

Micro800 可编程控制器通用指令

类型编号 2080-LC10, 2080-LC20, 2080-LC30, 2080-LC50



重要用户信息

在安装、配置、操作、或维护产品之前，请阅读此文件以及关于对设备安装、配置及操作的附加资料部分的文件。除了所用应用的代码、规则及标准外，用户还需要熟悉安装及线路说明。

按照应用的实验代码，由适当的训练人员执行所需的安装、调整、交付使用、使用、拆、装以及维护的活动。若不使用厂商指定的方式，设备提供的保护将会受损。

任何情况下，对于因使用或应用本设备造成的任何间接或连带损失，Rockwell Automation, Inc. 概不负责。

本手册中包含的示例和图表仅用于说明。由于任何具体安装都涉及众多变化和要求，Rockwell Automation, Inc. 对于依据这些示例和图表所进行的实际使用不承担任何责任。

手册中说明使用信息、电路、设备或软件，Rockwell Automation 公司没有专利责任。

未经 Rockwell Automation, Inc. 的书面许可，不得复制本手册的全部或部分内容。

在整本手册中，我们在必要的地方给出了说明，以提醒您留意相关安全注意事项。



警告：用于标识在危险环境中可能导致爆炸，进而导致人员伤亡、财产损失或经济损失的操作或情况信息。



注意：用于标识可能导致人员伤亡、财产损失或经济损失的操作或情况信息。注意信息可帮助您识别危险，避免危险并了解相关后果。

重要说明： 用于标识对成功应用和了解本产品有重要作用的信息。

位于设备上面或内部的标签提供了具体的预防措施。



触电危险：位于设备（例如，驱动器或电机）表面或内部的标签，提醒相关人员可能存在危险电压。



灼伤危险：位于设备（例如，驱动器或电机）表面或内部的标签，提醒相关人员表面可能达到危险温度。



电弧闪络危险：位于设备（例如，电机控制中心）表面或内部的标签，提醒相关人员可能存在危险电压。电弧闪络将会引起严重伤亡。穿着适当的“个人防护装备”安全工作和个人防护装备要按照所有管理法规要求。

Allen-Bradley、Rockwell Software 以及 Rockwell Automation 是 Rockwell Automation 公司的商标。

不属于 Rockwell Automation 的商标是其各自公司的财产。

前言

在本手册中..... 13

支持的控制器..... 13

附加资源..... 13

第 1 章

查找有关指令和梯形元素的信息

块..... 15

 运算符..... 16

 函数..... 16

 功能块..... 18

按字母顺序排列的指令集..... 20

按类型和功能排列的指令集..... 25

第 2 章

梯形图 (LD) 语言

LD 编程..... 35

LD 程序开发环境..... 36

第 3 章

梯形图 (LD) 元素

梯级..... 39

指令块 (LD)..... 43

线圈..... 48

触点..... 55

返回..... 59

跳转..... 60

LD 程序中的指令块..... 61

使用 LD 语言编辑器..... 62

梯形图 (LD) 程序示例..... 63

 示例：R_TRIG 功能块..... 63

 示例：使用 Subtraction (-)、ABS 和 Less than (<) 比较实型值..... 64

LD 键盘快捷键..... 65

第 4 章

警报指令	LIM_ALRM.....	69

第 5 章

算术指令

ABS.....	74
ACOS.....	76
ACOS_LREAL.....	77
Addition.....	79
ASIN.....	80
ASIN_LREAL.....	82
ATAN.....	84
ATAN_LREAL.....	86
COS.....	87
COS_LREAL.....	89
Division.....	91
EXPT.....	92
LOG.....	94
MOD.....	96
MOV.....	98
Multiplication.....	99
取反.....	100
POW.....	101
RAND.....	103
SIN.....	105
SIN_LREAL.....	107
SQRT.....	109
Subtraction.....	110
TAN.....	111
TAN_LREAL.....	113
TRUNC.....	115

第 6 章

ASCII 串行端口指令	ABL.....	119
	ACB	121
	ACL.....	123
	AHL	124
	ARD	127
	ARL.....	129
	AWA	131
	AWT	133
	ASCII 参数详细信息	135

ABL 错误代码	135
ABLACB 数据类型	136
ACL 数据类型	137
AHL ChannelSts 数据类型	137
AHLI 数据类型	138
ARDARL 数据类型	139
AWAAWT 数据类型	139

第 7 章

二进制指令

AND_MASK	141
NOT_MASK	143
OR_MASK	145
ROL	147
ROR	149
SHL	151
SHR	153
XOR_MASK	155

第 8 章

布尔指令

F_TRIG	159
R_TRIG	161
RS	164
OR	165
AND	166
XOR	167
非	167
SR	168
TTABLE	170
TTABLE 输入组合	173
MUX8B	174
MUX4B	179

第 9 章

通信指令

MSG_CIPGENERIC	183
CIPAPPCFG 数据类型	186
CIPCONTROLCFG 数据类型	187

CIPSTATUS 数据类型.....	188
CIPTARGETCFG 数据类型.....	191
MSG_CIPSYMBOLIC.....	195
CIPSYMBOLICCFG 数据类型.....	198
CIPAPPCFG 数据类型.....	200
CIPCONTROLCFG 数据类型.....	200
CIPSTATUS 数据类型.....	202
CIPTARGETCFG 数据类型.....	204
已支持的 CIP 串行功能数据包大小.....	208
MSG_MODBUS.....	209
Modbus 错误代码.....	211
MODBUSLOCPARA 数据类型.....	212
消息执行过程 (Rung = TRUE).....	215
MODBUSTARPARA 数据类型.....	217
MSG_MODBUS2.....	217
Modbus2 错误代码.....	220
MODBUS2LOCPARA 数据类型.....	221
MODBUS2TARPARA 数据类型.....	223
消息执行过程和时间图.....	225
消息执行过程 (常规).....	226
消息执行次序 (常规).....	227
消息执行过程 (Rung = TRUE).....	228
消息执行时间图 (Rung = TRUE).....	230
消息执行过程 (Rung = FALSE).....	231
消息执行时间图 (Rung = FALSE).....	233
消息执行过程 (错误).....	234
消息执行时间图 (错误).....	235
使用通信 (消息) 功能块.....	236
配置显式消息的对象数据值 (MSG_CIPGENERIC).....	236
示例：如何创建 MSG_CIPGENERIC 消息程序以从控制器中 读取数据.....	239

示例：如何创建 MSG_CIPSYMBOLIC 消息程序以将值写入 变量.....	253
示例：如果配置 Modbus 通信以读取和写入驱动器	268
通信协议支持.....	272
嵌入式通信信道.....	274

第 10 章

比较指令

等于.....	275
Greater than.....	277
Greater Than or Equal.....	279
Less Than.....	280
Less Than or Equal	281
Not equal.....	282

第 11 章

计数器指令

CTD.....	285
CTU.....	287
CTUD.....	289

第 12 章

数据转换指令

ANY_TO_BOOL.....	294
ANY_TO_BYTE	295
ANY_TO_DATE.....	296
ANY_TO_DINT.....	297
ANY_TO_DWORD	298
ANY_TO_INT.....	299
ANY_TO_LINT	300
ANY_TO_LREAL.....	301
ANY_TO_LWORD.....	302
ANY_TO_REAL	303
ANY_TO_SINT.....	304
ANY_TO_STRING	305
ANY_TO_TIME.....	307
ANY_TO_UDINT	308
ANY_TO_UINT.....	310
ANY_TO_ULINT.....	311
ANY_TO_USINT.....	312
ANY_TO_WORD	313

	第 13 章	
数据操作指令	AVERAGE	315
	COP	318
	COP 操作状态值.....	321
	复制到不同的数据类型.....	322
	MIN.....	322
	MAX	324
	第 14 章	
高速计数器 (HSC) 指令	什么是高速计数器?	327
	HSC.....	328
	HSCCmd 值.....	330
	HSCAPP 数据类型	332
	HSCSTS 数据类型.....	339
	PLS 数据类型.....	348
	HSC 状态代码 (STS).....	350
	HSC_SET_STS.....	351
	使用高速计数器指令.....	353
	更新 HSC 应用程序数据.....	354
	“高速计数器 (HSC) 用户中断”对话框.....	354
	配置高速计数器 (HSC) 用户中断	Error! Bookmark not defined.
	添加和配置高速计数器 (HSC) 用户中断	356
	配置可编程限位开关 (PLS).....	360
	示例：如何创建高速计数器 (HSC) 程序.....	361
	添加可编程限位开关 (PLS) 功能.....	Error! Bookmark not defined.
	示例：可编程限位开关 (PLS) 已启用	378
	第 15 章	
输入/输出指令	LCD.....	382
	LCD_BKLT_REM	384
	LCD_BKLT_REM 状态代码	387

LCD_REM.....	387
LCD_REM 状态代码.....	392
RHC	392
RPC.....	394
DLG	395
DLG 状态代码	397
DLG 错误代码	398
IIM	398
IIM 状态代码	400
IOM.....	401
IOM 状态代码.....	403
KEY_READ	404
KEY_READ_REM.....	407
KEY_READ_REM 操作.....	409
KEY_READ_REM 状态代码.....	409
键数据位字段表.....	410
MM_INFO	411
MMINFO 数据类型	413
PLUGIN_INFO.....	414
PLUGIN_READ.....	417
PLUGIN_READ 状态代码.....	419
PLUGIN_RESET.....	419
PLUGIN_WRITE	421
RCP.....	424
RCP 状态代码.....	426
RCP 错误代码	426
RTC_READ.....	427
RTC 数据类型.....	429
RTC_SET.....	430
RTC 设置状态值.....	432
SYS_INFO	432
SYS_INFO 数据类型	434
TRIMPOT_READ	436
微调电位 ID 定义	437
微调电位操作状态值.....	438

第 16 章

中断指令

STIS 439

UIC..... 441

UID 443

UIE..... 445

UIF 447

第 17 章

运动控制指令

运动控制功能块的一般规则..... 452

运动控制功能块参数详细信息..... 455

 运动控制轴状态..... 455

 运动控制功能块参数编号..... 458

 运动控制功能块错误 ID 460

 轴出错场景..... 462

 AXIS_REF 数据类型..... 462

 FB_AXIS_REF 数据类型..... 464

 轴变量..... 465

MC_AbortTrigger 466

MC_Halt 470

MC_Home..... 474

 归位模式..... 478

MC_MoveAbsolute..... 478

MC_MoveRelative..... 484

MC_MoveVelocity 490

MC_Power 495

MC_ReadActualPosition 499

MC_ReadActualVelocity..... 501

MC_ReadAxisError 503

 AxisErrorID 错误代码 507

MC_ReadBoolParameter 509

MC_ReadParameter 512

MC_ReadStatus 516

MC_Reset 522

MC_SetPosition 525

MC_Stop..... 529

MC_TouchProbe..... 533

 运动固定输入/输出 538

MC_WriteBoolParameter 539

	MC_WriteParameter	543
	第 18 章	
过程控制指令	DERIVATE.....	549
	HYSTER.....	551
	INTEGRAL.....	553
	PWM.....	560
	PWM 状态代码.....	563
	SCALER	564
	STACKINT.....	567
	TND	570
	LIMIT	572
	第 19 章	
程序控制指令	SUS	577
	第 20 章	
比例积分微分 (PID) 指令	什么是比例积分微分 (PID) 控制 ?	581
	IPIDController 功能块如何实施 PID 控制.....	581
	IPIDCONTROLLER	583
	GAIN_PID 数据类型	588
	AT_Param 数据类型	588
	IPIDController 功能块操作	589
	使用比例积分微分指令.....	590
	通过 IPIDController 功能块实现自动调节	591
	示例：具有自动调节的 IPIDController	599
	示例：如何创建操作值的反馈回路.....	601
	示例：如何将 UDFB 添加到 PID 程序	603
	示例：如何创建 IPIDController 程序以控制温度	604
	示例：如何创建 IPIDController 程序以控制供水水位	607

第 21 章

实时时钟 (RTC) 指令	RTC_READ.....	611
	RTC_SET.....	613
	RTC 数据类型.....	615
	RTC 设置状态值.....	616

第 22 章

字符串操作指令	ASCII	617
	CHAR	619
	DELETE	621
	FIND	624
	INSERT.....	626
	LEFT.....	628
	MID.....	630
	MLEN	633
	RIGHT	634
	REPLACE.....	637

第 23 章

计时器指令

计时器指令配置.....	641
TOF	642
TON	644
TONOFF	647
TP	650
RTO.....	652
DOY	654
DOYDATA 数据类型.....	656
TDF	656
TOW	658
TOWDATA 数据类型	662
索引.....	Error! Bookmark not defined.

在本手册中

本指南提供有关指令集的参考信息，这些指令集可用于开发在 Micro800 控制系统中使用的程序。该指令集包括结构化文本 (ST)、梯形图 (LD)、功能块图 (FBD) 编程语言支持。此外，还定义了 Connected Workbench 开发环境中支持的梯形图元素。

支持的控制器

Connected Components Workbench™ 包括用于下列 Micro800™ 控制器的配置支持。

- 2080-LC10-12AWA
- 2080-LC10-12DWD
- 2080-LC10-12QBB
- 2080-LC10-12QWB
- 2080-LC20-20AWB
- 2080-LC20-20QBB
- 2080-LC20-20QWB
- 2080-LC30-10QVB
- 2080-LC30-10QWB
- 2080-LC30-16AWB
- 2080-LC30-16QVB
- 2080-LC30-16QWB
- 2080-LC30-24QBB
- 2080-LC30-24QVB
- 2080-LC30-24QWB
- 2080-LC30-48AWB
- 2080-LC30-48QBB
- 2080-LC30-48QVB
- 2080-LC30-48QWB
- 2080-LC50-24AWB
- 2080-LC50-24QBB
- 2080-LC50-24QVB
- 2080-LC50-24QWB
- 2080-LC50-48AWB
- 2080-LC50-48QBB
- 2080-LC50-48QVB
- 2080-LC50-48QWB

附加资源

这些文档包含有关相关的 Rockwell Automation 产品的附加信息。

资源	说明
《工业自动化配线和接地指南》(出版号 1770-4.1 , 可通过 http://literature.rockwellautomation.com/idc/groups/literature/documents/in/1770-in041_-en-p.pdf 获取。	提供用于安装 Rockwell Automation 工业系统的一般指南。
产品认证网站 , http://www.ab.com	提供符合性声明、证书以及其他认证详细信息。

您可以通过 <http://www.rockwellautomation.com/literature> 查看或下载出版物。要订购技术文档的纸质副本 , 请联系您当地的 Rockwell Automation 经销商或销售代表。

查找有关指令和梯形元素的信息

Connected Components Workbench™ 包括一个含结构和数组的全面指令集、梯形逻辑的开发环境、结构化文本、功能块图和用户定义的功能块程序。

此外，Connected Components Workbench 还包括 Micro800™ 控制器、PowerFlex® 驱动器、安全继电器设备、PanelView™ Component 图形终端以及串行和网络连接选项的用户界面配置工具。

指令集

有关特定指令的信息，包括说明、参数详细信息以及语言示例，请从目录中或从以下参考主题中找到相应指令块。

[按字母顺序排列的指令参考](#) 参考页数 [20](#)

[按类型和功能排列的指令集](#) 参考页数 [25](#)

梯形图元素

有关 Connected Components Workbench 中使用的梯形元素的说明，请参见以下章节：

[梯形图 \(LD\) 语言参考](#) 参考页数 [35](#)

块

Connected Components Workbench 指令集包括符合 IEC 61131-3 标准指令块。指令块总的来说包括运算符、函数和功能块。

- [运算符](#) 参考页数 [16](#)
- [函数](#) 参考页数 [16](#)
- [功能块](#) 参考页数 [18](#)

另请参见

程序元素

程序

运算符

运算符是指诸如算术运算、布尔运算、比较运算或数据转换等基本逻辑操作。

另请参见

程序元素

程序

[功能块](#) 参考页数 [18](#)

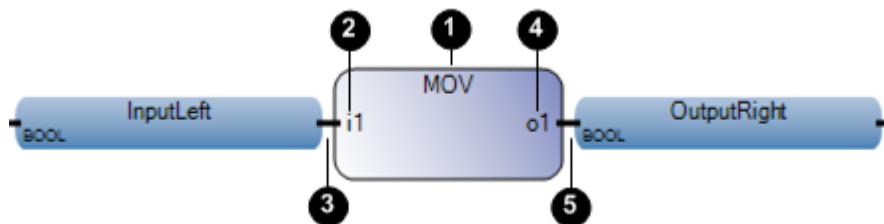
[函数](#) 参考页数 [16](#)

函数

函数具有一个或多个输入参数及一个输出参数。

指令块格式

一个指令块由单个矩形表示，并且具有固定数量的输入连接点和输出连接点。一个基本指令块执行一个功能。

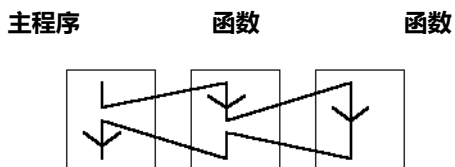


项目编号	项目	说明
❶	块名称	指令块要执行的功能的名称写在其矩形形状内（位于顶端）。
❷	输入	指令块的每个输入都带有标签且具有定义的类型。
❸	输入连接	输入在左边框进行连接。
❹	输出	指令块的每个输出都带有标签且具有定义的类型。
❺	输出连接	输出在右边框进行连接。

调用函数

Connected Components Workbench™ 不支持递归函数调用。当“函数”部分的某个函数由其自身或其被调用函数之一调用时，会发生运行时错误。此外，函数不会存储其局部变量的本地值。由于函数未经实例化，因而它们无法调用功能块。

- 函数可以由程序、函数或功能块加以调用。
- 任何部分的任何程序均可调用一个或多个函数。函数可具有局部变量。
- 函数没有实例，这意味着不会存储本地数据，因此本地数据通常无法在两次调用之间转承。
- 函数由其父程序来执行，因此父程序会在该函数执行结束前挂起。



定义函数和参数名称

必须为函数的每个调用（输入）参数或返回（输出）参数指定一种类型或唯一名称，来显式定义该函数的接口。一个函数具有一个返回参数。功能块返回参数的值因各种不同编程语言（FBD、LD、ST）而异。

函数名称和函数参数名称最多可包含 128 个字符。函数参数名称可以字母或下划线字符开头，后跟字母、数字和单个下划线字符。

另请参见

程序元素

程序

[功能块](#) 参考页数 [18](#)

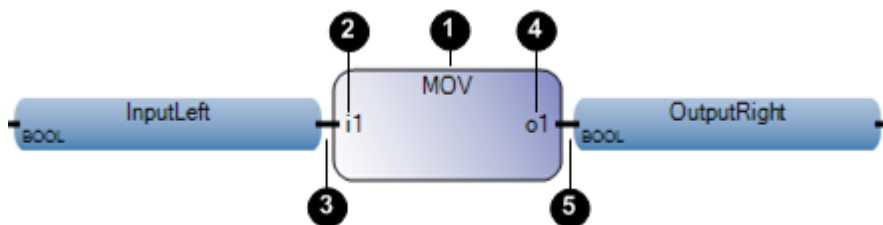
[运算符](#) 参考页数 [16](#)

功能块

功能块是一个具有输入和输出参数并且处理内部数据（参数）的指令块。它可以用结构化文本、梯形图或功能块图语言编写。

指令块格式

一个指令块由单个矩形表示，并且具有固定数量的输入连接点和输出连接点。一个基本指令块执行一个功能。



项目编号	项目	说明
❶	块名称	指令块要执行的功能的名称写在其矩形形状内（位于顶端）。
❷	输入	指令块的每个输入都带有标签且具有定义的类型。
❸	输入连接	输入在左边框进行连接。
❹	输出	指令块的每个输出都带有标签且具有定义的类型。
❺	输出连接	输出在右边框进行连接。

调用功能块

当在程序中调用功能块时，实际上调用的是这个块的实例。该实例使用相同的代码，但是输入和输出参数已经过实例化，这意味着将针对功能块的每个实例复制局部变量。功能块实例的变量值将从一个循环存储至另一个循环。

功能块可以由程序或其他功能块加以调用。它们无法由函数调用，因为函数未经实例化。

定义功能块和参数名称

必须使用功能块的每个调用（输入）参数或返回（输出）参数的类型或唯一名称，来显式定义该功能块的接口。功能块可具有多个输出参数。功能块返回参数的值因各种不同编程语言（FBD、LD、ST）而异。

功能块名称和功能块参数名称最多可包含 128 个字符。功能块参数名称可以字母或下划线字符开头，后跟字母、数字和单个下划线字符。

另请参见

程序元素

程序

[函数](#) 参考页数 [16](#)

[运算符](#) 参考页数 [16](#)

按字母顺序排列的指令集

下表按字母顺序列出了 Connected Components workbench 中提供的指令集。

指令	指令块类型
ABL 参考页数 119	功能块
ABS 参考页数 74	函数
ACB 参考页数 121	功能块
ACL 参考页数 123	功能块
ACOS 参考页数 76	函数
ACOS_LREAL 参考页数 77	函数
Addition 参考页数 79	运算符
AHL 参考页数 124	功能块
AND 参考页数 166	运算符
AND_MASK 参考页数 141	函数
ANY_TO_BOOL 参考页数 294	函数
ANY_TO_BYTE 参考页数 295	函数
ANY_TO_DATE 参考页数 296	函数
ANY_TO_DINT 参考页数 297	函数
ANY_TO_DWORD 参考页数 298	函数
ANY_TO_INT 参考页数 299	函数
ANY_TO_LINT 参考页数 300	函数
ANY_TO_LREAL 参考页数 301	函数
ANY_TO_LWORD 参考页数 302	函数
ANY_TO_REAL 参考页数 303	函数
ANY_TO_SINT 参考页数 304	函数
ANY_TO_STRING 参考页数 305	函数
ANY_TO_TIME 参考页数 307	函数
ANY_TO_UDINT 参考页数 308	函数
ANY_TO_UINT 参考页数 310	函数
ANY_TO_ULINT 参考页数 311	函数

指令	指令块类型
ANY_TO_USINT 参考页数 312	函数
ANY_TO_WORD 参考页数 313	函数
ARD 参考页数 127	功能块
ARL 参考页数 129	功能块
ASCII 参考页数 617	函数
ASIN 参考页数 80	函数
ASIN_LREAL 参考页数 82	函数
ATAN 参考页数 84	函数
ATAN_LREAL 参考页数 86	函数
AVERAGE 参考页数 315	功能块
AWA 参考页数 131	函数
AWT 参考页数 133	函数
CHAR 参考页数 619	函数
COP 参考页数 318	功能块
COS 参考页数 87	函数
COS_LREAL 参考页数 89	函数
CTD 参考页数 285	函数
CTU 参考页数 287	函数
CTUD 参考页数 289	函数
DELETE 参考页数 621	函数
DERIVATE 参考页数 549	功能块
Division 参考页数 91	运算符
DOY 参考页数 654	函数
等于 参考页数 275	运算符
EXPT 参考页数 92	函数
FIND 参考页数 624	函数
F_TRIG 参考页数 159	功能块
大于 参考页数 277	运算符
大于或等于 参考页数 279	运算符
HSC 参考页数 328	功能块
HSC_SET_STS 参考页数 351	功能块

指令	指令块类型
HYSTER 参考页数 551	功能块
IIM 参考页数 398	功能块
INSERT 参考页数 626	函数
INTEGRAL 参考页数 553	功能块
IOM 参考页数 401	功能块
IPIDCONTROLLER 参考页数 583	功能块
KEY_READ 参考页数 404	功能块
LCD 参考页数 382	函数
LEFT 参考页数 628	函数
小于 参考页数 280	运算符
小于或等于 参考页数 281	运算符
LIM_ALARM 参考页数 69	功能块
LIMIT 参考页数 572	函数
LOG 参考页数 94	函数
MAX 参考页数 324	函数
MC_AbortTrigger 参考页数 466	功能块
MC_Halt 参考页数 470	功能块
MC_Home 参考页数 474	功能块
MC_MoveAbsolute 参考页数 478	功能块
MC_MoveRelative 参考页数 484	功能块
MC_MoveVelocity 参考页数 490	功能块
MC_Power 参考页数 495	功能块
MC_ReadActualPosition 参考页数 501	功能块
MC_ReadActualVelocity 参考页数 501	功能块
MC_ReadAxisError 参考页数 503	功能块
MC_ReadBoolParameter 参考页数 509	功能块
MC_ReadParameter 参考页数 512	功能块
MC_ReadStatus 参考页数 516	功能块
MC_Reset 参考页数 522	功能块
MC_SetPosition 参考页数 525	功能块

指令	指令块类型
MC_Stop 参考页数 529	功能块
MC_TouchProbe 参考页数 533	功能块
MC_WriteBoolParameter 参考页数 539	功能块
MC_WriteParameter 参考页数 543	功能块
MID 参考页数 630	函数
MIN 参考页数 322	函数
MLEN 参考页数 633	函数
MM_INFO 参考页数 411	功能块
MOD 参考页数 96	函数
MOV 参考页数 98	运算符
MSG_CIPGENERIC 参考页数 183	函数
MSG_CIPSYMBOLIC 参考页数 195	函数
MSG_MODBUS 参考页数 209	函数
MSG_MODBUS2 参考页数 217	函数
Multiplication 参考页数 99	运算符
MUX4B 参考页数 179	函数
MUX8B 参考页数 174	函数
非 参考页数 167	运算符
不等于 参考页数 282	运算符
取反 参考页数 100	运算符
NOT_MASK 参考页数 143	函数
OR_MASK 参考页数 145	函数
OR 参考页数 165	运算符
PLUGIN_INFO 参考页数 414	功能块
PLUGIN_READ 参考页数 417	功能块
PLUGIN_RESET 参考页数 419	功能块
PLUGIN_WRITE 参考页数 421	功能块
POW 参考页数 101	函数
R_TRIG 参考页数 161	功能块
RAND 参考页数 103	函数
REPLACE 参考页数 637	函数

指令	指令块类型
RHC 参考页数 392	函数
RIGHT 参考页数 634	函数
ROL 参考页数 147	函数
ROR 参考页数 149	函数
RPC 参考页数 394	函数
RS 参考页数 164	功能块
RTC_READ 参考页数 427	功能块
RTC_SET 参考页数 430	功能块
RTO 参考页数 652	功能块
SCALER 参考页数 564	功能块
SHL 参考页数 151	函数
SHR 参考页数 153	函数
SIN 参考页数 105	函数
SIN_LREAL 参考页数 107	函数
SQRT 参考页数 109	函数
SR 参考页数 168	功能块
STACKINT 参考页数 567	功能块
STIS 参考页数 439	函数
Subtraction 参考页数 110	运算符
SUS 参考页数 577	功能块
SYS_INFO 参考页数 432	功能块
TAN 参考页数 111	函数
TAN_LREAL 参考页数 113	函数
TDF 参考页数 656	函数
TND 参考页数 570	函数
TOF 参考页数 642	功能块
TON 参考页数 644	功能块
TONOFF 参考页数 647	功能块
TOW 参考页数 658	函数
TP 参考页数 650	功能块
TRIMPOT_READ 参考页数 436	功能块

指令	指令块类型
TRUNC 参考页数 115	函数
TTABLE 参考页数 170	函数
UIC 参考页数 441	函数
UID 参考页数 443	函数
UIE 参考页数 445	函数
UIF 参考页数 447	函数
XOR_MASK 参考页数 155	函数
XOR 参考页数 167	运算符

另请参见

[查找有关指令和梯形元素的信息](#) 参考页数 [15](#)

[按指令类型排列的指令参考](#) 参考页数 [25](#)

按类型和功能排列的指令集

功能块

Connected Components Workbench™ 支持下列功能块：

类别	功能块	说明
警报 参考页数 69	LIM_ALARM 参考页数 69	高/低限制报警，并出现滞后现象
ASCII 串行端口 参考页数 119	ABL 参考页数 119	确定缓冲区中的字符数（甚至可包括行尾字符）
	ACB 参考页数 121	确定缓冲区中的字符总数
	ACL 参考页数 123	清除接收和传输的缓冲
	AHL 参考页数 124	设置或重置调制解调器握手信号线

	ARD 参考页数 127	从输入缓冲区读取字符，并将这些字符放置到某个字符串中
	ARL 参考页数 129	从输入缓冲区读取一行字符，并将这些字符放置到某个字符串中
	AWA 参考页数 131	写入某个字符串，该字符串包含追加到外部设备的用户配置字符
	AWT 参考页数 133	将字符从源字符串写入到外部设备中
布尔运算	F_TRIG 参考页数 159	下降沿检测
	RS 参考页数 164	双稳态重置基准
	R_TRIG 参考页数 161	上升沿检测
	SR 参考页数 168	设置双稳态基准
通信 参考页数 183	MSG_CIPGENERI C 参考页数 183	发送 CIP 显式消息
	MSG_CIPSYMBOL IC 参考页数 195	发送 CIP 符号消息
	MSG_MODBUS 考页数 209	发送 Modbus 消息

	MSG MODBUS2 参考页数 217	通过以太网通道发送 MODBUS/TCP 消息
计数器 参考页数 285	CTD 参考页数 285	下数计数器
	CTU 参考页数 287	上数计数器
	CTUD 参考页数 289	上/下数计数器
数据操作	AVERAGE 参考页数 315	在 N 个取样上运行平均值
	COP 参考页数 318	将源中的二进制数据复制到目标
输入/输出	HSC 参考页数 328	设置要应用到高速计数器 (HSC) 的高和低预设值及输出源
	HSC SET STS 参考页数 351	手动设置/重置高速计数器状态
	IIM 参考页数 398	在进行正常输出扫描前更新输入
	IOM 参考页数 401	在进行正常输出扫描前更新输出
	KEY READ 参考页数 404	读取可选 LCD 模块上的键状态 (仅限 Micro810™)

	MM_INFO 参考页数 411	读取内存模块标题信息
	PLUGIN_INFO 参考页数 414	从类属插件模块中获取模块信息（不包括模块信息）
	PLUGIN_READ 参考页数 417	从类属插件模块中获取数据
	PLUGIN_RESET 参考页数 419	重置类属插件模块（硬件重置）
	PLUGIN_WRITE 参考页数 421	将数据写入到类属插件模块
	RTC_READ 参考页数 427	读取实时时钟（RTC）模块信息
	RTC_SET 参考页数 430	将实时时钟数据设置到实时时钟模块中
	SYS_INFO 参考页数 432	读取 Micro800™ 系统状态
	TRIMPOT_READ 参考页数 436	从特定微调电位中读取微调电位值
运动控制 参考页数 451	MC_AbortTrigger 参考页数 466	中止连接到触发事件的功能块
	MC_Halt 参考页数 470	在正常操作条件下停止轴

MC_Home 参考页数 474	命令轴执行 <search home> 序列
MC_MoveAbsolute 参考页数 478	命令受控制的运动到指定的绝对位置
MC_MoveRelative 参考页数 484	在执行时命令与实际位置相对的指定距离的受控制运动。
MC_MoveVelocity 参考页数 490	以指定速率命令从未结束的受控制运动。
MC_Power 参考页数 495	控制功率（打开或关闭）
MC_ReadAxisError 参考页数 503	介绍一般轴错误
MC_ReadBoolParameter 参考页数 509	返回特定于供应商的类型为 BOOL 的参数的值
MC_ReadParameter 参考页数 512	返回特定于供应商的参数的值
MC_ReadStatus 参考页数 516	返回轴的状态
MC_Reset 参考页数 522	重置所有内部轴相关的错误
MC_SetPosition 参考页数 525	通过控制实际位置来转移轴坐标系统

	MC_Stop 参考页数 529	命令受控制的运动停止
	MC_TouchProbe 考页数 533	在触发事件中记录轴位置
	MC_WriteBoolParameter 参考页数 539	修改特定于供应商的类型为 BOOL 的参数 的值
	MC_WriteParameter 参考页数 543	修改特定于供应商的参数的值
流程控制	DERIVATE 参考 数 549	实型值的区分
	HYSTER 参考页数 551	在不同实型值上的布尔迟滞
	INTEGRAL 参考 数 553	随时间进行集成
	IPIDCONTROLLER 参考页数 583	比例积分导数
	SCALER 参考页数 564	根据输出范围调整输入值
	STACKINT 参考 数 567	整数堆栈
程序控制	SUS 参考页数 577	挂起应用程序的执行
时间	TOF 参考页数 642	关断延时计时

	TON 参考页数 644	接通延时计时
	TONOFF 参考页数 647	延迟打开真梯级上的输出，然后延迟关闭假梯级上的输出
	TP 参考页数 650	脉冲计时

函数

下表按功能类别列出了各种函数。

指令	功能类别
ABS 参考页数 74	算术指令 参考页数 73
ACOS 参考页数 76	
ACOS LREAL 参考页数 77	
AND MASK 参考页数 141	二进制指令 参考页数 141
ASCII 参考页数 617	字符串操作指令 参考页数 617
ASIN 参考页数 80	算术指令 参考页数 73
ASIN LREAL 参考页数 82	
ATAN 参考页数 84	
ATAN LREAL 参考页数 86	字符串操作指令 参考页数 617
CHAR 参考页数 619	
COS 参考页数 87	
COS LREAL 参考页数 89	算术指令 参考页数 73
DELETE 参考页数 621	
DOY 参考页数 654	
EXPT 参考页数 92	字符串操作指令 参考页数 617
FIND 参考页数 624	计时器指令 参考页数 641
INSERT 参考页数 626	
LCD 参考页数 382	
LEFT 参考页数 628	算术指令 参考页数 73
LIMIT 参考页数 572	字符串操作指令 参考页数 617
LOG 参考页数 94	输入/输出指令 参考页数 381
MAX 参考页数 324	字符串操作指令 参考页数 617
	过程控制指令 参考页数 549
	算术指令 参考页数 73
	数据操作指令 参考页数 315

指令	功能类别
MID 参考页数 630	字符串操作指令 参考页数 617
MIN 参考页数 322	数据操作指令 参考页数 315
MLEN 参考页数 633	字符串操作指令 参考页数 617
MOD 参考页数 96	算术指令 参考页数 73
MUX4B 参考页数 179	布尔指令 参考页数 159
MUX8B 参考页数 174	
NOT_MASK 参考页数 143	二进制指令 参考页数 141
OR_MASK 参考页数 145	
POW 参考页数 101	算术指令 参考页数 73
RAND 参考页数 103	
REPLACE 参考页数 637	字符串操作指令 参考页数 617
RHC 参考页数 392	输入/输出指令 参考页数 381
RIGHT 参考页数 634	字符串操作指令 参考页数 617
ROL 参考页数 147	二进制指令 参考页数 141
ROR 参考页数 149	
RPC 参考页数 394	输入/输出指令 参考页数 381
SHL 参考页数 151	二进制指令 参考页数 141
SHR 参考页数 153	
SIN 参考页数 105	算术指令 参考页数 73
SIN_LREAL 参考页数 107	
SQRT 参考页数 109	
STIS 参考页数 439	中断指令 参考页数 439
TAN 参考页数 111	算术指令 参考页数 73
TAN_LREAL 参考页数 113	
TDF 参考页数 656	计时器指令 参考页数 641
TND 参考页数 570	过程控制指令 参考页数 549
TOW 参考页数 658	计时器指令 参考页数 641
TRUNC 参考页数 115	算术指令 参考页数 73
TTABLE 参考页数 170	布尔指令 参考页数 159
UIC 参考页数 441	中断指令 参考页数 439
UID 参考页数 443	
UIE 参考页数 445	
UIF 参考页数 447	
XOR_MASK 参考页数 155	二进制指令 参考页数 141

运算符

下表按功能类别列出了各种运算符。

指令	功能类别
Addition 参考页数 79	算术指令 参考页数 73
AND 参考页数 166	布尔指令 参考页数 159
ANY TO BOOL 参考页数 294	数据转换指令 参考页数 293
ANY TO BYTE 参考页数 295	
ANY TO DATE 参考页数 296	
ANY TO DINT 参考页数 297	
ANY TO DWORD 参考页数 298	
ANY TO INT 参考页数 299	
ANY TO LINT 参考页数 300	
ANY TO LREAL 参考页数 301	
ANY TO LWORD 参考页数 302	
ANY TO REAL 参考页数 303	
ANY TO SINT 参考页数 304	
ANY TO STRING 参考页数 305	
ANY TO TIME 参考页数 307	
ANY TO UDINT 参考页数 308	
ANY TO UINT 参考页数 310	
ANY TO ULINT 参考页数 311	
ANY TO USINT 参考页数 312	
ANY TO WORD 参考页数 313	
Division 参考页数 91	算术指令 参考页数 73
等于 参考页数 275	比较指令 参考页数 275
大于 参考页数 277	
大于或等于 参考页数 279	
小于 参考页数 280	
小于或等于 参考页数 281	
MOV 参考页数 98	算术指令 参考页数 73
Multiplication 参考页数 99	
取反 参考页数 100	
非 参考页数 167	布尔指令 参考页数 159
不等于 参考页数 282	比较指令 参考页数 275
OR 参考页数 165	布尔指令 参考页数 159

指令	功能类别
Addition 参考页数 79	算术指令 参考页数 73
Subtraction 参考页数 110	算术指令 参考页数 73
XOR 参考页数 167	布尔指令 参考页数 159

另请参见

[查找有关指令和梯形元素的信息](#) 参考页数 [15](#)

[按字母顺序排列的指令集](#) 参考页数 [20](#)

梯形图 (LD) 语言

梯形图语言参考

元素	说明
LD 编程 参考页数 35	布尔方程的图形表示使用图形符号，将触点（输入参数）与线圈（输出结果）相结合。
LD 程序开发环境 参考页数 36	展示用于梯形图 (LD) 程序的语言编辑器的示例。
梯形图 (LD) 元素 参考页数 39	用于生成梯形图程序的组件。
LD 程序中的指令块 参考页数 61	符合 IEC 61131-3 标准的指令块总的来说包括功能块、函数和运算符。
使用 LD 语言编辑器 参考页数 62	向程序中添加元素。
梯形图 (LD) 程序示例 参考页数 63	梯形图 (LD) 程序的示例。
LD 键盘快捷键 参考页数 65	梯形图 (LD) 语言可用的键盘快捷键。

LD 编程

梯形图 (LD) 为布尔方程的图形表示，它将接触（输入参数）与线圈（输出结果）相结合。LD 语言通过程序图表（像继电器梯形布线图那样组织）中的图形符号，介绍了对布尔数据的测试和修改。

LD 图形符号在作为电接触图的图表内进行组织。“梯形”一词来源于两端均与垂直电源导轨相连的梯级概念，其中每个梯级均表示单个回路。

Connected Components Workbench™ 支持梯形图 (LD)

Connected Components Workbench™ 提供 LD 语言编辑器，仅支持 Connected Components Workbench™ 软件随附的元素和指令。

另请参见

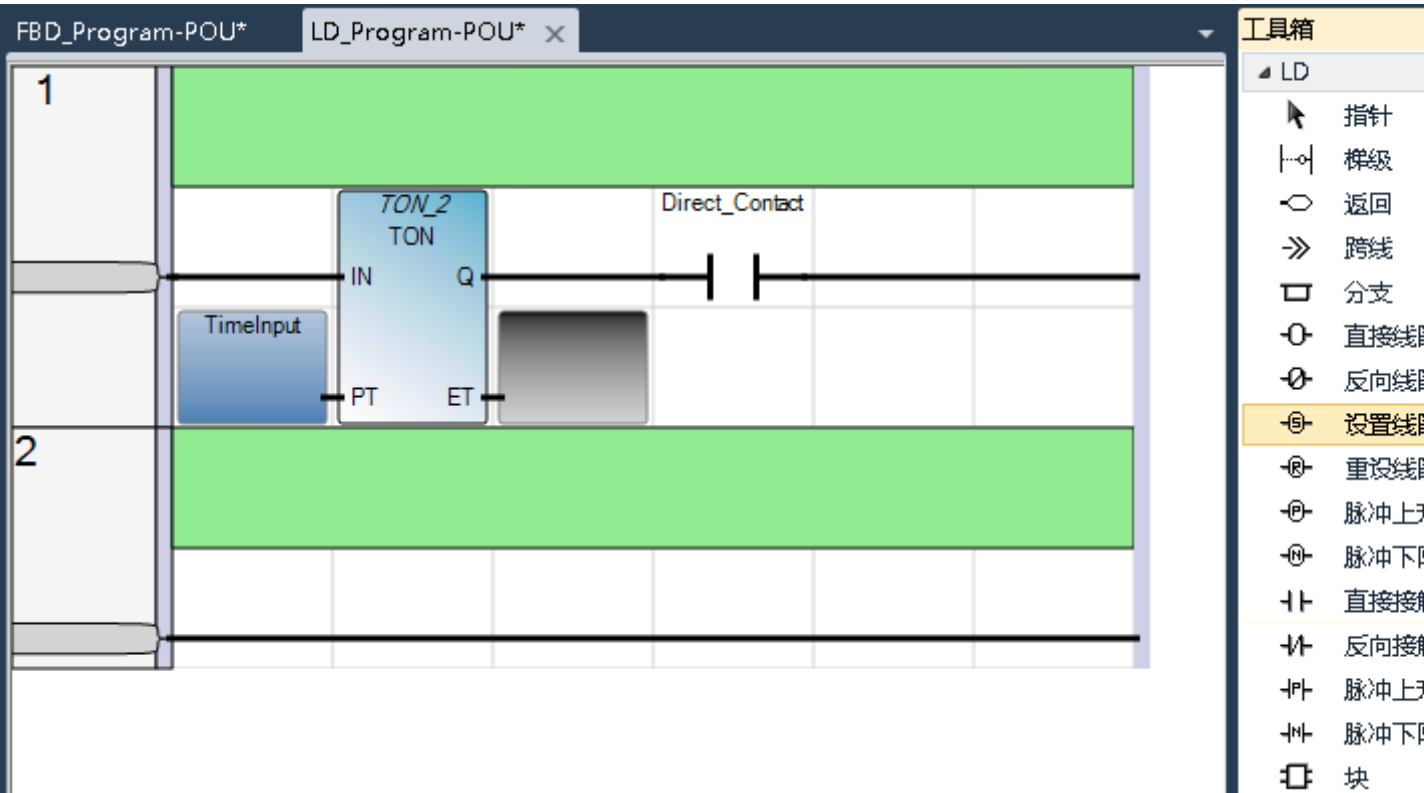
- [LD 程序开发环境](#) 参考页数 36
- [在 LD 语言编辑器中处理元素](#) 参考页数 62
- [梯形图 \(LD\) 元素](#) 参考页数 39

LD 工具箱元素

- [梯形图 \(LD\) 程序示例](#) 参考页数 63

LD 程序开发环境

下图显示的是用于 LD 程序的语言编辑器，您可以在上面开发 LD 程组织单元 (POU)。使用 LD 工具箱或 LD 键盘快捷键向 LD POU 添加元素。



另请参见

- [梯形图 \(LD\) 语言](#) 参考页数 35
- [梯形图 \(LD\) 元素](#)

[在 LD 语言编辑器中处理元素](#) 参考页数 [62](#)

[梯形图 \(LD\) 程序示例](#) 参考页数 [63](#)

[LD 键盘快捷键](#) 参考页数 [65](#)

梯形图 (LD) 元素

梯形图元素是用于生成梯形图程序的组件。下表中列出的所有元素都可通过 Connected Components Workbench 中的 LD 工具箱添加到梯形图。

元素	说明
梯级 参考页数 39	表示导致线圈被激活的一组回路元素。
块 (LD) 参考页数 43	指令包括运算符、函数和功能块（包括用户定义的功能块）。
分支 参考页数 42	两个或多个并行指令。
线圈 参考页数 48	表示输出或内部变量的赋值。在 LD 程序中，线圈表示操作。
触点 参考页数 55	表示输入或内部变量的值或函数。
返回 参考页数 59	表示功能块图输出的条件结束。
跳转 参考页数 60	表示控制梯形图执行的 LD 程序中的条件逻辑和无条件逻辑。

另请参见

[LD 编程](#) 参考页数 [35](#)

梯级

梯级为 LD 图的图形组件，表示导致线圈被激活的一组回路元素。梯级可以具有标签，标签用于在图内标识梯级。这些标签与跳转一起控制梯形图的执行。您可以在梯级的上方输入注释（任意格式的文本），仅用于存档。

更改默认梯级宽度

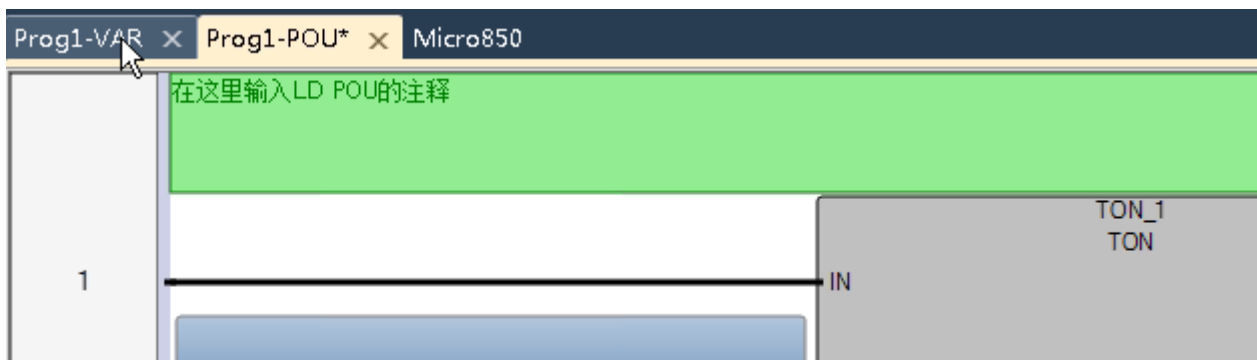
按照以下步骤使用梯级的新宽度。在项目中无法调整现有梯级的宽度。

1. 在工具菜单中，选择选项。
2. 单击 IEC 语言，然后单击梯形图。
3. 在视图设置下，单击默认单元格宽度。
4. 增大单元格宽度值，然后单击确定。
5. 创建一个新的 LD 程序并将其打开。
6. 按下 **Ctrl** 键的同时，向下滚动鼠标上的指轮，直到您的计算机上显示整个梯级为止。

梯级注释

在梯级上方的空间中输入的注释以 RTF 格式保存并存储在控制器中。

1. 在语言编辑器中，双击梯级上方的矩形区域，然后键入相应注释。
2. 单击语言编辑器工作区中的任意位置以保存注释。



向梯级添加标签

语言编辑器中的每个梯级的左侧都有一个区域，用于输入标签名称。

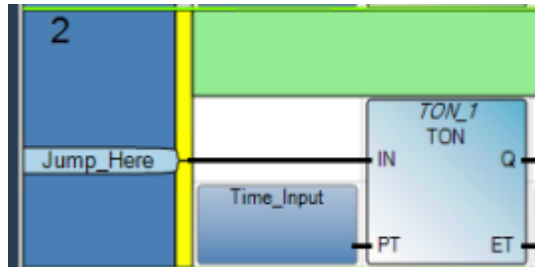
标签名称要求

标签的字符数不受限制，以字母或下划线字符开头，后跟字母、数字和下划线字符。标签中不能有空格或特殊字符（例如“+”、“-”或“\”）。

要为梯级添加标签，请执行以下操作

- 单击梯级左侧的区域，然后键入标签名称。

示例：标签



另请参见

[梯级](#) 参考页数 [39](#)

另请参见

LD 工具箱元素

[LD 编程](#) 参考页数 [35](#)

向 LD 程序添加元素

要向 LD 语言编辑器中添加梯级，可执行以下操作之一：

- 将梯级元素从工具箱拖动到语言编辑器中，或
- 在工具箱中，双击相应的梯级元素以将其添加到语言编辑器，或
- 右击现有梯级，选择复制，然后将相应梯级的副本粘贴到语言编辑器。

提示

将鼠标悬停在有效的目标后，工具箱元素上方会出现一个加号 (+)。松开鼠标键可添加元素。

提示

可使用键盘快捷键向 LD 程序添加元素。请参见 [LD 键盘快捷键](#) 参考页数 65。

提示

如果梯形图中包含的梯级超过 355 个，则使用倒三角形而不是滚动条查看附加梯级。



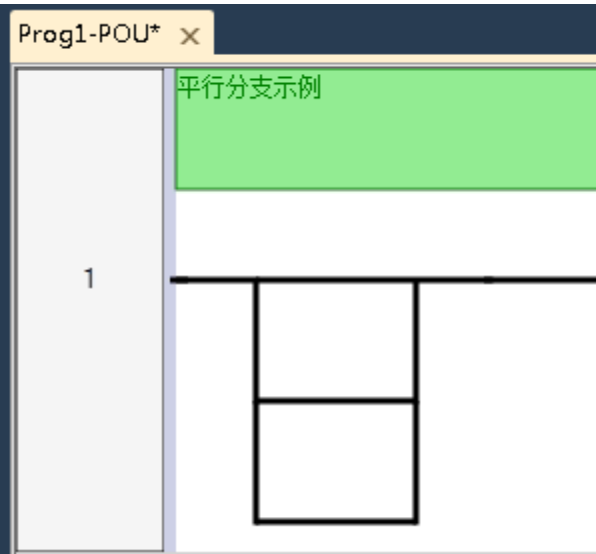
分支

分支可创建连接的备用路由。您可以将平行分支添加到梯级上的相应元素中。

添加分支

在工具箱中，将分支元素拖动到语言编辑器内的现有元素上。

示例：分支



另请参见

LD 工具箱元素

[LD 编程](#) 参考页数 [35](#)

向梯级添加标签

标签对于语言编辑器中的每个梯级来说都是可选添加项。

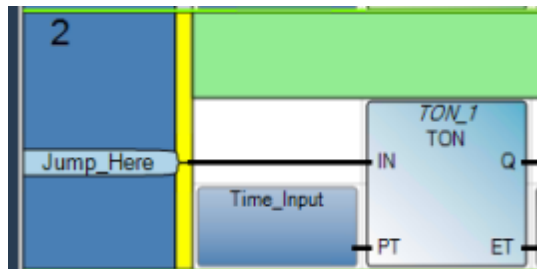
标签名称要求

标签的字符数不受限制，以字母或下划线字符开头，后跟字母、数字和下划线字符。标签中不能有空格或特殊字符（例如“+”、“-”或“\”）。

要为梯级添加标签，请执行以下操作

- 单击梯级左侧的区域，然后键入标签名称。

示例：标签



另请参见

[梯级](#) 参考页数 [39](#)

指令块 (LD)

LD 指令块元素是 LD 图中的一个功能元素，它可以是功能块、函数、用户定义的功能块或运算符。

另请参见

[LD 编程](#) 参考页数 [35](#)

LD 工具箱元素

LD 指令块

Connected Components Workbench 指令集包括符合 IEC 61131-3 标准的指令块。指令块总的来说包括运算符、函数和功能块。

向 LD 编程添加指令块

使用 LD 工具箱向 LD 编程添加块。在添加块后，可以通过“模块选择器”或“变量选择器”配置其变量。

按照以下步骤向 LD 编程添加块元素或更改现有块元素的类型。

向编程添加块元素

1. 从工具箱中，将块元素拖动到语言编辑器，然后将其放入正确的位置以显示“模块选择器”。
2. 在“搜索”中，键入您要添加或排序的块的名称，并滚动列表以找到它。
3. 双击块以将其添加到编程。

更改块元素类型

1. 在语言编辑器中，双击块以显示“模块选择器”。
2. 在“搜索”中，键入要排序的块的名称，并滚动列表以找到它。
3. 双击块对其进行更新。

启用 EN/ENO

您可以启用 EN 输入参数和 ENO 输出参数，以便始终通过指令块添加它们，即使有可用的布尔输入或输出也是如此。

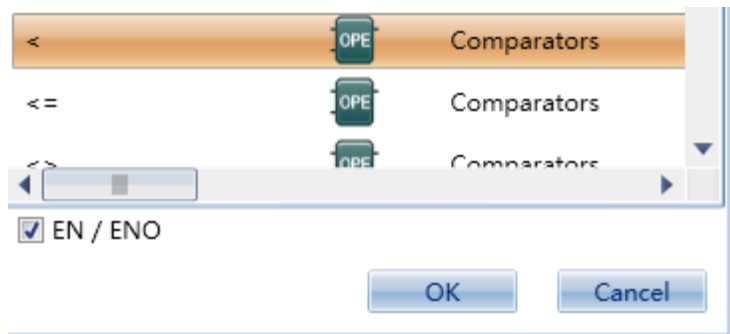
注意：EN 和 ENO 参数将仅添加到启用设置后添加的指令块，已在 LD 程序中的指令块不会受影响。

要针对添加到 LD 程序的所有块启用 EN/ENO，请执行以下操作

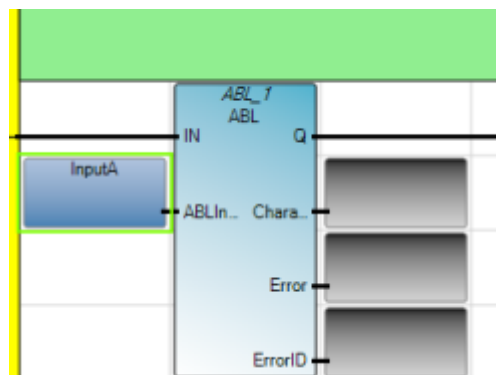
1. 在工具菜单中，单击选项。
2. 选择 IEC 语言 > 梯形图 (LD) 以显示语言编辑器属性。
3. 在块设置中，将启用 EN/ENO 设置为真。
4. 单击确定。

要针对块启用 EN/ENO，请执行以下操作

1. 在“模块选择器”中选择一个块后，选择 EN/ENO(位于列表底部)。
2. 在“模块选择器”中，在指令列表底部进行选择后选择 EN/ENO。



示例：指令块 (LD)



使用 LD 指令块中的 **enable** 输入和 **enable** 输出

LD 图中的梯级状态始终为布尔状态，并将块的第一个输入和第一个输出连接至梯级。

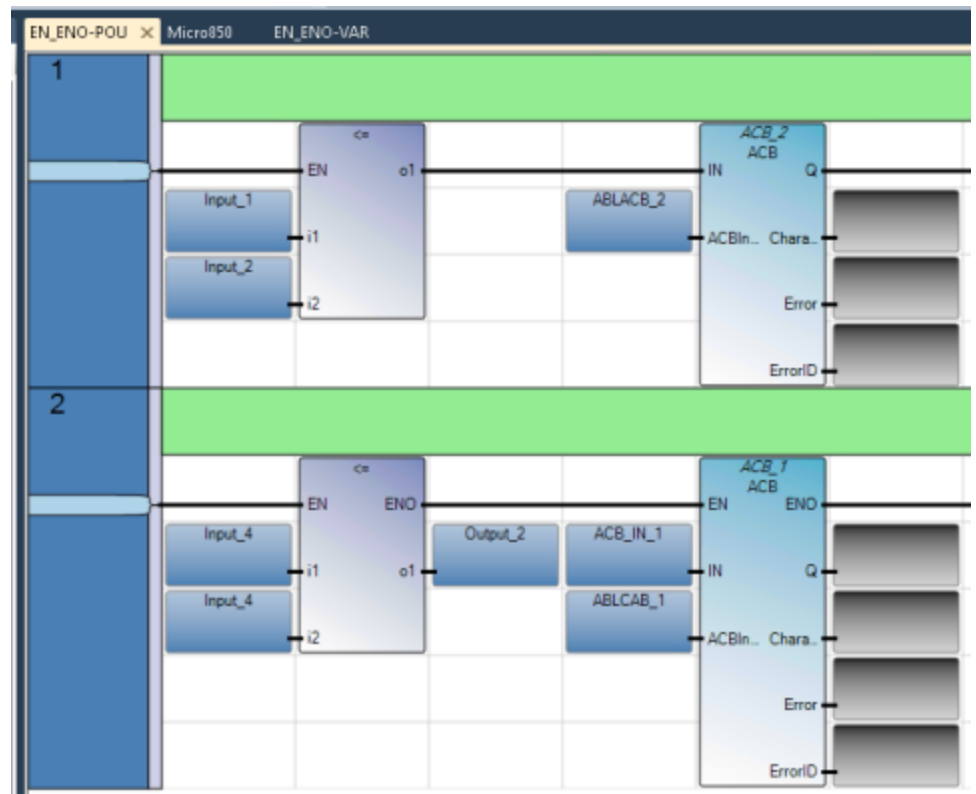
EN 输入/EN 输出

如果第一个布尔输入或输出不存在，则会将 EN 和/或 ENO 参数添加到块。

- 如果第一个块输入不是布尔输入，则将 EN 输入参数添加到块。仅当 EN 输入为“真”时，才会执行指令块。
- 如果第一个块输出不是布尔输出，则将 ENO 输出参数添加到块。ENO 输出的状态始终与指令块的首个输入相同。

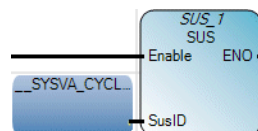
可以在“模块选择器”或在梯形图 (LD) 程序的“块设置”中为单个指令块启用/禁用 EN/ENO 块设置。下表介绍在图中显示的块中启用和禁用 EN/ENO 参数的结果。

梯级	EN/ENO	块	第一个布尔输入	是否已添加 EN 输入？	第一个布尔输出	是否已添加 ENO 输出？
1	False	<=	不支持	支持	支持	不支持
		ACB	支持	不支持	支持	不支持
2	真	<=	不支持	支持	支持	支持
		ACB	支持	支持	支持	支持



示例：启用输入

某些情况下，对于待命执行的指令块，需要提供启用参数。以下示例显示的是带有 Enable 输入的 SUS 指令块。



另请参见

[梯形图 \(LD\) 语言](#) 参考页数 [35](#)

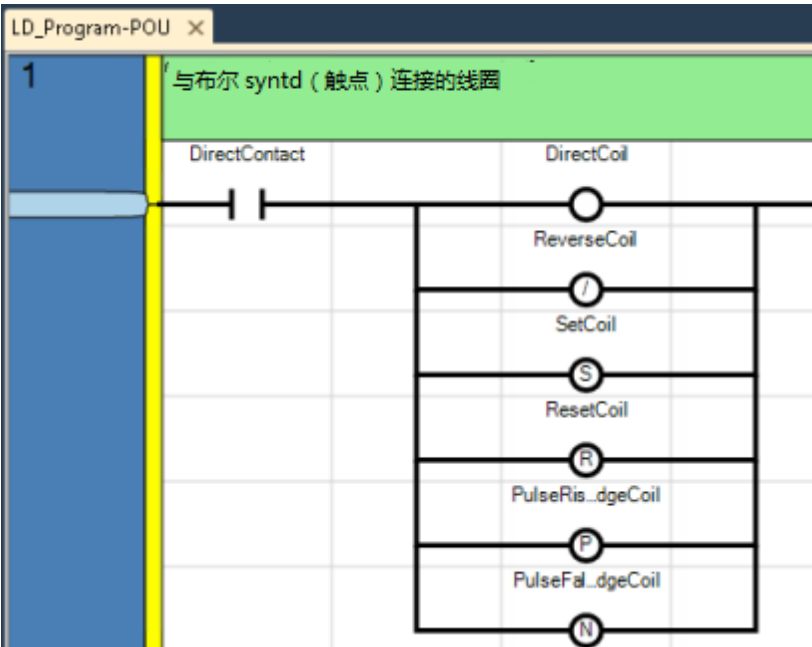
线圈

线圈是 LD 图的图形组件，表示输出或内部变量的赋值。在 LD 图中，线圈表示操作。线圈的左端必须与一个布尔符号（如接触或指令块的布尔输出）相连。因此，线圈只能添加到在 LD 语言编辑器中定义的梯级。在添加线圈后，可以修改其定义。

可以从“工具箱”向 LD 程序添加以下线圈元素类型。

线圈元素	说明
直接线圈 参考页数 50	直接线圈支持连接线布尔状态的布尔输出。
反向线圈 参考页数 51	反向线圈根据连接线状态的布尔非运算结果支持布尔输出。
脉冲上升沿的线圈 参考页数 52	脉冲上升沿（或正极）的线圈支持连接线布尔状态的布尔输出。
脉冲下降沿的线圈 参考页数 51	脉冲下降沿（或负值）的线圈支持连接线布尔状态的布尔输出。
设置线圈 参考页数 53	设置线圈支持连接线布尔状态的布尔输出。
重设线圈 参考页数 54	重设线圈支持连接线布尔状态的布尔输出。

示例：线圈



另请参见

[添加线圈元素](#) 参考页数 [49](#)

LD 工具箱元素

[LD 编程](#) 参考页数 [35](#)

添加线圈元素

按照以下步骤添加和修改线圈元素。

添加线圈元素

1. 验证 LD 程序是否已为线圈定义梯级。
2. 从工具箱中，将线圈拖动到 LD 语言编辑器，放置在布尔符号或布尔输出的右侧。
3. 为线圈分配变量。

提示 将鼠标悬停在有效的目标后，工具箱元素上方会出现一个加号 (+)。松开鼠标键可添加元素。

提示 可使用键盘快捷键向 LD 程序添加元素。请参见 [LD 键盘快捷键](#) 参考页数 [65](#)。

插入并联线圈

1. 从工具箱中，将相应分支元素拖动到语言编辑器中，并将其放置在梯级元素的稍上方。
2. 从工具箱中，将相应线圈元素拖动到语言编辑器中，并将其放置在相应分支元素上，以便在分支上显示该线圈。

更改线圈的类型

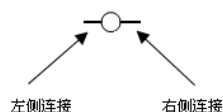
在语言编辑器中，选择相应的线圈，然后按住空格键，直到新的线圈类型可用。每按一次空格键，类型会从正向更改为反向、设置、重置、脉冲上升沿、脉冲下降沿。

另请参见

[线圈](#) 参考页数 [48](#)

直接线圈

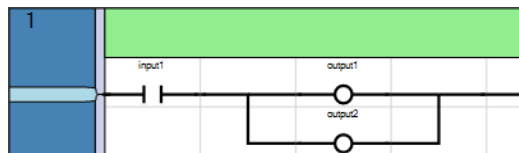
直接线圈支持连接线布尔状态的布尔输出。



关联变量被赋以左侧连接的布尔状态。左侧连接的状态将传播至右侧连接。右侧连接必须与右侧垂直电源导轨相连 (除非您采用的是并联线圈, 这种情况下仅上方的线圈必须与右侧垂直电源导轨相连)。

关联的布尔变量必须为输出变量或用户定义的变量。

示例: 直接线圈



另请参见

[添加线圈元素](#) 参考页数 [49](#)

[线圈](#) 参考页数 [48](#)

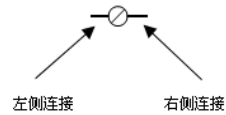
[反向线圈](#) 参考页数 [51](#)

LD 工具箱元素

[梯形图 \(LD\) 语言](#) 参考页数 [35](#)

反向线圈

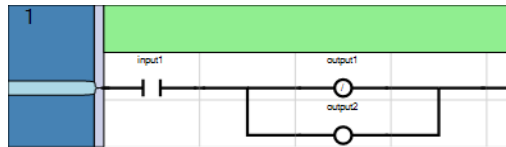
反向线圈元素根据连接线状态的布尔非运算结果支持布尔输出。



关联变量的值为左侧连接线状态的布尔非运算结果。左侧连接的状态将传播至右侧连接。右侧连接必须与右侧垂直电源导轨相连（除非您采用的是并联线圈，这种情况下仅上方的线圈必须与右侧垂直电源导轨相连）。

关联的布尔变量必须为输出变量或用户定义的变量。

示例：反向线圈



另请参见

[添加线圈元素](#) 参考页数 [49](#)

[线圈](#) 参考页数 [48](#)

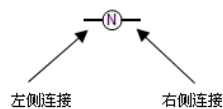
[直接线圈](#) 参考页数 [50](#)

LD 工具箱元素

[梯形图 \(LD\) 语言](#) 参考页数 [35](#)

脉冲下降沿的线圈

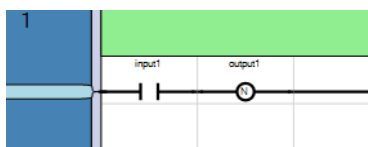
脉冲下降沿（或负值）的线圈支持连接线布尔状态的布尔输出。



左侧连接的布尔状态从“真”下降为“假”时，关联的变量将置为“真”。输出变量在所有其他情况下都将重置为“假”。左侧连接的状态将传播至右侧连接。右侧连接必须与右侧垂直电源导轨相连（除非您采用的是并联线圈，这种情况下仅上方的线圈必须与右侧垂直电源导轨相连）。

关联的布尔变量必须为输出变量或用户定义的变量。

示例：脉冲下降沿的线圈



另请参见

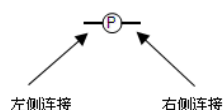
[线圈](#) 参考页数 [48](#)

[添加线圈元素](#) 参考页数 [49](#)

[脉冲上升沿的线圈](#) 参考页数 [52](#)

脉冲上升沿的线圈

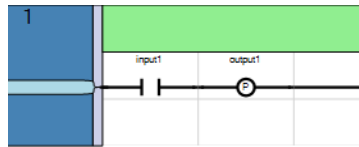
脉冲上升沿（或正极）的线圈支持连接线布尔状态的布尔输出。



左侧连接的布尔状态从“假”上升为“真”时，关联的变量将置为“真”。输出变量在所有其他情况下都将重置为“假”。左侧连接的状态将传播至右侧连接。右侧连接必须与右侧垂直电源导轨相连（除非您采用的是并联线圈，这种情况下仅上方的线圈必须与右侧垂直电源导轨相连）。

关联的布尔变量必须为输出变量或用户定义的变量。

示例：脉冲上升沿的线圈



另请参见

[线圈](#) 参考页数 [48](#)

[添加线圈元素](#) 参考页数 [49](#)

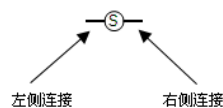
[脉冲下降沿的线圈](#) 参考页数 [51](#)

LD 工具箱元素

[梯形图 \(LD\) 语言](#) 参考页数 [35](#)

设置线圈

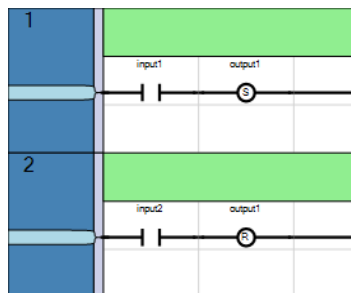
设置线圈支持连接线布尔状态的布尔输出。



左侧连接的布尔状态置为“真”时，关联的变量将置为“真”。输出变量将一直保持此值，直到复位线圈发出反向指令为止。左侧连接的状态将传播至右侧连接。右侧连接必须与右侧垂直电源导轨相连（除非您采用的是并联线圈，这种情况下仅上方的线圈必须与右侧垂直电源导轨相连）。

关联的布尔变量必须为输出变量或用户定义的变量。

示例：设置线圈



另请参见

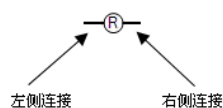
[线圈](#) 参考页数 48

[添加线圈元素](#) 参考页数 49

[重设线圈](#) 参考页数 54

重设线圈

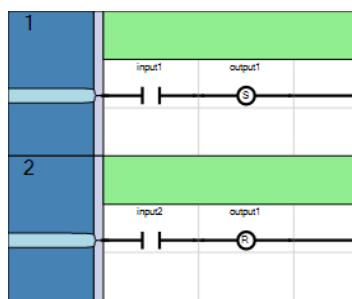
重设线圈支持连接线布尔状态的布尔输出。



左侧连接的布尔状态置为“真”时，关联的变量将重置为“假”。输出变量将一直保持此值，直到置位线圈发出反向指令为止。左侧连接的状态将传播至右侧连接。右侧连接必须与右侧垂直电源导轨相连（除非您采用的是并联线圈，这种情况下仅上方的线圈必须与右侧垂直电源导轨相连）。

关联的布尔变量必须为输出变量或用户定义的变量。

示例：重设线圈



另请参见

[线圈](#) 参考页数 48

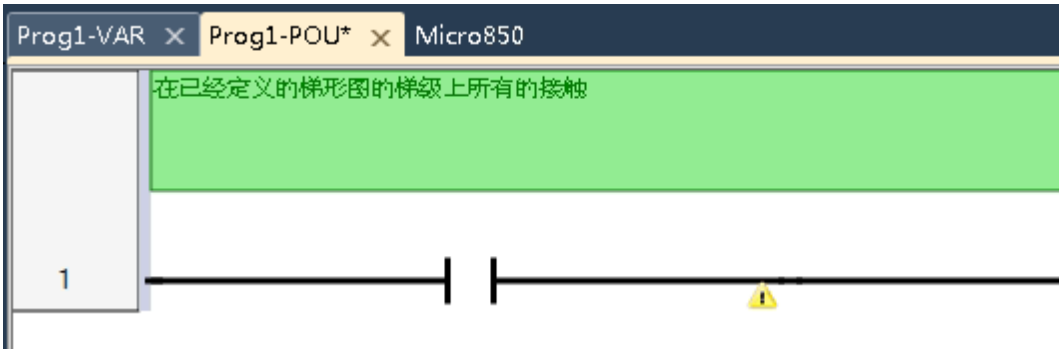
[添加线圈元素](#) 参考页数 49

[设置线圈](#) 参考页数 53

触点

接触是 LD 图的图形组件。根据类型不同，接触表示输入或内部变量的值或函数。接触只能添加到在 LD 语言编辑器中定义的梯级。在添加接触后，可以修改其定义。

示例：接触



可从 Connected Components Workbench 中的 LD 工具箱将以下接触元素类型添加到 LD 程序。

接触元素	说明
直接接触 参考页数 56	直接接触支持在连接线状态与布尔变量之间进行布尔运算。
反向接触 参考页数 57	反向接触支持对布尔变量进行布尔非运算后再与连接线状态进行布尔运算。
脉冲上升沿触点 参考页数 58	脉冲上升沿（或正向）接触支持在连接线状态与布尔变量的上升沿之间进行布尔运算。
脉冲下降沿触点	脉冲下降沿(负值)接触用于在连接线状态与布尔变量下降沿之间进行布尔运算。

另请参见

LD 工具箱元素

[LD 编程](#) 参考页数 [35](#)

添加接触元素

按照以下步骤向梯形图梯级添加接触元素或更改使用的接触类型。

添加接触元素

1. 验证 LD 程序是否已为接触定义梯级。
2. 从工具箱中，将相应的接触元素拖动到语言编辑器中，并将其放置在梯级上。

提示 将鼠标悬停在有效的目标后，工具箱元素上方会出现一个加号 (+)。松开鼠标键可添加元素。

提示 可使用键盘快捷键向 LD 程序添加元素。请参见 [LD 键盘快捷键](#) 参考页数 [65](#)。

插入并联接触

1. 从工具箱中，将相应分支元素拖动到语言编辑器中，并将其放置在现有接触上。
2. 从工具箱中，将相应接触元素拖动到语言编辑器中，并将其放置在相应分支上。

更改接触的类型

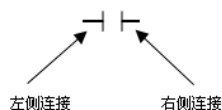
在语言编辑器中，选择相应触点，然后按住空格键，直到您要使用的接触类型显示在语言编辑器中。

另请参见

[接触](#) 参考页数 [55](#)

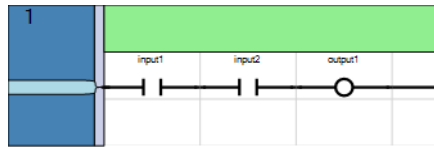
直接接触

直接接触支持在连接线状态与布尔变量之间进行布尔运算。



接触右侧连接线的状态是左侧连接线的状态与接触所关联变量的值之间进行逻辑“与”运算后得到的结果。

示例：直接接触



另请参见

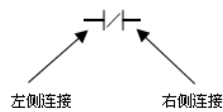
[接触](#) 参考页数 [55](#)

[添加接触元素](#) 参考页数 [55](#)

[反向接触](#) 参考页数 [57](#)

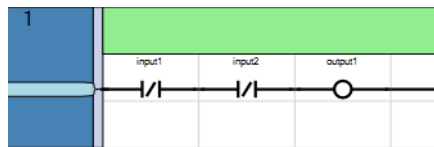
反向接触

反向接触支持对布尔变量进行布尔非运算后再与连接线状态进行布尔运算。



接触右侧连接线的状态是对触点所关联变量的值进行布尔非运算后再与左侧连接线的状态进行逻辑“与”运算后得到的结果。

示例：反向接触



另请参见

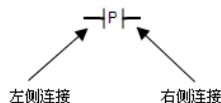
[接触](#) 参考页数 [55](#)

[添加接触元素](#) 参考页数 [55](#)

[直接接触](#) 参考页数 [56](#)

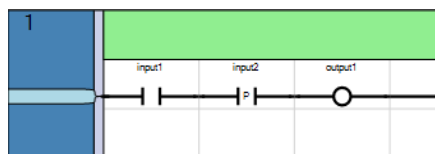
脉冲上升沿触点

脉冲上升沿（或正向）接触支持在连接线状态与布尔变量的上升沿之间进行布尔运算。



在左侧连接线的状态为“真”，所关联变量的状态由“假”上升为“真”时，接触右侧的连接线的状态将设为“真”。该状态在所有其他情况下都将重置为“假”。

示例：脉冲上升沿触点



建议：限制使用具有边沿触点的输出变量

建议您不要使用具有脉冲上升沿触点（正值）或脉冲下降沿触点（负值）的输出或变量。这些触点用于梯形图中的物理输入。如果您要检测变量或输出的沿，建议您使用 R_TRIG/F_TRIG 功能块，它支持所有语言，并且可在您程序中的任何位置使用。

另请参见

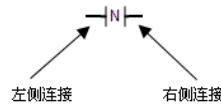
[接触](#) 参考页数 [55](#)

[添加接触元素](#) 参考页数 [55](#)

[脉冲下降沿触点](#) 参考页数 [59](#)

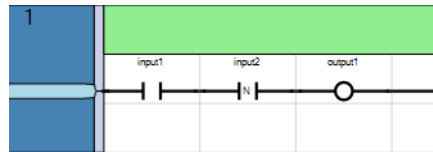
脉冲下降沿触点

脉冲下降沿（或负值）接触支持在连接线状态与布尔变量的下降沿之间进行布尔运算。



左侧连接线的状态为“真”，所关联变量的状态由“真”下降为“假”时，接触右侧连接线的状态将设为“真”。该状态在所有其他情况下都将重置为“假”。

示例：脉冲下降沿触点



建议：限制使用具有边沿触点的输出变量

建议您不要使用具有脉冲上升沿触点（正值）或脉冲下降沿触点（负值）的输出或变量。这些触点用于梯形图中的物理输入。如果您要检测变量或输出的沿，建议您使用 R_TRIG/F_TRIG 功能块，它支持所有语言，并且可在您程序中的任何位置使用。

另请参见

[接触](#) 参考页数 [55](#)

[添加接触元素](#) 参考页数 [55](#)

[脉冲上升沿触点](#) 参考页数 [58](#)

返回

返回是表示梯形图的条件性结束的输出。

提示 不能将连接放置在返回元素的右侧。

:

当左侧连接线的布尔状态为“真”时，图结束，而不会执行位于图的下几条线上的指令。

当 LD 图为函数时，其名称与用于设置返回值（返回到调用图）的输出线圈关联。

示例：返回



插入返回

在工具箱中，将返回元素拖动到语言编辑器中，并将其放置在梯级上。

另请参见

LD 工具箱元素

[LD 编程](#) 参考页数 [35](#)

跳转

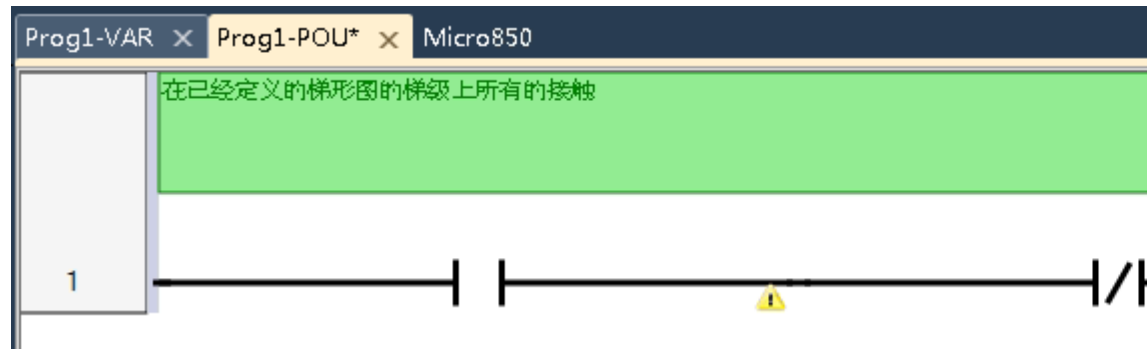
跳转是控制 LD 图执行的条件元素或无条件元素。

跳转符号

下面的符号表示跳转到 LAB 标签：

>>LAB - 跳转至标签（标签名称为“LAB”）

示例：跳转



另请参见

LD 工具箱元素

[LD 编程](#) 参考页数 [35](#)

[返回](#) 参考页数 [59](#)

LD 程序中的指令块

Connected Components Workbench 指令集包括符合 IEC 61131-3 标准指令块。指令块总的来说包括功能块、函数和运算符。可以将指令块的输入和输出连接到变量、触点、线圈或其他指令块的输入和输出。

指令块规范

IEC61131-3 编程语言规范针对可编程控制器的各个方面,包括操作系统执行、数据定义、编程语言以及指令集。IEC61131-3 规范提供可根据最终用户应用程序进行扩展的最小功能集合。

指令块名称

函数和功能块由一个显示指令名称和参数短名称的框表示。对于功能块,实例名称以斜体显示。



指令块返回参数

- 函数的返回参数的名称与该函数相同。返回参数是唯一的输出。
- 功能块返回参数的名称不受限制。多个返回参数可以提供多个输出。
- 可以通过在“参数”视图中显示的各个设备的选项卡之间进行导航来为多个设备定义程序的参数。

使用 LD 语言编辑器 如何将元素添加到梯形图

在 LD 程序中将项目添加到梯级时，根据以下条件进行添加。

- 梯级的第一个元素插入到在梯形图中选择的位置。
- 并将后续元素插入到梯级上选定项的右侧。
- 如果该元素对于当前梯级而言过大，则将它放置到下一个梯级。

向 LD 程序添加元素

从 LD 工具箱中，将相应的元素拖放到 LD 语言编辑器中，并将其放置在梯级上。

提示 将鼠标悬停在有效的目标后，工具箱元素上方会出现一个加号 (+)。松开鼠标键可添加元素。

提示 可使用键盘快捷键向 LD 程序添加元素。请参见 [LD 键盘快捷键](#) 参考页数 65。

替换分配给元素的变量

您可以直接从语言编辑器或“变量选择器”替换分配的变量。

要从语言编辑器修改变量，请执行以下操作

1. 在语言编辑器中，单击变量名称以显示全局变量和局部变量的下拉列表。
2. 请执行以下操作之一：

- 在文本框中键入新的变量名称。
-或-
- 从下拉列表中选择一个不同的变量名称。

要从“变量选择器”修改变量，请执行以下操作

1. 在语言编辑器中，双击变量以打开变量选择器。
2. 单击变量名称，然后从全局变量和局部变量的下拉列表选择一个不同的变量。
3. 单击现有变量，然后在提供的文本框中键入常量值。

另请参见

[梯形图 \(LD\) 语言](#) 参考页数 [35](#)

[梯形图 \(LD\) 程序示例](#) 参考页数 [63](#)

[LD 键盘快捷键](#) 参考页数 [65](#)

梯形图 (LD) 程序示例

请参见梯形图 (LD) 程序的以下示例。

[示例: R_TRIG 功能块](#) 参考页数 [63](#)

[示例: 使用 Subtraction \(-\)、ABS 和 Less than \(<\) 比较实型值](#) 参考页数 [64](#)

另请参见

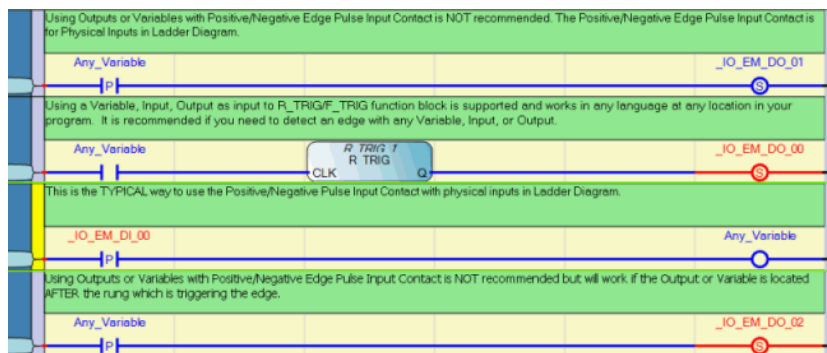
[梯形图 \(LD\) 语言](#) 参考页数 [35](#)

[LD 程序开发环境](#) 参考页数 [36](#)

[在 LD 语言编辑器中处理元素](#) 参考页数 [62](#)

示例：R_TRIG 功能块

以下示例程序展示了我们推荐您使用 R_TRIG 功能块在连接至控制器时检测沿的方法。

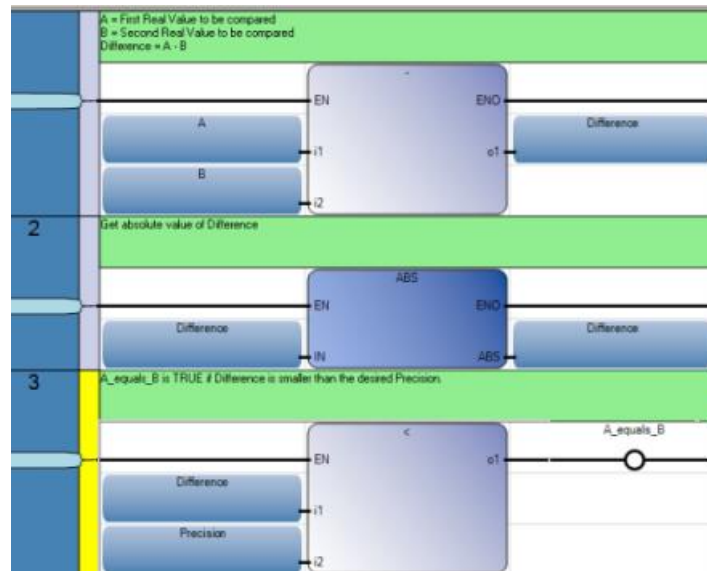


示例：使用 Subtraction (-)、ABS 和 Less than (<) 比较实型值

比较各值是否相等时，不建议使用实型数据类型，因为这比较的是经过舍入的数字。两个输出值在 Connected Components Workbench 画面中可能显示为相等，但评估为 FALSE。

例如，在变量输入画面中，23.500001 与 23.499999 都将显示为 23.5，但在控制器中则不相等。

要测试两个实型数据类型的值是否相等，您可以使用相减指令得出两值之差，然后判断该差值是否小于设定的精度值。有关比较两个实型数据类型的值的信息，请参见以下 LD 程序示例。



LD 键盘快捷键

以下键盘快捷键可用于 LD 语言。

快捷键	说明
Ctrl+0	将梯级插入到选定梯级之后。 ¹
Ctrl+Alt+0	将梯级插入到选定梯级之前。 ¹
Ctrl+ 1	将分支插入到选定元素之后。
Ctrl+Alt+ 1	将分支插入到选定元素之前。
Ctrl+2	将指令块插入到选定元素之后。 ²
Ctrl+Alt+2	将指令块插入到选定元素之前。 ²
Ctrl+3	将触点插入到选定元素之后。 ²
Ctrl+Alt+3	将触点插入到选定元素之前。 ²
Ctrl+4	将线圈插入到选定元素之后**。
Ctrl+Alt+4	将线圈插入到选定元素之前**。
Ctrl+5	将跳转插入到选定元素之后**。
Ctrl+Alt+5	将跳转插入到选定元素之前**。
Ctrl+6	将返回插入到选定元素之后**。

快捷键	说明
Ctrl+Alt+6	将返回插入到选定元素之后**。
空格键	针对线圈或触点，在可用类型之间切换。
Shift+Ctrl+Alt+G	选择/清除语言编辑器工作区中的网格。
Ctrl+下箭头键	移至下一个梯级。在某梯级上选定一个元素后，移至包含该选定元素的下一个梯级。
Ctrl+R	打开块选择器。
Ctrl+上箭头键	移至上一个梯级。在某梯级上选定一个元素后，移至包含该选定元素的上一个梯级。
向下键、向上键、向右键、向左键	在分支和子分支之间移动。
删除	删除选定的梯级或元素。
Enter	调用变量/模块选择器（具体取决于选定的元素）。
Shift+Enter	插入换行符
Ctrl+Enter	打开当前行之上的行
Ctrl+Shift+Enter	打开当前行之下的行
Ctrl+Shift+L	删除当前行
Ctrl+Delete	删除当前行中的下一个字
退格键	删除左侧的字符
Ctrl+退格键	删除当前行中的上一个字
Ctrl+C	将选定文本复制到剪贴板
Ctrl+Insert	将选定文本复制到剪贴板
Ctrl+L	将当前行剪切到剪贴板
Ctrl+X	将选定文本剪切到剪贴板
Shift+Delete	将选定文本剪切到剪贴板
Ctrl+V	将保存在剪贴板中的文本粘贴到插入点
Shift+Insert	将保存在剪贴板中的文本粘贴到插入点
Ctrl+Z	撤销上一个命令
Ctrl+Y	恢复上一个命令
Ctrl+Shift+Z	恢复上一个命令
Ctrl+向左箭头	移动到上一个语句或字
Ctrl+向右箭头	移动到下一个语句或字
Home	移动到行首

快捷键	说明
结束	移动到行末
Ctrl+Home	移动到文档开头
Ctrl+End	移动到文档末尾
Page Up	移动到可见代码的顶部
Page Down	移动到可见代码的底部
Ctrl+Page Up	移动到可见代码的顶部
Ctrl+Page Down	移动到可见代码的底部
Ctrl+J	移动到配对的括号
Ctrl+向下箭头	向下滚动
Ctrl+向上箭头	向上滚动
Shift+向下箭头	选择下方
Shift+向上箭头	选择上方
Shift+向左箭头	选择左侧
Shift+向右箭头	选择右侧
Ctrl+Shift+向左箭头	选择到上一个语句或字
Ctrl+Shift+向右箭头	选择到下一个语句或字
Shift+Home	从插入点选择到行首
Shift+End	从插入点选择到行末
Ctrl+Shift+Home	从插入点选择到文档开头
Ctrl+Shift+End	从插入点选择到文档末尾
Shift+Page Up	从插入点选择到可见代码的顶部
Shift+Page Down	从插入点选择到可见代码的底部
Ctrl+Shift+Page Up	从插入点选择到可见代码的顶部
Ctrl+Shift+Page Down	从插入点选择到可见代码的底部
Ctrl+A	选择整个文档
Ctrl+Shift+W	选择下一个字
Ctrl+Shift+J	选择到配对的括号
Shift+Alt+向下箭头	选择当前行和下一行
Shift+Alt+向上箭头	选择当前行和上一行
Shift+Alt+向左箭头	选择当前行的左侧
Shift+Alt+向右箭头	选择当前行的右侧

快捷键	说明
Ctrl+Shift+Alt+向左箭头	在代码行中从左向右选择可用列
Ctrl+Shift+Alt+向右箭头	在代码行中从右向左选择可用列
Ctrl+空格键	访问自动完成功能
Ctrl+Shift+空格键	访问自动完成功能
Ctrl+Shift+U	将选定文本更改为大写字母
Ctrl+U	将选定文本更改为小写字母
Esc	取消选择选定文本
Ctrl+I	打开变量选择器
Ctrl+Shift+I	打开变量选择器
Ctrl+Alt+R	打开块选择器
Ctrl+Shift+Alt+R	打开块选择器
Insert	在覆盖/插入键入模式之间切换
Ctrl+Shift+T	调换当前字与上一个字
Ctrl+Shift+Alt+T	调换当前行和下一行

¹ 如果未选定梯级，则会将某个梯级添加到梯级列表的末尾。

² 如果选定了某个分支，则会将某个元素插入到该分支的末尾。

另请参见

[梯形图 \(LD\) 语言参考](#) 参考页数 35

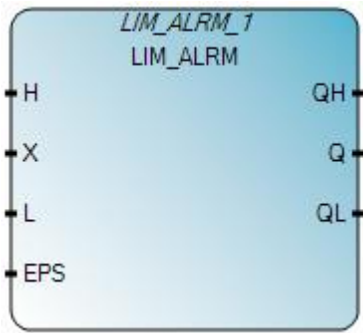
警报指令

警报指令用于在达到配置的上限或下限时发出提醒。

功能块	说明
LIM_ALARM 参考页数 69	上限和下限实值的滞后。

LIM_ALARM

LIM_ALARM 是关于上限和下限实型值滞后的警报。



LIM_ALARM 操作

滞后应用于上限和下限。用于上限或下限的滞后增量是 EPS 参数的一半。

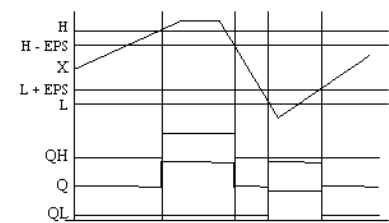
过程警报

当控制器收到并处理故障时会产生警报。当模块超过为每个通道配置的上限或下限时，进程级别警报将发出提醒。

参数

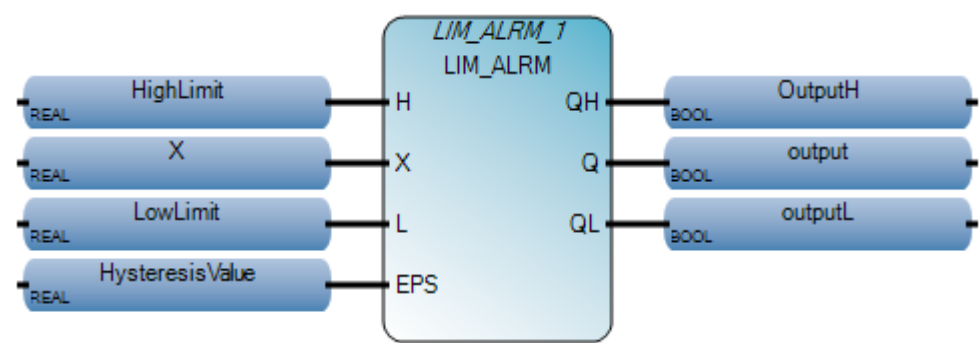
参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前的 LIM_ALRM 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
H	输入	REAL	上限值。
X	输入	REAL	输入：任何实型值。
L	输入	REAL	下限值。
EPS	输入	REAL	滞后值（必须大于 0）。
QH	输出	BOOL	上限警报：如果 X 超过上限 H 则为 TRUE。
Q	输出	BOOL	警报输出：如果 X 超出限制范围则为 TRUE。
QL	输出	BOOL	下限警报：如果 X 低于下限 L 则为 TRUE。

LIM_ALRM 时间图示例

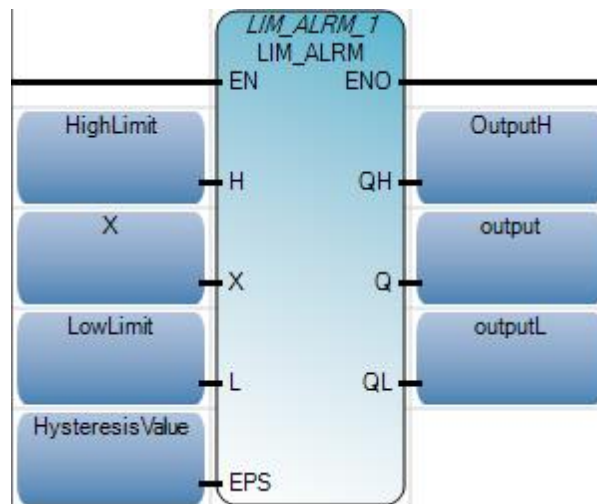


LIM_ALRM 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1 HighLimit := 10.0;
2 X := 15.0;
3 LowLimit := 5.0;
4 HysteresisValue := 2.0;
5 LIM_ALARM_1(HighLimit, X, LowLimit, HysteresisValue);
6 OutputH := LIM_ALARM_1.QH;
7 OutputL := LIM_ALARM_1.QL;
8 output := LIM_ALARM_1.Q;

```

LIM_ALARM_1 (

void LIM_ALARM_1 (REAL H, REAL X, REAL L, REAL EPS)
Type : LIM_ALARM, High/low limit alarm with hysteresis

结果



另请参见

[警报指令](#) 参考页数 [69](#)

算术指令

算术指令使控制器能够执行数学函数，例如数据的加减乘除。

函数	说明
ABS 参考页数 74	实型值的绝对值
ACOS 参考页数 76	实型值的反余弦
ACOS_LREAL 参考页数 77	执行 64 位实型反余弦计算
Addition 参考页数 79	加值
ASIN 参考页数 80	实型值的反正弦
ASIN_LREAL 参考页数 82	执行 64 位实型反正弦计算
ATAN 参考页数 84	实型值的反正切
ATAN_LREAL 参考页数 86	执行 64 位实型反正切计算
COS 参考页数 87	实型值的余弦
COS_LREAL 参考页数 89	执行 64 位实型余弦计算
Division 参考页数 91	除整型或实型值
EXPT 参考页数 92	实型值的指数计算
LOG 参考页数 94	实型值的对数
MOD 参考页数 96	模块
MOV 参考页数 98	移动值副本
Multiplication 参考页数 99	乘整型或实型值
取反 参考页数 100	将值取反
POW 参考页数 101	实型值的幂计算
RAND 参考页数 103	随机值
SIN 参考页数 105	实型值的正弦
SIN_LREAL 参考页数 107	执行 64 位实型正弦计算
SQRT 参考页数 109	实型值的平方根

函数	说明
Subtraction 参考页数 110	减值
TAN 参考页数 111	实型值的正切
TAN LREAL 参考页数 113	执行 64 位实型正切计算
TRUNC 参考页数 115	截断实型值，只保留整数。

ABS

ABS 生成实型值的绝对值（正值）。



参数

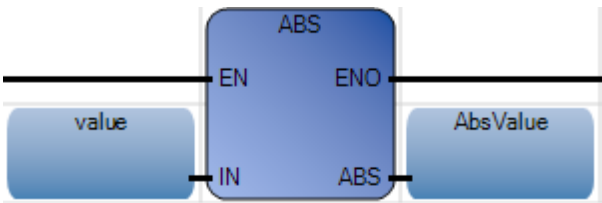
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前绝对值计算。 当 EN = FALSE 时，不执行计算。
IN	输入	REAL	任何有符号实型值。
ENO	输出	BOOL	启用输出。
ABS	输出	REAL	绝对值（始终为正数）。

ABS 函数语言示例

功能块图



梯形图



结构化文本图

```
1 value := -1.0;  
2 AbsValue := ABS(value);
```

ABS (|

REAL ABS(REAL IN)
Absolute value

(* 与之等效的 ST : *)

over := (ABS (delta) > range);

结果

变量监视					
用户全局变量 - 局部变量 - RA_ABS_LD 系统变量- Micro83					
	名称	逻辑值	实际值	锁定	数据类型
	Value	0.0	不可用	☐	REAL
	AbsValue	0.0	不可用	☐	REAL

关闭

另请参见

[算术指令](#) 参考页数 [73](#)

ACOS

ACOS 生成实型值的反余弦。输入值和输出值都以弧度表示。

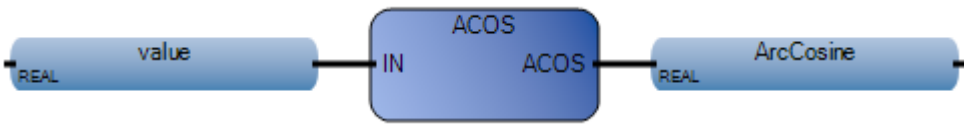


参数

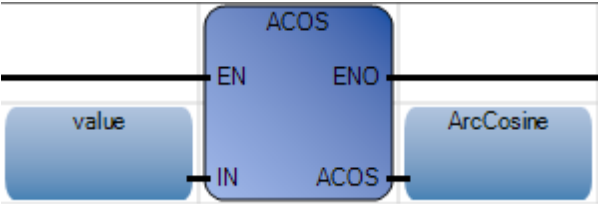
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前反余弦计算。 当 EN = FALSE 时，不执行计算。
IN	输入	REAL	必须位于以下集合内：[-1.0 ..+1.0]。
ENO	输出	BOOL	启用输出。
ACOS	输出	REAL	对于无效输入，输入值的反余弦（位于集合 [-p1/2..+p1/2] ）=0

ACOS 函数语言示例

功能块图

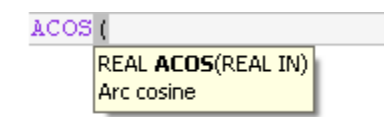


梯形图



结构化文本

```
1 value := 0.5;  
2 ArcCosine := ACOS(value);
```

(* 与之等效的 ST : *)

cosine := COS (angle);

result := ACOS (cosine); (* 结果为角度 *)

结果



另请参见

[算术指令](#) 参考页数 [73](#)

ACOS_LREAL

ACOS_LREAL 计算长实型值的反余弦。

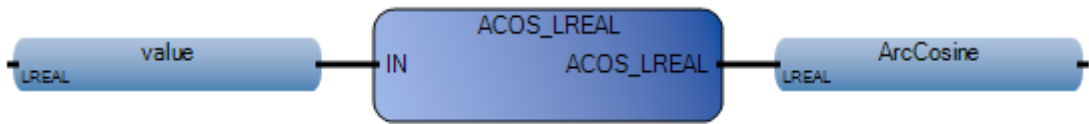


参数

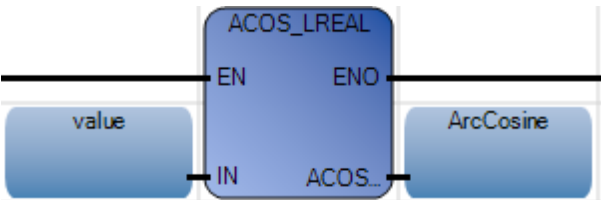
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前计算。 当 EN = FALSE 时，不执行计算。
IN	输入	LREAL	必须位于以下集合内：[-1.0 ..+1.0]。
ENO	输出	BOOL	启用输出。
ACOS_LREAL	输出	LREAL	对于无效输入，输入值的反余弦（在集合 [0.0 ..PI] 内）= 0.0。

ACOS_LREAL 函数语言示例

功能块图



梯形图



结构化文本

```
1 value := 0.5;  
2 ArcCosine := ACOS_LREAL(value);
```

ACOS_LREAL (

LREAL ACOS_LREAL(LREAL IN)
Perform 64-bit real arccosine calculation.

(* 与之等效的 ST : *)

```
cosine := COS_LREAL (angle);
```

result := ACOS_LREAL (cosine); (* 结果为角度 *)

结果

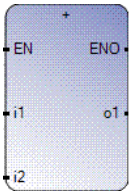


另请参见

[算术指令](#) 参考页数 [73](#)

Addition

Addition 将两个或多个整型、实型、时间或字符串值相加。



Addition 操作

Addition 函数支持附加输入。

参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行当前相加计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。

参数	参数类型	数据类型	说明
i1	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - STRING	整型、时间或字符串数据类型的加数。 所有输入的数据类型必须相同。
i2	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - STRING	整型、时间或字符串数据类型的加数。 所有输入的数据类型必须相同。
o1	输出	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - STRING	实型、时间或字符串格式的输入值的和。 输入和输出必须使用相同的数据类型。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

示例

(* 与之等效的 ST : *)

```
ao10 := ai101 + ai102;  
ao5  := (ai51 + ai52) + ai53;
```

另请参见

[算术指令](#) 参考页数 [73](#)

ASIN

ASIN 生成实型值的反正弦。输入值和输出值都以弧度表示。

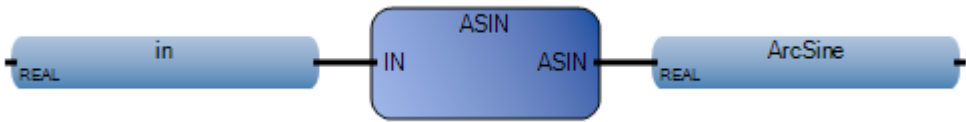


参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前反正弦计算。 当 EN = FALSE 时，不执行计算。
IN	输入	REAL	必须位于以下集合内：[-1.0 ..+1.0]。
ASIN	输出	REAL	对于无效输入，输入值的反正弦（位于集合 [-p1/2..+p1/2] 内）=0。
ENO	输出	BOOL	启用输出。

ASIN 函数语言示例

功能块图

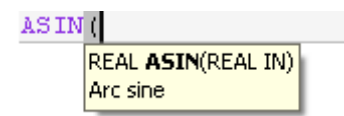


梯形图



结构化文本

```
1 in := 0.5;  
2 ArcSine := ASIN(in);
```



(* 与之等效的 ST : *)

```
sine := SIN (angle);  
  
result := ASIN (sine); (* 结果为角度 *)
```

结果



另请参见

[算术指令](#) 参考页数 [73](#)

ASIN_LREAL

ASIN_LREAL 计算长实型值的反正弦。

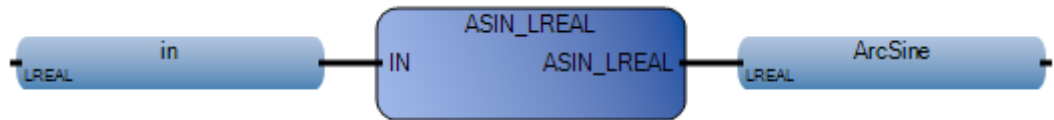


参数

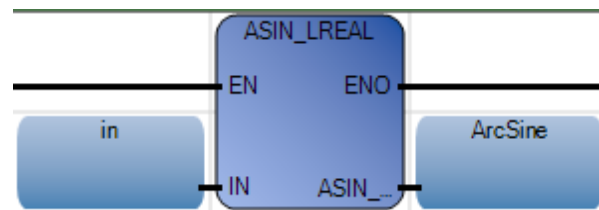
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前计算。 当 EN = FALSE 时，不执行计算。
IN	输入	LREAL	必须位于以下集合内：[-1.0 ..+1.0]。
ASIN_LREAL	输出	LREAL	对于无效输入，输入值的反正弦（位于集合 [-PI/2 ..+PI/2] 内）= 0.0。
ENO	输出	BOOL	启用输出。

ASIN_LREAL 函数语言示例

功能块图



梯形图



结构化文本

```

1  in := 0.5;
2  ArcSine := ASIN_LREAL(in);

```

ASIN_LREAL (

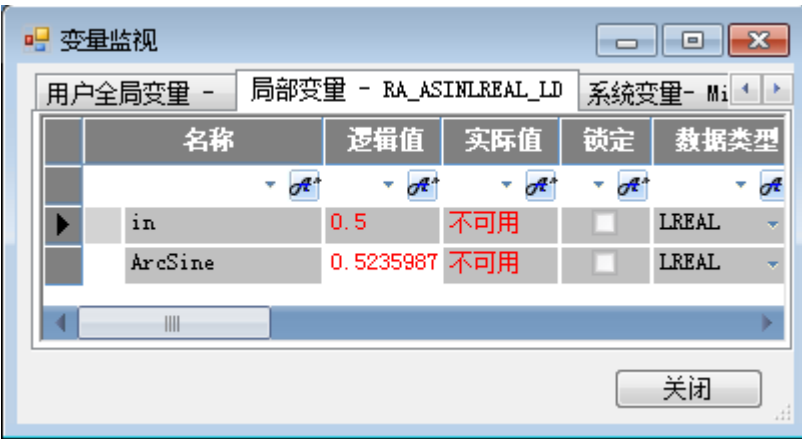
LREAL **ASIN_LREAL**(LREAL IN)
Perform 64-bit real arcsine calculation.

(* 与之等效的 ST : *)

sine := SIN_LREAL (angle);

result := ASIN_LREAL (sine); (* 结果为角度 *)

结果



另请参见

[算术指令](#) 参考页数 [73](#)

ATAN

ATAN 生成实型值的反正切。

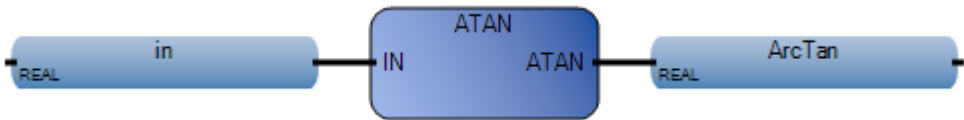


参数

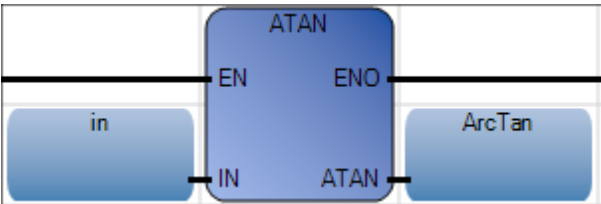
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前反正切计算。 当 EN = FALSE 时，不执行计算。
IN	输入	REAL	任何实型值。
ATAN	输出	REAL	对于无效输入 ,输入值的反正切(位于集合 $[-\pi/2 .. +\pi/2]$ 内)= 0.0。
ENO	输出	BOOL	启用输出。

ATAN 函数语言示例

功能块图



梯形图



结构化文本

```
1 in := 0.5;  
2 ArcTan := ATAN(in);
```

ATAN(

REAL **ATAN**(REAL IN)
Arc tangent

(* 与之等效的 ST : *)

tangent := TAN (angle);

result := ATAN (tangent); (* 结果为角度 *)

结果

变量监视

用户全局变量 - 局部变量 - RA_ATAN_LD 系统变量 - Mi

	名称	逻辑值	实际值	锁定	数据类型
▶	in	0.5	不可用	☐	LREAL
	ArcTan	0.4636476	不可用	☐	LREAL

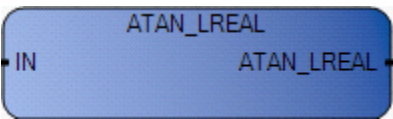
关闭

另请参见

[算术指令](#) 参考页数 [73](#)

ATAN_LREAL

ATAN_LREAL 计算长实型值的反正切。

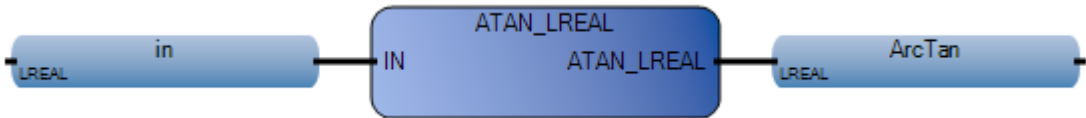


参数

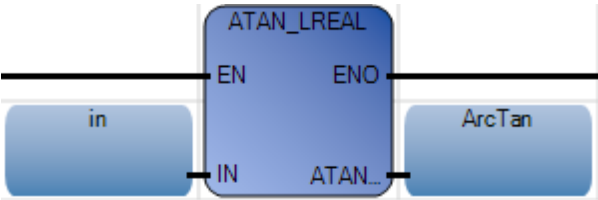
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前计算。 当 EN = FALSE 时，不执行计算。
IN	输入	LREAL	任何长实型值。
ATAN_LREAL	输出	LREAL	对于无效输入 ,输入值的反正切(位于集合 $[-\pi/2 \dots +\pi/2]$ 内)= 0.0。
ENO	输出	BOOL	启用输出。

ATAN_LREAL 函数语言示例

功能块图



梯形图

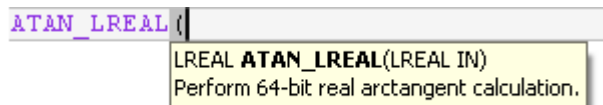


结构化文本

```

1 in := 0.5;
2 ArcTan := ATAN_LREAL(in);

```



(* 与之等效的 ST : *)

tangent := TAN_LREAL (angle);

result := ATAN_LREAL (tangent); (* 结果为角度 *)

结果



另请参见

[算术指令](#) 参考页数 [73](#)

COS

COS 生成实型值的余弦。

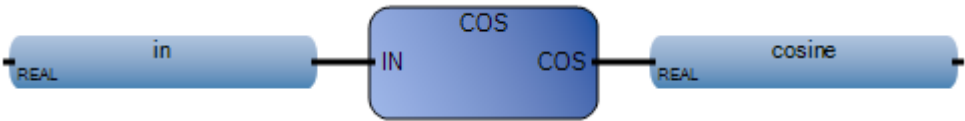


参数

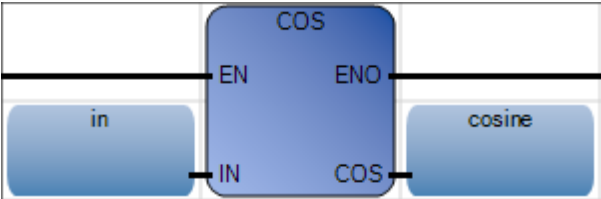
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前余弦计算。 当 EN = FALSE 时，不执行计算。
IN	输入	REAL	任何实型值。
COS	输出	REAL	输入值的余弦（位于集合 [-1.0 ..+1.0]。
ENO	输出	BOOL	启用输出。

COS 函数语言示例

功能块图

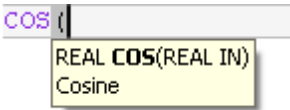


梯形图



结构化文本

```
1  in := 10.0;  
2  cosine := COS(in);
```



(* 与之等效的 ST : *)

```
cosine := COS (angle);
```

result := ACOS (cosine); (* 结果为角度 *)

结果

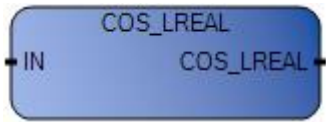


另请参见

[算术指令](#) 参考页数 [73](#)

COS_LREAL

COS_LREAL 计算长实型值的余弦。

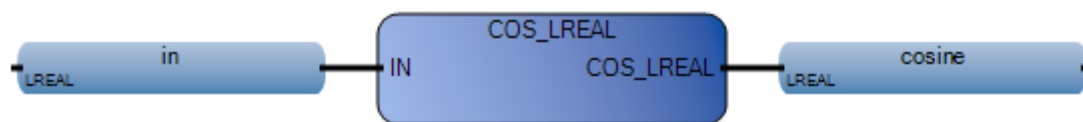


参数

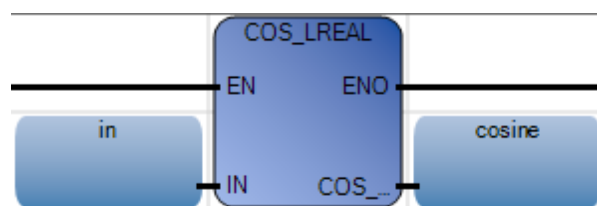
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前计算。 当 EN = FALSE 时，不执行计算。
IN	输入	LREAL	任何长实型值。
COS_LREAL	输出	LREAL	输入值的余弦（位于集合 [-1.0 ..+1.0]）。
ENO	输出	BOOL	启用输出。

COS_LREAL 函数语言示例

功能块图



梯形图



结构化文本

```

1 | in := 10.0;
2 | cosine := COS_LREAL(in);

```

COS_LREAL (

LREAL **COS_LREAL**(LREAL IN)
Perform 64-bit real cosine calculation.

(* 与之等效的 ST : *)

cosine := COS_LREAL (angle);

result := ACOS_LREAL (cosine); (* 结果为角度 *)

结果

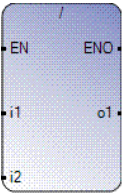


另请参见

[算术指令](#) 参考页数 [73](#)

Division

Division 将首个整型或实型输入值除以第二个整型或实型输入值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行当前相除计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL	非零整型或实型数据类型的被除数。 所有输入的数据类型必须相同。

参数	参数类型	数据类型	说明
i2	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL	非零整型或实型数据类型的除数。 所有输入的数据类型必须相同。
o1	输出	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL	非零整型或实型数据类型输入的商。 输入和输出必须使用相同的数据类型。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

示例

(* 与之等效的 ST : *)

```
ao10 := ai101 / ai102;  
ao5  := (ai5 / 2) / ai53;
```

另请参见

[算术指令](#) 参考页数 [73](#)

EXPT

如果“base”是第一个参数，“exponent”是第二个参数，则 EXPT 生成下列运算的实型结果：(base exponent)。

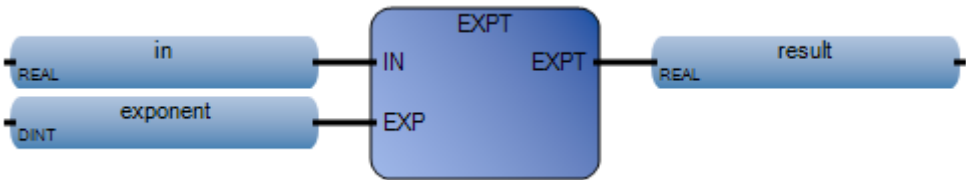


参数

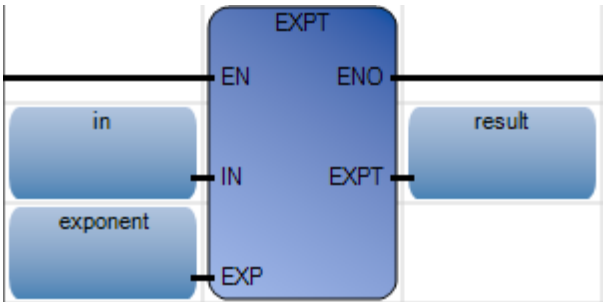
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前指数计算。 当 EN = FALSE 时，不执行计算。
IN	输入	REAL	任何有符号实型值。
EXP	输入	DINT	整型指数。
EXPT	输出	REAL	(IN EXP)。
ENO	输出	BOOL	启用输出。

EXPT 函数语言示例

功能块图

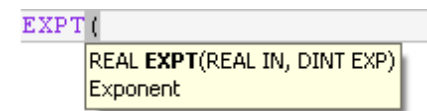


梯形图



结构化文本

```
1  in := 2.0;  
2  exponent := 3;  
3  result := EXPT(in, exponent);
```



(* 与之等效的 ST : *)

tb_size := ANY_TO_DINT (EXPT (2.0, range));

结果



另请参见

[算术指令](#) 参考页数 [73](#)

LOG

LOG 生成实型值的对数（以 10 为底）。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前对数计算。 当 EN = FALSE 时，不执行计算。
IN	输入	REAL	必须大于零。

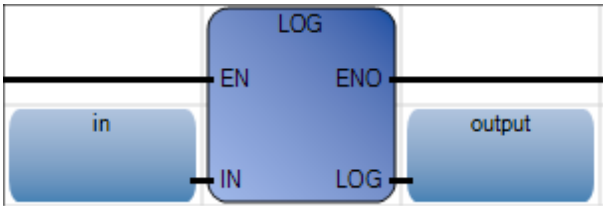
参数	参数类型	数据类型	说明
LOG	输出	REAL	输入值的对数(以 10 为底)。零 IN 值和负 IN 值的返回结果是 -3.4E+38。
ENO	输出	BOOL	启用输出。

LOG 函数语言示例

功能块图

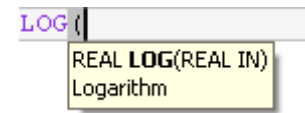


梯形图



结构化文本

```
1 in := 10.0;
2 output := LOG(in);
```



(* 与之等效的 ST : *)

```
xpos := ABS (xval);
xlog := LOG (xpos);
```

结果



另请参见

[算术指令](#) 参考页数 [73](#)

MOD

MOD 生成整型值的模。

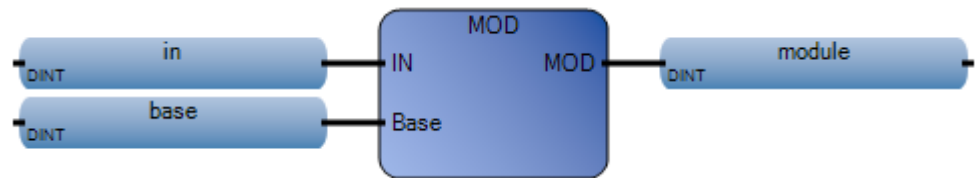


参数

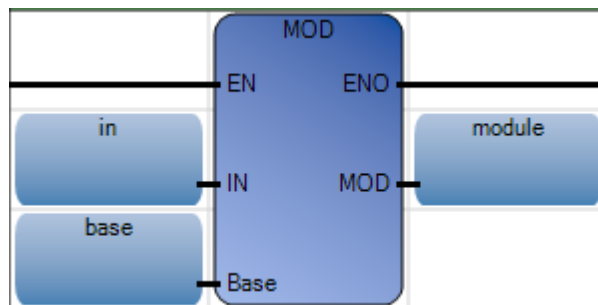
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行模计算。 当 EN = FALSE 时，不执行计算。
IN	输入	DINT	任何有符号整型值。
进制	输入	DINT	必须大于零。
MOD	输出	DINT	模计算 (输入 MOD Base) / 如果 Base <= 0 则返回 -1。
ENO	输出	BOOL	启用输出。

MOD 函数语言示例

功能块图



梯形图



结构化文本

```

1  in := 5;
2  base := 3;
3  module := MOD(in, base);

```

```

MOD (
  DINT MOD(DINT IN, DINT Base)
  Modulo

```

(* 与之等效的 ST : *)

division_result := (value / divider); (* 整型除法 *)

rest_of_division := MOD (value, divider); (* 相除后的余数 *)

结果

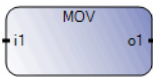


另请参见

[算术指令](#) 参考页数 [73](#)

MOV

MOV 将输入 (i1) 中值的副本移动到输出 (o1)。



提示 在梯形图 POU 或功能块图语言 POU 中启用的 MOV 指令在“模块选择器”中显示，而不在结构化文本 POU 中显示。
： 结构化文本程序使用 “=” 赋值运算符而不是 MOV 函数。

参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行直接链接到输出的计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。 仅适用于 LD 编程。
i1	输入	BOOL - DINT - REAL - TIME - STRING - SINT - USINT - INT - UINT - UDINT - LINT - ULINT - DATE - LREAL - BYTE - WORD - DWORD - LWORD	输入和输出必须使用相同的数据类型。

参数	参数类型	数据类型	说明
o1	输出	BOOL - DINT - REAL - TIME - STRING - SINT - USINT - INT - UINT - UDINT - LINT - ULINT - DATE - LREAL - BYTE - WORD - DWORD - LWORD	输入和输出必须使用相同的数据类型。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

示例

(* 与之等效的 ST : *)

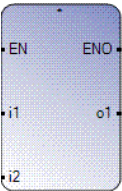
```
ao23 := ai10;
```

另请参见

[算术指令](#) 参考页数 [73](#)

Multiplication

Multiplication 将两个或多个整型或实型值相乘。Multiplication 函数支持附加输入。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行当前相乘计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL	整型或实型数据类型的因数。 所有输入的数据类型必须相同。

参数	参数类型	数据类型	说明
i2	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL	整型或实型数据类型的因数。 所有输入的数据类型必须相同。
o1	输出	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL	整型或实型数据类型输入的乘积。 输入和输出必须使用相同的数据类型。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

示例

(* ST 等效值 *)

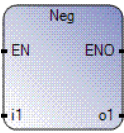
```
ao10 := ai101 * ai102;  
ao5  := (ai51 * ai52) * ai53;
```

另请参见

[算术指令](#) 参考页数 [73](#)

取反

取反将值转换为反值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行当前转换为反变量的计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	SINT - INT - DINT - LINT - REAL - LREAL	输入和输出的数据类型必须相同。
o1	输出	SINT - INT - DINT - LINT - REAL - LREAL	输入和输出的数据类型必须相同。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

示例

(* 与之等效的 ST : *)

```
ao23 := - (ai10);  
ro100 := - (ri1 + ri2);
```

另请参见

[算术指令](#) 参考页数 [73](#)

POW

如果第一个参数为“base”，第二个参数为“exponent”，则 POW 生成下列运算的实型结果：(base exponent)。



POW 操作

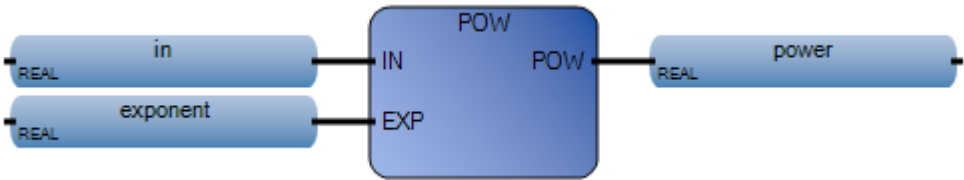
Exponent 为实型值。

参数

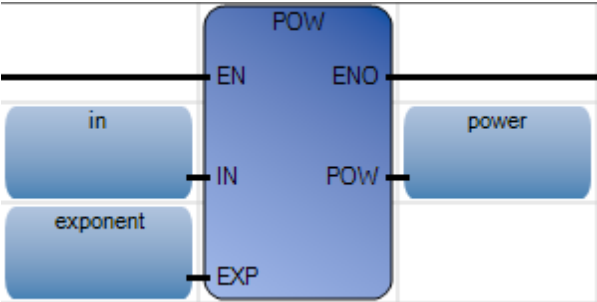
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前指数计算。 当 EN = FALSE 时，不执行计算。
IN	输入	REAL	要计算指数的实型数。
EXP	输入	REAL	幂（指数）。
POW	输出	REAL	(IN EXP) 1.0（如果 IN 不为 0.0 而 EXP 为 0.0） 0.0（如果 IN 为 0.0 而 EXP 为负数） 0.0（如果 IN 和 EXP 都为 0.0） 0.0（如果 IN 为负数而 EXP 不对应于整数）。
ENO	输出	BOOL	启用输出。

POW 函数语言示例

功能块图

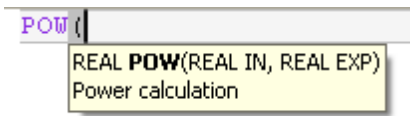


梯形图



结构化文本

```
1  in := 2.0;  
2  exponent := 3.0;  
3  power := POW(in, exponent);
```



(* 与之等效的 ST : *)

result := POW (xval, power);

结果

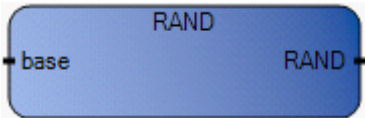


另请参见

[算术指令](#) 参考页数 [73](#)

RAND

RAND 从定义的范围生成随机整数值。



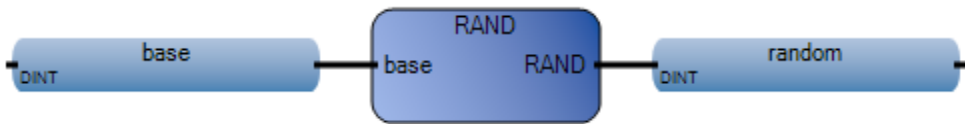
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行随机整数值计算 当 EN = FALSE 时，不执行计算。
进制	输入	DINT	定义支持的一组数字。

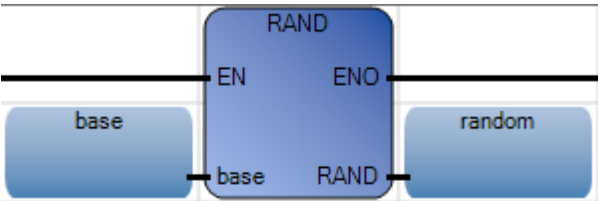
参数	参数类型	数据类型	说明
RAND	输出	DINT	集合 [0..base-1] 内的随机数。
ENO	输出	BOOL	启用输出。

RAND 函数语言示例

功能块图



梯形图



结构化文本

```
1 base := 10;  
2 random := RAND(base);
```



• (* 与之等效的 ST : *)

selected := MUX4 (RAND (4), 1, 4, 8, 16);

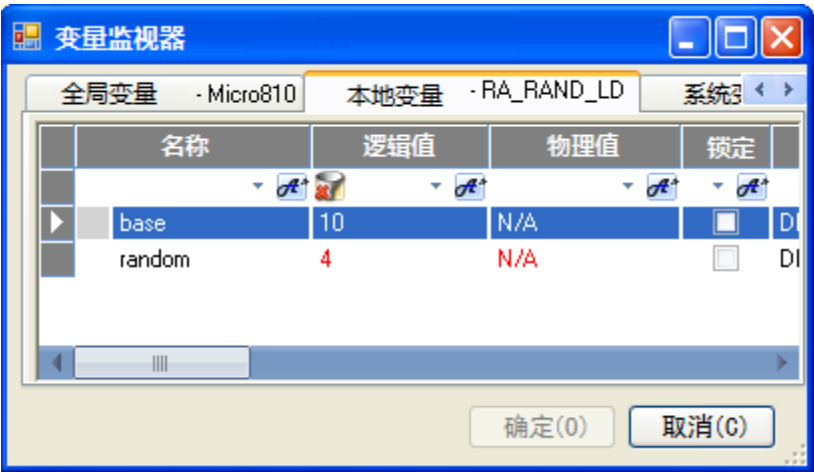
(*

随机选择 4 个预定义值之一

RAND 调用发出的值位于集合 [0..3] 内 ,

所以从 MUX4 发出的“选定值”将“随机”获取值

- 1 (如果从 RAND 发出 0)、
 - 或 4 (如果从 RAND 发出 1)
 - 或 8 (如果从 RAND 发出 2)
 - 或 16 (如果从 RAND 发出 3)
- *)
- 结果



另请参见

[算术指令](#) 参考页数 [73](#)

SIN

SIN 生成实型值的正弦。



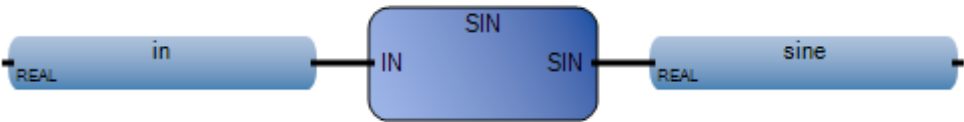
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前正弦计算。 当 EN = FALSE 时，不执行计算。

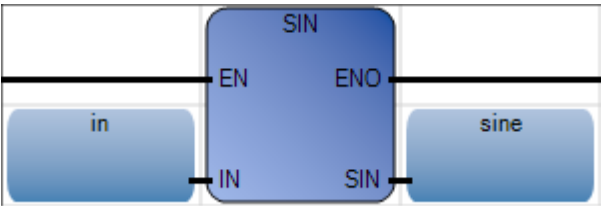
IN	输入	REAL	任何实型值。
SIN	输出	REAL	输入值的正弦（位于集合 [-1.0 ..+1.0]。
ENO	输出	BOOL	启用输出。

SIN 函数语言示例

功能块图

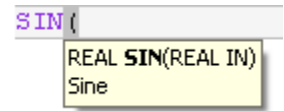


梯形图



结构化文本

```
1 in := 0.5;
2 sine := SIN(in);
```



(* 与之等效的 ST : *)

sine := SIN (angle);

result := ASIN (sine); (* 结果为角度 *)

结果

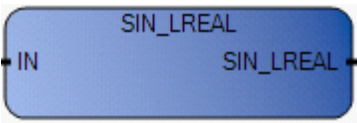


另请参见

[算术指令](#) 参考页数 [73](#)

SIN_LREAL

SIN_LREAL 计算长实型值的正弦。

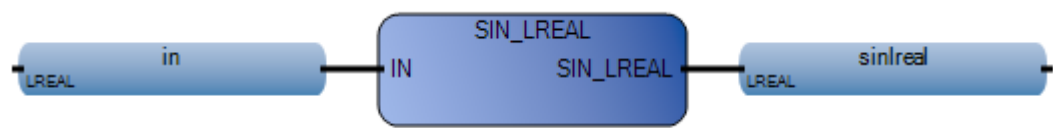


参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前计算。 当 EN = FALSE 时，不执行计算。
IN	输入	LREAL	任何长实型值。
SIN_LREAL	输出	LREAL	输入值的正弦（位于集合 [-1.0 ..+1.0]）。
ENO	输出	BOOL	启用输出。

SIN_LREAL 函数语言示例

功能块图



梯形图



结构化文本

```
1  in := 0.5;
2  sinreal := SIN_LREAL(in);
```

SIN_LREAL(

LREAL SIN_LREAL(LREAL IN)
Perform 64-bit real sine calculation.

(* 与之等效的 ST : *)

```
TESTOUTPUT1 := SIN_LREAL(TESTINPUT1);
```

结果

变量监视器

全局变量 · Micro810 本地变量 · RA_SINLREAL_LD 系

	名称	逻辑值	物理值	锁定	
	in	0.5	N/A	☐	LF
▶	sinreal	0.479425538604	N/A	☐	LF

确定(O) 取消(O)

另请参见

[算术指令](#) 参考页数 [73](#)

SQRT

SQRT 生成实型值的平方根。

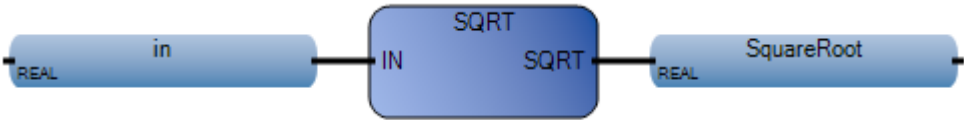


参数

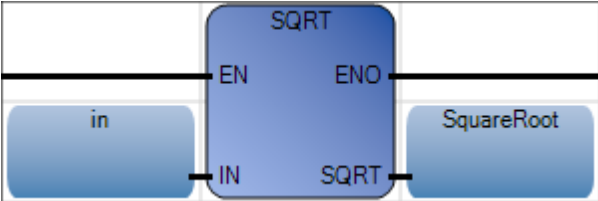
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前平方根计算。 当 EN = FALSE 时，不执行计算。
IN	输入	REAL	必须大于或等于零。
SQRT	输出	REAL	输入值的平方根。负 IN 值的返回结果是 0。
ENO	输出	BOOL	启用输出。

SQRT 函数语言示例

功能块图

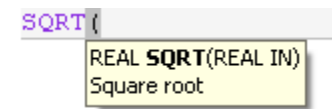


梯形图



结构化文本

```
1  in := 16.0;  
2  SquareRoot := SQRT(in);
```



(* 与之等效的 ST : *)

```
xpos := ABS (xval);  
xroot := SQRT (xpos);
```

结果

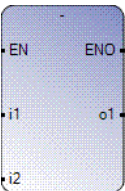


另请参见

[算术指令](#) 参考页数 [73](#)

Subtraction

Subtraction 用整型、实型或时间值减去另一个整型、实型或时间值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行当前相加计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME	任意整型、实型或时间数据类型的被减数。 所有输入的数据类型必须相同。
i2	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME	任意整型、实型或时间数据类型的减数。 所有输入的数据类型必须相同。
o1	输出	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME	任何整型、实型或时间数据类型的被减数和减数的区别。 输出数据类型必须与输入相同。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

示例

(* 与之等效的 ST : *)

```
ao10 := ai101 - ai102;  
ao5  := (ai51 - 1) - ai53;
```

另请参见

[算术指令](#) 参考页数 [73](#)

TAN

TAN 生成实型值的正切。

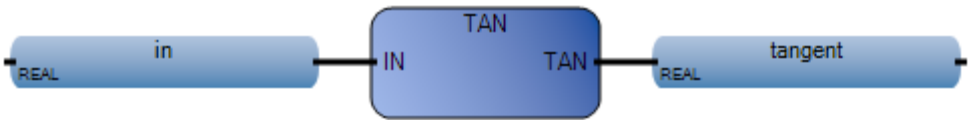


参数

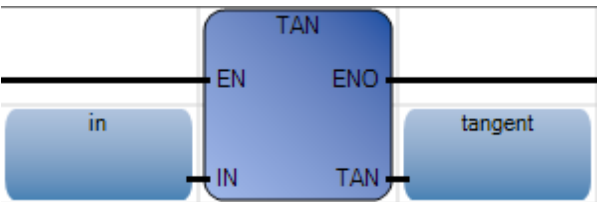
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前正切计算。 当 EN = FALSE 时，不执行计算。
IN	输入	REAL	不能等于 PI/2 模 PI。
TAN	输出	REAL	对于无效输入，输入值的正切 = 1E+38。
ENO	输出	BOOL	启用输出。

TAN 函数语言示例

功能块图



梯形图



结构化文本

```
1 in := 0.5;  
2 tangent := TAN(in);
```

TAN{

REAL TAN(REAL IN)
Tangent

(* 与之等效的 ST : *)

tangent := TAN (angle);

result := ATAN (tangent); (* 结果为角度 *)

结果

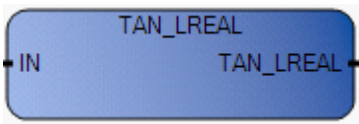


另请参见

[算术指令](#) 参考页数 [73](#)

TAN_LREAL

TAN_LREAL 计算长实型值的正切。



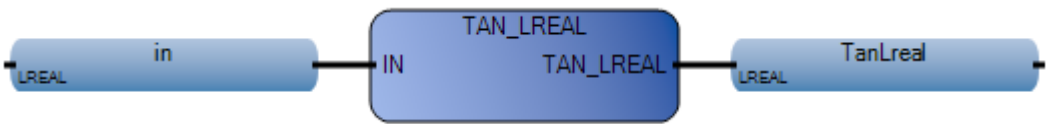
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前计算。 当 EN = FALSE 时，不执行计算。
IN	输入	LREAL	不能等于 PI/2 模 PI。
TAN_LREAL	输出	LREAL	对于无效输入，输入值的正切 = 1E+38。

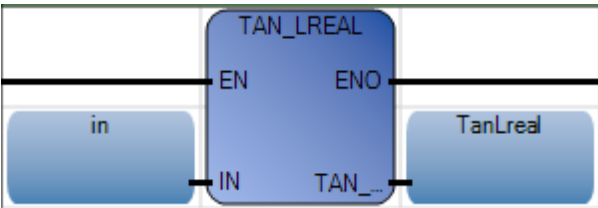
参数	参数类型	数据类型	说明
ENO	输出	BOOL	启用输出。

TAN_LREAL 函数语言示例

功能块图



梯形图



结构化文本

```
1 | in := 0.5;  
2 | TanLreal := TAN_LREAL(in);
```

TAN_LREAL {

LREAL TAN_LREAL(LREAL IN)
Perform 64-bit real tangent calculation.

```
(* 与之等效的 ST : *)  
  
tangent := TAN_LREAL (angle);  
  
result := ATAN_LREAL (tangent); (* 结果为角度 *)
```

结果



另请参见

[算术指令](#) 参考页数 [73](#)

TRUNC

TRUNC 截断实型值，只保留整数。

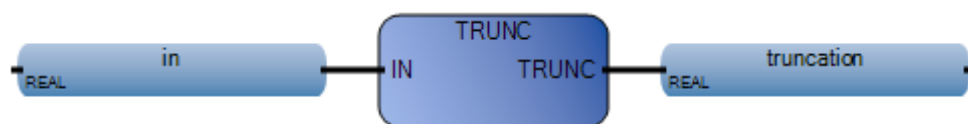


参数

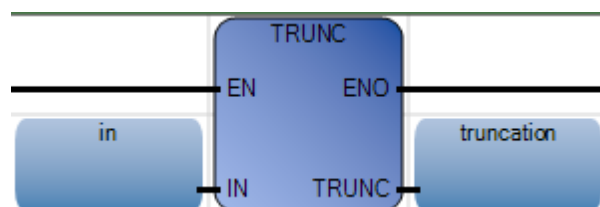
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行实型值截断计算。 当 EN = FALSE 时，不执行计算。
IN	输入	REAL	任何实型值。
TRUNC	输出	REAL	如果 IN>0，则最大整数值小于或等于输入值。 如果 IN<0，则最小整数值大于或等于输入值。
ENO	输出	BOOL	启用输出。

TRUNC 函数语言示例

功能块图



梯形图

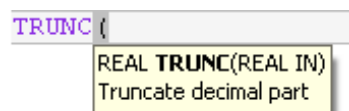


结构化文本

```

1 | in := 1.7;
2 | truncation := TRUNC(in);

```



(* 与之等效的 ST : *)

result := TRUNC (+2.67) + TRUNC (-2.0891);

(* 表示 : result := 2.0 + (-2.0) := 0.0; *)

结果



另请参见

[算术指令](#) 参考页数 [73](#)

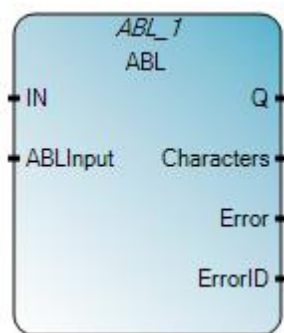
ASCII 串行端口指令

ASCII 串行端口指令是通信指令，这些指令通过使用或修改通信信道来接收或传输数据。

功能块	说明
ABL 参考页数 119	确定缓冲区中的字符数（甚至可包括行尾字符）
ACB 参考页数 121	确定缓冲区中的字符总数
ACL 参考页数 123	清除接收和传输的缓冲
AHL 参考页数 124	设置或重置调制解调器握手信号线
ARD 参考页数 127	从输入缓冲区读取字符，并将这些字符放置到某个字符串中
ARL 参考页数 129	从输入缓冲区读取一行字符，并将这些字符放置到某个字符串中
AWA 参考页数 131	写入某个字符串，该字符串包含追加到外部设备的用户配置字符
AWT 参考页数 133	将字符从源字符串写入到外部设备中

ABL

ABL 统计输入缓冲区中的字符总数，甚至可包括行尾终止符。

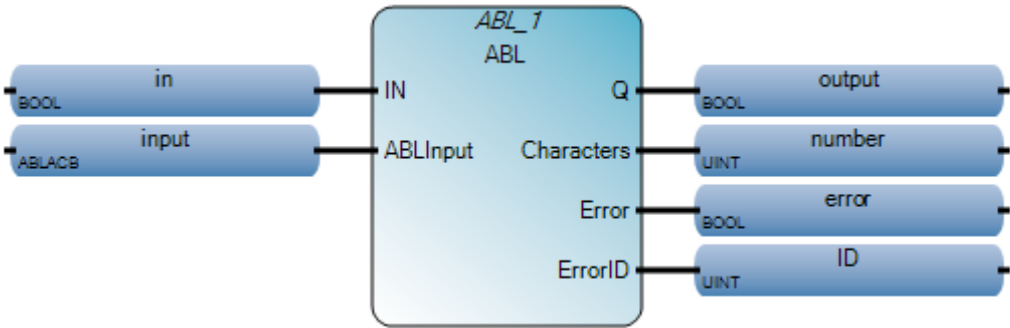


参数

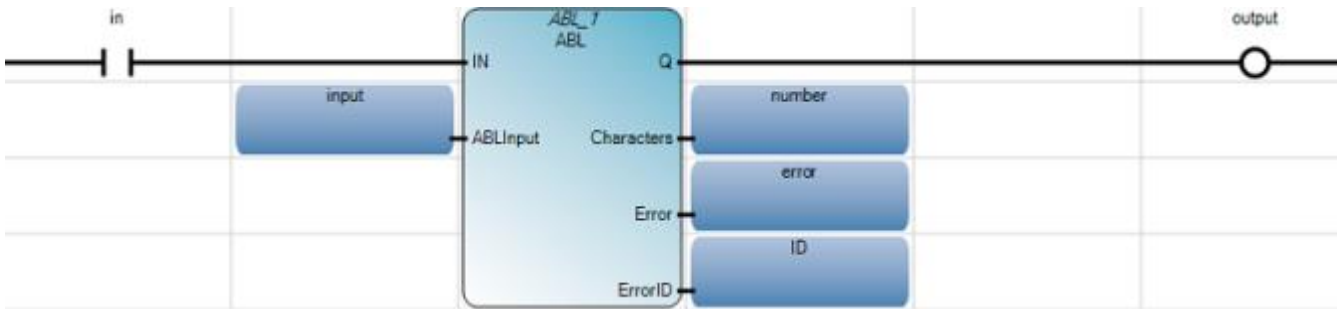
参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿 (IN) 从“假”变为“真”，则启动功能块，前提是上一操作已完成。
ABLInput	输入	ABLACB	要操作的通道。 请参见 ABLACB 数据类型 参考页数 136 。
Q	输出	BOOL	FALSE - 功能块未完成。 TRUE - 功能块完成。
Characters	输出	UINT	缓冲区中显示的字符数，最多为 82 个字符。
Error	输出	BOOL	FALSE - 没有错误。 TRUE - 检测到错误。
ErrorID	输出	UINT	请参见 ABL 错误代码 参考页数 135 。

ABL 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1  ABL_1(in, input);
2  output := ABL_1.Q;
3  number := ABL_1.Characters;
4  error := ABL_1.Error;
5  ID := ABL_1.ErrorID;
```

ABL_1{

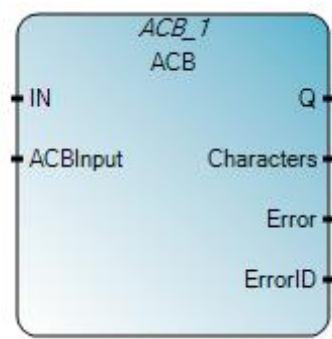
void **ABL_1**(BOOL IN, ABLACB ABLInput)
Type : ABL, Specify number of characters in buffer (including end of line).

另请参见

[ASCII 串行端口指令](#) 参考页数 [119](#)

ACB

ACB 确定缓冲区中的字符总数。



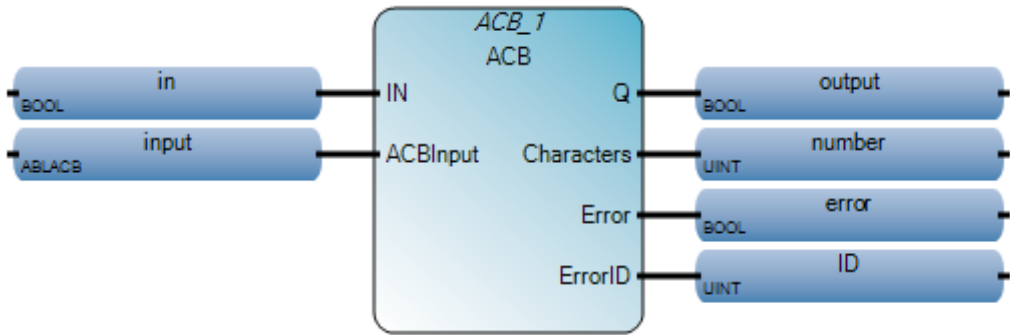
参数

参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿（IN 从 FALSE 变为 TRUE），则启动功能块，前提是上一操作已完成。
ACBInput	输入	ABLACB	要操作的通道。 请参见 ABLACB 数据类型 参考页数 136 。
Q	输出	BOOL	FALSE - 功能块未完成。 TRUE - 功能块完成。
Characters	输出	UINT	字符数。

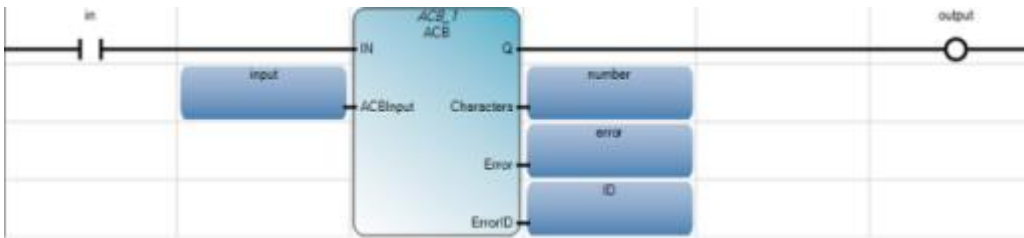
Error	输出	BOOL	FALSE - 没有错误。 TRUE - 检测到错误。
ErrorID	输出	UINT	请参见 ABL 错误代码 参考页数 135 。

ACB 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1  ACB_1(in, input);
2  output := ACB_1.Q;
3  number := ACB_1.Characters;
4  error := ACB_1.Error;
5  ID := ACB_1.ErrorID;
```

ACB_1 (

void **ACB_1**(BOOL IN, ABLACB ACBInput)
Type : ACB, Determine total number of characters in buffer.

另请参见

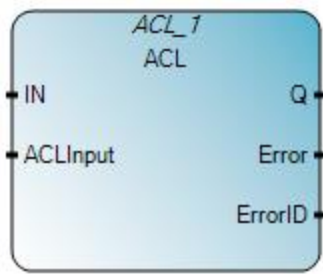
[ASCII 串行端口指令](#) 参考页数 [119](#)

[ABLACB 数据类型](#) 参考页数 [136](#)

[ABL 错误代码](#) 参考页数 [135](#)

ACL

ACL 清除接收和传输缓冲区，并将指令从 ASCII 队列中移除。

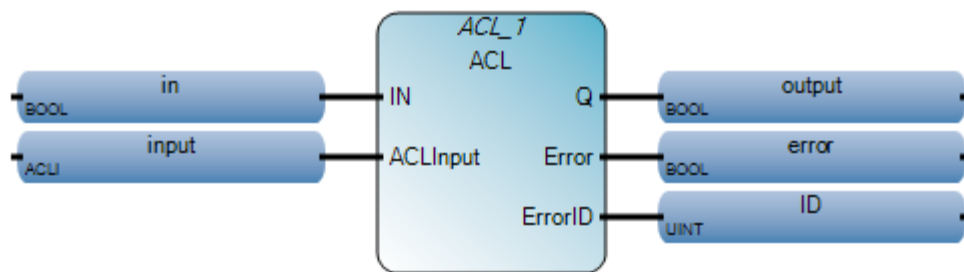


参数

参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿（IN 从 FALSE 变为 TRUE），则启动功能块，前提是上一操作已完成。
ACLInput	输入	ACL	传输和接收缓冲区的状态。 请参见 ACL 数据类型 参考页数 137
Q	输出	BOOL	FALSE - 功能块未完成。 TRUE - 功能块完成。
Error	输出	BOOL	FALSE - 没有错误。 TRUE - 检测到错误。
ErrorID	输出	UINT	请参见 ABL 错误代码 参考页数 135 。

ACL 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1  ACL_1(in, input);
2  output := ACL_1.Q;
3  error  := ACL_1.Error;
4  ID     := ACL_1.ErrorID;

```

```

ACL_1 {
  void ACL_1(BOOL IN, ACLI ACLInput)
  Type : ACL, Clear the receive and/or transmit buffers.
}

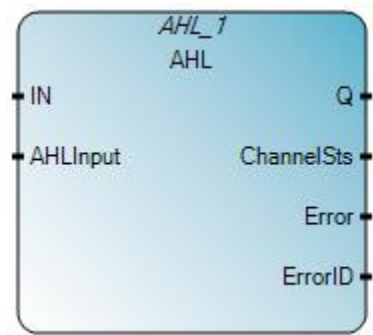
```

另请参见

[ASCII 串行端口指令](#) 参考页数 [119](#)
[ABL 错误代码](#) 参考页数 [135](#)

AHL

AHL 可设置或重置调制解调器的 RS-232 发送请求 (RTS) 握手信号制线。

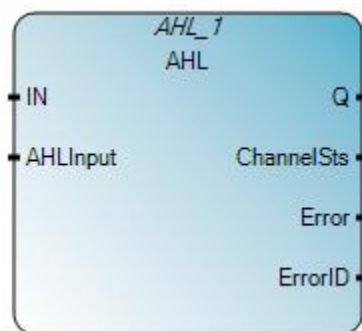


参数

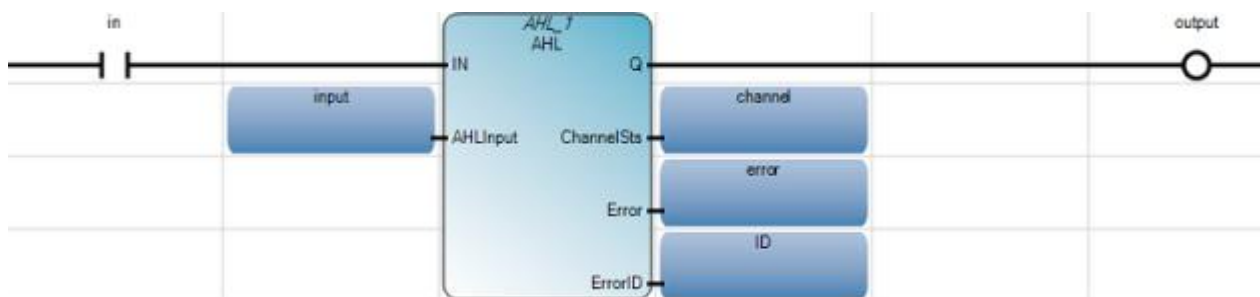
参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿 (IN 从 FALSE 变为 TRUE) , 则启动功能块, 前提是上一操作已完成。
AHLInput	输入	AHLI	置位或重置调制解调器的 RTS 控制行。 请参见 AHLI 数据类型 参考页数 138 。
Q	输出	BOOL	FALSE - 功能块未完成。 TRUE - 功能块完成。
ChannelSts	输出	WORD	显示指定通道的握手信号行的当前状态 (0000 到 001F) 。 请参见 AHL ChannelSts 数据类型 参考页数 137 。
Error	输出	BOOL	FALSE - 没有错误。 TRUE - 检测到错误。
ErrorID	输出	UINT	请参见 ABL 错误代码 参考页数 135 。

AHL 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1  AHL_1(in, input);
2  output := AHL_1.Q;
3  channel := AHL_1.ChannelSts;
4  error := AHL_1.Error;
5  ID := AHL_1.ErrorID;

```

```

AHL_1(
  void AHL_1(BOOL IN, ABL AHLInput)
  Type : AHL, Set or reset modem handshake lines.

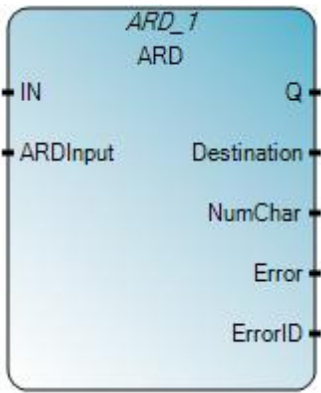
```

另请参见

[ASCII 串行端口指令](#) 参考页数 [119](#)
[ABL 错误代码](#) 参考页数 [135](#)

ARD

ARD 从缓冲区读取字符，并将这些字符存储在字符串中。



ARD 操作

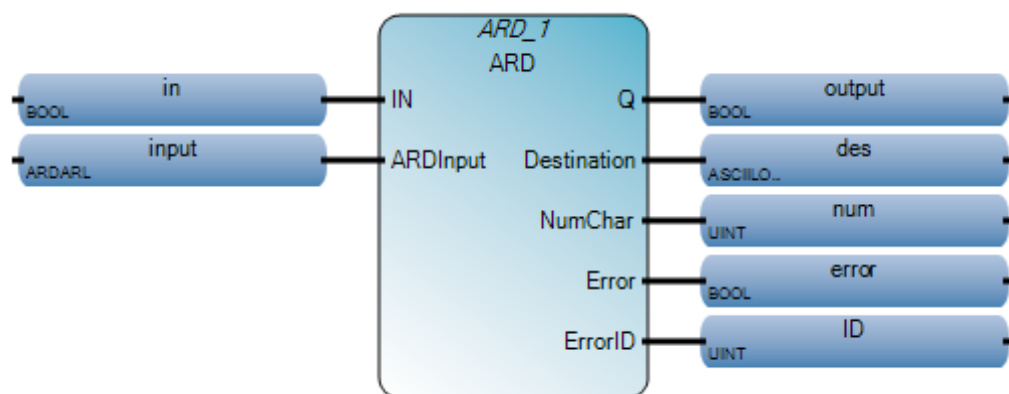
- 将执行 ARD，直至收到所有字符。如果执行其他 ASCII 命令，命令将排队等待 ARD 完成。可执行 [ACL](#) 参考页数 [123](#) 指令以中止 ARD 指令。
- 可使用 [ACB](#) 参考页数 [121](#) 指令的结果来触发 ARD 指令。这可以防止 ARD 指令在等待所需字符数时阻碍 ASCII 队列。

参数

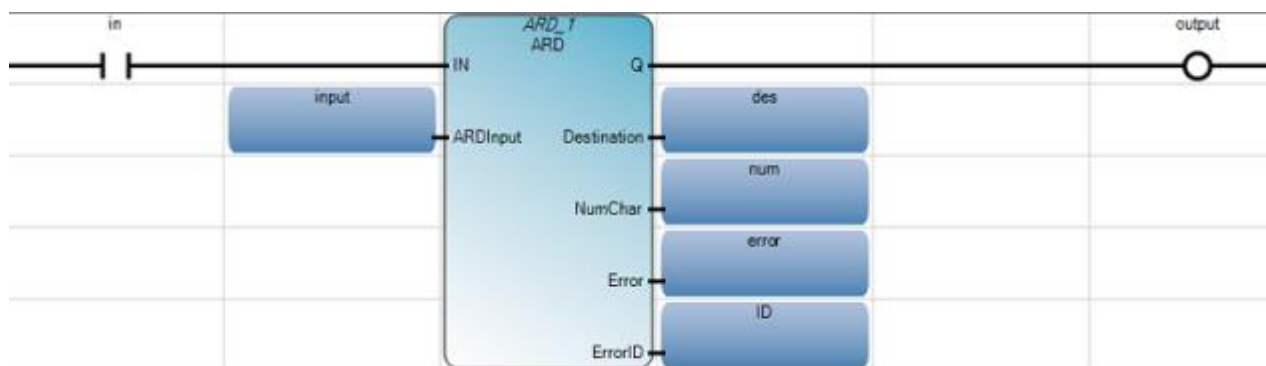
参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿（IN 从 FALSE 变为 TRUE），则启动功能块，前提是上一操作已完成。
ARDInput	输入	ARDARL	从缓冲区读取字符（最大值为 82）。 请参见 ARDARL 数据类型 参考页数 139 。
Q	输出	BOOL	FALSE - 功能块未完成。 TRUE - 功能块完成。
Destination	输出	ASCIILOCADDR	您希望在其中存储字符的字符串元素。
NumChar	输出	UINT	字符数。
Error	输出	BOOL	FALSE - 没有错误。 TRUE - 检测到错误。
ErrorID	输出	UINT	请参见 ABL 错误代码 参考页数 135 。

ARD 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1  ARD_1(in, input);
2  output := ARD_1.Q;
3  des := ARD_1.Destination;
4  num := ARD_1.NumChar;
5  error := ARD_1.Error;
6  ID := ARD_1.ErrorID;

```

```

ARD_1 {

```

```

void ARD_1(BOOL IN, ARDARL ARDInput)
Type : ARD, Read characters from the input buffer and place them into a string.

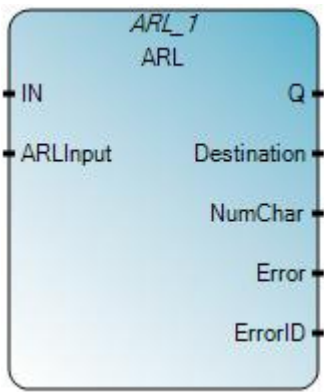
```

另请参见

[ASCII 串行端口指令](#) 参考页数 [119](#)

ARL

ARL 从缓冲区读取字符（甚至可包括终止符），并将这些字符存储在字符串中。

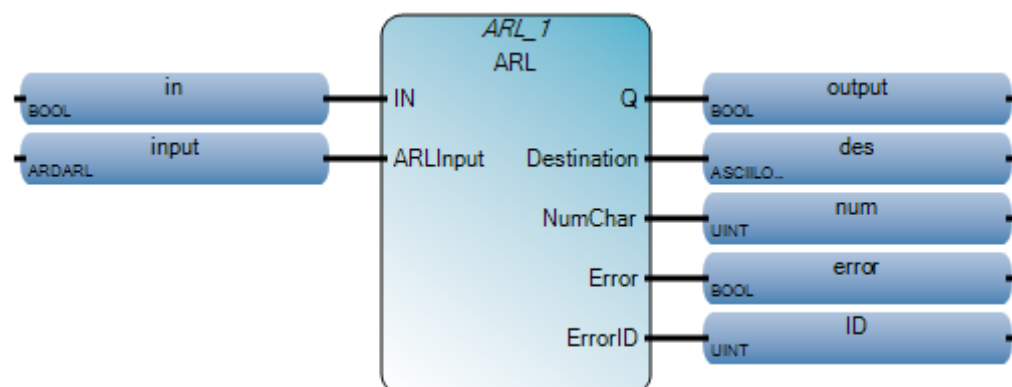


参数

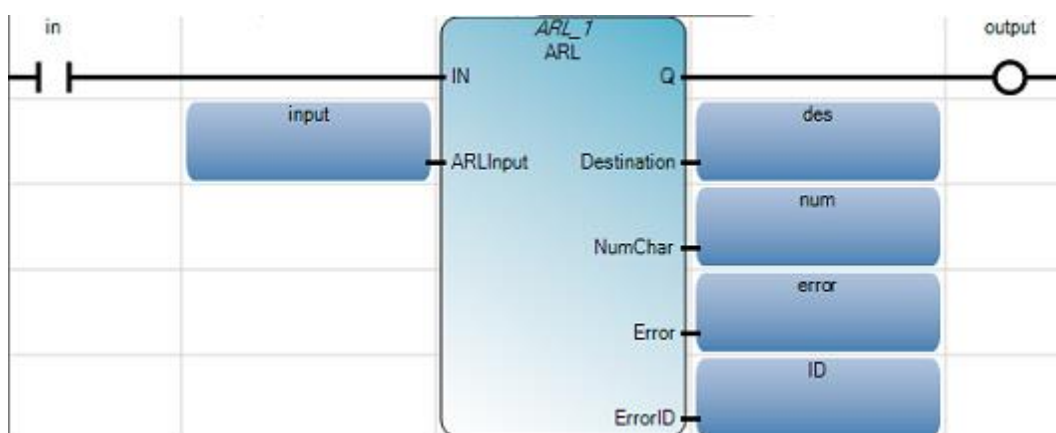
参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿（IN 从 FALSE 变为 TRUE），则启动功能块，前提是上一操作已完成。
ARLInput	输入	ARDARL	从缓冲区读取字符（最大值为 82）。 请参见 ARDARL 数据类型 参考页数 139 。
Q	输出	BOOL	FALSE - 功能块未完成。 TRUE - 功能块完成。
Destination	输出	ASCIILOCADDR	您希望在其中存储字符的字符串元素。
NumChar	输出	UINT	字符数。
Error	输出	BOOL	FALSE - 没有错误。 TRUE - 检测到错误。
ErrorID	输出	UINT	请参见 ABL 错误代码 参考页数 135 。

ARL 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1  ARL_1(in, input);
2  output := ARL_1.Q;
3  des := ARL_1.Destination;
4  num := ARL_1.NumChar;
5  error := ARL_1.Error;
6  ID := ARL_1.ErrorID;

```

```

ARL_1(
void ARL_1(BOOL IN, ARDARL ARLInput)
Type : ARL, Read line from the input buffer and place characters in a string.

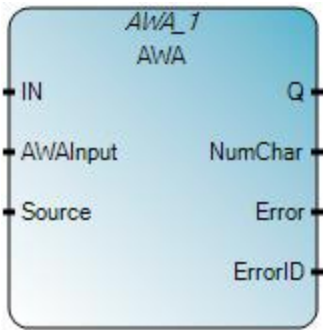
```

另请参见

[ASCII 串行端口指令](#) 参考页数 [119](#)

AWA

AWA 将源字符串中的字符写入到外部设备中。本指令添加两个附加的字符，这些字符在配置对话框中配置。

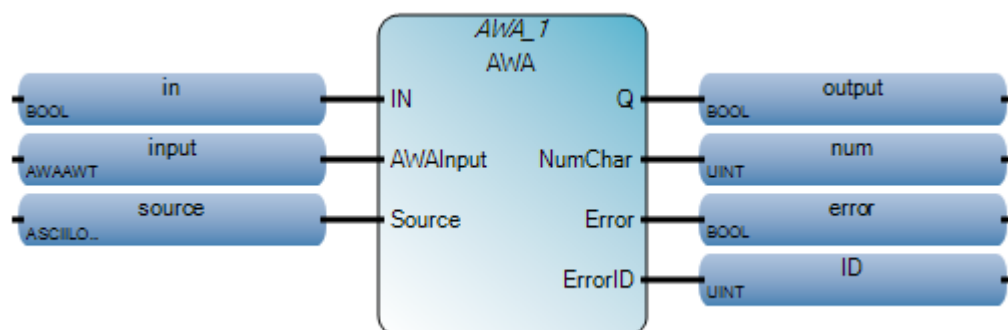


参数

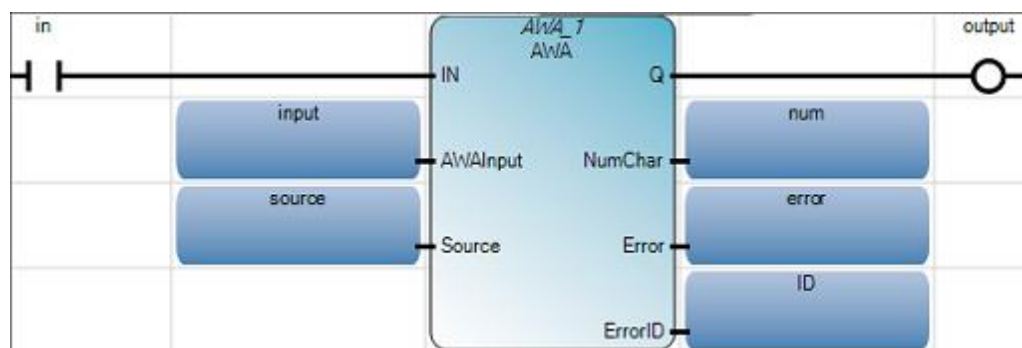
参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿（IN 从 FALSE 变为 TRUE），则启动功能块，前提是上一操作已完成。
AWAInput	输入	AWAAWT	要操作的通道和长度。 请参见 AWAAWT 数据类型 参考页数 139 。
Source	输入	ASCIILOCADDR	源字符串：字符数组。
Q	输出	BOOL	FALSE - 功能块未完成。 TRUE - 功能块完成。
NumChar	输出	UINT	字符数。
Error	输出	BOOL	FALSE - 没有错误。 TRUE - 检测到错误。
ErrorID	输出	UINT	请参见 ABL 错误代码 参考页数 135 。

AWA 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1  AWA_1(in, input, source);
2  output := AWA_1.Q;
3  num := AWA_1.NumChar;
4  error := AWA_1.Error;
5  ID := AWA_1.ErrorID;

```

```

AWA_1 {
void AWA_1(BOOL IN, AWAAWT AWAInput, ASCIILOCADDR Source, UINT __ADI_Source)
Type : AWA, Write a string with characters appended to an external device.

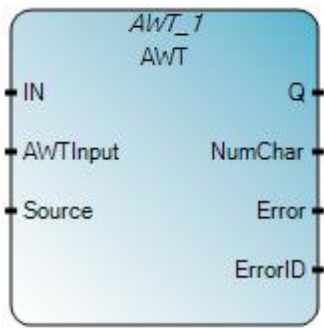
```


另请参见

[ASCII 串行端口指令](#) 参考页数 [119](#)

AWT

AWT 将源字符串中的字符写入到外部设备中。

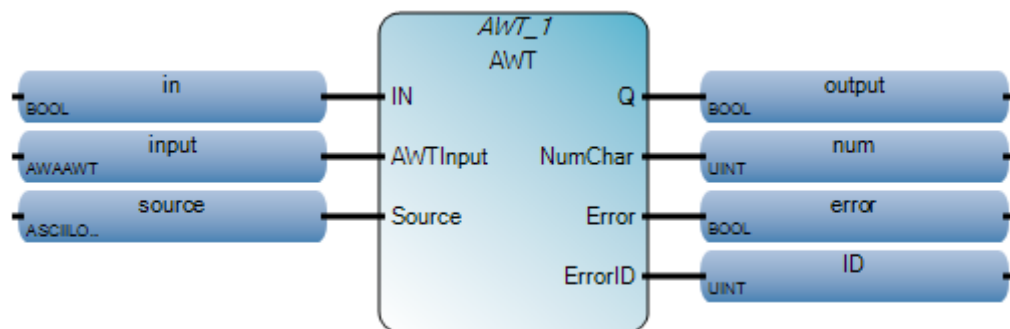


参数

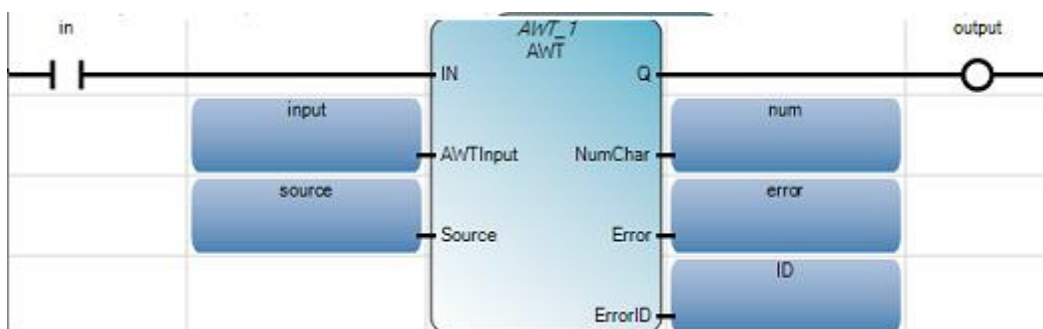
参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿（IN 从 FALSE 变为 TRUE），则启动功能块，前提是上一操作已完成。
AWTInput	输入	AWAAWT	要操作的通道和长度。 请参见 AWAAWT 数据类型 参考页数 139 。
Source	输入	ASCIILOCADDR	源字符串：字符数组。
Q	输出	BOOL	FALSE - 功能块未完成。 TRUE - 功能块完成。
NumChar	输出	UINT	表示传输的字符数。当传输完成并且 Q 为 TRUE 时将会更新。如果源字符串的长度短于请求的长度，则 NumChar 可能短于传输所要求的长度。
Error	输出	BOOL	FALSE - 没有错误。 TRUE - 检测到错误。
ErrorID	输出	UINT	请参见 ABL 错误代码 参考页数 135 。

AWT 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1  AWT_1(in, input, source);
2  output := AWT_1.Q;
3  num := AWT_1.NumChar;
4  error := AWT_1.Error;
5  ID := AWT_1.ErrorID;

```

AWT_1 {

void **AWT_1**(BOOL IN, AWAAWT AWTInput, ASCIILOCADDR Source, UINT __ADI_Source)
Type : AWT, Write characters from a source string to an external device.

另请参见

[ASCII 串行端口指令](#) 参考页数 [119](#)

ASCII 参数详细信息

以下主题提供 ASCII 参数和结构化数据类型的附加详细信息。

[ABL 错误代码](#) 参考页数 [135](#)

[ABLAB 数据类型](#) 参考页数 [136](#)

[ACL 数据类型](#) 参考页数 [137](#)

[AHL ChannelSts 数据类型](#) 参考页数 [137](#)

[AHLI 数据类型](#) 参考页数 [138](#)

[ARDARL 数据类型](#) 参考页数 [139](#)

[AWAAWT 数据类型](#) 参考页数 [139](#)

另请参见

[ASCII 串行端口指令](#) 参考页数 [119](#)

ABL 错误代码

下表描述了 ABL 错误代码。

错误代码	错误描述
02	操作无法完成，因为调制解调器已脱机。
03	传输无法完成，因为清除发送信号已丢失。
05	当尝试完成 ASCII 传输时，检测到系统模式 (DF1) 通信。
06	检测到非法参数。
07	无法完成 ASCII 发送或接收，因为已通过通道配置对话框关闭通道配置。
08	无法完成 ASCII 写入，因为 ASCII 传输已在进行。
09	当前的通道配置不支持请求的 ASCII 通信。
10	已设置 Cancel，从而停止指令的执行。不需要任何操作。
11	字符串的请求长度无效、为负数、大于 82 或为 0。适用于 ARD 参考页数 127 和 ARL 参考页数 129 功能块。
13	控制块中请求的 (.LEN) 长度为负数或大于存储在源字符串中的字符串大小。适用于 AWA 参考页数 131 和 AWT 参考页数 133 功能块。

14	ACL 参考页数 123 功能块已被中止。
15	通道配置模式已被更改。

另请参见

[ASCII 参数详细信息](#) 参考页数 [135](#)

[ABLACB 数据类型](#) 参考页数 [136](#)

ABLACB 数据类型

下表描述了 ABLACB 数据类型。

参数	数据类型	说明
通道	UINT	串行端口号： • 嵌入式串行端口为 2，或 • 安装在插槽 1 到插槽 5 中的串行端口插件为 5 到 9： • 插槽 1 为 5 • 插槽 2 为 6 • 插槽 3 为 7 • 插槽 4 为 8 • 插槽 5 为 9
TriggerType	USINT	表示以下之一： • 0：触发一次 Msg（当 IN 从 False 转为 True 时） • 1：当 IN 为 True 时，连续触发 Msg • 其他值：保留
取消	BOOL	当此输入设置为 TRUE 时，此功能块不会执行。

另请参见

[ASCII 参数详细信息](#) 参考页数 [135](#)

[ABL 错误代码](#) 参考页数 [135](#)

ACL 数据类型

下表描述了 ACL 数据类型。

参数	数据类型	说明
通道	UINT	串行端口号： • 嵌入式串行端口为 2，或 • 安装在插槽 1 到插槽 5 中的串行端口插件为 5 到 9： • 插槽 1 为 5 • 插槽 2 为 6 • 插槽 3 为 7 • 插槽 4 为 8 • 插槽 5 为 9
RXBuffer	BOOL	当设置为 TRUE 时清除接收缓冲区，并从 ASCII 队列中删除接收 ASCII 功能块（ARL 和 ARD）。
TXBuffer	BOOL	当设置为 TRUE 时清除传输缓冲区，并从 ASCII 队列中删除传输 ASCII 功能块（AWA 和 AWT）。

另请参见

[ASCII 参数详细信息](#) 参考页数 [135](#)

AHL ChannelSts 数据类型

下表描述了 AHL ChannelSts 数据类型。

参数	数据类型	说明
DTRstatus	UINT	用于 DTR 信号（已保留）
DCDstatus	UINT	用于 DCD 信号（字的位 3） 1 表示活动
DSRstatus	UINT	用于 DSR 信号（已保留）
RTSstatus	UINT	用于 RTS 信号（字的位 1） 1 表示活动

参数	数据类型	说明
CTSstatus	UINT	用于 CTS 信号 (字的位 0) 1 表示活动

另请参见

[ASCII 参数详细信息](#) 参考页数 [135](#)

[AHLI 数据类型](#) 参考页数 [138](#)

AHLI 数据类型

下表描述了 AHL 数据类型。

参数	数据类型	说明
通道	UINT	串行端口号： • 嵌入式串行端口为 2，或 • 安装在插槽 1 到插槽 5 中的串行端口插件为 5 到 9： • 插槽 1 为 5 • 插槽 2 为 6 • 插槽 3 为 7 • 插槽 4 为 8 • 插槽 5 为 9
ClrRts	BOOL	用于重置 RTS 控制行。
SetRts	BOOL	用于置位 RTS 控制行。
取消	BOOL	当此输入设置为 TRUE 时，此功能块不会执行。

另请参见

[ASCII 参数详细信息](#) 参考页数 [135](#)

[AHL ChannelSts 数据类型](#) 参考页数 [137](#)

ARDARL 数据类型

下表描述了 ARDARL 数据类型。

参数	数据类型	说明
通道	UINT	串行端口号： • 嵌入式串行端口为 2，或 • 安装在插槽 1 到插槽 5 中的串行端口插件为 5 到 9： • 插槽 1 为 5 • 插槽 2 为 6 • 插槽 3 为 7 • 插槽 4 为 8 • 插槽 5 为 9
Length	UINT	您希望从缓冲区读取的字符数（最大值为 82）。
取消	BOOL	当此输入设置为 TRUE 时，此功能块不会执行。如果已执行，则操作会停止。

另请参见

[ASCII 参数详细信息](#) 参考页数 [135](#)

AWAAWT 数据类型

下表描述了 AWAAWT 数据类型。

参数	数据类型	说明
通道	UINT	串行端口号： • 嵌入式串行端口为 2，或 • 安装在插槽 1 到插槽 5 中的串行端口插件为 5 到 9： • 插槽 1 为 5 • 插槽 2 为 6 • 插槽 3 为 7 • 插槽 4 为 8 • 插槽 5 为 9
Length	UINT	您想要写入缓冲区的字符数（最大值为 82）。 注意：如果您将 Length 设置为 0，则 AWA 将 0 个字节的用户数据和 2 个字节的附加字符发送到缓冲区。
取消	BOOL	当此输入设置为 TRUE 时，此功能块不会执行。如果已执行，则操作会停止。

另请参见

[ASCII 参数详细信息](#) 参考页数 [135](#)

二进制指令

二进制指令执行数学运算，即组合两个元素以获得单个结果。

运算符	说明
AND_MASK 参考页数 141	整型位到位 AND_MASK
NOT_MASK 参考页数 143	整型位到位取反 NOT_MASK
OR_MASK 参考页数 145	整型位到位 OR_MASK
ROL 参考页数 147	向左旋转整型值
ROR 参考页数 149	向右旋转整型值
SHL 参考页数 151	向左旋转整型值
SHR 参考页数 153	向左移动整型值
XOR_MASK 参考页数 155	整型位到位异或掩码

AND_MASK

整型位到位与掩码。



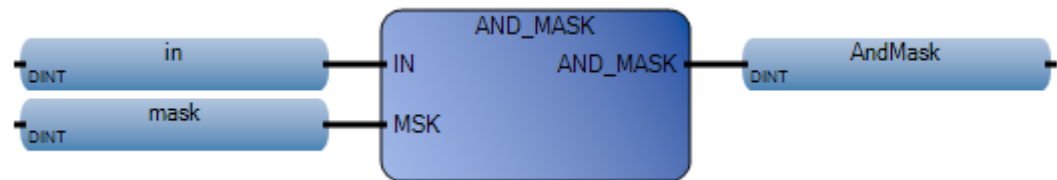
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行整型与位到位掩码计算。 当 EN = FALSE 时，不执行计算。

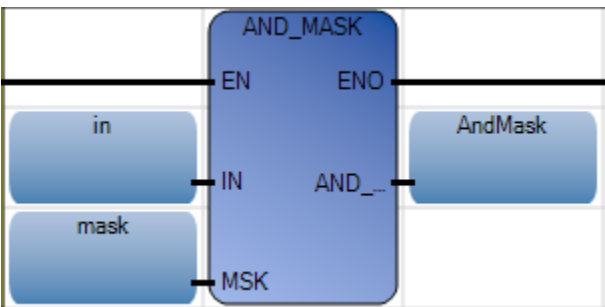
参数	参数类型	数据类型	说明
IN	输入	DINT	必须具有整型格式。
MSK	输入	DINT	必须具有整型格式。
AND_MAS K	输出	DINT	IN 与 MSK 之间的位到位逻辑与。
ENO	输出	BOOL	启用输出。

AND_MASK 函数语言示例

功能块图



梯形图



结构化文本

```
1  in := 5;  
2  mask := 6;  
3  AndMask := AND_MASK(in, mask);
```

AND_MASK(

DINT AND_MASK(DINT IN, DINT MSK)
Analog bit to bit AND mask

(* 与之等效的 ST : *)

parity := AND_MASK (xvalue, 1); (* 如果 xvalue 为奇数 , 则为 1 *)

result := AND_MASK (16#abc, 16#f0f); (* 等于 16#a0c *)

结果

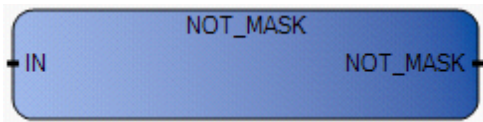


另请参见

[二进制指令](#) 参考页数 [141](#)

NOT_MASK

整型位到位取反掩码，NOT_MASK 将反转参数值。



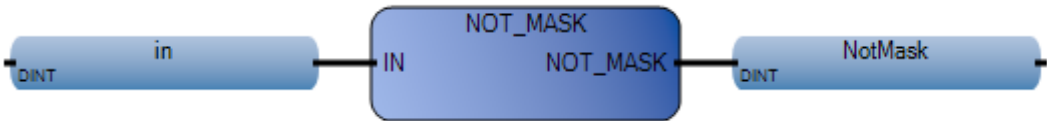
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行位到位非运算掩码计算。 当 EN = FALSE 时，不执行计算。
IN	输入	DINT	必须具有整型格式。
NOT_MASK	输出	DINT	32 位 IN 上的位到位非运算。

参数	参数类型	数据类型	说明
ENO	输出	BOOL	启用输出。

NOT_MASK 函数语言示例

功能块图



梯形图



结构化文本

```
1 | in := 6;  
2 | NotMask := NOT_MASK(in);
```

```
NOT_MASK(  
DINT NOT_MASK(DINT IN)  
bit to bit negation
```

(* 与之等效的 ST : *)

```
result := NOT_MASK (16#1234);
```

(* 结果为 16#FFFF_EDCB *)

结果

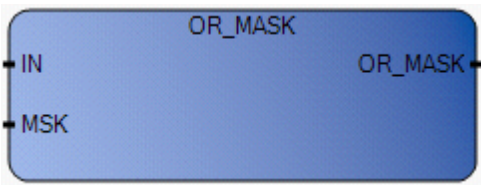


另请参见

[二进制指令](#) 参考页数 [141](#)

OR_MASK

整型或位到位掩码，OR_MASK 将启用位。

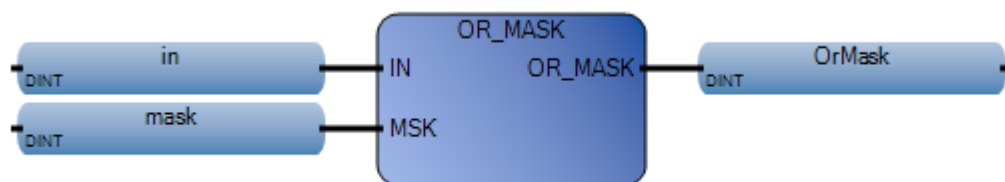


参数

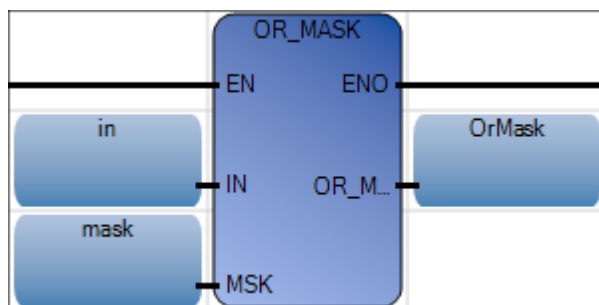
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行整型或位到位掩码计算。 当 EN = FALSE 时，不执行计算。
IN	输入	DINT	必须具有整型格式。
MSK	输入	DINT	必须具有整型格式。
OR_MASK	输出	DINT	IN 与 MSK 之间的位到位逻辑或。
ENO	输出	BOOL	启用输出。

OR_MASK 函数语言示例

功能块图



梯形图



结构化文本

```

1  in := 3;
2  mask := 6;
3  OrMask := OR_MASK(in, mask);

```

OR_MASK (

DINT OR_MASK(DINT IN, DINT MSK)
Analog bit to bit OR mask

(* 与之等效的 ST : *)

parity := OR_MASK (xvalue, 1); (* 使值始终为奇数 *)

result := OR_MASK (16#abc, 16#f0f); (* 等于 16#fbf *)

结果

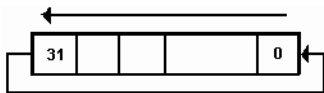


另请参见

[二进制指令](#) 参考页数 [141](#)

ROL

对于 32 位整数，ROL 将整数位旋转到左侧。

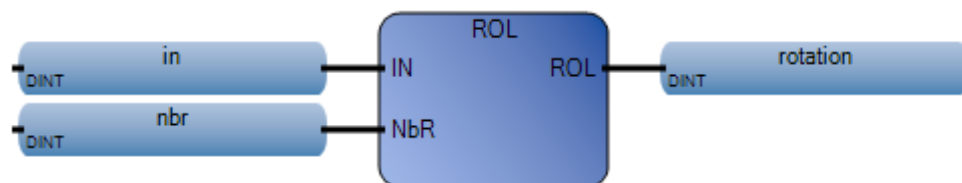


参数

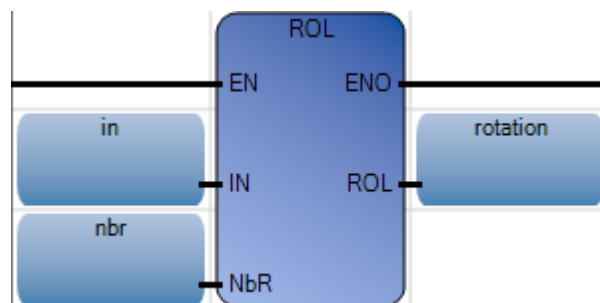
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行向左旋转位整数值计算。 当 EN = FALSE 时，不执行计算。
IN	输入	DINT	整数值。
NbR	输入	DINT	1 位旋转的数量（位于集合 [1..31] 内）。
ROL	输出	DINT	左旋转的值。当 NbR <= 0 时，不发生任何更改。
ENO	输出	BOOL	启用输出。

ROL 函数语言示例

功能块图



梯形图



结构化文本

```

1 | in := 123;
2 | nbr := 2;
3 | rotation := ROL(in, nbr);

```

ROL {
 DINT ROL(DINT IN, DINT NbR)
 Rotate Left

(* 与之等效的 ST : *)

result := ROL (register, 1);

(* register = 2#0100_1101_0011_0101*)

(* result = 2#1001_1010_0110_1010 *)

结果

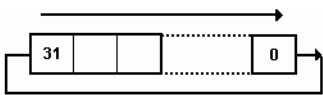


另请参见

[二进制指令](#) 参考页数 [141](#)

ROR

对于 32 位整数，ROR 将整数位旋转到右侧。

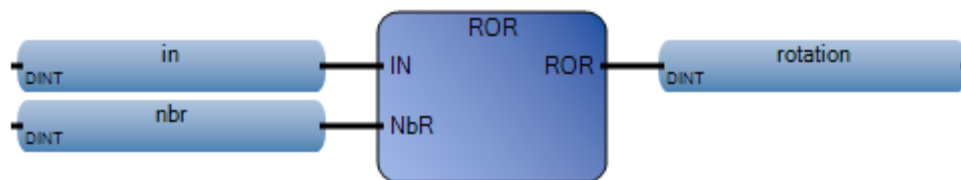


参数

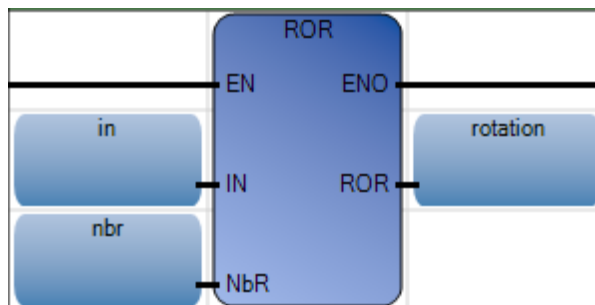
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行向右旋转位整数值计算。 当 EN = FALSE 时，不执行计算。
IN	输入	DINT	任何整数值。
NbR	输入	DINT	1 位旋转的数量（位于集合 [1..31] 内）。
ROR	输出	DINT	右旋转的值。如果 NbR <= 0，则无效。
ENO	输出	BOOL	启用输出。

ROR 函数语言示例

功能块图



梯形图



结构化文本

```

1  in := 123;
2  nbr := 2;
3  rotation := ROR(in, nbr);

```

ROR (

DINT ROR(DINT IN, DINT NbR)
Rotate Right

(* 与之等效的 ST : *)

```
result := ROR (register, 1);
```

(* register = 2#0100_1101_0011_0101 *)

(* result = 2#1010_0110_1001_1010 *)

结果

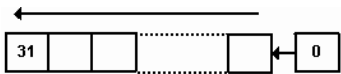


另请参见

[二进制指令](#) 参考页数 [141](#)

SHL

对于 32 位整数，SHL 将整数向左移动，并在最低有效位中置 0。

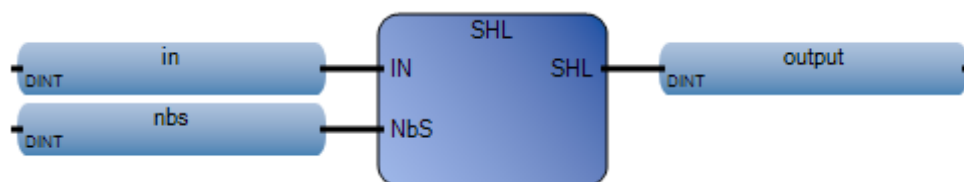


参数

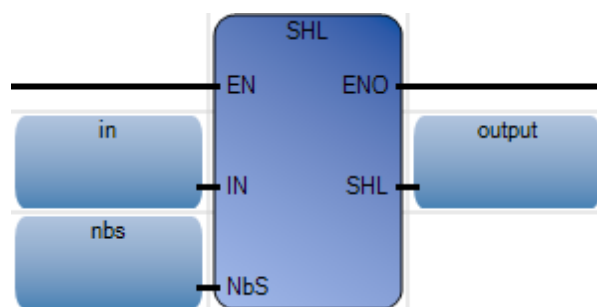
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，将整数向左移动。 当 EN = FALSE 时，不执行整数移动。
IN	输入	DINT	任何整数值。
NbS	输入	DINT	1 位移动的数量（位于集合 [1..31] 内）。
SHL	输出	DINT	左移动的值。如果 NbR <= 0，则无效。如果值为 0，则替换最低有效位。
ENO	输出	BOOL	启用输出。

SHL 函数语言示例

功能块图



梯形图

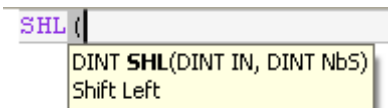


结构化文本

```

1  in := 123;
2  nbs := 2;
3  output := SHL(in, nbs);

```



(* 与之等效的 ST : *)

```
result := SHL (register,1);
```

(* register = 2#0100_1101_0011_0101 *)

(* result = 2#1001_1010_0110_1010 *)

结果

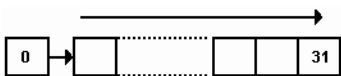


另请参见

[二进制指令](#) 参考页数 [141](#)

SHR

对于 32 位整数，SHR 将整数向右移动，并在最高有效位中置 0。



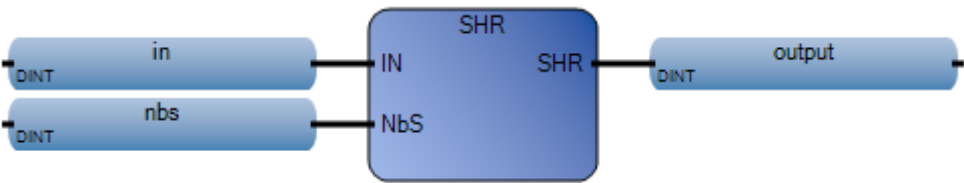
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，将整数向右移动。 当 EN = FALSE 时，不执行整数移动。
IN	输入	DINT	任何整数值。
NbS	输入	DINT	1 位移动的数量（位于集合 [1..31] 内）。
SHR	输出	DINT	右移动的值。如果 NbR <= 0，则无效。如果值为 0，则替换最高有效位。

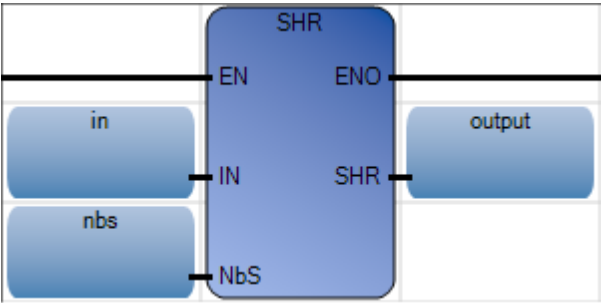
参数	参数类型	数据类型	说明
ENO	输出	BOOL	启用输出。

SHR 函数语言示例

功能块图



梯形图



结构化文本

```
1  in := 123;  
2  nbs := 2;  
3  output := SHR(in, nbs);
```

SHR(

DINT **SHR**(DINT IN, DINT NbS)
Shift Right

(* 与之等效的 ST : *)

result := SHR (register,1);

(* register = 2#1100_1101_0011_0101 *)

(* result = 2#0110_0110_1001_1010 *)

结果



另请参见

[二进制指令](#) 参考页数 [141](#)

XOR_MASK

整数异或位到位掩码，XOR_MASK 返回反转的位值。

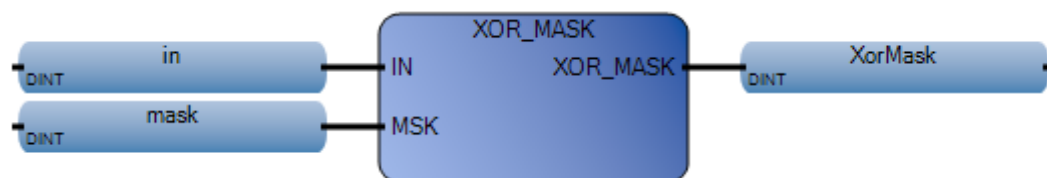


参数

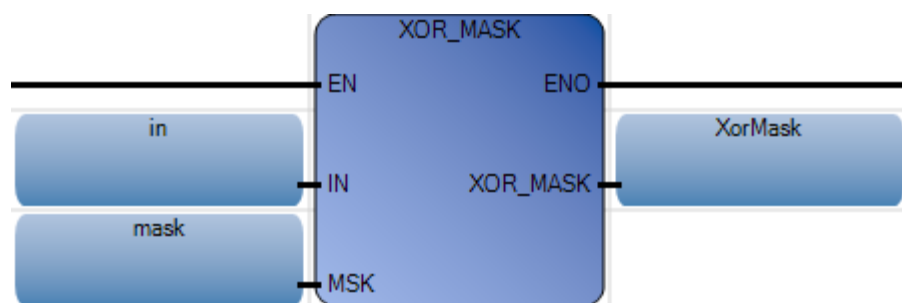
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行异或位到位掩码计算。 当 EN = FALSE 时，不执行计算。
IN	输入	DINT	必须具有整型格式。
MSK	输入	DINT	必须具有整型格式。
XOR_MASK	输出	DINT	IN 与 MSK 之间的位到位逻辑异或。
ENO	输出	BOOL	启用输出。

XOR_MASK 语言示例

功能块图



梯形图



结构化文本

```

1  in := 5;
2  mask := 6;
3  XorMask := XOR_MASK(in, mask);

```

XOR_MASK(

DINT XOR_MASK(DINT IN, DINT MSK)
Analog bit to bit Exclusive OR mask

(* 与之等效的 ST : *)

```
crc32 := XOR_MASK (prevcrc, nextc);
```

```
result := XOR_MASK (16#012, 16#011); (* 等于 16#003 *)
```


结果



另请参见

[二进制指令](#) 参考页数 [141](#)

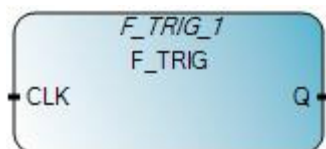
布尔指令

布尔指令根据输入的某些逻辑计算结果来确定值输出。模块输出既可以由程序直接控制，也可以由模块使用布尔指令独立控制。

函数	说明
MUX4B 参考页数 179	与 MUX4 类似，但是可接受 BOOL 类型输入和输出 BOOL 类型值
MUX8B 参考页数 174	与 MUX8 类似，但是可接受 BOOL 类型输入和输出 BOOL 类型值
TTABLE 参考页数 170	根据输入组合提供输出值
功能块	说明
F_TRIG 参考页数 159	下降沿检测
RS 参考页数 164	双稳态重置基准
R_TRIG 参考页数 161	上升沿检测
SR 参考页数 168	设置双稳态基准
运算符	说明
AND 参考页数 166	在两个或多个值之间执行布尔与操作。
非 参考页数 167	对于布尔表达式，将值转换为反值。
XOR 参考页数 167	两个值的布尔异或。
OR 参考页数 165	两个或更多值的布尔或。

F_TRIG

F_TRIG 检测布尔变量的下降沿。

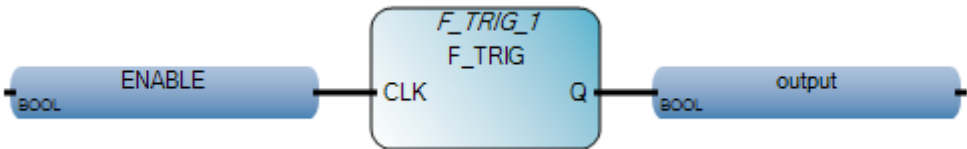


参数

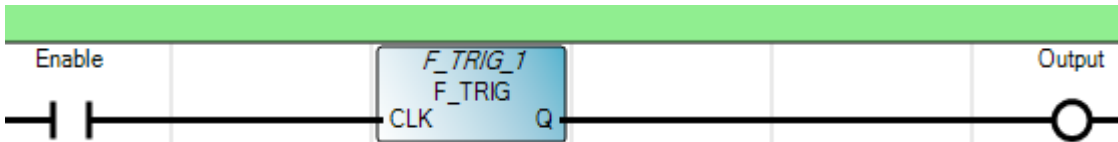
参数	参数类型	数据类型	说明
CLK	输入	BOOL	任何布尔变量。
Q	输出	BOOL	当 CLK 从 TRUE 变为 FALSE 时为 TRUE。 在所有其他情况下均为 FALSE。

F_TRIG 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 F_TRIG_1(ENABLE);
2 IF F_TRIG_1.Q THEN
3   output := TRUE;
4 END_IF;
```

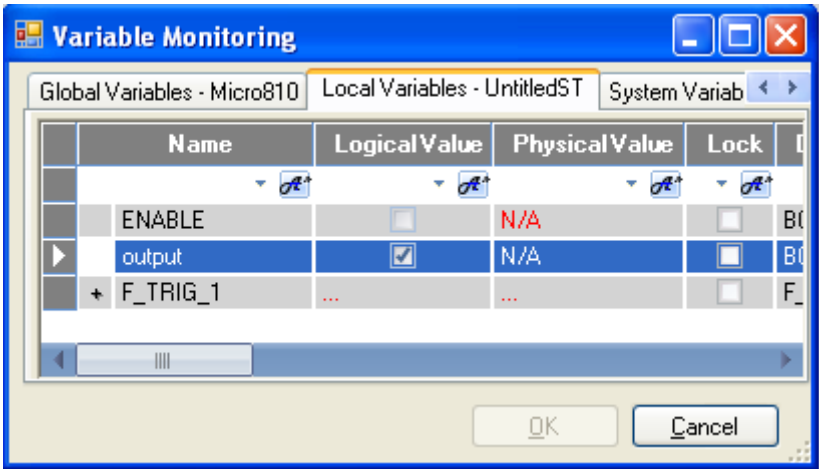
```
F_TRIG_1(
void F_TRIG_1(BOOL CLK)
Type : F_TRIG, Falling edge detection
```

(* 与之等效的 ST : F_TRIG1 是 F_TRIG 块的实例 *)

```
F_TRIG1(cmd);
```

```
nb_edge := ANY_TO_DINT(F_TRIG1.Q) + nb_edge;
```

结果

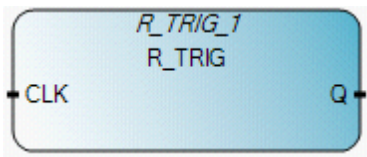


另请参见

[布尔指令](#) 参考页数 [159](#)

R_TRIG

R_TRIG 检测布尔变量的上升沿。

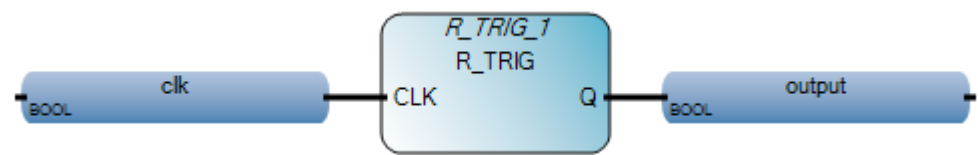


参数

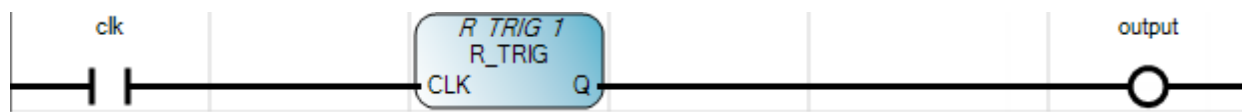
参数	参数类型	数据类型	说明
CLK	输入	BOOL	任何布尔变量。
Q	输出	BOOL	当 CLK 从 FALSE 上升为 TRUE 时为 TRUE。 在所有其他情况下均为 FALSE。

R_TRIG 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 R_TRIG_1(cmd);
2 output := R_TRIG_1.Q;
```

R_TRIG_1{

void R_TRIG_1(BOOL CLK)
Type : R_TRIG, Rising edge detection

(* 与之等效的 ST : R_TRIG1 是 R_TRIG 块的实例 *)

```
R_TRIG1(cmd);
nb_edge := ANY_TO_DINT(R_TRIG1.Q) + nb_edge;
```

结果



另请参见

[布尔指令](#) 参考页数 [159](#)

RS

RS 重置主导双稳态。

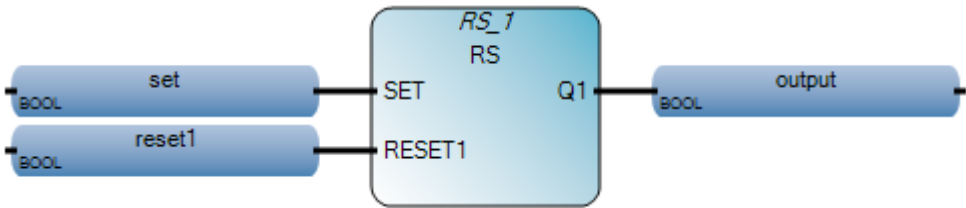


参数

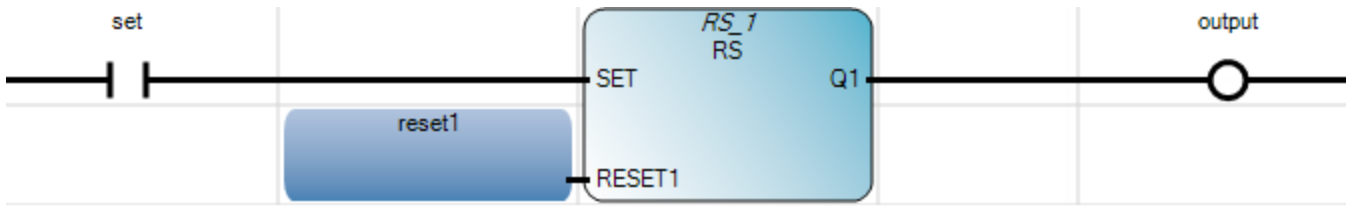
参数	参数类型	数据类型	说明
SET	输入	BOOL	如果为 TRUE，则将 Q1 设置为 TRUE。
RESET1	输入	BOOL	如果为“真”，则将 Q1 设置为“假”（优先）。
Q1	输出	BOOL	布尔内存状态。

RS 功能块语言示例

功能块图 (FBD)

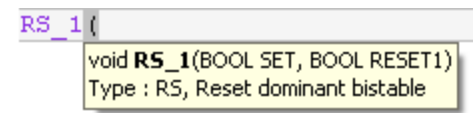


梯形图 (LD)



结构化文本 (ST)

```
1  set := TRUE;  
2  reset1 := FALSE;  
3  RS_1(set, reset1);  
4  output := RS_1.Q1;
```

(* 与之等效的 ST : RS1 是 RS 块的实例 *)

```
RS1(start_cmd, (stop_cmd OR alarm));
command := RS1.Q1;
```

结果

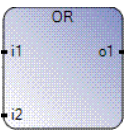


另请参见

[布尔指令](#) 参考页数 [159](#)

OR

两个或更多值的布尔或。



OR 运算

OR 运算符支持附加输入。

参数

参数	参数类型	数据类型	说明
i1	输入	BOOL	
i2	输入	BOOL	
o1	输出	BOOL	输入项的布尔或。

OR 运算符 ST 语言示例

(* 与之等效的 ST : *)

```
bo10 := bi101 OR NOT (bi102);  
bo5  := (bi51 OR bi52) OR bi53;
```

另请参见

[布尔指令](#) 参考页数 [159](#)

AND

AND 在两个或多个值之间执行布尔“与”操作。



AND 运算

AND 运算符支持附加输入。

参数

参数	参数类型	数据类型	说明
i1	输入	BOOL	布尔数据类型中的值。
i2	输入	BOOL	布尔数据类型中的值。
o1	输出	BOOL	对输入值执行布尔与操作所得结果。

AND 运算符 ST 语言示例

(* 与之等效的 ST : *)

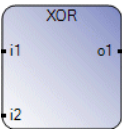
```
bo10 := bi101 AND NOT (bi102);  
bo5  := (bi51 AND bi52) AND bi53;
```

另请参见

[布尔指令](#) 参考页数 [159](#)

XOR

两个值的布尔异或。



参数

参数	参数类型	数据类型	说明
i1	输入	BOOL	
i2	输入	BOOL	
o1	输出	BOOL	两个输入项的布尔异或。

XOR 运算符 ST 语言示例

(* 与之等效的 ST : *)

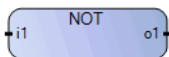
```
bo10 := bi101 XOR NOT (bi102);  
bo5  := (bi51 XOR bi52) XOR bi53;
```

另请参见

[布尔指令](#) 参考页数 [159](#)

非

对于布尔表达式，NOT 将值转换为反值。



参数

参数	参数类型	数据类型	说明
i1	输入	BOOL	任何布尔值或复杂表达式。
o1	输出	BOOL	TRUE (当 IN 为 FALSE 时)。 FALSE (当 IN 为 TRUE 时)。

NOT 运算符 ST 语言示例

(* 与之等效的 ST : *)

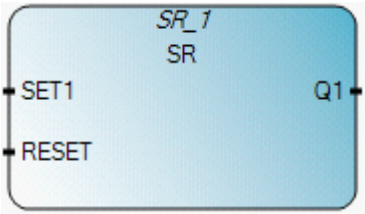
bo10 := bi101 NOT;

另请参见

[布尔指令](#) 参考页数 [159](#)

SR

SR 设置主导双稳态。



参数

参数	参数类型	数据类型	说明
SET1	输入	BOOL	如果为 TRUE , 则将 Q1 设置为 TRUE (优先)。
RESET	输入	BOOL	如果为 TRUE , 则将 Q1 重置为 FALSE。
Q1	输出	BOOL	布尔内存状态。

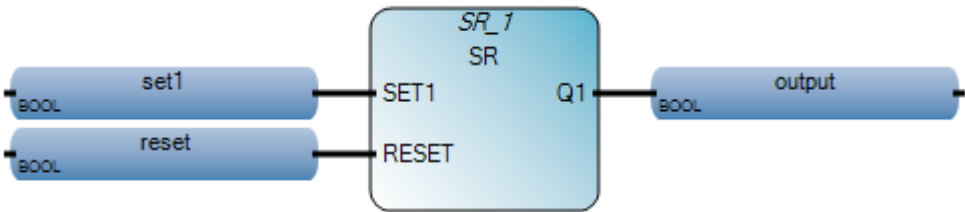
双稳态基准示例

Set1	Reset	Q1	结果 Q1
0	0	0	0
0	0	1	1
0	1	0	0

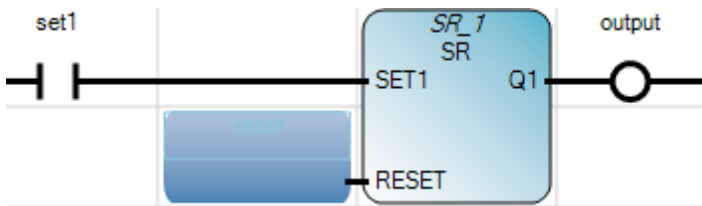
Set1	Reset	Q1	结果 Q1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

SR 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 set1 := TRUE;
2 reset := FALSE;
3 SR_1(set1, reset);
4 output := SR_1.Q1;
```

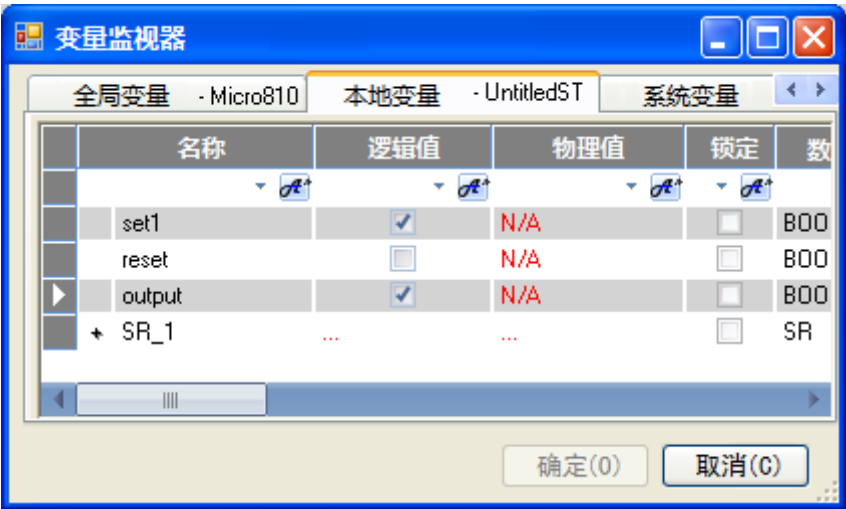
SR_1(

void **SR_1**(BOOL SET1, BOOL RESET)
Type : SR, Set dominant bistable

(* 与之等效的 ST : SR1 是 SR 块的实例 *)

```
SR1((auto_mode & start_cmd), stop_cmd);  
command := SR1.Q1;
```

结果



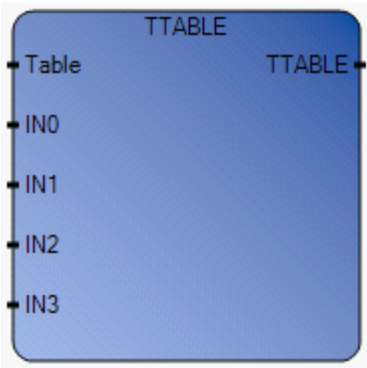
另请参见

[布尔指令](#) 参考页数 [159](#)

TTABLE

TTABLE 函数根据输入组合给出输出值。

如果值为 0xABCD 且 In3 到 In0 对应于数字 7，则 TTABLE 是表中位 7 的值（为 1）。表中的最低有效位是位 0。



参数

参数	参数类型	数据类型	说明
Table	输入	UINT	BOOLEAN 函数的真值表。
IN0	输入	BOOL	任何 BOOL 输入值。
IN1	输入	BOOL	任何 BOOL 输入值。
IN2	输入	BOOL	任何 BOOL 输入值。
IN3	输入	BOOL	任何 BOOL 输入值。
TTABLE	输出	BOOL	基于输入组合的输出值。 请参见 TTABLE 输入组合 参考页数 173 。

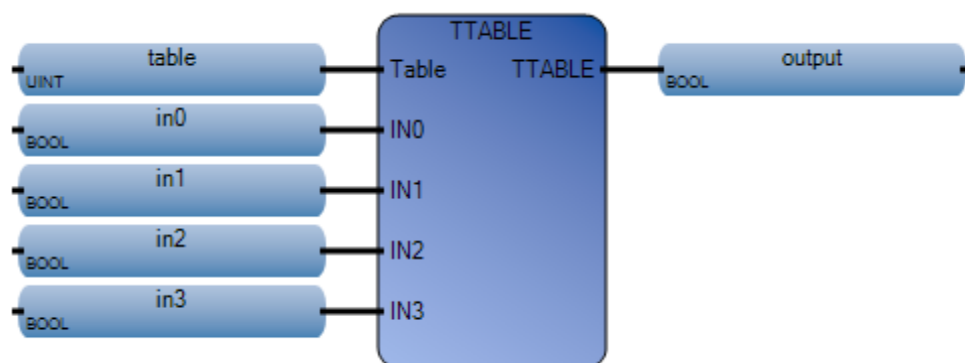
TTABLE 输入组合

该函数具有四个输入，因此有 16 种组合。可以在真值表找到这些组合；对于每个组合，可以调整输出值。可配置组合的数量取决于连接到该函数的输入的数量。例如：

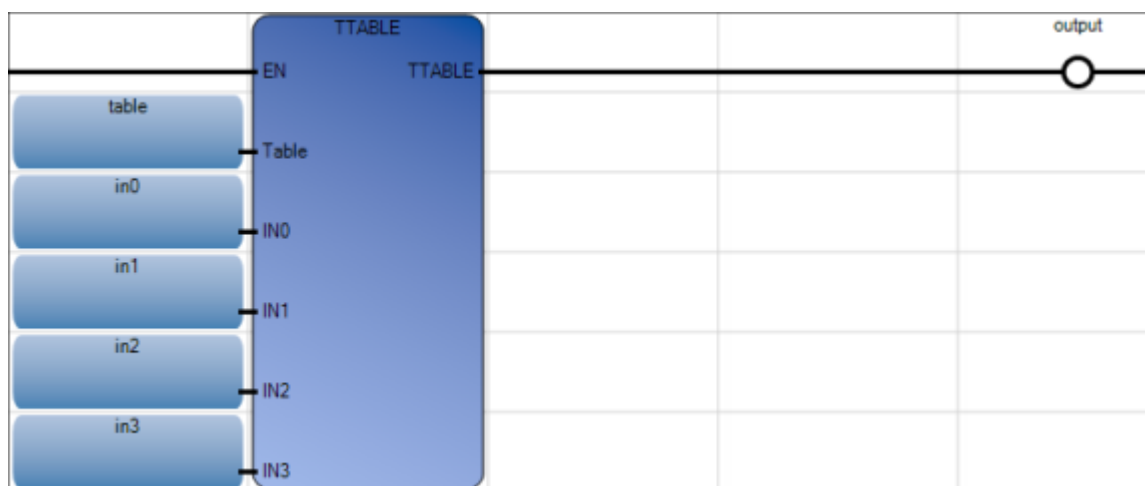
编号	In3	In2	In1	In0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

TTABLE 函数语言示例

功能块图



梯形图



结构化文本

```
1  table := 217;  
2  in0 := TRUE;  
3  in1 := TRUE;  
4  in2 := TRUE;  
5  in3 := FALSE;  
6  output := TTABLE(table, in0, in1, in2, in3);
```


TTABLE {

BOOL **TTABLE**(UINT Table, BOOL IN0, BOOL IN1, BOOL IN2, BOOL IN3)
Provide the value output based on the combination of inputs.

结果



另请参见

[布尔指令](#) 参考页数 [159](#)

TTABLE 输入组合

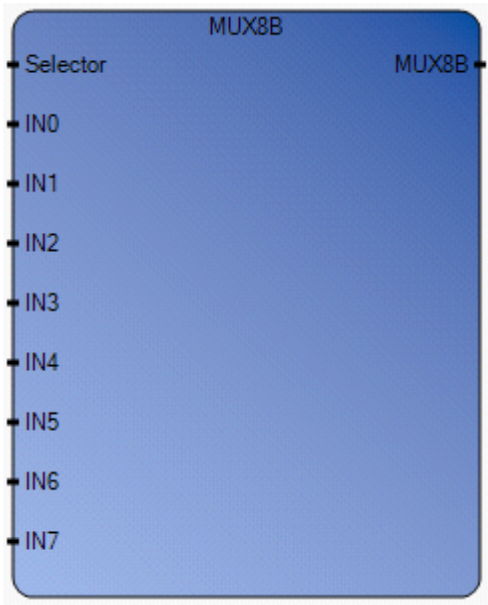
该函数具有四个输入，因此有 16 种组合。可以在真值表找到这些组合；对于每个组合，可以调整输出值。可配置组合的数量取决于连接到该函数的输入的数量。例如：

编号	In3	In2	In1	In0
0	0	0	0	0
1	0	0	0	1

编号	In3	In2	In1	In0
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

MUX8B

MUX8B 生成介于八个 BOOL 类型输入和输出值之间的值。



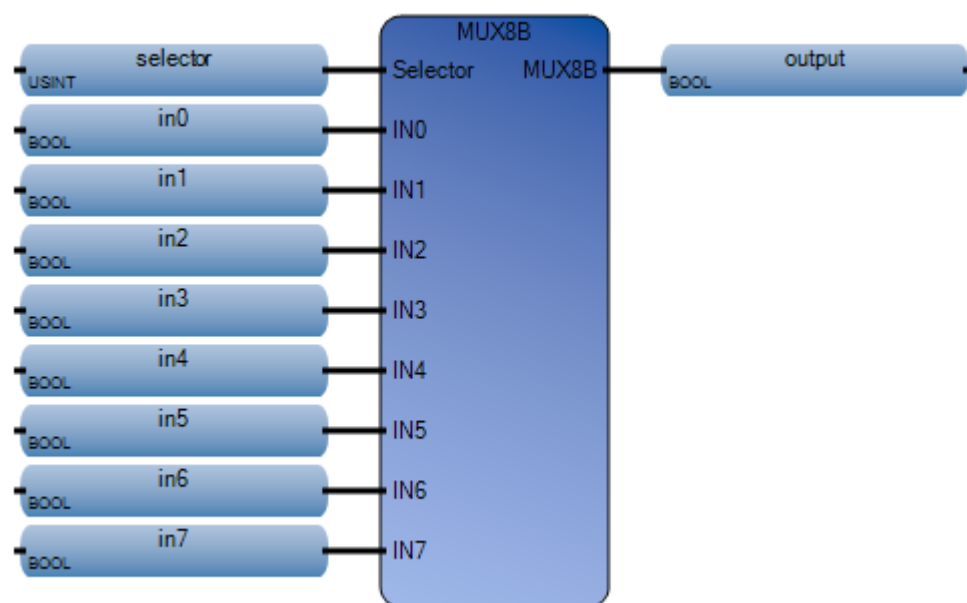
参数

参数	参数类型	数据类型	说明
Selector	输入	USINT	选择器整数值，必须位于集合 [0...7] 中。

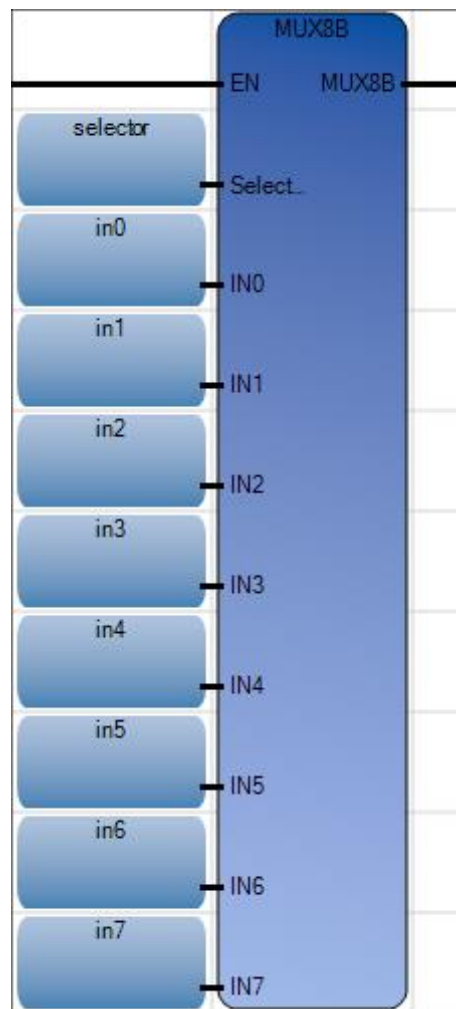
参数	参数类型	数据类型	说明
IN0	输入	BOOL	任何 BOOL 输入值。
IN1	输入	BOOL	任何 BOOL 输入值。
IN2	输入	BOOL	任何 BOOL 输入值。
IN3	输入	BOOL	任何 BOOL 输入值。
IN4	输入	BOOL	任何 BOOL 输入值。
IN5	输入	BOOL	任何 BOOL 输入值。
IN6	输入	BOOL	任何 BOOL 输入值。
IN7	输入	BOOL	任何 BOOL 输入值。
MUX8B	输出	BOOL	可以为下列值： • In0 (如果 Selector = 0) • In1 (如果 Selector = 1) • In2 (如果 Selector = 2) • In3 (如果 Selector = 3) • In4 (如果 Selector = 4) • In5 (如果 Selector = 5) • In6 (如果 Selector = 6) • In7 (如果 Selector = 7) 对于选择器的所有其他值，将为 FALSE。

MUX8B 函数语言示例

功能块图



梯形图



结构化文本

```
1 selector := 7;  
2 in0 := FALSE;  
3 in1 := FALSE;  
4 in2 := FALSE;  
5 in3 := FALSE;  
6 in4 := FALSE;  
7 in5 := FALSE;  
8 in6 := FALSE;  
9 in7 := TRUE;  
10 output := MUX8B(selector, in0, in1, in2, in3, in4, in5, in6, in7);
```

MUX8B{

BOOL **MUX8B**(USINT Selector, BOOL IN0, BOOL IN1, BOOL IN2, BOOL IN3, BOOL IN4, BOOL IN5, BOOL IN6, BOOL IN7)
Multiplexer(8 entries) - accepts BOOL inputs and output value.

(* 与之等效的 ST : *)

range := MUX8 (choice, 1, 5, 10, 50, 100, 500, 1000, 5000);

(* 从 8 个预定义的范围中进行选择 , 如果 choice 为 3 , 则 range 将为 50 *)

结果

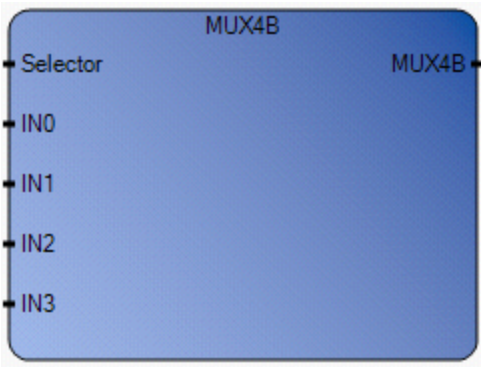


另请参见

[布尔指令](#) 参考页数 [159](#)

MUX4B

MUX4B 生成介于四个 BOOL 类型输入和输出值之间的值。

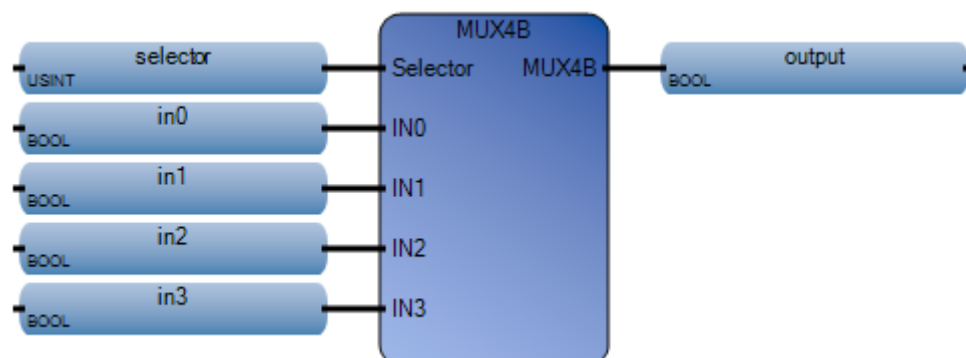


参数

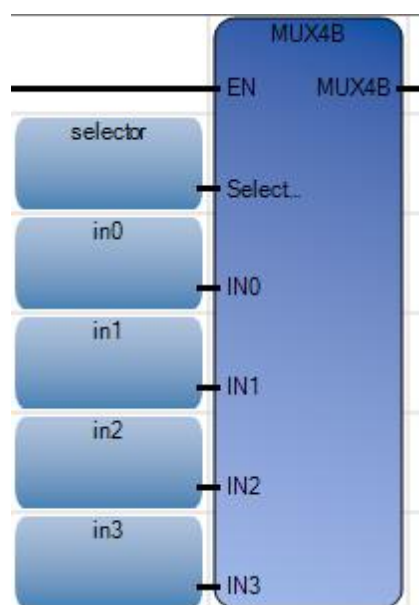
参数	参数类型	数据类型	说明
Selector	输入	USINT	选择器整数值，必须位于集合 [0...3] 中。
IN0	输入	BOOL	任何 BOOL 输入值。
IN1	输入	BOOL	任何 BOOL 输入值。
IN2	输入	BOOL	任何 BOOL 输入值。
IN3	输入	BOOL	任何 BOOL 输入值。
MUX4B	输出	BOOL	可以为下列值： • In0（如果 Selector = 0） • In1（如果 Selector = 1） • In2（如果 Selector = 2） • In3（如果 Selector = 3） 对于选择器的所有其他值，将为 FALSE。

MUX4B 函数语言示例

功能块图



梯形图



结构化文本

```

1 selector := 1;
2 in0 := FALSE;
3 in1 := TRUE;
4 in2 := FALSE;
5 in3 := FALSE;
6 output := MUX4B(selector, in0, in1, in2, in3);

```

MUX4B (

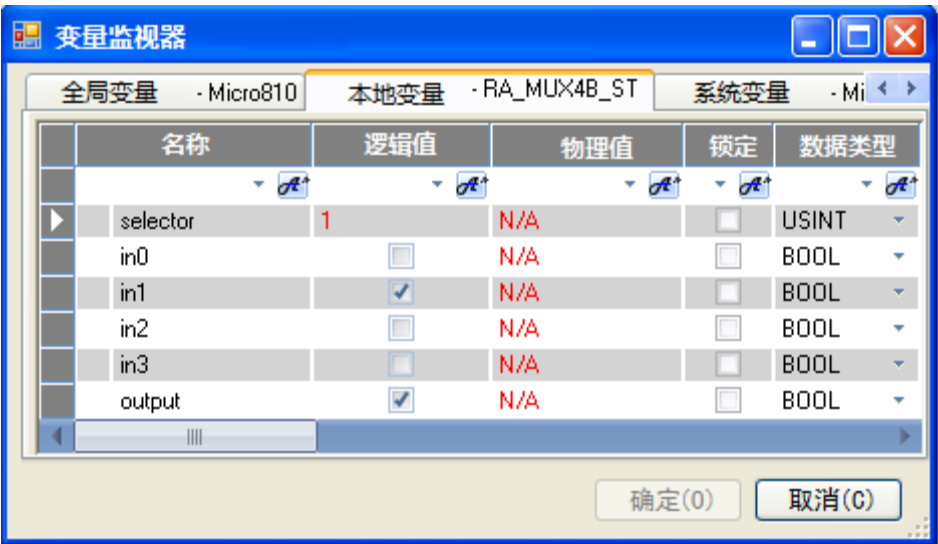
BOOL **MUX4B**(USINT Selector, BOOL IN0, BOOL IN1, BOOL IN2, BOOL IN3)
Multiplexer(4 entries) - accepts BOOL inputs and output value.

(* 与之等效的 ST : *)

range := MUX4 (choice, 1, 10, 100, 1000);

(* 从 4 个预定义的范围中进行选择 , 如果 choice 为 1 , 则 range 将为 10 *)

结果



另请参见

[布尔指令](#) 参考页数 [159](#)

通信指令

通信指令用于读取、写入、比较和转换通信字符串。

功能块	说明
MSG_CIPGENERIC 参考页数 183	发送 CIP 泛型显式消息
MSG_CIPSYMBOLIC 参考页数 195	发送 CIP 符号显式消息
MSG_MODBUS 参考页数 209	发送 Modbus 消息
MSG_MODBUS2 参考页数 217	通过以太网通道发送 MODBUS/TCP 消息

另请参见

[通信协议支持](#) 参考页数 [272](#)

指令块参考

MSG_CIPGENERIC MSG_CIPGENERIC 通过以太网通道或串行端口发送通用工业协议 (CIP) 显式消息。



MSG_CIPGENERIC 操作

每个通道在一次扫描中最多可以处理四个消息请求。对于梯形图程序，将在梯形扫描结束时执行消息请求。

参数

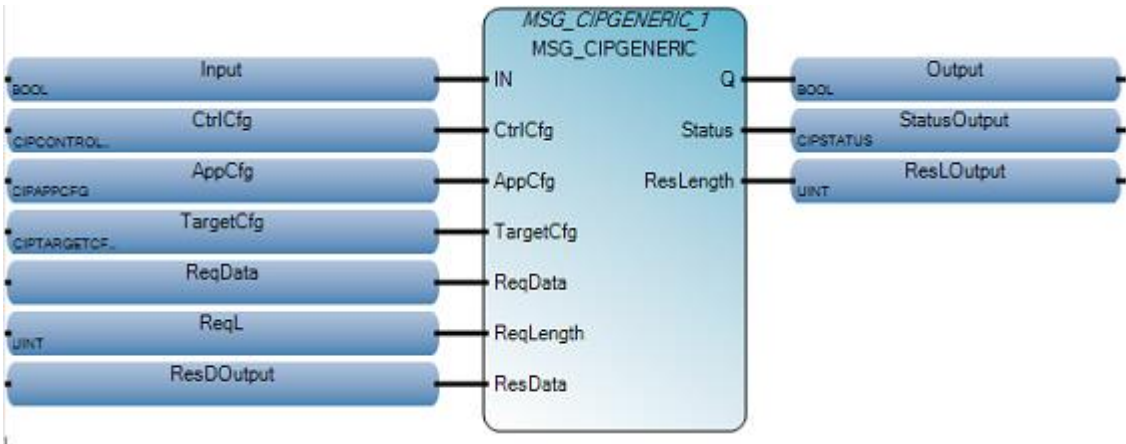
启用 MSG_CIPGENERIC 功能块后，读取操作的接收缓冲区会在 Enable 的上升沿被清除。

参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿（IN 从 FALSE 变为 TRUE），则启动功能块，前提是上一操作已完成。
CtrlCfg	输入	CIPCONTROLCFG	功能块执行控制配置 请参见 CIPCONTROLCFG 数据类型 参考页数 187。
AppCfg	输入	CIPAPPCFG	CIP 服务和应用程序路径 (EPATH) 配置 请参见 CIPAPPCFG 数据类型 参考页数 186。
TargetCfg	输入	CIPTARGETCFG	目标设备配置 请参见 CIPTARGETCFG 数据类型 参考页数 191。
ReqData	输入	USINT[1..1]	CIP 消息请求数据。数组大小不应小于 'ReqLength' 的大小 请参见 MSG_CIPGENERIC 消息行为。
ReqLength	输入	UINT	CIP 消息请求数据长度： • 0 - 490
ResData	输入	USINT[1..1]	CIP 消息响应数据。数组大小不应小于 'ResLength' 的大小。 触发或者重新触发 MSG 时，将清除 ResData 数组中的数据。
Q	输出	BOOL	TRUE - MSG 指令已完成。 FALSE - MSG 指令未完成。
状态	输出	CIPSTATUS	功能块执行状态 触发或者重新触发 MSG 时，Status 中的所有元素均会重置。 请参见 CIPSTATUS 数据类型 参考页数 188。

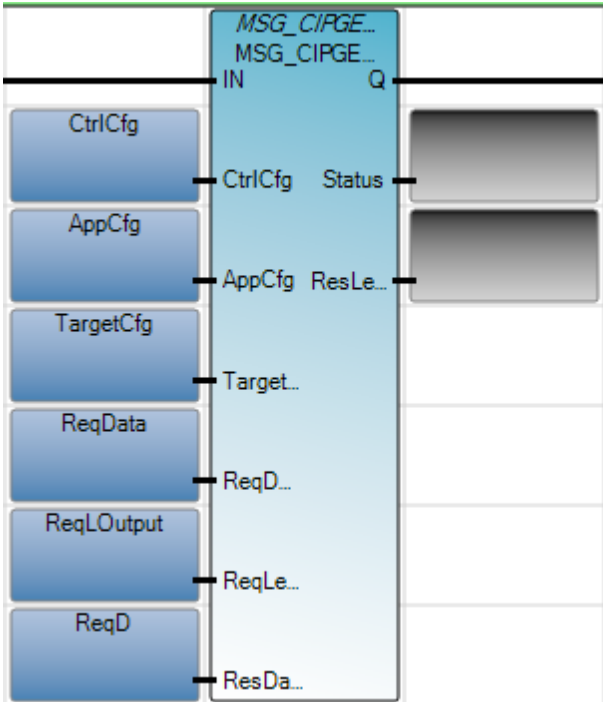
参数	参数类型	数据类型	说明
ResLength	输出	UINT	CIP 消息响应数据长度： • 0 - 490 触发或重新触发 MSG 时，ResLength 重置为 0。

MSG_CIPGENERIC 函数语言示例

功能块图



梯形图



另请参见

[配置显式消息的对象数据值](#) 参考页数 [236](#)

[示例：如何创建 MSG CIPGENERIC 消息程序以从控制器中读取数据](#)
参考页数 [239](#)

[消息执行过程和时间图](#) 参考页数 [225](#)

[通信指令](#) 参考页数 [183](#)

CIPAPPCFG 数据类型

下表描述了 CIPAPPCFG 数据类型。

参数	数据类型	说明
Service	USINT	服务代码： 1 - 127

参数	数据类型	说明
Class	UINT	逻辑段类 ID 值： 1 - 65535
实例	UDINT	逻辑段实例 ID 值： 0 - 4294967295
属性	UINT	逻辑段特性 ID 值： 1 - 65535，0 - 未使用特性 ID
MemberCnt	USINT	成员 ID 计数。使用的最大成员 ID 值： 1 - 3，0 - 未使用成员 ID
MemberId	UINT[3]	成员 ID 值： 0 - 65535

另请参见

[MSG_CIPSYMBOLIC](#) 参考页数 [195](#)

[MSG_CIPGENERIC](#) 参考页数 [183](#)

CIPCONTROLCFG 数据类型

下表描述了 CIPCONTROLCFG 数据类型。

参数	数据类型	说明
取消	BOOL	TRUE - 取消功能块的执行。 启用消息后将清除位。
TriggerType	USINT	表示以下之一： 0：触发一次 Msg（当 IN 从 False 转为 True 时） 1 到 65535：循环触发值（毫秒）。当 IN 为 True 时，定期触发 Msg。 将该值设为 1 以尽快触发 Msg。
StrMode	USINT	保留供将来使用。

消息取消

如果 Cancel 参数已设置，消息已启用（已设置 EN 位）并且未完成（未设置 DN 位），则消息执行会中止并设置 ER 位。

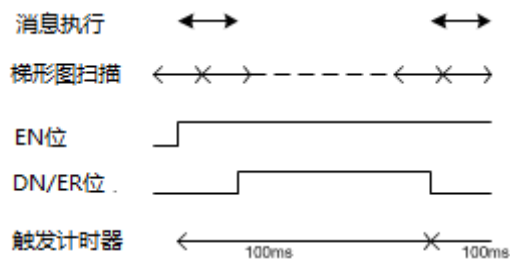
CIP 消息触发

通过将 TriggerType 参数设置为非零值，可以定期触发 CIP 消息。下表描述了在使用 TriggerType 参数时发生的情况。

操作	结果
消息已启用	触发器计时器启动
触发器计时器在消息完成前到期	消息在下一个梯形扫描周期中立即触发。
消息在触发器时间到期前完成	消息在触发器时间到期时触发。

示例：消息触发

在下例中，TriggerType 值设为 100。



另请参见

[MSG CIPSYMBOLIC](#) 参考页数 [195](#)

[MSG CIPGENERIC](#) 参考页数 [183](#)

CIPSTATUS 数据类型

下表描述了 CIPSTATUS 数据类型。

参数	数据类型	说明
Error	BOOL	如果功能块执行遇到错误条件，该位设为 TRUE。
ErrorID	UINT	错误代码值。 请参见 CIPSTATUS 错误代码 参考页数 190 。

SubErrorID	UINT	子错误代码值。 请参见 CIPSTATUS 错误代码 参考页数 190 。
ExtErrorID	UINT	CIP 扩展状态错误代码值。
StatusBits	UINT	此参数可用于验证控制位： • 位 0：EN - Enable • 位 1：EW - Enable Wait • 位 2：ST - Start • 位 3：ER - Error • 位 4：DN - Done • 其他位保留 • 请参见 CIPSTATUS 状态位 参考页数 189。

另请参见

[MSG CIPSYMBOLIC](#) 参考页数 [195](#)

[MSG CIPGENERIC](#) 参考页数 [183](#)

CIPSTATUS 状态位

CIPSTATUS 状态位的设置基于消息执行状态、通信缓冲以及梯级条件。

-	-	-	-	-	-	-	-	-	-	-	4	3	2	1	0

位	名称	说明	行为
0	EN	启用	当梯级变为 true 时设置，并且将一直设为该值，直至 DN 位或 ER 位已设置且梯级变为 false。
1	EW	Enable Wait	当为消息请求分配通信缓冲时设置。在设置 ST 位后清除。
2	ST	启动	当消息已传输并且在等待回复时设置。在设置 DN 位后清除。
3	ER	Error	当消息传输失败时设置。错误代码写入 ErrorID。ER 位和错误代码值会在下次梯级从 false 变为 true 时清除。
4	DN	Done	当消息已成功传输时设置。DN 位会在下次梯级从 false 变为 true 时清除。 如果已设置 Done 位，则将清除所有其他位以指示 MSG 已成功完成。如果检测到错误并且已设置 Error 位，则不会清除其他状态位 (EN/EW/ST)。

另请参见

[CIPSTATUS 数据类型](#) 参考页数 [188](#)

CIPSTATUS 错误代码

下表描述了在已设置 ER 位时 CIPSTATUS 参数的 ErrorID 字段和 SubErrorID 字段中显示的错误代码。

ErrorID 代码	SubErrorID	错误代码说明
33	与参数配置有关的错误	
	32	通道编号错误。
	36	CIP 连接类型不受支持。
	40	CIP 符号数据类型不受支持。
	41	CIP 符号名称无效。
	43	CIP 类值或 MemberID 计数不受支持。
	48	指令块的输入数据数组大小不够。
	49	目标路径无效。
	50	服务代码错误。
	51	指令块的传输数据数组对于 CIP 通信而言太大。 注意：要传输的用户数据的最大长度因消息配置不同而异。如果整条 CIP 消息的有效负载（包括用户数据和 CIP 消息过载）超过 504 个字节，则将报告错误 0x21 (subError 0x33)。
55	超时相关错误	
	112	在消息等待队列中等待时，消息超时。
	113	在等待建立与链接层的连接时，消息超时。
	114	在等待传输至链接层时，消息超时。
	115	在等待来自链接层的响应时，消息超时。
69	与服务器响应格式有关的错误代码	
	65	消息回复与请求不匹配。
	68	消息回复数据类型无效/不受支持。(MSG_CIPSYMBOLIC)。
208	未为该网络配置 IP 地址。	
209	已达到最大连接数 - 无可用连接。	

ErrorID 代码	SubErrorID	错误代码说明
210		Internet 地址或节点地址无效。
217		用户已取消消息执行。（ Cancel 参数已设为 TRUE ）。
218		无可用网络缓冲空间。
222		已保留。
224		CIP 响应错误代码。SubErrorID 指定 CIP 状态，ExtErrorID 指定 CIP 扩展状态值。有关可能出现的错误代码值，请参考 CIP 规范。
255		通道已关闭或正在进行重新配置。通电后立即出现错误代码，直至建立连接，然后行为正常。 以下情况下会出现： • 以太网无法连接 • 无法检测到 IP 地址 • 已显示串行端口插件，但无法支配。

另请参见

[CIPSTATUS 数据类型](#) 参考页数 [188](#)

CIPTARGETCFG 数据类型

下表描述了 CIPTARGETCFG 数据类型。

参数	数据类型	说明
Path	STRING[80]	目标的路径。可最多指定两个跃点。路径语法为： • {"<端口>,<节点/插槽地址>"2 另请参见 CIP 消息的目标路径 参考页数 192 。
CipConnMode	USINT	CIP 连接类型。 • 0 - 断开（默认） • 1 - Class3 连接 另请参见 CIP/EIP 消息连接 参考页数 193

参数	数据类型	说明
UcmmTimeout	UDINT	断开消息超时 (毫秒)。等待断开消息回复的时长, 包括已连接消息的连接建立。 • 有效值 : 250-10,000。 • 设为 0 以使用默认值 3000。 • 小于 250 的设置值将被设为 250 (最小值)。 • 大于 10,000 的设置值将被设为 10,000 (最大值)。 另请参见 CIP 消息超时计时器 参考页数 194 。
ConnMsgTimeout	UDINT	Class3 连接超时 (毫秒)。等待已连接消息回复的时长。该连接会在超时时间到期时关闭。 • 有效值 : 800-10,000 • 设为 0 以使用默认值 3000 • 小于 800 的设置值将被设为 800 (最小值) • 大于 10,000 的设置值将被设为 10,000 (最大值) 另请参见 CIP 消息超时计时器 参考页数 194 。
ConnClose	BOOL	连接关闭行为 : • TRUE - 当消息完成时关闭连接。 • FALSE - 当消息完成时不关闭连接 (默认)。 • 另请参见 CIP/EIP 消息连接 参考页数 193。

另请参见

[MSG CIPSYMBOLIC](#) 参考页数 [195](#)

[MSG CIPGENERIC](#) 参考页数 [183](#)

CIP 消息的目标路径

包含决定 CIP 消息的途径和目标的配置的 CIP 消息目标路径。

目标路径语法

目标路径字符串参数使用以下语法 :

- "<本地端口>, <第 1 个目标的地址>, [<第 1 个目标的本地端口>, <第 2 个目标的地址>]"

必须存在第 1 个跃点 ; 第 2 个跃点为可选。

字符串元素	说明
本地端口	用于发送消息的本地端口。此端口应为活动的 EtherNet/IP 或 CIP 串行端口 - 不支持 USB 端口。
第 1 个目标地址	第 1 个跃点的目标地址。 - 对于 EIP, 请指定目标的 IP 地址。IP 地址应为单播地址并且不应为 0、多播、广播、本地地址或回送 (127.x.x.x) 地址。 - 对于 CIP 串行, 请指定目标的节点地址。支持的值为 1。
第 1 个目标的本地端口	用于发送消息的本地端口。
第 2 个目标地址	第 2 个跃点的目标地址。

目标路径示例

下表列出了目标路径字符串中使用的值示例，并描述了每个字符串的结果。

字符串示例	结果
"0,0"	目标设备是本地设备。
"6,1"	通过端口 6 (Micro830 UPM 串行端口)，到达位于 1 处的节点。
"4,192.168.1.100"	通过端口 4 (Micro830 嵌入式以太网端口)，到达位于 192.168.1.100 处的节点。
"4,192.168.1.100,1,0"	通过端口 4(Micro830 嵌入式以太网端口) ,到达位于 192.168.0.100 处的节点(Logix ENET 模块)。 从 ENET 模块中，通过底板端口 (端口 1)，到达位于插槽 0 处的 Logix 控制器。

另请参见

[CIPTARGETCFG 数据类型](#) 参考页数 [191](#)

CIP/EIP 消息连接

对于客户端消息执行 ,最多支持 16 个 CIP(3 类)连接和 16 个 EIP 连接。下表描述了 CIP/EIP 连接行为。

场景	结果
消息请求已启用并且 CipConnMode=1。	如果不存在指向目标的连接，则建立 CIP 连接。 如果已存在指向目标的连接，则使用现有 CIP 连接。
消息请求已启用，CipConnMode=1，并且消息的本地端口为以太网。	如果不存在指向目标的 EIP 连接，则在建立 CIP 连接之前先建立 EIP 连接。
消息请求已启用，CipConnMode=0，并且消息的本地端口为以太网。	如果不存在指向目标的 EIP 连接，则建立 EIP 连接。
消息执行已完成，并且 ConnClose 已设为 True。	如果仅存在一个指向目标的连接，则关闭连接。 如果存在多个指向目标的连接，则在完成最后一个消息执行后，关闭连接。 当 CIP 连接关闭后，任何关联的 EIP 连接也会关闭。 如果多个 CIP 连接使用同一 EIP 连接，则该 EIP 连接将在所有关联的 CIP 连接均已关闭后关闭。
消息执行已完成，并且 ConnClose 已设为 False。	连接不关闭。
连接与活动的消息无关，并且在 ConnTimeOut 参数指定的时长内保持空闲。	关闭连接。
控制器从执行模式（运行、远程运行、远程测试单次扫描以及远程单个梯级）过渡到非执行模式。	强制关闭所有活动的连接。

另请参见

[CIPTARGETCFG 数据类型](#) 参考页数 [191](#)

CIP 消息超时计时器

下表描述了 CIPTARGETCFG 超时参数（UcmmTimeout 和 ConnMsgTimeout）的计时器如何根据消息请求和状态开展活动。

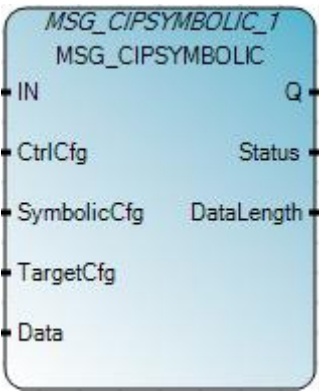
操作	结果
消息已启用	UcmmTimeout 计时器已激活
已请求连接	ConnMsgTimeout 计时器已激活
ConnMsgTimeout 计时器处于活动状态	UcmmTimeout 计时器已禁用
连接请求已完成	UcmmTimeout 计时器已重新激活

另请参见

[CIPTARGETCFG 数据类型](#) 参考页数 [191](#)

MSG_CIPSYMBOLIC

MSG_CIPGENERIC 通过以太网通道或串行端口发送通用工业协议 (CIP) 符号消息。



MSG_CIPSYMBOLIC 操作

启用该功能块后，读取操作的接收缓冲区会在 Enable 的上升沿被清除。
有关示例，请参见[消息执行过程和时间图](#) 参考页数 [225](#)。

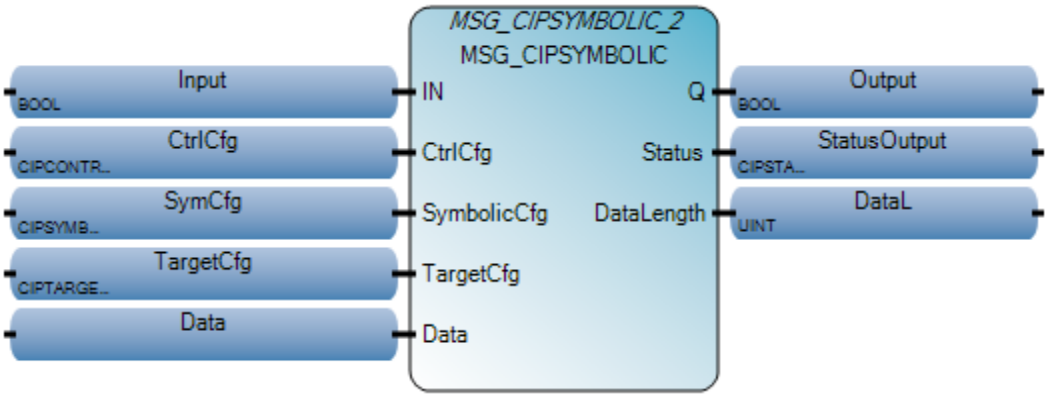
参数

参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿（IN 从 FALSE 变为 TRUE），则启动功能块，前提是上一操作已完成。
CtrlCfg	输入	CIPCONTROLCFG	功能块执行控制配置。 请参见 CIPCONTROLCFG 数据类型 参考页数 187 。
SymbolicCfg	输入	CIPSYMBOLICCFG	读取/写入符号信息。 请参见 CIPSYMBOLICCFG 数据类型 参考页数 198 。
TargetCfg	输入	CIPTARGETCFG	目标设备配置。 请参见 CIPTARGETCFG 数据类型 参考页数 191 。

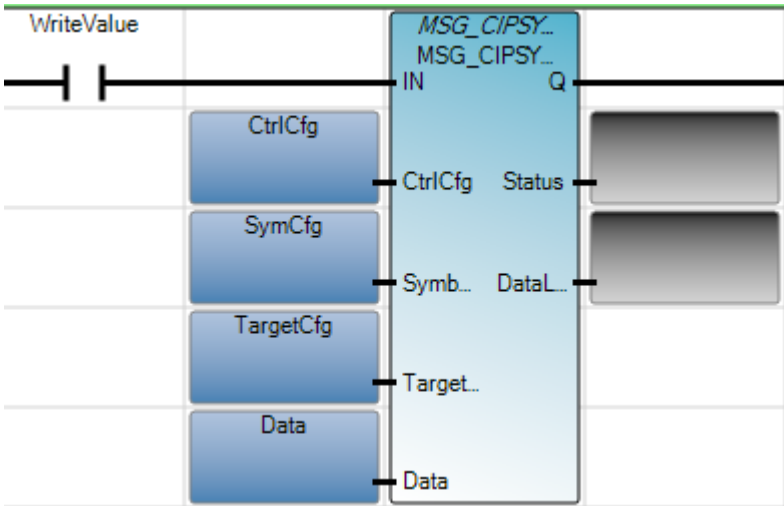
参数	参数类型	数据类型	说明
数据	输入	USINT[490]	读取命令存储从服务器返回的数据。 写入命令缓存要发往服务器的数据。 触发或重新触发 MSG 时，将清除 MSG 读取命令的 Data。
Q	输出	BOOL	FALSE - 功能块未完成。 TRUE - 功能块完成。
状态	输出	CIPSTATUS	功能块执行状态 触发或者重新触发 MSG 时，Status 中的所有元素均会重置。 请参见 CIPSTATUS 数据类型 参考页数 188。
DataLength	输出	UDINT	读取/写入的数据字节数。 触发或重新触发 MSG 时，MSG 读取命令的 DataLength 将重置为 0。

MSG_CIPSYMBOLIC 函数语言示例

功能块图



梯形图



结构化文本 (ST)

```
1 MSG_CIPSYMBOLIC_1(in1, Ctrl1, symbol1, target1, data1);
2 out1 := MSG_CIPGENERIC_1.Q;
3 statuel := MSG_CIPGENERIC_1.Status;
4 reslenq1 := MSG_CIPGENERIC_1.ResLength;
5
6 MSG_CIPSYMBOLIC_2(
    void MSG_CIPSYMBOLIC_2(BOOL IN, CIPCONTROLCFG CtrlCfg, CIPSYMBOLICCFG SymbolicCfg, CIPTARGETCFG TargetCfg, USINT[1..1] Data)
    Type: MSG_CIPSYMBOLIC, Send a CIP Symbolic message.
```

另请参见

- [示例：如何创建 MSG_CIPSYMBOLIC 消息程序以将值写入变量](#) 参考页数 [253](#)
- [消息执行过程和时间图](#) 参考页数 [225](#)
- [通信指令](#) 参考页数 [183](#)
- [符号读取/写入语法](#) 参考页数 [199](#)

CIPSYMBOLICCFG 数据类型

下表描述了 CIPSYMBOLICCFG 数据类型。

参数	数据类型	说明
Service	USINT	服务代码： • 0 - 读取（默认） • 1 - 写入
符号	STRING	要读取/写入的变量的名称。 • 最多 80 个字符。 • 字段不能为空。 请参见符号读取/写入语法 参考页数 199。
Count	UINT	要读取/写入的变量元素个数： • 有效值：1 - 490 • 如果该值设为 0，则使用 1。
类型	用户自定义	目标变量的用户自定义数据类型。 请参见符号数据类型支持 参考页数 198。
Offset	USINT	保留供将来使用。 用于读取/写入一条消息中无法处理的大型变量的读取/写入变量的字节偏移。 • 0 - 0xFF

保留供将来使用。

另请参见

[MSG_CIPSYMBOLIC](#) 参考页数 [195](#)

符号数据类型支持

下表列出了 MSG_CIPSYMBOLIC 功能块支持的数据类型。

数据类型	数据类型值（十六进制）	说明
BOOL	193 (0xC1)	逻辑布尔值可以是 TRUE (1) 和 FALSE (0)
SINT	194 (0xC2)	带符号的 8 位整型值
INT	195 (0xC3)	带符号的 16 位整型值
DINT	196 (0xC4)	带符号的 32 位整型值
LINT	197 (0xC5)	带符号的 64 位整型值
USINT	198 (0xC6)	无符号的 8 位整型值
UINT	199 (0xC7)	无符号的 16 位整型值
UDINT	200 (0xC8)	无符号的 32 位整型值
ULINT	201 (0xC9)	无符号的 64 位整型值
REAL	202 (0xCA)	32 位浮点值
LREAL	203 (0xCB)	64 位浮点值

符号读取/写入语法

语法是定义由有效的读取/写入功能块组成的符号的组合。

有效符号名称

每个符号名称必须符合以下要求才有效。

- 以字母或下划线字符开头，后跟字母、数字或单个下划线字符。
- 字符数不超过 40 个。
- 不含两个连续的下划线字符。
- 使用特殊字符 [] ., 作为分隔符。

符号语法

下表定义了有效的符号语法。

提示：仅支持全局变量。

符号	语法	示例
变量	PROGRAM:<程序名称>,<符号名称>	PROGRAM:POU1.MyTag

CIPAPPCFG 数据类型

参数	数据类型	说明
Service	USINT	服务代码： 1 - 127
Class	UINT	逻辑段类 ID 值： 1 - 65535
实例	UDINT	逻辑段实例 ID 值： 0 - 4294967295
属性	UINT	逻辑段特性 ID 值： 1 - 65535，0 - 未使用特性 ID
MemberCnt	USINT	成员 ID 计数。使用的最大成员 ID 值： 1 - 3，0 - 未使用成员 ID
MemberId	UINT[3]	成员 ID 值： 0 - 65535

MSG_CIPGENERIC 参考页数 183

CIPCONTROLCFG 数据类型

Rockwell Automation 出版号 2080-RM001D-ZH-E - 2015 年 2 月

参数	数据类型	说明
取消	BOOL	TRUE - 取消功能块的执行。 启用消息后将清除位。
TriggerType	USINT	表示以下之一： 0：触发一次 Msg（当 IN 从 False 转为 True 时） 1 到 65535：循环触发值（毫秒）。当 IN 为 True 时，定期触发 Msg。将该值设为 1 以尽快触发 Msg。
StrMode	USINT	保留供将来使用。

消息取消

如果 Cancel 参数已设置，消息已启用（已设置 EN 位）并且未完成（未设置 DN 位），则消息执行会中止并设置 ER 位。

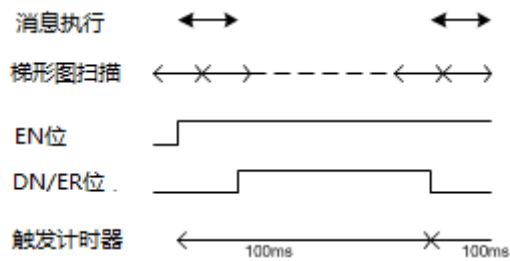
CIP 消息触发

通过将 TriggerType 参数设置为非零值，可以定期触发 CIP 消息。下表描述了在使用 TriggerType 参数时发生的情况。

操作	结果
消息已启用	触发器计时器启动
触发器计时器在消息完成前到期	消息在下一个梯形扫描周期中立即触发。
消息在触发器时间到期前完成	消息在触发器时间到期时触发。

示例：消息触发

在下例中，TriggerType 值设为 100。



另请参见

[MSG_CIPSYMBOLIC](#) 参考页数 [195](#)

[MSG_CIPGENERIC](#) 参考页数 [183](#)

CIPSTATUS 数据类型

下表描述了 CIPSTATUS 数据类型。

参数	数据类型	说明
Error	BOOL	如果功能块执行遇到错误条件，该位设为 TRUE。
ErrorID	UINT	错误代码值。 请参见 CIPSTATUS 错误代码 参考页数 190 。
SubErrorID	UINT	子错误代码值。 请参见 CIPSTATUS 错误代码 参考页数 190 。
ExtErrorID	UINT	CIP 扩展状态错误代码值。
StatusBits	UINT	此参数可用于验证控制位： • 位 0：EN - Enable • 位 1：EW - Enable Wait • 位 2：ST - Start • 位 3：ER - Error • 位 4：DN - Done • 其他位保留 • 请参见 CIPSTATUS 状态位 参考页数 189。

另请参见

[MSG_CIPSYMBOLIC](#) 参考页数 [195](#)

[MSG_CIPGENERIC](#) 参考页数 [183](#)

CIPSTATUS 状态位

CIPSTATUS 状态位的设置基于消息执行状态、通信缓冲以及梯级条件。

-	-	-	-	-	-	-	-	-	-	-	4	3	2	1	0

位	名称	说明	行为
0	EN	启用	当梯级变为 true 时设置，并且将一直设为该值，直至 DN 位或 ER 位已设置且梯级变为 false。
1	EW	Enable Wait	当为消息请求分配通信缓冲时设置。在设置 ST 位后清除。
2	ST	启动	当消息已传输并且在等待回复时设置。在设置 DN 位后清除。
3	ER	Error	当消息传输失败时设置。错误代码写入 ErrorID。ER 位和错误代码值会在下次梯级从 false 变为 tru 时清除。
4	DN	Done	当消息已成功传输时设置。DN 位会在下次梯级从 false 变为 true 时清除。 如果已设置 Done 位，则将清除所有其他位以指示 MSG 已成功完成。如果检测到错误并且已设置 Error 位，则不会清除其他状态位 (EN/EW/ST)。

另请参见

[CIPSTATUS 数据类型](#) 参考页数 [188](#)

CIPSTATUS 错误代码

下表描述了在已设置 ER 位时 CIPSTATUS 参数的 ErrorID 字段和 SubErrorID 字段中显示的错误代码。

ErrorID 代码	SubErrorID	错误代码说明
33		与参数配置有关的错误
	32	通道编号错误。
	36	CIP 连接类型不受支持。
	40	CIP 符号数据类型不受支持。
	41	CIP 符号名称无效。
	43	CIP 类值或 MemberID 计数不受支持。
	48	指令块的输入数据数组大小不够。
	49	目标路径无效。
	50	服务代码错误。
	51	指令块的传输数据数组对于 CIP 通信而言太大。 注意：要传输的用户数据的最大长度因消息配置不同而异。如果整条 CIP 消息的有效负载（包括用户数据和 CIP 消息过载）超过 504 个字节，则将报告错误 0x21 (subError 0x33)。

ErrorID 代码	SubErrorID	错误代码说明
55	超时相关错误	
	112	在消息等待队列中等待时，消息超时。
	113	在等待建立与链接层的连接时，消息超时。
	114	在等待传输至链接层时，消息超时。
	115	在等待来自链接层的响应时，消息超时。
69	与服务器响应格式有关的错误代码	
	65	消息回复与请求不匹配。
	68	消息回复数据类型无效/不受支持。(MSG_CIPSYMBOLIC)。
208	未为该网络配置 IP 地址。	
209	已达到最大连接数 - 无可用连接。	
210	Internet 地址或节点地址无效。	
217	用户已取消消息执行。(Cancel 参数已设为 TRUE)。	
218	无可用网络缓冲空间。	
222	已保留。	
224	CIP 响应错误代码。SubErrorID 指定 CIP 状态，ExtErrorID 指定 CIP 扩展状态值。有关可能出现的错误代码值，请参考 CIP 规范。	
255	通道已关闭或正在进行重新配置。通电后立即出现错误代码，直至建立连接，然后行为正常。 以下情况下会出现： - 以太网无法连接 - 无法检测到 IP 地址 - 已显示串行端口插件，但无法支配。	

另请参见

[CIPSTATUS 数据类型](#) 参考页数 [188](#)

CIPTARGETCFG 数据类型

下表描述了 CIPTARGETCFG 数据类型。

参数	数据类型	说明
Path	STRING[80]	目标的路径。可最多指定两个跃点。路径语法为： • {"<端口>,<节点/插槽地址>"}2 另请参见 CIP 消息的目标路径 参考页数 192 。
CipConnMode	USINT	CIP 连接类型。 • 0 - 断开（默认） • 1 - Class3 连接 另请参见 CIP/EIP 消息连接 参考页数 193
UcmmTimeout	UDINT	断开消息超时（毫秒）。等待断开消息回复的时长，包括已连接消息的连接建立。 • 有效值：250-10,000。 • 设为 0 以使用默认值 3000。 • 小于 250 的设置值将被设为 250（最小值）。 • 大于 10,000 的设置值将被设为 10,000（最大值）。 另请参见 CIP 消息超时计时器 参考页数 194 。
ConnMsgTimeout	UDINT	Class3 连接超时（毫秒）。等待已连接消息回复的时长。该连接会在超时时间到期时关闭。 • 有效值：800-10,000 • 设为 0 以使用默认值 3000 • 小于 800 的设置值将被设为 800（最小值） • 大于 10,000 的设置值将被设为 10,000（最大值） 另请参见 CIP 消息超时计时器 参考页数 194 。
ConnClose	BOOL	连接关闭行为： • TRUE - 当消息完成时关闭连接。 • FALSE - 当消息完成时不关闭连接（默认）。 • 另请参见 CIP/EIP 消息连接 参考页数 193。

另请参见

[MSG CIPSYMBOLIC](#) 参考页数 [195](#)

[MSG CIPGENERIC](#) 参考页数 [183](#)

CIP 消息的目标路径

包含决定 CIP 消息的途径和目标的配置的 CIP 消息目标路径。

目标路径语法

目标路径字符串参数使用以下语法：

- "<本地端口>,<第 1 个目标的地址>,[<第 1 个目标的本地端口>,<第 2 个目标的地址>]"

必须存在第 1 个跃点；第 2 个跃点为可选。

字符串元素	说明
本地端口	用于发送消息的本地端口。此端口应为活动的 EtherNet/IP 或 CIP 串行端口 - 不支持 USB 端口。
第 1 个目标地址	第 1 个跃点的目标地址。 - 对于 EIP，请指定目标的 IP 地址。IP 地址应为单播地址并且不应为 0、多播、广播、本地地址或回送 (127.x.x.x) 地址。 - 对于 CIP 串行，请指定目标的节点地址。支持的值为 1。
第 1 个目标的本地端口	用于发送消息的本地端口。
第 2 个目标地址	第 2 个跃点的目标地址。

目标路径示例

下表列出了目标路径字符串中使用的值示例，并描述了每个字符串的结果。

字符串示例	结果
"0,0"	目标设备是本地设备。
"6,1"	通过端口 6 (Micro830 UPM 串行端口) , 到达位于 1 处的节点。
"4,192.168.1.100"	通过端口 4 (Micro830 嵌入式以太网端口) , 到达位于 192.168.1.100 处的节点。
"4,192.168.1.100,1,0"	通过端口 4(Micro830 嵌入式以太网端口) ,到达位于 192.168.0.100 处的节点(Logix ENET 模块) 。 从 ENET 模块中, 通过底板端口 (端口 1) , 到达位于插槽 0 处的 Logix 控制器。

另请参见

[CIPTARGETCFG 数据类型](#) 参考页数 [191](#)

CIP/EIP 消息连接

对于客户端消息执行 ,最多支持 16 个 CIP(3 类)连接和 16 个 EIP 连接。下表描述了 CIP/EIP 连接行为。

场景	结果
消息请求已启用并且 CipConnMode=1。	如果不存在指向目标的连接，则建立 CIP 连接。 如果已存在指向目标的连接，则使用现有 CIP 连接。
消息请求已启用，CipConnMode=1，并且消息的本地端口为以太网。	如果不存在指向目标的 EIP 连接，则在建立 CIP 连接之前先建立 EIP 连接。
消息请求已启用，CipConnMode=0，并且消息的本地端口为以太网。	如果不存在指向目标的 EIP 连接，则建立 EIP 连接。
消息执行已完成，并且 ConnClose 已设为 True。	如果仅存在一个指向目标的连接，则关闭连接。 如果存在多个指向目标的连接，则在完成最后一个消息执行后，关闭连接。 当 CIP 连接关闭后，任何关联的 EIP 连接也会关闭。 如果多个 CIP 连接使用同一 EIP 连接，则该 EIP 连接将在所有关联的 CIP 连接均已关闭后关闭。
消息执行已完成，并且 ConnClose 已设为 False。	连接不关闭。
连接与活动的消息无关，并且在 ConnTimeOut 参数指定的时长内保持空闲。	关闭连接。
控制器从执行模式（运行、远程运行、远程测试单次扫描以及远程单个梯级）过渡到非执行模式。	强制关闭所有活动的连接。

另请参见

[CIPTARGETCFG 数据类型](#) 参考页数 [191](#)

CIP 消息超时计时器

下表描述了 CIPTARGETCFG 超时参数（UcmmTimeout 和 ConnMsgTimeout）的计时器如何根据消息请求和状态开展活动。

操作	结果
消息已启用	UcmmTimeout 计时器已激活
已请求连接	ConnMsgTimeout 计时器已激活
ConnMsgTimeout 计时器处于活动状态	UcmmTimeout 计时器已禁用
连接请求已完成	UcmmTimeout 计时器已重新激活

另请参见

[CIPTARGETCFG 数据类型](#) 参考页数 [191](#)

已支持的 CIP 串行功能数据包大小

对于 Micro830 和 Micro850 控制器，嵌入式串行端口和插件串行端口均能支持 CIP 串行通信。CIP 串行通信数据包包含用户数据和 CIP 数据包头。

用作 CIP 串行客户端时，Micro830/Micro850 串行端口最大可支持 490 字节的读入/写入用户数据。最大规范适用于数据包头最小的 CIP 串行数据包。若该数据包头大于最小数据包头，则 CIP 客户端可支持的最大用户数据不足 490 字节。若该数据包大于 CIP 客户端支持的最大数据，则功能块会报告错误 (0x21) 和子错误 (0x33)。

用作 CIP 串行服务器时，Micro830/Micro850 串行端口最小可支持 255 字节的读入/写入用户数据。此类最小用户数据规范适用于数据包头最大的 CIP 串行数据包。当 CIP 数据包头小于最大数据包头时，则 CIP 客户端可支持大于最小规范（即大于 255 字节）的数据包。但是，若用户数据大于 CIP 服务器功能支持的最大数据，则 CIP 数据包可能会被放弃且客户端将会超时。

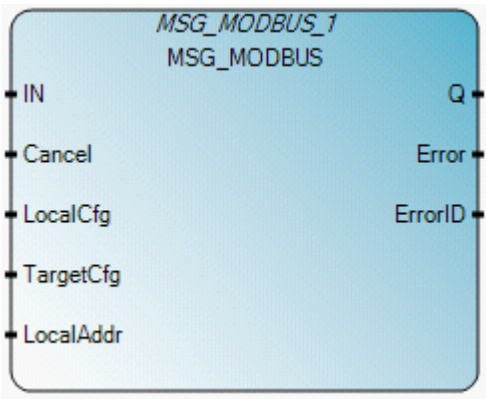
重要说明： 对于 CIP 串行服务器功能，建议不要在单个 CIP 消息中读入/写入大于 255 字节的用户数据。

另请参见

[MSG_CIPSYMBOLIC](#) 参考页数 [195](#)

MSG_MODBUS

MSG_MODBUS 通过串行端口发送 Modbus 消息。



MSG_MODBUS 操作

每个通道在一次扫描中最多可以处理四个消息请求。对于梯形图程序，将在梯形扫描结束时执行消息请求。

参数

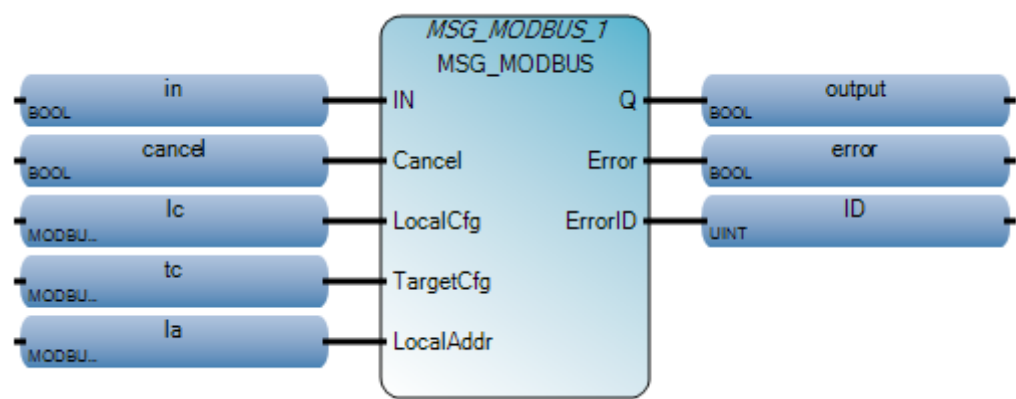
参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿（IN 从 FALSE 变为 TRUE），则启动功能块，前提是上一操作已完成。
取消	输入	BOOL	TRUE - 取消功能块的执行。
LocalCfg	输入	MODBUSLOCPARA	定义结构输入（本地设备）。 定义本地设备的输入结构。 请参见 MODBUSLOCPARA 数据类型 参考页数 212 。
TargetCfg	输入	MODBUSTARPARA	定义结构输入（目标设备）。 定义目标设备的输入结构。 请参见 MODBUSTARPARA 数据类型 参考页数 217 。
LocalAddr	输入	MODBUSLOCADDR	MODBUSLOCADDR 是一个大小为 125 个字的数组，由读取命令用来存储 Modbus 从站返回的数据（1-125 个字），并由写入命令用来缓存要发送到 Modbus 从站的数据（1-125 个字）。

参数	参数类型	数据类型	说明
Q	输出	BOOL	TRUE - MSG 指令已完成。 FALSE - MSG 指令未完成。
Error	输出	BOOL	TRUE - 发生错误时。 FALSE - 没有错误。
ErrorID	输出	UINT	当消息传输失败时显示错误代码。 请参见 Modbus 错误代码 参考页数 211 。

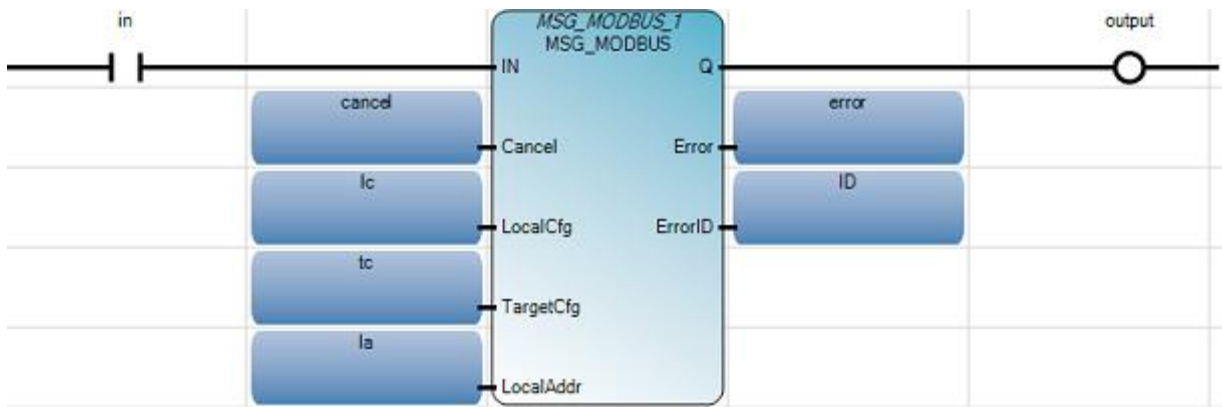
提示 当将触发器设置为继续时，错误代码也将继续清除。要查看错误代码，请在
： MSG_MODBUS 指令前添加一个梯级。

MSG_MODBUS 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 MSG_MODBUS_1(in, cancel, lc, tc, la);
2 output := MSG_MODBUS_1.Q;
3 error := MSG_MODBUS_1.Error;
4 ID := MSG_MODBUS_1.ErrorID;
```

```
MSG_MODBUS_1 {
void MSG_MODBUS_1(BOOL IN, BOOL Cancel, MODBUSLOCpara LocalCfg, MODBUSTARpara TargetCfg, MODBUSLOCADDR LocalAddr, UINT __ADI_LocalAddr)
Type : MSG_MODBUS, Send a modbus message.
```

另请参见

- [示例：如果配置 Modbus 通信以读取和写入驱动器](#) 参考页数 268
- [消息执行过程和时间图](#) 参考页数 225
- [通信指令](#) 参考页数 183

Modbus 错误代码

下表描述了 MSG_MODBUS 功能块的错误代码。

错误代码	错误描述
3	TriggerType 的值从 2 - 255 发生变更。
20	本地通信驱动程序与 MSG 指令不兼容。

错误代码	错误描述
21	存在本地通道配置参数错误。
22	Target 或 Local Bridge 地址高于最大节点地址。
33	存在错误的 MSG 文件参数。
54	缺少调制解调器。
55	消息在本地处理器中超时。链接层超时。
217	用户取消了消息。
129	非法函数。
130	非法数据地址。
131	非法数据值。
132	从属设备错误。
133	确认。
134	从属设备忙碌。
135	反确认。
136	内存奇偶校验错误。
137	非标准回复。
255	通道已关闭。

另请参见

[MSG MODBUS](#) 参考页数 [209](#)

MODBUSLOCPARA 数据类型

下表描述了 MODBUSLOCPARA 数据类型参数。

参数	数据类型	说明
通道	UINT	Micro800 PLC 串行端口号： • 嵌入式串行端口为 2，或 • 安装在插槽 1 到插槽 5 中的串行端口插件为 5-9 • 插槽 1 为 5 • 插槽 2 为 6 • 插槽 3 为 7 • 插槽 4 为 8 • 插槽 5 为 9
TriggerType	USINT	表示以下之一： • 0：触发一次 Msg（当 IN 从 False 转为 True 时） • 1：当 IN 为 True 时，连续触发 Msg • 其他值：保留
Cmd	USINT	表示以下之一： • 01：读取线圈状态 (0xxxx) • 02：读取输入状态 (1xxxx) • 03：读取保持寄存器 (4xxxx) • 04：读取输入寄存器 (3xxxx) • 05：写入单个线圈 (0xxxx) • 06：写入单个寄存器 (4xxxx) • 15：写入多个线圈 (0xxxx) • 16：写入多个寄存器 (4xxxx) • 其他：请参见 MODBUSLOCPARA 自定义命令支持。
ElementCnt	UINT	限制 • 对于读取线圈/离散输入：2000 位 • 对于读取寄存器：125 个字 • 对于写入线圈：1968 位 • 对于写入寄存器：123 个字

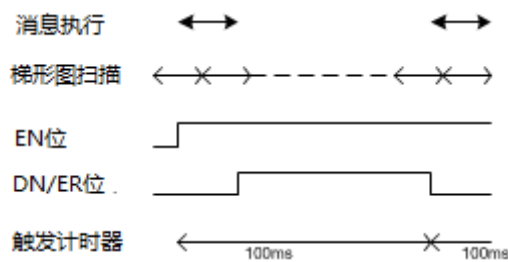
MSG_MODBUS 消息触发

通过将 TriggerType 参数设置为非零值，可以定期触发 Modbus 消息。下表描述了在将 TriggerType 参数与 MSG_MODBUS 功能块结合使用时发生的情况。

操作	结果
消息已启用	触发器计时器启动
触发器计时器在消息完成前到期	消息在下一个梯形扫描周期中立即触发。
消息在触发器时间到期前完成	消息在触发器时间到期时触发。

示例：消息触发

在下例中，TriggerType 值设为 100。



MODBUSLOCPARA 自定义命令支持

在尚未分配给 Modbus 命令的 0-255 范围中的自定义命令也受支持。如果使用了自定义命令，则 LocalCfg:ElementCnt 包含接收到的字节数。

响应将接收到 Local Address Data 并覆盖请求数据。

CMD=0x2B 示例

- Local Address Data 1:0x0E, READ_DEVICE_ID_MEI
- Local Address Data 2:0x01, READ_DEV_ID_BASIC
- Local Address Data 3:0x00, Read Vendor Object

另请参见

[MSG MODBUS](#) 参考页数 [209](#)

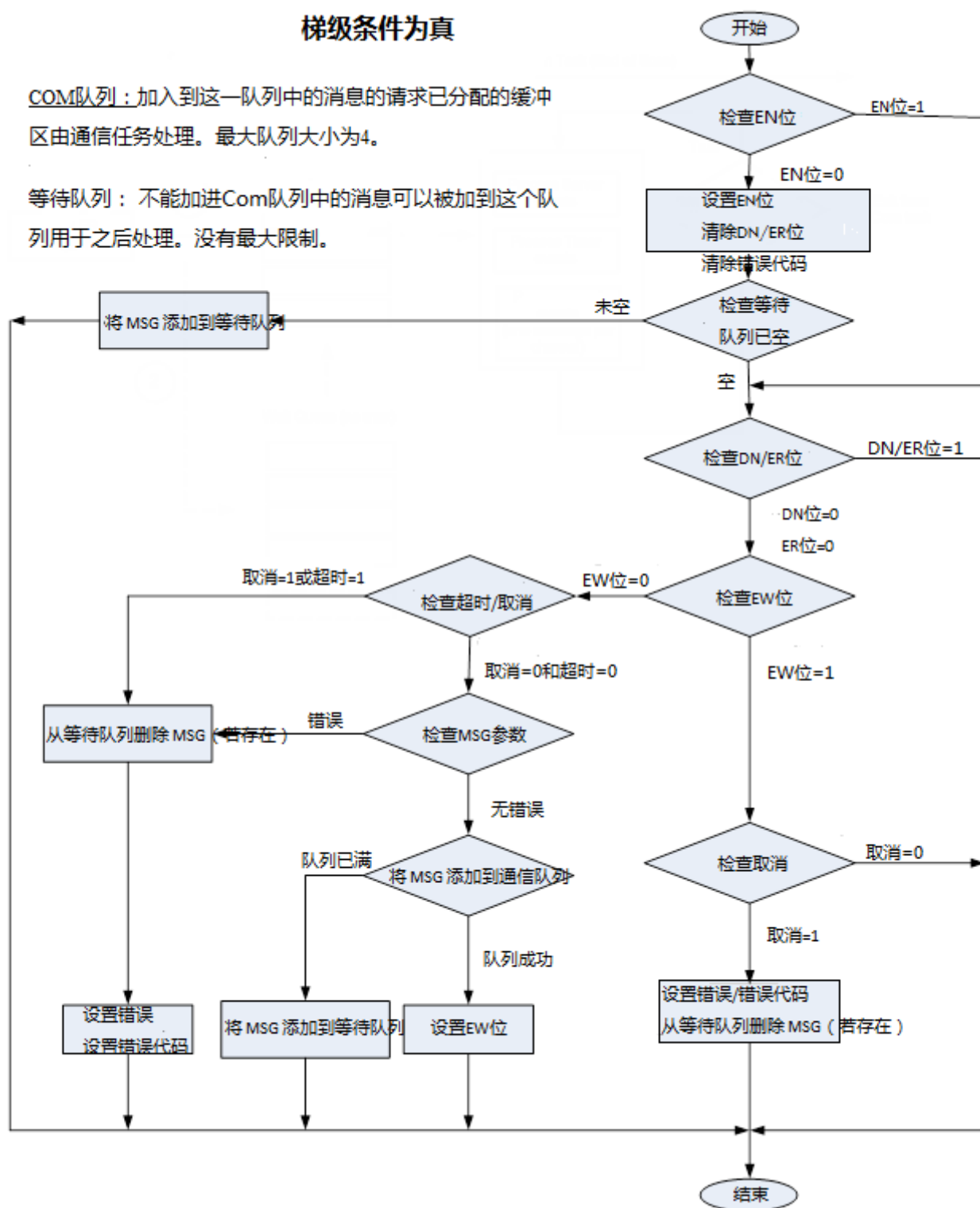
消息执行过程 (Rung = TRUE)

以下过程图描述了当 Rung 条件为 True 时发生的消息指令事件。

梯级条件为真

COM队列：加入到这一队列中的消息的请求已分配的缓冲区由通信任务处理。最大队列大小为4。

等待队列：不能加进Com队列中的消息可以被加到这个队列用于之后处理。没有最大限制。



通信队列：对于已添加到通信队列中的消息请求，它们有一个由通信任务分配和处理的缓冲区。最大队列大小限制为 4。

等待队列：对于无法添加到通信队列的消息，它们会被添加到将在稍后处理的等待队列。等待队列没有最大大小限制。

另请参见

[消息执行过程和时间图](#) 参考页数 [225](#)

[通信指令](#) 参考页数 [183](#)

[MSG MODBUS](#) 参考页数 [209](#)

MODBUSTARPARA 数据类型

下表描述了 MODBUSTARPARA 数据类型。

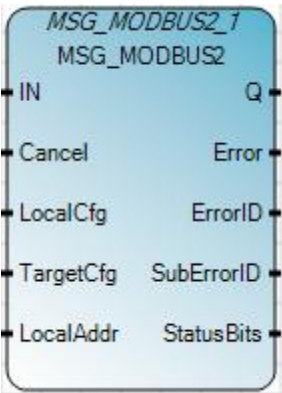
参数	数据类型	说明
Addr	UDINT	目标数据地址 (1 - 65536)。发送时减 1。
Node	USINT	默认从属节点地址为 1。范围为 1 - 247。“0”是 Modbus 广播地址，并且仅对 Modbus 写入命令（例如 5、6、15 和 16）有效。

另请参见

[MSG MODBUS](#) 参考页数 [209](#)

MSG_MODBUS2

MSG_MODBUS2 通过以太网通道发送 MODBUS/TCP 消息。



MSG_MODBUS2 操作

每个通道在一次扫描中最多可以处理四个消息请求。对于梯形图程序，将在梯形扫描结束时执行消息请求。

参数

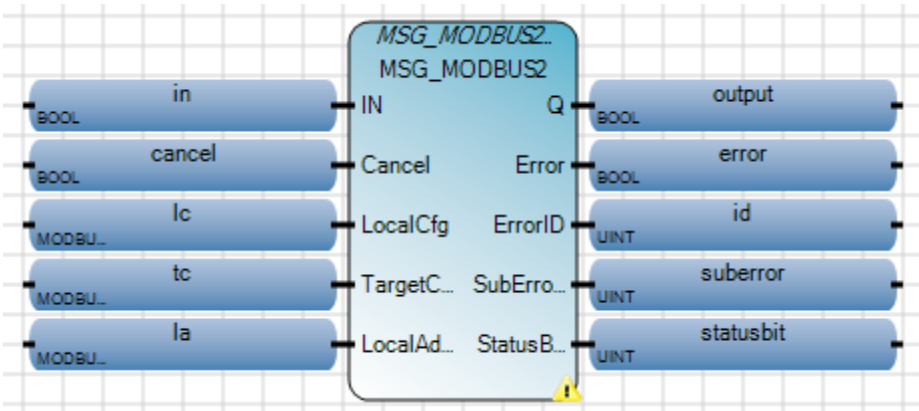
启用 MSG_MODBUS2 功能块后，读取操作的接收缓冲区会在 Enable 的上升沿被清除。

参数	参数类型	数据类型	说明
IN	输入	BOOL	如果为上升沿（IN 从 FALSE 变为 TRUE），则启动功能块，前提是上一操作已完成。
取消	输入	BOOL	TRUE - 取消功能块的执行。
LocalCfg	输入	MODBUS2LOCPARA	定义结构输入（本地设备）。 定义本地设备的输入结构。 请参见 MODBUS2LOCPARA 数据类型 参考页数 221 。
TargetCfg	输入	MODBUS2TARPAR	定义结构输入（目标设备）。 定义目标设备的输入结构。 请参见 MODBUS2TARPAR 数据类型 参考页数 223 。
LocalAddr	输入	MODBUSLOCADDR	MODBUSLOCADDR 数据类型是一个大小为 125 个字的数组，由读取命令用来存储 Modbus 从站返回的数据（1-125 个字），并由写入命令用来缓冲要发送到 Modbus 从站的数据（1-125 个字）。

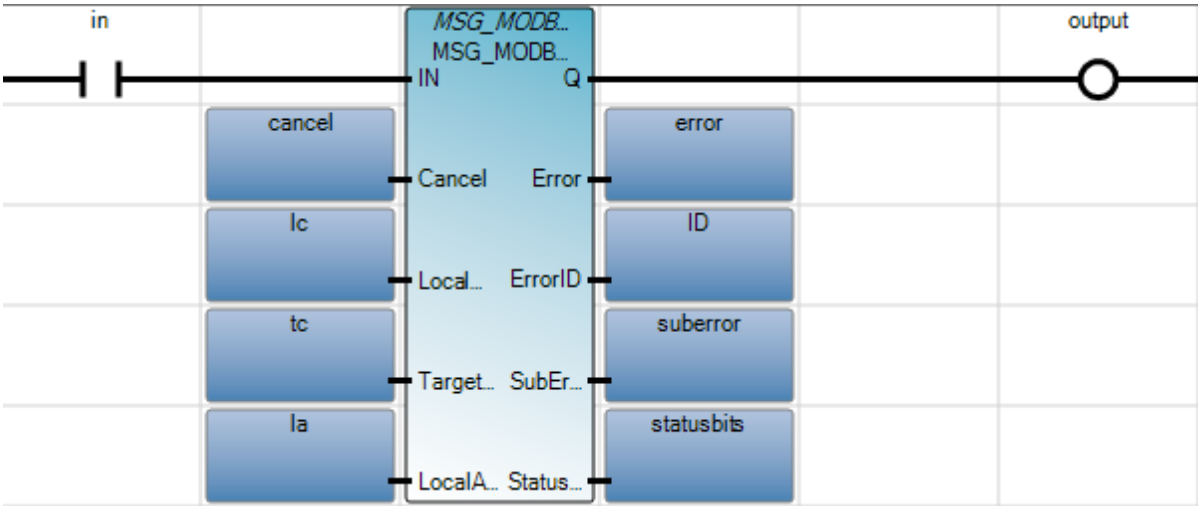
参数	参数类型	数据类型	说明
Q	输出	BOOL	TRUE - MSG 指令已完成。 FALSE - MSG 指令未完成。
Error	输出	BOOL	TRUE - 发生错误时。 FALSE - 没有错误。
ErrorID	输出	UINT	当消息传输失败时显示错误代码。 请参见 Modbus2 错误代码 参考页数 220 。
SuberrorID	输出	UINT	用于确认状态位： • 位 0：EN - Enable • 位 1：EW - Enable Wait • 位 2：ST - Start • 位 3：ER - Error • 位 4：DN - Done 其他位保留。
StatusBits	输出	UINT	当 Error 为 TRUE 时的子错误代码值。 触发或重新触发 MSG 时，将清除先前设置的 SubErrorID。

MSG_MODBUS2 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 MSG_MODBUS2_1(in, cancel, lc, tc, la);
2 output := MSG_MODBUS2_1.Q;
3 error := MSG_MODBUS2_1.Error;
4 ID := MSG_MODBUS2_1.ErrorID;
```

MSG_MODBUS2_1

void MSG_MODBUS2_1(BOOL IN, BOOL Cancel, MODBUS2LOCpara LocalCfg, MODBUS2TARpara TargetCfg, MODBUSLOCADDR LocalAddr)
Type : MSG_MODBUS2, Send a modbus message.

另请参见

[消息执行过程和时间图](#) 参考页数 [225](#)

[通信指令](#) 参考页数 [183](#)

Modbus2 错误代码

下表描述了 MSG_MODBUS2 功能块的错误代码。

错误代码	错误描述
3	TriggerType 的值从 2 - 255 发生变更。

错误代码	错误描述
20	本地通信驱动程序与 MSG 指令不兼容。
21	存在本地通道配置参数错误。
22	Target 或 Local Bridge 地址高于最大节点地址。
33	存在错误的 MSG 文件参数。
54	缺少调制解调器。
55	消息在本地处理器中超时。链接层超时。
217	用户取消了消息。
129	非法函数。
130	非法数据地址。
131	非法数据值。
132	从属设备错误。
133	确认。
134	从属设备忙碌。
135	反确认。
136	内存奇偶校验错误。
137	非标准回复。
255	通道已关闭。

另请参见

[MSG MODBUS2](#) 参考页数 [217](#)

MODBUS2LOCPARA 数据类型

下表描述了 MODBUSLOCPARA 数据类型参数。

参数	数据类型	说明
通道	UINT	本地以太网端口号： • 4 (对于 Micro850 和 Micro820 嵌入式以太网端口)

参数	数据类型	说明
TriggerType	UDINT	消息触发器类型： • 0：触发一次 Msg（当 IN 从 False 转为 True 时） • 1 到 65535 - 循环触发器值（毫秒）。当 IN 为 True 并且先前消息执行完成时，定期触发消息。 • 将该值设为 1 以尽快触发消息。
Cmd	USINT	Modbus 命令： • 01：读取线圈状态 (0xxxx) • 02：读取输入状态 (1xxxx) • 03：读取保持寄存器 (4xxxx) • 04：读取输入寄存器 (3xxxx) • 05：写入单个线圈 (0xxxx) • 06：写入单个寄存器 (4xxxx) • 15：写入多个线圈 (0xxxx) • 16：写入多个寄存器 (4xxxx) • 其他：请参见 MODBUS2LOCPARA 自定义命令支持
ElementCnt	UINT	限制 • 对于读取线圈/离散输入：2000 位 • 对于读取寄存器：125 个字 • 对于写入线圈：1968 位 • 对于写入寄存器：123 个字

MSG_MODBUS2 消息触发

通过将 TriggerType 参数设置为非零值，可以定期触发 Modbus 消息。下表描述了在将 TriggerType 参数与 MSG_MODBUS2 功能块结合使用时发生的情况。

操作	结果
消息已启用	触发器计时器启动
触发器计时器在消息完成前到期	消息在下一个梯形扫描周期中立即触发。
消息在触发器时间到期前完成	消息在触发器时间到期时触发。

MODBUS2LOCPARA 自定义命令支持

在尚未分配给 Modbus 命令的 0-255 范围中的自定义命令也受支持。
如果使用了自定义命令，则 LocalCfg:ElementCnt 包含接收到的字节数。

响应将接收到 Local Address Data 并覆盖请求数据。

CMD=0x2B 示例

- Local Address Data 1:0x0E, READ_DEVICE_ID_MEI
- Local Address Data 2:0x01, READ_DEV_ID_BASIC
- Local Address Data 3:0x00, Read Vendor Object

另请参见

[MSG MODBUS2](#) 参考页数 [217](#)

MODBUS2TARPARA 数据类型

下表描述了 MODBUSTARPARA 数据类型参数。

参数	数据类型	说明
Addr	UDINT	目标设备的 Modbus 数据地址： 1 - 65536。 - 发送时减 1。 - 如果地址值大于 65536，则固件使用地址的低字。
NodeAddress[4]	USINT	目标设备的 IP 地址。IP 地址应为有效的单播地址并且不应为 0、多播、广播、本地地址或回送地址 (127.x.x.x)。 例如，指定 192.168.2.100： - NodeAddress[0]=192 - NodeAddress[1]=168 - NodeAddress[2]=2 - NodeAddress[3]=100

参数	数据类型	说明
端口	UINT	目标 TCP 端口号。标准 Modbus/TCP 端口为 502。 1 - 65535 设为 0 以使用默认值 502
UnitId	USINT	单位标识符。用于通过 Modbus 桥与从属设备通信。有关更多详细信息，请参考 Modbus 规范。请注意，Micro800 不应尝试验证该值。 0 - 255 如果目标设备不是桥，请设为 255。
MsgTimeOut	UDINT	消息超时（毫秒）。等待已启动命令回复的时长。 • 250-10,000 • 设为 0 以使用默认值 3000。 • 小于 250（最小值）的值将被设为 250。 • 大于 10,000（最小值）的值将被设为 10,000。 另请参见 Modbus/TCP 消息超时计时器 参考页数 224 。
ConnTimeOut	UDINT	TCP 连接建立超时（毫秒）。等待与目标设备成功建立 TCP 连接的时长。 • 250-10,000 • 设为 0 以使用默认值 3000。 • 小于 250（最小值）的值将被设为 250。 • 大于 10,000（最小值）的值将被设为 10,000。 另请参见 Modbus/TCP 消息超时计时器 参考页数 224 。
ConnClose	BOOL	TCP 连接关闭行为。 • True：在消息完成时关闭 TCP 连接。 • False：在消息完成时不关闭 TCP 连接 [默认]。 另请参见 Modbus/TCP 消息连接 参考页数 225 。

另请参见

[MSG MODBUS2](#) 参考页数 [217](#)

Modbus/TCP 消息超时计时器

下表描述了 MODBUS2TARPARA 超时参数（UcmmTimeout 和 ConnMsgTimeout）的计时器如何根据消息请求和状态开展活动。

操作	结果
消息已启用	UcmmTimeout 计时器已激活
已请求连接	ConnMsgTimeout 计时器已激活

操作	结果
ConnMsgTimeout 计时器处于活动状态	UcmmTimeout 计时器已禁用
连接请求已完成	UcmmTimeout 计时器已重新激活

另请参见

[MODBUS2TARPARA 数据类型](#) 参考页数 [223](#)

Modbus/TCP 消息连接

Modbus/TCP 客户端最多支持 16 个连接。下表描述了 Modbus/TCP 连接行为。

场景	结果
消息请求已启用，并且不存在指向目标的连接。	如果不存在指向目标的连接，则建立新连接。 如果已存在指向目标的连接，则使用现有连接。
消息执行已完成，并且 ConnClose 已设为 True。 。	如果仅存在一个指向目标的连接，则关闭连接。 如果存在多个指向目标的连接，则在完成最后一个消息执行后，关闭连接。
消息执行已完成，并且 ConnClose 已设为 False。 。	连接不关闭。
连接与活动的消息无关，并且在 ConnTimeOut 参数指定的时长内保持空闲。	关闭连接。
控制器从执行模式（运行、远程运行、远程测试单次扫描以及远程单个梯级）过渡到非执行模式。	强制关闭所有活动的连接。

另请参见

[MODBUS2TARPARA 数据类型](#) 参考页数 [223](#)

消息执行过程和时间图

以下主题描述了 MSG_CIPGENERIC、MSG_CIPSYMBOLIC 和 MSG_MODBUS2 消息指令何时及如何根据它们的位和梯级条件执行。

[消息执行过程（常规）](#) 参考页数 [226](#)

[消息执行次序（常规）](#) 参考页数 [227](#)

[消息执行过程 \(Rung = TRUE\)](#) 参考页数 [215](#)

[消息执行时间图 \(Rung = True\)](#) 参考页数 [230](#)

[消息执行过程 \(Rung = FALSE\)](#) 参考页数 [231](#)

[消息执行时间图 \(Rung = FALSE\)](#) 参考页数 [233](#)

[消息执行过程（错误）](#) 参考页数 [234](#)

[消息执行时间图（错误）](#) 参考页数 [235](#)

另请参见

[通信指令](#) 参考页数 [183](#)

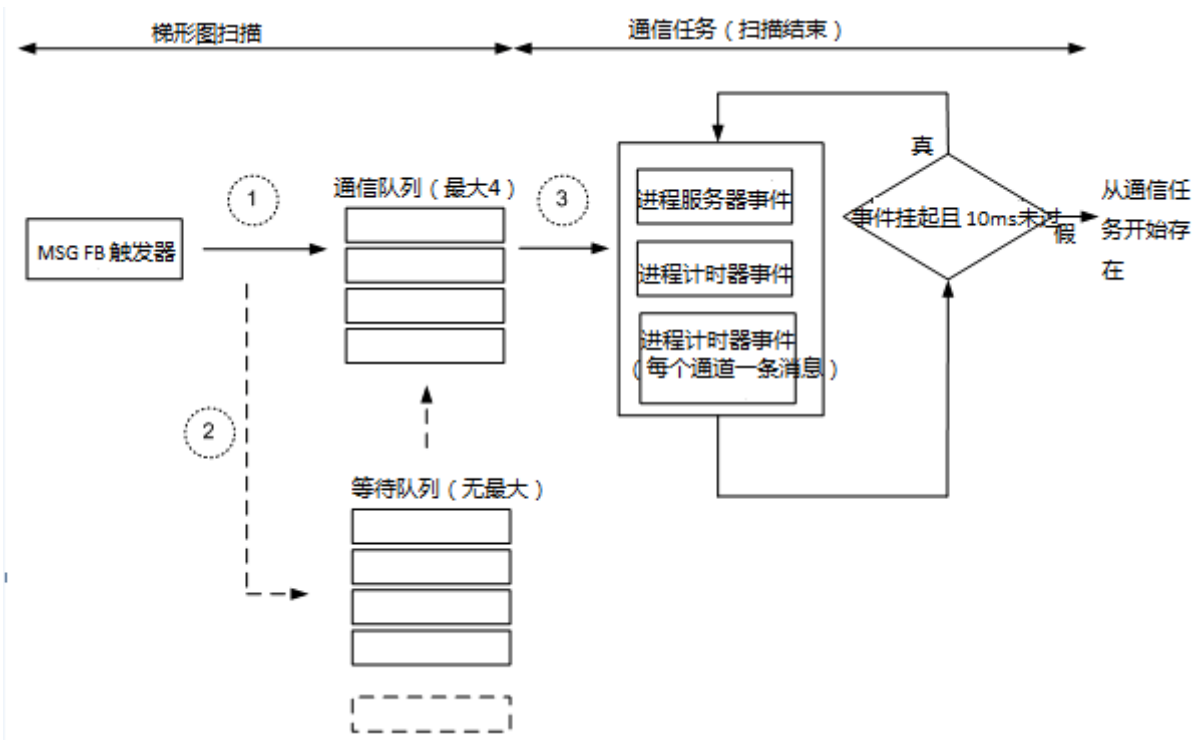
[MSG MODBUS2](#) 参考页数 [217](#)

[MSG CIPSYMBOLIC](#) 参考页数 [195](#)

[MSG CIPGENERIC](#) 参考页数 [183](#)

消息执行过程（常规）

下图展示了消息何时及如何根据通信队列状态执行。有关每个队列的详细说明，请参见下表。



另请参见

[消息执行过程和时间图](#) 参考页数 225

[通信指令](#) 参考页数 183

消息执行次序（常规）

下表描述了上一图中标识的事件序列。

编号	事件说明
1	消息已启用。 如果通信队列为空，则为消息分配缓冲区，并且消息会添加到通信队列以进行传输。 通信队列大小为 4，并且每个通道均具有单独的队列。

编号	事件说明
2	如果通信队列已满，则消息会添加到等待队列。 当通信队列为空时，等待队列中的消息会添加到通信队列。 等待队列不存在大小限制，每个通道均具有单独的队列。
3	在每次扫描结束时，通信任务将执行通信队列中的消息以进行传输。 将以轮流循环的方式逐个处理每个通道的队列。 执行每个通道的一条消息，然后该过程继续，直至所有消息均已执行或者通信计划 (10ms) 到期。 上次所处理通道的下一个通道会在下次扫描结束时安排在第一位。

另请参见

[消息执行过程和时间图](#) 参考页数 [225](#)

[通信指令](#) 参考页数 [183](#)

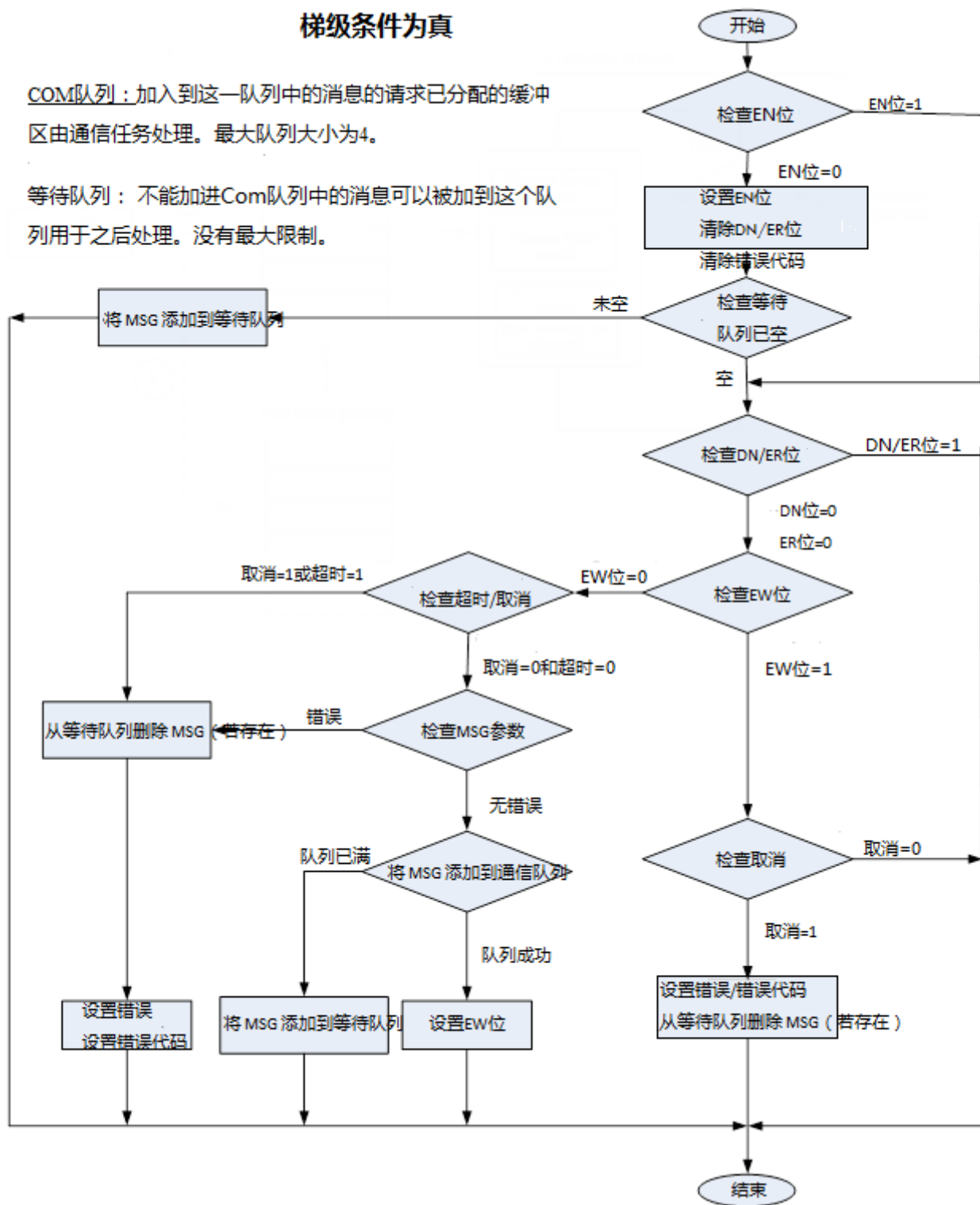
消息执行过程 (Rung = TRUE)

以下过程图描述了当 Rung 条件为 True 时发生的消息指令事件。

梯级条件为真

COM队列：加入到这一队列中的消息的请求已分配的缓冲区由通信任务处理。最大队列大小为4。

等待队列：不能加进Com队列中的消息可以被加到这个队列用于之后处理。没有最大限制。



通信队列：对于已添加到通信队列中的消息请求，它们有一个由通信任务分配和处理的缓冲区。最大队列大小限制为 4。

等待队列：对于无法添加到通信队列的消息，它们会被添加到将在稍后处理的等待队列。等待队列没有最大大小限制。

另请参见

[消息执行过程和时间图](#) 参考页数 [225](#)

[通信指令](#) 参考页数 [183](#)

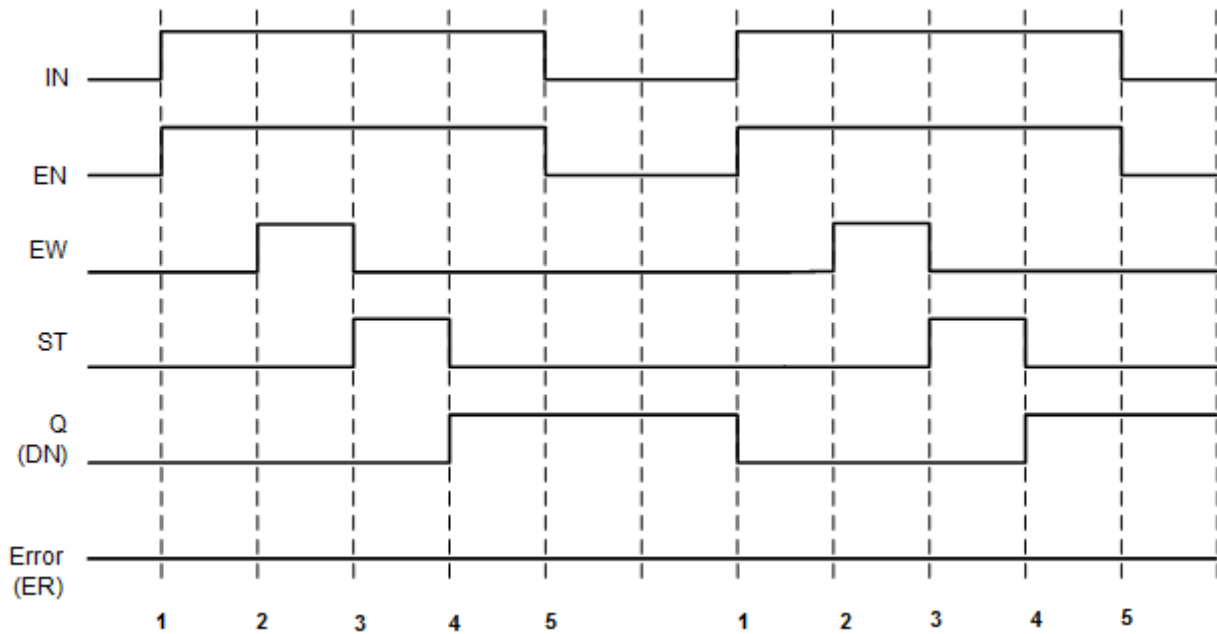
[MSG MODBUS](#) 参考页数 [209](#)

消息执行时间图 (Rung = TRUE)

下表描述了在 Rung 条件保持为 True 的情况下时间图中标识的每个执行步骤的消息条件和位状态。

步骤	消息说明	位状态
1	Rung 条件为 TRUE。 消息执行已启用。	EN 位已设置。 所有其他位已清除。
2	已获取消息控制缓冲区。此时，已复制输入数据（即用于写入消息的 data 参数）以进行传输。在传输的消息中不会反映对输入数据的后续更改。	EW 位已设置。
3	消息传输开始。	EW 位已清除。 ST 位已设置。
4	消息响应已收到。	ST 位已清除。 DN 位已设置。
5	Rung 条件为 FALSE。	EN 位已清除。

Rung = TRUE 的时间图



另请参见

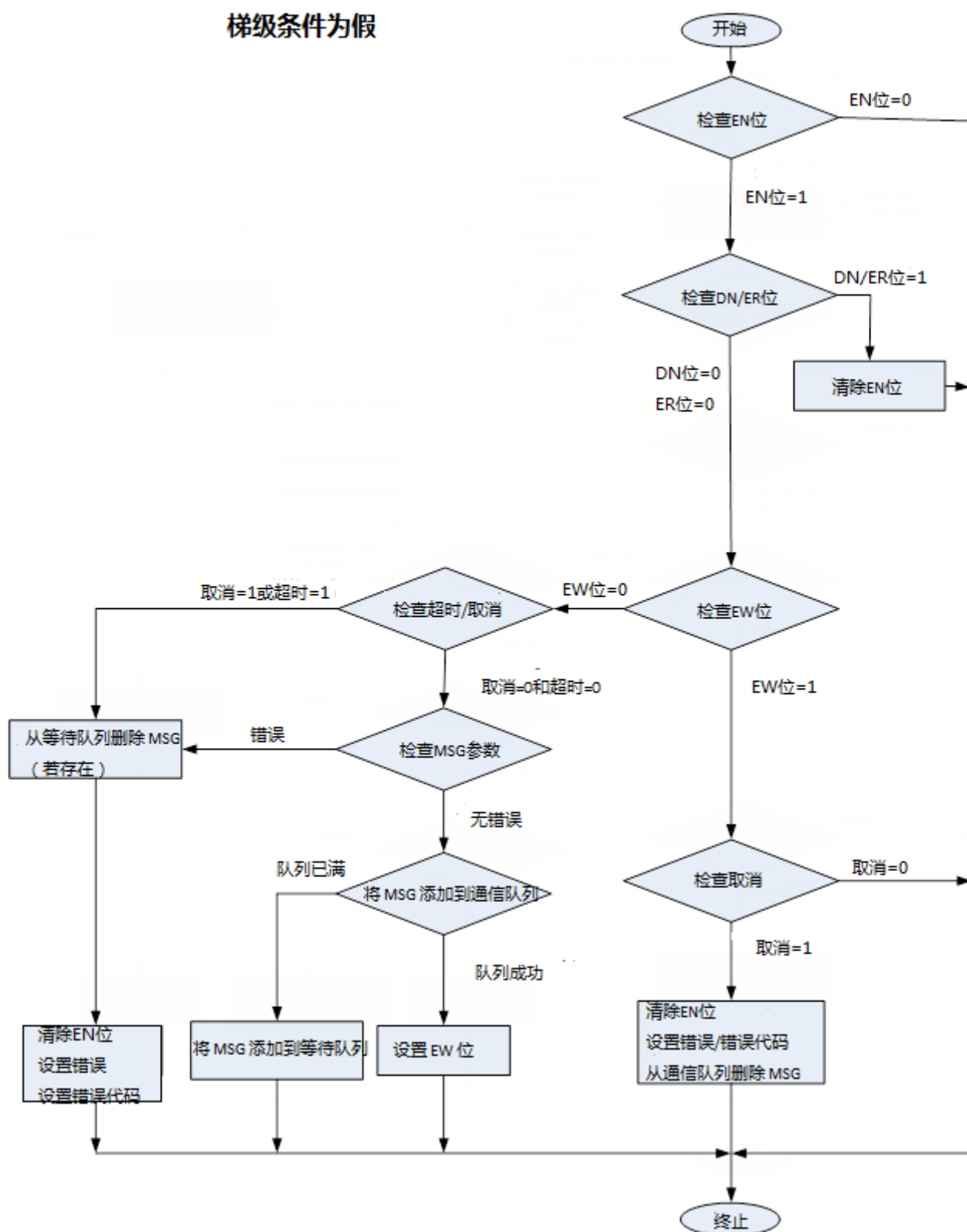
[消息执行过程和时间图](#) 参考页数 [225](#)

[通信指令](#) 参考页数 [183](#)

消息执行过程 (Rung = FALSE)

以下过程图描述了当 Rung 条件为 True 时发生的消息指令事件。

梯级条件为假



另请参见

[消息执行过程和时间图](#) 参考页数 [225](#)

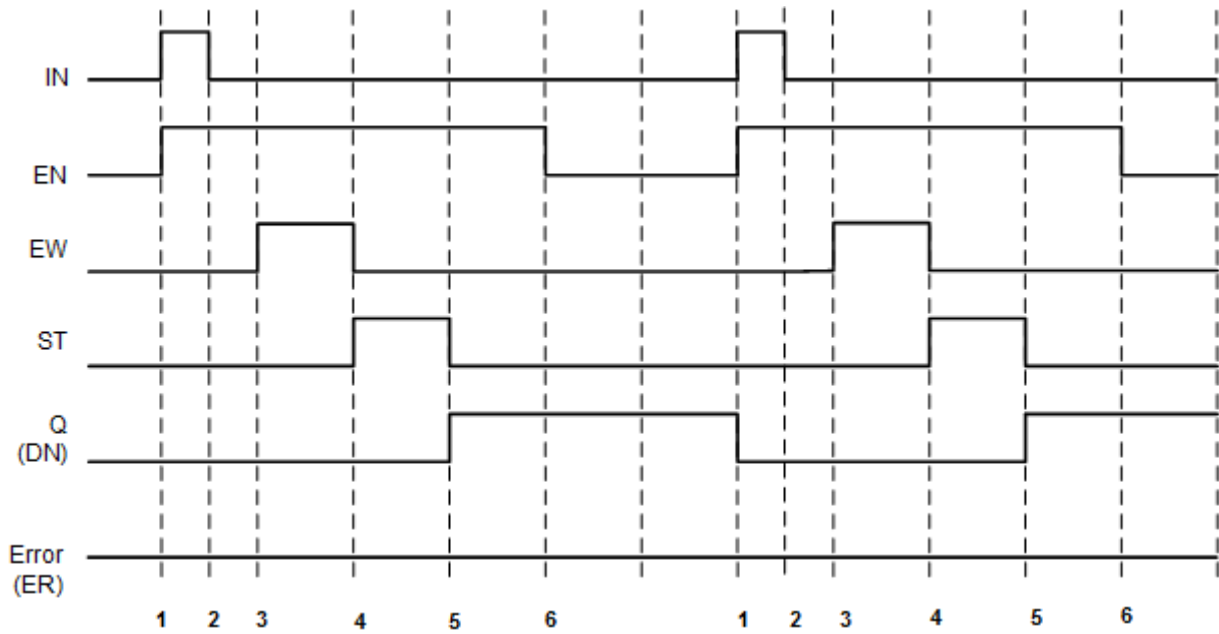
[通信指令](#) 参考页数 [183](#)

消息执行时间图 (Rung = FALSE)

下表描述了 Rung 在执行期间变为 FALSE 的情况下时间图中标识的每个执行步骤的消息条件和位状态。

步骤	消息说明	位状态
1	Rung 条件为 TRUE。 消息执行已启用。	EN 位已设置。 所有其他位已清除。
2	Rung 条件为 FALSE。 消息执行继续。	
3	已获取消息缓冲区。此时，已复制输入数据（即用于写入消息的 data 参数）以进行传输。在传输的消息中不会反映对输入数据的后续更改。	EW 位已设置。
4	消息传输开始。	EW 位已清除。 ST 位已设置。
5	消息响应已收到。	ST 位已清除。 DN 位已设置。
6	在步骤 5 后将再次扫描消息。	EN 位已清除。

Rung = FALSE 的时间图



另请参见

[消息执行过程和时间图](#) 参考页数 [225](#)

[通信指令](#) 参考页数 [183](#)

消息执行过程（错误）

下表描述了执行期间发生错误的情况下时间图中标识的每个执行步骤的消息条件和位状态。

步骤	消息说明	位状态
1	Rung 条件为 TRUE。 消息执行已启用。	EN 位已设置。 所有其他位已清除。
2	已获取消息缓冲区。	EW 位已设置。
3	消息传输开始。	EW 位已清除。 ST 位已设置。
4	消息传输超时。	EW 和 ST 位未改变。

步骤	消息说明	位状态
4-6	Rung 条件为 FALSE。	EW 位已清除。 ER 位已设置。

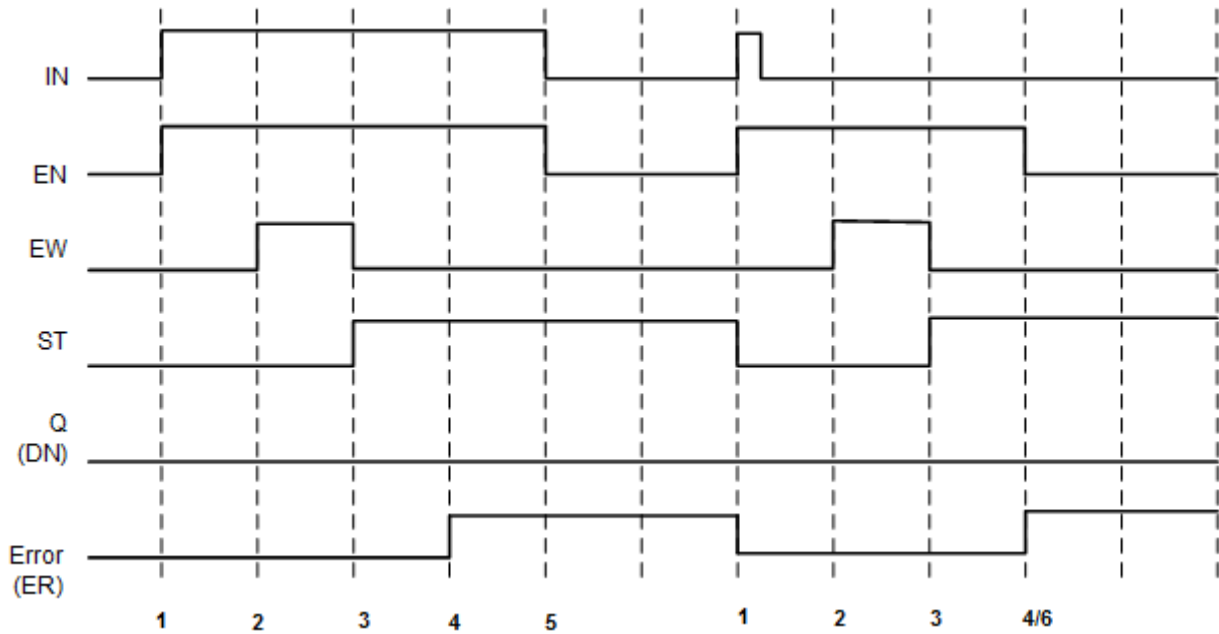
另请参见

[消息执行过程和时间图](#) 参考页数 [225](#)

[通信指令](#) 参考页数 [183](#)

消息执行时间图（错误）

以下时间图展示了在执行期间发生错误时的典型模式。



另请参见

[消息执行过程和时间图](#) 参考页数 [225](#)

[通信指令](#) 参考页数 [183](#)

使用通信（消息）功能块 本节介绍在逻辑程序中使用通信指令的特定详细信息和示例。请参见以下主题，以了解使用 MSG_CIPGENERIC 和 MSG_CIPSYMBOLIC 功能块创建程序的详细信息和示例。

[配置显式消息的对象数据值](#) 参考页数 [236](#)

[示例：如何创建 MSG_CIPGENERIC 消息程序以从控制器中读取数据](#)
参考页数 [239](#)

[示例：如何创建 MSG_CIPSYMBOLIC 消息程序以将值写入变量](#) 参考
页数 [253](#)

[示例：如果配置 Modbus 通信以读取和写入驱动器](#) 参考页数 [268](#)

另请参见

[通信指令](#) 参考页数 [183](#)

配置显式消息的对象数据值 (MSG_CIPGENERIC)

要使用 MSG_CIPGENERIC 功能块处理显式消息，您需要使用正确的值配置 AppCfg 参数。

关于消息通信的其他信息

存在多种消息来源，消息涵盖消息通信的实现和使用，来源包括 Connected Components Workbench 帮助、用户手册和 Rockwell Automation 文献库。

消息通信的信息来源

下表列出了与消息通信有关的其他信息来源。

信息来源	说明	如何查找信息
特定通信设备的用户手册	包含与消息传送有关的重要信息和有关配置消息功能块的特定信息。	Connected Components Workbench 帮助菜单

信息来源	说明	如何查找信息
以太网/IP 适配器 22-COMM-E FRN 1.xxx 附录 C	提供与可使用显式消息访问的以太网/IP 对象有关的信息。	Connected Components Workbench 帮助菜单
以太网/IP 规范	定义每个 CIP 设备中所含的对象：标识对象、消息路由器对象以及网络对象。	ODVA 网站 (http://www.odva.org)
Micro800 可编程控制器：CIP 客户端消息入门	提供在 Micro830 和 Micro850 可编程逻辑控制器 (PLC) 中使用 CIP GENERIC 和 CIP 符号消息的快速入门说明。	Rockwell Automation 文献库

参阅用户手册和快速入门指南

从“帮助”菜单中访问快速入门指南

1. 在帮助菜单中，单击查看帮助。
2. 双击 **Connected Components Workbench**。
3. 双击 **Connected Components Workbench** 入门。

从“帮助”菜单中访问驱动器手册

1. 在帮助菜单中，单击用户手册以显示“手册”对话框。
2. 单击“驱动器”旁的加 (+) 号，以展开目录，然后展开类，直至找到所需手册。
3. 双击手册名以打开 PDF 文件。

从“帮助”菜单中访问以太网/IP 手册

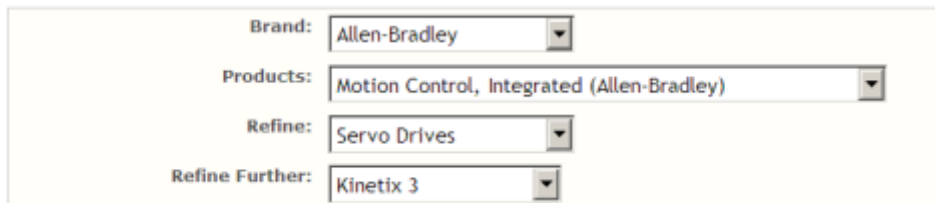
1. 在帮助菜单中，单击用户手册以显示“手册”对话框。
2. 单击驱动器旁的加 (+) 号，以展开类别，然后展开 PowerFlex 4 类外设类。

3. 双击“22-COMM-E 以太网/IP 适配器用户手册”以打开 PDF 文件。

从 Rockwell Automation 文献库访问手册

1. 转至 <http://literature.rockwellautomation.com>
<http://literature.rockwellautomation.com>。
2. 单击高级搜索。
3. 输入产品信息和其他搜索条件。以下示例显示的是 Kinetix 手册的搜索条件：

ADVANCED SEARCH



The image shows a screenshot of the 'ADVANCED SEARCH' form on the Rockwell Automation literature website. The form contains four dropdown menus with the following selected values:

- Brand: Allen-Bradley
- Products: Motion Control, Integrated (Allen-Bradley)
- Refine: Servo Drives
- Refine Further: Kinetix 3

4. 单击搜索。

访问用户手册的非英文语言版本

1. 从“出版物语言”下拉框（右上角）中选择语言。
2. 在搜索框中输入完整或部分设备类型编号。例如：输入 2080-LC30 以查看 Micro830 用户手册。

CIP 注册对象数据

MSG_CIPGENERIC 功能块使用 AppCfg 参数中的 CIP 注册对象数据。该对象数据包括以下内容：

- 类代码
- 实例
- 实例属性

- Service

MSG_CIPGENERIC AppCfg 参数值

可使用输入变量中 CIP 注册对象的值来配置 MSG_CIPGENERIC 功能块参数。下图展示了如何在 MSG_CIPGENERIC 功能块参数中使用 CIP 注册对象数据值。

	名称	数据类型	注释	初始值
	MSG_CIPSYMBOLIC_1	MSG_CIPSYMBOLIC		...
	MSG_ReadDrive	MSG_CIPSYMBOLIC		...
▶	MyAppCfg	CIPAPPCFG		...
	MyAppCfg.ServiceCode	USINT	CIP Service code: 1 - 127	14
	MyAppCfg.ClassID	UINT	CIP Class ID: 1 - 65535	7
	MyAppCfg.InstanceID	UDINT	CIP Instance ID: 0 - 0xFFFFFFFF	4
	MyAppCfg.Attribute	UINT	CIP Attribute: 1 - 65535, 0 - No Attribute	4
	MyAppCfg.MemberCount	USINT	CIP Member ID count: 1 - 3, 0 - No Members	...
	MyAppCfg.MemberID	CIPMEMBERID	CIP Member ID: 0 - 65535	...

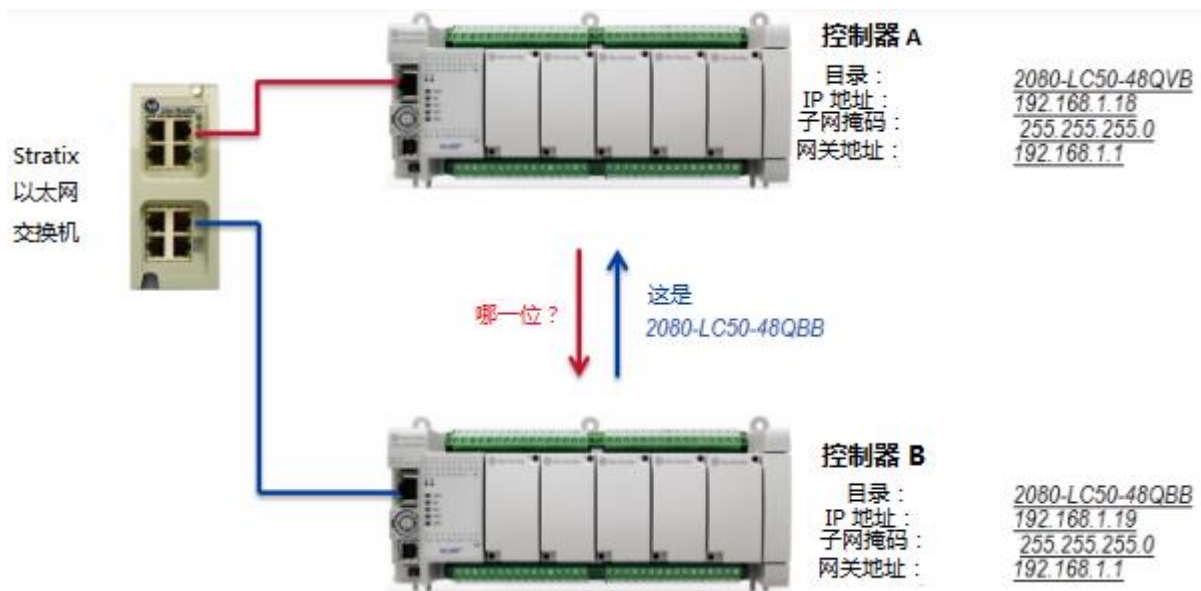
另请参见

[MSG_CIPGENERIC](#) 参考页数 183

[使用通信（消息）功能块](#) 参考页数 236

示例：如何创建 MSG_CIPGENERIC 消息程序以从控制器中读取数据

下例展示了如何使用 MSG_CIPGENERIC 功能块和 COP 功能块创建消息程序，以从控制器 A 检索控制器 B 的目录信息。



创建 MSG_CIPGENERIC 消息程序

执行以下任务，以创建可用于从不同控制器读取信息的 MSG_CIPGENERIC 消息程序。

不支持	任务
1	确定输入变量 (MSG_CIPGENERIC) 的初始值 参考页数 241
2	添加 MSG_CIPGENERIC 功能块和变量 参考页数 242
3	配置变量的初始值 参考页数 244
4	添加触点和线圈 参考页数 247
5	添加 COP 功能块、变量和触点 (MSG_CIPGENERIC) 参考页数 249
6	确认控制器 B 上正确的 IP 配置 参考页数 250

另请参见

[配置显式消息的对象数据值](#) 参考页数 [236](#)

[使用通信（消息）功能块](#) 参考页数 [236](#)

创建 MSG_CIPGENERIC 消息程序

执行以下任务，以创建可用于从不同控制器读取信息的 MSG_CIPGENERIC 消息程序。

不支持	任务
1	确定输入变量 (MSG_CIPGENERIC) 的初始值 参考页数 241
2	添加 MSG_CIPGENERIC 功能块和变量 参考页数 242
3	配置变量的初始值 参考页数 244
4	添加触点和线圈 参考页数 247
5	添加 COP 功能块、变量和触点 (MSG_CIPGENERIC) 参考页数 249
6	确认控制器 B 上正确的 IP 配置 参考页数 250

确定输入变量 (MSG_CIPGENERIC) 的初始值

请按照以下一般步骤获取标识对象值，以用于配置 AppCfg 参数初始值。
添加输入变量和初始值

1. 在“帮助”菜单中，单击用户手册。
2. 展开“驱动器”选择，并找到要使用的通信适配器类型的用户手册（以太网/IP 适配器用户手册）。
3. 双击手册将其打开。
4. 查看“附录”标题，找到内容关于可使用显式消息访问的以太网/IP 对象的部分（附录 C）。
5. 访问“附录”部分，并确定与显式消息有关的对象类型（标识对象）。

6. 根据您将检索到的类型信息，确定 AppCfg 参数的初始值。

以太网/IP 对象数据和 AppCfg 参数示例

下表标识了特定的以太网/IP 对象数据，这些数据将用于从控制器读取类型信息。

输入变量示例	AppCfg 参数	以太网/IP 对象数据选项	说明	初始值
MyAppCfg.Service	Service	服务代码	当 class = Yes 时实施 当 Instance = Yes 时实施 获取单个属性	14 (十六进制值 0x0E)
MyAppCfg.Class	Class	类代码	以太网/IP 对象类 = 标识对象	01
MyAppCfg.Instance	实例	实例	22-COMM-E	01
MyAppCfg.Attribute	属性	实例属性	以短字符串形式获取产品名称和 评级	07

另请参见

[创建 MSG_CIPGENERIC 消息程序](#) 参考页数 [241](#)

[添加泛型消息功能块和变量](#) 参考页数 [242](#)

添加 MSG_CIPGENERIC 功能块和变量

请按照以下步骤启动项目 将 MSG_CIPGENERIC 功能块添加到梯形图程序，然后将输入变量添加到功能块。

添加 MSG_CIPGENERIC 功能块

1. 添加控制器：
 - 展开控制器文件夹和 Micro850 文件夹，以查看所有 Micro850 控制器。
 - 双击控制器 (2080-LC50-48QVB)，以将其添加到“项目组织器”。
2. 添加梯形图程序：

- 在“项目组织器”中，右击程序，单击添加，然后单击新 LD:梯形图。
- 右击“项目组织器”中的梯形图图标，单击重命名，然后键入 CIPExplicitMessage。
- 双击“项目组织器”中的梯形图程序，以在语言编辑器中显示 LD POU。

3. 添加 MSG_CIPGENERIC 功能块：

- 在“工具箱”中，选择指令块，并将其拖放到梯形梯级以显示“模块选择器”。
- 在“搜索”中，键入 MSG 以显示消息功能块。
- 在实例字段中键入 MSG_ReadDrive。
- 双击 MSG_CIPGENERIC，以将功能块实例添加到梯形图。

添加 MSG_CIPGENERIC 变量

1. 添加局部输入变量：

- 在“项目组织器”中，双击局部变量以显示“局部变量”页面。
- 在“变量”页面中，添加表中所列的变量和数据类型。

参数	变量名	数据类型
CtrlCfg	MyCtrlCfg	CIPCONTROLCFG
AppCfg	MyAppCfg	CIPAPPCFG
TargetCfg	MyTargetCfg	CIPTARGETCFG
ReqData	MyReqData	USINT
ReqLength	MyReqLength	UINT
ResData	MyResData	USINT (数组)

2. 对于 MyResData 变量 ,请双击维度 ,并将数组大小更改为 [1..81]。

结果

变量页面应与下图相似。

	Name	Alias	Data Type	Dimension	Project Value	Initial Value
	MSG_ReadDrive		MSG_CIPGENERIC		...	...
	MyCtrlCfg		CIPCONTROLCFG		...	...
	MyAppCfg		CIPAPPCFG		...	...
	MyTargetCfg		CIPTARGETCFG		...	...
	MyReqData		USINT			
	MyReqLength		UINT			
	MyResData		USINT	[1..81]	...	...
*						

另请参见

[创建 MSG_CIPGENERIC 消息程序](#) 参考页数 [241](#)

[确定输入变量 \(MSG_CIPGENERIC\) 的初始值](#) 参考页数 [241](#)

[配置初始值 \(MSG_CIPGENERIC\)](#) 参考页数 [244](#)

配置变量的初始值

请按照以下步骤，将初始值添加到您先前创建的输入变量，然后将变量分配到正确的 MSG_CIPGENERIC 功能块输入参数。

配置 MyCtrlCfg 输入变量的初始值

1. 在局部变量页面中，展开 MyCtrlCfg 以查看其参数。
2. 在每个参数的初始值列，输入以下值。

参数	初始值	注释
MyCtrlCfg.Cancel	留空	不需要。
MyCtrlCfg.TriggerType	0	我们只需检索一次目录编号。
MyCtrlCfg.StrMode	留空	不需要。

配置 MyAppCfg 输入变量的初始值

1. 在局部变量页面中，展开 MyAppCfg 以查看其参数。

2. 在每个参数的初始值列，输入以下值。

参数	初始值
MyAppCfg.Service	14
MyAppCfg.Class	01
MyAppCfg.Instance	01
MyAppCfg.Attribute	07

配置 MyTargetCfg 输入变量的初始值

1. 在局部变量页面中，展开 MyTargetCfg 以查看其参数。
2. 在每个参数的初始值列，输入以下值。

参数	初始值	注释
MyTargetCfg.Path	'4,192.168.100.4'	第一个“4”表示消息将通过嵌入式以太网端口发送。 “192.168.100.4”是驱动器以太网接口的 IP 地址。
MyTargetCfg.CipConnMode	0	对于 CIP 消息，首选断开。
MyTargetCfg.UcmmTimeout	空白	如果消息初始值为空白，则已断开消息的默认超时值为 3000 毫秒。
MyTargetCfg.ConnMsgTimeout	空白	如果消息初始值为空白，则已连接消息的默认超时值为 3000 毫秒。
MyTargetCfg.ConnClose	FALSE	对于已连接的消息，通过将初始值设为 TRUE，可以在完成消息指令后立即关闭 CIP 连接。

结果

“变量”页面中的参数应与下图相似。

Name	Alias	Data Type	Dimension	Project Value	Initial Value
MSG_ReadDrive		MSG_CIPGENERIC		...	...
MyCtrlCfg		CIPCONTROLCFG		...	...
MyCtrlCfg.Cancel		BOOL			
MyCtrlCfg.TriggerType		UDINT			0
MyCtrlCfg.StrMode		USINT			
MyAppCfg		CIPAPPCFG		...	...
MyAppCfg.Service		USINT			14
MyAppCfg.Class		UINT			01
MyAppCfg.Instance		UDINT			01
MyAppCfg.Attribute		UINT			07
MyAppCfg.MemberCnt		USINT			
MyAppCfg.MemberId		CIPMEMBERID		...	...
MyTargetCfg		CIPTARGETCFG		...	...
MyTargetCfg.Path		STRING			'4,192.168.100.4'
MyTargetCfg.CipConnMode		USINT			0
MyTargetCfg.UcmmTimeout		UDINT			
MyTargetCfg.ConnMsgTime		UDINT			
MyTargetCfg.ConnClose		BOOL			FALSE
MyReqData		USINT			
MyReqLength		UINT			
MyResData		USINT	[1..81]	...	...
MyResData[1]		USINT			

将变量分配给参数

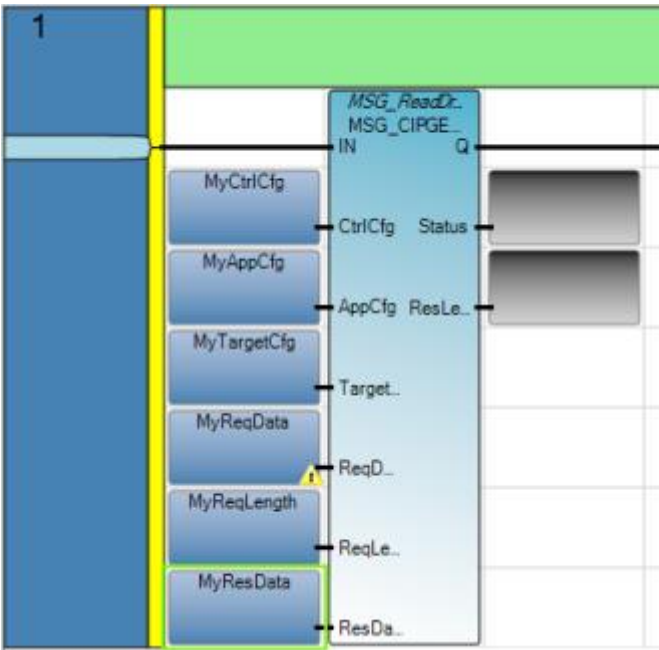
- 1. 在梯形图 POU 中，单击变量输入块顶部，以显示变量下拉列表。
- 2. 从该列表，将每个输入参数分配给如下表中所示的正确输入变量。

参数	输入变量	注释
CtrlCfg	MyCtrlCfg	必须仅检索目录编号一次，因此 MyCtrlCfg.TriggerType 初始值设为 0。
AppCfg	MyAppCfg	初始值是通过查找 Service、Class、Instance 和 Attribute 的对象数据值来确定的。
目标	MyTargetCfg	初始值适用于目标设备配置。
ReqData	MyReqData	由于这是个“读取”消息，不存在请求数据，因此将不使用 ReqData 参数。

ReqLength	MyReqLength	由于这是个“读取”消息，不存在请求数据，因此将不使用 ReqLength 参数。
ResData	MyResData	目录编号字符串以 ODVA 短字符串格式存储在数组中。 第一个数组元素定义字符串长度，其余元素用于存储字符串字符的十六进制值。 最大字符数为 80，再加上长度元素，因此 MyResData 定义为具有 81 个元素的 1 维数组。

结果

您的 MSG_CIPGENERIC 功能块实例应与下图相似。



另请参见

[创建 MSG_CIPGENERIC 消息程序](#) 参考页数 [241](#)

[添加泛型消息功能块和变量](#) 参考页数 [242](#)

[添加触点和线圈](#) 参考页数 [247](#)

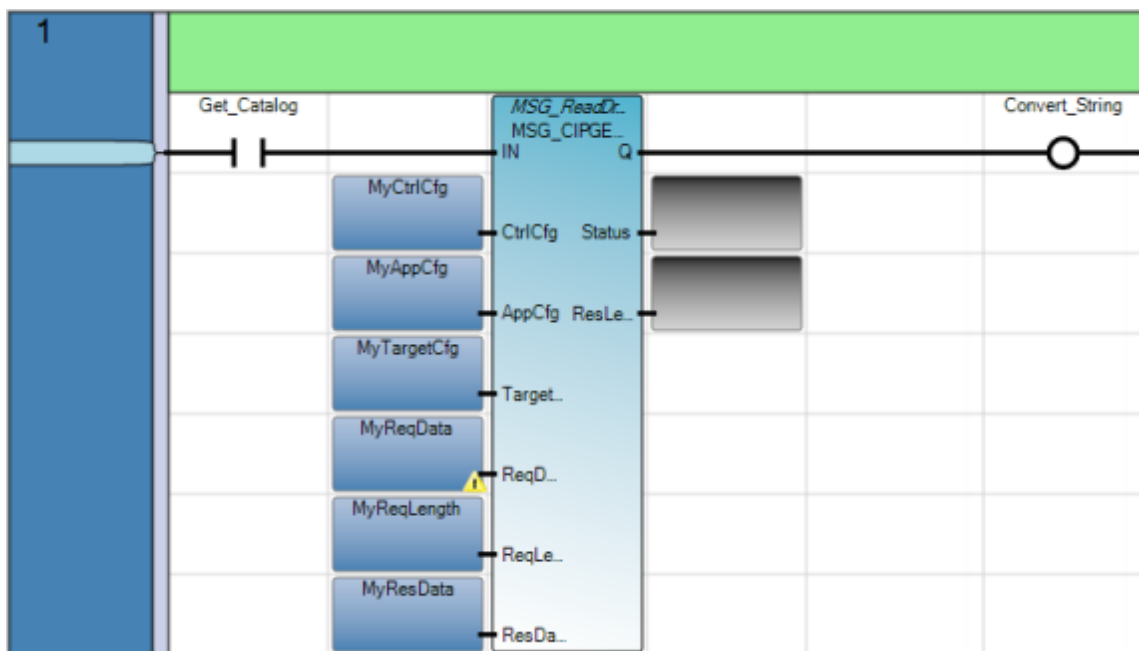
添加触点和线圈

请按照以下步骤，将线圈和触点添加到 MSG_CIPGENERIC 功能块，它们将用于将目录信息转换为人类可读的字符串。

1. 在“工具箱”中，选择正向触点，并将其拖放到第一个梯形梯级中 MSG_CIPGENERIC 功能块输入的左侧。
2. 在“变量选择器”中，在触点的“名称”字段键入 **Get_Catalog**。
3. 在“工具箱”中，选择直接线圈，并将其拖放到第一个梯形梯级中 MSG_CIPGENERIC 功能块输出的右侧。
4. 在“变量选择器”中，在线圈的“名称”字段键入 **Convert_String**。

结果

MSG_CIPGENERIC 消息的梯形图程序的第一个梯级应与下图相似。



另请参见

[创建 MSG_CIPGENERIC 消息程序](#) 参考页数 [241](#)

[配置初始值 \(MSG_CIPGENERIC\)](#) 参考页数 [244](#)

[添加 COP 功能块、变量和触点](#) 参考页数 [249](#)

添加 COP 功能块、变量和触点 (MSG_CIPGENERIC)

请按照以下步骤添加 COP 功能块、变量和触点。COP 指令用于将数据从源数据类型（例如，DINT 或 REAL）转换为目标数据类型。在本例中，目录信息将转换为人类可读的字符串。

添加 COP 功能块

1. 在“工具箱”中,选择梯级,并将其拖放到第一个梯形梯级的正下方,以添加第二个梯级。
2. 添加 COP 功能块：

在“工具箱”中,选择块,并将其拖放到第二个梯形梯级以显示“模块选择器”。

双击 COP,以将功能块实例添加到梯形图。

添加 COP 变量

1. 添加局部输入变量：

在“项目组织器”中,双击局部变量以显示“局部变量”页面。

在“变量”页面中,添加下表中所列的变量和数据类型。

参数	变量名	数据类型
Src	MyResData	数组 USINT
SrcOffset	0	UINT
Dest	CatalogID	数组 STRING
DestOffset	0	UINT
Length	1	UINT
Swap	FALSE	BOOLEAN

2. 对于 CatalogID 变量,请双击维度,并将数组大小改为 [1..1]

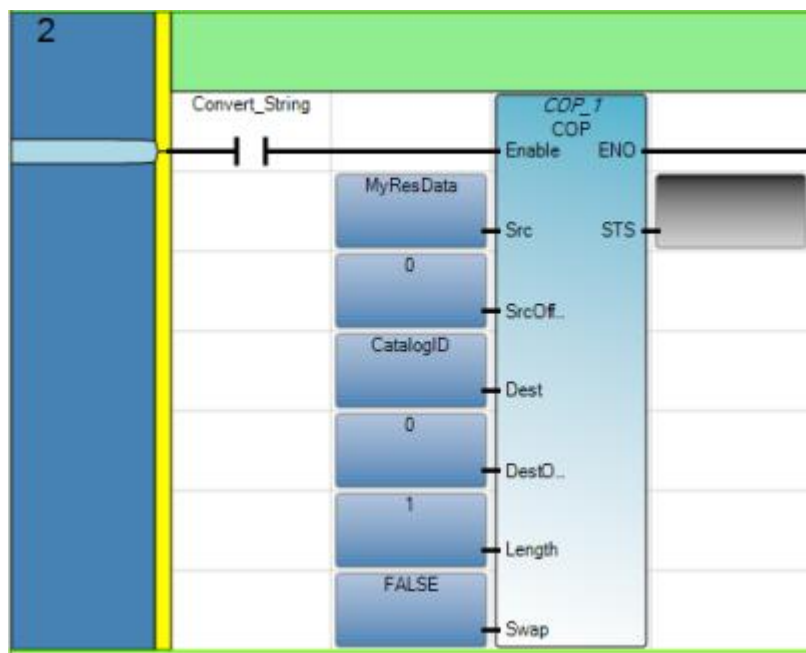
添加触点

1. 在“工具箱”中,选择正向触点,并将其拖放到第二个梯形梯级中 COP 功能块输入的左侧。

2. 在“变量选择器”中，为触点选择 **Convert_String** 变量。

结果

MSG_CIPGENERIC 消息的梯形图程序的第二个梯级应与下图相似。



另请参见

[创建 MSG_CIPGENERIC 消息程序](#) 参考页数 [241](#)

[添加触点和线圈](#) 参考页数 [247](#)

[确认控制器 B 上正确的 IP 配置](#) 参考页数 [250](#)

确认控制器 B 上正确的 IP 配置

请按照以下步骤，验证控制器 B 中的 IP 地址设置是否正确。

1. 打开控制器的应用程序工作区：
2. 在“项目组织器”中，双击控制器以将其显示在应用程序工作区中。

3. 在控制器配置工作区中，展开控制器树中的以太网，然后单击“Internet 协议”，以显示控制器配置页面。
4. 验证 IP 地址设置是否与下表中所示相符。

IP 配置选项	值
IP 地址	192.168.1.19
子网掩码	255.255.255.0
网关地址	192.168.1.1

结果

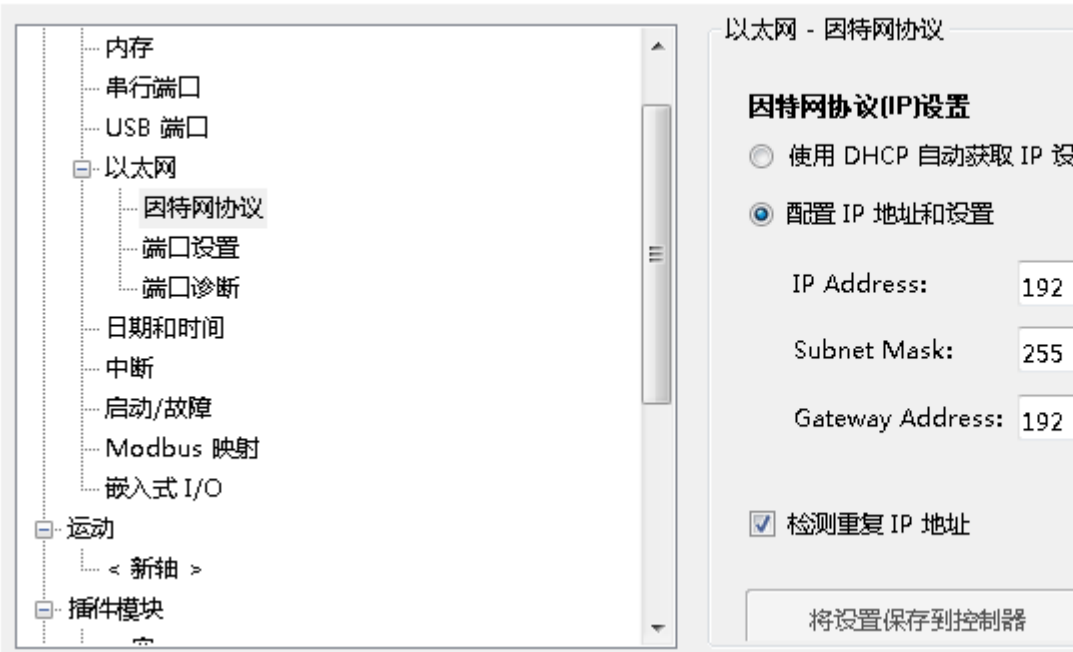
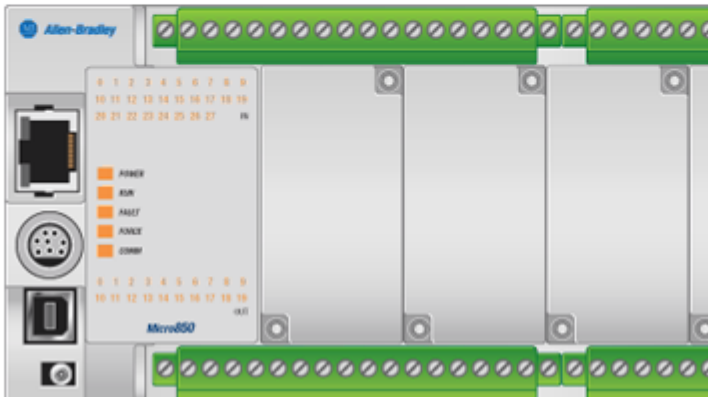
控制器配置页面中的“Internet 协议”选项应与下图相似。

Micro850

Micro850



2080-LC50-48QBB



另请参见

[添加 MSG_CIPSYMBOLIC 功能块、变量和触点](#) 参考页数 [258](#)

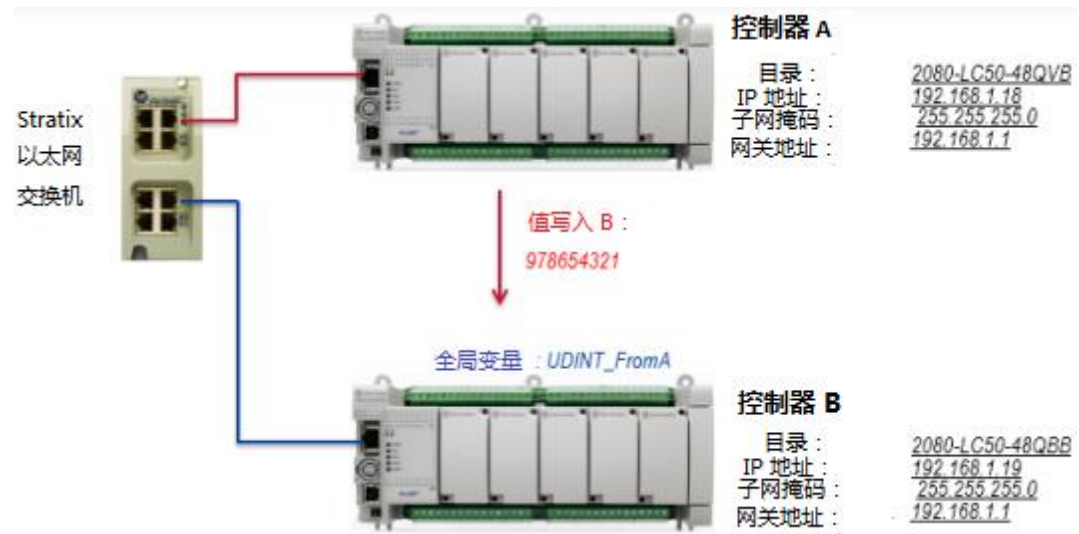
[为控制器 B 创建全局变量](#) 参考页数 [265](#)

[添加 COP 功能块、变量和触点](#) 参考页数 [249](#)

[创建 MSG_CIPGENERIC 消息程序](#) 参考页数 [241](#)

示例：如何创建 MSG_CIPSYMBOLIC 消息程序以将值写入变量

本例展示了如何创建消息程序以从控制器 A 将值写入控制器 B 的全局变量。



创建 MSG_CIPSYMBOLIC 消息程序

执行以下任务，以创建可用于将值写入其他控制器中全局变量的 MSG_CIPSYMBOLIC 消息程序。

不支持	任务
1	添加 COP 功能块、变量和触点 (MSG_CIPSYMBOLIC) 参考页数 254
2	添加 Equal 运算符和线圈 参考页数 257
3	添加 MSG_CIPSYMBOLIC 功能块、变量和触点 参考页数 258
4	确认控制器 B 上正确的 IP 配置 参考页数 250
5	为控制器 B 创建全局变量 参考页数 265

不支持	任务
6	查看整个程序结果 参考页数 265

另请参见

[使用通信（消息）功能块](#) 参考页数 [236](#)

创建 MSG_CIPSYMBOLIC 消息程序

执行以下任务，以创建可用于将值写入其他控制器中全局变量的 MSG_CIPSYMBOLIC 消息程序。

不支持	任务
1	添加 COP 功能块、变量和触点 (MSG_CIPSYMBOLIC) 参考页数 254
2	添加 Equal 运算符和线圈 参考页数 257
3	添加 MSG_CIPSYMBOLIC 功能块、变量和触点 参考页数 258
4	确认控制器 B 上正确的 IP 配置 参考页数 250
5	为控制器 B 创建全局变量 参考页数 265
6	查看整个程序结果 参考页数 265

添加 COP 功能块、变量和触点 (MSG_CIPSYMBOLIC)

请按照以下步骤添加 COP 功能块、变量和触点。COP 指令用于将您输入的数据转换为目标数据类型，从而使其与控制器变量兼容。

添加 COP 功能块

1. 添加控制器：

- 展开控制器文件夹和 Micro850 文件夹，以查看所有 Micro850 控制器。
 - 双击控制器 (2080-LC50-48QVB)，以将其添加到“项目组织器”。
2. 添加梯形图程序：
- 在“项目组织器”中，右击程序，单击添加，然后单击新 LD:梯形图。
 - 右击“项目组织器”中的梯形图图标，单击重命名，然后键入 **CIPSymbolicMessage**。
 - 双击“项目组织器”中的梯形图程序，以在语言编辑器中显示 LD POU。
3. 添加 COP 功能块：
- 在“工具箱”中，选择块，并将其拖放到第一个梯形梯级以显示“模块选择器”。
 - 在“搜索”中，键入 **COP**，然后双击 **COP**，以将功能块实例添加到梯形图。

添加 COP 变量和初始值

1. 添加变量：
- 在梯形图 POU 中，双击局部变量以显示“局部变量”页面。
 - 在“变量”页面中，添加下表中所列的变量和数据类型。
2. 创建数组：
- 对于 ValueToWrite，请双击维度，并将数组大小更改为 [1..1]。
 - 对于 A_Data，请双击维度，并将数组大小更改为 [1..4]。
3. 将下表值列的数据输入每个变量的初始值字段。

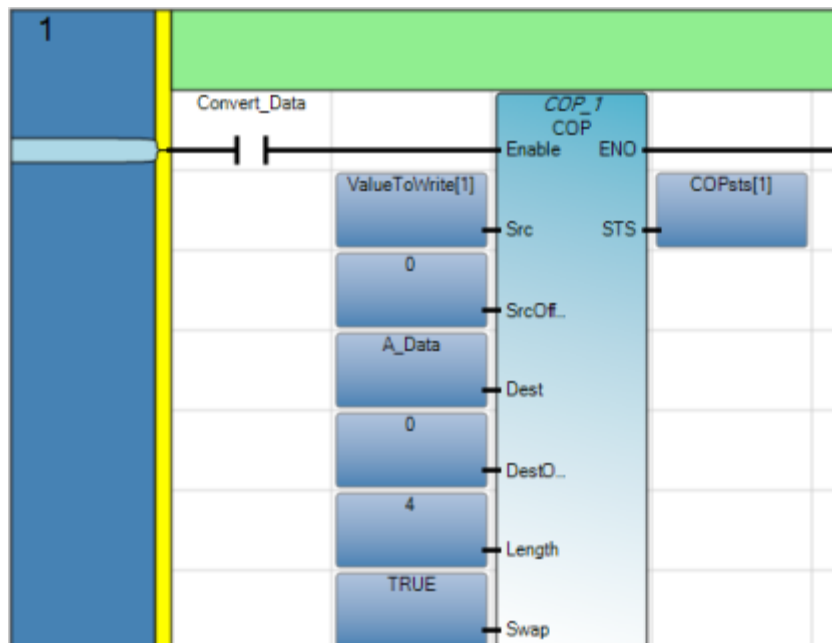
参数	变量名	数据类型
Src	ValueToWrite	数组 UINT
SrcOffset	0	UINT
Dest	A_Data	数组 USINT
DestOffset	0	UINT
Length	4	UINT
Swap	TRUE	BOOLEAN
STS	COPsts	数组 UINT

添加触点

1. 在“工具箱”中，选择正向触点，并将其拖放到第一个梯形梯级中 COP 功能块输入的左侧。
2. 在“变量选择器”中，将名为 **Convert_Data** 的变量分配给触点。

结果

MSG_CIPSYMBOLIC 消息的梯形图程序的第一个梯级应与下图相似。



另请参见

[创建 MSG CIPSYMBOLIC 消息程序](#) 参考页数 [254](#)

[添加 Equal 运算符和线圈](#) 参考页数 [257](#)

添加 **Equal** 运算符和线圈

请按照以下步骤添加 Equal (=) 运算符、变量和线圈。Equal 指令用于在数据类型转换成功时触发写入值。

添加 **Equal** 运算符

- 在“工具箱”中,选择梯级,并将其拖放到第一个梯形梯级的正下方,以添加第二个梯级。
- 添加 Equal 运算符：
 - 在“工具箱”中,选择块,并将其拖放到第二个梯形梯级以显示“模块选择器”。
 - 在“搜索”中,键入“=”,然后双击“=”,以将运算符实例添加到梯形图。

添加 **Equal** 变量

- 在梯形图 POU 中,双击变量以显示变量选择器。
- 在变量选择器中,根据下表所列内容,分配变量名。

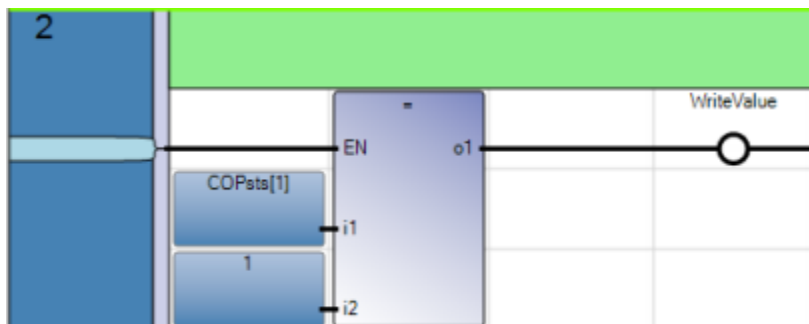
参数	变量名
i1	COPsts
i2	1

将线圈添加到 **Equal** 运算符

- 在“工具箱”中,选择直接线圈,并将其拖放到第二个梯形梯级中 Equal 运算符输出的右侧。
- 在“变量选择器”中,在线圈的“名称”字段键入 **WriteValue**。

结果

MSG_CIPGENERIC 消息的梯形图程序的第二个梯级应与下图相似。



另请参见

[创建 MSG_CIPSYMBOLIC 消息程序](#) 参考页数 [254](#)

[添加 COP 功能块、变量和触点](#) 参考页数 [254](#)

[添加 MSG_CIPSYMBOLIC 功能块、变量和触点](#) 参考页数 [258](#)

添加 MSG_CIPSYMBOLIC 功能块、变量和触点

请按照以下步骤，将 MSG_CIPSYMBOLIC 功能块、输入变量和触点添加到梯形图。

添加功能块和变量

1. 在工具箱中，选择梯级，并将其拖放到第二个梯形梯级的正下方，以添加第三个梯级。
2. 添加 MSG_CIPSYMBOLIC 功能块：

- 在工具箱中，选择指令块，并将其拖放到梯形梯级以显示模块选择器。
- 在“搜索”中，键入 **MSG** 以显示消息功能块。
- 在实例字段中键入 **MSG_WriteVariable**。
- 双击 **MSG_CIPSYMBOLIC**，以将名为 **MSG_WriteVariable** 的功能块的实例添加到梯形图。

3. 分配变量名：

- 在梯形图 POU 中，双击变量以显示变量选择器。
- 在变量选择器中，根据下表所列内容，分配变量名。

4. 对于“数据”变量，请双击维度，并将数组大小更改为 [1..4]。

配置局部变量的初始值

1. 添加 CtrlCfg 的初始值：

- 在“局部变量”页面中，展开 **CtrlCfg** 参数以查看附加参数。
- 在每个参数的“初始值”列，输入以下值。

参数	初始值
A_CtrlCfg.Cancel	留空
A_CtrlCfg.TriggerType	300
A_CtrlCfg.StrMode	留空

2. 添加 SymCfg 的初始值：

- 在“局部变量”页面中，展开 **SymCfg** 参数以查看附加参数。
- 在每个参数的“初始值”列，输入以下值。

参数	初始值
A_SymCfg.Service	1
A_SymCfg.Symbol	'UDINT_FromA'
A_SymCfg.Count	留空
A_SymCfg.DataType	200

A_SymCfg.Offset	留空

结果

局部变量选择器应与下图相似。

	Name	Alias	Data Type	Dimension	Project Value	Initial Value
	MSG_CIPSYMBOLIC_1		MSG_CIPSYMBOL		...	...
	A_CtrlCfg		CIPCONTROLCFG		...	...
	A_CtrlCfg.Cancel		BOOL			
	A_CtrlCfg.TriggerType		UDINT			300
	A_CtrlCfg.StrMode		USINT			
	A_SymCfg		CIPSYMBOLICCFG		...	...
	A_SymCfg.Service		USINT			1
	A_SymCfg.Symbol		STRING			'UDINT_FromA'
	A_SymCfg.Count		UINT			
	A_SymCfg.DataType		USINT			200
	A_SymCfg.Offset		USINT			

配置 TargetCfg 的初始值

- 1. 在梯形图 POU 中，双击 A_TarCfg 变量以打开全局变量选择器。
- 2. 展开 TargetCfg 参数以查看附加参数。
- 3. 在每个参数的“初始值”列，输入以下值。

参数	初始值
A_TarCfg.Path	'4,192.168.1.19'
A_TarCfg.CipConnMode	1
A_TarCfg.UcmmTimeout	0
A_TarCfg.ConnMsgTimeout	0
A_TarCfg.ConnClose	留空

结果

用户全局变量选择器应与下图相似。

Name		Alias	Data Type	Dimension	Project Value	Initial Value
			CIPTARG			
-	A_TarCfg		CIPTARGETCFG		...	...
	A_TarCfg.Path		STRING			'4,192.168.1.19'
	A_TarCfg.CipConnMode		USINT			1
	A_TarCfg.UcmmTimeout		UDINT			0
	A_TarCfg.ConnMsgTimeout		UDINT			0
▶	A_TarCfg.ConnClose		BOOL			

数据参数值

系统将自动从梯级 1 上的 COP 功能块中获取 A_Data 值。此外，请注意，UDINT 是 32 位数据，USINT 是 8 位数据，因此 A_Data 是具有 4 个元素的一维数组

添加触点

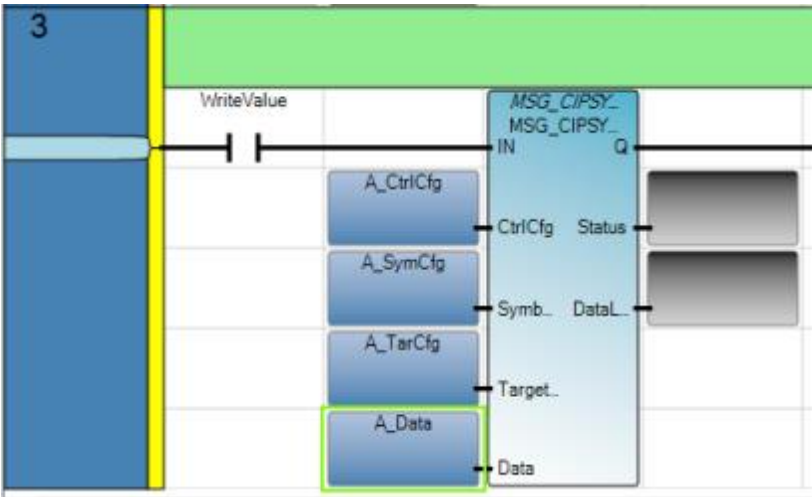
1.

在“工具箱”中，选择正向触点，并将其拖放到第三个梯形梯级中 MSG_CIPSYMBOLIC 功能块输入的左侧。
2.

在“变量选择器”中，将 WriteValue 变量分配给触点。

结果

MSG_CIPSYMBOLIC 消息的梯形图程序的第三个梯级应与下图相似。



另请参见

[创建 MSG_CIPSYMBOLIC 消息程序](#) 参考页数 [254](#)

[添加 Equal 运算符和线圈](#) 参考页数 [257](#)

[确认控制器 B 上正确的 IP 配置](#) 参考页数 [250](#)

确认控制器 B 上正确的 IP 配置

请按照以下步骤，验证控制器 B 中的 IP 地址设置是否正确。

1. 打开控制器的应用程序工作区：
2. 在“项目组织器”中，双击控制器以将其显示在应用程序工作区中。
3. 在控制器配置工作区中，展开控制器树中的以太网，然后单击“Internet 协议”，以显示控制器配置页面。
4. 验证 IP 地址设置是否与下表中所示相符。

IP 配置选项	值
IP 地址	192.168.1.19
子网掩码	255.255.255.0

网关地址	192.168.1.1

结果

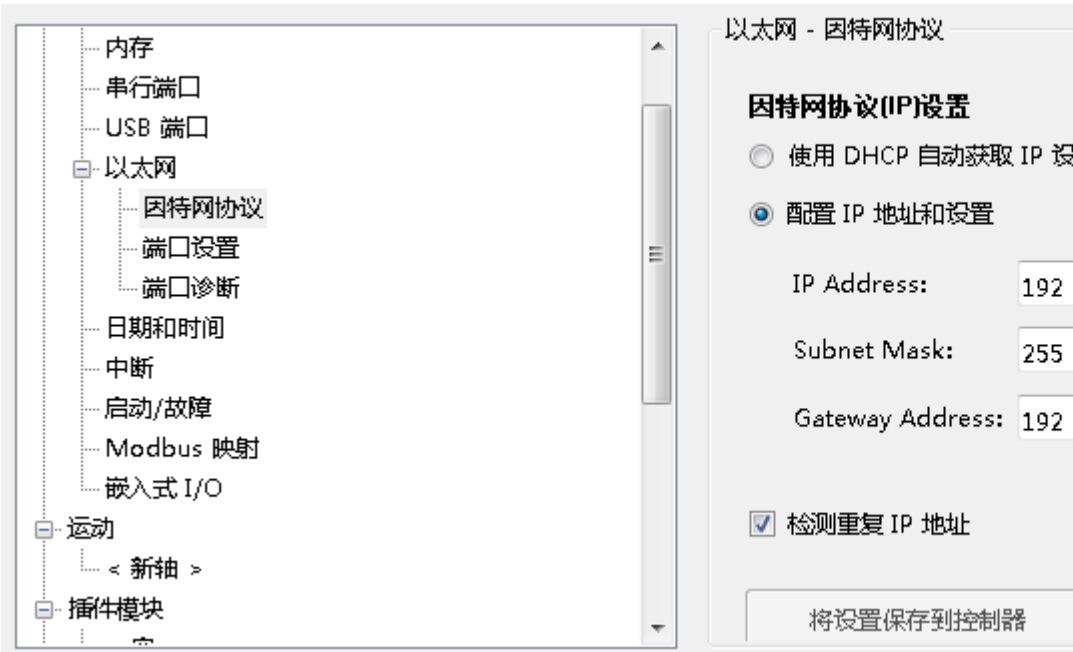
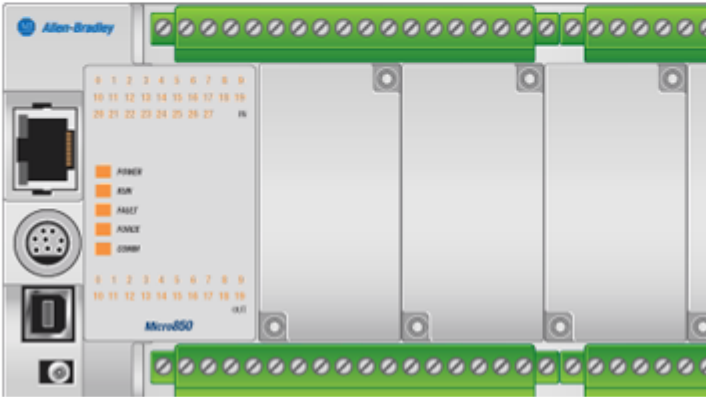
控制器配置页面中的“Internet 协议”选项应与下图相似。

Micro850

Micro850



2080-LC50-48QBB



另请参见

[添加 MSG_CIPSYMBOLIC 功能块、变量和触点](#) 参考页数 [258](#)

[为控制器 B 创建全局变量](#) 参考页数 [265](#)

[添加 COP 功能块、变量和触点](#) 参考页数 [249](#)

[创建 MSG CIPGENERIC 消息程序](#) 参考页数 [241](#)

为控制器 B 创建全局变量

请按照以下步骤，为控制器 B 创建全局变量。

1. 在“项目组织器”中，双击全局变量以显示全局变量选择器。
2. 在名称列中键入 UDINT_ToA。
3. 配置其余参数：
 - 验证数据类型是否为 UDINT。
 - 在初始值字段中键入 123456789。
 - 验证是否已选定读取/写入属性。

结果

全局变量选择器应与下图相似。

CIPSymbolicMessage-POU Micro850-VAR X						
	Name	Alias	Data Type	Dimension	Project Value	Initial Value
+	A_TarCfg		CIPTARGETCFG		...	...
	UDINT_ToA		UDINT			123456789
▶	_IO_EM_DI_00		BOOL			

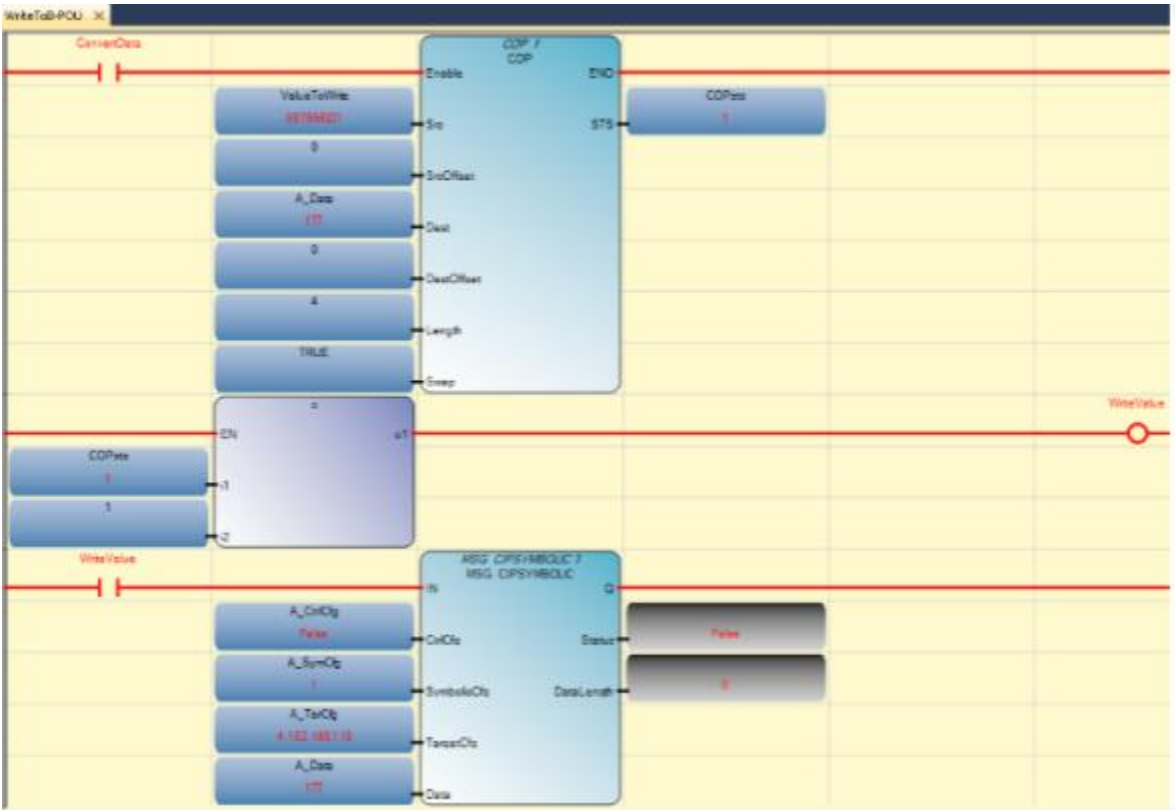
另请参见

[确认控制器 B 上正确的 IP 配置](#) 参考页数 [250](#)

[查看整个程序结果](#) 参考页数 [265](#)

查看整个程序结果

下例展示了整个 MSG_CIPSYMBOLIC 消息程序在其执行后的情况。



控制器 B 的结果

以下两张图片展示了在消息程序执行前后控制器 B 的值。

程序执行前

项目管理器			
名称: Project40_2			
Micro830 程序 全局变量 用户定义的功能块 数据类型			
Micro830-VAR			
名称	逻辑值	实际值	
UDINT_FromA	0	不可用	
_IO_EM_DO_01	☐	☐	
_IO_EM_DO_02	☐	☐	
_IO_EM_DO_03	☐	☐	
_IO_EM_DI_00	☐	☐	
_IO_EM_DI_01	☐	☐	
_IO_EM_DI_02	☐	☐	
_IO_EM_DI_03	☐	☐	
_IO_EM_DI_04	☐	☐	

程序执行后

项目管理器			
名称: Project40_2			
Micro830 程序 全局变量 用户定义的功能块 数据类型			
Micro830-VAR			
名称	逻辑值	实际值	
UDINT_FromA	987654321	不可用	
_IO_EM_DO_01	☐	☐	
_IO_EM_DO_02	☐	☐	
_IO_EM_DO_03	☐	☐	
_IO_EM_DI_00	☐	☐	
_IO_EM_DI_01	☐	☐	
_IO_EM_DI_02	☐	☐	
_IO_EM_DI_03	☐	☐	
_IO_EM_DI_04	☐	☐	

另请参见

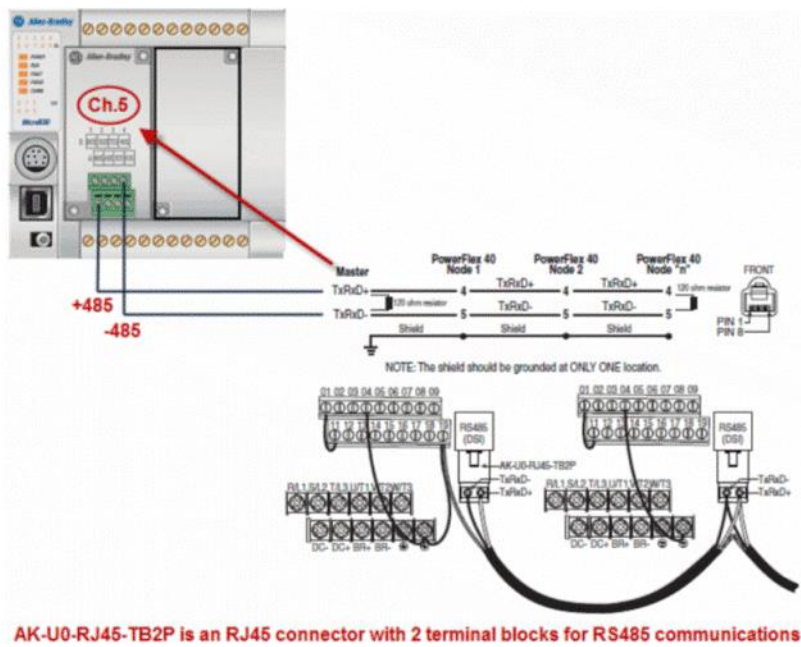
[为控制器 B 创建全局变量](#) 参考页数 265

示例：如果配置 Modbus 通信以读取和写入驱动器

这些示例展示了如何配置 Modbus 通信以便使用 MSG_MODBUS 功能块从 PowerFlex 4 驱动器读取状态数据和向其写入控制数据。

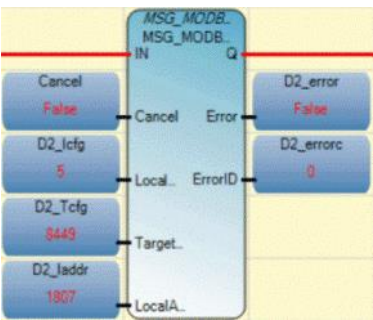
Micro830 配线

本示例使用在第一个插槽（通道 5）中插有 SERIALISOL 模块的 Micro830 控制器。已连接一个 PowerFlex 40 ,但下图说明了如何为多点通信配线。有关其他配线信息，请参考用户手册。



Modbus 读取示例

以下 MSG_MODBUS 指令可用于从 PowerFlex 40 驱动器读取状态数据。



驱动器状态

“1807”指示驱动器

- 已就绪（位 0 置位）
- 处于活动状态（位 1 置位）
- 被指示正向转动（位 2 置位）
- 正在正向转动（位 3 置位）
- 驱动器上某些数字输入的状态

“278”表示 27.8Hz。

有关逻辑状态字位、错误代码说明、指示的及实际速度以及其他状态代码的其他信息，请参考 PowerFlex 用户手册。

MSG_MODBUS 读取配置

下图展示了用于从 PowerFlex 40 驱动器读取状态数据的 MSG_MODBUS 指令块的变量选项。

	名称	数据类型	方向	维度
	MSG_MODBUS_1	MSG_MODI	Var	
	D2_lcfg	MODBUSLC	Var	
	D2_lcfg.Channel	UINT	Var	
	D2_lcfg.TriggerType	USINT	Var	
	D2_lcfg.Cmd	USINT	Var	
	D2_lcfg.ElementCnt	UINT	Var	
	D2_Tcfg	MODBUSL#	Var	
	D2_Tcfg.Addr	UDINT	Var	
	D2_Tcfg.Node	USINT	Var	
	D2_laddr	MODBUSLC	Var	
	D2_laddr[1]	WORD	Var	
	D2_laddr[2]	WORD	Var	
	D2_laddr[3]	WORD	Var	
	D2_laddr[4]	WORD	Var	
	D2_laddr[5]	WORD	Var	

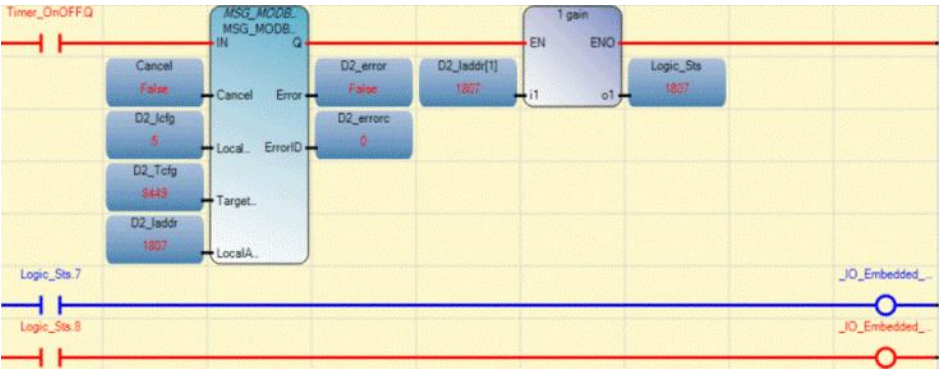
MSG_MODBUS 读取变量

下表标识了用于配置 MSG_MODBUS 指令以从 PowerFlex 4 驱动器读取状态数据的变量和值。

变量	值	说明
*.Channel	5	通道 5 - SERIALISOL 模块的位置
*.TriggerType	0	FALSE - TRUE 转换触发器
*.Cmd	3	Modbus 函数代码“03”- 读取保持寄存器
*.ElementCnt	4	Length
*.Addr	8449	PowerFlex 逻辑状态字地址 + 1
*.Node	2	PowerFlex 节点地址
*_laddr[1]	{data}	PowerFlex 逻辑状态字
*_laddr[2]	{data}	PowerFlex 错误代码
*_laddr[3]	{data}	PowerFlex 指示的速度（速度参考）
*_laddr[4]	{data}	PowerFlex 速度反馈（实际速度）

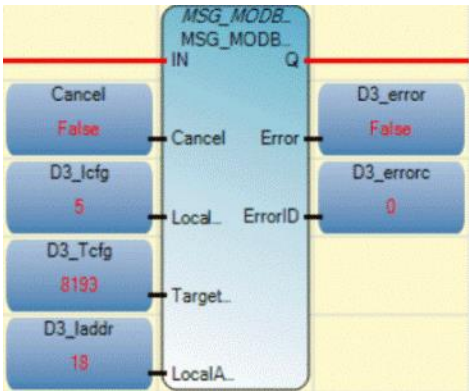
MOV 指令示例

下图展示了使用 MOV 指令将 *_l[1] 数组值移至一个字的示例，这使您能够直接访问各个位。



Modbus 写入示例

以下 MSG_MODBUS 指令用于向 PowerFlex 40 驱动器写入控制数据



MSG_MODBUS 写入配置

下图展示了用于配置 MSG_MODBUS 指令以将控制数据写入 PowerFlex 4 驱动器的变量和值。

	名称	数据类型	方向	维度
	D3_lcfg	MODBUSLC	Var	
	D3_lcfg.Channel	UINT	Var	
	D3_lcfg.TriggerType	USINT	Var	
	D3_lcfg.Cmd	USINT	Var	
	D3_lcfg.ElementCrit	UINT	Var	
	D3_Tcfg	MODBUSL	Var	
	D3_Tcfg.Addr	UDINT	Var	
	D3_Tcfg.Node	USINT	Var	
	D3_laddr	MODBUSLC	Var	
	D3_laddr[1]	WORD	Var	
	D3_laddr[2]	WORD	Var	
	D3_laddr[3]	WORD	Var	

MSG_MODBUS 写入变量

下表列出了这些变量及其值并说明了每个变量的用途。

变量	值	说明
*.Channel	5	通道 5 - SERIALISOL 模块的位置
*.TriggerType	0	FALSE - TRUE 转换触发器
*.Cmd	16	Modbus 函数代码“16”- 写入保持寄存器

变量	值	说明
*.ElementCnt	2	Length
*.Addr	8193	PowerFlex 逻辑状态字地址 + 1
*.Node	2	PowerFlex 节点地址
*_laddr[1]	{data}	PowerFlex 逻辑命令字
*_laddr[2]	{data}	PowerFlex 速度参考字

另请参见

[MSG_MODBUS](#) 参考页数 [209](#)

[使用通信（消息）功能块](#) 参考页数 [236](#)

通信协议支持

MSG_CIP 功能块支持本节所述的各种通信协议。

功能块支持消息通信协议

下表列出了 Modbus 和 CIP 消息功能块支持的通信协议。

消息协议	通信媒介	使用此功能块
Modbus/RTU 客户端和服务器	通过配置为 Modbus RTU 的串行端口	MSG_MODBUS 参考页数 209
Modbus/TCP 客户端和服务器	通过以太网，而非串行端口	MSG_MODBUS2 参考页数 217
以太网 IP 客户端和服务 器	通过嵌入式以太网信道	MSG_CIPSYMBOLIC 参 考页数 195 MSG_CIPGENERIC 参考 页数 183
CIP 串行客户端和服务 器	以太网电缆或串行电缆	MSG_CIPSYMBOLIC 参 考页数 195
ASCII	通过使用 ASCII 驱动程序配置的 RS-232 端口	请参见 ASCII 串行端口指 令 参考页数 119

Modbus RTU

Modbus 是半双工主从通信协议,它使单个主站能够与最多 247 个从属设备通信。Modbus 网络主站读取和写入位与寄存器。通过将串行端口配置为 Modbus RTU,可支持 Modbus RTU。

有关 Modbus 协议的更多信息,请参阅“Modbus 协议规范”(可从 <http://www.modbus.org> 上获取)。

Modbus/TCP

Modbus/TCP Server 通信协议使用与 Modbus RTU 相同的 Modbus 映射功能,但是它通过以太网提供支持,而非通过串行端口。

Micro850 控制器最多支持 16 个并发 Modbus TCP 服务器连接。除配置 Modbus 映射表外,无需配置任何协议。

以太网/IP

Micro850 控制器通过一个嵌入式以太网信道最多支持 16 个并发 EtherNet/IP 服务器连接。该信道可用于通过采用 10 Mbps/100 Mbps 传输速率的局域网将 Micro850 控制器连接到各种设备。

通用工业协议 (CIP)

CIP 串行

CIP 串行使用 DF1 全双工协议,并提供两个设备间的点到点连接。它将数据透明度(美国国家标准机构 ANSI - X3.28-1976 规范 D1 子类别)与具有嵌入式响应的双向并行传输(F1 子类别)相结合

Micro830 和 Micro850 控制器支持采用 CIP 串行通过 RS-232 连接到外部设备,例如运行 RSLinx Classic 软件的计算机、PanelView Component 终端(固件版本 1.70 及以上)或在 DF1 全双工上支持 CIP 串行的其他控制器(例如具有嵌入式串行端口的 ControlLogix 和 CompactLogix 控制器)。

可通过设备配置结构树 ,将串行端口驱动程序配置为 CIP 串行、Modbus RTU、ASCII 或“关闭”。

嵌入式通信信道

Micro830 和 Micro850 控制器具有以下附加的嵌入式通信信道：

- 非隔离 RS-232/485 复合端口
- 非隔离 USB 编程端口
- RJ-45 以太网端口 (仅限 Micro850)

另请参见

[通信协议支持](#) 参考页数 [272](#)

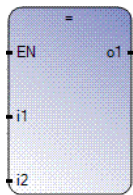
比较指令

比较指令用于使用表达式或特定的比较指令来比较值。

运算符	说明
(=) Equal 参考页数 275	将第一个输入与第二个输入进行比较，以确定整型、实型、时间、日期和字符串数据类型是否相等。
(>) Greater Than 参考页数 277	对于整型、实型、时间、日期和字符串值，比较输入值以确定第一个输入是否大于第二个输入。
(>=) Greater Than or Equal 参考页数 279	对于整型、实型、时间、日期和字符串值，比较输入值以确定第一个输入是否大于或等于第二个输入。
(<) Less Than 参考页数 280	对于整型、实型、时间、日期和字符串值，比较输入值以确定第一个输入是否小于第二个输入。
(<=) Less Than or Equal 参考页数 281	对于整型、实型、时间、日期和字符串值，比较输入值以确定第一个输入是否小于或等于第二个输入。
(<>) Not Equal 参考页数 282	对于整型、实型、时间、日期和字符串值，比较输入值以确定第一个输入是否不等于第二个输入。

等于

Equal (=) 将第一个输入与第二个输入进行比较，以确定整型、实型、时间、日期和字符串数据类型是否相等。



建议：使用 **Equal (=)** 运算符

不建议对 TON、TP 和 TOF 函数进行时间值相等测试。

不建议对实型数据类型进行相等值比较，因为数学运算中数字的舍入方法与变量输出画面中出现的数字不同。结果会造成两个输出值可能在显示器中显示相等，但评估结果仍为错误。例如，在变量输入画面中，23.500001 与 23.499999 都将显示为 23.5，但在控制器中则不相等。有关确定是否相等的其他方法，请参见以下主题。

参数

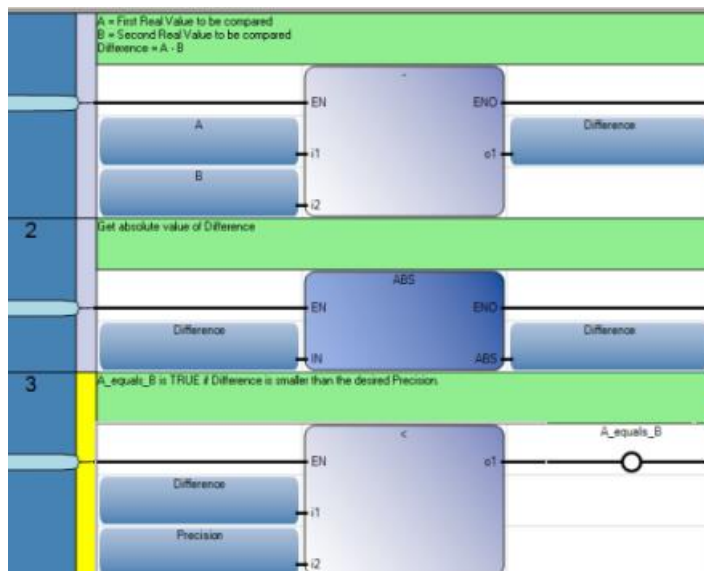
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行相等比较。 当启用 = 假时，不执行比较。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	所有输入的数据类型必须相同。 时间输入适用于 ST、LD 和 FBD 语言。 注意：不建议实型数据类型进行相等值比较。
i2	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	
o1	输出	BOOL	TRUE (如果 i1 = i2) 。

示例：使用 **Subtraction (-)**、**ABS** 和 **Less than (<)** 比较实型值

比较各值是否相等时，不建议使用实型数据类型，因为这比较的是经过舍入的数字。两个输出值在 Connected Components Workbench 画面中可能显示为相等，但评估为 FALSE。

例如，在变量输入画面中，23.500001 与 23.499999 都将显示为 23.5，但在控制器中则不相等。

要测试两个实型数据类型的值是否相等，您可以使用相减指令得出两值之差，然后判断该差值是否小于设定的精度值。有关比较两个实型数据类型的值的信息，请参见以下 LD 程序示例。



Equal (=) 运算符 ST 语言示例

示例

(* 与之等效的 ST : *)

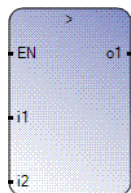
```
aresult := (10 = 25); (* aresult 为 FALSE *)
mresult := ('ab' = 'ab'); (* mresult 为 TRUE *)
```

另请参见

[比较指令](#) 参考页数 [275](#)

Greater than

对于整型、实型、时间、日期和字符串值，Greater Than 比较输入值以确定第一个输入是否大于第二个输入。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行输入比较。 当启用 = 假时，不执行比较。 仅适用于 LD 编程。
i1	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	所有输入的数据类型必须相同。
i2	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	
o1	输出	BOOL	TRUE (如果 i1 > i2) 。

Greater than (>) 运算符 ST 语言示例

(* 与之等效的 ST : *)

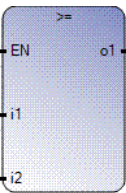
```
aresult := (10 > 25); (* aresult 为 FALSE *)  
mresult := ('ab' > 'a'); (* mresult 为 TRUE *)
```

另请参见

[比较指令](#) 参考页数 [275](#)

Greater Than or Equal

对于整型、实型、时间、日期和字符串值，Greater Than or Equal 比较输入值以确定第一个输入是否大于或等于第二个输入。



>= 运算符的特别建议

对于 TON、TP 和 TOF，不建议进行时间值的相等测试。

参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行输入比较。 当启用 = 假时，不执行比较。 仅适用于 LD 编程。
i1	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	所有输入的数据类型必须相同。时间输入适用于 ST、LD 和 FBD 语言。
i2	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	
o1	输出	BOOL	TRUE（如果 i1 >= i2）。

>= 运算符 ST 语言示例

(* 与之等效的 ST : *)

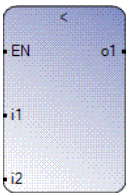
```
areult := (10 >= 25); (* areult 为 FALSE *)
mresult := ('ab' >= 'ab'); (* mresult 为 TRUE *)
```

另请参见

[比较指令](#) 参考页数 [275](#)

Less Than

对于整型、实型、时间、日期和字符串值，Less Than 比较输入值以确定第一个输入是否小于第二个输入。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行输入比较。 当启用 = 假时，不执行比较。 仅适用于 LD 编程。
i1	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	所有输入的数据类型必须相同。
i2	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	
o1	输出	BOOL	TRUE (如果 i1 < i2) 。

Less than (<) 运算符 ST 语言示例

(* 与之等效的 ST : *)

```
areult := (10 < 25); (* areult 为 TRUE *)
mresult := ('z' < 'B'); (* mresult 为 FALSE *)
```

(* 与之等效的 IL : *)

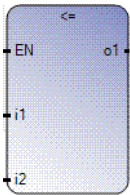
LD	10
LT	25
ST	areult
LD	'z'
LT	'B'
ST	mresult

另请参见

[比较指令](#) 参考页数 [275](#)

Less Than or Equal

对于整型、实型、时间、日期和字符串值，Less Than or Equal 比较输入值以确定第一个输入是否小于或等于第二个输入。



特别建议

对于 TON、TP 和 TOF，不建议进行时间值的相等测试。

参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行输入比较。 当启用 = 假时，不执行比较。 仅适用于 LD 编程。

参数	参数类型	数据类型	说明
i1	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	所有输入的数据类型必须相同。时间输入适用于 ST、LD 和 FBD 语言。
i2	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	
o1	输出	BOOL	TRUE （如果 i1 <= i2 ）。

Less than or equal (<=) 运算符 ST 语言示例

(* 与之等效的 ST : *)

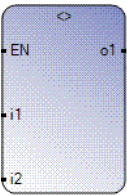
```
aresult := (10 <= 25); (* aresult 为 TRUE *)
mresult := ('ab' <= 'ab'); (* mresult 为 TRUE *)
```

另请参见

[比较指令](#) 参考页数 [275](#)

Not equal

对于整型、实型、时间、日期和字符串值，Not Equal 比较输入值以确第一个输入是否不等于第二个输入。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行当前比较计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	所有输入的数据类型必须相同。
i2	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	
o1	输出	BOOL	TRUE (如果第一个 <> 第二个) 。

Not equal (<>) 运算符 ST 语言示例

(* 与之等效的 ST : *)

```
areresult := (10 <> 25); (* areresult 为 TRUE *)  
mresult := ('ab' <> 'ab'); (* mresult 为 FALSE *)
```

另请参见

[比较指令](#) 参考页数 [275](#)

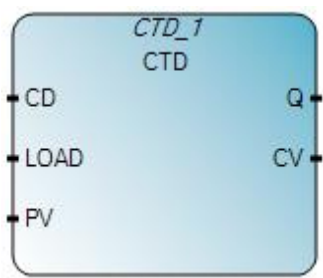
计数器指令

计数器指令用于根据事件数量控制操作。

函数	说明
CTD 参考页数 285	此功能块从给定值到 0 逐个向下计数（整数）。
CTU 参考页数 287	从 0 到给定值逐个向上计数（整数）。
CTUD 参考页数 289	从 0 到给定值逐个向上计数（整数），或从给定值到 0（逐个）向下计数。

CTD

CTD 从给定值到 0 逐个向下计数（整数）。



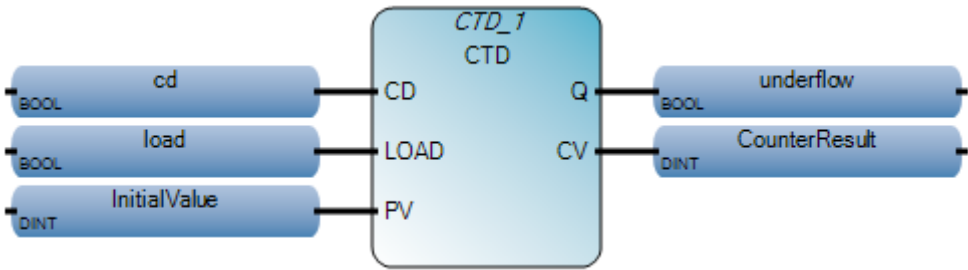
参数

参数	参数类型	数据类型	说明
CD	输入	BOOL	对输入进行计数 （当 CD 为上升沿时向下计数）。
LOAD	输入	BOOL	加载命令（基准） （当 LOAD 为 TRUE 时 CV = PV）。
PV	输入	DINT	编程初始值。
Q	输出	BOOL	下溢：当 CV <= 0 时为 TRUE。

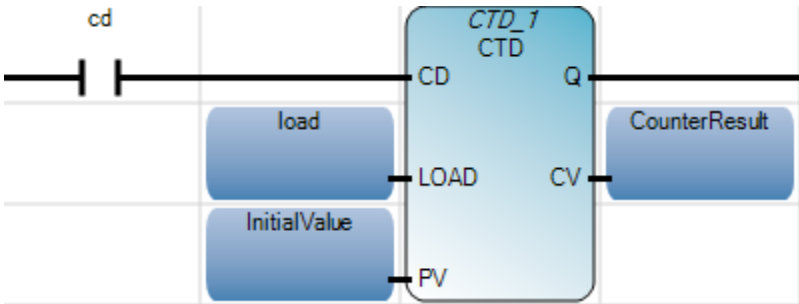
参数	参数类型	数据类型	说明
CV	输出	DINT	计数器结果。

CTD 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 InitialValue := 10;  
2 CTD_1(cd, load, InitialValue);
```

```
CTD_1(  
void CTD_1(BOOL CD, BOOL LOAD, DINT PV)  
Type : CTD, Down counter
```

```
(* 与之等效的 ST: CTD1 是块的一个实例 *)  
CTD1(trigger,load_cmd,100);  
underflow := CTD1.Q;  
result := CTD1.CV;
```

结果

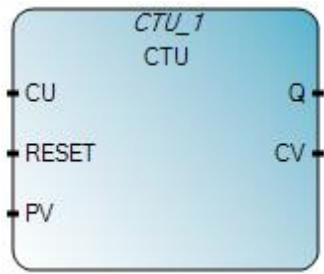


另请参见

[计数器指令](#) 参考页数 [285](#)

CTU

CTU 从 0 到给定值逐个向上计数（整数）。

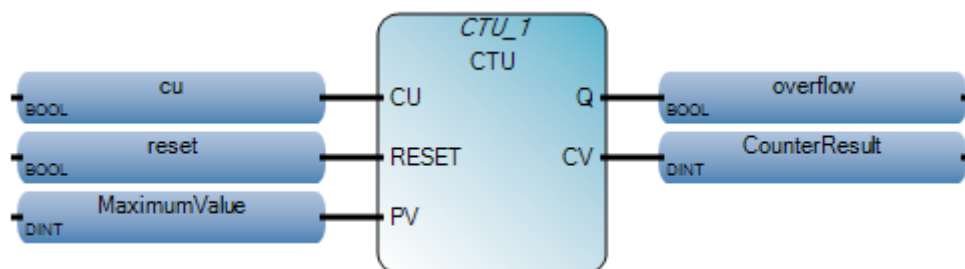


参数

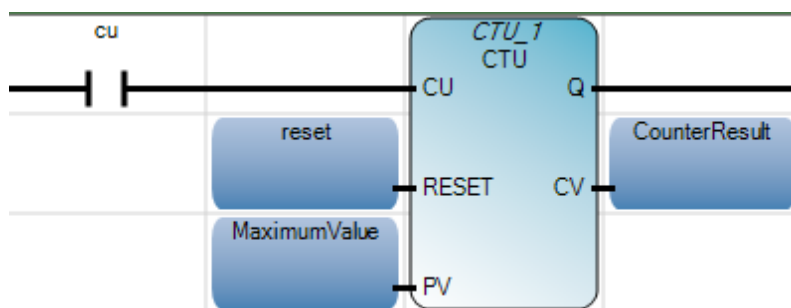
参数	参数类型	数据类型	说明
CU	输入	BOOL	对输入进行计数（当 CU 为上升沿时进行计数）。
RESET	输入	BOOL	重置基准命令。
PV	输入	DINT	编程最大值。
Q	输出	BOOL	溢出：当 CV >= PV 时为 TRUE。
CV	输出	DINT	计数器结果。

CTU 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构文本

```

1 MaximumValue := 10;
2 CTU_1(cu, reset, MaximumValue);

```

```

CTU_1(
void CTU_1(BOOL CU, BOOL RESET, DINT PV)
Type : CTU, Up counter

```

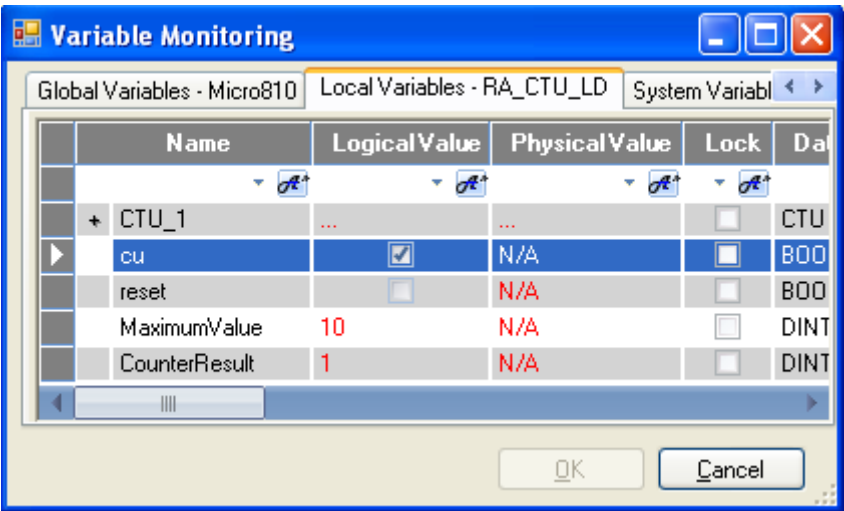
(* 与之等效的 ST : CTU1 是 CTU 块的一个实例 *)

```

CTU1(trigger, NOT(auto_mode), 100);
overflow := CTU1.Q;
result := CTU1.CV;

```

结果

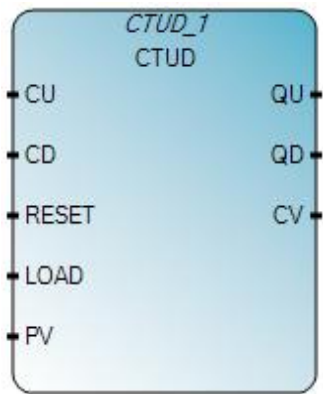


另请参见

[计数器指令](#) 参考页数 [285](#)

CTUD

CTUD 从 0 到给定值逐个向上计数（整数），或从给定值到 0（逐个）向下计数。



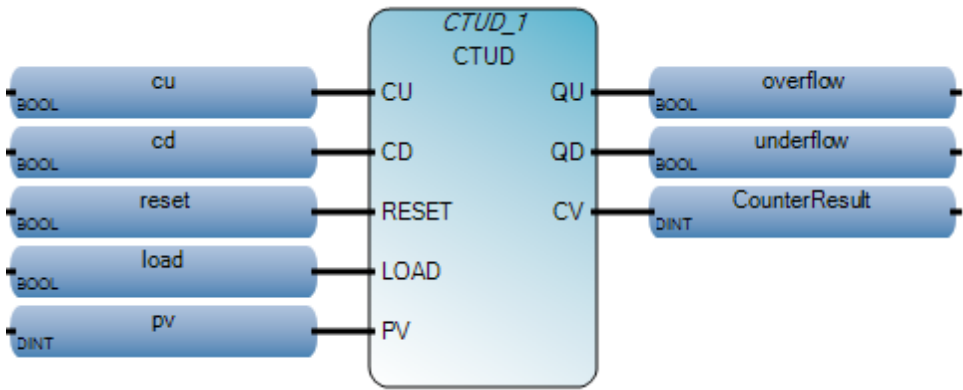
参数

参数	参数类型	数据类型	说明
CU	输入	BOOL	向上计数（当 CU 为上升沿时）。
CD	输入	BOOL	向下计数（当 CD 为上升沿时）。

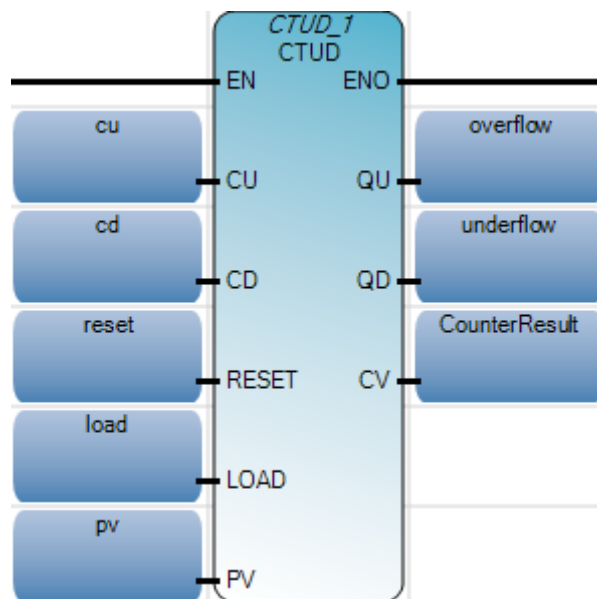
参数	参数类型	数据类型	说明
RESET	输入	BOOL	重置基准命令。 (当 RESET 为 TRUE 时 CV = 0)。
LOAD	输入	BOOL	加载命令 (当 LOAD 为 TRUE 时 CV = PV)。
PV	输入	DINT	编程最大值。
QU	输出	BOOL	溢出：当 $CV \geq PV$ 时为 TRUE。
QD	输出	BOOL	下溢：当 $CV \leq 0$ 时为 TRUE。
CV	输出	DINT	计数器结果。

CTUD 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1 cu := TRUE;
2 cd := TRUE;
3 reset := FALSE;
4 load := FALSE;
5 pv := 10;
6 CTUD_1(cu, cd, reset, load, pv);

```

CTUD_1(

void **CTUD_1**(BOOL CU, BOOL CD, BOOL RESET, BOOL LOAD, DINT PV)
Type : CTUD, Up-down counter

(* 与之等效的 ST: 我们假定 CTUD1 是块的实例 *)

```

CTUD1(trigger1, trigger2, reset_cmd, load_cmd, 100);
full := CTUD1.QU;
empty := CTUD1.QD;
nb_elt := CTUD1.CV;

```

另请参见

[计数器指令](#) 参考页数 [285](#)

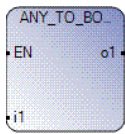
数据转换指令

数据转换指令用于将变量的数据类型转换为不同的数据类型。

运算符	说明
ANY_TO_BOOL 参考页数 294	转换为布尔
ANY_TO_BYTE 参考页数 295	转换为 BYTE
ANY_TO_DATE 参考页数 296	转换为日期
ANY_TO_DINT 参考页数 297	转换为双整型
ANY_TO_DWORD 参考页数 298	转换为双字
ANY_TO_INT 参考页数 299	转换为整型
ANY_TO_LINT 参考页数 300	转换为长整型
ANY_TO_LREAL 参考页数 301	转换为长实型
ANY_TO_LWORD 参考页数 302	转换为长字
ANY_TO_REAL 参考页数 303	转换为实型
ANY_TO_SINT 参考页数 304	转换为短整型
ANY_TO_STRING 参考页数 305	转换为字符串
ANY_TO_TIME 参考页数 307	转换为时间
ANY_TO_UDINT 参考页数 308	转换为无符号双整型
ANY_TO_UINT 参考页数 310	转换为无符号整型
ANY_TO_ULINT 参考页数 311	转换为无符号长整型
ANY_TO_USINT 参考页数 312	转换为无符号短整型
ANY_TO_WORD 参考页数 313	转换为字

ANY_TO_BOOL

ANY_TO_BOOL 将值转换为布尔值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 BOOLEAN 计算。 当 Enable = FALSE 时，不执行计算。
i1	输入	SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	任何非布尔值。
o1	输出	BOOL	布尔值。

ANY_TO_BOOL 运算符 ST 语言示例

(* 与之等效的 ST : *)

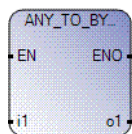
```
ares := ANY_TO_BOOL (10);           (* ares 为 TRUE *)  
tres := ANY_TO_BOOL (t#0s);         (* tres 为 FALSE *)  
mres := ANY_TO_BOOL ('FALSE');      (* mres 为 FALSE *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_BYTE

ANY_TO_BYTE 将值转换为 8 位字节值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 8 位 BYTE 计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	任何非字节值。
o1	输出	BYTE	8 位字节值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_BYTE 运算符 ST 语言示例

(* 与之等效的 ST : *)

```

bres := ANY_TO_BYTE (true);           (* bres 为 1 *)
tres := ANY_TO_BYTE (t#0s46ms);       (* tres 为 46 *)
mres := ANY_TO_BYTE ('0198');         (* mres 为 198 *)

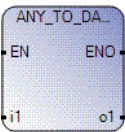
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_DATE

ANY_TO_DATE 将值转换为日期值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为日期计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - STRING	除日期之外的任何值。
o1	输出	DATE	IN 表示的日期。值 -1 表示无效日期。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_DATE 运算符 ST 语言示例

(* 与之等效的 ST : *)

```
ares := ANY_TO_DATE (1109110199);      (* ares := d#2005-02-22 *)  
rres := ANY_TO_DATE (1109110199.3);    (* rres := d#2005-02-22 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_DINT

ANY_TO_DINT 将值转换为 32 位双整型值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 32 位双整型计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	除双整型之外的任何值。
o1	输出	DINT	32 位双整型值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_DINT 运算符 ST 语言示例

(* 与之等效的 ST : *)

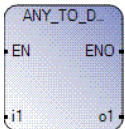
```
bres := ANY_TO_DINT (true);           (* bres 为 1 *)
tres := ANY_TO_DINT (t#1s46ms);       (* tres 为 1046 *)
mres := ANY_TO_DINT ('0198');         (* mres 为 198 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_DWORD

ANY_TO_DWORD 将值转换为 32 位双字值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 32 位双字计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	除双字之外的任何值。
o1	输出	DWORD	32 位双字值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_DWORD 运算符 ST 语言示例

(* 与之等效的 ST : *)

```
bres := ANY_TO_DWORD (true);           (* bres 为 1 *)
tres := ANY_TO_DWORD (t#1s46ms);       (* tres 为 1046 *)
```

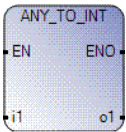
```
mres := ANY_TO_DWORD ('0198');      (* mres 为 198 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_INT

ANY_TO_INT 将值转换为 16 位整型值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 16 位整型计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	任何非 16 位整型值。
o1	输出	INT	16 位整型值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_INT 运算符 ST 语言示例

(* 与之等效的 ST : *)

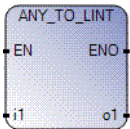
```
bres := ANY_TO_INT (true);           (* bres 为 1 *)
tres := ANY_TO_INT (t#0s46ms);       (* tres 为 46 *)
mres := ANY_TO_INT ('0198');          (* mres 为 198 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_LINT

ANY_TO_LINT 将值转换为 64 位整型值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 64 位长整型计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	除长整型之外的任何值。
o1	输出	LINT	64 位长整型值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_LINT 运算符 ST 语言示例

(* 与之等效的 ST : *)

```
bres := ANY_TO_LINT (true);           (* bres 为 1 *)
tres := ANY_TO_LINT (t#0s46ms);       (* tres 为 46 *)
mres := ANY_TO_LINT ('0198');          (* mres 为 198 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_LREAL

ANY_TO_LREAL 将任意值转换为长实型值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为长实型计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - TIME - DATE - STRING	除长实型之外的任何值。
o1	输出	LREAL	长实型值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_LREAL 运算符 ST 语言示例

(* 与之等效的 ST : *)

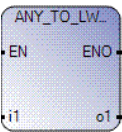
```
bres := ANY_TO_LREAL (true);           (* bres 为 1.0 *)
tres := ANY_TO_LREAL (t#1s46ms);       (* tres 为 1046.0 *)
ares := ANY_TO_LREAL (198);             (* ares 为 198.0 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_LWORD

ANY_TO_LWORD 将值转换为 64 位长字值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 64 位长字计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - REAL - LREAL - TIME - DATE - STRING	除长字之外的任何值。
o1	输出	LWORD	64 位长字值。

参数	参数类型	数据类型	说明
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_LWORD 运算符 ST 语言示例

(* 与之等效的 ST : *)

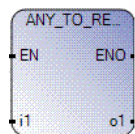
```
bres := ANY_TO_LWORD (true);           (* bres 为 1 *)  
tres := ANY_TO_LWORD (t#0s46ms);       (* tres 为 46 *)  
mres := ANY_TO_LWORD ('0198');         (* mres 为 198 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_REAL

ANY_TO_REAL 将值转换为实型值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为实型计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。

参数	参数类型	数据类型	说明
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - LREAL - TIME - DATE - STRING	除实型之外的任何值。
o1	输出	REAL	实型值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_REAL 运算符 ST 语言示例

(* 与之等效的 ST : *)

```
bres := ANY_TO_REAL (true);           (* bres 为 1.0 *)
tres := ANY_TO_REAL (t#1s46ms);       (* tres 为 1046.0 *)
ares := ANY_TO_REAL (198);             (* ares 为 198.0 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_SINT

ANY_TO_SINT 将值转换为短整型值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 8 位短整型计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	任何非短整型值。
o1	输出	SINT	短整型值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_SINT 运算符 ST 语言示例

(* 与之等效的 ST : *)

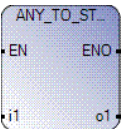
```
bres := ANY_TO_SINT (true);           (* bres 为 1 *)
tres := ANY_TO_SINT (t#0s46ms);       (* tres 为 46 *)
mres := ANY_TO_SINT ('0198');          (* mres 为 198 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_STRING

ANY_TO_STRING 将值转换为字符串值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	C 函数启用。 当 Enable = TRUE 时，执行转换为字符串计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE	除字符串之外的任何值。
o1	输出	STRING	如果 IN 是布尔值，则为 FALSE 或 TRUE。 如果 IN 是整型或实型数据，则为小数点表示法。 如果 IN 为 TIME，则可以是： TIME time1 STRING s1 time1 :=13 ms; s1:=ANY_TO_STRING(time1); (* s1 = '0s13' *)。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_STRING 运算符 ST 语言示例

(* 与之等效的 ST : *)

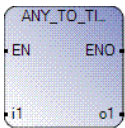
```
bres := ANY_TO_STRING (TRUE);      (* bres 为 TRUE *)  
ares := ANY_TO_STRING (125);      (* ares 为 125 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_TIME

ANY_TO_TIME 将非时间或非日期值转换为时间值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为时间计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - STRING	除时间或日期数据类型之外的任何正值。 IN (如果为实数，则为 IN 的整数部分) 为毫秒数。 STRING(毫秒数，例如值 300032 表示 5 分 32 毫 秒)。
o1	输出	TIME	IN 表示的时间值。值 1193h2m47s295ms 表示无 效时间。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_TIME 运算符 ST 语言示例

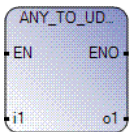
(* 与之等效的 ST : *)

```
ares := ANY_TO_TIME (1256);           (* ares := t#1s256ms *)  
rres := ANY_TO_TIME (1256.3);         (* rres := t#1s256ms *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_UDINT ANY_TO_UDINT 将值转换为 32 位无符号双整型值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 32 位双整型计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	除无符号双整型之外的任何值。
o1	输出	UDINT	32 位无符号双整型值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_UDINT 运算符 ST 语言示例

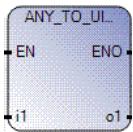
(* 与之等效的 ST : *)

```
bres := ANY_TO_UDINT (true);           (* bres 为 1 *)
tres := ANY_TO_UDINT (t#1s46ms);       (* tres 为 1046 *)
mres := ANY_TO_UDINT ('0198');         (* mres 为 198 *)
```


另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_UINT ANY_TO_UINT 将值转换为无符号整型值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 16 位无符号整型计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	任何非无符号整型值。
o1	输出	UINT	无符号整型值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_UINT 运算符 ST 语言示例

(* 与之等效的 ST : *)

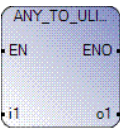
```
bres := ANY_TO_UINT (true);           (* bres 为 1 *)
tres := ANY_TO_UINT (t#0s46ms);       (* tres 为 46 *)
mres := ANY_TO_UINT ('0198');         (* mres 为 198 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_ULINT

ANY_TO_ULINT 将值转换为 64 位无符号长整型值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 64 位无符号长整型计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - LWORD - REAL - LREAL - TIME - DATE - STRING	除无符号长整型之外的任何值。
o1	输出	ULINT	64 位无符号长整型值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_ULINT 运算符 ST 语言示例

(* 与之等效的 ST : *)

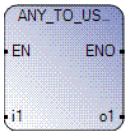
```
bres := ANY_TO_ULINT (true);           (* bres 为 1 *)
tres := ANY_TO_ULINT (t#0s46ms);       (* tres 为 46 *)
mres := ANY_TO_ULINT ('0198');         (* mres 为 198 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_USINT

ANY_TO_USINT 将值转换为无符号短整型值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 8 位无符号短整型计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - BYTE - INT - UINT - WORD - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	任何非短整型值。
o1	输出	USINT	无符号短整型值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_USINT 运算符 ST 语言示例

(* 与之等效的 ST : *)

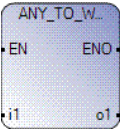
```
bres := ANY_TO_USINT (true);           (* bres 为 1 *)
tres := ANY_TO_USINT (t#0s46ms);       (* tres 为 46 *)
mres := ANY_TO_USINT ('0198');         (* mres 为 198 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

ANY_TO_WORD

ANY_TO_WORD 将值转换为 16 位字值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行转换为 16 位字计算。 当 Enable = FALSE 时，不执行计算。 仅适用于 LD 编程。
i1	输入	BOOL - SINT - USINT - BYTE - INT - - DINT - UDINT - DWORD - LINT - ULINT - LWORD - REAL - LREAL - TIME - DATE - STRING	任何非无符号整型值。
o1	输出	WORD	无符号整型值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

ANY_TO_WORD 运算符 ST 语言示例

(* 与之等效的 ST : *)

```
bres := ANY_TO_WORD (true);           (* bres 为 1 *)  
tres := ANY_TO_WORD (t#0s46ms);       (* tres 为 46 *)  
mres := ANY_TO_WORD ('0198');         (* mres 为 198 *)
```

另请参见

[数据转换指令](#) 参考页数 [293](#)

数据操作指令

数据操作指令用于在不修改程序的情况下修改输出数据以更改状态。

功能块	说明
AVERAGE 参考页数 315	在 N 个样本上运行平均值。
COP 参考页数 318	将源 (Src) 中的二进制数据复制到目标 (Dest)。
函数	说明
MAX 参考页数 324	最大
MIN 参考页数 322	最小

AVERAGE

AVERAGE 在每次循环时存储值，并计算所有已存储值的平均值。仅存储最后 N 个值。



AVERAGE 功能块操作

- 示例数 (N) 不能超过 128。
- 如果 RUN 命令为 FALSE (重置模式)，则输出值等于输入值。
- 如果已达到存储值的最大数量，则最后一个存储值将擦除首个存储值。

参数

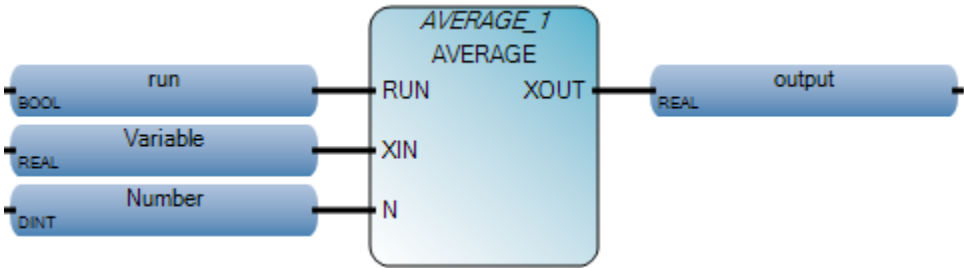
参数	参数类型	数据类型	说明
RUN	输入	BOOL	TRUE = 运行/FALSE = 重置。
XIN	输入	REAL	任何实型变量。
N	输入	DINT	应用程序定义的示例数量。
XOUT	输出	REAL	XIN 值的运行平均值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

提示：使用浮点型数据可能会导致计算不准确，这归因于浮点运算中继承的舍入限制。

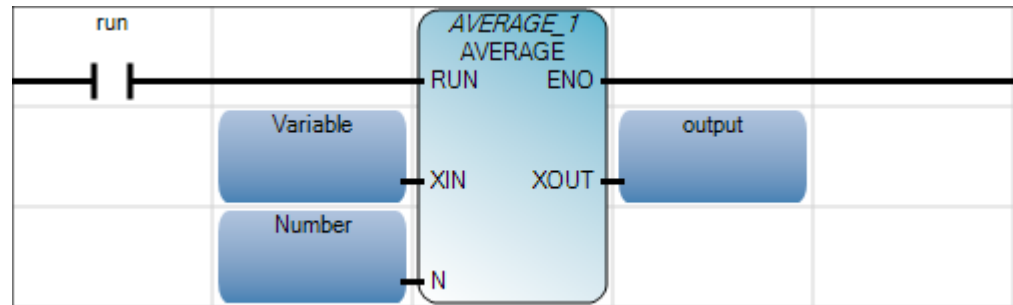
提示：设置或更改 N 的值时，您需要将 RUN 设为 FALSE，然后再将其设置回 TRUE。

AVERAGE 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1 | AVERAGE_1(run, Variable, Number);
2 | output := AVERAGE_1.XOUT;

```

```

AVERAGE_1(
void AVERAGE_1(BOOL RUN, REAL XIN, DINT N)
Type : AVERAGE, Running average over N samples

```

(* 与之等效的 ST: AVERAGE1 是 AVERAGE 块的实例 *)

```

AVERAGE1((auto_mode & store_cmd), sensor_value, 100);
ave_value := AVERAGE1.XOUT;

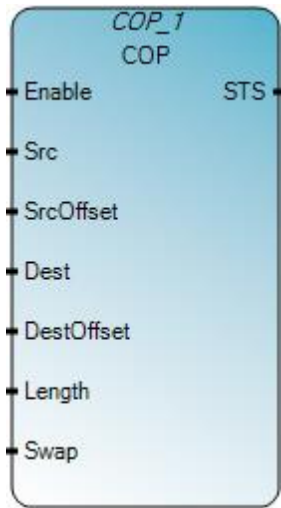
```

另请参见

[数据操作指令](#) 参考页数 [315](#)

COP

COP 将源中的二进制数据复制到目标，并保持源值不变。



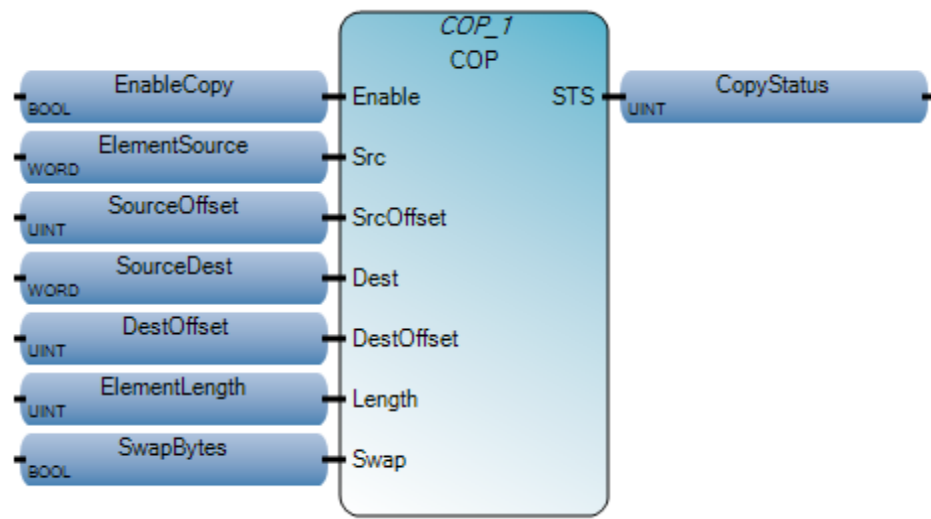
参数

参数	参数类型	数据类型		说明
启用	输入	BOOL		功能块启用。该 FB 是级别触发的。 当 Enable = TRUE 时，执行复制。 当 Enable = FALSE 时，不执行该功能块。
Src	输入	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT	DWORD REAL TIME DATE STRING LWORD ULINT LINT LREAL	要复制的初始元素。 如果源或目标是字符串数据类型，则另一方必须同样是字符串数据类型或者 USINT（UCHAR 和 BYTE）数据类型。否则，将报告数据类型不匹配。 另请参见 复制到不同的数据类型 参考页数 322。
SrcOffset	输入	UINT		源元素偏移。 如果源是数组数据类型，则元素会偏移。 在以下情况中，将偏移设为 0： • 如果不是数组数据类型，或者 • 从数组数据类型的首个元素开始复制。

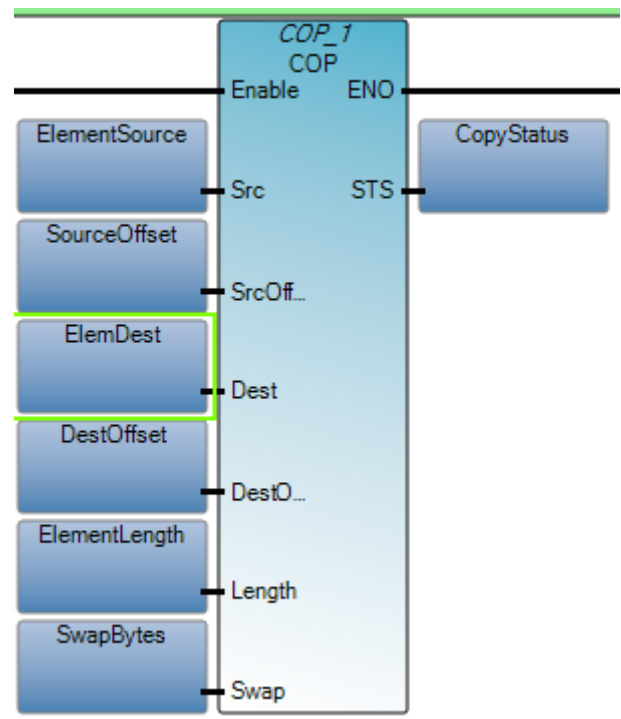
参数	参数类型	数据类型		说明
Dest	输入	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT	DWORD REAL TIME DATE STRING LWORD ULINT LINT	要被源覆盖的初始元素。
DestOffset	输入	UINT		目标元素偏移。 如果目标是数组数据类型，则元素会偏移。 在以下情况中，将偏移设为 0： • 如果不是数组数据类型，或者 • 从数组数据类型的首个元素开始复制。
Length	输入	UINT		要复制的目标元素个数。 如果目标是字符串数据类型，则它表示要复制的字符串数量。
Swap	输入	BOOL		TRUE：根据数据类型交换字节。 在以下情况中，不会执行交换操作： • 源数据类型或目标数据类型是字符串，或者 • 源和目标都是长度为 1 个字节的数据。
Sts	输出	UINT		复制操作的状态。 请参见 COP 操作状态值 参考页数 321
ENO	输出	BOOL		启用输出。 仅适用于 LD 编程。

COP 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 COP_1(EnableCopy, ElementSource, SourceOffset, SourceDest,  
2   DestOffset, ElementLength, SwapBytes);  
3 output :=COP_1.STS;
```

另请参见

[数据操作指令](#) 参考页数 [315](#)

COP 操作状态值

下表描述了 COP 操作状态值：

COP 状态值	状态说明
0x00	未采取行动（未启用）。
0x01	COP 功能块成功。
0x02	在从字符串复制时，目标中含有空余字节。
0x03	源数据被截断。
0x04	副本长度无效。
0x05	当源或目标为字符串数据类型时，数据类型不匹配。
0x06	源数据太小而无法复制。
0x07	目标数据太小而无法复制。
0x08	源数据偏移无效。
0x09	目标数据偏移无效。
0x0A	源或目标中的数据无效。

另请参见

[COP](#) 参考页数 [318](#)

复制到不同的数据类型

如果执行至/自字符串数据类型的复制，则将对 USINT 数组中的数据使用 ODVA 短字符串格式。

如果在任何其他数据类型对之间使用 COP，则即使源中的数据类型与目标中的数据类型不同，并且它们的格式无效，复制操作也有效。必须在应用程序级别验证该逻辑。

从 USINT 数组到字符串数组

要将 USINT 数组复制到字符串数组，USINT 数组中的数据必须采用以下格式：

- 字节 1：第一个字符串长度
- 字节 2：第一个字节字符
- 字节 3:第二个字节字符
- 字节 n：最后一个字节字符
- 字节 (n+1)：第二个字符串长度
- 字节 (n+2)：第二个字符串的第一个字节字符
- 等等

另请参见

[COP](#) 参考页数 [318](#)

MIN

MIN 生成两个整型值的最小值。



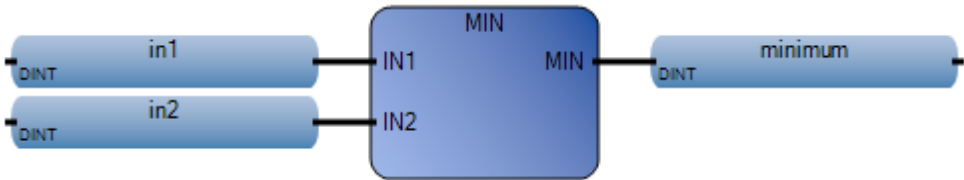
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行最小整型值计算 当 EN = FALSE 时，不执行计算。
IN1	输入	DINT	任何有符号整型值。
IN2	输入	DINT	不能为实型。
MIN	输出	DINT	两个输入值的最小值。

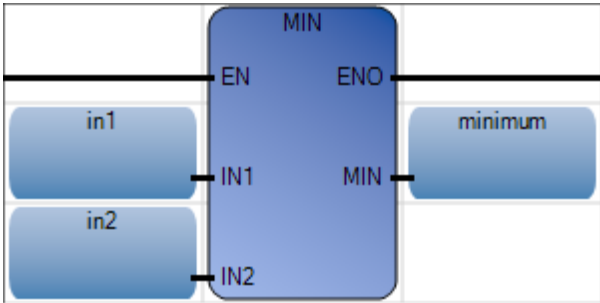
参数	参数类型	数据类型	说明
ENO	输出	BOOL	启用输出。

MIN 函数语言示例

功能块图

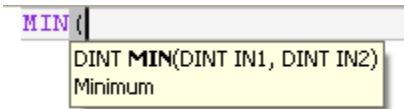


梯形图



结构化文本

```
1 | in1 := 3;  
2 | in2 := 10;  
3 | minimum := MIN(in1, in2);
```



(* 与之等效的 ST : *)

```
new_value := MAX (MIN (max_value, value), min_value);
```

(* 将值限制在 [min_value..max_value] 集合内 *)

结果



另请参见

[数据操作指令](#) 参考页数 [315](#)

MAX

MAX 生成两个整型值的最大值。

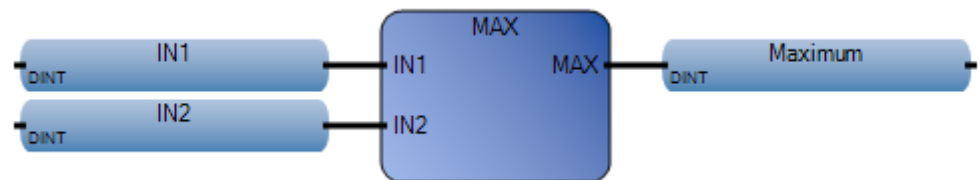


参数

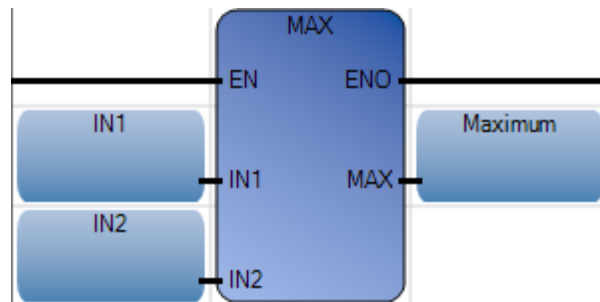
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行最大整型值计算。 当 EN = FALSE 时，不执行计算。
IN1	输入	DINT	任何有符号整型值。
IN2	输入	DINT	不能为实型。
MAX	输出	DINT	两个输入值的最大值。
ENO	输出	BOOL	启用输出。

MAX 函数语言示例

功能块图



梯形图



结构化文本

```

1  IN1 := 3;
2  IN2 := 10;
3  Maximum := MAX(IN1, IN2);

```

```

MAX(
  DINT MAX(DINT IN1, DINT IN2)
  Maximum
)

```

(* 与之等效的 ST : *)

```
new_value := MAX (MIN (max_value, value), min_value);
```

(* 将值限制在 [min_value..max_value] 集合内 *)

结果



另请参见

[数据操作指令](#) 参考页数 [315](#)

高速计数器 (HSC) 指令

高速计数器指令用于监视和控制高速计数器。

功能块	说明
HSC 参考页数 328	HSC 将高预设、低预设和输出源值应用到高速计数器。
HSC SET STS 参考页数 351	HSC_SET_STS 手动设置或重置 HSC 计数状态。

什么是高速计数器？

高速计数器会检测窄幅（快速）脉冲并进行计数，然后发布专门指令，
以在检测的计数达到其预设值时启动控制操作。控制操作包括根据配置的
源和掩码模式，自动即时执行高速计数器中断例程并即时更新输出。

高速计数器功能

由于 HSC 指令具有高性能要求，因此将由与主系统处理器并行运行的自定义电路来执行其操作。高速计数器的增强功能包括：

- 100 kHz 运行的高速输出直接控制
- 32 位带符号整型数据（计数范围为 $\pm 2,147,483,647$ ）
- 可编程的高预设与低预设
- 上溢和下溢设定点
- 基于累加计数的自动中断处理
- 运行时可编辑参数（来自用户控制程序）HSC 指令操作

Micro800 控制器对 HSC 的支持

除了 2080-LCxx-AWB，所有 Micro830 和 Micro850 控制器都最多支持六个 HSC 输入。Micro800 控制器中的 HSC 功能是通过使用高速计数器硬件（控制器中的嵌入式输入）以及应用程序中的 HSC 指令来实现的。HSC 指令用于配置高速计数器硬件和更新图像累加器。

重要说明： HSC 功能仅可与控制器的嵌入式 I/O 配合使用。它不能与扩展 I/O 模块配合使用。

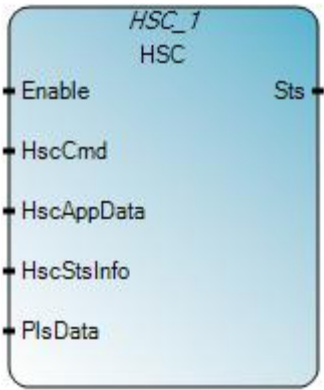
另请参见

[使用高速计数器指令](#) 参考页数 [353](#)

[高速计数器 \(HSC\) 指令](#) 参考页数 [327](#)

HSC

HSC 将高预设、低预设和输出源值应用到高速计数器。



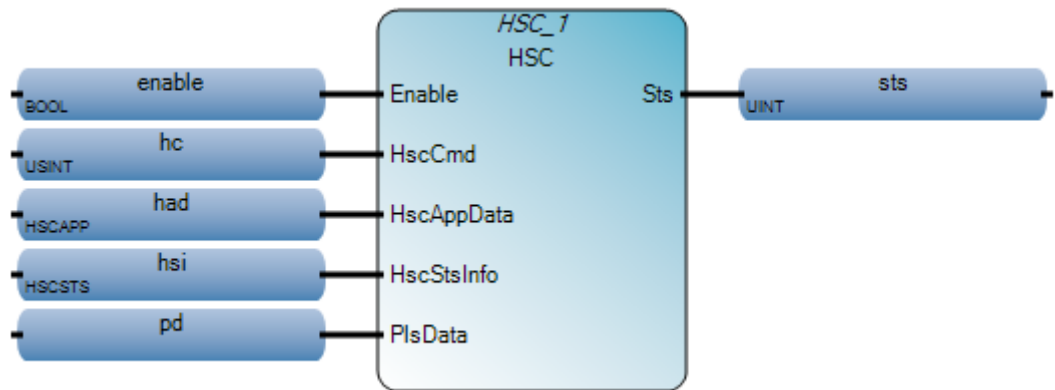
参数

参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行 HSC 命令参数中指定的 HSC 操作。 当 Enable = FALSE 时，不发布任何 HSC 命令。

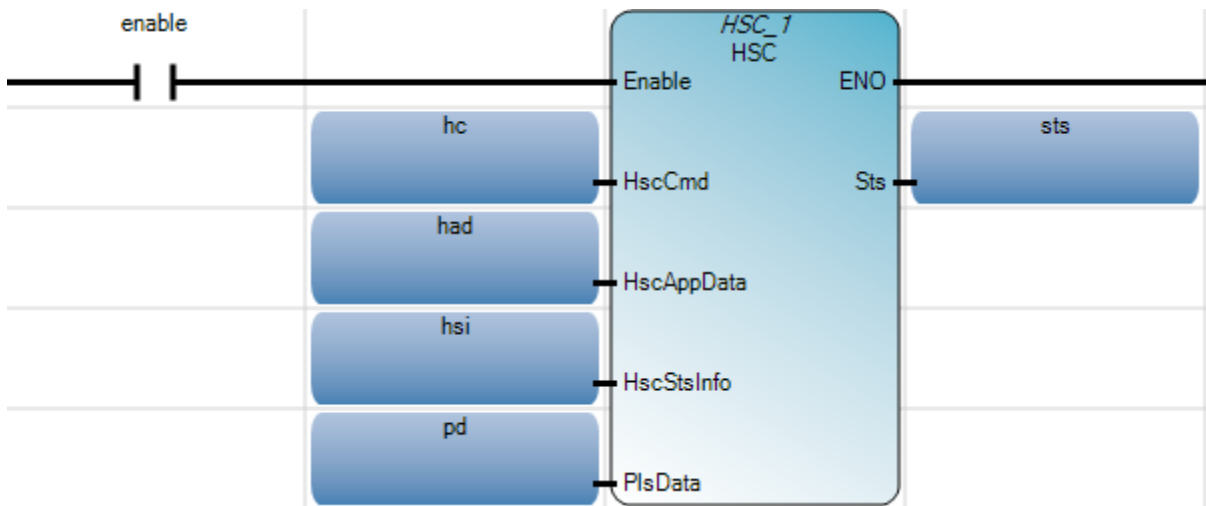
参数	参数类型	数据类型	说明
HscCmd	输入	USINT	向 HSC 发布命令。 请参见 HSCCmd 值 参考页数 330。
HSCAppData	输入	HSCAPP	HSC 应用程序配置 (通常仅需一次)。 请参见 HSC 应用数据类型 参考页数 332。
HSCStsInfo	输入	HSCSTS	HSC 动态状态, 在 HSC 计数期间不断更新。 请参见 HSCSTS 数据类型 参考页数 339。
PlsData	输入	DINT UDINT	可编程限位开关 (PLS) 数据结构。 请参见 PLS 数据类型 参考页数 348
Sts	输出	UINT	HSC 执行状态。 请参见 HSC 状态代码 (STS) 参考页数 350。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

HSC 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 HSC_1(enable, hc, had, hsi, pd);
2 sts := HSC_1.Sts;
```

```
HSC_1 {
void HSC_1(BOOL Enable, USINT HscCmd, HSCAPP HscAppData, HSCSTS HscStsInfo, PLS[1..1] PlsData, UINT __ADI_PlsData)
Type : HSC, Apply high/low presets and output source to high-speed counter.
```

另请参见

- [使用高速计数器指令](#) 参考页数 353
- [高速计数器 \(HSC\) 指令](#) 参考页数 327

HSCCmd 值

下表描述了每个 HSC 命令值的 HSC 命令。

HSC 命令	命令描述
0x01	HSC 运行 • 启动 HSC (如果 HSC 处于闲置模式 , 且已启用 rung) 。 • 仅更新 HSC 状态信息 (如果 HSC 处于运行模式 , 且已启用 rung) 。
0x02	HSC 停止 : 停止 HSC 计数 (如果 HSC 处于运行模式 , 且已启用 rung) 。
0x03	HSC 加载/设置 : 重新加载 6 个输入元素的 HSC 应用程序数据 (如果已启用 rung) : HPSetting、LPSetting、HPOutput、LPOutput、OFSetting 和 UFSetting。 注意: 此命令不会重新加载以下输入元素 : HSC 累加器。
0x04	HSC 累加器重置 (如果已启用 rung) 。

HSC 命令结果

下表描述了在不同情况下发出 HSC 命令的结果。

命令值	结果	条件
HscCmd =1	启动 HSC 机制 , 并且 HSC 会过渡到运行模式。	在运行模式中将 Enable 输入参数设置为 FALSE 无法停止计数。 必须发出 HscCmd =2 才能停止计数。
	HSC 机制会自动更新值。	HSC AppData.Accumalator 与 HSC Sts.Accumulator 一起更新
HscCmd =4 (重置)	将 HSC Acc 的值设为 HSC AppData.Accumalator 的值。	HscCmd =4 不会停止 HSC 计数。 如果发布 HscCmd =4 后 HSC 正在计数 , 则某些计数可能会丢失。
		要在计数时将 HSC Acc 设为特定的值 , 请在发布 HscCmd =4 前将值立即写入 HSC AppData.Accumalator。

另请参见

[HSC](#) 参考页数 [328](#)

HSCAPP 数据类型

HSCAppData (数据类型 HSCAPP) 用于配置 HSC 应用程序。

HSCAppData 参数

下表列出了 HSCAppData 参数。

参数	数据类型	数据格式	用户程序访问	说明
PLSEnable	BOOL	位	读取/写入	启用或禁用可编程限位开关 (PLS)。
HSCID	UINT	字	读取/写入	定义 HSC。
HSCMode	UINT	字	读取/写入	定义 HSC 模式。
Accumulator	DINT	长字	读取/写入	累加器初始值。
HPSetting	DINT	长字	读取/写入	高预设设置。
LPSetting	DINT	长字	读取/写入	低预设设置。
OFSetting	DINT	长字	读取/写入	溢出设置。
UFSetting	DINT	长字	读取/写入	下溢设置。
OutputMask	UDINT	字	读取/写入	输出掩码。
HPOutput	UDINT	长字	读取/写入	达到高预设 32 位输出设置。
LPOutput	UDINT	长字	读取/写入	达到低预设 32 位输出设置。

HSCAppData 参数详细信息

HSCApp 数据类型参数用于定义 HSC 配置数据。

PLSEnable

参数	数据类型	数据格式	用户程序访问
HSCApp.PLSEnable	BOOL	位	读取/写入

启用和禁用高速计数器可编程限位开关 (PLS) 功能。

HSCApp 设置与 PLSData 设置的比较

已启用 PLS 功能时，相应的 PLSData 设置会挂起相关的 HSCApp 设置，如下表所示。

HSCApp 设置	PLSData 设置
HSCAPP.HpSetting	HSCHP
HSCAPP.LpSetting	HSCLP
HSCAPP.HPOutput	HSCHPOutput
HSCAPP.LPOutput	HSCLPOutput

HSCID

参数	数据类型	数据格式	用户程序访问
HSCApp.HSCID	UINT	字	读取/写入

标识要使用的高速计数器。下表列出了用于定义 HSC ID 的值：

输出选择	位	说明
HSC 功能数据的第一个字	15-13	HSC 的模块类型： • 0x00 - 嵌入式。 • 0x01 - 扩展。 • 0x02 - 插件端口。
	12-8	模块的插槽 ID： • 0x00 - 嵌入式。 • 0x01-0x1F - 扩展模块的 ID。 • 0x01-0x05 - 插件端口的 ID。
	7-0	模块中的 HSC ID： • 0x00-0x0F - 嵌入式。 • 0x00-0x07 - HSC 的扩展 ID。 • 0x00-0x07 - HSC 的插件端口 ID。 注意：对于初始版 Connected Components Workbench，仅支持 ID 0x00-0x05。

HSCMode

参数	数据类型	用户程序访问
HSCApp.HSCMode	UINT	读取/写入

将高速计数器设为 10 种计数模式类型之一。模式值可通过编程设备配置，并可在控制程序中访问。

HSC 操作模式

主 HSC 和子 HSC 支持不同模式。

- 主高速计数器支持 10 种操作模式。
- 子高速计数器支持 5 种操作模式（模式 0、2、4、6、8）。
- 如果主高速计数器设为模式 1、3、5、7 或 9，则将禁用子高速计数器。

HSCMode	计数模式
0	增序计数器。累加器会在其达到高预设时立即清零 (0)。此模式下不能定义低预设。
1	带有外部重置和保存功能的增序计数器。累加器会在其达到高预设时立即清零 (0)。此模式下不能定义低预设。
2	采用外部方向的计数器。
3	采用外部方向并具有重置和保存功能的计数器。
4	双输入计数器（向上和向下）。
5	具有外部重置和保存功能的双输入计数器（向上和向下）。
6	正交计数器（带相位输入 A 和 B）。
7	具有外部重置和保存功能的正交计数器（带相位输入 A 和 B）。
8	正交 X4 计数器（带相位输入 A 和 B）。
9	具有外部重置和保存功能的正交 X4 计数器（带相位输入 A 和 B）。

有关 HSC 操作模式和输入分配的附加信息，请参见 *Micro830 和 Micro850 可编程控制器用户手册* 中的 HSC 输入和配线图。

如何访问控制器用户手册？

1. 在帮助菜单中,单击用户手册,展开控制器,然后展开控制器类别。
2. 双击控制器类别中的控制器用户手册以打开手册 PDF。

Accumulator

参数	数据类型	用户程序访问
HSCApp.Accumulator	DINT	读取/写入

当高速计数器启动时,设置累加器初始值。当 HSC 处于计数模式时,HSC 子系统会自动更新累加器以反映 HSC 累加器实际值。

HPSetting

参数	数据类型	用户程序访问
HSCApp.HPSetting	DINT	读取/写入

用于定义 HSC 子系统何时生成中断的上设定点(以计数为单位)。

载入高预设的数据必须少于或等于驻留在上溢(HSCAPP.OFSetting)参数中的数据,否则将生成 HSC 错误。

LPSetting

参数	数据类型	用户程序访问
HSCApp.LPSetting	DINT	读取/写入

用于定义 HSC 子系统何时生成中断的下设定点(以计数为单位)。

- 载入低预设的数据必须大于或等于驻留在下溢(HSCAPP.UFSetting)参数中的数据,否则将生成 HSC 错误。
- 如果下溢和低预设值为负数,则低预设值必须是绝对值小于下溢值的数字。

OFSetting

参数	数据类型	用户程序访问
HSCApp.OFSetting	DINT	读取/写入

用于定义计数器计数上限的上溢设置。

- 如果计数器的累加值增加至高于 UFSetting 中指定的值 ,则将生成上溢中断。
- 生成上溢中断时 , HSC 子系统会将累加值重置为下溢值 , 计数器将从下溢值开始计数 (在此过渡期间不会丢失计数) 。

OFSetting 值必须是 :

- 介于 -2,147,483,648 和 2,147,483,647 之间。
- 大于下溢值。
- 大于或等于驻留在高预设 (HSCAPP.HPSetting) 中的数据 , 否则将生成 HSC 错误。

UFSetting

参数	数据类型	用户程序访问
HSCApp.UFSetting	DINT	读取/写入

用于定义计数器计数下限的下溢设置。

- 如果计数器的累加值减少至低于 UFSetting 中指定的值 ,则将生成下溢中断。
- 生成下溢中断时 , HSC 子系统会将累加值重置为上溢值 , 计数器将从上溢值开始计数 (在此过渡期间不会丢失计数) 。

UFSetting 值必须是 :

- 介于 -2,147,483,648 和 2,147,483,647 之间。
- 小于上溢值。
- 小于或等于驻留在低预设 (HSCAPP.LPSetting) 中的数据 , 否则将生成 HSC 错误。

OutputMask

参数	数据类型	用户程序访问
HSCApp.OutputMask	UDINT	读取/写入

定义控制器上高速计数器可以直接控制的嵌入式输出。无需与控制程序交互 ,HSC 子系统就可以根据 HSC 累加器的高或低预设开启 (ON) 或关闭 (OFF) 输出。

OutputMask 位模式

HSCApp.OutputMask 中存储的位模式定义 HSC 控制哪些输出以及 HSC 不控制哪些输出。

HSCAPP.OutputMask 位模式与控制器中的输出位相对应 , 并且仅能在初始设置期间配置。

- 设置为 (1) 的位已启用 , 并可以由 HSC 子系统来开启或关闭。
- 设置为 (0) 的位不能由 HSC 子系统开启或关闭。

例如 , 要使用 HSC 来控制输出 0、1、3 , 请按如下方式赋值 :

- HscAppData.OutputMask = 2#1011 , 或
- HscAppData.OutputMask = 11

HPOutput

参数	数据类型	用户程序访问
HSCApp.HPOutput	UDINT	读取/写入

定义达到高预设时控制器上输出的状态 (1 = ON 或 0 = OFF)。有关如何根据高预设直接开启或关闭输出的更多信息 , 请参见 OutputMask。

您可以在初始设置期间配置高输出位模式 , 或在控制器操作期间使用 HSC 功能块加载新参数。

LPOutput

参数	数据类型	用户程序访问
HSCApp.LPOutput	UDINT	读取/写入

LPOutput (HSCApp.LPOutput) 定义达到低预设时控制器上输出的状态 (1 = "on" , 0 = "off ") 。有关如何根据低预设直接开启或关闭输出的更多信息，请参见 OutputMask。

您可以在初始设置期间配置低输出位模式，或在控制器操作期间使用 HSC 功能块加载新参数。

HSCAppData 参数示例

下图展示了变量选择器中的 HSCAppData 参数。

	名称	数据类型	维度	别名	初始值	
	HSC_1.HscAppData	HSCAPP		HscA	...	读/写
	HSC_1.HscA	BOOL				读/写
	HSC_1.HscB	UINT				读/写
	HSC_1.HscC	UINT				读/写
	HSC_1.HscD	DINT				读/写
	HSC_1.HscE	DINT				读/写
	HSC_1.HscF	DINT				读/写
	HSC_1.HscG	DINT				读/写
	HSC_1.HscH	DINT				读/写
	HSC_1.HscI	UDINT				读/写
	HSC_1.HscJ	UDINT				读/写
	HSC_1.HscK	UDINT				读/写
	HSC_1.HscStsInfo	HSCSTS		HscS	...	读/写
	HSC_1.PlsData			PlsD	...	读/写
	HSC_1.__ADI_PlsI	UINT		_ADI	0	读/写
	HSC_1.Sts	UINT		Sts		读/写
	*					

另请参见

[HSC](#) 参考页数 [328](#)

HSCSTS 数据类型

HSCSTSInfo (数据类型 HSCSTS) 显示高速计数器的状态。

HSCSTSInfo 状态操作

在 HSC 计数期间，将发生以下 HSC 状态操作。

- 如果 HSC 功能块使用命令 0x01 进行计数，则 HSC 状态会持续更新。
- 如果发生错误，则将开启 Error_Detected 标记并设置错误代码。

HSCSTSInfo 参数

下表描述了 HSCSTSInfo 参数。

参数	数据类型	HSC 模式	用户程序访问	说明
CountEnable	BOOL	0...9	只读	已启用计数。
ErrorDetected	BOOL	0...9	读取/写入	非零意味着检测到错误。
CountUpFlag	BOOL	0...9	只读	向上计数标志。
CountDwnFlag	BOOL	2...9	只读	向下计数标志。
Mode1Done	BOOL	0 或 1	读取/写入	HSC 是模式 1A 或模式 1B；累加器向上计数到 HP 值。
OVF	BOOL	0...9	读取/写入	检测到上溢。
UNF	BOOL	0...9	读取/写入	检测到下溢。
CountDir	BOOL	0...9	只读	1：向上计数；0：向下计数。
HPReached	BOOL	2...9	读取/写入	达到高预设。
LPReached	BOOL	2...9	只读	达到低预设。
OFCauseInter	BOOL	0...9	读取/写入	上溢造成 HSC 中断。

参数	数据类型	HSC 模式	用户程序访问	说明
CountEnable	BOOL	0...9	只读	已启用计数。
UFCauseInter	BOOL	2...9	读取/写入	下溢造成 HSC 中断。
HPCauseInter	BOOL	0...9	读取/写入	达到高预设，从而造成 HSC 中断。
LPCauseInter	BOOL	2...9	读取/写入	达到低预设，从而造成 HSC 中断。
PlsPosition	UINT	0...9	只读	可编程限位开关 (PLS) 的位置。
ErrorCode	UINT	0...9	读取/写入	显示 HSC 子系统检测到的错误代码。
Accumulator	DINT		读取/写入	实际累加器读取。
HP	DINT		只读	最终高预设设置。
LP	DINT		只读	最终低预设设置。
HPOutput	UDINT		读取/写入	最终高预设输出设置。
LPOutput	UDINT		读取/写入	最终低预设输出设置。

HSCSTSInfo 参数详细信息

HSCSTSInfo (数据类型 HSCSTS) 参数用于确定高速计数器的状态。

CountEnable

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.CountEnable	BOOL	0...9	只读

指示高速计数器的状态，即表示计数是已启用 (1) 还是已禁用 (0，默认值)。

ErrorDetected

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.ErrorDetected	BOOL	0...9	读取/写入

检测 HSC 子系统中是否存在错误。ErrorDetectedr 所指示的错误类型中最常见的是配置错误。当该位设为 (1) 时，查看参数 HSCSTS.ErrorCode (由控制器维护) 中的特定错误代码。您可以在必要时清除位。

CountUpFlag

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.CountUpFlag	BOOL	0...9	只读

与所有 HSC (模式 0...9) 配合使用。如果已设置 HSCSTS.CountEnable 位 , Count Up 位会设为 (1)。如果已清除 HSCSTS.CountEnable 位 , Count Up 位会清零 (0)。

CountDownFlag

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.CountDownFlag	BOOL	2...9	只读

与双向计数器(模式 2...9) 配合使用。如果已设置 HSCSTS.CountEnable 位 , Count Down 位会设为 (1)。如果已清除 HSCSTS.CountEnable 位 , Count Down 位会清零 (0)。

Mode1Done

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.Mode1Done	BOOL	0 或 1	读取/写入

如果 HSC 已配置为模式 0 或模式 1 行为 , 并且累加器向上计数到高预设值 , 则 HSC 子系统会将 HSCSTS.Mode1Done 状态标记设为 (1)。

OVF

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.OVF	BOOL	0...9	读取/写入

只要累积值 (HSCSTS.Accumulator) 计数已越过上溢变量 (HSCAPP.OFSetting) 值 , HSC 子系统就会将 HSCSTS.OVF 状态标记设为 (1)。该位是过渡位并由 HSC 子系统设置。上溢条件则由控制程序来使用、跟踪和清除 (0)。

上溢条件不会生成控制器故障。

UNF

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.UNF	BOOL	0...9	读取/写入

只要累积值 (HSCSTS.Accumulator) 计数已越过下溢变量 (HSCAPP.UFSetting) 值，HSC 子系统就会将 HSCSTS.UNF 状态标记设为 (1)。该位是过渡位并由 HSC 子系统设置。下溢条件则由控制程序来使用、跟踪和清除 (0)。

下溢条件不会生成控制器故障。

CountDir

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.CountDir	BOOL	0...9	只读

HSC 子系统将控制 HSCSTS.CountDir 状态标记。当 HSC 累加器向上计数时，方向标记设为 (1)。当 HSC 累加器向下计数时，方向标记清零 (0)。如果累加值停止计数，则方向位会保留其值。方向标记发生改变的唯一情况是累加计数反向进行。

只要控制器处于运行模式，HSC 子系统就会持续更新此位。

HPReached

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.HPReached	BOOL	2...9	读取/写入

只要累积值 (HSCSTS.Accumulator) 大于或等于高预设变量 (HSCAPP.HPSetting) 值，HSC 子系统就会将 HSCSTS.HPReached 状态标记设为 (1)。

只要控制器处于执行模式，HSC 子系统就会持续更新此位。不建议写入此元素。

LPReached

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.LPReached	BOOL	2...9	只读

只要累积值 (HSCSTS.Accumulator) 小于或等于低预设变量 (HSCAPP.LPSetting) 值, HSC 子系统就会将 HSCSTS.LPReached 状态标记设为 (1)。

只要控制器处于执行模式, HSC 子系统就会持续更新此位。不建议写入此元素。

OFCauseInter

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.OFCauseInter	BOOL	0...9	读取/写入

当 HSC 累加器越过上溢值计数并且已触发 HSC 中断时, Overflow Interrupt 状态位会设为 (1)。可在控制程序中使用该位来标识上溢变量导致 HSC 中断。如果控制程序需要执行任何基于上溢的特定控制操作, 该位可用作条件逻辑。

控制程序可以清除该位 (0), HSC 子系统也会在检测到这些条件时清除该位:

- 低预设中断执行
- 高预设中断执行
- 下溢中断执行

UFCauseInter

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.UFCauseInter	BOOL	2...9	读取/写入

当 HSC 累加器越过下溢值计数并且已触发 HSC 中断时, Underflow Interrupt 状态位会设为 (1)。可在控制程序中使用该位来标识下溢条件

致 HSC 中断。如果控制程序需要执行任何基于下溢的特定控制操作，该位可用作条件逻辑。

控制程序可以清除该位 (0)，HSC 子系统也会在检测到这些条件时清除该位：

- 发生低预设中断
- 发生高预设中断
- 发生上溢中断

HPCauseInter

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.HPCauseInter	BOOL	0...9	读取/写入

当 HSC 累加器达到高预设值并且已触发 HSC 中断时，High Preset Interrupt 状态位会设为 (1)。可在控制程序中使用该位来标识高预设条件导致 HSC 中断。如果控制程序需要执行任何基于高预设的特定控制操作，则该位可用作条件逻辑。

控制程序可以清除该位 (0)，HSC 子系统也会在检测到这些条件时清除该位：

- 发生低预设中断
- 发生下溢中断
- 发生上溢中断

LPCauseInter

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.LPCauseInter	BOOL	2...9	读取/写入

当 HSC 累加器达到低预设值并且已触发 HSC 中断时，Low Preset Interrupt 状态位会设为 (1)。可在控制程序中使用该位来标识低预设条件导致 HSC 中断。如果控制程序需要执行任何基于低预设的特定控制操作，则该位可用作条件逻辑。

控制程序可以清除该位 (0)，HSC 子系统也会在检测到这些条件时清除该位：

- 发生高预设中断
- 发生下溢中断
- 发生上溢中断

PlsPosition

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.PLSPosition	UINT	0...9	只读

如果 HSC 处于计数模式，并且已启用 PLS，则此参数指示用于当前 HSC 配置的具体 PLS 元素。

ErrorCode

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.ErrorCode	BOOL	0...9	只读

显示 HSC 子系统检测到的错误代码。

错误代码子元素	HSC 计数错误代码	错误描述
15-8 位 (高字节)	0-255	高字节值非零表明由于 PLS 数据设置导致 HSC 错误。高字节的值表明触发错误的 PLS 数据的元素。
7-0 位 (低字节)	0x00	未出现错误。
	0x01	HSC 计数模式无效。
	0x02	无效的高预设。
	0x03	无效的溢出。
	0x04	无效的下溢。
	0x05	无 PLS 数据。

Accumulator

参数	数据类型	用户程序访问
HSCApp.Accumulator	DINT	读取/写入

当高速计数器启动时，设置累加器初始值。当 HSC 处于计数模式时，HSC 子系统会自动更新累加器以反映 HSC 累加器实际值。

HP

参数	数据类型	用户程序访问
HSCSTS.HP	DINT	只读

HSCSTS.HP 是用于定义 HSC 子系统何时生成中断的上设定点（以计数为单位）。

载入高预设的数据必须少于或等于驻留在上溢 (HSCAPP.OFSetting) 参数中的数据，否则将生成 HSC 错误。

这是最新的高预设设置，可由 PLS 数据块的 PLS 功能更新。

LP

参数	数据类型	HSC 模式	用户程序访问
HSCSTS.LP	DINT		只读

HSCSTS.LP 是用于定义 HSC 子系统何时生成中断的下设定点(以计数为单位)。

载入低预设的数据必须大于或等于驻留在下溢 (HSCAPP.UFSetting) 参数中的数据，否则将生成 HSC 错误。如果下溢和低预设值为负数，则低预设值必须是绝对值更小的数字。

这是最新的低预设设置，可由 PLS 数据块的 PLS 功能更新。

HPOutput

参数	数据类型	用户程序访问
HSCApp.HPOutput	UDINT	读取/写入

定义达到高预设时控制器上输出的状态 (1 = ON 或 0 = OFF)。有关如何根据高预设直接开启或关闭输出的更多信息，请参见 OutputMask。

您可以在初始设置期间配置高输出位模式，或在控制器操作期间使用 HSC 功能块加载新参数。

LPOutput

参数	数据类型	用户程序访问
HSCApp.LPOutput	UDINT	读取/写入

LPOutput (HSCApp.LPOutput) 定义达到低预设时控制器上输出的状态 (1 = "on" , 0 = "off ")。有关如何根据低预设直接开启或关闭输出的更多信息，请参见 OutputMask。

您可以在初始设置期间配置低输出位模式，或在控制器操作期间使用 HSC 功能块加载新参数。

HSCSTSInfo 参数示例

下图展示了变量选择器中的 HSCStsInfo 参数。

全局变量 - Micro850		局部变量 - HSCSampleProgram		系统变量 - Micro850	
名称	数据类型	特性	注释		
HSC_1.HscStsInfo	HSCSTS	Read/Write	HSC dynamic status.		
HSC_1.HscStsInfo.CountEnable	BOOL	Read/Write			
HSC_1.HscStsInfo.ErrorDetected	BOOL	Read/Write			
HSC_1.HscStsInfo.CountUpFlag	BOOL	Read/Write			
HSC_1.HscStsInfo.CountDwnFlag	BOOL	Read/Write			
HSC_1.HscStsInfo.Mode1Done	BOOL	Read/Write			
HSC_1.HscStsInfo.OVF	BOOL	Read/Write			
HSC_1.HscStsInfo.UNF	BOOL	Read/Write			
HSC_1.HscStsInfo.CountDir	BOOL	Read/Write			
HSC_1.HscStsInfo.HPRReached	BOOL	Read/Write			
HSC_1.HscStsInfo.LPRReached	BOOL	Read/Write			
HSC_1.HscStsInfo.OFCauseInter	BOOL	Read/Write			
HSC_1.HscStsInfo.UFCauseInter	BOOL	Read/Write			
HSC_1.HscStsInfo.HPCauseInter	BOOL	Read/Write			
HSC_1.HscStsInfo.LPCauseInter	BOOL	Read/Write			
HSC_1.HscStsInfo.PlsPosition	UINT	Read/Write			
HSC_1.HscStsInfo.ErrorCode	UINT	Read/Write			
HSC_1.HscStsInfo.Accumulator	DINT	Read/Write			
HSC_1.HscStsInfo.HP	DINT	Read/Write			
HSC_1.HscStsInfo.LP	DINT	Read/Write			
HSC_1.HscStsInfo.HPOutput	UDINT	Read/Write			
HSC_1.HscStsInfo.LPOutput	UDINT	Read/Write			
HSC_1.PlsData		Read/Write	Programmable Limit S		

另请参见

[HSC](#) 参考页数 [328](#)

PLS 数据类型

PLSData (数据类型 PLS) 用于配置可编程限位开关。

PLSData 结构元素

PLS 数据结构是具有以下元素的灵活数组。

元素	元素顺序	数据类型	元素描述
HSCHP	字 0...1	DINT	高预设

HSCLP	字 2...3	DINT	低预设
HSCHPOutput	字 4...5	UDINT	输出上限数据
HSCLPOutput	字 6...7	UDINT	输出下限数据

一个 PLS 数据结构的元素总数不应超过 255。

PLSData 参数

下表列出了 PLSData 参数详细信息。

参数	数据类型	数据格式	HSC 模式	用户程序访问	说明
HSCHP	DINT	32 位带符号整数	0	读取/写入	高预设
HSCLP	DINT	32 位带符号整数	0	读取/写入	低预设
HSCHPOutput	UDINT	32 位二进制	0	读取/写入	输出上限数据
HSCLPOutput	UDINT	32 位二进制	0	读取/写入	输出下限数据

HSCApp 设置与 PLSData 设置的比较

已启用 PLS 功能时，相应的 PLSData 设置会挂起相关的 HSCApp 设置，如下表所示。

HSCApp 设置	PLSData 设置
HSCAPP.HpSetting	HSCHP
HSCAPP.LpSetting	HSCLP
HSCAPP.HPOutput	HSCHPOutput
HSCAPP.LPOutput	HSCLPOutput

PLSData 参数示例

下图展示了变量选择器中的 PLSData 参数。

Variables - Micro850				Local Variables - HSCSampleProgram				System Variables - Micro850				I/O - Micro850				Defined Words - Micro850			
名称				数据类型				特性				注释							
HSC_1.HscCmd				USINT				Read/Write				See HSC Command Values.							
HSC_1.HscAppData				HSCAPP				Read/Write				HSC application configuration.							
HSC_1.HscStsInfo				HSCSTS				Read/Write				HSC dynamic status.							
HSC_1.PlsData								Read/Write				Programmable Limit Switch (PLS) Data Str							
HSC_1.PlsData[1]				PLS				Read/Write											
HSC_1.PlsData[1].HscHP				DINT				Read/Write											
HSC_1.PlsData[1].HscLP				DINT				Read/Write											
HSC_1.PlsData[1].HscHPOutPut				UDINT				Read/Write											
HSC_1.PlsData[1].HscLPOutPut				UDINT				Read/Write											
HSC_1.__ADI_PlsData				UINT				Read/Write				ADI hidden parameter for <PlsData> array i							
HSC_1.Sts				UINT				Read/Write				Execution status. See HSC Status Values.							
md				USINT				Read/Write											
opData				HSCAPP				Read/Write											

另请参见

[配置可编程限位开关 \(PLS\)](#) 参考页数 [360](#)

[高速计数器 \(HSC\) 指令](#) 参考页数 [327](#)

HSC 状态代码 (STS)

下表描述了用于指示 HSC 功能块执行状态的代码。

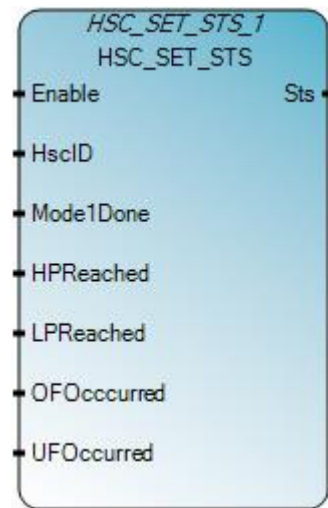
状态代码	状态说明
0x00	未采取行动（未启用）。
0x01	HSC 执行成功。
0x02	HSC 命令无效。
0x03	HSC ID 超出范围。
0x04	HSC 配置错误。

另请参见

[HSC](#) 参考页数 [328](#)

HSC_SET_STS

HSC_SET_STS 手动设置或重置 HSC 计数状态。



HSC_SET_STS 功能块操作

必须停止(不再计数)HSC 功能块，HSC_SET_STS 功能块才能设置或重置其 HTS 状态。如果未停止 HSC 功能块，则输入参数将继续更新，并且将忽略使用 HSC_SET_STS 功能块进行的任何更改。

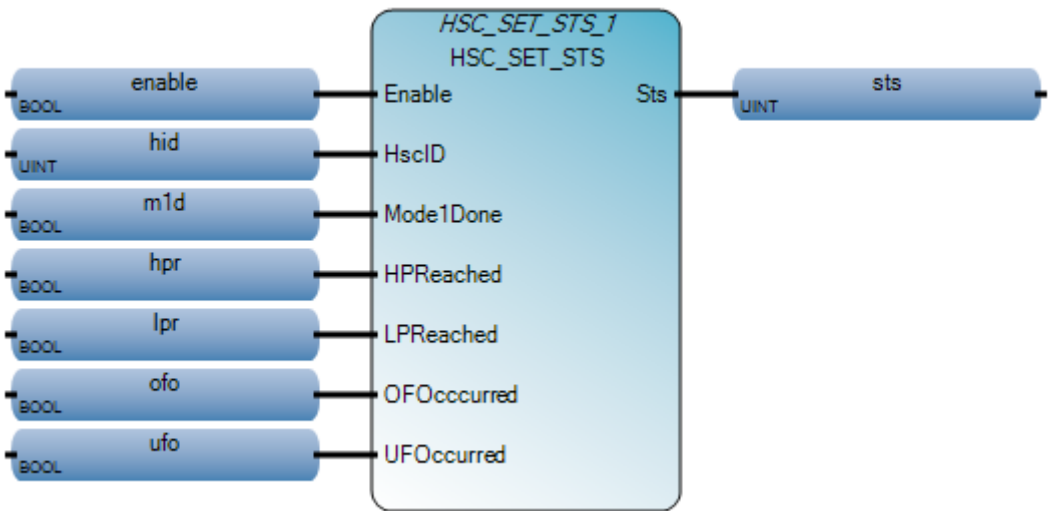
参数

参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，设置/重置 HSC 状态。 当 Enable = FALSE 时，HSC 状态不变。
HscID	输入	UINT	手动设置或重置 HSC 状态。
Mode1Done	输入	BOOL	模式 1A 或 1B 计数完成。
HPReached	输入	BOOL	达到高预设。 当 HSC 未计数时，该位可以重置为 FALSE。

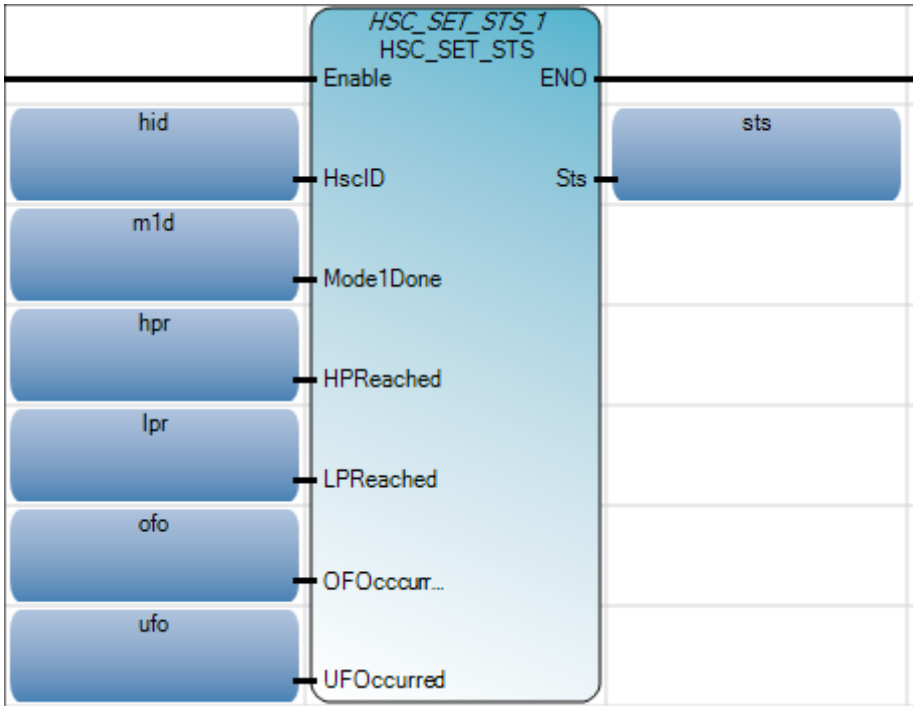
参数	参数类型	数据类型	说明
LPReached	输入	BOOL	达到低预设。 当 HSC 未计数时，该位可以重置为 FALSE。
OFOccurred	输入	BOOL	发生溢出。 需要时该位可以重置为 FALSE。
UFOccurred	输入	BOOL	发生下溢。 需要时该位可以重置为 FALSE。
Sts	输出	UINT	请参见 HSC 状态代码 (STS) 参考页数 350。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

HSC_SET_STS 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 HSC_SET_STS_1(enable, hid, mid, hpr, lpr, ofo, ufo);
2 sts := HSC_SET_STS_1.Sts;
```

HSC_SET_STS_1{

void HSC_SET_STS_1(BOOL Enable, UINT HscID, BOOL Mode1Done, BOOL HPReached, BOOL LPReached, BOOL OFOccurred, BOOL UFOccurred)

Type : HSC_SET_STS, Manually set/reset HSC status.

另请参见

- [使用高速计数器指令](#) 参考页数 353
- [高速计数器 \(HSC\) 指令](#) 参考页数 327

使用高速计数器指令

本节介绍在逻辑程序中使用高速计数器指令的特定详细信息和示例，具体包括：

[更新 HSC 应用程序数据](#) 参考页数 [354](#)

[配置高速计数器 \(HSC\) 用户中断](#) 参考页数 [355](#)

[配置可编程限位开关 \(PLS\)](#) 参考页数 [360](#)

[示例：如何创建高速计数器 \(HSC\)](#) 参考页数 [361](#)

[示例：如何添加可编程限位开关 \(PLS\) 功能](#) 参考页数 [375](#)

[示例：可编程限位开关 \(PLS\) 已启用](#) 参考页数 [378](#)

更新 HSC 应用程序数据

HSC 配置在 HSC 应用程序数据中定义，并且通常仅在 HSC 指令编程前配置一次。HSC 计数期间，将忽略对 HSC 应用程序数据（HSCAppData 参数）进行的更改。

更新 HSC 配置

1. 更新 HSCAppData。
2. 通过命令 0x03（设置/重新加载）调用 HSC 指令。

另请参见

[配置高速计数器 \(HSC\) 用户中断](#) 参考页数 [355](#)

[配置可编程限位开关 \(PLS\)](#) 参考页数 [360](#)

[示例：如何创建高速计数器 \(HSC\) 程序](#) 参考页数 [361](#)

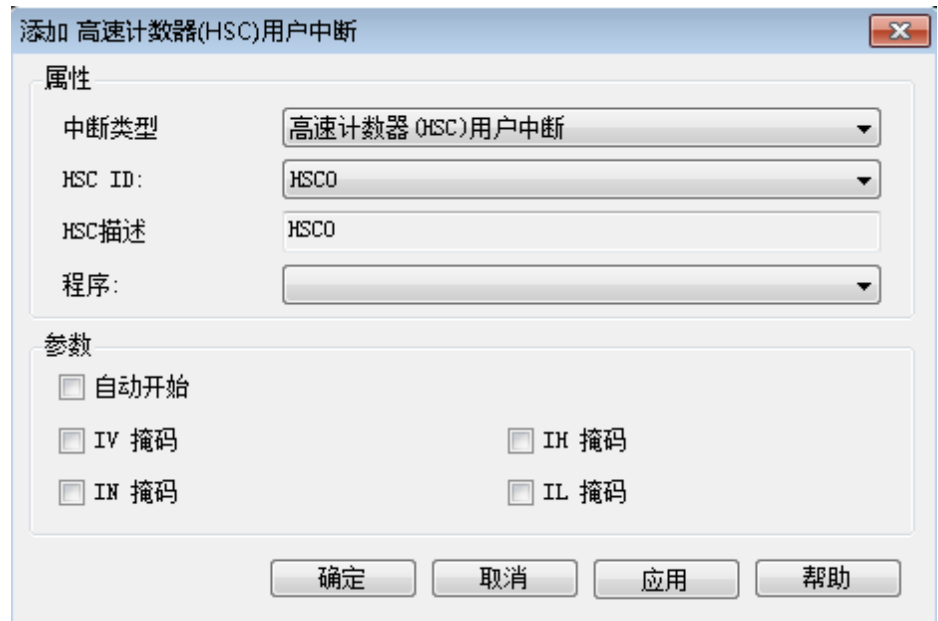
[示例：可编程限位开关 \(PLS\) 已启用](#) 参考页数 [378](#)

[HSC](#) 参考页数 [328](#)

“高速计数器 (HSC) 用户中断”对话框

HSC 中断对话框可用于执行以下操作：

- 配置中断属性，例如 ID 和使用该 ID 的程序。
- 配置中断参数。
- 在“中断类型”中，选择高速计数器 (HSC) 用户中断。



另请参见

[配置高速计数器 \(HSC\) 用户中断](#) 参考页数 [355](#)

[高速计数器指令](#) 参考页数 [327](#)

配置高速计数器 (HSC) 用户中断

用户中断将导致控制器挂起其当前正在执行的任务，执行其他任务，然后回到任务挂起点继续执行挂起的任务。

Micro830 和 Micro850 控制器最多支持六个 HSC 用户中断，这些中断可用于在预配置的事件中执行选定的用户逻辑。



另请参见

[更新 HSC 应用程序数据](#) 参考页数 [354](#)

[配置可编程限位开关 \(PLS\)](#) 参考页数 [360](#)

[示例：如何创建高速计数器 \(HSC\) 程序](#) 参考页数 [361](#)

[示例：可编程限位开关 \(PLS\) 已启用](#) 参考页数 [378](#)

[HSC](#) 参考页数 [328](#)

添加和配置高速计数器 (HSC) 用户中断

请按照以下步骤从控制器的配置工作区中添加和配置 HSC 中断。

添加 HSC 中断

1. 双击控制器以打开控制器配置工作区。
2. 在控制器树中，单击中断以显示“中断配置”页面。

3. 右击空行，然后单击添加以显示“中断属性”对话框。

配置 HSC 中断

1. 在“中断类型”中，选择高速计数器 (HSC) 用户中断。
2. 选择属性：

[HSC 中断属性](#) 参考页数 [357](#)

3. 选择参数：

[HSC 中断参数](#) 参考页数 [358](#)。

另请参见

[配置高速计数器 \(HSC\) 用户中断](#) 参考页数 [355](#)

[高速计数器指令](#) 参考页数 [327](#)

HSC 中断属性

HSC 中断属性状态位指示已启用/已禁用状态、执行状态以及中断条件是否已丢失。

用户中断启用 (HSC0.Enabled)

参数	数据格式	HSC 模式	用户程序访问
HSC0.Enabled	位	0...9	只读

Enabled 位用于指示 HSC 中断启用或禁用状态。

用户中断执行 (HSC0.EX)

参数	数据格式	HSC 模式	用户程序访问
HSC0.Ex	位	0...9	只读

只要 HSC 子系统出于以下任何条件而开始处理 HSC 子例程，EX（用户中断执行）位均会设为（1）：

- 达到低预设

- 达到高预设
- 上溢条件 - 向上计数越过上溢值
- 下溢条件 - 向下计数越过下溢值

在控制程序中，可将 HSC EX 位用作条件逻辑来检测是否正在执行 HSC 中断。

如果控制器完成处理其 HSC 子例程，则 HSC 子系统会将 EX 位清零 (0)。

用户中断挂起 (HSC0.PE)

参数	数据格式	HSC 模式	用户程序访问
HSC0.PE	位	0...9	只读

PE (用户中断挂起) 状态标记指示中断正挂起。如果需要确定何时不能立即执行子例程，则可以在控制程序中监视 PE 状态位或将该位用于逻辑目的。PE 位由控制器维护，并自动设置和清除。

用户中断丢失 (HSC0.LS)

参数	数据格式	HSC 模式	用户程序访问
HSC0.LS	位	0...9	只读

LS (用户中断丢失) 是一个用于指示中断已丢失的状态标记。在设置丢失位前，控制器可以处理 1 个活动的用户中断条件并保持 1 个挂起的用户中断条件。

LS 位由控制器设置。丢失条件则由控制程序来使用和监视。

另请参见

[配置高速计数器 \(HSC\) 用户中断](#) 参考页数 [355](#)

HSC 中断参数

HSC 中断参数用于配置开始和掩码选项。

自动开始 (HSC0.AS)

参数	数据格式	HSC 模式	用户程序访问
HSC0.AS	位	0...9	只读

“自动启动”由编程设备配置，并作为用户程序的一部分存储。自动启动位定义是否只要控制器进入任何运行或测试模式就自动启动 HSC 中断功能。

上溢掩码 (HSC0.MV)

参数	数据格式	HSC 模式	用户程序访问
HSC0.MV	位	0...9	只读

MV (上溢掩码) 控制位用于启用 (允许) 或禁止 (不允许) 发生上溢中断。如果该位已清除 (0)，且 HSC 检测到“达到上溢”条件，则不会执行 HSC 用户中断。

MV 位由用户程序控制，在重新上电过程中其值将保留。用户程序必须设置和清除 MV 位。

下溢掩码 (HSC0.MN)

参数	数据格式	HSC 模式	用户程序访问
HSC0.MN	位	2...9	只读

MN (下溢掩码) 控制位用于启用 (允许) 或禁止 (不允许) 发生下溢中断。如果该位已清除 (0)，且 HSC 检测到“达到下溢”条件，则不会执行 HSC 用户中断。

MN 位由用户程序控制，在重新上电过程中其值将保留。用户程序必须设置和清除 MN 位。

高预设掩码 (HSC0.MH)

参数	数据格式	HSC 模式	用户程序访问
HSC0.MH	位	0...9	只读

低预设掩码 (HSC0.ML)

另请参见

配置高速计数器 (HSC) 用户中断 参考页数 355

配置可编程限位开关 (PLS)

在 HSC 中启用 PLS

PLS 模式仅可与 Micro800 控制器的 HSC 共同运行，并且必须在 HSC 中启用，方法是将 HSCAppData.PLSEnable 参数设置为 True。

启用 PLS 时的 HSC 操作

PLS 功能可与所有其他 HSC 功能一起使用，包括能够选择通过哪个 HSC 事件来生成用户中断。已启用 PLS 功能且控制器处于运行模式下时，HSC 对传入脉冲计数，并发生以下事件。

- 当计数达到 PLS 数据中定义的首个预设 (HSCHP 或 HSCLP) 时，将通过 HSC 掩码 (HSCAPP.OutputMask) 写入输出源数据 (SCHPOutput 或 HSCLPOutput)。
- 此时，PLS 数据中定义的下个预设 (HSCHP 和 HSCLP) 将变为活动的。
- 当 HSC 计数至新预设时，将通过 HSC 掩码写入新的输出数据。
- 此过程会继续，直至载入 PLS 数据块中的最后一个元素。
- 此时，PLS 数据块中的活动元素会重置为零。
- 此行为称作循环操作

另请参见

[更新 HSC 应用程序数据](#) 参考页数 [354](#)

[配置高速计数器 \(HSC\) 用户中断](#) 参考页数 [355](#)

[示例：如何创建高速计数器 \(HSC\) 程序](#) 参考页数 [361](#)

[示例：可编程限位开关 \(PLS\) 已启用](#) 参考页数 [378](#)

[HSC](#) 参考页数 [328](#)

示例：如何创建高速计数器 (HSC) 程序

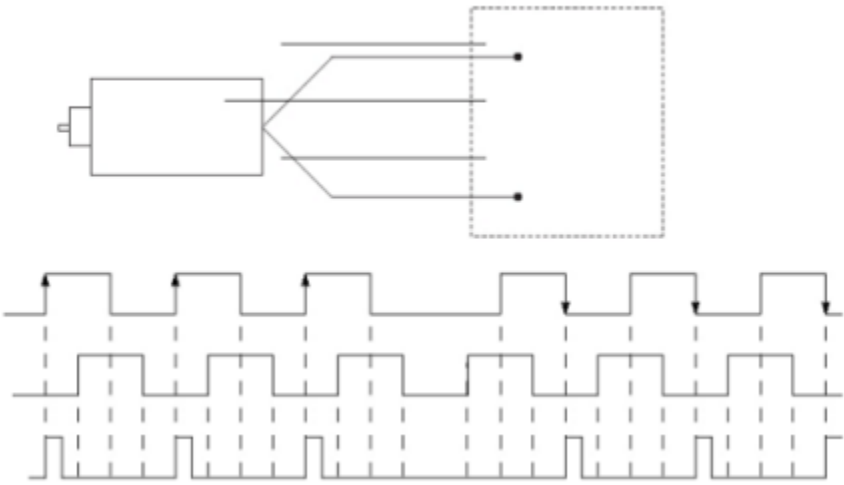
本例展示了如何创建使用正交编码器并具有可编程限位开关 (PLS) 功能的高速计数器 (HSC) 程序。

示例中使用的正交编码器

高速计数器程序示例采用 HSC 功能块和带相位输入 A 与 B 的正交计数器。正交编码器确定旋转方向和旋转设备（例如车床）位置。双向计数器将对正交编码器的旋转计数。

以下正交编码器连接到输入 0 和 1。计数方向由 A 和 B 之间的相位角决定：

- 如果 A 先于 B，则计数器递增。
- 如果 B 先于 A，则计数器递减。



创建高速计数器 (HSC) 程序

执行以下任务以创建、生成和测试 HSC 程序，然后添加 PLS 功能。

不支持	任务
1	创建梯形图并添加变量 参考页数 363
2	将值分配给 HSC 变量 参考页数 368
3	分配变量并生成程序 参考页数 369
4	测试程序并运行高速计数器 参考页数 372
5	添加可编程限位开关 (PLS) 功能 参考页数 375

另请参见

[更新 HSC 应用程序数据](#) 参考页数 [354](#)

[配置高速计数器 \(HSC\) 用户中断](#) 参考页数 [355](#)

[配置可编程限位开关 \(PLS\)](#) 参考页数 [360](#)

[示例：可编程限位开关 \(PLS\) 已启用](#) 参考页数 [378](#)

[HSC](#) 参考页数 [328](#)

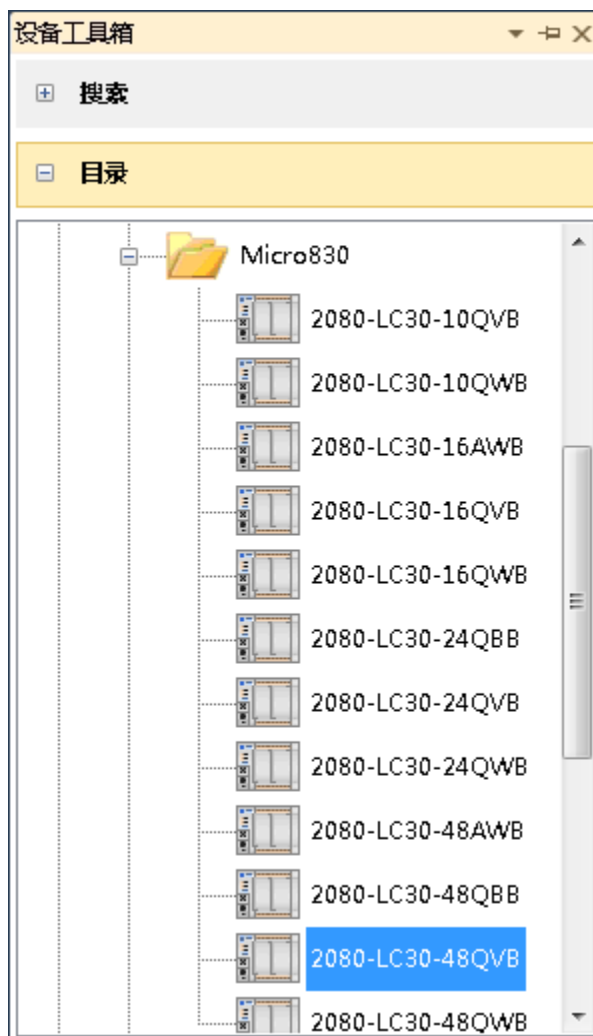
创建梯形图并添加变量

请按照以下步骤创建梯形图，然后将局部变量添加到梯级。此示例程序使用 2080-LC50-24QVB 控制器。除了 2080-LCxx-xxAWB 控制器类型，所有 Micro830 和 Micro850 控制器中均支持 HSC。

创建梯形图并添加变量

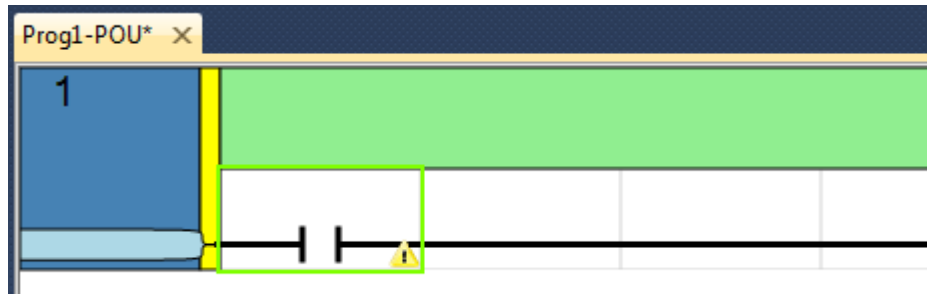
1. 在**设备工具箱**中，展开**类型**选项卡以查看设备文件夹。

2. 展开“控制器”文件夹和 Micro830 文件夹，以查看所有 Micro830 控制器。双击控制器 (2080-LC50-24QVB)，以将其添加到项目组织器。



3. 在项目组织器中，右击程序，单击添加，然后单击新 LD:梯形图，以添加新梯形逻辑程序。
4. 右击 **UntitledLD** 并选择打开。
5. 在工具箱中：

- 双击正向触点以将其添加到梯级，或者
- 将正向触点拖放到梯级上。

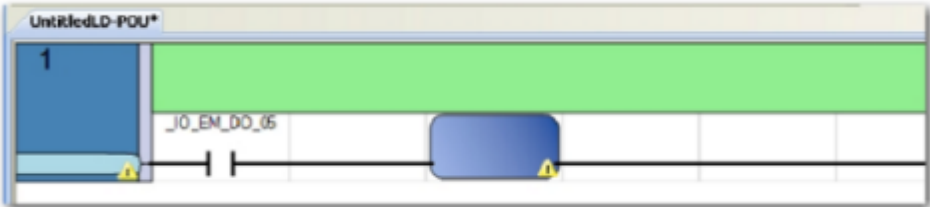


6. 将变量分配给正向触点:

- 双击正向触点以显示**变量选择器**，然后单击 **I/O - Micro830** 选项卡。
- 单击 **_IO_EM_DI_05**，然后单击**确定**，以将正向触点分配给输入 5。



7. 在工具箱中，选择功能块，并将其拖至正向触点右侧，如下图中所示。

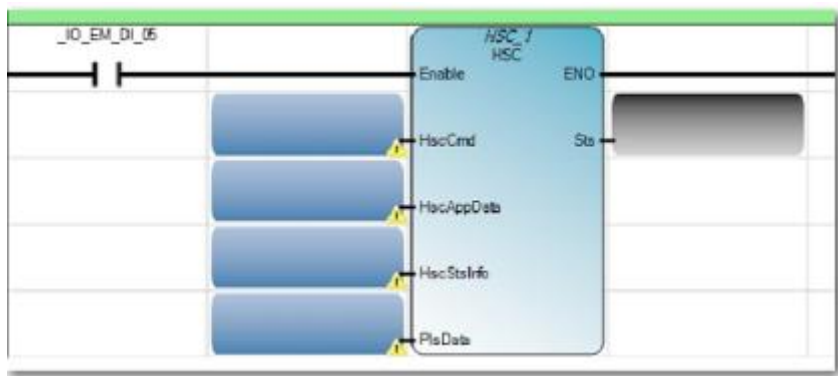


8. 双击功能块以显示**模块选择器**。

9. 在**模块选择器**中，选择 **HSC** 并单击**确定**。

提示：在**搜索**中键入 HSC 以显示所有 HSC 功能块。

10. 验证梯形梯级是否与下图相似。



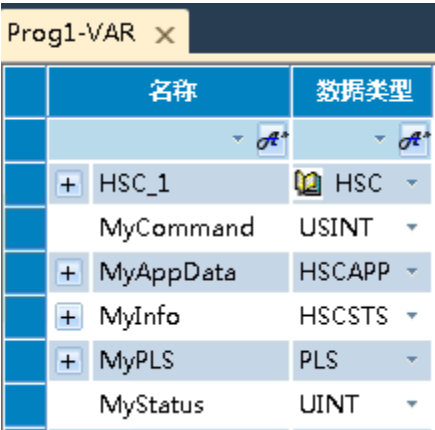
11. 在**项目组织器**中，双击**局部变量**以显示**变量**页面。

12. 在“**变量**”页面中，添加以下变量和数据类型。

变量名	数据类型
MyCommand	USINT
MyAppData	HSCAPP
MyInfo	HSCSTS
MyPLS	PLS
MyStatus	UINT

结果

变量页面应与下图相似。



另请参见

[示例：如何创建高速计数器 \(HSC\) 程序](#) 参考页数 [361](#)

[HSC](#) 参考页数 [328](#)

[配置可编程限位开关 \(PLS\)](#) 参考页数 [360](#)

[添加可编程限位开关 \(PLS\) 功能](#) 参考页数 [375](#)

将值分配给 HSC 变量

添加变量后，请按照以下步骤，通过使用变量选择器中的“初始值”列，将值添加到变量。标准程序通常使用例程将值分配给变量。

将值分配给 HSC 变量

1. 展开 **MyAppData** 以查看所有变量。
2. 分配 HSC 模式值：
 - 在 MyAppData.HSCMode 变量的“初始值”字段中，键入 6。
 - 有关每个值说明的更多信息，请参见 [HSCAPP 数据类型](#) 参考页数 [332](#)中的“SCMode”。
3. 将其余值分配给 MyAppData 变量，如下图所示。

- 在“初始值”字段中，输入该值。
- 有关每个值说明的更多信息，请参见 [HSCAPP 数据类型](#) 参考页数 [332](#)

	名称	别名	数据类型	维度	项目值
	MyAppData		HSCAPP		...
	MyAppData.PlsEnable		BOOL		
	MyAppData.HscID		UINT		
	MyAppData.HscMode		UINT		
	MyAppData.Accumulator		DINT		
	MyAppData.HPSetting		DINT		
	MyAppData.LPSetting		DINT		
	MyAppData.OFSetting		DINT		
	MyAppData.UFSetting		DINT		
	MyAppData.OutputMask		UDINT		
	MyAppData.HPOutput		UDINT		
	MyAppData.LPOutput		UDINT		

4. 分配 HSC 命令值：

- 在 MyCommand 变量的“初始值”字段中，键入 1。
- 有关命令值的更多信息，请参见 [HSCCmd 值](#) 参考页数 [330](#)。

后续步骤：

[分配变量并生成程序](#) 参考页数 [369](#)

另请参见

[示例：如何创建高速计数器 \(HSC\) 程序](#) 参考页数 [361](#)

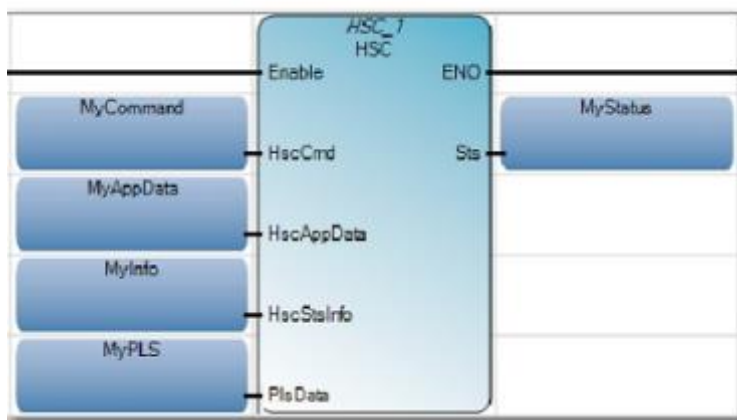
[配置可编程限位开关 \(PLS\)](#) 参考页数 [360](#)

[添加可编程限位开关 \(PLS\) 功能](#) 参考页数 [375](#)

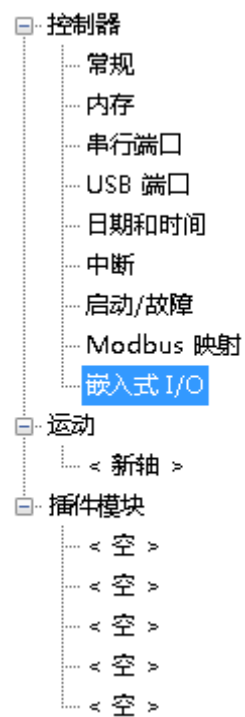
分配变量并生成程序

在 HSC 变量中输入值后，请按照以下步骤，将变量分配给功能块，然后生成程序。

1. 在梯形图中, 将每个变量分配给其 HSC 功能块元素, 如下图所示。



2. 在项目组织器中, 单击控制器以显示控制器树。



3. 在控制器树中，单击**嵌入式 I/O**，然后选择编码器的输入过滤器。

控制器 - 嵌入式 I/O

输入筛选器		输入闭锁信号和 EII 沿		
输入	输入筛选器	输入	启用锁存	EII 沿
0-1	默认	0	☐ 下降	下降
2-3	默认	1	☐ 下降	下降
4-5	默认	2	☐ 下降	下降
6-7	默认	3	☐ 下降	下降
8-9	默认	4	☐ 下降	下降
10-11	默认	5	☐ 下降	下降
12-13	默认	6	☐ 下降	下降
14-15	默认	7	☐ 下降	下降
16-23	默认	8	☐ 下降	下降
24-27	默认	9	☐ 下降	下降

4. 验证编码器是否与 Micro830 控制器相连。
5. 启动 Micro830 控制器，并将其连接到您的计算机。
6. 生成程序，然后将其下载到控制器。

后续步骤：

[测试程序并运行高速计数器](#) 参考页数 [372](#)

另请参见

[示例：如何创建高速计数器 \(HSC\) 程序](#) 参考页数 [361](#)

[创建梯形图并添加变量](#) 参考页数 [363](#)

[将值分配给 HSC 变量](#) 参考页数 [368](#)

[测试程序并运行高速计数器](#) 参考页数 [372](#)

[配置可编程限位开关 \(PLS\)](#) 参考页数 [360](#)

[添加可编程限位开关 \(PLS\) 功能](#) 参考页数 [375](#)

测试程序并运行高速计数器

在将 HSC 程序下载到控制器后，您可以测试该程序，然后运行高速计数器。

测试程序

- 1. 连接到控制器。
- 2. 在项目组织器中，双击 HSC 程序，然后双击局部变量。

您可以查看以下两个 HSC 输出的值：STS (MyStatus) 和 HSCSTS (MyInfo)。

- 3. 双击 _IO_EM_DI_05 正向触点，以显示变量选择器窗口。
- 4. 单击 I/O Micro830 选项卡，然后单击 _IO_EM_DI_05 行。
- 5. 选择锁定和逻辑值，以将输入强制设为开启位置。



查看结果

1. 单击**局部变量**选项卡以查看变量更改。
2. 展开 **MyAppData** 和 **MyInfo** 变量列表。
3. 打开编码器，以查看计数器的向上/向下计数。例如，如果编码器已连接到电机轴，则开启电机以触发 HSC 计数。
4. 验证 **MyStatus** 变量的逻辑值是否为 1，这表示 HSC 正在运行。
5. 查看 **MyInfo.Accumulator** 中的计数器值。

提示：有关状态代码的完整列表，请参见 [HSC 状态代码 \(STS\)](#) 参考页数 [350](#)。

结果

在本例中，一旦 **MyInfo.Accumulator** 达到高预设值 40，则输出 0 将开启，并且 **HPReached** 标记将开启。一旦 **MyInfo.Accumulator** 达到低预设值 -40，则输出 1 将开启，并且 **LPReached** 标志也将开启。

变量监视

用户全局变量 - 局部变量 - Progl 系统变量- Micro830 I/O - Micro830 已定义的字 - 0

名称	逻辑值	实际值	初始值
MyAppData.PlsEnable	☐	不可用	
MyAppData.HscID	0	不可用	0
MyAppData.HscMode	7	不可用	7
MyAppData.Accumulator	40	不可用	40
MyAppData.HPSSetting	40	不可用	40
MyAppData.LPSSetting	-40	不可用	-40
MyAppData.OFSSetting	50	不可用	50
MyAppData.UFSSetting	-50	不可用	-50
MyAppData.OutputMask	3	不可用	3
MyAppData.HPOutput	1	不可用	1
MyAppData.LPOutput	2	不可用	2
MyInfo	...	...	...
MyInfo.CountEnable	☒	不可用	
MyInfo.ErrorDetected	☐	不可用	
MyInfo.CountUpFlag	☒	不可用	
MyInfo.CountDwnFlag	☒	不可用	
MyInfo.ModelDone	☐	不可用	
MyInfo.OVF	☐	不可用	
MyInfo.UNF	☐	不可用	
MyInfo.CountDir	☐	不可用	
MyInfo.HPReached	☒	不可用	
MyInfo.LPReached	☐	不可用	
MyInfo.OFCauseInter	☐	不可用	
MyInfo.UFCauseInter	☐	不可用	
MyInfo.HPCauseInter	☐	不可用	
MyInfo.LPCauseInter	☐	不可用	
MyInfo.PlsPosition	0	不可用	0
MyInfo.ErrorCode	0	不可用	0
MyInfo.Accumulator	40	不可用	40
MyInfo.HP	40	不可用	40
MyInfo.LP	-40	不可用	-40
MyInfo.HPOutput	1	不可用	1
MyInfo.LPOutput	2	不可用	2
MyPLS	...	...	...
MyStatus	1	不可用	1

另请参见

[示例：如何创建高速计数器 \(HSC\) 程序](#) 参考页数 [361](#)

[配置可编程限位开关 \(PLS\)](#) 参考页数 [360](#)

[添加可编程限位开关 \(PLS\) 功能](#) 参考页数 [375](#)

监视列表

添加可编程限位开关 (PLS) 功能

本例展示了如何将可编程限位开关 (PLS) 功能添加到 HSC 程序。

用于计数器设置的变量值

- **MyAppData.PlsEnable** 用于启用或禁用 PLS 设置。如果使用了 MyAppData 变量，则应将它设置为 FALSE（已禁用）。
- **MyAppData.HscID** 用于指定基于模式和应用程序类型应使用哪些嵌入式输入。请参见 HSC 输入和配线图以了解可使用的各种 ID 以及嵌入式输入和其特性。
- 如果使用 ID 0，则不能在同一个控制器上使用 ID 1，因为输入已被“重置”和“保存”使用。
- **MyAppData.HscMode** 用于指定 HSC 将用于计数的操作类型。请参见 HSC 模式 (HSCAPP.HSCMode)。

启用 PLS

1. 在项目组织器中，双击局部变量以显示变量页面。
2. 启用 PLS 功能：
 - 在 MyAppData.PlsEnable 变量的“初始值”字段中，选择 TRUE。
3. 配置下溢和上溢设置：

- 在 MyAppData.OFSetting 的“初始值”字段中，键入 50。
 - 在 MyAppData.UFSetting 的“初始值”字段中，键入 -50。
4. 如果要使用输出，请配置输出掩码。

结果

在本例中，PLS 变量具有维度 [1..4]。这意味着，HSC 可以具有四对高预设和低预设。

- 高预设应始终设为低于 OFSetting，低预设则应始终高于 UFSetting。
- HscHPOutPut 值和 HscLPOutPut 值将确定在达到高预设和低预设时会开启的输出。

Prog1-POU		Micro830		Prog1-VAR x		
	名称	数据类型	维度	别名	初始值	特性
+	HSC_1	HSC			...	读/写
	MyCommand	USINT				读/写
+	MyAppData	HSCAPP			...	读/写
+	MyInfo	HSCSTS			...	读/写
-	MyPLS	PLS	[1..4]		...	读/写
	- MyPLS[1]	PLS			...	读/写
	MyPLS[1].Hs	DINT				读/写
	MyPLS[1].Hs	DINT				读/写
	MyPLS[1].Hs	UDINT				读/写
	MyPLS[1].Hs	UDINT				读/写
	- MyPLS[2]	PLS			...	读/写
	MyPLS[2].Hs	DINT				读/写
	MyPLS[2].Hs	DINT				读/写
	MyPLS[2].Hs	UDINT				读/写
	MyPLS[2].Hs	UDINT				读/写
	- MyPLS[3]	PLS			...	读/写
	MyPLS[3].Hs	DINT				读/写
	MyPLS[3].Hs	DINT				读/写
	MyPLS[3].Hs	UDINT				读/写
	MyPLS[3].Hs	UDINT				读/写
	- MyPLS[4]	PLS			...	读/写
	MyPLS[4].Hs	DINT				读/写
	MyPLS[4].Hs	DINT				读/写
	MyPLS[4].Hs	UDINT				读/写
	MyPLS[4].Hs	UDINT				读/写
	MyStatus	UINT				读/写

另请参见

[示例：如何创建高速计数器 \(HSC\) 程序](#) 参考页数 361

[配置可编程限位开关 \(PLS\)](#) 参考页数 [360](#)

示例：可编程限位开关 (PLS) 已启用

本主题描述了使用特定 HSC 和 PLSData 参数值启用 PLS 后的结果。

HSC 参数值

本例假定以下 HSC 参数使用这些值。

- HSCApp.OutputMask = 31
- HSCApp.HSCMode = 0
- HSC 仅控制嵌入式输出 0...4

PLSData 参数值

本例假定变量 (HSC_PLS) 的 PLSData 参数如下图所示配置。

	名称	数据类型	维度	字符串大小	初始值	特性
	 					
	HSC_1	 HSC			...	读/写
	HSC_PLS	PLS	[1..4]		...	读/写
	HSC_PLS[1]	PLS			...	读/写
	HSC_PLS[1]	DINT			250	读/写
	HSC_PLS[1]	DINT			-2	读/写
	HSC_PLS[1]	UDINT			3	读/写
	HSC_PLS[1]	UDINT			0	读/写
	HSC_PLS[2]	PLS			...	读/写
	HSC_PLS[2]	DINT			500	读/写
	HSC_PLS[2]	DINT			-2	读/写
	HSC_PLS[2]	UDINT			7	读/写
	HSC_PLS[2]	UDINT			0	读/写
	HSC_PLS[3]	PLS			...	读/写
	HSC_PLS[3]	DINT			750	读/写
	HSC_PLS[3]	DINT			-2	读/写
	HSC_PLS[3]	UDINT			15	读/写
	HSC_PLS[3]	UDINT			0	读/写
	HSC_PLS[4]	PLS			...	读/写
	HSC_PLS[4]	DINT			1000	读/写
	HSC_PLS[4]	DINT			-2	读/写
	HSC_PLS[4]	UDINT			31	读/写
	HSC_PLS[4]	UDINT			0	读/写
						

PLS 启用结果

在本例中，将发生以下事件。

- 当梯形逻辑首次运行时：HSCSTS.Accumulator = 1，这意味着所有输出均会关闭。
- 当 HSCSTS.Accumulator = 250 时，将通过 HSCAPP.OutputMask 发送 HSC_PLS[1].HSCHPOutput，并激发输出 0 和 1。

- 当 HSCSTS.Accumulator 达到 500、750 和 1000 时将反复通过输出掩码发送高预设输出，而控制器分别激发输出 0...2、0...3 和 0...4。
- 在整个操作完成后，该周期将重置并从 HSCSTS.HP = 250 开始重复。

另请参见

[更新 HSC 应用程序数据](#) 参考页数 [354](#)

[配置高速计数器 \(HSC\) 用户中断](#) 参考页数 [355](#)

[配置可编程限位开关 \(PLS\)](#) 参考页数 [360](#)

[示例：如何创建高速计数器 \(HSC\) 程序](#) 参考页数 [361](#)

[HSC](#) 参考页数 [328](#)

输入/输出指令

通过向以物理形式连接至可编程逻辑控制器的设备发送信号，输入/输出指令从控制器或模块读取或向其写入数据。输入继电器将信号传输至内部继电器，输出继电器将信号传输至外部输出设备。

函数	说明
LCD 参考页数 382	显示字符串或数字 (仅限 Micro810™)
LCD_BKLT_REM 参考页数 384	更改远程 LCD 背光颜色和模式
LCD_REM 参考页数 387	在远程 LCD 上显示消息
RHC 参考页数 392	读取高速时钟
RPC 参考页数 394	读取用户程序校验和
功能块	说明
DLG 参考页数 395	将数据和全局/局部变量保存到 SD 卡数据记录文件
IIM 参考页数 398	在进行正常输出扫描前更新输入
IOM 参考页数 401	在进行正常输出扫描前更新输出
KEY_READ 参考页数 404	在可选 LCD 模块上读取键状态 (仅限 Micro810™)
KEY_READ_REM 参考页数 407	读取远程 LCD 上的键状态
MM_INFO 参考页数 411	读取内存模块标题信息
PLUGIN_INFO 参考页数 414	从类属插件模块中获取模块信息 (不包括模块信息)
PLUGIN_READ 参考页数 417	从类属插件模块中获取数据
PLUGIN_RESET 参考页数 419	重置类属插件模块 (硬件重置)
PLUGIN_WRITE 参考页数 421	将数据写入到类属插件模块
RCP 参考页数 424	从 SD 内存卡读取配方数据/向其写入配方数据
RTC_READ 参考页数 427	读取实时时钟 (RTC) 模块信息
RTC_SET 参考页数 430	将实时时钟数据设置到实时时钟模块中

函数	说明
SYS_INFO 参考页数 432	读取 Micro800™ 的系统状态
TRIMPOT_READ 参考页数 436	从特定微调电位中读取微调电位值

LCD

LCD 在可选 LCD 模块上显示字符串或数字。



LCD 函数操作

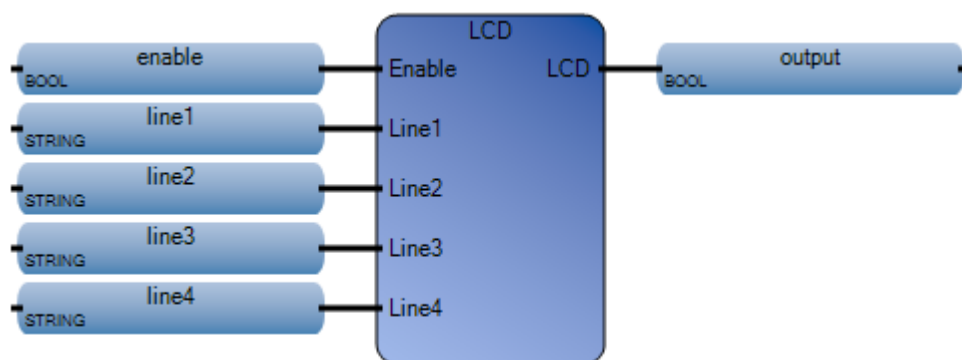
只有 Micro810 控制器支持 LCD 函数。

参数

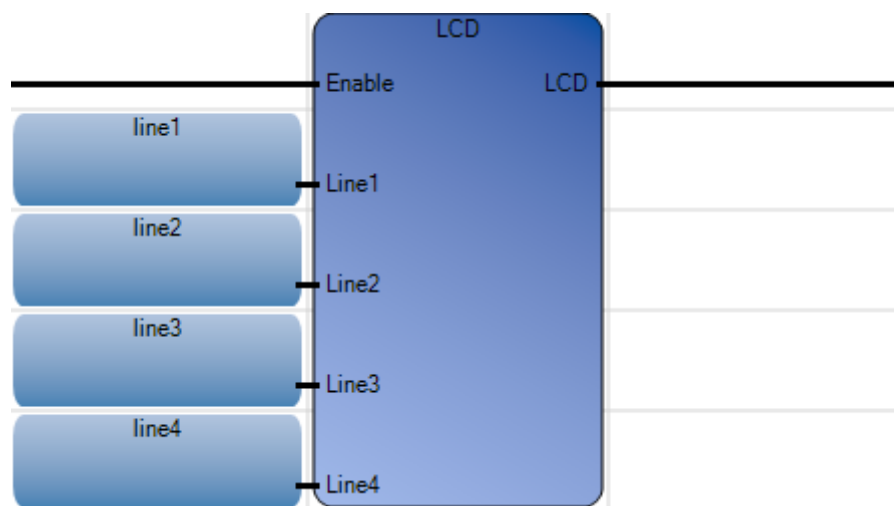
参数	参数类型	数据类型	说明
启用	输入	BOOL	函数启用。 当 Enable = TRUE 时，LCD 从 I/O 状态屏幕切换到用户定义的屏幕（字符串显示在 LCD 上）。 当 Enable = FALSE 时，LCD 显示 I/O 状态屏幕的内容。
Line1	输入	STRING	要显示在 LCD 第 1 行的字符串。
Line2	输入	STRING	要显示在 LCD 第 2 行的字符串。
Line3	输入	STRING	要显示在 LCD 第 3 行的字符串。
Line4	输入	STRING	要显示在 LCD 第 4 行的字符串。
LCD	输出	BOOL	当为 TRUE 时，启用该函数。

LCD 函数语言示例

功能块图

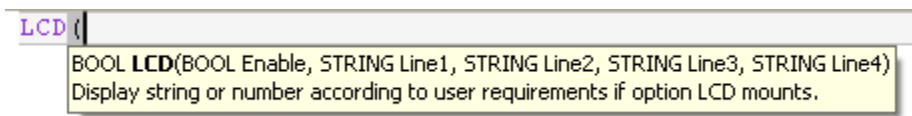


梯形图



结构化文本

```
1 enable := TRUE;  
2 line1 := 'R';  
3 line2 := 'O';  
4 line3 := 'C';  
5 line4 := 'K';  
6 output := LCD(enable, line1, line2, line3, line4);
```



(* 与之等效的 ST: *)

TESTOUTPUT := LCD(LCDENABLE, LINE1, LINE2, LINE3, LINE4);

结果



另请参见

[输入/输出指令](#) 参考页数 [381](#)

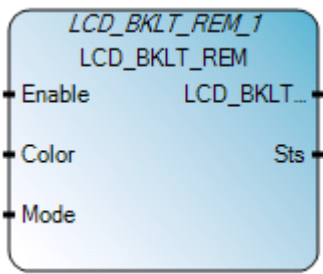
[KEY_READ](#) 参考页数 [404](#)

LCD_BKLT_REM

LCD_BKLT_REM 在用户程序中设置远程 LCD 背光参数。

可在用户程序中使用 LCD_BKLT_REM 功能块以设置远程 LCD 背光参数。只有 Micro 820 支持此功能。

LCD_BKLT_REM 仅在远程 LCD 正在显示用户定义的屏幕（通过使用 LCD_REM FB）或默认 IO 状态屏幕时才有效。对于所有其他屏幕，通过菜单完成的背光参数设置将生效。当 Enable 输入转为 False 时，最后的菜单设置将生效



LCD_BKLT_REM 操作

- 只有 Micro820 控制器支持 LCD_BKLT_REM。
- LCD_BKLT_REM 仅在远程 LCD 正在显示用户定义的屏幕（通过使用 LCD_REM 功能块）或默认 I/O 状态屏幕时才有效。对于所有其他屏幕，通过菜单配置的背光参数设置将生效。
- 当 Enable 输入转为 False 时，最后的菜单设置将生效。

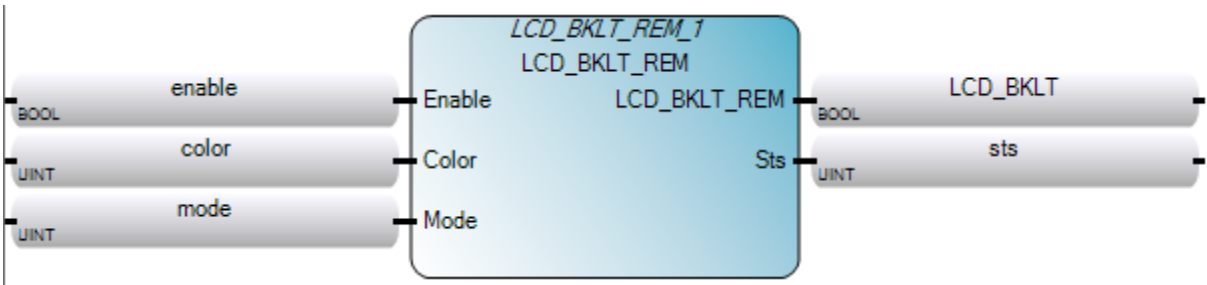
LCD_BKLT_REM 参数

参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行 REM_LCD_BKLT 功能，覆盖所有当前的背光设置。 当 Enable = FALSE 时，将禁用 REM_LCD_BKLT。
颜色	输入	UINT	背光颜色代码 • 0：白色 • 1：蓝色 • 2：红色 • 3：绿色 • 4-65535：保留
模式	输入	UINT	• 0：永久关闭 • 1：永久开启 • 2：闪存（1 秒间隔） • 3-65535：保留

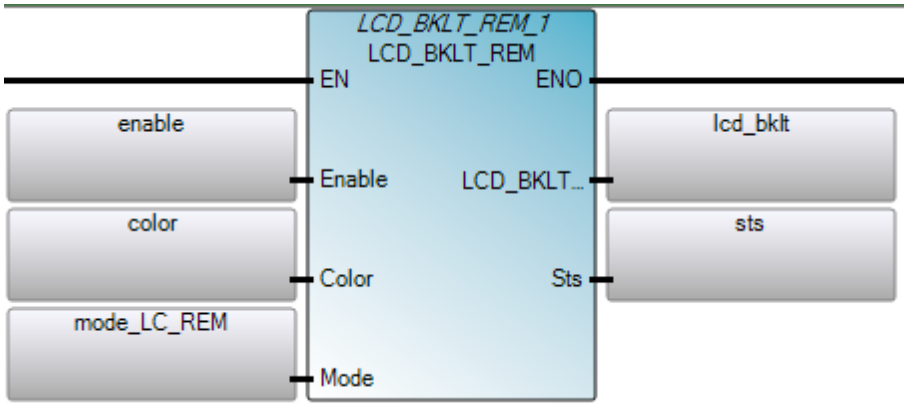
参数	参数类型	数据类型	说明
LCD_BKLT_REM	输出	BOOL	为 TRUE 时：成功执行了指令。 为 FALSE 时：在执行指令时发生错误。
Sts	输出	UINT	远程 LCD 操作的状态。 请参见 LCD_BKLT_REM 状态代码 参考页数 387 。

LCD_BKLT_REM 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 LCD_BKLT_REM_1 (EN, Enable, Color, Mode);
2 output := LCD_BKLT_REM_1.ENO
3 LCD_BKLT_REM_1 := LCD_BKLT_REM_1.LCD_BKLT_REM
4 sts_lcd_rem := LCD_REM_1.Sts
```

LCD_BKLT_REM_4(
void LCD_BKLT_REM_4(BOOL Enable, UINT Color, UINT Mode)
Type: LCD_BKLT_REM, Set the remote LCD backlight parameters.

另请参见

[输入/输出指令](#) 参考页数 [381](#)

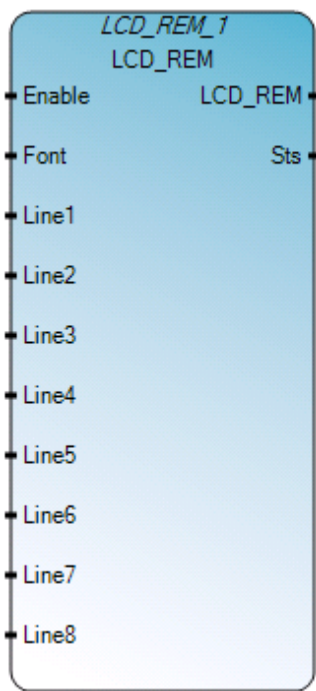
LCD_BKLT_REM 状态代码

状态代码	说明
0	Enable 输入为 False。
1	成功。
2	未检测到远程 LCD。 出现以下情况时可能会发生： - 未将远程 LCD 物理连接到控制器(或配线错误)。 - 串行端口设置不符合远程 LCD 要求。
3	连接错误。 当存在内部状态机器错误时可能会发生。(可能是控制器固件版本和 RLCD 固件版本之间的不兼容引起的。
4	颜色代码无效。
5	模式无效。
6-65535	已保留。

LCD_REM

可在程序中使用 LCD_REM 功能块，以便在远程 LCD 存在且已连接

时显示上面的用户字符串。只有 Micro820 支持此功能块。



LCD_REM 操作

- 只有 Micro820 控制器支持 LCD_REM。

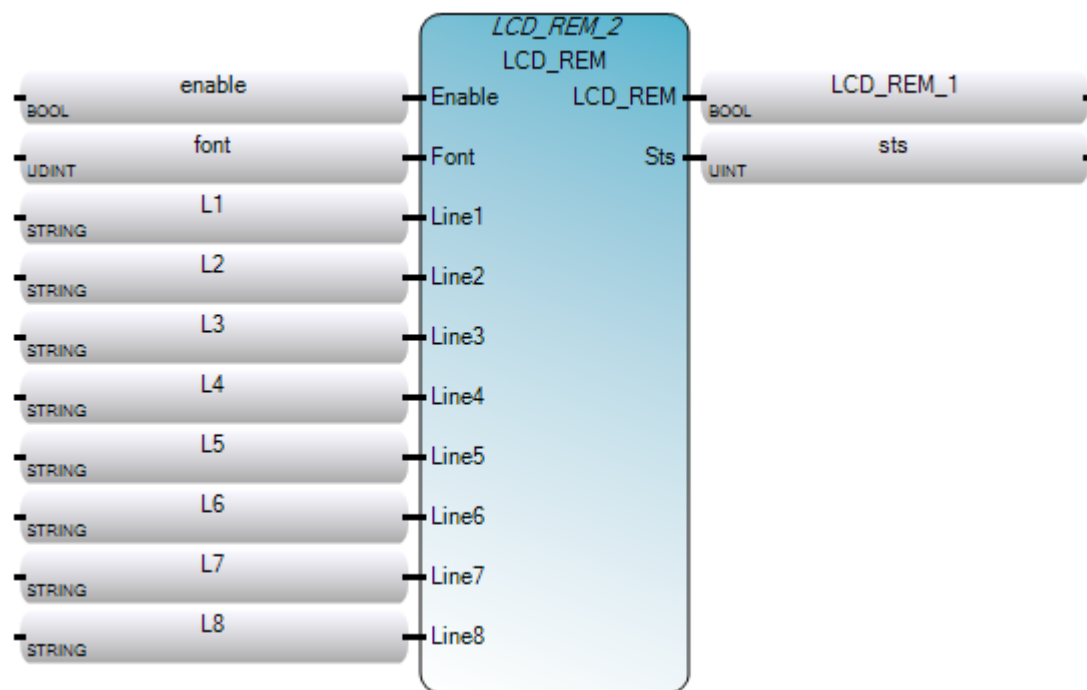
LCD_REM 参数

参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，远程 LCD 从 I/O 状态屏幕切换到用户定义的屏幕。 当 Enable = FALSE 时，远程 LCD 切换回 I/O 状态屏幕。

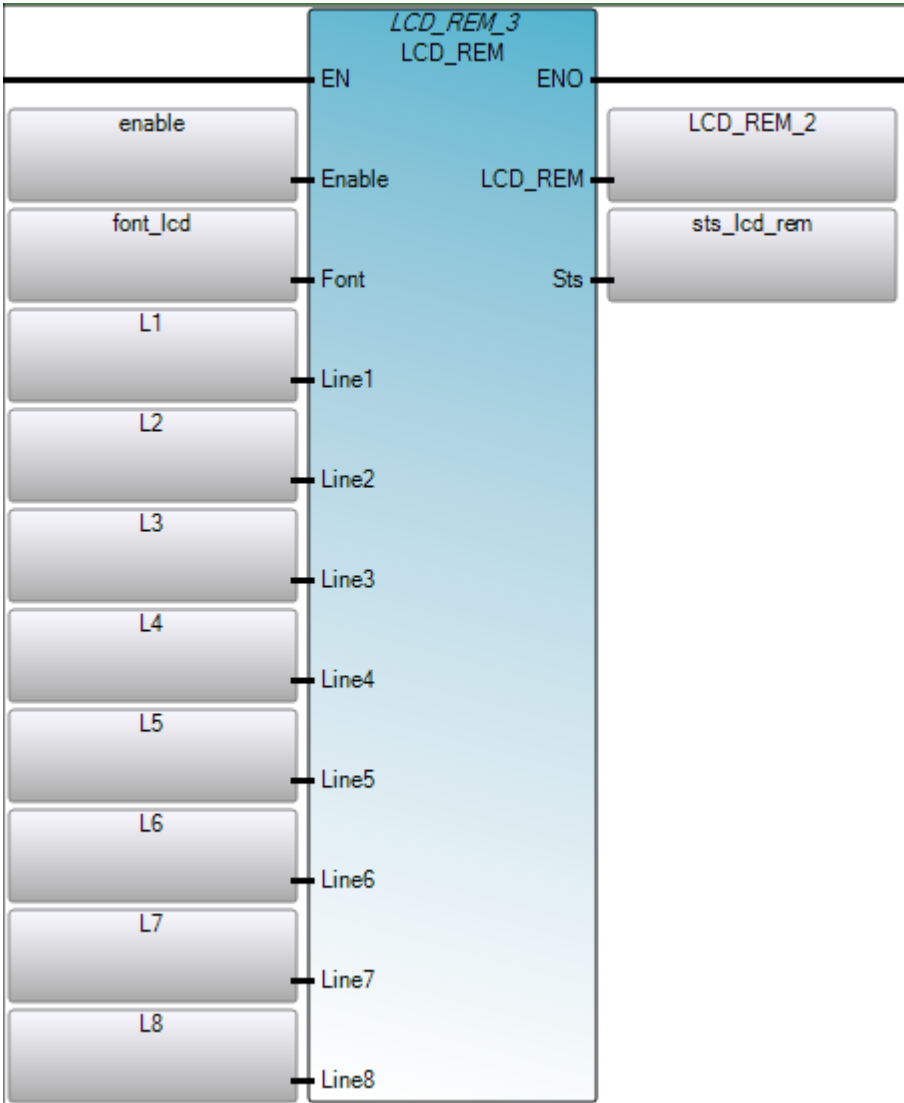
参数	参数类型	数据类型	说明
字体	输入	UDINT	启动消息字体大小的值。 有效值： • 0：默认（大 - 8x16） • 1：小（8x8） • 2：大（8x16） • 3：超大（16x16） • 4 向前：保留 远程 LCD 大小为 192x64 像素。如果选择大字体或超大字体，则远程 LCD 仅显示 Line1 到 Line4 字符串。在此情况下忽略 Line5 到 Line 8 字符串输入。 如果选择小字体或大字体，则远程 LCD 每行最多可以显示 24 个字符。对于超大字体，远程 LCD 每行最多可显示 12 个字符。
Line1	输入	字符串	要显示在 LCD 第 1 行的字符串。最多 24 个字符。
Line2	输入	字符串	要显示在 LCD 第 2 行的字符串。最多 24 个字符。
Line3	输入	字符串	要显示在 LCD 第 3 行的字符串。最多 24 个字符。
Line4	输入	字符串	要显示在 LCD 第 4 行的字符串。最多 24 个字符。
Line5	输入	字符串	要显示在 LCD 第 5 行的字符串。最多 24 个字符。
Line6	输入	字符串	要显示在 LCD 第 6 行的字符串。最多 24 个字符。
Line7	输入	字符串	要显示在 LCD 第 7 行的字符串。最多 24 个字符。
Line8	输入	字符串	要显示在 LCD 第 8 行的字符串。最多 24 个字符。
LCD_REM	输出	BOOL	功能块启用。 当 Enable = TRUE 时，用户显示处于活动状态。 当 Enable = FALSE 时，IO 状态/菜单显示处于活动状态。
Sts	输出	UINT	远程 LCD 操作的状态。 请参见LCD REM 状态代码。参考页数 392

LCD_REM 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 LCD_REM_1 (EN, Enable, Font, Line1, Line2, Line3, Line4, Line5, Line6, Line7, Line8);
2 output := LCD_REM_1.ENO
3 LCD_REM_1 := LCD_REM_1.LCD_REM
4 sts_lcd_rem := LCD_REM_1.Sts
```

LCD_REM_1

void LCD_REM_1(BOOL Enable, UDINT Font, STRING Line1, STRING Line2, STRING Line3, STRING Line4, STRING Line5, STRING Line6, STRING Line7, STRING Line8)
Type: LCD_REM, Display user strings on remote LCD when it is connected.

另请参见

[输入/输出指令](#) 参考页数 [381](#)

LCD_REM 状态代码

状态代码	说明
0	Enable 输入为 False。
1	成功显示用户消息。
2	未检测到远程 LCD。 出现以下情况时可能会发生： • 未将远程 LCD 物理连接到控制器（或配线错误）。 • 串行端口设置不符合远程 LCD 要求。
3	连接错误。 当存在内部状态机器错误时可能会发生。（可能是控制器固件版本和 RLCD 固件版本之间的不兼容引起的。
4	字体代码无效。
5-65535	已保留。

RHC

RHC 读取 Micro800™ 控制器中的高速时钟值。



参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，将字符串的一部分替换为新字符。 当 EN = FALSE 时，没有任何操作。
ENO	输出	BOOL	启用输出。

参数	参数类型	数据类型	说明
RHC	输出	UDINT	高速时钟的值。

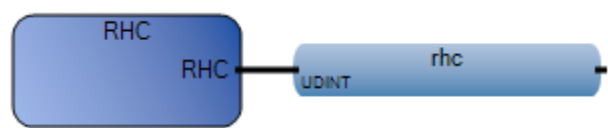
高速时钟分辨率

高速时钟指令的分辨率根据控制器类型有所变化：

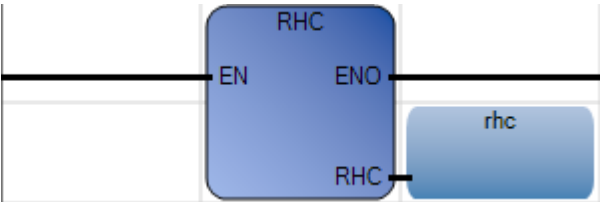
控制器类型	增量	时间基准	分辨率
Micro810	每 40 秒 4	10 秒	40 秒
Micro830	每 10 秒 1	10 秒	10 秒
Micro850	每 10 秒 1	10 秒	10 秒

RHC 函数语言示例

功能块图

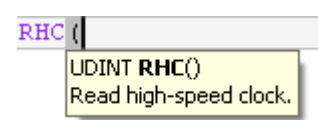


梯形图



结构化文本

```
1 | rhc := RHC();
```



(* 与之等效的 ST: *)

```
TESTOUTPUT2 := RHC();
```

另请参见

[输入/输出指令](#) 参考页数 [381](#)

RPC

RPC 从控制器或内存模块读取用户程序校验和。

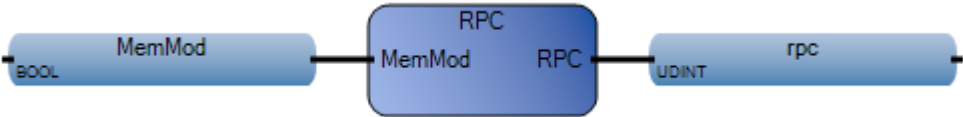


参数

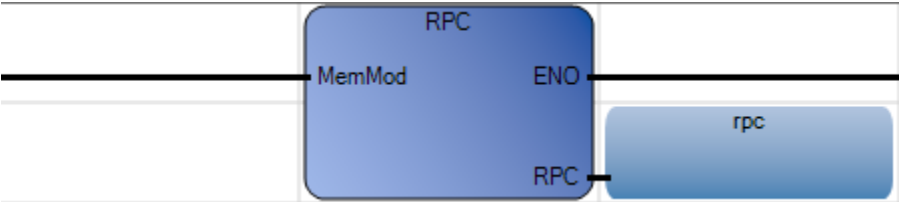
参数	参数类型	数据类型	说明
MemMod	输入	BOOL	如果为 TRUE，则值取自内存模块。 如果为 FALSE，则值取自 Micro800 控制器。
ENO	输出	BOOL	启用输出。
RPC	输出	UDINT	指定用户程序的校验和。

RPC 函数语言示例

功能块图

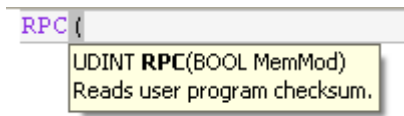


梯形图



结构化文本

```
1 MemMod := TRUE;  
2 rpc := RPC (MemMod);
```



(* 与之等效的 ST: *)

TESTOUTPUT2 := RPC(TESTINPUT);

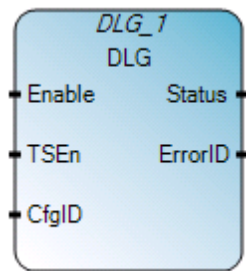
另请参见

[输入/输出指令](#) 参考页数 [381](#)

DLG

数据记录功能块可用于将变量值从运行时引擎写入到 SD 卡上的数据记录文件。

重要说明： 当写入到数据记录时，每天最多允许 50 个组文件夹。每个组文件夹最多可有 50 个文件大小为 4k-8k 的文件。



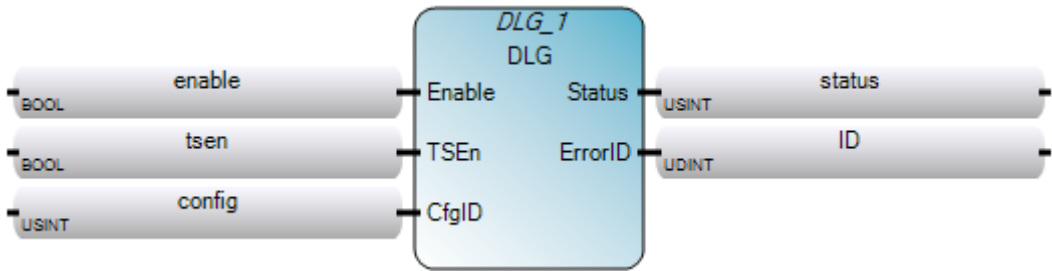
- DLG 操作**
- 只有 Micro820 控制器支持 DLG。

DLG 参数

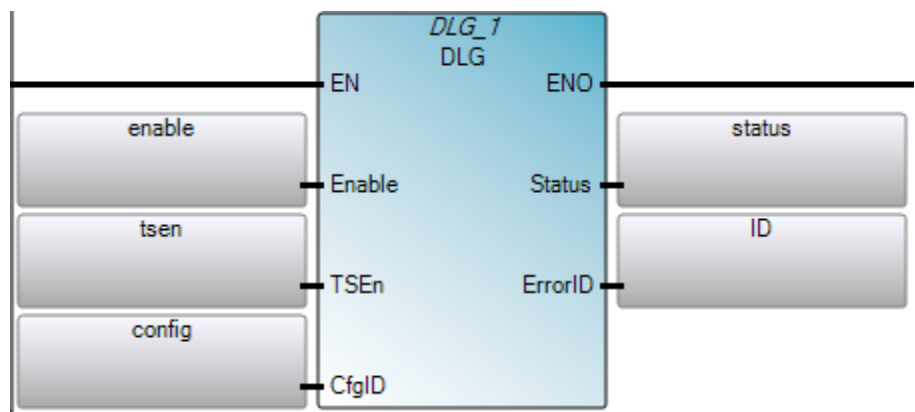
参数	参数类型	数据类型	说明
启用	输入	BOOL	数据记录写入启用。 当上升沿启用从“False”触发为“True”时，如果上一个 FB 操作已完成，则将执行数据记录功能块。
TSEnable	输入	BOOL	日期和时间戳记录启用标志。
CfgID	输入	USINT	数据记录配置 VA ID 编号，从 1-10。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
状态	输出	USINT	数据记录功能块当前状态。 请参见 DLG 状态代码 参考页数 397 。
ErrorID	输出	UDINT	FB 错误代码，请参阅数据记录 FB 错误 ID。 请参见 DLG 错误代码 参考页数 398 。

DLG 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 DLG_1 (EN, Enable, TSEn, CfgID);
2 output := DLG_1.ENO
3 status := DLG_1.Status
4 ID := DLG_1.ErrorID

DLG_1(
  void DLG_1(BOOL Enable, BOOL TSEn, USINT CfgID)
  Type: DLG, Save list of data to SD Card Data Log file.
```

另请参见

[输入/输出指令](#) 参考页数 [381](#)

DLG 状态代码

状态代码	说明
0	数据记录“闲置”状态。
1	数据记录“执行”状态。
2	数据记录已完成 -“成功”状态。
3	数据记录已完成 -“错误”状态。

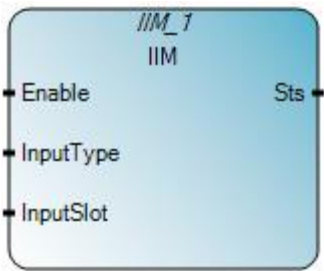
DLG 错误代码

下表描述了 DLG 错误代码。

错误代码	错误名称	注释
0	DLG_ERR_NONE	无错误。
1	DLG_ERR_NO_SDCARD	SD 卡不存在。
2	DLG_ERR_RESERVED	已保留。
3	DLG_ERR_DATAFILE_ACCE SS	访问数据记录文件错误。
4	DLG_ERR_CFG_ABSENT	数据记录配置文件不存在。
5	DLG_ERR_CFG_ID	数据记录配置文件中的配置 ID 不存在
6	DLG_ERR_RESOURCE_BUS Y	与此数据记录 ID 链接的数据 记录操作正在被另一个 FB 操 作使用。
7	DLG_ERR_CFG_FORMAT	数据记录配置文件格式无效。
8	DLG_ERR_RTC	实时时钟无效。
9	DLG_ERR_UNKNOWN	出现未指定的错误。

IIM

IIM 执行即时输入指令来更新输入数据，而不必等到下一次输入扫描开始。

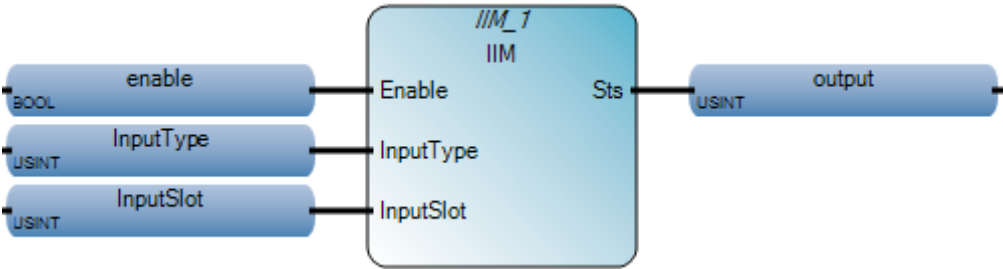


参数

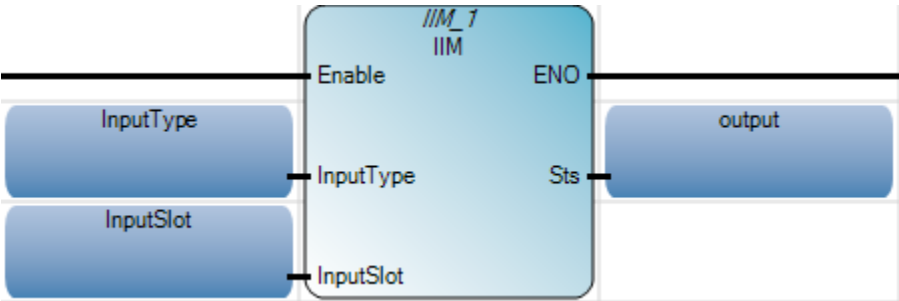
参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行函数。 当 Enable = FALSE 时，不执行函数。
InputType	输入	USINT	输入类型： 0 - 嵌入式输入。 1 - 插件输入。
InputSlot	输入	USINT	输入槽： 对于嵌入式输入，永远为 0。 对于插件输入，输入槽为 1、2、3、4、5（插件槽编号，从最左边的槽 [1] 开始）。
Sts	输出	USINT	即时输入扫描状态。 请参见 IIM 状态代码 参考页数 400 。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

IIM 功能块语言示例

功能块图 (FBD)

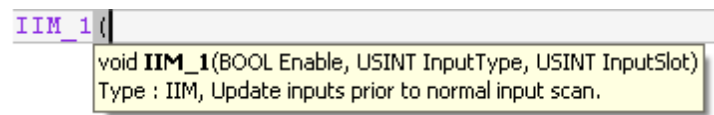


梯形图 (LD)

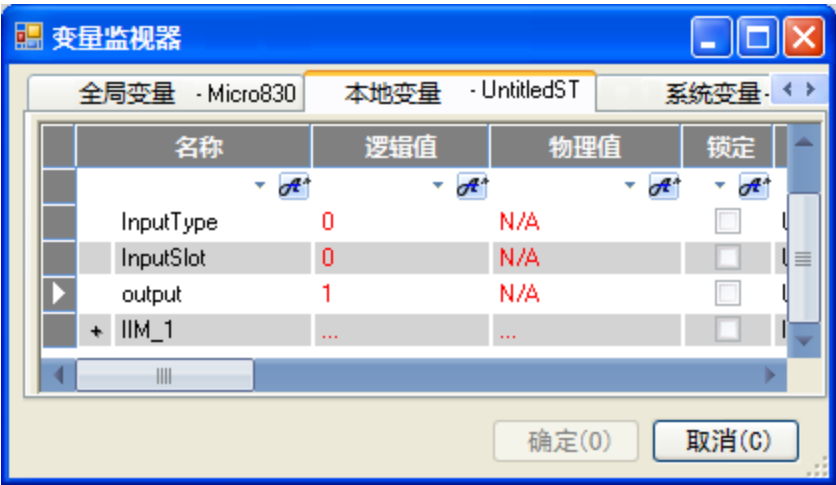


结构化文本 (ST)

```
1 enable := TRUE;  
2 InputType := 0;  
3 InputSlot := 0;  
4 IIM_1(enable, InputType, InputSlot);  
5 output := IIM_1.Sts;
```



结果



另请参见

[IIM 状态代码](#) 参考页数 [400](#)

[输入/输出指令](#) 参考页数 [381](#)

IIM 状态代码

下表描述了用于指示 IIM 功能块输入扫描状态的代码。

状态代码	说明
0x00	未启用（未执行操作）。
0x01	输入/输出扫描成功。

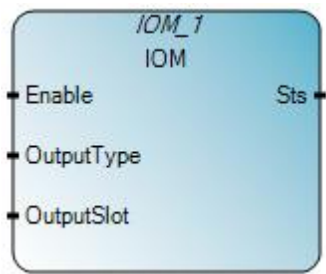
状态代码	说明
0x02	输入/输出类型无效。
0x03	输入/输出槽无效。

另请参见

[IIM](#) 参考页数 [398](#)

IOM

IOM 执行即时嵌入式输出数据更新，而无需等待自动输出扫描。

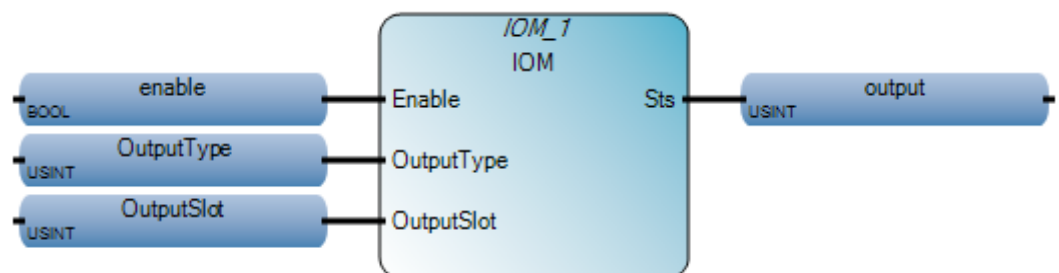


参数

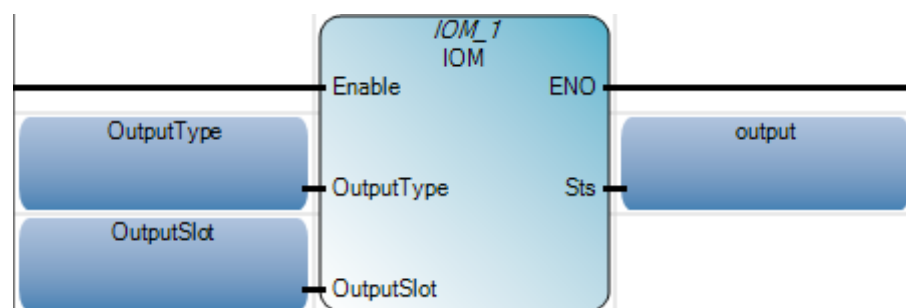
参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行函数。 当 Enable = FALSE 时，不执行函数。
OutputType	输入	USINT	输出类型： 0 - 嵌入式输出。 1 - 插件输出。
OutputSlot	输入	USINT	输出槽： 对于嵌入式输出，永远为 0。 对于插件输出，输出槽为 1、2、3、4、5（插件槽编号，从最左边的槽 [1] 开始）。
Sts	输出	USINT	即时输出扫描状态。 请参见 IOM 状态代码 参考页数 403 。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

IOM 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1  enable := TRUE;
2  OutputType := 0;
3  OutputSlot := 0;
4  IOM_1(enable, OutputType, OutputSlot);
5  output := IOM_1.Sts;

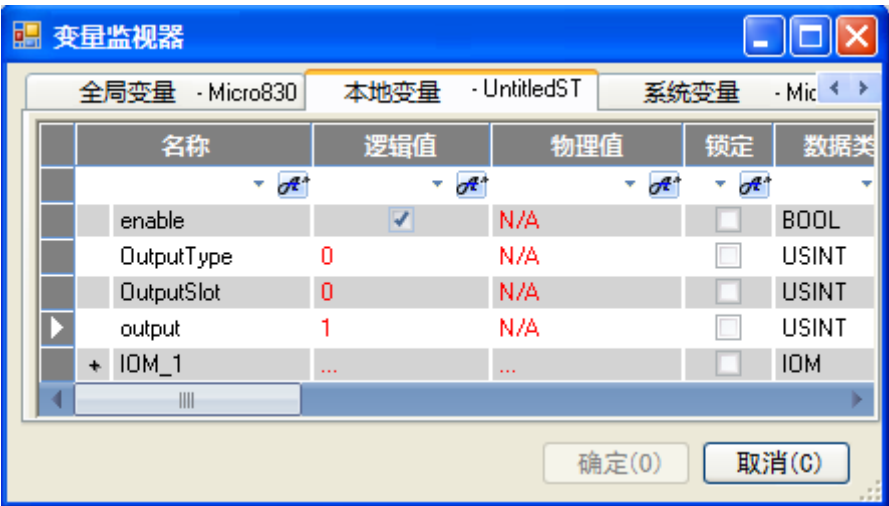
```

```

IOM_1(
  void IOM_1(BOOL Enable, USINT OutputType, USINT OutputSlot)
  Type : IOM, Update outputs prior to normal output scan.

```

结果



另请参见

[IOM 状态代码](#) 参考页数 [403](#)

[输入/输出指令](#) 参考页数 [381](#)

IOM 状态代码

下表描述了用于指示 IOM 功能块输出扫描状态的代码。

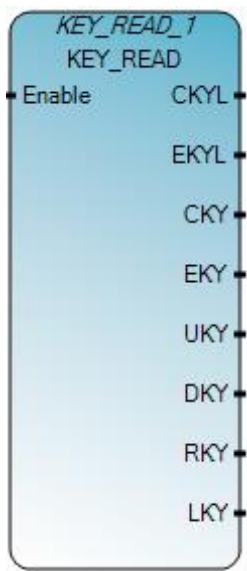
状态代码	说明
0x00	未启用（未执行操作）。
0x01	输入/输出扫描成功。
0x02	输入/输出类型无效。
0x03	输入/输出槽无效。

另请参见

[IOM](#) 参考页数 [401](#)

KEY_READ

KEY_READ 在用户显示处于活动状态时检查可选 LCD 模块上的键状态。这仅在 Micro810 上可用。

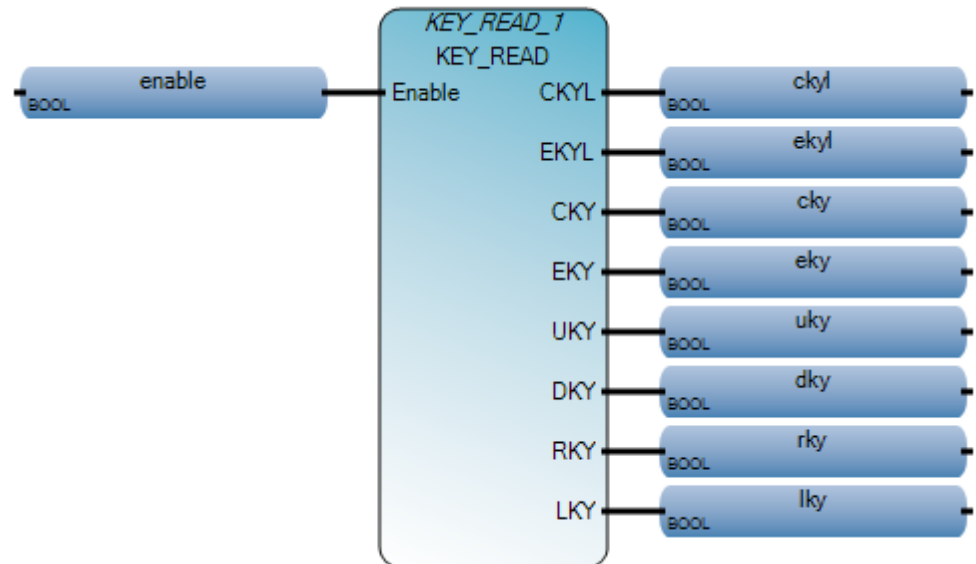


KEY_READ 操作

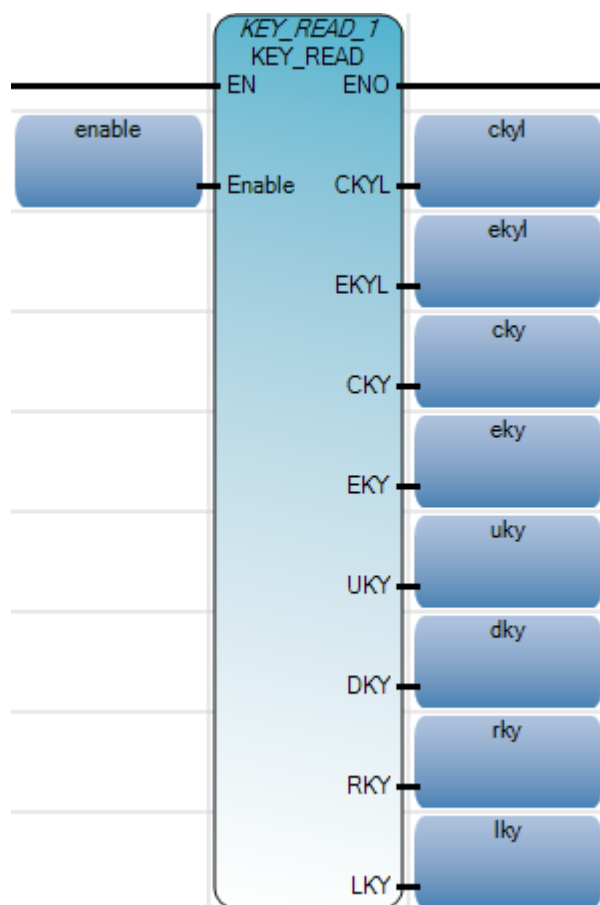
KEY_READ 功能块仅可用于 Micro810 控制器。

参数

参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行 LCD 读取。 当 Enable = FALSE 时，不执行读取操作。
CKYL	输出	BOOL	TRUE：按下 ESC 键持续 2 秒钟以上。
EKYL	输出	BOOL	TRUE：按下“确定”键持续 2 秒钟以上。
CKY	输出	BOOL	TRUE：按下 ESC 键。
EKY	输出	BOOL	TRUE：按下“确定”键。
UKY	输出	BOOL	TRUE：按下向上键。
DKY	输出	BOOL	TRUE：按下向下键。
LKY	输出	BOOL	TRUE：按下向左键。
RKY	输出	BOOL	TRUE：按下向右键。

KEY_READ 功能块语言示例**功能块图 (FBD)**

梯形图 (LD)



结构化文本 (ST)

```

1  KEY_READ_1(enable);
2  ckyl := KEY_READ_1.CKYL;
3  ekyl := KEY_READ_1.EKYL;
4  cky  := KEY_READ_1.CKY;
5  eky  := KEY_READ_1.EKY;
6  uky  := KEY_READ_1.UKY;
7  dky  := KEY_READ_1.DKY;
8  lky  := KEY_READ_1.LKY;
9  rky  := KEY_READ_1.RKY;

```

```
KEY_READ_1{
```

```
void KEY_READ_1(BOOL Enable)
Type : KEY_READ, Read key status on option LCD module.
```

(* 与之等效的 ST : *)

```

KEY_READ_1 (KEYENABLE) ;
KEY_EKYL := KEY_READ_1.EKYL ;
KEY_CKY := KEY_READ_1.CKY ;
KEY_EKY := KEY_READ_1.EKY ;
KEY_UKY := KEY_READ_1.UKY ;
KEY_DKY := KEY_READ_1.DKY ;
KEY_RKY := KEY_READ_1.RKY ;
KEY_LKY := KEY_READ_1.LKY ;

```

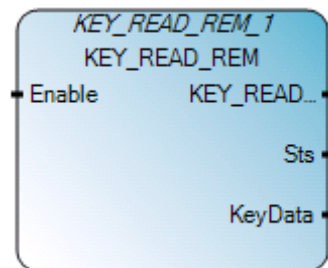
另请参见

[输入/输出指令](#) 参考页数 [381](#)

[LCD](#) 参考页数 [382](#)

KEY_READ_REM

KEY_READ_REM 在用户显示处于活动状态时检查远程 LCD 模块上的键状态。这仅在 Micro820 上可用。



KEY_READ_REM 操作

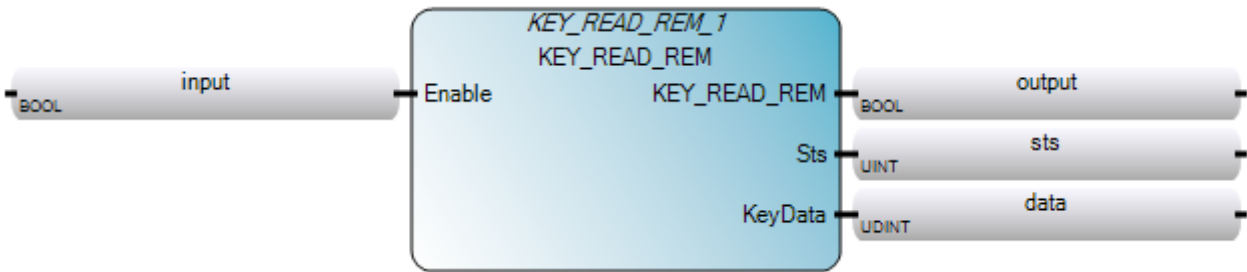
- 只有 Micro820 控制器支持 KEY_READ_REM。
- 当用户显示处于活动状态时(LCD_REM 指令用于激活用户显示), 可使用此功能块检查远程 LCD 模块上的键状态。如果用户显示未激活, KEY_READ_REM 指令将标记一个错误。
- 应激活 LCD 功能文件中的 P-BUTTON 属性 ; 否则所有键状态将为 FALSE。
- KEY_READ_REAM 指令只支持按单个键 ; 不支持组合按两个键。

KEY_READ_REM 参数

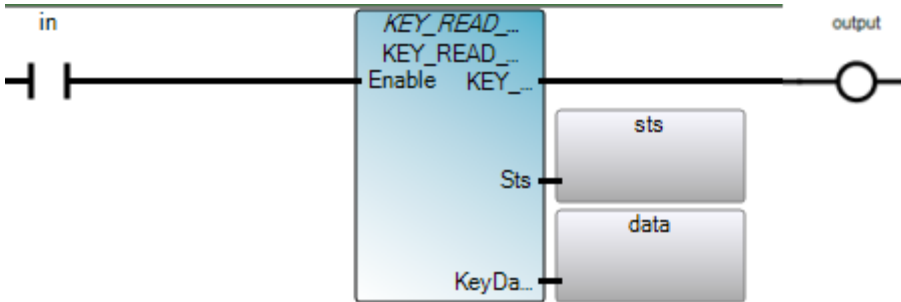
参数	参数类型	数据类型	说明
启用	输入	BOOL	已启用函数。 TRUE = Enable FALSE = Disable
KEY_READ_RE M	输出	BOOL	TRUE：已成功读取远程 LCD 键数据。 FALSE：启用为 False，则读取存在错误。 远程 LCD 键数据或用户显示未激活。
Sts	输出	UINT	KEY_READ_REM 操作的状态。 请参见 KEY_READ_REM 状态代码 参考页数 409。
键数据	输出	UDINT	远程 LCD 键盘数据。 请参见 键数据位字段表 参考页数 410。

KEY_READ_REM 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 KEY_READ_REM (Enable);
2 output := KEY_READ_REM.ENO
3 sts := KEY_READ_REM.Sts
4 data := KEY_READ_REM.KeyData
KEY_READ_REM_2{
void KEY_READ_REM_2(BOOL Enable)
Type: KEY_READ_REM, Check key status on remote LCD.
```

另请参见

[输入/输出指令](#) 参考页数 [381](#)

KEY_READ_REM 操作

- 只有 Micro820 控制器支持 KEY_READ_REM。
- 当用户显示处于活动状态时(LCD_REM 指令用于激活用户显示), 可使用此功能块检查远程 LCD 模块上的键状态。如果用户显示未激活 , KEY_READ_REM 指令将标记一个错误。
- 应激活 LCD 功能文件中的 P-BUTTON 属性 ;否则所有键状态将为 FALSE。
- KEY_READ_REAM 指令只支持按单个键 ;不支持组合按两个键。

KEY_READ_REM 状态代码

状态代码	说明
0	Enable 输入为 False。
1	已成功读取键数据。

状态代码	说明
2	未检测到远程 LCD。 出现以下情况时可能会发生： • 未将远程 LCD 物理连接到控制器(或配线错误)。 • 串行端口设置不符合远程 LCD 要求。
3	连接错误。 当存在内部状态机器错误时可能会发生。（可能是控制器固件版本和 RLCD 固件版本之间的不兼容引起的。）
4	用户显示未激活。
5-65535	已保留。

键数据位字段表

键数据中的位编号	名称	参数说明
0	UKY	TRUE = 按下向上键。
1	DKY	TRUE = 按下向下键。
2	LKY	TRUE = 按下向左键。
3	RKY	TRUE = 按下向右键。
4	F1KY	TRUE = 按下 F1 键。
5	F2KY	TRUE = 按下 F2 键。
6	F3KY	TRUE = 按下 F3 键。
7	F4KY	TRUE = 按下 F4 键。
8	F5KY	TRUE = 按下 F5 键。
9	F6KY	TRUE = 按下 F6 键。
10	EKY	TRUE = 按下 Enter 键。
11	CKY	TRUE = 按下 Cancel 键。
12	EKYL	TRUE = 按下 Enter 键持续 2 秒钟以上。

键数据中的位编号	名称	参数说明
13	CKYL	TRUE = 按下 Cancel 键持续 2 秒钟以上。
14-31	--	已保留。

MM_INFO

MM_INFO 检查内存模块信息。如果内存模块不存在，则所有值均变为零 (0)。

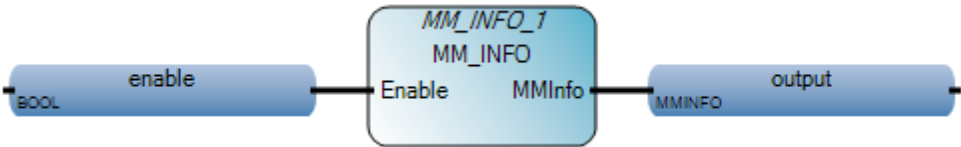


参数

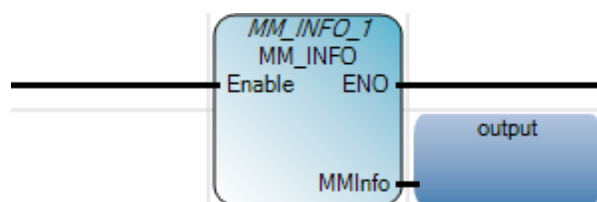
参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，读取内存模块标题信息。 当 Enable = FALSE 时，不执行读取操作，且输出内存模块信息无效。
MMInfo	输出	MMINFO	内存模块信息。 请参见 MMINFO 数据类型 参考页数 413。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

MM_INFO 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1 MM_INFO_1(enable);
2 output := MM_INFO_1.MMInfo;

```

MM_INFO_1(

void **MM_INFO_1**(BOOL Enable)
Type : MM_INFO, Read memory module header information.

结果



另请参见

[MMINFO 数据类型](#) 参考页数 [413](#)

[输入/输出指令](#) 参考页数 [381](#)

MMINFO 数据类型

下表描述了 MMINFO 数据类型参数。

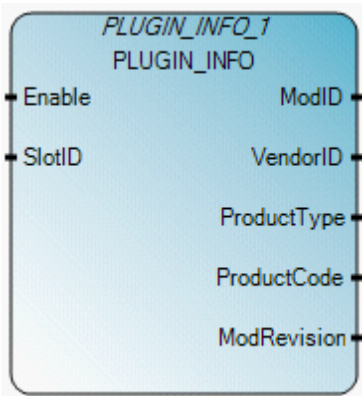
参数	数据类型	说明
MMCatalog	MMCATNUM	内存模块的类型编号。 注意：在带有 SD 卡的控制器上使用 MM_INFO 指令时，MMCatalog 为“SD CARD”。
系列	UINT	内存模块系列。 注意：在带有 SD 卡的控制器上使用 MM_INFO 指令时，系列为 0。
版本	UINT	内存模块版本。 注意：在带有 SD 卡的控制器上使用 MM_INFO 时，版本为 0。
UPValid	BOOL	用户程序有效（TRUE：有效）。
ModeBehavior	BOOL	模式行为（TRUE：通电后转至 RUN）。
LoadAlways	BOOL	每次打开电源时，内存模块总是恢复至控制器。
LoadOnError	BOOL	如果打开电源时出错，内存模块恢复至控制器。
FaultOverride	BOOL	打开电源时出现覆盖错误。
MMPresent	BOOL	内存模块存在。

另请参见

[MM_INFO](#) 参考页数 [411](#)

PLUGIN_INFO

PLUGIN_INFO 读取插件类属模块信息。它可读取任何插件模块信息（除 2080-MEMBAK-RTC 模块外）。如果插件类属模块不存在，则所有值恢复为零 (0)。

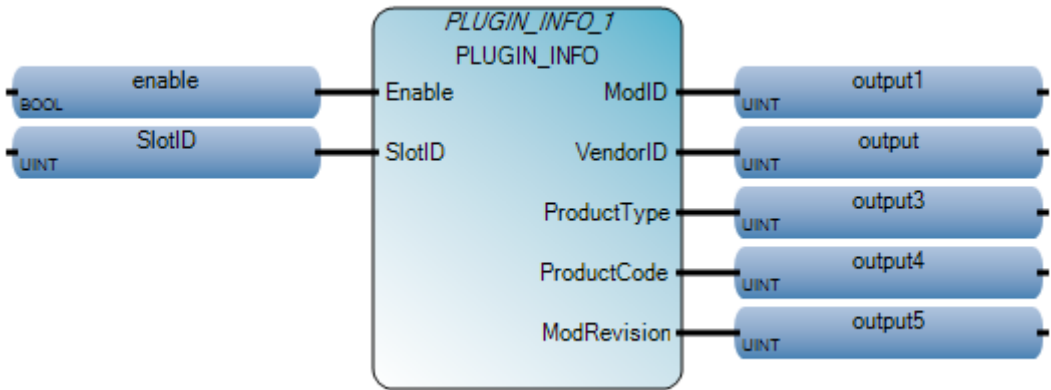


参数

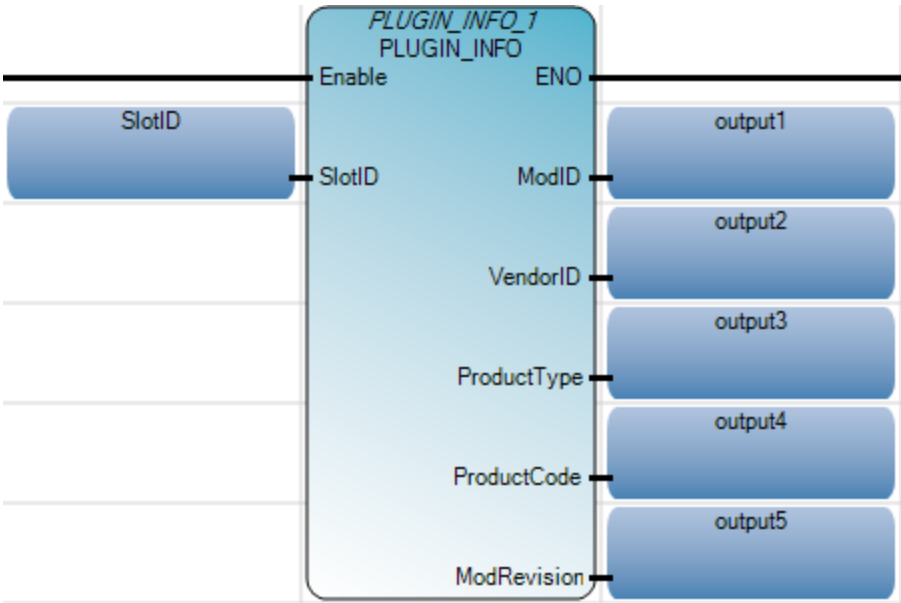
参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行 P_IM 信息读取。 当 Enable = FALSE 时，不执行该功能块。所有输出数据值重置为 0。
SlotID	输入	UINT	插件插槽编号。 插槽 ID = 1、2、3、4、5（从最左侧的插槽开始，第一个插槽的 ID = 1）。
ModID	输出	UINT	插件类属模块物理 ID。
VendorID	输出	UINT	插件类属模块供应商 ID。 对于 Allen Bradley 产品，供应商 ID = 1。
ProductType	输出	UINT	插件类属模块产品类型。
ProductCode	输出	UINT	插件类属模块产品代码。
ModRevision	输出	UINT	插件类属模块版本信息。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

PLUGIN_INFO 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 SlotID := 1;  
2 PLUGIN_INFO_1(enable, SlotID);  
3 output1 := PLUGIN_INFO_1.ModID;  
4 output2 := PLUGIN_INFO_1.VendorID;  
5 output3 := PLUGIN_INFO_1.ProductType;  
6 output4 := PLUGIN_INFO_1.ProductCode;  
7 output5 := PLUGIN_INFO_1.ModRevision;
```

PLUGIN_INFO_1{

void **PLUGIN_INFO_1**(BOOL Enable, UINT SlotID)
Type : PLUGIN_INFO, Get module information from a generic plug-in module.

结果

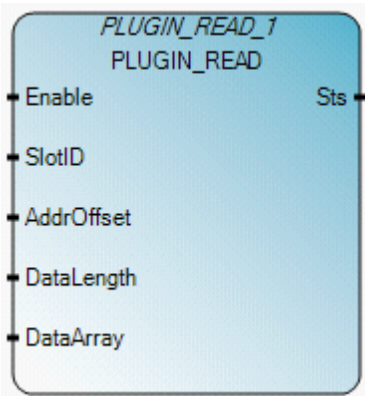


另请参见

[输入/输出指令](#) 参考页数 381

PLUGIN_READ

PLUGIN_READ 可从任何插件类属模块硬件（除 2080-MEMBAK-RTC 模块外）中读取大量数据。如果插件类属模块不存在，则所有值恢复为零 (0)。

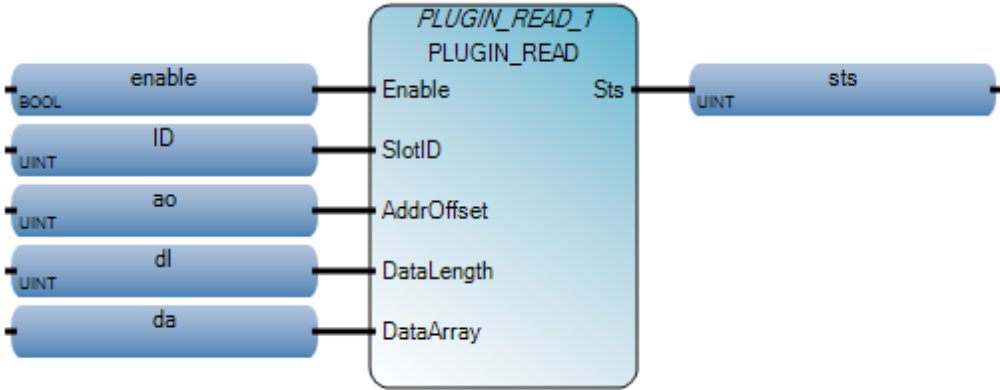


参数

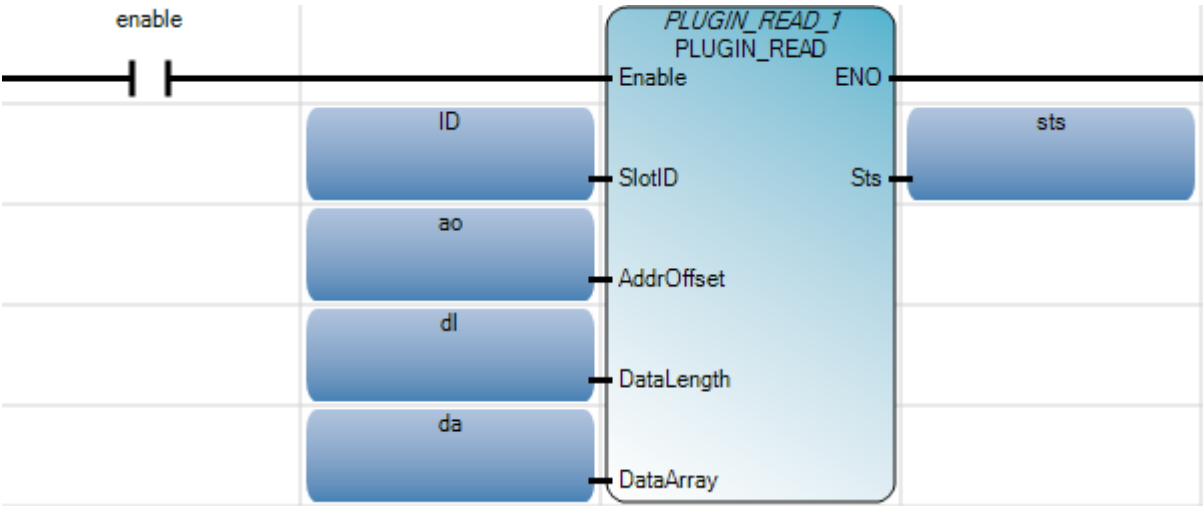
参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行 UPM 读取。 当 Enable = FALSE 时，不进行读取操作，并且数据数组中的数据无效。
SlotID	输入	UINT	插件插槽编号。 插槽 ID = 1、2、3、4、5（从最左侧的插槽开始，第一个插槽的 ID = 1）。
Offset	输入	UINT	从插件类属模块的首个字节中计算，要读取的首个数据的地址偏移。
DataLength	输入	UINT	要读取的字节数量。
DataArray	输入	USINT	用于存储从插件类属模块中读取的数据的数组。
Sts	输出	UINT	请参见 PLUGIN_READ 状态代码 参考页数 419。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

PLUGIN_READ 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1  PLUGIN_READ_1(enable, ID, ao, dl, da);
2  sts := PLUGIN_READ_1.Sts;
```

```
PLUGIN_READ_1 {
void PLUGIN_READ_1(BOOL Enable, UINT SlotID, UINT AddrOffset, UINT DataLength, USINT[1..1] DataArray, UINT __ADI_DataArray)
Type : PLUGIN_READ, Read data from a generic PLUGIN module.
```

另请参见

[PLUGIN_READ 状态代码](#) 参考页数 [419](#)

[输入/输出指令](#) 参考页数 [381](#)

PLUGIN_READ 状态代码

下表描述了 PLUGIN_READ 功能块的状态代码。

状态代码	状态说明
0x00	未启用功能块（无操作）。
0x01	插件操作成功。
0x02	由于插槽 ID 无效，插件操作失败。
0x03	由于插件类属模块无效，插件操作失败。
0x04	由于数据操作超出范围，插件操作失败。
0x05	由于出现数据访问奇偶性错误，插件操作失败。

另请参见

[PLUGIN_READ](#) 参考页数 [417](#)

PLUGIN_RESET

PLUGIN_RESET 可重置任何插件类属模块硬件（除 2080-MEMBAK-RTC 模块外）。重置硬件后，插件类属模块即可进行配置和操作。



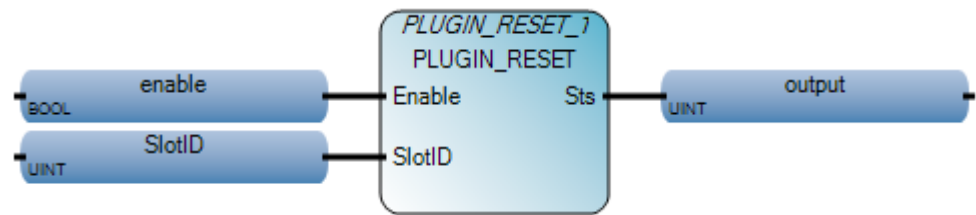
参数

参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行插件重置。 当 Enable = FALSE 时，不进行重置操作。

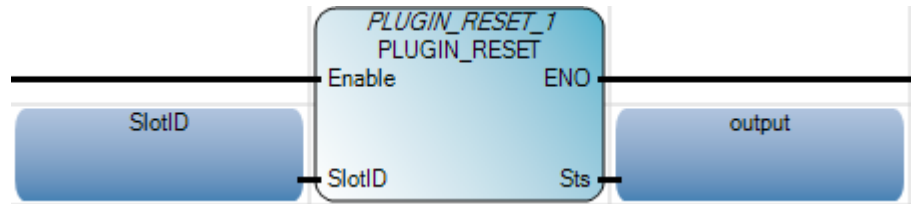
参数	参数类型	数据类型	说明
SlotID	输入	UINT	插件插槽编号。 插槽 ID = 1、2、3、4、5 (从最左侧的插槽开始，第一个插槽的 ID = 1) 。
Sts	输出	UINT	请参见 PLUGIN_READ 状态代码 参考页数 419 。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

PLUGIN_RESET 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 SlotID := 1;
2 PLUGIN_RESET_1(enable, SlotID);
3 output := PLUGIN_RESET_1.Sts;
```

`PLUGIN_RESET_1 {`

`void PLUGIN_RESET_1(BOOL Enable, UINT SlotID)``Type : PLUGIN_RESET, Reset a generic PLUGIN module(hardware reset).`

结果



另请参见

[输入/输出指令](#) 参考页数 [381](#)

PLUGIN_WRITE

PLUGIN_WRITE 可将大量数据写入任何插件类属模块硬件（除 2080-MEMBAK-RTC 模块外）中。



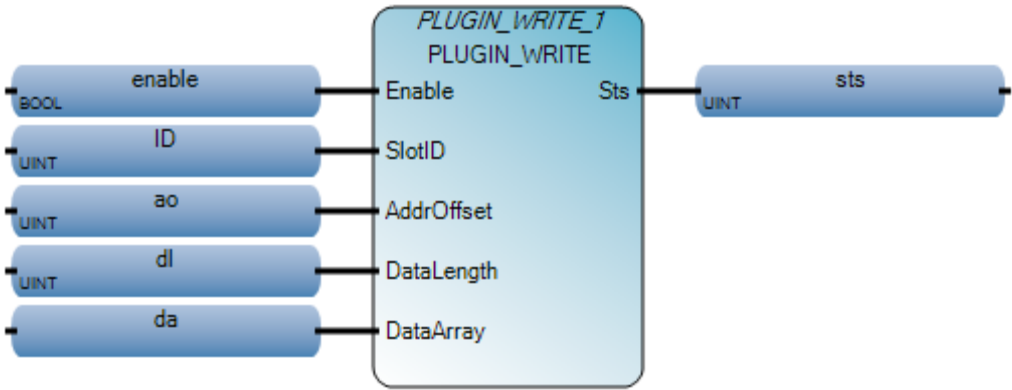
参数

参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行插件写入。 当 Enable = FALSE 时，不进行数据写入操作。

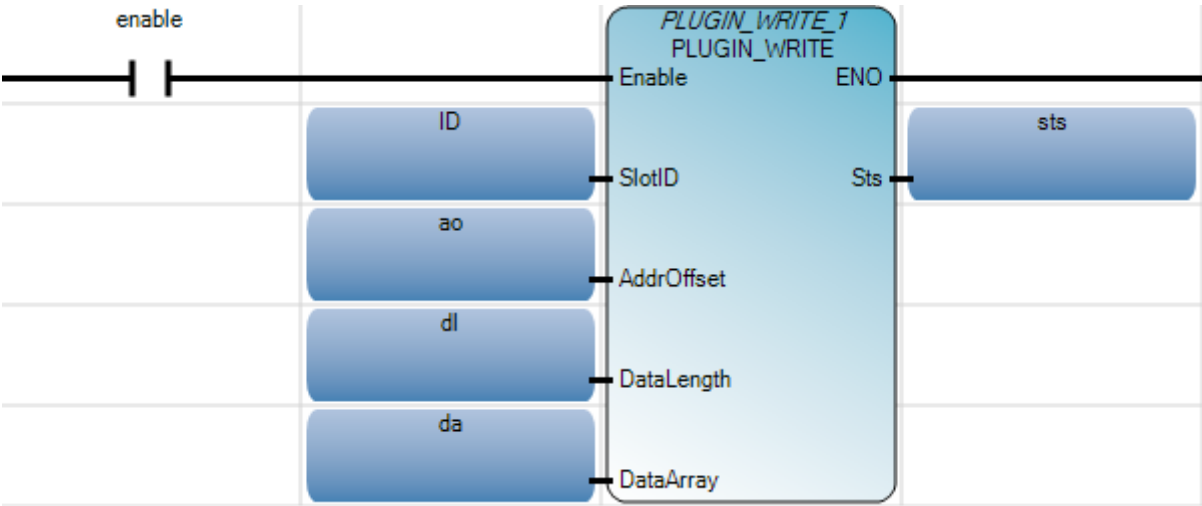
参数	参数类型	数据类型	说明
SlotID	输入	UINT	插件插槽编号。 插槽 ID = 1、2、3、4、5 (从最左侧的插槽开始，第一个插槽的 ID = 1) 。
Offset	输入	UINT	从插件类属模块的首个字节中计算，要写入的首个数据的地址偏移。
DataLength	输入	UINT	要写入的字节数量。
DataArray	输入	USINT	要写入插件类属模块的数据。
Sts	输出	UINT	请参见 PLUGIN_READ 状态代码 参考页数 419 。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

PLUGIN_WRITE 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 | PLUGIN_WRITE_1(enable, ID, ao, dl, da);
2 | sts := PLUGIN_WRITE_1.Sts;
```

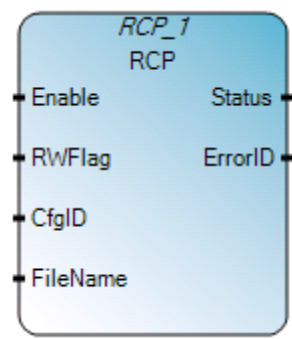
```
PLUGIN_WRITE_1(  
void PLUGIN_WRITE_1(BOOL Enable, UINT SlotID, UINT AddrOffset, UINT DataLength, USINT[1..1] DataArray, UINT __ADI_DataArray)  
Type : PLUGIN_WRITE, Write data to a generic PLUGIN module.
```

另请参见

[输入/输出指令](#) 参考页数 [381](#)

RCP

配方 C 功能块可用于从 SD 卡配方数据文件夹中的配方数据文件读取变量的数据值，并将值更新到运行时引擎。配方 C 功能块可用于通过运行时引擎将变量值写入到 SD 卡中的配方数据文件。



RCP 操作

- 只有 Micro820 控制器支持 RCP。

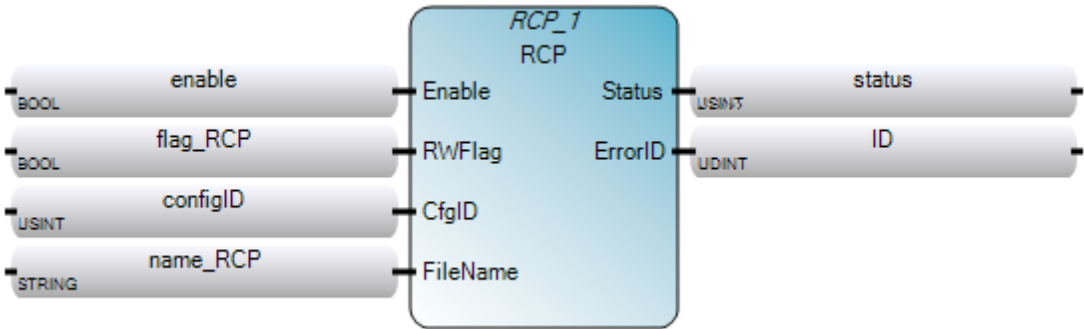
RCP 参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 RCP 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
启用	输入	BOOL	功能块读取/写入启用。 在上升沿（从 0 切换为 1 时启用）时，执行功能块，前提是上一操作已完成。
RWFlag	输入	BOOL	TRUE = 写入到 SD 卡。 FALSE = 从 SD 卡读取配方。
CfgID	输入	USINT	配方配置 VA ID 索引。
RcpName	输入	STRING	在 SD 卡中的配方文件夹中运行的配方数据文件名称（最大长度为 30 个字符）。
状态	输出	USINT	配方功能块当前状态。 请参见 RCP 状态代码 参考页数 426。
ErrorID	输出	UDINT	RCP 读取/写入出现故障时的错误 ID。 请参见 RCP 错误代码 参考页数 426。

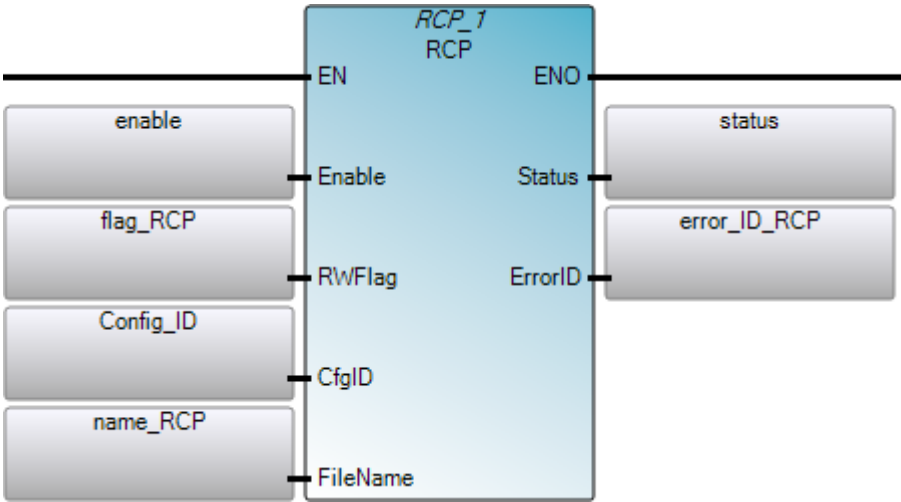
参数	参数类型	数据类型	说明
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

RCP 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 RCP_1 (EN, Enable, RWFlag, CfgID, FileName);
2 output := RCP_1.ENO
3 status := RCP_1.Status
4 error_ID_RCP := RCP_1.ErrorID
```

RCP_1(
void RCP_1(BOOL Enable, BOOL RWFlag, USINT CfgID, STRING FileName)
Type : RCP, Save/Restore list of data to/from SD Card Recipe file.

另请参见

[输入/输出指令](#) 参考页数 [381](#)

RCP 状态代码

状态代码	说明
0	配方“闲置”状态。
1	配方“执行”状态。
2	配方已完成 -“成功”状态。
3	配方已完成 -“错误”状态。

RCP 错误代码

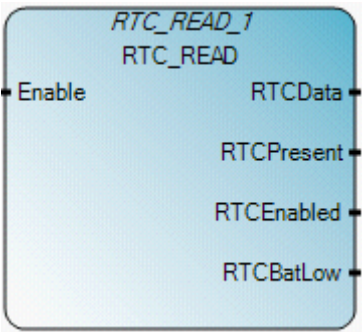
下表描述了 RCP 错误代码。

错误代码	错误名称
0	RCP_ERR_NONE
1	RCP_ERR_NO_SDCARD
2	RCP_ERR_DATAFILE_FULL
3	RCP_ERR_DATAFILE_ACCESS SD 卡可能被标识为 a) 已损坏；b) 已满 ；或 c) 只读。

错误代码	错误名称
4	RCP_ERR_CFG_ABSENT
5	RCP_ERR_CFG_ID
6	RCP_ERR_RESOURCE_BUSY
7	RCP_ERR_CFG_FORMAT
8	RCP_ERR_RESERVED 保留，以用于将来可能的扩展。
9	RCP_ERR_UNKNOWN
10	RCP_ERR_DATAFILE_NAME
11	RCP_ERR_DATAFOLDER_INVALID
12	RCP_ERR_DATAFILE_ABSENT
13	RCP_ERR_DATAFILE_FORMAT
14	RCP_ERR_DATAFILE_SIZE 配方数据文件大小太大 (>4kb)。

RTC_READ

RTC_READ 可读取 RTC 预置和 RTC 信息。



RTC_READ 操作

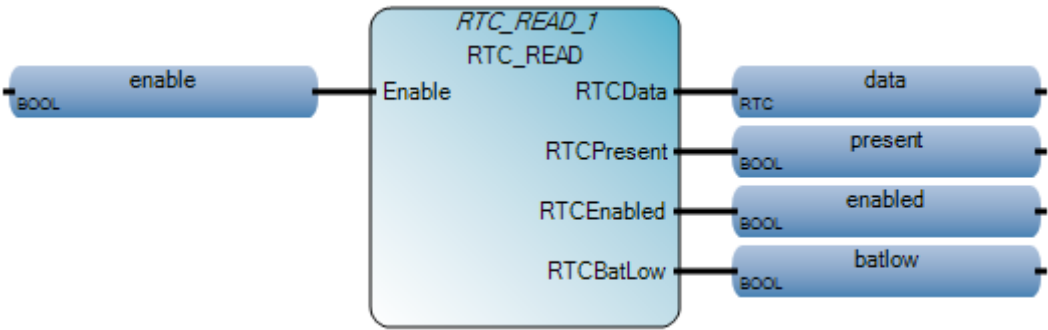
- 当与采用嵌入式 RTC 的 Micro810 控制器一起使用时，RTCBatLow 始终设置为零 (0)。
- 如果由于失去电源而使嵌入式 RTC 丢失其电荷/内存，则 RTCEnabled 始终设置为零 (0)。

参数

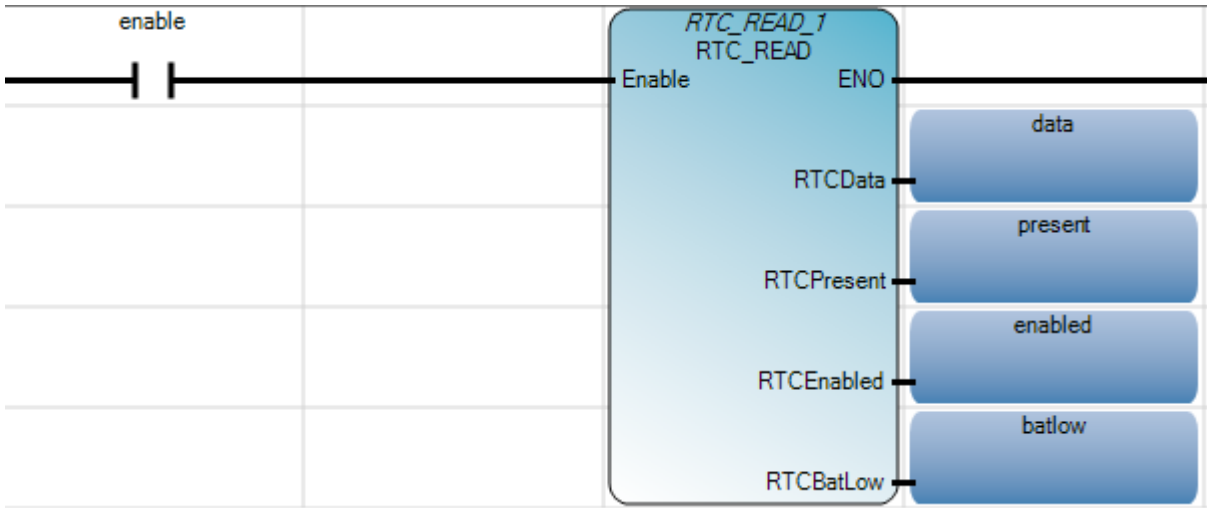
参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行 RTC 信息读取。 当 Enable = FALSE 时，不进行读取操作，并且 RTC 输出数据无效。
RTCData	输出	RTC	RTC 数据信息：yy/mm/dd、hh/mm/ss、星期。 请参见 RTC 数据类型 参考页数 429。
RTCPresent	输出	BOOL	TRUE - 已插入 RTC 硬件。 FALSE - 未插入 RTC 硬件。
RTCEnabled	输出	BOOL	TRUE - 已启用 RTC 硬件（计时）。 FALSE - 已禁用 RTC 硬件（未计时）。
RTCBatLow	输出	BOOL	TRUE - RTC 电池电量低。 FALSE - RTC 电池电量不低。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

RTC_READ 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本

```
1  RTC_READ_1(enable);
2  data := RTC_READ_1.RTCData;
3  present := RTC_READ_1.RTCPresent;
4  enabled := RTC_READ_1.RTCEnabled;
5  batlow := RTC_READ_1.RTCBatLow;
```

```
RTC_READ_1(|
void RTC_READ_1(BOOL Enable)
Type : RTC_READ, Read RTC module information.
```

另请参见

[RTC 数据类型](#) 参考页数 [429](#)

[输入/输出指令](#) 参考页数 [381](#)

RTC 数据类型

下表描述了 RTC 数据类型。

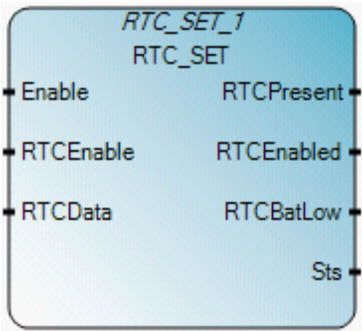
参数	数据类型	说明
Year	UINT	RTC 的年设置。16 位值，有效范围为 2000 (1 月 01 日 00:00:00) 到 2098 (12 月 31 日 23:59:59)
Month	UINT	RTC 的月设置。
Day	UINT	RTC 的日设置。
Hour	UINT	RTC 的小时设置。
Minute	UINT	RTC 的分钟设置。
Second	UINT	RTC 的秒钟设置。
DayOfWeek	UINT	RTC 的星期设置。RTC_SET 将忽略此参数。

另请参见

[RTC_READ](#) 参考页数 [427](#)

RTC_SET

RTC_SET 可设置 RTC 状态或写入 RTC 信息。



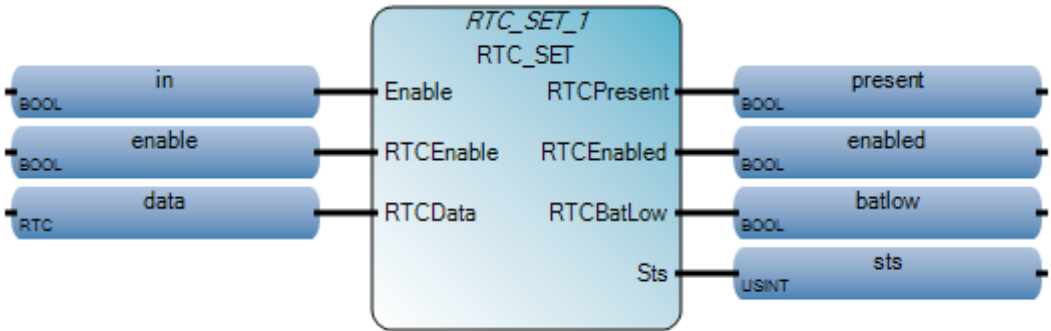
参数

参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，使用输入的 RTC 信息执行 RTC 设置。 当 Enable = FALSE 时，不进行读取操作，并且 RTC 输出数据无效。
RTCEnable	输入	BOOL	TRUE - 启用 RTC 及指定的 RTC 数据。 FALSE - 禁用 RTC。 注意：Micro810 控制器会忽略此参数。

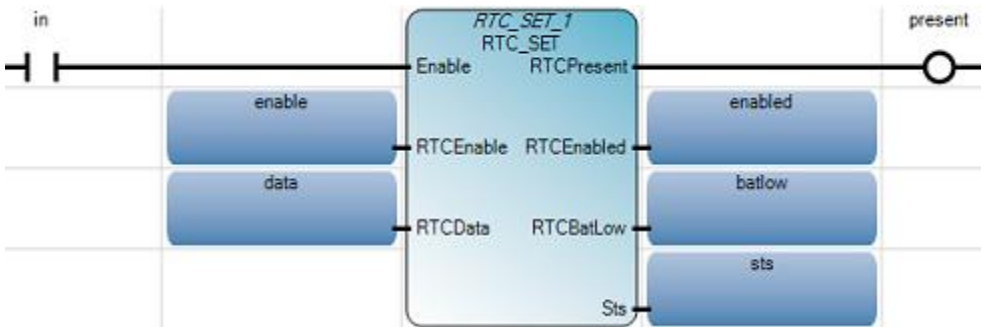
参数	参数类型	数据类型	说明
RTCData	输入	RTC	RTC 数据信息：yy/mm/dd、hh/mm/ss、星期。 当 RTCEnable = 0 时，则忽略此 RTC 数据。 请参见 RTC 数据类型 参考页数 429。
RTCPresent	输出	BOOL	TRUE - 已插入 RTC 硬件。 FALSE - 未插入 RTC 硬件。
RTCEnabled	输出	BOOL	TRUE - 已启用 RTC 硬件（计时）。 FALSE - 已禁用 RTC 硬件（未计时）。
RTCBatLow	输出	BOOL	TRUE - RTC 电池电量低。 FALSE - RTC 电池电量不低。
Sts	输出	USINT	读取操作状态。 请参见 RTC 设置状态值 参考页数 432

RTC_SET 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1  RTC_SET_1(in, enable, data);
2  present := RTC_SET_1.RTCPresent;
3  enabled := RTC_SET_1.RTCEnabled;
4  batlow := RTC_SET_1.RTCBatLow;
5  sts := RTC_SET_1.Sts;
```

RTC_SET_1{

void RTC_SET_1(BOOL Enable, BOOL RTCEnable, RTC RTCDData)
Type : RTC_SET, Set RTC data to RTC module.

另请参见

[RTC 数据类型](#) 参考页数 [429](#)

[输入/输出指令](#) 参考页数 [381](#)

RTC 设置状态值

下表描述了 RTCSet 值:

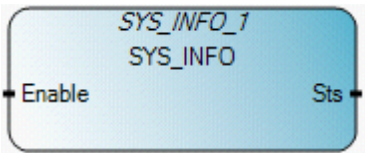
状态值	状态说明
0x00	未启用功能块（无操作）。
0x01	RTC 设置操作成功。
0x02	RTC 设置操作失败。

另请参见

[RTC_SET](#) 参考页数 [430](#)

SYS_INFO

SYS_INFO 读取状态数据块。

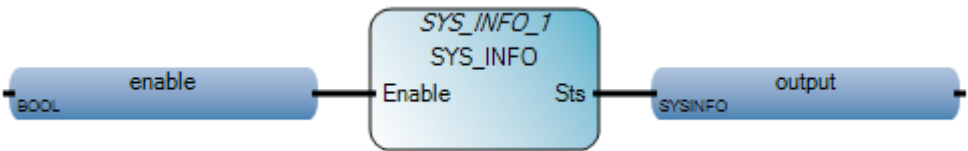


参数

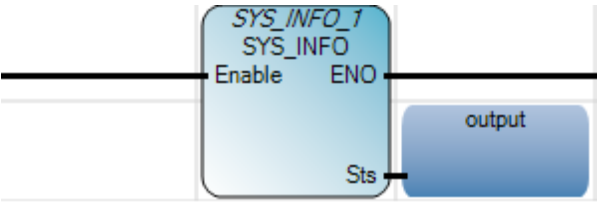
参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行函数。 当 Enable = FALSE 时，不执行函数。
Sts	输出	SYSINFO	系统状态数据块。 请参见 SYS_INFO 数据类型 参考页数 434。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

SYS_INFO 功能块语言示例

功能块图 (FBD)



梯形图 (LD)

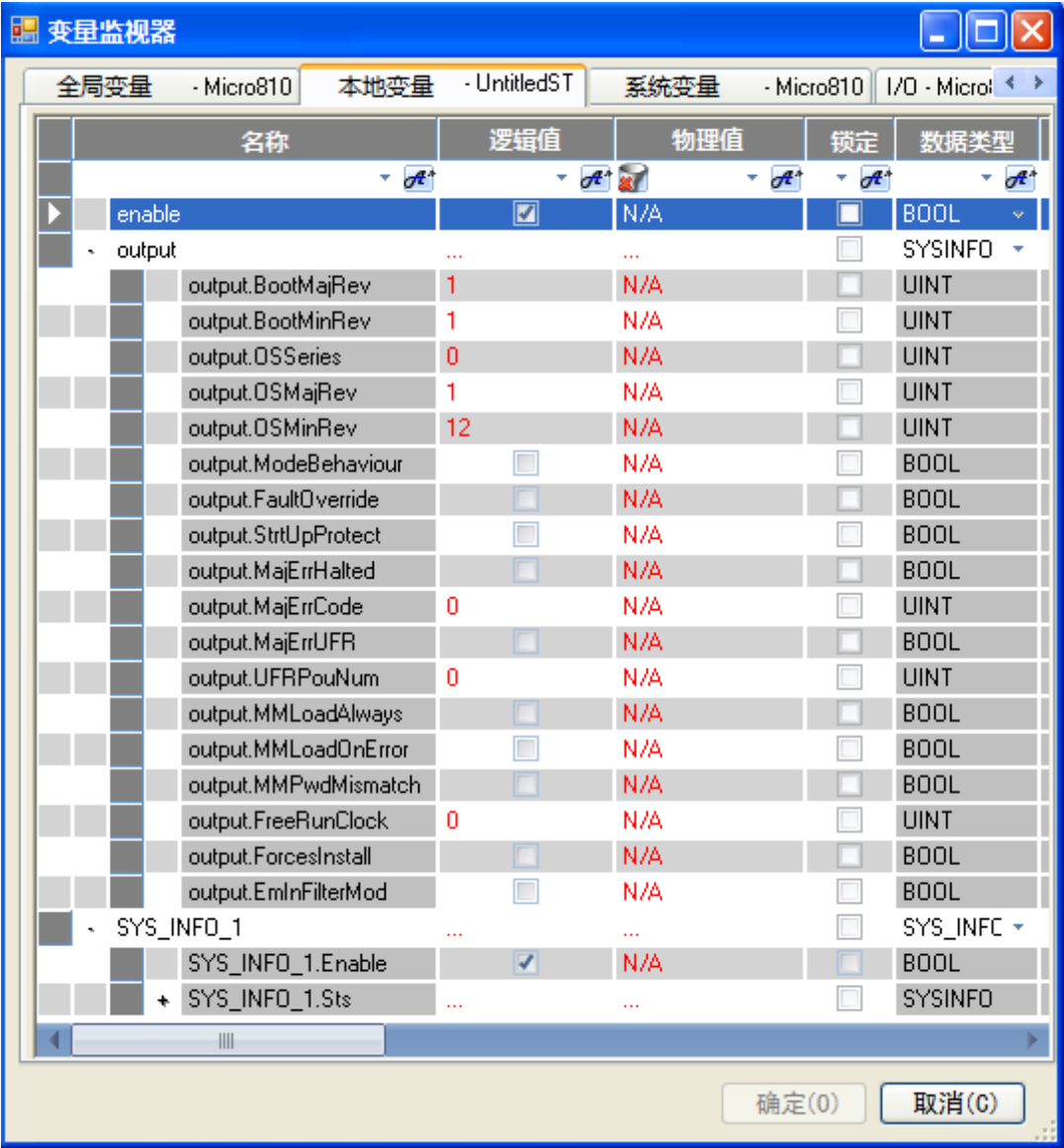


结构化文本 (ST)

```
1 SYS_INFO_1(enable);  
2 output := SYS_INFO_1.Sts;
```

```
SYS_INFO_1(  
void SYS_INFO_1(BOOL Enable)  
Type : SYS_INFO, Read Micro800 system status.
```

结果



另请参见

[输入/输出指令](#) 参考页数 [381](#)

SYS_INFO 数据类型

下表描述了 SYSINFO 数据类型。

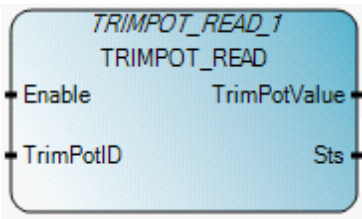
参数	数据类型	说明
BootMajRev	UINT	启动主要版本。
BootMinRev	UINT	启动次要版本。
操作系统系列	UINT	操作系统系列： 0 表示一系列 A 设备 1 表示一系列 B 设备
OSMajRev	UINT	操作系统主要版本。
OSMinRev	UINT	操作系统次要版本。
ModeBehaviour	BOOL	模式行为（TRUE：通电后转至 RUN）。
FaultOverride	BOOL	故障覆盖（TRUE：通电后覆盖错误）。
StrtUpProtect	BOOL	启动保护（TRUE：通电后运行启动保护程序）。 提示： 对于未来版本
MajErrHalted	BOOL	主要错误停止（TRUE：主要错误停止）。
MajErrCode	UINT	主要错误代码。
MajErrUFR	BOOL	用户错误例程期间的主要错误。 提示： 对于未来版本
UFRPouNum	UINT	用户错误例程程序编号。
MMLoadAlways	BOOL	通电后内存模块始终恢复到控制器（TRUE：已恢复）。
MMLoadOnError	BOOL	如果通电后发生错误，则内存模块恢复到控制器（TRUE：已恢复）。
MMPwdMismatch	BOOL	内存模块密码不匹配（TRUE：控制器与内存模块密码不匹配）。
FreeRunClock	UINT	从 0 到 65535 每 100 微秒递增一个数字，然后恢复为 0 的自由运行时钟。如果您需要比标准 1 毫秒计时器更高的分辨率，则可以使用可全局访问的时钟。 提示： 仅支持 Micro830 和 Micro850 控制器。Micro810 控制器的值保持为 0。
ForcesInstall	BOOL	启用强制（TRUE：已启用）。
EMINFilterMod	BOOL	修改的嵌入式过滤器（TRUE：已修改）。

另请参见

[SYS_INFO](#) 参考页数 [432](#)

TRIMPOT_READ

TRIMPOT_READ 读取一个微调电位当前值。

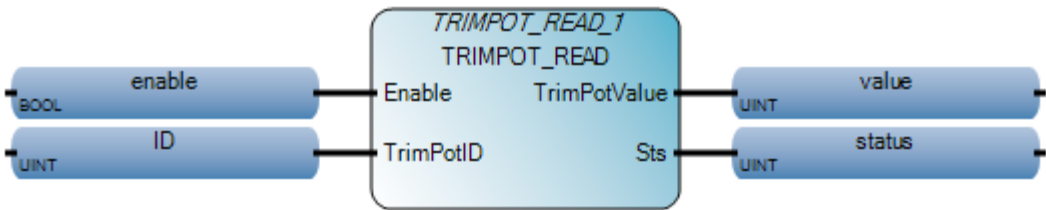


参数

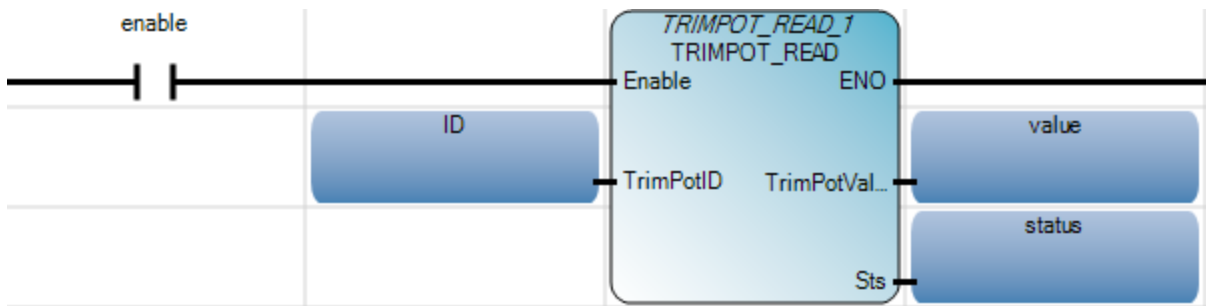
参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行微调电位读取。 当 Enable = FALSE 时，没有读取操作，且输出微调电位值无效。
TrimPotID	输入	UINT	要读取的微调电位的 ID。 请参见 微调电位 ID 定义 参考页数 437。
TrimPotValue	输出	UINT	当前微调电位值。
Sts	输出	UINT	读取操作状态。 请参见 微调电位操作状态值 参考页数 438。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

TRIMPOT 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 TRIMPOT_READ_1(enable, ID);
2 value := TRIMPOT_READ_1.TrimPotValue;
3 status := TRIMPOT_READ_1.Sts;
```

```
TRIMPOT_READ_1(  
void TRIMPOT_READ_1(BOOL Enable, UINT TrimPotID)  
Type : TRIMPOT_READ, Read the Trimpot value from a specific Trimpot.
```

另请参见

[输入/输出指令](#) 参考页数 [381](#)

微调电位 ID 定义

下表描述了微调电位 ID 定义。

输出选择	位	说明
微调电位 ID 定义	15 - 13	微调电位的模块类型： • 0x00 - 嵌入式。 • 0x01 - 扩展。 • 0x02 - 插件端口。

输出选择	位	说明
	12 - 8	模块的插槽 ID： • 0x00 - 嵌入式。 • 0x01-0x1F - 扩展模块的 ID。 • 0x01-0x05 - 插件端口的 ID。
	7 - 4	微调电位类型： • 0x00 - 保留。 • 0x01 - 数字微调电位类型 1 (LCD 模块 1)。 • 0x02 - 机械微调电位模块 1。
	3 - 0	模块内的微调电位 ID： • 0x00-0x0F - 嵌入式。 • 0x00-0x07 - 用于扩展的微调电位的 ID。 • 0x00-0x07 - 用于插件端口的微调电位的 ID。 微调电位 ID 从 0 开始。

另请参见

[TRIMPOT_READ](#) 参考页数 [436](#)

微调电位操作状态值

下表描述了微调电位操作状态值。

状态值	状态说明
0x00	未启用功能块 (无读取/写入操作)。
0x01	读取/写入操作成功。
0x02	读取/写入操作由于微调电位 ID 无效而失败。
0x03	写入操作由于超范围值而失败。

另请参见

[TRIMPOT_READ](#) 参考页数 [436](#)

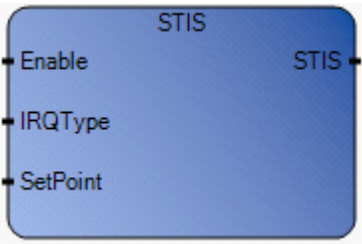
中断指令

中断指令用于向处理器发出信号，指示有事件需要注意。通常，中断信号用于需要中断处理器当前所执行代码的高优先级条件。

函数	说明
STIS 参考页数 439	从控制程序启动 STI 计时器，而不是自动启动
UIC 参考页数 441	清除特定用户中断
UID 参考页数 443	禁用特定用户中断
UIE 参考页数 445	启用特定用户输入
UIF 参考页数 447	刷新特定用户输入

STIS

STIS 启动可选择计时（计时器）用户中断。



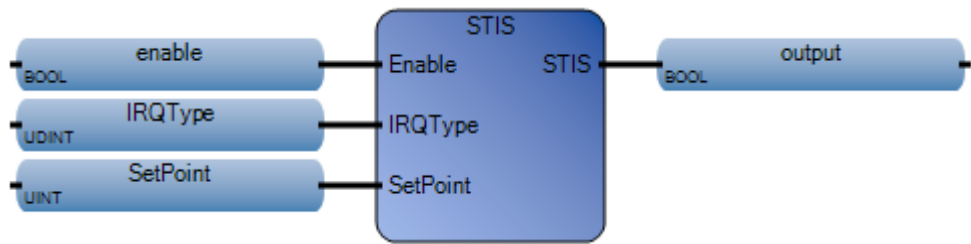
参数

参数	参数类型	数据类型	说明
启用	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行该函数。 当 Enable = FALSE 时，不执行该函数。

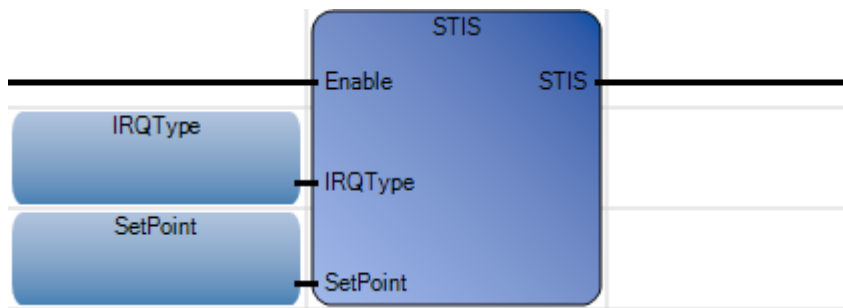
参数	参数类型	数据类型	说明
IRQType	输入	UDINT	使用 STI 定义字。 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3
SetPoint	输入	UINT	这是执行可选择计时中断之前必须过期的时间(单位为毫秒)。 值为 0 则禁用 STIS 函数。值介于 1 和 65535 之间则启用 STIS 函数。
STIS	输出	BOOL	Rung 状态 (与 Enable 相同) 。

STIS 函数语言示例

功能块图

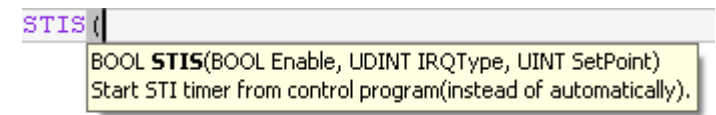


梯形图



结构化文本

```
1  enable := TRUE;  
2  IRQType := IRQ_STI2;  
3  SetPoint := 1000;  
4  output := STIS(enable, IRQType, SetPoint);
```



(* 与之等效的 ST : *)

TESTOUTPUT := STIS(TESTENABLE, 2, 1000);

结果



另请参见

[中断指令](#) 参考页数 [439](#)

UIC

UIC 清除选定用户中断的中断丢失位。

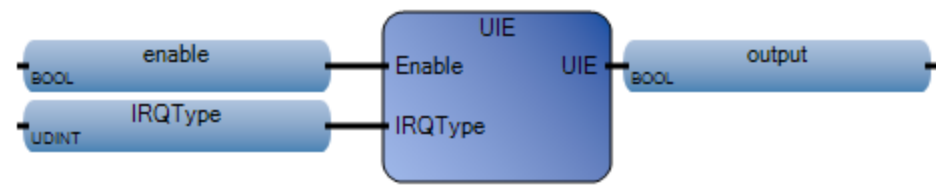


参数

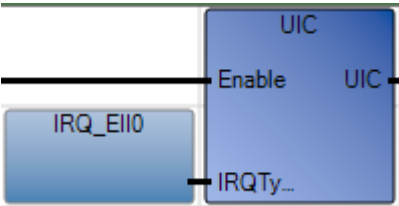
参数	参数类型	数据类型	说明	
启用	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行该函数。 当 Enable = FALSE 时，不执行该函数。	
IRQType	输入	UDINT	使用 STI 定义字。 - IRQ_EII0 - IRQ_EII1 - IRQ_EII2 - IRQ_EII3 - IRQ_EII4 - IRQ_EII5 - IRQ_EII6 - IRQ_EII7 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2	- IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UIC	输出	BOOL	Rung 状态 (与 Enable 相同) 。	

UIC 函数语言示例

功能块图



梯形图



结构文本

```
1 enable := TRUE;  
2 IRQType := 2;  
3 output := UIC (enable, IRQType);
```

结果

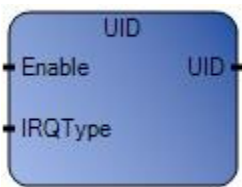
用户全局变量 - Micro850		局部变量 - Prog3		系统变量- Micro8	
	名称	初始值	数据类型		
	enable		BOOL		读
	output		BOOL		读
	IRQType	2	UINT		读
*					

另请参见

[中断指令](#) 参考页数 [439](#)

UID

UID 禁用选定的用户中断。



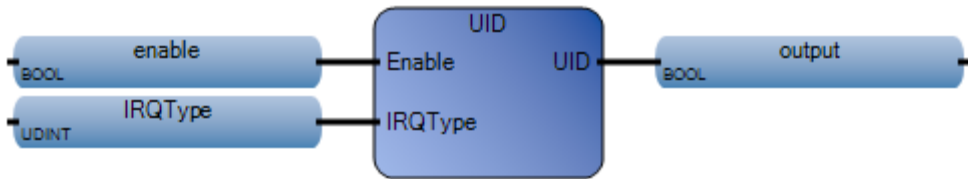
参数

参数	参数类型	数据类型	说明
启用	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行该函数。 当 Enable = FALSE 时，不执行该函数。

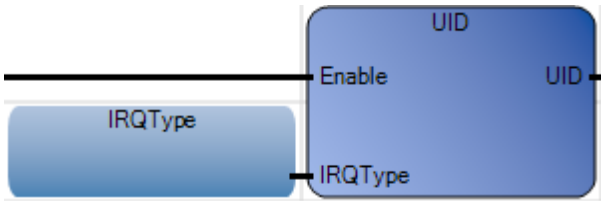
参数	参数类型	数据类型	说明	
IRQType	输入	UDINT	使用 STI 定义字。 - IRQ_EI0 - IRQ_EI1 - IRQ_EI2 - IRQ_EI3 - IRQ_EI4 - IRQ_EI5 - IRQ_EI6 - IRQ_EI7 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2	- IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UID	输出	BOOL	Rung 状态 (与 Enable 相同) 。	

UID 函数语言示例

功能块图

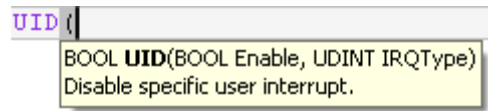


梯形图



结构化文本

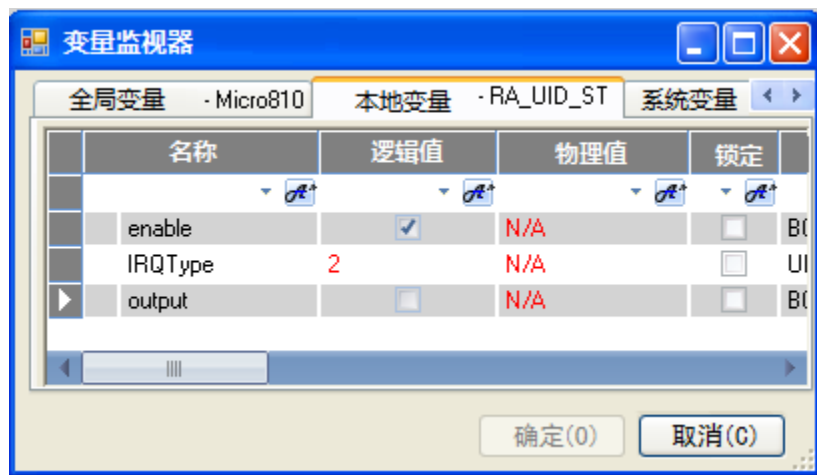
```
1 enable := TRUE;  
2 IRQType := 2;  
3 output := UID(enable, IRQType);
```

(* 与之等效的 ST : *)

TESTOUTPUT := UID(TESTENABLE, 2);

结果

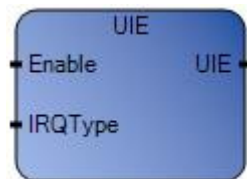


另请参见

[中断指令](#) 参考页数 [439](#)

UIE

UIE 启用用户中断。

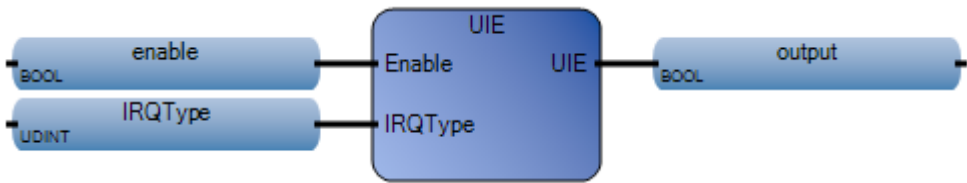


参数

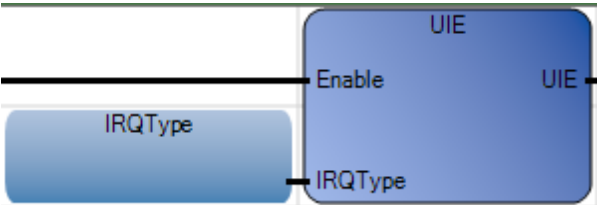
参数	参数类型	数据类型	说明	
启用	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行该函数。 当 Enable = FALSE 时，不执行该函数。	
IRQType	输入	UDINT	使用 STI 定义字。 - IRQ_EI0 - IRQ_EI1 - IRQ_EI2 - IRQ_EI3 - IRQ_EI4 - IRQ_EI5 - IRQ_EI6 - IRQ_EI7 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2	- IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UIE	输出	BOOL	Rung 状态 (与 Enable 相同) 。	

UIE 函数语言示例

功能块图



梯形图



结构化文本

```

1 enable := TRUE;
2 IRQType := 2;
3 output := UIF(enable, IRQType);

```

UIF {
 BOOL **UIF**(BOOL Enable, UDINT IRQType)
 Enable specific user interrupt.

(* 与之等效的 ST : *)

TESTOUTPUT := UIF(TESTENABLE, 2);

结果



另请参见

[中断指令](#) 参考页数 [439](#)

UIF

UIF 刷新 (删除) 选定用户中断的挂起用户中断。

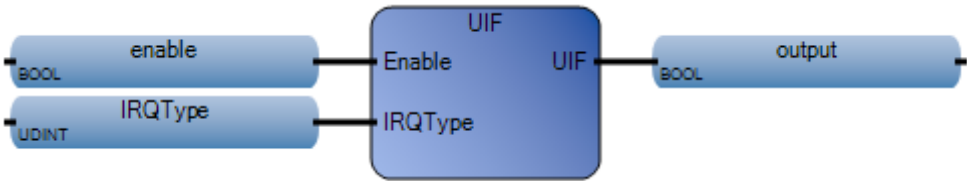


参数

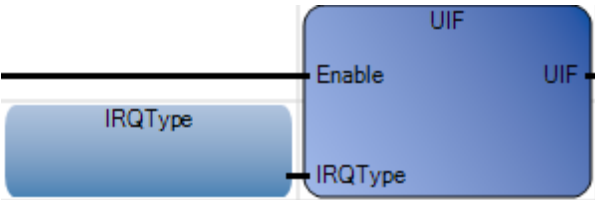
参数	参数类型	数据类型	说明
启用	输入	BOOL	函数启用。 当 Enable = TRUE 时，执行该函数。 当 Enable = FALSE 时，不执行该函数。
IRQType	输入	UDINT	使用 STI 定义字。 - IRQ_EI10 - IRQ_EI11 - IRQ_EI12 - IRQ_EI13 - IRQ_EI14 - IRQ_EI15 - IRQ_EI16 - IRQ_EI17 - IRQ_HSC0 - IRQ_HSC1 - IRQ_HSC2 - IRQ_HSC3 - IRQ_HSC4 - IRQ_HSC5 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3 - IRQ_UFR - IRQ_UPM0 - IRQ_UPM1 - IRQ_UPM2 - IRQ_UPM3 - IRQ_UPM4
UIF	输出	BOOL	Rung 状态 (与 Enable 相同) 。

UIF 函数语言示例

功能块图



梯形图

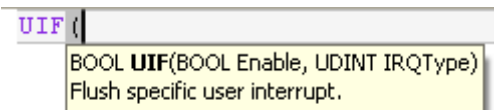


结构化文本

```

1 enable := TRUE;
2 IRQType := 2;
3 output := UIF(enable, IRQType);

```



(* 与之等效的 ST : *)

TESTOUTPUT := UIF(TESTENABLE, 2);

结果



另请参见

[中断指令](#) 参考页数 [439](#)

运动控制指令

在 Connected Components Workbench 2.0 及更高版本中，一组指令使用运动控制功能块对特定轴的运动进行编程和设计。

- 提示** 管理功能块支持 PTO 运动和反馈运动。
- ： 支持 FBAxis 的运动控制功能块有：MC_ReadActualPosition 和 MC_ReadActualVelocity。

功能块	说明
管理	
MC_AbortTrigger 参考页数 466	中止连接到触发事件的功能块。
MC_Power 参考页数 495	控制功率（打开或关闭）
MC_ReadAxisError 参考页数 503	介绍一般轴错误
MC_ReadBoolParameter 参考页数 509	返回特定于供应商的类型为 BOOL 的参数的值
MC_ReadParameter 参考页数 512	返回特定于供应商的参数的值
MC_ReadStatus 参考页数 516	返回轴的状态
MC_Reset 参考页数 522	重置所有内部轴相关的错误
MC_SetPosition 参考页数 525	通过控制实际位置来转移轴坐标系统
MC_TouchProbe 参考页数 533	在触发事件中记录轴位置
MC_WriteBoolParameter 参考页数 539	修改特定于供应商的类型为 BOOL 的参数的值
MC_WriteParameter 参考页数 543	修改特定于供应商的参数的值
运动	
MC_Halt 参考页数 470	在正常操作条件下停止轴
MC_Home 参考页数 474	命令轴执行 <search home> 序列
MC_MoveAbsolute 参考页数 478	命令受控制的运动到指定的绝对位置
MC_MoveRelative 参考页数 484	在执行时命令与实际位置相对的指定距离的受控制运动。
MC_MoveVelocity 参考页数 490	以指定速率命令从未结束的受控制运动。

功能块	说明
MC_ReadActualPosition 参考页数 499	返回 FBAxis 的实际位置 注意：在 CCW 8 或更高版本中可用
MC_ReadActualVelocity 参考页数 501	返回 FBAxis 实际速率 注意：在 CCW 8 或更高版本中可用
MC_Stop 参考页数 529	命令受控制的运动停止

运动控制功能块的一般规则

Micro800 运动控制功能块的一般规则遵从了 PLCopen 运动控制规范：下表提供有关运动控制功能块界面的一般规则。

规则适用于	规则
输入参数	借助 Execute：这些参数与执行输入的上升沿一起使用。若要修改任何参数，请更改输入参数并再次触发运动。 注意：如果功能块的实例在其完成之前接收到新的 Execute（作为相同实例上的一系列命令），新的 Execute 将被忽略，并且先前发布的指令将继续执行。 借助 Enable：这些参数与启用输入的上升沿一起使用，并能不断修改。
缺少输入参数	在用户应用程序编译期间将捕获缺少输入情况。没有在控制器级别处理的缺少输入错误。
输入超出应用程序限制	如果功能块使用会导致违反应用程序限制的参数来命令，则功能块的实例将生成错误。在此情况下，错误输出将被标记为“开启”，并且输出 ErrorID 将指出错误信息。在大部分情况下控制器将保持运行模式，并且运动错误也不会报告为严重的控制器故障。
输入的符号规则	Acceleration、Deceleration 和 Jerk 输入始终为正值。Velocity、Position 和 Distance 输入可以为正值和负值。
Position 与 Distance	Position 是在轴坐标系统内定义的一个值。Distance 是与技术单元相关的相对指标。Distance 是两个位置间的差异。
Position/Distance 输入	Micro800 控制器中仅支持线性运动。对于 MC_MoveAbsolute 功能块，Position 输入是要命令到轴中的绝对位置。对于 MC_MoveRelative，距离输入是从当前位置开始的相对位置（将当前轴位置视为 0）。
Velocity 输入	Velocity 可以是带符号的值，也可以使用 Direction 输入定义速率的符号（负速率 x 负方向 = 正速率）。出于兼容性考虑，E 参数“Direction”引用速率输入和输出。

规则适用于	规则
Direction 输入	对于距离（位置）运动，定义了目标位置（绝对或相对），运动方向唯一。距离移动的方向输入将忽略。 对于速率运动，方向输入值可以是 1（正方向）、0（当前方向）或 -1（负方向）。对于任何其他值，仅考虑符号。例如，-3 表示负方向，+2 表示正方向，等等。 对于速率移动 (MC_MoveVelocity)，符号（速率 x 方向）将确定当值不为 0 时的实际运动方向。例如，如果速率 x 方向 = +300，则方向为正。
Acceleration、Deceleration 和 Jerk 输入	- Deceleration 或 Acceleration 输入应为正值。如果 Deceleration 或 Acceleration 设置为非正值，则功能块会报告错误（错误 ID：MC_FB_ERR_RANGE）。 - Jerk 输入应为非负值。如果 Jerk 设为负值，则功能块会报告错误（错误 ID：MC_FB_ERR_RANGE）。 - 如果 Jerk 最大值设为零，则运动控制功能块的所有加加速度参数（包括 MC_Stop 的加加速度设置）必须设为零。否则，功能块会报告错误（错误 ID：MC_FB_ERR_RANGE）。 - 如果 Jerk 设为非零值，系统将生成 S-Curve 配置；如果 Jerk 设置为 0，系统将生成 trapezoidal 配置。 - 归位加加速度配置不仅限于最大加加速度配置。 - 如果运动引擎无法生成动态输入参数预设的运动配置，则功能块会报告错误（错误 ID：MC_FB_ERR_PROFILE）。
输出排斥性	借助 Execute：当 Execute 为 TRUE 时，Busy、Done、Error 或 CommandAborted 输出必须也为 TRUE。这些输出互斥：一个功能块上只有一个输出可以为 TRUE。一次只能设置输出 Active、Error、Done 和 CommandAborted 中的一个。 借助 Enable：Valid 和 Error 输出互斥：一个功能块上只有一个输出可以为 TRUE。
输出状态	借助 Execute：使用 Execute 指令的下降沿重置 Done、Error、ErrorID 和 CommandAborted 输出。但是，Execute 的下降沿不会停止也不会影响实际功能块的执行。即使在功能块完成之前重置 Execute，相应的输出也会至少设置一次循环。 如果功能块的实例在其完成之前接收到新的 Execute 命令（作为相同实例上的一系列命令），新的 Execute 命令将被忽略，并且先前发布的指令将继续执行。 借助 Enable：使用 Enable 的下降沿重置 Valid、Enabled、Busy、Error 和 ErrorID 输出。
Done 输出的行为	当命令的操作已成功完成时将设置 Done 输出。 当多个功能块按顺序在同一轴上运行时，以下适用： 如果轴上的一个运动被同一轴上的其他运动中断，而并未达到最终目标，则在第一个功能块上将不会设置 Done。

规则适用于	规则
Busy 输出的行为	每个功能块可具有 Busy 输出，表示功能块尚未完成（对于使用 Execute 输入的功能块）或并非正在运行，并且需要新的输出值（在 Enable 输入的情况下）。 Busy 会在 Execute 的上升沿进行设置，并在设置 Done、Aborted 或 Error 等输出中的一个时进行重置。至少在 Busy 为 TRUE 之前，应该将此功能块保留在应用程序的活动回路中，因为输出可能会发生变化。 除非不再忙碌，否则带有相同实例的忙碌功能块无法执行。带有不同实例的功能块可以中止当前执行的功能块。
CommandAborted 输出的行为	当命令的运动被其他运动命令中断时，将设置 CommandAborted 输出。 CommandAborted 输出的重置行为与 Done 输出的重置行为类似。当出现 CommandAborted 时，将重置其他输出符号（例如 InVelocity）。
输出 Active	在缓冲的功能块上需要 Active 输出且在功能块控制了所依据轴时进行设置。 对于非缓冲模式，Active 和 Busy 输出可具有相同值。
Enable 和 Valid 状态	Enable 输入与 Valid 输出成对出现。Enable 对层级敏感，而 Valid 将显示功能块上一组有效输出可用。 只要有效输出值可用且 Enable 输入为 TRUE，Valid 输出即为 TRUE。只要输入 Enable 为 TRUE，即可刷新相关输出值。 如果出现功能块错误，输出将无效（将 Valid 设置为 FALSE）。当错误条件消失时，值将重新显示并再次设置 Valid 输出。
输出错误处理	用于定义错误的输出 所有块均具有以下两个输出，可用于处理执行期间发生的错误： • Error - “Error”的上升沿通知在功能块的执行期间发生了错误。 • ErrorID - 错误编号。 注意：Done 和 InVelocity 输出用于成功完成，因此这些输出在逻辑上不包括 Error。实例错误不会始终导致轴错误（将轴置于 ErrorStop 状态）。 如何重置错误输出 • 可使用 Execute 和 Enable 的下降沿重置相关功能块的错误输出。 • 对于使用 Enable 的功能块，可以在操作期间重置该功能块的错误输出，而不必重置 Enable。

规则适用于	规则
	错误类型 • 功能块逻辑（例如，参数超出范围、状态机尝试冲突等等） • 硬件限制或软件限制 • 机制/电机 • 驱动器
命名规范 ENUM 类型	由于 IEC 标准中对变量名称唯一性的命名限制，因此 ENUM 使用了对 PLCopen 运动控制命名空间的“mc”引用。 因此，我们避免使用 ENUM 类型“正值”和“负值”作为变量的实例与整个其余项目中的名称相冲突，因为它们分别称为 mcPositive 和 mcNegative。

另请参见

[运动控制指令](#) 参考页数 [451](#)

运动控制功能块参数 详细信息

以下主题提供了与所有运动控制功能块有关的运动控制参数的详细信息。

[运动控制轴状态](#) 参考页数 [455](#)

[运动控制功能块参数编号](#) 参考页数 [458](#)

[运动控制功能块错误 ID](#) 参考页数 [460](#)

[轴出错场景](#) 参考页数 [462](#)

[AXIS REF 数据类型](#) 参考页数 [462](#)

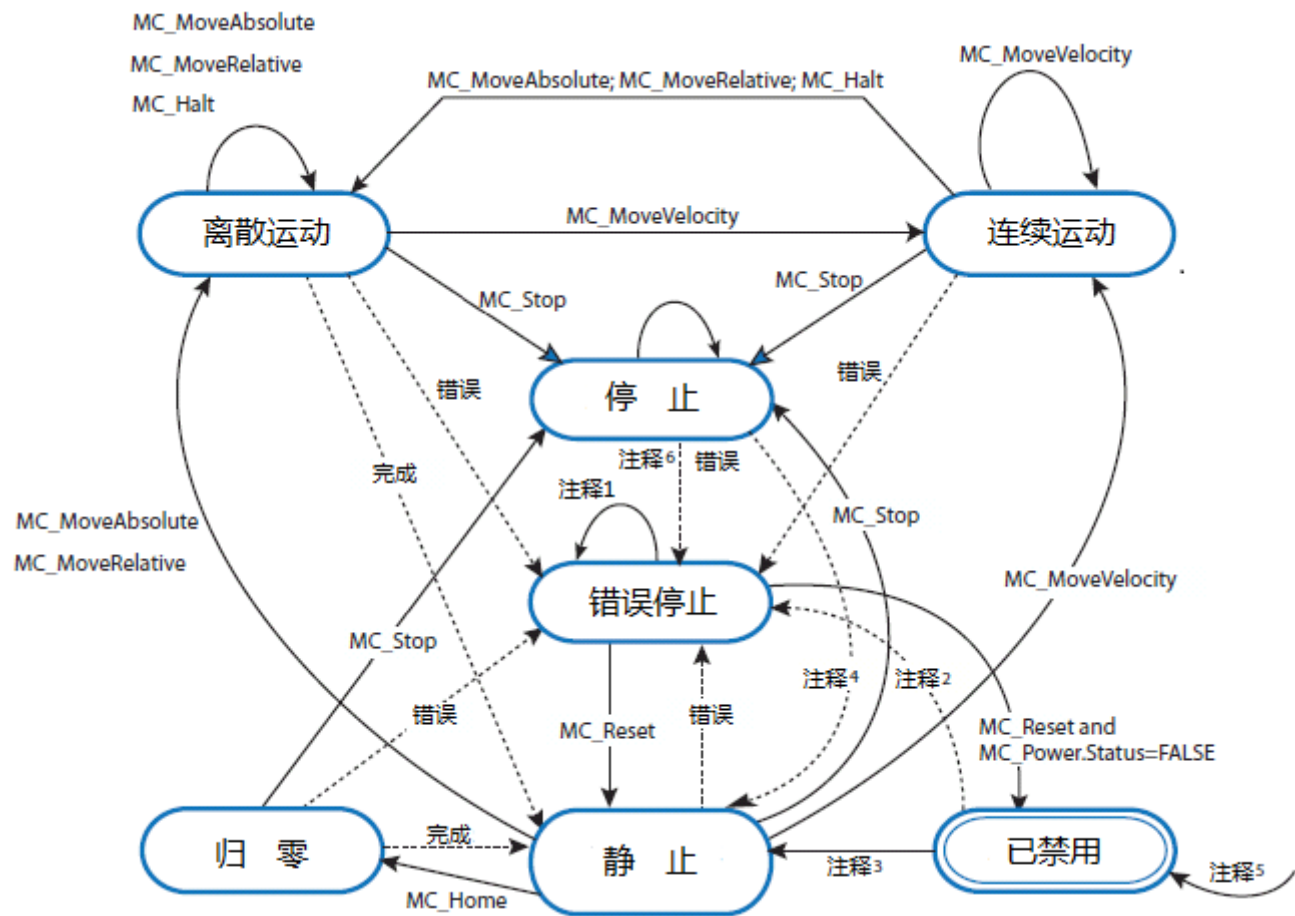
[轴变量](#) 参考页数 [465](#)

运动控制轴状态

当多个运动控制功能块均已激活时，高水平轴行为的基本规则是始终按顺序执行运动命令，即使控制器能够进行实时并行处理也是如此。任何运动命令都是一种使轴状态发生变化的转换，因此它可以修改当前运动的计算方式。

运动控制轴状态图

轴将始终处于下图所示的其中一个定义的状态。



运动控制轴状态行为

下表描述了运动控制轴状态和参数。

编号	注意
1	在 ErrorStop 和 Stopping 状态中，虽然不可执行，但所有功能块（除了 MC_Reset）都可以调用。 MC_Reset 生成向 Standstill 状态的转换。如果在状态机器处于 Stopping 状态时出现错误，那么会生成向 ErrorStop 状态的转换。
2	Power.Enable = TRUE 且轴中存在错误。
3	Power.Enable = TRUE 且轴中不存在错误

编号	注意
4	MC_Stop.Done AND NOT MC_Stop.Execute.
5	当使用 Enable = FALSE 调用 MC_Power 时，轴的每个状态（包括 ErrorStop）会变为 Disabled 状态。
6	如果在状态机器处于 Stopping 状态时出现错误，则会生成向 ErrorStop 状态的过渡。

运动控制轴状态代码值

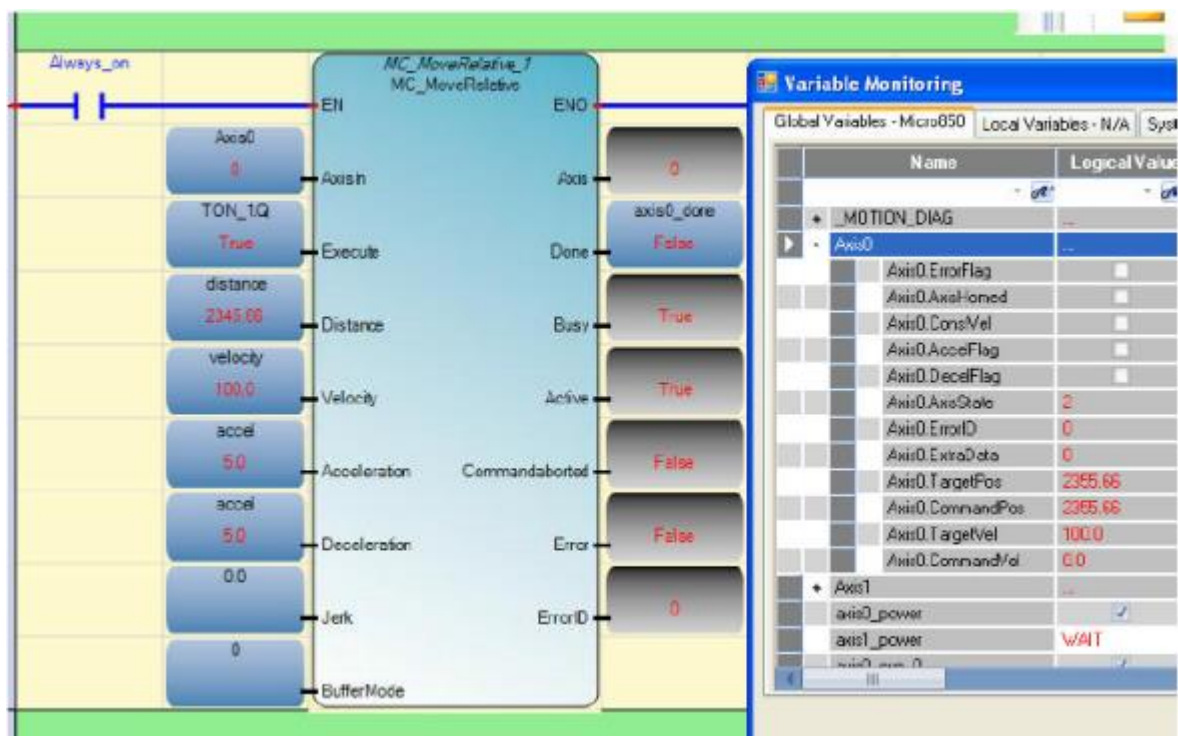
您可以使用“轴监视器”功能监视轴状态。下表标识了用于定义每个预定义轴状态的值。

状态值	状态名称
0x00	已禁用
0x01	静止
0x02	离散运动
0x03	连续运动
0x04	归位
0x06	正在停机
0x07	错误停止

轴状态更新

在运动执行中，轴状态更新取决于 POU 扫描何时调用相关的运动功能块。即使运动配置由运动引擎作为后台任务进行控制，也是如此，并且与 POU 扫描无关。

例如，在梯形 POU（状态为 rung=true）的运动轴上，会扫描梯级中的 MC_MoveRelative 功能块，而轴将开始运动。在 MC_MoveRelative 完成前，梯级的状态会变为 False，并且不再扫描 MC_MoveRelative。在此情况下，即使在轴完全停止后，速率变为 0，轴状态也不能从 Discrete Motion 切换为 StandStill。



另请参见

[运动控制指令](#) 参考页数 451

[运动控制功能块参数详细信息](#) 参考页数 455

运动控制功能块参数编号

以下功能块在功能块编程时使用特定的参数编号。

- MC_ReadParameter
- MC_ReadBoolParameter
- MC_WriteParameter
- MC_WriteBoolParameter

参数编号标识

为标准参数保留的参数编号在 0 和 999 之间。也允许供应商或用户的扩展，尽管这可能会影响在不同平台间的可移植性。如果参数编号大于 999，那么这是供应商特定的参数。

参数编号	参数名称	数据类型	R/W	说明
1	命令位置	REAL	R	命令位置。
2	SWLimitPos	REAL	R/W	正软件限制开关位置。
3	SWLimitNeg	REAL	R/W	负软件限制开关位置。
4	EnableLimitPos	BOOL	R/W	启用正软件限制开关。
5	EnableLimitNeg	BOOL	R/W	启用负软件限制开关。
8	MaxVelocitySystem	REAL	R	运动系统中允许的最大轴速率。
9	MaxVelocityAppl	REAL	R/W	应用程序中允许的最大轴速率。
11	CommandedVelocity	REAL	R	命令速率。
12	MaxAccelerationSystem	REAL	R	运动系统中允许的最大轴加速。
13	MaxAccelerationAppl	REAL	R/W	应用程序中允许的最大轴加速。
14	MaxDecelerationSystem	REAL	R	系统中允许的最大轴减速。
15	MaxDecelerationAppl	REAL	R/W	应用程序中允许的最大轴减速。
16	MaxJerk	REAL	R/W	允许的最大轴加加速度。
1001	TargetPosition	REAL	R	当前活动的移动功能块的最终目标位置
1002	TargetVelocity1	REAL	R	当前活动的移动功能块的最终目标速率。
1005	工作周期	REAL	R/W	一个脉冲的脉冲工作周期。有效值为 0 - 100，表示 0% - 100%。（通过调整该值可实现 PWM 功能）。 只能使用该功能块配置该参数。默认值由控制器设置为 50.0。 注意：对于工作周期，当控制器从“运行”模式切换为“编程”模式，然后再切换为“运行”模式时，或者当控制器重新上电时，该值会被默认设置 50.0 覆盖。
1006	PulsePerRevolution	REAL	R	用户在 CCW GUI 中输入的“每转的脉冲数”设置。
1007	TravelPerRevolution	REAL	R	用户在 CCW GUI 中输入的“每转的行程”设置。

另请参见

[运动控制指令](#) 参考页数 [451](#)

[运动控制功能块的一般规则](#) 参考页数 [452](#)

[运动控制功能块参数详细信息](#) 参考页数 [455](#)

运动控制功能块错误 ID

当运动控制功能块以错误结束，并且轴处于 ErrorStop 状态，在大部分情况下，MC_Reset 功能块 (或 MC_Power Off/On 和 MC_Reset) 可用来恢复轴。可将轴重置为正常运动操作，而无需停止控制器操作。

值	MACRO ID	说明
00	MC_FB_ERR_NO	功能块执行成功。
01	MC_FB_ERR_WRONG_STATE	功能块无法执行，因为轴未处于正确的状态下。检查轴状态。
02	MC_FB_ERR_RANGE	功能块无法执行，因为功能块中设置了无效的轴动态参数 (Velocity、Acceleration、Deceleration 或 Jerk)。 根据轴动态配置页面更正功能块中的动态参数设置。
03	MC_FB_ERR_PARAM	功能块无法执行，因为功能块中设置了除 Velocity、Acceleration、Deceleration 或 Jerk 之外的无效参数。 更正功能块的参数设置 (例如模式或位置)。
04	MC_FB_ERR_AXISNUM	由于轴不存在、轴配置数据遭到破坏或者轴未正确配置而无法执行功能块。
05	MC_FB_ERR_MECHAN	功能块无法执行，因为此轴由于驱动器或机械问题导致出现故障。检查驱动程序和控制器之间的连接 (指示驱动程序就绪和处于原位信号)，并确保驱动程序运行正常。
06	MC_FB_ERR_NOPOWER	功能块无法执行，因为轴未接通电源。 使用 MC_Power 功能块为轴接通电源。

值	MACRO ID	说明
07	MC_FB_ERR_RESOURCE	功能块无法执行，因为功能块所需的资源受某些其他功能块控制或不可用。 确保功能块所需的资源可供使用。 示例： • MC_Power 尝试控制相同的轴。 • MC_Stop 在同一时间对相同轴执行。 • MC_TouchProbe 在同一时间对相同轴执行。） • MC_TouchProbe 已执行，但是触摸探针输入未在运动配置中启用。
08	MC_FB_ERR_PROFILE	功能块无法执行，因为功能块中定义的运动配置无法实现。 更正功能块中的配置。
09	MC_FB_ERR_VELOCITY	功能块无法执行，因为功能块中请求的运动配置由于当前轴速率问题而无法实现。 示例： • 功能块要求轴在移动时反向进行。 • 由于当前速率过低或过高，而导致无法实现所需的运动配置。 当轴速率与请求的运动配置兼容时，检查功能块中的运动配置设置并更正配置，或重新执行功能块。
0A	MC_FB_ERR_SOFT_LIMIT	此功能块无法执行，因为超出软限位范围之外功能块将结束移动，或者由于达到软限位而中止功能块。 检查功能块中的速率或目标位置设置，或者调整软限位设置。
0B	MC_FB_ERR_HARD_LIMIT	由于在轴移动过程中或轴移动开始之前检测到硬限位开关活动状态，因此该功能块中止。 朝相反方向将轴从硬限位开关中移开。
0C	MC_FB_ERR_LOG_LIMIT	此功能块无法执行，因为超出 PTO 累加器逻辑限制范围之外功能块将结束移动，或者由于达到 PTO 累加器逻辑限制而中止功能块。 检查功能块的速率或目标位置设置。或者使用 MC_SetPosition 功能块调整轴坐标系。
0D	MC_FB_ERR_ERR_ENGINE	在执行此功能块过程中检测到运动引擎执行错误。 断开再接通整个运动设备（包括控制器、驱动程序和执行器）的电源，然后重新下载用户应用程序。如果故障持续存在，请致电技术支持。
10	MC_FB_ERR_NOT_HOMED	功能块无法执行，因为轴需要先进行归位。 使用 MC_Home 功能块对轴执行归位。
80	MC_FB_ERR_PARAM_MODIFIED	警告：该轴请求的速率已调整到较低值。 功能块以较低速率成功执行。

另请参见

- [运动控制指令](#) 参考页数 451
- [运动控制功能块的一般规则](#) 参考页数 452
- [运动控制轴状态](#) 参考页数 455
- [运动控制功能块参数详细信息](#) 参考页数 455

轴出错场景

在大多数情况下，如果向轴发出移动功能块指令而导致功能块出错，则轴也会被标记为处于“错误”状态 ,并在轴的 AXIS_REF 数据上设置相应的 ErrorID 元素。但是，如果出现以下情况，则可能不会一直标记轴出错，因此，在轴状态发生变化后，用户应用程序仍可向轴发出成功的移动功能块。

场景	示例
移动功能块指示一个轴，但该轴处于无法正确执行功能块的状态。	轴未接通电源，或轴处于“归位”序列或 Error Stop 状态。
移动功能块指示一个轴，但该轴仍被其他移动功能块控制。在未进入完全停止的状态前，该轴不允许新的功能块控制运动。	新的功能块命令轴改变运动方向。
如果一个移动功能块尝试控制某个轴，但该轴仍被另一个移动功能块所控制，则控制器无法实现新定义的运动配置文件。	用户应用程序向移动时既定距离太短的某个轴发出 S-Curve MC_MoveAbsolute 功能块。

AXIS_REF 数据类型

AXIS_REF 数据类型是一种包含运动轴信息的数据结构。它用作所有运动控制功能块中的输入和输出变量。将运动轴添加到配置中时，会自动创建 AXIS_REF 数据类型的实例。

参数	数据类型	说明
Axis_ID	AXIS_REF	Connected Components Workbench 将自动分配逻辑轴 ID。用户无法对其进行编辑或查看。
错误标记	BOOL	表示错误是否在轴中存在。
AxisHomed	BOOL	表示是否已对轴成功执行归位操作。 当用户对已设置 AxisHomed(已成功执行归位)的轴尝试重新执行归位操作时,结果并不成功,并且 AxisHomed 状态将被清除。
ConstVel	BOOL	表示轴是否以恒定速率运动。恒定速率中不考虑固定轴。
AccFlag	BOOL	表示轴是否以加速率运动。
DecelFlag	BOOL	表示轴是否以减速率运动。
AxisState	USINT	表示轴的当前状态。
ErrorID	UINT	当错误标志显示错误时指出轴错误的原因。此错误通常是由运动控制功能块执行失败造成的。
ExtraData	UINT	已保留。
TargetPos	REAL	表示 MoveAbsolute 和 MoveRelative 功能块的轴的最终目标位置。 对于 MoveVelocity、Stop 和 Halt 功能块,TargetPos 为 0,除非由先前位置功能块设置的 TargetPos 未清除。
CommandPos	REAL	在运动期间,这是控制器命令轴使用的当前位置。在轴实际位置和此 CommandPos 之间可能会有一小段延迟。
TargetVel	REAL	对移动功能块的轴指示的最大目标速率。当前功能块中的 TargetVel 值,或比其小的值,具体取决于相同功能块中的其他参数。
CommandVel	REAL	在运动期间,此元素表示控制器指示轴使用的当前速率。 请注意,由于驱动器延迟或驱动器调整过头,因此在轴实际速率和 CommandVel 之间可能会有一小段延迟。

重要说明： 如果轴已标记为出错,而错误 ID 不为零,则必须在发布任何其他移动功能块前,使用 MC_Reset 将轴重置。

另请参见

[运动控制功能块参数详细信息](#) 参考页数 [455](#)

[轴变量](#) 参考页数 [465](#)

FB_AXIS_REF 数据类型

FB_AXIS_REF 数据类型是一种包含运动反馈轴信息的数据结构。它用作运动控制功能块中的输入和输出变量。当您添加 HSC 模块且将模式配置为“反馈轴”模式，则会自动创建 FB_AXIS_REF 数据类型实例。

参数	数据类型	说明
Axis_ID	FB_AXIS_REF	Connected Components Workbench 将自动分配逻辑轴 ID。用户无法对其进行编辑或查看。
ErrorFlag	BOOL	表示反馈轴中是否存在错误。
ConstVel	BOOL	表示反馈轴是否以恒定速率运动。恒定速率中不考虑固定轴。
AccelFlag	BOOL	表示反馈轴是否以加速率运动。
DecelFlag	BOOL	表示反馈轴是否以减速率运动。
AxisState	USINT	表示反馈轴的当前状态。
ErrorID	UINT	当错误标志显示错误时指出轴错误的原因。此错误通常是由运动控制功能块执行失败造成的。
ExtraData	UINT	已保留。
ActualPos	REAL	实际的机械位置从运动反馈通道 (HSC) 中读回。
ActualVel	REAL	实际的机械速度从运动反馈通道 (HSC) 中读回。

重要说明： 如果反馈轴已标记为出错，且错误 ID 不为零，则必须在发布任何其他移动功能块前，使用 MC_Reset 将 FBAxis 重置。

另请参见

[运动控制功能块参数详细信息](#) 参考页数 455

[轴变量](#) 参考页数 465

轴变量

轴变量用来控制给定的运动控制轴位置、速度、加速和错误。

将变量分配给轴输出参数

在功能块图中

为了方便，您可以以图形化的方式将运动控制功能块的 `Axis` 输出参数连接至另一个运动控制功能块的 `AxisIn` 输入参数。例如，将 `MC_POWER` 的 `Axis` 输出参数连接至 `MC_HOME` 的 `AxisIn` 输入参数。

在梯形图中

不得将变量分配给运动控制功能块的 `Axis` 输出参数，因为该参数是只读的。

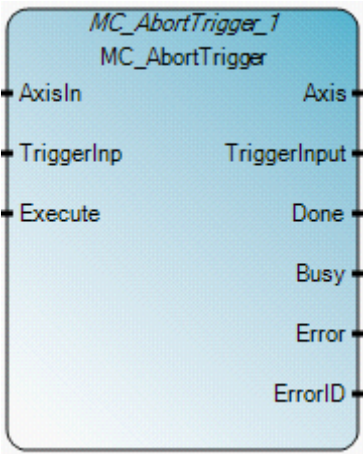
监视 `AXIS_REF` 变量

当运动引擎处于活动状态或者在用户应用程序中作为用户逻辑的一部分时，用户可以在控制器调试模式下监视软件中的 `AXIS_REF` 变量。您还可以通过各种通信信道远程监视 `AXIS_REF` 变量。

另请参见

[运动控制指令](#) 参考页数 [451](#)

MC_AbortTrigger MC_AbortTrigger 中止连接到触发事件的其他功能块。例如，MC_TouchProbe。



MC_AbortTrigger 操作

- 只有分配到由 MC_TouchProbe 控制的轴时，MC_AbortTrigger 功能块才会执行。

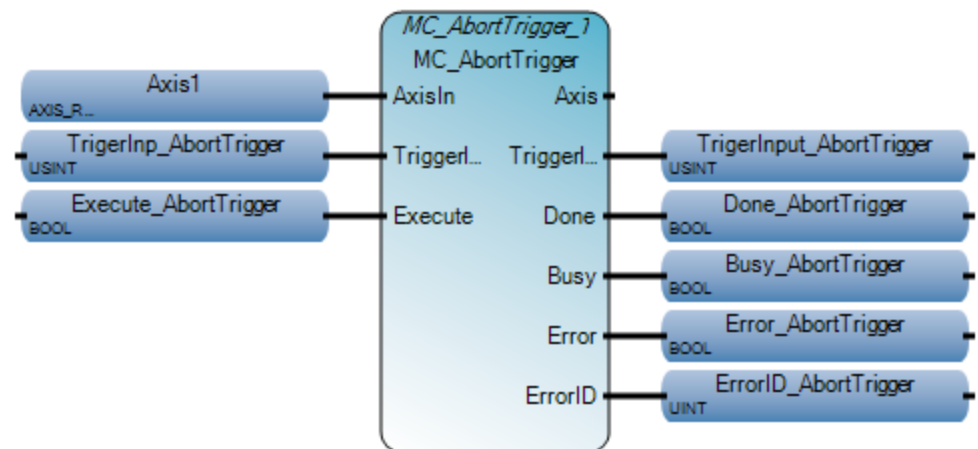
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_AbortTrigger 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
TriggerInp	输入	USINT	此参数将被忽略。
Execute	输入	BOOL	当为 TRUE 时，在上升沿中止触发事件。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
TriggerInput	输出	USINT	此参数将被忽略。
Done	输出	BOOL	当触发事件中止时为 TRUE。

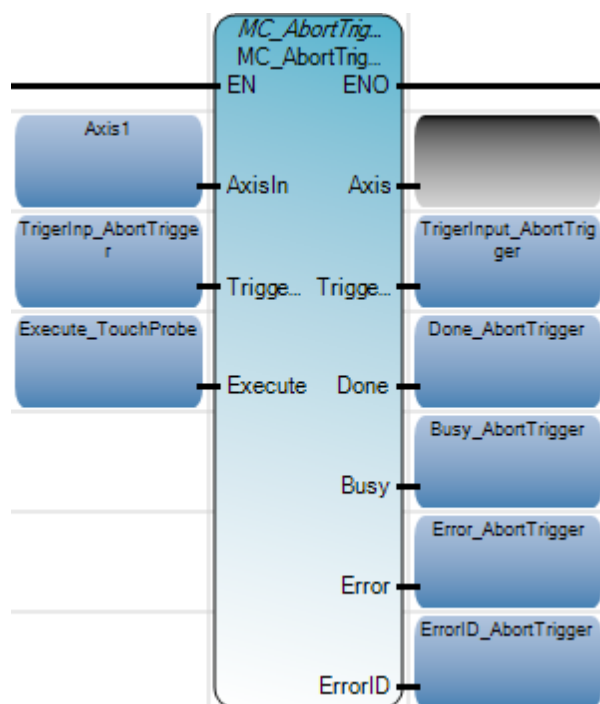
参数	参数类型	数据类型	说明
Busy	输出	BOOL	当功能块未完成时为 TRUE。
Error	输出	BOOL	当检测到错误时为 TRUE。
ErrorID	输出	UINT	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460。

MC_AbortTrigger 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

MC_AbortTrigger_1(Axis1,TrigerInp_AbortTrigger,Execute_AbortTrigger);
Done_AbortTrigger := MC_AbortTrigger_1.Done;
Busy_AbortTrigger := MC_AbortTrigger_1.Busy;
Error_AbortTrigger := MC_AbortTrigger_1.Error;
ErrorID_AbortTrigger := MC_AbortTrigger_1.ErrorID;

```

```

MC_AbortTrigger_1(

```

```

void MC_AbortTrigger_1(AXIS_REF AxisIn, USINT TriggerInp, BOOL Execute)
Type : MC_AbortTrigger, Abort Function Blocks connected to trigger events (e.g. MC_TouchProbe)

```

结果



另请参见

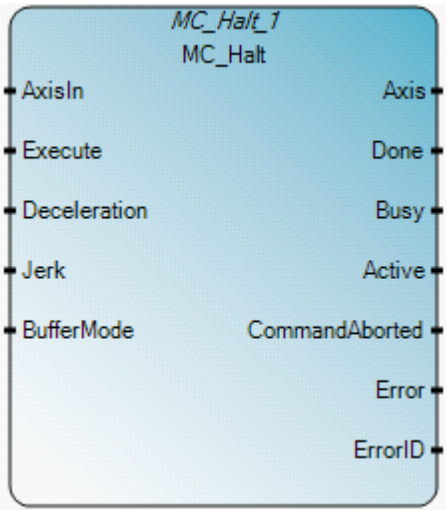
[运动控制指令](#) 参考页数 [451](#)

[运动控制功能块参数详细信息](#) 参考页数 [455](#)

[运动控制功能块的一般规则](#) 参考页数 [452](#)

MC_Halt

MC_Halt 命令受控制的运动停止。在正常操作条件下，使用 MC_Halt 来停止轴。轴状态更改为 DiscreteMotion，直至速率为零。当速率达到零时，Done 将设置为 TRUE，并且轴状态更改为 Standstill。



MC_Halt 操作

- 在轴减速期间可以执行其他运动命令，而这将中止 MC_Halt 功能块。
- 如果在轴正处于 Homing 状态时发布 MC_Halt 功能块，则该功能块会报告错误，而归位进程不会中断。

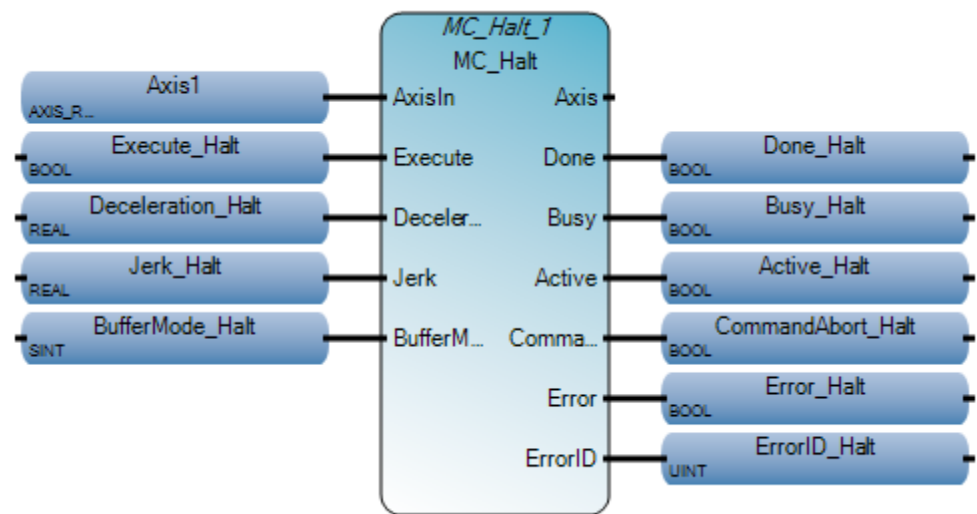
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_Halt 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
Execute	输入	BOOL	当为 TRUE 时，在上升沿开始运动。 注意：在归位期间执行 MC_Halt 时，MC_Halt 将被设置为 MC_FB_ERR_STATE，并且归位进程继续。

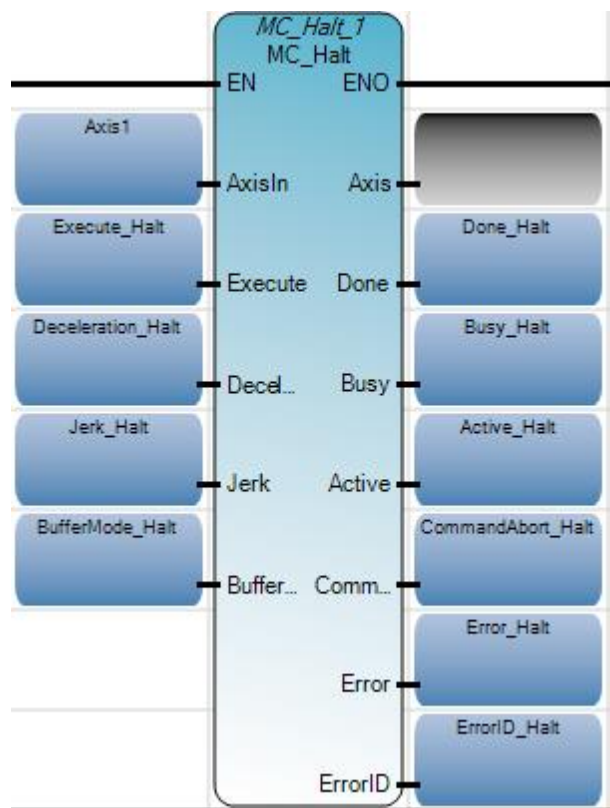
参数	参数类型	数据类型	说明
减速	输入	REAL	减速的值（始终为正值）（减少电机的能量）。 注意：如果 Deceleration ≤ 0 且轴未处于 Standstill 状态，则 MC_Halt 将被设置为 MC_FB_ERR_RANGE。
加加速度	输入	REAL	加加速度的值（始终为正值）。 注意：如果 Jerk < 0 且轴处于 Standstill 状态，则 MC_Halt 将被设置为 MC_FB_ERR_RANGE。
BufferMode	输入	SINT	未使用。该模式始终为 MC_Aborting。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
Done	输出	BOOL	达到零速率。
Busy	输出	BOOL	功能块未完成。
Active	输出	BOOL	表示功能块可控制轴。
CommandAborted	输出	BOOL	命令已被其他命令或 Error Stop 中止。
Error	输出	BOOL	FALSE - 没有错误。 TRUE - 检测到错误。
ErrorID	输出	UINT	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460。

MC_Halt 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
Deceleration_Halt := 10.0;  
Jerk_Halt := 10.0;  
MC_Halt_1(Axis1,Execute_Halt,Deceleration_Halt,Jerk_Halt,BufferMode_Halt);  
Done_Halt := MC_Halt_1.Done;  
Busy_Halt := MC_Halt_1.Busy;  
Active_Halt := MC_Halt_1.Active;  
Command&abort_Halt := MC_Halt_1.Command&aborted;  
Error_Halt := MC_Halt_1.Error;  
ErrorID_Halt := MC_Halt_1.ErrorID;
```

MC_Halt_1{

void MC_Halt_1(AXIS_REF AxisIn, BOOL Execute, REAL Deceleration, REAL Jerk, SINT BufferMode)

Type : MC_Halt, Commands a controlled motion stop. The axis is moved to the state Discrete Motion, until the velocity is zero.

结果



另请参见

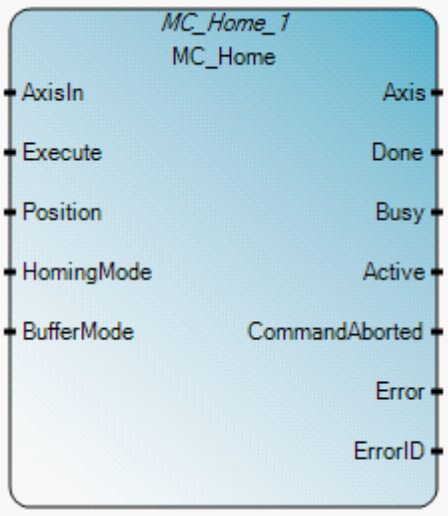
[运动控制指令](#) 参考页数 [451](#)

[运动控制功能块参数详细信息](#) 参考页数 455

[运动控制功能块的一般规则](#) 参考页数 452

MC_Home

MC_Power 命令轴执行 <search home> 序列。此序列的详细信息依赖于制造商，可根据轴参数进行设置。“Position”输入用于在检测到参考信号且达到配置的 Home 偏移时设置绝对位置。



MC_Home 操作

- 发布 MC_Power 后，轴 Homed 状态会重置为 0（未归位）。多数情况下，轴接通电源后，需要执行 MC_Home 功能块，才可以校准轴位置和归位参考。
- MC_Home 功能块仅可使用 MC_Stop 或 MC_Power 功能块中止。如果在完成之前中止，则先前搜索到的归位位置会被视为无效并清除轴的 Homed 状态。

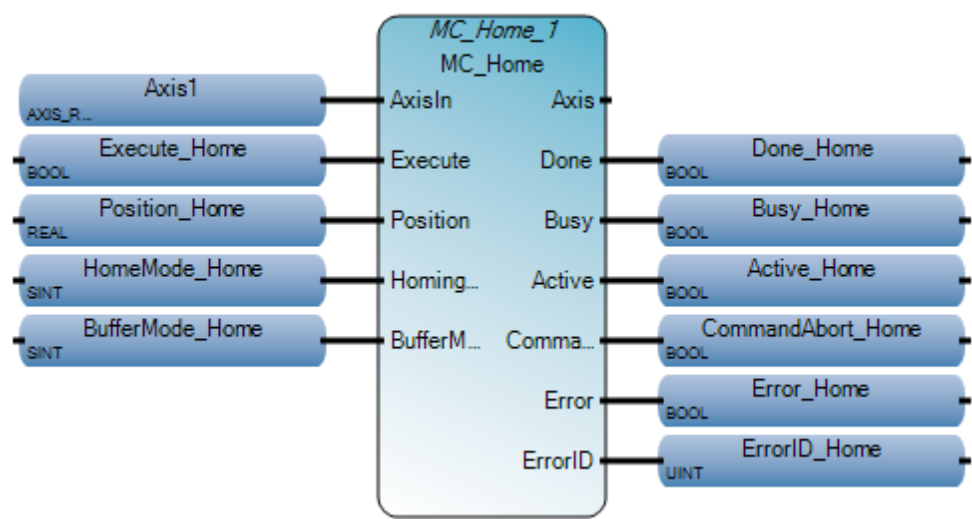
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_Home 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。

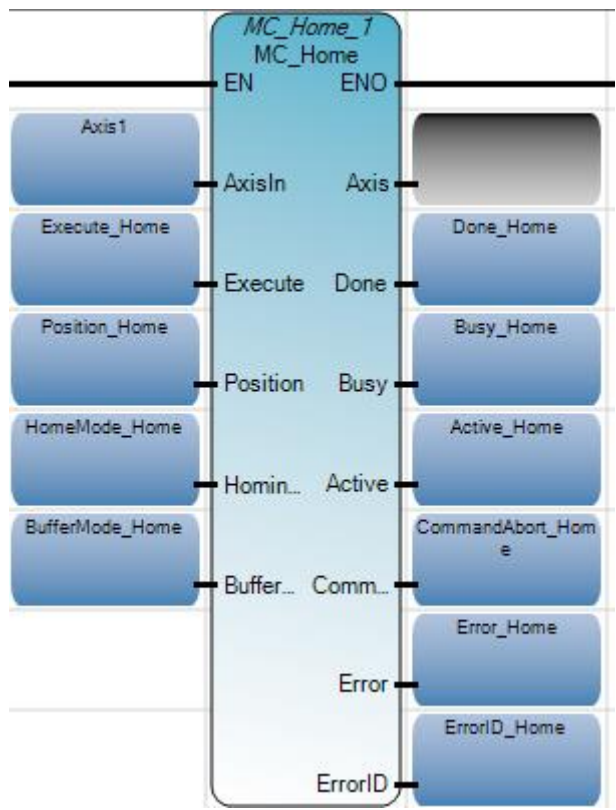
参数	参数类型	数据类型	说明
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
Execute	输入	BOOL	当为 TRUE 时，在上升沿开始运动。
Position	输入	REAL	当检测到参考信号且达到配置的主偏移时设置绝对位置。 在位置从用户位置转换到 PTO 脉冲时，此输入的值范围为 -0x40000000 - 0x40000000 实际值脉冲。设置软限位内的位置值。 无效的输入值导致产生了错误。 错误 ID = MC_FB_ERR_PARAM。
HomingMode	输入	SINT	归位模式的枚举输入 请参见归位模式 参考页数 478。
BufferMode	输入	SINT	未使用。该模式始终为 mcAborting。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
Done	输出	BOOL	当为 TRUE 时，归位操作会成功完成且轴状态设置为 Standstill。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Active	输出	BOOL	当为 TRUE 时，表示功能块可控制轴。
CommandAborted	输出	BOOL	当为 TRUE 时，表示命令已被其他命令或 Error Stop 功能块中止。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	单位	错误标识。 另请参见运动控制功能块错误 ID 参考页数 460。

MC_Home 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)


```

Position_Home := -50000.0;
HomeMode_Home := 4;  (*1*)
MC_Home_1(Axis1,Execute_Home,Position_Home,HomeMode_Home,BufferMode_Home);
Done_Home := MC_Home_1.Done;
Busy_Home := MC_Home_1.Busy;
Active_Home := MC_Home_1.Active;
CommandAbort_Home := MC_Home_1.CommandAborted;
Error_Home := MC_Home_1.Error;
ErrorID_Home := MC_Home_1.ErrorID;

```

MC_Home_1 {

void MC_Home_1(AXIS_REF AxisIn, BOOL Execute, REAL Position, SINT HomingMode, SINT BufferMode)
Type : MC_Home, Commands the axis to perform the home searching sequence

结果

The screenshot displays two windows from the Micro850 software. The 'Micro850 - 轴监视器' (Axis Monitor) window on the left shows configuration for 'Axis1'. The '轴状态' (Axis Status) is '归位' (Homed), '轴已归位' (Axis Homed) is '否' (No), and '移动' (Move) is '恒定速率' (Constant Rate). The '位置 and 速率' (Position and Rate) section shows '命令位置' (Command Position) at -176.07 mm and '命令速率' (Command Rate) at -25.0. The '目标位置' (Target Position) is 0.0 mm and '目标速率' (Target Rate) is -25.0. The '变量监视器' (Variable Monitor) window on the right shows a table of variables for 'MC_Home_1'.

名称	逻辑值	物理值	锁定	数据类型
Execute_Home	☒	N/A	☐	BOOL
Position_Home	-50000.0	N/A	☐	REAL
HomeMode_Home	1	N/A	☐	SINT
BufferMode_Home	0	N/A	☐	SINT
Done_Home	☐	N/A	☐	BOOL
Busy_Home	☒	N/A	☐	BOOL
Active_Home	☒	N/A	☐	BOOL
CommandAbort_Hc	☐	N/A	☐	BOOL
Error_Home	☐	N/A	☐	BOOL
ErrorID_Home	0	N/A	☐	UINT

另请参见

[运动控制指令](#) 参考页数 [451](#)

[运动控制功能块参数详细信息](#) 参考页数 [455](#)

[运动控制功能块的一般规则](#) 参考页数 [452](#)

归位模式

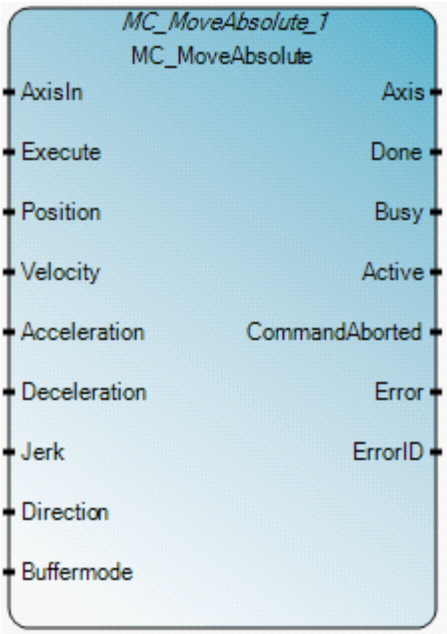
值	名称	说明
0x00	MC_HOME_ABS_SWITCH	通过搜索归位绝对开关进行的归位进程
0x01	MC_HOME_LIMIT_SWITCH	通过搜索限位开关进行的归位进程
0x02	MC_HOME_REF_WITH_ABS	通过搜索归位绝对开关加上使用编码参考脉冲进行的归位进程
0x03	MC_HOME_REF_PULSE	通过搜索限位开关加上使用编码参考脉冲进行的归位进程
0x04	MC_HOME_DIRECT	通过从用户参考直接强制归位位置进行的静态归位进程。功能块会将机制所在的当前位置设置为归位位置，其位置由输入参数“Position”决定

另请参见

[MC_Home](#) 参考页数 [474](#)

MC_MoveAbsolute

MC_MoveAbsolute 命令受控制的运动到指定的绝对位置。



MC_MoveAbsolute 操作

- 对于 Micro800 控制器,由于运动方向是由当前位置和目标位置决定的,因此会忽略 MC_MoveAbsolute 功能块的输入速率符号。
- 对于 Micro800 控制器,由于仅有一个数学解答能到达目标位置,因此会忽略 MC_MoveAbsolute 功能块的输入方向。
- 如果在 Micro800 控制器轴的状态为 Standstill 且移动相对距离为零时发布 MC_MoveAbsolute 功能块,则对功能块的执行将直接报告为 Done。
- 如果将 MC_MoveAbsolute 功能块发布到非 Homed 位置中的轴,则该功能块将报告错误。
- 如果未被其他功能块中止,则 MoveAbsolute 功能块会以零速率完成。

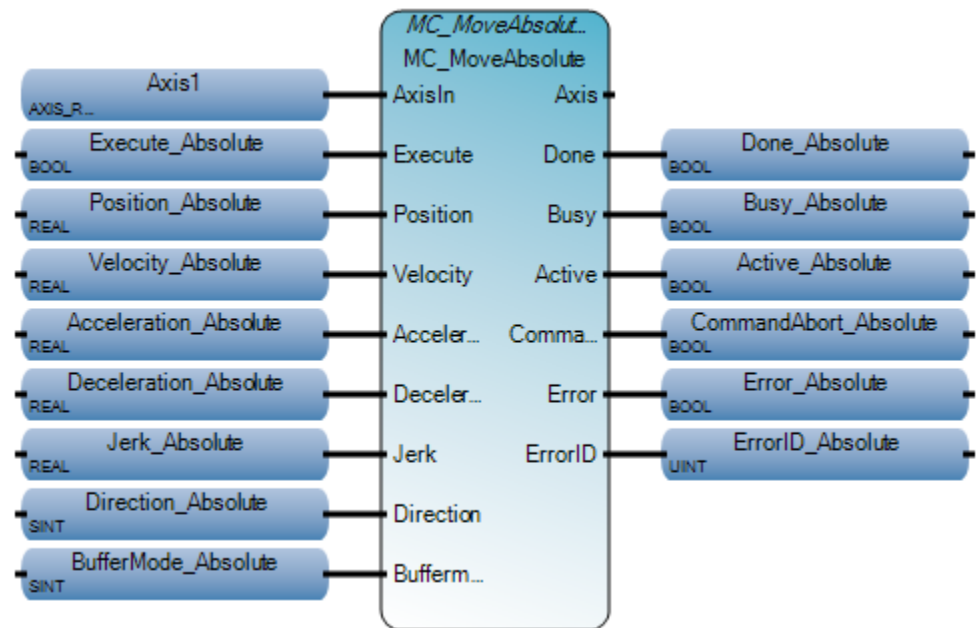
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时,执行当前 MC_MoveAbsolute 计算。 当 EN = FALSE 时,不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
Execute	输入	BOOL	当为 TRUE 时,在上升沿开始运动。 当发出此执行命令或者出现错误 MC_FB_ERR_NOT_HOMED 时,轴应处于归位位置。
Position	输入	REAL	技术单元中运动的目标位置(负值或正值)。 注意: 技术单元在轴的“运动 - 常规”配置页面进行定义。
速率	输入	REAL	速率最大值。 当 Jerk = 0 时可能不会达到最大速率。 速率的符号会被忽略,运动方向由输入位置来决定。

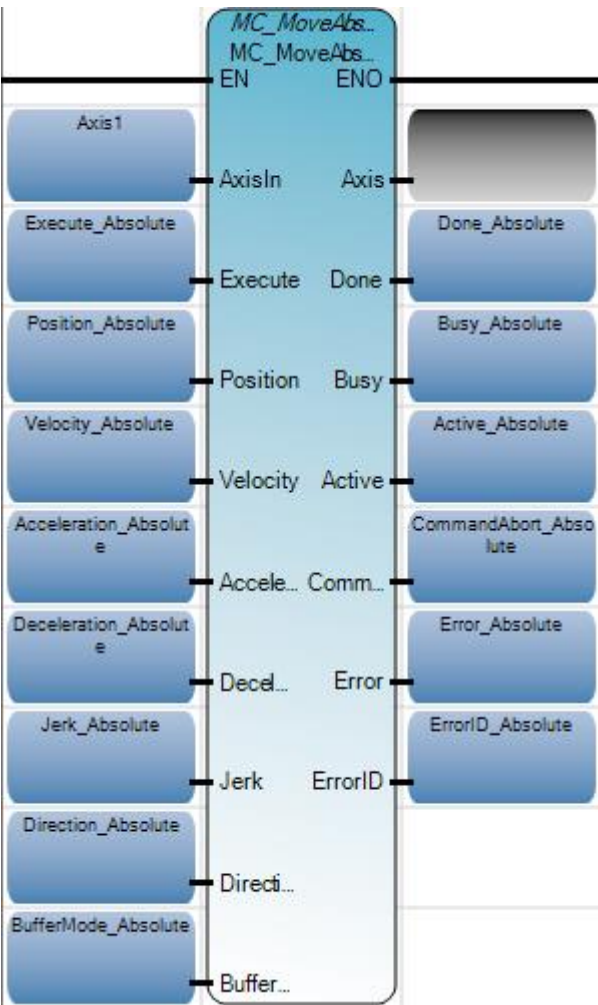
参数	参数类型	数据类型	说明
Acceleration	输入	REAL	加速的值（始终为正值 - 可增加电机的能量）。 用户单位/sec ²
减速	输入	REAL	减速的值（始终为正值 - 可减少电机的能量）。 u/秒 ²
加加速度	输入	REAL	加加速度的值（始终为正值）。 u/秒 ³ 注意： 当输入加加速度的值为 0 时，Trapezoid 配置将由运动引擎来计算。当 Jerk > 0 时，将计算 S-Curve 配置。
方向	输入	SINT	此参数未使用。
BufferMode	输入	SINT	此参数未使用。
ENO	输出	BOOL	启用输出 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462 。
Done	输出	BOOL	当为 TRUE 时，将达到命令位置。 当此轴的原位输入配置为 Enabled 时，则在此操作位变为 TRUE 之前，驱动器需要将原位输入符号设置为活动。 此操作完成，速率为零（除非操作中止）。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Active	输出	BOOL	当为 TRUE 时，表示功能块可控制轴
CommandAborted	输出	BOOL	当为 TRUE 时，表示命令已被其他命令或 Error Stop 功能块中止。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	UINT	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460 。

MC_MoveAbsolute 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
Position_Absolute := 50000.0;
Velocity_Absolute := 500.0;
Acceleration_Absolute := 1000.0;
Deceleration_Absolute := 1000.0;
Jerk_Absolute := 10.0;
MC_MoveAbsolute_1(Axis1,Execute_Absolute,Position_Absolute,Velocity_Absolute,
Acceleration_Absolute,Deceleration_Absolute,Jerk_Absolute,Direction_Absolute,BufferMode_Absolute);
Done_Absolute := MC_MoveAbsolute_1.Done;
Busy_Absolute := MC_MoveAbsolute_1.Busy;
Active_Absolute := MC_MoveAbsolute_1.Active;
CommandAbort_Absolute := MC_MoveAbsolute_1.CommandAborted;
Error_Absolute := MC_MoveAbsolute_1.Error;
ErrorID_Absolute := MC_MoveAbsolute_1.ErrorID;
```

MC_MoveAbsolute_1

void MC_MoveAbsolute_1(AXIS_REF AxisIn, BOOL Execute, REAL Position, REAL Velocity, REAL Acceleration, REAL Deceleration, REAL Jerk, SINT Direction, SINT Buffermode)
Type : MC_MoveAbsolute, Commands a controlled motion to a specified absolute position

结果

Micro850 - 轴监视器

轴名称(X):
Axis1

轴状态:
离散运动

轴已归位:
是

移动:
恒定速率

错误描述:

位置和速率

命令位置:
-44400.6 mm

命令速率:
500.0 mm/sec

目标位置:
50000.0 mm

目标速率:
500.0 mm/sec

变量监视器

全局变量 - Micro850

本地变量 - N/A

系统变量 - Micro850

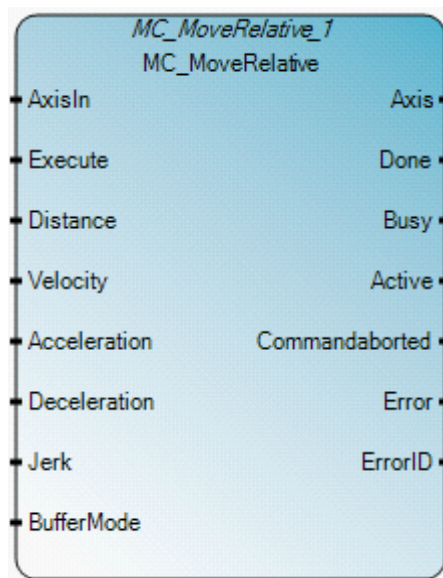
名称	逻辑值	物理值	锁定	注释
Execute_Absolute	☒	N/A	☐	BOI
Position_Absolute	50000.0	N/A	☐	REA
Velocity_Absolute	500.0	N/A	☐	REA
Acceleration_Absolute	1000.0	N/A	☐	REA
Deceleration_Absolute	1000.0	N/A	☐	REA
Jerk_Absolute	10.0	N/A	☐	REA
Direction_Absolute	0	N/A	☐	SIN
BufferMode_Absolute	0	N/A	☐	SIN
Done_Absolute	☐	N/A	☐	BOI
Busy_Absolute	☒	N/A	☐	BOI
Active_Absolute	☒	N/A	☐	BOI
CommandAbort_Absolute	☐	N/A	☐	BOI
Error_Absolute	☐	N/A	☐	BOI
ErrorID_Absolute	0	N/A	☐	UIN

另请参见

- [运动控制指令](#) 参考页数 451
- [运动控制指令的公共信息](#) 参考页数 455

MC_MoveRelative

MC_MoveRelative 在执行时命令与实际位置相对的指定距离的受控制运动。



MC_MoveRelative 操作

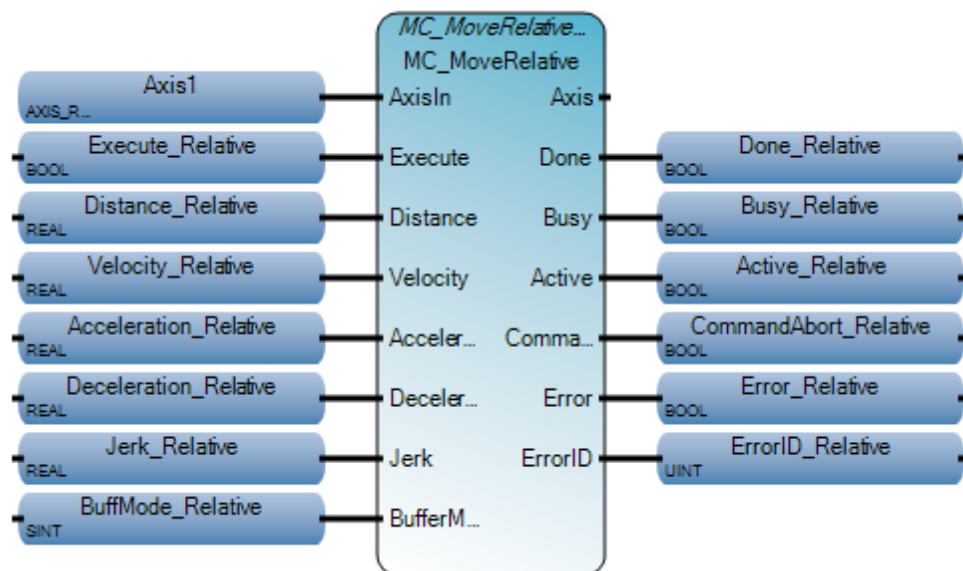
- 由于 MC_MoveRelative 功能块的运动方向取决于当前位置和目标位置，因此速率的符号被忽略。
- 如果未被其他功能块中止，则 MoveRelative 功能块会以零速率完成。
- 如果在 Micro800 控制器轴的状态为 StandStill 且移动相对距离为零时发布 MC_MoveRelative 功能块，则对功能块的执行将直接报告为 Done。
- 对于 Micro800 控制器，由于运动方向是由当前位置和目标位置决定的，因此会忽略 MC_MoveRelative 功能块的输入速率符号。

参数

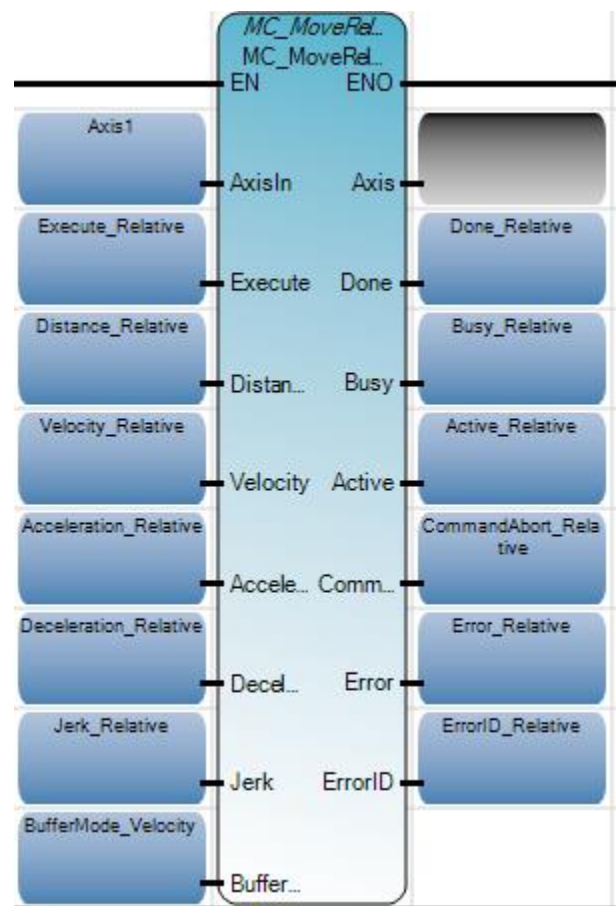
参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_MoveRelative 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
Execute	输入	BOOL	当为 TRUE 时，在上升沿开始运动。
Distance	输入	REAL	运动的相对距离（以技术单元 [u] 为单位）。
速率	输入	REAL	速率最大值（不一定达到）[u/s]。由于运动方向取决于输入位置，因此速率的符号会被功能块忽略。 提示： 当 Jerk = 0 时可能不会达到最大速率。
Acceleration	输入	REAL	加速的值（可增加电机的能量）[u/s ²]
减速	输入	REAL	减速的值（可减少电机的能量）[u/s ²]
加加速度	输入	REAL	加加速度的值 [u/s ³]
BufferMode	输入	SINT	此参数未使用。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
Done	输出	BOOL	当为 TRUE 时，将达到命令距离。 如果轴的原位输入已启用，则在 Done = True 之前，必须将原位输入符号设置为活动。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Active	输出	BOOL	当为 TRUE 时，表示功能块可控制轴
CommandAborted	输出	BOOL	命令已被其他命令或 Error Stop 中止。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	UINT	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460。

MC_MoveRelative 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

Distance_Relative := 100000.0;
Velocity_Relative := 300.0;
Acceleration_Relative := 100.0;
Deceleration_Relative := 100.0;
Jerk_Relative := 100.0;
MC_MoveRelative_1(Axis1,Execute_Relative,Distance_Relative,
Velocity_Relative,Acceleration_Relative,Deceleration_Relative,Jerk_Relative,BufferMode_Relative);
Done_Relative := MC_MoveRelative_1.Done;
Busy_Relative := MC_MoveRelative_1.Busy;
Active_Relative := MC_MoveRelative_1.Active;
CommandAbort_Relative := MC_MoveRelative_1.CommandAbort;
Error_Relative := MC_MoveRelative_1.Error;
ErrorID_Relative := MC_MoveRelative_1.ErrorID;

```

MC_MoveRelative_1

void MC_MoveRelative_1(AXIS_REF AxisIn, BOOL Execute, REAL Distance, REAL Velocity, REAL Acceleration, REAL Deceleration, REAL Jerk, SINT BufferMode)
Type : MC_MoveRelative, Commands a controlled motion of a specified distance relative to the actual position at the time of the execution.

结果

Micro850 - 轴监视器

轴名称(X): Axis1

轴状态: 离散运动

轴已归位: 否

移动: 恒定速率

错误描述:

位置和速率

命令位置:	60875.68 mm	命令速率:	300.0 mm/sec
目标位置:	100000.0 mm	目标速率:	300.0 mm/sec

变量监视器

全局变量 - Micro850

名称
Execute_Relative
Distance_Relative
Velocity_Relative
Acceleration_Relative
Deceleration_Relative
Jerk_Relative
BufferMode_Relative
Done_Relative
Busy_Relative
Active_Relative
CommandAbort_Relative
Error_Relative
ErrorID_Relative

另请参见

[运动控制指令](#) 参考页数 [451](#)

[运动控制功能块参数详细信息](#) 参考页数 [455](#)

[运动控制功能块的一般规则](#) 参考页数 [452](#)

MC_MoveVelocity

MC_MoveVelocity 命令以指定速率执行从不结束的受控制运动。



MC_MoveVelocity 操作

- 如果 MC_MoveVelocity 功能块的 DirectionIn 输入等于 0 且轴处于移动状态，则将忽略 Velocity 输入的符号；轴继续按照当前方向移动并应用新动态参数。
- 如果 MC_MoveVelocity 功能块的 DirectionIn 输入等于 0 且轴并非处于移动状态，则该功能块会报告错误。
- 如果在执行 MC_MoveVelocity 功能块期间达到 PTO 脉冲限制，则 PTO 累加器值会翻转到 0（如果已激活软限位，则翻转到软限位的相反极限），且该功能块会继续执行。
- 如果轴处于移动状态，且 MC_MoveVelocity 功能块发布与当前运动方向相反（Velocity * Direction 的符号）的运动，则该功能块会报告错误。
- 一旦已设置信号“InVelocity”，则表示已完成 MC_MoveVelocity 功能块。除信号“InVelocity”外，任何其他后续运动事件都不会影响功能块输出。
- 一旦轴的速率达到指示的速率，那么 MC_MoveVelocity 功能块的 InVelocity 输出始终为 True，直到该功能块被中止。

- (Velocity * Direction) 的符号决定 MC_MoveVelocity 功能块的运动方向。如果 Velocity 符号和 Direction 符号相同，则会发布正向运动。如果 Velocity 符号和 Direction 符号相反，那么会发布负向运动。
- 当 MC_MoveVelocity 被其他功能块/运动事件中止，或在“执行”的下降沿被中止，则将重置“InVelocity”信号。
- 要停止或更改 MC_MoveVelocity 功能块启动的运动，必须使用其他功能块（包括使用不同参数重新执行 MC_MoveVelocity 功能块）来中断或中止该功能块。
- 如果发布 MC_MoveVelocity 功能块时轴处于 StandStill 状态（不受其他功能块控制）且出现功能块错误，则轴状态将转为 ErrorStop。

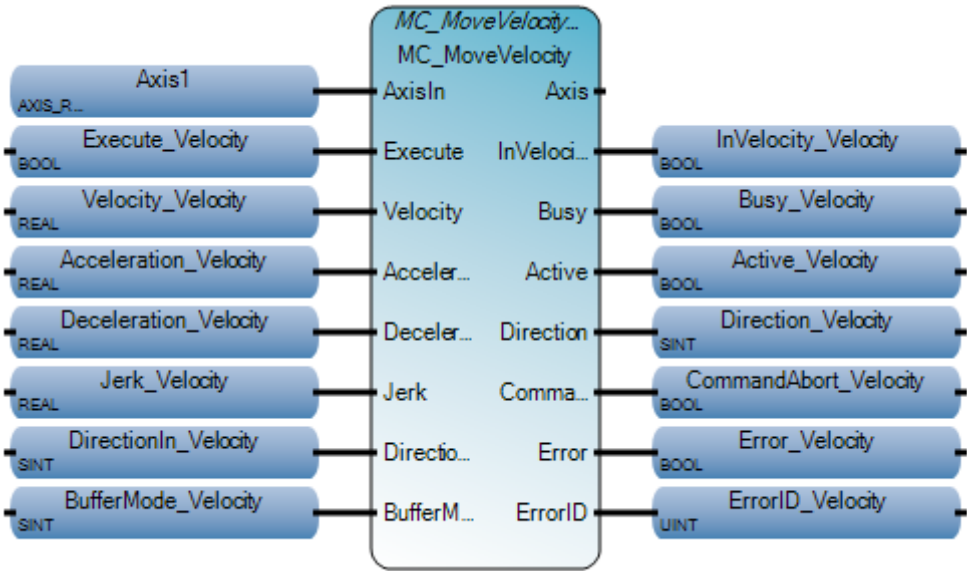
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_MoveVelocity 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
Execute	输入	BOOL	当为 TRUE 时，在上升沿开始运动。
速率	输入	REAL	速率最大值 [u/s]。
Acceleration	输入	REAL	加速的值（可增加电机的能量）[u/s ²]
减速	输入	REAL	减速的值（可减少电机的能量）[u/s ²]
加加速度	输入	REAL	加加速度的值 [u/s ³]
DirectionIn	输入	SINT	有效值包括：-1、0、1。
BufferMode	输入	SINT	此参数未使用。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。

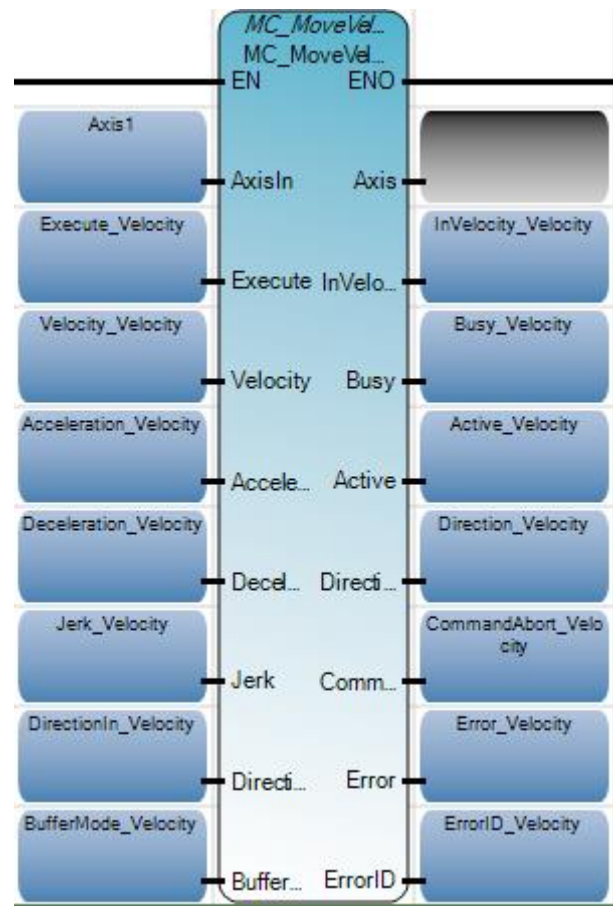
参数	参数类型	数据类型	说明
InVelocity	输出	BOOL	当为 TRUE 时，会达到命令的速率（首次）。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Active	输出	BOOL	当为 TRUE 时，表示功能块可控制轴。
方向	输出	SINT	有效值包括：-1、0、1。
CommandAborted	输出	BOOL	当为 TRUE 时，表示命令已被其他命令或 Error Stop 功能块中止。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	UINT	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460。

MC_MoveVelocity 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
Velocity_Velocity := 400.0;
Acceleration_Velocity := 100.0;
Deceleration_Velocity := 100.0;
Jerk_Velocity := 100.0;
DirectionIn_Velocity := 1;
MC_MoveVelocity_1(Axis1,Execute_Velocity,Velocity_Velocity,
Acceleration_Velocity,Deceleration_Velocity,Jerk_Velocity,DirectionIn_Velocity,BufferMode_Velocity);
InVelocity_Velocity := MC_MoveVelocity_1.InVelocity;
Busy_Velocity := MC_MoveVelocity_1.Busy;
Active_Velocity := MC_MoveVelocity_1.Active;
Direction_Velocity := MC_MoveVelocity_1.Direction;
CommandAbort_Velocity := MC_MoveVelocity_1.CommandAborted;
Error_Velocity := MC_MoveVelocity_1.Error;
ErrorID_Velocity := MC_MoveVelocity_1.ErrorID;
```

MC_MoveVelocity_1

void MC_MoveVelocity_1(AXIS_REF AxisIn, BOOL Execute, REAL Velocity, REAL Acceleration, REAL Deceleration, REAL Jerk, SINT DirectionIn, SINT BufferMode)
Type : MC_MoveVelocity, Commands a never ending controlled motion at a specified velocity.

结果

Micro850 - 轴监视器

轴名称(X):
轴状态:
轴已归位:
移动:
错误描述:

Axis1
连续运动
否
恒定速率

位置和速率

命令位置: 1330.095 mm
目标位置: 0.0 mm

命令速率:
目标速率:

变量监视器

全局变量 - Micro850 本地变量-N/A 系统变量 - Micro850

名称	逻辑值	物理值	锁定	
Execute_Home	☒	N/A	☐	B
Position_Home	-50000.0	N/A	☐	R
HomeMode_Home	1	N/A	☐	S
BufferMode_Home	0	N/A	☐	S
Done_Home	☐	N/A	☐	B
Busy_Home	☒	N/A	☐	B
Active_Home	☒	N/A	☐	B
CommandAbort_Hc	☐	N/A	☐	B
Error_Home	☐	N/A	☐	B
ErrorID_Home	0	N/A	☐	U

另请参见

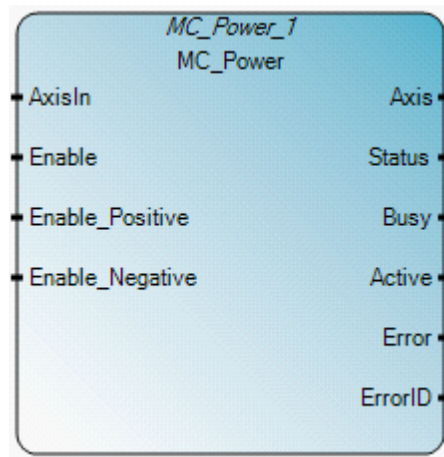
[运动控制指令](#) 参考页数 [451](#)

[运动控制功能块参数详细信息](#) 参考页数 [455](#)

[运动控制功能块的一般规则](#) 参考页数 [452](#)

MC_Power

MC_Power 控制功率（打开或关闭）



MC_Power 操作

- 轴通电完成后，轴 Homed 状态会重置为 0（未归位）。
- MC_Power 功能块的 Enable_Positive 输入和 Enable_Negative 输入都是级别触发的，Enable 输入由 OFF 变为 ON 时会对其进行检查。未经 Enable 输入切换的 Enable_Positive 输入和 Enable_Negative 输入的即时更改未经检查。
- 如果操作过程（检测到伺服就绪时）中出现断电，则轴状态将变为 ErrorStop。
- 如果调用 Enable 设为 True 的 MC_Power 功能块而轴状态为 Disabled，则轴状态会在未出现错误时变为 StandStill；在出现错误时变为 ErrorStop。
- 每个轴仅可发布一个 MC_Power 功能块。运动引擎会拒绝使用不同的 MC_Power 功能块同时控制相同的轴。
- 轴存在电源开/关状态切换时，则不会重置绝对轴位置。

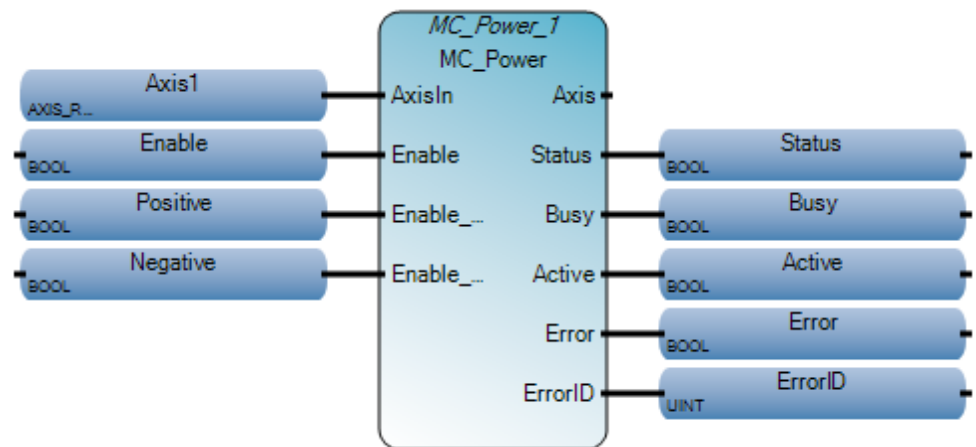
- 如果调用 Enable 设置为 False 的 MC_Power 功能块，则轴所有状态（包括 ErrorStop）变为 Disabled。MC_Power 功能块可以完成以下任务：
- 在 Enable 设置为 True 时接通轴的电源；为 False 时切断轴的电源。

参数

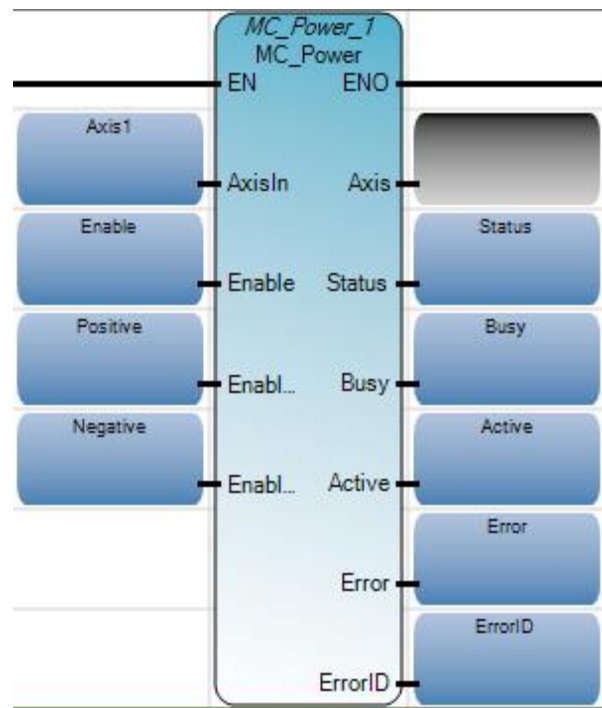
参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_Power 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
启用	输入	BOOL	当为 TRUE 时，开启电源。
Enable_Positive	输入	BOOL	当为 TRUE 时，运动方向仅为正。
Enable_Negative	输入	BOOL	当为 TRUE 时，运动方向仅为负。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
状态	输出	BOOL	功率的状态： • 当为 TRUE 时，驱动器通电完成
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Active	输出	BOOL	当为 TRUE 时，表示功能块可控制轴。
Error	输出	BOOL	当为 TRUE 时，表示检测到一个错误。
ErrorID	输出	UINT	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460。

MC_Power 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



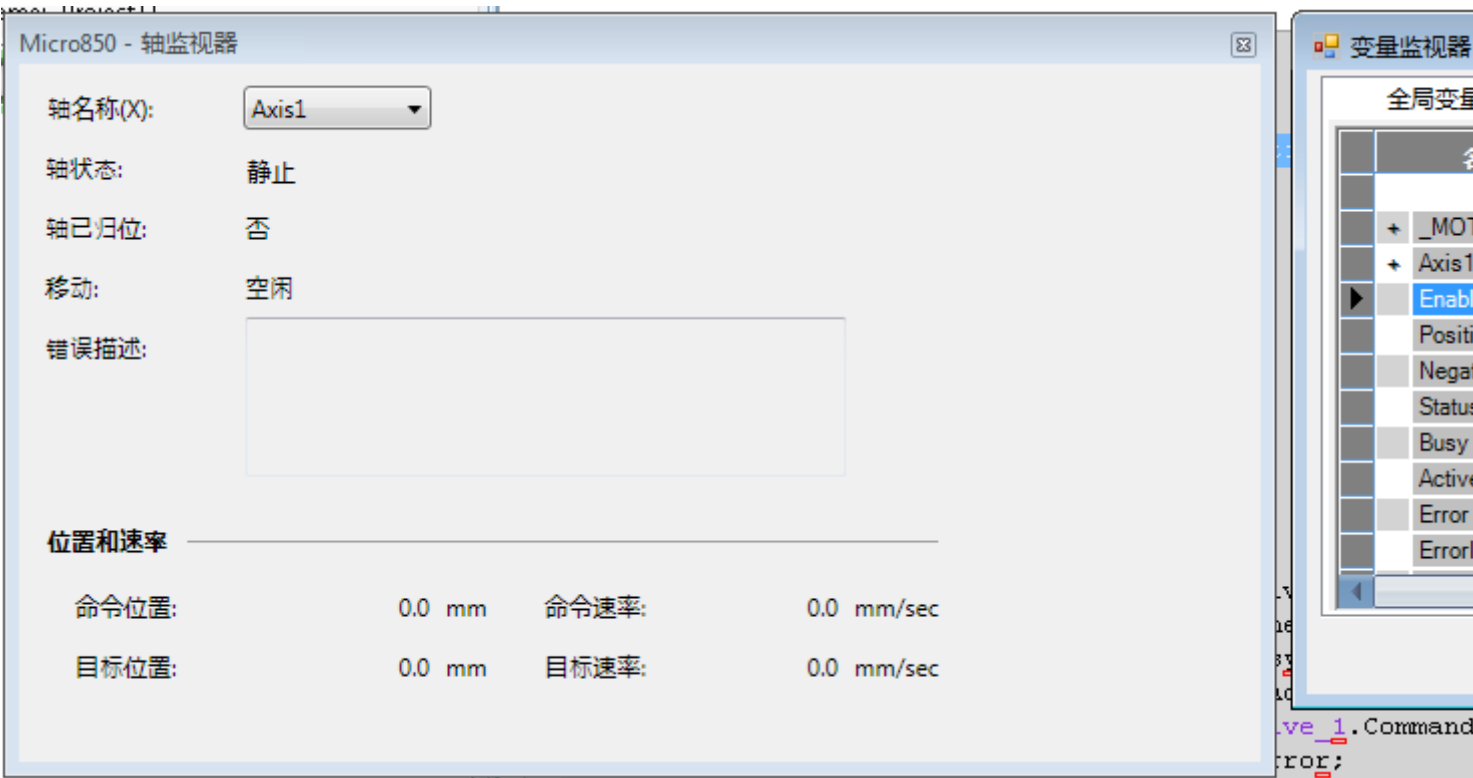
结构化文本 (ST)

```
Positive := True;
Negative := True;
MC_Power_1(Axis1, Enable, Positive, Negative);
Status := MC_Power_1.Status;
Busy := MC_Power_1.Busy;
Active := MC_Power_1.Active;
Error := MC_Power_1.Error;
ErrorID := MC_Power_1.ErrorID;
```

MC_Power_1(

void MC_Power_1(Axis1: AXIS_REF; Enable: BOOL; Enable_Positive: BOOL; Enable_Negative: BOOL)
Type : MC_Power, Controls the power stage (On or Off)

结果



另请参见

[运动控制指令](#) 参考页数 451

[运动控制功能块参数详细信息](#) 参考页数 455

[运动控制功能块的一般规则](#) 参考页数 [452](#)

MC_ReadActualPosition

MC_ReadActualPosition 返回轴的实际位置。

提示 MC_ReadActualPosition 功能仅适用于反馈运动。
：

MC_ReadActualPosition 操作

执行之前 MC_ReadActualPosition 功能块，请确认轴处于以下轴状态之一：

- 已禁用
- 静止
- 离散运动
- 错误停止

参数

参数	参数类型	数据类型	说明
AxisIn	输入	FB_AXIS_REF	另请参见 FB_AXIS_REF 数据类型 参考页数 464 。
启用	输入	BOOL	在启用后持续获取参数值。
Axis	输出	FB_AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 FB_AXIS_REF 数据类型 参考页数 464 。
Valid	输出	BOOL	当为 TRUE 时，表示功能块处于活动状态，并且需要新的输出值。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	单位	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460 。
Position	输出	REAL	反馈运动轴的实际绝对位置的值。（以 [u] 为轴单位）

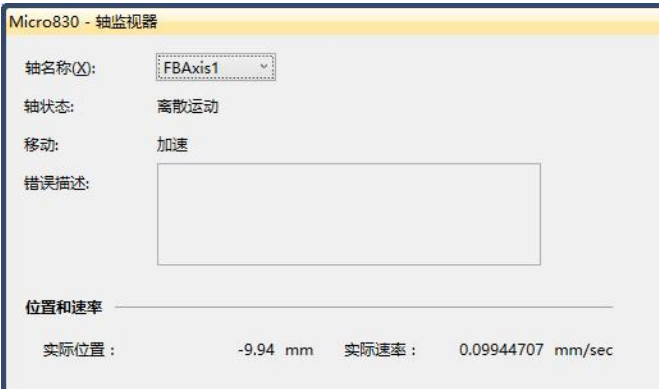
MC_ReadActualPosition 功能块示例

功能块图 (FBD)

梯形图 (LD)

结构化文本 (ST)

结果



名称		别名
MC_ReadActualPosition_2		
+	MC_ReadActualPosition_2.AxisIn	Axis
	MC_ReadActualPosition_2.Enable	En
+	MC_ReadActualPosition_2.Axis	Axis
	MC_ReadActualPosition_2.Valid	Val
	MC_ReadActualPosition_2.Busy	Busy
	MC_ReadActualPosition_2.Error	Err
	MC_ReadActualPosition_2.ErrorID	ErID
	MC_ReadActualPosition_2.Position	Pos

另请参见

- [运动控制指令](#) 参考页数 [451](#)
- [运动控制功能块参数详细信息](#) 参考页数 [455](#)
- [运动控制功能块的一般规则](#) 参考页数 [452](#)

MC_ReadActualVelocity

MC_ReadActualVelocity 返回轴的实际速率。

city

提示 MC_ReadActualVelocity 功能仅适用于反馈运动。
：

MC_ReadActualVelocity 操作

只要 Enable 设置为“真”，MC_ReadActualVelocity 功能块就会返回实际速率值。如果重置 Enable，则数据不再有效，所有输出都将重置为 0，Valid 设置为 False。

执行之前 MC_ReadActualVelocity 功能块，请确认轴处于以下轴状态之一：

- 静止
- 离散运动
- 错误停止

参数

参数	参数类型	数据类型	说明
AxisIn	输入	FB_AXIS_REF	另请参见 FB_AXIS_REF 数据类型 参考页数 464 。
启用	输入	BOOL	在启用后持续获取参数值。
Axis	输出	FB_AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 FB_AXIS_REF 数据类型 参考页数 464 。
Valid	输出	BOOL	当为 TRUE 时，表示功能块处于活动状态，并且需要新的输出值。

参数	参数类型	数据类型	说明
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	单位	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460。
Position	输出	REAL	反馈运动轴的实际绝对位置的值。（以 [u] 为轴单位）

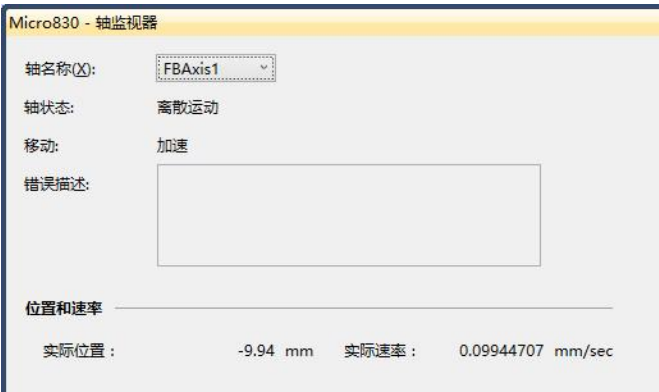
CTD 功能块语言示例

功能块图 (FBD)

梯形图 (LD)

结构化文本 (ST)

结果

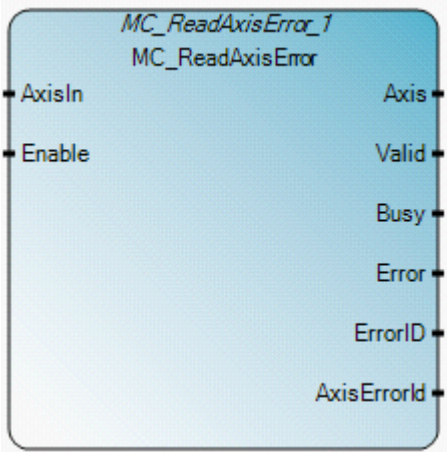


	名称	别名
	MC_ReadActualPosition_1	
	MC_ReadActualPosition_1.AxisIn	Axis
	MC_ReadActualPosition_1.Enable	En
	MC_ReadActualPosition_1.Axis	Axis
	MC_ReadActualPosition_1.Valid	Val
	MC_ReadActualPosition_1.Busy	Busy
	MC_ReadActualPosition_1.Error	Err
	MC_ReadActualPosition_1.ErrorID	ErID
	MC_ReadActualPosition_1.Position	Pos

另请参见

- [运动控制指令](#) 参考页数 451
- [运动控制功能块参数详细信息](#) 参考页数 455
- [运动控制功能块的一般规则](#) 参考页数 452

MC_ReadAxisError MC_ReadAxisError 描述了与运动控制功能块无关的一般轴错误。



MC_ReadAxisError 操作

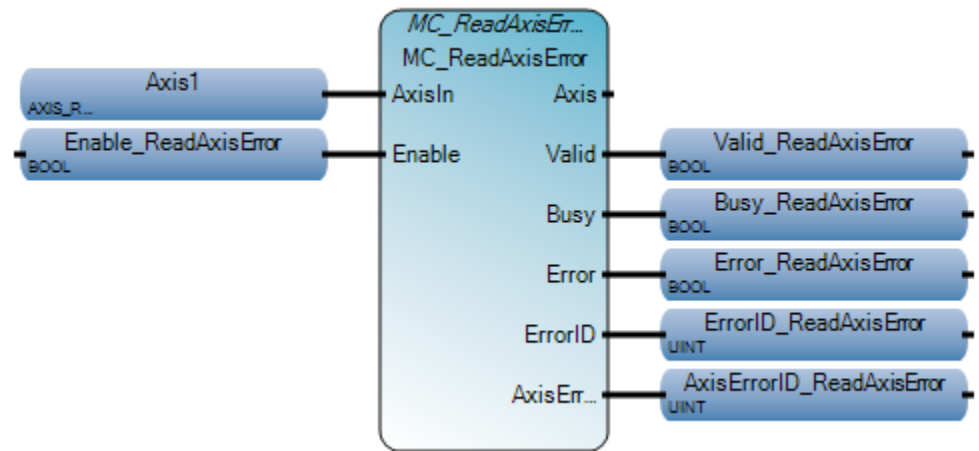
- 由于 Disabled 状态的轴可以包含错误，也可以是无误的，所以当轴处于 Disabled 状态时，MC_ReadAxisError 功能块可能也可能不针对此轴获得非零的错误 ID。
- 当 MC_ReadAxisError 功能块的 Enable 输入设为 False 时，则 Error、ErrorID 和 AxisErrorID 输出都被重置为 False 或 0。

参数

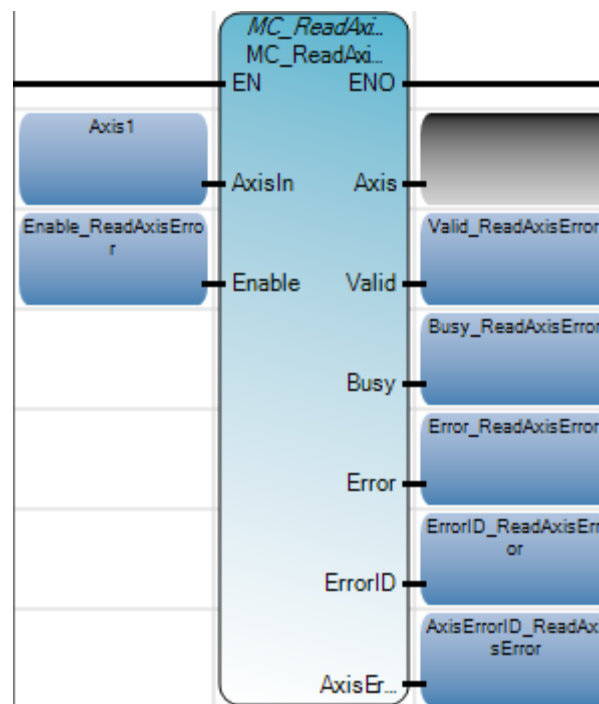
参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_ReadAxisError 计算。 当 EN = FALSE 时，Error、ErrorID 和 AxisErrorID 被重置为 False (或 0)。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462 。
启用	输入	BOOL	当为 TRUE 时，可连续获取参数的值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
Valid	输出	BOOL	当为 TRUE 时，表示功能块处于活动状态，并且需要新的输出值。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	UINT	错误标识。 另请参见运动控制功能块错误 ID 参考页数 460。
AxisErrorID	输出	UINT	错误标识。请参见 AxisErrorID 错误代码 参考页数 507 。

MC_ReadAxisError 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



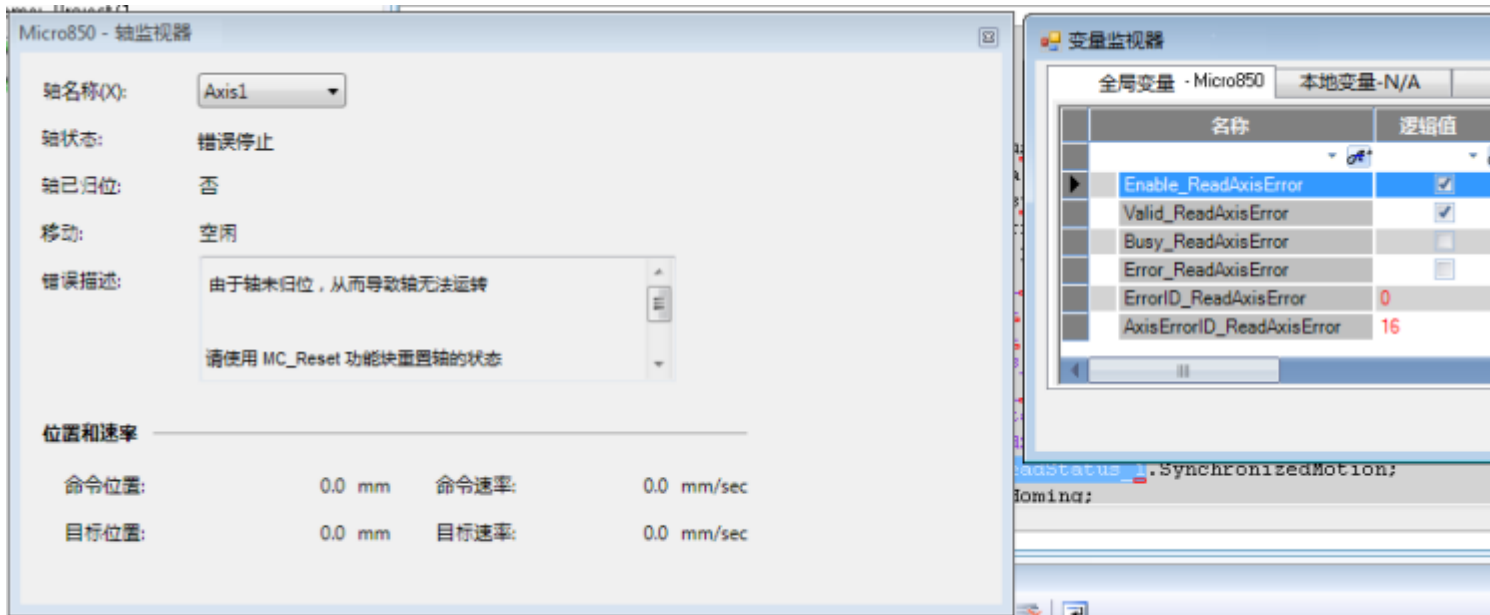
结构化文本 (ST)

```
MC_ReadAxisError_1(Axis1,Enable_ReadAxisError);
Valid_ReadAxisError := MC_ReadAxisError_1.Valid;
Busy_ReadAxisError := MC_ReadAxisError_1.Busy;
Error_ReadAxisError := MC_ReadAxisError_1.Error;
ErrorID_ReadAxisError := MC_ReadAxisError_1.ErrorID;
AxisErrorID_ReadAxisError := MC_ReadAxisError_1.AxisErrorId;
```

MC_ReadAxisError_1(

void **MC_ReadAxisError_1**(AXIS_REF AxisIn, BOOL Enable)
Type : MC_ReadAxisError, Reads the error information for an axis

结果



另请参见

- [AxisErrorID 错误代码](#) 参考页数 507
- [运动控制指令](#) 参考页数 451
- [运动控制功能块参数详细信息](#) 参考页数 455
- [运动控制功能块的一般规则](#) 参考页数 452

AxisErrorID 错误代码

下表描述了 AxisErrorID 中针对 MC_ReadAxis 错误的轴错误代码。

值	MACRO ID	说明
00	MC_FB_ERR_NO	轴处于操作状态（不显示任何内容）。
01	MC_FB_ERR_WRONG_STATE	由于执行功能块的过程中检测到不正确的轴状态，所以轴无法运转。 请使用 MC_Reset 功能块重置轴的状态。
02	MC_FB_ERR_RANGE	由于功能块中设置的无效轴动态参数（Velocity、Acceleration、Deceleration 或 Jerk）而导致轴无法运转。 请使用 MC_Reset 功能块重置轴的状态。 在功能块中，更正与“轴动态”配置页面中的设置相冲突的所有动态参数设置。
03	MC_FB_ERR_PARAM	由于功能块中设置的无效轴参数（除 Velocity、Acceleration、Deceleration 或 Jerk 之外）而导致轴无法运转。 请使用 MC_Reset 功能块重置轴的状态。 在功能块中，更正参数设置（例如模式或位置）。
04	MC_FB_ERR_AXISNUM	运动内部故障，错误 ID = 0x04。 与当地的 Rockwell Automation 技术支持代表联系。有关联系信息，请参见： http://www.rockwellautomation.com/support http://www.rockwellautomation.com/support
05	MC_FB_ERR_MECHAN	由于检测到驱动程序或机械问题，从而导致轴无法运转。 检查驱动程序和控制器之间的连接（指示驱动程序就绪和处于原位信号），并确保驱动程序运行正常。 请使用 MC_Reset 功能块重置轴的状态。
06	MC_FB_ERR_NOPOWER	轴未接通电源。 使用 MC_Power 功能块为轴接通电源。 请使用 MC_Reset 功能块重置轴的状态。
07	MC_FB_ERR_RESOURCE	由于功能块所需的轴或与其相关的资源处于其他功能块的控制之下或者不可用，所以轴无法运转。 确保功能块所需的轴或其相关资源可供使用。 请使用 MC_Reset 功能块重置轴的状态。 如果有同一功能块的多个实例尝试同时控制轴，则检查并更正应用程序。

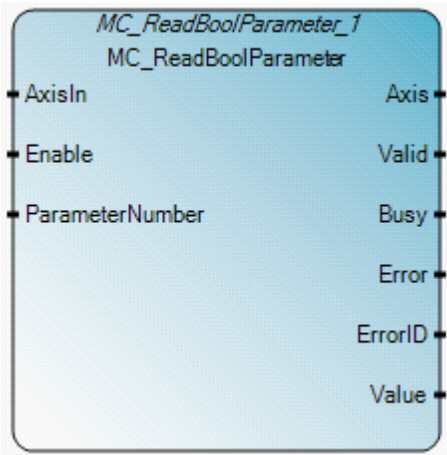
值	MACRO ID	说明
08	MC_FB_ERR_PROFILE	由于功能块中定义的运动配置无效，所以轴无法运转。 请使用 MC_Reset 功能块重置轴的状态。 更正功能块中的配置。
09	MC_FB_ERR_VELOCITY	由于功能块中请求的运动配置与当前轴速率冲突，所以轴无法运转。 可能的原因： • 功能块要求轴在移动时反向进行。 • 当前速率对于请求的运动配置过低或过高。 请使用 MC_Reset 功能块重置轴的状态。 当轴速率与请求的运动配置兼容时，更正功能块中的运动配置或重新执行功能块。
10	MC_FB_ERR_SOFT_LIMIT	由于检测到软限位错误，或者执行功能块将造成软限位错误，所以轴无法运转。 请使用 MC_Reset 功能块重置轴的状态。 检查功能块的速率或目标位置设置，或者调整软限位设置。
11	MC_FB_ERR_HARD_LIMIT	由于检测到硬限位错误而导致轴无法运转。 请使用 MC_Reset 功能块重置轴的状态，然后朝相反方向将轴从硬限位开关中移开。
12	MC_FB_ERR_LOG_LIMIT	由于检测到 PTO 累加器逻辑限制错误，或者执行功能块将造成 PTO 累加器逻辑限制错误，所以轴无法运转。 请使用 MC_Reset 功能块重置轴的状态。 检查功能块的速率或目标位置设置。使用 MC_SetPosition 功能块调整轴坐标系统。
13	MC_FB_ERR_ERR_ENGINE	由于检测到运动引擎执行错误而导致轴无法运转。 断开再接通整个计算机的电源，然后将用户应用程序重新下载到控制器。 如果故障仍然存在，请与当地的 Rockwell Automation 技术支持代表联系。有关联系信息，请参见： http://www.rockwellautomation.com/support http://www.rockwellautomation.com/support。
16	MC_FB_ERR_NOT_HOMED	由于轴未归位，从而导致轴无法运转。 请使用 MC_Reset 功能块重置轴的状态。 使用 MC_Home 功能块对轴执行归位。
128	MC_FB_ERR_PARAM_MODIFIED	运动内部警告，警告 ID = 0x80。 与当地的 Rockwell Automation 技术支持代表联系。有关联系信息，请参见： http://www.rockwellautomation.com/support http://www.rockwellautomation.com/support。

另请参见

[MC_ReadAxisError](#) 参考页数 [503](#)

MC_ReadBoolParameter

MC_ReadBoolParameter 可返回特定于供应商的参数值，数据类型为 BOOL。



MC_ReadBoolParameter 操作

如果将 MC_ReadBoolParameter 功能块的 Enable 输入设为 False，则 Value 输出将被重置为 0。

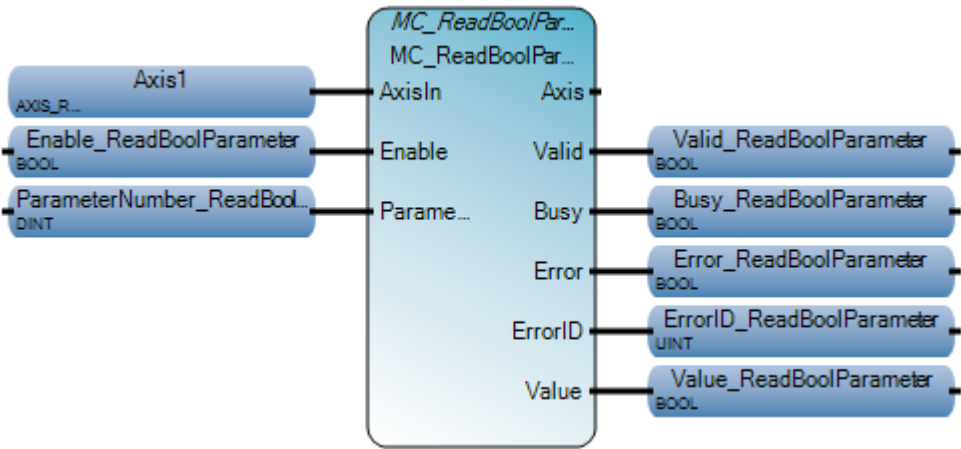
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_ReadBoolParameter 计算。 当 EN = FALSE 时，值输出被重置为 0。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462 。
启用	输入	BOOL	当为 TRUE 时，可连续获取参数的值。
ParameterNumber	输入	DINT	参数标识。 另请参见 运动控制功能块参数编号 参考页数 458 。

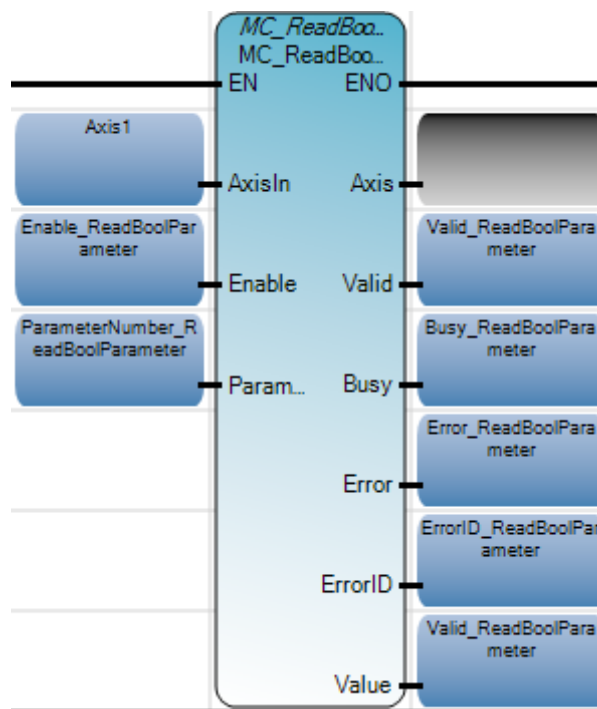
参数	参数类型	数据类型	说明
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
Valid	输出	BOOL	当为 TRUE 时，参数可用。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	UINT	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460。
值	输出	BOOL	由供应商指定数据类型的指定参数值

MC_ReadBoolParameter 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

ParameterNumber_ReadBoolParameter := 4;
MC_ReadBoolParameter_1(Axis1, Enable_ReadBoolParameter, ParameterNumber_ReadBoolParameter);
Valid_ReadBoolParameter := MC_ReadBoolParameter_1.Valid;
Busy_ReadBoolParameter := MC_ReadBoolParameter_1.Busy;
Error_ReadBoolParameter := MC_ReadBoolParameter_1.Error;
ErrorID_ReadBoolParameter := MC_ReadBoolParameter_1.ErrorID;
Value_ReadBoolParameter := MC_ReadBoolParameter_1.Value;

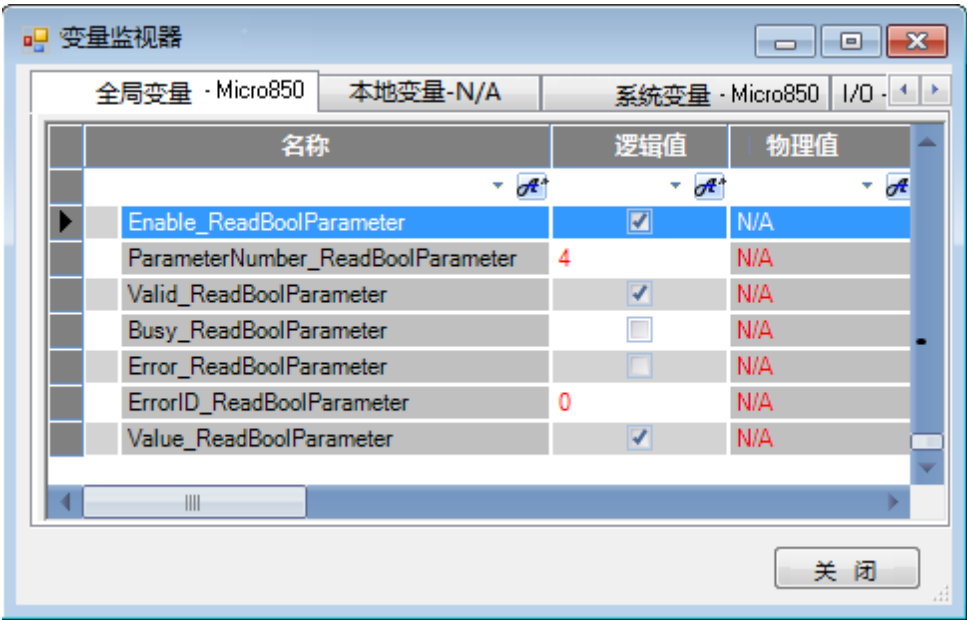
```

```

MC_ReadBoolParameter_1(
void MC_ReadBoolParameter_1(AXIS_REF AxisIn, BOOL Enable, DINT ParameterNumber)
Type : MC_ReadBoolParameter, Returns the value of a motion specific BOOL parameter.

```

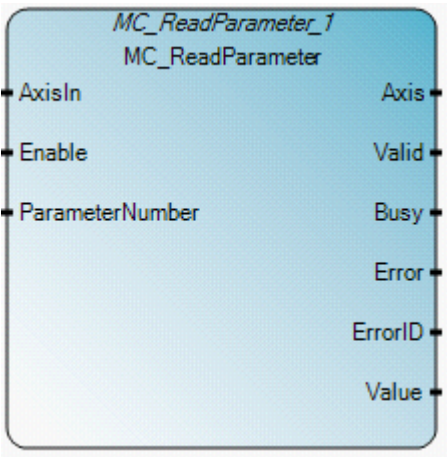
结果



另请参见

- [运动控制指令](#) 参考页数 451
- [运动控制功能块参数详细信息](#) 参考页数 455
- [运动控制功能块的一般规则](#) 参考页数 452

MC_ReadParameter MC_ReadParameter 返回特定于供应商的参数值。



MC_ReadParameter 操作

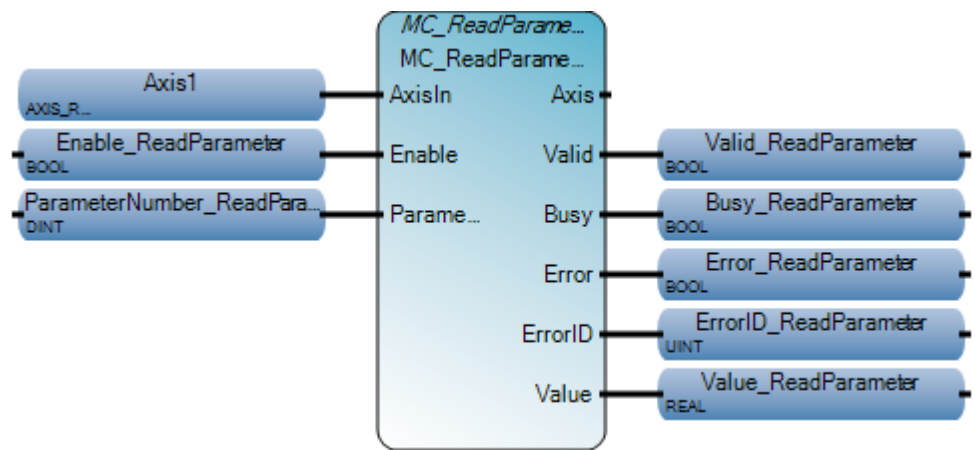
- 如果将 MC_ReadParameter 功能块的 Enable 输入设为 False，那么 Value 输出将被重置为 0。
- 仅支持 REAL 数据类型。

参数

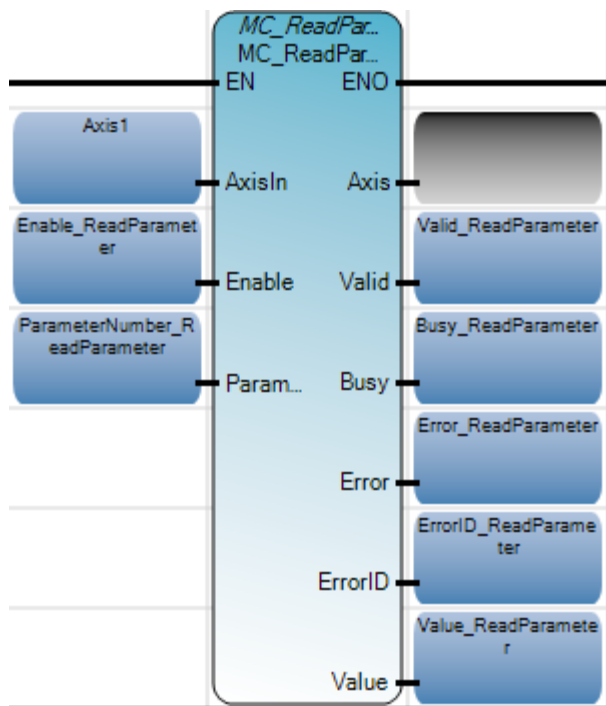
参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_ReadParameter 计算。 当 EN = FALSE 时，值输出被重置为 0。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
启用	输入	BOOL	当为 TRUE 时，可连续获取参数的值。
ParameterNumber	输入	DINT	参数标识。 另请参见 运动控制功能块参数编号 参考页数 458。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
Valid	输出	BOOL	当为 TRUE 时，参数可用。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	UINT	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460。
值	输出	REAL	由供应商指定数据类型的指定参数值

MC_ReadParameter 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

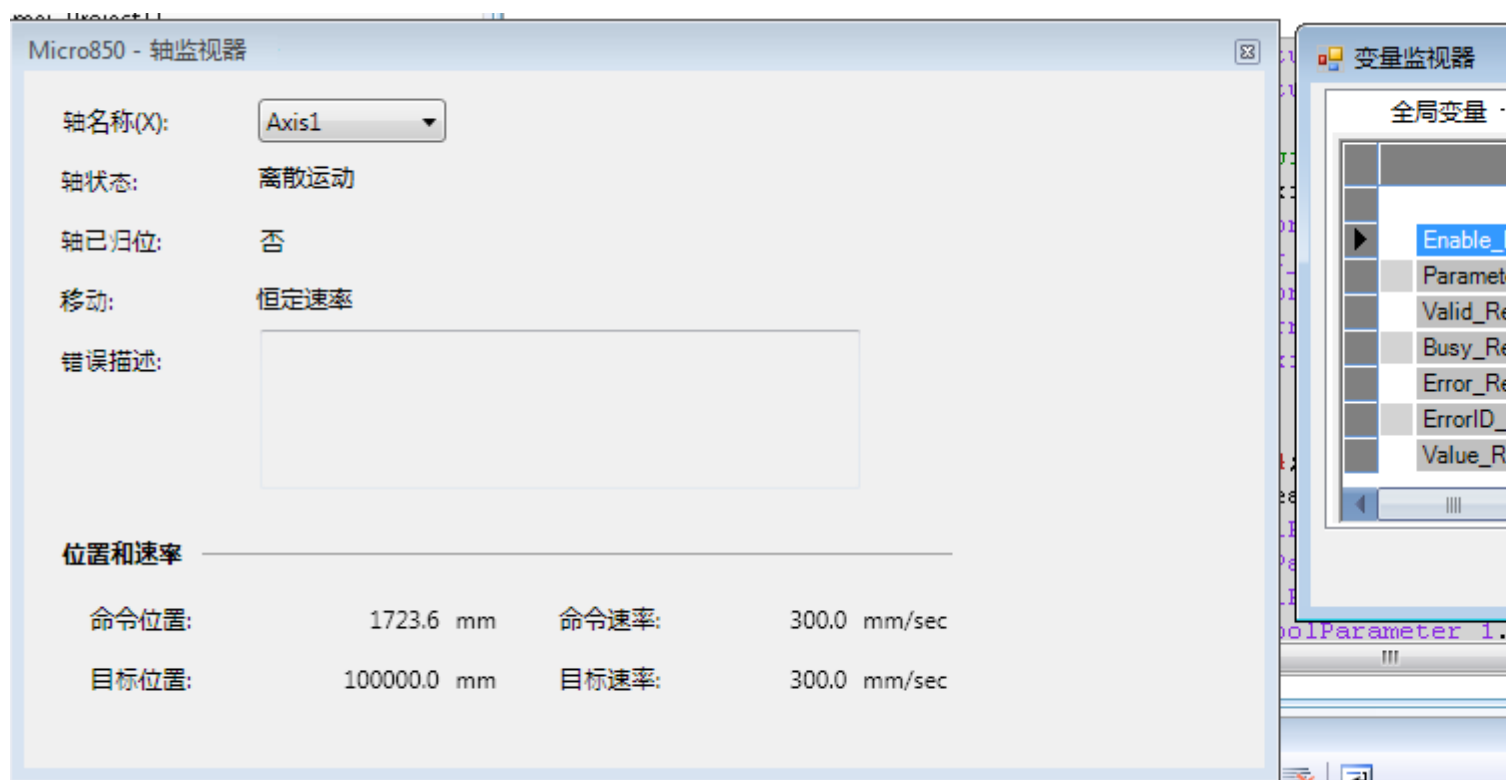
ParameterNumber_ReadParameter := 11;
MC_ReadParameter_1(Axis1, Enable_ReadParameter, ParameterNumber_ReadParameter);
Valid_ReadParameter := MC_ReadParameter_1.Valid;
Busy_ReadParameter := MC_ReadParameter_1.Busy;
Error_ReadParameter := MC_ReadParameter_1.Error;
ErrorID_ReadParameter := MC_ReadParameter_1.ErrorID;
Value_ReadParameter := MC_ReadParameter_1.Value;

```

MC_ReadParameter_1

void MC_ReadParameter_1 (AXIS_REF AxisIn, BOOL Enable, DINT ParameterNumber)
Type : MC_ReadParameter, Returns the value of a motion specific REAL parameter.

结果



另请参见

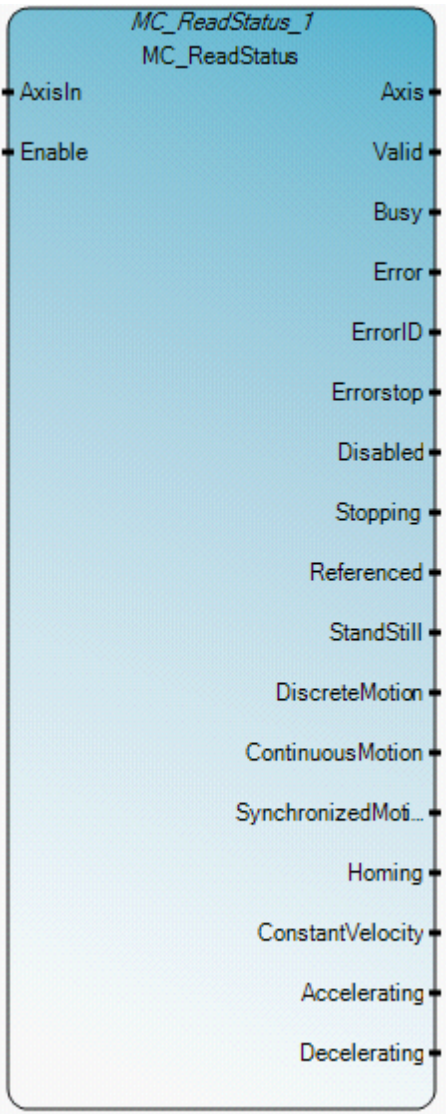
[运动控制指令](#) 参考页数 [451](#)

[运动控制功能块参数详细信息](#) 参考页数 [455](#)

[运动控制功能块的一般规则](#) 参考页数 [452](#)

MC_ReadStatus

MC_ReadStatus 返回与当前正在进行中的运动相关的轴的状态。



MC_ReadStatus 操作

如果将 MC_ReadStatus 功能块的 Enable 设为 False , 则所有状态输出将被重置为 False 或 0。

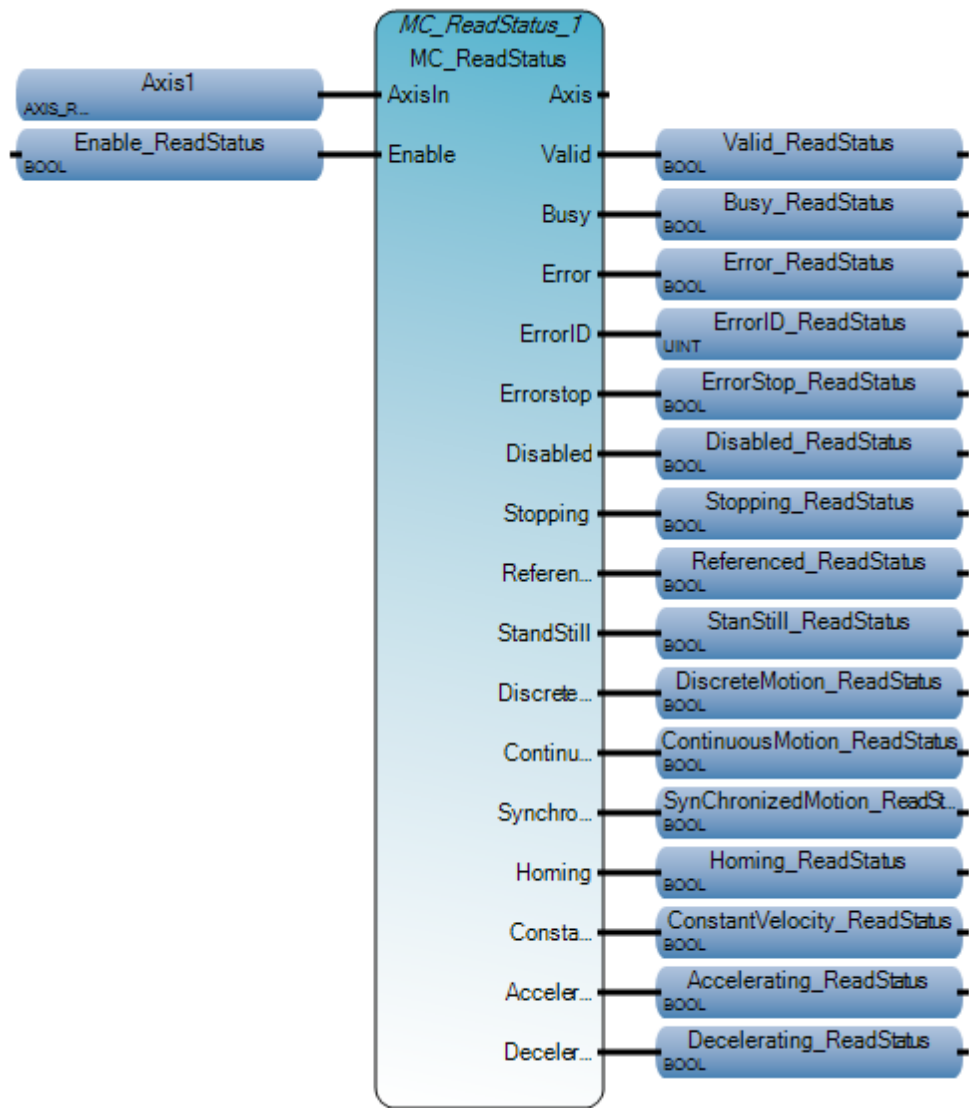
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_ReadStatus 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
启用	输入	BOOL	当为 TRUE 时，可连续获取参数的值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
Valid	输出	BOOL	当为 TRUE 时，有效输出可用。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	UINT	错误标识。 有关错误 ID 编号和描述，请参见运动控制功能块错误 ID 参考页数 460。
ErrorStop	输出	BOOL	当为 TRUE 时，轴状态为 ErrorStop。 请参见运动控制轴状态值和名称。
已禁用	输出	BOOL	当为 TRUE 时，轴状态为 Disabled。
正在停机	输出	BOOL	当为 TRUE 时，轴状态为 Stopping。 请参见运动控制轴状态值和名称。
Referenced	输出	BOOL	当为 TRUE 时，轴的绝对引用位置可知（轴已归位）。
静止	输出	BOOL	当为 TRUE 时，轴状态为 StandStill。 请参见运动控制轴状态值和名称。
DiscreteMotion	输出	BOOL	当为 TRUE 时，轴状态为 DiscreteMotion。 请参见运动控制轴状态值和名称。
ContinuousMotion	输出	BOOL	当为 TRUE 时，轴状态为 ContinuousMotion。 请参见运动控制轴状态值和名称。
SynchronizedMotion	输出	BOOL	此输出始终为 FALSE。 Micro800 控制器中不支持已同步运动。
归位	输出	BOOL	当为 TRUE 时，轴状态为 Homing。 请参见运动控制轴状态值和名称。

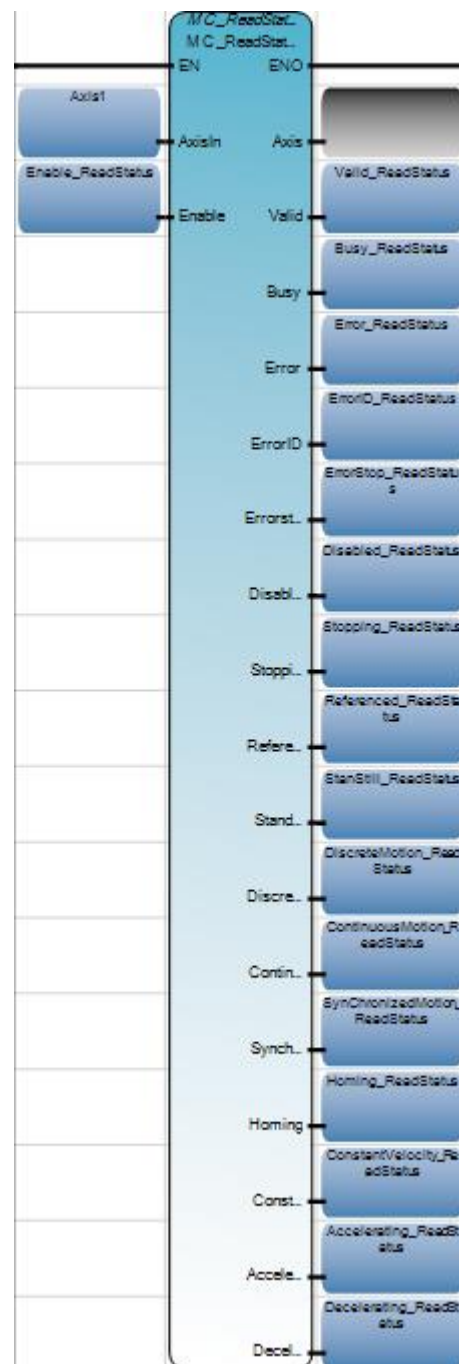
参数	参数类型	数据类型	说明
ConstantVelocity	输出	BOOL	当为 TRUE 时，电机的速率为常量。
加速	输出	BOOL	当为 TRUE 时，可增加电机的能量。
减速	输出	BOOL	当为 TRUE 时，可减少电机的能量。

MC_ReadStatus 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



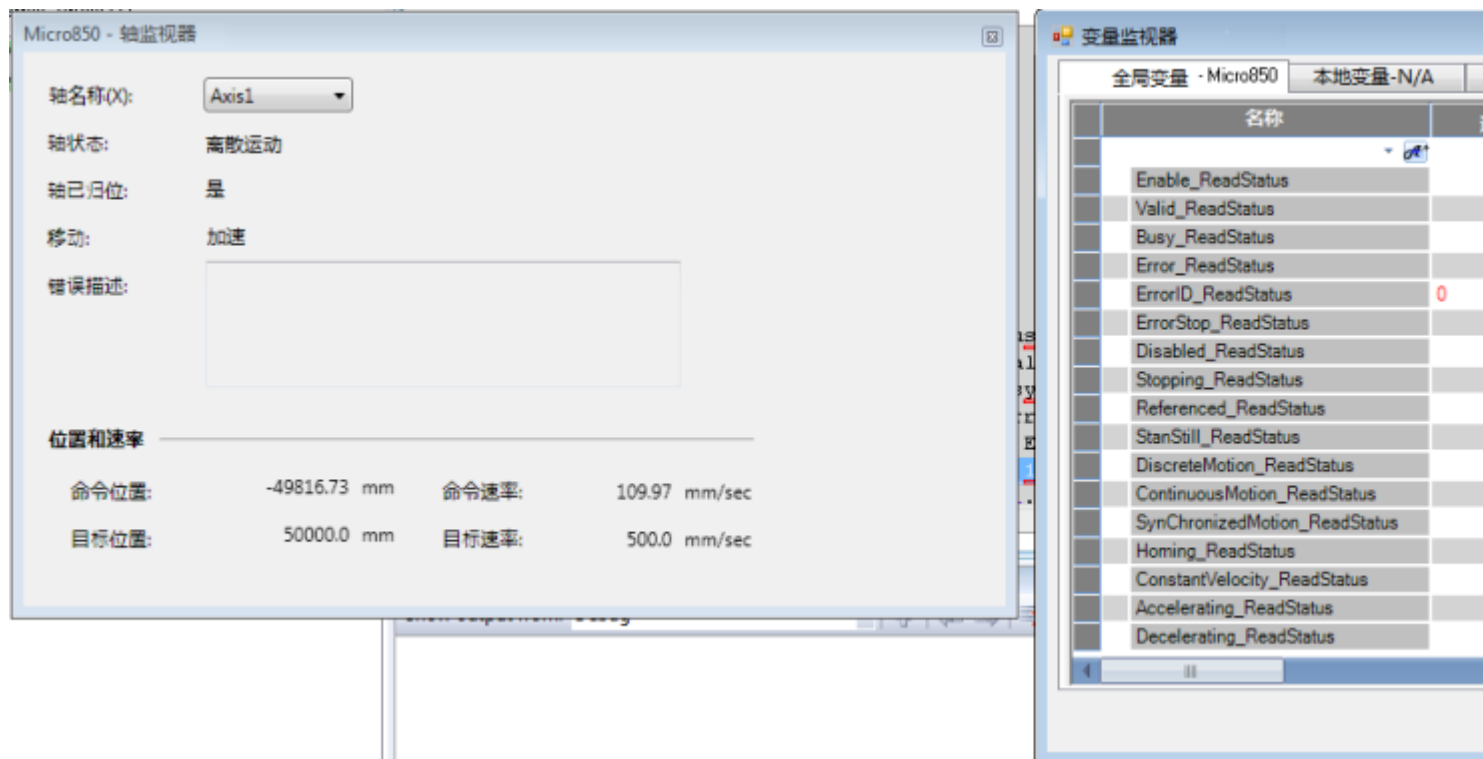
结构化文本 (ST)

```
MC_ReadStatus_1(Axis1,Enable_ReadStatus);
Valid_ReadStatus := MC_ReadStatus_1.Valid;
Busy_ReadStatus := MC_ReadStatus_1.Busy;
Error_ReadStatus := MC_ReadStatus_1.Error;
ErrorID_ReadStatus := MC_ReadStatus_1.ErrorID;
ErrorStop_ReadStatus := MC_ReadStatus_1.Errorstop;
Disabled_ReadStatus := MC_ReadStatus_1.Disabled;
Stopping_ReadStatus := MC_ReadStatus_1.Stopping;
Referenced_ReadStatus := MC_ReadStatus_1.Referenced;
StandStill_ReadStatus := MC_ReadStatus_1.StandStill;
DiscreteMotion_ReadStatus := MC_ReadStatus_1.DiscreteMotion;
ContinuousMotion_ReadStatus := MC_ReadStatus_1.ContinuousMotion;
SynChronizedMotion_ReadStatus := MC_ReadStatus_1.SynChronizedMotion;
Homing_ReadStatus := MC_ReadStatus_1.Homing;
ConstantVelocity_ReadStatus := MC_ReadStatus_1.ConstantVelocity;
Accelerating_ReadStatus := MC_ReadStatus_1.Accelerating;
Decelerating_ReadStatus := MC_ReadStatus_1.Decelerating;
```

```
MC_ReadStatus_1(
```

```
void MC_ReadStatus_1(AxisIn, BOOL Enable)
Type : MC_ReadStatus, Returns in detail the status of the axis with respect to the motion currently in progress.
```

结果

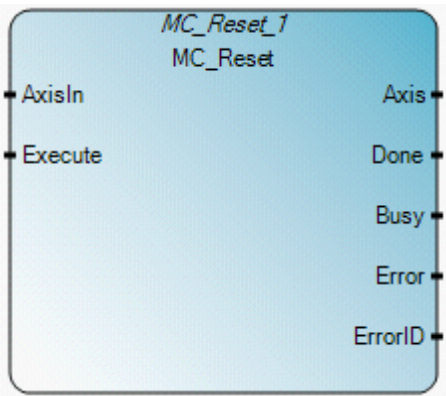


另请参见

- [运动控制指令](#) 参考页数 451
- [运动控制功能块参数详细信息](#) 参考页数 455
- [运动控制功能块的一般规则](#) 参考页数 452

MC_Reset

MC_Reset 可通过重置所有内部轴相关错误，将轴状态从 ErrorStop 转换为 StandStill。功能块实例的输出不会改变。



MC_Reset 操作

MC_Reset 功能块仅可以将轴的状态由 ErrorStop 重置为 StandStill。处于其他状态 (包括 Disabled) 的 MC_Reset 功能块的应用程序会导致出现错误，但对正在进行的运动或轴的状态没有影响。

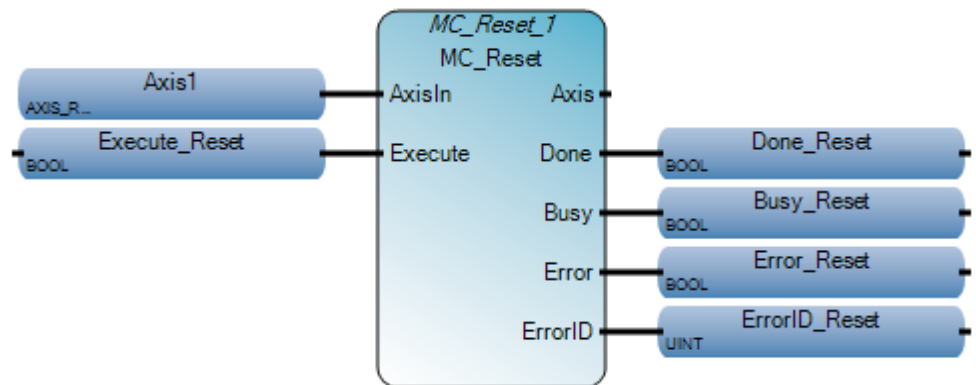
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_Reset 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
Execute	输入	BOOL	当为 TRUE 时，将轴重置在上升沿。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
Done	输出	BOOL	当为 TRUE 时，轴状态为 StandStill。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Error	输出	BOOL	当为 TRUE 时，表示检测到一个错误。

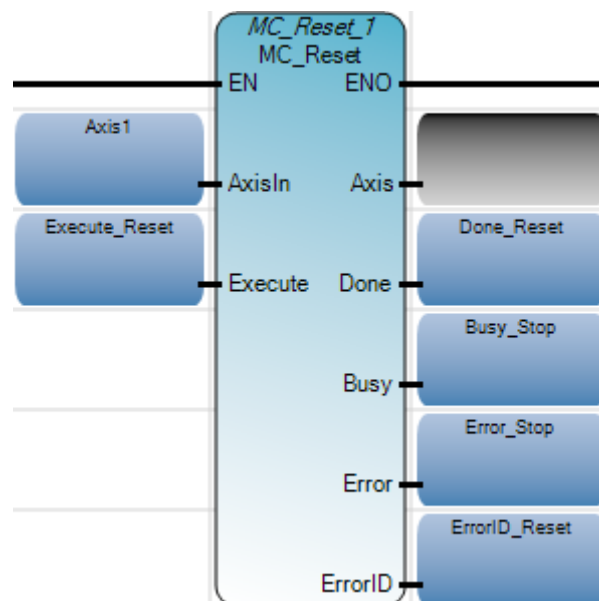
参数	参数类型	数据类型	说明
ErrorID	输出	UINT	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460。

MC_Reset 功能块语言示例

功能块图 (FBD)



梯形图 (LD)

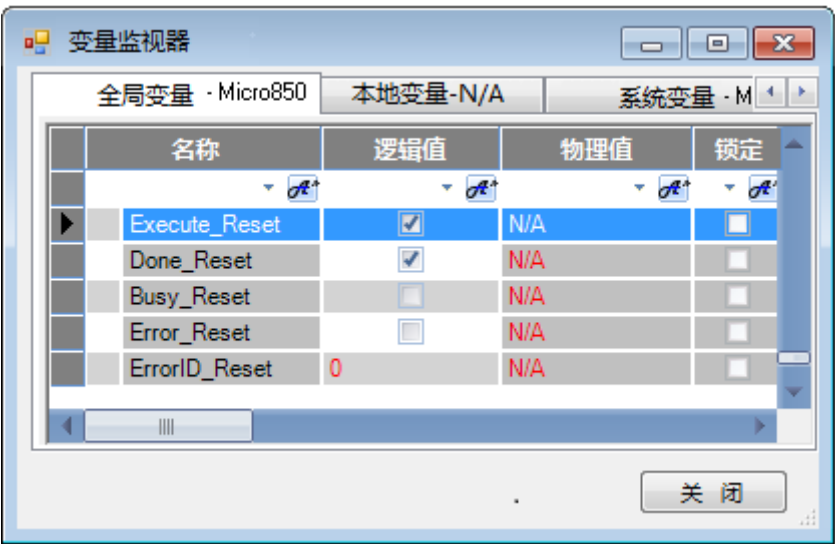


结构化文本 (ST)

```
MC_Reset_1(Axis1,Execute_Reset);  
Done_Reset := MC_Reset_1.Done;  
Busy_Reset := MC_Reset_1.Busy;  
Error_Reset := MC_Reset_1.Error;  
ErrorID_Reset := MC_Reset_1.ErrorID;
```

MC_Reset_1(
void **MC_Reset_1**(AXIS_REF AxisIn, BOOL Execute)
Type : MC_Reset, Resets all internal axis-related errors

结果

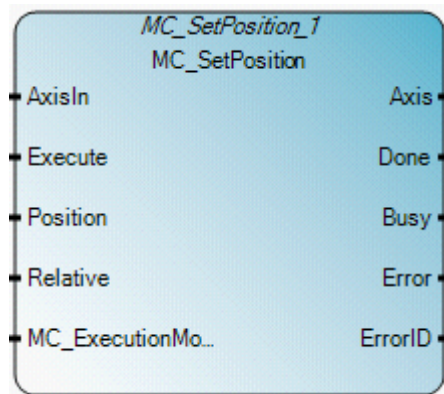


另请参见

- [运动控制指令](#) 参考页数 451
- [运动控制功能块参数详细信息](#) 参考页数 455
- [运动控制功能块的一般规则](#) 参考页数 452

MC_SetPosition

MC_SetPosition 可通过控制具有相同值的轴的实际位置而不造成任何移动，从而转移轴坐标系。



MC_SetPosition 操作

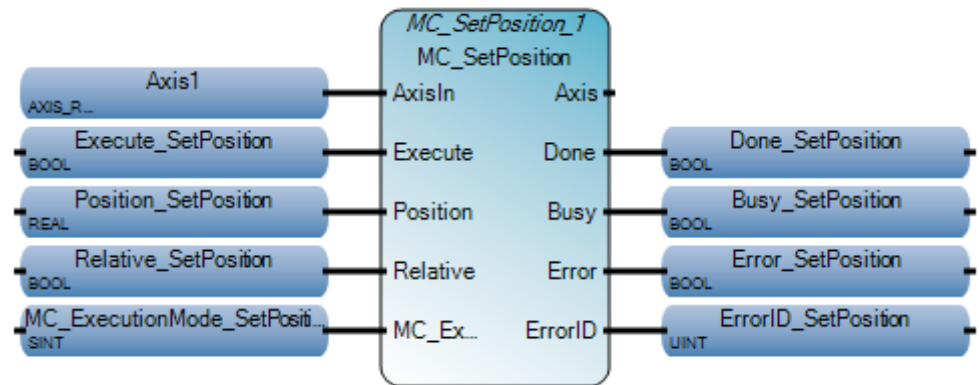
- 只有在轴的状态为 StandStill、continuous Motion (MC_ExecutionMode = 0) 或当正在运行的运动完成且处于 StandStill 状态 (MC_ExecutionMode = 1) 时，MC_SetPosition 功能块才可以成功完成。
- 除非 MC_Home 功能块将轴设置为 Homed 状态，否则 MC_SetPosition 功能块在 HomingMode = MC_HOME_DIRECT (0x04) 时与 MC_Home 的运行方式相同。
- 当 MC_ExecutionMode = 0 (mcImmediately) 时，如果存在正在运行的非连续运动的轴，则 MC_SetPosition 功能块的执行会报告错误。
- MC_ExecutionMode = 1 (mcQueued) 时，只有所有之前正在运行的运动都停止，才需要设置实际位置。也就是说，每个之前的功能块必须至少有一个等于 TRUE 的 Done、Aborted 或 Error 输出。

参数

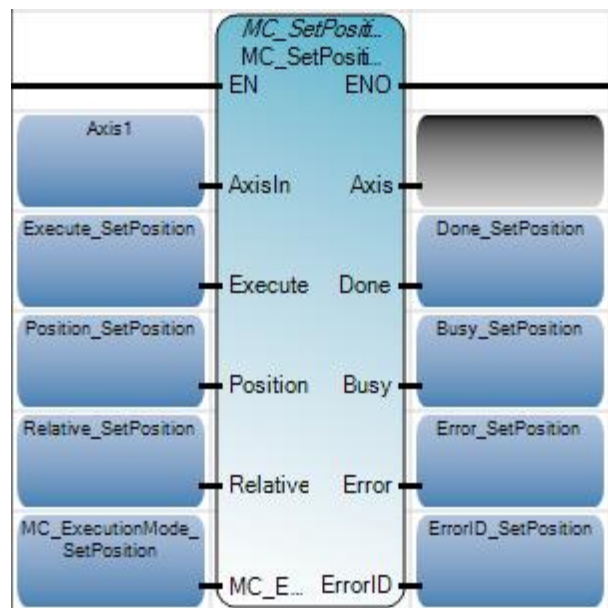
参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_SetPosition 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
Execute	输入	BOOL	当为 TRUE 时，开始设置轴位置。
Position	输入	REAL	轴要设置的绝对位置或相对距离。
Relative	输入	BOOL	当为 TRUE 时，设置轴的相对距离。 当为 FALSE 时，设置轴的绝对位置。
MC_ExecutionMode	输入	SINT	值包括： 0 (<i>mcImmediately</i>) - 功能立即生效。 1 (<i>mcQueued</i>) - 在以下情况下新功能变为有效： - 所有以前的运动命令设置以下输出参数之一： Done、Aborted 或 Error。 - 轴未处于移动状态。 对于 (MC_ExecutionMode = 0)，仅当轴状态为 Disabled 或 Standstill 时，此功能块才能成功完成。如果此模式中存在正在运行的非连续运动的轴，则此功能块的执行将报告错误。 对于 (MC_ExecutionMode = 1)，仅当轴状态为 Disabled、Standstill 或正在运行的运行能以 Standstill 的状态完成时，此功能块才能成功完成。 当前将保留其他输入值，并将其视为无效参数。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
Done	输出	BOOL	当为 TRUE 时，Position 具有新值。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	UINT	错误标识。 另请参见运动控制功能块错误 ID 参考页数 460。

MC_SetPosition 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
Position_SetPosition := 10000.0;
Relative_SetPosition := TRUE;
MC_ExecutionMode_SetPosition := 0;
MC_SetPosition_1(Axis1,Execute_SetPosition,Position_SetPosition,Relative_SetPosition,MC_ExecutionMode_SetPosition);
Done_SetPosition := MC_SetPosition_1.Done;
Busy_SetPosition := MC_SetPosition_1.Busy;
Error_SetPosition := MC_SetPosition_1.Error;
ErrorID_SetPosition := MC_SetPosition_1.ErrorID;
```

MC_SetPosition_1

void MC_SetPosition_1(AXIS_REF AxisIn, BOOL Execute, REAL Position, BOOL Relative, SENT MC_ExecutionMode)
Type : MC_SetPosition, Shifts the coordinate system of an axis by manipulating the actual position of the axis with the same value without any movement caused

结果

Micro850 - 轴监视器

轴名称(X):
Axis1

轴状态:
静止

轴已归位:
否

移动:
空闲

错误描述:

位置和速率

命令位置:10000.0 mm

命令速率:0.0 mm/sec

目标位置:10000.0 mm

目标速率:0.0 mm/sec

变量监视器

全局变量 - Micro850

本地变量-N/

名称	逻辑值
Execute_SetPosition	☒
Position_SetPosition	10000.0
Relative_SetPosition	☒
MC_ExecutionMode_Se	0
Done_SetPosition	☒
Busy_SetPosition	☐
Error_SetPosition	☐
ErrorID_SetPosition	0

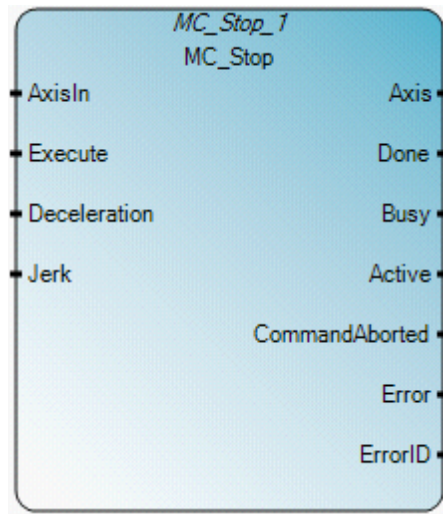
Trigger,Execute_AbortTrigger);
Done;
Busy;
.Error;

另请参见

- [运动控制指令](#) 参考页数 451
- [运动控制功能块参数详细信息](#) 参考页数 455
- [运动控制功能块的一般规则](#) 参考页数 452

MC_Stop

MC_Stop 命令受控制的运动停止，并将轴状态转为 Stopping。任何正在进行的功能块执行都将中止。所有功能块移动命令也就忽略，直至轴状态转变为 Standstill。



MC_Stop 操作

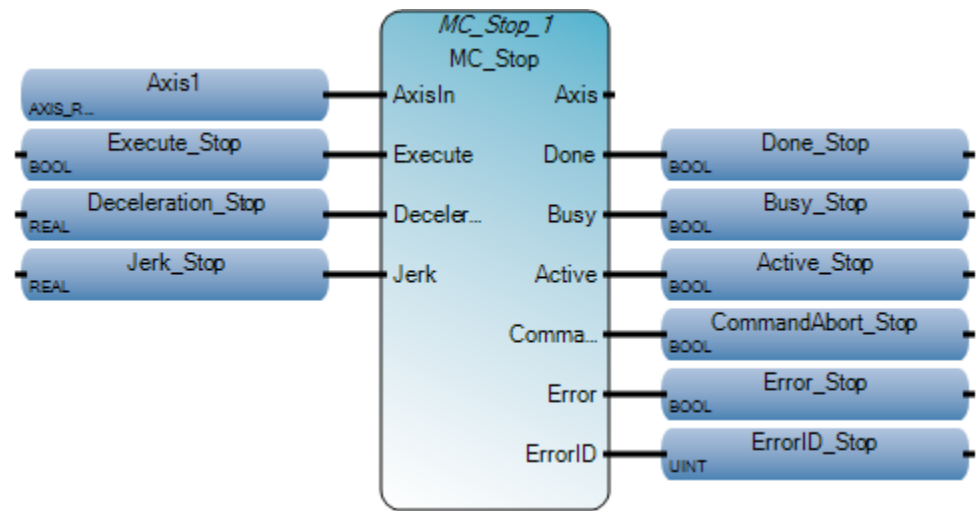
- 只要 Execute 输入高，轴便保持在 Stopping 状态。轴处于 Stopping 状态时，任何其他运动功能块都无法在相同的轴上执行任何运动。
- 如果 Deceleration 等于零，则 MC_Stop 功能块参数取决于轴配置紧急停止设置，包括紧急停止类型、紧急停止减速和紧急停止加速度。
- 如果在停止序列期间未检测到错误，则在 Done 位置位且 Execute 输入改为 False 后，轴的状态会变为 StandStill。
- MC_Stop 功能块主要用于紧急停止功能或异常情况。对于正常运动停止，通常会使用 MC_Halt 功能块。

参数

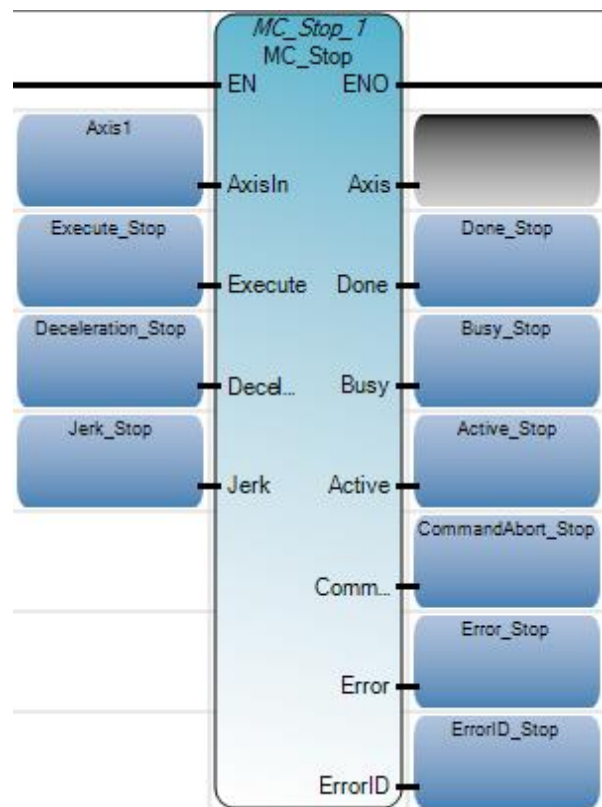
参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时, 执行当前 MC_Stop 计算。 当 EN = FALSE 时, 不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462 。
Execute	输入	BOOL	当为 TRUE 时, 在上升沿开始运动。
减速	输入	REAL	减速的值 [u/s²]。
加加速度	输入	REAL	加加速度的值 [u/s³]。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462 。
Done	输出	BOOL	当为 TRUE 时, 将达到零速率, 并且在停止序列期间没有出现错误。
Busy	输出	BOOL	当为 TRUE 时, 功能块未完成。
Active	输出	BOOL	当为 TRUE 时, 表示功能块可控制轴。
CommandAborted	输出	BOOL	当为 TRUE 时, 表示命令已被 MC_Power(OFF) 或 ErrorStop 功能块中止。
Error	输出	BOOL	当为 TRUE 时, 表示检测到错误。
ErrorID	输出	UINT	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460 。

MC_Stop 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
Deceleration_Stop := 10.0;
Jerk_Stop := 10.0;
MC_Stop_1(Axis1,Execute_Stop,Deceleration_Stop,Jerk_Stop);
Done_Stop := MC_Stop_1.Done;
Busy_Stop := MC_Stop_1.Busy;
Active_Stop := MC_Stop_1.Active;
CommandAbort_Stop := MC_Stop_1.CommandAborted;
Error_Stop := MC_Stop_1.Error;
ErrorID_Stop := MC_Stop_1.ErrorID;
```

MC_Stop_1(

void **MC_Stop_1**(**AXIS_REF** AxisIn, **BOOL** Execute, **REAL** Deceleration, **REAL** Jerk)
Type : MC_Stop, Commands a controlled motion stop and transfers the axis to the state Stopping.

结果



另请参见

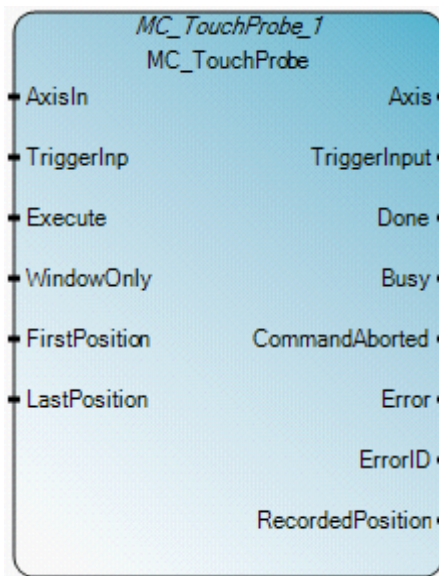
[运动控制指令](#) 参考页数 [451](#)

[运动控制功能块参数详细信息](#) 参考页数 455

[运动控制功能块的一般规则](#) 参考页数 452

MC_TouchProbe

MC_TouchProbe 记录在触发事件时的轴位置。



MC_TouchProbe 操作

- 如果窗口方向（第一个位置 --> 最后一个位置）与运动方向相反，则不会激活触摸探针窗口。
- 如果窗口设置（FirstPosition 或 LastPosition）无效，则 MC_TouchProbe 功能块会报告错误。
- 如果在同一个轴上发布 MC_TouchProbe 功能块的第二个实例，并且第一个功能块实例处于 Busy 状态，则第二个功能块实例将报告错误。
- 一个轴上只应发布一个 MC_TouchProbe 功能块实例。

参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_TouchProbe 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
TriggerInp	输入	USINT	仅支持嵌入式运动。
Execute	输入	BOOL	当为 TRUE 时，在上升沿开始触摸探针记录。
WindowOnly	输入	BOOL	当为 TRUE 时，仅使用窗口（此处定义）接受触发事件。 运动分辨限制为运动引擎间隔，由用户来配置。 对于 WindowOnly TouchProbe 功能，FirstPosition 和 LastPosition 激活均存在与运动引擎间隔相等的最大响应时间延迟。 触发位置（FirstPosition 和 LastPosition）中最大的可能延迟可通过 (运动引擎间隔 * 移动速率) 来计算。
FirstPosition	输入	REAL	触发事件从其接受的窗口的开始位置（以技术单元 [u] 为单位）。窗口中包含值。
LastPosition	输入	REAL	从其未接受触发事件的窗口的停止位置（以技术单元 [u] 为单位）。窗口中包含值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
TriggerInput	输出	USINT	仅支持嵌入式运动。
Done	输出	BOOL	当为 TRUE 时，会记录触发事件。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
CommandAborted	输出	BOOL	当为 TRUE 时，表示命令已被 MC_Power(OFF) 或 Error Stop 功能块中止。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	UINT	错误标识。 另请参见 运动控制功能块错误 ID 参考页数 460。

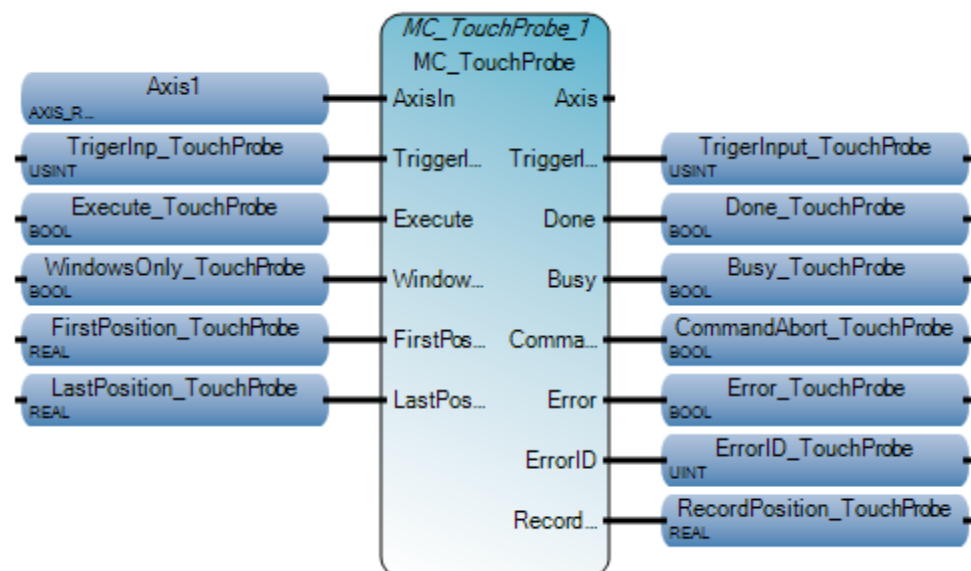
参数	参数类型	数据类型	说明
RecordedPosition	输出	REAL	触发事件发生的位置（以技术单元 [u] 为单位） 运动为开环式运动。 发生触发事件时的轴位置。如果轴运动是开环式运动，则为发生触发事件时的命令位置（不是实际位置），并假定驱动器与电机之间没有运动延迟。

运动固定输入/输出

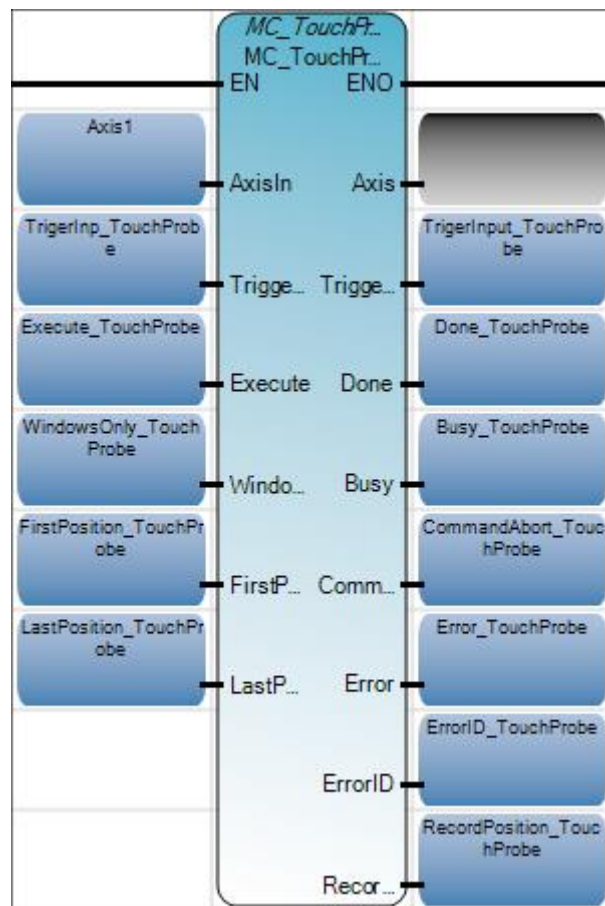
运动信号	PTO0	PTO1	PTO2
PTO 脉冲	Output_0	Output_1	Output2
PTO 方向	Output_3	Output_4	Output_5
较低（负）限制开关	Input_0	Input_4	Input_8
较高（正）限制开关	Input_1	Input_5	Input_9
绝对归位开关	Input_2	Input_6	Input_10
触摸探针输入开关	Input_3	Input_7	Input_11

MC_TouchProbe 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

FirstPosition_TouchProbe := 10000.0;
LastPosition_TouchProbe := 50000.0;
MC_TouchProbe_1(Axis1, TriggerInp_TouchProbe, Execute_TouchProbe, WindowsOnly_TouchProbe,
FirstPosition_TouchProbe, LastPosition_TouchProbe);
Done_TouchProbe := MC_TouchProbe_1.Done;
Busy_TouchProbe := MC_TouchProbe_1.Busy;
Error_TouchProbe := MC_TouchProbe_1.Error;
ErrorID_TouchProbe := MC_TouchProbe_1.ErrorID;
RecordPosition_TouchProbe := MC_TouchProbe_1.RecordedPosition;

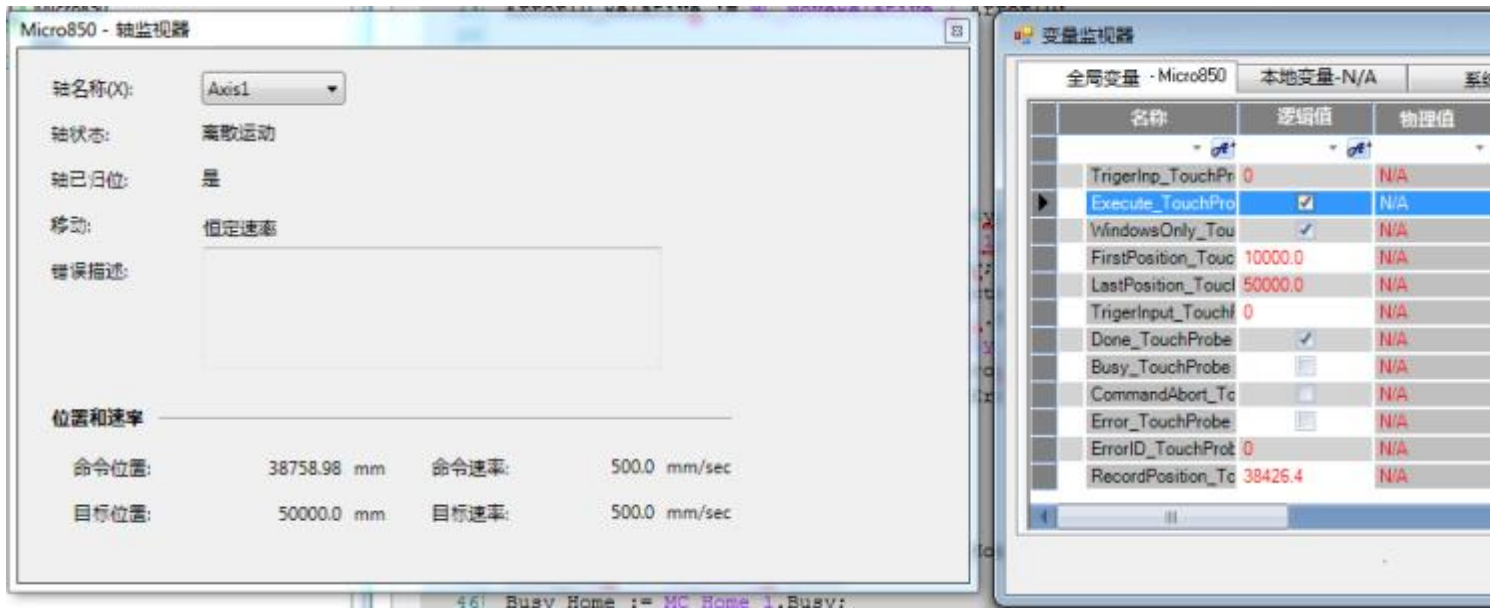
```

```

MC_TouchProbe_1(
void MC_TouchProbe_1(AXIS_REF AxisIn, USINT TriggerInp, BOOL Execute, BOOL WindowOnly, REAL FirstPosition, REAL LastPosition)
Type : MC_TouchProbe, Records an axis position at a trigger event

```

结果



另请参见

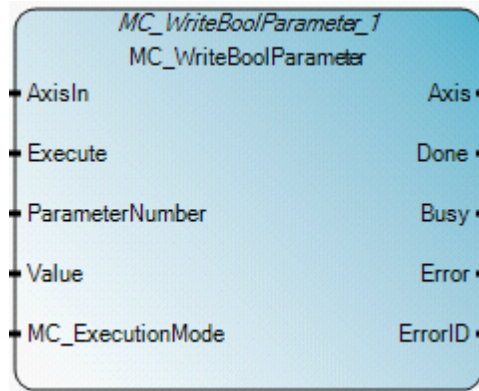
- [运动控制指令](#) 参考页数 451
- [运动控制功能块参数详细信息](#) 参考页数 455
- [运动控制功能块的一般规则](#) 参考页数 452

运动固定输入/输出

运动信号	PTO0	PTO1	PTO2
PTO 脉冲	Output_0	Output_1	Output2
PTO 方向	Output_3	Output_4	Output_5
较低（负）限制开关	Input_0	Input_4	Input_8
较高（正）限制开关	Input_1	Input_5	Input_9
绝对归位开关	Input_2	Input_6	Input_10
触摸探针输入开关	Input_3	Input_7	Input_11

MC_WriteBoolParameter

MC_WriteBoolParameter 修改特定于供应商且类型为 BOOL 的参数的值。



MC_WriteBoolParameter 操作

MC_WriteBoolParameter 功能块设置的参数仅暂时应用于应用程序。将控制器由 PRG 切换到 RUN，或将控制器电源断开再接通时，它们会被用户在 Connect Component Workbench 运动配置中配置的永久设置覆盖。

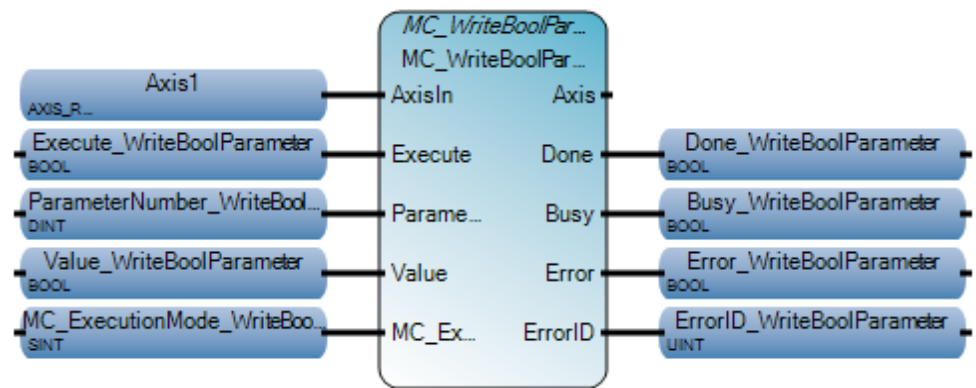
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_WriteBoolParameter 计算。 当 EN = FALSE 时，值输出被重置为 0。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462 。
Execute	输入	BOOL	当为 TRUE 时，在上升沿写入参数的值。
ParameterNumber	输入	DINT	参数 ID。请参见下面“另请参见”中的附加主题。
值	输入	BOOL	当为 TRUE 时，指定的参数具有新值。

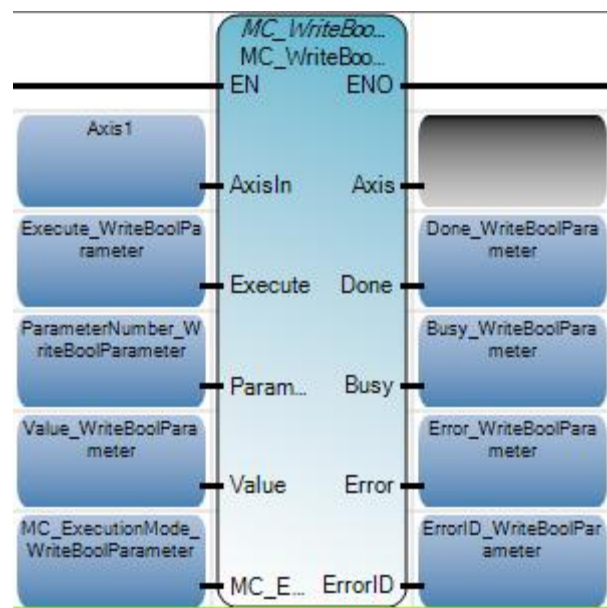
参数	参数类型	数据类型	说明
MC_ExecutionMode	输入	SINT	值包括： • 0 (<i>mcImmediately</i>) - 功能立即生效。 • 1 (<i>mcQueued</i>) - 在以下情况下新功能变为有效： • 所有以前的运动命令设置以下输出参数之一： Done、Aborted 或 Error • 轴未处于移动状态。 注意：当 (MC_ExecutionMode = 0) 时，对于除工作周期 (1005) 以外的所有参数，仅当轴状态为“已禁用”或“静止”时此 FB 才能成功完成。 当 (MC_ExecutionMode = 0) 时，对于工作周期参数 (1005)，除轴处于归位或 ErrorStop 状态以外，此 FB 可以成功完成。 对于 (MC_ExecutionMode = 1)，仅当轴状态为 Disabled、Standstill 或正在运行的运动以 Standstill 的状态完成时，此功能块才能成功完成。 当前将保留其他输入值，并将其视为无效参数。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
Done	输出	BOOL	当为 TRUE 时，参数会成功写入。
Busy	输出	BOOL	当为 TRUE 时，功能块未完成。
Error	输出	BOOL	当为 TRUE 时，表示检测到一个错误。
ErrorID	输出	UINT	错误标识。 另请参见运动控制功能块错误 ID 参考页数 460。

MC_WriteBoolParameter 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
ParameterNumber_WriteBoolParameter := 5;  
MC_WriteBoolParameter_1(Axis1,Execute_WriteBoolParameter,ParameterNumber_WriteBoolParameter,  
Value_WriteBoolParameter,MC_ExecutionMode_WriteBoolParameter);  
Done_WriteBoolParameter := MC_WriteBoolParameter_1.Done;  
Busy_WriteBoolParameter := MC_WriteBoolParameter_1.Busy;  
Error_WriteBoolParameter := MC_WriteBoolParameter_1.Error;  
ErrorID_WriteBoolParameter := MC_WriteBoolParameter_1.ErrorID;
```

MC_WriteBoolParameter_1

void MC_WriteBoolParameter_1(AXIS_REF AxisIn, BOOL Execute, DINT ParameterNumber, BOOL Value, SINT MC_ExecutionMode)
Type : MC_WriteBoolParameter, Modifies the value of a motion specific BOOL parameter

结果

变量监视器

全局变量 · Micro850本地变量-N/A系统变量 · Micro850

名称	逻辑值	物理值
Execute_WriteBoolParameter	☒	N/A
ParameterNumber_WriteBoolParameter	5	N/A
Value_WriteBoolParameter	☐	N/A
MC_ExecutionMode_WriteBoolParameter	0	N/A
Done_WriteBoolParameter	☒	N/A
Busy_WriteBoolParameter	☐	N/A
Error_WriteBoolParameter	☐	N/A
ErrorID_WriteBoolParameter	0	N/A

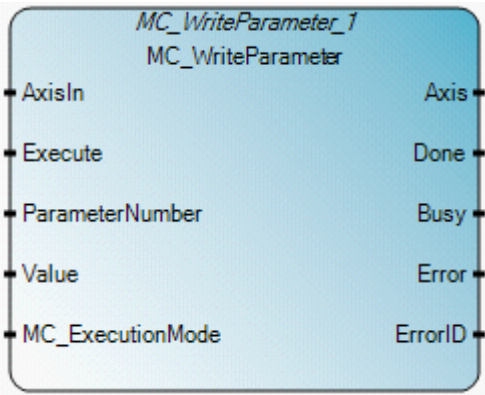
关闭

另请参见

- [运动控制指令](#) 参考页数 451
- [运动控制功能块参数详细信息](#) 参考页数 455
- [运动控制功能块的一般规则](#) 参考页数 452

MC_WriteParameter

MC_WriteParameter 修改特定于供应商的参数的值。



MC_WriteParameter 操作

MC_WriteParameter 功能块设置的参数仅暂时应用于应用程序。将控制器由 PRG 切换到 RUN，或将控制器电源断开再接通时，它们会被用户在 Connect Component Workbench 运动配置中配置的永久设置覆盖。

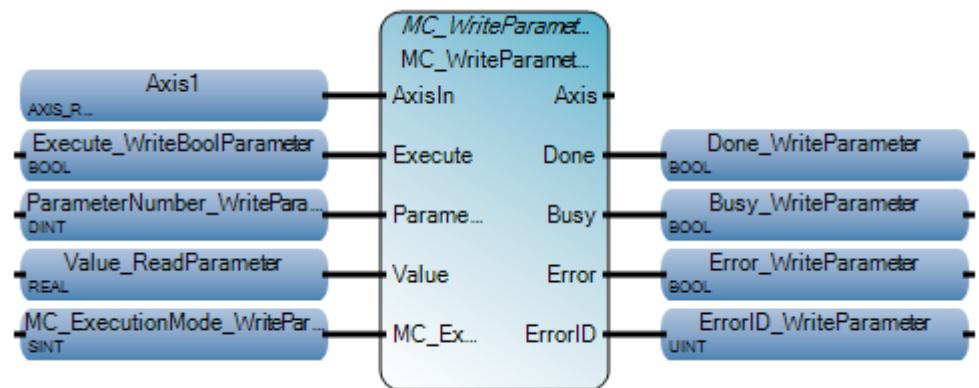
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前 MC_WriteParameter 计算。 当 EN = FALSE 时，不执行计算。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF	另请参见 AXIS_REF 数据类型 参考页数 462。
Execute	输入	BOOL	当为 TRUE 时，在上升沿写入参数的值。
ParameterNumber	输入	DINT	参数标识。 另请参见 运动控制功能块参数编号 参考页数 458。
值	输入	REAL	指定参数的新值。

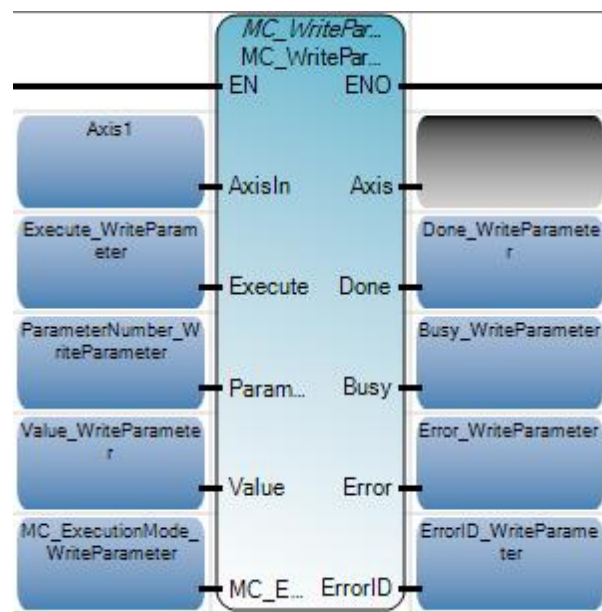
参数	参数类型	数据类型	说明
MC_ExecutionMode	输入	SINT	值包括： • 0 (<i>mcImmediately</i>) - 功能立即生效。 • 1 (<i>mcQueued</i>) - 在以下情况下新功能变为有效： • 所有以前的运动命令设置以下输出参数之一： Done、Aborted 或 Error • 轴未处于移动状态 • 表明输出参数 Busy 设置为 FALSE。 注意：当 (MC_ExecutionMode = 0) 时，对于除工作周期 (1005) 以外的所有参数，仅当轴状态为“已禁用”或“静止”时此 FB 才能成功完成。 当 (MC_ExecutionMode = 0) 时，对于工作周期参数 (1005)，除轴处于归位或 ErrorStop 状态以外，此 FB 可以成功完成。 对于 (MC_ExecutionMode = 1)，仅当轴状态为 Disabled、Standstill 或正在运行的运动以 Standstill 的状态完成时，此功能块才能成功完成。 当前将保留其他输入值，并将其视为无效参数。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Axis	输出	AXIS_REF	LD 程序中的 Axis 输出为只读。 另请参见 AXIS_REF 数据类型 参考页数 462。
Done	输出	BOOL	当为 TRUE 时，参数会成功写入。
Busy	输出	BOOL	当为 TRUE 时，表示功能块可控制轴。
Error	输出	BOOL	当为 TRUE 时，表示检测到错误。
ErrorID	输出	UINT	错误标识。 另请参见运动控制功能块错误 ID 参考页数 460。

MC_WriteParameter 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



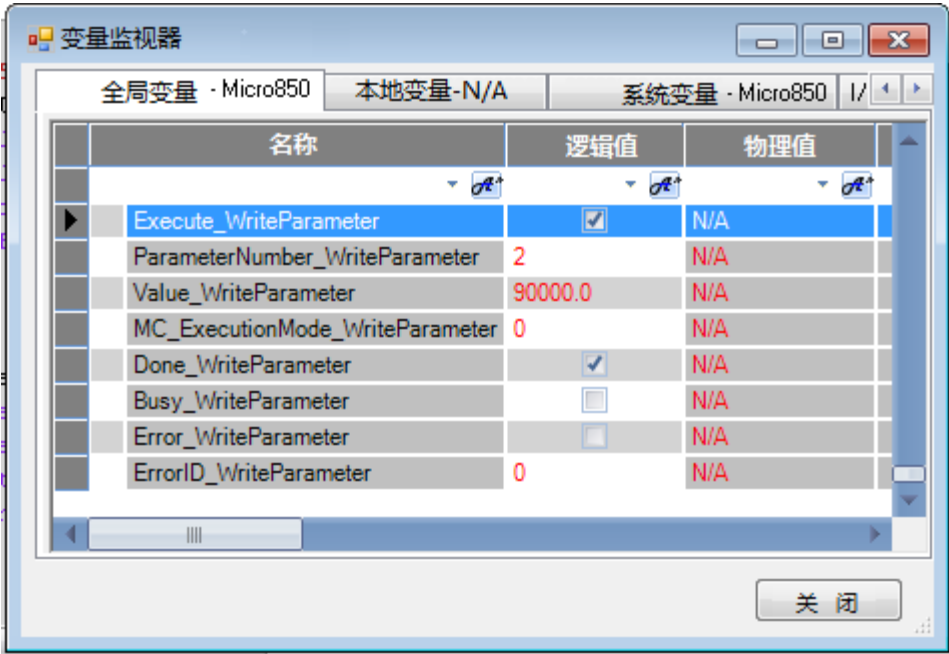
结构化文本 (ST)

```
ParameterNumber_WriteParameter := 2;  
Value_WriteParameter := 90000.0;  
MC_WriteParameter_1(Axis1,Execute_WriteParameter,ParameterNumber_WriteParameter,  
Value_WriteParameter,MC_ExecutionMode_WriteParameter);  
Done_WriteParameter := MC_WriteParameter_1.Done;  
Busy_WriteParameter := MC_WriteParameter_1.Busy;  
Error_WriteParameter := MC_WriteParameter_1.Error;  
ErrorID_WriteParameter := MC_WriteParameter_1.ErrorID;
```

MC_WriteParameter_1(

void MC_WriteParameter_1(AXIS_REF AxisIn, BOOL Execute, DINT ParameterNumber, REAL Value, SINT MC_ExecutionMode)
Type : MC_WriteParameter, Modifies the value of a motion specific REAL parameter

结果



另请参见

- [运动控制指令](#) 参考页数 451
- [运动控制功能块参数详细信息](#) 参考页数 455

[运动控制功能块的一般规则](#) 参考页数 [452](#)

过程控制指令

过程控制指令用于监视和保持流过程循环的几个方面，例如压力、温度、流速以及液面。过程控制通过将输出信号发送到控制阀来调节过程。

功能块	说明
DERIVATE 参考页数 549	实型值的区分
HYSTER 参考页数 551	在不同实型值上的布尔迟滞
INTEGRAL 参考页数 553	随时间进行集成
PWM 参考页数 560	针对配置的通道打开或关闭输出
SCALER 参考页数 564	根据输出范围调整输入值
STACKINT 参考页数 567	整数堆栈
函数	说明
LIMIT 参考页数 572	限制
TND 参考页数 570	停止用户程序扫描的当前循环

DERIVATE

DERIVATE 区分实型值。如果 CYCLE 参数值小于设备的执行周期时间，则将采样周期强制为该周期时间。



Derivate 操作

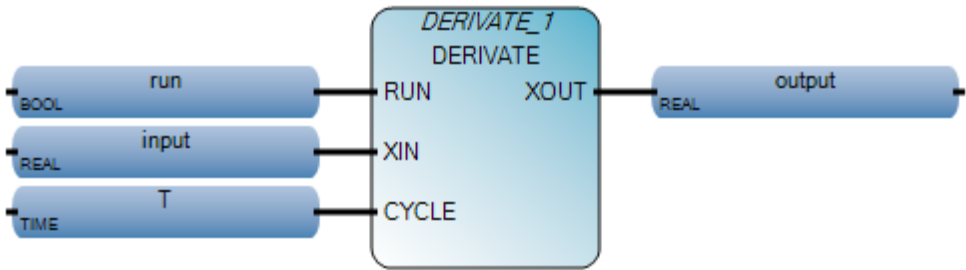
微分是以毫秒为时间基准计算的（也就是说，对在 1 秒钟时限内变为 2000 的输入 1000 求取微分会得到值 1）。要将该指令的输出换算成以秒为单位表示的值，必须将该输出乘以 1000。

参数

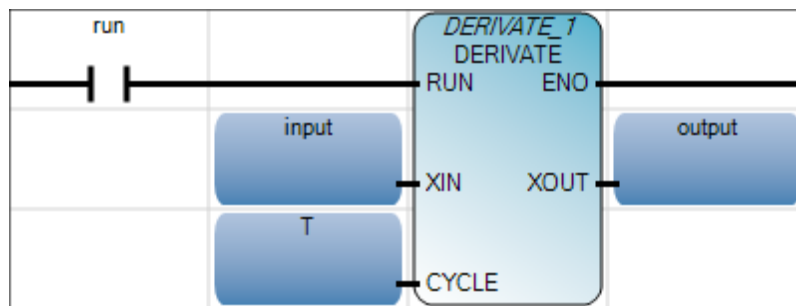
参数	参数类型	数据类型	说明
RUN	输入	BOOL	模式：TRUE = 正常/FALSE = 重置。
XIN	输入	REAL	输入：任何实型值。
CYCLE	输入	TIME	采样周期。可以采用的值的范围为 0ms 到 23h59m59s999ms。
XOUT	输出	REAL	差分输出。
ENO	输出	BOOL	启用输出 仅适用于 LD 编程。

DERIVATE 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1  DERIVATE_1(run, input, T);
2  output := DERIVATE_1.XOUT;

```

```

DERIVATE_1(
void DERIVATE_1(BOOL RUN, REAL XIN, TIME CYCLE)
Type : DERIVATE, Differentiation according to time

```

(* 与之等效的 ST : DERIVATE1 是 DERIVATE 块的实例 *)

```

DERIVATE1(manual_mode, sensor_value, t#100ms);
derivated_value := DERIVATE1.XOUT;

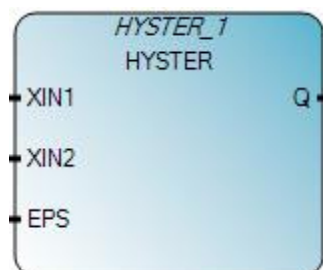
```

另请参见

[过程控制指令](#) 参考页数 [549](#)

HYSTER

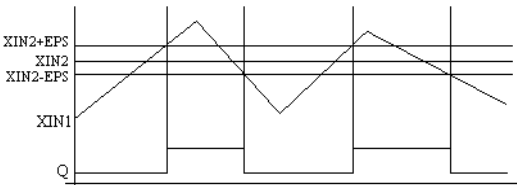
HYSTER 执行上限实型值的滞后。



参数

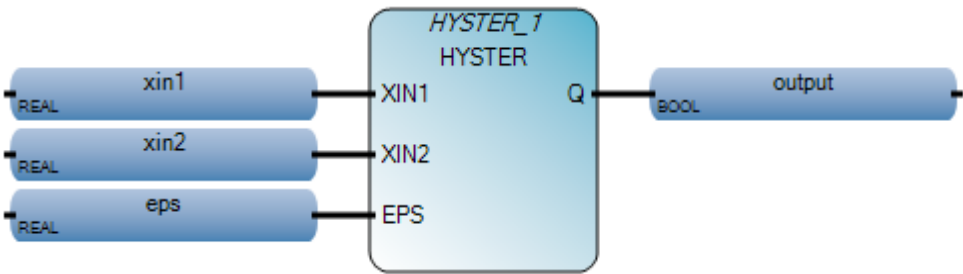
参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行函数。 当 EN = FALSE 时，不执行函数。 仅适用于 LD 编程。
XIN1	输入	REAL	任何实型值。
XIN2	输入	REAL	测试 XIN1 是否已超过 XIN2 + EPS。
EPS	输入	REAL	滞后值（必须大于 0）。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。
Q	输出	BOOL	如果 XIN1 已超过 XIN2 + EPS 并且尚未低于 XIN2 - EPS，则为 TRUE。

Hyster 时间图示例

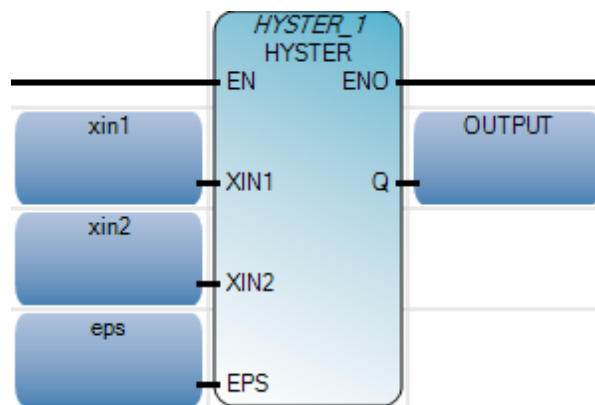


HYSTER 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1 | xin1 := 10.0;
2 | xin2 := 1.0;
3 | eps := 1.0;
4 | HYSTER_1(xin1, xin2, eps);
5 | output := HYSTER_1.Q;

```

HYSTER_1{

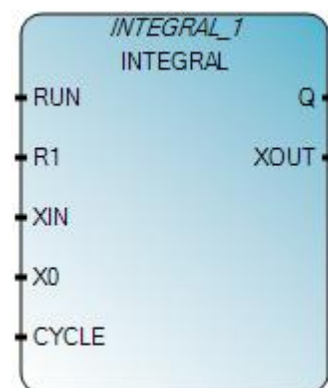
void **HYSTER_1**(REAL XIN1, REAL XIN2, REAL EPS)
Type : HYSTER, Boolean hysteresis on difference of reals

另请参见

[过程控制指令](#) 参考页数 [549](#)

INTEGRAL

INTEGRAL 计算实型值的积分。



INTEGRAL 操作

- 首次初始化 INTEGRAL 功能块时，不会考虑其初始值。使用 R1 参数来设置要用于计算的初始值。
- 为防止丢失积分值，控制器从 PROGRAM 转换为 RUN 或 RUN 参数从 FALSE 转换为 TRUE 时，不会自动清除积分值。首次将控制器从 PROGRAM 转换到 RUN 模式以及启动新的积分时，使用 R1 参数可清除积分值。
- 建议您不要将 EN 或 ENO 参数与此功能块搭配使用，因为当 EN 为 FALSE 时将会中断循环时间计算，从而得到错误积分。如果选择使用 EN 或 ENO 参数，请在 EN 等于 TRUE 的情况下切换 R1 参数，以清除当前结果并确保积分正确。
- 积分是以毫秒为时间基准计算的（也就是说，对初始值为 0 的输入 1 在 1 秒钟时间内求取积分会得到值 1000）。要将该指令的输出换算成以秒为单位表示的值，可将该输出除以 1000。
- 如果 CYCLE 参数值小于设备的执行周期时间，则将采样周期强制为该周期时间。
- XIN 采样和功能块的执行周期为循环时间 + 扫描时间抖动。
- 对于给定用户程序，扫描时间抖动会随控制器不同而发生变化。
- 循环时间决定 Integral 功能块的敏感性。计算积分 XOUT 值时，不会将 XIN 中两次采样之间（或循环时间之内）的更改计算在内。
- 循环时间和扫描时间抖动共同导致 Integral 输出的全部误差。
- 另请参见 XIN 与功能块执行同步的示例以及 XIN 与功能块执行不同步的示例。

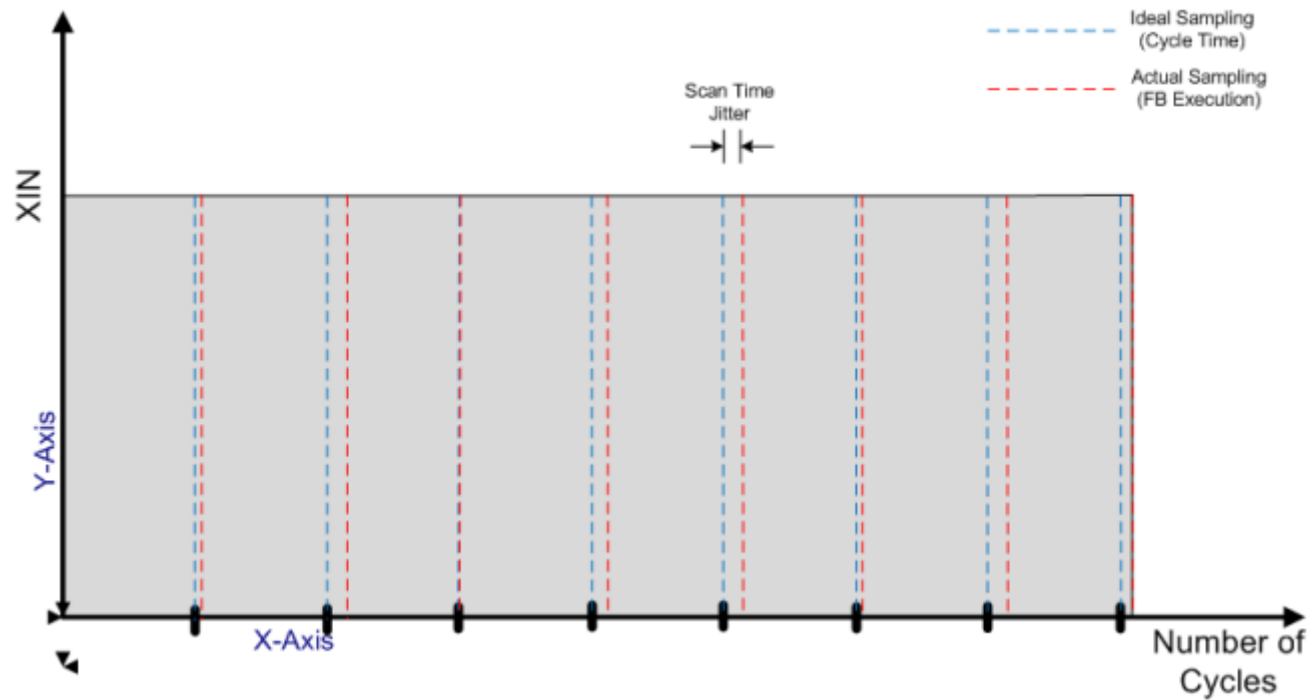
参数

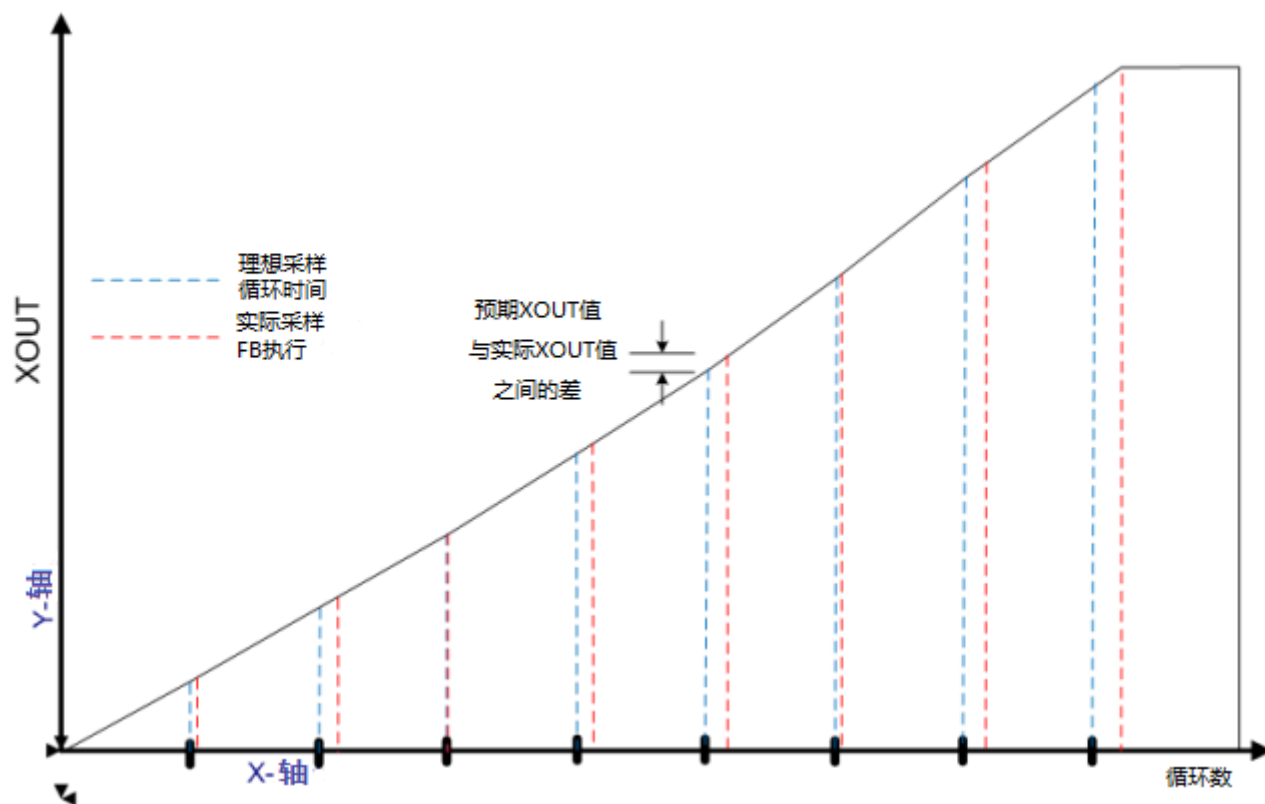
参数	参数类型	数据类型	说明
RUN	输入	BOOL	模式：TRUE = 积分/FALSE = 保留。
R1	输入	BOOL	取消重置。

参数	参数类型	数据类型	说明
XIN	输入	REAL	输入：任何实型值。
X0	输入	REAL	初始值。
CYCLE	输入	TIME	采样周期。可以采用的值的范围为 0ms 到 23h59m59s999ms。
Q	输出	BOOL	非 R1。
XOUT	输出	REAL	积分输出。

示例：XIN 与功能块执行同步

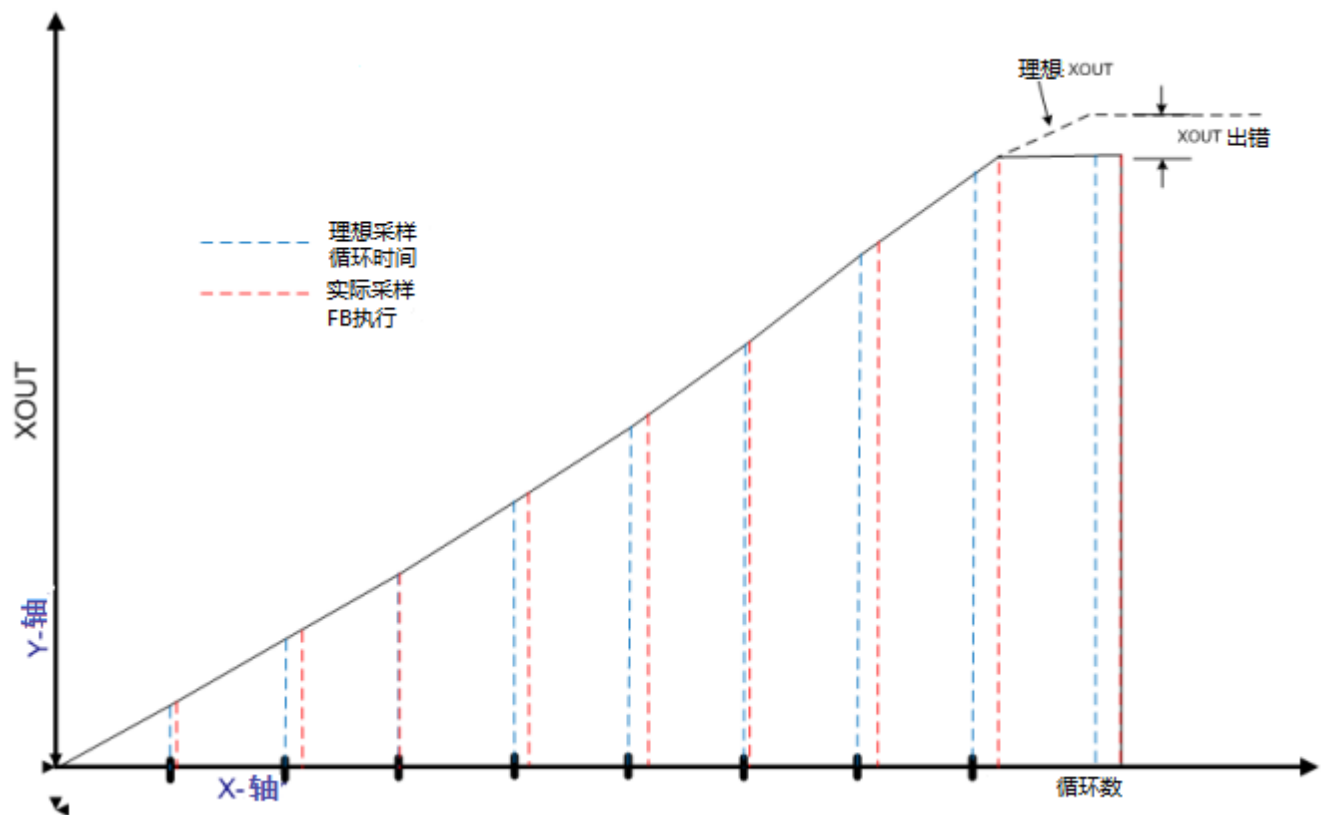
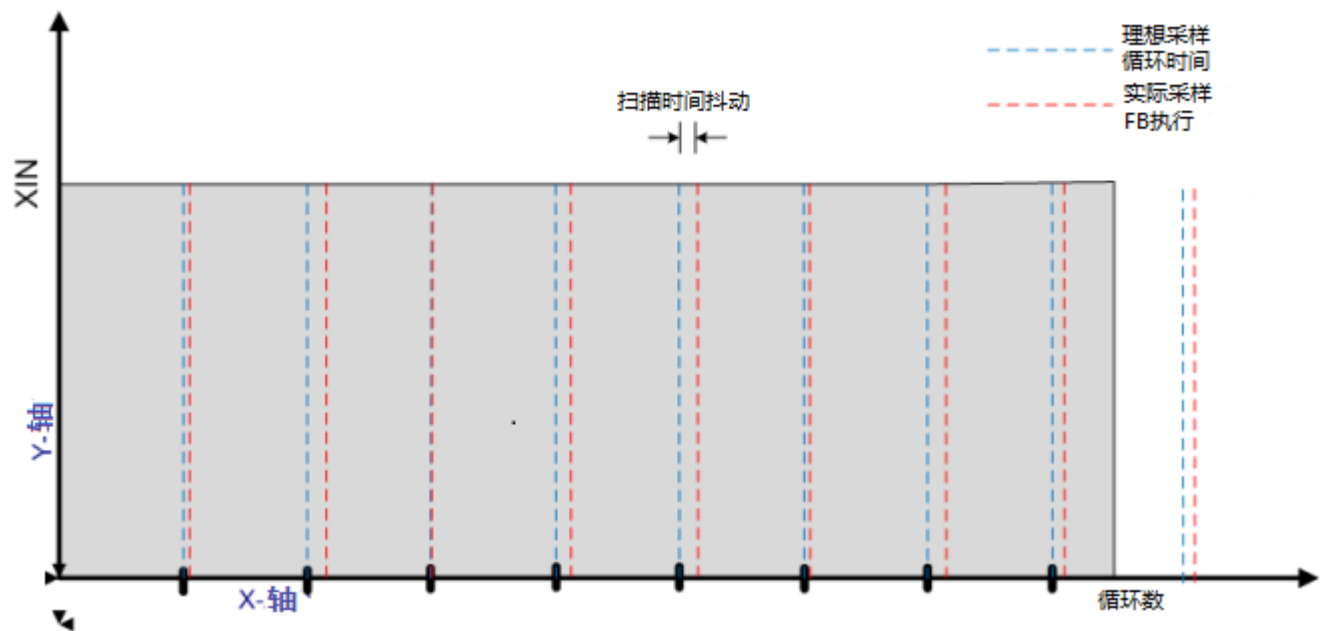
以下图片显示扫描时间抖动对 XOUT 值的影响：





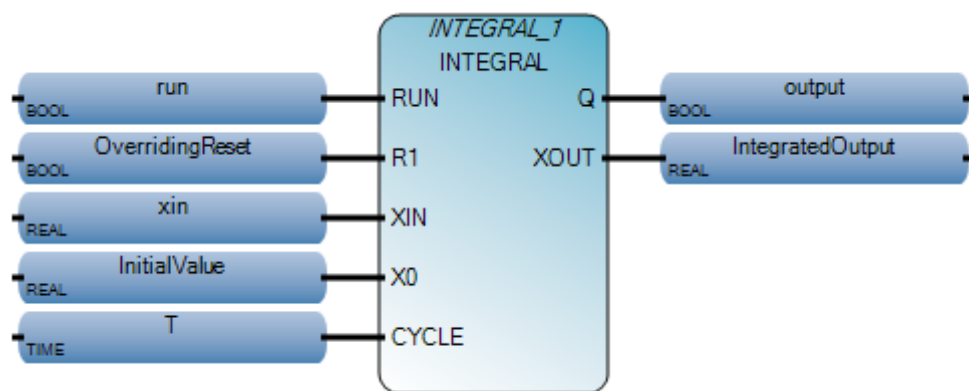
示例：XIN 与功能块执行不同步

以下图片显示将错误引入 Integral 功能块的 XOUT 值中的示例：

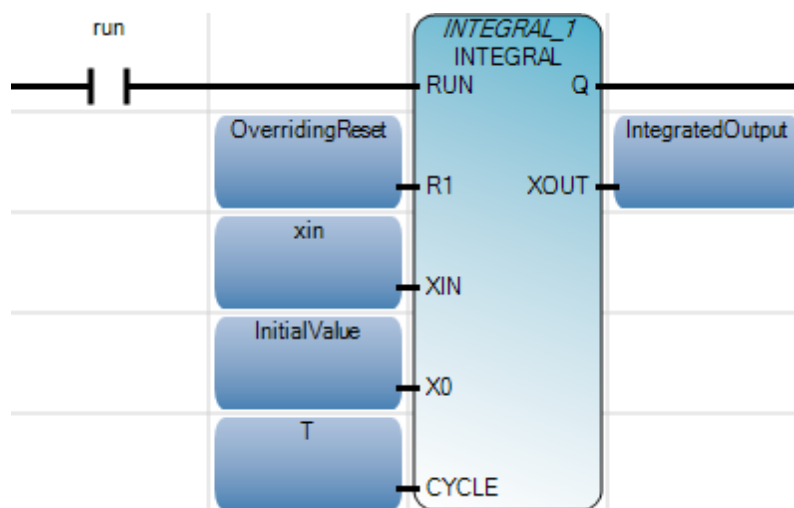


INTEGRAL 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1  run := TRUE;
2  xin := 10.0;
3  InitialValue := 5.0;
4  T := T#10s;
5  INTEGRAL_1(run, OverridingReset, xin, InitialValue, T);
6  output := NOT OverridingReset;
7  IntegratedOutput := INTEGRAL_1.XOUT;

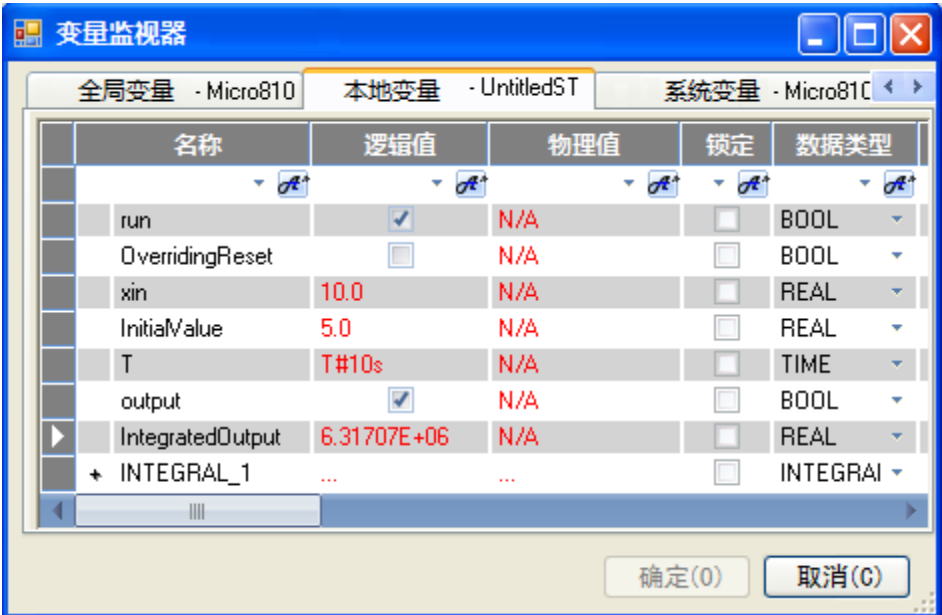
```

```
INTEGRAL_1 {  
void INTEGRAL_1(BOOL RUN, BOOL R1, REAL XIN, REAL X0, TIME CYCLE)  
Type : INTEGRAL, Integration over time
```

(* 与之等效的 ST : INTEGRAL1 是 INTEGRAL 块的实例 *)

```
INTEGRAL1(manual_mode, NOT(manual_mode), sensor_value,  
init_value, t#100ms);  
controlled_value := INTEGRAL1.XOUT;
```

结果

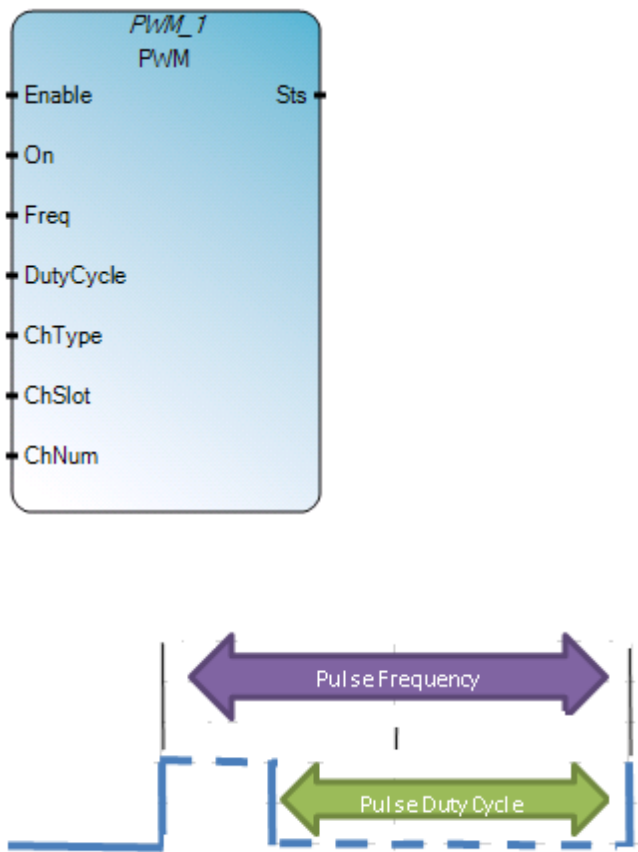


另请参见

[过程控制指令](#) 参考页数 [549](#)

PWM

PWM (脉宽调制) 针对配置的 PWM 通道打开或关闭 PWM 输出。该指令块与 Micro820 2080-LC20-20QBB 控制器配合使用，支持 PWM 单通道 (使用嵌入式输出通道 6)。



PWM 操作

- 只有 Micro820 控制器支持 PWM。

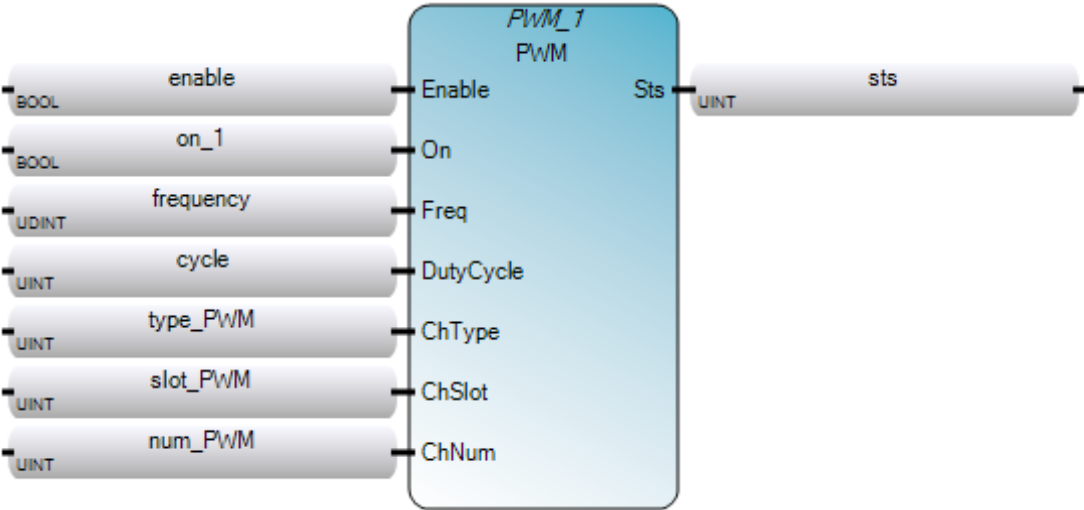
PWM 参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行当前计算。 当 EN = FALSE 时，不执行计算。

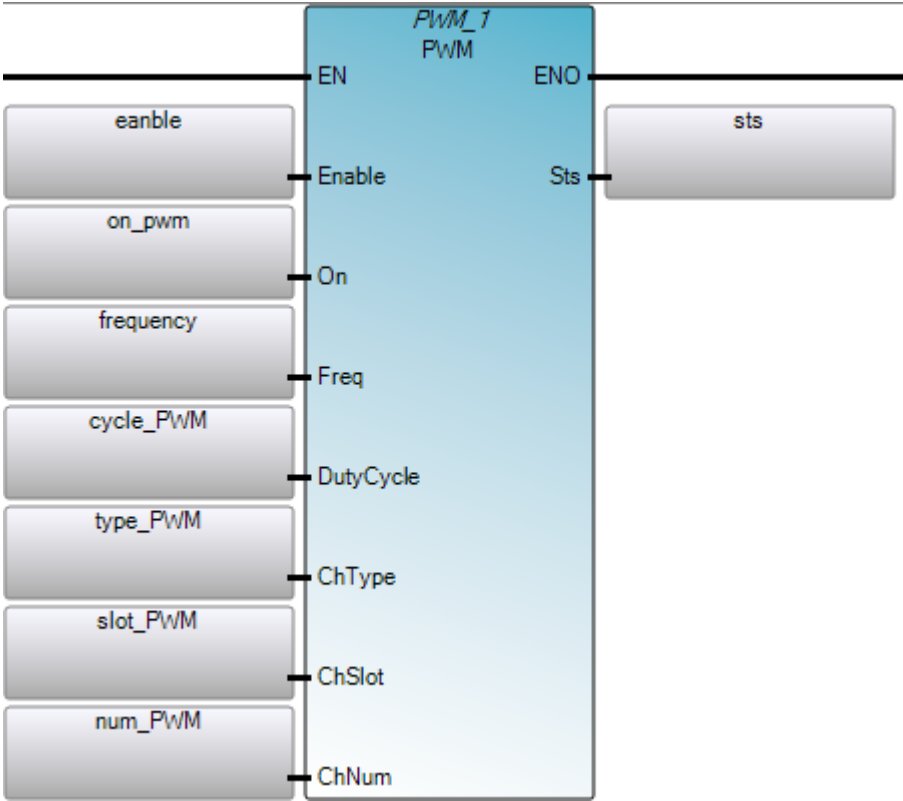
参数	参数类型	数据类型	说明
打开	输入	BOOL	打开或关闭 PWM 输出。 当为 TRUE 时，打开 PWM 输出。如果已打开 PWM 通道，则使用新配置进行覆盖。 当为 FALSE 时，关闭 PWM 输出。
频率	输入	USINT	频率 (Hz) • 1 - 5000
DutyCycle	输入	字符串	工作周期 • 0 - 1000 (0% - 100%)
ChType	输入	USINT	通道类型 • 0 - 嵌入式 • 1 - 插件 • 2 - 扩展
ChSlot	输入	UINT	通道插槽 • 0 - 嵌入式
ChNum	输入	UINT	通道编号 • 0 - PWM CH0 // 对于 2080-LC20-20QBB，将嵌入式输出引脚 6 用作 PWM CH0。
Sts	输出	UINT	功能块执行状态。 请参见 PWM 状态代码 参考页数 563 。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

PWM 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 PWM (EN, Enable, On, Freq, DutyCycle, ChType, ChSlot, ChNum);
2 output := PWM.ENO
3 sts := PWM.Sts
```

```
PWM_1 (
void PWM_1(BOOL Enable, BOOL On, UDINT Freq, UINT DutyCycle, UINT ChType, UINT ChSlot, UINT ChNum)
Type: PWM, Enable PWM output.
```

另请参见

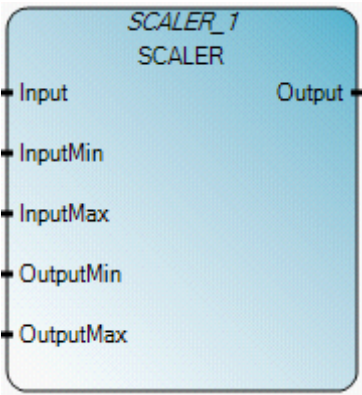
[过程控制指令](#) 参考页数 [549](#)

PWM 状态代码

状态代码	说明
0	未启用功能块（无操作）。
1	PWM 配置成功。
2	工作周期无效。
3	频率无效。
4	通道类型无效。
5	通道插槽无效。
6	通道编号无效。
7	目录无效。所使用的目录不支持 PWM 功能。

SCALER

SCALER 根据输出范围调整输入值。

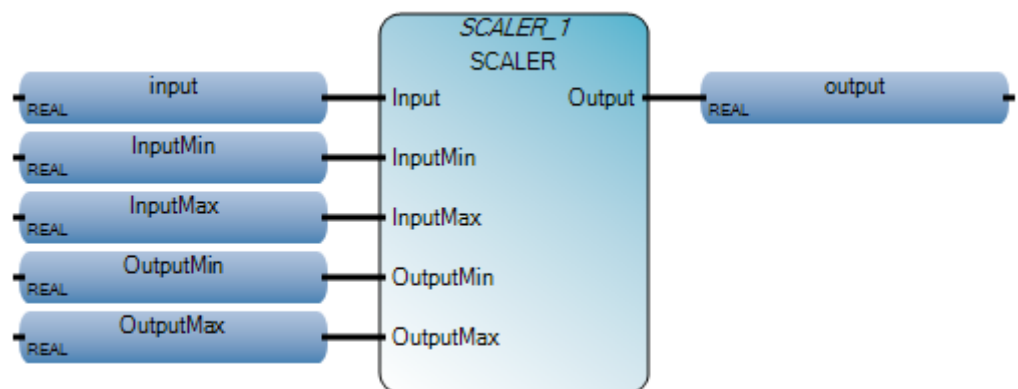


参数

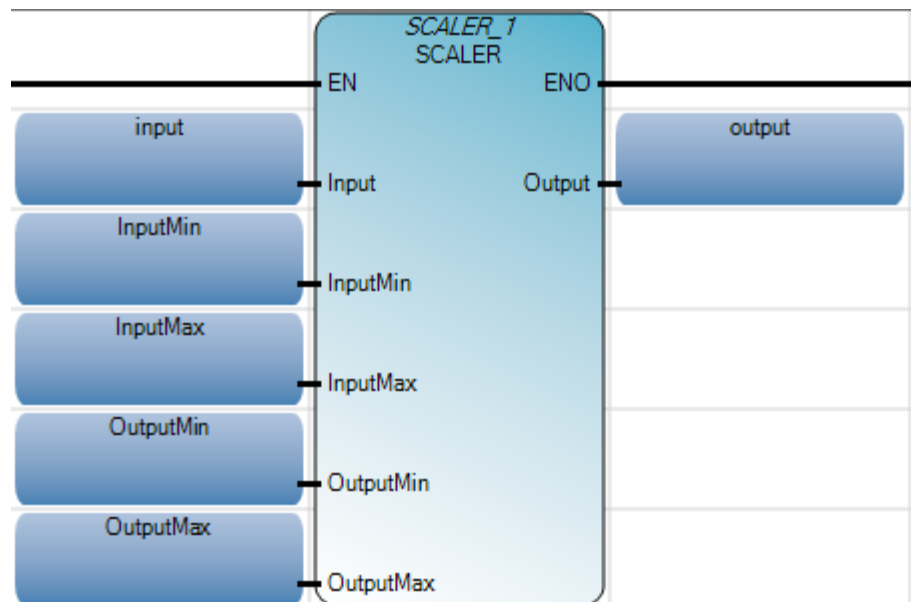
参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行调整方程。 当 EN = FALSE 时，没有调整方程。 仅适用于 LD 编程。
输入	输入	REAL	输入信号。
InputMin	输入	REAL	输入的最小值。
InputMax	输入	REAL	输入的最大值。
OutputMin	输入	REAL	输出的最小值。
OutputMax	输入	REAL	输出的最大值。
输出	输出	REAL	输出值。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

SCALER 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 input := 10.0;
2 InputMin := 5.0;
3 InputMax := 15.0;
4 OutputMin := 1.0;
5 OutputMax := 10.0;
6 SCALER_1(input, InputMin, InputMax, OutputMin, OutputMax);
7 output := SCALER_1.Output;
```

SCALER_1(

void **SCALER_1**(REAL Input, REAL InputMin, REAL InputMax, REAL OutputMin, REAL OutputMax)
Type : SCALER, Scale input value according to output range.

(* 与之等效的 ST : SCALER1 是 SCALER 块的实例 *)

```
SCALER1(Signal_In, 4.0, 20.0 , 0.0 , 150.0) ;
Out_Temp := SCALER1.Output ;
```

结果

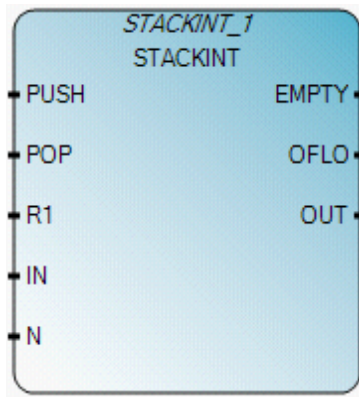


另请参见

[过程控制指令](#) 参考页数 [549](#)

STACKINT

STACKINT 管理整数值堆栈。



STACKINT 操作

STACKINT 功能块包括 PUSH 和 POP 命令的上升沿检测。堆栈的最大大小为 128。OFLO 值仅在重置之后有效（R1 至少有一次设置为 TRUE，然后再设置回 FALSE）。应用程序定义的堆栈大小 N 不能小于 1 或大于 128。STACKINT 按以下方式管理无效值：

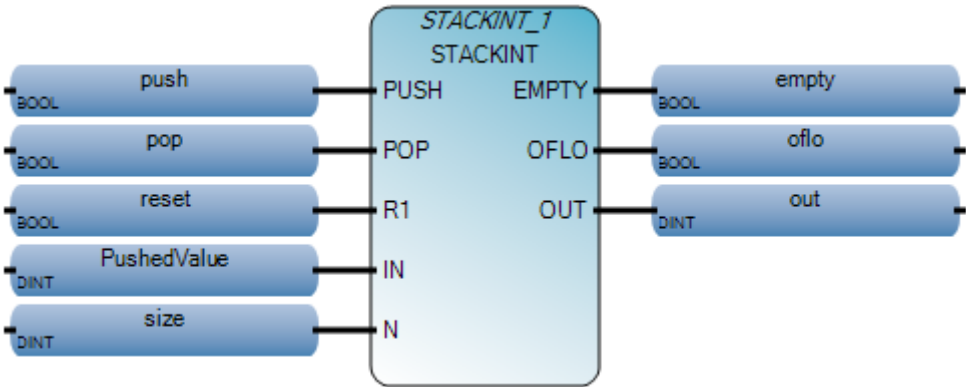
- 如果 $N < 1$ ，STACKINT 假设大小为 1。
- 如果 $N > 128$ ，STACKINT 假设大小为 128。

参数

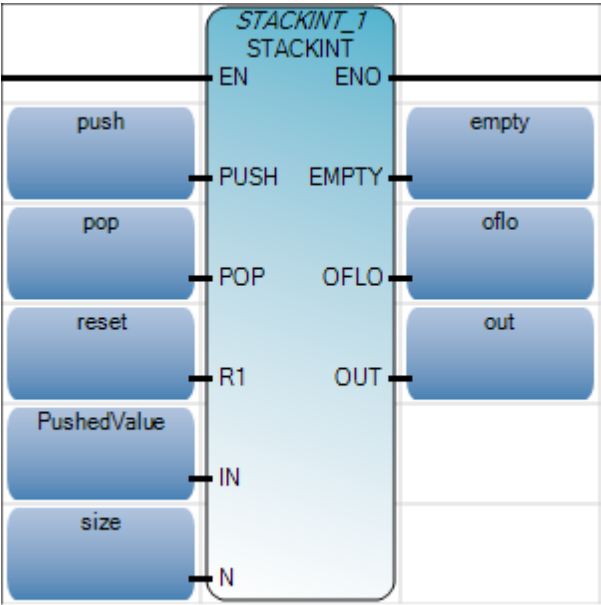
参数	参数类型	数据类型	说明
PUSH	输入	BOOL	Push 命令（仅限上升沿）。在堆栈顶部添加 IN 值。
POP	输入	BOOL	Pop 命令（仅限上升沿）。删除推入堆栈顶部的最后一个值。
R1	输入	BOOL	重置堆栈至其空状态。
IN	输入	DINT	推送的值。
N	输入	DINT	应用程序定义的堆栈大小。
EMPTY	输出	BOOL	如果堆栈为空则为 TRUE。
OFLO	输出	BOOL	溢出：堆栈满了之后即为 TRUE。
OUT	输出	DINT	堆栈顶部的值。 当 OFLO 为 TRUE 时，OUT 等于 0。

STACKINT 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1 PushedValue := 5;
2 size := 10;
3 STACKINT_1(push, pop, reset, PushedValue, size);
4 empty := STACKINT_1.EMPTY;
5 oflo := STACKINT_1.OFLO;
6 out := STACKINT_1.OUT;

```

STACKINT_1{

void **STACKINT_1**(BOOL PUSH, BOOL POP, BOOL R1, DINT IN, DINT N)
Type : STACKINT, Stack of integer analogs

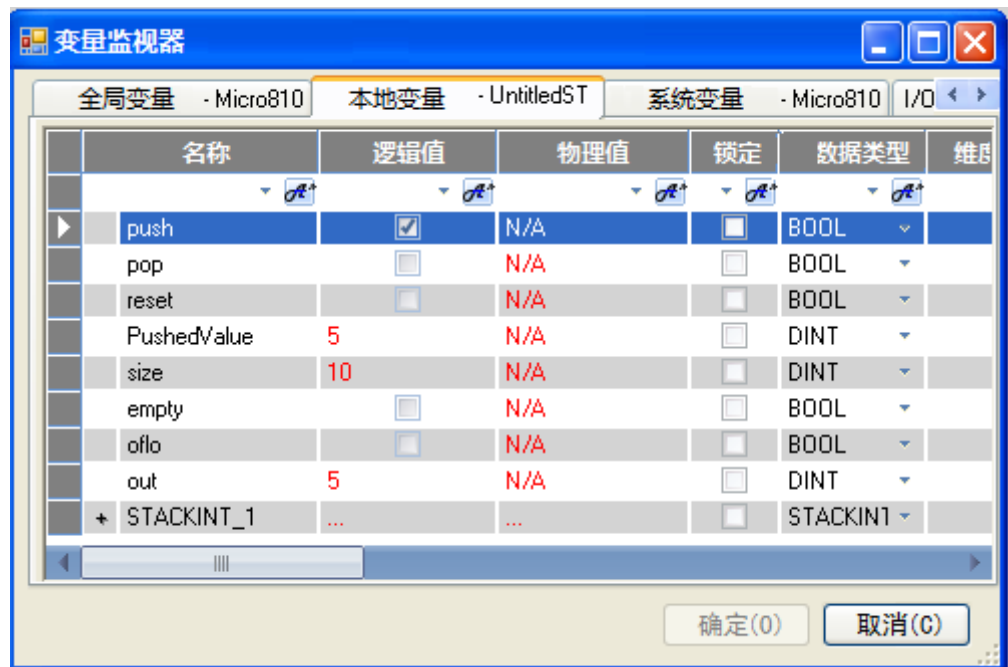
(* 与之等效的 ST : STACKINT1 是 STACKINT 块的实例 *)

```

STACKINT1(err_detect, acknowledge, manual_mode, err_code,
max_err);
appli_alarm := auto_mode AND NOT(STACKINT1.EMPTY);
err_alarm := STACKINT1.OFLO;
last_error := STACKINT1.OUT;

```

结果



另请参见

[过程控制指令](#) 参考页数 [549](#)

TND

TND 停止用户程序扫描的当前循环。然后，在输出扫描、输入扫描和清理后，将从第一个例程的开始处重新执行用户程序。

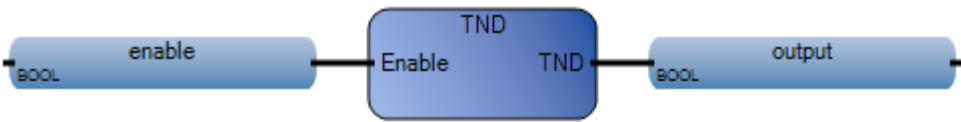


参数

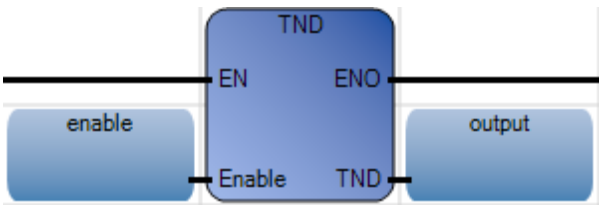
参数	参数类型	数据类型	说明
启用	输入	BOOL	函数启用。 当 Enable = TRUE 时，则执行该函数。 当 Enable = FALSE 时，则不执行该函数。
TND	输出	BOOL	如果为 TRUE，则执行该函数。 提示：如果启用了变量监视，则监视变量的值将分配给块的输出。 如果关闭了变量监视，则输出变量的值将分配给块的输出。

TND 函数语言示例

功能块图



梯形图



结构化文本

```
1 enable := TRUE;  
2 output := TND(enable);
```

TND (

BOOL TND(BOOL Enable)
Abort current user program scan.

(* 与之等效的 ST : *)

TESTOUTPUT := TND(TESTENABLE);

结果



另请参见

[过程控制指令](#) 参考页数 [549](#)

LIMIT

LIMIT 将整型值限制为给定的间隔。不能更改最小值和最大值之间的整型值。大于最大值的整型值将替换为最大值。小于最小值的整型值将替换为最小值。



参数

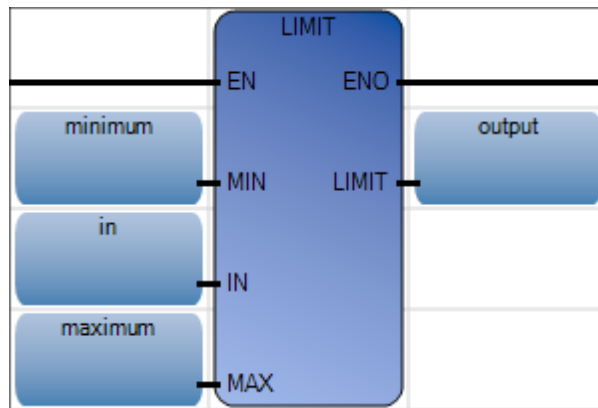
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前 LIMIT 计算。 当 EN = FALSE 时，不执行计算。
MIN	输入	DINT	支持最小值。
IN	输入	DINT	任何有符号整型值。
MAX	输入	DINT	支持最大值。
LIMIT	输出	DINT	位于支持的范围内的输入值。
ENO	输出	BOOL	启用输出。

LIMIT 函数语言示例

功能块图



梯形图



结构化文本

```
1 minimum := 2;  
2 in := 5;  
3 maximum := 10;  
4 output := LIMIT(minimum, in, maximum);
```

```
LIMIT(  
DINT LIMIT(DINT MIN, DINT IN, DINT MAX)  
Limit
```

(* 与之等效的 ST : *)

new_value := LIMIT (min_value, value, max_value);

(* 将值限制在 [min_value..max_value] 集合内 *)

结果



另请参见

[过程控制指令](#) 参考页数 549

程序控制指令

程序控制指令用于同时从用户程序和操作员接口设备中控制指令。

功能块	说明
SUS 参考页数 577	挂起应用程序的执行。

SUS

SUS 将挂起 Micro800 控制器的执行。控制器将一直处于运行模式，但是执行将无限期暂停。“挂起”将会捕捉用户程序错误并帮助用户程序监视。将 SUS 指令放置在您想要捕捉异常情况的用户程序部分。在暂停模式下，RUN LED 将设置为“关闭”以指示程序扫描处于闲置状态。

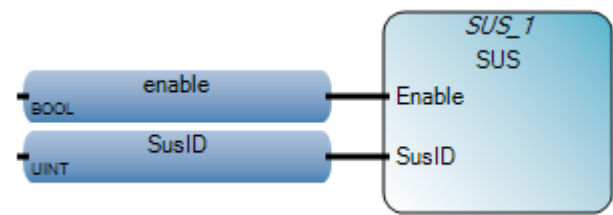


参数

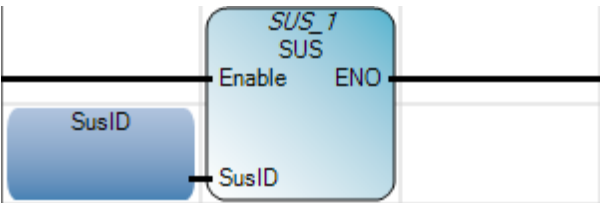
参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行函数。 当 Enable = FALSE 时，不执行函数。
SusID	输入	UINT	挂起 ID。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

SUS 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 SusID := 1;
2 SUS_1(enable, SusID);
```

SUS_1(

void **SUS_1**(BOOL Enable, UINT SusID)
Type : SUS, Suspend the execution of the application.

结果



另请参见

[程序控制指令](#) 参考页数 [577](#)

比例积分微分 (PID) 指令

比例积分微分 (PID) 指令用于使用 PID 功能来更为准确地控制过程。

功能块	说明
IPIDCONTROLLER 参考页数 583	比例积分导数

什么是比例积分微分 (PID) 控制？

比例积分微分 (PID) 控制使过程控制能够通过调整控制输出来准确保持设定值。PID 功能块将所有必需的逻辑合并以执行比例/积分/微分 (PID) 控制。

另请参见

[比例积分微分 \(PID\) 指令](#) 参考页数 [581](#)

[IPIDController 功能块操作](#) 参考页数 [589](#)

[使用比例积分微分指令](#) 参考页数 [590](#)

IPIDController 功能块如何实施 PID 控制

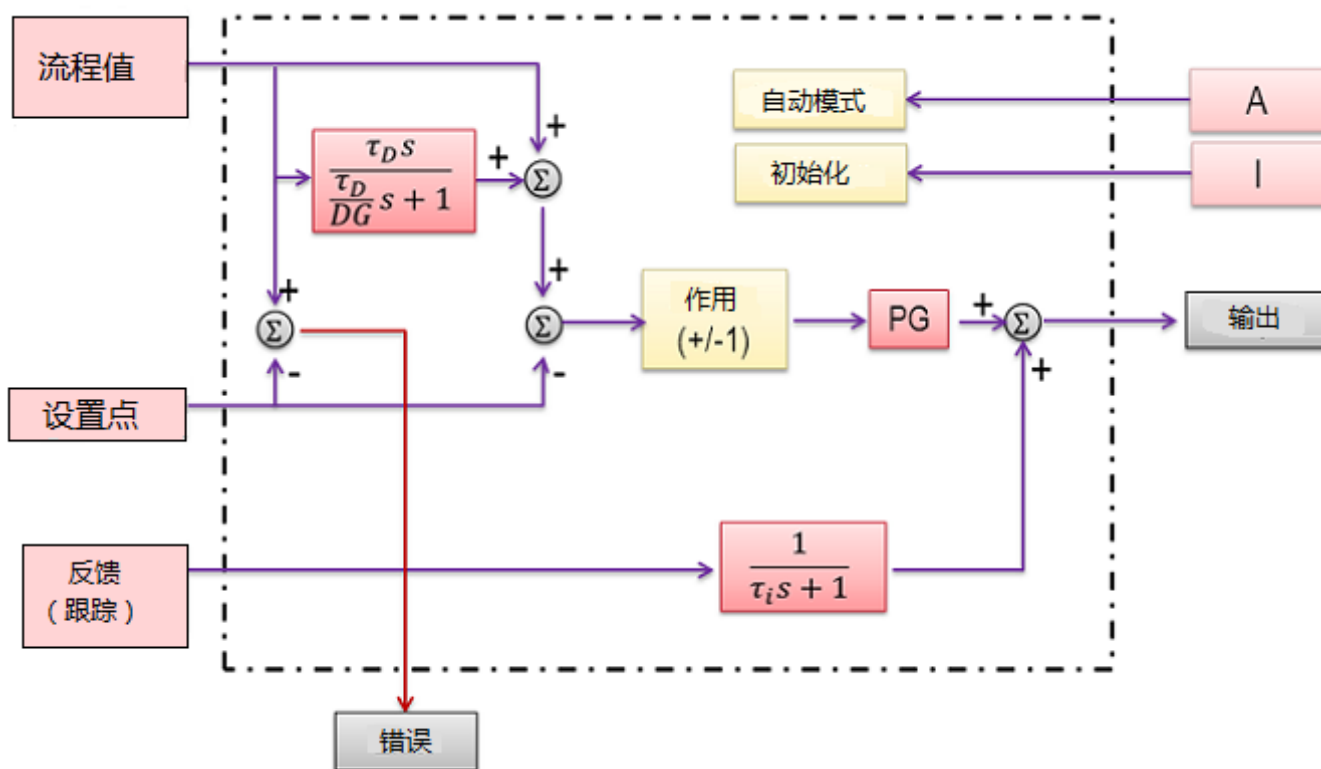
Connected Components Workbench 指令集中提供的 IPIDController 功能块基于 PID 控制理论，它将所有必需的逻辑合并以执行模拟输入通道处理以及比例积分微分 (PID) 控制。在 HMI 中，IPID 面板可用于 IPIDController 功能块。

IPIDController 功能块说明

IPIDController 功能块使用以下功能块组件：

- A：作用 (+/- 1)
- PG：比例增益

- DG : 导数增益
- td : $\hat{a}D$
- ti : $\hat{a}I$



防止积分饱卷

如果设定点值与过程值之间的差异较大，则输出值会大幅攀升，而在其降低时，过程将失控。IPIDController 功能块以交互方式跟踪反馈，并防止积分饱卷。当输出饱和时，会重新计算控制器的积分项，其新值会在达到饱和限制时提供输出。

另请参见

[比例积分微分 \(PID\) 指令](#) 参考页数 581

[IPIDController 功能块操作](#) 参考页数 589

[使用比例积分微分指令](#) 参考页数 [590](#)

IPIDCONTROLLER

IPIDCONTROLLER 用于比例积分微分 (PID) 逻辑，该逻辑使用过程循环来控制诸如温度、压力、液面或流速等物理属性。



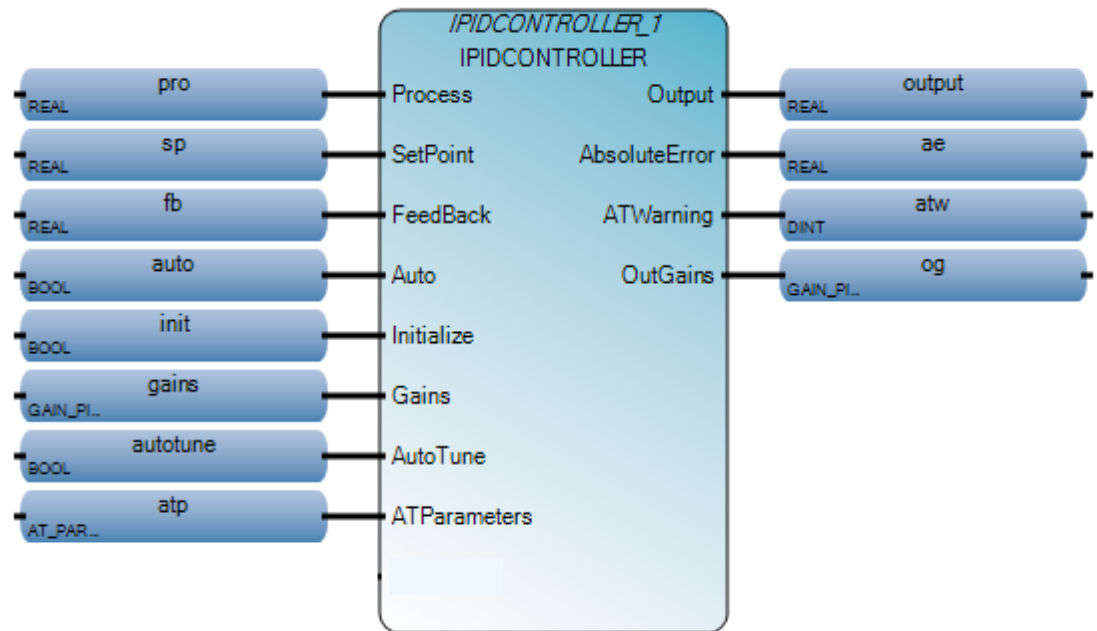
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	功能块启用。 当 EN = TRUE 时，执行函数。 当 EN = FALSE 时，不执行函数。 仅适用于 LD 编程。
Process	输入	REAL	流程值，是根据流程输出测量到的值。
SetPoint	输入	REAL	设置点。
反馈	输入	REAL	反馈信号，是应用于流程的控制变量的值。 例如，反馈可以为 IPIDCONTROLLER 输出。
自动	输入	BOOL	PID 控制器的操作模式： • TRUE - 控制器以正常模式运行。 • FALSE - 控制器导致将 R 重置为跟踪 (F-GE)。
Initialize	输入	BOOL	值的更改 (TRUE 更改为 FALSE 或 FALSE 更改为 TRUE) 导致在相应循环期间控制器消除任何比例增益。同时还会初始化 AutoTune 序列。

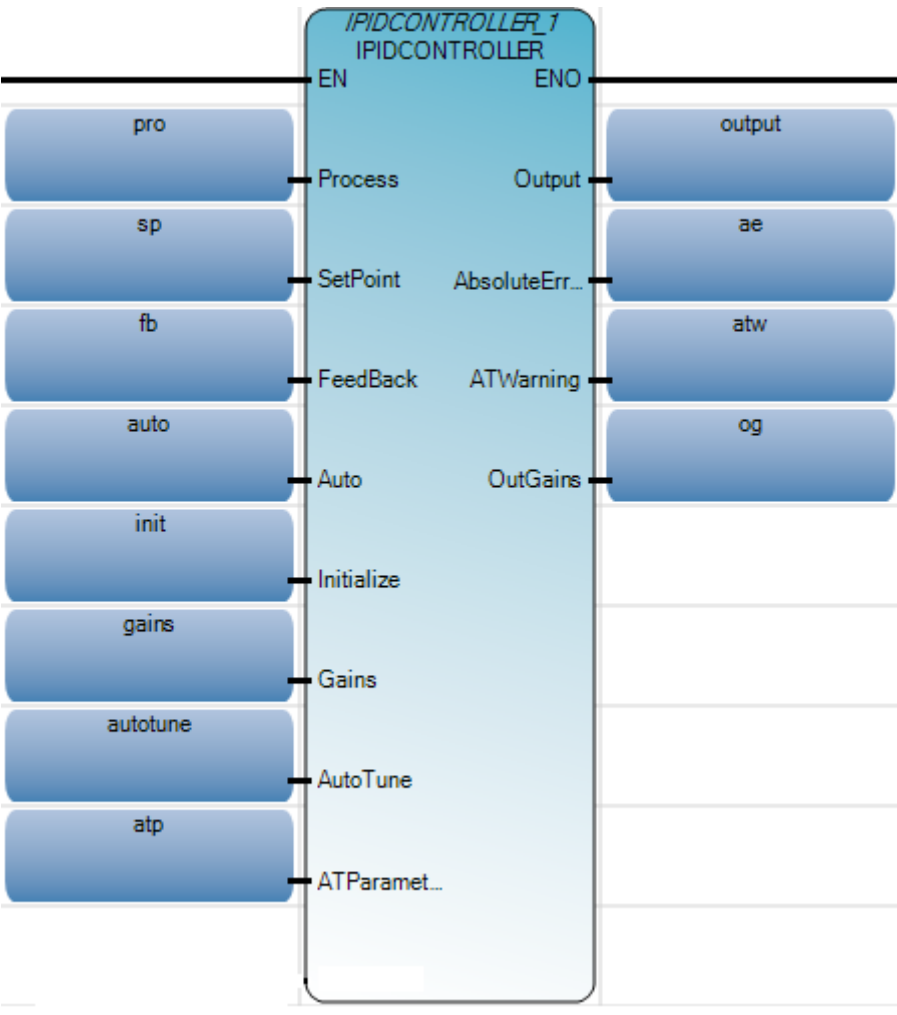
参数	参数类型	数据类型	说明
Gains	输入	GAIN_PID	IPIDController 的增益 PID。 请参见 GAIN_PID 数据类型 参考页数 588 。
AutoTune	输入	BOOL	当设置为 TRUE 且 Auto 和 Initialize 为 FALSE 时，会启动 AutoTune 序列。
ATParameters	输入	AT_Param	自动调节参数。 请参见 AT_Param 数据类型 参考页数 588 。
输出	输出	REAL	来自控制器的输出值。
AbsoluteError	输出	REAL	来自控制器的绝对错误 (Process - SetPoint)。
ATWarnings	输出	DINT	(ATWarning) 自动调节序列的警告。可能的值有： • 0 - 没有执行自动调节。 • 1 - 处于自动调节模式。 • 2 - 已执行自动调节。 • -1 - ERROR 1 输入自动设置为 TRUE，不可能进行自动调节。 • -2 - ERROR 2 自动调节错误，ATDynaSet 已过期。
OutGains	输出	GAIN_PID	在 AutoTune 序列之后计算的增益。 请参见 GAIN_PID 数据类型 参考页数 588 。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

IPIDCONTROLLER 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 IPIDCONTROLLER_1(pro, sp, fb, auto, init, gains, autotune, atp, em);
2 output := IPIDCONTROLLER_1.Output;
3 ae := IPIDCONTROLLER_1.AbsoluteError;
4 atw := IPIDCONTROLLER_1.ATWarning;
5 og := IPIDCONTROLLER_1.OutGains;
```

IPIDCONTROLLER_1

void IPIDCONTROLLER_1(REAL Process, REAL SetPoint, REAL FeedBack, BOOL Auto, BOOL Initialize, GAIN_PID Gains, BOOL AutoTune, AT_PARAM ATParameters, DINT ErrorMode)

Type : IPIDCONTROLLER, Proportional Integral Derivative.

(* 与之等效的 ST: IPIDController1 是 IPIDController 块的实例 *)

```
IPIDController1(Proc,
                SP,
                FBK,
                Auto,
                Init,
                G_In,
                A_Tune,
                A_TunePar,
                Err );
Out_process := IPIDController1.Output ;
A_Tune_Warn := IPIDController1.ATWarning ;
Gain_Out := IPIDController1.OutGains ;
```

结果

变量监视器

全局变量 · Micro810 本地变量 · UntitledST 系统变量 · Micro81

名称	逻辑值	物理值	锁定	数据类型
HighLimit	10.0	N/A	☐	REAL
X	15.0	N/A	☐	REAL
LowLimit	5.0	N/A	☐	REAL
HysteresisValue	2.0	N/A	☐	REAL
OutputH	☒	N/A	☐	BOOL
output	☒	N/A	☐	BOOL
OutputL	☐	N/A	☐	BOOL
LIM_ALARM_1	...	...	☐	LIM_ALARM

确定(O) 取消(C)

另请参见

- [GAIN PID 数据类型](#) 参考页数 588
- [AT Param 数据类型](#) 参考页数 588
- [什么是比例积分微分 \(PID\) 控制？](#) 参考页数 581
- [IPIDController 功能块操作](#) 参考页数 589
- [使用比例积分微分指令](#) 参考页数 590

GAIN_PID 数据类型

下表描述了 GAIN_PID 数据类型。

参数	数据类型	说明
DirectActing	BOOL	作用类型： • TRUE - 正向作用（输出与误差沿同一方向移动）。也就是说，实际的过程值要大于设定值，并且适当的控制器操作会增加输出（例如：降温）。 • FALSE - 反向作用（输出与误差沿相反方向移动）。也就是说，实际的过程值要大于设定值，并且适当的控制器操作会降低输出（例如：加热）。
ProportionalGain	REAL	PID 的比例增益 (≥ 0.0001)。
TimeIntegral	REAL	PID 的时间积分值 (≥ 0.0001)。
TimeDerivative	REAL	PID 的时间微分值 (> 0.0)。
DerivativeGain	REAL	PID 的微分增益 (> 0.0)。

另请参见

[IPIDCONTROLLER](#) 参考页数 [583](#)

AT_Param 数据类型

下表描述了 AT_Param 数据类型参数。

参数	数据类型	说明
Load	REAL	自动调节的加载参数。它是启动 AutoTune 时的输出值。
Deviation	REAL	自动调节的偏差。这是用于评估 AutoTune 所需噪声频带的标准偏差。
步骤	REAL	AutoTune 的步长值。必须大于噪声频带并小于 $\frac{1}{2}$ Load。
ATDynamSet	REAL	放弃自动调节之前等待的时间（以秒为单位）。
ATReset	BOOL	指示在 AutoTune 序列之后是否要将输出值重置为零： • TRUE - 将输出重置为零。 • FALSE - 将输出保留为 Load 值。

另请参见

[IPIDCONTROLLER](#) 参考页数 583

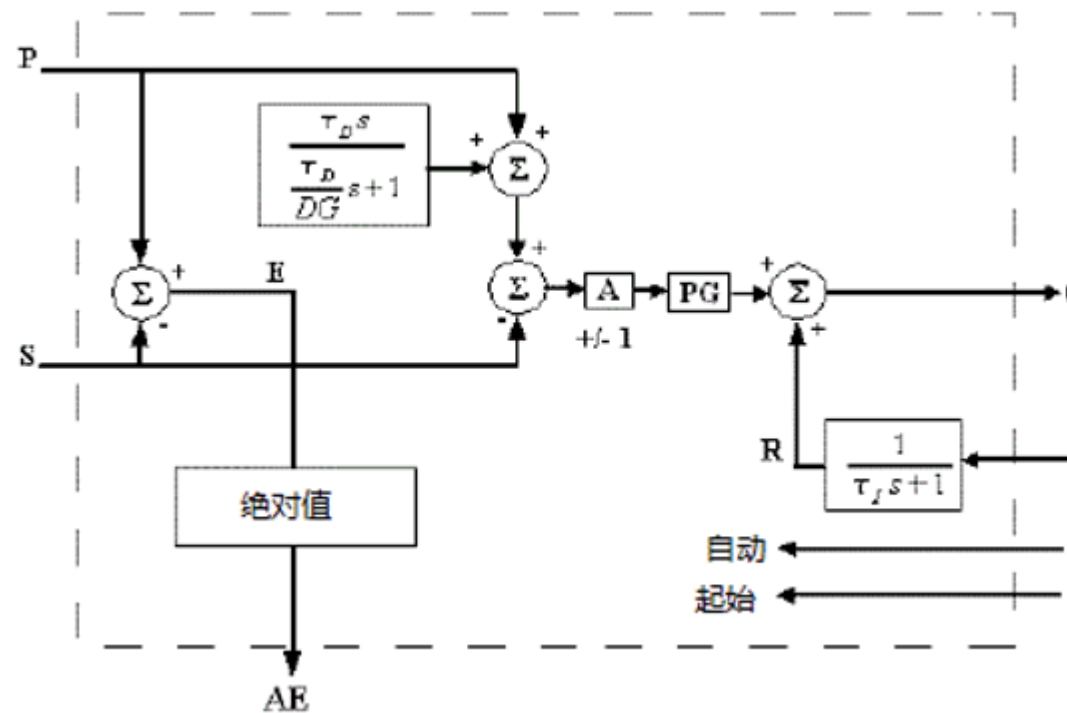
[创建操作值的反馈回路](#) 参考页数 601

IPIDController 功能块操作

Input Auto

当 **Input Auto** 为 **TRUE** 时, IPIDController 以正常的自动模式运行。

当 **Input Auto** 为 **FALSE** 时, 它将使得 R 被重置为跟踪 (F-GE), 强制 IPIDController 输出跟踪 IPIDController 限定范围内的反馈, 此时控制器会切换回自动, 而不会增加输出。



输入初始化

对于输入初始化, 当 AutoTune 为 **FALSE** 时, 从 **FALSE** 更改为 **TRUE** 或从 **TRUE** 更改为 **FALSE**, 会导致在相应循环期间 IPIDController 消除任何比例增益操作(例如初始化)。当使用切换功能块对 SetPoint 进行更改时, 您可以使用此进程来防止增加输出。

运行 AutoTune 序列

要运行 AutoTune 序列，必须完成输入 ATParameters。必须根据进程和所设置的 DerivativeGain，来设置输入增益和 DirectActing 参数（通常为 0.1）。AutoTune 序列从以下序列开始：

1. 将输入初始化设置为 TRUE。
2. 将输入自动调节设置为 TRUE。
3. 将输入初始化更改为 FALSE。
4. 稍候，直到输出 ATWarning 更改为 2。
5. 将输出 OutGains 的值传输至输入增益。

最终确定调节

要完成调节，可性能需要某些微调，具体取决于进程和需求。当将 TimeDerivative 设置为 0.0 时，IPIDController 会强制 DerivativeGain 设置为 1.0，然后作为 PI 控制器进行工作。

另请参见

[比例积分微分 \(PID\) 指令](#) 参考页数 [581](#)

[什么是比例积分微分](#) 参考页数 [581](#)

[IPIDController 功能块如何实施 PID 控制](#) 参考页数 [581](#)

[IPIDController 功能块操作](#) 参考页数 [589](#)

[使用比例积分微分指令](#) 参考页数 [590](#)

使用比例积分微分指令

本节介绍使用比例积分微分指令的特定详细信息和示例，具体包括以下内容：

[示例：如何创建操作值的反馈回路](#) 参考页数 [601](#)

[示例：如何在 IPIDController 功能块中实施自动调节](#) 参考页数 [591](#)

[示例：如何将 UDFB 添加到 PID 程序](#) 参考页数 [603](#)

通过 IPIDController 功能块实现自动调节

可在控制程序中使用 IPIDController 功能块的 AutoTune 参数来实现自动调节。

自动调节要求和建议

下面总结了成功实施自动调节的要求和建议。

- 自动调节必须使控制回路的输出发生振荡,这意味着必须足够频繁地调用 IPIDController,以对振荡进行充分采样。
- IPIDController 指令块必须以相对恒定的时间间隔执行。
- 将程序的扫描时间配置为小于振荡周期的一半。
- 可考虑使用结构化文本中断 (STI) 指令块来控制 IPIDController 指令块。

另请参见

[比例积分微分 \(PID\) 指令](#) 参考页数 [581](#)

[什么是比例积分微分](#) 参考页数 [581](#)

[IPIDController 功能块操作](#) 参考页数 [589](#)

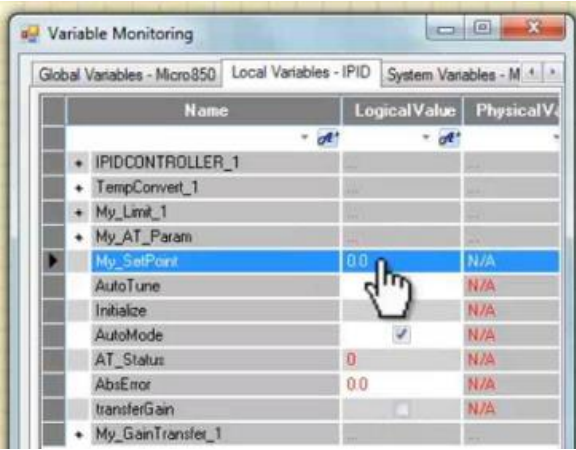
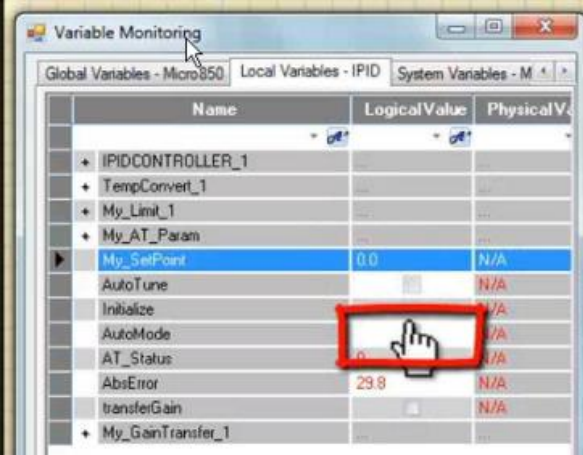
[使用比例积分微分指令](#) 参考页数 [590](#)

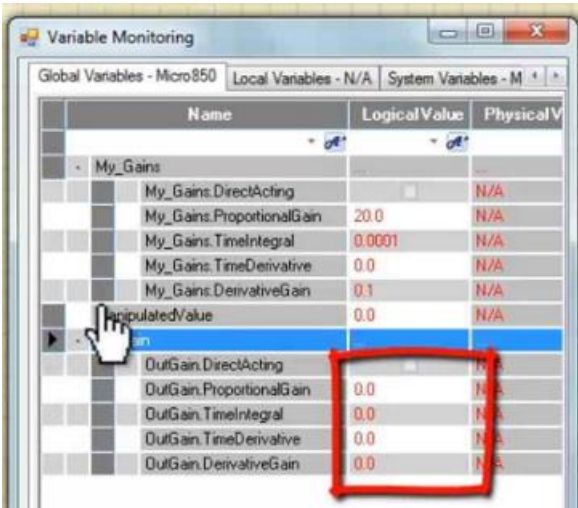
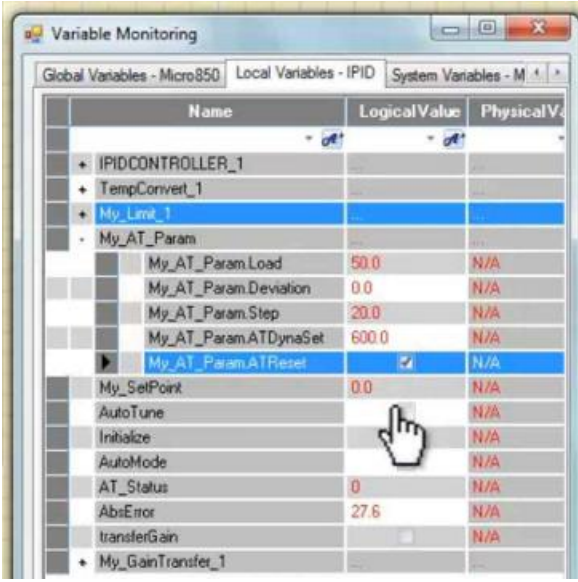
一阶和二阶系统中的自动调节

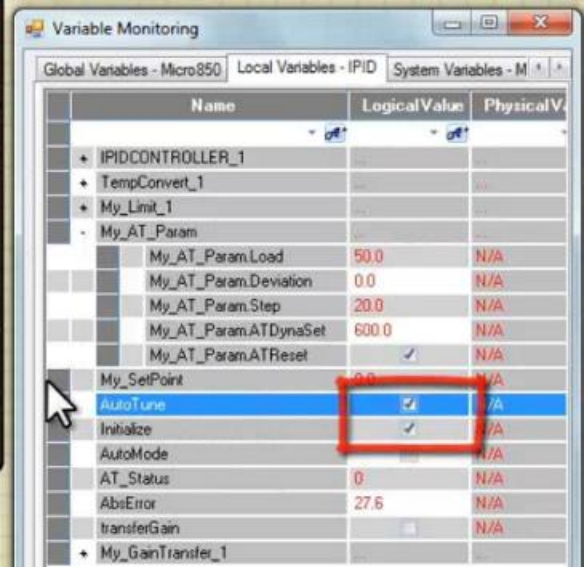
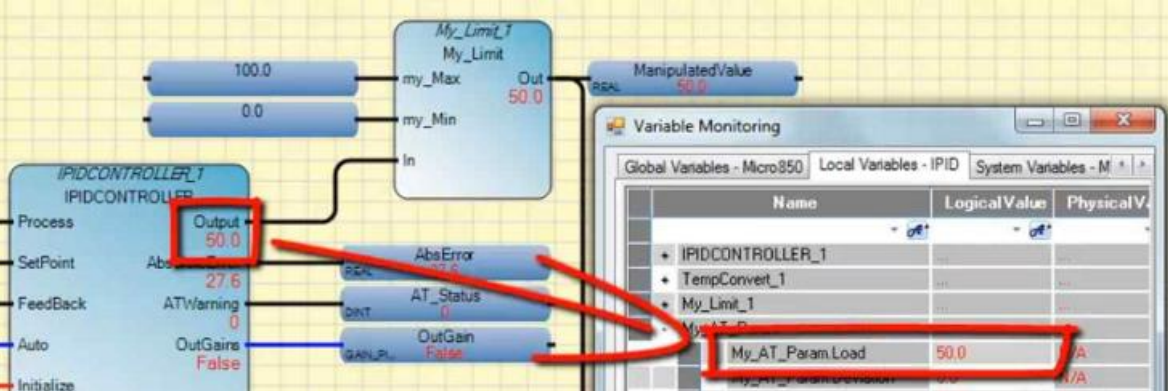
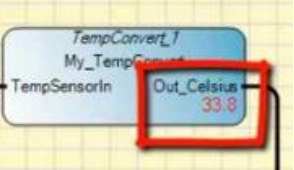
自动调节既可在使用单个元素的一阶系统中使用,又可在使用两个独立元素的二阶系统中使用。

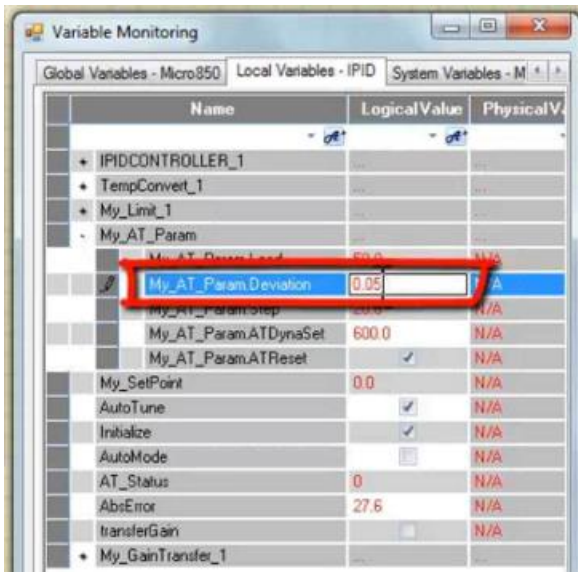
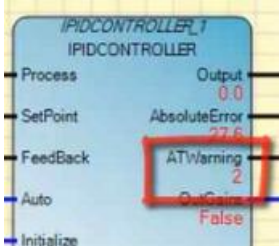
配置自动调节

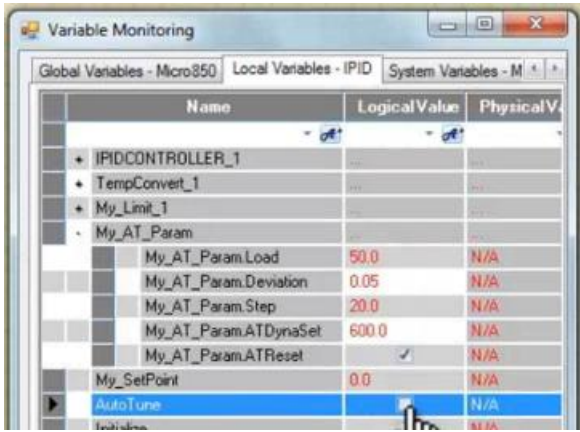
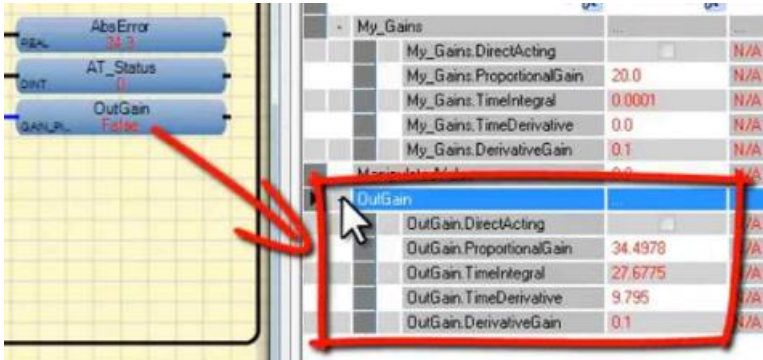
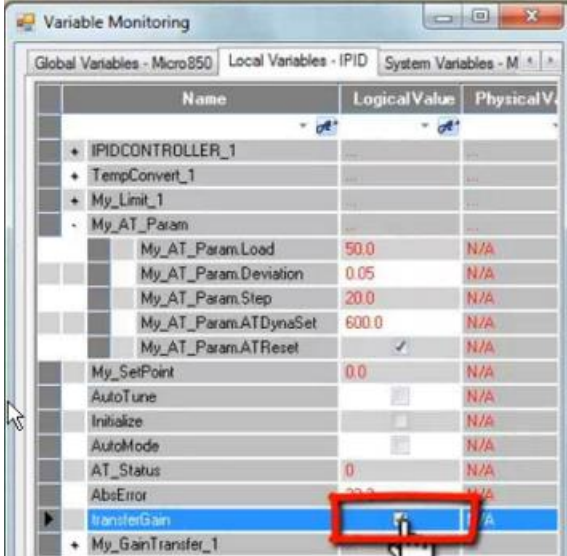
以下是通过使用 IPIDController 函数实现自动调节的一般步骤。

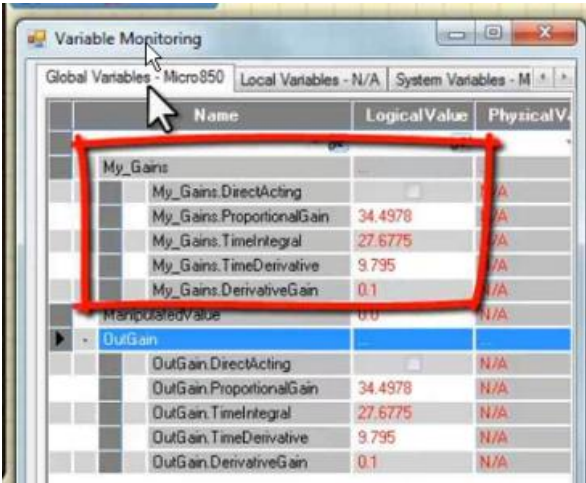
编号	步骤	示例
1	将设定点重置为零。	
2	将 Auto 模式切换为 False	

编号	步骤	示例
3	设置增益参数。	
4	设置自动调节参数。	设置自动调节参数，包括初始加载值、输出的分步变更、完成自动调节的估计时间以及自动调节重置。 

编号	步骤	示例
5	将 Initialize 和 AutoTune 设置为 TRUE。	
6	请注意，当将 AutoTune 设为 True 时，输出会变为 Load 的值。	
7	观察到过程值快速增加，直至其接近其饱和点。	
8	观察过程值的稳定和波动情况。	

编号	步骤	示例																																																			
9	设置偏移量。	 The screenshot shows the 'Variable Monitoring' window with the 'Local Variables - IPID' tab selected. A table lists various variables. The variable 'My_AT_Param.Deviation' is highlighted with a red box, and its 'Logical Value' is set to '0.05'. <table><tr><th>Name</th><th>Logical Value</th><th>Physical Value</th></tr><tr><td>IPIDCONTROLLER_1</td><td></td><td></td></tr><tr><td>TempConvert_1</td><td></td><td></td></tr><tr><td>My_Limit_1</td><td></td><td></td></tr><tr><td>My_AT_Param</td><td></td><td></td></tr><tr><td>My_AT_Param.Deviation</td><td>0.05</td><td>N/A</td></tr><tr><td>My_AT_Param.Step</td><td>20.0</td><td>N/A</td></tr><tr><td>My_AT_Param.ATDynaSet</td><td>600.0</td><td>N/A</td></tr><tr><td>My_AT_Param.ATReset</td><td></td><td>N/A</td></tr><tr><td>My_SetPoint</td><td>0.0</td><td>N/A</td></tr><tr><td>AutoTune</td><td></td><td>N/A</td></tr><tr><td>Initialize</td><td></td><td>N/A</td></tr><tr><td>AutoMode</td><td></td><td>N/A</td></tr><tr><td>AT_Status</td><td>0</td><td>N/A</td></tr><tr><td>AbsError</td><td>27.6</td><td>N/A</td></tr><tr><td>TransferGain</td><td></td><td>N/A</td></tr><tr><td>My_GainTransfer_1</td><td></td><td></td></tr></table>	Name	Logical Value	Physical Value	IPIDCONTROLLER_1			TempConvert_1			My_Limit_1			My_AT_Param			My_AT_Param.Deviation	0.05	N/A	My_AT_Param.Step	20.0	N/A	My_AT_Param.ATDynaSet	600.0	N/A	My_AT_Param.ATReset		N/A	My_SetPoint	0.0	N/A	AutoTune		N/A	Initialize		N/A	AutoMode		N/A	AT_Status	0	N/A	AbsError	27.6	N/A	TransferGain		N/A	My_GainTransfer_1		
Name	Logical Value	Physical Value																																																			
IPIDCONTROLLER_1																																																					
TempConvert_1																																																					
My_Limit_1																																																					
My_AT_Param																																																					
My_AT_Param.Deviation	0.05	N/A																																																			
My_AT_Param.Step	20.0	N/A																																																			
My_AT_Param.ATDynaSet	600.0	N/A																																																			
My_AT_Param.ATReset		N/A																																																			
My_SetPoint	0.0	N/A																																																			
AutoTune		N/A																																																			
Initialize		N/A																																																			
AutoMode		N/A																																																			
AT_Status	0	N/A																																																			
AbsError	27.6	N/A																																																			
TransferGain		N/A																																																			
My_GainTransfer_1																																																					
10	将 Initialize 设为 False。	 The screenshot shows the 'Variable Monitoring' window with the 'Local Variables - IPID' tab selected. The variable 'Initialize' is highlighted with a red box, and its 'Logical Value' is set to 'False'. <table><tr><th>Name</th><th>Logical Value</th><th>Physical Value</th></tr><tr><td>IPIDCONTROLLER_1</td><td></td><td></td></tr><tr><td>TempConvert_1</td><td></td><td></td></tr><tr><td>My_Limit_1</td><td></td><td></td></tr><tr><td>My_AT_Param</td><td></td><td></td></tr><tr><td>My_AT_Param.Load</td><td>50.0</td><td>N/A</td></tr><tr><td>My_AT_Param.Deviation</td><td>0.05</td><td>N/A</td></tr><tr><td>My_AT_Param.Step</td><td>20.0</td><td>N/A</td></tr><tr><td>My_AT_Param.ATDynaSet</td><td>600.0</td><td>N/A</td></tr><tr><td>My_AT_Param.ATReset</td><td></td><td>N/A</td></tr><tr><td>My_SetPoint</td><td>0.0</td><td>N/A</td></tr><tr><td>Initialize</td><td>False</td><td>N/A</td></tr><tr><td>AutoMode</td><td></td><td>N/A</td></tr><tr><td>AT_Status</td><td></td><td>N/A</td></tr><tr><td>AbsError</td><td></td><td>N/A</td></tr><tr><td>TransferGain</td><td></td><td>N/A</td></tr><tr><td>My_GainTransfer_1</td><td></td><td></td></tr></table>	Name	Logical Value	Physical Value	IPIDCONTROLLER_1			TempConvert_1			My_Limit_1			My_AT_Param			My_AT_Param.Load	50.0	N/A	My_AT_Param.Deviation	0.05	N/A	My_AT_Param.Step	20.0	N/A	My_AT_Param.ATDynaSet	600.0	N/A	My_AT_Param.ATReset		N/A	My_SetPoint	0.0	N/A	Initialize	False	N/A	AutoMode		N/A	AT_Status		N/A	AbsError		N/A	TransferGain		N/A	My_GainTransfer_1		
Name	Logical Value	Physical Value																																																			
IPIDCONTROLLER_1																																																					
TempConvert_1																																																					
My_Limit_1																																																					
My_AT_Param																																																					
My_AT_Param.Load	50.0	N/A																																																			
My_AT_Param.Deviation	0.05	N/A																																																			
My_AT_Param.Step	20.0	N/A																																																			
My_AT_Param.ATDynaSet	600.0	N/A																																																			
My_AT_Param.ATReset		N/A																																																			
My_SetPoint	0.0	N/A																																																			
Initialize	False	N/A																																																			
AutoMode		N/A																																																			
AT_Status		N/A																																																			
AbsError		N/A																																																			
TransferGain		N/A																																																			
My_GainTransfer_1																																																					
11	控制器开始自动调节。等待 ATWarning 变为 2。	 The screenshot shows the 'IPIDCONTROLLER_1' block. The 'ATWarning' output is highlighted with a red box and set to '2'. <table><tr><th>Input/Output</th><th>Value</th></tr><tr><td>Process</td><td>Output</td></tr><tr><td>SetPoint</td><td>0.0</td></tr><tr><td>FeedBack</td><td>AbsoluteError</td></tr><tr><td>Auto</td><td>27.6</td></tr><tr><td>Initialize</td><td>ATWarning</td></tr><tr><td></td><td>2</td></tr><tr><td></td><td>OutGain</td></tr><tr><td></td><td>False</td></tr></table>	Input/Output	Value	Process	Output	SetPoint	0.0	FeedBack	AbsoluteError	Auto	27.6	Initialize	ATWarning		2		OutGain		False																																	
Input/Output	Value																																																				
Process	Output																																																				
SetPoint	0.0																																																				
FeedBack	AbsoluteError																																																				
Auto	27.6																																																				
Initialize	ATWarning																																																				
	2																																																				
	OutGain																																																				
	False																																																				

编号	步骤	示例
12	将 AutoTune 设为 False。	
13	观察调节值出现在 OutGains 中。	
14	将参数从 OutGain 传输到 My_Gains。	

编号	步骤	示例
15	观察使用调节的增益参数更新控制器。	

另请参见

[通过 IPIDController 功能块实现自动调节](#) 参考页数 [591](#)

通过自动调节实现结构化定时中断 (STI)

虽然 PID 指令在没有结构化定时中断 (STI) 控制下也有效，但是使用 STI 可提高自动调节成功率，因为自动调节将按固定周期运作。

添加 可选定时中断(STI)

属性

中断类型

可选定时中断 (STI)

STI ID:

STI0

STI描述

STI0

程序:

参数

☐ 自动开始

设置点

0

ms

确定

取消

应用

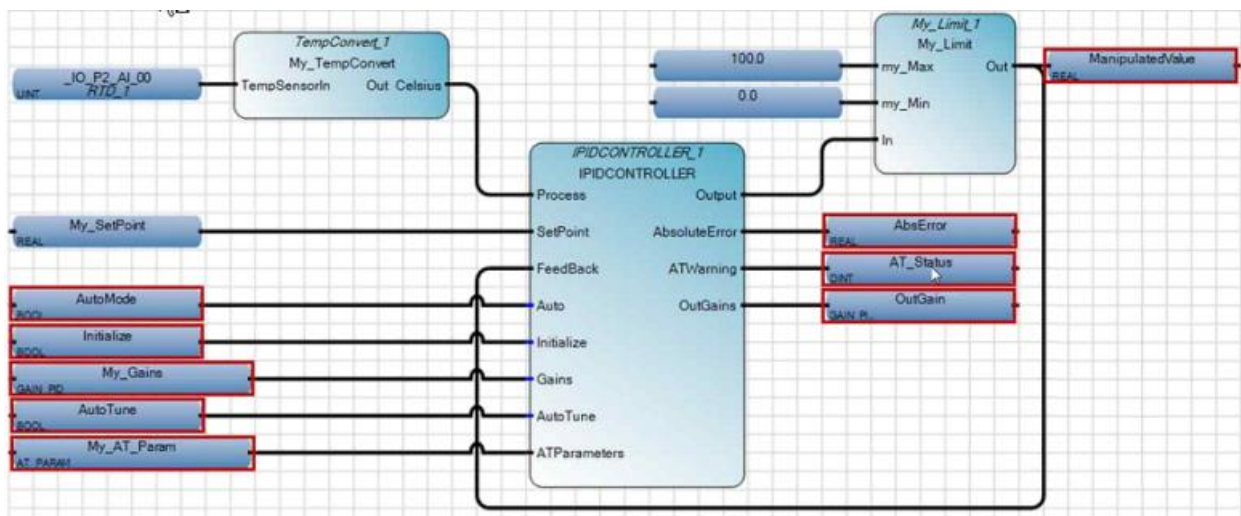
帮助

另请参见

[通过 IPIDController 功能块实现自动调节](#) 参考页数 [591](#)

示例：具有自动调节的 IPIDController

以下示例程序展示了用于配置自动调节参数的变量。



自动调节参数

下表描述了与示例中的每个参数一起使用以配置自动调节的变量。

输入参数		
变量	参数	说明
AutoMode	自动	PID 控制器的操作模式： TRUE - 控制器以正常模式运行。 FALSE - 忽略衍生项，强制控制器输出跟踪控制器限定范围内的反馈，使控制器能够切换回到自动，而不会增加输出
Initialize	Initialize	初始化 AutoTune 序列。 值的更改 (TRUE 更改为 FALSE 或 FALSE 更改为 TRUE) 会导致控制器在相应循环期间消除任何比例增益。
My_Gains	Gains	建立 IPIDController 的增益 PID。

输入参数

My_Gains.DirectActing	DirectActing	定义输出的操作类型。 TRUE - 正向作用 (输出与误差沿同一方向移动)。也就是说, 实际过程值要大于设定点值, 并且适当的控制器操作会增加输出。例如, 降温。 FALSE - 反向作用 (输出与误差沿相反方向移动)。也就是说, 实际过程值大于设定点值, 并且适当的控制器操作会降低输出。例如, 加热。
My_Gains.ProportionalGain	ProportionalGain	PID 的比例增益 (≥ 0.0001)。
My_Gains.TimeIntegral	TimeIntegral	PID 的时间积分值 (≥ 0.0001)。 振荡趋势会随着 ti 降低而提高。
My_Gains.TimeDerivative	TimeDerivative	PID 的时间微分值 (> 0.0)。 随着微分时间增加, 阻尼也会增加, 但是如果微分时间太大, 则会减小。
My_Gains.DerivativeGain	PID 的微分增益 (> 0.0)。	
AutoTune	当设置为 TRUE 且 Auto 和 Initialize 为 FALSE 时, 会启动 AutoTune 序列。	

ATParameters

Load	- 自动调节期间的初始输出值。 - 使过程值在加载时稳定下来
Deviation	- 一系列稳定的过程值的标准偏移。例如, 如果过程值稳定在 31.4 到 32.0 之间, 则偏移值将是 $(32.0-31.4)/2 = 0.3$。 - 某些过程值 (例如温度) 需要很久才能稳定。
步骤	自动调节过程考虑过程值如何对步长值的变更做出反应并派生 Gain 参数。
ATDynaSet	- 为要完成的自动调节分配时间。它必须长于自动调节过程所需的时间。 - 许多系统的常用值是 600 秒, 但是有些系统可能需要更多时间。
ATReset	- 如果为 TRUE, 则在自动调节完成后, 输出将被重置为“0”。 - 如果为 FALSE, 则在自动调节完成后, 输出仍将保留在加载值。

输出参数

参数	说明
AbsoluteError	来自控制器的绝对错误 (Process - SetPoint)。

输入参数	
ATWarning	自动调节序列的警告。可能的值有： 0 - 没有执行自动调节。 1 - 处于自动调节模式。 2 - 已执行自动调节。 -1 - ERROR 1 输入自动设置为 TRUE，不可能进行自动调节。 -2 - ERROR 2 自动调节错误，ATDynaSet 已过期。
OutGains	在 AutoTune 序列之后计算的增益。

另请参见

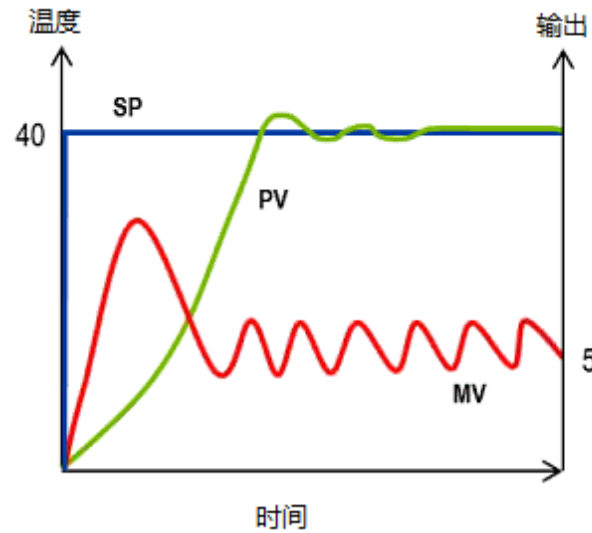
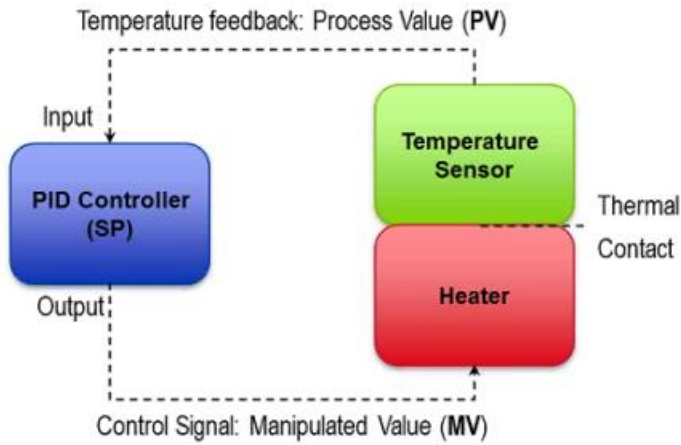
[使用比例积分微分指令](#) 参考页数 [590](#)

示例：如何创建操作值的反馈回路

通过添加操作值的反馈回路可提供 MV 的最大值和最小值，从而防止调节过头。

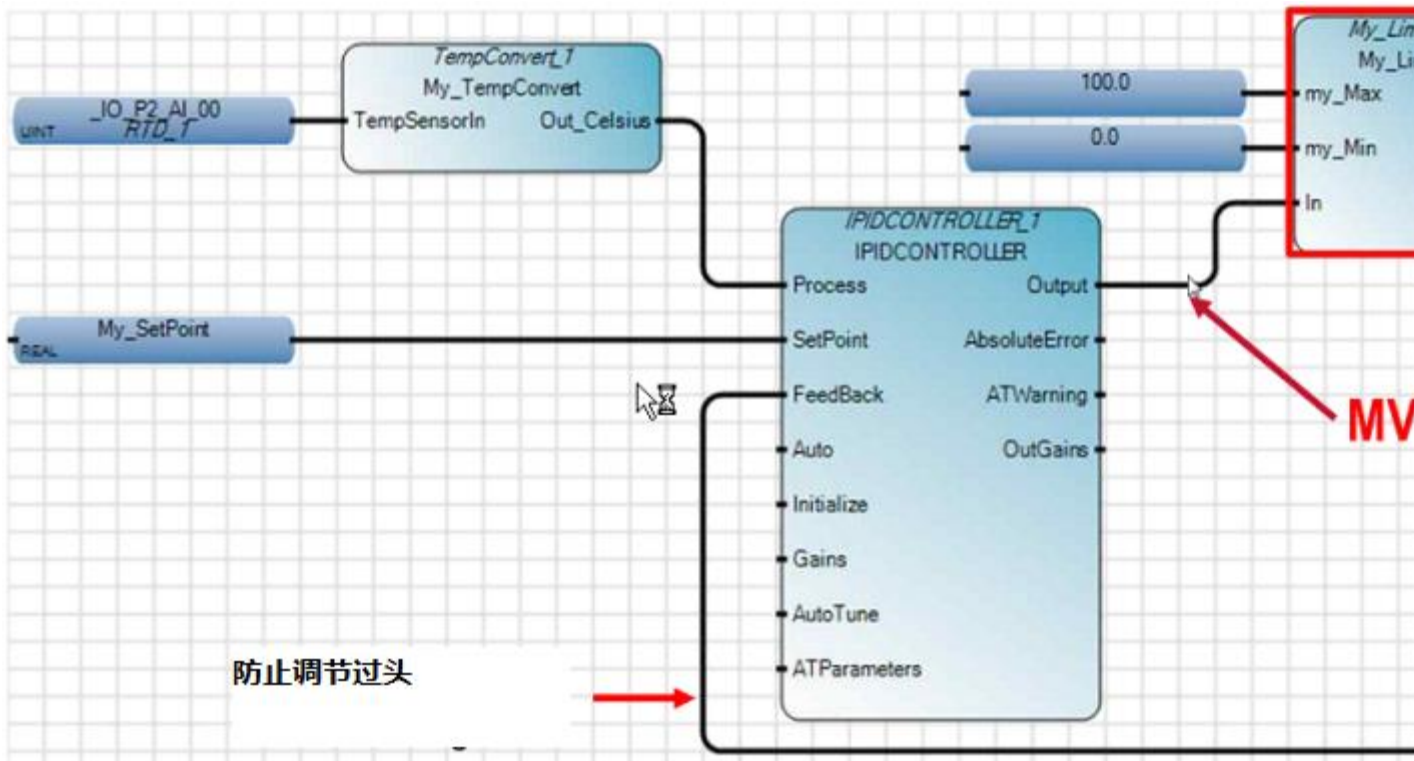
温度反馈回路示例

在温度控制过程开始时，过程值 (PV) 和设定值 (SP) 之间的差异较大，如下图所示。在此温度反馈回路示例中，PV 从 0 摄氏度开始，向 40 摄氏度的 SP 值移动。另请注意，高操作值和低操作值 (MV) 之间的波动会随时间推移而减少并逐渐稳定。MV 的行为取决于各个 P、I 和 D 参数中使用的值。



具有反馈回路的 IPIDController

以下功能块图包括操作值的反馈回路，它通过提供 MV 的最大值和最小值来防止调节过头。



另请参见

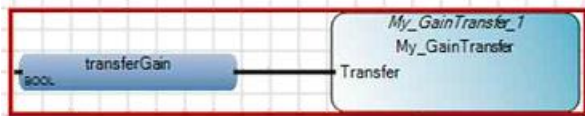
[使用比例积分微分指令](#) 参考页数 [590](#)

示例：如何将 UDFB 添加到 PID 程序

可在主程序外部添加 UDFB 以执行专门的功能，例如转换单位或传输值。

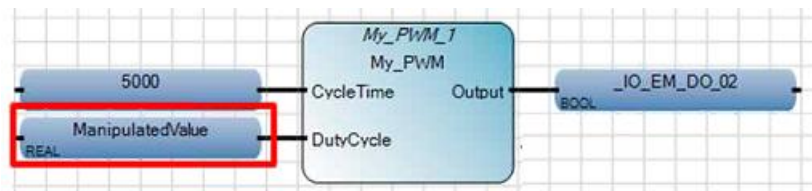
传输自动调节增益值

该 UDFB 将 Autotune 增益值传输到 My_GainTransfer，从而使控制器可以使用该值。



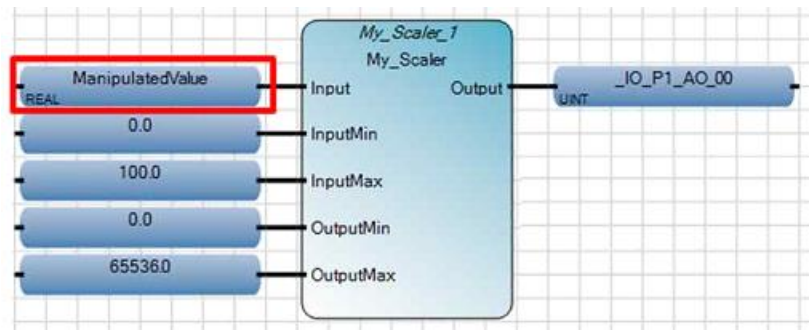
将操作值转换为数字输出

此 UDFB 将操作值 (MV) 转换为数字输出 (DO) 从而可用其控制数字输入 n(DI)。



将操作值转换为模拟输出

此 UDFB 将操作值 (MV) 转换为模拟输出 (AO) ,从而可用其控制模拟输入 (AI)。



另请参见

[使用比例积分微分指令](#) 参考页数 [590](#)

示例：如何创建 IPIDController 程序以控制温度

温度控制程序用于维持控制区中的温度。

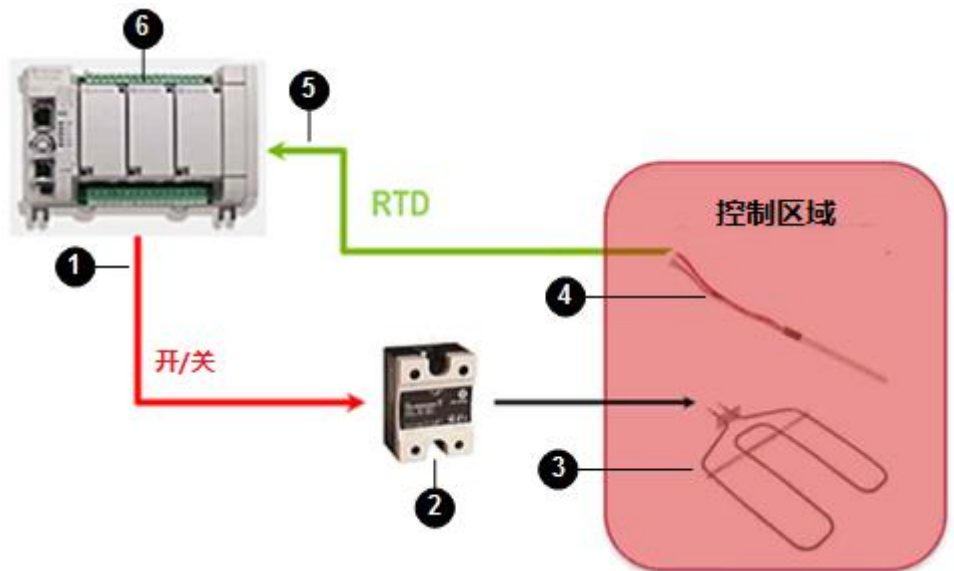
设定点、过程值和操作值

下表定义了 SP、PV 和 MV 值在温度控制程序中的使用方式。

项目	说明
设定点 (SP)	用于定义控制区温度的温度测量值（以摄氏度表示）。
过程值 (PV)	必须换算成与 SP 相同的单位，即温度测量值（摄氏度）。
操作值 (MV)	必须转换为模拟值，从而可将其输出至 PWM 以控制加热元素。

温度控制系统

下图展示了温度控制系统中由温度控制程序控制的组件。图后的表格描述了在控制程序运行时发生的事件。



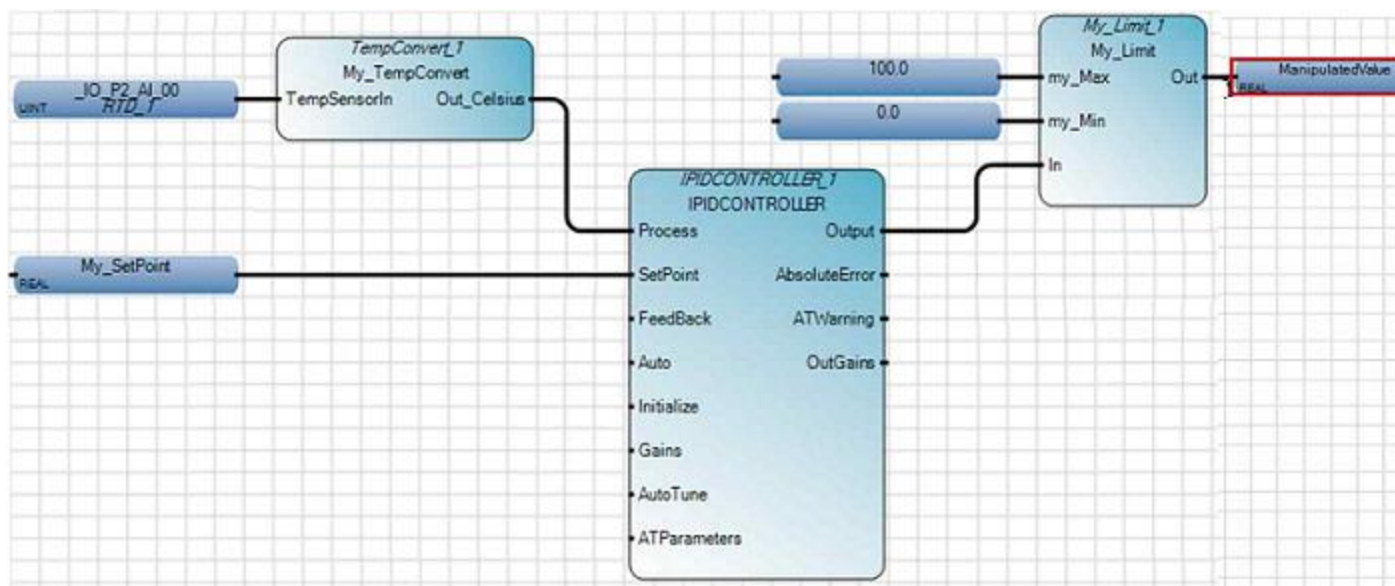
温度控制程序的事件顺序

下表标识了温度控制系统中的组件，并按顺序描述了温度控制程序运行时在系统中发生的事件。

不支持	项目	说明
①	控制器输出	将 MV 发送至 PWM (On/Off)。
②	脉宽调制 (PWM 温度控制器	用于控制加热元素的固态中继。
③	加热元素	提高控制区温度。
④	电阻温度检测器 (RTD)	测量控制区中的温度，并将 PV (RTD 信号) 发送至控制器输入。
⑤	控制器输入	接收 PV (RTD 信号) 。
⑥	PLC 程序	将 PV(RTD 信号) 转换为与 SP(摄氏度) 相同的单位，确定 PV 和 SP 之间的差异，并根据 P、I 和 D 参数中定义的参数值调整 MV。

示例：温度控制功能块图

本功能块图展示了应用程序中用于在控制区控制温度的预定义和用户自定义功能块。



另请参见

[温度控制程序](#) 参考页数 [604](#)

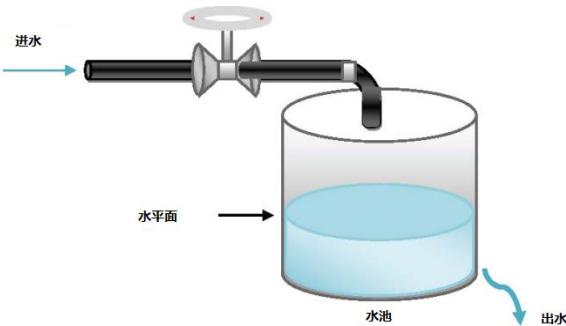
另请参见

[示例：如何创建 IPIDController 程序以控制供水水位](#) 参考页数 [607](#)

[使用比例积分微分指令](#) 参考页数 [590](#)

示例：如何创建 IPIDController 程序以控制供水水位

供水水位控制程序示例用于使正在向外供水的供水水箱保持有足够的水量。电磁阀控制进水量并按预设的速率加注水箱；外流的水也会按预设速率加以控制。



程序示例信息

供水水位程序示例包括以下信息。

- 在控制过程中发生的事件顺序
- 设定点、过程值以及操作值在控制系统中的使用方式
- 用于展示 IPIDController 和其他指令块的功能块图示例

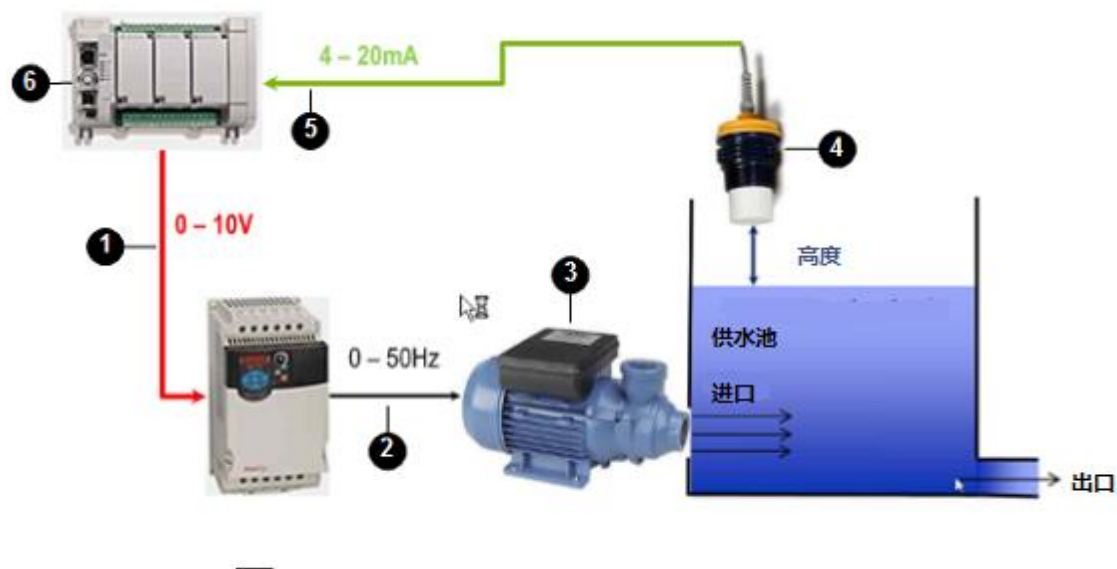
设定点、过程值和操作值

下表定义了 SP、PV 和 MV 值在供水水位程序中的使用方式。

项目	说明
设定点 (SP)	用于定义目标供水水位的高度测量值。
过程值 (PV)	必须将 4-20mA 换算成与 SP 相同的单位，即高度测量值。
操作值 (MV)	必须转换为模拟值，从而可将其输出至驱动器以控制泵。

供水水位系统

下图展示了供水水位系统中由供水水位程序控制的组件。图后的表格描述了在控制程序运行时发生的事件。



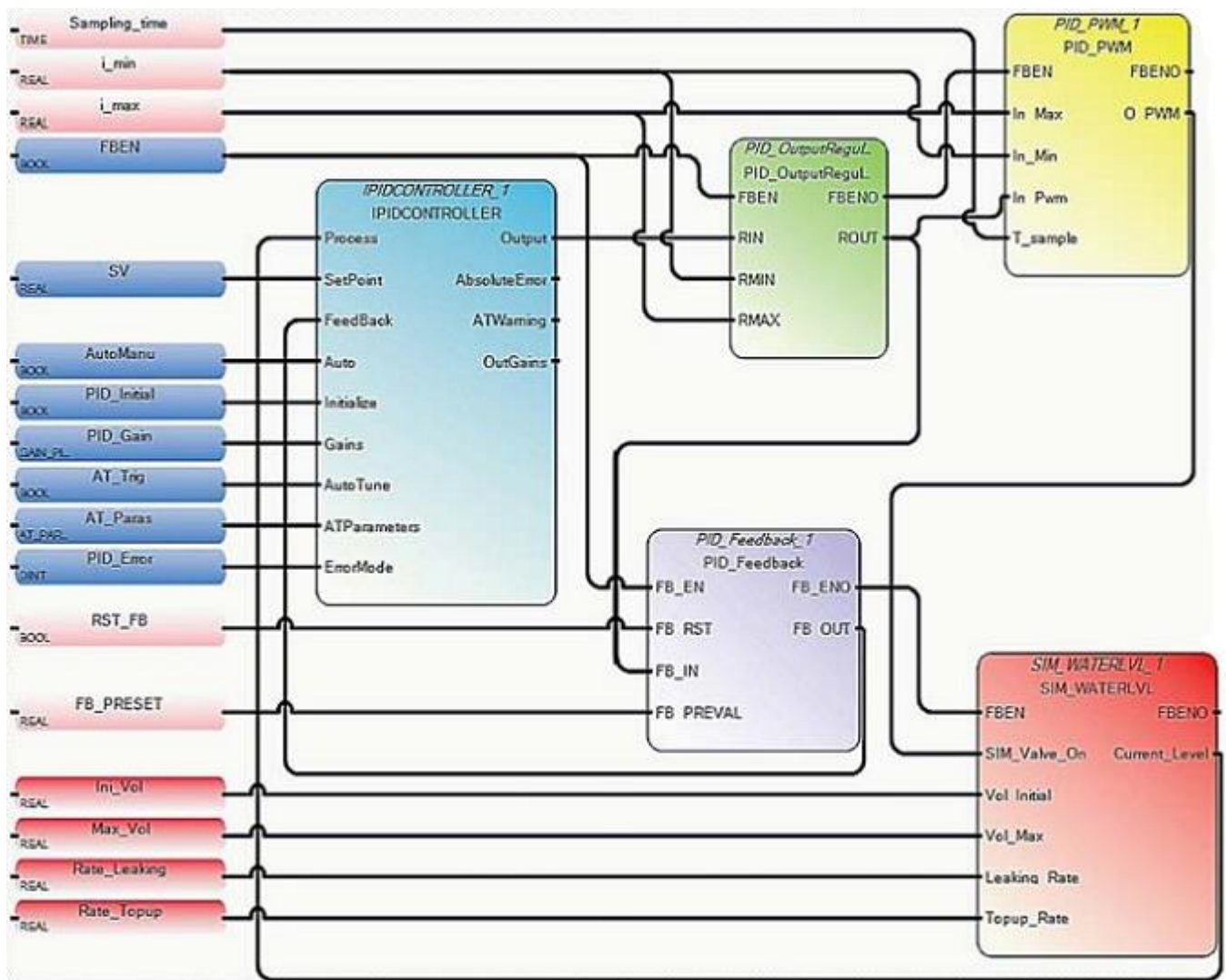
供水水位系统中的事件顺序

下表标识了供水系统中的组件，并按顺序描述了供水水位程序运行时在系统中发生的事件。

不支持	项目	说明
❶	控制器输出	将 MV 发送至 PowerFlex 驱动器 (0-10V)。
❷	PowerFlex 驱动器	控制水泵 (0-50Hz)。
❸	水泵	控制供水池中的水位。
❹	输出传输设备	测量供水水位高度 (4-20mA)，并将 PV 发送至控制器。
❺	控制器输入	接收 PV (供水水位 4-20mA)。
❻	PLC 程序	将 PV 转换为与 SP (高度测量值) 相同的单位，确定 PV 和 SP 之间的差异，并根据 P、I 和 D 参数中定义的参数值调整 MV。

示例：供水水位控制功能块图

以下功能块图展示了用于控制供水水位的程序中预定义和用户定义的功能块。



在水位 FBD 中使用的功能块和 UDFB

此应用程序采用功能块图 (FBD) 语言开发，使用下表中所述的功能块。

功能块	说明
IPIDController 功能块	提供 PID 过程控制
PID_OutputRegulator UDFB	在安全范围内调节 IPIDCONTROLLER 输出,以确保不会损坏该过程中使用的硬件 示例代码: IF RMIN ≤ RIN ≤ RMAX, then ROUT = RIN, IF RIN < RMIN, then ROUT = RMIN, IF RIN > RMAX, then ROUT = RMAX
PID_Feedback UDFB	用作多路复用器 示例代码: IF "FB_RST" is false, FB_OUT=FB_IN; If "FB_RST" is true, then FB_OUT=FB_PREVAL.
PID_PWM UDFB	提供 PWM 函数, 用于将实型值转换为与时间相关的 ON/OFF 输出
SIM_WATERLVL UDFB	模拟应用程序示例中的过程

另请参见

[供水水位控制程序](#) 参考页数 [607](#)

另请参见

[示例：如何创建 IPIDController 程序以控制温度](#) 参考页数 [604](#)

[使用比例积分微分指令](#) 参考页数 [590](#)

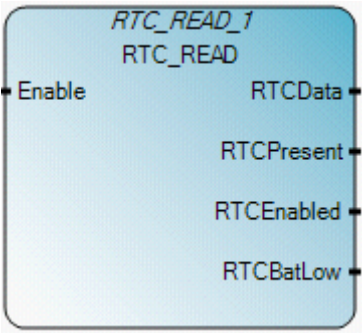
实时时钟 (RTC) 指令

实时时钟指令用于配置日历和时钟。

功能块	说明
RTC_READ 参考页数 427	RTC_READ 可读取 RTC 预置和 RTC 信息。
RTC_SET 参考页数 430	RTC_SET 可设置 RTC 状态或写入 RTC 信息。

RTC_READ

RTC_READ 可读取 RTC 预置和 RTC 信息。



RTC_READ 操作

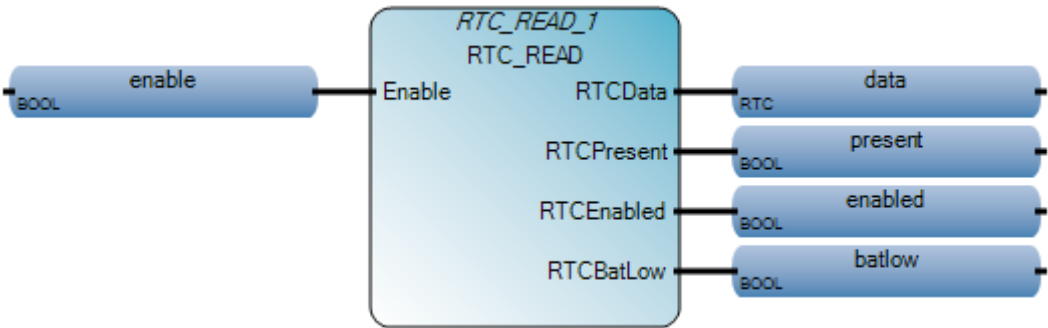
- 当与采用嵌入式 RTC 的 Micro810 控制器一起使用时，RTCBatLow 始终设置为零 (0)。
- 如果由于失去电源而使嵌入式 RTC 丢失其电荷/内存，则 RTCEnabled 始终设置为零 (0)。

参数

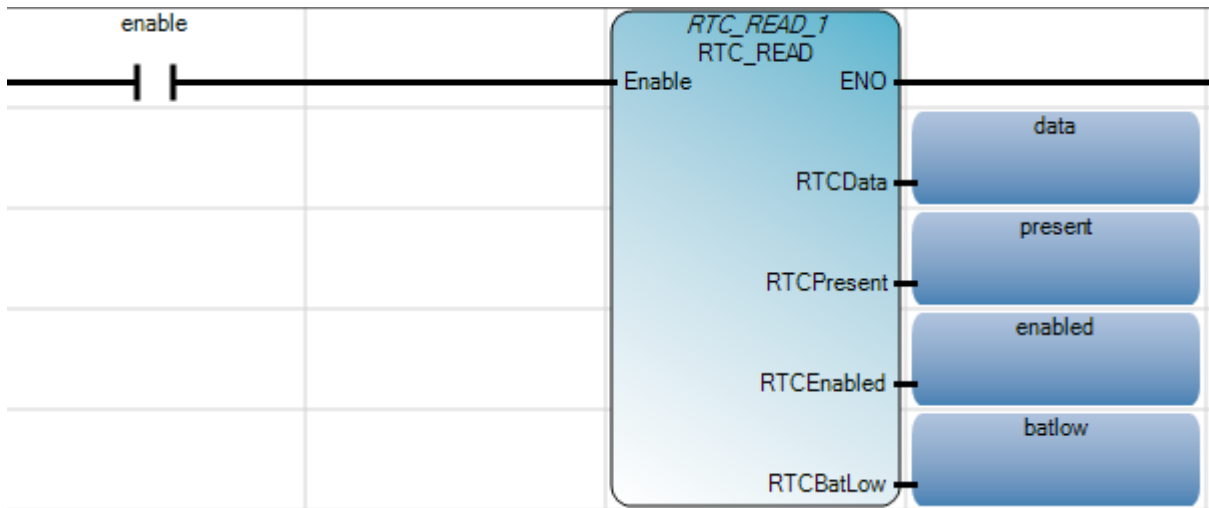
参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，执行 RTC 信息读取。 当 Enable = FALSE 时，不进行读取操作，并且 RTC 输出数据无效。
RTCData	输出	RTC	RTC 数据信息：yy/mm/dd、hh/mm/ss、星期。 请参见 RTC 数据类型 参考页数 429。
RTCPresent	输出	BOOL	TRUE - 已插入 RTC 硬件。 FALSE - 未插入 RTC 硬件。
RTCEnabled	输出	BOOL	TRUE - 已启用 RTC 硬件（计时）。 FALSE - 已禁用 RTC 硬件（未计时）。
RTCBatLow	输出	BOOL	TRUE - RTC 电池电量低。 FALSE - RTC 电池电量不低。
ENO	输出	BOOL	启用输出。 仅适用于 LD 编程。

RTC_READ 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本

```
1  RTC_READ_1(enable);
2  data := RTC_READ_1.RTCDData;
3  present := RTC_READ_1.RTCPresent;
4  enabled := RTC_READ_1.RTCEnabled;
5  batlow := RTC_READ_1.RTCBatLow;
```

RTC_READ_1(

void **RTC_READ_1**(BOOL Enable)
Type : RTC_READ, Read RTC module information.

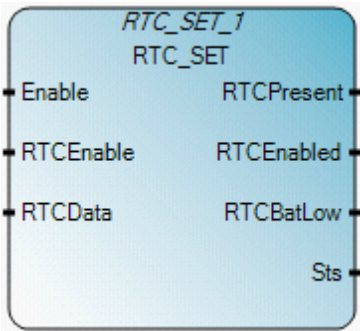
另请参见

[RTC 数据类型](#) 参考页数 [429](#)

[输入/输出指令](#) 参考页数 [381](#)

RTC_SET

RTC_SET 可设置 RTC 状态或写入 RTC 信息。

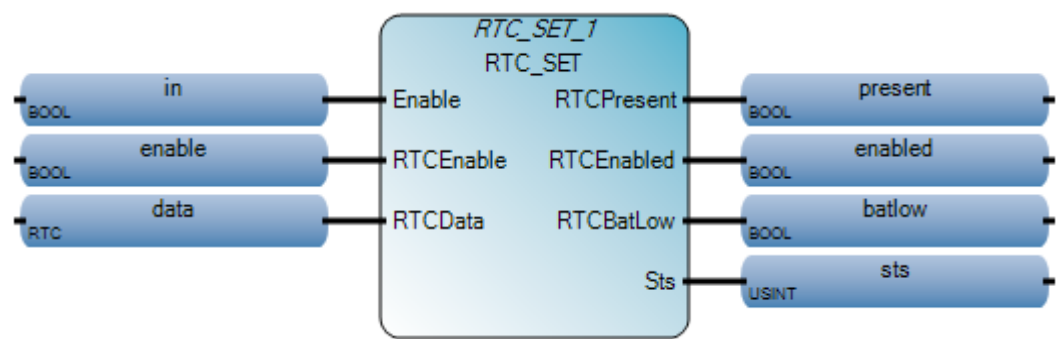


参数

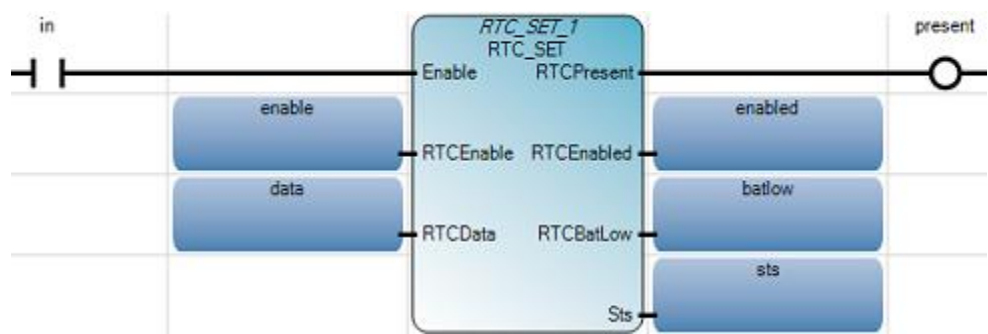
参数	参数类型	数据类型	说明
启用	输入	BOOL	功能块启用。 当 Enable = TRUE 时，使用输入的 RTC 信息执行 RTC 设置。 当 Enable = FALSE 时，不进行读取操作，并且 RTC 输出数据无效。
RTCEnable	输入	BOOL	TRUE - 启用 RTC 及指定的 RTC 数据。 FALSE - 禁用 RTC。 注意：Micro810 控制器会忽略此参数。
RTCData	输入	RTC	RTC 数据信息：yy/mm/dd、hh/mm/ss、星期。 当 RTCEnable = 0 时，则忽略此 RTC 数据。 请参见 RTC 数据类型 参考页数 429。
RTCPresent	输出	BOOL	TRUE - 已插入 RTC 硬件。 FALSE - 未插入 RTC 硬件。
RTCEnabled	输出	BOOL	TRUE - 已启用 RTC 硬件（计时）。 FALSE - 已禁用 RTC 硬件（未计时）。
RTCBatLow	输出	BOOL	TRUE - RTC 电池电量低。 FALSE - RTC 电池电量不低。
Sts	输出	USINT	读取操作状态。 请参见 RTC 设置状态值 参考页数 432

RTC_SET 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1  RTC_SET_1(in, enable, data);
2  present := RTC_SET_1.RTCPresent;
3  enabled := RTC_SET_1.RTCEnabled;
4  batlow := RTC_SET_1.RTCBatLow;
5  sts := RTC_SET_1.Sts;
```

RTC_SET_1{

void RTC_SET_1(BOOL Enable, BOOL RTCEnable, RTC RTCData)
Type : RTC_SET, Set RTC data to RTC module.

另请参见

[RTC 数据类型](#) 参考页数 [429](#)

[输入/输出指令](#) 参考页数 [381](#)

RTC 数据类型

下表描述了 RTC 数据类型。

参数	数据类型	说明
Year	UINT	RTC 的年设置。16 位值，有效范围为 2000 (1 月 01 日 00:00:00) 到 2098 (12 月 31 日 23:59:59)
Month	UINT	RTC 的月设置。

参数	数据类型	说明
Day	UINT	RTC 的日设置。
Hour	UINT	RTC 的小时设置。
Minute	UINT	RTC 的分钟设置。
Second	UINT	RTC 的秒钟设置。
DayOfWeek	UINT	RTC 的星期设置。RTC_SET 将忽略此参数。

另请参见

[RTC_READ](#) 参考页数 [427](#)

RTC 设置状态值

下表描述了 RTCSet 值:

状态值	状态说明
0x00	未启用功能块 (无操作)。
0x01	RTC 设置操作成功。
0x02	RTC 设置操作失败。

另请参见

[RTC_SET](#) 参考页数 [430](#)

字符串操作指令

字符串操作指令用于修改从集合或字母表选择的符号序列，以更改输出状态。

提示 要正确读取包含特殊字符的输入字符串，请在包含功能块实例的程序联机后输入字符串字符。
：

函数	说明
ASCII 参考页数 617	字符 -> ASCII 代码
CHAR 参考页数 619	ASCII 代码 -> 字符
DELETE 参考页数 621	删除子字符串
FIND 参考页数 624	查找子字符串
INSERT 参考页数 626	插入字符串
LEFT 参考页数 628	提取字符串的左侧
MID 参考页数 630	提取字符串的中间
MLEN 参考页数 633	获取字符串长度
REPLACE 参考页数 637	替换子字符串
RIGHT 参考页数 634	提取字符串的右侧

ASCII

ASCII 生成字符串中字符的 ASCII 代码。

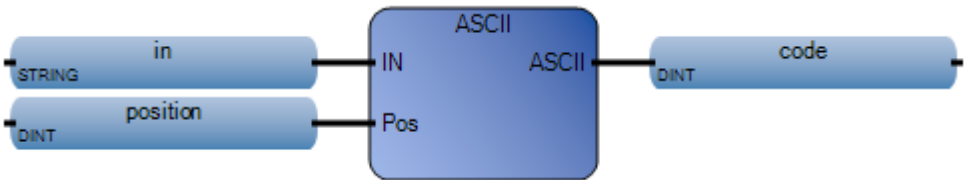


参数

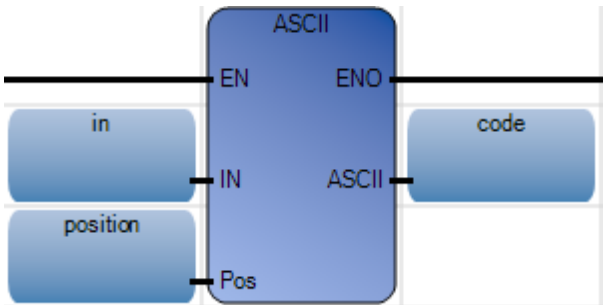
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，显示字符的 ASCII 代码。 当 EN = FALSE 时，没有显示操作。
IN	输入	STRING	任何非空字符串。
Pos	输入	DINT	选定字符的位置（位于集合 [1.. len] 内，其中 len 是 IN 字符串的长度）。
ASCII	输出	DINT	当 Pos 超出字符串范围时，选定字符的代码（位于集合 [0 ..255] 内）生成 0。
ENO	输出	BOOL	启用输出。

ASCII 函数语言示例

功能块图



梯形图



结构化文本

```
1 position := 1;  
2 code := ASCII(in, position);
```

ASCII (

DINT **ASCII**(STRING IN, DINT Pos)
Character -> ASCII code

(* 与之等效的 ST : *)

FirstChr := ASCII (message, 1);

(* FirstChr 是字符串第一个字符的 ASCII 代码 *)

结果

变量监视

用户全局变量 - 局部变量 - RA_ASCII_LD 系统变量- Micro830 I/C

	名称	逻辑值	实际值	锁定	数据类型	维度
▶	in	abcd	不可用	☐	STRING	
	position	1	不可用	☐	DINT	
	code	97	不可用	☐	DINT	

关闭

另请参见

[字符串操作指令](#) 参考页数 [617](#)

CHAR

对于给定的 ASCII 代码，CHAR 提供包含一个字符的字符串。

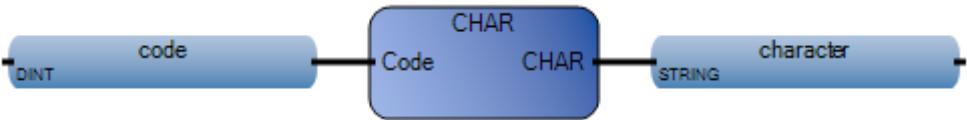


参数

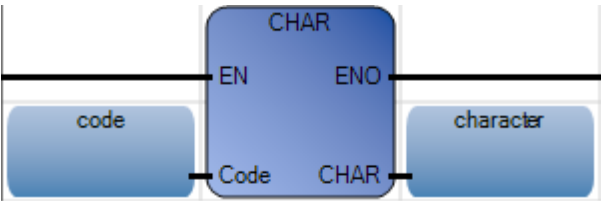
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，提供由一个字符组成的字符串。 当 EN = FALSE 时，没有任何操作。
Code	输入	DINT	位于以下集合内的代码：[0 ..255]。
CHAR	输出	STRING	由一个字符组成的字符串。 该字符具有输入代码中给定的 ASCII 代码。
ENO	输出	BOOL	启用输出。

CHAR 函数语言示例

功能块图

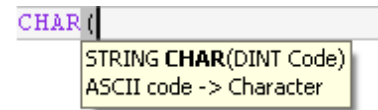


梯形图



结构化文本

```
1 code := 97;  
2 character := CHAR(code);
```



(* 与之等效的 ST : *)

Display := CHAR (value + 48);

(* 值位于集合 [0..9] 内 *)

(* 48 为 ascii 代码“0” *)

(* 结果为由一个字符组成的字符串，值范围为从“0”到“9” *)

结果



另请参见

[字符串操作指令](#) 参考页数 [617](#)

DELETE

DELETE 删除字符串的一部分。

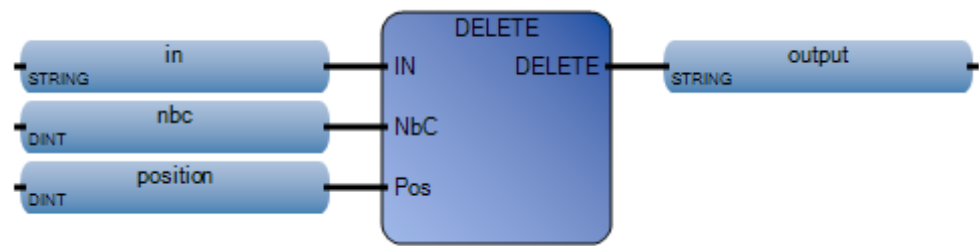


参数

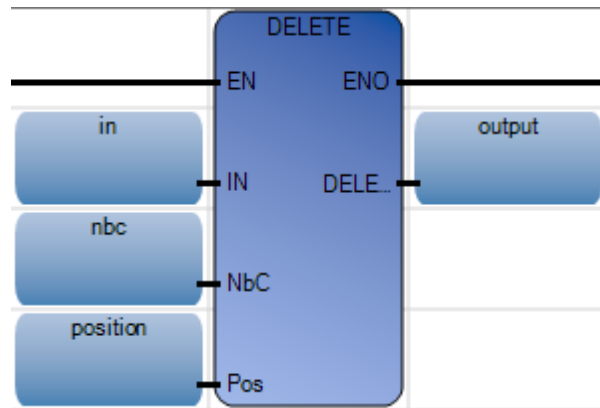
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，删除字符串的指定部分。 当 EN = FALSE 时，没有任何操作。
IN	输入	STRING	任何非空字符串。
NbC	输入	DINT	要删除的字符数。
Pos	输入	DINT	第一个删除的字符的位置 (字符串的第一个字符的位置为 1)。
DELETE	输出	STRING	可以为以下字符串： • 已修改的字符串 • 空字符串 (如果 Pos < 1) • 初始字符串 (如果 Pos > IN 字符串长度) • 初始字符串 (如果 NbC <= 0)
ENO	输出	BOOL	启用输出。

DELETE 函数语言示例

功能块图



梯形图



结构化文本

```

1  nbc := 3;
2  position := 2;
3  output := DELETE(in, nbc, position);

```

DELETE {
 STRING **DELETE**(STRING IN, DINT NbC, DINT Pos)
 Delete sub-string

(* 与之等效的 ST : *)

complete_string := INSERT ('ABCD ', 'EFGH', 5); (* complete_string 为
 “ABCDEFGH” *)

sub_string := DELETE (complete_string, 4, 3); (* sub_string 为“ABGH” *)

结果



另请参见

[字符串操作指令](#) 参考页数 [617](#)

FIND

FIND 在字符串中定位并提供子字符串的位置。



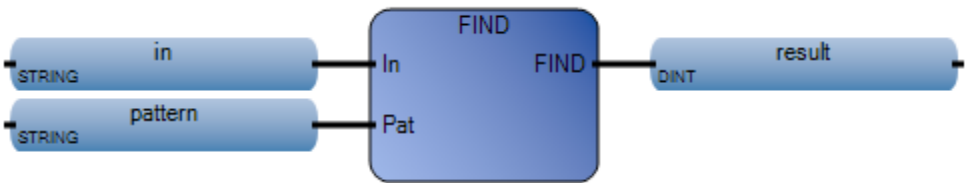
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，在字符串中定位位置。 当 EN = FALSE 时，没有定位操作。
In	输入	STRING	任何非空字符串。
Pat	输入	STRING	任何非空字符串（模式）。
FIND	输出	DINT	可以为下列值： • 0（如果找不到子字符串 Pat） • 子字符串 Pat 第一次出现的第一个字符的位置（第一个位置为 1） 注意：此函数区分大小写。

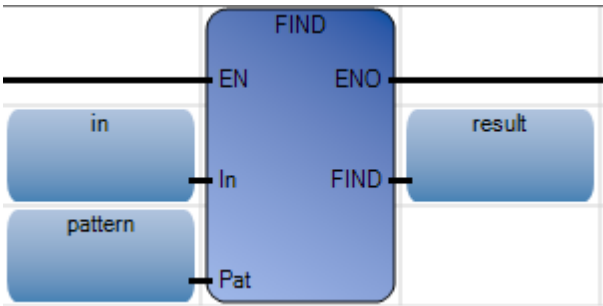
参数	参数类型	数据类型	说明
ENO	输出	BOOL	启用输出。

FIND 函数语言示例

功能块图

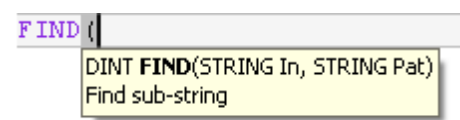


梯形图



结构化文本

```
1 result := FIND(in, pattern);
```



(* 与之等效的 ST : *)

```
complete_string := 'ABCD' + 'EFGH'; (* complete_string 为“ABCDEFGH” *)
```

```
found := FIND (complete_string, 'CDEF'); (* found 为 3 *)
```

结果



另请参见

[字符串操作指令](#) 参考页数 [617](#)

INSERT

INSERT 在字符串中用户指定的位置插入子字符串。



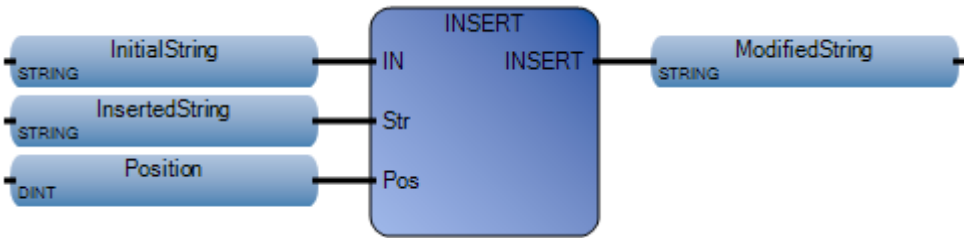
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，在字符串中插入子字符串。 当 EN = FALSE 时，没有任何操作。
IN	输入	STRING	初始字符串。
Str	输入	STRING	要插入的字符串。
Pos	输入	DINT	插入的位置 在该位置前完成插入 (第一个有效位置为 1)。

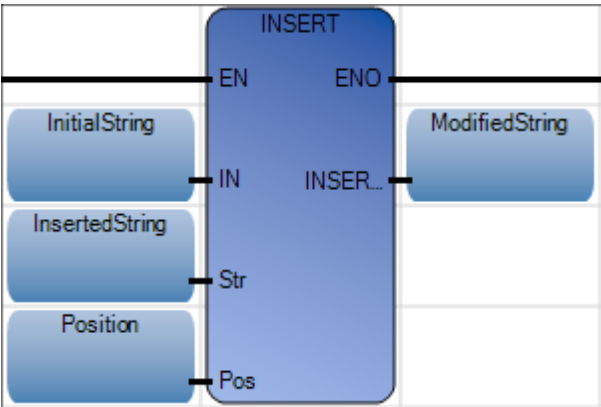
参数	参数类型	数据类型	说明
INSERT	输出	STRING	已修改的字符串。可以为下列值： - 空字符串（如果 Pos <= 0） - 两个字符串的连接（如果 Pos 大于 IN 字符串的长度）
ENO	输出	BOOL	启用输出。

INSERT 函数语言示例

功能块图



梯形图



结构化文本

```
1 Position := 3;  
2 ModifiedString := INSERT(InitialString, InsertedString, Position);
```

```
INSERT {  
  STRING INSERT (STRING IN, STRING Str, DINT Pos)  
  Insert string
```

(* 与之等效的 ST : *)

MyName := INSERT ('Mr JONES', 'Frank ', 4);

(* MyName 为“Mr Frank JONES” *)

结果



另请参见

[字符串操作指令](#) 参考页数 [617](#)

LEFT

从字符串的左端，LEFT 生成定义的字符数。

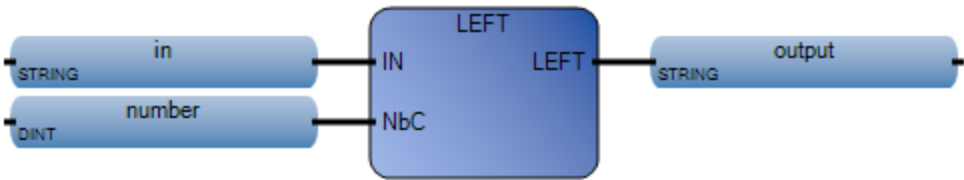


参数

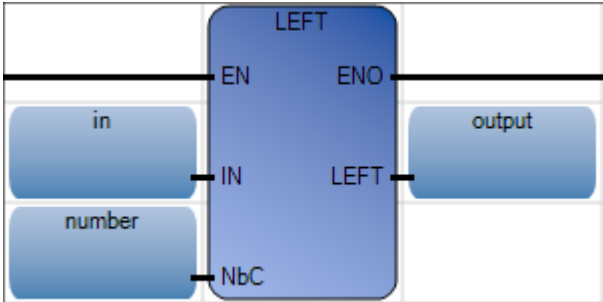
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，从字符串左端生成字符数。 当 EN = FALSE 时，没有任何操作。
IN	输入	STRING	任何非空字符串。
NbC	输入	DINT	要提取的字符数。此数字不能大于 IN 字符串的长度。
LEFT	输出	STRING	IN 字符串的左侧部分（其长度 = NbC）。可以为下列值： - 空字符串（如果 NbC <= 0） - 完整 IN 字符串（如果 NbC >= IN 字符串长度）
ENO	输出	BOOL	启用输出。

LEFT 函数语言示例

功能块图

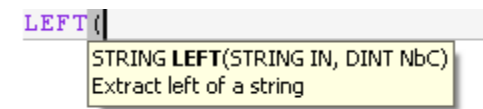


梯形图



结构化文本

```
1 number := 3;  
2 output := LEFT(in, number);
```



(* 与之等效的 ST : *)

```
complete_string := RIGHT ('12345678', 4), LEFT ('12345678', 4), 5;
```

(* 完整字符串为“56781234”

从 RIGHT 调用发出的值为“5678”

从 LEFT 调用发出的值为“1234”*)

结果



另请参见

[字符串操作指令](#) 参考页数 [617](#)

MID

使用提供的字符位置和数量，MID 生成所需的字符串部分。



参数

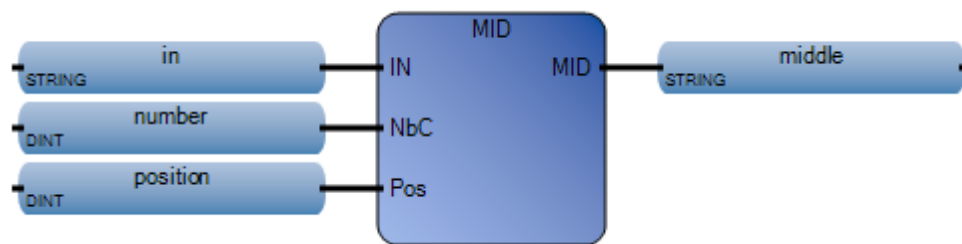
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，生成字符串的一部分。 当 EN = FALSE 时，不生成任何操作。
IN	输入	STRING	任何非空字符串。
NbC	输入	DINT	提取的字符数不能大于 IN 字符串的长度。
Pos	输入	DINT	子字符串的位置。子字符串的第一个字符将是 Pos 指向的字符（第一个有效位置为 1）。
MID	输出	STRING	字符串的中间部分（其长度 = NbC）。 当提取的字符数超出 IN 字符串长度时，将自动重新计算 NbC 以仅获取字符串的其余部分。当 NbC 或 Pos 为零或负数时，返回空字符串。
ENO	输出	BOOL	启用输出。

MID 函数语言示例

功能块图



梯形图



结构化文本

```

1  number := 3;
2  position := 2;
3  middle := MID(in, number, position);

```

MID {
 STRING **MID**(STRING IN, DINT NbC, DINT Pos)
 Extract middle of a string

(* 与之等效的 ST : *)

```
sub_string := MID('abcdefgh', 2, 4);
```

(* sub_string 为“de” *)

结果



另请参见

[字符串操作指令](#) 参考页数 [617](#)

MLEN

MLEN 成字符串的长度。

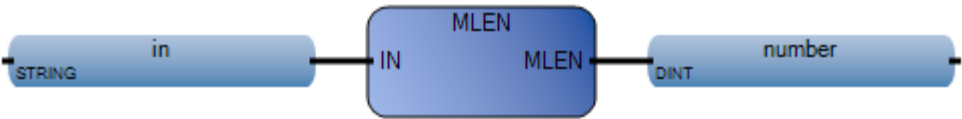


参数

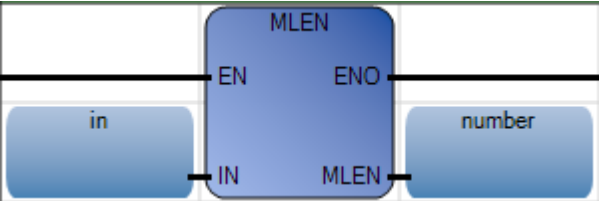
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，生成字符串长度。 当 EN = FALSE 时，没有任何操作。
IN	输入	STRING	任何字符串。
MLEN	输出	DINT	IN 字符串中的字符数。
ENO	输出	BOOL	启用输出。

MLEN 函数语言示例

功能块图



梯形图



结构化文本

```
1 number := MLEN(in);
```



(* 与之等效的 ST : *)

```
nbchar := MLEN (complete_string);
```

```
If (nbchar < 3) Then Return; End_if;
```

```
prefix := LEFT (complete_string, 3);
```

(* 此程序提取字符串左侧的 3 个字符，将结果放在前缀字符串变量中。
。如果字符串长度小于 3 个字符，则不执行任何操作 *)

结果



另请参见

[字符串操作指令](#) 参考页数 [617](#)

RIGHT

从字符串的右端，RIGHT 生成定义的字符数。

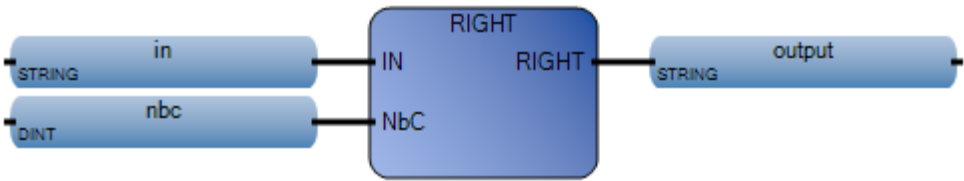


参数

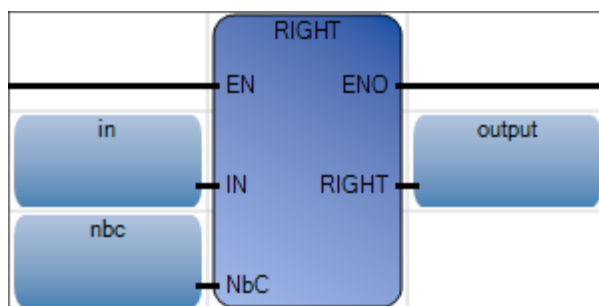
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，从字符串右端生成指定的字符数。 当 EN = FALSE 时，没有任何操作。
IN	输入	STRING	任何非空字符串。
NbC	输入	DINT	要提取的字符数。此数字不能大于 IN 字符串的长度。
RIGHT	输出	STRING	字符串的右侧部分 (长度 = NbC)。可以为下列值： - 空字符串 (如果 NbC <= 0) - 完整字符串 (如果 NbC >= 字符串长度)
ENO	输出	BOOL	启用输出。

RIGHT 函数语言示例

功能块图



梯形图



结构化文本

```

1  nbc := 2;
2  output := RIGHT(in, nbc);

```

RIGHT (

STRING **RIGHT**(STRING IN, DINT NbC)
Extract right of a string

(* 与之等效的 ST : *)

```
complete_string := RIGHT ('12345678', 4), LEFT ('12345678', 4),5;
```

(* 完整字符串为“56781234”

从 RIGHT 调用发出的值为“5678”

从 LEFT 调用发出的值为“1234”

*)

结果



另请参见

[字符串操作指令](#) 参考页数 [617](#)

REPLACE

REPLACE 将字符串的一部分替换为新的字符集。



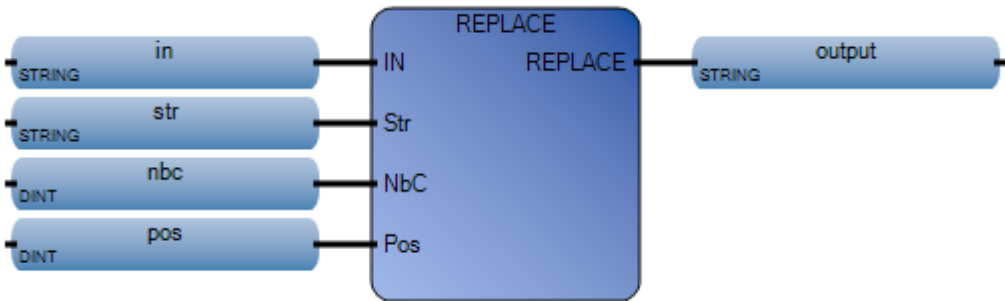
参数

参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，将字符串的一部分替换为新字符。 当 EN = FALSE 时，没有任何操作。
IN	输入	STRING	任何字符串。
Str	输入	STRING	要插入的字符串（用于替换 NbC 字符）。
NbC	输入	DINT	要删除的字符数。
Pos	输入	DINT	第一个已修改字符的位置 （第一个有效位置为 1）。

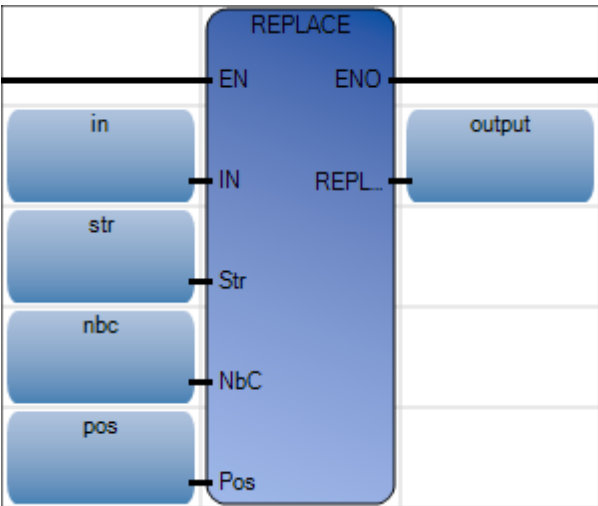
参数	参数类型	数据类型	说明
REPLACE	输出	STRING	已修改的字符串。在位置 Pos 处删除 NbC 字符，然后在此位置插入子字符串 Str。可以为下列值： - 空字符串（如果 Pos <= 0） - 字符串连接 (IN+Str)（如果 Pos 大于 IN 字符串的长度） - 初始字符串 IN（如果 NbC <= 0）
ENO	输出	BOOL	启用输出。

REPLACE 函数语言示例

功能块图

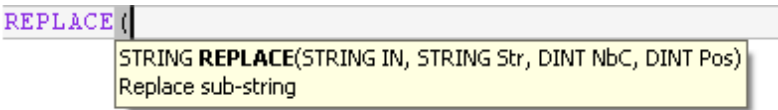


梯形图



结构化文本

```
1 nbc := 4;  
2 pos := 2;  
3 output := REPLACE(in, str, nbc, pos);
```



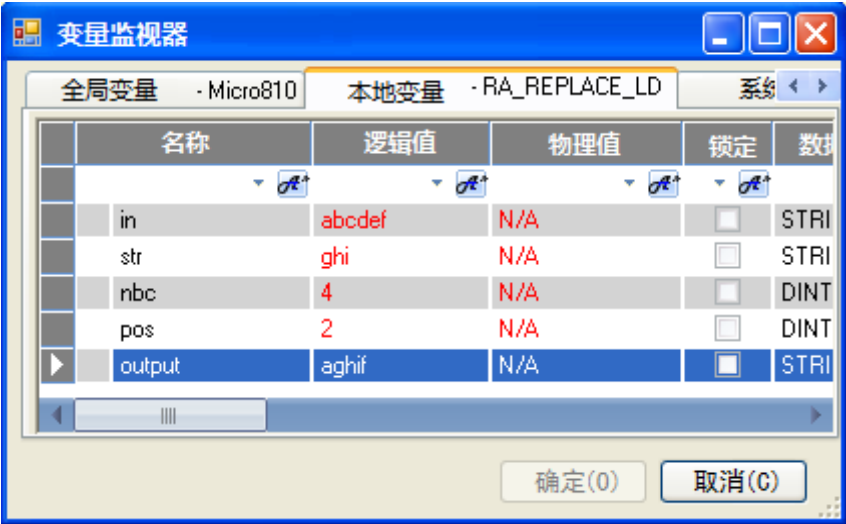
将字符串的一部分替换为一组新的字符。

(* 与之等效的 ST : *)

MyName := REPLACE ('Mr X JONES, 'Frank', 1, 4);

(* MyName 为“Mr Frank JONES” *)

结果



另请参见

[字符串操作指令](#) 参考页数 [617](#)

计时器指令

计时器指令用于根据时间控制操作。

功能块	说明
TOF 参考页数 642	关断延时计时
TON 参考页数 644	接通延时计时
TONOFF 参考页数 647	延迟打开 TRUE 梯级上的输出，然后延迟关闭 FALSE 梯级上的输出
TP 参考页数 650	脉冲计时
RTO 参考页数 652	保持时间。保存累加时间直到重置指令启动。
函数	说明
DOY 参考页数 654	如果实时时钟的值位于“年时间”设置范围内，则开启输出。
IDF 参考页数 656	计算时间差。
TOW 参考页数 658	如果实时时钟的值位于“周时间”设置范围内，则开启输出。

计时器指令配置

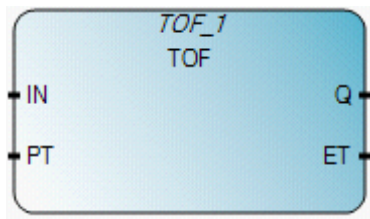
时间准确度指在处理器启用计时器指令的时刻与处理器完成计时间隔的时刻之间存在的时间差。

处理器使用计时器指令提供的以下信息：

- 计时器 - 计时器控制数据存储的计时器区域的地址。
- 时间基准 - 确定计时器的工作方式。
- 预设 - 指定在处理器设置 Done 位前，计时器必须达到的值。
- 累加值 - 该指令统计的时间增量数。如已启用，则计时器将连续更新该值。

TOF

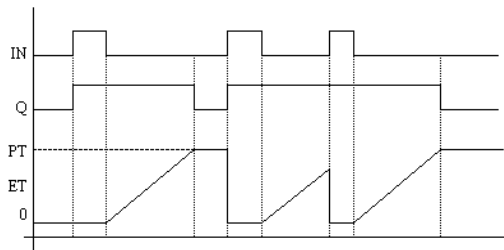
TOF 将内部计时器增加至指定值。



参数

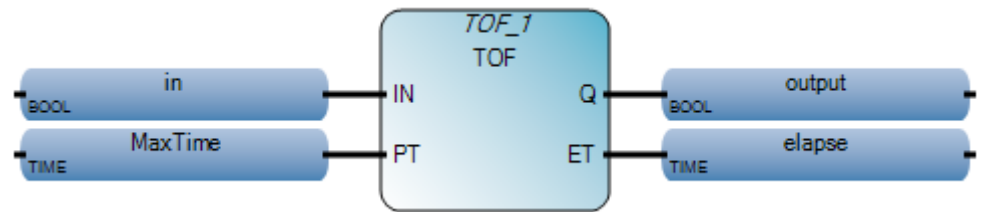
参数	参数类型	数据类型	说明
IN	输入	BOOL	如果是下降沿，内部计时器开始递增。 如果是上升沿，停止并重置内部计时器。
PT	输入	TIME	最大编程时间。 请参见 Time 数据类型。
Q	输出	BOOL	如果为 TRUE：总时间未过。
ET	输出	TIME	当前已过去的时间。值的可能范围从 0ms 到 1193h2m47s294ms。 提示： 如果您对本块使用 EN 参数，则计时器在 EN 设为 TRUE 时开始递增，即使将 EN 设为 FALSE 也继续递增。 请参见 Time 数据类型。

TOF 时间图

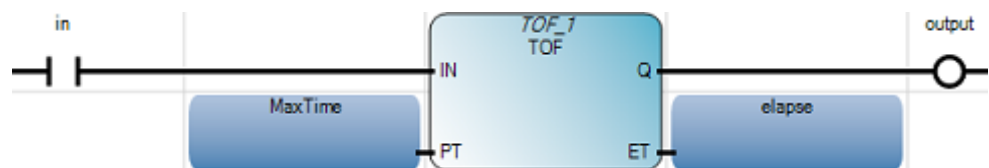


TOF 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1  MaxTime := T#3s;
2  TOF_1(in, MaxTime);
3  output := TOF_1.Q;
4  elapsed := TOF_1.ET;

```

```

TOF_1(
void TOF_1(BOOL IN, TIME PT)
Type : TOF, Off-delay timing

```

结果



另请参见

[计时器指令配置](#) 参考页数 [641](#)

[计时器指令](#) 参考页数 [641](#)

时间数据类型

TON

TON 将内部计时器增加至指定值。



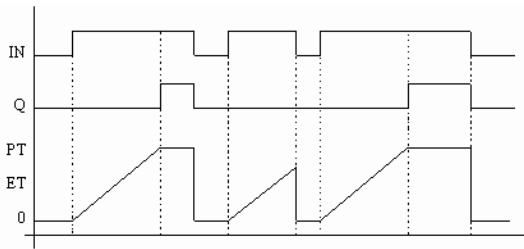
TON 操作

- 不要使用跳转跳过梯形图 (LD) 中的 TON 功能块。如果您的确执行了此操作，则 TON 计时器将在延时时间之后继续。
- 例如：梯级 1 包含一个跳转；梯级 2 包含一个 TON 功能块，延时时间为 10 秒；启用从梯级 1 到梯级 3 的跳转；在 30 秒后禁用跳转；延时时间将为 30 秒，而非延时时间中定义的 10 秒。

参数

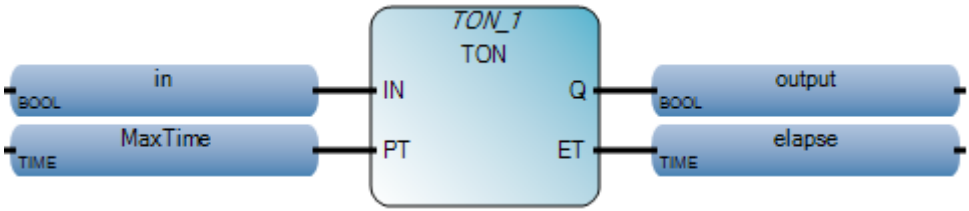
参数	参数类型	数据类型	说明
IN	输入	BOOL	如果是上升沿，内部计时器开始递增。 如果是下降沿，停止并重置内部计时器。
PT	输入	TIME	最大编程时间。 请参见 Time 数据类型。
Q	输出	BOOL	如果为 TRUE，则设定的时间已过。
ET	输出	TIME	当前已过去的时间。值的可能范围从 0ms 到 1193h2m47s294ms。 提示： 如果您对本块使用 EN 参数，则计时器在 EN 设为 TRUE 时开始递增，即使将 EN 设为 FALSE 也继续递增。 请参见 Time 数据类型。

TON 时间图

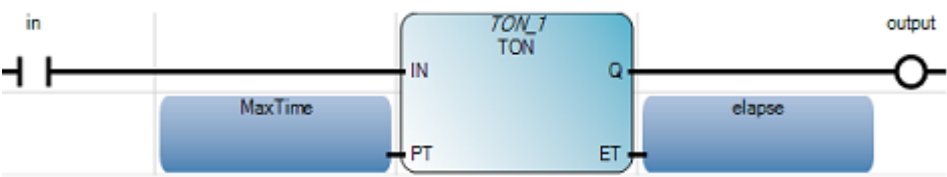


TON 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```
1 MaxTime := T#3s;  
2 TON_1(in, MaxTime);  
3 output := TON_1.Q;  
4 elapsed := TON_1.ET;
```

TON_1 {

void TON_1(BOOL IN, TIME PT)
Type : TON, On-delay timing

结果

Variable Monitoring					
全局变量 - Micro810		本地变量 - UntitledST		系统变量	
	名称	逻辑值	物理值	锁定	数
	in	☒	N/A	☐	BOC
	MaxTime	T#3s	N/A	☐	TIM
	output	☐	N/A	☐	BOC
	elapsed	T#1s493ms	N/A	☐	TIM
+	TON_1	...	...	☐	TON

确定(O) 取消(C)

另请参见

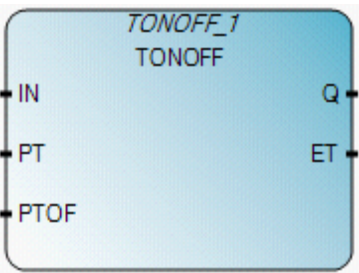
[计时器指令配置](#) 参考页数 [641](#)

[计时器指令](#) 参考页数 [641](#)

时间数据类型

TONOFF

TONOFF 首先会延迟打开 TRUE 梯级上的输出，然后会延迟关闭 FALSE 梯级上的输出。

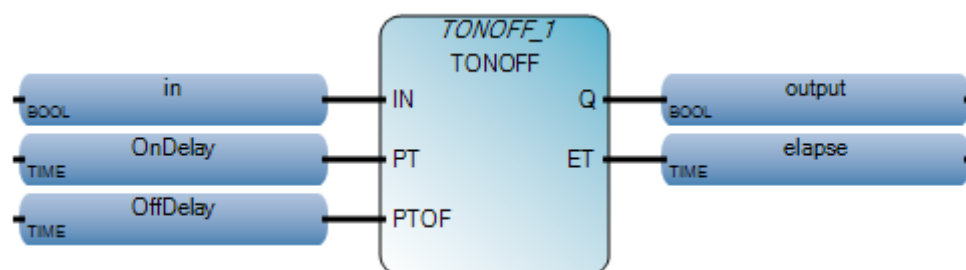


参数

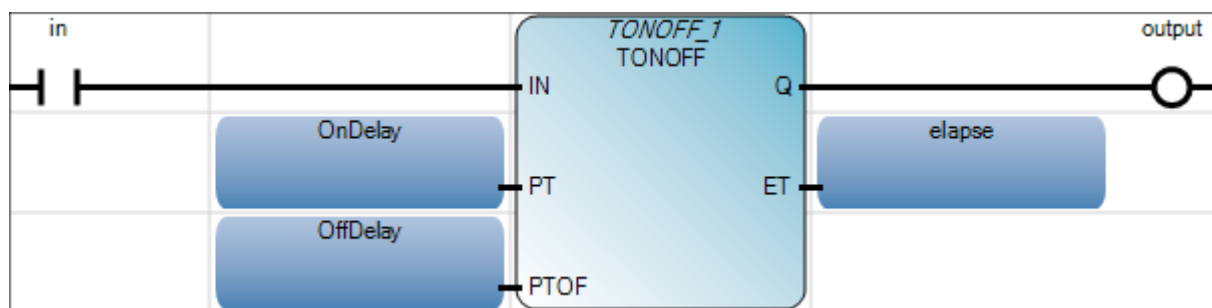
参数	参数类型	数据类型	说明
IN	输入	BOOL	如果上升沿 (IN) 从 0 变为 1，则接通延时计时器会启动。 如果设定的接通延时时间已过，且下降沿 (IN) 从 1 变为 0，则关断延时计时器会启动并复位已用时间 (ET)。 如果设定的接通延时时间已过，且下降沿 (IN) 从 1 变为 0，则关断延时计时器会启动。如果设定的接通延时时间未过，且上升沿 (IN) 从 0 变为 1，则接通延时计时器会启动。
PT	输入	TIME	接通延时时间设置。 请参见 Time 数据类型。
PTOF	输入	TIME	关断延时时间设置。 请参见 Time 数据类型。
Q	输出	BOOL	如果为 TRUE，则设定的接通延时时间已过，而设定的关断延时时间未过。
ET	输出	TIME	当前已过去的时间。值的可能范围从 0ms 到 1193h2m47s294ms。 如果设定的接通延时时间已过，且关断延时计时器未启动，则已过时间 (ET) 仍为接通延时 (PT) 值。如果设定的关断延时时间已过，且关断延时计时器未启动，则上升沿再次发生之前，已过时间 (ET) 仍为关断延时 (PTOF) 值。 提示：如果您对本块使用 EN 参数，则计时器在 EN 设为 TRUE 时开始递增，即使将 EN 设为 FALSE 也继续递增。 请参见 Time 数据类型。

TONOFF 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1 OnDelay := T#3s;
2 OffDelay := T#5s;
3 TONOFF_1(in, OnDelay, OffDelay);
4 output := TONOFF_1.Q;
5 elapsed := TONOFF_1.ET;

```

TONOFF_1 (

void **TONOFF_1**(BOOL IN, TIME PT, TIME PTOF)
Type : TONOFF, Delay an output-on(true), then delay an output-off(false).

结果



另请参见

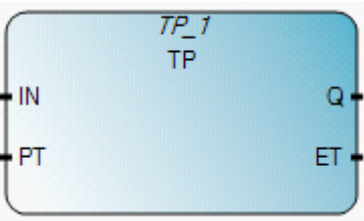
[计时器指令配置](#) 参考页数 [641](#)

[计时器指令](#) 参考页数 [641](#)

时间数据类型

TP

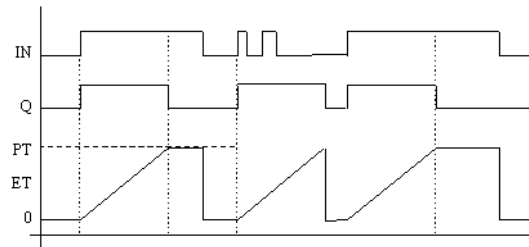
在上升沿时，TP 将内部计时器增加至指定值。如果计时器结束，请重置内部计时器。



参数

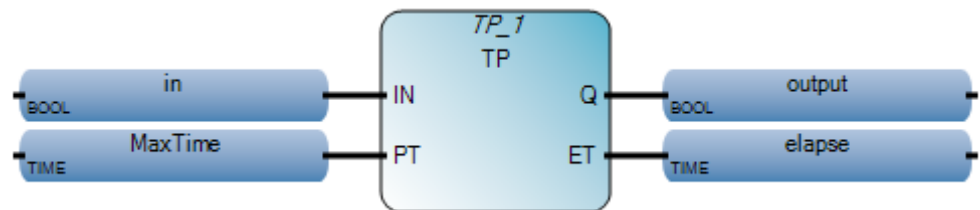
参数	参数类型	数据类型	说明
IN	输入	BOOL	如果是上升沿，内部计时器开始递增（如果尚未递增）。 如果为 FALSE 且仅在计时器结束时，重置内部计时器。 计数期间对 IN 的任何更改都不生效。
PT	输入	TIME	最大编程时间。 请参见 Time 数据类型。
Q	输出	BOOL	如果 TRUE，计时器正在计时。
ET	输出	TIME	当前已过去的时间。值的可能范围从 0ms 到 1193h2m47s294ms。 提示：如果您对本块使用 EN 参数，则计时器在 EN 设为 TRUE 时开始递增，即使将 EN 设为 FALSE 也继续递增。 请参见 Time 数据类型。

TP 时间图

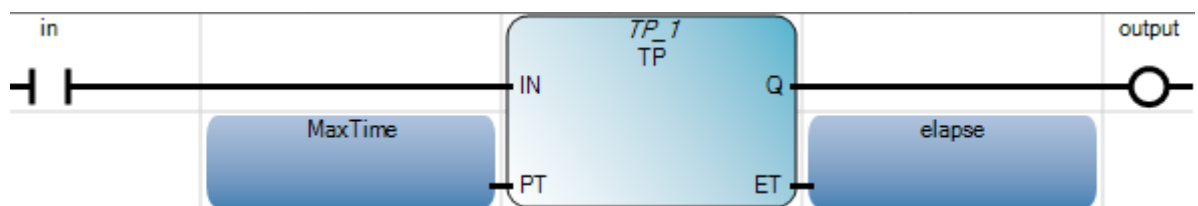


TP 功能块语言示例

功能块图 (FBD)



梯形图 (LD)



结构化文本 (ST)

```

1 | MaxTime := T#3s;
2 | TP_1(in, MaxTime);
3 | output := TP_1.Q;
4 | elapsed := TP_1.ET;

```

```

TP_1 {
  void TP_1(BOOL IN, TIME PT)
  Type : TP, Pulse timing
}

```

结果



另请参见

[计时器指令配置](#) 参考页数 [641](#)

[计时器指令](#) 参考页数 [641](#)

时间数据类型

RTO

当输入激活时，RTO 增加内部计时器，但当输入变为不活动状态，不会重置内部计时器。

如果使用 Micro810 或 Micro820 控制器，RTO 内部计时器不会默认在重新上电过程中保持不变。若要保持内部计时器不变，请将保留的配置参数设置为真。

如果使用 Micro830 或 Micro850 控制器，RTO 内部计时器会在重新上电过程中保留不变。

参数

参数	参数类型	数据类型	说明
IN	输入	BOOL	如果是上升沿，内部计时器开始递增。 如果是下降沿，停止并不要重置内部计时器。

参数	参数类型	数据类型	说明
RST	输入	BOOL	在上升沿重置内部计时器。
PT	输入	TIME	最大编程时间。 请参见 Time 数据类型。
Q	输出	BOOL	如果为 TRUE，则设定的时间已过。
ET	输出	TIME	已过去的时间。值的可能范围从 0ms 到 1193h2m47s294ms。 请参见 Time 数据类型。

RTO 时间图

RTO 功能块语言示例

功能块图 (FBD)

梯形图 (LD)

结构化文本 (ST)

另请参见

[计时器指令配置](#) 参考页数 [641](#)

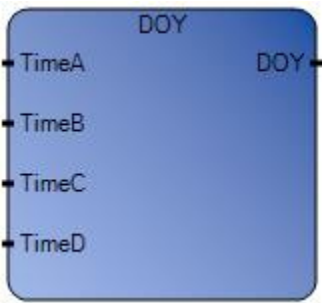
[计时器指令](#) 参考页数 [641](#)

时间数据类型

变量参数

DOY

DOY 函数有四个通道输入 ;如果实时时钟 (RTC) 的值在四个通道任意之一的“年时间”设置范围内 , 则会开启输出。如果 RTC 不存在 , 则输出始终为关闭。



DOY 操作

在“DOYDATA 数据类型”表中指定的有效范围内配置任意 Time 输入参数。如果 TimeX.Enable 设为 TRUE 且 RTC 存在并已启用 , 则无效的值会导致控制器出现故障。

参数

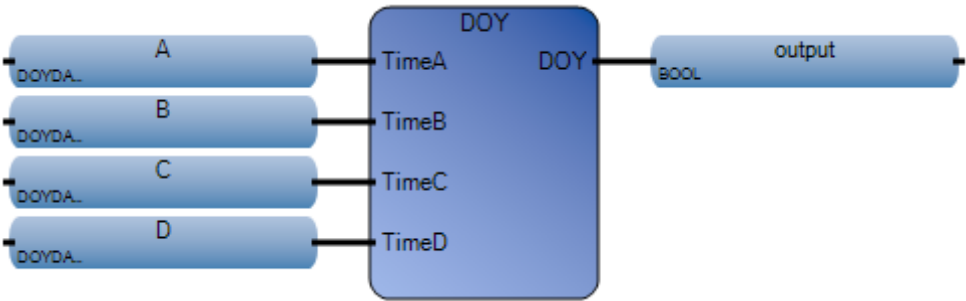
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时 , 执行该函数。 当 EN = FALSE 时 , 不执行该函数。
TimeA	输入	DOYDATA	通道 A 的年时间设置。 请参见 DOYDATA 数据类型 参考页数 656。
TimeB	输入	DOYDATA	通道 B 的年时间设置。 请参见 DOYDATA 数据类型 参考页数 656。
TimeC	输入	DOYDATA	通道 C 的年时间设置。 请参见 DOYDATA 数据类型 参考页数 656。
TimeD	输入	DOYDATA	通道 D 的年时间设置。 请参见 DOYDATA 数据类型 参考页数 656。
DOY	输出	BOOL	如果为 TRUE , 则实时时钟的值在四个通道任意之一的“年时间”设置范围内。

另请参见

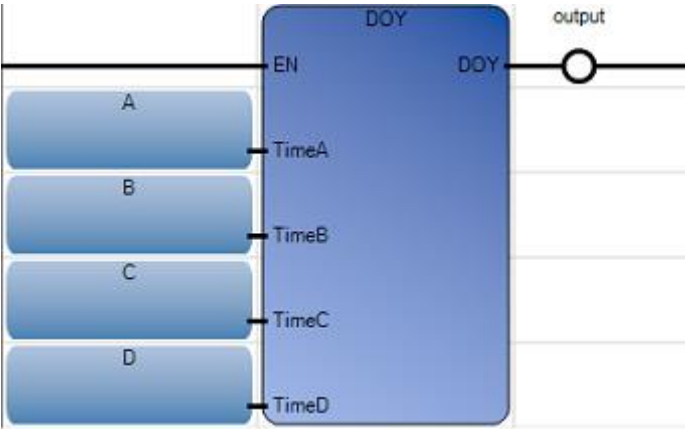
[DOY](#) 参考页数 [654](#)

DOY 指令语言示例

功能块图



梯形图



结构化文本

```
1 output := DOY(A, B, C, D);
```

DOY (

BOOL **DOY**(DOYDATA TimeA, DOYDATA TimeB, DOYDATA TimeC, DOYDATA TimeD)
Turn on output when real-time clock value is within year range.

(* 与之等效的 ST : *)

TESTOUTPUT := DOY(TIMEA1, TIMEB1, TIMEC1, TIMED1);

另请参见

[计时器指令配置](#) 参考页数 [641](#)

[计时器指令](#) 参考页数 [641](#)

DOYDATA 数据类型

下表描述了 DOYDATA 数据类型。

参数	数据类型	说明
启用	BOOL	TRUE：启用；FALSE：禁用
YearlyCenturial	BOOL	计时器类型（0：年计时器；1：世纪计时器）。
YearOn	UINT	年开始值（必须位于集合 [2000...2098] 内）。
MonthOn	USINT	月开始值（必须位于集合 [1...12] 内）。
DayOn	USINT	日期开始值（必须位于“MonthOn”值确定的集合 [1...31] 内）。
YearOff	UINT	年结束值（必须位于集合 [2000...2098] 内）。
MonthOff	USINT	月结束值（必须位于集合 [1...12] 内）。
DayOff	USINT	日期结束值（必须位于“MonthOn”值确定的集合 [1...31] 内）。

TDF

TDF 计算时间差。

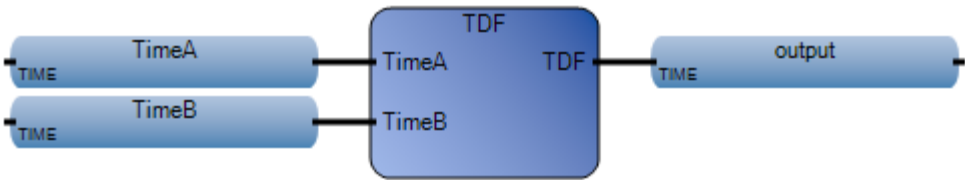


参数

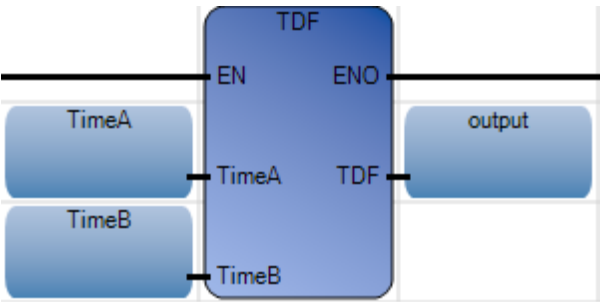
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行当前计算。 当 EN = FALSE 时，不执行计算。
TimeA	输入	TIME	时间差计算的开始时间。
TimeB	输入	TIME	时间差计算的结束时间。
ENO	输出	BOOL	启用输出。
TDF 或 Q	输出	TIME	两个时间输入的时间差。 TDF 为名称或 PIN ID Q 为 PIN ID

TDF 函数语言示例

功能块图



梯形图



结构化文本

```
1 TimeA := T#10s;  
2 TimeB := T#5s;  
3 output := TDF(TimeA, TimeB);
```

TDF {

TIME **TDF**(TIME TimeA, TIME TimeB)
Compute time difference.

(* 与之等效的 ST : *)

TESTTIMEDIFF := TDF(TESTTIME1, TESTTIME2) ;

结果



另请参见

[计时器指令配置](#) 参考页数 [641](#)

[计时器指令](#) 参考页数 [641](#)

时间数据类型

TOW

TOW 函数有四个通道输入 ;如果实时时钟 (RTC) 的值在四个通道任意之一的“周时间”设置范围内，则会开启输出。如果 RTC 不存在，则输出始终为关闭。



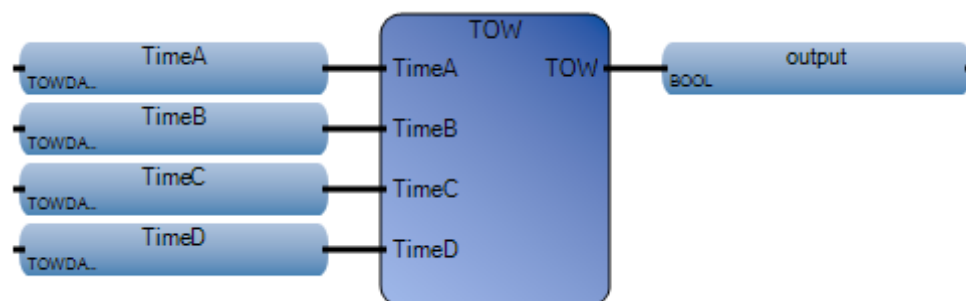
提示： 确保您通过“TOWDATA 数据类型”表中指定的有效范围配置任意 TimeX 输入参数。如果 TimeX.Enable 设置为 TRUE 且 RTC 存在并已启用，则无效的值会导致控制器出现故障。

参数

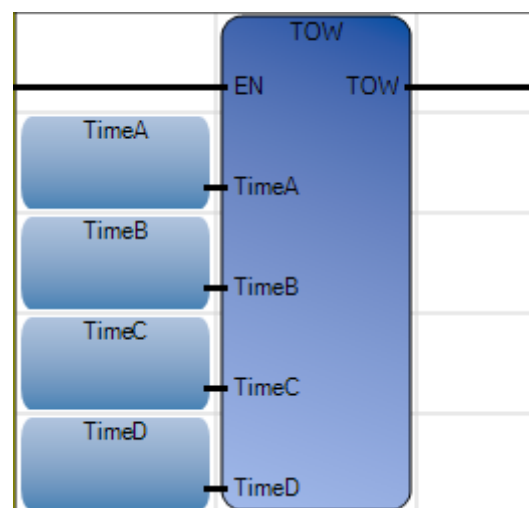
参数	参数类型	数据类型	说明
EN	输入	BOOL	函数启用。 当 EN = TRUE 时，执行该函数。 当 EN = FALSE 时，不执行该函数。
TimeA	输入	TOWDATA	通道 A 的“日时间”设置。 请参见 TOWDATA 数据类型 参考页数 662。
TimeB	输入	TOWDATA	通道 B 的“日时间”设置。 请参见 TOWDATA 数据类型 参考页数 662。
TimeC	输入	TOWDATA	通道 C 的“日时间”设置。 请参见 TOWDATA 数据类型 参考页数 662。
TimeD	输入	TOWDATA	通道 D 的“日时间”设置。 请参见 TOWDATA 数据类型 参考页数 662。
TOW	输出	BOOL	如果为 TRUE，则实时时钟的值在四个通道任意之一的“日时间”设置范围内。

TOW 函数语言示例

功能块图



梯形图



结构化文本

```
1 output := TOW(TimeA, TimeB, TimeC, TimeD);
```

```
TOW{
  BOOL TOW(TOWDATA TimeA, TOWDATA TimeB, TOWDATA TimeC, TOWDATA TimeD)
  Turn on output when real-time clock value is within week range.
```

(* 与之等效的 ST : *)

TESTOUTPUT := TOW(TIMEA, TIMEB, TIMEC, TIMED);

结果

	名称	逻辑值	物理值	锁定	数据类型
-	TimeA	...	...	☐	TOWDAT, ▾
	TimeA.Enable	☒	N/A	☐	BOOL
	TimeA.Daily/Weekly	☒	N/A	☐	BOOL
	TimeA.DayOn	1	N/A	☐	USINT
	TimeA.HourOn	10	N/A	☐	USINT
	TimeA.MinOn	20	N/A	☐	USINT
	TimeA.DayOff	2	N/A	☐	USINT
	TimeA.HourOff	15	N/A	☐	USINT
	TimeA.MinOff	30	N/A	☐	USINT
-	TimeB	...	...	☐	TOWDAT, ▾
	TimeB.Enable	☒	N/A	☐	BOOL
	TimeB.Daily/Weekly	☒	N/A	☐	BOOL
	TimeB.DayOn	2	N/A	☐	USINT
	TimeB.HourOn	15	N/A	☐	USINT
	TimeB.MinOn	20	N/A	☐	USINT
	TimeB.DayOff	3	N/A	☐	USINT
	TimeB.HourOff	10	N/A	☐	USINT
	TimeB.MinOff	30	N/A	☐	USINT
-	TimeC	...	...	☐	TOWDAT, ▾
	TimeC.Enable	☒	N/A	☐	BOOL
	TimeC.Daily/Weekly	☒	N/A	☐	BOOL
	TimeC.DayOn	3	N/A	☐	USINT
	TimeC.HourOn	20	N/A	☐	USINT
	TimeC.MinOn	10	N/A	☐	USINT
	TimeC.DayOff	4	N/A	☐	USINT
	TimeC.HourOff	25	N/A	☐	USINT
	TimeC.MinOff	55	N/A	☐	USINT
-	TimeD	...	...	☐	TOWDAT, ▾
	TimeD.Enable	☒	N/A	☐	BOOL
	TimeD.Daily/Weekly	☒	N/A	☐	BOOL
	TimeD.DayOn	5	N/A	☐	USINT
	TimeD.HourOn	10	N/A	☐	USINT
	TimeD.MinOn	15	N/A	☐	USINT
	TimeD.DayOff	6	N/A	☐	USINT
	TimeD.HourOff	35	N/A	☐	USINT
	TimeD.MinOff	40	N/A	☐	USINT
▶	output	☒	N/A	☐	BOOL ▾

另请参见

[计时器指令配置](#) 参考页数 [641](#)

[计时器指令](#) 参考页数 [641](#)

TOWDATA 数据类型

下表描述了 TOWDATA 数据类型：

参数	数据类型	说明
启用	BOOL	TRUE：启用；FALSE：禁用。
DailyWeekly	BOOL	计时器类型（0：日计时器；1：周计时器）。
DayOn	USINT	星期开始值（必须位于集合 [0...6] 内）。
HourOn	USINT	小时开始值（必须位于集合 [0...23] 内）。
MinOn	USINT	分钟开始值（必须位于集合 [0...59] 内）。
DayOff	USINT	星期结束值（必须位于集合 [0...6] 内）。
HourOff	USINT	小时结束值（必须位于集合 [0...23] 内）。
MinOff	USINT	分钟结束值（必须位于集合 [0...59] 内）。

另请参见

[TOW](#) 参考页数 [658](#)

符号

- 运算符, 110
* 运算符, 99
/ 运算符, 91
+ 运算符, 79
< 运算符, 280
<= 运算符, 281
<> 运算符, 282
= 运算符, 275
> 运算符, 277
>= 运算符, 279

数字

1Gain 运算符, 98

字母

ABL 功能块, 119
ABS 函数, 74
ACB 功能块, 121
ACL 功能块, 123
ACOS 函数, 76
ACOS_LREAL 函数, 77
AHL 功能块, 124
AND_MASK 函数, 141
ANY_TO_BOOL 运算符, 294
ANY_TO_BYTE 运算符, 295
ANY_TO_DATE 运算符, 296
ANY_TO_DINT 运算符, 297
ANY_TO_DWORD 运算符, 298

ANY_TO_INT 运算符, 299
ANY_TO_LINT 运算符, 300
ANY_TO_LREAL 运算符, 301
ANY_TO_LWORD 运算符, 302
ANY_TO_REAL 运算符, 303
ANY_TO_SINT 运算符, 304
ANY_TO_STRING 运算符, 305
ANY_TO_TIME 运算符, 306
ANY_TO_UDINT 运算符, 307
ANY_TO_UINT 运算符, 309
ANY_TO_ULINT 运算符, 310
ANY_TO_USINT 运算符, 310
ANY_TO_WORD 运算符, 312
ARD 功能块, 127
ARL 功能块, 129
ASCII 函数, 613
ASIN 函数, 80
ASIN_LREAL 函数, 82
ATAN 函数, 84
ATAN_LREAL 函数, 86
AVERAGE 功能块, 313
AWA 功能块, 131
AWT 功能块, 133
CHAR 函数, 616
COP 功能块, 316
COS 函数, 87
COS_LREAL 函数, 89
CTD 功能块, 285
CTU 功能块, 287
CTUD 功能块, 289
DERIVATE 功能块, 545

DOY 函数, 650

EXPT 函数, 92

F_TRIG 功能块, 159

FBD 中的重置线圈, 54

FBD (功能块图)

- 触点, 使用情况和可用类型, 54

- 反向接触, 57

- 反向线圈, 51

- 返回, 插入, 59

- 脉冲上升沿 (正值) 触点, 57

- 设置线圈, 53

- 梯级, 插入, 39

- 跳转到标签, 插入, 60

- 线圈, 使用情况和可用类型, 48

- 直接接触, 56

- 直接线圈, 50

- 指令块, 插入, 44

- 重设线圈, 54

FIND 函数, 620

HSC 功能块, 326

HSC_SET_STS 功能块, 349

HYSTER 功能块, 547

IIM 功能块, 396

INSERT 函数, 622

INTEGRAL 功能块, 549

IOM 功能块, 399

IPIDCONTROLLER 功能块, 579

KEY_READ 功能块, 402

LCD 函数, 380

LD 图的平行分支, 42

LD 语言

- 键盘快捷键, 64

LD (梯形图)

- 触点, 使用情况和可用类型, 54

- 反向接触, 57

- 反向线圈, 51

- 返回语句, 59

- 分支, 42

- 跨线, 60

- 脉冲上升沿 (正值) 触点, 57

- 脉冲上升沿 (正值) 线圈, 52

- 脉冲下降沿 (负值) 触点, 58

- 脉冲下降沿 (负值) 线圈, 51

- 设置线圈, 53

- 梯级, 插入, 39

- 线圈, 使用情况和可用类型, 48

- 直接接触, 56

- 直接线圈, 50

- 指令块, 44

- 重设线圈, 54

LEFT 函数, 624

LIM_ALARM 功能块, 69

LIMIT 函数, 568

LOG 函数, 94

MAX 函数, 322

MC_AbortTrigger 功能块, 464

MC_Halt 功能块, 468

MC_Home 功能块, 472

MC_MoveAbsolute 功能块, 476

MC_MoveRelative 功能块, 482

MC_MoveVelocity 功能块, 487

MC_Power 功能块, 492

MC_ReadAxisError 功能块, 500

MC_ReadBoolParameter 功能块, 506

- MC_ReadParameter 功能块, 509
MC_ReadStatus 功能块, 513
MC_Reset 功能块, 519
MC_SetPosition 功能块, 522
MC_Stop 功能块, 526
MC_TouchProbe 功能块, 530
MC_WriteBoolParameter 功能块, 535
MC_WriteParameter 功能块, 539
MID 函数, 626
MIN 函数, 320
MLen 函数, 629
MMINFO 功能块, 409
MOD 函数, 96
MOV 运算符, 98
MSG_CIPGENERIC 功能块, 183
MSG_CIPSYMBOLIC 功能块, 195
MSG_MODBUS 功能块, 209
MSG_MODBUS2 功能块, 217
MUX4B 函数, 179
MUX8B 函数, 174
Neg 运算符, 100
NOT_MASK 函数, 143
OR_MASK 函数, 145
PLUGIN_INFO 功能块, 412
PLUGIN_READ 功能块, 415
PLUGIN_RESET 功能块, 417
PLUGIN_WRITE 功能块, 419
POW 函数, 100
R_TRIG 功能块, 161
RAND 函数, 103
REPLACE 函数, 633
RHC 函数, 390
RIGHT 函数, 630
ROL 函数, 147
ROR 函数, 149
RPC 函数, 392
RS 功能块, 164
RTC_READ 功能块, 425
RTC_SET 功能块, 428
RTO 功能块, 648
SCALER 功能块, 560
SHL 函数, 151
SHR 函数, 153
SIN 函数, 105
SIN_LREAL 函数, 107
SQRT 函数, 109
SR 功能块, 168
STACKINT 功能块, 563
STIS 函数, 437
SUS 功能块, 573
SYS_INFO 功能块, 430
TAN 函数, 111
TAN_LREAL 函数, 113
TDF 函数, 652
TND 函数, 566
TOF 功能块, 638
TON 功能块, 640
TONOFF 功能块, 643
TOW 函数, 654
TP 功能块, 646
TRIMPOT_READ 功能块, 434
TRUNC 函数, 115
TTABLE 函数, 170
UIC 函数, 439

UID 函数, 441

UIE 函数, 443

UIF 函数, 445

XOR_MASK 函数, 155

B

比较运算

不等于运算符, 282

大于或等于运算符, 279

大于运算符, 277

等于运算符, 275

小于或等于运算符, 281

小于运算符, 280

标签

梯级, 插入, 39

不等于运算符, 282

布尔

MUX4B 函数, 179

MUX8B 函数, 174

TTABLE 函数, 170

布尔运算

F_TRIG 功能块, 159

R_TRIG 功能块, 161

RS 功能块, 164

SR 功能块, 168

非运算符, 167

或运算符, 165

异或运算符, 167

与运算符, 166

C

参数

使用, 44

用于功能块, 18

在 LD 中使用, 44

插入

返回 (FBD 单元) , 59

功能块图中的指令块, 44

梯级 (FBD 单元) , 39

梯级标签, 39

跳转到标签 (FBD 单元) , 60

程序控制

SUS 功能块, 573

TND 函数, 566

D

大于或等于运算符, 279

大于运算符, 277

等于运算符, 275

调用

功能块, 18

E

二进制运算

AND_MASK 函数, 141

NOT_MASK 函数, 143

OR_MASK 函数, 145

ROL 函数, 147

ROR 函数, 149

SHL 函数, 151

SHR 函数, 153

XOR_MASK 函数, 155

F

反向

FBD 中的触点, 57

FBD 中的线圈, 51

接触, 57

线圈, 51

返回符号

插入, FBD 单元, 59

返回语句

插入, 59

用于 LD 图, 59

非运算符, 167

分支

插入, 42

用于 LD, 42

G

高级控制块

SCALER, 560

功能块

ABL, 119

ACB, 121

ACL, 123

AHL, 124

ARD, 127

ARL, 129

AVERAGE, 313

AWA, 131

AWT, 133

COP, 316

CTD, 285

CTU, 287

CTUD, 289

DERIVATE, 545

F_TRIG, 159

HSC, 326

HSC_SET_STS, 349

HYSTER, 547

IIM, 396

INTEGRAL, 549

IOM, 399

IPIDCONTROLLER, 579

KEY_READ, 402

LIM_ALARM, 69

MC_AbortTrigger, 464

MC_Halt, 468

MC_Home, 472

MC_MoveAbsolute, 476

MC_MoveRelative, 482

MC_MoveVelocity, 487

MC_Power, 492

MC_ReadAxisError, 500

MC_ReadBoolParameter, 506

MC_ReadParameter, 509

MC_ReadStatus, 513

MC_Reset, 519

MC_SetPosition, 522

MC_Stop, 526

MC_TouchProbe, 530

MC_WriteBoolParameter, 535

MC_WriteParameter, 539

MMINFO, 409

MSG_CIPGENERIC, 183

MSG_CIPSYMBOLIC, 195

MSG_MODBUS, 209

MSG_MODBUS2, 217

PLUGIN_INFO, 412

PLUGIN_READ, 415

PLUGIN_RESET, 417

PLUGIN_WRITE, 419

R_TRIG, 161

RS, 164

RTC_READ, 425

RTC_SET, 428

RTO, 648

SCALER, 560

SR, 168

STACKINT, 563

SUS, 573

SYS_INFO, 430
TOF, 638
TON, 640
TONOFF, 643
TP, 646
TRIMPOT_READ, 434
命名和参数, 18
调用, 18
在功能块图中插入, 44

H

函数

ABS, 74
ACOS, 76
ACOS_LREAL, 77
AND_MASK, 141
ASCII, 613
ASIN, 80
ASIN_LREAL, 82
ATAN, 84
ATAN_LREAL, 86
CHAR, 616
COS, 87
COS_LREAL, 89
DELETE, 617
DOY, 650
EXPT, 92
FIND, 620
INSERT, 622
LCD, 380
LEFT, 624
LIMIT, 568
LOG, 94
MAX, 322
MID, 626
MIN, 320
MLen, 629
MOD, 96
MUX4B, 179
MUX8B, 174
NOT_MASK, 143
OR_MASK, 145
POW, 100
RAND, 103

REPLACE, 633
RHC, 390
RIGHT, 630
ROL, 147
ROR, 149
RPC, 392
SHL, 151
SHR, 153
SIN, 105
SIN_LREAL, 107
SQRT, 109
STIS, 437
TAN, 111
TAN_LREAL, 113
TDF, 652
TND, 566
TOW, 654
TRUNC, 115
TTABLE, 170
UIC, 439
UID, 441
UIE, 443
UIF, 445
XOR_MASK, 155
或运算符, 165

J

计数器

CTD 功能块, 285
CTU 功能块, 287
CTUD 功能块, 289

键盘快捷键

LD 语言, 64

接触

FBD 中的反向, 57
FBD 中的脉冲上升沿 (正值), 57
FBD 中的使用情况和可用类型, 54
FBD 中的正向, 56
并联, 插入, 54

插入, 54

反向, 57

更改类型, 54

脉冲上升沿 (正值), 57

脉冲下降沿 (正值), 58

使用情况和可用类型, 54

直接, 56

警报

LIM_ALARM 功能块, 69

K

跨线

功能块图, 插入, 60

梯级, 插入, 39

梯形图, 插入, 60

用于 LD 图, 60

L

流程控制

DERIVATE 功能块, 545

HYSTER 功能块, 547

INTEGRAL 功能块, 549

IPIDCONTROLLER 功能块, 579

LIMIT 函数, 568

STACKINT 功能块, 563

M

脉冲上升沿 (正值)

FBD 中的触点, 57

接触, 57

线圈, 52

脉冲下降沿 (负值)

接触, 58

线圈, 51

命名规范

功能块, 18

模具

可用于 FBD 的线圈类型, 48

可用于 LD 图的触点元素, 54

可用于 LD 图的线圈类型, 48

梯级, 插入到 LD 容器中, 39

S

算术

ABS 函数, 74

ACOS 函数, 76

ACOS_LREAL 函数, 77

ASIN 函数, 80

ASIN_LREAL 函数, 82

ATAN 函数, 84

ATAN_LREAL 函数, 86

COS 函数, 87

COS_LREAL 函数, 89

EXPT 函数, 92

LOG 函数, 94

MOD 函数, 96

MOV 运算符, 98

Neg 运算符, 100

POW 函数, 100

RAND 函数, 103

SIN 函数, 105

SIN_LREAL 指令, 107

SQRT 函数, 109

TAN 函数, 111

TAN_LREAL 函数, 113

TRUNC 函数, 115

相乘运算符, 99

相除运算符, 91

相加运算符, 79

相减运算符, 110

算术运算

1Gain 运算符, 98

时间

DOY 函数, 650

RTO 功能块, 648

TDF 函数, 652

TOF 功能块, 638

TON 功能块, 640

TONOFF 功能块, 643

TOW 函数, 654

TP 功能块, 646

输入/输出

HSC 功能块, 326

HSC_SET_STS 功能块, 349

IIM 功能块, 396

IOM 功能块, 399

KEY_READ 功能块, 402

LCD 函数, 380

MMINFO 功能块, 409

PLUGIN_INFO 功能块, 412

PLUGIN_READ 功能块, 415

PLUGIN_RESET 功能块, 417

PLUGIN_WRITE 功能块, 419

RHC 函数, 390

RPC 函数, 392

RTC_READ 功能块, 425

RTC_SET 功能块, 428

SYS_INFO 功能块, 430

TRIMPOT_READ 功能块, 434

数据操作

AVERAGE 功能块, 313

COP 功能块, 316

MAX 函数, 322

MIN 函数, 320

数据转换

ANY_TO_BOOL 运算符, 294

ANY_TO_BYTE 运算符, 295

ANY_TO_DATE 运算符, 296

ANY_TO_DINT 运算符, 297

ANY_TO_DWORD 运算符, 298

ANY_TO_INT 运算符, 299

ANY_TO_LINT 运算符, 300

ANY_TO_LREAL 运算符, 301

ANY_TO_LWORD 运算符, 302

ANY_TO_REAL 运算符, 303

ANY_TO_SINT 运算符, 304

ANY_TO_STRING 运算符, 305

ANY_TO_TIME 运算符, 306

ANY_TO_UDINT 运算符, 307

ANY_TO_UINT 运算符, 309

ANY_TO_ULINT 运算符, 310

ANY_TO_USINT 运算符, 310

ANY_TO_WORD 运算符, 312

T

梯级

- 标签, 插入, 39
- 插入, 39
- 功能块图, 插入, 39

梯形图 (LD)

- 语言, 35
- 指令, 35

通信

- ABL 功能块, 119
- ACB 功能块, 121
- ACL 功能块, 123
- AHL 功能块, 124
- ARD 功能块, 127
- ARL 功能块, 129
- AWA 功能块, 131
- AWT 功能块, 133
- MSG_CIPGENERIC 功能块, 183
- MSG_CIPSYMBOLIC 功能块, 195
- MSG_MODBUS 功能块, 209
- MSG_MODBUS2 功能块, 217

X

线圈

- FBD 的使用情况和可用类型, 48
- FBD 中的反向类型, 51
- FBD 中的正向类型, 50
- FBD 中的置位, 53
- FBD 中的重置, 54
- 并联, 插入, 48
- 并联, 在 FBD 中插入, 48

插入, 48

反向类型, 51

更改类型, 48

脉冲上升沿 (正值), 52

脉冲下降沿 (正值), 51

设置, 53

使用情况和可用类型, 48

在 FBD 更改类型, 48

在 FBD 中插入, 48

直接类型, 50

重设, 54

相乘运算符, 99

相除运算符, 91

相加运算符, 79

相减运算符, 110

小于或等于运算符, 281

小于运算符, 280

Y

以下项的指令块说明

- ABL, 119
- ACB, 121
- ACL, 123
- AHL, 124
- ARD, 127
- ARL, 129
- AVERAGE, 313
- AWA, 131
- AWT, 133
- COP, 316
- CTD, 285
- CTU, 287
- CTUD, 289
- DERIVATE, 545
- F_TRIG, 159
- HSC, 326
- HSC_SET_STS, 349

HYSTER, 547
IIM, 396
INTEGRAL, 549
IOM, 399
IPIDCONTROLLER, 579
KEY_READ, 402
LIM_ALARM, 69
MC_AbortTrigger, 464
MC_Halt, 468
MC_Home, 472
MC_MoveAbsolute, 476
MC_MoveRelative, 482
MC_MoveVelocity, 487
MC_Power, 492
MC_ReadAxisError, 500
MC_ReadBoolParameter, 506
MC_ReadParameter, 509
MC_ReadStatus, 513
MC_Reset, 519
MC_SetPosition, 522
MC_Stop, 526
MC_TouchProbe, 530
MC_WriteBoolParameter, 535
MC_WriteParameter, 539
MM_INFO, 409
MSG_CIPGENERIC, 183
MSG_CIPSYMBOLIC, 195
MSG_MODBUS, 209
MSG_MODBUS2, 217
PLUGIN_INFO, 412
PLUGIN_READ, 415
PLUGIN_RESET, 417
PLUGIN_WRITE, 419
R_TRIG, 161
RST, 164
RTC_READ, 425
RTC_SET, 428
RTO, 648
SCALER, 560
SRT, 168
STACKINT, 563
SUS, 573
SYS_INFO, 430
TOF, 638
TON, 640
TONOFF, 643
TP, 646

TRIMPOT_READ, 434

异或运算符, 167

与运算符, 166

语言

梯形图 (LD), 35

运动

MC_AbortTrigger 功能块, 464

MC_Halt 功能块, 468

MC_Home 功能块, 472

MC_MoveAbsolute 功能块, 476

MC_MoveRelative 功能块, 482

MC_MoveVelocity 功能块, 487

MC_Power 功能块, 492

MC_ReadAxisError 功能块, 500

MC_ReadBoolParameter 功能块, 506

MC_ReadParameter 功能块, 509

MC_ReadStatus 功能块, 513

MC_Reset 功能块, 519

MC_SetPosition 功能块, 522

MC_Stop 功能块, 526

MC_TouchProbe 功能块, 530

MC_WriteBoolParameter 功能块, 535

MC_WriteParameter 功能块, 539

运动控制功能块, 449

运算符

1Gain, 98

AND, 166

ANY_TO_BOOL, 294

ANY_TO_BYTE, 295

ANY_TO_DATE, 296

ANY_TO_DINT, 297

ANY_TO_DWORD, 298

ANY_TO_INT, 299

ANY_TO_LINT, 300

ANY_TO_LREAL, 301

ANY_TO_LWORD, 302
ANY_TO_REAL, 303
ANY_TO_SINT, 304
ANY_TO_STRING, 305
ANY_TO_TIME, 306
ANY_TO_UDINT, 307
ANY_TO_UINT, 309
ANY_TO_ULINT, 310
ANY_TO_USINT, 310
ANY_TO_WORD, 312
MOV, 98
OR, 165
XOR, 167
不等于, 282
大于, 277
大于或等于, 279
等于, 275
非, 167
取反, 100
相乘, 99
相除, 91
相加, 79
相减, 110
小于, 280
小于或等于, 281

Z

字符串操作

ASCII 函数, 613
CHAR 函数, 616
DELETE 函数, 617
FIND 函数, 620
INSERT 函数, 622
LEFT 函数, 624
MID 函数, 626

MLEN 函数, 629
REPLACE 函数, 633
RIGHT 函数, 630

直接

FBD 中的触点, 56
FBD 中的线圈, 50
接触, 56
线圈, 50

指令, 35, 44

指令块

EN 和 ENO 参数, 44
EN 输出, 44
ENO 输出, 44
并联, 插入, 44
插入, 44
设置类型, 44
用于 LD 图, 44
在功能块图中插入, 44

指令块说明, 183, 195, 217, 313, 316

中断

STIS 函数, 437
UIC 函数, 439
UID 函数, 441
UIE 函数, 443
UIF 函数, 445

重设

线圈, 54

Rockwell Automation 支持

网站上提供了罗克韦尔自动化在技术信息，以帮助您使用其产品。在 <http://www.rockwellautomation.com/support> 您可以找到技术和应用手册，实例代码及软件服务包的链接。您还可以在 <https://rockwellautomation.custhelp.com> 访问"支持中心"，进行软件的更新，支持聊天和论坛，技术信息，FAQs，注册产品更新通知。

此外，这里为安装、配置及排除故障提供了多样化的支持程序。更多信息，请联系您当地的经销商或 Rockwell Automation 代表，也可以访问 <http://www.rockwellautomation.com/support/>。

安装帮助

如果您在安装后的 24 小时内遇到问题，请查阅本手册中的相关信息。您也可以联系客户支持获取有关使产品恢复正常运行的初步帮助。

美国或加拿大	1.440.646.3434
美国或加拿大以外地区	使用 http://www.rockwellautomation.com/support/americas/phone_en.html 上的“全球地点”或联系您当地的 "Rockwell Automation" 代理。

新产品退货

所有产品出厂前，Rockwell Automation 都会进行相关测试，以确保产品能够全面运转。但是，如果您的产品不能正常工作，需要退货，请遵循下列步骤。

美国	请联系您的经销商。您必须向经销商提供客户支持案例号码（可拨打以上电话号码获取）以完成退货流程。
美国以外地区	请联系您当地的 Rockwell Automation 代表以了解退货流程。

Rockwell Otomasyon Ticaret A.Ş., Kar Plaza İş Merkezi E Blok Kat:6 34752 İçerenköy, İstanbul, Tel: +90 (216) 5698400

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Rockwell Automation 出版号 2080-RM001D-ZH-E - 2015 年 2 月

文档反馈

您的意见将有助于我们更好地满足您对文档的需求。如果您有更好的建议，请完成此反馈表，出版号 RA-DU002。

Rockwell Otomasyon Ticaret A.Ş., Kar Plaza İş Merkezi E Blok Kat:6 34752 İçerenköy, İstanbul, Tel: +90 (216) 5698400

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444

Europe/Middle East/Africa: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640

Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Rockwell Automation 出版号 2080-RM001D-ZH-E - 2015 年 2 月