凌臣 PLC 高速 IO 说明书

版本说明

版本	时间	修改内容
V1.2.0	2024 年 10 月 23 日	初版
V1.3.0	2025 年 1 月 22 日	软件更新，修改部分参数
V1.4.0	2025 年 2 月 14 日	完善 CC_Counter 指令说明
V1.5.0	2025 年 5 月 26 日	新脉冲方案说明书初版
V1.6.0	2025 年 7 月 31 日	新高速计数脉冲方案说明书初版（修正）
V1.7.0	2026 年 1 月 31 日	新高速计数脉冲方案说明书

目录

第一章 高速 IO 功能

特别说明：

高速 IO 新方案与旧方案在**软件上不兼容**，硬件适配 LC1500 系列和 LC1800P 系列（包括 LC1800_NOTV 与 LC1800_TV 系列）。**需要注意新版本的设备描述文件和库都是全新的。**

更新说明：

计数器	编码器计数器 1*8（不变）
比较器	通用比较器 8 路（不变）； 步长/数组比较器变为 7 个（**比原来减少 1 个**）；
编码器探针	编码器探针 2*8（不变）
脉冲轴	脉冲运控轴 8 路（**比原来增加 4 路**）
新增功能	EtherCAT 探针 2*4； 脉冲输出探针 2*4

S 系列 PLC（S500,S300）

注意事项：新系列 PLC **删除了步长/数组比较器，减少了速度采样 simple，计数器预置去掉窗口滤波**

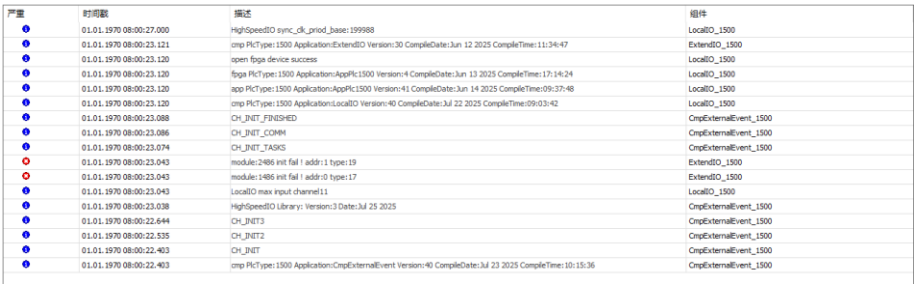
计数器	编码器计数器 4
比较器	通用比较器 2 路； 步长/数组比较器变为 0 个；
编码器探针	编码器探针 2*4
脉冲轴	脉冲运控轴 5 路
新增功能	EtherCAT 探针 2*2； 脉冲输出探针 2*2

版本识别：

XML	c617 及更高版本
库	HighSpeedIO（Version:3 Date:Jun 25 2025）版本及更高版本

底层组件版本	LC1500 系列：V27 及更高版本 LC1800P 系列：V21 及更高版本 S500/S300 系列：V30 及更高版本

注：可以在日志页面查看各个组件和 Lib 库的版本时间信息。

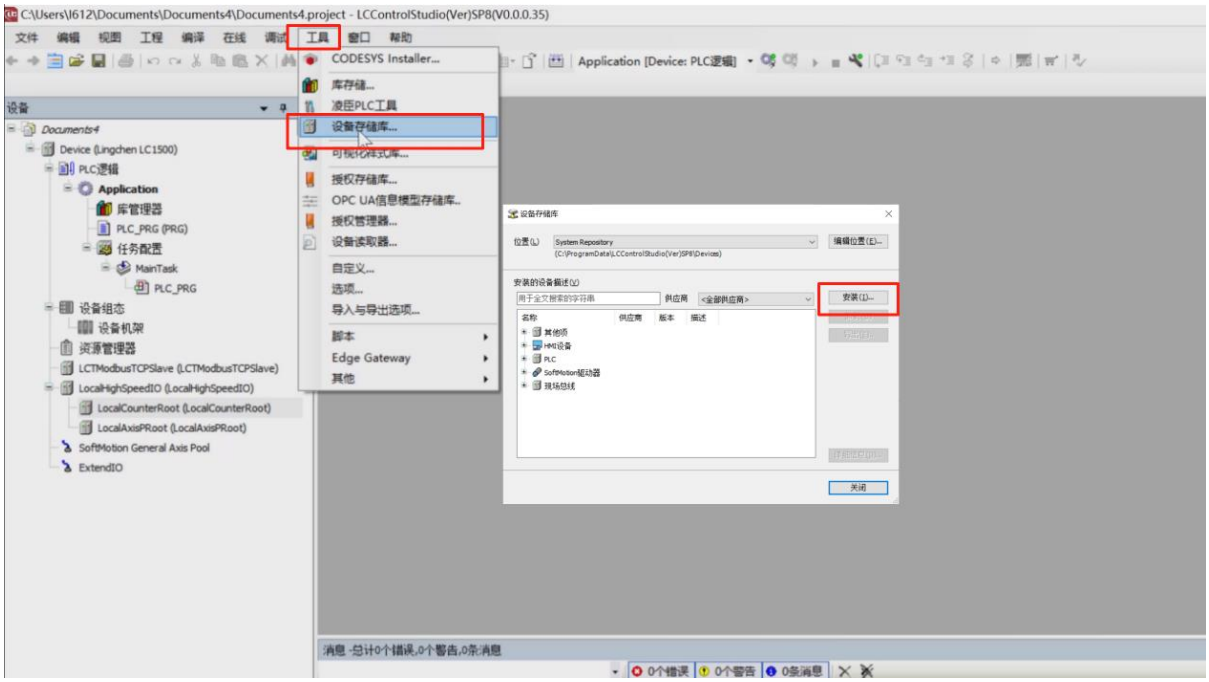


严重	时间戳	描述	组件
ℹ	01.01.1970 08:00:27.000	HighSpeedIO sync_clk_priod_base: 199988	LocalIO_1500
ℹ	01.01.1970 08:00:23.121	cmp PlcType:1500 Application:ExtendIO Version:30 CompileDate:Jun 12 2025 CompileTime:11:34:47	ExtendIO_1500
ℹ	01.01.1970 08:00:23.120	open fpga device success	LocalIO_1500
ℹ	01.01.1970 08:00:23.120	fpga PlcType:1500 Application:AppPlc1500 Version:4 CompileDate:Jun 13 2025 CompileTime:17:14:24	LocalIO_1500
ℹ	01.01.1970 08:00:23.120	app PlcType:1500 Application:AppPlc1500 Version:41 CompileDate:Jun 14 2025 CompileTime:09:37:48	LocalIO_1500
ℹ	01.01.1970 08:00:23.120	cmp PlcType:1500 Application:LocalIO Version:40 CompileDate:Jul 22 2025 CompileTime:09:03:42	LocalIO_1500
ℹ	01.01.1970 08:00:23.088	CH_INIT_FINISHED	CmpExternalEvent_1500
ℹ	01.01.1970 08:00:23.086	CH_INIT_COMM	CmpExternalEvent_1500
ℹ	01.01.1970 08:00:23.074	CH_INIT_TASKS	CmpExternalEvent_1500
✖	01.01.1970 08:00:23.043	module:2486 init fail ! addr:1 type:19	ExtendIO_1500
✖	01.01.1970 08:00:23.043	module:1486 init fail ! addr:0 type:17	ExtendIO_1500
ℹ	01.01.1970 08:00:23.043	LocalIO max input channel11	LocalIO_1500
ℹ	01.01.1970 08:00:23.038	HighSpeedIO Library: Version:3 Date:Jul 25 2025	CmpExternalEvent_1500
ℹ	01.01.1970 08:00:22.644	CH_INIT3	CmpExternalEvent_1500
ℹ	01.01.1970 08:00:22.535	CH_INIT2	CmpExternalEvent_1500
ℹ	01.01.1970 08:00:22.403	CH_INIT	CmpExternalEvent_1500
ℹ	01.01.1970 08:00:22.403	cmp PlcType:1500 Application:CmpExternalEvent Version:40 CompileDate:Jul 23 2025 CompileTime:10:15:36	CmpExternalEvent_1500

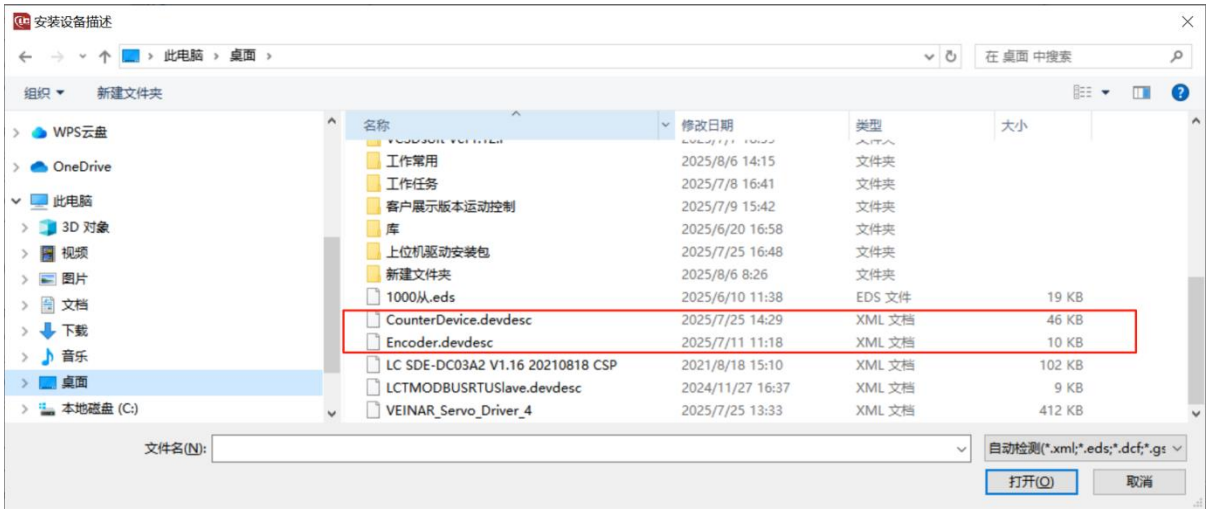
1.1 计数器配置

1.1.1 库文件安装及应用

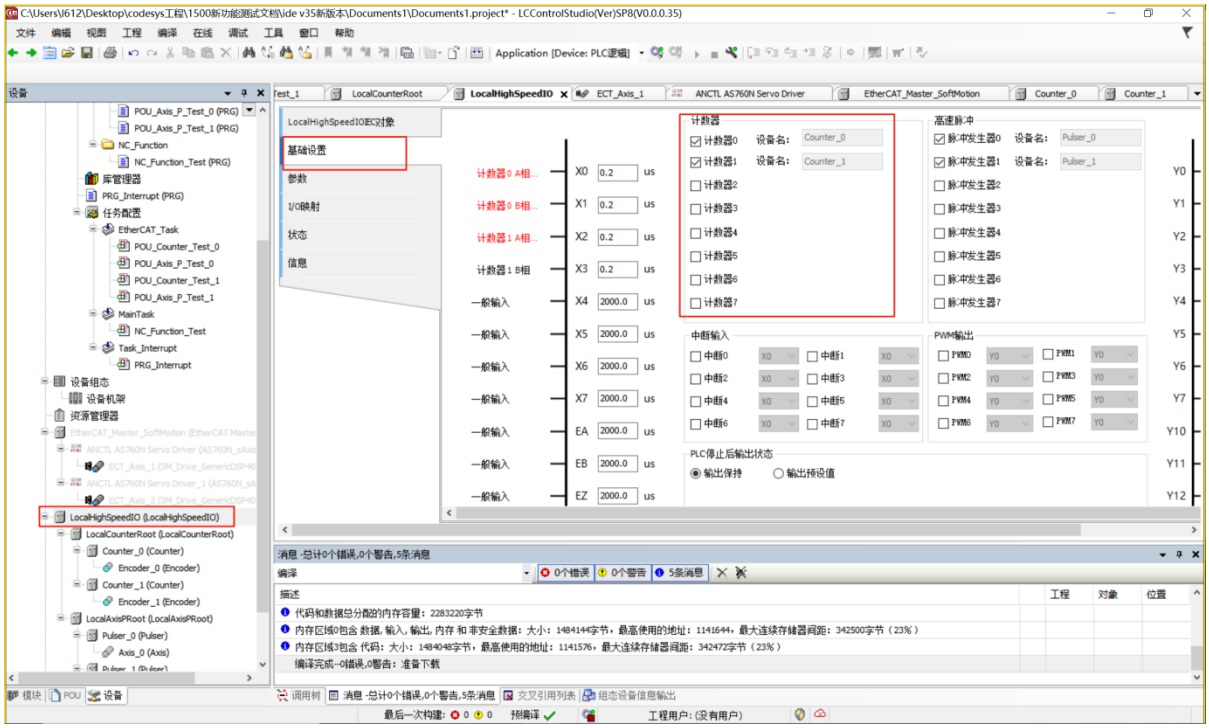
点击工具栏“工具”选项，选择“设备存储库”，然后点击“安装”



选择“CounterDevice.devdesc”和“Encoder.devdesc”文件，点击打开进行安装。



选择“LocalHighSpeedIO”，选择“基础设置”，在右侧的图形化界面中选择“计数器 0”，选择后可以在“LocalHighSpeedIO”下看到添加的“Counter_0”以及“Encoder_0”



1.1.2 计数器参数配置

点击计数器“Counter_0”,对“Counter_0”进行参数设置，配置“计数器模式”、“信号源”、“外部触发输出”、“预置计数器”等

1.2 计数器功能

COUNTER 参数说明。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
CounterID	计数器 ID	USINT	[0,7]	0	计数器 ID
CounterPort	计数器输入硬件编号	USINT	[0,15]	0	计数器输入硬件编号 CounterType=5 单脉冲输入，CounterPort 范围是 0-7：X0-X7；8：EA；9：EB；10：EZ**(LC1500)**； 0-7：X0-X7；8：EA1；9：EB1；10：EA1；11：EB2**(LC1800P)**。 单脉冲输入模式下，0~11 是真实的输入端口，12 是虚拟 1us，13 是虚拟 1ms 其它 CounterType 选项。 0：X0-X1； 1：X2-X3； 2：X4-X5； 3：X6-X7； 4：EA-EB（1500 系列）/EA1-EB1（1800P 系列）； 5：EA2-EB2（仅限 1800P 系列） 14 与 15 无实际意义，预留。 本说明书以 LC1500 系列为主，后不多做赘述。
CounterType	计数器类型	USINT	[0,5]	2	0：AB 相一倍频计数 1：AB 相二倍频计数 2：AB 相四倍频计数 3：CW+CCW 正逻辑 4：脉冲+方向正逻辑 5：单脉冲
CounterDIStartNum	DI 启动编号	USINT	[0,10]	0	0-7：X0-X7；8：EA；9：EB；10：EZ
PresetCounterEnable	计数器预置使能	BOOL	[FALSE, TRUE]	FALSE	计数器预置使能。
PresetCounterDINum	计数器预置 DI 输入端口	USINT	[0,10]	0	计数器预置 DI 输入端口 0-7：X0-X7； 8：EA；9：EB；10：EZ
SetCompareEnable	比较一致输出使能	BOOL	[FALSE, TRUE]	FALSE	比较一致输出使能
SetCompareDONum	比较一致 DO 端口	USINT	[0,15]	0	比较一致 DO 端口 0-7：Y0-Y7； 8-15：Y10-Y17
SetArrayCompareEnable	数组比较一致输出使能	BOOL	[FALSE, TRUE]	FALSE	数组比较一致输出使能
SetArrayCompareDONum	数组比较一致 DO 端	USINT	[0,15]	0	数组比较一致 DO 端口

	口				0-7：Y0-Y7； 8-15：Y10-Y17
SetStepCompareEnable	步长比较一致输出使能	BOOL	[FALSE, TRUE]	FALSE	步长比较一致输出使能
SeStepCompareDONum	步长比较一致 DO 端口	USINT	[0,15]	0	步长比较一致 DO 端口 0-7：Y0-Y7； 8-15：Y10-Y17
TouchProbe0Enable	探针 0 使能	BOOL	[FALSE, TRUE]	FALSE	探针 0 使能
TouchProbe0DINum	探针 0 DI 输入端口	USINT	[0,10]	0	探针 0DI 输入端口 0-7：X0-X7； 8：EA；9：EB；10：EZ
TouchProbe1DINum	探针 1DI 输入端口	USINT	[0,10]	0	探针 1DI 输入端口 0-7：X0-X7； 8：EA；9：EB；10：EZ
TouchProbe1Enable	探针 1 使能	BOOL	[FALSE, TRUE]	FALSE	探针 1 使能
HardwareResetEnable	硬件复位使能	BOOL	[FALSE, TRUE]	FALSE	硬件复位使能
HardwareResetDINum	硬件复位 DI 输入端口	USINT	[0,10]	0	硬件复位 DI 输入端口号 0-7：X0-X7； 8：EA；9：EB；10：EZ
HardwareResetMode	硬件复位模式	USINT	[0,1]	0	0：DI 上升沿复位； 1：DI 下降沿复位
VirtualTouchProbe0Enable	虚拟探针 0 使能	BOOL	[FALSE, TRUE]	FALSE	虚拟探针 0 使能
VirtualTouchProbe0DINum	虚拟探针 0 DI 输入端口	USINT	[0,10]	0	虚拟探针 0DI 输入端口 0-7：X0-X7； 8：EA；9：EB；10：EZ
VirtualTouchProbe1Enable	虚拟探针 1 使能	BOOL	[FALSE, TRUE]	FALSE	虚拟探针 1 使能
VirtualTouchProbe1DINum	虚拟探针 1DI 输入端口	USINT	[0,10]	0	虚拟探针 1DI 输入端口 0-7：X0-X7； 8：EA；9：EB；10：EZ

ENCODER 参数说明。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
CounterEnable	计数器使能	BOOL	[FALSE, TRUE]	TRUE	计数器使能
wDriveID	计数器 ID	USINT	[0,7]	0	计数器 ID，必须要与 Counter 父设备保持一致
iMovementType	计数器计数模式	INT	[0,1]	1	计数器计数模式：0：模数，1：线性
fPositionPeriod	计数器模数值	LREAL	>0	360.0	计数器模数值
numerator	齿轮比分子	UDINT	(0,	1	齿轮比分子

			2147483647)		脉冲数 pluse=numerator/denominator * 移动距离(用户单位 unit)
denominator	齿轮比分母	LREAL	>0	1.0	齿轮比分母 脉冲数 pluse=numerator/denominator * 移动距离(用户单位 unit)

计数器错误 ID 说明（CC_ERROR）

错误代码	枚举值	说明
000	CC_NO_ERROR	无错误
100	CC_ERROR_COMMUNICATION	通讯故障
101	CC_ERROR_COUNTER_DISABLE	计数器未使能
105	CC_ERROR_NOT_CONFIG	图形化界面未配置 DI 或者 DO
150	CC_ERROR_DO_CONFIGURE_ACTIVE	DO 端口配置冲突
151	CC_ERROR_DO_CONFIGURE_INVALID	DO 端口配置无效
152	CC_ERROR_DI_CONFIGURE_INVALID	DI 端口配置无效
153	CC_ERROR_NOT_SUPPORT_ONLINE_CHANGE	计数器在线修改错误
154	CC_ERROR_COUNTER_ID_INVALID	计数器 ID 不合理
155	CC_ERROR_COUNTER_PORT_INVALID	计数器输入硬件端口配置不合理
156	CC_ERROR_COUNTER_TYPE_INVALID	计数器计数方式不合理
157	CC_ERROR_COUNTER_START_MODE_INVALID	启动模式不合理
158	CC_ERROR_PV_MODE_INVALID	预置，预置类型不合理
159	CC_ERROR_PV_PRESET_VALUE_INVALID	预置，预置值不合理
160	CC_ERROR_PV_PRESET_LENGTH_INVALID	预置，预置 DI 信号长度不合理
161	CC_ERROR_CP_MODE_INVALID	比较器模式不合理
162	CC_ERROR_CP_COMPARE_VAULE_INVALID	比较位置不合理
163	CC_ERROR_CP_START_POS_INVALID	等距比较，开始位置不合理
164	CC ERROR_CP_END_POS_INVALID	等距比较，结束位置不合理
165	CC_ERROR_CP_STEP_LEN_INVALID	等距比较，步长不合理
166	CC_ERROR_CP_ARRAY_ADDR_INVALID	多点比较，数组首地址不合理
167	CC_ERROR_CP_ARRAY_COUNT_INVALID	多点比较，数组个数不合理
168	CC_ERROR_CP_OUTPUT_TYPE_INVALID	比较一致,DO 输出类型不合理
169	CC_ERROR_CP_OUTPUT_VALUE_INVALID	比较一致,DO 输出保持参数不合理
170	CC_ERROR_TP_ID_INVALID	探针,ID 不合理
171	CC_ERROR_TP_TRGIGGER_TYPE_INVALID	探针,触发类型超出范围
172	CC ERROR_TP_EDG_TYPE_INVALID	探针,边沿触发类型超出范围
173	CC_ERROR_TP_MODE_INVALID	探针,触发模式值超出范围
174	CC_ERROR_SAMEPLE_TIME_INVALID	采用时间不合理
175	CC_ERROR_SAMEPLE_FILTER_INVALID	滤波值不合理
176	CC_ERROR_MEASURE_PULSE_MODE_INVALID	测量脉宽模式不合理
177	CC_ERROR_INTERRUPT_MODE_INVALID	中断模式不合理
178	CC_ERROR_INTERRUPT_COMPAREEVENT_INVALID	中断计数器 ID 配置不合理
179	CC_ERROR_INTERRUPT_EVENTHANDLE_CONFIGURE_INVALID	中断函数句柄无效
180	CC_ERROR_PWM_PERIOD_INVALID	PWM 周期不合理

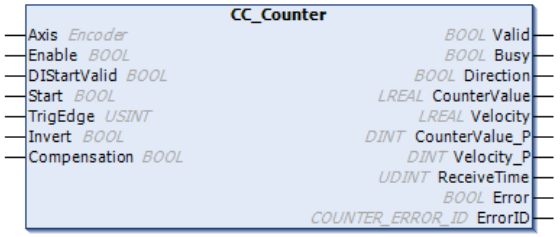
181	CC_ERROR_PWM_WIDTH_INVALID	PWM 占空比不合理
182	CC_ERROR_READPARA_NUM_INVALID	读计数器参数编号无效
183	CC_ERROR_WRITEPARA_NUM_INVALID	写计数器参数编号无效
184	CC_ERROR_DEVICE_NOT_IN_USE	设备未使用
185	CC_ERROR_FB_CANNOT_INTERRUPT	指令不能被打断
186	CC_ERROR_ENABLE_UNIQUE_INSTANTIATION	Enable 类型指令只能存在唯一实例

指令列表

序号	指令名称	FB/FC	功能
1.2.1	CC_Counter	FB	计数器
1.2.2	CC_PresetCounter	FB	计数器预置
1.2.3	CC_SetCompare	FB	比较器
1.2.4	CC_SetArrayCompare	FB	数组比较器
1.2.5	CC_SetStepCompare	FB	步长比较器
1.2.6	CC_TouchProbe	FB	计数器探针
1.2.7	Virtual_TouchProbe	FB	Ethercat 总线轴探针
1.2.8	CC_MeasurePulseWidth	FB	脉宽测量
1.2.9	CC_Sample	FB	计数器采样
1.2.10	CC_ReadParameter	FB	计数器读参数（保留）
1.2.11	CC_WriteParameter	FB	计数器写参数（保留）

1.2.1 CC_Counter

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
CC_Counter	计数器	FB		CC_Counter(Axis:= , Enable:= , DIStartValid:= , Start:= , TrigEdge:= , Invert:= , Compensation:= , Valid=> , Busy=> , Direction=> , CounterValue=> , Velocity=> , CounterValue_P=> , Velocity_P=> , ReceiveTime=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Encoder	-	-	计数器编号 [0..7]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Enable	使能	BOOL	[FALSE, TRUE]	FALSE	电平触发，TRUE：使能计数器。
Start	软件启动信号	BOOL	[FALSE, TRUE]	FALSE	0→1（上升沿）跳变：启动计数器。
DIStartValid	DI 启动计数器	USINT	[FALSE, TRUE]	FALSE	0：无效-软件启动计数器； 1：有效-DI 启动计数器。
TrigEdge	DI 启动方式	USINT	[0,1]	0	0：上升沿；1：下降沿
Invert	计数取反	BOOL	[FALSE, TRUE]	FALSE	TRUE：反向计数。
Compensation	补偿方式	BOOL	[FALSE, TRUE]	FALSE	FALSE：用户手动补偿或者不补偿； TRUE：系统自动补偿。

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明

Valid	有效标志	BOOL	[FALSE, TRUE]	-	FALSE：计数器无效 TRUE：计数器有效
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：计数器未执行 TRUE：计数器已执行
Direction	计数方向	BOOL	[FALSE, TRUE]	-	FALSE：正方向（加法计数） TRUE： 反方向（减法计数）
CounterValue	计数器值	LREAL	-	-	计数器当前计数值，单位：unit。
Velocity	计数器速度	LREAL	-	-	计数器当前速度，单位：unit/s。
CounterValue_P	计数器值	DINT	[-2147483648, 2147483647]	-	计数器当前计数值，单位：pulse。
Velocity_P	计数器速度	DINT	[-2147483648, 2147483647]	-	计数器当前速度，单位：pulse/s。
ReceiveTime	读取计数值时刻	UDINT	[0, 4294967295]	-	读取 CounterValue 的系统时刻。
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

本功能块主要实现本地计数器开始/停止，有两种触发模式。一种是 Start 软件信号启动计数器，当 Enable 置 TRUE 后，只是使能当前计数器，并不会使计数器开始计数，只有当 DIStartValid=FALSE，且 Start 开始从 0--->1 跳变时，才会开始计数，当 Enable 置 FALSE 后，计数器实效，停止计数；另一种是硬件 DI 信号启动计数器，当 DIStartValid=TRUE 时，会根据设定的 DI 硬件接口收到信号开始启动计数器。

计数器值和计数器速度有两种输出，一种带“_P”引脚在 DINT 范围内变化，单位是 Pulse，计数器频率也即计数器速度单位为 Pulse/s；另一种则是在 LREAL 范围内变化，单位为 Unit（用户单位），计数器频率也即计数器速度单位为 Unit/s。计数器可测量的最小速度为 1s 时间内计数器 1 个脉冲对应的速度。计数器的最大输入频率是单边 200kHz。 正向计数：当计数值超过 2147483647 时，继续从 -2147483648 开始计数。

负向计数：当计数值低于-2147483648 时，继续从 2147483647 开始计数。

环形计数：iMovementType 为模数时，按照设定的 fPositionPeriod（计数器模数值）循环计数。

【注意事项】

1、当使用硬件 DI 信号启动计数器时，可以是任意输入端口，当使用差分信号口 EA、EB、EZ 时，请按照使用说明接线。

2、当使用硬件 DI 信号启动计数器时，可通过 TrigEdge 设置 DI 接口是上升沿或下降沿信号启动计数器。

3、Invert 参数可以设置计数器的计数方向，不同计数器类型的计数方向选择定义如下表所示。更改 Invert 的设置后，需要重新使能功能块指令生效。

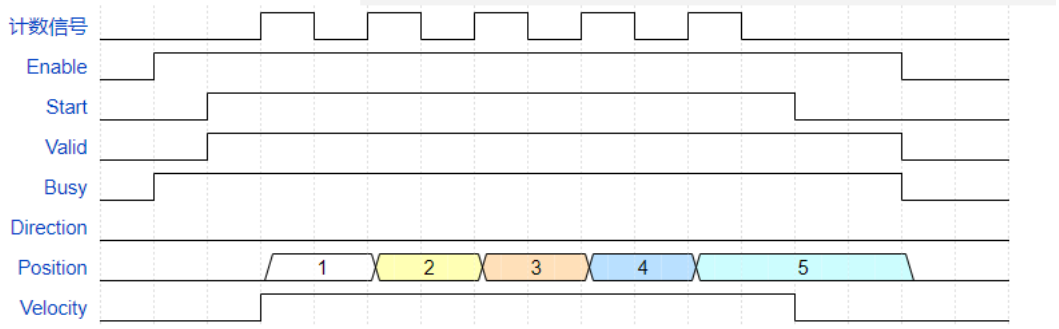
Invert	AB 相一倍、二倍、四倍频计数	CW+CCW 正逻辑	脉冲＋方向 正逻辑	单脉冲
FALSE	A 相超前 B 相增计数 B 相超前 A 相减计数	A 相超前 B 相 90° CW 增计数 B 相超前 A 相 90° CCW 减计数	方向信号低电平减计数 方向信号高电平增计数	增计数
TRUE	A 相超前 B 相减计数	A 相超前 B 相 90° CW 减计数	方向信号低电平增计数	减计数

	B 相超前 A 相增计数	B 相超前 A 相 90° CCW 增计数	方向信号高电平减计数	

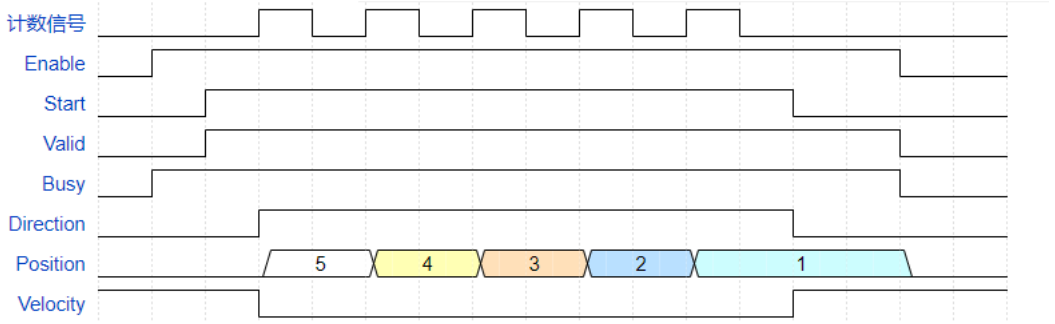
【程序示例】

CC_Counter 指令时序图，以计数器类型为“脉冲＋方向 正逻辑”为例。

方向信号=ON，Invert=FALSE 或方向信号=OFF，Invert=TRUE 时，计数器增计数，如下图所示。需要注意，Direction=False 代表正方向（加法计数），Direction=True 代表反方向（减法计数）。

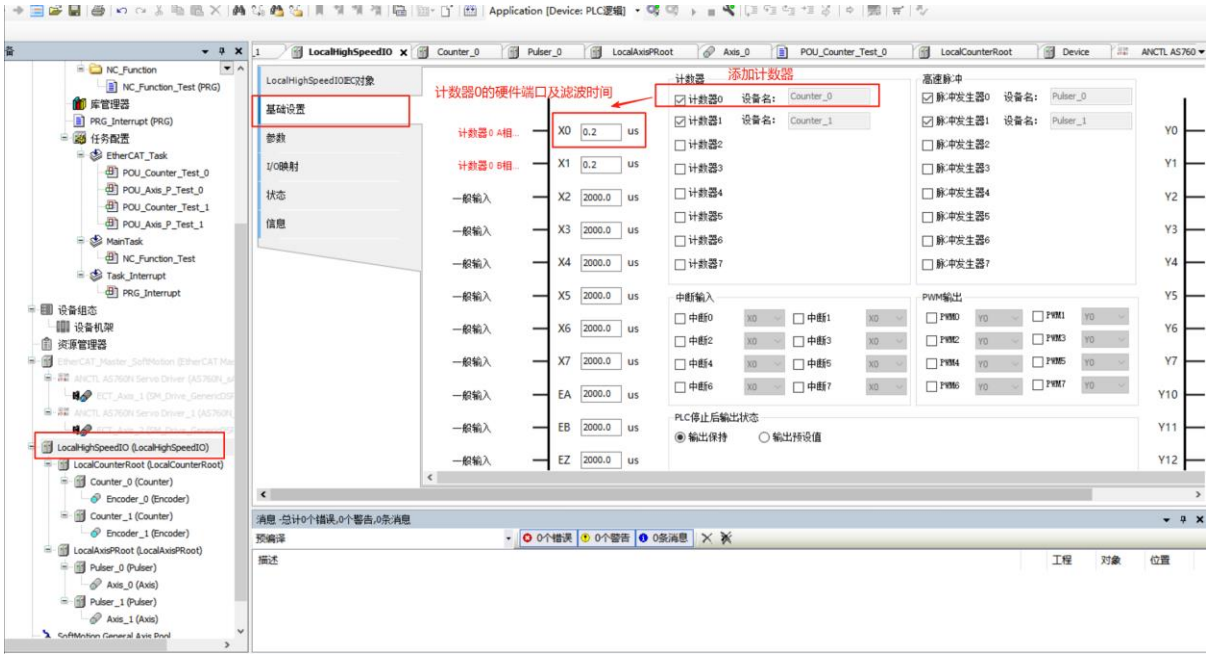


方向信号=ON，Invert=TRUE 或方向信号=OFF，Invert=FALSE 时，计数器减计数，如下图所示。



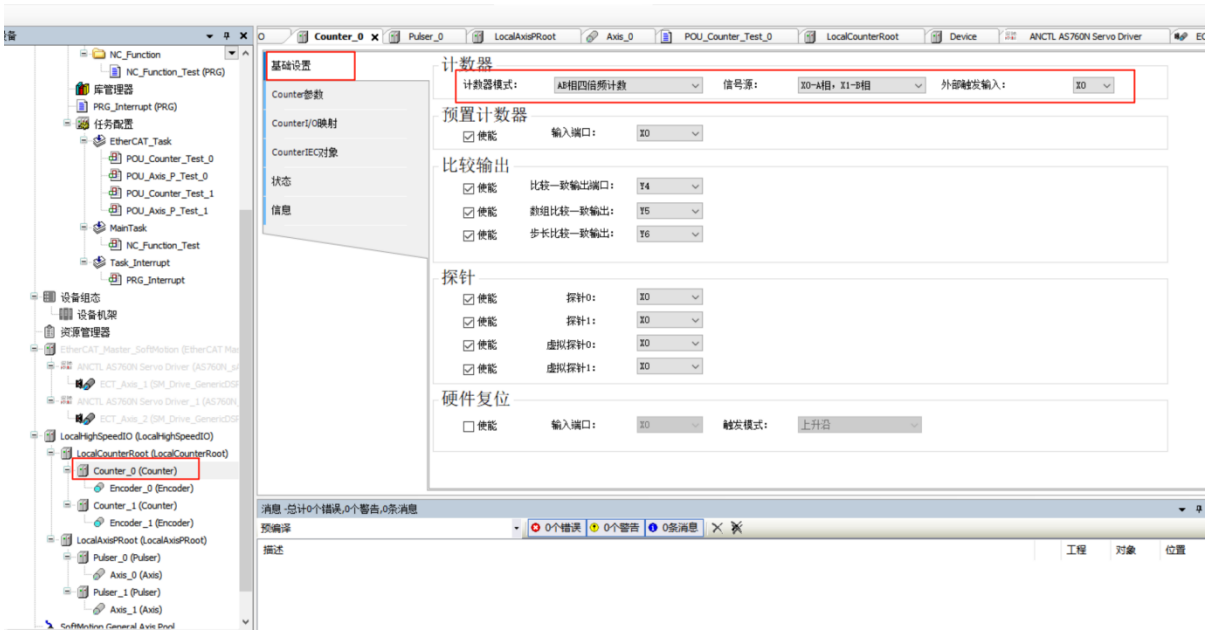
【图形化操作说明】

双击“LocalHighSpeedIO”，在“基础设置”中添加计数器，添加计数器后会自动按顺序分配硬件输入端口，并设置滤波时间

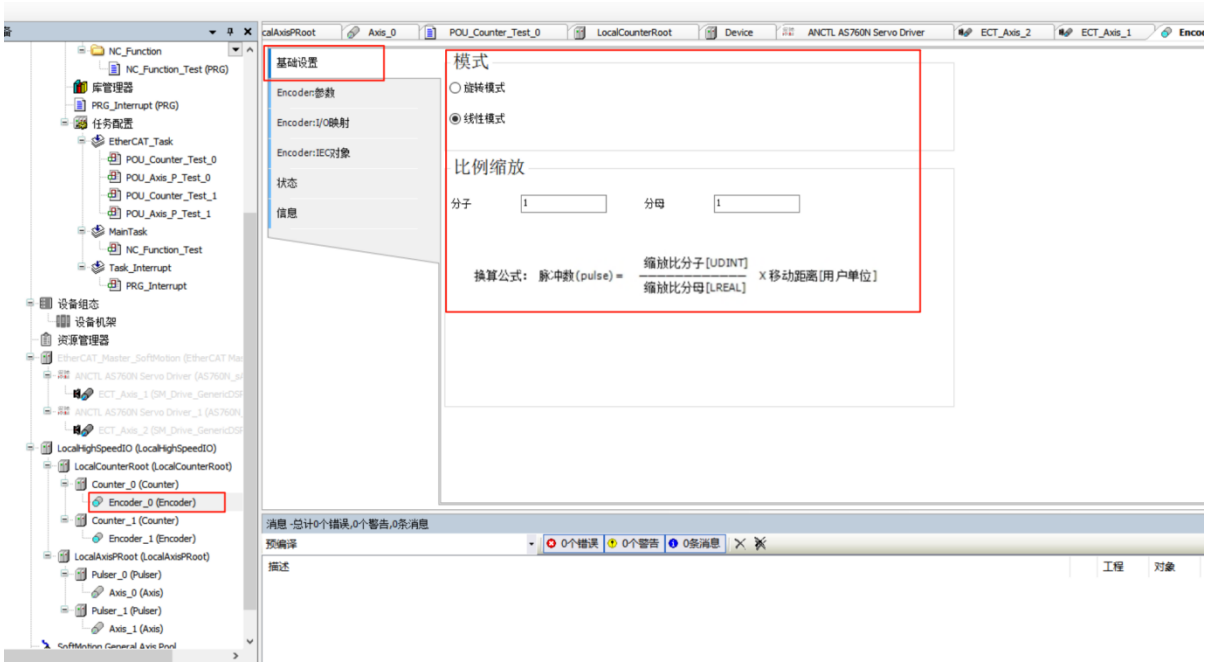


双击已添加的计数器“Counter_0”，在“基础设置页面”选择“计数器模式”、“信号源”、

“外部触发输入”

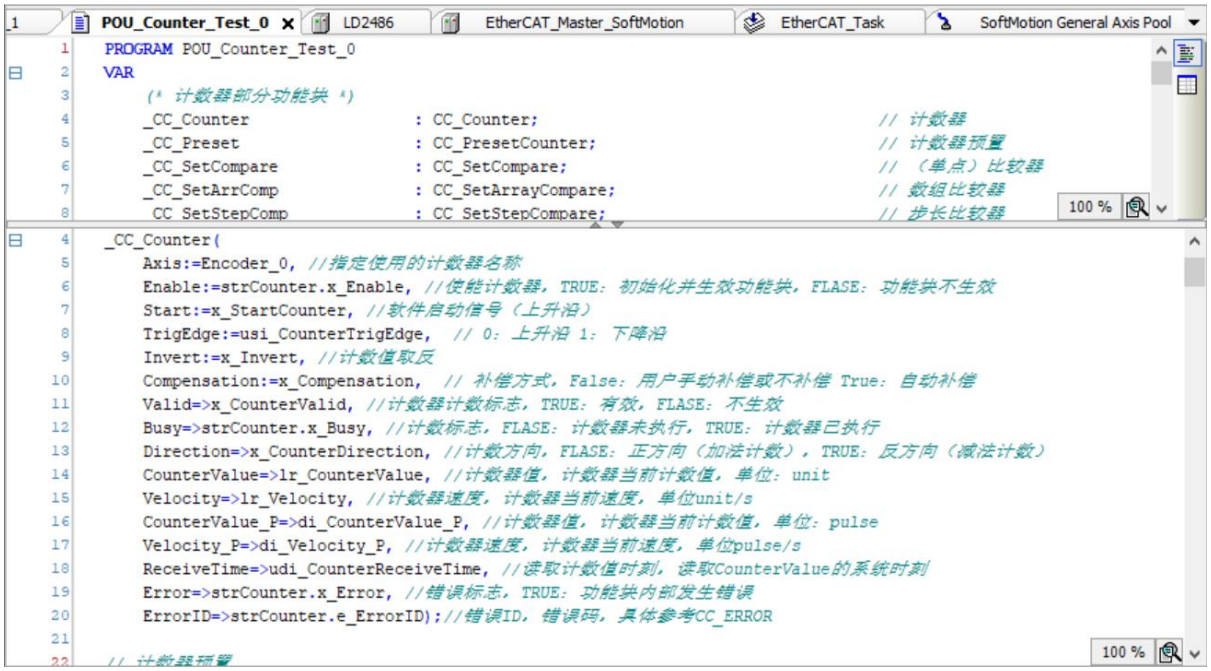


双击“Encoder”，在“基础设置”界面设置编码器轴的“模式”及“线性模式”



【功能块操作说明】

声明计数器“_CC_Counter: CC_Counter;”实例化



```
PROGRAM POU_Counter_Test_0
VAR
    (* 计数器部分功能块 *)
    _CC_Counter             : CC_Counter;                   // 计数器
    _CC_Preset              : CC_PresetCounter;             // 计数器预置
    _CC_SetCompare          : CC_SetCompare;                // (单点)比较器
    _CC_SetArrComp          : CC_SetArrayCompare;           // 数组比较器
    _CC_SetStepComp         : CC_SetStepCompare;            // 步长比较器

_CC_Counter(
    Axis:=Encoder_0, //指定使用的计数器名称
    Enable:=strCounter.x_Enable, //使能计数器，TRUE：初始化并生效功能块，FLASE：功能块不生效
    Start:=x_StartCounter, //软件启动信号(上升沿)
    TrigEdge:=usi_CounterTrigEdge,  // 0：上升沿 1：下降沿
    Invert:=x_Invert, //计数值取反
    Compensation:=x_Compensation,  // 补偿方式，False：用户手动补偿或不补偿 True：自动补偿
    Valid=>x_CounterValid, //计数器计数标志，TRUE：有效，FLASE：不生效
    Busy=>strCounter.x_Busy, //计数标志，FLASE：计数器未执行，TRUE：计数器已执行
    Direction=>x_CounterDirection, //计数方向，FLASE：正方向(加法计数)，TRUE：反方向(减法计数)
    CounterValue=>lr_CounterValue, //计数器值，计数器当前计数值，单位：unit
    Velocity=>lr_Velocity, //计数器速度，计数器当前速度，单位unit/s
    CounterValue_P=>di_CounterValue_P, //计数器值，计数器当前计数值，单位：pulse
    Velocity_P=>di_Velocity_P, //计数器速度，计数器当前速度，单位pulse/s
    ReceiveTime=>udi_CounterReceiveTime, //读取计数值时刻，读取CounterValue的系统时刻
    Error=>strCounter.x_Error, //错误标志，TRUE：功能块内部发生错误
    ErrorID=>strCounter.e_ErrorID);//错误ID，错误码，具体参考CC_ERROR
```

```
// 计数器
_CC_Counter(
    Axis:=Encoder_0, //指定使用的计数器名称
    Enable TRUE :=strCounter.x_Enable TRUE , //使能计数器，TRUE：初始化并生效功能块，FLASE：功能块不生效
    Start TRUE :=x_StartCounter TRUE , //软件启动信号(上升沿)
    TrigEdge 0 :=usi_CounterTrigEdge 0 ,  // 0：上升沿 1：下降沿
    Invert FALSE :=x_Invert FALSE , //计数值取反
    Compensation FALSE :=x_Compensation FALSE ,  // 补偿方式，False：用户手动补偿或不补偿 True：自动补偿
    Valid TRUE =>x_CounterValid TRUE , //计数器计数标志，TRUE：有效，FLASE：不生效
    Busy TRUE =>strCounter.x_Busy TRUE , //计数标志，FLASE：计数器未执行，TRUE：计数器已执行
    Direction FALSE =>x_CounterDirection FALSE , //计数方向，FLASE：正方向(加法计数)，TRUE：反方向(减法计数)
    CounterValue 6.36E+04 =>lr_CounterValue 6.36E+04 , //计数器值，计数器当前计数值，单位：unit
    Velocity 2.47E-323 =>lr_Velocity 2.47E-323 , //计数器速度，计数器当前速度，单位unit/s
    CounterValue_P 63556 =>di_CounterValue_P 63556 , //计数器值，计数器当前计数值，单位：pulse
    Velocity_P 0 =>di_Velocity_P 0 , //计数器速度，计数器当前速度，单位pulse/s
    ReceiveTime 2116590931 =>udi_CounterReceiveTime 2116590931 , //读取计数值时刻，读取CounterValue的系统时刻
    Error FALSE =>strCounter.x_Error FALSE , //错误标志，TRUE：功能块内部发生错误
    ErrorID CC_NO_ERRO =>strCounter.e_ErrorID CC_NO_ERRO );//错误ID，错误码，具体参考CC_ERROR
```

1，“Axis”引脚绑定添加的计数器名称，该功能块使用前必须使能计数器“Enable” 引脚，否则计数器不生效。

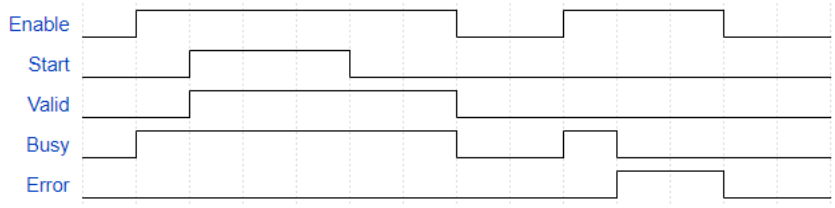
2， “Start”，0->1 跳变：启动计数器。

3，“Invert”引脚决定计数器计数方向，默认 FALSE，随接收脉冲量累加；置 TRUE，随接收脉冲量递减。

4，“Direction”引脚决定计数器计数方向，默认 FALSE，为正方向（加法计数）置 TRUE，为反方向（减法计数）

【时序图】

以 start 软件启动信号为 True 时示例：

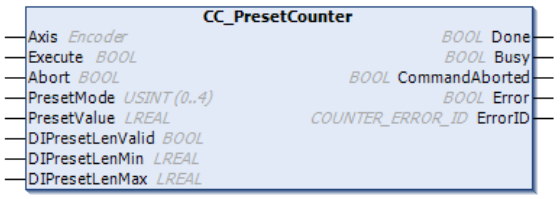


【错误说明】

详情查看 CC_ERROR-计数器错误 ID 说明表格。

1.2.2 CC_PresetCounter

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
CC_PresetCounter	计数器预置	FB		CC_PresetCounter(Axis:= , Execute:= , Abort:= , PresetMode:= , PresetValue:= , DIPresetLenValid:= , DIPresetLenMin:= , DIPresetLenMax:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Encoder	-	-	计数器编号 [0..7]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Execute	使能	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：使能计数器预置。
Abort	终止	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：终止计数器预置。
PresetMode	预制模式	USINT	[0,4]	0	预置模式， 0：软 Execute 上升沿触发， 1：比较一致输出时触发， 2：DI 上升沿触发， 3：DI 下降沿触发 ， 4：DI 上升沿/下降沿触发
PresetValue	预置值	LREAL	-	0	预置值，单位：unit
DIPresetLenValid	DI 有效方式	BOOL	[FALSE, TRUE]	FALSE	PresetMode=2，3，4 时， TRUE：启动预置信号长度滤波。
DIPresetLenMin	最小长度	LREAL	-	0	预置信号最小长度，单位：unit
DIPresetLenMax	最大长度	LREAL	-	0	预置信号最大长度，单位：unit

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Done	完成标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块执行完成
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
CommandAborted	终止标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块被终止
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

本功能块主要实现计数器的计数预置。

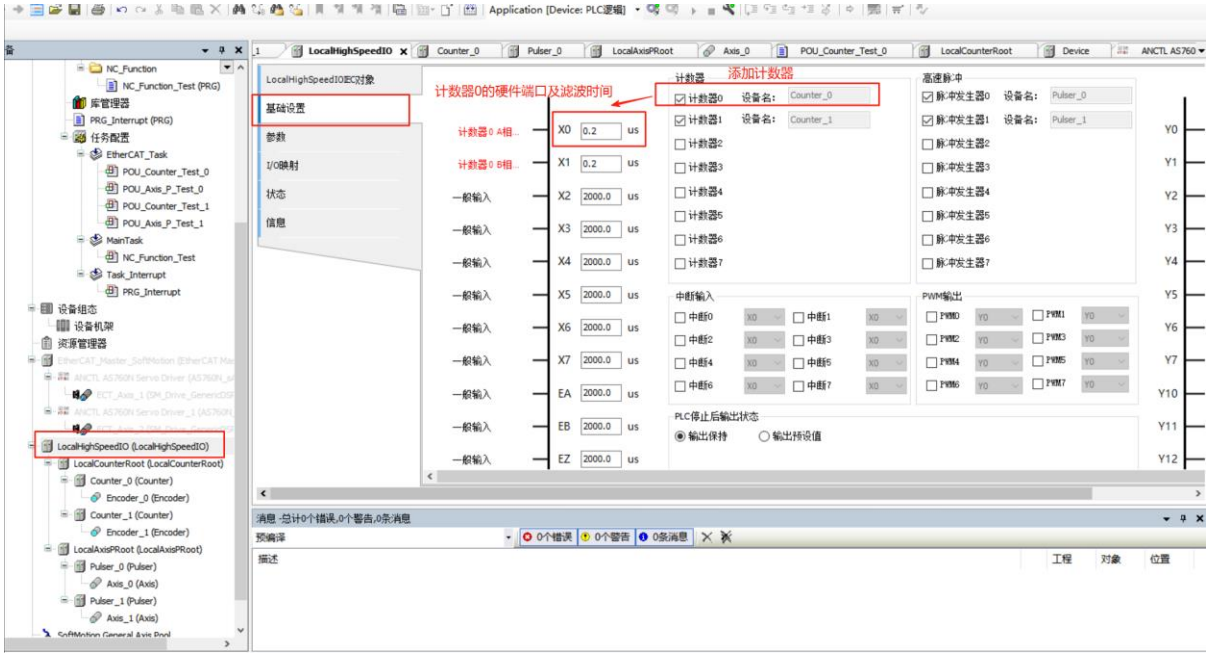
【注意事项】

PresetMode=0，即软 Execute 上升沿触发时，预置很快结束，此时 Abort 不容易触发打断，其他几种模式，预置计数器功能块在 Busy 状态时，Abort 都可以生效。

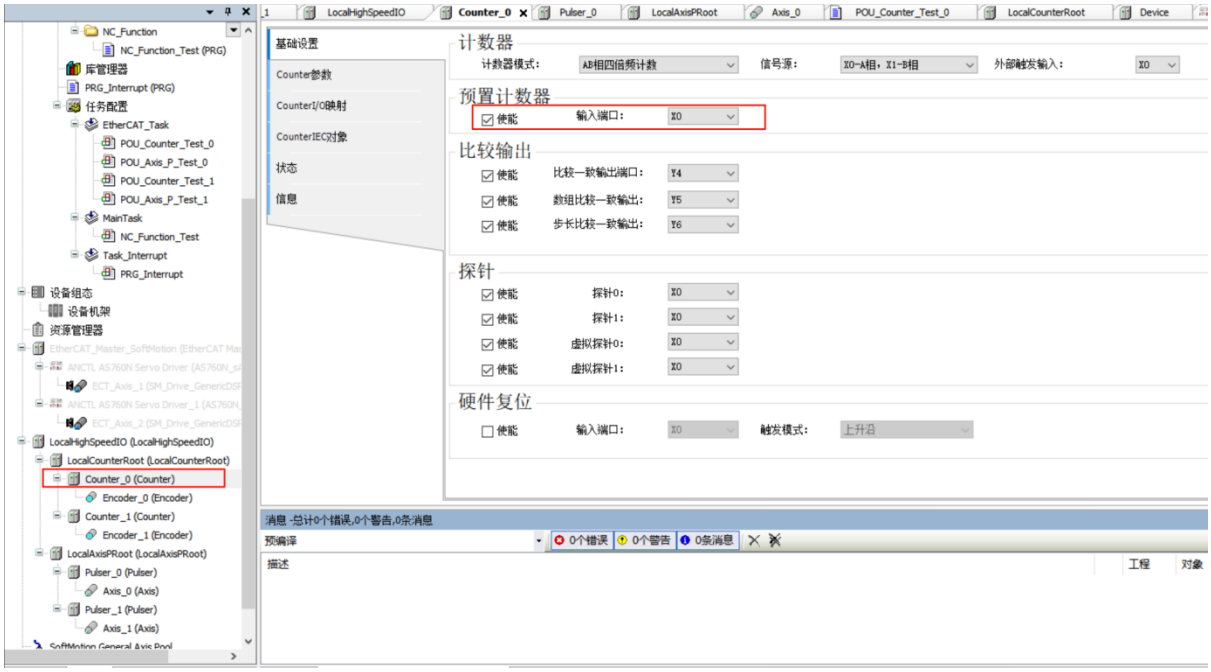
DIPresetLenValid 生效时，也即启动预置信号长度滤波的情况，需要注意这个长度与 PresetMode 有关，例如 PresetMode=3 时，设置的长度滤波范围检验的是上升沿之后的高电平的长度。

【图形化操作说明】

双击“LocalHighSpeedIO”，在“基础设置”中添加计数器，添加计数器后会自动按顺序分配硬件输入端口，并设置滤波时间

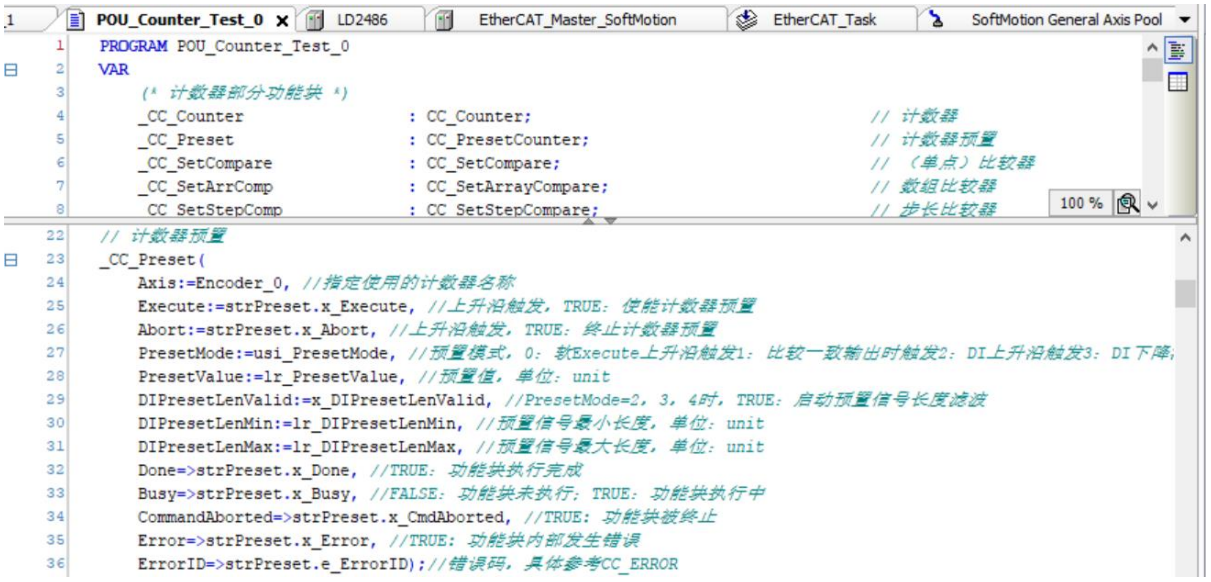


双击已添加的计数器“Counter_0”，使能“预置计数器”，选择相应的“输入端口”



【功能块操作说明】

声明计数器预置“_CC_Preset: CC_PresetCounter;”实例化



```
PROGRAM POU_Counter_Test_0
VAR
    (* 计数器部分功能块 *)
    _CC_Counter          : CC_Counter;          // 计数器
    _CC_Preset           : CC_PresetCounter;    // 计数器预置
    _CC_SetCompare       : CC_SetCompare;       // （单点）比较器
    _CC_SetArrComp       : CC_SetArrayCompare;  // 数组比较器
    _CC_SetStepComp      : CC_SetStepCompare;   // 步长比较器
```

```
// 计数器预置
_CC_Preset(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute:=strPreset.x_Execute, //上升沿触发，TRUE：使能计数器预置
    Abort:=strPreset.x_Abort, //上升沿触发，TRUE：终止计数器预置
    PresetMode:=usi_PresetMode, //预置模式，0：软Execute上升沿触发1：比较一致输出时触发2：DI上升沿触发3：DI下降
    PresetValue:=lr_PresetValue, //预置值，单位：unit
    DIPresetLenValid:=x_DIPresetLenValid, //PresetMode=2，3，4时，TRUE：启动预置信号长度滤波
    DIPresetLenMin:=lr_DIPresetLenMin, //预置信号最小长度，单位：unit
    DIPresetLenMax:=lr_DIPresetLenMax, //预置信号最大长度，单位：unit
    Done=>strPreset.x_Done, //TRUE：功能块执行完成
    Busy=>strPreset.x_Busy, //FALSE：功能块未执行；TRUE：功能块执行中
    CommandAborted=>strPreset.x_CmdAborted, //TRUE：功能块被终止
    Error=>strPreset.x_Error, //TRUE：功能块内部发生错误
    ErrorID=>strPreset.e_ErrorID);//错误码，具体参考CC_ERROR
```

```
// 计数器预置
_CC_Preset(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute:=strPreset.x_Execute, //上升沿触发，TRUE：使能计数器预置
    Abort:=strPreset.x_Abort, //上升沿触发，TRUE：终止计数器预置
    PresetMode:=usi_PresetMode, //预置模式，0：软Execute上升沿触发1：比较一致输出时触发2：DI上升沿触发3：DI下降沿触发4：DI上升沿/下降沿触发
    PresetValue:=lr_PresetValue, //预置值，单位：unit
    DIPresetLenValid:=x_DIPresetLenValid, //PresetMode=2，3，4时，TRUE：启动预置信号长度滤波
    DIPresetLenMin:=lr_DIPresetLenMin, //预置信号最小长度，单位：unit
    DIPresetLenMax:=lr_DIPresetLenMax, //预置信号最大长度，单位：unit
    Done=>strPreset.x_Done, //TRUE：功能块执行完成
    Busy=>strPreset.x_Busy, //FALSE：功能块未执行；TRUE：功能块执行中
    CommandAborted=>strPreset.x_CmdAborted, //TRUE：功能块被终止
    Error=>strPreset.x_Error, //TRUE：功能块内部发生错误
    ErrorID=>strPreset.e_ErrorID);//错误码，具体参考CC_ERROR
```

1，“Axis”引脚绑定添加的计数器名称;计数器预置功能块能够实现设定计数器预置 PresetCounter 的“PresetValue”引脚的值赋值到计数器 Counter 的“CounterValue” 引脚的计数值。

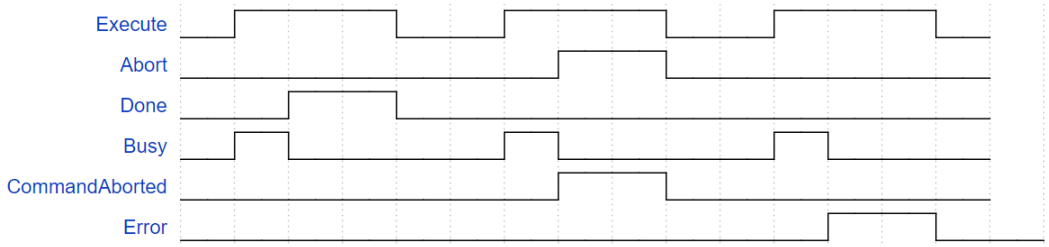
2，“Execute”引脚，默认 FALSE，置 TRUE 上升沿，“PresetValue”引脚当前值 赋值到计数器 Counter“CounterValue”引脚的值；该功能需在“PresetMode”引脚值 为 0 时实现。

3，“PresetMode”引脚决定计数器预置的模式，0：“Execute”引脚触发，1：比较器 PresetCounter 比较完成后触发，2：设定的端口信号置 TRUE,上升沿触发一次，3：设定的端口信号置 TRUE,下降沿触发一次，4：设定的端口信号置 TRUE 上升沿和下降沿各触发一次。

4，“DIPressetLenValid”引脚默认 FALSE；置 TRUE，功能块会过滤掉“DIPresetLenMin”引脚和“DIPresetLenMax”引脚分别设定的预置最大值和最小值外的长度不符高低电平。

5，若“Done”引脚输出表示功能块预置完成，“Execute”引脚需要重新触发上升沿，预置器重启。

【时序图】



【错误说明】

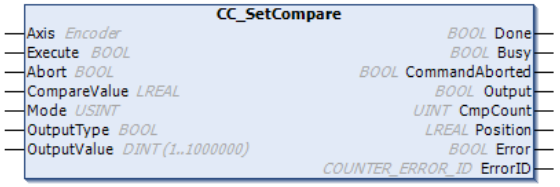
详情查看 CC_ERROR-计数器错误 ID 说明表格。

【注意事项】

计数器预置去掉窗口滤波

1.2.3 CC_SetCompare

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
CC_SetCompare	比较器	FB		CC_SetCompare(Axis:= , Execute:= , Abort:= , CompareValue:= , Mode:= , OutputType:= , OutputValue:= , Done=> , Busy=> , CommandAborted=> , Output=> , CmpCount=> , Position=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Encoder	-	-	计数器编号 [0..7]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Execute	使能	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：使能功能块。
Abort	终止	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：终止功能块。
CompareValue	比较值	LREAL	-	0	比较值，单位：unit
Mode	比较模式	USINT	[0,1]	0	0：单次模式 1：连续模式
OutputType	输出方式	BOOL	[FALSE, TRUE]	FALSE	FALSE：时间方式 TRUE：脉冲方式
OutputValue	输出值	DINT	[1, 1000000]	1000	如为时间方式，单位为 100us。 如为脉冲方式，单位为 pulse。

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Done	完成标志	BOOL	[FALSE, TRUE]	-	如为单次模式，执行一次输出后，Done，功能自动终止。连续模式，此信号无用。

Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
CommandAborted	终止标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块终止。
Output	输出	BOOL	[FALSE, TRUE]	-	比较一致输出，与 DO 一致
CmpCount	次数	UINT	[0,65535]	-	比较输出次数。
Position	比较值	LREAL	-	-	当前需要被比较的值。
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

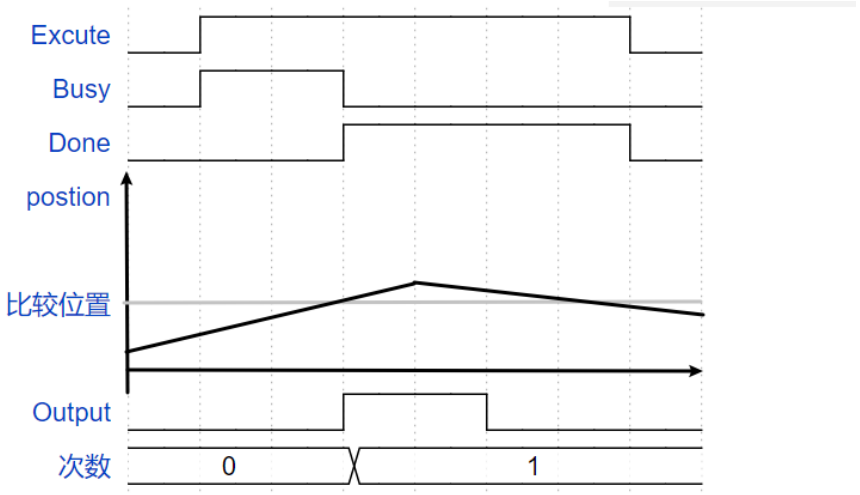
本功能块实现计数器当前计数值与设置的比较值相等时，触发一个硬件输出 DO 点，并保持一段时间。

【注意事项】

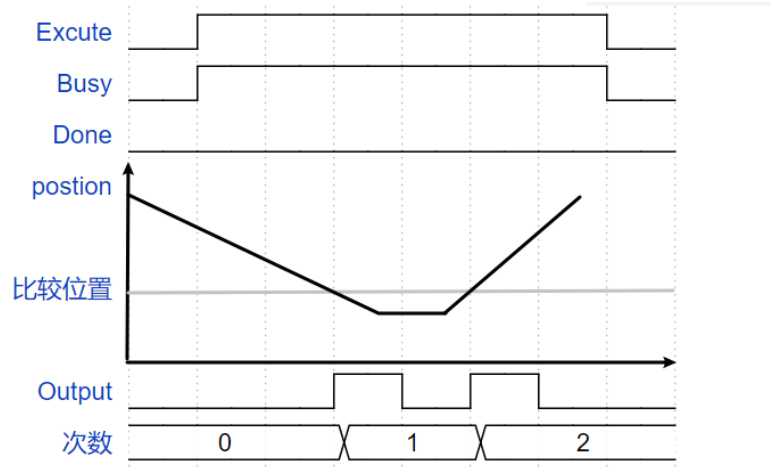
1、比较器支持单次和连续两种模式。如果选择单次模式，当执行完一次输出后，Done=TRUE，比较一致功能完成，如果需要继续使用，需要再次触发 Execute；如果选择连续模式，Done 信号不会有输出（此信号无用），只要计数器值与设置比较值一致，就会触发输出。

2、比较输出保持宽度支持两种方式，时间方式或脉冲方式。时间方式/脉冲方式输出值的具体计算：当计数器值等于设置的比较值时，比较输出 DO 点打开，保持 OutputValue*100us 时间/OutputValue 计数脉冲后，关闭输出 DO 点。比较功能块 Excute 触发前，必须配置好 DO 输出端口编号，后期修改无效，必须重新触发 Excute。

当比较模式为单次模式（Mode=0）时，时序如下：

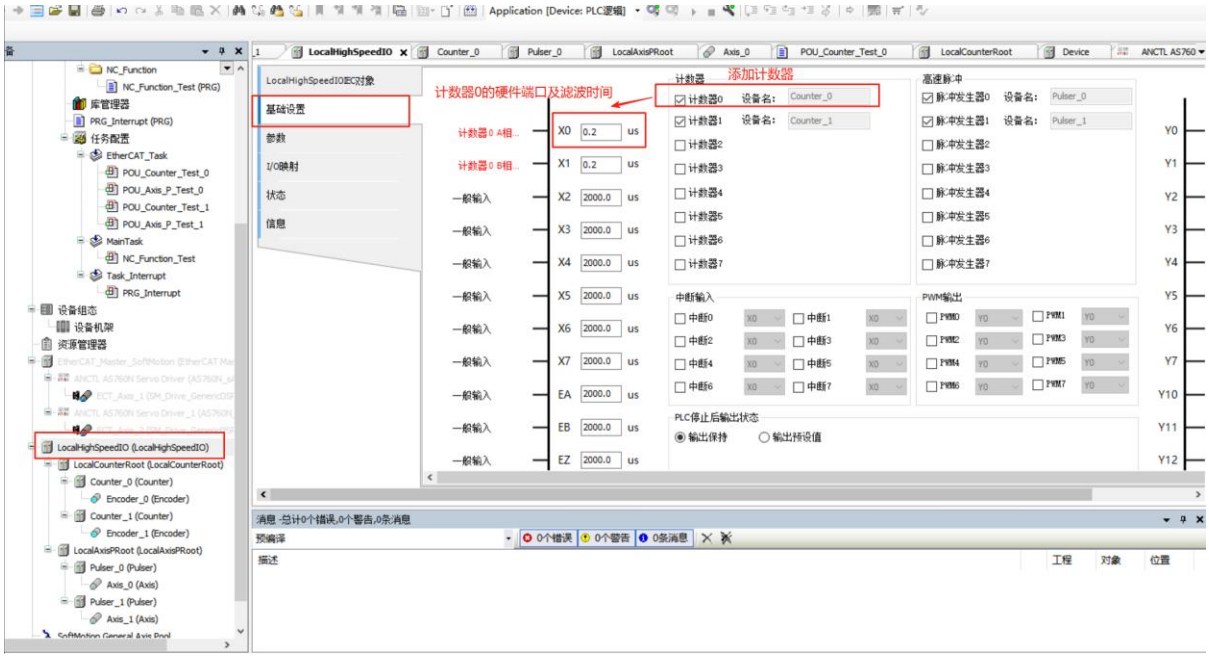


当比较模式为连续模式（Mode=1）时，时序如下：

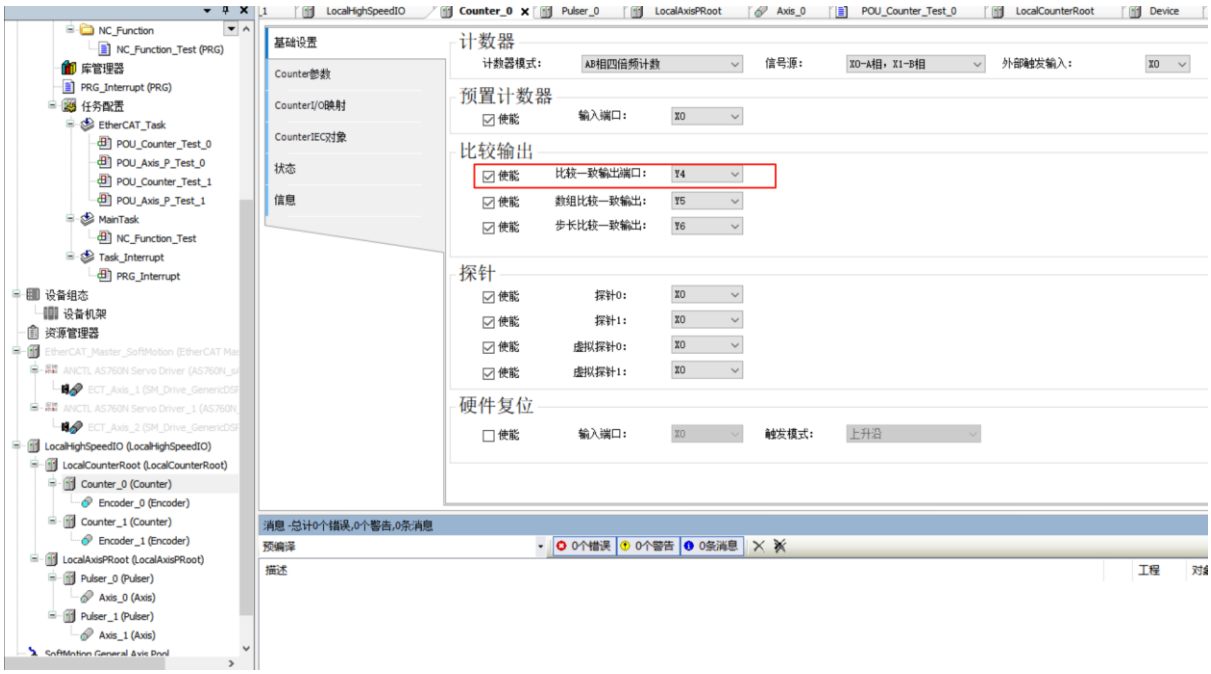


【图形化操作说明】

双击“LocalHighSpeedIO”，在“基础设置”中添加计数器，添加计数器后会自动按顺序分配硬件输入端口，并设置滤波时间

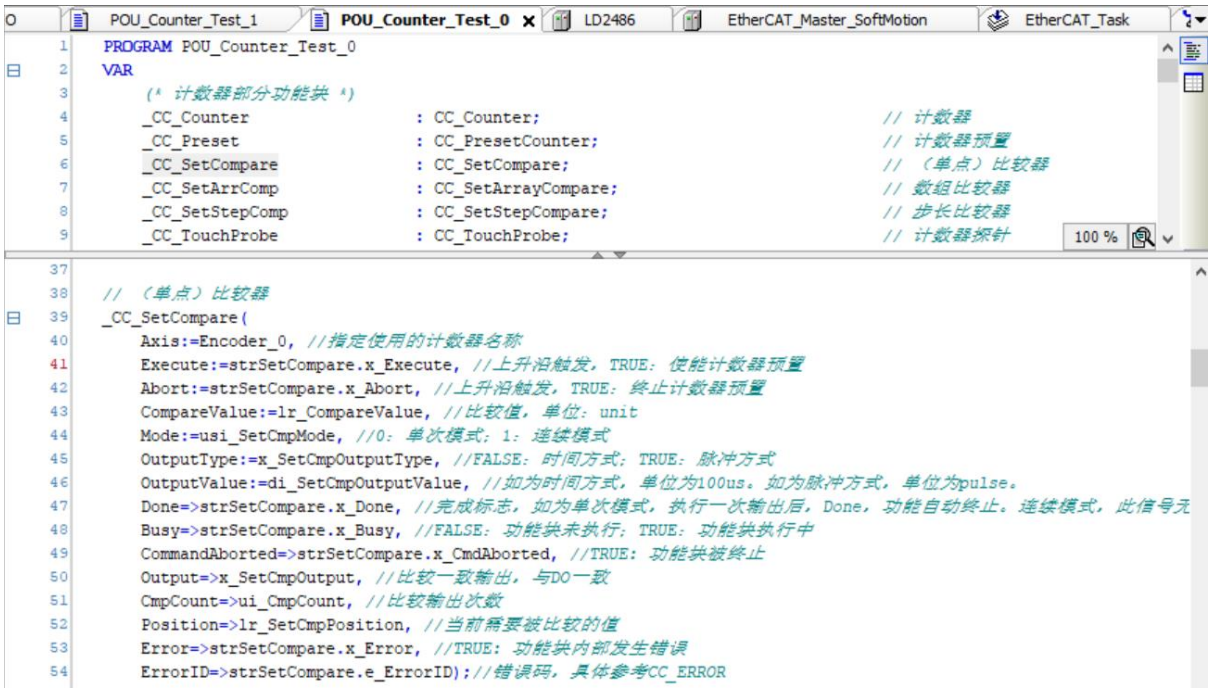


双击已添加的计数器“Counter_0”，使能“比较一致输出”，选择相应的“比较一致输出端口”



【功能块操作说明】

声明比较器“_CC_SetCompare : CC_SetCompare;”实例化



```
// （单点）比较器
_CC_SetCompare(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute TRUE :=strSetCompare.x_Execute TRUE , //上升沿触发，TRUE：使能计数器预置
    Abort FALSE :=strSetCompare.x_Abort FALSE , //上升沿触发，TRUE：终止计数器预置
    CompareValue 85 :=lr_CompareValue 85 , //比较值，单位：unit
    Mode 0 :=usi_SetCmpMode 0 , //0：单次模式；1：连续模式
    OutputType FALSE :=x_SetCmpOutputType FALSE , //FALSE：时间方式；TRUE：脉冲方式
    OutputValue 1000000 :=di_SetCmpOutputValue 1000000 , //如为时间方式，单位为100us。如为脉冲方式，单位为pulse。
    Done TRUE =>strSetCompare.x_Done TRUE , //完成标志，如为单次模式，执行一次输出后，Done，功能自动终止。连续模式，此信号无用
    Busy FALSE =>strSetCompare.x_Busy FALSE , //FALSE：功能块未执行；TRUE：功能块执行中
    CommandAborted FALSE =>strSetCompare.x_CmdAborted FALSE , //TRUE：功能块被终止
    Output FALSE =>x_SetCmpOutput FALSE , //比较一致输出，与DO一致
    CmpCount 1 =>ui_CmpCount 1 , //比较输出次数
    Position 115 =>lr_SetCmpPosition 115 , //当前需要被比较的值
    Error FALSE =>strSetCompare.x_Error FALSE , //TRUE：功能块内部发生错误
    ErrorID CC_NO_ERRO =>strSetCompare.e_ErrorID CC_NO_ERRO );//错误码，具体参考CC_ERROR
```

1，“Axis”引脚绑定添加的计数器名称;比较器功能块能够实现设定比较器 SetCompar 的“CompareValue”引脚的设定值与计数器 Counter 的“CounterValue”引脚的计数值比较，若相等则完成比较。

2，“Mode”引脚决定比较器比较模式；0：单次模式，比较器在引脚“Execute”上升沿触发后，执行一次比较，完成后功能块结束，设定的 DO 端口和“Output”引脚输出，“Done”引脚置 TRUE。1：循环模式，比较器在引脚“Execute”上升沿触发后一直执行比较，完成后设定的 DO 端口和“Output”引脚输出，“Done”不变，“Abort”引脚置 TRUE 后功能块比较停止。

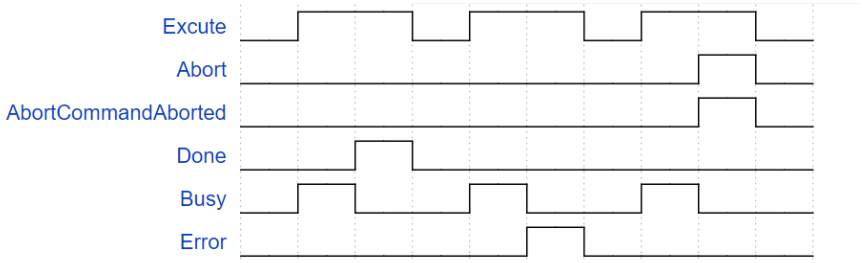
3，“CmpCount”引脚显示比较完成次数。

4，“Position”引脚显示当前需要比较的设定值。

5，“OutputType”引脚可以设定输出

【时序图】

注：以单次模式为例：



【错误说明】

详情查看 CC_ERROR-计数器错误 ID 说明表格。

1.2.4 CC_SetArrayCompare

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
CC_SetArrayCompare	数组比较器	FB		CC_SetArrayCompare(Axis:= , Execute:= , Abort:= , CompareArray:= , CompareArrayLength:= , Mode:= , OutputType:= , OutputValue:= , Done=> , Busy=> , CommandAborted=> , Output=> , Index=> , Position=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Encoder	-	-	计数器编号 [0..6]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Execute	使能	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：使能功能块。
Abort	终止	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：终止功能块。
CompareArray	比较值数组	LREAL[]	-	0	比较值数组，单位：unit
CompareArrayLength	比较数组长度	UINT	[1,1000]	1	比较值数组长度，最多 1000 个。
Mode	比较模式	USINT	[0,1]	0	0：单次模式 1：连续模式
OutputType	输出方式	BOOL	[FALSE, TRUE]	FALSE	FALSE：时间方式 TRUE：脉冲方式
OutputValue	输出值	DINT	[1, 1000000]	1000	打开输出口保持时间 / 脉冲，最小单位为 100us。 默认值：1000，

					1000*100us=100ms。

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Done	完成标志	BOOL	[FALSE, TRUE]	-	如为单次模式，执行一次输出后，Done，功能自动终止。连续模式，此信号无用。
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
CommandAborted	终止标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块终止。
Output	输出	BOOL	[FALSE, TRUE]	-	比较一致输出，与 DO 一致
Index	序号	UINT	[0,65535]	-	当前需要被比较的序号。
Position	比较值	LREAL	-		当前需要被比较的值。
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

本功能块实现计数器当前计数值与数组待比较序号对应设置的比较值相等时，触发一个硬件输出 DO 点，并保持一段时间。

【注意事项】

1、数组比较功能可以设置单次模式或连续模式。如果选择单次模式，当执行完一次输出后，Done=TRUE，比较一致功能完成，如果需要继续使用，需要再次触发 Execute；如果选择连续模式，Done 信号不会有输出（此信号无用），只要计数器值与设置比较值一致，就会触发输出。

2、比较输出保持宽度支持两种方式，时间方式或脉冲方式。时间方式/脉冲方式输出值的具体计算：当计数器值等于设置的比较值时，比较输出 DO 点打开，保持 OutputValue*100us 时间/OutputValue 计数脉冲后，关闭输出 DO 点。比较功能块 Excute 触发前，必须配置好 DO 输出端口编号，后期修改无效，必须重新触发 Excute。

3、用户程序中定义用于存储比较值数组的长度，必须大于 CompareArrayLength，否则，数组越界操作可能导致程序执行异常，严重时可能导致 PLC 死机。

4、数组比较过程中，会按照数组顺序进行比较，如果跳过当前待比较值，那么不会触发后面的比较一致输出。

示例：

定义比较值数组如下（CompareArrayLength=3）：

```
1  PROGRAM PLC_PRG
2  VAR
3      _arrComp : ARRAY [0..2] OF DINT := [1000,2000,3000];
4  END_VAR
```

当计数器值大于 1000（如 1001 时）并继续增加时，再触发数组比较，那么因为无法与数组_arrComp[0]对应的 1000 比较一致，那么就不会出现触发，后面的值也不会进行比较。

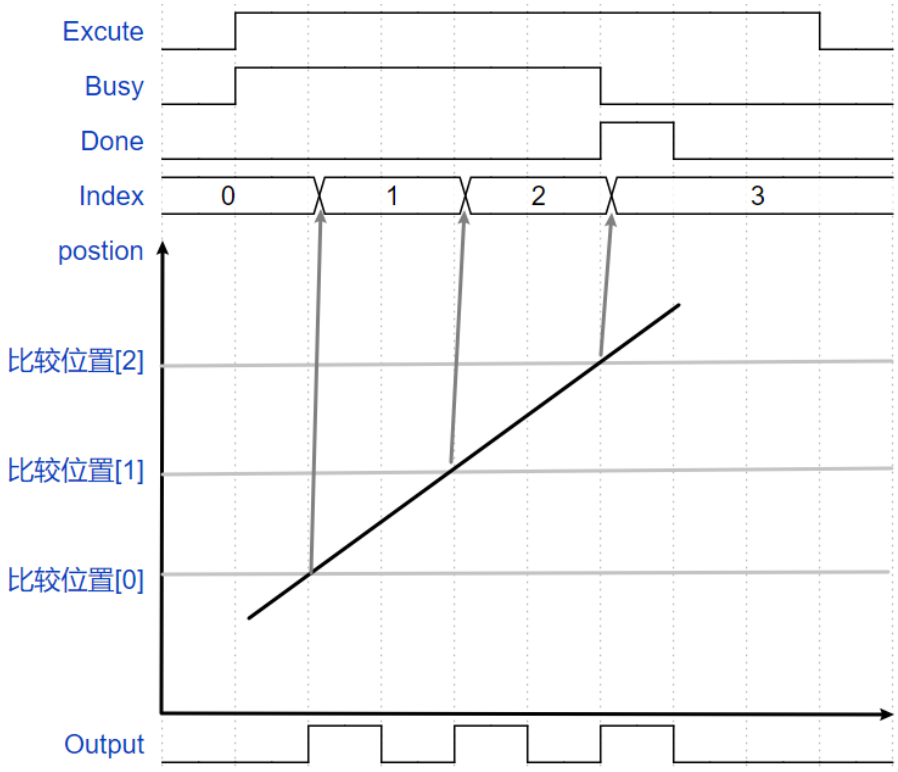
1、数组比较，相邻两点之间的距离至少大于最小距离（最小距离=计数器计数速度[频率]*扫描

周期），否则，单次模式下，功能块也不能执行剩余的比较点。

2、数组比较过程中，再次触发 Excute 上升沿，功能块会从第一个点重新执行比较输出功能。

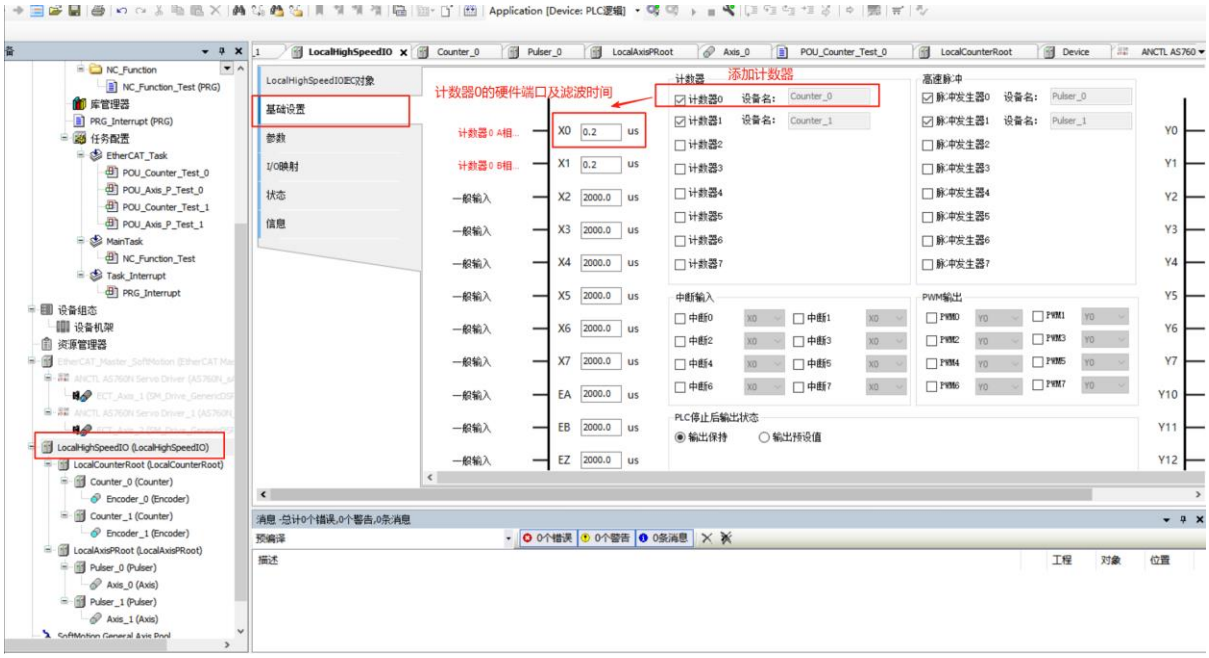
3、相邻两点的触发间隔的时间小于比较输出保持时间的情况下，输出口将连续保持输出状态。总保持输出的时间=本次输出保持时间+下次保持输出时间，多个比较值保持输出总时间为叠加的保持输出时间。

比较 3 个位置（CompareArrayLength=3），指令时序图如下图所示：

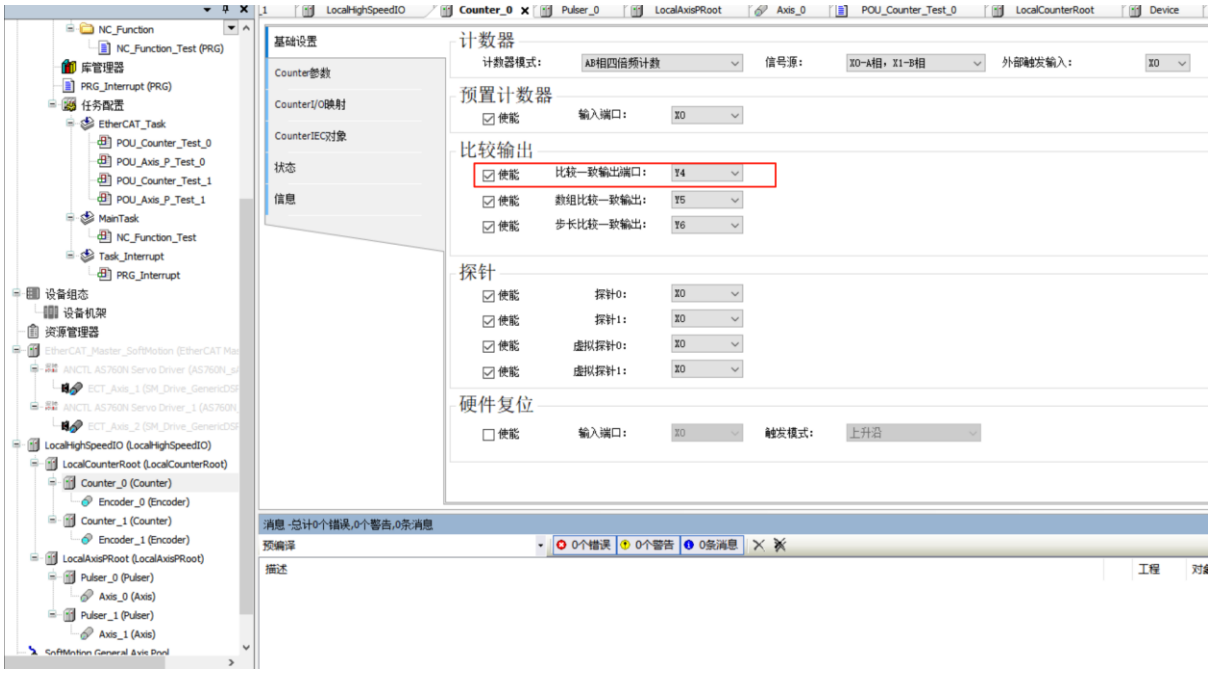


【图形化操作说明】

双击“LocalHighSpeedIO”，在“基础设置”中添加计数器，添加计数器后会自动按顺序分配硬件输入端口，并设置滤波时间

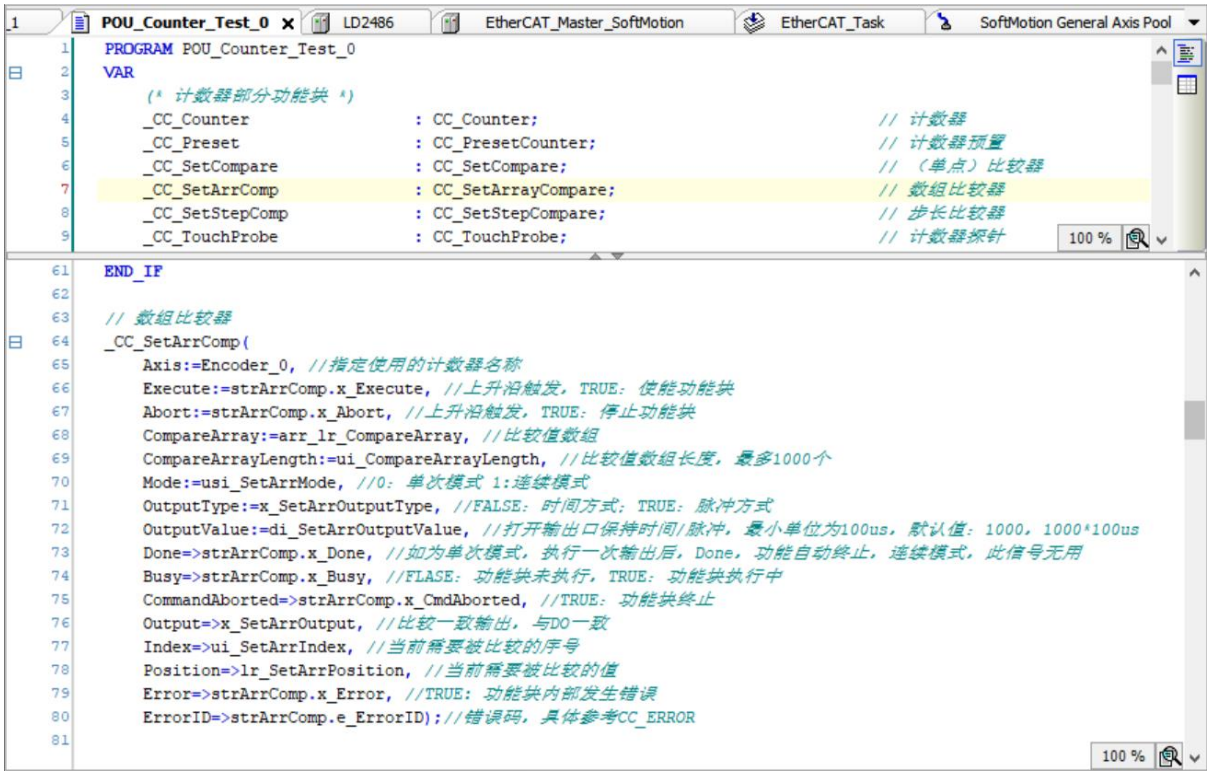


双击已添加的计数器“Counter_0”，使能“数组比较一致输出”，选择相应的“数组比较一致输出端口”



【功能块操作说明】

声明数组比较器“_CC_SetArrComp: CC_SetArrayCompare;”实例化



【功能块操作说明】

```
// 数组比较器
_CC_SetArrComp(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute TRUE :=strArrComp.x_Execute TRUE , //上升沿触发，TRUE：使能功能块
    Abort FALSE :=strArrComp.x_Abort FALSE , //上升沿触发，TRUE：停止功能块
    CompareArray:=arr_lr_CompareArray, //比较值数组
    CompareArrayLength 10 :=ui_CompareArrayLength 10 , //比较值数组长度，最多1000个
    Mode 0 :=usi_SetArrMode 0 , //0：单次模式 1:连续模式
    OutputType FALSE :=x_SetArrOutputType FALSE , //FALSE：时间方式；TRUE：脉冲方式
    OutputValue 1000000 :=di_SetArrOutputValue 1000000 , //打开输出口保持时间/脉冲，最小单位为100us，默认值：1000，1000*100us
    Done FALSE =>strArrComp.x_Done FALSE , //如为单次模式，执行一次输出后，Done，功能自动终止，连续模式，此信号无用
    Busy TRUE =>strArrComp.x_Busy TRUE , //FLASE：功能块未执行，TRUE：功能块执行中
    CommandAborted FALSE =>strArrComp.x_CmdAborted FALSE , //TRUE：功能块终止
    Output FALSE =>x_SetArrOutput FALSE , //比较一致输出，与DO一致
    Index 3 =>ui_SetArrIndex 3 , //当前需要被比较的序号
    Position 0 =>lr_SetArrPosition 0 , //当前需要被比较的值
    Error FALSE =>strArrComp.x_Error FALSE , //TRUE：功能块内部发生错误
    ErrorID CC_NO_ERRO =>strArrComp.e_ErrorID CC_NO_ERRO );//错误码，具体参考CC_ERROR
```

1，“Axis”引脚绑定添加的计数器名称;数组比较器功能块能够实现设定数组比较器的“ CompareArray ”引脚的数组设定值与计数器 Counter 的“CounterValue”引脚的计数值比较，若所有设定的数值依次相等则完成比较。

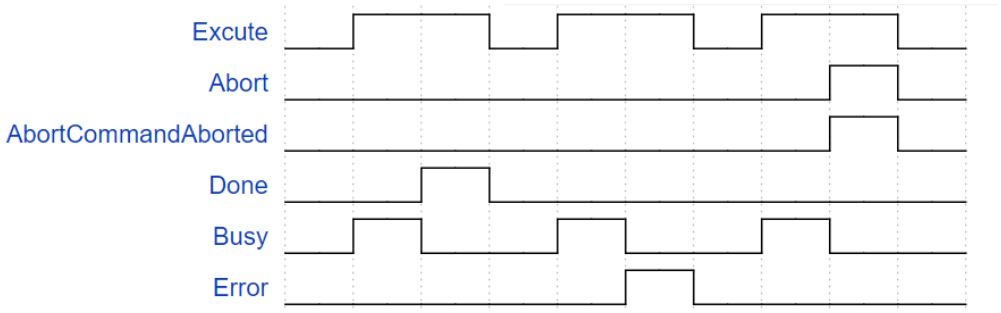
2，“Mode”引脚决定数组比较器比较模式；0：单次模式，数组比较器在引脚“Execute”上升沿触发后，执行一次数组依次比较，每完成一个数值比较，设定的 DO 端口和“Output”引脚输出，完成所有设定的数组后功能块结束，“Done”引脚置 TRUE。1：循环模式，比较器在引脚“Execute”上升沿触发后依次循环执行比较，每比较完成一个设定的 DO 端口和“Output”引脚输出，“Done”不变，“Abort”引脚置 TRUE 后功能块比较停止。

3，“CompareArrayLength”引脚设定数组比较个数。

4，“CmpCount”引脚显示比较完成次数。

5，数组比较是按顺序依次比较，若第一个设定数值比较完成，则进行第二个设定数值比较；反之，若第一个数值比较未完成，则后面所有数值都不会进行比较。

【时序图】



【错误说明】

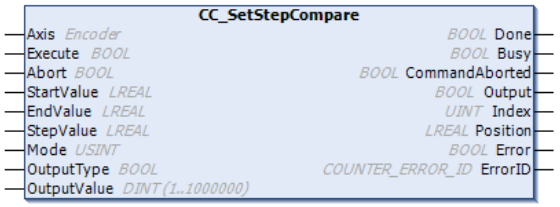
详情查看 CC_ERROR-计数器错误 ID 说明表格。

【注意事项】

S 系列 S500/S300 不支持数组比较器

1.2.5 CC_SetStepCompare

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
CC_SetStepCompare	步长比较器	FB		CC_SetStepCompare(Axis:= , Execute:= , Abort:= , StartValue:= , EndValue:= , StepValue:= , Mode:= , OutputType:= , OutputValue:= , Done=> , Busy=> , CommandAborted=> , Output=> , Index=> , Position=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Encoder	-	-	计数器编号 [0..6]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Execute	使能	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：使能功能块。
Abort	终止	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：终止功能块。
StartValue	比较起始值	LREAL	-	0	比较起始值，单位：unit
EndValue	比较结束值	LREAL	-	0	比较结束值，单位：unit
StepValue	等距步长	LREAL	>0	0	等距步长，单位：unit
Mode	比较模式	USINT	[0,1]	0	0：单次模式 1：连续模式
OutputType	输出方式	BOOL	[FALSE, TRUE]	FALSE	FALSE：时间方式 TRUE：脉冲方式
OutputValue	输出值	DINT	[1, 1000000]	1000	如为时间方式，单位：100us。 如为脉冲方式，单位：pulse。

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Done	完成标志	BOOL	[FALSE, TRUE]	-	如为单次模式，执行一次输出后，Done，功能自动终止。连续模式，此信号无用。
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
CommandAborted	终止标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块终止。
Output	输出	BOOL	[FALSE, TRUE]	-	比较一致输出，与 DO 一致
Index	序号	UINT	[0,65535]	-	当前需要被比较的序号。
Position	比较值	LREAL	-		当前需要被比较的值。
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

本功能块实现计数器当前计数值与按照指定步长变化的比较值相等时，触发一个硬件输出 DO 点，并保持一段时间。

【注意事项】

1、步长比较功能可以设置单次模式或连续模式。如果选择单次模式，当执行完一次输出后，Done=TRUE，比较一致功能完成，如果需要继续使用，需要再次触发 Execute；如果选择连续模式，Done 信号不会有输出（此信号无用），只要计数器值与设置比较值一致，就会触发输出，执行完一遍后会从极小值继续计算下一次步长比较值。

2、比较输出保持宽度支持两种方式，时间方式或脉冲方式。时间方式/脉冲方式输出值的具体计算：当计数器值等于设置的比较值时，比较输出 DO 点打开，保持 OutputValue*100us 时间/OutputValue 计数脉冲后，关闭输出 DO 点。比较功能块 Excute 触发前，必须配置好 DO 输出端口编号，后期修改无效，必须重新触发 Excute。

3、步长比较，相邻两点之间的距离至少大于最小距离（最小距离=计数器计数速度[频率]*扫描周期），否则，单次模式下，功能块也不能执行剩余的比较点。

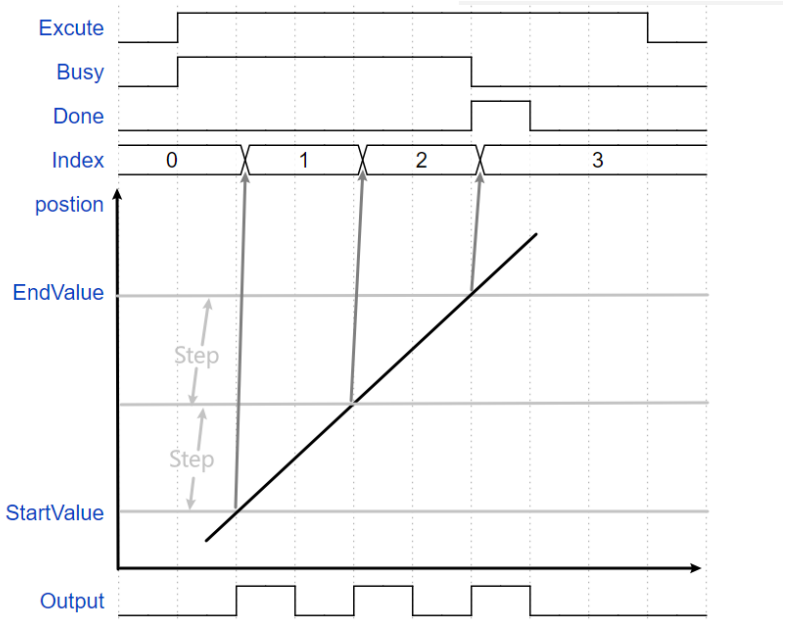
4、步长比较过程中，再次触发 Excute 上升沿，功能块会从第一个点重新执行比较输出功能。

5、步长比较过程中，会按照按步长累加的比较值进行比较，如果跳过当前待比较值，那么不会触发后面的比较一致输出。

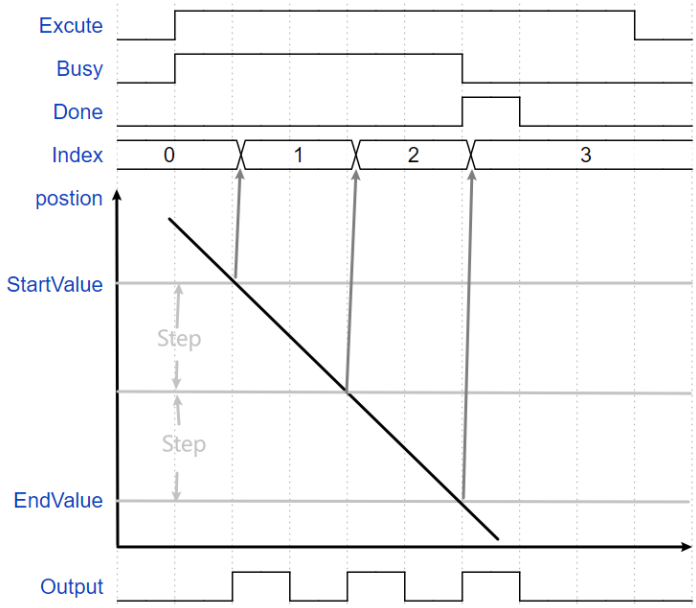
6、相邻两点的触发间隔的时间小于比较输出保持时间的情况下，输出口将连续保持输出状态。总保持输出的时间=本次输出保持时间+下次保持输出时间，多个比较值保持输出总时间为叠加的保持输出时间。

7、注意 StartValue 和 EndValue 的大小。

StartValue<EndValue，指令时序图如下：

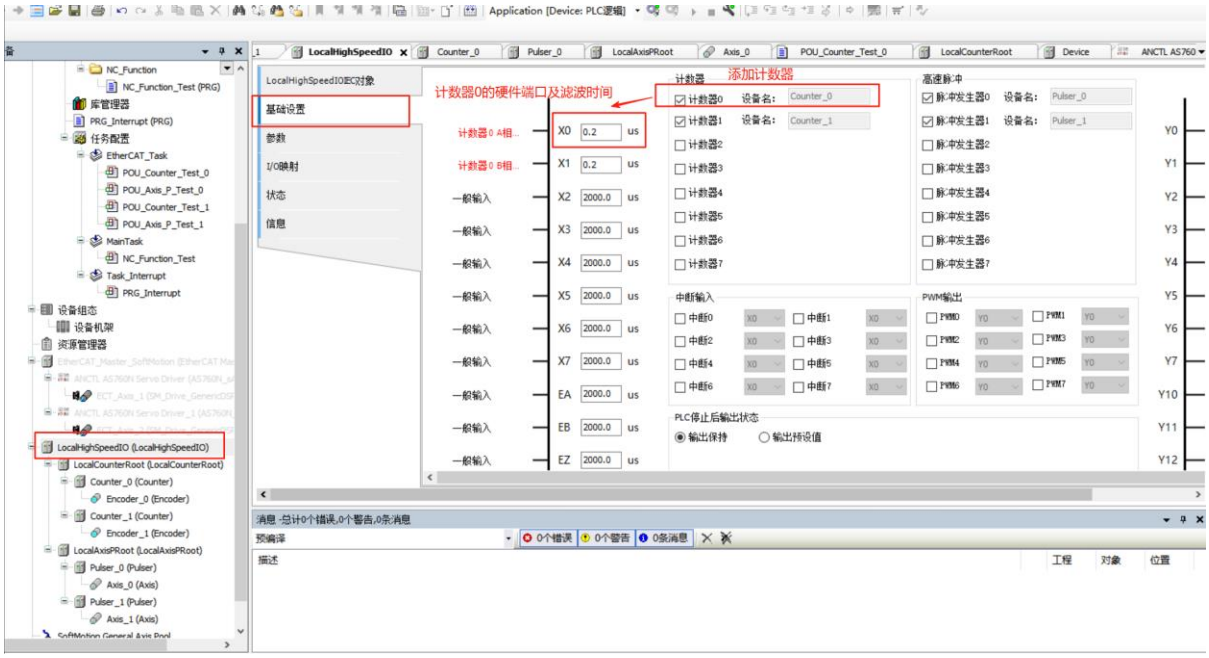


StartValue>EndValue，指令时序图如下：

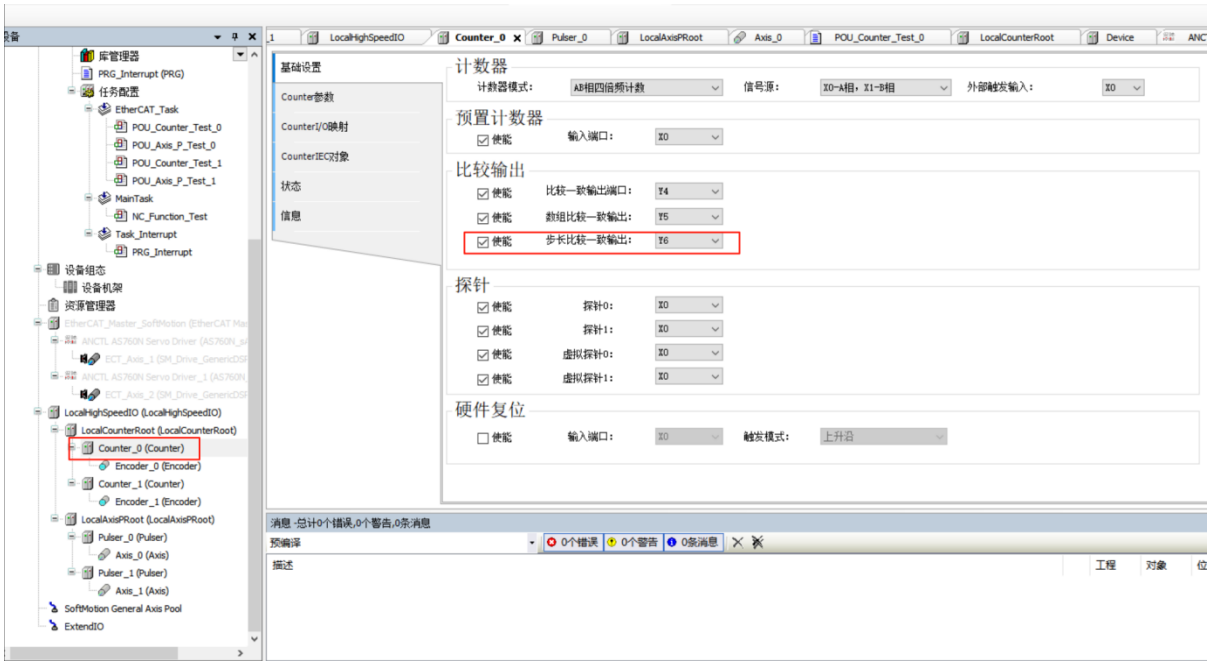


【图形化操作说明】

双击“LocalHighSpeedIO”，在“基础设置”中添加计数器，添加计数器后会自动按顺序分配硬件输入端口，并设置滤波时间



双击已添加的计数器“Counter_0”，使能“步长比较一致输出”，选择相应的“步长比较一致输出端口”



【功能块操作说明】

声明步长比较器“_CC_SetStepComp: CC_SetStepCompare;”

```
PROGRAM POU_Counter_Test_0
VAR
    (* 计数器部分功能块 *)
    _CC_Counter             : CC_Counter;                   // 计数器
    _CC_Preset              : CC_PresetCounter;             // 计数器预置
    _CC_SetCompare          : CC_SetCompare;                // （单点）比较器
    _CC_SetArrComp          : CC_SetArrayCompare;           // 数组比较器
    _CC_SetStepComp         : CC_SetStepCompare;            // 步长比较器
    _CC_TouchProbe          : CC_TouchProbe;                // 计数器探针
```

```
// 步长比较器
_CC_SetStepComp(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute:=strStepComp.x_Execute, //上升沿触发，TRUE：使能功能块
    Abort:=strStepComp.x_Abort, //上升沿触发，TRUE：停止功能块
    StartValue:=lr_StartValue, //比较起始值，单位：pluse
    EndValue:=lr_EndValue, //比较结束值，单位：pluse
    StepValue:=lr_StepValue, //等距步长，单位：pluse
    Mode:=usi_SetStepMode, //0：单次模式 1:连续模式
    OutputType:=x_SetStepOutType, //FALSE：时间方式；TRUE：脉冲方式
    OutputValue:=di_SetStepOutputValue, //打开输出口保持时间/脉冲，最小单位为100us，默认值：1000，1000*100us
    Done=>strStepComp.x_Done,  //如为单次模式，执行一次输出后，Done，功能自动终止，连续模式，此信号无用
    Busy=>strStepComp.x_Busy, //FLASE：功能块未执行，TRUE：功能块执行中
    CommandAborted=>strStepComp.x_CmdAborted, //TRUE：功能块终止
    Output=>x_SetStepOutput, //比较一致输出，与DO一致
    Index=>ui_SetStepIndex,  //当前需要被比较的序号
    Position=>lr_SetStepPosition, //当前需要被比较的值
    Error=>strStepComp.x_Error, //TRUE：功能块内部发生错误
    ErrorID=>strStepComp.e_ErrorID);//错误码，具体参考CC_ERROR
```

功能块操作说明

```
// 步长比较器
_CC_SetStepComp(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute TRUE :=strStepComp.x_Execute TRUE , //上升沿触发，TRUE：使能功能块
    Abort FALSE :=strStepComp.x_Abort FALSE , //上升沿触发，TRUE：停止功能块
    StartValue 30 :=lr_StartValue 30 , //比较起始值，单位：pluse
    EndValue 70 :=lr_EndValue 70 , //比较结束值，单位：pluse
    StepValue 10 :=lr_StepValue 10 , //等距步长，单位：pluse
    Mode 0 :=usi_SetStepMode 0 , //0：单次模式 1:连续模式
    OutputType FALSE :=x_SetStepOutType FALSE , //FALSE：时间方式；TRUE：脉冲方式
    OutputValue 1000000 :=di_SetStepOutputValue 1000000 , //打开输出口保持时间/脉冲，最小单位为100us，默认值：1000，1000*100us
    Done FALSE =>strStepComp.x_Done FALSE ,  //如为单次模式，执行一次输出后，Done，功能自动终止，连续模式，此信号无用
    Busy TRUE =>strStepComp.x_Busy TRUE , //FLASE：功能块未执行，TRUE：功能块执行中
    CommandAborted FALSE =>strStepComp.x_CmdAborted FALSE , //TRUE：功能块终止
    Output FALSE =>x_SetStepOutput FALSE , //比较一致输出，与DO一致
    Index 0 =>ui_SetStepIndex 0 ,  //当前需要被比较的序号
    Position 30 =>lr_SetStepPosition 30 , //当前需要被比较的值
    Error FALSE =>strStepComp.x_Error FALSE , //TRUE：功能块内部发生错误
    ErrorID CC_NO_ERRO =>strStepComp.e_ErrorID CC_NO_ERRO );//错误码，具体参考CC_ERROR
```

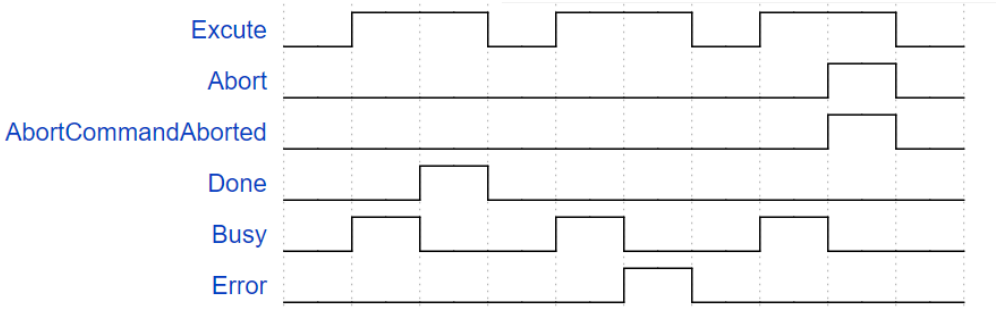
1，“Axis”引脚绑定添加的计数器名称;数组比较器功能块能够实现设定步长比较 SetStepCompare 的“StepValue”引脚设定值与计数器 Counter 接收的脉冲步长值比较，若设定计数范围步长依次比较完成，则功能块完成比较。

2，“StartValue”引脚和“EndValue”引脚分别设定步长比较器 SetStepCompare 比较起始计数值和结束计数值，当计数器 Counter 的引脚“CounterValue”数值满足设定的范围，则开始比较，若比较长度在范围不能整除，末尾有剩余则不作比较输出，循环比较会从起始值开始。

3，“StepValue”引脚设定步长比较器 SetStepCompare 的比较值,从起始位置开始每一段步长输出一次。

4，“Mode”引脚决定数组比较器比较模式；0：单次模式，数组比较器在引脚“Execute”上升沿触发后，执行一次数组依次比较，每完成一个数值比较，设定的 DO 端口和“Output”引脚输出，完成所有设定的数组后功能块结束，“Done”引脚置 TRUE。1：循环模式，比较器在引脚“Execute”上升沿触发后依次循环执行比较，每比较完成一个设定的 DO 端口和“Output”引脚输出，“Done”不变，“Abort”引脚置 TRUE 后功能块比较停止。

【时序图】



【错误说明】

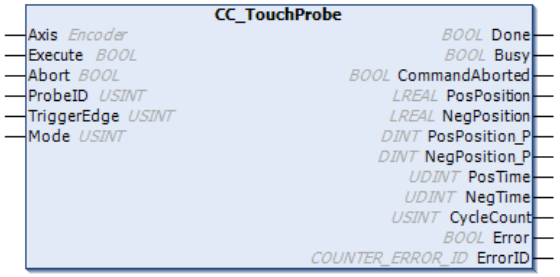
详情查看 CC_ERROR-计数器错误 ID 说明表格。

【注意事项】

S 系列 S500/S300 不支持步长比较器

1.2.6 CC_TouchProbe

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
CC_TouchProbe	计数器探针	FB		CC_TouchProbe(Axis:= , Execute:= , Abort:= , ProbeID:= , TriggerEdge:= , Mode:= , Done=> , Busy=> , CommandAborted=> , PosPosition=> , NegPosition=> , PosPosition_P=> , NegPosition_P=> , PosTime=> , NegTime=> , CycleCount=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Encoder	-	-	计数器编号 [0..7]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Execute	使能	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：使能功能块。
Abort	终止	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：终止功能块。
ProbeID	探针 ID	USINT	[0,1]	0	探针 ID
TriggerEdge	触发边沿	USINT	[0,1]	0	0：上升沿 1：下降沿
Mode	触发模式	USINT	[0,1]	0	0：单次模式 1：连续模式

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Done	完成标志	BOOL	[FALSE,	-	如为单次模式，执行一次输出后，Done，功能自

			TRUE]		动终止。连续模式，此信号无用。
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
CommandAborted	终止标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块终止。
PosPosition	上升沿锁存位置	LREAL	-	-	上升沿锁存位置，单位：unit
NegPosition	下降沿锁存位置	LREAL	-	-	下降沿锁存位置，单位：unit
PosPosition_P	上升沿锁存位置	DINT	[-2147483648, 2147483647]	-	上升沿锁存位置，单位：pulse
NegPosition_P	下降沿锁存位置	DINT	[-2147483648, 2147483647]	-	下降沿锁存位置，单位：pulse
PosTime	上升沿锁存时刻	UDINT	[0, 4294967295]	-	上升沿锁存时刻，单位：us
NegTime	下降沿锁存时刻	UDINT	[0, 4294967295]	-	下降沿锁存时刻，单位：us
CycleCount	锁存次数	USINT	[0,31]	-	锁存次数
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

本功能块主要实现计数器的探针锁存功能，根据设置的触发边沿，当外部信号触发时，读取计数器当前值与系统当前时间输出，同时记录锁存次数。

【注意事项】

1、本功能块每路探针可以设置单次模式或连续模式，当使用连续模式时，每次触发都会更新对应输出值。同时，连续模式下，Done 信号不会有输出，此信号无效。

2、CycleCount 锁存次数存在范围，当锁存次数达到有效范围边界并继续增加时，该输出项会清零重新计数。

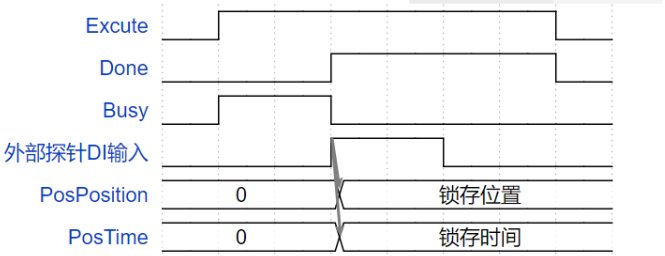
3、探针 ID 不支持在线修改。

4、每路计数器支持两种边沿触发模式，即上升沿锁存、下降沿锁存。

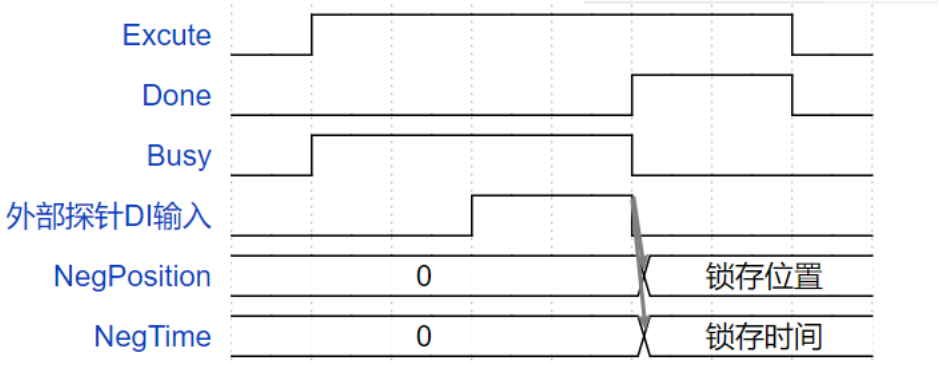
5、本功能块支持编码器 Z 信号锁存，需要使用编码器输入，即信号源设置为 EA-EB。

6、每路探针均同时支持时间、位置锁存。

单次触发模式（Mode=0），外部上升沿触发（TriggerEdge=0），输出端口号为 Y00-Y07/Y10-Y17，指令时序图如下图所示：



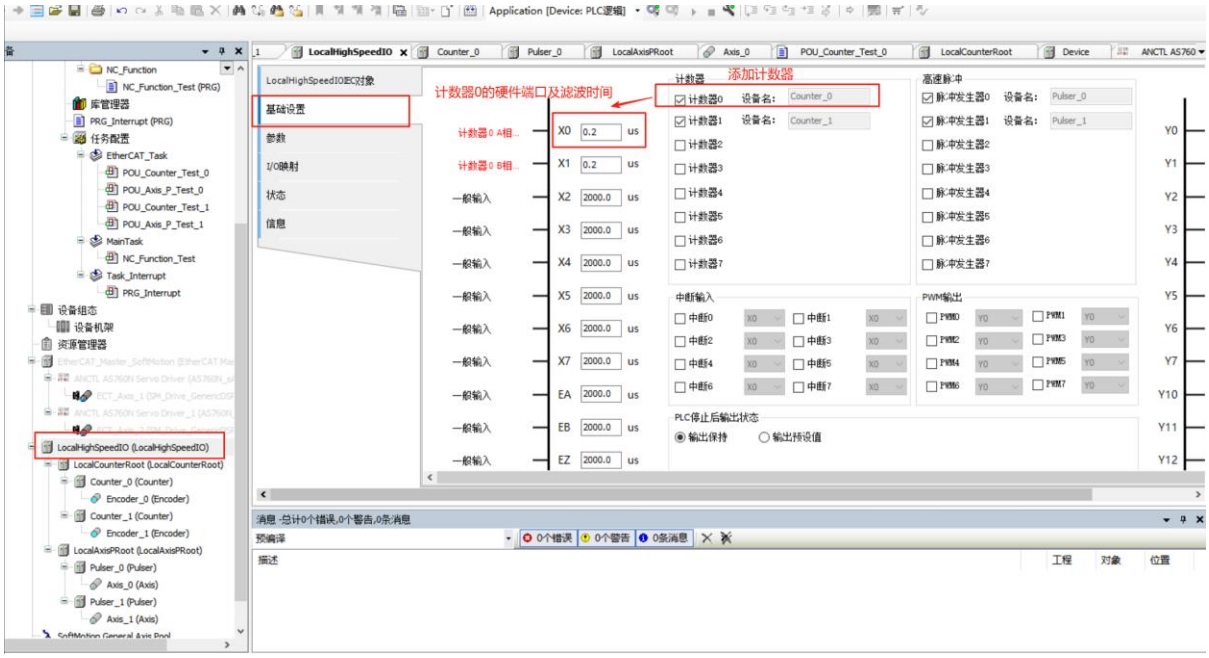
单次触发模式（Mode=0），外部下降沿触发（TriggerEdge=1）,输出端口号为 Y00-Y07/Y10-Y17，指令时序图如下图所示：



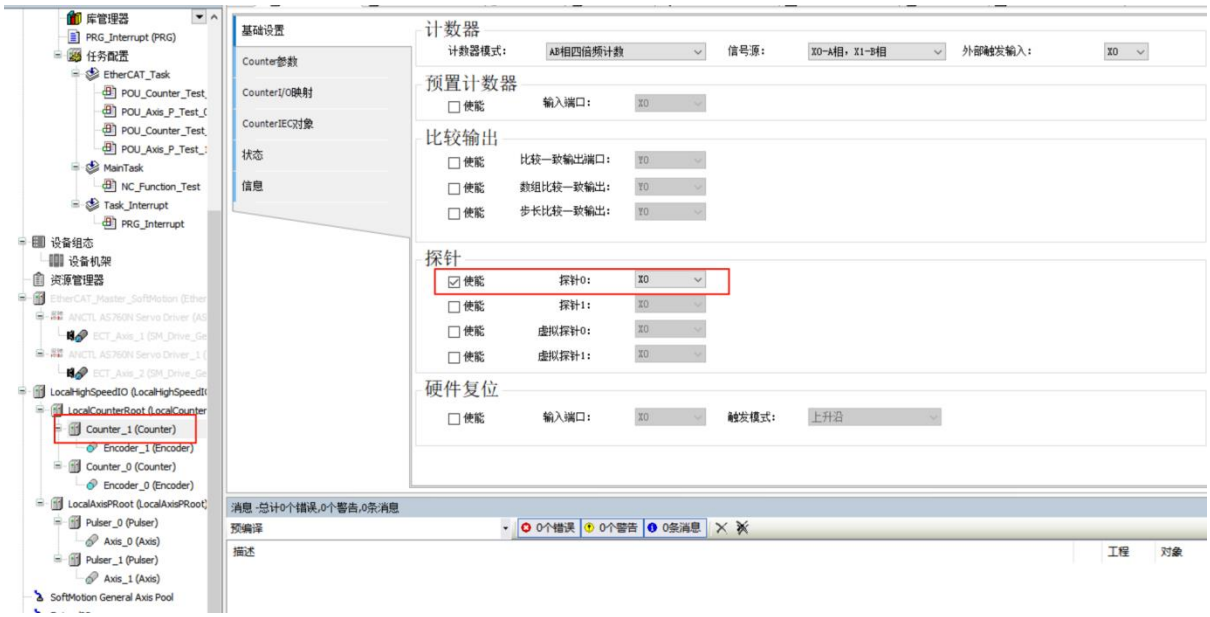
锁存计数+1

【图形化操作说明】

双击“LocalHighSpeedIO”，在“基础设置”中添加计数器，添加计数器后会自动按未占用硬件端口顺序分配硬件输入端口，并设置滤波时间

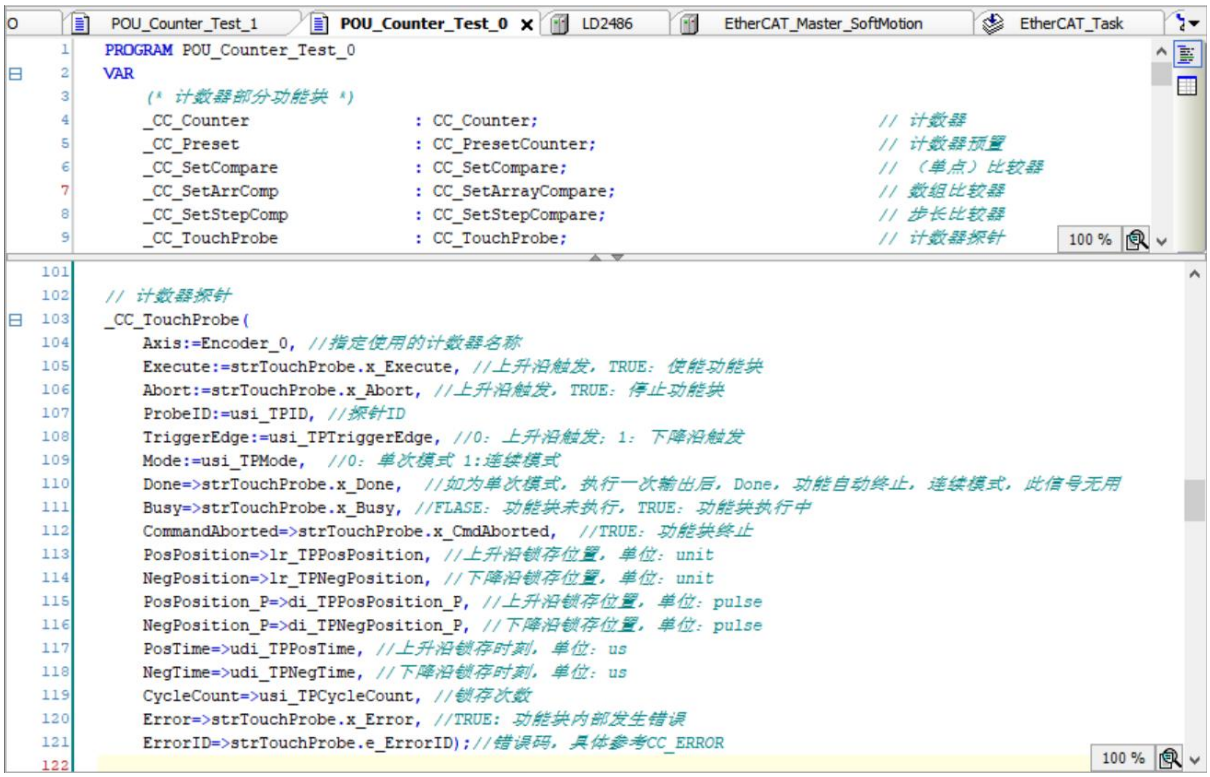


双击已添加的计数器“Counter_0”，使能“探针 0”或“探针 1”，选择“探针 0”或”探针 1“的硬件端口



【功能块操作说明】

声明计数器探针“_CC_TouchProbe: CC_TouchProbe;”



```
PROGRAM POU_Counter_Test_0
VAR
    (* 计数器部分功能块 *)
    _CC_Counter          : CC_Counter;          // 计数器
    _CC_Preset           : CC_PresetCounter;    // 计数器预置
    _CC_SetCompare       : CC_SetCompare;       // （单点）比较器
    _CC_SetArrComp       : CC_SetArrayCompare;  // 数组比较器
    _CC_SetStepComp      : CC_SetStepCompare;   // 步长比较器
    _CC_TouchProbe       : CC_TouchProbe;       // 计数器探针
```

```
// 计数器探针
_CC_TouchProbe(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute:=strTouchProbe.x_Execute, //上升沿触发，TRUE：使能功能块
    Abort:=strTouchProbe.x_Abort, //上升沿触发，TRUE：停止功能块
    ProbeID:=usi_TPID, //探针ID
    TriggerEdge:=usi_TPTriggerEdge, //0：上升沿触发；1：下降沿触发
    Mode:=usi_TPMode,  //0：单次模式 1:连续模式
    Done=>strTouchProbe.x_Done,  //如为单次模式，执行一次输出后，Done，功能自动终止，连续模式，此信号无用
    Busy=>strTouchProbe.x_Busy, //FLASE：功能块未执行，TRUE：功能块执行中
    CommandAborted=>strTouchProbe.x_CmdAborted,  //TRUE：功能块终止
    PosPosition=>lr_TPPosPosition, //上升沿锁存位置，单位：unit
    NegPosition=>lr_TPNegPosition, //下降沿锁存位置，单位：unit
    PosPosition_P=>di_TPPosPosition_P, //上升沿锁存位置，单位：pulse
    NegPosition_P=>di_TPNegPosition_P, //下降沿锁存位置，单位：pulse
    PosTime=>udi_TPPosTime, //上升沿锁存时刻，单位：us
    NegTime=>udi_TPNegTime, //下降沿锁存时刻，单位：us
    CycleCount=>usi_TPCycleCount, //锁存次数
    Error=>strTouchProbe.x_Error, //TRUE：功能块内部发生错误
    ErrorID=>strTouchProbe.e_ErrorID);//错误码，具体参考CC_ERROR
```

```
// 计数器探针
_CC_TouchProbe(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute TRUE :=strTouchProbe.x_Execute TRUE , //上升沿触发，TRUE：使能功能块
    Abort FALSE :=strTouchProbe.x_Abort FALSE , //上升沿触发，TRUE：停止功能块
    ProbeID 0 :=usi_TPID 0 , //探针ID
    TriggerEdge 0 :=usi_TPTriggerEdge 0 , //0：上升沿触发；1：下降沿触发
    Mode 0 :=usi_TPMode 0 , //0：单次模式 1:连续模式
    Done FALSE =>strTouchProbe.x_Done FALSE , //如为单次模式，执行一次输出后，Done，功能自动终止，连续模式，此信号无用
    Busy TRUE =>strTouchProbe.x_Busy TRUE , //FLASE：功能块未执行，TRUE：功能块执行中
    CommandAborted FALSE =>strTouchProbe.x_CmdAborted FALSE , //TRUE：功能块终止
    PosPosition 0 =>lr_TPPosPosition 0 , //上升沿锁存位置，单位：unit
    NegPosition 0 =>lr_TPNegPosition 0 , //下降沿锁存位置，单位：unit
    PosPosition_P 0 =>di_TPPosPosition_P 0 , //上升沿锁存位置，单位：pulse
    NegPosition_P 0 =>di_TPNegPosition_P 0 , //下降沿锁存位置，单位：pulse
    PosTime 0 =>udi_TPPosTime 0 , //上升沿锁存时刻，单位：us
    NegTime 0 =>udi_TPNegTime 0 , //下降沿锁存时刻，单位：us
    CycleCount 0 =>usi_TPCycleCount 0 , //锁存次数
    Error FALSE =>strTouchProbe.x_Error FALSE , //TRUE：功能块内部发生错误
    ErrorID CC_NO_ERRO =>strTouchProbe.e_ErrorID CC_NO_ERRO );//错误码，具体参考CC_ERROR
```

1，“Axis”引脚绑定添加的计数器名称;计数器探针功能块能够实现在设定的信号触发时锁存脉冲计数值。

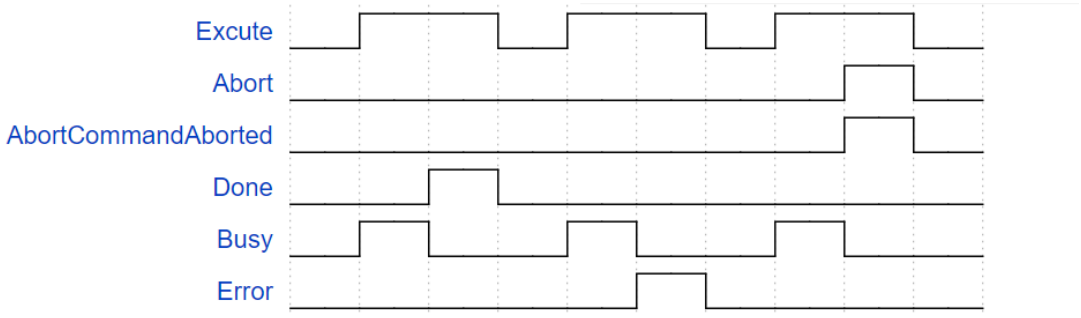
2，“ProbeID”引脚为探针的 ID

3，“Mode”引脚可以设定单次模式或连续模式，0：单次模式，1：连续模式

4，“CycleCount”引脚根据触发次数进行计数。

【时序图】

因为在连续模式下，Done 信号不会有输出，因此这里只展示单次模式下的时序图。

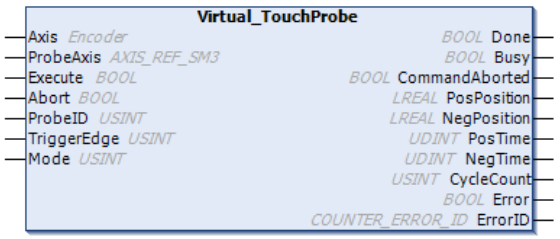


【错误说明】

详情查看 CC_ERROR-计数器错误 ID 说明表格。

1.2.7 Virtual_TouchProbe

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
Virtual_Touch Probe	EtherCAT 总线轴探针	FB		Virtual_TouchProbe(Axis:= , ProbeAxis:= , Execute:= , Abort:= , ProbeID:= , TriggerEdge:= , Mode:= , Done=> , Busy=> , CommandAborted=> , PosPosition=> , NegPosition=> , PosTime=> , NegTime=> , CycleCount=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Encoder	-	-	计数器编号 [0..3]。
ProbeAxis	EtherCAT 轴	AXIS_REF_SM3	-	-	EtherCAT 实轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Execute	使能	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：使能功能块。
Abort	终止	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：终止功能块。
ProbeID	探针 ID	USINT	[0,1]	0	探针 ID
TriggerEdge	触发边沿	USINT	[0,1]	0	0：上升沿 1：下降沿
Mode	触发模式	USINT	[0,1]	0	0：单次模式 1：连续模式

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Done	完成标志	BOOL	[FALSE,	-	如为单次模式，执行一次输出后，Done，功能自

			TRUE]		动终止。连续模式，此信号无用。
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
CommandAborted	终止标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块终止。
PosPosition	上升沿锁存位置	LREAL	-	-	上升沿锁存位置，单位：unit
NegPosition	下降沿锁存位置	LREAL	-	-	下降沿锁存位置，单位：unit
PosTime	上升沿锁存时刻	UDINT	[0, 4294967295]	-	上升沿锁存时刻，单位：us
NegTime	下降沿锁存时刻	UDINT	[0, 4294967295]	-	下降沿锁存时刻，单位：us
CycleCount	锁存次数	USINT	[0,31]	-	锁存次数
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

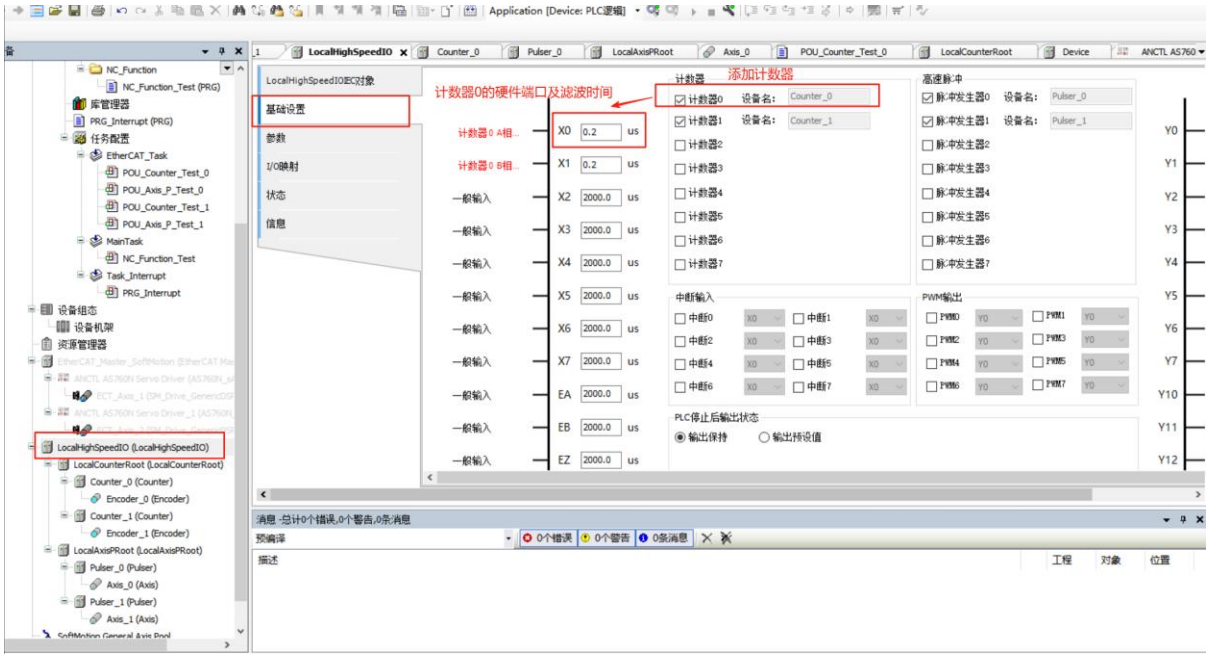
本功能块主要实现 EtherCAT 轴虚拟探针功能。

【注意事项】

见计数器探针部分。

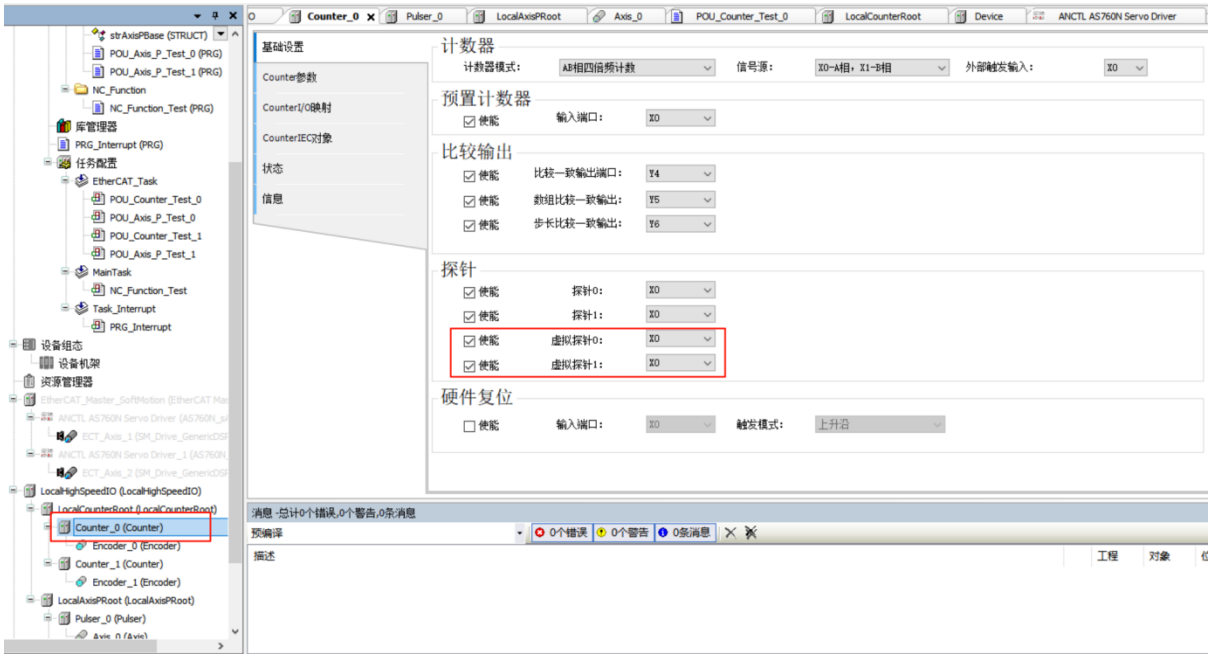
【图形化操作说明】

双击“LocalHighSpeedIO”，在“基础设置”中添加计数器，添加计数器后会自动按顺序分配硬件输入端口，并设置滤波时间



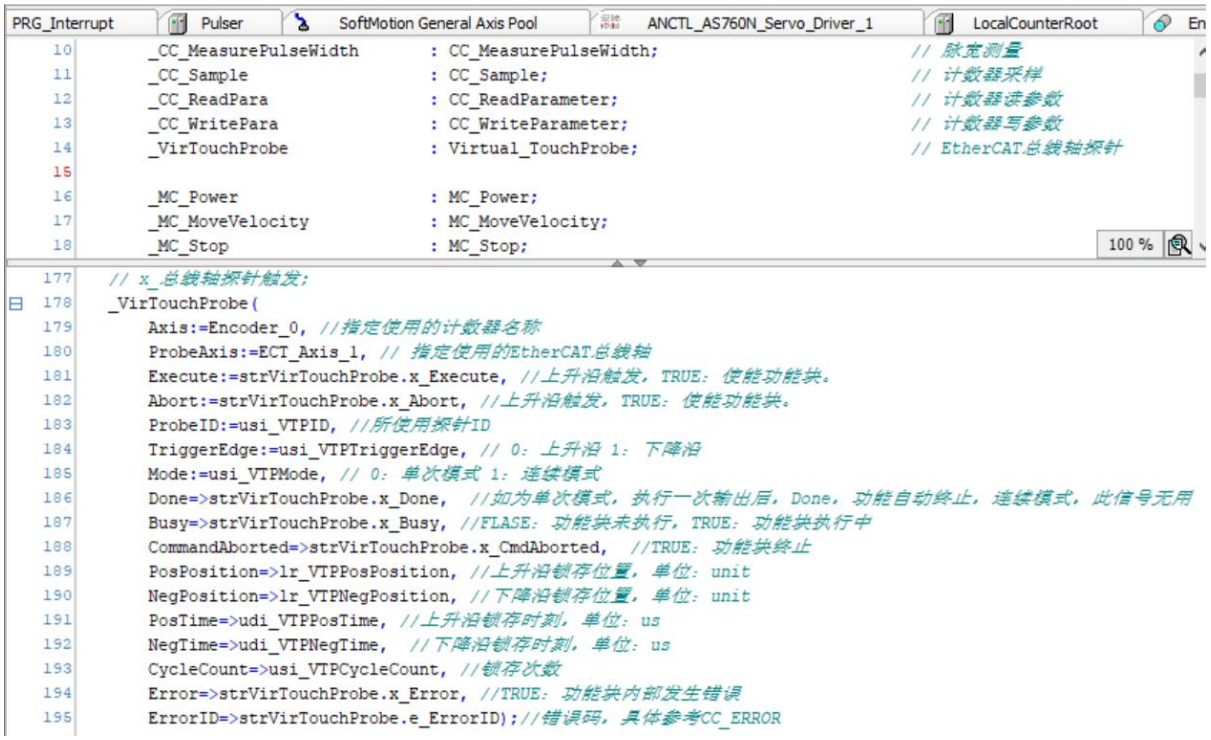
双击已添加的计数器“Counter_0”，使能“虚拟探针 0”或”虚拟探针 1“，选择“虚拟探针

0”和“虚拟探针 1”的硬件端口



【功能块操作说明】

声明 EtherCAT 总线轴探针功能块“_VirTouchProbe: Virtual_TouchProbe;”



```
// x_总线轴探针触发;
_VirTouchProbe(
    Axis:=Encoder_0, //指定使用的计数器名称
    ProbeAxis:=ECT_Axis_1, // 指定使用的EtherCAT总线轴
    Execute TRUE :=strVirTouchProbe.x_Execute TRUE , //上升沿触发，TRUE：使能功能块。
    Abort FALSE :=strVirTouchProbe.x_Abort FALSE , //上升沿触发，TRUE：使能功能块。
    ProbeID 0 :=usi_VTPID 0 , //所使用探针ID
    TriggerEdge 0 :=usi_VTPTriggerEdge 0 , // 0：上升沿 1：下降沿
    Mode 0 :=usi_VTPMode 0 , // 0：单次模式 1：连续模式
    Done TRUE =>strVirTouchProbe.x_Done TRUE ,  //如为单次模式，执行一次输出后，Done，功能自动终止，连续模式，此信号无用
    Busy FALSE =>strVirTouchProbe.x_Busy FALSE , //FLASE：功能块未执行，TRUE：功能块执行中
    CommandAborted FALSE =>strVirTouchProbe.x_CmdAborted FALSE ,  //TRUE：功能块终止
    PosPosition 189 =>lr_VTPPosPosition 189 , //上升沿锁存位置，单位：unit
    NegPosition 0 =>lr_VTPNegPosition 0 , //下降沿锁存位置，单位：unit
    PosTime 4271777766 =>udi_VTPPosTime 4271777766 , //上升沿锁存时刻，单位：us
    NegTime 0 =>udi_VTPNegTime 0 ,  //下降沿锁存时刻，单位：us
    CycleCount 1 =>usi_VTPCycleCount 1 , //锁存次数
    Error FALSE =>strVirTouchProbe.x_Error FALSE , //TRUE：功能块内部发生错误
    ErrorID CC_NO_ERRO =>strVirTouchProbe.e_ErrorID CC_NO_ERRO );//错误码，具体参考CC_ERROR
```

1，“Axis”引脚绑定添加的计数器名称;计数器探针功能块能够实现在设定的信号触发时锁存脉冲计数值。

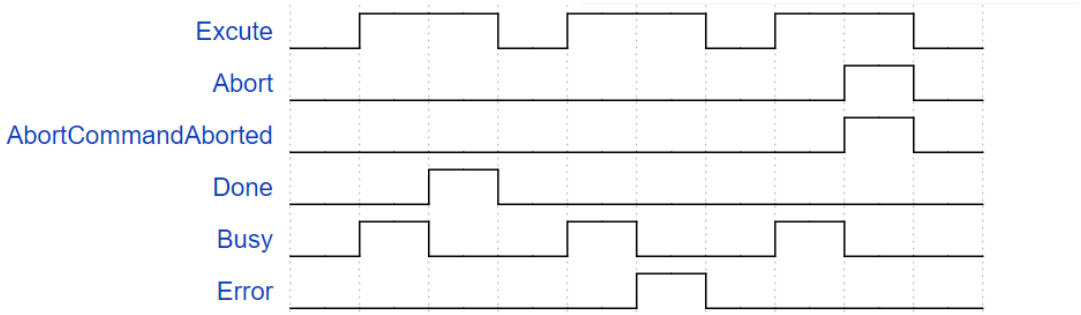
2，“ProbeAxis”引脚指定使用的 EtherCAT 总线轴

3，“Mode”引脚可以设定单次模式或连续模式，0：单次模式，1：连续模式

5，“CycleCount”引脚输出值为锁存次数。

【时序图】

因为在连续模式下，Done 信号不会有输出，因此这里只展示单次模式下的时序图。

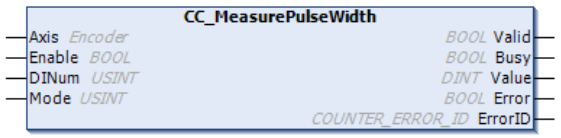


【错误说明】

详情查看 CC_ERROR-计数器错误 ID 说明表格。

1.2.8 CC_MeasurePulseWidth

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
CC_MeasurePulseWidth	脉宽测量	FB		CC_MeasurePulseWidth(Axis:= , Enable:= , DINum:= , Mode:= , Valid=> , Busy=> , Value=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Encoder	-	-	计数器编号 [0..7]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Enable	使能	BOOL	[FALSE, TRUE]	FALSE	电平触发，TRUE：使能功能块。
DINum	DI 端口号	USINT	[0,10]	0	DI 端口号 0-7：X0-X7； 8：EA；9：EB；10：EZ
Mode	模式	USINT	[0,1]	0	0：测量高电平脉宽 1：测量低电平脉宽

输出变量

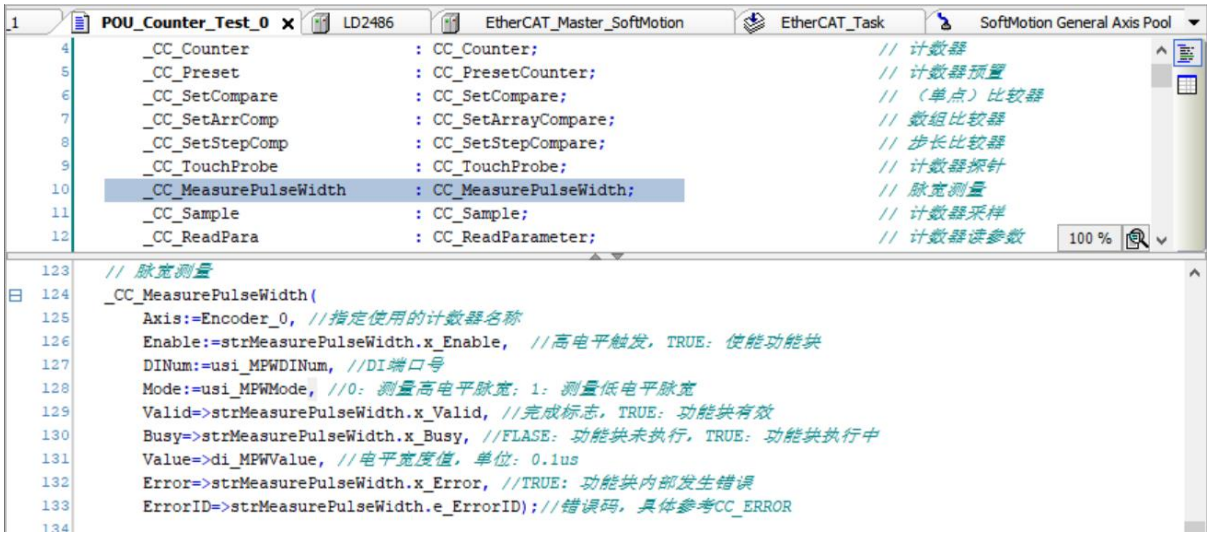
输出变量	名称	数据类型	有效范围	初始值	说明
Valid	完成标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块有效。
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
Value	电平宽度值	DINT	[0, 2147483647]	-	电平宽度值，单位：0.1us
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

本功能块主要实现 DI 端口的脉冲宽度测量功能，Enable 使能时，测量外部信号脉宽。Mode=0，测量外部信号高电平脉宽，对应输出 Value 值为高电平宽度值；Mode=1，测量外部信号低电平脉宽，对应输出 Value 值为低电平宽度值。

【功能块操作说明】

声明脉宽测量“_CC_MeasurePulseWidth: CC_MeasurePulseWidth;”实例化



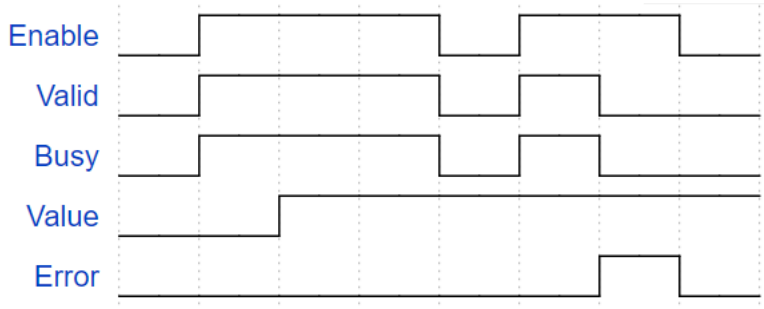

```
// 脉宽测量
_CC_MeasurePulseWidth(
    Axis:=Encoder_0, //指定使用的计数器名称
    Enable TRUE :=strMeasurePulseWidth.x_Enable TRUE , //高电平触发，TRUE：使能功能块
    DINum 0 :=usi_MPWDINum 0 , //DI端口号
    Mode 1 :=usi_MPWMode 1 , //0：测量高电平脉宽；1：测量低电平脉宽
    Valid TRUE =>strMeasurePulseWidth.x_Valid TRUE , //完成标志，TRUE：功能块有效
    Busy TRUE =>strMeasurePulseWidth.x_Busy TRUE , //FLASE：功能块未执行，TRUE：功能块执行中
    Value 39999980 =>di_MPWValue 39999980 , //电平宽度值，单位：0.1us
    Error FALSE =>strMeasurePulseWidth.x_Error FALSE , //TRUE：功能块内部发生错误
    ErrorID CC_NO_ERRO =>strMeasurePulseWidth.e_ErrorID CC_NO_ERRO );//错误码，具体参考CC_ERROR
```

1，“Axis”引脚绑定添加的计数器名称;脉宽测量功能块能够实现测量计数器 Counter 接收的脉冲宽度。

2，“Enable”引脚触发激活功能块，开始测量脉宽。

3，“Mode”引脚可以设定测量的脉宽高低电平，0：测量高电平脉宽，1：测量低电平脉宽。

【时序图】

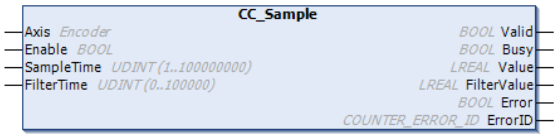


【错误说明】

详情查看 CC_ERROR-计数器错误 ID 说明表格。

1.2.9 CC_Sample

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
CC_Sample	计数器采样	FB		CC_Sample(Axis:= , Enable:= , SampleTime:= , FilterTime:= , Valid=> , Busy=> , Value=> , FilterValue=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Encoder	-	-	计数器编号 [0..7]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Enable	使能	BOOL	[FALSE, TRUE]	FALSE	电平触发，TRUE：使能功能块。
SampleTime	采样时间	UDINT	[1,100000000]	-	采样时间，单位：us
FilterTime	软件滤波	UDINT	[0,100000]	0	软件滤波，单位：ms

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Valid	有效标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块有效。
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
Value	采样值	LREAL	-	-	采样值，单位：unit
FilterValue	滤波后的采样值	LREAL	-	-	滤波后的采样值，单位：unit
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

本功能块主要实现计数器的采样功能，统计一段时间中的计数个数。

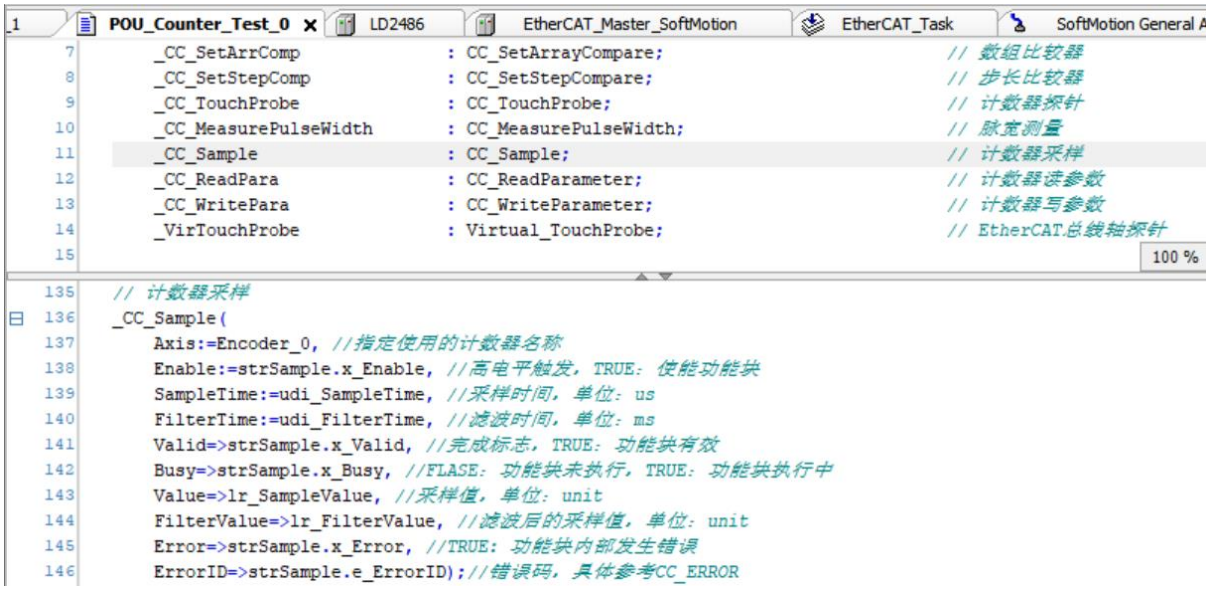
【注意事项】

输入变量中的软件滤波用于提升采样质量，一般情况下，采样精度足够使用，可以不设置该滤波时间。

采样时间和软件滤波参数的单位不一致，在使用时注意区分。

【功能块操作说明】

声明计数器采样“_CC_Sample: CC_Sample;”实例化




```
// 计数器采样
_CC_Sample(
    Axis:=Encoder_0, //指定使用的计数器名称
    Enable TRUE :=strSample.x_Enable TRUE , //高电平触发，TRUE：使能功能块
    SampleTime 5000000 :=udi_SampleTime 5000000 , //采样时间，单位：us
    FilterTime 1 :=udi_FilterTime 1 , //滤波时间，单位：ms
    Valid TRUE =>strSample.x_Valid TRUE , //完成标志，TRUE：功能块有效
    Busy TRUE =>strSample.x_Busy TRUE , //FLASE：功能块未执行，TRUE：功能块执行中
    Value 2 =>lr_SampleValue 2 , //采样值，单位：unit
    FilterValue 2 =>lr_FilterValue 2 , //滤波后的采样值，单位：unit
    Error FALSE =>strSample.x_Error FALSE , //TRUE：功能块内部发生错误
    ErrorID CC_NO_ERRO =>strSample.e_ErrorID CC_NO_ERRO );//错误码，具体参考CC_ERROR
```

1，“Axis”引脚绑定添加的计数器名称;计数器采样 Sample 功能块能够实现将计数器 Counter 接收的脉冲进一步筛选，得到目标脉冲信号。

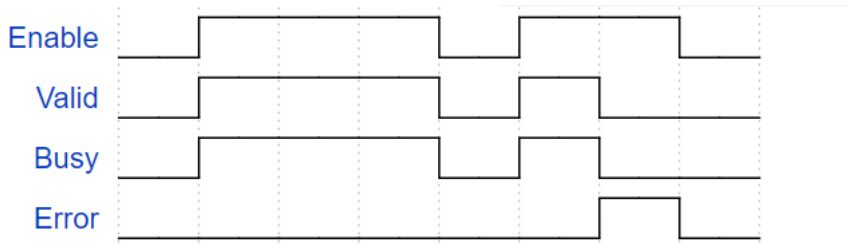
2，“Enable”引脚触发，功能块生效。

3，“SampleTime”引脚设定采样时间，“FilterTime”设定滤波值，在功能块生效后根据接收到的脉冲进一步筛选，采样值在“FilterValue”中体现。

4，“Value”引脚体现接收到的值，从而可以和滤波后的“FilterValue”引脚值进行比对，确定样本中符合与不符合的脉冲量。

【时序图】

时序图暂时只展示不考虑软件滤波的情况。



注：采样时间间隔内，采取该段时间内计数个数，当采样时间结束，输出采样值。

【错误说明】

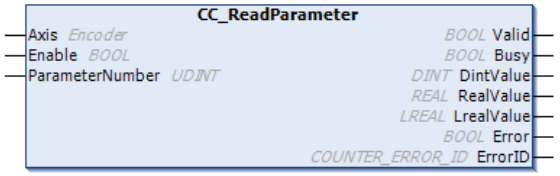
详情查看 CC_ERROR-计数器错误 ID 说明表格。

【注意事项】

S 系列 S500/S300 不支持计数器采样

1.2.10 CC_ReadParameter

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
CC_ReadParameter	计数器读参数	FB		CC_ReadParameter(Axis:= , Enable:= , ParameterNumber:= , Valid=> , Busy=> , DintValue=> , RealValue=> , LrealValue=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Encoder	-	-	计数器编号 [0..7]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Enable	使能	BOOL	[FALSE, TRUE]	FALSE	电平触发，TRUE：使能功能块。
ParameterNum	参数编号	UINT	-	-	参数编号，请查表

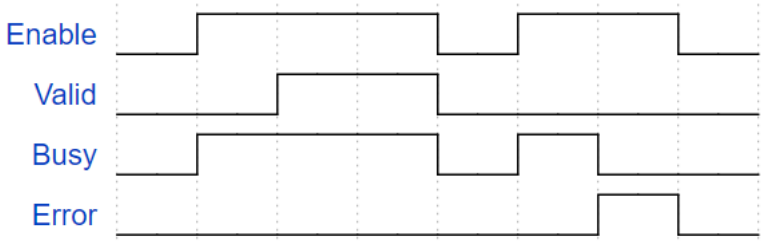
输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Valid	完成标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块有效。
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
DintValue	32 位整型	DINT	遵照数据类型	-	32 位整型，如果是参数编号为整型数据（BOOL BYTE,WORD 等），从该变量中读取
RealValue	32 位浮点型	Real	遵照数据类型	-	32 位浮点型，如果是参数编号为浮点数据（REAL），从该变量中读取
LrealValue	64 位浮点型	LReal	遵照数据类型	-	64 位浮点型，如果是参数编号为双精度浮点数据（LREAL），从该变量中读取
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

注：本功能保留，暂不支持使用。

【时序图】

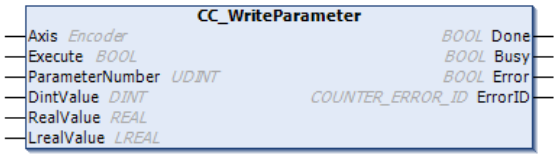


【错误说明】

详情查看 CC_ERROR-计数器错误 ID 说明表格。

1.2.11 CC_WriteParameter

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
CC_WriteParameter	计数器写参数	FB		CC_WriteParameter(Axis:= , Execute:= , ParameterNumber:= , DintValue:= , RealValue:= , LrealValue:= , Done=> , Busy=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Encoder	-	-	计数器编号 [0..7]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Execute	使能	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：使能计数器预置。
ParameterNum	参数编号	UINT	-	-	参数编号，暂无可写。预留功能块。
DintValue	32 位整型	DINT	遵照数据类型	-	32 位整型，如果参数为整型（BOOL,BYTE,WORD 等），赋值到此变量
RealValue	32 位浮点型	Real	遵照数据类型	-	32 位浮点型，如果参数为浮点型（REAL），赋值到此变量
LrealValue	64 位浮点型	LReal	遵照数据类型	-	64 位浮点型，如果参数为双精度浮点型（LREAL），赋值到此变量

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Done	完成标志	BOOL	[FALSE, TRUE]	-	TRUE：预置完成
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【注意！！！】

本功能块尚未完善，参数编号未给出，属于预留功能块。

【错误说明】

详情查看 CC_ERROR-计数器错误 ID 说明表格。

1.3 辅助功能

辅助功能参数说明。

输入变量

FilterParam X_	输入滤波参数				分别对应硬件输入端口 X0-X7，EA，EB，EZ
ON_OFF	滤波启用	BOOL	[FALSE, TRUE]	TRUE	输入滤波启用
FilterTime	滤波时间	UDINT	[0,10000000]	20000	输入滤波时间，单位 0.1us，默认值 20000→2ms
PlcStopDoValue	PLC 停止输出模式	BOOL	[FALSE, TRUE]	FALSE	FALSE：keepCurrent，保持当前值； TRUE：PresetDoValue，设置为预设输出值
PresetDoValue	PLC 停止预设输出值	UINT	[0,65535]	0	当 PlcStopDoValue 为 PresetDoValue 时，PLC 停止时，将会按照预设的输出值进行输出
EnableInterruptEnable	中断使能	BOOL	[FALSE, TRUE]	FALSE	中断使能
EnableInterruptID	中断 ID 号	USINT	[0,7]	0	中断 ID 号
EnableInterruptDINum	中断 DI 输入端口	USINT	[0,10]	0	中断 DI 输入端口 0-7：X0-X7； 8：EA；9：EB；10：EZ
PwmEnable	Pwm 使能	BOOL	[FALSE, TRUE]	FALSE	Pwm 使能
PwmID	PwmID 号	USINT	[0,7]	0	PwmID 号
PwmDONum	PwmDO 端口	USINT	[0,15]	Y0	PwmDO 端口 0-7：Y0-Y7； 8-15：Y10-Y17

1.3.1 NC_EnableInterrupt

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
NC_EnableInterrupt	中断	FB		NC_EnableInterrupt(IRQNum:= , Enable:= , Mode:= , TriggerEdge:= , CompareEvent:= , Valid=> , Busy=> , Error=> , ErrorID=>);

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
IRQNum	中断 ID	USINT	[0,7]	0	中断 ID 号
Enable	使能	BOOL	[FALSE, TRUE]	FALSE	电平触发，TRUE：使能功能块。
Mode	模式	USINT	[0,1]	0	0：输入触发 1：计数比较一致触发
TriggerEdge	触发边沿	USINT	[0,2]	0	0：上升沿 1：下降沿 2：上升沿或下降沿 注意：当边沿为上升或下降沿时，是指上升沿或下降沿都会触发，而不是与的关系。
CompareEvent	比较器触发	USINT	[0,7]	0	比较器事件

输出变量

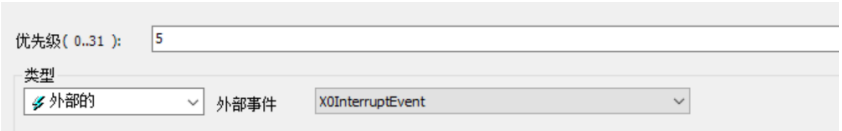
输出变量	名称	数据类型	有效范围	初始值	说明
Valid	完成标志	BOOL	[FALSE, TRUE]	-	TRUE：触发中断时有效。
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

本功能块主要实现本地打开外部中断或计数器比较一致中断，并可以设置触发的边沿为上升沿还是下降沿。

示例：

使用时，除了实例化该功能块外，可以在任务配置添加外部中断任务，如下图所示，



当有中断触发时，执行该外部事件任务下的 POU 程序。

【注意事项】

中断的 ID 号 IRQNum 不能在线修改，否则会报错。

计数器比较输出事件设定的计数器必须要在 Mode 模式选择 1-计数器中断，以及该计数器开启比较一致输出功能才可使用。

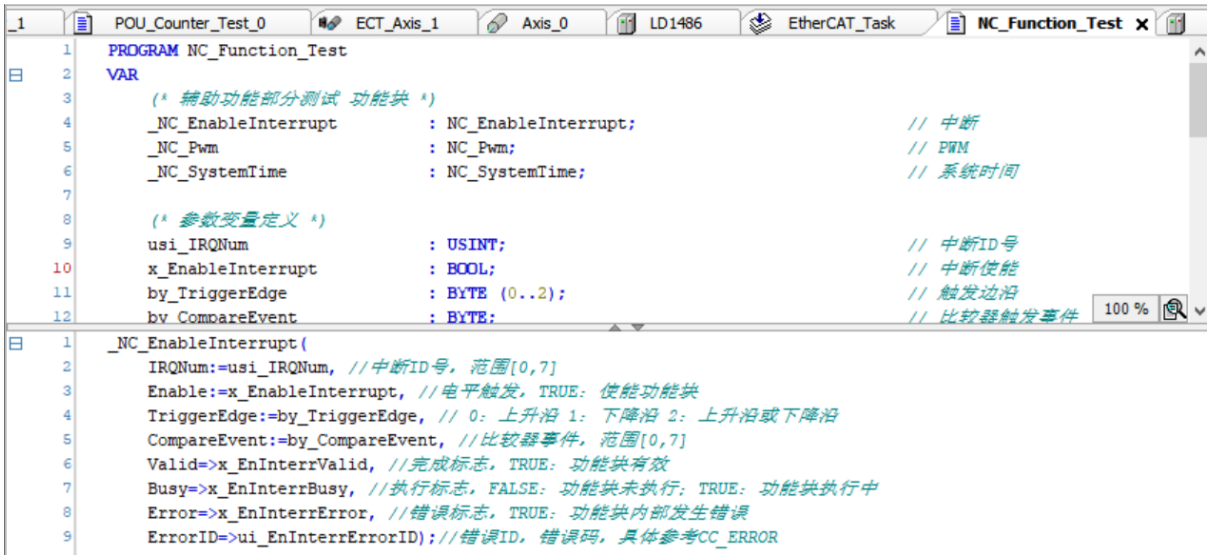
为保证该功能块正常使用，请先设置好各项参数，再使能该功能块。

【图形化操作说明】

双击“LocalHighSpeedIO”，在“基础设置”界面使能“中断输入”，并设置中断输入的硬件端口，产生中断后会跳入中断处理程序

【功能块操作说明】

声明计数器采样“_NC_EnableInterrupt: NC_EnableInterrupt; ”实例化



```
PROGRAM NC_Function_Test
VAR
    (* 辅助功能部分测试 功能块 *)
    _NC_EnableInterrupt         : NC_EnableInterrupt;          // 中断
    _NC_Pwm                     : NC_Pwm;                      // PWM
    _NC_SystemTime              : NC_SystemTime;               // 系统时间

    (* 参数变量定义 *)
    usi_IRQNum                  : USINT;                       // 中断ID号
    x_EnableInterrupt           : BOOL;                        // 中断使能
    by_TriggerEdge              : BYTE (0..2);                 // 触发边沿
    by_CompareEvent             : BYTE;                        // 比较器触发事件
```

```
_NC_EnableInterrupt(
    IRQNum:=usi_IRQNum, //中断ID号，范围[0,7]
    Enable:=x_EnableInterrupt, //电平触发，TRUE：使能功能块
    TriggerEdge:=by_TriggerEdge, // 0：上升沿 1：下降沿 2：上升沿或下降沿
    CompareEvent:=by_CompareEvent, //比较器事件，范围[0,7]
    Valid=>x_EnInterrValid, //完成标志，TRUE：功能块有效
    Busy=>x_EnInterrBusy, //执行标志，FALSE：功能块未执行；TRUE：功能块执行中
    Error=>x_EnInterrError, //错误标志，TRUE：功能块内部发生错误
    ErrorID=>ui_EnInterrErrorID);//错误ID，错误码，具体参考CC_ERROR
```

```
_NC_EnableInterrupt(
    IRQNum 0 :=usi_IRQNum 0 , //中断ID号，范围[0,7]
    Enable TRUE :=x_EnableInterrupt TRUE , //电平触发，TRUE：使能功能块
    TriggerEdge 0 :=by_TriggerEdge 0 , // 0：上升沿 1：下降沿 2：上升沿或下降沿
    CompareEvent 0 :=by_CompareEvent 0 , //比较器事件，范围[0,7]
    Valid FALSE =>x_EnInterrValid FALSE , //完成标志，TRUE：功能块有效
    Busy TRUE =>x_EnInterrBusy TRUE , //执行标志，FALSE：功能块未执行；TRUE：功能块执行中
    Error FALSE =>x_EnInterrError FALSE , //错误标志，TRUE：功能块内部发生错误
    ErrorID 0 =>ui_EnInterrErrorID 0 );//错误ID，错误码，具体参考CC_ERROR
```

1，“IRQNum”引脚绑定中断 ID 号。

2，“TriggerEdge”引脚指定触发边缘，0：上升沿；1：下降沿；2：上升沿或下降沿。

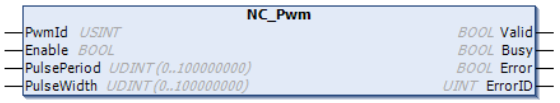
3，“ComparentEvent”引脚可以设定比较器事件，范围为 0-7。

【错误说明】

详情查看 CC_ERROR-计数器错误 ID 说明表格。

1.3.2 NC_Pwm

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
NC_Pwm	PWM	FB		NC_Pwm(PwmId:= , Enable:= , PulsePeriod:= , PulseWidth:= , Valid=> , Busy=> , Error=> , ErrorID=>);

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
PwmId	PWM ID	USINT	[0,7]	0	PWM ID 号
Enable	使能	BOOL	[FALSE, TRUE]	FALSE	电平触发，TRUE：使能功能块。
PulsePeriod	脉冲频率	UDINT	[0, 100000000]	0	输出脉冲周期，单位：0.1us 最大输出频率实测可达 400kHz，即 2.5us。
PulseWidth	脉冲宽度	UDINT	[0, 100000000]	0	输出脉冲高电平，单位：0.1us

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Valid	完成标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块有效。
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

该功能块用于实现动态的脉宽调制输出。

当该功能被触发时，如果修改 DO 输出端口号，则会直接跳转新设定的输出端口进行输出，不必将 Enable 引脚重新置 True。

如果设定脉冲高电平时长大于脉冲频率，不会报错，会持续有高电平输出。

注：PwmID 要与当前设备计数器 CounterID、EnableInterruptID 保持一致。

【图形化操作说明】

双击“LocalHighSpeedIO”，在“基础设置”中添加 PWM 输出，添加 PWM 后会自动按顺序分配硬件输出端口，也可自行手动设置

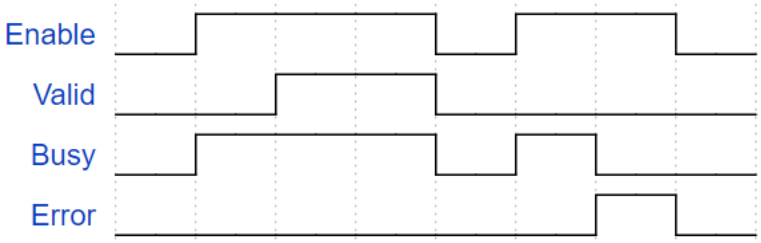
【功能块操作说明】

声明 PWM“_NC_Pwm: NC_Pwm;”实例化

```
PROGRAM NC_Function_Test
VAR
    (* 辅助功能部分测试 功能块 *)
    _NC_EnableInterrupt         : NC_EnableInterrupt;          // 中断
    _NC_Pwm                     : NC_Pwm;                      // PWM
    _NC_SystemTime              : NC_SystemTime;               // 系统时间

    (* 参数变量定义 *)
    usi_IRQNum                  : USINT;                       // 中断ID号
    x_EnableInterrupt           : BOOL;                        // 中断使能
    by_TriggerEdge              : BYTE (0..2);                 // 触发边沿
    by_CompareEvent             : BYTE;                        // 比较器触发事件

_NC_Pwm(
    PwmId:=usi_PwmId, // PWM ID号，范围[0,7]
    Enable:=x_EnablePwm, //电平触发，TRUE：使能功能块
    PulsePeriod:=udi_PulsePeriod, // 输出脉冲周期，单位：0.1us 最大输出频率200kHz，也即5.0us 范围[50,100000000]
    PulseWidth:=udi_PulseWidth, // 输出脉冲高电平，单位：0.1us
    Valid=>x_PwmValid, //完成标志，TRUE：功能块有效
    Busy=>x_PwmBusy, //执行标志，FALSE：功能块未执行；TRUE：功能块执行中
    Error=>x_PwmError, //错误标志，TRUE：功能块内部发生错误
    ErrorID=>ui_PwmErrorID);//错误ID，错误码，具体参考CC_ERROR
```

```
_NC_Pwm(
    PwmId 0 :=usi_PwmId 0 , // PWM ID号，范围[0,7]
    Enable TRUE :=x_EnablePwm TRUE , //电平触发，TRUE：使能功能块
    PulsePeriod 100000000 :=udi_PulsePeriod 100000000 , // 输出脉冲周期，单位：0.1us 最大输出频率200kHz，也即5.0us 范围[50,100000000]
    PulseWidth 90000000 :=udi_PulseWidth 90000000 , // 输出脉冲高电平，单位：0.1us
    Valid TRUE =>x_PwmValid TRUE , //完成标志，TRUE：功能块有效
    Busy TRUE =>x_PwmBusy TRUE , //执行标志，FALSE：功能块未执行；TRUE：功能块执行中
    Error FALSE =>x_PwmError FALSE , //错误标志，TRUE：功能块内部发生错误
    ErrorID 0 =>ui_PwmErrorID 0 );//错误ID，错误码，具体参考CC_ERROR
```

【时序图】

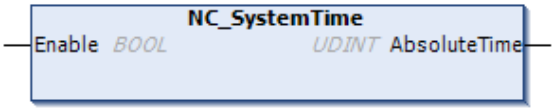


【错误说明】

详情查看 CC_ERROR-计数器错误 ID 说明表格。

1.3.3 NC_SystemTime

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
NC_SystemTime	系统时间	FB		NC_SystemTime(Enable:= , AbsoluteTime=>);

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Enable	使能	BOOL	[FALSE, TRUE]	FALSE	电平触发，TRUE：使能功能块。

输出变量

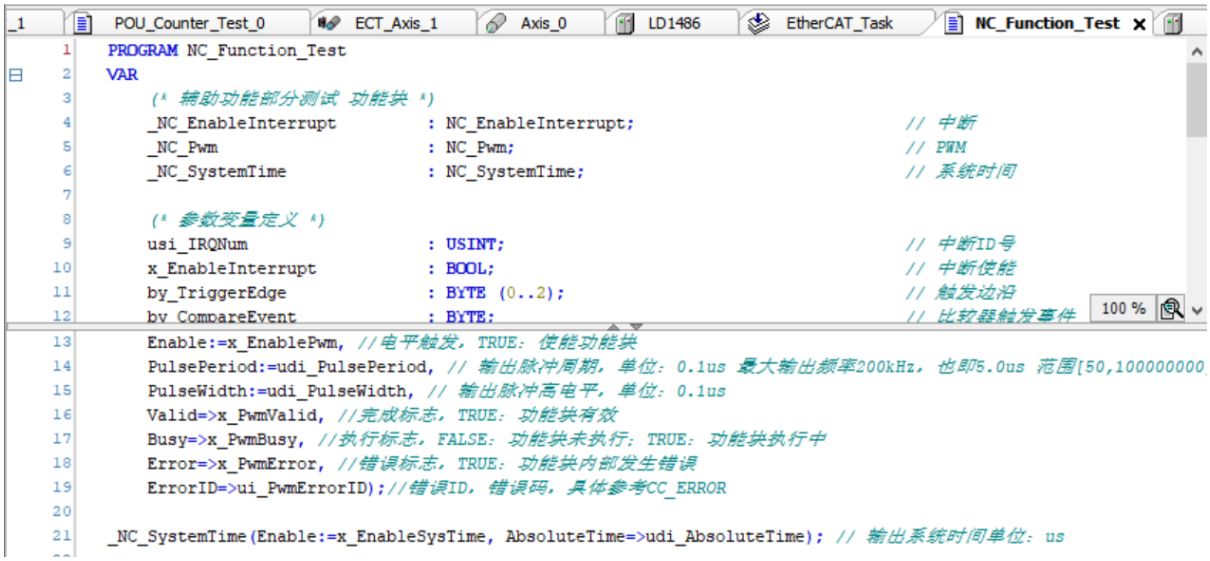
输出变量	名称	数据类型	有效范围	初始值	说明
AbsoluteTime	系统时间	UDINT	[0, 4294967295]	-	系统时间，可以用于补偿计算，单位：us。

【功能说明】

该功能块用于获取系统时间，可用于计数器值补偿计算，Enable 置 True 即可触发，置 False，则关闭该功能。

【功能块操作说明】

声明系统时间“_NC_SystemTime: NC_SystemTime;”

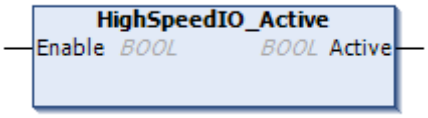


功能块操作说明

```
_NC_SystemTime(Enable TRUE :=x_EnableSysTime TRUE , AbsoluteTime 3213384490 =>udi_AbsoluteTime 3213384490 ); // 输出系统时间单位：us
```

1.3.4 HighSpeedIO_Active

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
HighSpeedIO_Active	高速 IO 状态	FB		HighSpeedIO_Active(Enable:= , Active=>);

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Enable	使能	BOOL	[FALSE, TRUE]	FALSE	电平触发，TRUE：使能功能块。

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Active	高速 IO 功能状态	BOOL	[FALSE, TRUE]	-	FALSE：高速 IO 功能停止 TRUE：高速 IO 功能运行中

【功能说明】

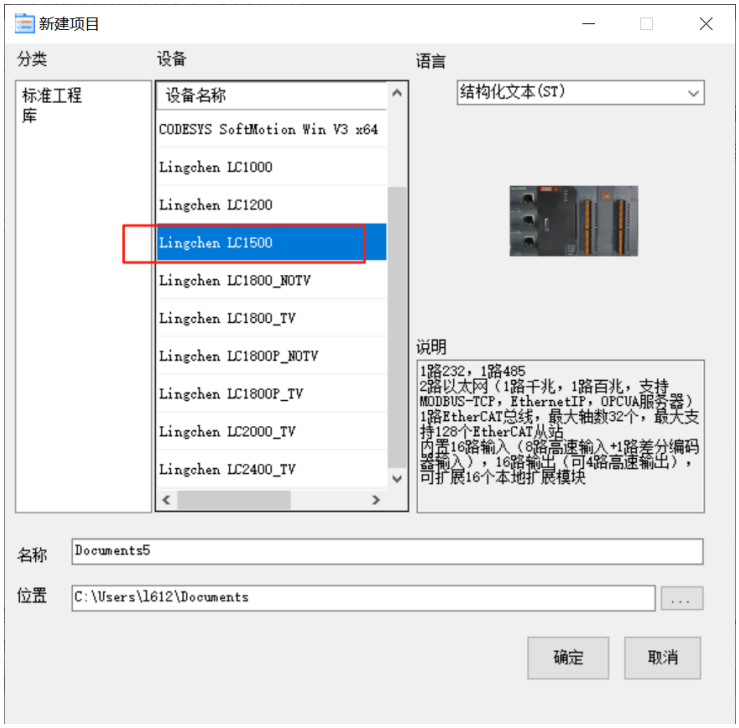
该功能块用于获取高速 IO 功能的运行状态。高速 IO 功能运行后，与其相关的功能才能使用。

1.4 脉冲输出功能

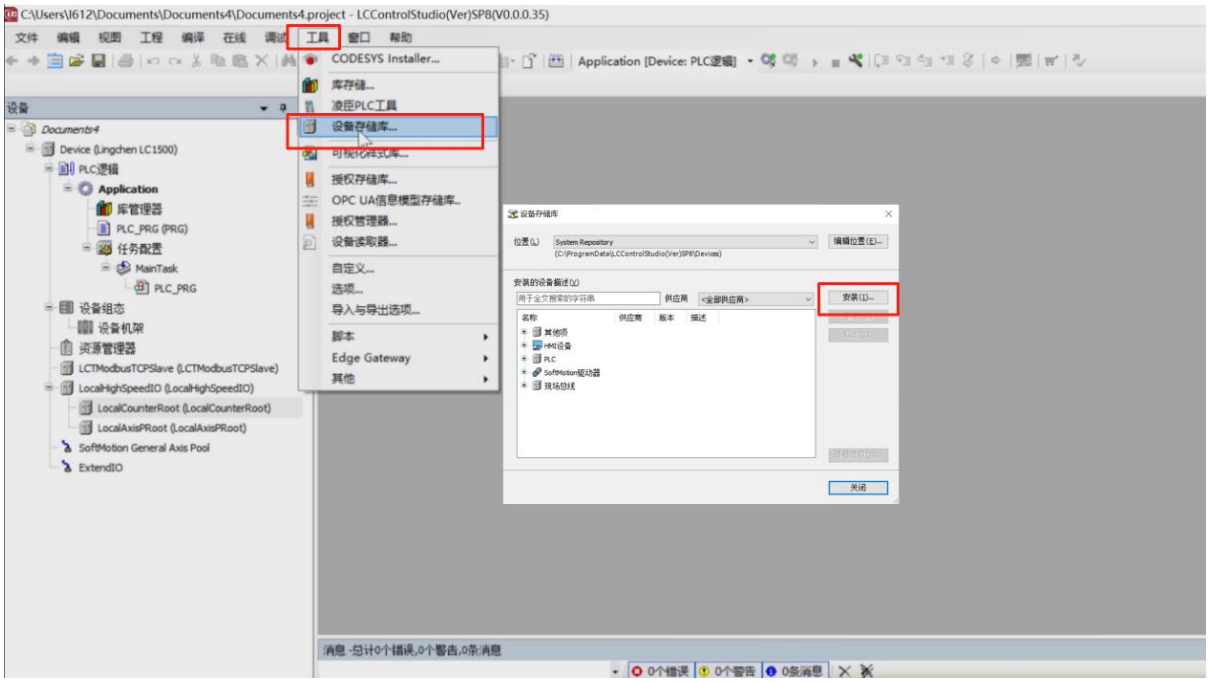
脉冲轴配置

1 XML 文件安装及应用

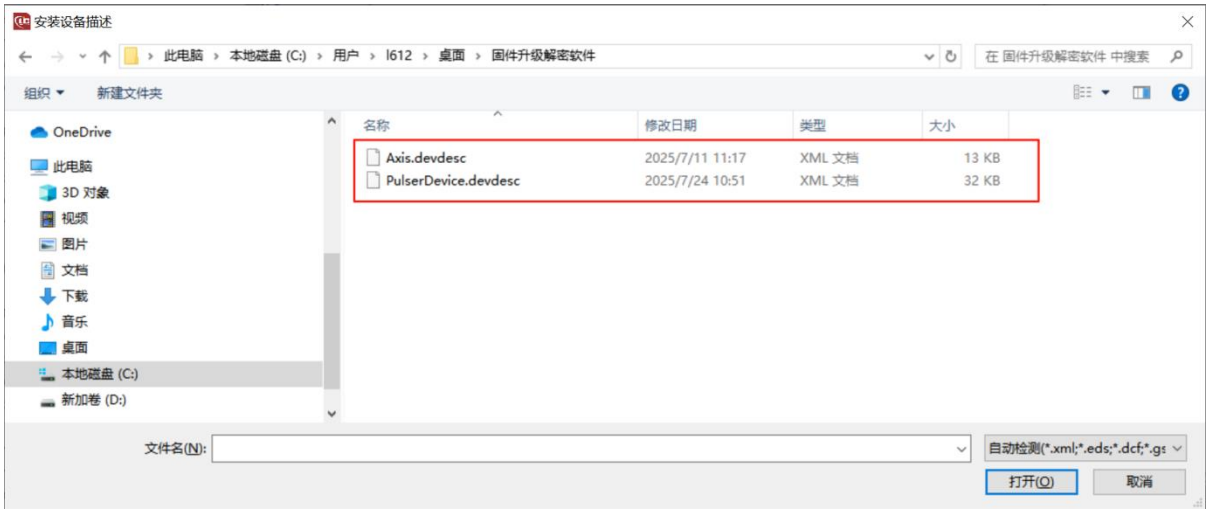
新建 IDE 工程，设备选择 Lingchen LC1500



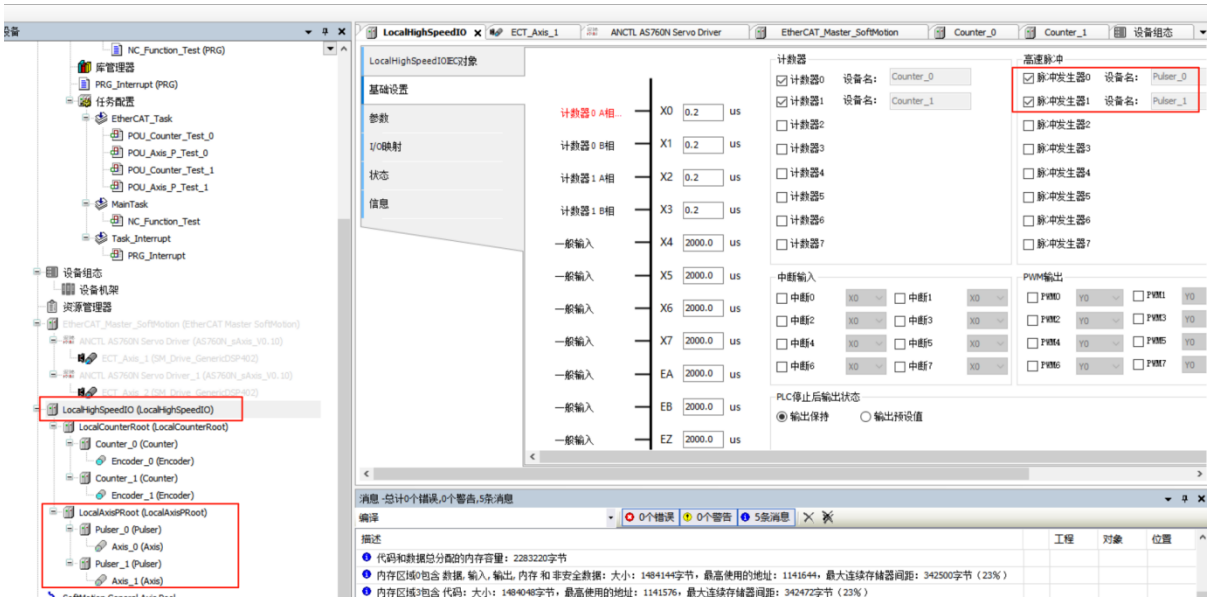
点击工具栏“工具”选项，选择“设备存储库”，然后点击“安装”



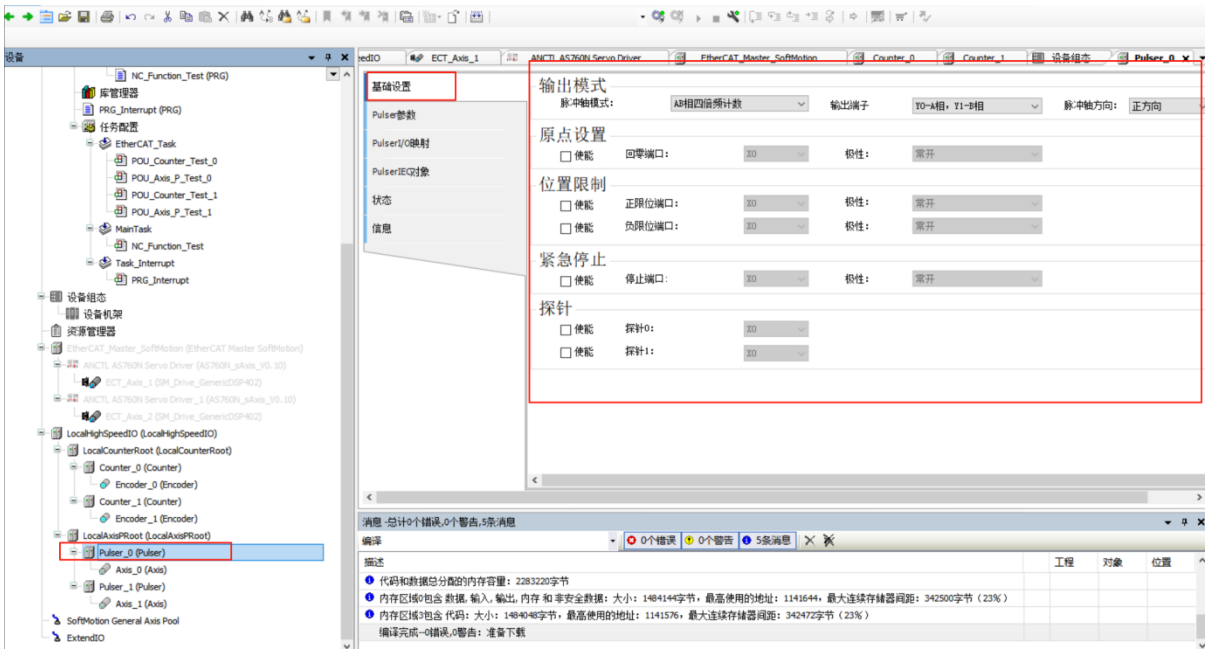
选择“Axis.devdesc”和“PulserDevice.devdesc”，点击打开，安装设备



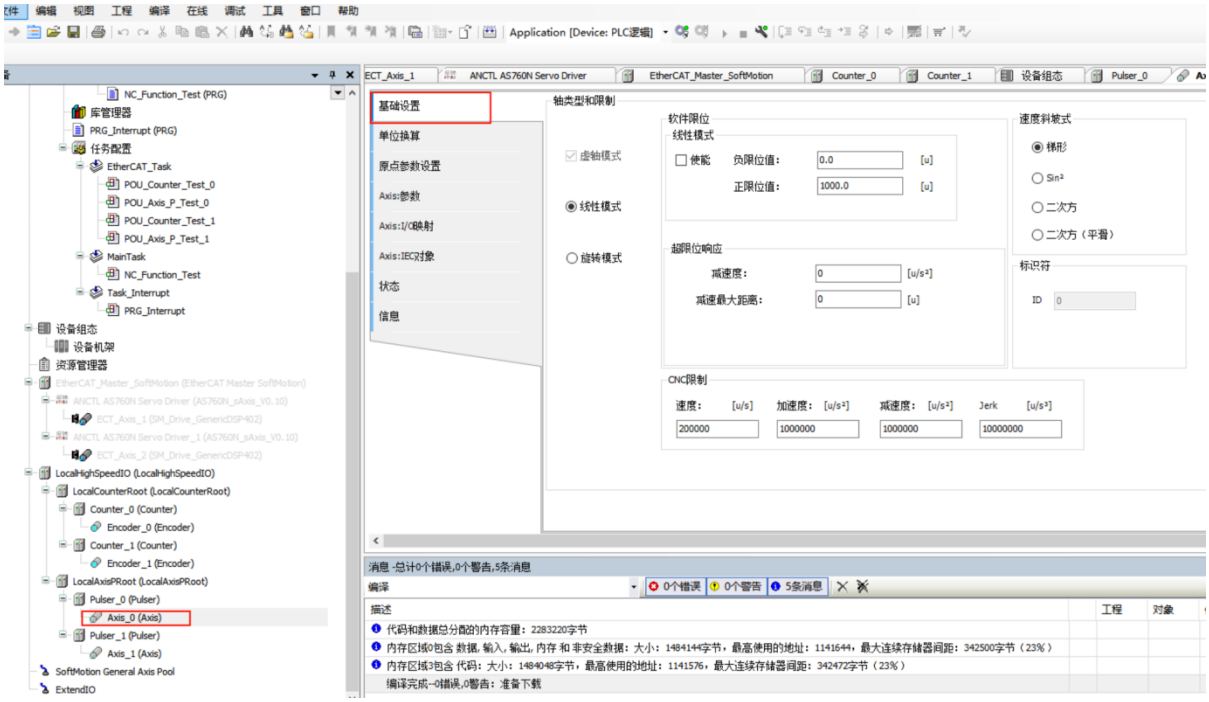
安装完成后，添加高速脉冲轴，点击“LocalHighSpeedIO”,在“基础设置”界面，在右侧选择“高速脉冲”，添加“脉冲发生器 0”、“脉冲发生器 1”，添加之后可以在左侧“LocalAxisPRoot”下看到刚刚添加的两个脉冲发生器：“Pulser_0”和“Pulser_1”；以及两个脉冲发生器下对应的脉冲轴“Axis_0”和“Axis_1”

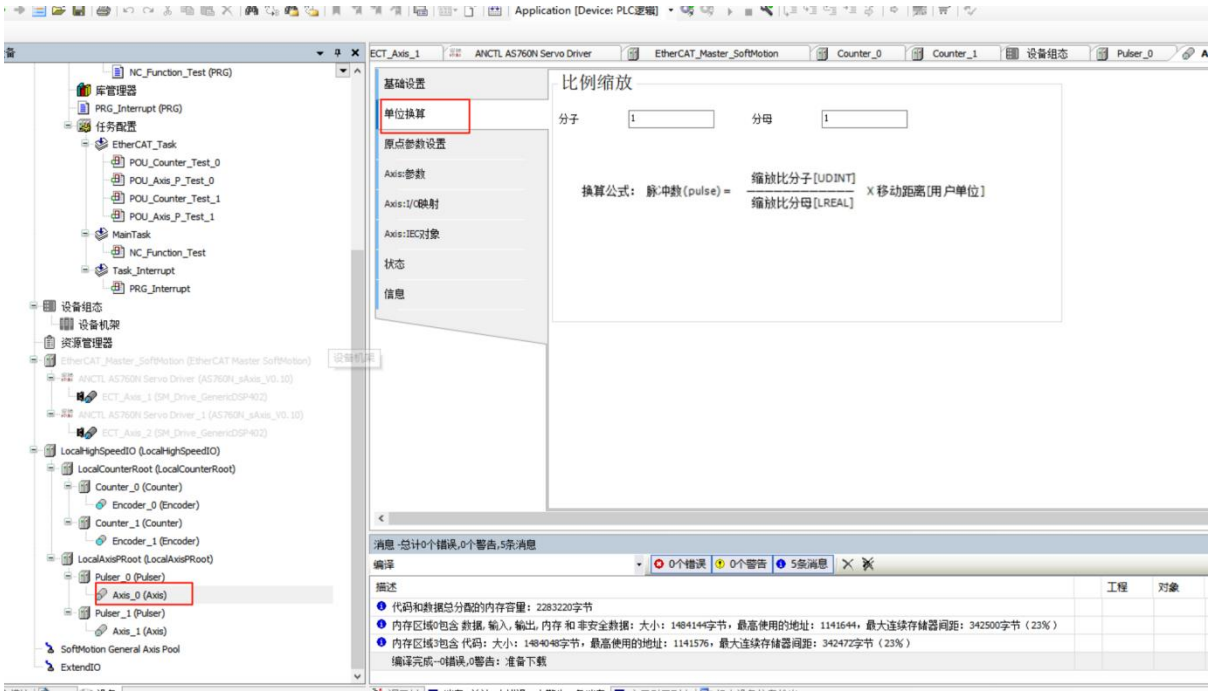


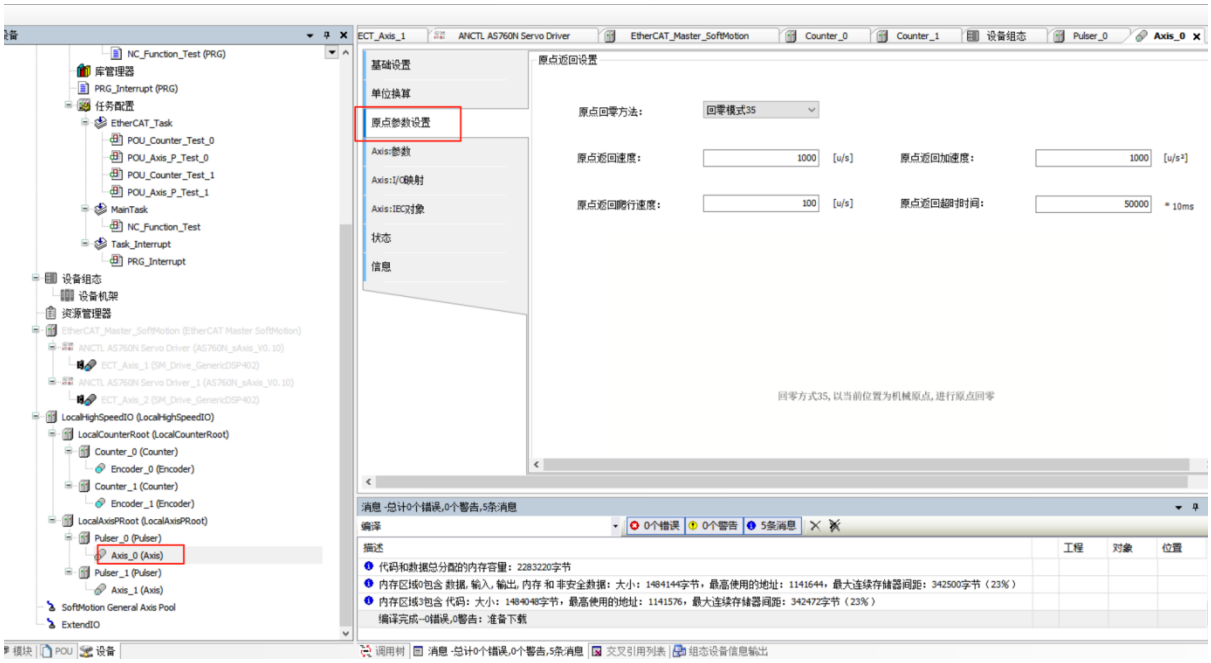
添加完成后，双击添加的“Pulser_0”，在“基础设置”界面对脉冲发生器参数进行设置，设置“脉冲轴模式”对应的硬件“输出端子”以及“脉冲轴方向”　　，也可以修改“输出模式”、“原点设置”、“位置限制”、“紧急停止”、“探针”等相关参数



设置完脉冲发生器参数后，双击“Axis_0”，对脉冲轴参数进行设置，在“基础设置”、“单位换算”及“原点参数设置”界面对脉冲轴的类型和限制、比例缩放和原点返回设置进行设置。




比例缩放
基础设置
单位换算
原点参数设置
分子
分母
换算公式：脉冲数(pulse) = 缩放比分子[UDINT] / 缩放比分母[LREAL] X 移动距离[用户单位]


原点返回设置
原点回零方法：
回零模式35
原点返回速度：
1000
[u/s]
原点返回加速度：
1000
[u/s²]
原点返回爬行速度：
100
[u/s]
原点返回超时时间：
50000
* 10ms
回零方式35, 以当前位置为机械原点, 进行原点回零

PULSER 参数说明。

注意：脉冲输出的最大频率是单边 200kHz。如果输出脉冲类型是 AB 相四倍频脉冲，则脉冲频率是 800kHz。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Axis_P_ID	轴 ID	USINT	[0,7]	0	轴 ID
Axis_P_PulseType	脉冲类型	BYTE	[0,5]	0	0:AB 相四倍频脉冲 1:脉冲+方向 正逻辑 2:脉冲+方向 负逻辑 3:CW+CCW 正逻辑 4:CW+CCW 负逻辑 5:单脉冲
Axis_P_PulseDirection	脉冲方向取反	BYTE	[0,1]	0	0：正方向 1：方向取反
Axis_P_Port	脉冲硬件端口	USINT	[0,15]	0	如果 Axis_P_PulseType=5 单脉冲模式， Axis_P_Port 范围是[0,15]。 其它脉冲模式： 0：Y0-Y1； 1：Y2-Y3； 2：Y4-Y5； 3：Y6-Y7； 4：Y10-Y11； 5：Y12-Y13； 6：Y14-Y15； 7：Y16-Y17；
Home_P_Enable	回原点使能	BOOL	[FALSE, TRUE]	FALSE	回原点使能
HomingDINum	原点 DI 端口	USINT	[0,10]	0	原点 DI 端口 0-7：X0-X7； 8：EA；9：EB；10：EZ
HomingPolarity	原点 DI 极性	BOOL	[FALSE, TRUE]	FALSE	原点 DI 极性 FALSE：常开 TRUE：常闭
PosLimitEnable	正向限位使能	BOOL	[FALSE, TRUE]	FALSE	正向限位使能
PosLimitDINum	正向限位 DI 端口	USINT	[0,10]	0	正向限位 DI 端口 0-7：X0-X7； 8：EA；9：EB；10：EZ
PosLimitPolarity	正向限位 DI 极性	BOOL	[FALSE, TRUE]	FALSE	正向限位 DI 极性 FALSE：常开 TRUE：常闭
NegLimitEnable	负向限位使能	BOOL	[FALSE, TRUE]	FALSE	负向限位使能
NegLimitDINum	负向限位 DI 端口	USINT	[0,10]	0	负向限位 DI 端口

					0-7：X0-X7； 8：EA；9：EB；10：EZ
NegLimitPolarity	负向限位 DI 极性	BOOL	[FALSE, TRUE]	FALSE	负向限位 DI 极性 FALSE：常开 TRUE：常闭
EmergencyStopEnable	急停使能	BOOL	[FALSE, TRUE]	FALSE	急停使能
EmergencyStopDINum	急停 DI 端口	USINT	[0,10]	0	急停 DI 端口 0-7：X0-X7； 8：EA；9：EB；10：EZ
EmergencyStopPolarity	急停 DI 极性	BOOL	[FALSE, TRUE]	FALSE	急停 DI 极性 FALSE：常开 TRUE：常闭
TouchProbe0Enable	探针 0 使能	BOOL	[FALSE, TRUE]	FALSE	探针 0 使能
TouchProbe0DINum	探针 0 DI 输入端口	USINT	[0,10]	0	探针 0DI 输入端口 0-7：X0-X7； 8：EA；9：EB；10：EZ
TouchProbe1Enable	探针 1 使能	BOOL	[FALSE, TRUE]	FALSE	探针 1 使能
TouchProbe1DINum	探针 1DI 输入端口	USINT	[0,10]	0	探针 1DI 输入端口 0-7：X0-X7； 8：EA；9：EB；10：EZ

AXIS 参数说明。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Axis_P_Enable	轴使能	BOOL	[FALSE, TRUE]	TRUE	轴使能
wDriveID	轴 ID	WORD	[0,7]	0	轴 ID，必须与父设备 Pulser 保持一致
bVirtual	是否是虚轴	BOOL	[FALSE, TRUE]	TRUE	FALSE：实轴 TRUE：虚轴
iMovementType	轴运动类型	INT	[0,1]	1	Movement type: 0: modulo, 1: finite
fPositionPeriod	模数值	LREAL	遵从数据类型	360.0	模数值，单位：unit
eRampType	速率斜坡类型	INT	[0,3]	0	The velocity ramp used for trajectories.
fSWMaxVelocity	最大速度	LREAL	遵从数据类型	200000.0	最大速度，单位：unit/s
fSWMaxAcceleration	最大加速度	LREAL	遵从数据类型	1000000.0	最大加速度，单位：unit/s²
fSWMaxDeceleration	最大减速度	LREAL	遵从数据类型	1000000.0	最大减速度，单位：unit/s²
fSWMaxJerk	最大加加速度	LREAL	遵从数据类型	10000000.0	最大加加速度，单位：unit/s³
fRampJerk	Sin²最大加加速度	LREAL	遵从数据类型	0.0	Jerk used for bringing acceleration to 0 when sin² ramp is interrupted.，单位：unit/s³
fSWLimitPositive	正向软件限位值	LREAL	遵从数据类型	1000.0	Software limit in positive direction，单位：unit
fSWLimitNegative	负向软件限位值	LREAL	遵从数据类型	0.0	Software limit in negative direction，单位：unit
fSWLimitDeceleration	软件限位减速度	LREAL	遵从数据类型	0.0	Deceleration for stop on software error，单

					位：unit/s²
bSWLimitEnable	软件限位使能	BOOL	[FALSE, TRUE]	FALSE	Activate/deactivate software limits
fSWErrorMaxDistance	软件错误停止最大行程	LREAL	遵从数据类型	0.0	Maximum distance that may be travelled for ramping down after a software error has been detected，单位：unit
numerator	齿轮比分子	UDINT	(0, 2147483647)	1	齿轮比分子 脉冲数 pluse=numerator/denominator * 移动距离(用户单位 unit)
denominator	齿轮比分母	LREAL	>0	1.0	齿轮比分母 脉冲数 pluse=numerator/denominator * 移动距离(用户单位 unit)
HomingMethods	回原方法	BYTE	[17,35]	35	回原方法其中 31-34 悬空保留
HomingVel	回原点速度	LREAL	遵从数据类型	1000	回原点第一段速，单位：unit/s
HomingCrawlVel	回原点爬行速度	LREAL	遵从数据类型	100	回原点第二段速，单位：unit/s
PositionMethods	回原点后，位置设定模式	BOOL	[FALSE, TRUE]	FALSE	FALSE：绝对回零 TRUE：相对回零
HomingAcc	回原点加速度	LREAL	遵从数据类型	1000.0	回原点最大加速度，单位：unit/s²
HomingTimeLimit	回原点超时时间	UDINT	[1,100000000]	50000	回原点超时时间，单位：10ms

轴错误 ID（MC_ERROR）说明

错误代码	枚举值	说明
000	MC_NO_ERROR	无错误
001	MC_ERROR_POS_DECPOINT_OVERLOW	减速无效：位置模式下，重新定位时，减速长度大于实际距离
002	MC_ERROR_VEL_DECPOINT_OVERLOW	减速无效：速度模式下，切换为定位时，减速长度大于实际距离
005	MC_ERROR_UP_SOFTWARE_LIMIT	当前位置超出软件行程限制 +
006	MC_ERROR_DOWN_SOFTWARE_LIMIT	当前位置超出软件行程限制 -
020	MC_ERROR_COMMUNICATION	底层通信错误
021	MC_ERROR_NOT_POWER	MC_Power 没使能
022	MC_ERROR_NOT_READY_FOR_MOTION	轴没准备好，不能运行
023	MC_ERROR_DEST_POS_OVER_SOFT_UP_LIMIT	目标位置超出软件上限
024	MC_ERROR_DEST_POS_OVER_SOFT_DOWN_LIMIT	目标位置超出软件下限
025	MC_ERROR_WASNT_STANDSTILL	当前状态不是 STANDSTILL
026	MC_ERROR_WASNT_DISABLED	当前状态不是 DISABLE
027	MC_ERROR_IN_ERRORSTOP	当前状态是 ERRORSTOP
028	MC_ERROR_NOT_CONFIG	图形化界面未配置 DI 或者 DO
050	MC_ERROR_AXIS_NUM_INVALID	轴号无效，有效范围是 [0,3]
051	MC_ERROR_VEL_SET_OUT_OF_RANGE	速度设置超出允许范围
052	MC_ERROR_ACC_SET_OUT_OF_RANGE	加速度超出允许范围
053	MC_ERROR_DEC_SET_OUT_OF_RANGE	减速度超出允许范围

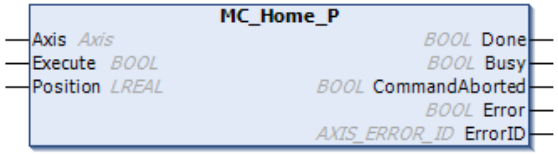
054	MC_ERROR_JERK_SET_OUT_OF_RANGE	加速度变化率 Jerk 设定超出允许范围
055	MC_ERROR_FBD_MOVEMODE_INVALIAD	运动模式无效
056	MC_ERROR_INVLALID_VELOCITY_MODE	速度模式无效
057	MC_ERROR_INVLALID_POSTION_MODE	位置模式无效
058	MC_ERROR_AXIS_WRITEPARAMETER_UNVALIAD	MC_WriteParameter_P 参数无效
059	MC_ERROR_AXIS_READPARAMETER_UNVALIAD	MC_ReadParameter_P 参数无效
060	MC_ERROR_HOME_MODE_UNVALIAD	回原模式无效
061	MC_ERROR_DEVICE_NOT_IN_USE	轴设备未使用
062	MC_ERROR_FB_CANNOT_INTERRUPT	功能块无法打断
063	MC_ERROR_ENABLE_UNIQUE_INSTANTIATION	enable 使能的指令只能有一个实例。
064	MC_ERROR_DI_NUM_INVALID	DI 配置不合理
065	MC_ERROR_PULSE_TYPE_INVALID	脉冲类型无效
066	MC_ERROR_HOME_OVER_TIME	回零超时

指令列表

序号	指令名称	FB/FC	功能
1.4.1	MC_Home_P	FB	高速轴回零
1.4.2	MC_TouchProbe	FB	高速轴探针

1.4.1 MC_Home_P

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
MC_Home_P	高速轴回零	FB		MC_Home_P(Axis:= , Execute:= , Position:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Axis	-	-	高速轴编号 [0..7]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Execute	使能	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：使能功能块。
Position	回零位置	DINT	LREAL	0	回零位置，单位：unit

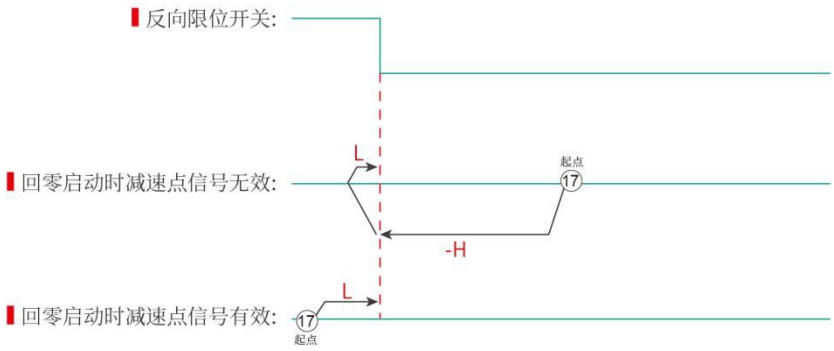
输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Done	完成标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块执行完成。
Busy	执行标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块正在执行。
CommandAborted	终止标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块被终止。
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码

【功能说明】

本功能块支持标准回零方式 17-30&35。配合硬件输入正负向限位和原点使用，需注意正负限位和原点的极性。以下回零方式的情形解释均以正负向限位和原点的极性为**常开**的情形下进行解释说明。注意甄别和转换。

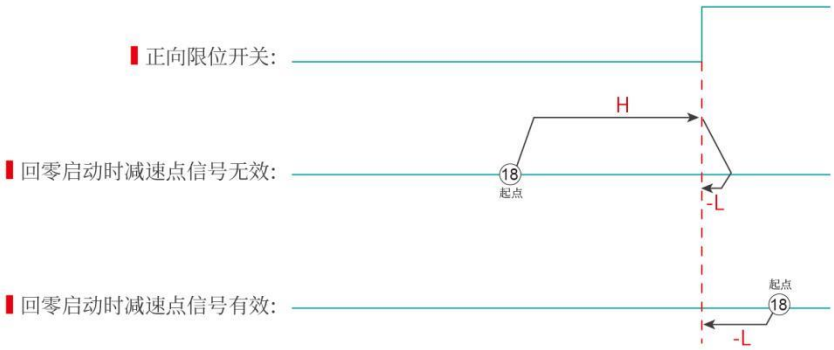
回零方式 17：负向运转反向极限开关下降沿



情形一：当执行回零时，若负限位端口状态处于低电平时，那么轴开始以第一段速度反向运动，当遇负限位端口处于高电平时，轴运动方向改变且以第二段速度开始运动；在负限位端口状态重新处于低电平时的位置就是原点位置。

情形二：当执行回零时，若负限位端口处于高电平时，那么轴直接会以第二段速度开始正向运动，在负限位端口状态处于低电平时的位置就是原点位置。

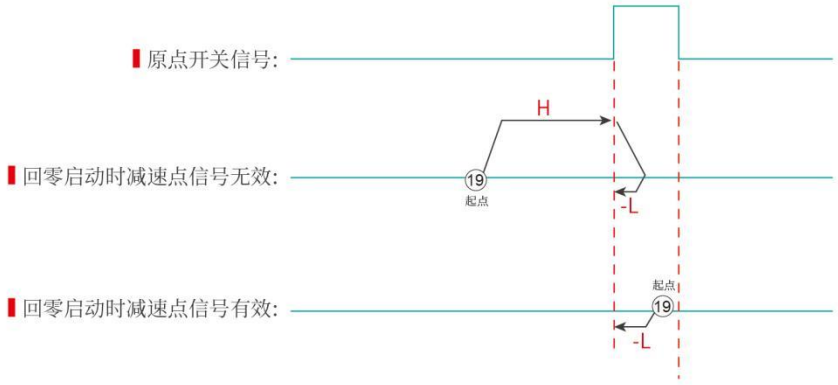
回零方式 18：正向运转正向极限开关下降沿



情形一：当执行回零时，若正限位端口状态处于低电平时，那么轴开始以第一段速度正向运动，当遇正限位端口处于高电平时，轴运动方向改变且以第二段速度开始运动；在正限位端口状态处于低电平时的位置就是原点位置。

情形二：当执行回零时，若正限位端口处于高电平时，那么轴直接会以第二段速度开始反向运动，在正限位端口状态处于低电平时的位置就是原点位置。

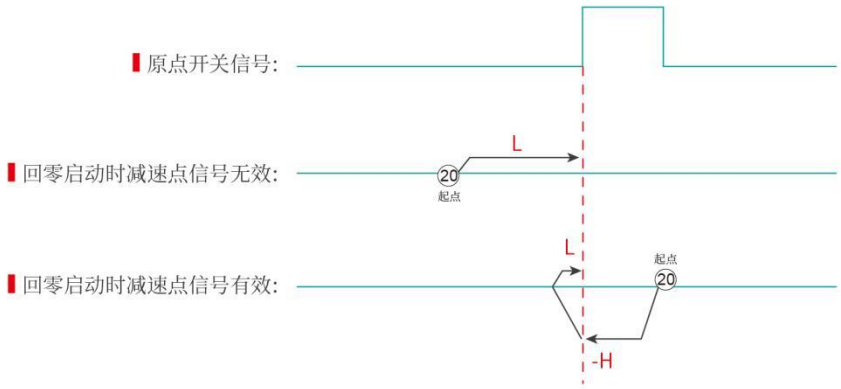
回零方式 19：取决于原点开关的原点回零（正向回零下降沿）



情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速正向运动，当遇到原点开关处于高位时，运动方向改变且以第二段速开始反向运动，当遇到原点开关处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速开始反向运动，当遇到原点开关处于低位时的位置就是原点位置。

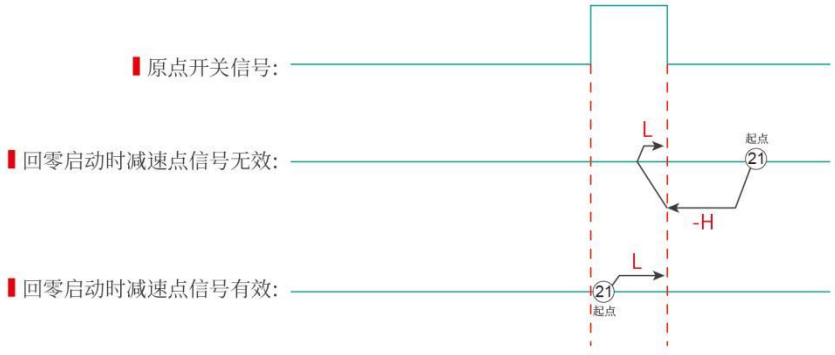
回零方式 20：取决于原点开关的原点回零（正向回零上升沿）



情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第二段速度正向运动，当遇到原点开关处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第一段速度开始反向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速度开始运动，当遇到原点开关处于高位时的位置就是原点位置。

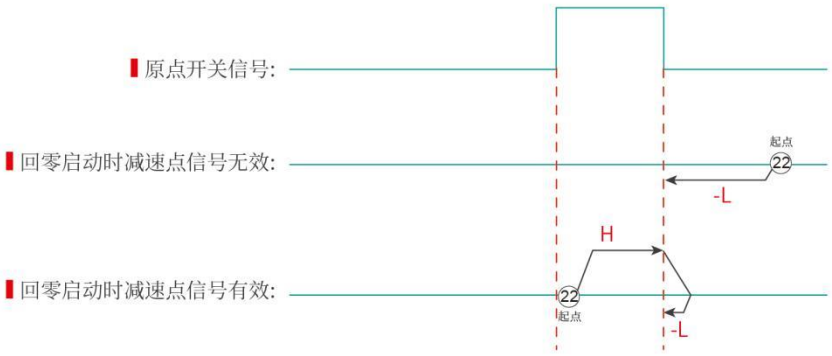
回零方式 21：取决于原点开关的原点回零（负向回零下降沿）



情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度反向运动，当遇到原点开关处于高位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于低位时的位置就是原点位置。

情形二： 用户触发执行回零时， 若原点开关状态处于高位， 那么轴直接以第二段速度开始正向运动， 当遇到原点开关处于低位时的位置就是原点位置。

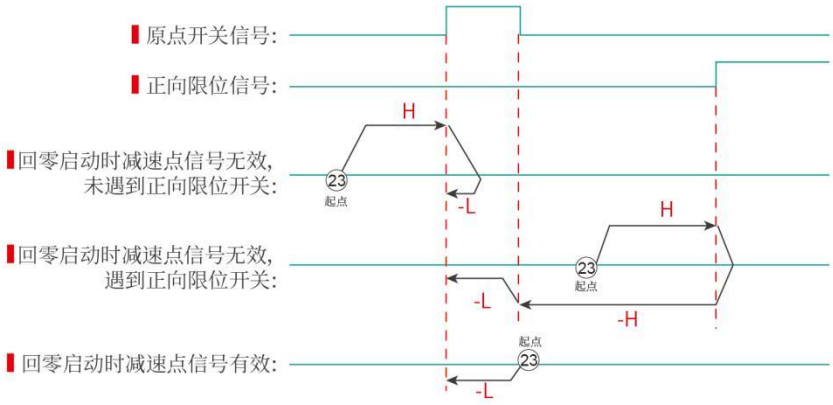
回零方式 22：取决于原点开关的原点回零（负向回零上升沿）



情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第二段速度反向运动，当遇到原点开关处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第一段速度开始正向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速开始运动，当遇到原点开关处于高位时的位置就是原点位置。

回零方式 23：正向运动找零点和极限

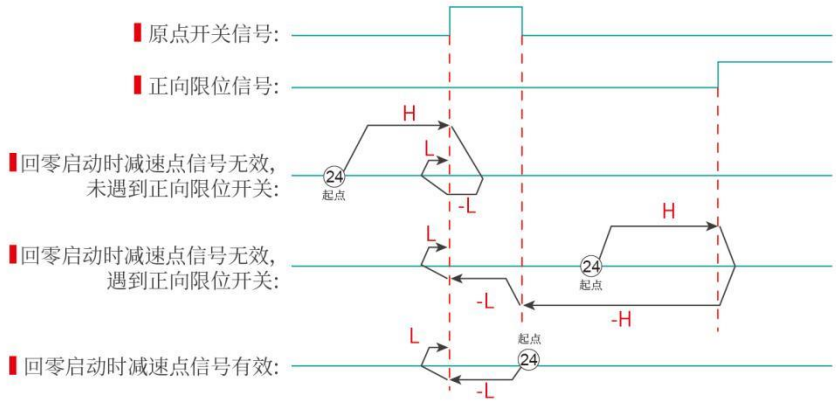


情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度正向运动，当遇到原点开关处于高位时，运动方向改变且以第二段速度开始运动，在原点开关状态处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方向改变且以第一段速度开始反向运动，当遇到原点开关处于高位时，以第二段速度开始运动，在原点开关状态处于低位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速度开始反向运动，在原点开关状态处于低位时的位置就是原点位置。

回零方式 24：正向运动找零点和极限

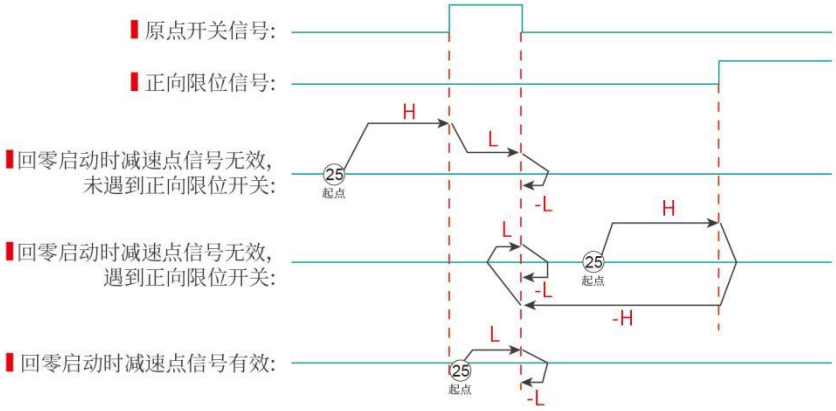


情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度正向运动，当遇到原点开关处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方向改变且以第一段速度开始反向运动，当遇到原点开关处于高位时，方向不变改为第二段速度继续运动，在原点开关状态处于低位时，运动方向改变，速度继续保持第二段速度，当遇到原点开关处于高位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速度开始反向运动，当遇到原点开关处于低位时，运动方向改变且继续以第二段速度开始运动，遇到原点开关处于高位时的位置就是原点位置。

回零方式 25：正向运动找零点和极限

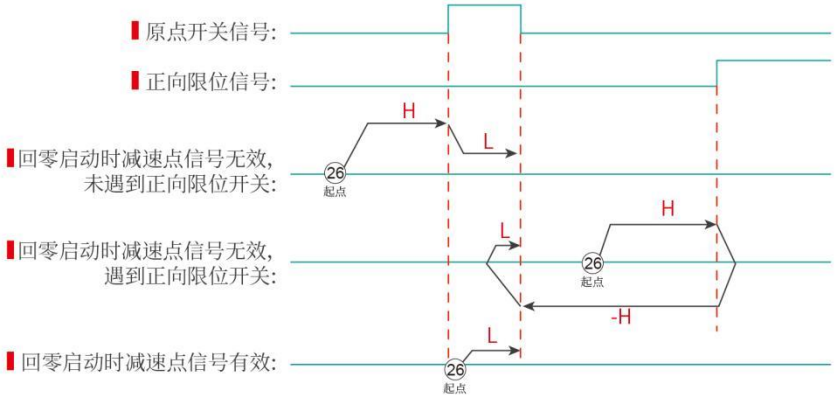


情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度正向运动，当遇到原点开关处于高位时，以第二段速度开始运动，当遇到原点开关处于低位时，运动方向改变且以第二段速度开始运动，当遇到原点开关处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方向改变且以第一段速度开始运动，当遇到原点开关处于高位时，以第二段速度开始反向运动，当原点开关再次处于低位时再次反向，再碰到原点开关处于高位的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于高位，那么轴开始以第二段速度正向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速度开始运动，当遇到原点开关处于高位时的位置就是原点位置。

回零方式 26：正向运动找零点和极限

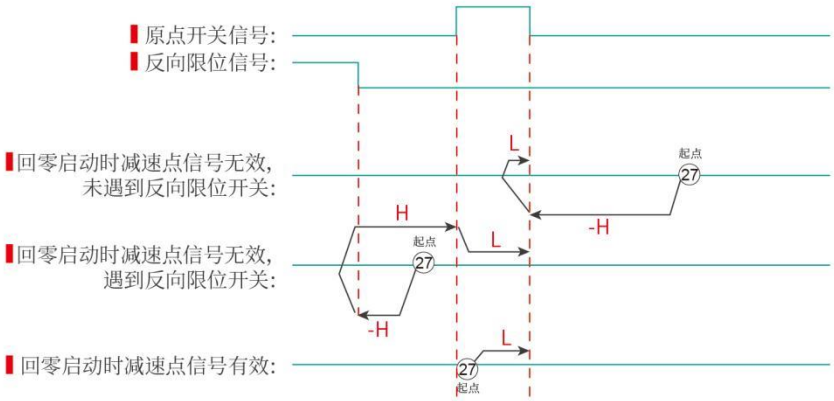


情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度正向运动，当遇到原点开关处于高位时，以第二段速度开始运动，当遇到原点开关处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度正向运动，当原点开关处于低位且遇到正向运转极限开关处于高位时，运动方向改变且以第一段速度开始反向运动，当遇到原点开关处于高位时，运动方向再次改变且以第二段速度开始运动，当原点开关处于低位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于高位，那么轴开始以第二段速度正向运动，当遇到原点开关处于低位时的位置就是原点位置。

回零方式 27：负向运动找零点和极限

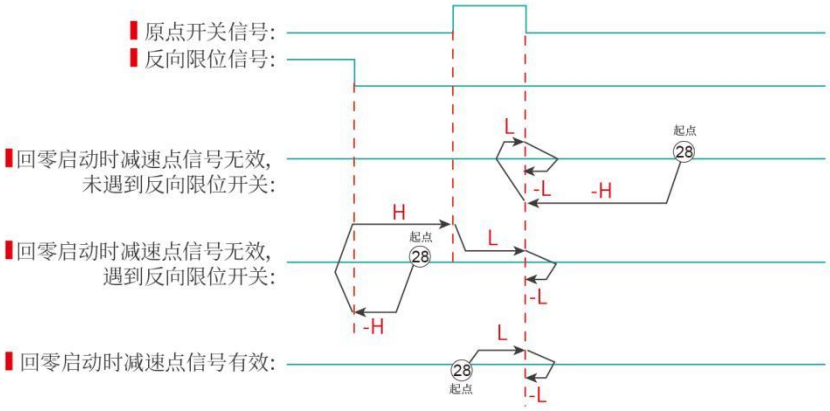


情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度反向运动，当遇到原点开关处于高位时，运动方向改变且以第二段速度开始运动，在原点开关状态处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方向改变且以第一段速度开始正向运动，当遇到原点开关处于高位时，以第二段速度开始运动，在原点开关状态处于低位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速度开始正向运动，在原点开关状态处于低位时的位置就是原点位置。

回零方式 28：负向运动找零点和极限

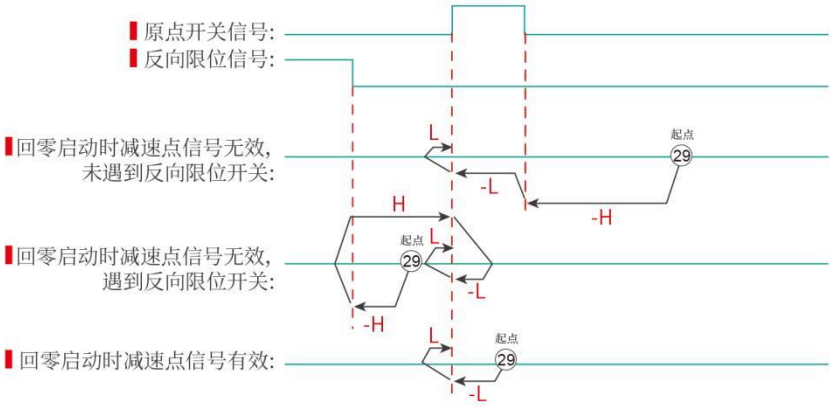


情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度反向运动，当遇到原点开关处于高位时，方向改变并以第二段速度开始运动，当原点开关状态处于低位，再次反向，继续以第二段速度运动，直到原点开关状态再次处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方向改变且以第一段速度开始运动，当遇到原点开关处于高位时，方向不变继续以第二段速度运动，在原点开关状态处于低位时，运动方向改变且以第二段速度开始运动，当遇到原点开关处于高位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速度开始正向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速度开始运动，当遇到原点开关处于高位时的位置就是原点位置。

回零方式 29：负向运动找零点和极限

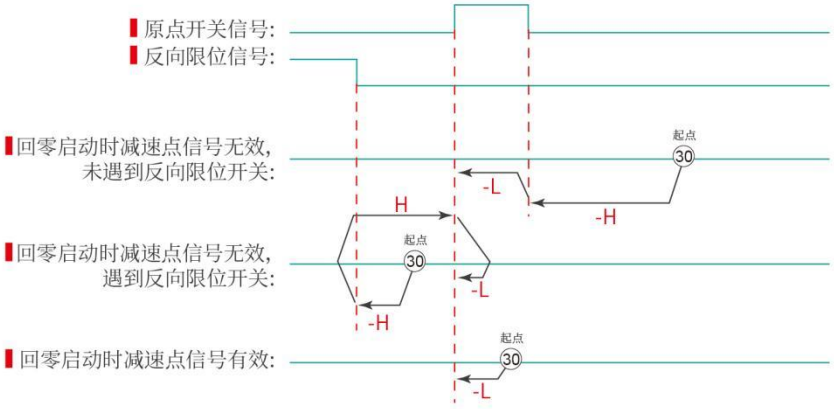


情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度反向运动，当遇到原点开关处于高位时，以第二段速度开始运动，当遇到原点开关处于低位时，运动方向改变且以第二段速度开始运动，当遇到原点开关处于高位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方向改变且以第一段速度开始运动，当遇到原点开关处于高位时，运动方向改变且以第二段速度开始运动，当原点开关状态处于低位，保持第二段速度但是反向运动，再次遇到原点开关状态处于高位时的位置就是原点位置。

情形三：用户触发执行回零时，若原点开关状态处于高位，那么轴直接以第二段速度反向运动，当遇到原点开关处于低位时，运动方向改变且以第二段速度开始运动，当遇到原点开关处于低位时的位置就是原点位置。

回零方式 30：负向运动找零点和极限



情形一：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度反向运动，当遇到原点开关处于高位时，以第二段速度开始运动，当遇到原点开关处于低位时的位置就是原点位置。

情形二：用户触发执行回零时，若原点开关状态处于低位，那么轴开始以第一段速度反向运动，当原点开关处于低位且遇到反向运转极限开关处于高位时，运动方向改变且以第一段速度开始正向运动，当遇到原点开关处于高位时，运动方向再次改变且以第二段速度开始运动，当遇到原点开关处于低位时的位置就是原点位置。

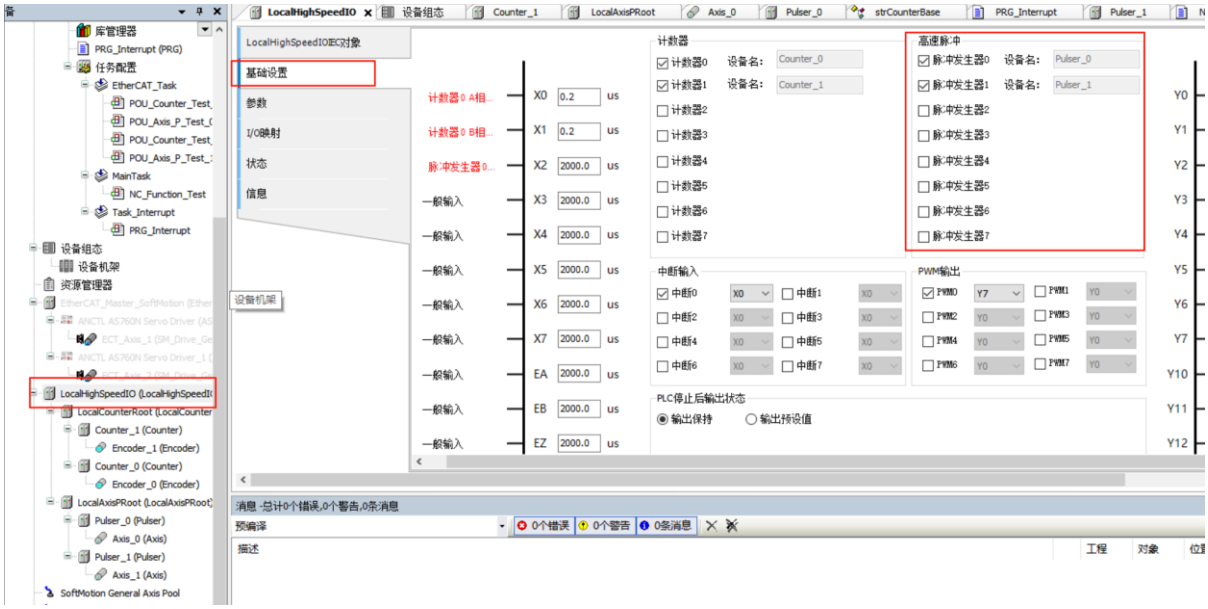
情形三：用户触发执行回零时，若原点开关状态处于高位，那么轴开始以第二段速度反向运动，当遇到原点开关处于低位时的位置就是原点位置。

回零方式 35：轴的当前位置为原点

在回零方式 35 下执行回零时，轴不运动，轴的当前位置被认为是原点回零的位置。

【图形化操作说明】

双击“LocalHighSpeedIO”，在“基础设置”中添加高速脉冲功能的“脉冲发生器”，高速脉冲轴会跟随“脉冲发生器”自动添加



添加“脉冲发生器”之后，在左侧“LocalAxisPRoot”下可以看到所添加的高速脉冲轴。双击高速脉冲轴“Axis_0”，在“原点参数设置”中对“原点回零方法”、“原点返回速度”、“原点返回加速度”、“原点返回爬行速度”、“原点返回超时时间”进行设置。

【功能块操作说明】

声明高速轴功能块“_MC_Home_P: MC_Home_P;”实例化

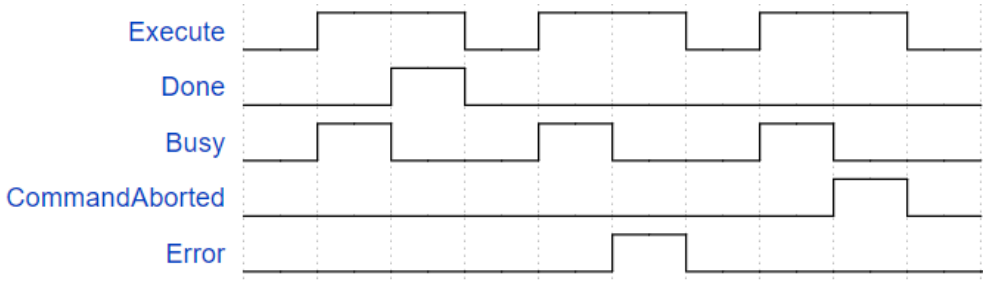
```
POU_Axis_P_Test_0 | PRG_Interrupt | Pulser | SoftMotion General Axis Pool | ANCTL_AS760N_Servo_Driver_1
13      _MC_MoveAdditive        : MC_MoveAdditive;          // 叠加运动
14      _MC_Jog                 : MC_Jog;                   // Jog点动
15      _MC_Home_P              : MC_Home_P;                // 脉冲轴回零
16      _MC_TouchProbe_P        : MC_TouchProbe_P;          // 脉冲轴探针锁存
17      _MC_SetPosition         : MC_SetPosition;           // 设定位置
18
19      (* 基础引脚结构体调用 *)
20      strMC_Power             : strAxisPBase;             // 结构体_使能
                                                            100 %
169
170  _MC_Home_P(
171      Axis:=Axis_0, //指定使用的轴名称
172      Execute:=strMC_Home_P.x_Execute, //上升沿触发，TRUE：使能功能块
173      Position:= , //回零位置，单位：unit
174      Done=>strMC_Home_P.x_Done, //TRUE：功能块执行完成
175      Busy=>strMC_Home_P.x_Busy, //TRUE：功能块正在执行
176      CommandAborted=>strMC_Home_P.x_CmdAborted, //TRUE：功能块被终止
177      Error=>strMC_Home_P.x_Error, //TRUE：功能块内部发生错误
178      ErrorID=>e_MC_Home_P_ErrorID);//错误码，具体参考CC_ERROR
```

功能块操作说明

```
_MC_Home_P(
    Axis:=Axis_0, //指定使用的轴名称
    Execute TRUE :=strMC_Home_P.x_Execute TRUE , //上升沿触发，TRUE：使能功能块
    Position:= , //回零位置，单位：unit
    Done TRUE =>strMC_Home_P.x_Done TRUE , //TRUE：功能块执行完成
    Busy FALSE =>strMC_Home_P.x_Busy FALSE , //TRUE：功能块正在执行
    CommandAborted FALSE =>strMC_Home_P.x_CmdAborted FALSE , //TRUE：功能块被终止
    Error FALSE =>strMC_Home_P.x_Error FALSE , //TRUE：功能块内部发生错误
    ErrorID MC_NO_ERRO =>e_MC_Home_P_ErrorID MC_NO_ERRO );//错误码，具体参考CC_ERROR
```

1. Axis 引脚绑定添加的轴名称。
2. Execute 引脚，默认 FALSE，置 TRUE 上升沿使能功能块。
3. Position 引脚，轴回零位置。
4. Done 引脚，完成标志，置 TRUE，功能块执行完成。
5. Busy 引脚，执行标志，置 TRUE，功能块正在执行。
6. CommandAborted 引脚，中止标志，功能块执行过程中被打断时置 TRUE。

【时序图】

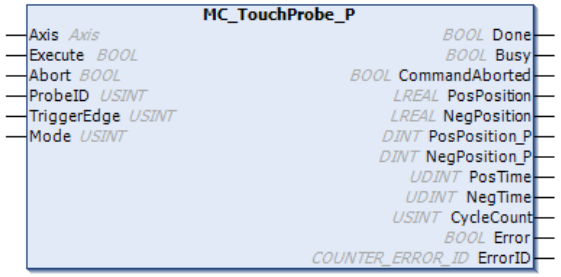


【错误说明】

详情查看 AXIS_P_REF 轴错误 ID 说明表格。

1.4.2 MC_TouchProbe_P

指令格式

指令	名称	FB/FC	LD 表现	ST 表现
MC_TouchProbe_P	高速轴探针	FB		MC_TouchProbe_P(Axis:= , Execute:= , Abort:= , ProbeID:= , TriggerEdge:= , Mode:= , Done=> , Busy=> , CommandAborted=> , PosPosition=> , NegPosition=> , PosPosition_P=> , NegPosition_P=> , PosTime=> , NegTime=> , CycleCount=> , Error=> , ErrorID=>);

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	说明
Axis	计数器	Axis	-	-	计数器编号 [0..3]。

输入变量

输入变量	名称	数据类型	有效范围	初始值	说明
Execute	使能	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：使能功能块。
Abort	终止	BOOL	[FALSE, TRUE]	FALSE	上升沿触发，TRUE：终止功能块。
ProbeID	探针 ID	USINT	[0,1]	0	探针 ID
TriggerEdge	触发边沿	USINT	[0,1]	0	0：上升沿 1：下降沿
Mode	触发模式	USINT	[0,1]	0	0：单次模式 1：连续模式

输出变量

输出变量	名称	数据类型	有效范围	初始值	说明
Done	完成标志	BOOL	[FALSE,	-	如为单次模式，执行一次输出后，Done，功能自

			TRUE]		动终止。连续模式，此信号无用。
Busy	执行标志	BOOL	[FALSE, TRUE]	-	FALSE：功能块未执行 TRUE：功能块执行中
CommandAborted	终止标志	BOOL	[FALSE, TRUE]	-	TRUE：功能块终止。
PosPosition	上升沿锁存位置	LREAL	-	-	上升沿锁存位置，单位：unit
NegPosition	下降沿锁存位置	LREAL	-	-	下降沿锁存位置，单位：unit
PosPosition_P	上升沿锁存位置	DINT	[-2147483648, 2147483647]	-	上升沿锁存位置，单位：pulse
NegPosition_P	下降沿锁存位置	DINT	[-2147483648, 2147483647]	-	下降沿锁存位置，单位：pulse
PosTime	上升沿锁存时刻	UDINT	[0, 4294967295]	-	上升沿锁存时刻，单位：us
NegTime	下降沿锁存时刻	UDINT	[0, 4294967295]	-	下降沿锁存时刻，单位：us
CycleCount	锁存次数	USINT	[0,31]	-	锁存次数
Error	错误标志	BOOL	[FALSE, TRUE]	-	TRUE: 功能块内部发生错误
ErrorID	错误 ID	UINT	-	-	错误码，具体参考 CC_ERROR

【功能说明】

本功能块主要实现计数器的探针锁存功能，根据设置的触发边沿，当外部信号触发时，读取计数器当前值与系统当前时间输出，同时记录锁存次数。

【注意事项】

1、本功能块每路探针可以设置单次模式或连续模式，当使用连续模式时，每次触发都会更新对应输出值。同时，连续模式下，Done 信号不会有输出，此信号无效。

2、CycleCount 锁存次数存在范围，当锁存次数达到有效范围边界并继续增加时，该输出项会清零重新计数。

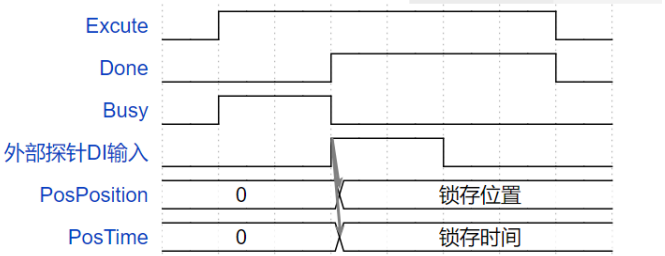
3、探针 ID 不支持在线修改。

4、每路计数器支持两种边沿触发模式，即上升沿锁存、下降沿锁存。

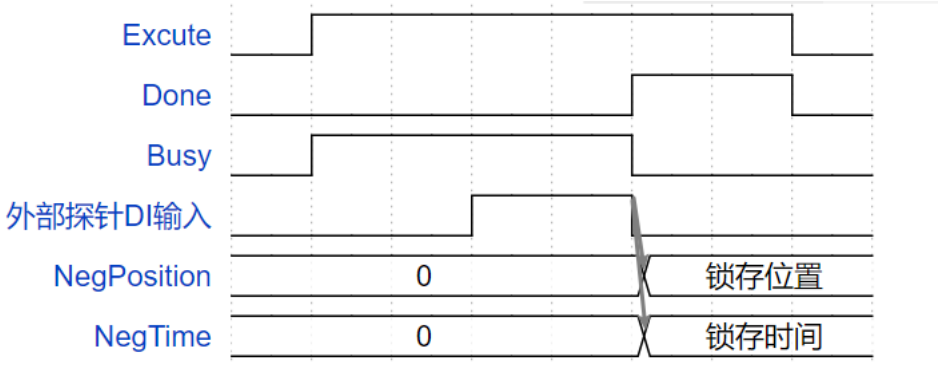
5、本功能块支持编码器 Z 信号锁存，需要使用编码器输入，即信号源设置为 EA-EB。

6、每路探针均同时支持时间、位置锁存。

单次触发模式（Mode=0），外部上升沿触发（TriggerEdge=0）,输出端口号为 Y00-Y07/Y10-Y17，指令时序图如下图所示：



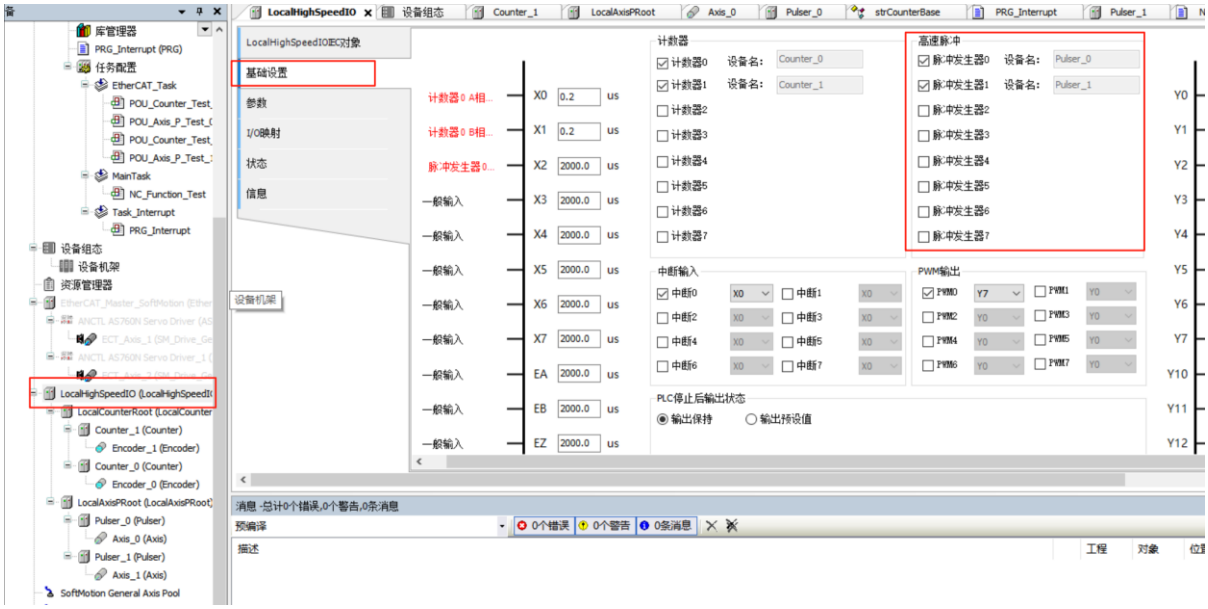
单次触发模式（Mode=0），外部下降沿触发（TriggerEdge=1）,输出端口号为 Y00-Y07/Y10-Y17，指令时序图如下图所示：



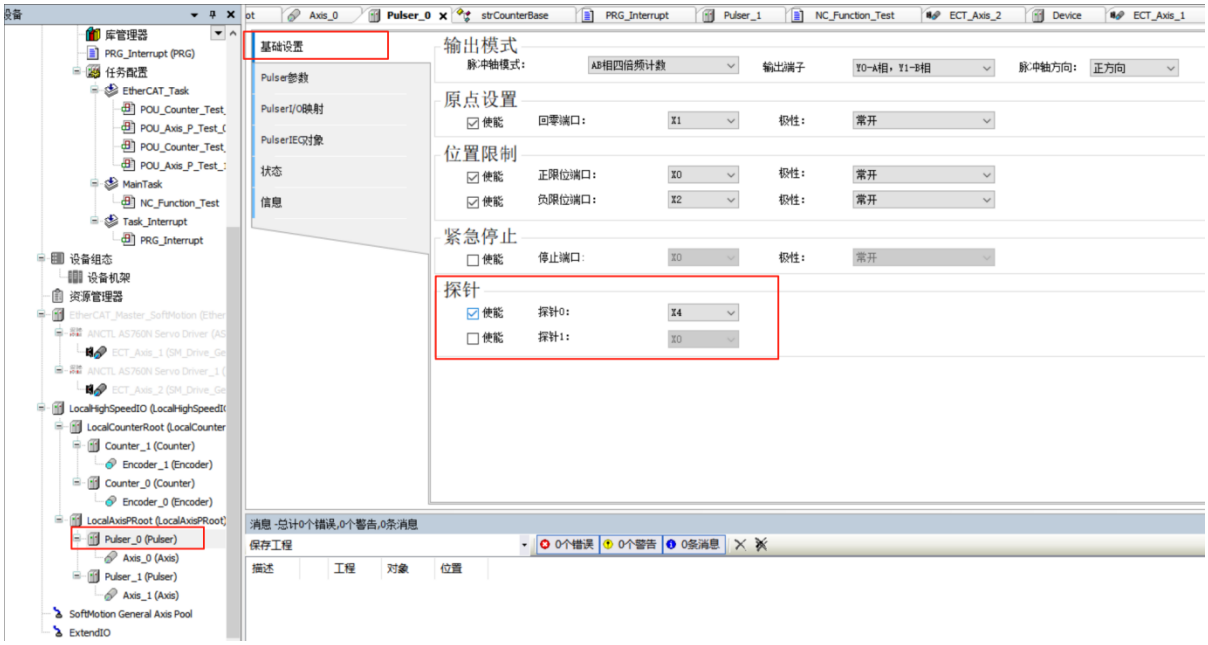
锁存计数+1

【功能块操作说明】

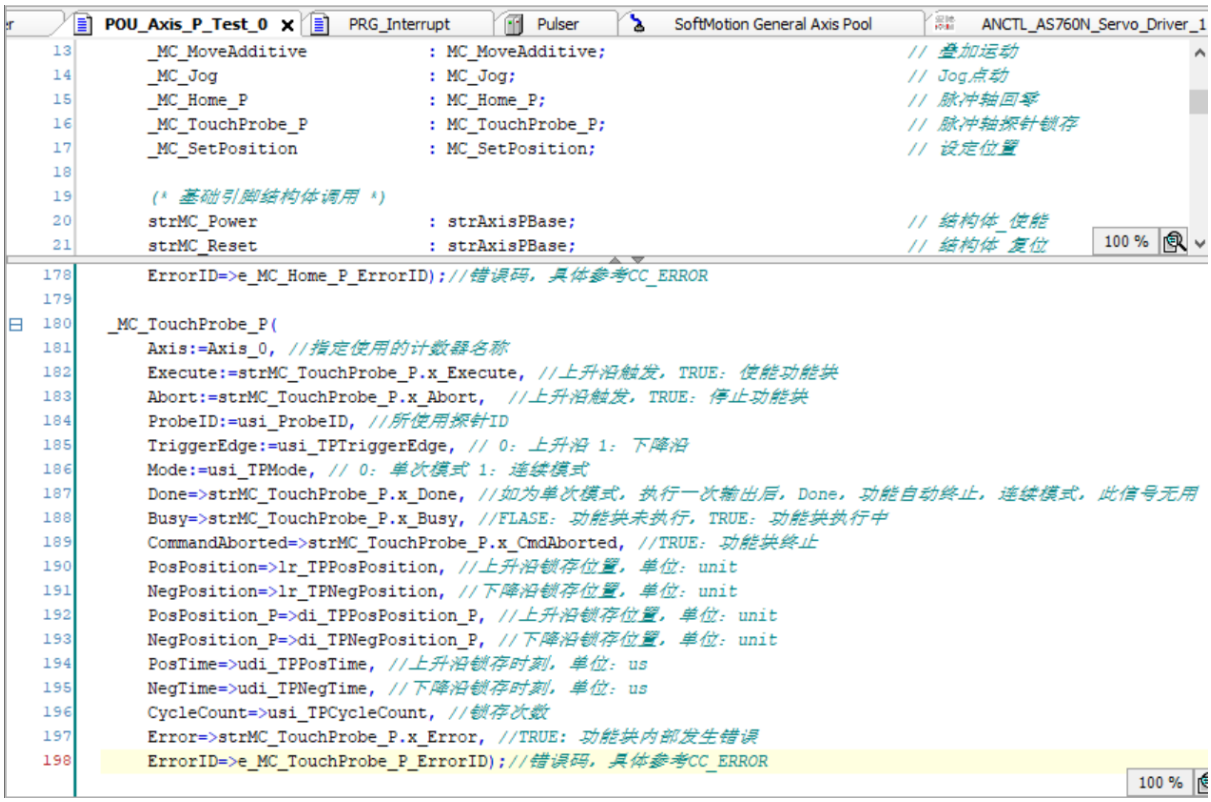
双击“LocalHighSpeedIO”，在“基础设置”中添加高速脉冲功能的“脉冲发生器”，高速脉冲轴会跟随“脉冲发生器”自动添加



双击“Pulser_0”设备，在“基础设置”界面使能“探针”，并选择相应的探针输入端口


基础设置
Pulser参数
Pulser I/O映射
Pulser IEC对象
状态
信息
输出模式
脉冲轴模式:
AB相四倍频计数
输出端子
Y0-A相，Y1-B相
脉冲轴方向:
正方向
原点设置
使能
回零端口:
X1
极性:
常开
位置限制
使能
正限位端口:
X0
极性:
常开
使能
负限位端口:
X2
极性:
常开
紧急停止
使能
停止端口:
X0
极性:
常开
探针
使能
探针0:
X4
使能
探针1:
X0

【功能块操作说明】


POU_Axis_P_Test_0
PRG_Interrupt
Pulser
SoftMotion General Axis Pool
ANCTL_AS760N_Servo_Driver_1
_MC_MoveAdditive : MC_MoveAdditive; // 叠加运动
_MC_Jog : MC_Jog; // Jog点动
_MC_Home_P : MC_Home_P; // 脉冲轴回零
_MC_TouchProbe_P : MC_TouchProbe_P; // 脉冲轴探针锁存
_MC_SetPosition : MC_SetPosition; // 设定位置
(* 基础引脚结构体调用 *)
strMC_Power : strAxisPBase; // 结构体_使能
strMC_Reset : strAxisPBase; // 结构体_复位
100 %
ErrorID=>e_MC_Home_P_ErrorID);//错误码，具体参考CC_ERROR
_MC_TouchProbe_P(
Axis:=Axis_0, //指定使用的计数器名称
Execute:=strMC_TouchProbe_P.x_Execute, //上升沿触发，TRUE：使能功能块
Abort:=strMC_TouchProbe_P.x_Abort, //上升沿触发，TRUE：停止功能块
ProbeID:=usi_ProbeID, //所使用探针ID
TriggerEdge:=usi_TPTriggerEdge, // 0：上升沿 1：下降沿
Mode:=usi_TPMode, // 0：单次模式 1：连续模式
Done=>strMC_TouchProbe_P.x_Done, //如为单次模式，执行一次输出后，Done，功能自动终止，连续模式，此信号无用
Busy=>strMC_TouchProbe_P.x_Busy, //FLASE：功能块未执行，TRUE：功能块执行中
CommandAborted=>strMC_TouchProbe_P.x_CmdAborted, //TRUE：功能块终止
PosPosition=>lr_TPPosPosition, //上升沿锁存位置，单位：unit
NegPosition=>lr_TPNegPosition, //下降沿锁存位置，单位：unit
PosPosition_P=>di_TPPosPosition_P, //上升沿锁存位置，单位：unit
NegPosition_P=>di_TPNegPosition_P, //下降沿锁存位置，单位：unit
PosTime=>udi_TPPosTime, //上升沿锁存时刻，单位：us
NegTime=>udi_TPNegTime, //下降沿锁存时刻，单位：us
CycleCount=>usi_TPCycleCount, //锁存次数
Error=>strMC_TouchProbe_P.x_Error, //TRUE：功能块内部发生错误
ErrorID=>e_MC_TouchProbe_P_ErrorID);//错误码，具体参考CC_ERROR
100 %

```
_MC_TouchProbe_P(
    Axis:=Axis_0,
    Execute TRUE :=strMC_TouchProbe_P.x_Execute TRUE ,
    Abort FALSE :=strMC_TouchProbe_P.x_Abort FALSE ,
    ProbeID 0 :=usi_ProbeID 0 ,
    TriggerEdge 0 :=usi_TPTriggerEdge 0 ,
    Mode 0 :=usi_TPMode 0 ,
    Done TRUE =>strMC_TouchProbe_P.x_Done TRUE ,
    Busy FALSE =>strMC_TouchProbe_P.x_Busy FALSE ,
    CommandAborted FALSE =>strMC_TouchProbe_P.x_CmdAborted FALSE ,
    PosPosition 2.75E+07 =>lr_TPPosPosition 2.75E+07 ,
    NegPosition 0 =>lr_TPNegPosition 0 ,
    PosPosition_P 29813713 =>di_TPPosPosition_P 29813713 ,
    NegPosition_P 0 =>di_TPNegPosition_P 0 ,
    PosTime 2458312358 =>udi_TPPosTime 2458312358 ,
    NegTime 0 =>udi_TPNegTime 0 ,
    CycleCount 1 =>usi_TPCycleCount 1 ,
    Error FALSE =>strMC_TouchProbe_P.x_Error FALSE ,
    ErrorID CC_NO_ERRO =>e_MC_TouchProbe_P_ErrorID CC_NO_ERRO );RETURN
```

1，“Axis”引脚绑定添加的计数器名称;计数器探针功能块能够实现在设定的信号触发时锁存脉冲计数值。

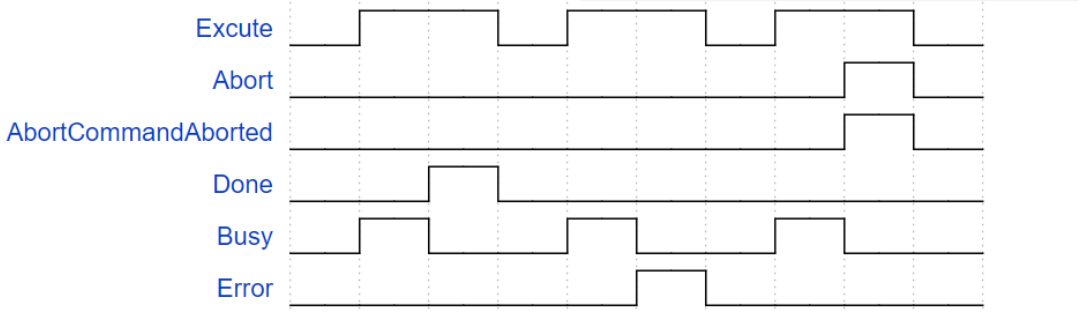
2，“ProbeID”引脚

3，“Mode”引脚可以设定单次模式或连续模式，0：单次模式，1：连续模式

4，“CycleCount”引脚根据触发次数进行计数。

【时序图】

因为在连续模式下，Done 信号不会有输出，因此这里只展示单次模式下的时序图。



【错误说明】

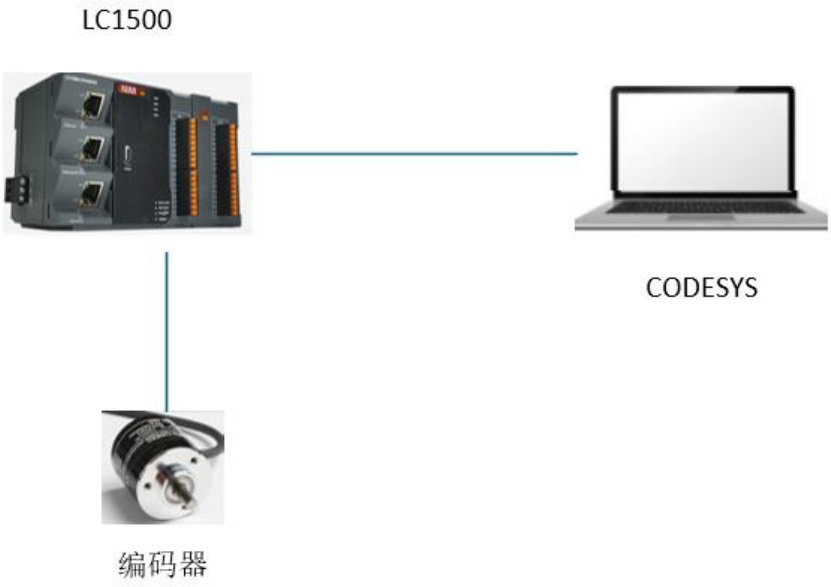
详情查看 MC_ERROR-脉冲输出错误 ID 说明表格。

2.1 高速脉冲计数应用功能案例

2.1.1 硬件选择

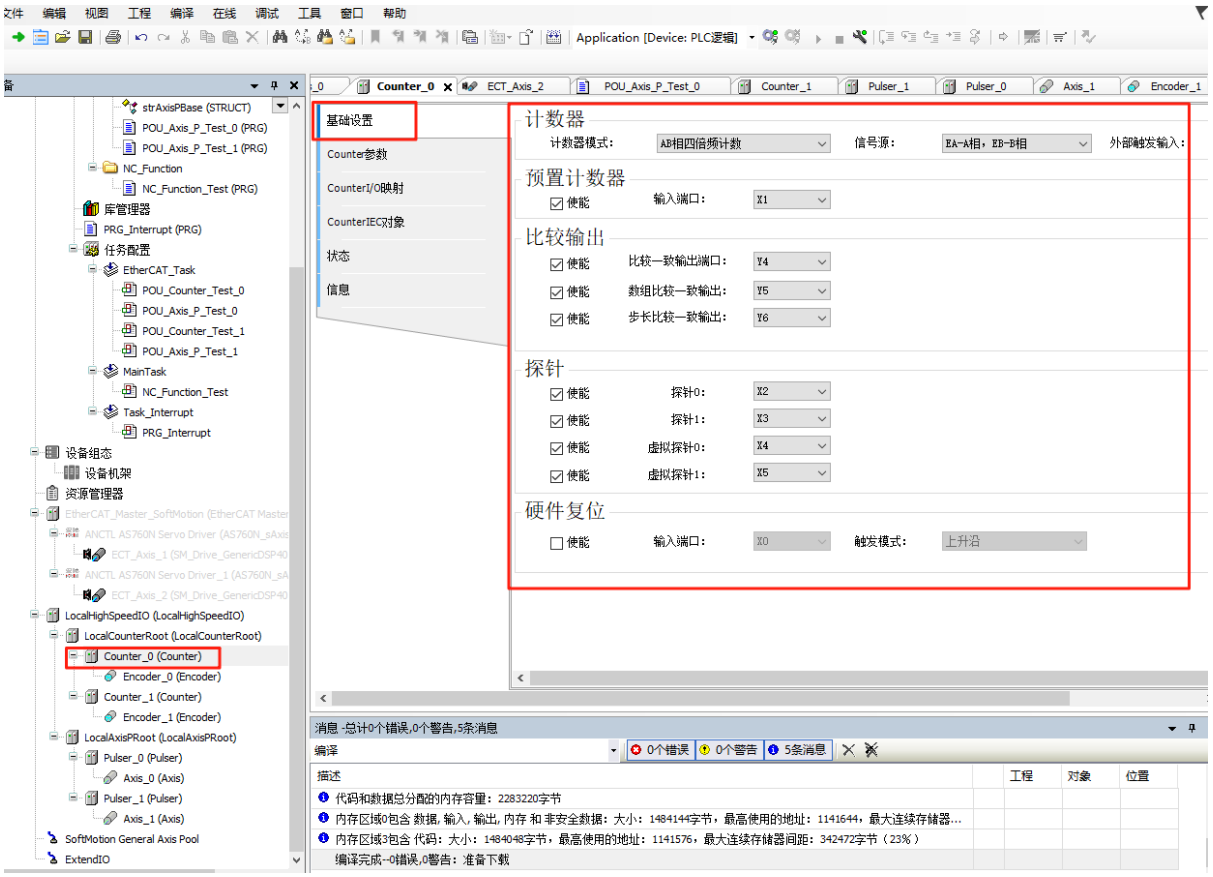
1）LC1500PLC
2）编码器
3）CODESYS

2.1.2 硬件接线图

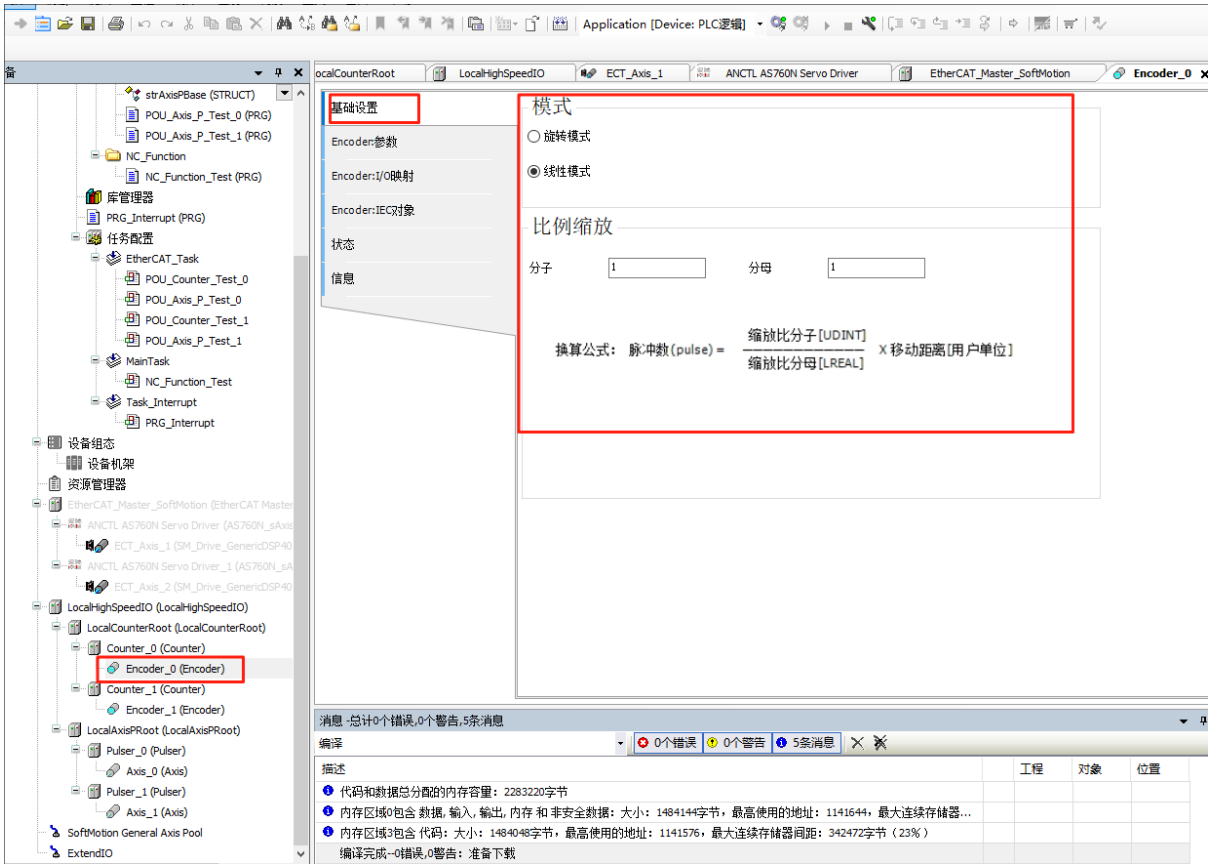


2.2 软件参数配置

打开 IDE 软件，新建“标准工程”；编写高速计数器样例，添加计数器设备“Counter_0”打开已建立好的高速脉冲计数测试程序样例，点击所添加的计数器设备“Counter_0”，在“基础设置”页面对计数器的参数进行配置：选择“计数器模式”为“AB 相四倍频计数”，选择“信号源”为“EA-A 相，EB-B 相”，设置“外部触发输入”端口为 X0；使能所要使用的“预置计数器”、“比较输出”、“探针”等，并设置相应的硬件端口，注意硬件端口不要重复使用。



点击计数器设备“Counter_0”下的轴“Encoder_0”，在“基础设置”界面选择“模式”与“比例缩放”

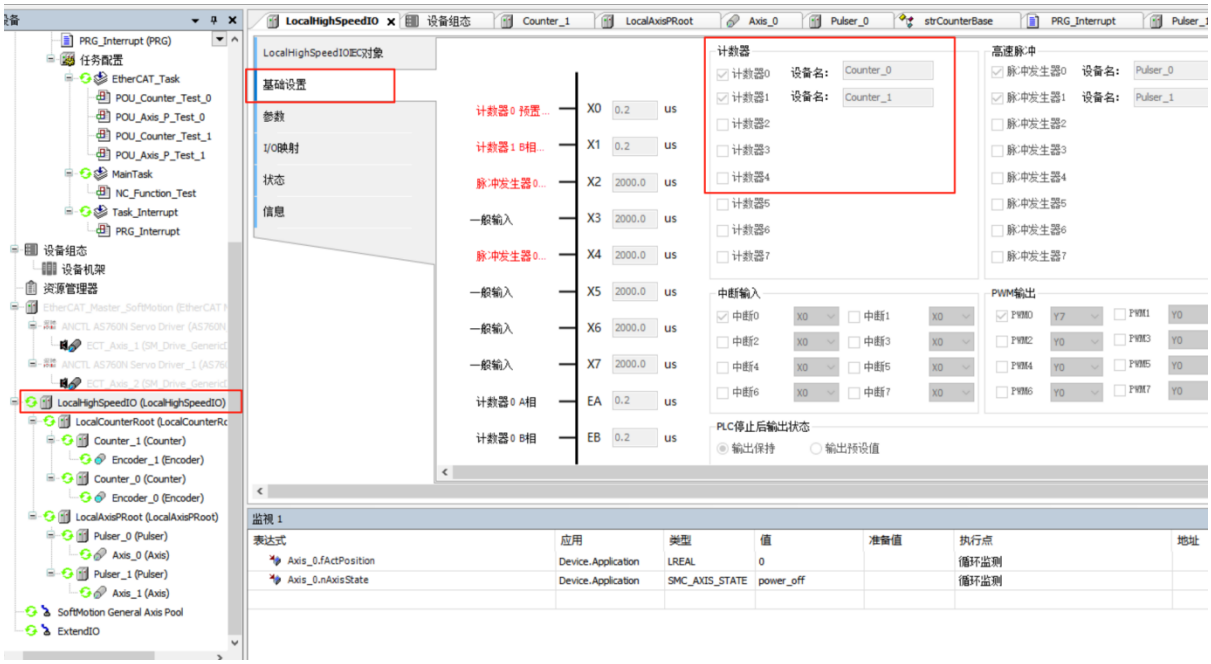


连接并登录 PLC，开始调用高速计数功能块，如调用“CC_Counter”功能块，使能功能块后，转动编码器，可以看到“CounterValue”计数值发生变化，可以正常使用。

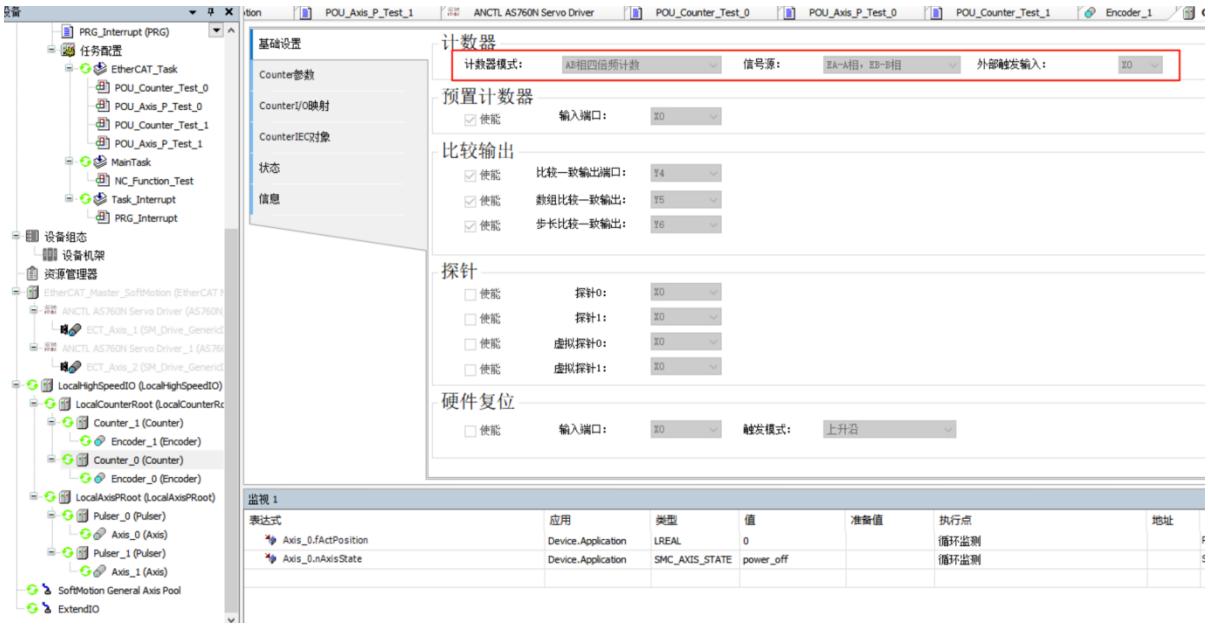
2.3 功能块应用实例

2.3.1 计数器计数实例

双击“LocalHighSpeedIO”，在“基础设置”添加计数器“Counter_0”



双击计数器“Counter_0”，在“基础设置”界面中选择计数器模式“AB 相四倍频计数”，信号源选择“EA-A 相，EB-B 相”



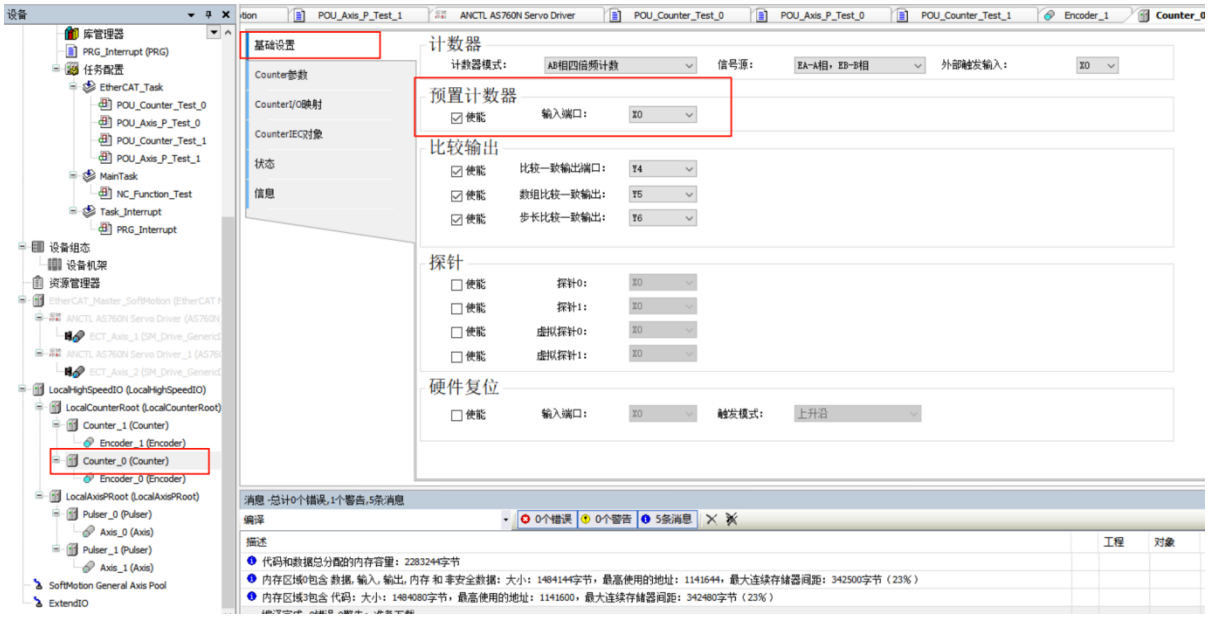
打开 POU，将 Enable、Start 引脚对应变量置为 TRUE，此时 Valid、Busy 引脚对应变量为 TRUE，说明计数器正在工作，转动 PLC 上连接的编码器，可以看到 CounterValue 引脚有数值变化，计数值

为“82”

```
strCounter.x_Enable(TRUE)    :=TRUE;
x_StartCounter(TRUE)    :=TRUE;
// 计数器
_CC_Counter(
    Axis:=Encoder_0, //指定使用的计数器名称
    Enable(TRUE):=strCounter.x_Enable(TRUE), //使能计数器，TRUE：初始化并生效功能块，FLASE：功能块不生效
    Start(TRUE):=x_StartCounter(TRUE), //软件启动信号（上升沿）
    TrigEdge(0):=usi_CounterTrigEdge(0), // 0：上升沿 1：下降沿
    Invert(FALSE):=x_Invert(FALSE), //计数值取反
    Compensation(FALSE):=x_Compensation(FALSE), // 补偿方式，False：用户手动补偿或不补偿 True：自动补偿
    Valid(TRUE)=>x_CounterValid(TRUE), //计数器计数标志，TRUE：有效，FLASE：不生效
    Busy(TRUE)=>strCounter.x_Busy(TRUE), //计数标志，FLASE：计数器未执行，TRUE：计数器已执行
    Direction(TRUE)=>x_CounterDirection(TRUE), //计数方向，FLASE：正方向（加法计数），TRUE：反方向（减法计数）
    CounterValue(82)=>lr_CounterValue(82), //计数器值，计数器当前计数值，单位：unit
    Velocity(0)=>lr_Velocity(0), //计数器速度，计数器当前速度，单位unit/s
    CounterValue_P(82)=>di_CounterValue_P(82), //计数器值，计数器当前计数值，单位：pulse
    Velocity_P(0)=>di_Velocity_P(0), //计数器速度，计数器当前速度，单位pulse/s
    ReceiveTime(2427665169)=>udi_CounterReceiveTime(2427665169), //读取计数值时刻，读取CounterValue的系统时刻
    Error(FALSE)=>strCounter.x_Error(FALSE), //错误标志，TRUE：功能块内部发生错误
    ErrorID(CC_NO_ERRO)=>strCounter.e_ErrorID(CC_NO_ERRO));//错误ID，错误码，具体参考CC_ERROR
```

2.3.2 计数器预置实例

双击添加的计数器“Counter_0”，在“基础设置”界面使能“预置计数器”



打开 POU，可以看到此时计数器的计数值 CounterValue 为 82

```
_CC_Counter(
    Axis:=Encoder_0, //指定使用的计数器名称
    Enable(TRUE):=strCounter.x_Enable(TRUE), //使能计数器，TRUE：初始化并生效功能块，FLASE：功能块不生效
    Start(TRUE):=x_StartCounter(TRUE), //软件启动信号（上升沿）
    TrigEdge(0):=usi_CounterTrigEdge(0), // 0：上升沿 1：下降沿
    Invert(FALSE):=x_Invert(FALSE), //计数值取反
    Compensation(FALSE):=x_Compensation(FALSE), // 补偿方式，False：用户手动补偿或不补偿 True：自动补偿
    Valid(TRUE)=>x_CounterValid(TRUE), //计数器计数标志，TRUE：有效，FLASE：不生效
    Busy(TRUE)=>strCounter.x_Busy(TRUE), //计数标志，FLASE：计数器未执行，TRUE：计数器已执行
    Direction(TRUE)=>x_CounterDirection(TRUE), //计数方向，FLASE：正方向（加法计数），TRUE：反方向（减法计数）
    CounterValue(82)=>lr_CounterValue(82), //计数器值，计数器当前计数值，单位：unit
    Velocity(0)=>lr_Velocity(0), //计数器速度，计数器当前速度，单位unit/s
    CounterValue_P(82)=>di_CounterValue_P(82), //计数器值，计数器当前计数值，单位：pulse
    Velocity_P(0)=>di_Velocity_P(0), //计数器速度，计数器当前速度，单位pulse/s
    ReceiveTime(3214817177)=>udi_CounterReceiveTime(3214817177), //读取计数值时刻，读取CounterValue的系统时刻
    Error(FALSE)=>strCounter.x_Error(FALSE), //错误标志，TRUE：功能块内部发生错误
    ErrorID(CC_NO_ERRO)=>strCounter.e_ErrorID(CC_NO_ERRO));//错误ID，错误码，具体参考CC_ERROR
```

设置 PresetValue 的值为 0，将 Execute 对应的变量值置为 TRUE；引脚 Done 值为 TRUE，说明功能块执行完成

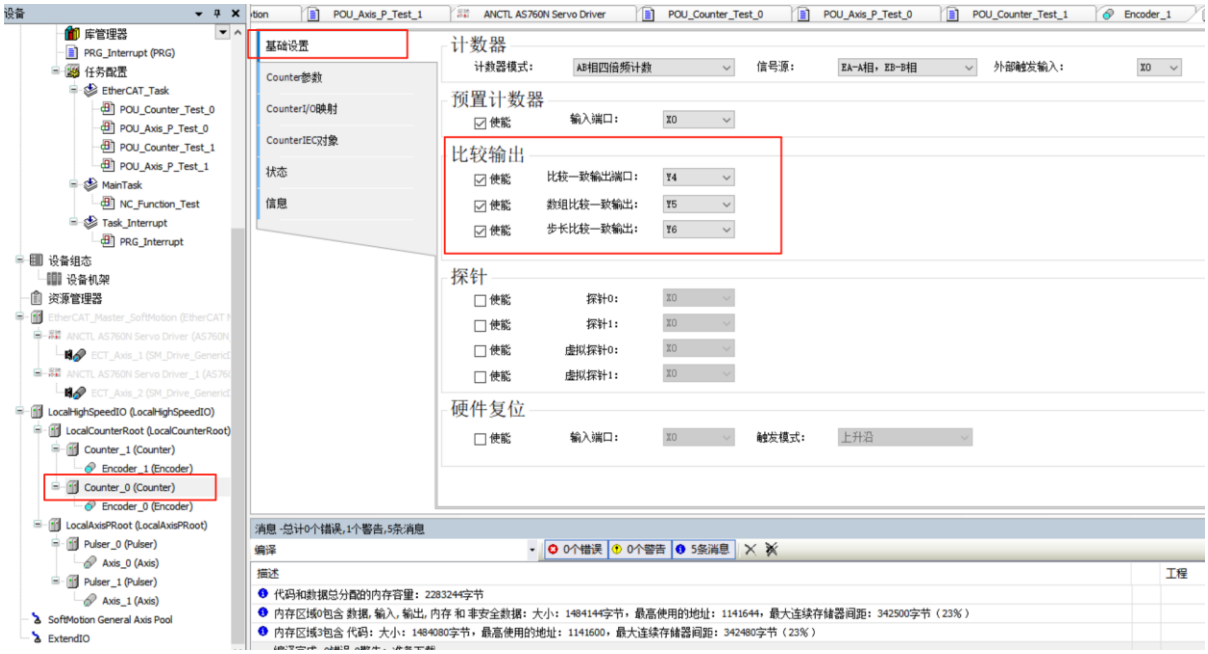
```
// 计数器预置
_CC_Preset(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute:=strPreset.x_Execute, //上升沿触发，TRUE：使能计数器预置
    Abort:=strPreset.x_Abort, //上升沿触发，TRUE：终止计数器预置
    PresetMode:=usi_PresetMode, //预置模式，0：软Execute上升沿触发1：比较一致输出时触发2：DI上升沿触发3：DI下降沿触发4：DI上升沿/下降沿触发
    PresetValue:=lr_PresetValue, //预置值，单位：unit
    DIPresetLenValid:=x_DIPresetLenValid, //PresetMode=2，3，4时，TRUE：启动预置信号长度滤波
    DIPresetLenMin:=lr_DIPresetLenMin, //预置信号最小长度，单位：unit
    DIPresetLenMax:=lr_DIPresetLenMax, //预置信号最大长度，单位：unit
    Done=>strPreset.x_Done, //TRUE：功能块执行完成
    Busy=>strPreset.x_Busy, //FALSE：功能块未执行；TRUE：功能块执行中
    CommandAborted=>strPreset.x_CmdAborted, //TRUE：功能块被终止
    Error=>strPreset.x_Error, //TRUE：功能块内部发生错误
    ErrorID=>strPreset.e_ErrorID);//错误码，具体参考CC_ERROR
```

此时可以看到计数器计数值由 82 变为 0，功能块执行成功

```
// 计数器
_CC_Counter(
    Axis:=Encoder_0, //指定使用的计数器名称
    Enable:=strCounter.x_Enable, //使能计数器，TRUE：初始化并生效功能块，FLASE：功能块不生效
    Start:=x_StartCounter, //软件启动信号（上升沿）
    TrigEdge:=usi_CounterTrigEdge, // 0：上升沿 1：下降沿
    Invert:=x_Invert, //计数值取反
    Compensation:=x_Compensation, // 补偿方式，False：用户手动补偿或不补偿 True：自动补偿
    Valid=>x_CounterValid, //计数器计数标志，TRUE：有效，FLASE：不生效
    Busy=>strCounter.x_Busy, //计数标志，FLASE：计数器未执行，TRUE：计数器已执行
    Direction=>x_CounterDirection, //计数方向，FLASE：正方向（加法计数），TRUE：反方向（减法计数）
    CounterValue:=>lr_CounterValue, //计数器值，计数器当前计数值，单位：unit
    Velocity:=>lr_Velocity, //计数器速度，计数器当前速度，单位unit/s
    CounterValue_P:=>di_CounterValue_P, //计数器值，计数器当前计数值，单位：pulse
    Velocity_P:=>di_Velocity_P, //计数器速度，计数器当前速度，单位pulse/s
    ReceiveTime:=>udi_CounterReceiveTime, //读取计数值时刻，读取CounterValue的系统时刻
    Error=>strCounter.x_Error, //错误标志，TRUE：功能块内部发生错误
    ErrorID=>strCounter.e_ErrorID);//错误ID，错误码，具体参考CC_ERROR
```

2.3.3 单点比较实例

双击计数器“Counter_0”，在“基础设置”界面中使能“比较一致输出端口”并设置相应的输出端口为 Y4



设置比较值，将引脚 CompareValue 为 120；设置比较模式为“单次模式”，即将 Mode 值置为 0；设置输出方式为“时间方式”，将 OutputType 引脚置为 False；将输出时间设置为 1s，即将 OutputValue

数值设为 10000；

```
// （单点）比较器
_CC_SetCompare(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute:=strSetCompare.x_Execute, //上升沿触发，TRUE：使能计数器预置
    Abort:=strSetCompare.x_Abort, //上升沿触发，TRUE：终止计数器预置
    CompareValue:=lr_CompareValue, //比较值，单位：unit
    Mode:=usi_SetCmpMode, //0：单次模式；1：连续模式
    OutputType:=x_SetCmpOutputType, //FALSE：时间方式；TRUE：脉冲方式
    OutputValue:=di_SetCmpOutputValue, //如为时间方式，单位为100us，如为脉冲方式，单位为pulse。
    Done=>strSetCompare.x_Done, //完成标志，如为单次模式，执行一次输出后，Done，功能自动终止。连续模式，此信号无用
    Busy=>strSetCompare.x_Busy, //FALSE：功能块未执行；TRUE：功能块执行中
    CommandAborted=>strSetCompare.x_CmdAborted, //TRUE：功能块被终止
    Output=>x_SetCmpOutput, //比较一致输出，与DO一致
    CmpCount=>ui_CmpCount, //比较输出次数
    Position=>lr_SetCmpPosition, //当前需要被比较的值
    Error=>strSetCompare.x_Error, //TRUE：功能块内部发生错误
    ErrorID=>strSetCompare.e_ErrorID);//错误码，具体参考CC_ERROR
```

将 Execute 引脚置为 TRUE，使能功能块，此时计数器计数值为 75

```
// （单点）比较器
_CC_SetCompare(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute:=strSetCompare.x_Execute, //上升沿触发，TRUE：使能计数器预置
    Abort:=strSetCompare.x_Abort, //上升沿触发，TRUE：终止计数器预置
    CompareValue:=lr_CompareValue, //比较值，单位：unit
    Mode:=usi_SetCmpMode, //0：单次模式；1：连续模式
    OutputType:=x_SetCmpOutputType, //FALSE：时间方式；TRUE：脉冲方式
    OutputValue:=di_SetCmpOutputValue, //如为时间方式，单位为100us，如为脉冲方式，单位为pulse。
    Done=>strSetCompare.x_Done, //完成标志，如为单次模式，执行一次输出后，Done，功能自动终止。连续模式，此信号无用
    Busy=>strSetCompare.x_Busy, //FALSE：功能块未执行；TRUE：功能块执行中
    CommandAborted=>strSetCompare.x_CmdAborted, //TRUE：功能块被终止
    Output=>x_SetCmpOutput, //比较一致输出，与DO一致
    CmpCount=>ui_CmpCount, //比较输出次数
    Position=>lr_SetCmpPosition, //当前需要被比较的值
    Error=>strSetCompare.x_Error, //TRUE：功能块内部发生错误
    ErrorID=>strSetCompare.e_ErrorID);//错误码，具体参考CC_ERROR

IF x_数组初始化 THEN
```

监视 1

表达式	应用	类型	值	准备值	执行点	地址	注释
POU_Counter_Test_0.lr_CounterValue	Device.Application	LREAL	75		循环监测		计数器值
Axis_0.fActPosition	Device.Application	LREAL	0		循环监测		Paramet...
Axis_0.nAxisState	Device.Application	SMC_AXIS_STATE	power_off		循环监测		State of...

转动编码器，逐渐增大编码器计数值，当编码器计数值经过 120 时，Done 引脚被置为 TRUE，同时 CmpCount 计数值增加 1，且可以观察到 PLC 的 Y4 输出端口会输出 1s 的高电平（Y4 端口的灯会亮 1s）

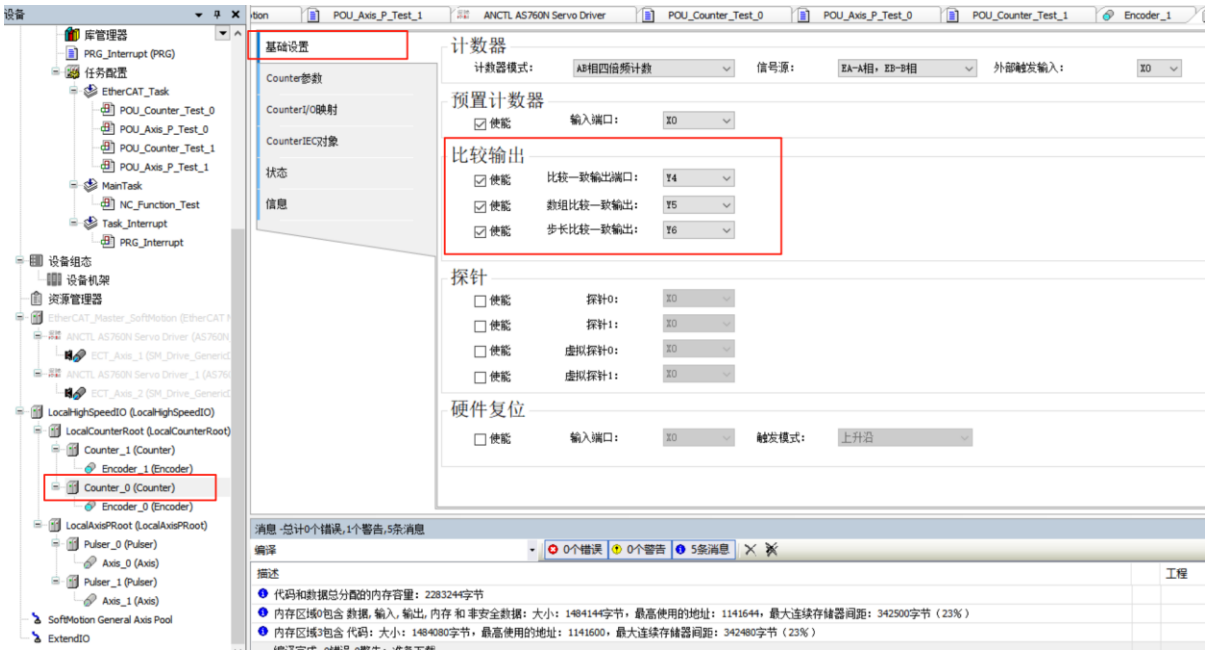
```
// （单点）比较器
_CC_SetCompare(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute:=strSetCompare.x_Execute, //上升沿触发，TRUE：使能计数器预置
    Abort:=strSetCompare.x_Abort, //上升沿触发，TRUE：终止计数器预置
    CompareValue:=lr_CompareValue, //比较值，单位：unit
    Mode:=usi_SetCmpMode, //0：单次模式；1：连续模式
    OutputType:=x_SetCmpOutputType, //FALSE：时间方式；TRUE：脉冲方式
    OutputValue:=di_SetCmpOutputValue, //如为时间方式，单位为100us，如为脉冲方式，单位为pulse。
    Done=>strSetCompare.x_Done, //完成标志，如为单次模式，执行一次输出后，Done，功能自动终止。连续模式，此信号无用
    Busy=>strSetCompare.x_Busy, //FALSE：功能块未执行；TRUE：功能块执行中
    CommandAborted=>strSetCompare.x_CmdAborted, //TRUE：功能块被终止
    Output=>x_SetCmpOutput, //比较一致输出，与DO一致
    CmpCount=>ui_CmpCount, //比较输出次数
    Position=>lr_SetCmpPosition, //当前需要被比较的值
    Error=>strSetCompare.x_Error, //TRUE：功能块内部发生错误
    ErrorID=>strSetCompare.e_ErrorID);//错误码，具体参考CC_ERROR

IF x_数组初始化 THEN
```

监视 1

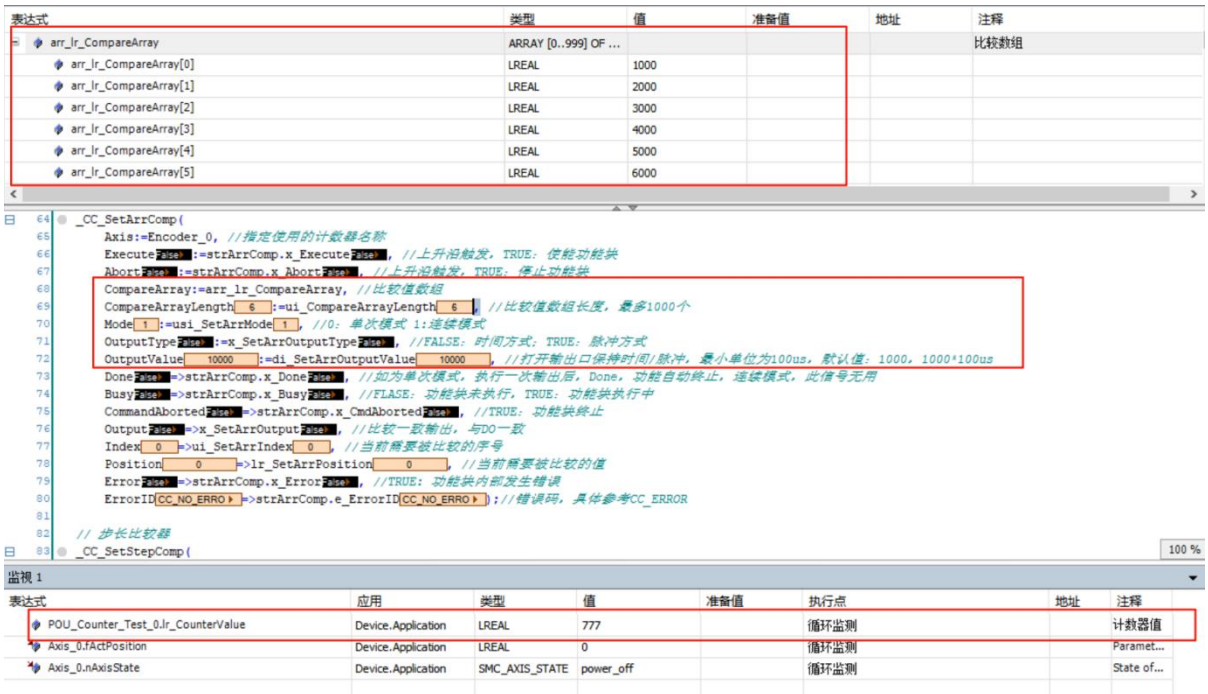
表达式	应用	类型	值	准备值	执行点	地址	注释
POU_Counter_Test_0.lr_CounterValue	Device.Application	LREAL	297		循环监测		计数器值
Axis_0.fActPosition	Device.Application	LREAL	0		循环监测		Paramet...
Axis_0.nAxisState	Device.Application	SMC_AXIS_STATE	power_off		循环监测		State of...

2.3.4 数组比较实例

双击计数器“Counter_0”，在“基础设置”界面中使能“数组比较一致输出”并设置相应的输出端口为 Y5

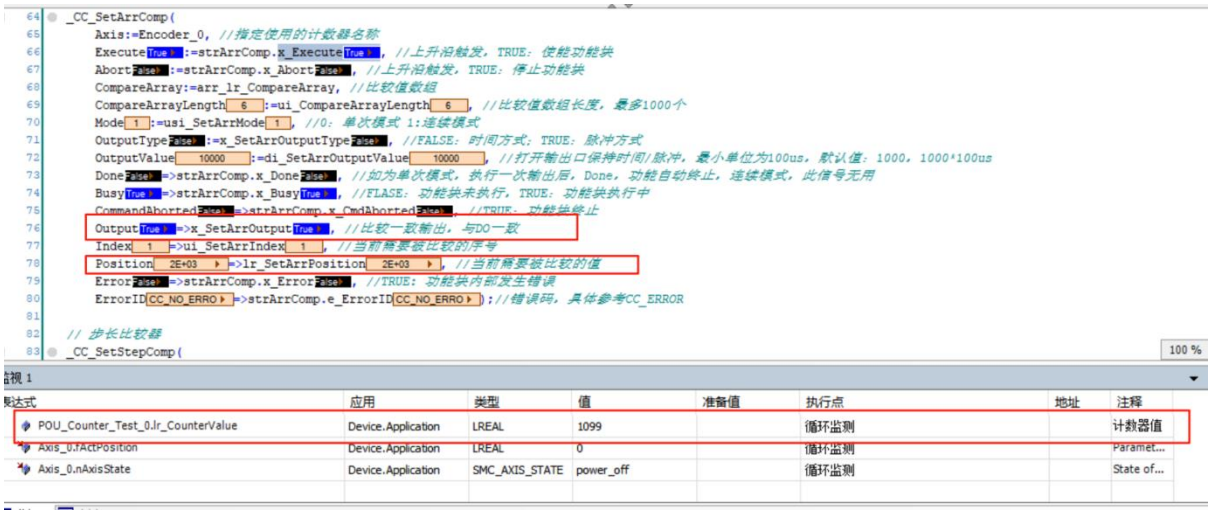


设置比较数组 arr_lr_CompareArray，将比较数组的前 6 个元素分别设为 1000、2000、3000、4000、5000、6000；设置比较数组长度为 6，即将计数器计数值与比较数组的前 6 个元素比较；设置比较模式为“连续模式”，即将 Mode 值置为 1；设置输出方式为“时间方式”，将 OutputType 引脚置为 False；将输出时间设置为 1s，即将 OutputValue 数值设为 10000；此时编码器计数值为 777。

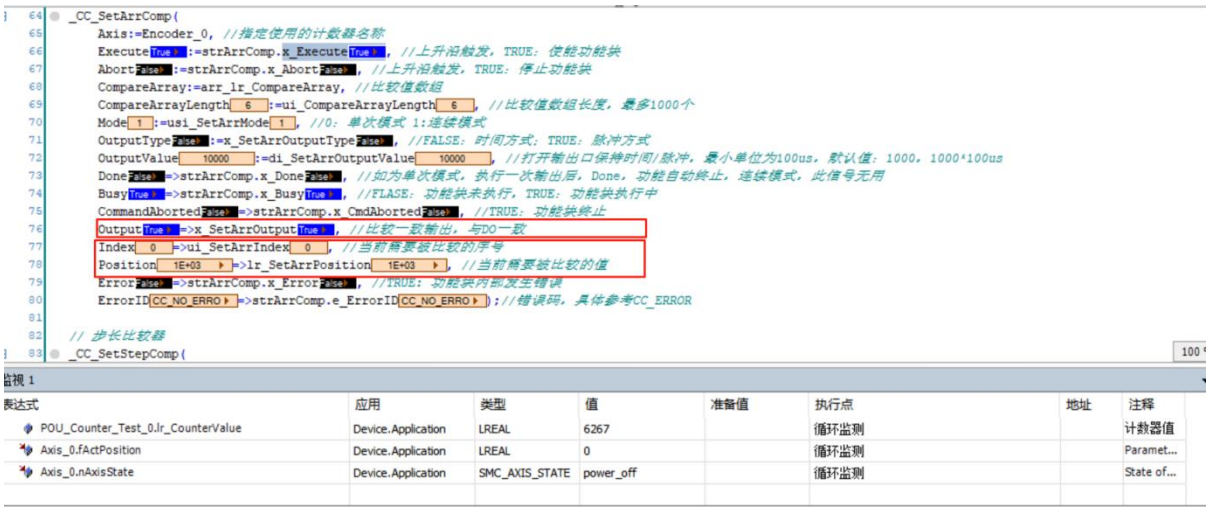


使能功能块，将 Execute 引脚置为 TRUE；此时 BUSY 引脚输出为 TURE，说明功能块正在工作

转动编码器，改变编码器计数值，当编码器计数值改变为 1000 时，Output 引脚会被置为 TRUE，时间为 1s，对应的硬件端口 Y5 会输出 1s 高电平（Y5 端口灯会亮起 1s）；同时 Position 引脚会显示下一个等待比较的数值



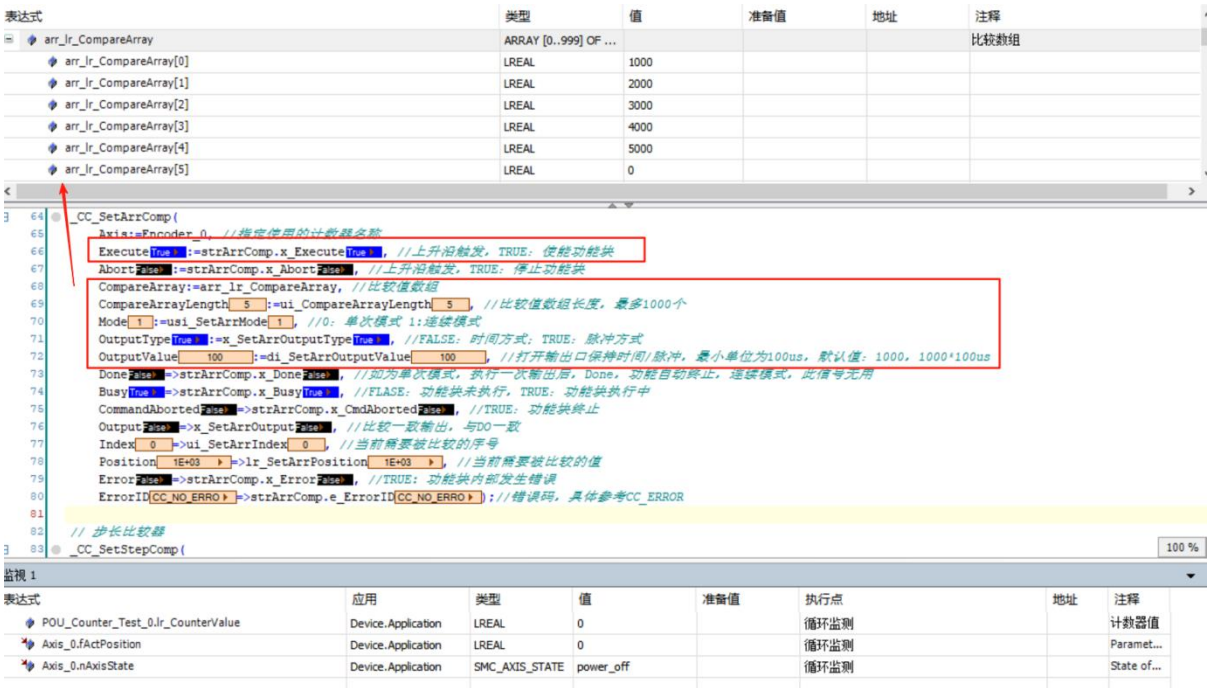
继续改变编码器计数值，当编码器计数值依次改变为 2000、3000、4000、5000、6000 时，Output 引脚会被置为 TRUE，时间为 1s，对应的硬件端口 Y5 会输出 1s 高电平（Y5 端口灯会亮起 1s）。在 6 个元素比较完成后会重新比较比较数组的第一个元素 1000。



2.3.5 探针实例

可结合“数组比较一致输出”功能来验证探针位置锁存，双击添加的计数器“Counter_0”，在“基础设置”界面使能“数组比较一致输出”，设置输出端口为 Y5；使能“探针 0”，设置探针输入端口为 X5；将数组比较一致的输出端口 Y5 短接到计数器探针的输入端口 X5，当数组比较一致时会触发探针位置锁存

设置比较值数组 CompareArray 前 5 个元素为 1000、2000、3000、4000、5000；设置 CompareArrayLength 比较值数组长度为 5；设置比较模式 Mode 为 1，即设置为连续模式；设置 TriggerEdge 值为 0，即设置探针为上升沿触发；设置输出模式 OutputType 引脚为 TRUE，即设置输出为脉冲输出；设置 OutputValue 为 100，输出脉冲为 10000us；将 Execute 引脚置为 TRUE



设置 Mode 引脚值为 1，即设置探针为连续模式；将探针 Execute 引脚置为 TRUE，使能功能块

```
_CC_TouchProbe(
    Axis:=Encoder_0, //指定使用的计数器名称
    Execute:=strTouchProbe.x_Execute, //上升沿触发，TRUE：使能功能块
    Abort:=strTouchProbe.x_Abort, //上升沿触发，TRUE：停止功能块
    ProbeID:=usi_TPID, //探针ID
    TriggerEdge:=usi_TPTriggerEdge, //0：上升沿触发；1：下降沿触发
    Mode:=usi_TPMode, //0：单次模式 1:连续模式
    Done=>strTouchProbe.x_Done, //如为单次模式，执行一次输出后，Done，功能自动终止，连续模式，此值号无用
    Busy=>strTouchProbe.x_Busy, //FLASE：功能块未执行，TRUE：功能块执行中
    CommandAborted=>strTouchProbe.x_CmdAborted, //TRUE：功能块终止
    PosPosition=>lr_TPPosPosition, //上升沿锁存位置，单位：unit
    NegPosition=>lr_TPNegPosition, //下降沿锁存位置，单位：unit
    PosPosition_P=>di_TPPosPosition_P, //上升沿锁存位置，单位：pulse
    NegPosition_P=>di_TPNegPosition_P, //下降沿锁存位置，单位：pulse
    PosTime=>udi_TPPosTime, //上升沿锁存时刻，单位：us
    NegTime=>udi_TPNegTime, //下降沿锁存时刻，单位：us
    CycleCount=>usi_TPCycleCount, //锁存次数
    Error=>strTouchProbe.x_Error, //TRUE：功能块内部发生错误
    ErrorID=>strTouchProbe.e_ErrorID);//错误码，具体参考CC_ERROR
```

定义一个锁存数组 ar_lockarry 来存储探针触发时的上升沿锁存位置，即探针功能块的 PosPosition 引脚，对应变量为 lr_TPPosPosition；锁存数组元素标号设置为比较数组当前需要被比较的序号 ui_SetArrIndex；

```
ar_lockarry[ui_SetArrIndex] := lr_TPPosPosition;
```

转动编码器，或者用 CC_PresetCounter 功能块依次改变编码器计数值为 1000、2000、3000、4000、5000，观察到探针锁存位置与设定的数组位置一致

表达式	类型	值	准备值	地址	注释
arr_lr_CompareArray	ARRAY [0..999] OF ...				比较数组
arr_lr_CompareArray[0]	LREAL	1000			
arr_lr_CompareArray[1]	LREAL	2000			
arr_lr_CompareArray[2]	LREAL	3000			
arr_lr_CompareArray[3]	LREAL	4000			
arr_lr_CompareArray[4]	LREAL	5000			
arr_lr_CompareArray[5]	LREAL	0			

表达式	类型	值	准备值	地址	注释
ar_lockarry	ARRAY [0..999] OF ...				位置锁存数组
ar_lockarry[0]	LREAL	5000			
ar_lockarry[1]	LREAL	1000			
ar_lockarry[2]	LREAL	2000			
ar_lockarry[3]	LREAL	3000			
ar_lockarry[4]	LREAL	4000			
ar_lockarry[5]	LREAL	0			
ar_lockarry[6]	LREAL	0			

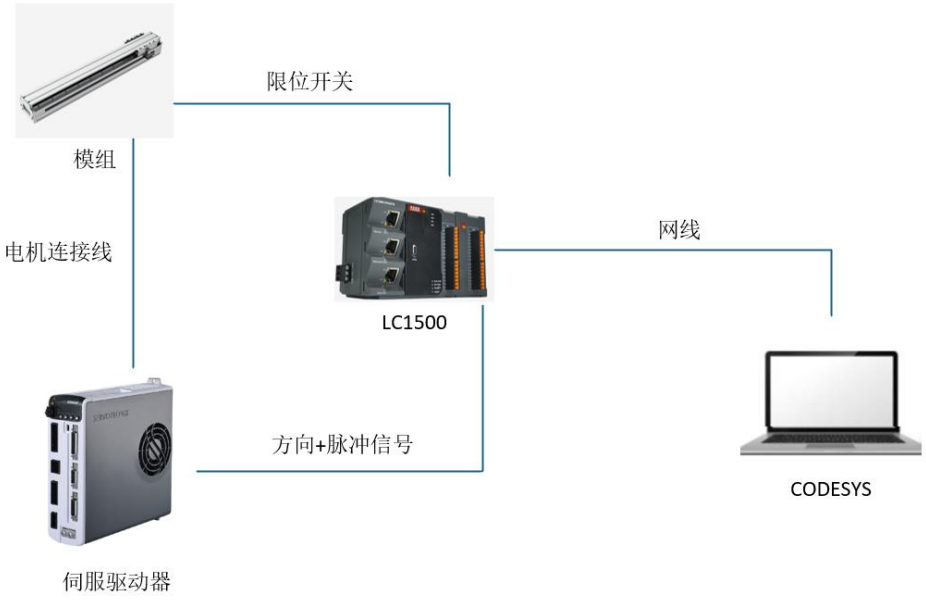
第二章 脉冲输出实例说明

1.1 硬件配置

1.1.1 硬件选择

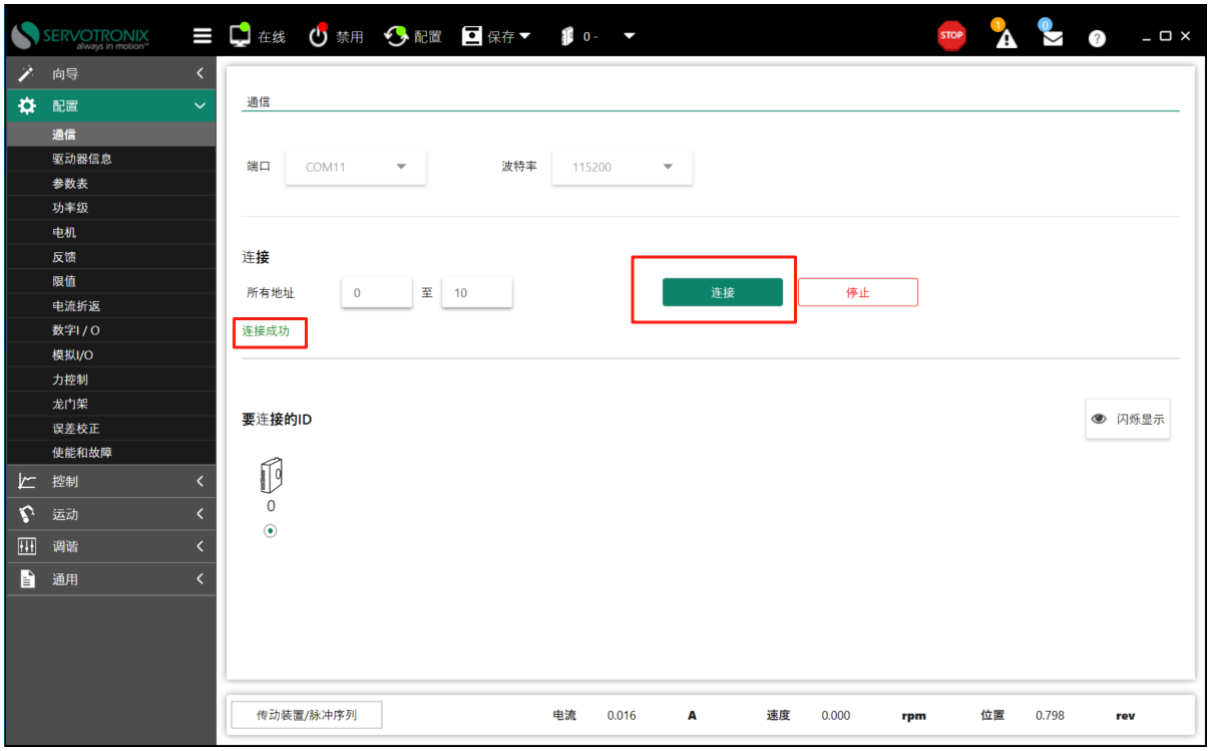
1）LC1500PLC
2）丝杆模组
3）高创 CDHD2 伺服驱动器
4）CODESYS

1.1.2 硬件接线图

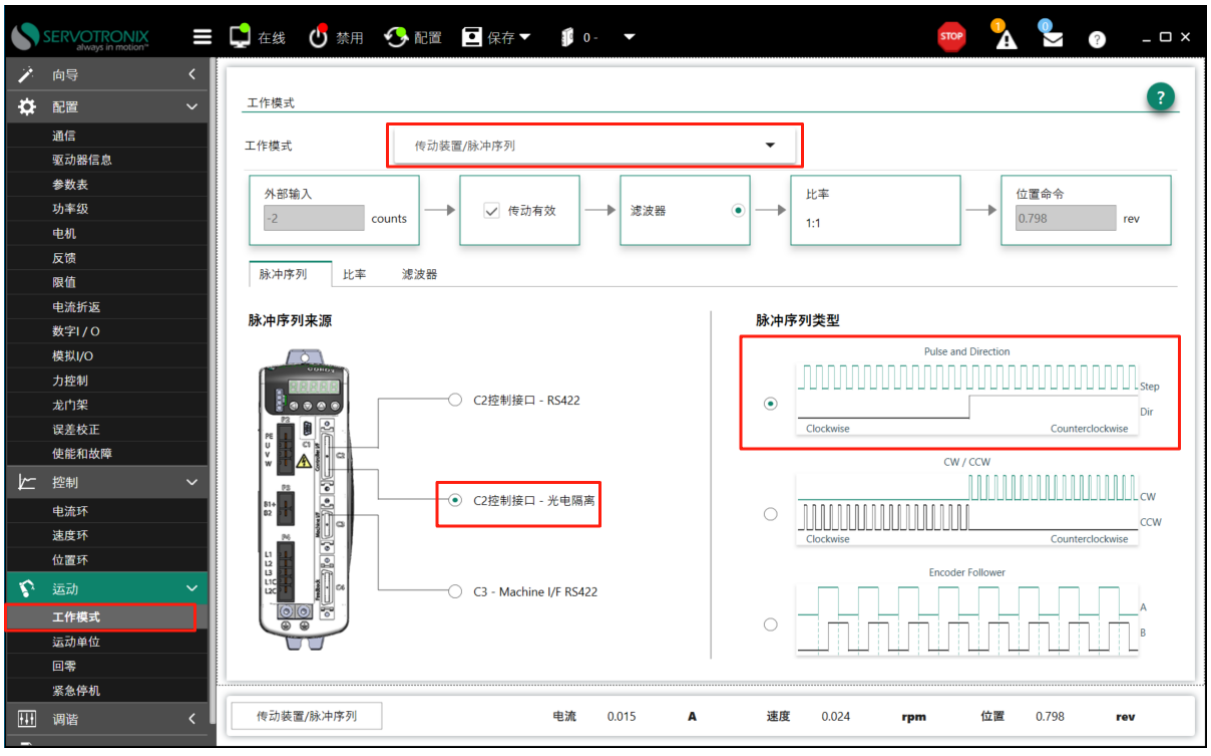


1.2 软件配置

打开高创伺服调试软件“ServoStudio”，配置好端口号与波特率，点击“连接”，连接伺服驱动器

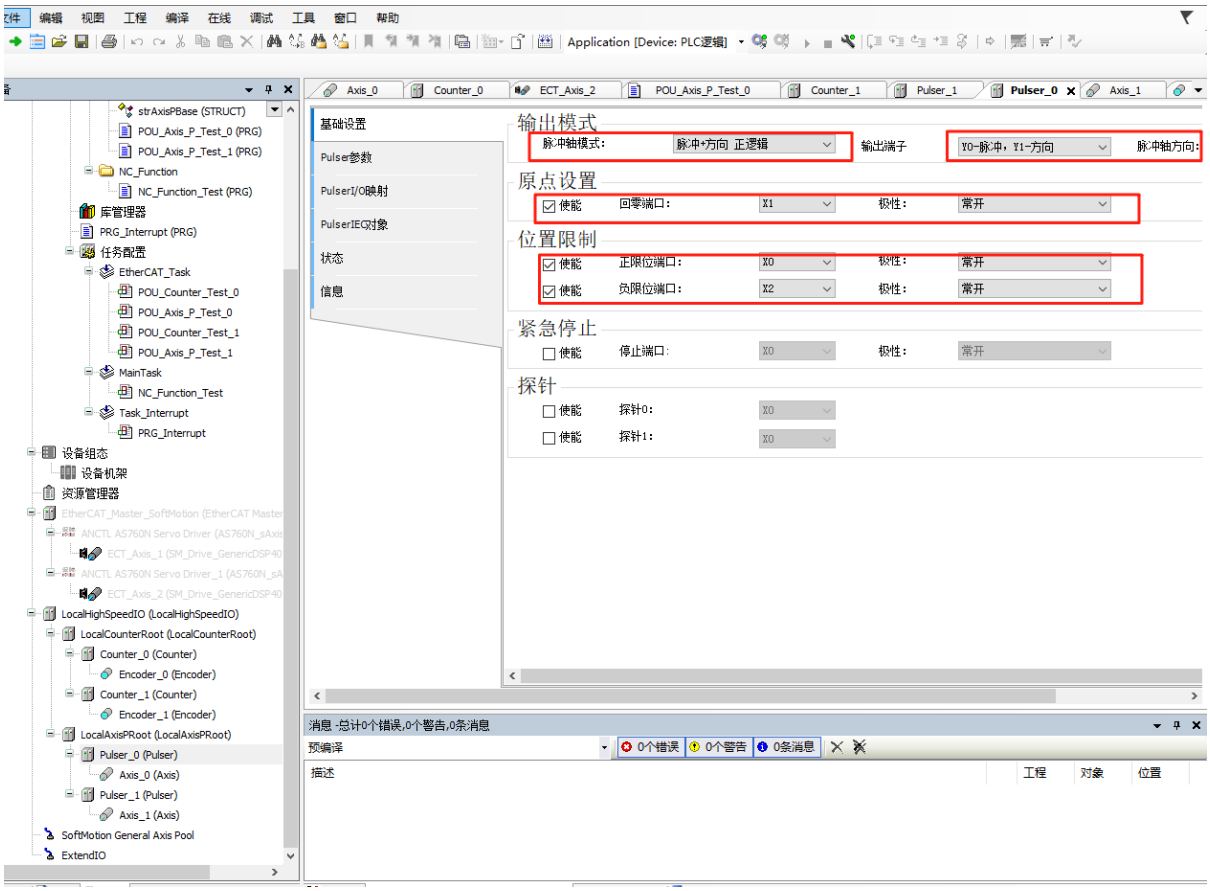


连接成功后，点击“运动”下的“工作模式”：“工作模式”处选择“传动装置/脉冲序列”；“脉冲序列来源”处选择“C2 控制接口-光电隔离”；“脉冲序列类型”处选择“Pulse and Direction”

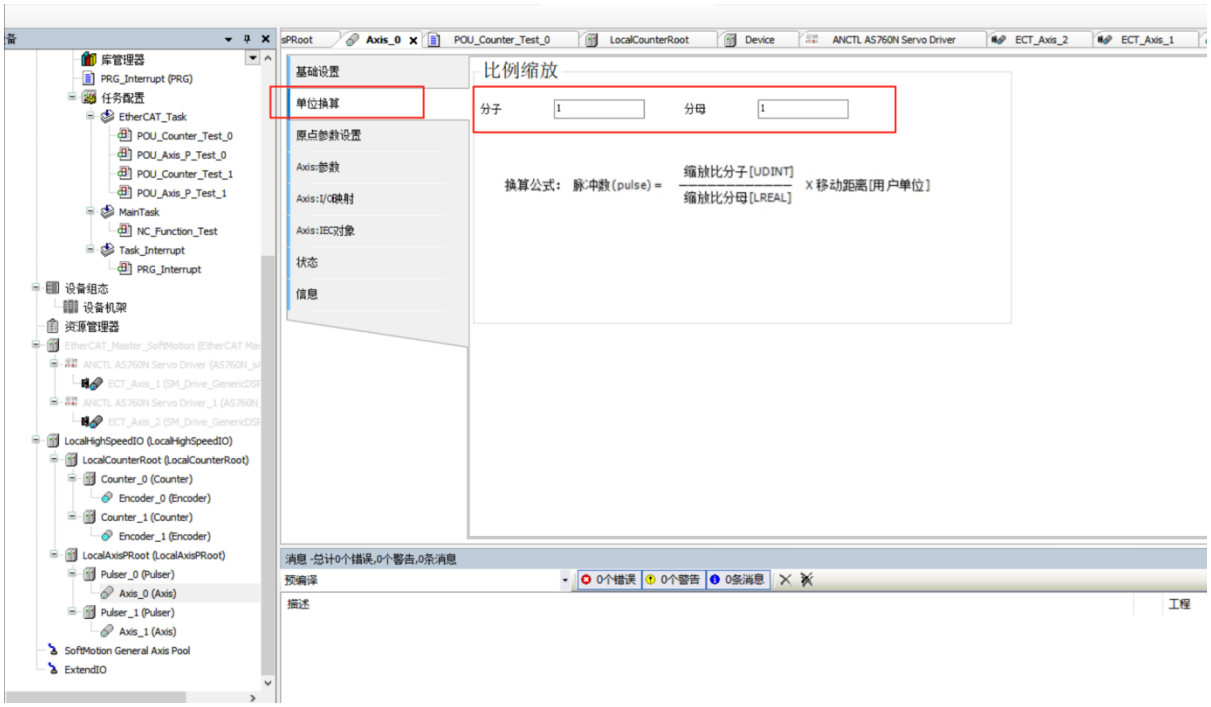


打开凌臣 IDE 编程软件，选择对应的高速脉冲轴设备，点击“Pulser_0”,在“输出模式”中选择“脉冲+方向 正逻辑”的脉冲轴模式，并选择 Y0 口为“脉冲”输出、Y1 口为“方向”输出，选择脉冲轴方向为“正方向”；在“基础设置”界面对“原点设置”中的“回零端口”、“位置限制”中的“正限位端口”与“负限位端口”进行使能，并选择对应的硬件输入端口与极性：本次实例中原点限位开关输入接在 PLC 的 X1 端口，因此“回零端口”选择 X1,极性选择“常开”；正限位端口输入接在 PLC 的 X0 端口，因此“正限位端口”选择 X0,极性选择“常开”；负限位开关输入接

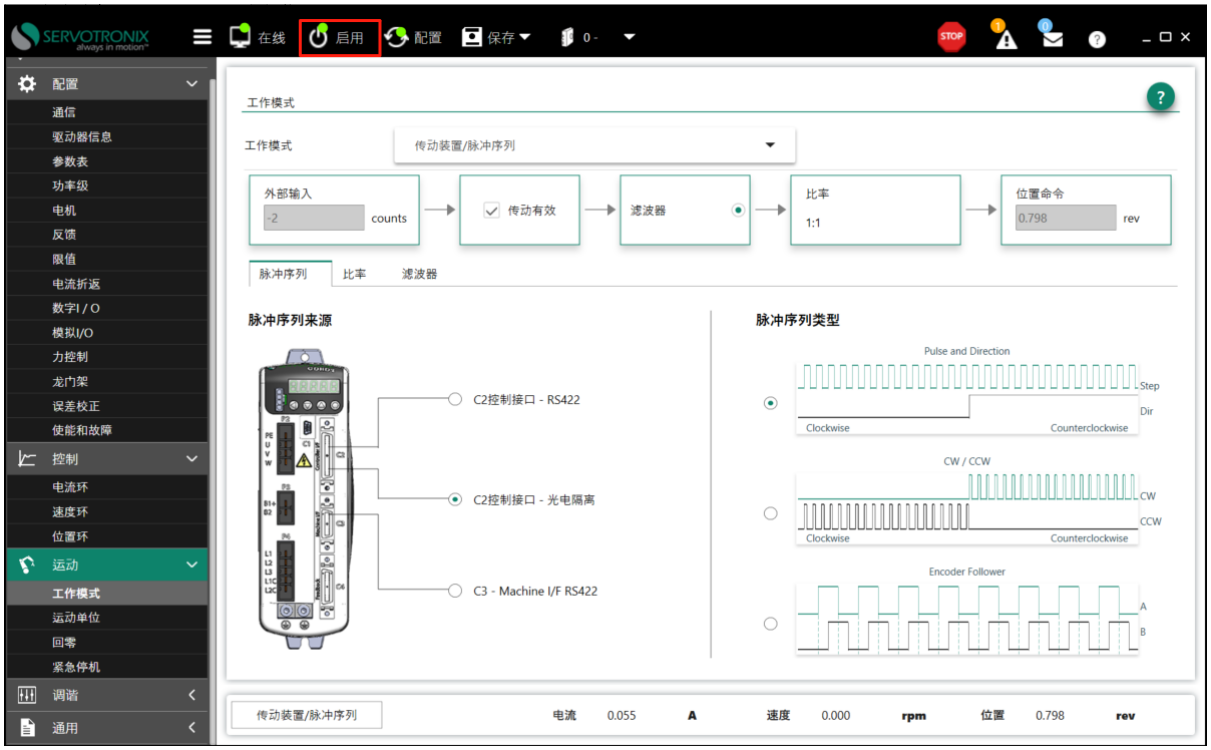
在 PLC 的 X2 端口，因此“负限位端口”选择 X2，极性选择“常开”。

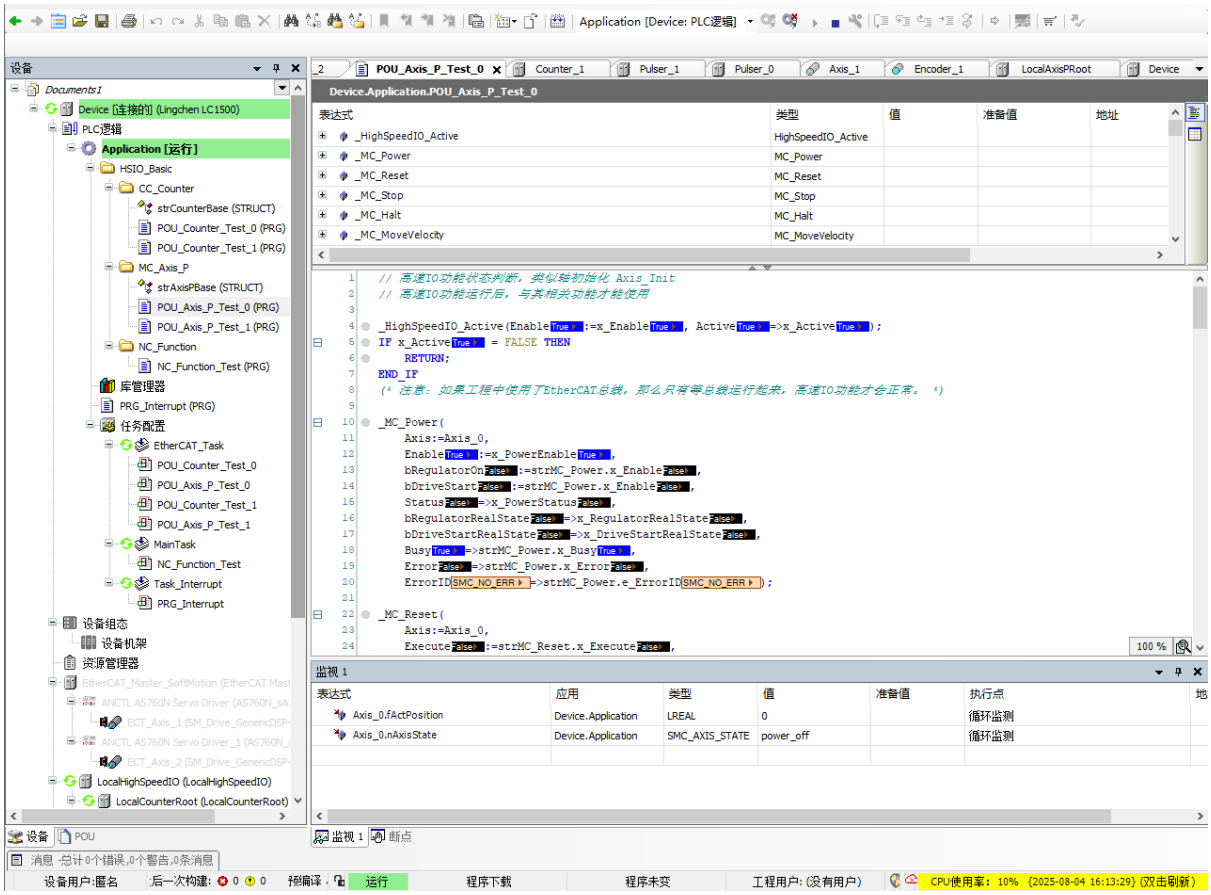


点击所要控制的脉冲轴“Axis_0”，在“基础设置”选择“旋转模式”；在“单位换算”根据公式设置相应的分子、分母，本次实例选择 1：1；

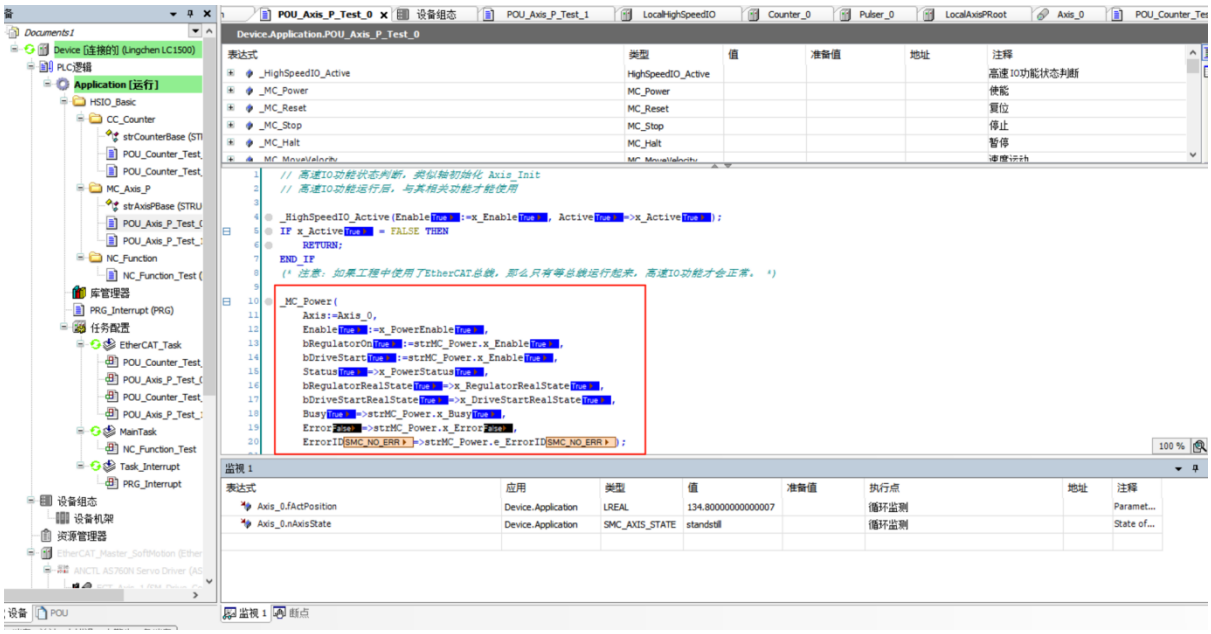


点击“ServoStudio”中的“启用”使能电机；在 IDE 中连接 PLC 后“登录”





在 IDE 中启用“MC_Power”功能块，对功能块进行使能，开始进行功能块测试即可（只有在使能 MC_Power 功能块之后才能使用其他高速脉冲相关的功能块）

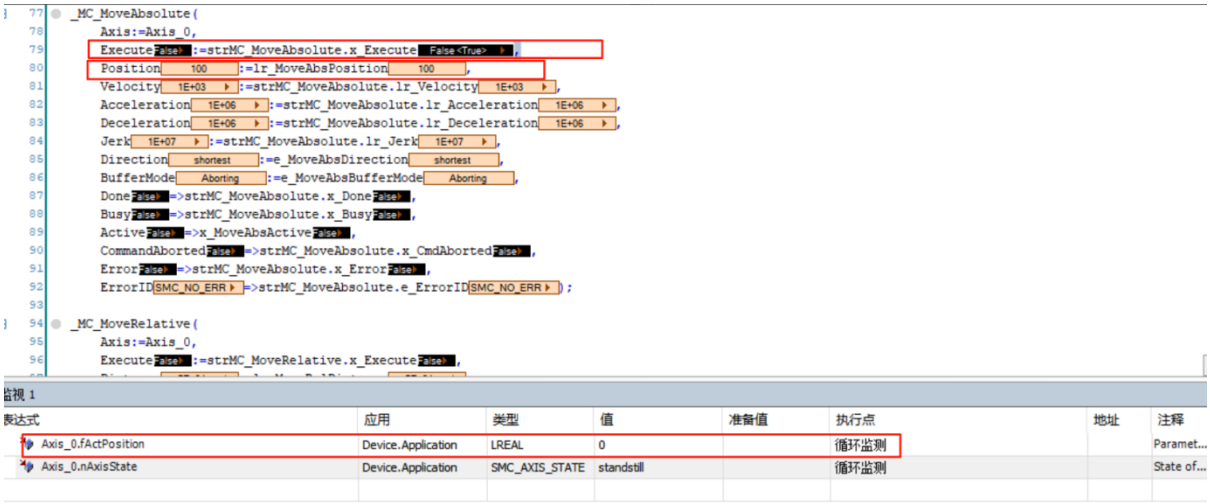


2.1 功能块应用实例

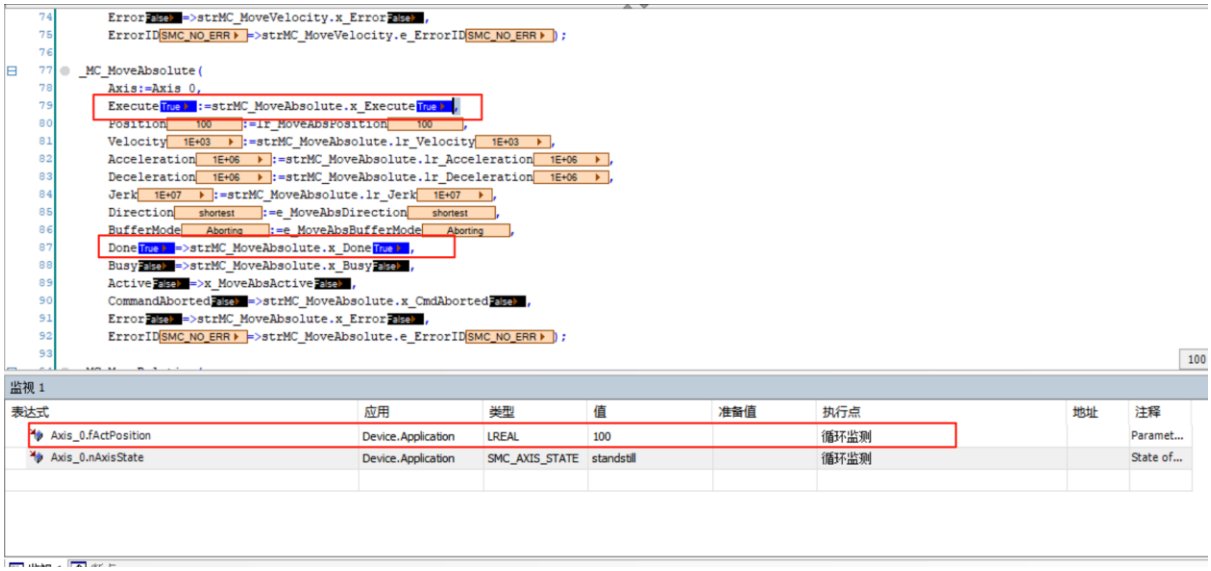
2.1.1 轴绝对位置控制实例

以轴“Axis_0”的绝对控制为例，如下所示：

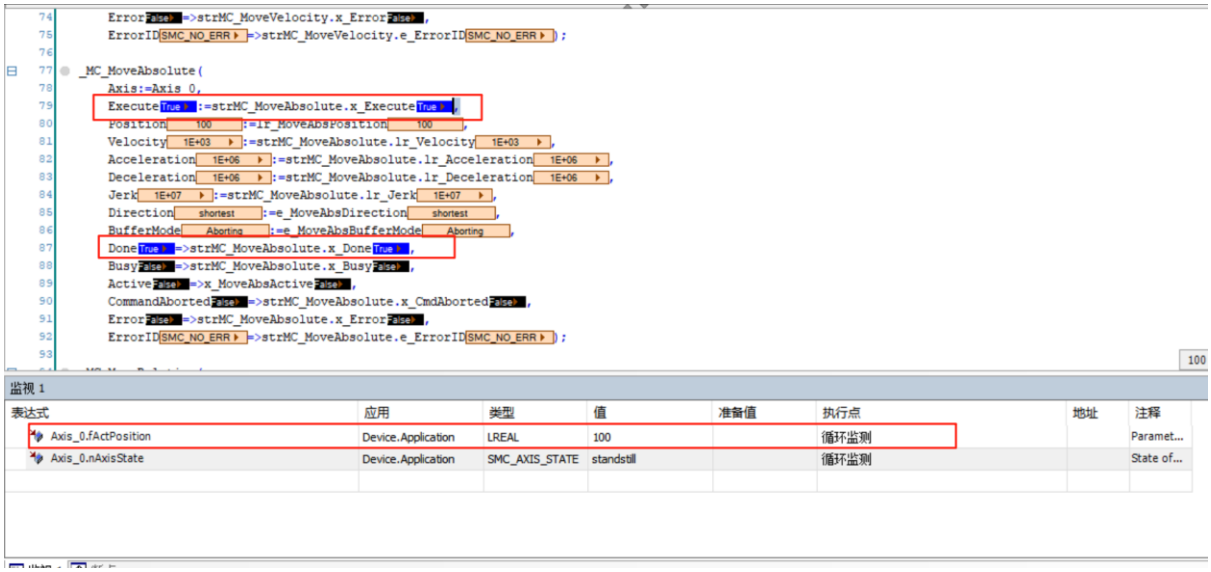
此时轴实际位置“Axis_0.fActPosition”为 0，绝对位置“lr_MoveAbsPosition”设为 100，设置好速度值后，点击绝对位置触发“strMC_MoveAbsolute.x_Execute”预值<TRUE>，后右击选择写入其所有值,或者按“Ctrl+F7”。



绝对位置触发后，电机以 1000pul/s 的速度开始运行 strMC_MoveAbsolute.x_Done 为 TRUE 时停止，此时测试轴位置 Axis_0.fActPosition 为 100，正好等于设置值。



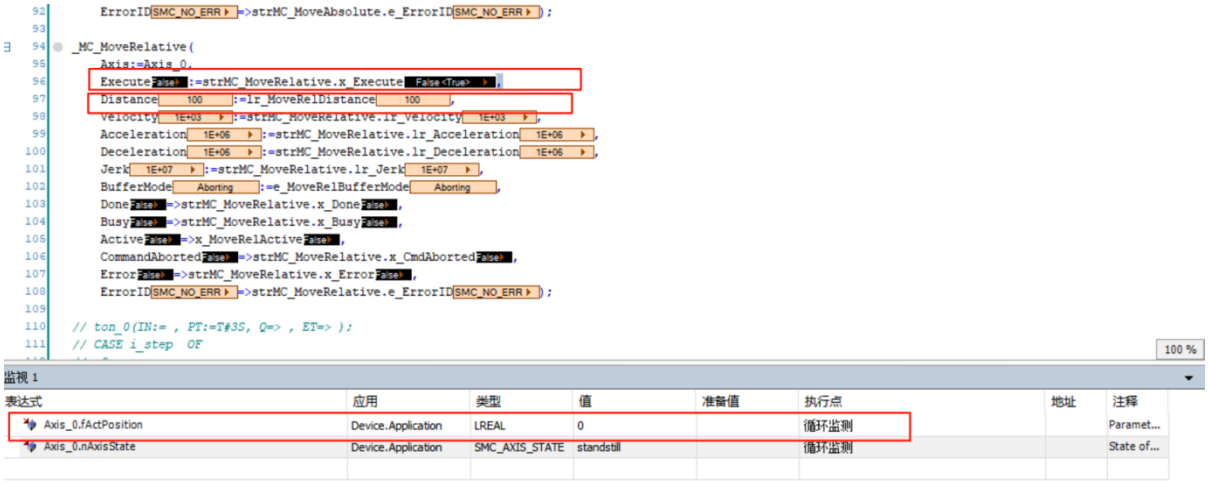
在设置值为 100 不变情况下，再次触发“strMC_MoveAbsolute.x_Execute”，给其上值 TRUE 后，轴位置仍然为 100 等于设置值，电机也处在当前位置不会运动。



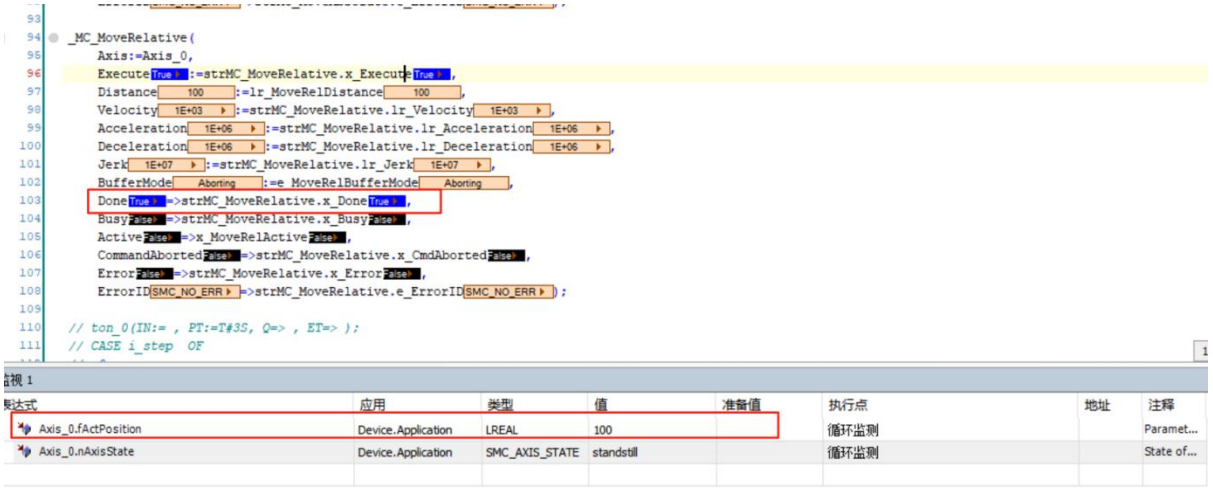
2.1.2 轴相对位置控制实例

以轴“Axis_0”的相对位置为例，如下所示

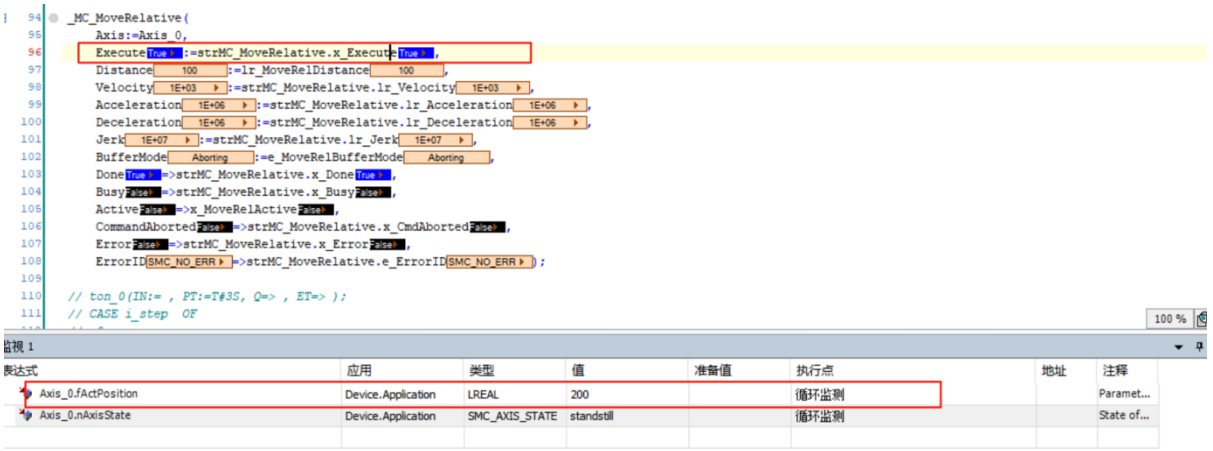
轴实际位置“Axis_0.fActPosition”为 0 时，相对位置值 lr_MoveRelDistance 设为 100，触发 strMC_MoveRelative.x_Execute 启动模块运行，同时所接的电机开始运动。



strMC_MoveRelative.x_Done 亮时相对运动结束，电机也停止运动，此时轴位置等于设置值 100.



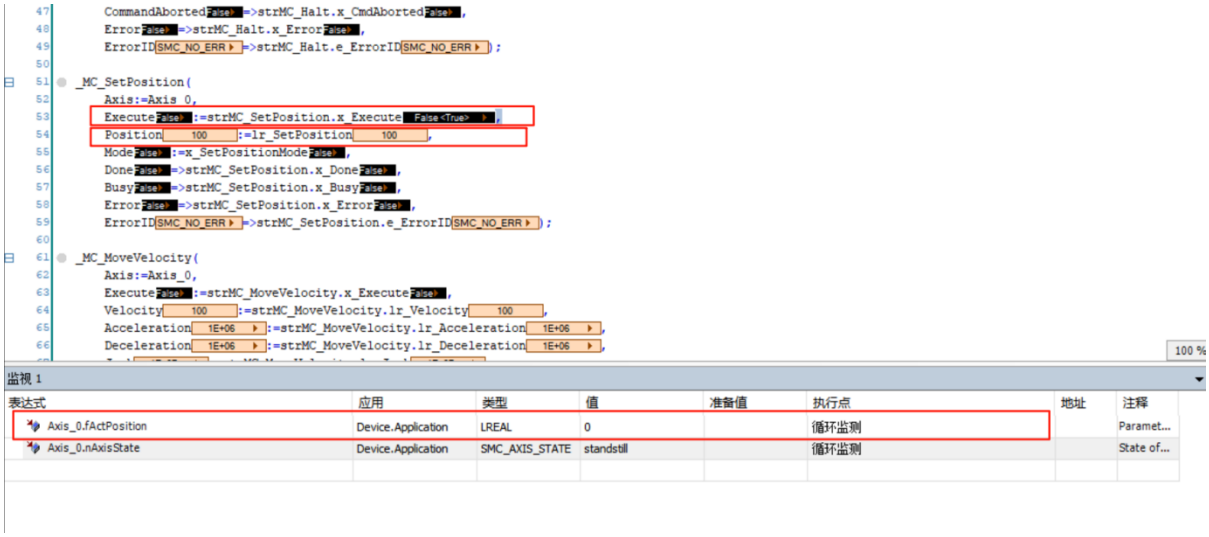
保持 lr_MoveRelDistance 值为 100 不变，再次触发 strMC_MoveRelative.x_Execute，模块程序和电机再次运行至轴位置为 200 时停止，可见此功能块不同与绝对运动，而是可以再当前位置上不断做相对运动。



2.1.3 高速轴设置位置实例

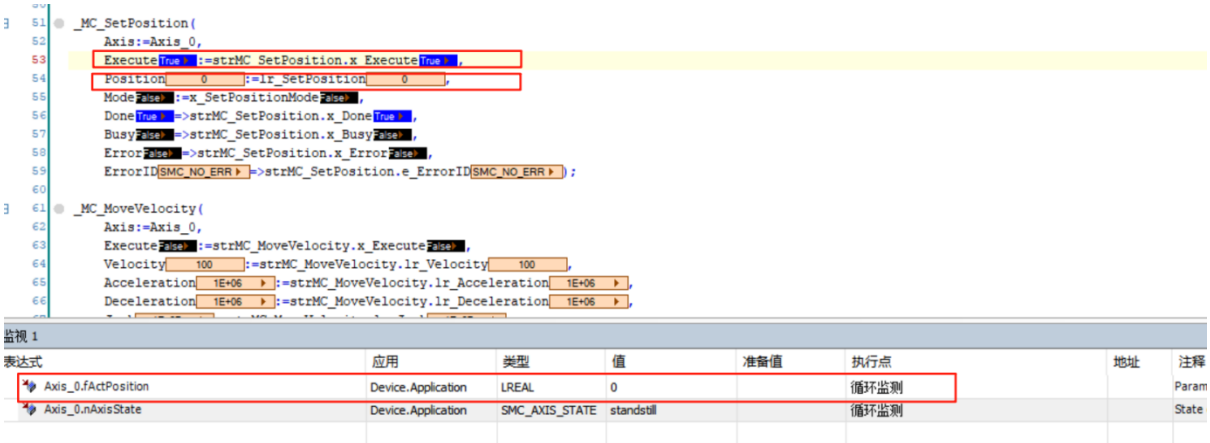
以轴“Axis0”的设置位置为例，如下所示：

此时轴实际位置 Axis_0.fActPosition 为 0，设置位值 lr_SetPosition 设为 100，触发 strMC_SetPosition.x_Execute 后功能模块运行，但电机不会运动。



strMC_SetPosition.x_Done 完成标志亮起，模块运行结束，此时实际位置为 100。

再把设置位置改为 0，重新触发 strMC_SetPosition.x_Execute 后，轴实际位置 Axis_0.fActPosition 变为设置值 0，由此可见，此功能可设置改变标定轴位置值，但不需要电机运动。

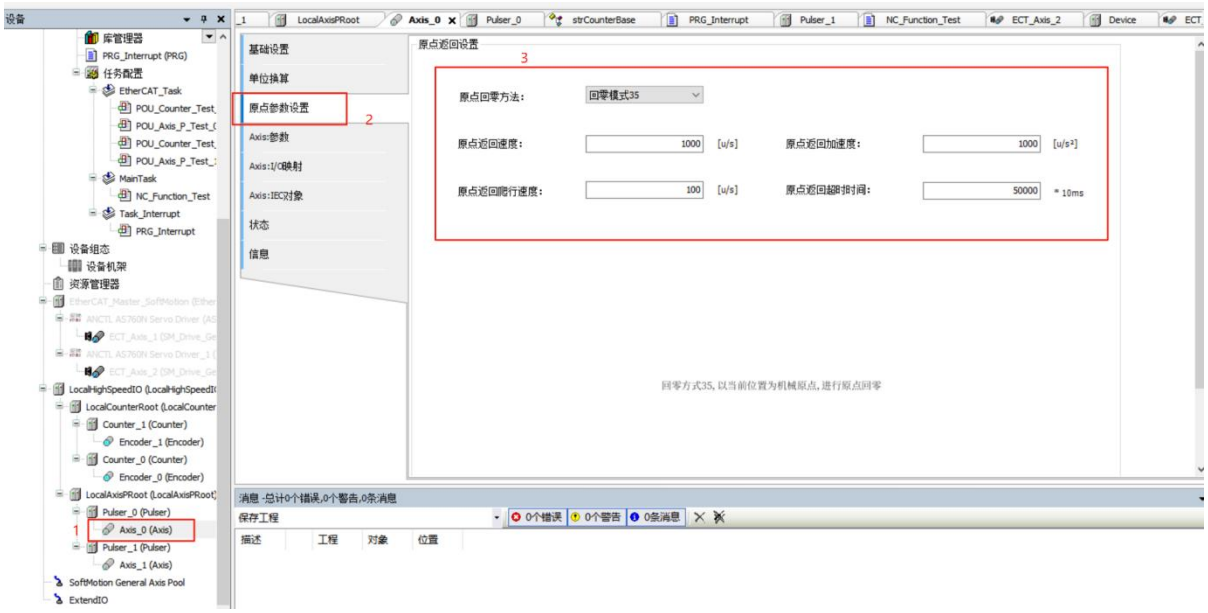


2.1.4 回原方式

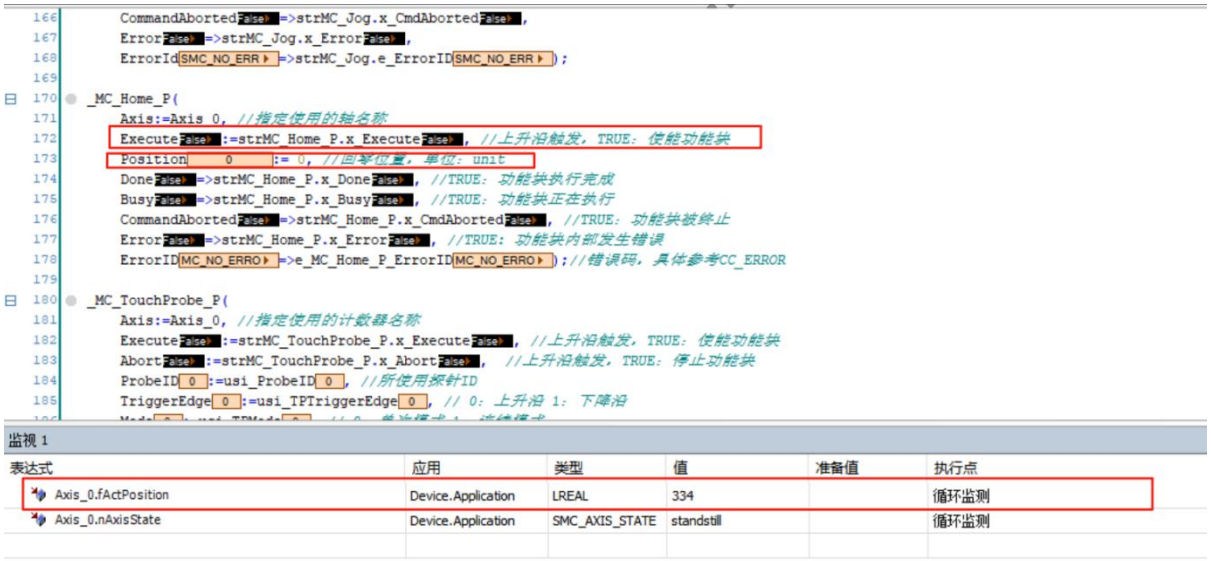
以回零方式 35：轴当前位置为原点例子

以轴“Axis_0”为例进行以轴当前位置为原点回原，如下所示：

依次点击脉冲轴“Axis_0”、“原点参数设置”，进行回零参数设置



随后打开 POU，此时轴位置为 334，赋值回零功能块触发变 strMC_Home_P.x_Execute 即可。



触发后当前位置值立即变为 0 等于零点设置位置值，strMC_Home_P.x_Done 亮起功能块运行结束，此过程中电机不会运动而当前位置就变为零点设置值，符合回零方式 35 的功能。

```
166         CommandAborted=>strMC_Jog.x_CmdAborted,
167         Error=>strMC_Jog.x_Error,
168         ErrorId=>strMC_Jog.e_ErrorID);
169
170     _MC_Home_P(
171         Axis:=Axis_0, //指定使用的轴名称
172         Execute:=strMC_Home_P.x_Execute, //上升沿触发，TRUE：使能功能块
173         Position:= 0, //回零位置，单位：unit
174         Done=>strMC_Home_P.x_Done, //TRUE：功能块执行完成
175         Busy=>strMC_Home_P.x_Busy, //TRUE：功能块正在执行
176         CommandAborted=>strMC_Home_P.x_CmdAborted, //TRUE：功能块被终止
177         Error=>strMC_Home_P.x_Error, //TRUE：功能块内部发生错误
178         ErrorID=>e_MC_Home_P_ErrorID);//错误码，具体参考CC_ERROR
179
180     _MC_TouchProbe_P(
181         Axis:=Axis_0, //指定使用的计数器名称
182         Execute:=strMC_TouchProbe_P.x_Execute, //上升沿触发，TRUE：使能功能块
183         Abort:=strMC_TouchProbe_P.x_Abort, //上升沿触发，TRUE：停止功能块
184         ProbeID:=usi_ProbeID, //所使用探针ID
185         TriggerEdge:=usi_TPTriggerEdge, // 0：上升沿 1：下降沿
```

监视 1

表达式	应用	类型	值	准备值	执行点
Axis_0.fActPosition	Device.Application	LREAL	0		循环监测
Axis_0.nAxisState	Device.Application	SMC_AXIS_STATE	standstill		循环监测

以回零方式 18：正向运动找极限的下降沿回原例子

以轴“Axis_0”为例，如下所示：

配置好回原轴参数，以 X0 为原点信号并选择常开，打开回原使能，选择相应的回原方式 18 即可。在正向限位没有被触发时，触发 strMC_Home_P.x_Execute 开启回原运动

```
164         Jerk:=strMC_Jog.lr_Jerk,
165         Busy=>strMC_Jog.x_Busy,
166         CommandAborted=>strMC_Jog.x_CmdAborted,
167         Error=>strMC_Jog.x_Error,
168         ErrorId=>strMC_Jog.e_ErrorID);
169
170     _MC_Home_P(
171         Axis:=Axis_0, //指定使用的轴名称
172         Execute:=strMC_Home_P.x_Execute, //上升沿触发，TRUE：使能功能块
173         Position:= 0, //回零位置，单位：unit
174         Done=>strMC_Home_P.x_Done, //TRUE：功能块执行完成
175         Busy=>strMC_Home_P.x_Busy, //TRUE：功能块正在执行
176         CommandAborted=>strMC_Home_P.x_CmdAborted, //TRUE：功能块被终止
177         Error=>strMC_Home_P.x_Error, //TRUE：功能块内部发生错误
178         ErrorID=>e_MC_Home_P_ErrorID);//错误码，具体参考CC_ERROR
179
180     _MC_TouchProbe_P(
181         Axis:=Axis_0, //指定使用的计数器名称
182         Execute:=strMC_TouchProbe_P.x_Execute, //上升沿触发，TRUE：使能功能块
183         Abort:=strMC_TouchProbe_P.x_Abort, //上升沿触发，TRUE：停止功能块
```

监视 1

表达式	应用	类型	值	准备值	执行点
Axis_0.fActPosition	Device.Application	LREAL	294		循环监测
Axis_0.nAxisState	Device.Application	SMC_AXIS_STATE	homing		循环监测

在正向移动至正限位端口后，限位获得高电平，出现正限位上升沿，电机以第二速度开始做反向运动；在出现正限位下降沿后，回原完成标志 strMC_Home_P.x_Done 亮起，轴位置为 0，本次回原结束

```
169
170 _MC_Home_P(
171     Axis:=Axis_0, //指定使用的轴名称
172     Execute:=strMC_Home_P.x_Execute, //上升沿触发，TRUE：使能功能块
173     Position:= 0, //回零位置，单位：unit
174     Done=>strMC_Home_P.x_Done, //TRUE：功能块执行完成
175     Busy=>strMC_Home_P.x_Busy, //TRUE：功能块正在执行
176     CommandAborted=>strMC_Home_P.x_CmdAborted, //TRUE：功能块被终止
177     Error=>strMC_Home_P.x_Error, //TRUE：功能块内部发生错误
178     ErrorID=>e_MC_Home_P_ErrorID);//错误码，具体参考CC_ERROR
179
180 _MC_TouchProbe_P(
181     Axis:=Axis_0, //指定使用的计数器名称
182     Execute:=strMC_TouchProbe_P.x_Execute, //上升沿触发，TRUE：使能功能块
183     Abort:=strMC_TouchProbe_P.x_Abort, //上升沿触发，TRUE：停止功能块
184     ProbeID:=usi_ProbeID, //所使用探针ID
185     TriggerEdge:=usi_TPTriggerEdge, // 0：上升沿 1：下降沿
```

监视 1

表达式	应用	类型	值	准备值	执行点
Axis_0.fActPosition	Device.Application	LREAL	0		循环监测
Axis_0.nAxisState	Device.Application	SMC_AXIS_STATE	standstill		循环监测