中型 PLC 编程指令手册

前言

资料简介

微秒控制中型 PLC（PC4MR、PC4M、PC5M）系列设备，是一套集成了高性能与丰富运动控制功能的中型可编程逻辑控制器，满足用户对复杂自动化设备、多轴协同控制、高速数据采集及复杂通信组网等需求，适用于高端装备制造、精密加工、过程控制等场景。
本手册介绍 PC4MR、PC4M、PC5M 系列产品编程应用时使用的基本指令及指令示例、复杂应用指令及指令示例等。

面向的读者

本手册面向以下读者对象：

- 电气工程师
- 软件工程师
- 系统工程师

初次使用

对于初次使用本产品的用户，应先认真阅读本手册。若对一些功能及性能方面有所疑惑，请咨询我公司的技术支持人员，以获得帮助，对正确使用本产品有利。

版本变更记录

资料版本　V1.0
归档日期　2026-09-16

日期	变更后版本	变更内容
2026 年 9 月	V1.0	初版发行。

关于手册获取

本手册不随产品发货，如需获取电子版 PDF 文件，可以通过以下方式获取：
登录深圳微秒控制技术官方网站（http://www.vmmore.com）下载 PDF 文件。

目录

1 指令概述

1.1 概述

本手册面向微秒中型 PLC，包括 PC4MR、PC4M 和 PC5M。本手册主要介绍微秒中型 PLC 所需要的指令、程序等内容，提供相关指令使用示范，指导客户快速使用。

1.2 标准数据类型

指令类别	数据类型	关键字	占用内存位数	取值范围
布尔类型	布尔	BOOL	8	FALSE 或 TRUE
整型	字节	BYTE	8	0~255
	字	WORD	16	0~65535
	双字	DWORD	32	0~4294967295
	长字	LWORD	64	0~(2^64-1)
	有符号短整型	SINT	8	-128~127
	无符号短整型	USINT	8	0~255
	整型	INT	16	-32768~32767
	无符号整型	UINT	16	0~65535
	双整型	DINT	32	-2147483648~2147483647
	无符号双整型	UDINT	32	0~4294967295
	长整型	LINT	64	-2^63~(2^63-1)
	无符号长整型	ULINT	64	0~(2^64-1)
浮点	单精度	REAL	32	1.175e-38~3.402e+38
	双精度	LREAL	64	2.225e-308~1.797e+308
字符串	字符串	STRING	8*N	

2 指令一览表

指令类别	名称	FB/FC	功能
比较	GT	FC	大于
	LT	FC	小于
	GE	FC	大于等于
	LE	FC	小于等于
	EQ	FC	等于
	NE	FC	不等于
选择	SEL	FC	二选一
	MUX	FC	多选一
	MAX	FC	取最大值
	MIN	FC	取最小值
	LIMIT	FC	极限值
计数器	CTD	FB	减计数器
	CTU	FB	加计数器
	CTUD	FB	增/减双向计数器
定时器	TP	FB	普通定时器
	TON	FB	通电延时定时器
	TOF	FB	断电延时定时器
	RTC	FB	实时时钟
位和逻辑字	AND	FC	与
	OR	FC	或
	NOT	FC	非
	XOR	FC	异或
	SR	FB	置位优先
	RS	FB	复位优先
	R_TRIG	FC	上升沿检测
	F_TRIG	FC	下降沿检测
数据移位	SHR	FC	按位右移
	SHL	FC	按位左移
	ROR	FC	循环右移
	ROL	FC	循环左移
类型转换	BOOL_TO_<TYPE>	FC	BOOL 类型转换
	BYTE_TO_<TYPE>	FC	字节类型转换
	WORD_TO_<TYPE>	FC	字类型转换
	DWORD_TO_<TYPE>	FC	双字类型转换
	INT_TO_<TYPE>	FC	整数型类型转换
	SINT_TO_<TYPE>	FC	短整型转换
	DINT_TO_<TYPE>	FC	长整型转换
	UDINT_TO_<TYPE>	FC	无符号长整型转换
	REAL_TO_<TYPE>	FC	实数类型转换
	STRING_TO_<TYPE>	FC	字符类型转换
	TIME_TO_<TYPE>	FC	时钟类型转换

	TOD_TO_<TYPE>	FC	时间类型转换
	DATE_TO_<TYPE>	FC	日期类型转换
	DT_TO_<TYPE>	FC	日期时间类型转换
数据处理	MOVE	FC	赋值
	HEXinASCII_TO_BYTE	FC	ASCII 转数字
	BYTE_TO_ HEXinASCII	FC	数字转 ASCII
	WORD_AS_STRING	FC	数字转字符串
	BYTE_TO_HEXSTRING	FC	BYTE 型转 16 进制字符串
	WORD_ TO_HEXSTRING	FC	WORD 型转 16 进制字符串
	DWORD_ TO_HEXSTRING	FC	DWORD 型转 16 进制字符串
数学运算	ADD	FC	加法
	SUB	FC	减法
	MUL	FC	乘法
	DIV	FC	除法
	MOD	FC	取余
	ABS	FC	绝对值
	SQRT	FC	平方根
	LN	FC	自然对数
	LOG	FC	常用对数
	EXP	FC	指数
	EXPT	FC	幂指数
	SIN	FC	正弦
	COS	FC	余弦
	TAN	FC	正切
	ASIN	FC	反正弦
	ACOS	FC	反余弦
	ATAN	FC	反正切
	SIZEOF	FC	数据类型大小
字符串指令	LEN	FC	获取字符串长度
	LEFT	FC	从左边取字符串
	RIGHT	FC	从右边取字符串
	MID	FC	从中间取字符串
	CONCAT	FC	字符串连接
	INSERT	FC	字符串插入
	DELETE	FC	字符串删除
	FIND	FC	字符串查找
	REPLACE	FC	字符串替换
地址操作	ADR	FC	取地址指令
	^	FC	取地址内容指令
	BITADR	FC	位地址
模拟量计算	PD	FB	比例微分控制
	PID	FB	比例积分微分控制
	PID_FIXCYCLE	FB	可手动设置循环周期的比例积分微分控制
BCD 码转换指令	BCD_TO_INT	FC	BCD 码转整型
	INT_TO_BCD	FC	整型转 BCD 码

	BCD_TO_BYTE	FC	BCD 码转字节
	BYTE_TO_BCD	FC	字节转 BCD 码
	BCD_TO_WORD	FC	BCD 码转字
	WORD_TO_BCD	FC	字转 BCD 码
	BCD_TO_DWORD	FC	BCD 码转双字
	DWORD_TO_BCD	FC	双字转 BCD 码
模拟波形	BLINK	FB	脉冲信号发生器
	GEN	FB	周期性信号发生器
	FREQ_MEASURE	FB	频率测量指令
单轴状态监控	MC_Power	FB	轴使能
	MC_Reset	FB	复位轴
	MC_ReadStatus	FB	读轴状态
	MC_ReadAxisError	FB	读轴错误
	MC_ReadParameter	FB	读参数
	MC_ReadBoolParameter	FB	读布尔参数
	MC_WriteParameter	FB	写参数
	MC_WriteBoolParameter	FB	写布尔参数
	MC_ReadActualPosition	FB	读实际位置
	MC_ReadActualVelocity	FB	读实际速度
	MC_ReadActualTorque	FB	读实际转矩
	MC_SetPosition	FB	设置位置
	SMC_ReadSetPosition	FB	读取设置位置
	SMC_ReadFBError	FB	读取历史错误信息
	SMC_ClearFBError	FB	清除历史错误信息
	SMC_ErrorString	FB	错误码到错误信息
单轴运动控制	MC_Home	FB	回零
	MC_MoveAbsolute	FB	绝对运动
	MC_MoveRelative	FB	相对运动
	MC_MoveVelocity	FB	定速运动
	MC_Stop	FB	停止
	MC_Halt	FB	暂停
	MC_Jog	FB	点动运动
	MC_MoveAdditive	FB	附加运动
	MC_MoveSuperImposed	FB	叠加运动
	MC_PositionProfile	FB	位置规划
	MC_VelocityProfile	FB	速度规划
	MC_AccelerationProfile	FB	加速度规划
	SMC_Homing	FB	上位机回原
	SMC_Inch	FB	寸动
同步运动功能	MC_GearIn	FB	电子齿轮耦合
	MC_GearInPos	FB	电子齿轮平滑耦合
	MC_GearOut	FB	电子齿轮脱离
	MC_CamTableSelect	FB	凸轮挺杆关联
	MC_CamIn	FB	电子凸轮关联
	MC_CamOut	FB	电子凸轮脱离
	SMC_GetTappetValue	FB	读取挺杆状态

	MC_MoveAdditive	FB	附加运动
	MC_MoveSuperImposed	FB	叠加运动
	MC_PositionProfile	FB	位置规划
自由 TCP 通信	TCP_Client	FB	创建 TCP 客户端通信服务
	TCP_Write	FB	TCP 通信数据发送
	TCP_Read	FB	TCP 通信数据接收
	TCP_Connection	FB	创建 TCP 连接，并连接到服务器
	TCP_Server	FB	创建 TCP 服务器端通信服务
自由 UDP 通信	UDP_Peer	FB	创建 UDP 通信连接
	UDP_Receive	FB	UDP 通信数据接收
	UDP_Send	FB	UDP 通信数据发送
EtherCAT 通信	ETC_CO_SdoReadDWord	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoRead4	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoRead	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoWrite_Dword	FB	EtherCAT 从站 SDO 写入
	ETC_CO_SdoWrite4	FB	EtherCAT 从站 SDO 写入
	ETC_CO_SdoWrite	FB	EtherCAT 从站 SDO 写入
	IoDrvEtherCAT_Diag	FB	EtherCAT 主站实例
	ETCSlave	FB	EtherCAT 从站实例
系统指令	VM_GetCPUInfo	FB	获取 CPU 信息
	VM_GetDateAndTime	FB	获取 PLC 的日期和时间
	VM_SetDateAndTime	FB	设置 PLC 的日期和时间
	VM_GetPLCVersion	FB	获取 PLC 版本号
	VM_GetNetInfo	FB	获取 PLC 网络信息
	VM_SetNetInfo	FB	设置 PLC 网络信息
	VM_GetIpAddress	FB	获取 PLC 的 IP 地址
	VM_SetIpAddress	FB	设置 PLC 的 IP 地址
	VM_GetPLCName	FB	获取 PLC 名称
文件操作指令	VM_FileWrite	FB	文件写入
	VM_FileWriteAppend	FB	文件追加写入
	VM_FileRead	FB	文件读取
	VM_FileCopy	FB	文件复制
	VM_FileRename	FB	文件重命名
	VM_FileDelete	FB	文件删除
	VM_CopyFromSDCard	FB	从 SD 卡拷贝文件
	VM_CopyToSDCard	FB	拷贝文件至 SD 卡
	VM_CopyFromUDisk	FB	从 U 盘拷贝文件
	VM_CopyToUDisk	FB	拷贝文件至 U 盘
Modbus 通信指令	ModbusMasterCtrl	FB	Modbus 主站控制器指令
	ModbusMasterRequest	FB	Modbus 主站请求指令
	ModbusMasterStatus	FB	Modbus 主站状态指令
	ModbusSlaveCtrl	FB	Modbus 从站控制器指令
自由串口通信指令	FreeProtocolCtrl	FB	串口自由协议配置指令
	FreeProtocolReceive	FB	串口自由协议接收指令
	FreeProtocolSend	FB	串口自由协议发送指令

3 基本指令

3.1 比较

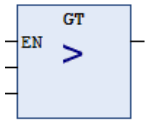
3.1.1 指令列表

指令类别	名称	FB/FC	功能
比较	GT	FC	大于
	LT	FC	小于
	GE	FC	大于等于
	LE	FC	小于等于
	EQ	FC	等于
	NE	FC	不等于

3.1.2 GT

判断两个输入值的大小，GT 指令等同于数学符号中的大于号，即 input1 和 input2 相比较，当 input1>input2 时， 输出 output 为 TRUE， 反之则为 FALSE。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
GT	大于	FC		>	-

相关变量

输入变量

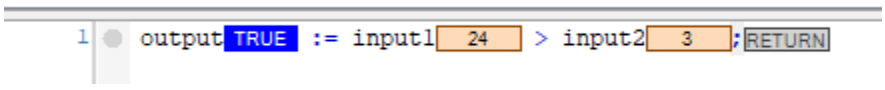
输入变量	名称	数据类型	有效范围	初始值	描述
Input1	输入 1	任何类型	遵照数据类型	0	比较值 1
Input2	输入 2	任何类型	遵照数据类型	0	比较值 2

输出变量

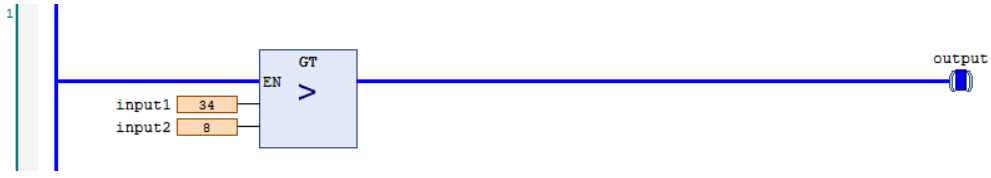
输出变量	名称	数据类型	有效范围	初始值	描述
output	返回值	BOOL	TRUE-FALSE	FALSE	比较结果

程序示例

ST:



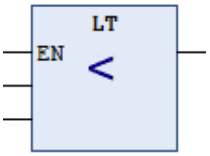
LD:



3.1.3 LT

判断两个输入值的大小，即 input1 和 input2 相比较，当 input1<input2 时，输出 output 为 TRUE，反之则为 FALSE。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LT	小于	FC		<	-

相关变量

输入变量

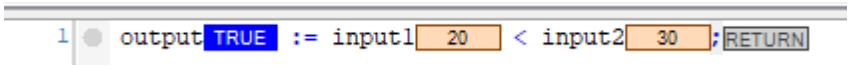
输入变量	名称	数据类型	有效范围	初始值	描述
Input1	输入 1	任何类型	遵照数据类型	0	比较值 1
Input2	输入 2	任何类型	遵照数据类型	0	比较值 2

输出变量

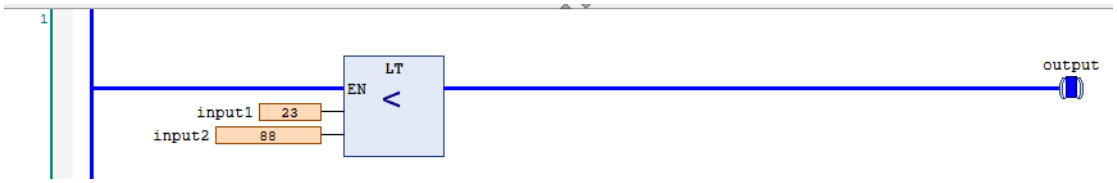
输出变量	名称	数据类型	有效范围	初始值	描述
output	返回值	BOOL	TRUE-FALSE	FALSE	比较结果

程序示例

ST:



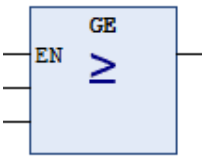
LD:



3.1.4 GE

判断两个输入值的大小，即 input1 和 input2 相比较，当 input1=input2 时，输出 output 为 TRUE，反之则为 FALSE。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
GE	等于	FC		>=	-

相关变量

输入变量

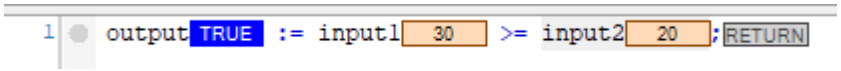
输入变量	名称	数据类型	有效范围	初始值	描述
Input1	输入 1	任何类型	遵照数据类型	0	比较值 1
Input2	输入 2	任何类型	遵照数据类型	0	比较值 2

输出变量

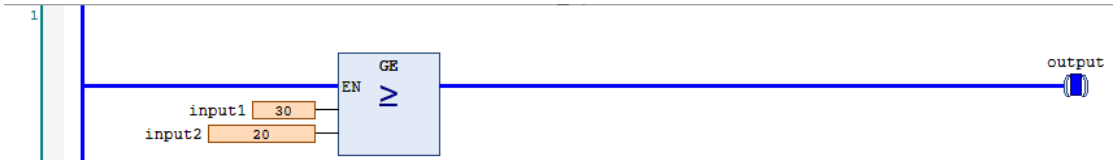
输出变量	名称	数据类型	有效范围	初始值	描述
output	返回值	BOOL	TRUE-FALSE	FALSE	比较结果

程序示例

ST:



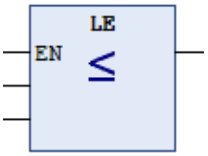
LD:



3.1.5 LE

判断两个输入值的大小，即 input1 和 input2 相比较，当 input1<=input2 时，输出 output 为 TRUE，反之则为 FALSE。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LE	小于等于	FC		<=	-

相关变量

输入变量

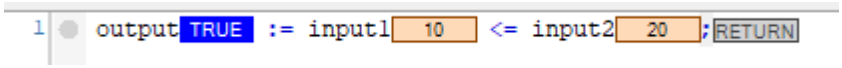
输入变量	名称	数据类型	有效范围	初始值	描述
Input1	输入 1	任何类型	遵照数据类型	0	比较值 1
Input2	输入 2	任何类型	遵照数据类型	0	比较值 2

输出变量

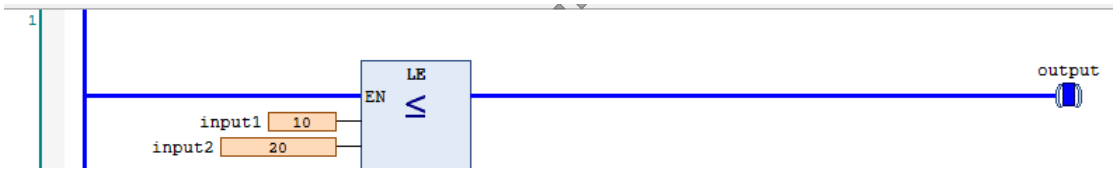
输出变量	名称	数据类型	有效范围	初始值	描述
output	返回值	BOOL	TRUE-FALSE	FALSE	比较结果

程序示例

ST:



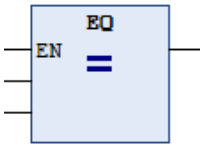
LD:



3.1.6 EQ

判断两个输入值的大小，即 input1 和 input2 相比较，当 input1=input2 时，输出 output 为 TRUE，反之则为 FALSE。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
EQ	等于	FC		=	-

相关变量

输入变量

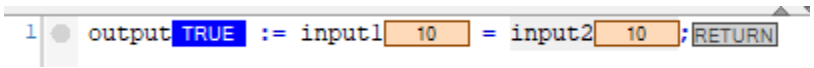
输入变量	名称	数据类型	有效范围	初始值	描述
Input1	输入 1	任何类型	遵照数据类型	0	比较值 1
Input2	输入 2	任何类型	遵照数据类型	0	比较值 2

输出变量

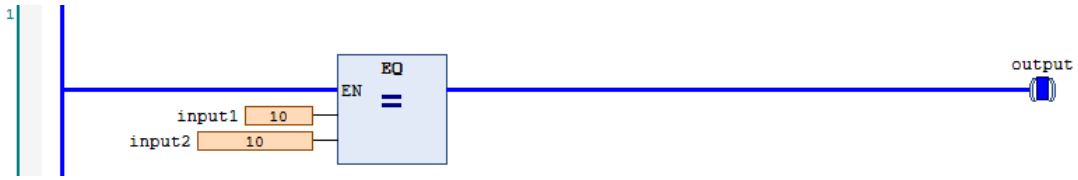
输出变量	名称	数据类型	有效范围	初始值	描述
output	返回值	BOOL	TRUE-FALSE	FALSE	比较结果

程序示例

ST:



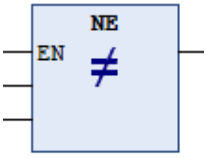
LD:



3.1.7 NE

判断两个输入值的大小，即 input1 和 input2 相比较，当 input1≠input2 时，输出 output 为 TRUE，反之则为 FALSE。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
NE	不等于	FC		<>	-

相关变量

输入变量

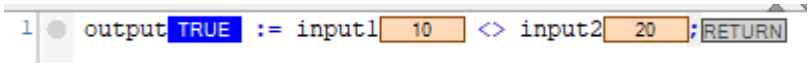
输入变量	名称	数据类型	有效范围	初始值	描述
Input1	输入 1	任何类型	遵照数据类型	0	比较值 1
Input2	输入 2	任何类型	遵照数据类型	0	比较值 2

输出变量

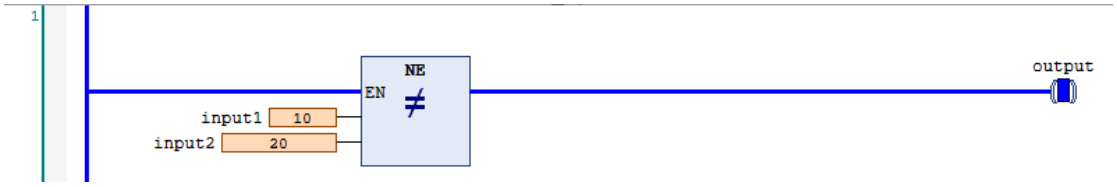
输出变量	名称	数据类型	有效范围	初始值	描述
output	返回值	BOOL	TRUE-FALSE	FALSE	比较结果

程序示例

ST:



LD:



3.2 选择

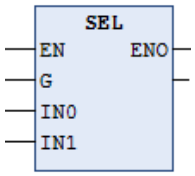
3.2.1 指令列表

指令类别	名称	FB/FC	功能
选择	SEL	FC	二选一
	MUX	FC	多选一
	MAX	FC	取最大值
	MIN	FC	取最小值
	LIMIT	FC	极限值

3.2.2 SEL

从两个操作数中选择一个。由 G 决定 IN0 还是 IN1 为输出。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SEL	二选一	FC		SEL	-

相关变量

输入变量

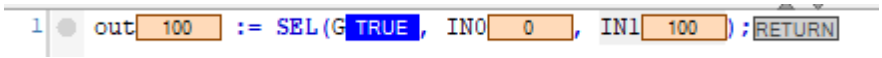
输入变量	名称	数据类型	有效范围	初始值	描述
G	输出控制	BOOL	TRUE-FALSE	FALSE	FALSE：输出 IN0 数据 TRUE：输出 IN1 数据
IN0	输入 0	任何类型	遵照数据类型	0	G 为 FALSE：输出 IN0 数据
IN1	输入 1	任何类型	遵照数据类型	0	G 为 TRUE：输出 IN1 数据

输出变量

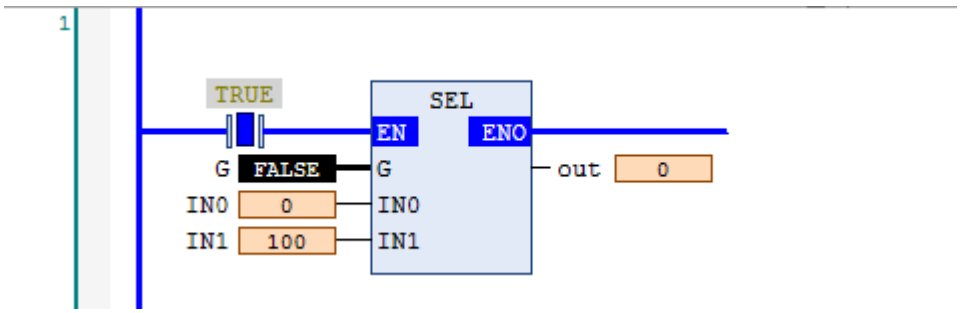
输出变量	名称	数据类型	有效范围	初始值	描述
output	选择结果	任何类型	遵照数据类型	0	选择结果

程序示例

ST：



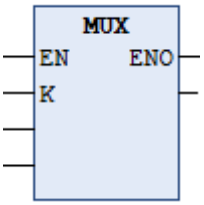
LD：



3.2.3 MUX

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

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MUX	多选一	FC		MUX	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
K	选择位	任何类型	遵照数据类型	FALSE	选择的数字位
IN1	比较数 1	任何类型	遵照数据类型	0	数据 1
INn	比较数 n	任何类型	遵照数据类型	0	数据 n

输出变量

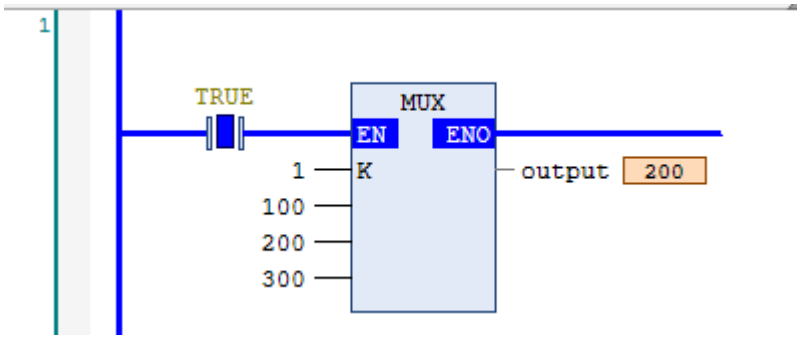
输出变量	名称	数据类型	有效范围	初始值	描述
output	选择结果	任何类型	遵照数据类型	0	选择结果

程序示例

ST:

```
1  output 200 := MUX(1, 100, 200, 300);RETURN
```

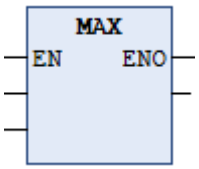
LD:



3.2.4 MAX

MAX 取多个输入值中的最大值，并将最大值从右侧输出。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MAX	取最大值	FC		MAX	-

相关变量

输入变量

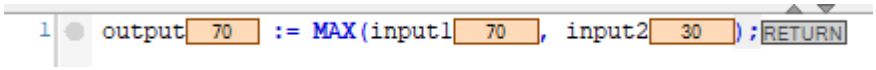
输入变量	名称	数据类型	有效范围	初始值	描述
IN1	比较数 1	任何类型	遵照数据类型	0	数据 0
INn	比较数 n	任何类型	遵照数据类型	0	数据 n

输出变量

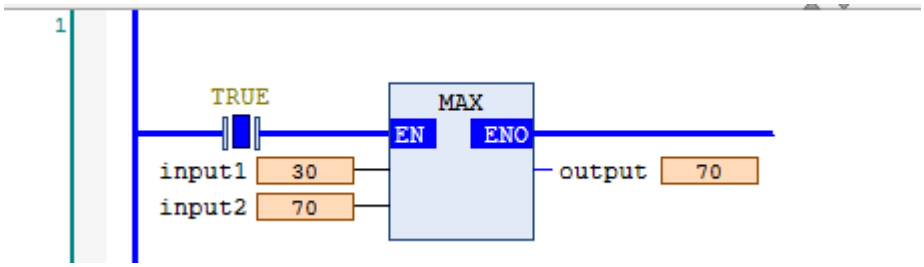
输出变量	名称	数据类型	有效范围	初始值	描述
output	输出变量	任何类型	遵照数据类型	0	取输入数中的最大值

程序示例

ST:



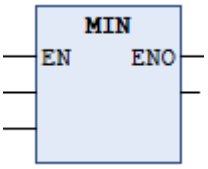
LD:



3.2.5 MIN

MIN 取多个输入值中的最小值，并将最小值从右侧输出。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MIN	取最小值	FC		MIN	-

相关变量

输入变量

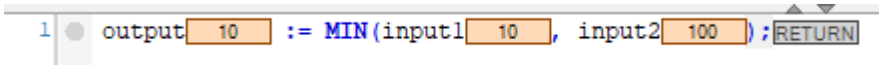
输入变量	名称	数据类型	有效范围	初始值	描述
IN1	比较数 1	任何类型	遵照数据类型	0	数据 0
INn	比较数 n	任何类型	遵照数据类型	0	数据 n

输出变量

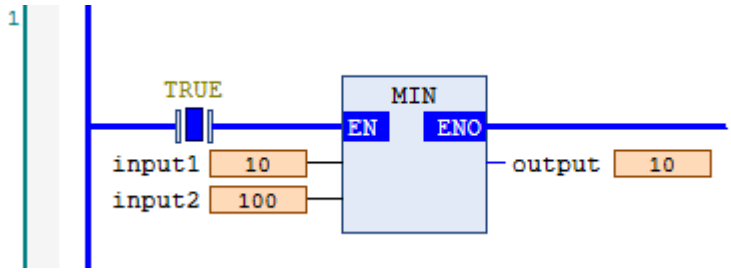
输出变量	名称	数据类型	有效范围	初始值	描述
output	输出变量	任何类型	遵照数据类型	0	取输入数中的最小值

程序示例

ST:



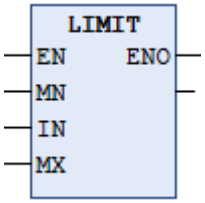
LD:



3.2.6 LIMIT

取极限值指令，MX 是结果的上限值并且 MN 是结果的下限值。如果 IN 得值超过了 MX 的上限值，则输出运算结果为 MX。如果 IN 的值小于 MN 的下限值，则输出运算结果为 MN。 当输入 IN 的值在 MN 和 MX 范围之内的时候，则输出的值为输入的 IN 的值。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LIMIT	取极限值	FC		LIMIT	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
MN	下限值	任何类型	遵照数据类型	0	最小值
IN	输入变量	任何类型	遵照数据类型	0	输入数据
MX	上限值	任何类型	遵照数据类型	0	最大值

输出变量

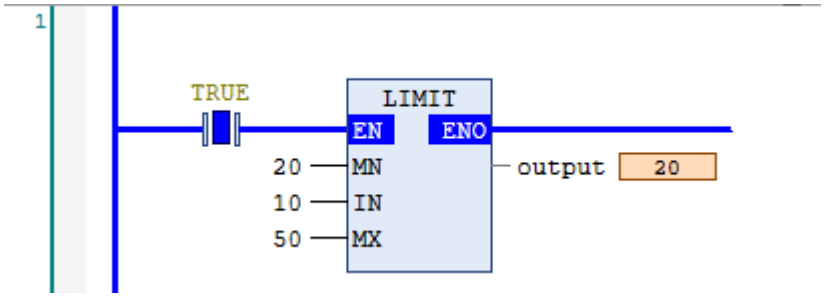
输出变量	名称	数据类型	有效范围	初始值	描述
output	输出变量	任何类型	遵照数据类型	0	输出结果

程序示例

ST:

```
1  output 20 := LIMIT(20, 10, 50);RETURN
```

LD:



3.3 计数

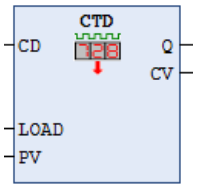
3.3.1 指令列表

指令类别	名称	FB/FC	功能
计数器	CTD	FB	减计数器
	CTU	FB	加计数器
	CTUD	FB	增/减双向计数器

3.3.2 CTD

该功能块用作一个递减计数器。通过预设值代入变量，每次检测到上升沿便减 1，到 0 时输出状态 true。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
CTD	减计数器	FB		CTD_0(CD:= , LOAD:= , PV:= , Q=> , CV=>);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
CD	计数信号输入	BOOL	TRUE-FALSE	FALSE	上升沿：CV 计数值减 1
LOAD	加载信号	BOOL	TRUE-FALSE	FALSE	TRUE：将 CV 设为初始值 PV
PV	递减起始值	WORD	0~65535	0	递减的起始值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Q	状态输出	BOOL	TRUE-FALSE	FALSE	CV=0 时为 TRUE
CV	当前计数值	WORD	0~65535	0	当前计数值

功能说明

1）需要将加载信号 LOAD 从“FALSE”变为“TRUE”，CV 将设置为 PV 给定的起始值、 输出端 Q 由初始“TRUE”变为“FALSE”，LOAD 再变为状态“FALSE”后功能块才能生效。

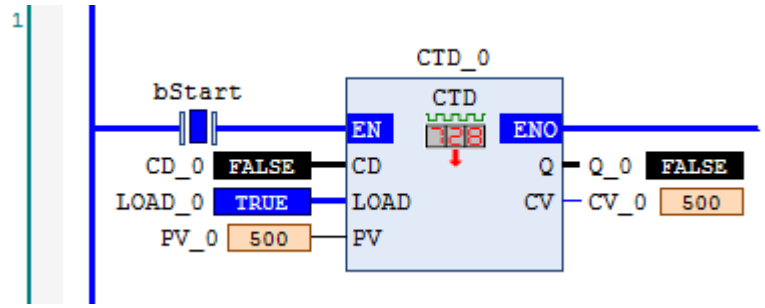
2）当减计数器输入端的 CD 信号从“FALSE”变为状态“TRUE”时，当前计数值减 1，并在输出端 CV 显示当前值。CV 的值递减至 0 时，则 Q 将被设置为 TRUE，计数值将不再会减少。

程序示例

ST:

```
1  CTD_0(
2      CD FALSE := CD_0 FALSE,
3      LOAD TRUE := LOAD_0 TRUE,
4      PV 500 := PV_0 500,
5      Q FALSE => Q_0 FALSE,
6      CV 500 => CV_0 500 );RETURN
```

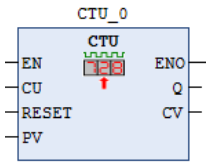
LD:


1
CTD_0
CTD
bStart
EN
ENO
CD_0
FALSE
CD
Q
Q_0
FALSE
LOAD_0
TRUE
LOAD
CV
CV_0
500
PV_0
500
PV

3.3.3 CTU

该功能块用作一个递增计数器。通过预设值代入变量，每次检测到上升沿便加 1，到预设值时输出状态 true。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
CTU	增计数器	FB		CTU_0(CU:= , RESET:= , PV:= , Q=> , CV=>);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
CU	计数信号输入	BOOL	TRUE-FALSE	FALSE	上升沿：CV 计数值加 1
RESET	复位输入	BOOL	TRUE-FALSE	FALSE	TRUE：CV 复位为 0
PV	设定计数上限	WORD	0~65535	0	设定计数的上限

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Q	状态输出	BOOL	TRUE-FALSE	FALSE	CV>=PV 时为 TRUE
CV	当前计数值	WORD	0~65535	0	当前计数值

功能说明

1）当 CU 的信号从“FALSE”变为“TRUE”时，当前计算值加 1，并通过输出端 CV 进行显示，当计数达到上限 65535 后，计数器将不会再增加。

2）当 RESET 的信号状态为“TRUE”时，计数器的 CV 初始化为 0、Q 为“FALSE”，上升沿对 CU 就不再起作用。

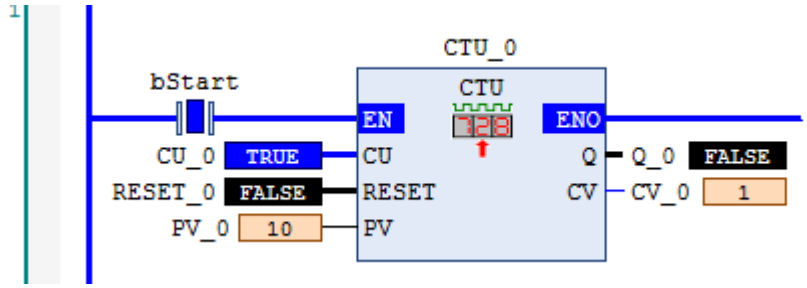
3）当 CV 值大于或等于 PV 时，输出端 Q 为“TRUE”。此时 CV 仍可继续累加，输出端 Q 继续为输出“TRUE”。

程序示例

ST：

```
CTU_0(
    CU TRUE := CU_0 TRUE,
    RESET FALSE := RESET_0 FALSE,
    PV 10 := PV_0 10,
    Q FALSE => Q_0 FALSE,
    CV 1 => CV_0 1 );RETURN
```

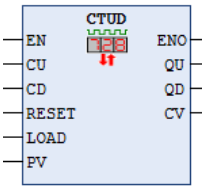
LD：


1
CTU_0
CTU
bStart
EN
ENO
CU_0
TRUE
CU
Q
Q_0
FALSE
RESET_0
FALSE
RESET
CV
CV_0
1
PV_0
10
PV

3.3.4 CTUD

根据加法计数器输入及减法计数器输入进行加减运算的计数器。预设值、计数值的数据类型为 WORD。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
CTUD	增/减双向计数器	FB		CTUD_0(CU:= , CD:= , RESET:= , LOAD:= , PV:= , QU=> , QD=> , CV=>);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
CU	增计数输入	BOOL	TRUE-FALSE	FALSE	上升沿：CV 计数值加 1
CD	减计数输入	BOOL	TRUE-FALSE	FALSE	上升沿：CV 计数值减 1
RESET	复位输入	BOOL	TRUE-FALSE	FALSE	TRUE：CV 复位为 0
LOAD	加载信号	BOOL	TRUE-FALSE	FALSE	TRUE：CV 设为初始值 PV
PV	设定计数上限	WORD	0~65535	0	设定的计数上限

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
QU	增计数状态输出	BOOL	TRUE-FALSE	FALSE	CV>=PV 时为 TRUE
QD	减计数状态输出	BOOL	TRUE-FALSE	FALSE	CV=0 时为 TRUE
CV	当前计数值	WORD	0~65535	0	当前计数值

功能说明

1）当加计数输入端的 CU 信号从“FALSE”变为状态“TRUE”时，当前计数值加 1，并在输出 CV 上显示。

2）当减计数输入端的 CD 的信号从“FALSE”变为状态“TRUE”时（需要提前将加载信号 LOAD 从“FALSE”变为“TRUE”、再变为状态“FALSE”后功能块才能生效），当前计数值减 1，并在输出端 CV 上显示。如果两个输入端都是上升沿，当前计数值将保持不变。

3）当计数值达到上限值 65535 后，加计数输入端 CU 的上升沿不再起作用。因此即使加计数输入端 CU 出现上升沿，其数值也不会增加。同理，当计数值达到下极限值 0 后，减计数输入端 CD 也不会在起作用，因此，即使减计数输入端 CD 出现上升沿，计数值也不会减少。

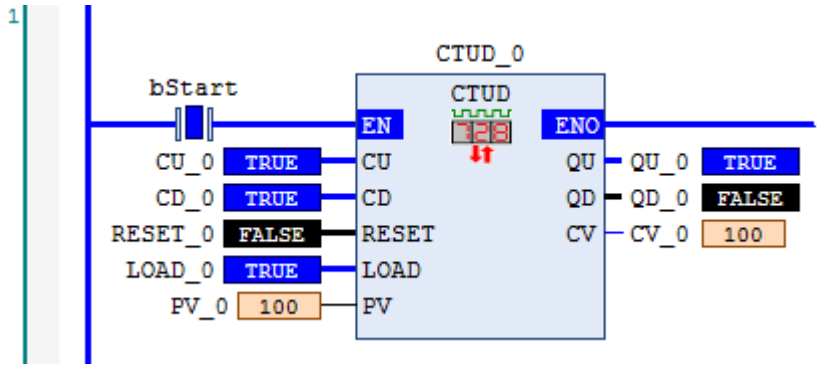
4）当 CV 值大于或等于 PV 值时，输出 QU 为“TRUE”。当 CV 值等于 0 时，输出 QD 为“TRUE”。

程序示例

ST:

```
CTUD_0(
    CU TRUE := CU_0 TRUE,
    CD TRUE := CD_0 TRUE,
    RESET FALSE := RESET_0 FALSE,
    LOAD TRUE := LOAD_0 TRUE,
    PV 100 := PV_0 100,
    QU TRUE => QU_0 TRUE,
    QD FALSE => QD_0 FALSE,
    CV 100 => CV_0 100);RETURN
```

LD:



3.4 定时器

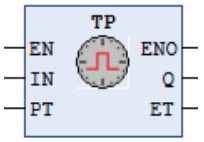
3.4.1 指令列表

指令类别	名称	FB/FC	功能
定时器	TP	FB	普通定时器
	TON	FB	通电延时定时器
	TOF	FB	断电延时定时器
	RTC	FB	实时时钟

3.4.2 TP

输入计时时间， 输出当前计时时间与计时状态信号。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
TP	定时器	FB		TP_0(IN:= , PT:= , Q=> , ET=>);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入信号	BOOL	TRUE-FALSE	FALSE	FALSE→TRUE 开始计时
PT	设定时间	TIME	0-4294967295	0	设定时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Q	计时完成	BOOL	TRUE-FALSE	FALSE	TRUE→FALSE 计时完成
ET	当前计时	TIME	0-4294967295	0	设定时间

功能说明

1）一旦 IN 变为“TRUE”，则 Q 为“TRUE”，时间将开始在 ET 中以毫秒为单位计数，直到其值等于 PT 时 Q 为“FALSE”。

2）当 PT 时间到时，ET 会保持定时时间直到 IN 变为“FALSE”时恢复为 T#0ms。如果在达到 PT 定时时间之前输入 IN 已经变成“FALSE”，则 ET 计时到设置的 PT 时间后会立即复位为 T#0ms。

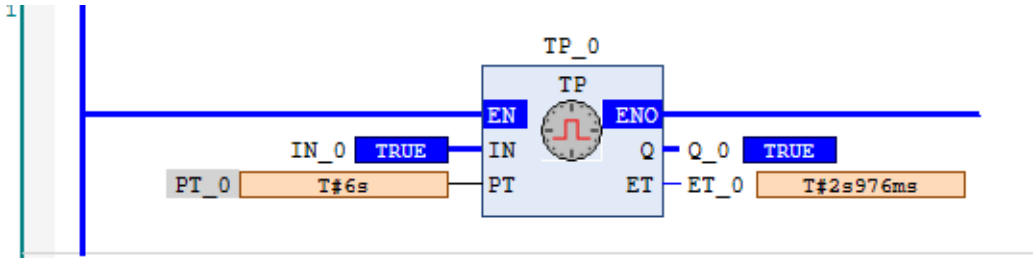
3）设置 IN 为 FALSE 且 PT=T#0ms 时即可复位该定时器。

程序示例

ST:

```
TP_0(
    IN TRUE := IN_0 TRUE,
    PT T#6s := PT_0 T#6s,
    Q TRUE => Q_0 TRUE,
    ET T#3s200ms => ET_0 T#3s200ms); RETURN
```

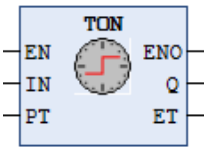
LD:



3.4.3 TON

通电延时一段时间之后输出计时完成信号。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
TON	通电延时定时器	FB		TON_0(IN:= , PT:= , Q=> , ET=>);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入信号	BOOL	TRUE-FALSE	FALSE	FALSE→TRUE 开始计时
PT	设定时间	TIME	0-4294967295	0	设定时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Q	计时完成	BOOL	TRUE-FALSE	FALSE	FALSE→TRUE 计时完成
ET	当前计时	TIME	0-4294967295	0	设定时间

功能说明

1）输入端 IN 从“FALSE”变为“TRUE”时，定时器启动。当输入端的信号 IN 始终维持在“TRUE”且 ET 到达定时时间 PT 时，其输出端 Q 为“TRUE”。

2）输出端 ET 提供定时时间，定时从 T#0s 开始，到设置的 PT 时间结束定时。ET 到达 PT 时将会保持定时时间直到 IN 变为“FALSE”为止。

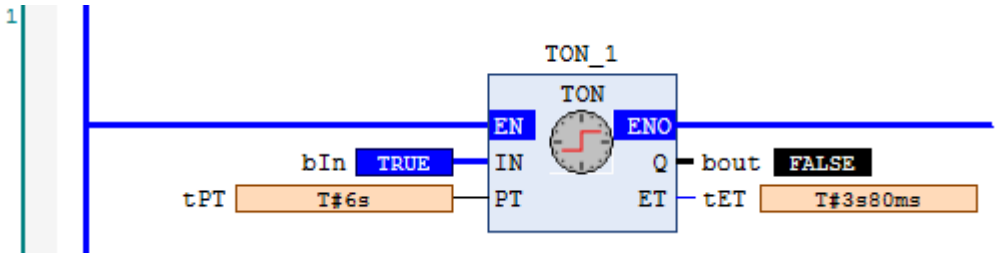
3）如果在 ET 达到 PT 定时时间之前，输入 IN 变为“FALSE”，输出 ET 则立即变为 T#0s。所以为了重启定时器，可以将 IN 设为“FALSE”。

程序示例

ST:

```
TON_0(
      IN TRUE := bIN TRUE,
      PT T#6s := tPT T#6s,
      Q FALSE => bout FALSE,
      ET T#3s464ms => tET T#3s464ms);RETURN
```

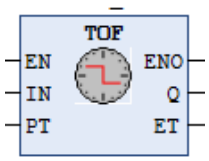
LD:


1
TON_1
TON
EN
ENO
bIn
TRUE
IN
Q
bout
FALSE
tPT
T#6s
PT
ET
tET
T#3s80ms

3.4.4 TOF

断电延时一段时间之后输出计时完成信号。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
TOF	断电延时定时器	FB		TOF_0(IN:= , PT:= , Q=> , ET=>);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入信号	BOOL	TRUE-FALSE	FALSE	TRUE→FALSE 开始计时
PT	计时时间	TIME	0-4294967295	0	设定时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Q	计时完成	BOOL	TRUE-FALSE	FALSE	FALSE→TRUE 计时完成
ET	当前计时	TIME	0-4294967295	0	设定时间

功能说明

1）输入端 IN 从“FALSE”变为“TRUE”时，定时器的 Q 输出信号为“TRUE”，当启动输入端 IN 变为“FALSE”时，定时器启动，定时器计时过程中其输出 Q 一直为“TRUE”，当到达定时时间 PT 时，输出端 Q 复位。

2）输出端 ET 提供定时时间，定时从 T#0s 开始，到设置的 PT 时间结束定时。ET 到达 PT 时将会保持定时时间直到 IN 变为“TRUE”为止。

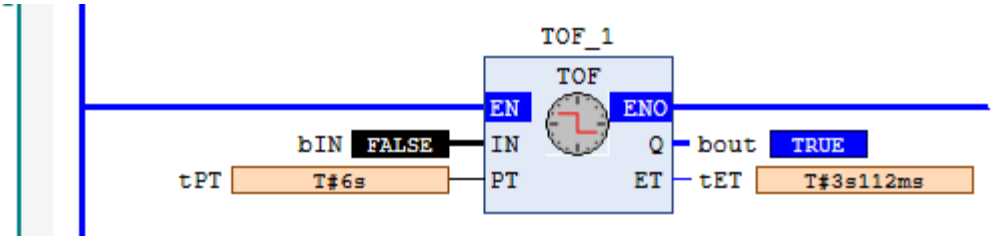
3）如果在 ET 达到 PT 定时时间之前，输入 IN 变为“TRUE”，输出 ET 则立即变为 T#0s。所以为了重启定时器，可以将 IN 设为“TRUE”。

程序示例

ST:

```
1  TOF_0(
2        IN:= bIN,
3        PT:= tPT,
4        Q=> bout,
5        ET=> tET);RETURN
```

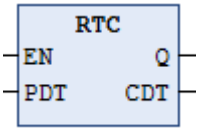
LD:



3.4.5 RTC

时钟功能，从设定时间开始计时。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
RTC	实时时钟	FB		RTC_0(EN:= , PDT:= , Q=> , CDT=>);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
EN	启动使能	BOOL	TRUE-FALSE	FALSE	上升沿：CDT 设置为 PDT，CDT 开始上升； 下降沿：CDT 设置为 DT#1970-01-01-00:00:00
PDT	设置将要启动的时间和日期	DATE_AND_TIME	(DT#1970-1-1-0:0:0) -(2106-2-7-6:28:15)	DT#1970-1-1-0:0:0	预设日期和时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Q	状态输出	BOOL	TRUE-FALSE	FALSE	CDT 在计数时为 TRUE
CDT	当前计数时间和日期状态	DATE_AND_TIME	(DT#1970-1-1-0:0:0) -(2106-2-7-6:28:15)	DT#1970-1-1-0:0:0	当前计数时间和日期状

功能说明

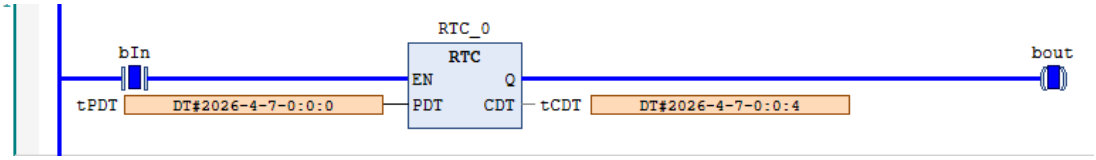
一旦 EN 为“TRUE”，CDT 会被设置为 PDT 给予的时间，并且将会以秒进行计数。一旦 EN 被复位为“FALSE”，输出变量 Q 为“FALSE”，CDT 将会被复位为初始值 DT#1970-01-01-00:00:00。

程序示例

ST:

```
RTC_0(
      EN TRUE := bIN TRUE,
      PDT DT#2026-4-7-0:0:0 := tPDT DT#2026-4-7-0:0:0,
      Q TRUE => bout TRUE,
      CDT DT#2026-4-7-0:0:4 => tCDT DT#2026-4-7-0:0:4 );RETURN
```

LD：


RTC_0
RTC
bIn
EN
Q
bout
tPDT
DT#2026-4-7-0:0:0
PDT
CDT
tCDT
DT#2026-4-7-0:0:4

3.5 位和逻辑字

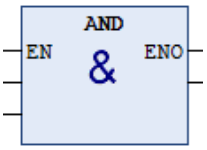
3.5.1 指令列表

指令类别	名称	FB/FC	功能
位和逻辑字	AND	FC	与
	OR	FC	或
	NOT	FC	非
	XOR	FC	异或
	SR	FB	置位优先
	RS	FB	复位优先
	R_TRIG	FC	上升沿检测
	F_TRIG	FC	下降沿检测

3.5.2 AND

与运算，可以分为：按位逻辑运算以及布尔逻辑运算。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
AND	与指令	FC		AND	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入 1	-	-	0	数据 1
IN2	输入 2	-	-	0	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出结果	-	-	0	输出结果

功能说明

1）按位“与”：按位“与”运算指令是比较两个整数的相应位。当两个数的相应位都是“1”时，返回相应的结果位是“1”；当两个整数的相应位都是“0”或者其中一个位是“0”时，则返回相应的结果位是“0”。

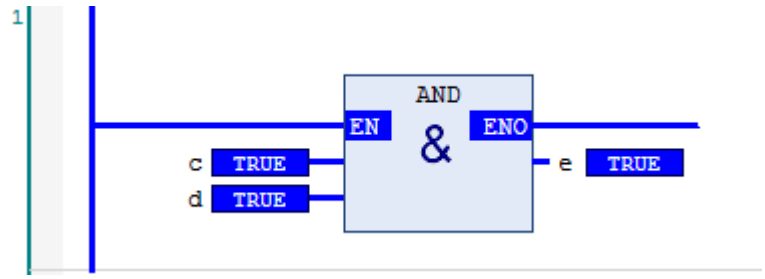
2）布尔“与”：布尔“与”运算用于计算连个布尔表达式的“与”结果。当两个布尔表达式的结果都为真时，则返回为真，其中只要有一个为假，则返回为假。

程序示例

ST：

```
1  e TRUE := c TRUE AND d TRUE ;RETURN
```

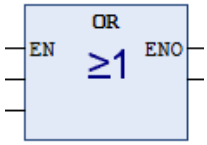
LD：



3.5.3 OR

或运算，可以分为：按位逻辑运算以及布尔逻辑运算。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
OR	或指令	FC		OR	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入 1	-	-	0	数据 1
IN2	输入 2	-	-	0	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出结果	-	-	0	输出结果

功能说明

1）按位“或”：按位“或”运算指令是比较两个整数的相应位。当两个整数的相应位有一个是“1”时，返回相应的结果位为“1”。当两个整数的相应位都是“0”时，则返回相应的结果位是“0”。

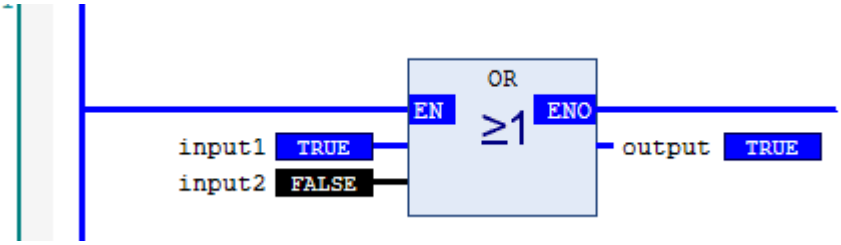
2）布尔“或”：布尔“或”运算指令是计算两个布尔表达式的“或”结果。当两个布尔表达式中有一个表达式返回为真时，则结果位真；当两个表达式的结果都是假时，则结果为假。

程序示例

ST：

```
output TRUE := input1 TRUE OR input2 FALSE ;RETURN
```

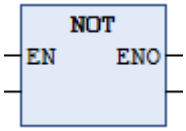
LD：



3.5.4 NOT

非运算，可以分为：按位逻辑运算以及布尔逻辑运算。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
NOT	非指令	FC		NOT	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入 1	-	-	0	数据 1

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出结果	-	-	0	输出结果

功能说明

1）按位“非”：按逻辑进行取反，将当前的值由“0”变为“1”，或由“1”变为“0”。按位“非”运算指令是将变量或常量逐一取非。

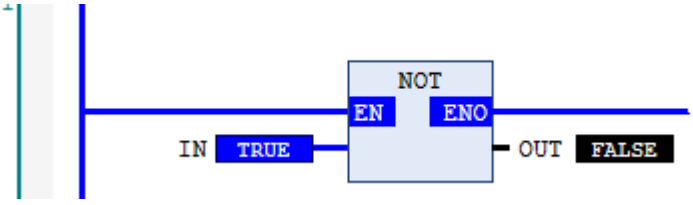
2）布尔“非”：布尔“非”运算指令用于计算单个布尔表达式的结果。当输入为真时，结果为假；当输入为假时，结果为真。

程序示例

ST：

```
output := NOT IN;RETURN
```

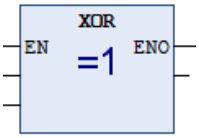
LD：



3.5.5 XOR

按位“异或”运算指令比较两个整数的相应位。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
XOR	异或指令	FC		XOR	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入 1	-	-	0	数据 1
IN2	输入 2	-	-	0	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出结果	-	-	0	输出结果

功能说明

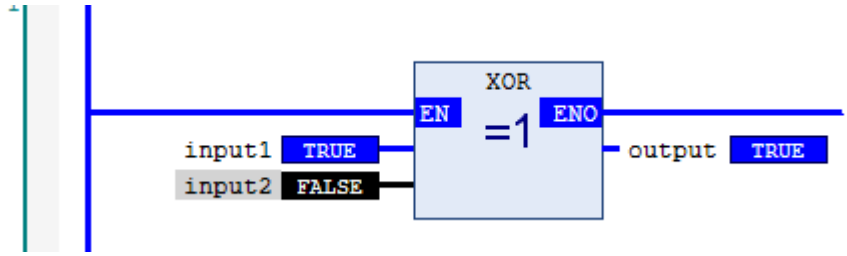
当两个整数的相应位一个是“1”而另一个是“0”时，返回相应的结果位是“1”。当两个整数的相应位都是“1”或者都是“0”时，则返回相应的结果位是“0”。

程序示例

ST:

```
output TRUE := input1 TRUE XOR input2 FALSE ;RETURN
```

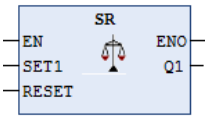
LD:



3.5.6 SR

置位输入信号和复位输入信号均有效时，置位输入信号优先。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SR	置位优先指令	FB		SR_0(SET1:= , RESET:= , Q1=>);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
SET1	置位	BOOL	TRUE-FALSE	FALSE	数据 1
RESET	复位	BOOL	TRUE-FALSE	FALSE	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Q1	输出值	BOOL	TRUE-FALSE	FALSE	置位优先输出

功能说明

1）逻辑关系：Q1=(NOT RESET AND Q1) OR SET1，其中 SET1 为置位信号，RESET 为复位信号。

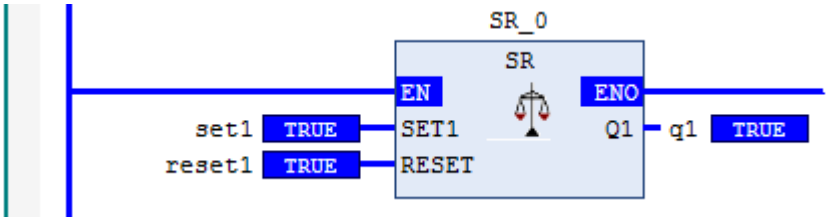
2）当 SET1 为“1”时，不论 RESET 是否为“1”，Q1 输出都为“1”；当 SET1 为“0”时，如果 Q1 输出为“1”，一旦 RESET 为“1”，Q1 输出立刻复位为“0”。如果 Q1 输出为“0”，不论 RESET 为“1”或者“0”，Q1 输出保持为“0”。

程序示例

ST:

```
SR_0(
    SET1 TRUE := set1 TRUE,
    RESET TRUE := reset1 TRUE,
    Q1 TRUE => q1 TRUE);
```

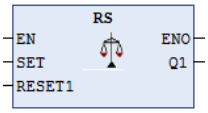
LD:



3.5.7 RS

置位输入信号和复位输入信号均有效时，复位输入信号优先。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
RS	复位优先指令	FB		RS_0(SET:= , RESET1:= , Q1=>);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
SET	置位	BOOL	TRUE-FALSE	FALSE	数据 1
RESET1	复位	BOOL	TRUE-FALSE	FALSE	数据 2

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Q1	输出值	BOOL	TRUE-FALSE	FALSE	复位优先输出

功能说明

1）逻辑关系：Q1=NOT RESET1 AND (Q1 OR SET)，其中 SET 为置位信号，RESET1 为复位信号。

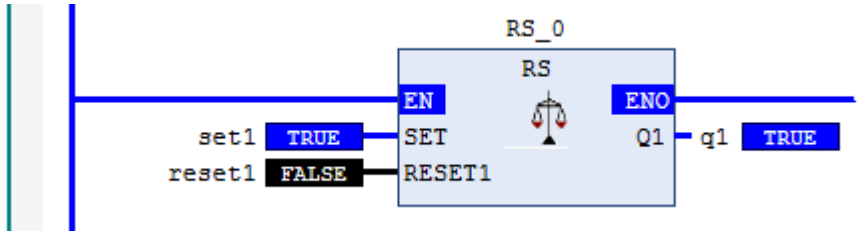
2）当 RESET1 为“1”时，不论 SET 是否为“1”，Q1 输出都为“0”；当 RESET1 为“0”时，如果 Q1 输出为“0”，一旦 SET 为“1”，Q1 输出立刻置位为“1”。如果 Q1 输出为“1”，不论 SET 为“1”或者“0”，Q1 输出保持为“1”。

程序示例

ST:

```
RS_0(
    SET TRUE := set1 TRUE,
    RESET1 FALSE := reset1 FALSE,
    Q1 TRUE => q1 TRUE);
```

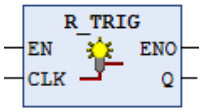
LD:



3.5.8 R_TRIG

在每个扫描周期中，把信号状态和它再前一个扫描周期的状态进行比较，若不同，则表明有一个跳变沿。用于检测上升沿。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
R_TRIG	上升沿检测	FB		R_TRIG_0(CLK:= , Q=>);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
CLK	输入信号	BOOL	TRUE-FALSE	FALSE	信号上升沿（由 0->1）有效

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Q	输出信号	BOOL	TRUE-FALSE	FALSE	检测上升沿

功能说明

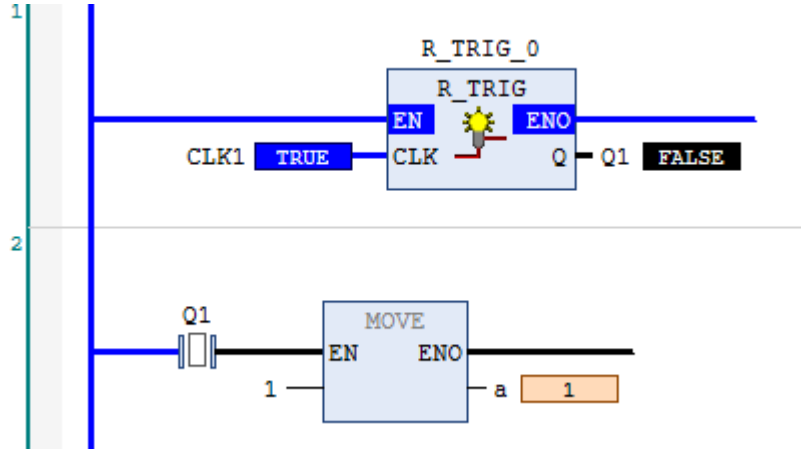
当 CLK 从“0”变为“1”时，该上升沿检测器开始启动，Q 输出先为“1”然后输出变为“0”，持续一个 PLC 运算周期；如果 CLK 持续保持为“1”或者“0”，Q 输出一直保持为“0”。

程序示例

ST:

```
R_TRIG_0(
      CLK TRUE := CLK1 TRUE,
      Q FALSE => Q1 FALSE);
IF Q1 FALSE THEN
    a  1  := 1;
END_IF
```

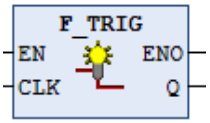
LD:



3.5.9 F_TRIG

在每个扫描周期中，把信号状态和它再前一个扫描周期的状态进行比较，若不同，则表明有一个跳变沿。用于检测下降沿。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
F_TRIG	下降沿检测	FB		F_TRIG_0(CLK:= , Q=>);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
CLK	输入信号	BOOL	TRUE-FALSE	FALSE	信号下降沿（由 1->0）有效

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Q	输出信号	BOOL	TRUE-FALSE	FALSE	检测下降沿

功能说明

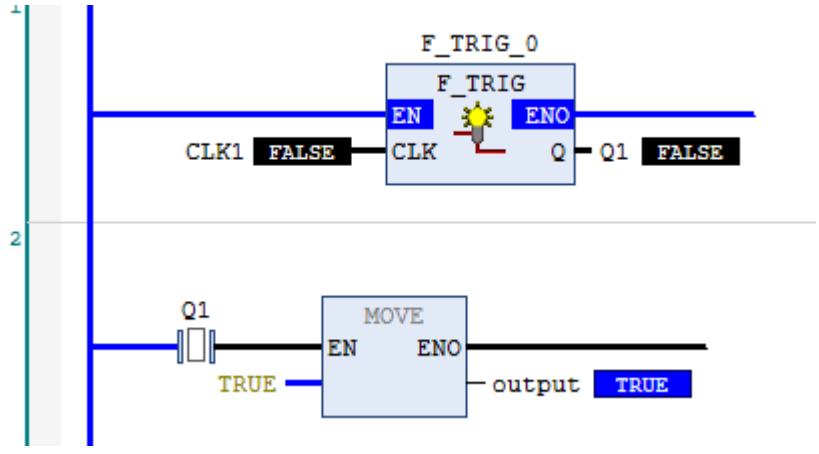
当 CLK 从“1”变为“0”时，该下降沿检测器开始启动，Q 输出先为“1”然后输出变为“0”，持续一个 PLC 运算周期；如果 CLK 持续保持为“1”或者“0”，Q 输出一直保持为“0”。

程序示例

ST:

```
F_TRIG_0(
     CLK FALSE := CLK1 FALSE,
     Q FALSE => Q1 FALSE );
IF Q1 FALSE THEN
    output TRUE := TRUE;
END_IF
```

LD:



3.6 数据移位

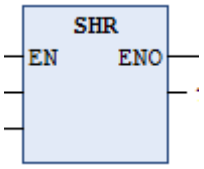
3.6.1 指令列表

指令类别	名称	FB/FC	功能
数据移位指令	SHR	FC	按位右移
	SHL	FC	按位左移
	ROR	FC	循环右移
	ROL	FC	循环左移

3.6.2 SHR

对操作数进行按位右移，右边移出位不做处理，左边空位自动补 0。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SHR	按位右移	FC		SHR(IN,N)	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	整型	遵照数据类型	0	数据来源
N	移位位数	整型	遵照数据类型	0	右移位位数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	偏移后数据	整型	遵照数据类型	0	偏移后的数据

功能说明

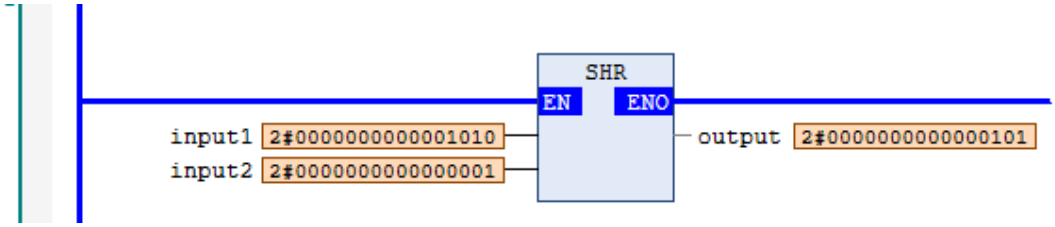
指令可以将输入 IN 中的数据右移 N 位，输出结果赋值至 OUT，二进制数右移一位相当于将原数除以 2，如果 N 设置值大于 IN 的数据类型宽度时，IN 值将填补为 0。

程序示例

ST:

```
output 2#0000000000000101 := SHR(input1 2#0000000000001010, input2 2#0000000000000001);
```

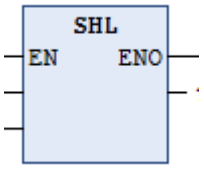
LD:



3.6.3 SHL

对操作数进行按位左移，左边移出位不做处理，右边空位自动补 0。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SHL	按位左移	FC		SHL(IN,N)	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	整型	遵照数据类型	0	数据来源
N	移位位数	整型	遵照数据类型	0	左移位位数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	偏移后数据	整型	遵照数据类型	0	偏移后的数据

功能说明

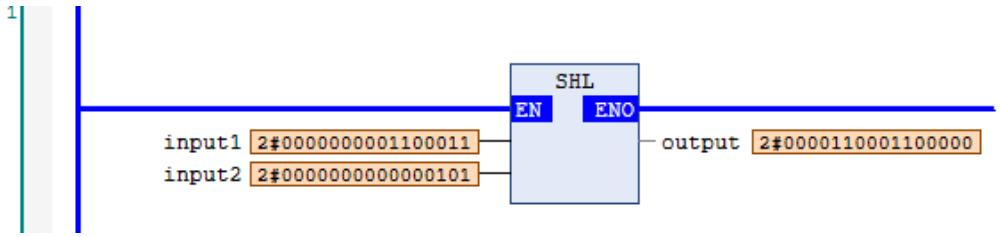
指令可以将输入 IN 中的数据左移 N 位，输出结果赋值至 OUT，二进制数左移一位相当于将原数乘以 2，如果 N 设置值大于 IN 的数据类型宽度时，IN 值将填补为 0。

程序示例

ST:

```
output 2#0000110001100000 := SHL(input1 2#0000000001100011, input2 2#0000000000000101);
```

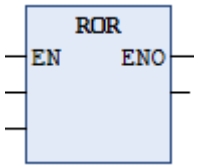
LD:



3.6.4 ROR

对操作数进行循环右移，右边移出的位直接补充到左边最高位。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ROR	循环右移	FC		ROR(IN,N)	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	整型	遵照数据类型	0	数据来源
N	移位位数	整型	遵照数据类型	0	右移位位数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	偏移后数据	整型	遵照数据类型	0	偏移后的数据

功能说明

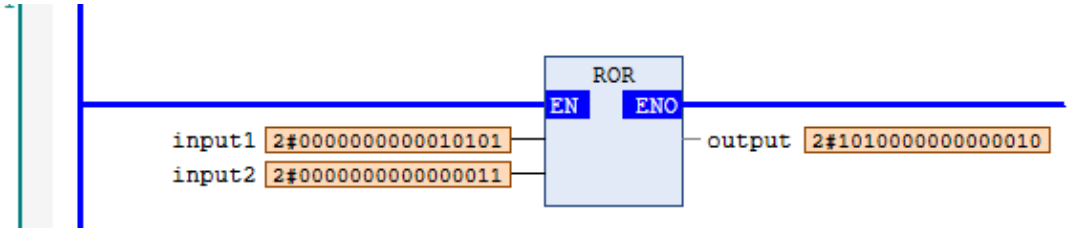
使用该指令可以将输 IN 中的全部内容循环地逐位右移，空出的位用移出位的信号状态填充。输入参数 N 提供的数值表示循环移动的位数，OUT 是循环移位的操作结果。

程序示例

ST:

```
output 2#1010000000000010 := ROR(input1 2#0000000000010101, input2 2#0000000000000011);
```

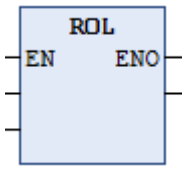
LD:



3.6.5 ROL

对操作数进行循环左移，左边移出的位直接补充到右边最高位。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ROL	循环左移	FC		ROL(IN,N)	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	整型	遵照数据类型	0	数据来源
N	移位位数	整型	遵照数据类型	0	左移位位数

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	偏移后数据	整型	遵照数据类型	0	偏移后的数据

功能说明

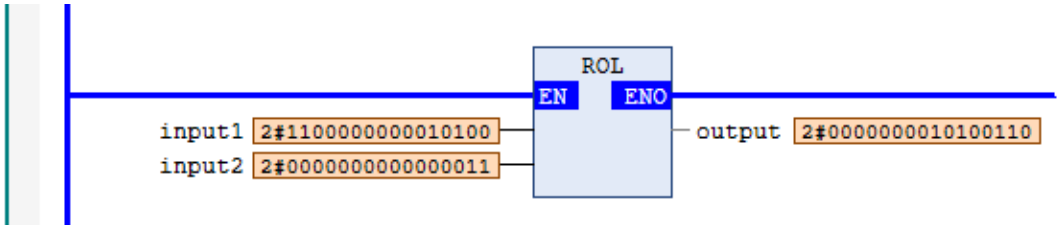
使用该指令可以将输 IN 中的全部内容循环地逐位左移，空出的位用移出位的信号状态填充。输入参数 N 提供的数值表示循环移动的位数，OUT 是循环移位的操作结果。

程序示例

ST:

```
output 2#0000000010100110 := ROL(input1 2#1100000000010100, input2 2#0000000000000011);
```

LD:



3.7 数据类型转换

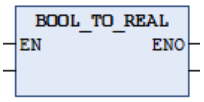
3.7.1 指令列表

指令类别	名称	FB/FC	功能
数据类型转换	BOOL_TO_<TYPE>	FC	BOOL 类型转换
	BYTE_TO_<TYPE>	FC	字节类型转换
	WORD_TO_<TYPE>	FC	字类型转换
	DWORD_TO_<TYPE>	FC	双字类型转换
	INT_TO_<TYPE>	FC	整数型类型转换
	SINT_TO_<TYPE>	FC	短整型转换
	DINT_TO_<TYPE>	FC	长整型转换
	UDINT_TO_<TYPE>	FC	无符号长整型转换
	REAL_TO_<TYPE>	FC	实数类型转换
	STRING_TO_<TYPE>	FC	字符类型转换
	TIME_TO_<TYPE>	FC	时钟类型转换
	TOD_TO_<TYPE>	FC	时间类型转换
	DATE_TO_<TYPE>	FC	日期类型转换
	DT_TO_<TYPE>	FC	日期时间类型转换

3.7.2 BOOL_TO_<TYPE>

把布尔数据类型转换为其他数据类型。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
BOOL_TO_<TYPE>	布尔类型转换指令	FC		:=BOOL_TO_<TYPE>();	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	BOOL	TRUE-FALSE	FALSE	数据来源

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	结果数据	<TYPE>	-	0	转换后的数据

功能说明

1）支持的数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT 和 STRING.

2）在输出为数字类型时，如果输入是 TRUE，则输出为 1；如果输入是 FALSE，则输出为 0。在输出为字符串类型时，如果输入是 TRUE，则输出字符串“TRUE”；如果输入是 FALSE，则输出字符串‘FALSE’。

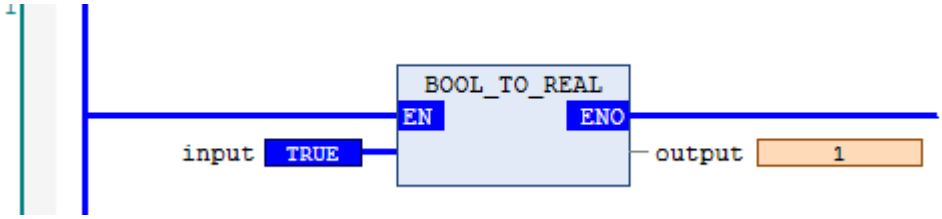
程序示例

以 BOOL_TO_REAL 为例

ST:

```
output 1 := BOOL_TO_REAL(input TRUE);
```

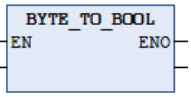
LD:



3.7.3 BYTE_TO_<TYPE>

把字节数据类型转换为其他数据类型。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
BYTE_TO_<TYPE>	字节类型转换指令	FC		:=BYTE_TO_<TYPE>();	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	BYTE	0-255	0	数据来源

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	结果数据	<TYPE>	-	0	转换后的数据

功能说明

1）支持的数据类型：BYTE、WORD、DWORD、SINT、USINT、INT、UINT、DINT、UDINT、REAL、TIME、DATE、TOD、DT 和 STRING.

2）在输出为 BOOL 时：如果输入不等于 0，输出为 TRUE；如果输入等于 0，输出为 FALSE。输出为 TIME 或 TOD 时：输入将以毫秒为单位进行替换。输出为 DATE 或 DT 时：输入将以秒为单位进行转换。

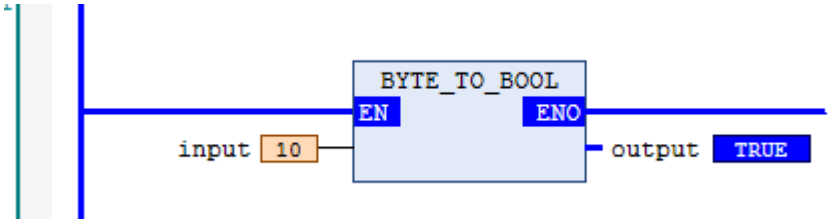
程序示例

以 BYTE_TO_BOOL 为例

ST:

```
output TRUE := BYTE_TO_BOOL(input 10);
```

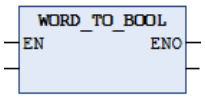
LD:



3.7.4 <整型数据>_TO_<TYPE>

把整型数据类型转换为其他数据类型。由于整型数据类型很多，且转换过程相似，故以 WORD_TO_BOOL 举例。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
WORD_TO_BOOL	字类型转换指令	FC		:=WORD_TO_BOOL();	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	WORD	0-65535	0	数据来源

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	结果数据	BOOL	TRUE-FALSE	FALSE	转换后的数据

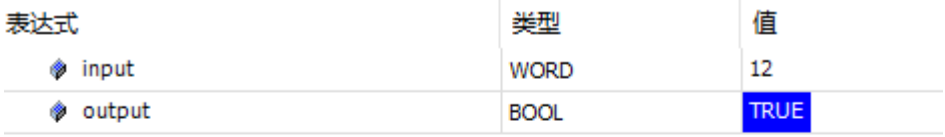
程序示例

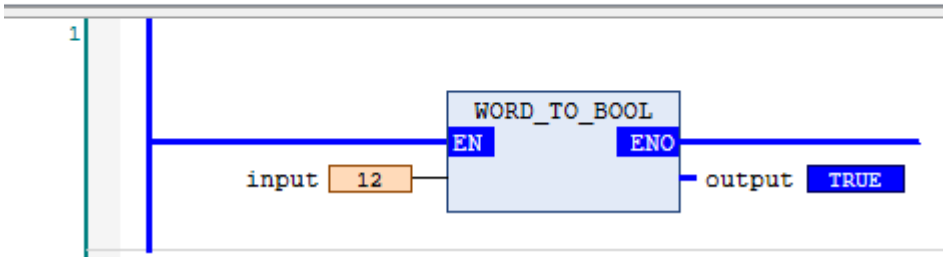
以 WORD_TO_BOOL 为例

ST:


```
1  output TRUE := WORD_TO_BOOL(input 12);
2
3
```

LD:

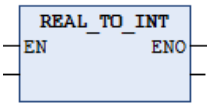




3.7.5 <实数型数据>_TO_<TYPE>

把实数型数据类型转换为其他数据类型。由于可转换的数据类型较多，且转换过程相似，故以 REAL_TO_INT 举例。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
REAL_TO_INT	REAL 类型转换为 INT 型	FC		:=REAL_TO_INT( );	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	数据来源	REAL	遵照数据类型	-	数据来源

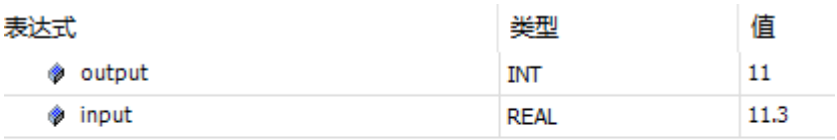
输出变量

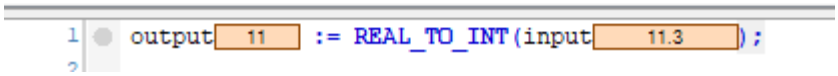
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	结果数据	INT	遵照数据类型	-	转换后的数据

程序示例

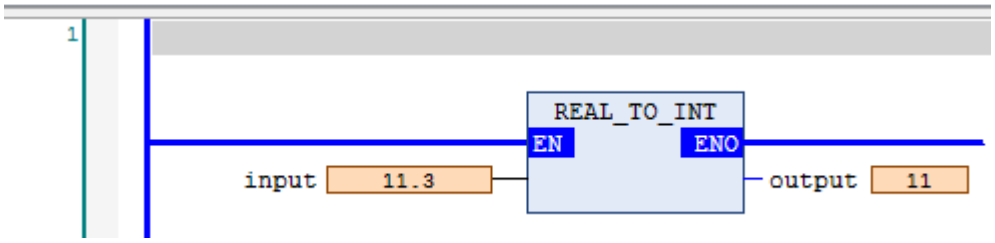
以 REAL_TO_INT 为例

ST:





LD:



3.8 数据处理指令

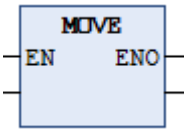
3.8.1 指令列表

指令类别	名称	FB/FC	功能
数据处理指令	MOVE	FC	赋值
	HEXinASCII_TO_BYTE	FC	ASCII 转数字
	BYTE_TO_ HEXinASCII	FC	数字转 ASCII
	WORD_AS_STRING	FC	数字转字符串
	BYTE_TO_HEXSTRING	FC	BYTE 型转 16 进制字符串
	WORD_ TO_HEXSTRING	FC	WORD 型转 16 进制字符串
	DWORD_ TO_HEXSTRING	FC	DWORD 型转 16 进制字符串

3.8.2 MOVE

此运算符用于将一个变量的值分配到另一个相同类型的变量中。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MOVE	赋值	FC		A2:=MOVE(a1);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	传送数据	-	遵照数据类型	-	输入数据

输出变量

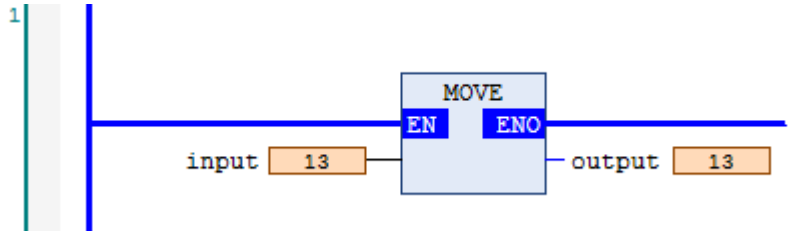
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	接收数据	-	遵照数据类型	-	输出数据

程序示例

ST:

```
output 13 := MOVE(input 13);
```

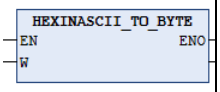
LD:



3.8.3 HEXinASCII_TO_BYTE

此运算符将 16 进制的 ASCII 码转换成 BYTE 类型数据。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
HEXinASCII_TO_BYTE	ASCII 码转 BYTE 字符	FC		output:=HEXinASCII_TO_BYTE(a1);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入数据	WORD	0-FFFF	0	输入数据

输出变量

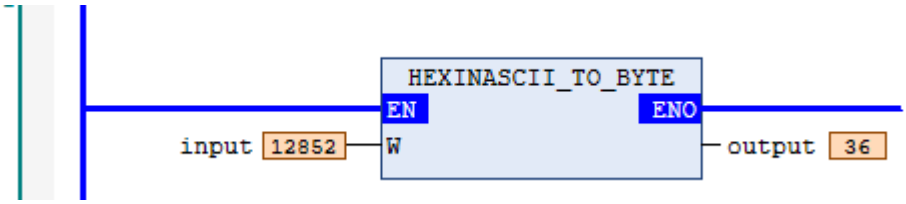
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	BYTE	0-255	0	输出数据

程序示例

ST:

```
output 36 := HEXinASCII_TO_BYTE(input 12852);
```

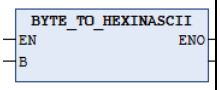
LD:



3.8.4 BYTE_TO_ HEXinASCII

此运算符将 BYTE 类型数据转换成 16 进制的 ASCII 码。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
BYTE_TO_ HEXinASCII	BYTE 字符转 ASCII 码	FC		output:=BYTE _TO_HEXinASCII (a1);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入数据	BYTE	0-255	0	输入数据

输出变量

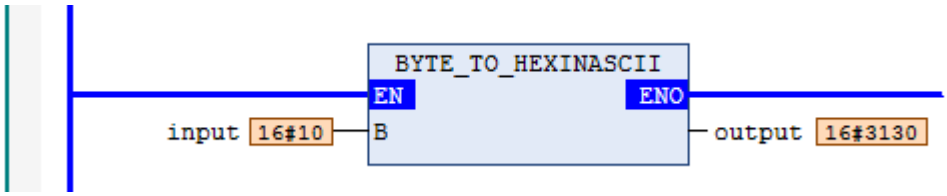
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	WORD	0-FFFF	0	输出数据

程序示例

ST:

```
output 16#3130 := BYTE_TO_HEXinASCII(input 16#10);
```

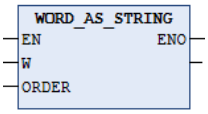
LD:



3.8.5 WORD_AS_STRING

当指令触发时，将源数据的 WORD 类型数据转换成 STRING 类型的数据。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
WORD_AS_STRING	数字转字符串	FC		STR:=WORD_AS_STRING (var1,order);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
W	输入数据	WORD	0-FFFF	0	输入数据
ORDER	高低字节转换	BOOL	FALSE-TRUE	FALSE	用于转换变量的高低字节

输出变量

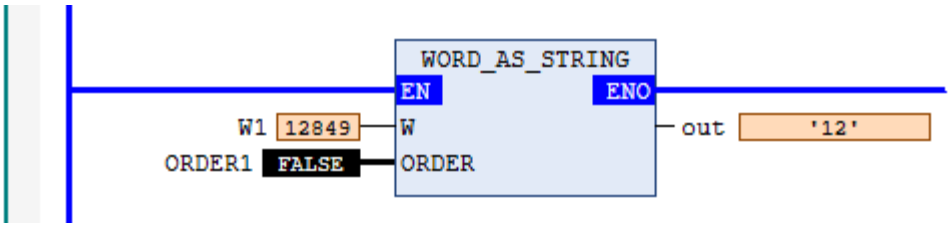
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	STRING	-	''	输出数据

程序示例

ST:

```
str '1' := WORD_AS_STRING(W1 49, order1 FALSE);
```

LD:



注意事项

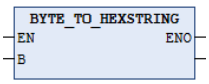
1）该指令在不使用 ORDER 引脚切换高低位的情况下，不修改数值。使用 ORDER 引脚为 True 时，则切换数值的高低位。

2）STRING 默认 80 字符量，可定义变量为 STRING（具体字符量）。

3.8.6 BYTE_TO_HEXSTRING

此运算符将 BYTE 类型数据转换成 16 进制的 ASCII 码。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
BYTE_TO_HEXSTRING	BYTE 类型转换为 16 进制字符串	FC		STR:=BYTE_TO_HEXSTRING (var);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
B	输入数据	BYTE	0-255	0	输入数据

输出变量

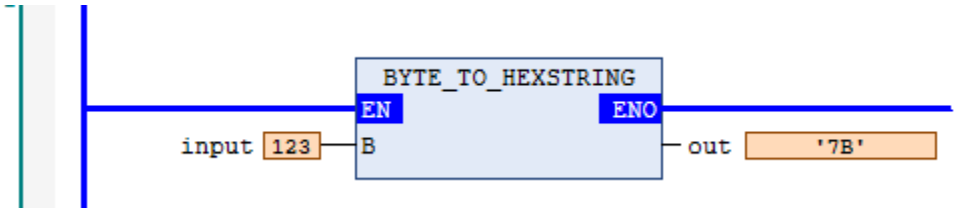
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	STRING	-	''	输出数据

程序示例

ST:

```
str '7B' := BYTE_TO_HEXSTRING(input 123);
```

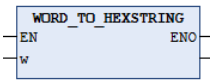
LD:



3.8.7 WORD_TO_HEXSTRING

此运算符将 WORD 类型数据转换成 16 进制的 ASCII 码。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
WORD_TO_HEX STRING	WORD 类型转换为 16 进制字符串	FC		STR:=WORD_TO_HEXSTRING (var);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
W	输入数据	WORD	0-FFFF	0	输入数据

输出变量

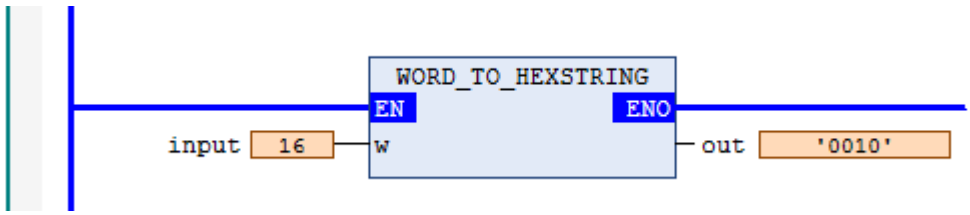
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	STRING	-	''	输出数据

程序示例

ST:

```
str '0010' := WORD_TO_HEXSTRING(input 16 );
```

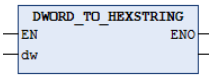
LD:



3.8.8 DWORD_TO_HEXSTRING

此运算符将 DWORD 类型数据转换成 16 进制的 ASCII 码。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
DWORD_TO_HEXSTRING	DWORD 类型转换为 16 进制字符串	FC		STR:=DWORD_TO_HEXSTRING (var);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
DW	输入数据	DWORD	0-FFFF FFFF	0	输入数据

输出变量

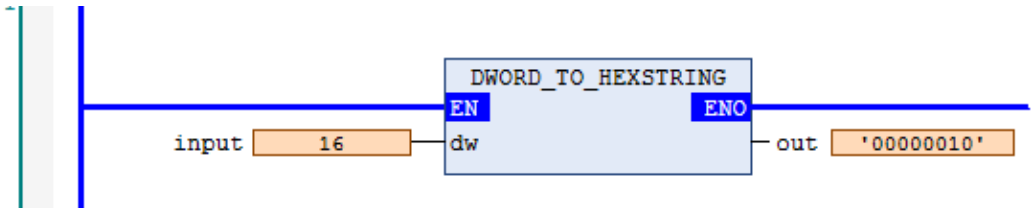
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	STRING	-	''	输出数据

程序示例

ST:

```
str '00003039' := DWORD_TO_HEXSTRING(input 12345 );
```

LD:



3.9 数学运算指令

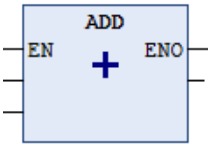
3.9.1 指令列表

指令类别	名称	FB/FC	功能
数学运算指令	ADD	FC	加法
	SUB	FC	减法
	MUL	FC	乘法
	DIV	FC	除法
	MOD	FC	取余
	ABS	FC	绝对值
	SQRT	FC	平方根
	LN	FC	自然对数
	LOG	FC	常用对数
	EXP	FC	指数
	EXPT	FC	幂指数
	SIN	FC	正弦
	COS	FC	余弦
	TAN	FC	正切
	ASIN	FC	反正弦
	ACOS	FC	反余弦
	ATAN	FC	反正切
	SIZEOF	FC	数据类型大小

3.9.2 ADD

将输入的变量相加，输出相加的结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ADD	加法指令	FC		OUT:=IN1 + IN2;	-

相关变量

输入变量

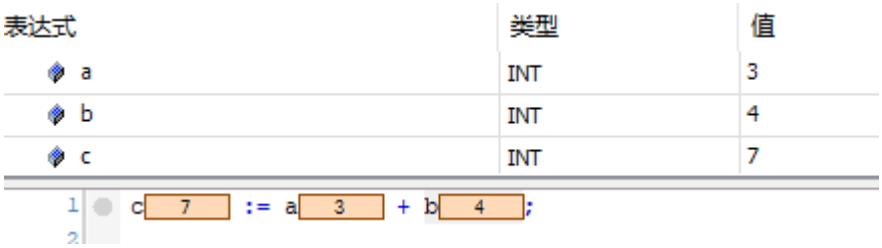
输入变量	名称	数据类型	有效范围	初始值	描述
IN1	相加值	-	遵照数据类型	-	输入
IN2	相加值	-	遵照数据类型	-	输入

输出变量

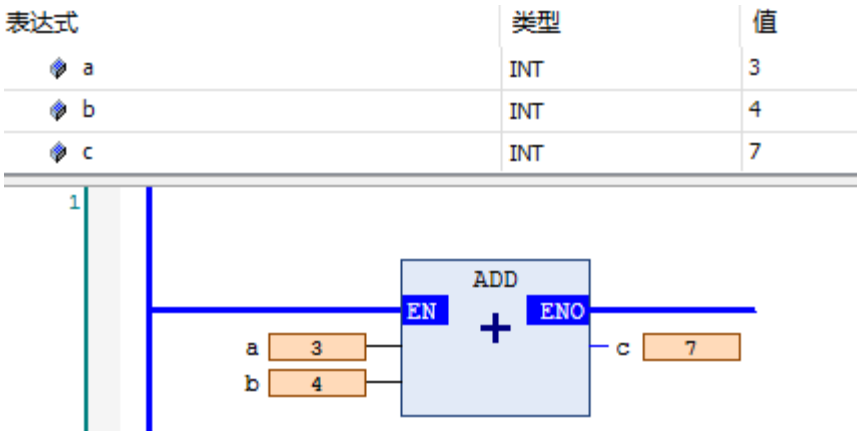
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出值	-	遵照数据类型	-	相加后的结果

程序示例

ST:



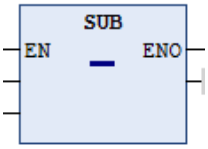
LD:



3.9.3 SUB

将左侧两个输入相减，从右侧输出减法运算结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SUB	减法指令	FC		OUT:=IN1 - IN2;	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	相减值	-	遵照数据类型	-	输入
IN2	相减值	-	遵照数据类型	-	输入

输出变量

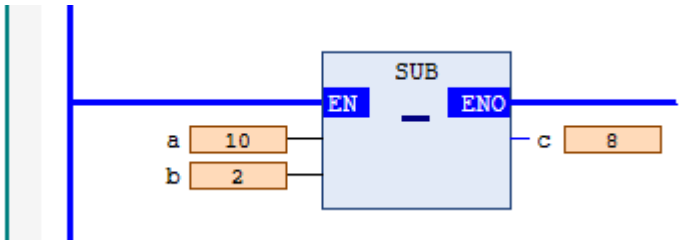
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出值	-	遵照数据类型	-	相减后的结果

程序示例

ST:

```
c 8 := a 10 - b 2;
```

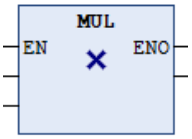
LD:



3.9.4 MUL

将左侧两个输入相乘，从右侧输出乘法运算结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MUL	乘法指令	FC		OUT:=IN1 * IN2;	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	相乘值	-	遵照数据类型	-	输入
IN2	相乘值	-	遵照数据类型	-	输入

输出变量

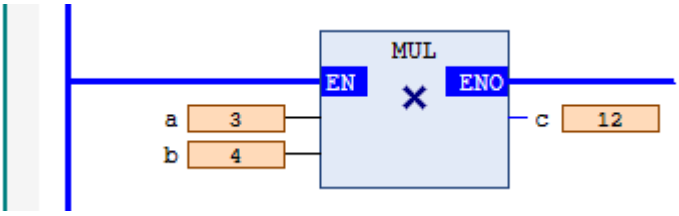
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出值	-	遵照数据类型	-	相乘后的结果

程序示例

ST:

```
c 12 := a 3 * b 4;
```

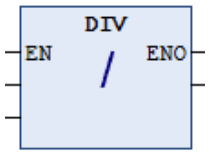
LD:



3.9.5 DIV

将左侧两个输入相除，从右侧输出除法运算后的商值。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
DIV	除法指令	FC		OUT:=IN1 / IN2;	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	被除值	-	遵照数据类型	-	输入
IN2	除值	-	遵照数据类型	-	输入

输出变量

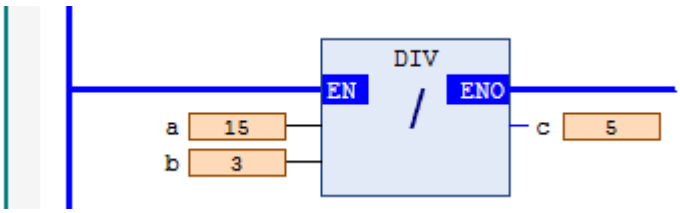
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出值	-	遵照数据类型	-	相除后的结果

程序示例

ST:

```
c 5 := a 15 / b 3;
```

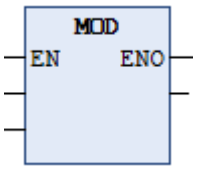
LD:



3.9.6 MOD

将左侧两个输入相除，从右侧输出相除后非负的整数余数。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MOD	取余指令	FC		OUT:=IN1 MOD IN2;	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	被除值	-	遵照数据类型	-	输入
IN2	除值	-	遵照数据类型	-	输入

输出变量

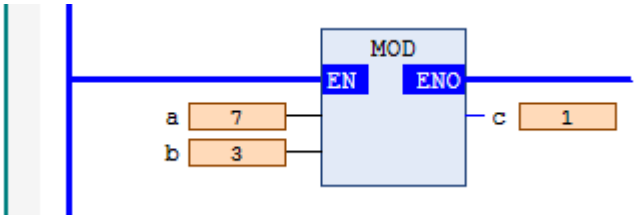
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	余数	-	遵照数据类型	-	取余后的结果

程序示例

ST:

```
c 1 := a 7 MOD b 3;
```

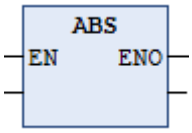
LD:



3.9.7 ABS

将输入数据取绝对值后赋值给输出变量。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ABS	绝对值指令	FC		OUT:=ABS(IN);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入	-	遵照数据类型	-	输入

输出变量

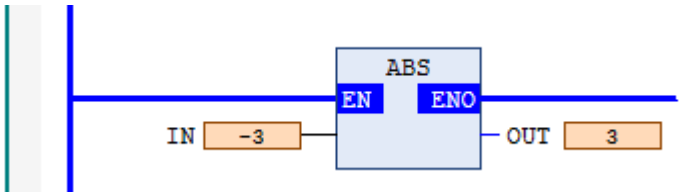
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	绝对值

程序示例

ST:

```
OUT 3 := ABS(IN -3);
```

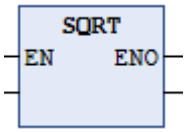
LD:



3.9.8 SQRT

将输入数据进行平方根运算，输出平方根运算结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SQRT	平方根指令	FC		OUT:=SQRT(IN);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入	-	遵照数据类型	-	输入

输出变量

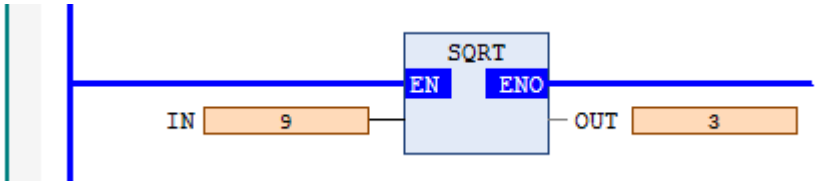
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	平方根

程序示例

ST:

```
OUT 3 := SQRT(IN 9);
```

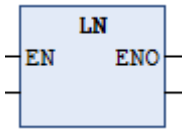
LD:



3.9.9 LN

将输入数据进行对数运算，输出对数运算结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LN	自然对数指令	FC		OUT:=LN(IN);	-

相关变量

输入变量

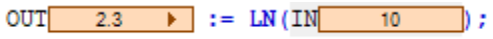
输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入	-	遵照数据类型	-	输入

输出变量

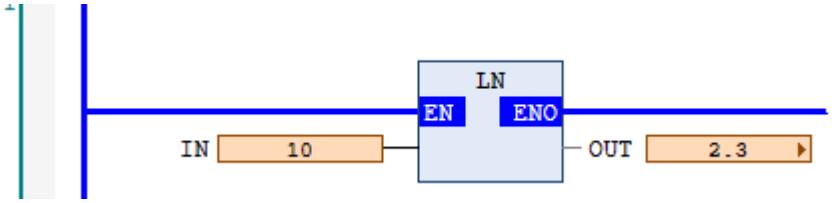
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	自然对数值

程序示例

ST:



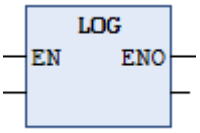
LD:



3.9.10 LOG

将输入数据进行以 10 为底的对数运算，输出对数运算结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LOG	计算 10 的对数	FC		OUT:=LN(IN);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入	-	遵照数据类型	-	输入

输出变量

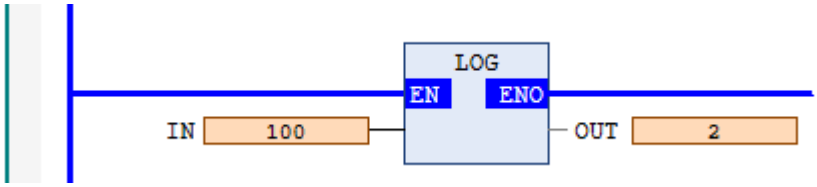
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	以 10 为底的对数值

程序示例

ST:

```
OUT 2 := LOG(IN 100);
```

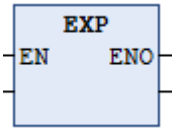
LD:



3.9.11 EXP

将输入数据进行指数运算，输出运算结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
EXP	计算指数值	FC		OUT:=EXP(IN);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入	-	遵照数据类型	-	输入

输出变量

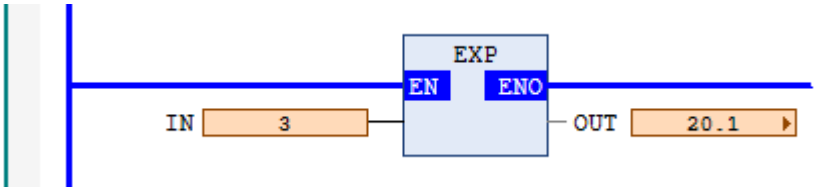
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	指数值

程序示例

ST:

```
OUT 20.1 := EXP(IN 3);
```

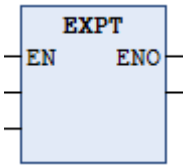
LD:



3.9.12 EXPT

将输入数据 1 作为底数，输入数据 2 作为次方数，输出幂运算的结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
EXPT	计算次方值	FC		OUT:=EXPT(IN1, IN2);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入底数	-	遵照数据类型	-	输入
IN2	输入次方	-	遵照数据类型	-	输入

输出变量

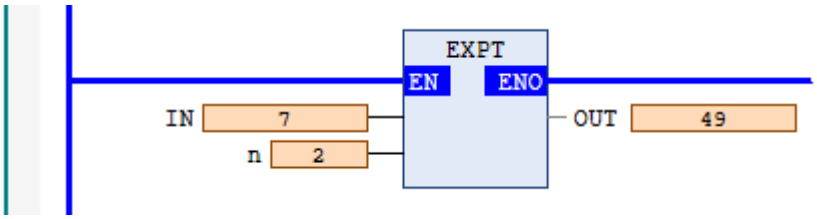
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	次方值

程序示例

ST:

```
OUT 49 := EXPT(IN 7, n 2);
```

LD:



注意事项

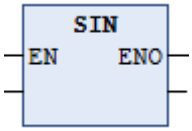
如果出现以下情况，则函数的结果是未定义的:

1. 基值为负。
2. 基数为 0 并且指数为<=0。

3.9.13 SIN

将输入值进行正弦运算，输出正弦运算结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SIN	正弦	FC		OUT:=SIN(IN);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入	-	遵照数据类型	-	输入

输出变量

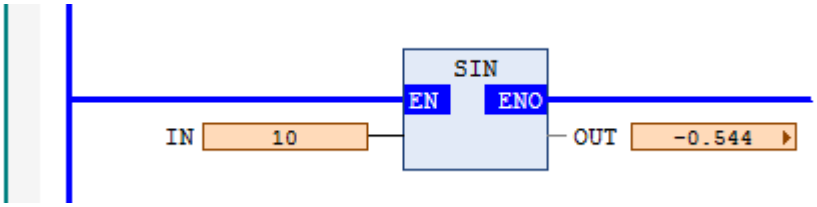
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	正弦值

程序示例

ST:

```
OUT -0.544 := SIN(IN 10);
```

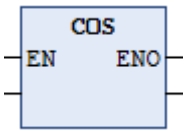
LD:



3.9.14 COS

将输入值进行余弦运算，输出余弦运算结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
COS	余弦	FC		OUT:=COS(IN);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入	-	遵照数据类型	-	输入

输出变量

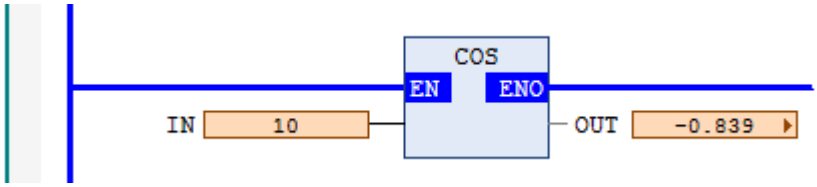
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	余弦值

程序示例

ST:

```
OUT -0.839 := COS(IN 10);
```

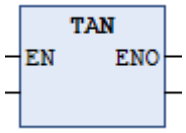
LD:



3.9.15 TAN

将输入数据作为正切运算，输出正切运算的结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
TAN	正切	FC		OUT:=TAN(IN);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入	-	遵照数据类型	-	输入

输出变量

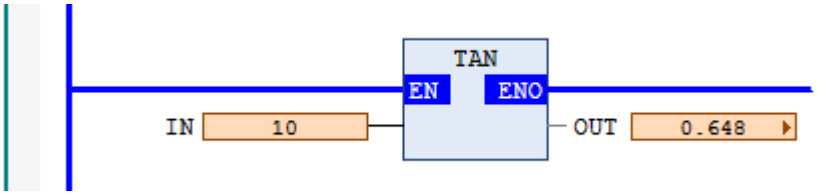
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	正切值

程序示例

ST:

```
OUT 0.648 := TAN(IN 10);
```

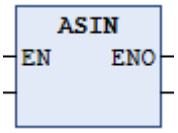
LD:



3.9.16 ASIN

将输入数据作为反正弦运算，输出反正弦运算的结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ASIN	反正弦	FC		OUT:=ASIN(IN);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入	-	遵照数据类型	-	输入

输出变量

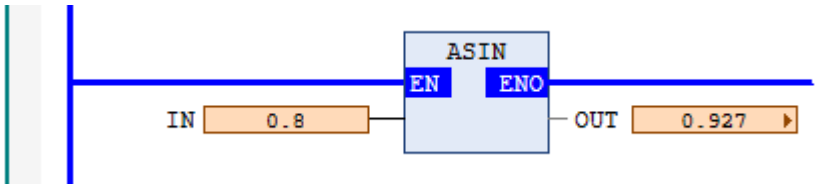
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	反正弦值

程序示例

ST:

```
OUT 0.927 := ASIN(IN 0.8 );
```

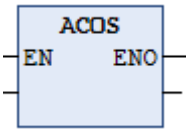
LD:



3.9.17 ACOS

将输入数据作为反余弦运算，输出反余弦运算的结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ACOS	反余弦	FC		OUT:=ACOS(IN) ;	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入	-	遵照数据类型	-	输入

输出变量

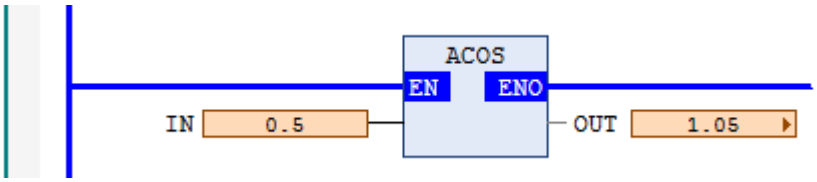
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	反余弦值

程序示例

ST:

```
OUT 1.05 := ACOS(IN 0.5);
```

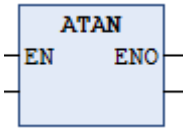
LD:



3.9.18 ATAN

将输入数据作为反正切运算，输出反正切运算的结果。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ATAN	反正切	FC		OUT:=ATAN(IN) ;	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入	-	遵照数据类型	-	输入

输出变量

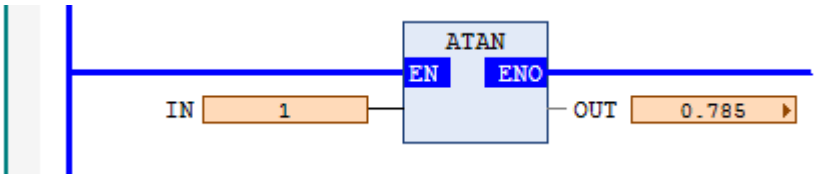
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	反正切值

程序示例

ST:

```
OUT 0.785 := ATAN(IN 1);
```

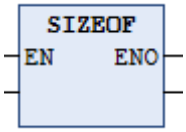
LD:



3.9.19 SIZEOF

用来确定输入变量所占用的字节数。SIZEOF 操作符通常返回一个无符号数。返回值的大小为输入变量的类型所占的字节数。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SIZEOF	数据类型大小	FC		OUT:=SIZEOF(IN);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN1	输入	-	遵照数据类型	-	输入

输出变量

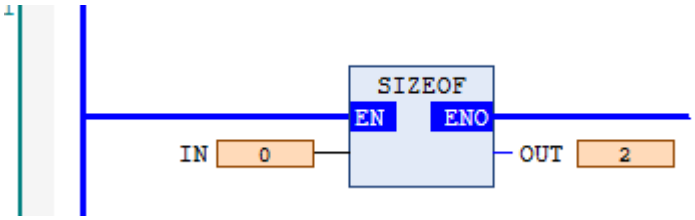
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出	-	遵照数据类型	-	数据类型字节数

程序示例

ST:

```
OUT 2 := SIZEOF(IN 0);
```

LD:



3.10 字符串

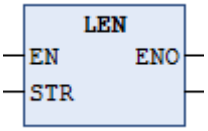
3.10.1 指令列表

指令类别	名称	FB/FC	功能
字符串指令	LEN	FC	获取字符串长度
	LEFT	FC	从左边取字符串
	RIGHT	FC	从右边取字符串
	MID	FC	从中间取字符串
	CONCAT	FC	字符串连接
	INSERT	FC	字符串插入
	DELETE	FC	字符串删除
	FIND	FC	字符串查找
	REPLACE	FC	字符串替换

3.10.2 LEN

获取字符串的长度指令。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LEN	获取字符串长度	FC		OUT:=LEN(STR);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	输入	STRING	-	''	输入

输出变量

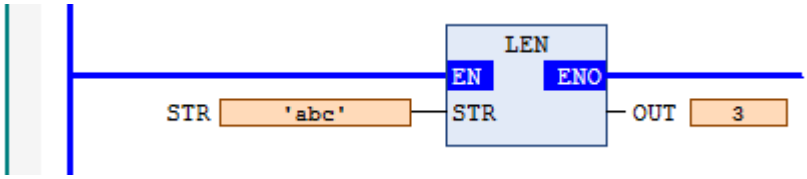
输出变量	名称	数据类型	有效范围	初始值	描述
LEN	返回值	INT	0-65535	0	目标字符串个数

程序示例

ST:

```
OUT 3 := LEN(STR 'abc');
```

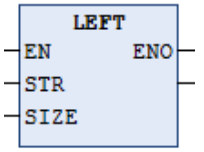
LD:



3.10.3 LEFT

从字符串左侧开始向右提取 SIZE 个字符，返回所取长度的字符串。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
LEFT	取左边字符串	FC		STR1:=LEFT(STR ,SIZE);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	源字符串	STRING	-	''	输入
SIZE	取字符的个数	INT	0-65535	0	输入

输出变量

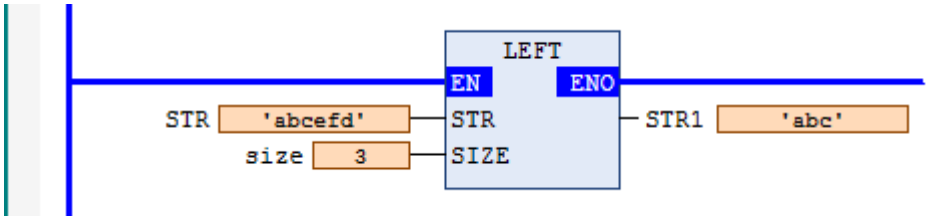
输出变量	名称	数据类型	有效范围	初始值	描述
STR1	返回值	STRING	-	''	目标字符串

程序示例

ST:

```
STR1 'abc' := LEFT(STR 'abcefd', size 3);
```

LD:



3.10.4 RIGHT

从字符串右侧开始向左提取 SIZE 个字符，返回所取长度的字符串。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
RIGHT	取右边字符串	FC	RIGHT EN ENO STR SIZE	STR1:=RIGHT(STR,SIZE);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	源字符串	STRING	-	''	输入
SIZE	取字符的个数	INT	0-65535	0	输入

输出变量

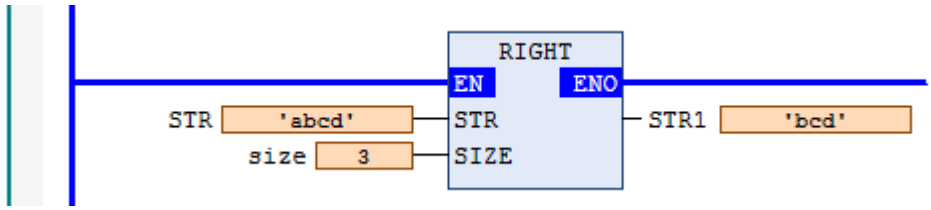
输出变量	名称	数据类型	有效范围	初始值	描述
STR1	返回值	STRING	-	''	目标字符串

程序示例

ST:

```
STR1 'bcd' := RIGHT(STR 'abcd', size 3);
```

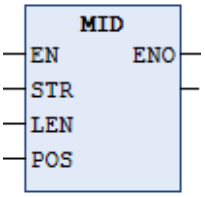
LD:



3.10.5 MID

从字符串指定位置取指定长度的字符串，返回所取长度的字符串。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MID	取中间字符串	FC		STR1:= MID(STR,LE N,POS);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	源字符串	STRING	-	''	输入字符串
LEN	取字符的个数	INT	0-65535	0	数据长度
POS	取字符串位置	INT	0-65535	0	取字符串位置

输出变量

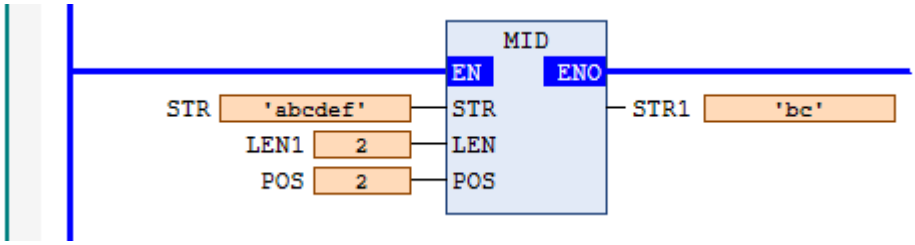
输出变量	名称	数据类型	有效范围	初始值	描述
STR1	返回值	STRING	-	''	目标字符串

程序示例

ST:

```
STR1 'bc' := MID(STR 'abcde', LEN1 2, POS1 2);
```

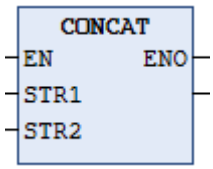
LD:



3.10.6 CONCAT

将输入的两字符串连接并从右侧输出。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
CONCAT	字符串连接	FC		STR:= CONCAT(STR1,STR2);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	源字符串 1	STRING	-	''	输入字符串 1
STR2	源字符串 2	STRING	-	''	输入字符串 2

输出变量

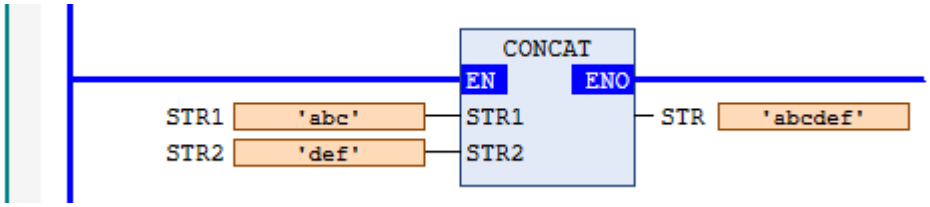
输出变量	名称	数据类型	有效范围	初始值	描述
STR	返回值	STRING	-	''	输出合成的字符串

程序示例

ST:

```
STR 'abcdef' := CONCAT(STR1 'abc', STR2 'def');
```

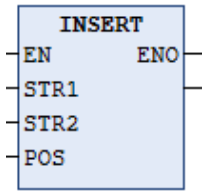
LD:



3.10.7 INSERT

在输入源字符串中插入新的字符串，合并成新的字符串输出。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
INSERT	插入字符串	FC		STR:= INSERT(STR1,STR2,POS);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	源字符串 1	STRING	-	''	输入字符串 1
STR2	源字符串 2	STRING	-	''	输入字符串 2
POS	插入位置	INT	0-65535	0	字符串插入位置

输出变量

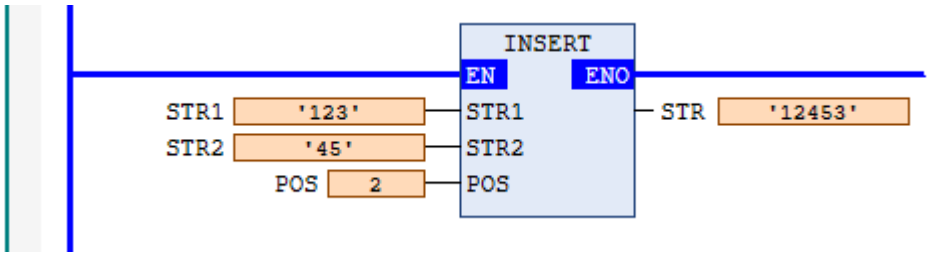
输出变量	名称	数据类型	有效范围	初始值	描述
STR	返回值	STRING	-	''	输出新的字符串

程序示例

ST:

```
STR '12453' := INSERT(STR1 '123', STR2 '45', POS 2);
```

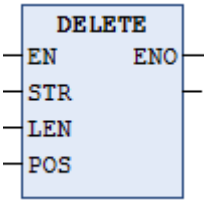
LD:



3.10.8 DELETE

在输入源字符串中删除指定位置指定长度的字符串，合并成新的字符串输出。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
DELETE	删除字符串	FC		STR:= DELETE(STR,LEN ,POS);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR	源字符串	STRING	-	''	输入字符串
LEN	字符串长度	INT	0-65535	0	字符串长度
POS	字符串位置	INT	0-65535	0	删除的字符位置

输出变量

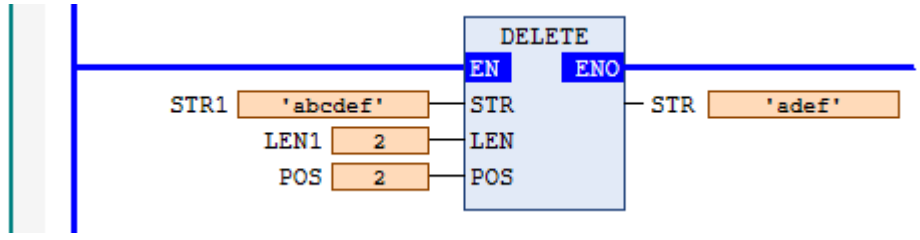
输出变量	名称	数据类型	有效范围	初始值	描述
STR	返回值	STRING	-	''	输出新的字符串

程序示例

ST:

```
STR 'adef' := DELETE(STR1 'abcdef', LEN1 2, POS 2);
```

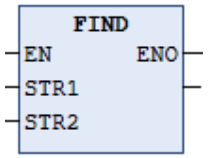
LD:



3.10.9 FIND

检测目标字符串在源字符串中的位置。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
FIND	查找字符串	FC		STR:=FIND(STR1,STR2);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	源字符串	STRING	-	''	输入字符串
STR2	源字符串	STRING	-	''	输入字符串

输出变量

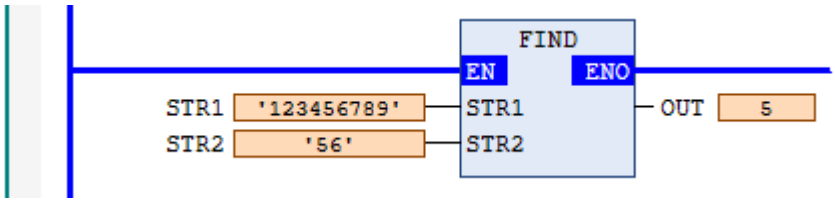
输出变量	名称	数据类型	有效范围	初始值	描述
OUTPUT	返回值	INT	0-65535	0	目标字符串在源字符串中的位置

程序示例

ST:

```
OUT 5 := FIND(STR1 '123456789', STR2 '56');
```

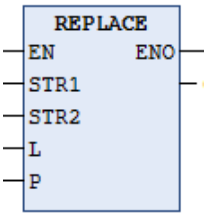
LD:



3.10.10 REPLACE

用目标字符串替换掉原字符串中的一部分，将替换后生成的新字符串作为返回值。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
REPLACE	替换字符串	FC		STR:= REPLACE(STR1,STR2,L,P);	Standard

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
STR1	输入字符串	STRING	-	''	输入字符串
STR2	输入字符串	STRING	-	''	输入字符串
L	替换长度	INT	0-65535	0	替换长度
P	替换位置	INT	0-65535	0	替换位置

输出变量

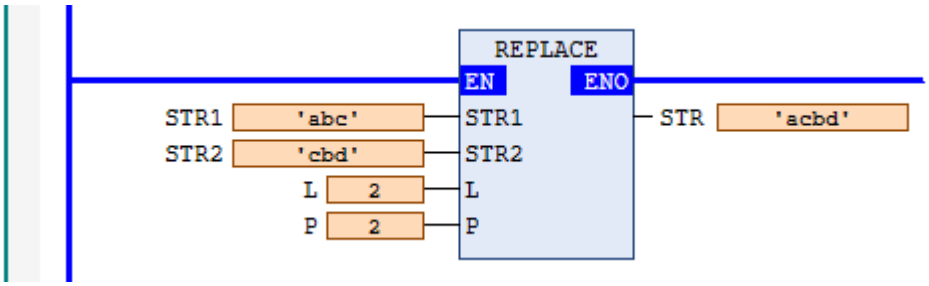
输出变量	名称	数据类型	有效范围	初始值	描述
OUTPUT	返回值	STRING	-	''	目标字符串

程序示例

ST:

```
STR 'acbd' := REPLACE(STR1 'abc', STR2 'cbd', L 2, P 2);
```

LD:



3.11 地址操作

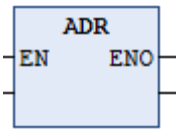
3.11.1 指令列表

指令类别	名称	FB/FC	功能
地址操作	ADR	FC	取地址指令
	^	FC	取地址内容指令
	BITADR	FC	位地址

3.11.2 ADR

取得输入变量的内存地址，结果赋值给输出变量。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ADR	取地址	FC		OUT:=ADR(IN);	-

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	目标数据	任何类型	遵照数据类型	-	目标数据

输出变量

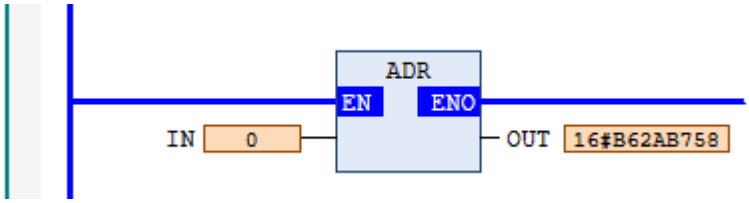
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	目标数据地址	POINTER_TO_<TYPE>	-	-	目标数据地址

程序示例

ST:

```
OUT 16#B62AB770 := ADR(IN 0);
```

LD:



3.11.3 ^

取地址内容指令。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
^	取地址内容	FC	^	^	-

相关变量

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
OUT	指向地址的内容	-	-	-	指向地址内容

程序示例

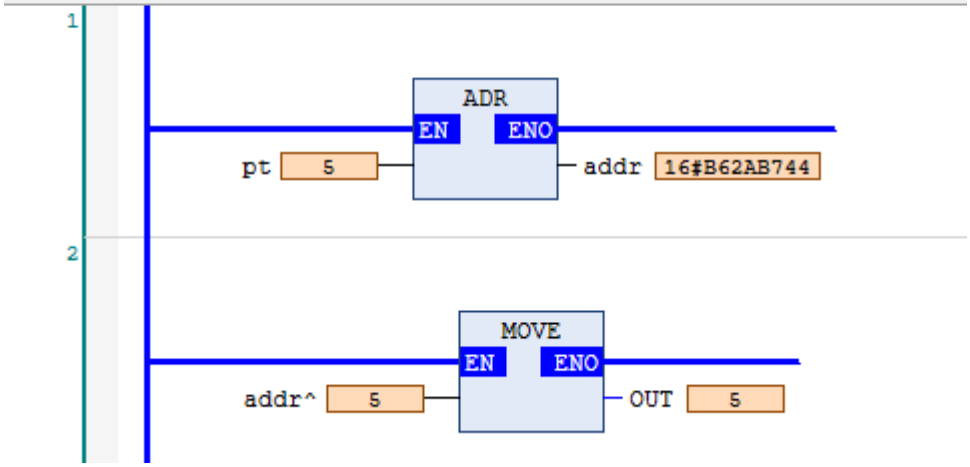
ST:

表达式	类型	值
pt	INT	5
addr	POINTER TO INT	16#B62AB746
out	INT	5

```
1  addr 16#B62AB746 := ADR(pt 5);
2  out 5 := addr^ 5;
```

LD:

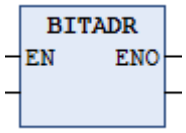
表达式	类型	值
pt	INT	5
addr	POINTER TO INT	16#B62AB744
out	INT	5



3.11.4 BITADR

返回分配变量的位地址信息偏移量。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
BITADR	位地址指令	FC		OUT:=BITADR(IN);	-

相关变量

输入变量

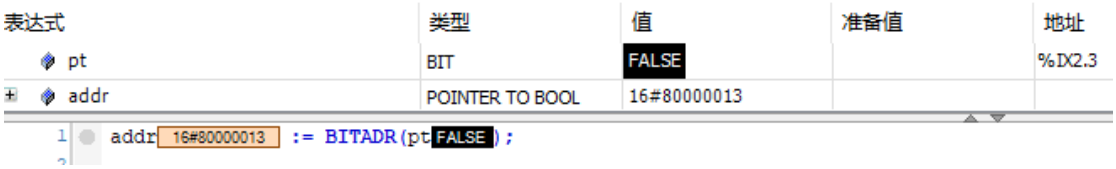
输入变量	名称	数据类型	有效范围	初始值	描述
IN	目标数据	BOOL	FALSE-TRUE	-	目标数据

输出变量

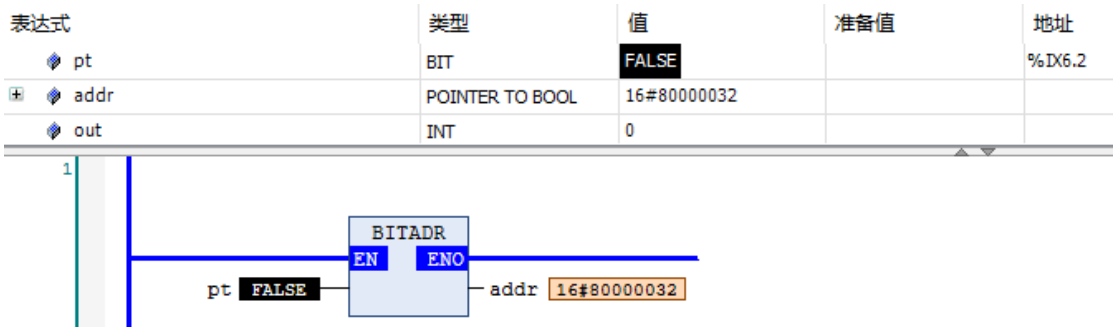
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	指向目标地址	DWORD	遵照数据类型	-	指向目标地址

程序示例

ST:



LD:



3.12 模拟量计算

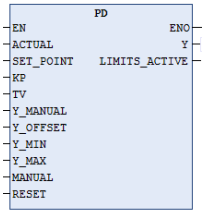
3.12.1 指令列表

指令类别	名称	FB/FC	功能
模拟量计算	PD	FB	比例微分控制
	PID	FB	比例积分微分控制
	PID_FIXCYCLE	FB	可手动设置循环周期的比例积分微分控制

3.12.2 PD

比例微分控制器。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
PD	比例微分控制	FB		PD_0(ACTUAL:= , SET_POINT:= , KP:= , TV:= , Y_MANUAL:= , Y_OFFSET:= , Y_MIN:= , Y_MAX:= , MANUAL:= , RESET:= , Y=> , LIMITS_ACTIVE=>);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
ACTUAL	控制变量当前值	REAL	-	0	控制变量当前值
SET_POINT	设置值	REAL	-	0	设置值
KP	比例系数	REAL	-	0	比例系数
TV	微分时间	REAL	-	0	微分时间
Y_MANUAL	手动输出值	REAL	-	0	手动输出值
Y_OFFSET	输出 Y 的偏移量	REAL	-	0	输出 Y 的偏移量
Y_MIN	输出 Y 的最小值	REAL	-	0	输出 Y 的最小值
Y_MAX	输出 Y 的最大值	REAL	-	0	输出 Y 的最大值
MANUAL	手动输出	BOOL	FALSE-TRUE	FALSE	手动输出
RESET	复位	BOOL	FALSE-TRUE	FALSE	复位

输出变量

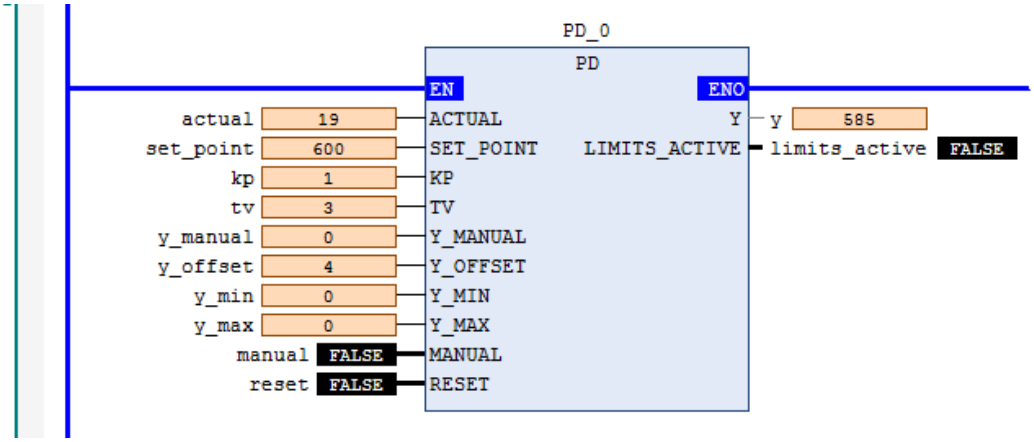
输出变量	名称	数据类型	有效范围	初始值	描述
Y	输出值	REAL	-	0	输出值
LIMITS_ACTIVE	到达给定限制值	BOOL	FALSE-TRUE	FALSE	到达给定限制值

程序示例

ST:

```
PD_0(
    ACTUAL 19 := actual 19,
    SET_POINT 600 := set_point 600,
    KP 1 := kp 1,
    TV 3 := tv 3,
    Y_MANUAL 0 := y_manual 0,
    Y_OFFSET 4 := y_offset 4,
    Y_MIN 0 := y_min 0,
    Y_MAX 0 := y_max 0,
    MANUAL FALSE := manual FALSE,
    RESET FALSE := reset FALSE,
    Y 585 => y 585,
    LIMITS_ACTIVE FALSE => limits_active FALSE);
```

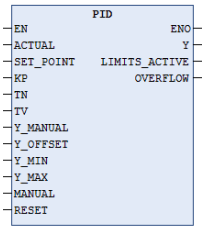
LD:



3.12.3 PID

比例积分微分控制器。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
PID	比例积分微分控制	FB		PID_0(ACTUAL:= , SET_POINT:= , KP:= , TN:= , TV:= , Y_MANUAL:= , Y_OFFSET:= , Y_MIN:= , Y_MAX:= , MANUAL:= , RESET:= , Y=> , LIMITS_ACTIVE=> , OVERFLOW=>);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
ACTUAL	控制变量当前值	REAL	-	0	当前值，过程变量
SET_POINT	设置值	REAL	-	0	所需值，设定值
KP	比例系数	REAL	-	0	比例系数
TN	积分时间	REAL	-	0	重置时间，I 形部件的相互单位增益
TV	微分时间	REAL	-	0	微分作用时间，D 部分的单位增益
Y_MANUAL	手动输出值	REAL	-	0	定义手动
Y_OFFSET	输出 Y 的偏移量	REAL	-	0	操纵变量 Y 的偏移量
Y_MIN	输出 Y 的最小值	REAL	-	0	操纵变量 Y 的最小值
Y_MAX	输出 Y 的最大值	REAL	-	0	操纵变量 Y 的最大值
MANUAL	手动输出	BOOL	FALSE-TRUE	FALSE	激活手动操作
RESET	复位	BOOL	FALSE-TRUE	FALSE	重置控制器

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Y	输出值	REAL	-	0	输出值
LIMITS_ACTIVE	到达给定限制值	BOOL	FALSE-TRUE	FALSE	到达给定限制值

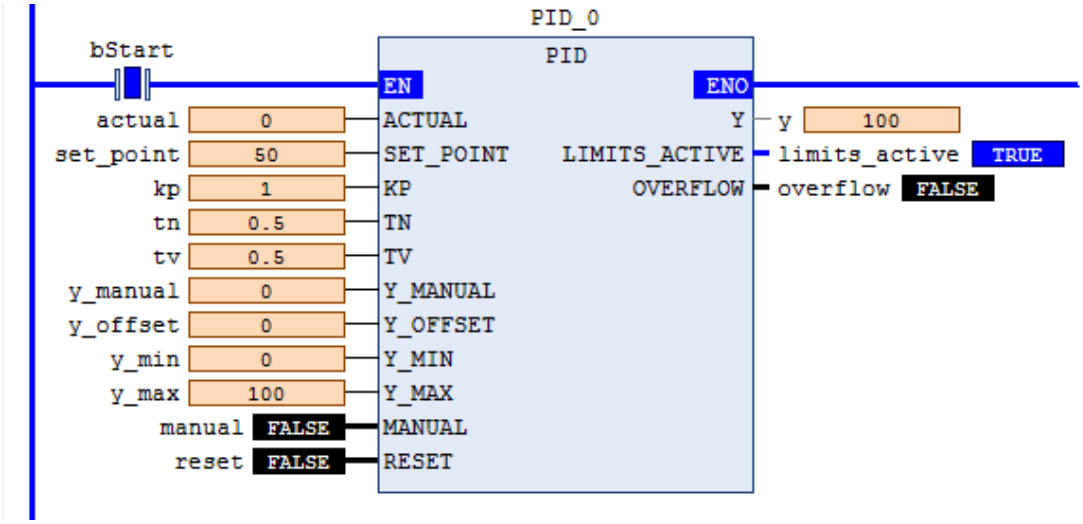
OVERFLOW	溢出标志	BOOL	FALSE-TRUE	FALSE	溢出标志

程序示例

ST:

```
PID_0(
    ACTUAL 0 := actual 0,
    SET_POINT 50 := set_point 50,
    KP 1 := kp 1,
    TN 0.5 := tn 0.5,
    TV 0.5 := tv 0.5,
    Y_MANUAL 0 := y_manual 0,
    Y_OFFSET 0 := y_offset 0,
    Y_MIN 0 := y_min 0,
    Y_MAX 100 := y_max 100,
    MANUAL FALSE := manual FALSE,
    RESET FALSE := reset FALSE,
    Y 100 => y 100,
    LIMITS_ACTIVE TRUE => limits_active TRUE,
    OVERFLOW FALSE => overflow FALSE);
```

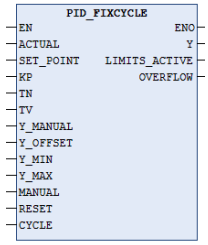
LD:



3.12.4 PID_FIXCYCLE

可手动设置循环周期的比例积分微分控制器。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
PID_FIXCYCLE	可手动设置循环周期的比例积分微分控制	FB		PID_FIXCYCLE_0(ACTUAL:= , SET_POINT:= , KP:= , TN:= , TV:= , Y_MANUAL:= , Y_OFFSET:= , Y_MIN:= , Y_MAX:= , MANUAL:= , RESET:= , CYCLE:= , Y=> , LIMITS_ACTIVE=> , OVERFLOW=>);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
ACTUAL	控制变量当前值	REAL	-	0	当前值，过程变量
SET_POINT	设置值	REAL	-	0	所需值，设定值
KP	比例系数	REAL	-	0	比例系数
TN	积分时间	REAL	-	0	重置时间，I 形部件的相互单位增益
TV	微分时间	REAL	-	0	微分作用时间，D 部分的单位增益
Y_MANUAL	手动输出值	REAL	-	0	定义手动
Y_OFFSET	输出 Y 的偏移量	REAL	-	0	操纵变量 Y 的偏移量
Y_MIN	输出 Y 的最小值	REAL	-	0	操纵变量 Y 的最小值
Y_MAX	输出 Y 的最大值	REAL	-	0	操纵变量 Y 的最大值
MANUAL	手动输出	BOOL	FALSE-TRUE	FALSE	激活手动操作
RESET	复位	BOOL	FALSE-TRUE	FALSE	重置控制器
CYCLE	采样周期时间	REAL	-	0	采样周期时间

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述

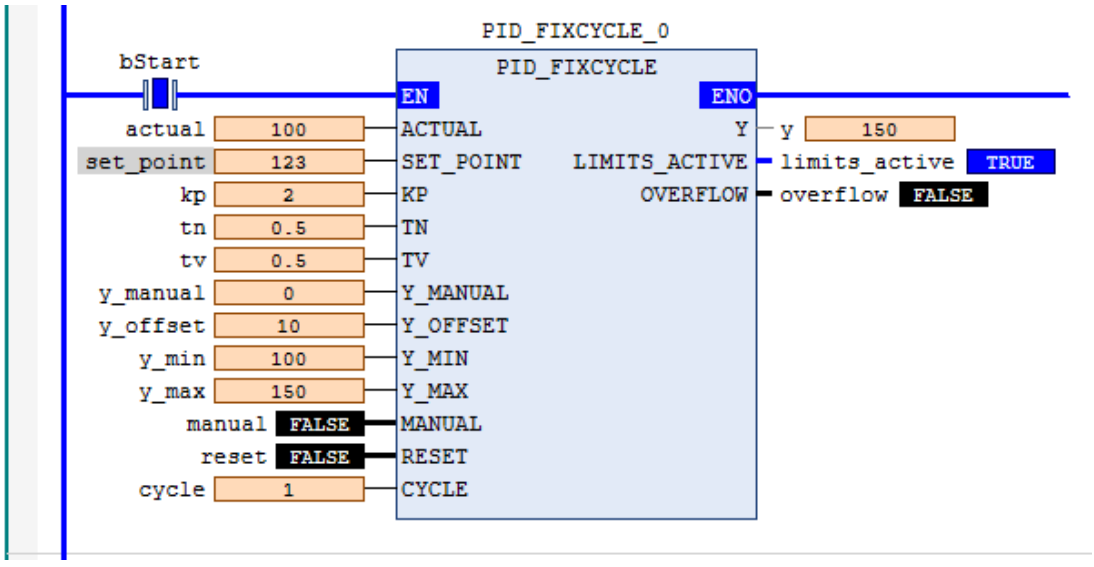
Y	输出值	REAL	-	0	输出值
LIMITS_ACTIVE	到达给定限制值	BOOL	FALSE-TRUE	FALSE	到达给定限制值
OVERFLOW	溢出标志	BOOL	FALSE-TRUE	FALSE	溢出标志

程序示例

ST:

```
PID_FIXCYCLE_0(
    ACTUAL 19 := actual 19,
    SET_POINT 21 := set_point 21,
    KP 1 := kp 1,
    TN 10 := tn 10,
    TV 0 := tv 0,
    Y_MANUAL 0 := y_manual 0,
    Y_OFFSET 0 := y_offset 0,
    Y_MIN 0 := y_min 0,
    Y_MAX 0 := y_max 0,
    MANUAL FALSE := manual FALSE,
    RESET FALSE := reset FALSE,
    CYCLE 0.05 := cycle 0.05,
    Y 59 => y 59,
    LIMITS_ACTIVE FALSE => limits_active FALSE,
    OVERFLOW FALSE => overflow FALSE);
```

LD:



3.13 BCD 码转换

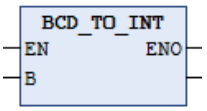
3.13.1 指令列表

指令类别	名称	FB/FC	功能
BCD 码转换指令	BCD_TO_INT	FC	BCD 码转整型
	INT_TO_BCD	FC	整型转 BCD 码
	BCD_TO_BYTE	FC	BCD 码转字节
	BYTE_TO_BCD	FC	字节转 BCD 码
	BCD_TO_WORD	FC	BCD 码转字
	WORD_TO_BCD	FC	字转 BCD 码
	BCD_TO_DWORD	FC	BCD 码转双字
	DWORD_TO_BCD	FC	双字转 BCD 码

3.13.2 BCD_TO_INT

将源数据的 BCD 码转换成 INT 类型的数据。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
BCD_TO_INT	BCD 码转整型	FC		Output:=BCD_TO_INT(B);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
B	输入数据	BYTE	0-255	0	输入数据

输出变量

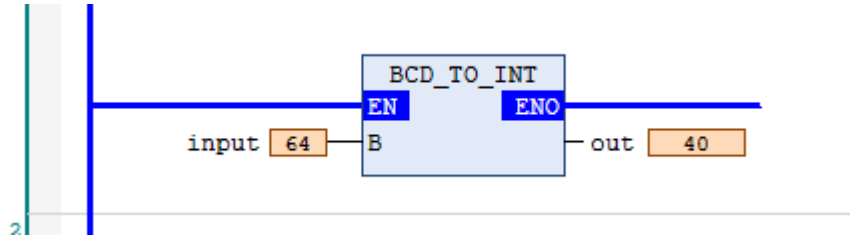
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	INT	-32768-32768	0	输出数据

程序示例

ST:

```
out 40 := BCD_TO_INT(input 64);
```

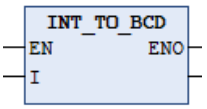
LD:



3.13.3 INT_TO_BCD

将源数据的 INT 型数据转换成 BCD 码。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
INT_TO_BCD	整型转 BCD 码	FC		Output:= INT_TO_ BCD (I);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
I	输入数据	INT	0-99	0	输入数据

输出变量

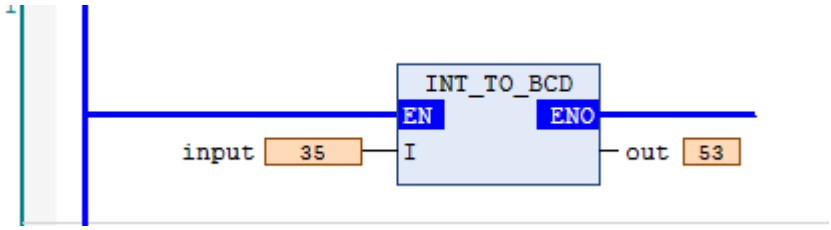
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	BYTE	0-255	0	输出数据

程序示例

ST:

```
out 53 := INT_TO_BCD(input 35 );
```

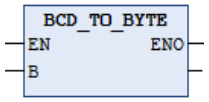
LD:



3.13.4 BCD_TO_BYTE

将源数据的 BCD 码转换成 BYTE 类型的数据。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
BCD_TO_BYTE	BCD 码转 BYTE 型	FC		Output:= BCD_TO_BYTE (B);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
B	输入数据	BYTE	0-255	0	输入数据

输出变量

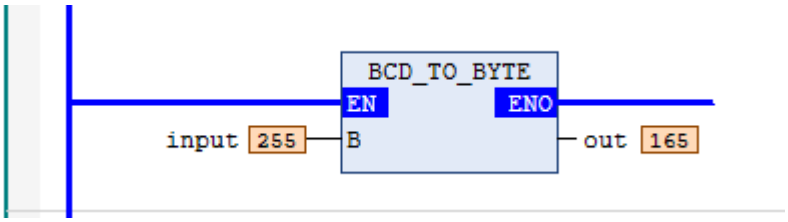
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	BYTE	0-255	0	输出数据

程序示例

ST:

```
out[165] := BCD_TO_BYTE(input[255]);
```

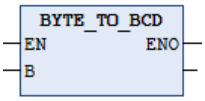
LD:



3.13.5 BYTE_TO_BCD

将源数据的 BYTE 型数据转换成 BCD 码。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
BYTE_TO_BCD	BYTE 型转 BCD 码	FC		Output:= BYTE_TO_BCD (B);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
B	输入数据	BYTE	0-255	0	输入数据

输出变量

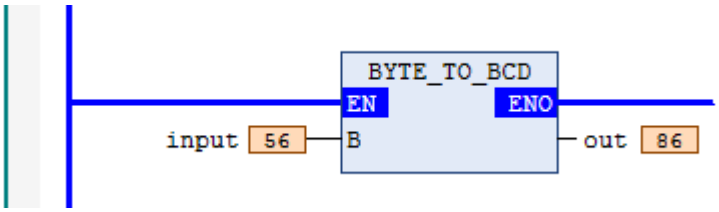
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	BYTE	0-255	0	输出数据

程序示例

ST:

```
out 86 := BYTE_TO_BCD(input 56 );
```

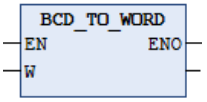
LD:



3.13.6 BCD_TO_WORD

将源数据的 BCD 码转换成 WORD 类型的数据。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
BCD_TO_WORD	BCD 码转 WORD 型	FC		Output:= BCD_TO_WORD (W);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
W	输入数据	WORD	0-FFFF	0	输入数据

输出变量

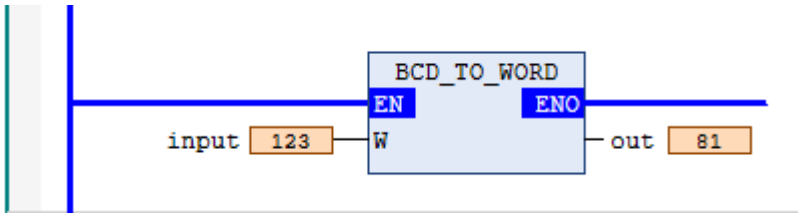
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	WORD	0-FFFF	0	输出数据

程序示例

ST:

```
out 81 := BCD_TO_WORD(input 123 );
```

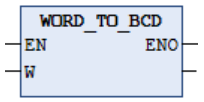
LD:



3.13.7 WORD_TO_BCD

将源数据的 WORD 型数据转换成 BCD 码。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
WORD_TO_BCD	WORD 型转 BCD 码	FC		Output:= WORD_TO_ BCD (W);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
W	输入数据	WORD	0-FFFF	0	输入数据

输出变量

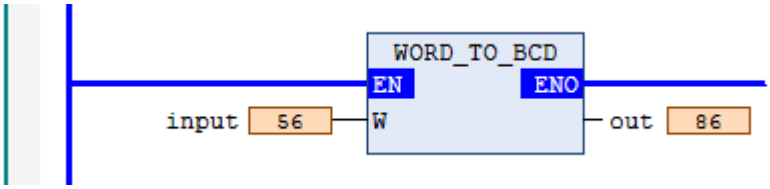
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	WORD	0-FFFF	0	输出数据

程序示例

ST:

```
out 86 := WORD_TO_BCD(input 56 );
```

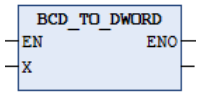
LD:



3.13.8 BCD_TO_DWORD

将源数据的 BCD 码转换成 DWORD 类型的数据。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
BCD _TO_DWORD	BCD 码转 DWORD 型	FC		Output:=BCD_TO_DWORD (W);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
X	输入数据	DWORD	0-FFFF FFFF	0	输入数据

输出变量

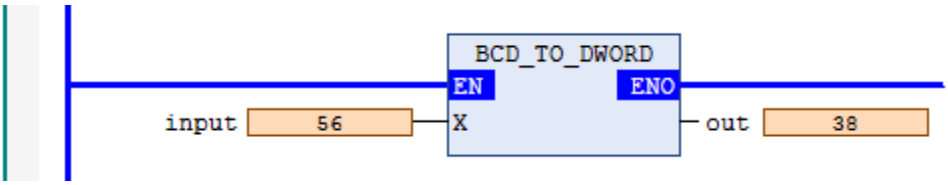
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	DWORD	0-FFFF FFFF	0	输出数据

程序示例

ST:

```
out 38 := BCD_TO_DWORD(input 56 );
```

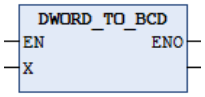
LD:



3.13.9 DWORD _TO_BCD

将源数据的 DWORD 型数据转换成 BCD 码。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
DWORD _TO_BCD	DWORD 型转 BCD 码	FC		Output:= DWORD _TO_ BCD (W);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
X	输入数据	DWORD	0-FFFF FFFF	0	输入数据

输出变量

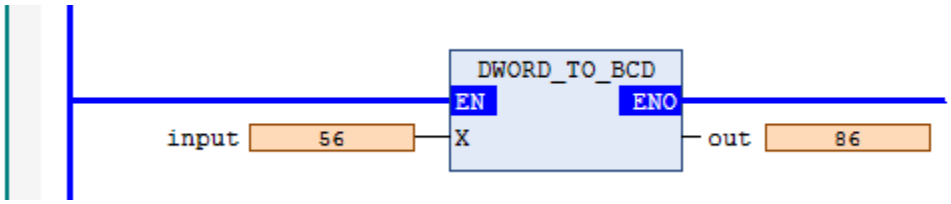
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	输出数据	DWORD	0-FFFF FFFF	0	输出数据

程序示例

ST:

```
out 86 := DWORD_TO_BCD(input 56);
```

LD:



3.14 模拟波形

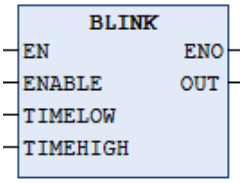
3.14.1 指令列表

指令类别	名称	FB/FC	功能
模拟波形	BLINK	FB	脉冲信号发生器
	GEN	FB	周期性信号发生器
	FREQ_MEASURE	FB	频率测量指令

3.14.2 BLINK

脉冲信号发生器，在时间周期 TIMEHIGH 内将输出设置为 TRUE，随后在 TIMELOW 周期内将输出设置为 FALSE。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
BLINK	脉冲信号发生器	FB		BLINK_0(ENABLE:= , TIMELOW:= , TIMEHIGH:= , OUT=>);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
ENABLE	使能	BOOL	0-255	0	TRUE 时，指令开始工作
TIMELOW	低电平时间	TIME	-	-	脉冲信号保持低电平的时间长短
TIMEHIGH	高电平时间	TIME	-	-	脉冲信号保持高电平的时间长短

输出变量

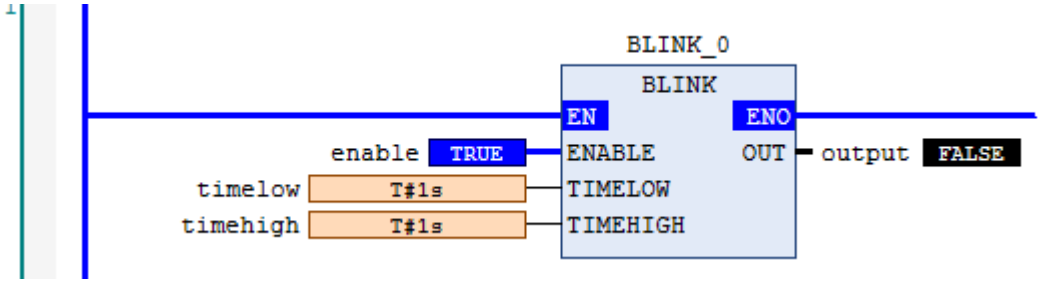
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	脉冲信号输出	BOOL	TRUE-FALSE	FALSE	脉冲信号输出

程序示例

ST:

```
BLINK_0(
        ENABLE TRUE := enable TRUE,
        TIMELOW T#1s := timelow T#1s,
        TIMEHIGH T#1s := timehigh T#1s,
        OUT FALSE => output FALSE);
```

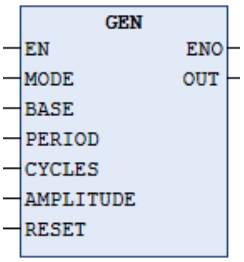
LD:



3.14.3 GEN

根据输入参数生成指定的函数波形。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
GEN	周期信号发生器	FB		GEN_0(MODE:= , BASE:= , PERIOD:= , CYCLES:= , AMPLITUDE:= , RESET:= , OUT=>);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
MODE	信号类型	GEN_MODE	-	TRIANGLE	GEN_MODE 类型的枚举变量
BASE	循环方式选择	BOOL	FALSE-TRUE	FALSE	TURE：以 PERIOD 为循环周期输出信号 FALSE：以 CYCLES 为循环次数输出信号
PERIOD	循环周期	TIME	-	TIME#1s0ms	脉冲信号保持高电平的时间长短
CYCLES	循环次数	INT	-32768-32768	1000	BASE 为 FALSE 时有效， 输出信号的循环次数
AMPLITUDE	信号振幅	INT	-32768-32768	0	输出信号的振幅值
RESET	复位	BOOL	FALSE-TRUE	FALSE	TRUE 时输出信号复位为 0，为 FALSE 时方可正常输出信号

输出变量

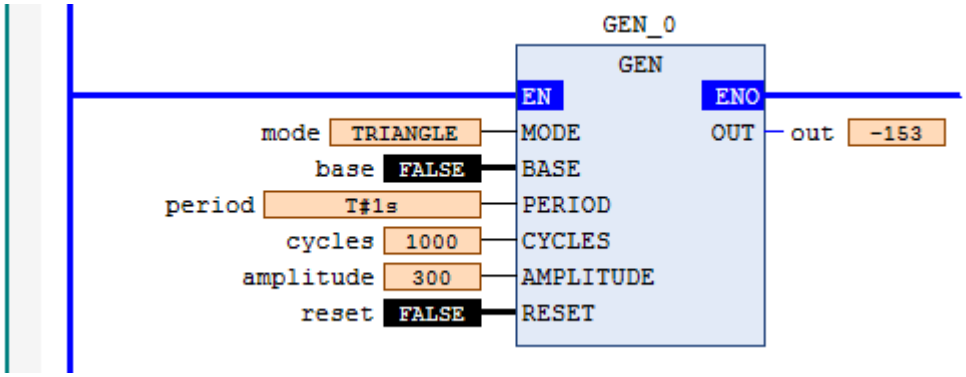
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	信号输出值	INT	-32768-32768	0	周期信号的输出值

程序示例

ST:

```
GEN_0(
    MODE TRIANGLE := mode TRIANGLE,
    BASE FALSE := base FALSE,
    PERIOD T#1s := period T#1s,
    CYCLES 1000 := cycles 1000,
    AMPLITUDE 300 := amplitude 300,
    RESET FALSE := reset FALSE,
    OUT 286 => out 286);
```

LD：



MODE 为 GEN_MODE 类型的一个枚举变量，有 7 个枚举变量，如：

0- TRIANGLE：输出三角波函数，幅值：-Amplitude~+Amplitude。

1- TRIANGLE_POS：输出三角波函数，幅值：0~+Amplitude。

2- SAWTOOTH_RISE：上升锯齿波函数，幅值：-Amplitude~+Amplitude。

3- SAWTOOTH_FALL：下降锯齿波函数，幅值：-Amplitude~+Amplitude。

4- RECTANGLE：矩形锯齿波函数转换，幅值：-Amplitude~+Amplitude。

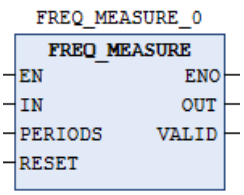
5- SINE：正弦函数。

6- COSINE：余弦函数。

3.14.4 FREQ_MEASURE

测量输入的布尔类型信号的（平均）频率值（Hz）。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
FREQ_MEASURE	频率测量指令	FB		FREQ_MEASURE_0(IN:= , PERIODS:= , RESET:= , OUT=> , VALID=>);	Util

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
IN	输入信号	BOOL	TRUE-FALSE	FALSE	IN 置 TURE，指令开始工作
PERIOD	测量周期	INT	-32768-32768	1	范围 1 到 10， 默认值为 1 输入信号两个上升沿的时间间隔为一个周期，将 N 个周期的频率值求平均值就得到输入信号的频率， 此参数即为求平均值运算的周期次数
RESET	复位	BOOL	FALSE-TRUE	FALSE	为 TRUE 时重新测量

输出变量

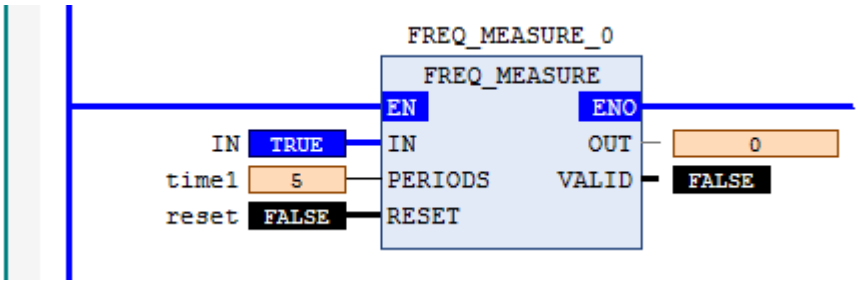
输出变量	名称	数据类型	有效范围	初始值	描述
OUT	频率输出	REAL	-	0	输入信号的频率值，单位为 Hz
VALID	运算标志	BOOL	FALSE-TRUE	FALSE	第一次运算完成后或运算错误时置 TRUE，其余时刻为 FALSE

程序示例

ST:

```
FREQ_MEASURE_0(
    IN TRUE := IN TRUE,
    PERIODS 5 := time1 5,
    RESET FALSE := reset FALSE,
    OUT=> ,
    VALID=> );
```

LD：



4 运动指令

4.1单轴状态监控

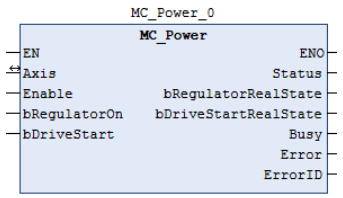
4.1.1 指令列表

指令类别	名称	FB/FC	功能
单轴状态监控	MC_Power	FB	轴使能
	MC_Reset	FB	复位轴
	MC_ReadStatus	FB	读轴状态
	MC_ReadAxisError	FB	读轴错误
	MC_ReadParameter	FB	读参数
	MC_ReadBoolParameter	FB	读布尔参数
	MC_WriteParameter	FB	写参数
	MC_WriteBoolParameter	FB	写布尔参数
	MC_ReadActualPosition	FB	读实际位置
	MC_ReadActualVelocity	FB	读实际速度
	MC_ReadActualTorque	FB	读实际转矩
	MC_SetPosition	FB	设置位置
	SMC_ReadSetPosition	FB	读取设置位置
	SMC_ReadFBError	FB	读取历史错误信息
	SMC_ClearFBError	FB	清除历史错误信息
	SMC_ErrorString	FB	错误码到错误信息

4.1.2 MC_Power

本指令用于使能指定轴，使轴进入可运行状态或退出可运行状态，也叫轴使能。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_Power	轴使能	FB		MC_Power_0(Axis:= , Enable:= , bRegulatorOn:= , bDriveStart:= , Status=> , bRegulatorRealState=> , bDriveStartRealState=> , Busy=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	有效	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活功能块的处理
bRegulatorOn	使能	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以激活轴的使能状态
bDriveStart	驱动启用	BOOL	TRUE-FALSE	FALSE	必须设置为 TRUE，以关闭轴的紧急停止处理

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Status	可运行	BOOL	TRUE-FALSE	-	轴准备好则为 TRUE
bRegulator RealState	使能有效	BOOL	TRUE-FALSE	-	轴使能的有效状态
bDriveStar tRealState	驱动可使用	BOOL	TRUE-FALSE	-	驱动器没有被快速停止机制中断为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	功能块的处理没有完成为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	发生异常时为 TRUE
ErrorID	错误代码	SMC_ERROR	-	-	正常时数值为 0，发生异常

					时，输出报错代码

功能说明

1）将 Enable 设为 TRUE 时，该功能块进入可运行状态；将 Enable 设为 FALSE，功能块不运行，但是可以执行 MC_Power 指令、MC_Reset（轴错误复位）指令。

2）将 bRegulatorOn 设为 TRUE，如果轴没有错误则轴状态为 standstill。如果轴有错误，轴状态为 errorstop。

3）将 bRegulatorOn 设为 FALSE，轴状态为 Disabled，表明轴没有做好运动准备。

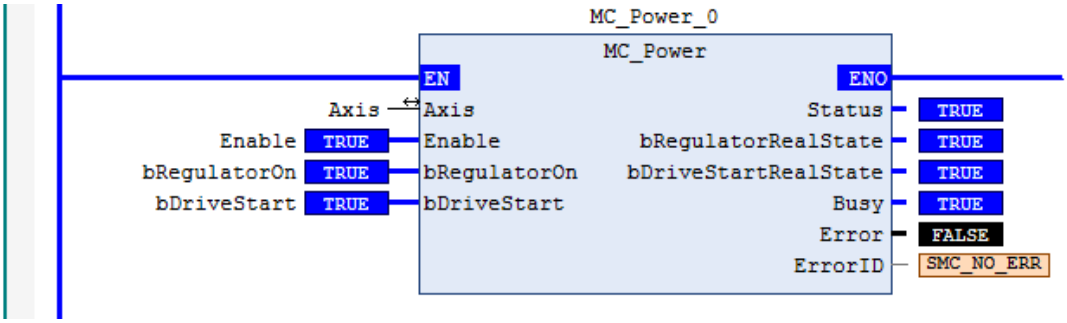
4）将 bDriveStart 设为 FALSE，轴急停。

程序示例

ST：当 Enable，RegulatorOn，DriveStart 为 TRUE 时，Status 变为 TRUE，表示轴 Axis 已使能

```
MC_Power_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    bRegulatorOn TRUE := bRegulatorOn TRUE,
    bDriveStart TRUE := bDriveStart TRUE,
    Status TRUE => Status TRUE,
    bRegulatorRealState TRUE => bRegulatorRealState TRUE,
    bDriveStartRealState TRUE => bDriveStartRealState TRUE,
    Busy TRUE => Busy TRUE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0 );
```

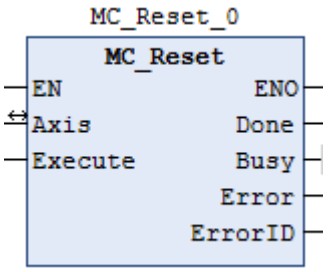
LD：当 Enable，RegulatorOn，DriveStart 为 TRUE 时，Status 变为 TRUE，表示轴 Axis 已使能



4.1.3 MC_Reset

本指令用于复位（清除）轴的错误。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_Reset	轴复位	FB		MC_Reset_0(Axis:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	输入值的上升沿将启动该功能块的执行

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	-	如果复位被执行则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

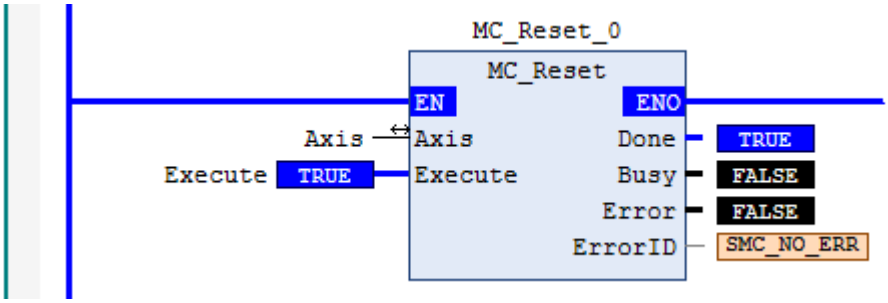
轴在报错后，将不能继续执行运动指令，只有调用 MC_Reset 指令清除错误之后才能操作轴。MC_Reset 便是用于将轴的状态从 ErrorStop 状态转换成 StandStill 状态，消除轴报错状态，变成可执行状态。

程序示例

ST：当 Execute 变为 TRUE 时，Done 变为 TRUE，表示轴 Axis 错误已清除。

```
MC_Reset_0(
    Axis:= Axis,
    Execute TRUE := Execute TRUE,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0);
```

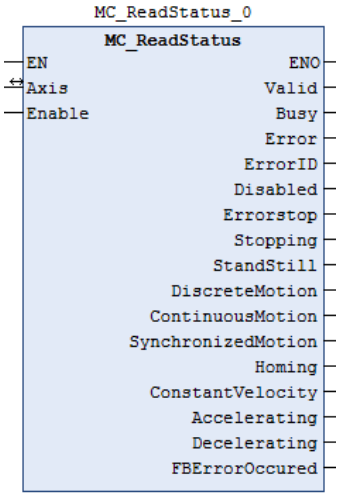
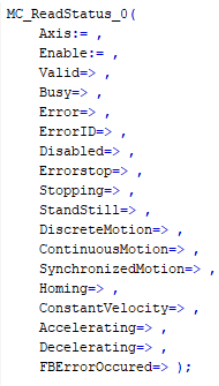
LD：当 Execute 变为 TRUE 时，Done 变为 TRUE，表示轴 Axis 错误已清除。



4.1.4 MC_ReadStatus

本指令用于读取轴的详细状态。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_ReadStatus	读轴状态	FB			SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	有效	BOOL	TRUE-FALSE	FALSE	设置为 TRUE 时激活功能块

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效	BOOL	TRUE-FALSE	-	如果轴准备好则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
Disabled	未使能	BOOL	TRUE-FALSE	-	如果轴状态为 Disabled 则为 TRUE
Errorstop	报错	BOOL	TRUE-FALSE	-	如果轴状态为 Errorstop 则为 TRUE

Stopping	停止中	BOOL	TRUE-FALSE	-	如果轴状态为 Stopping 则为 TRUE
StandStill	准备好	BOOL	TRUE-FALSE	-	如果轴状态为 Standstill 则为 TRUE
DiscreteMotion	点位运动	BOOL	TRUE-FALSE	-	如果轴状态为 DiscreteMotion 则为 TRUE
ContinuousMotion	连续运动	BOOL	TRUE-FALSE	-	如果轴状态为 ContinuousMotion 则为 TRUE
SynchronizedMotion	同步运动	BOOL	TRUE-FALSE	-	如果轴状态为 SynchronizedMotion 则为 TRUE
Homing	回零中	BOOL	TRUE-FALSE	-	如果轴状态为 Homing 则为 TRUE
ConstantVelocity	恒速运动	BOOL	TRUE-FALSE	-	如果电机以恒定速度移动则为 TRUE
Accelerating	加速中	BOOL	TRUE-FALSE	-	如果电机正在加速则为 TRUE
Decelerating	减速中	BOOL	TRUE-FALSE	-	如果电机正在减速则为 TRUE
FBErrorOccured	FB 报错	BOOL	TRUE-FALSE	-	如果功能块错误被探测到并且还没有由 SMC_ClearFBError 清除，则为 TRUE

功能说明

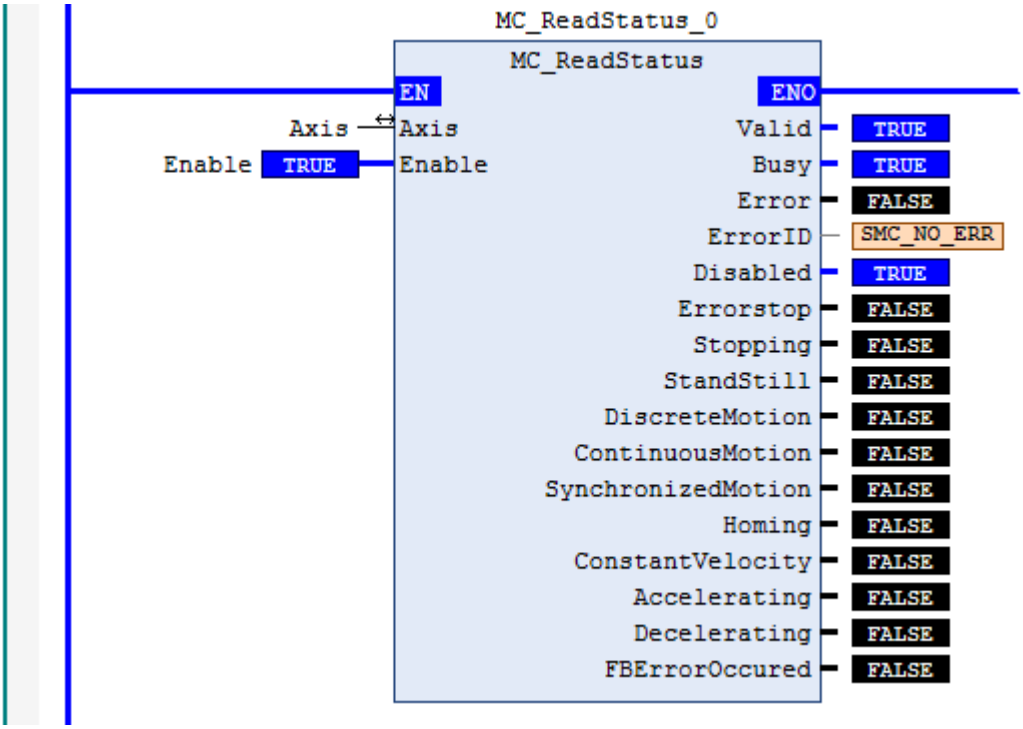
可以用来读取轴的状态，并将状态引用到对应的程序步骤之中。Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的状态。

程序示例

ST：当 Enable 为 TRUE 时，持续读取轴的当前状态。

```
MC_ReadStatus_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    Valid TRUE => Valid TRUE,
    Busy TRUE => Busy TRUE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0,
    Disabled TRUE => Disabled TRUE,
    Errorstop FALSE => Errorstop FALSE,
    Stopping FALSE => Stopping FALSE,
    StandStill FALSE => StandStill FALSE,
    DiscreteMotion FALSE => DiscreteMotion FALSE,
    ContinuousMotion FALSE => ContinuousMotion FALSE,
    SynchronizedMotion FALSE => SynchronizedMotion FALSE,
    Homing FALSE => Homing FALSE,
    ConstantVelocity FALSE => ConstantVelocity FALSE,
    Accelerating FALSE => Accelerating FALSE,
    Decelerating FALSE => Decelerating FALSE,
    FBErrorOccured FALSE => FBErrorOccured FALSE);
```

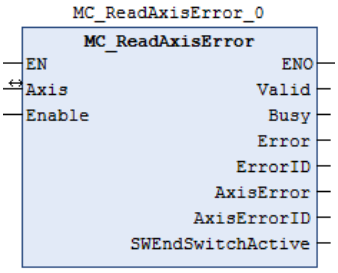
LD：当 Enable 为 TRUE 时，持续读取轴的当前状态。



4.1.5 MC_ReadAxisError

本指令用于读取轴的错误。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_ReadAxisError	读轴错误	FB		MC_ReadAxisError_0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , AxisError=> , AxisErrorID=> , SWEndSwitchActive=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	有效	BOOL	TRUE-FALSE	FALSE	设置为 TRUE 时激活功能块

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效	BOOL	TRUE-FALSE	-	如果轴准备好则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
AxisError	轴报错	BOOL	TRUE-FALSE	-	轴错误的标志
AxisErrorID	轴错误值	DWORD	-	-	轴错误值
SWEndSwitchActive	软限位	BOOL	TRUE-FALSE	-	如果超过软限位则为 TRUE

功能说明

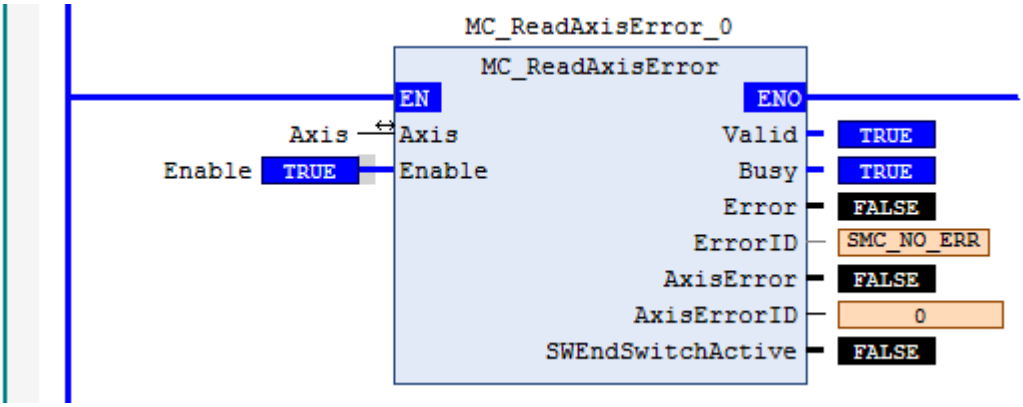
可以用来读取轴的错误，并将错误状态引用到对应的程序步骤之中。Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的状态。

程序示例

ST：当 Enable 为 TRUE 时，持续读取轴的错误。

```
MC_ReadAxisError_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    Valid TRUE => Valid TRUE,
    Busy TRUE => Busy TRUE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0,
    AxisError FALSE => AxisError FALSE,
    AxisErrorID 0 => AxisErrorID 0,
    SWEndSwitchActive FALSE => SWEndSwitchActive FALSE);
```

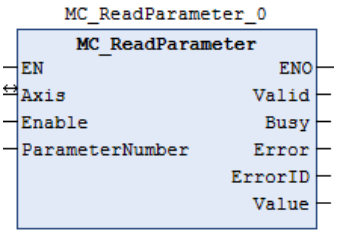
LD：当 Enable 为 TRUE 时，持续读取轴的错误。



4.1.6 MC_ReadParameter

本指令用于读取指定的参数值。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_ReadParameter	读轴参数	FB		MC_ReadParameter_0(Axis:= , Enable:= , ParameterNumber:= , Valid=> , Busy=> , Error=> , ErrorID=> , Value=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	有效	BOOL	TRUE-FALSE	FALSE	设置为 TRUE 时激活功能块
ParameteNumber	轴参数号	DINT	遵照数据类型	0	访问轴参数的索引和子索引和序号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效	BOOL	TRUE-FALSE	-	如果参数已读取则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
Value	参数值	LREAL	遵照数据类型	-	读取参数的值

功能说明

1）可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到 ParameterNumber 参数具体的参数序号所代表的参数用途。

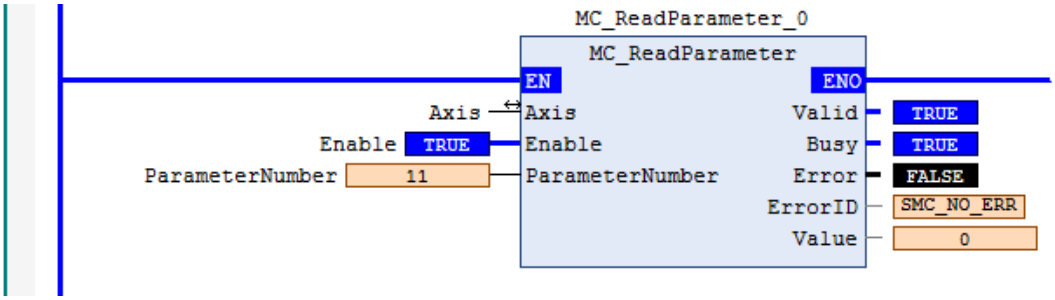
2）Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的参数。

程序示例

ST：当 Enable 为 TRUE 时，持续读取轴的参数。

```
MC_ReadParameter_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    ParameterNumber 11 := ParameterNumber 11,
    Valid TRUE => Valid TRUE,
    Busy TRUE => Busy TRUE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0,
    Value 0 => Value 0);
```

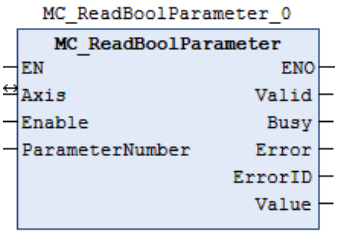
LD：当 Enable 为 TRUE 时，持续读取轴的参数。



4.1.7 MC_ReadBoolParameter

本指令用于读取指定的 BOOL 型变量的值。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_ReadBoolParameter	读布尔参数	FB		MC_ReadBoolParameter_0(Axis:= , Enable:= , ParameterNumber:= , Valid=> , Busy=> , Error=> , ErrorID=> , Value=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	有效	BOOL	TRUE-FALSE	FALSE	设置为 TRUE 时激活功能块
ParameteNumber	轴参数号	DINT	遵照数据类型	0	访问轴参数的索引和子索引和序号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效	BOOL	TRUE-FALSE	-	如果参数已读取则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
Value	参数值	BOOL	TRUE-FALSE	-	读取参数的值

功能说明

1）可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到 ParameterNumber 参数具体的参数序号所代表的参数用途。

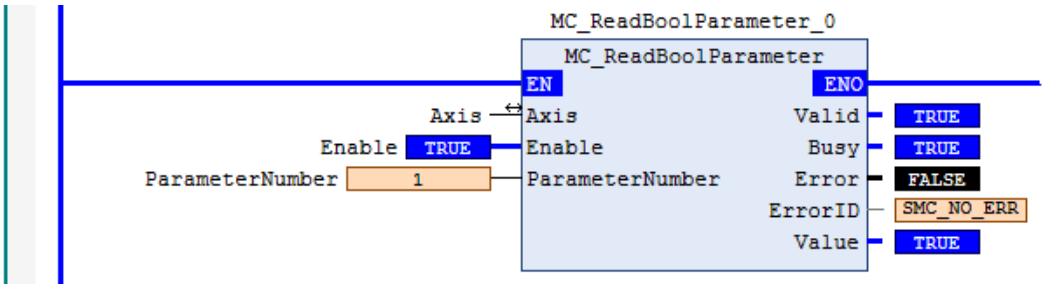
2）Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的参数。

程序示例

ST：当 Enable 为 TRUE 时，持续读取轴的参数。

```
MC_ReadBoolParameter_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    ParameterNumber 1 := ParameterNumber 1,
    Valid TRUE => Valid TRUE,
    Busy TRUE => Busy TRUE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0,
    Value TRUE => Value TRUE);
```

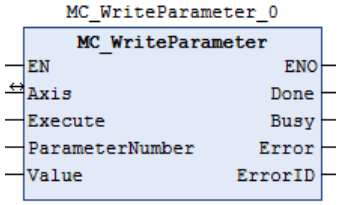
LD：当 Enable 为 TRUE 时，持续读取轴的参数。



4.1.8 MC_WriteParameter

本指令用于写入指定的参数值。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_WriteParameter	写轴参数	FB		MC_WriteParameter_0(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿激活功能块
ParameteNumber	轴参数号	DINT	遵照数据类型	0	访问轴参数的索引和子索引和序号
Value	参数值	LREAL	遵照数据类型	0	需要写入的值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	-	如果参数已写入则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到 ParameterNumber 参数具体的参数序号所代表的参数用途。

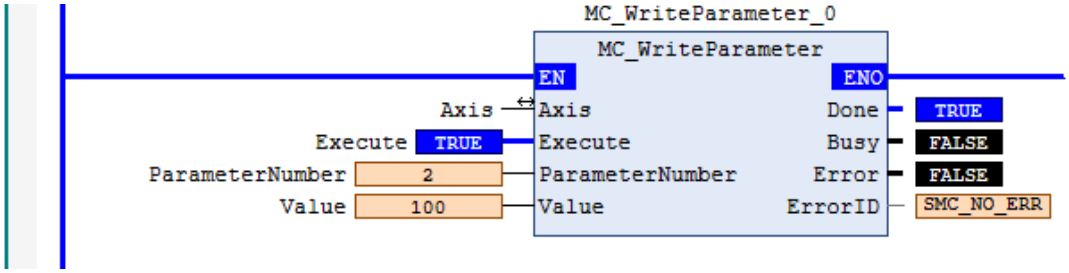
2）Execute 为上升沿时，写入指定的轴参数。

程序示例

ST：当 Execute 变为 TRUE 时，写入指定的轴参数。

```
MC_WriteParameter_0(
    Axis:= Axis,
    Execute TRUE := Execute TRUE,
    ParameterNumber 2 := ParameterNumber 2,
    Value 100 := Value 100,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0);
```

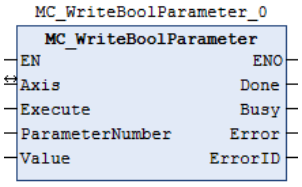
LD：当 Execute 变为 TRUE 时，写入指定的轴参数。



4.1.9 MC_WriteBoolParameter

本指令用于写入 BOOL 类型的参数值。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_WriteBoolParameter	写轴布尔参数	FB		MC_WriteBoolParameter_0(Axis:= , Execute:= , ParameterNumber:= , Value:= , Done=> , Busy=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿激活功能块
ParameteNumber	轴参数号	DINT	遵照数据类型	0	访问轴参数的索引和子索引和序号
Value	参数值	BOOL	TRUE-FALSE	0	需要写入的值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	-	如果参数已写入则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）可以在 SM3_Basic 库的 AXIS_REF_SM3 之中，找到 ParameterNumber 参数具体的参数序号所代表的参数用途。

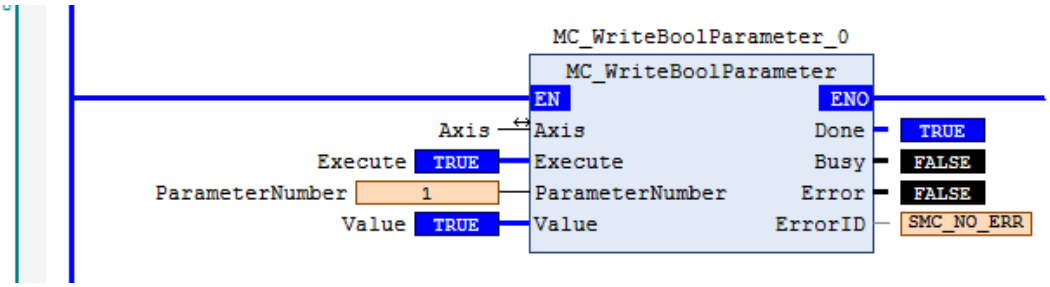
2）Execute 为上升沿时，写入指定的轴参数。

程序示例

ST：当 Execute 变为 TRUE 时，写入指定的轴参数。

```
MC_WriteBoolParameter_0(
    Axis:= Axis,
    Execute TRUE := Execute TRUE,
    ParameterNumber 1 := ParameterNumber 1,
    Value FALSE := Value FALSE,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0);
```

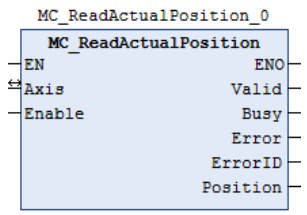
LD：当 Execute 变为 TRUE 时，写入指定的轴参数。



4.1.10 MC_ReadActualPosition

本指令用于读取轴当前实际位置值。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_ReadActualPosition	读轴实际位置	FB		MC_ReadActualPosition_0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Position=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	设置为 TRUE 激活功能块

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效	BOOL	TRUE-FALSE	-	如果输出值有效则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
Position	位置	LREAL	遵照数据类型	-	新的绝对位置（以轴的单位表达[u]）

功能说明

1）这个指令用来读取驱动器中轴的实际位置，即 fActPosition。

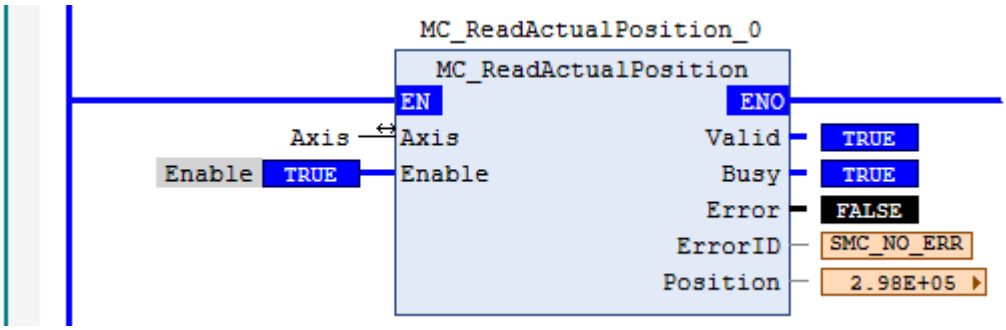
2）Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的实际位置。

程序示例

ST：当 Enable 为 TRUE 时，持续读取轴的实际位置。

```
MC_ReadActualPosition_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    Valid TRUE => Valid TRUE,
    Busy TRUE => Busy TRUE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0,
    Position 2.98E+05 => Position 2.98E+05 );
```

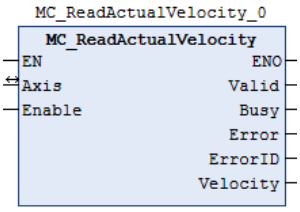
LD：当 Enable 为 TRUE 时，持续读取轴的实际位置。



4.1.11 MC_ReadActualVelocity

本指令用于读取轴当前实际速度值。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_ReadActualVelocity	读轴实际速度	FB		MC_ReadActualVelocity_0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Velocity=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	设置为 TRUE 激活功能块

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效	BOOL	TRUE-FALSE	-	如果输出值有效则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
Velocity	速度	LREAL	遵照数据类型	-	实际速度的值（以[计数单位/秒]表达）

功能说明

1）这个指令用来读取驱动器中轴的实际速度，即 fActVelocity。

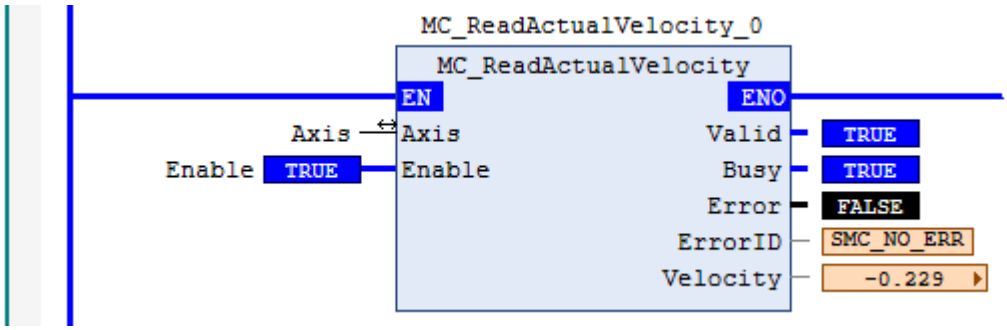
2）Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的实际速度。

程序示例

ST：当 Enable 为 TRUE 时，持续读取轴的实际速度。

```
MC_ReadActualVelocity_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    Valid TRUE => Valid TRUE,
    Busy TRUE => Busy TRUE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0,
    Velocity 0.229 => Velocity 0.229 );
```

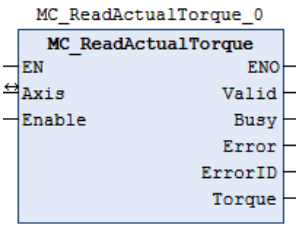
LD：当 Enable 为 TRUE 时，持续读取轴的实际速度。



4.1.12 MC_ReadActualTorque

本指令用于读取轴当前实际转矩值。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_ReadActualTorque	读轴实际力矩	FB		MC_ReadActualTorque_0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Torque=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	设置为 TRUE 激活功能块

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效	BOOL	TRUE-FALSE	-	如果输出值有效则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
Torque	力矩	LREAL	遵照数据类型	-	实际转矩或力矩的值（以计数单位表达）

功能说明

1）这个指令用来读取驱动器中轴的实际力矩。

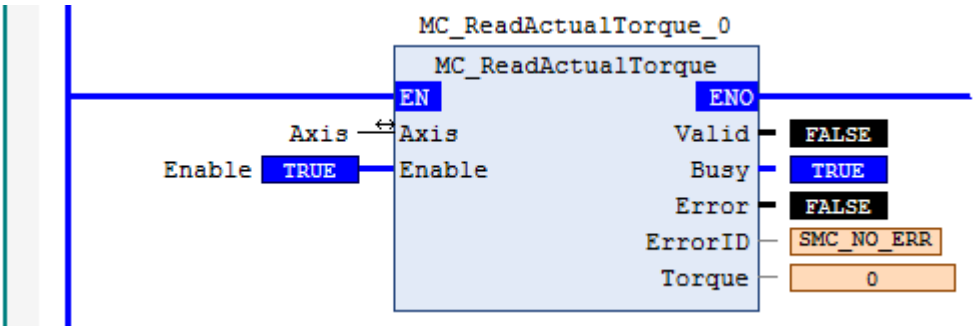
2）Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的实际转矩。

程序示例

ST：当 Enable 为 TRUE 时，持续读取轴的实际转矩。

```
MC_ReadActualTorque_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    Valid FALSE => Valid FALSE,
    Busy TRUE => Busy TRUE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0,
    Torque 0 => Torque 0);
```

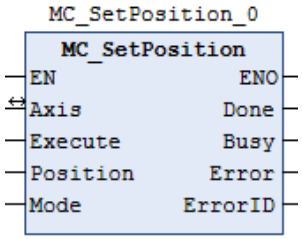
LD：当 Enable 为 TRUE 时，持续读取轴的实际转矩。



4.1.13 MC_SetPosition

本指令用于设置轴当前的位置（不会引起轴的运动）。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_SetPosition	设置轴位置	FB		MC_SetPosition_0(Axis:= , Execute:= , Position:= , Mode:= , Done=> , Busy=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	输入值的上升沿将启动该功能块的执行
Position	位置	LREAL	遵照数据类型	0	位置单位[u]（表示“距离”如果 Mode = RELATIVE）
Mode	模式	BOOL	TRUE-FALSE	FALSE	TRUE =相对，FALSE =绝对

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	-	功能块执行完成则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

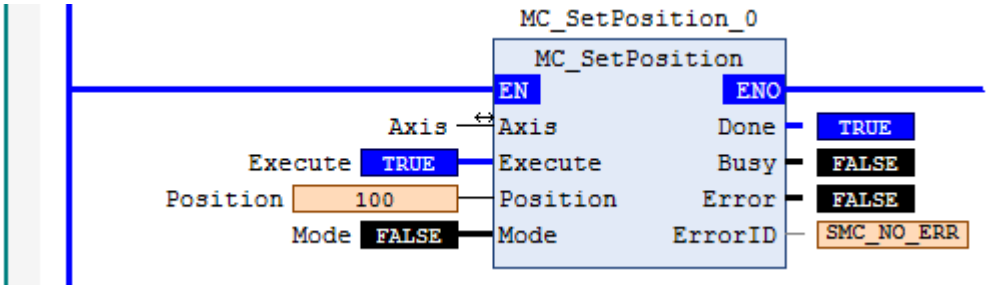
Execute 的上升沿将触发这个指令的执行，指令执行后会根据 Mode 设置模式，将 Position 参数设置到轴的当前位置，但不会引起轴的运动。

程序示例

ST：当 Execute 变为 TRUE 时，将 Position 参数设置到轴的当前位置。

```
MC_SetPosition_0(
    Axis:= Axis,
    Execute TRUE := Execute TRUE,
    Position 0 := Position 0,
    Mode FALSE := Mode FALSE,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0);
```

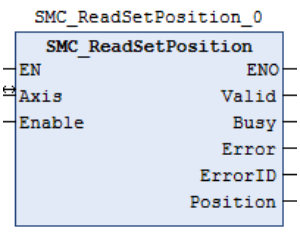
LD：当 Execute 变为 TRUE 时，将 Position 参数设置到轴的当前位置。



4.1.14 SMC_ReadSetPosition

本指令用于读取当前轴的设置位置。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SMC_ReadSetPosition	读取设置位置	FB		SMC_ReadSetPosition_0(Axis:= , Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , Position=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	设置为 TRUE 激活功能块

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效	BOOL	TRUE-FALSE	-	如果输出值有效则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	功能块执行错误
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
Position	位置	LREAL	遵照数据类型	-	设置的位置值

功能说明

1）这个指令用来读取轴的设置位置。

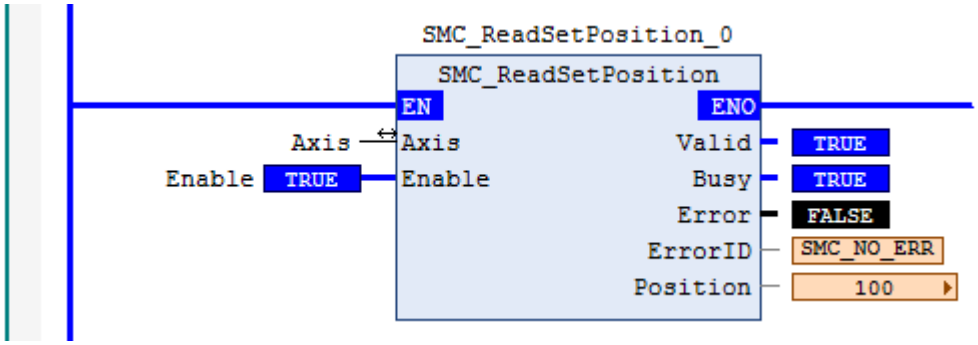
2）Enable 为 false，则所有状态输出将被置为 false；Enable 为 true 时，持续读取轴的设置位置。

程序示例

ST：当 Enable 为 TRUE 时，持续读取轴的设置位置。

```
SMC_ReadSetPosition_0(
    Axis:= Axis,
    Enable TRUE := Enable TRUE,
    Valid TRUE => Valid TRUE,
    Busy TRUE => Busy TRUE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID 0,
    Position 100 => Position 100 );
```

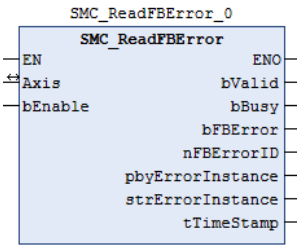
LD：当 Enable 为 TRUE 时，持续读取轴的设置位置。



4.1.15 SMC_ReadFBError

本指令用于读取轴功能块的错误信息。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SMC_ReadFBError	读功能块错误信息	FB		SMC_ReadFBError_0(Axis:= , bEnable:= , bValid=> , bBusy=> , bFBError=> , nFBErrorID=> , pbyErrorInstance=> , strErrorInstance=> , tTimeStamp=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	使能	BOOL	TRUE-FALSE	FALSE	设置为 TRUE 激活功能块

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bValid	获取标志	BOOL	TRUE-FALSE	-	如果参数有效则为 TRUE
bBusy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
bFBError	错误	BOOL	TRUE-FALSE	-	功能块执行错误
nFBErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
pbyErrorInstance	错误指针	POINTER TO BYTE	-	-	指针指向报错的 FB
strErrorInstance		STRING	-	-	指向错误功能块（程序，子程序，功能块）
tTimeStamp	时间戳	TIME	-	-	错误发生的时间戳

功能说明

1）bEnable 为 TRUE 时，无错误则 bValid、bBusy 输出为 TURE；有功能块报警则 bFBError 输出为 true。

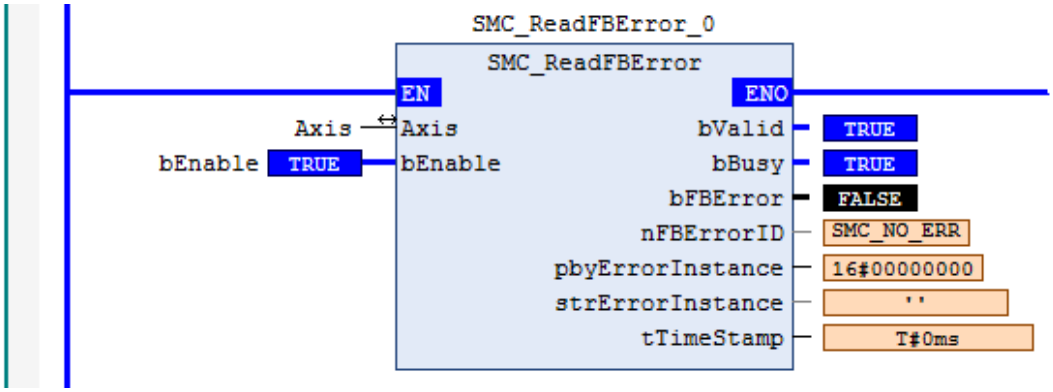
2）bEnable 变为 FALSE，则 bValid、bBusy 输出为 FALSE。

程序示例

ST：当 Enable 为 TRUE 时，读取轴功能块错误信息。

```
SMC_ReadFBError_0(
    Axis:= Axis,
    bEnable TRUE := bEnable TRUE,
    bValid TRUE => bValid TRUE,
    bBusy TRUE => bBusy TRUE,
    bFBError FALSE => bFBError FALSE,
    nFBErrorID SMC_NO_ERR => nFBErrorID SMC_NO_ERR,
    pbyErrorInstance 16#00000000 => pbyErrorInstance 16#00000000,
    strErrorInstance '' => strErrorInstance '',
    tTimeStamp T#0ms => tTimeStamp T#0ms);
```

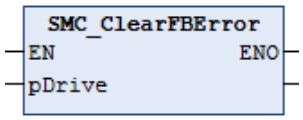
LD：当 Enable 为 TRUE 时，读取轴功能块错误信息。



4.1.16 SMC_ClearFBError

本指令用于清除功能块的历史错误信息。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SMC_ClearFBError	清除功能块错误信息	FC		SMC_ClearFBError(pDrive:=);	SM3_Basic

相关变量

输入变量

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

功能说明

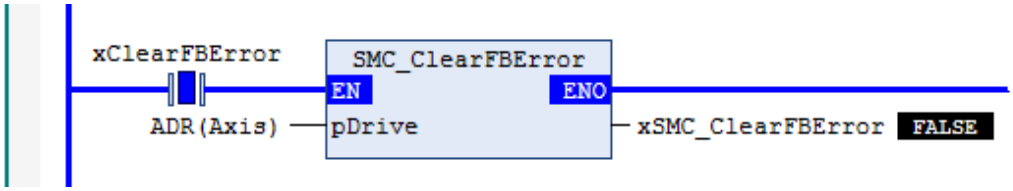
当轴出现错误，调用复位功能块将轴复位后，需要调用该功能块清除轴的历史错误状态。

程序示例

ST：当 xClearFBError 变为 TRUE 时，清除轴 Axis 的历史错误状态。

```
IF xClearFBError TRUE THEN
    xSMC_ClearFBError FALSE := SMC_ClearFBError(pDrive:= ADR(Axis));
END_IF
```

LD：当 xClearFBError 变为 TRUE 时，清除轴 Axis 的历史错误状态。



4.2 单轴运动控制

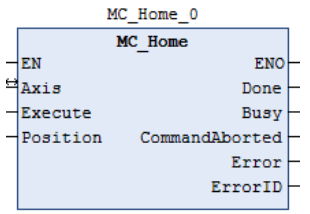
4.2.1 指令列表

指令类别	名称	FB/FC	功能
单轴运动控制	MC_Home	FB	回零
	MC_MoveAbsolute	FB	绝对运动
	MC_MoveRelative	FB	相对运动
	MC_MoveVelocity	FB	定速运动
	MC_Stop	FB	停止
	MC_Halt	FB	暂停
	MC_Jog	FB	点动运动
	MC_MoveAdditive	FB	附加运动
	MC_MoveSuperImposed	FB	叠加运动
	MC_PositionProfile	FB	位置规划
	MC_VelocityProfile	FB	速度规划
	MC_AccelerationProfile	FB	加速度规划
	SMC_Homing	FB	上位机回原
	SMC_Inch	FB	寸动

4.2.2 MC_Home

本指令用于执行总线电机回零动作，具体回零过程由总线驱动设计的回零模式决定。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_Home	总线轴回原	FB		MC_Home_0(Axis:= , Execute:= , Position:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
Position	位置	LREAL	遵照数据类型	0	驱动器的回零偏置 16#607C

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	-	如果回零已完成则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）这是总线驱动的回零运动指令，当所控制的运动轴是一个总线轴时，需要调用这个指令，以实现运动轴的回零。

2）此回零指令的回零模式由连接在总线上的从站决定，控制端只是将需要的回零参数发给驱动器，具体的回零动作，在驱动器端完成。

3）一般驱动器的回零偏移地址为 16#607C，各驱动器的处理不同，具体请参考该驱动器的回零流程及回零参数说明。

4）在执行回零之前，需要配置总线驱动的回零参数，如回零模式、速度、加速度等。总线驱动回零需要配置哪些参数，请参考所使用驱动的手册。

5）一般的总线驱动回零，需要设置索引和子索引的数据如下表所示。

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

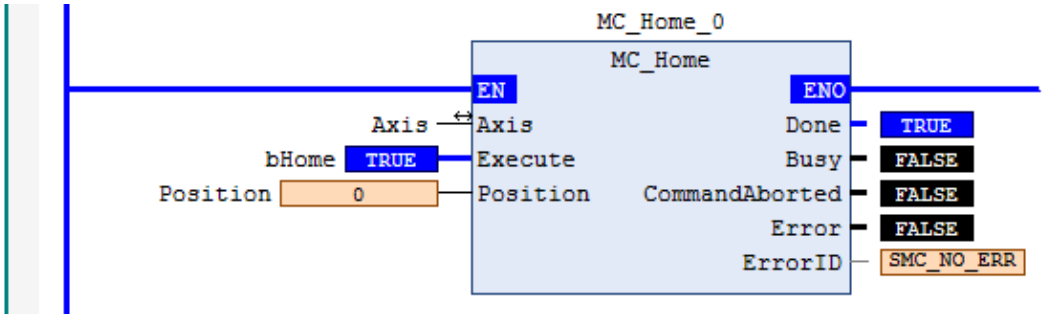
6）修改回零模式的三种方式：1.控制器通过 SDO 服务向驱动器写入回零参数；2.通过配置 PDO 参数并映射相关回零参数进行回零设置；3.通过配置启动参数向驱动器写入回零参数。

程序示例

ST：当 Execute 变为 TRUE 时，执行回零指令。

```
MC_Home_0(
    Axis:= Axis,
    Execute TRUE := bHome TRUE,
    Position 0 := Position 0,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

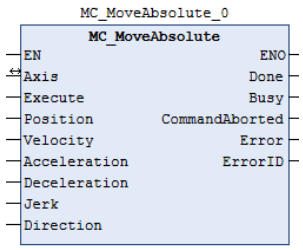
LD：当 Execute 变为 TRUE 时，执行回零指令。



4.2.3 MC_MoveAbsolute

本指令用于实现将控制轴按照设定参数，运动到指定的绝对位置。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_MoveAbsolute	绝对位置	FB		MC_MoveAbsolute_0(Axis:= , Execute:= , Position:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Direction:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
Position	位置	LREAL	遵照数据类型	0	运动的目标位置[u]（负或正）
Velocity	目标速度	LREAL	遵照数据类型	0	速度最大值（总是为正）[u/s]
Acceleration	目标加速度	LREAL	遵照数据类型	0	加速度值（总是为正）[u/s2]
Deceleration	目标减速度	LREAL	遵照数据类型	0	减速度值（总是为正）[u/s2]
Jerk	目标加加速度	LREAL	遵照数据类型	0	加加速度值（总是为正）[u/s3]
Direction	方向	MC_Direction	[-1,3]	shortest	3:fastest，最快； 2:current，当前； 1: Positive，正； 0:shortest，最短； -1:Negative，负；

输出变量

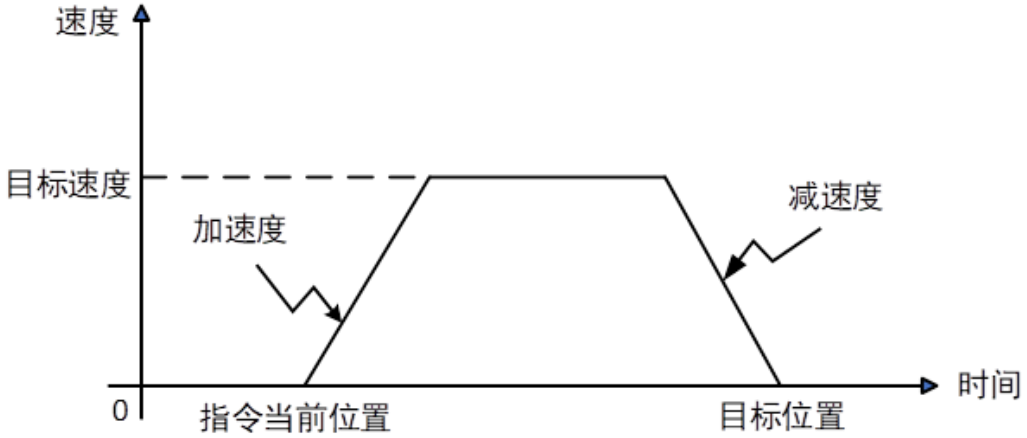
输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	-	指令执行完成时

					为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）如果是线性轴的绝对点位运动，方向值将被忽略。

2）当速度曲线是梯形曲线时，指令执行时的“速度-时间”曲线如下图所示。



3）如果距离过短，可能达不到目标速度。

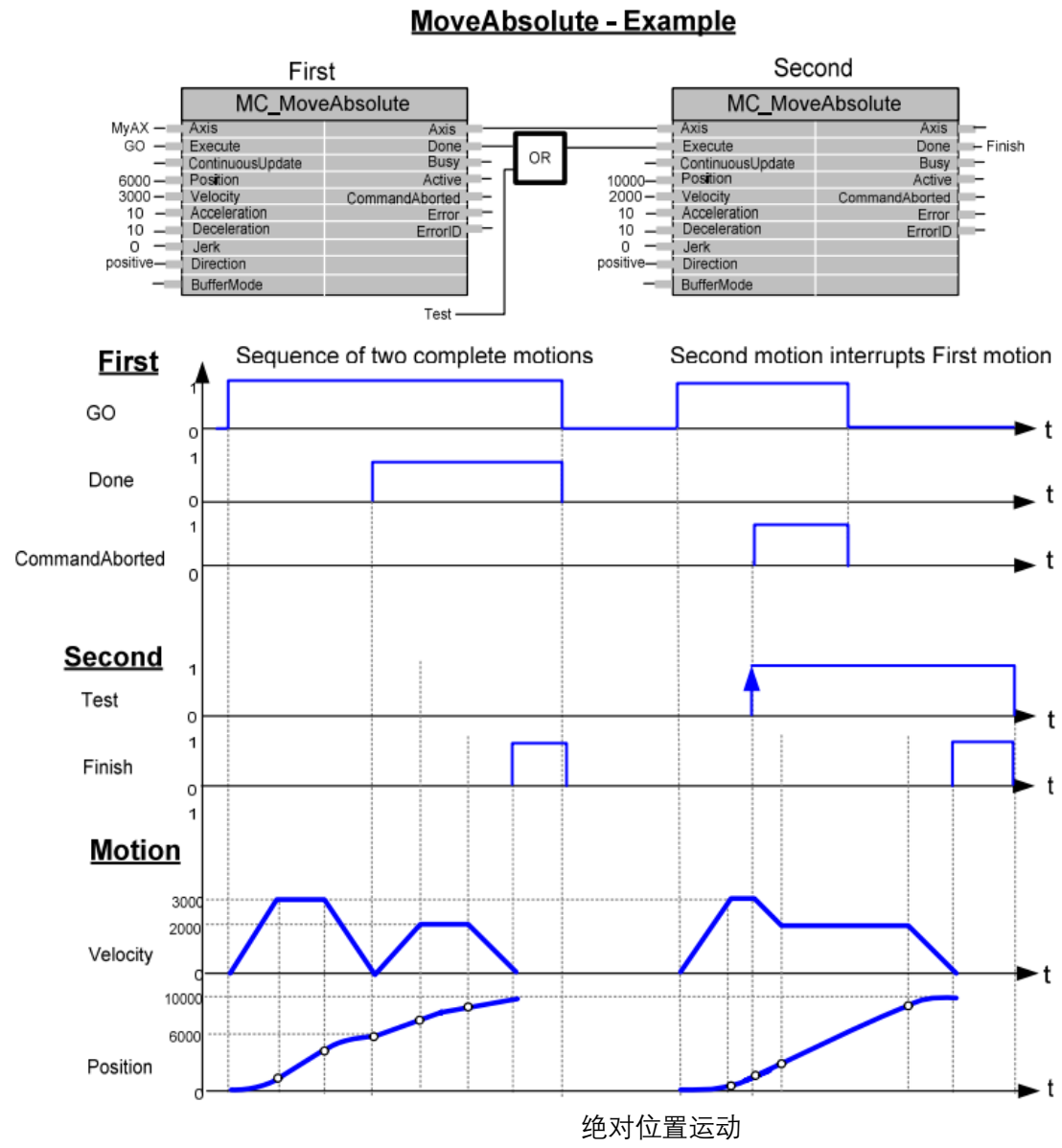
例 MC_MoveAbsolute

实现功能：

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

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

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



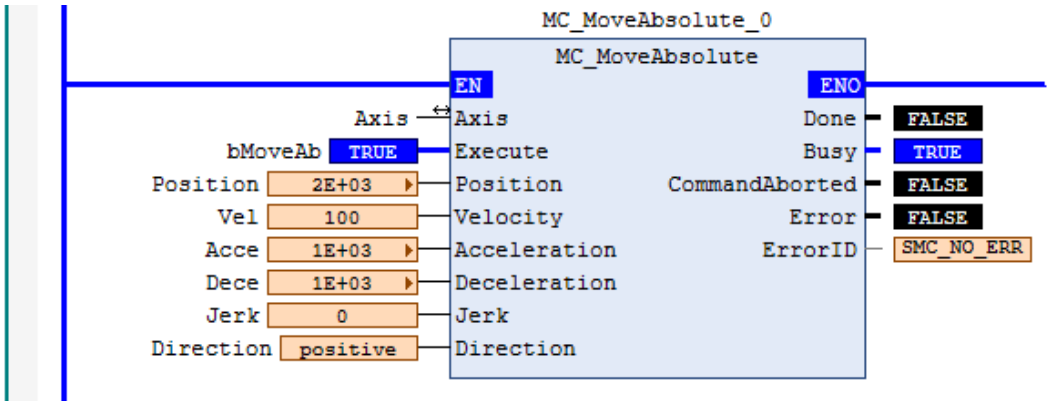
绝对位置运动

程序示例

ST：当 bMoveAb 变为 TRUE 时，执行绝对定位指令。

```
MC_MoveAbsolute_0(
    Axis:= Axis,
    Execute TRUE := bMoveAb TRUE,
    Position 3E+03 := Position 3E+03,
    Velocity 100 := Vel 100,
    Acceleration 1E+03 := Acce 1E+03,
    Deceleration 1E+03 := Dece 1E+03,
    Jerk 0 := Jerk 0,
    Direction shortest := Direction shortest,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

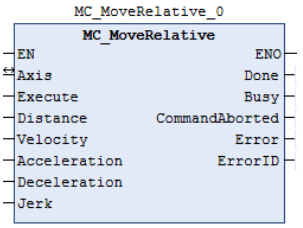
LD：当 bMoveAb 变为 TRUE 时，执行绝对定位指令。


MC_MoveAbsolute_0
MC_MoveAbsolute
EN
ENO
Axis
Axis
bMoveAb
TRUE
Execute
Position
2E+03
Position
Vel
100
Velocity
Acce
1E+03
Acceleration
Dece
1E+03
Deceleration
Jerk
0
Jerk
Direction
positive
Direction
Done
FALSE
Busy
TRUE
CommandAborted
FALSE
Error
FALSE
ErrorID
SMC_NO_ERR

4.2.4 MC_MoveRelative

本指令用于实现将控制轴按照设定参数，运动一段相对距离。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_MoveRelative	相对位置	FB		MC_MoveRelative_0(Axis:= , Execute:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
Distance	距离	LREAL	遵照数据类型	0	目标位置与当前位置的相对距离[u]（负或正）
Velocity	目标速度	LREAL	遵照数据类型	0	速度最大值（总是为正）[u/s]
Acceleration	目标加速度	LREAL	遵照数据类型	0	加速度值（总是为正）[u/s2]
Deceleration	目标减速度	LREAL	遵照数据类型	0	减速度值（总是为正）[u/s2]
Jerk	目标加加速度	LREAL	遵照数据类型	0	加加速度值（总是为正） [u/s3]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成	BOOL	TRUE-FALSE	-	指令执行完成时为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE

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

功能说明

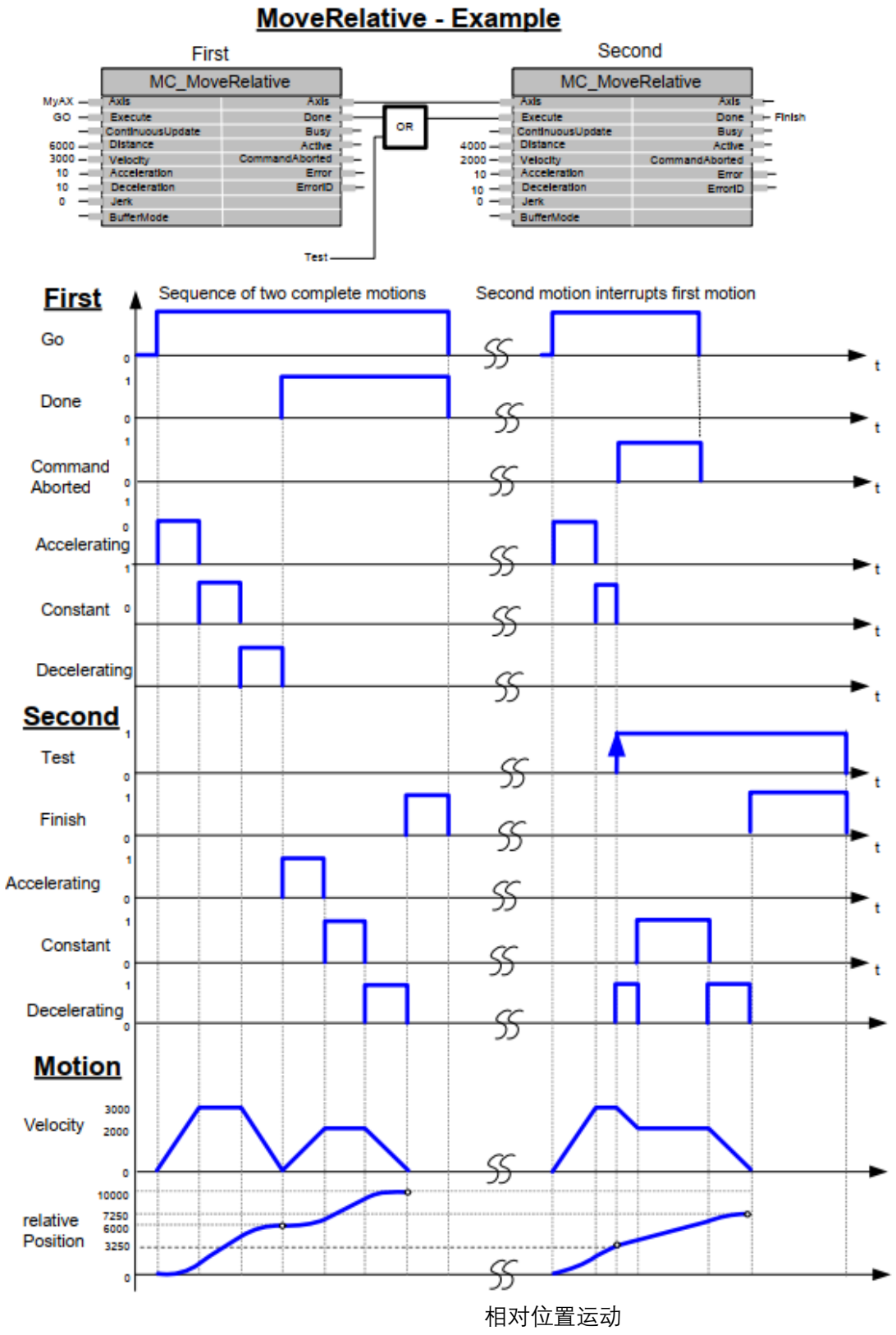
以当前位置为原点做相对运动，终点坐标即为起点到终点的距离。

例 MC_MoveRelative

下图是两个 MC_MoveRelative 功能块的实例“First”和“Second”连接的例程：

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

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



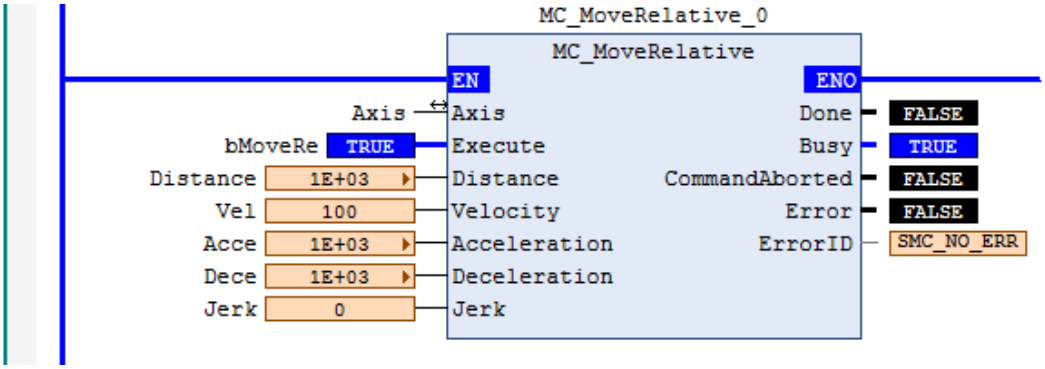
相对位置运动

程序示例

ST：当 bMoveRe 变为 TRUE 时，执行相对定位指令。

```
MC_MoveRelative_0(
    Axis:= Axis,
    Execute TRUE := bMoveRe TRUE,
    Distance 1E+03 := Distance 1E+03,
    Velocity 100 := Vel 100,
    Acceleration 1E+03 := Acce 1E+03,
    Deceleration 1E+03 := Dece 1E+03,
    Jerk 0 := Jerk 0,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

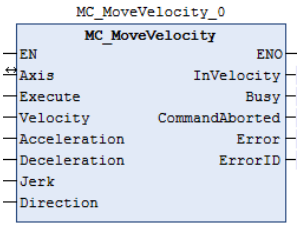
LD：当 bMoveRe 变为 TRUE 时，执行相对定位指令。



4.2.5 MC_MoveVelocity

本指令用于实现将控制轴按照设定参数，保持恒速运动。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_MoveVelocity	轴恒速运动	FB		MC_MoveVelocity_0(Axis:= , Execute:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Direction:= , InVelocity=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
Velocity	目标速度	LREAL	遵照数据类型	0	速度最大值（总是为正）[u/s]
Acceleration	目标加速度	LREAL	遵照数据类型	0	加速度值（总是为正）[u/s2]
Deceleration	目标减速度	LREAL	遵照数据类型	0	减速度值（总是为正）[u/s2]
Jerk	目标加加速度	LREAL	遵照数据类型	0	加加速度值（总是为正） [u/s3]
Direction	方向	MC_Direction	[-1,3]	shortest	3:fastest，最快； 2:current，当前； 1: Positive，正； 0:shortest，最短； -1:Negative，负；

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InVelocity	速度到达	BOOL	TRUE-FALSE	-	设定速度到达之后则为 TRUE

Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）Execute 的上升沿启动恒速运动，而后只能由其他的命令来中止恒速运动。

2）当恒速运动到达设定速度后，InVelocity 信号将变为 TRUE。当恒速运动被其它运动中断之后，InVelocity 信号将被重置为 FALSE。

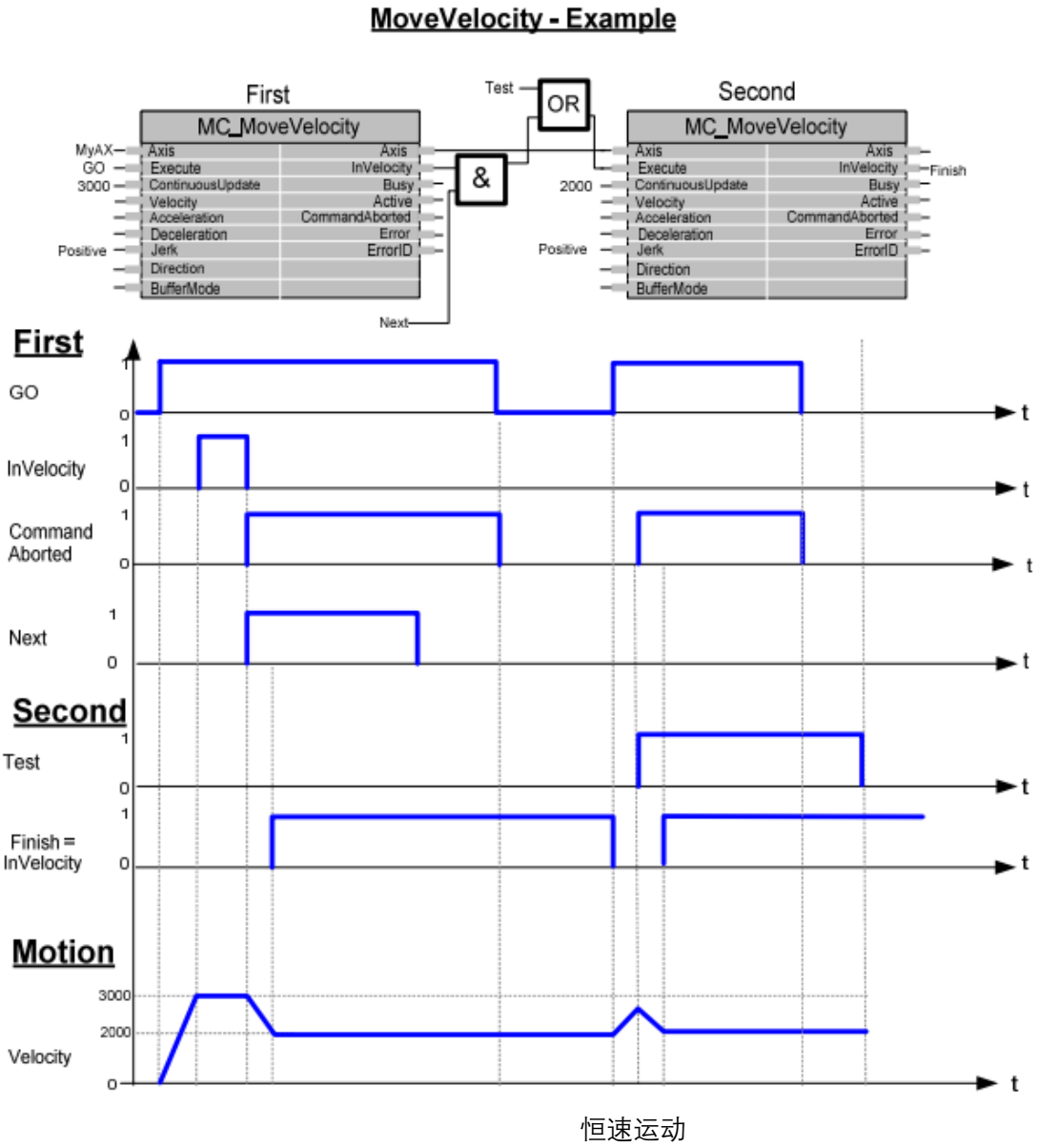
3）控制电机以指定的加速度加速到最大速度，然后以该最大速度一直运行，直到调用停止指令或者其他的中断指令中断该指令。

例 MC_MoveVelocity

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

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

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



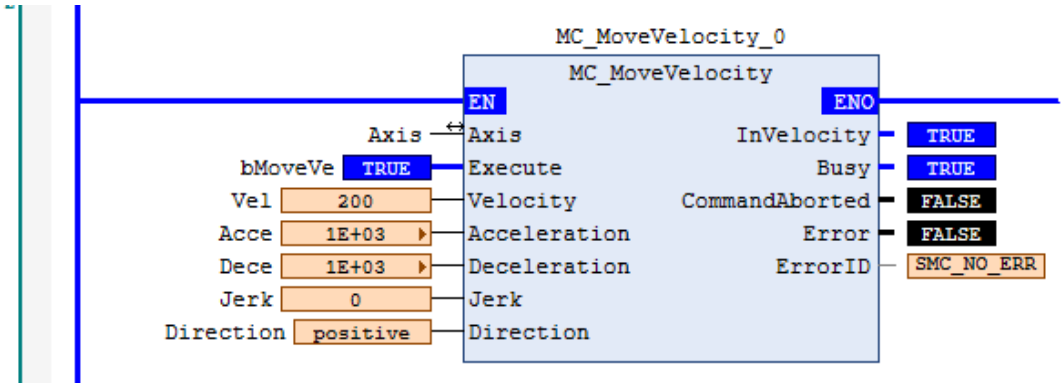
恒速运动

程序示例

ST：当 bMoveVe 变为 TRUE 时，执行恒速运动指令。

```
MC_MoveVelocity_0(
    Axis:= Axis,
    Execute TRUE := bMoveVe TRUE,
    Velocity 100 := Vel 100,
    Acceleration 1E+03 := Acce 1E+03,
    Deceleration 1E+03 := Dece 1E+03,
    Jerk 0 := Jerk 0,
    Direction positive := Direction positive,
    InVelocity TRUE => InVelocity TRUE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

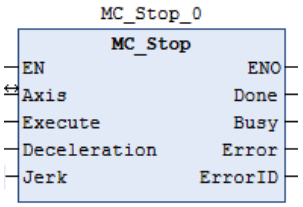
LD：当 bMoveVe 变为 TRUE 时，执行恒速运动指令。


MC_MoveVelocity_0
MC_MoveVelocity
EN
ENO
Axis
Axis
InVelocity
TRUE
bMoveVe
TRUE
Execute
Busy
TRUE
Vel
200
Velocity
CommandAborted
FALSE
Acce
1E+03
Acceleration
Error
FALSE
Dece
1E+03
Deceleration
ErrorID
SMC_NO_ERR
Jerk
0
Jerk
Direction
positive
Direction

4.2.6 MC_Stop

本指令用于中断轴正在进行的运动，对轴进行减速停止。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_Stop	轴停止运动	FB		MC_Stop_0(Axis:= , Execute:= , Deceleration:= , Jerk:= , Done=> , Busy=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
Deceleration	目标减速度	LREAL	遵照数据类型	0	减速度值（总是为正）[u/s2]
Jerk	目标加加速度	LREAL	遵照数据类型	0	加加速度值（总是为正） [u/s3]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	-	如果指令执行完成则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

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

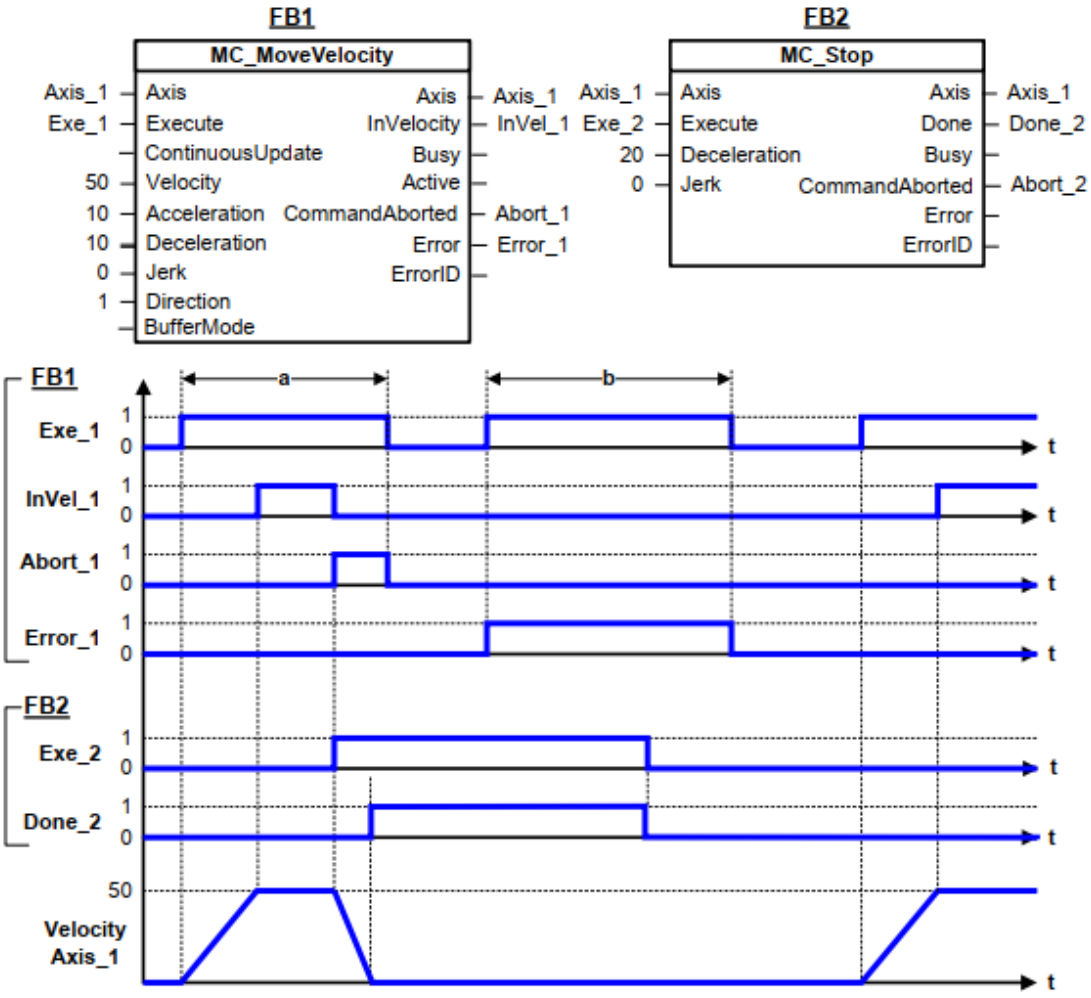
例 MC_Stop

下图为 FB2（MC_Stop）实例与 FB1（MC_MoveVelocity）实例相结合时的使用方法。

1) 由图中时序图可以看到，在 FB1 的运动过程中，启动了 FB2，轴的速度会呈现倾斜向下的趋势，直到为 0。

2) 只要 FB2 的 Execute 信号为 TRUE，对应的轴就不会执行任何运动命令。

3) 在 FB2 启动后，FB1 再启动 FB1 的 Error 信号会置为 TRUE，因为当前轴处于停止状态。



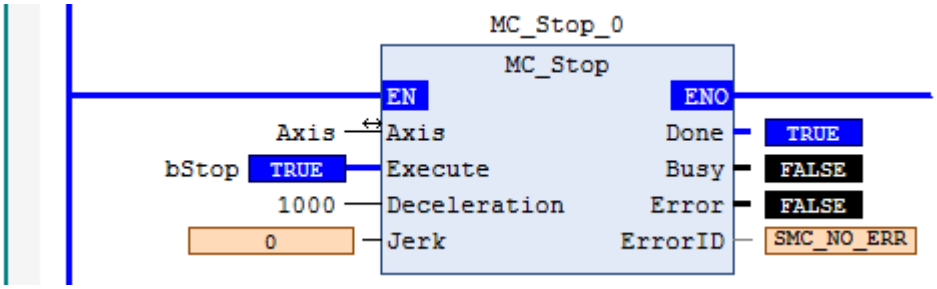
停止运动

程序示例

ST：当 bStop 变为 TRUE 时，对轴进行减速停止。

```
MC_Stop_0(
    Axis:= Axis,
    Execute TRUE := bStop TRUE,
    Deceleration 1E+03 := 1000,
    Jerk:= ,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

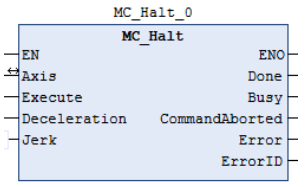
LD：当 bStop 变为 TRUE 时，对轴进行减速停止。



4.2.7 MC_Halt

本指令用于减速停止轴正在执行的运动，停止的运动可恢复执行未完成部分。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_Halt	轴暂停运动	FB		MC_Halt_0(Axis:= , Execute:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
Deceleration	目标减速度	LREAL	遵照数据类型	0	减速度值（总是为正）[u/s2]
Jerk	目标加加速度	LREAL	遵照数据类型	0	加加速度值（总是为正） [u/s3]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	-	如果指令执行完成则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）对于设定轴，在 Execute 的上升沿处开始暂停运动。

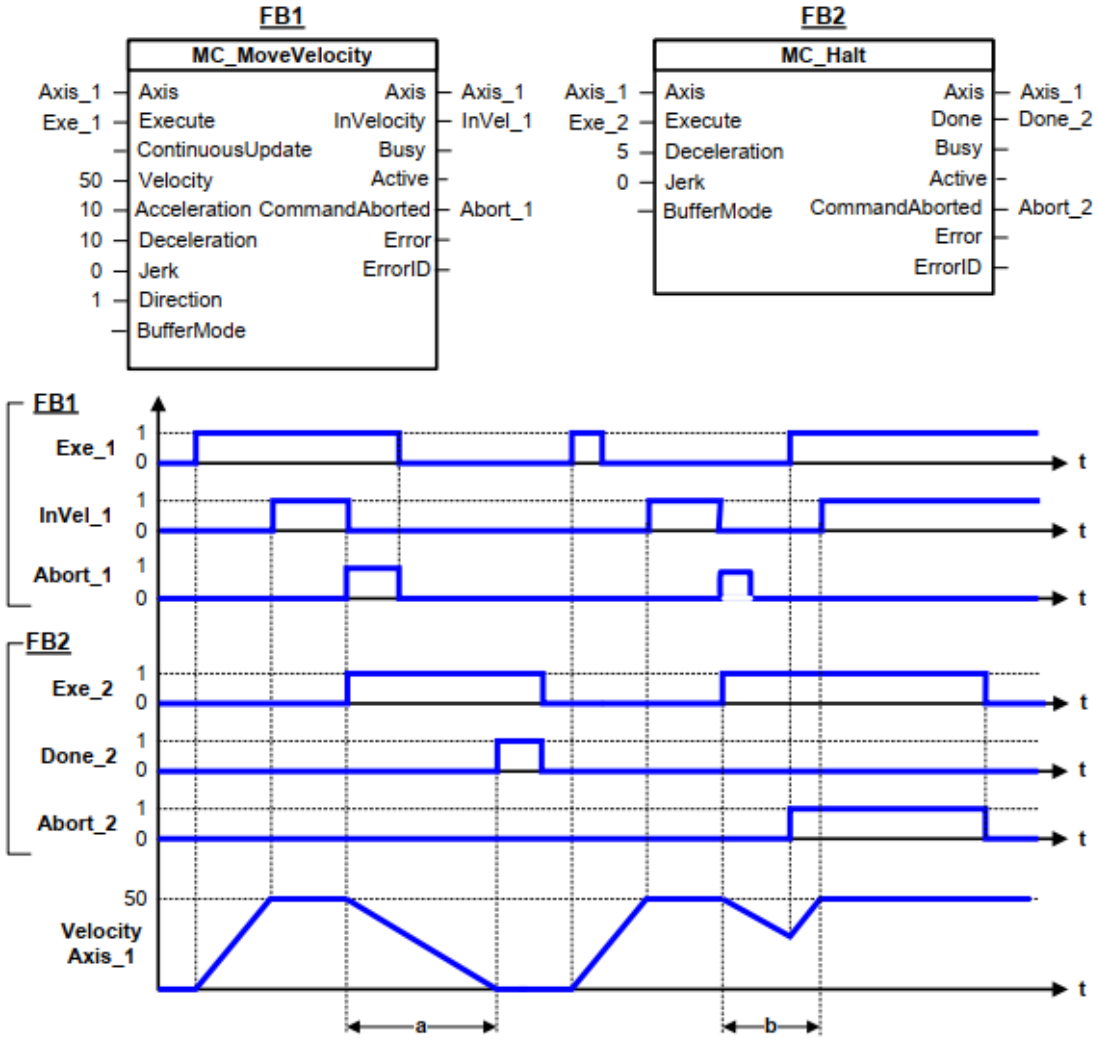
2）重新执行原来运动指令可以恢复原运动的执行。

例 MC_Halt

下图显示了 MC_MoveVelocity 指令正在执行运动的过程中，被 MC_Halt 指令中断的行为。

1) 与 MC_Stop 相比，MC_Halt 只是暂停运动，运动在后续还能够继续执行直到指令完成。

2) 在被 MC_Halt 指令暂停后，轴可以在速度没有为 0 时，重新调用运动指令，进行再次加速。



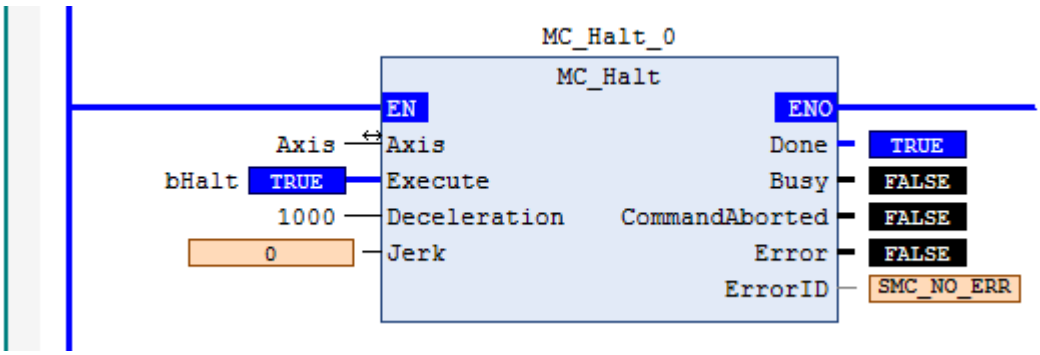
暂停运动

程序示例

ST：当 bHalt 变为 TRUE 时，对轴执行减速暂停指令。

```
MC_Halt_0(
    Axis:= Axis,
    Execute TRUE := bHalt TRUE,
    Deceleration 1E+03 := 1000,
    Jerk:= ,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

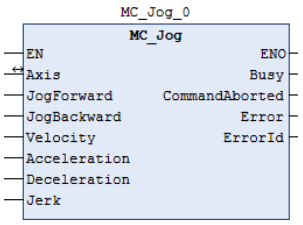
LD：当 bHalt 变为 TRUE 时，对轴执行减速暂停指令。


MC_Halt_0
MC_Halt
EN
ENO
Axis
Axis
Done
TRUE
bHalt
TRUE
Execute
Busy
FALSE
1000
Deceleration
CommandAborted
FALSE
0
Jerk
Error
FALSE
ErrorID
SMC_NO_ERR

4.2.8 MC_Jog

本指令用于手动控制轴朝指定方向运动。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_Jog	点动运动	FB		MC_Jog_0(Axis:= , JogForward:= , JogBackward:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Busy=> , CommandAborted=> , Error=> , ErrorId=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
JogForward	向前点动	BOOL	TRUE-FALSE	FALSE	如果 JogForward 为 TRUE，轴将按给定参数（Velocity、Acceleration、Deceleration）朝正向运动，如果 JogBackward 同时为 TRUE，轴不动
JogBackward	向后点动	BOOL	TRUE-FALSE	FALSE	如果 JogBackward 为 TRUE，轴将按给定参数（Velocity、Acceleration、Deceleration）朝负向运动，如 JogForward 同时为 TRUE，轴不动。
Velocity	目标速度	LREAL	遵照数据类型	0	速度最大值（总是为正）[u/s]

Acceleration	目标加速度	LREAL	遵照数据类型	0	加速度值（总是为正）[u/s2]
Deceleration	目标减速度	LREAL	遵照数据类型	0	减速度值（总是为正）[u/s2]
Jerk	目标加加速度	LREAL	遵照数据类型	0	加加速度值（总是为正） [u/s3]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）当“JogForward”或“JogBackward”为 TRUE 时，指定轴分别往正方向或负方向执行恒速运动。

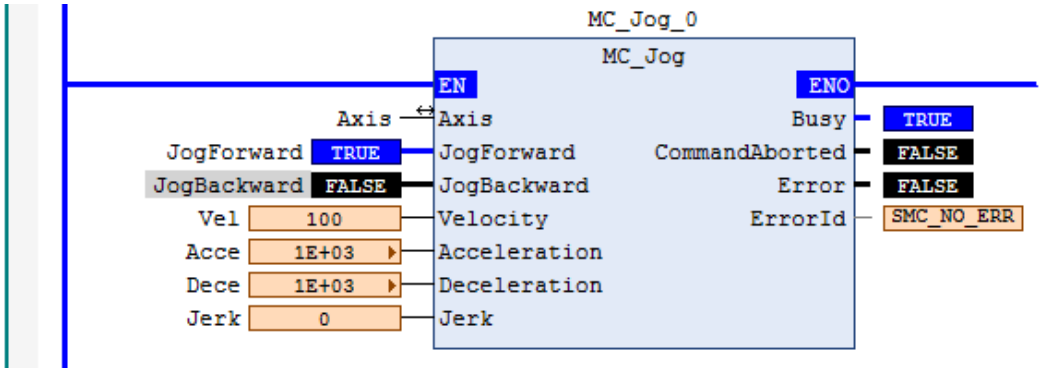
2）同时将 JogForward 和 JogBackward 置为 TRUE，将不会有运动发生。

程序示例

ST：当 JogForward 变为 TRUE 时，指定轴往正方向执行恒速运动。

```
MC_Jog_0(
    Axis:= Axis,
    JogForward TRUE := JogForward TRUE,
    JogBackward FALSE := JogBackward FALSE,
    Velocity 100 := Vel 100,
    Acceleration 1E+03 := Acce 1E+03,
    Deceleration 1E+03 := Dece 1E+03,
    Jerk 0 := Jerk 0,
    Busy TRUE => Busy FALSE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorId SMC_NO_ERR => ErrorId SMC_NO_ERR );
```

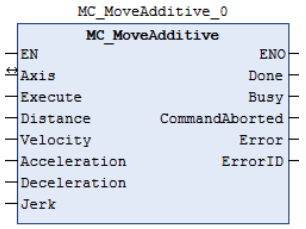
LD：当 JogForward 变为 TRUE 时，指定轴往正方向执行恒速运动。



4.2.9 MC_MoveAdditive

本指令用于控制终端执行机构在运动过程中做位置叠加运动。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_MoveAdditive	位置叠加	FB		MC_MoveAdditive_0(Axis:= , Execute:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
Distance	距离	LREAL	遵照数据类型	0	运动的相对距离[u]（负或正）
Velocity	目标速度	LREAL	遵照数据类型	0	速度最大值（总是为正）[u/s]
Acceleration	目标加速度	LREAL	遵照数据类型	0	加速度值（总是为正）[u/s2]
Deceleration	目标减速度	LREAL	遵照数据类型	0	减速度值（总是为正）[u/s2]
Jerk	目标加加速度	LREAL	遵照数据类型	0	加加速度值（总是为正） [u/s3]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	-	如果指令执行完成则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE

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

功能说明

1）此指令用于控制终端执行机构按给定的速度、加速度移动一段附加的距离。

2）终端执行机构的最终位置为前一个位移指令和此指令给定的距离总和。

3）打断其它指令的运动时，轴的速度为此功能块的速度。

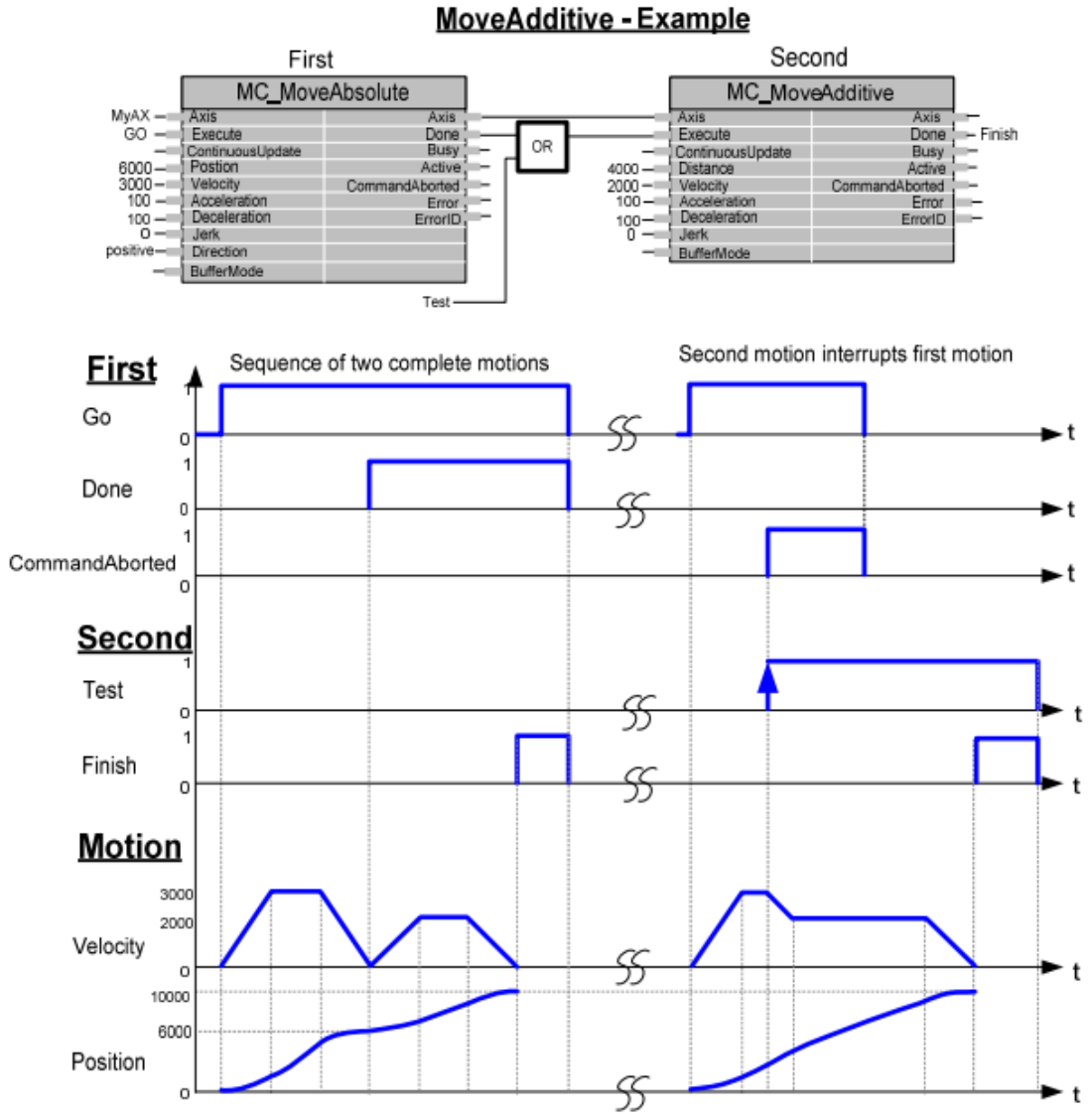
4）当前一个指令为速度指令时，此指令执行时会终止速度指令执行，并按给定的速度，加减速移动给定的距离后停止。

例 MC_MoveAdditive

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

2) 在图的前段，Second 实例在 First 实例之后被调用。当 First 实例运动到指定的位置 6000，则速度为 0 并且 Done 信号为 True，开始执行实例 Second，使轴移动到位置 10000。

3) 在图的后段，在实例 First 执行的过程中，启动 Second 实例，First 实例的运动此刻正处在 4000 的位置，此时 First 的运动被打断，开始减速，当到达位置 6000 时，以 Second 指定的速度，将轴移动到最终的位置 10000。



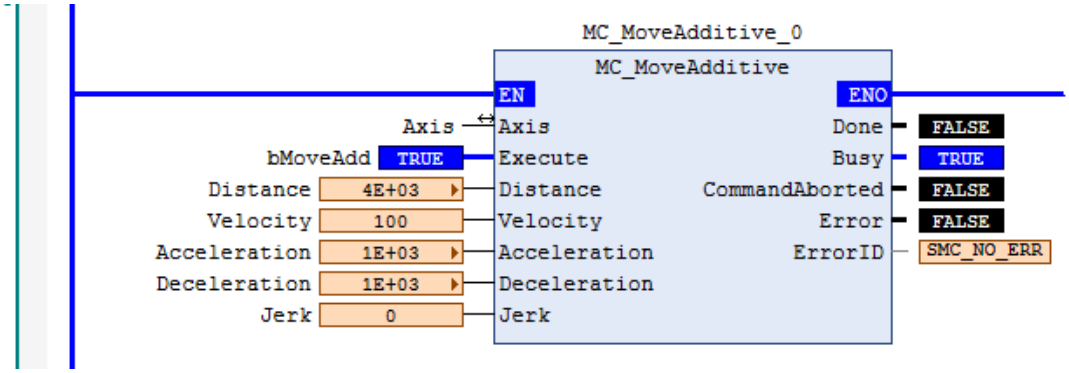
位置叠加运动

程序示例

ST：当 bMoveAdd 变为 TRUE 时，执行位置叠加运动指令。

```
MC_MoveAdditive_0(
    Axis:= Axis,
    Execute TRUE := bMoveAdd TRUE,
    Distance 4E+03 := Distance 4E+03,
    Velocity 500 := Velocity 500,
    Acceleration 1E+03 := Acceleration 1E+03,
    Deceleration 1E+03 := Deceleration 1E+03,
    Jerk 0 := Jerk 0,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy FALSE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

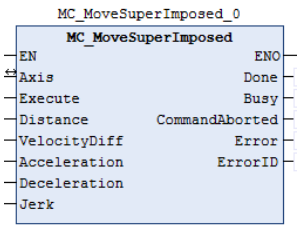
LD：当 bMoveAdd 变为 TRUE 时，执行位置叠加运动指令。



4.2.10 MC_MoveSuperImposed

本指令用于控制终端执行机构在运动过程中叠加运动。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_MoveSuperImposed	位置速度叠加	FB		MC_MoveSuperImposed_0(Axis:= , Execute:= , Distance:= , VelocityDiff:= , Acceleration:= , Deceleration:= , Jerk:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
Distance	距离	LREAL	遵照数据类型	0	运动的相对距离
VelocityDiff	叠加速度	LREAL	遵照数据类型	0	叠加的速度
Acceleration	目标加速度	LREAL	遵照数据类型	0	加速度值
Deceleration	目标减速度	LREAL	遵照数据类型	0	减速度值
Jerk	目标加加速度	LREAL	遵照数据类型	0	加加速度值

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	-	如果指令执行完成则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）此指令用于控制终端执行机构按给定的速度、加速度移动一段附加的距离。

2）终端执行机构的最终位置为前一个位移指令和此指令给定的距离总和。

3）第一段运动未完成时，速度为两段运动之和；第一段运动完成时，速度为此功能块的速度。

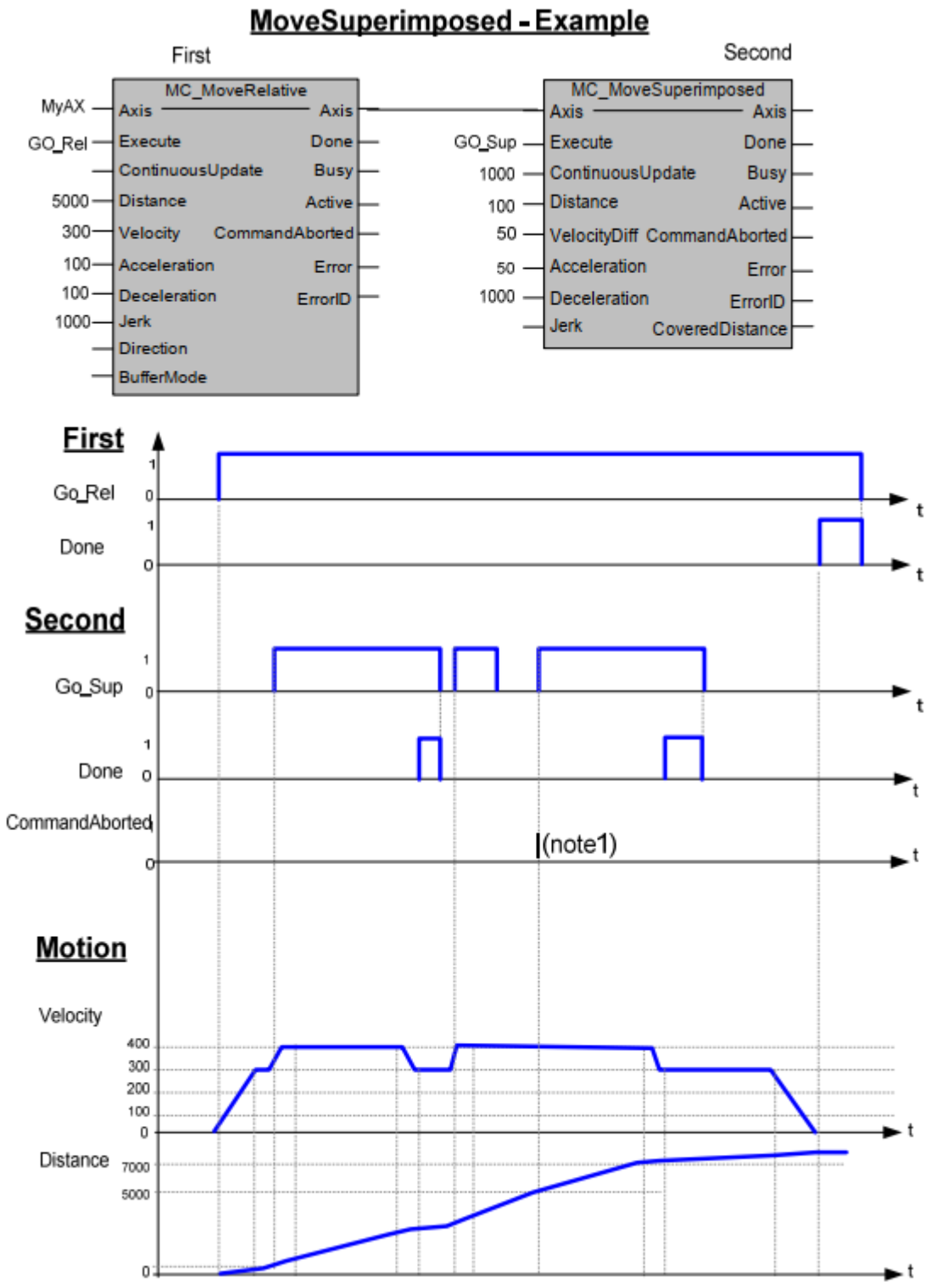
4）当前一个指令为速度指令时，此指令执行时会终止速度指令执行，并按给定的速度，加减速移动给定的距离后停止。

例 MC_MoveSuperImposed

1) 下图展示了 MC_MoveRelative 功能块的实例“First”和 MC_MoveSuperImposed 功能块的实例“Second”连接的例子。

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

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



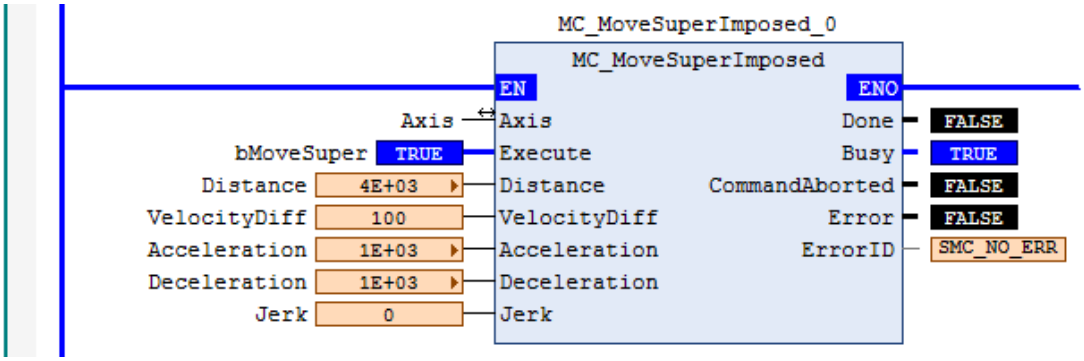
位置速度叠加运动

程序示例

ST：当 bMoveAdd 变为 TRUE 时，执行位置速度叠加运动指令。

```
MC_MoveSuperImposed_0(
    Axis:= Axis,
    Execute TRUE := bMoveSuper TRUE,
    Distance 4E+03 := Distance 4E+03,
    VelocityDiff 100 := VelocityDiff 100,
    Acceleration 1E+03 := Acceleration 1E+03,
    Deceleration 1E+03 := Deceleration 1E+03,
    Jerk 0 := Jerk 0,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy FALSE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR);
```

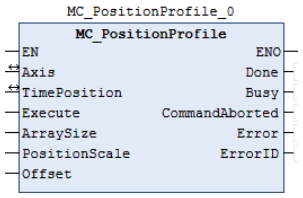
LD：当 bMoveAdd 变为 TRUE 时，执行位置速度叠加运动指令。



4.2.11 MC_PositionProfile

本指令使用户可以自己规划“时间-位置”数据表，控制器将按照规划的数据完成运动。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_PositionProfile	时间-位置规划	FB		MC_PositionProfile_0(Axis:= , TimePosition:= , Execute:= , ArraySize:= , PositionScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴
TimePosition	数据表	MC_TP_REF	-	-	用户规划的时间-位置数据表

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
ArraySize	数据点数	INT	遵照数据类型	0	数据表大小
PositionScale	比例	LREAL	遵照数据类型	1	整个数据表位置的比例系数
Offset	偏移	LREAL	遵照数据类型	0	位置偏置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	-	如果指令执行完成则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）用户可以自己规划“时间-位置”数据表，控制器将按照规划的数据完成运动。

2）PVT 运动是指通过定义各个时刻点轴达到的位置、速度、加速度等参数来规划一个运动。

3）控制器支持的 PVT 运动有三种，如下表所示。

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

MC_PositionProfile 运动所用到的数据结构定义如下：

1）“时间-位置”数据表的结构体：

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

2）“时间-位置”数据的结构体：

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

例 MC_PositionProfile

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

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

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

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

程序 PT 的数据

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

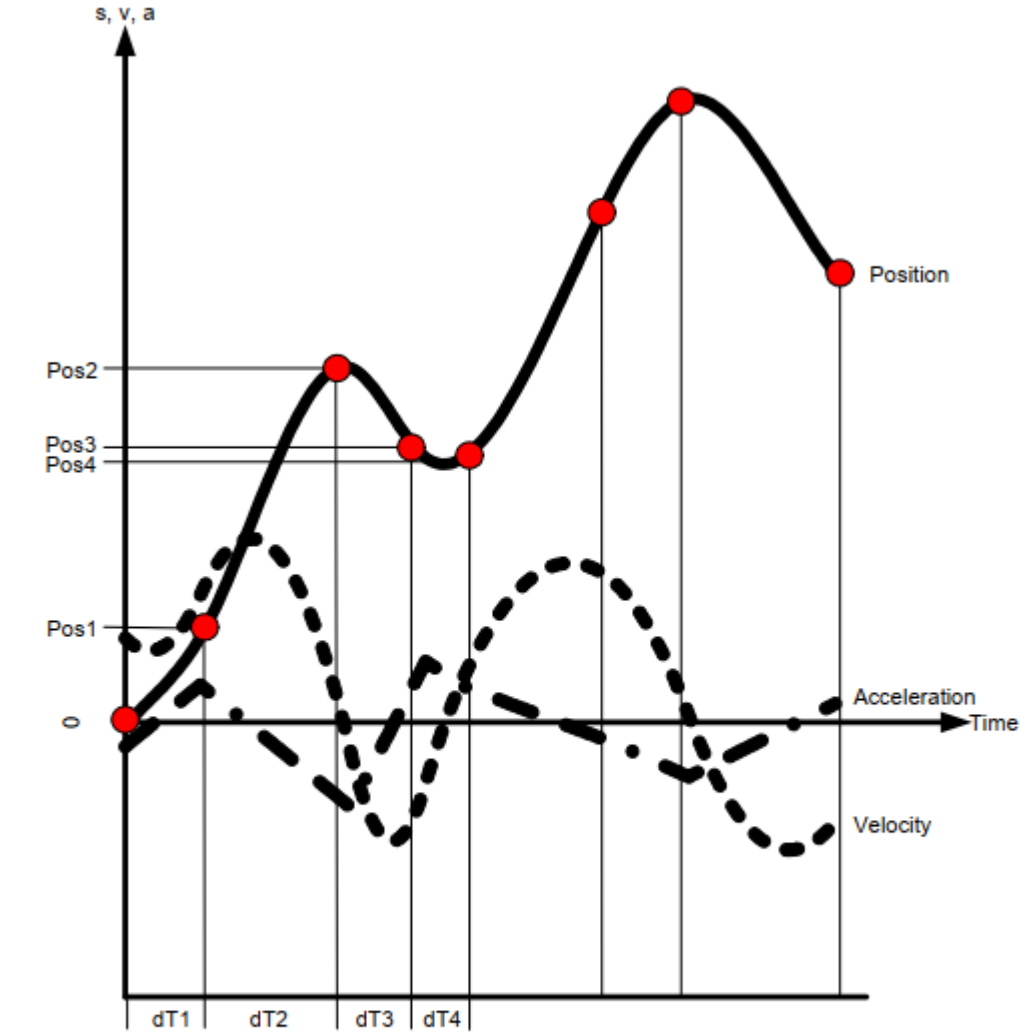
第一阶段：1 秒，运动 10000，

第二阶段：2 秒，运动 40000，

第三阶段：1 秒，运动 -10000，

第四阶段：0.5 秒，运动 -5000。

4）例程“PT”的运行结果如下图所示。



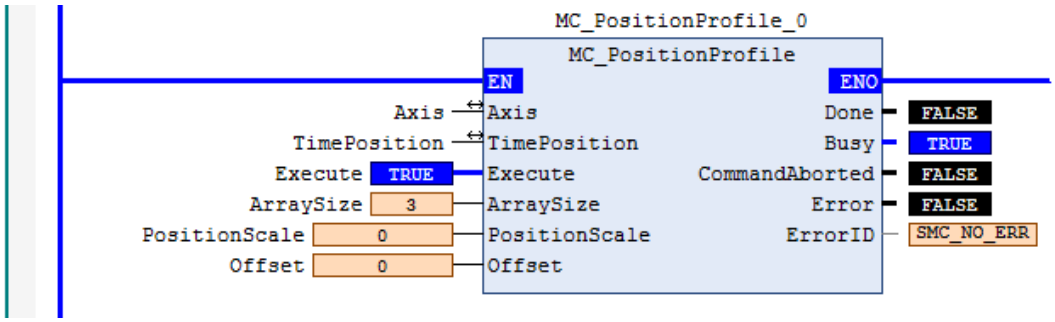
PT 运行结果

程序示例

ST：当 Execute 变为 TRUE 时，执行时间-位置规划运动指令。

```
MC_PositionProfile_0(
    Axis:= Axis,
    TimePosition:= TimePosition,
    Execute TRUE := Execute TRUE,
    ArraySize 3 := ArraySize 3,
    PositionScale 0 := PositionScale 0,
    Offset 0 := Offset 0,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

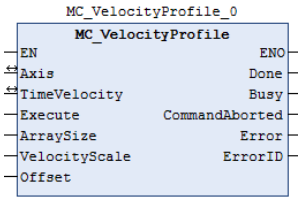
LD：当 Execute 变为 TRUE 时，执行时间-位置规划运动指令。


MC_PositionProfile_0
MC_PositionProfile
EN
ENO
Axis
Axis
TimePosition
TimePosition
Execute
TRUE
Execute
ArraySize
3
ArraySize
PositionScale
0
PositionScale
Offset
0
Offset
Done
FALSE
Busy
TRUE
CommandAborted
FALSE
Error
FALSE
ErrorID
SMC_NO_ERR

4.2.12 MC_VelocityProfile

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

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_VelocityProfile	时间-速度规划	FB		MC_VelocityProfile_0(Axis:= , TimeVelocity:= , Execute:= , ArraySize:= , VelocityScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴
TimeVelocity	数据表	MC_TV_REF	-	-	用户规划的时间-速度数据表

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
ArraySize	数据点数	INT	遵照数据类型	0	数据表大小
VelocityScale	比例	LREAL	遵照数据类型	1	整个数据表速度的比例系数
Offset	偏移	LREAL	遵照数据类型	0	位置偏置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	-	如果指令执行完成则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

MC_VelocityProfile 功能块为时间和速度的轮廓运动模型，按照用户在“时间-速度”表变量中设定的数据执行运动。

VT 运动所用到的数据表：

1）“时间-速度”数据表的结构体：

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

2）“时间-速度”数据的结构体：

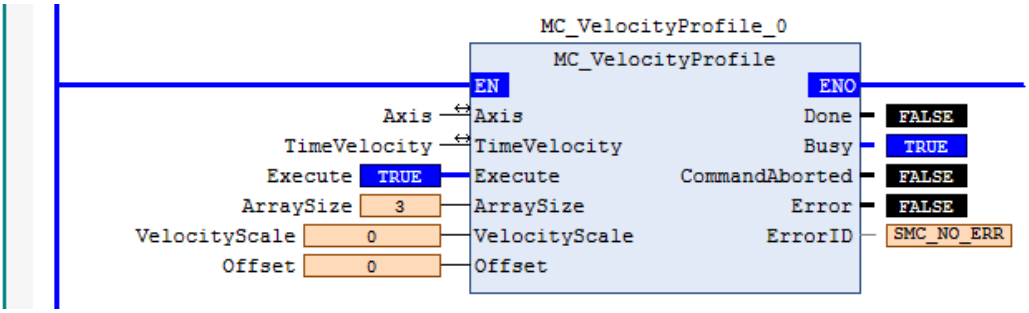
```
TYPE SMC_TV
STRUCT
   delta_time : TIME;        //速度-时间数据之时间
   velocity : REAL;          //速度-时间数据之位置（绝对值或相对值）
END_STRUCT
END_TYPE
```

程序示例

ST：当 Execute 变为 TRUE 时，执行时间-速度规划运动指令。

```
MC_VelocityProfile_0(
    Axis:= Axis,
    TimeVelocity:= TimeVelocity,
    Execute TRUE := Execute TRUE,
    ArraySize 3 := ArraySize 3,
    VelocityScale 0 := VelocityScale 0,
    Offset 0 := Offset 0,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR);
```

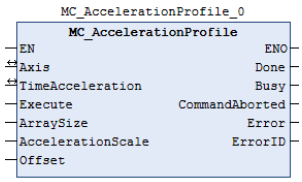
LD：当 Execute 变为 TRUE 时，执行时间-速度规划运动指令。



4.2.13 MC_AccelerationProfile

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

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_AccelerationProfile	时间-加速度规划	FB		MC_AccelerationProfile_0(Axis:= , TimeAcceleration:= , Execute:= , ArraySize:= , AccelerationScale:= , Offset:= , Done=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴
TimeAcceleration	数据表	MC_TA_REF	-	-	用户规划的时间-加速度数据表

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿触发指令运行
ArraySize	数据点数	INT	遵照数据类型	0	数据表大小
AccelerationScale	比例	LREAL	遵照数据类型	1	整个数据表加速度的比例系数
Offset	偏移	LREAL	遵照数据类型	0	位置偏置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	-	如果指令执行完成则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

MC_AccelerationProfile 功能块为时间和加速度的轮廓运动模型，按照用户在“时间-加速度”表变量中设定的数据执行运动。

AT 运动所用到的数据表：

1）“时间-加速度”数据表的结构体：

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

2) “时间-加速度”数据的结构体：

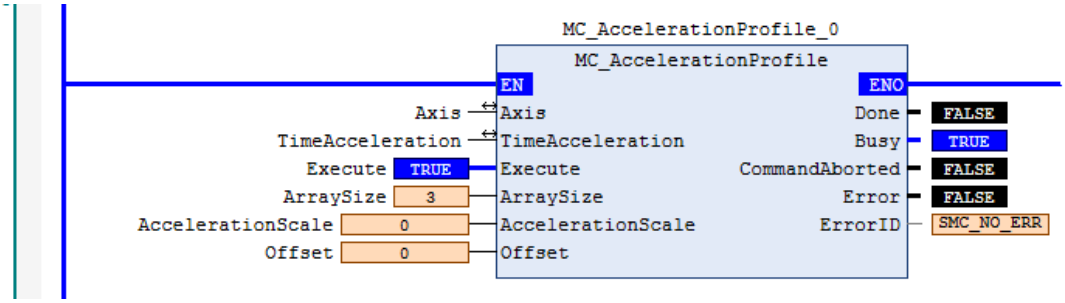
```
TYPE SMC_TA
STRUCT
   delta_time : TIME;            //加速度-时间数据之时间
   acceleration : REAL;          //加速度-时间数据之位置（绝对值或相对值）
END_STRUCT
END_TYPE
```

程序示例

ST：当 Execute 变为 TRUE 时，执行时间-加速度规划运动指令。

```
MC_AccelerationProfile_0(
    Axis:= Axis,
    TimeAcceleration:= TimeAcceleration,
    Execute TRUE := Execute TRUE,
    ArraySize 3 := ArraySize 3,
    AccelerationScale 0 := AccelerationScale 0,
    Offset 0 := Offset 0,
    Done FALSE => Done FALSE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

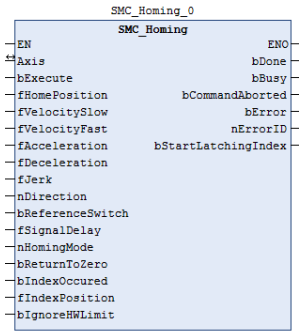
LD：当 Execute 变为 TRUE 时，执行时间-加速度规划运动指令。



4.2.14 SMC_Homing

轴回零指令，本指令是通过控制器控制回原，与 MC_Home 有区别。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SMC_Homing	轴回零	FB		SMC_Homing_0(Axis:= , bExecute:= , fHomePosition:= , fVelocitySlow:= , fVelocityFast:= , fAcceleration:= , fDeceleration:= , fJerk:= , nDirection:= , bReferenceSwitch:= , fSignalDelay:= , nHomingMode:= , bReturnToZero:= , bIndexOccured:= , fIndexPosition:= , bIgnoreHWLimit:= , bDone=> , bBusy=> , bCommandAborted=> , bError=> , nErrorID=> , bStartLatchingIndex=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExecute	执行	BOOL	TRUE-FALSE	FALSE	TRUE 功能块执行，FALSE 不执行功能块
fHomePosition	原点设定位置	LREAL	遵照数据类型	0	回零后原点设定位置，单位为用户标定后的单位
fVelocitySlow	慢速	LREAL	遵照数据类型	0	离开参考开关后慢速设定速度
fVelocityFast	快速	LREAL	遵照数据类型	0	离开参考开关置位时，快速设定速度
fAcceleration	加速度	LREAL	遵照数据类型	0	加速度设定值
fDeceleration	减速度	LREAL	遵照数据类型	0	减速度设定值
fJerk	加加速度	LREAL	遵照数据类型	0	Jerk [u/s3]
nDirection	回零方向	MC_DIRECTION	遵照数据类型	negative	回零开始方向，参考 MC_DIRECTION
bReferenceSwitch	参考开关	BOOL	TRUE-FALSE	FALSE	连接参考开关，TRUE:参考开关

					触发，FALSE:参考开关闭合
fSignalDelay	延迟	LREAL	遵照数据类型	0	参考开关的传输时间，用来补偿死区时间，单位为秒
nHomingMode	回零模式	SMC_HOMING_MODE	-	-	参考 SMC_HOMING_MODE
bReturnTozero	返回零位	BOOL	TRUE-FALSE	FALSE	TRUE:回零完成后轴运行到位置零（注意：如果 fHomePosition=10，则回零完成后轴位置变为 10，bReturnTozero 为 ture 则回零完成后轴反向走 10 个单位到 0 位）
bIndexOccured	脉冲信号	BOOL	TRUE-FALSE	FALSE	TRUE：标志脉冲记录，回零模式为 FAST_BSLOW_I_S_STOP，FAST_SLOW_I_S_STOP 时生效
fIndexPosition	记录位置	LREAL	遵照数据类型	0	标志脉冲时记录的位置
bIgnoreHWLimit	忽略硬限位	BOOL	TRUE-FALSE	FALSE	TRUE 设定硬件限位开关使能为 false，如果相同的物理开关用于硬件限位开关和参考开关，那么硬件控制将被设置为假

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	-	如果指令执行完成则为 TRUE
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE

CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
bStartLatchingIndex	启动锁存	BOOL	TRUE-FALSE	FALSE	由“bIndexOccured”和“fIndexPosition”共同作用产生

功能说明

1）SMC_HOMING 通过 bExecute 的上升沿启动之后，轴将会按照速度 fVelocityFast 并以 nDirection 定义的方向开始运动，直到 bReferenceSwitch = FALSE。然后轴将会缓慢停止并按照相反的方向以速度 fVelocitySlow 离开参考开关。bReferenceSwitch = TRUE 后回零完成，使能回零指令后 bReferenceSwitch 的状态为 ON->OFF->ON，在 OFF->ON 的上升沿回零完成，设置参考位置。

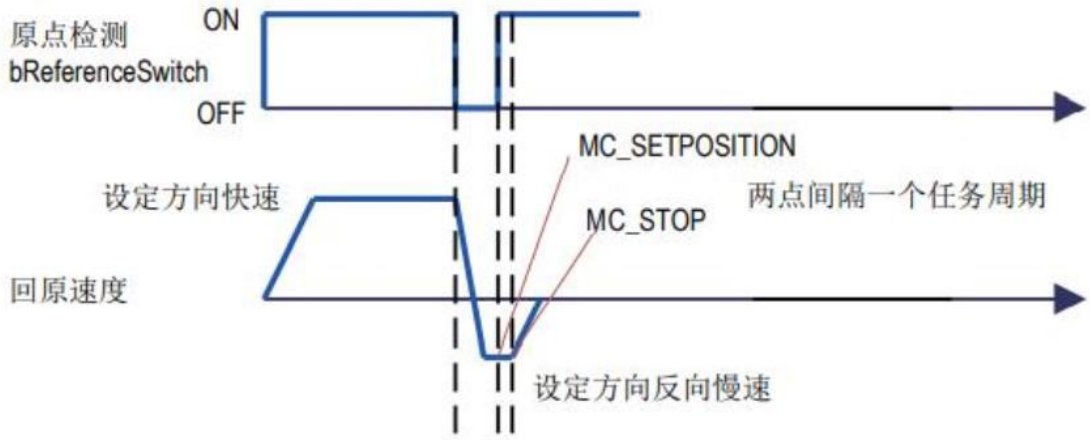
2）参考位置=fHomePostion+（(fSignalDelay*1000+1 个 DC 时钟周期)/1000*fVelocitySlow 实际就是补偿了设置的 bReferenceSwitch 采样延迟和一个通讯周期位移延迟。

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

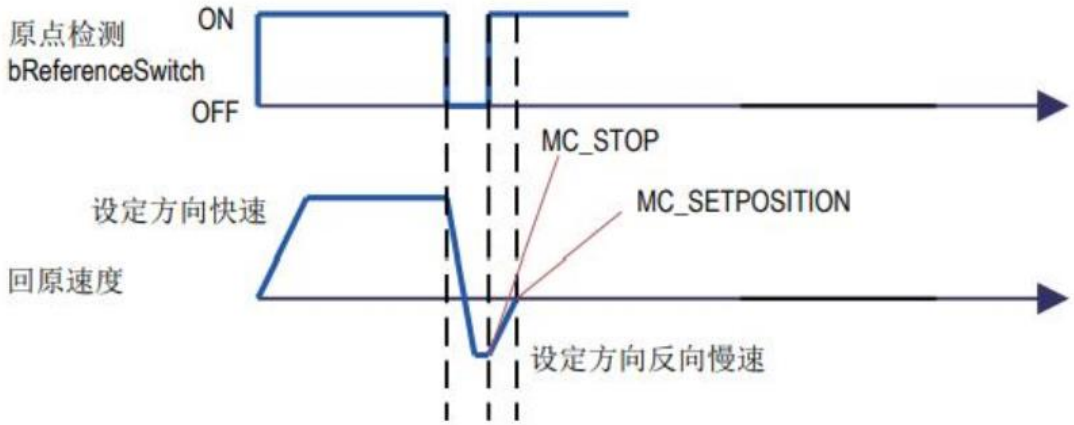
注意：Done 完成信号后，轴位置设定为：fHomePosition。设定的时机跟 nHomingMode 有关（详情参考 SMC_HOMING_MODE）。

下图为几种回零模式：

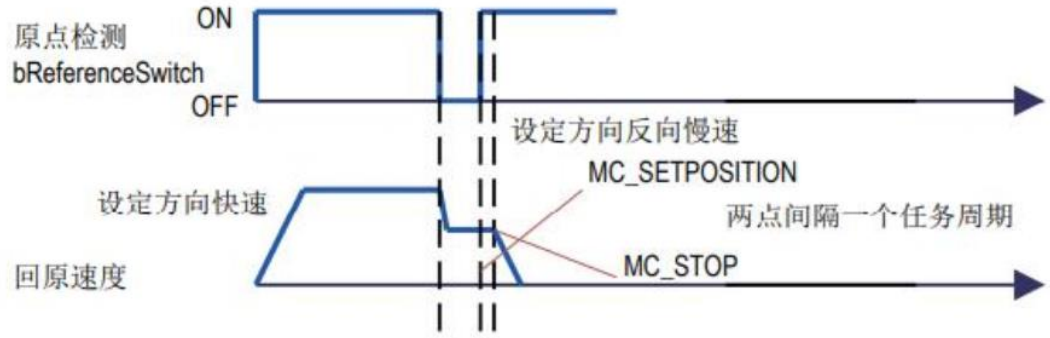
1) 回零模式“0”时



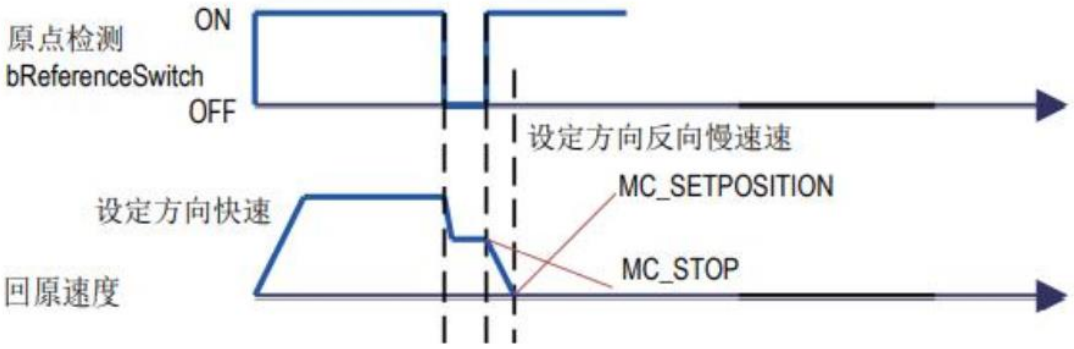
2) 回零模式“1”时



3） 回零模式“4”时



4） 回零模式“5”时

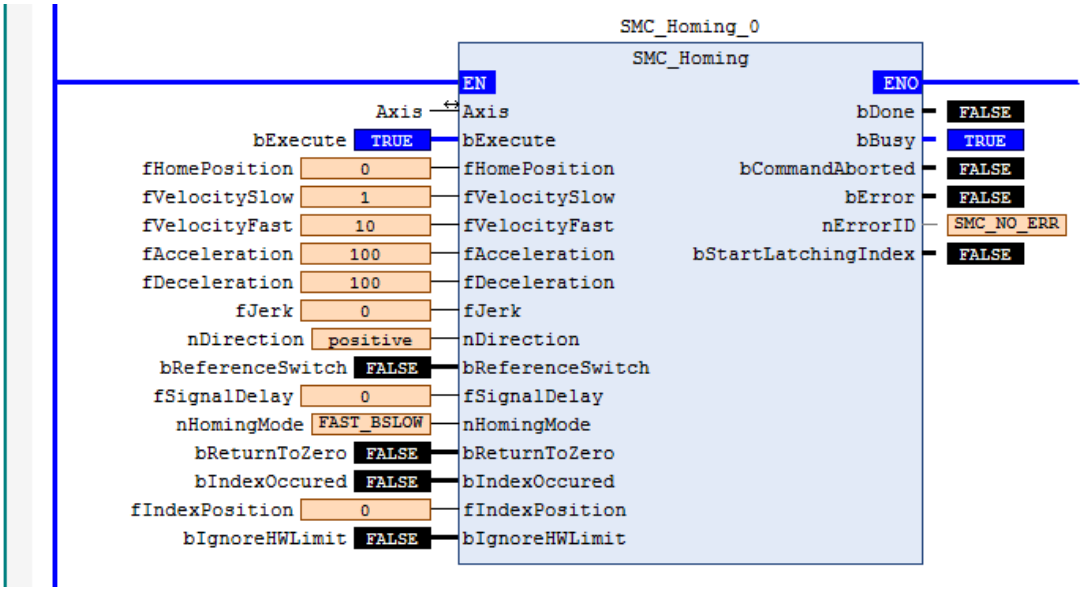


程序示例

ST：当 bExecute 变为 TRUE 时，执行轴回零指令。

```
SMC_Homing_0(
    Axis:= Axis,
    bExecute TRUE := bExecute TRUE,
    fHomePosition 0 := fHomePosition 0,
    fVelocitySlow 1 := fVelocitySlow 1,
    fVelocityFast 10 := fVelocityFast 10,
    fAcceleration 100 := fAcceleration 100,
    fDeceleration 100 := fDeceleration 100,
    fJerk 0 := fJerk 0,
    nDirection positive := nDirection positive,
    bReferenceSwitch FALSE := bReferenceSwitch FALSE,
    fSignalDelay 0 := fSignalDelay 0,
    nHomingMode FAST_BSLOW := nHomingMode FAST_BSLOW,
    bReturnToZero FALSE := bReturnToZero FALSE,
    bIndexOccured FALSE := bIndexOccured FALSE,
    fIndexPosition 0 := fIndexPosition 0,
    bIgnoreHWLimit FALSE := bIgnoreHWLimit FALSE,
    bDone FALSE => bDone FALSE,
    bBusy TRUE => bBusy TRUE,
    bCommandAborted FALSE => bCommandAborted FALSE,
    bError FALSE => bError FALSE,
    nErrorID SMC_NO_ERR => nErrorID SMC_NO_ERR,
    bStartLatchingIndex FALSE => bStartLatchingIndex FALSE);
```

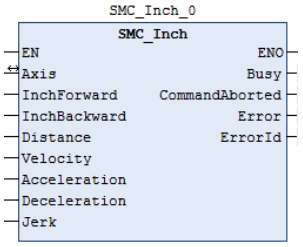
LD：当 bExecute 变为 TRUE 时，执行轴回零指令。



4.2.15 SMC_Inch

本指令用于手动控制轴按指定的距离单位，一段一段的朝指定方向运动。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SMC_Inch	寸动	FB		SMC_Inch_0(Axis:= , InchForward:= , InchBackward:= , Distance:= , Velocity:= , Acceleration:= , Deceleration:= , Jerk:= , Busy=> , CommandAborted=> , Error=> , ErrorId=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Axis	轴	AXIS_REF_SM3	-	-	指定轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
InchForward	向前寸动	BOOL	TRUE-FALSE	FALSE	如果 InchForward 为 TRUE，轴朝正向运动设定的距离，如果 InchForward 再次设 FALSE 到 TRUE，则轴再移动一段设定的距离。如果轴还没有运动到设定的距离，InchForward 就变为 FALSE，轴立即减速到 0，Busy 输出为 FALSE。如果 InchBackward 同时为真，轴不动。
InchBackward	向后寸动	BOOL	TRUE-FALSE	FALSE	如果 InchBackward 为 TRUE，轴朝

					反向运动设定的距离，如果 InchBackward 再次设定 FALSE 到 TRUE，则轴再移动一段设定的距离。如果轴还没有运动到设定的距离，InchBackward 就变为 FALSE，轴立即减速到 0，Busy 输出为 FALSE。如果 InchForward 同时为真，轴不动。
Distance	寸动距离	LREAL	遵照数据类型	0	定义一次输入轴要运动的距离
Velocity	目标速度	LREAL	遵照数据类型	0	速度最大值[u/s]
Acceleration	目标加速度	LREAL	遵照数据类型	0	加速度值[u/s2]
Deceleration	目标减速度	LREAL	遵照数据类型	0	减速度值[u/s2]
Jerk	加加速度	LREAL	遵照数据类型	0	加加速度值[u/s3]

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）一次运动的最大距离是固定的，由参数 Distance 定义。

2）如果需要再次运行 Distance 的距离，需要重置输入（InchForward OR InchBackward）。

3）如果距离 Distance 还没有到达，输入（InchForward OR InchBackward）就重置为 FALSE，运动立即减速停止。

4）输入 InchForward 和 InchBackward 都为 TRUE 时，轴不会运动。此时，当其中一个信号变成 FALSE，则轴将执行还为 TRUE 的信号所指定的运动。

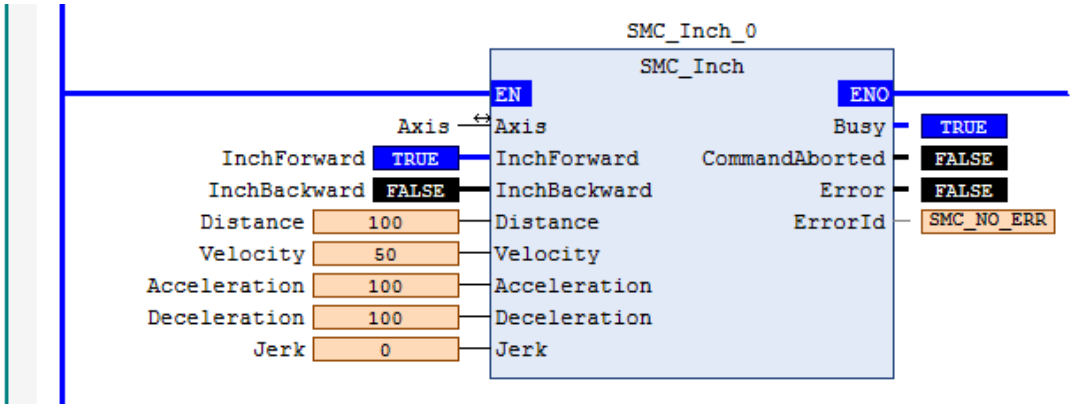
5）每次正向或反向运动的速度、加减速度等参数由 Velocity、Acceleration、Deceleration 和 Jerk 决定。

程序示例

ST：当 InchForward 变为 TRUE 时，指定轴往正方向运动指定距离。

```
SMC_Inch_0(
    Axis:= Axis,
    InchForward TRUE := InchForward TRUE,
    InchBackward FALSE := InchBackward FALSE,
    Distance 100 := Distance 100,
    Velocity 50 := Velocity 50,
    Acceleration 100 := Acceleration 100,
    Deceleration 100 := Deceleration 100,
    Jerk 0 := Jerk 0,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorId SMC_NO_ERR => ErrorId SMC_NO_ERR );
```

LD：当 InchForward 变为 TRUE 时，指定轴往正方向运动指定距离。



4.3 同步运动功能

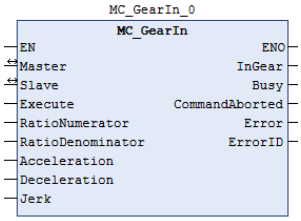
4.3.1 指令列表

指令类别	名称	FB/FC	功能
同步运动控制	MC_GearIn	FB	电子齿轮耦合
	MC_GearInPos	FB	电子齿轮平滑耦合
	MC_GearOut	FB	电子齿轮脱离
	MC_CamTableSelect	FB	凸轮挺杆关联
	MC_CamIn	FB	电子凸轮关联
	MC_CamOut	FB	电子凸轮脱离
	SMC_GetTappetValue	FB	读取挺杆状态
	MC_MoveAdditive	FB	附加运动
	MC_MoveSuperImposed	FB	叠加运动
	MC_PositionProfile	FB	位置规划

4.3.2 MC_GearIn

本指令用于设置主从轴齿轮比并启动电子齿轮。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_GearIn	电子齿轮	FB		MC_GearIn_0(Master:= , Slave:= , Execute:= , RatioNumerator:= , RatioDenominator:= , Acceleration:= , Deceleration:= , Jerk:= , InGear=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF_SM3	-	-	指定主轴
Slave	从轴	AXIS_REF_SM3	-	-	指定从轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	此输入的一个上升沿将启动功能块的处理
RatioNumerator	分子	DINT	遵照数据类型	1	齿轮比分子
RatioDenominator	分母	UDINT	遵照数据类型	1	齿轮比分母
Acceleration	目标加速度	LREAL	遵照数据类型	0	电子齿轮比加速度
Deceleration	目标减速度	LREAL	遵照数据类型	0	电子齿轮比减速度
Jerk	目标加加速度	LREAL	遵照数据类型	0	加加速度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InGear	齿轮啮合	BOOL	TRUE-FALSE	-	TRUE 表示齿轮比的处理已经完成
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE

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

功能说明

1）执行电子齿轮后，从轴将处于同步状态，要解除耦合，必须调用 MC_GearOut 指令。

2）指令属于速度同步类型，当启动指令后，从轴速度将上升至给定的齿轮比。在完成这个过程时，从轴耦合完成。在耦合期间造成的同步距离丢失不会得到补偿。

3）在 MC_GearIn 指令处于运行状态时，可以通过重新触发 MC_GearIn 指令，去改变主从轴的电子齿轮比，且中间不需要先调用 MC_GearOut 指令去暂停原来的电子齿轮运动。

4）到达目标速度，InGear 信号为 TRUE，此后，从轴移动量 = 主轴移动量 × RatioNumerator / RatioDenominator。

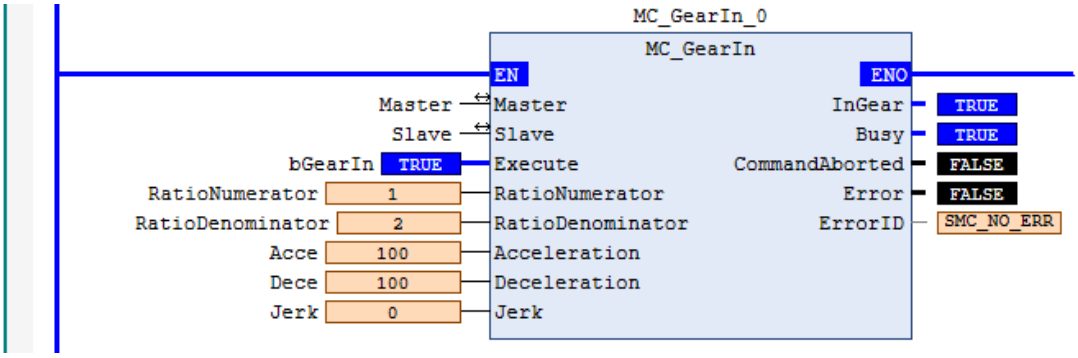
5）该指令一般适用于主轴速度稳定的情况，如果主轴速度实时变化的情况下，请注意慎重使用该指令。

程序示例

ST：当 bGearIn 变为 TRUE 时，设定主从轴齿轮比并启动电子齿轮。

```
MC_GearIn_0(
    Master:= Master,
    Slave:= Slave,
    Execute TRUE := bGearIn TRUE,
    RatioNumerator 1 := RatioNumerator 1,
    RatioDenominator 2 := RatioDenominator 2,
    Acceleration 100 := Acce 100,
    Deceleration 100 := Dece 100,
    Jerk 0 := Jerk 0,
    InGear TRUE => InGear TRUE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

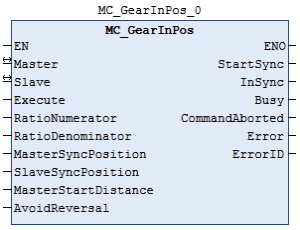
LD：当 Execute 变为 TRUE 时，设定主从轴齿轮比并启动电子齿轮。



4.3.3 MC_GearInPos

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

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_GearInPos	齿轮平滑耦合	FB		MC_GearInPos_0(Master:= , Slave:= , Execute:= , RatioNumerator:= , RatioDenominator:= , MasterSyncPosition:= , SlaveSyncPosition:= , MasterStartDistance:= , AvoidReversal:= , StartSync=> , InSync=> , Busy=> , CommandAborted=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF_SM3	-	-	指定主轴
Slave	从轴	AXIS_REF_SM3	-	-	指定从轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	此输入的一个上升沿将启动功能块的处理
RatioNumerator	分子	DINT	遵照数据类型	1	齿轮比分子
RatioDenominator	分母	UDINT	遵照数据类型	1	齿轮比分母
MasterSyncPosition	主轴同步位置	REAL	遵照数据类型	0	主从轴达到同步时主轴的位置
SlaveSyncPosition	从轴同步位置	REAL	遵照数据类型	0	主从轴达到同步时从轴的位置
MasterStartDistance	啮合段主轴运动距离	REAL	遵照数据类型	0	从轴启动到和主轴同步时主轴运动的距离
AvoidReversal	禁止反转	BOOL	TRUE-FALSE	FALSE	设置为 FALSE 表示从轴物理位置超前的情况下进行反转；设置为 TRUE 表示从轴物理上不能实现反转或者导致危险，只在模态

					轴下适用。如果反转不能被避免，那么轴将错误停止。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
StartSync	开始同步	BOOL	TRUE-FALSE	-	TRUE 表示电子齿轮开始进行处理
InSync	到达同步	BOOL	TRUE-FALSE	-	TRUE 表示电子齿轮命令完成
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）这个指令的工作过程可以简单概括为三步走：1）**等待起点**：功能块激活后，从轴不会立即动作。它会等待主轴到达一个计算好的起始点（MasterSyncPosition - MasterStartDistance）。 2）**加速追赶**：一旦主轴到达这个起始点，从轴就会开始加速，在主轴接下来的 MasterStartDistance 这段行程内，调整自己的位置和速度。 3）**完美同步**：当主轴恰好到达 MasterSyncPosition 时，从轴也刚好到达 SlaveSyncPosition，并且二者速度比完全符合 RatioNumerator / RatioDenominator，此时 InSync 输出变为 TRUE。

2）注意，如果主从轴工作在线性模式，需保证上述几个参数设置合理否则齿轮动作无法正确进行，故而建议使用该指令时主从轴为模数轴模式。比如主从轴线性工作模式都向正向运动，如果执行指令时主轴位置>MasterSyncPosition-MasterStartDistance，或者从轴位置> SlaveSyncPosition 则无法切入电子齿轮动作。

3）与 MC_GearIn 的区别是：MC_GearIn 是立即按照齿轮比同步，不关心位置偏差；而 MC_GearInPos 则在同步的同时，还要消除指定的位置偏差，实现位置上的锁合。

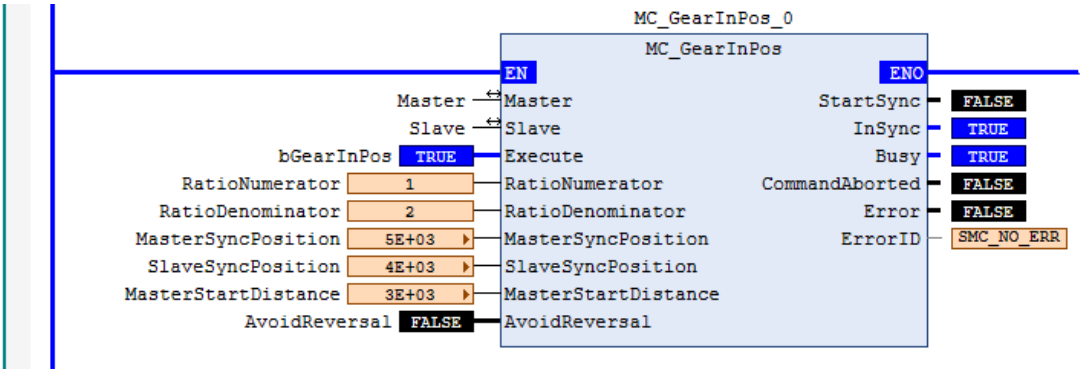
4）AvoidReversal 参数：当从轴是模态轴（如旋转轴）时，将此参数设为 TRUE 可以指示功能块尽量通过正向延长运动来避免轴反转，防止机械损伤。如果无法避免，功能块会报错。

程序示例

ST：当 bGearInPos 变为 TRUE 时，执行齿轮平滑耦合指令。

```
MC_GearInPos_0(
    Master:= Master,
    Slave:= Slave,
    Execute TRUE := bGearInPos TRUE,
    RatioNumerator 1 := RatioNumerator 1,
    RatioDenominator 2 := RatioDenominator 2,
    MasterSyncPosition 5E+03 := MasterSyncPosition 5E+03,
    SlaveSyncPosition 1E+03 := SlaveSyncPosition 1E+03,
    MasterStartDistance 3E+03 := MasterStartDistance 3E+03,
    AvoidReversal FALSE := AvoidReversal FALSE,
    StartSync FALSE => StartSync FALSE,
    InSync TRUE => InSync TRUE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

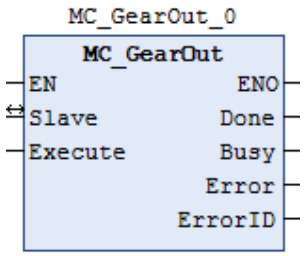
LD：当 bGearInPos 变为 TRUE 时，执行齿轮平滑耦合指令。



4.3.4 MC_GearOut

本指令用于将从轴与主轴的电子齿轮耦合断开。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_GearOut	齿轮解耦合	FB		MC_GearOut_0(Slave:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Slave	从轴	AXIS_REF_SM3	-	-	指定从轴

输入变量

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

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	-	TRUE 表示电子齿轮解耦合完成
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

1）电子齿轮解耦合动作完成后，此时从轴的速度为解耦合时的速度，所以需配合以 MC_Stop 指令停止从轴。

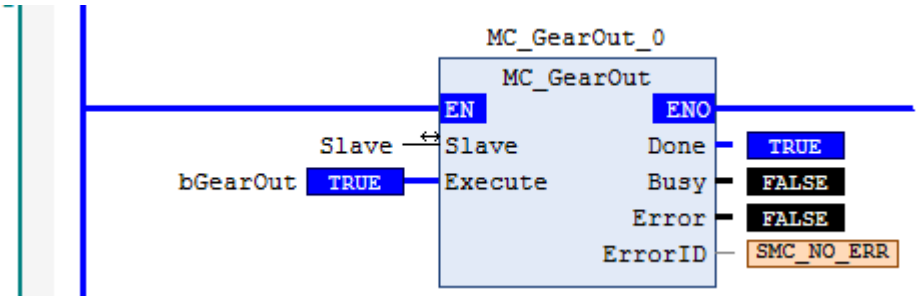
2）本指令对 MC_GearIn（齿轮动作开始）指令、MC_GearInPos（位置指定齿轮动作）指令的主轴动作没有影响。

程序示例

ST：当 bGearOut 变为 TRUE 时，执行齿轮解耦合指令。

```
MC_GearOut_0(
    Slave:= Slave,
    Execute TRUE := bGearOut TRUE,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

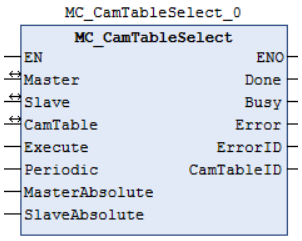
LD：当 bGearOut 变为 TRUE 时，执行齿轮解耦合指令。



4.3.5 MC_CamTableSelect

本指令用于选择需要执行的 cam 表，需要和 MC_CamIn 指令配合使用。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_CamTableSelect	凸轮表选择	FB		MC_CamTableSelect_0(Master:= , Slave:= , CamTable:= , Execute:= , Periodic:= , MasterAbsolute:= , SlaveAbsolute:= , Done=> , Busy=> , Error=> , ErrorID=> , CamTableID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF_SM3	-	-	指定主轴
Slave	从轴	AXIS_REF_SM3	-	-	指定从轴
CamTable	凸轮表	MC_CAM_REF	-	-	指定凸轮表

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	此输入的一个上升沿将启动功能块的处理
Periodic	重复模式	BOOL	TRUE-FALSE	FALSE	True：周期、重复地执行所指定的凸轮表 FALSE：仅执行一次凸轮表
MasterAbsolute	主轴绝对模式	BOOL	TRUE-FALSE	FALSE	TRUE 表示绝对坐标， FALSE 表示相对坐标
SlaveAbsolute	从轴绝对模式	BOOL	TRUE-FALSE	FALSE	TRUE 表示绝对坐标， FALSE 表示相对坐标

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	-	TRUE 表示电子齿轮解耦合完成
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE

Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
CamTableID	凸轮表 ID	MC_CAM_ID	-	-	cam 表的 ID，用于功能块 MC_CamIn

功能说明

1）这个指令用于指定电子凸轮运行所需凸轮表，所以在使用本指令之前先要将凸轮表编辑好（凸轮编辑器编辑或者在线编辑好）。

2）Execute 上升沿，执行指定凸轮表，亦可凸轮表更新后刷新指定的凸轮表。

3）Done 信号输出为 TRUE 时，则输出变量“CamTableID”生成并且生效。

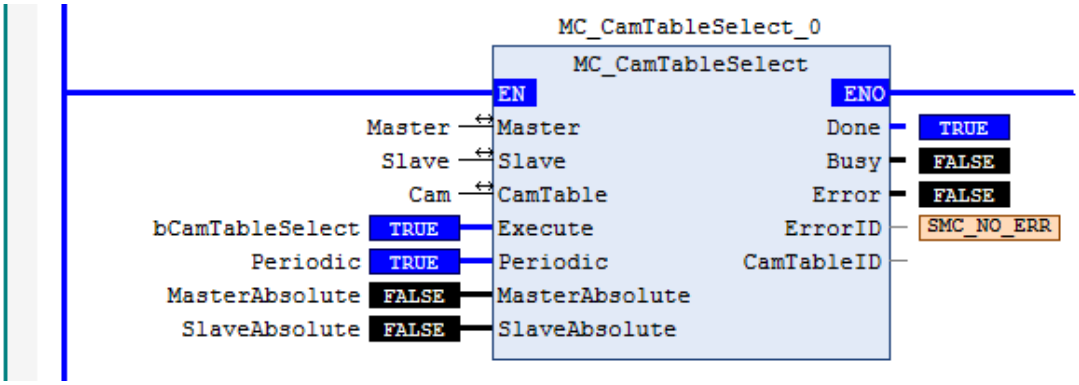
4）主轴和从轴不能指定为同一轴，否则会有报错输出。

程序示例

ST：当 bCamTableSelect 变为 TRUE 时，执行指定凸轮表指令。

```
MC_CamTableSelect_0(
    Master:= Master,
    Slave:= Slave,
    CamTable:= Cam,
    Execute TRUE := bCamTableSelect TRUE,
    Periodic TRUE := Periodic TRUE,
    MasterAbsolute FALSE := MasterAbsolute FALSE,
    SlaveAbsolute FALSE := SlaveAbsolute FALSE,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR,
    CamTableID=> CamTableID);
```

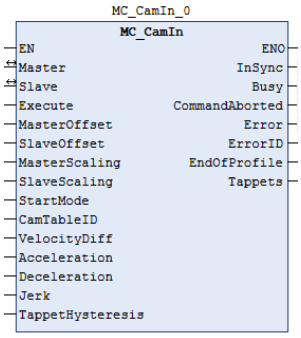
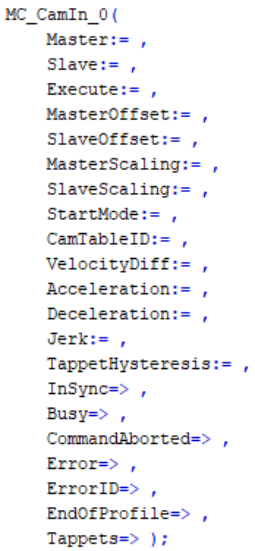
LD：当 bCamTableSelect 变为 TRUE 时，执行指定凸轮表指令。



4.3.6 MC_CamIn

使用指定的凸轮表开始执行电子凸轮动作。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_CamIn	电子凸轮耦合	FB			SM3_Basic


```
MC_CamIn_0(
    Master:= ,
    Slave:= ,
    Execute:= ,
    MasterOffset:= ,
    SlaveOffset:= ,
    MasterScaling:= ,
    SlaveScaling:= ,
    StartMode:= ,
    CamTableID:= ,
    VelocityDiff:= ,
    Acceleration:= ,
    Deceleration:= ,
    Jerk:= ,
    TappetHysteresis:= ,
    InSync=> ,
    Busy=> ,
    CommandAborted=> ,
    Error=> ,
    ErrorID=> ,
    EndOfProfile=> ,
    Tappets=> );
```

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Master	主轴	AXIS_REF_SM3	-	-	指定主轴
Slave	从轴	AXIS_REF_SM3	-	-	指定从轴

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	此输入的一个上升沿将启动功能块的处理
MasterOffset	主轴偏移	LREAL	遵照数据类型	0	主轴表格的偏移
SlaveOffset	从轴偏移	LREAL	遵照数据类型	0	从轴列表的偏移
MasterScaling	主轴比例	LREAL	遵照数据类型	1	主轴缩放比例
SlaveScaling	从轴比例	LREAL	遵照数据类型	1	从轴缩放比例
StartMode	从轴啮合方式	MC_StartMode	0-4	0	0：absolute 绝对位置 1：relative 相对位置 2：ramp_in 斜坡切入 3：ramp_in_pos 正向斜坡切入 4：ramp_in_neg 反向斜坡切入

CamTableID	凸轮表	MC_CAM_ID	-	-	定义 cam 表格的使用，是 MC_CamID 的输出
VelocityDiff	速度	LREAL	遵照数据类型	0	与 ramp_in 不同的最大速度
Acceleration	加速度	LREAL	遵照数据类型	0	对 ramp_in 的最大加速度
Deceleration	减速度	LREAL	遵照数据类型	0	对 ramp_in 的最大减速度
Jerk	目标加加速度	LREAL	遵照数据类型	0	加加速度
TappetHysteresis	挺杆阻尼	LREAL	遵照数据类型	0	挺杆滞后的尺寸大小

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
InSync	同步中	BOOL	TRUE-FALSE	-	TRUE 表示从轴根据 cam 表与主轴同步
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
CommandAborted	命令中断	BOOL	TRUE-FALSE	-	如果该命令已被其他命令终止则为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error
EndOfProfile	曲线完成	BOOL	TRUE-FALSE	-	在 cam 编译周期结束输出的脉冲信号
Tappets	挺杆表	SMC_TappetData	-	-	用于 SMC_GetTappetValue 处理的挺杆信号

功能说明

1）本指令可使主轴和从轴按照凸轮表进行同步凸轮动作。

2）由本指令指定的凸轮表有两种制作方式：1）使用凸轮编辑器编制；2）通过编程自建凸轮表数据结构。

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

4）如果要断开主从轴的凸轮耦合关系需使用 MC_CamOut 指令。

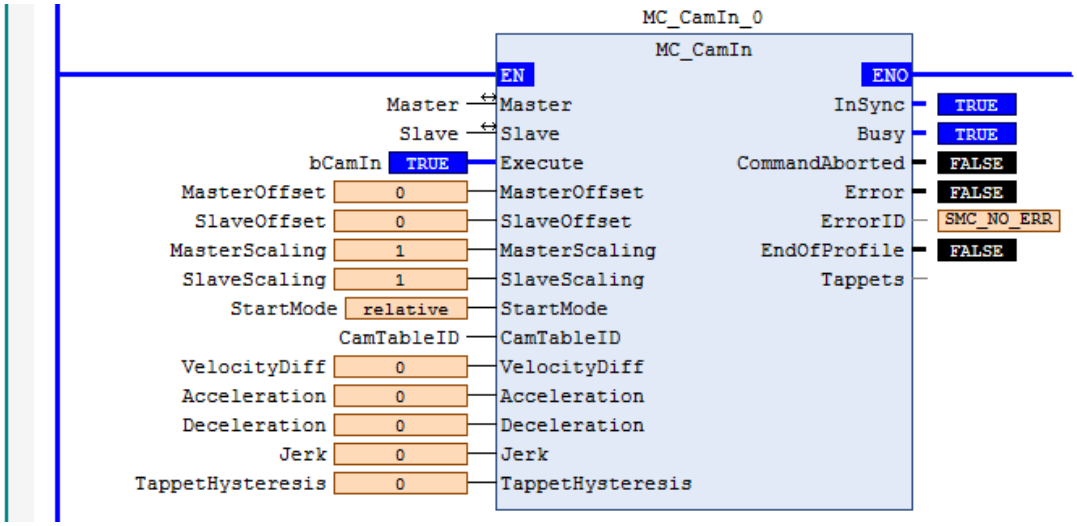
5)该指令执行时，该指令的从轴再执行其它运动指令时，从轴和主轴之间的凸轮关系会解除，并且 MC_CamIn 的 CommandAborted 变量输出为 TRUE。

程序示例

ST：当 bCamIn 变为 TRUE 时，执行电子凸轮耦合指令。

```
MC_CamIn_0(
    Master:= Master,
    Slave:= Slave,
    Execute TRUE := bCamIn TRUE,
    MasterOffset 0 := MasterOffset 0,
    SlaveOffset 0 := SlaveOffset 0,
    MasterScaling 1 := MasterScaling 1,
    SlaveScaling 1 := SlaveScaling 1,
    StartMode relative := StartMode relative,
    CamTableID:= CamTableID,
    VelocityDiff 0 := VelocityDiff 0,
    Acceleration 0 := Acceleration 0,
    Deceleration 0 := Deceleration 0,
    Jerk 0 := Jerk 0,
    TappetHysteresis 0 := TappetHysteresis 0,
    InSync TRUE => InSync TRUE,
    Busy TRUE => Busy TRUE,
    CommandAborted FALSE => CommandAborted FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR,
    EndOfProfile FALSE => EndOfProfile FALSE,
    Tappets=> Tappets);
```

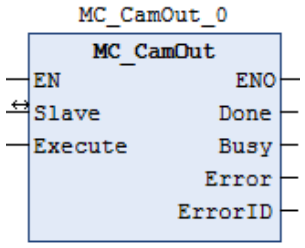
LD：当 bCamIn 变为 TRUE 时，执行电子凸轮耦合指令。



4.3.7 MC_CamOut

解除指定的从轴与其对应主轴的凸轮耦合关系。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
MC_CamOut	电子凸轮解耦合	FB		MC_CamOut_0(Slave:= , Execute:= , Done=> , Busy=> , Error=> , ErrorID=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Slave	从轴	AXIS_REF_SM3	-	-	指定从轴

输入变量

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

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	执行完成	BOOL	TRUE-FALSE	-	TRUE 表示电子凸轮解耦合完成
Busy	执行中	BOOL	TRUE-FALSE	-	当功能块执行还没结束时为 TRUE
Error	错误	BOOL	TRUE-FALSE	-	在功能块内部发生错误的信号
ErrorID	错误代码	SMC_ERROR	-	-	错误指示，见 SMC_Error

功能说明

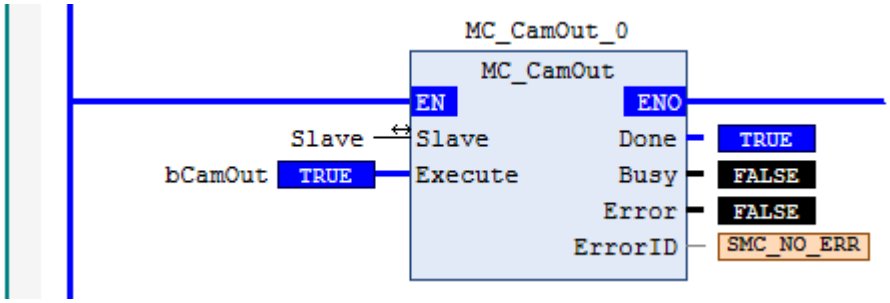
本指令用于解除指定的从轴与其对应主轴的凸轮耦合关系。

程序示例

ST：当 bCamOut 变为 TRUE 时，执行电子凸轮解耦合指令。

```
MC_CamOut_0(
    Slave:= Slave,
    Execute TRUE := bCamOut TRUE,
    Done TRUE => Done TRUE,
    Busy FALSE => Busy FALSE,
    Error FALSE => Error FALSE,
    ErrorID SMC_NO_ERR => ErrorID SMC_NO_ERR );
```

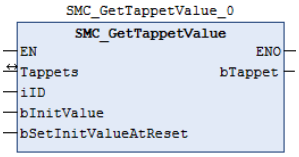
LD：当 bCamOut 变为 TRUE 时，执行电子凸轮解耦合指令。



4.3.8 SMC_GetTappetValue

读取当前的挺杆状态，需要和 MC_CamIn 指令配合使用。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
SMC_GetTappetValue	读取挺杆状态	FB		SMC_GetTappetValue_0(Tappets:= , iID:= , bInitValue:= , bSetInitValueAtReset:= , bTappet=>);	SM3_Basic

相关变量

输入输出变量

输入输出变量	名称	数据类型	有效范围	初始值	描述
Tappets	挺杆	SMC_TappetData	-	-	连接到 MC_CamIn 功能块的 Tappets 输出引脚

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
iID	挺杆组号	INT	遵照数据类型	0	要读取的挺杆组 ID
bInitValue	初始值	BOOL	TRUE-FALSE	FALSE	第一次调用时分配的挺杆初始值
bSetInitValueAtReset	挺杆复位	BOOL	TRUE-FALSE	FALSE	如果为 TRUE，在 CamIn 功能块重启的时候挺杆的输出值将会被设置为初始值。 如果为 FALSE，挺杆的值在 CamIn 功能块重启的时候将会被保持。

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
bTappet	挺杆值	BOOL	TRUE-FALSE	FALSE	挺杆的值

功能说明

1）在机械凸轮结构中，凸轮的旋转运动靠挺杆转换为垂直方向的运动。电子凸轮的挺杆，与机械挺杆类似，相当于电子化的机械挺杆。电子凸轮的挺杆就是凸轮主轴在规定的位置， 给出个高电平或者低电平，作为开关去使用。

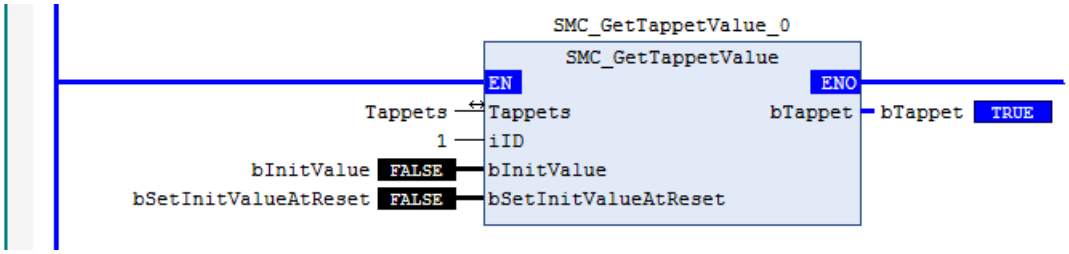
2）该功能块需要和 MC_CamIn 指令配合使用，用来获取凸轮表管理的挺杆的数值。

程序示例

ST:

```
SMC_GetTappetValue_0(
    Tappets:= Tappets,
    iID 1 := 1,
    bInitValue FALSE := bInitValue FALSE,
    bSetInitValueAtReset FALSE := bSetInitValueAtReset FALSE,
    bTappet TRUE => bTappet TRUE );
```

LD:



5 通信指令

5.1 自由 TCP 通信指令

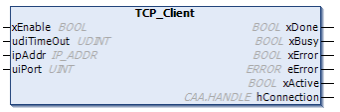
5.1.1 指令列表

指令类别	名称	FB/FC	功能
自由 TCP 通信指令	TCP_Client	FB	创建 TCP 客户端通信服务
	TCP_Write	FB	TCP 通信数据发送
	TCP_Read	FB	TCP 通信数据接收
	TCP_Connection	FB	创建 TCP 连接，并连接到服务器
	TCP_Server	FB	创建 TCP 服务器端通信服务

5.1.2 TCP_Client

创建 TCP 客户端通信服务。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
TCP_Client	创建 TCP 客户端	FB		TCP_Client_0(xEnable:= , xDone=> , xBusy=> , xError=> , udiTimeOut:= , ipAddr:= , uiPort:= , eError=> , xActive=> , hConnection=>);	CAA Net Base Services

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则启动功能块运行
udiTimeOut	超时时间	UDINT	0~2^32-1	0	请求建立连接的超时时间，单位：μs
ipAddr	服务器 IP 地址	IP_ADDR	-	-	客户端所连接的服务器 IP 地址，具体参考 CAA Net Base Services 库的结构体 IP_ADDR
uiPort	端口号	UINT	0~65535	0	客户端所连接的服务器端口号

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	FALSE-TRUE	-	客户端通信完成
xBusy	功能块运行中	BOOL	FALSE-TRUE	-	功能块运行中标志
xError	错误标志	BOOL	FALSE-TRUE	-	错误状态
eError	错误	ERROR	--	-	具体错误请参考 CAA Net Base Services 库的枚举体 ERROR
xActive	连接成功标志	BOOL	FALSE-TRUE	-	客户端与服务器连接成功标志
hConnection	连接句柄	CAA.HANDLE	--	-	客户端与服务器的连接句柄

功能说明

1）创建 TCP 客户端通信服务。

2）“ipAddr”的数据类型为结构体 IP_ADDR，结构体成员的含义如下表所示。

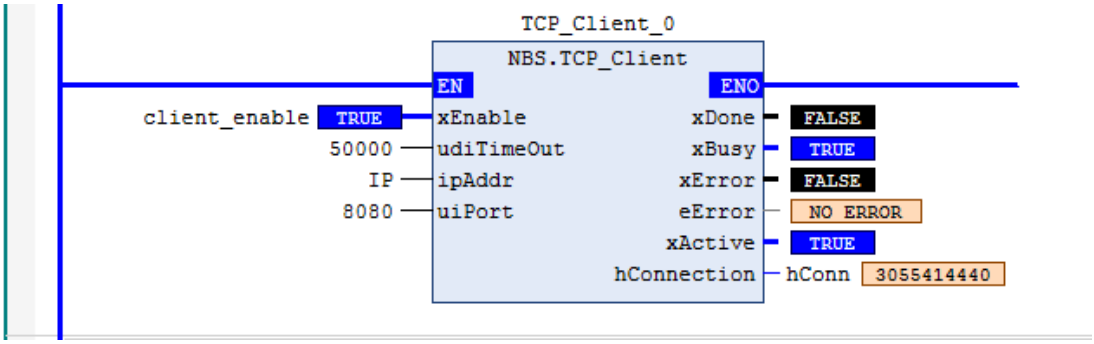
结构体成员	数据类型	描述
sAddr	STRING(80)	客户端所连接服务器的 IP 地址

程序示例

ST：创建 TCP 客户端通信服务。当 TCP_Client 指令 xEnable 为 TRUE 时，将创建本地服务器端与远端客户端 TCP 通信的有效句柄（hConnection 大于 0），xActive 为 TRUE。

```
TCP_Client_0(
    xEnable TRUE := client_enable TRUE,
    xDone FALSE => Done FALSE,
    xBusy TRUE => Busy TRUE,
    xError FALSE => bError FALSE,
    udiTimeOut 50000 := 50000,
    ipAddr:= IP,
    uiPort 8080 := 8080,
    eError=> ,
    xActive=> ,
    hConnection 3055415700 => hConn 3055415700 );
```

LD：创建 TCP 客户端通信服务。当 TCP_Client 指令 xEnable 为 TRUE 时，将创建本地服务器端与远端客户端 TCP 通信的有效句柄（hConnection 大于 0），xActive 为 TRUE。



注意事项

如果功能块出错，xError 将置 TRUE，eError 将被赋相关错误值（枚举型 ERROR 里的各种值），hConnection 将无效（等于 0）。详细请看以下错误码。

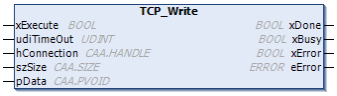
错误码	定义	描述
0	NO_ERROR	无错误
6000	FIRST_ERROR	保留
6001	TIME_OUT	超时
6002	INVALID_ADDR	客户端所连接服务器的 IP 地址无效
6003	INVALID_HANDLE	连接句柄无效
6004	INVALID_DATAPOINTER	数据指针无效
6005	INVALID_DATASIZE	数据大小无效
6006	UDP_RECEIVE_ERROR	UDP 接收错误
6007	UDP_SEND_ERROR	UDP 发送错误
6008	UDP_SEND_NOT_COMPLETE	UDP 发送未完成
6009	UDP_OPEN_ERROR	UDP 打开错误

6010	UDP_CLOSE_ERROR	UDP 关闭错误
6011	TCP_SEND_ERROR	TCP 发送错误
6012	TCP_RECEIVE_ERROR	TCP 接收错误
6013	TCP_OPEN_ERROR	TCP 打开错误
6014	TCP_CONNECT_ERROR	TCP 连接错误
6015	TCP_CLOSE_ERROR	TCP 关闭错误
6016	TCP_SERVER_ERROR	TCP 服务器错误
6017	WRONG_PARAMETER	参数错误
6018	ERROR_UNKNOWN	未知错误
6019	TCP_NO_CONNECTION	无 TCP 连接
6020	LOCTL_ERROR	内部错误（本机不支持）
6050	FIRST_MF	保留
6099	LAST_ERROR	保留

5.1.3 TCP_Write

将数据发送给 hConnection 所建立的连接。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
TCP_Write	TCP 通信数据发送	FB		TCP_Write_0(xExecute:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> , hConnection:= , szSize:= , pData:= , eError=>);	CAA Net Base Services

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	FALSE	若为 TRUE（上升沿信号），则启动功能块运行
udiTimeOut	超时时间	UDINT	0~2^32-1	0	请求建立连接的超时时间，单位：μs
hConnection	连接句柄	CAA.HANDLE	-	-	客户端与服务器的连接句柄
szSize	数据大小	CAA.SIZE	-	-	发送数据区域大小，byte
pData	发送缓存	CAA.PVOID	-	-	指向属性数据的指针，发送该数据到目标设备

输出变量

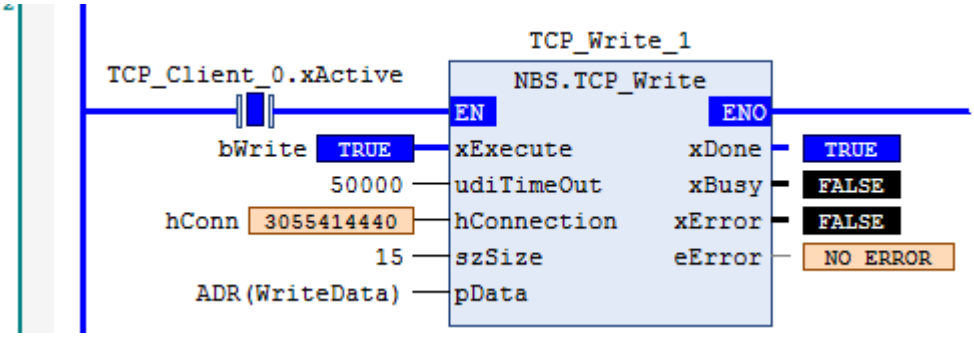
输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	FALSE-TRUE	-	客户端通信完成
xBusy	功能块运行中	BOOL	FALSE-TRUE	-	功能块运行中标志
xError	错误标志	BOOL	FALSE-TRUE	-	错误状态
eError	错误	ERROR	--	-	具体错误请参考 CAA Net Base Services 库的枚举体 ERROR

程序示例

ST: 此功能块的作用为将数据写入在 hConnection 所建立的连接。当 TCP_Write 指令 xExecute 为 TRUE 时，会将用户设置的发送缓冲区 pData 为首地址的长度为 szSize 的数据发送到目标设备。如果在超时时间内发送成功，xDone 置 TRUE。

```
TCP_Write_0(
    xExecute TRUE := bWrite TRUE,
    udiTimeOut 50000 := 50000,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    hConnection 3055415700 := hConn 3055415700,
    szSize 15 := 15,
    pData 3055428200 := ADR(WriteData),
    eError=> );
```

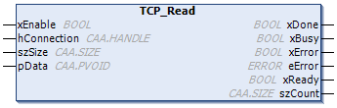
LD：此功能块的作用为将数据写入在 hConnection 所建立的连接。当 TCP_Write 指令 xExecute 为 TRUE 时，会将用户设置的发送缓冲区 pData 为首地址的长度为 szSize 的数据发送到目标设备。如果在超时时间内发送成功，xDone 置 TRUE。



5.1.4 TCP_Read

从 hConnection 所建立的连接的通信缓冲区中读取数据。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
TCP_Read	TCP 通信数据接收	FB		TCP_Read_0(xEnable:= , xDone=> , xBusy=> , xError=> , hConnection:= szSize:= , pData:= , eError=> , xReady=> , szCount=>);	CAA Net Base Services

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则启动功能块运行
hConnection	连接句柄	CAA.HANDLE	-	-	客户端与服务器的连接句柄
szSize	数据大小	CAA.SIZE	-	-	接收数据区域大小，byte
pData	发送缓存	CAA.PVOID	-	-	指向属性数据的指针，从目标设备接受数据

输出变量

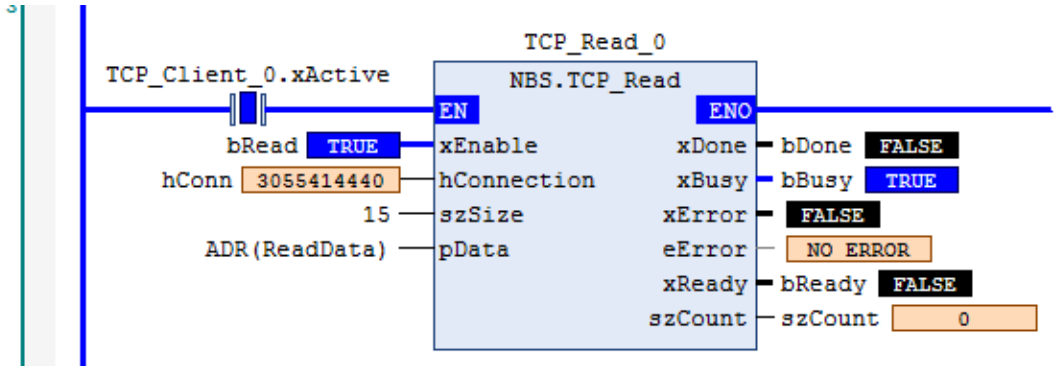
输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	FALSE-TRUE	-	客户端通信完成
xBusy	功能块运行中	BOOL	FALSE-TRUE	-	功能块运行中标志
xError	错误标志	BOOL	FALSE-TRUE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考 CAA Net Base Services 库的枚举体 ERROR
xReady	成功建立连接标志位	BOOL	FALSE-TRUE	-	从缓冲区内读取数据，如果数据不为空，置位一个扫描周期的标志位
szCount	数据大小	CAA.SIZE	-	-	接收数据区域的实际大小

程序示例

ST：从 hConnection 所建立的连接的通信缓冲区中读取数据。当 TCP_Read 指令 xEnable 为 TRUE 时，将会从 TCP 通信缓冲区读取数据，xBusy 为 TRUE。若读取成功，xDone 置位一个扫描周期；读取的数据会被放置到地址为 pData 的变量里；同时 xReady 置位一个扫描周期；接收数据区域的实际大小值将会赋给 szCount，在一个扫描周期后 szCount 的值会被清零。

```
TCP_Read_0(
    xEnable TRUE := bRead TRUE,
    xDone FALSE => bDone FALSE,
    xBusy TRUE => bBusy TRUE,
    xError=> ,
    hConnection 3055415700 := hConn 3055415700,
    szSize 15 := 15,
    pData 3055426544 := ADR(ReadData),
    eError=> ,
    xReady FALSE => bReady FALSE,
    szCount 0 => szCount 0);
```

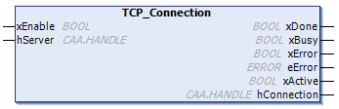
LD：从 hConnection 所建立的连接的通信缓冲区中读取数据。当 TCP_Read 指令 xEnable 为 TRUE 时，将会从 TCP 通信缓冲区读取数据，xBusy 为 TRUE。若读取成功，xDone 置位一个扫描周期；读取的数据会被放置到地址为 pData 的变量里；同时 xReady 置位一个扫描周期；接收数据区域的实际大小值将会赋给 szCount，在一个扫描周期后 szCount 的值会被清零。



5.1.5 TCP_Connection

创建 TCP 连接并连接到服务器。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
TCP_Connection	创建 TCP 连接，并连接到服务器	FB		TCP_Connection_0(xEnable:= , xDone=> , xBusy=> , xError=> , hServer:= , eError=> , xActive=> , hConnection=>);	CAA Net Base Services

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则启动功能块运行
hServer	服务器端句柄	CAA.HANDLE	-	-	TCP 服务器端的句柄

输出变量

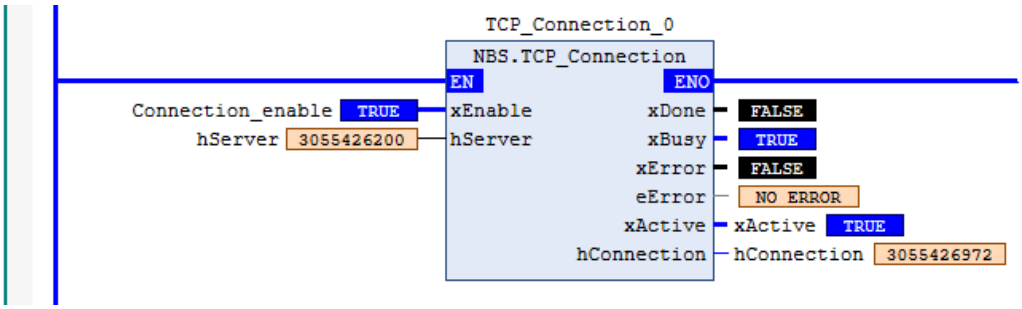
输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	FALSE-TRUE	-	客户端通信完成
xBusy	功能块运行中	BOOL	FALSE-TRUE	-	功能块运行中标志
xError	错误标志	BOOL	FALSE-TRUE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考 CAA Net Base Services 库的枚举体 ERROR
xActive	连接成功标志	BOOL	FALSE-TRUE	-	客户端与服务器连接成功标志
hConnection	连接句柄	CAA.HANDLE	-	-	客户端与服务器的连接句柄

程序示例

ST：建立 TCP 连接并连接到服务器。当 TCP_Connection 指令 xEnable 为 TRUE 时，本地服务器端将监听远端客户端的连接请求，若客户端和服务器连接成功，将创建服务器与远端客户的通信连接句柄 hConnection。

```
TCP_Connection_0(
xEnable TRUE := Connection_enable TRUE,
xDone=> ,
xBusy=> ,
xError=> ,
hServer 3055413824 := hServer 3055413824,
eError=> ,
xActive TRUE => bActive TRUE,
hConnection 3055413640 => hConn 3055413640 );
```

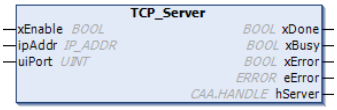
LD：建立 TCP 连接并连接到服务器。当 TCP_Connection 指令 xEnable 为 TRUE 时，本地服务器端将监听远端客户端的连接请求，若客户端和服务器连接成功，将创建服务器与远端客户的通信连接句柄 hConnection。



5.1.6 TCP_Server

创建 TCP 服务端的通信服务。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
TCP_Server	创建 TCP 服务端	FB		TCP_Server_0(xEnable:= , xDone=> , xBusy=> , xError=> , ipAddr:= , uiPort:= , eError=> , hServer=>);	CAA Net Base Services

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则启动功能块运行
IpAddr	服务器 IP 地址	IP_ADDR	-	-	服务器 IP 地址，具体参考 CAA Net Base Services 库的结构体 IP_ADDR
uiPort	本地端口号	UINT	0-65535	-	服务器端口号

输出变量

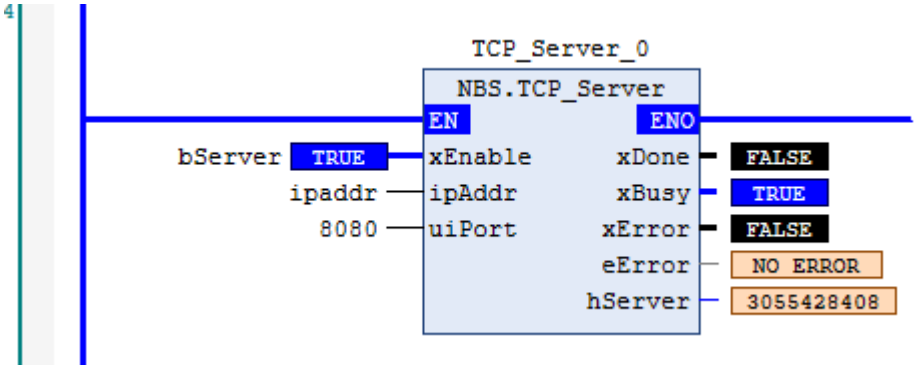
输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	FALSE-TRUE	-	客户端通信完成
xBusy	功能块运行中	BOOL	FALSE-TRUE	-	功能块运行中标志
xError	错误标志	BOOL	FALSE-TRUE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考 CAA Net Base Services 库的枚举体 ERROR
hServer	服务器句柄	CAA.HANDLE	-	-	服务器句柄

程序示例

ST：创建 TCP 服务器端通信服务。当 TCP_Server 指令 xEnable 为 TRUE 时，将创建本地服务器端与远端客户端 TCP 通信的有效句柄（hServer 大于 0），xBusy 为 TRUE。

```
TCP_Server_0(
xEnable TRUE := bServer TRUE,
xDone=> ,
xBusy TRUE => xBusy TRUE,
xError=> ,
ipAddr:= ipaddr,
uiPort 8080 := 8080,
eError=> ,
hServer 3055430680 => hServer 3055430680 );
```

LD：创建 TCP 服务器端通信服务。当 TCP_Server 指令 xEnable 为 TRUE 时，将创建本地服务器端与远端客户端 TCP 通信的有效句柄（hServer 大于 0），xBusy 为 TRUE。



5.2 自由 UDP 通信指令

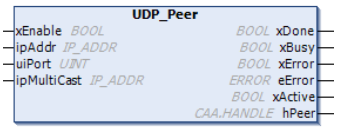
5.2.1 指令列表

指令类别	名称	FB/FC	功能
自由 UDP 通信指令	UDP_Peer	FB	创建 UDP 通信连接
	UDP_Receive	FB	UDP 通信数据接收
	UDP_Send	FB	UDP 通信数据发送

5.2.2 UDP_Peer

创建 UDP 通信连接。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
UDP_Peer	创建 UDP 通信连接	FB		UDP_Peer_0(xEnable:= , xDone=> , xBusy=> , xError=> , ipAddr:= , uiPort:= , ipMultiCast:= , eError=> , xActive=> , hPeer=>);	CAA Net Base Services

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则启动功能块运行
IpAddr	IP 地址	IP_ADDR	-	-	IP 地址
uiPort	通信端口号	UINT	0-65535	-	通信端口号
IpMultiCast	多播地址	IP_ADDR	-	-	多播地址

输出变量

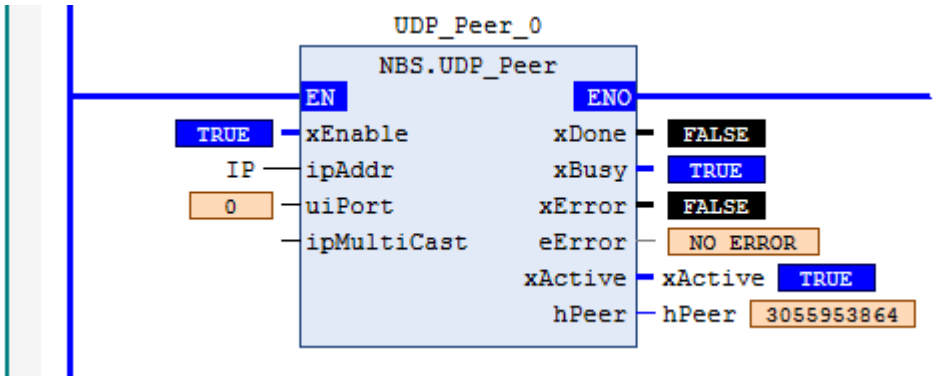
输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	FALSE-TRUE	-	客户端通信完成
xBusy	功能块运行中	BOOL	FALSE-TRUE	-	功能块运行中标志
xError	错误标志	BOOL	FALSE-TRUE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考 CAA Net Base Services 库的枚举体 ERROR
hPeer	通信句柄	CAA.HANDLE	-	-	UDP 通信句柄

程序示例

ST：创建 UDP 通信连接。当 UDP_Peer 指令 xEnable 为 TRUE 时，将创建 UDP 通信的有效句柄（hPeer 大于 0），xBusy 及 xActive 为 TRUE。

```
UDP_Peer_0(
    xEnable TRUE := enable TRUE,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    ipAddr:= IP,
    uiPort 5000 := 5000,
    ipMultiCast:= IP_MULTICAST,
    eError=> ,
    xActive TRUE => xActive TRUE,
    hPeer 3055429312 => hPeer 3055429312 );
```

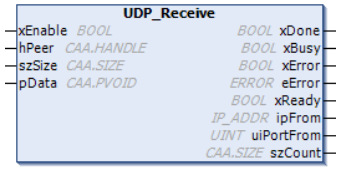
LD：创建 UDP 通信连接。当 UDP_Peer 指令 xEnable 为 TRUE 时，将创建 UDP 通信的有效句柄（hPeer 大于 0），xBusy 及 xActive 为 TRUE。



5.2.3 UDP_Receive

UDP 通信数据接收。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
UDP_Receive	UDP 通信数据接收	FB		UDP_Receive_0(xEnable:= , xDone=> , xBusy=> , xError=> , hPeer:= , szSize:= , pData:= , eError=> , xReady=> , ipFrom=> , uiPortFrom=> , szCount=>);	CAA Net Base Services

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xEnable	功能块使能	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则启动功能块运行
hPeer	通信句柄	CAA.HANDLE	-	-	UDP 通信句柄
szSize	数据大小	CAA.SIZE	-	-	接收数据区域大小，byte
pData	发送缓存数据	CAA.PVOID	-	-	指向属性数据的指针，从目标设备接收数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	FALSE-TRUE	-	客户端通信完成
xBusy	功能块运行中	BOOL	FALSE-TRUE	-	功能块运行中标志
xError	错误标志	BOOL	FALSE-TRUE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考 CAA Net Base Services 库的枚举体 ERROR
xReady	成功建立连接标志	BOOL	FALSE-TRUE	-	从缓冲区内读取数据，如果数据不为空，置位一个扫描周期的标志位
ipFrom	当前数据包源 IP	IP_ADDR	-	-	当前数据包源 IP

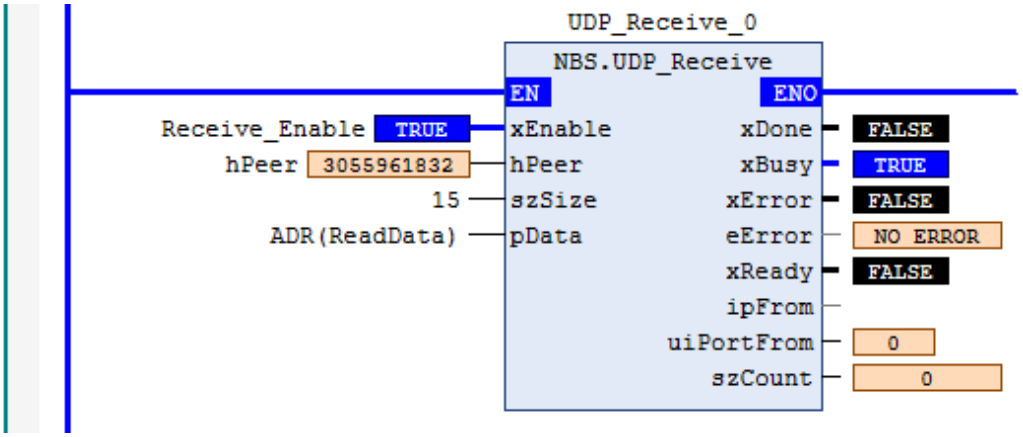
uiPortFrom	当前数据包源端口号	UINT	0-65535	-	当前数据包源端口号
szCount	数据大小	CAA.SIZE	-	-	接收数据区域的实际大小

程序示例

ST：UDP 通信数据接收指令。当 UDP_Receive 指令 xEnable 为 TRUE 时，将从 UDP 通信缓冲区读取数据，xBusy 为 TRUE。若读取成功，xDone 置位一个扫描周期；读取的数据会被放置到地址为 pData 的变量里；同时 xReady 置位一个扫描周期；接收数据区域的实际大小值将会赋给 szCount，在一个扫描周期后 szCount 的值会被清零。

```
UDP_Receive_0(
    xEnable TRUE := bReceive TRUE,
    xDone FALSE => xDone FALSE,
    xBusy TRUE => xBusy TRUE,
    xError=> ,
    hPeer 3055960720 := hPeer 3055960720,
    szSize 15 := 15,
    pData 3055946732 := ADR(ReadData),
    eError=> ,
    xReady FALSE => xReady FALSE,
    ipFrom=> ipfrom,
    uiPortFrom 0 => portfrom 0,
    szCount 0 => szCount 0);
```

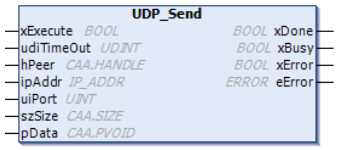
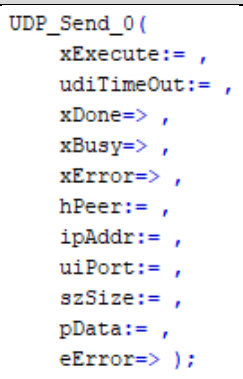
LD：UDP 通信数据接收指令。当 UDP_Receive 指令 xEnable 为 TRUE 时，将从 UDP 通信缓冲区读取数据，xBusy 为 TRUE。若读取成功，xDone 置位一个扫描周期；读取的数据会被放置到地址为 pData 的变量里；同时 xReady 置位一个扫描周期；接收数据区域的实际大小值将会赋给 szCount，在一个扫描周期后 szCount 的值会被清零。



5.2.4 UDP_Send

UDP 通信数据发送。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
UDP_Send	UDP 通信数据发送	FB			CAA Net Base Services


```
UDP_Send_0(
    xExecute:= ,
    udiTimeOut:= ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    hPeer:= ,
    ipAddr:= ,
    uiPort:= ,
    szSize:= ,
    pData:= ,
    eError=> );
```

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	FALSE	若为 TRUE（上升沿信号），则启动功能块运行
udiTimeOut	超时时间	UDINT	0~2^32-1	0	建立通信后执行数据发送的超时时间，单位：μs
hPeer	通信句柄	CAA.HANDLE	-	-	UDP 通信句柄
ipAddr	目标 IP	CAA.PVOID	-	-	发送数据的目标 IP
uiPort	通信端口号	UINT	0-65535	0	通信端口号
szSize	数据大小	CAA.SIZE	-	-	发送数据区域大小，byte
pData	发送缓存数据	CAA.PVOID	-	-	指向属性数据的指针，向目标设备发送数据

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	完成	BOOL	FALSE-TRUE	-	客户端通信完成
xBusy	功能块运行中	BOOL	FALSE-TRUE	-	功能块运行中标志
xError	错误标志	BOOL	FALSE-TRUE	-	错误状态
eError	错误	ERROR	-	-	具体错误请参考 CAA Net Base Services 库的枚举体 ERROR

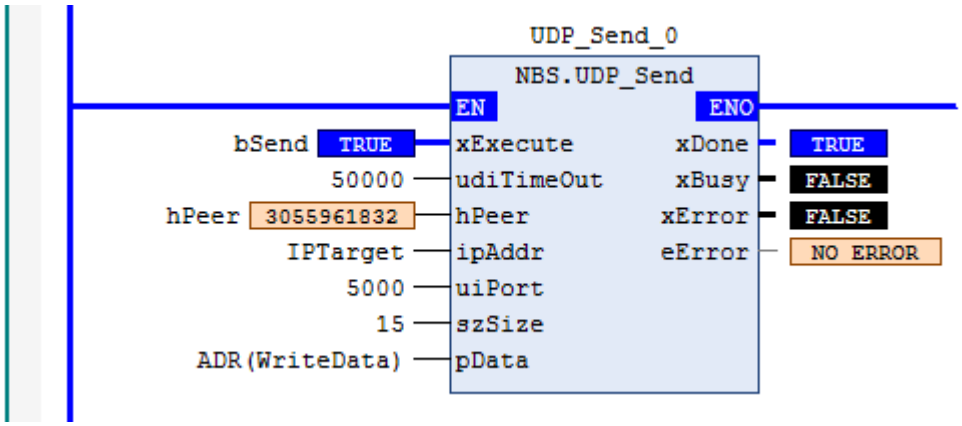
xReady	成功建立连接标志	BOOL	FALSE-TRUE	-	发送缓冲区的数据，如果数据不为空，置位一个扫描周期的标志位
szCount	数据大小	CAA.SIZE	-	-	发送数据的实际大小

程序示例

ST：UDP 通信数据发送指令。当 UDP_Send 指令 xExecute 为 TRUE 时，会将用户设置的发送缓冲区 pData 为首地址的长度为 szSize 的数据发送到目标设备。如果在超时时间内发送成功，xDone 置 TRUE。

```
UDP_Send_0(
    xExecute TRUE := Send_Enable TRUE,
    udiTimeOut 50000 := 50000,
    xDone TRUE => send_xDone TRUE,
    xBusy FALSE => send_xBusy FALSE,
    xError=> ,
    hPeer 3055960720 := hPeer 3055960720,
    ipAddr:= IP_Target,
    uiPort 5000 := port 5000,
    szSize 15 := 15,
    pData 3055946612 := ADR(WriteData),
    eError=> );
```

LD：UDP 通信数据发送指令。当 UDP_Send 指令 xExecute 为 TRUE 时，会将用户设置的发送缓冲区 pData 为首地址的长度为 szSize 的数据发送到目标设备。如果在超时时间内发送成功，xDone 置 TRUE。



5.3 EtherCAT 通信指令

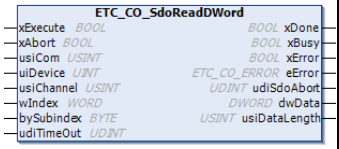
5.3.1 指令列表

指令类别	名称	FB/FC	功能
EtherCAT 通信指令	ETC_CO_SdoReadDWord	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoRead4	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoRead	FB	EtherCAT 从站 SDO 读取
	ETC_CO_SdoWrite_Dword	FB	EtherCAT 从站 SDO 写入
	ETC_CO_SdoWrite4	FB	EtherCAT 从站 SDO 写入
	ETC_CO_SdoWrite	FB	EtherCAT 从站 SDO 写入
	IoDrvEtherCAT_Diag	FB	EtherCAT 主站实例
	ETCSlave	FB	EtherCAT 从站实例

5.3.2 ETC_CO_SdoReadDWord

EtherCAT 从站 SDO 数据读取，且读取对象字典数值长度不大于 4 个字节。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ETC_CO_SdoReadDWord	读取 EtherCAT 从站对象字典 SDO 数据	FB		ETC_CO_SdoReadDWord_0(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=> , dwData=> , usiDataLength=>);	IODrvEtherCAT

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为 TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT 主站个数	USINT	1-2	1	如果只使用一个 EtherCAT 主站， 则 usiCom 为 1。 若使用多个主站， 则第一个主站为 1，第二个主站为 2，以此类推。
uiDevice	从站站号	UINT	1-65535	0	从站站号如 1001，1002 等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	0	例如伺服的控制字对象字典 16#6040
bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引 16#00

udiTimeOut	超时时间	UDINT	1-65535	0	读取从站参数的超时时间，单位：ms

输出变量

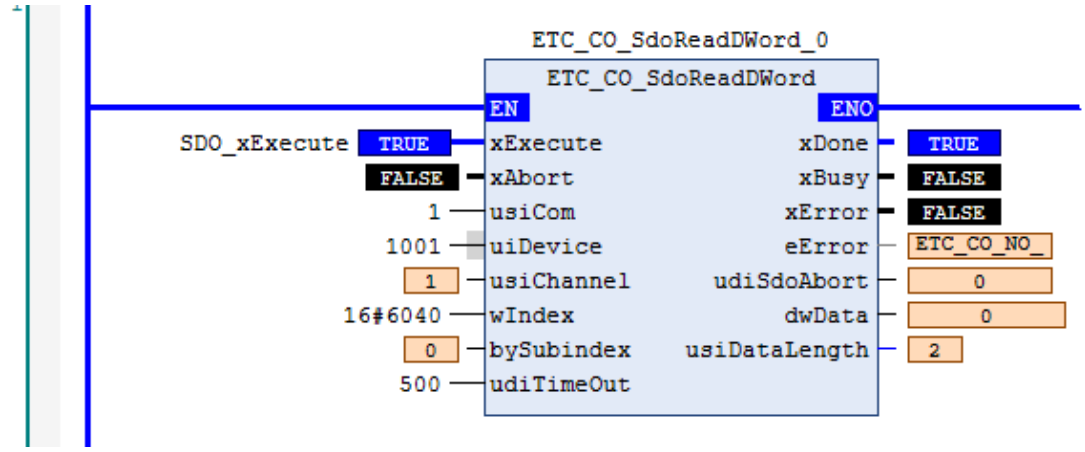
输出变量	名称	数据类型	有效范围	初始值	描述
xDone	读取完成	BOOL	TRUE-FALSE	-	读取参数已经成功，xDone 为 TRUE
xBusy	读取参数中	BOOL	TRUE-FALSE	-	读取参数中，xBusy 为 TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ETC_CO_ERROR	-	-	具体错误请参考 IODrvEtherCAT 库的枚举体 ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-2^32-1	0	当设备出错时，可从此输出变量读取更多错误信息
dwData	读取的数据值	DWORD	0-2^32-1	0	从从站读取到的 SDO 数据值
usiDataLength	读取到的数据长度	USINT	0-255	0	从从站读取到的 SDO 数据值长度

程序示例

ST：读伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。

```
ETC_CO_SdoReadDWord_0(
    xExecute TRUE := SDO_xExecute TRUE,
    xAbort:= ,
    usiCom:= ,
    uiDevice 1001 := 1001,
    usiChannel:= ,
    wIndex 24640 := 16#6040,
    bySubindex:= ,
    udiTimeOut 500 := 500,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> ,
    dwData 0 => SDO_Data 0,
    usiDataLength 2 => SDO_Length 2 );
```

LD：读伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。



注意事项

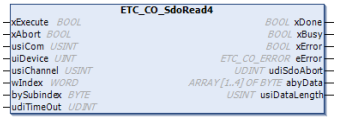
如果使用功能块中出现错误，xError 将置 TRUE，eError 将被赋相关错误值（枚举型 ETC_CO_ERROR 里的各种值）。详细请看以下错误代码。

名称	数据类型	描述
ETC_CO_NO_ERROR	WORD	-
ETC_CO_FIRST_ERROR	WORD	-
ETC_CO_OTHER_ERROR	WORD	此错误保留给所有 SDO 通道繁忙中！ 禁止更改
ETC_CO_DATA_OVERFLOW	WORD	-
ETC_CO_TIMEOUT	WORD	-
ETC_CO_FIRST_MF	WORD	-
ETC_CO_LAST_ERROR	WORD	-

5.3.3 ETC_CO_SdoRead4

EtherCAT 从站 SDO 数据读取，且读取对象字典数值长度不大于 4 个字节。与 ETC_CO_SdoReadDWord 功能块不同在于，ETC_CO_SdoRead4 是将读取到的数值存储在字节类型数组中。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ETC_CO_SdoRead4	读取 EtherCAT 从站对象字典 SDO 数据	FB		ETC_CO_SdoRead4_0(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=> , abyData=> , usiDataLength=>);	IODrvEtherCAT

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为 TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT 主站个数	USINT	1-2	1	如果只使用一个 EtherCAT 主站， 则 usiCom 为 1。 若使用多个主站， 则第一个主站为 1，第二个主站为 2，以此类推。
uiDevice	从站站号	UINT	1-65535	0	从站站号如 1001，1002 等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	0	例如伺服的控制字对象字典 16#6040
bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引 16#00

udiTimeOut	超时时间	UDINT	1-65535	0	读取从站参数的超时时间，单位：ms

输出变量

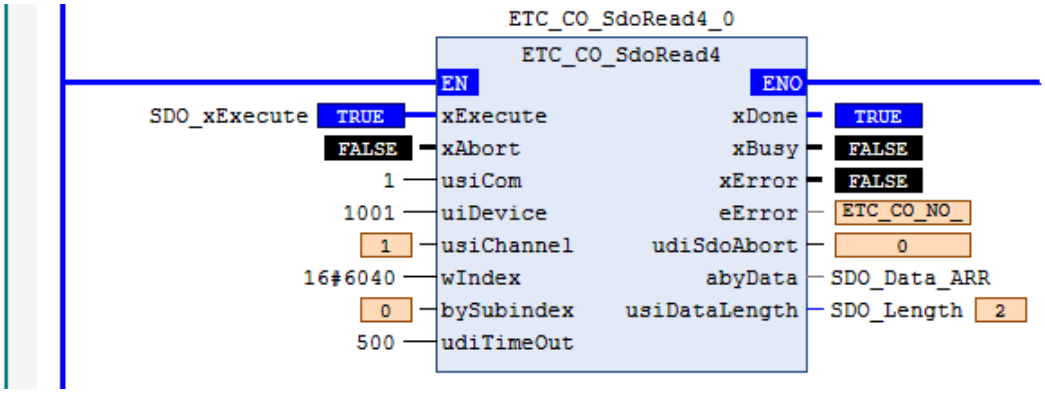
输出变量	名称	数据类型	有效范围	初始值	描述
xDone	读取完成	BOOL	TRUE-FALSE	-	读取参数已经成功，xDone 为 TRUE
xBusy	读取参数中	BOOL	TRUE-FALSE	-	读取参数中，xBusy 为 TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ETC_CO_ERROR	-	-	具体错误请参考 IODrvEtherCAT 库的枚举体 ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-2^32-1	0	当设备出错时，可从此输出变量读取更多错误信息
abyData	读取的数据值	ARRAY[1..4] OF BYTE	-	0	从从站读取到的 SDO 数据值，数值填充顺序从数组索引 1 到 4
usiDataLength	读取到的数据长度	USINT	0-255	0	从从站读取到的 SDO 数据值长度

程序示例

ST：读伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。（注意：读取的数值存储在字节类型数组，数组大小为 4 个字节）

```
ETC_CO_SdoRead4_0(
    xExecute TRUE := SDO_xExecute TRUE,
    xAbort:= ,
    usiCom:= ,
    uiDevice 1001 := 1001,
    usiChannel:= ,
    wIndex 24640 := 16#6040,
    bySubindex:= ,
    udiTimeOut 500 := 500,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> ,
    abyData=> SDO_Data_ARR,
    usiDataLength 2 => SDO_Length 2 );
```

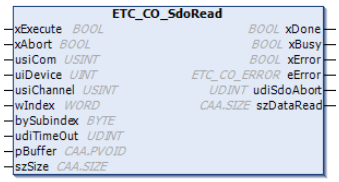
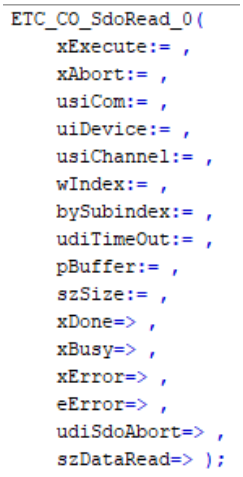
LD：读伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。（注意：读取的数值存储在字节类型数组，数组大小为 4 个字节）


ETC_CO_SdoRead4_0
ETC_CO_SdoRead4
EN
ENO
SDO_xExecute
TRUE
FALSE
1
1001
1
16#6040
0
500
xExecute
xAbort
usiCom
uiDevice
usiChannel
wIndex
bySubindex
udiTimeOut
xDone
xBusy
xError
eError
udiSdoAbort
abyData
usiDataLength
TRUE
FALSE
FALSE
ETC_CO_NO_
0
SDO_Data_ARR
SDO_Length
2

5.3.4 ETC_CO_SdoRead

EtherCAT 从站 SD0 数据读取，与 ETC_CO_SdoReadDWord 或 ETC_CO_SdoRead4 功能块不同，ETC_CO_SdoRead 功能块读取的对象字典数值长度可大于 4 个字节。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ETC_CO_SdoRead	读取 EtherCAT 从站对象字典 SDO 数据	FB			IODrvEtherCAT


```
ETC_CO_SdoRead_0(
    xExecute:= ,
    xAbort:= ,
    usiCom:= ,
    uiDevice:= ,
    usiChannel:= ,
    wIndex:= ,
    bySubindex:= ,
    udiTimeOut:= ,
    pBuffer:= ,
    szSize:= ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> ,
    szDataRead=> );
```

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为 TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT 主站个数	USINT	1-2	1	如果只使用一个 EtherCAT 主站，则 usiCom 为 1。若使用多个主站，则第一个主站为 1，第二个主站为 2，以此类推。
uiDevice	从站站号	UINT	1-65535	0	从站站号如 1001，1002 等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	0	例如伺服的控制字对象字典 16#6040

bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引 16#00
udiTimeOut	超时时间	UDINT	1-65535	0	读取从站参数的超时时间，单位：ms
pBuffer	数据缓冲区的指针	CAA.PVOID	-	0	数据成功读取后的存储区域
szSize	数据缓冲区大小	CAA.SIZE	-	0	数据缓冲（pBuffer）的大小

输出变量

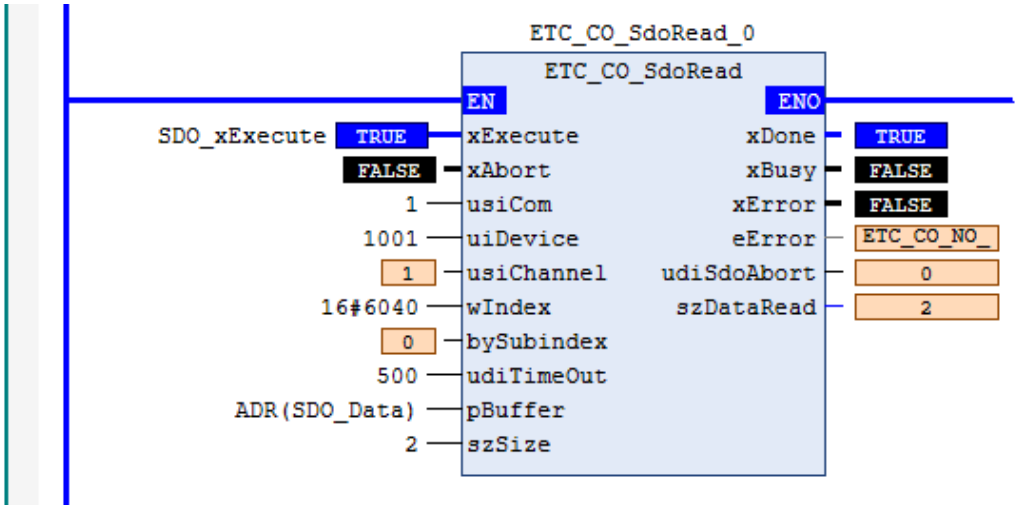
输出变量	名称	数据类型	有效范围	初始值	描述
xDone	读取完成	BOOL	TRUE-FALSE	-	读取参数已经成功，xDone 为 TRUE
xBusy	读取参数中	BOOL	TRUE-FALSE	-	读取参数中，xBusy 为 TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ETC_CO_ERROR	-	-	具体错误请参考 IODrvEtherCAT 库的枚举体 ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-2^32-1	0	当设备出错时，可从此输出变量读取更多错误信息

程序示例

ST：读伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。

```
ETC_CO_SdoRead_0(
    xExecute TRUE := SDO_xExecute TRUE,
    xAbort:= ,
    usiCom:= ,
    uiDevice 1001 := 1001,
    usiChannel:= ,
    wIndex 24640 := 16#6040,
    bySubindex:= ,
    udiTimeOut 500 := 500,
    pBuffer 3055982836 := ADR(SDO_Data),
    szSize 2 := 2,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> ,
    szDataRead 2 => SDO_Length 2 );
```

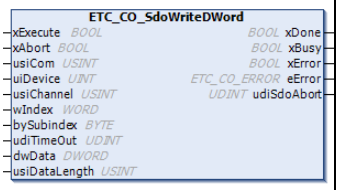
LD：读伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。


ETC_CO_SdoRead_0
ETC_CO_SdoRead
EN
ENO
SDO_xExecute
TRUE
xExecute
xDone
TRUE
FALSE
xAbort
xBusy
FALSE
1
usiCom
xError
FALSE
1001
uiDevice
eError
ETC_CO_NO_
1
usiChannel
udiSdoAbort
0
16#6040
wIndex
szDataRead
2
0
bySubindex
500
udiTimeOut
ADR(SDO_Data)
pBuffer
2
szSize

5.3.5 ETC_CO_SdoWriteDWord

EtherCAT 从站 SDO 数据写入，且写入对象字典数值长度不大于 4 个字节。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ETC_CO_SdoWriteDWord	EtherCAT 从站对象字典 SDO 数据写入	FB		ETC_CO_SdoWriteDWord_0(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , dwData:= , usiDataLength:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=>);	IODrvEtherCAT

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为 TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT 主站个数	USINT	1-2	1	如果只使用一个 EtherCAT 主站， 则 usiCom 为 1。 若使用多个主站， 则第一个主站为 1，第二个主站为 2，以此类推。
uiDevice	从站站号	UINT	1-65535	0	从站站号如 1001，1002 等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	0	例如伺服的控制字对象字典 16#6040
bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引 16#00

udiTimeOut	超时时间	UDINT	1-65535	0	读取从站参数的超时时间，单位：ms
dwData	写入的数据值	DWORD	0-4294967295	0	写入从站 SDO 的数据值
usiDataLength	写入的数据长度	USINT	0-255	0	写入从站 SDO 数据值的长度

输出变量

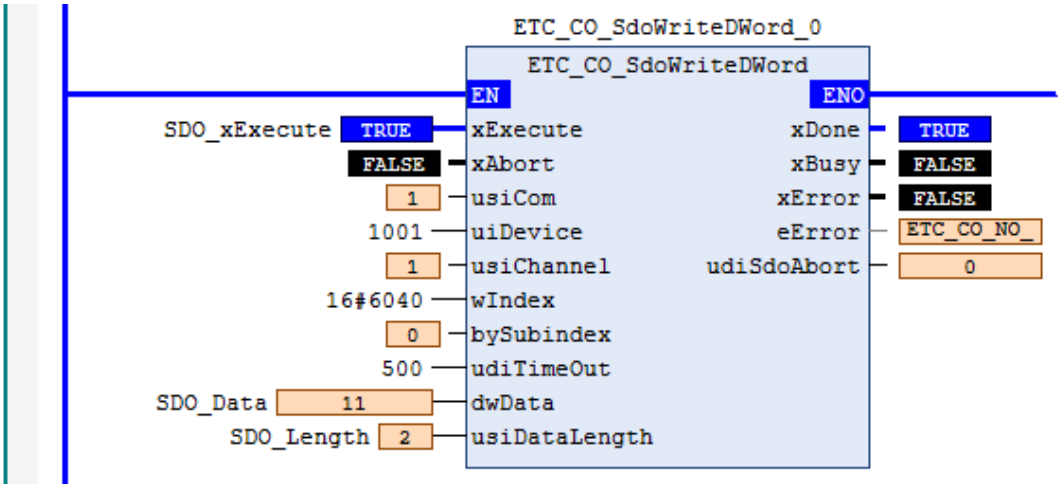
输出变量	名称	数据类型	有效范围	初始值	描述
xDone	写入完成	BOOL	TRUE-FALSE	-	写入参数已经成功，xDone 为 TRUE
xBusy	写入参数中	BOOL	TRUE-FALSE	-	写入参数中，xBusy 为 TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ETC_CO_ERROR	-	-	具体错误请参考 IODrvEtherCAT 库的枚举体 ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-2^32-1	0	当设备出错时，可从此输出变量读取更多错误信息

程序示例

ST：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。

```
ETC_CO_SdoWriteDWord_0(
    xExecute TRUE := SDO_xExecute TRUE,
    xAbort:= ,
    usiCom:= ,
    uiDevice 1001 := 1001,
    usiChannel:= ,
    wIndex 24640 := 16#6040,
    bySubindex:= ,
    udiTimeOut 500 := 500,
    dwData 20 := SDO_Data 20,
    usiDataLength 2 := SDO_Length 2,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> );
```

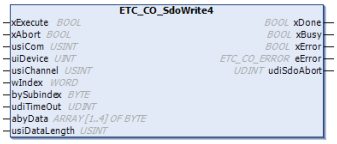
LD：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。


ETC_CO_SdoWriteDWord_0
ETC_CO_SdoWriteDWord
EN
ENO
SDO_xExecute
TRUE
xExecute
xDone
TRUE
FALSE
xAbort
xBusy
FALSE
1
usiCom
xError
FALSE
1001
uiDevice
eError
ETC_CO_NO_
1
usiChannel
udiSdoAbort
0
16#6040
wIndex
0
bySubindex
500
udiTimeOut
SDO_Data
11
dwData
SDO_Length
2
usiDataLength

5.3.6 ETC_CO_SdoWrite4

EtherCAT 从站 SD0 数据写入，且写入对象字典数值长度不大于 4 个字节。与 ETC_CO_SdoWriteDWord 功能块不同在于，ETC_CO_SdoWrite4 是将数值以字节类型数组是写入。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ETC_CO_SdoWrite4	EtherCAT 从站对象字典 SDO 数据写入	FB		ETC_CO_SdoWrite4_0(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , abyData:= , usiDataLength:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=>);	IODrvEtherCAT

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为 TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT 主站个数	USINT	1-2	1	如果只使用一个 EtherCAT 主站， 则 usiCom 为 1。 若使用多个主站， 则第一个主站为 1，第二个主站为 2，以此类推。
uiDevice	从站站号	UINT	1-65535	0	从站站号如 1001，1002 等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	0	例如伺服的控制字对象字典 16#6040

bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引 16#00
udiTimeOut	超时时间	UDINT	1-65535	0	读取从站参数的超时时间，单位：ms
abyData	写入的数据值	ARRAY[1..4] OF BYTE	-	0	写入从站 SDO 的数据值，数值填充顺序从数组索引 1 到 4
usiDataLength	写入的数据长度	USINT	0-255	0	写入从站 SDO 数据值的长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xDone	写入完成	BOOL	TRUE-FALSE	-	写入参数已经成功，xDone 为 TRUE
xBusy	写入参数中	BOOL	TRUE-FALSE	-	写入参数中，xBusy 为 TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ETC_CO_ERROR	-	-	具体错误请参考 IODrvEtherCAT 库的枚举体 ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-2^32-1	0	当设备出错时，可从此输出变量读取更多错误信息

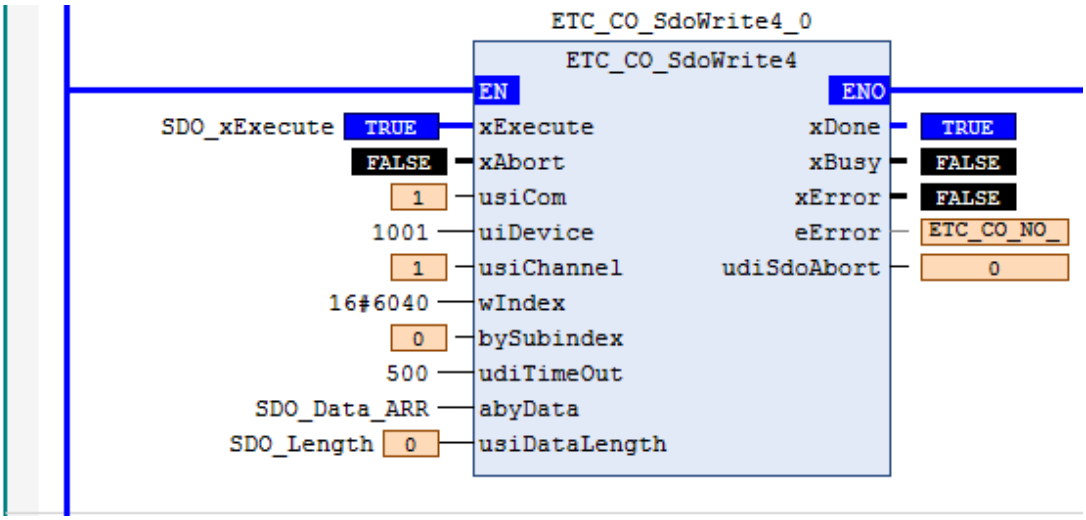
程序示例

ST：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。（注意：写入的数值存储在字节类型数组，数组大小为 4 个字节）

```
ETC_CO_SdoWrite4_0(
    xExecute TRUE := SDO_xExecute TRUE,
    xAbort:= ,
    usiCom:= ,
    uiDevice 1001 := 1001,
    usiChannel:= ,
    wIndex 24640 := 16#6040,
    bySubindex:= ,
    udiTimeOut 500 := 500,
    abyData:= SDO_Data_ARR,
    usiDataLength 0 := SDO_Length 0,
    xDone TRUE => xDone TRUE,
    xBusy FALSE => xBusy FALSE,
    xError=> ,
    eError=> ,
    udiSdoAbort=> );
```

LD：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。注意：写入的数值存储在字节类型数

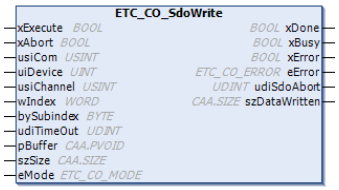
组，数组大小为 4 个字节）


ETC_CO_SdoWrite4_0
ETC_CO_SdoWrite4
EN
ENO
SDO_xExecute
TRUE
xExecute
xDone
TRUE
FALSE
xAbort
xBusy
FALSE
1
usiCom
xError
FALSE
1001
uiDevice
eError
ETC_CO_NO_
1
usiChannel
udiSdoAbort
0
16#6040
wIndex
0
bySubindex
500
udiTimeOut
SDO_Data_ARR
abyData
SDO_Length
0
usiDataLength

5.3.7 ETC_CO_SdoWrite

EtherCAT 从站 SDO 数据写入。与 ETC_CO_SdoWriteDWord 或 ETC_CO_SdoWrite4 功能块不同，ETC_CO_SdoWrite 功能块写入的对象字典数值长度可大于 4 个字节。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ETC_CO_SdoWrite	EtherCAT 从站对象字典 SDO 数据写入	FB		ETC_CO_SdoWrite_0(xExecute:= , xAbort:= , usiCom:= , uiDevice:= , usiChannel:= , wIndex:= , bySubindex:= , udiTimeOut:= , pBuffer:= , szSize:= , eMode:= , xDone=> , xBusy=> , xError=> , eError=> , udiSdoAbort=> , szDataWritten=>);	IODrvEtherCAT

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xExecute	功能块使能	BOOL	TRUE-FALSE	-	若为 TRUE（上升沿信号），则启动功能块运行
xAbort	功能块中止执行	BOOL	TRUE-FALSE	FALSE	若为 TRUE，则当前功能块读取命令将终止
usiCom	EtherCAT 主站个数	USINT	1-2	1	如果只使用一个 EtherCAT 主站，则 usiCom 为 1。若使用多个主站，则第一个主站为 1，第二个主站为 2，以此类推。
uiDevice	从站站号	UINT	1-65535	0	从站站号如 1001，1002 等
usiChannel	保留	USINT	1	1	保留
wIndex	对象字典主索引	WORD	1-65535	0	例如伺服的控制字对象字典 16#6040

bySubindex	对象字典子索引	BYTE	0-255	0	例如伺服控制字对象字典子索引 16#00
udiTimeOut	超时时间	UDINT	1-65535	0	读取从站参数的超时时间，单位：ms
pBuffer	数据缓冲区的指针	CAA.PVOID	-	-	写入的数据指针起始地址
szSize	数据缓冲区大小	CAA.SIZE	-	-	写入数据长度
eMode	写入模式选择	ETC_CO_MODE	-	-	写入模式选择

输出变量

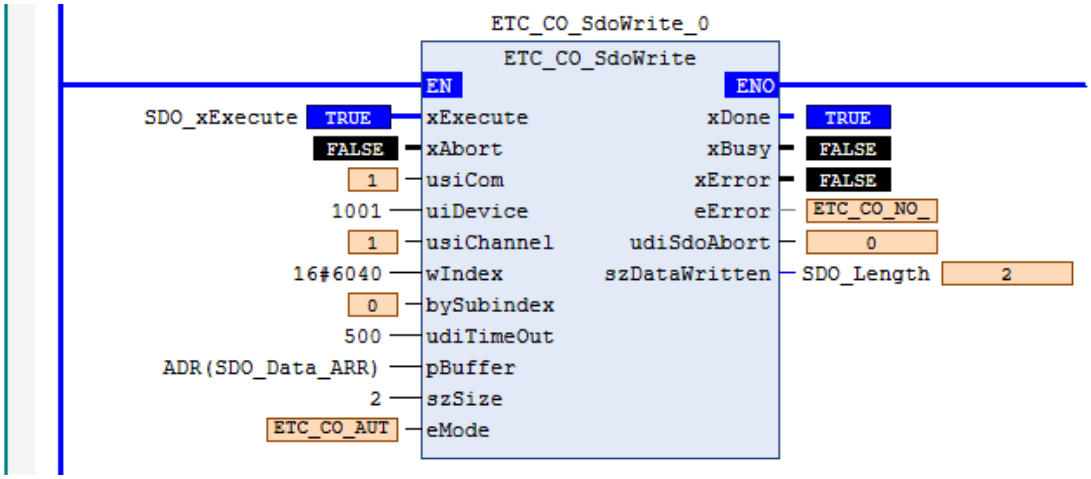
输出变量	名称	数据类型	有效范围	初始值	描述
xDone	写入完成	BOOL	TRUE-FALSE	-	写入参数已经成功，xDone 为 TRUE
xBusy	写入参数中	BOOL	TRUE-FALSE	-	写入参数中，xBusy 为 TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
eError	错误	ETC_CO_ERROR	-	-	具体错误请参考 IODrvEtherCAT 库的枚举体 ETC_CO_ERROR
udiSdoAbort	中止	UDINT	0-2^32-1	0	当设备出错时，可从此输出变量读取更多错误信息
szDataWritten	完成写入数据长度	CAA.SIZE	-	-	完成写入数据长度

程序示例

ST：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。

```
ETC_CO_SdoWrite_0(
    xExecute TRUE := SDO_xExecute TRUE,
    xAbort:= ,
    usiCom:= ,
    uiDevice 1001 := 1001,
    usiChannel:= ,
    wIndex 24640 := 16#6040,
    bySubindex:= ,
    udiTimeOut 500 := 500,
    pBuffer 3056096892 := ADR(SDO_Data 0 ),
    szSize 2 := 2,
    eMode:= ,
    xDone=> ,
    xBusy=> ,
    xError=> ,
    eError=> ,
    udiSdoAbort=> ,
    szDataWritten 2 => SDO_Length 2 );
```

LD：写伺服从站的控制字模式对象字典索引 16#6040，子索引 16#00。



5.3.8 IoDrvEtherCAT_Diag

EtherCAT 主站实例，实现 EtherCat 主站重启。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
IoDrvEtherCAT	EtherCAT 主站实例	FB		IoDrvEtherCAT_0(xRestart:= , xStopBus:= , xConfigFinished=> , xDistributedClockInSync=> , xError=> , xSyncInWindow=>);	IODrvEtherCAT

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xRestart	EtherCAT 主站重启	BOOL	TRUE-FALSE	FALSE	若为 TRUE（上升沿信号），则 EtherCAT 主站重启
xStopBus	EtherCAT 主站停止	BOOL	TRUE-FALSE	FALSE	停止 EtherCAT 主站

输出变量

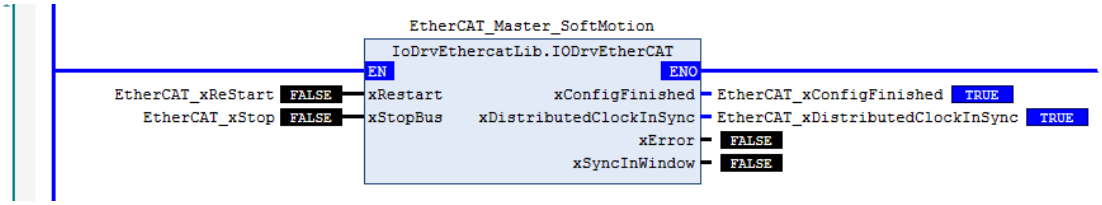
输出变量	名称	数据类型	有效范围	初始值	描述
xConfigFinished	EtherCAT 主站配置完成	BOOL	TRUE-FALSE	FALSE	若 EtherCAT 配置完成通讯正在运行，此值输出为 TRUE
xDistributedClockInSync	DC 同步信号	BOOL	TRUE-FALSE	FALSE	若同步模式为 DC 模式完成后，此时输出为 TRUE
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
xSyncInWindow	同步窗口	BOOL	TRUE-FALSE	FALSE	若同步性在同步窗口内，此时输出为 TRUE

程序示例

ST：若设备只有一个主站，实例化名称为 EtherCAT_Master _A，若 EtherCAT 配置完成，通讯正在运行且同步模式为 DC 模式完成后，则 xConfigFinished 及 xDistributedClockInSync 此值输出为 TRUE。

```
EtherCAT_Master_SoftMotion(
    xRestart FALSE := EtherCAT_xRestart FALSE,
    xStopBus FALSE := EtherCAT_xStop FALSE,
    xConfigFinished TRUE => EtherCAT_xConfigFinished TRUE,
    xDistributedClockInSync TRUE => EtherCAT_xDistributedClockInSync TRUE,
    xError=> ,
    xSyncInWindow=> );
```

LD：若设备只有一个主站，实例化名称为 EtherCAT_Master _A，若 EtherCAT 配置完成，通讯正在运行且同步模式为 DC 模式完成后，则 xConfigFinished 及 xDistributedClockInSync 此值输出为 TRUE。



5.3.9 ETCSlave

获取从站状态。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ETCSlave	EtherCAT 主站实例	FB		ETCSlave_0(xSetOperational:= , wState=>);	IODrvEtherCAT

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
xSetOperational	尝试将从站设置成 ETC_SLAVE_OPERATIONAL 状态	BOOL	TRUE-FALSE	FALSE	若输入为 TRUE（上升沿信号）则尝试将从站的状态机设置成 ETC_SLAVE_OPERATIONAL 状态

输出变量

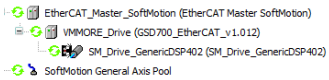
输出变量	名称	数据类型	有效范围	初始值	描述
wState	从站状态	ETC_SLAVE _STATE	-	-	读取从站状态

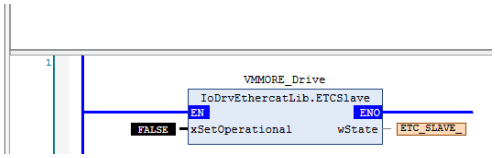
程序示例

ST：以 VMMORE_Drive 为例添加相关伺服从站的实例名称，定义 ETCSlave_State 为 ETC_SLAVE_STATE 枚举体变量。（详细状态请看注意事项）

```
VMMORE_Drive(
      xSetOperational:= ,
      wState ETC_SLAVE_ ▸ => ETCSlave_State ETC_SLAVE_ ▸ );
```

LD：以 VMMORE_Drive 为例添加相关伺服从站的实例名称，定义 ETCSlave_State 为 ETC_SLAVE_STATE 枚举体变量。（详细状态请看注意事项）





6 专用指令

6.1 系统指令

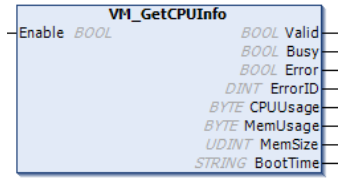
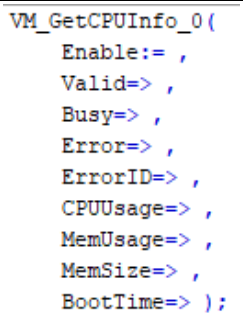
6.1.1 指令列表

指令类别	名称	FB/FC	功能
系统指令	VM_GetCPUInfo	FB	获取 CPU 信息
	VM_GetDateAndTime	FB	获取 PLC 的日期和时间
	VM_SetDateAndTime	FB	设置 PLC 的日期和时间
	VM_GetPLCVersion	FB	获取 PLC 版本号
	VM_GetNetInfo	FB	获取 PLC 网络信息
	VM_SetNetInfo	FB	设置 PLC 网络信息
	VM_GetIpAddress	FB	获取 PLC 的 IP 地址
	VM_SetIpAddress	FB	设置 PLC 的 IP 地址
	VM_GetPLCName	FB	获取 PLC 名称

6.1.2 VM_GetCPUInfo

获取 CPU 信息指令，CPU 信息包括 CPU 负载率、内存占用率、内存大小、开机时间。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_GetCPUInfo	获取 CPU 信息指令	FB			VM_SysLib


```
VM_GetCPUInfo_0(
    Enable:= ,
    Valid=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    CPUUsage=> ,
    MemUsage=> ,
    MemSize=> ,
    BootTime=> );
```

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	使能

输出变量

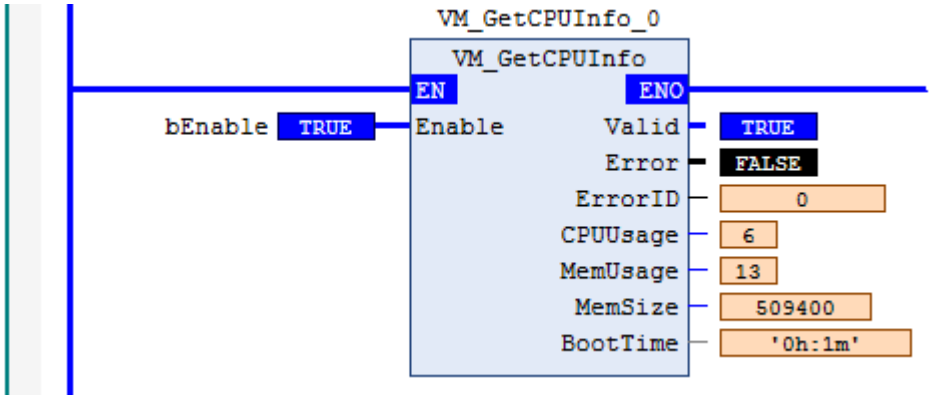
输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效标志	BOOL	TRUE-FALSE	-	输出值有效
Error	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_SYS_ERROR	遵照数据范围	-	错误信息
CPUUsage	CPU 负载率	BYTE	0-255	-	CPU 负载率
MemUsage	内存占用率	BYTE	0-255	-	内存占用率
MemSize	内存大小	UDINT	0~2^32-1	-	内存总大小，单位：Kb
BootTime	开机时间	STRING	-	-	开机时间

程序示例

ST:

```
VM_GetCPUInfo_0(
    Enable TRUE := bEnable TRUE,
    Valid TRUE => Valid TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0,
    CPUUsage 7 => CPUUsage 7,
    MemUsage 13 => MemUsage 13,
    MemSize 509400 => MemSize 509400,
    BootTime '0h:1m' => BootTime '0h:1m');
```

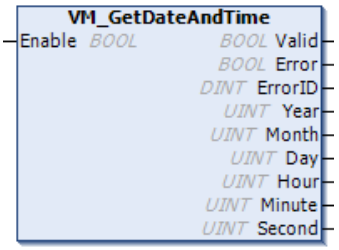
LD:


VM_GetCPUInfo_0
VM_GetCPUInfo
EN
ENO
bEnable
TRUE
Enable
Valid
TRUE
Error
FALSE
ErrorID
0
CPUUsage
6
MemUsage
13
MemSize
509400
BootTime
'0h:1m'

6.1.3 VM_GetDateAndTime

获取 PLC 的日期和时间指令。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_GetDateAndTime	获取 PLC 日期和时间指令	FB		VM_GetDateAndTime_0(Enable:= , Valid=> , Error=> , ErrorID=> , Year=> , Month=> , Day=> , Hour=> , Minute=> , Second=>);	VM_SysLib

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	使能

输出变量

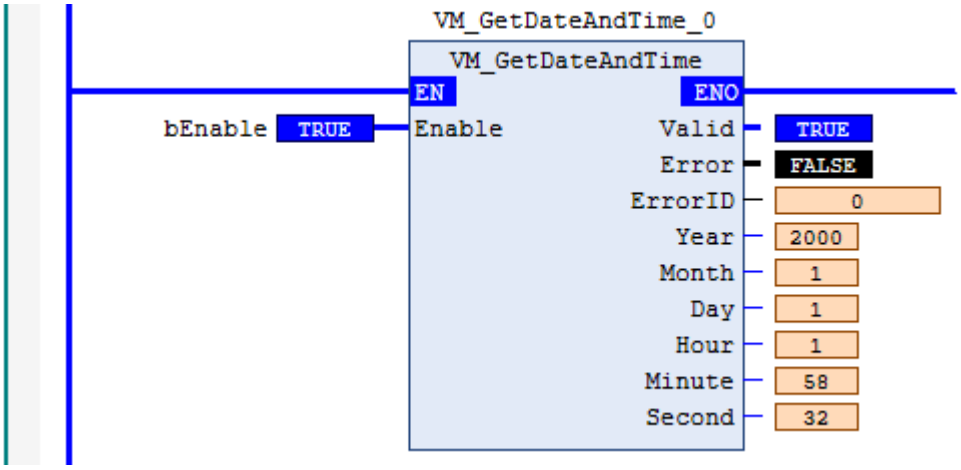
输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效标志	BOOL	TRUE-FALSE	-	输出值有效
Error	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_SYS_ERROR	遵照数据范围	-	错误信息
Year	年	UINT	0-65535	-	年
Month	月	UINT	1-12	-	月
Day	日	UINT	1-31	-	日
Hour	时	UINT	0-23	-	时
Minute	分	UINT	0-59		分
Second	秒	UINT	0-59		秒

程序示例

ST:

```
VM_GetDateAndTime_0(
    Enable TRUE := bEnable TRUE,
    Valid TRUE => Valid TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0,
    Year 2000 => Year 2000,
    Month 1 => Month 1,
    Day 1 => Day 1,
    Hour 1 => Hour 1,
    Minute 58 => Minute 58,
    Second 52 => Second 52 );
```

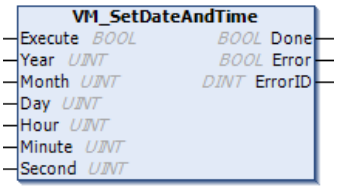
LD：


VM_GetDateAndTime_0
VM_GetDateAndTime
EN
ENO
bEnable
TRUE
Enable
Valid
TRUE
Error
FALSE
ErrorID
0
Year
2000
Month
1
Day
1
Hour
1
Minute
58
Second
32

6.1.4 VM_SetDateAndTime

设置 PLC 的日期和时间指令。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_SetDateAndTime	设置 PLC 日期和时间指令	FB		VM_SetDateAndTime_0(Execute:= , Year:= , Month:= , Day:= , Hour:= , Minute:= , Second:= , Done=> , Error=> , ErrorID=>);	VM_SysLib

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	使能	BOOL	TRUE-FALSE	FALSE	上升沿使能功能块
Year	年	UINT	0-65535	0	年
Month	月	UINT	1-12	0	月
Day	日	UINT	1-31	0	日
Hour	时	UINT	0-23	0	时
Minute	分	UINT	0-59	0	分
Second	秒	UINT	0-59	0	秒

输出变量

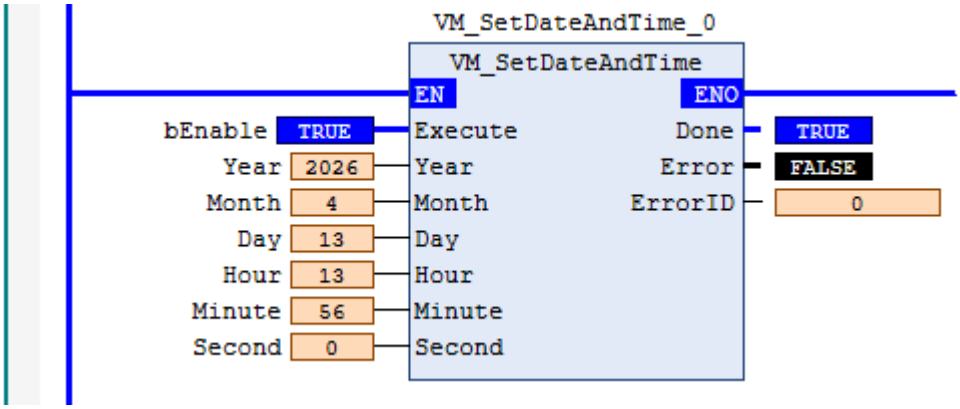
输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	TRUE-FALSE	-	完成设置日期和时间
Error	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_SYS_ERROR	遵照数据范围	-	错误信息

程序示例

ST:

```
VM_SetDateAndTime_0(
    Execute TRUE := bSetDataTime TRUE,
    Year 2026 := Year 2026,
    Month 4 := Month 4,
    Day 13 := Day 13,
    Hour 13 := Hour 13,
    Minute 56 := Minute 56,
    Second 0 := Second 0,
    Done TRUE => Done TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0);
```

LD:



6.1.5 VM_GetPLCVersion

获取 PLC 的版本号，包括软件版本、硬件版本和运行时版本。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_GetPLCVersion	获取 PLC 版本号指令	FB		VM_GetPLCVersion_0(Enable:= , Valid=> , Error=> , ErrorID=> , SoftWareVersion=> , HardWareVersion=> , RuntimeVersion=>);	VM_SysLib

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	使能功能块

输出变量

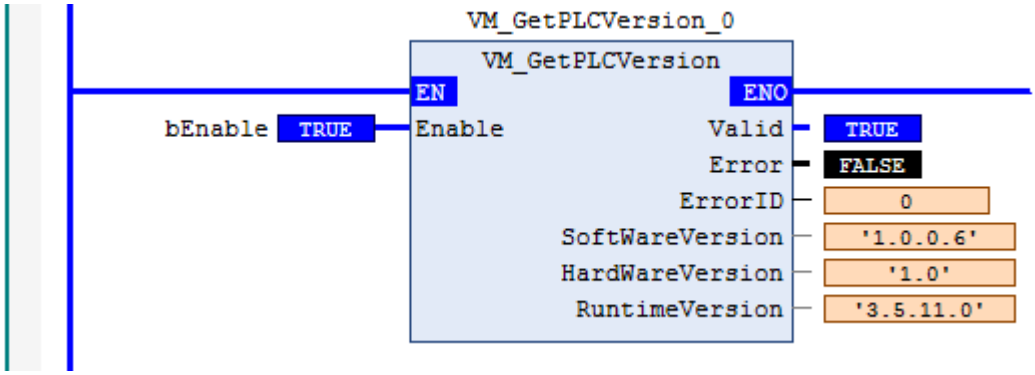
输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效标志	BOOL	TRUE-FALSE	-	输出值有效
Error	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_SYS_ERROR	遵照数据范围	-	错误信息
SoftWareVersion	软件版本号	STRING	-	-	软件版本号
HardWareVersion	硬件版本号	STRING	-	-	硬件版本号
RuntimeVersion	运行时版本号	STRING	-	-	运行时版本号

程序示例

ST:

```
VM_GetPLCVersion_0(
    Enable TRUE := bEnable TRUE,
    Valid TRUE => Valid TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0,
    SoftWareVersion '1.0.0.6' => SoftWareVersion '1.0.0.6',
    HardWareVersion '1.0' => HardWareVersion '1.0',
    RuntimeVersion '3.5.11.0' => RuntimeVersion '3.5.11.0');
```

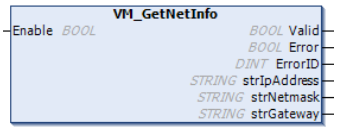
LD:


VM_GetPLCVersion_0
VM_GetPLCVersion
EN
ENO
bEnable
TRUE
Enable
Valid
TRUE
Error
FALSE
ErrorID
0
SoftWareVersion
'1.0.0.6'
HardWareVersion
'1.0'
RuntimeVersion
'3.5.11.0'

6.1.6 VM_GetNetInfo

获取 PLC 的 eth1(CN4 网口)网络信息，包括 IP 地址、子网掩码和网关。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_GetNetInfo	获取 PLC 网络信息指令	FB		VM_GetNetInfo_0(Enable:= , Valid=> , Error=> , ErrorID=> , strIpAddress=> , strNetmask=> , strGateway=>);	VM_SysLib

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	使能功能块

输出变量

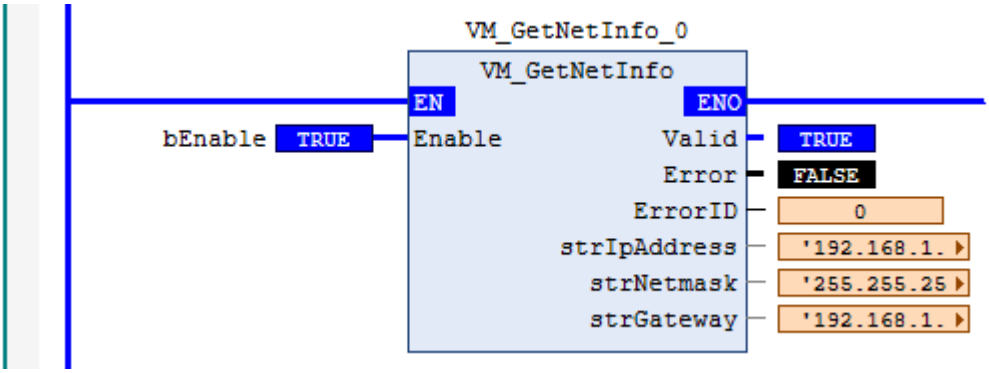
输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效标志	BOOL	TRUE-FALSE	-	输出值有效
Error	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_SYS_ERROR	遵照数据范围	-	错误信息
strIpAddress	IP 地址	STRING	-	-	IP 地址
strNetmask	子网掩码	STRING	-	-	子网掩码
strGateway	网关	STRING	-	-	网关

程序示例

ST:

```
VM_GetNetInfo_0(
    Enable TRUE := bEnable TRUE,
    Valid TRUE => Valid TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0,
    strIpAddress '192.168.1. => strIpAddress '192.168.1.,
    strNetmask '255.255.25 => strNetmask '255.255.25,
    strGateway '192.168.1. => strGateway '192.168.1. );
```

LD:


VM_GetNetInfo_0
VM_GetNetInfo
EN
ENO
bEnable
TRUE
Enable
Valid
TRUE
Error
FALSE
ErrorID
0
strIpAddress
'192.168.1.
strNetmask
'255.255.25
strGateway
'192.168.1.

6.1.7 VM_SetNetInfo

设置 PLC 的网络信息，包括 IP 地址、子网掩码和网关。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_SetNetInfo	设置 PLC 网络信息指令	FB		VM_SetNetInfo_0(Execute:= , strIpAddress:= , strNetmask:= , strGateway:= , Done=> , Error=> , ErrorID=>);	VM_SysLib

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	使能	BOOL	TRUE-FALSE	FALSE	上升沿使能功能块
strIpAddress	IP 地址	STRING	-	-	IP 地址
strNetmask	子网掩码	STRING	-	-	子网掩码
strGateway	网关	STRING	-	-	网关

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	TRUE-FALSE	-	完成设置日期和时间
Error	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_SYS_ERROR	遵照数据范围	-	错误信息

注意事项

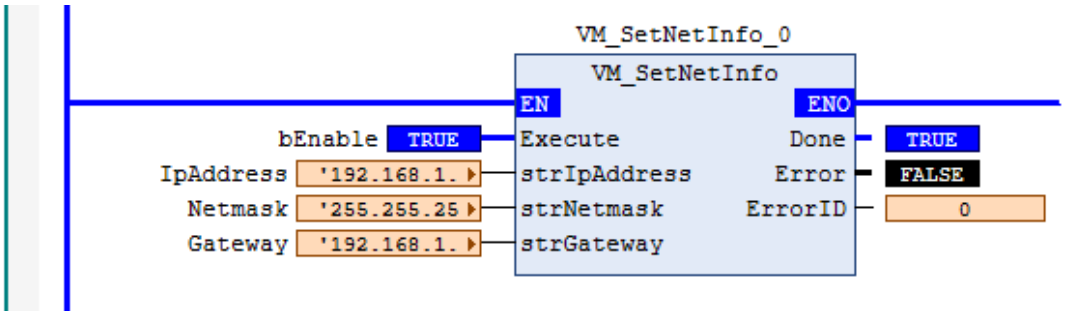
只能修改 eth1（CN4 网口）的网络信息，修改后需要断电重启 PLC 后才生效。

程序示例

ST:

```
VM_SetNetInfo_0(
    Execute TRUE := bSetNetInfo TRUE,
    strIpAddress '192.168.1. ▸ := IpAddress '192.168.1. ▸,
    strNetmask '255.255.25 ▸ := Netmask '255.255.25 ▸,
    strGateway '192.168.1. ▸ := Gateway '192.168.1. ▸,
    Done TRUE => Done TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0 );
```

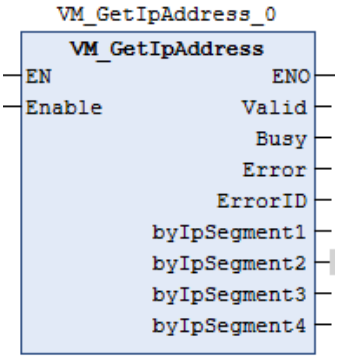
LD:


VM_SetNetInfo_0
VM_SetNetInfo
EN
ENO
bEnable
TRUE
Execute
Done
TRUE
IpAddress
'192.168.1.
strIpAddress
Error
FALSE
Netmask
'255.255.25
strNetmask
ErrorID
0
Gateway
'192.168.1.
strGateway

6.1.8 VM_GetIpAddress

获取 PLC 的 eth1(CN4 网口)的 IP 地址。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_GetIpAddress	获取 PLC 的 IP 地址	FB		VM_GetIpAddress_0(Enable:= , Valid=> , Busy=> , Error=> , ErrorID=> , byIpSegment1=> , byIpSegment2=> , byIpSegment3=> , byIpSegment4=>);	VM_SysLib

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	使能功能块

输出变量

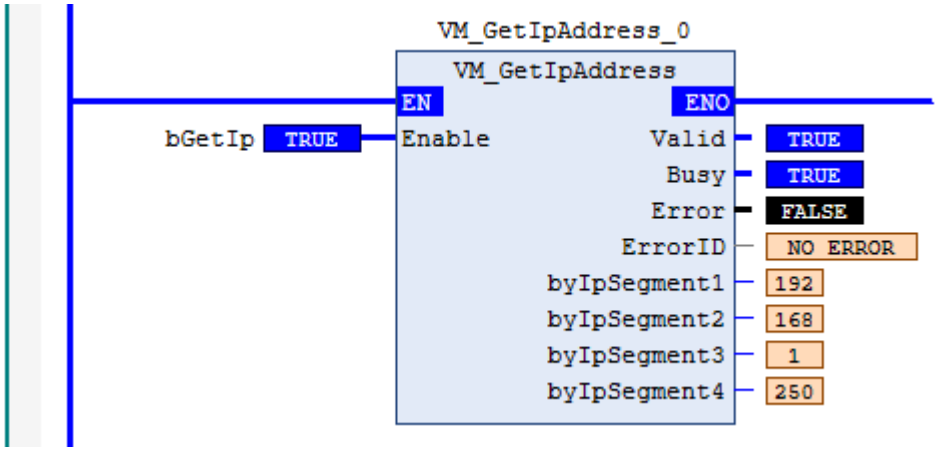
输出变量	名称	数据类型	有效范围	初始值	描述
Valid	有效标志	BOOL	TRUE-FALSE	-	输出值有效
Busy	忙碌标志	BOOL	TRUE-FALSE	-	功能块正在运行
Error	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_SYS_ERROR	遵照数据范围	-	错误信息
byIpSegment1	IP 地址第 1 段	BYTE	0-255	-	获取到的 IP 地址第 1 段
byIpSegment2	IP 地址第 2 段	BYTE	0-255	-	获取到的 IP 地址第 2 段
byIpSegment3	IP 地址第 3 段	BYTE	0-255	-	获取到的 IP 地址第 3 段
byIpSegment4	IP 地址第 4 段	BYTE	0-255	-	获取到的 IP 地址第 4 段

程序示例

ST:

```
VM_GetIpAddress_0(
    Enable TRUE := bGetIp TRUE,
    Valid=> ,
    Busy=> ,
    Error=> ,
    ErrorID=> ,
    byIpSegment1 192 => byIpSegment1 192,
    byIpSegment2 168 => byIpSegment2 168,
    byIpSegment3  1  => byIpSegment3  1 ,
    byIpSegment4 250 => byIpSegment4 250 );
```

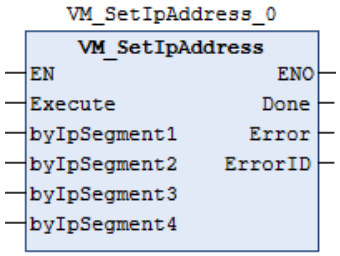
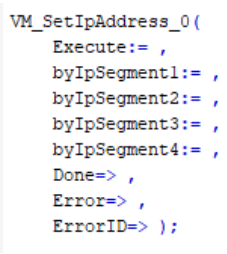
LD:



6.1.9 VM_SetIpAddress

设置 PLC 的 eth1(CN4 网口)的 IP 地址。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_SetIpAddress	设置 PLC 的 IP 地址	FB			VM_SysLib

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	使能	BOOL	TRUE-FALSE	FALSE	上升沿使能功能块
byIpSegment1	IP 地址第 1 段	BYTE	0-255	0	需要配置的 IP 地址第 1 段
byIpSegment2	IP 地址第 2 段	BYTE	0-255	0	需要配置的 IP 地址第 2 段
byIpSegment3	IP 地址第 3 段	BYTE	0-255	0	需要配置的 IP 地址第 3 段
byIpSegment4	IP 地址第 4 段	BYTE	0-255	0	需要配置的 IP 地址第 4 段

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	TRUE-FALSE	-	IP 地址设置完成标志
Error	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_SYS_ERROR	遵照数据范围	-	错误信息

注意事项

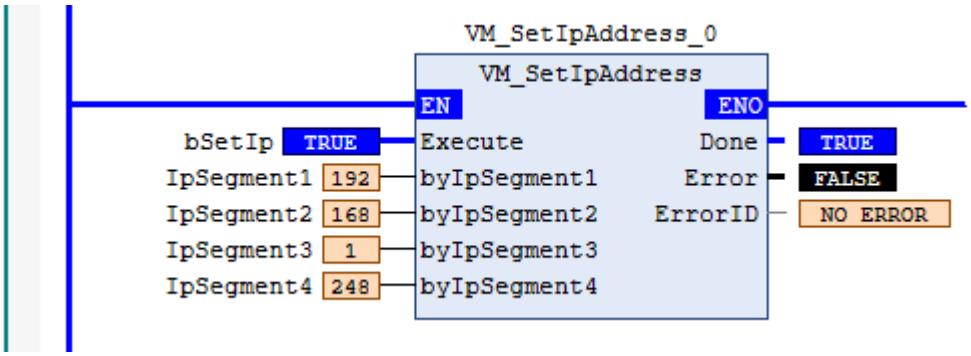
只能修改 eth1（CN4 网口）的 IP 地址，修改后需要断电重启 PLC 后才生效。

程序示例

ST:

```
VM_SetIpAddress_0(
    Execute TRUE := bSetIp TRUE,
    byIpSegment1 192 := IpSegment1 192,
    byIpSegment2 168 := IpSegment2 168,
    byIpSegment3 1 := IpSegment3 1,
    byIpSegment4 248 := IpSegment4 248,
    Done TRUE => Done TRUE,
    Error FALSE => Error FALSE,
    ErrorID NO_ERROR => ErrorID NO_ERROR );
```

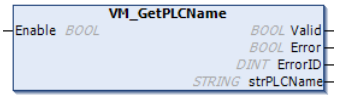
LD：



6.1.10 VM_GetPLCName

获取 PLC 的名称。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_GetPLCName	获取 PLC 名称指令	FB		VM_GetPLCName_0(Enable:= , Valid=> , Error=> , ErrorID=> , strPLCName=>);	VM_SysLib

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	使能功能块

输出变量

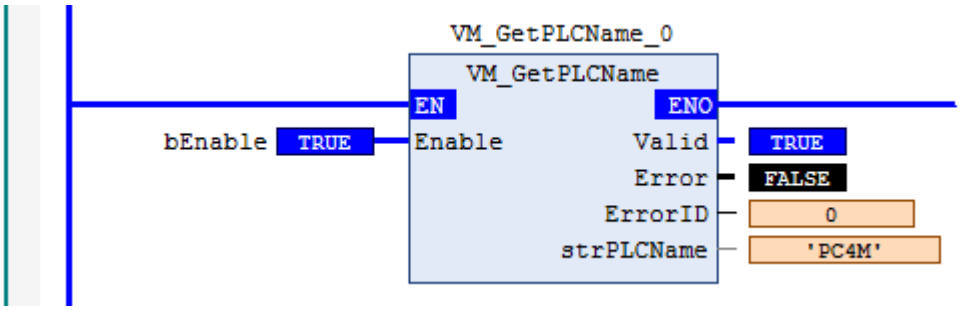
输出变量	名称	数据类型	有效范围	初始值	描述
Valid	完成标志	BOOL	TRUE-FALSE	-	输出值有效
Error	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_SYS_ERROR	遵照数据范围	-	错误信息
strPLCName	PLC 名称	STRING	-	-	输出 PLC 名称

程序示例

ST:

```
VM_GetPLCName_0(
    Enable TRUE := bEnable TRUE,
    Valid TRUE => Valid TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0,
    strPLCName 'PC4M' => strPLCName 'PC4M');
```

LD:



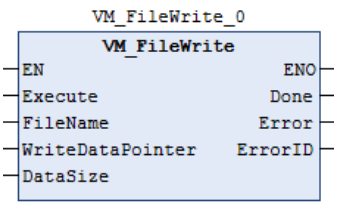
6.2 文件操作指令

6.2.1 指令列表

指令类别	名称	FB/FC	功能
文件操作指令	VM_FileWrite	FB	文件写入
	VM_FileWriteAppend	FB	文件追加写入
	VM_FileRead	FB	文件读取
	VM_FileCopy	FB	文件复制
	VM_FileRename	FB	文件重命名
	VM_FileDelete	FB	文件删除
	VM_CopyFromSDCard	FB	从 SD 卡拷贝文件
	VM_CopyToSDCard	FB	拷贝文件至 SD 卡
	VM_CopyFromUDisk	FB	从 U 盘拷贝文件
	VM_CopyToUDisk	FB	拷贝文件至 U 盘

6.2.2 VM_FileWrite

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_FileWrite	文件写入指令	FB		VM_FileWrite_0(Execute:= , FileName:= , WriteDataPointer:= , DataSize:= , Done=> , Error=> , ErrorID=>);	VM_FileManage

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE 时使能功能块，上升沿有效；
FileName	文件名	STRING	遵照数据类型	-	文件名
WriteDataPointer	写入文件内容的指针	POINTER TO BYTE	遵照数据类型	-	写入文件内的数据内容的地址
DataSize	数据大小	DWORD	遵照数据类型	0	写入文件大小，单位是字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	-	完成信号；True-写入文件完成；
Error	错误	BOOL	FALSE-TRUE	-	错误状态；FALSE-没有错误；
ErrorID	错误代码	VM_FILE_ERROR	遵照数据类型	-	错误代码

功能说明

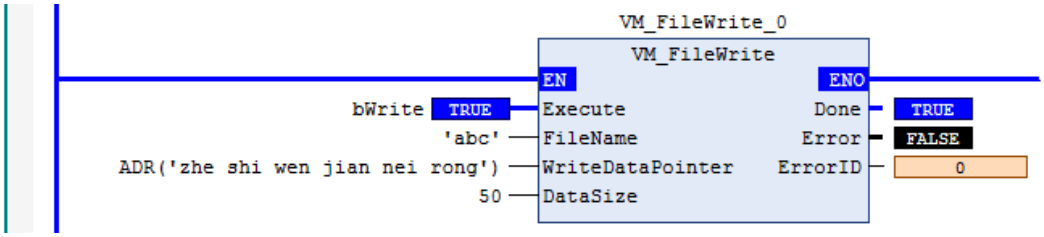
在控制器“UsrData”文件夹内创建新的文件（文件名称由 FileName 指定，文件大小由 DataSize 指定），在文件内写入数据内容（数据内容是 WriteDataPointer 指向的地址内储存的数据）。若该文件名已存在，旧的文件内容将会被覆盖。“UsrData”文件夹在文件窗口点击右上角“更新”按钮可以看见。

程序示例

ST： 当启动信号 bWrite 变为 True 时，新建文件（文件名称为“abc”）并在文件内写入数据内容(内容为“zhe shi xie ru nei rong”)。

```
VM_FileWrite_0(
    Execute:= bWrite,
    FileName:= 'abc',
    WriteDataPointer:= ADR('zhe shi xie ru nei rong'),
    DataSize:= 50,
    Done=> Done,
    Error=> Error,
    ErrorID=> ErrorID);
```

LD：当启动信号 bWrite 变为 True 时，新建文件（文件名称为“abc”）并在文件内写入数据内容(内容为“zhe shi xie ru nei rong”)。



注意事项

1）DataSize 是新建立文件的大小，需要比 WriteDataPointer 所指定的数据大小大，否则写入的数据可能丢失。

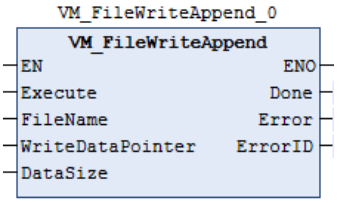
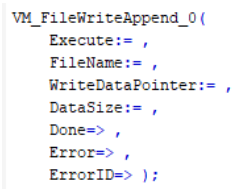
2）注意 PLC 不支持建立中文名字的文件，文件内容最好不包含中文，否则 VM_FileRead 指令读取到的可能无法识别。

3）WriteDataPointer 是写入文件内的数据内容的地址。

4）指令默认在控制器“UsrData”文件夹内创建新的文件，不支持在线仿真。

6.2.3 VM_FileWriteAppend

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_FileWriteAppend	文件追加写入指令	FB			VM_FileManage

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE 时使能功能块，上升沿触发；
FileName	文件名	STRING	-	-	待追加写入数据内容的文件名称
WriteDataPointer	写入数据内容的指针	POINTERTO BYTE	-	-	追加写入的数据内容的地址
DataSize	数据大小	DWORD	遵照数据类型	0	追加写入的数据大小，单位是字节

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	-	完成信号；True-文件追加写入完成；
Error	错误	BOOL	FALSE-TRUE	-	错误状态；FALSE-没有错误；
ErrorID	错误代码	VM_FILE_ERROR	遵照数据类型	-	错误代码

功能说明

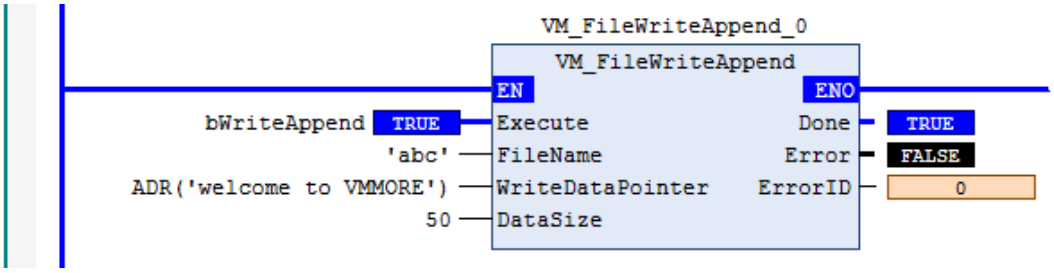
在控制器“UsrData”文件夹内，对 FileName 指定的文件以追加（扩大原文件大小）的方式写入 WriteDataPointer 指定的数据内容。

程序示例

ST：当启动信号 bWriteAppend 变为 TRUE 时，在指定文件（abc）中追加写入数据内容（welcome to VMMORE）

```
VM_FileWriteAppend_0(
    Execute TRUE := bWriteAppend TRUE,
    FileName 'abc' := 'abc',
    WriteDataPointer 16#B403315C := ADR('welcome to VMMORE'),
    DataSize 50 := 50,
    Done TRUE => Done TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0);
```

LD：当启动信号 bWriteAppend 变为 TRUE 时，在指定文件（abc）中追加写入数据内容（welcome to VMMORE）



注意事项

1）DataSize 是追加写入的数据大小，需要比 WriteDataPointer 所指定的数据大小大，否则追加写入的数据可能丢失。

2）Execute 上升沿触发，每次执行文件追加写入指令后，文件大小变为原文件大小加上 DataSize 的大小；追加写入的数据从增加的字节开始写入，可以用 VM_FileRead 指令读取（设置偏移值 OffSetPosition，从新增加的字节处开始读取）。

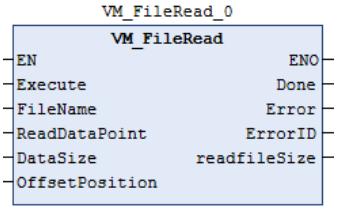
3）注意 PLC 不支持建立中文名字的文件，文件内容最好不包含中文，否则 VM_FileRead 指令读取到的可能无法识别。

4）WriteDataPointer 是追加写入的数据内容的地址。

5）该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

6.2.4 VM_FileRead

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_FileRead	文件读取指令	FB		VM_FileRead_0(Execute:= , FileName:= , ReadDataPoint:= , DataSize:= , OffsetPosition:= , Done=> , Error=> , ErrorID=> , readfileSize=>);	VM_FileManage

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE 时使能功能块， 上升沿触发；
FileName	文件名	STRING	遵照数据类型	-	要读取数据内容的文件名称
ReadDataPoint	数据接收指针	POINTERTO BYTE	遵照数据类型	-	将读取到的数据内容保存到指定的地址
DataSize	数据大小	DWORD	遵照数据类型	0	最大能够读取的数据大小
OffSetPosition	位置偏移	DWORD	遵照数据类型	0	从指定字节之后开始读取

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	-	完成信号；True-读取文件完成；
Error	错误	BOOL	FALSE-TRUE	-	错误状态；FALSE-没有错误；
ErrorID	错误代码	VM_FILE_ ERROR	遵照数据类型	-	错误代码
readfileSize	读取的文件大小	DWORD	遵照数据类型	-	实际读取到的数据大小

功能说明

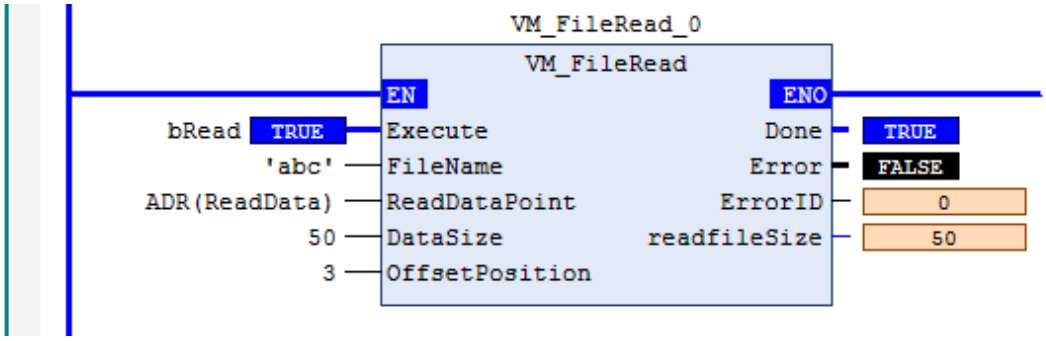
在控制器“UsrData”文件夹内读取 FileName 所指定名称的文件中的数据内容，读取的数据内容存储在 ReadDataPointer 所指定的地址中。

程序示例

ST：当启动信号 bRead 变为 TRUE 时，读取指定文件（abc）的数据内容保存到 ReadDataPoint 所指定的地址(ADR(ReadData))。

```
VM_FileRead_0(
    Execute TRUE := bRead TRUE,
    FileName 'abc' := 'abc',
    ReadDataPoint 16#B6274E40 := ADR(ReadData),
    DataSize 50 := 50,
    OffsetPosition 3 := 3,
    Done TRUE => Done TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0,
    readfileSize 50 => readfileSize 50);
```

LD：当启动信号 ghi 变为 True 时， 读取指定文件（xin jian wen jian ming） 的数据内容保存到 ReadDataPoint 所指定的地址(ADR(wen_jian_nei_rong))。



注意事项

1）DataSize 是最大能读取到的数据大小（要比被读取文件大，否则数据可能读取不完整），readfileSize 是实际读取到的数据大小（取决于文件大小与 DataSize）。

2）OffSetPosition 是从指定的字节后开始读取数据内容。

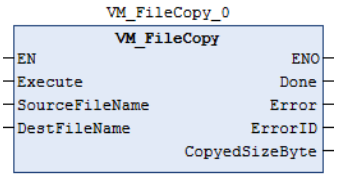
3）读取到的文件数据内容存储在 ReadDataPointer 所指定的地址中。

4）注意 PLC 不支持建立中文名字的文件，文件内容最好不包含中文，否则 VM_FileRead 指令读取到的可能无法识别。

5）该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

6.2.5 VM_FileCopy

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_FileCopy	文件拷贝指令	FB		VM_FileCopy_0(Execute:= , SourceFileName:= , DestFileName:= , Done=> , Error=> , ErrorID=> , CopyedSizeByte=>);	VM_FileManage

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE 时使能功能块，上升沿触发；
SourceFileName	源文件名	STRING	-	-	需要拷贝的源文件名称
DestFileName	目标文件名	STRING	-	-	拷贝后生成的新文件名称

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	-	完成信号；True-拷贝文件完成；
Error	错误	BOOL	FALSE-TRUE	-	错误状态；FALSE-没有错误；
ErrorID	错误代码	VM_FILE_ERROR	遵照数据类型	-	错误代码
CopyedSizeByte	拷贝的字节数	DWORD	遵照数据类型	-	拷贝后生成的新文件大小

功能说明

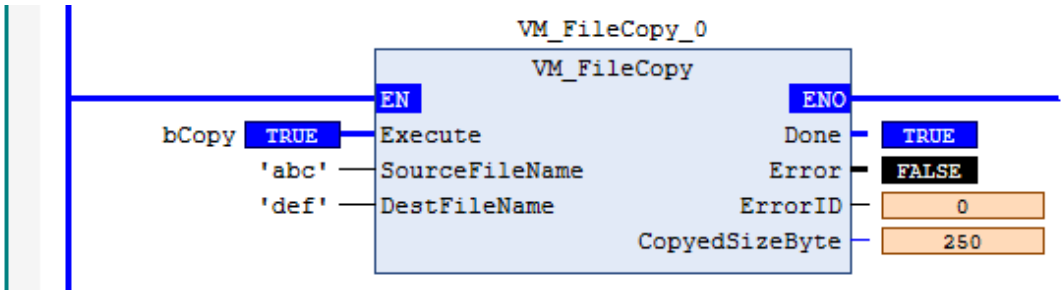
在控制器“UsrData”文件夹内将 SourceFileName 指定的源文件拷贝到 DestFileName 所指定的目标文件中，拷贝的字节数 CopyedSizeBytes 是源文件大小（拷贝的是整个源文件， 拷贝的字节数就是源文件大小）。

程序示例

ST：当启动信号 bCopy 变为 TRUE 时，将指定的源文件（abc）拷贝到指定的目标文件（def）中。

```
VM_FileCopy_0(
    Execute TRUE := bCopy TRUE,
    SourceFileName 'abc' := 'abc',
    DestFileName 'def' := 'def',
    Done TRUE => Done TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0,
    CopyedSizeByte 250 => CopyedSizeByte 250);
```

LD：当启动信号 bCopy 变为 TRUE 时，将指定的源文件（abc）拷贝到指定的目标文件（def）中。

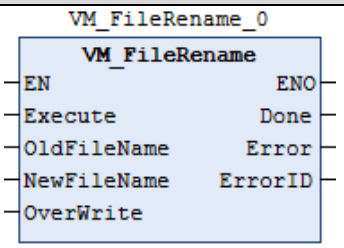


注意事项

1）VM_FileCopy 指令实际上是复制整个源文件，另保存为另一个名称的文件。
2）如果源文件在控制器“UsrData”文件夹内不存在，运行 VM_FileCopy 指令会报错。
3）如果目标文件名字在控制器“UsrData”文件夹内已经存在，则会覆盖旧文件内容。。
4）注意 PLC 不支持建立中文名字的文件。
5）该指令在控制器“UsrData”文件夹内进行操作，不支持在线仿真。

6.2.6 VM_FileRename

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_FileRename	文件拷贝指令	FB		VM_FileRename_0(Execute:= , OldFileName:= , NewFileName:= , OverWrite:= , Done=> , Error=> , ErrorID=>);	VM_FileManage

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号; TRUE 时使能功能块，上升沿触发;
OldFileName	原文件名	STRING	遵照数据类型	-	原文件名称
NewFileName	新文件名	STRING	遵照数据类型	-	新文件名称
OverWrite	允许覆盖	BOOL	FALSE-TRUE	FALSE	TRUE：允许覆盖 FALSE：禁止覆盖

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	-	完成信号; True-文件重命名完成;
Error	错误	BOOL	FALSE-TRUE	-	错误状态; FALSE-没有错误;
ErrorID	错误代码	VM_FILE_ERROR	遵照数据类型	-	错误代码

功能说明

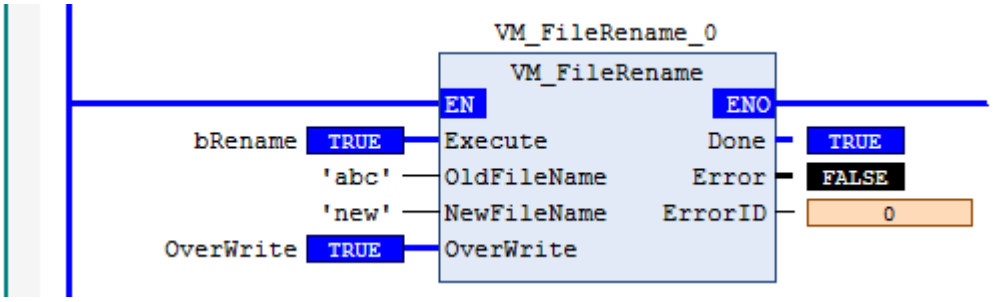
在控制器“UsrData”文件夹内对OldFileName所指定的文件名称进行重命名操作，新文件名称由NewFileName指定。

程序示例

ST：当启动信号 bRename 变为 TRUE 时，将指定的原文件名称（abc）改成 NewFileName 指定的新名称（new）。

```
VM_FileRename_0(
    Execute TRUE := bRename TRUE,
    OldFileName 'asd.txt' := 'asd.txt',
    NewFileName 'new' := 'new',
    OverWrite TRUE := OverWrite TRUE,
    Done TRUE => Done TRUE,
    Error=> ,
    ErrorID 0 => ErrorID 0);
```

LD：当启动信号 bRename 变为 TRUE 时，将指定的原文件名称（abc）改成 NewFileName 指定的新名称（new）。



注意事项

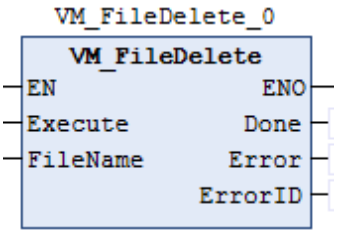
1）将原文件名“OldFileName”变更为文件名“NewFileName”，若”UsrData”目录中不存在与“NewFileName”同名的文件时，将重命名文件；若”UsrData”目录中存在与“NewFileName”同名的文件，根据输入“OverWrite”的值的不同，处理如下。

“OverWrite”的值	操作
TRUE	覆盖与“NewFileName”同名的文件
FALSE	不覆盖与“NewFileName”同名的文件，发生异常

2）如果 OldFileName 所指定的原文件在控制器“UsrData”文件夹内不存在，运行 VM_FileRename 指令会报错。

6.2.7 VM_FileDelete

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_FileDelete	文件拷贝指令	FB		VM_FileDelete_0(Execute:= , FileName:= , Done=> , Error=> , ErrorID=>);	VM_FileManage

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	FALSE-TRUE	FALSE	启动信号；TRUE 时使能功能块，上升沿触发；
FileName	文件名	STRING	遵照数据类型	-	需要删除的文件名称

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	-	完成信号；True-文件删除完成；
Error	错误	BOOL	FALSE-TRUE	-	错误状态；FALSE-没有错误；
ErrorID	错误代码	VM_FILE_ERROR	遵照数据类型	-	错误代码

功能说明

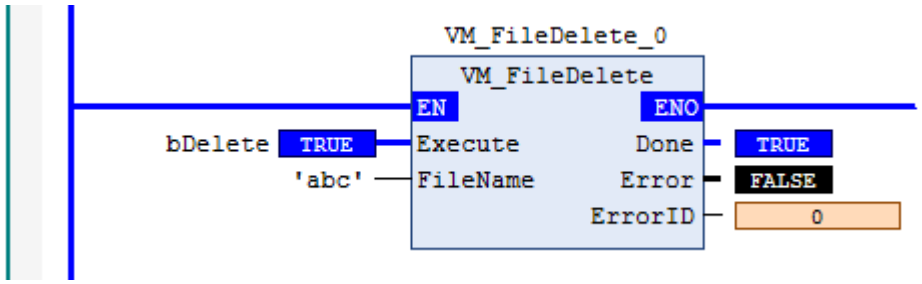
在控制器“UsrData”文件夹内删除 FileName 所指定名称的文件。

程序示例

ST：当启动信号 bDelete 变为 TRUE 时，将指定名称的文件（abc）删除。

```
VM_FileDelete_0(
    Execute TRUE := bDelete TRUE,
    FileName 'abc' := 'abc',
    Done TRUE => Done TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0);
```

LD：当启动信号 bDelete 变为 TRUE 时，将指定名称的文件（abc）删除。

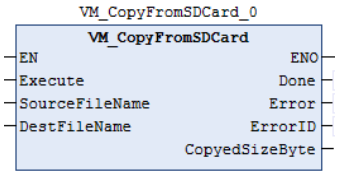


注意事项

如果 FileName 所指定名称的文件在控制器“UsrData”文件夹内不存在，运行 VM_FileDelete 指令会报错。

6.2.8 VM_CopyFromSDCard

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_CopyFromSDCard	从 SD 卡拷贝文件到本地	FB		VM_CopyFromSDCard_0(Execute:= , SourceFileName:= , DestFileName:= , Done=> , Error=> , ErrorID=> , CopyedSizeByte=>);	VM_FileManage

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块
SourceFileName	源文件名	STRING	-	-	SD 卡中需要拷贝的源文件名
DestFileName	目标文件名	STRING	-	-	拷贝后在控制器"UsrData"文件夹内生成的新文件名

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	-	完成信号；True-文件拷贝完成；
Error	错误	BOOL	FALSE-TRUE	-	错误状态；FALSE-没有错误；
ErrorID	错误代码	VM_FILE_ERROR	遵照数据类型	-	错误代码
CopyedSizeByte	拷贝的字节长度	DWORD	遵照数据类型	-	拷贝的字节长度

功能说明

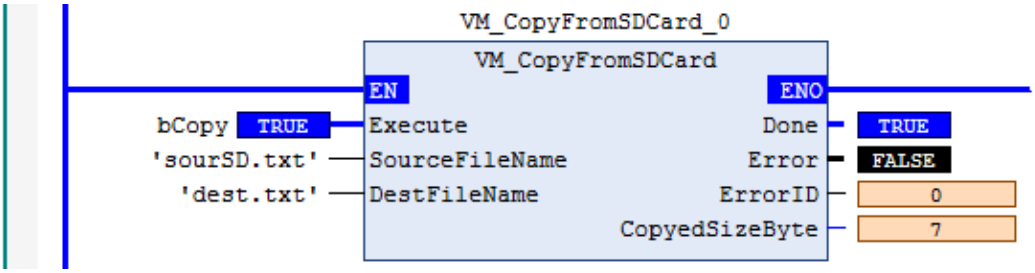
从 SD 卡根目录拷贝文件（文件名称由 SourceFileName 指定）至控制器"UsrData"文件夹内（拷贝生成的新文件名称由 DestFileName 指定）。

程序示例

ST：当启动信号 bCopy 变为 TRUE 时，从 SD 卡根目录拷贝文件（sourSD.txt）至控制器"UsrData"文件夹内（新生成文件 dest.txt）。

```
VM_CopyFromSDCard_0(
    Execute TRUE := bCopy TRUE,
    SourceFileName 'sourSD.txt' := 'sourSD.txt',
    DestFileName 'dest.txt' := 'dest.txt',
    Done TRUE => Done TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0,
    CopyedSizeByte 7 => CopyedSizeByte 7);
```

LD：当启动信号 bCopy 变为 TRUE 时，从 SD 卡根目录拷贝文件（sourSD.txt）至控制器“UsrData”文件夹内（新生成文件 dest.txt）。



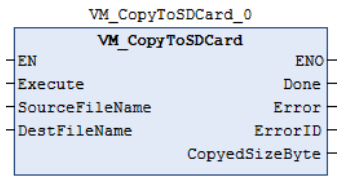
注意事项

1）如果 SourceFileName 指定的源文件在 SD 卡根目录中不存在，运行 VM_CopyFromSDCard 后将会报错。

2）如果 DestFileName 指定的目标文件在控制器“UsrData”文件夹内已存在，运行 VM_CopyFromSDCard 后，旧文件的数据内容会被覆盖。

6.2.9 VM_CopyToSDCard

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_CopyToSDCard	将本地文件拷贝至 SD 卡	FB		VM_CopyToSDCard_0(Execute:= , SourceFileName:= , DestFileName:= , Done=> , Error=> , ErrorID=> , CopyedSizeByte=>);	VM_FileManage

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块
SourceFileName	源文件名	STRING	-	-	控制器"UsrData"文件夹内需要拷贝的源文件名
DestFileName	目标文件名	STRING	-	-	拷贝后在 SD 卡生成的新文件名

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	-	完成信号；True-文件拷贝完成；
Error	错误	BOOL	FALSE-TRUE	-	错误状态；FALSE-没有错误；
ErrorID	错误代码	VM_FILE_ERROR	遵照数据类型	-	错误代码
CopyedSizeByte	拷贝的字节长度	DWORD	遵照数据类型	-	拷贝的字节长度

功能说明

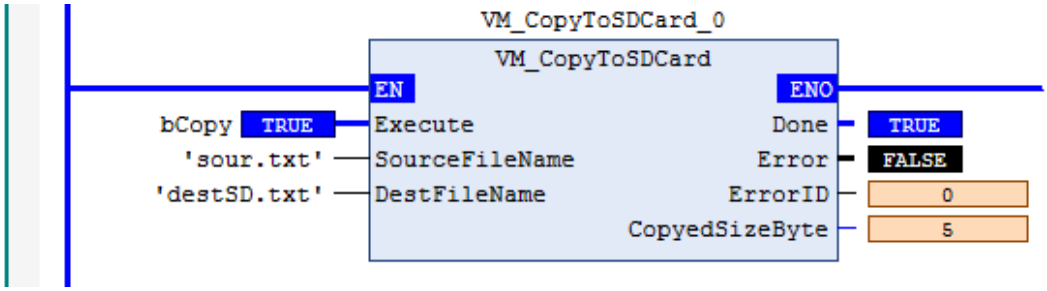
从控制器"UsrData"文件夹内拷贝文件（文件名称由 SourceFileName 指定）至 SD 卡根目录（拷贝生成的新文件名称由 DestFileName 指定）。

程序示例

ST：当启动信号 bCopy 变为 TRUE 时，从控制器"UsrData"文件夹内拷贝文件（sour.txt）至 SD 卡根目录（新生成文件 destSD.txt）。

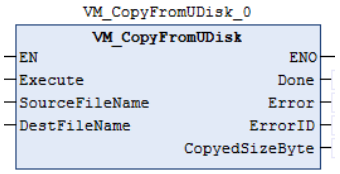
```
VM_CopyToSDCard_0(
    Execute TRUE := bCopy TRUE,
    SourceFileName 'sour.txt' := 'sour.txt',
    DestFileName 'destSD.txt' := 'destSD.txt',
    Done TRUE => Done TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0,
    CopyedSizeByte 5 => CopyedSizeByte 5);
```

LD：当启动信号 bCopy 变为 TRUE 时，从控制器“UsrData”文件夹内拷贝文件（sour.txt）至 SD 卡根目录（新生成文件 destSD.txt）。



6.2.10 VM_CopyFromUDisk

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_CopyFromUDisk	从 U 盘拷贝文件至本地	FB		VM_CopyFromUDisk_0(Execute:= , SourceFileName:= , DestFileName:= , Done=> , Error=> , ErrorID=> , CopyedSizeByte=>);	VM_FileManage

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块
SourceFileName	源文件名	STRING	-	-	U 盘中需要拷贝的源文件名
DestFileName	目标文件名	STRING	-	-	拷贝后在控制器"UsrData"文件夹内生成的新文件名

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	-	完成信号；True-文件拷贝完成；
Error	错误	BOOL	FALSE-TRUE	-	错误状态；FALSE-没有错误；
ErrorID	错误代码	VM_FILE_ERROR	遵照数据类型	-	错误代码
CopyedSizeByte	拷贝的字节长度	DWORD	遵照数据类型	-	拷贝的字节长度

功能说明

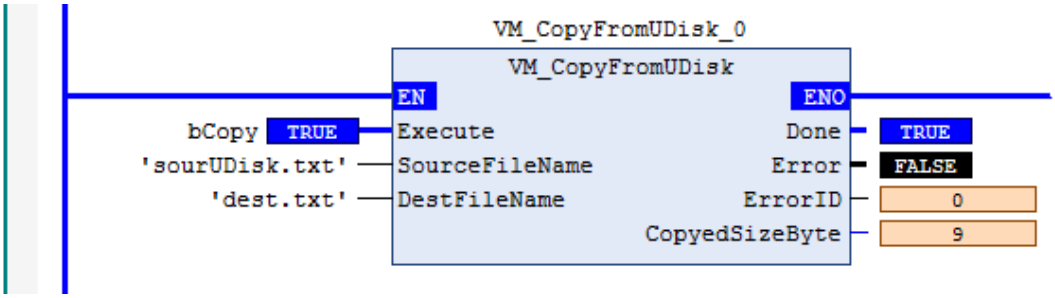
从 U 盘根目录拷贝文件（文件名称由 SourceFileName 指定）至控制器"UsrData"文件夹内（拷贝生成的新文件名称由 DestFileName 指定）。

程序示例

ST：当启动信号 bCopy 变为 TRUE 时，从 U 盘根目录拷贝文件（sourUDisk.txt）至控制器"UsrData"文件夹内（新生成文件 dest.txt）。

```
VM_CopyFromUDisk_0(
    Execute TRUE := bCopy TRUE,
    SourceFileName 'sourUDisk. ▸ := 'sourUDisk.txt',
    DestFileName 'dest.txt' := 'dest.txt',
    Done TRUE => Done TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0,
    CopyedSizeByte 9 => CopyedSizeByte 9 );
```

LD：当启动信号 bCopy 变为 TRUE 时，从 U 盘根目录拷贝文件（sourUDisk.txt）至控制器“UsrData”文件夹内（新生成文件 dest.txt）。



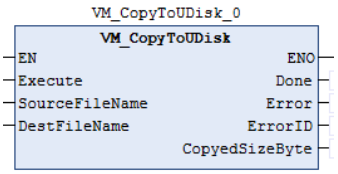
注意事项

1）如果 SourceFileName 指定的源文件在 U 盘根目录中不存在，运行 VM_CopyFromUDisk 后将会报错。

2）如果 DestFileName 指定的目标文件在控制器“UsrData”文件夹内已存在，运行 VM_CopyFromUDisk 后，旧文件的数据内容会被覆盖。

6.2.11 VM_CopyToUDisk

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
VM_CopyToUDisk	拷贝文件至 U 盘	FB		VM_CopyToUDisk_0(Execute:= , SourceFileName:= , DestFileName:= , Done=> , Error=> , ErrorID=> , CopyedSizeByte=>);	VM_FileManage

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Execute	启动	BOOL	TRUE-FALSE	FALSE	上升沿有效，状态从 FALSE 切换到 TRUE，执行功能块
SourceFileName	源文件名	STRING	-	-	控制器"UsrData"文件夹内需要拷贝的源文件名
DestFileName	目标文件名	STRING	-	-	拷贝后在 U 盘生成的新文件名

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	返回值	BOOL	FALSE-TRUE	-	完成信号；True-文件拷贝完成；
Error	错误	BOOL	FALSE-TRUE	-	错误状态；FALSE-没有错误；
ErrorID	错误代码	VM_FILE_ERROR	遵照数据类型	-	错误代码
CopyedSizeByte	拷贝的字节长度	DWORD	遵照数据类型	-	拷贝的字节长度

功能说明

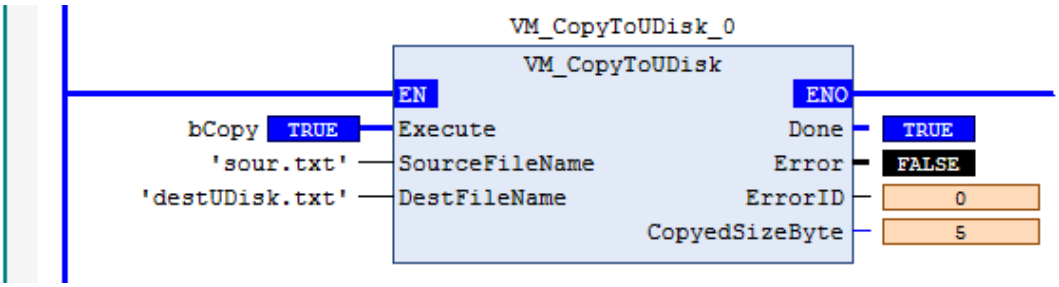
从控制器"UsrData"文件夹内拷贝文件（文件名称由 SourceFileName 指定）至 U 盘根目录（拷贝生成的新文件名称由 DestFileName 指定）。

程序示例

ST：当启动信号 bCopy 变为 TRUE 时，从控制器"UsrData"文件夹内拷贝文件（sour.txt）至 U 盘根目录（新生成文件 destUDisk.txt）。

```
VM_CopyToUDisk_0(
    Execute TRUE := bCopy TRUE,
    SourceFileName 'sour.txt' := 'sour.txt',
    DestFileName 'destUDisk. := 'destUDisk.txt',
    Done TRUE => Done TRUE,
    Error FALSE => Error FALSE,
    ErrorID 0 => ErrorID 0,
    CopyedSizeByte 5 => CopyedSizeByte 5 );
```

LD：当启动信号 bCopy 变为 TRUE 时，从控制器“UsrData”文件夹内拷贝文件（sour.txt）至 U 盘根目录（新生成文件 destUDisk.txt）。



注意事项

1）如果 SourceFileName 指定的源文件在控制器“UsrData”文件夹内不存在，运行 VM_CopyToUDisk 后将会报错。

2）如果 DestFileName 指定的目标文件在 U 盘根目录内已存在，运行 VM_CopyToUDisk 后，旧文件的数据内容会被覆盖。

6.3 Modbus 通信指令

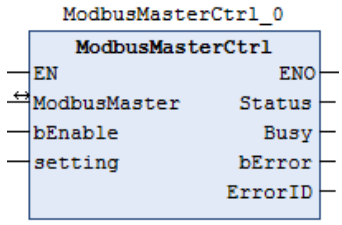
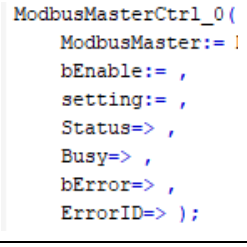
6.3.1 指令列表

指令类别	名称	FB/FC	功能
Modbus 通信指令	ModbusMasterCtrl	FB	Modbus 主站控制器指令
	ModbusMasterRequest	FB	Modbus 主站请求指令
	ModbusMasterStatus	FB	Modbus 主站状态指令
	ModbusSlaveCtrl	FB	Modbus 从站控制器指令

6.3.2 ModbusMasterCtrl

Modbus 主站控制器指令。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ModbusMasterCtrl	Modbus 主站控制器指令	FB			VM_ModbusMaster

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
ModbusMaster	Modbus 主站实例	MODBUS_MASTER	-	-	Modbus 主站实例
bEnable	使能	BOOL	TRUE-FALSE	FALSE	电平使能
setting	参数设置	COMM_SETTING	-	-	参数设置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Status	状态	BOOL	TRUE-FALSE	-	状态
Busy	忙碌	BOOL	TRUE-FALSE	FALSE	功能块正在运行中
bError	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_Common.ERROR_NO	-	-	错误信息

注意事项

使用 ModbusMasterCtrl 指令同时需要安装 VM_Common 库。

程序示例

ST： 如果进行 Modbus TCP 通信，则输入变量 setting 中的 commType 必须设置为 VM_Common.COMM_TYPE.COMM_TCP，具体参数设置和实例如下

表达式	类型	值	准备值	地址	注释
ModbusMasterCtrl_0	VM_ModbusMaster....				
Master	VM_ModbusMaster....				
Master	UDINT	5078392			主站实例
ACK	UDINT	2896997548			参数检验标志
setting	VM_ModbusMaster....				
commType	COMM_TYPE	COMM_TCP			通讯类型，默认 TCP
serial	SERIAL_COM	COM1			串口号，通信类型选择 RTU 有效
baudRate	BAUD_RATE	BAUD_RATE_9600			波特率，通信类型选择 RTU 有效
parity	PARITY	EVEN			校验，默认偶校验，通信类型选择 RTU 有效
stopBit	STOP_BIT	ONE_BIT			停止位，默认 1位，通信类型选择 RTU 有效
ScanRate	UINT	100			如果 ModbusRequest 使用周期模式，此处定义扫描周期 默认 1s
ResponseTime	UDINT	400			ModbusMaster 主站超时时间，单位 ms，默认 1s
frame_byte_timeout	UINT	100			单位 ms ,modbus 帧和帧之间的分辨时间,默认前后 100ms 内的字节流算一帧
retryCount	UINT	2			超时后,重发次数，默认 3次

```
ModbusMasterCtrl_0(
    ModbusMaster:= Master,
    bEnable TRUE := bMasterCtrl TRUE ,
    setting:= setting,
    Status TRUE => Status TRUE ,
    Busy TRUE => Busy TRUE ,
    bError FALSE => bError FALSE ,
    ErrorID  NO_ERROR  => ErrorID  NO_ERROR  );
```

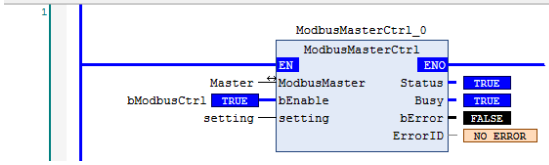
如果进行 Modbus RTU 通信，则输入变量 setting 中的 commType 必须设置为 VM_Common.COMM_TYPE.COMM_RTU，具体参数设置和实例如下

表达式	类型	值	准备值	地址	注释
ModbusMasterCtrl_0	VM_ModbusMaster....				
Master	VM_ModbusMaster....				
Master	UDINT	4104130168			主站实例
ACK	UDINT	2896997548			参数检验标志
setting	VM_ModbusMaster....				
commType	COMM_TYPE	COMM_RTU			通讯类型，默认TCP
serial	SERIAL_COM	COM1			串口号，通信类型选择RTU 有效
baudRate	BAUD_RATE	BAUD_RATE_9600			波特率，通信类型选择RTU 有效
parity	PARITY	EVEN			校验，默认偶校验，通信类型选择RTU 有效
stopBit	STOP_BIT	ONE_BIT			停止位，默认1位，通信类型选择RTU 有效
ScanRate	UINT	100			如果ModbusRequest 使用周期模式，此处定义扫描周期 默认 1s
ResponseTime	UDINT	400			ModbusMaster 主站超时时间，单位ms，默认 1s
frame_byte_timeout	UINT	100			单位 ms,modbus 帧和帧之间的分辨时间,默认前后100ms 内的字节流算一帧
retryCount	UINT	2			超时后,重发次数，默认3次

```
ModbusMasterCtrl_0(
    ModbusMaster:= Master,
    bEnable:= bMasterCtrl,
    setting:= setting,
    Status=> Status,
    Busy=> Busy,
    bError=> bError,
    ErrorID=> ErrorID );
```

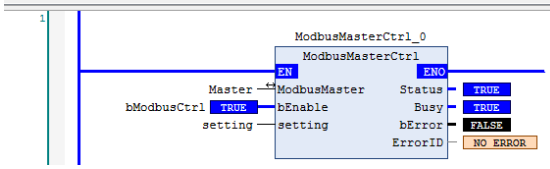
LD：如果需要进行 Modbus TCP 通信，则输入变量 setting 中的 commType 必须设置为 VM_Common.COMM_TYPE.COMM_TCP，具体参数设置和实例如下

表达式	类型	值	准备值	地址	注释
ModbusMasterCtrl_0	VM_ModbusMaster....				
Master	VM_ModbusMaster....				
Master	UDINT	4104130168			主站实例
ACK	UDINT	2896997548			参数检验标志
setting	VM_ModbusMaster....				
commType	COMM_TYPE	COMM_TCP			通讯类型，默认TCP
serial	SERIAL_COM	COM1			串口号，通信类型选择RTU 有效
baudRate	BAUD_RATE	BAUD_RATE_9600			波特率，通信类型选择RTU 有效
parity	PARITY	EVEN			校验，默认偶校验，通信类型选择RTU 有效
stopBit	STOP_BIT	ONE_BIT			停止位，默认1位，通信类型选择RTU 有效
ScanRate	UINT	100			如果ModbusRequest 使用周期模式，此处定义扫描周期 默认 1s
ResponseTime	UDINT	400			ModbusMaster 主站超时时间，单位ms，默认 1s
frame_byte_timeout	UINT	100			单位 ms,modbus 帧和帧之间的分辨时间,默认前后100ms 内的字节流算一帧
retryCount	UINT	2			超时后,重发次数，默认3次



如果进行 Modbus RTU 通信，则输入变量 setting 中的 commType 必须设置为 VM_Common.COMM_TYPE.COMM_RTU，具体参数设置和实例如下

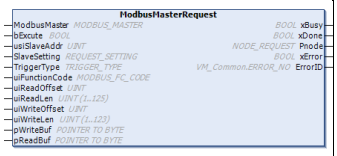
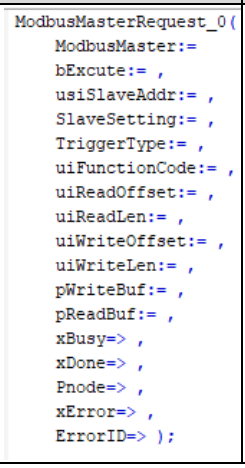
表达式	类型	值	准备值	地址	注释
ModbusMasterCtrl_0	VM_ModbusMaster....				
Master	VM_ModbusMaster....				
Master	UDINT	5077976			主站实例
ACK	UDINT	2896997548			参数检验标志
setting	VM_ModbusMaster....				
commType	COMM_TYPE	COMM_RTU			通讯类型，默认TCP
serial	SERIAL_COM	COM1			串口号，通信类型选择RTU 有效
baudRate	BAUD_RATE	BAUD_RATE_9600			波特率，通信类型选择RTU 有效
parity	PARITY	EVEN			校验，默认偶校验，通信类型选择RTU 有效
stopBit	STOP_BIT	ONE_BIT			停止位，默认1位，通信类型选择RTU 有效
ScanRate	UINT	100			如果ModbusRequest 使用周期模式，此处定义扫描周期 默认 1s
ResponseTime	UDINT	400			ModbusMaster 主站超时时间，单位ms，默认 1s
frame_byte_timeout	UINT	100			单位 ms,modbus 帧和帧之间的分辨时间,默认前后100ms 内的字节流算一帧
retryCount	UINT	2			超时后,重发次数，默认3次



6.3.3 ModbusMasterRequest

Modbus 主站请求指令。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ModbusMasterRequest	Modbus 主站请求指令	FB			VM_ModbusMaster


```
ModbusMasterRequest_0(
    ModbusMaster:=
    bExcute:= ,
    usiSlaveAddr:= ,
    SlaveSetting:= ,
    TriggerType:= ,
    uiFunctionCode:= ,
    uiReadOffset:= ,
    uiReadLen:= ,
    uiWriteOffset:= ,
    uiWriteLen:= ,
    pWriteBuf:= ,
    pReadBuf:= ,
    xBusy=> ,
    xDone=> ,
    Pnode=> ,
    xError=> ,
    ErrorID=> );
```

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
ModbusMaster	Modbus 主站实例	MODBUS_MASTER	-	-	Modbus 主站实例
bExcute	使能	BOOL	TRUE-FALSE	FALSE	上升沿使能
usiSlaveAddr	从站地址	UINT	0-65535	0	从站地址
SlaveSetting	从站设置	REQUEST_SETTING	-	-	从站设置，仅 Master 为 Master TCP 类型时该参数才有效
TriggerType	执行方式	TRIGGER_TYPE	-	-	执行方式
uiFunctionCode	功能码	MODBUS_FC_CODE	-	-	功能码
uiReadOffset	读偏移地址	UINT	0-65535	0	读偏移地址
uiReadLen	读长度	UINT	1-120	-	读长度
uiWriteOffset	写偏移地址	UINT	0-65535	0	写偏移地址
uiWriteLen	写长度	UINT	1-120	-	写长度
pWriteBuf	写区域	POINTER TO <TYPE>	-	-	写区域
pReadBuf	读区域	POINTER TO <TYPE>	-	-	读区域

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
xBusy	忙碌	BOOL	TRUE-FALSE	-	功能块正在运行

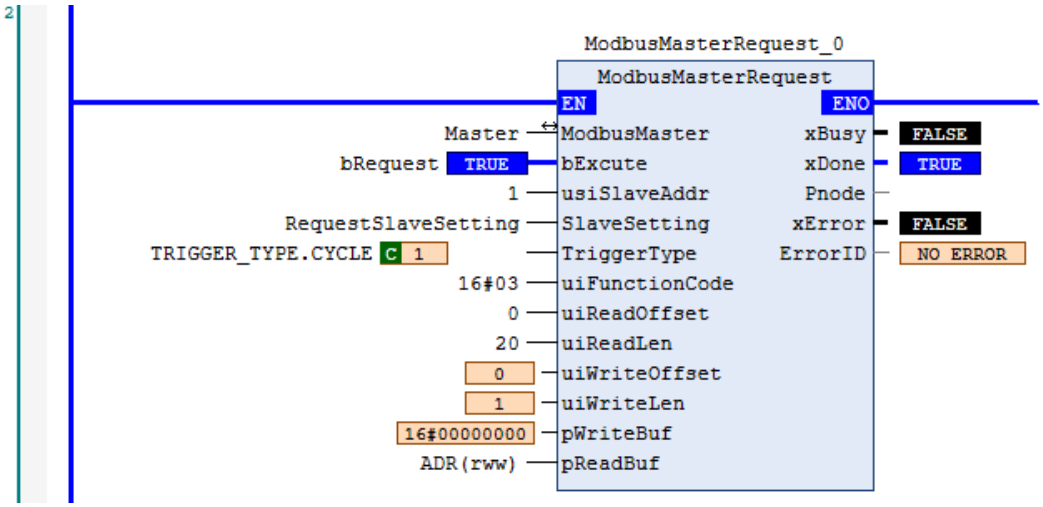
					中
xDone	完成标志	BOOL	TRUE-FALSE	-	Modbus 请求完成加入 Modbus 主站
Pnode	请求节点	NODE_REQUEST	-	-	请求节点
xError	错误状态	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_Common.ERROR_NO	-	-	错误信息

程序示例

ST： 执行一个从站地址为 1，执行方式为 TRIGGER_TYPE.CYCLE，功能码为 3 的 Modbus 请求。

```
ModbusMasterRequest0(
    ModbusMaster:=  Master,
    bExcute TRUE := bRequest TRUE,
    usiSlaveAddr 1 :=1 ,
    SlaveSetting:= RequestSlaveSetting,
    TriggerType CYCLE := TRIGGER_TYPE.CYCLE,
    uiFunctionCode MODBUS_FC_ := 16#03,
    uiReadOffset 0 := 0,
    uiReadLen 20 := 20,
    uiWriteOffset:= ,
    uiWriteLen:= ,
    pWriteBuf:= ,
    pReadBuf 16#F6A2FB78 :=ADR(rww) , //读
    xBusy FALSE => xBusy FALSE,
    xDone TRUE => xDone TRUE,
    Pnode=> ,
    xError FALSE => xError FALSE,
    ErrorID=> );
```

LD：执行一个从站地址为 1，执行方式为 TRIGGER_TYPE.CYCLE，功能码为 3 的 Modbus 请求。



6.3.4 ModbusMasterStatus

Modbus 主站状态指令。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ModbusMasterStatus	Modbus 主站状态指令	FB		ModbusMasterStatus_0(bExcute:= , pNode:= , Status=> , xError=> , ErrorID=> , NodeErrorID=> , xNodeError=> , ConnectState=>);	VM_ModbusMaster

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bExcute	使能	BOOL	TRUE-FALSE	FALSE	上升沿使能
pNode	Modbus 请求节点	NODE_REQUEST	-	-	Modbus 请求节点

输出变量

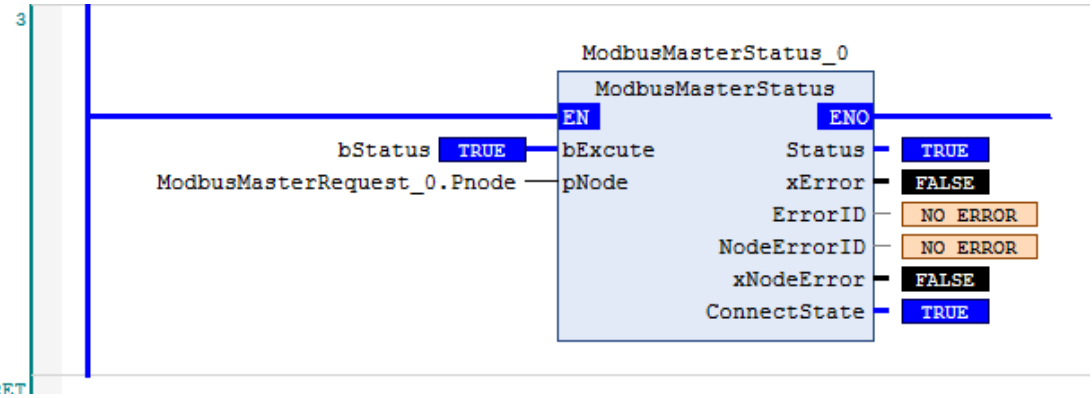
输出变量	名称	数据类型	有效范围	初始值	描述
Status	状态	BOOL	TRUE-FALSE	-	TRUE：Modbus 请求节点正常 FALSE：Modbus 请求节点异常
xError	错误状态	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_Common.ERROR_NO	-	-	错误信息
NodeErrorID	节点错误信息	VM_Common.ERROR_NO	-	-	Modbus 请求节点当前错误
xNodeError	节点错误状态	BOOL	TRUE-FALSE	-	Modbus 请求节点错误状态
ConnectState	连接状态	BOOL	TRUE-FALSE	-	连接状态，TRUE 为已连接，FALSE 为已掉线

程序示例

ST:

```
ModbusStatus0(
    bExcute TRUE := bStatus TRUE,
    pNode:= ModbusMasterRequest_0.Pnode,
    Status TRUE => Status TRUE,
    xError FALSE => xError FALSE,
    ErrorID NO_ERROR => ErrorID NO_ERROR,
    NodeErrorID NO_ERROR => NodeErrorID NO_ERROR,
    xNodeError FALSE => xNodeError FALSE,
    ConnectState TRUE => ConnectState TRUE);
```

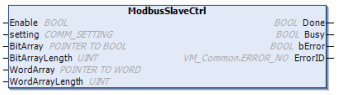
LD:



6.3.5 ModbusSlaveCtrl

Modbus 从站控制器指令。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
ModbusSlaveCtrl	Modbus 从站控制器指令	FB		ModbusSlaveCtrl_0(Enable:= , setting:= , BitArray:= , BitArrayLength:= , WordArray:= , WordArrayLength:= , Done=> , Busy=> , bError=> , ErrorID=>);	VM_ModbusSlave

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
Enable	使能	BOOL	TRUE-FALSE	FALSE	使能
setting	参数设置	COMM_SETTING	-	-	参数设置
BitArray	位数据区域地址	POINTER TO BOOL	-	-	位数据区域地址
BitArrayLength	位数据区域长度	UINT	0-65535	0	位数据区域长度
WordArray	字数据区域地址	POINTER TO WORD	-	-	字数据区域地址
WordArrayLength	字数据区域长度	UINT	0-65535	0	字数据区域长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	TRUE-FALSE	-	完成标志
Busy	忙碌	BOOL	TRUE-FALSE	FALSE	功能块正在运行中
bError	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_Common.ERROR_NO	-	-	错误信息

注意事项

使用 ModbusSlaveCtrl 指令同时需要安装 VM_Common 库。

程序示例

ST： 如果进行 Modbus TCP 通信，则输入变量 setting 中的 commType 必须设置为 VM_Common.COMM_TYPE.COMM_TCP，具体参数设置和实例如下

表达式	类型	值	准...	地址	注释
ModbusSlaveCtrl_0	ModbusSlaveCtrl				
setting	VM_ModbusSlave.COMM_SETTI...				
commType	COMM_TYPE	COMM_TCP			通讯类型，默认 TCP
slaveID	BYTE	1			从站 ID，默认 1
serial	SERIAL_COM	COM1			串口号
baudRate	BAUD_RATE	BAUD_RATE_9600			波特率
parity	PARITY	EVEN			校验，默认偶校验
stopBit	STOP_BIT	ONE_BIT			停止位，默认 1位
ipAddr	ARRAY [0..3] OF BYTE				允许连接的Modbus主站地址，默认不赋值(所有地址的Modbus主站都可以连接)
port	UINT	502			端口 默认 502
timeout	UINT	0			default frame response of timeout
bEnable	BOOL	TRUE			
bitData	ARRAY [1..2000] OF BOOL				
wordData	ARRAY [1..2000] OF WORD				
Done	BOOL	TRUE			

```
ModbusSlaveCtrl_0(
    Enable:= bEnable,
    setting:= setting,
    BitArray:= ADR(bitData),
    BitArrayLength:= SIZEOF(bitData),
    WordArray:= ADR(wordData),
    WordArrayLength:= SIZEOF(wordData),
    Done=> Done,
    Busy=> Busy,
    bError=> bError,
    ErrorID=> ErrorID);
```

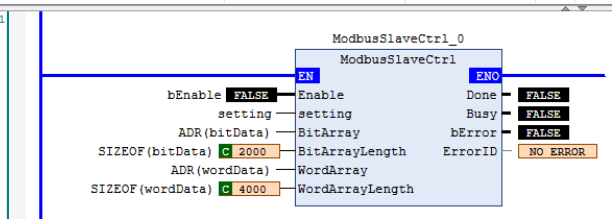
如果进行 Modbus RTU 通信，则输入变量 setting 中的 commType 必须设置为 VM_Common.COMM_TYPE.COMM_RTU，具体参数设置和实例如下

表达式	类型	值	准...	地址	注释
ModbusSlaveCtrl_0	ModbusSlaveCtrl				
setting	VM_ModbusSlave.COMM_SETTI...				
commType	COMM_TYPE	COMM_RTU			通讯类型，默认 TCP
slaveID	BYTE	1			从站 ID，默认 1
serial	SERIAL_COM	COM2			串口号
baudRate	BAUD_RATE	BAUD_RATE_115200			波特率
parity	PARITY	EVEN			校验，默认偶校验
stopBit	STOP_BIT	ONE_BIT			停止位，默认 1位
ipAddr	ARRAY [0..3] OF BYTE				允许连接的Modbus主站地址，默认不赋值(所有地址的Modbus主站都可以连接)
port	UINT	502			端口 默认 502
timeout	UINT	0			default frame response of timeout
bEnable	BOOL	TRUE			
bitData	ARRAY [1..2000] OF BOOL				
wordData	ARRAY [1..2000] OF WORD				

```
ModbusSlaveCtrl_0(
    Enable:= bEnable,
    setting:= setting,
    BitArray:= ADR(bitData),
    BitArrayLength:= SIZEOF(bitData),
    WordArray:= ADR(wordData),
    WordArrayLength:= SIZEOF(wordData),
    Done=> Done,
    Busy=> Busy,
    bError=> bError,
    ErrorID=> ErrorID);
```

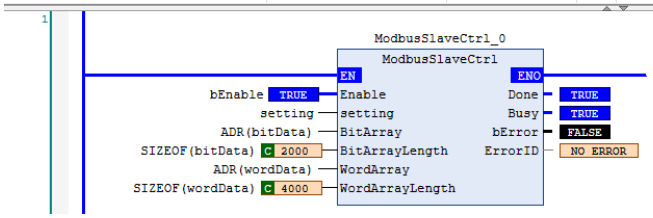
LD：如果需要进行 Modbus TCP 通信，则输入变量 setting 中的 commType 必须设置为 VM_Common.COMM_TYPE.COMM_TCP，具体参数设置和实例如下

表达式	类型	值	准...	地址	注释
ModbusSlaveCtrl_0	ModbusSlaveCtrl				
setting	VM_ModbusSlave.COMM_SETTI...				
commType	COMM_TYPE	COMM_TCP			通讯类型，默认 TCP
slaveID	BYTE	1			从站 ID，默认 1
serial	SERIAL_COM	COM2			串口号
baudRate	BAUD_RATE	BAUD_RATE_115200			波特率
parity	PARITY	EVEN			校验，默认偶校验
stopBit	STOP_BIT	ONE_BIT			停止位，默认 1位
ipAddr	ARRAY [0..3] OF BYTE				允许连接的Modbus主站地址，默认不赋值(所有地址的Modbus主站都可以连接)
port	UINT	502			端口 默认 502
timeout	UINT	0			default frame response of timeout
bitData	ARRAY [1..2000] OF BOOL				
wordData	ARRAY [1..2000] OF WORD				



如果进行 Modbus RTU 通信，则输入变量 setting 中的 commType 必须设置为 VM_Common.COMM_TYPE.COMM_RTU，具体参数设置和实例如下

表达式	类型	值	准...	地址	注释
ModbusSlaveCtrl_0	ModbusSlaveCtrl				
setting	VM_ModbusSlave.COMM_SETTI...				
commType	COMM_TYPE	COMM_RTU			通讯类型，默认 TCP
slaveID	BYTE	1			从站ID，默认 1
serial	SERIAL_COM	COM2			串口号
baudRate	BAUD_RATE	BAUD_RATE_115200			波特率
parity	PARITY	EVEN			校验，默认偶校验
stopBit	STOP_BIT	ONE_BIT			停止位，默认 1位
ipAddr	ARRAY [0..3] OF BYTE				允许连接的Modbus主站地址，默认不赋值(所有地址的Modbus主站都可以连接)
port	UINT	502			端口 默认 502
timeout	UINT	0			default frame response of timeout
bitData	ARRAY [1..2000] OF BOOL				
wordData	ARRAY [1..2000] OF WORD				



6.4 自由串口通信指令

6.4.1 指令列表

指令类别	名称	FB/FC	功能
自由串口通信指令	FreeProtocolCtrl	FB	串口自由协议配置指令
	FreeProtocolReceive	FB	串口自由协议接收指令
	FreeProtocolSend	FB	串口自由协议发送指令

6.4.2 FreeProtocolCtrl

串口自由协议配置指令。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
FreeProtocolCtrl	串口自由协议配置指令	FB		FreeProtocolCtrl_0(DeviceFd:= fd, bEnable:= , ComSetting:= , Status=> , Busy=> , xError=> , ErrorID=>);	VM_FreeProtocol

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
bEnable	使能	BOOL	TRUE-FALSE	FALSE	使能功能块
ComSetting	参数配置	COMM_SETTING	-	-	自由口参数配置

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Status	状态标志	BOOL	TRUE-FALSE	-	配置成功标志
Busy	忙碌	BOOL	TRUE-FALSE	-	功能块正在运行中
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_Common.ERROR_NO	-	-	错误信息

输入输出变量

输入变量	名称	数据类型	有效范围	初始值	描述
DeviceFd	设备描述符	DEVICE_FD	-	-	自由口设备描述符，供自由口发送接收使用

注意事项

使用 FreeProtocolCtrl 指令同时需要安装 VM_Common 库。

程序示例

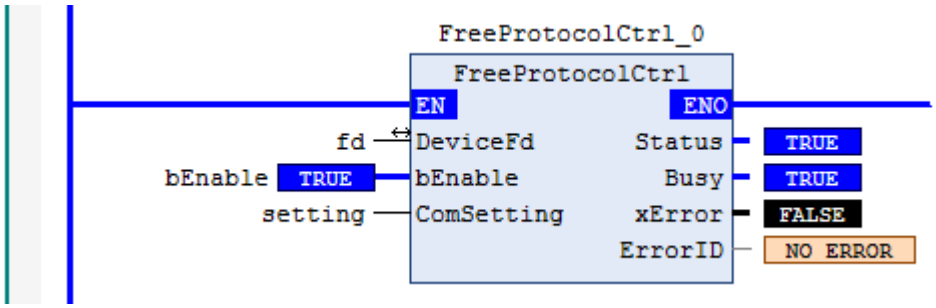
ST:

```
VAR
    FreeProtocolCtrl_0: FreeProtocolCtrl;
    setting:VM_FreeProtocol.COMM_SETTING :=(serial:=VM_Common.SERIAL_COM.COM2,
                                            baudRate:=VM_Common.BAUD_RATE.BAUD_RATE_115200,
                                            parity:=VM_Common.PARITY.NONE,
                                            dataBit:=VM_Common.DATA_BIT.DATA_BIT_8,
                                            stopBit:=VM_Common.STOP_BIT.ONE_BIT);
    fd: DEVICE_FD;
```

```
FreeProtocolCtrl_0(
    DeviceFd:= fd,
    bEnable TRUE := bEnable TRUE,
    ComSetting:= setting,
    Status TRUE => Status TRUE,
    Busy TRUE => Busy TRUE,
    xError FALSE => xError FALSE,
    ErrorID  NO_ERROR  => ErrorID  NO_ERROR  );
```

LD：

```
VAR
    FreeProtocolCtrl_0: FreeProtocolCtrl;
    setting:VM_FreeProtocol.COMM_SETTING :=(serial:=VM_Common.SERIAL_COM.COM2,
                                            baudRate:=VM_Common.BAUD_RATE.BAUD_RATE_115200,
                                            parity:=VM_Common.PARITY.NONE,
                                            dataBit:=VM_Common.DATA_BIT.DATA_BIT_8,
                                            stopBit:=VM_Common.STOP_BIT.ONE_BIT);
    fd: DEVICE_FD;
```



6.4.3 FreeProtocolReceive

串口自由协议接收指令。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
FreeProtocolReceive	串口自由协议接收指令	FB		FreeProtocolReceive_0(DeviceFd:= fd, bExcute:= , ReceiveArray:= Done=> , Busy=> , ReceiveLength=> , xError=> , ErrorID=>);	VM_FreeProtocol

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
DeviceFd	设备描述符	DEVICE_FD	-	-	自由口设备描述符，从 FreeProtocolCtrl 功能块获取
bExcute	使能	BOOL	TRUE-FALSE	FALSE	上升沿使能功能块
ReceiveArray	接收数据区域	ARRAY[1..512] OF BYTE	-	-	接收数组，每次接收都会覆盖之前的内容

输出变量

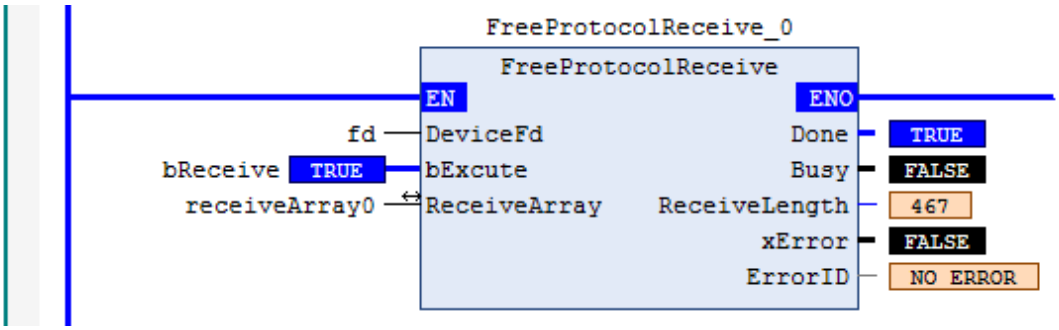
输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	TRUE-FALSE	-	接收数据完成标志
Busy	忙碌	BOOL	TRUE-FALSE	-	功能块正在运行中
ReceiveLength	接收数据长度	UINT	0-512	-	本次接收数据的实际长度，单位：byte
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_Common.ERROR_NO	-	-	错误信息

程序示例

ST:

```
FreeProtocolReceive_0(
    DeviceFd:= fd,
    bExcute TRUE := bReceive TRUE ,
    ReceiveArray:= receiveArray0,
    Done TRUE => Done TRUE ,
    Busy=> ,
    ReceiveLength 467 => ReceiveLength 467 ,
    xError=> ,
    ErrorID=> );
```

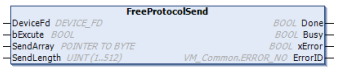
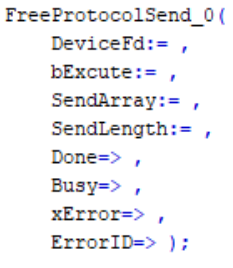
LD:



6.4.4 FreeProtocolSend

串口自由协议发送指令。

指令格式

指令	名称	FB/FC	LD 表现	ST 表现	库
FreeProtocolSend	串口自由协议发送指令	FB			VM_FreeProtocol


```
FreeProtocolSend_0(
    DeviceFd:= ,
    bExcute:= ,
    SendArray:= ,
    SendLength:= ,
    Done=> ,
    Busy=> ,
    xError=> ,
    ErrorID=> );
```

相关变量

输入变量

输入变量	名称	数据类型	有效范围	初始值	描述
DeviceFd	设备描述符	DEVICE_FD	-	-	自由口设备描述符，从 FreeProtocolCtrl 功能块获取
bExcute	使能	BOOL	TRUE-FALSE	FALSE	上升沿使能功能块
SendArray	发送数据区域	POINTER TO BYTE	-	-	指向发送数据区域的指针
SendLength	发送数据长度	UINT	1-512	1	将要发送的数据长度

输出变量

输出变量	名称	数据类型	有效范围	初始值	描述
Done	完成标志	BOOL	TRUE-FALSE	-	发送数据完成标志
Busy	忙碌	BOOL	TRUE-FALSE	-	功能块正在运行中
xError	错误标志	BOOL	TRUE-FALSE	-	错误状态
ErrorID	错误信息	VM_Common.ERROR_NO	-	-	错误信息

程序示例

ST:

```
FreeProtocolSend_0(
    DeviceFd:= fd,
    bExcute TRUE := bSend TRUE ,
    SendArray 16#F6A8EAEC := ADR(sendArray0),
    SendLength 50 := 50,
    Done TRUE => Done TRUE ,
    Busy FALSE => Busy FALSE ,
    xError=> ,
    ErrorID=> );
```

LD: