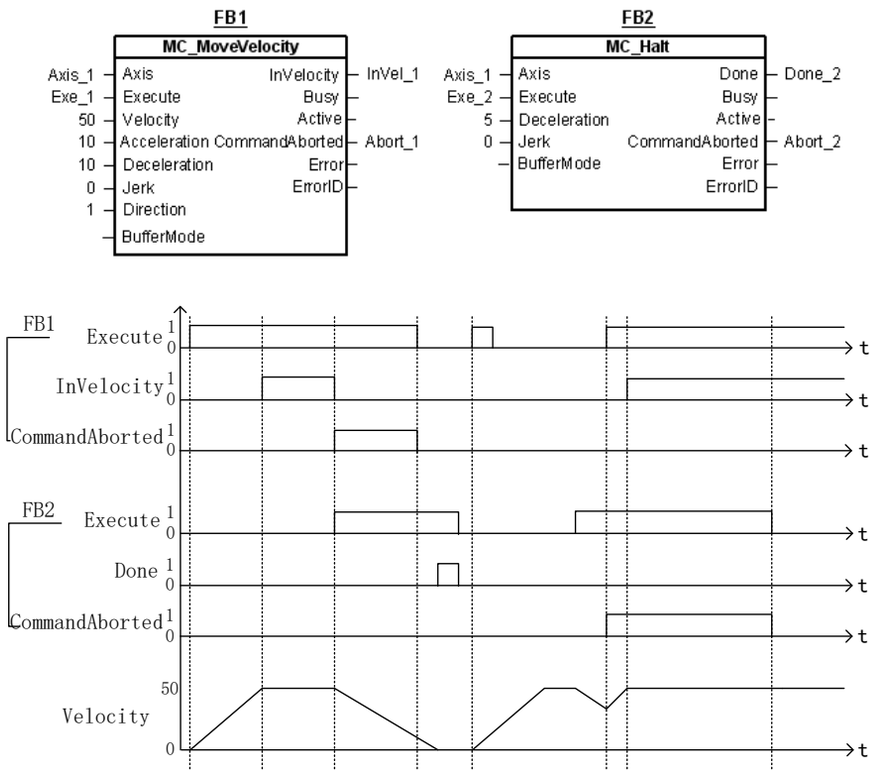
功能块的 Done 表示指令正常执行完成;

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。



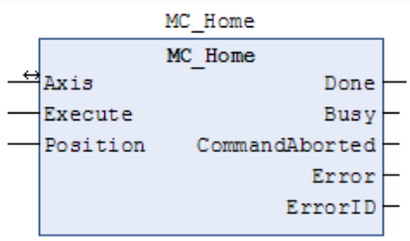
5） 错误说明

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。

MC_Home

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
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

本功能块为回零操作 , Position 数据位轴的零点位置。

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。

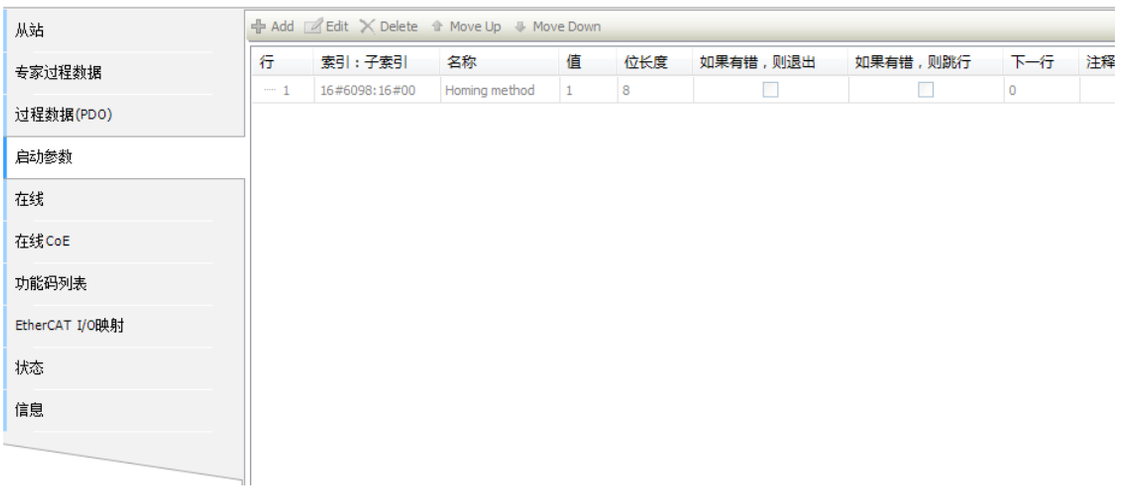
汇川伺服设置说明 :

◆ 在使用每根伺服轴在原点回归时必须要设定伺服参数的回归模式 ; 设置模式可以手工设置伺服的功能码 ;

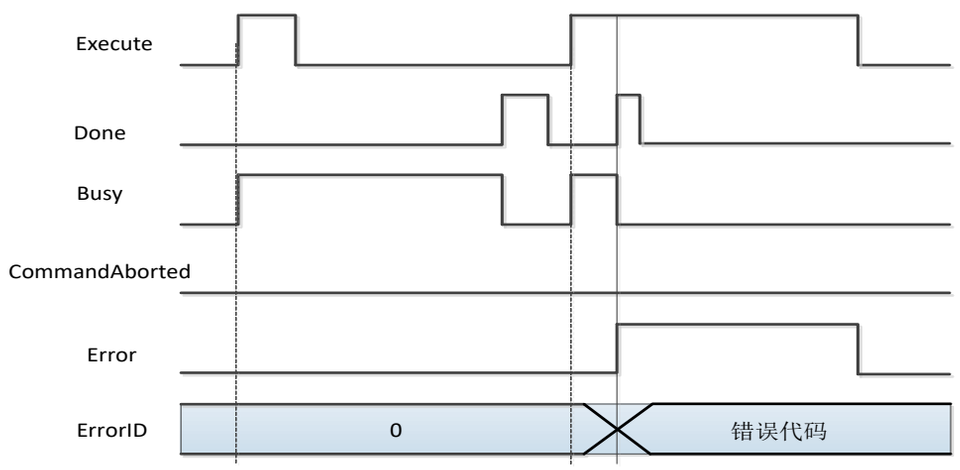
◆ 通过 AM600 从站的启动参数也可以配置相应的功能码 ; 采用通讯的方式必须设置如下索引和子索引的数据 ;

项目	索引	子索引	描述
回零方式	0x6098		根据伺服手册可以选择设定的具体参数
找原点速度	0x6099	0x01	一般定义的速度相对比较高 , 减少归零时间
找零点速度	0x6099	0x02	一般定义的速度相对比较低
原点回归加减速	0x609A		在原点回归时的加减速变化
原点回归超时时间	0x2005	0x24	回归时间超过设定时间 , 系统报 Err.601 错误

AM600 的画面设置参考：



4） 时序图



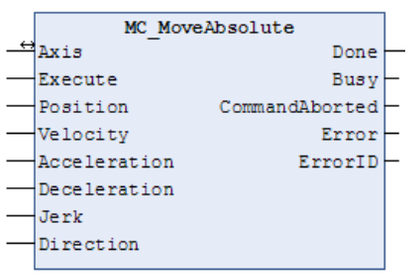
MC_MoveAbsolute

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

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_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的处理
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	指令极性	MC_DIRECTION	Negative,shortest Positive, 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)

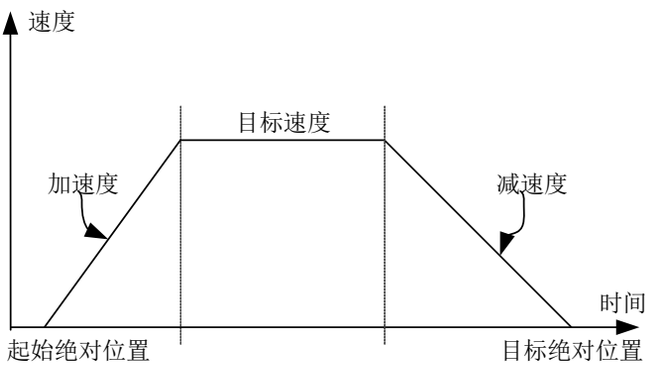
◆ 输出变量

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
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

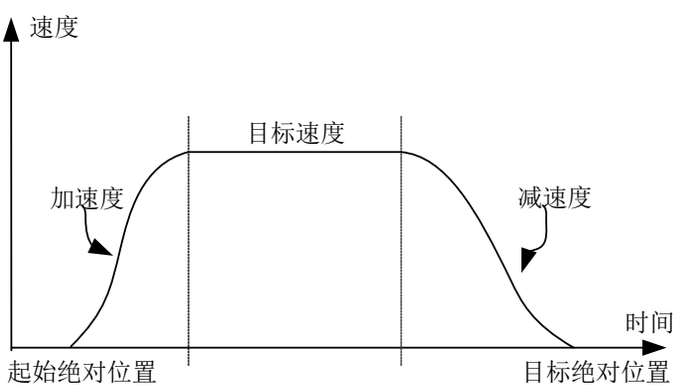
- 本功能块为轴绝对定位指令 , Position 数据为轴的绝对位置。
- 本功能块运行状态为 Standstill 中 , 指令运行时的状态为 Discrete Motion, 一个完整的运行过程一定要控制轴的不同运动状态。
- 启动指令为 Execute 的上升沿启动 , 本指令在 Discrete Motion 可以重复上升沿有效 , 每次都可以刷新最新的 Position 位置。
- Acceleration 或 Deceleration 为零，指令运行都为异常状态，但轴的状态为 Discrete Motion；
- 梯形加减速动作

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



- S 曲线加减速动作

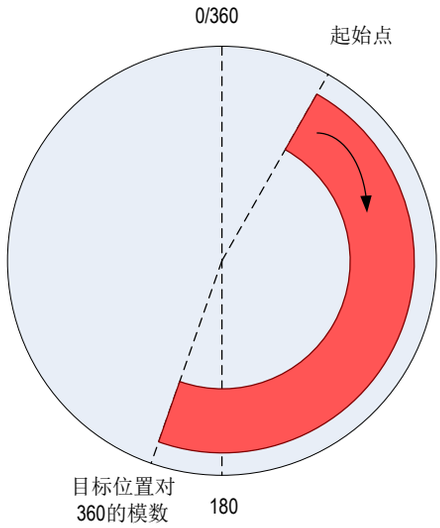
  Velocity、Acceleration、 Deceleration 和 Jerk 有数据；



- 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。

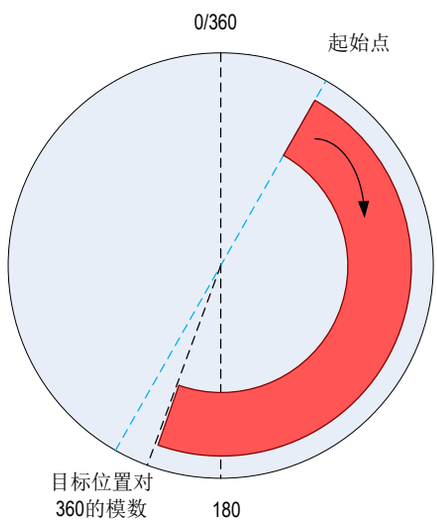
当 Position 对 360 的模值（Position/360 取余，比如 Position 为 380 则对 360 的模值为 20，Position 为 350 则对 360 模值为 350）> 起始绝对位置时，此时轴朝正向运行 Position 对 360 的模值 - 起始绝对位置的距离。



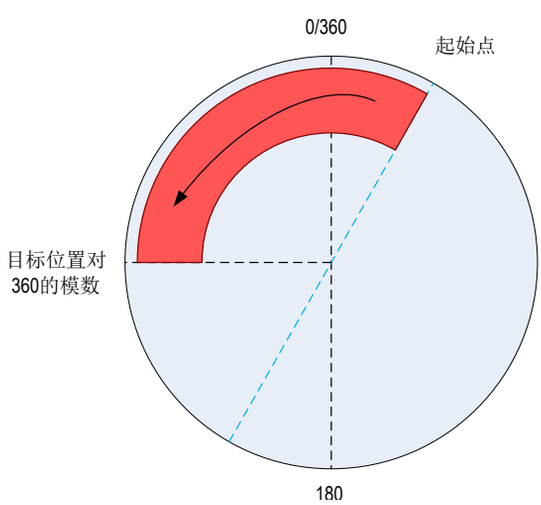
当 Position 对 360 的模值（Position/360 取余，比如 Position 为 380 则对 360 的模值为 20）< 起始绝对位置时，此时轴朝正向运行 360- 起始绝对位置 + Position 对 360 的模值的距离。

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

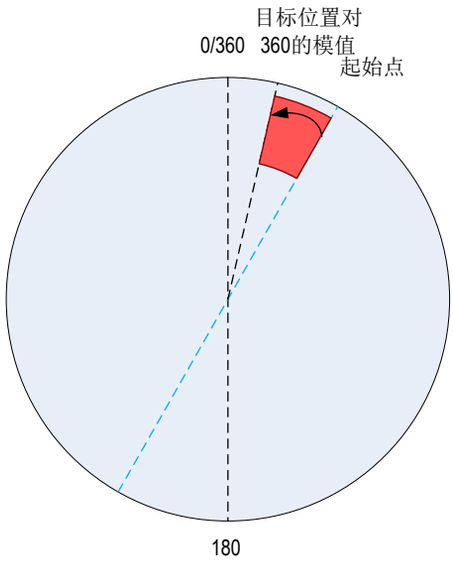
当 =0<XPosition- 起始绝对位置 <180 时 , 轴朝正向运行 XPosition- 起始绝对位置的距离。



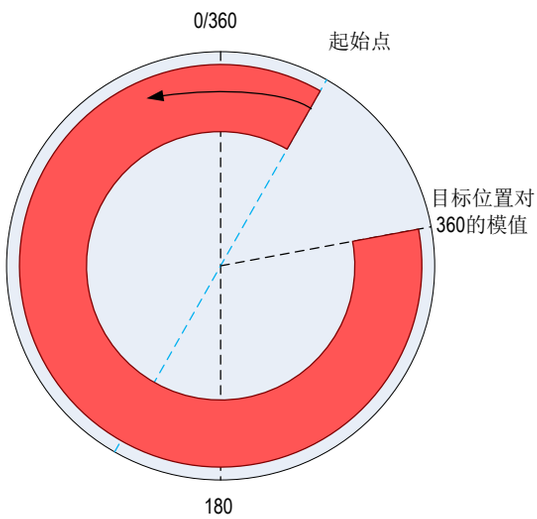
当 180<XPosition- 起始绝对位置时 , 轴朝反向运行 360-XPosition+ 起始绝对位置的距离。



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。

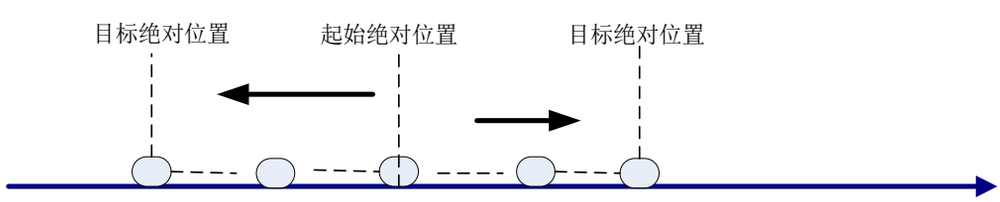


③ 轴旋转周期设置为 360、Direction 设置为 shortest 或者 Negative。Position 对 360 的模值为 XPosition 轴朝反向运行起始绝对位置 +360-XPosition 的距离。



◆ 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。



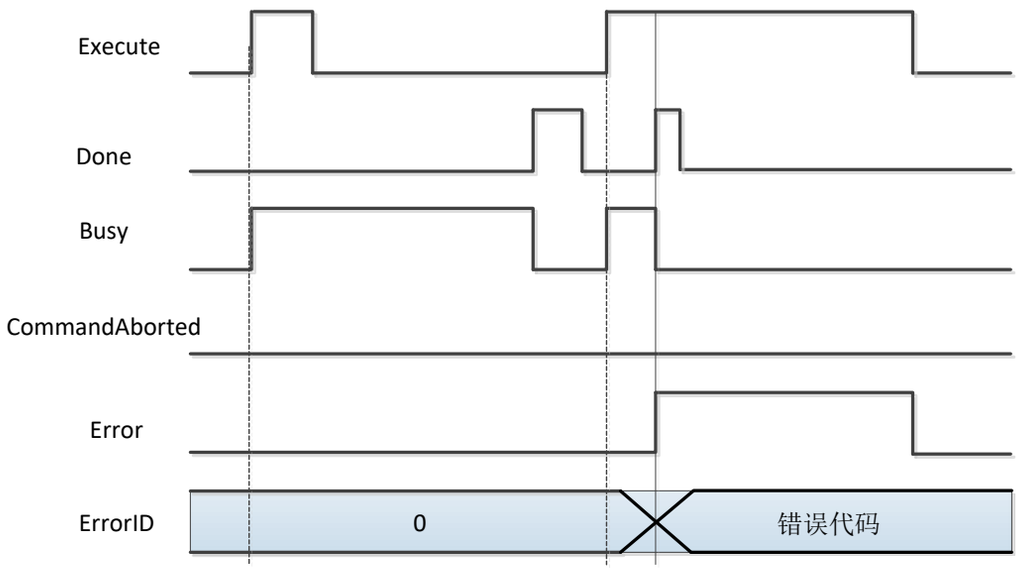
4） 时序图

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；



MC_MoveAdditive

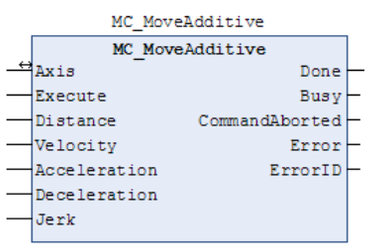
轴在原来指令位置上再叠加 Distance 指定的数据，用于运动轴控制过程的在线叠加位置；本指令运行前设置好相关的

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) 的赋值为 0

指令运行错误；本指令在 Discrete Motion 状态下可以在任何时刻添加 MC_MoveAddtive 的相关执行过程；在 Continuous Motion

中只能在指令执行的某段中；MC_MoveAddtive 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_MoveAdditive	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

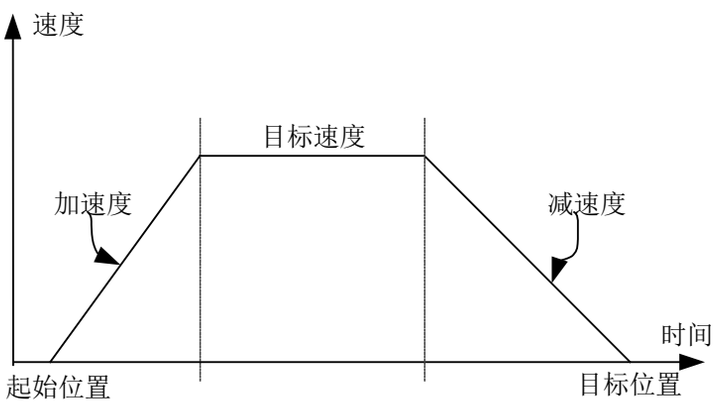
输出变量	名称	数据类型	有效范围	初始值	描述
Done	指令执行完成	BOOL	TRUE,FALSE	FALSE	轴指令执行完成，置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码

3） 功能说明

- 本功能块为叠加位置指令 , Distance 数据为轴的叠加数据；
- 本功能块运行状态如果为 Discrete Motion, 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 Discrete Motion；
- 启动指令为 Execute 的上升沿启动。

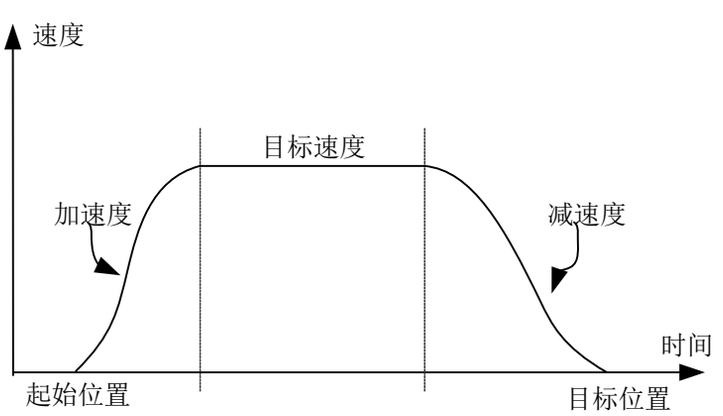
梯形加减速动作

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



S 曲线加减速动作

Velocity、Acceleration、 Deceleration 和 Jerk 有数据；



4） 时序图

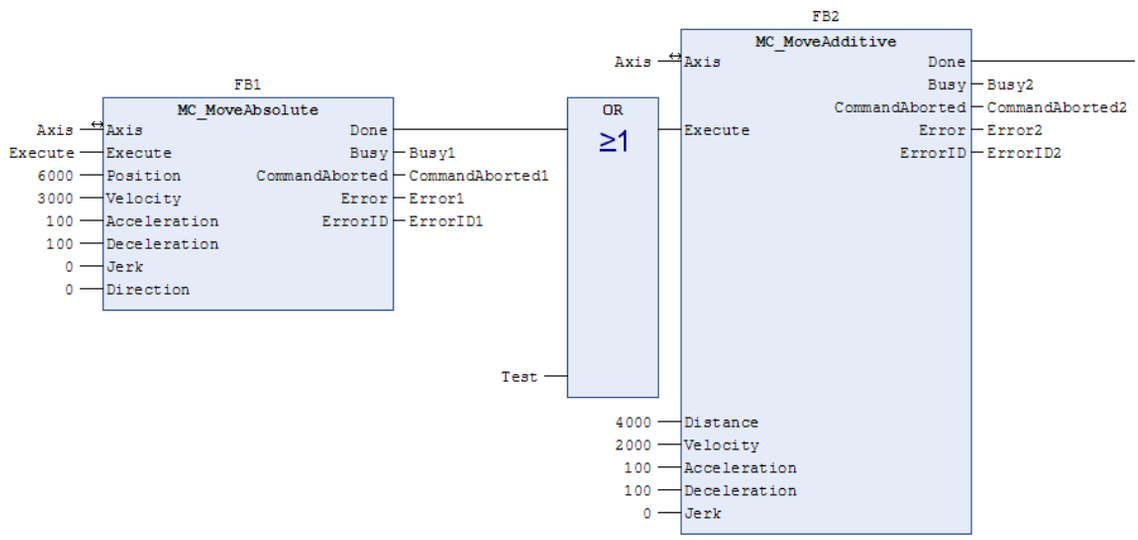
轴必须处于 Standstill 状态指令才能运行；

功能块的 Execute 必须有上升沿的条件；

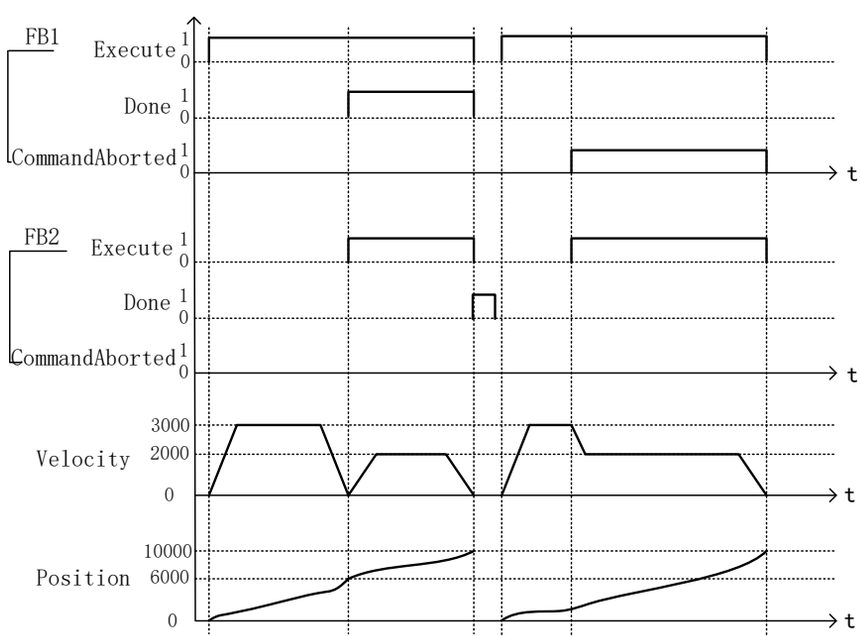
功能块的 Done 表示指令正常执行完成；

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：



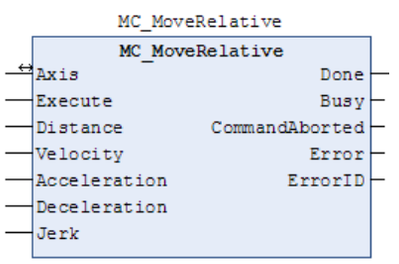
MC_MoveRelative

轴按相对位置运行 (单位按轴设置), 相对位置由 Distance 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

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_MoveRelative	轴相对定位指令		MC_MoveRelative(Axis:= , Execute:= , Distance:= , Velocity:= 10, 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	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

◆ 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

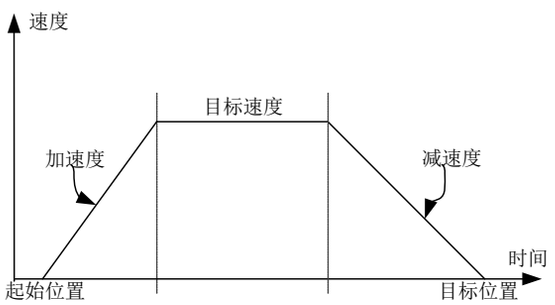
本功能块运行状态为 Standstill 中， 指令运行时的状态为 Discrete Motion，在指令执行中关注本轴的运行状态，避免打断本轴的其他指令或被其他指令打断本轴的执行。

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；

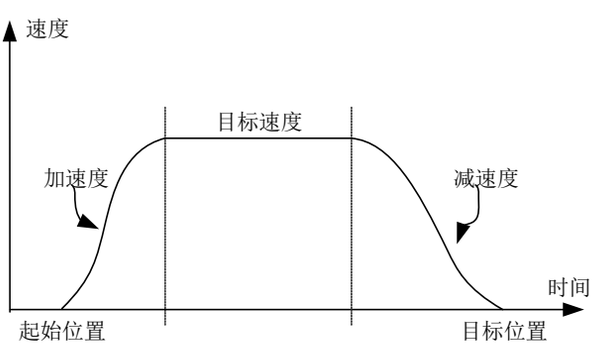
- 梯形加减速动作

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



- S 曲线加减速动作

  Velocity、Acceleration、 Deceleration 和 Jerk 有数据；

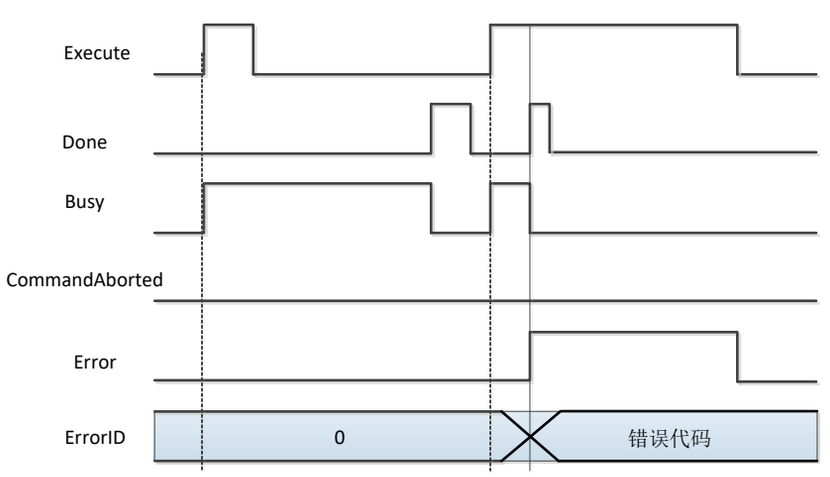


4） 时序图

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；



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 MC_MoveSuperImposed Axis — Done Execute — Busy Distance — CommandAborted VelocityDiff — Error Acceleration — ErrorID Deceleration Jerk	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。

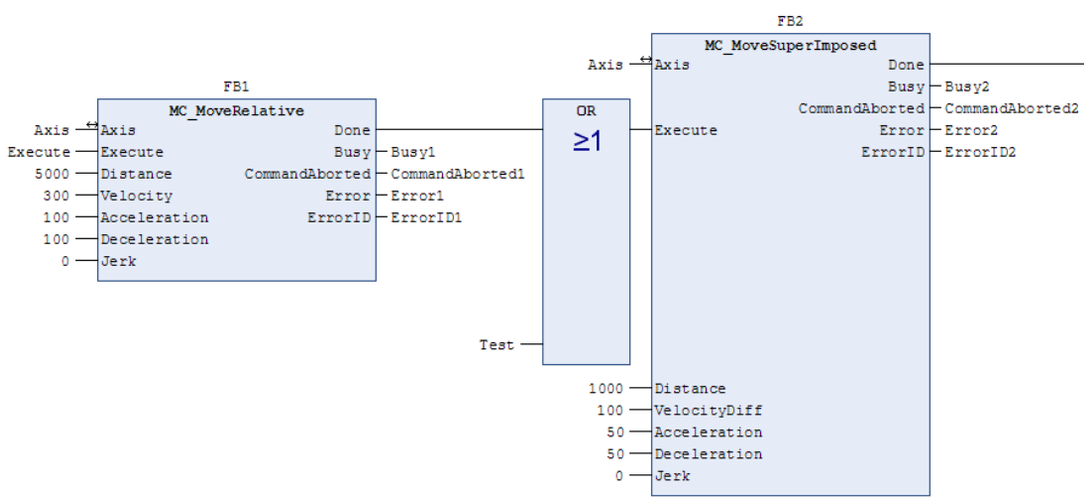
4） 时序图

功能块的 Execute 必须有上升沿的条件；

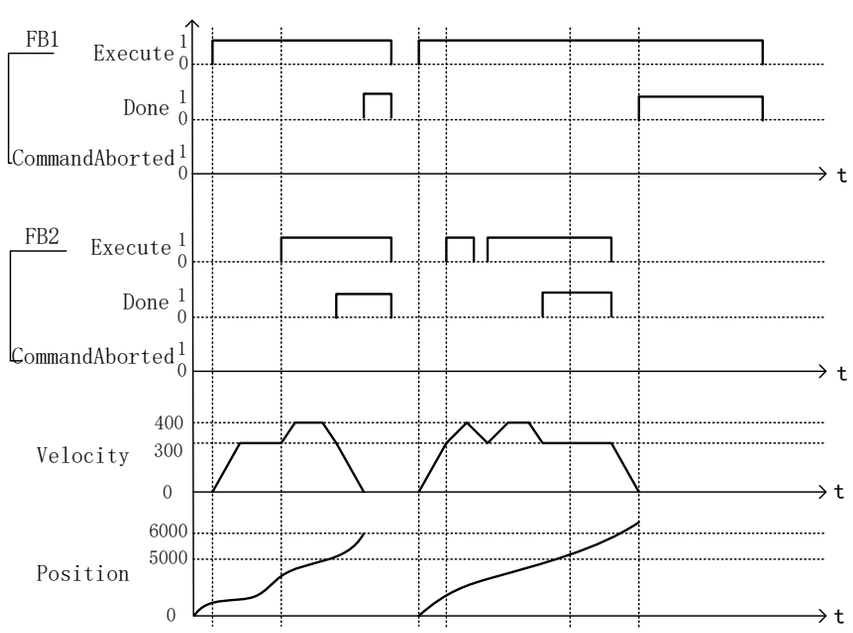
功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；

◆ 举例说明



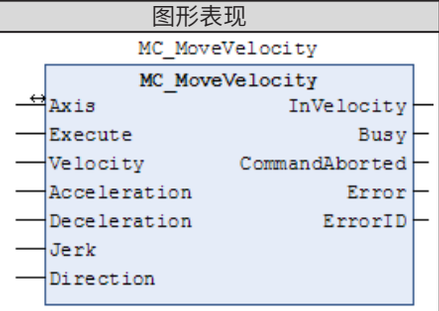
◆ 时序操作说明 :



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	速度设定值	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	positive, negative, current	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_ER-ROR	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。

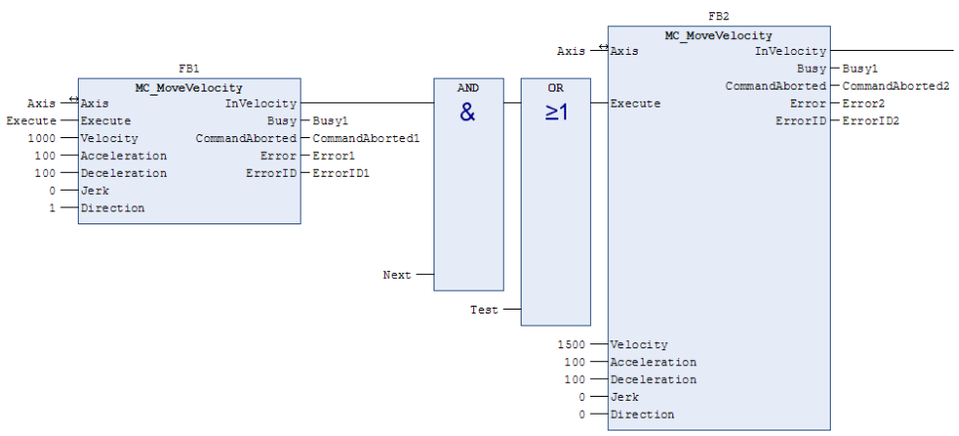
◆ 时序图

功能块的 Execute 必须有上升沿的条件；

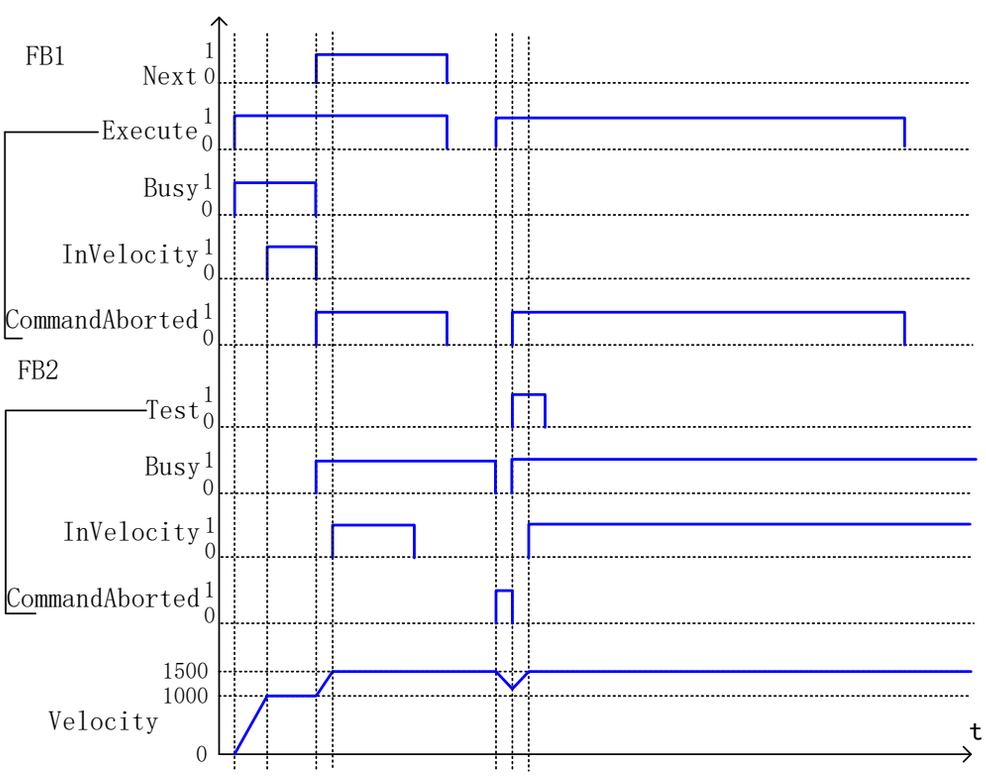
功能块的 InVelocity 表示指令的运行速度达到设定值；

功能块的 Busy 表示当前功能块正在执行中；

◆ 举例说明

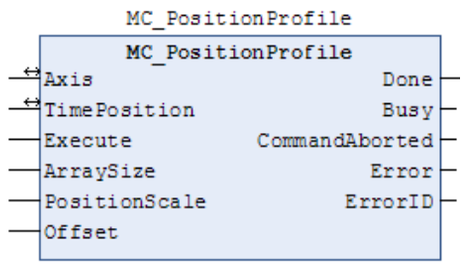


◆ 时序操作说明：



MC_PositionProfile

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_ER-ROR	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 Discrete Motion, 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 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。

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

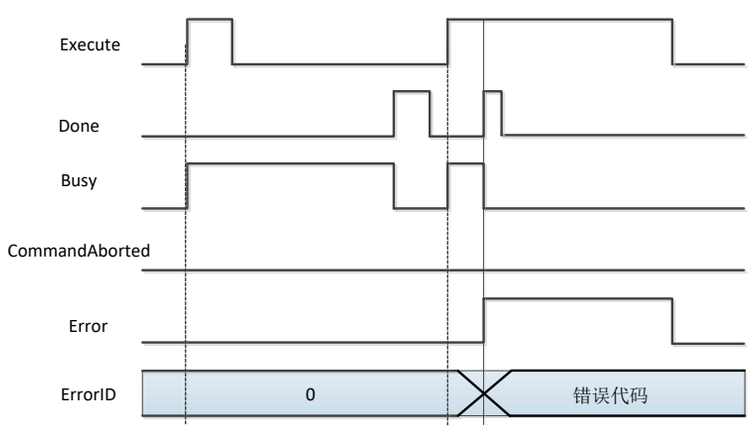
条件 MC_TP_Array 通过其他方式已经设置才能运行位置轮廓曲线指令;

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

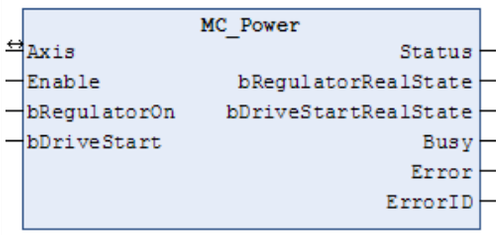
错误的出现为轴状态不是在 Standstill 中启动指令或指令系统中的参数错误，出现轴错误只能清除错误后才开始运行。

【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。

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
Enable	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

◆ 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

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。

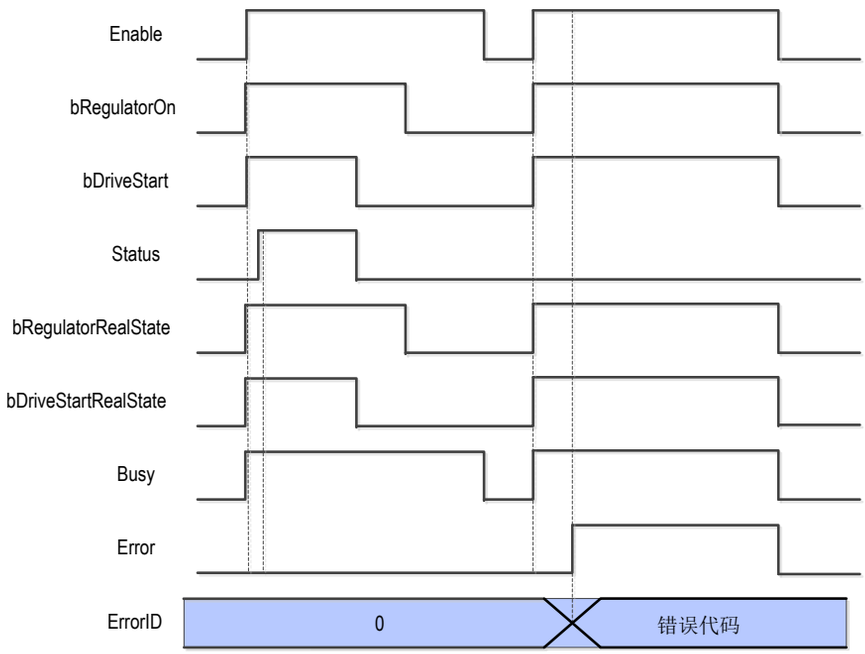
如果功能块 MC_Power 已经被调用，并且 bRegulatorOn=TRUE，如果此时轴没有错误发生，那么功能块将设置相关轴的轴状态（nAxisState）为 standstill 状态；如果有错误发生，将输出相应的错误状态。

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。

◆ 时序图

将 Enable 设为 TRUE,bRegulatorOn 设为 TRUE,bDriveStart 设为 TRUE, 表示正在受理指令的 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 根轴只能设 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。

【注意】：请阅读“附录 C 错误代码说明”以了解相关错误代码说明。

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_ReadActualPosition	实际位置读取指令	MC_ReadActualPosition Axis Enable Valid Busy Error ErrorID Position	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

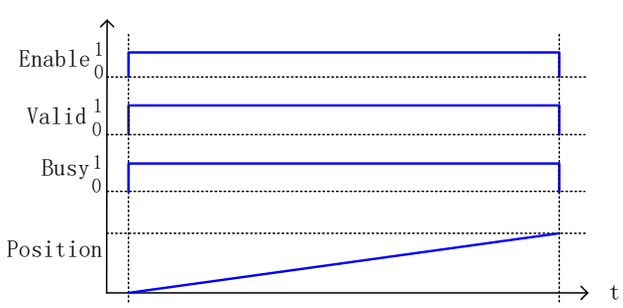
3） 功能说明

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；

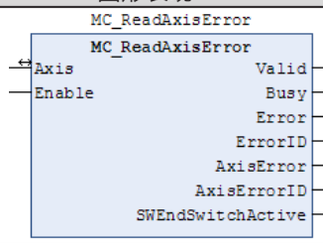
时序操作说明：



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_ReadAxisEr-ror	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，对应的标示置位
AxisErrorID	轴错误码	DWORD		0	读出轴为错误码
SWEndSwitchActive	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；

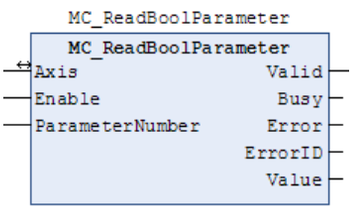
功能块的 Valid 表示读出的 AxisError 和 AxisErrorID 为有效的数据值；

功能块的 Busy 表示当前功能块正在执行中；

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_ReadBoolParameter	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
ParameterNumber	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_DOWRD(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。

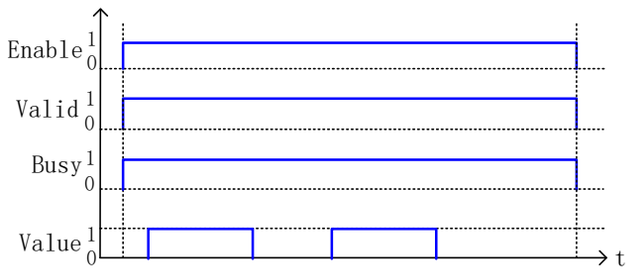
◆ 时序图

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；

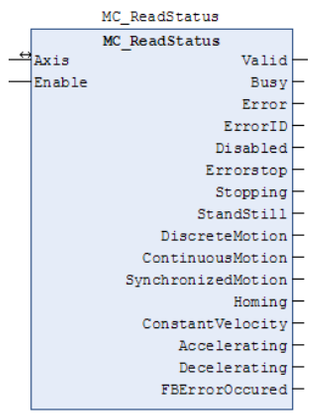
◆ 时序操作说明：



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	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
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
DiscreteMotion	轴离散运动状态	BOOL	TRUE,FALSE	FALSE	轴在离散运动状态中为 TRUE
ContinuousMotion	轴连续运动状态	BOOL	TRUE,FALSE	FALSE	轴在连续运动状态中为 TRUE
SynchronizedMotion	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
ConstantVelocity	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

FBErrorOccured	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；

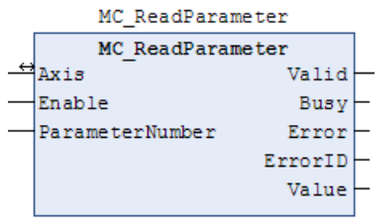
功能块的 Valid 表示读出的后面个状态标志的各种数据；

功能块的 Busy 表示当前功能块正在执行中。

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
ParameterNumber	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_DOWRD(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

3） 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。

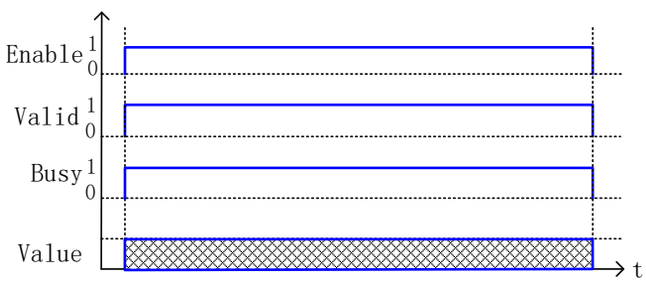
◆ 时序图

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 :



MC_Reset

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 MC_Reset Axis Done Execute Busy Error ErrorID	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

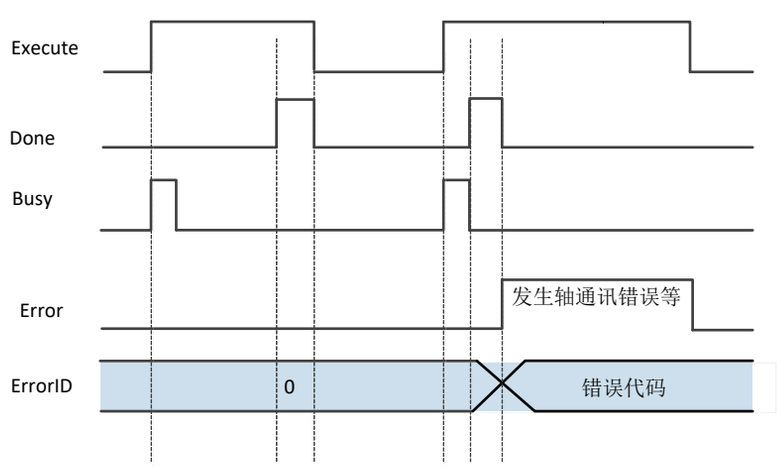
3） 功能说明

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



MC_Stop

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	MC_Stop(Axis:= , Execute:= , Deceleration:= , Jerk:= , 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
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
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。

当轴状态处于 stopping 时 ,Execute 为 Flase 状态 ,Done 输出状态为 True, 轴状态变为 Standstill;

本功能块运行状态为运行状态 (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 指令系统处于 Errorstop 状态

◆ 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；

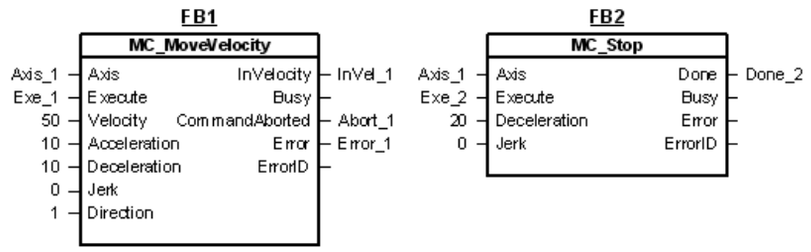
功能块的 Done 表示指令正常执行完成；

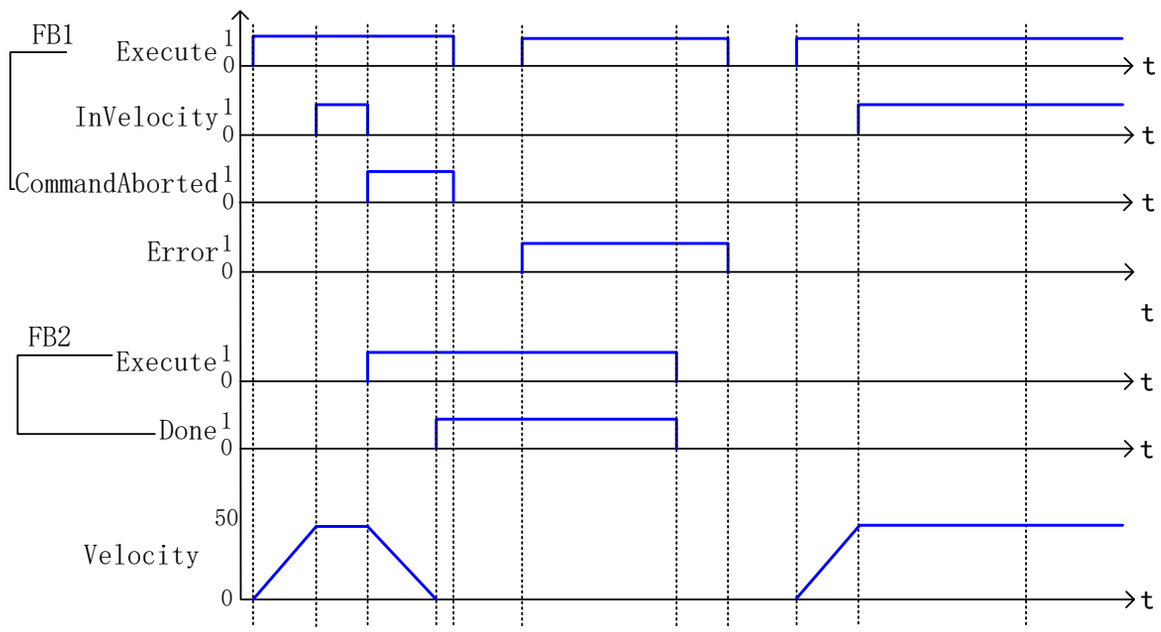
功能块的 Busy 表示当前功能块正在执行中；

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_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 .





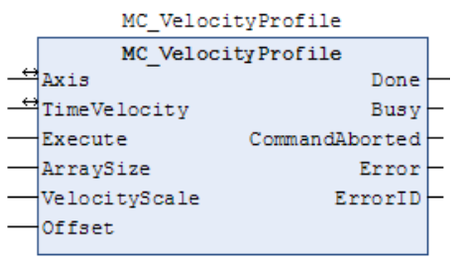
4） 错误说明

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。

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
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_ER-ROR	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

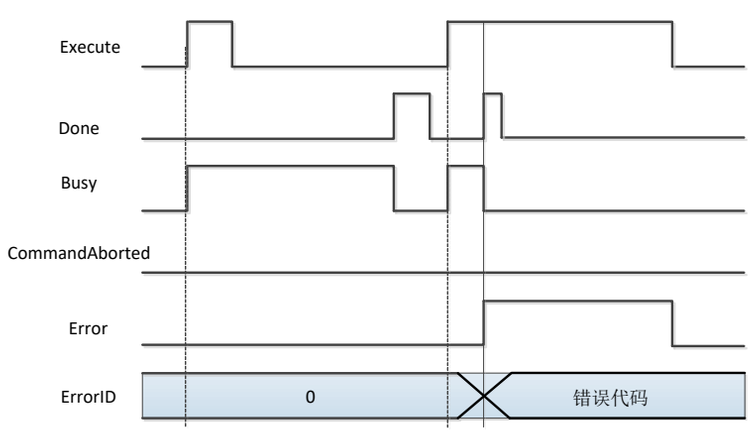
条件 MC_TV_Array 通过其他方式已经设置才能运行位置轮廓曲线指令；

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。

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 MC_WriteBoolParameter Axis Execute ParameterNumber Value Done Busy Error ErrorID	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
ParameterNumber	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_DOWRD(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。

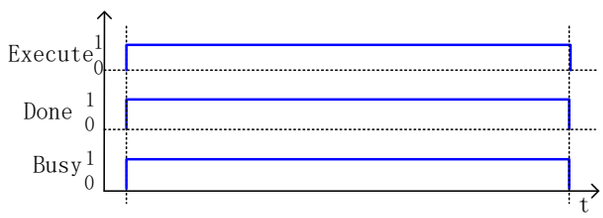
◆ 时序图

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：



MC_WriteParameter

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_ WriteParameter	设置轴参数	MC_WriteParameter MC_WriteParameter Axis Done Execute Busy ParameterNumber Error Value ErrorID	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
ParameterNumber	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_DOWRD(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

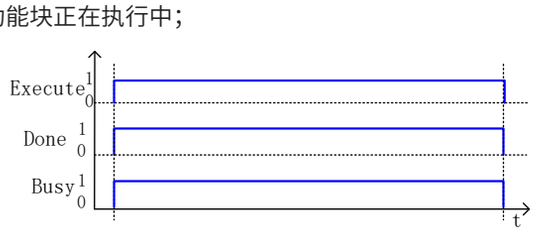
通过 MC_ WriteParameter 设置轴的位参数，指令为 Execute 上升沿触发。指令可以重复多次使用，互不影响。

◆ 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；

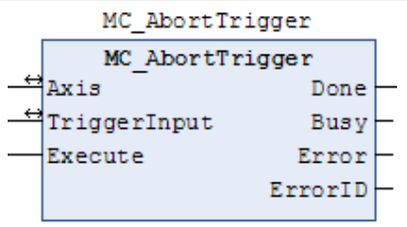
功能块的 Busy 表示当前功能块正在执行中；



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	iTriggerNumber	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 详细参考《IS620N 系列伺服设计维护使用手册》
	bFastLatching	BOOL	TRUE	指定锁存触发的模式： TRUE：驱动器模式 FALSE：控制器模式
	bInput	BOOL		当 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

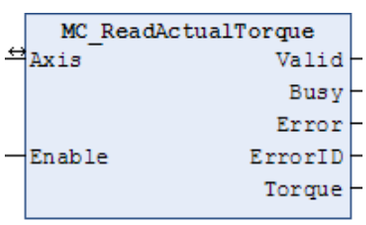
3） 功能说明

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；

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

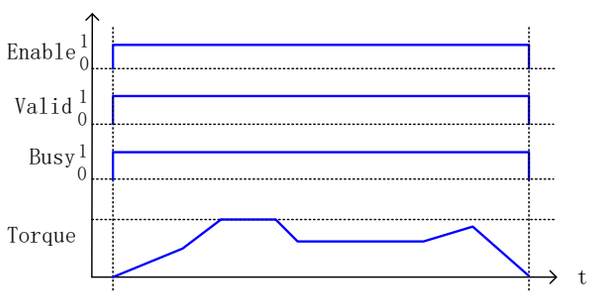
通过 MC_ReadActualTorque 读取驱动器中的当前力矩值指令，指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

◆ 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；



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_ReadActualVelocity Axis Valid Busy Error Enable ErrorID Velocity	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

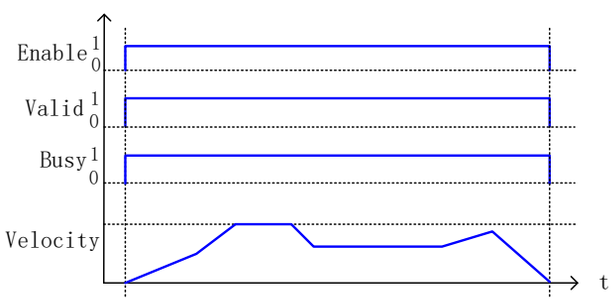
通过 MC_ReadActualVelocity 读取驱动器中的当前速度值指令，指令为 Enable 电平使能效应。指令可以重复多次使用，互不影响。

◆ 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；

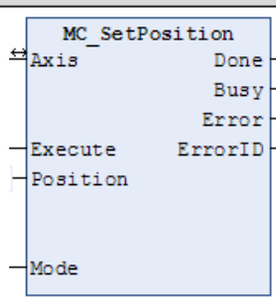
功能块的 Busy 表示当前功能块正在执行中；



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

3） 功能说明

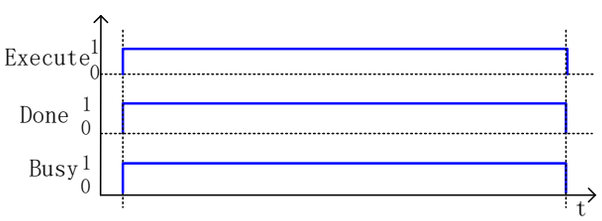
通过 MC_ SetPosition 设置轴的位置参数，不产生任何位移，但形成了坐标偏移；指令为 Execute 上升沿触发；指令可以重复多次使用，互不影响。

◆ 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；

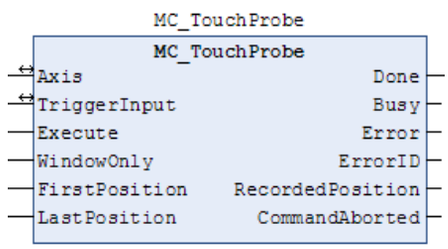
功能块的 Busy 表示当前功能块正在执行中；



MC_TouchProbe

指令通过外部信号触发，保存当前轴的位置数据。

1） 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=>);

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
WindowOnly	触发窗口	BOOL	TRUE,FALSE	FALSE	
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

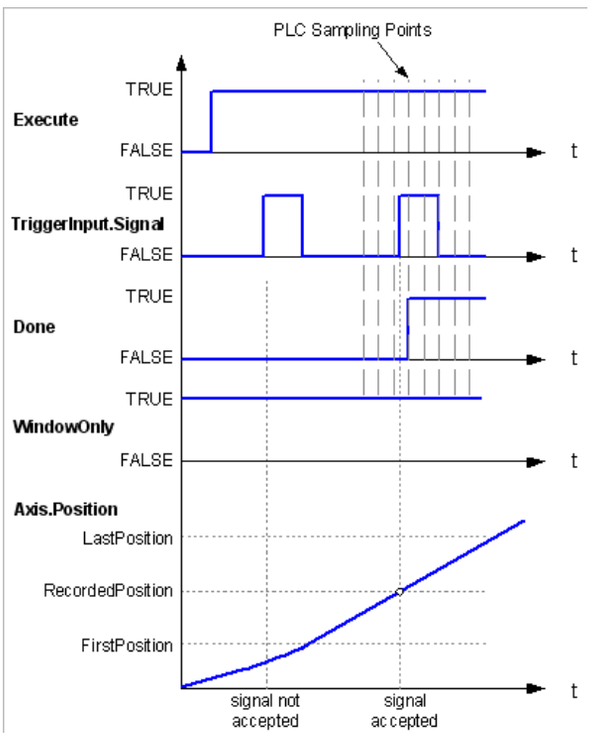
输出变量	名称	数据类型	有效范围	初始值	描述
Done	设置操作成功	BOOL	TRUE,FALSE	FALSE	设置操作成功置为 TRUE
Busy	指令正在执行	BOOL	TRUE,FALSE	FALSE	当前指令正在执行中，置为 TRUE
Error	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
ErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码
RecordedPosition	触发记录位置	LREAL	—	0	触发发生时当前的位置
CommandAbort	指令被中断	BOOL	TRUE,FALSE	FALSE	当前指令被中断，置为 TRUE

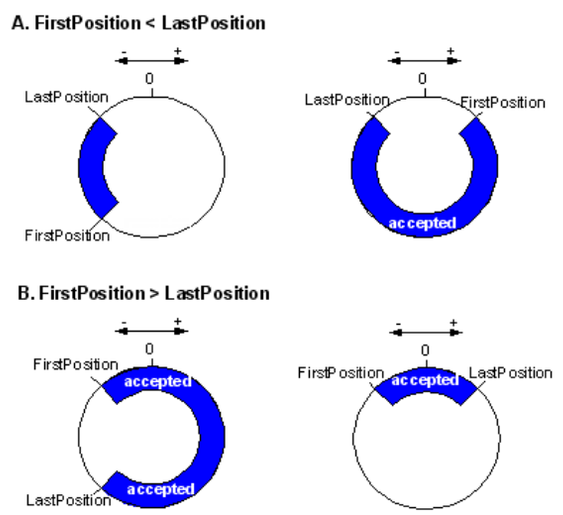
3） 功能说明

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，等到控制器

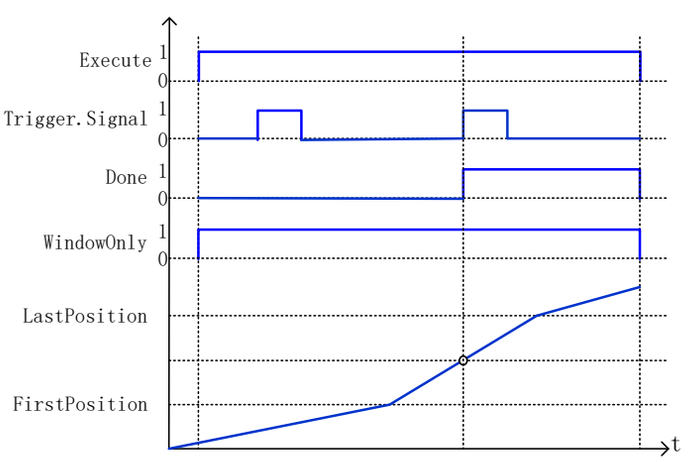




◆ 时序图

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；



在 Position 为循环计数单元，触发信号的位置可以重复使用。

SMC_MoveContinuousAbsolute

轴按绝对位置连续运行 (单位按轴设置), 绝对位置由 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_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
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
Direction	运行方向	shortest	数据范围	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；

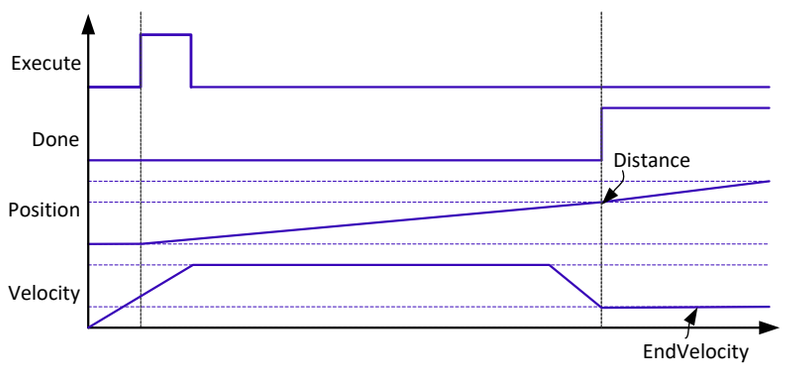
◆ 时序图

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；



SMC_MoveContinuousRelative

轴按相对位置连续运行 (单位按轴设置), 绝对位置由 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_MoveContinuousRelative	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
EndVelocityDirection	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
Direction	运行方向	shortest	数据范围	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。

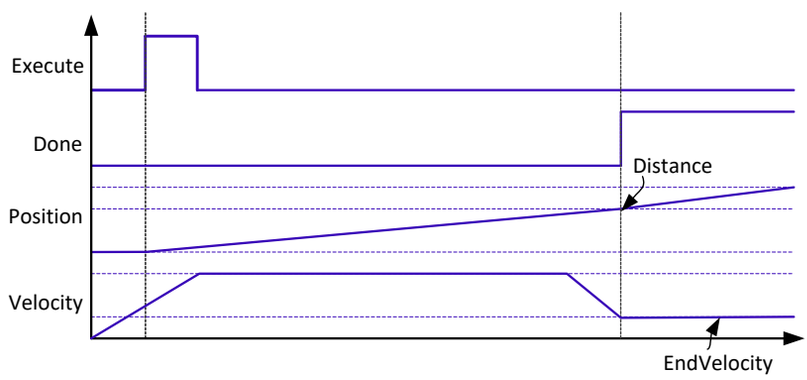
Acceleration 或 Deceleration 为零，指令运行都为异常状态，但轴的状态为 Discrete Motion；

◆ 时序图

功能块的 Execute 必须有上升沿的条件；

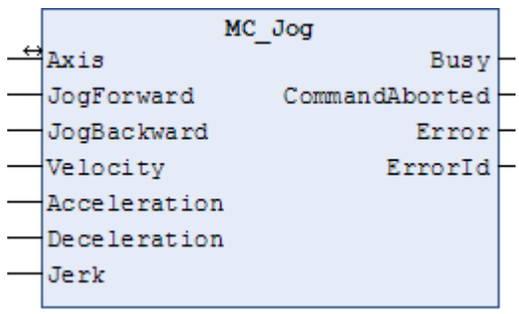
功能块的 Done 表示指令正常执行完成；

功能块的 Busy 表示当前功能块正在执行中；



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 /s]
Deceleration	减速度	LREAL	正数或“0”	0	指定减速度。单位：[指令单位 /s]

◆ 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。

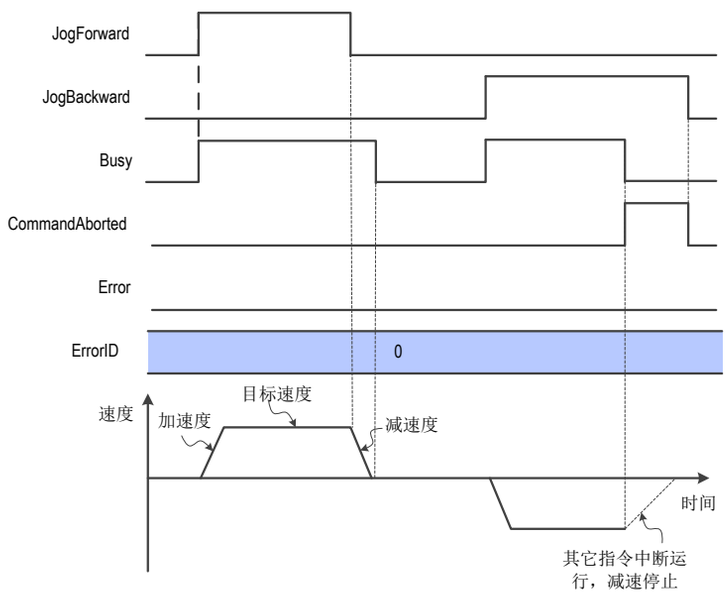
如果 MC_Jog 指令的指令速度设置值超过轴参数中的点动最高速度，则以点动最高速度执行。

◆ 时序图

在启动 JogForward（正向运行有效）或 JogBackward（负向运行有效）的同时，Busy（执行中）变为 TRUE。

在 JogForward（正向运行有效）或 JogBackward（负向运行有效）的下降沿开始减速并停止轴的同时，Busy（执行中）变为 FALSE。

利用其它指令中止本指令时，CommandAborted（执行中断）变为 TRUE，Busy（执行中）变为 FALSE。

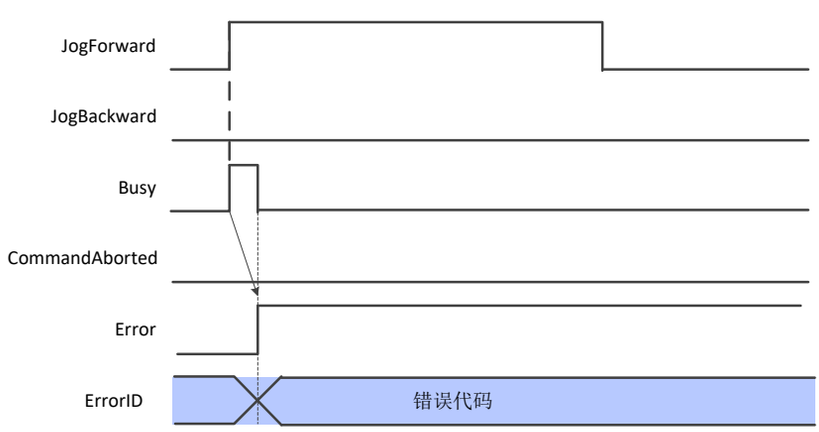


4） 错误说明

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

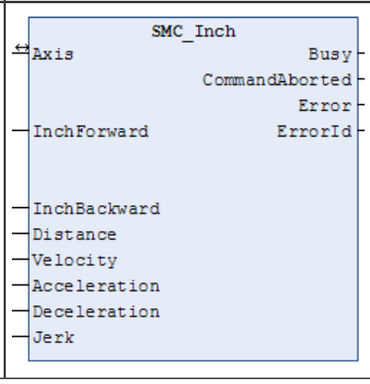


关于指令发生的异常，请阅读“附录 C 错误代码说明”以了解相关错误代码说明。。

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) 以正向运动到设定位置。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
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

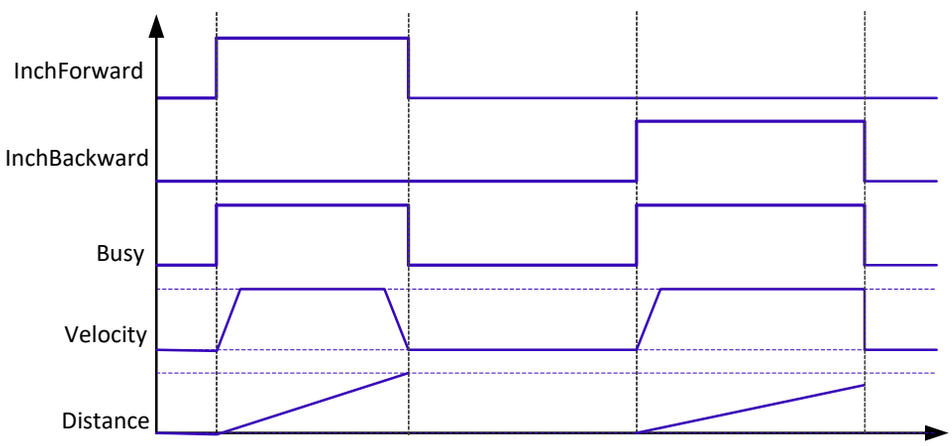
本功能块运行状态为 Standstill 中， 指令运行时的状态为 Discrete Motion，在指令执行中关注本轴的运行状态，避免打断本轴的其他指令或被其他指令打断本轴的执行。

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；

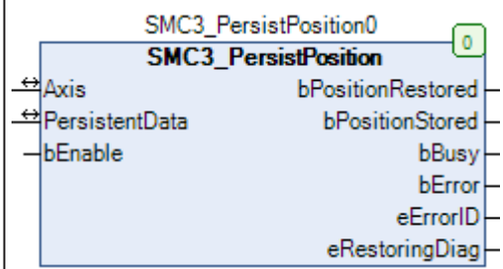
功能块的 Busy 表示当前功能块正在执行中；



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_Persist-PositionDiag		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

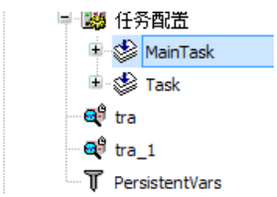
PLC 重启 bEnable 信号为 TRUE，则 bPositionRestroed 输出为 TRUE。

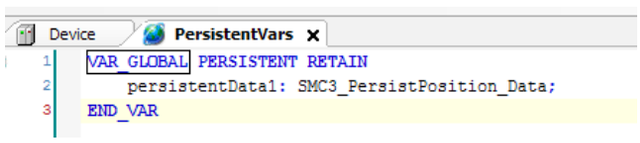
不支持虚轴跟逻辑轴。

在此声明：AM600 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

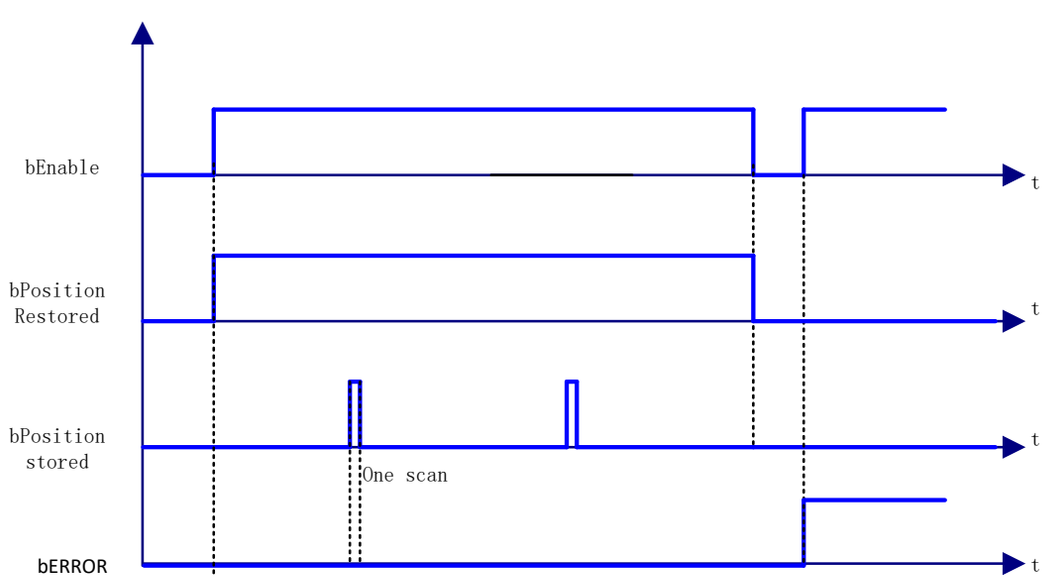
◆ 申明部分：

```
VAR
    SMC3_PersistPosition_1:SMC3_PersistPosition;
END_VAR
```

程序部分：

```
//绝对值位置保存
SMC3_PersistPosition_1(Axis:=X_Axis , PersistentData:=persistentData1 ,bEnable:=TRUE );
```

◆ 时序图



4） 错误说明

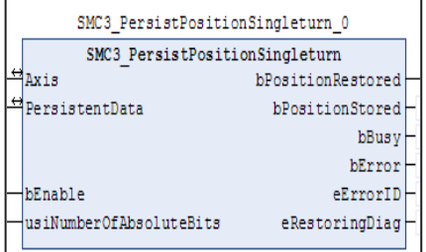
输入轴为虚拟轴或者逻辑轴会导致错误输出；轴有错误时会导致错误输出。

【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。

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_ PersistPositionSingleturn	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
PersistentData	保持数据	SMC3_ PersistPositionSingletrun_ 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

输出变量	名称	数据类型	有效范围	初始值	描述
eRestoringDiag	恢复诊断	SMC3_ PersistPosi-tionDiag		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

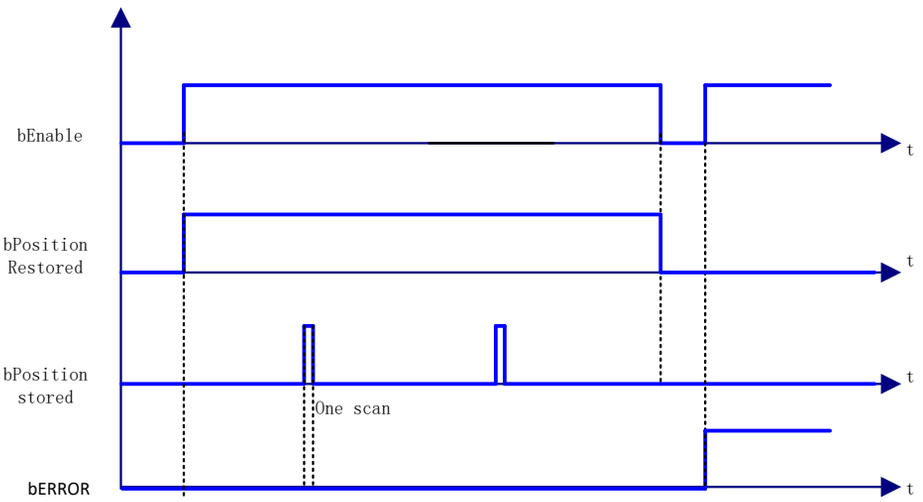
◆ 申明部分：

```
VAR
    SMC3_PersistPosition_2: SMC3_PersistPositionSingleTurn_Data;
END_VAR
```

◆ 程序部分：

```
//绝对值位置保存
SMC3_PersistPosition_2(Axis:=Y_Axis , PersistentData:=persistentData2 ,bEnable:=TRUE );
```

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。

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_PersistPositionLogical Axis → bPositionRestored PersistentData → bPositionStored bBusy bError bEnable → eErrorID eRestoringDiag	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,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
eRestoring-Diag	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

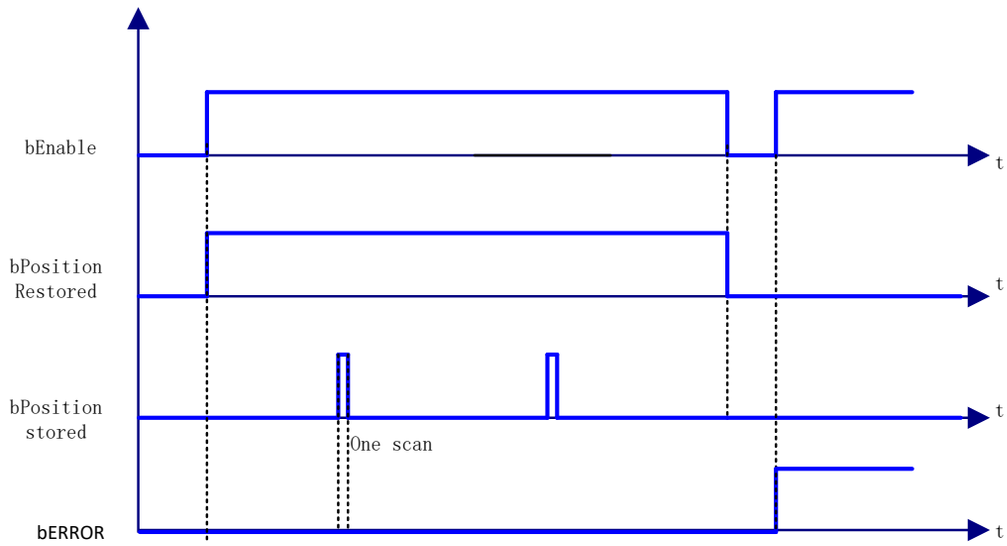
◆ 申明部分：

```
VAR
    SMC3_PersistPosition_3:SMC3_PersistPositionLogical;
END_VAR
```

◆ 程序部分：

```
//绝对值位置保存
SMC3_PersistPosition_1(Axis:=X_Axis , PersistentData:=persistentData1 ,bEnable:=TRUE );
```

◆ 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。

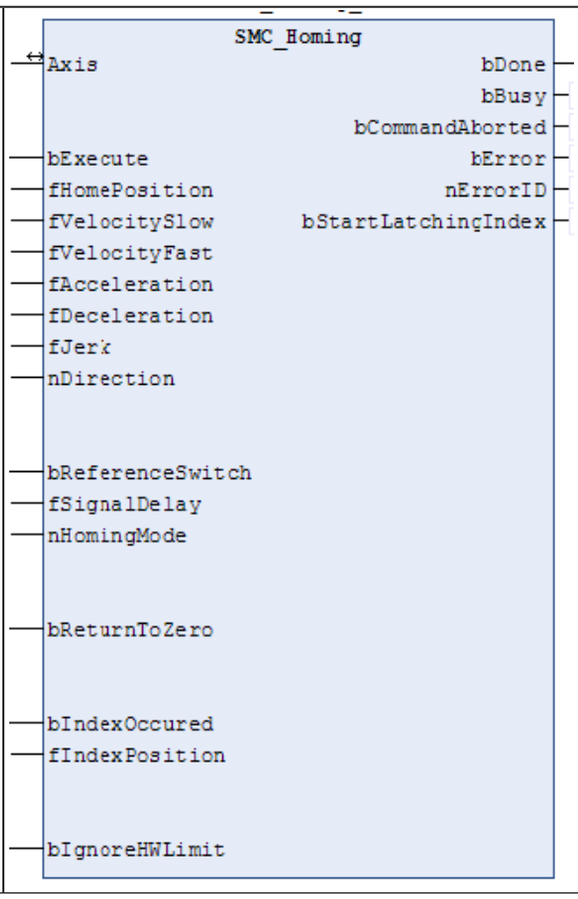
【注意】：请阅读“附录 C 错误代码说明”以了解相关错误代码说明。

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
bReferenceSwitch	参考开关	BOOL	TRUE,FALSE	FALSE	连接参考开关，TRUE: 参开开关触发，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。
nHomingMode	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）

输入变量	名称	数据类型	有效范围	初始值	描述
bIndexOccured		BOOL	TRUE,FALSE	FALSE	True, 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
bStartLatchingIndex		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，

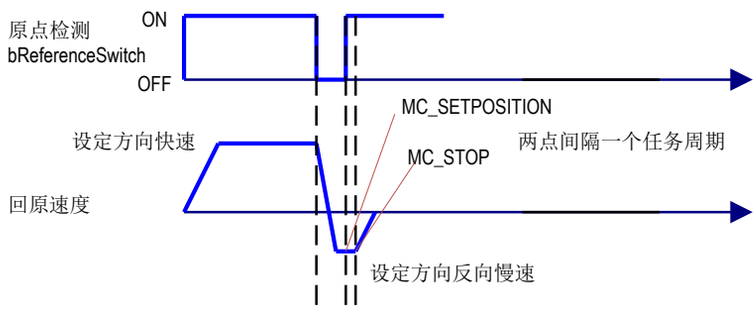
使能回零指令后 bReferenceSwitch 的状态为 ON->OFF->ON，在 OFF->ON 的上升沿回零完成，设置参考位置。

参考位置 =fHomePostion+（（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。

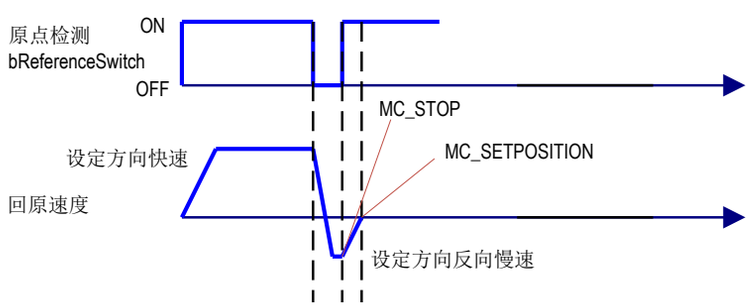
如果 bReturnToZero=TRUE，bReferenceSwitch 的状态在 OFF->ON 的上升沿将参考位置设置为 fHomePostion+（（fSignalDelay*1000+1 个 DC 时钟周期）/1000）*fVelocitySlow，然后按照速度 fVelocityFast 运行到 0 位置。

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：

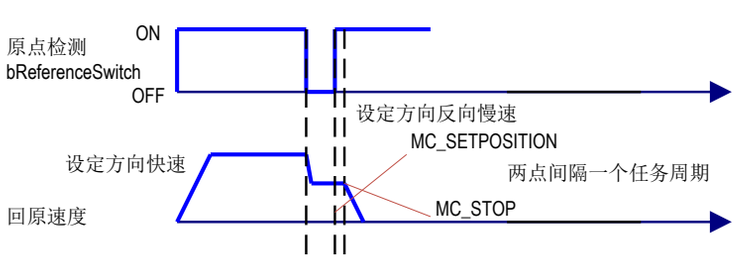
① 回零模式“0”时：



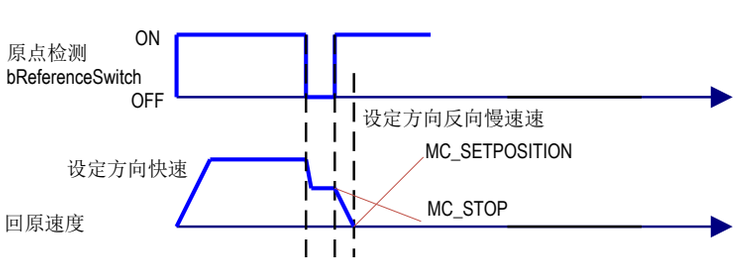
② 回零模式“1”时



③ 回零模式“4”时

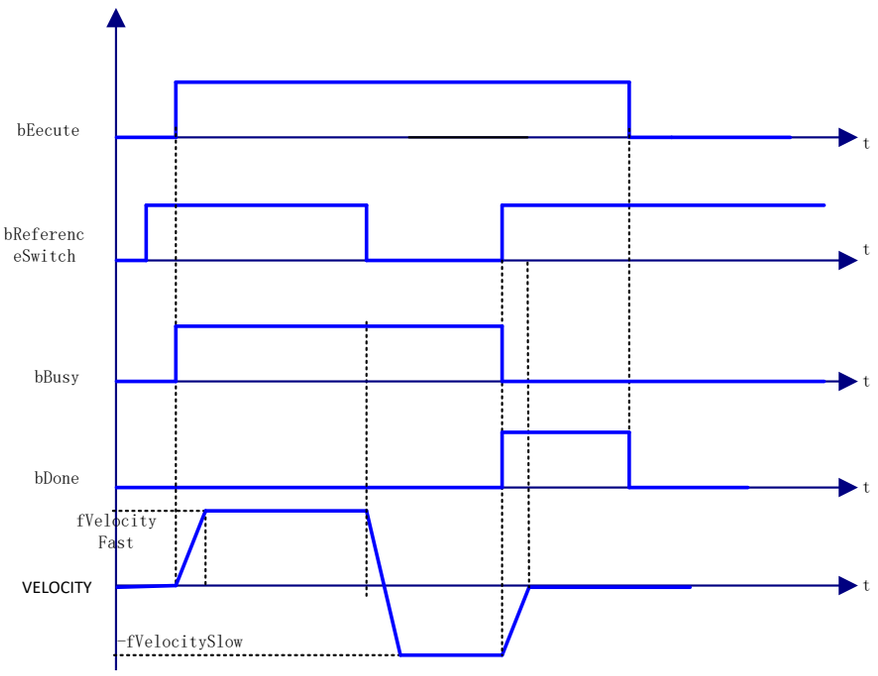


④ 回零模式“5”时

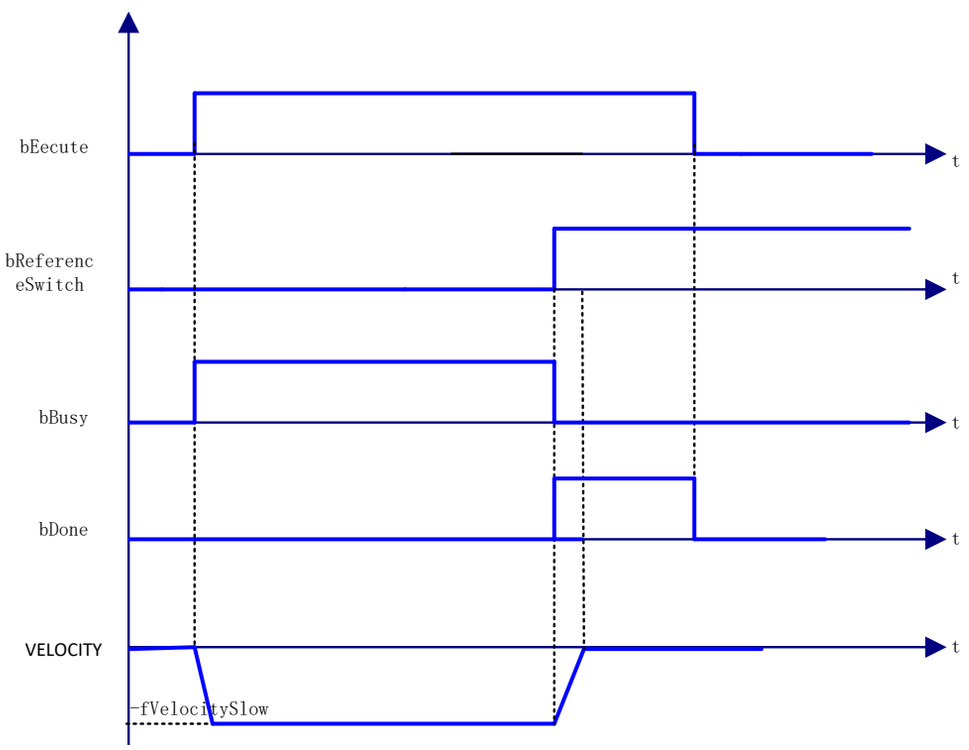


◆ 时序图

① 指令执行时 bReferenceSwitch TRUE 时



② 指令执行时 bReferenceSwitch FALSE 时



4） 错误说明

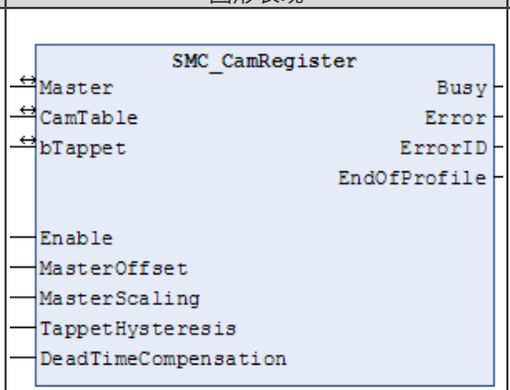
输入轴类型出错。
轴有错误。
轴没有使能
速度或加速度无效。
【注意】：请阅读“附录 C 错误代码说明”以了解相关错误代码说明。

6.2 轴组指令 (主 / 从轴指令)

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_ERROR	异常发生时，输出错误代码
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=> );
```

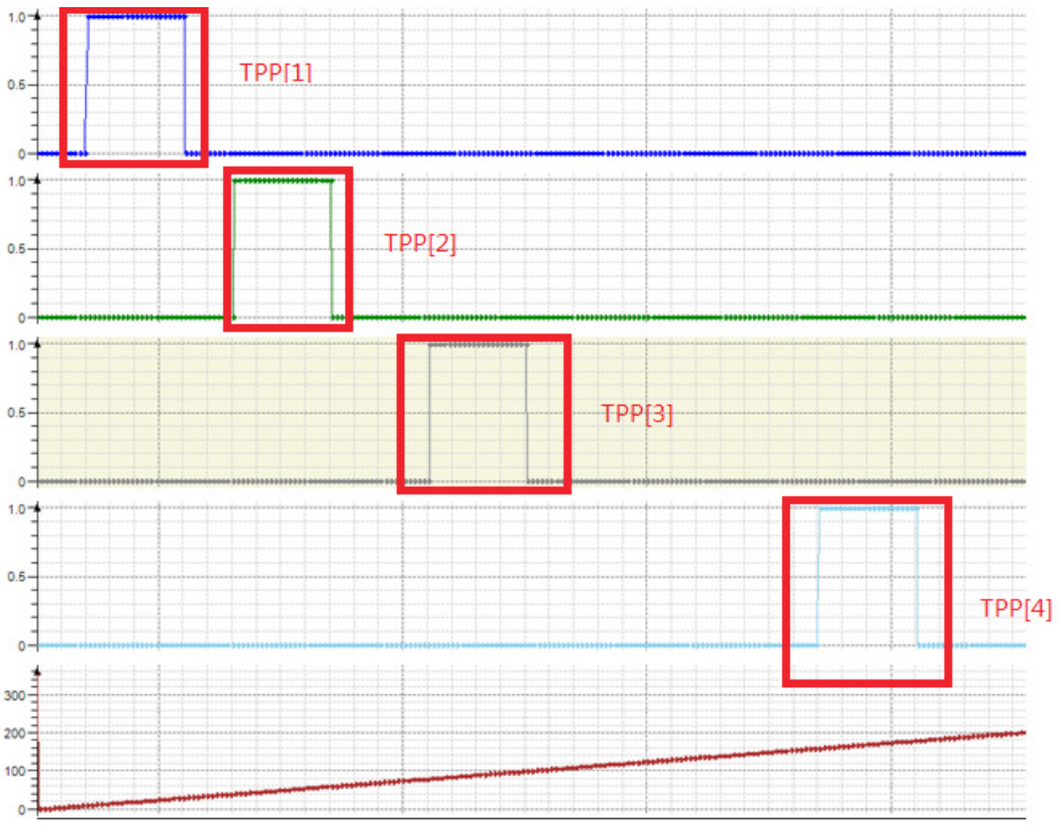
  Cam 编辑如下图：

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：

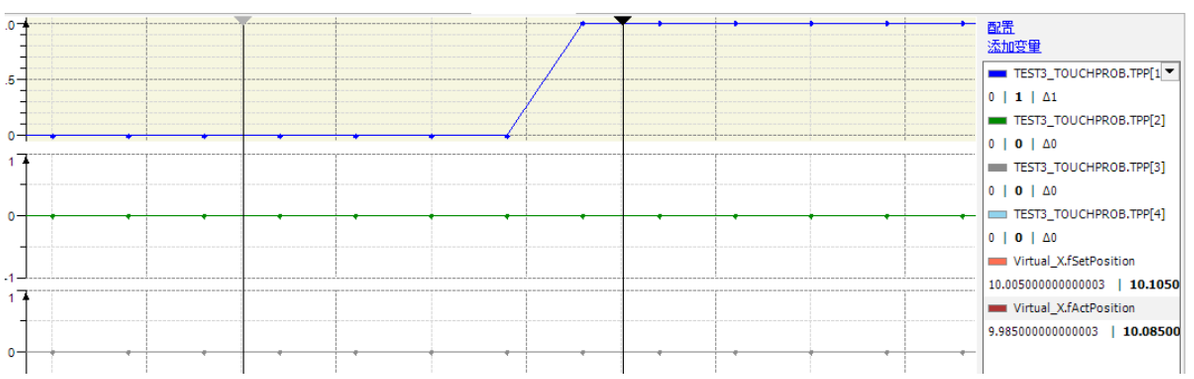
  监控曲线如下图：



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

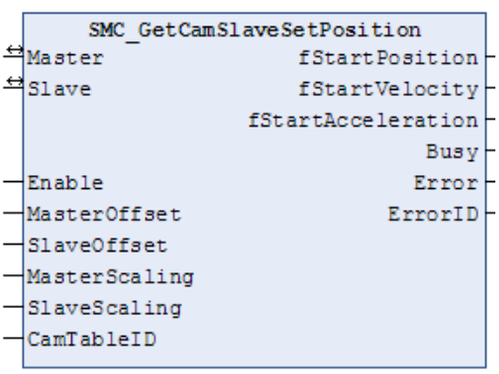
轴有错误、轴没有使能、偏移值或者标尺值设置超过主轴范围。

【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。

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	主轴缩放	LREAL		1	凸轮表主轴缩放因子
SlaveScaling	从轴缩放	LREAL		1	凸轮表从轴轴缩放因子
CamTableID	凸轮 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
fStartAcceleration	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=> );
```

Enable	BOOL	TRUE
MasterOffset	LREAL	100
SlaveOffset	LREAL	0
MasterScaling	LREAL	1
SlaveScaling	LREAL	1
CamTableID	MC_CAM_ID	
fStartPosition	LREAL	33.580246913580254

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。

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_GetTappetValue Tappets bTappet iID bInitValue bSetInitValueAtReset	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
bSetInitValueAtReset		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

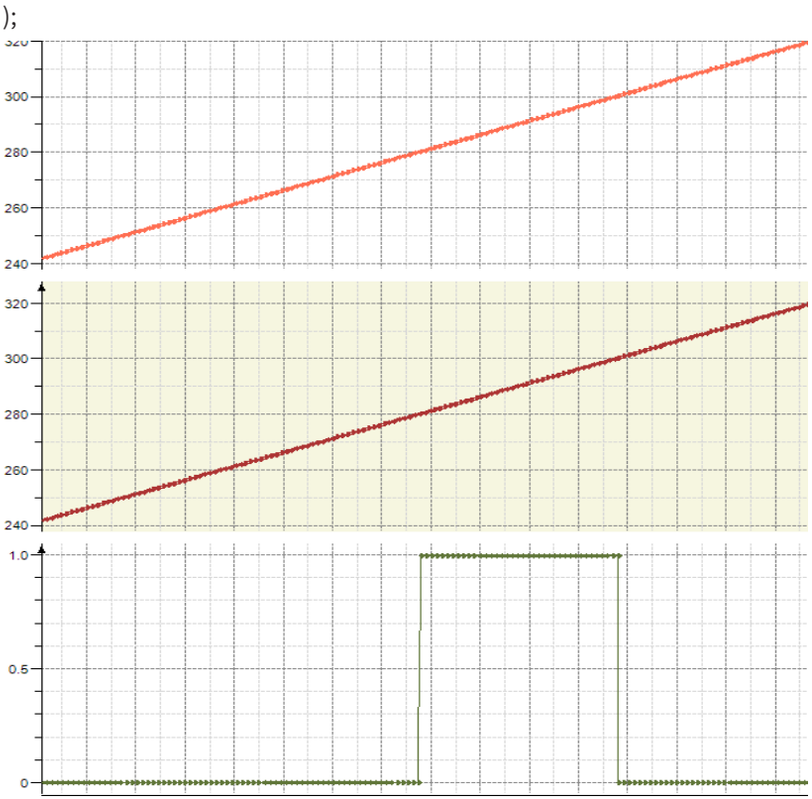
◆ 该功能块需要与 MC_CamIn 指令配合使用。

◆ 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 ;

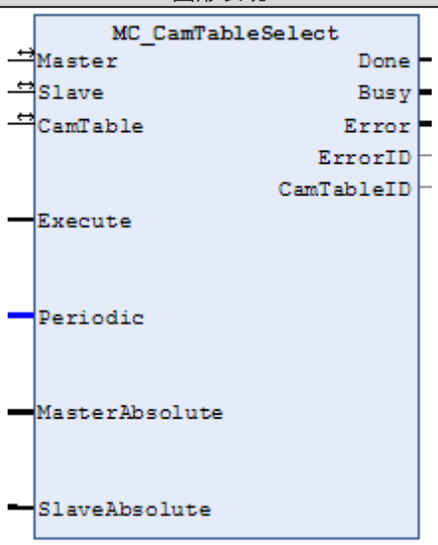
CamTable ID 没有指向。

【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。

MC_CamTableSelect

指定凸轮表，与 MC_CamIn 指令配合使用。

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。

◆ MasterAbsolute、SlaveAbsolute、Periodic 参数具体作用参考 MC_CamIn 指令详情

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。。

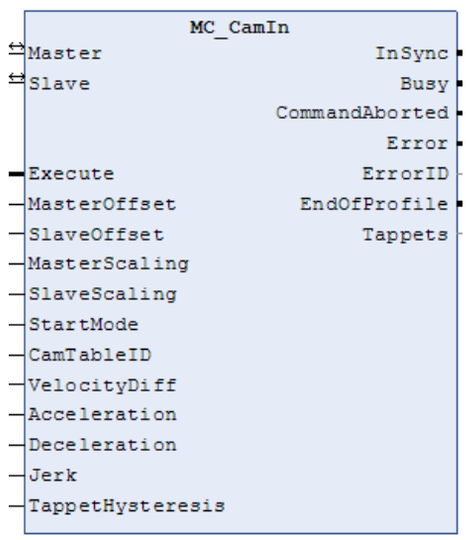
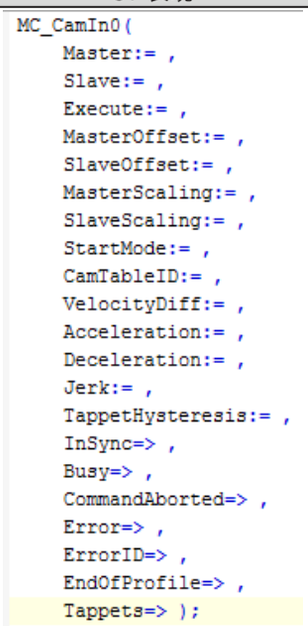
◆ CamTable 所对应的凸轮表编辑需要正确无误否则会错误输出。

【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。

MC_Camin

使用指定的凸轮表开始执行电子凸轮动作，可根据应用需求指定主从轴偏置值、缩放比和工作模式。

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_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	从轴偏执	LREAL	负数，正数，0	0	以指定的偏移值移动从轴的相位
MasterScaling	主轴预编译比例	LREAL	>0.0	1	以指定的比例放大 / 缩小主轴的相位
SlaveScaling	从轴预编译比例	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

输入变量	名称	数据类型	有效范围	初始值	描述
Deceleration		LREAL			ramp_in 时减速度
Jerk		LREAL			ramp_in 的加速度
TappetHysteresis		LREAL			挺杆的阻尼系数

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。
CommandAborted	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
EndOfProfile	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 Command-Aborted 输出为 TRUE。

4） 指令详情

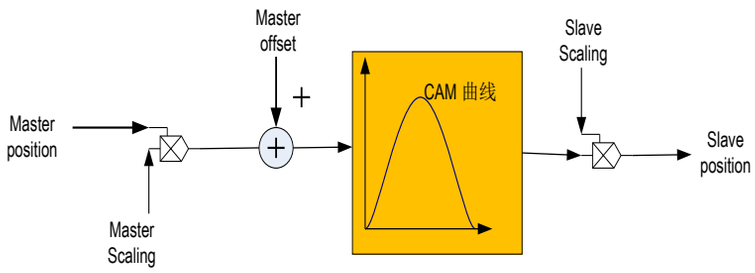
下面对指令详细说明：

◆ 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方法如下：



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。

注意：主轴位置、从轴位置均指的是凸轮函数曲线所需的主从轴位置，而不是主轴、从轴物理实轴的位置。

◆ 周期模式与 EndOfProfile 的关系：

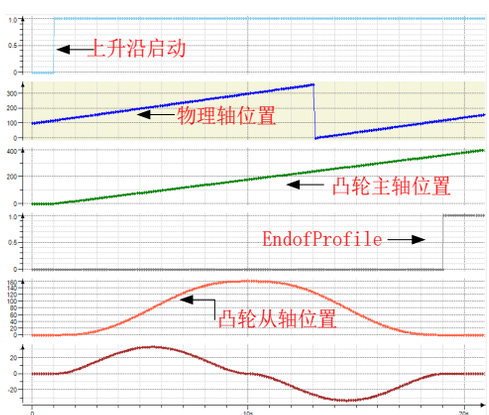
周期模式非周期模式决定了主轴到终止位置后电子凸轮是否要再次进行。

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。

① 周期模式：MC_CamTableSelect 指令 Periodic 选择 TRUE



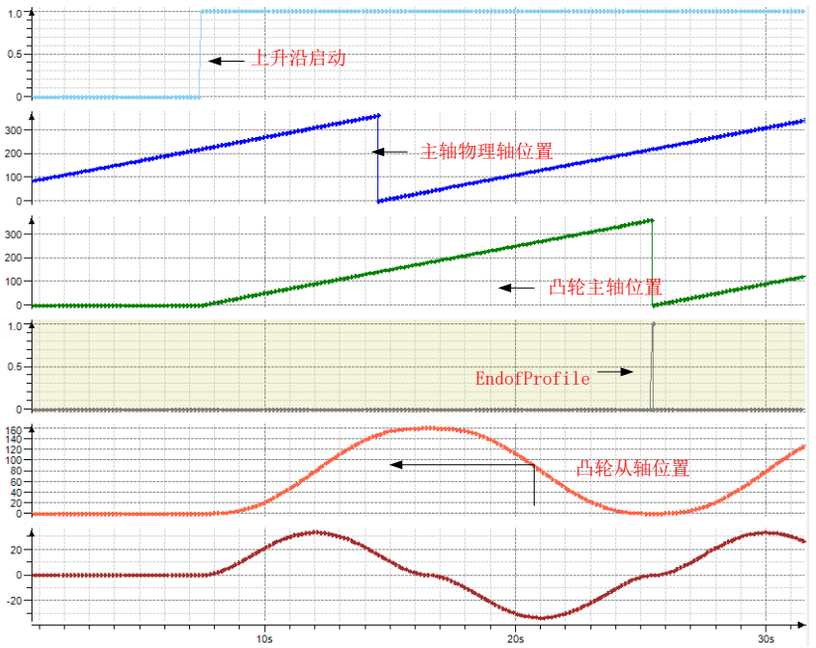
此时凸轮一个主轴周期完成后会接着执行下一个周期，且 EndofProfile 信号 TRUE 输出只维持一个任务周期。

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。



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：

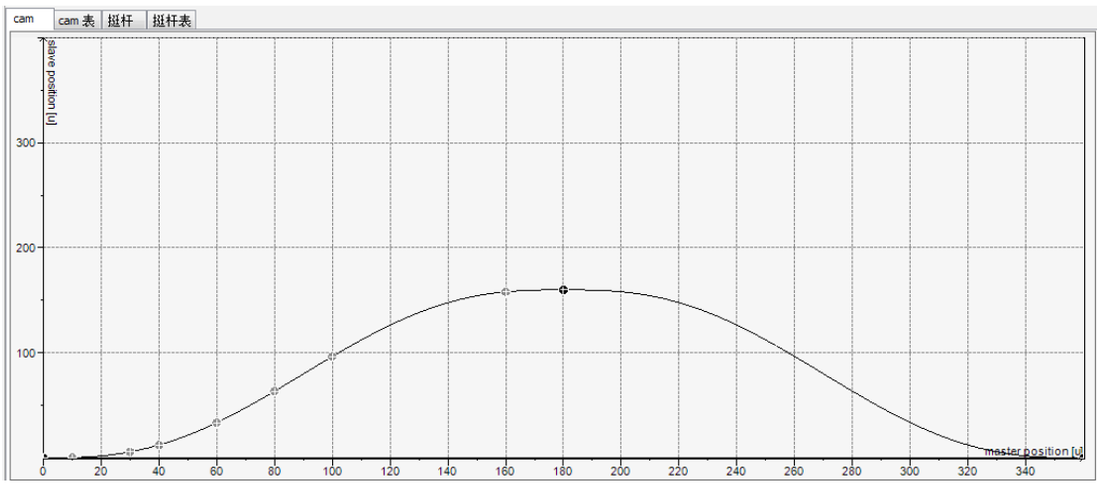
MC_CamTableSelect.MasterAbsolute	主轴模式
absolute	绝对模式
relative	相对模式

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

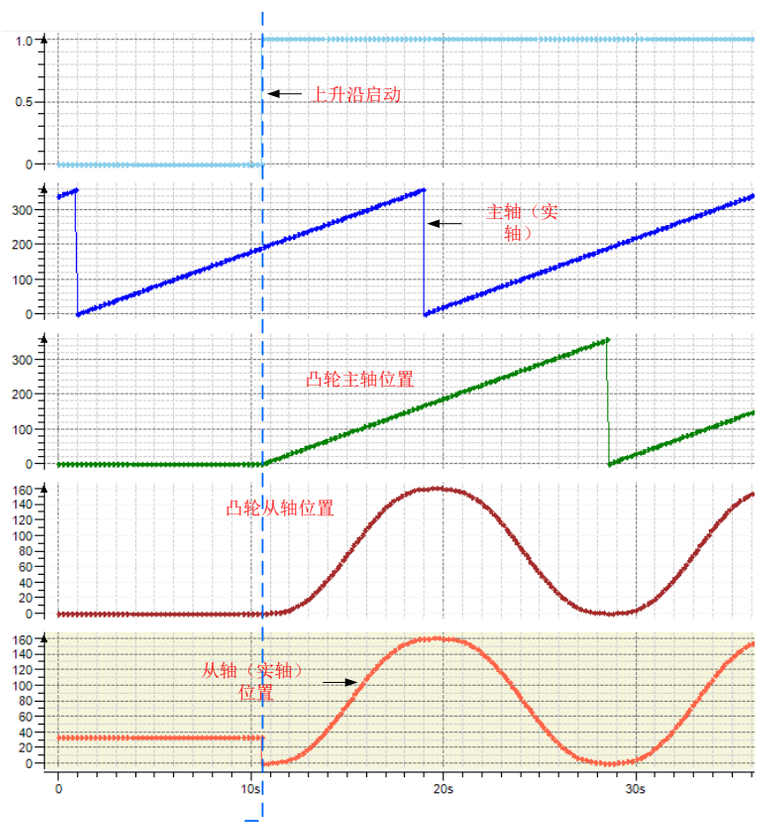
的凸轮表如下图所示：



② StartMode 为 0（绝对模式）

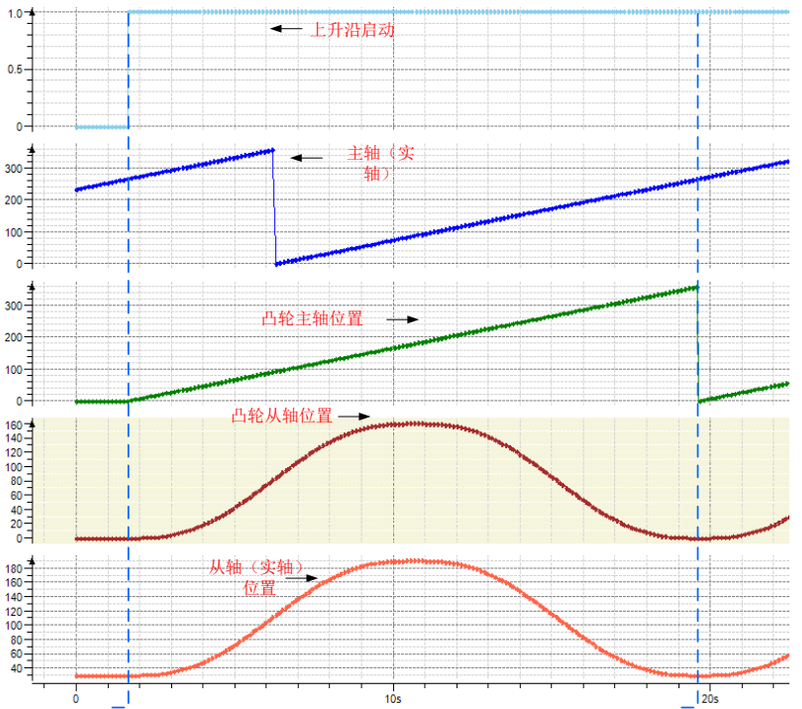
当 MC_CamTableSlect 指令 MasterAbsolute 设置为 FALSE，SlaveAbsolute 设置为 TRUE 时，此时主轴工作在相对模式，从轴工作在绝对模式。当 Excute 上升沿 , 凸轮启动时，凸轮主轴从凸轮表“起始位置”（0）开始，凸轮从轴按照上述的“凸轮表齿合公式”计算输出，从轴实轴指令位置等于齿合计算输出值。比如说凸轮从轴起始位置为 0，凸轮启动时从轴实轴位置为 20，则启动开始从轴实轴的位置指令为 0 产生跳变。

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。



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。



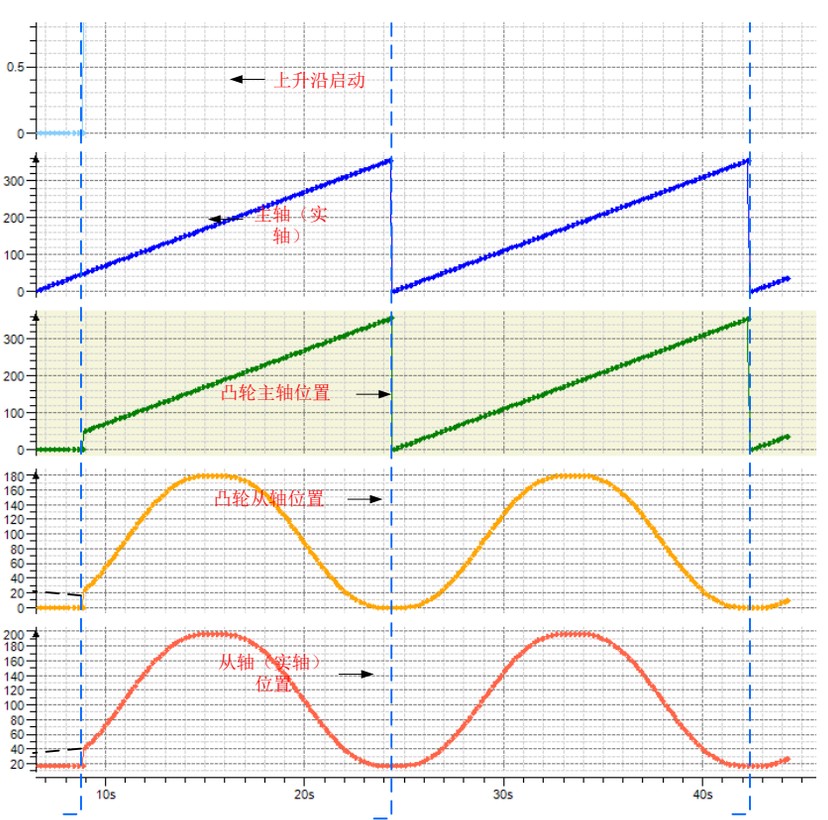
5） 错误说明

- 指令设置信息与 Camslect 指令设置信息不相符合。
- 周没有使能情况。

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：1 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。

2 主轴位置应在凸轮主轴位置范围内

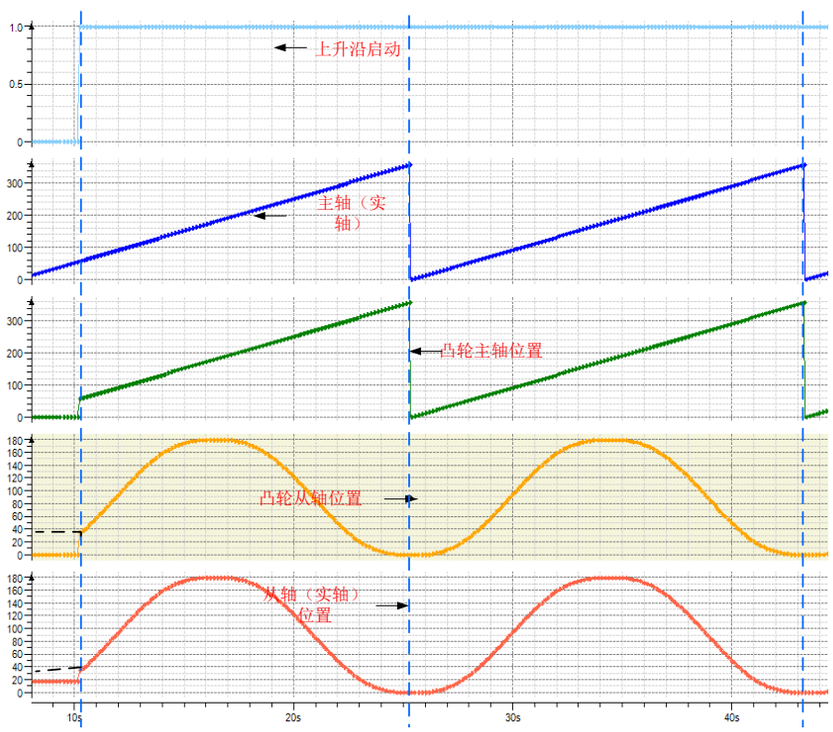


当 MC_CamTableSlect 指令 MasterAbsolute 设置为 TRUE，SlaveAbsolute 设置为 TRUE 时，此时主轴工作在绝对模式，从轴工作在绝对模式。当 Excute 上升沿 , 凸轮启动时，凸轮主轴从当前“主轴实轴位置开始”，从轴实轴位置指令 = 凸轮表齿合计算值（凸轮从轴位置）。

注意：

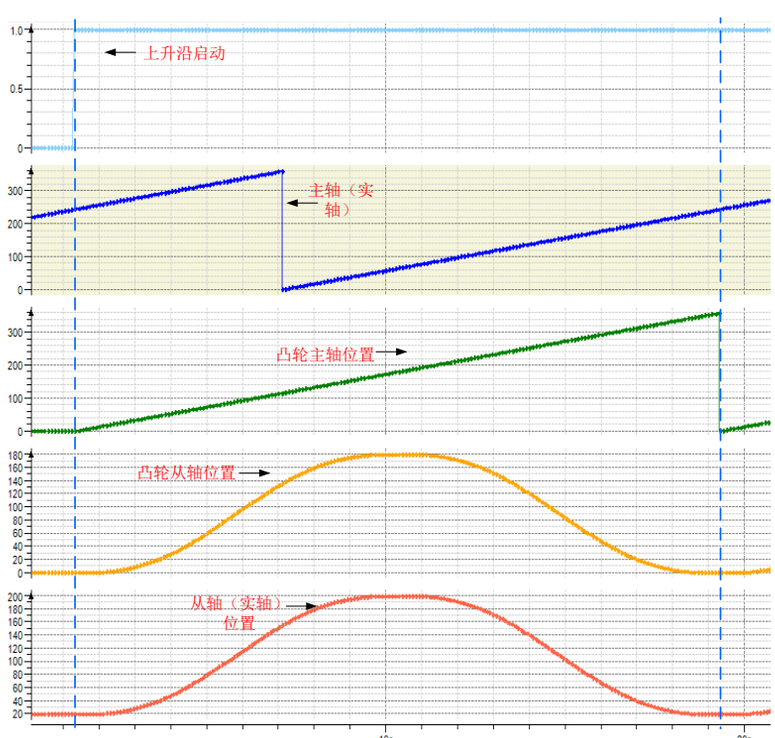
1 如果该情况下主轴（实轴）开始位置不在凸轮主轴开始位置、从轴位置不在凸轮从轴开始位置则会产生跳变。

2 主轴位置应在凸轮主轴位置范围内



2） StartMode 为 1（相对模式）

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）。

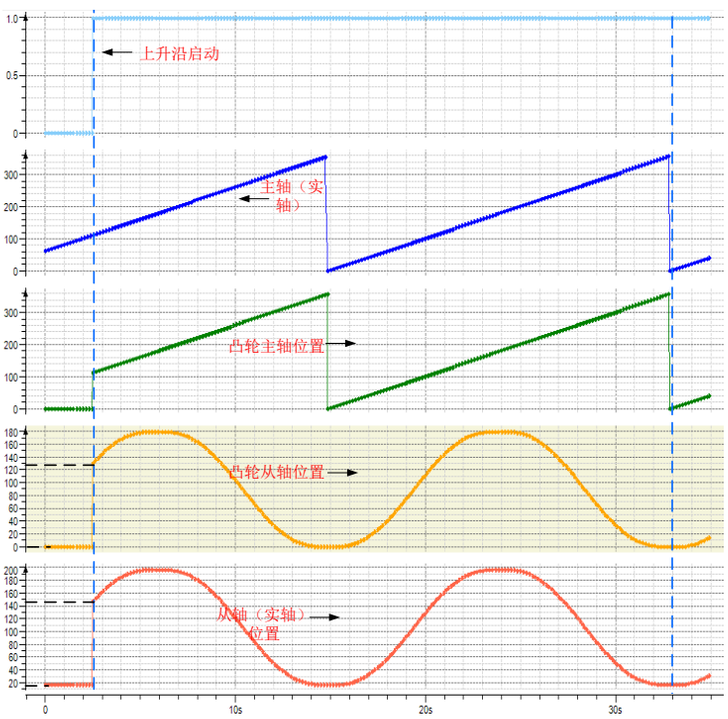


当 MC_CamTableSlect 指令 MasterAbsolute 设置为 TRUE，SlaveAbsolute 设置为 TRUE 或者为 False 时，此时主轴工作在绝对模式，从轴工作在相对模式。当 Excute 上升沿 , 凸轮启动时，凸轮主轴从 “主轴当前位置开始”，从轴实轴位置指令 = 启动时从轴位置 + 凸轮表齿合计算值（凸轮从轴位置）。

注意：

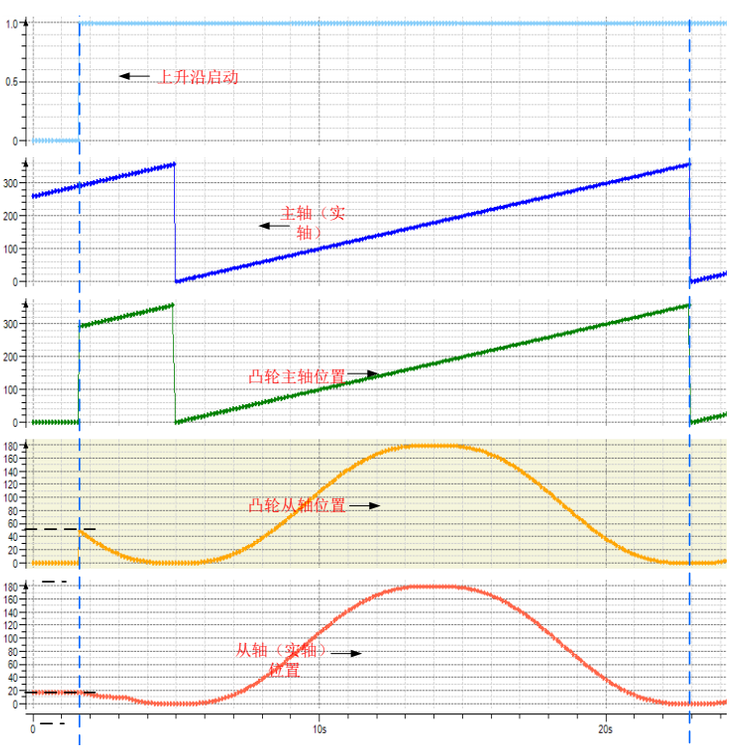
1 如果该情况下主轴（实轴）开始位置不在凸轮主轴开始位置则会产生跳变。

2 主轴位置应在凸轮主轴位置范围内

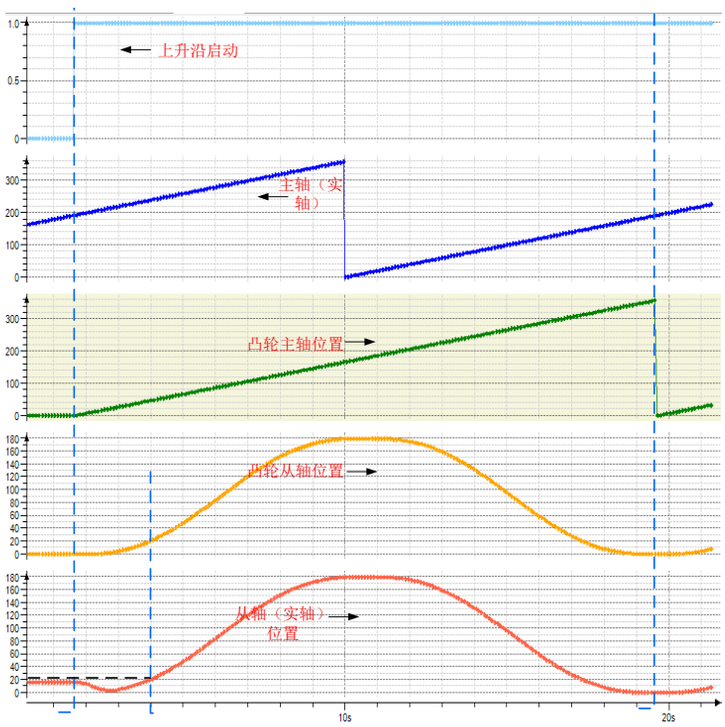


3） StartMode 为 2（rampin 斜坡切入模式）

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)。

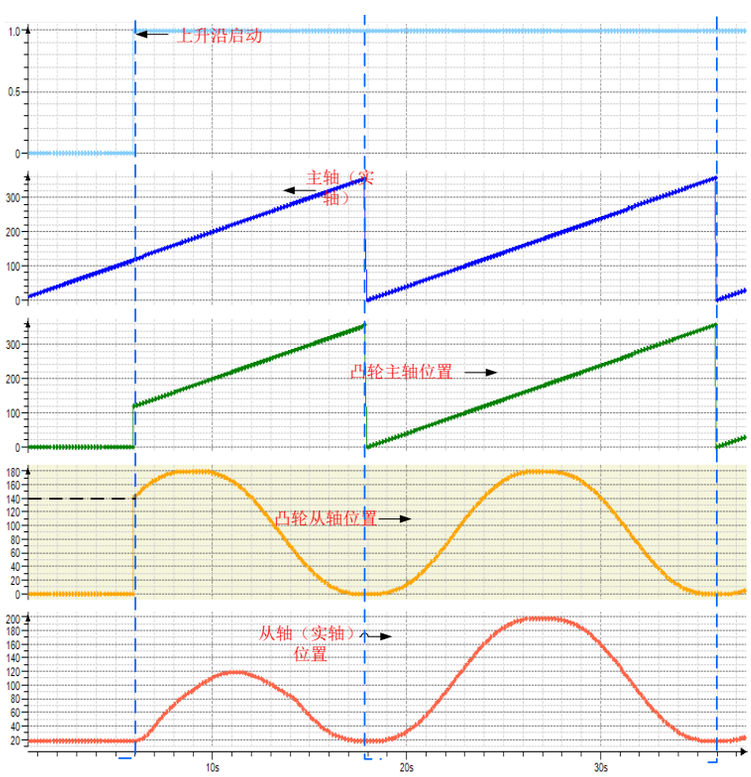


当 MC_CamTableSlect 指令 MasterAbsolute 设置为 FALSE，SlaveAbsolute 设置为 TRUE 时，此时主轴工作在相对模式，从轴工作在绝对模式。当 Excute 上升沿 , 凸轮启动时，凸轮主轴从“凸轮主轴起始位置开始”，从轴通过设定的 VelocityDiff、Acceleration、Deceleration 增加一个补偿运动避免切入时的潜在跳变。从轴实轴位置指令 = 凸轮表齿合计算值（凸轮从轴位置）+f(VelocityDiff,Acceleration,Deceleration)。



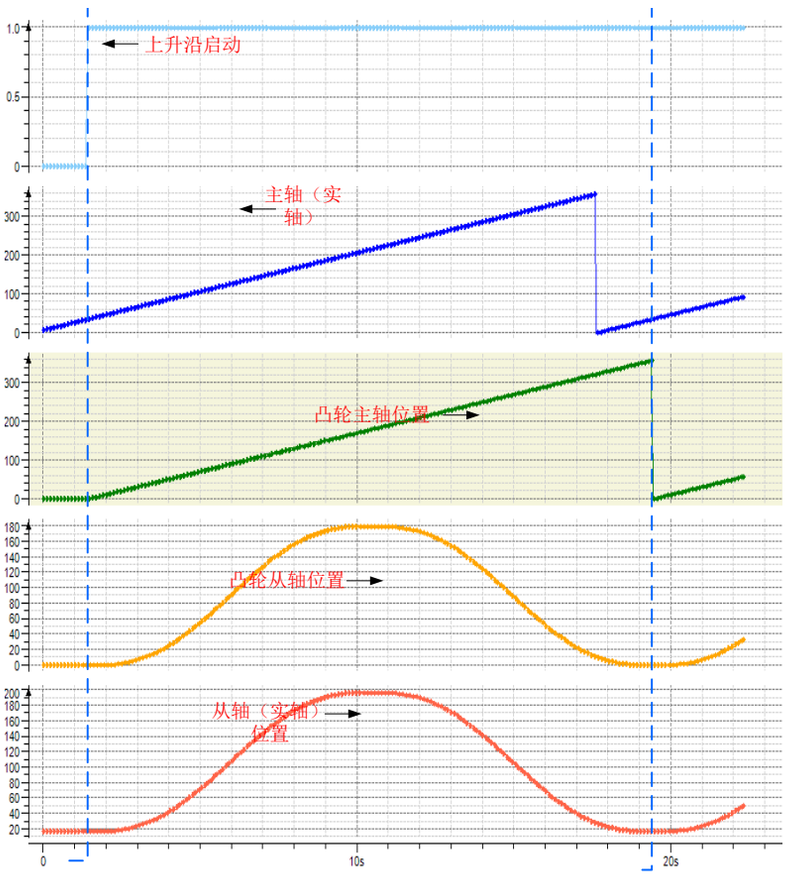
当 MC_CamTableSlect 指令 MasterAbsolute 设置为 TRUE，SlaveAbsolute 设置为 FALSE 时，此时主轴工作在绝对模式，从轴工作在相对模式。当 Excute 上升沿 , 凸轮启动时，凸轮主轴从 “主轴当前位置开始”，从轴通过设定的 VelocityDiff、Acceleration、Deceleration 增加一个补偿运动避免切入时的潜在跳变。从轴实轴位置指令 = 从轴当前位置 + 凸轮表齿合计算值（凸轮从轴位置）+f(VelocityDiff,Acceleration,Deceleration)。

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



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



4） 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。

缩放比、主从轴偏移：

参照上述凸轮齿合计算公式可知：输入量 MasterOffset 和 MasterScaling 按照下面公式变换主轴位置，并且电子凸轮会用变换后的位置 X 进行计算：

X = MasterScaling*MasterPosition + MasterOffset

因此，如果 MasterScaling 的值大于 1，电子凸轮将在较高速度下运行；如果值小于 1，电子凸轮将在较低速度运行。

SlaveOffset 参数会使电子凸轮纵向（从轴方向）移动，SlaveScaling 参数则会在从

轴方向拉伸电子凸轮。根据下述公式说明可知，第一步电子凸轮先进行拉伸动作，

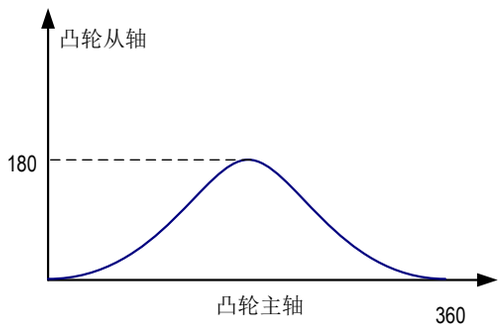
然后再进行移动：

Y = SlaveScaling*CAM(X) + SlaveOffset

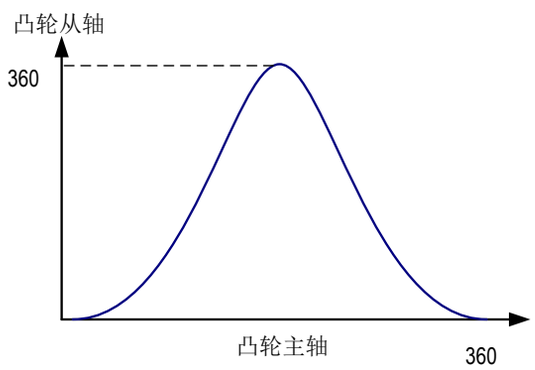
A SlaveScaling >1 ，拉伸电子凸轮，从轴运动范围增大；同理，SlaveScaling<1，

缩小电子凸轮从轴运动范围。

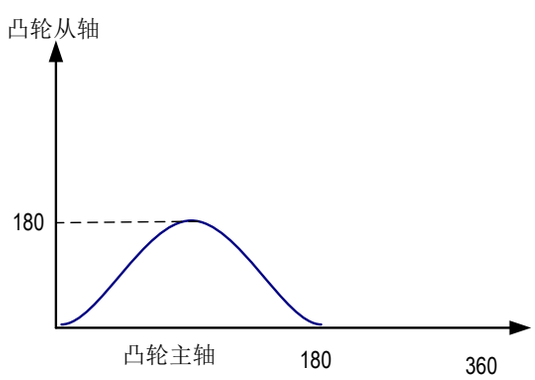
当 MasterScaling=1.0、SlaveScaling=1.0、MasterOffset=0、SlaveOffset=0，此时凸轮曲线为规划的凸轮曲线如下图所示：



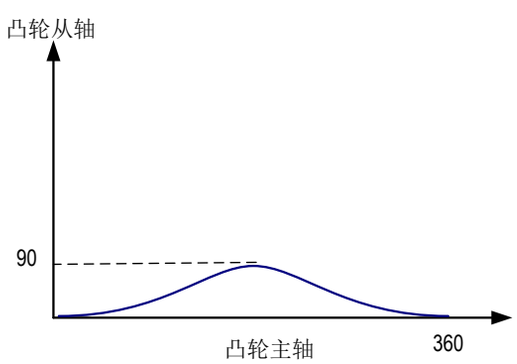
当 MasterScaling=1.0、SlaveScaling=2.0、MasterOffset=0、SlaveOffset=0，此时凸轮曲线如下图所示：



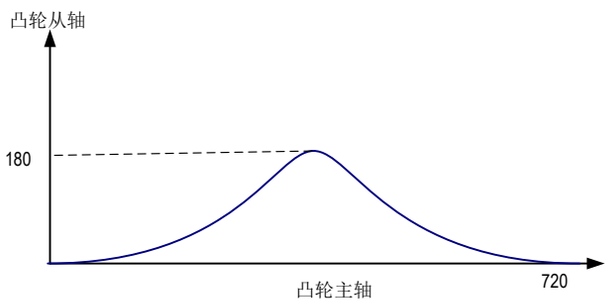
当 MasterScaling=2.0、SlaveScaling=1.0、MasterOffset=0、SlaveOffset=0，此时凸轮曲线如下图所示：



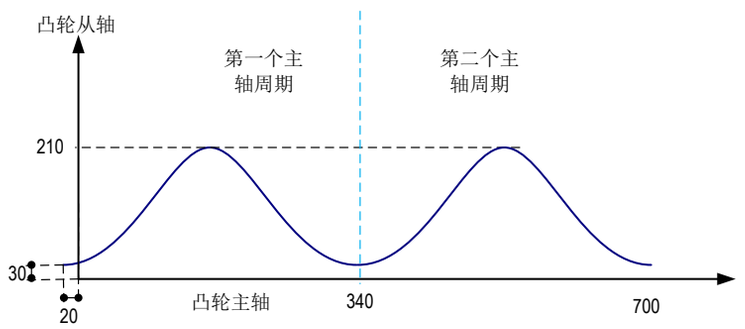
当 MasterScaling=1.0、SlaveScaling=0.5、MasterOffset=0、SlaveOffset=0，此时凸轮曲线如下图所示：



当 MasterScaling=0.5、SlaveScaling=1、MasterOffset=0、SlaveOffset=0，此时凸轮曲线如下图所示：



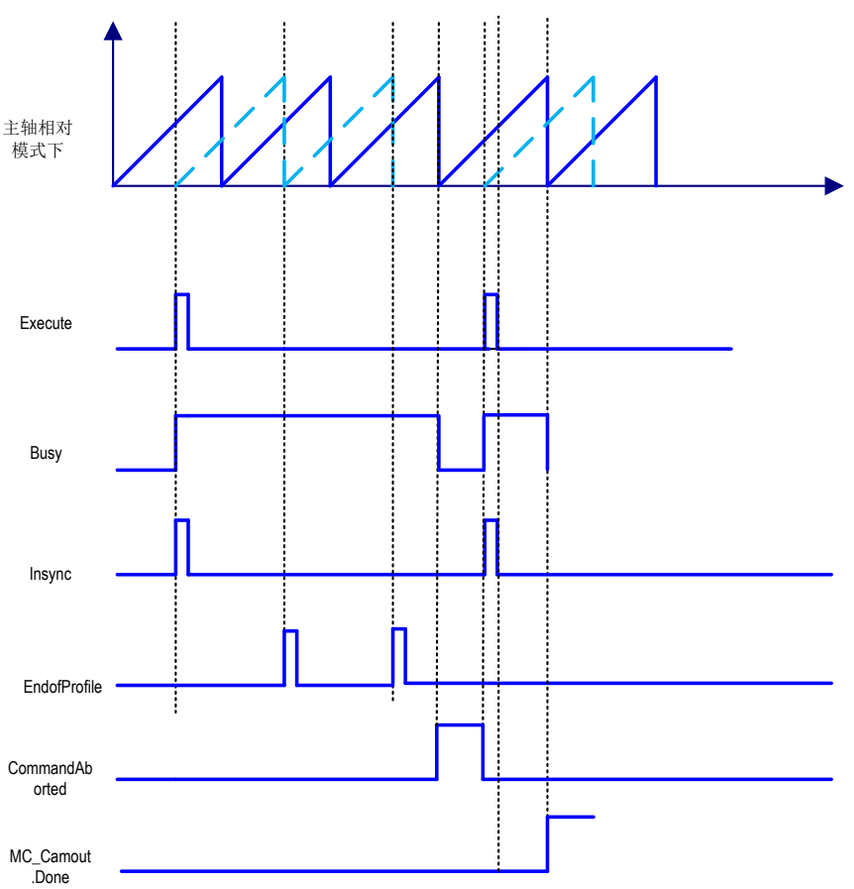
当 MasterScaling=1、SlaveScaling=1、MasterOffset=20、SlaveOffset=30，此时凸轮曲线如下图所示：



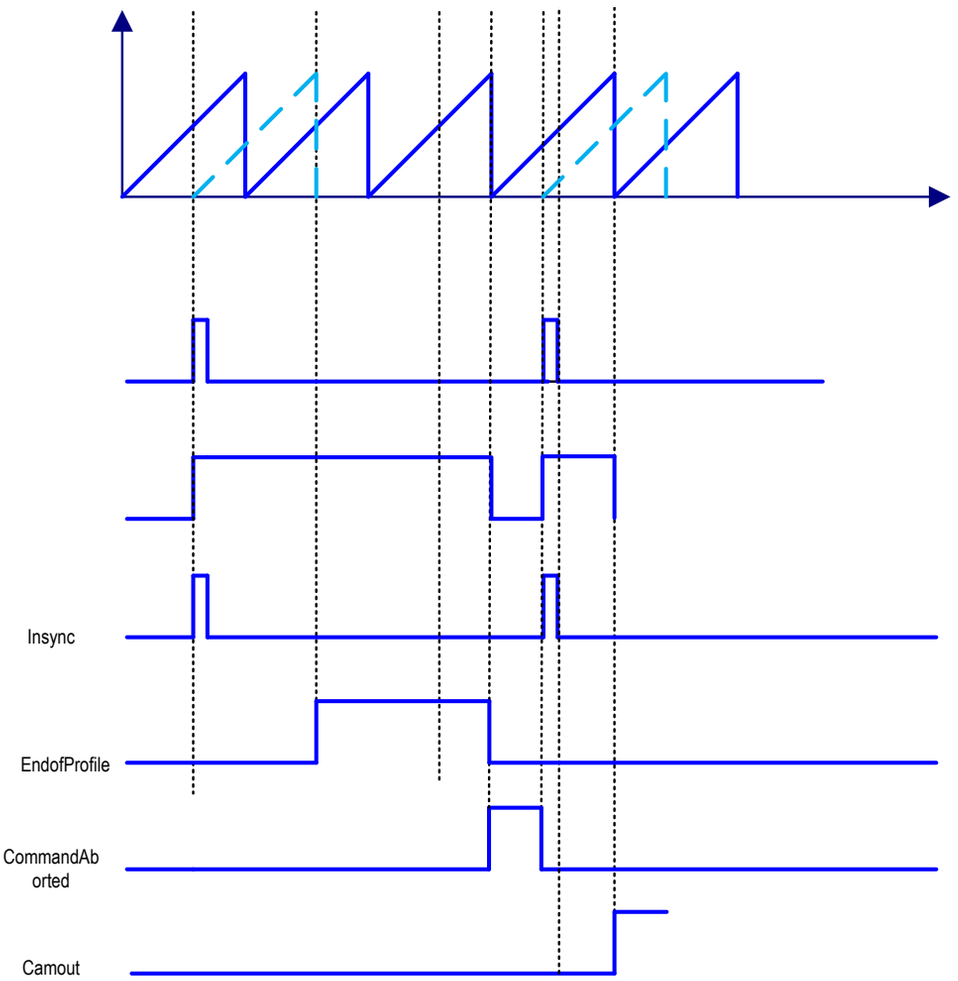
6） 时序图：

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：



◆ 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。

注：电子凸轮切换请参考运动控制功能篇。
电子凸轮样例参考运动控制功能篇：
挺杆功能请参考运动控制功能篇

7） 错误说明

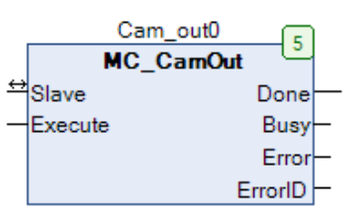
启动本指令检测到异常时，Error(错误) 变为 TRUE。

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。

MC_CamOut

断开从轴的凸轮耦合关系。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

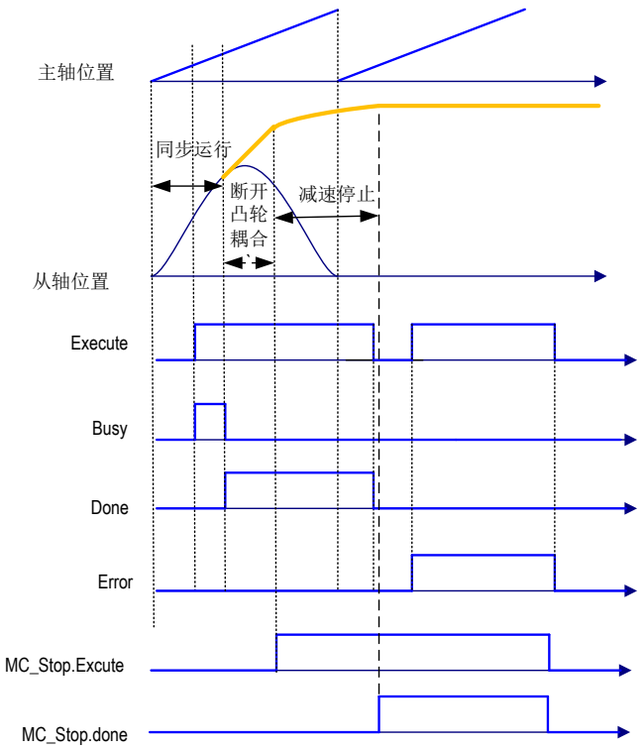
◆ 输出

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，ERROR 输出。

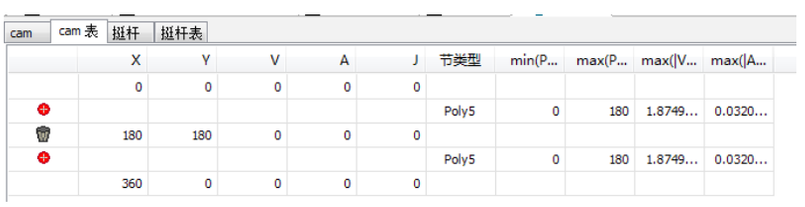
4） 时序图



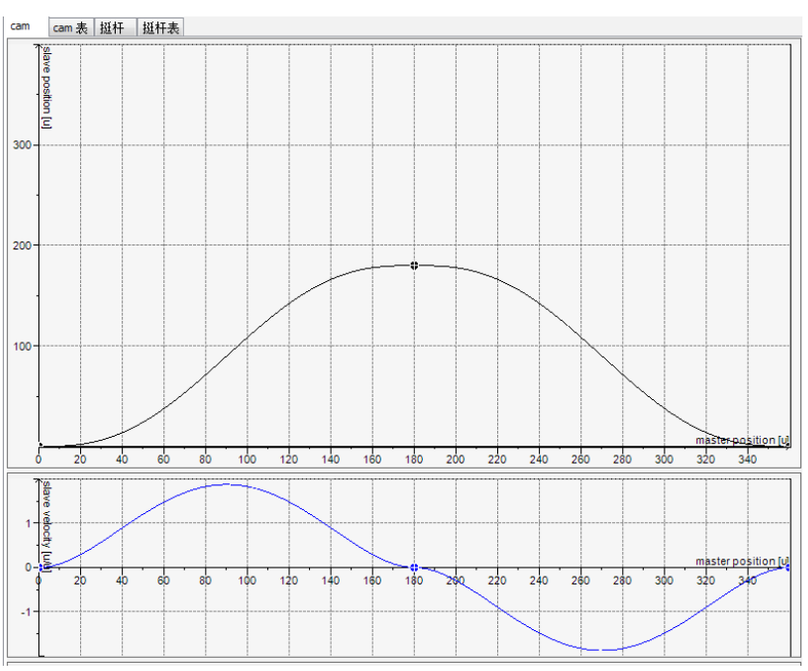
5） 使用范例

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

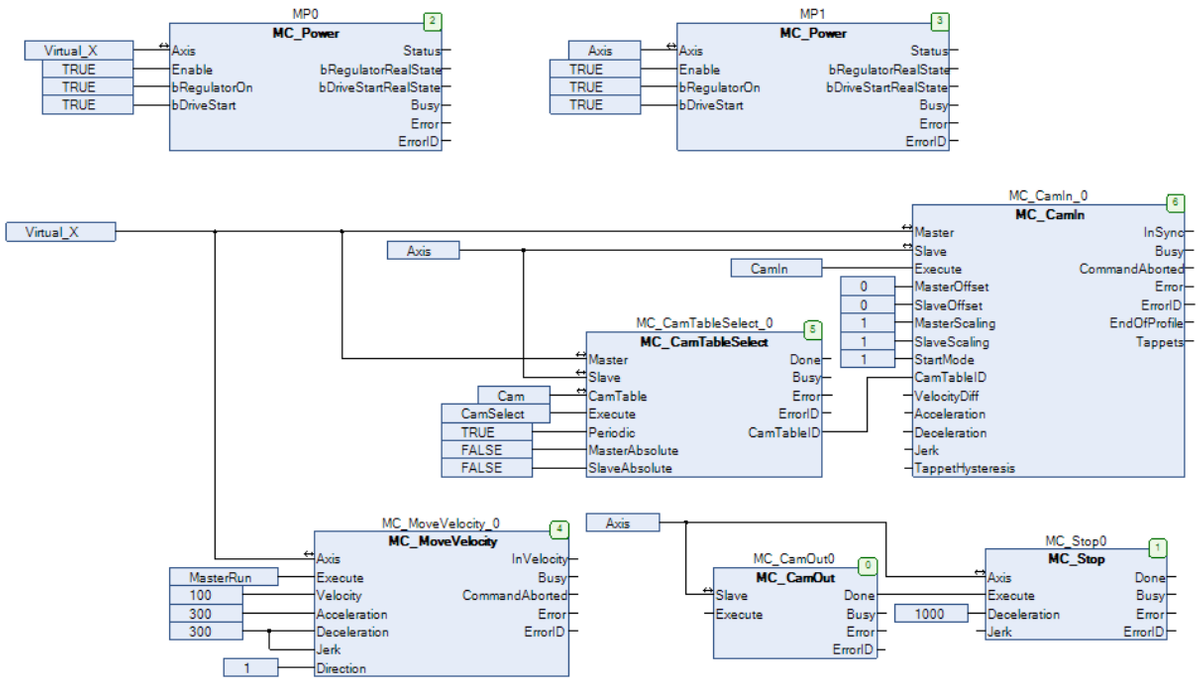
凸轮编辑器建立如下凸轮表（cam）：



	X	Y	V	A	J	节类型	min(P...	max(P...	max(|V...	max(|A...
	0	0	0	0	0					
						Poly5	0	180	1.8749...	0.0320...
	180	180	0	0	0					
						Poly5	0	180	1.8749...	0.0320...
	360	0	0	0	0					



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。

注意：

凸轮表在线修改等请参考运动控制功能篇

6） 错误说明

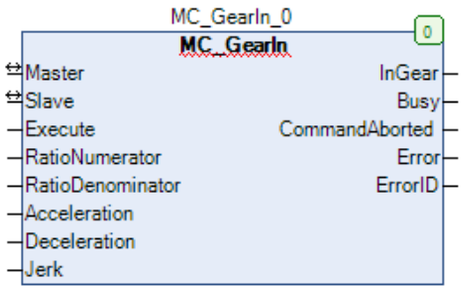
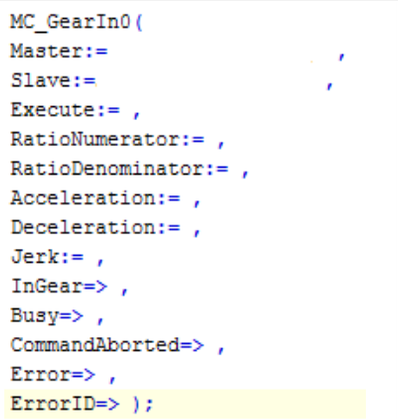
启动本指令发生异常时，Error 输出为 True。

可查看 ERRORID，参考“附录 C 错误代码说明”以了解 SMC_ERROR 错误代码。

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	-FAKSE	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
RatioDenominator	齿轮比分母	UDINT	正数	1	齿轮比分母
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
CommandAborted	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。

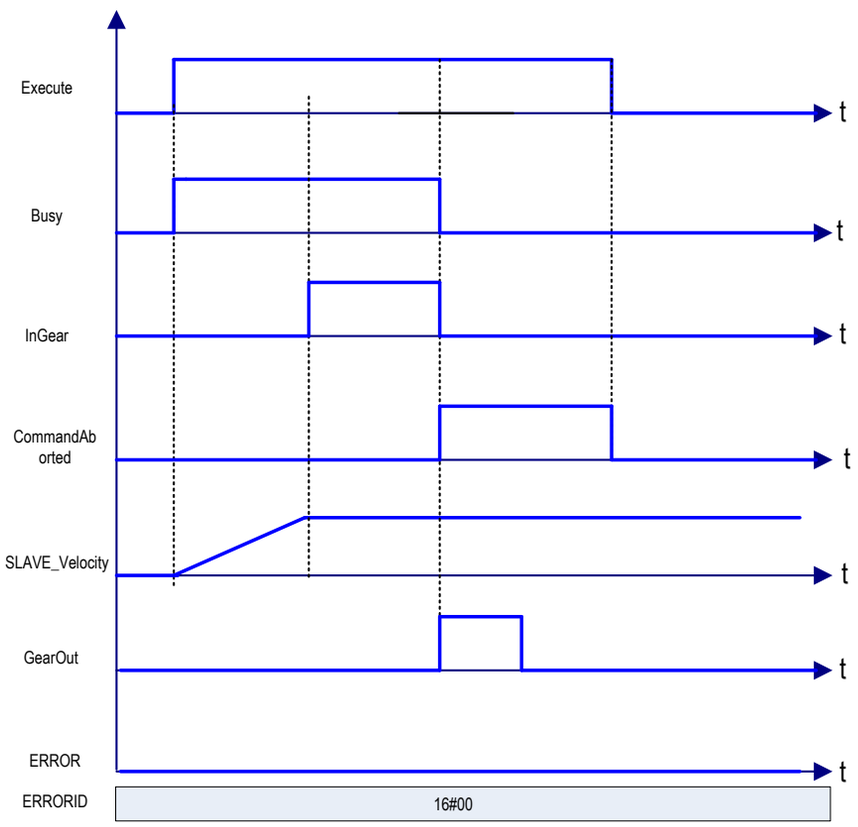
指令执行中 Busy 信号为 TRUE 时，如果从轴目标速度达到此时 Execute 新的上升沿不会影响它。

到达目标速度，InGear 为 TRUE，此后从轴移动量 = 主轴移动量 * RatioNumerator/ RatioDenominator。

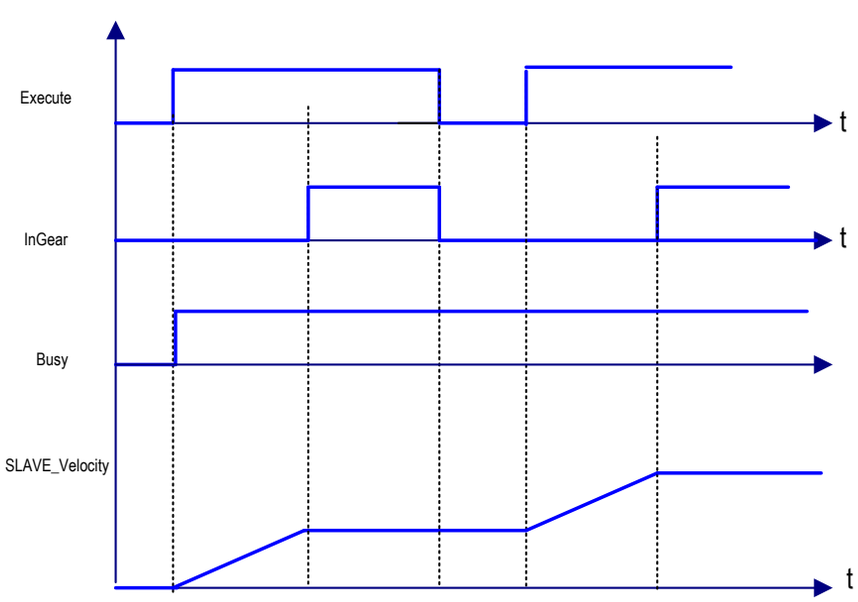
如果主轴速度实时变化的，情况下请注意慎重使用该指令。

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

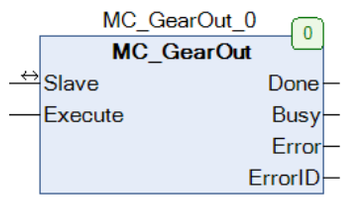
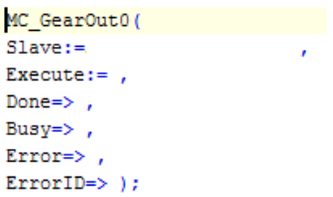
启动指令，ERROR 为 TRUE, 则有错误输出。

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。

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

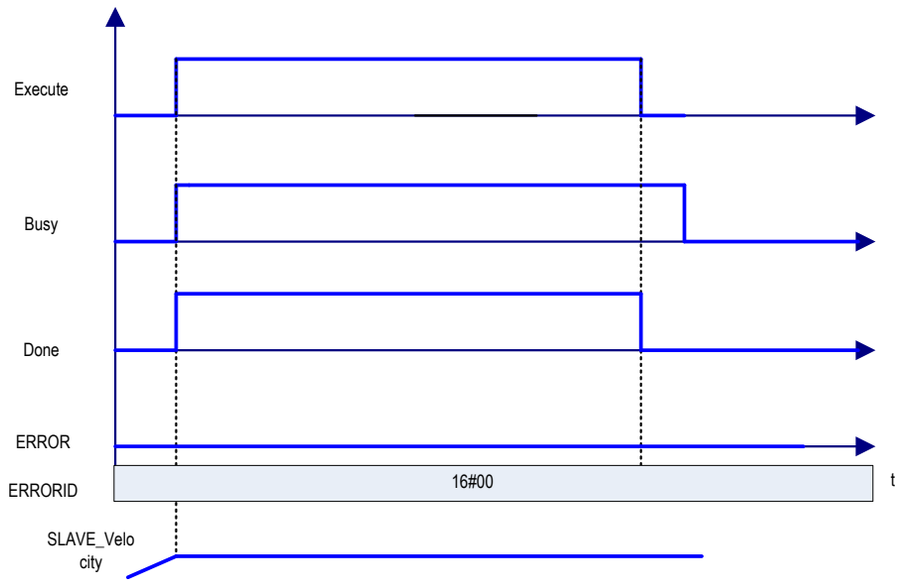
Execute 上升沿 , 执行切出电子齿轮动作。

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 FALSE.

MC_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。

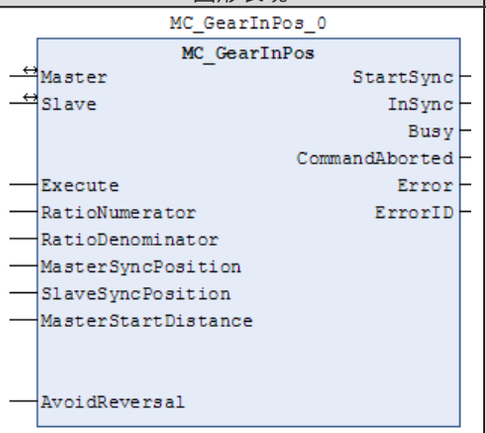
【注意】：请阅读“附录 C 错误代码说明”以了解相关错误代码说明。

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
RatioNumerator	齿轮比分子	DINT	-	1-	主从轴速度比的分子
RatioDenominator	齿轮比分母	DINT		1	主从轴速度比的分母
MasterSyncPosition	主轴同步位置	LREAL			主从轴齿轮比耦合时主轴位置
SlaveSyncPosition	从轴同步位置	LREAL			主从轴齿轮比耦合时从轴位置
MasterStartDistance	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

输出变量	名称	数据类型	有效范围	初始值	描述
Busy	指令处理中	BOOL	TRUE,FALSE	FALSE	True，指令正在处理中
CommandAborted	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。

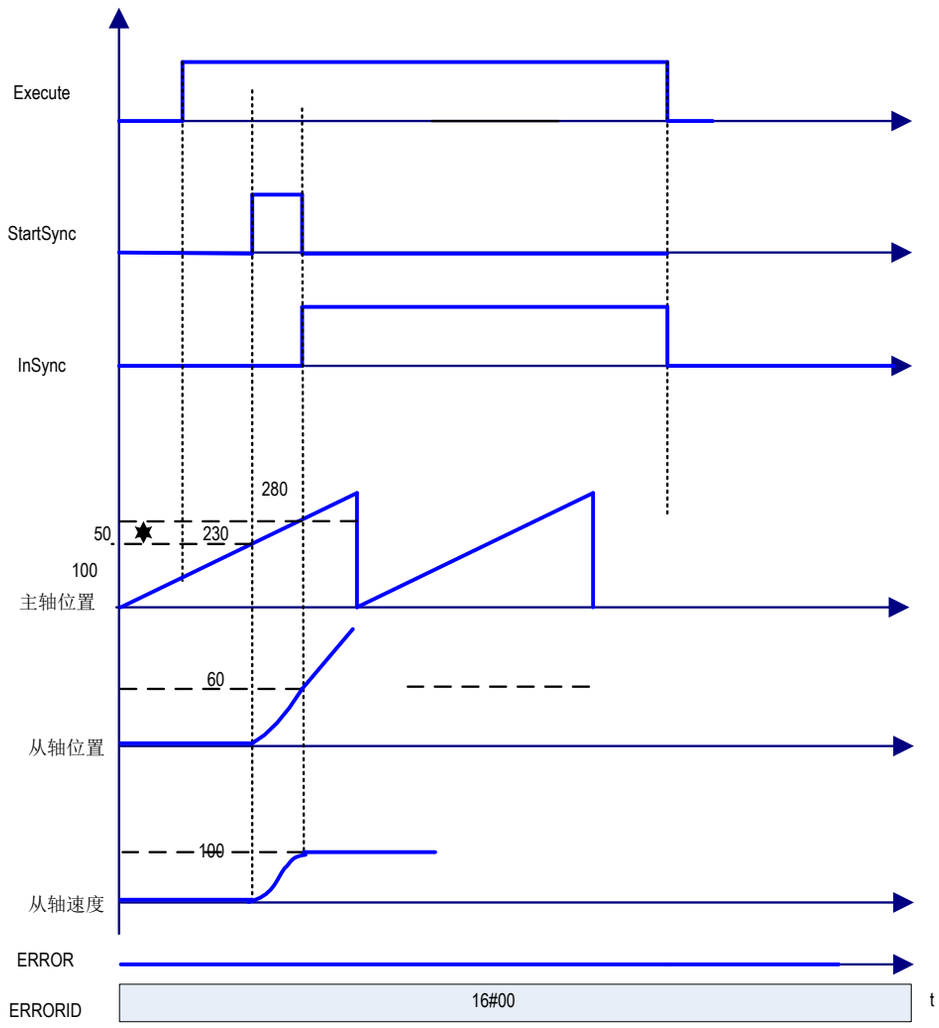
注意如果主从轴工作在线性模式需保证需保证上述几个参数设置合理否则齿轮动作无法正确进行，故而建议使用该指令时主从轴工作在周期模式。

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 >MasterSyncPosition-MasterStartDistance，或者从轴位置 > SlaveSyncPosition 则无法切入电子齿轮动作。

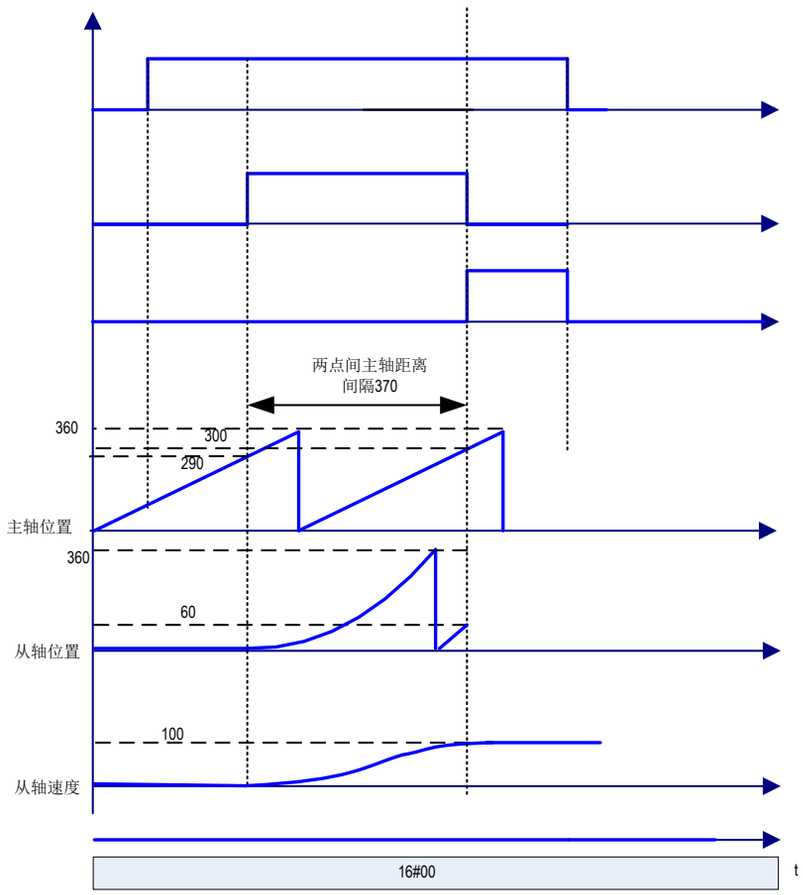
下面给出了几种不同参数下的样例时序图：

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）：

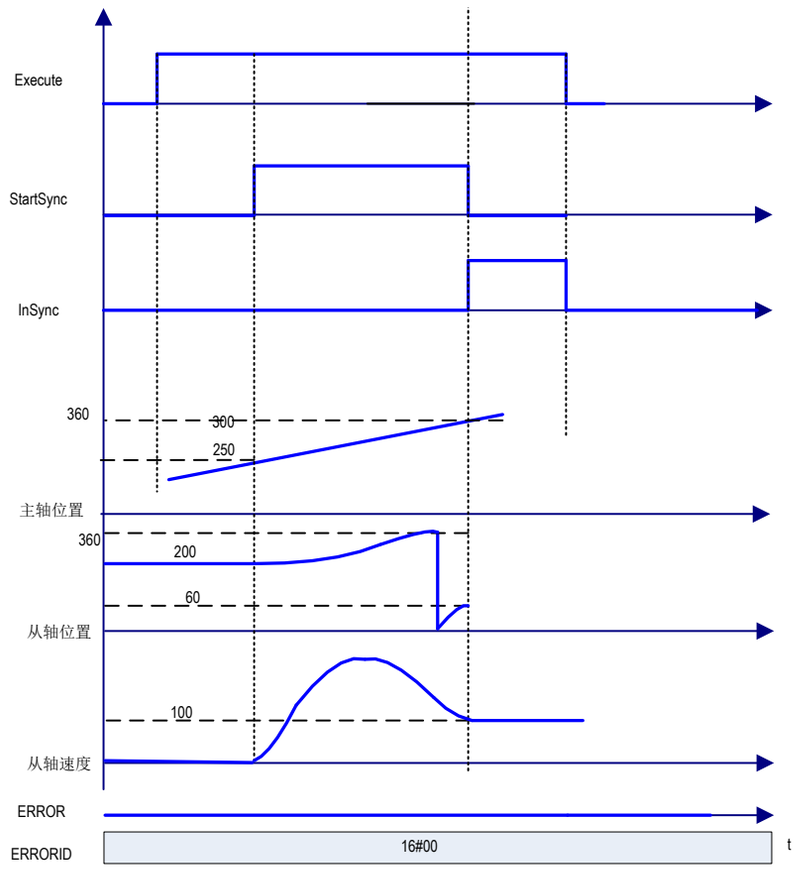
① MasterSyncPosition=280、MasterStartDistance=50、SlaveSyncPosition=60，主轴速度为 50、AvoidReversal=FALSE



② MasterSyncPosition=300、MasterStartDistance=370、SlaveSyncPosition=60，主轴速度为 50、AvoidReversal=FALSE



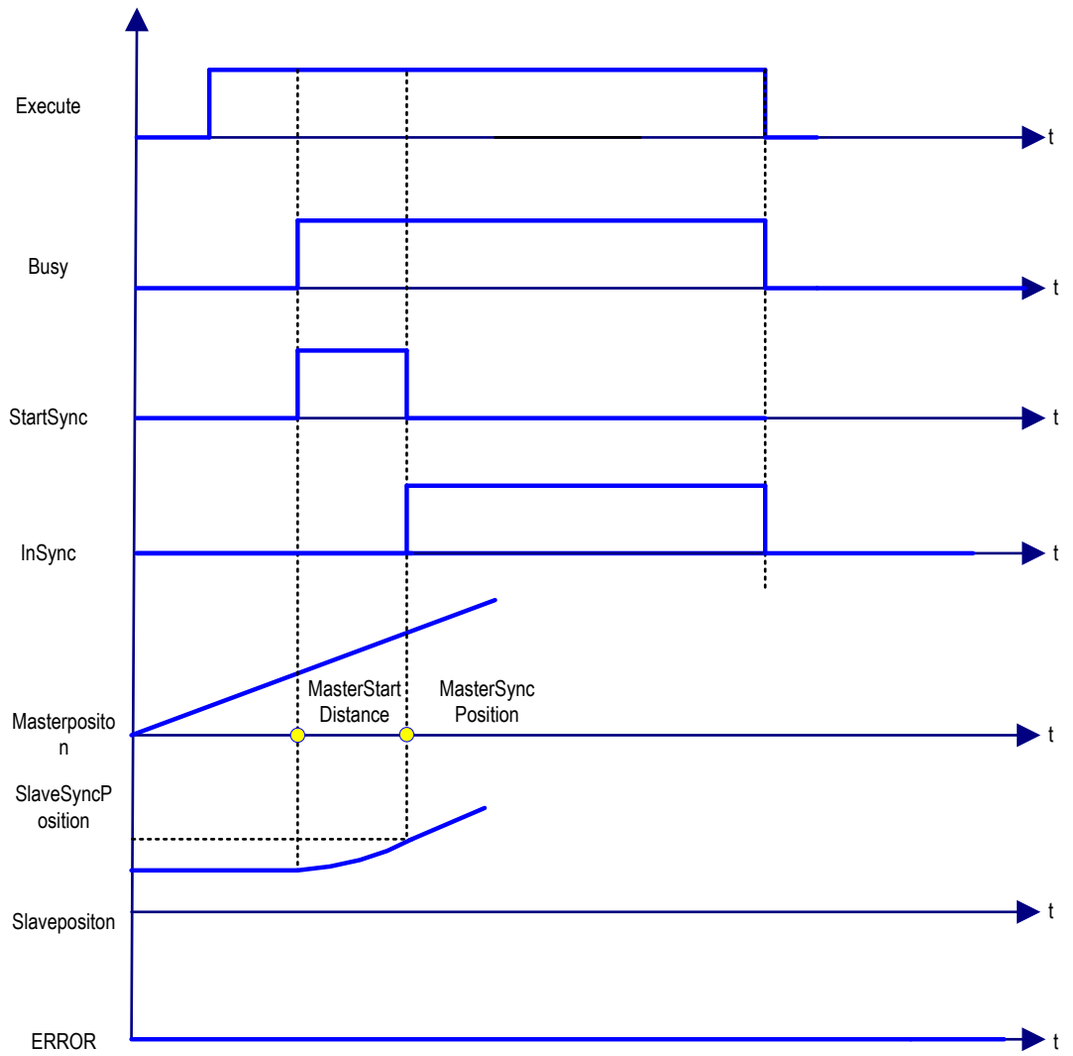
③ MasterSyncPosition=300、MasterStartDistance=50、SlaveSyncPosition=60，主轴速度为 50、AvoidReversal=FALSE、从轴起始位置大于 60



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。

4） 时序图



5） 错误说明

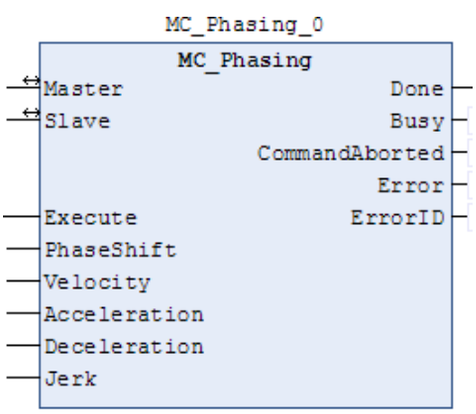
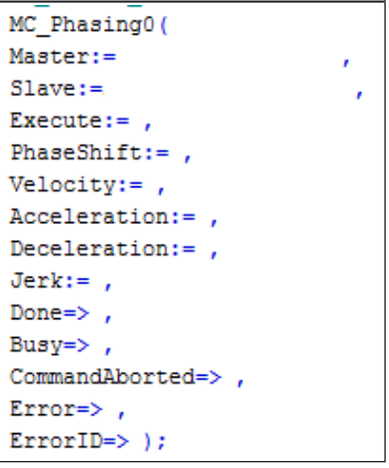
- 相关参数设置有错，会导致指令报警。
- 轴没有使能会导致指令报警。

【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。

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
Acceleration	加速度	LREAL		0	执行相位偏移时最大加速度值
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
CommandAborted	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。

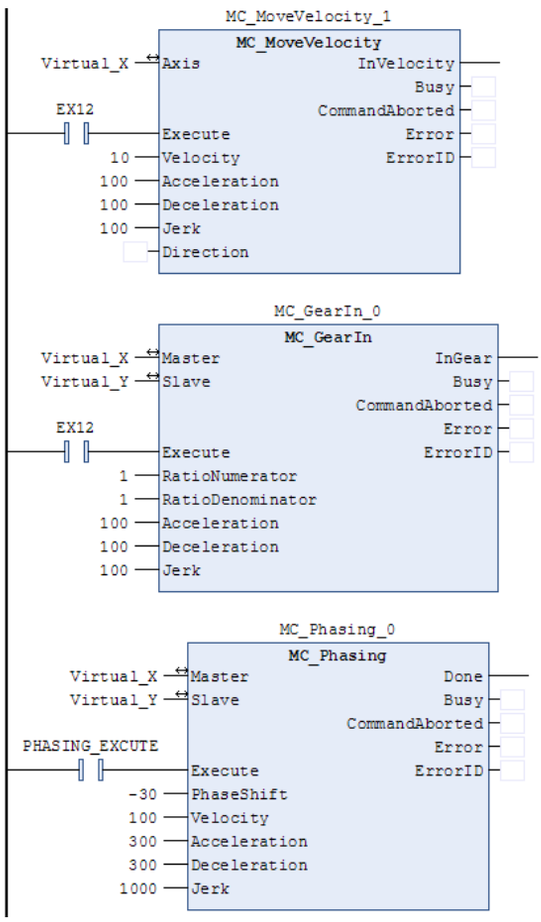
根据设定的 PhaseShift、Velocity、Acceleration、Deceleration 对主从轴相位差进行补偿。

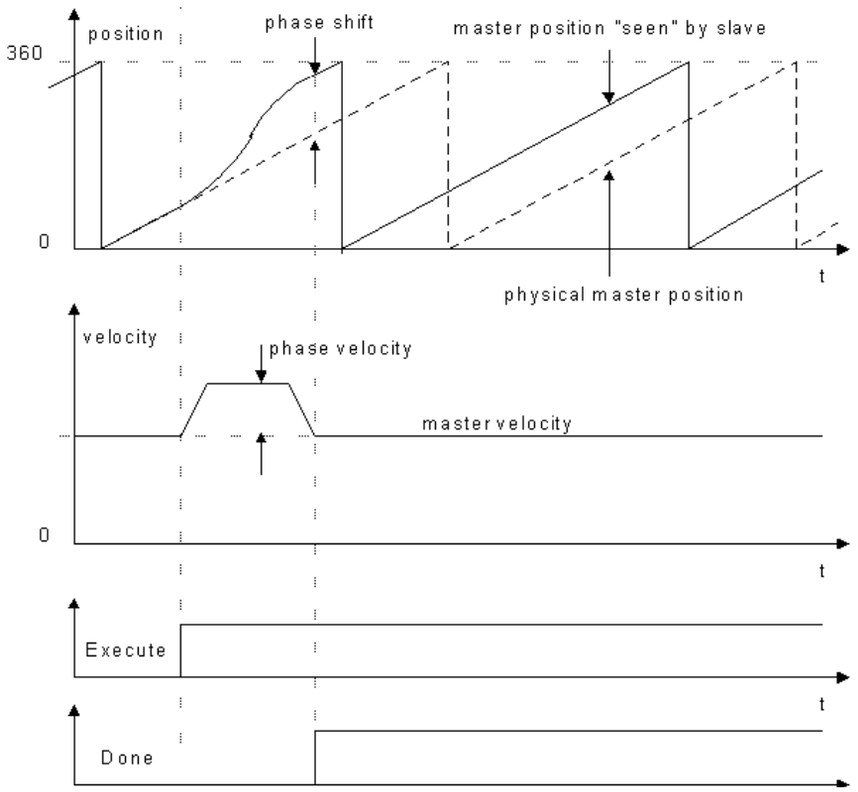
主从轴相位差达到 PhaseShift 时，Done 信号输出。

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。

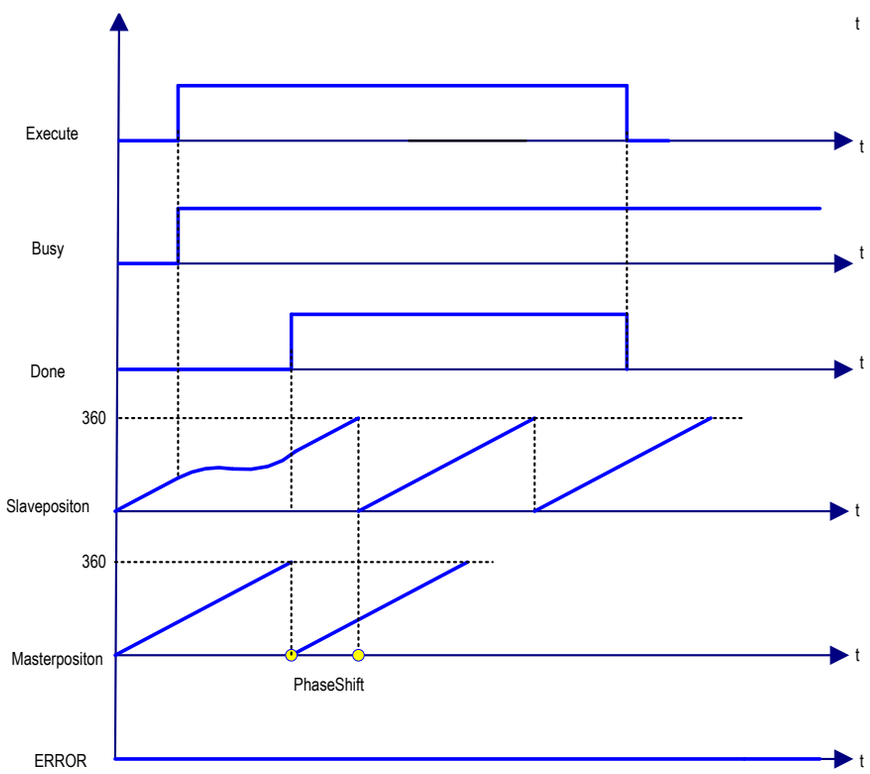
该指令跟 MC_GearIn 指令配合使用，如下：主轴为 Virtual_x，从轴为 Virtual_y,EX12 上升沿执行主轴速度控制以及主从轴电子齿轮动作，然后执行相位偏移。此外可以跟电子凸轮配合使用，此时从轴作为“电子凸轮主轴”以达到电子凸轮主轴相位偏移的效果。





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

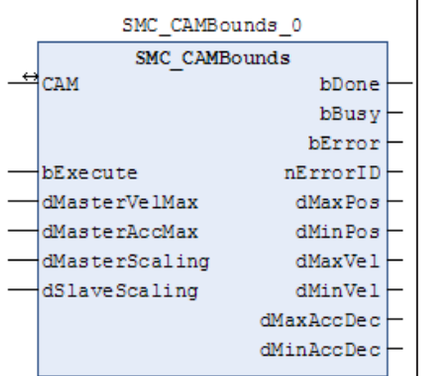
- 启动指令时，Error 输出为 TRUE，则有错误发生。
- 查看 ErrorID, 查看帮助中 SMC_ERROR 确定报警信息，请阅读“附录 C 错误代码说明”了解相关错误代码说明。

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		SMC_CAMBounds_0(CAM:= , bExecute:= , dMasterVelMax:= , dMasterAccMax:= , dMasterScaling:= , dSlaveScaling:= , bDone=> , bBusy=> , bError=> , nErrorID=> , dMaxPos=> , dMinPos=> , dMaxVel=> , dMinVel=> , dMaxAccDec=> , dMinAccDec=>);

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

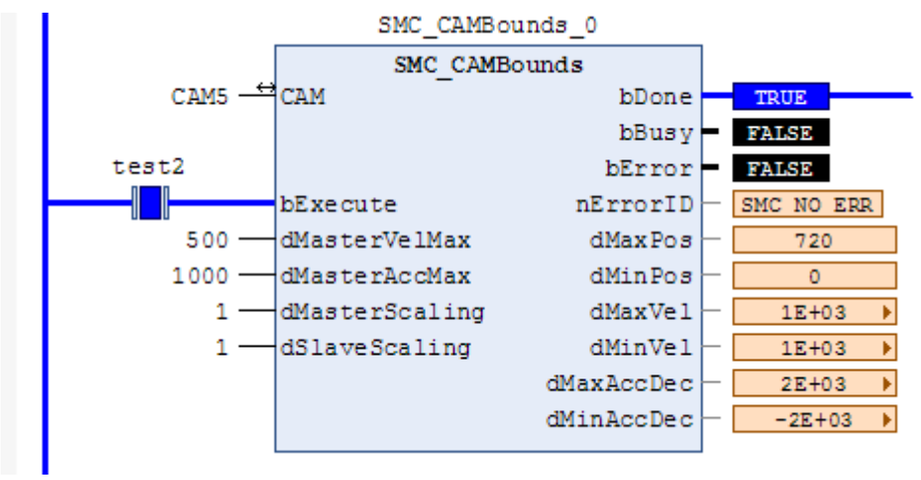
输出变量	名称	数据类型	有效范围	初始值	描述
bDone	完成	BOOL	TRUE- FALSE	FALSE	True, 如果计算完成
bBusy	指令处理中	BOOL	TRUE,FALSE	FALSE	True，指令正在处理中
bError	错误	BOOL	TRUE,FALSE	FALSE	异常发生时，置为 TRUE
nErrorID	错误代码	SMC_ERROR	参阅 SMC_ERROR	0	异常发生时，输出错误代码
dMaxPos	最大位置	LREAL		0	根据凸轮表计算出从轴最大位置
dMinPos	最小位置	LREAL		0	根据凸轮表计算出从轴最小位置
dMaxVel	最大速度	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

3） 功能说明

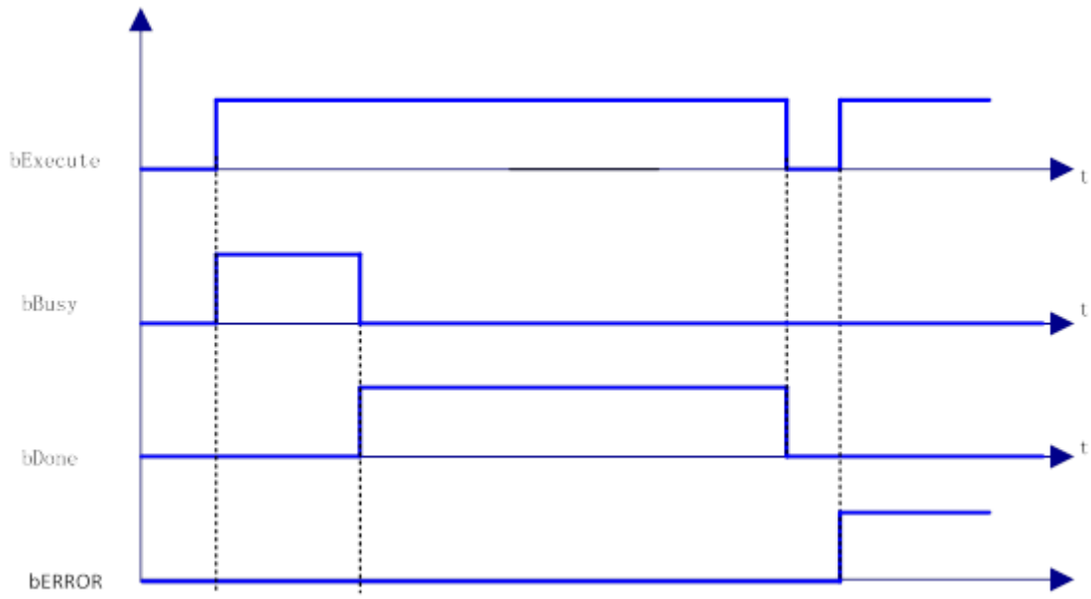
bExecute 上升沿来综合输入变量的“dMasterVelMax”，“dMasterAccMax”，“dMasterScaling”，“dSlaveScaling”值与凸轮表数据计算出从轴“最大位置”最小位置等值。例如：主轴周期 360，凸轮表为一条斜率为 2 的直线，计算出的结果如下图所示：

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



4） 时序图



5） 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。

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_Pos_0 SMC_CAMBounds_Pos CAM bDone bBusy bError bExecute nErrorID dMasterVelMax dMaxPos dMasterAccMax dMinPos dMasterScaling dSlaveScaling	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
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

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

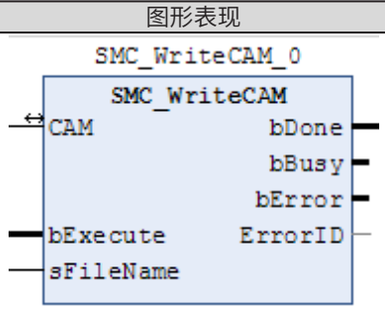
凸轮表格式不是多项式模式；凸轮表 MC_CAM_REF 设定值与实际凸轮表不匹配。
【注意】：请阅读“附录 C 错误代码说明”以了解相关错误代码说明。

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，输出错误代码

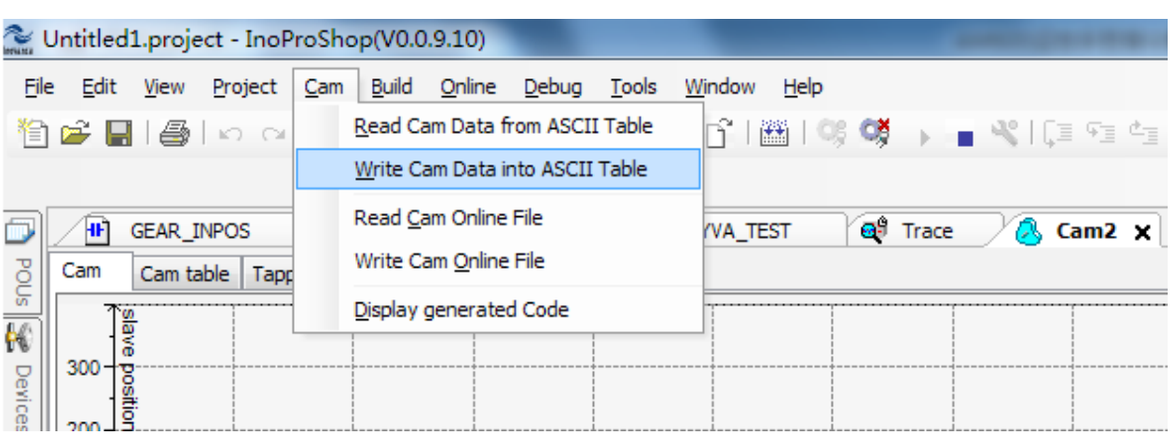
3） 功能说明

◆ 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

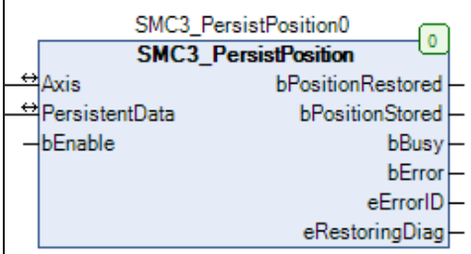
◆ sFileName 连接的文件名不存在或者信息错误。

【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。

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_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

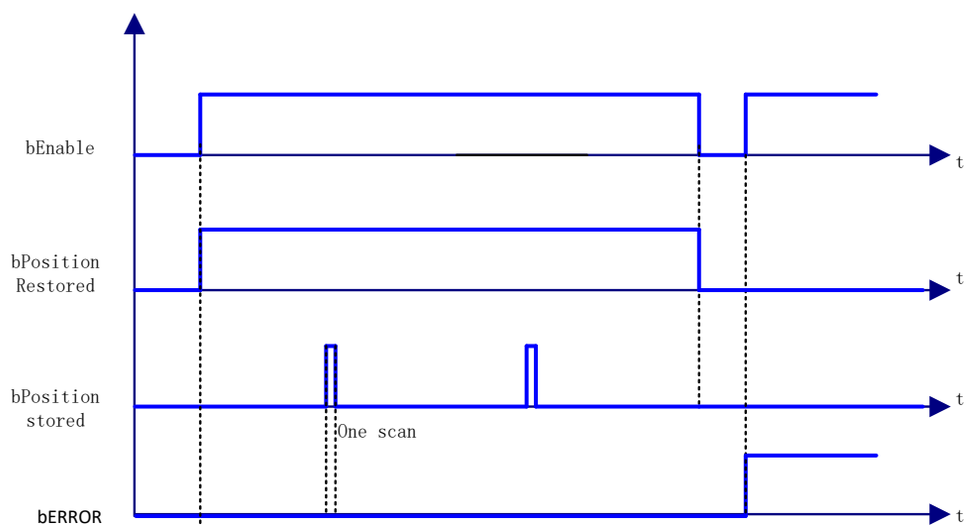
◆ 输出变量

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_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。
- 时序图



4） 错误说明

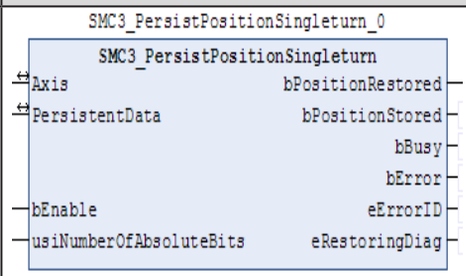
- 输入轴为虚拟轴或者逻辑轴会导致错误输出。
- 轴有错误。

【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。

SMC3_PersistPositionSingleturn

该指令用来保持纪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_PersistPositionSingleturn	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
PersistentData	保持数据	SMC3_PersistPositionSingletrun_Data			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

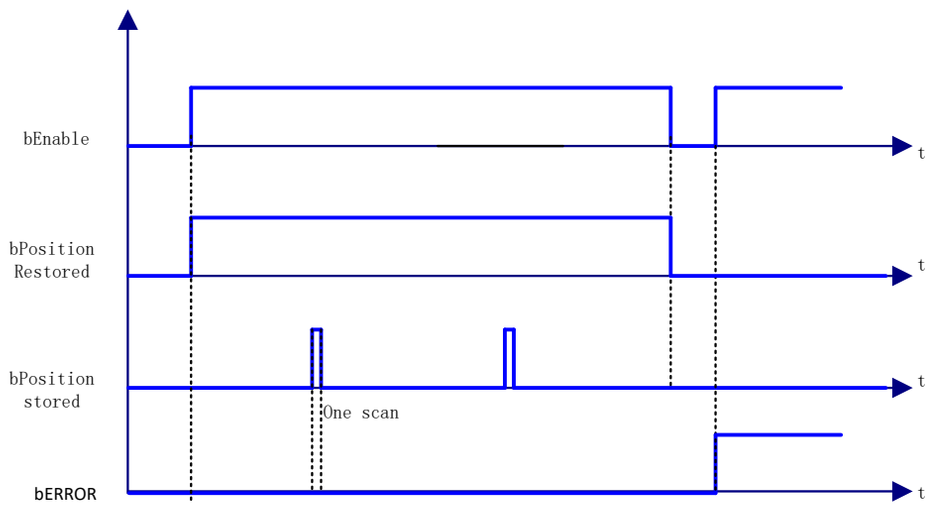
输出变量	名称	数据类型	有效范围	初始值	描述
bPositionRestored	位置恢复	BOOL	TRUE,FALSE	FALSE	TRUE, 轴重启后位置恢复完成
bPositionStored	位置保存	BOOL	TRUE,FALSE	FALSE	TRUE, 调用功能快后保存位置完成
bBusy	FB 执行中	BOOL	TRUE,FALSE	FALSE	TRUE，功能块没有执行完成
bError	错误	BOOL	TRUE,FALSE	FALSE	TRUE, 异常发生
eErrorID	错误代码	SMC_ERROR		SMC_NO_ERROR	异常发生时，输出错误代码
eRestoringDiag	恢复诊断	SMC3_Persist PositionDiag		SMC3_PersistPositionDiag. SMC3_PPD_RESTORING_ OK	位置恢复中的诊断信息

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。

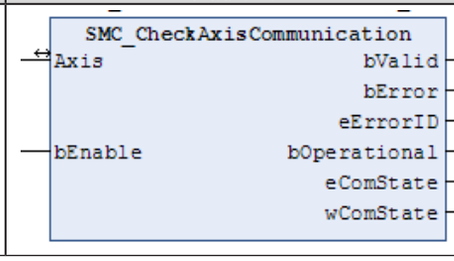
轴有错误。

【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。

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		SMC_CheckAxisCommunication0(Axis:= , bEnable:= , bValid=> , bError=> , 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_ERROR			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

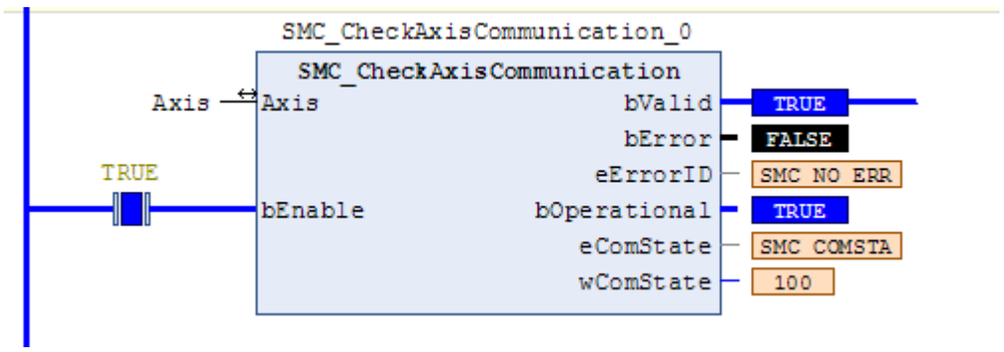
eComState	通讯状态	SMC_ COMMUNI- CATIONSTATE			包含： SMC_COMSTATE_NOT_STARTED, 通讯没有启动 SMC_COMSTATE_VARIABLE_INITIALIZATION, 通讯变量初始化 SMC_COMSTATE_BASE_COM_INITIALIZATION, 基本端口初始化 SMC_COMSTATE_DRIVE_INITIALIZATION, 通讯驱动初始化 SMC_COMSTATE_DRIVE_WAITING_FOR_SYNC, 同步警告 SMC_COMSTATE_INITIALIZATION_DONE, 初始化完成 SMC_COMSTATE_OPERATIONAL, 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。

SMC3_PersistPositionSingleturn

该指令用来保持纪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_PersistPosition Singleturn	轴位置保持	SMC3_PersistPositionSingleturn_0 SMC3_PersistPositionSingleturn Axis PersistentData bEnable usiNumberOfAbsoluteBits bPositionRestored bPositionStored bBusy bError eErrorID eRestoringDiag	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
PersistentData	保持数据	SMC3_ PersistPositionSingletrun_ Data			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
usiNumberofAbsoluteBites	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

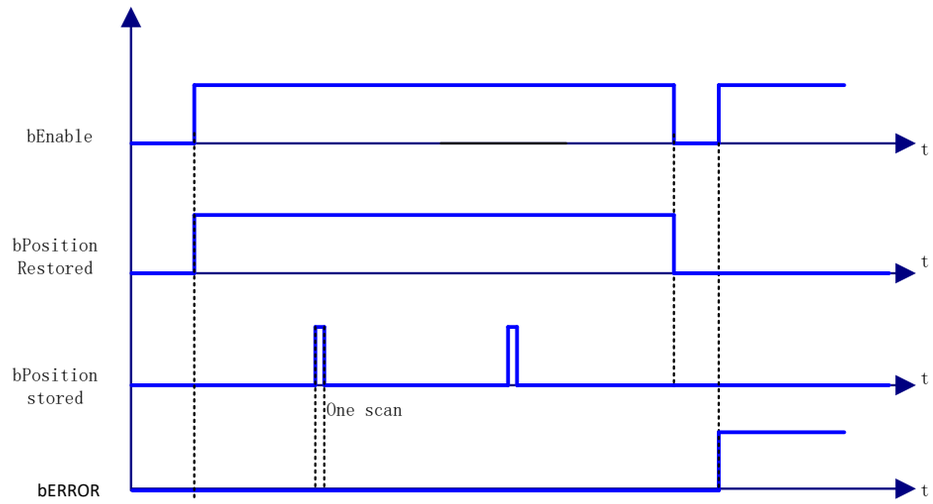
输出变量	名称	数据类型	有效范围	初始值	描述
bPositionRestored	位置恢复	BOOL	TRUE,FALSE	FALSE	TRUE, 轴重启后位置恢复完成
bPositionStored	位置保存	BOOL	TRUE,FALSE	FALSE	TRUE, 调用功能快后保存位置完成
bBusy	FB 执行中	BOOL	TRUE,FALSE	FALSE	TRUE，功能块没有执行完成
bError	错误	BOOL	TRUE,FALSE	FALSE	TRUE, 异常发生
eErrorID	错误代码	SMC_ERROR		SMC_NO_ERROR	异常发生时，输出错误代码
eRestoringDiag	恢复诊断	SMC3_ PersistPositionDiag		SMC3_ PersistPositionDiag. SMC3_PPD_ RESTORING_OK	位置恢复中的诊断信息

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；

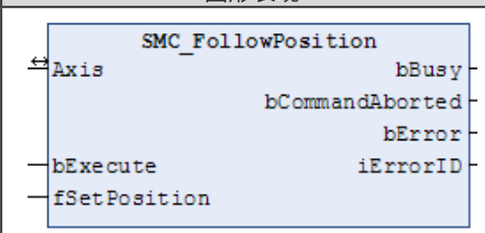
轴有错误。

【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。

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
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

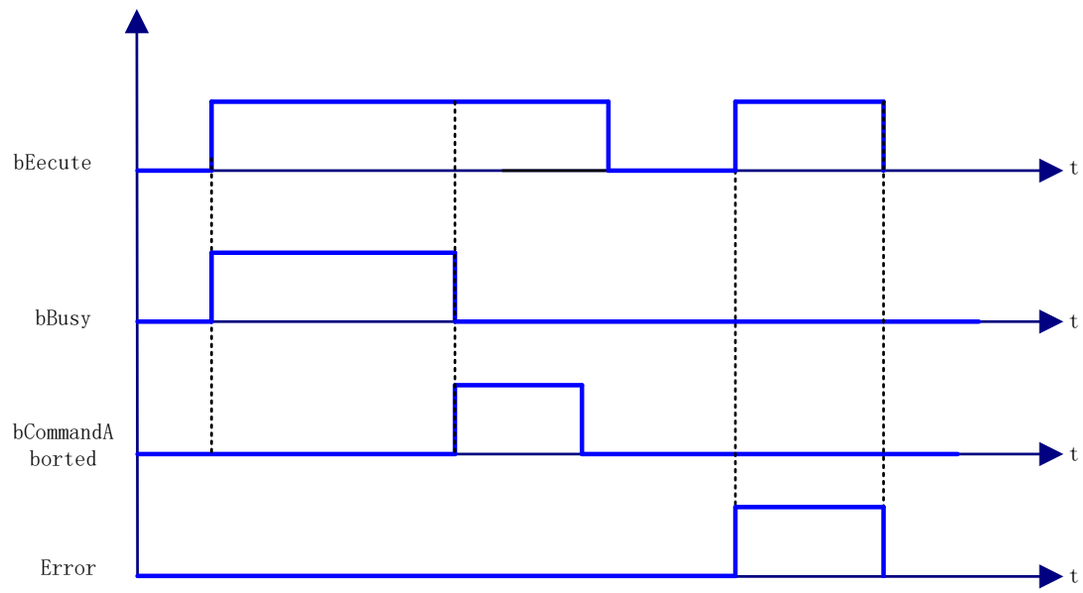
SMC_FollowPosition 通过 bExecute 的上升沿启动之后，轴会每个任务周期给轴发送位置指令。

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：$\Delta L / \Delta t$ ，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；

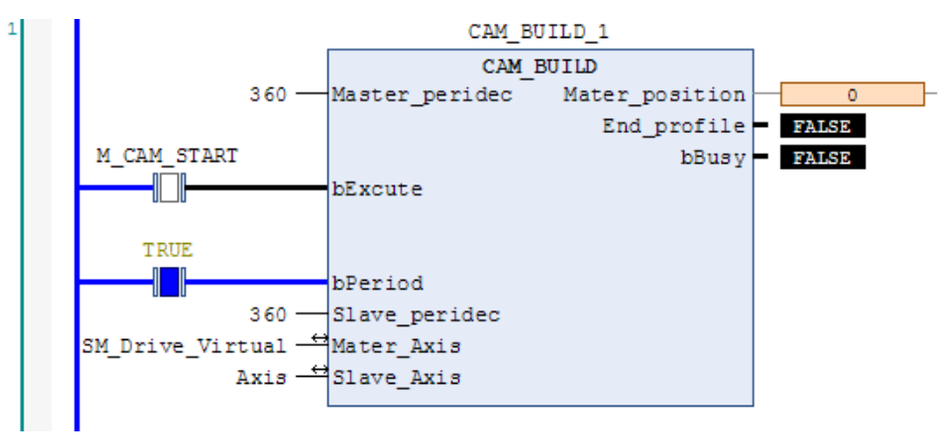
轴没使能，Error 输出。

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
```

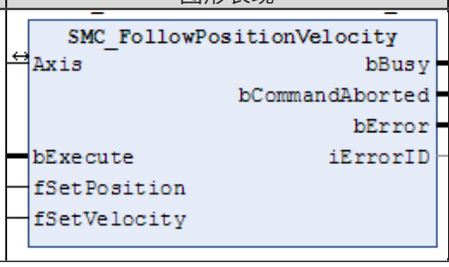
```
                    SET_POSITION:=SET_POSITION+SET_INC;
        END_IF
      END_IF
      IF SET_POSITION>=Slave_Axis.fPositionPeriod THEN
            SET_POSITION:=SET_POSITION-Slave_Axis.fPositionPeriod;
      END_IF
END_IF
SMC_FollowPosition_0(
      Axis:=Slave_Axis,
      bExecute:=bExcute,
      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;
```

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

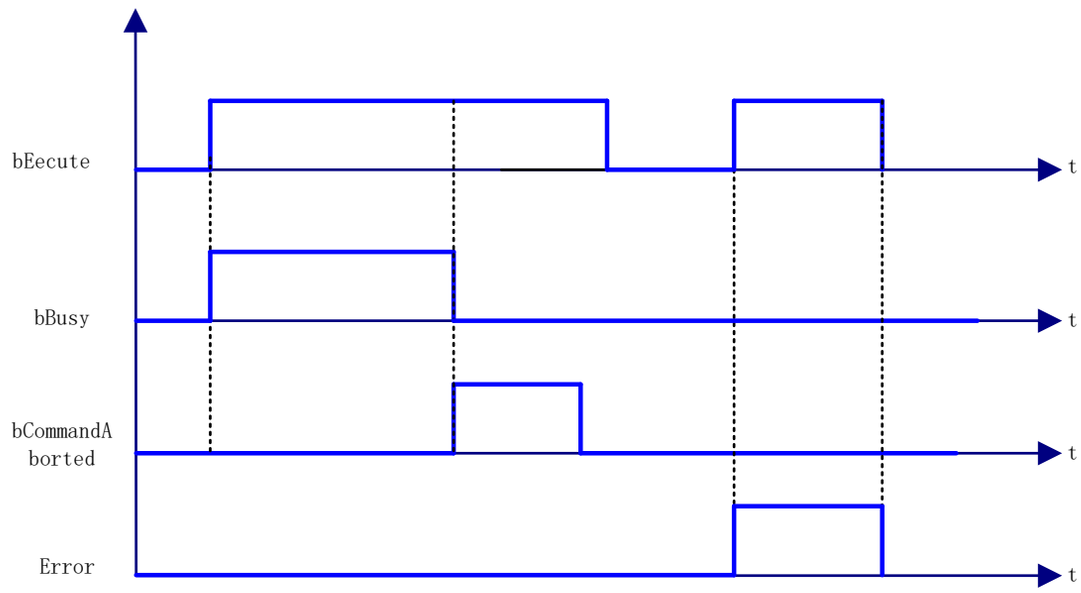
SMC_FollowPositionVelocity 通过 bExecute 的上升沿启动之后，轴会每个任务周期给轴发送设定位置、设定速度指令。

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= $\Delta L / \Delta t$ ，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；

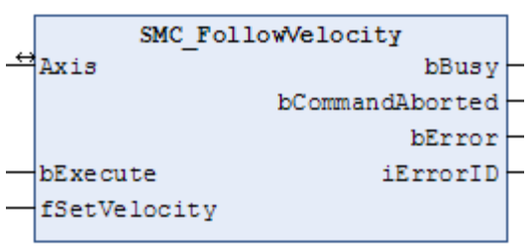
指令运行中，轴出错，Error 输出。

【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。

SMC_FollowVelocity

指令功能为不做任何检查直接给轴设定速度。该指令与 MC_MoveVelocity 有所不同 , 执行上升沿型号来后，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	设定位置	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
bCommandAborted	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

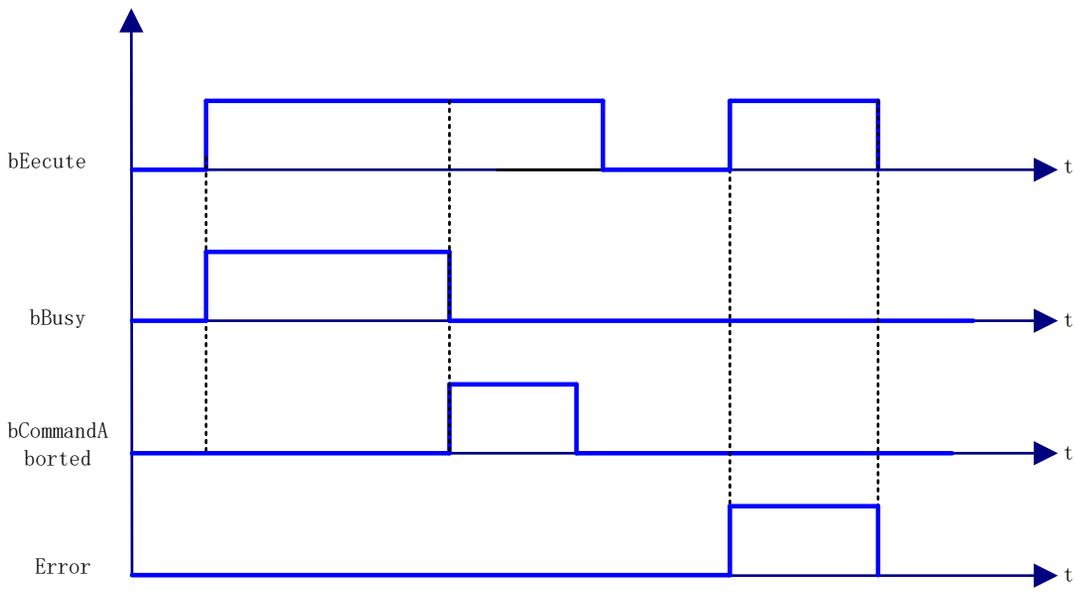
3） 功能说明

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；

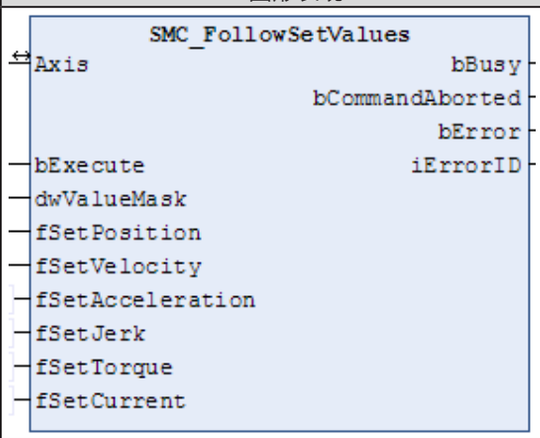
指令运行中，轴出错，Error 输出。

【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。

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

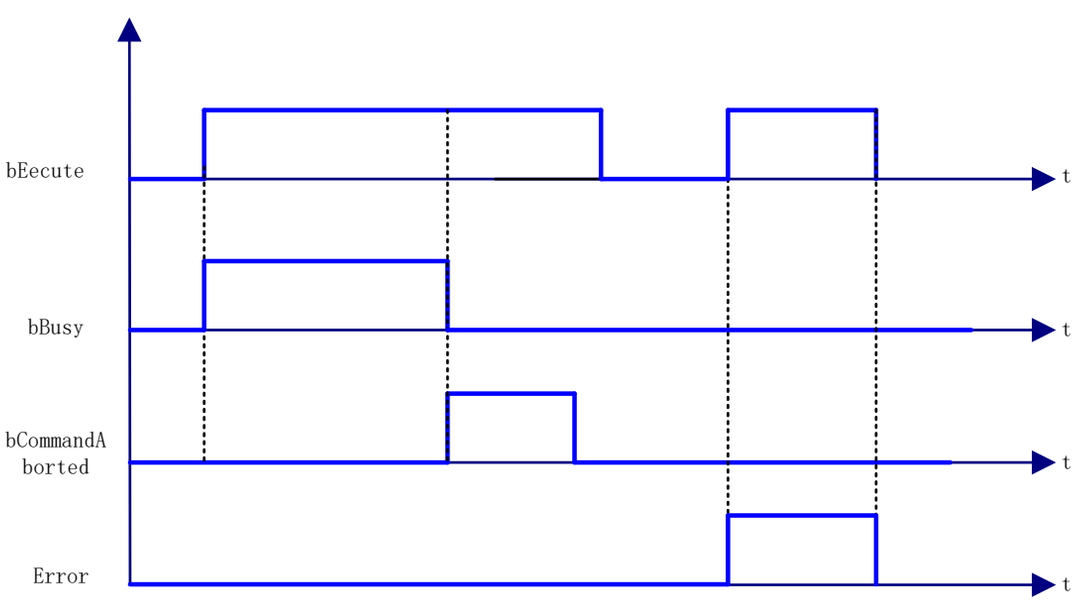
输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE,FALSE	FALSE	上升沿执行功能块
DwValueMask	控制管理	DWORD		0	Bite0:TRUE:fSetPosition 激活 FALSE: 忽略 Bite1:TRUE: fSetVelocity 激活 FALSE: 忽略 Bite2:TRUE: fSetAcceleration 激活 FALSE: 忽略 Bite3:TRUE: fSetJerk 激活 FALSE: 忽略 Bite4:TRUE: fSetTorque 激活 FALSE: 忽略 Bite5:TRUE: fSetCurrent 激活 FALSE: 忽略
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	轴设定的跃度值（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
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_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。

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《IS620N 系列伺服设计维护手册》控制模式。

1） 指令格式

指令	名称	图形表现	ST 表现
SMC_SetControllerMode	设定轴控制模式	SMC_SetControllerMode Axis bExecute nControllerMode bDone bBusy bError nErrorID	SMC_SetControllerMode0(Axis:= , bExecute:= , nControllerMode:= , 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

◆ 输入变量

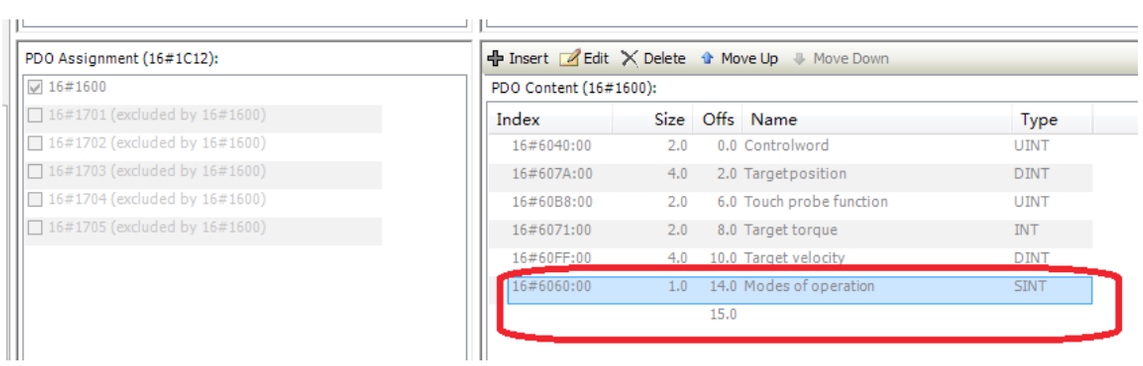
输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE,FALSE	FALSE	上升沿执行功能块
nControllerMode	控制模式	SMC_CONTROLLER_MODE		SMC_Position	轴控制模式 1：转矩控制模式，SMC_torque 2：速度控制模式，SMC_Velocity 3：位置控制模式，SMC_Position 4: 电流控制模式，SMC_Current

◆ 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：

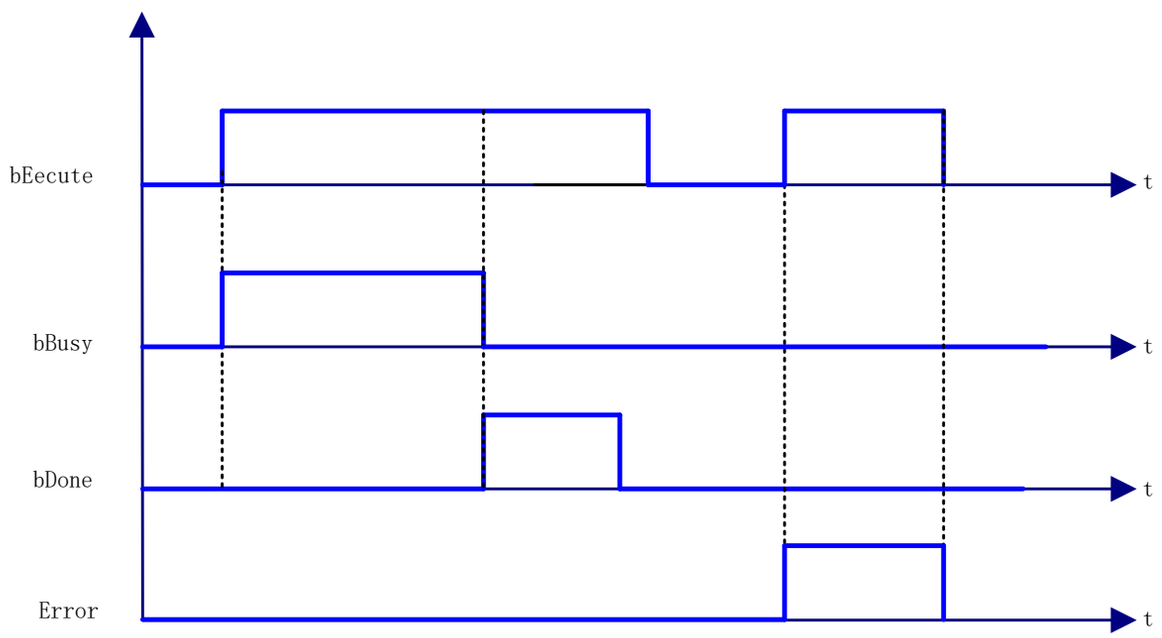
1：轴必须满足这些控制条件，比如虚轴不能使用该功能块。

2：各模式支持的同步周期必须保证一致（参考《IS620N 系列伺服设计维护手册》7.3.3 各模式支持通讯周期）

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。

4） 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。

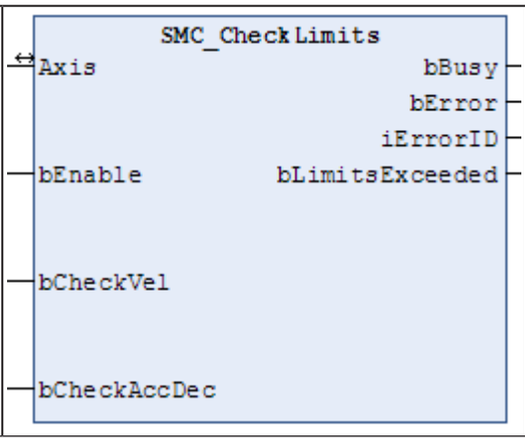
【注意】：请阅读“附录 C 错误代码说明”以了解相关错误代码说明。

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
bLimitsExceeded	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

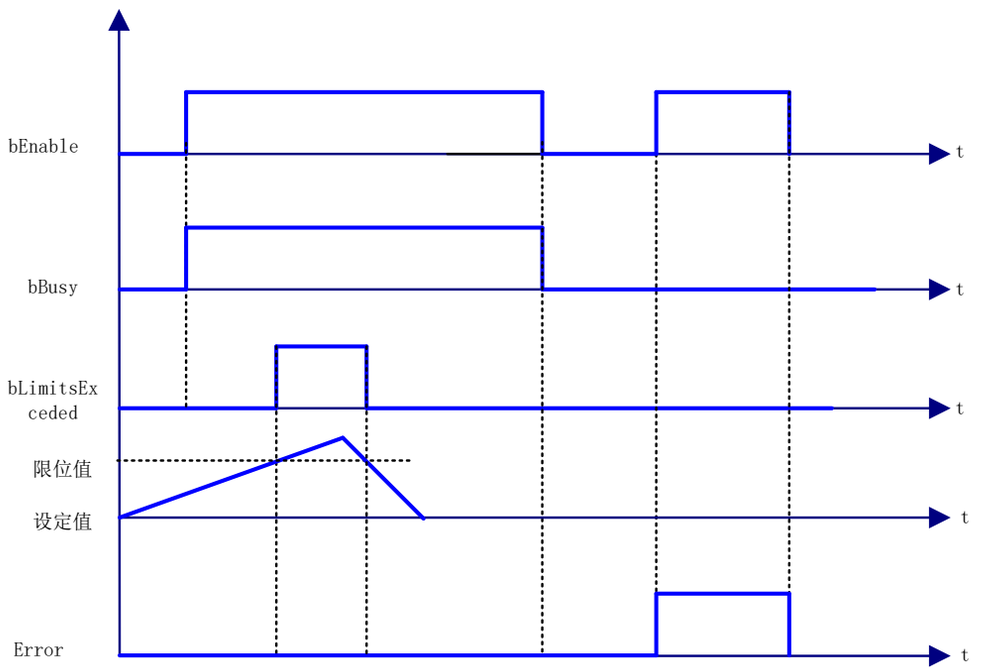
3） 功能说明

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。

4） 时序图



5） 错误说明

bExecute 上升沿时：

轴报错，Error 输出。

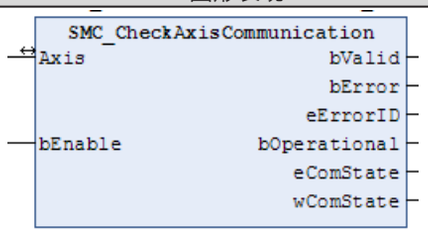
无效的轴输入，Error 输出。

【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。

SMC_CheckAxisCommuication

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		SMC_CheckAxisCommunication0(Axis:= , bEnable:= , bValid=> , bError=> , 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_ERROR			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

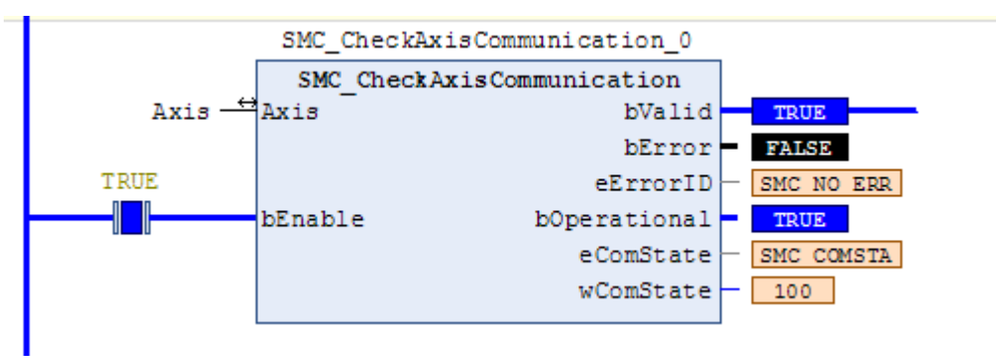
eComState	通讯状态	SMC_COMMUNICATIONSTATE			包含： SMC_COMSTATE_NOT_STARTED, 通讯没有启动 SMC_COMSTATE_VARIABLE_INITIALIZATION, 通讯变量初始化 SMC_COMSTATE_BASE_COM_INITIALIZATION, 基本端口初始化 SMC_COMSTATE_DRIVE_INITIALIZATION, 通讯驱动初始化 SMC_COMSTATE_DRIVE_WAITING_FOR_SYNC, 同步警告 SMC_COMSTATE_INITIALIZATION_DONE, 初始化完成 SMC_COMSTATE_OPERATIONAL, 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。

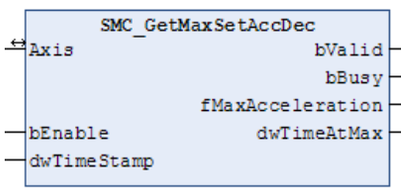
无效的轴输入，Error 输出。

【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。

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
bValid	有效	BOOL	TRUE,FALSE	FALSE	True, 指令执行有效
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True, 异常产生
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

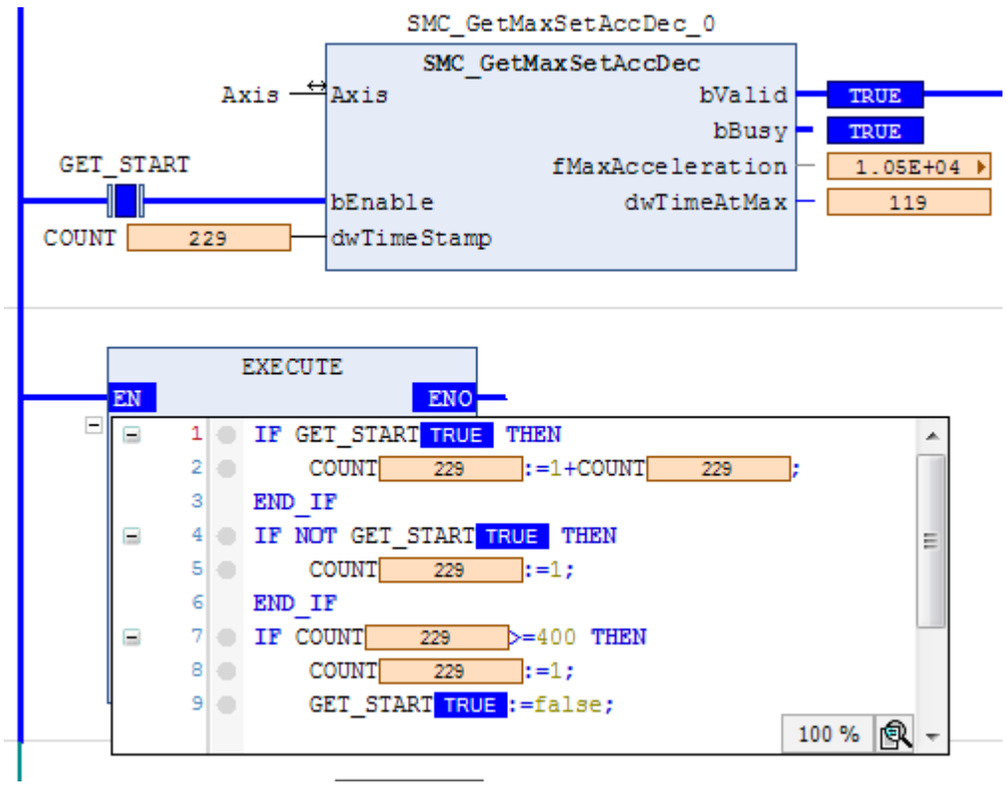
bEnable 为 TRUE, 无错误，bValid 输出 TRUE。执行轴最大加减速检测。

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）

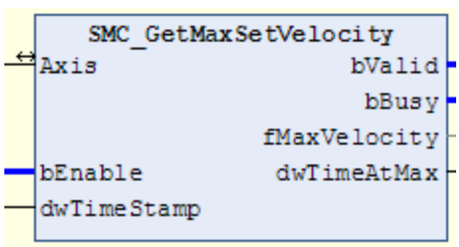
◆ 样列程序



SMC_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,FALSE	FALSE	True, 异常产生
fMaxVelocity	最大加速度值	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）

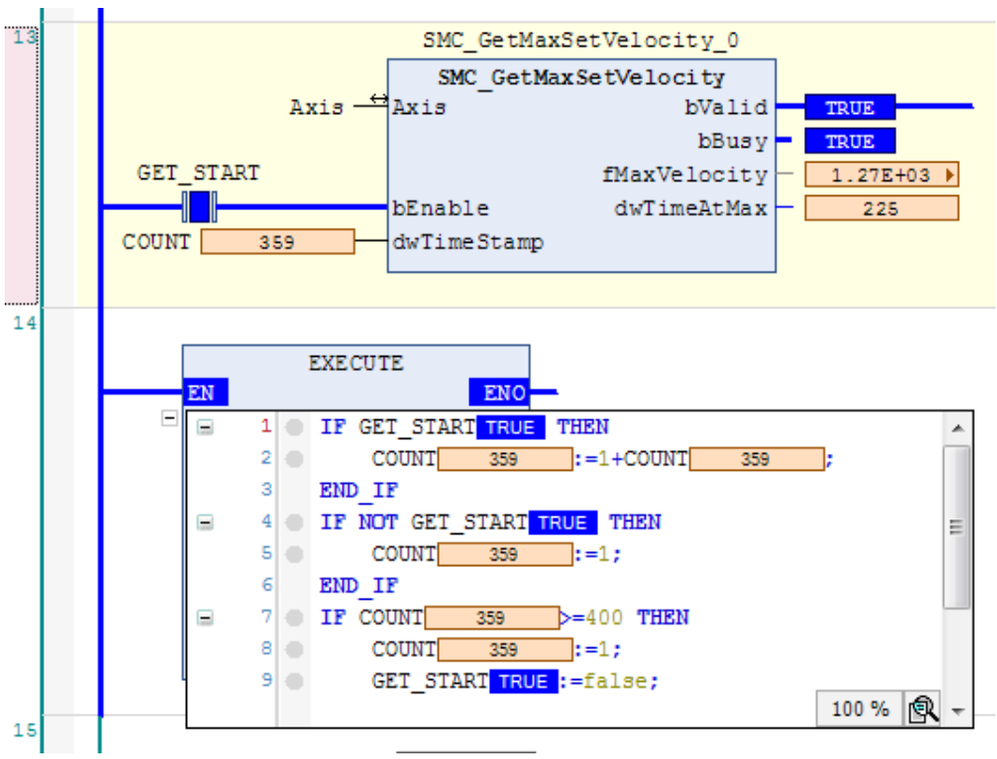
3） 功能说明

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）

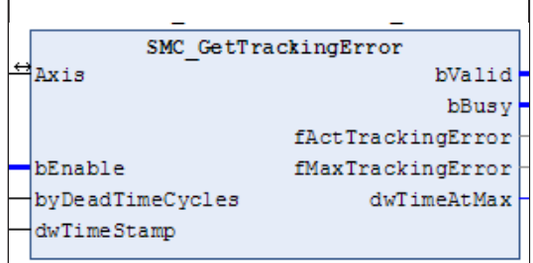
◆ 样列程序



MC_GetTrackingError

该指令功能为：测量当前或者最大滞后误差 (指令与轴实际位置差) 用于补偿死区时间。

1） 指令格式

指令	名称	图形表现	ST 表现
SMC_GeTrackingError	轴滞后偏差读取		SMC_GetTrackingError(Axis:= , bEnable:= , byDeadTimeCycles:= , dwTimeStamp:= , bValid=> , bBusy=> , fActTrackingError=> , fMaxTrackingError=> , 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
byDeadTimeCycles		Byte		2	死区周期数，bEnable 触发延迟多少个 dwTimeStamp 值开始滞后检测。
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,FALSE	FALSE	True, 异常产生
fActTrackingError	当前滞后	LREAL		0	跟 byDeaTimeCycles 值相关的当前偏差检测
fMaxTrackingError	最大滞后	LREAL		0	当前的偏差值（指令位置，跟反馈位置偏差）
dwTimeAtMax	最大值对应时间戳	Dword		0	最大的偏差值（正为滞后，负为超前，绝对值最大值为最终值） 注意：byDeaTimeCycles 值会影响该值

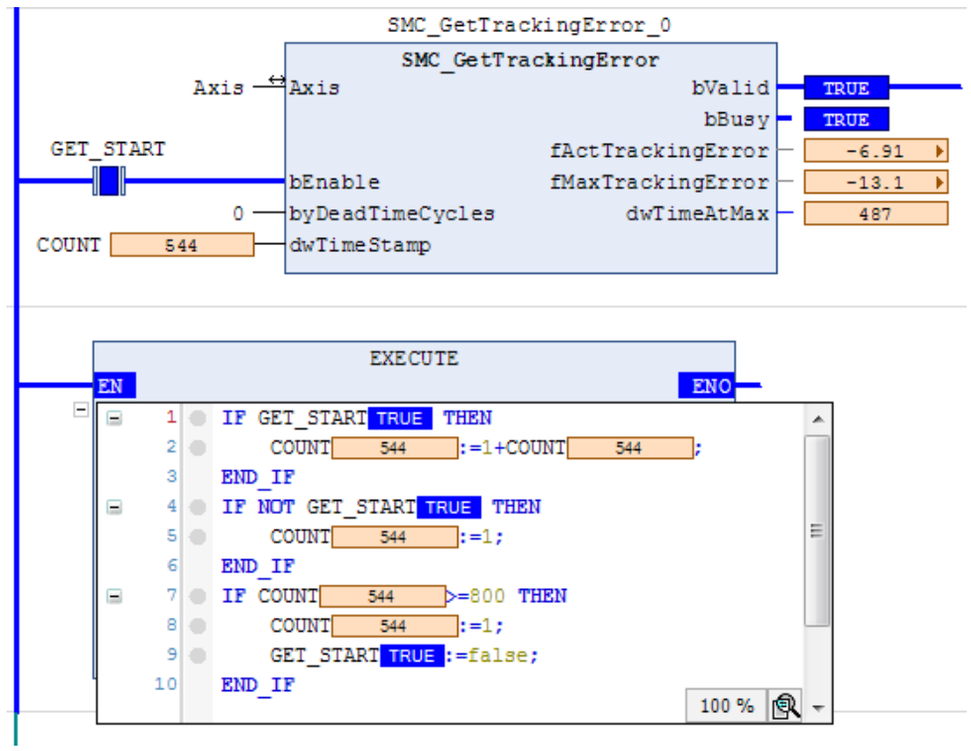
3） 功能说明

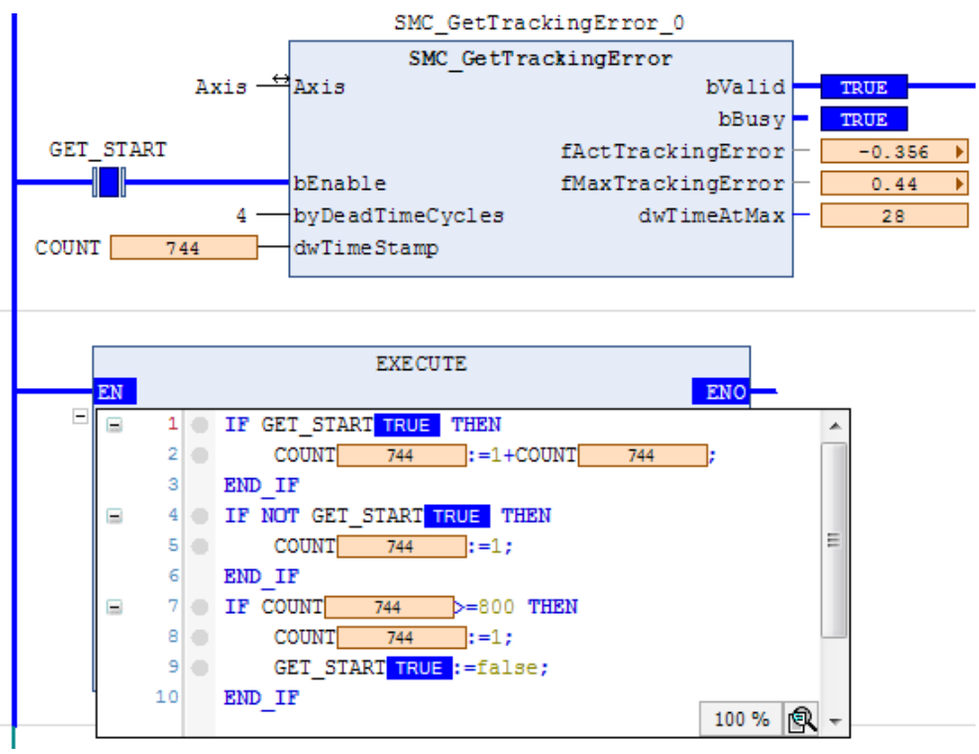
bEnable 为 TRUE，bValid 输出 TRUE。执行轴滞后偏差检测。

当偏差绝对值大于前面记录值时，fMaxTrackingError 跟 dwTimeAtMax 会刷新。

dwTimeAtMax 值为最大偏差时对应 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

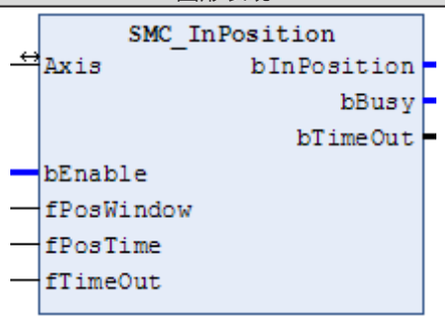

SMC_GetTrackingError_0
SMC_GetTrackingError
Axis
Axis
bValid
TRUE
bBusy
TRUE
GET_START
fActTrackingError
-6.91
bEnable
fMaxTrackingError
-13.1
0
byDeadTimeCycles
dwTimeAtMax
487
COUNT
544
dwTimeStamp
EXECUTE
EN
ENO
1 IF GET_START TRUE THEN
2 COUNT 544 :=1+COUNT 544 ;
3 END_IF
4 IF NOT GET_START TRUE THEN
5 COUNT 544 :=1;
6 END_IF
7 IF COUNT 544 >=800 THEN
8 COUNT 544 :=1;
9 GET_START TRUE :=false;
10 END_IF
100 %


SMC_GetTrackingError_0
SMC_GetTrackingError
Axis
Axis
bValid
TRUE
bBusy
TRUE
GET_START
fActTrackingError
-0.356
bEnable
fMaxTrackingError
0.44
4
byDeadTimeCycles
dwTimeAtMax
28
COUNT
744
dwTimeStamp
EXECUTE
EN
ENO
1 IF GET_START TRUE THEN
2 COUNT 744 :=1+COUNT 744 ;
3 END_IF
4 IF NOT GET_START TRUE THEN
5 COUNT 744 :=1;
6 END_IF
7 IF COUNT 744 >=800 THEN
8 COUNT 744 :=1;
9 GET_START TRUE :=false;
10 END_IF
100 %

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
fPosTime	触发时间	LREAL		0	偏差在窗口范围内时间，用来触发 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

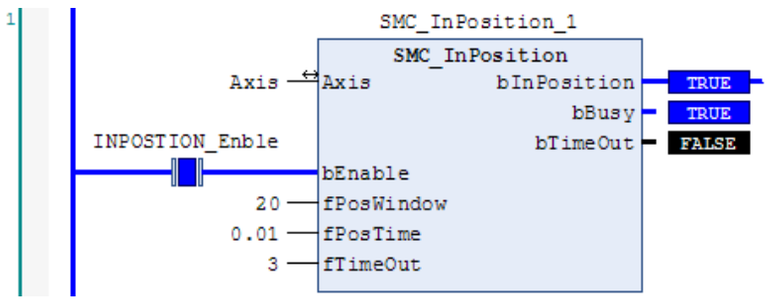
bEnable 为 TRUE，一旦检测到的偏差小于设置的窗口 fPosWindow 持续 fPosTime 秒则 bInPosition 触发为 TRUE。一旦检测到的偏差大于设置的窗口，bInPosition 立马输出为 FALSE。注意：fPosTime 时间设定需合理否则会造成 bTimeOut 触发（比如一个凸轮周期为 2 秒的凸轮，连续偏差没有超过设定窗口的时间为 1.5 秒，fPosTime 如果设定的大于 1.5 秒则会造成 bInPosition 不会触发）。

bEnable 为 TRUE，bBusy 输出为 true。

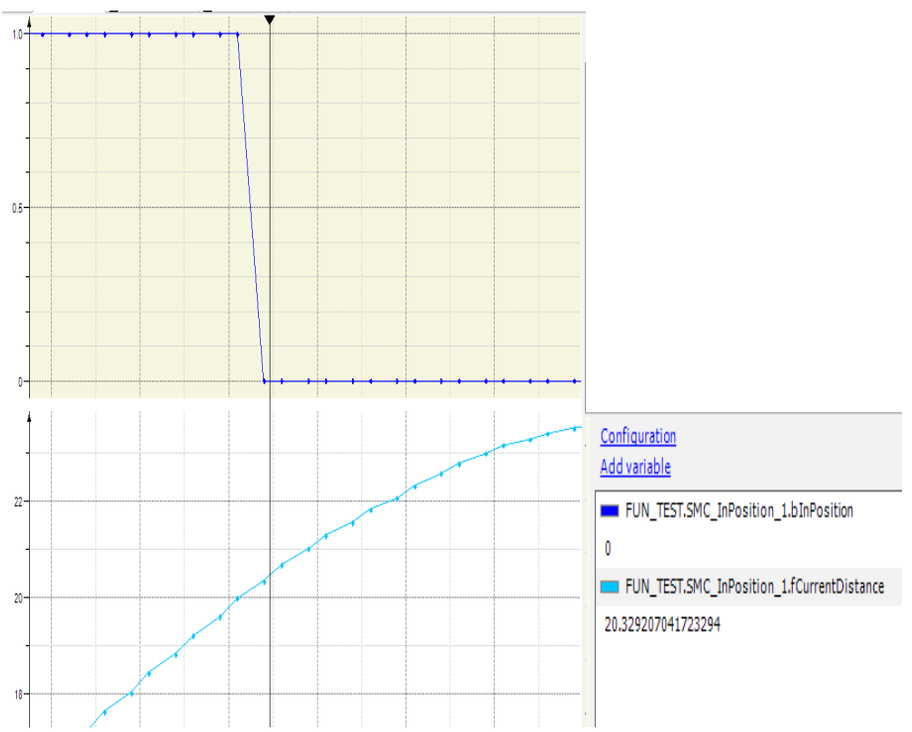
偏差值可监控 SMC_InPosition 结构体中的数据 fCurrentDistance。

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。

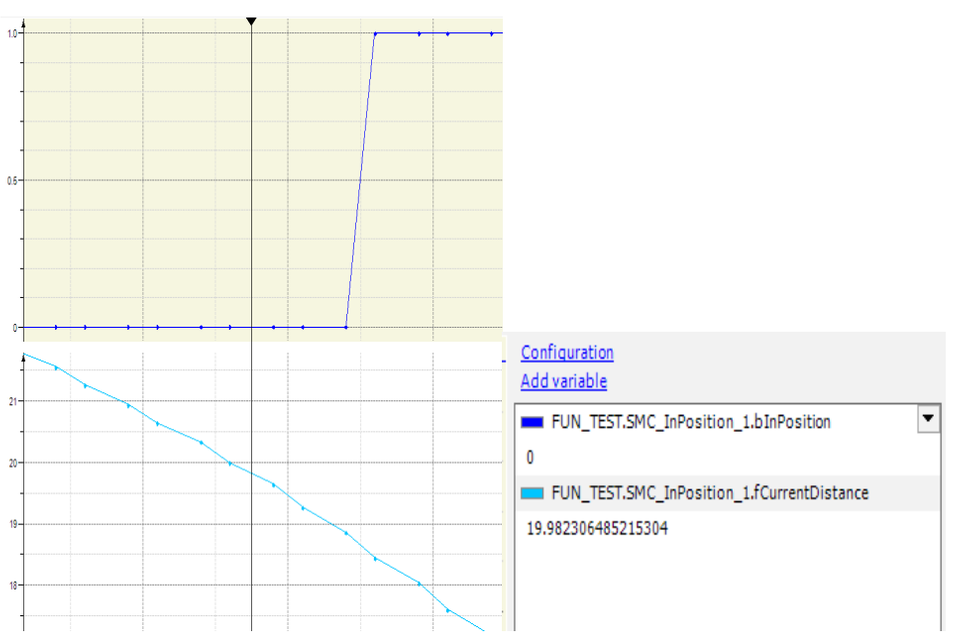
◆ 时序图样列程序



◆ 样例程序

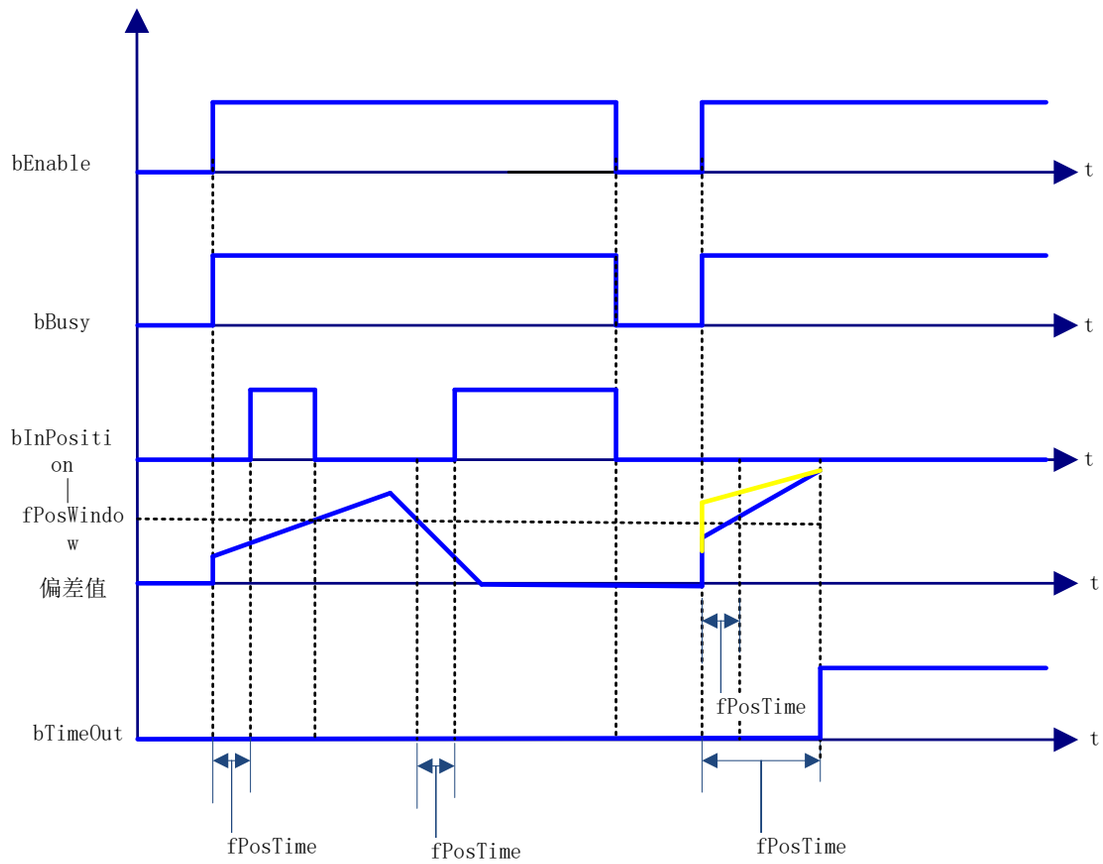


大于窗口设定 bInPosition 立即由 true 变为 FALSE



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

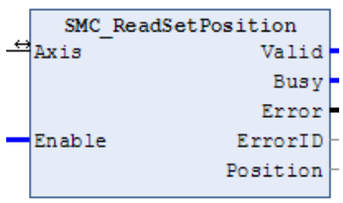
4） 时序图



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

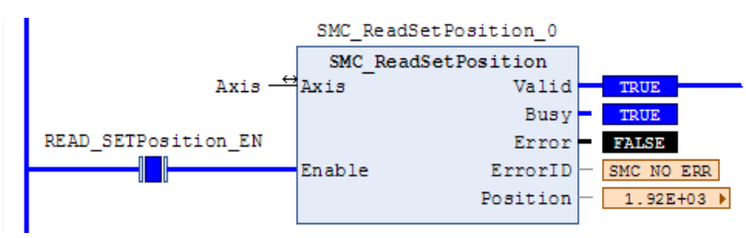
3） 功能说明

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

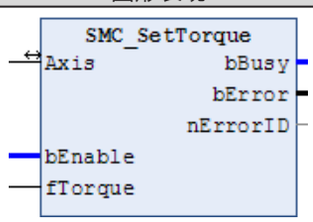
bExecute 上升沿时：轴报错，Error 输出；无效的轴输入，Error 输出。

【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。

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

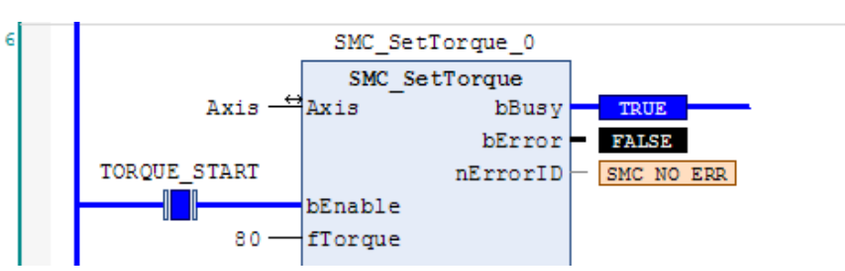
输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行中	BOOL	TRUE,FALSE	FALSE	True, 执行中
Error	错误	BOOL	TRUE,FALSE	FALSE	True, 异常产生
ErrorID	错误代码	SMC_ERROR			参考 SMC_Error

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。

◆ 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

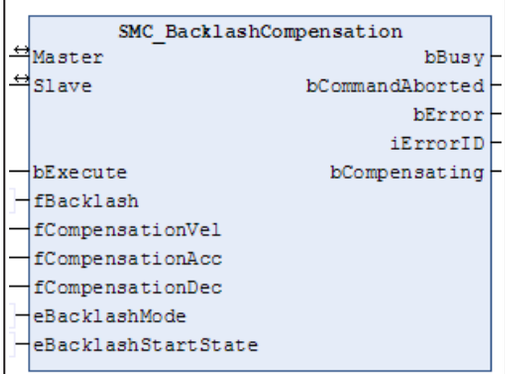
【注意】：请阅读“附录 C 错误代码说明”以了解相关错误代码说明。

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=>);

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
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
eBacklashStartState		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

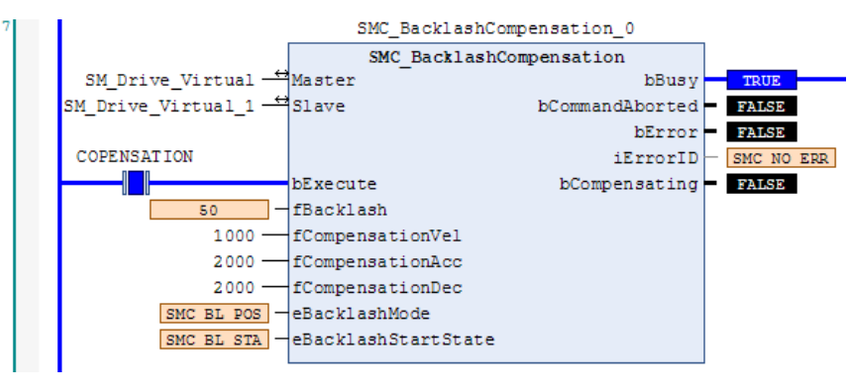
输出变量	名称	数据类型	有效范围	初始值	描述
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True, 执行中
bCommandAborted	指令被中断	BOOL	TRUE,FALSE	FALSE	True- 被其他控制命令打断
bError	错误	BOOL	TRUE,FALSE	FALSE	True, 异常产生
iErrorID	错误代码	SMC_ERROR			参考 SMC_Error
bCompsating	补偿中	BOOL	TRUE,FALSE	FALSE	

3） 功能说明

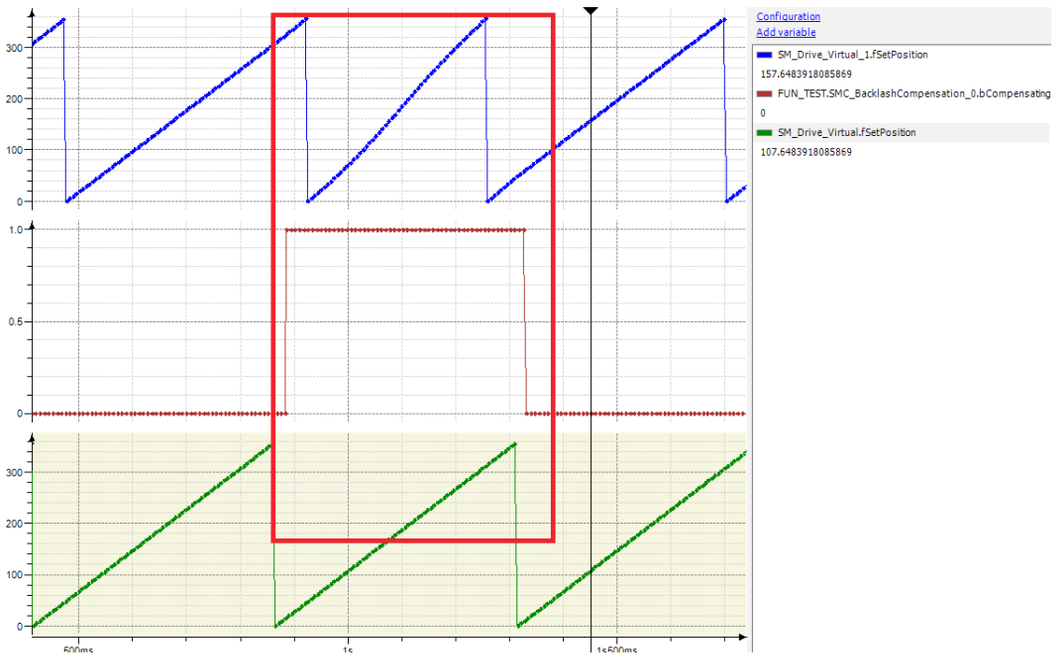
bEecute 上升沿，无错误则，bBusy 输出为 TURE，bCompsating 输出为 true，补偿完成后 bCompsating 输出为 false。

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

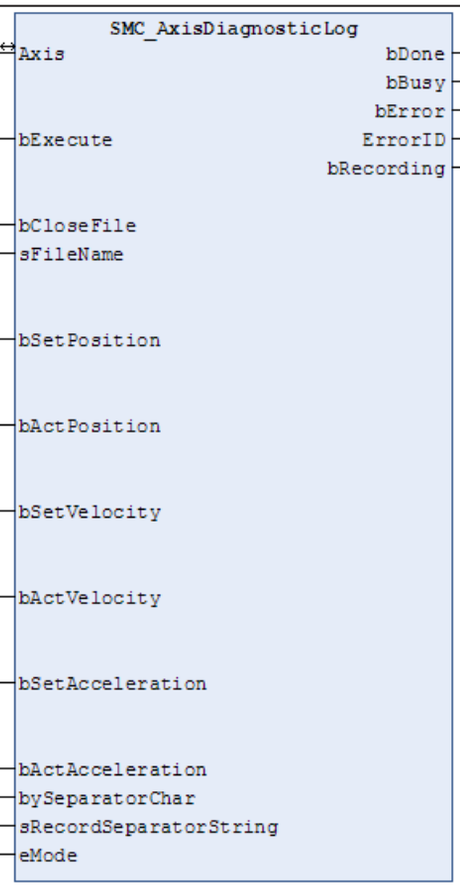
bExecute 上升沿时：
轴报错，Error 输出；无效的轴输入，Error 输出。
【注意】：请阅读“附录 C 错误代码说明”以了解相关错误代码说明。

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_AxisDiagnosticLog	轴参数写入文件		SMC_AxisDiagnosticLog(Axis:= , bExecute:= , bCloseFile:= , sFileName:= , bSetPosition:= , bActPosition:= , bSetVelocity:= , bActVelocity:= , bSetAcceleration:= , bActAcceleration:= , bySeparatorChar:= , sRecordSeparatorString:= , eMode:= , bDone=> , bBusy=> , bError=> , ErrorID=> , bRecording=>);

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
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
sRecordSeparatorString				‘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

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：

4） 错误说明

bExecute 上升沿时：

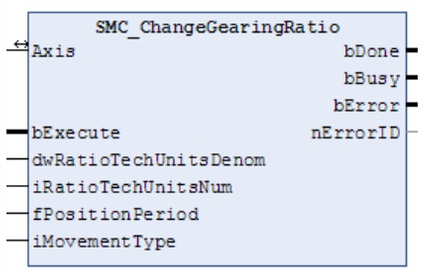
轴报错，Error 输出；无效的轴输入，Error 输出。

【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。

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

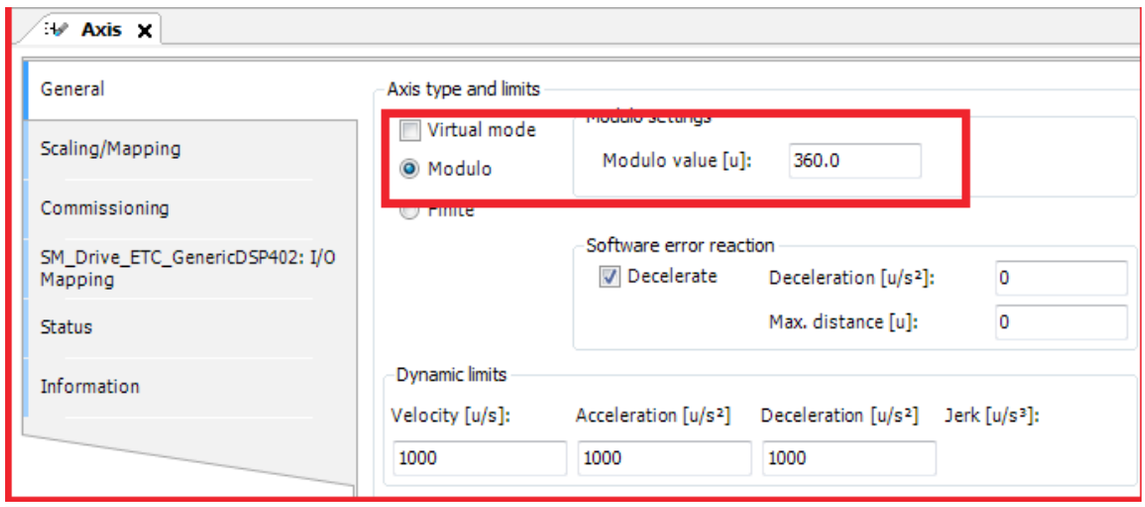
输出变量	名称	数据类型	有效范围	初始值	描述
bDone	完成	BOOL	TRUE,FALSE	FALSE	True, 执行设定完成
bBusy	执行中	BOOL	TRUE,FALSE	FALSE	True, 执行中
bError	错误	BOOL	TRUE,FALSE	FALSE	True, 异常产生
nErrorID	错误代码	SMC_ERROR			参考 SMC_Error

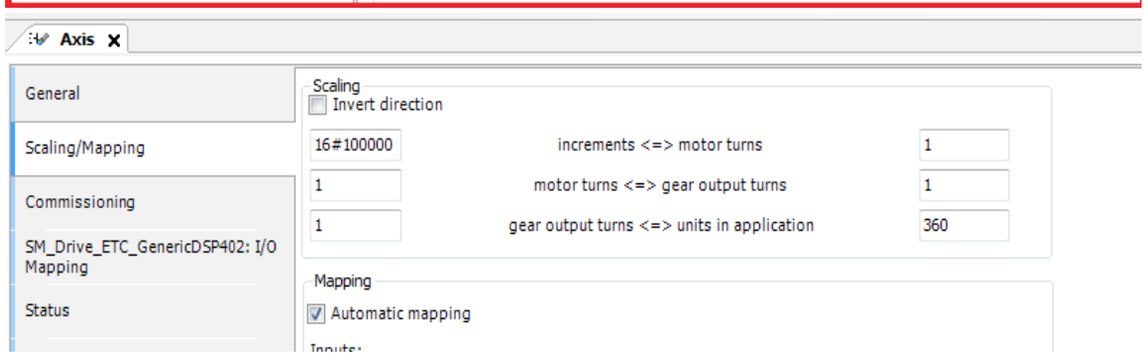
3） 功能说明

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：

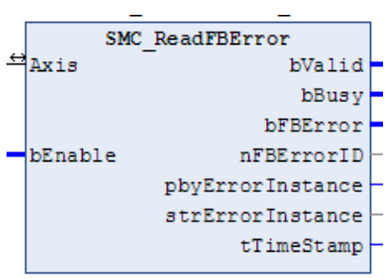
- 轴报错，Error 输出。
- 输入值无效，Error 输出 , 错误代码 SMC_CGR_ZERO_VALUES
- 轴在指令控制运行中，Error 输出 , 错误代码 SMC_CGR_DRIVE_POWERED
- 输入的模数值无效（eg：<0），Error 输出 , 错误代码 SMC_CGR_INVALID_POSPERIOD

【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。

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_ ReadFBError	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					输出点的功能块报错
strErrorInstance					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

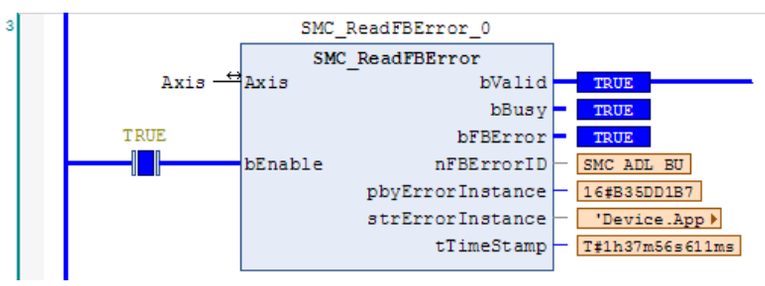
3） 功能说明

Enable 为 TRUE，无错误则 Valid，Busy 输出为 TURE。

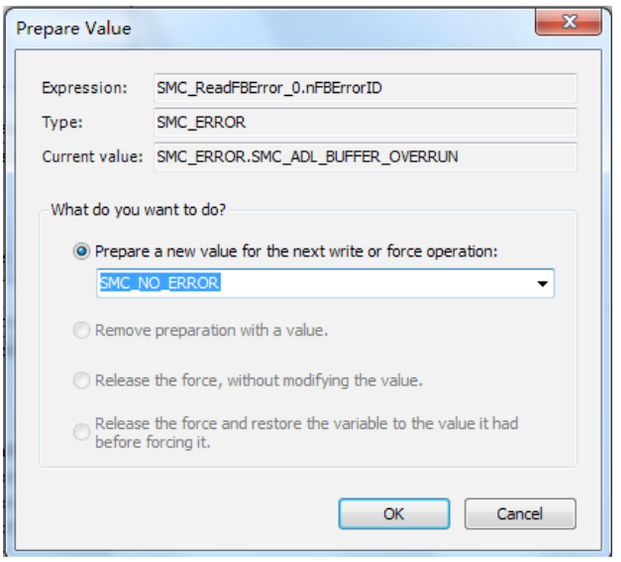
有功能块报警则，bFBError 输出为 true。

Enable 变为 FALSE，, 则 Valid，Busy 输出为 FALSE。

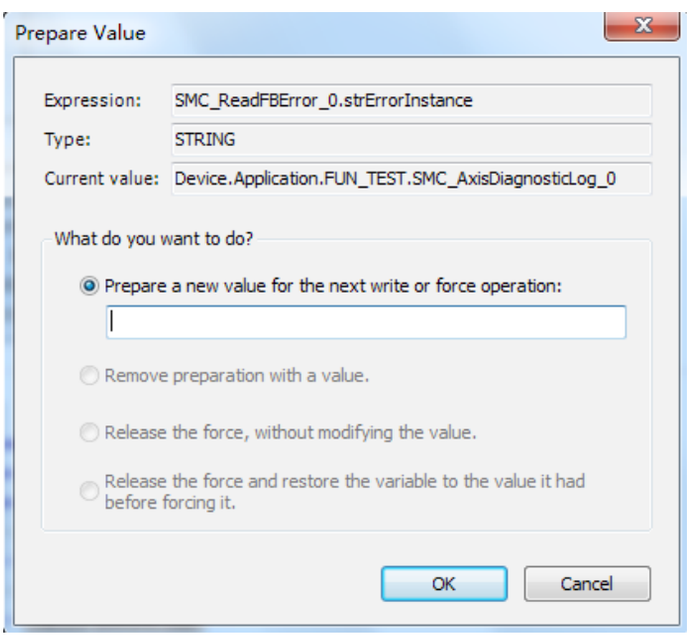
◆ 时序图样列程序

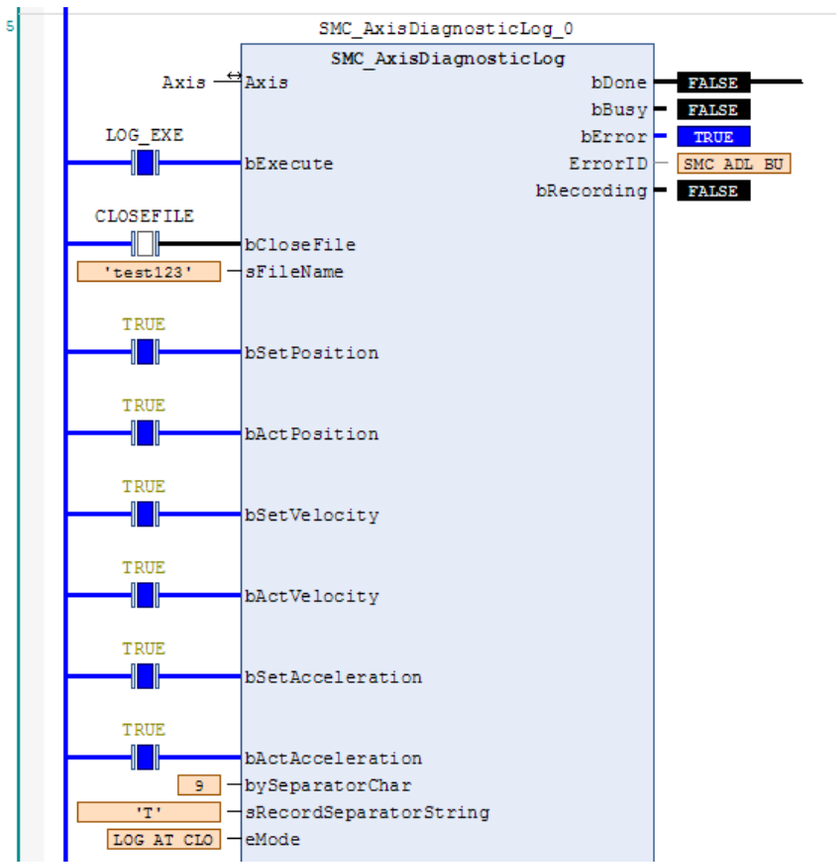


◆ 样例程序


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

4） 错误说明

bExecute 上升沿时：

轴报错，Error 输出；

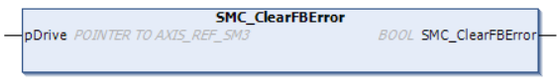
无效的轴输入，Error 输出。

【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。

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_ ClearFBError	读功能块错误		TEST:=SMC_ ClearFBError(pDrive:=ADR(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

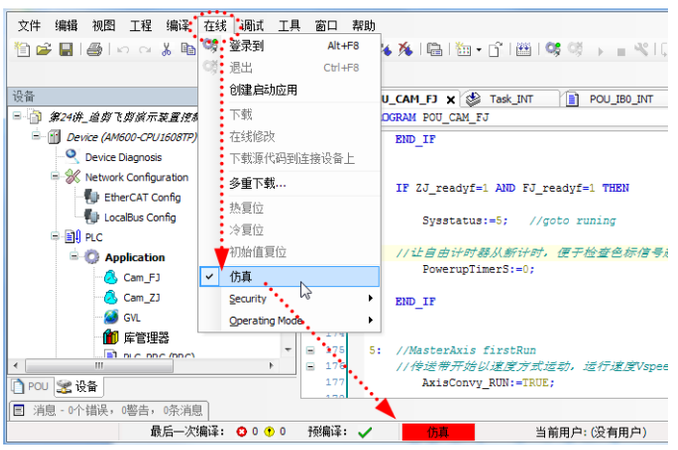
输出变量	名称	数据类型	有效范围	初始值	描述
SMC_ClearFBError	清除错误	BOOL	TRUE,FALSE	FALSE	True, 清除

第7章 仿真与调试

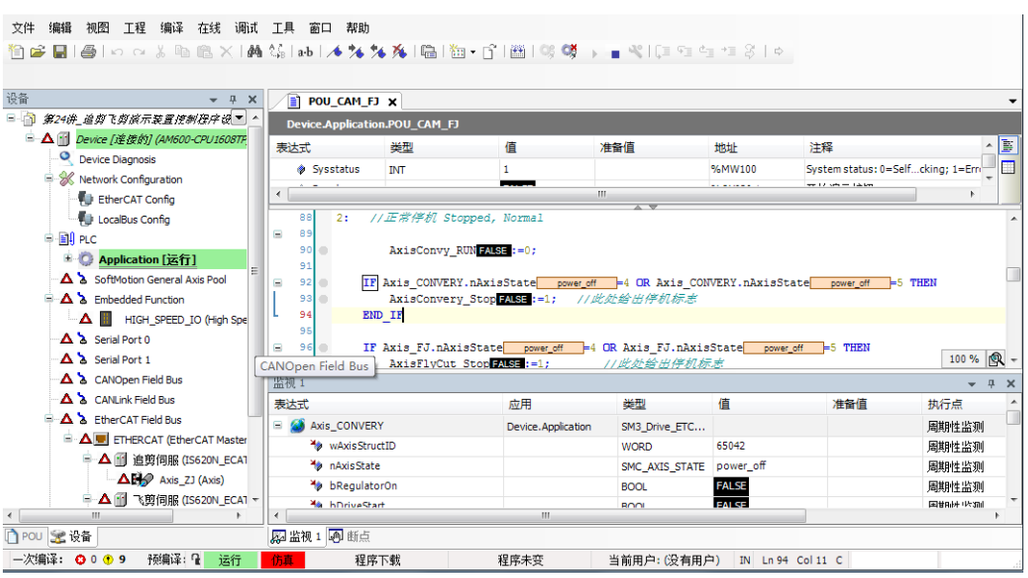
七、仿真与调试

7.1 仿真 AM600 控制器

用户在编程调试时，可能手头可能没有 AM600 控制器硬件，此时就可以使用 InoProShop 的仿真功能，可以调试用户程序的逻辑。开启仿真功能的方法如下图，在仿真状态，编程软件下方有红色字体的仿真状态提示。



在仿真状态，同样可以编译用户程序，“登录”到控制器，则只是将用户程序加载到 PC 的仿真器中，可以像实际接入了控制器一样进行用户程序的监控、强制修改参数的操作，观察用户程序的执行效果，如下图：

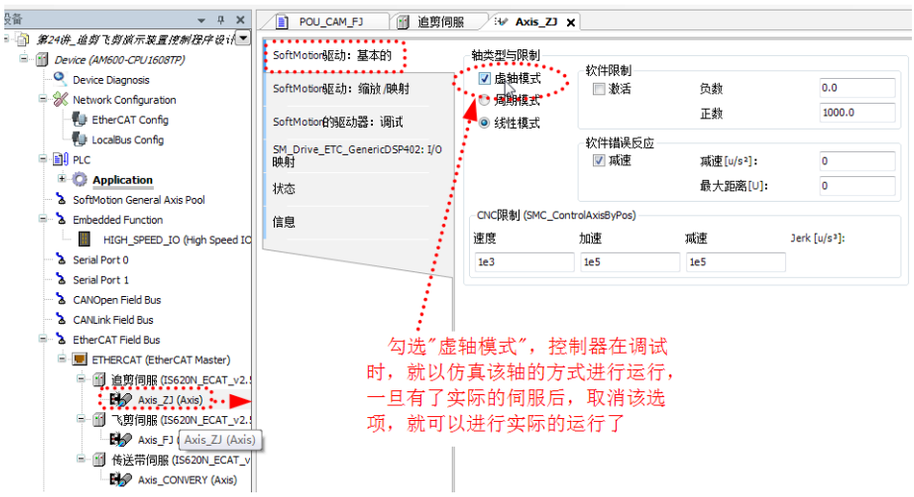


虽然不能对网络总线的运行进行仿真，但可以对伺服轴数据结构参数进行强制后，仍然可以观察程序的执行逻辑，检查程序的执行情况。

仿真监控调试程序的步骤与有 AM600 的情景相同，“登录”后，可以点击“运行”、“停止”执行用户程序，需要修改用户程序之前，要“退出”登录状态，等等。

7.2 仿真伺服驱动器

在编写调试 MC 运控应用程序时，编程人员手头有 AM600 控制器，但没有伺服驱动器，或没有足够数量的伺服驱动器，若要调试调试用户程序，可以用“虚轴”的方式，来代替伺服驱动器实轴，如下图的方法：



在编程调试时，若接入的伺服轴数量与用户程序中配置的数量不同，系统就会报警，不能正常调试，若接入了这个虚轴后，系统就不会报警，而是以软件仿真该伺服的方式运行。可以直观地看到该轴的“运行”状态，检验我们的 MC 控制程序的正确性。

虚轴也是轴。虽然是“虚轴”，但对该轴状态的操作逻辑仍然需按 PLCopen 规范中的状态转移逻辑进行编程设计，比如运行之前必需先运行 MC_Power，出现错误后需运行 MC_Reset 等等，这样便于我们调试和排除用户程序中的逻辑错误。

若接入了实际的伺服轴后，只需取消上图中对应轴的“虚轴模式”，即可正常运行了。

Memo NO. _____

Date / /

第8章 附 录

附录 A IS620N 支持的原点回归模式

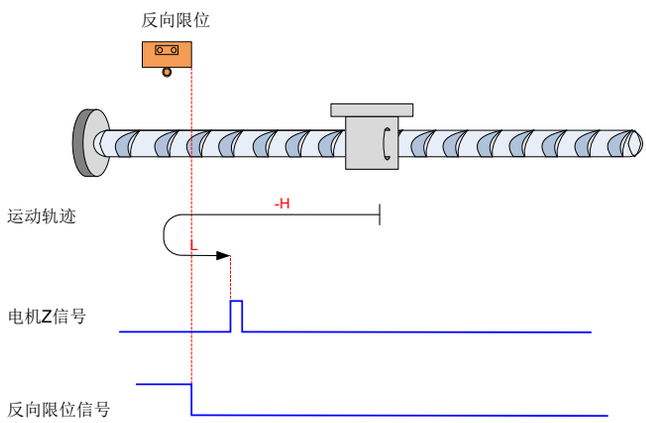
A.1 回零模式介绍：

1） 6098h=1

机械原点：电机 Z 信号

减速点：反向超程开关

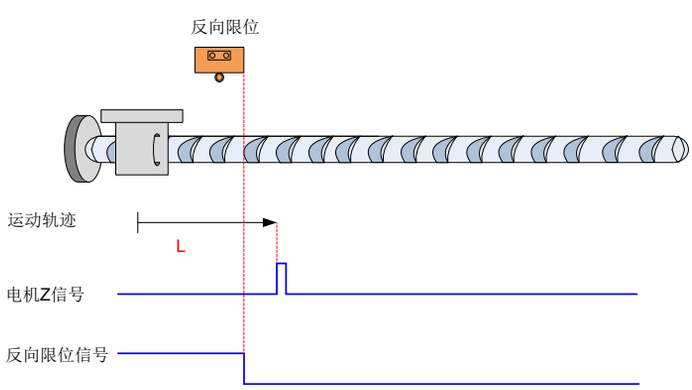
◆ 回零启动时减速点信号无效



注：图中“H”代表高速 6099-1h，“L”代表低速 6099-2h

开始回零时 N-OT=0，以反向高速开始回零，遇到 N-OT 上升沿后，减速，反向，正向低速运行，遇到 N-OT 下降沿后的第一个 Z 停机；

◆ 回零启动时减速点信号有效



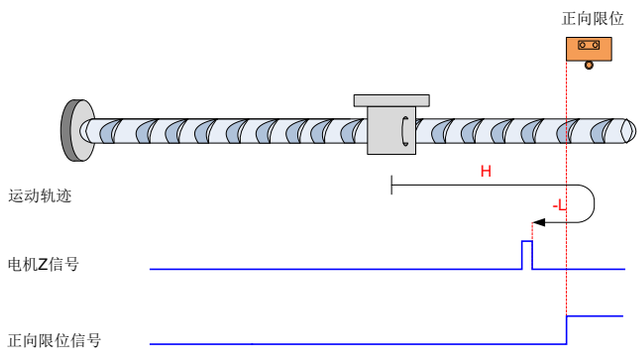
回零启动时 N-OT=1，直接正向低速开始回零，遇到 N-OT 下降沿后的第一个 Z 停机。

2） 6098h=2

原点：Z 信号

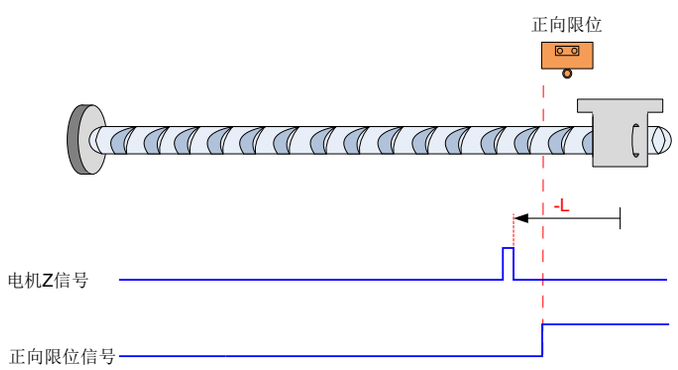
减速点：正向超程开关

◆ 回零启动时减速点信号无效



开始回零时 P-OT=0，以正向高速开始回零，遇到 P-OT 上升沿后，减速，反向，反向低速运行，遇到 P-OT 下降沿后的第一个 Z 停机；

◆ 回零启动时减速点信号有效



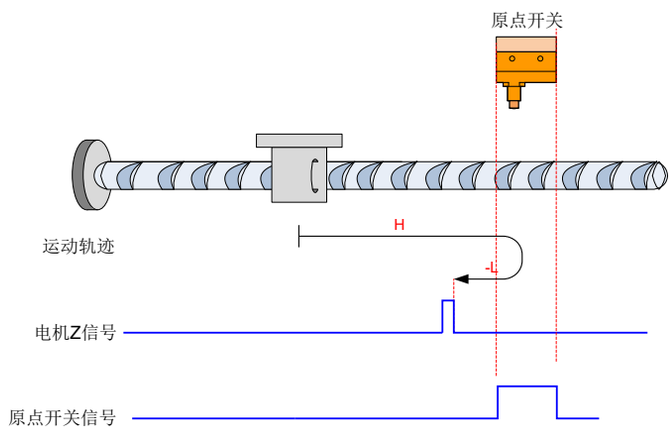
回零启动时 P-OT=1，直接反向低速开始回零，遇到 P-OT 下降沿后的第一个 Z 停机；

3） 6098h=3

原点：Z 信号

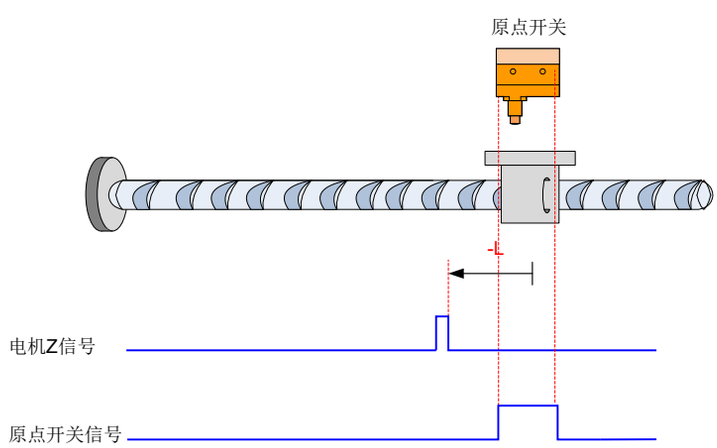
减速点：原点开关 (HW)

◆ 回零启动时减速点信号无效



开始回零时 HW=0，以正向高速开始回零，遇到 HW 上升沿后，减速，反向，反向低速运行，遇到 HW 下降沿后，继续运行，之后遇到第一个 Z 停机；

◆ 回零启动时减速点信号有效



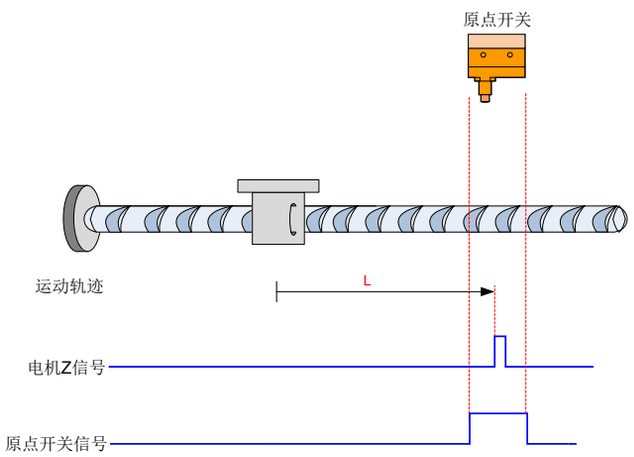
回零启动时 HW=1，直接反向低速开始回零，遇到 HW 下降沿后的第一个 Z 停机；

4） 6098 = 4

原点：Z 信号

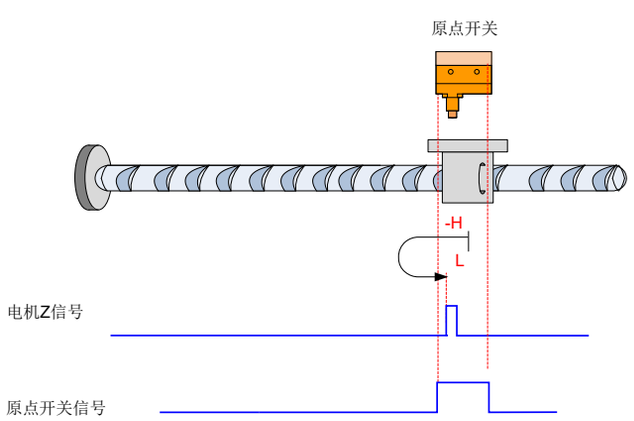
减速点：原点开关 (HW)

◆ 回零启动时减速点信号无效



开始回零时 HW=0，直接正向低速开始回零，遇到 HW 上升沿后第一个 Z 停机；

◆ 回零启动时减速点信号有效



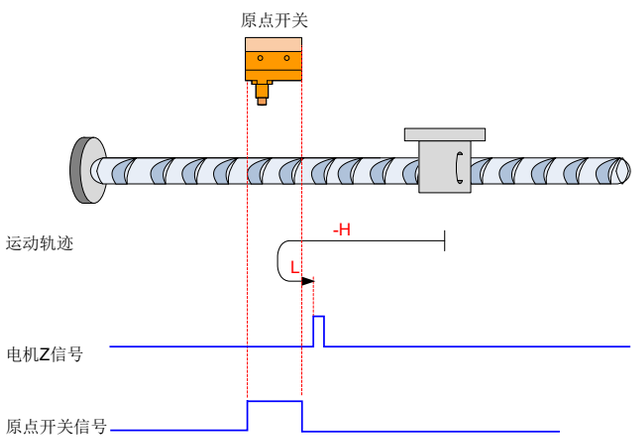
回零启动时 HW=1，以反向高速开始回零，遇到 HW 下降沿后，减速，反向，正向低速运行，遇到 HW 上升沿后的第一个 Z 停机；

5） 6098h=5

原点：Z 信号

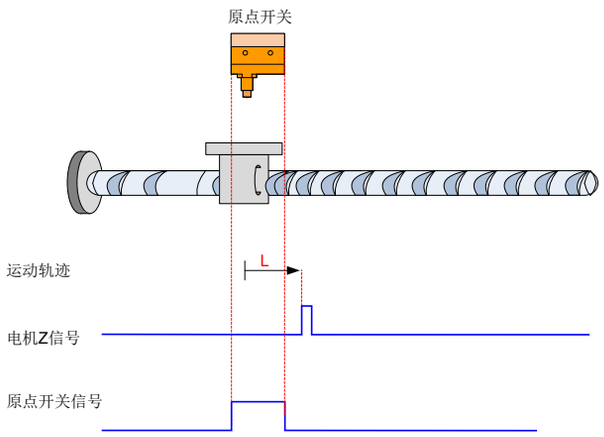
减速点：原点开关 (HW)

◆ 回零启动时减速点信号无效



开始回零时 HW=0，以反向高速开始回零，遇到 HW 上升沿后，减速，反向，正向低速运行，遇到 HW 下降沿后的第一个 Z 停机；

◆ 回零启动时减速点信号有效



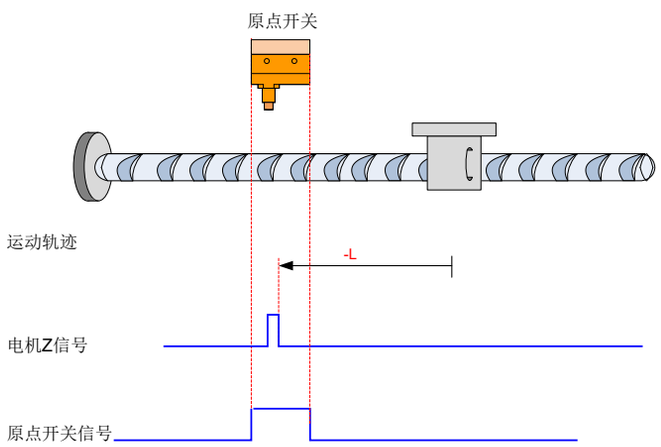
回零启动时 HW=1，则直接正向低速开始回零 , 遇到 HW 下降沿后的第一个 Z 停机；

6） 6098 =6

原点：Z 信号

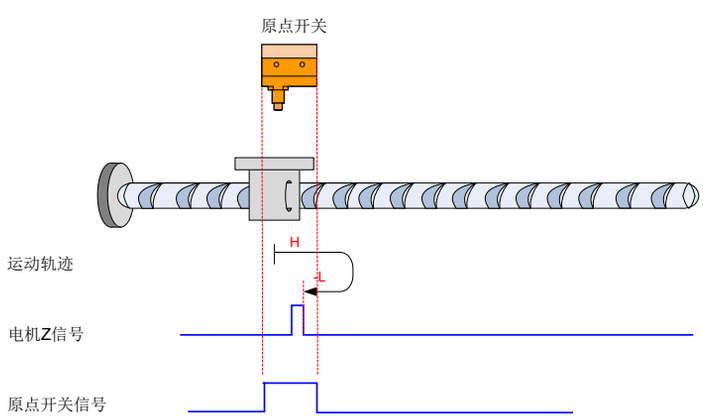
减速点：原点开关 (HW)

◆ 回零启动时减速点信号无效



开始回零时 HW=0，直接反向低速开始回零，遇到 HW 上升沿后第一个 Z 停机；

◆ 回零启动时减速点信号有效



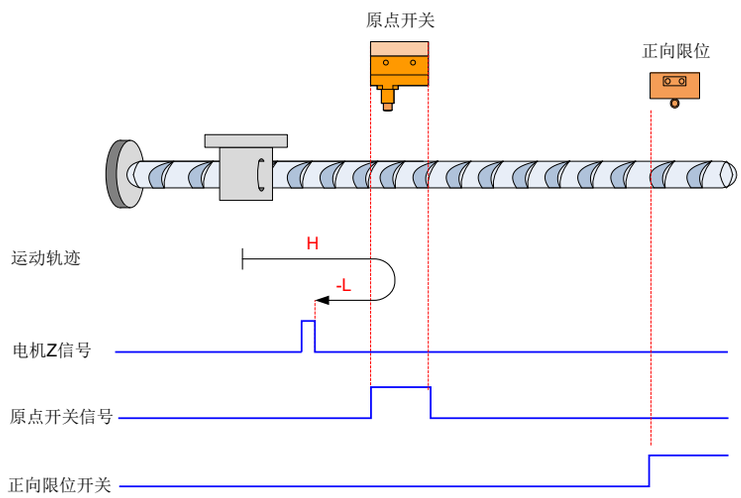
回零启动时 HW=1，以正向高速开始回零，遇到 HW 下降沿后，减速，反向，反向低速运行，遇到 HW 上升沿后的第一个 Z 停机；

7） 6098 = 7

原点：Z 信号

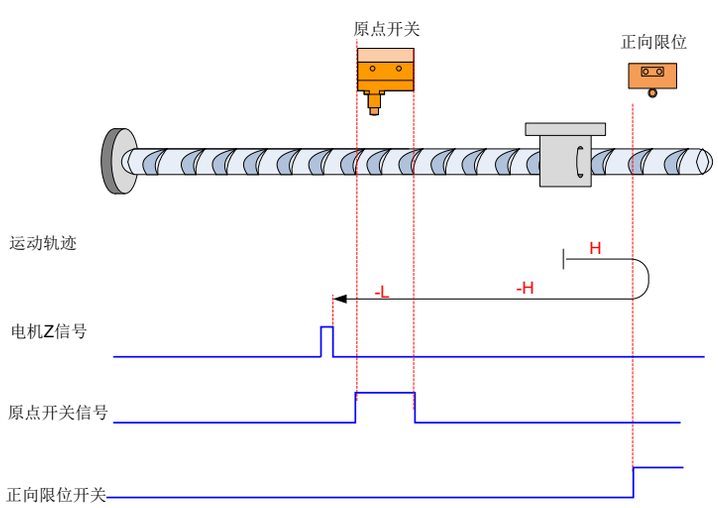
减速点：原点开关 (HW)

◆ 回零启动时减速点信号无效，未遇到正向限位开关



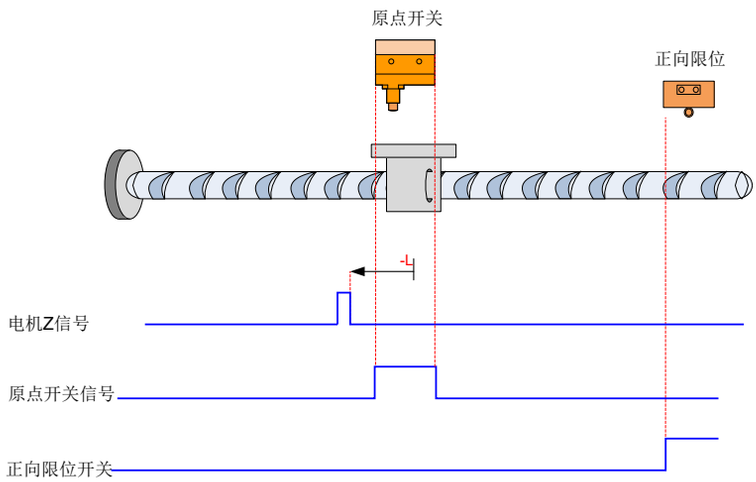
开始回零时 HW=0，以正向高速开始回零，若未遇到限位开关，遇到 HW 上升沿后，减速，反向，反向低速运行，遇到 HW 下降沿后的第一个 Z 停机；

◆ 回零启动时减速点信号无效，遇到正向限位开关



开始回零时 HW=0，以正向高速开始回零，若遇到限位开关，自动反向，反向高速运行，遇到 HW 上升沿后，减速，继续反向低速运行，遇到 HW 下降沿后的第一个 Z 停机；

◆ 回零启动时减速点信号有效



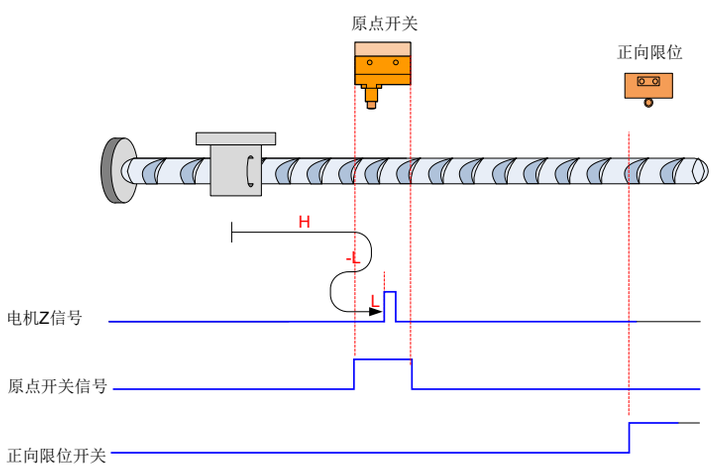
回零启动时 HW=1，则直接反向低速开始回零，遇到 HW 下降沿后的第一个 Z 停机；

8） 6098 =8

原点：Z 信号

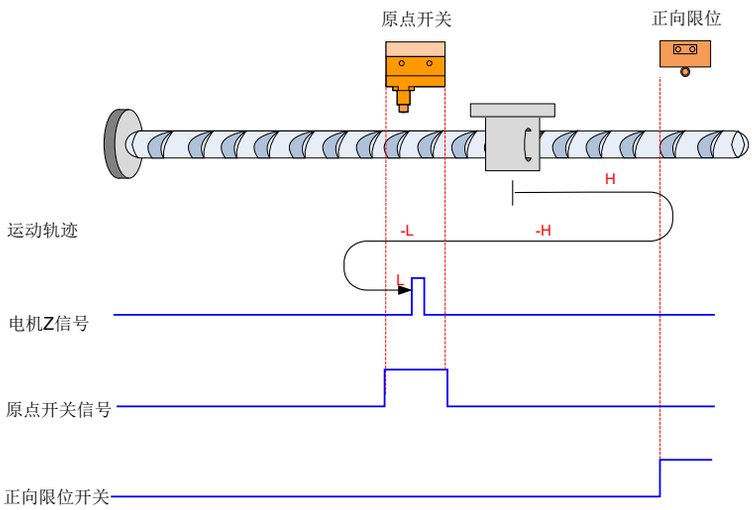
减速点：原点开关 (HW)

◆ 回零启动时减速点信号无效，未遇到正向限位开关



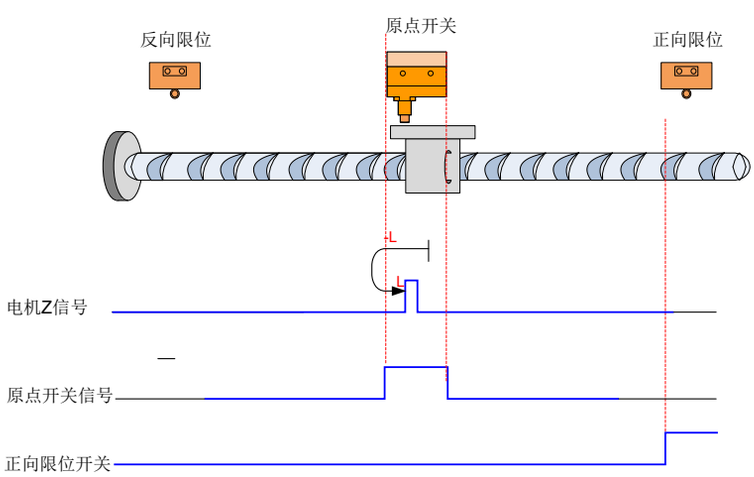
开始回零时 HW=0，以正向高速开始回零，若未遇到限位开关，遇到 HW 上升沿后，减速，反向，反向低速运行，遇到 HW 下降沿后，反向，正向低速运行，遇到 HW 上升沿后的第一个 Z 停机；

◆ 回零启动时减速点信号无效，遇到正向限位开关



开始回零时 HW=0，以正向高速开始回零，若遇到限位开关，自动反向，反向高速运行，遇到 HW 上升沿后，减速，反向低速运行，遇到 HW 下降沿后，反向，正向低速，遇到 HW 上升沿后的第一个 Z 停机；

◆ 回零启动时减速点信号有效



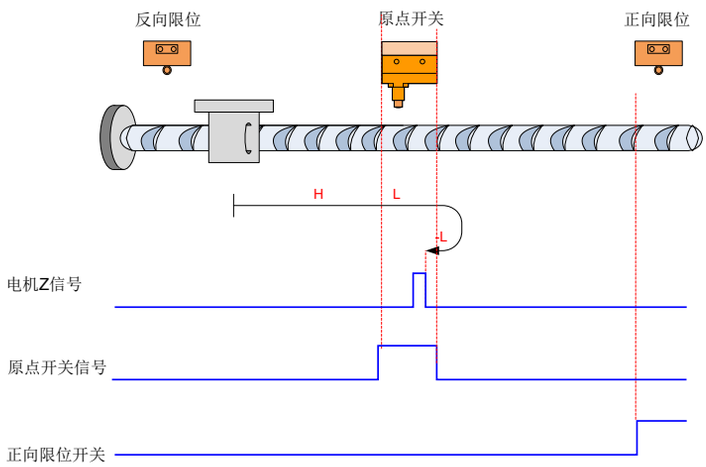
回零启动时 HW=1，则直接反向低速开始回零，遇到 HW 下降沿后，反向，正向低速，遇到 HW 上升沿后的第一个 Z 停机；

9） 6098 =9

原点：Z 信号

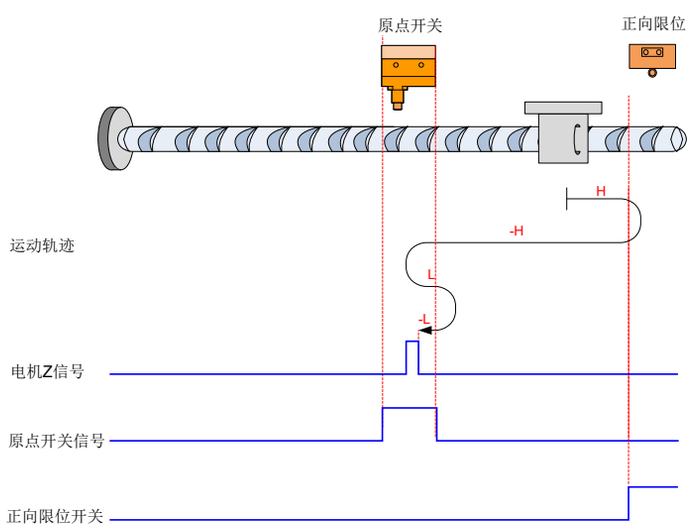
减速点：原点开关 (HW)

◆ 回零启动时减速点信号无效，未遇到正向限位开关



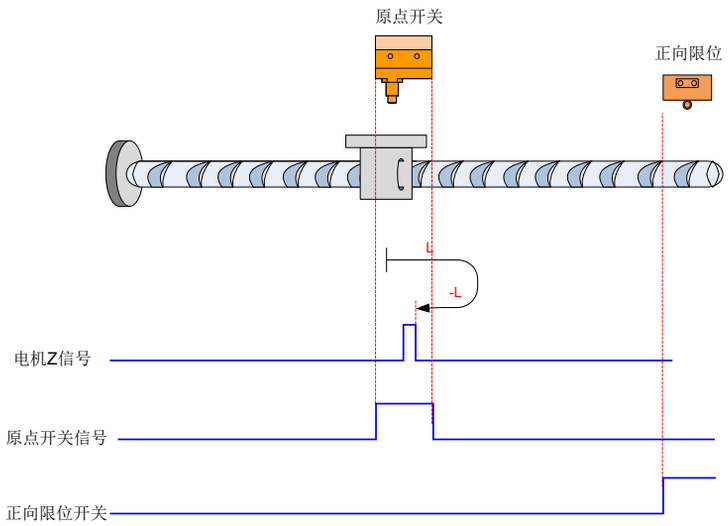
开始回零时 HW=0，以正向高速开始回零，若未遇到限位开关，遇到 HW 上升沿后，减速，正向低速运行，遇到 HW 下降沿后，反向，反向低速运行，遇到 HW 上升沿后的第一个 Z 停机；

◆ 回零启动时减速点信号无效，遇到正向限位开关



开始回零时 HW=0，以正向高速开始回零，若遇到限位开关，自动反向，反向高速运行，遇到 HW 上升沿后，减速反向即恢复正向运行，正向低速遇到 HW 下降沿后，反向，反向低速运行中遇到 HW 上升沿后的第一个 Z 停机；

◆ 回零启动时减速点信号有效



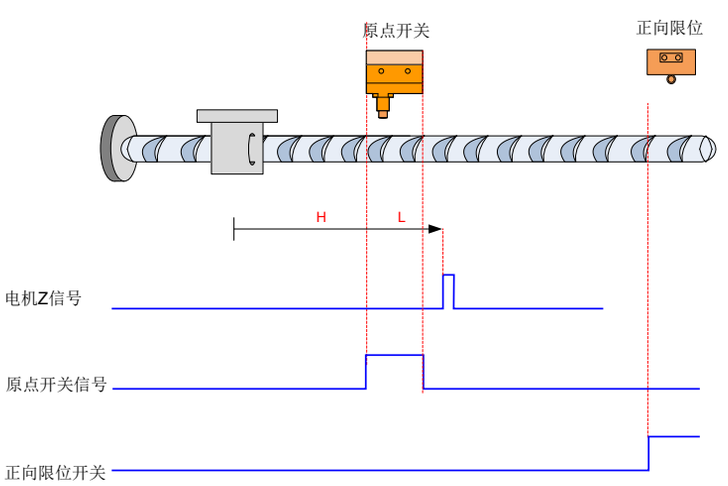
回零启动时 HW=1，则直接正向低速开始回零，遇到 HW 下降沿后，反向，反向低速运行中，遇到 HW 上升沿后的第一个 Z 停机。

10） 6098 =10

原点：Z 信号

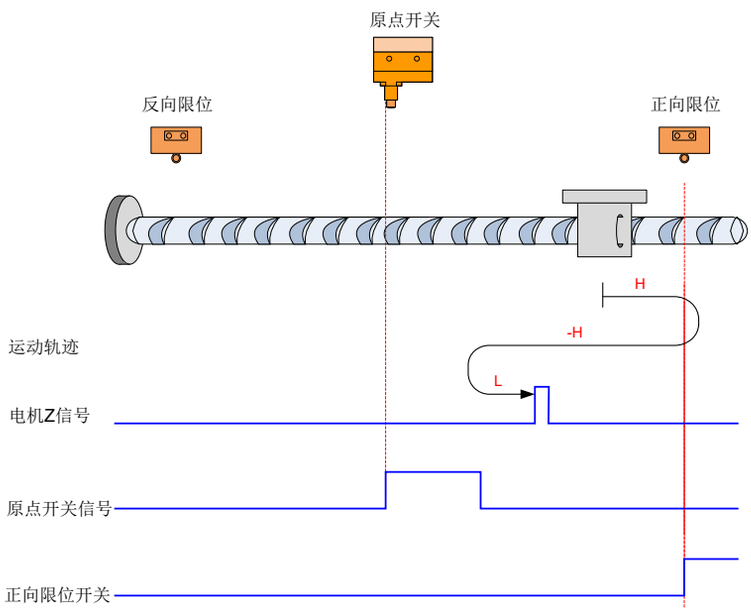
减速点：原点开关 (HW)

◆ 回零启动时减速点信号无效，未遇到正向限位开关



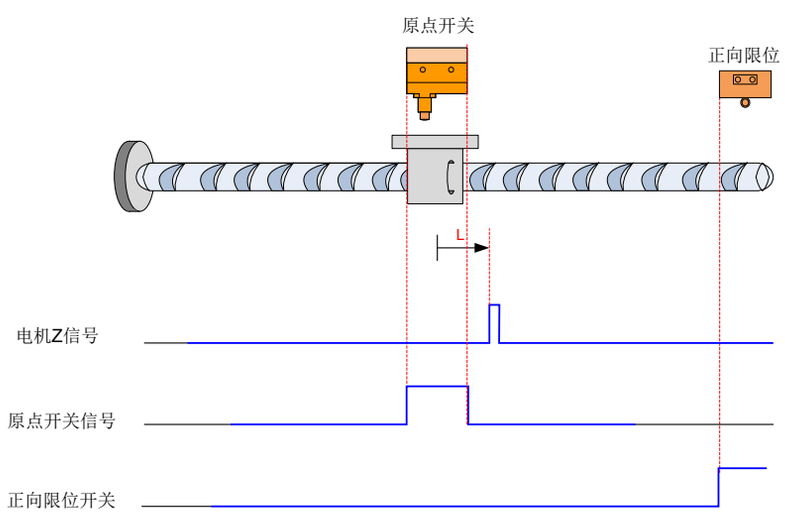
开始回零时 HW=0，以正向高速开始回零，若未遇到限位开关，遇到 HW 上升沿后，减速，正向低速运行，遇到 HW 下降沿后，继续正向低速运行，之后遇到的第一个 Z 停机；

◆ 回零启动时减速点信号无效，遇到正向限位开关



开始回零时 HW=0，以正向高速开始回零，若遇到限位开关，自动反向，反向高速运行，遇到 HW 上升沿后，减速反向即恢复正向运行，正向低速遇到 HW 下降沿后的第一个 Z 停机；

◆ 回零启动时减速点信号有效



回零启动时 HW=1，则直接正向低速开始回零，遇到 HW 下降沿后的第一个 Z 停机；

11）6098h=11&12&13&14

与 6098 = 7~10 运动曲线相似，仅初始运行方向相反。

12） 6098h=17~30

与 6098=1~14 运动曲线相同，仅最后一步找 Z 信号的步骤省去。遇到以下原点信号立即停机。

回零方式 6098	原点信号
17	N-OT 下降沿
18	P-OT 下降沿
19	HW 下降沿
20	HW 上升沿
21	HW 下降沿
22	HW 上升沿
23	HW 下降沿
24	HW 上升沿
25	HW 上升沿
26	HW 下降沿
27	HW 下降沿
28	HW 上升沿
29	HW 上升沿
30	HW 下降沿

13） 6098h=31~32

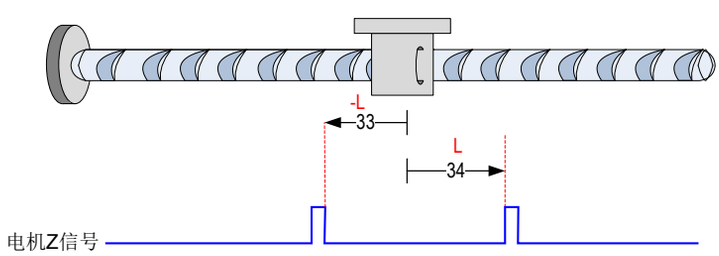
标准 402 协议中未定义此模式，可用于扩展。

14） 6098h=33 和 34

原点：Z 信号

减速点：无

- 回零方式 33：反向低速运行，遇到的第一个 Z 信号停机
- 回零方式 34：正向低速运行，遇到的第一个 Z 信号停机



15） 6098h=35

回零方式 35，以当前位置为机械原点，触发原点回零后 (6040 控制字：0x0F→0x1F)，用户当前位置 6064 = 607C

附录 B IS620N 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 IS620 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。								
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								
6060	00	伺服模式选择	RW	INT8	-	0~10	0	RPDO
1- 轮廓位置模式 (pp) 3- 轮廓速度模式 (pp) 4- 轮廓转矩模式 (pp) 6- 原点回零模式 (hm) 8- 周期同步位置模式 (csp) 9- 周期同步速度模式 (csv) 10- 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								
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。								
606C	00	实际速度	RO	INT32	指令单位 /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。								
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。								

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
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								
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								
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							
	00	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停止。								
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	UINT32	指令单位 /s	0~232-1	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								
6081	00	轮廓运行速度	RW	UINT32	用户速度单位	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

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
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								
60FC	00	位置指令	RO	INT32	编码器单位	-231-(231-1)	0	TPDO
位置指令，编码器单位								
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：记录了运动控制功能块返回的错误序号。

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 fSWLimitPositive 和 fSWLimitNegative 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_ACTIVE_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_VELOCITY_EXCEEDED	达到最大速度 (fMaxVelocity)
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_EXCEEDED	达到最大减速度 (fMaxDeceleration)
50	SMC_Homing	SMC_3SH_INVALID_VELACC_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_DRIVE_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_DRIVE_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_DRIVE_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

错误代码	产生源	变量名称	错误原因描述
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 ReadDriveParameter 的错误 (SM_DriveBasic.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 WriteDriveParameter 的错误 (Drive_Basic.lib)
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_MoveAbsolute	SMC_MA_INVALID_VELACC_VALUES	不合适的速度或者加速度值
202	MC_MoveAbsolute	SMC_MA_INVALID_DIRECTION	方向错误
226	MC_MoveRelative	SMC_MR_INVALID_VELACC_VALUES	不合适的速度或者加速度值
227	MC_MoveRelative	SMC_MR_INVALID_DIRECTION	方向错误
251	MC_MoveAdditive	SMC_MAD_INVALID_VELACC_VALUES	不合适的速度或者加速度值
252	MC_MoveAdditive	SMC_MAD_INVALID_DIRECTION	方向错误
276	MC_MoveSuperImposed	SMC_MSI_INVALID_VELACC_VALUES	不合适的速度或者加速度值
277	MC_MoveSuperImposed	SMC_MSI_INVALID_DIRECTION	方向错误
301	MC_MoveVelocity	SMC_MV_INVALID_ACCDEC_VALUES	不合适的速度或者加速度值
302	MC_MoveVelocity	SMC_MV_DIRECTION_NOT_APPLICABLE	Direction=shortest/fastest 不支持
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_VelocityProfile	SMC_VP_ARRAYSIZE	错误排列尺寸
351	MC_VelocityProfile	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_MoveContinuousRelative	SMC_MCR_INVALID_VELACC_VALUES	不合适的速度或者加速度值
427	SMC_MoveContinuousRelative	SMC_MCR_INVALID_DIRECTION	方向错误
451	SMC_MoveContinuousAbsolute	SMC_MCA_INVALID_VELACC_VALUES	不合适的速度或者加速度值
452	SMC_MoveContinuousAbsolute	SMC_MCA_INVALID_DIRECTION	方向错误
453	SMC_MoveContinuousAbsolute	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

错误代码	产生源	变量名称	错误原因描述
626	MC_CamIN	SMC_CI_MASTER_OUT_OF_SCALE	主轴超出范围
627	MC_CamIN	SMC_CI_RAMPIN_NEEDS_VELACC_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_VELACCDEC	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_COVER_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_VEL_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 Vel_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 比路径编译模块处理速度快
1004	SMC_Interpolator	SMC_INT_VEL_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_CHECKVELOCITIES	Input-OutQueue DataIn 没有作为 SMC_CHeckVelocities 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_VEL_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_CheckVelocities	SMC_CV_ACC_DEC_VEL_NONPOSITIVE	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 fGapVelocity / 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
1301	SMC_GCodeViewer	SMC_GCV_BUFFER_WRONG_TYPE	缓冲区元素类型错误
1302	SMC_GCodeViewer	SMC_GCV_UNKNOWN_IPO_LINE	当前插补行不能被找到
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

错误代码	产生源	变量名称	错误原因描述
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_OVERRUN	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
2202	SMC_ReadCAM	SMC_RCAM_WRONG_COMPILE_TYPE	错误编译模式
2203	SMC_ReadCAM	SMC_RCAM_WRONG_VERSION	文件版本错误
2204	SMC_ReadCAM	SMC_RCAM_UNEXPECTED_EOF	未知的文件结尾
3001	SMC_ WriteDriveParamsToFile	SMC_WDPF_CHANNEL_OCCUPIED	SMC_WDPF_TIMEOUT_PREPARING_LIST
3002	SMC_ WriteDriveParamsToFile	SMC_WDPF_CANNOT_CREATE_FILE	文件不能被创建
3003	SMC_ WriteDriveParamsToFile	SMC_WDPF_ERROR_WHEN_READING_PARAMS	读取文件参数的时候错误
3004	SMC_ WriteDriveParamsToFile	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

销售服务联络地址

深圳市汇川技术股份有限公司
Shenzhen Inovance Technology Co., Ltd.
地址：深圳市宝安区宝城70区留仙二路鸿威工业区E栋
总机：(0755)2979 9595
传真：(0755)2961 9897
http://www.inovance.com

苏州汇川技术有限公司
Suzhou Inovance Technology Co., Ltd.
地址：苏州市吴中区越溪友翔路16号
总机：(0512)6637 6666
传真：(0512)6285 6720
http://www.inovance.com

19010539A02