techservo

CVM 索引器编程手册

-适用 AP 系列驱动器配置的 MCK 软件

 深圳市泰科智能智能伺服技术有限公司

版权说明

本文档仅供用户参考，文档中的内容力图精确和可靠，但错误和疏忽之处在所难免，如果您发现错误，请不吝赐教。泰科智能智能保留随时修改和完善本文档的权利，有疑问请咨询我们，谢谢。

版次	发布时间	修订内容	修订前	修订后
V1.0	2014-5-5	全面修改		

目录

1.关于该手册

本手册描述了泰科智能虚拟机（CVM）中的**索引器程序**（英文名字的直译，即 Indexer 2 Program）使用说明。本手册适合对运动控制理论和操作，泰科智能驱动器及泰科智能 CME2 软件有一定基础知识的人使用。

备注：光电式增量编码器上的索引脉冲（index pulse），又叫“Z”或者“标志脉冲”，是编码器每转一圈产生的一个数字脉冲信号，主要用做回零点或验证增量信号计数是否正确。通常是光电式增量编码器顶端的标志和轴上的标志对齐时就会产生一个索引脉冲。索引脉冲也标志着换相中 U 通道的上升沿，该上升沿对确定无刷直流电机的大概初始时间很重要。

2.泰科智能索引器程序

泰科智能索引器程序和泰科智能驱动器一起，可以由对计算机和运动控制知识有一定基础的人编程使用，从而实现强大的单轴或多轴控制。利用 PC 和泰科智能 MCK 软件中内置的工具，用户可以对索引器程序进行配置和编程，然后将其下载到驱动器中。在驱动器中，索引器程序是运行在虚拟机(CVM)上的，虚拟机是一个嵌入的虚拟可编程控制器。

用户可以创建多达 32 个序列（sequence）。每个序列包含一个或多个步骤，每个步骤可以是回零指令，运动，增益调整和时间延时的组合。序列的步骤也可编程为条件转跳，驱动器数字输出控制，和输入检测。

在最简单的应用中，可以使用一个 PLC 或开关使驱动器的相连的数字输入有效，以选择并执行指定的序列。驱动器的数字输出可以用来控制机械动作或者给 PLC 提供状态反馈。

在更复杂的应用中，索引器程序有 32 个寄存器，它们可以用来控制程序。通过驱动器的 RS232 串口，可以发送 ASCII 指令来获取或修改这些寄存器。这些寄存器可以用来选择并执行序列，然后给索引器程序传递数字参数。这些寄存器还可以由驱动器支持的其它网络如 CANopen 或 DeviceNet 来存取。

泰科智能驱动器的多点特性也支持多轴应用。多点配置中，通过串口和外部控制器相连的驱动器，是 CAN 总线网络中其它多个驱动器的网关。这样，一个 PLC 通过串口可以控制多大 128 个轴。

此外，一些型号的泰科智能驱动器可以驱动三个独立的轴。这些驱动器中，特定的索引器程序函数可以对各个轴独立编程。具体参考多轴驱动器支持。

2.1. 程序特性

索引器程序的特性包括：

- 简单直观的编程工具
- 32 个可编程多步骤序列
- 优先序列，由单个数字收入选择并执行
- 32 个寄存器，可通过 RS-232 串口或其它界面存取

- 数字输入或寄存器选择并执行指定序列
- 程序可上电自动运行
- 序列错误可编程响应
- 标准函数包括：

• 运动（位置,速度,电流,凸轮,回零）
• 设置增益，限位和跟踪窗口
• 数学函数
• 设置允许模式
• 等待（输入,延时,位置,运动完成,参数或事件）
• 设置输出，位置触发的输出
• 设置，获取参数
• 逻辑运算
• 条件转跳
• 驱动器停止工作
• 速度一阶滤波

3.描述和操作

本节描述了索引器程序的工作原理。

3.1. 程序特性

启动后，索引器程序会清空所有的程序寄存器并进入主循环，首先不断检测用户自定义的优先输入（如果已配置）的状态，然后检测自定义的 Go 数字输入状态或寄存器的值。可以将一个序列选择为上电或驱动器复位后直接执行。如果优先输入或者 Go 命令有效，程序即会执行相应的序列。程序会自动将驱动器设置为相应的工作模式，顺序执行序列中的各步。序列执行过程中，忽略优先输入，Go 输入和序列选择数字输入的状态。成功执行完一个序列后，程序返回到主循环。
序列执行过程中，如果发生错误，可以从以下两种处理方法中选择一个进行处理：

- 放弃当前序列的执行，返回到主循环，或者
- 程序立即执行另外一个序列（即直接转跳）。执行完后再返回主循环

3.2. 序列选择

索引器程序可以存储多达 32 个序列（0-31）。接收到一个 Go 命令后，程序开始执行由寄存器或数字输入选择的序列。

3.2.1. 使用寄存器选择序列

当配置为使用寄存器选择序列的方法时，程序接收到一个 Go 命令后，就读取特殊寄存器前 5 位（0-4）来选择序列。因此，要选择指定的序列，请向寄存器写入一个十六进制的值（0x00 – 0x1f）或十进制的值（0 – 31）。要使用 ASCII 命令向一个寄存器写值，请参考寄存器读写。

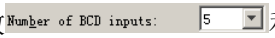
3.2.2. 使用数字输入选择序列

使用数字输入可能会产生冲突

使用 CME 2 主界面中的 Input/Output 按钮，打开输入输出界面，配置数字输入功能的时候，可能会和索引器程序中 BCD 序列选择、优先和 Go 数字输入口的指定产生冲突。一个输入对应的两个功能会同时起作用，可能产生意想不到的结果。请谨慎操作，避免在指定输入端口功能的时候产生冲突。

忽略此警告可能会导致设备损坏，人身受伤，或死亡。

当索引器程序接收到 Go 指令，并且设定的是**使用数字输入**选择序列的方法，驱动器就会根据数字输入的状态选择相应的序列去执行。最多可以使用 5 个输入端口来表示 0-31 的数，对应 32 个序列号。

下表给出了选择输入的个数 

和可选择的范围：

BCD 使用的输入的个数	可表示的序列数量	序列范围
5	32	0-31
4	16	0-15
3	8	0-7
2	4	0-3
1	2	0-1
0	1	0

例如，假设程序配置为使用三个数字输入来选择序列，输入口从 IN2 开始，下表给出了每个组合可选择的序列号：

	输入口		
输入号	IN4	IN3	IN2
等价的十进制数	4	2	1
选择的序列号	输入状态		
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0

3.3. Go 命令

Go 命令会使程序开始执行选择的序列。Go 命令可编程的选项包括寄存器，数字输入和启动或复位后立即执行单个 Go 命令。

3.3.1. 使用寄存器初始化 Go

当编程为**使用寄存器初始化 Go（Use Register to Initiate Go）**时，程序监测 Go 寄存器的第 15 位。如果该位为 1，则程序执行选择的序列。

当计划由数字输入选择序列，且 Go 命令由寄存器触发时，请通过向 Go 寄存器写入十六进制数 0x8000 或者十进制数 32768 来置位 Go 寄存器的第 15 位（其它位的数直接忽略）。要使用 ASCII 命令向寄存器写值，请参考寄存器读写一节。

备注：Go 寄存器其实是用来触发 Go 命令的 Rn（n 是 0-31 的正整数）寄存器。例如由 R0 触发 Go 命令，数字输入选择序列时，先设置好输入状态，再发送写寄存器命令：
i r0 0x8000<回车> 就会转入到相应的序列中（如果原序列中正在执行的函数中没有或者未选择 wait move done 等类似条件）。
当 Go 命令和序列选择同时使用寄存器编程时，同一个寄存器产生的效果相同。这种情况下，用相同的写操作将序列号码写到寄存器中，并同时将第 15 位置 1.例如，要执行序列 12，向寄存器中写入十六进制数 0x800c 或者等价的十进制数 32780.

注意: 在执行序列前，索引器程序会清零第 15 位，以便在第 15 位在下次置位前，寄存器或数字输入不会触发另外一个 Go 命令。（第 15 位的状态不会影响**立即单个 Go 命令**（Immediate Single Go）的操作）

3.3.2. 使用数字输入初始化 Go

当编程为**使用数字输入初始化 Go（Use Input to Initiate Go）**时，程序监控编程的 Go 输入端口的状态。可将其编程为电平触发（高或低电平），或者输入电平跳变触发（上升沿或下降沿）。

3.3.3. 启动或复位时使用立即执行单个 Go（Immediate Single Go）命令

该选项编程驱动器在上电/复位/按下运行按钮时触发 Go 命令。具体执行哪个序列由编程确定。注意该处的启动（Startup）指的是 CVM 界面中工具栏中的启动按钮  。与控制面板（Control Panel）中的使能（Enable）按钮无关。使能按钮用来执行 CME 中的程序，如果程序编程为使用寄存器选择序列，则会执行序号为 0 的序列，因为程序刚开始的时候所有寄存器的值都是 0，如果程序编程为使用数字输入选择序列，数字输入的状态决定了哪个序列被执行。立即单个 Go 命令执行后，才会根据条件执行编程的其它 Go 命令。

3.4. 优先输入命令

除了 32 个正常的序列外，索引器程序还支持一个优先序列（Priority sequence）。与其它序列不同，优先序列可以：

- 由单个数字输入选择和初始化
- 有更高的处理优先级——相对其它序列选择方法

注意：优先序列并不会停止正在执行的序列，也不会覆盖任何序列。

3.5. 索引器程序寄存器

索引器程序有 32 个寄存器，可以用来选择序列，初始化 Go 命令，向序列各步中的函数传递数值。每个寄存器的长度都是 32 位。

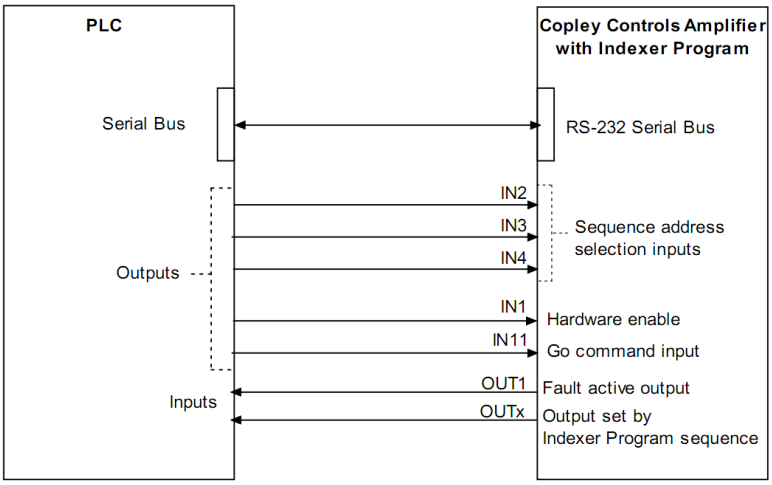
寄存器可以用来向大多数函数传递数值参数，例如增益和轨迹设置等。支持的协议包括泰科智能 ASCII 接口，CANopen 和 DeviceNet。

做调试或简单配置及控制，可以使用 ASCII 编程工具如串口调试助手实现，也可以使用 Windows 操作系统中自带的远程终端 Telnet 来实现。请参考串口 ASCII 命令。

要查看当前寄存器的值，可以在索引器程序界面中的点击菜单 View->Register Values 查看。

3.6. 典型连接框图

典型的程序应用中，驱动器和 PLC 的连接如下图所示：



这里，IN2-IN4 用来进行序列选择，IN11 用来初始化 Go 命令。IN1 是驱动器的硬件使能输入，由 PLC 控制。OUT1 配置为当驱动器发生错误时输出有效电平，并且索引器程序中的每一步都可以置位任一数字输出。串行连接总线可以用来设置寄存器的值。

4.编程

本章节介绍如何对索引器程序进行配置。

4.1. 步骤概述

创建和执行一个序列的典型步骤如下表所示：

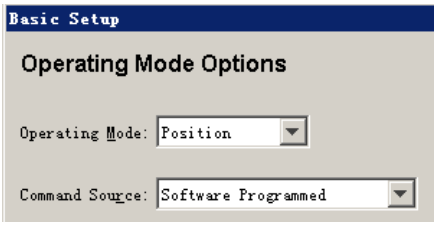
基本步骤	详细描述
设置驱动器从 CVM 程序执行	参考驱动器基本配置
设置，整定和测试驱动器	执行程序前请确保驱动器各参数设置合理
保存驱动器设置到 flash 中	Flash 中保存一些默认值，设置更新后需保存
打开索引器程序	参考下面小节所示
配置序列选择，优先级，Go 命令	参考下面小节所示
编程序列	参考下面小节所示
保存索引器程序到 flash 中	保存索引器程序到 flash 中才可运行 
保存索引器程序到磁盘	保存索引器程序到电脑中，方便以后调用 
确保驱动器硬件使能状态	参考 CME2 用户手册
运行索引器程序	参考工具栏功能
选择合适的序列	设置输入或给寄存器赋值
激活 Go 命令	设置输入或给寄存器赋值
根据需要对索引器程序进行调试	参考下面小节
停止索引器程序运行	如果需要可选择停止控制程序命令。位于工具栏中 注意：执行中的运动不会停止直到完成
如果需要，配置程序为自动启动	参考下面小节
保存最终的程序到驱动器中	参考下面小节
改变程序后重新保存	根据需要保存到驱动器中或电脑中

4.2. 驱动器基本设置

4.2.1. 典型的驱动器基本设置

典型的应用中，驱动器设置为工作在位置控制模式，命令源为软件编程，步骤如下

- 双击打开 CME2 软件，弹出的 **F12 禁止驱动器运行**对话框中点击 OK，进入主界面。
- 点击左上角工具栏中的第一个工具——基本设置（Basic Setup），然后选择左下角的修改设置。根据自己的配置进行合适的选择，逐步设置，直到操作模式选项中，按如下所示设置：
- 继续下一步，直到设置完成。

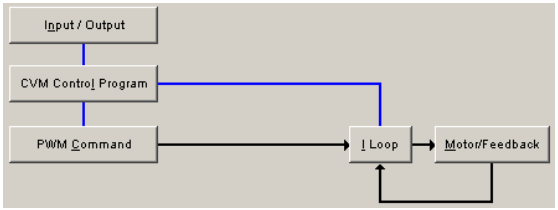


- 确保驱动器和电机的参数设置，各环整定及测试都准确无误。具体参考软件用户手册。

4.2.2. 使用索引器程序配置为其他模式和命令源

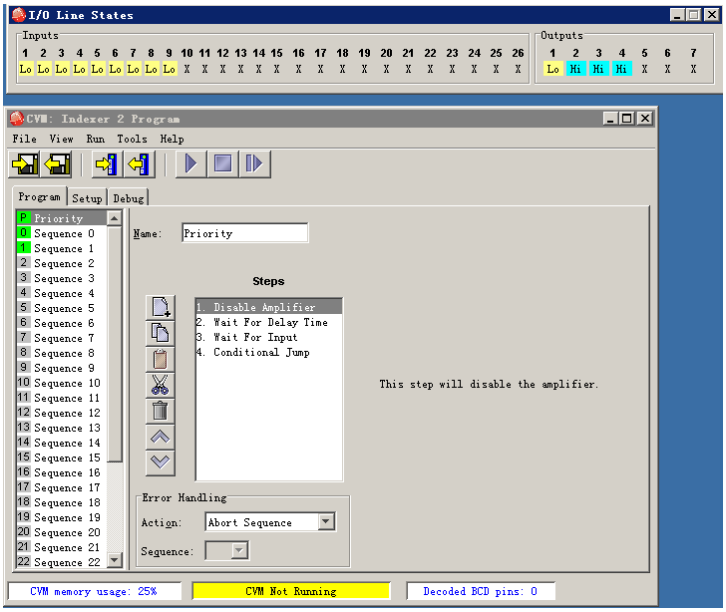
在任一操作模式/命令源下，索引器程序的设置都是有效的。同样，任意工作模式/命令源下，程序序列都是可以加载的。

例如，驱动器配置为：命令源为外部 PWM 输入，系统工作在电流环控制模式。这种情况下，可以通过 CME2 主界面中的 CVM Control Program 按钮打开索引器程序。



4.3. 打开索引器程序

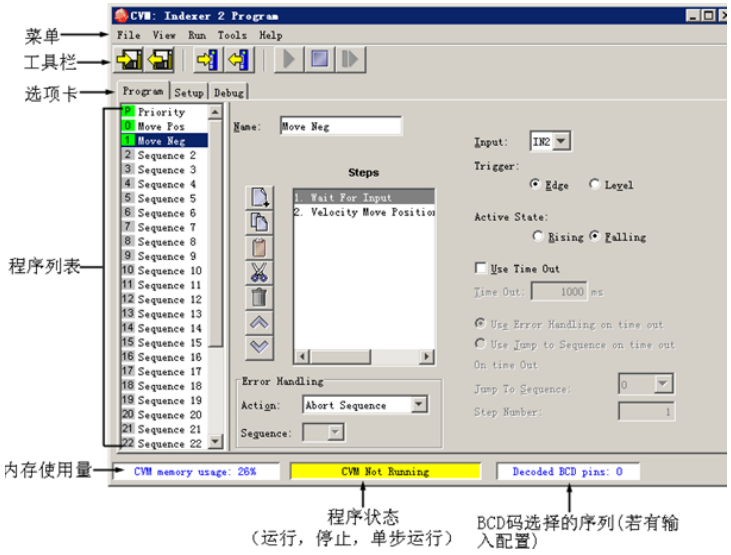
点击 CME2 主界面中的 CVM Control Program 按钮打开虚拟机：索引器程序和 I/O 状态界面，如下图所示：



驱动器的 flash 中保存的程序会自动呈现出来。有编程的序列会在左边的序列列表中以绿色背景显示，如上图所示。I/O 界面中则会实时显示各数字输入输出口的状态。

4.4. 索引器程序界面概述

索引器程序主界面的主要部分如下图所示：



其中的程序列表用来创建和修改序列。其它在后面会逐一介绍。

4.5. 菜单

多数的菜单中的功能，都可以通过工具栏中相应的工具来实现。用户请自行查看各菜单功能。

注意：菜单 Run 下面有 Enable Control program on Startup 和 Disable Control program on Startup 两个选项，决定着上电或复位后程序从哪里开始执行。如果设置为 Enable，则上电或复位后直接执行 CVM 中的程序；如果设置为 Disable，则不自动运行程序。例如驱动器设置为凸轮控制，要想上电后直接执行主界面 Camming 按钮下的凸轮程序（假设 CVM 不是用户想要的执行的程序），就必须在 CVM 中设置 Disable Control program on Startup。

4.6. 工具栏功能

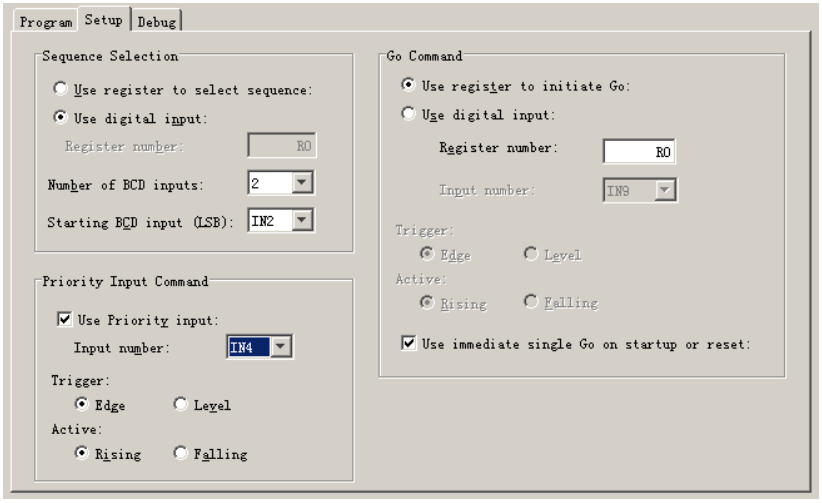
工具栏共有 7 个工具，方便用户操作：

其各工具的介绍如下表所示：

图标	工具	描述
	保持程序到电脑中	保存当前的 CVM 程序到电脑硬盘中，扩展名为.ccp
	从电脑硬盘中导出到 CVM 程序	打开一个保存的 CVM 程序。导入进来的程序会自动覆盖当前的程序*
	程序保存到驱动器中	将设置好的 CVM 程序保存到驱动器中，以便开始执行
	从驱动器中读取程序	读取驱动器中保存的 CVM 程序，覆盖当前值
	运行控制程序	开始执行程序 警告：根据配置及输入状态，运动可能会立刻执行 注意：如果程序更改后但未保存到驱动器中，该按钮会不可用。
	停止控制程序	停止程序的执行 警告：正在执行中的运动不会停止，直到运动完成
	单步控制	开始单步执行，并打开 Debug 选项，进行调试
*注意：当控制程序是从磁盘中恢复的，Enable Control program on Startup 会被自动选择		

4.7. 设置序列选择和 Go 命令

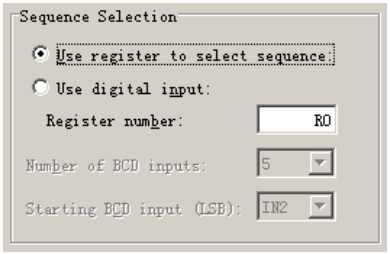
点击 Setup 选项卡打开序列选择，Go 命令和优先级输入命令控制。



4.7.1. 序列选择

使用寄存器选择序列

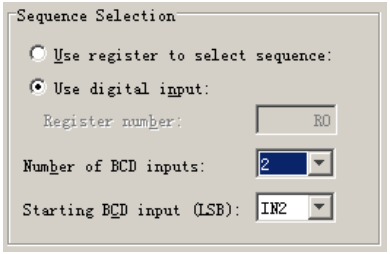
要配置为使用寄存器选择，请点击 Use register to select sequence 选项，并在下面的文本框中输入寄存器号（范围 R0-R31），该寄存器中包含了要执行的序列的号码。



关于写序列号码的更多信息，请参考 3.2.2 节所述。

使用数字输入选择序列

若希望用数字输入状态来选择执行的序列号码，请点击 Use digital input 选项。



然后在下面的两个下拉列表中，选择合适的值：

	描述
Number of BCD inputs	BCD 码用几个输入表示
Starting BCD input（LSB）	被选择表示 BCD 码的输入从哪个输入口开始，必须是连续的

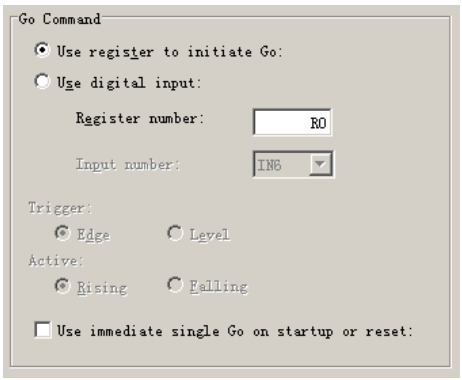
注意：这些 BCD 输入只在接收到 Go 命令时才会被读取。直到下一个 Go 命令前，这些输入可以用来作为其它目的使用。使用时要特别注意，避免输入端口使用的冲突。

4.7.2. Go 命令

选择一种下面描述的 Go 命令。点击选择 Use immediate single Go on startup or reset 的话，会在上电或复位或点击运行按钮时触发一个 Go 命令。这之后，寄存器或数字输入根据编程起作用。

使用寄存器初始化 Go

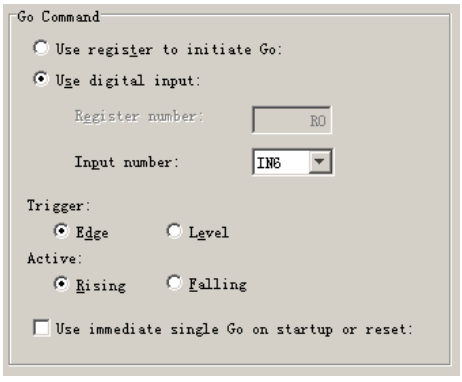
- 点击 **Use register to initial Go**
- 输入 Go 命令寄存器的寄存器号（R0-R31），如果同时选择了使用寄存器选择序列，应该修改序列选择寄存器号和 Go 命令寄存器号匹配。



关于 Go 位（第 15 位）的信息，请参看 3.2.2 小节所述。

使用数字输入初始化 Go

点击 **Use digital input**



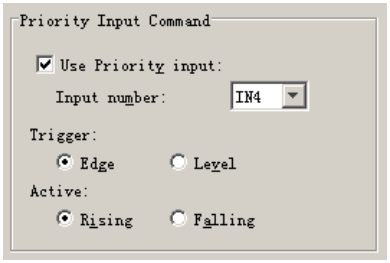
为下述区域选择合适的值：

输入端口号	选择使用哪个输入来选择序列
触发方式	边沿触发（Edge）：输入电平发生跳变时才会开始执行序列 电平触发（Level）：输入必须在合适的电平（高/低）才会开始执行序列。注意，如果输入电平一直保持有效，则会使序列重复执行。

有效条件	边沿触发时： • 上升沿（Rising）：Go 输入端口由低电平到高电平的跳变时开始执行序列 • 下降沿（Falling）：Go 输入端口由高电平到低电平的跳变时开始执行序列 电平触发时： • 高电平（High）：当 Go 输入为高电平时开始执行序列 • 低电平（Low）：当 Go 输入为低电平时开始执行序列

4.7.3. 优先输入命令

点击 **Use priority input**

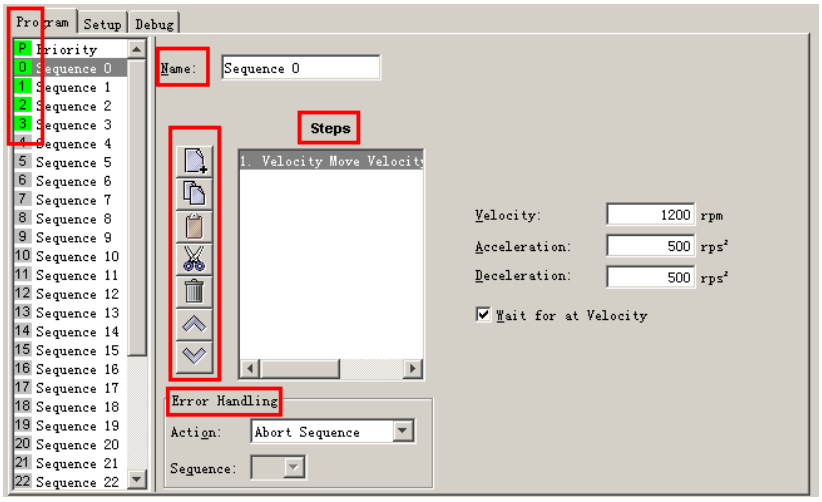


为下述区域选择合适的值：

输入端口号	选择使用哪个输入来选择序列
触发方式	边沿触发（Edge）：输入电平发生跳变时才会开始执行序列 电平触发（Level）：输入必须在合适的电平（高/低）才会开始执行序列。注意，如果输入电平一直保持有效，则会使序列重复执行。
有效条件	边沿触发时： • 上升沿（Rising）：Go 输入端口由低电平到高电平的跳变时开始执行序列 • 下降沿（Falling）：Go 输入端口由高电平到低电平的跳变时开始执行序列 电平触发时： • 高电平（High）：当 Go 输入为高电平时开始执行序列 • 低电平（Low）：当 Go 输入为低电平时开始执行序列

4.8. 创建和修改序列

Program 选项卡中包含所有创建序列的工具。



工具和控制：

图标	工具	描述
	Name	显示选择序列的名字，最多支持 16 个字符，可直接修改
	Add Step（添加步）	打开功能函数界面，选择一个函数后点击 Add Step，会将其添加到运动序列中，作为序列中的一步
	Copy Step	复制选择的序列中的一步
	Paste Step	将复制的步骤粘贴到序列中
	Cut Step	剪切掉选择的序列中的一步
	Delete Step	删除步
	Step Up	将选择的步骤向上移动一步
	Step Down	将选择的步骤向下移动一步
	Parameters	右边是对应函数的相关参数。具体描述参考后面的函数介绍
	Error Handling: Action	Abort Seq（放弃序列）：序列执行中发生错误时，索引器程序会停止执行该序列，等待 Go 命令。接收到 Go 命令后，会转跳到相应选择的序列上继续执行。 Jump to Seq（转跳到指定序列）：序列执行中发生错误时，直接转跳到指定的序列继续执行（不需要 Go 命令） 注意：序列执行错误不一定会导致驱动器错误，反之也是一样。
	Error Handling: Sequence	

序列列表包括一个优先序列和 32 个普通序列。
序列号有绿色背景表示序列中有编程的程序，灰色背景则表示序列为空。
点击序列号，在右边修改序列的名字后即可向序列中添加函数，每一个函数对应一个步（骤）。编辑完成后，将程序保存到驱动器中，即可开始运行。

4.9. 使用单步/调试模式

4.9.1. 总述

当索引器程序工作在单步/调试模式时，Debug 选项卡用来显示状态信息

4.9.2. 进入单步/调试模式

略。

4.9.3. 退出单步/调试模式

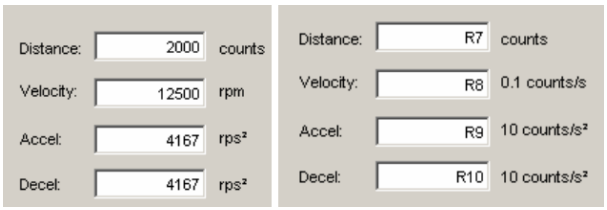
略。

5.函数

本章节详细描述了组成序列的各个功能函数的含义。

5.1. 使用寄存器向函数传值

在许多函数的参数区域，可以使用 32 个寄存器来代替直接输入的数字。例如，Move 参数可以直接输入数字（如下左图），也可以输入寄存器（如下右图）。



注意：输入寄存器后，光标切换出文本框后，后面的单位会自动变化。多数情况下直接输入数字的单位和寄存器输入的单位是不同的。请确保输入参数的准确性。

运行时会对 Move 运动中的参数做处理，如果是寄存器号，则会将寄存器中的值代入。如果其中的值无效，就会产生一个序列错误。

可以通过菜单 Tools→View Register Values 查看寄存器中的内容。

5.2. 多轴驱动器支持

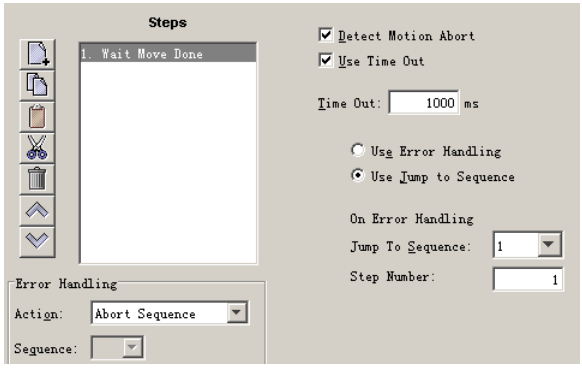
泰科智能驱动器最多支持三个独立的轴，分别是 Axis A，Axis B 和 Axis C。多轴驱动器中，可以对各个轴单独编程。可使用的函数分别为：

- 等待运动完成，等待事件，等待指定位置，等待参数
- 设置电流限制，设置跟踪窗口，设置增益
- 位置环速度运动，运动，回零

停止驱动器，位置触发输出，获取参数。

5.3. 等待运动完成（Wait Move Done）

等待运动完成功能会暂停正在执行的序列，直到正在运行中的运动完成。



为各参数设置合适的值。Time out（超时时间）中直接输入数值，或者输入寄存器号（R0-R31）。
其它选项的含义一目了然，用户可以根据需要进行选择。

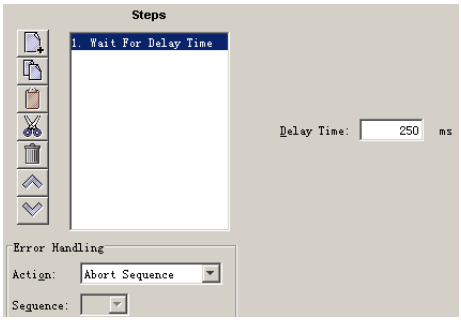
注意：

等待期间如果发生以下情况，会产生序列错误：

- 驱动器被硬件停止运行（如 IN1 的硬件使能开关）
- 驱动器发生错误
- 达到了行程的软件限制，或者硬件限位开关有效
- 使用寄存器给参数传值时，寄存器中的值无效

5.4. 等待一个延时时间（Wait for Delay Time）

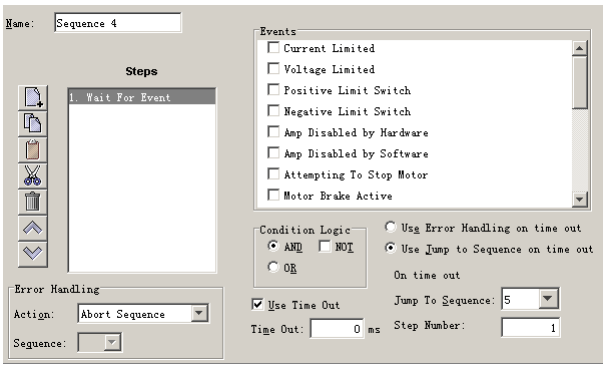
等待一个延时时间函数的功能是暂停序列的执行，等待一个指定的时间。



延时时间中可以直接输入数值，也可以输入寄存器值。单位都是 ms（毫秒）。

5.5. 等待事件（Wait for Event）

等待事件函数的功能是暂停序列的执行，直到特定的事件条件发生。



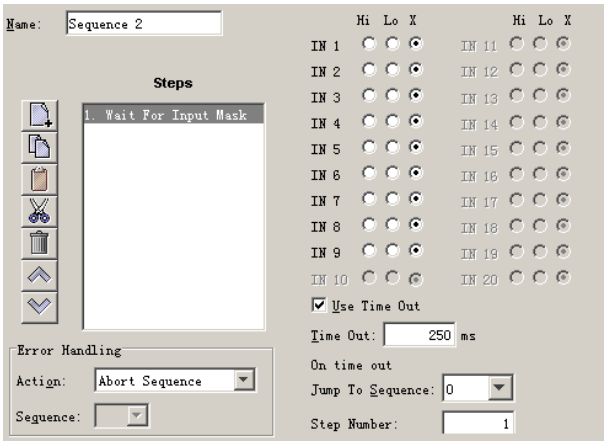
在对话框中选择或输入合适的参数（数值或寄存器）。

在 Condition Logic 中选择 AND：等待直到所选的事件条件全部发生；OR：等待直到所选的事件中有任一个发生；NOT：等待直到选择的事件条件没有出现。

使用超时时间，等待指定时间后，如果上述条件没有满足，即进行错误处理或转跳到指定的序列中继续运行。
当使用寄存器输入时，若寄存器内的值无效则会报序列错误。

5.6. 等待输入掩码（Wait for Input Mask）

这个函数的功能是等待设定的驱动器数字输入状态和设定的状态一致，然后才会继续下一步的执行。



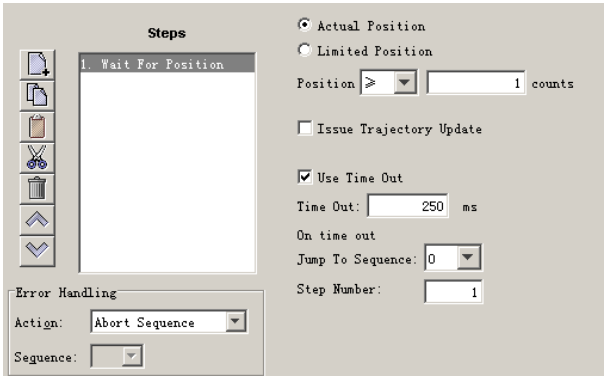
在对话框中选择或输入合适的参数（数值或寄存器）。参数描述如下表：

参数	描述
Hi Lo X (IN1-IN16)	定义： Hi：当输入是高电平时满足掩码条件 Lo：当输入是低电平时满足掩码条件 X：与此输入无关
Use Time Out	如果选择该项，且在设定的超时时间内未达到掩码条件，程序会转跳到指定序列的指定步中
Time out	超时时间：超时选项使用。寄存器单位：ms
On Time out Jump to Sequence	如果超时时间到，程序转跳到指定序列的指定步中

当使用寄存器输入时，若寄存器内的值无效则会报序列错误。

5.7. 位置等待（Wait for Position）

位置等待函数的功能是暂停序列的执行，直到轴的位置值达到编程的条件时。

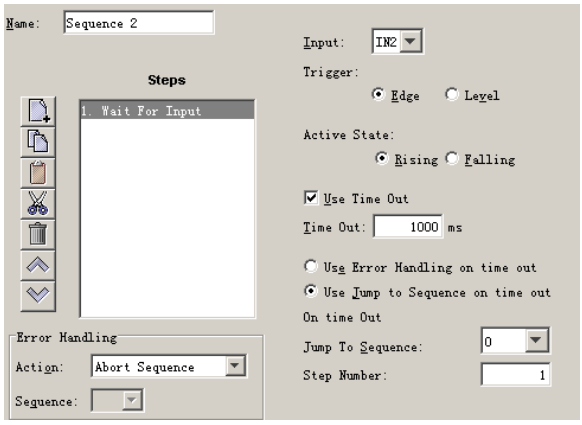


请为以下参数选择合适的值。

参数	描述
Position	等待，直到轴的位置值满足如下条件： • ≥大于等于设定的位置值 • ≤小于等于设定的位置值 寄存器单位：counts（脉冲数）
Actual Position	使用实际位置值和设定值进行比较
Limited Position	使用限制的位置值和设定值进行比较。通常用在步进电机操作的开环步进模式中
Issue Trajectory Update	位置条件满足时发送一个轨迹更新命令
Use Time Out	如果选择该项，且在设定的超时时间内未达到位置条件时，程序会转跳到指定序列的指定步中
Time out	超时时间：超时选项使用。寄存器单位：ms
On Time out Jump to Sequence	如果超时时间到，程序转跳到指定序列的指定步中

5.8. 输入等待（Wait for Input）

输入等待函数的功能是暂停序列的执行，直到满足指定的输入条件。



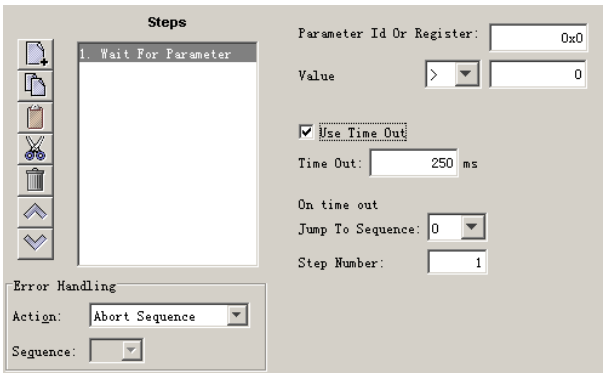
请为以下参数选择合适的值：

参数	描述
Input	选择监测的输入端口号
Trigger	Edge：等待，直到选择输入的电平发生跳变 Level：等待，直到输入的电平满足条件。如果执行到该步时输入电平已经是有效状态，则会直接执行下一步。
Active State	边沿触发时： • Rising：输入端口由低电平到高电平的跳变时继续下一步的执行 • Falling：输入端口由高电平到低电平的跳变时继续下一步的执行 电平触发时： • High：当输入为高电平时继续下一步的执行 • Low：当输入为低电平时继续下一步的执行
Use Time Out	如果选择该项，且在设定的超时时间内未达到位置条件时，程序会转跳到指定序列的指定步中
Time out	超时时间：超时选项使用。寄存器单位：ms

On Time out Jump to Sequence	如果超时时间到，程序转跳到指定序列的指定步中
Use Error Handling on time out	如果超时时间到，产生一个序列错误信号

5.9. 参数等待（Wait for Parameter）

参数等待函数的功能是暂停序列的执行，直到满足设定的条件。



请为以下参数选择合适的值。

参数	**描述**
Parameter Id or Register	驱动器参数的 ID 号或者监测的寄存器
Parameter Value	根据选择的操作符，参数值或者寄存器的内容会和文本框中设定的值进行比较
Use Time Out	如果选择该项，且在设定的超时时间内未达到位置条件时，程序会转跳到指定序列的指定步中
Time out	超时时间：超时选项使用。寄存器单位：ms
On Time out Jump to Sequence	如果超时时间到，程序转跳到指定序列的指定步中

备注：

要查看参数 ID 列表，请参考*泰科智能参数字典*。使用 ASCII ID 值。

参数 IDs 可以以十进制或者十六进制数的方式输入，十六进制输入时请在数字前面加 0x 标号（例如 0x002c）。当程序被重新加载时，所有的值都会以十进制格式显示。

由于该函数的扫描时间有限，对一些快速变化的参数，例如实际速度和实际电流，请不要使用“=”操作符。可以选择使用大于等于或小于等于。

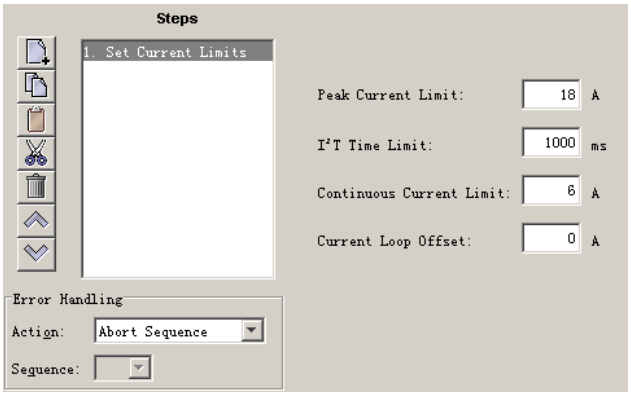
错误：

以下情况会发生序列错误：

- 指定的参数 ID 不存在
- 指定的参数 ID 是只存在于 flash 中的参数
- 指定的参数返回值超过两个字

5.10. 设置电流上限（Set Current Limits）

该函数的功能是改变电流相关参数的上限值。



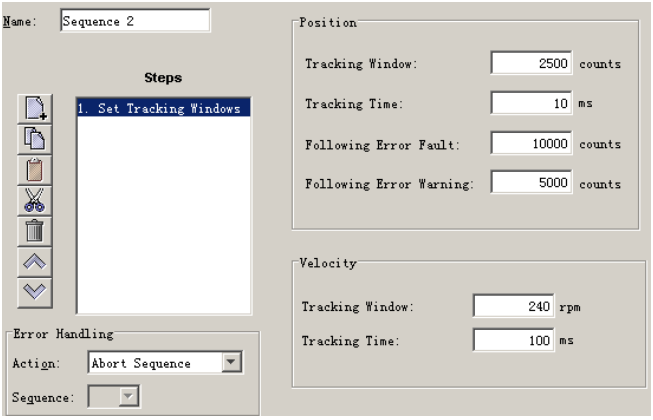
可以直接输入数值，也可以使用寄存器。各参数描述如下：

参数	描述
Peak Current Limit	短时间内驱动器产生的最大电流。该值不能超过驱动器的峰值电流。寄存器单位：0.01A
I2T Time Limit	在电流必须减少到连续电流上限或产生错误报警前，电机承受峰值电流时持续的最大时间。这是个等价量，I2T 的具体解释请参考泰科智能驱动器使用手册。寄存器单位：ms
Continuous Current Limit	驱动器输出的最大连续电流。寄存器单位：0.01A
Current Loop Offset	设定电流偏移量。寄存器单位：0.01A

当使用寄存器输入时，若寄存器内的值无效则会报序列错误。

5.11. 设置跟踪窗口（Set Tracking Windows）

该函数的功能是修改速度和位置的跟踪窗口。



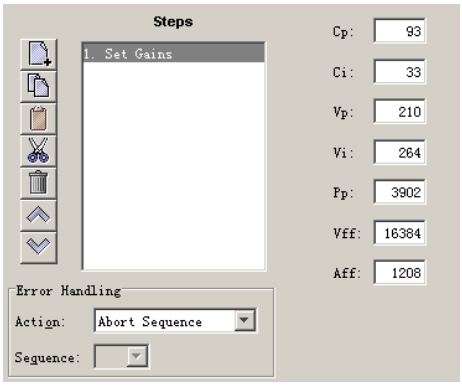
请为以下参数选择合适的值（立即数或寄存器）：

Position Tracking	
参数	**描述**
Tracking Window	跟踪窗口的宽度。寄存器单位：counts
Tracking Time	跟踪时间内，位置必须在上述跟踪窗口内，才会认为跟踪正常。寄存器单位：ms
Following Error Fault	跟踪误差累计达到该值时就会产生错误。寄存器单位：counts
Following Error Warning	跟踪误差累计达到该值时就会产生警告。寄存器单位：counts
Velocity Tracking	
参数	**描述**
Tracking Window	跟踪窗口的宽度。寄存器单位：0.1counts/s
Tracking Time	跟踪时间内，速度必须在上述跟踪窗口内，才会认为跟踪正常。寄存器单位：ms
Current Loop Offset	设定电流偏移量。寄存器单位：0.01A

当使用寄存器输入时，若寄存器内的值无效则会报序列错误。

5.12. 设置增益（Set Gain）

该函数可以在执行的序列中重新设置电流环，速度环和位置环的增益。例如，它可以用在电机轴上的负载发生变化时。

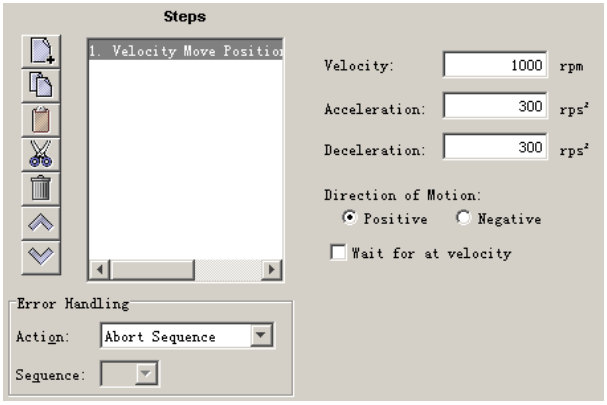


请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Cp	电流环比例增益
Ci	电流环积分增益
Vp	速度环比例增益
Vi	速度环积分增益
Pp	位置环比例增益
Vff	速度前馈增益
Aff	加速度前馈增益

5.13. 位置模式下速度运动（Velocity Move Position Mode）

该函数可以用来改变驱动器的操作模式，将其编程为位置模式，并设置一个恒定的速度轨迹值。



请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Velocity	给定速度值。寄存器单位：0.1 counts/s （只能是正值）
Acceleration	加速度。寄存器单位：10 counts/s^2
Deceleration	减速度。寄存器单位：10 counts/s^2
Direction of Motion	运动方向：正 或 负
Wait for at velocity	如果选择了该项，序列的执行会停留在这一步，直到给定速度达到了这个新值。

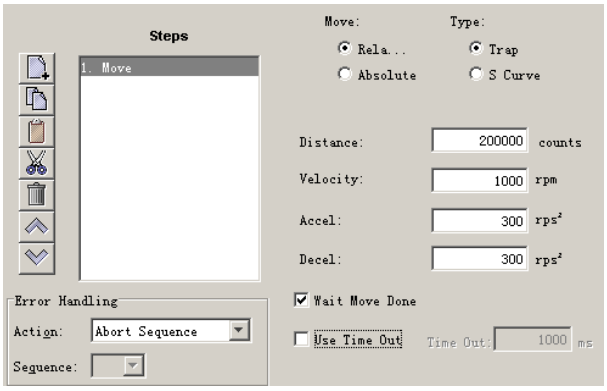
错误：

当该函数执行时碰到以下情况，就会产生序列错误：

- 驱动器被硬件停止运行
- 驱动器发生错误
- 电机相位未初始化
- 使用的寄存器中的内容是无效的参数

5.14. 运动（Move）

该函数的功能是执行一个指定参数的运动曲线。



请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Move	运动类型： • Relative：相对运动 • Absolute：绝对运动
Type	曲线类型： • Trap：梯形曲线 • S Curve：S 型曲线
Distance	相对运动的距离。寄存器单位：counts
Position	绝对运动的位置点。寄存器单位：counts
Velocity	运动达到匀速时的速度。寄存器单位：0.1 counts/s
Accel	梯形运动曲线时的加速度，或者 S 型曲线时的加减速度。寄存器单位：10 counts/s^2
Decel(Trap move only)	梯形运动曲线时的减速度。寄存器单位：10 counts/s^2
Jerk(S-Curve only)	S 型运动曲线时的加速率。寄存器单位：100 counts/s^3
Wait Move Done	选择此项后，序列执行会在此步进行等待，直到运动完成（相当于进入该步后关闭了所有中断，包括优先命令中断，运动完成后再打开）。如果运动期间发生错误，序列会根据编程的方式退出。

注意：

如果没有选择 **Wait Move Done** 选项，若有命令（如 Go 命令或优先级命令）要求执行另一个运动，第二运动会立即执行，并产生如下结果：

- 如果第二运动是相对运动，走梯形曲线，则所要走过的距离是从当前开始执行的位置点开始计算
- 如果第二运动是绝对运动，梯形曲线，轴走到第二运动指定的位置点
- 如果第二运动是 S 型曲线，上一个运动还没停止，就会报错

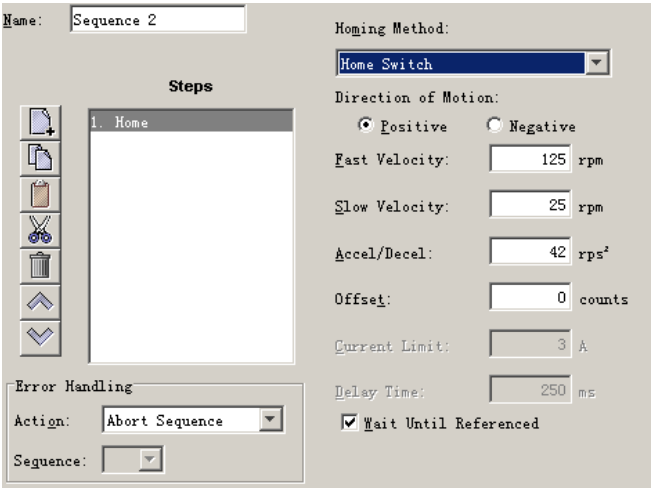
错误：

下述情况下会报序列错误：

- 运动开始时驱动器被硬件停止，或者运动中硬件停止驱动器工作
- 运动开始时驱动器有错误，或者运动期间发生了错误
- 运动中达到了软件行程限位或者硬件限位开关有效

5.15. 回零点（Home）

回零点的功能是按指定的方法和参数执行一个回零点命令。



请为以下参数选择合适的值（立即数或寄存器）。

参数	描述
Homing Method	选择一个回零点的方法。参考后面的回零方法描述。
Direction of Motion	设置运动的初始方向。正（positive）或负（Negative）
Fast Velocity	使用这个速度找限位或回零开关。当移动到一个偏移位置，或者旋转变压器或管状伺服的索引位置时，也用该速度。寄存器单位：0.1counts/s
Slow Velocity	找到开关边沿、索引脉冲或急停时使用的速度。寄存器单位：0.1counts/s
Accel/Decel	回零点时的加速度和减速度。寄存器单位：10 counts/s2
Offset	找到参考点后，轴还会移动这么远的偏移值。然后置实际位置为 0，并将该值作为零点。寄存器单位：counts
Current Limit	回零点或急停时的最大电流。寄存器单位：0.01A
Delay Time	回零点或急停时连续使用最大电流所允许的时间。寄存器单位：ms
Wait until reference	选择该项后，序列的执行会在该步进行等待，直到回零点运动结束。如果回零点过程中发生了错误，程序会按下述方法进行处理： • 停止序列 • 转跳到指定的序列

备注：

回零结束前，不要执行运动函数。序列错误并不会报警，控制面板中的错误日志（error log）也不会记录，只是序列不再正常执行，会按左下角的 Error Handling 的设置处理。

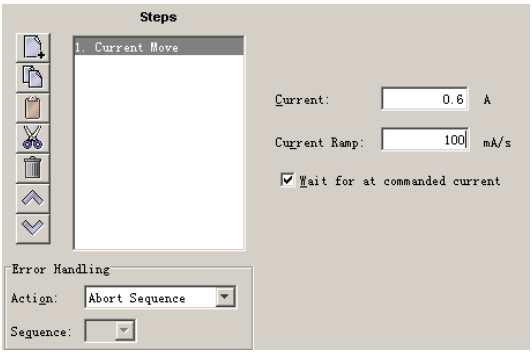
错误：

下述情况下会发生序列错误：

- 回零过程中，驱动器被硬件停止工作
- 回零点开始时，驱动器发生了错误。或者在回零时发生了错误。

5.16. 电流运动（Current Move）

该函数用来改变驱动器的操作模式为可编程电流模式，并对该模式进行配置。



请为以下参数选择合适的值（立即数或寄存器）。

参数	描述
Current	给定电流。寄存器单位：0.01A
Current Ramp	电流达到给定电流过程中增加的速度。寄存器单位：mA/s
Wait for commanded cur	选择该项后，序列的执行会在该步进行等待（不允许任何条件来打断），直到指定的电流达到了这个设定的值。

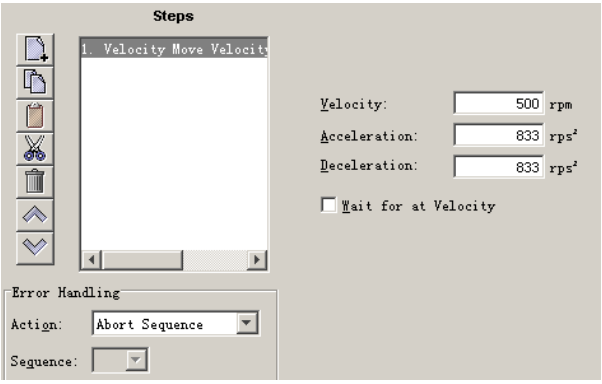
错误：

下述情况下会报序列错误：

- 驱动器被硬件停止工作
- 驱动器有错误
- 电机相位未初始化
- 使用寄存器时，寄存器中的内容对该参数来说无效

5.17. 速度模式下的速度运动（Velocity Move Velocity Mode）

该函数用来改变驱动器的工作模式，将其改为速度控制模式，并设置一个恒定的速度轨迹。



请为以下参数选择合适的值（立即数或寄存器）。

参数	描述
Velocity	给定速度值。寄存器单位：0.1 counts/s （只能是正值）
Acceleration	加速度。寄存器单位：10 counts/s^2
Deceleration	减速度。寄存器单位：10 counts/s^2
Wait for at velocity	如果选择了该项，序列的执行会停留在这一步，直到给定速度达到了这个新值。

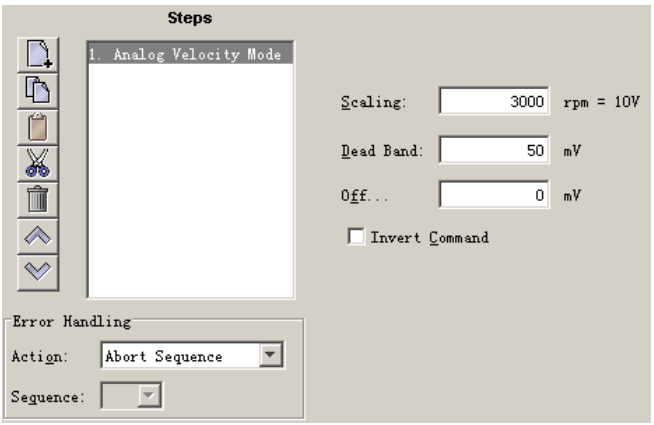
错误：

下述情况下会报序列错误：

- 驱动器被硬件停止工作
- 驱动器有错误
- 电机相位未初始化
- 使用寄存器时，寄存器中的内容对该参数来说无效

5.18. 模拟量速度模型（Analog Velocity Mode）

该函数用来改变驱动器的工作模式，将其改为外部模拟量控制的速度模式并进行配置。



请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Scaling	设置输入电压和速度的对应关系。寄存器单位：counts/second=10V
Dead Band	设置死区。驱动器将死区范围内的值都认为是 0，其它值都和死区相减。寄存器单位：mV
Offset	输入电压的偏移量。寄存器单位：mV
Invert Command	对输入信号，驱动器改变其输出极性（即正电压时电机反向运行，负电压时驱动器正向运行）。

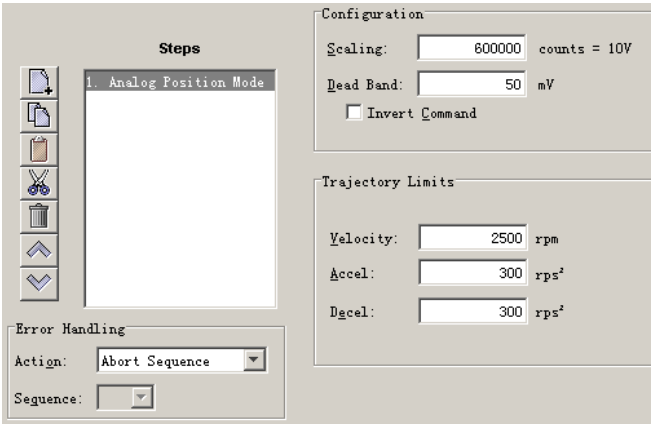
错误：

下述情况下会报序列错误：

- 驱动器被硬件停止工作
- 驱动器有错误
- 电机相位未初始化
- 使用寄存器时，寄存器中的内容对该参数来说无效

5.19. 模拟量位置模式（Analog Position Mode）

该函数用来改变驱动器的工作模式，将其改为外部模拟量控制的位置模式并进行配置。



请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Scaling	最大输入电压和最大位置值的对应关系。寄存器单位：counts=10V
Dead Band	设置死区。驱动器将死区范围内的值都认为是 0，其它值都和死区相减。寄存器单位：mV
Invert Command	对输入信号，驱动器改变其输出极性（即正电压时电机反向运行，负电压时驱动器正向运行）。
Velocity	运动期间的最大速度。寄存器单位：0.1 counts/s
Accel	运动期间的最大加速度。寄存器单位：10 counts/s^2
Decel	运动期间的最大加速度。寄存器单位：10 counts/s^2

模拟量位置模式备注：

该模式下模拟位置命令都是指相对的运动命令。驱动器使能后即开始读取模拟输入端口的电压值。然后，电压的任何变化都会使电机轴移动一个相对的距离，该距离正比于编程值对输入电压的变化，是对使能时的位置点而言的。要想编程该模式为绝对位置命令，驱动器应该在每次使能时都在零点。要实现这种功能，其中一个简单的方法是，编程优先序列和驱动器使能共用同一个输入口。优先序列中编程轴回零点函数。这样，驱动器每次使能时，轴都会自动回零点。

错误：

下述情况下会报序列错误：

- 驱动器被硬件停止工作
- 驱动器有错误
- 电机相位未初始化
- 使用寄存器时，寄存器中的内容对该参数来说无效

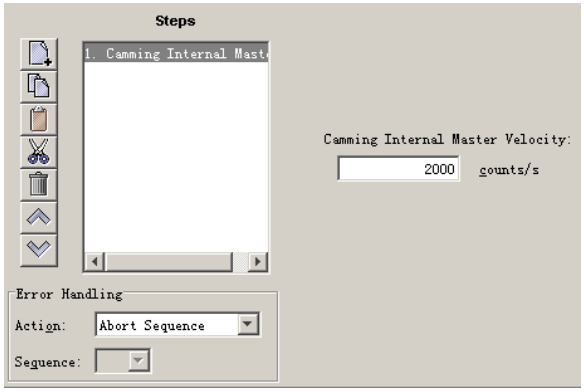
5.20. 停止驱动器工作（Disable Amplifier）

该函数属于软件停止驱动器的工作。

备注：驱动器会自动重新使能任一要求轴运动的函数，例如回零点或运动(Move)。且该函数不会产生错误。

5.21. 凸轮内部主轴（Camming Internal Master）

该函数用来修改凸轮内部主轴生成脉冲的速度。



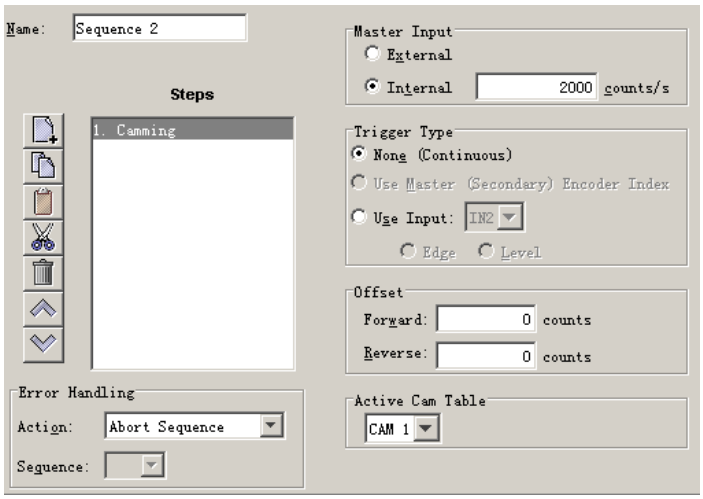
请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Velocity	凸轮的内部主轴生成脉冲的速度。寄存器单位：0.1 counts/s

使用寄存器时，寄存器中的内容对该参数来说无效时会报序列错误。

5.22. 电子凸轮（Camming）

该函数用来改变驱动器的工作模式，改为电子凸轮模式，并对模式进行配置。



请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Master Input	External：凸轮主轴脉冲源来自外部 Internal：凸轮主轴脉冲源来自驱动器内部的生成器。脉冲生成速度可在后面的文本框中修改。也可以使用 Camming Internal Master 函数对速度进行复位。
Triger Type	控制如何触发凸轮运动的执行： None(Continuous)：连续执行有效的凸轮表 Use Master(Secondary) Encoder Index：当驱动器接收到凸轮主编码器的索引脉冲时开始执行有效凸轮表。凸轮表执行中若接收到索引脉冲则直接忽略。 Use Input，Edge：当接收到指定输入引脚的上升沿时开始执行有效的凸轮表 Use Input，Level：只要指定输入引脚为有效高电平就连续执行有效凸的轮表
Offset	Forward：当主轴正向移动时，执行有效的凸轮表前延时时间，单位是脉冲数 Reverse：当主轴负向移动时，执行有效的凸轮表前延时时间，单位是脉冲数
Active Cam Table	有效的凸轮表。凸轮触发信号有效时，要执行的包含运动曲线点的凸轮表

备注：

- 内部脉冲生成的速度也可以通过 Camming Internal Master 函数进行修改设置。
- 驱动器会保持在凸轮工作模式，直到另外一个索引器程序函数对其进行了修改，或者驱动器复位，或者掉电重启。

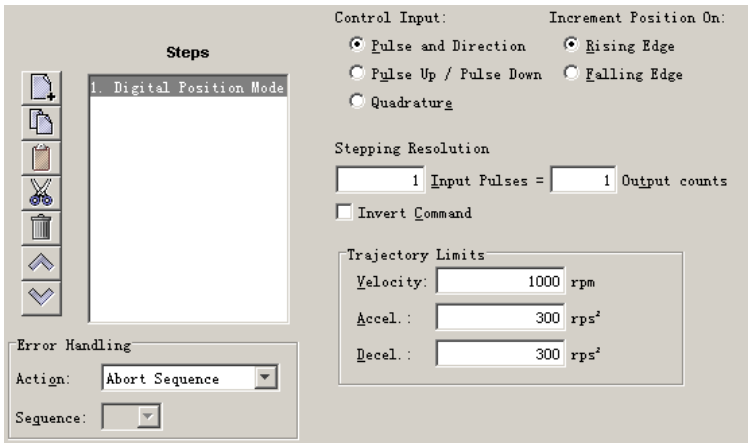
错误：

下述情况下会报序列错误：

- 驱动器被硬件停止工作
- 驱动器有错误
- 电机相位未初始化
- 使用寄存器时，寄存器中的内容对该参数来说无效

5.23. 外部数字量输入控制位置模式（Digital Position Mode）

该函数用来改变驱动器的工作模式，可将其修改为由外部数字量输入进行控制的位置控制模式，并进行参数设置。



请为以下参数选择合适的值（立即数或寄存器）。

参数	描述
Control Input	Pulse and Direction：一个输入引脚控制方向，一个输入引脚控制脉冲。 Pulse Up/Pulse Down：一个输入引脚作为正向的脉冲信号，一个输入引脚作为负向运动的脉冲信号。 Quadrature：来自主编码器（两个输入引脚）的 A/B 正交命令，可提供速度和方向命令。
Increment position on	Rising Edge：输入脉冲的上升沿增加位置 Falling Edge：输入脉冲的下降沿增加位置
Stepping Resolution	Input Pulses：输入引脚的脉冲数 Output Counts：指定的输入脉冲对应的输出脉冲数。范围是 1-32767。寄存器单位：Pulses/Counts
Invert Command	选择后，对方向命令取反。
Velocity	运动达到匀速时的速度。寄存器单位：0.1 counts/s
Accel	梯形运动曲线时的加速度。寄存器单位：10 counts/s^2
Decel	梯形运动曲线时的减速度。寄存器单位：10 counts/s^2

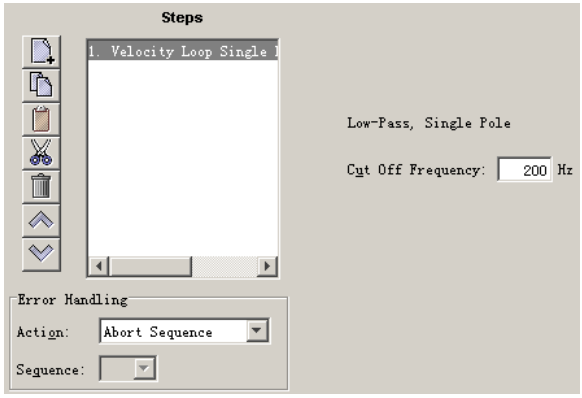
错误：

下述情况下会报序列错误：

- 驱动器被硬件停止工作
- 驱动器有错误
- 电机相位未初始化
- 使用寄存器时，寄存器中的内容对该参数来说无效

5.24. 速度环单级点输出滤波（Velocity Loop Single Pole Output Filer）

该函数用来修改速度环输出滤波器。



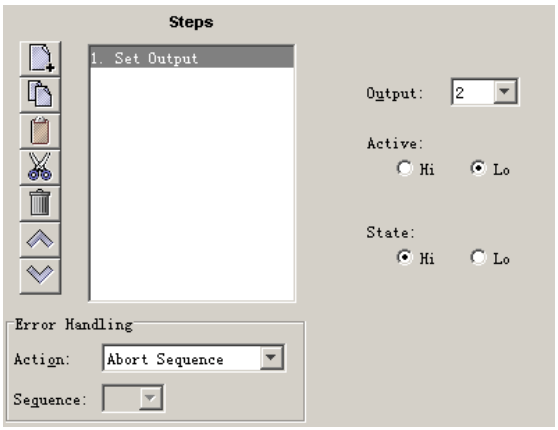
单极点的低通滤波器，请在文本框中输入截止频率。范围：1-1500

该函数不会产生序列错误。

5.25. 设置输出（Set Output）

该函数用来设置指定输出口的状态。

注意：执行该函数时，会修改选择输出的设置为“程序控制，低电平有效”。“程序控制，低电平有效”的设置在驱动器复位前会一直有效。在编程数字输出时请考虑到这种情况。

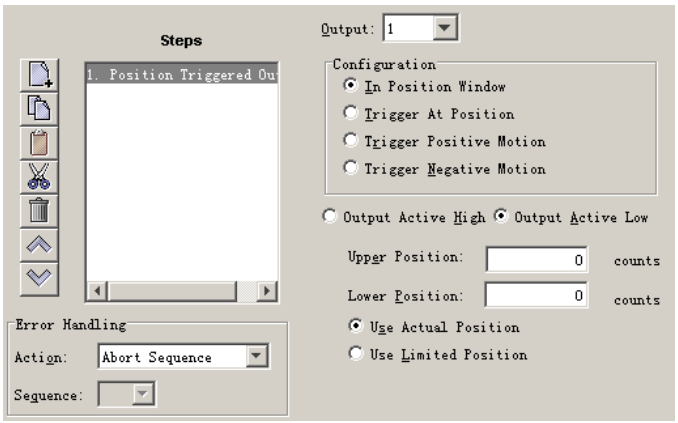


请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Output	选择要设置的输出口。
State	• Hi：设置选择的输出为高电平或者说关闭输出口 • Lo：设置选择的输出为低电平或者说打开输出口

5.26. 位置触发输出（Position Triggered Output）

该函数的功能是，当指定的位置条件满足时，编程驱动器的一个输出为有效状态。



请为以下参数选择合适的值（立即数或寄存器）：

基本设置	描述
Output	要编程的数字输出口
Output Active High	有效时输出为高电平
Output Active Low	有效时输出为低电平
Use Actual Position	目标位置使用实际位置
Use Limited Position	目标位置使用限制的位置。通常用在步进电机操作的开环步进模式中

配置	描述和参数
In Position Window	当轴位置在下面设置的 Upper Position 和 Lower Position 之间时使输出有效
Trigger at Position	轴行程通过编程的位置时，使输出在设定的时间（Time）内有效
Trigger Position Motion	当轴正向运行且通过编程的位置时，使输出在设定的时间内有效
Trigger Negative Motion	当轴负向运行且通过编程的位置时，使输出在设定的时间内有效

注意：执行该函数时，会修改选择输出的配置。输出的配置在驱动器复位或重新配置前会一直有效。

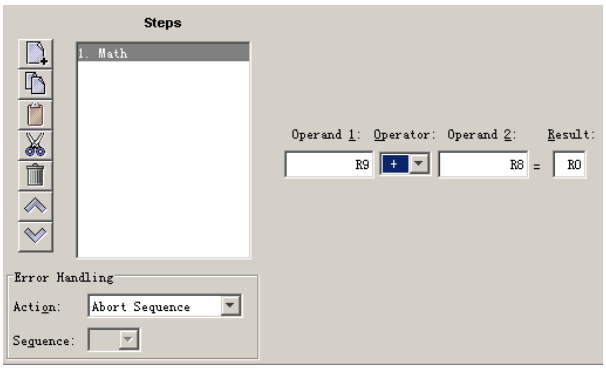
错误：

下述情况下会报序列错误：

- 使用寄存器时，寄存器中的内容对该参数来说无效

5.27. 数学运算（Math）

该函数用来进行基本的整数运算，并将结果赋给程序中的寄存器。



请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Operand 1	第一个操作数。可以是整型常量或索引器程序的寄存器
Operator	操作符：加/减/乘/除
Operand 2	第二个操作数。可以是整型常量或索引器程序的寄存器
Result	结果写入到哪个寄存器中。对于除法，写入的是结果的整数部分

Remainder	将除法时的余数部分写入到指定的寄存器中
使用乘法运算时，该函数允许操作数的范围是-32768 到 32767.超出此范围的操作请使用 Extended Math 函数（参考下面的介绍）	

备注：

该函数不支持加法的进位和减法的借位。因此，如果数学运算的结果数超过寄存器可接收的最大范围（2^{31}-1 到-2^{31}之间），保存的结果是不正确的，但是并不会报错。

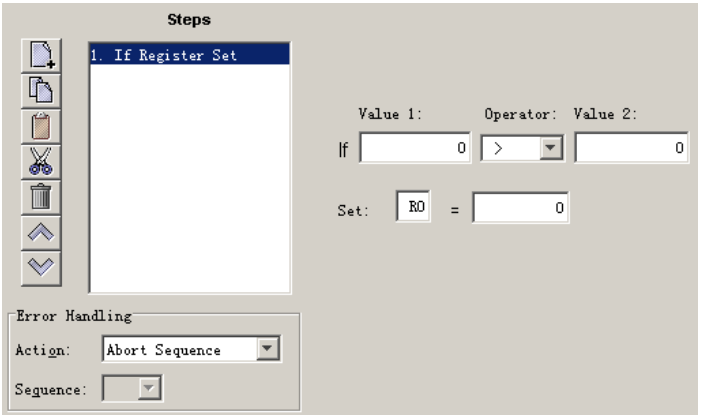
错误：

下述情况下会产生序列错误：

- 乘法运算时，一个操作数是寄存器且其值小于-32768 或大于 32767
- 除法运算时，第二个操作数是寄存器且其值为 0

5.28. 条件寄存器设置（If Register Set）

该函数用来根据条件设置寄存器的值。



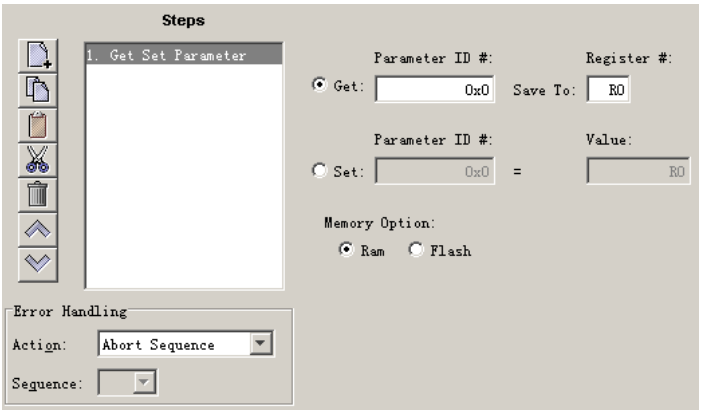
请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Value 1	要做比较的第一个数。可以是整型常量或索引器程序的寄存器
Operator	操作符：大于，小于，等于，不等于，大于等于，小于等于
Value 2	要做比较的第二个数。可以是整型常量或索引器程序的寄存器
Set	方程左边输入要设置的寄存器，右边输入整型常量或寄存器

当索引器程序设置为使用寄存器来选择序列时，条件寄存器设置函数可以决定当前序列执行完后接着执行哪个序列。该函数不会产生错误。

5.29. 获取设置参数（Get Set Parameter）

该函数用来获取或者设置驱动器的内部参数。



请为以下参数选择合适的值（立即数或寄存器）：

设置	描述
Get	将指定的驱动器参数的当前值拷贝到寄存器中
Parameter ID #	函数要读取的驱动器参数 ID。驱动器参数 ID 列表可以在*泰科智能参数字典*文档中查看。请使用文档中的 ASCII ID 值。
Register #	要写入值的寄存器
Set	将一个数拷贝到指定的驱动器参数 ID 中
Parameter ID #	要写入值的驱动器参数 ID
Value	要拷贝到驱动器参数中的数值或者指定的寄存器中的内容
Memory Option	选择是在参数的 RAM 或者 flash 中使用 Get 或 Set

参数 ID 可以是十进制或十六进制数。当输入十六进制数时，前面要加 0x（例如，0x002c）。索引器程序会自动将数转换成十六进制，并以十六进制显示。

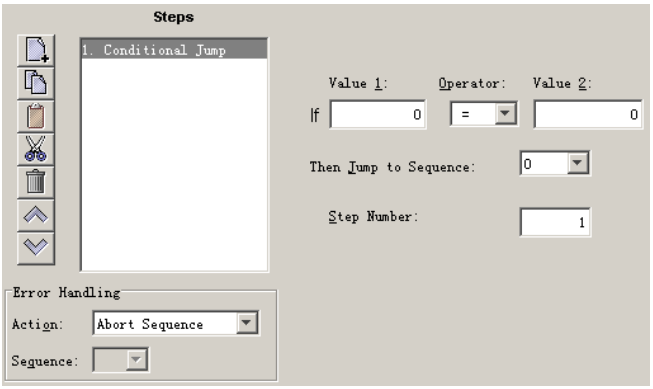
错误：

下述情况下会产生序列错误：

- 指定的参数 ID 不存在
- 指定的参数返回值超过两个字
- 向只读的参数写数值
- 向参数写入的数值非法

5.30. 条件跳转（Conditional Jump）

该函数功能是根据条件判断转跳到哪个序列，哪一步。



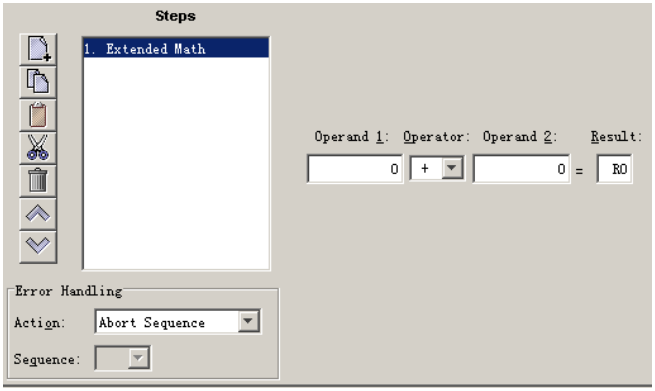
请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Value 1	要做比较的第一个数。可以是整型常量或索引器程序的寄存器
Operator	操作符：大于，小于，等于，不等于，大于等于，小于等于
Value 2	要做比较的第二个数。可以是整型常量或索引器程序的寄存器
Then Jump to Seq	要转跳到哪个序列。可以是当前的序列
Step Number	序列中的哪一步

备注：
如果转跳到的序列中没有编程，或者指定的步不存在，并不会产生错误。取而代之的是，驱动器会返回到一个低功耗状态，等待下一个 Go 命令。该函数不会产生错误。

5.31. 扩展的数学运算（Extended Math）

该函数用来进行整数运算，并将结果写入到指定的寄存器中。该函数和前面的 Math 函数基本是一样的，除了对操作数的范围不做限制之外。



请为以下参数选择合适的值（立即数或寄存器）：

参数	描述
Operand 1	第一个操作数。可以是整型常量或索引器程序的寄存器
Operator	操作符：加/减/乘/除
Operand 2	第二个操作数。可以是整型常量或索引器程序的寄存器
Result	结果写入到哪个寄存器中。对于除法，写入的是结果的整数部分
Remainder	将除法时的余数部分写入到指定的寄存器中

Math 函数不支持加法进位或减法借位。而且超过范围不会报错。因此较大数的运算可以选择用扩展的数学运算。

错误：

下述情况下会产生序列错误：

• 除法运算时，第二个操作数是寄存器且其内容为 0.

5.32. 逻辑运算（Logic）

该函数用来对驱动器参数值或索引器程序寄存器值进行逻辑运算。

请为以下参数选择合适的值（立即数或寄存器）。

参数	描述
Get Parameter or Register	操作数。指定一个驱动器的参数 ID 或索引器程序寄存器
Logic Operator	操作符：与（AND），或（OR），非（XOR）
Value or Register	操作数。指定一个数或者寄存器
Result	将结果写入到哪个寄存器中

参数 ID 可以是十进制或十六进制数。当输入十六进制数时，前面要加 0x（例如，0x002c）。索引器程序会自动将数转换成十六进制，并以十六进制显示。

错误:

下述情况下会产生序列错误：

• 指定的参数 ID 不存在

• 指定的参数返回值超过两个字

• 向只读的参数写数值

附录 A：串口发送 ASCII 命令

索引器程序提供 32 个寄存器，每个寄存器有 4 个字节（两个字）。它们可以用来：

- 选择序列
- 初始化一个 Go 命令
- 向索引器程序传值

本节描述如何通过串口和驱动器建立连接，并发送 ASCII 命令对索引器程序的寄存器进行存取。

A.1. 连接

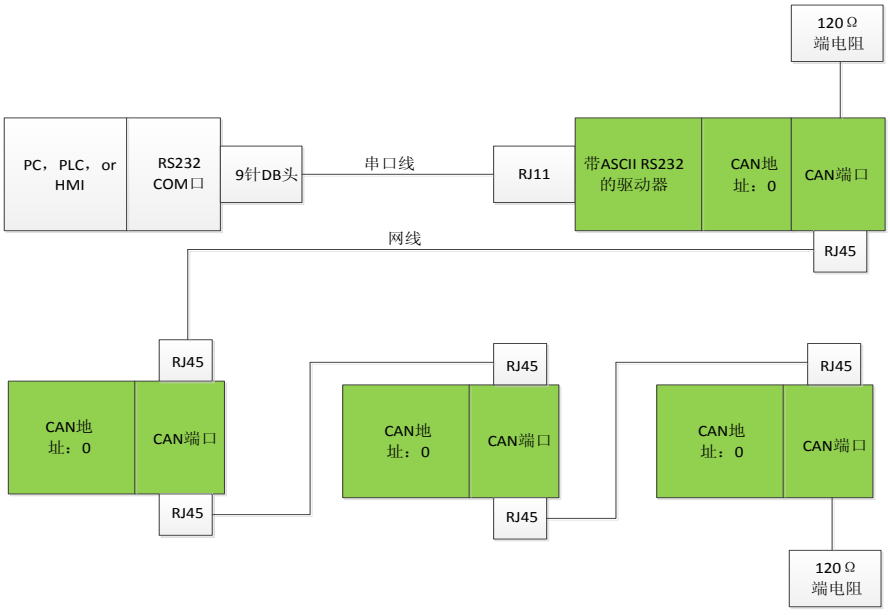
本小节描述如何通过 RS232 总线和驱动器进行连接通信控制。在多点网络中，与串口线相连的驱动器被用做网关，可以通过它访问 CAN 网络上的其它驱动器。单轴连接和多点网络连接的说明如下面两小节所述。

A.1.1. 单轴连接

通过 RS232 串口总线控制单个轴的话，请设置该轴的 CAN 节点地址为 0. 注意，如果上电后 CAN 节点地址切换为 0，则必须进行复位或者掉电重启，新的设置才会生效。

A.1.2. 多点网络连接

在多点网络中，与串口线相连的驱动器被用做网关，可以通过它存取 CAN 网络上的其它驱动器。请将与串口线相连的驱动器（网关）的 CAN 节点地址设置为 0，网络链上的其它驱动器赋予一个唯一的 CAN 节点地址（范围是 1-127）。更多关于 CAN 节点地址设置的信息，请参考 *CME2 用户手册*。CAN 网络链上首尾的驱动器都要加 120Ω 的端电阻。



A.2. 通讯协议

请使用如下的协议对寄存器进行读写：

波特率(Baud Rate)	9600 —— 上电或复位后的默认值
数据格式(Data Format)	N, 8, 1 —— 无奇偶校验，8 个数据位，1 个停止位
流控(Flow Control)	无（None）

A.3. 读写寄存器

要使用泰科智能 ASCII 命令读或写程序的 32 个寄存器，请用 i 命令后跟 r 变量来实现：
[节点号-可选] i rn [要写的值-可选]<cr>

其中：

- **[节点号可选]:** 是多点网络中的 CAN 节点地址，范围是 0-127. 节点号后面应跟一个空格。如果该处不填，默认是和串口线直接相连的驱动器。
- **i rn:** 是访问程序寄存器的指令码。小写的 ASCII 字符 i 后跟一个空格，后面再跟一个小写字母 r，然后是一个数字 n。其中 n 是要读写的寄存器号，范围是 0-31.
- **[要写的值-可选]:** 是要写入寄存器的值。该值和前面的寄存器号之间应该有一个空格。如果省略该值，则默认是读寄存器，返回相应寄存器中的内容。输入的值应该是十六进制格式的正整数或负整数。
- **<cr>:** 是回车键，表示指令结束。

如果写指令被程序接受，就会返回“ok”字符，后跟一个回车符。如果读指令被程序接受，会返回一个“r”字母后跟寄存器中的内容（十进制），最后是一个回车字符。**示例：**

操作	**指令**	**返回值**
读直接与串口线相连的驱动器中 R31 中的内容	i r31<cr>	r [寄存器内容]<cr>
向与串口线相连的驱动器中的 R0 写 500	i r0 500<cr>	ok<cr>
向与串口线相连的驱动器中的 R31 写-500	i r31 -500<cr>	ok<cr>
读 CAN 网络上节点地址为 15 的驱动器中 R0 的值	15 i r0<cr>	r [寄存器内容]<cr>
向 CAN 网络上节点地址为 15 的驱动器中的 R31 写十六进制值 800d	15 i r31 0x800d<cr>	ok<cr>

如果发送的命令有误，驱动器会返回字符“e”后跟错误代码。错误代码含义如下：

代码	**含义**
10	传值超过范围
31	无效的节点号
32	CAN 网络通信错误
33	ASCII 命令语法错误
34	驱动器内部错误

注意，索引器程序启动时，所有的寄存器都被清零。

附录 B：回零点方法描述

该附录对索引器程序中回零函数的几种回零方法进行详细描述。

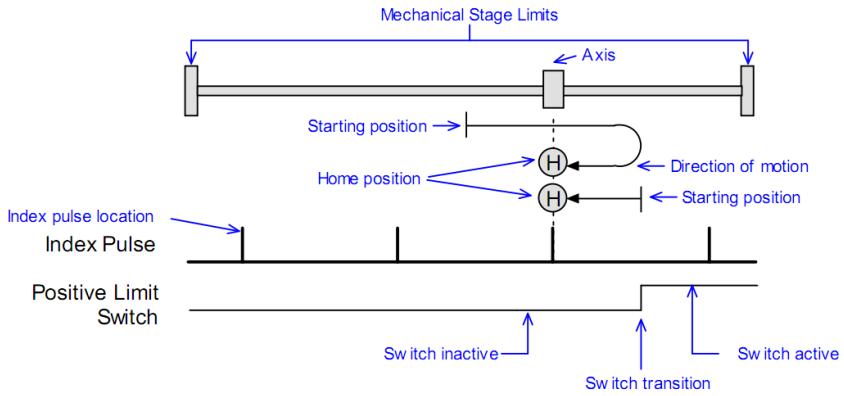
B.1. 回零点方法总述

回零方法有多种，每种方法都会建立如下条件：

- 参考零点（限位或回零开关的电平跳变，或者编码器的索引脉冲）
- 运动方向，以及和限位/回零开关索引脉冲的关系

B.2. 回零点方法图例说明

下面的每个回零方法的示意图都给出了机械段的起始位置。箭头表示运动方向，圆圈中加一个 H 表示零点位置。索引脉冲行的粗竖条表示索引脉冲位置。竖条下面的加粗黑色横线无实意，只是为了增加视觉上的理解。最后，给出了相应的限位开关，其有效区和无效区，以及跳变点。



注意在下面的回零方法描述中，负方向的运动都是向左的，正向运动都是向右的。

B.3. 回零点方法描述

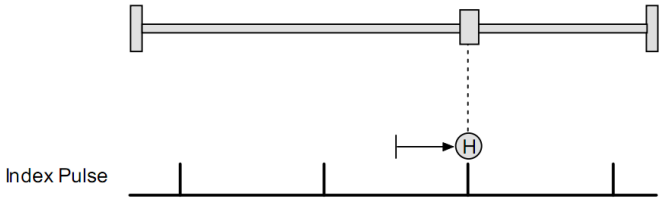
B.3.1. 设置当前位置为零点（Set current position as home）

当前位置是零点位置

B.3.2. 下一个索引脉冲（Next Index）

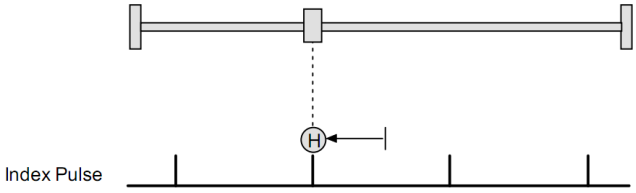
运动方向：正

正向检测到的第一个索引脉冲作为零点。运动方向为正。如果检测到第一个索引脉冲前，正向限位开关有效，则产生序列错误。



运动方向：负

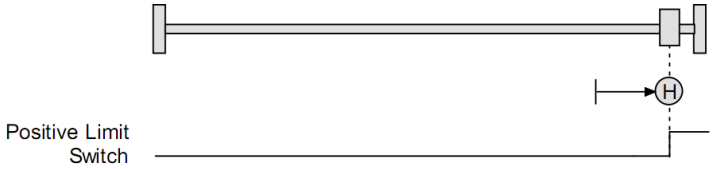
负向检测到的第一个索引脉冲作为零点。运动方向为负。如果检测到第一个索引脉冲前，负向限位开关有效，则产生序列错误。



B.3.3. 限位开关（Limit Switch）

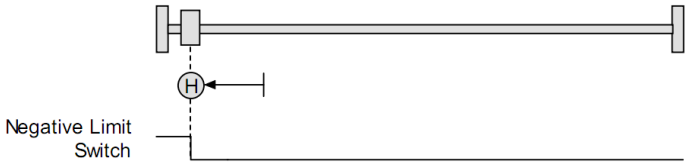
运动方向：正

正向限位开关的跳变点作为零点。初始条件是正向限位开关无效，运动方向为正。



运动方向：负

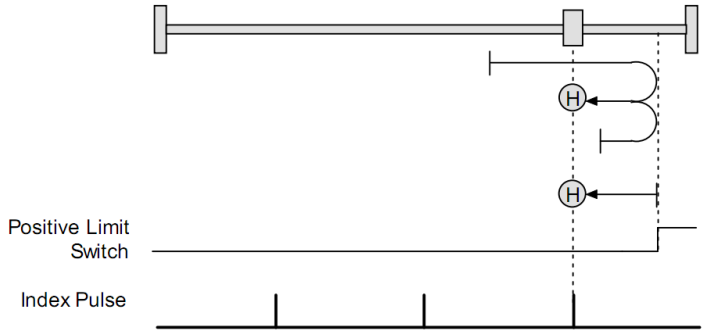
负向限位开关的跳变点作为零点。初始条件是负向限位开关无效，运动方向为负。



B.3.4. 碰到限位开关后反向第一个索引脉冲点（Limit Switch Out to Index）

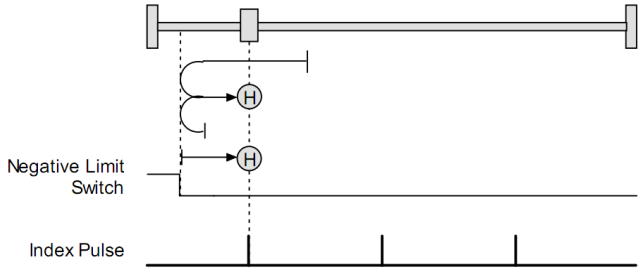
运动方向：正

碰到正向限位开关信号后，反向运动直至接收到第一个索引脉冲，将该点作为零点。初始条件是正向限位开关无效，运动方向为正（如下图所示）。



运动方向：负

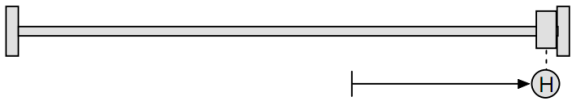
碰到负向限位开关信号后，反向运动直至接收到第一个索引脉冲，将该点作为零点。初始条件是负向限位开关无效，运动方向为负（如下图所示）。



B.3.5. 硬停止（Hardstop）

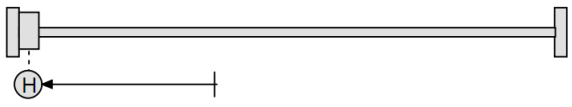
运动方向：正

正向最大行程处作为零点。硬停止指的是指定时间内，电机连续为最大电流但是位置不再增加的地方（通常也即到达了最大行程的地方）。如果在硬停止前正向限位开关有效，则产生序列错误。



运动方向：负

负向最大行程处作为零点。硬停止指的是指定时间内，电机连续为最大电流但是位置不再增加的地方（通常也即到达了最大行程的地方）。如果在硬停止前负向限位开关有效，则产生序列错误。



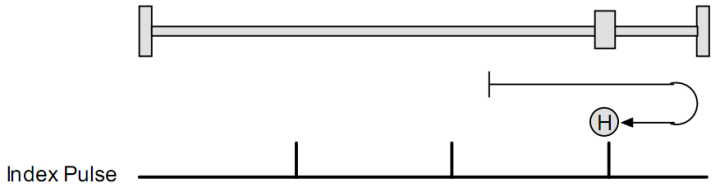
步进电机中的硬停止方法

工作在带有增量式编码器的步进模式中的驱动器，当发生跟随错误时即认为达到硬停止条件。步进模式下使用硬停止方法时，请不要禁止跟随错误（Following error）。

B.3.6. 硬停止后反向第一个索引脉冲（Hardstop Out to Index）

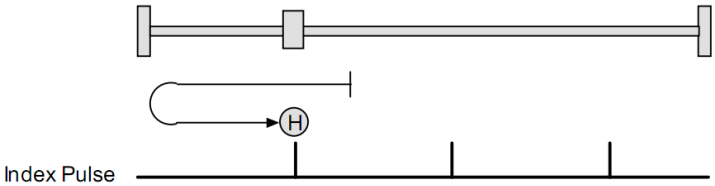
运动方向：正

硬停止后反向运动，接收到第一个索引脉冲时的位置作为零点。硬停止指的是指定时间内，电机连续为最大电流但是位置不再增加的地方（通常也即到达了最大行程的地方）。如果在硬停止前正向限位开关有效，则产生序列错误。



运动方向：负

硬停止后反向运动，接收到第一个索引脉冲时的位置作为零点。硬停止指的是指定时间内，电机连续为最大电流但是位置不再增加的地方（通常也即到达了最大行程的地方）。如果在硬停止前负向限位开关有效，则产生序列错误。



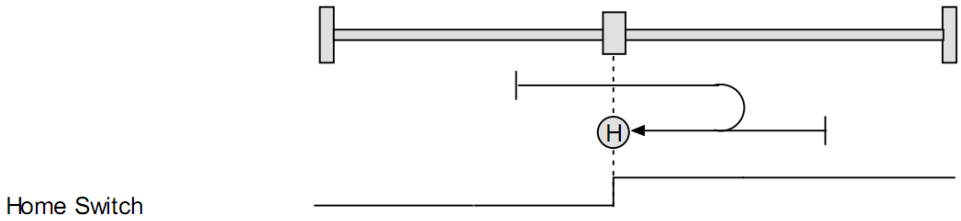
步进电机中的硬停止方法

工作在带有增量式编码器的步进模式中的驱动器，当发生跟随错误时即认为达到硬停止条件。步进模式下使用硬停止方法时，请不要禁止跟随错误（Following error）。

B.3.7. 零点开关（Home Switch）

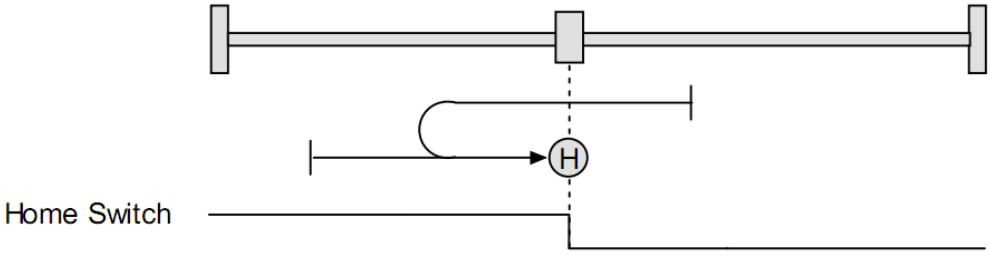
运动方向：正

零点开关电平发生跳变时为零点。初始运动方向为正向，如果零点开关无效。如果零点开关电平跳变前，正向限位开关有效，则产生序列错误。



运动方向：负

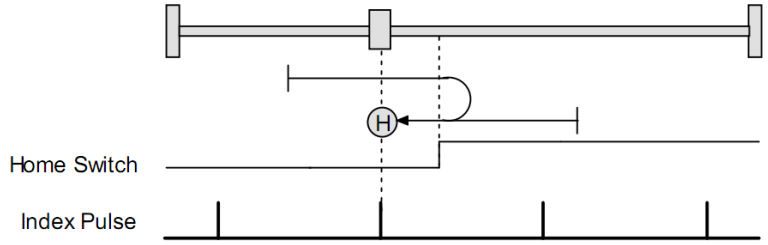
零点开关电平发生跳变时为零点。初始运动方向为负向，如果零点开关无效。如果零点开关电平跳变前，负向限位开关有效，则产生序列错误。



B.3.8. 零点开关输出到第一个索引脉冲（Home Switch Out to Index）

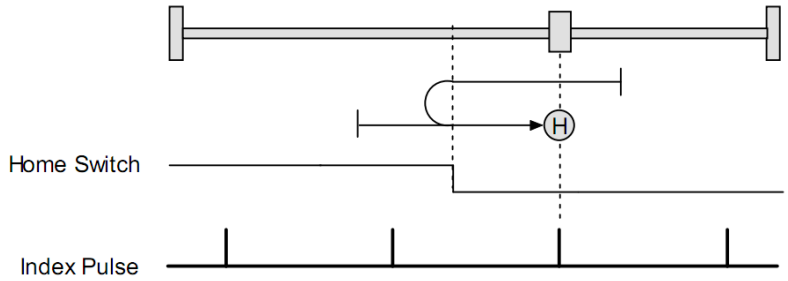
运动方向：正

检测到零点开关跳变后反向运动，接收到第一个索引脉冲时的位置作为零点。运动的初始方向为正，如果零点开关是无效状态。如果零点开关电平跳变前，正向限位开关有效，则产生序列错误。



运动方向：负

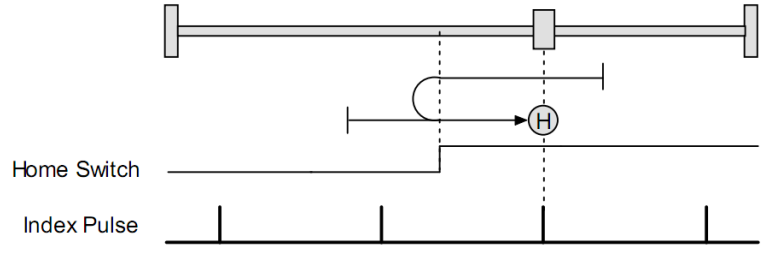
检测到零点开关跳变后反向运动，接收到第一个索引脉冲时的位置作为零点。运动的初始方向为负，如果零点开关是无效状态。如果零点开关电平跳变前，负向限位开关有效，则产生序列错误。



B.3.9. 零点开关输入到第一个索引脉冲（Home Switch In to Index）

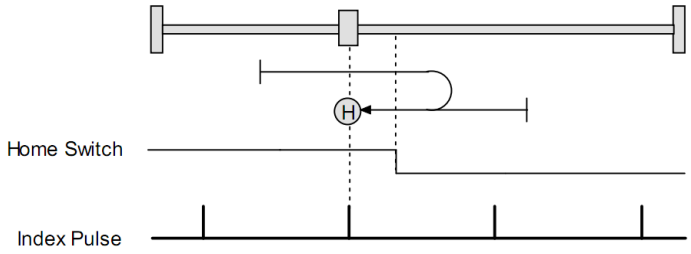
运动方向：正

检测到零点开关跳变后继续正向运动，当接收到第一个索引脉冲时的位置作为零点。运动的初始方向为正，如果零点开关是无效状态。如果零点开关电平跳变前，正向限位开关有效，则产生序列错误。



运动方向：负

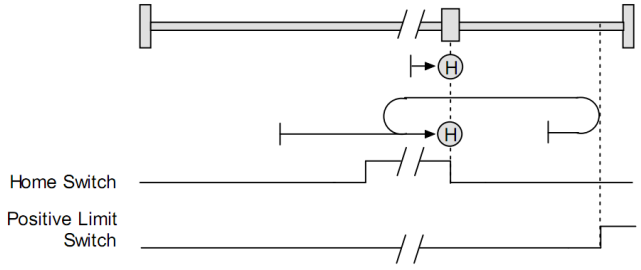
检测到零点开关跳变后继续负向运动，当接收到第一个索引脉冲时的位置作为零点。运动的初始方向为负，如果零点开关是无效状态。如果零点开关电平跳变前，负向限位开关有效，则产生序列错误。



B.3.10. 零点开关下降沿（Lower Home）

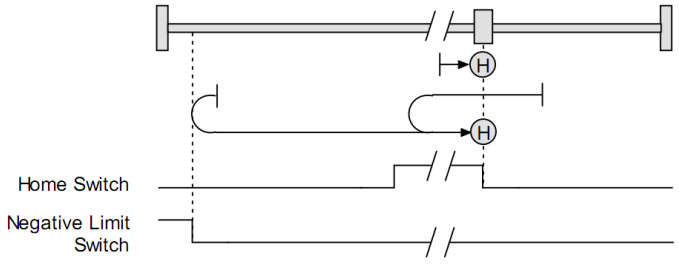
运动方向：正

检测到零点开关电平的下降沿时的位置作为零点。运动的初始方向为正，如果零点开关是无效状态。如果零点开关电平跳变前，正向限位开关有效，则会反向运动。如果零点开关有效前，负限位开关有效，则会产生错误。



运动方向：负

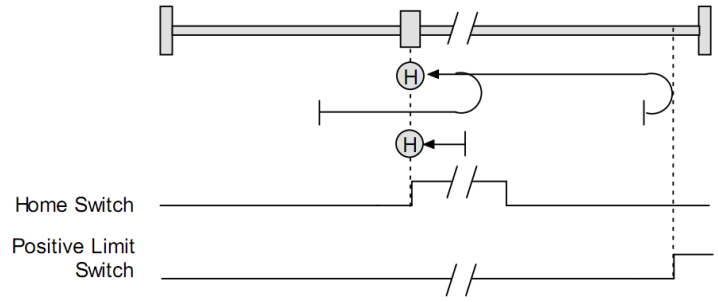
检测到零点开关电平的下降沿时的位置作为零点。运动的初始方向为负，如果零点开关是无效状态。如果零点开关电平跳变前，负向限位开关有效，则会反向运动。如果零点开关有效前，正限位开关有效，则会产生错误。



B.3.11. 零点开关上升沿（Upper Home）

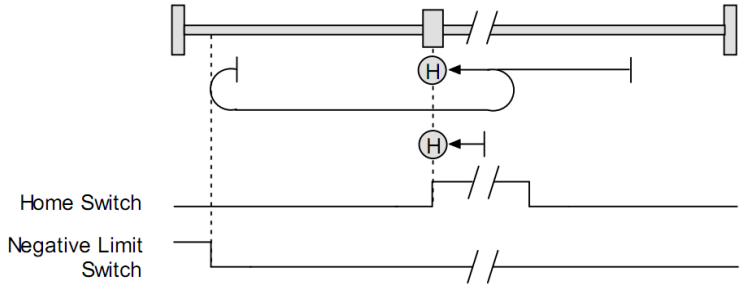
运动方向：正

检测到零点开关电平的上升沿时的位置作为零点。运动的初始方向为正，如果零点开关是无效状态。如果零点开关电平跳变前，正向限位开关有效，则会反向运动。如果零点开关有效前，负限位开关有效，则会产生错误。



运动方向：负

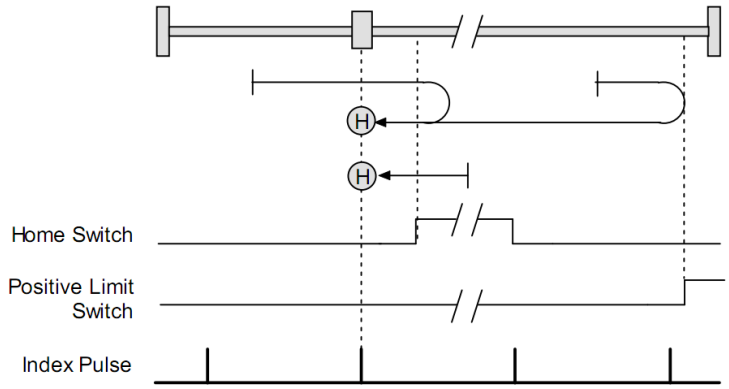
检测到零点开关电平的上升沿时的位置作为零点。运动的初始方向为负，如果零点开关是无效状态。如果零点开关电平跳变前，负向限位开关有效，则会反向运动。如果零点开关有效前，正限位开关有效，则会产生错误。



B.3.12. 零点开关下降沿外第一个索引脉冲（Lower Home Outside to Index）

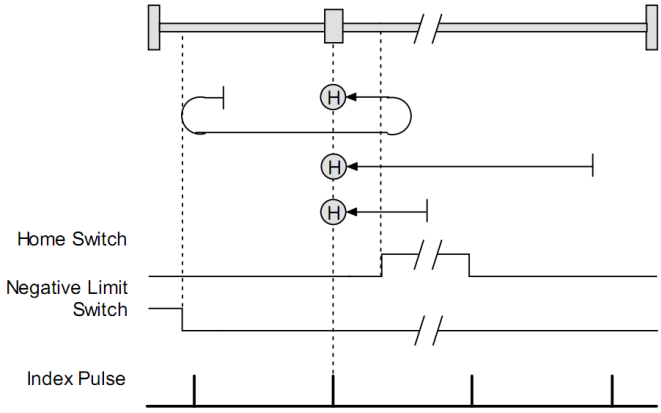
运动方向：正

检测到零点开关下降沿的负边后的第一个索引脉冲处的位置作为零点。运动的初始方向为正，如果零点开关是无效状态。如果初始运动方向是背离零点开关方向的，轴会在遇到正限位开关信号后反向运动。然后如果零点开关有效前，负限位开关有效，则会产生错误。



运动方向：负

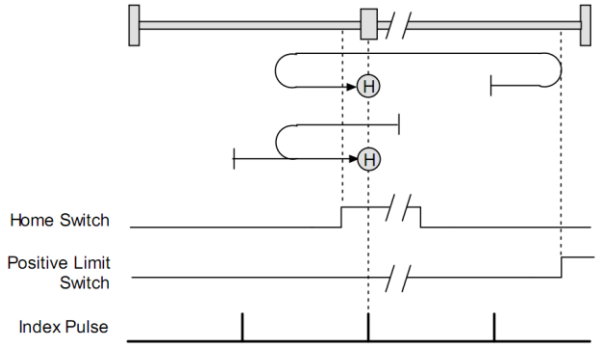
检测到零点开关下降沿的负边后的第一个索引脉冲处的位置作为零点。运动的初始方向为负，如果零点开关是无效状态。如果初始运动方向是背离零点开关方向的，轴会在遇到负限位开关信号后反向运动。然后如果零点开关有效前，正限位开关有效，则会产生错误。



B.3.13. 零点开关下降沿内第一个索引脉冲（Lower Home Inside to Index）

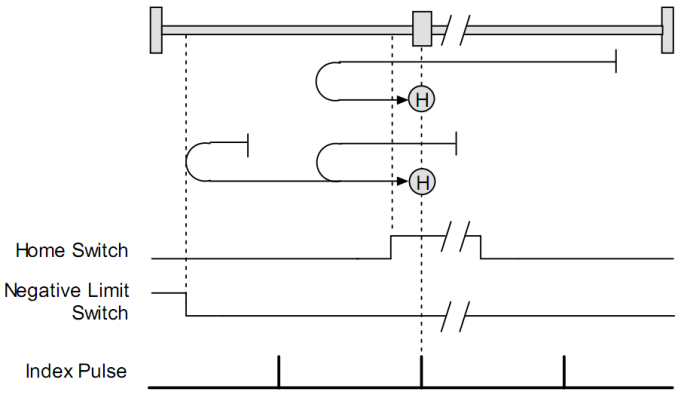
运动方向：正

检测到零点开关下降沿的正边后的第一个索引脉冲处的位置作为零点。运动的初始方向为正，如果零点开关是无效状态。如果初始运动方向是背离零点开关方向的，轴会在遇到正限位开关信号后反向运动。然后如果零点开关有效前，负限位开关有效，则会产生错误。



运动方向：负

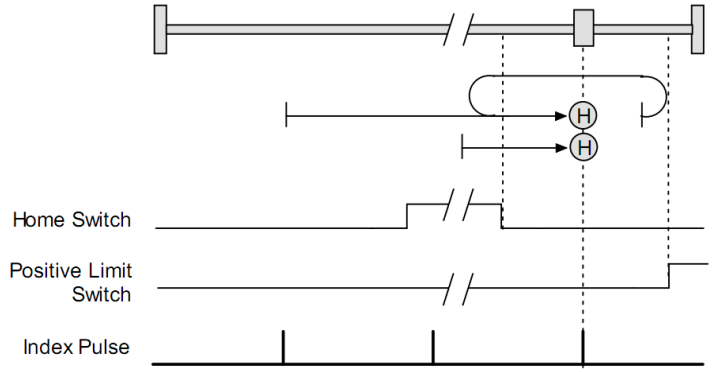
检测到零点开关下降沿的负边后的第一个索引脉冲处的位置作为零点。运动的初始方向为负，如果零点开关是无效状态。如果初始运动方向是背离零点开关方向的，轴会在遇到负限位开关信号后反向运动。然后如果零点开关有效前，正限位开关有效，则会产生错误。



B.3.14. 零点开关上升沿外的第一个索引脉冲（Upper Home Outside to Index）

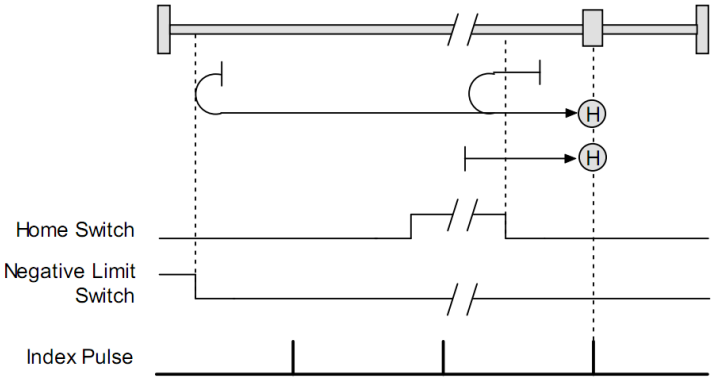
运动方向：正

检测到零点开关上升沿后的正边的第一个索引脉冲处的位置作为零点。运动的初始方向为正。如果初始运动方向是背离零点开关方向的，轴会在遇到正限位开关信号后反向运动。然后如果零点开关有效前，负限位开关有效，则会产生错误。



运动方向：负

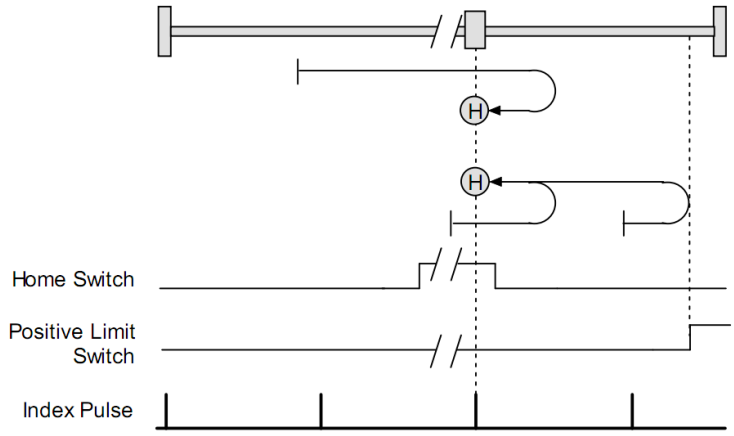
检测到零点开关上升沿的负边后的第一个索引脉冲处的位置作为零点。运动的初始方向为负，如果零点开关是无效状态。如果初始运动方向是背离零点开关方向的，轴会在遇到负限位开关信号后反向运动。然后如果零点开关有效前，正限位开关有效，则会产生错误。



B.3.15. 零点开关上升沿内的第一个索引脉冲（Upper Home Inside to Index）

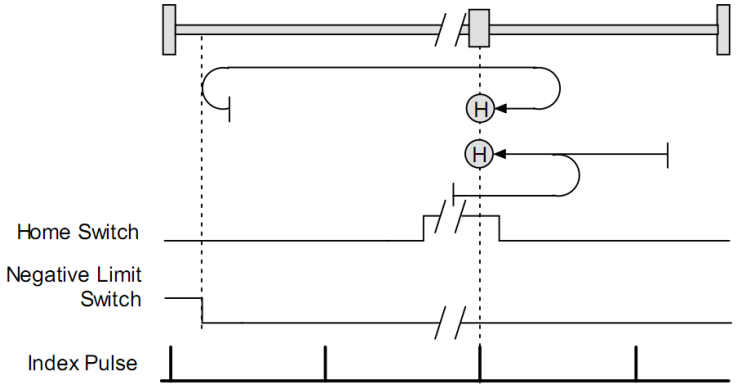
运动方向：正

检测到零点开关上升沿后的负边的第一个索引脉冲处的位置作为零点。运动的初始方向为正，如果零点开关是无效状态。如果初始运动方向是背离零点开关方向的，轴会在遇到正限位开关信号后反向运动。然后如果零点开关有效前，负限位开关有效，则会产生错误。



运动方向：负

检测到零点开关上升沿的负边后的第一个索引脉冲处的位置作为零点。运动的初始方向为负，如果零点开关是无效状态。如果初始运动方向是背离零点开关方向的，轴会在遇到负限位开关信号后反向运动。然后如果零点开关有效前，正限位开关有效，则会产生错误。



深圳市泰科智能智能伺服技术有限公司

Techservo(ShenZhen)Co., LTD.

地址：深圳市南山区科技园中区麻雀岭工业区 M-4 栋深健大厦 5D1-1
TEL：0755-26712201　26712221
FAX：0755-26712958
E-mail：sales@techservo.com
网站：http://www.techservo.com