

Micro800 可编程控制器一般说明

类型编号 2080-L20E, 2080-L50E, 2080-L70E, 2080-LC10, 2080-LC20, 2080-LC30, 2080-LC50, 2080-LC70

Rockwell Automation 出版号 2080-RM001M-ZH-E - 2025 年 5 月
取代出版号 2080-RM001L-ZH-E - 2023 年 11 月

Rockwell Automation

参考手册

原始说明

重要用户信息

在安装、配置、操作或维护本产品之前，请阅读本文件以及其他资源部分列出的有关安装、配置和操作此设备的文件。用户需要了解所有适用规范、法律和标准的要求，除此之外，还需要熟悉安装和接线说明。

应由接受过专门培训的人员依照适用的操作规范，执行安装、调整、交付使用、使用、装配、拆卸及维护等活动。

如果未按制造商指定的方式使用本设备，会削弱设备本身的防护作用。

对于使用或应用本设备所导致的间接或后续性损害，Rockwell Automation, Inc. 概不负责。

本手册中的示例和图表仅供说明之用。由于任何特殊安装都存在诸多不确定因素和要求，因此，对于按照示例和图表使用产品所产生的后果，Rockwell Automation, Inc. 不承担任何责任。

对于使用本手册所描述的信息、电路、设备或软件的情况，Rockwell Automation, Inc. 不承担专利责任。

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

本手册通篇都会在必要的情况下使用说明注释来提醒安全注意事项。



警告：标识可能导致暴露于危险环境从而造成人员伤亡、财物损害或经济损失的做法或环境的相关信息。



注意：标识可能导致人员伤亡、财物损害或经济损失的做法或环境的相关信息。注意事项可帮助您识别危险，避开危险，以及意识到后果。

重要 标识对成功应用和了解产品至关重要的信息。

还会在设备上或内部使用标签来提供具体预防措施。



电击危险：标签可能贴在设备（例如驱动器或电机）上或内部，提醒人们可能存在危险电压。



烧灼危险：标签可能贴在设备（例如驱动器或电机）上或内部，提醒人们设备表面可能达到危险温度。



弧闪危险：标签可能贴在设备（例如驱动器或电机）上或内部，提醒人们可能发生弧闪。弧闪将导致重伤或死亡。穿戴适当的个人防护装置 (PPE)。遵循安全作业方法和个人防护装置 (PPE) 的全部监管要求。

Rockwell Automation 认识到，当前在本行业和本出版物中使用的一些术语与中性技术语言的发展趋势不一致。我们正积极与业界同行合作，寻找此类术语的替代术语并对我们的产品和内容进行更改。在实施这些更改的同时，我们会在我们的内容中使用此类术语，敬请谅解。

目录

前言

更改摘要 15
在本手册中 15
支持的控制器 15
附加资源 16
法律声明 17

章节 1

查找有关指令和梯形图元素的信息

指令块 19
- 运算符 19
- 功能 19
- 功能块 20

按字母顺序排列的指令集 21

章节 2

梯形图 (LD) 语言

梯形图 (LD) 程序 29
LD 程序开发环境 29
- 指令工具栏 (LD) 30
  - 从指令工具栏添加指令元素 30

章节 3

梯形图 (LD) 元素

- 梯级 31
  - 向 LD 程序添加梯级 31
  - 梯级标签 33
  - 梯级注释 33
- 分支 34
  - 向 LD 程序添加分支 35
- 指令块 (LD) 37
- 线圈 37
  - 添加线圈单元 37
  - 直接线圈 39
  - 反向线圈 40
  - 脉冲下降沿线圈 40
  - 脉冲上升沿线圈 41
  - 置位线圈 41
  - 复位线圈 42
- 触点 43
  - 向 LD 程序添加触点 43
  - 直接接触 45
  - 反向接触 46
  - 脉冲上升沿触点 46

脉冲下降沿触点 47
返回 48
跳转 48
LD 程序中的指令块 49
使用 LD 语言编辑器 50
梯形图 (LD) 程序示例 51
LD 键盘快捷键 52

章节 4

警报指令

LIM_ALRM（限制警报） 55

章节 5

算术指令

ABS（绝对值） 59
ACOS（源的反余弦） 61
ACOS_LREAL（长实型反余弦） 62
Addition 64
ASIN（反正弦） 65
ASIN_LREAL（长实型反正弦） 66
ATAN（反正切） 68
ATAN_LREAL（长实型反正切） 69
COS（余弦） 71
COS_LREAL（余弦长实型） 72
Division 74
EXPT（指数） 75
LOG（对数底为 10） 77
MOD（模） 78
MOV（移动） 80
Multiplication 81
Neg（取反） 83
POW（增加幂） 83
RAND（随机值） 85
SIN（正弦） 87
SIN_LREAL（正弦长实型） 88
SQRT（平方根） 89
Subtraction 91
TAN（正切） 92
TAN_LREAL（正切长实型） 94
TRUNC（截断） 95

章节 6

ASCII 串行端口指令

ABL（缓冲区行的 ASCII 测试） 97
ACL（ASCII 清除缓冲区） 99

AHL（ASCII 握手信号行） 101
ACB（缓冲区中的 ASCII 字符） 103
ARD（ASCII 读取） 105
ARL（ASCII 读取行） 107
AWA（ASCII 写入附加） 109
AWT（ASCII 写入） 111
ASCII 参数详细信息 113
ABLACB 数据类型 114
ACL 数据类型 114
AHL ChannelSts 数据类型 114
AHLI 数据类型 114
ARDARL 数据类型 115
AWAAWT 数据类型 115

章节 7

二进制指令

AND_MASK（AND 掩码） 117
BSL（向左移动位） 119
BSR（向右移动位） 122
NOT_MASK（位到位 NOT 掩码） 125
OR_MASK（位到位 OR 掩码） 126
ROL（向左旋转） 128
ROR（向右旋转） 129
SHL（向左移动） 131
SHR（向右移动） 133
XOR_MASK（异或掩码） 134

章节 8

布尔指令

F_TRIG（下降沿检测） 137
R_TRIG（上升沿探测器） 139
RS（复位/设置） 140
OR 142
AND 142
XOR（异或） 143
NOT 143
SR（设置/复位） 144
NOT（真值表） 146
MUX8B（8 个 BOOL 输入的乘法器） 148
MUX4B（4 个 BOOL 输入的乘法器） 152

章节 9

通信指令

COM_IO_WDOG 155
MSG_CIPGENERIC（通用工业协议泛型消息） 158

CIPAPPCFG 数据类型 160
CIPCONTROLCFG 数据类型 160
CIPSTATUS 数据类型 161
CIPSTATUS 状态位 161
CIPSTATUS 错误代码 162
CIPTARGETCFG 数据类型 163
CIP 消息的目标路径 163
CIP/EIP 消息连接 164
CIP 消息超时计时器 164
MSG_CIPSYMBOLIC（通用工业协议符号消息） 165
CIPSYMBOLICCFG 数据类型 166
符号读取/写入语法 167
已支持的 CIP 串行功能数据包大小 168
MSG_MODBUS（modbus 消息） 168
MODBUSLOCPARA 数据类型 171
消息执行过程 (Rung = TRUE) 172
MODBUSTARPARA 数据类型 173
MSG_MODBUS2（MODBUS/TCP 消息） 174
MODBUS2LOCPARA 数据类型 176
MODBUS2TARPARA 数据类型 177
MSG_PCCC（可编程控制器通信命令消息） 179
CIPCONTROLCFG 数据类型 181
PCCCCFG 数据类型 181
TARGETCFG 数据类型 182
CIPSTATUS 数据类型 184
CIPSTATUS 状态位 184
CIP/EIP 消息连接 185
消息超时 185
MSG_PCCC 错误代码 186
示例：在 MicroLogix 中创建 MSG_PCCC 消息程序以将值写入地址和从地址读取值 188
PCCC 写入示例：添加 MSG_ PCCC 功能块、变量和触点 188
PCCC 读取示例：添加 PCCC_MSG 功能块、变量和触点 191
消息执行过程和时间图 194
消息执行过程（常规） 195
消息执行过程 (Rung = TRUE) 195
消息执行时间图 (Rung = TRUE) 196
消息执行过程 (Rung = FALSE) 197
消息执行时间图 (Rung = FALSE) 198
消息执行过程（错误） 199
消息执行时间图（错误） 199

消息执行时间图（PCCC 配置错误）.....200
消息执行时间图（消息已取消）.....201
使用通信（消息）功能块.....202
配置显式消息的对象数据值 (MSG_CIPGENERIC).....202
示例：如何创建 MSG_CIPGENERIC 消息程序以从控制器中读取数据.....204
确定输入变量 MSG_CIPGENERIC 的初始值.....204
添加 MSG_CIPGENERIC 功能块和变量.....205
配置变量的初始值.....206
添加触点和线圈.....208
添加 COP 功能块、变量和触点 (MSG_CIPGENERIC).....209
确认控制器 B 上正确的 IP 配置.....210
示例：如何创建 MSG_CIPSYMBOLIC 消息程序以将值写入变量.....210
添加 COP 功能块、变量和触点 (MSG_CIPSYMBOLIC).....211
添加 Equal 运算符和线圈.....212
添加 MSG_CIPSYMBOLIC 功能块、变量和触点.....213
确认控制器 B 上正确的 IP 配置.....215
为控制器 B 创建全局变量.....216
查看整个程序结果.....216
示例：如果配置 Modbus 通信以读取和写入驱动器.....217
通信协议支持.....221
嵌入式通信信道.....222

章节 10

比较指令

Equal.....223
Greater than.....225
Greater Than or Equal.....227
Less Than.....228
Less Than or Equal.....230
Not equal.....230

章节 11

计数器指令

CTD（向下计数）.....233
CTU（向上计数）.....235
CTUD(向上/向下计数).....237

章节 12

数据转换指令

ANY_TO_BOOL.....239
ANY_TO_BYTE.....240
ANY_TO_DATE.....241
ANY_TO_DINT.....243

ANY_TO_DWORD 244
ANY_TO_INT 245
ANY_TO_LINT 246
ANY_TO_LREAL 247
ANY_TO_LWORD 248
ANY_TO_REAL 248
ANY_TO_SINT 249
ANY_TO_STRING 250
ANY_TO_TIME 252
ANY_TO_UDINT 253
ANY_TO_UINT 253
ANY_TO_ULINT 254
ANY_TO_USINT 255
ANY_TO_WORD 256

章节 13

数据操作指令

AVERAGE 259
COP 260
复制到不同的数据类型 263
COP 字符串数组示例 264
MIN（最小值） 265
MAX（最大值） 267

章节 14

高速计数器 (HSC) 指令

什么是高速计数器？ 269
HSC（高速计数器） 269
HSCCmd 值 271
HSCAPP 数据类型 272
HSCSTS 数据类型 275
PLS 数据类型 281
HSCE_CHANNEL 数据类型 282
HSCE_STS 数据类型 283
PLS_HSCE 数据类型 283
HSC_SET_STS（高速计数器设置状态） 284
使用高速计数器指令 286
更新 HSC 应用程序数据 286
“高速计数器 (HSC) 用户中断”对话框 287
配置高速计数器 (HSC) 用户中断 287
添加和配置高速计数器 (HSC) 用户中断 288
HSC 中断属性 288
HSC 中断参数 289
配置可编程限位开关 (PLS) 290

示例：如何创建高速计数器 (HSC) 程序 291
创建梯形图并添加变量 292
将值分配给 HSC 变量 295
分配变量并生成程序 296
测试程序并运行高速计数器 297
添加可编程限位开关 (PLS) 功能 299
示例：可编程限位开关 (PLS) 已启用 300

章节 15

HSCE 指令

HSCE 303
HSCE_CFG 306
HSCE_CFG_PLS 308
HSCE_READ_STS 310
HSCE_SET_STS 311
HSCE 错误代码 313

章节 16

输入/输出指令

LCD 317
LCD_BKLT_REM（远程 LCD 背光） 319
LCD_REM（远程 LCD） 321
RHC（读取高速时钟） 324
RPC（读取程序校验和） 325
DLG（数据记录） 326
IIM（即时输入） 328
IOM（即时输出） 330
KEY_READ（读取 LCD 上的键） 332
KEY_READ_REM（远程 LCD 读取键） 334
MM_INFO（内存模块信息） 336
MMINFO 数据类型 338
MODULE_INFO 338
MODULE_INFO - 插件和扩展模块信息 342
MODULE_INFO 指令时间图 344
PLUGIN_INFO（插件信息） 348
PLUGIN_READ（读取插件） 351
PLUGIN_RESET（复位插件） 353
PLUGIN_WRITE（写入插件） 354
RCP（配方） 356
RTC_READ（读取实时时钟） 358
RTC_SET（设置实时时钟） 360
SYS_INFO 362
SYS_INFO 数据类型 363
TRIMPOT_READ（读取微调电位计） 364

微调电位 ID 定义 365

中断指令

章节 17

STIS（选择计时启动） 367
UIC（清除中断丢失位） 369
UID（禁用中断） 370
UIE（启用中断） 372
UIF（刷新挂起中断） 374

运动控制指令

章节 18

运动控制功能块的一般规则 377
运动控制功能块参数详细信息 380
运动控制轴状态 380
运动控制功能块参数编号 382
运动控制功能块错误 ID 383
轴出错场景 384
AXIS_REF 数据类型 384
FB_AXIS_REF 数据类型 385
轴变量 386
MC_AbortTrigger（运动控制中止触发器） 386
MC_Halt（运动控制停止） 389
MC_Home（运动控制原位） 391
MC_MoveAbsolute（运动控制移动绝对值） 395
MC_MoveRelative（运动控制移动相对值） 398
MC_MoveVelocity（运动控制移动速度） 402
MC_Power（运动控制电源） 406
MC_ReadActualPosition（运动控制读取实际位置） 409
MC_ReadActualVelocity（运动控制读取实际速度） 412
MC_ReadAxisError（运动控制读取轴错误） 415
AxisErrorID 错误代码 418
MC_ReadBoolParameter（运动控制读取 BOOL 参数） 419
MC_ReadParameter（运动控制读取参数） 422
MC_ReadStatus（运动控制读取状态） 424
MC_Reset（运动控制复位） 428
MC_SetPosition（运动控制设置位置） 431
MC_Stop（运动控制停止） 433
MC_TouchProbe（运动控制触摸探针） 436
MC_WriteBoolParameter（运动控制写入 BOOL 参数） 440
MC_WriteParameter（运动控制写入参数） 443

过程控制指令

章节 19

DERIVATE 447

FFL（FIFO 加载）.....449
FFL 和 FFU 指令时间图.....453
FFU（FIFO 卸载）.....457
HYSTER（滞后）.....462
INTEGRAL.....464
AND.....468
LFL（LIFO 加载）.....469
LFU（LIFO 卸载）.....472
PWM（脉宽调制）.....474
SCALER（刻度盘）.....477
STACKINT（堆栈整数）.....479
SCL.....481
SCL 指令时间图示例.....484
TND（停止当前程序）.....488
LIMIT（限制测试）.....489

章节 20

程序控制指令

AFI（始终为 False）.....493
NOP（无操作）.....493
SUS（挂起）.....493

章节 21

比例积分微分 (PID) 指令

什么是比例积分微分 (PID) 控制？.....497
IPIDCONTROLLER（比例-积分-微分控制器）.....497
GAIN_PID 数据类型.....500
AT_Param 数据类型.....501
IPIDController 功能块如何实施 PID 控制.....501
IPIDController 功能块操作.....502
使用比例积分微分指令.....503
通过 IPIDController 功能块实现自动调节.....505
一阶和二阶系统中的自动调节.....505
配置自动调节.....506
通过自动调节使用可选定时中断 (STI).....512
示例：具有自动调节的 IPIDController.....512
示例：如何创建操作值的反馈回路.....513
示例：如何将 UDFB 添加到 PID 程序.....514
示例：如何创建 IPIDController 程序以控制温度.....515
示例：如何创建 IPIDController 程序以控制供水水位.....516
PID（比例-积分-微分）.....518
PID 指令状态机.....522
PID 指令时间图.....523

章节 22

实时时钟（RTC）指令

RTC_READ（读取实时时钟） 527
RTC_SET（设置实时时钟） 529

章节 23

套接字指令

SOCKET_ACCEPT 534
SOCKET_CREATE 537
SOCKET_DELETE 541
SOCKET_DELETEALL 543
SOCKET_INFO 545
SOCKET_OPEN 550
SOCKET_READ 554
SOCKET_WRITE 558
SOCKADDR_CFG 数据类型 561
SOCK_STATUS 数据类型 562
套接字错误代码 562
套接字指令状态位 565
套接字指令时间图 566
套接字指令处理图 572
对应于 TCP 客户端的处理 572
对应于 TCP 服务器的处理 573
对应于采用 Open Connection 的 UDP 的处理 573
对应于未采用 Open Connection 的 UDP 的处理 574
TCP 的状态机图 574
UDP 的状态机图 576

章节 24

字符串操作指令

ASCII 579
CHAR（字符串字符的 ASCII 代码） 580
DELETE（删除子字符串） 582
FIND（查找子字符串） 584
INSERT（插入字符串） 586
LEFT（提取字符串的左侧） 588
MID（提取字符串的中间） 589
MLEN（字符串长度） 591
RIGHT（提取字符串的右侧） 593
REPLACE（替换子字符串） 595

章节 25

计时器指令

计时器指令配置 599
TOF（计时器，关闭延时） 599
TON（计时器，打开延时） 601

TONOFF（时间延迟，打开，关闭） 603
TP（脉冲计时） 605
RTO（保持计时器，打开延时） 607
DOY（检查实时时钟的年份） 609
TDF（时间差） 611
TOW（实时时钟复选周） 613

索引

更改摘要

本出版物包含以下新增或更新的信息。此列表仅包含重大更新，并不旨在反映所有更改。

主题	更改原因
支持的控制器 参考页数 15	支持 2080-L20E 控制器并更新 2080-LC20 控制器的目录号。
AND 参考页数 142	更新 AND 指令的描述。
LCD_BKLT_REM（远程 LCD 背光） 参考页数 319	2080-L50E 和 2080-L70E 控制器支持 LCD_BKLT_REM 指令。
LCD_REM（远程 LCD） 参考页数 321	2080-L50E 和 2080-L70E 控制器支持 LCD_REM 指令。
KEY_READ_REM（远程 LCD 读取键） 参考页数 334	2080-L50E 和 2080-L70E 控制器支持 KEY_READ_REM 指令。
MSG_PCCC（可编程控制器通信命令消息） 参考页数 179	2080-L50E 控制器支持 MSG_PCCC 指令。
LFL（LIFO 加载） 参考页数 469	添加 LFL 错误代码表。
LFU（LIFO 卸载） 参考页数 472	添加 LFU 错误代码表。
COP（复制文件） 参考页数 260	更新 COP 指令的描述

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® 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 Components Workbench™ 开发环境中支持的梯形图元素。

支持的控制器

Connected Components Workbench 包括对以下 Micro800 控制器的配置支持。

- 2080-LC10-12AWA
- 2080-LC10-12DWD
- 2080-LC10-12QBB
- 2080-LC10-12QWB
- 2080-LC20-20AWB(R)
- 2080-LC20-20QBB(R)
- 2080-LC20-20QWB(R)
- 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
- 2080-LC50-48QWB - SIM
- 2080-L20E-20AWB(R)
- 2080-L20E-20QBB(R)
- 2080-L20E-20QWB(R)
- 2080-L50E-24AWB
- 2080-L50E-24QBB
- 2080-L50E-24QVB
- 2080-L50E-24QWB
- 2080-L50E-48AWB
- 2080-L50E-48QBB
- 2080-L50E-48QVB
- 2080-L50E-48QWB
- 2080-L70E-24AWB
- 2080-L70E-24QBB
- 2080-L70E-24QBBN
- 2080-L70E-24QWB
- 2080-L70E-24QWBN
- 2080-LC70-24AWB
- 2080-LC70-24QBB
- 2080-LC70-24QWB

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	描述
《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。
MicroLogix 控制器到 Micro800 控制器的迁移指南：https://literature.rockwellautomation.com/idc/groups/literature/documents/rm/2080-rm002_-en-e.pdf	介绍如何使用软件工具选择合适的 Micro800 控制器，以及如何转换 MicroLogix 程序以用于 Micro800 控制器。

资源	描述
Micro800 控制器新手程序包快速入门：https://literature.rockwellautomation.com/idc/groups/literature/documents/qs/2080-qs004_-en-e.pdf	介绍如何将 Micro800 控制器与 PanelView 800 终端配合使用。
在 FactoryTalk Gateway 上配置 Micro800 控制器：https://literature.rockwellautomation.com/idc/groups/literature/documents/qs/2080-qs005_-en-e.pdf	提供在 FactoryTalk Linx Gateway 上配置 Micro800 控制器的快速入门说明。

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。

法律声明

Rockwell Automation 在 Rockwell Automation 网站的法律声明页面上发布法律声明，如隐私政策、许可协议、商标披露以及其他条款和条件。

软件和云服务协议

请在此处查看并接受《Rockwell Automation 软件和云服务协议》。

开源软件许可证

本产品中包括的软件包含受版权保护的软件，其通过一个或多个开源许可授权。

您可以打开硬盘驱动器上产品 OPENSOURCE 文件夹内的 oss_license.txt 文件，查看本产品中使用的所有开源软件及其相应许可证的完整列表。该文件分为以下几个部分：

- 组件
  包括开源组件的名称、其版本号和许可证类型。
- 版权文本
  包括开源组件的名称、其版本号和版权声明。
- 许可证
  包括许可证的名称、引用该许可证的开源组件的列表，以及许可证的条款。

该文件的默认位置为：

C:\Program Files (x86)\Common Files\Rockwell\Help\<*product name*>\Release Notes\OPENSOURCE\oss_licenses.txt。

您可以从其各自的项目网站获得本产品中包含的开源程序包的相应源代码。或者，您可以通过 Rockwell Automation 网站：http://www.rockwellautomation.com/global/about-us/contact/contact.page 上的“联系”表单联系 Rockwell Automation，获得完整的相应源代码。请包含“开放源代码”作为申请文本的一部分。

商业软件许可证

下表列出了 Connected Components Workbench 中的商业许可的软件组件。

组件	版权
Actipro v17.1.0651	版权所有 (c) 2002-2013 Actipro Software LLC
Actipro v4.0.289	版权所有 (c) 2001-2009 Actipro Software LLC
Actipro v1.0.104	版权所有 (c) 2001-2008 Actipro Software LLC
ag-grid v21.0.1	版权所有 (c) 2015-2020 AG GRID LTD
GoDiagram for WinForms v4.1.0.4	版权所有 (c) Northwoods Software Corporation, 1998-2010
GoXam for WPF v1.2.6.4	版权所有 (c) Northwoods Software, 1998-2011

查找有关指令和梯形图元素的信息

Connected Components Workbench 包括一个全面指令集，其中包含结构和数组、梯形图逻辑开发环境、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。

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 21

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

- 梯形图 (LD) 语言参考 参考页数 29

指令块

Connected Components Workbench 指令集包括适用于 Micro800 控制器的、符合 IEC 61131-3 标准的指令块。指令块总的来说包括运算符 参考页数 19、函数 参考页数 19和功能块 参考页数 20。

运算符

Connected Components Workbench 指令集包括用于 Micro800 控制器的符合 IEC 61131-3 标准的指令块；Connected Components Workbench 指令集中包含运算符。

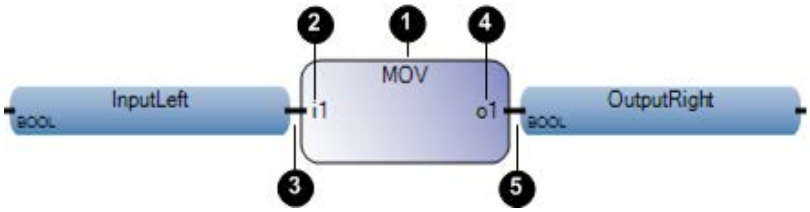
运算符是指诸如算术运算、布尔运算、比较运算或数据转换等基本逻辑操作。

功能

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。



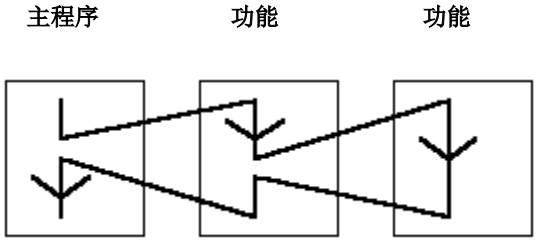
项目编号	项目	描述
1	块名称	指令块要执行的功能的名称写在其矩形形状内（位于顶端）。

项目编号	项目	描述
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。

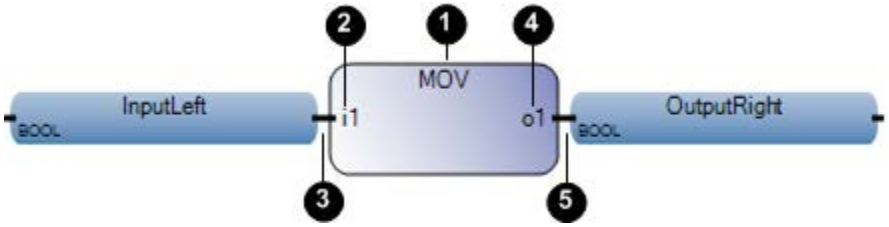
函数名称和函数参数名称最多可包含 128 个字符。函数参数名称可以字母或下划线字符开头，后跟字母、数字和单个下划线字符。

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	描述
1	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）。
2	输入	指令块的每个输入都带有标签且具有定义的类型。
3	输入连接	输入在左边框进行连接。
4	输出	指令块的每个输出都带有标签且具有定义的类型。
5	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。

按字母顺序排列的指令集

下表按字母顺序列出了 Connected Components Workbench 控制器中的可用 Micro800 控制器指令以及它们在 Logix 主题中的映射指令。

指令	映射指令（Logix 主题）	类别	类型	描述
- 参考页数 91	SUB	算术	运算符	用一个整型、实型或时间值减去另一个整型、实型或时间值。
* 参考页数 81	MUL	算术	运算符	乘以两个或多个整型或实型值。
/ 参考页数 74	DIV	算术	运算符	两个整型值或实型值相除。
+ 参考页数 64	ADD	算术	运算符	添加两个或多个整型、实型、时间或字符串值。

指令	映射指令（Logix 主题）	类别	类型	描述
< 参考页数 228	LES	比较	运算符	比较整型、实型、时间、日期和字符串输入值，以确定第一个输入是否小于第二个输入。
<= 参考页数 230	LEQ	比较	运算符	比较整型、实型、时间、日期和字符串输入值，以确定第一个输入是否小于或等于第二个输入。
<> 参考页数 230	NEQ	比较	运算符	比较整型、实型、时间、日期和字符串输入值，以确定第一个输入是否不等于第二个输入。
= 参考页数 223	EQU	比较	运算符	测试一个值是否等于另一个值。
> 参考页数 225	GRT	比较	运算符	比较整型、实型、时间、日期和字符串输入值，以确定第一个输入是否大于第二个输入。
>= 参考页数 227	GEQ	比较	运算符	比较整型、实型、时间、日期和字符串输入值，以确定第一个输入是否大于或等于第二个输入。
ABL 参考页数 97	ABL	通信	功能块	计算缓冲区中的字符数（甚至可包括行尾字符）。
ABS 参考页数 59	ABS	算术	功能	返回实型值的绝对值。
ACB 参考页数 103	ACB	通信	功能块	计算缓冲区中的字符总数。
ACL 参考页数 99	ACL	通信	功能块	清除接收和传输缓冲区。
ACOS 参考页数 61	ACS	算术	功能	计算实型值的反余弦。
ACOS_LREAL 参考页数 62	ACOS_LREAL	算术	功能	计算长实型值的反余弦。
AFI 参考页数 493	AFI	程序控制	功能	调试时临时禁用梯级。
AHL 参考页数 101	AHL	通信	功能块	设置或复位调制解调器握手信号行。
AND 参考页数 142	AND	布尔运算	运算符	在两个或多个值之间执行布尔与操作。
AND_MASK 参考页数 117	AND_MASK	二进制运算	功能	在两个整型值之间执行位到位 AND 运算。
ANY_TO_BOOL 参考页数 239	ANY_TO_BOOL	数据转换	功能	将非布尔值转换为布尔值。
ANY_TO_BYTE 参考页数 240	ANY_TO_BYTE	数据转换	功能	将值转换为字节。
ANY_TO_DATE 参考页数 241	ANY_TO_DATE	数据转换	功能	将字符串、整型、实型或时间数据类型转换为日期数据类型。
ANY_TO_DINT 参考页数 243	STOD	数据转换	功能	将值转换为双整型。
ANY_TO_DWORD 参考页数 244	ANY_TO_DWORD	数据转换	功能	将值转换为双字值。
ANY_TO_INT 参考页数 245	ACI	数据转换	功能	将值转换为整型。
ANY_TO_LINT 参考页数 246	ANY_TO_LINT	数据转换	功能	将值转换为长整型。
ANY_TO_LREAL 参考页数 247	ANY_TO_LREAL	数据转换	功能	将值转换为长实型。
ANY_TO_LWORD 参考页数 248	ANY_TO_LWORD	数据转换	功能	将值转换为长字型。
ANY_TO_REAL 参考页数 248	STOR	数据转换	功能	将值转换为实型。
ANY_TO_SINT 参考页数 249	ANY_TO_SINT	数据转换	功能	将值转换为短整型。
ANY_TO_STRING 参考页数 250	DTOS	数据转换	功能	将值转换为字符串。
ANY_TO_TIME 参考页数 252	ANY_TO_TIME	数据转换	功能	将值转换为时间数据类型。
ANY_TO_UDINT 参考页数 253	ANY_TO_UDINT	数据转换	功能	将值转换为无符号双整型。
ANY_TO_UINT 参考页数 253	ANY_TO_UINT	数据转换	功能	将值转换为无符号整型。
ANY_TO_ULINT 参考页数 254	ANY_TO_ULINT	数据转换	功能	将值转换为无符号长整型。
ANY_TO_USINT 参考页数 255	ANY_TO_USINT	数据转换	功能	将值转换为无符号短整型。
ANY_TO_WORD 参考页数 256	ANY_TO_WORD	数据转换	功能	将值转换为字。
ARD 参考页数 105	ARD	通信	功能块	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 107	ARL	通信	功能块	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。
ASCII 参考页数 579	ASCII	字符串操纵	功能	返回字符串形式的字符 ASCII 代码。字符 -> ASCII 代码。

指令	映射指令（Logix 主题）	类别	类型	描述
ASIN 参考页数 65	ASN	算术	功能	计算实型值的反正弦。
ASIN_LREAL 参考页数 66	ASN_LREAL	算术	功能	计算长实型值的反正弦。
ATAN 参考页数 68	ATN	算术	功能	计算实型值的反正切。
ATAN_LREAL 参考页数 69	ATAN_LREAL	算术	功能	计算长实型值的反正切。
AVERAGE 参考页数 259	AVE	数据操纵	功能块	计算若干定义样本上的运行平均值。
AWA 参考页数 109	AWA	通信	功能	将包含两个附加（用户配置）字符的字符串写入外部设备。
AWT 参考页数 111	AWT	通信	功能	将字符从源字符串写入外部设备。
BSL 参考页数 119	BSL	二进制运算	功能块	将数组元素中的位向左移动。
BSR 参考页数 122	BSR	二进制运算	功能块	将数组元素中的位向右移动。
CHAR 参考页数 580	CHAR	字符串操纵	功能	返回 ASCII 代码的一个字符字符串。ASCII 代码 -> 字符。
COM_IO_WDOG 参考页数 155	COM_IO_WDOG	通信	功能块	监控与控制器的通信。
COP 参考页数 260	COP	数据转换	功能块	将源元素中的二进制数据复制到目标元素。
COS 参考页数 71	COS	算术	功能	计算实型值的余弦。
COS_LREAL 参考页数 72	COS_LREAL		功能	计算长实型值的余弦。
CTD 参考页数 233	CTD	计数器	功能	它从给定值到 0 逐个向下计数（整数）。
CTU 参考页数 235	CTU	计数器	功能	从 0 到给定值逐个向上计数（整数）。
CTUD 参考页数 237	CTUD	计数器	功能	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。
DELETE 参考页数 582	DELETE	字符串操纵	功能	从字符串中删除字符。
DERIVATE 参考页数 447	DERIVATE	进程控制	功能块	实型值在定义的循环时间内的差分。
DLG 参考页数 326	DLG	输入/输出	功能块	将变量值从运行时引擎写入到 SD 卡上的数据记录文件。
DOY 参考页数 609	DOY	时间	功能	如果实时时钟的值位于"年时间"设置范围内，则开启输出。
EXPT 参考页数 75	EXPT	算术	功能	计算增加至整型指数的幂的基数的实型值。
F_TRIG 参考页数 137	OSF	布尔运算	功能块	检测布尔变量的下降沿。
FFL 参考页数 457	FFL	进程控制	功能块	将 8 位、16 位、32 位或 64 位数据加载到调用 FIFO 堆栈的用户创建的数组。
FFU 参考页数 457	FFU	进程控制	功能块	以使用 FFL 指令加载数据的相同顺序，从调用 FIFO（先进先出）堆栈的用户创建数组卸载 8 位、16 位、32 位或 64 位数据。
FIND 参考页数 584	FIND	字符串操纵	功能	在字符串中定位并提供子字符串的位置。
HSC 参考页数 269	HSC	输入/输出	功能块	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 284	HSC_SET_STS	输入/输出	功能块	HSC_SET_STS 手动设置或重置 HSC 计数状态。
HSCE 参考页数 303	HSCE	输入/输出	功能块	HSCE 开始、停止和读取累加值。
HSCE_CFG 参考页数 306	HSCE_CFG	输入/输出	功能块	HSCE_CFG 是高速计数器配置。
HSCE_CFG_PLS 参考页数 308	HSCE_CFG_PLS	输入/输出	功能块	HSCE_CFG_PLS 是高速计数器 PLS 配置。
HSCE_READ_STS 参考页数 310	HSCE_READ_STS	输入/输出	功能块	HSCE_READ_STS 读取高速计数器状态。
HSCE_SET_STS 参考页数 311	HSCE_SET_STS	输入/输出	功能块	HSCE_SET_STS 手动设置/复位高速计数器状态。
HYSTER 参考页数 462	HYSTER	进程控制	功能块	在不同实型值上的布尔迟滞。
IIM 参考页数 328	IIM	输入/输出	功能块	在进行正常输出扫描前更新输入。
INSERT 参考页数 586	INSERT	字符串操纵	功能	在字符串中用户指定的位置插入子字符串。
INTEGRAL 参考页数 464	INTEGRAL	进程控制	功能块	在定义的循环时间期间集成实型值。

指令	映射指令（Logix 主题）	类别	类型	描述
IOM 参考页数 330	IOM	输入/输出	功能块	在进行正常输出扫描前更新输出。
IPIDCONTROLLER 参考页数 497	IPIDCONTROLLER	进程控制	功能块	配置和控制用于比例积分微分 (PID) 逻辑的输入和输出。
KEY_READ 参考页数 332		输入/输出	功能块	仅限 Micro810。 在用户显示处于活动状态时读取可选 LCD 模块上的键状态。
KEY_READ_REM 参考页数 334	KEY_READ_REM	输入/输出	功能块	仅限 Micro820、2080-L50E 和 2080-L70E。 在用户显示处于活动状态时读取可选远程 LCD 模块上的键状态。
LCD 参考页数 317		输入/输出	功能	仅限 Micro810。 在 LCD 屏幕上显示字符串或数字。
LCD_BKLT_REM 参考页数 319	LCD_BKLT_REM	输入/输出	功能	在用户程序中设置远程 LCD 背光参数。
LCD_REM 参考页数 321	LCD_REM	输入/输出	功能	显示远程 LCD 屏幕的用户定义消息。
LEFT 参考页数 588	LEFT	字符串操纵	功能	从字符串左侧提取字符。
LFL（LIFO 加载） 参考页数 469	LFL	进程控制	功能块	将 8 位、16 位、32 位或 64 位数据加载到调用 LIFO 堆栈的用户创建数组。
LFU（LIFO 卸载） 参考页数 472	LFU	进程控制	功能块	以使用 LFL 指令加载数据的相同顺序，从调用 LIFO（后进先出）堆栈的用户创建数组卸载 8 位、16 位、32 位或 64 位数据。
LIM_ALRM 参考页数 55	LIM	报警	功能块	关于上限和下限实型值滞后的警报。
LIMIT 参考页数 489	LIMIT	进程控制	功能	将整型值限制为给定的间隔。
LOG 参考页数 77	LOG	算术	功能	计算实型值的对数（以 10 为底）。
MAX 参考页数 267	MAX	数据操纵	功能	计算两个整型值的最大值。
MC_AbortTrigger 参考页数 386	MC_AbortTrigger	运动	功能块	中止连接到触发事件的运动功能块。
MC_Halt 参考页数 389	MC_Halt	运动	功能块	命令受控制的运动在正常操作条件下停止。
MC_Home 参考页数 391	MAH	运动	功能块	命令轴执行 *<search home>* 序列。
MC_MoveAbsolute 参考页数 395	MAM	运动	功能块	命令受控制的运动到指定的绝对位置。
MC_MoveRelative 参考页数 398	MC_MoveRelative	运动	功能块	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 402	MCD	运动	功能块	以指定速率命令从未结束的受控制运动。
MC_Power 参考页数 406	MSO	运动	功能块	控制功率（打开或关闭）。
MC_ReadActualPosition 参考页数 412	MC_ReadActualPosition	运动	功能块	返回反馈轴的实际位置。
MC_ReadActualVelocity 参考页数 412	MC_ReadActualVelocity	运动	功能块	返回反馈轴的实际速度。
MC_ReadAxisError 参考页数 415	MC_ReadAxisError	运动	功能块	读取与运动控制指令块无关的轴错误。
MC_ReadBoolParameter 参考页数 419	MC_ReadBoolParameter	运动	功能块	返回特定于供应商的类型为 BOOL 的参数的值。
MC_ReadParameter 参考页数 422	MC_ReadParameter	运动	功能块	返回特定于供应商的类型为实型的参数的值。
MC_ReadStatus 参考页数 424	MC_ReadStatus	运动	功能块	返回与当前正在进行中的运动相关的轴的状态。
MC_Reset 参考页数 428	MAFR	运动	功能块	通过复位所有内部轴相关错误将轴状态从 ErrorStop 转换为 StandStill。
MC_SetPosition 参考页数 431	MRP	运动	功能块	通过控制实际位置来转移轴坐标系统。
MC_Stop 参考页数 433	MAS	运动	功能块	命令受控制的运动停止并将轴状态转为 Stopping。
MC_TouchProbe 参考页数 436	MC_TouchProbe	运动	功能块	在触发事件中记录轴位置。
MC_WriteBoolParameter 参考页数 440	MC_WriteBoolParameter	运动	功能块	修改特定于供应商的类型为 BOOL 的参数的值。

指令	映射指令（Logix 主题）	类别	类型	描述
MC_WriteParameter 参考页数 443	MC_WriteParameter	运动	功能块	修改特定于供应商的类型为 REAL 的参数的值。
MID 参考页数 589	MID	字符串操纵	功能	从字符串中间提取字符。
MIN 参考页数 265	MIN	数据操纵	功能	计算两个整型值的最小值。
MLEN 参考页数 591	MLEN	字符串操纵	功能	计算字符串的长度。
MM_INFO 参考页数 336	MM_INFO	输入/输出	功能块	读取内存模块标题信息。
MOD 参考页数 78	MOD	算术	功能	对整型值执行模计算。
MODULE_INFO 参考页数 338	MODULE_INFO	输入/输出	功能块	从插件模块或扩展模块读取模块信息。
MOV 参考页数 80	MOV	算术	运算符	将输入值分配给输出。
MSG_CIPGENERIC 参考页数 158	MSG	通信	功能	发送 CIP 泛型显式消息。
MSG_CIPSYMBOLIC 参考页数 165	MSG_CIPSYMBOLIC	通信	功能	发送 CIP 符号显式消息。
MSG_MODBUS 参考页数 168	MSG_MODBUS	通信	功能	发送 Modbus 消息。
MSG_MODBUS2 参考页数 174	MSG_MODBUS2	通信	功能	通过以太网通道发送 MODBUS/TCP 消息。
MSG_PCCC 参考页数 179	MSG_PCCC	通信	功能	通过以太网通道或串行端口发送 PCCC 消息。
MUX4B 参考页数 152	MUX4B	布尔	功能	四个布尔型输入之间的乘法器，输出布尔型值。
MUX8B 参考页数 148	MUX8B	布尔	功能	八个布尔型输入之间的乘法器，输出布尔型值。
Neg 参考页数 83	NEG	算术	运算符	将值转换为负值。
NOP 参考页数 493	NOP	程序控制	功能	起占位符作用。
NOT 参考页数 143	NOT	布尔运算	运算符	将布尔值转换为反值。
NOT_MASK 参考页数 125	NOT_MASK	二进制运算	功能	整型位到位取反掩码，将反转参数值。
OR 参考页数 142	OR	布尔运算	运算符	两个或更多值的布尔或。
OR_MASK 参考页数 126	OR_MASK	二进制运算	功能	整型 OR 位到位掩码，将启用位。
PID 参考页数 518	PID	进程控制	功能块	使用过程循环控制诸如温度、压力、液面或流速等物理属性的输出指令。
PLUGIN_INFO 参考页数 348	PLUGIN_INFO	输入/输出	功能块	从类属插件模块中获取模块信息（不包括内存模块）。
PLUGIN_READ 参考页数 351	PLUGIN_READ	输入/输出	功能块	从类属插件模块中读取数据（不包括内存模块）。
PLUGIN_RESET 参考页数 353	PLUGIN_RESET	输入/输出	功能块	复位类属插件模块，硬件复位（不包括内存模块）。
PLUGIN_WRITE 参考页数 354	PLUGIN_WRITE	输入/输出	功能块	将数据写入类属插件模块（不包括内存模块）。
POW 参考页数 83	XPY	算术	功能	计算增加至实型指数的幂的实型数的值。
PWM 参考页数 474	PWM	进程控制	功能块	针对配置的 PWM 通道打开或关闭 PWM（脉宽调制）输出。
R_TRIG 参考页数 139	OSR	布尔运算	功能块	检测布尔变量的上升沿。
RAND 参考页数 85	RAND	算术	功能	从定义的范围计算随机整数值。
RCP 参考页数 356	RCP	输入/输出	功能块	从 SD 内存卡读取配方数据/向其写入配方数据。
REPLACE 参考页数 595	REPLACE	字符串操纵	功能	将字符串的一部分替换为新的字符集。
RHC 参考页数 324	RHC	输入/输出	功能	读取高速时钟。
RIGHT 参考页数 593	RIGHT	字符串操纵	功能	从字符串右侧提取字符。
ROL 参考页数 128	ROL	二进制运算	功能	对于 32 位整数，将整数位旋转到左侧。
ROR 参考页数 129	ROR	二进制运算	功能	对于 32 位整数，将整数位旋转到右侧。
RPC 参考页数 325	RPC	输入/输出	功能	读取用户程序校验和。
RS 参考页数 140	RS	布尔运算	功能块	重置优先指令。
RTC_READ 参考页数 358	RTC_READ	输入/输出	功能块	读取实时时钟 (RTC) 模块信息。
RTC_SET 参考页数 360	RTC_SET	输入/输出	功能块	将 RTC（实时时钟）数据设置为 RTC 模块信息。

指令	映射指令（Logix 主题）	类别	类型	描述
RTO 参考页数 607	RTO	时间	功能块	保持时间。当输入处于活动状态时增加内部计时器，但当输入变为不活动状态时不复位内部计时器。
SCALER 参考页数 477	SCP	进程控制	功能块	根据输出范围调整输入值。
SCL 参考页数 481	SCL	进程控制	功能块	将未缩放的输入值转换为采用工程单位的浮点值。
SHL 参考页数 131	SHL	二进制运算	功能	对于 32 位整数，将整数向左移动，并在最低有效位中置 0。
SHR 参考页数 133	SHR	二进制运算	功能	对于 32 位整数，将整数向右移动，并在最高有效位中置 0。
SIN 参考页数 87	SIN	算术	功能	计算实型值的正弦。
SIN_LREAL 参考页数 88	SIN_LREAL	算术	功能	计算长实型值的正弦。
SOCKET_ACCEPT 参考页数 534	SOCKET_ACCEPT	通信	功能块	接受来自远程目标的 TCP 连接请求并返回用于发送和接收新创建连接上的数据的套接字实例。
SOCKET_CREATE 参考页数 537	SOCKET_CREATE	通信	功能块	创建套接字实例并返回实例编号，以在任何后续套接字操作中用作输入。
SOCKET_DELETE 参考页数 541	SOCKET_DELETE	通信	功能块	删除创建的套接字实例。
SOCKET_DELETEALL 参考页数 543	SOCKET_DELETEALL	通信	功能块	删除创建的所有套接字实例。
SOCKET_INFO 参考页数 545	SOCKET_INFO	通信	功能块	返回套接字信息，如错误代码和执行状态。
SOCKET_OPEN 参考页数 550	SOCKET_OPEN	通信	功能块	打开指定目标地址的连接，以建立传输控制协议 (TCP) 连接。对于用户数据报协议 (UDP) 连接，会将目标 IP 地址和端口号关联到指定套接字。
SOCKET_READ 参考页数 554	SOCKET_READ	通信	功能块	读取套接字上的数据。
SOCKET_WRITE 参考页数 558	SOCKET_WRITE	通信	功能块	发送套接字上的数据。
SQRT 参考页数 89	SQR	算术	功能	计算实型值的平方根。
SR 参考页数 144	SR	布尔运算	功能块	置位优先指令。
STACKINT 参考页数 479	STACKINT	进程控制	功能块	管理整数堆栈。
STIS 参考页数 367	STS	中断	功能	从控制程序启动选定的定时用户中断 (STI) 计时器，而不是自动启动。
SUS 参考页数 493	SUS	程序控制	功能块	挂起 <M800 控制器> 的执行。
SYS_INFO 参考页数 362	SYS_INFO	输入/输出	功能块	读取 Micro800 控制器的状态数据块。
TAN 参考页数 92	TAN	算术	功能	计算实型值的正切。
TAN_LREAL 参考页数 94	TAN_LREAL	算术	功能	计算长实型值的正切。
TDF 参考页数 611	TDF	时间	功能	计算 TimeA 和 TimeB 之间的时间差。
TND 参考页数 488	TND	进程控制	功能	停止用户程序扫描的当前循环。
TOF 参考页数 599	TOF	时间	功能块	关闭延时计时。将内部计时器增加至指定值。
TON 参考页数 601	TON	时间	功能块	打开延时计时。将内部计时器增加至指定值。
TONOFF 参考页数 603	TONOFF	时间	功能块	延迟打开 TRUE 梯级上的输出，然后延迟关闭 FALSE 梯级上的输出。
TOW 参考页数 613	TOW	时间	功能	如果实时时钟的值位于"周时间"设置范围内，则开启输出。
TP 参考页数 605	TP	时间	功能块	脉冲计时。在上升沿时，将内部计时器增加至指定值。
TRIMPOT_READ 参考页数 364	TRIMPOT_READ	输入/输出	功能块	从特定微调电位中读取微调电位计数值。
TRUNC 参考页数 95	TRN	算术	功能	截断实型值，只保留整数。
TTABLE 参考页数 146	TTABLE	布尔	功能	根据输入组合提供输出的值。
UIC 参考页数 369	UIC	中断	功能	清除选定用户中断的丢失位。
UID 参考页数 370	UID	中断	功能	禁用特定用户中断。
UIE 参考页数 372	UIE	中断	功能	启用特定用户输入。

指令	映射指令（Logix 主题）	类别	类型	描述
UIF 参考页数 374	UIF	中断	功能	刷新或删除挂起的用户输入。
XOR 参考页数 143	XOR	布尔运算	运算符	两个值的布尔异或。
XOR_MASK 参考页数 134	XOR_MASK	二进制运算	功能	整数异或位到位掩码，返回反转的位值。

梯形图 (LD) 语言

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 提供 LD 语言编辑器，仅支持 Connected Components Workbench 软件随附的元素 参考页数 31和指令。

梯形图 (LD) 程序

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 语言通过程序图表（像继电器梯形图布线图那样组织）中的图形符号介绍对布尔数据的测试和修改。

Connected Components Workbench 提供梯形图语言编辑器 参考页数 50，仅支持 Connected Components Workbench 软件随附的元素和指令。

LD 程序开发环境

用于梯形图 (LD) 程序的语言编辑器，您可以在其中开发 LD 程序组织单元 (POU)。

下图显示了 LD 程序开发环境的主要区域。



编号	名称	描述
1	指令工具栏 参考页数 30	快速选择指令元素并将其放置在 LD 图形编辑器中，或者单击添加到 LD 文本编辑器中。
2	LD 文本编辑器	使用 ASCII 指令助记符编辑逻辑。
3	LD 图形编辑器	使用图形指令元素编辑逻辑。
4	LD 工具箱	将元素添加到 LD 图形编辑器。

指令工具栏 (LD)

指令工具栏是语言编辑器窗格中的辅助窗格的通俗名称，其功能类似于工具栏，用于向语言编辑器工作区中添加指令等语言元素 参考页数 30。它是对常规工作台工具箱的补充。



项目	描述
选项卡	按类别列出指令元素。单击类别可查看该类别中的指令。也可以使用箭头键选择不同的类别。单击标签左侧的箭头可滚动浏览类别。单击标签右侧的向下箭头可查看所有类别的列表。
指令	列出对应于所选选项卡类别的指令。类别中还可能有一些基本指令元素，这些指令元素固定在列表开头处。单击指令左侧的箭头可滚动浏览各指令。
搜索	在输入名称或描述关键字后立即按类别筛选指令。单击类别名称可查看针对该类别筛选的指令。
收藏夹	自定义可以快速找到并添加到程序的指令的列表。

从指令工具栏添加指令元素

使用指令工具栏 参考页数 30 来搜索、导航指令并将指令添加到梯形图 (Ladder Diagram, LD) 程序的语言编辑器工作区。

要将指令元素从指令工具栏添加到语言编辑器

1. 单击其中包含您希望添加的指令的类别选项卡。可以使用箭头键导航选项卡，并通过拖放操作重新排列其位置。
2. 通过拖放操作或单击来选择指令。
3. （可选）要快速查找指令，可在搜索字段中单击，然后键入内容以通过名称或关键字查找指令元素。要退出搜索并启用箭头键导航，请按“Esc”键。
4. （可选）右键单击指令并选择“添加到收藏夹”可将其添加到“收藏夹”选项卡，选择“从收藏夹移除”可将其移除。将自动保存收藏夹设置。

梯形图 (LD) 元素

梯形图 (LD) 元素是用于生成梯形图编程的组件。可以将下表中列出的所有元素添加到 Connected Components Workbench 中的梯形图编程。

元素	描述
梯级 参考页数 31	表示导致线圈被激活的一组回路元素。
指令块 (LD) 参考页数 37	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 34	两个或多个并行指令。
线圈 参考页数 37	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 43	表示输入或内部变量的值或函数。
返回 参考页数 48	表示功能块图输出的条件结束。
跳转 参考页数 48	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。

梯级

梯级为梯形图 (LD) 程序的图形组件，表示导致线圈被激活的一组回路元素。使用标签识别图中的梯级。这些标签与跳转一起控制梯形图的执行。注释是您可以在梯级上方添加用于存档的自由格式文本。

向 LD 程序添加梯级

梯级 参考页数 31 为梯形图 (Ladder Diagram, LD) 程序的图形组件，表示导致线圈被激活的一组回路元素。

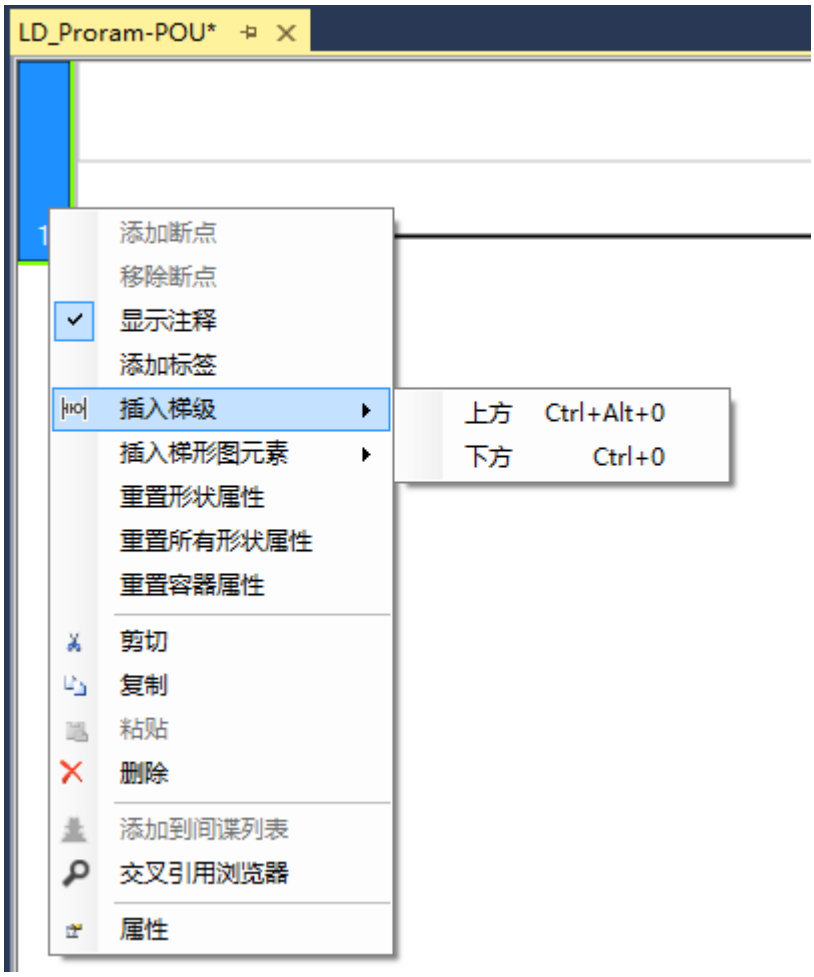
在 Connected Components Workbench 中，可以从下列对象添加梯级到梯形图 (Ladder Diagram, LD) 编程：

- 语言图 (LD) 语言编辑器
- 位于“工具”菜单上的“多语言编辑器”
- LD 工具箱

要将梯级元素添加到梯形图编程

1. 在 LD 语言编辑器中，右键单击现有梯级，然后：
   - 单击“复制”，然后单击“粘贴”以将梯级的副本插入到语言编辑器中。
   - 单击“插入梯级”，然后：
     - 单击“上方”以添加选定梯级上方的梯级。

- 单击“下方”以添加选定梯级下方的梯级。



- 在 LD 语言编辑器中选择梯级或元素，然后按：
  - “CTRL+ALT+o”- 添加选定梯级上方的梯级。
  - “CTRL+o”- 添加选定梯级下方的梯级。
- 在 LD 语言编辑器中选择梯级或元素，然后按：
  - 单击“工具”>“多语言编辑器”>“插入下方梯级”以添加选定梯级下方的梯级。
  - 单击“工具”>“多语言编辑器”>“插入上方梯级”以添加选定梯级上方的梯级。

1. （可选）打开 LD 工具箱以显示梯级元素，
   - 要在现有梯级下方插入梯级，请在 LD 语言编辑器中选择梯级，然后在 LD 工具箱中双击“梯级”。

     如果在 LD 工具箱中双击“梯级”之前未选择元素，梯级会插入到 LD 语言编辑器中最后一个梯级下方。
   - 选择“梯级”，然后将元素拖动到 LD 语言编辑器中。

     LD 语言编辑器中会出现加号 (+) 以显示有效的目标。松开鼠标键可添加元素。

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。



梯级标签

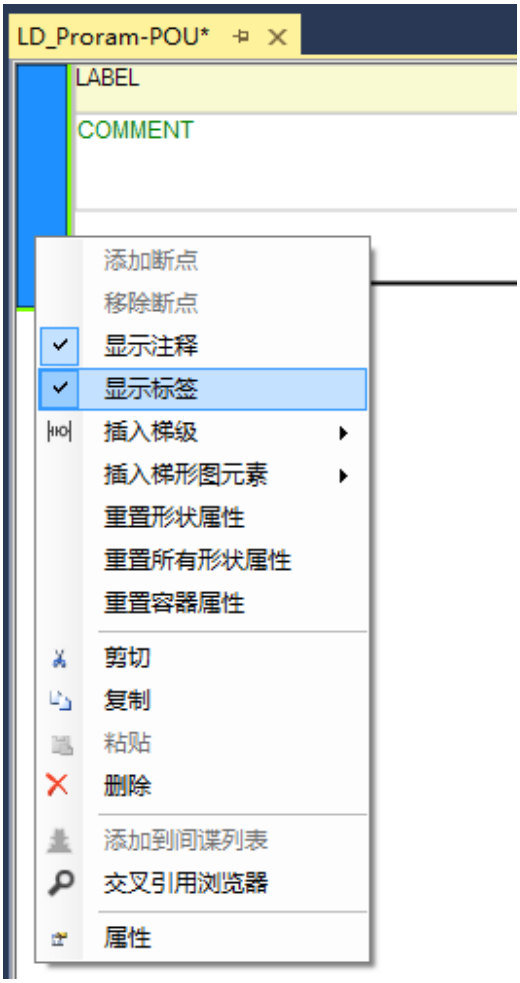
标签是梯形图 (LD) 语言编辑器中每个梯级的可选添加项。

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“\”）。

要在 LD 语言编辑器中为梯级添加标签

1. 要向梯级添加标签，请右键单击梯级以打开 LD 语言编辑器菜单，然后选择“显示标签”。

   梯级即会更新以包括标签，并且 LD 语言编辑器菜单会在“显示标签”旁显示复选标记。



2. 选择“标签”，然后键入描述。
3. （可选）要删除标签，请在 LD 语言编辑器菜单中单击“显示标签”。

梯级注释

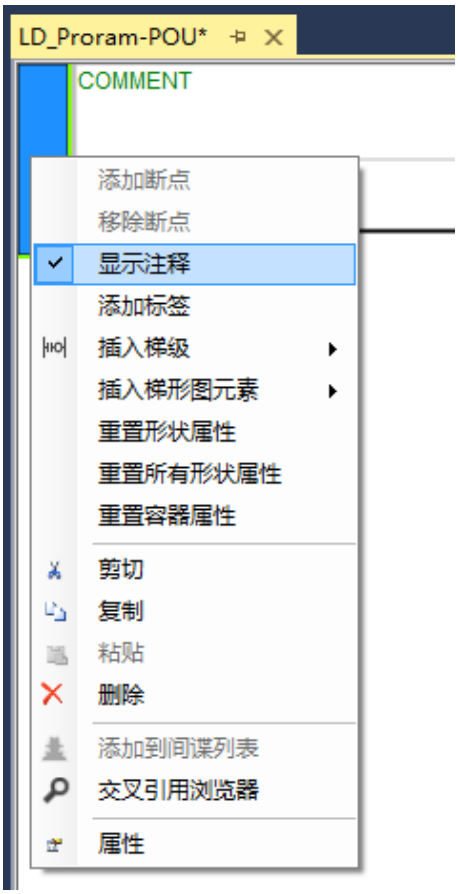
注释对于梯形图 (LD) 语言编辑器中每个梯级是可选的。默认情况下，在 LD 语言编辑器中添加梯级元素 参考页数 31时，会包含注释。

注释：

- 在梯级上方的空间中输入。
- 以富文本格式保存。
- 存储在控制器中。

要在 LD 语言编辑器中添加或删除梯级的注释

1. 要删除注释，请右键单击梯级以打开 LD 语言编辑器菜单。

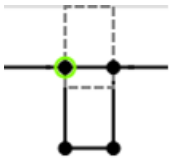


2. 在 LD 语言编辑器菜单中，单击“显示注释”。注释即会从梯级删除，并且 LD 语言编辑器菜单上“显示注释”旁的复选标记会被删除。
3. （可选）要将注释添加到梯级，请在 LD 语言编辑器菜单中单击“显示注释”。

分支

分支可创建连接的备用路由。可以使用梯形图 (Ladder Diagram, LD) 语言编辑器在梯级上将并联分支添加到元素 参考页数 35。

分支示例



向 LD 程序添加分支

分支 参考页数 34 是梯形图 (Ladder Diagram, LD) 编程的图形组件，其创建用于连接的备用路由，并且可以包括并联分支。

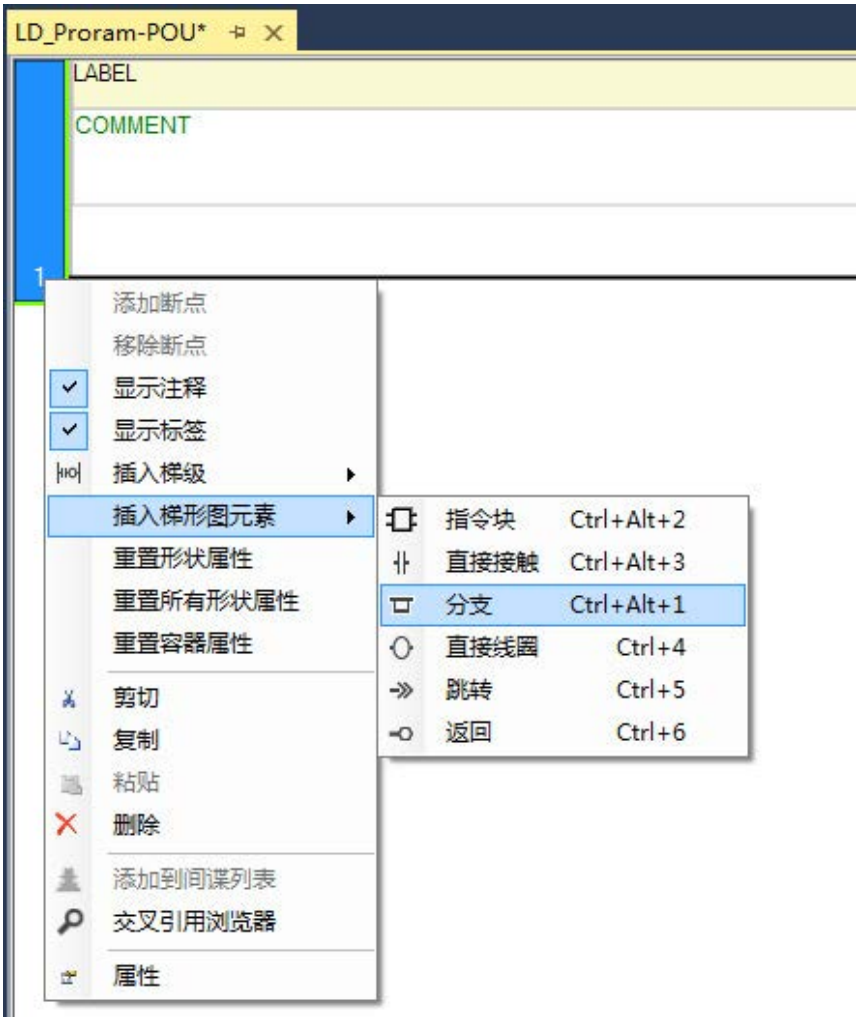
在 Connected Components Workbench 中，可以从下列对象添加分支到梯形图 (Ladder Diagram, LD) 编程：

- 梯形图 (LD) 语言编辑器
- 位于“工具”菜单上的“多语言编辑器”
- LD 工具箱
- 指令工具栏 (LD)

要将分支添加到梯形图编程

1. 在 LD 语言编辑器中，验证 LD 程序是否已为分支定义梯级。然后执行以下操作之一：
   - 在 LD 语言编辑器中，右键单击梯级或元素，选择“插入梯形图元素”，然后单击“分支”。

     如果“变量选择器”打开，选择变量或单击“确定”而不选择变量，以添加“分支”元素。



   - 在 LD 语言编辑器中选择梯级或元素，然后按：

“CTRL+ALT+1”以将分支添加到所选元素或梯级的左侧。
“CTRL+1”以将分支添加到所选元素的右侧。

- 在 LD 语言编辑器中选择梯级或元素，然后：

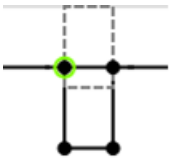
  单击“工具”>“多语言编辑器”>“在之前插入分支”以将分支添加到选定元素的左侧。
  单击“工具”>“多语言编辑器”>“在之后插入分支”以将分支添加到选定元素的右侧。

- 打开 LD 工具箱以显示分支元素，然后：

  双击分支元素以将其添加到 LD 语言编辑器。
  将分支元素拖动到 LD 语言编辑器，并将其放置在梯级上。
  LD 语言编辑器中会出现加号 (+) 以显示有效的目标。松开鼠标键可添加元素。

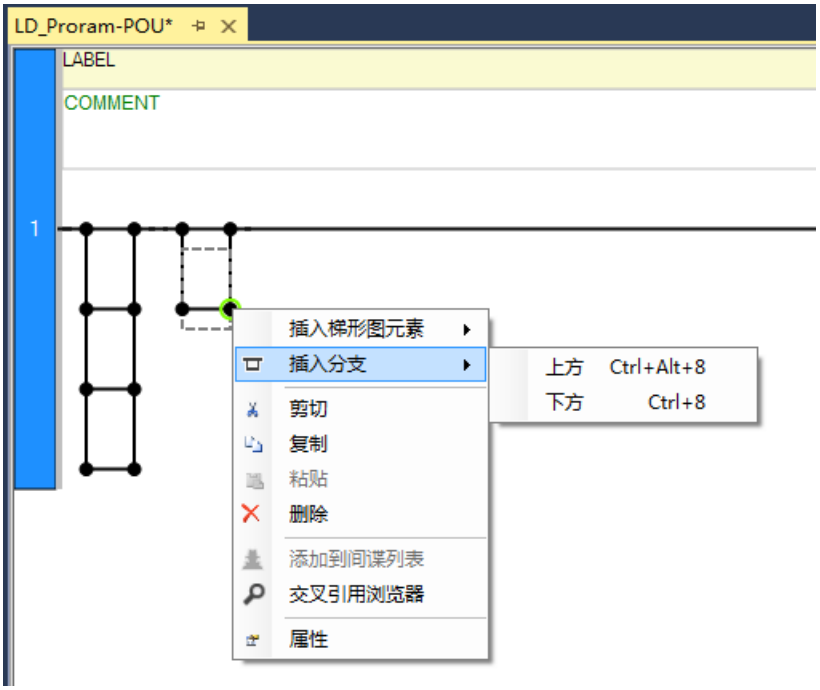
1. （可选）要插入并联分支：

   a. 在 LD 语言编辑器中，右键单击分支以打开 LD 语言编辑器菜单。

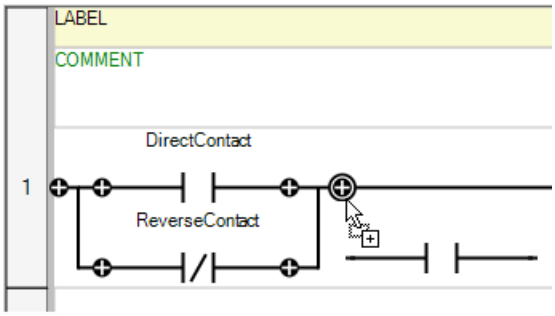
   b. 在 LD 语言编辑器菜单中，选择“插入分支”，然后单击：

   “上方”以将分支添加到选定分支之上。
   “下方”以将分支添加到选定分支之下。

2. （可选）要将分支元素移动到 LD 编程中的另一个位置，请选择此元素并将其拖动到 LD 编程中的新位置。

LD 语言编辑器中会出现加号 (+) 以显示有效的目标。松开鼠标按钮以将元素插入到目标位置。



指令块 (LD)

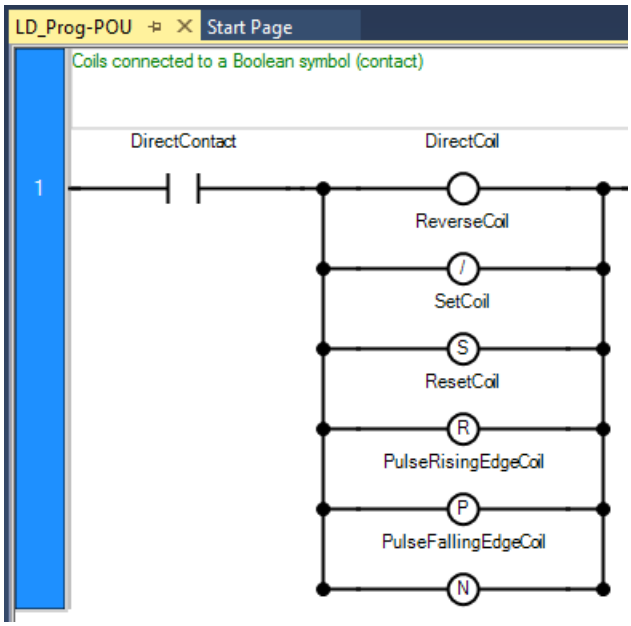
梯形图 (LD) 指令块元素是 LD 程序中符合 IEC 61131-3 的功能元素，可以是功能块、函数、用户定义的功能块、用户定义的函数或运算符。

线圈

线圈是梯形图 (Ladder Diagram, LD) 程序的图形组件，表示输出或内部变量的赋值。在 LD 程序中，线圈代表操作。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。线圈只能添加到在 LD 语言编辑器中定义的梯级。在将线圈添加到梯级后，可以修改线圈定义。

以下示例显示可用于梯形图编程的线圈单元类型。

示例：线圈



添加线圈单元

线圈是梯形图 (LD) 程序的图形组件，代表执行输出的操作或内部变量的操作。

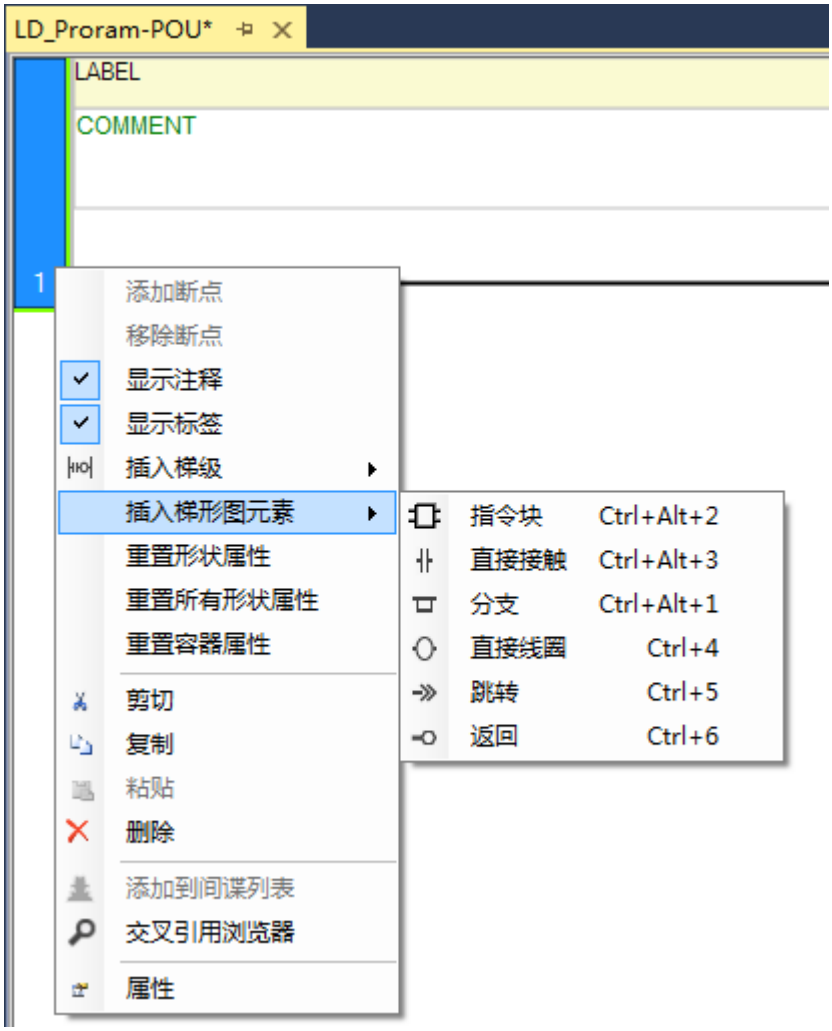
在 Connected Components Workbench 中，可以从下列对象添加线圈到梯形图 (Ladder Diagram, LD) 程序：

- 梯形图 (LD) 语言编辑器
- 位于“工具”菜单上的“多语言编辑器”
- LD 工具箱

- 指令工具栏 (LD)

1. 要将线圈元素添加到梯形图编程：在 LD 语言编辑器中，验证 LD 程序是否已为线圈定义梯级。然后执行以下操作之一：
   - 右键单击 LD 语言编辑器中的梯级或元素，选择“插入梯形图元素”，然后单击“正向线圈”。

     如果“变量选择器”打开，选择变量或单击“确定”而不选择变量，以添加“正向线圈”元素。

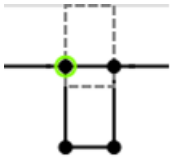


- 在 LD 语言编辑器中选择梯级或元素，然后按“CTRL+4”将“正向线圈”添加到梯级的右侧。
- 在 LD 语言编辑器中选择梯级或元素，然后单击“工具”>“多语言编辑器”>“插入线圈”以将线圈添加到梯级的右侧。
- 打开 LD 工具箱以显示线圈单元（正向线圈、反向线圈、置位线圈、复位线圈、脉冲上升沿线圈、脉冲下降沿线圈），然后：

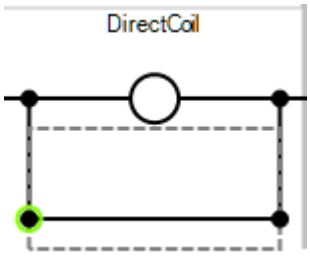
  双击线圈单元以将其添加到 LD 语言编辑器。
  将线圈单元拖动到 LD 语言编辑器中，并将其放置在梯级上。
  LD 语言编辑器中会出现加号 (+) 以显示有效的目标。松开鼠标键可添加元素。

1. （可选）要插入并联线圈

a. 在 LD 语言编辑器中，验证 LD 程序是否有已定义的分支，然后右键单击该分支 参考页数 34 的最高级别，以打开 LD 语言编辑器菜单。

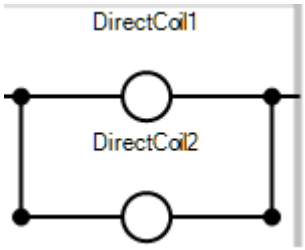


b. 在 LD 语言编辑器菜单中，选择“插入梯形图元素”，然后单击“正向线圈”。元素即会插入该分支的最高级别。



c. 右键单击该分支的最低级别，以打开 LD 语言编辑器菜单。

d. 在 LD 语言编辑器菜单中，选择“插入梯形图元素”，然后单击“正向线圈”。元素即会插入该分支的最低级别。



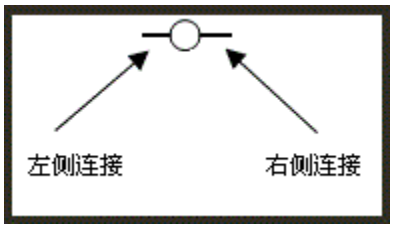
2. （可选）要更改线圈的类型，请在 LD 语言编辑器中选择线圈，然后按“空格键”直到线圈类型显示在语言编辑器中。

   每按一次空格键，线圈类型会从正向更改为反向、置位、复位、脉冲上升沿、脉冲下降沿。

直接线圈

线圈 参考页数 37 是梯形图 (Ladder Diagram, LD) 程序的图形组件，表示输出或内部变量的赋值。

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。

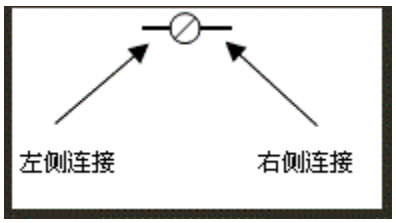
直接线圈示例



反向线圈

线圈 参考页数 37是梯形图 (Ladder Diagram, LD) 程序的图形组件，表示输出或内部变量的赋值。

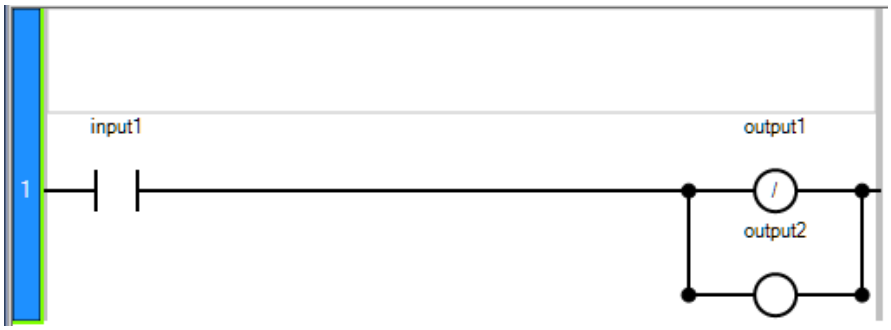
反向线圈元素根据连接线状态的布尔非运算结果支持布尔输出。



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。

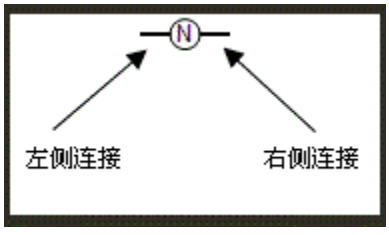
反向线圈示例



脉冲下降沿线圈

线圈 参考页数 37是梯形图 (Ladder Diagram, LD) 程序的图形组件，表示输出或内部变量的赋值。

脉冲下降沿 参考页数 41（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。

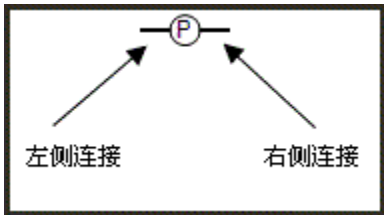
脉冲下降沿线圈示例



脉冲上升沿线圈

线圈 参考页数 37是梯形图（Ladder Diagram, LD）程序的图形组件，表示输出或内部变量的赋值。

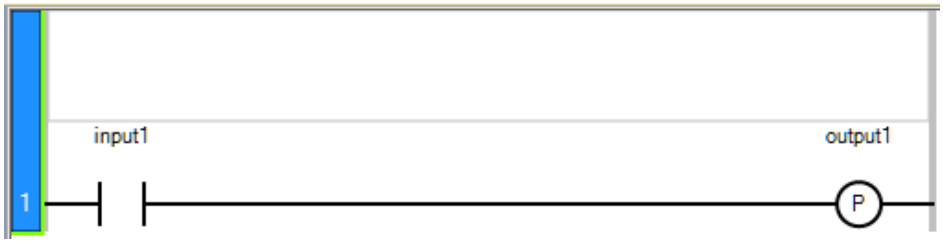
脉冲上升沿（或正极）的线圈支持连接线布尔状态的布尔输出。



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。

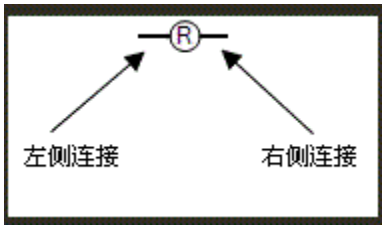
脉冲上升沿线圈示例



置位线圈

线圈是梯形图（Ladder Diagram, LD）程序的图形组件，表示输出或内部变量的赋值。在 LD 程序中，线圈代表操作。

置位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。

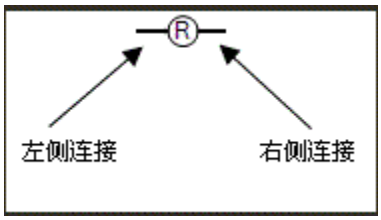
置位线圈示例



复位线圈

线圈 参考页数 37 是梯形图（Ladder Diagram, LD）程序的图形组件，表示输出或内部变量的赋值。

复位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。

复位线圈示例

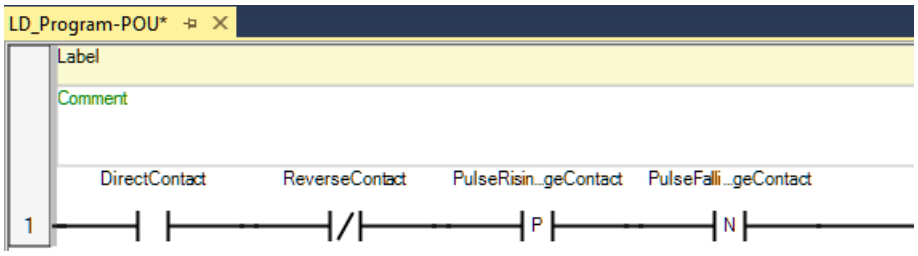


触点

触点是梯形图 (Ladder Diagram, LD) 程序的图形组件。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。

以下示例显示可用于梯形图编程的触点元素类型。

示例：接触



向 LD 程序添加触点

接触是梯形图 (LD) 程序的图形组件。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。

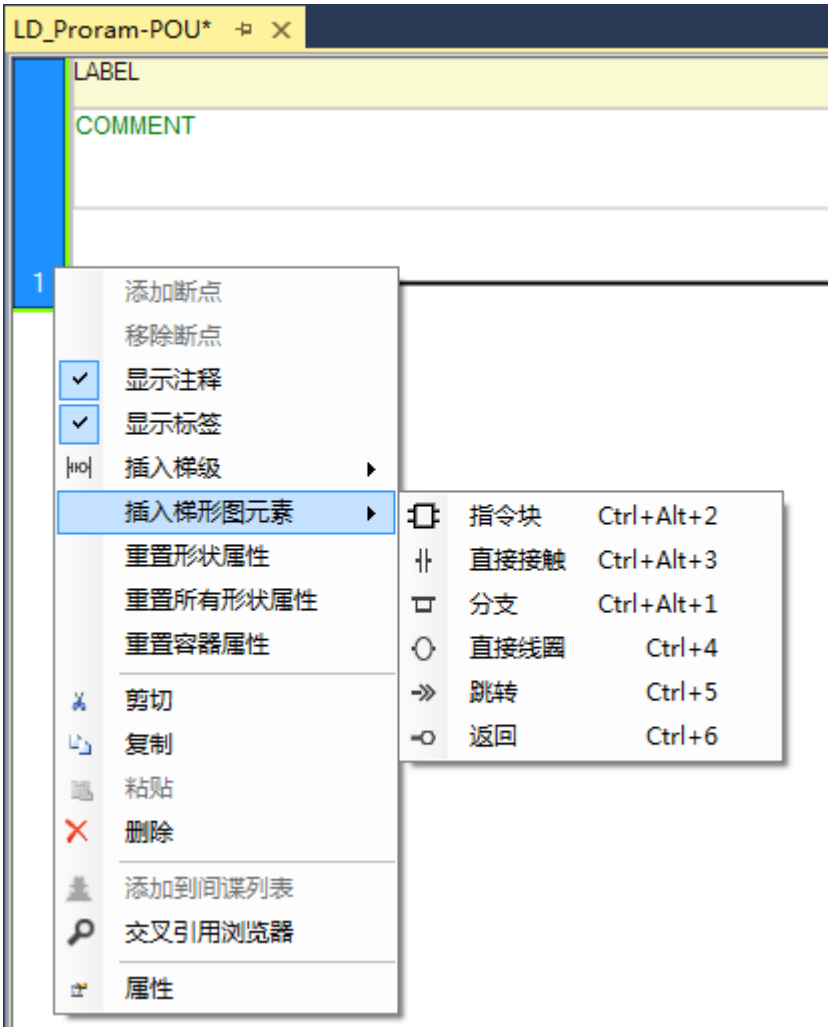
在 Connected Components Workbench 中，可以从下列对象添加触点到梯形图 (Ladder Diagram, LD) 编程：

- 梯形图 (LD) 语言编辑器
- 位于“工具”菜单上的“多语言编辑器”
- LD 工具箱
- 指令工具栏 (LD)

要将触点元素添加到梯形图编程

1. 在 LD 语言编辑器中，验证 LD 程序是否已为触点定义梯级。然后执行以下操作之一：
   - 右键单击 LD 语言编辑器中的梯级或元素，选择“插入梯形图元素”，然后单击“正向触点”。

如果“变量选择器”打开，选择变量或单击“确定”而不选择变量，以添加“正向触点”元素。



- 在 LD 语言编辑器中选择梯级或元素，然后按：

  “CTRL+ALT+3”以将“正向触点”元素添加到所选的元素或梯级的左侧。
  “CTRL+3”以将“正向触点”元素添加到所选的元素或梯级的右侧。

- 在 LD 语言编辑器中选择梯级或元素，然后：

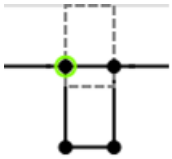
  单击“工具”>“多语言编辑器”>“在之前插入触点”以将触点添加到选定元素或梯级的左侧。
  单击“工具”>“多语言编辑器”>“在之后插入触点”以将触点添加到选定元素或梯级的右侧。

- 打开 LD 工具箱以显示触点元素（正向触点、反向触点、脉冲上升沿触点、脉冲下降沿触点），然后：

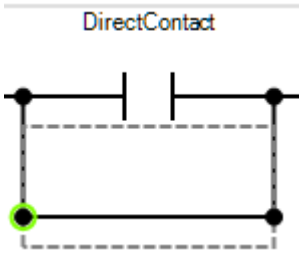
  双击触点元素以将其添加到 LD 语言编辑器。
  将触点元素拖动到 LD 语言编辑器中，并将其放置在梯级上。
  LD 语言编辑器中会出现加号 (+) 以显示有效的目标。松开鼠标键可添加元素。

1. （可选）要插入并联触点：

a. 在 LD 语言编辑器中，验证 LD 程序是否有已定义的分支 参考页数 34，然后右键单击该分支的最高级别，以打开 LD 语言编辑器菜单。

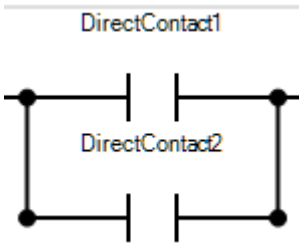


b. 在 LD 语言编辑器菜单中，选择“插入梯形图元素”，然后单击“正向触点”。元素即会插入该分支的最高级别。



c. 右键单击该分支的最低级别，以打开 LD 语言编辑器菜单。

d. 在 LD 语言编辑器菜单中，选择“插入梯形图元素”，然后单击“正向触点”。元素即会插入该分支的最低级别。



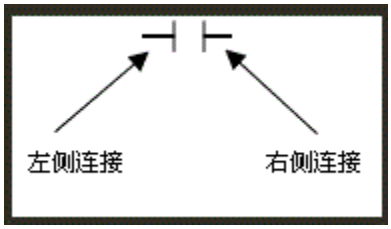
2. （可选）要更改触点的类型，请在语言编辑器中选择触点，然后按“空格键”直到触点类型显示在语言编辑器中。

   每按一次空格键，触点类型会从正向更改为反向、脉冲上升沿、脉冲下降沿。

直接接触

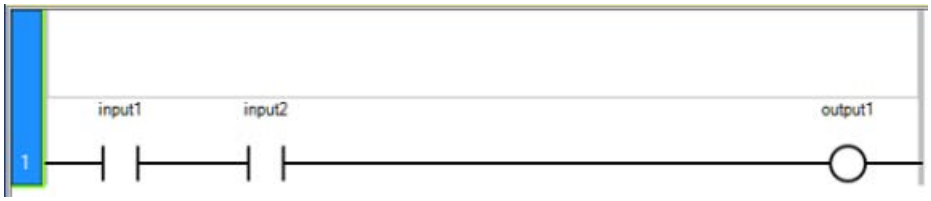
触点 参考页数 43 是梯形图 (Ladder Diagram, LD) 程序的图形组件。

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。

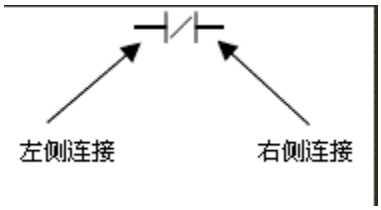
直接接触示例



反向接触

触点 参考页数 43 是梯形图 (Ladder Diagram, LD) 程序的图形组件。

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。

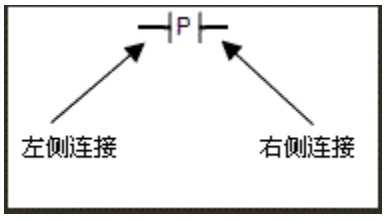
反向接触示例



脉冲上升沿触点

触点 参考页数 43 是梯形图 (Ladder Diagram, LD) 程序的图形组件。

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”。

脉冲上升沿触点示例



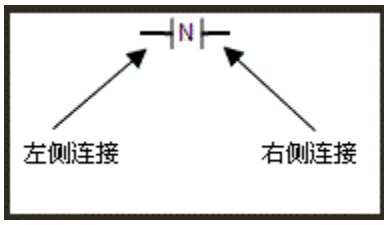
建议：限制使用具有边沿触点的输出变量

建议不使用具有脉冲上升沿触点 参考页数 46（正值）或脉冲下降沿触点 参考页数 47（负值）的输出或变量。这些触点用于梯形图中的物理输入。要检测变量或输出的沿，建议使用 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。

脉冲下降沿触点

触点是梯形图 (Ladder Diagram, LD) 程序的图形组件。

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”。

脉冲下降沿触点示例



建议：限制使用具有边沿触点的输出变量

建议不使用具有脉冲上升沿触点 参考页数 46（正值）或脉冲下降沿触点 参考页数 47（负值）的输出或变量。这些触点用于梯形图中的物理输入。要检测变量或输出的沿，建议使用 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。

返回

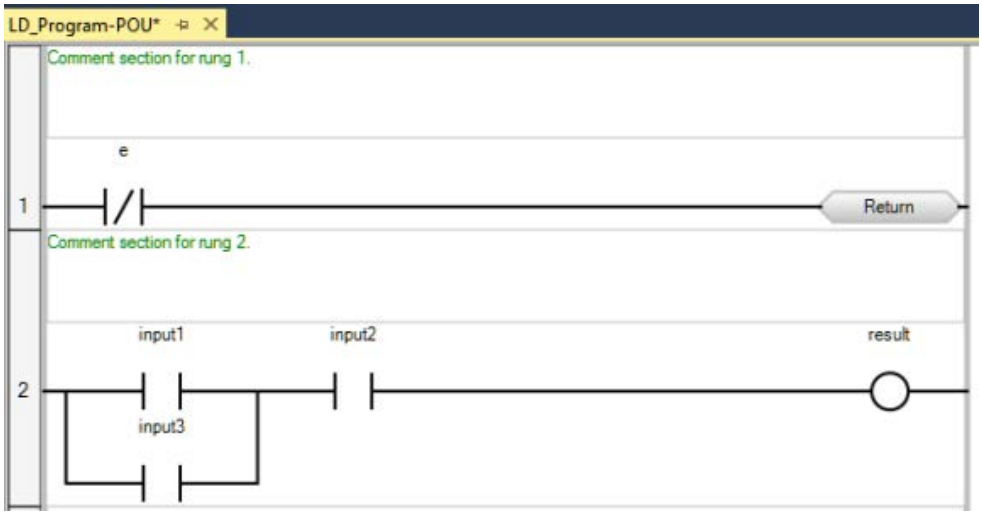
返回是表示梯形图 (LD) 程序的条件性结束的输出。

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。

返回示例



要在梯形图编程中插入返回

请执行以下操作之一：

- 在 LD 语言编辑器中，右键单击梯级或元素，选择“插入梯形图元素”，然后单击“返回”。
- 在 LD 语言编辑器中，选择梯级或元素，然后按“CTRL + +6”。
- 在 LD 语言编辑器中，选择梯级或元素，然后单击“工具”>“多语言编辑器”>“插入返回”。
- 打开 LD 工具箱以显示返回元素，然后：

  双击返回元素以将其添加到 LD 语言编辑器。
  将返回元素拖动到 LD 语言编辑器，并将其放置在梯级上。
  LD 语言编辑器中会出现加号 (+) 以显示有效的目标。松开鼠标键可添加元素。

跳转

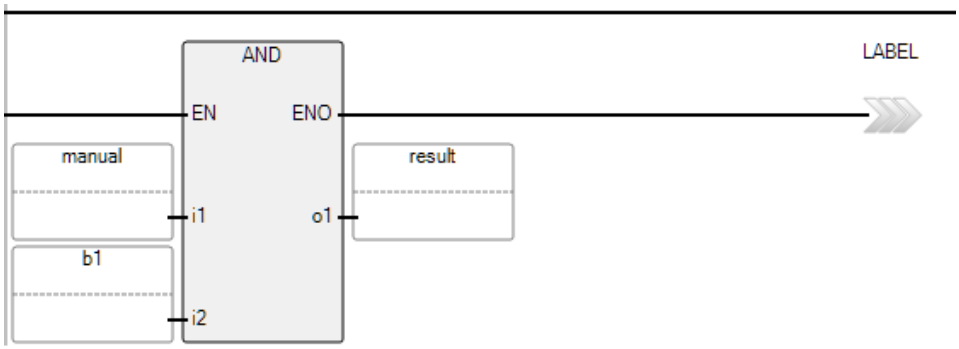
跳转是控制梯形图 (LD) 程序执行的条件元素或无条件元素。

跳转符号

下面的符号表示跳转到标签：

>>LABEL - 跳转至标签（标签名称为“LABEL”）

跳转示例



要插入跳转

请执行以下操作之一：

- 在 LD 语言编辑器中右键单击梯级或元素，选择“插入梯形图元素”，然后单击“跳转”。
- 在 LD 语言编辑器中，选择梯级或元素，然后按“CTRL + +5”。
- 在 LD 语言编辑器中，选择梯级或元素，然后单击“工具”>“多语言编辑器”>“插入跳转”。
- 打开 LD 工具箱以显示跳转元素，然后：

  双击跳转元素以将其添加到 LD 语言编辑器。
  将跳转元素拖动到 LD 语言编辑器，并将其放置在梯级上。
  LD 语言编辑器中会出现加号 (+) 以显示有效的目标。松开鼠标键可添加元素。

LD 程序中的指令块

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，实例名称显示在功能块名称的上方。

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

在梯形图 (Ladder Diagram, LD) 编程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。
- 并会将后续元素插入到梯级上选定项的右侧。
- 不能将元素插入到线圈返回或跳转的右侧。

将元素添加到梯形图编程的不同方法：

- **LD 梯形图编辑器**
  - 添加元素、删除元素，以及复制和粘贴元素。
  - 使用 LD 键盘快捷键 参考页数 52 添加元素。
- **LD 文本编辑器**
  - 添加、修改和删除元素。
  - 从 RSLogix 500 和 RSLogix 5000 复制并粘贴元素/将元素复制并粘贴到 RSLogix 500 和 RSLogix 5000。
- 位于“工具”菜单上的“多语言编辑器”
  - 添加元素。
  - 导出 LD 程序的映像。
  - 启用或禁用“变量选择器”和“指令块选择器”的自动打开。
- **LD 工具箱**
  - 添加元素。
- **指令工具栏**
  - 添加元素。

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

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：
   - 输入新的变量名称，双击该变量以打开“变量选择器”。按 Enter 确认新变量。
   - 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

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中选择一个不同的变量。

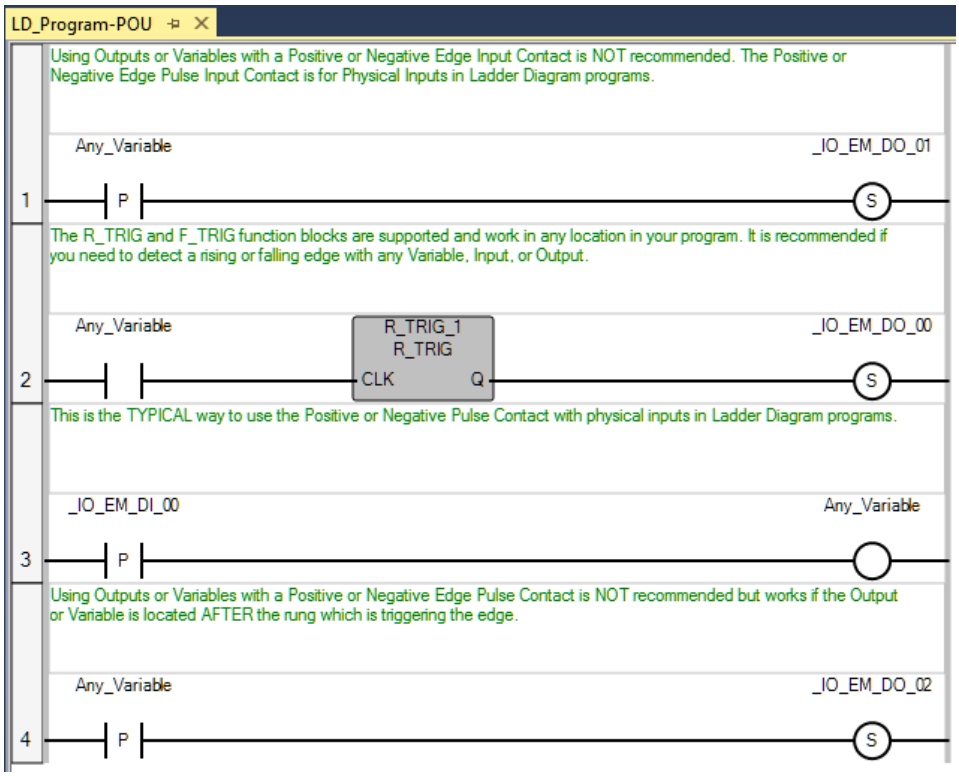
3. 单击现有变量，然后在提供的文本框中键入常量值。

梯形图 (LD) 程序示例

下例是梯形图 (LD) 程序。

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。

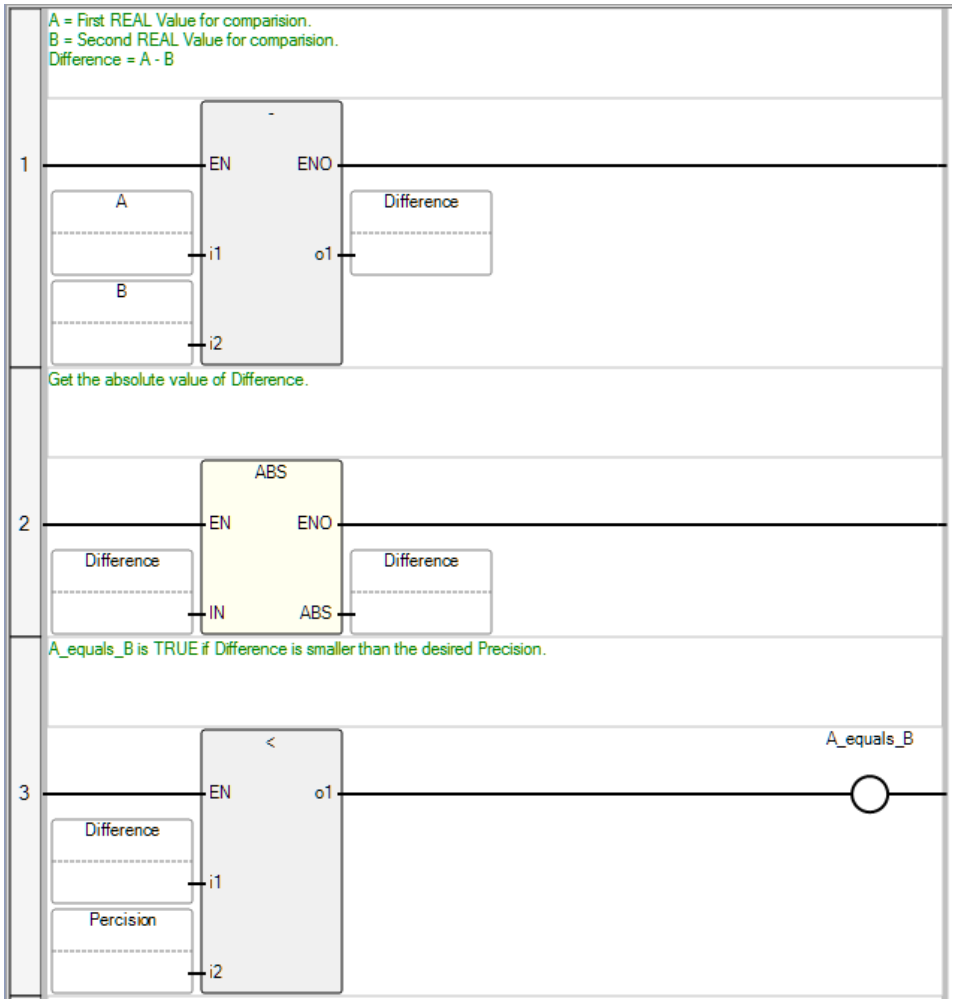


示例：使用 Subtraction (-)、ABS 和 Less than (<) 比较实型值

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

下列键盘快捷键可用于梯形图语言 参考页数 29。

快捷键	描述
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+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后。
Ctrl+6	将返回插入到选定元素之后。
Ctrl+8	将分支插入到选定分支之上。
Ctrl+Alt+8	将分支插入到选定分支之下。
删除	删除选定的梯级或元素。
Enter	选定梯级后，按 Enter 键会选择梯级的第一个元素。如果没有梯级元素，将不会发生任何操作。

快捷键	描述
空格键	选定线圈或触点后，按空格键会更改触点或线圈类型。
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+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	移动到选定梯级的第一个元素，如果没有梯级元素，将不会发生任何操作。
结束	移动到选定梯级的最后一个元素，如果没有梯级元素，将不会发生任何操作。
Ctrl+Home	移动到第一个梯级的第一个元素，如果没有梯级元素，将选中第一个梯级。
Ctrl+End	移动到最后一个梯级的最后一个元素，如果没有梯级元素，将选中最后一个梯级。
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+J	移动到匹配的括号。
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+鼠标左键单击	选择多个梯级。分别单击每个梯级。
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。

快捷键	描述
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+D	如果选定梯级或梯级的一个元素，在按 Ctrl+D 后，用户可编辑梯级注释。
Ctrl+R	启用或禁用自动选择器调用。 默认情况下，元素添加到梯形图编程时，指令块选择器或变量选择器对话框会打开。
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。
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。
Esc	取消选择选定文本。
Insert	在覆盖/插入键入模式之间切换。

[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。

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	描述
LIM_ALRM 参考页数 55	上限和下限实值的滞后。

LIM_ALRM（限制警报）

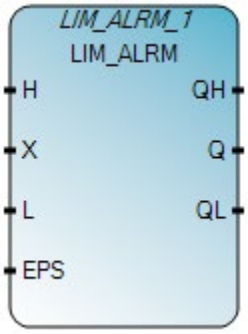
LIM_ALRM 是关于上限和下限实型值滞后的警报。

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。

过程警报是当控制器收到并处理故障时产生的警报。当模块超过为每个通道配置的上限或下限时，过程级别警报将发出提醒。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

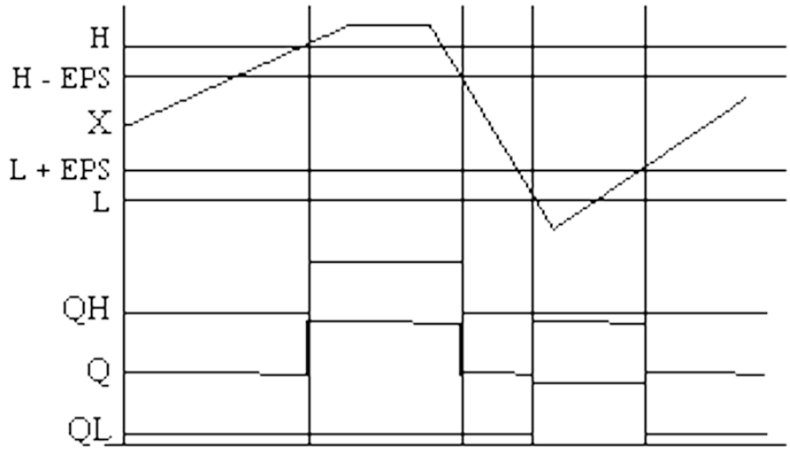
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



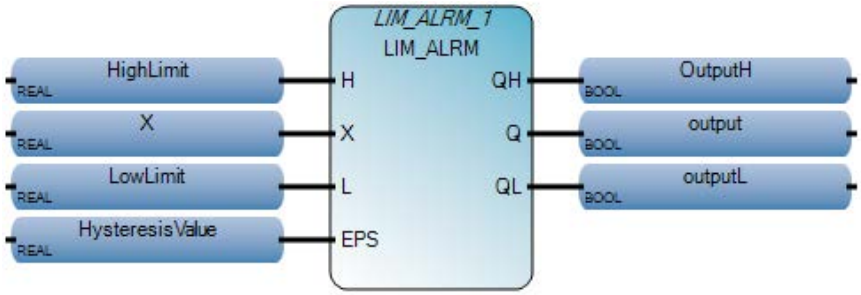
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	当为 TRUE 时，启用指令块。 TRUE - 执行当前 LIM_ALRM 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
H	输入	REAL	上限值。
X	输入	REAL	输入为任何实型值。
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。
ENO	输出	BOOL	启用输出。 仅适用于梯形图编程。

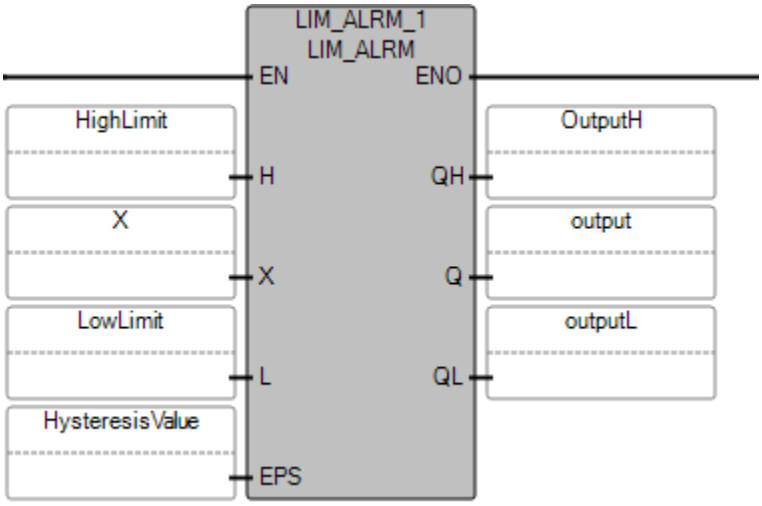
LIM_ALRM 时间图示例



LIM_ALRM 功能块图示例



LIM_ALRM 梯形图示例



LIM_ALRM 结构化文本示例

```
LIM_ALRM_1(
```

void **LIM_ALRM_1**(REAL **H**, REAL **X**, REAL **L**, REAL **EPS**)
类型： LIM_ALRM, 带迟滞的高/低限制警报

```
1  HighLimit := 10.0;
2  X := 15.0;
3  LowLimit := 5.0;
4  HysteresisValue := 2.0;
5  LIM_ALRM_1(HighLimit, X, LowLimit, HysteresisValue);
6  OutputH := LIM_ALRM_1.QH;
7  OutputL := LIM_ALRM_1.QL;
8  output := LIM_ALRM_1.Q;
```

结果



算术指令

使用算术指令执行数学计算。

功能	描述
ABS 参考页数 59	返回实型值的绝对值。
ACOS 参考页数 61	计算实型值的反余弦。
ACOS_LREAL 参考页数 62	计算长实型值的反余弦。
Addition 参考页数 64	添加两个或多个整型、实型、时间或字符串值。
ASIN 参考页数 65	计算实型值的反正弦。
ASIN_LREAL 参考页数 66	计算长实型值的反正弦。
ATAN 参考页数 68	计算实型值的反正切。
ATAN_LREAL 参考页数 69	计算长实型值的反正切。
COS 参考页数 71	计算实型值的余弦。
COS_LREAL 参考页数 72	计算长实型值的余弦。
Division 参考页数 74	两个整型值或实型值相除。
EXPT 参考页数 75	计算增加至整型指数的幂的基数的实型值。
LOG 参考页数 77	计算实型值的对数（以 10 为底）。
MOD 参考页数 78	对整型值执行模计算。
MOV 参考页数 80	将输入值复制到输出。
Multiplication 参考页数 81	乘以两个或多个整型或实型值。
Neg 参考页数 83	将值转换为负值。
POW 参考页数 83	计算增加至实型指数的幂的实型数的值。
RAND 参考页数 85	从定义的范围计算随机整数值。
SIN 参考页数 87	计算实型值的正弦。
SIN_LREAL 参考页数 88	计算长实型值的正弦。
SQRT 参考页数 89	计算实型值的平方根。
Subtraction 参考页数 91	用一个整型、实型或时间值减去另一个整型、实型或时间值。
TAN 参考页数 92	计算实型值的正切。
TAN_LREAL 参考页数 94	计算长实型值的正切。
TRUNC 参考页数 95	截断实型值，只保留整数。

ABS（绝对值）

返回实型值的绝对值（正值）。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



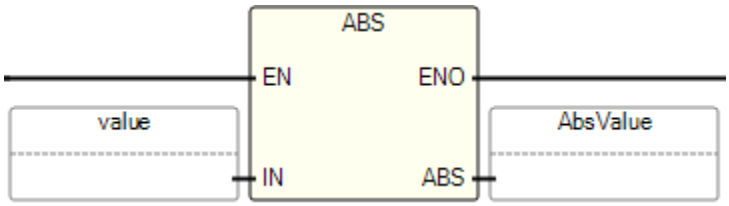
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前绝对计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
IN	输入	REAL	任何有符号实型值。
ENO	输出	BOOL	启用"输出"。 适用于梯形图编程。
ABS	输出	REAL	绝对值（始终为正数）。

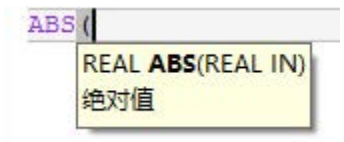
ABS 功能块图示例



ABS 梯形图示例



ABS 结构化文本图示例




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

(* 与之等效的 ST: *)

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

结果



ACOS（源的反余弦）

计算实型值的反余弦。输入值和输出值都以弧度表示。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



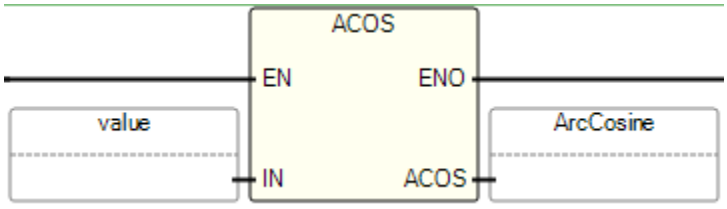
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前反余弦计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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”。 适用于梯形图编程。
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 功能块图示例



ACOS 梯形图示例



ACOS 结构化文本示例

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

```
ACOS(
```

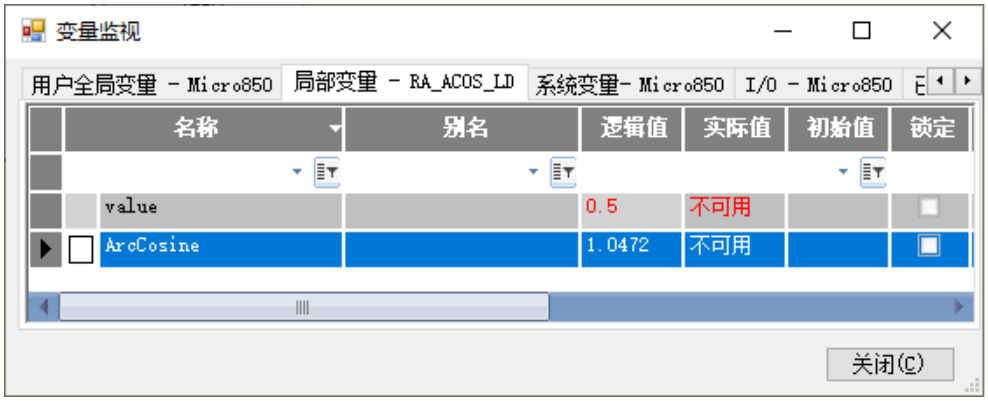
REAL **ACOS**(REAL IN)
反余弦

(* 与之等效的 ST：*)

cosine := COS (angle);

result := ACOS (cosine); (* 结果为角度 *)

结果

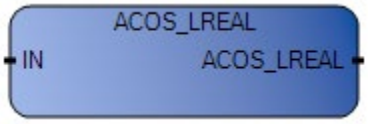


ACOS_LREAL（长实型反余弦）

计算长实型值的反余弦。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

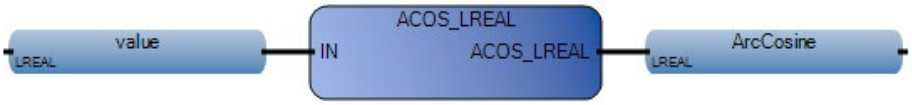
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



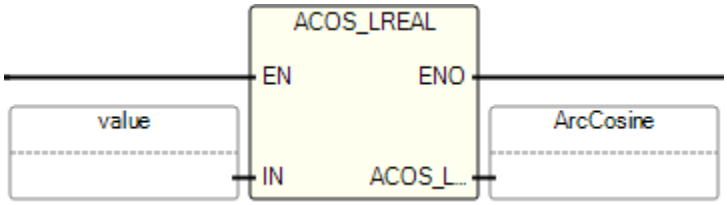
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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”。 适用于梯形图编程。
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 功能块图示例



ACOS_LREAL 梯形图示例



ACOS_LREAL 结构化文本示例



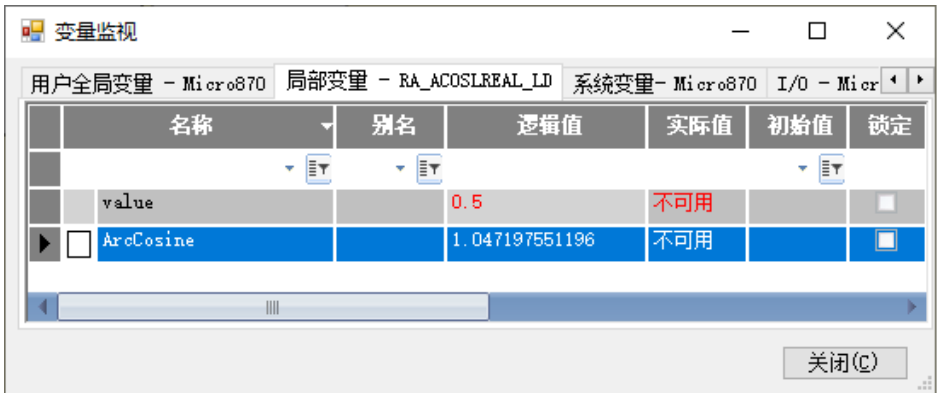

```
1  value := 0.5;
2  ArcCosine := ACOS_LREAL(value);
```

(* 与之等效的 ST: *)

cosine := COS_LREAL (angle);

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

结果



Addition

添加两个或多个整型、实型、时间或字符串值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前相加计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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。

参数	参数类型	数据类型	描述
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"。 仅适用于梯形图编程。

相加结构化文本示例

(* 与之等效的 ST：*)

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

ASIN（反正弦）

计算实型值的反正弦。输入值和输出值都以弧度表示。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



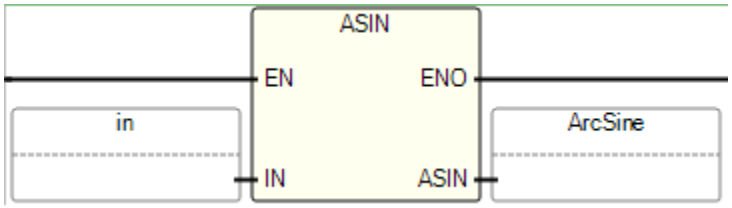
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前反正弦计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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"。 适用于梯形图编程。

ASIN 功能块图示例

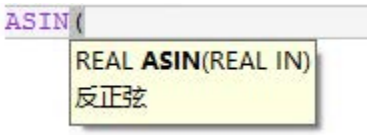


ASIN 梯形图示例



ASIN 结构化文本示例

```
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

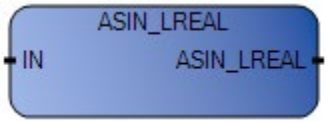


ASIN_LREAL（长实型反正弦）

计算长实型值的反正弦。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

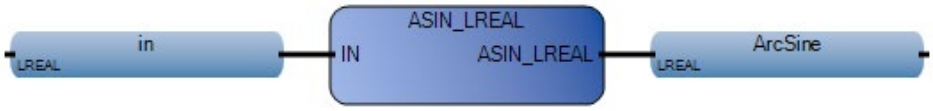
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



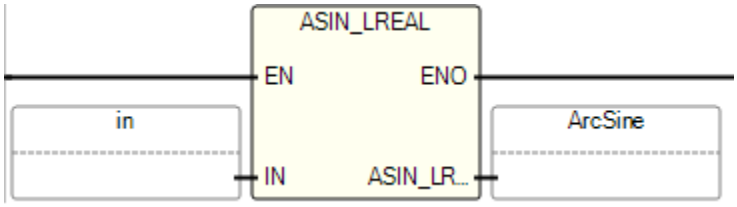
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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"。 适用于梯形图编程。

ASIN_LREAL 功能块图示例



ASIN_LREAL 梯形图示例



ASIN_LREAL 结构化文本示例

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

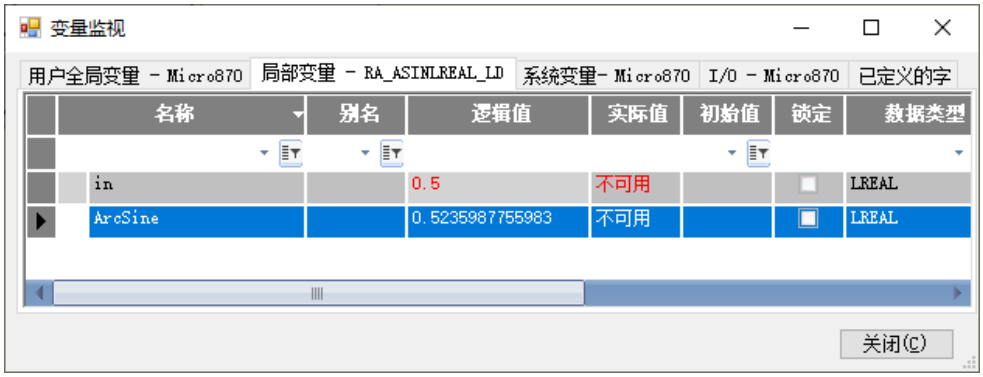


(* 与之等效的 ST： *)

sine := SIN_LREAL (angle);

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

结果

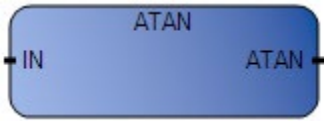


ATAN（反正切）

计算实型值的反正切。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

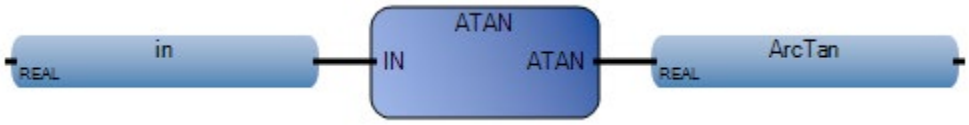
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



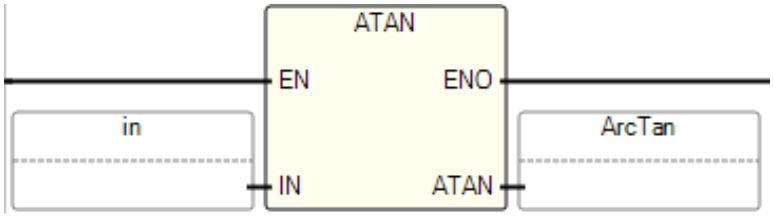
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前反正切计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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”。 适用于梯形图编程。

ATAN 功能块图示例

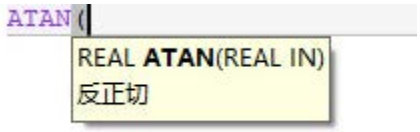


ATAN 梯形图示例



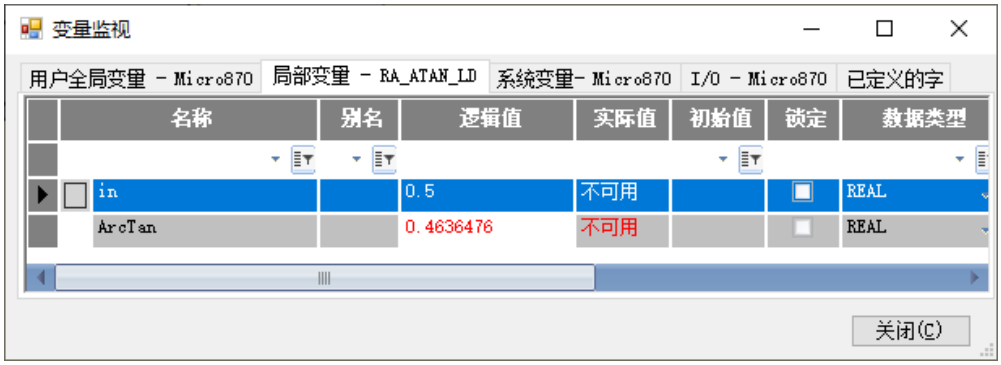
ATAN 结构化文本示例

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




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

结果

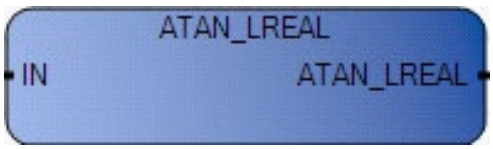


ATAN_LREAL（长实型反正切）

计算长实型值的反正切。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

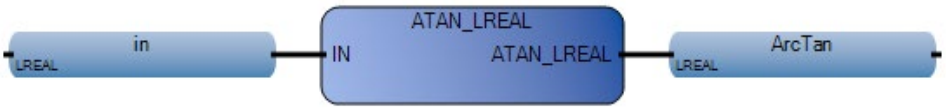
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



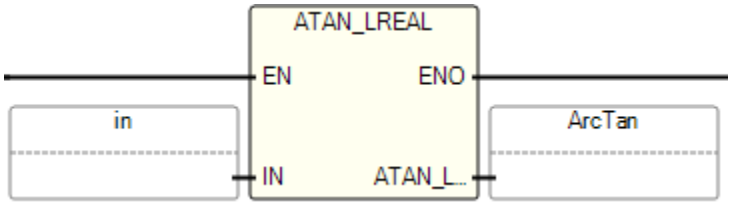
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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 ..+PI/2] 内）= 0.0。
ENO	输出	BOOL	启用"输出"。 适用于梯形图编程。

ATAN_LREAL 功能块图示例



ATAN_LREAL 梯形图示例



ATAN_LREAL 结构化文本示例

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

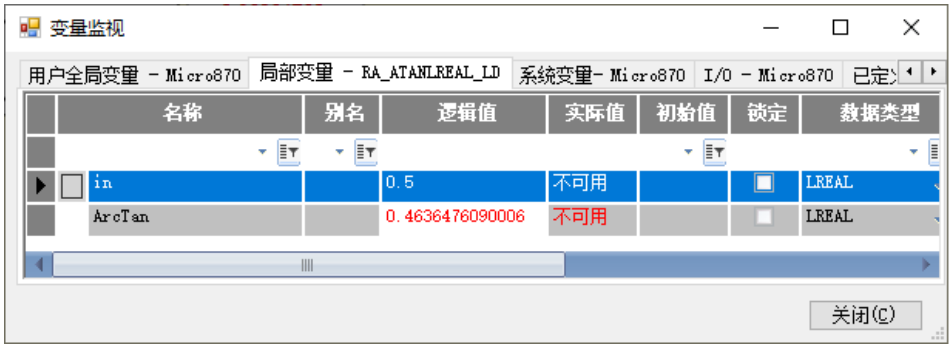


(* 与之等效的 ST：*)

tangent := TAN_LREAL (angle);

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

结果



COS（余弦）

计算实型值的余弦。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



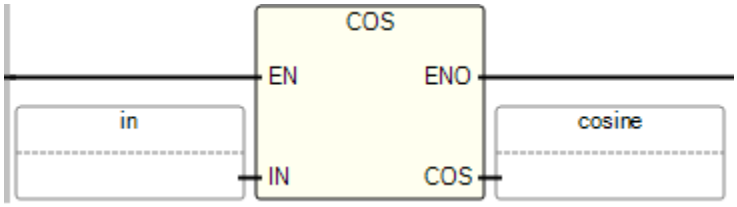
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前余弦计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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”。 适用于梯形图编程。

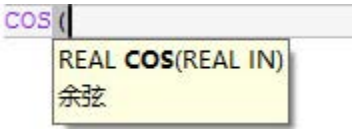
COS 功能块图示例



COS 梯形图示例



COS 结构化文本示例



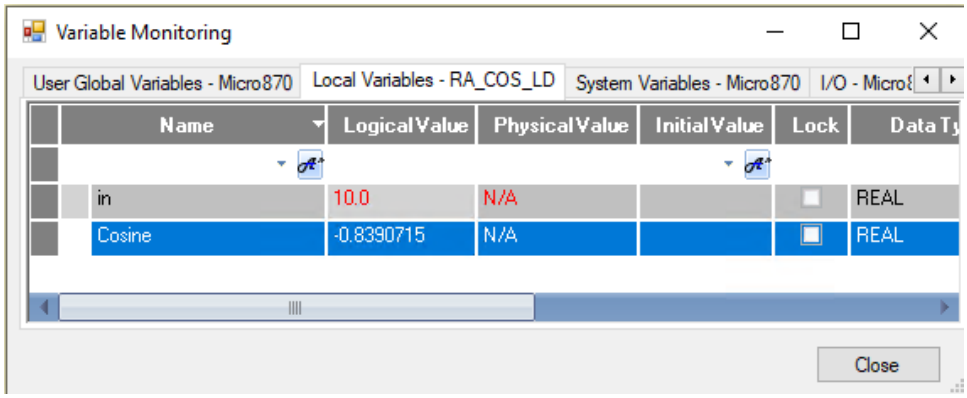

```
1  in := 10.0;
2  cosine := COS(in);
```

(* 与之等效的 ST: *)

cosine := COS (angle);

result := ACOS (cosine); (* 结果为角度 *)

结果

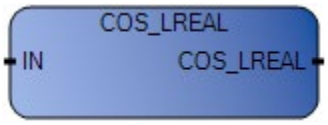


COS_LREAL（余弦长实型）

计算长实型值的余弦。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

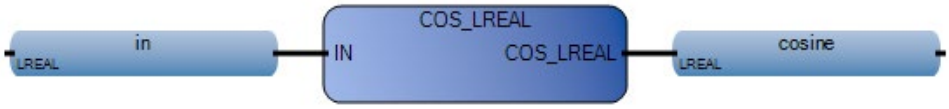
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



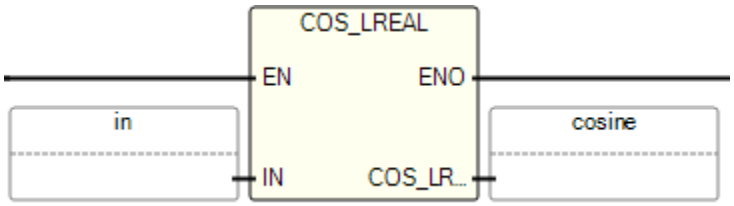
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前余弦计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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"。 适用于梯形图编程。

COS_LREAL 功能块图示例



COS_LREAL 梯形图示例



COS_LREAL 结构化文本示例




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

(* 与之等效的 ST： *)

cosine := COS_LREAL (angle);

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

结果



Division

用第一个整型或实型输入值除以第二个整型或实型输入值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前相除计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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。
i2	输入	SINT USINT BYTE	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。

参数	参数类型	数据类型	描述
		INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL	
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。 仅适用于梯形图编程。

Division 结构化文本示例

(* 与之等效的 ST：*)

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

EXPT（指数）

将 IN（基数）的值增加至 EXP（指数）的幂，并输出运算的实型结果。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



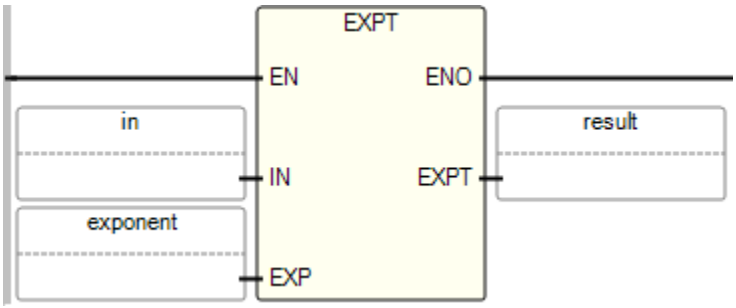
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前指数计算。 FALSE - 不执行任何计算。
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 的实型值与 EXP 的幂。
ENO	输出	BOOL	启用"输出"。 适用于梯形图编程。

EXPT 功能块图示例

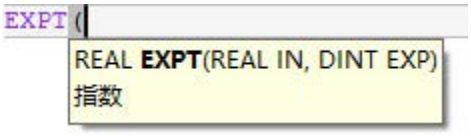


EXPT 梯形图示例



EXPT 结构化文本示例

```
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



LOG（对数底为 10）

计算实型值的对数（以 10 为底）。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



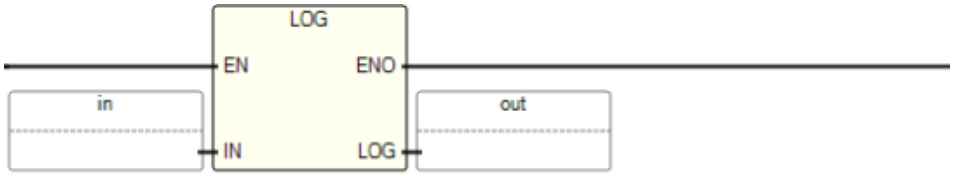
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前对数计算。 FALSE - 不执行任何计算。
IN	输入	REAL	必须大于零。
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”。 适用于梯形图编程。

LOG 功能块图示例



LOG 梯形图示例



LOG 结构化文本示例

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



(* 与之等效的 ST：*)

xpos := ABS (xval);

xlog := LOG (xpos);

结果



MOD（模）

将 IN 输入除以基输入，并将其余部分放置在 MOD 输出。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



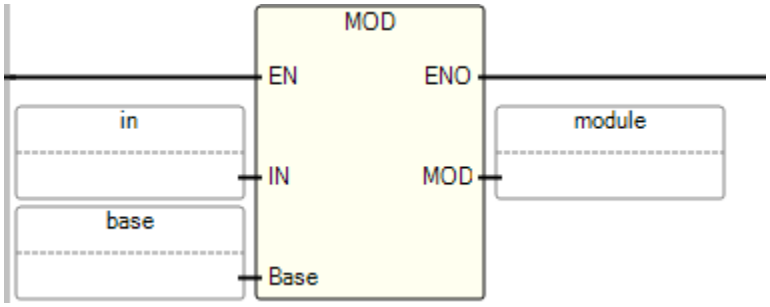
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行模块计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
IN	输入	DINT	任何有符号整型值。
Base	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"。 适用于梯形图编程。

MOD 功能块图示例



MOD 梯形图示例



MOD 结构化文本示例




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

(* 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



MOV（移动）

将输入 (i1) 值分配给输出 (o1)。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

对于结构化文本程序，使用 Equal (=) 运算符而不是 MOV。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行直接链接到输出的计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
i1	输入	BOOL DINT REAL TIME STRING SINT USINT INT UINT UDINT	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	描述
		LINT ULINT DATE LREAL BYTE WORD DWORD LWORD	
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"。 仅适用于梯形图编程。

结构化文本示例

```
(* 与之等效的 ST：*)
   ao23 := ai10;
```

Multiplication

乘以两个或多个整型或实型值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前相乘计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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。
i2	输入	SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT LINT 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。 仅适用于梯形图编程。

相乘结构化文本示例

(* ST 等效值 *)

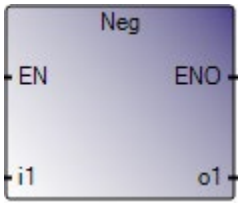
```
ao10 := ai101 * ai102;
ao5 := (ai51 * ai52) * ai53;
```

Neg（取反）

将值转换为反值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前转换为反变量的计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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”。 仅适用于梯形图编程。

Neg 结构化文本示例

(* 与之等效的 ST：*)

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

POW（增加幂）

如果第一个参数为“base”，第二个参数为“exponent”，则计算 (base exponent) 的实型结果。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



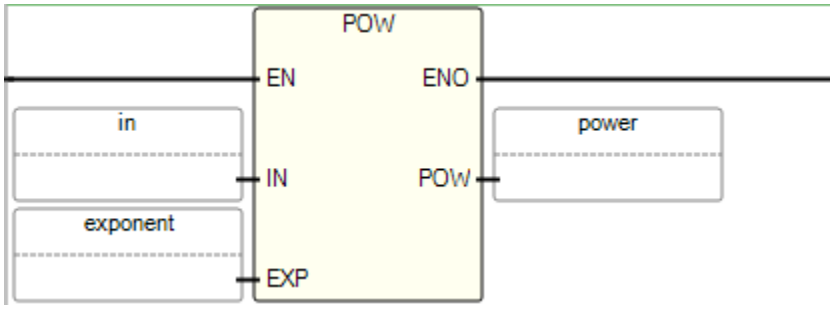
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前指数计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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"。 适用于梯形图编程。

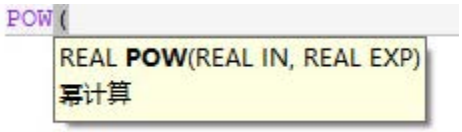
POW 功能块图示例



POW 梯形图示例



POW 结构化文本示例




```
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



RAND（随机值）

从定义的范围计算随机整数值。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



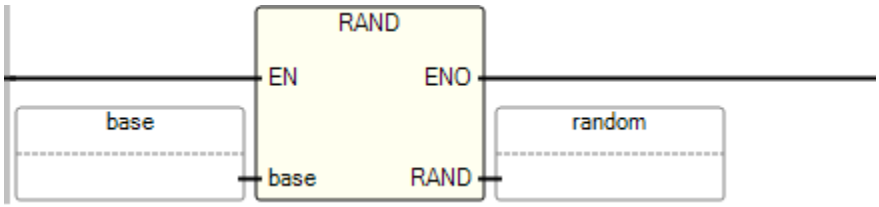
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行随机整数值计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
base	输入	DINT	定义支持的一组数字。
RAND	输出	DINT	集合 [0..base-1] 内的随机数。
ENO	输出	BOOL	启用"输出"。 适用于梯形图编程。

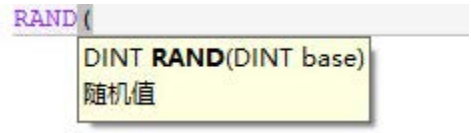
RAND 功能块图示例



RAND 梯形图示例



RAND 结构化文本示例




```
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



SIN（正弦）

计算实型值的正弦。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



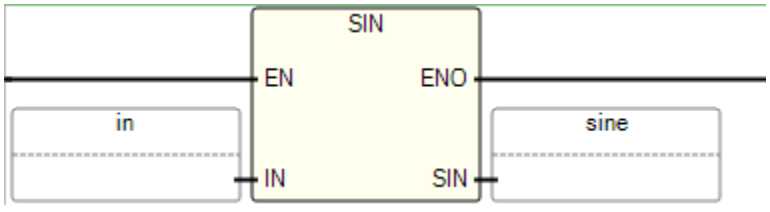
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前正弦计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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”。 适用于梯形图编程。

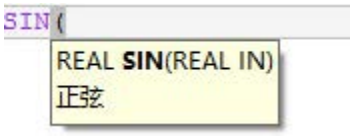
SIN 功能块图示例



SIN 梯形图示例



SIN 结构化文本示例



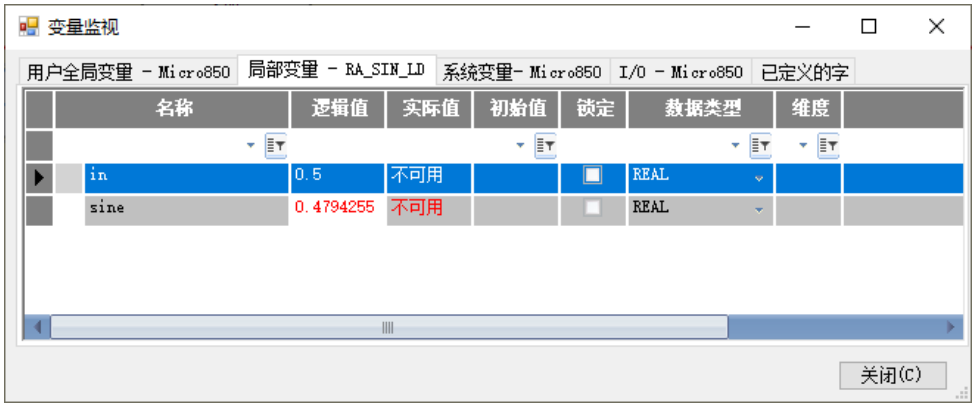

```
1  in := 0.5;
2  sine := SIN(in);
```

(* 与之等效的 ST: *)

sine := SIN (angle);

result := ASIN (sine); (* 结果为角度 *)

结果

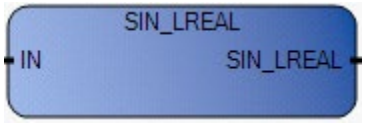


SIN_LREAL（正弦长实型）

计算长实型值的正弦。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

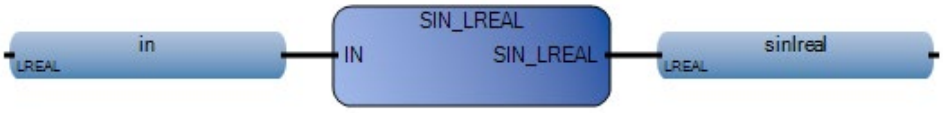
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



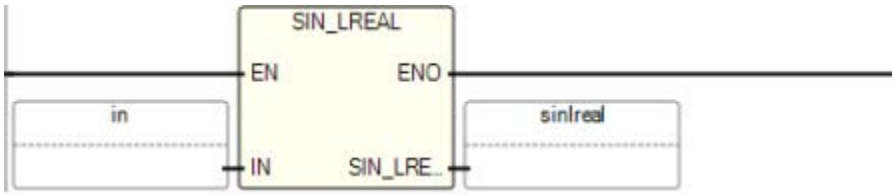
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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”。 适用于梯形图编程。

SIN_LREAL 功能块图示例



SIN_LREAL 梯形图示例



SIN_LREAL 结构化文本示例

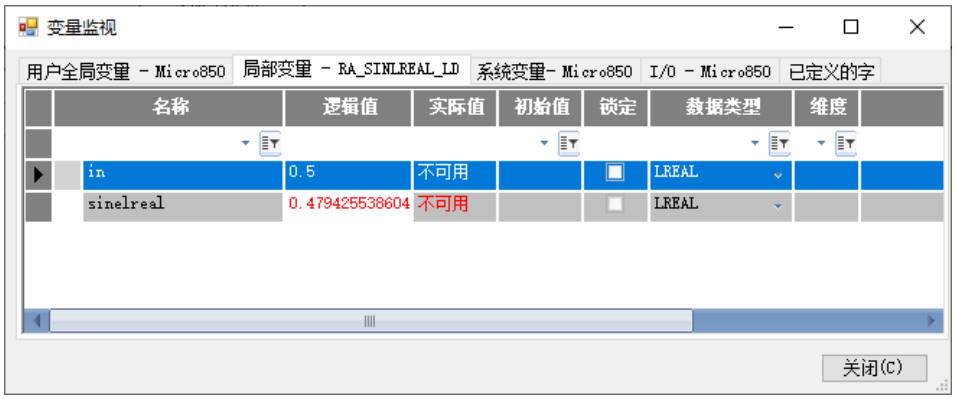



```
1  in := 0.5;
2  sinlreal := SIN_LREAL(in);
```

(* 与之等效的 ST: *)

TESTOUTPUT1 := SIN_LREAL(TESTINPUT1) ;

结果



SQRT（平方根）

计算实型值的平方根。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



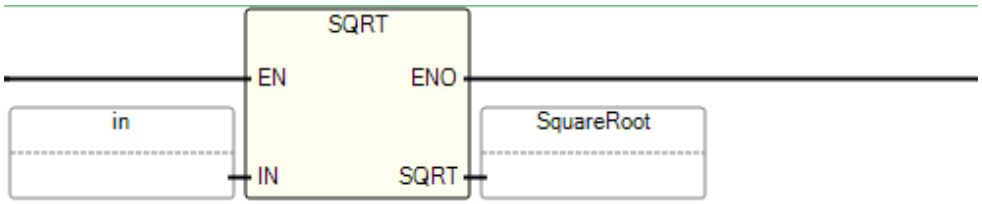
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前平方根计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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”。 适用于梯形图编程。

SQRT 功能块图示例



SQRT 梯形图示例



SQRT 结构化文本示例

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




```
(* 与之等效的 ST： *)
xpos := ABS (xval);
xroot := SQRT (xpos);
```

结果

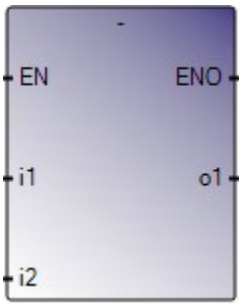


Subtraction

用一个整型、实型或时间值减去另一个整型、实型或时间值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前相加计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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。

参数	参数类型	数据类型	描述
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"。 仅适用于梯形图编程。

相减结构化文本示例

(* 与之等效的 ST：*)

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

TAN（正切）

计算实型值的正切。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



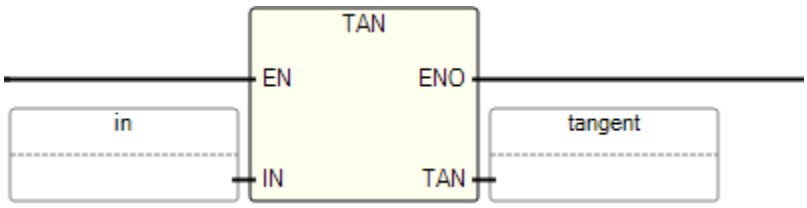
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前正切计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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"。 适用于梯形图编程。

TAN 功能块图示例



TAN 梯形图示例



TAN 结构化文本示例




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

(* 与之等效的 ST: *)

tangent := TAN (angle);

result := ATAN (tangent); (* 结果为角度 *)

结果

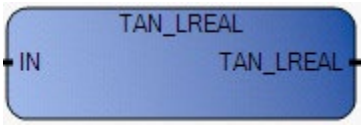


TAN_LREAL（正切长实型）

计算长实型值的正切。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

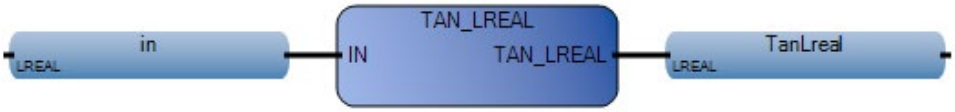
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



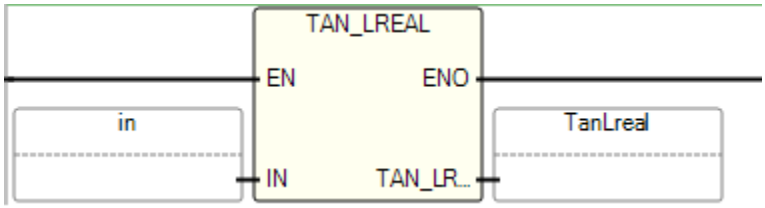
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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。
ENO	输出	BOOL	启用“输出”。 适用于梯形图编程。

TAN_LREAL 功能块图示例



TAN_LREAL 梯形图示例



TAN_LREAL 结构化文本示例



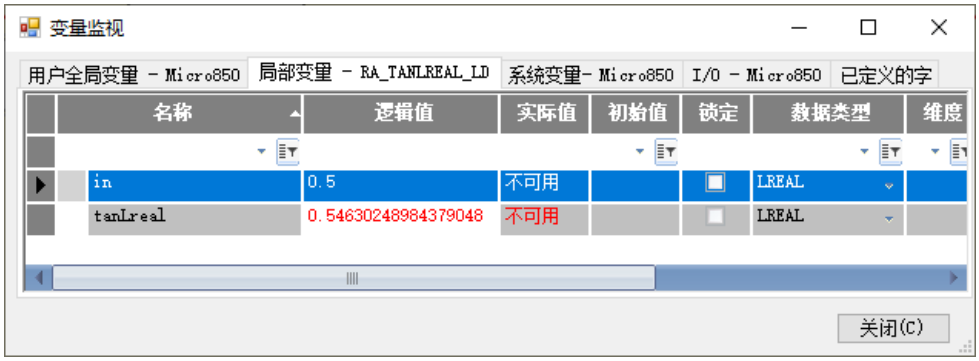

```
1  in := 0.5;
2  TanLreal := TAN_LREAL(in);
```

(* 与之等效的 ST: *)

tangent := TAN_LREAL (angle);

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

结果



TRUNC（截断）

截断实型值，只保留整数。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

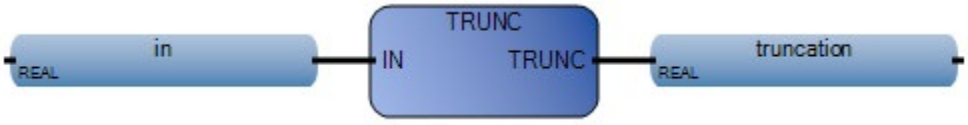
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



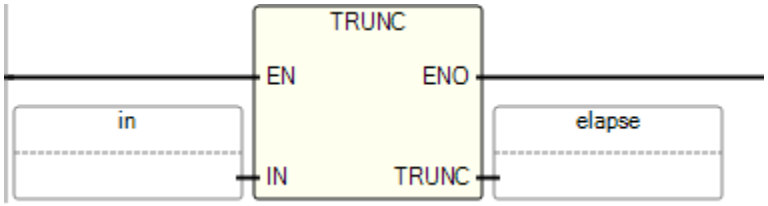
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行实型值截断计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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"。 适用于梯形图编程。

TRUNC 功能块图示例



TRUNC 梯形图示例



TRUNC 结构化文本示例



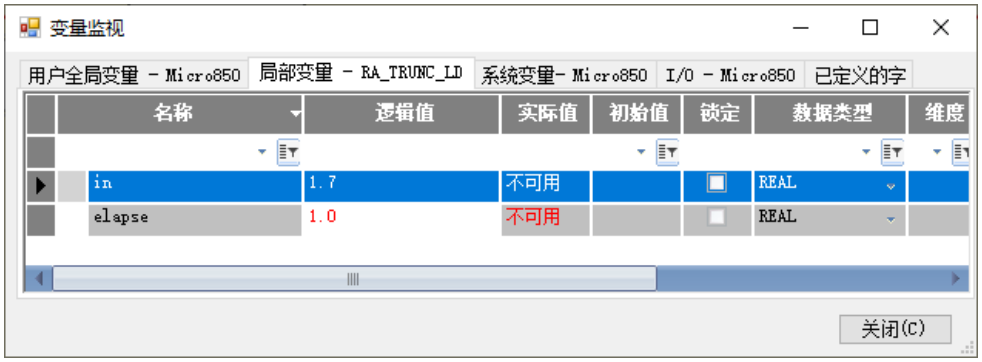

```
1  in := 1.7;
2  truncation := TRUNC(in);
```

(* 与之等效的 ST： *)

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

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

结果



ASCII 串行端口指令

ASCII 串行端口指令用于使用或修改通信信道来接收或传输数据。

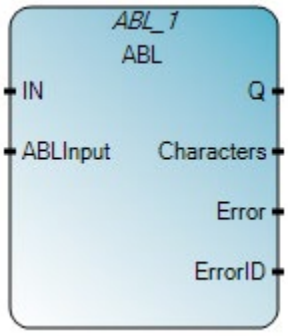
功能块	描述
ABL 参考页数 97	计算缓冲区中的字符数（甚至可包括行尾字符）。
ACB 参考页数 103	计算缓冲区中的字符总数。
ACL 参考页数 99	清除接收和传输缓冲区。
AHL 参考页数 101	设置或复位调制解调器握手信号行。
ARD 参考页数 105	从输入缓冲区读取字符，并将这些字符放置到某个字符串中。
ARL 参考页数 107	从输入缓冲区读取一行字符，并将这些字符放置到某个字符串中。
AWA 参考页数 109	将包含两个附加（用户配置）字符的字符串写入外部设备。
AWT 参考页数 111	将字符从源字符串写入外部设备。

ABL（缓冲区行的 ASCII 测试）

统计输入缓冲区中的 ASCII 字符数，包括行尾终止符。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 如果检测到上升沿，则启动功能块，前提是上一个操作已完成。 FALSE - 指令块处于空闲状态。
ABLInput	输入	ABLACB	要操作的通道。 使用 ABLACB 数据类型 参考页数 114 定义 ABLInput 的通道、触发类型和取消参数。
Q	输出	BOOL	指示字符计数正在进行或完成的时间。 输出与程序扫描异步更新。输出 Q 无法用于重新触发说明，因 IN 已进行沿触发。 TRUE - 功能块完成。

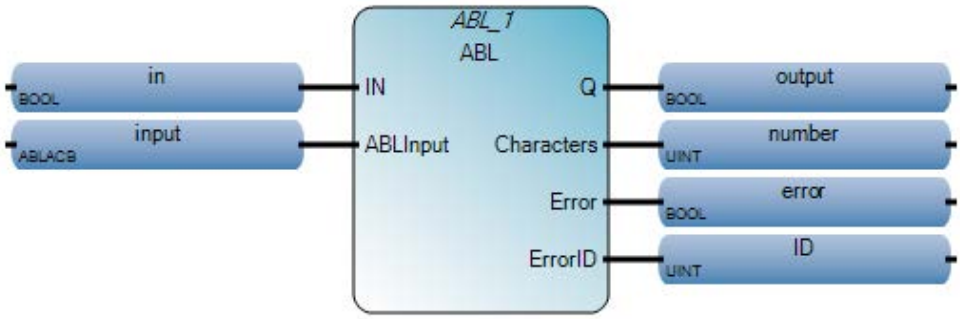
参数	参数类型	数据类型	描述
			FALSE - 功能块未完成。
Characters	输出	UINT	缓冲区中的字符数。缓冲区限制为 82 个字符。
Error	输出	BOOL	表示存在错误条件。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。在 ABL 错误代码中定义的该指令的错误。

ABL 错误代码

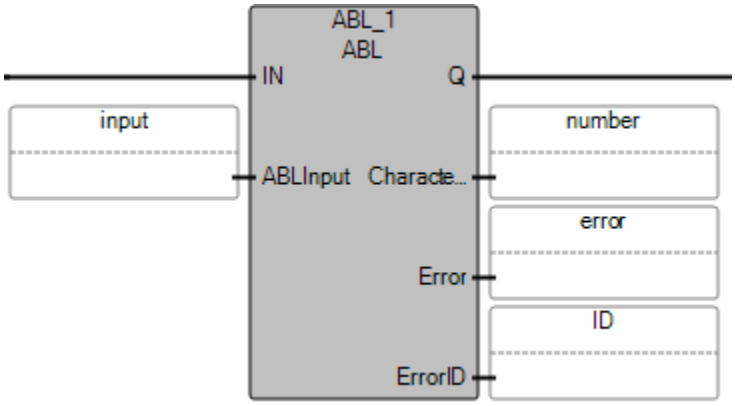
使用此表确定 ABL 错误代码和描述。

错误代码	错误描述
03	传输无法完成，因为清除发送信号已丢失。
06	检测到非法参数。
07	无法完成 ASCII 发送或接收，因为已使用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 和 ARL 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 和 AWT 功能块。
14	ACL 功能块已被取消。
16	串行端口不支持 RTS 或 CTS 控制行。

ABL 功能块图示例



ABL 梯形图示例



ABL 结构化文本示例



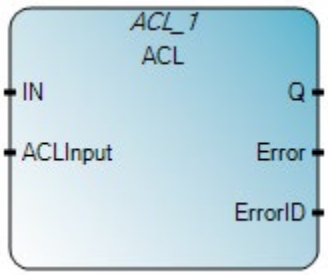

```
1  ABL_1(in, input);
2  output := ABL_1.Q;
3  number := ABL_1.Characters;
4  error := ABL_1.Error;
5  ID := ABL_1.ErrorID;
```

ACL（ASCII 清除缓冲区）

清除接收和传输缓冲区，并从 ASCII 队列删除指令。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 如果检测到上升沿，则启动功能块，前提是上一个操作已完成。 FALSE - 指令块处于空闲状态。

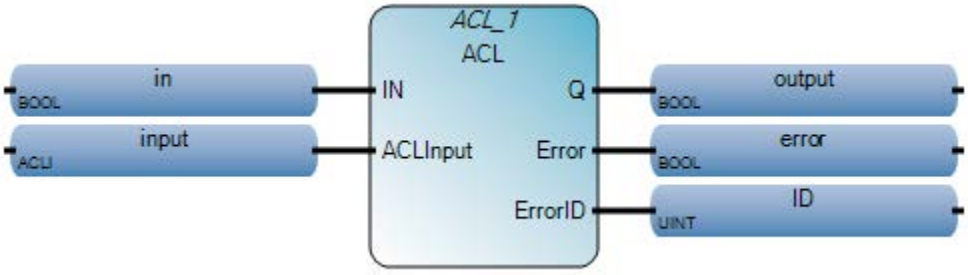
参数	参数类型	数据类型	描述
ACLInput	输入	ACLI	要操作的通道，以及传输和接收缓冲区的状态。 对于 RXBuffer，清除接收缓冲区，并从 ASCII 队列删除接收 ASCII 功能块（ARL 和 ARD）。 对于 TXBuffer，清除传输缓冲区，并从 ASCII 队列删除传输 ASCII 功能块（AWA 和 AWT）。 使用 ACLI 数据类型 参考页数 114 定义 ACLInput 的通道、RXBuffer 和 TXBuffer 参数。
Q	输出	BOOL	表示 ASCII 队列清除进程正在进行或完成的时间。 TRUE - 功能块完成。 FALSE - 功能块未完成。
Error	输出	BOOL	表示存在错误条件。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。在 ABL 错误代码中定义的该指令的错误。

ABL 错误代码

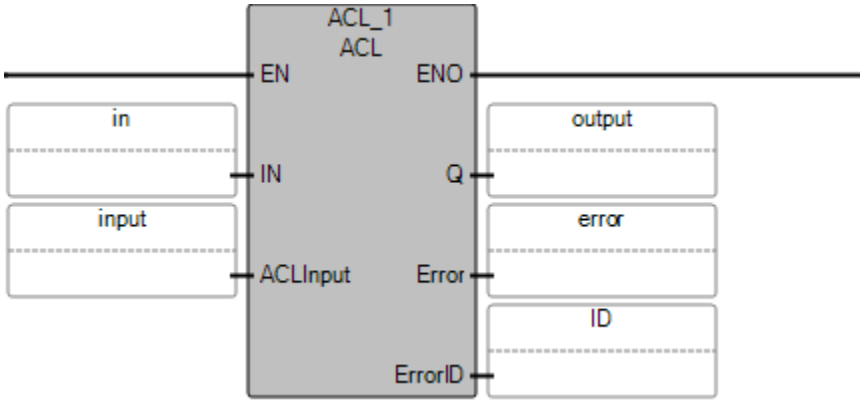
使用此表确定 ABL 错误代码和描述。

错误代码	错误描述
03	传输无法完成，因为清除发送信号已丢失。
06	检测到非法参数。
07	无法完成 ASCII 发送或接收，因为已使用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 和 ARL 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 和 AWT 功能块。
14	ACL 功能块已被取消。
16	串行端口不支持 RTS 或 CTS 控制行。

ACL 功能块图示例



ACL 梯形图示例



ACL 结构化文本示例



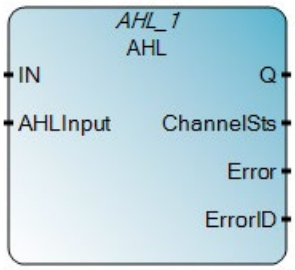

```
1  ACL_1(in, input);
2  output := ACL_1.Q;
3  error := ACL_1.Error;
4  ID := ACL_1.ErrorID;
```

AHL（ASCII 握手信号行）

设置或复位调制解调器的 RS-232 请求发送 (RTS) 握手信号控制行。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 如果检测到上升沿，则启动指令块，前提是上一个操作已完成。 FALSE - 指令块处于空闲状态。
AHLInput	输入	AHLI	要操作的通道，以及调制解调器的 RTS 控制行的设置或复位。 使用 AHLI 数据类型 参考页数 114定义 AHLInput 的通道、SetRts、ClrRts 和取消参数。

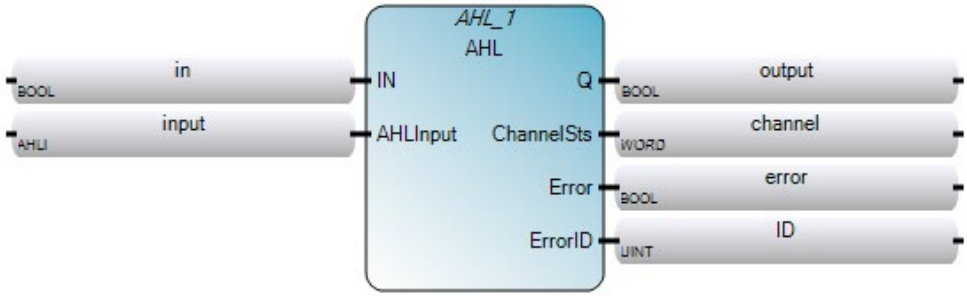
参数	参数类型	数据类型	描述
Q	输出	BOOL	指示设置或复位的完成时间。 从程序扫描中同步更新此说明的输出。输出 Q 无法用于重新触发说明，因 IN 已进行沿触发。 TRUE - 功能块完成。 FALSE - 功能块未完成。
ChannelSts	输出	WORD	显示指定通道的握手信号行的当前状态（0000 到 001F）。 有关当前状态的详细信息，请参阅 AHLI 数据类型 参考页数 114。
Error	输出	BOOL	表示存在错误条件。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。在 ABL 错误代码中定义的该指令的错误。

ABL 错误代码

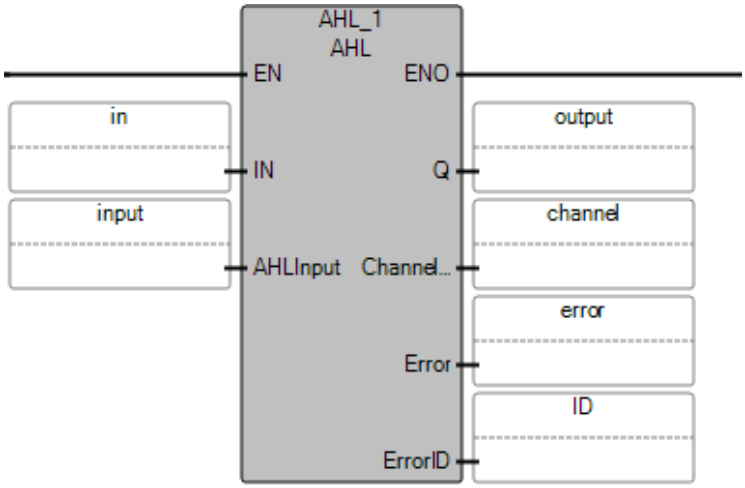
使用此表确定 ABL 错误代码和描述。

错误代码	错误描述
03	传输无法完成，因为清除发送信号已丢失。
06	检测到非法参数。
07	无法完成 ASCII 发送或接收，因为已使用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 和 ARL 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 和 AWT 功能块。
14	ACL 功能块已被取消。
16	串行端口不支持 RTS 或 CTS 控制行。

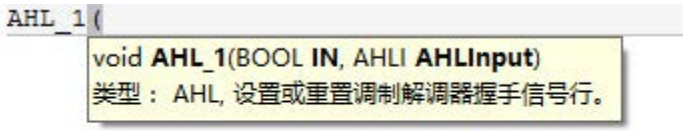
AHL 功能块图示例



AHL 梯形图示例



AHL 结构化文本示例



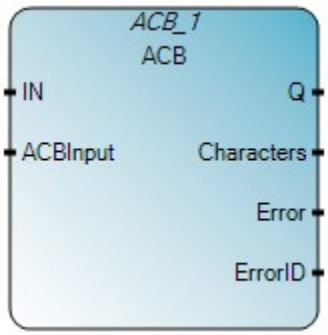

```
1  AHL_1(in, input);
2  output := AHL_1.Q;
3  channel := AHL_1.ChannelSts;
4  error := AHL_1.Error;
5  ID := AHL_1.ErrorID;
```

ACB（缓冲区中的 ASCII 字符）

统计缓冲区中的 ASCII 字符总数，包括行尾。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 如果检测到上升沿，则启动指令块，前提是上一个操作已完成。 FALSE - 指令块处于空闲状态。

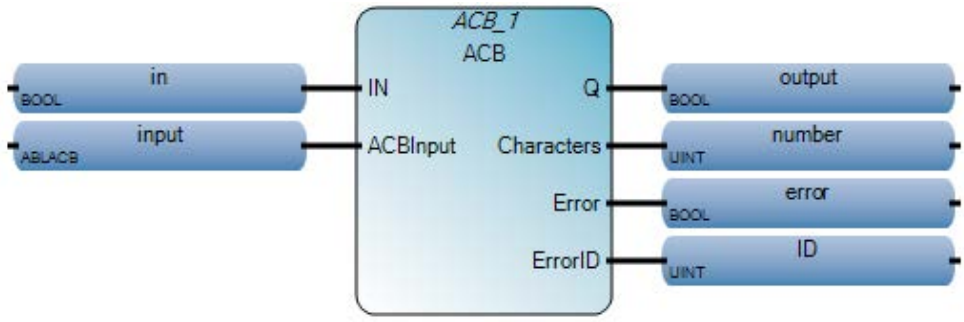
参数	参数类型	数据类型	描述
ACBInput	输入	ABLACB 参考页数 114	要操作的通道。 使用 ABLACB 数据类型定义 ACBInput 的通道、触发类型和取消参数。
Q	输出	BOOL	表示字符计数是正在进行还是已完成。 此功能块的输出与程序扫描同步更新。输出 Q 无法用于重新触发说明，因 IN 已进行沿触发。 TRUE - 计数已完成。 FALSE - 计数正在进行。
Characters	输出	UINT	缓冲区中的字符数。
Error	输出	BOOL	表示存在错误条件。 FALSE - 没有错误。 TRUE - 检测到错误。
ErrorID	输出	UINT	标识错误的唯一数字。在 ABL 错误代码中定义的该指令的错误。

ACB 错误代码

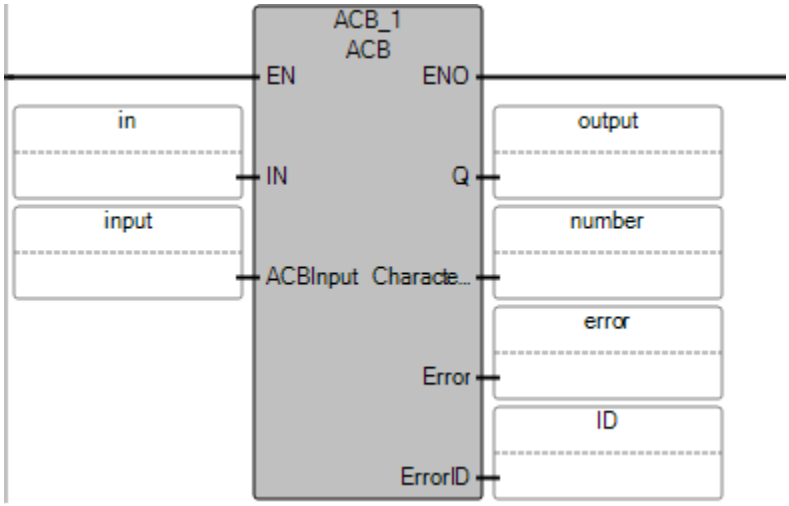
使用此表确定 ABL 错误代码和描述。

错误代码	错误描述
03	传输无法完成，因为清除发送信号已丢失。
06	检测到非法参数。
07	无法完成 ASCII 发送或接收，因为已使用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 和 ARL 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 和 AWT 功能块。
14	ACL 功能块已被取消。
16	串行端口不支持 RTS 或 CTS 控制行。

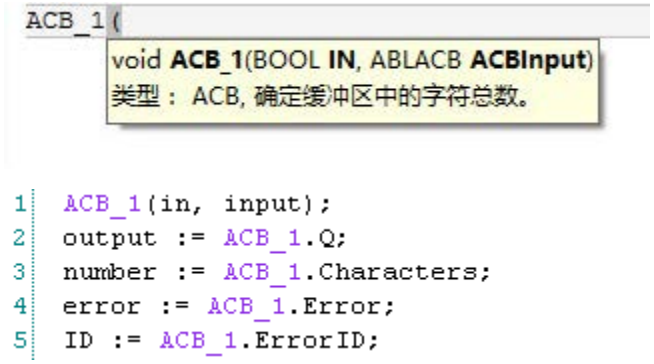
ACB 功能块图示例



ACB 梯形图示例



ACB 结构化文本示例




```
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;
```

ARD（ASCII 读取）

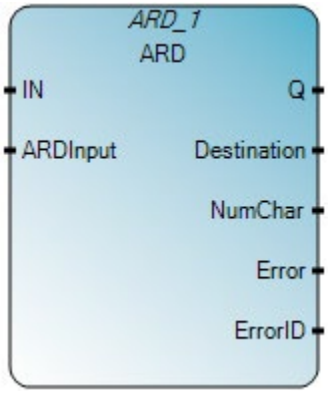
从输入缓冲区读取 ASCII 字符，并将这些字符存储到某个字符串中。

操作详细信息：

- ARD 指令运行直至收到 ASCII 缓冲区中的所有字符。如果执行其他 ASCII 指令 参考页数 97，该命令将排队等待直至 ARD 完成。
- 要取消 ARD 指令，请执行 ACL 指令。
- 为了防止在等待所需的字符数时 ARD 指令延迟 ASCII 队列，使用 ACB 指令的结果来触发 ARD 指令。
- 指令状态可从指令实例的控制位（如 ARD_1.controlbit）提取。若指令阻塞，这显示正在等待更多数字符的 ASCII 指令队列。
  - 7th bit = 指令可用。
  - 6th bit = 指令列队中。
  - 5th bit = 指令完成。
  - 3rd bit = 指令错误。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

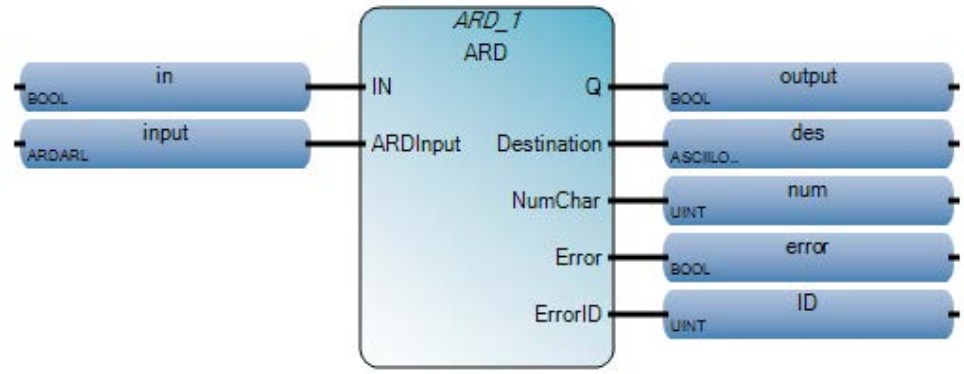
参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 如果检测到上升沿，则启动指令块，前提是上一个操作已完成。 FALSE - 指令块处于空闲状态。
ARDInput	输入	ARDARL 参考页数 115	从缓冲区读取字符。最多 82 个。 使用 ARDARL 数据类型定义 ARDInput 的通道、长度和取消参数。
Q	输出	BOOL	表示缓冲区读取正在进行或完成的时间。 从程序扫描中同步更新此说明的输出。输出 Q 无法用于重新触发说明，因 IN 已进行沿触发。 TRUE - 功能块完成。 FALSE - 功能块未完成。
Destination	输出	ASCIILOCADDR	用于存储字符的字符串元素。
NumChar	输出	UINT	字符数。
Error	输出	BOOL	表示存在错误条件。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。在 ABL 错误代码中定义的该指令的错误。

ABL 错误代码

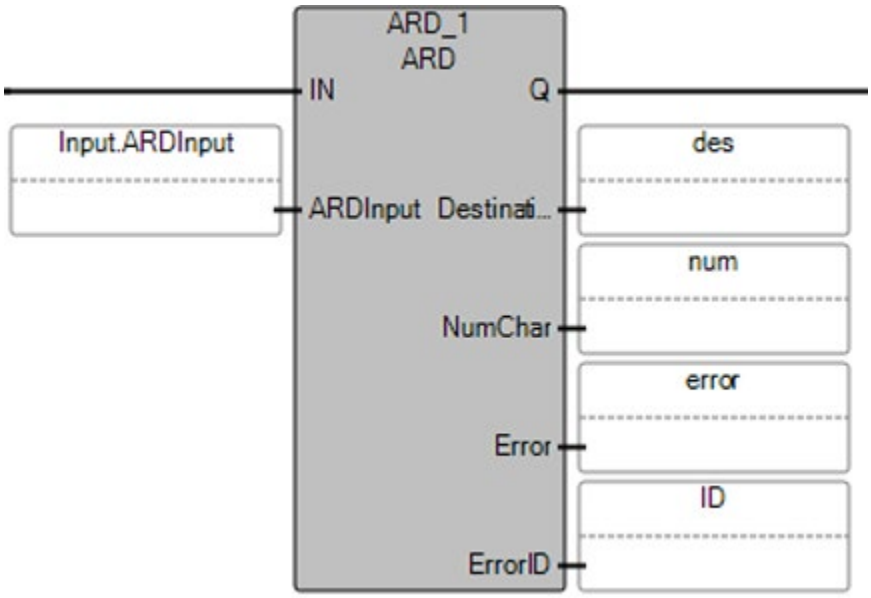
使用此表确定 ABL 错误代码和描述。

错误代码	错误描述
03	传输无法完成，因为清除发送信号已丢失。
06	检测到非法参数。
07	无法完成 ASCII 发送或接收，因为已使用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 和 ARL 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 和 AWT 功能块。
14	ACL 功能块已被取消。
16	串行端口不支持 RTS 或 CTS 控制行。

ARD 功能块图示例



ARD 梯形图示例



ARD 结构化文本示例



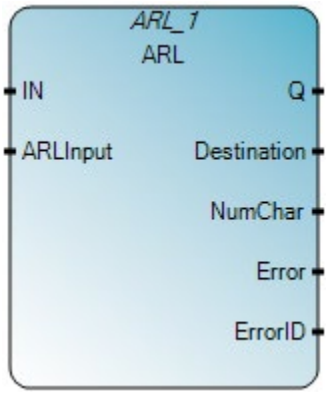

```
1  ARD_1(in, input);
2  output := ARD_1.Q;
3  des := ARD_1.Destination;
4  num := ARD_1.NumChar;
5  error := ARD_1.Error;
6  ID := ARD_1.ErrorID;
```

ARL（ASCII 读取行）

从缓冲区读取一行 ASCII 参考页数 97 字符（包括终止符），并将这些字符存储在字符串中。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 如果检测到上升沿，则启动指令块，前提是上一个操作已完成。 FALSE - 指令块处于空闲状态。
ARLInput	输入	ARDARL 参考页数 115	从缓冲区读取一行 ASCII 字符。最多 82 个。 使用 ARDARL 数据类型定义 ARDLInput 的通道、长度和取消参数。
Q	输出	BOOL	指示从输入缓冲区读取行的操作正在进行或完成的时间。 从程序扫描中同步更新此说明的输出。输出 Q 无法用于重新触发说明，因 IN 已进行沿触发。 TRUE - 功能块完成。 FALSE - 功能块未完成。
Destination	输出	ASCIILOCADDR	用于存储字符的字符串元素。
NumChar	输出	UINT	行中的字符数，包括终止字符。
Error	输出	BOOL	表示存在错误条件。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。在 ABL 错误代码中定义的该指令的错误。

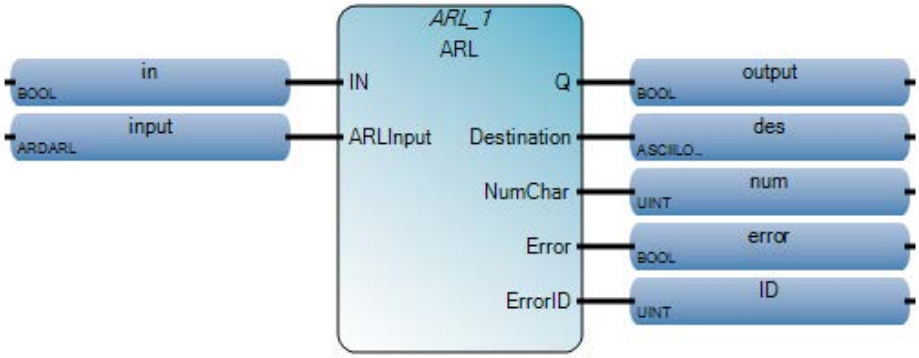
ABL 错误代码

使用此表确定 ABL 错误代码和描述。

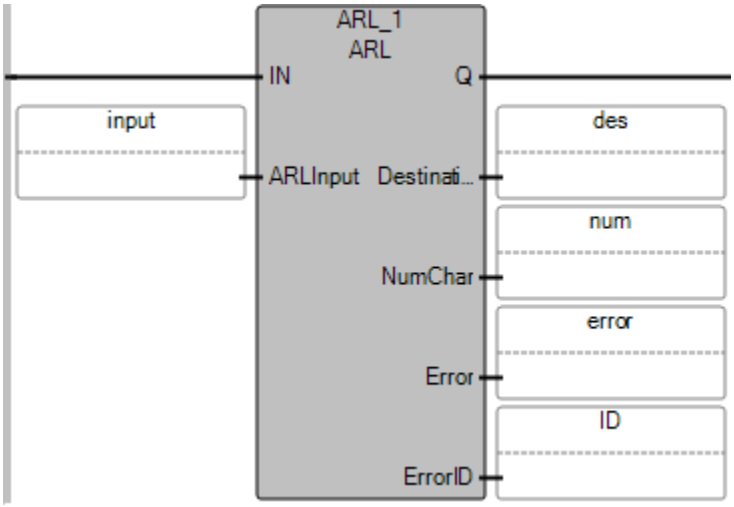
错误代码	错误描述
03	传输无法完成，因为清除发送信号已丢失。
06	检测到非法参数。
07	无法完成 ASCII 发送或接收，因为已使用通道配置对话框关闭通道配置。
08	无法完成 ASCII 写入，因为 ASCII 传输已在进行。
09	当前的通道配置不支持请求的 ASCII 通信。
10	已设置 Cancel，从而停止指令的执行。不需要任何操作。
11	字符串的请求长度无效、为负数、大于 82 或为 0。适用于 ARD 和 ARL 功能块。
13	控制块中请求的 (.LEN) 长度为负数或大于存储在源字符串中的字符串大小。适用于 AWA 和 AWT 功能块。
14	ACL 功能块已被取消。

错误代码	错误描述
16	串行端口不支持 RTS 或 CTS 控制行。

ARL 功能块图示例



ARL 梯形图示例



ARL 结构化文本示例



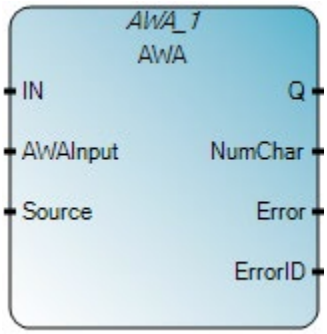

```
1  ARL_1(in, input);
2  output := ARL_1.Q;
3  des := ARL_1.Destination;
4  num := ARL_1.NumChar;
5  error := ARL_1.Error;
6  ID := ARL_1.ErrorID;
```

AWA（ASCII 写入附加）

将包含两个附加（用户配置）字符的字符串写入外部设备。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 指令块处于空闲状态。
AWAInput	输入	AWAAWT	通道和要写入缓冲区的字符数（长度）。最大值为 82。 使用 AWAAWT 数据类型 参考页数 115 定义 AWAInput 的通道、长度和取消参数。
Source	输入	ASCIILOCADDR	属于 ARD 或 ARL 指令的输出（作为字符数组）的源字符串。
Q	输出	BOOL	表示写入正在进行或完成的时间。 从程序扫描中同步更新此说明的输出。输出 Q 无法用于重新触发说明，因 IN 已进行沿触发。 TRUE - 功能块完成。 FALSE - 功能块未完成。
NumChar	输出	UINT	字符数。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。 当传输完成并且 Q 为 TRUE 时将会更新。
Error	输出	BOOL	表示存在错误条件。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。在 ABL 错误代码中定义的该指令的错误。

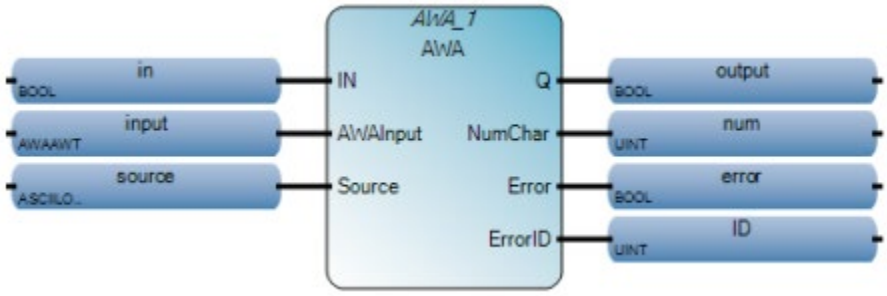
ABL 错误代码

使用此表确定 ABL 错误代码和描述。

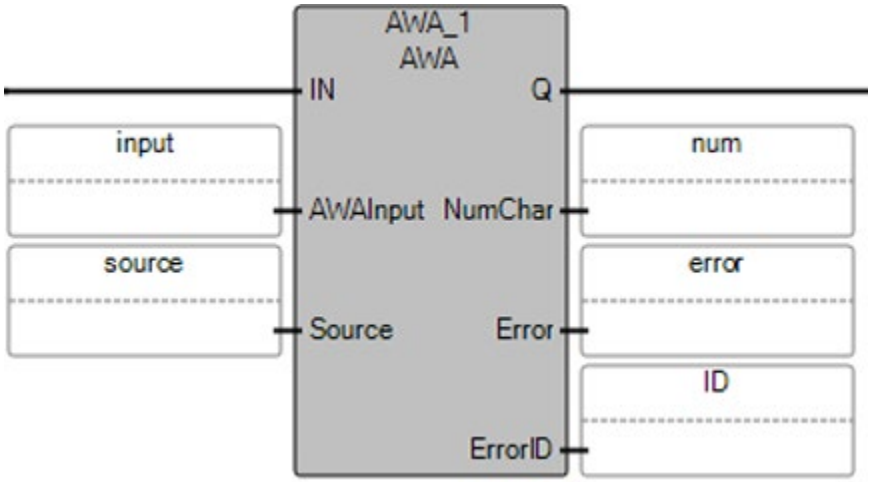
错误代码	错误描述
03	传输无法完成，因为清除发送信号已丢失。
06	检测到非法参数。
07	无法完成 ASCII 发送或接收，因为已使用通道配置对话框关闭通道配置。
08	无法完成 ASCII 写入，因为 ASCII 传输已在进行。
09	当前的通道配置不支持请求的 ASCII 通信。
10	已设置 Cancel，从而停止指令的执行。不需要任何操作。
11	字符串的请求长度无效、为负数、大于 82 或为 0。适用于 ARD 和 ARL 功能块。
13	控制块中请求的 (.LEN) 长度为负数或大于存储在源字符串中的字符串大小。适用于 AWA 和 AWT 功能块。

错误代码	错误描述
14	ACL 功能块已被取消。
16	串行端口不支持 RTS 或 CTS 控制行。

AWA 功能块图示例



AWA 梯形图示例



AWA 结构化文本示例



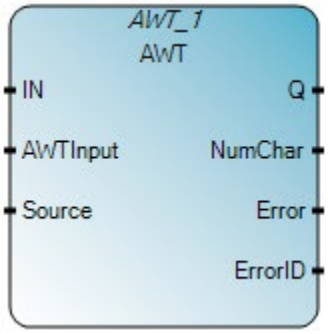

```
1  AWA_1(in, input, source);
2  output := AWA_1.Q;
3  num := AWA_1.NumChar;
4  error := AWA_1.Error;
5  ID := AWA_1.ErrorID;
```

AWT（ASCII 写入）

将 ASCII 字符从源字符串写入外部设备。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

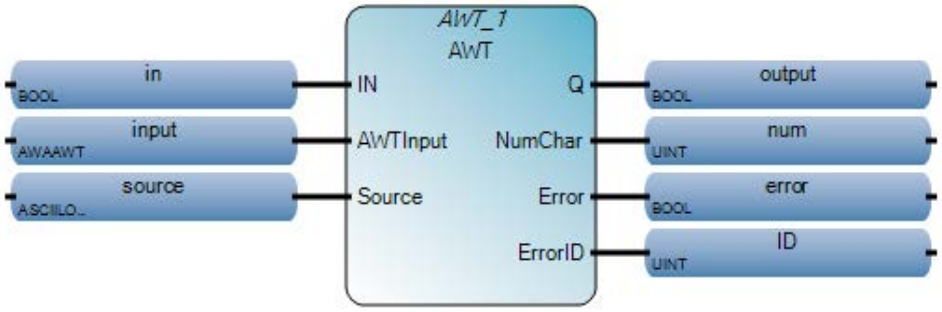
参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 如果检测到上升沿，则启动指令块，前提是上一个操作已完成。 FALSE - 指令块处于空闲状态。
AWTInput	输入	AWAAWT	通道和要写入缓冲区的字符数（长度）。最大值为 82。 使用 AWAAWT 数据类型 参考页数 115 定义 AWTInput 的通道、长度和取消参数。
Source	输入	ASCIILOCADDR	属于 ARD 或 ARL 指令的输出（作为字符数组）的源字符串。
Q	输出	BOOL	表示写入正在进行或完成的时间。 从程序扫描中同步更新此说明的输出。输出 Q 无法用于重新触发说明，因 IN 已进行沿触发。 TRUE - 功能块完成。 FALSE - 功能块未完成。
NumChar	输出	UINT	字符数。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。 当传输完成并且 Q 为 TRUE 时将会更新。
Error	输出	BOOL	表示存在错误条件。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。在 ABL 错误代码中定义的该指令的错误。

ABL 错误代码

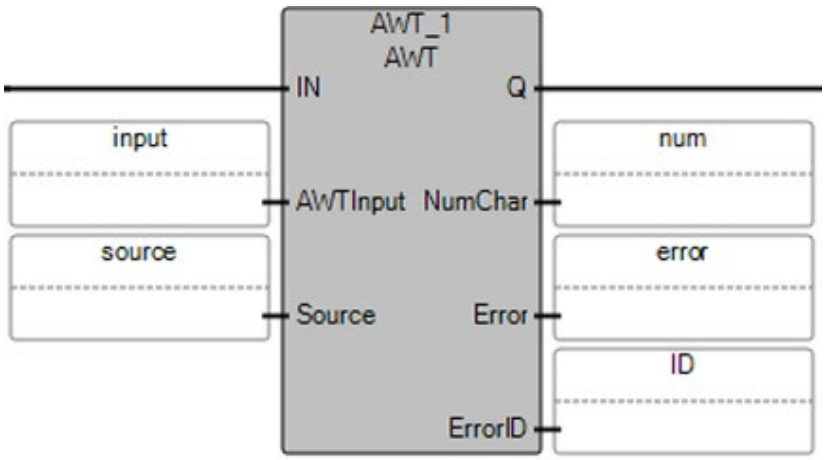
使用此表确定 ABL 错误代码和描述。

错误代码	错误描述
03	传输无法完成，因为清除发送信号已丢失。
06	检测到非法参数。
07	无法完成 ASCII 发送或接收，因为已使用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 和 ARL 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 和 AWT 功能块。
14	ACL 功能块已被取消。
16	串行端口不支持 RTS 或 CTS 控制行。

AWT 功能块图示例



AWT 梯形图示例



AWT 结构化文本示例

```
AWT_1(
```

void **AWT_1**(BOOL **IN**, AWAAWT **AWTInput**, ASCIILOCADDR **Source**)
类型： AWT, 将字符从源字符串写入到外部设备中。

```
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;
```

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。

- ABLACB 数据类型
- ACL 数据类型
- AHL ChannelSts 数据类型
- AHLI 数据类型
- ARDARL 数据类型
- AWAAWT 数据类型

ABLACB 数据类型

使用此表帮助确定 ABLACB 数据类型的参数值。

参数	数据类型	描述
Channel	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
Cancel	BOOL	当此输入设置为 TRUE 时，此功能块不会执行。

ACL 数据类型

使用此表帮助确定 ABL 数据类型的参数值。

参数	数据类型	描述
Channel	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）。

AHL ChannelSts 数据类型

使用此表帮助确定 AHL ChannelSts 数据类型的参数值。

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

AHLI 数据类型

使用此表帮助确定 AHL 数据类型的参数值。

参数	数据类型	描述

参数	数据类型	描述
Channel	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。
Cancel	BOOL	当此输入设置为 TRUE 时，此功能块不会执行。

ARDARL 数据类型

使用此表帮助确定 ARDARL 数据类型的参数值。

参数	数据类型	描述
Channel	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）。
Cancel	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。

AWAAWT 数据类型

使用此表帮助确定 AWAAWT 数据类型的参数值。

参数	数据类型	描述
Channel	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	定义要写入缓冲区的字符数（最大值为 82）。 如果将 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。
Cancel	BOOL	如果设置为 TRUE，则此功能块不执行。如果已执行，则操作会停止。

二进制指令

使用二进制指令执行数学运算。

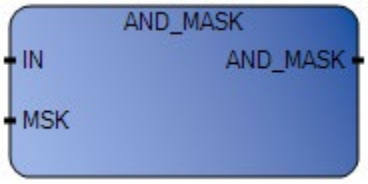
运算符	描述
AND_MASK 参考页数 117	在两个整型值之间执行位到位 AND 运算。
NOT_MASK 参考页数 125	整型位到位取反掩码，将反转参数值。
BSL 参考页数 119	将数组元素中的位向左移动。
BSR 参考页数 122	将数组元素中的位向右移动。
OR_MASK 参考页数 126	整型 OR 位到位掩码，将启用位。
ROL 参考页数 128	对于 32 位整数，将整数位旋转到左侧。
ROR 参考页数 129	对于 32 位整数，将整数位旋转到左侧。
SHL 参考页数 131	对于 32 位整数，将整数向左移动，并在最低有效位中置 0。
SHR 参考页数 133	对于 32 位整数，将整数向右移动，并在最高有效位中置 0。
XOR_MASK 参考页数 134	整数异或位到位掩码，返回反转的位值。

AND_MASK（AND 掩码）

在两个整型值之间执行位到位 AND 运算。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

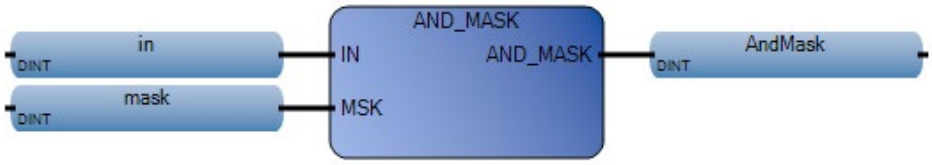
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



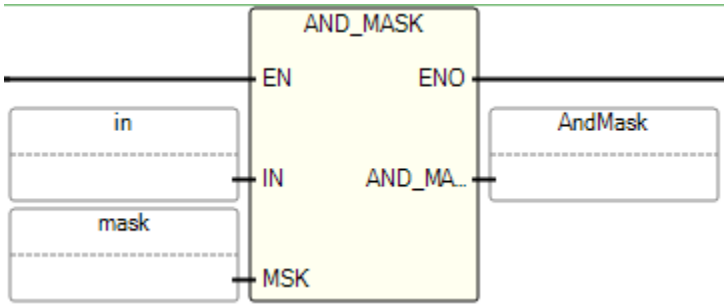
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行整型与位到位掩码计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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_MASK	输出	DINT	IN 与 MSK 之间的位到位逻辑与。
ENO	输出	BOOL	启用“输出”。 适用于梯形图编程。

AND_MASK 功能块图示例



AND_MASK 梯形图示例



AND_MASK 结构化文本示例




```
1  in := 5;
2  mask := 6;
3  AndMask := AND_MASK(in, 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#fof); (* 等于 16#aoc *)

结果



BSL（向左移动位）

将数组元素中的位向左移动。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

操作详细信息：

BSL 指令是 false-true 梯级转换时的即时进程，并同步更新输出。当 Execution 为 TRUE 时，将最左侧位（Src + SrcOffset 和 Length）复制到 Unload 位，并且数组或非数组中的所有位都向左移一位。考虑 Length 和 16 位边界，BOOL 数据类型除外。然后，外部位移动到第一个元素的 0 位 (Src + SrcOffset)。

对于环绕操作，将 BitAddr 的位置设置为最后一个位或 Unload 位。BSL 指令的可能使用情况，通过容器生产线跟踪容器（其中每个位表示一个容器）。



使用此表有助于确定该指令的参数值。

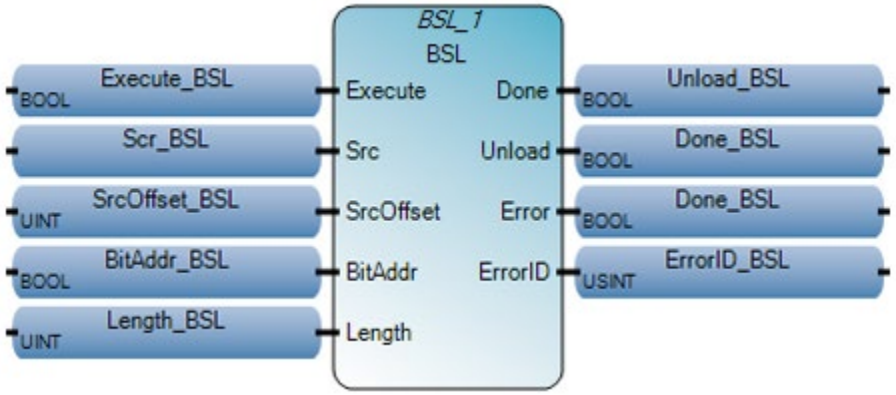
参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令。 TRUE - 检测到上升沿，将位向左移动一个位置。 • 首先确认故障条件。 • 如果 Length = 0，外部位已移动到 Unload 位。未在 Scr 上完成位移动。Error 和 ErrorID 位已复位。Done 位已置位。 • 如果 Length > 0 并且 Length ≤ 2048，Error 和 ErrorID 位已复位。在位移动完成后，Done 位被置位。 • 如果 Length > 0 并且 Length ≤ 2048，最左侧位（通过 Src + SrcOffset 和 Length 寻址）将复制到 Unload 位中，并且数组或非数组中的所有位向左移一位（最多为 Length 位和 16 位边界，BOOL 除外）。外部位移动到第一个元素的 0 位 (Src + SrcOffset)。 FALSE - 未检测到上升沿，不启用 BSL 操作。
Scr	输入	ANY_ELEMENTARY	要移动的 Src（位）的地址。支持的数据类型：BOOL、DWORD、INT、UINT、WORD、DINT 和 UDINT。 • 数组：将 Scr 设置为基于变量的地址，如：Source1、Source1[0] 或 Source1[1]。 • 非数组：将 Scr 设置为变量地址，如 Source1。
SrcOffset	输入	UINT	如果 SrcOffset 为 0，则从第一个元素开始。 • 数组：将 SrcOffset 设置为 0。如果设置为 Source1[0] 或 Source1[1]，将出现错误：“源偏移超出数组大小”。

参数	参数类型	数据类型	描述
			• 非数组：将 SrcOffset 设置为 0，否则将发生错误："源偏移超出数组大小"。
BitAddr	输入	BOOL	移动到 Src 中的位的位置。
Length	输入	UINT	Length 包含 Src 中要移动的位的数量。支持在数组元素之间移动。 • 对于 BOOL 数据类型，为数组中要移动的布尔值的数量。 • 对于 16 位和 32 位数据类型，将以 16 的倍数（如 16、32 和 64）移动位。 如果 Length 不是 16 的偶倍数，则移动的位的数量将发送到下一个 16 位边界。 • Length 基于数据类型的大小。如果 Length 超出范围，将会导致发生错误"源偏移超出数组大小"。Length 值： • BOOL： 1 • 16 位字：1-16 • 32 位字：1-32 • 64 位字：1-64
Done	输出	BOOL	当为 TRUE 时，操作成功完成。 当为 FALSE 时，操作遇到错误条件。
Unload	输出	BOOL	位从 Src 地址移出。
Error	输出	BOOL	当发生故障时，Error 设置为 True。
ErrorID	输出	USINT	当发生故障时，ErrorID 包含错误代码。

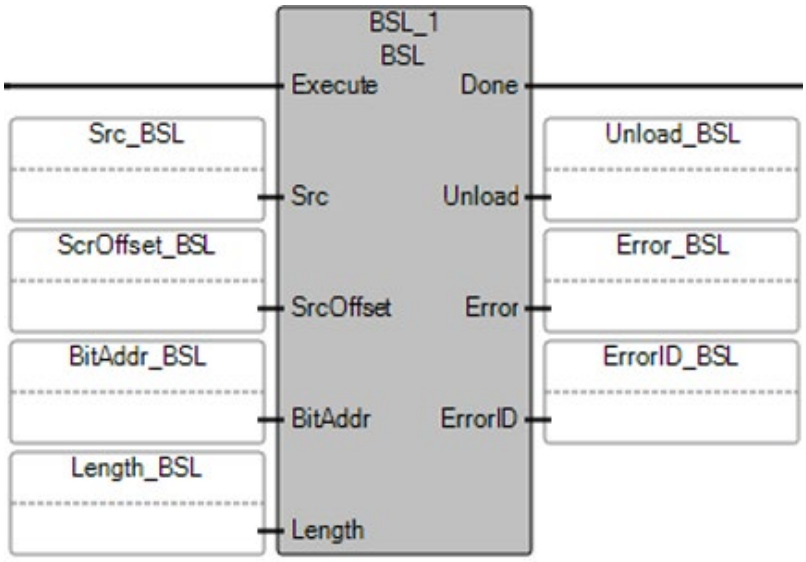
BSL 错误代码

错误代码	错误描述
01	不支持维度。
02	不支持数据类型。
03	位长度超出 2048。
04	源偏移超出数组大小。
05	位长度超出数组大小。
07	参数无效。

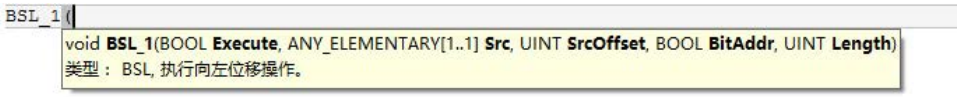
BSL 功能块图示例



BSL 梯形图示例



BSL 结构化文本示例




```
1  BSL_1(Execute,Src,SrcOffset,BitAddr,Length);
2  Done_BSL := BSL_1.Done;
3  Unload_BSL := BSL_1.Unload;
4  Error_BSL := BSL_1.Error;
5  ErrorID_BSL := BSL_1.ErrorID;
```

结果

	名称	别名	逻辑值	实际值	初始值	锁定	数据类型	维度
▶	+ BSL_1		...	...	...	☐	BSL	
	Execute		☐	不可用		☐	BOOL	
	+ Src		...	...	...	☐	BOOL	[1..1]
	SrcOffset		0	不可用		☐	UINT	
	BitAddr		☐	不可用		☐	BOOL	
	Length		0	不可用		☐	UINT	
	Done_BSL		☐	不可用		☐	BOOL	
	Upload_BSL		☐	不可用		☐	BOOL	
	Error_BSL		☐	不可用		☐	BOOL	
	ErrorID_BSL		0	不可用		☐	USINT	

BSR（向右移动位）

将数组元素中的位向右移动。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

操作详细信息：

BSR 指令是 false-true 梯级转换时的即时进程，并同步更新输出。当 Execution 为 TRUE 时，将最右侧位（元素的第 0 位，通过 Src + SrcOffset 寻址）复制到 Unload 位，并且数组或非数组中的所有位向右移一位。考虑 Length 和 16 位边界，BOOL 数据类型除外。然后，外部位移动到第一个元素的 0 位 (Src + SrcOffset)。

对于环绕操作，将 BitAddr 的位置设置为最后一个位或 Unload 位。BSL 指令的可能使用情况，通过容器生产线跟踪容器（其中每个位表示一个容器）。



使用此表有助于确定该指令的参数值。

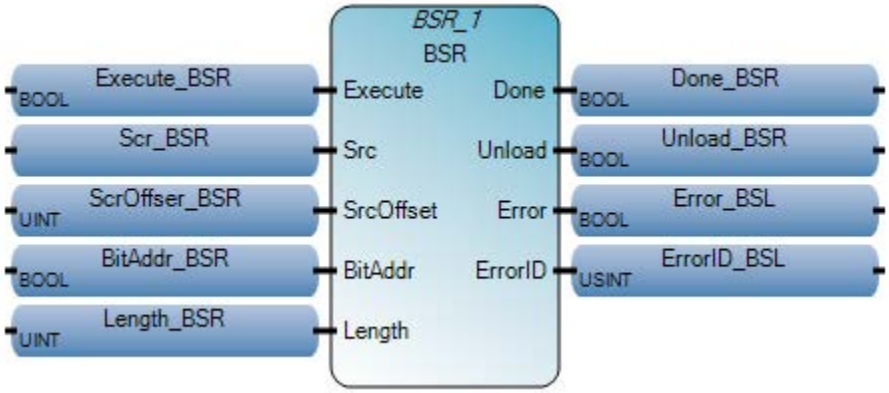
参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令。 TRUE - 检测到上升沿，将位向右移动一个位置。 FALSE - 未检测到上升沿，不启用 BSR 操作。

参数	参数类型	数据类型	描述
Scr	输入	ANY_ELEMENTARY	要移动的 Src（位）的地址。支持的数据类型：BOOL、DWORD、INT、UINT、WORD、DINT 和 UDINT。 • 数组：将 Scr 设置为基于变量的地址，如：Source1、Source1[0] 或 Source1[1]。 • 非数组：将 Scr 设置为变量地址，如 Source1。
SrcOffset	输入	UINT	如果 SrcOffset 为 0，则从第一个元素开始。 • 数组：将 SrcOffset 设置为 0。如果设置为 Source1[0] 或 Source1[1]，将出现错误："源偏移超出数组大小"。 • 非数组：将 SrcOffset 设置为 0，否则将发生错误："源偏移超出数组大小"。
BitAddr	输入	BOOL	移动到 Src 中的位的位置。
Length	输入	UINT	Length 包含 Src 中要移动的位的数量。支持在数组元素之间移动。 • 对于 BOOL 数据类型，为数组中要移动的布尔值的数量。 • 对于 16 位和 32 位数据类型，将以 16 的倍数（如 16、32 和 64）移动位。如果 Length 不是 16 的偶倍数，则移动的位的数量将发送到下一个 16 位边界。 • Length 基于数据类型的大小。如果 Length 超出范围，将会导致发生错误"源偏移超出数组大小"。Length 值： • BOOL： 1 • 16 位字：1-16 • 32 位字：1-32 • 64 位字：1-64
Done	输出	BOOL	当为 TRUE 时，操作成功完成。 当为 FALSE 时，操作遇到错误条件。
Unload	输出	BOOL	位从 Src 地址移出。
Error	输出	BOOL	当发生故障时，Error 设置为 True。
ErrorID	输出	USINT	当发生故障时，ErrorID 包含错误代码。

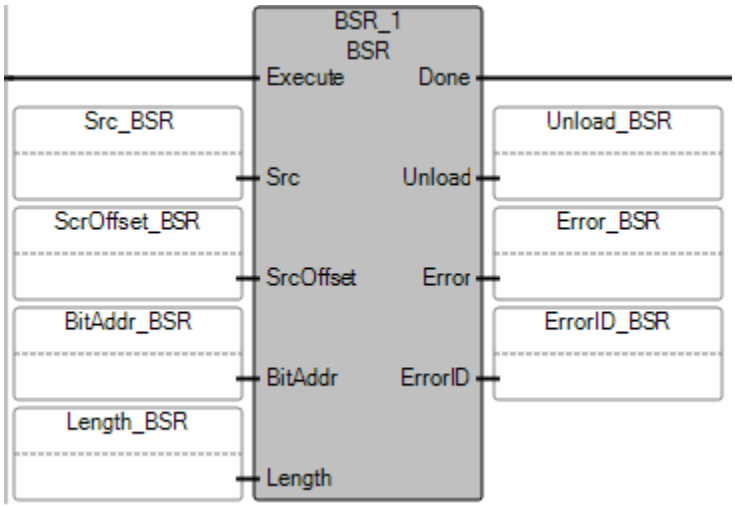
BSR 错误代码

错误代码	错误描述
01	不支持维度。
02	不支持数据类型。
03	位长度超出 2048。
04	源偏移超出数组大小。
05	位长度超出数组大小。
07	参数无效。

BSR 功能块图示例



BSR 梯形图示例



BSR 结构化文本示例




```
1  BSR_1(Execute,Scr,SrcOffset,BitAddr,Length);
2  Done_BSR := BSR_1.Done;
3  Unload_BSR := BSR_1.Unload;
4  Error_BSR := BSR_1.Error;
5  ErrorID_BSR := BSR_1.ErrorID;
```

结果

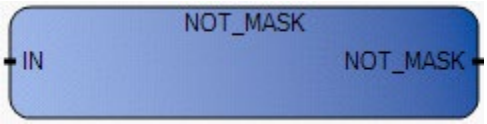
	名称	别名	逻辑值	实际值	初始值	锁定	数据类型	维度
+	BSR_1		...	...	...		BSR	
	Execute			不可用			BOOL	
+	Scr		...	...	...		DINT	[1..4]
	SrcOffset		0	不可用			UINT	
	BitAddr			不可用			BOOL	
	Length		0	不可用			UINT	
	Done_BSR			不可用			BOOL	
	Unload_BSR			不可用			BOOL	
	Error_BSR			不可用			BOOL	
	ErrorID_BSR		0	不可用			USINT	

NOT_MASK（位到位 NOT 掩码）

整型位到位取反掩码，将反转参数值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

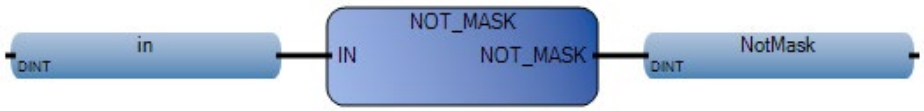
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



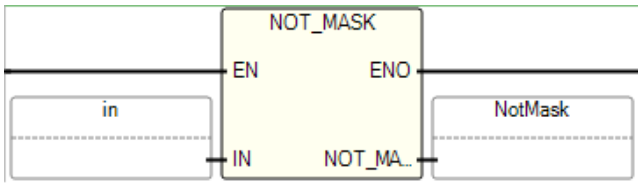
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行位到位非运算掩码计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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。
ENO	输出	BOOL	启用“输出”。 适用于梯形图编程。

NOT_MASK 功能块图示例



NOT_MASK 梯形图示例



NOT_MASK 结构化文本示例




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

(* 与之等效的 ST: *)

result := NOT_MASK (16#1234);

(* 结果为 16#FFFF_EDCB *)

结果



OR_MASK（位到位 OR 掩码）

整型 OR 位到位掩码，将启用位。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

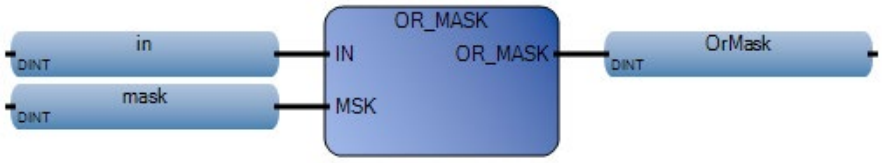
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



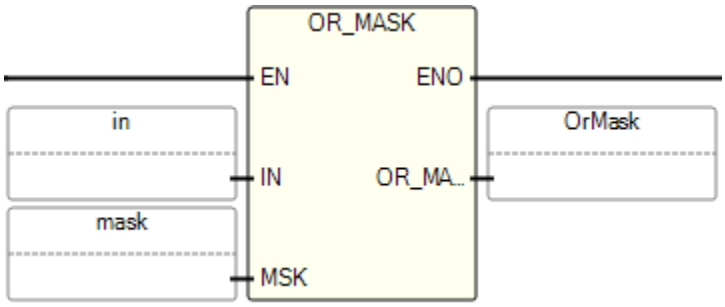
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行整型或位到位掩码计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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"。 适用于梯形图编程。

OR_MASK 功能块图示例



OR_MASK 梯形图示例



OR_MASK 结构化文本示例




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



ROL（向左旋转）

将 DINT 类型输入以圆形向左旋转 NbR 位，并用旋转的位填充右侧的位。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。





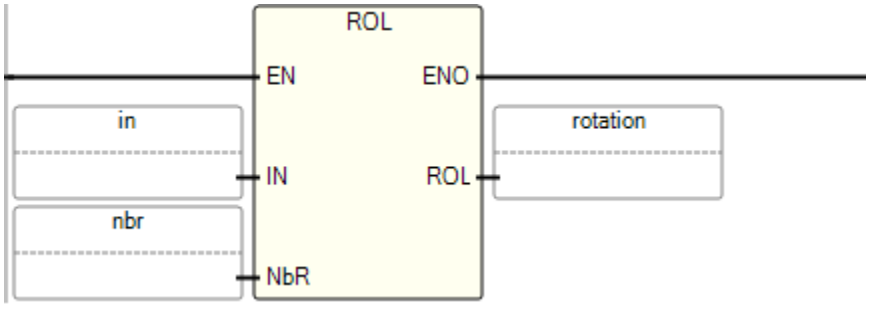
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行向左旋转位整数值计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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”。 适用于梯形图编程。

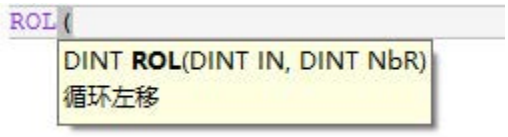
ROL 功能块图示例



ROL 梯形图示例



ROL 结构化文本示例




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

(* 与之等效的 ST: *)

result := ROL (register, 1);

(* register = 2#0100_1101_0011_0101*)

(* result = 2#1001_1010_0110_1010 *)

结果

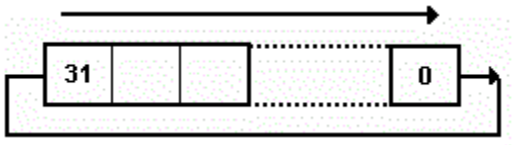


ROR（向右旋转）

将 DINT 类型输入以圆形向右旋转 NbR 位，并用旋转的位填充左侧的位。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。





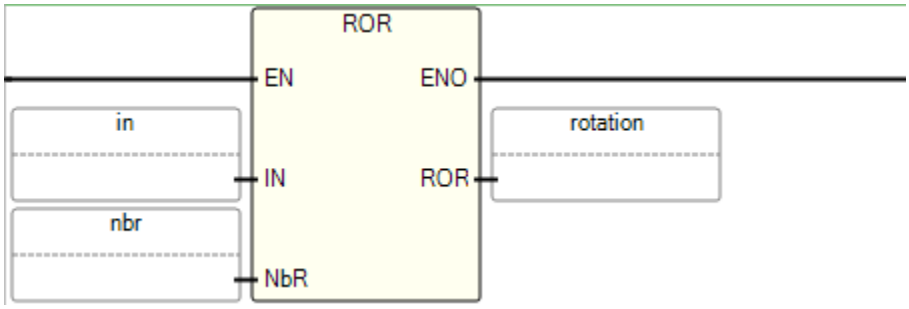
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行向右旋转位整数值计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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"。 适用于梯形图编程。

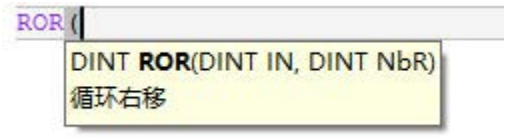
ROR 功能块图示例



ROR 梯形图示例



ROR 结构化文本示例




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

(* 与之等效的 ST：*)

result := ROR (register, 1);

(* register = 2#0100_1101_0011_0101 *)

(* result = 2#1010_0110_1001_1010 *)

结果



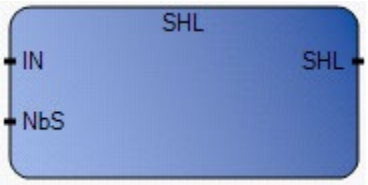
SHL（向左移动）

对于 32 位整数，将整数向左移动，并在最低有效位中置 0。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。





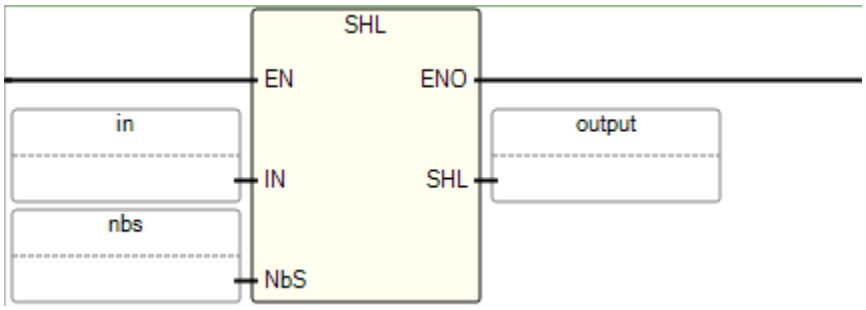
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 将整数向左移动。 FALSE - 不执行整数移动。 适用于梯形图编程。
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”。 适用于梯形图编程。

SHL 功能块图示例



SHL 梯形图示例



SHL 结构化文本示例




```
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

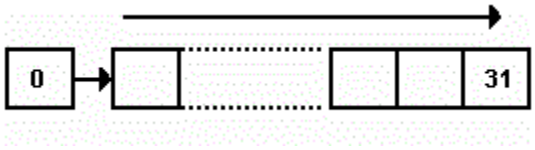


SHR（向右移动）

将整数的 32 位右移，并复制最左侧位（有效位）以填充空位。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。





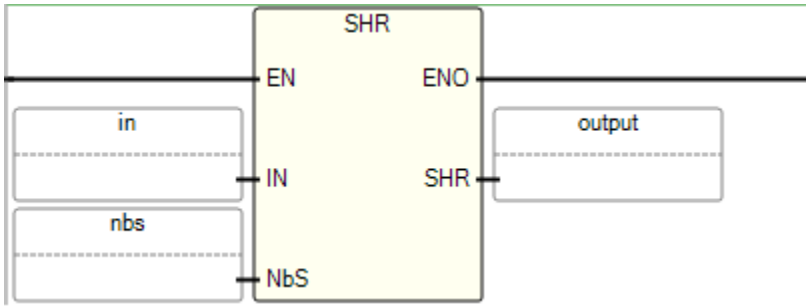
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 将整数向右移动。 FALSE - 不执行整数移动。 适用于梯形图编程。
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。
ENO	输出	BOOL	启用“输出”。 适用于梯形图编程。

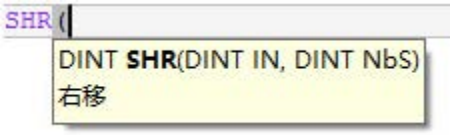
SHR 功能块图示例



SHR 梯形图示例



SHR 结构化文本示例




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

(* 与之等效的 ST: *)

result := SHR (register,1);

(* register = 2#1100_1101_0011_0101 *)

(* result = 2#0110_0110_1001_1010 *)

结果



XOR_MASK（异或掩码）

整数异或位到位掩码，返回反转的位值。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



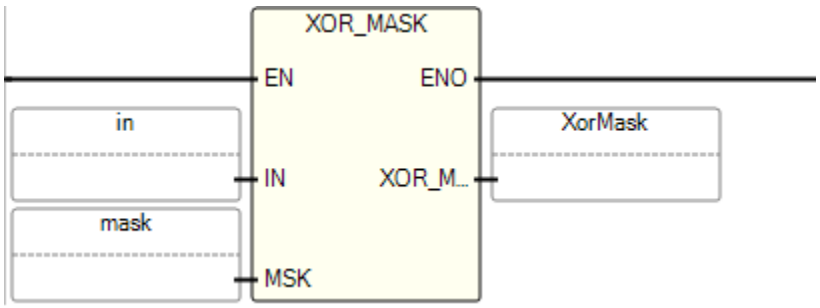
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行异或位到位掩码计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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"。 适用于梯形图编程。

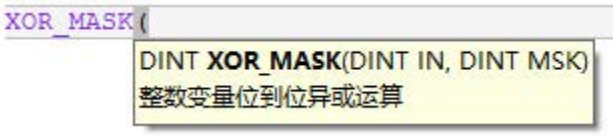
XOR_MASK 功能块图示例



XOR_MASK 梯形图示例



XOR_MASK 结构化文本示例




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



布尔指令

布尔指令用于根据输入的逻辑计算结果确定输出值。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。

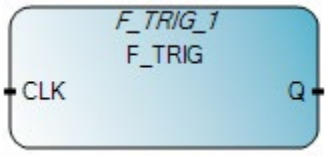
功能	描述
MUX4B 参考页数 152	四个布尔型输入之间的乘法器，输出布尔型值。
MUX8B 参考页数 148	八个布尔型输入之间的乘法器，输出布尔型值。
TTABLE 参考页数 146	根据输入组合提供输出的值。
功能块	**描述**
F_TRIG 参考页数 137	检测布尔变量的下降沿。
RS 参考页数 140	复位主导（确定指令行为时的最高优先级）双稳态。
R_TRIG 参考页数 139	检测布尔变量的上升沿。
SR 参考页数 144	置位优先指令。
运算符	**描述**
AND 参考页数 142	在两个或多个值之间执行布尔与操作。
NOT 参考页数 143	将布尔值转换为反值。
XOR 参考页数 143	两个值的布尔异或。
OR 参考页数 142	两个或更多值的布尔或。

F_TRIG（下降沿检测）

检测布尔变量的下降沿。当输入 CLK 从置位切换为清零（即，在输入 CLK 处检测到下降沿）时，F_TRIG 块会对于一个周期将输出 Q 置位。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

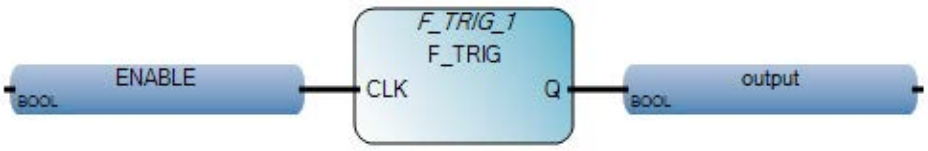


使用此表有助于确定该指令的参数值。

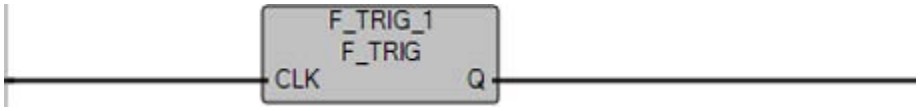
参数	参数类型	数据类型	描述
CLK	输入	BOOL	检查下降沿的输入。任何布尔变量。 TRUE = 未检测到下降沿。 FALSE = 在输入 CLK 上检测到下降沿，将输出 Q 设置为 TRUE。

参数	参数类型	数据类型	描述
Q	输出	BOOL	表示 Q 输出的状态。 TRUE = 检测到下降沿，设置又一个循环的输出 Q。 FALSE = 输出 Q 不变。

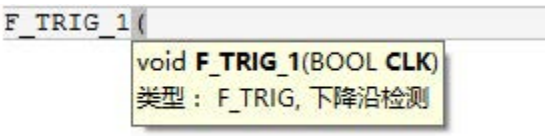
F_TRIG 功能块图示例



F_TRIG 梯形图示例



F_TRIG 结构化文本示例




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

(* 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

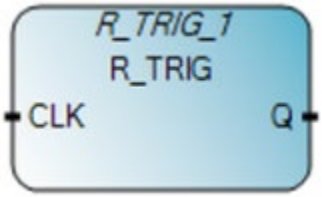


R_TRIG（上升沿探测器）

检测布尔变量的上升沿。当输入 CLK 从清零切换为置位（即，在输入 CLK 处检测到上升沿）时，R_TRIG 块会对于一个周期将输出 Q 置位。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

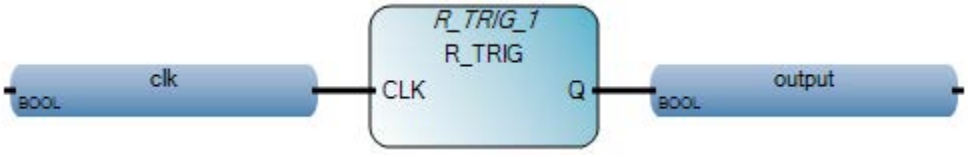
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
CLK	输入	BOOL	任何布尔变量。 TRUE - 检测到上升沿，将 Q 设置为 TRUE。 FALSE - 未检测到上升沿，将 Q 设置为 FALSE。
Q	输出	BOOL	TRUE - 当 CLK 为 TRUE 时。 FALSE - 在所有其他情况下。

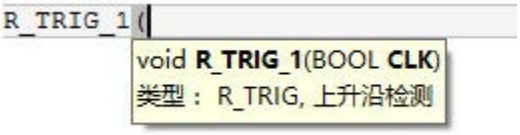
R_TRIG 功能块图示例



R_TRIG 梯形图示例



R_TRIG 结构化文本示例



(* 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



RS（复位/设置）

复位或设置主导（确定指令行为时的最高优先级）双稳态。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



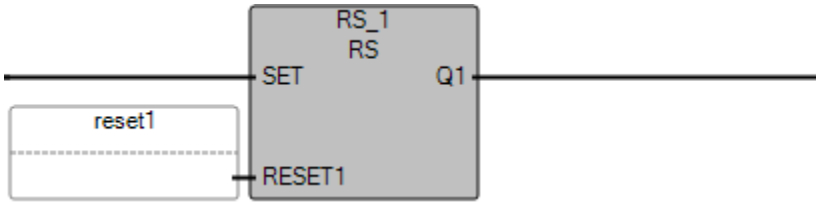
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
SET	输入	BOOL	TRUE - 将 Q1 设置为 TRUE。
RESET1	输入	BOOL	TRUE - 将 Q1 复位为 FALSE（主导）。
Q1	输出	BOOL	布尔内存状态。

RS 功能块图示例



RS 梯形图示例



RS 结构化文本示例




```
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

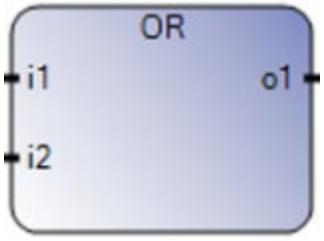


OR

执行两个或多个布尔值的逻辑 OR 运算，并返回布尔值 True（如果输入为 true），否则返回 false。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
i1	输入	BOOL	
i2	输入	BOOL	
o1	输出	BOOL	输入项的布尔或。 TRUE - 当一个或多个输入为 TRUE 时。 FALSE - 当输入为 FALSE 时。

OR 结构化文本示例

(* 与之等效的 ST：*)

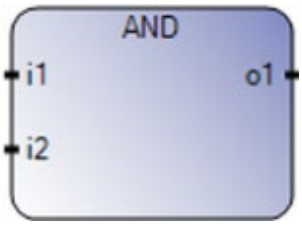
```
bo10 := bi101 OR NOT (bi102);
bo5 := (bi51 OR bi52) OR bi53;
```

AND

在两个或多个值之间执行布尔与操作。可以表示为 &。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
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 结构化文本示例

(* 与之等效的 ST：*)

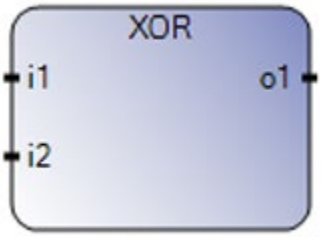
```
bo10 := bi101 AND NOT (bi102);
bo5 := (bi51 AND bi52) AND bi53;
```

XOR（异或）

执行两个布尔值的异或运算。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
i1	输入	BOOL	
i2	输入	BOOL	
o1	输出	BOOL	两个输入项的布尔异或。 TRUE - 当任一输入为 TRUE 时。 FALSE - 当两个输入均为 FALSE 或两个输入均为 TRUE 时。

XOR 结构化文本示例

(* 与之等效的 ST：*)

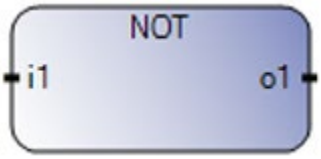
```
bo10 := bi101 XOR NOT (bi102);
bo5 := (bi51 XOR bi52) XOR bi53;
```

NOT

将布尔值转换为反值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
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 结构化文本示例

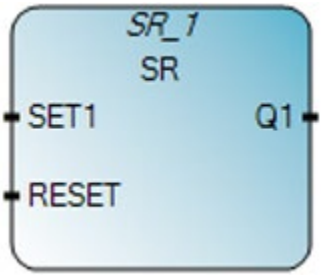
```
(* 与之等效的 ST：*)
   bo10 := NOT (bi101);
```

SR（设置/复位）

设置主导双稳态。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
SET1	输入	BOOL	TRUE - 将 Q1 设置为 TRUE（主导：确定指令行为时的最高优先级）。
RESET	输入	BOOL	TRUE - 将 Q1 复位为 FALSE。
Q1	输出	BOOL	布尔内存状态。 TRUE - 当 SET1 为 TRUE 时。 FALSE - 当 RESET 为 TRUE 时。

双稳态基准示例

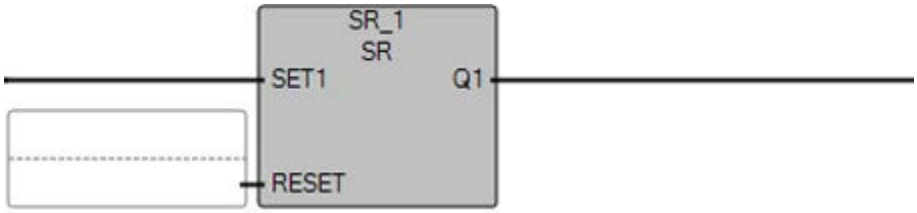
Set1	复位	Q1	结果 Q1
0	0	0	0
0	0	1	1

0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

SR 功能块图示例



SR 梯形图示例



SR 结构化文本示例




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

(* 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



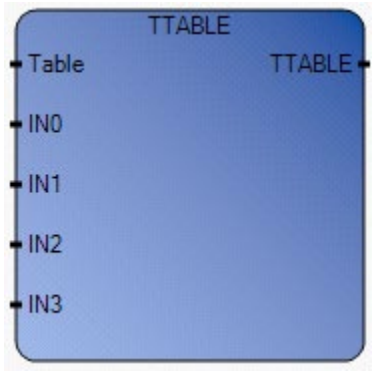
NOT（真值表）

根据输入组合提供输出的值。

如果值为 0x 参考页数 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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
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。

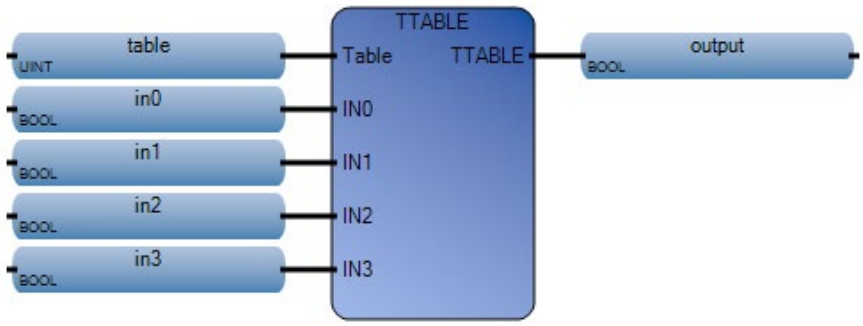
TTABLE 输入组合

TTABLE 指令拥有四个输入，因此拥有 16 种组合。可以在真值表中找到这些组合；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。

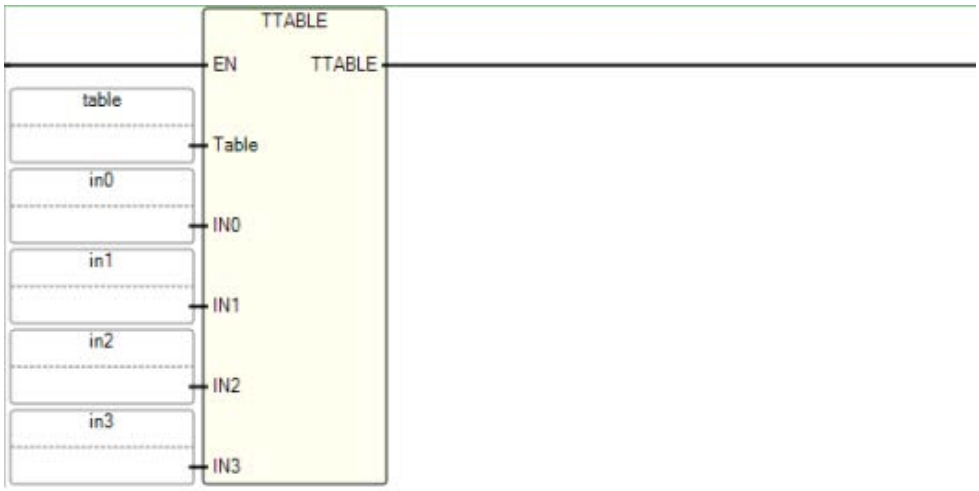
真值表组合示例。

编号	In3	In2	In1	In0
1	0	0	0	0
2	0	0	0	1
3	0	0	1	0
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 功能块图示例



TTABLE 梯形图示例



TTABLE 结构化文本示例

```
TTABLE(
```

BOOL **TTABLE**(UINT Table, BOOL IN0, BOOL IN1, BOOL IN2, BOOL IN3)
根据输入组合提供值输出。

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

结果

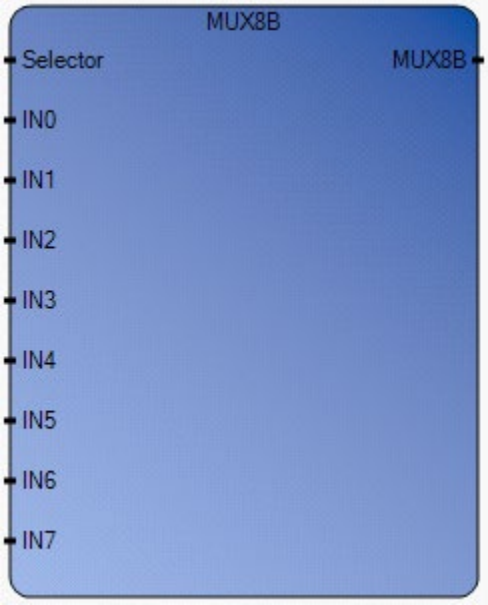


MUX8B（8 个 BOOL 输入的乘法器）

八个布尔型输入之间的乘法器，输出布尔型值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

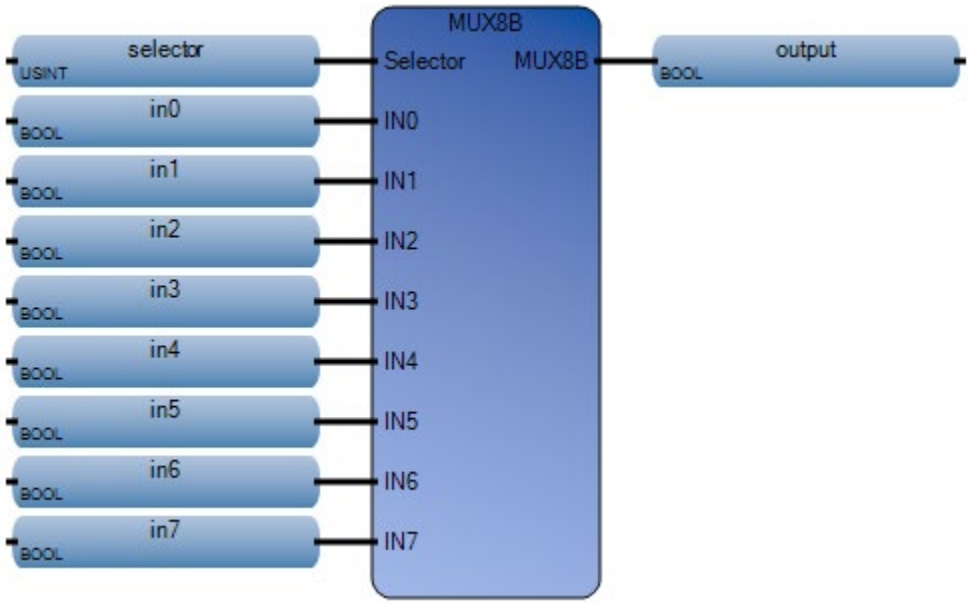


使用此表有助于确定该指令的参数值。

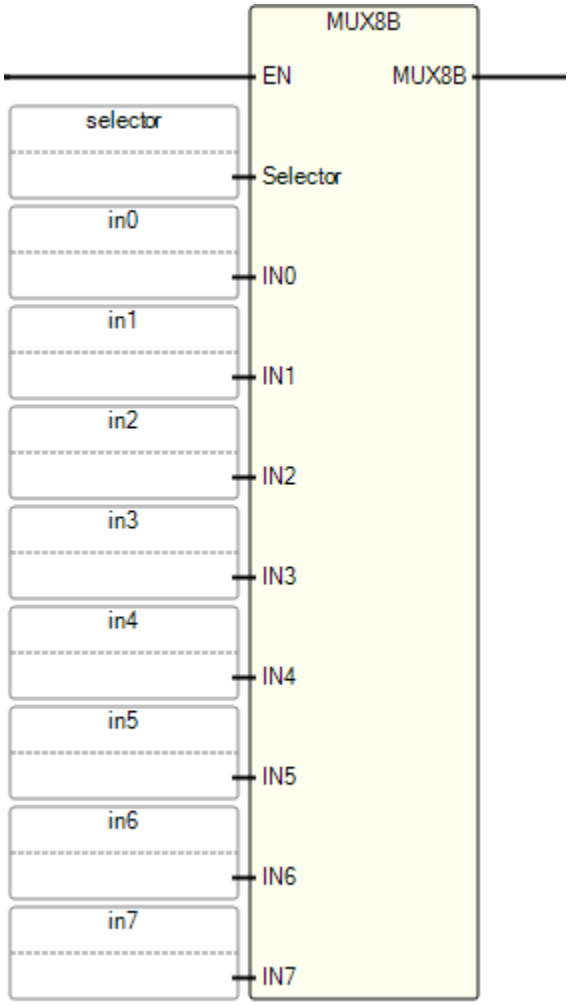
参数	参数类型	数据类型	描述
Selector	输入	USINT	选择器整数值，必须位于集合 [0...7] 内。
IN0	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 0。 FALSE - 当选择器不为 0。
IN1	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 1。 FALSE - 当选择器不为 1。
IN2	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 2。 FALSE - 当选择器不为 2。
IN3	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 3。 FALSE - 当选择器不为 3。
IN4	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 4。 FALSE - 当选择器不为 4。
IN5	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 5。 FALSE - 当选择器不为 5。
IN6	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 6。 FALSE - 当选择器不为 6。
IN7	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 7。 FALSE - 当选择器不为 7。
MUX8B	输出	BOOL	TRUE - 在下列情况下时： • In0（如果 Selector = 0） • In1（如果 Selector = 1） • In2（如果 Selector = 2） • In3（如果 Selector = 3）

参数	参数类型	数据类型	描述
			• In4（如果 Selector = 4） • In5（如果 Selector = 5） • In6（如果 Selector = 6） • In7（如果 Selector = 7） FALSE - 对于选择器的所有其他值。

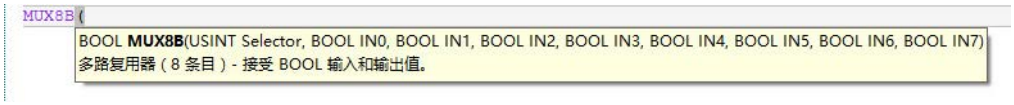
MUX8B 功能块图示例



MUX8B 梯形图示例



MUX8B 结构化文本示例



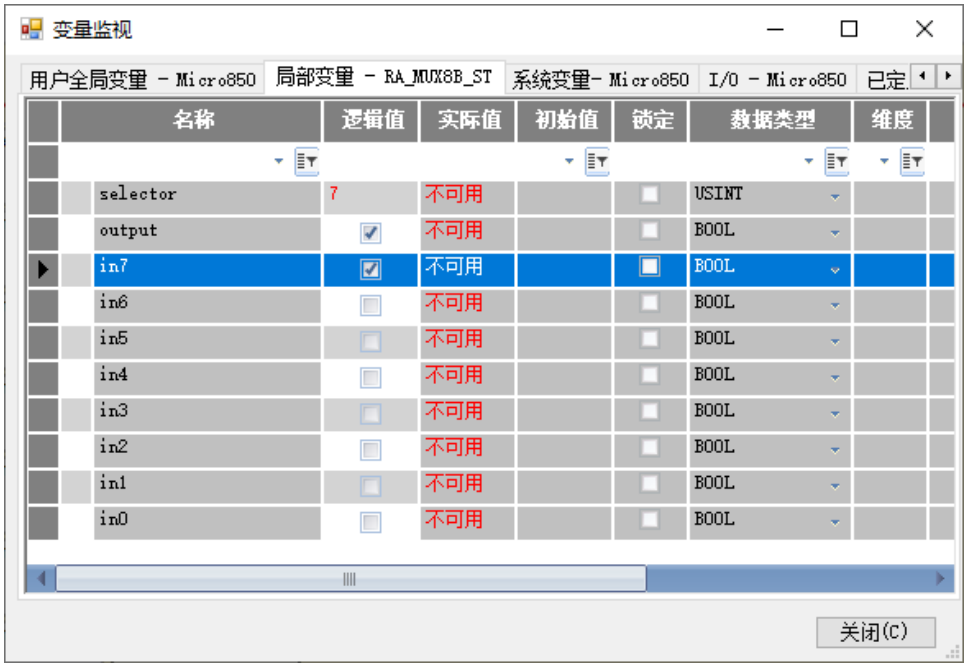

```
1  selector := 7;
2  in0 := FALSE;
3  in1 := FALSE;
4  in2 := FALSE;
5  in3 := FALSE;
6  in4 := FALSE;
7  in5 := FALSE;
8  in6 := FALSE;
9  in7 := TRUE;
10 output := MUX8B(selector, in0, in1, in2, in3, in4, in5, in6, in7);
```

(* 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

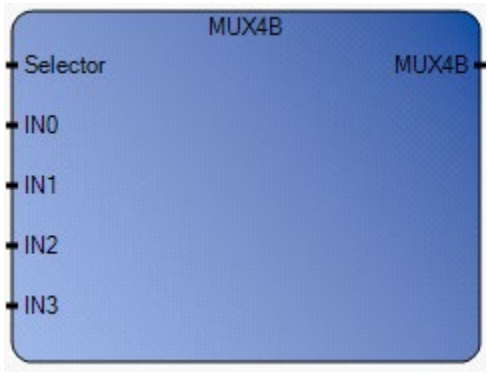


MUX4B（4 个 BOOL 输入的乘法器）

四个布尔型输入之间的乘法器，输出布尔型值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

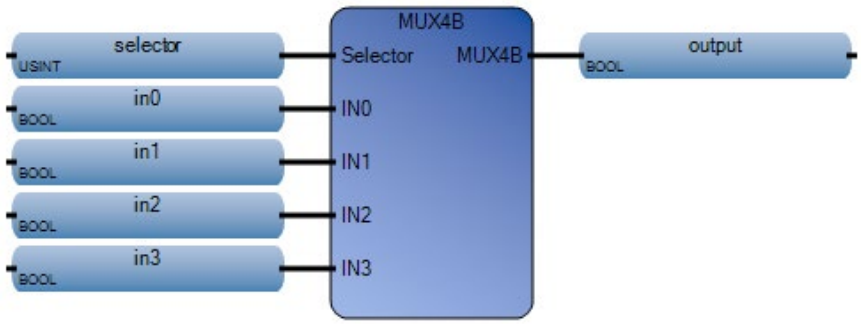


使用此表有助于确定该指令的参数值。

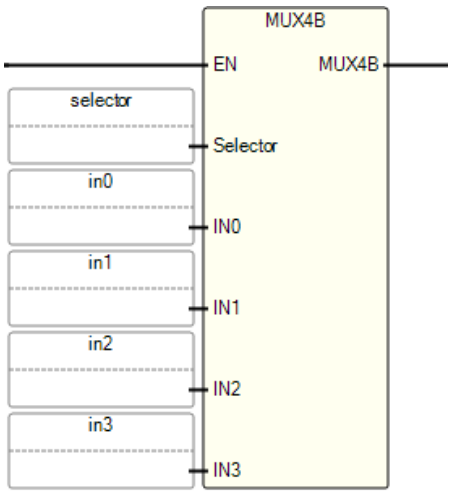
参数	参数类型	数据类型	描述
Selector	输入	USINT	选择器整数值，必须位于集合 [0...3] 内。
IN0	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 0。 FALSE - 当选择器不为 0。

参数	参数类型	数据类型	描述
IN1	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 1。 FALSE - 当选择器不为 1。
IN2	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 2。 FALSE - 当选择器不为 2。
IN3	输入	BOOL	任何 BOOL 输入值。 TRUE - 当选择器为 3。 FALSE - 当选择器不为 3。
MUX4B	输出	BOOL	TRUE - 在下列情况下时： • In0（如果 Selector = 0） • In1（如果 Selector = 1） • In2（如果 Selector = 2） • In3（如果 Selector = 3） FALSE - 对于选择器的所有其他值。

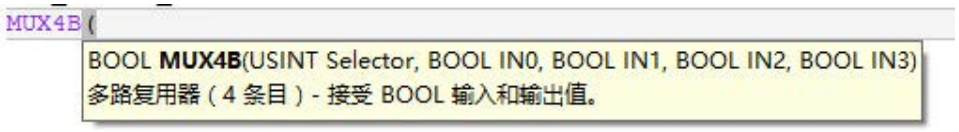
MUX4B 功能块图示例



MUX4B 梯形图示例



MUX4B 结构化文本示例




```
selector := 1;
in0 := FALSE;
in1 := TRUE;
in2 := FALSE;
in3 := FALSE;
output := MUX4B(selector, in0, in1, in2, in3);
```

(* 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



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	描述
COM_IO_WDOG 参考页数 155	监控与控制器的通信。
MSG_CIPGENERIC 参考页数 158	发送 CIP 泛型显式消息。
MSG_CIPSYMBOLIC 参考页数 165	发送 CIP 符号显式消息。
MSG_MODBUS 参考页数 168	发送 Modbus 消息。
MSG_MODBUS2 参考页数 174	通过以太网通道发送 MODBUS/TCP 消息。
MSG_PCCC 参考页数 179	通过以太网通道或串行端口发送 PCCC 消息。

COM_IO_WDOG

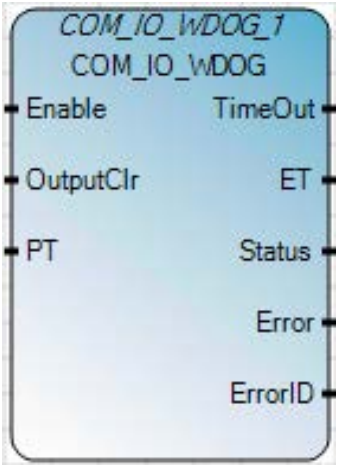
监控到控制器输入和输出的外部消息传送。例如，如果到变量 _IO_EM_DO_00 的 CIP 写入命令未在配置的超时时间内通过 EtherNet/IP 接收到，则看门狗计时器会过期，并且所有控制器输出均会被复位。

支持 EtherNet/IP、Modbus TCP 和 Modbus RTU 协议。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。 只有固件版本 12.00 及更高版本才支持它。

注意：
- 尽管可以存在该指令的多个实例，但只能启用一个实例。否则将发生错误，并且 ErrorID 设置为 2。
- 仅支持嵌入式控制器、插件或扩展数字量 I/O。



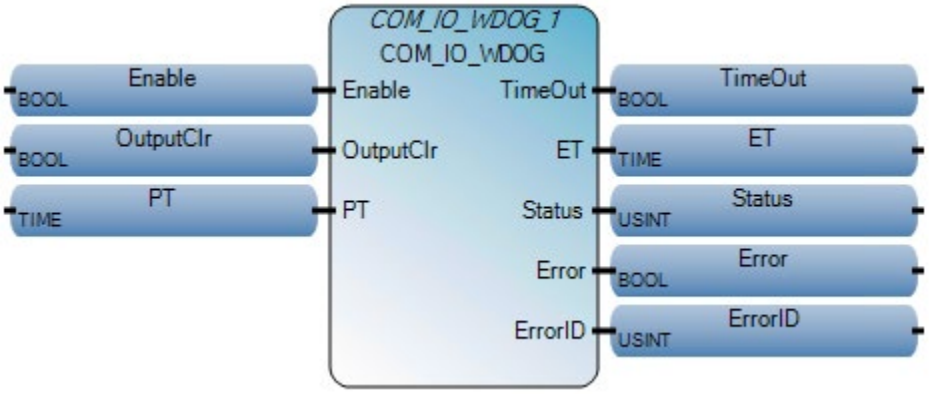
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	TRUE - TRUE 条件有三种情况。 1. 在 TRUE 的上升沿，功能块开始执行。状态位设置为 0 同时其余位被清除。然后验证以下错误条件。如果未发生错误，则继续下一步骤。 • 如果 PresetValue 小于一秒，则 Error 设置为 TRUE，ErrorID 设置为 1，设置状态 位 4，TimeOut 和 ElapsedTime 被清除。 • 如果计时器已被其他实例获取， 则 Error 设置为 TRUE，ErrorID 设置为 2，设置状态 位 4，TimeOut 和 ElapsedTime 被清除。 2. PresetTime > ElapseTime。在 Enable 设置为 TRUE 且功能块开始执行时。检查任何命令 是否 收到。如果 收到 任何 命令，则立即复位 计时器。将 ElapsedTime 设置为 0 和 设置状态位 2 的常数值。其余位被清除。 如果 未收到任何命令， 则设置状态 位 1，同时其余位被清除。 3. PresetTime = ElapsedTime。在 Enable 设置为 TRUE 且功能块开始执行时。检查是否 收到任何命令。如果收到，则立即复位计时器。将 ElapsedTime 设置为 0 和 设置状态位 2 的常数值。其余位被清除。否则设置 TimeOut 位， 设置状态 位 3，同时其余位被清除。 设置内部变量，以在扫描结束后清除数字量输出（如果配置的操作 是在扫描结束后 清除 所有数字量输出）。 FALSE - 指令未执行，同时功能块输出被清除。
OutputClr	输入	BOOL	0 - 如果发生超时，不执行任何操作。 1 - 如果发生超时，扫描结束后清除所有数字量输出 （Embedded、EXIO 和 UPM）。
PT	输入	TIME	超时前等待的持续时间。 超时值不能小于一秒，否则会发生错误。PresetTime 的最大值可以是 TIME 数据类型允许的最大值。
TimeOut	输出	BOOL	TRUE - ElapsedTime 等于 PresetTime。 FALSE - Enable 设置为 FALSE；计时器的时间未经过，或发生错误。
ET	输出	TIME	当前已经过的时间。 值的可能范围是从 0 ms 到 1193h2m47s294ms。
Status	输出	USINT	功能块的状态。 位 0 - Enable 位 1 - 计时器正在运行，还未收到任何输出或输入。 位 2 - 收到输出或输入 。 位 3 - 发生超时。未收到任何输出或输入命令。 位 4 - 发生错误。 其他位保留。
Error	输出	BOOL	指示发生了错误。
ErrorID	输出	USINT	当发生错误时，ErrorID 包含错误代码。

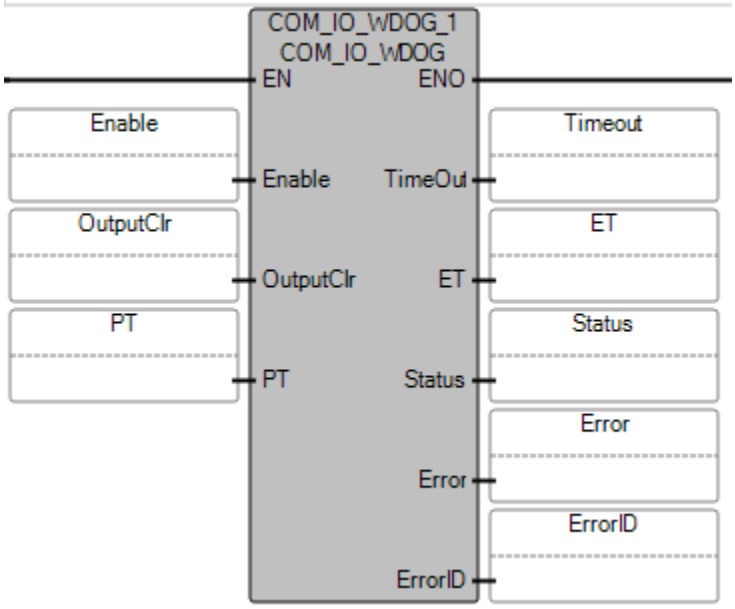
COM_IO_WDOG 错误代码

ErrorID 代码	错误描述
1	PresetTime 小于一秒。
2	另一 COM_IO_WDOG 功能块实例已在执行。

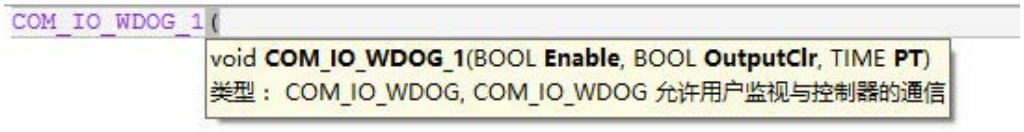
COM_IO_WDOG 功能块图示例



COM_IO_WDOG 梯形图示例



COM_IO_WDOG 结构化文本示例




```
1  COM_IO_WDOG_1(Enable, OutputCrl, PT);
2   Timeout :=COM_IO_WDOG_1.Timeout;
3   ET :=COM_IO_WDOG_1.ET;
4   Status :=COM_IO_WDOG_1.Status;
5   Error :=COM_IO_WDOG_1.Error;
6   ErrorID :=COM_IO_WDOG_1.ErrorID;
```

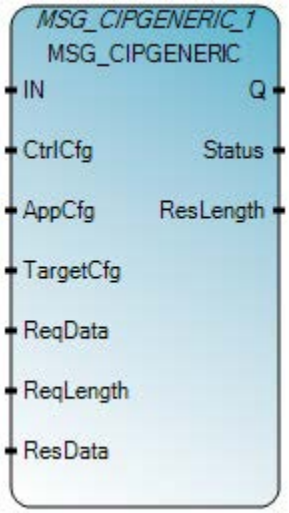
MSG_CIPGENERIC（通用工业协议泛型消息）

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。

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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

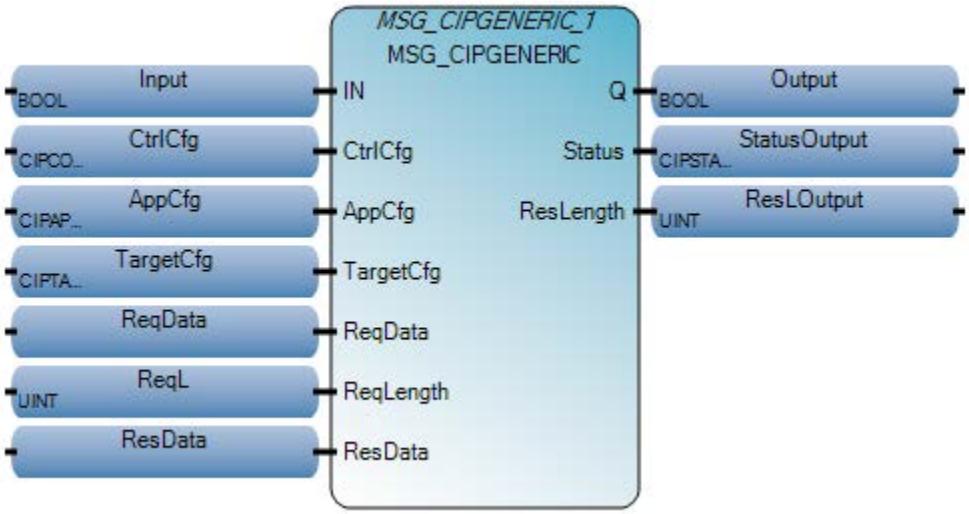
该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



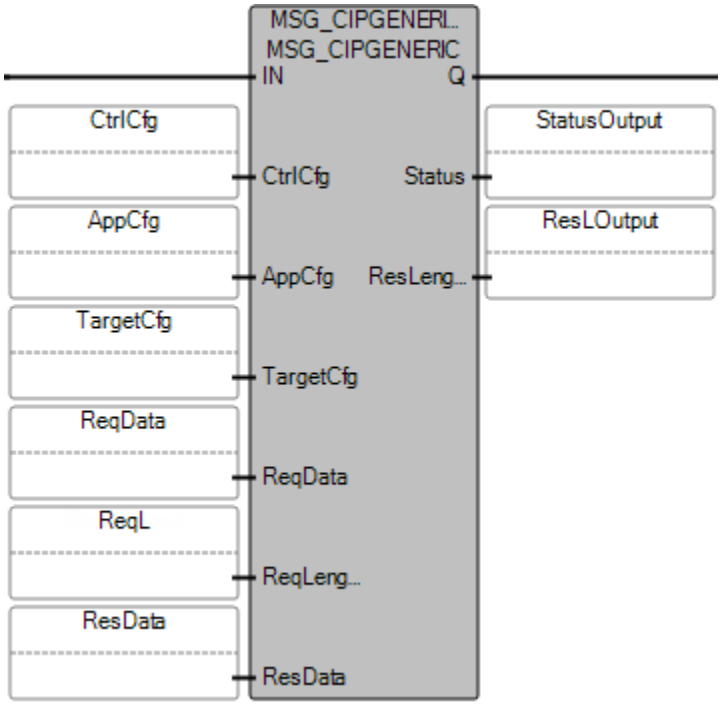
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 未检测到上升沿，空闲。
CtrlCfg	输入	CIPCONTROLCFG	指令块执行控制配置。使用 CIPCONTROLCFG 数据类型 参考页数 160 参数定义 CtrlCfg。
AppCfg	输入	CIPAPPCFG	CIP 服务和应用程序路径 (EPATH) 配置。 使用 CIPAPPCFG 数据类型 参考页数 160 参数定义 AppCfg。
TargetCfg	输入	CIPTARGETCFG	目标设备配置。 使用 CIPTARGETCFG 数据类型 参考页数 163 参数定义 TargetCfg。
ReqData	输入	USINT[1..1]	CIP 消息请求数据。数组大小应大于 ReqLength 大小。
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。数组大小应大于 ReqLength 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	从程序扫描中同步更新此说明的输出。输出 Q 无法用于重新触发说明，因 IN 已进行沿触发。 TRUE - MSG 指令已成功完成。 FALSE - MSG 指令未完成。
Status	输出	CIPSTATUS	指令块状态。 触发或者重新触发 MSG 时，Status 中的所有元素均会重置。 状态输出在 CIPSTATUS 数据类型 参考页数 161 中定义。
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 功能块图示例



MSG_CIPGENERIC 梯形图示例



MSG_CIPGENERIC 结构化文本示例



CIPAPPCFG 数据类型

使用此表帮助定义 CIPAPPCFG 数据类型的参数。

参数	数据类型	描述
Service	USINT	服务代码： 1 - 127
Class	UINT	逻辑段类 ID 值： 1 - 65535
Instance	UDINT	逻辑段实例 ID 值： 0 - 4294967295
Attribute	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

CIPCONTROLCFG 数据类型

使用此表帮助确定 CIPCONTROLCFG 数据类型的参数值。

参数	数据类型	描述
Cancel	BOOL	TRUE - 取消功能块的执行。 启用消息后将清除位。 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），则消息执行会取消并设置 ER 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。

CIP 消息触发

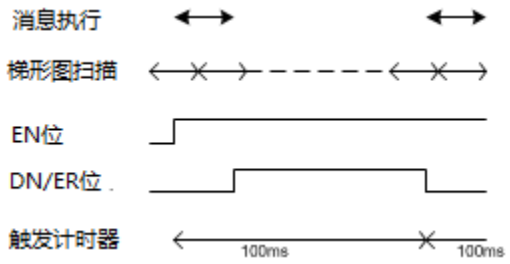
通过将 TriggerType 参数设置为非零值，可以定期触发 CIP 消息。

使用此表帮助定义触发类型参数的操作。

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。



CIPSTATUS 数据类型

使用该表帮助确定 CIPSTATUS 数据类型的参数值。

参数	数据类型	描述
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。 在 CIPSTATUS 错误代码 参考页数 162 中定义 ErrorID。
SubErrorID	UINT	子错误代码值。 在 CIPSTATUS 错误代码 参考页数 162 中定义 SubErrorID。
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 • 位 5：CIPCONN - CIP 连接关闭 • 位 6：EIPSESS - EIP 会话关闭 • 其他位保留 为 CIPSTATUS 状态位 参考页数 161 定义 StatusBits。

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	描述	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	启用等待	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	错误	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	完成	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)。
5	CIPCONN	完成	当通信的 CIP 连接关闭时置位。当 ConnClose 为 True 时，CIPCONN 位适用，其他情况下 CIPCONN 位为 False。CIPCONN 位还用于串行、以太网和 USB。
6	EIPSESS	完成	当通信的封装 CIP 会话关闭时置位。当 ConnClose 为 True 时，EIPSESS 位适用，其他情况下 EIPSESS 位为 False。该位仅用于以太网。

CIPSTATUS 错误代码

使用此表帮助确定在设置 ER 位时 CIPSTATUS 参数的 ErrorID 和 SubErrorID 字段的参数值。

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。 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)。
	52	错误段类型值。
	53	错误 UCCM 超时值。 如果封装超时值小于 UCCM 超时，或封装超时和 UCCM 超时之间的差异小于或等于一秒，将报告错误 0x21(subError 0x35)。
	54	连接的超时值错误。 如果封装超时值小于 CONNECTED 消息超时，或封装超时和 CONNECTED 消息超时之间的差异小于或等于一秒，将报告错误 0x21(subError 0x36)。
55	超时相关错误	
	112	在消息等待队列中等待时，消息超时。
	113	在等待建立与链路层的连接时，消息超时。
	114	在等待传输至链路层时，消息超时。
	115	在等待来自链路层的响应时，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。	
223	链接地址不可用。正在进行 TCP/IP 或以太网配置更改。	
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 • 串行端口插件已显示，但未配置	

CIPTARGETCFG 数据类型

使用此表帮助确定 CIPTARGETCFG 数据类型的参数值。

参数	数据类型	描述
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
CipConnMode	USINT	CIP 连接类型。 • 0 - 断开（默认） • 1 - Class3 连接
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 毫秒。 • 设为 0 以使用默认值 3000 毫秒（3 秒）。 • 设为小于 250 毫秒的值将被设为 250 毫秒（最小值）。 • 设为大于 10,000 毫秒的值将被设为 10,000 毫秒（最大值）。
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 毫秒。 • 设为 0 以使用默认值 10,000 毫秒（10 秒）。 • 设为小于 800 的值将被设为 800 毫秒（最小值）。 • 设为大于 10,000 毫秒的值将被设为 10,000 毫秒（最大值）。
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）。

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	描述
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。

字符串元素	描述
"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。

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。
当 ConnClose 为 true 时，消息执行完成时关闭消息连接和 EIP 会话。	如果一个以上的消息共享相同连接，则最后一条消息完成时关闭连接。
如果不与任何活动消息关联的 CIP 或 EIP 连接空闲 x 秒钟，则将其关闭。 其中 x 是可配置的封装不活动超时值，可以使用 CIP 设置服务设置。	有关 CIP 设置服务的详细信息，请参见 CIP 规范第 II 卷 TCP/IP 对象。
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。

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

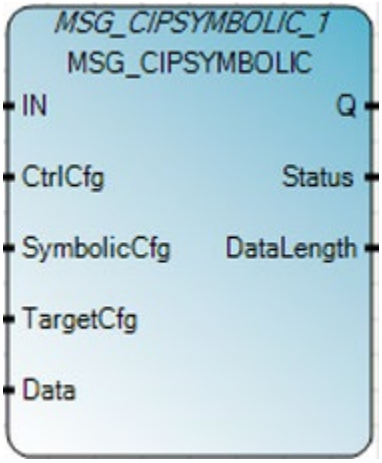
操作	结果
ConnMsgTimeout 计时器处于活动状态	UcmmTimeout 计时器已禁用
连接请求已完成	UcmmTimeout 计时器已重新激活

MSG_CIPSYMBOLIC（通用工业协议符号消息）

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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



MSG_CIPSYMBOLIC 操作

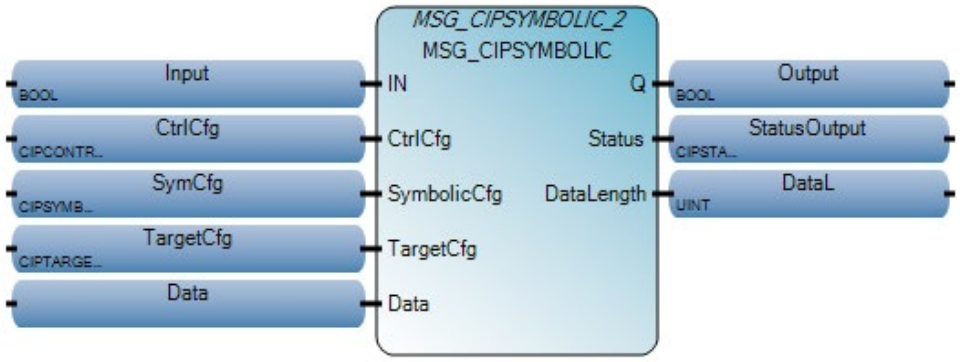
启用该功能块后，读取操作的接收缓冲区会在 Enable 的上升沿被清除。

参数

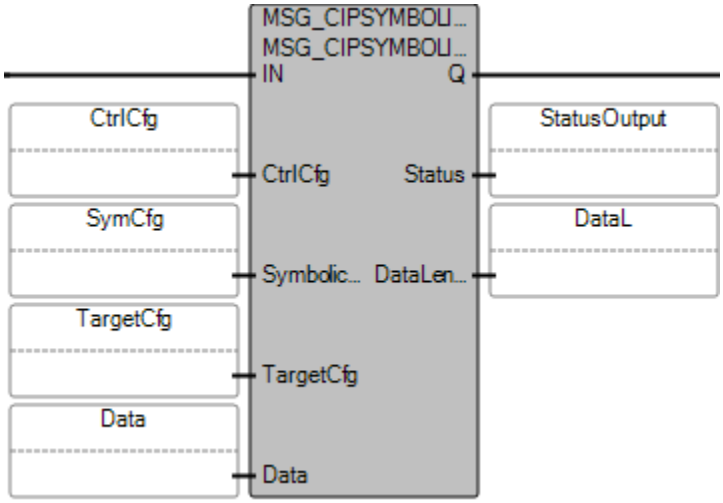
参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 未检测到上升沿，空闲。
CtrlCfg	输入	CIPCONTROLCFG	指令块执行控制配置。使用 CIPCONTROLCFG 数据类型 参考页数 160 参数定义 CtrlCfg。
SymbolicCfg	输入	CIPSYMBOLICCFG 参考页数 166	读取和写入符号信息。
TargetCfg	输入	CIPTARGETCFG	目标设备配置。 使用 CIPTARGETCFG 数据类型 参考页数 163 参数定义 TargetCfg。
Data	输入	USINT[1..1]	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	从程序扫描中同步更新此说明的输出。输出 Q 无法用于重新触发说明，因 IN 已进行沿触发。 TRUE - MSG 指令已成功完成。 FALSE - MSG 指令未完成。

参数	参数类型	数据类型	描述
Status	输出	CIPSTATUS	功能块执行状态 触发或者重新触发 MSG 时，“状态”中的所有元素均会复位。 状态输出在 CIPSTATUS 数据类型 参考页数 161 中定义。
DataLength	输出	UDINT	“读取”服务的数据字节数。对于“写入”服务，它是 0。 触发或重新触发 MSG 时，MSG 读取命令的 DataLength 将复位为 0。

MSG_CIPSYMBOLIC 功能块图示例



MSG_CIPSYMBOLIC 梯形图示例



MSG_CIPSYMBOLIC 结构化文本示例

```
MSG_CIPGENERIC_1(in1,Ctr11,appa,target1,request1,length1,data1);
out1 := MSG_CIPGENERIC_1.Q;
status1 := MSG_CIPGENERIC_1.Status;
resleng1 := MSG_CIPGENERIC_1.Resleng1;

MSG_CIPSYMBOLIC_2(
```

void **MSG_CIPSYMBOLIC_2**(BOOL **IN**, CIPCONTROLCFG **CtrlCfg**, CIPSYMBOLICCFG **SymbolicCfg**, CIPTARGETCFG **TargetCfg**, USINT[1..1] **Data**)
类型： MSG_CIPSYMBOLIC, 发送 CIP 符号消息。

CIPSYMBOLICCFG 数据类型

使用此表帮助确定 CIPSYMBOLICCFG 数据类型的参数值。

参数	数据类型	描述

Service	USINT	服务代码： • 0 - 读取（默认） • 1 - 写入
Symbol	STRING	要读取/写入的变量的名称。 • 最多 80 个字符。 • 字段不能为空。 在符号读取/写入语法 参考页数 167 中定义的符号语法。
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。
Type	User-defined	目标变量的用户自定义数据类型。 在符号数据类型支持中定义的类型。
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。		

符号数据类型支持

使用此表帮助确定 MSG_CIPSYMBOLIC 支持的数据类型。

数据类型	数据类型值（十六进制）	描述
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
STRING	218 (0xDA)	字符串

符号读取/写入语法

语法定义有效读取/写入指令块符号的组合。

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

使用此表帮助定义符号的有效语法。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
数组	<符号名称>[dim3, dim2, dim1] （受支持的最大维度为 3。	MyTag1[0] MyTag2[3,6] MyTag3[1,0,4]
结构	<符号名称>.<结构字段的符号名称>	MyTag4.time.year MyTag5.local.time[1].year

已支持的 CIP 串行功能数据包大小

对于 Micro830、Micro850 和 Micro870 控制器，内置串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 和 Micro870 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。

重要事项 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。

MSG_MODBUS（modbus 消息）

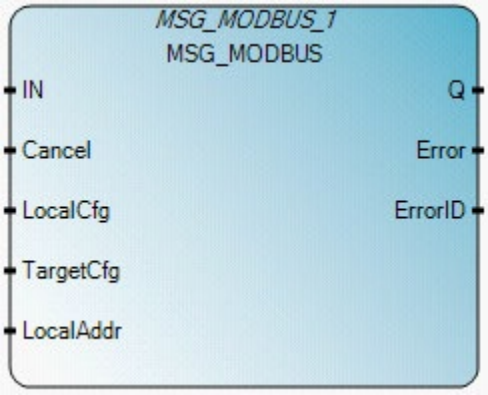
通过串行端口发送 Modbus 消息。

操作详细信息：

- 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。
- 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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 未检测到上升沿，不启动。
Cancel	输入	BOOL	TRUE - 取消指令块的执行。 FALSE - 当 IN 为 TRUE 时。 取消输入为主导。
LocalCfg	输入	MODBUSLOCPARA	定义结构输入（本地设备）。 使用 MODBUSLOCPARA 数据类型 参考页数 171 定义本地设备的输入结构。
TargetCfg	输入	MODBUSTARPARA	定义结构输入（目标设备）。 使用 MODBUSTARPARA 数据类型 参考页数 173 定义目标设备的输入结构。
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）。
Q	输出	BOOL	从程序扫描中同步更新此说明的输出。输出 Q 无法用于重新触发说明，因 IN 已进行沿触发。 TRUE - MSG 指令已成功完成。 FALSE - MSG 指令未完成。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。在 MSG_MODBUS 错误代码中定义该指令的错误。

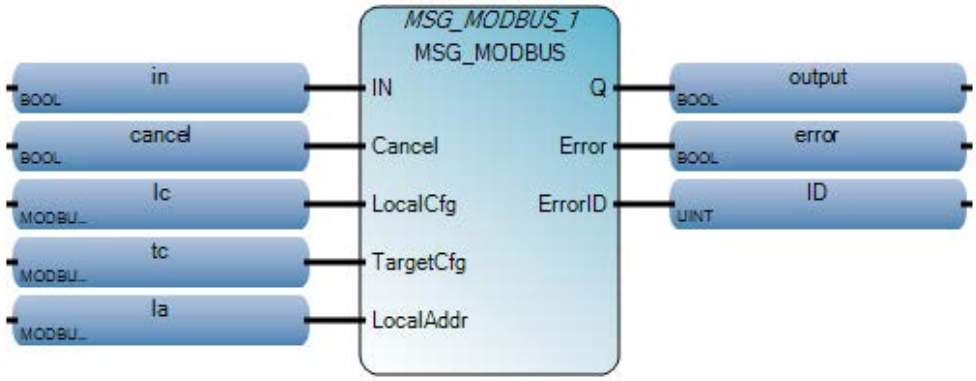
MSG_MODBUS 错误代码

该表介绍了 MSG_MODBUS 的错误代码。

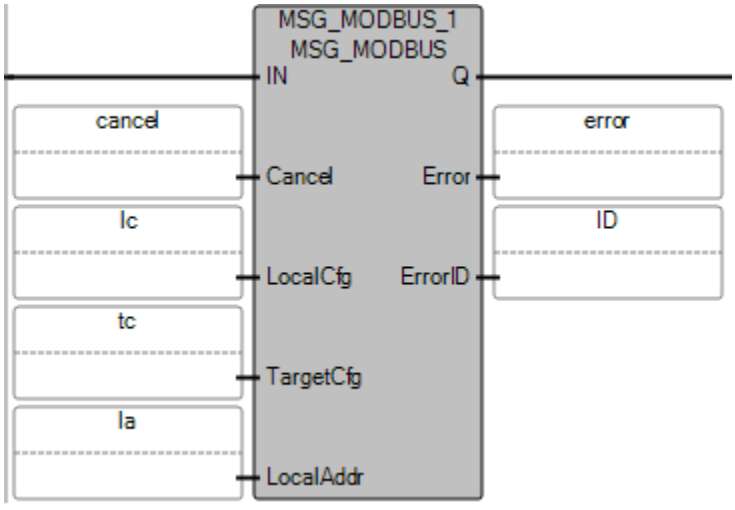
错误代码	错误描述
3	TriggerType 的值从 2 - 255 发生变更。
20	本地通信驱动程序与 MSG 指令不兼容。
21	存在本地通道配置参数错误。
22	Target 或 Local Bridge 地址高于最大节点地址。
33	存在错误的 MSG 文件参数。
54	缺少调制解调器。
55	消息在本地处理器中超时。链路层超时。

错误代码	错误描述
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	否定应答。
136	内存奇偶校验错误。
137	非标准回复。
255	通道已关闭。

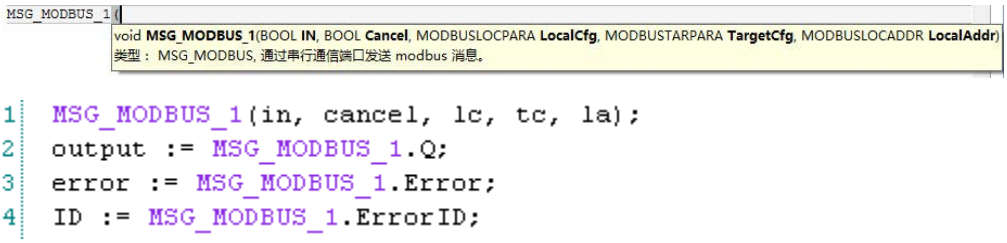
MSG_MODBUS 功能块图示例



MSG_MODBUS 梯形图示例



MSG_MODBUS 结构化文本示例



MODBUSLOCPARA 数据类型

使用此表帮助确定 MODBUSLOCPARA 数据类型的参数值。

参数	数据类型	描述
Channel	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：自定义命令支持。 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
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

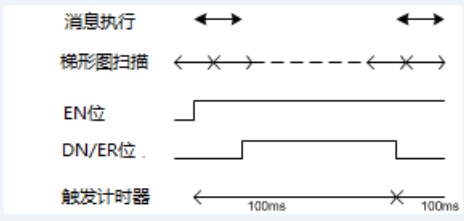
通过将 TriggerType 参数设置为非零值，可以定期触发 Modbus 消息。

该表描述了当与 MSG_MODBUS 参考页数 168 功能块一起使用时的 TriggerType 参数行为。

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。

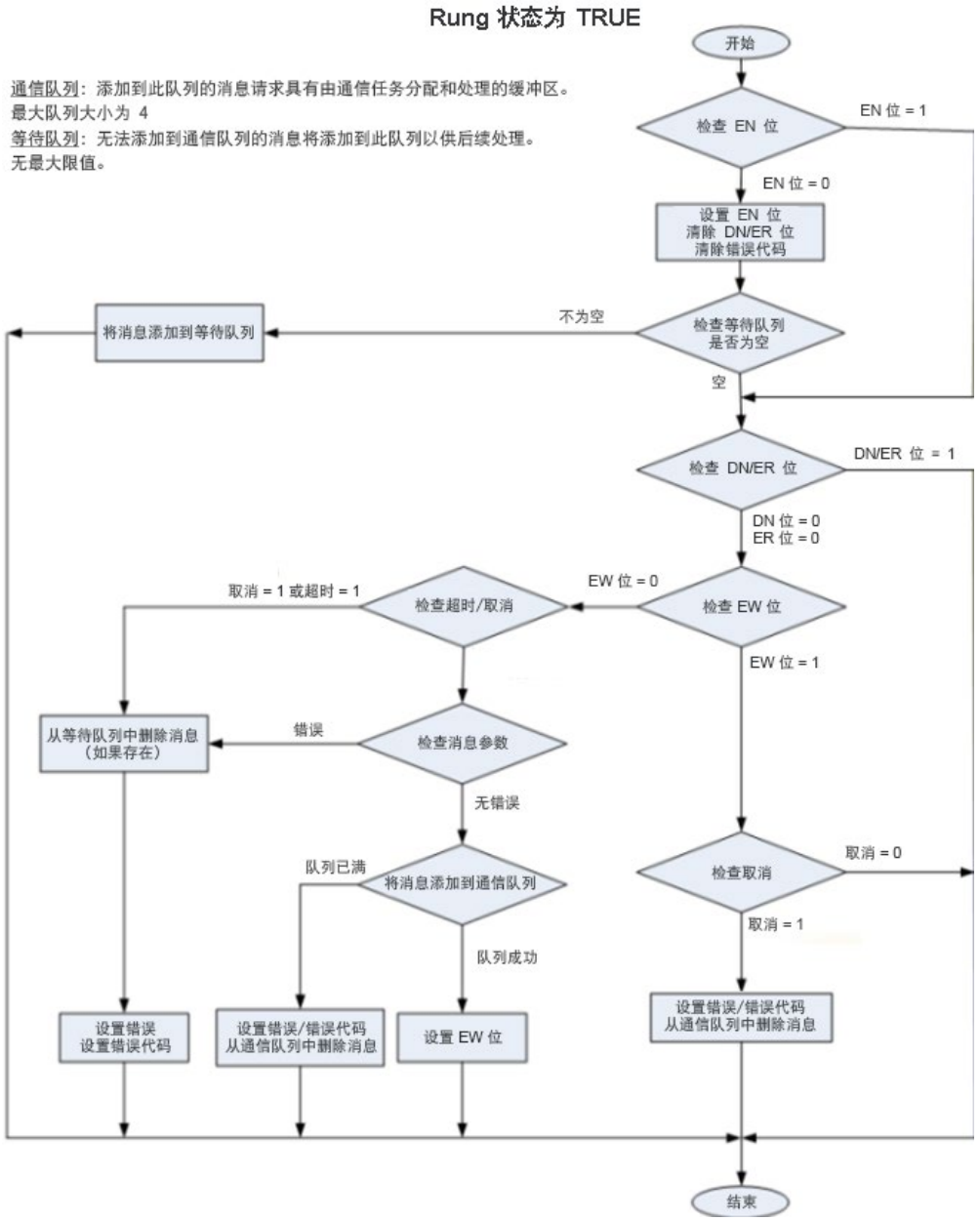
示例：消息触发

在下例中，TriggerType 值设为 100。



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。



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。

MODBUSTARPARA 数据类型

下表描述了 MODBUSTARPARA 数据类型。

参数	数据类型	描述
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。

MSG_MODBUS2（MODBUS/TCP 消息）

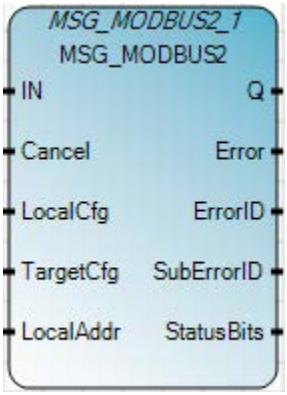
通过以太网通道发送 MODBUS/TCP 消息。

操作详细信息：

- 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。
- 启用 MSG_MODBUS2 后，读取操作的接收缓冲区会在 Enable 的上升沿被清除。
- 如果取消 MSG_MODBUS2 指令的执行，不会保证取消要求发出的消息，但可保证不处理响应。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro850 和 Micro870 控制器。对于 Micro800 Simulator，此指令可以添加到程序，但只是占位符，用于防止在控制器更改期间删除指令。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 未检测到上升沿，空闲。
Cancel	输入	BOOL	TRUE - 取消指令块的执行。如果取消 MSG_MODBUS2 指令的执行，不会保证取消要求发出的消息，但可保证不处理响应。 FALSE - 当 IN 为 TRUE 时。 取消输入为主导。
LocalCfg	输入	MODBUS2LOCPARA	定义结构输入（本地设备）。 使用 MODBUS2LOCPARA 数据类型 参考页数 176 定义本地设备的输入结构。
TargetCfg	输入	MODBUS2TARPARA	定义结构输入（目标设备）。 使用 MODBUS2TARPARA 数据类型 参考页数 177 定义目标设备的输入结构。
LocalAddr	输入	MODBUSLOCADDR	MODBUSLOCADDR 数据类型为 125 字数组。 LocalAddr 使用情况： • 对于读取命令，存储 Modbus 从站返回的数据（1-125 个字）。 • 对于写入命令，缓冲要发送到 Modbus 从站的数据（1-125 个字）。
Q	输出	BOOL	从程序扫描中同步更新此说明的输出。输出 Q 无法用于重新触发说明，因 IN 已进行沿触发。 TRUE - MSG 指令已成功完成。 FALSE - MSG 指令未完成。
Error	输出	BOOL	表示检测到错误。 TRUE - 发生错误。 FALSE - 没有错误。

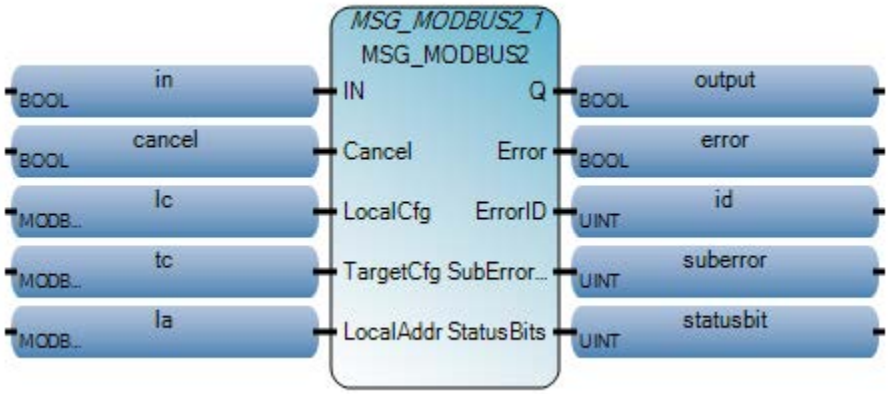
参数	参数类型	数据类型	描述
ErrorID	输出	UINT	标识错误的唯一数字。在 Modbus2 错误代码中定义该指令的错误。
SuberrorID	输出	UINT	用于确认状态位： • 位 0：EN - Enable • 位 1：EW - Enable Wait • 位 2：ST - Start • 位 3：ER - Error • 位 4：DN - Done 其他位保留。
StatusBits	输出	UINT	当 Error 为 TRUE 时的 SubError 代码值。 触发或重新触发 MSG 时，将清除先前设置的 SubErrorID。

MSG_MODBUS2 错误和子错误代码

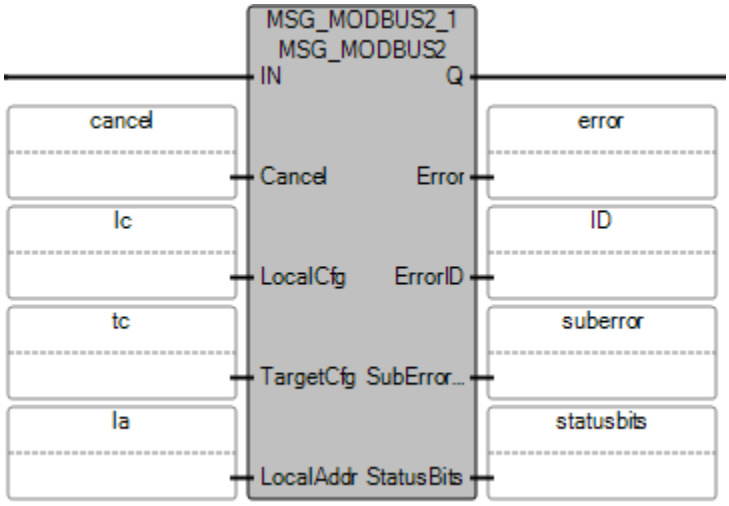
如果已设置 ER 位，则 ErrorID 和 SubErrorID 字段将显示以下错误代码。

错误 ID	SubErrorID	描述
33	**与参数配置有关的错误**	
	32	通道编号错误。
	37	元素计数错误。
	38	数据地址错误。
55	**超时相关错误**	
	112	在消息等待队列中等待时，消息超时。
	113	在等待建立与链路层的连接时，消息超时。
	114	在等待传输至链路层时，消息超时。
	115	在等待来自链路层的响应时，消息超时。
69	**与服务器响应格式有关的错误代码。**	
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）。	
222	无法在超时前建立网络连接。	
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。当以太网电缆已断开或检测不到 IP 地址时，也可能发生此情况。	
	从站响应错误代码	
129	非法函数代码	
130	非法数据地址	
131	非法数据值	
132	服务器故障	
133	确认	
134	否定应答	
136	内存奇偶校验错误	
137	非标准回复错误代码。可以在 SubErrorID 中找到实际的错误代码。	

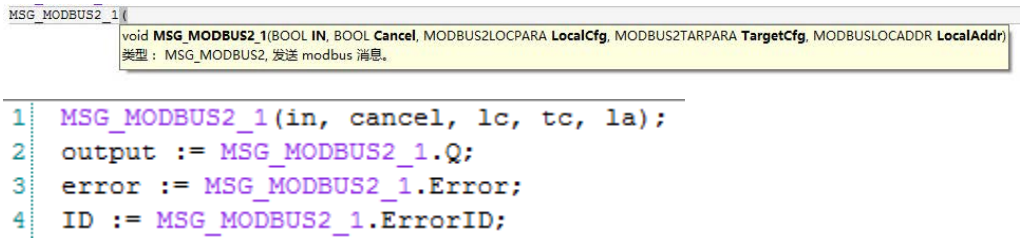
MSG_MODBUS2 功能块图示例



MSG_MODBUS2 梯形图示例



MSG_MODBUS2 结构化文本示例



MODBUS2LOCPARA 数据类型

使用此表帮助确定 MODBUS2LOCPARA 数据类型的参数值。

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

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。 请参见下面的 MSG_MODBUS2 消息触发。
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：自定义命令支持 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。如果使用自定义命令，则 LocalCfg:ElementCnt 包含接收到的字节数。 响应将接收到本地地址数据并覆盖请求数据。 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
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。

该表描述了在将 TriggerType 参数与 MSG_MODBUS2 参考页数 174 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。

MODBUS2TARPARA 数据类型

使用此表帮助确定 MODBUS2TARPARA 数据类型的参数值。

参数	数据类型	描述
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
Port	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 3,000。 • 将小于 250（最小值）的值设置为 250。 • 将大于 10,000（最大值）的值设置为 10,000。 请参见 Modbus/TCP 消息超时计时器。
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 5,000。 • 将小于 250（最小值）的值设置为 250。 • 将大于 10,000（最大值）的值设置为 10,000。 请参见 Modbus/TCP 消息超时计时器。
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]。 请参见 Modbus/TCP 消息连接。

Modbus/TCP 消息超时计时器

该表描述了基于消息请求和状态的 MsgTimeOut 和 ConnTimeOut 行为 。

操作	结果
消息已启用。	激活 MsgTimeOut 计时器。
已请求 TCP 连接。	激活 ConnTimeOut 计时器。
ConnTimeOut 计时器处于活动状态。	禁用 MsgTimeOut 计时器。
连接请求已完成。	重新激活 MsgTimeOut 计时器。

Modbus/TCP 消息连接

Modbus/TCP 客户端最多支持 16 个连接。该表描述了 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。

MSG_PCCC（可编程控制器通信命令消息）

通过以太网通道或串行端口发送 PCCC 消息。

支持的命令：SLC 类型读取和 SLC 类型写入。

操作详细信息：

- 通过以太网通道发送 PCCC 消息，并且队列中有 4 个以上的功能块请求用于 CIP_GENERIC_FB、CIP_SYMBOLIC 或 PCCC_MSG_FB 时，每个通道可在一次扫描中最多处理 4 条消息 。
  对于梯形图程序，将在梯形扫描结束时执行消息请求。
- 通过串行端口发送 PCCC 消息，并且队列中有 8 个以上的功能块请求用于串行通信时，所有通道可在一次扫描中总共处理 8 条消息。
  对于梯形图程序，将在梯形扫描结束时执行消息请求。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro870 L70E 控制器（自版本 22.00 开始）和 Micro850 L50E 控制器（自版本 23.00 开始）。

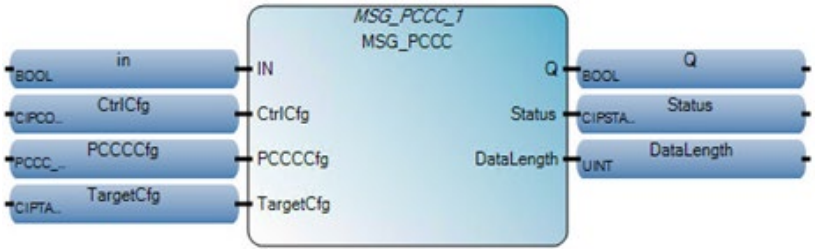


使用此表有助于确定该指令的参数值。

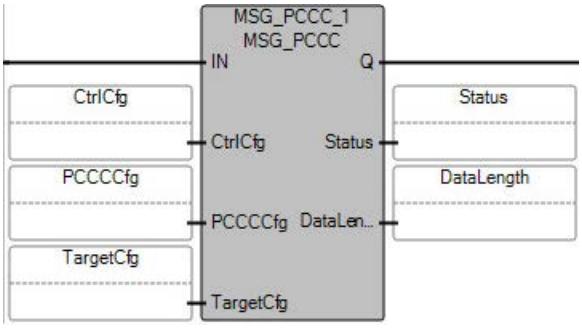
参数	参数类型	数据类型	描述
IN	输入	BOOL	梯级输入状态。 TRUE - 检测到上升沿，启动功能块，前提是上一个操作已完成。 FALSE - 未检测到上升沿，不启动。

参数	参数类型	数据类型	描述
CtrlCfg	输入	CIPCONTROLCFG	功能块执行控制配置。使用 CIPCONTROLCFG 数据类型 参考页数 181 定义 CtrlCfg。
PCCCCfg	输入	PCCC_CFG	PCCC 命令和地址。使用 PCCCCFG 数据类型 参考页数 181 定义 PCCCCfg。
TargetCfg	输入	CIPTARGETCFG	目标设备配置。使用 TARGETCFG 数据类型 参考页数 182 定义 TargetCfg。
Q	输出	BOOL	True - MSG 指令已完成。 False - MSG 指令未完成。
Status	输出	CIPSTATUS	功能块的执行状态。 触发或者重新触发 MSG 指令时，所有"状态"元素均会复位。 状态输出在 CIPSTATUS 数据类型 参考页数 184 中定义。
DataLength	输出	UINT	CIPPCCC 消息响应的数据字节数。 触发或重新触发 MSG 时，DataLength 复位为 0。 当 PCCC 消息通过以太网通道发送时，支持 DataLength 。

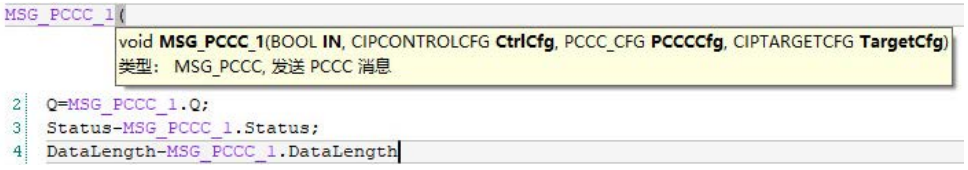
MSG_PCCC 功能块图示例



MSG_PCCC 梯形图示例



MSG_PCCC 结构化文本示例



CIPCONTROLCFG 数据类型

使用此表帮助确定 CIPCONTROLCFG 数据类型的参数值。

参数	数据类型	描述
Cancel	BOOL	TRUE - 取消功能块的执行。 启用消息后将清除位。
TriggerType	UDINT	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。

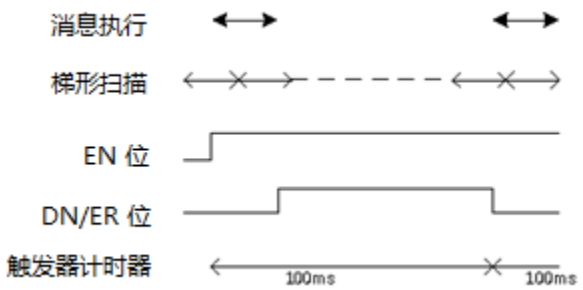
PCCC 消息触发

通过将 TriggerType 参数设置为非零值，可以定期触发消息。

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。



PCCCCFG 数据类型

使用此表定义 PCCCCFG 数据类型的参数。

参数	数据类型	描述
Service	USINT	服务代码： • 0 - 读取（默认） • 1 - 写入
SrcAdr	STRING	源地址： • 源文件名（如果 Service 的值为 0） 语法：*<文件类型><文件编号>:<元素编号>* 示例：N7:1 支持的数据类型包括： • N（整型） • F（浮点型）

参数	数据类型	描述
		• B（布尔型） • L（长整型） • ST（字符串型） • 标签变量名（如果 Service 的值为 1） • 一维数组示例：testarrayVariable1*[x]*，其中 x 是数组在数组区间内的位置。 支持一维数组。 • 正常变量或索引从 0 开始的一维数组示例：VariableName1。
NumOfElements	UINT	元素个数： • 当 Service 为 0 时，最大元素数（元素值 * 单个元素占用字节）不能超过 244。 • 当 Service 为 1 时，最大元素数（元素值 * 单个元素占用字节）不能超过 232。
DstAdr	STRING	目标地址： • 目标文件名（如果 Service 的值为 1） 语法：*<文件类型><文件编号>:<元素编号>* 示例：N7:1 支持的数据类型包括： • N（整型） • F（浮点型） • B（布尔型） • L（长整型） • ST（字符串型） • 标签变量名（如果 Service 的值为 0） • 一维数组示例：testarrayVariable2*[x]*，其中 x 是数组在数组区间内的位置。 支持一维数组。 • 正常变量或索引从 0 开始的一维数组示例：VariableName2。

TARGETCFG 数据类型

使用此表帮助确定 TARGETCFG 数据类型的参数值。

参数	数据类型	描述
Path	STRING[80]	使用语法 *{"<端口>,<节点/插槽地址>"}* 定义的目标路径 端口号定义： • 嵌入式串行端口为 2 • 嵌入式以太网端口为 4 • 安装在插槽 1-3 中的 2080-SERIALISOL 插件模块为 5-7
CipConnMode	USINT	CIP 连接类型。 • 1 - Class3 连接 对于以太网和串行 PCCC 消息，此参数必须设置为 1。

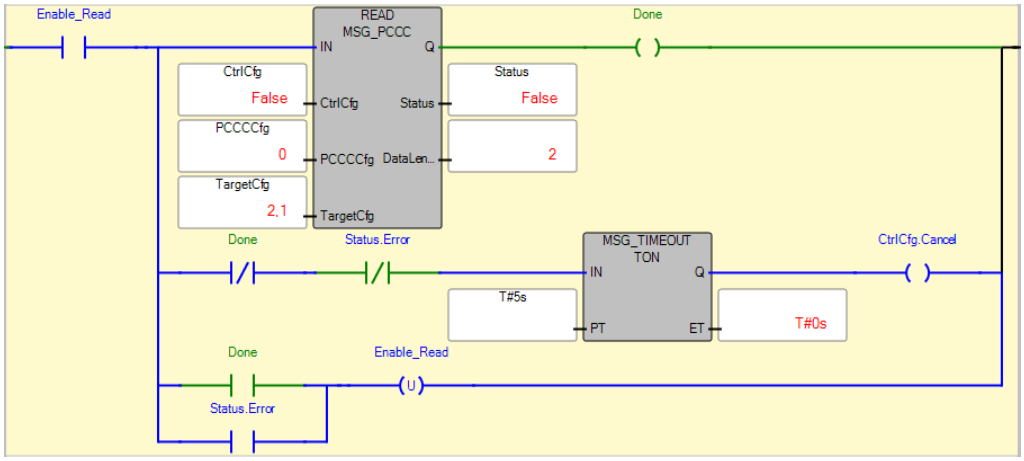
参数	数据类型	描述
UcmmTimeout	UDINT	断开消息超时（毫秒）。等待断开消息回复的时长，包括已连接消息的连接建立。 • 有效值：250-10,000 毫秒。 • 设为 0 以使用默认值 3000 毫秒（3 秒）。 • 设为小于 250 毫秒的值将被设为 250 毫秒（最小值）。 • 设为大于 10,000 毫秒的值将被设为 10,000 毫秒（最大值）。 该参数适用于 EtherNet/IP。
ConnMsgTimeout	UDINT	Class3 连接超时（毫秒）。等待已连接消息回复的时长。该连接会在超时时间到期时关闭。 • 有效值：800-10,000 毫秒。 • 设为 0 以使用默认值 10,000 毫秒（10 秒）。 • 设为小于 800 的值将被设为 800 毫秒（最小值）。 • 设为大于 10,000 毫秒的值将被设为 10,000 毫秒（最大值）。 该参数适用于 EtherNet/IP 和串行端口。
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）。 该参数适用于 EtherNet/IP。

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。

字符串元素	描述
"6,1"	通过安装在插槽 2 中的串行端口插件模块连接到地址为 1 的节点。请注意，插件插槽 1-3 等同于端口 5-7。
"4,192.168.1.100"	通过端口 4（Micro870 嵌入式以太网端口），到达位于 192.168.1.100 处的 节点。
本地端口	用于发送消息的本地端口。该端口应是活动的 EtherNet/IP 或 CIP 串行端口 - 不支持 USB 端口。
目标地址	• 对于 EIP，目标地址指定目标的 IP 地址。IP 地址必须为单播地址，不能是 0、多播、广播、本地地址或回送 (127.x.x.x) 地址。 • 对于 CIP 串行，目标地址指定目标的节点地址。支持的值范围为 0-254。

重要事项 从串行端口触发 MSG_PCCC 指令时，用户必须包含消息超时计时器，以在合理的时间内未收到无错误响应（尤其是使用无线调制解调器时）时引发消息指令错误。计时器预设时间应由用户根据其应用程序进行设置。下面显示了一个示例。



CIPSTATUS 数据类型

使用此表确定 CIPSTATUS 数据类型的参数值。

参数	数据类型	描述
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。有关更多信息，请参见错误代码表 参考页数 186
SubErrorID	UINT	子错误代码 参考页数 186 值。
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 • 位 5：CIPCONN - CIP 连接关闭 • 位 6：EIPSESS - EIP 会话关闭 位 5 和位 6 用于 Ethernet/IP，其他位保留。 为 CIPSTATUS 状态位 参考页数 184 定义 StatusBits。

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。

位名称	描述	行为
EN	启用	当梯级变为 true 时置位，并且一直置位，直至 DN 位或 ER 位已置位且梯级变为 false。
EW	启用等待	当为消息请求分配通信缓冲时设置。在设置 ST 位后清除。
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。
ER	错误	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。
DN	完成	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。

位名称	描述	行为
CIPCONN	完成	当通信的 CIP 连接关闭时置位。当 ConnClose 为 True 时，CIPCONN 位适用，其他情况下 CIPCONN 位为 False。 该位仅用于以太网。
EIPSESS	完成	当通信的封装 CIP 会话关闭时置位。当 ConnClose 为 True 时，EIPSESS 位适用，其他情况下 EIPSESS 位为 False。 该位仅用于以太网。



提示：当消息已启用（EN 位已置位）且未完成（DN 位未置位）时，如果已设置 Cancel 参数，则消息执行结束且 ER 位置位。

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。以下所有情况均适用于以太网。

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。 如果消息的本地端口是以太网，且不存在指向目标的 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。 如果将多条消息指定给同一目标，这些消息必须使用相同的连接。
当 ConnClose 为 true 时，消息执行完成时关闭消息连接和 EIP 会话。	如果多个消息共享同一连接，则最后一条消息完成时关闭连接。
消息执行已完成，并且 ConnClose 已设为 False。	完成消息执行后，连接不会关闭。
如果不与任何活动消息关联的 CIP 或 EIP 连接空闲 x 秒钟，则将其关闭，其中 x 是可以使用"CIP 设置服务"配置的闲置状态超时值。	有关 CIP 设置服务的详细信息，请参见 CIP 规范第 II 卷 TCP/IP 对象。
CIP 连接会关闭，任何 EIP 连接也会关闭。	如果多个 CIP 连接使用同一 EIP 连接，将在关闭所有关联的 CIP 连接后再关闭该 EIP 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。

消息超时

下表描述了 inUcmmTimeout 和 ConnMsgTimeout 参数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 CIP 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 适用于 Ethernet/IP

操作	结果
连接请求已完成	UcmmTimeout 计时器已重新激活 适用于 Ethernet/IP

MSG_PCCC 错误代码

使用此表确定在设置了 ER 位时 ErrorID 和 SubErrorID 字段的参数值 。

ErrorID 代码	SubErrorID 代码	错误代码说明
30		SubErrorID 表示状态 (STS) 远程错误。有关详细信息，请参阅 DF1 协议和命令集参考手册的第 8 章。
31		SubErrorID 表示扩展状态 (EXT STS) 远程错误。有关详细信息，请参阅 DF1 协议和命令集参考手册的第 8 章。
32	**PCCC 配置错误**	
	1	不支持的服务代码 受支持的服务代码如下： • 0 - 读取（默认） • 1 - 写入
	2	源地址无效 有关正确的源地址，请参见 PCCCCFG 数据类型 参考页数 181 表中的 SrcAdr。 对于字符串数据类型，支持的固定字符串大小为 82。
	3	目标地址无效 有关正确的源地址，请参见 PCCCCFG 数据类型 参考页数 181 表中的 DstAdr。 对于字符串数据类型，支持的固定字符串大小为 82。
	4	源数据类型无效 支持的数据类型包括： • N（整型） • F（浮点型） • B（布尔型） • L（长整型） • ST（字符串型）
	5	目标数据类型无效 支持的数据类型包括： • N（整型） • F（浮点型） • B（布尔型） • L（长整型） • ST（字符串型）
	6	文件类型和符号数据类型不兼容。浮点型数据类型仅向浮点数据类型发送消息，或从浮点数据类型发送消息。
	41	CIP 符号名称无效
	48	功能块的输入数据数组大小不足。 请验证数据缓冲区大小是否超出请求的数据长度。
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。 对于 PCCC 消息，确保将参数 CipConnMode 设置为 1。
	48	元素计数无效 元素计数的范围为 1 到最大值。 有关更多信息，请参见“元素计数”表。

ErrorID 代码	SubErrorID 代码	错误代码说明
	49	目标路径无效。
	51	有效负载无效，或者功能块的传输数据数组对于 CIP 通信而言过大。
	52	段类型值有误
	53	UCCM 超时值错误 如果封装超时值小于 UCCM 超时，或封装超时和 UCCM 超时之间的差异小于或等于一秒，将报告错误。 该代码适用于 Ethernet/IP。
	54	连接的超时值错误 如果封装超时值小于 CONNECTED 消息超时，或封装超时和 CONNECTED 消息超时之间的差异小于或等于一秒，将报告错误。 该代码适用于 EtherNet/IP。
	超时相关错误	
55	112	在消息等待队列中等待时，消息超时。 要避免此问题，请验证每个端口正在执行的功能块数是否等于或小于 4。
	113	在等待建立与链路层的连接时，消息超时。 要避免此问题，必须确保： • 电缆正常工作且目标存在。 • 控制器最多支持处理 8 条 CIP 客户端消息。
	114	在等待传输至链路层时，消息超时。 要避免此问题，请验证控制器是否支持处理最多 8 条 CIP 客户端消息。
	115	在等待来自链路层的响应时，消息超时。 要避免此问题，请验证目标是否存在且电缆是否正常工作。
	与服务器响应格式有关的错误代码	
69	65	消息回复与请求不匹配。
	68	消息回复数据类型无效或不受支持。
208		未为该网络配置 IP 地址。要解决此问题，请配置有效的 IP 地址。 该代码适用于 EtherNet/IP。
209		已达到最大连接数 - 无可用连接。 要避免此问题，请减少已使用的连接数。 该代码适用于 EtherNet/IP。
210		Internet 地址或节点地址无效 要避免此问题，请勿对目标 IP 地址使用 0、广播、多播或回送。 该代码适用于 EtherNet/IP。
217		用户取消了消息执行（Cancel 参数已设置为 TRUE）。 要避免此问题，请勿将 Cancel 位置位或执行 RMD 操作。
218		无可用网络缓冲空间 要避免此问题，请验证当前正在进行的通信活动没有过量。 该代码适用于 EtherNet/IP。
222		已保留。
223		链接地址不可用。正在进行 TCP/IP 或以太网配置更改。 该代码适用于 EtherNet/IP。

ErrorID 代码	SubErrorID 代码	错误代码说明
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，请参见 CIP 规范。 该代码适用于 EtherNet/IP。
255		通道已关闭或正在进行重新配置。 在重新配置过程中，或在执行 MSG_CIPPCCC 指令时端口不可用，可能会遇到此错误。 要避免此问题，请等待配置完成后再执行指令。 该代码适用于 EtherNet/IP。

元素计数表

源/目标数据类型	服务代码的最大元素计数 0 - 读取服务	服务代码的最大元素计数 1 - 写入服务
BIT	122	116
INT	122	116
FLOAT	61	58
STRING	2	2
DOUBLE	61	58

有关创建 MSG_PCCC 程序的信息，请参见示例：如何在 MicroLogix 中创建 MSG_PCCC 消息程序以将值写入地址和从地址读取值 参考页数 188。

示例：在 MicroLogix 中创建 MSG_PCCC 消息程序以将值写入地址和从地址读取值

本示例显示了如何创建消息程序，以从控制器 B 全局变量将值写入控制器 A 旧地址，以及从控制器 A 旧地址读取值，然后通过以太网保存到控制器 B 全局变量。



PCCC 写入示例：添加 MSG_PCCC 功能块、变量和触点

要创建用于从 Micro870E 控制器将值写入 MicroLogix 1400 中的旧地址的 MSG_PCCC 消息程序，请执行以下任务。

为源变量创建全局变量

1. 在“项目管理器”中，双击“全局变量”以显示“全局变量选择器”。
2. 在“名称”列中输入 **Write_Float_Data**。
3. 验证数据类型是否为 REAL。

“全局变量选择器”应与下图相似。

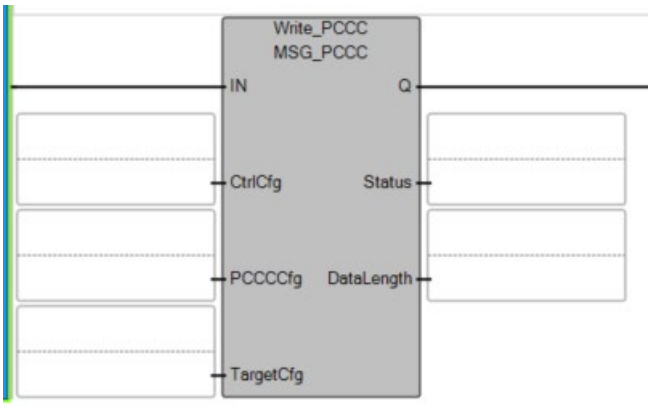


添加功能块和变量

1. 在“指令工具栏 - 消息传送”中，选择 **MSG_PCCC** 并将其拖到梯形图梯级中。



2. （可选）双击 **MSG_PCCC** 以将功能块的实例更改为 Write_PCCC。



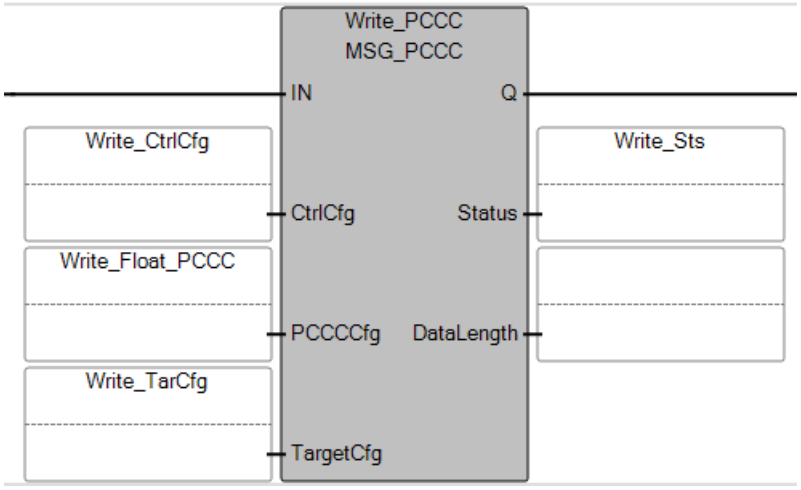
3. 分配变量名。

a. 在“项目管理器”中，双击“局部变量”以显示“局部变量”页面。

b. 在“变量选择器”中，添加表中所列的变量和数据类型。

c. 将变量分配给指令输入和输出参数。

参数	变量名	数据类型
CtrlCfg	Write_CtrlCfg	CIPCONTROLCFG
PCCCCfg	Write_PCCC	PCCC_CFG
TargetCfg	Write_TarCfg	CIPTARGETCFG
Status	Write_Sts	CIPSTATUS



4. 配置局部变量的初始值，并添加 CtrlCfg 初始值。

a. 在“局部变量”页面中，展开 CtrlCfg 参数以查看附加参数。

b. 在每个参数的“初始值”列中输入以下值。

参数	初始值
Write_CtrlCfg.Cancel	留空
Write_CtrlCfg.TriggerType	50
Write_CtrlCfg.StrMode	留空

提示：在本示例中输入了 **50** 以定义 50 毫秒的循环触发间隔。

5. 添加 PCCCCfg 的初始值。

a. 在“局部变量”页面中，展开 PCCCCfg 参数以查看附加参数。

b. 在每个参数的“初始值”列中输入以下值。

参数	初始值
Write_Float_PCCC.Service	1
Write_Float_PCCC.SrcAdr	'Write_Float_Data'
Write_Float_PCCC.NumOfElements	1
Write_Float_PCCC.DstAdr	'F8:0'

提示：在 MicroLogix 控制器中将 **F8:0** 输入为目标地址。

局部变量选择器应与下图相似。

名称	初始值	数据类型	注释
Write_CtrlCfg	...	CIPCONTR...	
Write_CtrlCfg.Cancel		BOOL	中止消息执行
Write_CtrlCfg.TriggerType	50	UDINT	0 - 触发一次，n - 循环...
Write_CtrlCfg.StrMode		USINT	保留参数
Write_Float_PCCC	...	PCCC_CFG	
Write_Float_PCCC.Service	1	USINT	0 - 读，1 - 写
Write_Float_PCCC.SrcAdr	'"Write_Float_D...	STRING	源地址: 如果是"读": <文...
Write_Float_PCCC.NumOfElements	1	USINT	元素数目
Write_Float_PCCC.DstAdr	'F8:0'	STRING	目标地址: 如果是"读": ...

6. 配置 TarCfg 的初始值。

a. 在“局部变量”页面中，展开 TarCfg 参数以查看附加参数。

b. 在每个参数的“初始值”列中输入以下值。

参数	初始值
Write_TarCfg.Path	'4,192.168.3.33'
Write_TarCfg.CipConnMode	1
Write_TarCfg.UcmmTimeout	0
Write_TarCfg.ConnMsgTimeout	0
Write_TarCfg.ConnClose	FALSE

局部变量选择器应与下图相似。

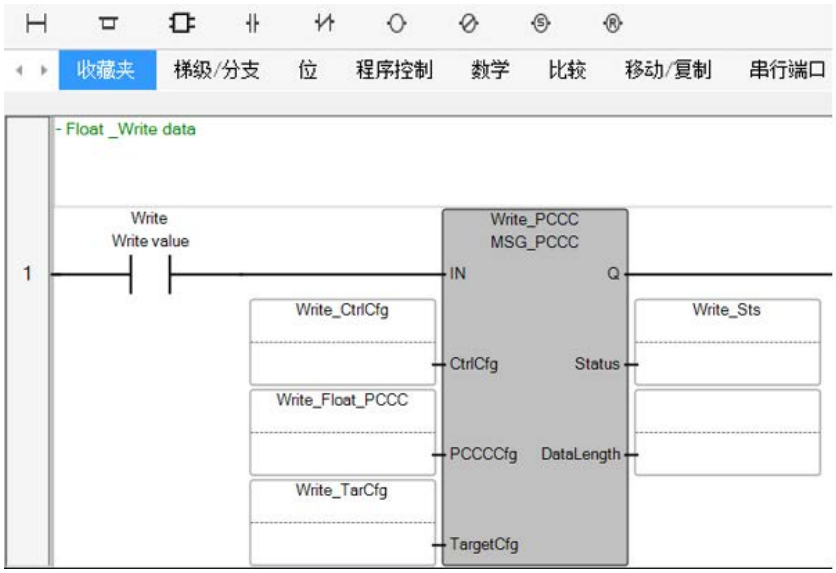
名称	初始值	数据类型	注释
Write_TarCfg	...	CIPTARGE...	
Write_TarCfg.Path	'4,192.168.3.33'	STRING	CIP 目标路径
Write_TarCfg.CipConnMode	1	USINT	0 - 未连接，1 - ...
Write_TarCfg.UcmmTimeout	0	UDINT	未连接消息超时。
Write_TarCfg.ConnMsgTimeout	0	UDINT	连接消息超时。
Write_TarCfg.ConnClose	FALSE	BOOL	TRUE: 消息完...

7. 添加触点。

a. 在“指令工具栏 - 收藏夹”中，选择“正向触点”，并将其拖到第一个梯形图梯级中 MSG_PCCC 功能块输入的左侧。

b. 单击触点，并将“写入”变量分配给触点。

MSG_PCCC 消息的梯形图编程的梯级应与下图相似。



8. 下载程序。

下载完成后，将要写入 MicroLogix 的值输入到 Micro800 控制器中的 **Write_Float_Data** 变量，然后触发“写入”标签以执行从 Write_Float_Data 到 MicroLogix 控制器 F8:0 地址的值写入。观察 MicroLogix 地址将更新为该值。

PCCC 读取示例：添加 PCCC_MSG 功能块、变量和触点

要创建用于将 MicroLogix 1400 中旧地址的值读取到 Micro870E 控制器的 MSG_PCCC 消息程序，请执行以下任务。

创建用于存储结果的全局变量

1. 在“项目管理器”中，双击“全局变量”以显示“全局变量选择器”。
2. 在“名称”列中输入 **Read_Float_Data**。
3. 验证数据类型是否为 REAL。

“全局变量选择器”应与下图相似。

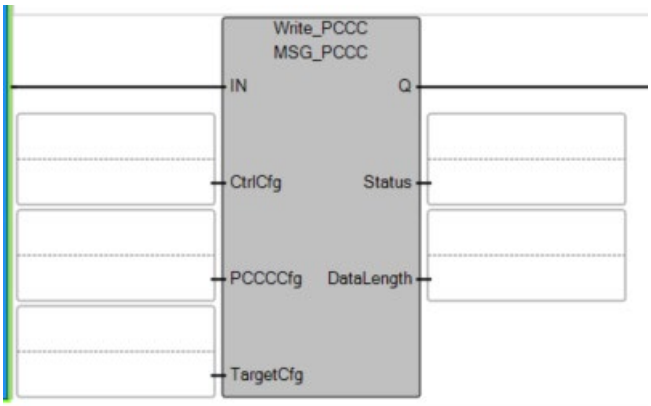


添加功能块和变量

1. 在“指令工具栏 - 消息传送”中，选择 **MSG_PCCC** 并将其拖到梯形图梯级中。



2. （可选）双击 **MSG_PCCC** 以将功能块的实例更改为 Read_PCCC。



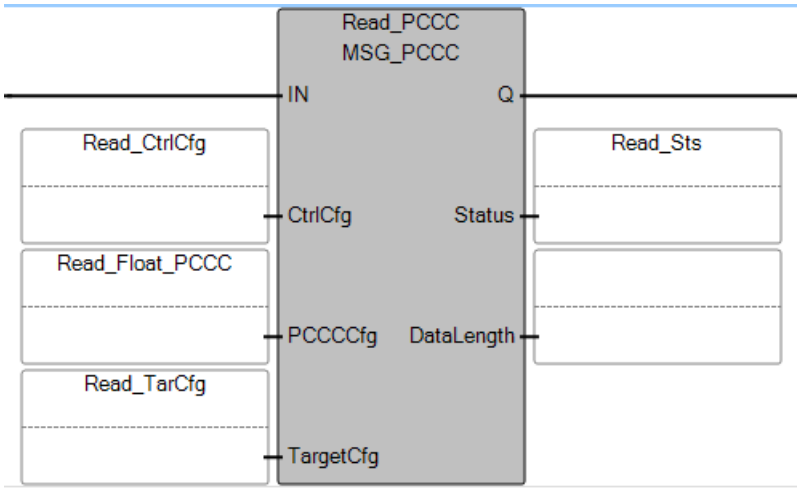
3. 分配变量名。

a. 在“项目管理器”中，双击“局部变量”以显示“局部变量”页面。

b. 在“变量选择器”中，添加表中所列的变量和数据类型。

c. 将变量分配给指令输入和输出参数。

参数	变量名	数据类型
CtrlCfg	Read_CtrlCfg	CIPCONTROLCFG
PCCCCfg	Read_PCCC	PCCC_CFG
TargetCfg	Read_TarCfg	CIPTARGETCFG
Status	Read_Sts	CIPSTATUS



4. 配置局部变量的初始值，并添加 CtrlCfg 初始值。

a. 在“局部变量”页面中，展开 CtrlCfg 参数以查看附加参数。

b. 在每个参数的“初始值”列中输入以下值。

参数	初始值
Read_CtrlCfg.cancel	留空
Read_CtrlCfg.TriggerType	50
Read_CtrlCfg.StrMode	留空

提示：在本示例中输入了 **50** 以定义 50 毫秒的循环触发间隔。

5. 添加 PCCCCfg 的初始值。

a. 在“局部变量”页面中，展开 PCCCCfg 参数以查看附加参数。

b. 在每个参数的“初始值”列中输入以下值。

参数	初始值
Read_Float_PCCC.Service	1
Read_Float_PCCC.SrcAdr	'F8:0'
Read_Float_PCCC.NumOfElements	1
Read_Float_PCCC.DstAdr	'Read_Float_Data'

提示：在 MicroLogix 控制器中将 **F8:0** 输入为源地址。

局部变量选择器应与下图相似。

名称	初始值	数据类型	注释
Read_Ctrlcfg	...	CIPCONTR...	
Read_Ctrlcfg.Cancel		BOOL	中止消息执行
Read_Ctrlcfg.TriggerType	50	UDINT	0 - 触发一次，...
Read_Ctrlcfg.StrMode		USINT	保留参数
Read_Float_PCCC	...	PCCC_CFG	
Read_Float_PCCC.Service	1	USINT	0 - 读，1 - 写
Read_Float_PCCC.SrcAdr	'F8:0'	STRING	源地址: 如果是"...
Read_Float_PCCC.NumOfElements	1	USINT	元素数目
Read_Float_PCCC.DstAdr	'Read_Float_D...	STRING	目标地址: 如果...

6. 配置 TarCfg 的初始值。

a. 在“局部变量”页面中，展开 TarCfg 参数以查看附加参数。

b. 在每个参数的“初始值”列中输入以下值。

参数	初始值
Read_TarCfg.Path	'4,192.168.3.33'
Read_TarCfg.CipConnMode	1
Read_TarCfg.UcmmTimeout	0
Read_TarCfg.ConnMsgTimeout	0
Read_TarCfg.ConnClose	FALSE

局部变量选择器应与下图相似。

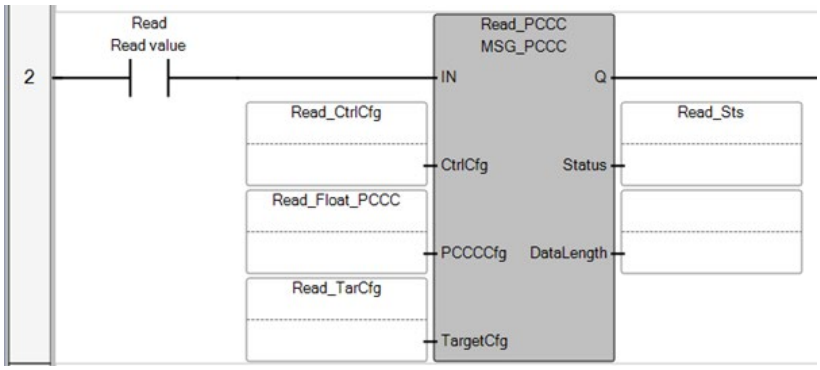
名称	初始值	数据类型	注释
Read_TarCfg	...	CIPTARGE...	
Read_TarCfg.Path	'4,192.168.3.33'	STRING	CIP 目标路径
Read_TarCfg.CipConnMode	1	USINT	0 - 未连接，1 - ...
Read_TarCfg.UcmmTimeout	0	UDINT	未连接消息超时。
Read_TarCfg.ConnMsgTimeout	0	UDINT	连接消息超时。
Read_TarCfg.ConnClose	FALSE	BOOL	TRUE: 消息完...

7. 添加触点。

a. 在“指令工具栏 - 收藏夹”中，选择“正向触点”，并将其拖放到第二个梯形图梯级中 MSG_PCCC 功能块输入的左侧。

b. 单击触点，并将“读取”变量分配给触点。

MSG_PCCC 消息的梯形图编程的梯级应与下图相似。



8. 下载程序。

下载完成后，在 MicroLogix 控制器 F8:0 地址中输入值并在 Micro800 控制器中触发读取标签以执行从 MicroLogix F8:0 地址读取值，然后观察值写入 Micro800 控制器中的 Read_Float_Data 。

消息执行过程和时间图

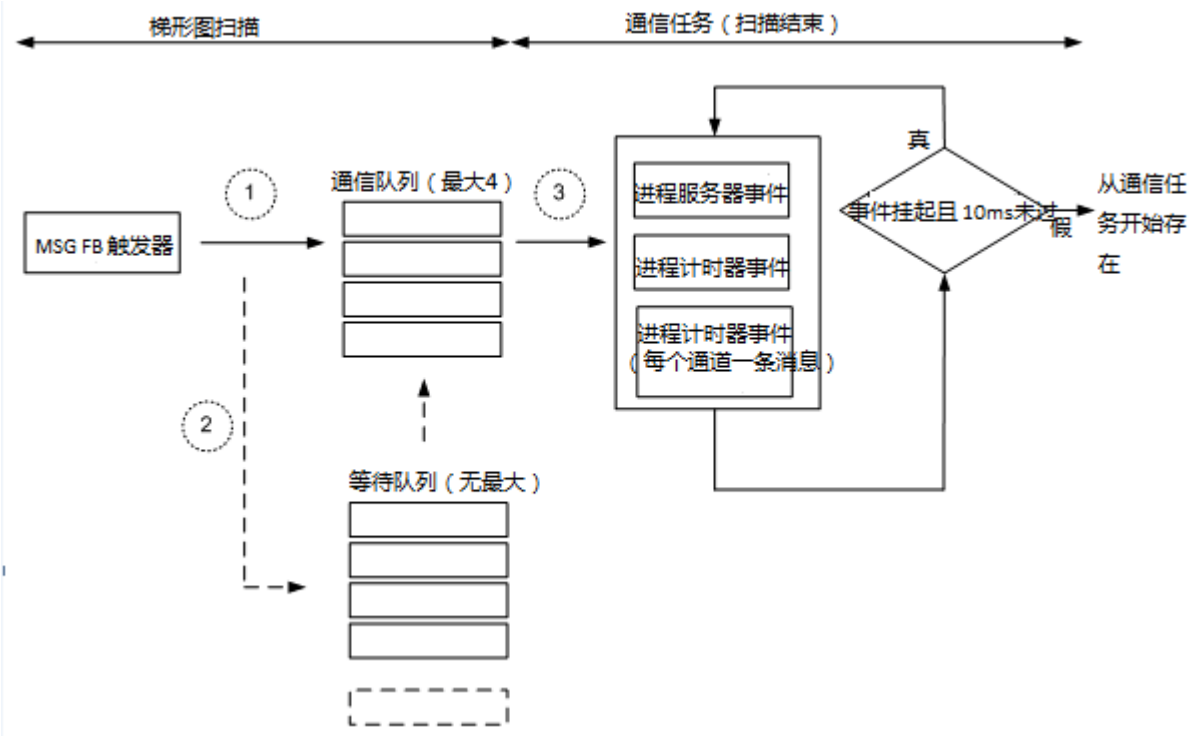
以下主题描述了 MSG_CIPGENERIC 参考页数 158、MSG_CIPSYMBOLIC 参考页数 165、MSG_MODBUS2 参考页数 174 和 MSG_PCCC 参考页数 179 消息指令何时及如何根据它们的位和梯级条件执行。

- 消息执行过程（常规）
- 消息执行过程 (Rung = TRUE)
- 消息执行时间图 (Rung = True)

- 消息执行过程 (Rung = FALSE)
- 消息执行时间图 (Rung = FALSE)
- 消息执行过程（错误）
- 消息执行时间图（PCCC 配置错误）
- 消息执行时间图（消息已取消）

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。

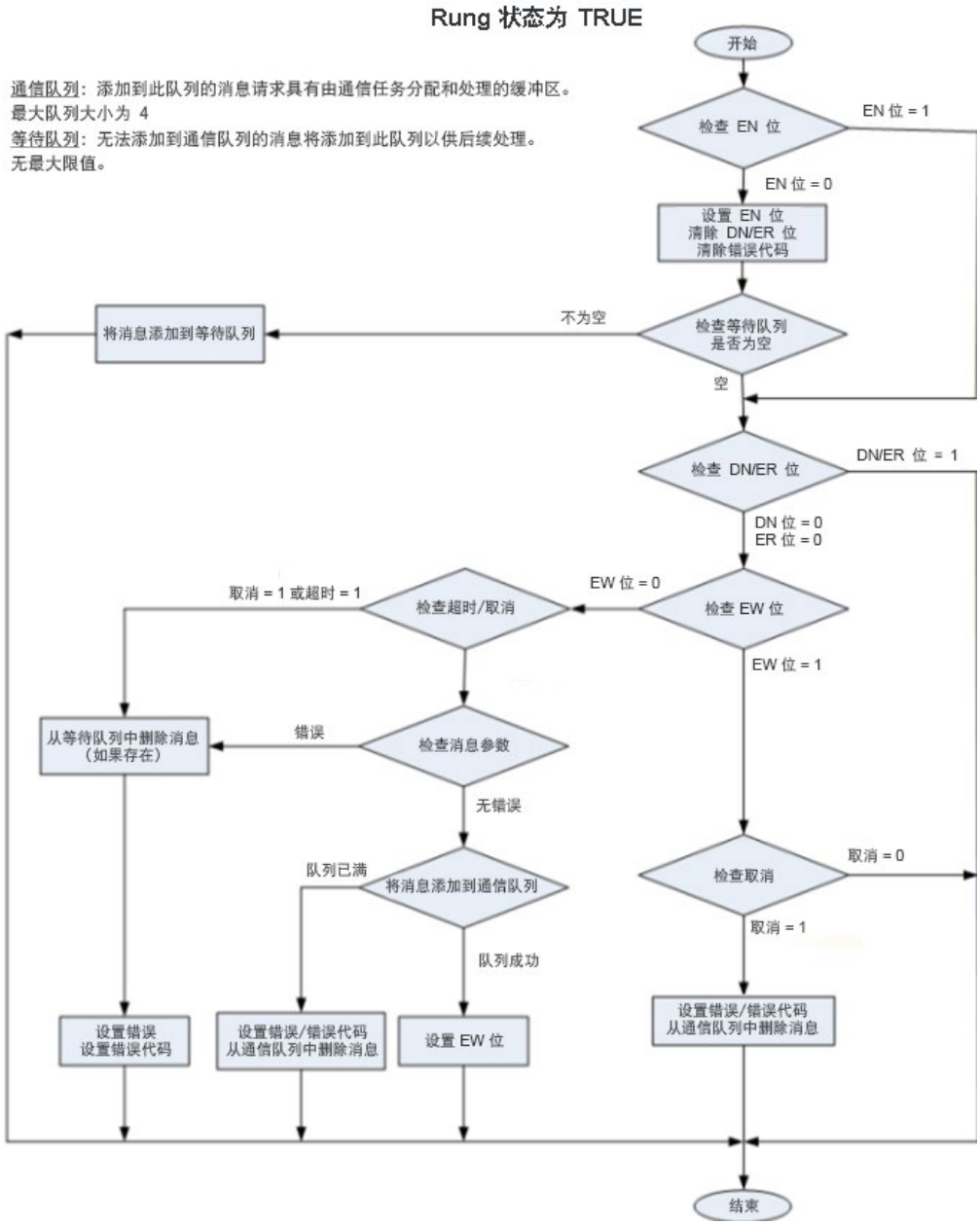


下表描述了上一图中标识的事件序列。

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。
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。

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。



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。

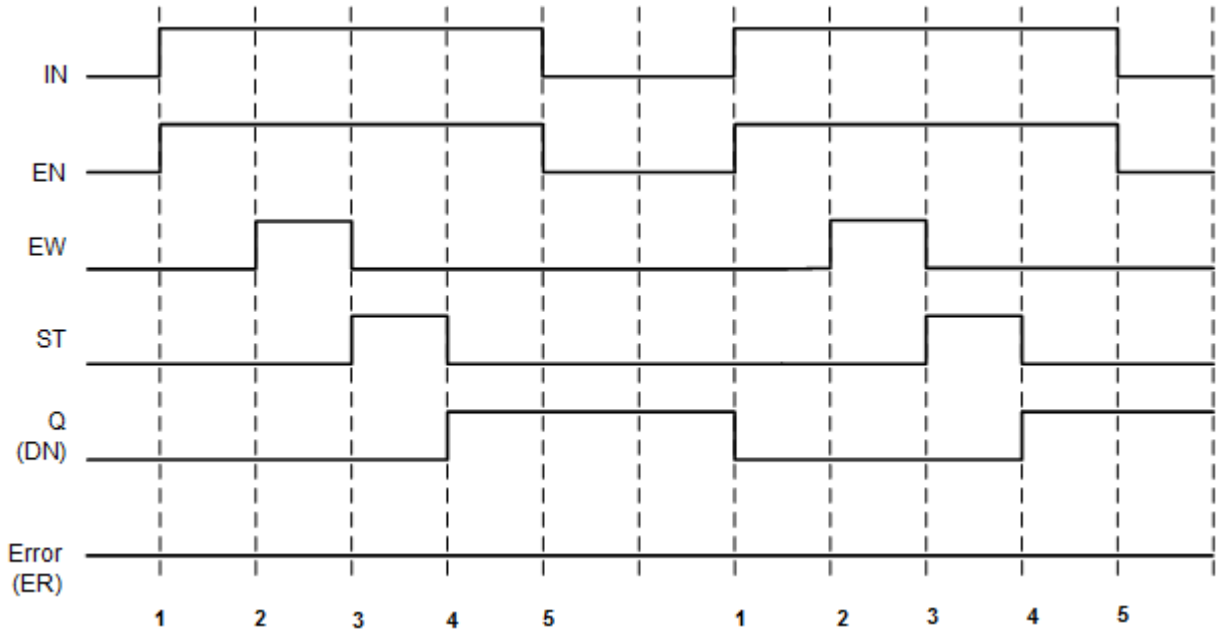
消息执行时间图（Rung = TRUE）

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。

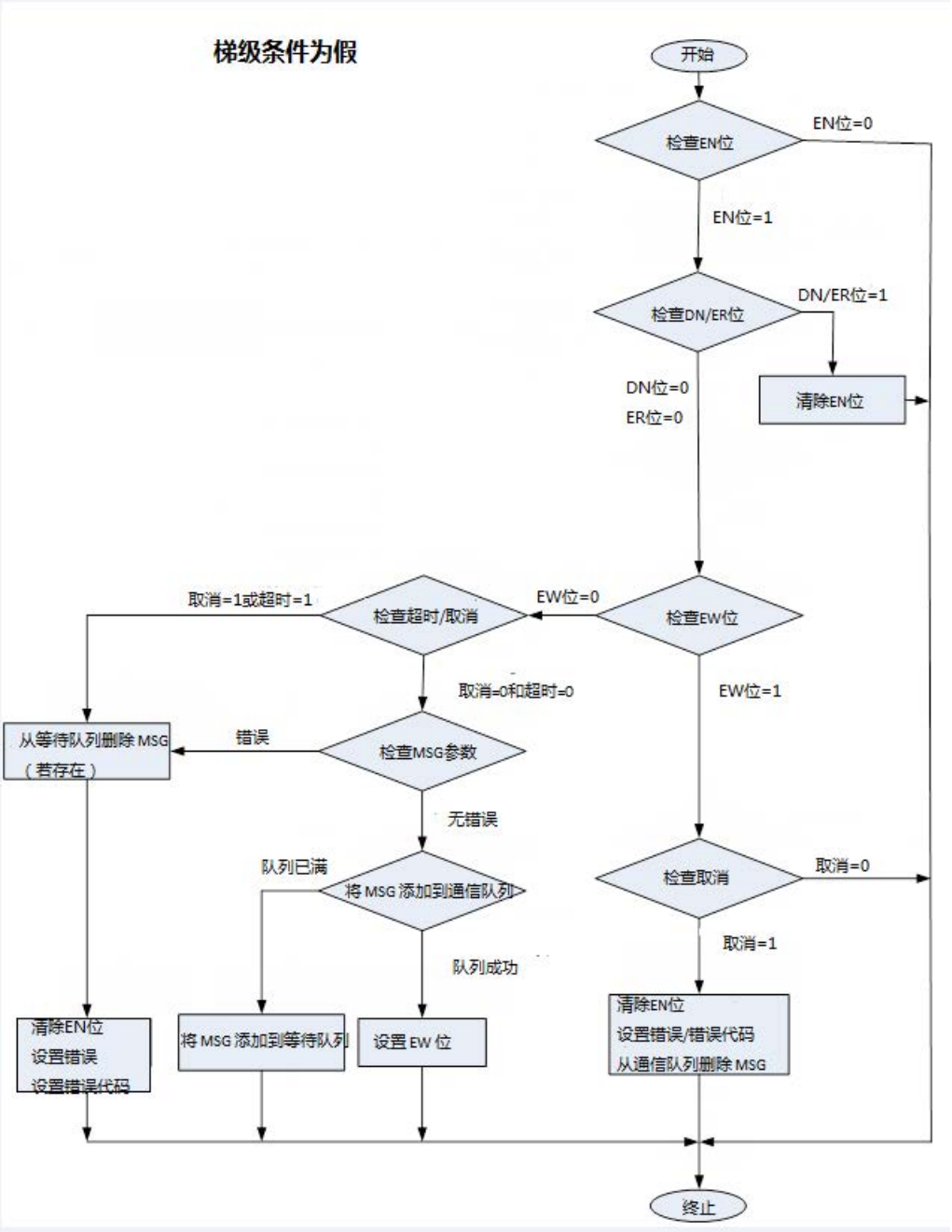
步骤	消息说明	位状态
4	消息响应已收到。	ST 位已清除。 DN 位已设置。
5	Rung 条件为 FALSE。	EN 位已清除。

Rung = TRUE 的时间图



消息执行过程 (Rung = FALSE)

以下过程图描述了当 Rung 条件为 False 时发生的消息指令事件。



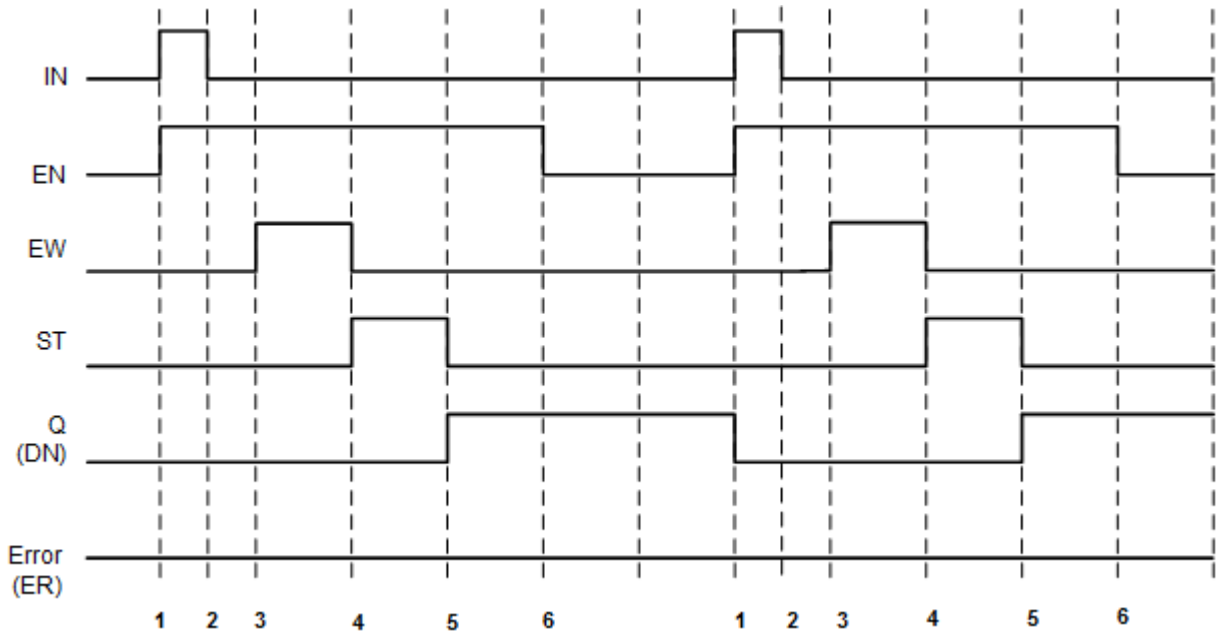
消息执行时间图（Rung = FALSE）

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。

步骤	消息说明	位状态
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



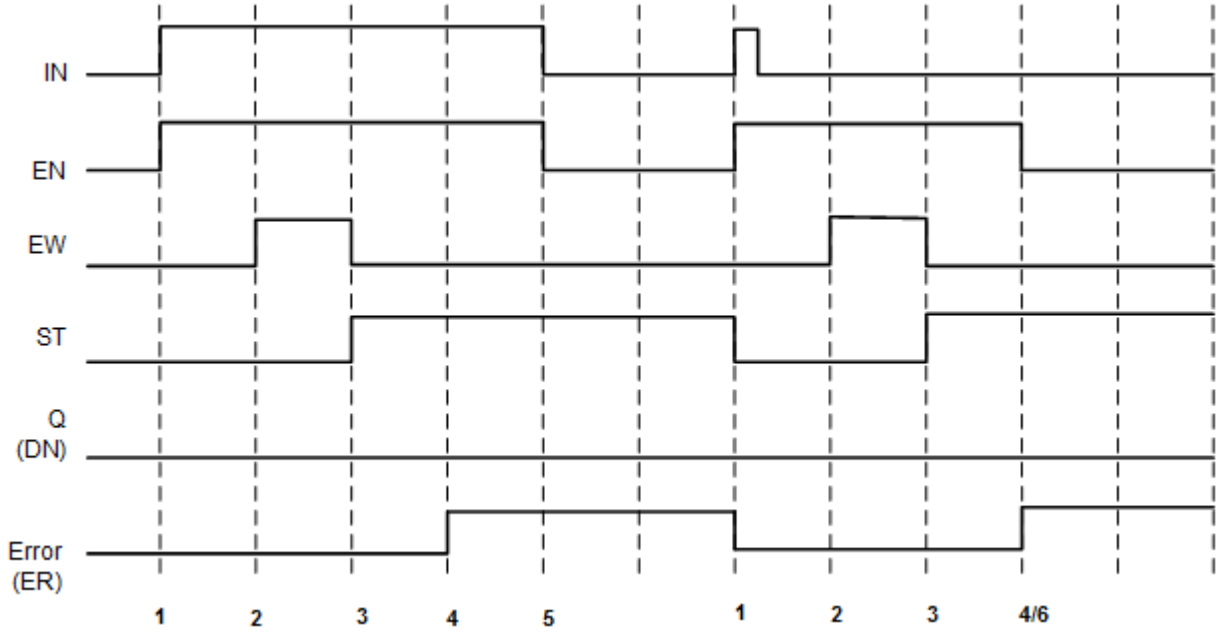
消息执行过程（错误）

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。
4-6	Rung 条件为 FALSE。	EW 位已清除。 ER 位已设置。

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。

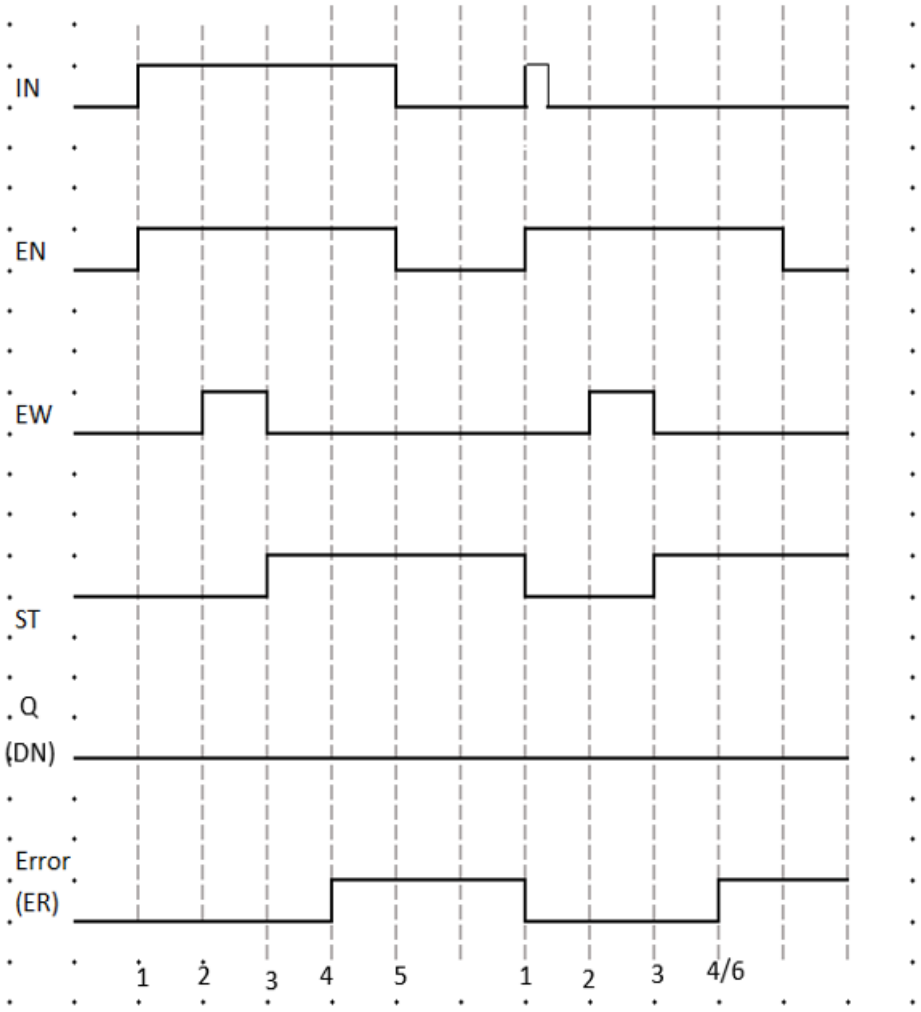


消息执行时间图（PCCC 配置错误）

下表描述了执行期间发生 PCCC 配置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。
2	已获取消息缓冲区。	EW 位已设置。
3	PCCCCfg 中存在无效元素。	ER 位先置位为 1，ST 位再置位为 1。 EW 和 ST 位未改变。
4	Rung 条件为 FALSE。	EN、EW 和 ER 位已清除。

下图显示了将 PCCCCfg 中的元素数更改为 17（无效值）时的典型错误状况。

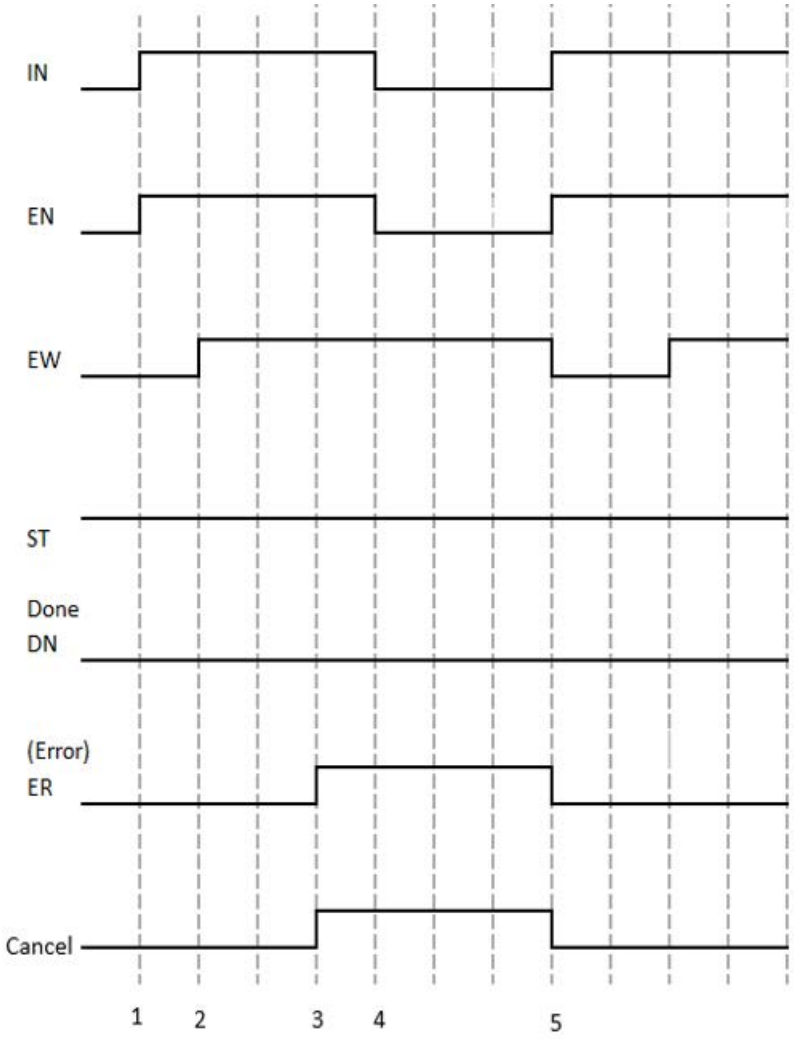


消息执行时间图（消息已取消）

下表描述了将消息取消后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。 所有其他位均已清除。
2	已获取消息缓冲区。	EW 位已设置。
3	消息执行已取消。	ER 位已设置。 EW 和 ST 位未改变。
4	Rung 条件为 FALSE。	EN 位已清除。
5	Rung 条件为 TRUE。	Cancel 输入已清除。 功能块开始执行。

将消息取消后的时间图



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 参考页数 155 的特定详细信息和示例。请参见以下主题，以了解使用 MSG_CIPGENERIC 参考页数 204 和 MSG_CIPSYMBOLIC 功能块创建程序的详细信息和示例 参考页数 210。

配置显式消息的对象数据值（MSG_CIPGENERIC）

要使用 MSG_CIPGENERIC 功能块 参考页数 158 处理显式消息，请使用正确的值配置 AppCfg 参数。

关于消息通信的其他信息

存在多种信息来源，信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。

下表列出了与消息通信有关的其他信息来源。

信息来源	描述	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	描述	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、消息路由器对象和网络对象。	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

要访问用户手册和快速入门指南：

1. 要访问快速入门指南，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**。
4. 要访问驱动器手册，在“帮助”菜单中单击“用户手册”以显示“手册”对话框。
5. 单击“驱动器”旁的加 (+) 号，以展开目录，然后展开类，直至找到手册。
6. 双击手册名以打开 .pdf 文件。
7. 要访问以太网/IP 手册，在“帮助”菜单中单击“用户手册”以显示“手册”对话框。
8. 单击**驱动器**旁的加 (+) 号，以展开类别，然后展开 PowerFlex 4 类外设类。
9. 双击“22-COMM-E 以太网/IP 适配器用户手册”以打开 .pdf 文件。

要访问 Rockwell Automation 文献库中的手册：

1. 转至 http://literature.rockwellautomation.com。
2. 要访问用户手册的非英语语言版本，请从“出版物语言”下拉框（右上角）中选择语言。
3. 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。
4. 在“搜索”框中，键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

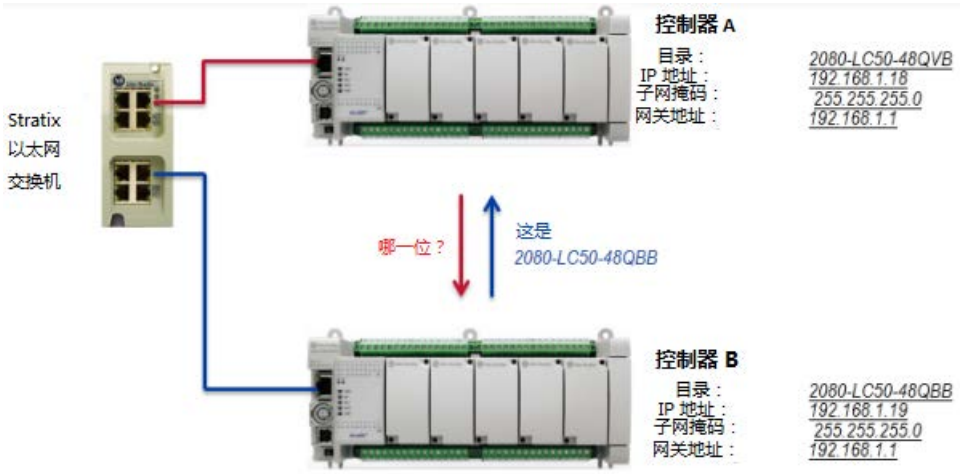
MSG_CIPGENERIC AppCfg 参数值

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 MSG_CIPGENERI 功能块参数中使用的 CIP 注册对象数据值。

名称	别名	数据类型	维度	项目值	初始值
MSG_ReadDrive		MSG_CIPGEN		...	...
MyAppCfg		CIPAPPCFG		...	...
MyAppCfg.Service		USINT			14
MyAppCfg.Class		UINT			7
MyAppCfg.Instance		UDINT			4
MyAppCfg.Attribute		UINT			4
MyAppCfg.MemberCnt		USINT			

示例：如何创建 MSG_CIPGENERIC 消息程序以从控制器中读取数据

本示例展示如何使用 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 目录信息。



执行以下任务以创建从不同控制器读取信息的 MSG_CIPGENERIC 消息程序。

否	任务
1	确定输入变量 (MSG_CIPGENERIC) 的初始值 参考页数 204
2	添加 MSG_CIPGENERIC 功能块和变量 参考页数 205
3	配置变量的初始值 参考页数 206
4	添加触点和线圈 参考页数 208
5	添加 COP 功能块、变量和触点 (MSG_CIPGENERIC) 参考页数 209
6	确认控制器 B 上正确的 IP 配置 参考页数 210

确定输入变量 MSG_CIPGENERIC 的初始值

按照以下一般步骤添加输入变量和初始值，并获取“标识对象”值以配置 AppCfg 参数初始值。

要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. 双击手册以将其打开。
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. 根据检索的类型信息，确定 AppCfg 参数的初始值。

以太网/IP 对象数据和 AppCfg 参数示例

下表标识了用于从控制器读取类别信息的特定以太网/IP 对象数据。

输入变量示例	AppCfg 参数	以太网/IP 对象数据选项	描述	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	类	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

添加 MSG_CIPGENERIC 功能块和变量

要将 MSG_CIPGENERIC 功能块添加到梯形图编程，然后将输入变量添加到功能块，请执行以下步骤。

要添加 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“项目管理器”。
2. 添加梯形图程序：
   - 在“项目管理器”中，右键单击“程序”，单击“添加”，然后单击“新 LD：梯形图”。
   - 右键单击“项目管理器”中的梯形图图标，单击“重命名”，然后键入 CIPExplicitMessage。
   - 双击“项目管理器”中的梯形图编程，以便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”，并将其拖放到梯形图梯级以显示“指令块选择器”。
   - 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。
4. 添加 MSG_CIPGENERIC 局部输入变量：

- 在“项目管理器”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）

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

“变量”页面应与下图相似。

	名称	别名	数据类型	维度	项目值	初始值
	MSG_ReadDrive		MSG_CIPGENER		...	...
	MyCtrlCfg		CIPCONTROLCF		...	...
	MyAppCfg		CIPAPPCFG		...	...
	MyTargetCfg		CIPTARGETCFG		...	...
	MyReqData		UDINT			
	MyReqLength		UDINT			
▶	MyResData		UDINT	[1..81]	...	...
✳						

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。

要将变量分配给 MSG_CIPGENERIC：

1. 要配置 MyCtrlCfg 输入变量的初始值：
   - 在**局部变量**页面中，展开 MyCtrlCfg 以查看其参数。
   - 在每个参数的“初始值”列中输入以下值。

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。

2. 配置 MyAppCfg 输入变量的初始值
   - 在“局部变量”页面中，展开 MyAppCfg 以查看其参数。
   - 在每个参数的“初始值”列中输入以下值。

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

3. 配置 MyTargetCfg 输入变量的初始值

- 在“局部变量”页面中，展开 MyTargetCfg 以查看其参数。
- 在每个参数的“初始值”列中输入以下值。

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。

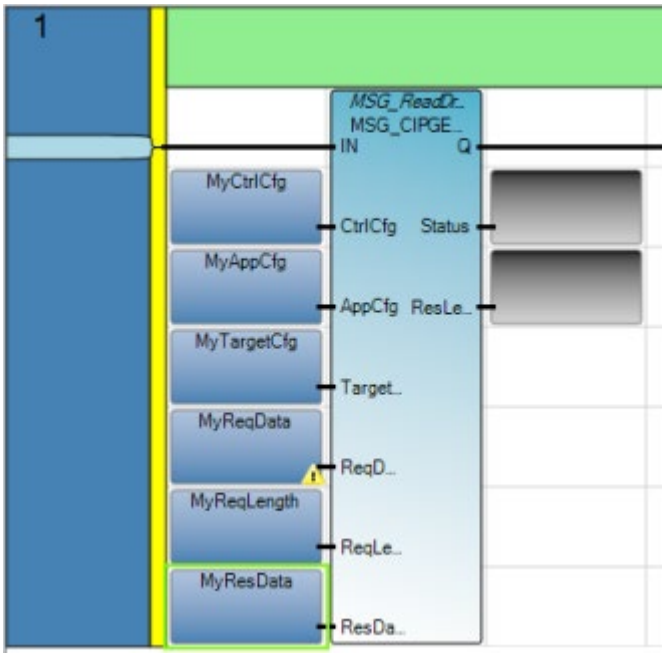
变量页面中的参数应与下图相似。

名称	别名	数据类型	维度	项目值	初始值
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.ConnMsgTimeout		UDINT			
MyTargetCfg.ConnClose		BOOL			FALSE
MyResData		UDINT	[1..81]	...	...
MyResData[1]		UDINT			

4. 将变量分配给参数
   - 在梯形图 POU 中，单击变量输入块顶部，以显示变量下拉列表。
   - 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。
Target	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。

MSG_CIPGENERIC 功能块的实例应与下图相似。



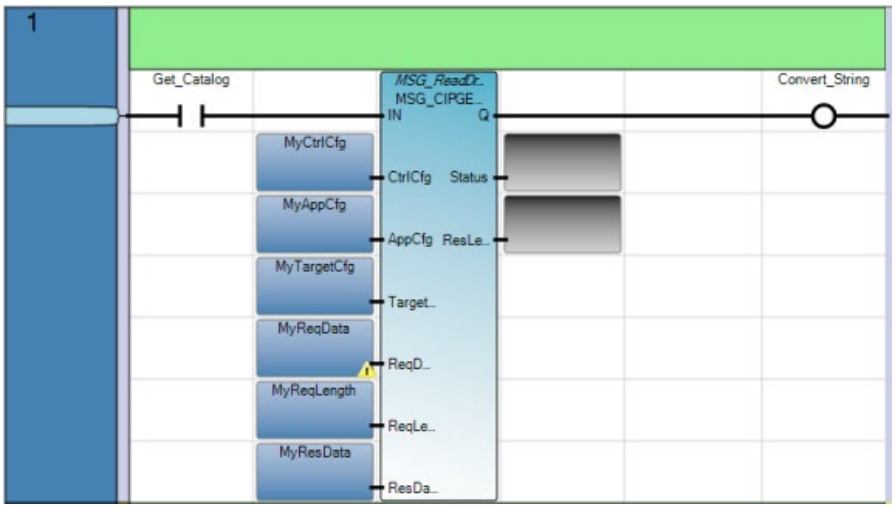
添加触点和线圈

使用以下步骤将线圈和触点添加到 MSG_CIPGENERIC 指令，以将目录信息转换为人类可读的字符串。

要向 MSG_CIPGENERIC 添加线圈：

1. 在“工具箱”中，选择“正向触点”，并将其拖放到第一个梯形图梯级中 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图梯级中 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”。

MSG_CIPGENERIC 消息的梯形图程序的第一个梯级应与下图相似。



添加 COP 功能块、变量和触点（MSG_CIPGENERIC）

使用以下步骤添加 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，将目录信息转换为人类可读的字符串。

要添加 COP 功能块：

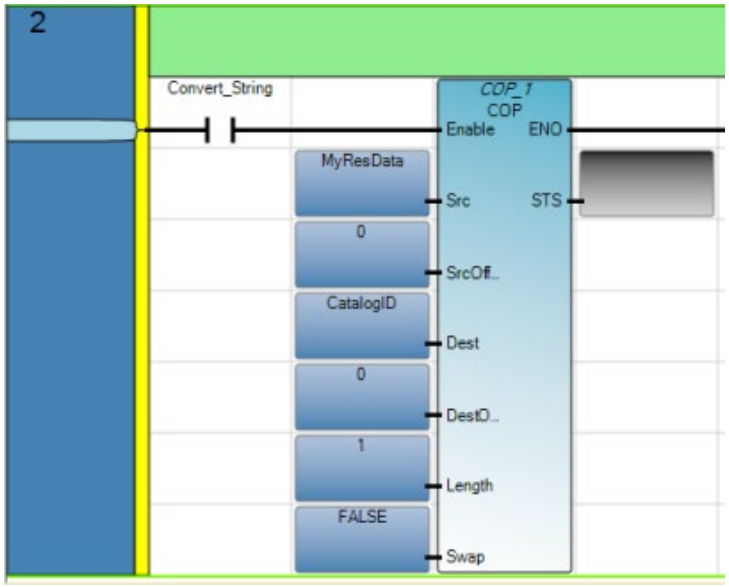
1. 在“工具箱”中，选择“梯级”，并将其拖放到第一个梯形图梯级的正下方以添加第二个梯级。
2. 添加 COP 功能块：
   - 在“工具箱”中，选择“块”，并将其拖放到第二个梯形图梯级以打开“指令块选择器”。
   - 双击 **COP**，以将功能块实例添加到梯形图。
3. 为 COP 添加局部输入变量：
   - 在“项目管理器”中，双击“局部变量”以显示“局部变量”页面。
   - 在“局部变量”页面中，添加下表中所列的变量和数据类型。

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

4. 对于 CatalogID 变量，双击“维度”，并将数组大小更改为 [1..1]。
5. 添加触点：
   - 在“工具箱”中，选择“正向触点”，并将其拖放到第二个梯形图梯级中 COP 功能块输入的左侧。
   - 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。



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。

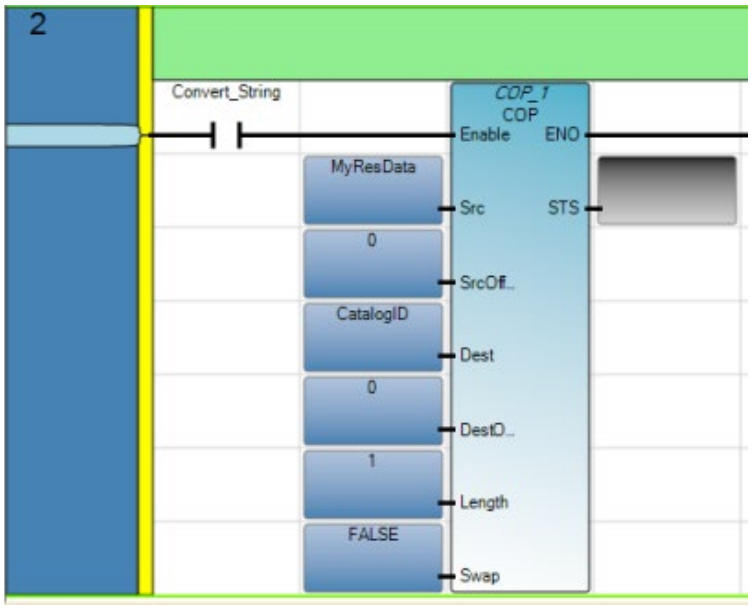
要验证 IP 地址：

1. 打开控制器的应用程序工作区：
2. 在“**项目组织管理器**”中，双击控制器以打开控制器工作区。
3. 在控制器工作区中，展开“控制器”树中的“以太网”，然后单击“Internet 协议”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

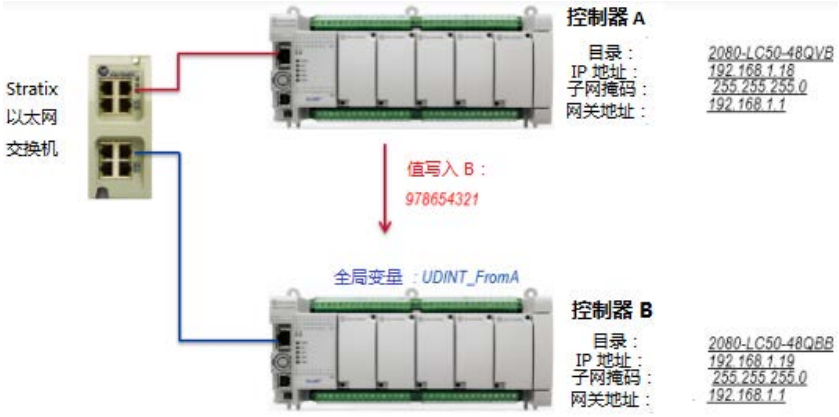
结果

控制器配置页面中的“Internet 协议”选项应与下图相似。



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。



要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，请执行以下任务。

否	任务
1	添加 COP 功能块、变量和触点 (MSG_CIPSYMBOLIC) 参考页数 211
2	添加 Equal 运算符和线圈 参考页数 212
3	添加 MSG_CIPSYMBOLIC 功能块、变量和触点 参考页数 213
4	确认控制器 B 上正确的 IP 配置 参考页数 210
5	为控制器 B 创建全局变量 参考页数 216
6	查看整个程序结果 参考页数 216

添加 COP 功能块、变量和触点 (MSG_CIPSYMBOLIC)

COP 指令用于将您输入的数据转换为目标数据类型，从而使数据与控制器变量兼容。

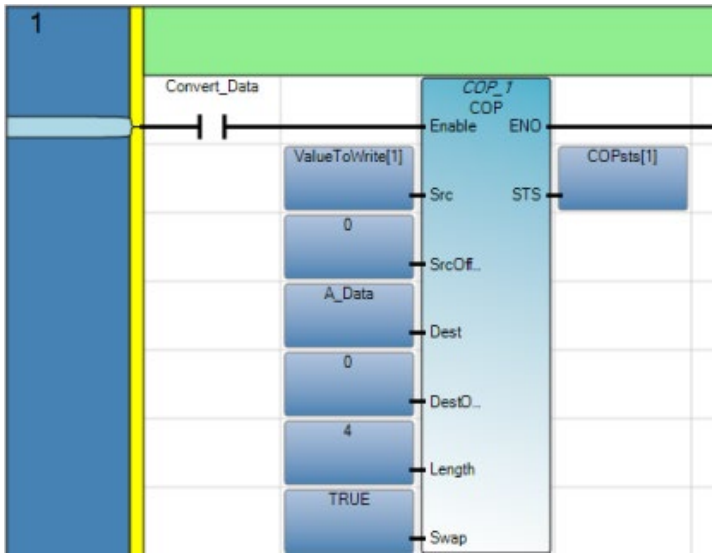
要添加 COP 功能块：

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“项目管理器”。
2. 添加梯形图程序：
   - 在“项目管理器”中，右键单击“程序”，单击“添加”，然后单击“新 LD：梯形图”。
   - 右键单击“项目管理器”中的梯形图图标，单击“重命名”，然后键入“CIPSymbolicMessage”。
   - 双击“项目管理器”中的梯形图编程，以便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“指令块”，并将其拖放到第一个梯形图梯级以打开“指令块选择器”。
   - 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。
4. 添加 COP 变量和初始值：
   - 在梯形图 POU 中，双击“局部变量”以打开“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。
5. 创建数组：
   - 对于 ValueToWrite，双击“维度”，并将数组大小更改为 [1..1]。
   - 对于 A_Data，双击“维度”，并将数组大小更改为 [1..4]。
6. 将下表**值**列的数据输入每个变量的**初始值**字段。
7. 添加触点：
   - 在“工具箱”中，选择“正向触点”，并将其拖放到第一个梯形图梯级中 COP 功能块输入的左侧。
   - 在“变量选择器”中，将名为“Convert_Data”的变量分配给触点。

使用 COP 功能块表中定义的变量。

参数	变量名	数据类型
Src	ValueToWrite	数组 UDINT 初始值： 987654321
SrcOffset	0	UINT
Dest	A_Data	数组 USINT
DestOffset	0	UINT
Length	4	UINT
Swap	TRUE	BOOLEAN
STS	COPsts	数组 UINT

MSG_CIPSYMBOLIC 消息的梯形图程序的第一个梯级应与下图相似。



添加 Equal 运算符和线圈

Equal 指令用于在数据类型转换成功时触发写入值。要添加 Equal (=) 运算符、变量和线圈，请执行以下步骤。

要添加 Equal 运算符：

1. 在“工具箱”中，选择“梯级”，并将其拖放到第一个梯形图梯级的正下方以添加第二个梯级。
2. 添加 Equal 运算符：
   - 在“工具箱”中，选择“指令块”，并将其拖放到第二个梯形图梯级以打开“指令块选择器”。
   - 在“搜索”中，键入“=”，然后双击**“=”**，以将运算符实例添加到梯形图。
3. 要添加 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，分配如下表中所列的变量名。

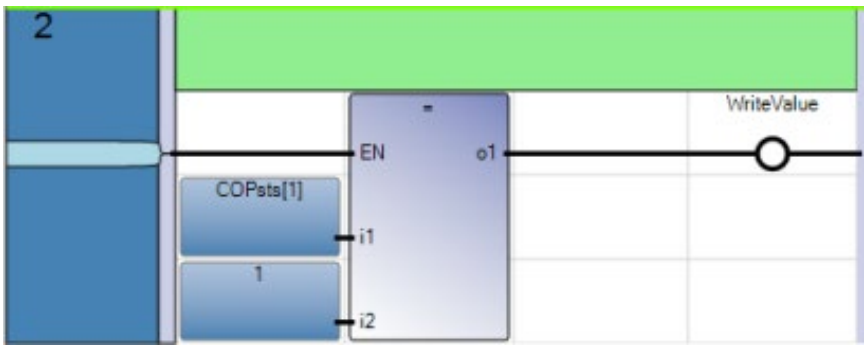
参数	变量名
i1	COPsts
i2	1

4. 要将线圈添加到 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图梯级中 Equal 运算符输出的右侧。

   在“变量选择器”中，在线圈的“名称”字段中键入“WriteValue”。

MSG_CIPGENERIC 消息的梯形图程序的第二个梯级应与下图相似 。



添加 MSG_CIPSYMBOLIC 功能块、变量和触点

要将 MSG_CIPSYMBOLIC 功能块、输入变量和触点添加到梯形图，请执行以下步骤。

要添加功能块和变量：

1. 在“工具箱”中，选择“梯级”，并将其拖放到第二个梯形图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”，并将其拖放到梯形图梯级以显示“指令块选择器”。
   - 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，分配如下表中所列的变量名。
4. 对于“数据”变量，请双击“维度”，并将数组大小更改为 [1...4]。
5. 配置局部变量的初始值，添加 CtrlCfg 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”列中输入以下值。

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

6. 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”列中输入以下值。

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

局部变量选择器应与下图相似。



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

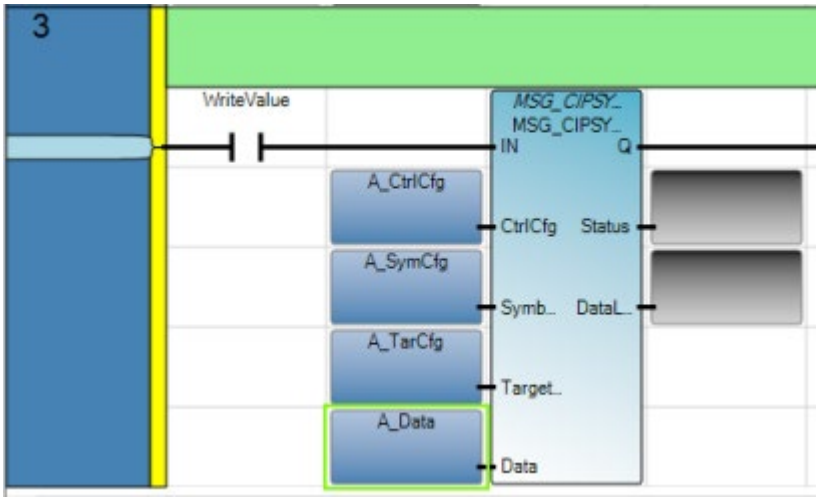
参数	初始值
A_TarCfg.Path	'4,192.168.1.19'
A_TarCfg.CipConnMode	1
A_TarCfg.UcmmTimeout	0
A_TarCfg.ConnMsgTimeout	0
A_TarCfg.ConnClose	留空

用户全局变量选择器应与下图相似。



8. 添加触点：
   - 在“工具箱”中，选择“正向触点”，并将其拖放到第三个梯形图梯级中 MSG_CIPSYMBOLIC 功能块输入的左侧。
   - 在“变量选择器”中，将“WriteValue”变量分配给触点。

MSG_CIPSYMBOLIC 消息的梯形图程序的第三个梯级应与下图相似。



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。

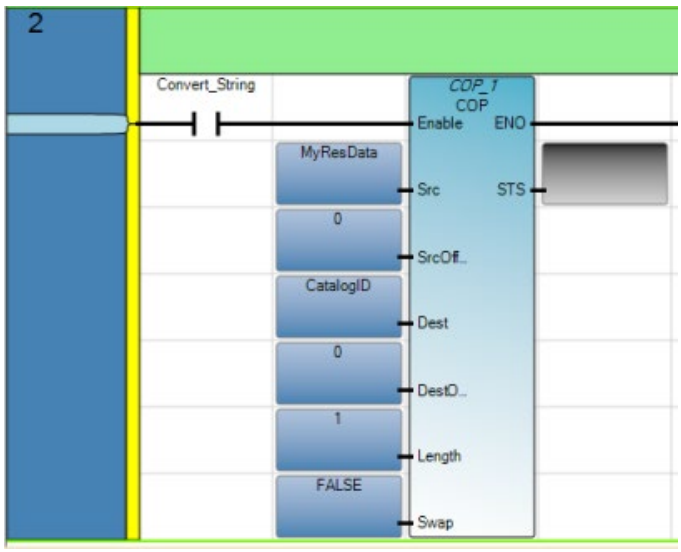
要验证 IP 地址：

1. 打开控制器的应用程序工作区：
2. 在“**项目组织管理器**”中，双击控制器以打开控制器工作区。
3. 在控制器工作区中，展开“控制器”树中的“以太网”，然后单击“Internet 协议”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。



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。

要创建全局变量：

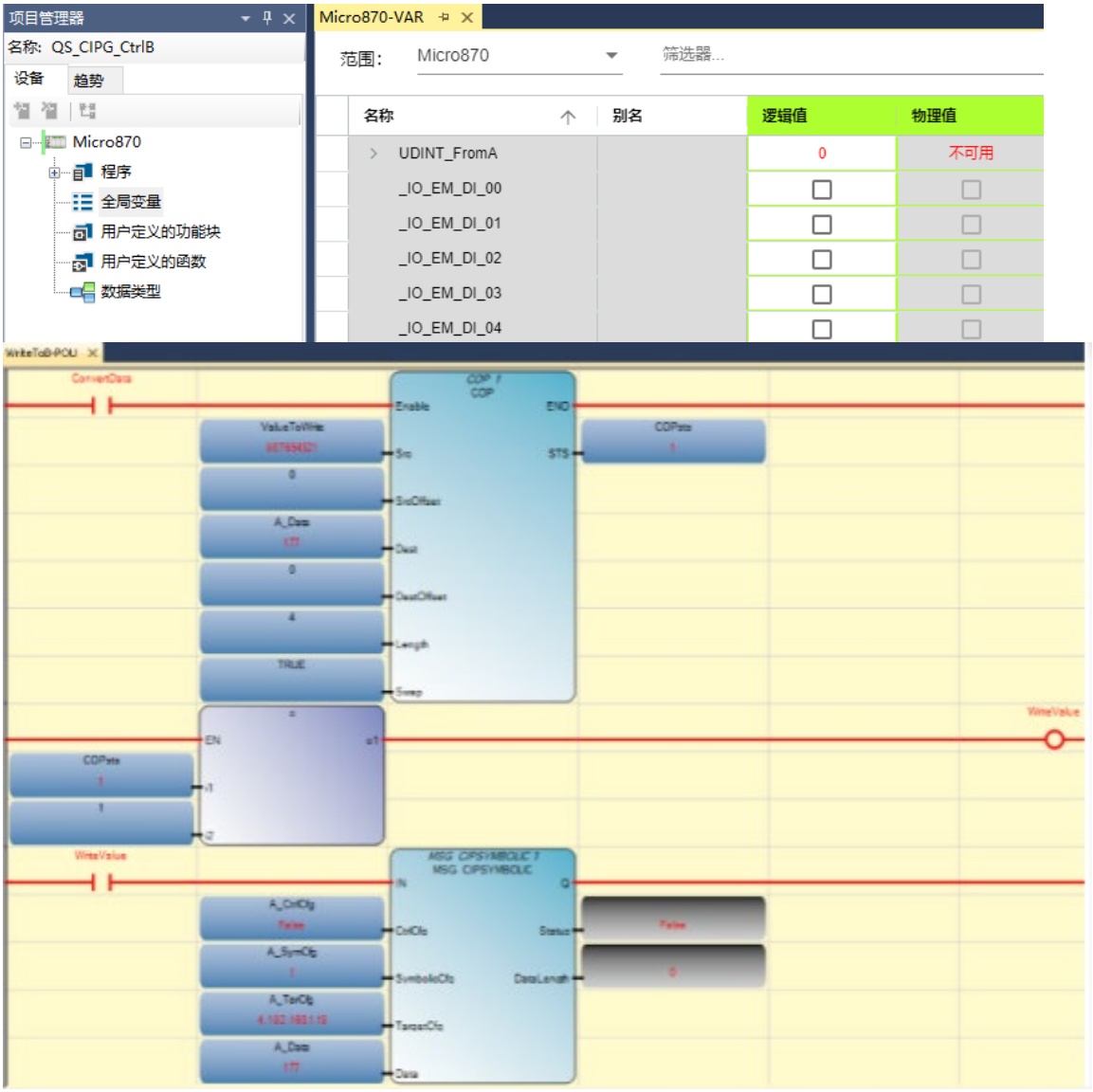
1. 在“项目管理器”中，双击“全局变量”以显示“全局变量选择器”。
2. 在**名称**列中键入 UDINT_FromA。
3. 配置其余参数：
   - 验证数据类型是否为 UDINT。
   - 验证是否已选定“读/写”属性。

“全局变量选择器”应与下图相似。



查看整个程序结果

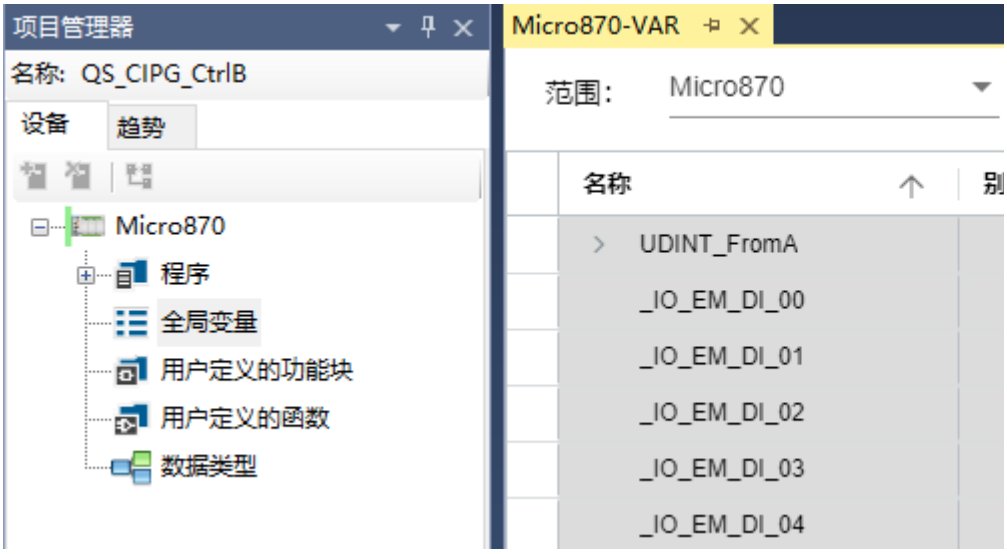
下例展示了整个 MSG_CIPSYMBOLIC 消息程序在其执行后的情况。



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



程序执行后

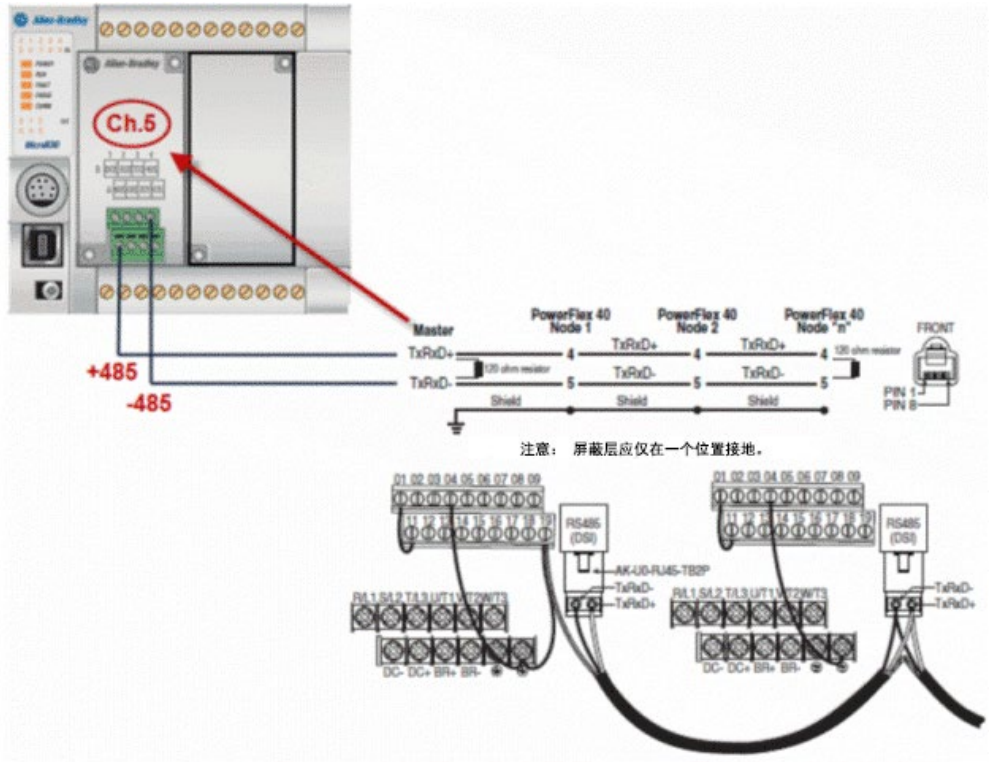


示例：如果配置 Modbus 通信以读取和写入驱动器

这些示例展示了如何将 Modbus 通信配置为使用 MSG_MODBUS 参考页数 168 指令从 PowerFlex 4 驱动器读取状态数据和向其写入控制数据。

Micro830 配线

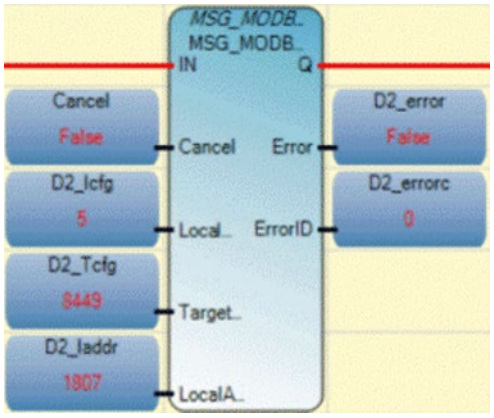
本示例使用在第一个插槽（通道 5）中插有 SERIALISOL 模块的 Micro830 控制器。已连接一个 PowerFlex 40，但下图说明了如何为多点通信配线。有关其他配线信息，请参考用户手册。



AK-U0-RJ45-TB2P 是 RJ45 连接器，带有用于 RS485 通信的 2 个接线端子

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）

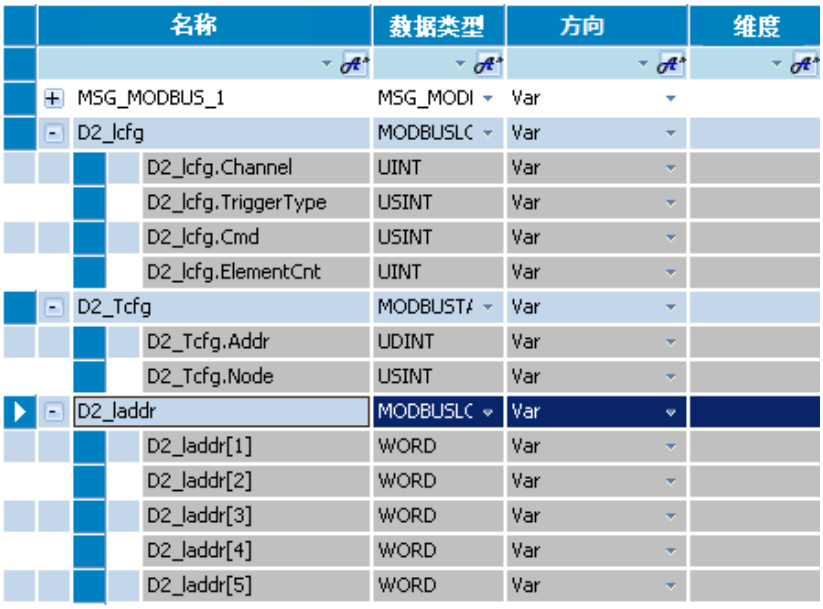
- 处于活动状态（位 1 置位）
- 被指示正向转动（位 2 置位）
- 正在正向转动（位 3 置位）
- 驱动器上某些数字输入的状态

“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。



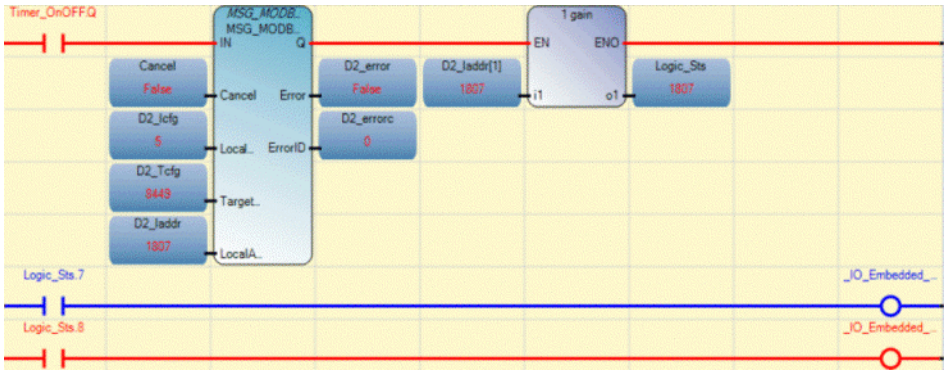
MSG_MODBUS 读取变量

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	描述
*.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	长度
*.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）

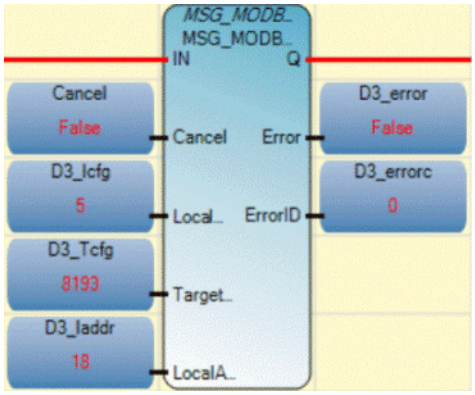
MOV 指令示例

下图展示了使用 MOV 指令将 *_I[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_Icfg	MODBUSLC	Var	
D3_Icfg.Channel	UINT	Var	
D3_Icfg.TriggerType	USINT	Var	
D3_Icfg.Cmd	USINT	Var	
D3_Icfg.ElementCnt	UINT	Var	
D3_Tcfg	MODBUSTA	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	描述
*.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
*.ElementCnt	2	长度
*.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

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。

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 168
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 174
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 165 MSG_CIPGENERIC 参考页数 158
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 165
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 97。

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。

EtherNet/IP

Micro820 和 Micro850 控制器通过一个嵌入式以太网信道 参考页数 222 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 协议是用于经由 EtherNet/IP、ControlNet 和 DeviceNet 网络发送消息的传输和应用层协议，该协议使 HMI 能够轻松连接到 Micro830 或 Micro850 控制器。

CIP 显式消息

CIP 协议用于显式消息。显式消息（配置、数据收集以及诊断）用于传输无需持续更新的数据。显式消息的支持方式为通过 MSG_CIPGENERIC 功能块使用 CIP Generic 客户端消息，以及通过 MSG_CIPSYMBOLIC 功能块使用 CIP Symbolic 客户端消息。

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（例如具有内置串口的 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

Micro820、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
- 非隔离的 USB 编程端口（仅限 Micro830 和 Micro850）
- RJ-45 以太网端口（仅限 Micro820 和 Micro850）

比较指令

使用比较指令，通过表达式或特定比较指令比较整型、实型、时间、日期和字符串值。

指令	描述
(=) Equal 参考页数 223	将第一个输入与第二个输入进行比较以确定是否相等。适用于整形、实型、时间、日期和字符串数据类型。
(>) Greater Than 参考页数 225	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 227	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 228	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 230	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 230	比较输入值以确定第一个输入是否不等于第二个输入。

Equal

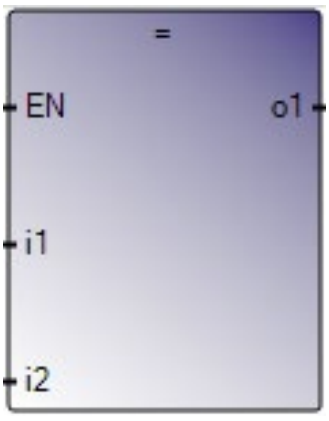
执行将第一个输入与第二个输入进行比较，以确定整型、实型、时间、日期和字符串数据类型是否相等的运算。

操作详细信息：

- 不建议对 TON、TP 和 TOF 指令块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，尽管评估结果仍为错误。例如，在变量输入显示屏上 23.500001 与 23.499999 都将显示为 23.5，但在控制器上则不相等。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

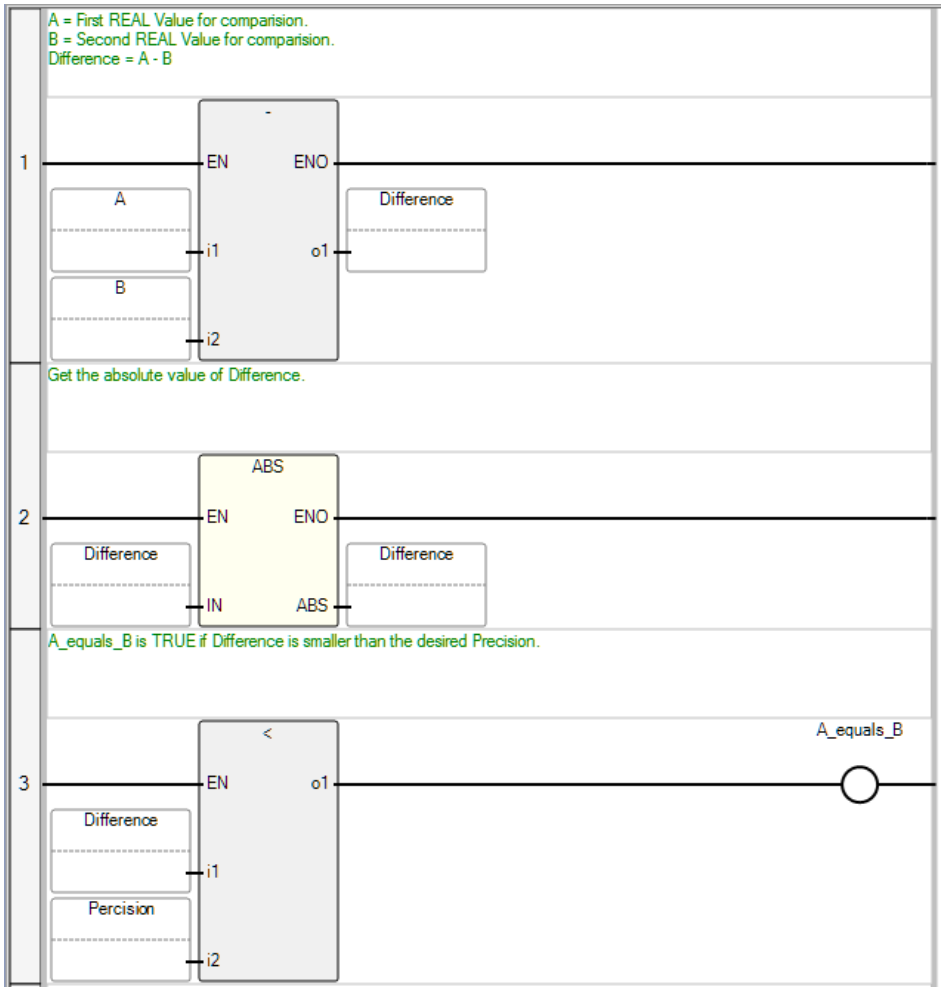
参数	参数类型	数据类型	描述
EN	输入	BOOL	函数启用。 TRUE - 执行相等比较。 FALSE - 不执行任何比较。 仅适用于梯形图编程。
i1	输入	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	所有输入的数据类型必须相同。 时间输入适用于结构化文本、梯形图和功能块图语言。 不建议实型数据类型进行相等值比较。
i2	输入	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	
o1	输出	BOOL	TRUE（如果 i1 = i2）。

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

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 (=) 结构化文本示例

(* 与之等效的 ST：*)

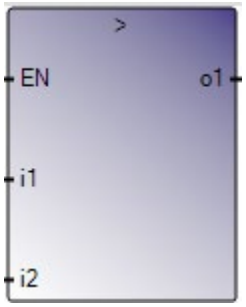
```
aresult := (10 = 25); (* aresult 为 FALSE *)
mresult := ('ab' = 'ab'); (* mresult 为 TRUE *)
```

Greater than

比较整型、实型、时间、日期和字符串输入值，以确定第一个输入是否大于第二个输入。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行输入比较。 FALSE - 不执行任何比较。 仅适用于梯形图编程。
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 (>) 结构化文本示例

(* 与之等效的 ST：*)

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

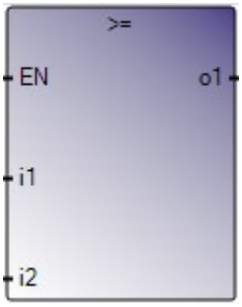
Greater Than or Equal

比较整型、实型、时间、日期和字符串输入值，以确定第一个输入是否大于或等于第二个输入。

对于 TON、TP 和 TOF，不建议进行时间值的相等测试。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行输入比较。 FALSE - 不执行任何比较。 仅适用于梯形图编程。
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。时间输入适用于结构化文本、梯形图和功能块图语言。

参数	参数类型	数据类型	描述
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 or equal (>=) 结构化文本示例

(* 与之等效的 ST：*)

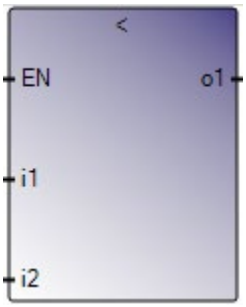
```
aresult := (10 >= 25); (* aresult 为 FALSE *)
mresult := ('ab' >= 'ab'); (* mresult 为 TRUE *)
```

Less Than

比较整型、实型、时间、日期和字符串输入值，以确定第一个输入是否小于第二个输入。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行输入比较。 FALSE - 不执行任何比较。 仅适用于梯形图编程。

参数	参数类型	数据类型	描述
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 (<) ST 结构化文本示例

```
(* 与之等效的 ST: *)
   aresult := (10 < 25); (* aresult 为 TRUE *)
   mresult := ('z' < 'B'); (* mresult 为 FALSE *)
(* 与之等效的 IL: *)
```

LD	10
LT	25
ST	aresult
LD	'z'
LT	'B'

ST	mresult

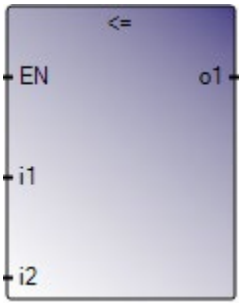
Less Than or Equal

比较整型、实型、时间、日期和字符串输入值，以确定第一个输入是否小于或等于第二个输入。

对于 TON、TP 和 TOF，不建议进行时间值的相等测试。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行输入比较。 FALSE - 不执行任何比较。 仅适用于梯形图编程。
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。时间输入适用于结构化文本、梯形图和功能块图语言。
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 (<=) 结构化文本示例

(* 与之等效的 ST：*)

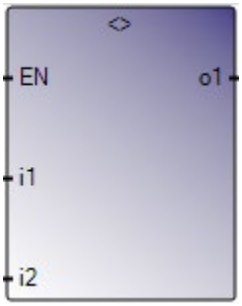
```
aresult := (10 <= 25); (* aresult 为 TRUE *)
mresult := ('ab' <= 'ab'); (* mresult 为 TRUE *)
```

Not equal

比较整型、实型、时间、日期和字符串输入值，以确定第一个输入是否不等于第二个输入。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前比较计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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	

参数	参数类型	数据类型	描述
		DATE STRING	
o1	输出	BOOL	TRUE - 如果第一个 <> 第二个。

Not equal (<>) 结构化文本示例

(* 与之等效的 ST：*)

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

计数器指令

计数器指令用于根据事件数量控制操作。

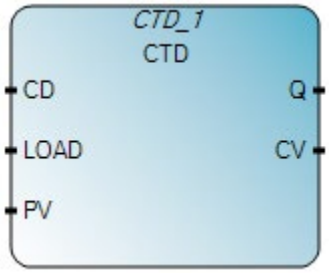
指令	描述
CTD 参考页数 233	它从给定值到 0 逐个向下计数（整数）。
CTU 参考页数 235	从 0 到给定值逐个向上计数（整数）。
CTUD 参考页数 237	从 0 到给定值逐个向上计数（整数），或从给定值到 0（逐个）向下计数。

CTD（向下计数）

它从给定值到 0 逐个向下计数（整数）。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

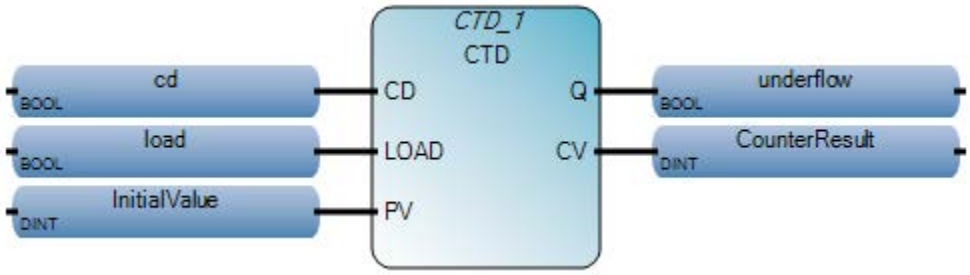
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



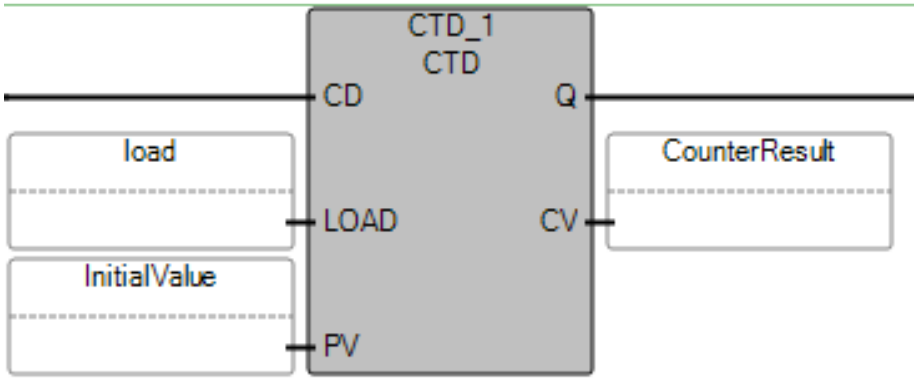
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
CD	输入	BOOL	向下计数。 TRUE - 检测到上升沿，以 1 为增量向下计数。 FALSE - 检测到下降沿，计数器值保持为相同值。
LOAD	输入	BOOL	Load 用于根据向下计数值验证 PV 值。 TRUE - 设置 CV=PV。 FALSE - 以 1 为增量继续向下计数。
PV	输入	DINT	计数器的编程最大值。
Q	输出	BOOL	表示向下计数指令是否已生成小于或等于计数器最大值的数。 TRUE - 计数器结果 <=0（上溢条件）。 FALSE - 计数器结果 >0。
CV	输出	DINT	当前计数器值。

CTD 功能块图示例



CTD 梯形图示例



CTD 结构化文本示例




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

(* 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

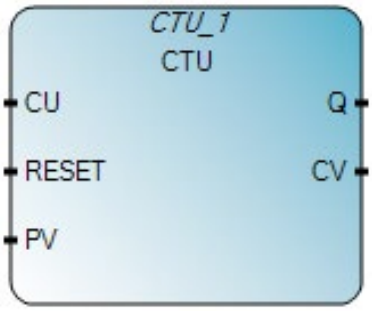


CTU（向上计数）

CTU 从 0 到给定值逐个向上计数（整数）。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

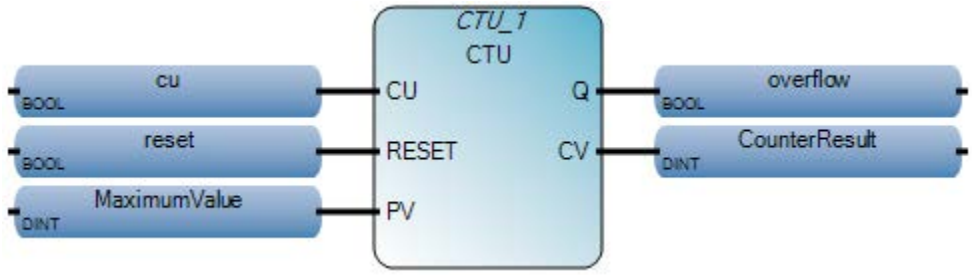
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



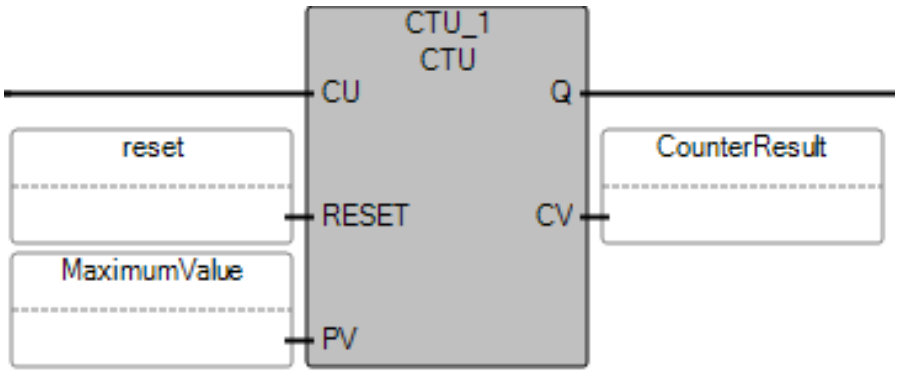
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
CU	输入	BOOL	向上计数。 TRUE - 检测到上升沿，以 1 为增量向上计数。 FALSE - 检测到下降沿，计数器值保持为相同值。
RESET	输入	BOOL	Reset 用于根据向上计数值验证 PV 值。 TRUE - 将 CV 值设置为零。 FALSE - 以 1 为增量继续向上计数。
PV	输入	DINT	计数器的编程最大值。
Q	输出	BOOL	表示向上计数指令是否已生成大于或等于计数器最大值的数。 TRUE - 计数器结果 => PV（上溢条件）。 FALSE - 计数器结果 < PV
CV	输出	DINT	当前计数器结果。

CTU 功能块图示例



CTU 梯形图示例



CTU 结构化文本示例

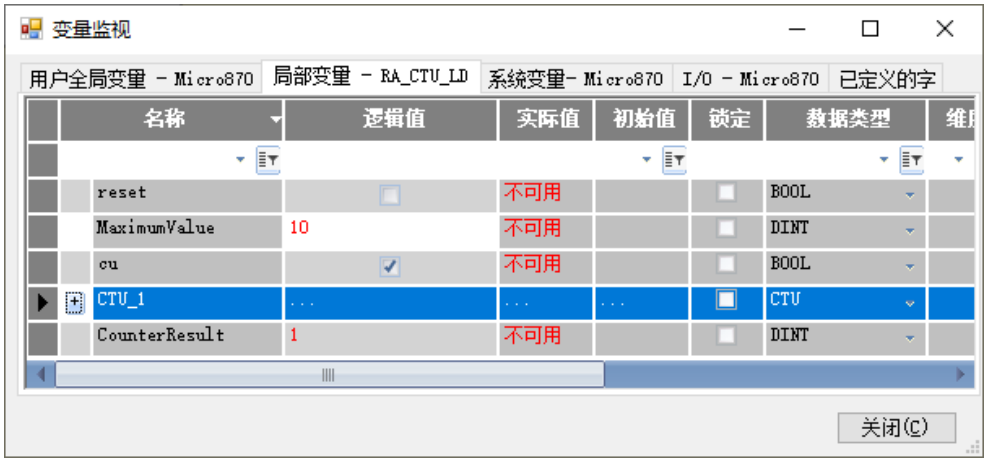



```
1  MaximumValue := 10;
2  CTU_1(cu, reset, MaximumValue);
```

(* 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

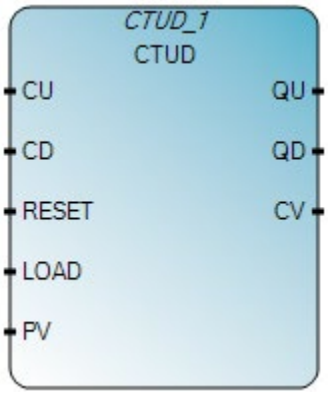


CTUD(向上/向下计数)

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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

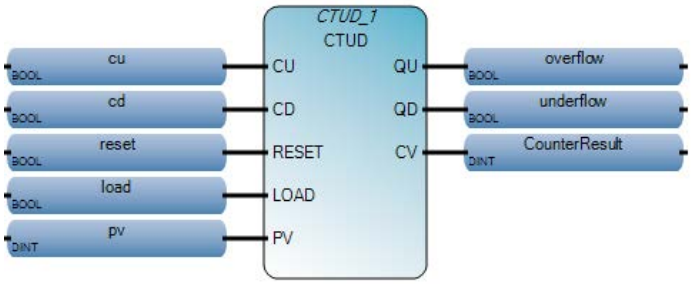
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



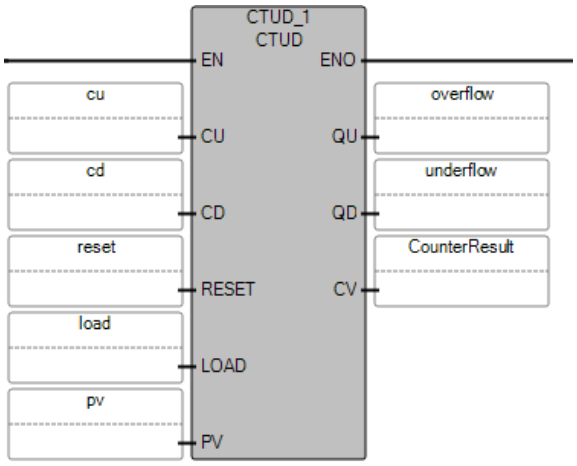
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
CU	输入	BOOL	TRUE - 检测到上升沿，向上计数。
CD	输入	BOOL	TRUE - 检测到上升沿，向下计数。
RESET	输入	BOOL	复位主导（确定指令行为时的最高优先级）命令。 （当 RESET 为 TRUE 时 CV = 0）。
LOAD	输入	BOOL	Load 命令。 TRUE - 设置 CV = PV。
PV	输入	DINT	编程最大值。
QU	输出	BOOL	溢出。 TRUE - 当 CV >= PV 时。
QD	输出	BOOL	下溢。 TRUE - 当 CV <= 0 时。
CV	输出	DINT	计数器结果。

CTUD 功能块图示例

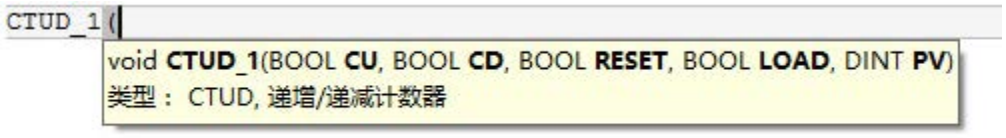


CTUD 梯形图示例



CTUD 结构化文本示例

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



(* 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;
```

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。

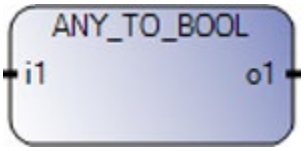
指令	描述
ANY_TO_BOOL 参考页数 239	将非布尔值转换为布尔值。
ANY_TO_BYTE 参考页数 240	将值转换为字节。
ANY_TO_DATE 参考页数 241	将字符串、整型、实型或时间数据类型转换为日期数据类型。
ANY_TO_DINT 参考页数 243	将值转换为双整型。
ANY_TO_DWORD 参考页数 244	将值转换为双字值。
ANY_TO_INT 参考页数 245	将值转换为整型。
ANY_TO_LINT 参考页数 246	将值转换为长整型。
ANY_TO_LREAL 参考页数 247	将值转换为长实型。
ANY_TO_LWORD 参考页数 248	将值转换为长字型。
ANY_TO_REAL 参考页数 248	将值转换为实型。
ANY_TO_SINT 参考页数 249	将值转换为短整型。
ANY_TO_STRING 参考页数 250	将值转换为字符串。
ANY_TO_TIME 参考页数 252	将值转换为时间数据类型。
ANY_TO_UDINT 参考页数 253	将值转换为无符号双整型。
ANY_TO_UINT 参考页数 253	将值转换为无符号整型。
ANY_TO_ULINT 参考页数 254	将值转换为无符号长整型。
ANY_TO_USINT 参考页数 255	将值转换为无符号短整型。
ANY_TO_WORD 参考页数 256	将值转换为字。

ANY_TO_BOOL

将非布尔值转换为布尔值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为布尔计算。 FALSE - 不执行任何计算。 适用于梯形图编程。

参数	参数类型	数据类型	描述
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 结构化文本示例

(* 与之等效的 ST：*)

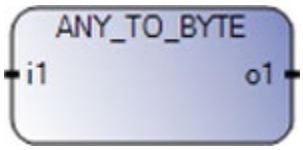
```
ares := ANY_TO_BOOL(10);          (* ares 为 TRUE *)
tres := ANY_TO_BOOL(t#0s);        (* tres 为 FALSE *)
mres := ANY_TO_BOOL('FALSE');     (* mres 为 FALSE *)
```

ANY_TO_BYTE

将值转换为 8 位字节值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为字节计算。 FALSE - 不执行任何计算。 适用于梯形图编程。

参数	参数类型	数据类型	描述
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。 仅适用于梯形图编程。

ANY_TO_BYTE 结构化文本示例

(* 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 *)
```

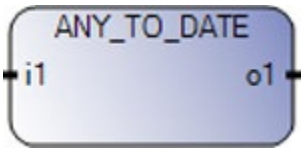
ANY_TO_DATE

将字符串、整型、实型或时间数据类型转换为日期数据类型。

布尔、Byte 和 Word 日期类型受支持，但不建议使用。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



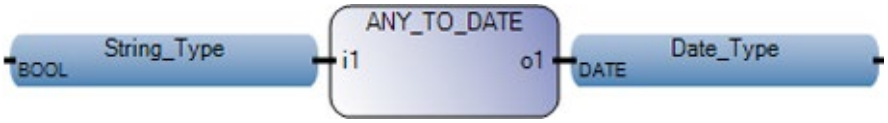
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为日期计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。

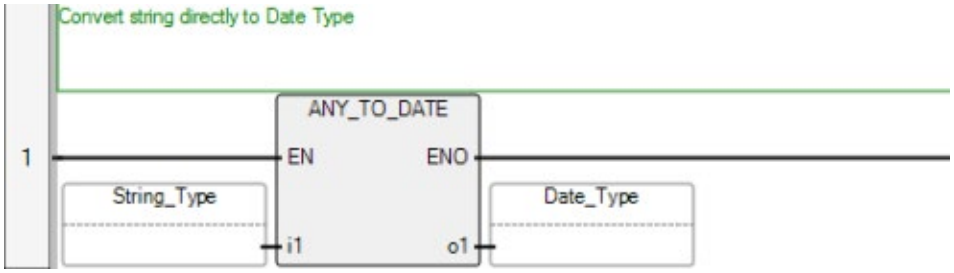
参数	参数类型	数据类型	描述
i1	输入	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME STRING	• 字符串将直接转换为日期数据类型，并且必须具有 YYYY-MM-DD、DATE#YYYY-MM-DD 或 D#YYYY-MM-DD 格式。 • 整型和实型必须为正值（以秒为单位），并添加到日期 1970-01-01。 • 时间添加到日期 1970-01-01。
o1	输出	DATE	转换的日期值。转换期间发生错误（如为负值的日期）会导致无效的结果。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

ANY_TO_DATE 功能块图示例

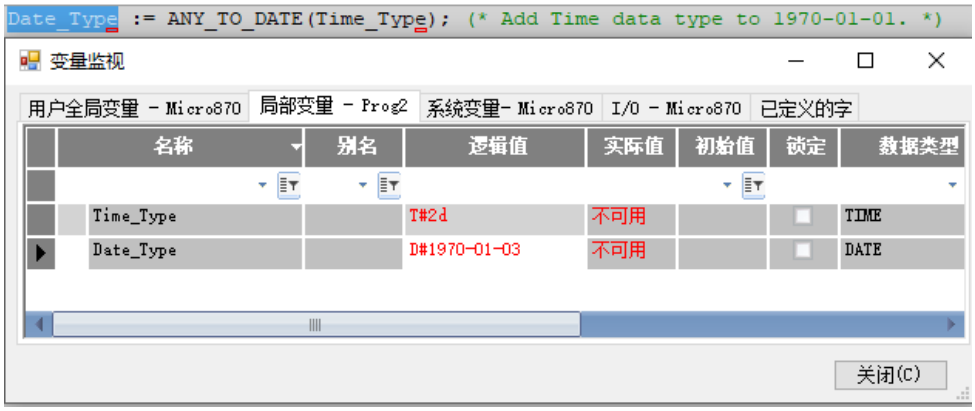
ANY_TO_DATE



ANY_TO_DATE 梯形图示例



ANY_TO_DATE 结构化文本示例

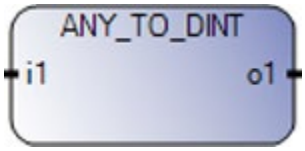


ANY_TO_DINT

将值转换为 32 位双整型值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为 32 位双整型计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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”。 仅适用于梯形图编程。

ANY_TO_DINT 结构化文本示例

```
(* 与之等效的 ST：*)

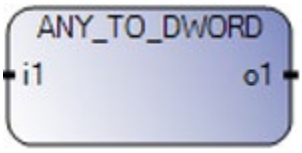
bres := ANY_TO_DINT (true);          (* bres 为 1 *)
tres := ANY_TO_DINT (t#1s46ms);      (* tres 为 1046 *)
mres := ANY_TO_DINT ('0198');        (* mres 为 198 *)
```

ANY_TO_DWORD

将值转换为 32 位双字值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为 32 位双字计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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”。 仅适用于梯形图编程。

ANY_TO_DWORD 结构化文本示例

(* 与之等效的 ST：*)

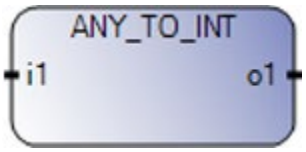
```
bres := ANY_TO_DWORD (true);          (* bres 为 1 *)
tres := ANY_TO_DWORD (t#1s46ms);      (* tres 为 1046 *)
mres := ANY_TO_DWORD ('0198');        (* mres 为 198 *)
```

ANY_TO_INT

将值转换为 16 位整型值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为 16 位整型计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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”。 仅适用于梯形图编程。

ANY_TO_INT 结构化文本示例

(* 与之等效的 ST：*)

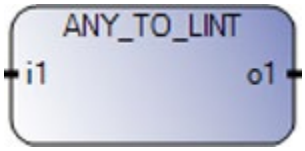
```
bres := ANY_TO_INT (true);                (* bres 为 1 *)
tres := ANY_TO_INT (t#0s46ms);            (* tres 为 46 *)
mres := ANY_TO_INT ('0198');              (* mres 为 198 *)
```

ANY_TO_LINT

将值转换为 64 位长整型值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为 64 位长整型计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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”。 仅适用于梯形图编程。

ANY_TO_LINT 结构化文本示例

(* 与之等效的 ST： *)

```
bres := ANY_TO_LINT (true);               (* bres 为 1 *)
tres := ANY_TO_LINT (t#0s46ms);           (* tres 为 46 *)
```

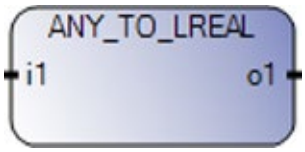
```
mres := ANY_TO_LINT ('0198');                    (* mres 为 198 *)
```

ANY_TO_LREAL

将任意值转换为长实型值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为长实型计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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”。 仅适用于梯形图编程。

ANY_TO_LREAL 结构化文本示例

(* 与之等效的 ST： *)

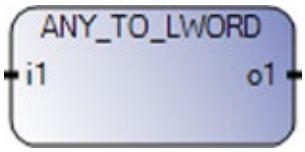
```
bres := ANY_TO_LREAL (true);                     (* bres 为 1.0 *)
tres := ANY_TO_LREAL (t#1s46ms);                 (* tres 为 1046.0 *)
ares := ANY_TO_LREAL (198);                      (* ares 为 198.0 *)
```

ANY_TO_LWORD

将值转换为 64 位长字值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为 64 位长字计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
i1	输入	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT DWORD LINT ULINT REAL LREAL IME DATE STRING	除长字之外的任何值。
o1	输出	LWORD	64 位长字值。
ENO	输出	BOOL	启用“输出”。 仅适用于梯形图编程。

ANY_TO_LWORD 结构化文本示例

(* 与之等效的 ST： *)

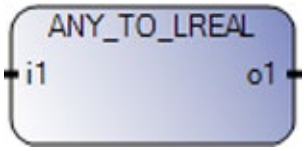
```
bres := ANY_TO_LWORD (true);          (* bres 为 1 *)
tres := ANY_TO_LWORD (t#0s46ms);      (* tres 为 46 *)
mres := ANY_TO_LWORD ('0198');        (* mres 为 198 *)
```

ANY_TO_REAL

将值转换为实型值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为实型计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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"。 仅适用于梯形图编程。

ANY_TO_REAL 结构化文本示例

(* 与之等效的 ST： *)

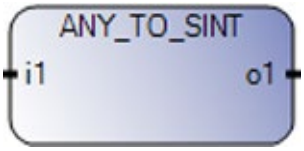
```
bres := ANY_TO_REAL(true);          (* bres 为 1.0 *)
tres := ANY_TO_REAL(t#1s46ms);      (* tres 为 1046.0 *)
ares := ANY_TO_REAL(198);           (* ares 为 198.0 *)
```

ANY_TO_SINT

将值转换为短整型值。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为 8 位短整型计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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"。 仅适用于梯形图编程。

ANY_TO_SINT 结构化文本示例

(* 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 -58*）
```

ANY_TO_STRING

将值转换为字符串值。

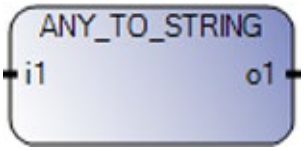
操作详细信息：

- 将 REAL 数据类型转换为 String 时，ANY_TO_STRING 指令使用 IEEE 754 格式。
  - ANY_TO_STRING 将 125.0 转换为 1.25000E+02
- 当目标字符串长度为 5 个字符时：

- ANY_TO_STRING 将 125.0 转换为 1.25000E+02 并向目标字符串输出 1.250。
- 考虑创建用户定义的功能块以从指数符号转换为数。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为字符串计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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”。 仅适用于梯形图编程。

ANY_TO_STRING 结构化文本示例

(* 与之等效的 ST： *)

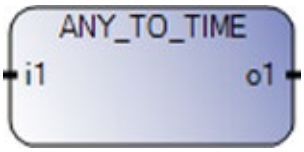
```
bres := ANY_TO_STRING (TRUE);          (* bres 为 TRUE *)
ares := ANY_TO_STRING (125);           (* ares 为 125 *)
```

ANY_TO_TIME

将非时间或非日期值转换为时间值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为时间计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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”。 仅适用于梯形图编程。

ANY_TO_TIME 结构化文本示例

(* 与之等效的 ST： *)

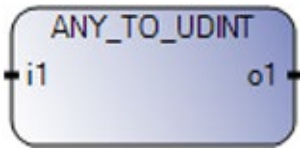
```
ares := ANY_TO_TIME (1256);            (* ares := t#1s256ms *)
rres := ANY_TO_TIME (1256.3);          (* rres := t#1s256ms *)
```

ANY_TO_UDINT

将值转换为 32 位无符号双整型值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为 32 位双整型计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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”。 仅适用于梯形图编程。

ANY_TO_UDINT 结构化文本示例

(* 与之等效的 ST： *)

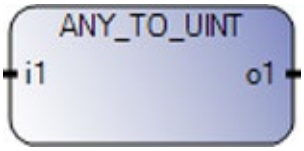
```
bres := ANY_TO_UDINT (true);          (* bres 为 1 *)
tres := ANY_TO_UDINT (t#1s46ms);      (* tres 为 1046 *)
mres := ANY_TO_UDINT ('0198');        (* mres 为 198 *)
```

ANY_TO_UINT

将值转换为无符号整型值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为 16 位无符号整型计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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"。 仅适用于梯形图编程。

ANY_TO_UINT 结构化文本示例

(* 与之等效的 ST：*)

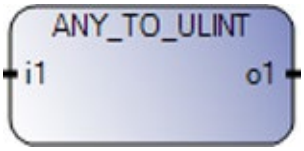
```
bres := ANY_TO_UINT (true);          (* bres 为 1 *)
tres := ANY_TO_UINT (t#0s46ms);      (* tres 为 46 *)
mres := ANY_TO_UINT ('0198');        (* mres 为 198 *)
```

ANY_TO_ULINT

将值转换为 64 位无符号长整型值。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为 64 位无符号长整型计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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”。 仅适用于梯形图编程。

ANY_TO_ULINT 结构化文本示例

(* 与之等效的 ST：*)

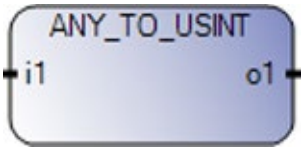
```
bres := ANY_TO_ULINT (true);          (* bres 为 1 *)
tres := ANY_TO_ULINT (t#0s46ms);      (* tres 为 46 *)
mres := ANY_TO_ULINT ('0198');        (* mres 为 198 *)
```

ANY_TO_USINT

将值转换为无符号短整型值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为 8 位无符号短整型计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
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”。 仅适用于梯形图编程。

ANY_TO_USINT 结构化文本示例

(* 与之等效的 ST：*)

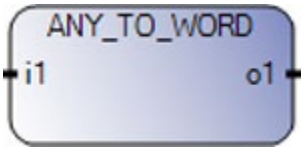
```
bres := ANY_TO_USINT (true);          (* bres 为 1*)
tres := ANY_TO_USINT (t#0s46ms);      (* tres 为 46 *)
mres := ANY_TO_USINT ('0198');        (* mres 为 198 *)
```

ANY_TO_WORD

将值转换为 16 位 Word 值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行转换为 16 位字计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
i1	输入	BOOL SINT USINT BYTE INT DINT UDINT DWORD LINT ULINT LWORD REAL LREAL TIME DATE STRING	除字值之外的任何值。
o1	输出	WORD	字值。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

ANY_TO_WORD 结构化文本示例

(* 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 *)
```

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。

指令	描述
AVERAGE 参考页数 259	计算若干定义样本上的运行平均值。
COP 参考页数 260	将源元素中的二进制数据复制到目标元素。
MAX 参考页数 267	计算两个整型值的最大值。
MIN 参考页数 265	计算两个整型值的最小值。

AVERAGE

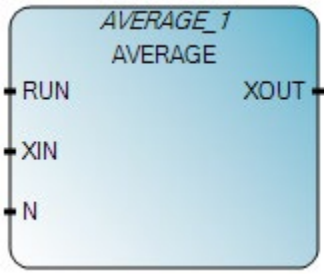
计算若干定义的示例上的运行平均值，并在每次循环时存储值。

操作详细信息：

- 定义的示例数 (N) 不能超过 127。
- 如果设置或更改 N 的值，则将 RUN 设为 FALSE，然后再将其设置回 TRUE。
- 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。
- 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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

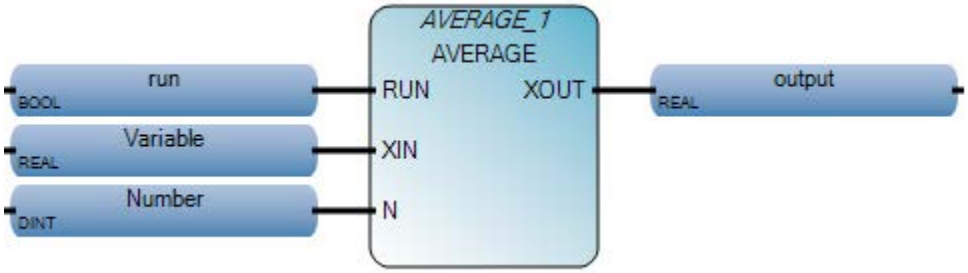


使用此表有助于确定该指令的参数值。

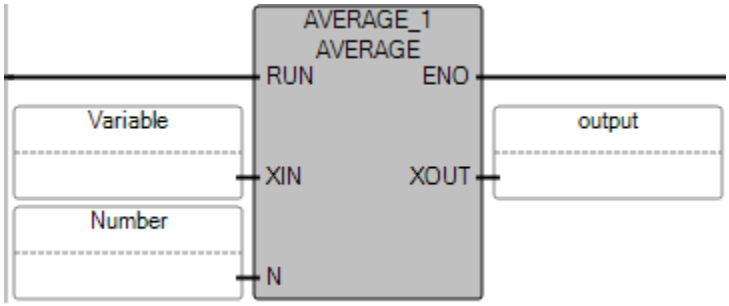
参数	参数类型	数据类型	描述
RUN	输入	BOOL	TRUE = 运行 FALSE = 复位
XIN	输入	REAL	任何实型变量。
N	输入	DINT	应用程序定义的示例数量。

参数	参数类型	数据类型	描述
XOUT	输出	REAL	XIN 值的运行平均值。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

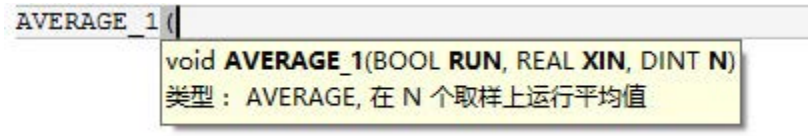
AVERAGE 功能块图示例



AVERAGE 梯形图示例



AVERAGE 结构化文本示例




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

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

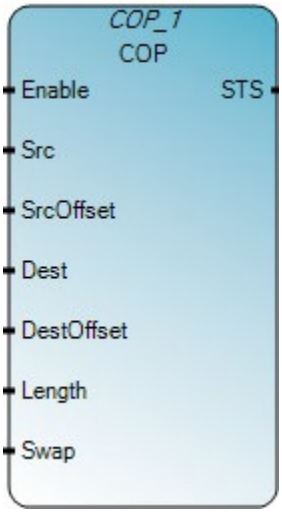
```
AVERAGE1((auto_mode & store_cmd), sensor_value, 100);
ave_value := AVERAGE1.XOUT;
```

COP

将源元素中的二进制数据复制到目标元素。源元素保持不变。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型		描述
Enable	输入	BOOL		启用指令块。COP 由级别触发。 TRUE - 执行复制。 FALSE - 功能块处于空闲状态。
Src	输入	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT	DWORD REAL TIME DATE STRING LWORD ULINT LINT LREAL	要复制的源元素（一维数组或非数组）。 如果源为 STRING 数据类型，则目标必须为 STRING 数据类型或 USINT（UCHAR 和 BYTE）数据类型。否则，将报告数据类型不匹配。
SrcOffset	输入	UINT		输入参数，用于定义源数据一维数组中开始复制过程的起始点。 将偏移设置为 0： • 如果不是数组数据类型， 或 • 从一维数组数据类型的首个元素开始复制。
Dest	输入	BOOL SINT USINT BYTE INT UINT WORD DINT UDINT	DWORD REAL TIME DATE STRING LWORD ULINT LINT	目标元素，用于存储从一维数组或非数组源元素复制的值。 如果目标为 STRING 数据类型，则源必须为 STRING 数据类型或 USINT（UCHAR 和 BYTE）数据类型。否则，将报告数据类型不匹配。
DestOffset	输入	UINT		输入参数，用于定义在目标数据中存储所复制元素的起始点。 将偏移设置为 0： • 如果不是数组数据类型， 或 • 从一维数组数据类型的首个元素开始复制。

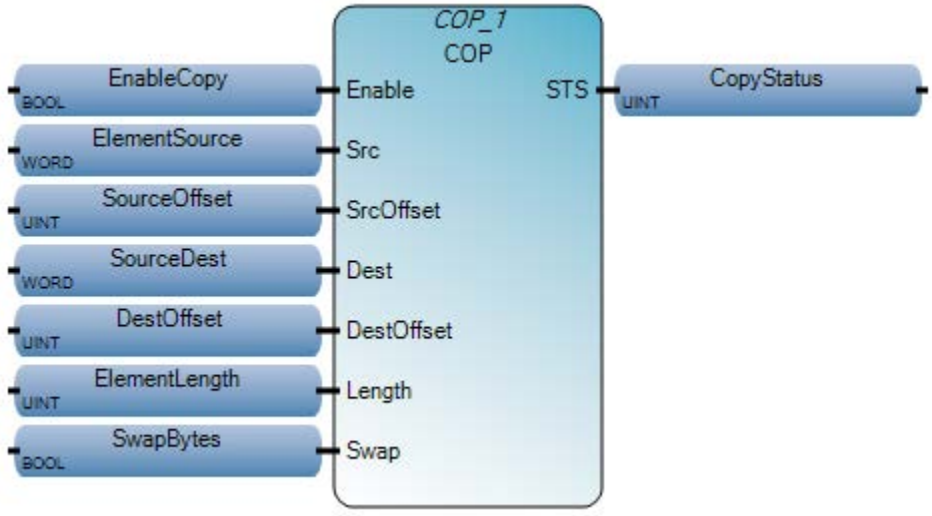
参数	参数类型	数据类型	描述
Length	输入	UINT	要复制的目标元素个数。 如果目标为 STRING 数据类型，则它表示要复制的字符串数量。
Swap	输入	BOOL	字节交换发生在提取和操作高低 8 位参数之前 • 启用 Swap 时，在处理高低字节之前字节顺序反向。例如，像 0x1234 这种高字节为 0x12、低字节为 0x34 的 16 位值，在启用 Swap 的情况下，其字节顺序将反向为 0x3412。 • 禁用 Swap 时，高低字节将保持原始顺序。例如，像 0x 参考页数这种高字节为 0xAB、低字节为 0xCD 的 16 位值，在禁用 Swap 的情况下，其字节顺序将保持不变。 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为 STRING， 或 • 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。Sts 参数的定义在 COP 状态代码中定义。
ENO	输出	BOOL	启用输出。 仅适用于梯形图编程。

COP 状态代码 (Sts)

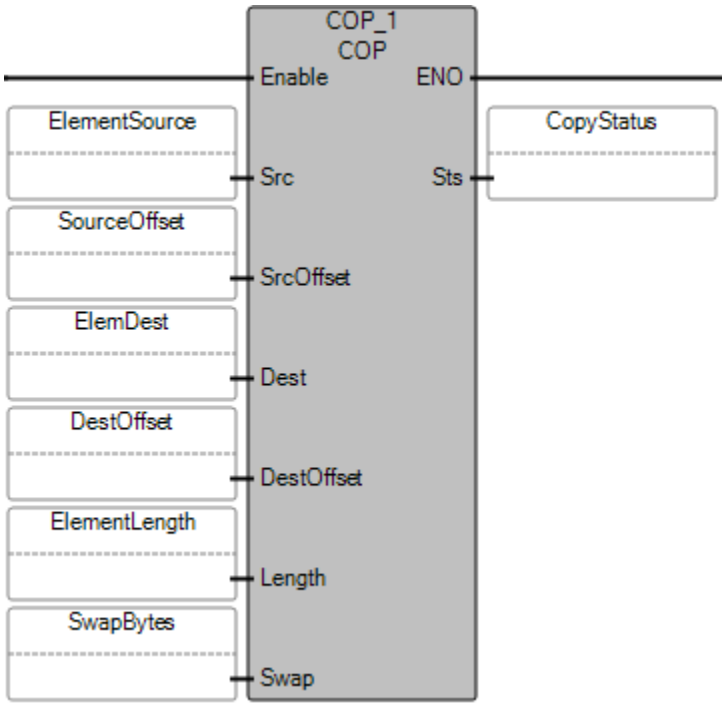
下表介绍了 COP 状态代码。

COP 状态代码	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。

COP 功能块图示例



COP 梯形图示例



COP 结构化文本示例

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

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 参考页数 260，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

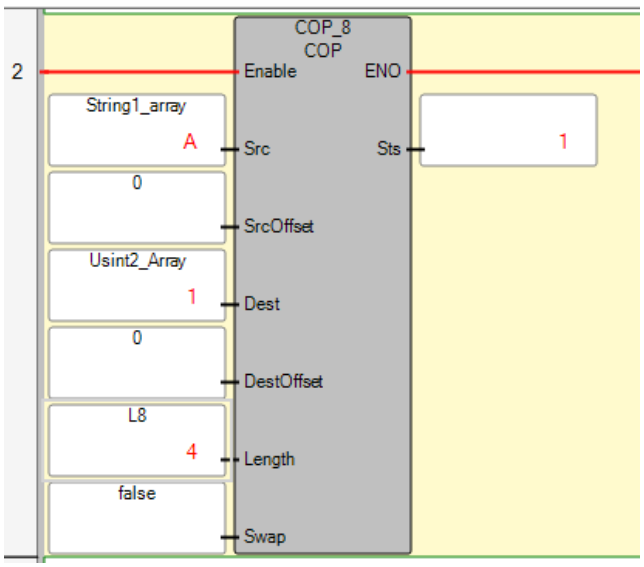
COP 字符串数组示例

以下示例显示了用于将字符串数组复制到 Usint 数组的 COP 参考页数 260 指令。此 COP 指令将跳过源数组中长度为零的所有元素。

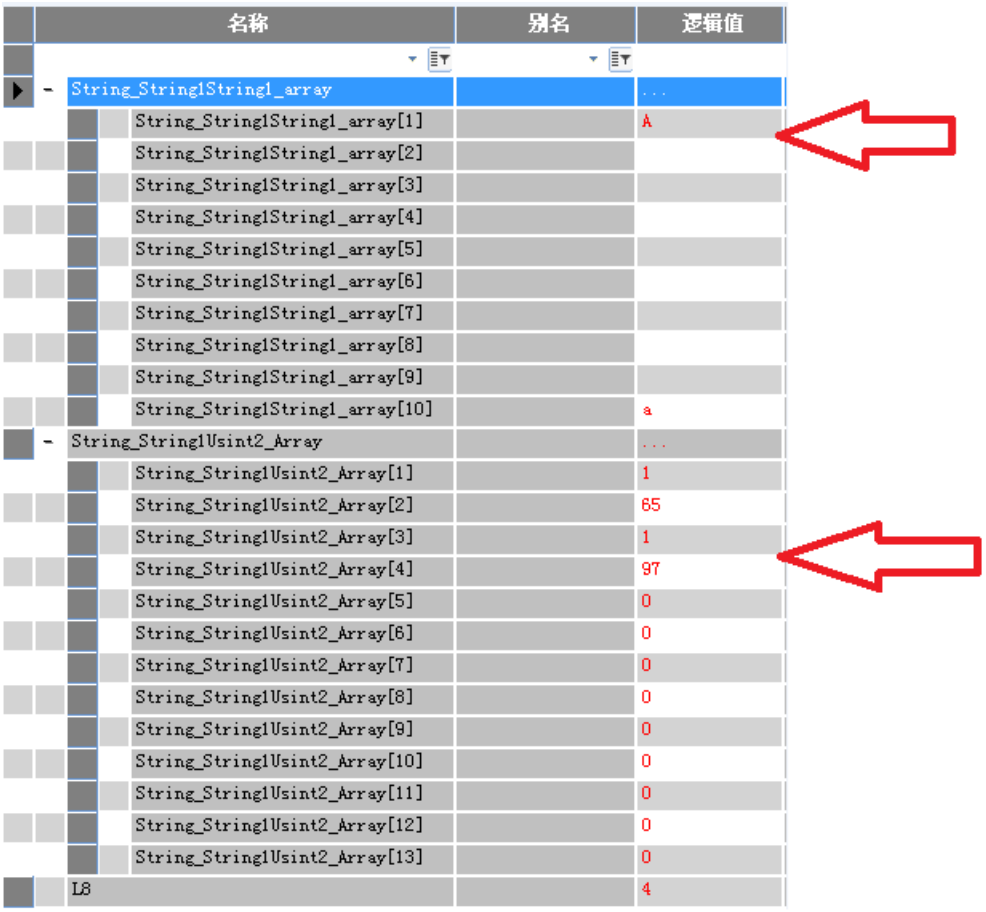
对于本示例而言：

- 长度被指定为 4 个字节。
- 要复制的目标元素数量为 4 个字节。
- 长度为零的所有数组元素（空格）都将被跳过。
- 此 COP 指令会查找字符串数组 [1] 中长度不为零的元素，并使用 1 将其复制到目标 USINT 数组 [1]。 一是字符串数组 [1] 中字符串的长度，USINT 阵列 [2] 为 65（是“A”的 ASCII 码）。
- 此 COP 指令会查找字符串数组 [10] 中长度不为零的元素（为 a），并使用 1 将其复制到目标 USINT 数组 [3]。一是字符串数组 [10] 中字符串的长度，USINT 阵列 [4] 为 97（是“a”的 ASCII 码）。

梯形图



含有逻辑值的数组元素



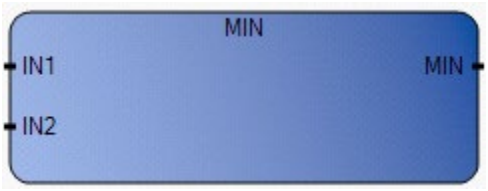
名称	别名	逻辑值
String_String1String1_array		...
String_String1String1_array[1]		A
String_String1String1_array[2]		
String_String1String1_array[3]		
String_String1String1_array[4]		
String_String1String1_array[5]		
String_String1String1_array[6]		
String_String1String1_array[7]		
String_String1String1_array[8]		
String_String1String1_array[9]		
String_String1String1_array[10]		a
String_String1Usint2_Array		...
String_String1Usint2_Array[1]		1
String_String1Usint2_Array[2]		65
String_String1Usint2_Array[3]		1
String_String1Usint2_Array[4]		97
String_String1Usint2_Array[5]		0
String_String1Usint2_Array[6]		0
String_String1Usint2_Array[7]		0
String_String1Usint2_Array[8]		0
String_String1Usint2_Array[9]		0
String_String1Usint2_Array[10]		0
String_String1Usint2_Array[11]		0
String_String1Usint2_Array[12]		0
String_String1Usint2_Array[13]		0
L8		4

MIN（最小值）

计算两个整型值的最小值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

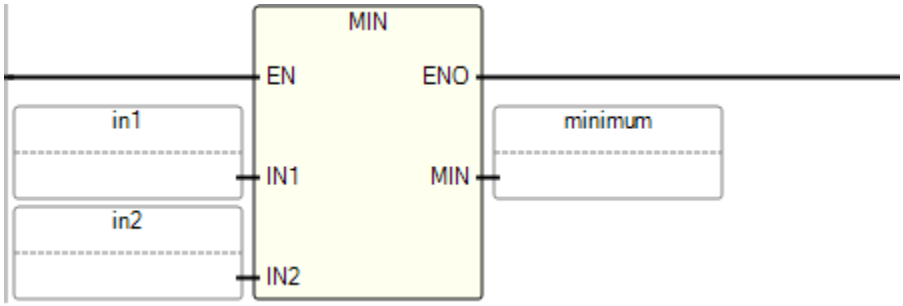
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行最小整数值计算 FALSE - 不执行任何计算。 适用于梯形图编程。
IN1	输入	DINT	任何有符号整型值。
IN2	输入	DINT	不能为实型。
MIN	输出	DINT	两个输入值的最小值。

参数	参数类型	数据类型	描述
ENO	输出	BOOL	启用"输出"。 适用于梯形图编程。

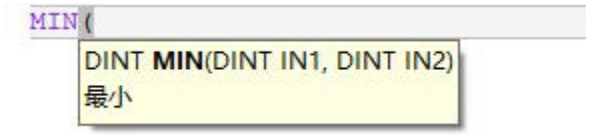
MIN 功能块图示例



MIN 梯形图示例



MIN 结构化文本示例



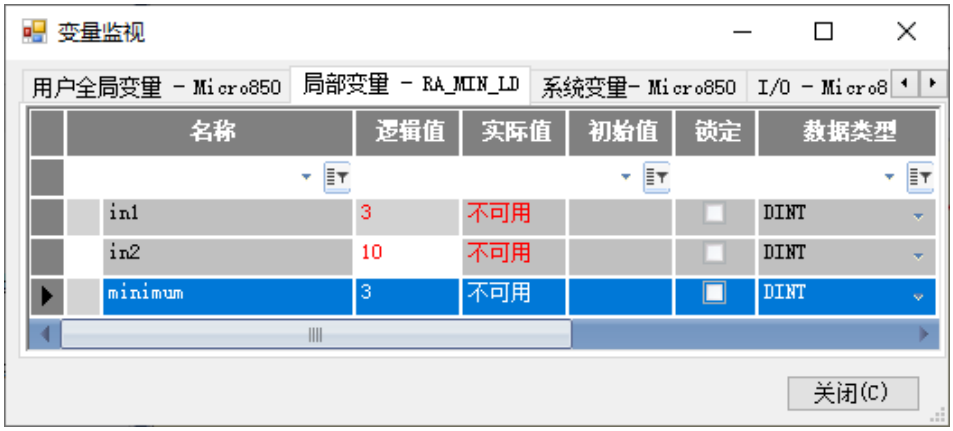

```
1  in1 := 3;
2  in2 := 10;
3  minimum := MIN(in1, in2);
```

(* 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



MAX（最大值）

计算两个整型值的最大值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



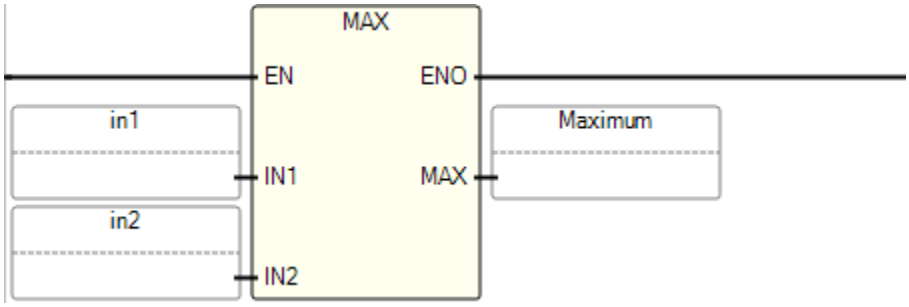
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行最大整数值计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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”。 适用于梯形图编程。

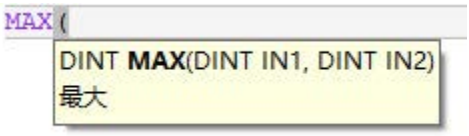
MAX 功能块图示例



MAX 梯形图示例



MAX 结构化文本示例



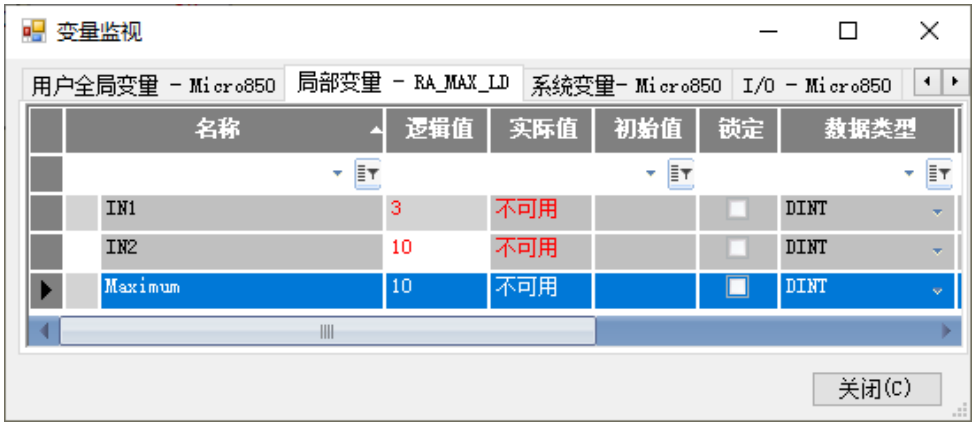

```
1  IN1 := 3;
2  IN2 := 10;
3  Maximum := MAX(IN1, IN2);
```

(* 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



高速计数器 (HSC) 指令

使用高速计数器指令监视和控制高速计数器。

指令	描述
高速计数器 参考页数 269	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 284	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 (HSC) 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 ±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、Micro850 和 Micro870 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 参考页数 269 用于配置高速计数器硬件和更新图像累加器。

重要事项 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。

HSC（高速计数器）

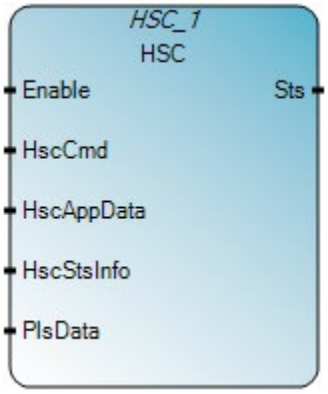
HSC 将高预设、低预设和输出源值应用到高速计数器。

操作详细信息：

- 通过将 HSCAppData.PLSEnable 参数设置为 True 启用可编程限位开关 (PLS)。
- 完整周期完成并达到 HSCSTS.HP 值后，PLSPosition 参数被复位 。

该指令适用于 Micro830、Micro850 和 Micro870 控制器。

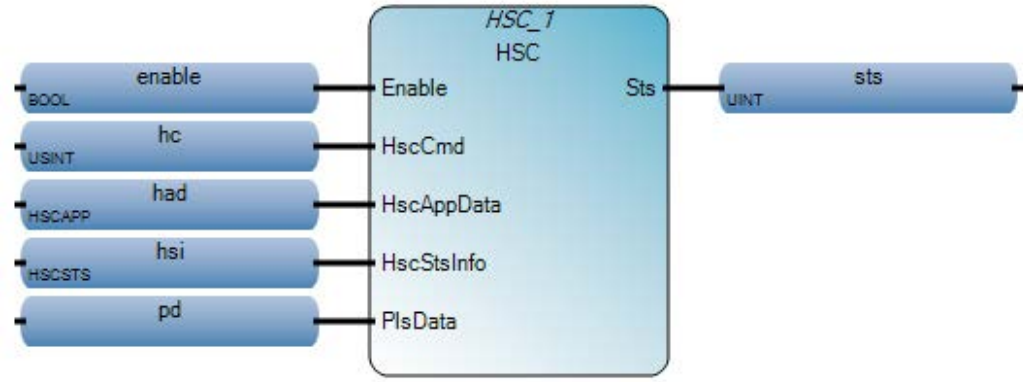
支持的语言：“功能块图”、“梯形图”和“结构化文本”。



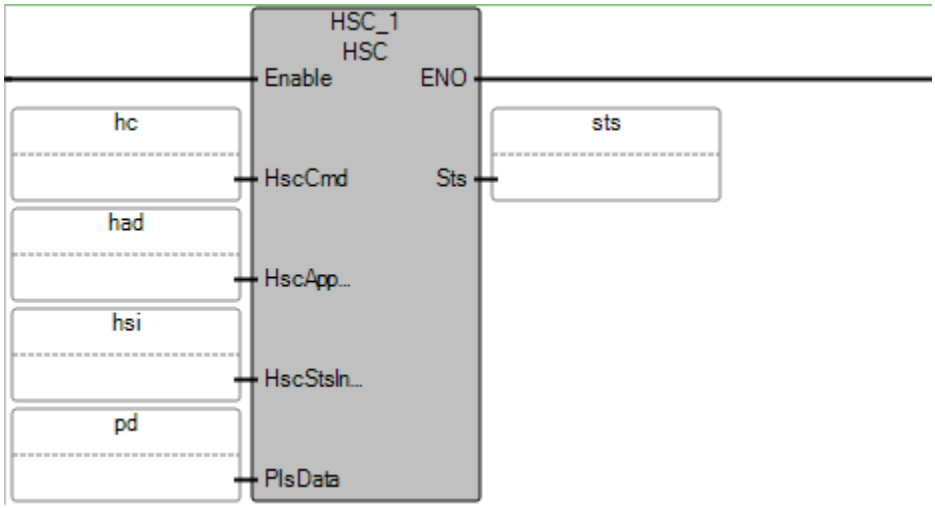
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	指令块梯级状态。 TRUE - 计时器开始递增。 FALSE - 功能块处于空闲状态。 我们不建议与 HSC 功能块一起使用 EN 参数，因为在将 EN 设置为 FALSE 时，计时器继续递增。 仅适用于梯形图编程。
Enable	输入	BOOL	启用指令块。 TRUE - 执行 HSC 命令参数中指定的 HSC 操作。 FALSE - 不发布任何 HSC 命令。
HscCmd	输入	USINT	向 HSC 发布命令。
HSCAppData	输入	HSCAPP 参考页数 272	HSC 应用程序配置（通常仅需一次）。
HSCStsInfo	输入	HSCSTS 参考页数 275	HSC 动态状态，在 HSC 计数期间不断更新。
PlsData	输入	PLS 参考页数 281	可编程限位开关 (PLS) 数据结构。
Sts	输出	UINT	HSC 执行状态。 HSC 状态代码： • 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。
ENO	输出	BOOL	启用“输出”。 仅适用于梯形图编程。

HSC 功能块图示例



HSC 梯形图示例



HSC 结构化文本示例



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）。

HSC 命令	命令描述
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

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。

HSCAPP 数据类型

使用 HSCAPP 数据类型定义 HSC 指令中的 HSCAppData 参数。HSCApp 数据类型参数用于定义 HSC 配置数据。

使用此表帮助确定 HSCAPP 数据类型的参数值。

参数	数据类型	数据格式	用户程序访问	描述
PLSEnable	BOOL	位	读取/写入	启用或禁用高速计数器可编程限位开关 (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。 当高速计数器启动时，HSCApp.Accumulator 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	DINT	长字	读取/写入	高预设设置。 HSCApp.HPSetting 参数设置用于定义 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	DINT	长字	读取/写入	低预设设置。 HSCApp.LPSetting 设置用于定义 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。

参数	数据类型	数据格式	用户程序访问	描述
OFSetting	DINT	长字	读取/写入	溢出设置。 HSCApp.OFSetting 上溢设置用于定义计数器的计数上限。 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	DINT	长字	读取/写入	下溢设置。 HSCApp.UFSetting 下溢设置用于定义计数器的计数下限。 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	UDINT	字	读取/写入	输出掩码。 HSCApp.OutputMask 用于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。 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	UDINT	长字	读取/写入	达到高预设 32 位输出设置。 HSCApp.HPOutput 用于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。 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	UDINT	长字	读取/写入	达到低预设 32 位输出设置。 HSCApp.LPOutput 用于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。 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。

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

HSCApp.HSCID

HSCApp.HSCID 参数用于标识高速计数器。

下表列出了 HSC 的值：

输出选择	位	描述
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。 对于初始版 Connected Components Workbench，仅支持 ID 0x00-0x05。

HSCApp.HSCMode

HSCApp.HSCMode 参数用于将高速计数器设置为 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。

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。

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）。
- 如果将主高速计数器设置为模式 1、3、5、7 或 9，则禁用子高速计数器。

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。

HSCMode	计数模式
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	X4 解码型计数器（带相位输入 A 和 B）。
9	具有外部重置和保存功能的 X4 解码型计数器（带相位输入 A 和 B）。

HSCAppData 参数示例

下图展示了**变量选择器**中的 HSCAppData 参数。



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

参数	数据类型	HSC 模式	用户程序访问	描述
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。

参数	数据类型	HSC 模式	用户程序访问	描述
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	读取/写入	达到低预设。
OFCauseInter	BOOL	0...9	读取/写入	上溢造成 HSC 中断。
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。完成完整周期并达到 HP 值后，PLSPosition 参数被复位。
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 ErrorDetected 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

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导致 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	用户程序访问
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。

错误代码子元素	HSC 计数错误代码	用户程序访问
	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。

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。

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。

名称	别名	数据类型	维度	项目值	初始值
HSC_1		HSC		...	...
HSC_1.Enable		BOOL			
HSC_1.HscCmd		USINT			
HSC_1.HscAppData		HSCAPP		...	...
HSC_1.HscStsInfo		HSCSTS		...	...
HSC_1.HscStsInfo.CountEnable		BOOL			
HSC_1.HscStsInfo.ErrorDetected		BOOL			
HSC_1.HscStsInfo.CountUpFlag		BOOL			
HSC_1.HscStsInfo.CountDwnFlag		BOOL			
HSC_1.HscStsInfo.Mode1Done		BOOL			
HSC_1.HscStsInfo.OVF		BOOL			
HSC_1.HscStsInfo.UNF		BOOL			
HSC_1.HscStsInfo.CountDir		BOOL			
HSC_1.HscStsInfo.HPReached		BOOL			
HSC_1.HscStsInfo.LPReached		BOOL			
HSC_1.HscStsInfo.OFCauseInter		BOOL			
HSC_1.HscStsInfo.UFCauseInter		BOOL			
HSC_1.HscStsInfo.HPCauseInter		BOOL			
HSC_1.HscStsInfo.LPCauseInter		BOOL			
HSC_1.HscStsInfo.PlsPosition		UINT			
HSC_1.HscStsInfo.ErrorCode		UINT			
HSC_1.HscStsInfo.Accumulator		DINT			
HSC_1.HscStsInfo.HP		DINT			
HSC_1.HscStsInfo.LP		DINT			
HSC_1.HscStsInfo.HPOutput		UDINT			
HSC_1.HscStsInfo.LPOutput		UDINT			

PLS 数据类型

PLSData（数据类型 PLS）用于配置可编程限位开关 参考页数 290。

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。

元素	数据类型	数据格式	HSC 模式	用户程序访问	描述
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。



HSCE_CHANNEL 数据类型

以下内容描述了 HSCE_CHANNEL 数据类型：

```
@typedef struct struct_HSCE_channel
{

USINT    ModuleType;
USINT    SlotID;
USINT    HSCID;
} HSCE_CHANNEL;
```

以下内容是描述信息：

字节	描述
模块类型	0x00：嵌入式 0x01：扩展（切片）

字节	描述
	0x02：通用端口
插槽 ID	0x00：嵌入式 0x01-0x1F：扩展（切片）模块的 ID 0x01-0x05：通用端口的 ID
HSCID	0x00-0x0F：嵌入式 0x00-0x07：用于扩展的 HSC 参考页数 269 的 ID 0x00-0x01：用于通用端口的 HSC 的 ID

HSCE_STS 数据类型

以下内容是 HSCE_STS 数据类型的描述信息：

参数	数据类型	HSC 模式	描述
CountEnable	BOOL	0...13	已启用计数。
ErrorDetected	BOOL	0...13	非零意味着检测到错误。
CountUpFlag	BOOL	0...13	向上计数标志。
CountDwnFlag	BOOL	2...13	向下计数标志。
Mode1Done	BOOL	0 或 1	HSC 是模式 1A 或模式 1B；累加器向上计数到 HP 值。
OVF	BOOL	0...13	检测到上溢。
UNF	BOOL	0...13	检测到下溢。
CountDir	BOOL	0...13	1：向上计数；0：向下计数。
HPReached	BOOL	2...13	达到高预设。
LPReached	BOOL	2...13	达到低预设。
OFCauseInter	BOOL	0...13	上溢造成 HSC 参考页数 269 中断。
UFCauseInter	BOOL	2...13	下溢造成 HSC 中断。
HPCauseInter	BOOL	0...13	达到高预设，从而造成 HSC 中断。
LPCauseInter	BOOL	2...13	达到低预设，从而造成 HSC 中断。
StateInfo	USINT		HSCE 计数器状态机信息
PlsPosition	UINT	0...9	可编程限位开关 (PLS) 的位置。完成完整周期并达到 HP 值后，PLSPosition 参数被复位。
ErrorCode	UINT	0...13	显示 HSC 子系统检测到的错误代码。
Accumulator	DINT		实际累加器读取。
HP	DINT		最终高预设设置。
LP	DINT		最终低预设设置。
HPOutput	UDINT		最终高预设输出设置。
LPOutput	UDINT		最终低预设输出设置。

PLS_HSCE 数据类型

“可编程限位开关”功能是高速计数器的一组额外操作模式。如果使用这些模式操作，则每次达到预设之一时，将使用用户提供的数据更新预设和输出数据值。通过提供包含设置为要使用的数据的 PLS 文件，可对这些模式进行编程。PLS_HSCE 数据结构是灵活的数组，其中每个元素的定义如下：

元素	数据类型	元素描述
HighPreset	LINT	高预设设置
LowPreset	LINT	低预设设置
HiPresetOutput	UDINT	高预设输出数据
LoPresetOutput	UDINT	低预设输出数据

```
// PLS 元素的结构
typedef struct PLS_HSCE_EleStruct
{
   LINT HighPreset;      // HSC 高预设值
   LINT LowPreset;       // HSC 低预设值
   UDINT HiPresetOutput; // HSC 高预设输出
   UDINT LoPresetOutput; // HSC 低预设输出
} PLS_HSCE_EleStruct;
```

对于 HSC 插件，一个 PLS_HSCE 数据的元素总数不应超过 24。

HSC_SET_STS（高速计数器设置状态）

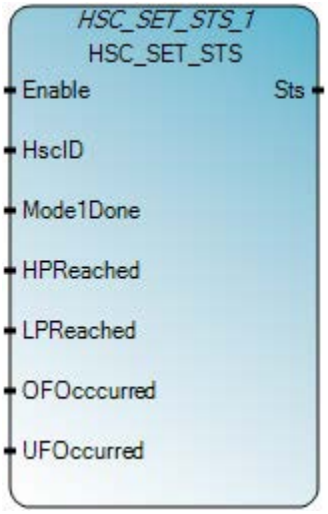
HSC_SET_STS 手动设置或重置 HSC 计数状态。

操作详细信息：

- 必须停止（不再计数）HSC 功能块，HSC_SET_STC 功能块才能设置或复位 HTS 状态。如果未停止 HSC 功能，则输入参数继续更新且忽略使用 HSC_SET_STS 所作的更改。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 设置/复位 HSC 状态。 FALSE - 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。 当 HSC 未计数时，可以设置或复位该位。
HPReached	输入	BOOL	达到高预设。 当 HSC 未计数时，可以设置或复位该位。
LPReached	输入	BOOL	达到低预设。 当 HSC 未计数时，可以设置或复位该位。
OFOccurred	输入	BOOL	发生溢出。 当 HSC 未计数时，可以设置或复位该位。

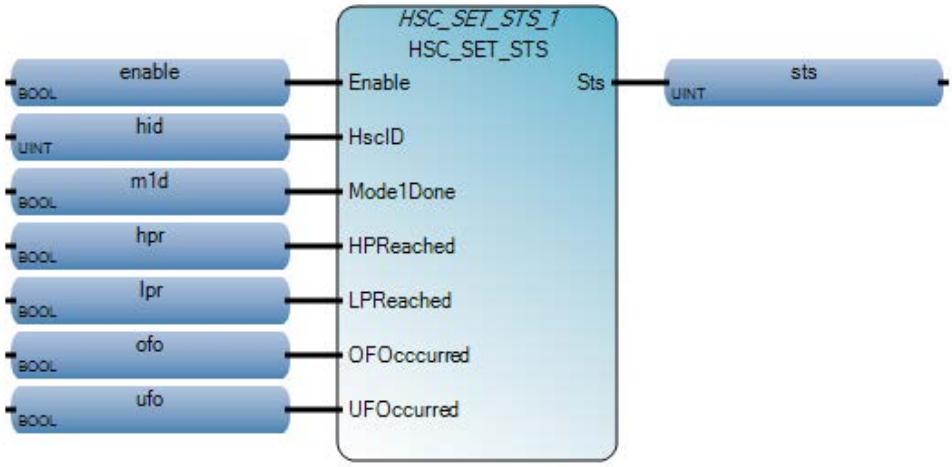
参数	参数类型	数据类型	描述
UFOccurred	输入	BOOL	发生下溢。 当 HSC 未计数时，可以设置或复位该位。
Sts	输出	UINT	状态代码在 HSC 状态代码 (Sts) 中定义。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

HSC 状态代码 (STS)

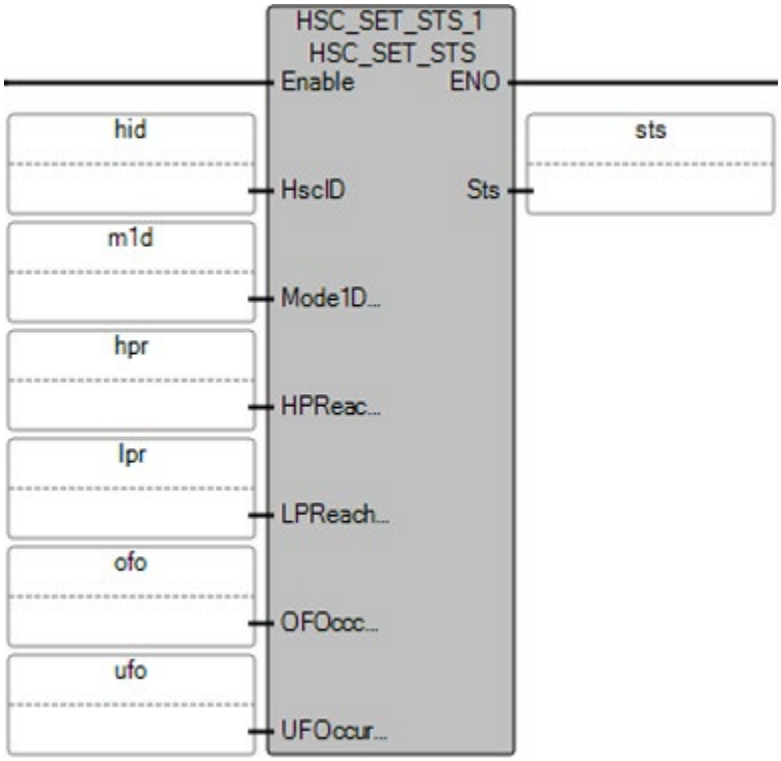
下表描述了 HSC 功能块的状态代码。

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。

HSC_SET_STS 功能块图示例



HSC_SET_STS 梯形图示例



HSC_SET_STS 结构化文本示例

```
HSC_SET_STS_1(
```

void **HSC_SET_STS_1**(BOOL **Enable**, UINT **HscID**, BOOL **Mode1Done**, BOOL **HPReached**, BOOL **LPReached**, BOOL **OFOcccurred**, BOOL **UFOccurred**)
类型： HSC_SET_STS, 手动设置/重置 HSC 状态。

```
1  HSC_SET_STS_1(enable, hid, m1d, hpr, lpr, ofo, ufo);
2  sts := HSC_SET_STS_1.Sts;
```

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 286

配置高速计数器 (HSC) 用户中断 参考页数 287

配置可编程限位开关 (PLS) 参考页数 290

示例：如何创建高速计数器 (HSC) 参考页数 291

示例：如何添加可编程限位开关 (PLS) 功能 参考页数 299

示例：可编程限位开关 (PLS) 已启用 参考页数 300

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 参考页数 269。

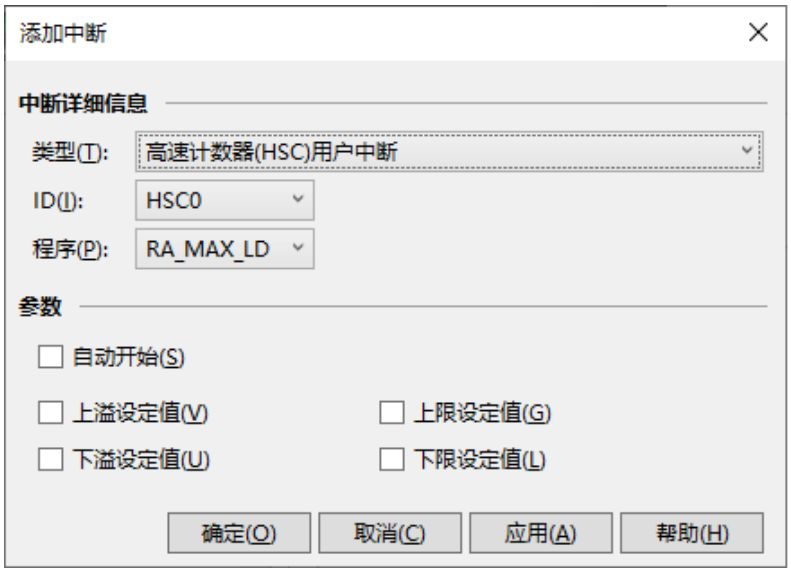
“高速计数器 (HSC) 用户中断”对话框

我如何才能打开“高速计数器 (HSC) 用户中断”对话框？

在“中断类型”中，选择**高速计数器 (HSC) 用户中断**。

HSC 中断对话框可用于执行以下操作：

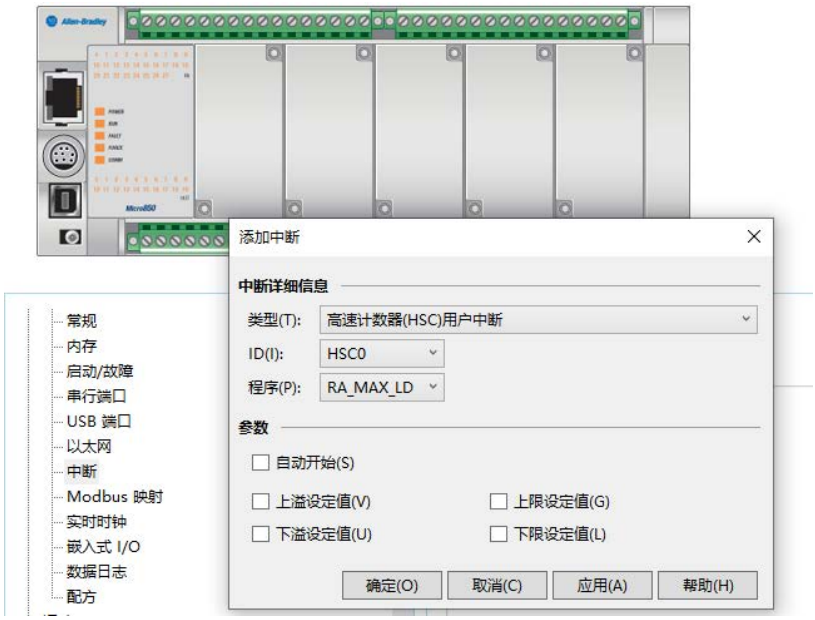
- 配置中断属性 参考页数 287，例如 ID 和使用该 ID 的程序。
- 配置中断参数。



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、Micro850 和 Micro870 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。



添加和配置高速计数器（HSC）用户中断

要从控制器的配置工作区添加和配置 HSC 中断 参考页数 287，请执行以下步骤。

要添加 HSC 中断：

1. 在“**项目组织管理器**”中，双击控制器以打开控制器工作区。
2. 在“控制器”树中，单击“中断”以显示“中断”配置页面。
3. 右键单击空行，然后单击“添加”以打开“中断”属性对话框。
4. 要配置 HSC 中断：
   - 在“中断类型”中，选择**高速计数器（HSC）用户中断**。
   - 选择“HSC 中断”属性 参考页数 288。
   - 选择“HSC 中断”参数 参考页数 289。
5. 关闭“中断”属性对话框。

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。

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)

“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，在断电重启过程中其值将保留。用户程序必须设置和清除“MV”位。

参数	数据格式	HSC 模式	用户程序访问
HSC0.MV	位	0...9	只读

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，在断电重启过程中其值将保留。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

“MH”（高预设掩码）控制位用于启用（允许）或禁止（不允许）发生高预设中断。如果该位已清除 (0)，且 HSC 检测到“达到高预设”条件，则不会执行 HSC 用户中断。

“MH”位由用户程序控制，在断电重启过程中其值将保留。用户程序必须设置和清除 MH 位。

低预设掩码（HSC0.ML）

参数	数据格式	HSC 模式	用户程序访问
HSC0.ML	位	2...9	只读

“ML”（低预设掩码）控制位用于启用（允许）或禁止（不允许）发生低预设中断。如果该位已清除 (0)，且 HSC 检测到“达到低预设”条件，则不会执行 HSC 用户中断。

“ML”位由用户程序控制，在断电重启过程中其值将保留。用户程序必须设置和清除“ML”位。

配置可编程限位开关（PLS）

高速计数器 参考页数 269 具有用于实施可编程限位开关 (Programmable Limit Switch, PLS) 的附加操作模式。PLS 功能用于将高速计数器配置为用作 PLS 或旋转凸轮开关。PLS 功能最多支持 255 对高预设和低预设，可供您在需要多对高预设和低预设时使用。

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 指令中启用，方法是将 HSCAppData.PLSEnable 参数设置为 True。

完成完整周期并达到 HSCSTS.HP 值后，PLSPosition 参数被复位。复位 HSC 指令或将 0 移动到 PLSPositon 参数的操作不会复位 PLSPosition 。

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 参考页数 300 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。

嵌入式 HSC 和插件 HSC 模块之间的 PLS 预设差异

嵌入式 HSC 和插件 HSC 模块之间的 PLS HSCHP 和 HSCLP 预设行为不同。仅当执行最后一个 PLS 后设置嵌入式 HSC 高预设位，在执行第一个 PLS 后设置插件 HSC 模块高预设位。例如，

- 嵌入式 HSC PLS0-PLS23：

  当 PLS23HP=累加器值时，将设置高预设
  当 PLS23LP=累加器值时，将设置低预设

- 插件 HSC PLS0-PLS23：

  当 PLS0HP=累加器值时，将设置高预设
  当 PLS0LP=累加器值时，将设置低预设

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 (Programmable Limit Switch, PLS) 功能的高速计数器 (High-Speed Counter, HSC) 参考页数 269程序。

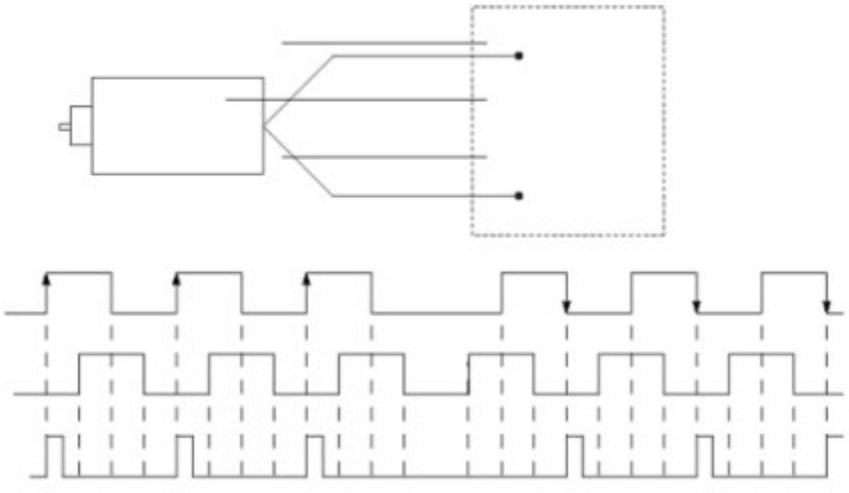
示例中使用的正交编码器

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。

步骤	任务
1	创建梯形图并添加变量 参考页数 292
2	将值分配给 HSC 变量 参考页数 295
3	分配变量并生成程序 参考页数 296
4	测试程序并运行高速计数器 参考页数 297
5	添加可编程限位开关 (PLS) 功能 参考页数 299

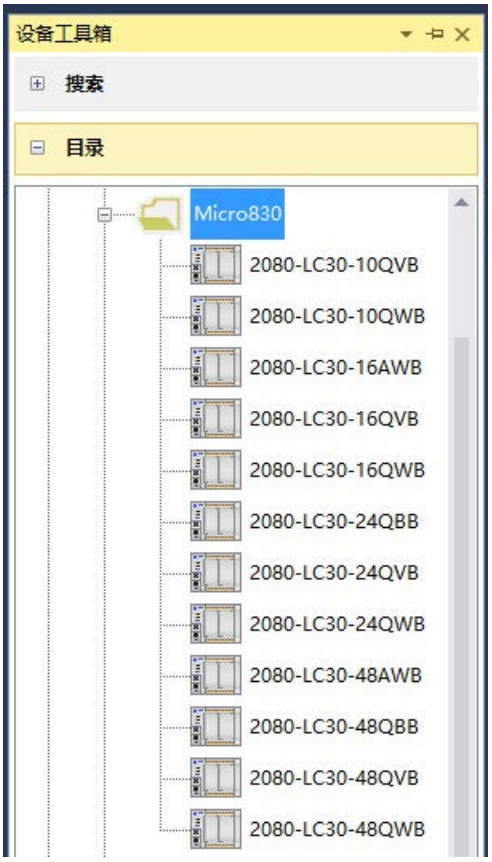
创建梯形图并添加变量

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 参考页数 269。

要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**项目管理器**。



3. 在**项目管理器**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”对话框中。
   - 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。

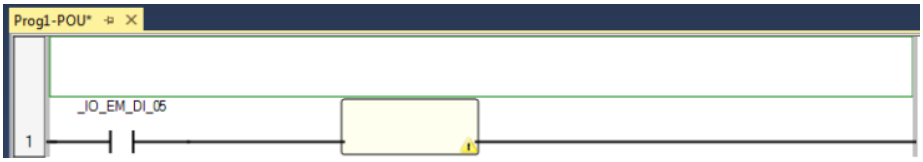


6. 将变量分配给正向触点：
   - 双击正向触点以显示**变量选择器**，然后单击 **I/O - Micro830** 选项卡。

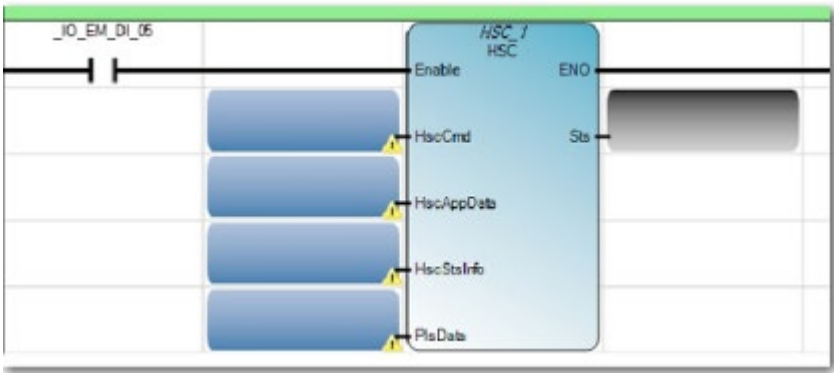
- 单击 **_IO_EM_DI_05**，然后单击**确定**，以将正向触点分配给输入 5。



7. 在“工具箱”对话框中，选择功能块，并将其拖至正向触点右侧，如下图中所示。



8. 双击功能块以打开“指令块选择器”。
9. 在“指令块选择器”中，选择“HSC”并单击“确定”。
10. 验证梯形梯级是否与下图相似。



11. 在**项目管理器**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”页面应如下图所示：



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。

要将值分配给 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 272 中的 HSCMode。
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。
   - 有关每个值说明的详细信息，请参见 HSCAPP 数据 参考页数 272 类型。

名称	别名	数据类型	维度	项目值	初始值
MyAppData		HSCAPP		...	...
MyAppData.PlsEnable		BOOL			FALSE
MyAppData.HscID		UINT			0
MyAppData.HscMode		UINT			6
MyAppData.Accumulator		DINT			
MyAppData.HPSetting		DINT			40
MyAppData.LPSetting		DINT			-40
MyAppData.OFSetting		DINT			50
MyAppData.UFSetting		DINT			-50
MyAppData.OutputMask		UDINT			3
MyAppData.HPOutput		UDINT			1
MyAppData.LPOutput		UDINT			2

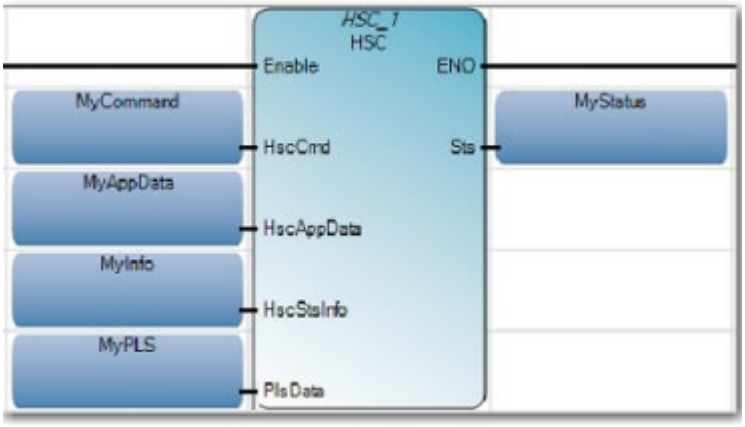
4. 分配 HSC 命令值：
   - 在 MyCommand 变量的“初始值”字段中，键入 1。
   - 有关命令值的更多信息，请参见 HSCCmd 值 参考页数 271。

分配变量并生成程序

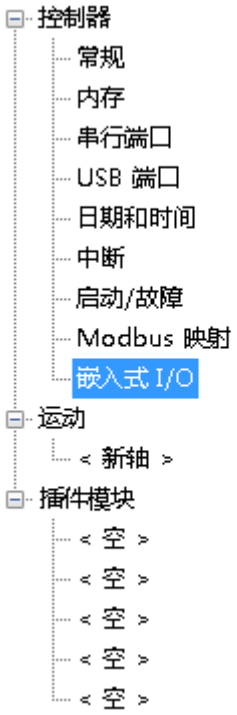
在 HSC 变量中输入值后，请按照以下步骤，将变量分配给功能块，然后生成程序。

要分配变量并生成程序：

1. 在梯形图编辑器中，将每个变量分配给 HSC 功能块元素，如下所示。



2. 在“项目管理器”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筛选器。



4. 验证编码器是否与 Micro830 控制器相连。
5. 启动 Micro830 控制器，并将该控制器连接到计算机。
6. 生成程序，然后将程序下载到控制器。

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。

要测试程序：

1. 连接到控制器。
2. 在**项目管理器**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。

名称	别名	逻辑值	实际值	初始值	锁定	数据类型	维度
_IO_EM_DO_00						BOOL	
_IO_EM_DO_01						BOOL	
_IO_EM_DO_02						BOOL	
_IO_EM_DO_03						BOOL	
_IO_EM_DO_04						BOOL	
_IO_EM_DO_05						BOOL	
_IO_EM_DO_06						BOOL	
_IO_EM_DI_00						BOOL	
_IO_EM_DI_01						BOOL	
_IO_EM_DI_02						BOOL	
_IO_EM_DI_03						BOOL	
_IO_EM_DI_04						BOOL	
_IO_EM_DI_05						BOOL	
_IO_EM_DI_06						BOOL	
_IO_EM_DI_07						BOOL	
_IO_EM_DI_08						BOOL	
_IO_EM_DI_09						BOOL	

6. 要查看结果，单击“局部变量”选项卡以查看变量更改。
7. 展开 **MyAppData** 和 **MyInfo** 变量列表。
8. 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。
9. 验证 MyStatus 变量的逻辑值是否为 1，这表示 HSC 正在运行。
10. 查看 MyInfo.Accumulator 中的计数器值。

    有关状态代码的完整列表，请参见 HSC（高速计数器，High-Speed Counter） 参考页数 269。

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。如果 MyInfo.Accumulator 达到低预设值 -40，则输出 1 将开启，并且 LPReached 标志开启。

变量监视

用户全局变量 - Micro870 | 局部变量 - UntitledLD | 系统

名称	逻辑值	实际值
HSC_1	...	...
MyAppData	...	...
MyAppData.PlsEnabl	☐	不可用
MyAppData.HscID	0	不可用
MyAppData.HscMode	7	不可用
MyAppData.Accumula	40	不可用
MyAppData.HPSettin	40	不可用
MyAppData.LPSettin	-40	不可用
MyAppData.OFSettin	50	不可用
MyAppData.UFSettin	-50	不可用
MyAppData.OutputMa	3	不可用
MyAppData.HPOutput	1	不可用
MyAppData.LPOutput	2	不可用
MyCommand	0	不可用
MyInfo	...	...
MyInfo.CountEnable	☑	不可用
MyInfo.ErrorDetect	☐	不可用
MyInfo.CountUpFlag	☑	不可用
MyInfo.CountDownFl	☑	不可用
MyInfo.Mode1Done	☐	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.OFCauseInte	☐	不可用
MyInfo.UFCauseInte	☐	不可用
MyInfo.HPCauseInte	☐	不可用
MyInfo.LPCauseInte	☐	不可用
MyInfo.StateInfo	0	不可用
MyInfo.PlsPosition	0	不可用

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：

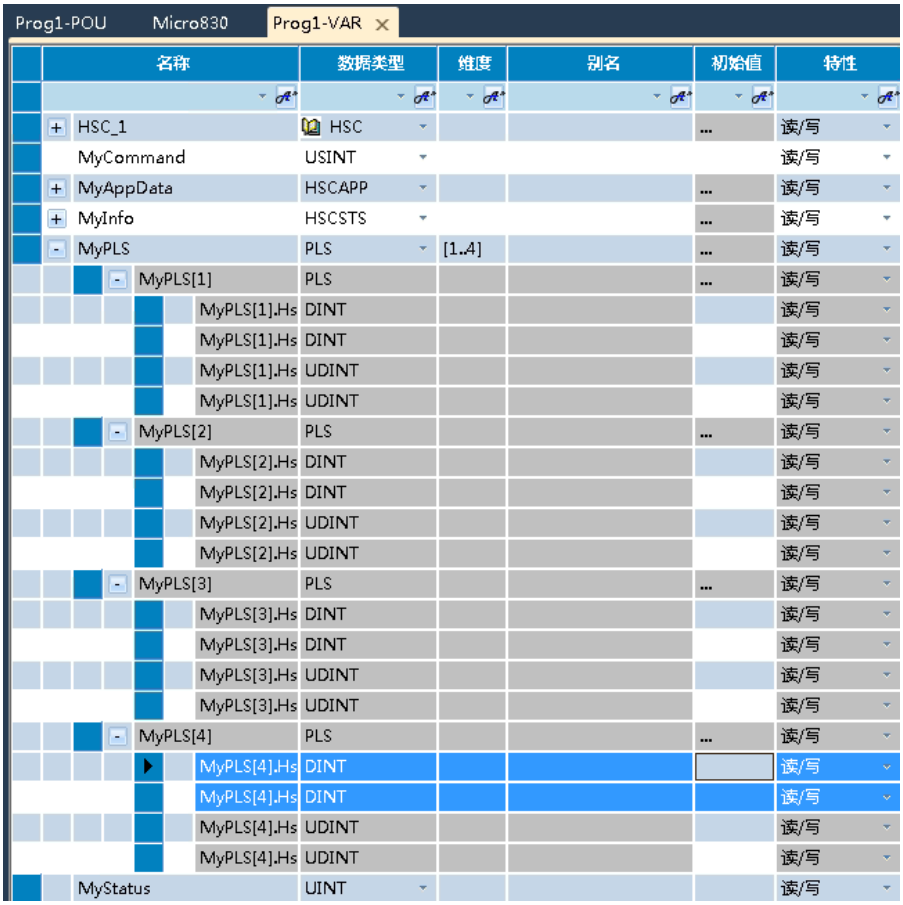
- **MyAppData.PlsEnable** 用于启用或禁用 PLS 设置。如果使用了 MyAppData 变量，则应将它设置为 FALSE（已禁用）。
- **MyAppData.HscID** 用于指定基于模式和应用程序类型应使用哪些嵌入式输入。请参见 HSC 输入和配线图以了解可使用的各种 ID 以及嵌入式输入和其特性。
- 如果使用 ID 0，则不能在同一个控制器上使用 ID 1，因为输入已被“重置”和“保存”使用。

- “MyAppData.HscMode”用于指定 HSC 用于计数的操作类型。请参见 HSC 模式 (HSCAPP.HSCMode) 参考页数 272。

要启用 PLS：

1. 在“项目管理器”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. （可选）如果输出，则配置输出掩码。

   本示例的结果：
   - 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。
   - 高预设应低于 OFSetting，低预设应高于 UFSetting。
   - HscHPOutPut 值和 HscLPOutPut 值确定在达到高预设和低预设时会开启的输出。



示例：可编程限位开关 (PLS) 已启用

本示例介绍了使用特定 HSC 参考页数 269 和 PLSData 参数值启用 PLS 后的结果。

HSC 参数值

本示例假定将 HSC 参数设置为下列值：

- HSCApp.OutputMask = 31
- HSCApp.HSCMode = 0
- HSC 仅控制嵌入式输出 0...4

PLSData 参数值

本示例假定变量 (HSC_PLS) 的 PLSData 参数如下所示配置。

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

在本示例中，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。
- 在操作完成后，该周期将复位并从 HSCSTS.HP = 250 开始重复。
- 完整循环完成并达到 HSCSTS.HP 值时，PLSPositon 参数被复位。

HSCE 指令

使用 HSCE 指令监视和控制高速计数器。

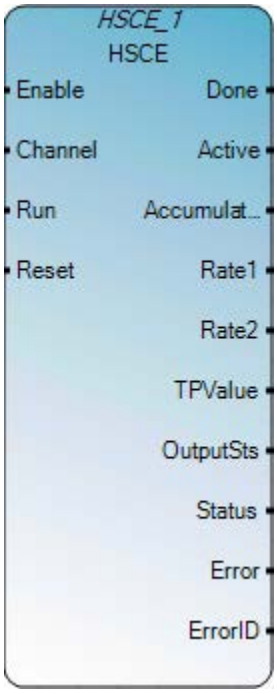
指令	描述
HSCE 参考页数 303	HSCE 开始、停止和读取累加值。
HSCE_CFG 参考页数 306	HSCE_CFG 是高速计数器配置。
HSCE_CFG_PLS 参考页数 308	HSCE_CFG_PLS 是高速计数器 PLS 配置。
HSCE_READ_STS 参考页数 310	HSCE_READ_STS 读取高速计数器状态。
HSCE_SET_STS 参考页数 303	HSCE_SET_STS 手动设置/复位高速计数器状态。

HSCE

HSCE 用于控制和读取 HSC 计数器。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。

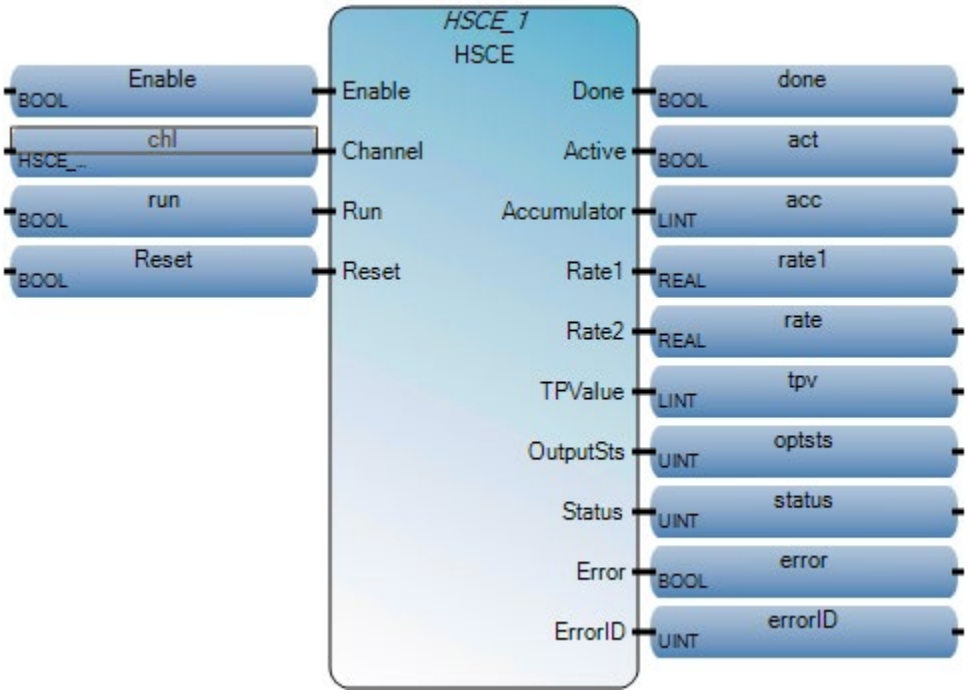


使用此表有助于确定该指令的参数值。

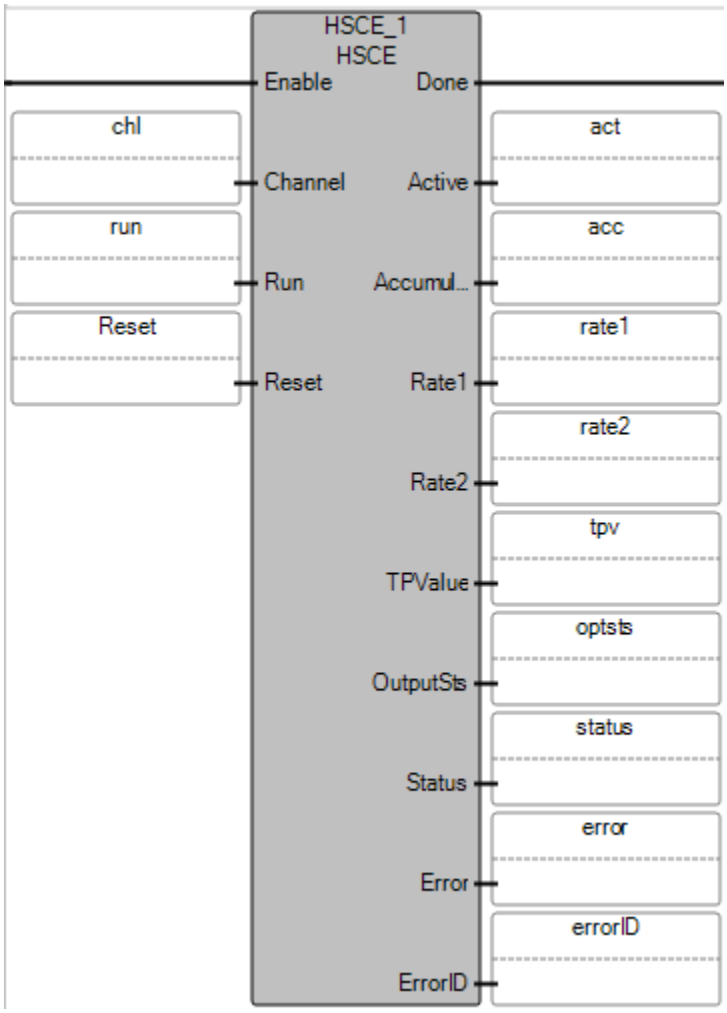
参数	参数类型	数据类型	描述
Enable	输入	BOOL	TRUE - HSCE 启动功能块和插件 HSC 模块。累加器 = InitAcc。将复位 HSCE 的状态。速率 1 = 0，速率 2 = 0。 FALSE - 通过从插件模块进行读取来更新累加器。完成、活动、错误 = FALSE，而 ErrorID = 0，速率 1、速率 2 = 0。
Channel	输入	HSCE_CHANNEL	HSCE 通道。

参数	参数类型	数据类型	描述
Run	输入	BOOL	对于 HSCE，对操作状态进行计数。 TRUE - HSCE 对脉冲进行计数。 FALSE - HSCE 停止计数。
Reset	输入	BOOL	True - 清除所有输出和累加器。也将速率 1 和速率 2 清除为 0。清除插件 HSC 模块状态。复位输入的优先级高于 Run 输入的优先级
Done	输出	BOOL	True - 当 HSCE Enable 为 True 且没有检测到错误时 False - 当 HSCE Enable 为 True 但 Run 为 False 时。
Active	输出	BOOL	True - 当 HSCE Enable 为 True，Done 为 True 且 Run 为 False 时。 False - 当 HSCE Enable 为 True 但 Run 为 False 时。
Rate1	输出	REAL	采用用户单位的每秒电流脉冲率（每脉冲方法）。
Rate2	输出	REAL	采用用户单位的每秒电流脉冲率（循环方法）。
TPValue	输出	LINT	触发触摸探针后捕获累加器值。
OutputSts	输出	UINT	HSC 插件物理和虚拟输出状态，且仅适用于 HSC 计数器 0。 位 0：输出 0（物理输出状态） 位 1 到 15：输出 1 到输出 15（虚拟输出状态）
Status	输出	UINT	HSC 状态信息。
Error	输出	BOOL	指示发生了错误。
ErrorID	输出	UINT	当发生错误时，ErrorID 包含错误代码 参考页数 313。

HSCE 功能块图示例



HSCE 梯形图示例



HSCE 文本结构示例

```
HSCE_1(
```

void **HSCE_1**(BOOL **Enable**, HSCE_CHANNEL **Channel**, BOOL **Run**, BOOL **Reset**)
类型： HSCE, 开始、停止和读取累加器值

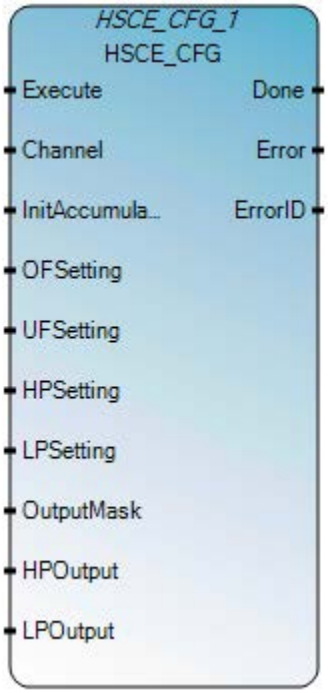
```
1  HSCE_1(Enable,chl,Run,Reset);
2  Done_HSCE := HSCE_1.Done;
3  Active_HSCE := HSCE_1.Active;
4  Accumulator_HSCE := HSCE_1.Accumulator;
5  Rate1_HSCE := HSCE_1.Rate1;
6  Rate2_HSCE := HSCE_1.Rate2;
7  TPValue_HSCE := HSCE_1.TPValue;
8  Outputsts_HSCE := HSCE_1.OutputSts;
9  Status_HSCE := HSCE_1.Status;
10 Error_HSCE := HSCE_1.Error;
11 ErrorID_HSCE := HSCE_1.ErrorID;
```

HSCE_CFG

HSCE_CFG 用于配置高速计数器。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

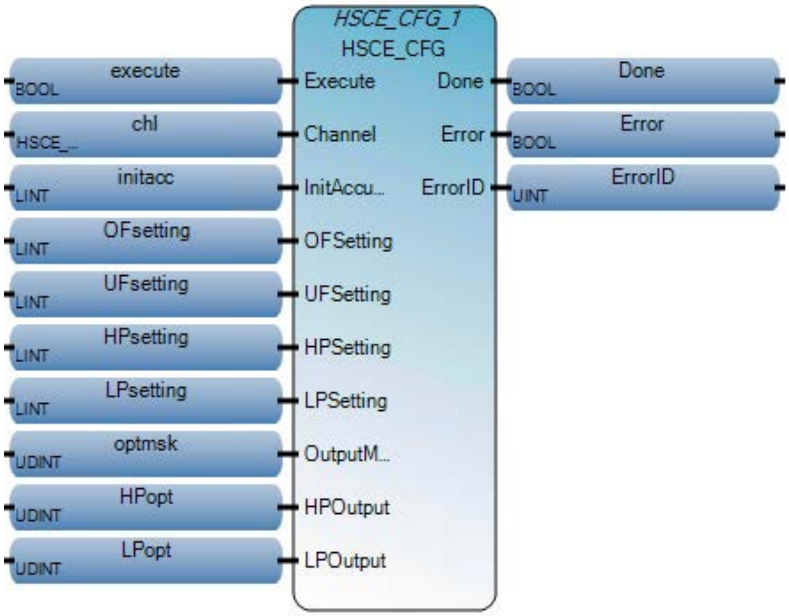
该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



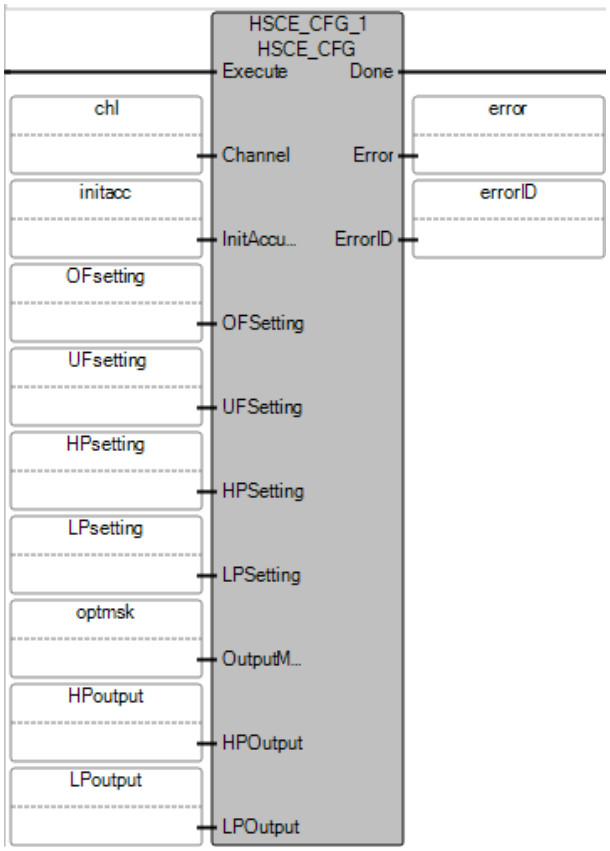
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Execute	输入	BOOL	上升沿启动 HSC 配置。（HSCE Enable 应等于 FALSE） 下降沿将清除所有输出值。
Channel	输入	HSCE_CHANNEL	HSCE 通道。
InitAccumulator	输入	LINT	累加器初始值。
OFSetting	输入	LINT	计数器溢出限制值。
UFSetting	输入	LINT	计数器下溢限制值。
HPSetting	输入	LINT	HSCE 的高预设 (HP) 值。
LPSetting	输入	LINT	HSCE 的低预设 (LP) 值。
PLS_Offset	输入	USINT	在 PLS 数据数组中开始偏移。
OutputMask	输入	UDINT	PLS 功能的输出掩码。
HPOutput	输入	UDINT	高预设输出状态。
LPOutput	输入	UDINT	低预设输出状态。
Done	输出	BOOL	HSC 配置操作（由此指令启动）成功。
Error	输出	BOOL	指示发生了错误。
ErrorID	输出	UINT	当发生错误时，ErrorID 包含错误代码 参考页数 313。

HSCE_CGF 功能块图示例



HSCE_CGF 梯形图示例



HSCE_CGF 文本结构示例

```
HSCE_CFG_1(
void HSCE_CFG_1(BOOL Execute, HSCE_CHANNEL Channel, LINT InitAccumulator, LINT OFSetting, LINT UFSetting, LINT HPSetting, LINT LPSetting, UDINT OutputMask, UDINT HPOutput, UDINT LPOutput)
类型: HSCE_CFG, 高速计数器配置

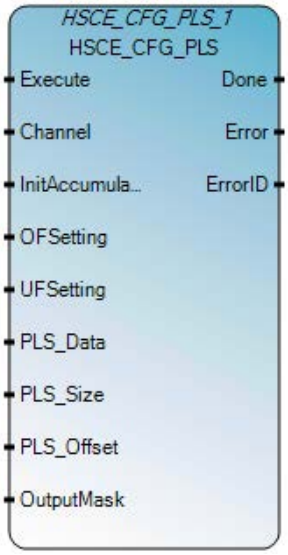
1  HSCE_CFG_1(execute,chl,initacc,OFsetting,UFsetting,HPsetting,LPsetting,optmsk,HPopt,LPopt);
2  Done_HSCE_CFG :=HSCE_CFG_1.Done;
3  Error_HSCE_CFG :=HSCE_CFG_1.Error;
4  ErrorID_HSCE_CFG := HSCE_CFG_1.ErrorID;
```

HSCE_CFG_PLS

该指令用于带可编程限位开关 (PLS) 的 HSC 配置。该功能是高速计数器的一组额外操作模式。如果使用这些模式操作，则每次达到预设之一时，将使用用户提供的数据更新预设和输出数据值。 通过提供包含设置为要使用的数据的 PLS 文件，可对这些模式进行编程。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

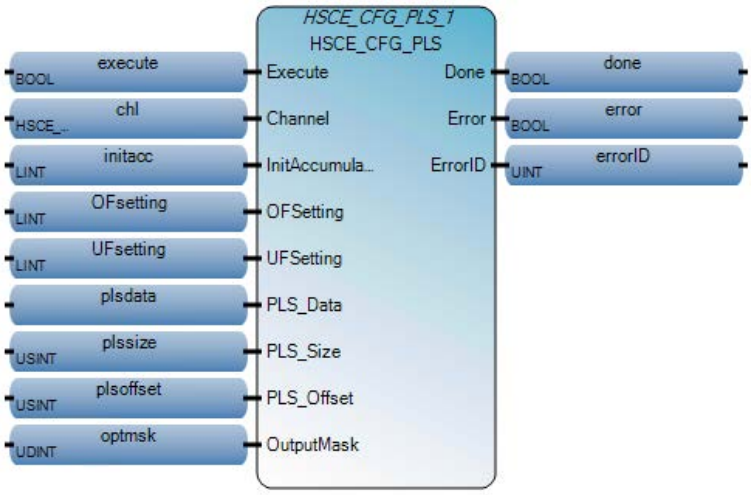
该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



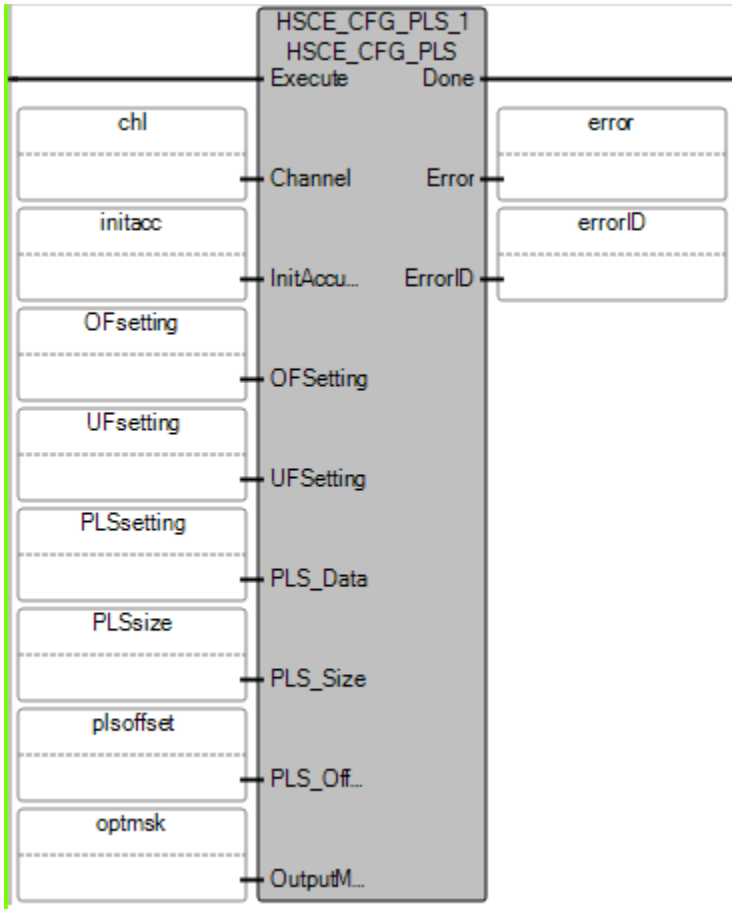
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Execute	输入	BOOL	上升沿启动 HSC 配置。（HSCE Enable 应等于 FALSE） 下降沿将清除所有输出值。
Channel	输入	HSCE_CHANNEL	HSCE 通道。
InitAccumulator	输入	LINT	累加器初始值。
OFSetting	输入	LINT	计数器溢出限制值。
UFSetting	输入	LINT	计数器下溢限制值。
PLS_Data	输入	PLS2	高预设 (LINT)、低预设 (LINT)、高预设输出 (UDINT) 和低预设输出 (UDINT) 的配置值。
PLS_Size	输入	USINT	PLS 数据大小，插件的最大值为 24。
PLS_Offset	输入	USINT	在 PLS 数据数组中开始偏移。
OutputMask	输入	UDINT	PLS 功能的输出掩码。
Done	输出	BOOL	HSC 配置操作（由此指令启动）成功。
Error	输出	BOOL	指示发生了错误。
ErrorID	输出	UINT	当发生错误时，ErrorID 包含错误代码 参考页数 313。

HSCE_CGF_PLS 功能块图示例



HSCE_CGF_PLS 梯形图示例



HSCE_CGF_PLS 文本结构示例

```
HSCE_CFG_PLS_1(
void HSCE_CFG_PLS_1(BOOL Execute, HSCE_CHANNEL Channel, LINT InitAccumulator, LINT OFSetting, LINT UFSetting, PLS_HSCE[1..1] PLS_Data, USINT PLS_Size, USINT PLS_Offset, UDINT OutputMask)
类型： HSCE_CFG_PLS, 高速计数器 PLS 配置

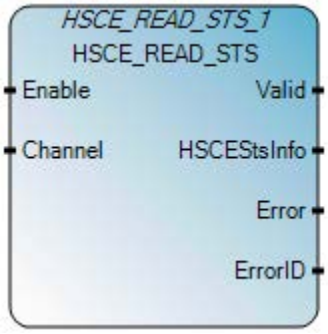
1 HSCE_CFG_PLS_1(Execute,chl,initacc,OFsetting,UFsetting,PLS_data,PLS_Size,PLS_Offset,optmsk);
2 Done_HSCE_CFG_PLS :=HSCE_CFG_PLS_1.Done;
3 Error_HSCE_CFG_PLS :=HSCE_CFG_PLS_1.Error;
4 ErrorID_HSCE_CFG_PLS := HSCE_CFG_PLS_1.ErrorID;
```

HSCE_READ_STS

该指令用于读取当前 HSC 状态。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

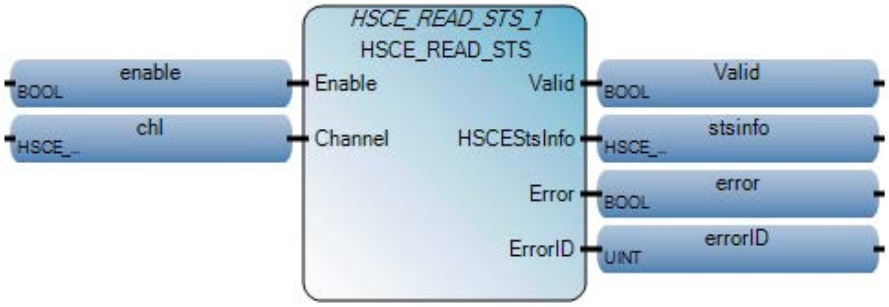
该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



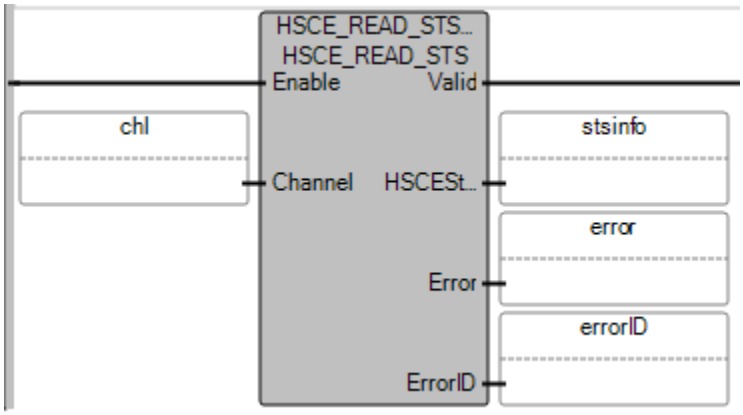
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	如果 Enable 为 True，则更新 HSC2StsInfo。
Channel	输入	HSCE_CHANNEL	HSCE 通道。
Valid	输出	BOOL	如果为 TRUE，则 HSC2StsInfo 有效。
HSCEStsInfo	输入	HSCE_STS	计数器溢出限制值。
Error	输出	BOOL	指示发生了错误。
ErrorID	输出	UINT	当发生错误时，ErrorID 包含错误代码 参考页数 313。

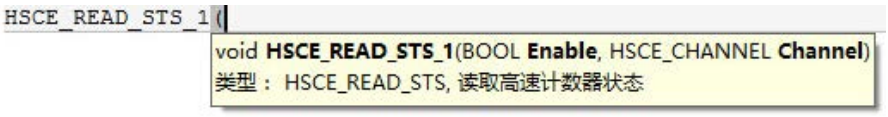
HSCE_READ_STS 功能块图示例



HSCE_READ_STS 梯形图示例



HSCE_READ_STS 文本结构示例



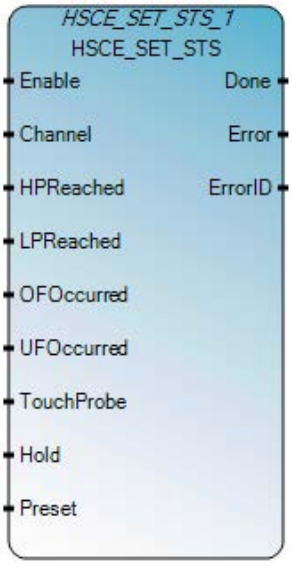

```
1  HSCE_READ_STS_1(Enable,Channel);
2  Valid_HSCE_READ_STS :=HSCE_READ_STS_1.Valid;
3  HSCEstsInfo_HSCE_READ_STS := HSCE_READ_STS_1.HSCEStsInfo;
4  Error_HSCE_READ_STS := HSCE_READ_STS_1.Error;
5  ErrorID HSCE READ STS := HSCE READ STS 1.ErrorID;
```

HSCE_SET_STS

该指令手动设置或复位 HSC 计数状态标记。必须停止 HSC 功能块，HSCE_SET_STC 功能块才能设置或复位其 HTS 状态。如果未停止 HSC 功能块或未配置 HSC 通道，则 HSC2_SET_STS 功能块将发生错误。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

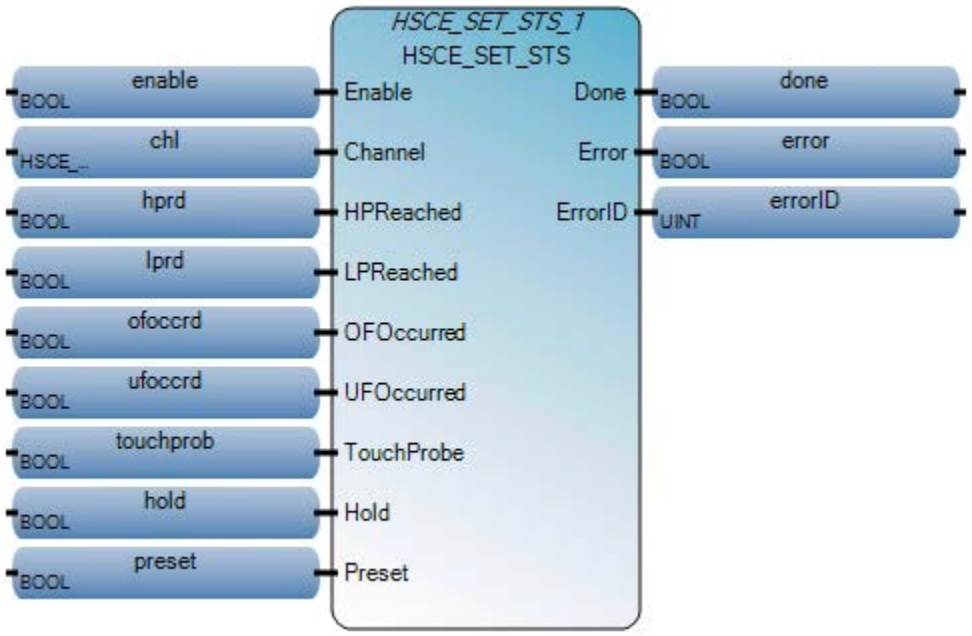
该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



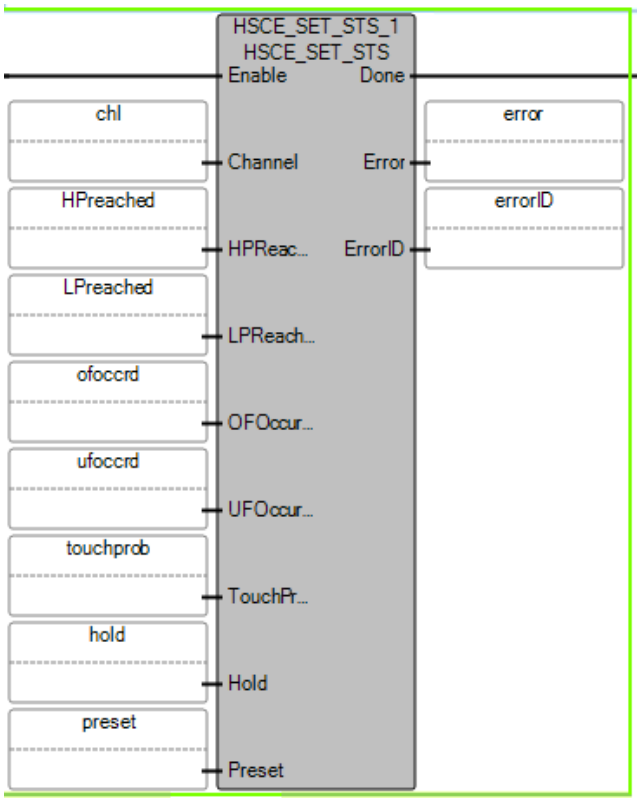
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	功能块启用。 TRUE - 设置/复位 HSC 状态。 FALSE - HSC 状态不变。
Channel	输入	HSCE_CHANNEL	HSCE 通道。
HPReached	输入	BOOL	当达到高预设值时，会通过插件模块将此位设置为 TRUE。通过 HSCE_SET_STS 功能块设置或复位此位。
LPReached	输入	BOOL	当达到低预设值时，会通过插件模块将此位设置为 TRUE。通过 HSCE_SET_STS 功能块设置或复位此位。
OFOccurred	输入	BOOL	当发生溢出时，会通过插件模块将此位设置为 TRUE。通过 HSCE_SET_STS 功能块设置或复位此位。
UFOccurred	输入	BOOL	当发生下溢时，会通过插件模块将此位设置为 TRUE。通过 HSCE_SET_STS 功能块设置或复位此位。
TouchProbe	输入	BOOL	当触发触摸探针时，会通过插件模块将此位设置为 TRUE。
Hold	输入	BOOL	当触发 HSC 保持时，会通过插件模块将此位设置为 TRUE。通过 HSCE_SET_STS 功能块设置或复位此位。该输入参数仅适用于计数器 0。对于计数器 1，始终复位
Preset	输入	BOOL	当触发 Z ACC 复位时，会通过插件模块将此位设置为 TRUE。 通过 HSCE_SET_STS 功能块设置或复位此位。 该输入参数仅适用于计数器 0。对于计数器 1，始终复位。
Done	输出	BOOL	HSC 配置操作（由此指令启动）成功。
Error	输出	BOOL	指示发生了错误。
ErrorID	输出	UINT	当发生错误时，ErrorID 包含错误代码 参考页数 313。

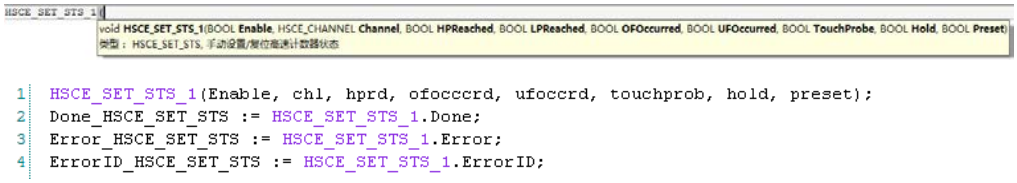
HSCE_SET_STS 功能块图示例



HSCE_SET_STS 梯形图示例



HSCE_SET_STS 文本结构示例



HSCE 错误代码

下表描述了 HSCE 指令 参考页数 303 的状态错误代码：

ErrorID 代码	错误描述	更正措施
0	指令已成功完成操作。	
1	HSC 配置文件无效。	与 Rockwell Automation 技术支持代表联系。有关联系信息，请参见：http://www.rockwellautomation.com/support
2	HSC 模块类型无效。	更正模块类型。例如，选择模块类型作为插件。
3	HSC 插槽 ID 无效。	更正功能块的通道输入中的插槽 ID。
4	HSC ID 无效。	更正 HSC ID。例如，为计数器 0 设置 0。
5	通道的模式无效。	与 Rockwell Automation 技术支持代表联系。
6	PLS 大小无效。	PLS 大小应 ≤24，PLS 数据数组的大小 ≥PLS 大小。
7	PLS 偏移量无效。	PLS 偏移量 + PLS 大小应在 PLS 数据数组的大小范围内。

ErrorID 代码	错误描述	更正措施
8	InitAccumulator 值无效。	更正 InitAccumulator 值。它可能超过边界值或超出 LP 或 HP 限值范围 (LP ≤ InitACC ≤ HP)。
9	LP 无效。	更正 LP 值。它可能超过边界值。
10	OF 无效。	更正 OF 值。
11	UF 无效。	更正 UF 值。
12	HP 无效。	更正 HP 值。
13	没有 HSCE 的配置。	与 Rockwell Automation 技术支持代表联系。
14	HSCE 状态无效。	检查 HSCE 相关的功能块以确认此功能块的状态。
15	插件模块无效	检查插件模块以确认它是 HSC 模块。
16	HSCE 正在运行。	当 HSC 正在计数时，不允许配置和设置状态。在 HSCE 运行时执行 HSCE_CFG 参考页数 306 和 HSCE_CFG_PLS 参考页数 308。
17	停止 HSC 插件失败。	与 Rockwell Automation 技术支持代表联系。
18	更新速率写入失败。	与 Rockwell Automation 技术支持代表联系。
19	写入脉冲编号失败。	与 Rockwell Automation 技术支持代表联系。
20	写入应用 ACC 失败。	与 Rockwell Automation 技术支持代表联系。
21	写入 ACC 失败。	与 Rockwell Automation 技术支持代表联系。
22	写入应用失败。	与 Rockwell Automation 技术支持代表联系。
23	写入 PLS 编号失败。	与 Rockwell Automation 技术支持代表联系。
24	写入下溢失败。	与 Rockwell Automation 技术支持代表联系。
25	写入上溢失败。	与 Rockwell Automation 技术支持代表联系。
26	写入低预设失败。	与 Rockwell Automation 技术支持代表联系。
27	写入高预设失败。	与 Rockwell Automation 技术支持代表联系。
28	写出低预设输出失败。	与 Rockwell Automation 技术支持代表联系。
29	写出高预设输出失败。	与 Rockwell Automation 技术支持代表联系。
30	写出输出掩码失败。	与 Rockwell Automation 技术支持代表联系。
31	写出 PLS 低预设输出失败。	与 Rockwell Automation 技术支持代表联系。
32	写出 PLS 高预设输出失败。	与 Rockwell Automation 技术支持代表联系。
33	写入 PLS 低预设失败。	与 Rockwell Automation 技术支持代表联系。
34	写出 PLS 高预设输出失败。	与 Rockwell Automation 技术支持代表联系。
35	写入 PLS 偏移量失败。	与 Rockwell Automation 技术支持代表联系。
36	写入 PLS 编号失败。	与 Rockwell Automation 技术支持代表联系。
37	读取状态失败。	与 Rockwell Automation 技术支持代表联系。
38	读取高预设失败。	与 Rockwell Automation 技术支持代表联系。
39	读取低预设失败。	与 Rockwell Automation 技术支持代表联系。
40	读取 ACC 失败。	与 Rockwell Automation 技术支持代表联系。
41	读取脉冲宽度编号失败。	与 Rockwell Automation 技术支持代表联系。
42	读取脉冲宽度失败。	与 Rockwell Automation 技术支持代表联系。
43	读取脉冲编号失败。	与 Rockwell Automation 技术支持代表联系。
44	读取更新速率失败。	与 Rockwell Automation 技术支持代表联系。
45	写入状态失败。	与 Rockwell Automation 技术支持代表联系。
46	读取低预设输出失败。	与 Rockwell Automation 技术支持代表联系。
47	读取高预设输出失败。	与 Rockwell Automation 技术支持代表联系。
48	读取 PLS 编号失败。	与 Rockwell Automation 技术支持代表联系。
49	不适用	与 Rockwell Automation 技术支持代表联系。
50	写入应用状态失败。	与 Rockwell Automation 技术支持代表联系。
51	通道输入无效。	与 Rockwell Automation 技术支持代表联系。
52	读取触摸探针失败。	与 Rockwell Automation 技术支持代表联系。

ErrorID 代码	错误描述	更正措施
53	写入复位 ACC 失败。	与 Rockwell Automation 技术支持代表联系。
54	启动 HSC 失败。	与 Rockwell Automation 技术支持代表联系。
55	计数器被禁用。	已禁用 HSC 计数器。检查通道配置，以验证 HSC 计数器是否已启用。
56	输出掩码值无效。	检查输出掩码是否在有效范围内。对于插件 HSC 模块，范围为 0-65535。
57	高预设输出无效。	检查 HP 输出是否在有效范围内。对于插件 HSC 模块，范围为 0-65535。
58	低预设输出无效。	检查 LP 输出是否在有效范围内。对于插件 HSC 模块，范围为 0-65535。
59	不是受支持的 UPM 版本。	检查 HSC 插件模块的版本配置。
60	未添加 HSC 模块。	检查实际插件模块是否为 HSC 模块。
61	未配置 HSCE 插件模块。	与 Rockwell Automation 技术支持代表联系。
62	将 HSC 插件模式更改为"未运行"时发生 UPM 插件写入错误。	与 Rockwell Automation 技术支持代表联系。
63	将 HSC 插件模式更改为运行模式时发生 UPM 插件写入错误。	与 Rockwell Automation 技术支持代表联系。
64	读取输出状态时发生 UPM 插件读取错误。	与 Rockwell Automation 技术支持代表联系。

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，使用“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。

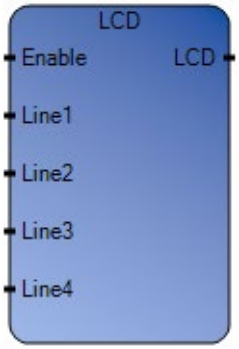
指令	描述
LCD 参考页数 317	仅限 Micro810。 在 LCD 屏幕上显示字符串或数字。
LCD_BKLT_REM 参考页数 319	在用户程序中设置远程 LCD 背光参数。
LCD_REM 参考页数 321	显示远程 LCD 屏幕的用户定义消息。
RHC 参考页数 324	读取 Micro800 控制器中的高速时钟值。
RPC 参考页数 325	从控制器或内存模块读取用户程序校验和。
DLG 参考页数 326	将变量值从运行时引擎写入到 SD 卡上的数据记录文件。
IIM 参考页数 328	在进行正常输出扫描前更新输入。
IOM 参考页数 330	在进行正常输出扫描前更新输出。
KEY_READ 参考页数 332	仅限 Micro810。 在用户显示处于活动状态时读取可选 LCD 模块上的键状态。
KEY_READ_REM 参考页数 334	仅限 Micro820。 在用户显示处于活动状态时读取可选远程 LCD 模块上的键状态。
MM_INFO 参考页数 336	读取内存模块标题信息。
MODULE_INFO 参考页数 338	从插件或扩展模块读取模块信息，2080-MEMBAK-RTC 内存模块除外。
PLUGIN_INFO 参考页数 348	从类属插件或扩展模块读取模块信息，2080-MEMBAK-RTC 内存模块除外。
PLUGIN_READ 参考页数 351	从类属插件模块读取数据，2080-MEMBAK-RTC 内存模块除外。
PLUGIN_RESET 参考页数 353	复位类属插件模块的硬件，2080-MEMBAK-RTC 内存模块除外。
PLUGIN_WRITE 参考页数 354	将数据写入类属插件模块，2080-MEMBAK-RTC 内存模块除外。
RCP 参考页数 356	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 358	读取实时时钟 (RTC) 模块信息。
RTC_SET 参考页数 360	将 RTC 数据设置为 RTC 模块信息。
SYS_INFO 参考页数 362	读取 Micro800 控制器的状态数据块。
TRIMPOT_READ 参考页数 364	从特定微调电位中读取微调电位计数值。

LCD

在可选 LCD 屏幕上显示字符串或数字。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

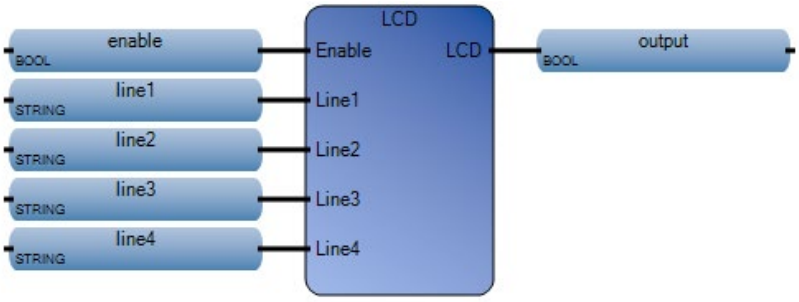
该指令适用于 Micro810 控制器。



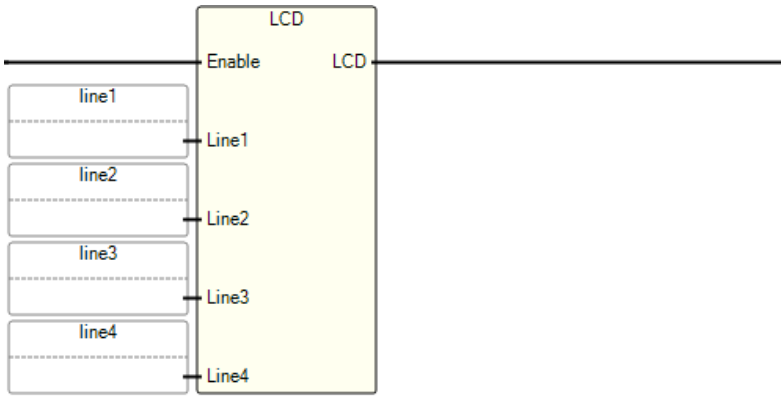
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令。 TRUE - LCD 切换至用户定义的屏幕（字符串显示在 LCD 屏幕上），而不是 I/O 状态屏幕。 FALSE - LCD 显示 I/O 状态屏幕的内容。
Line1	输入	STRING	显示在 LCD 第 1 行的字符串。
Line2	输入	STRING	显示在 LCD 第 2 行的字符串。
Line3	输入	STRING	显示在 LCD 第 3 行的字符串。
Line4	输入	STRING	显示在 LCD 第 4 行的字符串。
LCD	输出	BOOL	TRUE - 功能已启用。

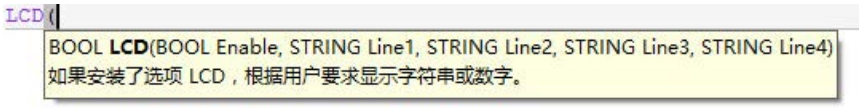
LCD 功能块图示例



LCD 梯形图示例



LCD 结构化文本示例




```
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



LCD_BKLT_REM（远程 LCD 背光）

在用户程序中设置远程 LCD 背光参数。

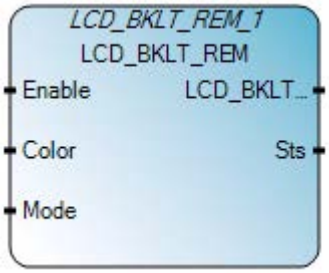
操作详细信息：

当远程 LCD 显示屏为下列屏幕时，使用在 LCD_BKLT_REM 中定义的背光设置：

- 使用 LCD_REM 定义的用户定义屏幕。
- 默认 IO 状态屏幕。
- 对于所有其他屏幕，使用的背光设置是使用远程 LCD 的菜单所定义的背光设置。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

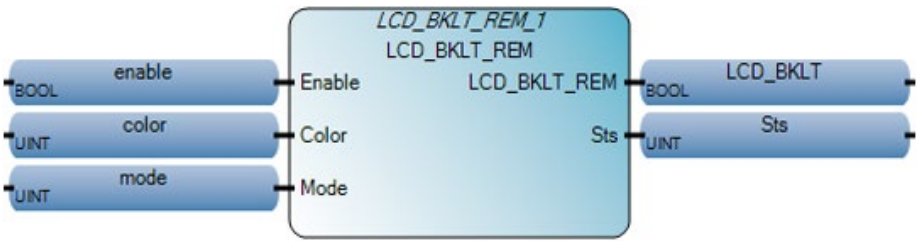
该指令适用于 Micro820、L50E 和 L70E 控制器。



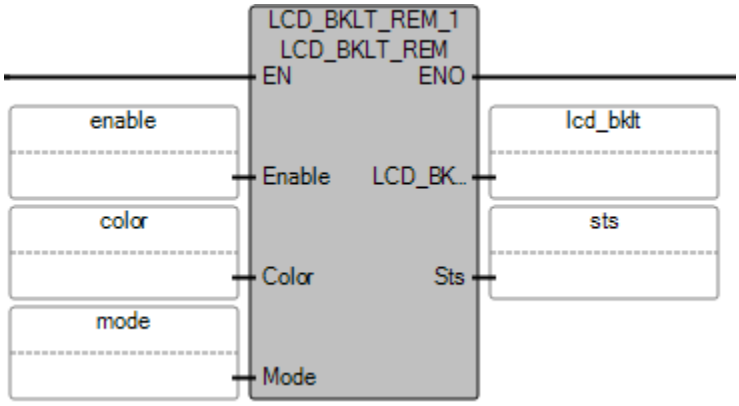
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令。 TRUE - 执行 REM_LCD_BKLT，覆盖所有当前的背光设置。 FALSE - 禁用 REM_LCD_BKLT，且远程 LCD 菜单设置生效。
Color	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
Mode	输入	UINT	• 0：永久关闭 • 1：永久开启 • 2：闪存（1 秒间隔） • 3-65535：保留
LCD_BKLT_REM	输出	BOOL	TRUE - 成功执行了指令。 FALSE - 在执行指令时发生错误。
Sts	输出	UINT	远程 LCD 操作的状态。 LCD_BKLT_REM 状态代码： • 0 - 启用输入为 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。 当存在内部状态机错误时可能会出现此错误，如控制器固件版本和 RLCD 固件版本之间的不兼容。 • 4 - 颜色代码无效。 • 5 - 模式无效。 • 6-65535 - 保留。

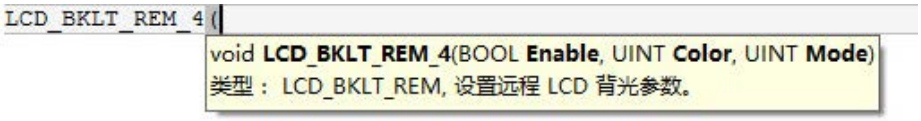
LCD_BKLT_REM 功能块图示例



LCD_BKLT_REM 梯形图示例



LCD_BKLT_REM 结构化文本示例




```
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_REM（远程 LCD）

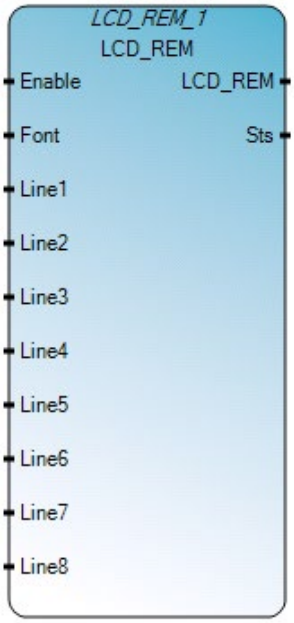
显示远程 LCD 屏幕的用户定义消息。

操作详细信息：

- 行 1 至行 8 的字符串最大长度为 24 个字符。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

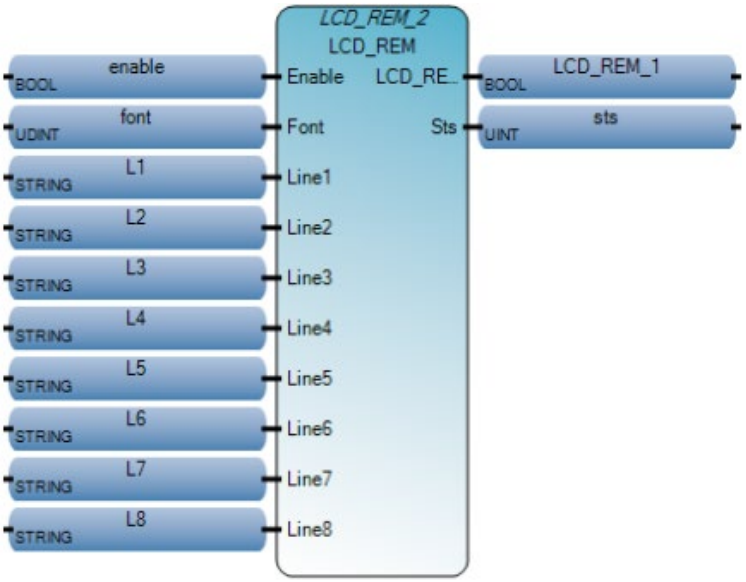
该指令适用于 Micro820、L50E 和 L70E 控制器。



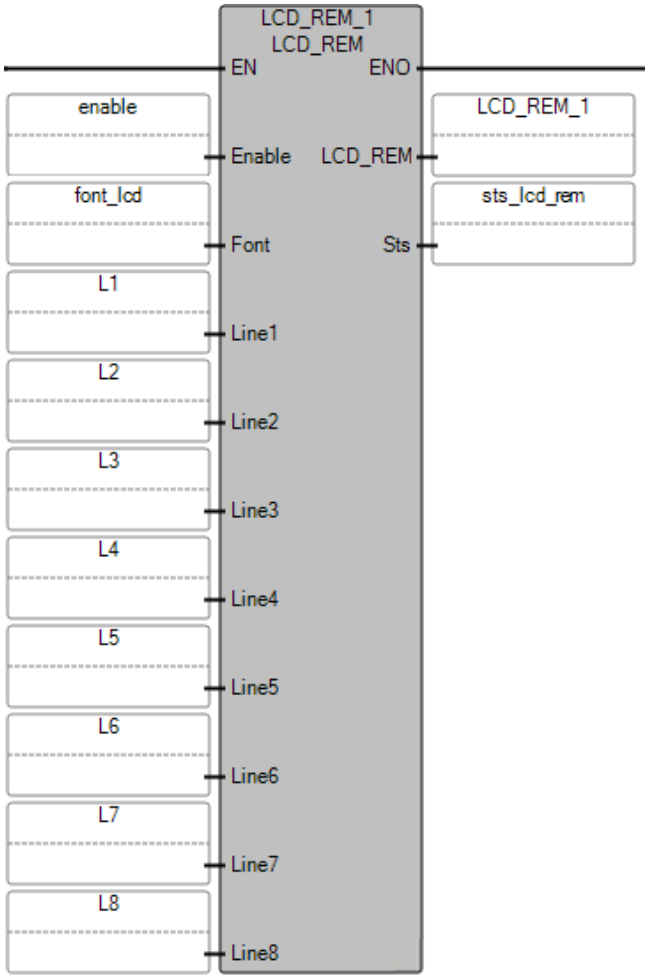
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 远程 LCD 从 I/O 状态屏幕切换到用户定义的屏幕。 FALSE - 远程 LCD 切换回 I/O 状态屏幕。
Font	输入	UDINT	启动消息的字体大小： • 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。 当启动消息的字体大小为小字体时，远程 LCD 显示： • 行 1 至行 8 的字符串。 • 每行最多 24 个字符。 当启动消息的字体大小为大或超大字体时，远程 LCD 显示： • 行 1 至行 4 的字符串。 • 忽略行 5 至行 8 的字符串。 • 每行最多 12 个字符。
Line1	输入	String	要显示在 LCD 第 1 行的字符串。 行 1 至行 8 的字符串最大长度为 24 个字符。
Line2	输入	String	显示在 LCD 第 2 行的字符串。
Line3	输入	String	显示在 LCD 第 3 行的字符串。
Line4	输入	String	显示在 LCD 第 4 行的字符串。
Line5	输入	String	显示在 LCD 第 5 行的字符串。
Line6	输入	String	显示在 LCD 第 6 行的字符串。
Line7	输入	String	显示在 LCD 第 7 行的字符串。
Line8	输入	String	显示在 LCD 第 8 行的字符串。
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。 LCD_REM 状态代码： • 0 - 启用输入为 False。 • 1 - 成功显示用户消息。 • 2 - 未检测到远程 LCD。 • 3 - 连接错误。 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。 • 4 - 字体代码无效。 • 5 - 5-65535 - 保留。

LCD_REM 功能块图示例



LCD_REM 梯形图示例



LCD_REM 结构化文本示例

```
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)
    类型： LCD_REM, 当连接远程 LCD 时，在其上显示用户字符串。

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
```

RHC（读取高速时钟）

读取 Micro800 控制器中的高速时钟值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



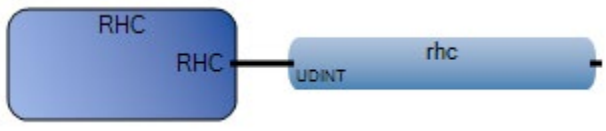
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 读取高速时钟。 FALSE - 不执行任何操作。 适用于梯形图编程。
ENO	输出	BOOL	启用“输出”。 适用于梯形图编程。
RHC	输出	UDINT	高速时钟的值。

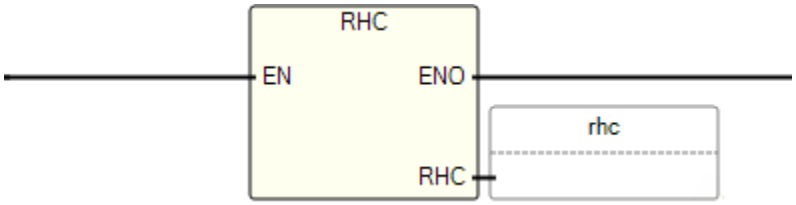
高速时钟分辨率

控制器类型	增量	时间基准	分辨率
Micro810	每 40 微秒对应 4	10 微秒	40 微秒
Micro820 Micro830 Micro850	每 10 微秒对应 1	10 微秒	10 微秒

RHC 功能块图示例



RHC 梯形图示例



RHC 结构化文本示例




```
1  rhc := RHC();
```

(* 与之等效的 ST: *)

TESTOUTPUT2 := RHC() ;

RPC（读取程序校验和）

从控制器或内存模块读取用户程序校验和。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



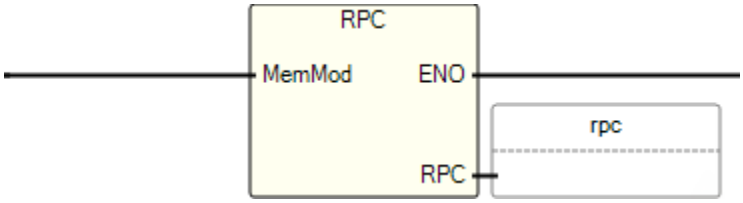
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
MemMod	输入	BOOL	TRUE - 值取自内存模块。 FALSE - 值取自 Micro800 控制器。
ENO	输出	BOOL	启用"输出"。 适用于梯形图编程。
RPC	输出	UDINT	指定用户程序的校验和。

RPC 功能块图示例



RPC 梯形图示例



RPC 结构化文本示例




```
1  MemMod := TRUE;
2  rpc := RPC(MemMod);
```

(* 与之等效的 ST： *)

TESTOUTPUT2 := RPC(TESTINPUT) ;

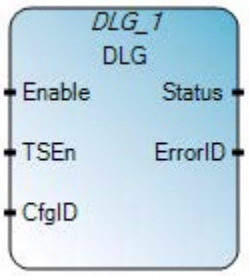
DLG（数据记录）

将变量值从运行时引擎写入到 SD 卡上的数据记录文件。

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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

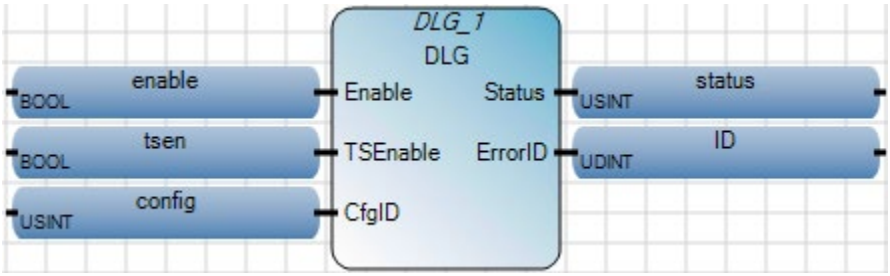
参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用数据记录写入。 TRUE - 检测到上升沿启用，在之前的指令操作都已完成后启动数据记录操作。 FALSE - 未检测到上升沿。
TSEnable	输入	BOOL	TRUE - 日期和时间戳记录启用标志。
CfgID	输入	USINT	数据记录配置 VA ID 编号，从 1-10。
ENO	输出	BOOL	启用“输出”。 适用于梯形图编程。

参数	参数类型	数据类型	描述
Status	输出	USINT	指令的当前状态。 数据记录状态代码： • 0 - 空闲 • 1 - 执行 • 2 - 成功，表示数据记录已完成。 • 3 - 错误，指示数据记录已完成，但出错。
ErrorID	输出	UDINT	DLG 的唯一数字错误代码。

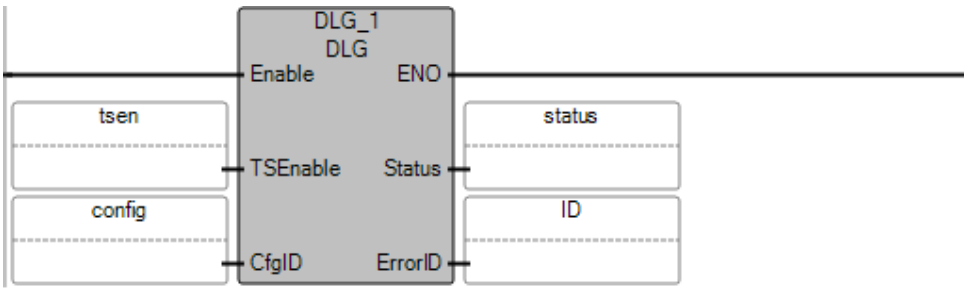
DLG 错误代码

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_ACCESS	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_BUSY	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。

DLG 功能块图示例

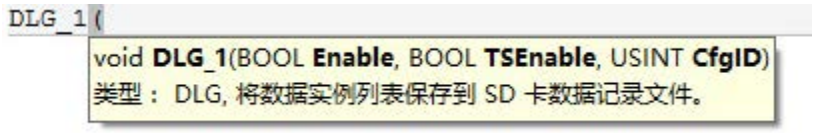


DLG 梯形图示例



LG

DLG 结构化文本示例




```
1  DLG_1 (Enable, TSEnable, CfgID)
2  status := DLG_1.Status
3  ID := DLG_1.ErrorID
```

IIM（即时输入）

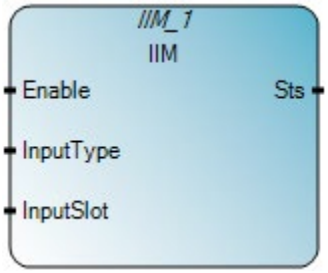
在进行正常输出扫描前更新输入。

操作详细信息：

- 通常用于中断程序开始时选择或屏蔽即时扫描的输入，以获取当前输入值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro830、Micro850 和 Micro870 控制器。

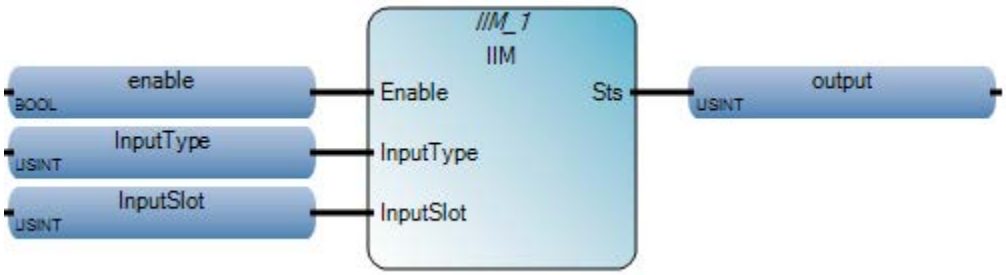


使用此表有助于确定该指令的参数值。

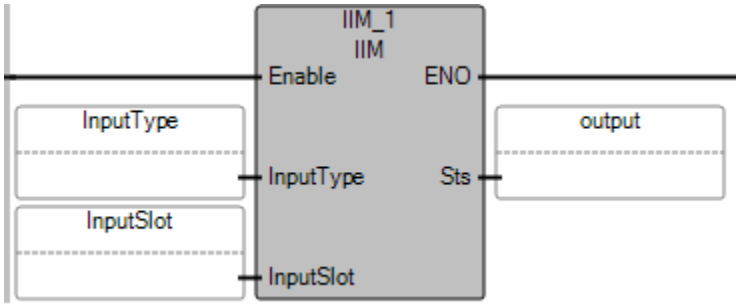
参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行指令块。 FALSE - 不执行。
InputType	输入	USINT	标识输入类型： 0 - 嵌入式输入。 1- 插件输入。
InputSlot	输入	USINT	标识输入插槽。 0 - 嵌入式输入 1、2、3、4、5 – 插件的插槽编号。（插槽从左到右进行编号，起始编号为 1。） 对于嵌入式输入，永远为 0。 对于插件输入，输入插槽为 1、2、3、4、5（插件插槽编号，从最左边的插槽 [1] 开始）。
Sts	输出	USINT	即时输入扫描状态。 IIM 状态 (Sts) 代码： • 0x00 - 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/输出插槽无效。

参数	参数类型	数据类型	描述
ENO	输出	BOOL	启用"输出"。 TRUE - 输入已更新。 FALSE - 输入未更新。 仅适用于梯形图编程。

IIM 功能块图示例



IIM 梯形图示例



IIM 结构化文本示例




```
1  enable := TRUE;
2  InputType := 0;
3  InputSlot := 0;
4  IIM_1(enable, InputType, InputSlot);
5  output := IIM_1.Sts;
```

结果



IOM（即时输出）

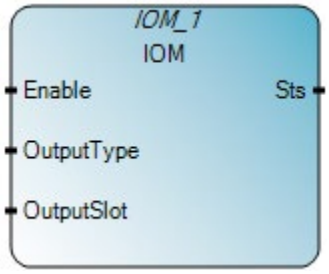
在进行正常输出扫描前更新输出。

操作详细信息：

- 通常用于中断编程结束时选择或屏蔽即时扫描和更新的输出。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro830、Micro850 和 Micro870 控制器。

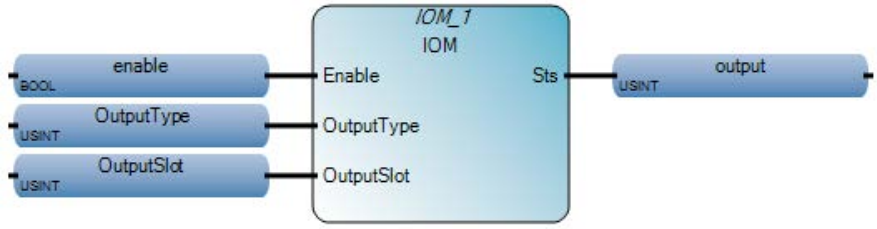


使用此表有助于确定该指令的参数值。

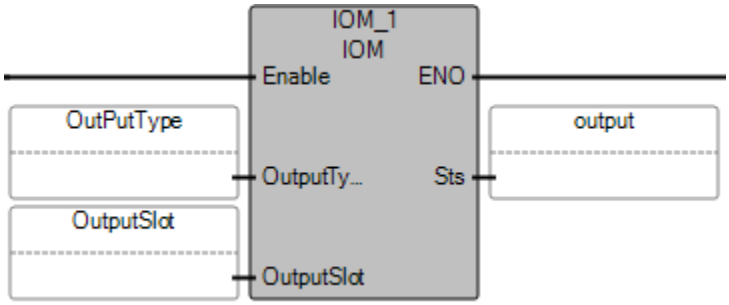
参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行指令块。 FALSE - 不执行。
OutputType	输入	USINT	标识输出类型： 0 - 嵌入式输出。 1 - 插件输出。
OutputSlot	输入	USINT	标识输出插槽： 0 - 嵌入式输出 1、2、3、4、5 – 插件的插槽编号。（插槽从左到右进行编号，起始编号为 1。） 对于嵌入式输出，永远为 0。 对于插件输出，输出插槽为 1、2、3、4、5（插件插槽编号，从最左边的插槽 [1] 开始）。
Sts	输出	USINT	即时输出扫描状态。 IOM (Sts) 状态代码： • 0x00 - 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。

参数	参数类型	数据类型	描述
			• 0x03 - 输入/输出插槽无效。
ENO	输出	BOOL	启用"输出"。 TRUE - 输出更新。 FALSE - 输出不更新。 仅适用于梯形图编程。

IOM 功能块图示例



IOM 梯形图示例



IOM 结构化文本示例




```
1  enable := TRUE;
2  OutputType := 0;
3  OutputSlot := 0;
4  IOM_1(enable, OutputType, OutputSlot);
5  output := IOM_1.Sts;
```

结果

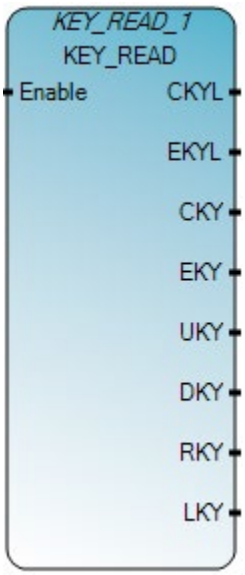


KEY_READ（读取 LCD 上的键）

在用户显示处于活动状态时读取可选 LCD 模块上的键状态。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

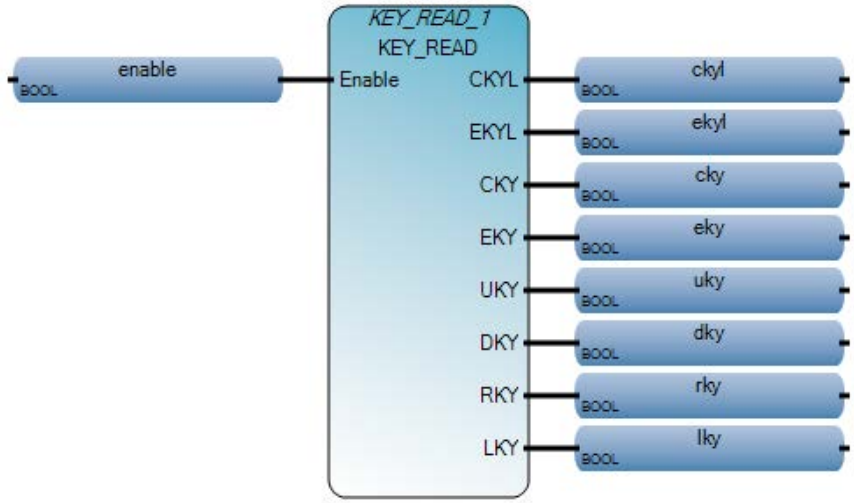
该指令适用于 Micro810 控制器。



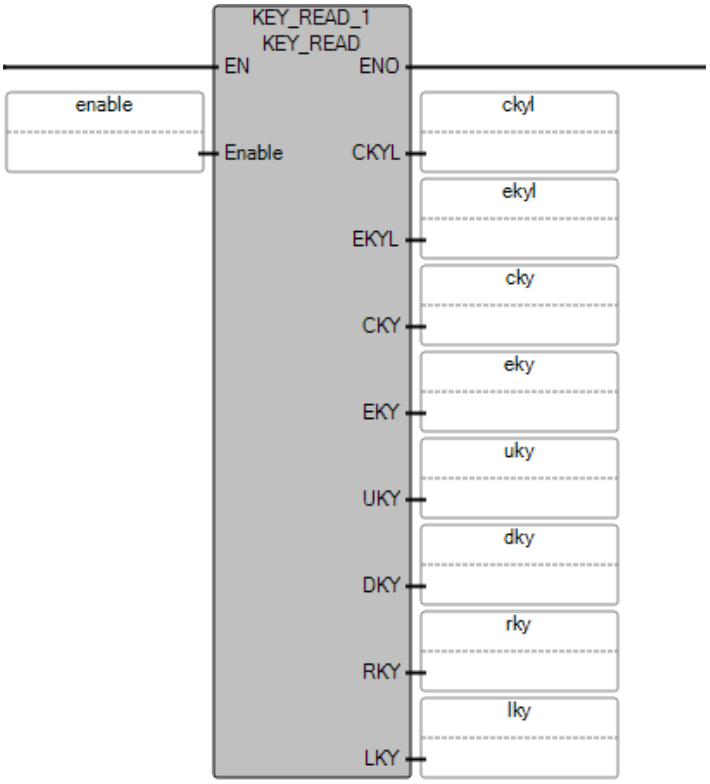
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 启用远程 LCD 键盘上的读取键。 FALSE - 禁用远程 LCD 键盘上的读取键。
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。

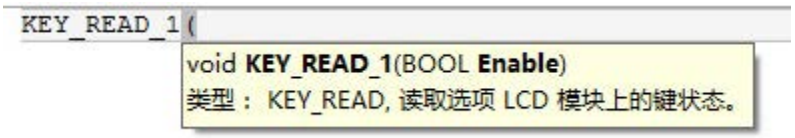
KEY_READ 功能块图示例



KEY_READ 梯形图示例



KEY_READ 结构化文本示例



```
KEY_READ_1(enable);
ckyl := KEY_READ_1.CKYL;
ekyl := KEY_READ_1.EKYL;
cky := KEY_READ_1.CKY;
eky := KEY_READ_1.EKY;
uky := KEY_READ_1.UKY;
dky := KEY_READ_1.DKY;
lky := KEY_READ_1.LKY;
rky := KEY_READ_1.RKY;
```

(* 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 ;
```

KEY_READ_REM（远程 LCD 读取键）

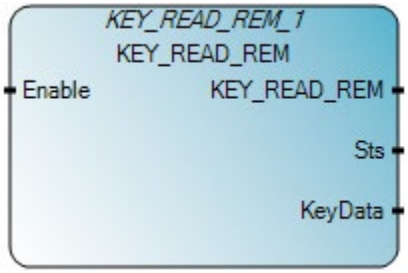
在用户显示处于活动状态时读取可选远程 LCD 模块上的键状态。

操作详细信息：

- LCD_REM 指令用于激活远程 LCD 模块的用户显示。如果用户显示未激活，KEY_READ_REM 执行期间会发生错误。
- LCD 功能文件中的 P-BUTTON 属性激活；否则所有键状态为 FALSE。
- KEY_READ_REAM 指令只支持按单个键；不支持组合按两个键。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、L50E 和 L70E 控制器。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 启用 FALSE - 禁用
KEY_READ_REM	输出	BOOL	TRUE - 已成功读取远程 LCD 键数据。 FALSE - 启用为 False，读取存在错误。远程 LCD 键数据或用户显示未激活。

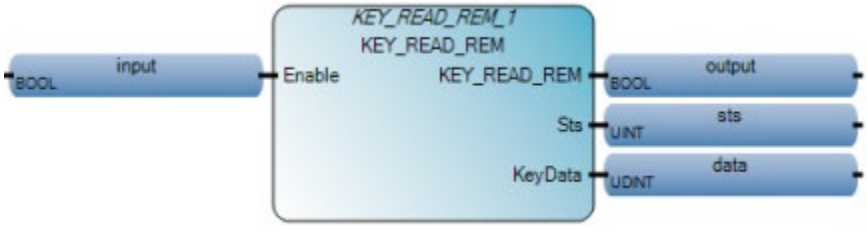
参数	参数类型	数据类型	描述
Sts	输出	UINT	KEY_READ_REM 操作的状态。 KEY_READ_REM 状态代码： • 0 - 启用输入为 False。 • 1 - 已成功读取键数据。 • 2 - 未检测到远程 LCD。 出现以下情况时可能会发生： • 未将远程 LCD 物理连接到控制器（或配线错误）。 • 串行端口设置不符合远程 LCD 要求。 • 3 - 连接错误。 当存在内部状态机器错误时可能会发生。可能导致控制器固件版本和 RLCD 固件版本之间的不兼容。 • 4 - 用户显示未激活。 • 5-65535 - 保留。
KeyData	输出	UDINT	远程 LCD 键盘数据。 键数据位字段表中定义的键数据定义。

键数据位字段表

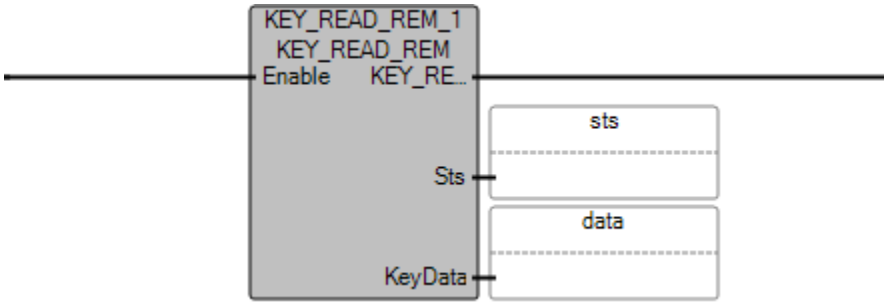
使用此表帮助确定 KEY_READ_REM 的键数据位字段。

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。
13	CKYL	TRUE = 按下 Cancel 键持续 2 秒钟以上。
14-31	--	已保留。

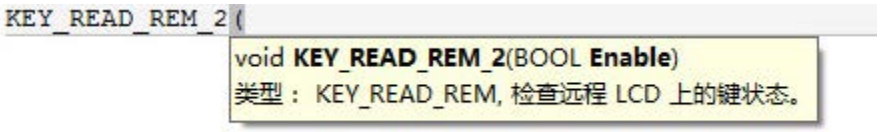
KEY_READ_REM 功能块图示例



KEY_READ_REM 梯形图示例



KEY_READ_REM 结构化文本示例




```
1  KEY_READ_REM (Enable);
2  output := KEY_READ_REM.ENO
3  sts := KEY_READ_REM.Sts
4  data := KEY_READ_REM.KeyData
```

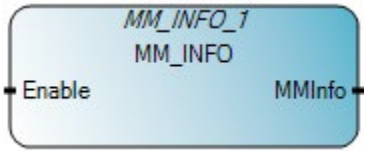
MM_INFO（内存模块信息）

读取内存模块标题信息。如果内存模块不存在，则所有值均变为零 (0)。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

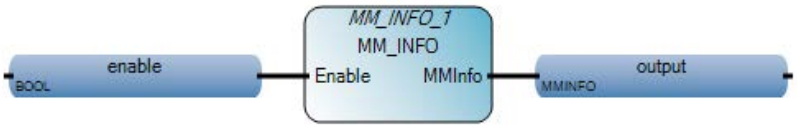
当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。



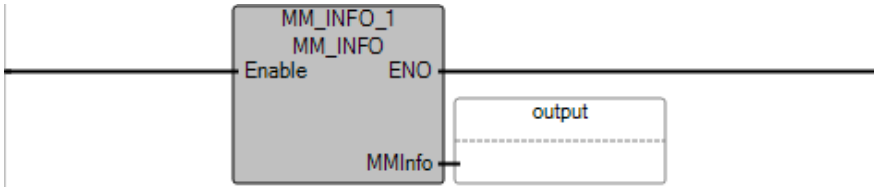
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 读取内存模块标题信息。 FALSE - 不执行读取操作，且输出内存模块信息无效。
MMInfo	输出	MMINFO	内存模块信息在 MMINFO 数据类型 参考页数 338 中定义。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

MM_INFO 功能块图示例



MM_INFO 梯形图示例



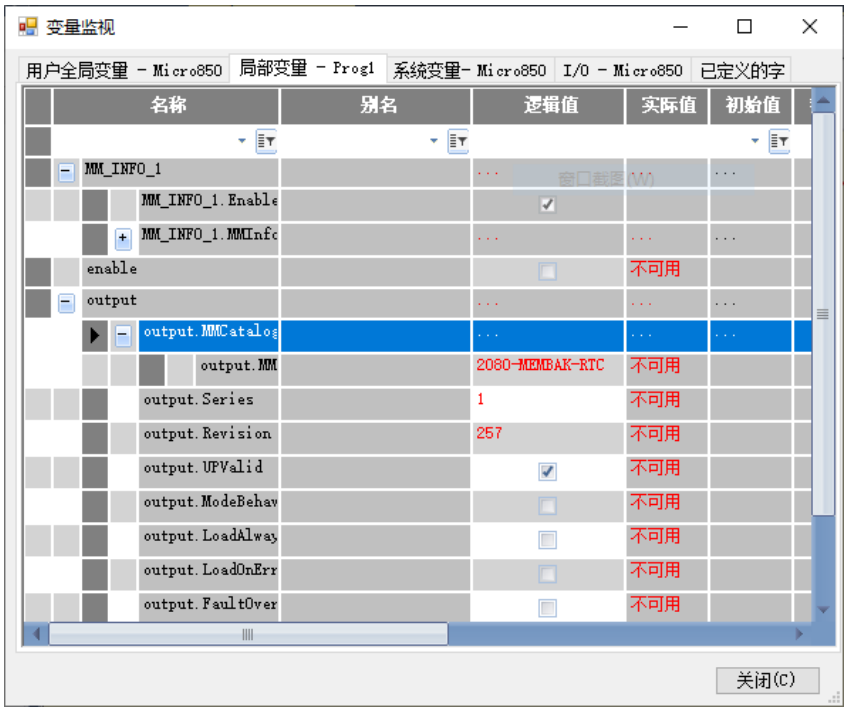
MM_INFO 结构化文本示例



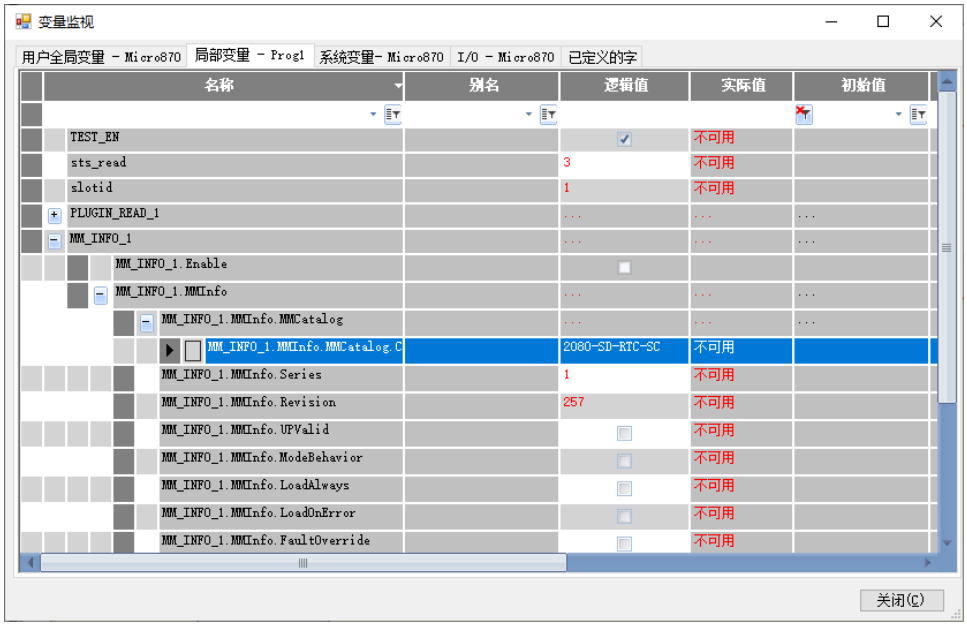

```
1  MM_INFO_1(enable);
2  output := MM_INFO_1.MMInfo;
```

结果

对于使用 2080-MEMBAK-RTC 的控制器：



对于使用 SD 卡的控制器：



MMINFO 数据类型

下表描述了 MMINFO 数据类型参数。

参数	数据类型	描述
MMCatalog	MMCATNUM	内存模块的类型编号。 在带有 SD 卡的控制器上使用 MM_INFO 参考页数 336 指令时，MMCatalog 为“SD CARD”。
Series	UINT	内存模块系列。 在带有 SD 卡的控制器上使用 MM_INFO 指令时，系列为 1。
Revision	UINT	内存模块版本。 在带有 SD 卡的控制器上使用 MM_INFO 时，版本为 257。
UPValid	BOOL	用户程序存在（TRUE：找到了可能有效的项目）。 注意：即使为 TRUE，如果单个文件丢失或损坏，仍可能会在下载或恢复过程中将项目检测为无效项目。
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。

MODULE_INFO

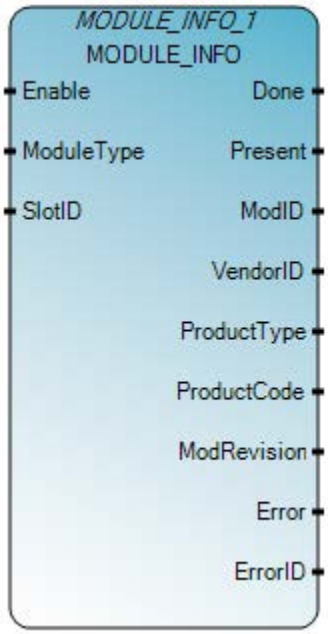
从插件模块或扩展模块读取模块信息。

操作详细信息：

- 在运行时间读取插件模块信息。
- 2080-MEMBAK-RTC 内存插件模块不受支持。
- 在模块通电时读取扩展模块信息。
- 当没有为插件或扩展模块定义 ModuleID、ProductType 或 ProductCode 时，MODULE_INFO 操作会为相应的输出参数返回 0。
- 插件和扩展模块标识信息由 Allen-Bradley 定义，并在下面作为 MODULE_INFO 描述信息的一部分提供。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。扩展模块仅在 Micro850 和 Micro870 控制器上受支持。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行 MODULE_INFO 读取操作。 FALSE - 不执行读取操作。所有输出数据值重置为 0。
ModuleType	输入	USINT	标识模块类型： • 1 - 2085 扩展模块。 • 2 - 2080 插件模块。
SlotID	输入	USINT	插件或扩展模块所在的插槽编号。 插槽 ID 为：1、2、3、4、5 插槽 1 位于最左侧。
Done	输出	BOOL	TRUE - 操作已成功完成。 FALSE - 未执行操作或发生错误条件。
Present	输出	BOOL	检测控制器插槽中的插件或扩展模块。 TRUE - 模块物理存在。 FALSE - 模块实际上不存在。
ModID	输出	UINT	控制器插槽中模块的标识。 • 已为插件模块定义唯一的模块标识符。 • 没有为扩展模块定义唯一的模块标识符，ModID 返回 0。
VendorID	输出	UINT	插件或扩展模块的供应商 ID。 对于 Allen-Bradley 产品，供应商 ID 为 1。
ProductType	输出	UINT	插件或扩展模块的产品类型。
ProductCode	输出	UINT	插件或扩展模块的产品代码。
ModRevision	输出	UINT	插件或扩展模块的版本信息。
Error	输出	BOOL	表示存在错误条件。 TRUE - 检测到错误。 FALSE - 没有错误。

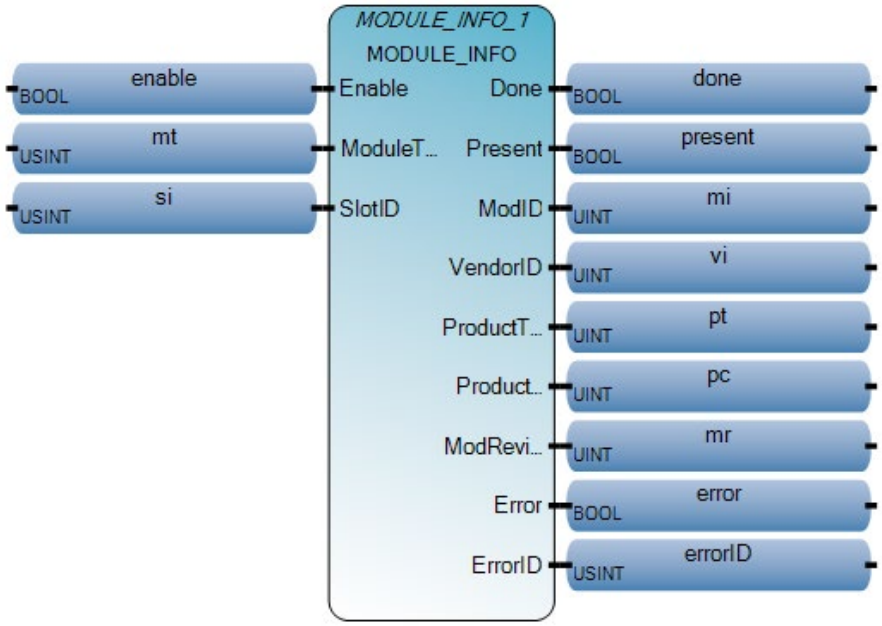
参数	参数类型	数据类型	描述
ErrorID	输出	USINT	标识错误的唯一数字。这些错误在 MODULE_INFO 错误代码中定义。

MODULE_INFO 错误代码

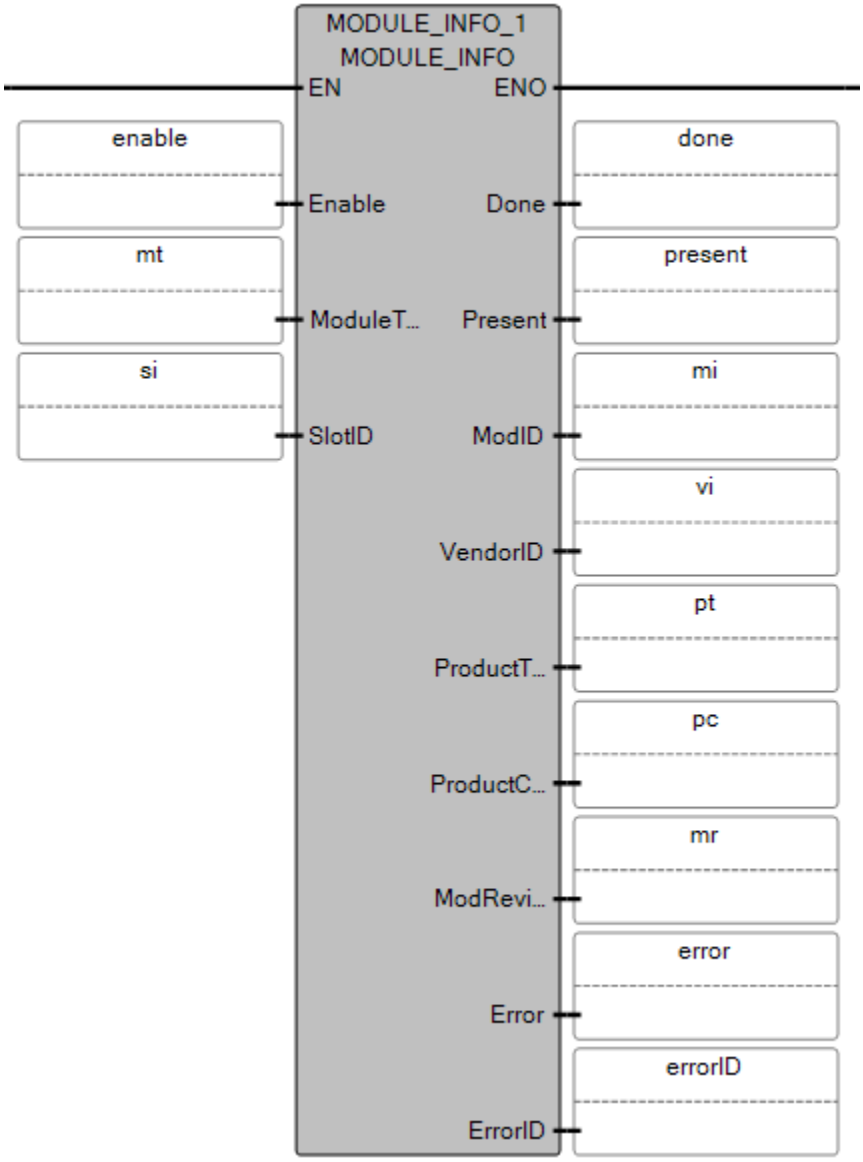
使用此表可确定 MODULE_INFO 错误代码和描述信息。

错误代码	错误描述
1	模块类型无效。 更改为有效模块类型。 有效模块类型为： 1 - 2085 扩展模块。仅在 Micro850 控制器上受支持。 2 - 2080 插件模块。在 Micro820、Micro830 和 Micro850 控制器上受支持。
2	插槽编号无效。 更改为有效插槽编号。
3	扩展模块类型无效。
4	扩展模块严重错误。
5	不支持插件模块读取信息。
6	读取模块 ID 时发生插件模块读取错误。
7	读取供应商 ID 时发生插件模块读取错误。
8	读取产品类型时发生插件模块读取错误。
9	读取产品代码时发生插件模块读取错误。
10	读取模块版本时发生插件模块读取错误。

MODULE_INFO 功能块图示例



MODULE_INFO 梯形图示例



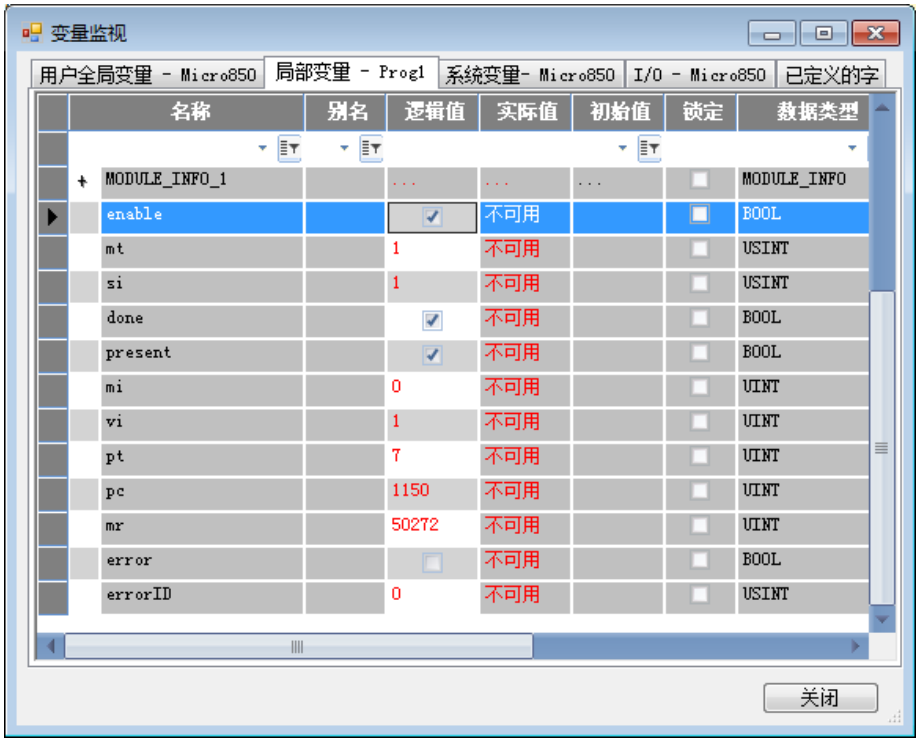
MODULE_INFO 结构化文本示例

```
1  MODULE_INFO_1(enable, mt, si);
2  done := MODULE_INFO_1.Done;
3  present :=MODULE_INFO_1.Present;
4  mi :=MODULE_INFO_1.ModID;
5  vi :=MODULE_INFO_1.VendorID;
6  pt :=MODULE_INFO_1.ProductType;
7  pc :=MODULE_INFO_1.ProductCode;
8  mr :=MODULE_INFO_1.ModRevision;
9  error :=MODULE_INFO_1.Error;
10 errorID :=MODULE_INFO_1.ErrorID;
```

```
MODULE_INFO_1(
```

void **MODULE_INFO_1**(BOOL **Enable**, USINT **ModuleType**, USINT **SlotID**)
（无注释）

结果



MODULE_INFO - 插件和扩展模块信息

以下信息提供插件和扩展模块类型、模块 ID、供应商 ID、产品代码，以及扩展模块的版本字说明信息（由 Allen-Bradley 定义）。

插件模块信息

使用此表可确定 Allen-Bradley 定义的插件模块信息。

插件模块	插件类型	模块 ID	供应商 ID	产品类型	产品代码
2080-IF2	模拟	96	1	10	32
2080-IF4	模拟	98	1	10	33
2080-OF2	模拟	100	1	10	34
2080-TC2	模拟	102	1	10	35
2080-RTD2	模拟	104	1	10	36
2080-DNET20	通信	34	1	12	249
2080-SERIALISOL	通信	32	1	-	-
2080-IQ4	数字	192	1	7	192
2080-OB4	数字	193	1	7	193
2080-OV4	数字	194	1	7	194
2080-IQ4OB4	数字	195	1	7	195
2080-IQ4OV4	数字	196	1	7	196
2080-OW4I	数字	197	1	7	197
2080-MOT-HSC	专用	48	1	43	48
2080-TRIMPOT6	专用	72	1	-	-

扩展模块信息

使用此表可确定 Allen-Bradley 定义的扩展模块信息。

模型编号	扩展类型	模块 ID	供应商 ID	产品类型	产品代码
2085-IF4	模拟	-	1	10	208
2085-IF8	模拟	-	1	10	206
2085-IRT4	模拟	-	1	10	213
2085-OF4	模拟	-	1	10	214
2085-IA8	数字	-	1	7	1148
2085-IM8	数字	-	1	7	1152
2085-IQ16	数字	-	1	7	1144
2085-IQ32T	数字	-	1	7	1145
2085-OA8	数字	-	1	7	1149
2085-OB16	数字	-	1	7	1146
2085-OV16	数字	-	1	7	1147
2085-OW16	数字	-	1	7	1151
2085-OW8	数字	-	1	7	1150

扩展模块版本字说明

使用此表可确定 Allen-Bradley 扩展模块的版本字信息。

位	名称	描述
15 14	最大波特率 (1:0)	这些位确定了对应于最大波特率 (1:0) 的最大频率。 00 (bin) 为 2Mbps 01 (bin) 为 4Mbps 10 (bin) 为 8Mbps 11 (bin) 为 16Mbps
13 12 11 10	次要版本 (3:0)	次要版本 (3:0) 是产品的次要版本名称。此字段表示通过供应商 ID、产品类型、产品代码、系列和主要版本指定的类型编号的次要版本。 次要版本的范围为 0 到 15。
9 8 7 6 5	主要版本 (4:0)	主要版本 (4:0) 是产品的主要版本名称。此字段表示通过供应商 ID、产品类型、产品代码和系列指定的类型编号的主要版本。 主要版本的范围为 0 到 31。

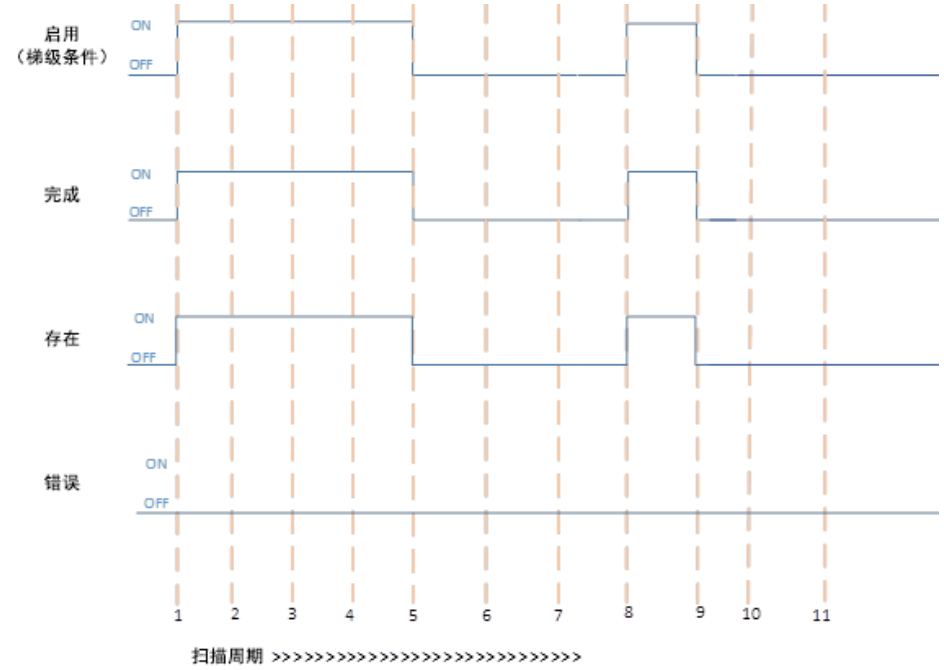
位	名称	描述
4	系列 (4:0)	系列 (4:0) 是产品的系列名称。此字段表示通过供应商 ID、产品类型和产品代码指定的类型编号的系列字母。

系列 (4:0)	系列	系列 (4:0)	系列	系列 (4:0)	系列	系列 (4:0)	系列
0	A	8	I	16	Q	24	Y
1	B	9	J	17	R	25	Z
2	C	10	K	18	S	26	AA
3	D	11	L	19	T	27	AB
4	E	12	M	20	U	28	AC
5	F	13	N	21	V	29	AD
6	G	14	O	22	W	30	AE
7	H	15	P	23	X	31	>AE

MODULE_INFO 指令时间图

以下时序图示例描述了 MODULE_INFO 指令 参考页数 338 的执行场景。

当模块物理存在时成功执行

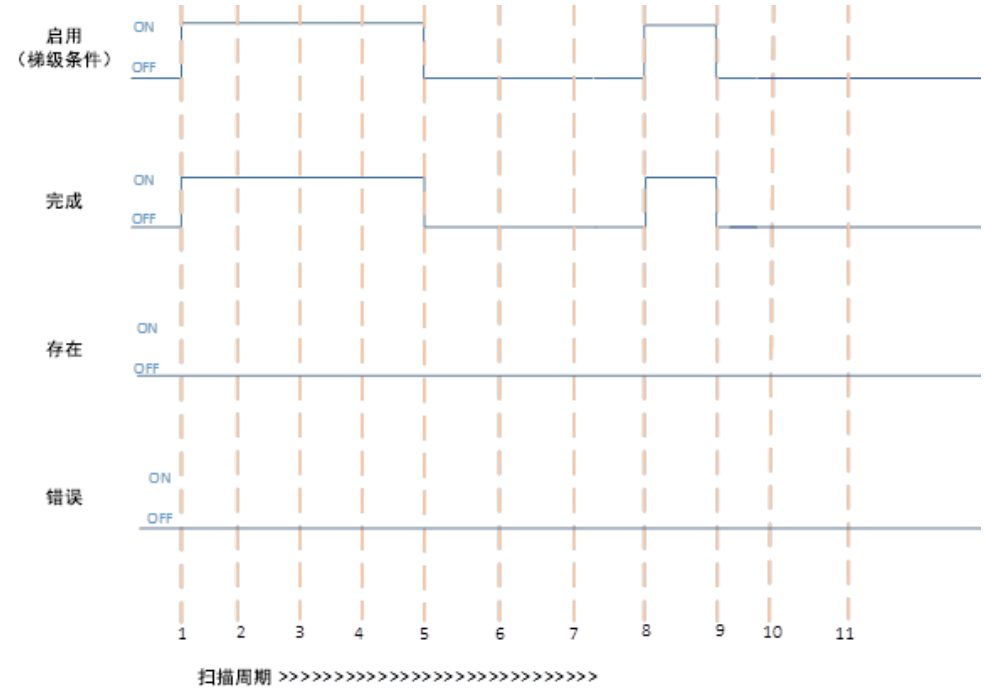


使用此表可帮助确定每个扫描周期的 MODULE_INFO 参数值。

扫描周期	描述
1，8	在下列情况下梯级条件变为 TRUE： • “启用”输入位为 TRUE。 • ModuleType 和 SlotID 有效。实际模块存在。 • “完成”和“存在”输出位为 TRUE。

扫描周期	描述
	• "错误"输出位为 FALSE。 • 为模块 ID、供应商 ID、产品类型、产品代码和模块版本相应地更新模块相关信息。
2，3，4	梯级条件未发生更改。
5 的常数值。9	当"启用"位为 FALSE 时梯级条件变为 FALSE。所有输出参数被清除。
6、7、10、11	梯级条件未发生更改。"启用"位为 FALSE。所有输出参数被清除。

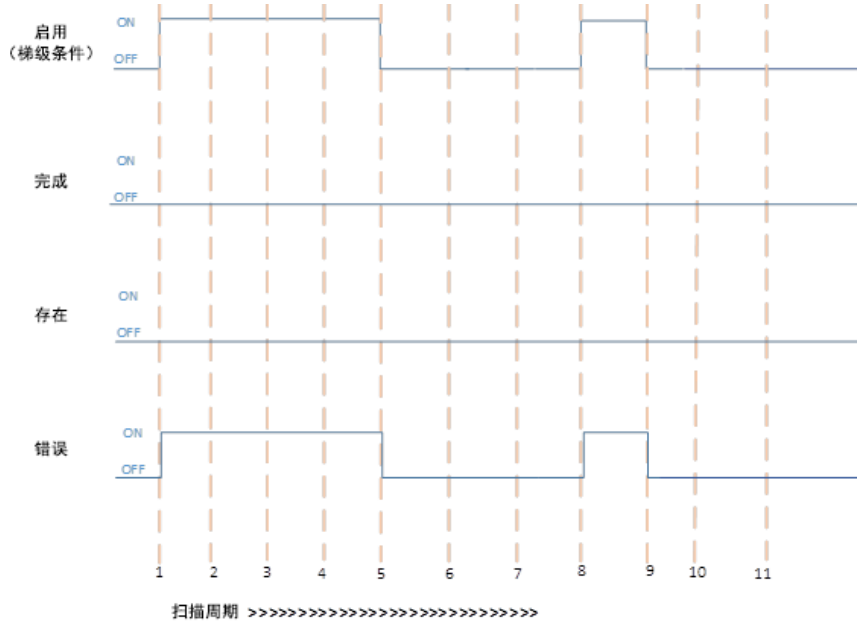
当模块物理上不存在时成功执行



使用此表可帮助确定每个扫描周期的 MODULE_INFO 参数值。

扫描周期	描述
1，8	在下列情况下梯级条件变为 TRUE： • "启用"输入位为 TRUE。 • ModuleType 和 SlotID 有效。实际模块不存在。 • "完成"输出位为 TRUE。 • "错误"和"存在"输出位为 FALSE。 • 为模块 ID、供应商 ID、产品类型、产品代码和模块版本相应地更新模块相关信息。
2，3，4	梯级条件未发生更改。 • "启用"输入位为 TRUE。 • 输入参数有效，并且实际模块不存在。 • 相应地更新输出参数。
5 的常数值。9	在下列情况下梯级条件变为 FALSE： • "启用"位为 FALSE。 • 所有输出参数被清除。
6、7、10、11	梯级条件未发生更改。"启用"位为 FALSE。所有输出参数被清除。

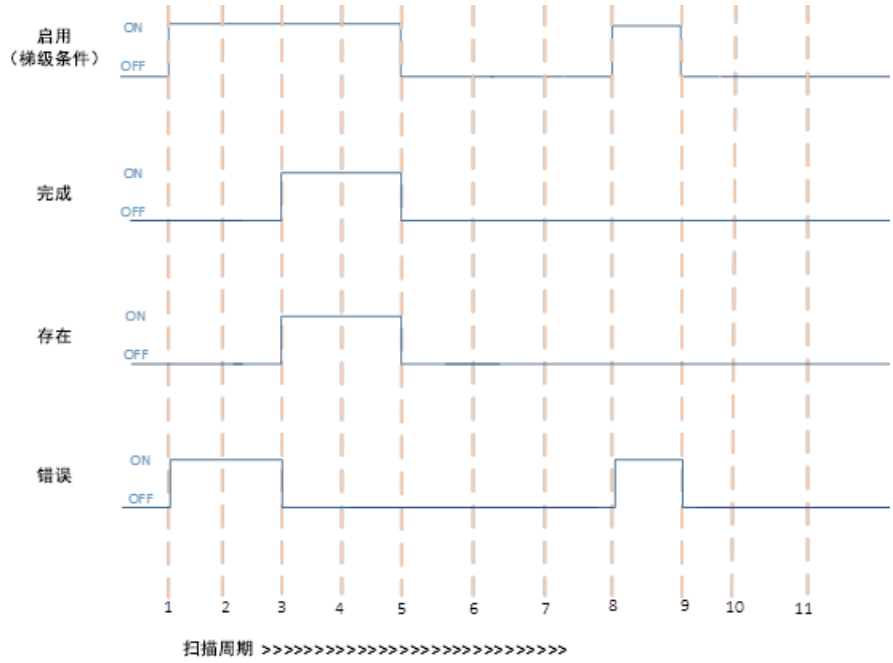
MODULE_INFO 执行出错



使用此表可帮助确定每个扫描周期的 MODULE_INFO 参数值。

扫描周期	描述
1，8	在下列情况下梯级条件变为 TRUE： • “启用”输入位为 TRUE。 • ModuleType 和 SlotID 有效。实际模块不存在。 • “完成”和“存在”输出位为 TRUE。 • “错误”输出位为 TRUE。 • 模块 ID、供应商 ID、产品类型、产品代码和模块版本的模块相关信息被清除。
2，3，4	梯级条件未发生更改。 • “启用”输入位为 TRUE。 • ModuleType、SlotID 或两者无效。 • 相应地更新输出参数。
5 的常数值。9	在下列情况下梯级条件变为 FALSE： • “启用”位为 FALSE。 • 所有输出参数被清除。
6、7、10、11	梯级条件未发生更改。“启用”位为 FALSE。所有输出参数被清除。

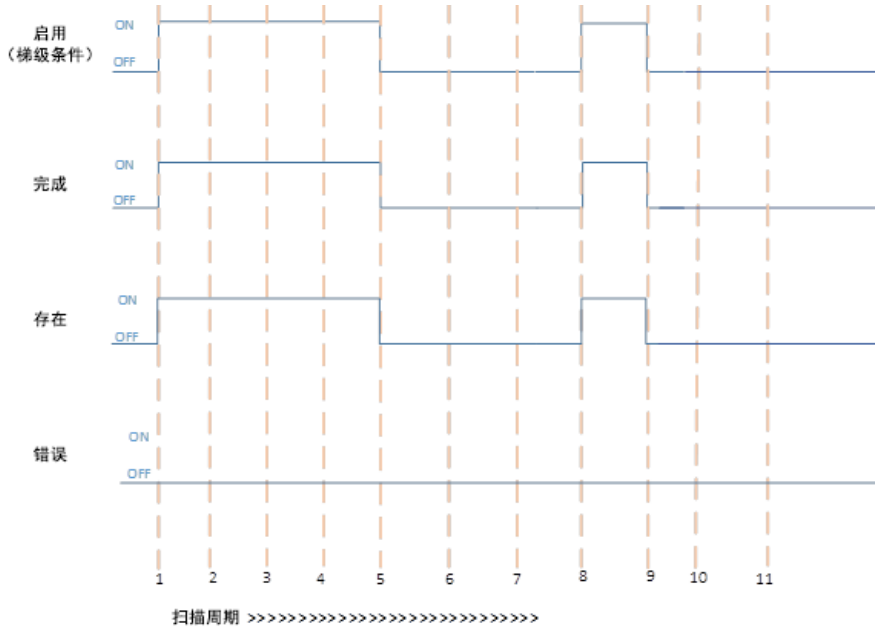
MODULE_INFO 成功执行但出错 - 无实际模块



使用此表可帮助确定每个扫描周期的 MODULE_INFO 参数值。

扫描周期	描述
1，8	在下列情况下梯级条件变为 TRUE： • “启用”输入位为 TRUE。 • ModuleType、SlotID 或两者无效。实际模块不存在。 • “完成”和“存在”输出位为 FALSE。 • “错误”输出位为 TRUE。 • 模块 ID、供应商 ID、产品类型、产品代码和模块版本的模块相关信息被清除。
2	梯级条件未发生更改。 • “启用”输入位为 TRUE。 • ModuleType、SlotID 或两者无效。 • 相应地更新输出参数。
3，4	梯级条件未发生更改。 • “启用”输入位为 TRUE。 • ModuleType 和 SlotID 或两者有效。模块物理存在。 • “完成”和“存在”输出位为 TRUE。 • “错误”输出位为 FALSE。 • 为模块 ID、供应商 ID、产品类型、产品代码和模块版本相应地更新模块信息。
5 的常数值。9	在下列情况下梯级条件变为 FALSE： • “启用”位为 FALSE。 • 所有输出参数被清除。
6、7、10、11	梯级条件未发生更改。“启用”位为 FALSE。所有输出参数被清除。

当实际模块存在时，MODULE_INFO 成功执行但出现错误



使用此表可帮助确定每个扫描周期的 MODULE_INFO 参数值。

扫描周期	描述
1，8	在下列情况下梯级条件变为 TRUE： • “启用”输入位为 TRUE。 • ModuleType、SlotID 或两者有效。模块物理存在。 • “完成”和“存在”输出位为 TRUE。 • “错误”输出位为 FALSE • 为模块 ID、供应商 ID、产品类型、产品代码和模块版本相应地更新模块相关信息。
2	梯级条件未发生更改。 • “启用”输入位为 TRUE。 • 模块输入参数有效，并且模块物理存在。 • 相应地更新输出参数。
3，4	梯级条件未发生更改。 • “启用”输入位为 TRUE。 • ModuleType 和 SlotID 或两者无效。模块物理存在。 • “完成”和“存在”输出位为 FALSE。 • “错误”输出位为 TRUE 并且被清除。 • 为模块 ID、供应商 ID、产品类型、产品代码和模块版本相应地更新模块信息。
5 的常数值。9	在下列情况下梯级条件变为 FALSE： • “启用”位为 FALSE。 • 所有输出参数被清除。
6、7、10、11	梯级条件未发生更改。“启用”位为 FALSE。所有输出参数被清除。

PLUGIN_INFO（插件信息）

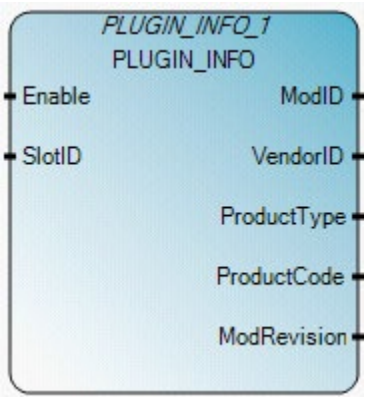
从类属插件或扩展模块读取模块信息。

操作详细信息：

- 在 Connected Components Workbench 10 或更高版本中，PLUGIN_INFO 指令可读取任何类属插件或扩展模块信息，2080-MEMBAK-RTC 模块除外。
- 如果类属插件或扩展模块不存在，则所有值恢复为零 (0)。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

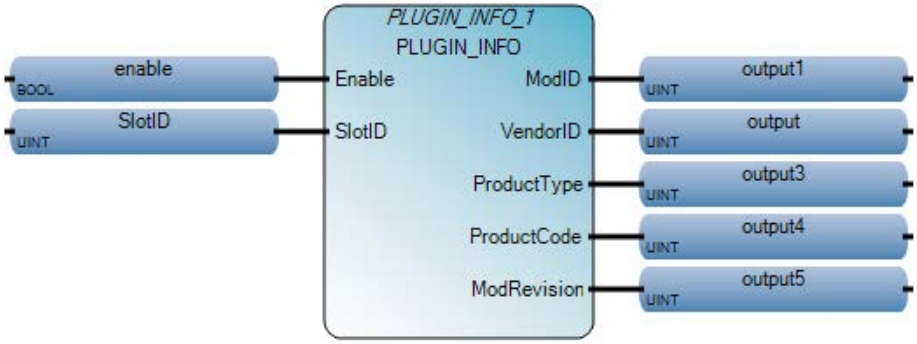
该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



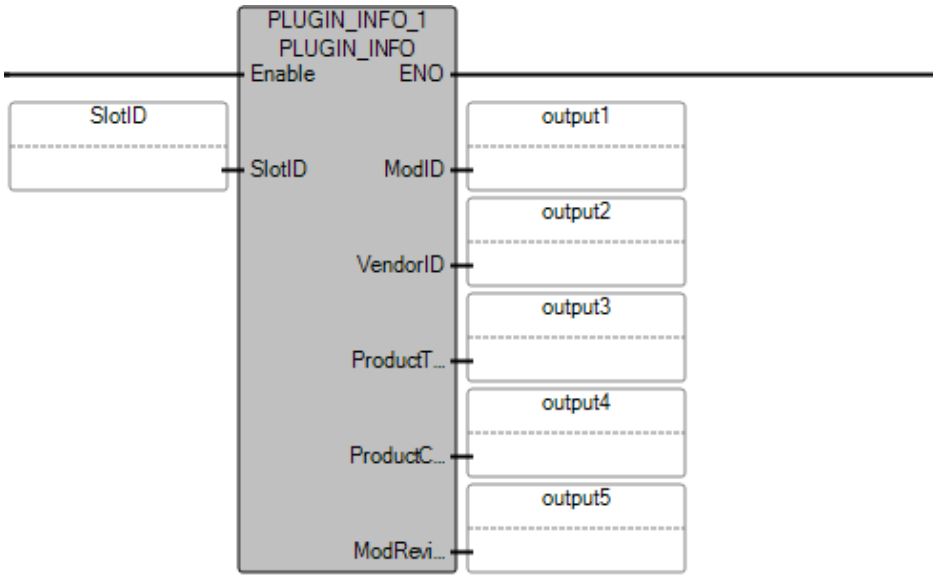
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行插件或扩展模块信息读取。 FALSE - 指令块未执行。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 （从左边的第一个插槽开始 = 1）。 扩展插槽编号： 插槽 ID = 101、102、103、104 （从左边的第一个插槽开始 = 101）。
ModID	输出	UINT	插件类属模块物理 ID。 • 如果扩展模块不受支持 ModID = 0xFFF • 如果插件或扩展模块不存在 ModID = 0x0000
VendorID	输出	UINT	插件或扩展类属模块供应商 ID。 对于 Allen Bradley 产品，供应商 ID = 1。 如果插件或扩展模块不存在 VendorID = 0x0000
ProductType	输出	UINT	插件或扩展类属模块产品类型。 如果插件或扩展模块不存在 ProductType = 0x0000
ProductCode	输出	UINT	插件或扩展类属模块产品代码。 如果插件或扩展模块不存在 ProductCode = 0x0000
ModRevision	输出	UINT	插件或扩展类属模块版本信息。 如果插件或扩展模块不存在 ModRevision = 0x0000
ENO	输出	BOOL	启用“输出”。 仅适用于梯形图编程。梯形图会自动添加 ENO 输出。

PLUGIN_INFO 功能块图示例



PLUGIN_INFO 梯形图示例



PLUGIN_INFO 结构化文本示例

```
PLUGIN_INFO_1(
```

void **PLUGIN_INFO_1**(BOOL **Enable**, UINT **SlotID**)
类型： PLUGIN_INFO, 从类属插件模块中获取模块信息。

```
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;
```

结果



PLUGIN_READ（读取插件）

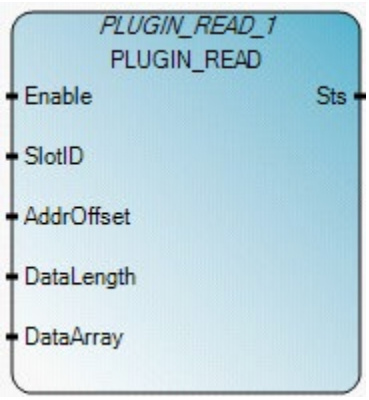
从类属插件模块读取数据。

操作详细信息：

- 除 2080-MEMBAK-RTC 模块之外的任何插件模块。
- 如果插件类属模块不存在，则所有值恢复为零 (0)。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。

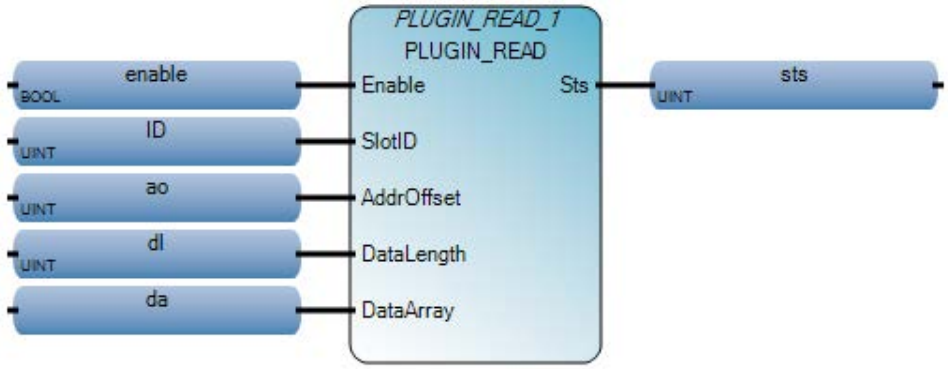


使用此表有助于确定该指令的参数值。

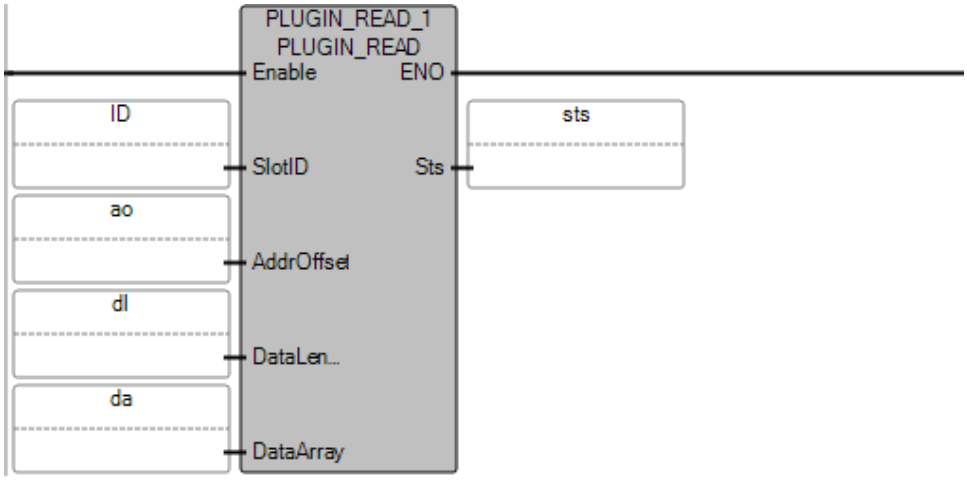
参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行 UPM 读取。 FALSE - 不执行读取操作，并且数据数组中的数据无效。
SlotID	输入	UINT	插件插槽编号。 插槽 ID = 1、2、3、4、5（从最左侧的插槽开始，第一个插槽的 ID = 1）。
Offset	输入	UINT	从插件类属模块的首个字节中计算，要读取的首个数据的地址偏移。
DataLength	输入	UINT	要读取的字节数量。
DataArray	输入	USINT	用于存储从插件类属模块中读取的数据的数组。

参数	参数类型	数据类型	描述
Sts	输出	UINT	PLUGIN_READ 的状态代码。 状态 (Sts) 代码： • 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。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

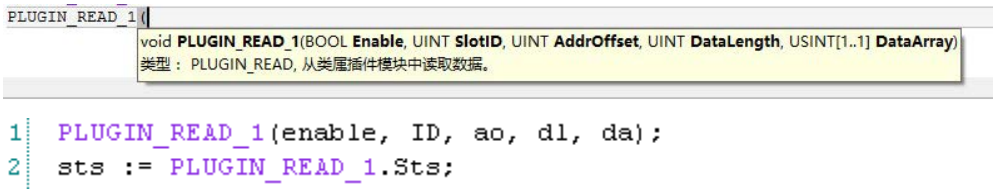
PLUGIN_READ 功能块图示例



PLUGIN_READ 梯形图示例



PLUGIN_READ 结构化文本示例



PLUGIN_RESET（复位插件）

复位除 2080-MEMBAK-RTC 模块外的任何插件类模块硬件。重置硬件后，插件类属模块即可进行配置和操作。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

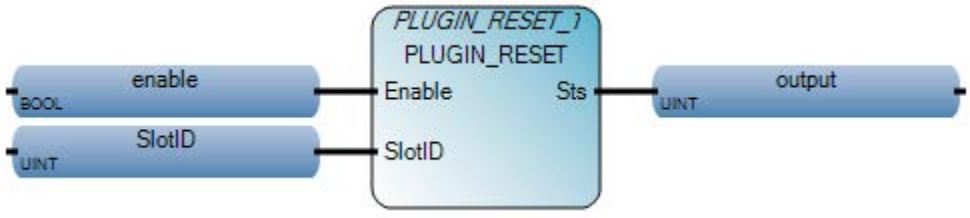
该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



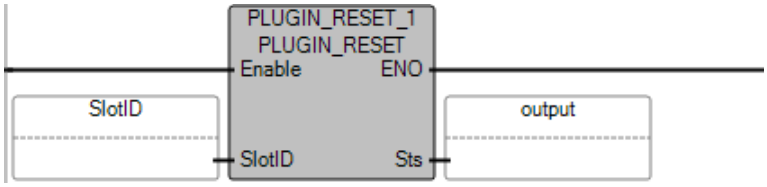
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行插件复位。 FALSE - 不执行复位操作。
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	PLUGIN_RESET 状态代码。 状态 (Sts) 代码： • 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 - 由于插件类模块无效，插件操作失败。 -2080-MOT-HSC 模块配置处于高速计数器指令模式。 • 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。
ENO	输出	BOOL	启用“输出”。 仅适用于梯形图编程。

PLUGIN_RESET 功能块图示例



PLUGIN_RESET 梯形图示例



PLUGIN_RESET 结构化文本示例



结果

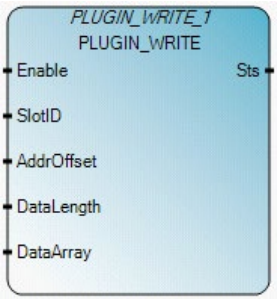


PLUGIN_WRITE（写入插件）

将大量数据写入任何插件类模块硬件（除 2080-MEMBAK-RTC 模块外）。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。

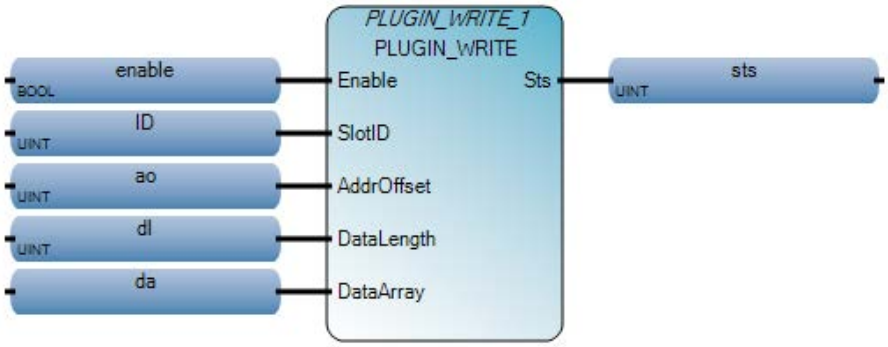


使用此表有助于确定该指令的参数值。

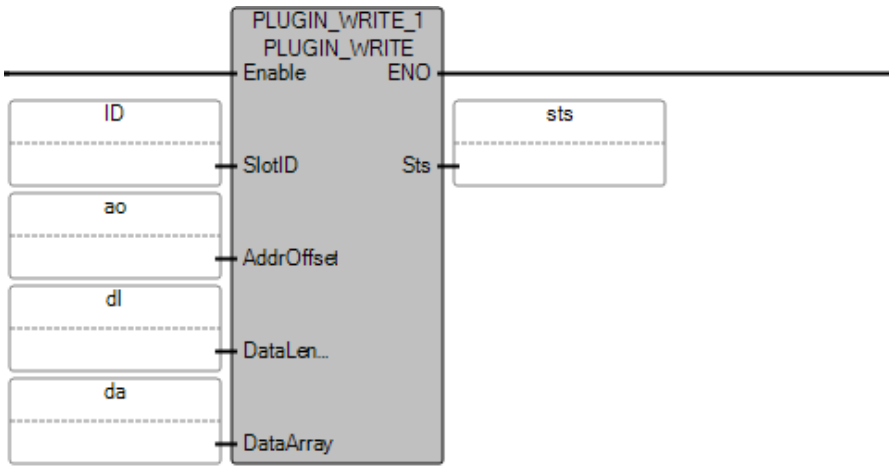
参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行插件写入。 FALSE - 不执行数据写入操作。

参数	参数类型	数据类型	描述
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）。
AddrOffset	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	PLUGIN_WRITE 状态代码。 状态 (Sts) 代码： • 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。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

PLUGIN_WRITE 功能块图示例



PLUGIN_WRITE 梯形图示例



PLUGIN_WRITE 结构化文本示例



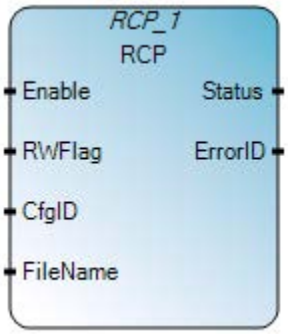

```
1  PLUGIN_WRITE_1(enable, ID, ao, dl, da);
2  sts := PLUGIN_WRITE_1.Sts;
```

RCP（配方）

从配方数据文件（驻留在 SD 卡的配方数据文件文件夹中）读取变量的数据值，并将值更新到运行时引擎。通过运行时引擎将变量值写入 SD 卡上的配方数据文件。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。



使用此表有助于确定该指令的参数值。

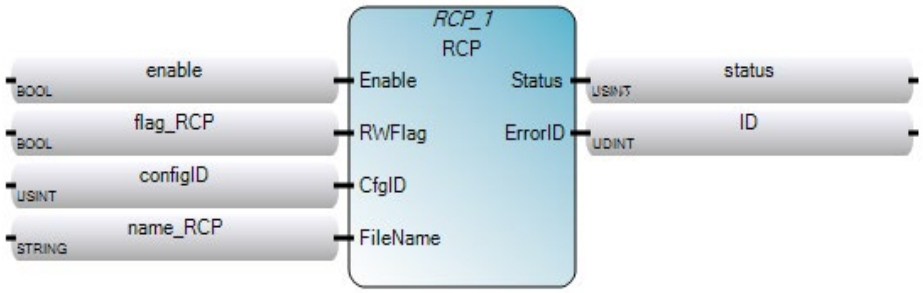
参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用配方读取/写入指令块。 TRUE - 检测到上升沿，执行配方指令，前提是上一个操作已完成。 FALSE - 未检测到上升沿，不执行配方指令。
RWFlag	输入	BOOL	TRUE - RWFlag（写入操作）。配方通过运行时引擎将变量值写入 SD 卡上的配方数据文件。 FALSE - RWFlag（读取操作）。配方从 SD 卡读取变量的值，并将对应变量的值更新到运行时引擎。
CfgID	输入	USINT	配方配置 VA ID 编号 1-10。
FileName	输入	STRING	配方数据文件名称（最大长度 30 个字符）。
Status	输出	USINT	配方指令块当前状态。 RCP 状态代码： • 0 空闲 • 1 执行 • 2 成功，完成且没有出现错误。 • 3 错误，完成但出现错误。
ErrorID	输出	UDINT	数字 RCP 错误代码。 这些定义在 RCP 错误代码中定义。

RCP 错误代码

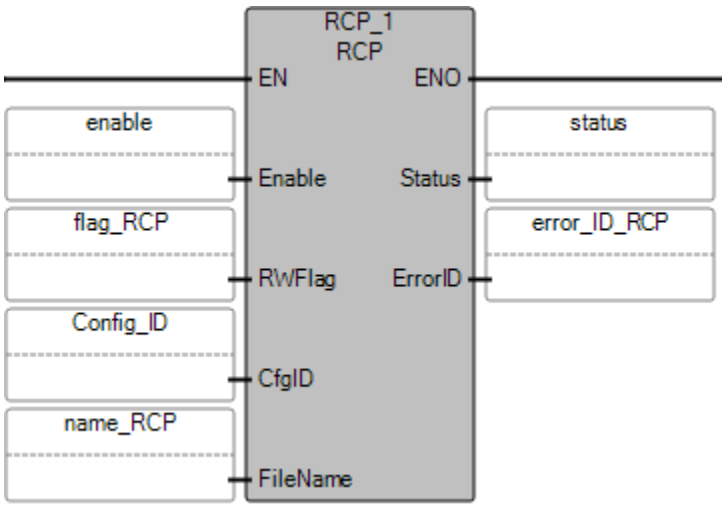
错误代码	错误名称
0	RCP_ERR_NONE
1	RCP_ERR_NO_SDCARD

错误代码	错误名称
2	RCP_ERR_DATAFILE_FULL
3	RCP_ERR_DATAFILE_ACCESS SD 卡被识别为： • 已损坏。 • 已满。 • 只读。
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)。

RCP 功能块图示例



RCP 梯形图示例



RCP 结构化文本示例




```
1  RCP_1 (Enable, RWFlag, CfgID, FileName)
2  status := RCP_1.Status
3  error_ID_RCP := RCP_1.ErrorID
```

RTC_READ（读取实时时钟）

读取实时时钟 (RTC) 模块信息。

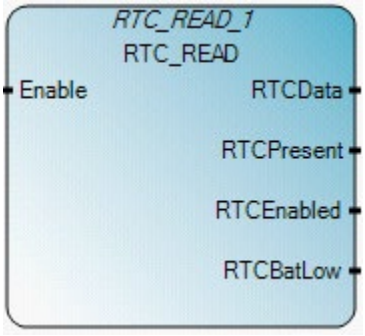
操作详细信息：

- 采用嵌入式 RTC 的 Micro810 或 Micro820 控制器：
  - RTCBatLow 始终设置为零 (0)。
  - RTCEnabled 始终设置为一 (1)。
- 如果由于失去电源而使嵌入式 RTC 丢失其电荷/内存:
  - RTCData 设置为 2000/1/1/0/0/0。
  - RTCEnabled 设置为一 (1)。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行 RTC 信息读取。 FALSE - 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。 使用 RTC 数据类型定义 RTCData 输出。
RTCPresent	输出	BOOL	TRUE - 已使用“自由运行”时钟，或已插入 RTC 硬件。 FALSE - 未使用“自由运行”时钟，或未插入 RTC 硬件。
RTCEnabled	输出	BOOL	TRUE - 已使用“自由运行”时钟，或已启用 RTC 硬件（计时）。 FALSE - 未使用“自由运行”时钟，已禁用 RTC 硬件（不计时）。

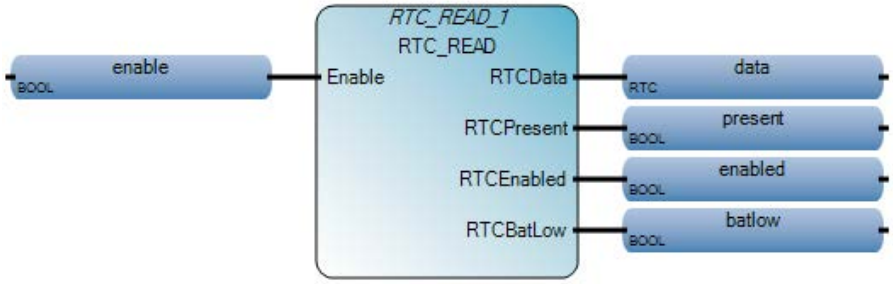
参数	参数类型	数据类型	描述
RTCBatLow	输出	BOOL	TRUE - RTC 电池电量低。 FALSE - RTC 电池电量不低。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

RTC 数据类型

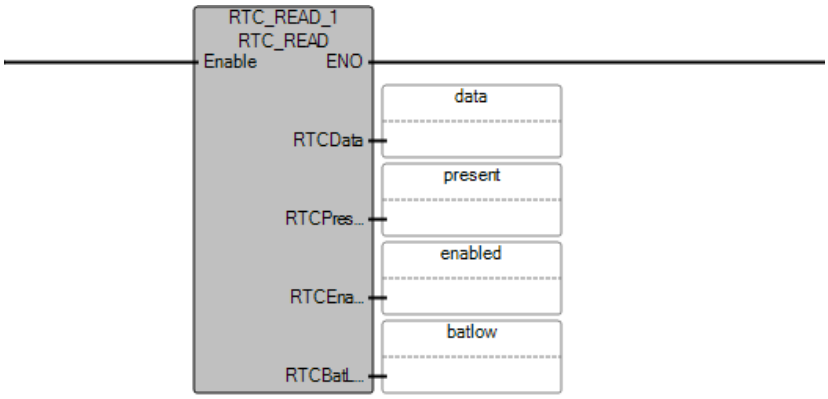
使用此表帮助确定 RTC 数据类型的参数值。

参数	数据类型	描述
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。

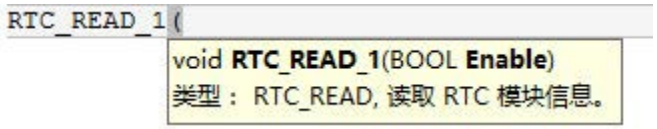
RTC_READ 功能块图示例



RTC_READ 梯形图示例



RTC_READ 结构化文本示例




```
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;
```

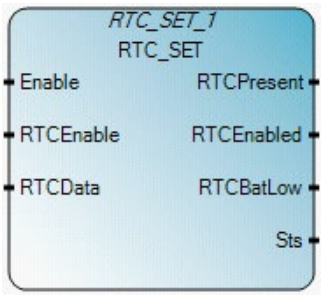
RTC_SET（设置实时时钟）

将 RTC（实时时钟）数据设置为 RTC 模块信息。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。



使用此表有助于确定该指令的参数值。

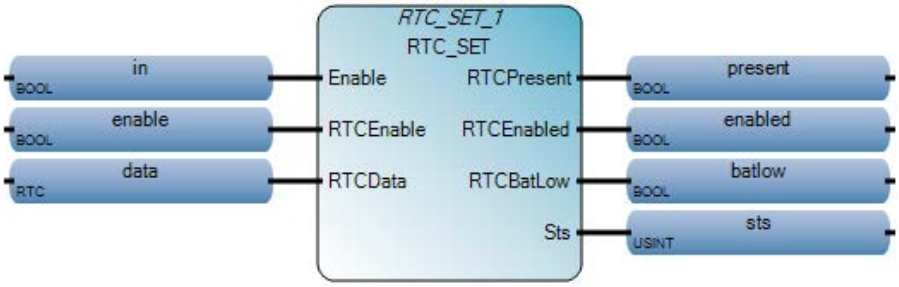
参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 使用输入的 RTC 信息执行 RTC_SET。在更新 RTC 时通常仅为 1 次程序扫描执行。 FALSE - 不执行 RTC_SET。设置为 FALSE 以正常操作 RTC。
RTCEnable	输入	BOOL	TRUE - 启用 RTC 及指定的 RTC 数据。 FALSE - 禁用 RTC。
RTCData	输入	RTC	RTC 数据信息：RTC 数据类型中定义的 yy/mm/dd、hh/mm/ss、星期。 当 RTCEnable = 0 时会忽略 RTCData。
RTCPresent	输出	BOOL	TRUE - 已使用“自由运行”时钟，或已插入 RTC 硬件。 FALSE - 未使用“自由运行”时钟，或未插入 RTC 硬件。
RTCEnabled	输出	BOOL	TRUE - 已使用“自由运行”时钟，或已启用 RTC 硬件（计时）。 FALSE - 未使用“自由运行”时钟，或已禁用 RTC 硬件（不计时）。
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。 RTC_Set 状态 (Sts) 值： • 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。

RTC 数据类型

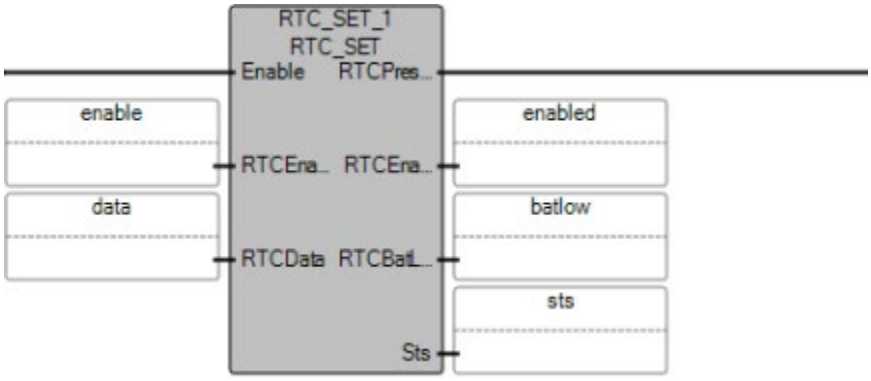
使用此表帮助确定 RTC 数据类型的参数值。

参数	数据类型	描述
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。

RTC_SET 功能块图示例



RTC_SET 梯形图示例



RTC_SET 结构化文本示例




```
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;
```

SYS_INFO

读取 Micro800 控制器的状态数据块。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

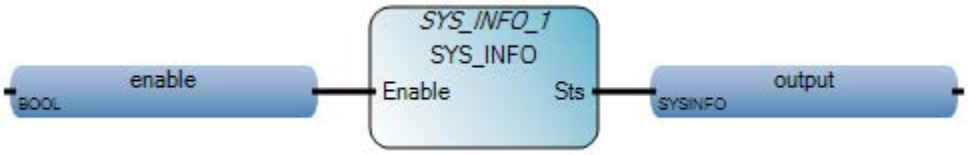
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



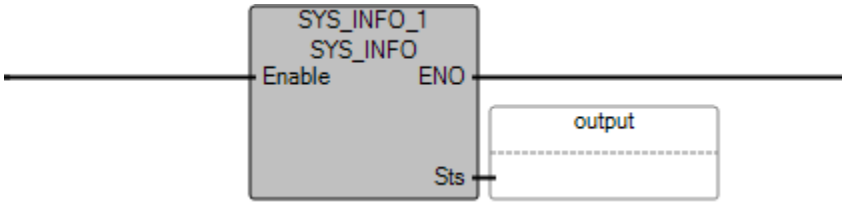
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行读取操作。 FALSE - 不执行功能。
Sts	输出	SYSINFO	系统状态数据块。 Sts 输出在 SYS_INFO 数据类型 参考页数 363 中定义。
ENO	输出	BOOL	启用“输出”。 仅适用于梯形图编程。

SYS_INFO 功能块图示例



SYS_INFO 梯形图示例



SYS_INFO 结构化文本示例

```
SYS_INFO_1(
```

void **SYS_INFO_1**(BOOL **Enable**)
类型： SYS_INFO, 读取 Micro800 系统状态。

```
1  SYS_INFO_1(enable);
2  output := SYS_INFO_1.Sts;
```

结果



SYS_INFO 数据类型

下表描述了 SYSINFO 数据类型。

参数	数据类型	描述
BootMajRev	UINT	启动主要版本。
BootMinRev	UINT	启动次要版本。
Operating System Series	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。

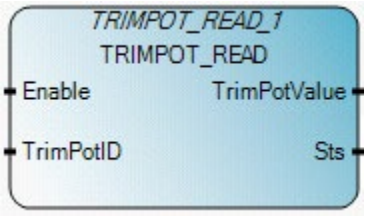
参数	数据类型	描述
OSMinRev	UINT	操作系统次要版本。
ModeBehaviour	BOOL	模式行为（TRUE：通电后转至 RUN）。
FaultOverride	BOOL	故障覆盖（TRUE：通电后覆盖错误）。
StrtUpProtect	BOOL	启动保护（TRUE：通电后运行启动保护程序）。 对于未来版本。
MajErrHalted	BOOL	主要错误停止（TRUE：主要错误停止）。
MajErrCode	UINT	主要错误代码。
MajErrUFR	BOOL	用户错误例程期间的主要错误。 对于未来版本。
UFRPouNum	UINT	用户错误例程程序编号。
MMLoadAlways	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。 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）。

TRIMPOT_READ（读取微调电位计）

从特定微调电位中读取微调电位计数值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850 和 Micro870 控制器。

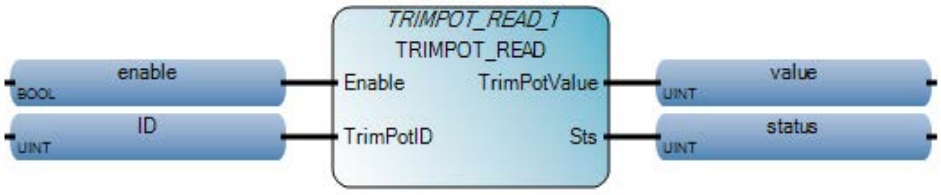


使用此表有助于确定该指令的参数值。

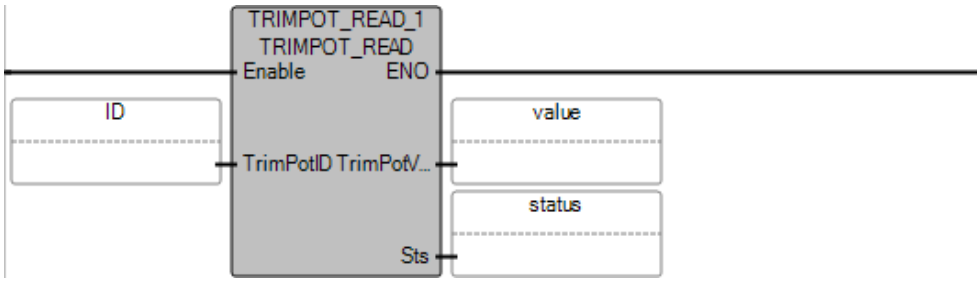
参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行微调电位读取。 FALSE - 不执行读取操作，且输出微调电位计数值无效。
TrimPotID	输入	UINT	要读取的微调电位的 ID。TrimPotID 在微调电位 ID 定义 参考页数 365 中定义。
TrimPotValue	输出	UINT	当前微调电位计数值。
Sts	输出	UINT	微调电位计读取操作状态。 TRIMPOT 状态 (Sts) 代码： • 0x00 - 未启用功能块（无读取/写入操作）。

参数	参数类型	数据类型	描述
			• 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 - 写入操作由于值超范围而失败。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

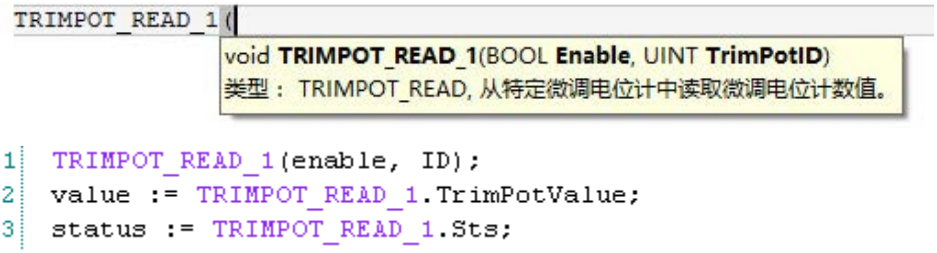
TRIMPOT 功能块图示例



TRIMPOT 梯形图示例



TRIMPOT 结构化文本示例



微调电位 ID 定义

下表描述了在 TRIMPOT_read 指令 参考页数 364 中使用的微调电位 ID 定义。

输出选择	位	描述
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	描述
	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。

中断指令

使用中断指令向处理器发出信号，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。

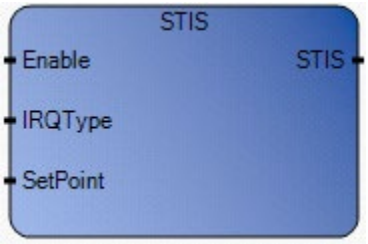
功能	描述
STIS 参考页数 367	从控制程序启动选定的定时用户中断 (STI) 计时器，而不是自动启动。
UIC 参考页数 369	清除选定用户中断的丢失位。
UID 参考页数 370	禁用特定用户中断。
UIE 参考页数 372	启用特定用户输入。
UIF 参考页数 374	刷新或删除挂起的用户输入。

STIS（选择计时启动）

从控制程序启动可选定时用户中断 (STI) 计时器，而不是自动启动。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

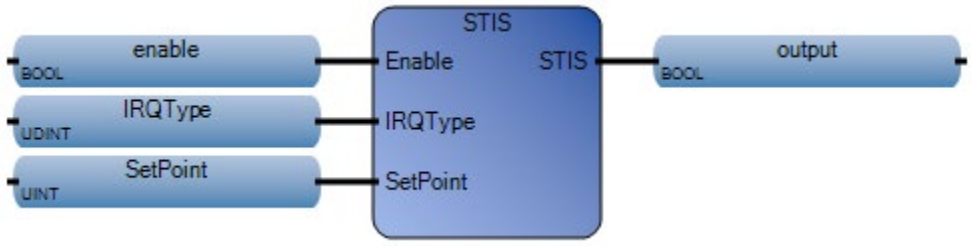
该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



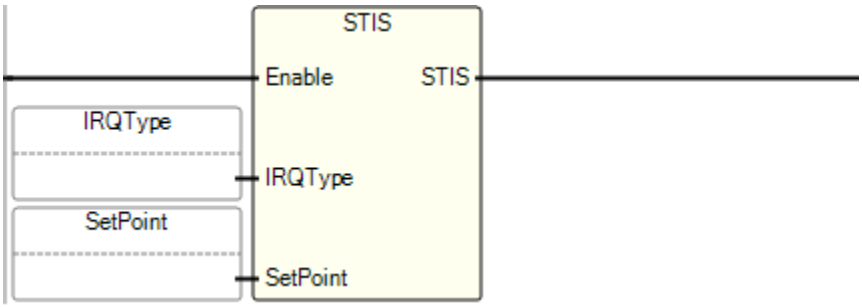
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令。 TRUE - 从控制程序启动 STI 计时器。 FALSE - 不执行功能。
IRQType	输入	UDINT	使用 STI 定义字。 - IRQ_STI0 - IRQ_STI1 - IRQ_STI2 - IRQ_STI3
SetPoint	输入	UINT	执行可选计时中断之前必须经过的时间长度（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 功能块图示例



STIS 梯形图示例



STIS 结构化文本示例




```
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

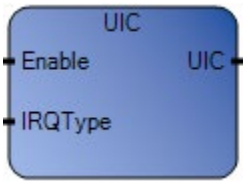


UIC（清除中断丢失位）

清除选定用户中断的丢失位。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



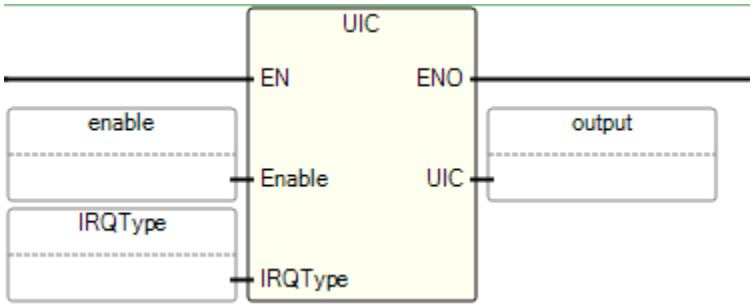
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述	
Enable	输入	BOOL	启用指令。 TRUE - 开始清除位操作。 FALSE - 不执行功能。	
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 功能块图示例



UIC 梯形图示例



UIC 结构化文本示例

```
1  enable := TRUE;
2  IRQType := 2;
3  output := UIC (enable, IRQType);
```

结果



UID（禁用中断）

禁用特定用户中断。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



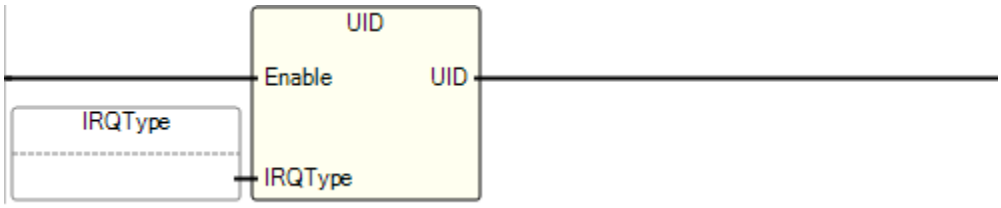
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述	
Enable	输入	BOOL	启用指令。 TRUE - 启动禁用操作。 FALSE - 不执行功能。	
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
UID	输出	BOOL	Rung 状态（与 Enable 相同）。	

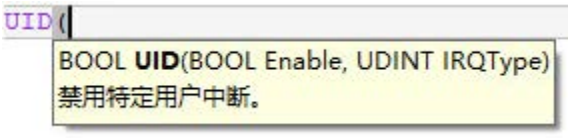
UID 功能块图示例



UID 梯形图示例



UID 结构化文本示例




```
1  enable := TRUE;
2  IRQType := 2;
3  output := UID(enable, IRQType);
```

(* 与之等效的 ST: *)

TESTOUTPUT := UID(TESTENABLE, 2) ;

结果



UIE（启用中断）

启用特定用户中断。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

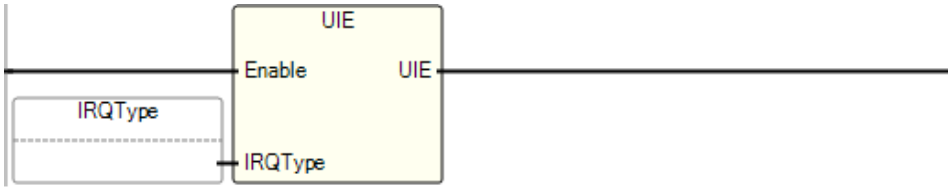
参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令。 TRUE - 启动启用操作。 FALSE - 不执行功能。

参数	参数类型	数据类型	描述	
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
UIE	输出	BOOL	Rung 状态（与 Enable 相同）。	

UIE 功能块图示例



UIE 梯形图示例



UIE 结构化文本示例




```
1  enable := TRUE;
2  IRQType := 2;
3  output := UIE(enable, IRQType);
```

(* 与之等效的 ST: *)

```
TESTOUTPUT := UIE(TESTENABLE, 2) ;
```

结果



UIF（刷新挂起中断）

刷新或删除挂起的用户中断。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



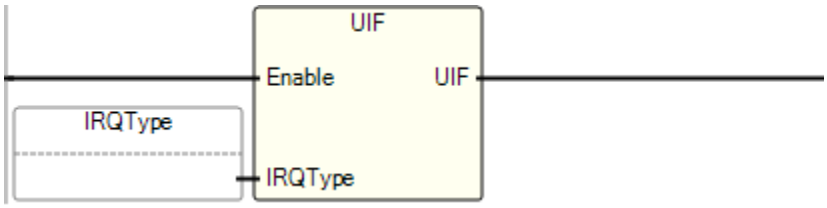
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述	
Enable	输入	BOOL	启用指令。 TRUE - 开始 UIF 操作。 FALSE - 不执行功能。	
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
UIF	输出	BOOL	Rung 状态（与 Enable 相同）。	

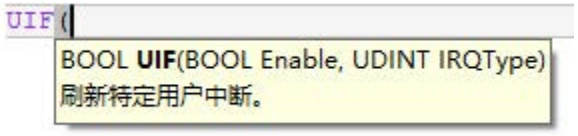
UIF 功能块图示例



UIF 梯形图示例



UIF 结构化文本示例




```
1  enable := TRUE;
2  IRQType := 2;
3  output := UIF(enable, IRQType);
```

(* 与之等效的 ST: *)

TESTOUTPUT := UIF(TESTENABLE, 2) ;

结果



运动控制指令

使用运动控制指令对特定轴的运动进行编程和设计。运动控制要求 Connected Components Workbench 2.0 或更高版本。



提示：

- 管理指令支持 PTO 运动和反馈运动。
- 支持 FBAxis 的运动控制指令有：MC_ReadActualPosition 和 MC_ReadActualVelocity。

指令	描述
管理	
MC_AbortTrigger 参考页数 386	取消与触发事件相连的运动功能块。
MC_Power 参考页数 406	控制功率（打开或关闭）。
MC_ReadAxisError 参考页数 415	读取与运动控制指令块无关的轴错误。
MC_ReadBoolParameter 参考页数 419	返回特定于供应商的类型为 BOOL 的参数的值。
MC_ReadParameter 参考页数 422	返回特定于供应商的类型为实型的参数的值。
MC_ReadStatus 参考页数 424	返回与当前正在进行中的运动相关的轴的状态。
MC_Reset 参考页数 428	通过复位所有内部轴相关错误将轴状态从 ErrorStop 转换为 StandStill。
MC_SetPosition 参考页数 431	通过控制实际位置来转移轴坐标系统。
MC_TouchProbe 参考页数 436	在触发事件中记录轴位置。
MC_WriteBoolParameter 参考页数 440	修改特定于供应商的类型为 BOOL 的参数的值。
MC_WriteParameter 参考页数 443	修改特定于供应商的类型为 REAL 的参数的值。
运动	
MC_Halt 参考页数 389	命令受控制的运动在正常操作条件下停止。
MC_Home 参考页数 391	命令轴执行 *<search home>* 序列。
MC_MoveAbsolute 参考页数 395	命令受控制的运动到指定的绝对位置。
MC_MoveRelative 参考页数 398	控制伺服以相对值运动。
MC_MoveVelocity 参考页数 402	控制伺服以速度值运动。
MC_ReadActualPosition 参考页数 409	返回反馈轴的实际位置。 要求 Connected Components Workbench 8 或更高版本。
MC_ReadActualVelocity 参考页数 412	返回反馈轴的实际速度。 要求 Connected Components Workbench 8 或更高版本。
MC_Stop 参考页数 433	命令受控制的运动停止并将轴状态转为 Stopping。

运动控制功能块的一般规则

Micro800 运动控制功能块 参考页数 377 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。 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。
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。

规则适用于	规则
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。
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覆盖当前执行的功能块。
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错误。 • ErrorID - 错误编号。 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 的下降沿复位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。
	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 • 变频器

规则适用于	规则
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。

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 380

运动控制功能块参数编号 参考页数 382

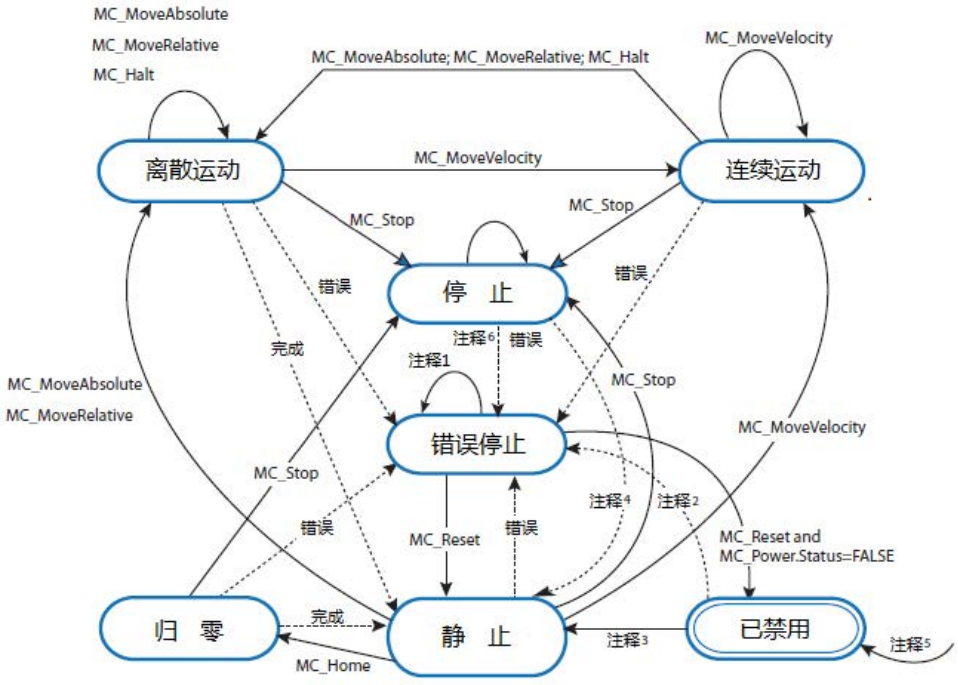
运动控制功能块错误 ID 参考页数 383

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

否	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
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

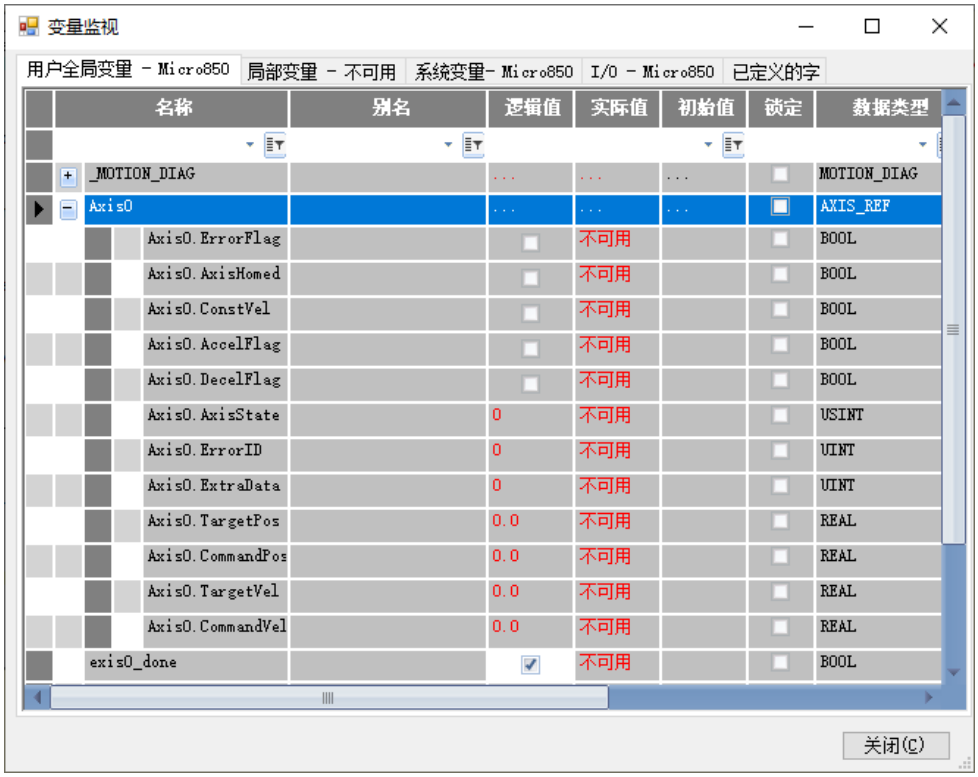
您可以使用“轴监视器”功能监视轴状态。下表标识了用于定义每个预定义轴状态的值。

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	原位
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。



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	描述
1	Commanded Position	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度。
13	MaxAccelerationAppl	REAL	R/W	应用程序中允许的最大轴加速度。
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。

参数编号	参数名称	数据类型	R/W	描述
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	Duty Cycle	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。

运动控制功能块错误 ID

当运动控制功能块以错误结束且轴处于 ErrorStop 状态时，使用 MC_Reset 功能块或 MC_Power Off/On 和 MC_Reset 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。

使用此表帮助确定运动控制功能块的错误。

值	MACRO ID	描述
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 参考页数 380。
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。
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。

值	MACRO ID	描述
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取消功能块。 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，因此该功能块取消。 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 累加器逻辑限制而取消功能块。 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统。
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。

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，则轴转换为“错误”状态，并在轴的 AXIS_REF 数据上设置相应的 ErrorID 元素。

在以下情况下，轴可能不会转换为“错误”状态，并且用户应用程序能够在轴状态更改后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，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	描述
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。
Error Flag	BOOL	表示错误是否在轴中存在。 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 将轴复位。
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。

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。

重要事项 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。

参数	数据类型	描述
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。

参数	数据类型	描述
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。

轴变量

周变量用来控制给定的运动控制轴位置、速度、加速度和错误。

要将变量分配给轴输出参数：

- 在功能块图中

  为了方便，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，在连接至控制器的情况下监视软件中的 AXIS_REF 或 FBAXIS_REF 变量。您还可以通过各种通信通道远程监视 AXIS_REF 变量。

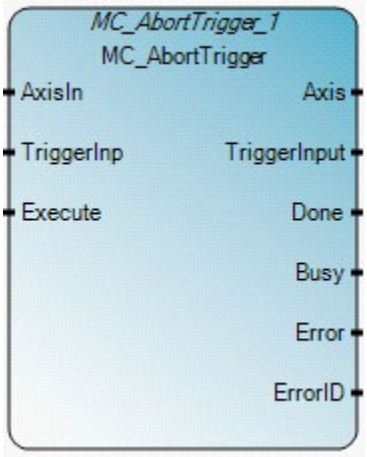
MC_AbortTrigger（运动控制中止触发器）

取消与触发事件相连的运动功能块。

MC_AbortTrigger 只有当分配到由 MC_TouchProbe 控制的轴时才会执行 。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

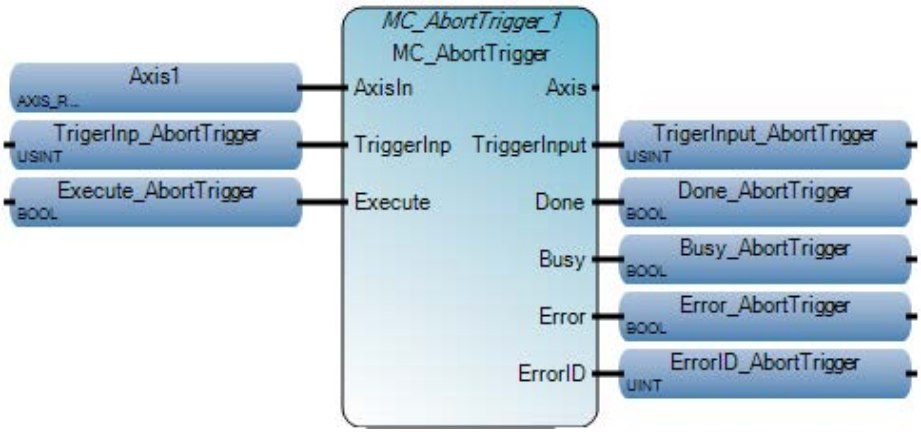
该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。



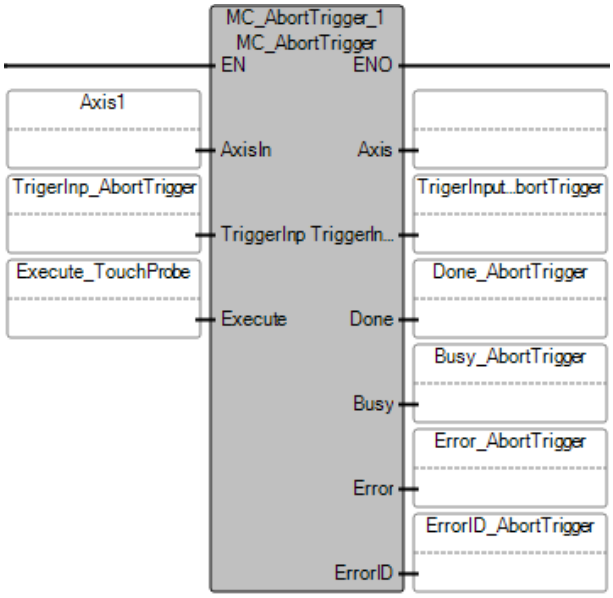
使用此表有助于确定该指令的参数值。

参数	参数	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_AbortTrigger 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF 参考页数 384 FB_AXIS_REF 参考页数 385	使用 AXIS_REF 数据类型定义 AxisIn 参数。 如果轴为 FB_Axis（反馈轴），使用 FB_AXIS_REF 数据类型定义 AxisIn 参数。
TriggerInp	输入	USINT	此参数将被忽略。
Execute	输入	BOOL	当为 TRUE 时，在上升沿取消触发事件。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。
TriggerInput	输出	USINT	此参数将被忽略。
Done	输出	BOOL	TRUE - 已取消触发事件。
Busy	输出	BOOL	TRUE - 功能块未完成。
Error	输出	BOOL	TRUE - 检测到错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 中定义该指令的错误。

MC_AbortTrigger 功能块图示例



MC_AbortTrigger 梯形图示例

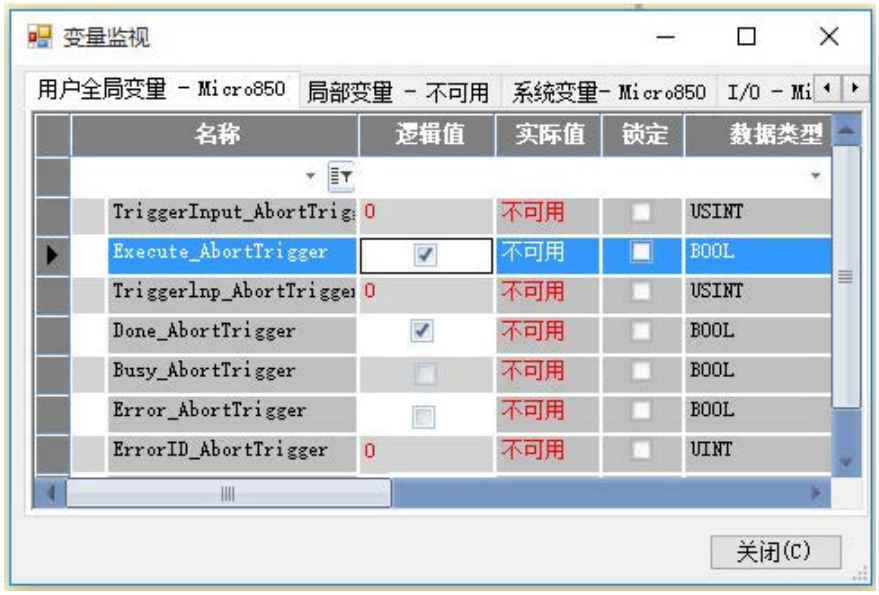


MC_AbortTrigger 结构化文本示例




```
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;
```

结果



MC_Halt（运动控制停止）

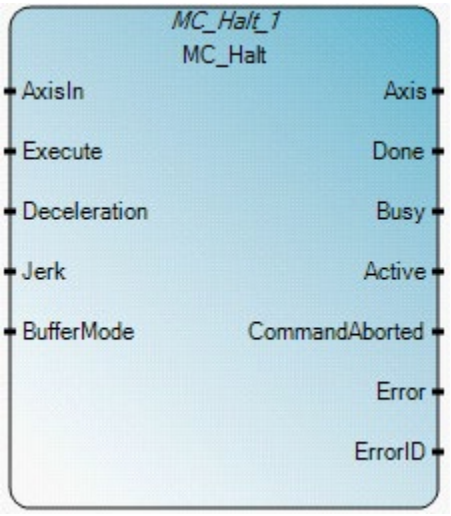
命令受控制的运动在正常操作条件下停止。

操作详细信息：

- 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。
  - 在轴减速期间可以执行其他运动命令，而这将覆盖 MC_Halt。
  - 如果在轴正处于 Homing 状态时发布 MC_Halt，则该指令块会报告错误，而归位进程不会中断。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。

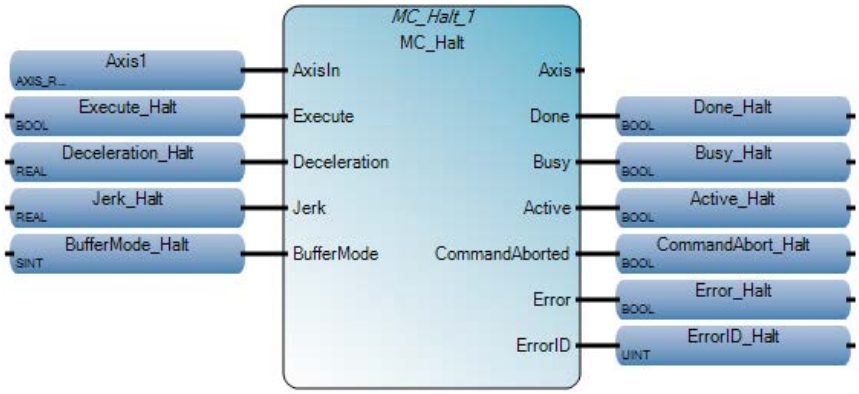


使用此表有助于确定该指令的参数值。

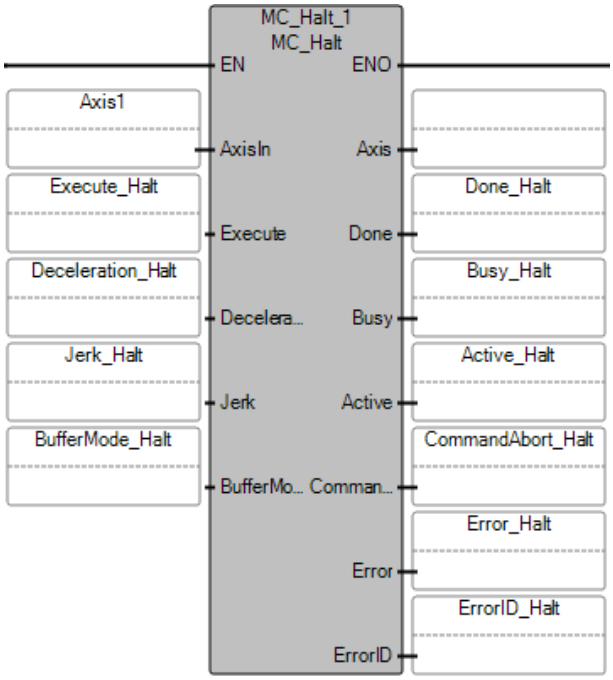
	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_Halt 计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
AxisIn	输入	AXIS_REF	使用 AXIS_REF 数据类型参数定义 AxisIn。
Execute	输入	BOOL	指示何时开始运动。 TRUE - 在上升沿开始运动。 FALSE - 不开始运动。 在归位期间执行 MC_Halt 时，MC_Halt 将被设置为 MC_FB_ERR_STATE，并且归位进程继续。
Deceleration	输入	REAL	减速的值（始终为正值）（减少电机的能量）。 如果 Deceleration <= 0 且轴未处于 Standstill 状态，则 MC_Halt 将被设置为 MC_FB_ERR_RANGE。
Jerk	输入	REAL	加加速度的值（始终为正值）。 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”。 仅适用于梯形图编程。

	参数类型	数据类型	描述
Axis	输出	AXIS_REF 参考页数 384	梯形图编程中的轴输出为只读。
Done	输出	BOOL	达到零速率。
Busy	输出	BOOL	指令块未完成。
Active	输出	BOOL	表示指令块可控制轴。
CommandAborted	输出	BOOL	命令已被其他命令或 Error Stop 覆盖。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。

MC_Halt 功能块图示例



MC_Halt 梯形图示例



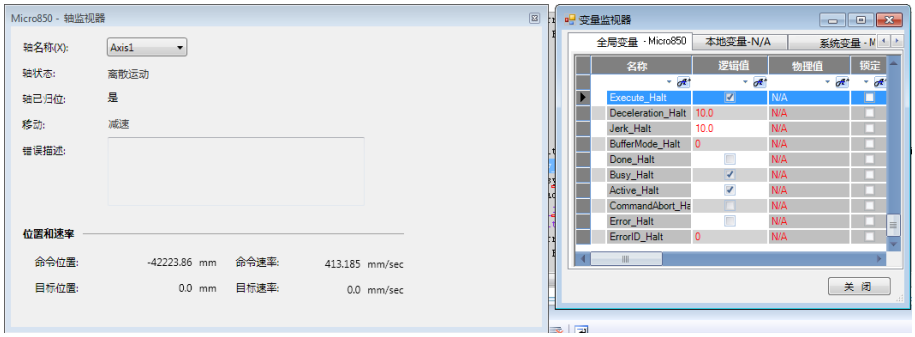
MC_Halt 结构化文本示例

```
MC_Halt_1(
```

void **MC_Halt_1**(AXIS_REF **AxisIn**, BOOL **Execute**, REAL **Deceleration**, REAL **Jerk**, SINT **BufferMode**)
类型： MC_Halt, 命令受控制的运动停止。轴移动到状态"离散运动"直到速率变为零。

```
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;
CommandAbort_Halt := MC_Halt_1.CommandAborted;
Error_Halt := MC_Halt_1.Error;
ErrorID_Halt := MC_Halt_1.ErrorID;
```

结果



MC_Home（运动控制原位）

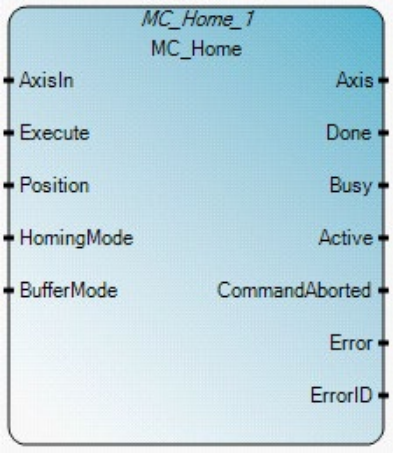
命令轴执行 *<search home>* 序列。此序列的详细信息依赖于制造商，可根据轴参数进行设置。Position 输入用于在检测到参考信号且达到配置的 Home 偏移时设置绝对位置。

操作详细信息：

- 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 功能块只能使用 MC_Stop 或 MC_Power 功能块取消 。如果在完成之前取消，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。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。



使用此表有助于确定该指令的参数值。

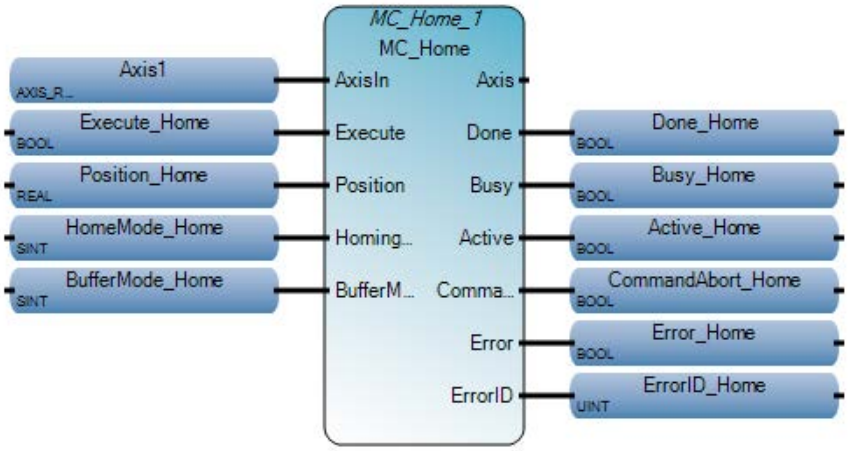
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_Home 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF	使用 AXIS_REF 数据类型参数定义 AxisIn。
Execute	输入	BOOL	指示何时开始运动。 TRUE - 在上升沿开始运动。 FALSE - 不开始运动。
Position	输入	REAL	当检测到参考信号且达到配置的主偏移时设置绝对位置。 在位置从用户位置转换到 PTO 脉冲时，此输入的值范围为 -0x40000000 - 0x40000000 实际值脉冲。设置软限位内的位置值。 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。
BufferMode	输入	SINT	未使用。该模式始终为 mcAborting。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF 参考页数 384	梯形图编程中的轴输出为只读。
Done	输出	BOOL	TRUE - 归位操作会成功完成且轴状态设置为 Standstill。 FALSE - 原位操作正在进行或未完成。
Busy	输出	BOOL	TRUE - 指令块未完成。 FALSE - 指令块已完成。
Active	输出	BOOL	TRUE - 表示指令块可控制轴。
CommandAborted	输出	BOOL	TRUE - 命令已被其他命令或 Error Stop 覆盖。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UNIT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。

HomingMode

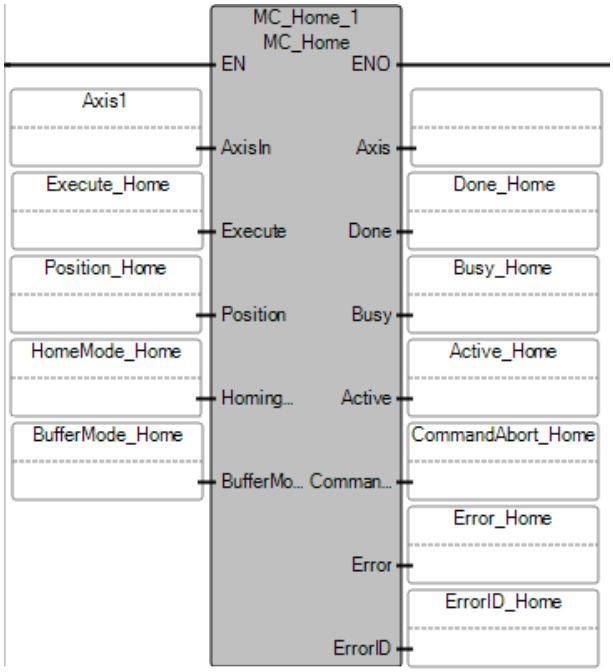
使用此表帮助确定 MC_Home 运动控制指令 参考页数 377 中 HomingMode 参数的值。

值	名称	描述
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

MC_Home 功能块图示例



MC_Home 梯形图示例



MC_Home 结构化文本示例




```
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;
```

结果





MC_MoveAbsolute（运动控制移动绝对值）

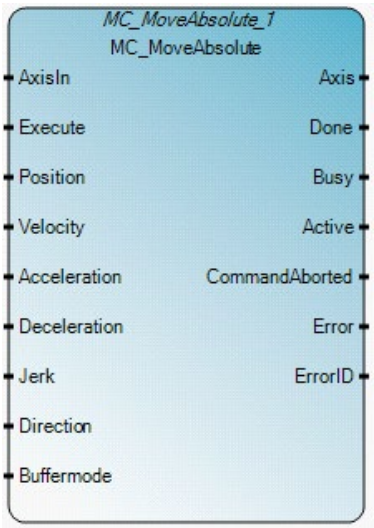
命令受控制的运动到指定的绝对位置。

操作详细信息：

- 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 功能块的输入速度符号。
  - 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覆盖，则 MoveAbsolute 功能块会以零速度完成。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

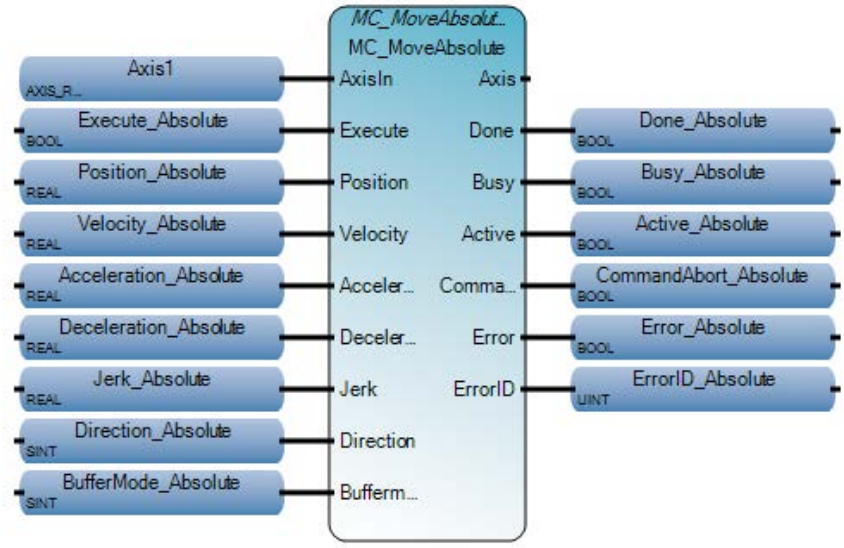
该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。



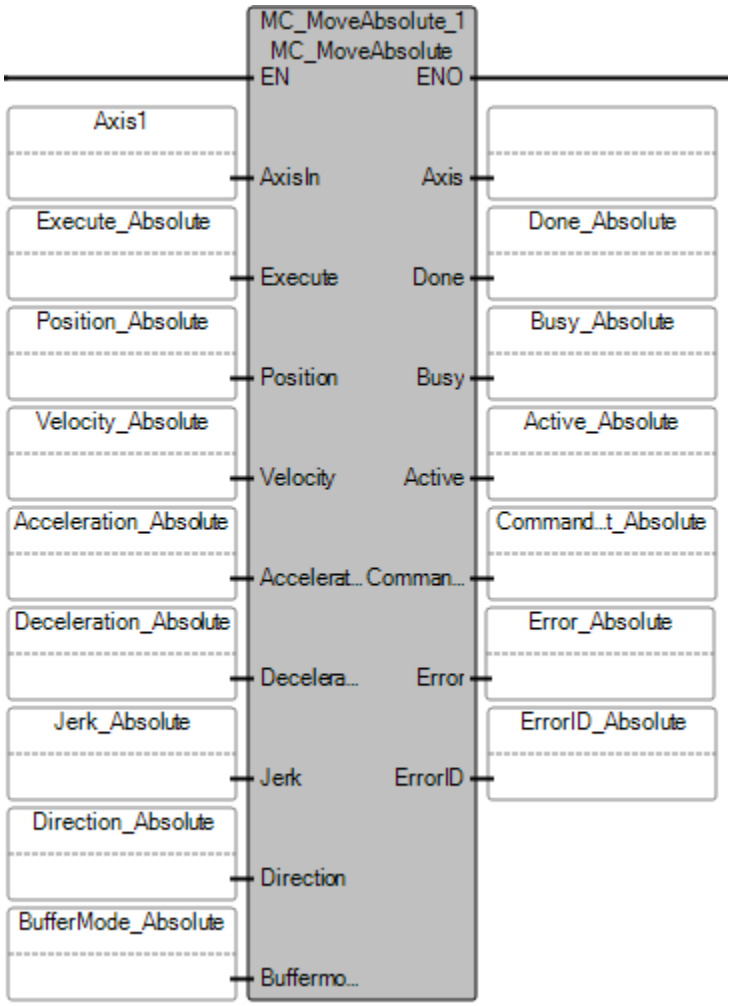
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_MoveAbsolute 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF	使用 AXIS_REF 数据类型参数定义 AxisIn。
Execute	输入	BOOL	指示何时开始运动。 TRUE - 在上升沿开始运动。 FALSE - 不开始运动。 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）。 技术单元在轴的"运动 - 常规"配置页面进行定义。
Velocity	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。
Acceleration	输入	REAL	加速度的值（始终为正值 - 可增加电机的能量）。 用户单位/sec^2
Deceleration	输入	REAL	减速的值（始终为正值 - 可减少电机的能量）。 u/秒 ^{2}
Jerk	输入	REAL	加加速度的值（始终为正值）。 u/秒 ^{3} 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。
Direction	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"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF 参考页数 384	梯形图编程中的轴输出为只读。
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，速度为零（除非操作被取消）。
Busy	输出	BOOL	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 覆盖。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。

MC_MoveAbsolute 功能块图示例



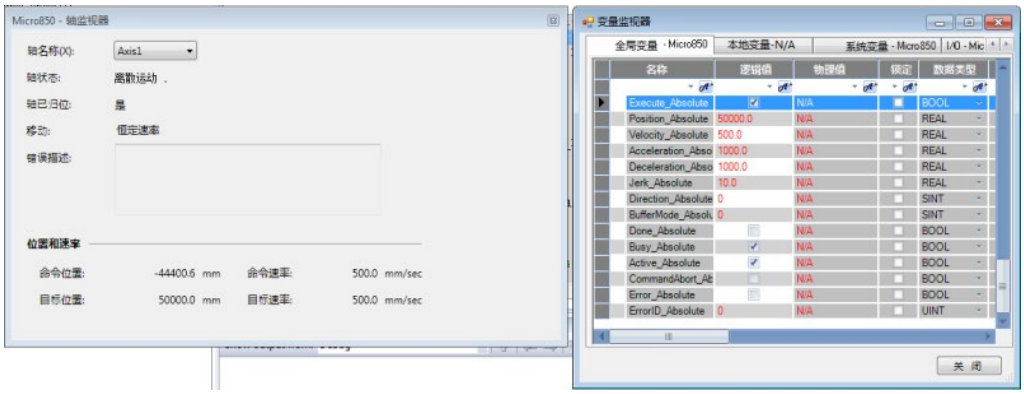
MC_MoveAbsolute 梯形图示例



MC_MoveAbsolute 结构化文本示例

```
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;
```

结果



MC_MoveRelative（运动控制移动相对值）

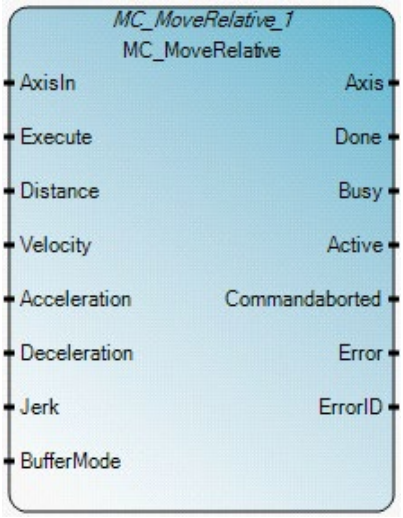
在执行时命令与实际位置相对的指定距离的受控制运动。

操作详细信息：

- 由于 MC_MoveRelative 的运动方向取决于当前位置和目标位置，因此速度的符号被忽略。
- 如果未被其他功能块覆盖，则 MoveRelative 会以零速度完成。
- 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，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 的输入速度符号。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。



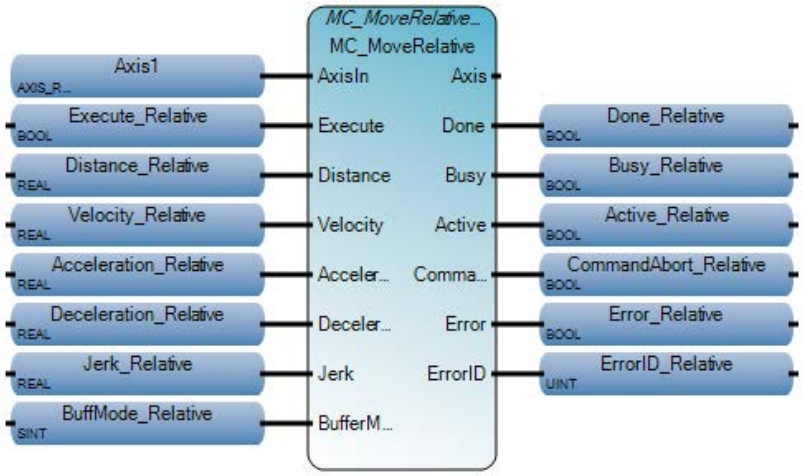
MC_MoveRelative 操作

使用此表有助于确定该指令的参数值。

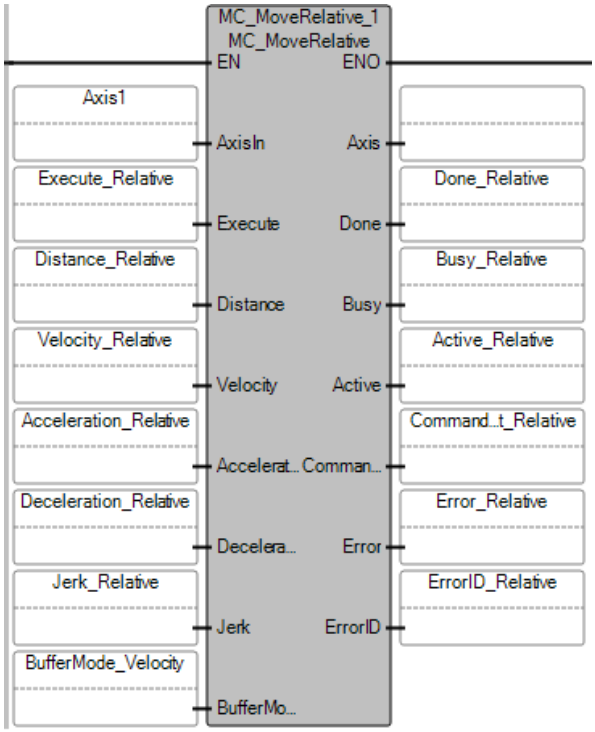
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_MoveRelative 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF 参考页数 384	使用 AXIS_REF 数据类型参数定义 AxisIn。
Execute	输入	BOOL	指示何时开始运动。 TRUE - 在上升沿开始运动。 FALSE - 不开始运动。
Distance	输入	REAL	运动的相对距离（以技术单元 [u] 为单位）。
Velocity	输入	REAL	速率最大值（不一定达到）[u/s]。由于运动方向取决于输入位置，因此速率的符号会被功能块忽略。 当 Jerk = 0 时可能不会达到最大速率。
Acceleration	输入	REAL	加速度的值（可增加电机的能量）[u/s2]
Deceleration	输入	REAL	减速的值（可减少电机的能量）[u/s2]
Jerk	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"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。
Done	输出	BOOL	TRUE - 已达到命令的距离。 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	TRUE - 指令块未完成。 FALSE - 指令块已完成。
Active	输出	BOOL	TRUE - 表示指令块可控制轴。
CommandAborted	输出	BOOL	TRUE - 命令已被其他命令或 Error Stop 覆盖。

参数	参数类型	数据类型	描述
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。

MC_MoveRelative 功能块图示例



MC_MoveRelative 梯形图示例



MC_MoveRelative 结构化文本示例

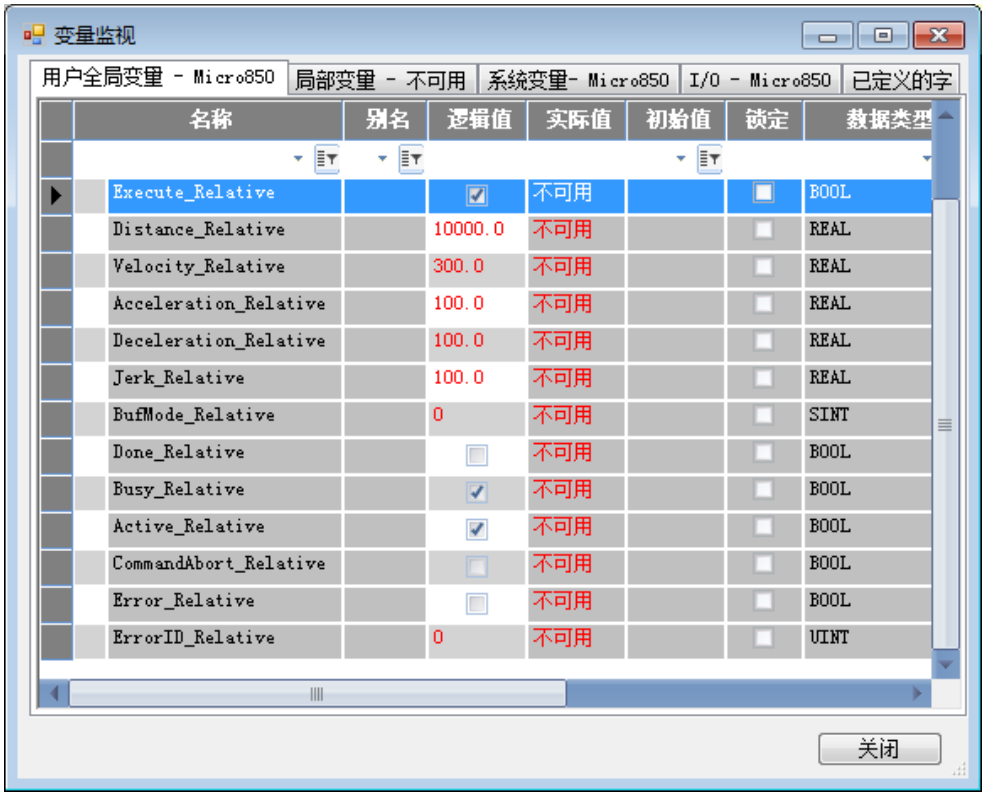
```
MC_MoveRelative_1(
```

```
void MC_MoveRelative_1(AXIS_REF AxisIn, BOOL Execute, REAL Distance, REAL Velocity, REAL Acceleration, REAL Deceleration, REAL Jerk, SINT BufferMode)
类型： MC_MoveRelative, 控制伺服以相对值运动。
```

```
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,BuffMode_Relative);
Done_Relative := MC_MoveRelative_1.Done;
Busy_Relative := MC_MoveRelative_1.Busy;
Active_Relative := MC_MoveRelative_1.Active;
CommandAbort_Relative := MC_MoveRelative_1.Commandaborted;
Error_Relative := MC_MoveRelative_1.Error;
ErrorID_Relative := MC_MoveRelative_1.ErrorID;
```

结果





MC_MoveVelocity（运动控制移动速度）

以指定速率命令从未结束的受控制运动。

操作详细信息：

- 如果 MC_MoveVelocity 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。
  - 轴未处于移动状态，则 MC_MoveVolecity 会报告错误。
- 如果在执行 MC_MoveVelocity 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），且会继续执行。
- 如果轴处于移动状态，且 MC_MoveVelocity 发布与当前运动方向相反（Velocity * Direction 的符号）的运动，则 MC_MoveVelocity会报告错误。
- 一旦已设置信号“InVelocity”，则表示 MC_MoveVelocity 完成。除信号“InVelocity”外，任何其他后续运动事件都不会影响 MC_MoveVelocity 输出。
- 一旦轴的速度达到命令的速度，MC_MoveVelocity 的 InVelocity输出始终为 True，或直到停止 MC_MoveVelocity。
- (Velocity * Direction) 的符号决定 MC_MoveVelocity 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 被其他功能块或运动事件覆盖，或在“执行”的下降沿被中止，则将复位“InVelocity”信号。

- 要停止或更改 MC_MoveVelocity 启动的运动，必须使用其他指令块（包括使用不同参数重新执行 MC_MoveVelocity）来中断或覆盖该指令块。
- 如果发布 MC_MoveVelocity 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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。

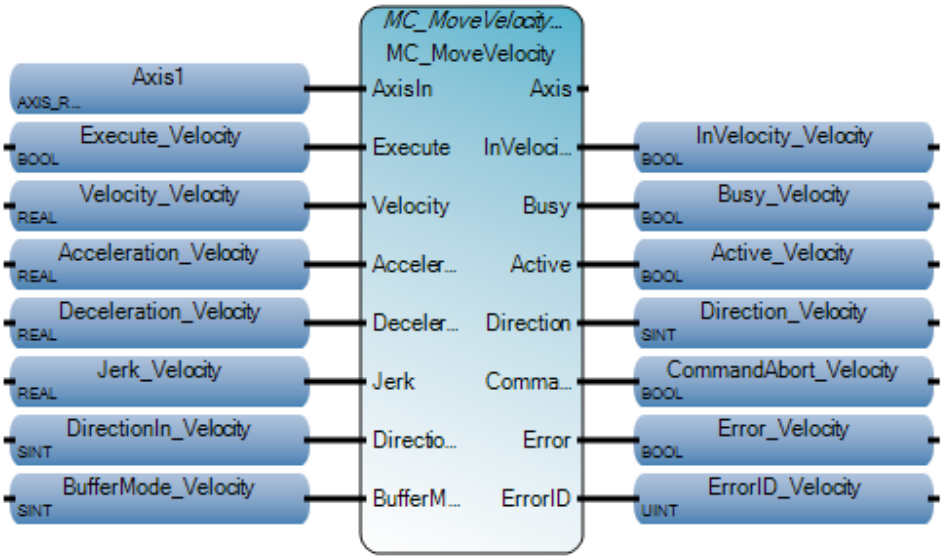


使用此表有助于确定该指令的参数值。

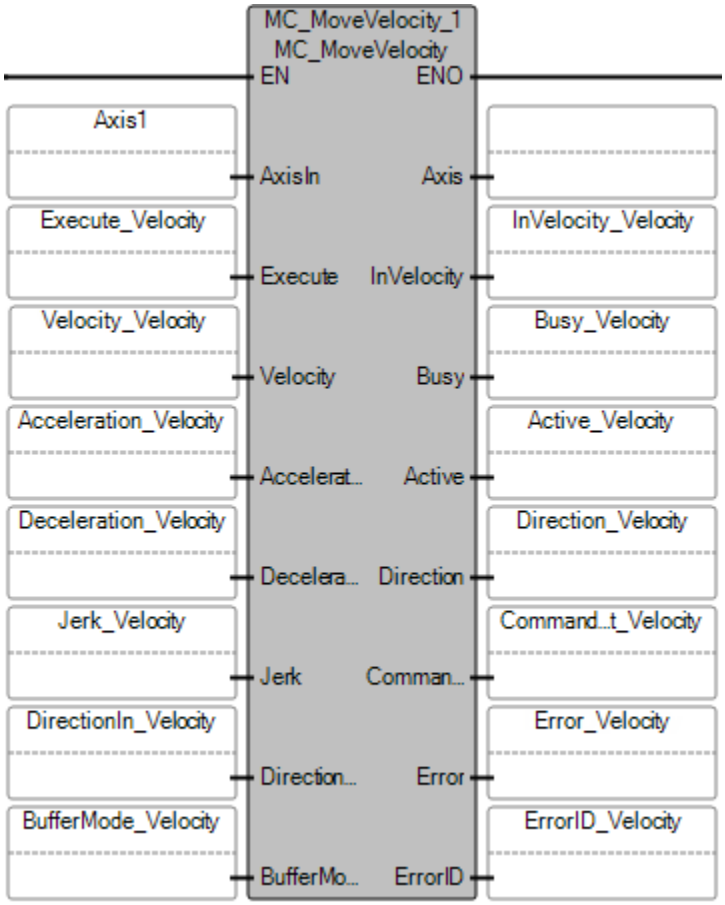
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_MoveVelocity 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF	使用 AXIS_REF 数据类型参数定义 AxisIn。
Execute	输入	BOOL	指示何时开始运动。 TRUE - 在上升沿开始运动。 FALSE - 不开始运动。
Velocity	输入	REAL	速率最大值 [u/s]。
Acceleration	输入	REAL	加速度的值（可增加电机的能量）[u/s^2]
Deceleration	输入	REAL	减速的值（可减少电机的能量）[u/s^2]
Jerk	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”。 仅适用于梯形图编程。
Axis	输出	AXIS_REF 参考页数 384	梯形图编程中的轴输出为只读。
InVelocity	输出	BOOL	TRUE - 已达到命令的速度（首次）。
Busy	输出	BOOL	TRUE - 指令块未完成。 FALSE - 指令块已完成。
Active	输出	BOOL	TRUE - 表示功能块可控制轴。
Direction	输出	SINT	有效值包括：-1、0、1。

参数	参数类型	数据类型	描述
CommandAborted	输出	BOOL	TRUE - 命令已被其他命令或 Error Stop 覆盖。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。

MC_MoveVelocity 功能块图示例



MC_MoveVelocity 梯形图示例



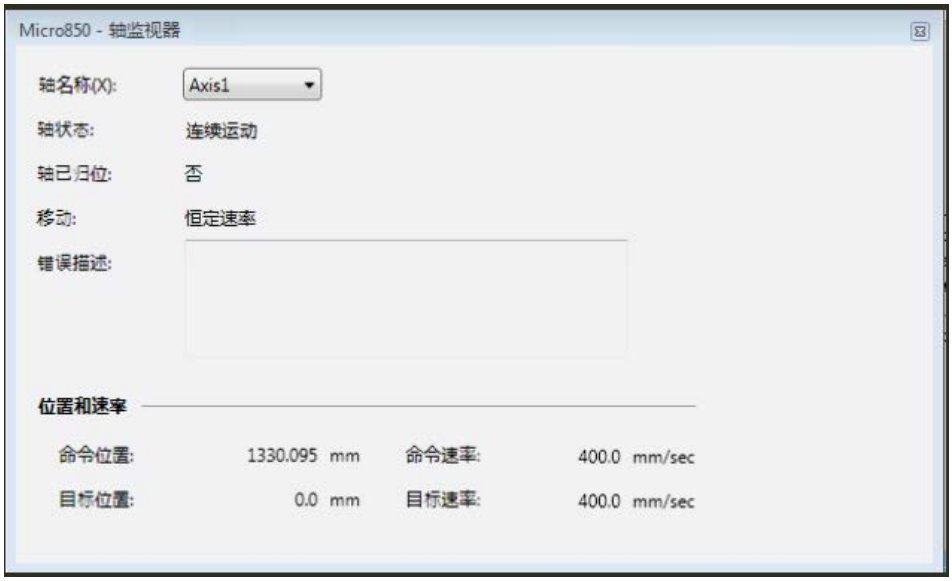
MC_MoveVelocity 结构化文本示例

```
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.

```
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;
```

结果



	名称	别名	逻辑值	实际值	初始值	锁定	数据类型	维度
▶	Execute_Velocity		☑	不可用		☐	BOOL	
	Velocity_Velocit		400.0	不可用		☐	REAL	
	Acceleration_Vel		100.0	不可用		☐	REAL	
	Deceleration_Vel		100.0	不可用		☐	REAL	
	Jerk_Velocity		100.0	不可用		☐	REAL	
	DirectionIn_Velc		1	不可用		☐	SINT	
	BufferMode_Veloc		0	不可用		☐	SINT	
	InVelocity_Veloc		☑	不可用		☐	BOOL	
	Busy_Velocity		☐	不可用		☐	BOOL	
	Active_Velocity		☐	不可用		☐	BOOL	
	Direction_Veloci		1	不可用		☐	SINT	
	CommandAbort_Vel		☐	不可用		☐	BOOL	
	Error_Velocity		☐	不可用		☐	BOOL	
	ErrorID_Velocity		0	不可用		☐	UINT	

MC_Power（运动控制电源）

控制功率（打开或关闭）。

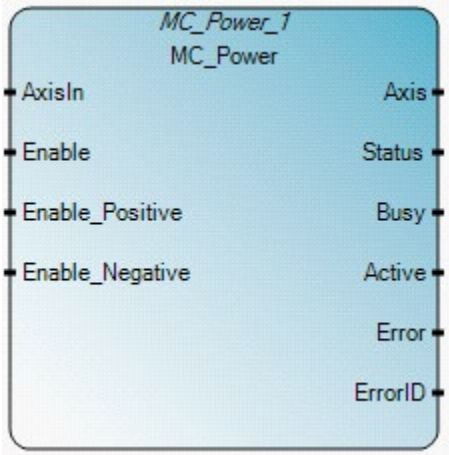
操作详细信息：

- 如果将在 CCW 7 中创建的项目导入 CCW 8，则显示 Mc_Power 新输入参数 __DTI_AxisIn。若出现生成错误，请重新选择指令并重新生成。
- 轴通电完成后，轴 Homed 状态会重置为 0（未归位）。
- MC_Power 的 Enable_Positive 输入和 Enable_Negative 输入都是级别触发的，并且 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，则轴状态将更改为 ErrorStop。

- MC_Power 指令具有 2 分钟的超时值。超时期限到期和“驱动器就绪输入”为 FALSE 时，MC_Power 会返回错误。
- 如果调用 Enable 设置为 True 的 MC_Power 功能块而轴状态为 Disabled，则轴状态会更改为 StandStill，除非检测到错误，在这种情况下轴状态更改为 ErrorStop。
- 每个轴仅可发布一个 MC_Power 功能块。如果使用不同的 MC_Power 功能块同时控制同一轴，则运动引擎会拒绝该轴。
- 轴存在电源开或关状态切换时，则不会复位绝对轴位置。
- 如果调用 Enable 设置为 False 的 MC_Power 功能块，则轴所有状态（包括 ErrorStop）更改为 Disabled。
- 如果将 Enable 设置为 True，MC_Power 功能块可以接通轴的电源；如果将 Enable 设置为 False，则可以切断轴的电源。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。

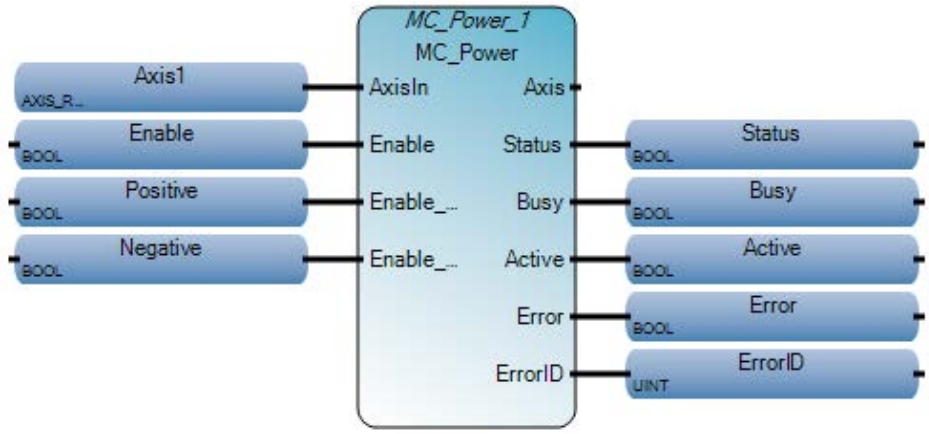


使用此表有助于确定该指令的参数值。

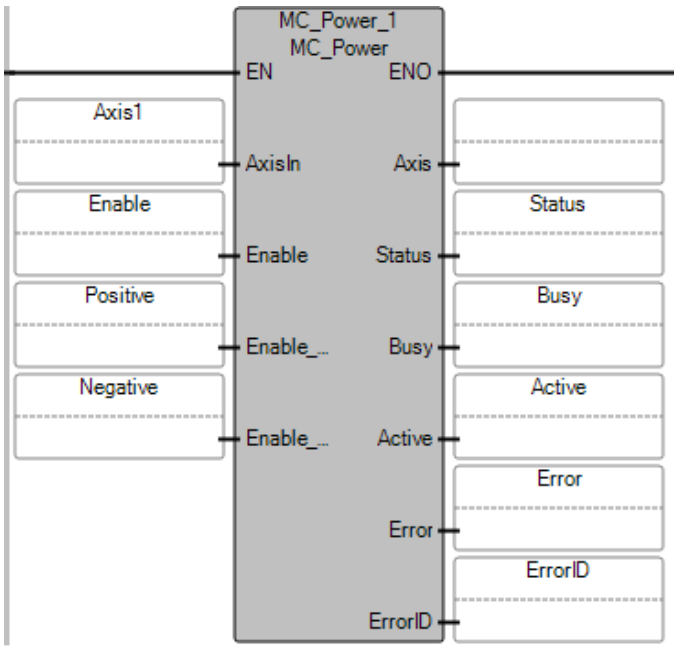
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_Power 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF FB_AXIS_REF	使用 AXIS_REF 数据类型 参考页数 384 参数定义 AxisIn。 对于 FB_Axis（反馈轴），使用 FB_AXIS_REF 数据类型 参考页数 385 定义 AxisIn。
Enable	输入	BOOL	TRUE - 开启电源。 FALSE - 关闭电源
Enable_Positive	输入	BOOL	TRUE - 允许沿正方向运动。
Enable_Negative	输入	BOOL	TRUE - 允许沿负方向运动。
ENO	输出	BOOL	启用“输出”。 仅适用于梯形图编程。
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。 AXIS_REF 数据类型。

参数	参数类型	数据类型	描述
Status	输出	BOOL	功率的状态： • TRUE - 驱动器通电完成。 • FALSE - 驱动器通电未完成。
Busy	输出	BOOL	TRUE - 指令块未完成。 FALSE - 指令块已完成。
Active	输出	BOOL	TRUE - 表示功能块可控制轴。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。

MC_Power 功能块图示例



MC_Power 梯形图示例



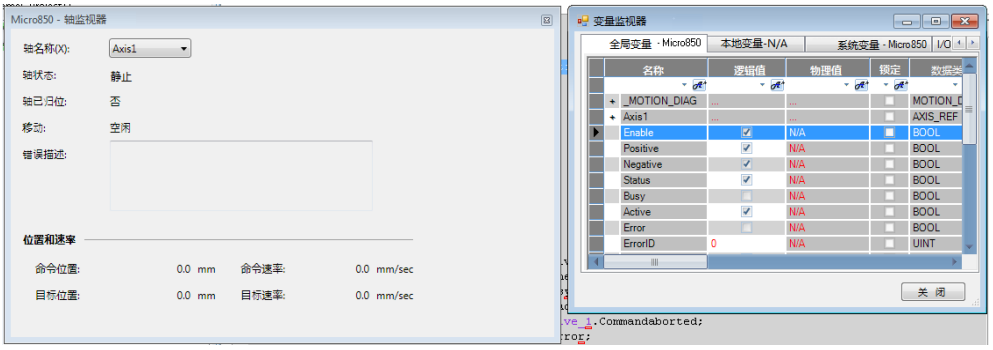
MC_Power 结构化文本示例

```
MC_Power_1(
```

```
void MC_Power_1([AXIS_REF|FB_AXIS_REF] AxisIn, BOOL Enable, BOOL Enable_Positive, BOOL Enable_Negative)
类型： MC_Power, 伺服模块上电
```

```
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;
```

结果



MC_ReadActualPosition（运动控制读取实际位置）

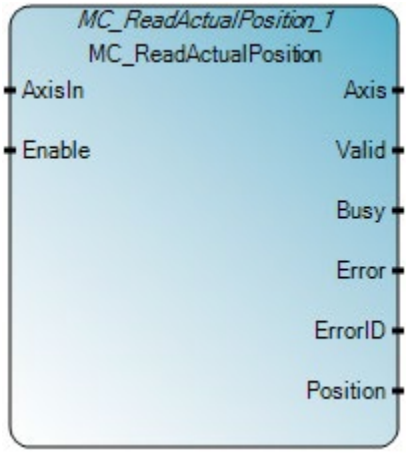
返回反馈轴的实际位置。MC_ReadActualPosition 功能仅适用于反馈运动 。

操作详细信息：

- 执行 MC_ReadActualPosition 之前，确认轴是否处于以下轴状态之一：
  - Disabled
  - 静止
  - 离散运动
  - 错误停止
- 下载后，反馈轴的实际位置不会复位为 0。要复位或清除反馈轴的位置，请使用 MC_Home 指令或 MC_SetPosition 指令。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

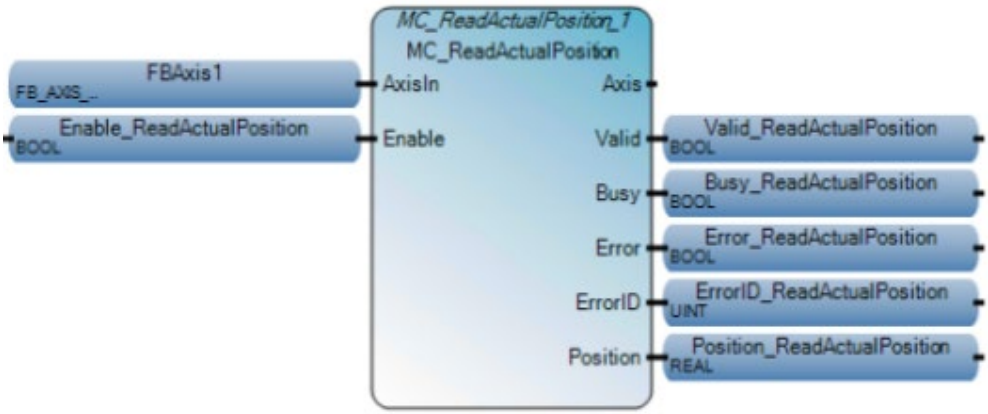
该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。



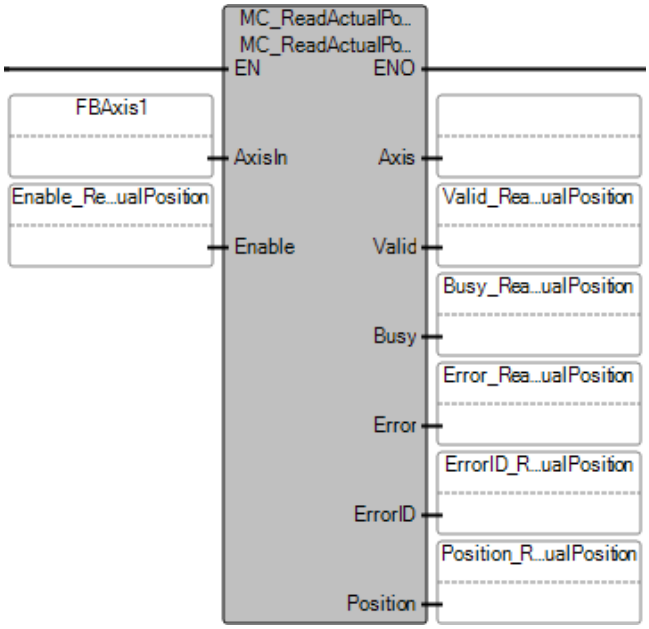
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
AxisIn	输入	FB_AXIS_REF	对于 FB_Axis（反馈轴），使用 FB_AXIS_REF 数据类型 参考页数 385 定义 AxisIn。
Enable	输入	BOOL	TRUE - 在启用后持续获取参数值。 FALSE - 空闲。
Axis	输出	FB_AXIS_REF	梯形图编程中的轴输出为只读。 轴输出参数在 FB_AXIS_REF 数据类型中定义。
Valid	输出	BOOL	TRUE - 指令块处于活动状态，并且预期新的输出值。 FALSE - 指令块处于不活动状态。
Busy	输出	BOOL	TRUE - 指令块未完成。 FALSE - 指令块已完成。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UNIT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。
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 功能块图示例



MC_ReadActualPosition 梯形图示例



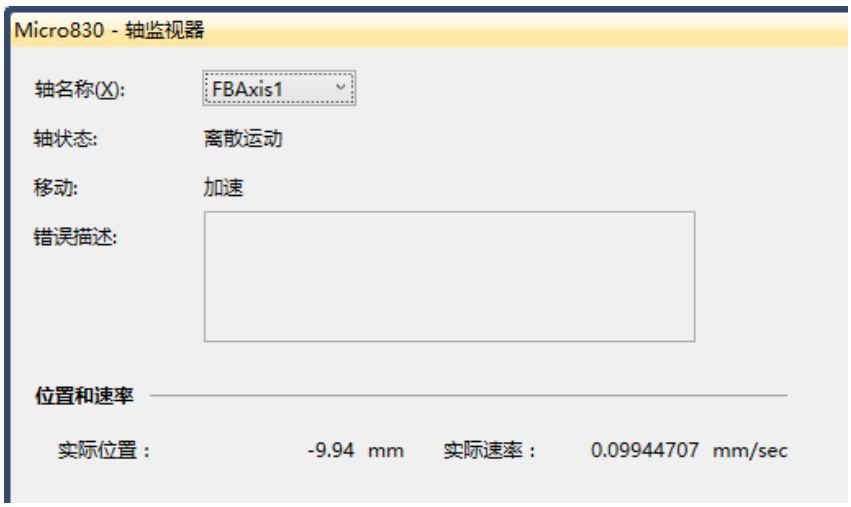
MC_ReadActualPosition 结构化文本示例

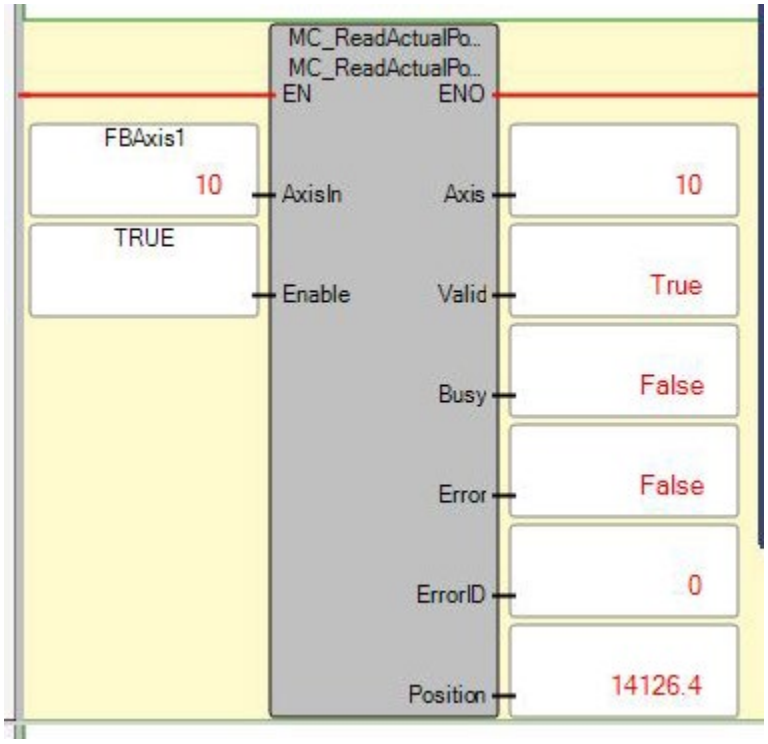
```
void MC_ReadActualPosition_1(
```

void **MC_ReadActualPosition_1**(FB_AXIS_REF **AxisIn**, BOOL **Enable**)
类型： MC_ReadActualPosition, 返回轴的实际位置

```
MC_ReadActualPosition_1(FBAxis, Enable_ReadActualPosition)
Valid_ReadActuralPosition := MC_ReadActualPosition_1.Valid
Busy_ReadActuralPosition := MC_ReadActualPosition_1.Busy
Error_Valid_ReadActuralPosition := MC_ReadActualPosition_1.Error
ErrorID_Valid_ReadActuralPosition := MC_ReadActualPosition_1.ErrorID
Position_Valid_ReadActuralPosition := MC_ReadActualPosition_1.Position
```

结果





名称	别名	逻辑值
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	0
MC_ReadActualPosition_2.Position	Pos	32.16

MC_ReadActualVelocity（运动控制读取实际速度）

返回反馈轴的实际速度值。

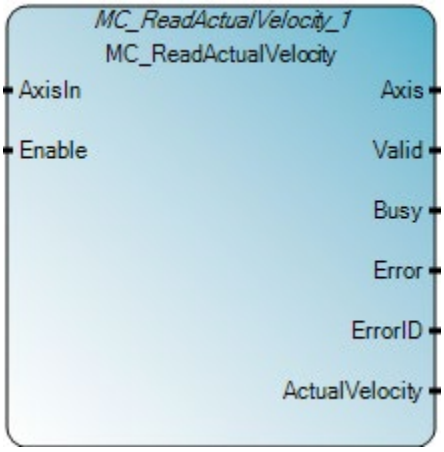
MC_ReadActualVelocity 功能仅适用于反馈运动。

执行 MC_ReadActualVelocity 之前，确认轴是否处于以下轴状态之一：

- 静止
- 离散运动
- 错误停止

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

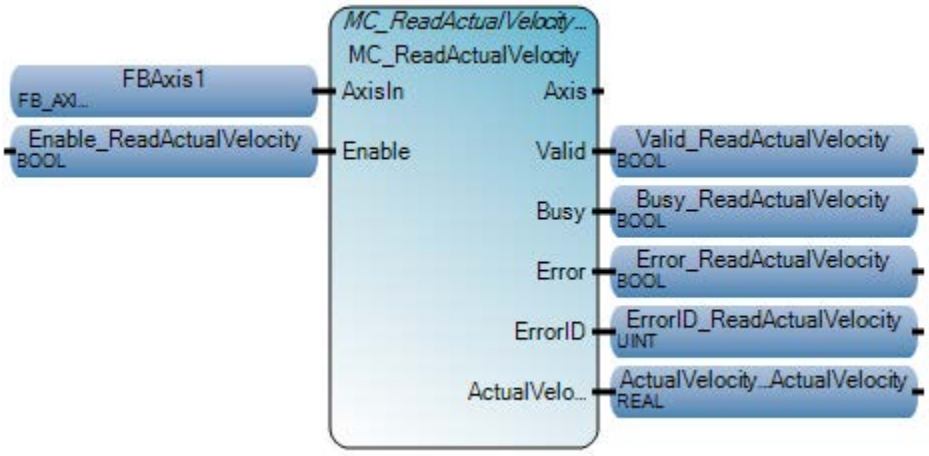
该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。



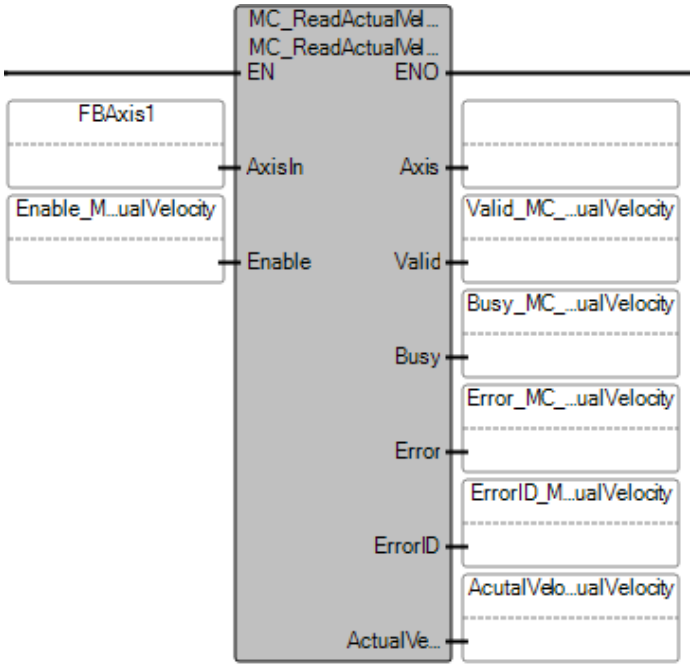
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
AxisIn	输入	FB_AXIS_REF	对于 FB_Axis（反馈轴），使用 FB_AXIS_REF 数据类型定义 AxisIn。
Enable	输入	BOOL	TRUE - 持续获取实际速度的值。 FALSE - 数据不再有效，所有输出都将复位为 0，Valid 设置为 False。
Axis	输出	FB_AXIS_REF	梯形图编程中的轴输出为只读。 轴输出参数在 FB_AXIS_REF 数据类型 参考页数 385 中定义。
Valid	输出	BOOL	TRUE - 功能块处于活动状态，并且预期新的输出值。 FALSE - 功能不预期新的输出值。
Busy	输出	BOOL	TRUE - 功能块未完成。 FALSE - 功能处于空闲状态。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UNIT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。
ActualVelocity	输出	REAL	反馈运动轴的实际速度值（以 [u/s] 为轴单位）。 ActualVelocity 是带符号的值，其中包括方向信息。

MC_ReadActualVelocity 功能块图示例



MC_ReadActualVelocity 梯形图示例



MC_ReadActualVelocity 结构化文本示例

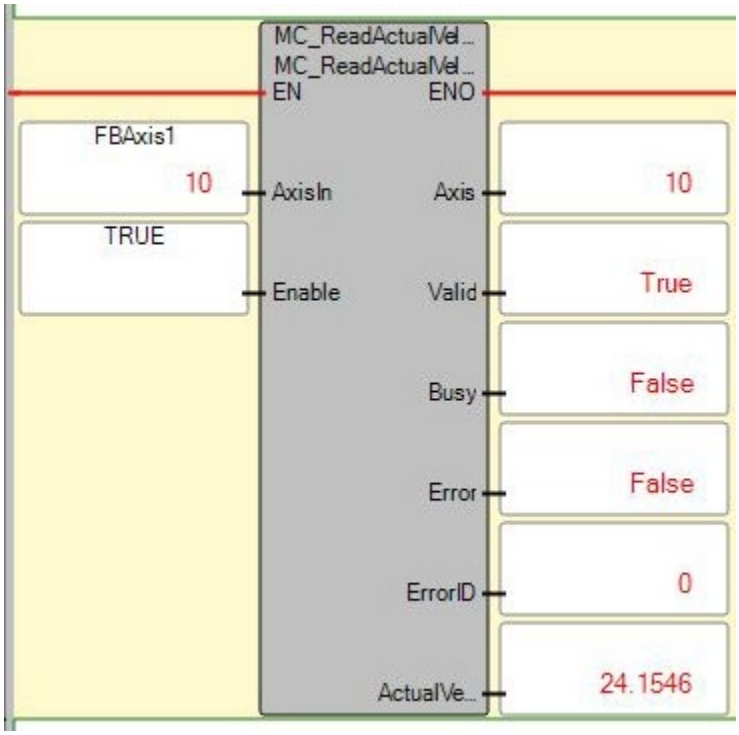
```
void MC_ReadActualVelocity_1(
```

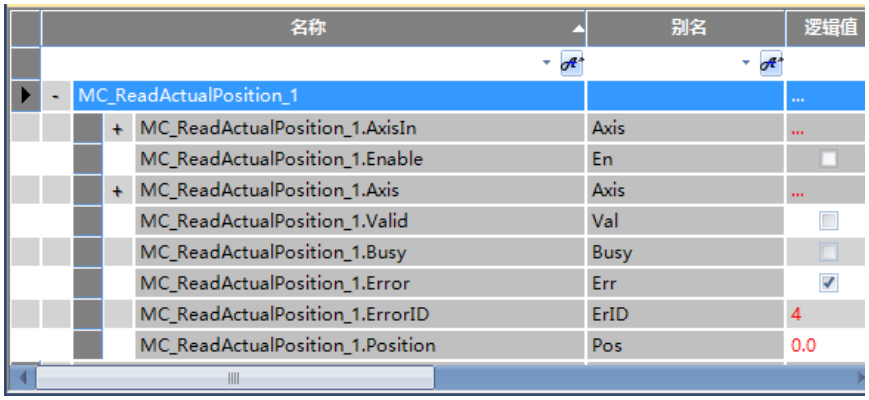
void **MC_ReadActualVelocity_1**(FB_AXIS_REF **AxisIn**, BOOL **Enable**)
类型： MC_ReadActualVelocity, 返回轴的实际速率

```
MC_ReadActualVelocity_1(FBAxis1,Enable_ReadActualVelocity)
Valid_ReadActualVelocity := MC_ReadActualVelocity_1.Valid
Busy_ReadActualVelocity := MC_ReadActualVelocity_1.Busy
Error_ReadActualVelocity := MC_ReadActualVelocity_1.Error
ErrorID_ReadActualVelocity := MC_ReadActualVelocity_1.ErrorID
Position_ReadActualVelocity := MC_ReadActualVelocity_1.ErrorID
```

结果







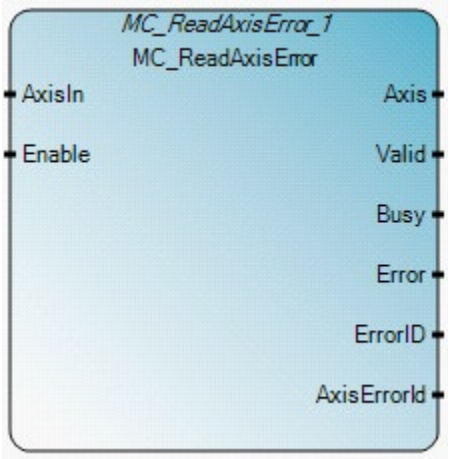
MC_ReadAxisError（运动控制读取轴错误）

读取与运动控制指令块无关的轴错误。

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 可能也可能不针对此轴获得非零的错误 ID。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

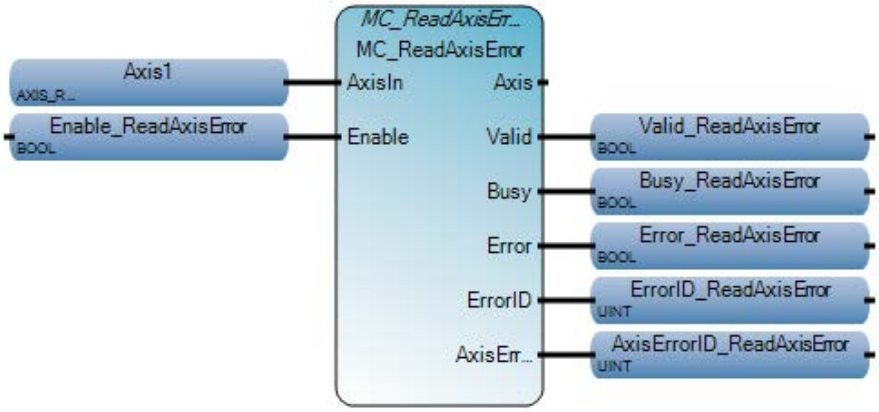
该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。



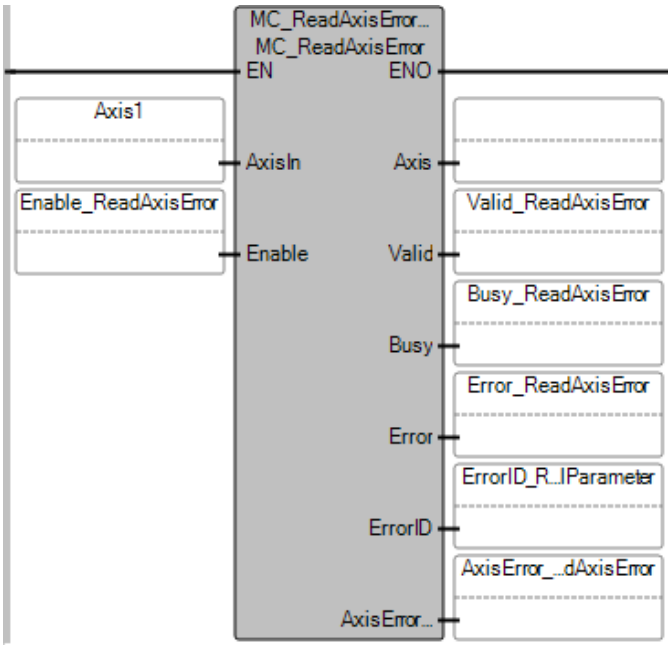
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_ReadAxisError 计算。 FALSE - 将 Error、ErrorID 和 AxisErrorID 复位为 False（或 0）。 仅适用于 LD 编程。
AxisIn	输入	AXIS_REF FB_AXIS_REF	使用 AXIS_REF 数据类型 参考页数 384 参数定义 AxisIn。 对于 FB_Axis（反馈轴），使用 FB_AXIS_REF 数据类型定义 AxisIn。
Enable	输入	BOOL	TRUE - 在启用后持续获取参数值。 FALSE - 将 Error、ErrorID 和 AxisErrorID 输出复位为 0。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。 轴输出参数在 FB_AXIS_REF 数据类型 参考页数 385 中定义。
Valid	输出	BOOL	TRUE - 指令块处于活动状态，并且预期新的输出值。 FALSE - 指令块处于不活动状态。
Busy	输出	BOOL	TRUE - 指令块未完成。 FALSE - 指令块已完成。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 中定义该指令的错误。
AxisErrorID	输出	UINT	标识轴错误的唯一数字。在 AxisErrorID 错误代码 参考页数 418 中定义该指令的错误。

MC_ReadAxisError 功能块图示例



MC_ReadAxisError 梯形图示例

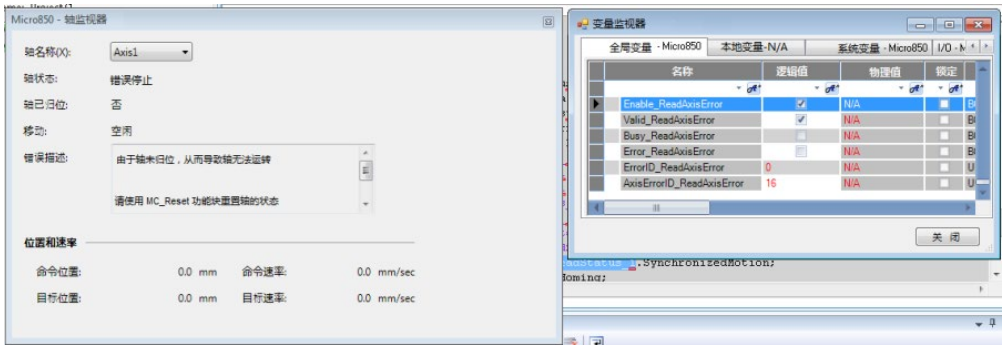


MC_ReadAxisError 结构化文本示例




```
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;
```

结果



AxisErrorID 错误代码

下表描述了 AxisErrorID 中针对 MC_ReadAxisError 参考页数 415 标识的轴错误代码。

值	MACRO ID	描述
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_Power 和 MC_Reset 功能块复位轴的状态。
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_Power 和 MC_Reset 功能块复位轴的状态。 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_Power 和 MC_Reset 功能块复位轴的状态。 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
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_Power 和 MC_Reset 功能块复位轴的状态。
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_Power 和 MC_Reset 功能块复位轴的状态。 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。

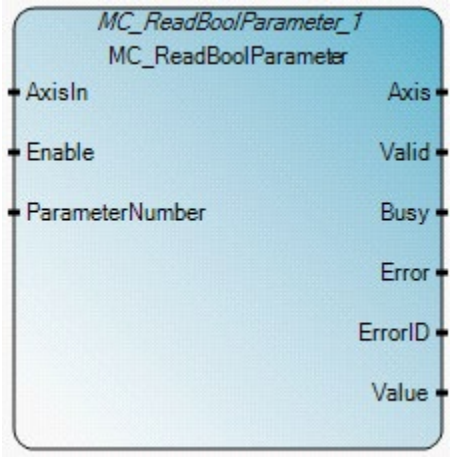
值	MACRO ID	描述
08	MC_FB_ERR_PROFILE	由于功能块中定义的运动配置无效，所以轴无法运转。 请使用 MC_Power 和 MC_Reset 功能块复位轴的状态。 更正功能块中的配置。
09	MC_FB_ERR_VELOCITY	由于功能块中请求的运动配置与当前轴速率冲突，所以轴无法运转。 可能的原因： • 功能块要求轴在移动时反向进行。 • 当前速率对于请求的运动配置过低或过高。 请使用 MC_Power 和 MC_Reset 功能块复位轴的状态。 当轴速度与请求的运动配置兼容时，更正功能块中的运动配置或重新执行功能块。
10	MC_FB_ERR_SOFT_LIMIT	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_Power 和 MC_Reset 功能块复位轴的状态。 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_Power 和 MC_Reset 功能块复位轴的状态。 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。
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_Power 和 MC_Reset 功能块复位轴的状态。 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。

MC_ReadBoolParameter（运动控制读取 BOOL 参数）

返回特定于供应商的类型为 BOOL 的参数的值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

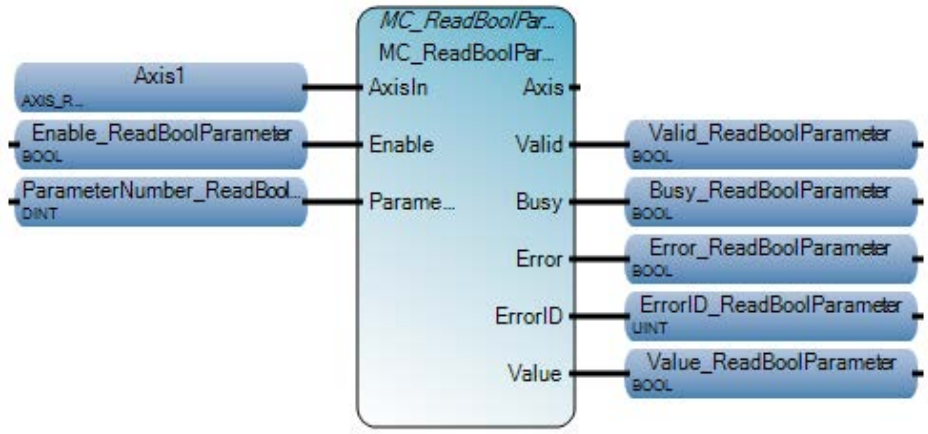
该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。



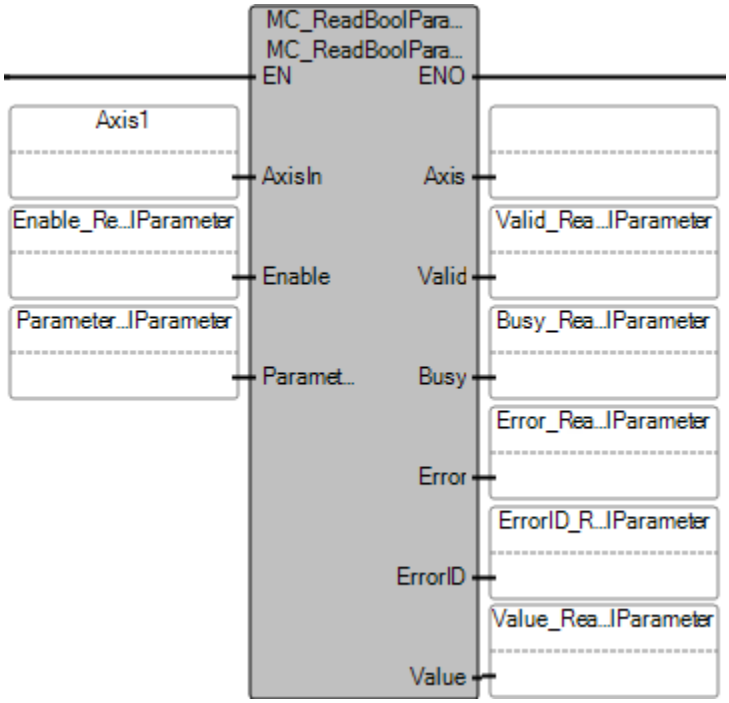
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_ReadBoolParameter 计算。 FALSE - 将值输出复位为 0。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF FB_AXIS_REF	使用 AXIS_REF 数据类型 参考页数 384 定义 AxisIn。 对于 FB_Axis（反馈轴），使用 FB_AXIS_REF 数据类型 参考页数 385 定义 AxisIn。
Enable	输入	BOOL	TRUE - 在启用后持续获取参数值。 FALSE - 将值输出复位为 0。
ParameterNumber	输入	DINT	参数标识。 参数编号定义在运动控制功能块参数编号中定义。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。
Valid	输出	BOOL	TRUE - 参数的值可用。 FALSE - 参数值不可用。
Busy	输出	BOOL	TRUE - 功能块正在运行，并且预期新的输出值。 FALSE - 功能处于空闲状态。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。
Value	输出	BOOL	由供应商指定数据类型的指定参数值

MC_ReadBoolParameter 功能块图示例



MC_ReadBoolParameter 梯形图示例



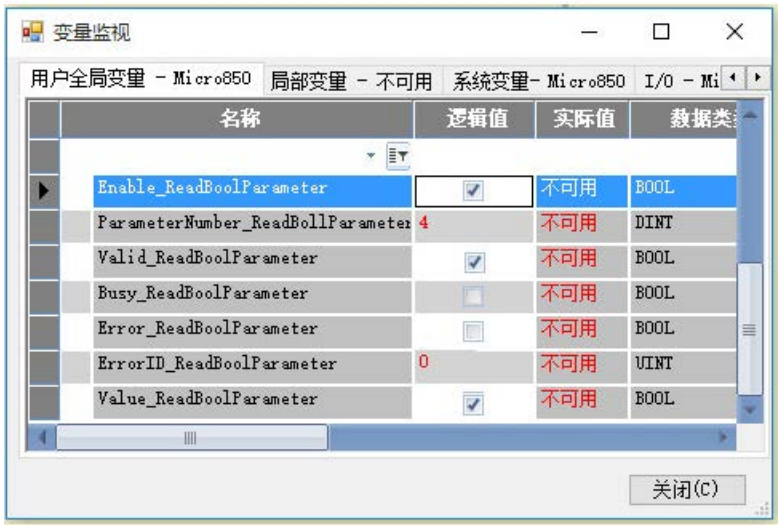
MC_ReadBoolParameter 结构化文本示例

```
MC_ReadBoolParameter_1(
```

void **MC_ReadBoolParameter_1**([AXIS_REF|FB_AXIS_REF] **AxisIn**, BOOL **Enable**, DINT **ParameterNumber**)
类型： MC_ReadBoolParameter, 返回运动特定 BOOL 参数的值。

```
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;
```

结果

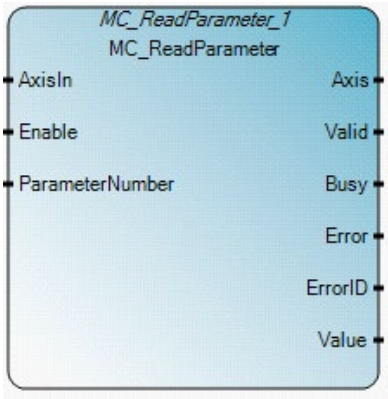


MC_ReadParameter（运动控制读取参数）

以实型数据类型返回供应商特定参数的值。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。

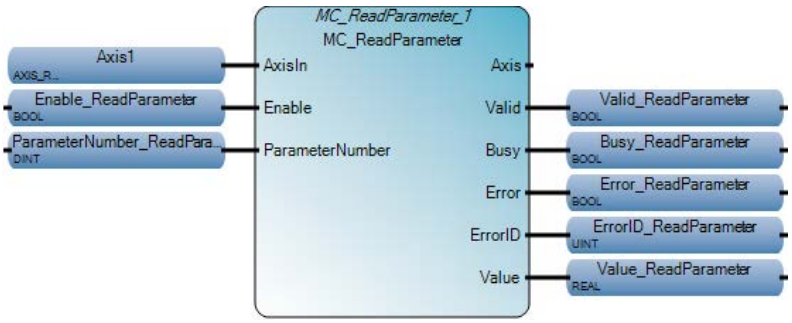


使用此表有助于确定该指令的参数值。

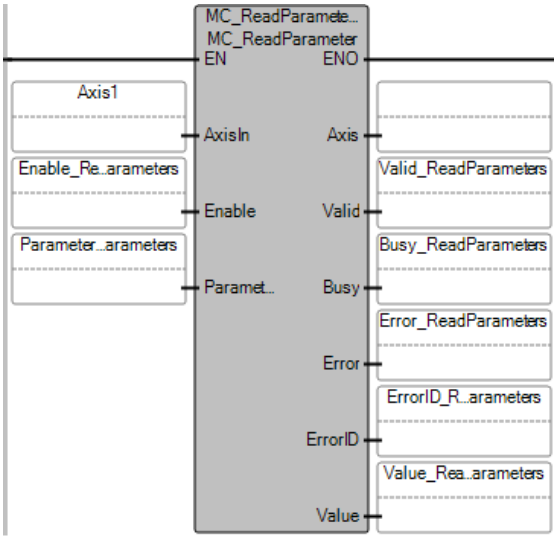
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_ReadParameter 计算。 FALSE - 将值输出复位为 0。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF FB_AXIS_REF	使用 AXIS_REF 数据类型 参考页数 384 定义 AxisIn。 对于 FB_Axis（反馈轴），使用 FB_AXIS_REF 数据类型 参考页数 385 定义 AxisIn。
Enable	输入	BOOL	TRUE - 持续获取参数编号的值。 FALSE - 将值输出复位为 0。
ParameterNumber	输入	DINT	参数标识。 参数编号定义在运动控制功能块参数编号中定义。
ENO	输出	BOOL	启用“输出”。 仅适用于梯形图编程。

参数	参数类型	数据类型	描述
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。
Valid	输出	BOOL	TRUE - 有效输出可用。 FALSE - 有效输出不可用。
Busy	输出	BOOL	TRUE - 功能块正在运行，并且预期新的输出值。 FALSE - 功能块处于空闲状态。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。
Value	输出	REAL	由供应商指定数据类型的指定参数值

MC_ReadParameter 功能块图示例



MC_ReadParameter 梯形图示例

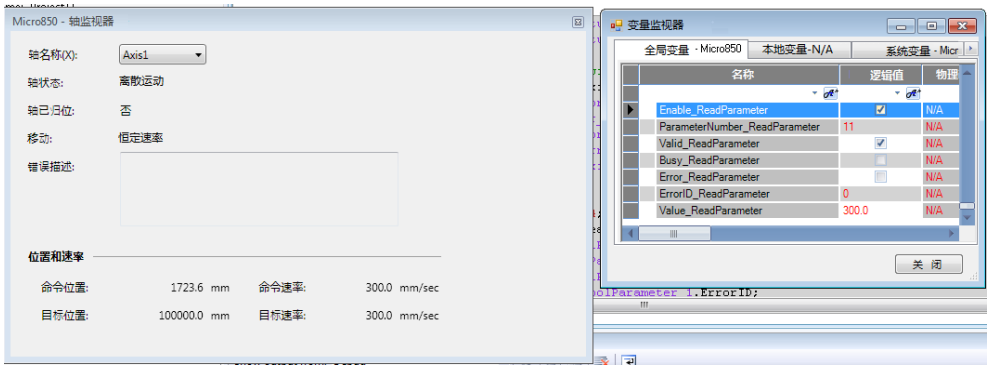


MC_ReadParameter 结构化文本示例



```
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;
```

结果

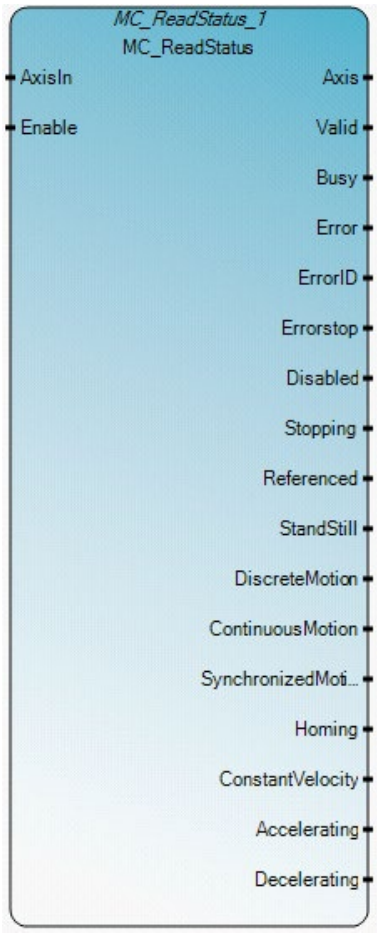


MC_ReadStatus（运动控制读取状态）

返回与当前正在进行中的运动相关的轴的状态。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

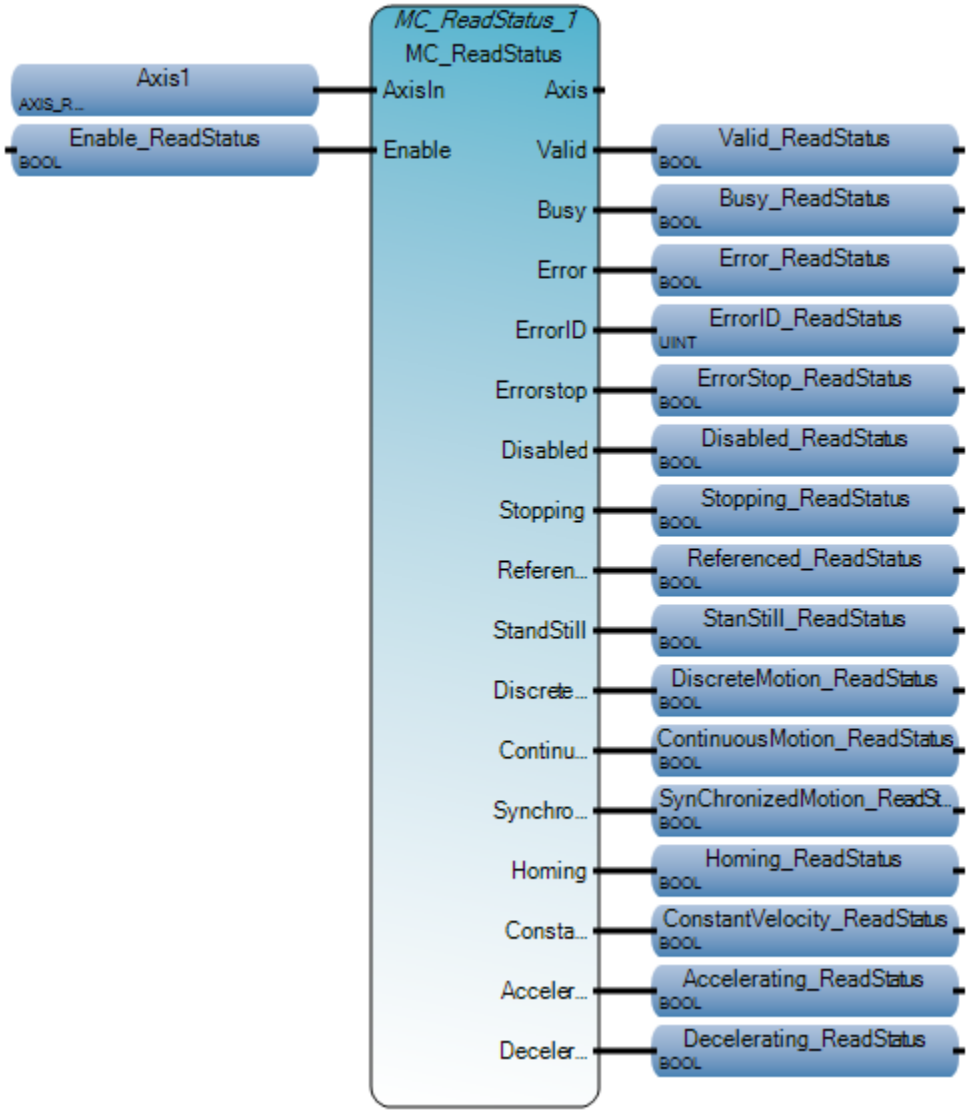
该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。



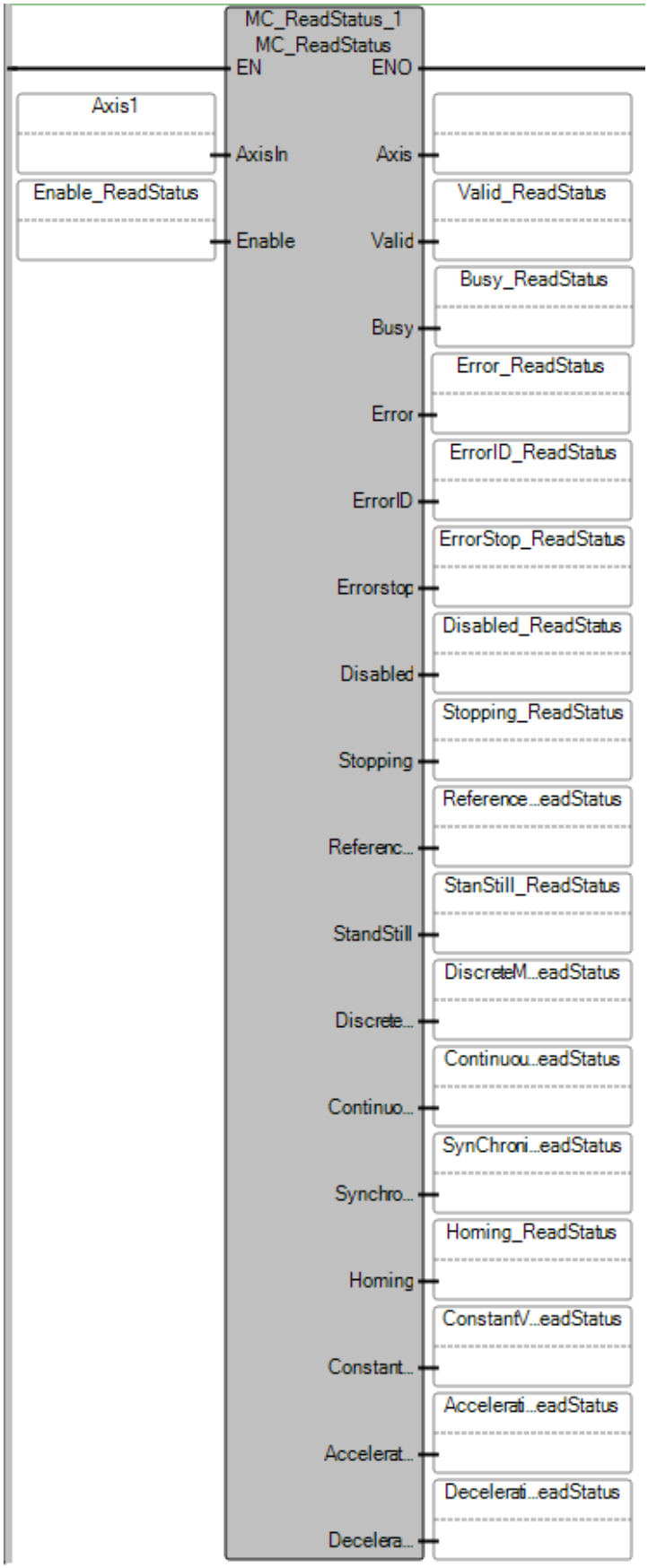
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_ReadStatus 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF FB_AXIS_REF	使用 AXIS_REF 数据类型 参考页数 384 参数定义 AxisIn。 对于 FB_Axis（反馈轴），使用 FB_AXIS_REF 数据类型 参考页数 385 定义 AxisIn。
Enable	输入	BOOL	TRUE - 持续获取轴状态。 FALSE - 将所有状态输出复位为 0。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。
Valid	输出	BOOL	TRUE - 有效输出可用。 FALSE - 输出不可用。
Busy	输出	BOOL	TRUE - 功能块正在运行，并且预期新的输出值。 FALSE - 功能块处于空闲状态。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 中定义该指令的错误。
ErrorStop	输出	BOOL	TRUE - 轴状态为 ErrorStop。 轴状态在运动控制轴状态值和名称中定义。
Disabled	输出	BOOL	TRUE - 轴状态为 Disabled。
Stopping	输出	BOOL	TRUE - 轴状态为 Stopping。
Referenced	输出	BOOL	TRUE - 轴状态已归位，轴的绝对引用位置已知。
StandStill	输出	BOOL	TRUE - 轴状态为 StandStill。
DiscreteMotion	输出	BOOL	TRUE - 轴状态为 DiscreteMotion。
ContinuousMotion	输出	BOOL	TRUE - 轴状态为 ContinuousMotion。
SynchronizedMotion	输出	BOOL	Micro800 控制器中不支持已同步运动。 TRUE - 从不为 True。 FALSE - 始终为 False。
Homing	输出	BOOL	TRUE - 轴状态为 Homing。
ConstantVelocity	输出	BOOL	TRUE - 电机的速度为常量。
Accelerating	输出	BOOL	TRUE - 轴正在加速，增加电机的能量。
Decelerating	输出	BOOL	TRUE - 轴正在减速，减少电机的能量。

MC_ReadStatus 功能块图示例



MC_ReadStatus 梯形图示例



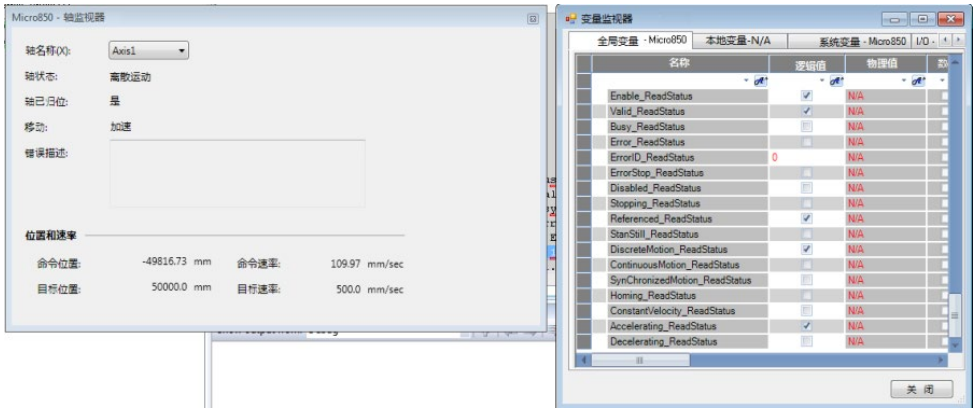
MC_ReadStatus 结构化文本示例

```
MC_ReadStatus_1(
```

```
void MC_ReadStatus_1([AXIS_REF|FB_AXIS_REF] AxisIn, BOOL Enable)
类型： MC_ReadStatus, 返回与当前正在进行中的运动相关的轴的详细状态。
```

```
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;
StanStill_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;
```

结果



MC_Reset（运动控制复位）

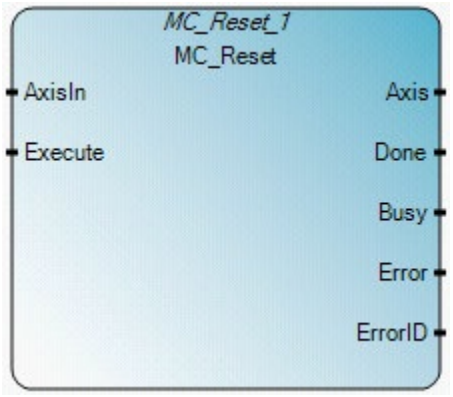
通过复位所有内部轴相关错误将轴状态从 ErrorStop 转换为 StandStill。功能块实例的输出不会改变。

操作详细信息：

- 如果在执行 MC_Reset 后轴警报状态保持不变，则执行 MC_Power，然后执行 MC_Reset。
- MC_Reset 仅将轴状态从 ErrorStop 复位为 StandStill。当轴处于其他状态（包括 Disabled）时执行 MC_Reset 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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

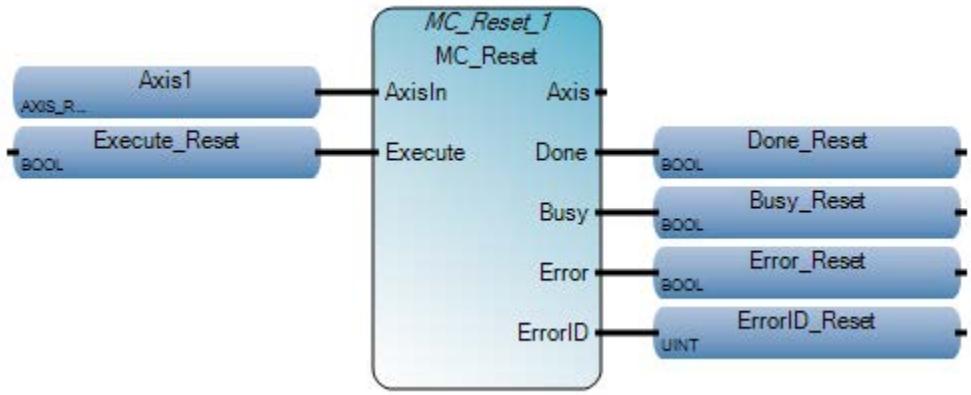
该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。



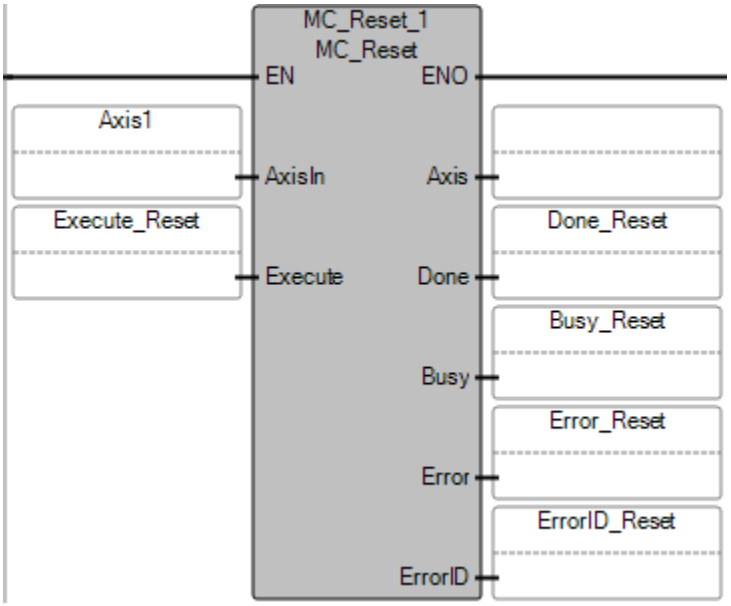
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_Reset 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF FB_AXIS_REF	使用 AXIS_REF 数据类型定义 AxisIn。 对于 FB_Axis（反馈轴），使用 FB_AXIS_REF 数据类型 参考页数 385 定义 AxisIn 。
Execute	输入	BOOL	TRUE - 将轴复位为上升沿。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。
Done	输出	BOOL	TRUE - 轴状态为 StandStill 或 Disabled。
Busy	输出	BOOL	TRUE - 功能块未完成。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。

MC_Reset 功能块图示例



MC_Reset 梯形图示例

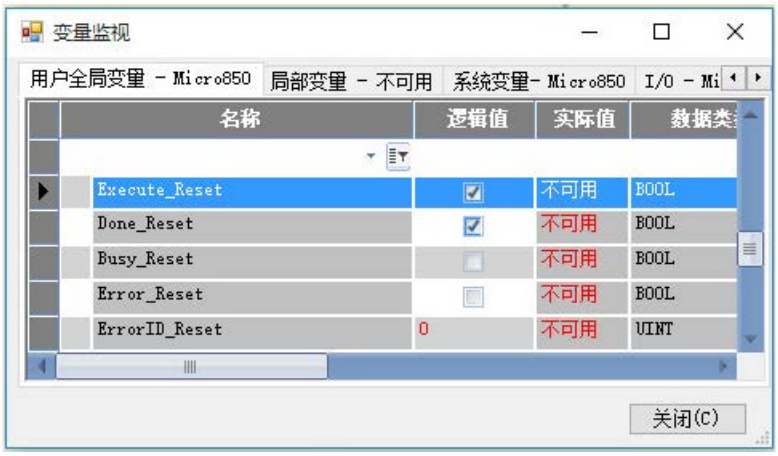


MC_Reset 结构化文本示例




```
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;
```

结果



MC_SetPosition（运动控制设置位置）

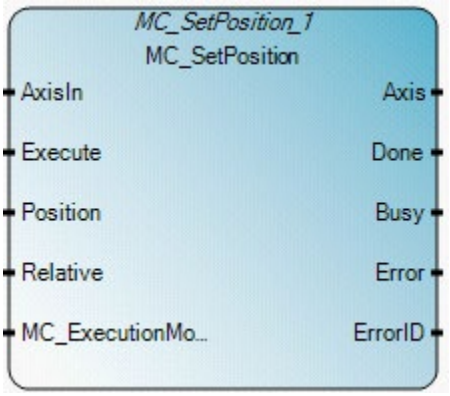
通过控制具有相同值的轴的实际位置而不造成任何移动，从而转移轴坐标系。

操作详细信息：

- 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_SetPostion 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 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 。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。

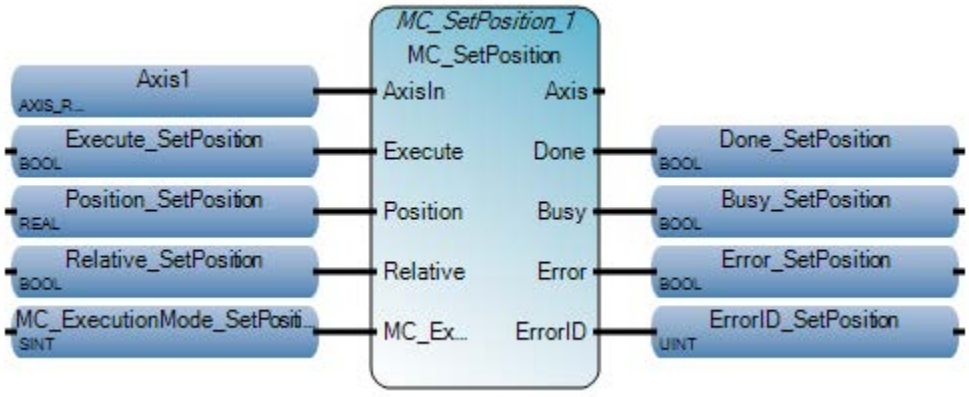


使用此表有助于确定该指令的参数值。

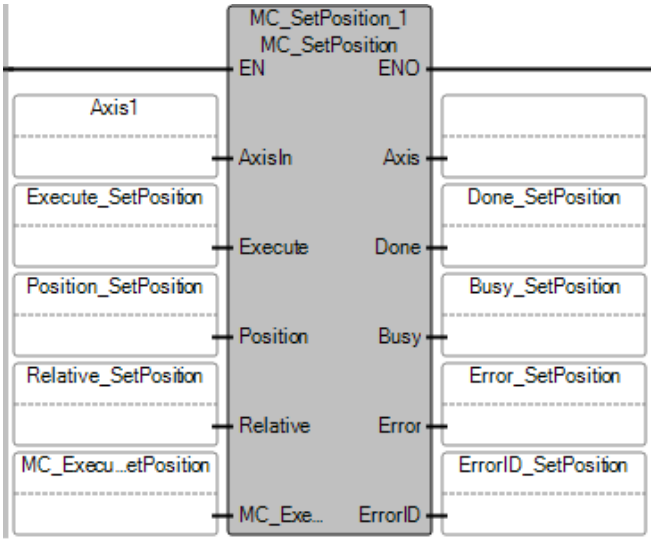
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_SetPosition 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF FB_AXIS_REF	使用 AXIS_REF 数据类型定义 AxisIn。 对于 FB_Axis（反馈轴），使用 FB_AXIS_REF 数据类型 参考页数 385 定义 AxisIn。
Execute	输入	BOOL	TRUE - 开始设置轴位置。
Position	输入	REAL	要为轴设置的绝对位置或相对距离。
Relative	输入	BOOL	TRUE - 设置轴的相对距离。 FALSE - 设置轴的绝对位置。

参数	参数类型	数据类型	描述
MC_ExecutionMode	输入	SINT	值包括： • 0 (*mcImmediately*) - 功能立即生效。 • 1 (*mcQueued*) - 在以下情况下新功能变为有效： • 将所有以前的运动命令设置为以下输出参数之一：Done、Aborted 或 Error。 • 轴未处于移动状态。 对于 (MC_ExecutionMode = 0)，当轴状态为 Disabled 或 Standstill 时，此功能块成功完成。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)，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"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。
Done	输出	BOOL	TRUE - Position 具有新值。
Busy	输出	BOOL	TRUE - 功能块未完成。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。

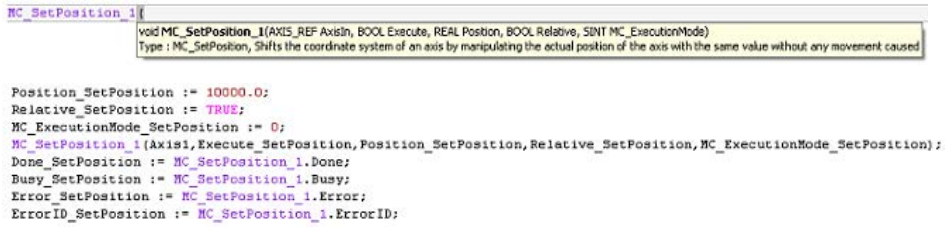
MC_SetPosition 功能块图示例



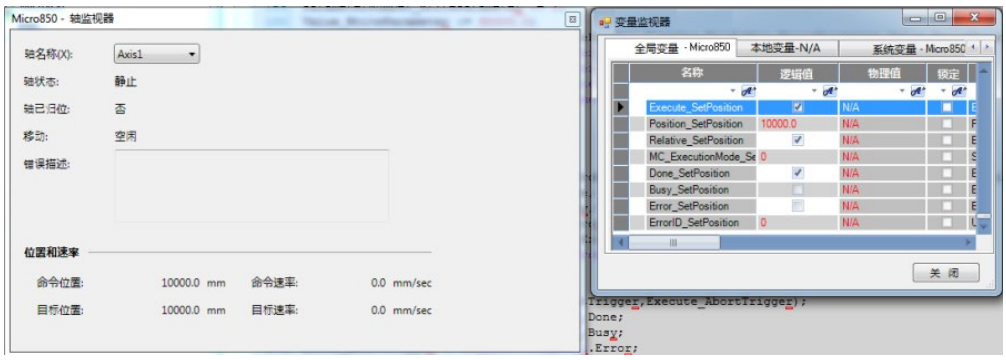
MC_SetPosition 梯形图示例



MC_SetPosition 结构化文本示例



结果



MC_Stop（运动控制停止）

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被取消。所有功能块移动命令也就忽略，直至轴状态转变为 Standstill。

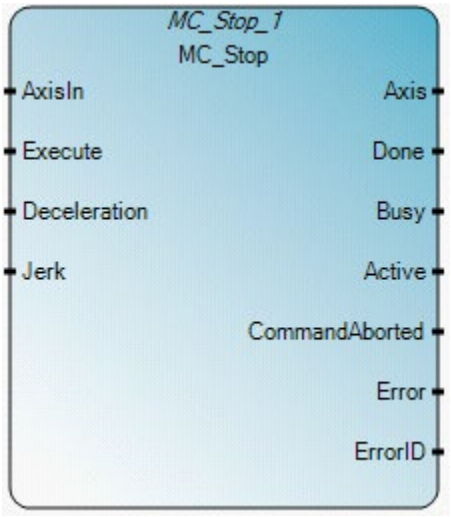
操作详细信息：

- 只要 Execute 输入高，轴便保持在 Stopping 状态。轴处于 Stopping 状态时，其他运动功能块无法在相同的轴上执行任何运动。

- 如果 Deceleration 等于零，则 MC_Stop 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加速度。
- 如果在停止序列期间未检测到错误，则在 Done 位为 SET 且 Execute 输入更改为 False 后，轴状态会变为 StandStill。
- 对于紧急停止功能或异常情况使用 MC_Stop。对于正常运动停止使用 MC_Halt。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。

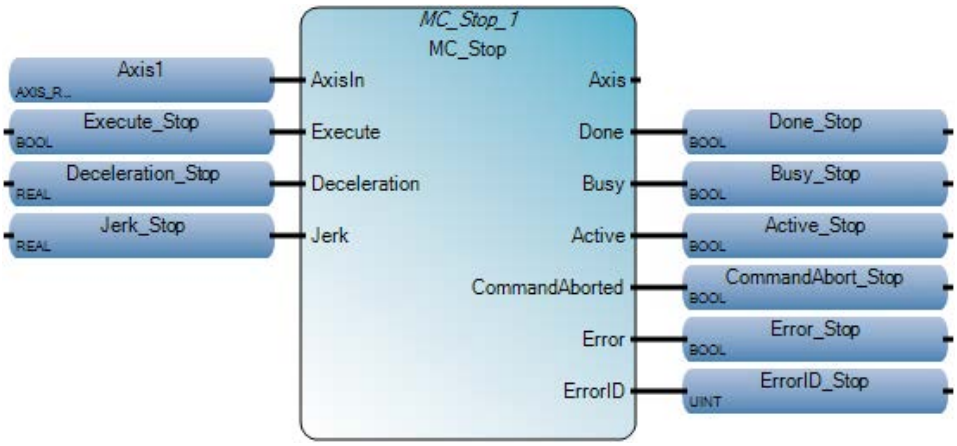


使用此表有助于确定该指令的参数值。

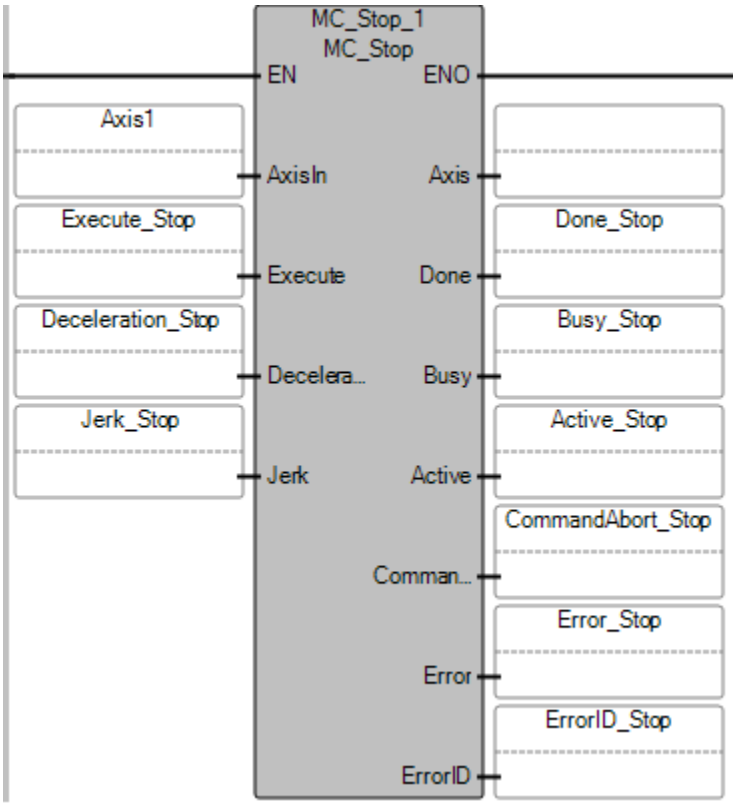
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_Stop 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF	使用 AXIS_REF 数据类型定义 AxisIn 的参数。
Execute	输入	BOOL	TRUE - 在上升沿开始运动。 FALSE - 不执行。
Deceleration	输入	REAL	减速的值 [u/s^2]。
Jerk	输入	REAL	加加速度的值 [u/s^3]。
ENO	输出	BOOL	启用“输出”。 仅适用于梯形图编程。
Axis	输出	AXIS_REF 参考页数 384	梯形图编程中的轴输出为只读。
Done	输出	BOOL	TRUE - 达到零速度，并且在停止序列期间没有出现错误。
Busy	输出	BOOL	TRUE - 功能块未完成。
Active	输出	BOOL	TRUE - 表示功能块可控制轴。
CommandAborted	输出	BOOL	TRUE - 命令已被 MC_Power(OFF) 功能块或 ErrorStop 覆盖。

参数	参数类型	数据类型	描述
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。

MC_Stop 功能块图示例



MC_Stop 梯形图示例

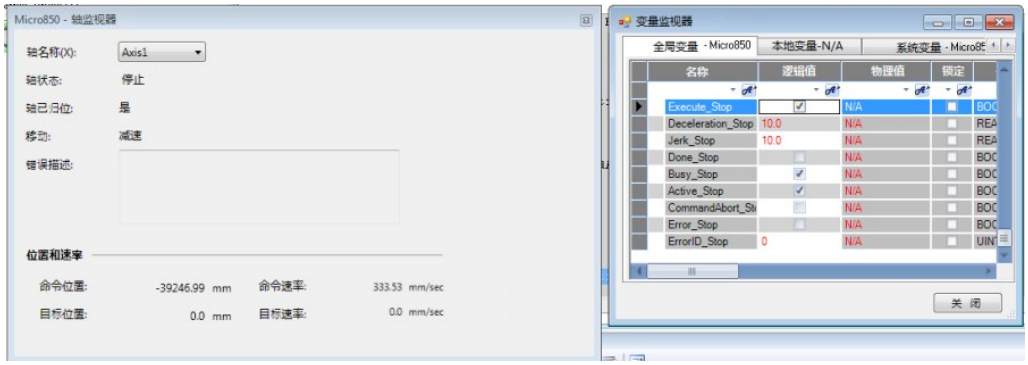


MC_Stop 结构化文本示例




```
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;
```

结果



MC_TouchProbe（运动控制触摸探针）

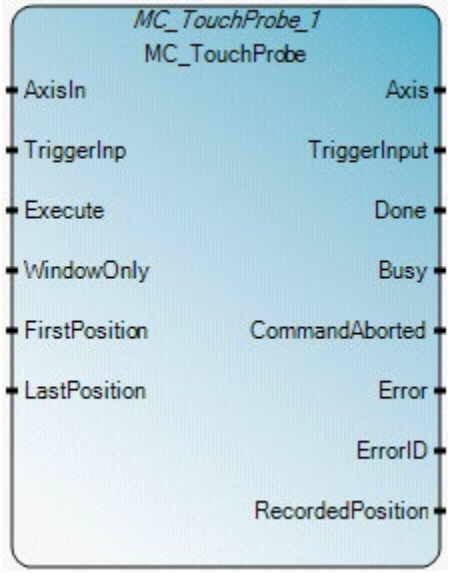
在触发事件中记录轴位置。

操作详细信息：

- 如果窗口方向（第一个位置 --> 最后一个位置）与运动方向相反，则触摸探针窗口不会激活。
- 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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。



使用此表有助于确定该指令的参数值。

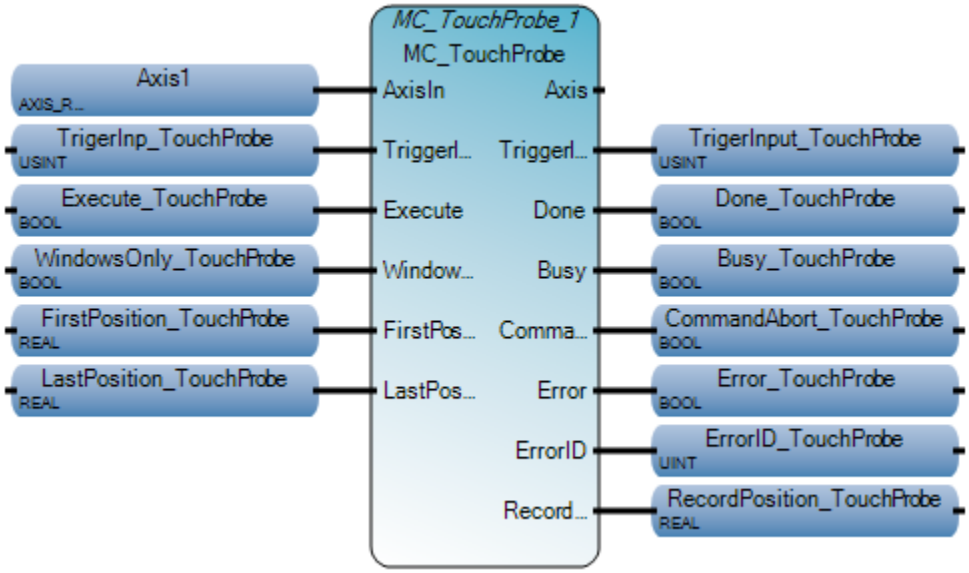
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_TouchProbe 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF FB_AXIS_REF	使用 AXIS_REF 数据类型定义 AxisIn 的参数。 对于反馈轴，使用 FB_AXIS_REF 数据类型 参考页数 385 定义 AxisIn 的参数。
TriggerInp	输入	USINT	目前不支持。在轴配置中配置输入触发器。
Execute	输入	BOOL	TRUE - 在上升沿开始触摸探针记录。 FALSE - 不执行。
WindowOnly	输入	BOOL	TRUE - 仅使用窗口接受触发事件。 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"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。
TriggerInput	输出	USINT	目前不支持。
Done	输出	BOOL	TRUE - 记录触发事件。
Busy	输出	BOOL	TRUE - 功能块未完成。
CommandAborted	输出	BOOL	TRUE - 命令已被 MC_Power(OFF) 或 Error Stop 功能块覆盖。

参数	参数类型	数据类型	描述
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。
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），并且如果驱动器与电机之间没有运动延迟。

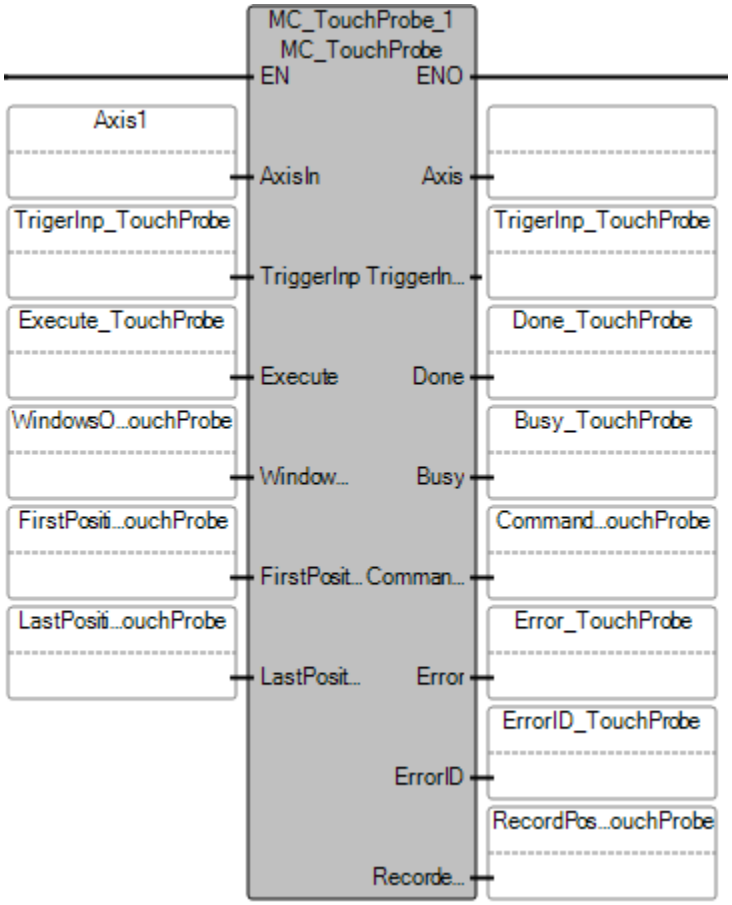
运动固定输入/输出

运动信号	PT00	PT01	PT02
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 功能块图示例



MC_TouchProbe 梯形图示例

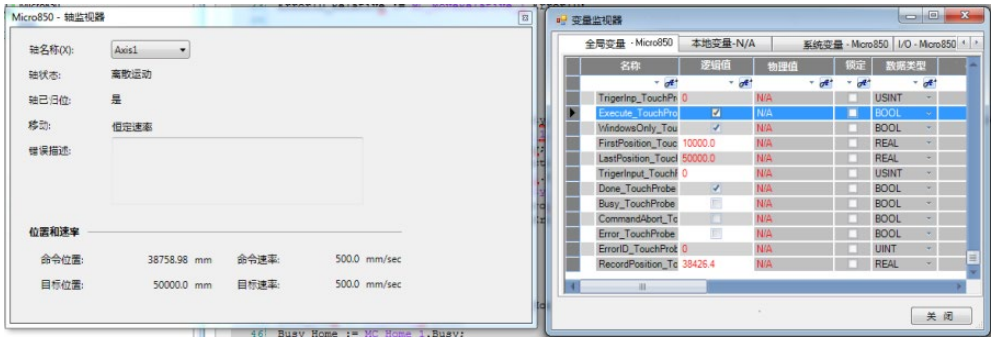


MC_TouchProbe 结构化文本示例




```
FirstPosition_TouchProbe := 10000.0;
LastPosition_TouchProbe := 50000.0;
MC_TouchProbe_1(Axis1,TrigerInp_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;
```

结果



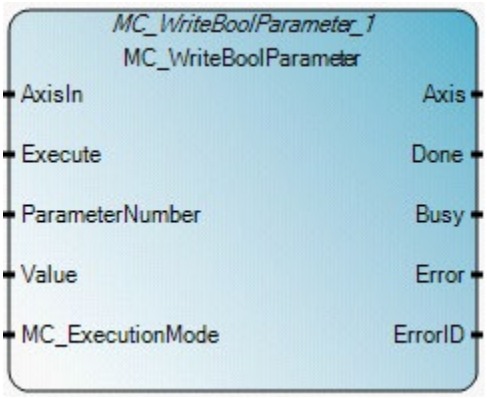
MC_WriteBoolParameter（运动控制写入 BOOL 参数）

修改特定于供应商的类型为 BOOL 的参数的值。

MC_WriteBoolParameter 功能块设置的参数仅暂时应用于应用程序。将控制器从 PRG 切换到运行模式，或将控制器电源断电重启时，它们会被用户在 Connected Components Workbench 运动配置中配置的永久设置覆盖。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。

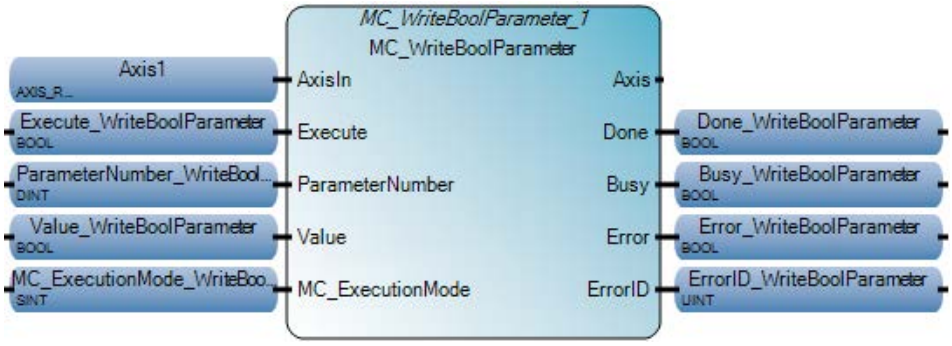


使用此表有助于确定该指令的参数值。

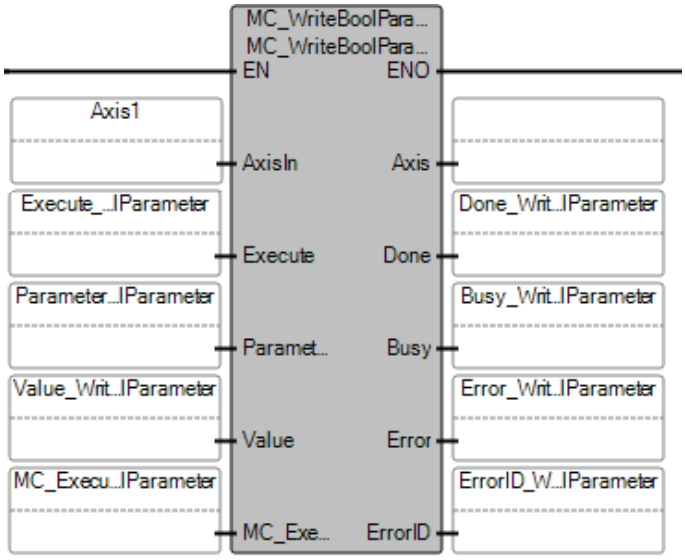
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_WriteBoolParameter 计算。 FALSE - 将值输出复位为 0。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF FB_AXIS_REF	使用 AXIS_REF 数据类型定义 AxisIn 的参数。 对于反馈轴，使用 FB_AXIS_REF 数据类型定义 AxisIn 的参数。
Execute	输入	BOOL	TRUE - 在上升沿写入参数的值。
ParameterNumber	输入	DINT	参数标识。 参数编号在运动控制功能块参数详细信息 参考页数 380 中定义。
Value	输入	BOOL	TRUE - 指定的参数具有新值。

参数	参数类型	数据类型	描述
MC_ExecutionMode	输入	SINT	值包括： • 0 (*mcImmediately*) - 功能立即生效。 • 1 (*mcQueued*) - 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。 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"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。
Done	输出	BOOL	TRUE - 参数已成功写入。
Busy	输出	BOOL	TRUE - 功能块未完成。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。

MC_WriteBoolParameter 功能块图示例



MC_WriteBoolParameter 梯形图示例

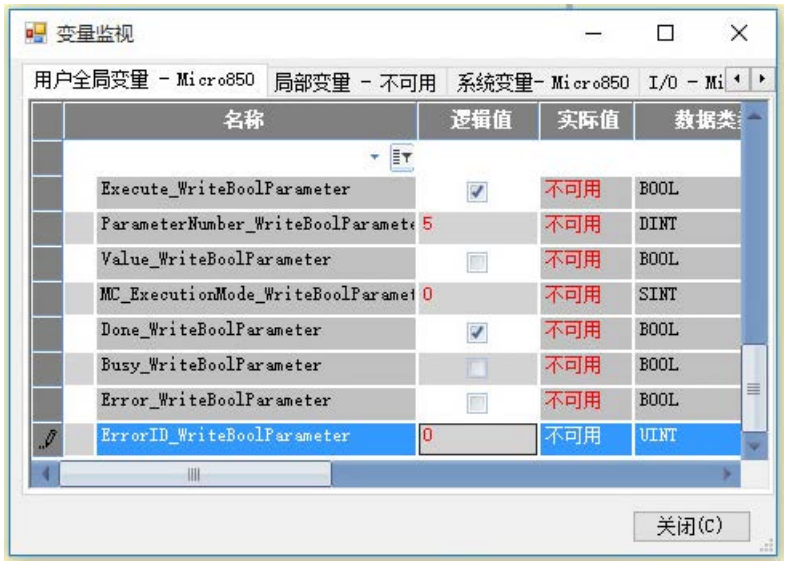


MC_WriteBoolParameter 结构化文本示例




```
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;
```

结果



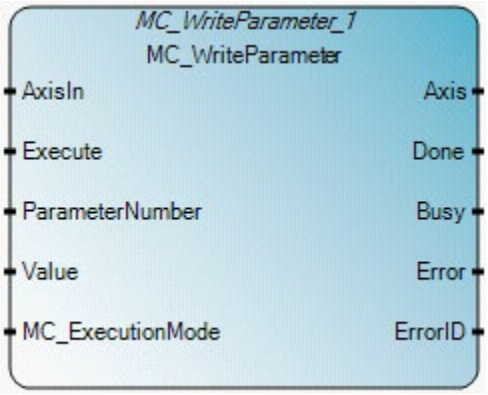
MC_WriteParameter（运动控制写入参数）

修改特定于供应商的类型为 REAL 的参数的值。

MC_WriteParameter 功能块设置的参数仅暂时应用于应用程序。将控制器从 PRG 切换到 RUN，或将控制器断电重启时，它们会被用户在 Connected Components Workbench 运动配置中配置的永久设置覆盖。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于支持运动控制的 Micro830、Micro850 和 Micro870 控制器 。

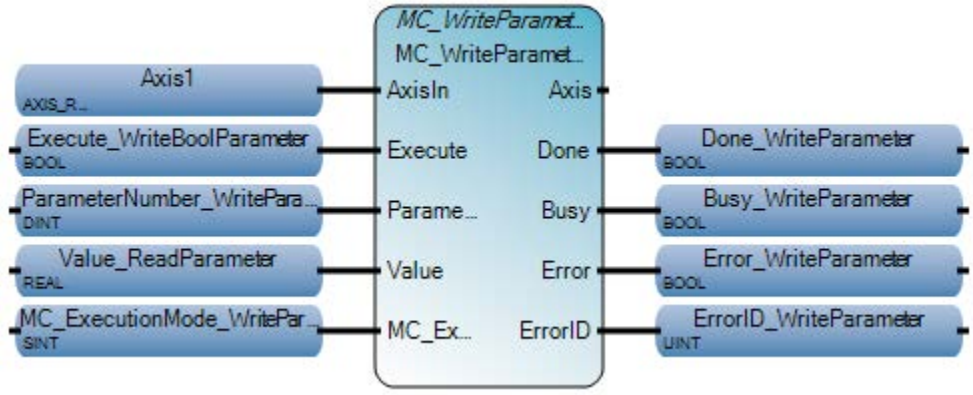


使用此表有助于确定该指令的参数值。

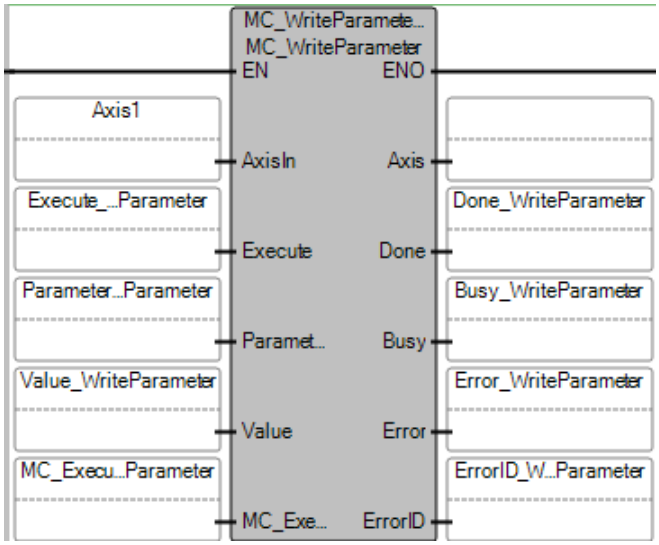
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行当前 MC_WriteParameter 计算。 FALSE - 不执行任何计算。 仅适用于梯形图编程。
AxisIn	输入	AXIS_REF FB_AXIS_REF	使用 AXIS_REF 数据类型 参考页数 384 定义 AxisIn 的参数。 对于反馈轴，使用 FB_AXIS_REF 数据类型 参考页数 385 定义 AxisIn 的参数。
Execute	输入	BOOL	TRUE - 在上升沿写入参数的值。
ParameterNumber	输入	DINT	参数标识。 参数编号在运动控制功能块参数详细信息 参考页数 380 中定义。
Value	输入	REAL	指定参数的新值。

参数	参数类型	数据类型	描述
MC_ExecutionMode	输入	SINT	值包括： • 0 (*mcImmediately*) - 功能立即生效。 • 1 (*mcQueued*) - 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。 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"。 仅适用于梯形图编程。
Axis	输出	AXIS_REF	梯形图编程中的轴输出为只读。
Done	输出	BOOL	TRUE - 参数已成功写入。
Busy	输出	BOOL	TRUE - 表示功能块可控制轴。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
ErrorID	输出	UINT	标识错误的唯一数字。已在运动控制功能块错误 ID 参考页数 383 中定义该指令的错误。

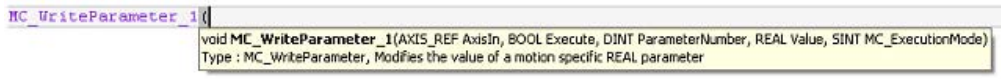
MC_WriteParameter 功能块图示例



MC_WriteParameter 梯形图示例



MC_WriteParameter 结构化文本示例




```
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;
```

结果



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。

指令	描述
DERIVATE 参考页数 447	在定义的循环时间内区分实型值。
FFL 参考页数 449	将 8 位、16 位、32 位或 64 位数据加载到调用 FIFO 堆栈的用户创建的数组。
FFU 参考页数 457	从调用 FIFO 堆栈的用户创建的数组卸载 8 位、16 位、32 位或 64 位数据。使用 FFL 指令按照与加载相同的顺序卸载数据。
HYSTER 参考页数 462	在不同实型值上执行布尔迟滞。
INTEGRAL 参考页数 464	在定义的循环时间期间集成实型值。
LFL（LIFO 加载） 参考页数 469	将 8 位、16 位、32 位或 64 位数据加载到调用 LIFO 堆栈的用户创建数组。
LFU（LIFO 卸载） 参考页数 472	从调用 LIFO 堆栈的用户创建的数组卸载 8 位、16 位、32 位或 64 位数据。使用 LFL 指令按照与加载相同的顺序卸载数据。
PWM 参考页数 474	打开或关闭配置的 PWM 通道的脉宽调制 (PWM) 输出。
SCALER 参考页数 477	根据输出范围调整输入值。
STACKINT 参考页数 479	管理整数值的堆栈。
LIMIT 参考页数 489	将整型值限制为给定的间隔。
TND 参考页数 488	停止用户程序扫描的当前循环。

DERIVATE

实型值在定义的循环时间内的差分。

操作详细信息：

- 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。
- 对时间执行微分计算以毫秒为基数。例如，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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

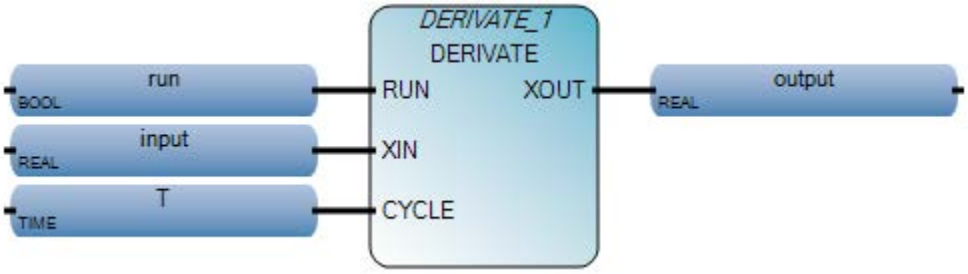
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



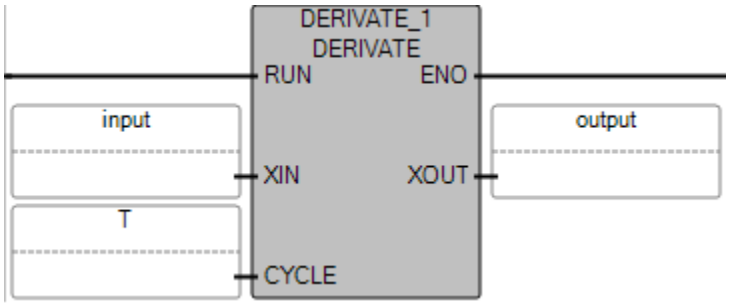
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
RUN	输入	BOOL	指示指令的操作模式。 TRUE - 正常（执行计算） FALSE - 复位
XIN	输入	REAL	定义对其执行微分计算的值。值必须为实型值。
CYCLE	输入	TIME	定义收集值的采样时间段。可能的时间段值范围为 0ms 至 49d17h2m47s294ms。
XOUT	输出	REAL	差分输出。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

DERIVATE 功能块图示例



DERIVATE 梯形图示例



DERIVATE 结构化文本示例




```
1  DERIVATE_1(run, input, T);
2  output := DERIVATE_1.XOUT;
```

(* 与之等效的 ST：DERIVATE1 是 DERIVATE 块的实例 *)

```
DERIVATE1(manual_mode, sensor_value, t#100ms);
derivated_value := DERIVATE1.XOUT;
```

FFL（FIFO 加载）

将 8 位、16 位、32 位或 64 位数据加载到调用 FIFO 堆栈的用户创建的一维数组。

操作详细信息：

- FFL 指令 - 非执行模式到执行模式
  - 当 Execute 从 FALSE 转变为 TRUE 时：
    - 错误条件已验证。
    - Src 的内容已加载到可用位置中的 FIFO 堆栈中；如果 Position 小于或等于零且小于 Length，则 Position 增加 1。
    - 如果 Length 等于 Position，则 Full 设置为 TRUE。如果 Position 小于或等于零且小于 Length，则 Full 为 FALSE。
    - 当指令成功执行时，Done 被设置。
  - 当 Execute 从 TRUE 转变为 FALSE 时：
    - Error、Done 和 ErrorID 设置为 FALSE。
    - 如果 Position 等于零，则 Empty 设置为 TRUE。
    - 当 Length 等于 Position 时，Full 设置为 TRUE。如果 Position 小于或等于零且小于 Length，则 Full 为 FALSE。
    - 未验证 FFL 错误条件。
  - 当 Execute 从 TRUE 转变为 TRUE 时：
    - 未执行加载操作。
    - 如果 Position 等于零，则 Empty 设置为 TRUE。
    - 如果 Position 小于或等于零且小于 Length，则 Full 设置为 FALSE。如果 Length 等于 Position，则 Full 设置为 TRUE。
    - 未验证 FFL 错误条件。
  - 当 Execute 从 FALSE 转变为 FALSE 时：
    - Error、Done 和 ErrorID 位设置为 FALSE。
    - Full 和 Empty 位保留来自先前执行状态的值。
  - 如果 Position 等于零，则 Empty 设置为 TRUE。
    - 如果 Length 等于 Position，则 Full 设置为 TRUE。如果 Position 小于或等于零且小于 Length，则 Full 为 FALSE。

- 未验证 FLL 错误条件。
- FFL 指令 - 执行模式到非执行模式：
  - Error、ErrorID、Empty 和 Full 保留 Execute 模式状态。

- 要为 FIFO 参数创建单个元素：
  - 非数组：
    - 允许将基于地址的变量（如 Fifo1）用于 FIFO。
    - Length 应配置为 1。
  - 一维数组：
    - 允许将基于地址的变量（如 Fifo1 或 Fifo1[0]）用于 FIFO。
    - Length 应配置为 1。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令块。 TRUE - 如果检测到上升沿，则启动 FFL 操作。 FALSE - 未检测到上升沿。
Src	输入	ANY_ELEMENTARY	Src 操作数是用于填充 FIFO 堆栈中当前可用位置的值的地址。 Scr 支持的基本数据类型： • DWORD、REAL、TIME、DATE、LWORD、ULINT、LINT、LREAL、BOOL、SINT、USINT、BYTE、INT、UINT、WORD、DINT、UDINT。 • 不支持字符串。 • 支持诸如 Array[1] 或 Array[Index] 等变量和一维数组。
FIFO	输入	ANY_ELEMENTARY	堆栈的起始地址。必须为 FFL 和 FFU 指令进行相同的 FIFO 配置。 FIFO 支持的基本数据类型： DWORD、REAL、TIME、DATE、LWORD、ULINT、LINT、LREAL、BOOL、SINT、USINT、BYTE、INT、UINT、WORD、DINT、UDINT。 • 不支持字符串。 • FIFO 仅支持一维数组。
FIFOCon	输入	FF_LF_CON	FIFO 配置和控制。必须为 FFL 和 FFU 指令配置相同的设置。 使用 FF_LF_CON 数据类型配置 Position 和 Length。

参数	参数类型	数据类型	描述
Done	输出	BOOL	指示 FFU 操作完成的时间。 TRUE - 操作已成功完成。 FALSE - 操作过程遇到错误条件或 FFU 指令未执行。
Empty	输出	BOOL	指示 FIFO 堆栈为空的时间。 TRUE - 当 Position 等于 0 时。 FALSE - 当 Position 不等于 0 时。
Full	输出	BOOL	指示 FIFO 堆栈已满的时间。 TRUE - 当 Length 等于 Position 时。 FALSE - 当 Position 大于或等于零且小于 Length 时。
Error	输出	BOOL	表示存在错误条件。 TRUE - 操作过程遇到错误。 FALSE - 操作过程成功完成或指令未执行。
ErrorID	输出	USINT	标识错误的唯一数字。这些错误在 FFL 错误代码中定义。

FF_LF_CON 数据类型

使用此表帮助确定 FF_LF_CON 数据类型的参数值。

参数	数据类型	描述
Length	UINT	用于 FIFO 操作的元素数量。最大限值为 1024。
Position	USINT	确定 FIFO 中下一个可用于输入或删除 Src 的位置。位置是数组的偏移量。 示例 1： • 用户配置的一维数组，arr[0..5]。初始位置被配置为 1。数据被推入 arr[1]，并且位置按"位置 +1"递增。 示例 2： • 用户将一维数组配置为 arr[1..5]。初始位置被配置为 1。数据被推入 arr[2]，并且位置按"位置 +1"递增。

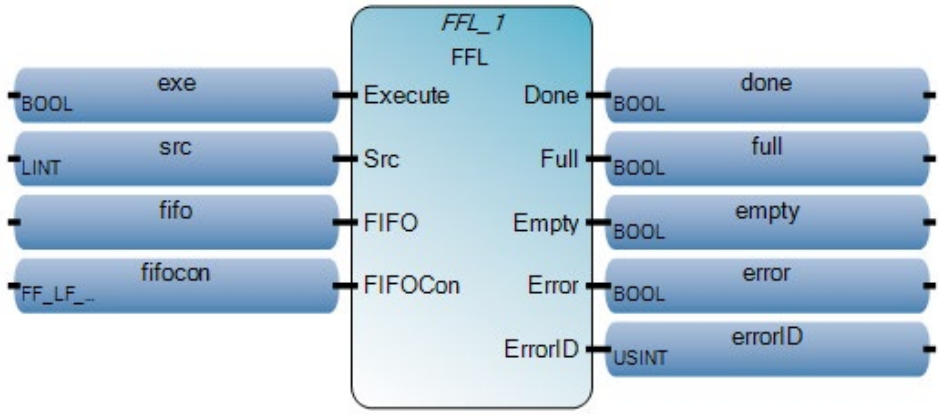
FFL 错误代码

使用此表确定 FFL 和 FFU 错误代码和描述。

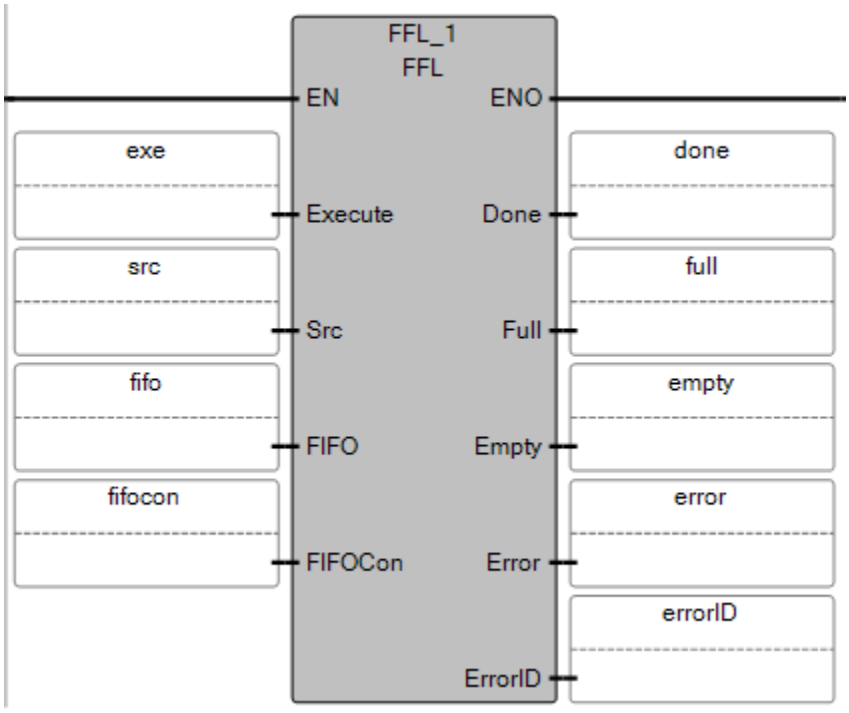
错误代码	错误描述
0	无错误。
1	不支持 FFL Src 数据类型。
2	不支持 FFU Dest 数据类型。
3	不支持 FIFO 数据类型。
4	Src 和 Dest 数据类型与 FIFO 数据类型不匹配。 更正措施： FFL Src 参数和 FFU Dest 参数数据类型应与 FIFO 数组数据类型匹配。
5	FIFO - 数组维度不受支持。 更正措施： FIFO 仅支持一维数组。
6	FIFOCon 控制长度超过 FIFO 数组大小。 更正措施： FIFOCon 控制长度不能超过 FIFO 数组大小。
7	FIFOCon 长度超过最大长度。
8	FIFOCon 长度为零。
9	FIFOCon 位置超过 FIFOCon 长度。

错误代码	错误描述
10	FFL 控制长度和位置相等。
11	FFU 控制位置为零。
12	不支持 FFL 或 FFU 数组维度。 更正措施： FFL 和 FFU 仅支持一维数组。
13	FFL 或 FFU DestOffset 超出 Dest 数组大小。

FFL 功能块图示例



FFL 梯形图示例



FFL 结构化文本示例

```
1  FFL_1(exe, src, fifo, fifocon);
2  done := FFL_1.Done;
3  full := FFL_1.Full;
4  empty := FFL_1.Empty;
5  error := FFL_1.Error;
6  errorID := FFL_1.ErrorID;
```

```
FFL_1(
```

```
void FFL_1(BOOL Execute, ANY_ELEMENTARY Src, ANY_ELEMENTARY[1..1] FIFO, FF_LF_CON FIFOCon)
类型： FFL, FIFO 加载
```

结果

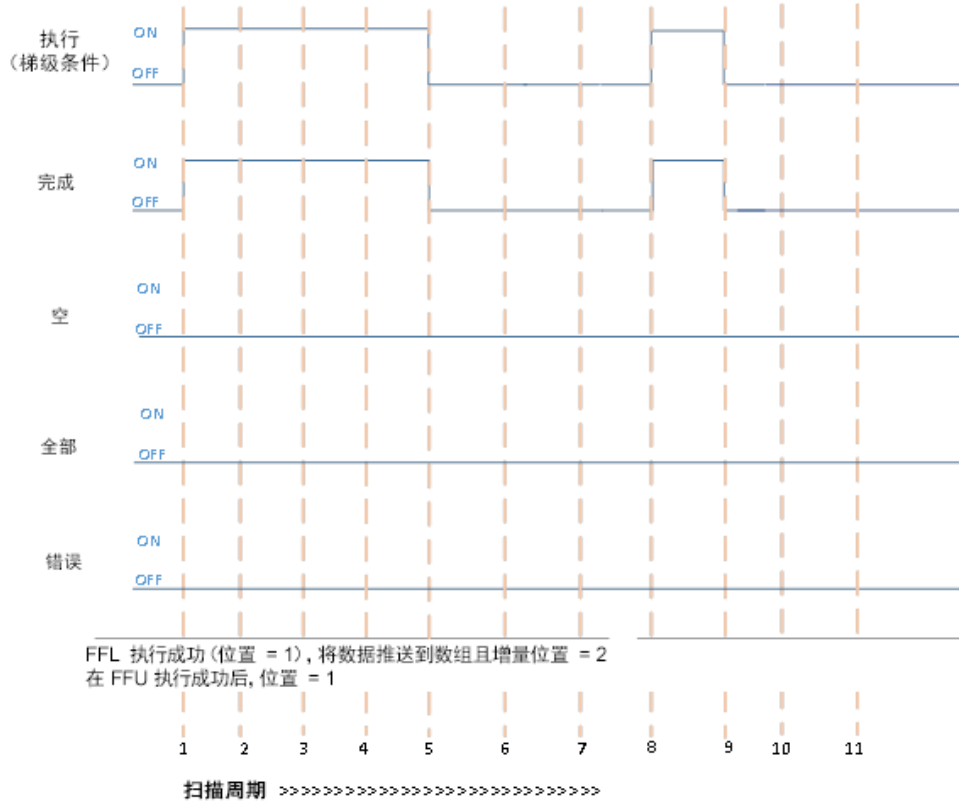


名称	逻辑值	实际值	初始值	锁定	数据类型	维度
src		不可用			BOOL	
full		不可用			BOOL	
fifocon	...	...	...		FF_LF_CON	
fifocon.Leng	10	不可用			UINT	
fifocon.Posi	1	不可用			UINT	
fifo	...	...	...		BOOL	[1..10]
fifo[1]	☑	不可用			BOOL	
fifo[2]		不可用			BOOL	
fifo[3]		不可用			BOOL	
fifo[4]		不可用			BOOL	
fifo[5]		不可用			BOOL	
fifo[6]		不可用			BOOL	
fifo[7]		不可用			BOOL	
fifo[8]		不可用			BOOL	
fifo[9]		不可用			BOOL	
fifo[10]		不可用			BOOL	
FFL_1	...	...	...		FFL	
exe		不可用			BOOL	
errorID	0	不可用			USINT	
error		不可用			BOOL	
empty		不可用			BOOL	
done		不可用			BOOL	

FFL 和 FFU 指令时间图

以下时序图示例说明了 FFL 参考页数 449（FIFO 加载）和 FFU 参考页数 457（FIFO 卸载）指令的执行情况。

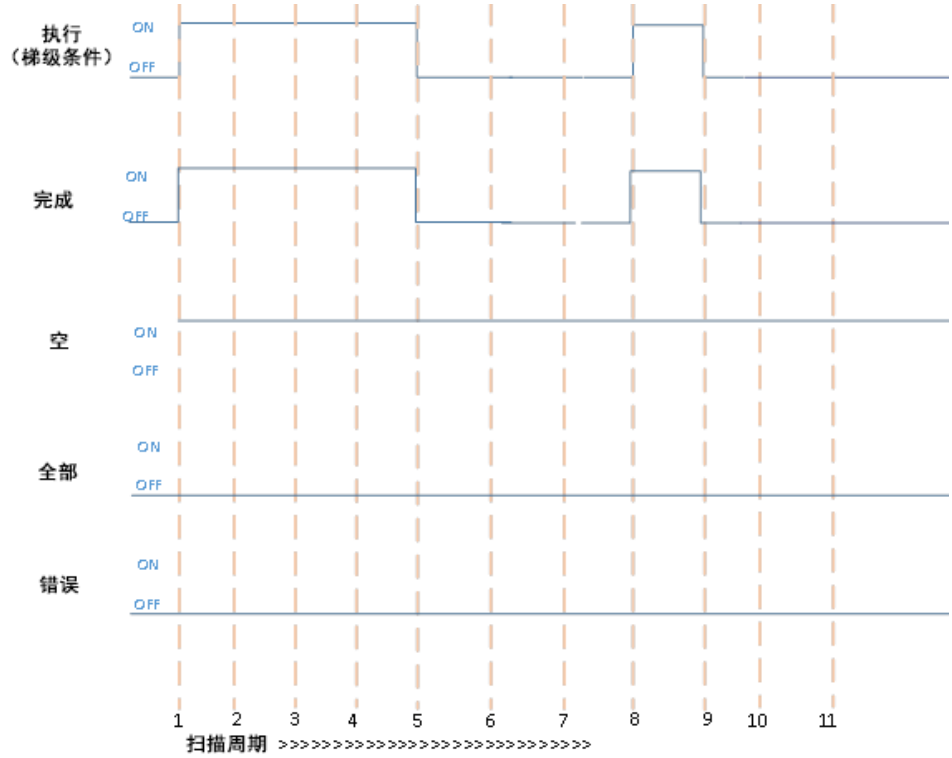
成功执行 FFL，然后成功执行 FFU



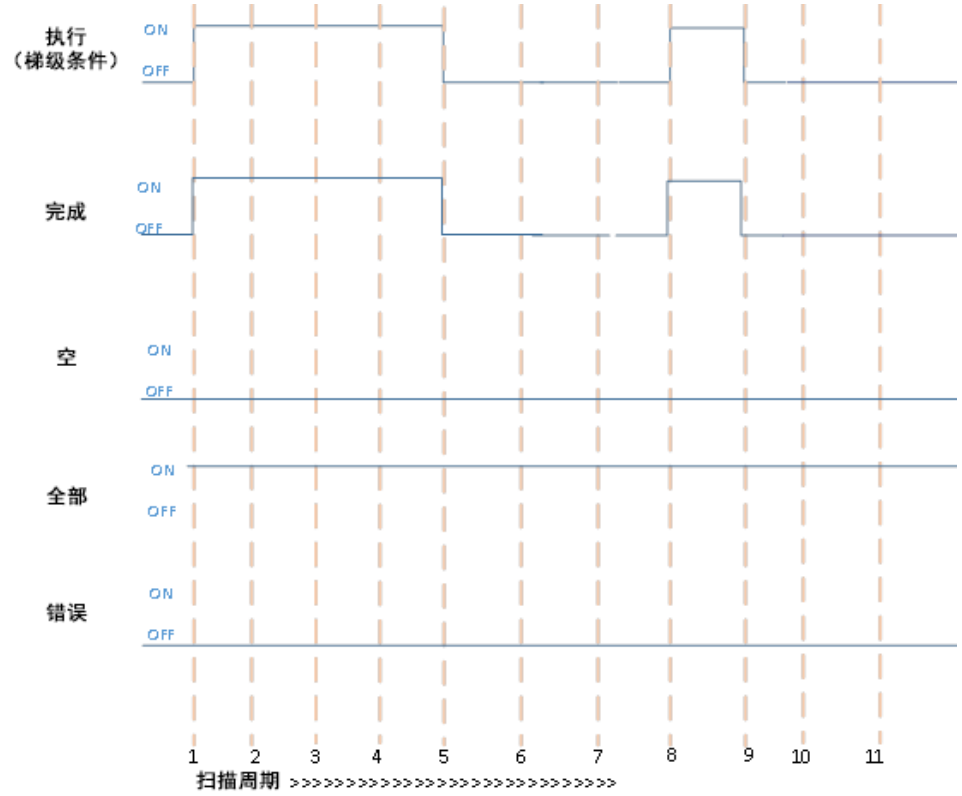
使用此表可帮助确定每个扫描周期的参数值。

扫描周期	描述
1	在下列情况下梯级条件变为 TRUE： • “执行”输入位为 TRUE。 • 将数据加载（推入）至 FIFO 堆栈。 • “完成”输出位为 TRUE。
2，3，4	梯级条件未发生更改。
5	在下列情况下梯级条件变为 FALSE： • “执行”位为 FALSE。 • “完成”输出位为 FALSE。
6，7	梯级条件未发生更改。 • “执行”位为 FALSE。 • “完成”输出位为 FALSE。
8	在下列情况下梯级变为 TRUE： • “执行”输入位为 TRUE。 • 从 FIFO 堆栈卸载数据。 • “完成”输出位为 TRUE。
9	梯级条件未发生更改。 • “执行”位为 FALSE。 • “完成”输出位为 FALSE。
10, 11	梯级条件未发生更改。

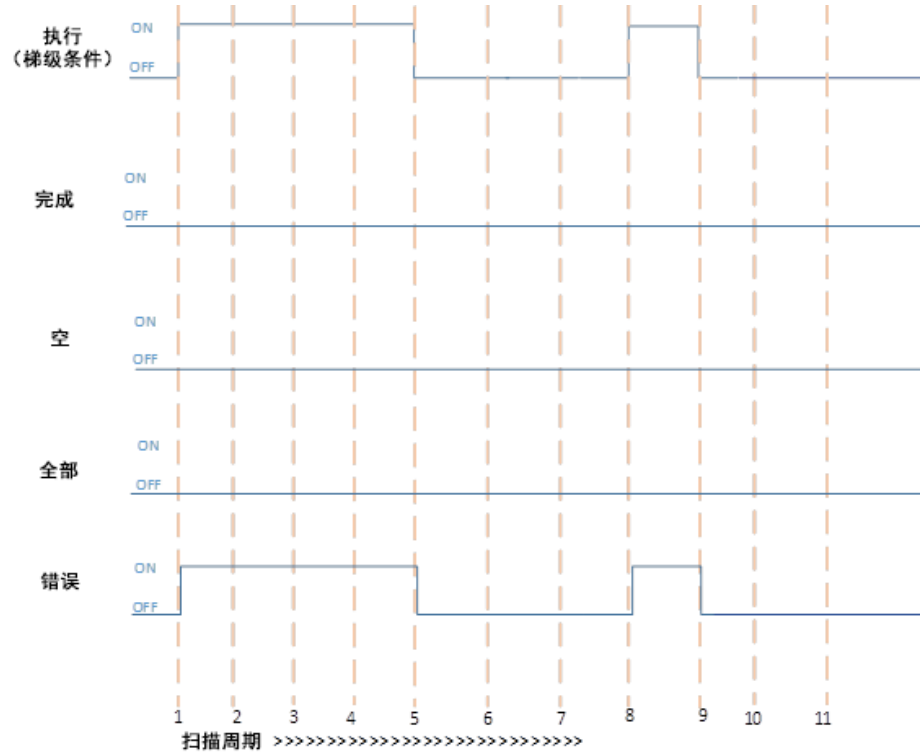
“Empty”位为 TRUE 时成功执行


执行
（梯级条件）
ON
OFF
完成
ON
OFF
空
ON
OFF
全部
ON
OFF
错误
ON
OFF
1
2
3
4
5
6
7
8
9
10
11
扫描周期 >>>>

“Empty”位为 TRUE 时成功执行


执行
（梯级条件）
ON
OFF
完成
ON
OFF
空
ON
OFF
全部
ON
OFF
错误
ON
OFF
1
2
3
4
5
6
7
8
9
10
11
扫描周期 >>>>

在执行 FFL 和 FFU 期间遇到错误



使用此表可帮助确定每个扫描周期的参数值。

扫描周期	描述
	在下列情况下梯级条件变为 TRUE： • “执行”输入位为 TRUE。开始执行。 • “错误”位为 TRUE。
2，3，4	梯级条件未发生更改。
5	在下列情况下梯级条件变为 FALSE： • “执行”位为 FALSE。 • Error 和 ErrorID 位为 FALSE。
6，7	梯级条件未发生更改。
8	在下列情况下梯级变为 TRUE： • “执行”输入位为 TRUE。开始执行。 • “错误”位为 TRUE。
9	在下列情况下梯级条件变为 FALSE： • “执行”位为 FALSE。 • Error 和 ErrorID 位为 FALSE。
10, 11	梯级条件未发生更改。

FFU（FIFO 卸载）

以使用 FFL 指令加载数据的相同顺序，从调用 FIFO（先进先出）堆栈的用户创建的一维数组卸载 8 位、16 位、32 位或 64 位数据。

操作详细信息：

- FFU 指令 - 非执行模式到执行模式：
  - 当 Execute 从 FALSE 转变为 TRUE 时：
    - FFU 错误条件已验证。
    - 如果 Position 大于零且小于或等于 Length，则卸载 FIFO 堆栈中零位置的内容。
    - 其余元素将向零移动一个位置，并且 FIFO 堆栈的最高元素设置为零，Position 递减 1。
    - 如果 Position 等于零，则 Empty 设置为 TRUE。
    - 当指令成功执行时，Done 被设置。
  - 当 Execute 从 TRUE 转变为 FALSE 时：
    - Error、Done 和 ErrorID 设置为 FALSE。
    - 如果 Position 等于零，则 Empty 设置为 TRUE。
    - 如果 Length 等于 Position，则 Full 设置为 TRUE。如果 Position 小于或等于零且小于 Length，则 Full 设置为 FALSE。
    - 未验证 FFU 错误条件。
  - 当 Execute 从 TRUE 转变为 TRUE 时：
    - 未执行卸载操作。
    - 如果 Position 等于零，则 Empty 设置为 TRUE。
    - 如果 Length 等于 Position，则 Full 设置为 TRUE。
    - 未验证 FFU 错误条件。
  - 当 Execute 从 FALSE 转变为 FALSE 时：
    - Error、Done 和 ErrorID 设置为 FALSE。
    - 如果 Position 等于零，则 Empty 设置为 TRUE。
    - 如果 Length 等于 Position，则 Full 设置为 TRUE。如果 Position 小于或等于零且小于 Length，则 Full 设置为 FALSE。
    - 未验证 FLU 错误条件。
- FFU 指令 - 执行模式到非执行模式：
  - Error、ErrorID、Empty 和 Full 保留 Execute 模式状态。
- 要为 FIFO 参数创建单个元素：
  - 非数组：
    - 允许将基于地址的变量（如 Fifo1）用于 FIFO。
    - Length 应配置为 1。
  - 一维数组：
    - 允许将基于地址的变量（如 Fifo1 或 Fifo1[0]）用于 FIFO。
    - Length 应配置为 1。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令块。 TRUE - 如果检测到上升沿，则启动 FFU 操作。 FALSE - 未检测到上升沿。
Dest	输入	ANY_ELEMENTARY	保持 FIFO 堆栈中存在的值。 Dest 支持的基本数据类型： • DWORD、REAL、TIME、DATE、LWORD、ULINT、LINT、LREAL、BOOL、SINT、USINT、BYTE、INT、UINT、WORD、DINT、UDINT。 • 不支持字符串。 • 仅支持诸如 Array[1] 或 Array[Index] 等变量和一维数组。
DestOffset	输入	UINT	目标元素偏移。 如果目标是一维数组数据类型，则元素偏移；否则，偏移应设置为 0。 对于数组数据类型，要卸载到第一个元素，应将偏移设置为 0。
FIFO	输入	ANY_ELEMENTARY	堆栈的起始地址。必须在 FFL 和 FFU 指令中进行相同的 FIFO 配置。 • FIFO 支持的基本数据类型： DWORD、REAL、TIME、DATE、LWORD、ULINT、LINT、LREAL、BOOL、SINT、USINT、BYTE、INT、UINT、WORD、DINT、UDINT。 • 不支持字符串。 • FIFO 仅支持一维数组。
FIFOCon	输入	FF_LF_CON	FIFO 配置和控制。必须为 FFL 和 FFU 指令配置相同的设置。 使用 FF_LF_CON 数据类型配置 Position 和 Length。
Done	输出	BOOL	指示 FFU 操作完成的时间。 TRUE - 操作已成功完成。 FALSE - 操作过程遇到错误条件或 FFU 指令未执行。
Full	输出	BOOL	指示 FIFO 堆栈已满的时间。 TRUE - 当 Length 等于 Position 时。 FALSE - 当 Position 大于零且小于 Length 时。
Empty	输出	BOOL	指示 FIFO 堆栈为空的时间。 TRUE - 当 Position 等于 0 时。 FALSE - 当 Position 不等于 0 时。
Error	输出	BOOL	表示存在错误条件。 TRUE - 操作过程遇到错误。 FALSE - 操作过程成功完成或指令未执行。
ErrorID	输出	USINT	标识错误的唯一数字。这些错误在 FFU 错误代码中定义。

FF_LF_CON 数据类型

使用此表帮助确定 FF_LF_CON 数据类型的参数值。

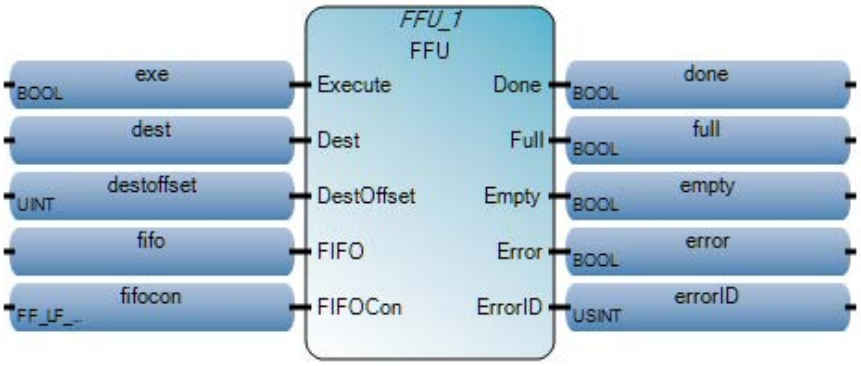
参数	数据类型	描述
Length	UINT	用于 FIFO 操作的元素数量。最大限值为 1024。
Position	USINT	确定 FIFO 中下一个可用于输入或删除 Src 的位置。位置是数组的偏移量。 示例 1： • 用户配置的一维数组，arr[0..5]。初始位置被配置为 1。数据被推入 arr[1]，并且位置按"位置 +1"递增。 示例 2： • 用户将一维数组配置为 arr[1..5]。初始位置被配置为 1。数据被推入 arr[2]，并且位置按"位置 +1"递增。

FFU 错误代码

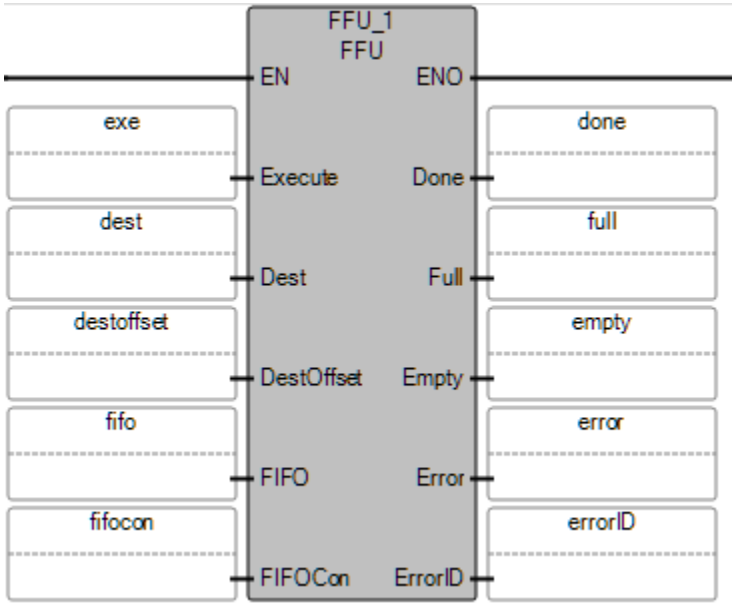
使用此表确定 FFL 和 FFU 错误代码和描述。

错误代码	错误描述
0	无错误。
1	不支持 FFL Src 数据类型。
2	不支持 FFU Dest 数据类型。
3	不支持 FIFO 数据类型。
4	Src 和 Dest 数据类型与 FIFO 数据类型不匹配。 更正措施： FFL Src 参数和 FFU Dest 参数数据类型应与 FIFO 数组数据类型匹配。
5	FIFO - 数组维度不受支持。 更正措施： FIFO 仅支持一维数组。
6	FIFOCon 控制长度超过 FIFO 数组大小。 更正措施： FIFOCon 控制长度不能超过 FIFO 数组大小。
7	FIFOCon 长度超过最大长度。
8	FIFOCon 长度为零。
9	FIFOCon 位置超过 FIFOCon 长度。
10	FFL 控制长度和位置相等。
11	FFU 控制位置为零。
12	不支持 FFL 或 FFU 数组维度。 更正措施： FFL 和 FFU 仅支持一维数组。
13	FFL 或 FFU DestOffset 超出 Dest 数组大小。

FFU 功能块图示例



FFU 梯形图示例



FFU 结构化文本示例

```
1  FFU_1(exe, dest, destoffset, fifo, fifocon);
2  done := FFU_1.Done;
3  full := FFU_1.Full;
4  empty := FFU_1.Empty;
5  error := FFU_1.Error;
6  errorID := FFU_1.ErrorID;
```



结果



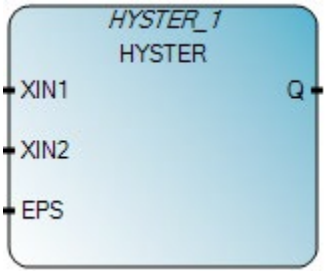
名称	逻辑值	实际值	初始值	锁定	数据类型	维
dest	...	...	...		BOOL	[1..
destoffset	0	不可用			UINT	
done		不可用			BOOL	
empty		不可用			BOOL	
error		不可用			BOOL	
errorID	0	不可用			USINT	
exe		不可用			BOOL	
FFU_1	...	...	...		FFU	
fifo	...	...	...		BOOL	[1..
fifo[1]		不可用			BOOL	
fifo[2]		不可用			BOOL	
fifo[3]		不可用			BOOL	
fifo[4]		不可用			BOOL	
fifo[5]		不可用			BOOL	
fifo[6]		不可用			BOOL	
fifo[7]		不可用			BOOL	
fifo[8]		不可用			BOOL	
fifo[9]		不可用			BOOL	
fifo[10]		不可用			BOOL	
fifocon	...	...	...		FF_LF_CON	
fifocon.Leng	10	不可用			UINT	
fifocon.Posi	0	不可用			UINT	
full		不可用			BOOL	

HYSTER（滞后）

在不同实型值上的布尔迟滞。将输入的当前值与通过将按滞后测量的延迟历史量添加到输入的预期值所确定的上限进行比较，并评估当前值是否超出该限制。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

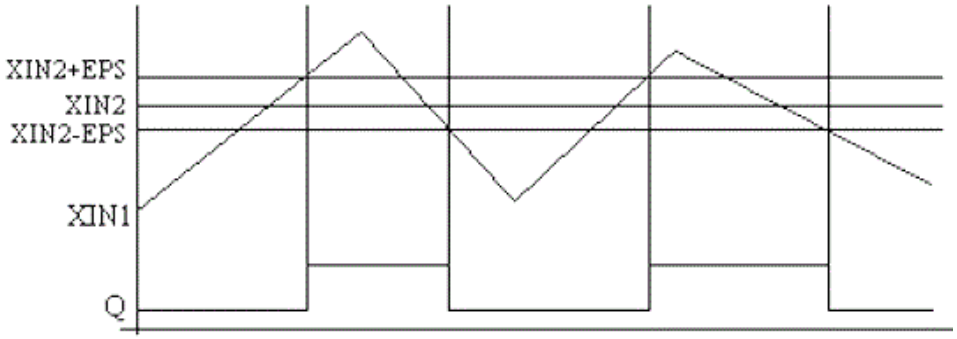


使用此表有助于确定该指令的参数值。

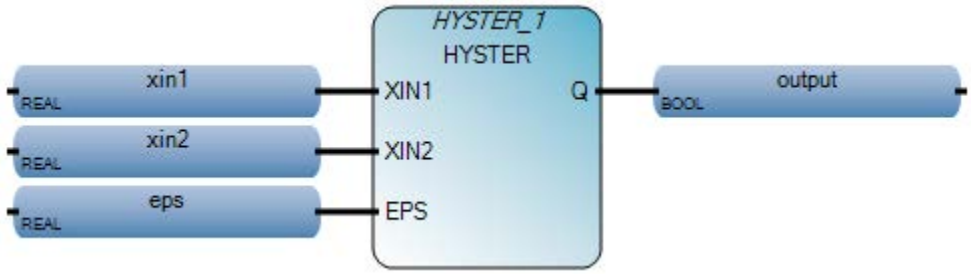
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行指令块。 FALSE - 不执行指令块。 仅适用于梯形图编程。
XIN1	输入	REAL	任何实型值。
XIN2	输入	REAL	测试输入值 XIN1 是否已超过为此输入 XIN2 + EPS 定义的上限。
EPS	输入	REAL	滞后值（必须大于 0）。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。
Q	输出	BOOL	HYSTER 指令的结果。 TRUE - 输入超出上限，但不低于下限。 FALSE - 输入未超过上限。

HYSTER 时间图示例

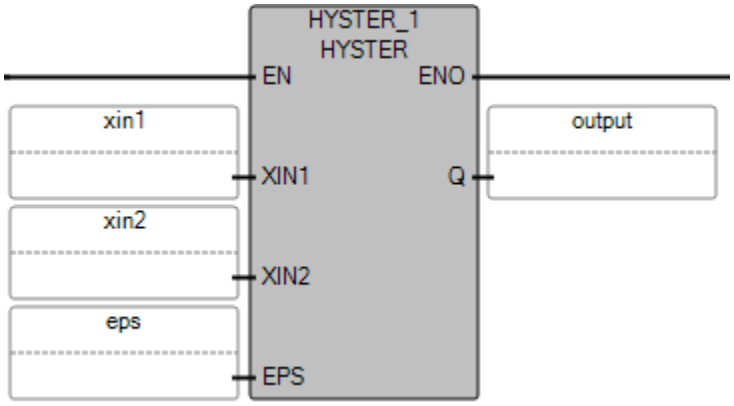
在下图中，由于电机摩擦时间段超过 5 秒，因此 HYSTER 用于评估性能滞后。指令每隔 10 毫秒运行。在启动阶段，电机运转效率较低。



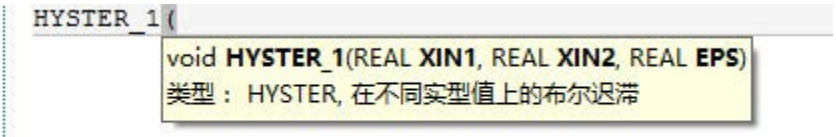
HYSTER 功能块图示例



HYSTER 梯形图示例



HYSTER 结构化文本示例




```
1  xin1 := 10.0;
2  xin2 := 1.0;
3  eps := 1.0;
4  HYSTER_1(xin1, xin2, eps);
5  output := HYSTER_1.Q;
```

INTEGRAL

在定义的循环时间期间集成实型值。

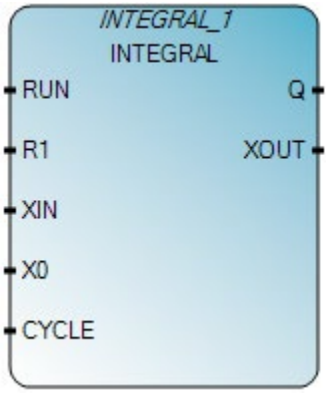
操作详细信息：

- 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。首次将控制器从编程转换到运行模式以及启动新的积分时，使用 R1 参数可清除积分值。
- 建议不将可选的 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。
- 积分以毫秒为时间基准执行（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都会造成 Integral 输出的整体不精确性，如 XIN 与功能块执行示例同步和不同步所示。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

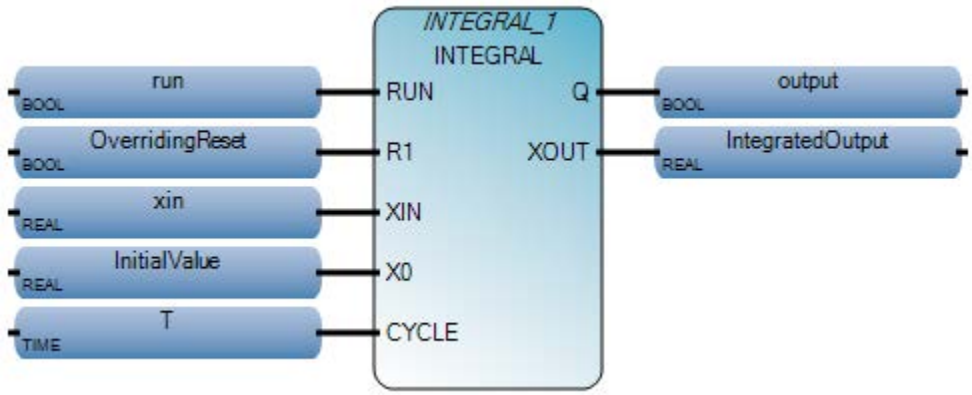
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



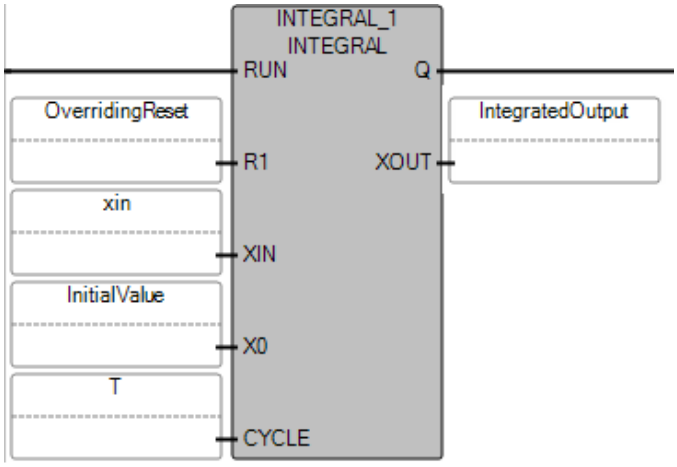
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
RUN	输入	BOOL	模式：TRUE = 积分/FALSE = 保留。
R1	输入	BOOL	取消重置。
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 49d17h2m47s294ms。
Q	输出	BOOL	非 R1。
XOUT	输出	REAL	积分输出。

INTEGRAL 功能块图示例



INTEGRAL 梯形图示例



INTEGRAL 结构化文本示例




```
run := TRUE;
xin := 10.0;
InitialValue := 5.0;
T := T#10s;
INTEGRAL_1(run, OverridingReset, xin, InitialValue, T);
output := NOT OverridingReset;
IntegratedOutput := INTEGRAL_1.XOUT;
```

(* 与之等效的 ST：INTEGRAL1 是 INTEGRAL 块的实例 *)

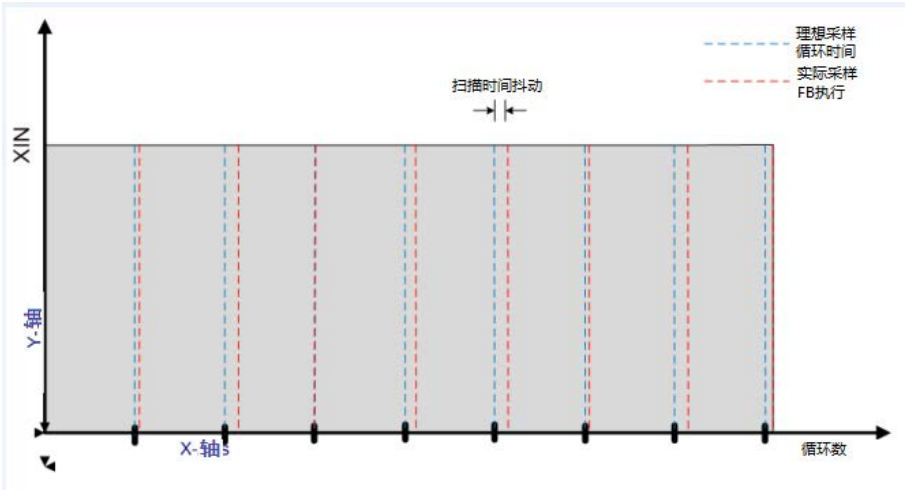
```
INTEGRAL1(manual_mode, NOT(manual_mode), sensor_value,
init_value, t#100ms);
controlled_value := INTEGRAL1.XOUT;
```

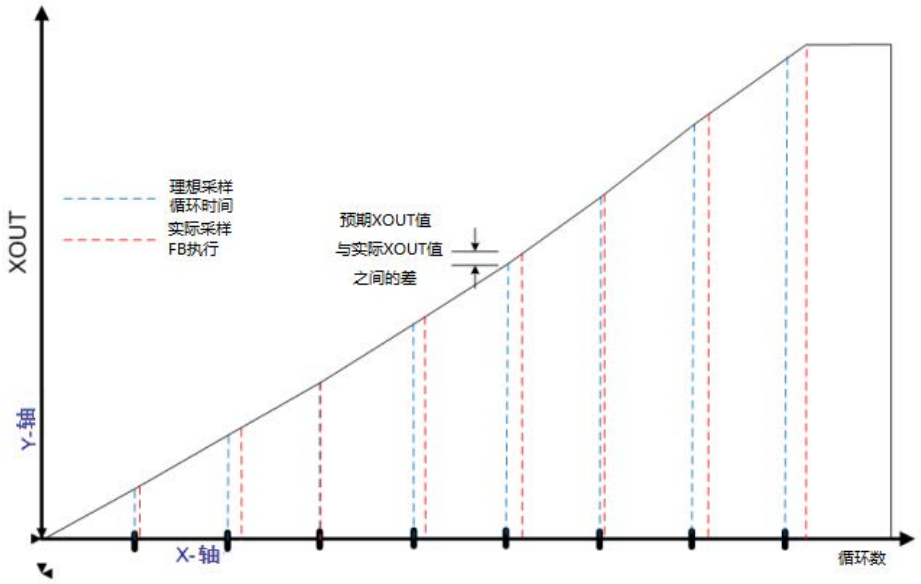
结果



XIN 与功能块执行同步的示例

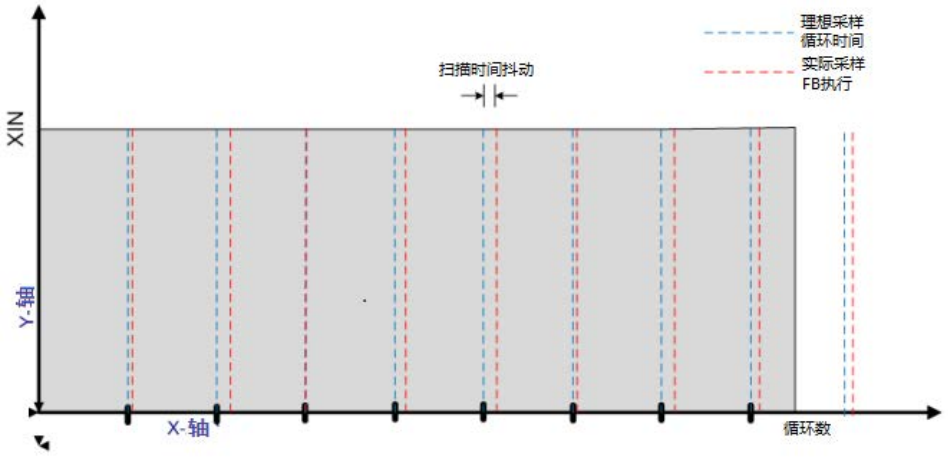
下图显示了扫描时间抖动对 XOUT 值的影响：

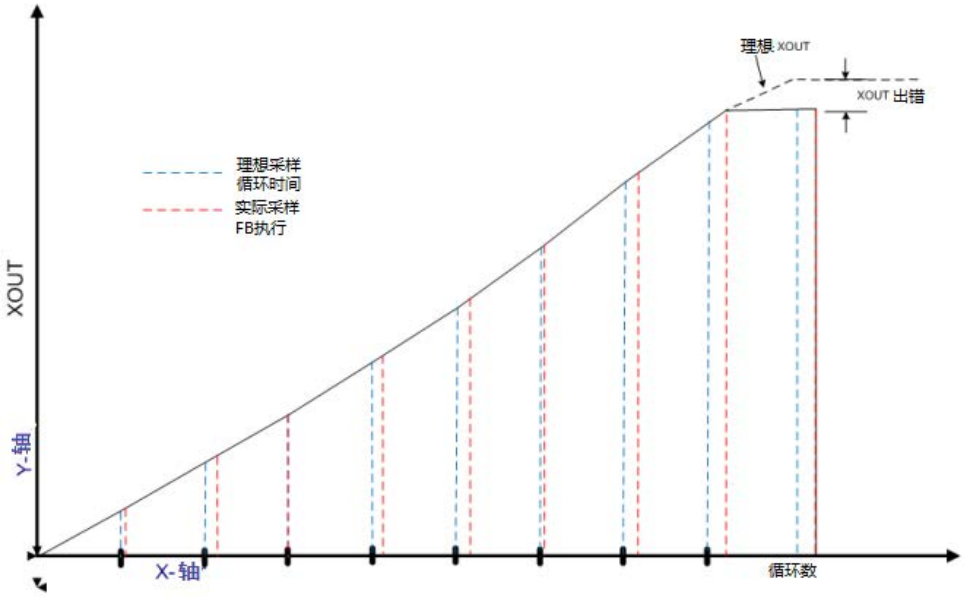




XIN 与功能块执行不同步的示例

下图显示了将错误引入 Integral 功能块的 XOUT 值中的示例：



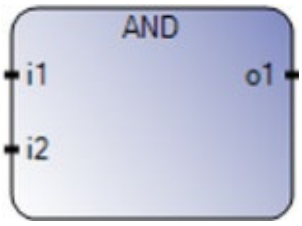


AND

在两个或多个值之间执行布尔与操作。可以表示为 &。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
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 结构化文本示例

```
(* 与之等效的 ST：*)
   bo10 := bi101 AND NOT (bi102);
   bo5 := (bi51 AND bi52) AND bi53;
```

LFL（LIFO 加载）

LFL 指令用于将数据（8 位、16 位、32 位和 64 位）加载到调用 LIFO 堆栈的用户创建的一维数组。LFL 和 LFU 参考页数 472指令成对使用。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令块。 TRUE - 如果检测到上升沿，则启动 LFL 操作。 FALSE - 未检测到上升沿。
Src	输入	ANY_ELEMENTARY	Src 操作数是用于填充 LIFO 堆栈中当前可用位置的值的地址。 支持的元素数据类型： • DWORD、REAL、TIME、DWORD、REAL、TIME、DATE、LWORD、ULINT、LINT、LREAL、BOOL、SINT、USINT、BYTE、INT、UINT、WORD、DINT、UDINT。\ • 诸如 Array[1] 或 Array[Index] 等变量和一维数组。 • 不支持字符串
LIFO	输入	ANY_ELEMENTARY	堆栈的起始地址。必须为 LFL 和 LFU 指令进行相同的 LIFO 配置。 支持的元素数据类型： • DWORD、REAL、TIME、DWORD、REAL、TIME、DATE、LWORD、ULINT、LINT、LREAL、BOOL、SINT、USINT、BYTE、INT、UINT、WORD、DINT、UDINT。 • 仅支持一维。 • 不支持字符串。
LIFOCon	输入	FF_LF_CON	LIFO 配置和控制。必须为 LFL 和 LFU 指令配置相同的配置。使用 FF_LF_CON 数据类型配置 Position 和 Length。
Full	输出	BOOL	指示 LIFO 堆栈已满的时间。 TRUE - 当 Length 等于 Position 时。 FALSE - 当 Position 大于或等于零且小于 Length 时。

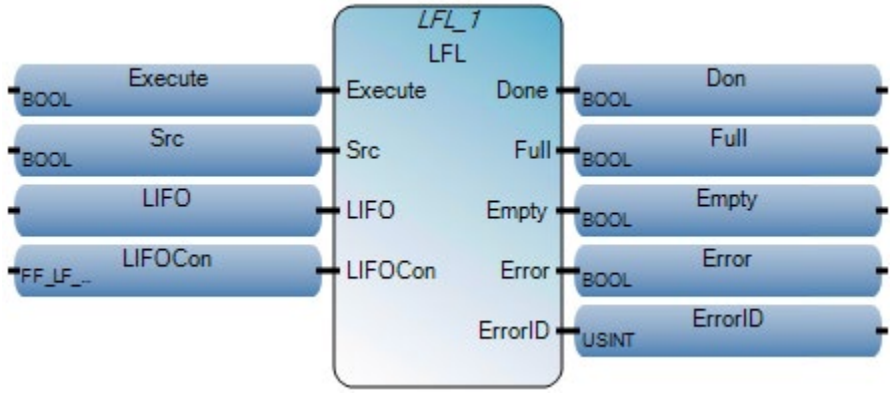
参数	参数类型	数据类型	描述
Empty	输出	BOOL	指示 LIFO 堆栈为空的时间。 TRUE - 当 Position 等于零时。 FALSE - 当 Position 不等于零时。
Error	输出	BOOL	表示存在错误条件。 TRUE - 操作过程遇到错误。 FALSE - 操作过程成功完成或指令未执行。
ErrorID	输出	USINT	标识错误的唯一数字。这些错误在 LFL 错误代码中定义。
Done	输出	BOOL	指示操作完成的时间。 TRUE - 操作过程成功完成。 FALSE - 操作过程遇到错误条件或 LFL 指令未执行。

LFL 错误代码

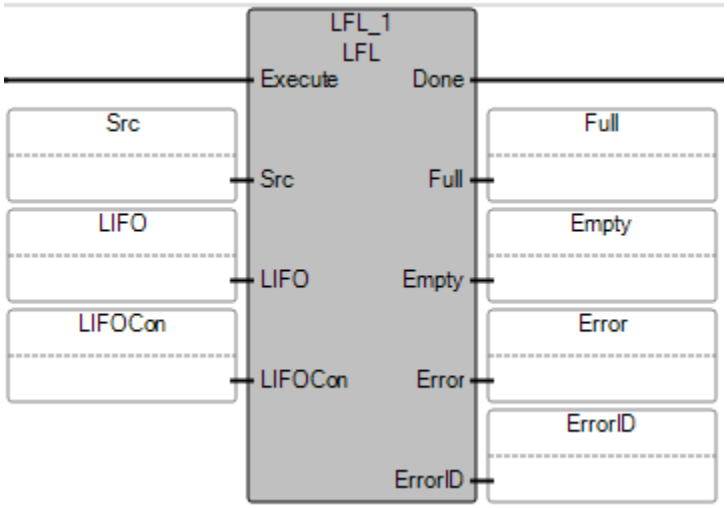
使用此表确定 LFL 和 LFU 错误代码和描述。

错误代码	错误描述
0	无错误。
1	不支持 LFL Src 数据类型。
2	不支持 LFU Dest 数据类型。
3	不支持 LIFO 数据类型。
4	Src 和 Dest 数据类型与 LIFO 数据类型不匹配。 更正措施： LFL Src 参数和 LFU Dest 参数数据类型应与 LIFO 数组数据类型匹配。
5	LIFO - 数组维度不受支持。 更正措施： LIFO 仅支持一维数组。
6	LIFOCon 控制 Length 超过 LIFO 数组大小。 更正措施： LIFOCon 控制 Length 不得超过 LIFO 数组大小。
7	LIFOCon Length 超过最大长度。
8	LIFOCon Length 为零。
9	LIFOCon Position 超过 LIFOCon Length。
10	LFL 控制 Length 和 Position 相等。
11	LFU 控制 Position 为零。
12	不支持 LFL 或 LFU 数组维度。 更正措施： LFL 和 LFU 仅支持一维数组。
13	LFL 或 LFU DestOffset 超过 Dest 数组大小。

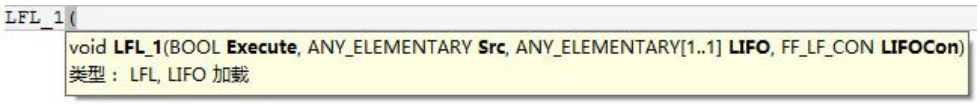
LFL 功能块图示例



LFL 梯形图示例



LFL 结构化文本示例




```
1  LFL_1(exe, Src, fifo, fifocon);
2  Done:=LFL_1.Done;
3  Full:=LFL_1.Full;
4  Empty:=LFL_1.Empty;
5  Error:=LFL_1.Error;
6  ErrorID:=LFL_1.ErrorID;
```

LFU（LIFO 卸载）

LFU 指令从调用 LIFO 堆栈的用户创建的一维数组卸载数据（8 位、16 位、32 位、64 位）。LFU 和 LFL 参考页数 469指令成对使用。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

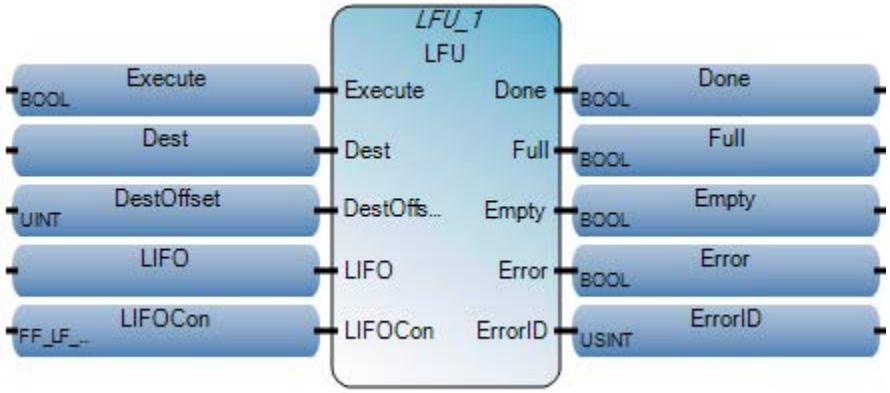
参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令块。 TRUE - 如果检测到上升沿，则启动 LFU 操作。 FALSE - 未检测到上升沿。
Dest	输入	ANY_ELEMENTARY	保持 LIFO 堆栈中存在的值。 Dest 支持的基本数据类型： • DWORD、REAL、TIME、DATE、LWORD、ULINT、LINT、LREAL、BOOL、SINT、USINT、BYTE、INT、UINT、WORD、DINT、UDINT。 • 不支持字符串。 • 仅支持诸如 Array[1] 或 Array[Index] 等变量和一维数组。
DestOffset	输入	UINT	目标元素偏移。 如果目标是数组数据类型，则为元素偏移；否则，偏移应设置为 0。 对于数组数据类型，要卸载到第一个元素，应将偏移设置为 0。
LIFO	输入	ANY_ELEMENTARY	堆栈的起始地址。必须为 LFL 和 LFU 指令进行相同的 LIFO 配置。 支持的元素数据类型： • DWORD、REAL、TIME、DWORD、REAL、TIME、DATE、LWORD、ULINT、LINT、LREAL、BOOL、SINT、USINT、BYTE、INT、UINT、WORD、DINT、UDINT。 • 仅支持一维。 • 不支持字符串。
LIFOCon	输入	FF_LF_CON	LIFO 配置和控制。必须为 LFL 和 LFU 指令配置相同的配置。使用 FF_LF_CON 数据类型配置 Position 和 Length。
Full	输出	BOOL	指示 LIFO 堆栈已满的时间。 TRUE - 当 Length 等于 Position 时。 FALSE - 当 Position 大于或等于零且小于 Length 时。
Error	输出	BOOL	表示存在错误条件。 TRUE - 操作过程遇到错误。 FALSE - 操作过程成功完成或指令未执行。
ErrorID	输出	USINT	标识错误的唯一数字。这些错误在 LFU 错误代码中定义。
Done	输出	BOOL	指示操作完成的时间。 TRUE - 操作过程成功完成。 FALSE - 操作过程遇到错误条件或 LFU 指令未执行。

LFU 错误代码

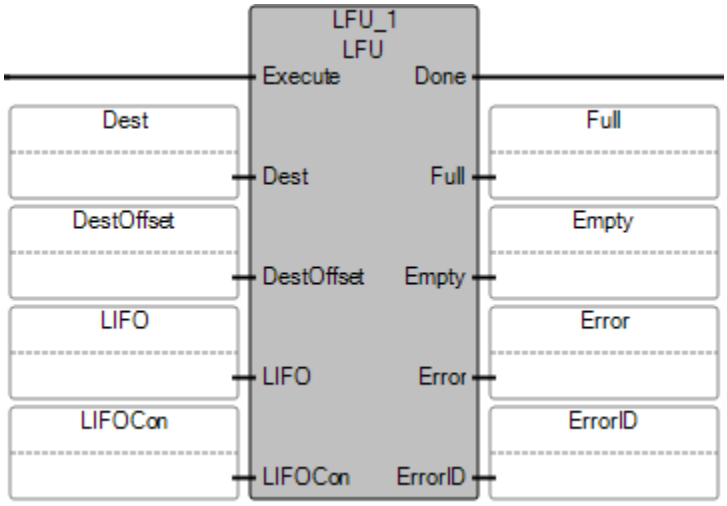
使用此表确定 LFL 和 LFU 错误代码和描述。

错误代码	错误描述
0	无错误。
1	不支持 LFL Src 数据类型。
2	不支持 LFU Dest 数据类型。
3	不支持 LIFO 数据类型。
4	Src 和 Dest 数据类型与 LIFO 数据类型不匹配。 更正措施： LFL Src 参数和 LFU Dest 参数数据类型应与 LIFO 数组数据类型匹配。
5	LIFO - 数组维度不受支持。 更正措施： LIFO 仅支持一维数组。
6	LIFOCon 控制 Length 超过 LIFO 数组大小。 更正措施： LIFOCon 控制 Length 不得超过 LIFO 数组大小。
7	LIFOCon Length 超过最大长度。
8	LIFOCon Length 为零。
9	LIFOCon Position 超过 LIFOCon Length。
10	LFL 控制 Length 和 Position 相等。
11	LFU 控制 Position 为零。
12	不支持 LFL 或 LFU 数组维度。 更正措施： LFL 和 LFU 仅支持一维数组。
13	LFL 或 LFU DestOffset 超过 Dest 数组大小。

LFU 功能块图示例



LFU 梯形图示例



LFU 结构化文本示例




```
1  LFU_1(exe, Dest, destoffset, lifo, lifocon);
2  Done:=LFU_1.Done;
3  Full:=LFU_1.Full;
4  Empty:=LFU_1.Empty;
5  Error:=LFU_1.Error;
6  ErrorID:=LFU_1.ErrorID;
```

PWM（脉宽调制）

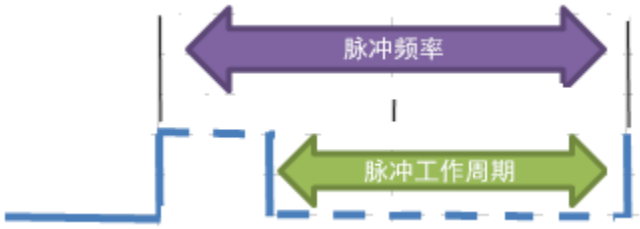
针对配置的 PWM 通道打开或关闭 PWM（脉宽调制）输出。

该指令块与 Micro820 2080-LC20-20QBB 控制器配合使用，并使用嵌入式输出通道 6 支持 PWM 单通道。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令仅适用于 Micro820 控制器。



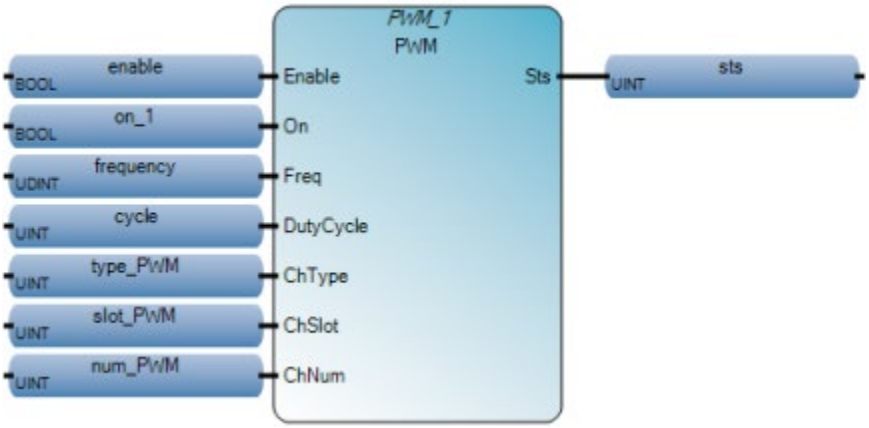


使用此表有助于确定该指令的参数值。

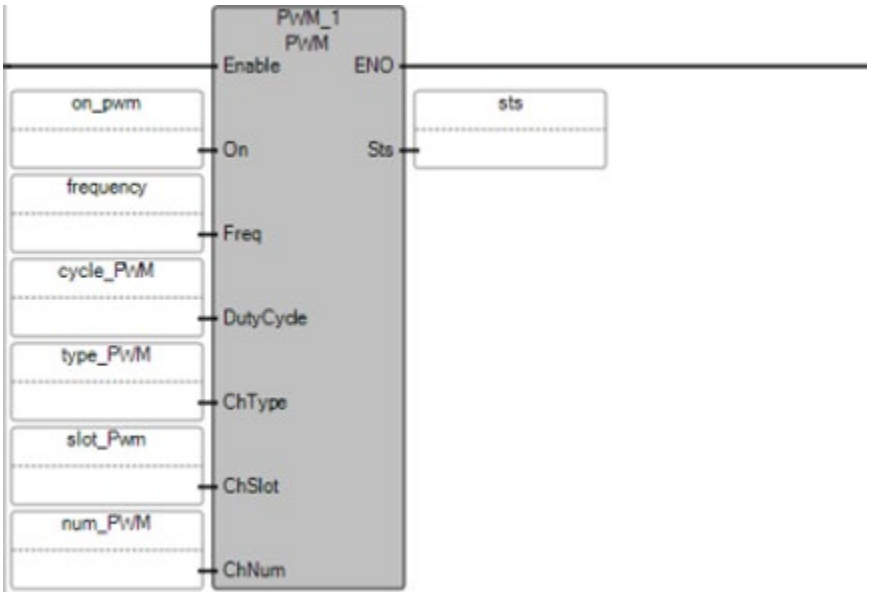
参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。该级别为指令块触发。 TRUE - 更新 Sts。根据"打开"输入参数和有效配置，PWM 为活动或不活动。 FALSE - 仅更新 Sts。PWM 状态（活动或不活动）未受影响。
On	输入	BOOL	打开/活动或关闭/不活动 PWM 输出。 TRUE - 根据最新有效配置，PWM 输出为活动或继续为活动。当 PWM 为活动时，输出 LED 亮起，即使工作周期设置为 0%。 FALSE - 仅当配置也有效时 PWM 输出为不活动。
Freq	输入	UDINT	脉冲频率。 • 1 - 100000 Hz
DutyCycle	输入	UINT	脉冲工作周期。 • 0 - 1000 (0% - 100%)
ChType	输入	UINT	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
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

参数	参数类型	数据类型	描述
Sts	输出	UINT	PWM 状态代码： • 00 - 未启用功能块（无操作。） • 01 - PWM 配置成功。 • 02 - 工作周期无效。 • 03 - 频率无效。 • 04 - 通道类型无效。 • 05 - 通道插槽无效。 • 06 - 通道编号无效。 • 07 - 目录无效。所使用的目录不支持 PWM 功能。

PWM 功能块图示例



PWM 梯形图示例



PWM 结构化文本示例

```
PWM_1(
    void PWM_1(BOOL Enable, BOOL On, UDINT Freq, UINT DutyCycle, UINT ChType, UINT ChSlot, UINT ChNum)
    类型： PWM, 启用 PWM 输出。

1  PWM (EN, Enable, On, Freq, DutyCycle, ChType, ChSlot, ChNum);
2  output := PWM.ENO
3  sts := PWM.Sts
```

SCALER（刻度盘）

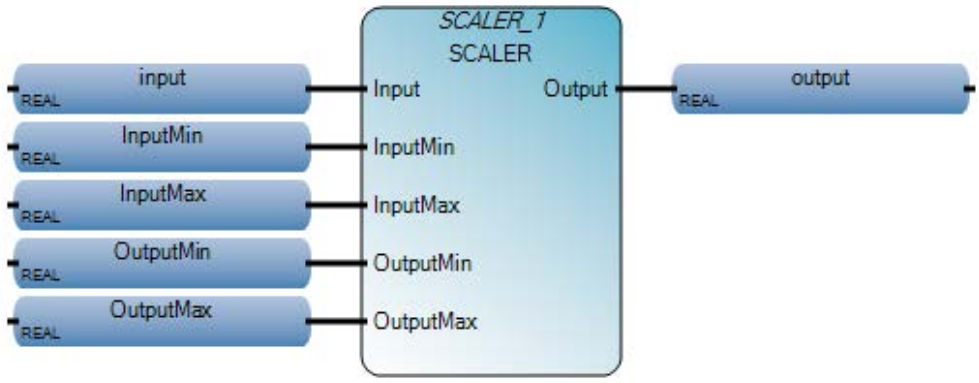
支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

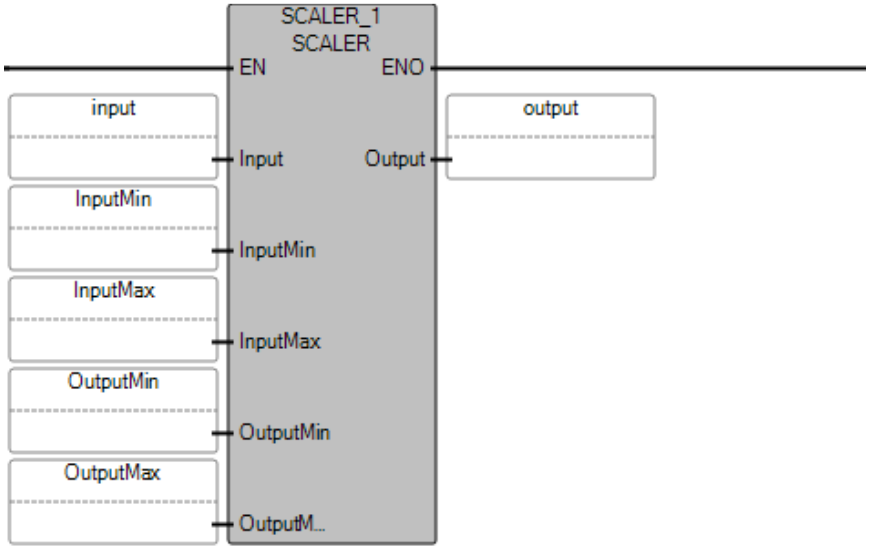
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令块。 TRUE - 执行调整方程。 FALSE - 没有调整方程。 仅适用于梯形图编程。
Input	输入	REAL	输入信号。输入不受 InputMin 和 InputMax 限制。要限制输入，需要在将其输入到 SCALER 指令之前使用 LIM_ALRM 指令以调整输入。
InputMin	输入	REAL	确定斜率和偏移值。
InputMax	输入	REAL	确定斜率和偏移值。
OutputMin	输入	REAL	确定斜率和偏移值。
OutputMax	输入	REAL	确定斜率和偏移值。
Output	输出	REAL	已缩放输出。输出未被钳制在 OutputMin 和 OutputMax 之间。
ENO	输出	BOOL	启用“输出”。 仅适用于梯形图编程。

SCALER 功能块图示例



SCALER 梯形图示例



SCALER 结构化文本示例

```
SCALER_1(
```

void **SCALER_1**(REAL **Input**, REAL **InputMin**, REAL **InputMax**, REAL **OutputMin**, REAL **OutputMax**)
类型： SCALER, 根据输出范围调整输入值。

```
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;
```

(* 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



STACKINT（堆栈整数）

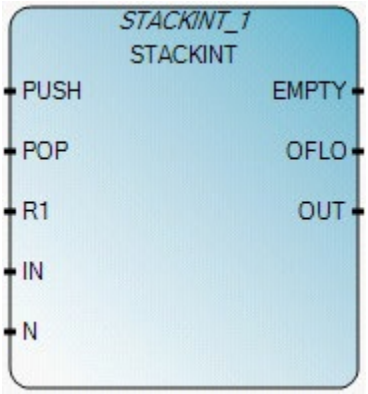
管理整数值的堆栈。

操作详细信息：

- STACKINT 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。
  - 如果 N < 1，STACKINT 假设大小为 1。
  - 如果 N > 128，STACKINT 假设大小为 128。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

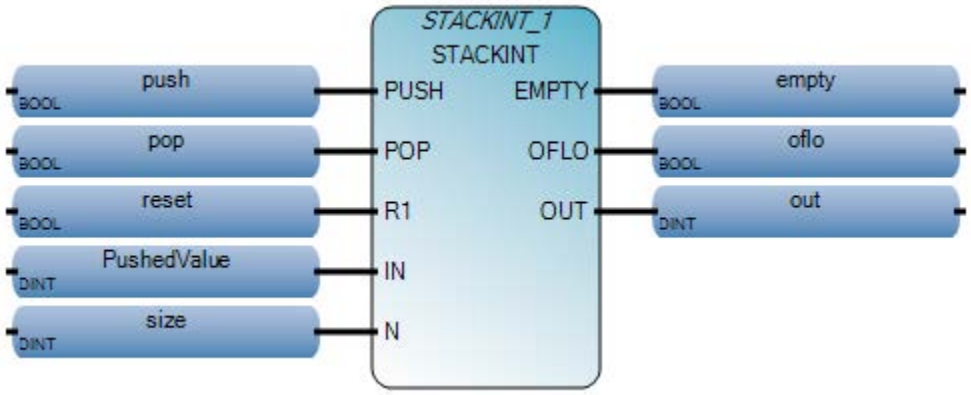
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



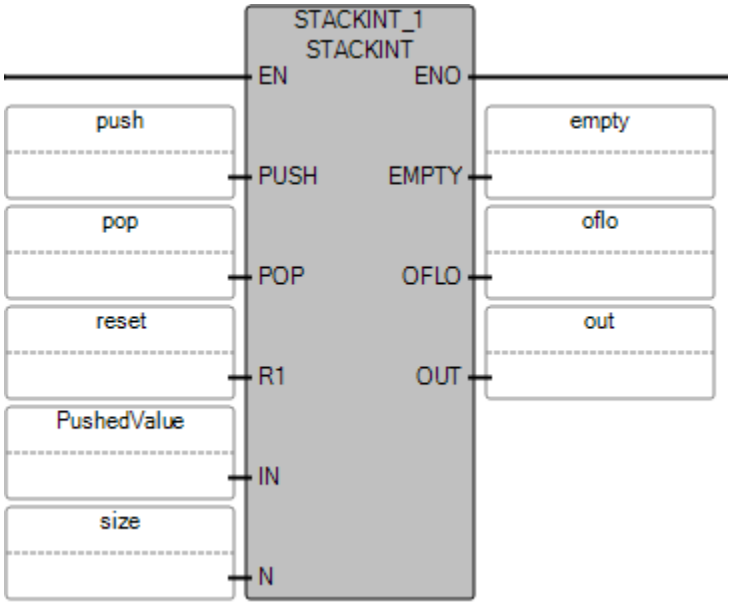
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
PUSH	输入	BOOL	TRUE - 在 PUSH 命令上检测到上升沿。将 IN 值添加在堆栈的顶部。 FALSE - 在 PUSH 命令上未检测到上升沿。
POP	输入	BOOL	TRUE - 在 PUSH 命令上检测到上升沿。删除上一个被推送到堆栈顶部的值。 FALSE - 在 POP 命令上未检测到上升沿。
R1	输入	BOOL	TRUE - 将堆栈复位至其空状态。 FALSE - 不复位。
IN	输入	DINT	推送的值。
N	输入	DINT	应用程序定义的堆栈大小。堆栈的最大大小为 128
EMPTY	输出	BOOL	TRUE - 堆栈为空。 FALSE - 堆栈包含值。
OFLO	输出	BOOL	TRUE - 上溢，堆栈已满，且至少有一次将 R1 设置为 TRUE，然后再设置回 FALSE。 FALSE - 堆栈大小为 128 或更小。无上溢。
OUT	输出	DINT	堆栈顶部的值。 当 OFLO 为 TRUE 时，OUT 等于 0。

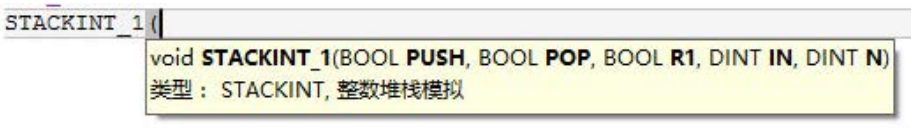
STACKINT 功能块图示例



STACKINT 梯形图示例



STACKINT 结构化文本示例




```
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;
```

(* 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

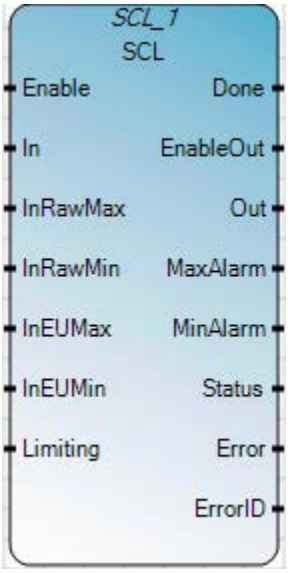


SCL

将未缩放的 REAL 输入值转换为采用工程单位的 REAL 浮点值，并包括输出的报警和限制。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

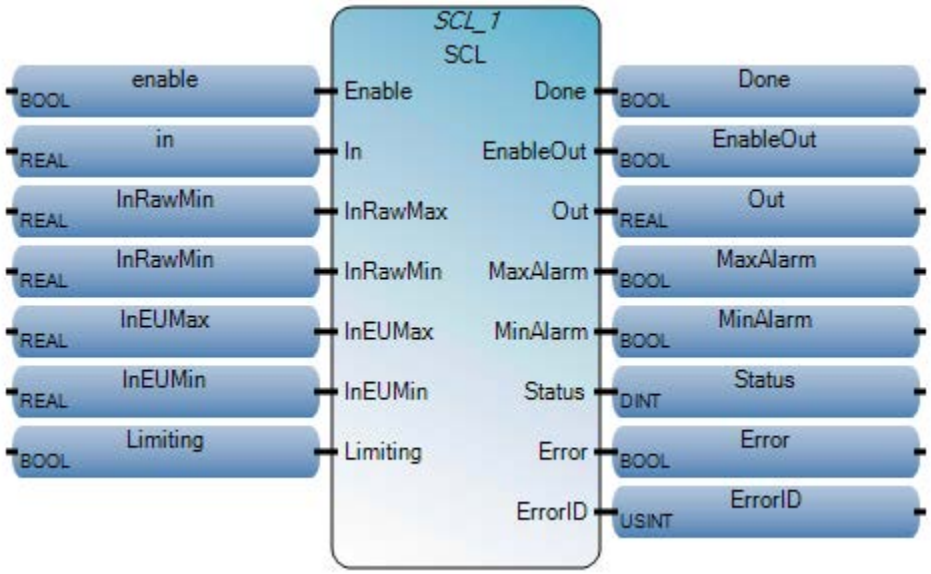
参数	参数类型	数据类型	描述
Enable	输入	BOOL	TRUE - 检测到上升沿。 • 如果 InRawMin >= InRawMax，Status.0 和 Status.1 位设置为 1。Done 被清除，MaxAlarm 和 MinAlarm 被清除。Error 设置为 TRUE 和 ErrorID 设置为 1。 • 首先计算 Else Out。然后验证报警条件。如果设置了 MaxAlarm，则清除 MinAlarm，反之亦然。然后，如果设置了限制，则 Out 将在 InEUMin 到 InEUMax 范围内。然后，Done 位设置为 TRUE。状态位设置为 0。Error 和 ErrorID 设置为 0。 • 计算出的 Out 值与 NAN（不是数字）进行比较。如果 NAN 是 Out 值 ，则清除 EnableOut。如果已设置故障条件，则清除 Done。 FALSE - 未检测到上升沿。 • 指令不执行。 • 除非 Error、ErrorID、EnableOut 和 Done 设置为 0，否则输出不执行。
In	输入	REAL	模拟量信号输入。 有效 = 任何浮点 默认值 = 0.0
InRawMax	输入	REAL	指令输入可达到的最大值。如果 InRawMax<= InRawMin，则指令会将 Status 和 Error 中的相应位置位。ErrorID 被更新。Out 更新停止。 有效 = InRawMax > InRawMin 默认值 =0.0
InRawMin	输入	REAL	指令输入可达到的最小值。如果 InRawMin >= InRawMax，指令会将 Status 和 Error 中的相应位置位。ErrorID 被更新。Out 更新停止。 有效 = InRawMin < InRawMax 默认值 = 0.0
InEUMax	输入	REAL	对应于 InRawMax 的输入的经缩放值。 有效 = 任何实型值 默认值 = 0.0
InEUMin	输入	REAL	对应于 InRawMin 的输入的经缩放值。 有效 = 任何实型值 默认值 = 0.0
Limiting	输入	BOOL	限制选择器。 TRUE - Out 被限制在 InEUMin 与 InEUMax 之间。
Done	输出	BOOL	指示操作完成的时间。 TRUE - 操作已成功完成。 FALSE - 操作遇到错误条件或者 Enable 被设置为 FALSE。
EnableOut	输出	BOOL	指示指令是否已启用。如果 Out 上溢，则设置为 False。
Out	输出	REAL	代表模拟量输入的缩放值。
MaxAlarm	输出	BOOL	最大输入报警指示器。 当 Input > InRawMax 时，该值设置为 TRUE。
MinAlarm	输出	BOOL	最小输入报警指示器。 当 Input < InRawMin 时，该值设置为 TRUE。
Status	输出	DINT	功能块的状态。 InstructFault (Status.0) 在检测到指令的执行错误时被置位。这不是次要或主要控制器错误。检查剩余状态位以确定发生了什么。 InRawRangeInv (Status.1) InRawMin >= InRawMax Status.3 到 Status.31 保留供将来使用，它们的值为 0。

参数	参数类型	数据类型	描述
Error	输出	BOOL	表示存在错误条件。 TRUE - 操作过程遇到错误。 FALSE - 操作过程成功完成或指令未执行。
ErrorID	输出	BOOL	标识错误的唯一数字。这些错误在错误代码中定义。

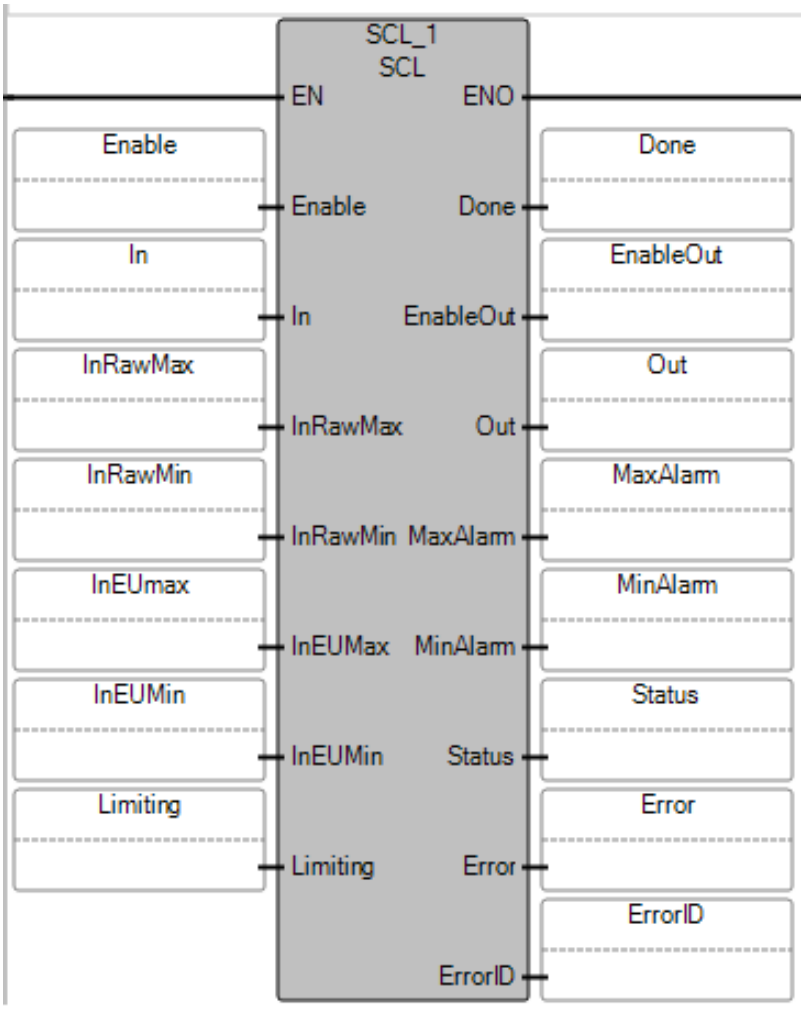
错误代码

ErrorID 代码	错误描述
1	InRawMax<= InRawMin

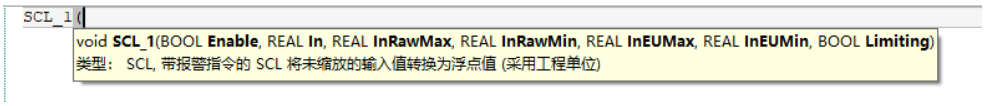
SCL 功能块图示例



SCL 梯形图示例



SCL 结构化文本示例

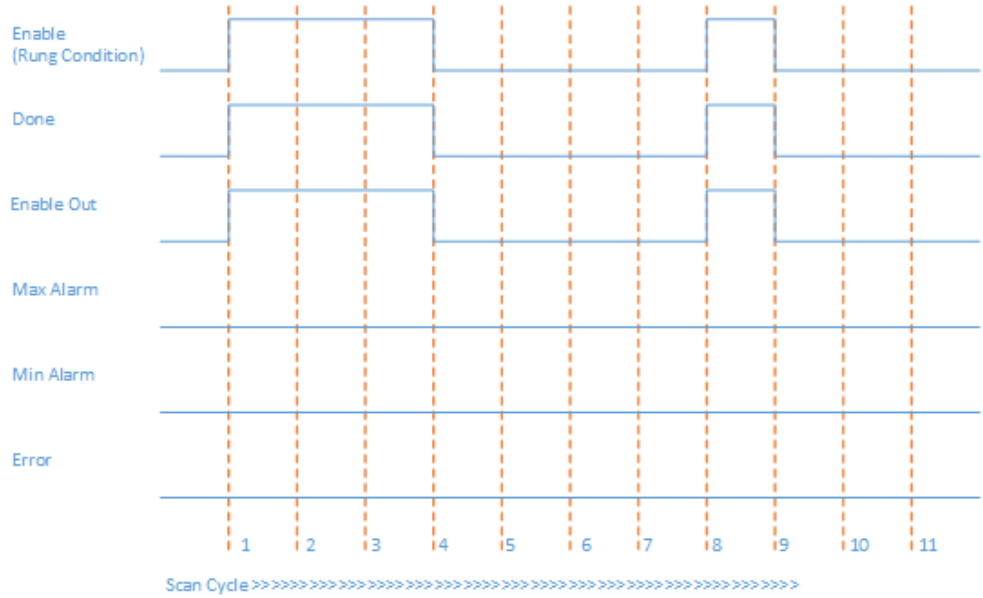



```
SCL_1(Enable, In, InRawMax, InRawMin, InEUMax, InEUMin, Limiting);
Done :=SCL_1.Done;
EnableOut :=SCL_1.EnableOut;
Out :=SCL_1.Out;
MaxAlarm :=SCL_1.MaxAlarm;
MinAlarm :=SCL_1.MinAlarm;
Status :=SCL_1.Status;
Error :=SCL_1.Error;
ErrorID :=SCL_1.ErrorID;
```

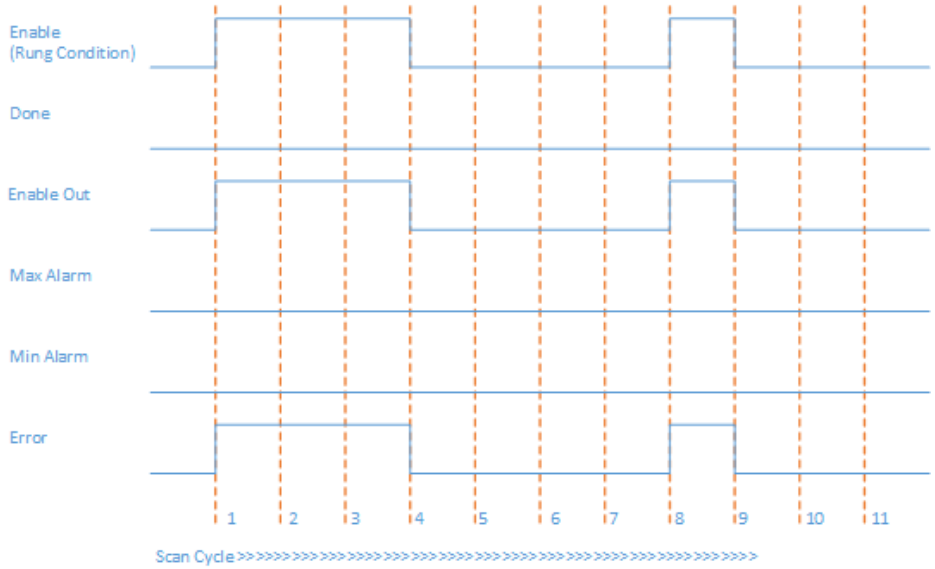
SCL 指令时间图示例

以下时序图示例描述了 SCL 参考页数 481 指令的执行场景。

SCL 执行成功

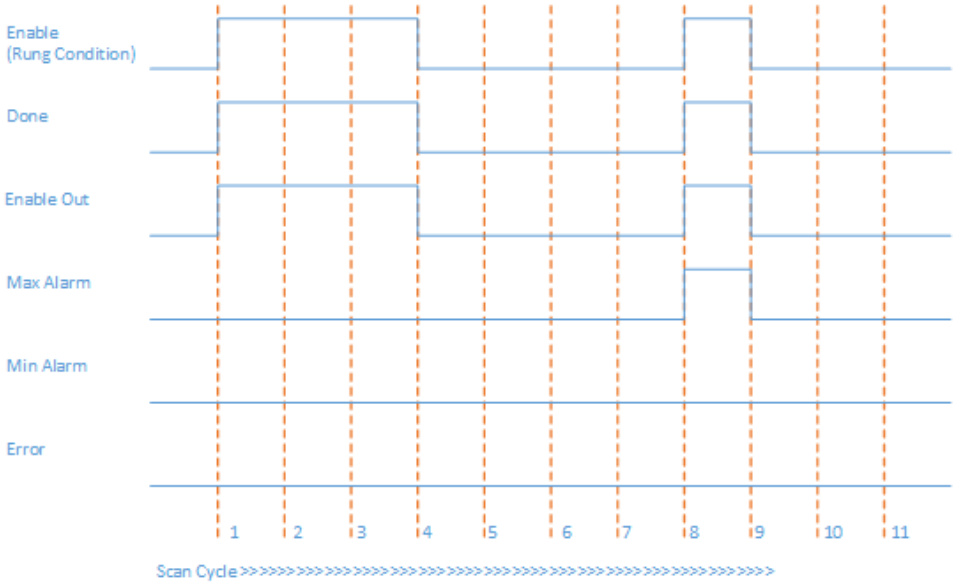

Enable
(Rung Condition)
Done
Enable Out
Max Alarm
Min Alarm
Error
1
2
3
4
5
6
7
8
9
10
11
Scan Cycle >>>>

SCL 执行失败



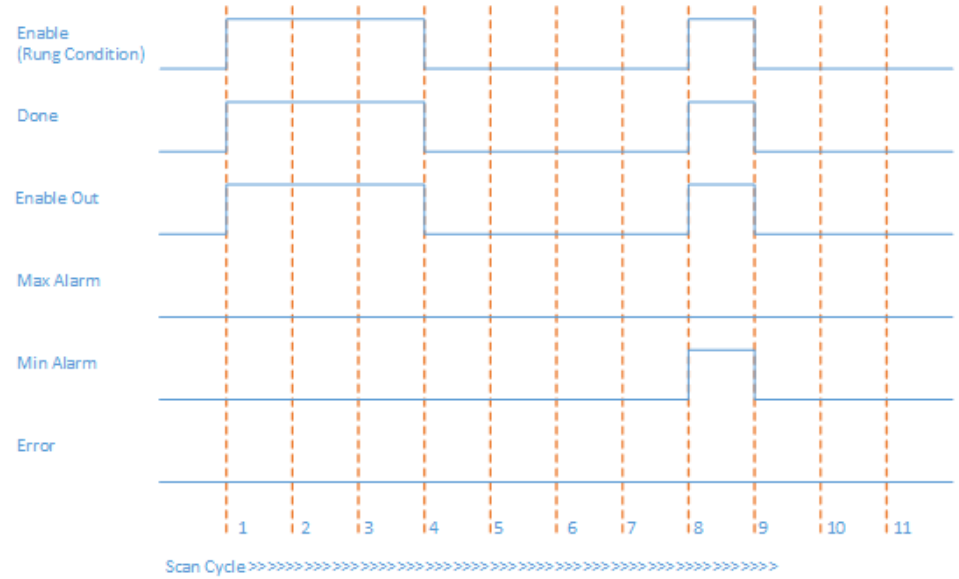
在该示例中，所有输入参数均有效且在范围内，但 InRawMin >= InRawMax。在扫描周期 1 和 8 中，当 Enable 设置为 TRUE 且功能块执行开始时 ，Error 设置为 True 且 ErrorID 设置为 1。

生成 MaxAlarm



在该示例的扫描周期 8 中，所有输入参数均有效且在范围内，但 In > InRawMax。当 Enable 设置为 TRUE 且功能块执行开始时，MaxAlarm 设置为 TRUE。

生成 MinAlarm


Enable
(Rung Condition)
Done
Enable Out
Max Alarm
Min Alarm
Error
1
2
3
4
5
6
7
8
9
10
11
Scan Cycle >>>>

TND（停止当前程序）

停止用户程序扫描的当前循环。在输出扫描、输入扫描和清理完成后，将从第一个例程的开始处重新执行用户程序。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

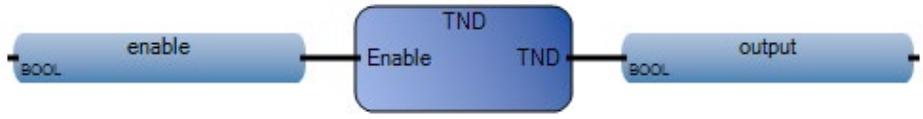
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



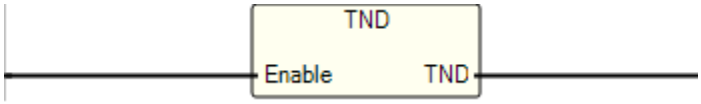
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	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。 如果启用变量监视，则将监视变量的值分配给指令的输出。 如果关闭变量监视，则将输出变量的值分配给指令的输出。

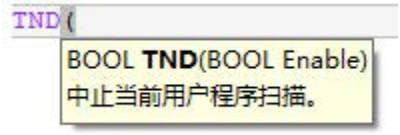
TND 功能块图示例



TND 梯形图示例



TND 结构化文本示例




```
1  enable := TRUE;
2  output := TND(enable);
```

(* 与之等效的 ST：*)

TESTOUTPUT := TND(TESTENABLE) ;

结果



LIMIT（限制测试）

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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



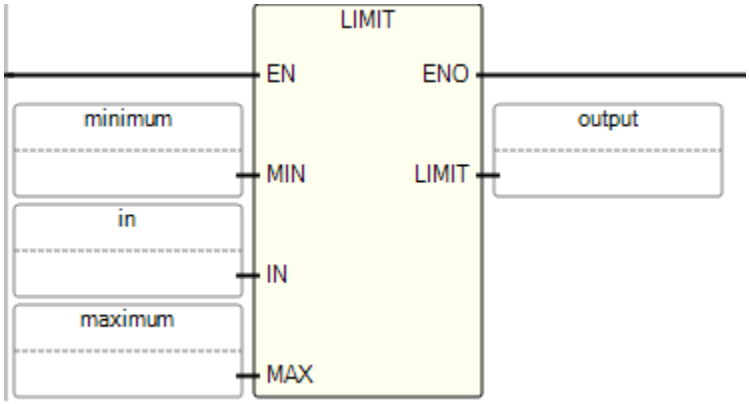
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	函数启用。 TRUE - 执行当前 LIMIT 计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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”。 适用于梯形图编程。

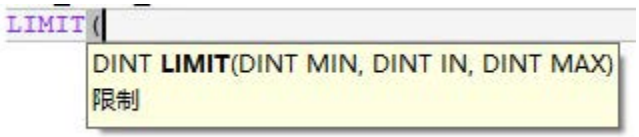
LIMIT 功能块图示例



LIMIT 梯形图示例



LIMIT 结构化文本示例




```
1  minimum := 2;
2  in := 5;
3  maximum := 10;
4  output := LIMIT(minimum, in, maximum);
```

(* 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




变量监视
用户全局变量 - Micro870
局部变量 - RA_LIMIT_LD
系统变量- Micro870
I/O - Micro870
已定义的字
名称
别名
逻辑值
实际值
初始值
锁定
数据类型
in
1
不可用
DINT
maximum
10
不可用
DINT
minimum
2
不可用
DINT
output
2
不可用
DINT
关闭(C)


变量监视
用户全局变量 - Micro870
局部变量 - RA_LIMIT_LD
系统变量- Micro870
I/O - Micro870
已定义的字
名称
别名
逻辑值
实际值
初始值
锁定
数据类型
in
11
不可用
DINT
maximum
10
不可用
DINT
minimum
2
不可用
DINT
output
10
不可用
DINT
关闭(C)

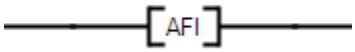
程序控制指令

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。

指令	描述
AFI 参考页数 493	禁用梯级。
NOP 参考页数 493	起占位符作用。
SUS 参考页数 493	挂起 <M800 控制器> 的执行。

AFI（始终为 False）

在梯级的开始处使用 AFI 指令在调试期间临时禁用梯级，而不必从程序中删除梯级。该指令的输出始终为 FALSE。

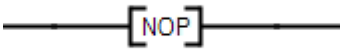


支持的语言：梯形图。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

NOP（无操作）

NOP 指令起占位符作用。您可以将 NOP 指令放置在梯级上的任何位置。



支持的语言：梯形图。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

SUS（挂起）

挂起 <M800 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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

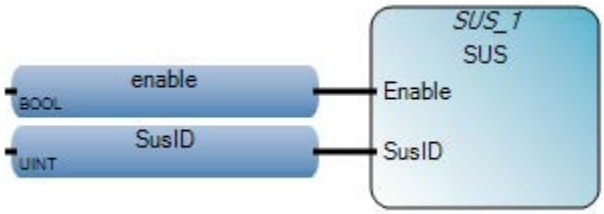
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



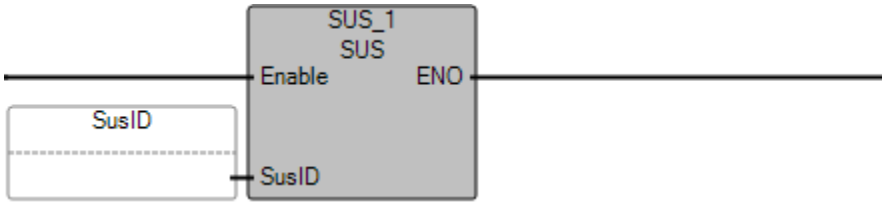
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行功能。 FALSE - 不执行功能。
SusID	输入	UINT	挂起 ID。
ENO	输出	BOOL	启用输出。 仅适用于梯形图编程。

SUS 功能块图示例



SUS 梯形图示例



SUS 结构化文本示例




```
1  SusID := 1;
2  SUS_1(enable, SusID);
```

结果


变量监视
用户全局变量 - Micro850
局部变量 - Prog2
系统变量- Micro850
I/O - Micro850
已定义的字
名称
逻辑值
实际值
初始值
锁定
数据类型
维度
SUS_1
SUS
SUSID
1
不可用
UINT
enable
不可用
BOOL
关闭(C)

比例积分微分 (PID) 指令

比例积分微分 (PID) 指令用来使用 PID 功能更准确地控制过程。

指令	描述
IPIDCONTROLLER 参考页数 497	配置和控制用于比例积分微分 (PID) 逻辑的输入和输出。
PID 参考页数 518	使用过程循环，配置和控制用于控制诸如温度、压力、液面或流速等物理属性的输出。

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。

IPIDCONTROLLER（比例-积分-微分控制器）

配置和控制用于比例积分微分 (PID) 逻辑的输入和输出。PID 逻辑通过使用为所需设定点和测量过程变量之间的差异计算错误值的过程循环，用来控制诸如温度、压力、液面、级别或流速等物理属性。控制器尝试通过调整控制变量最小化随时间出现的错误。计算包括使用的比例 (P)、积分 (I) 和微分 (D) 项，如下所示：

- P - 错误的当前值。

- I - 错误的上一个值。

- D - 错误的可能未来值，基于其当前更改速率。使用过程循环控制诸如温度、压力、液面或流速等物理属性。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

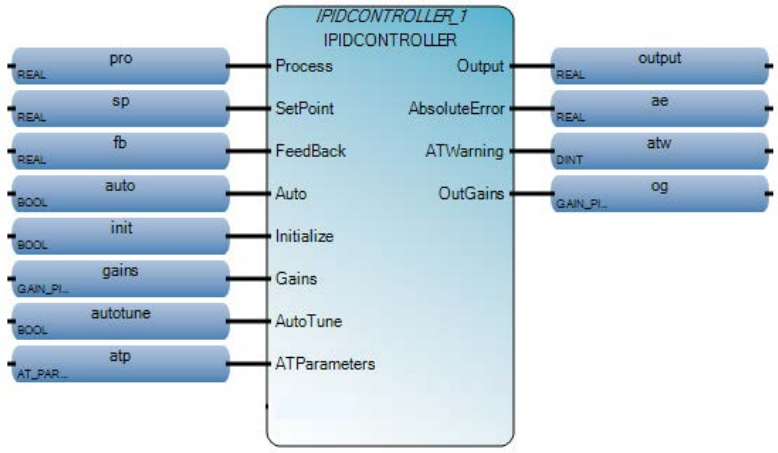
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



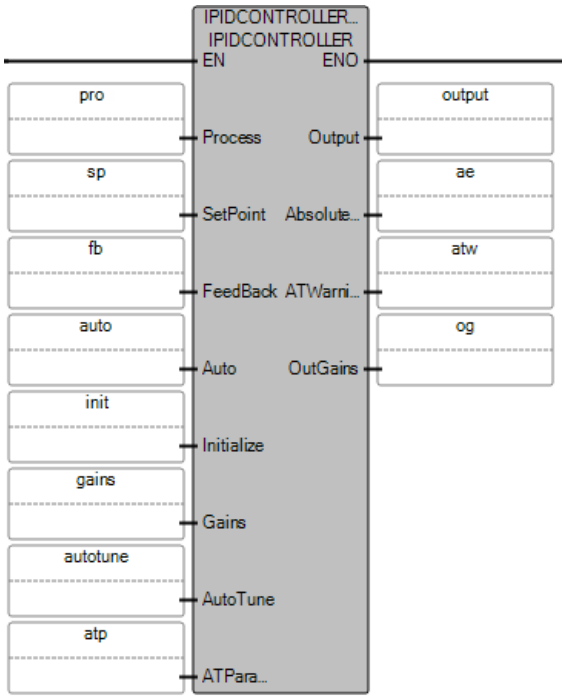
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	当为 TRUE 时，启用指令块。 TRUE - 执行 PID 计算。 FALSE - 指令块处于空闲状态。 适用于梯形图编程。
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。
FeedBack	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。
Auto	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) 导致在该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。
Gains	输入	GAIN_PID	IPIDController 的增益 PID。 使用 GAIN_PID 数据类型 参考页数 500 定义增益输入的参数。
AutoTune	输入	BOOL	TRUE - 当 AutoTune 为 TRUE 且 Auto 和 Initialize 为 FALSE 时，会启动 AutoTune 序列。 FALSE - 不启动 Autotune。
ATParameters	输入	AT_Param	自动调节参数。 使用 AT_Param 数据类型 参考页数 501 定义 ATParameters 输入的参数。
Output	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。 使用 GAIN_PID 数据类型定义 OutGains 输出。
ENO	输出	BOOL	启用“输出”。 适用于梯形图编程。

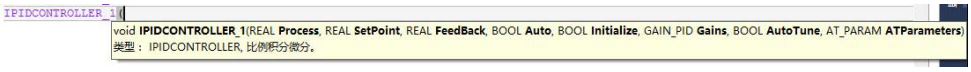
IPIDCONTROLLER 功能块图示例



IPIDCONTROLLER 梯形图示例



IPIDCONTROLLER 结构化文本示例




```
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;
```

(* 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



GAIN_PID 数据类型

下表描述了 IPIDCONTROLLER 参考页数 497 指令的 GAIN_PID 数据类型。

参数	数据类型	描述
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)。 当 ProportionalGain 为 (< 0.0001) 时，ProportionalGain = 0.0001 **PID 的比例增益 (P_Gain)** 更高的比例增益可能会导致基于 PV（测量的过程值）和 SV（设置点值）之间差值的输出发生较大的变化。增益越高，减少发生错误的速度越快，但这可能会导致不稳定，如振荡。增益越低，减少发生错误的速度越慢，但系统更稳定且发生重大错误的可能性更低。通常，P_Gain 是要调整的最重要增益，同时也是在运行时要调整的第一个增益。
TimeIntegral	REAL	PID 的时间积分值（以秒为单位）(>= 0.0001)。 当 TimeIntegral 为 (< 0.0001) 时，TimeIntegral = 0.0001 **PID 的时间积分值**

参数	数据类型	描述
		较小的积分时间常量会导致基于在此时间内积分的 PV（测量的过程值）和 SV（设置点值）之间差值的输出更快发生变化。较小的积分时间常量会减少稳态错误（当 SV 不发生变化时的错误），但同时也会增加出现不稳定的机会，如振荡。较大的积分时间常量会降低系统响应的速度并使系统变得更稳定，但 PV 会在较慢的速度下接近 SV。
TimeDerivative	REAL	PID 的时间微分值（以秒为单位）(> 0.0)。 当 TimeDerivative 为 (< = 0.0) 时，TimeDerivative = 0.0 当 TimeDerivative = 0 时，IPID 用作 PI。 **PID 的时间微分值 (Td)** 较小的微分时间常量会导致基于 PV（测量的过程值）和 SV（设置点值）之间差值的更改速率的输出更快发生变化。较小的微分时间常量会使系统对错误（SV 发生变化）的突然变化作出更快的响应，但同时也会增加系统的不稳定性，如振荡。较大的微分时间常量会降低系统对错误的突然变化作出响应的速度，同时也会使系统对 PV 中的噪声和步阶变化变得不敏感。 TimeDerivative (Td) 与微分增益有关联，但允许使用时间调整导致 PID 的微分，因此必须考虑采样时间。
DerivativeGain	REAL	"PID 的微分增益"(> 0.0)。 当 DerivativeGain 为 (< 0.0) 时，DerivativeGain = 0.1 **PID 的微分增益 (D_Gain)** 较高的微分增益会导致 PV（测量的过程值）和 SV（设置点值）之间差值的变化速率的输出更快发生变化。较高的增益会使系统对错误的突然变化作出更快的响应，但同时也会增加不稳定的机会，如振荡。较低的增益会降低系统对错误的突然变化作出响应的速度，同时也使系统对 PV 中的噪声和步阶变化变得不敏感。

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	描述
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。
Step	REAL	AutoTune 的步长值。必须大于噪声频带并小于 ½ 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。

IPIDController 功能块如何实施 PID 控制

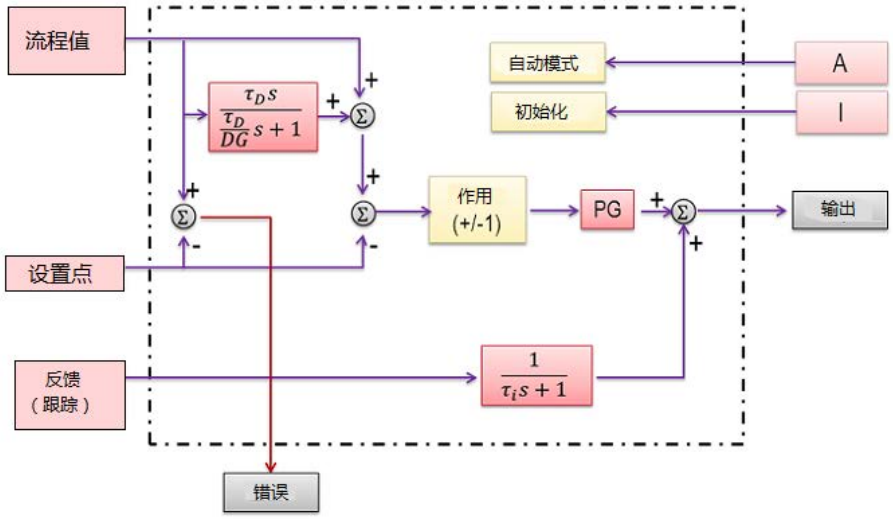
Connected Components Workbench 指令集中提供的 IPIDController 功能块基于 PID 控制理论，它将所有必需的逻辑合并以执行模拟输入通道处理以及比例积分微分 (PID) 控制。在 HMI 中，IPID 面板可用于 IPIDController 功能块。

IPIDController 功能块说明

IPIDController 功能块使用以下功能块组件：

- 答：作用 (+/- 1)
- PG：比例增益

- DG：导数滤波器增益
- td：ãD
- ti：ãI



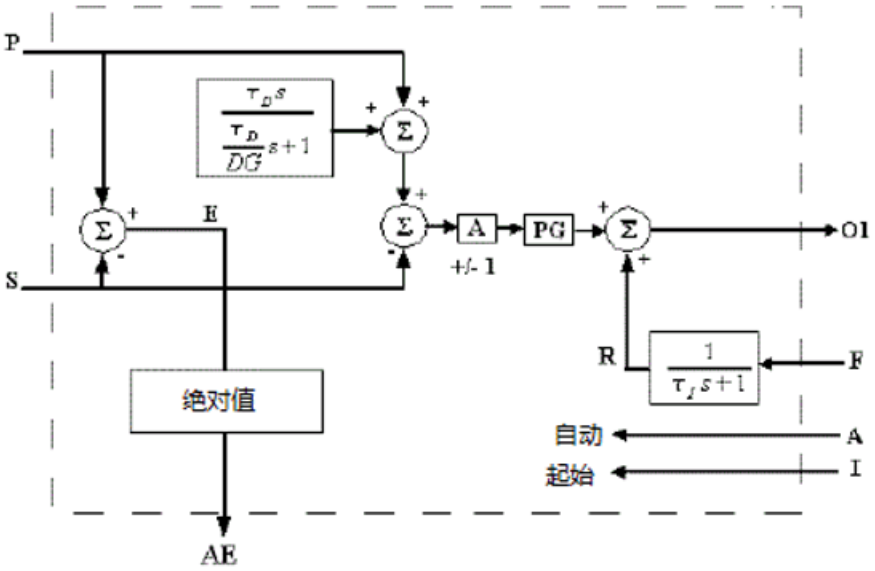
防止积分饱卷

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，过程会失控。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。

IPIDController 功能块操作

当“输入自动”为“TRUE”时，IPIDController 以正常的自动模式运行。

当“输入自动”为“FALSE”时，它将使得 R 被复位为跟踪 (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。



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，使用此进程来防止增加输出。

要运行 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。

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。

使用比例积分微分指令

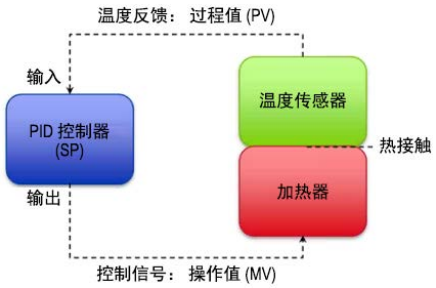
本节介绍使用比例积分微分指令的特定详细信息和示例，具体包括以下内容：

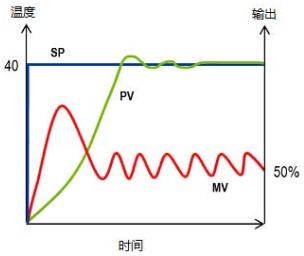
示例：如何创建操作值的反馈回路

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

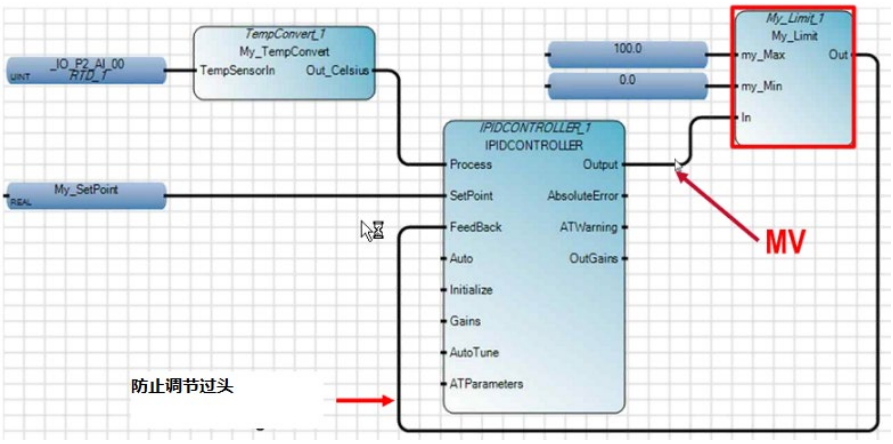
在温度控制过程开始时，过程值 (PV) 和设定点值 (SP) 之间的差异较大，如下图所示。在此温度反馈回路示例中，PV 从 0 摄氏度开始，向 40 摄氏度的 SP 值移动。另请注意，高操作值和低操作值 (MV) 之间的波动会随时间推移而减少并逐渐稳定。MV 的行为取决于各个 P、I 和 D 参数中使用的值。





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。



示例：如何在 IPIDController 功能块中实施自动调节

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。

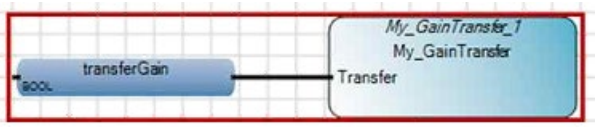
- 自动调节必须使控制回路的输出发生振荡，这意味着必须足够频繁地调用 IPIDController，以对振荡进行充分采样。
- IPIDController 指令块必须以相对恒定的时间间隔执行。
- 将程序的扫描时间配置为小于振荡周期的一半。
- 可考虑使用结构化文本中断 (STI) 指令块来控制 IPIDController 指令块。

示例：如何将 UDFB 添加到 PID 程序

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。

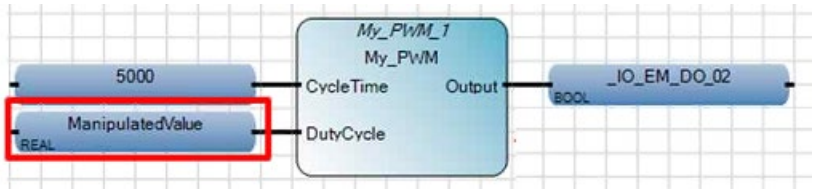
传输自动调节增益值

该 UDFB 将 Autotune 增益值传输到 My_GainTransfer，从而使控制器可以使用该值。



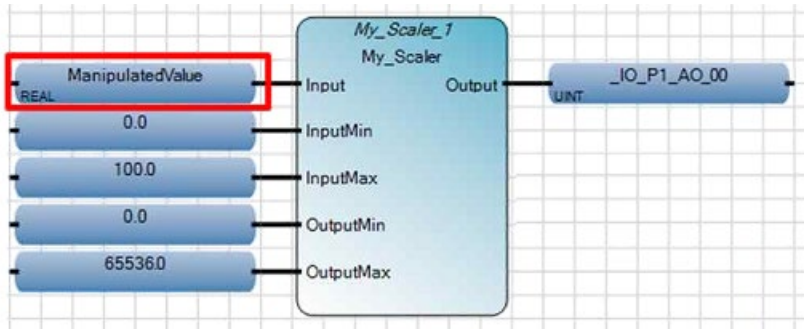
将操作值转换为数字输出

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)。



通过 IPIDController 功能块实现自动调节

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。
- 考虑使用可选定时中断 (STI) 指令块来控制 IPIDController 指令块。

一阶和二阶系统中的自动调节

在使用单个元素的一阶系统或在使用两个独立元素的二阶系统中使用自动调节。

一阶系统使用单个独立的储能元素。以下是一些示例：

- 储液箱冷却，将热能作为储存单位。
- 自储液箱的液体流动，将势能作为储存单位。
- 以恒定力矩驱动盘形飞轮的电机，将转动动能作为储存单位。
- 电力 RC 超前网络，将电容能量作为储存单位。

在一阶系统中，函数可以按标准格式编写，例如 $f(t) = \tau\, dy/dt + y(t)$

其中：

变量	描述	示例：将热能作为储存元素的储液箱冷却
t	系统时间常量	ls 等于 RC 其中 R = 箱壁的热阻 C = 液体的热容
f	强制函数	即环境温度
y	系统状态变量	即液体温度

二阶系统使用两个可以交换储能的独立储能元素。以下是一些示例：

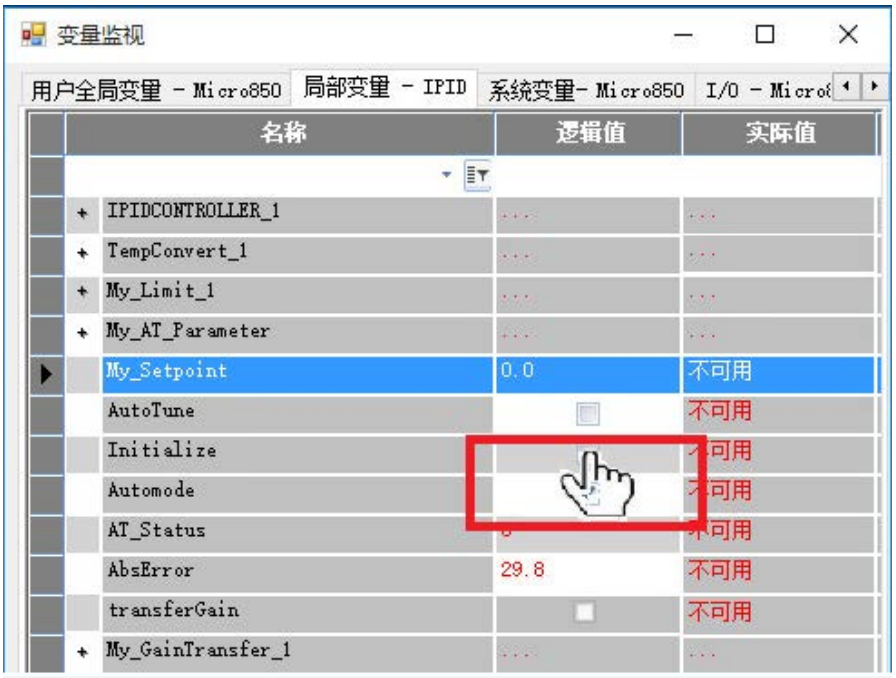
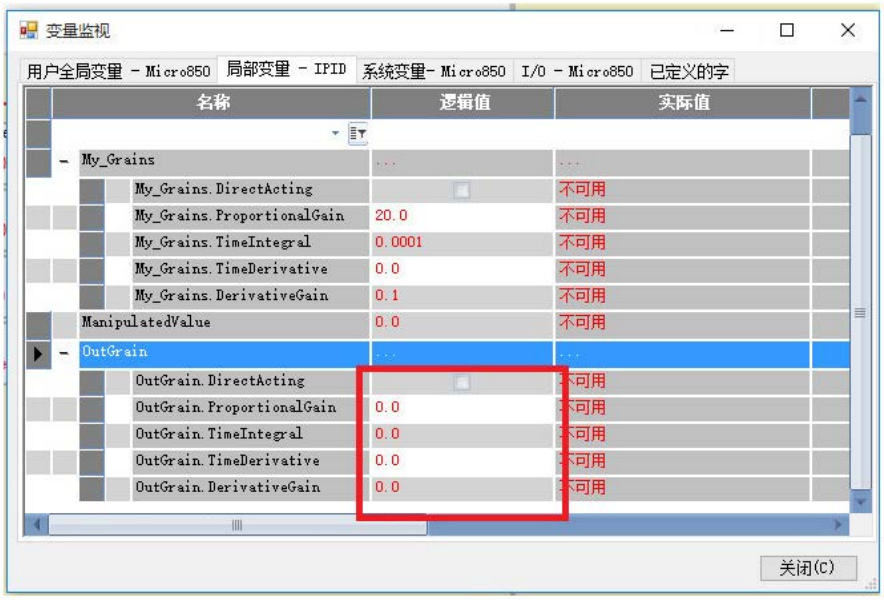
- 用于驱动盘式飞轮的电机（电机通过具有扭转刚度的轴与飞轮联接）；将转动动能和扭力弹簧能量作为储存单位。
- 电路，其中含有用于驱动串联 LR（电感和电阻）和分流 C（电容）的电流源；将电感能量和电容能量作为储存单位。

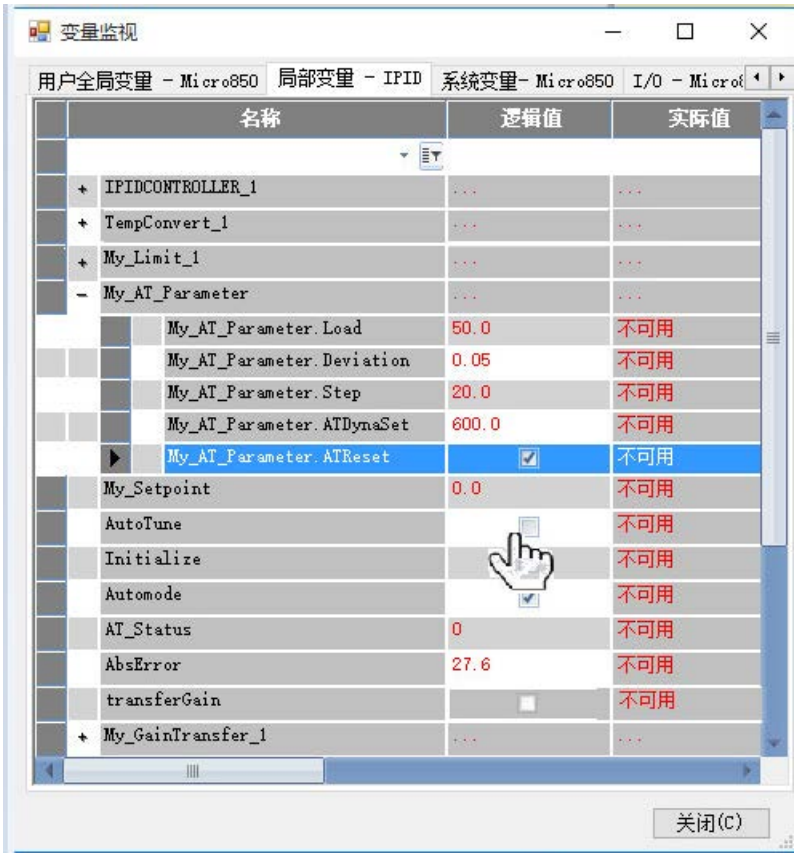
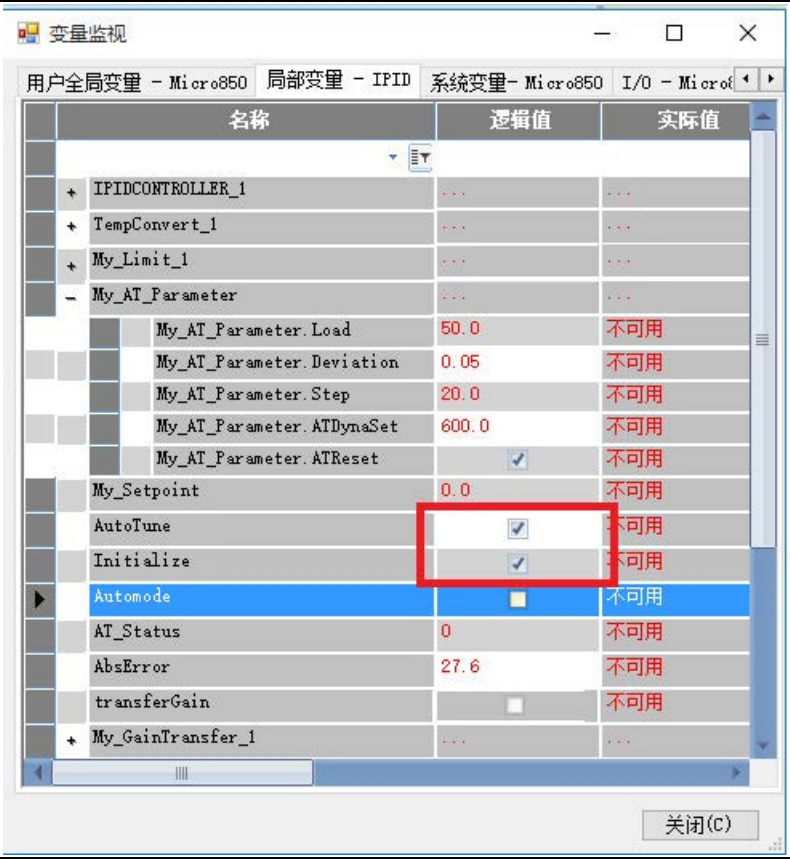
电机驱动的系统和加热系统通常能够通过 LR 和 C 电路建模。

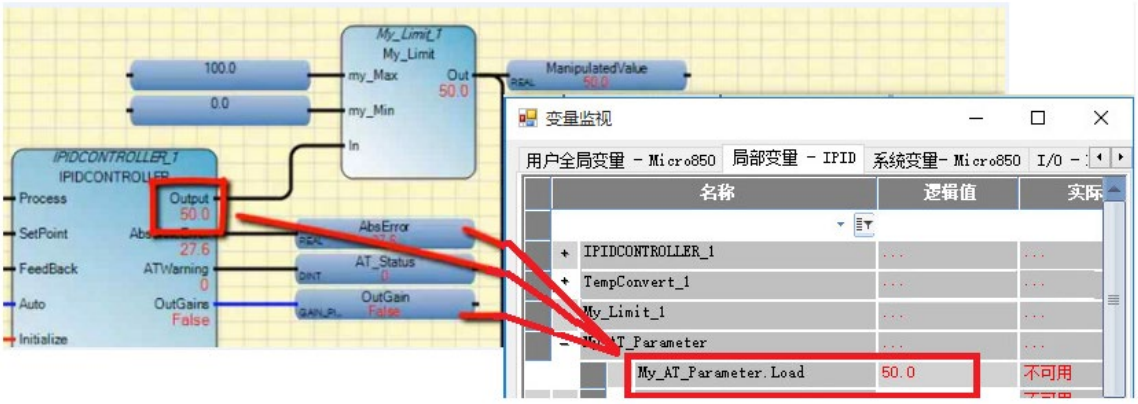
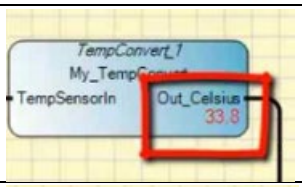
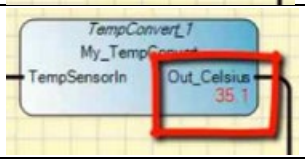
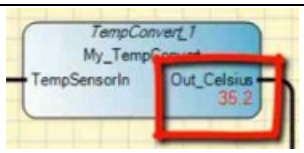
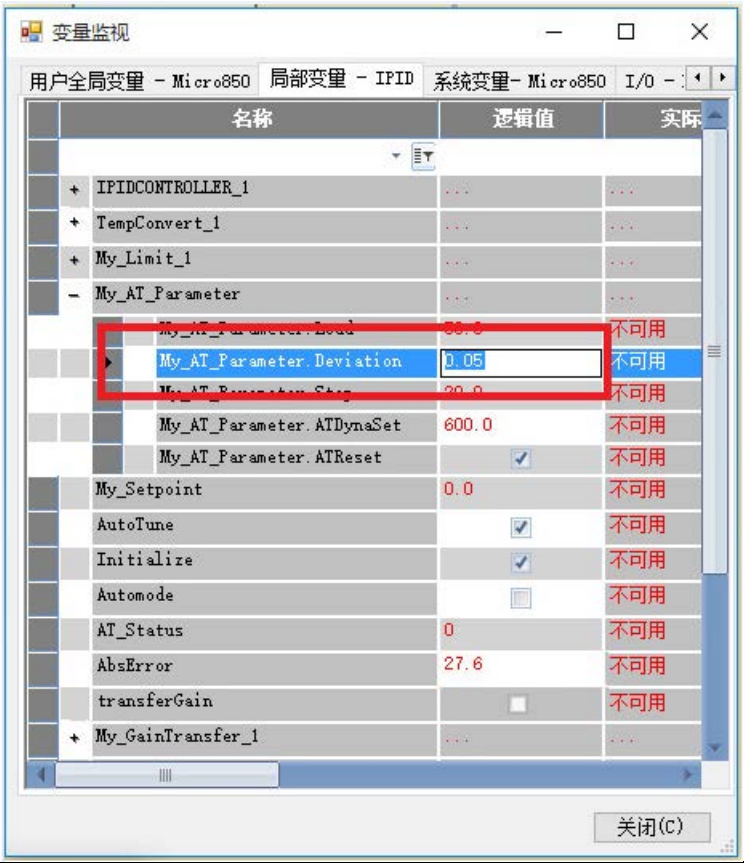
配置自动调节

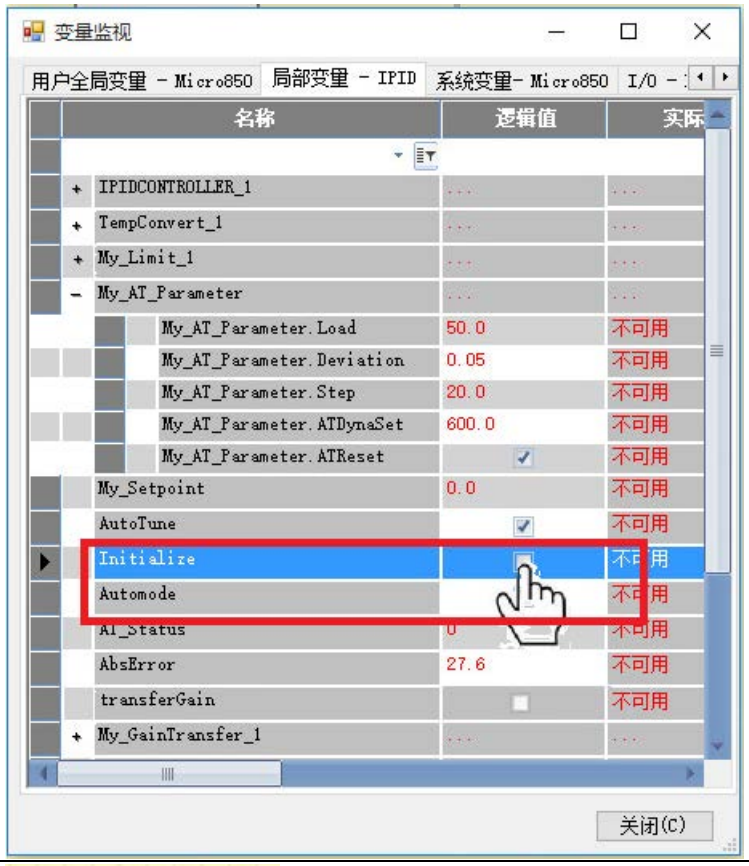
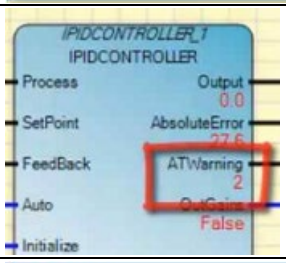
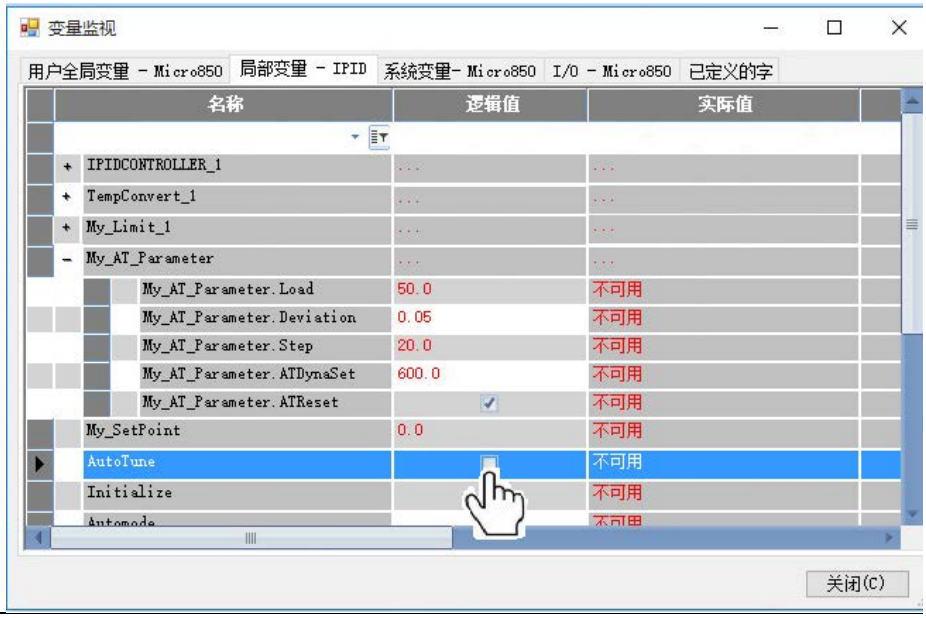
使用 IPIDController 函数实现自动调节 参考页数 505时，使用以下一般步骤。

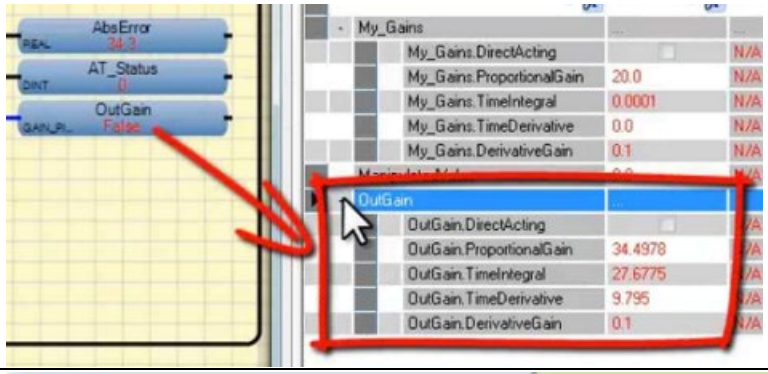
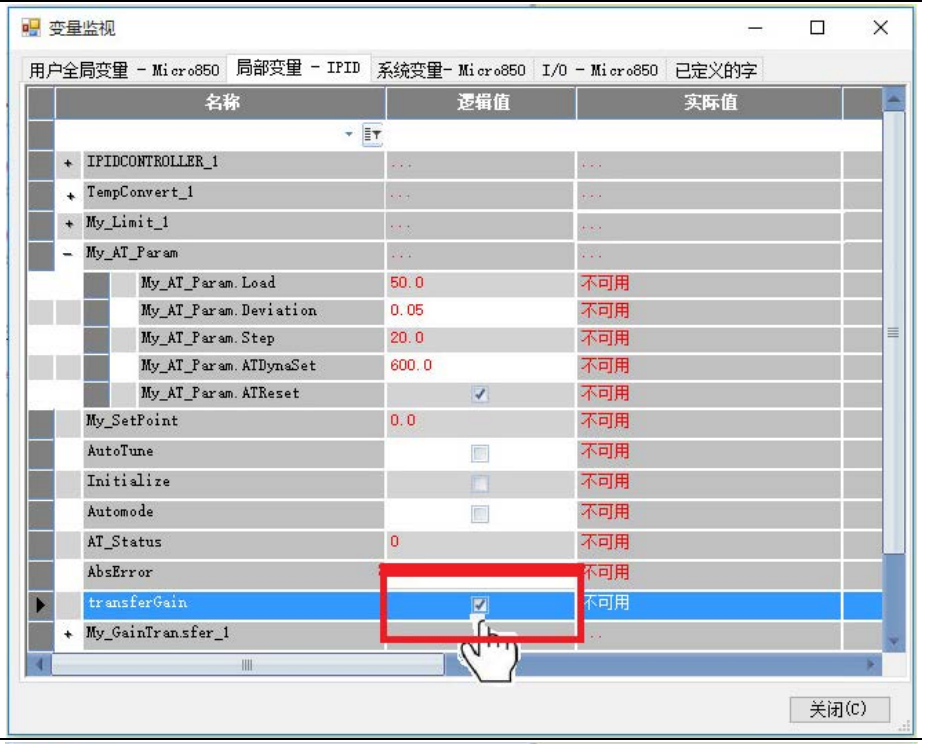
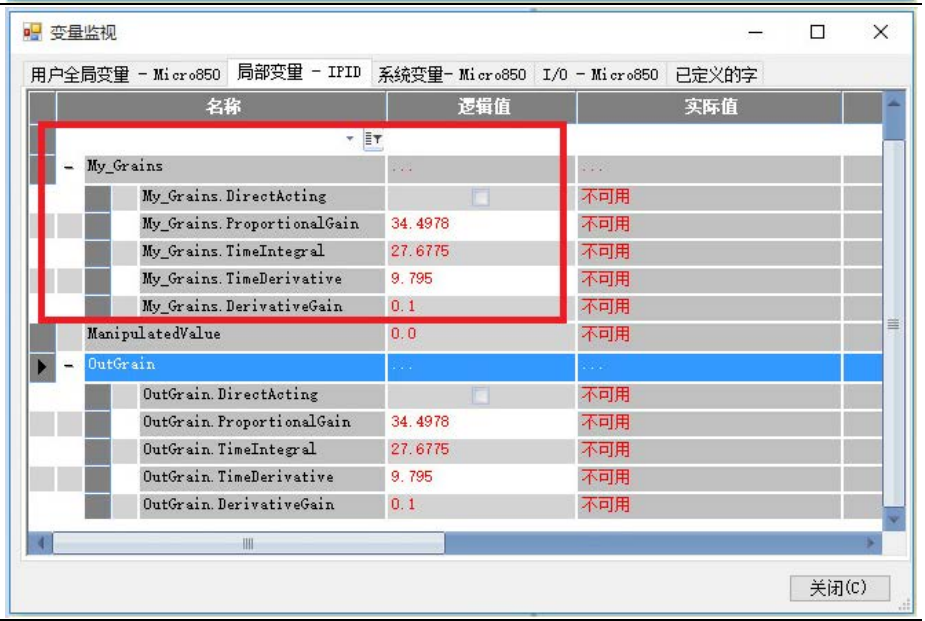
编号	步骤	示例
1	将设定点重置为零。	

编号	步骤	示例
2	将 Auto 模式切换为 False	
3	设置增益参数。	

编号	步骤	示例
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。
5	将 Initialize 和 AutoTune 设置为 TRUE。	

编号	步骤	示例
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。	 
9	设置偏移量。	

编号	步骤	示例
10	将 Initialize 设为 False。	
11	控制器开始自动调节。等待 ATWarning 变为 2。	
12	将 AutoTune 设为 False。	

编号	步骤	示例
13	观察调节值出现在 OutGains 中。	
14	将参数从 OutGain 传输到 My_Gains。	
15	观察使用调节的增益参数更新控制器。	

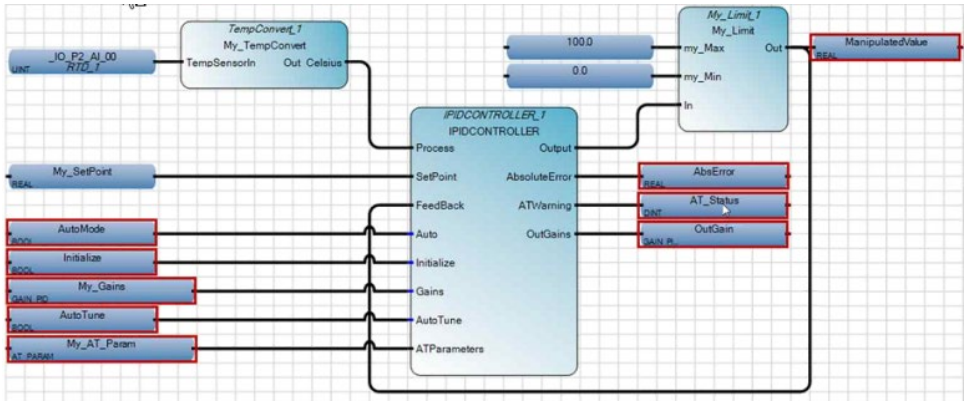
通过自动调节使用可选定时中断 (STI)

虽然 PID 指令在没有可选定时中断 (STI) 控制下也有效，但是使用 STI 可提高自动调节成功率，因为自动调节按固定周期运作。



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**	**描述**
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。
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。

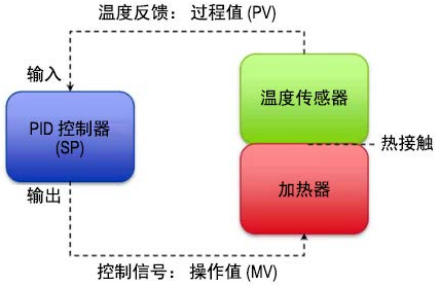
输入参数		
My_Gains.ProportionalGain	ProportionalGain	"PID 的比例增益"(>= 0.0001)。
My_Gains.TimeIntegral	TimeIntegral	"PID 的时间积分值"(>= 0.0001)。 振荡趋势会随着 ti 降低而提高。
My_Gains.TimeDerivative	TimeDerivative	"PID 的时间微分值"(> 0.0)。 随着微分时间增加，阻尼也会增加，但是如果微分时间太大，则会减小。
My_Gains.DerivativeGain	"PID 的微分增益"(> 0.0)。	
AutoTune	当设置为 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	**描述**	
AbsoluteError	来自控制器的绝对错误 (Process - SetPoint)。	
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。	

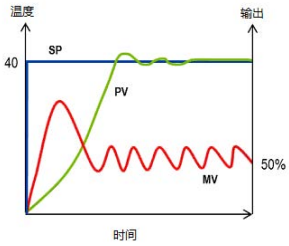
示例：如何创建操作值的反馈回路

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

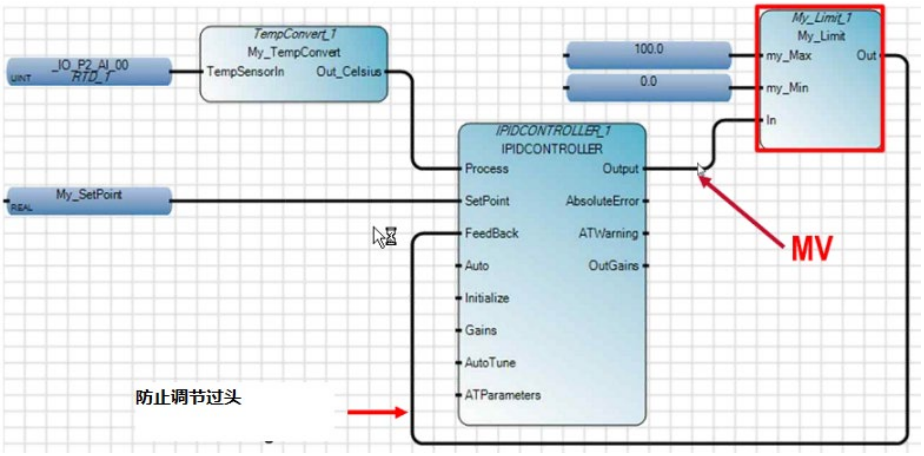
在温度控制过程开始时，过程值 (PV) 和设定点值 (SP) 之间的差异较大，如下图所示。在此温度反馈回路示例中，PV 从 0 摄氏度开始，向 40 摄氏度的 SP 值移动。另请注意，高操作值和低操作值 (MV) 之间的波动会随时间推移而减少并逐渐稳定。MV 的行为取决于各个 P、I 和 D 参数中使用的值。





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。



示例：如何将 UDFB 添加到 PID 程序

对于 PID 程序，在主程序外部使用用户定义的功能块 (UDFB) 以执行特殊功能，如转换单位或传输值。以下为 UDFB 示例：

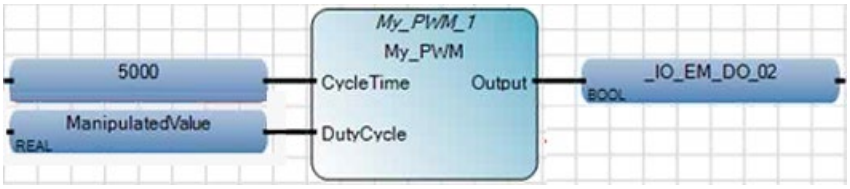
传输自动调节增益值

该 UDFB 将 Autotune 增益值传输到 My_GainTransfer，从而使控制器可以使用该值。



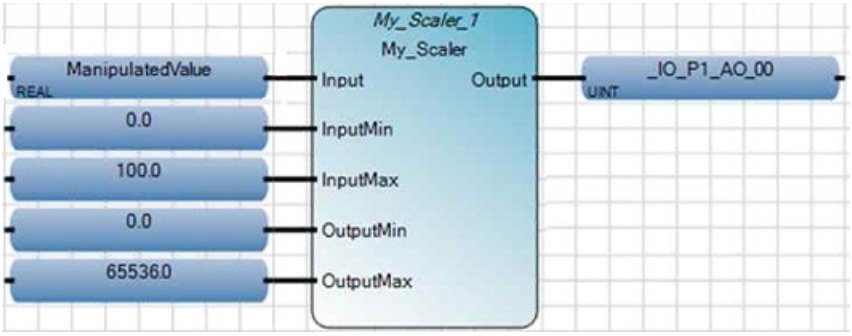
将操作值转换为数字输出

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)。



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。

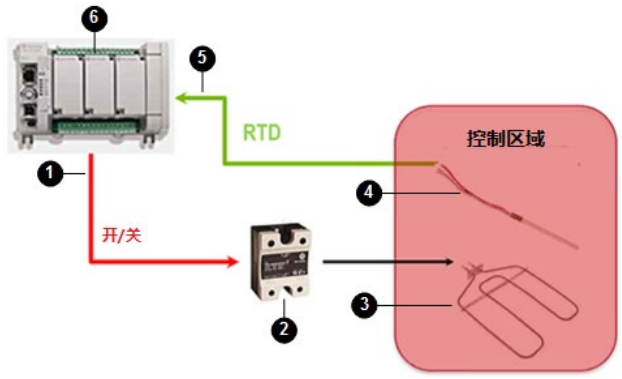
设定点、过程值和操作值

下表定义了在温度控制程序中使用的 SP、PV 和 MV 值。

项目	描述
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

下图和下表定义了在温度控制系统中由温度控制程序控制的组件，以及在控制程序运行时发生的事件。



温度控制程序的事件序列

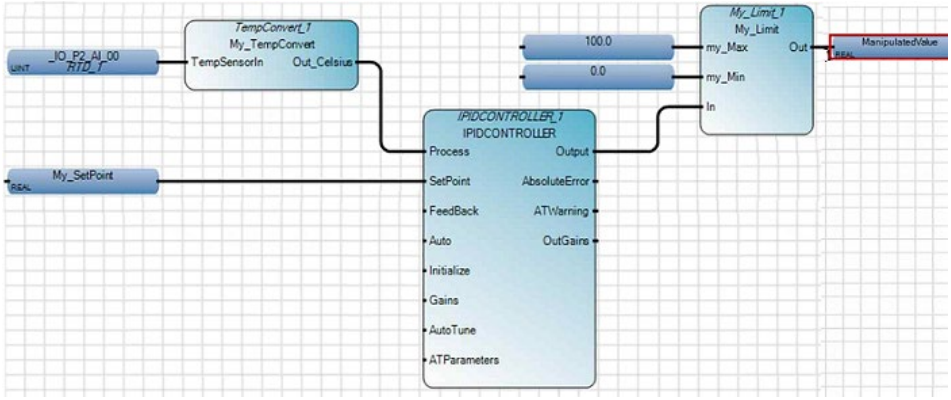
下表标识了温度控制系统中的组件，并描述了温度控制程序运行时在系统中发生的事件序列。

否	项目	描述
1	控制器输出	将 MV 发送至 PWM (On/Off)。

否	项目	描述
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	电阻温度探测器 (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。

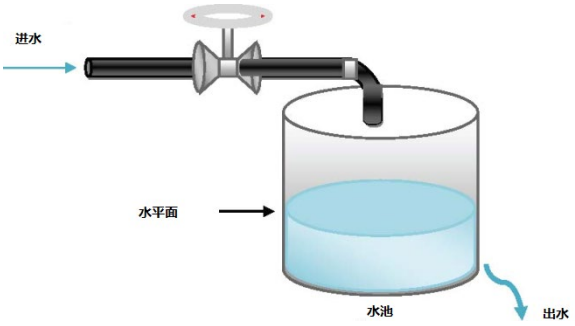
示例：温度控制功能块图

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。



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序列
- 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

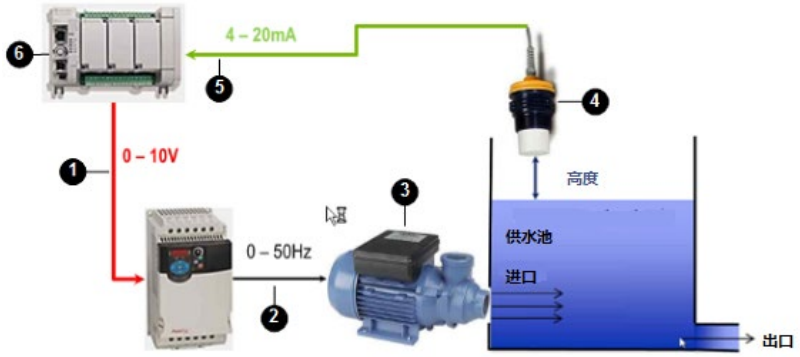
设定点、过程值和操作值

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	描述
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。



—

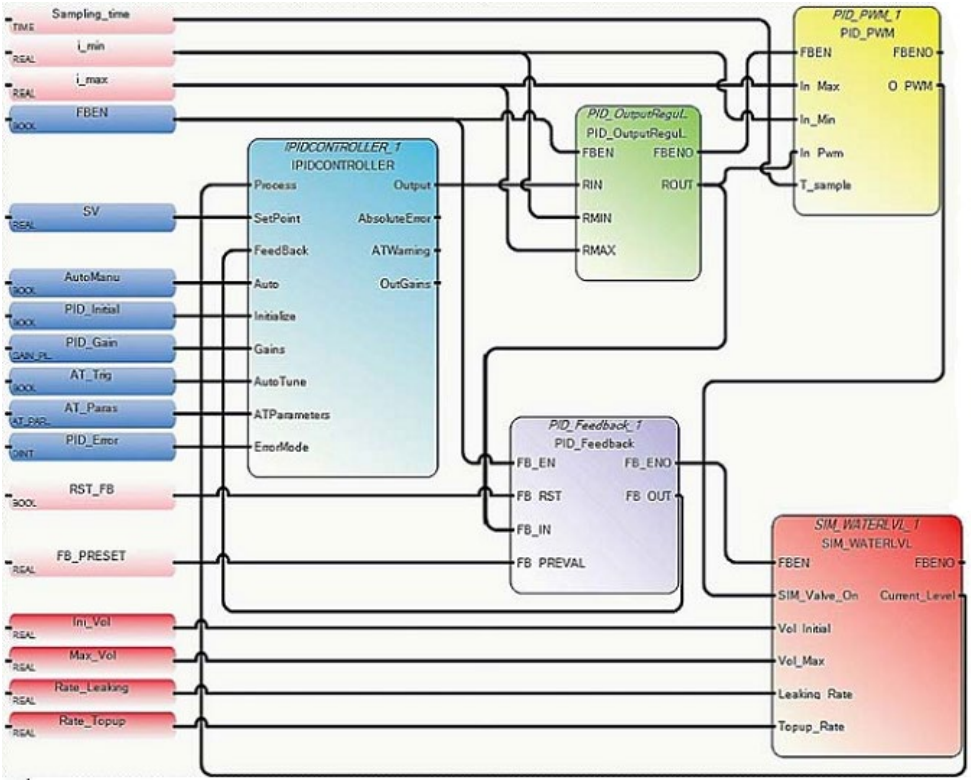
供水水位系统中的事件序列

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。

否	项目	描述
1	控制器输出	将 MV 发送至 PowerFlex 驱动器 (0-10V)。
2	PowerFlex 驱动器	控制水泵 (0-50Hz)。
3	水泵	控制供水池中的水位。
4	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。
5	控制器输入	接收 PV（供水水位 4-20mA）。
6	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

此应用程序使用下表中所述的指令，采用功能块图 (FBD) 语言开发。

功能块	描述
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。

PID（比例-积分-微分）

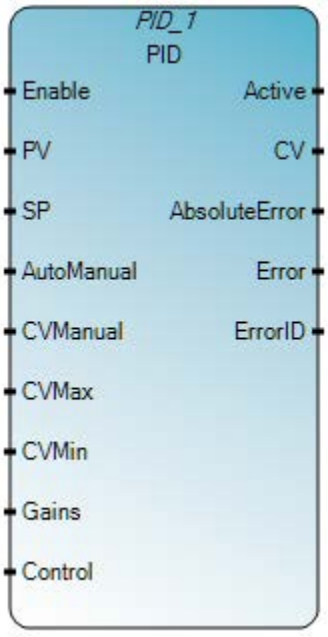
使用过程循环控制诸如温度、压力、液面或流速等物理属性的输出指令。

操作详细信息：

- 在启用后，PID 将使用包括 PID 控制器的 SP 和 Gains 在内的输入参数来控制过程。
- “运行”到“编程”模式转换，PID 指令将禁用，参数值将保留。
- 编程到运行模式转换，PID 指令保持禁用状态，直到用户将 Enable 复位为 true。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令。 TRUE - 使用当前输入参数开始执行。 FALSE - PID 不执行。将 CV 设置为 0，并计算 AbsoluteError。
PV	输入	REAL	进程值。该值通常读取自模拟输入模块。 SI 单位必须与 Setpoint 相同。
SP	输入	REAL	过程的设置点值。
AutoManual	输入	BOOL	自动或手动模式选择。 TRUE - CV 由 PID 控制。 FALSE - PID 正在运行，并且 CV 由 CVManual 输入控制。
CVManual	输入	REAL	为手动模式操作定义的控制值输入。 CVManual 的有效范围为： CVMin < CVManual < CVMax
CVMin	输入	REAL	控制值的最小限值。 如果 CV < CVMin，则 CV = CVMin。 如果 CVMin > CVMax，则发生错误。
CVMax	输入	REAL	控制值的最大限值。 如果 CV > CVMax，则 CV = CVMax。 如果 CVMax < CVMin，则发生错误。
Gains	输入	PID_GAINS	控制器 PID 的增益。 使用 PID_GAINS 数据类型可配置 Gains 参数。
Control	输入	BOOL	控制过程的方向： TRUE - 正向作用，如冷却。 FALSE - 反向作用，如加热。
Active	输出	BOOL	PID 控制器的状态。 TRUE - PID 处于活动状态。 FALSE - PID 已停止。

参数	参数类型	数据类型	描述
CV	输出	REAL	控制值输出。 如果发生任何错误，则 CV 为 0。
AbsoluteError	输出	REAL	绝对误差是过程值 (PV) 和设置点 (SP) 值之间的差值。
Error	输出	BOOL	表示存在错误条件。 TRUE - 操作过程遇到错误。 FALSE - 操作过程成功完成或指令未执行。
ErrorID	输出	USINT	标识错误的唯一数字。这些错误在 PID 错误代码中定义。

PID_GAINS 数据类型

下表描述了 PID 指令 参考页数 497的 PID_GAINS 数据类型。

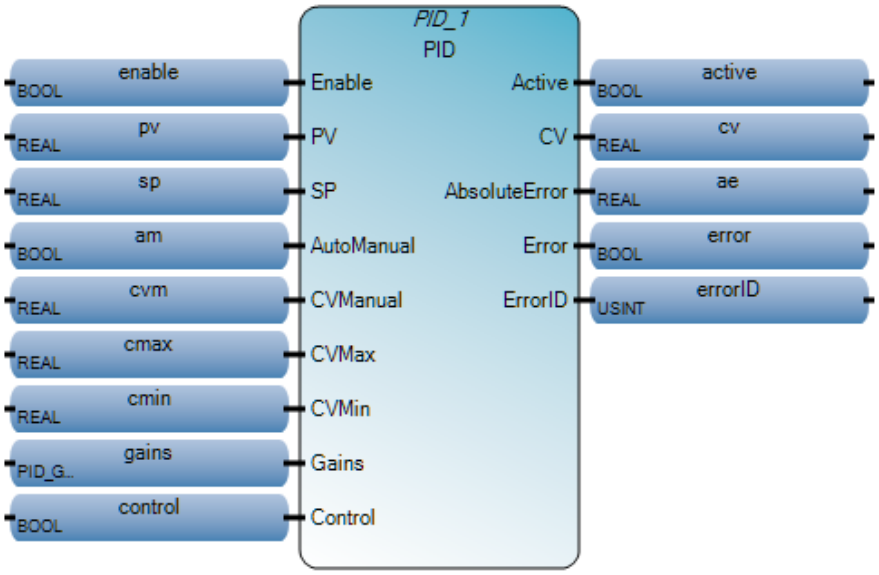
参数	参数类型	数据类型	描述
Kc	输入	REAL	PID 的控制器增益。 比例和积分取决于此增益。(>= 0.0001)。 增大 Kc 可提高响应时间性能，但也会增加 PID 的过冲和振荡。 如果 Kc 无效，则发生错误。
Ti	输入	REAL	以秒为单位的时间积分常数 (>= 0.0001)。 增大 Ti 会降低 PID 的过冲和振荡。 如果 Ti 无效，则发生错误。
Td	输入	REAL	以秒为单位的时间微分常数 (>= 0.0)。 当 Td 等于 0 时，不存在微分作用，PID 成为 PI 控制器。 增加 Td 会减少过冲，并消除 PID 控制器的振荡。 如果 Td 无效，则发生错误。
FC	输入	REAL	滤波常数 (>= 0.0)。建议的 FC 范围为 0 至 20。 增大 FC 会平滑 PID 控制器的响应。 如果 FC 无效，则发生错误。

PID 错误代码

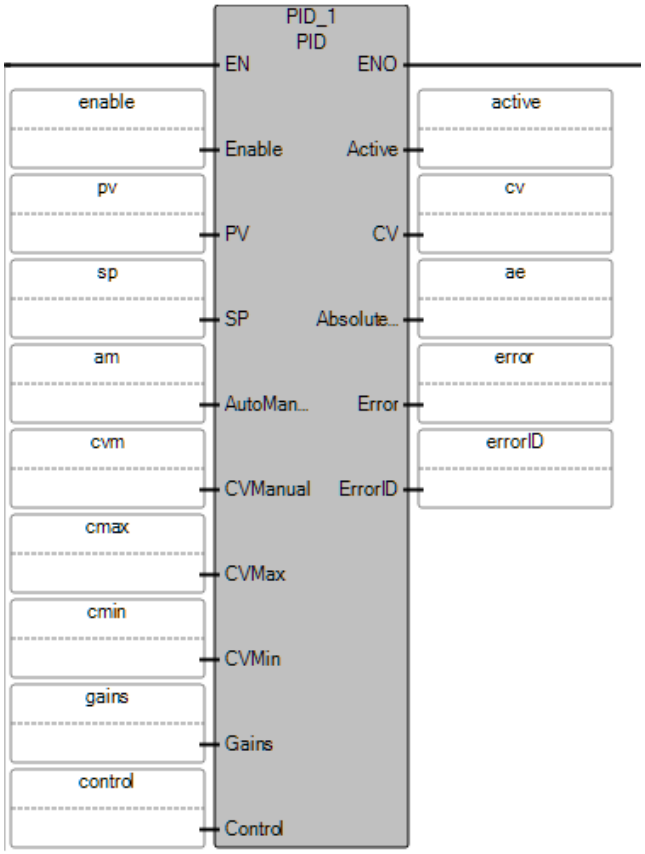
使用此表可确定 PID 错误代码和描述信息。

错误代码	错误描述
0	PID 正常工作。
1	Kc 无效。
2	Ti 无效。
3	Td 无效。
4	FC 无效。
5	CVMin > CVMax 或 CVMax < CVMin
6	CVManual < CVMin CVManual 无效。
7	CVManual > CVMax CVManual 无效。

PID 功能块图示例



PID 梯形图示例



PID 结构化文本示例

```
1  PID_1(enable, pv, sp, am, cvm, cmax, cmin, gains, control);
2  active := PID_1.Active;
3  cv := PID_1.CV;
4  ae := PID_1.AbsoluteError;
5  error := PID_1.Error;
6  errorID := PID_1.ErrorID;
```

```
PID_1(
void PID_1(BOOL Enable, REAL PV, REAL SP, BOOL AutoManual, REAL CVManual, REAL CVMax, REAL CVMin, PID_GAINS Gains, BOOL Control)
类型： PID, 比例积分微分
```

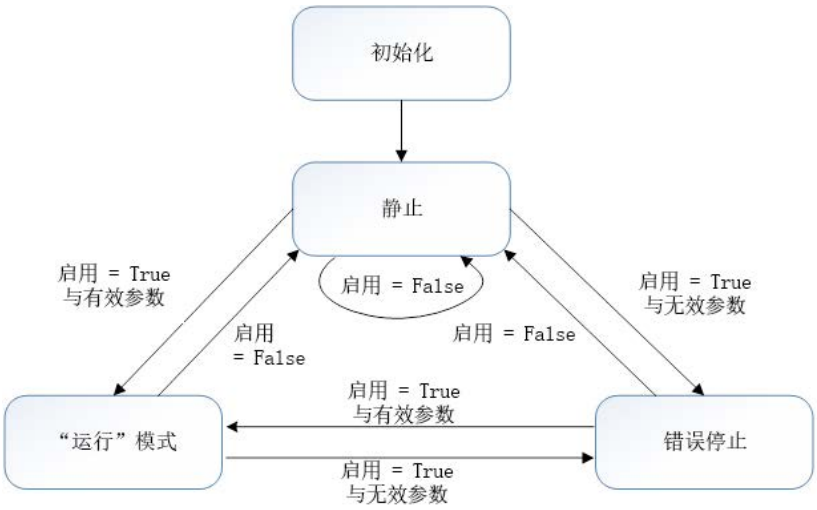
PID 结果



名称	别名	逻辑值	实际值	初始值	锁定	数据类型
PID_1		...	...	...		PID
enable			不可用			BOOL
pv		1.0	不可用	1.0		REAL
sp		10.0	不可用	10.0		REAL
am			不可用			BOOL
cvm		20.0	不可用	20.0		REAL
cmax		30.0	不可用	30.0		REAL
cmin		1.0	不可用	1.0		REAL
gains		...	...	...		PID_GAIN
control			不可用			BOOL
errorID		0	不可用			USINT
error			不可用			BOOL
ae		0.0	不可用			REAL
active			不可用			BOOL
cv		0.0	不可用			REAL

PID 指令状态机

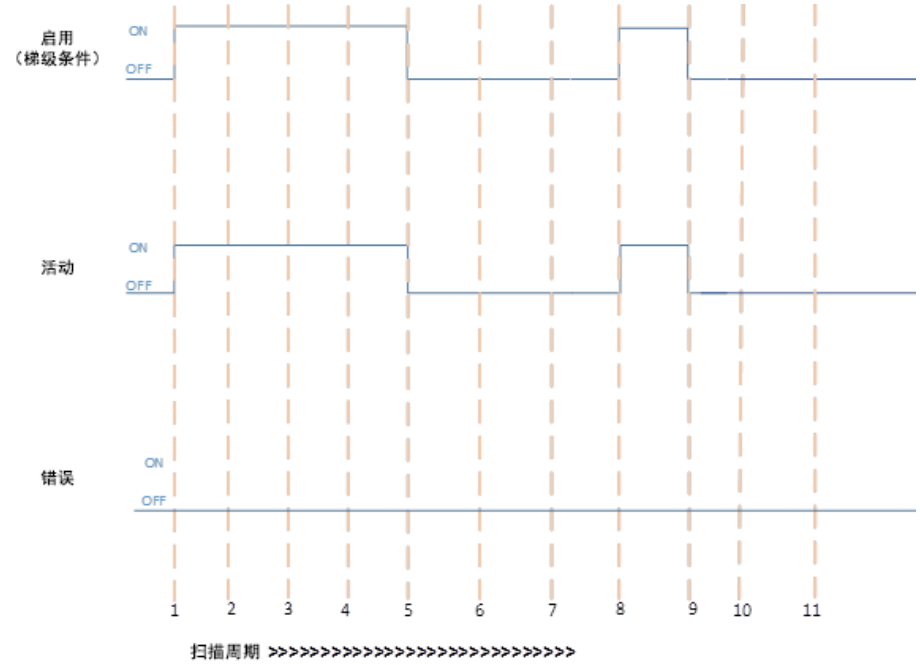
PID 状态机图描述了 PID 指令 参考页数 518 的处理状态。



PID 指令时间图

以下时序图示例描述了 PID（比例-积分-微分）指令 参考页数 518 的执行场景。

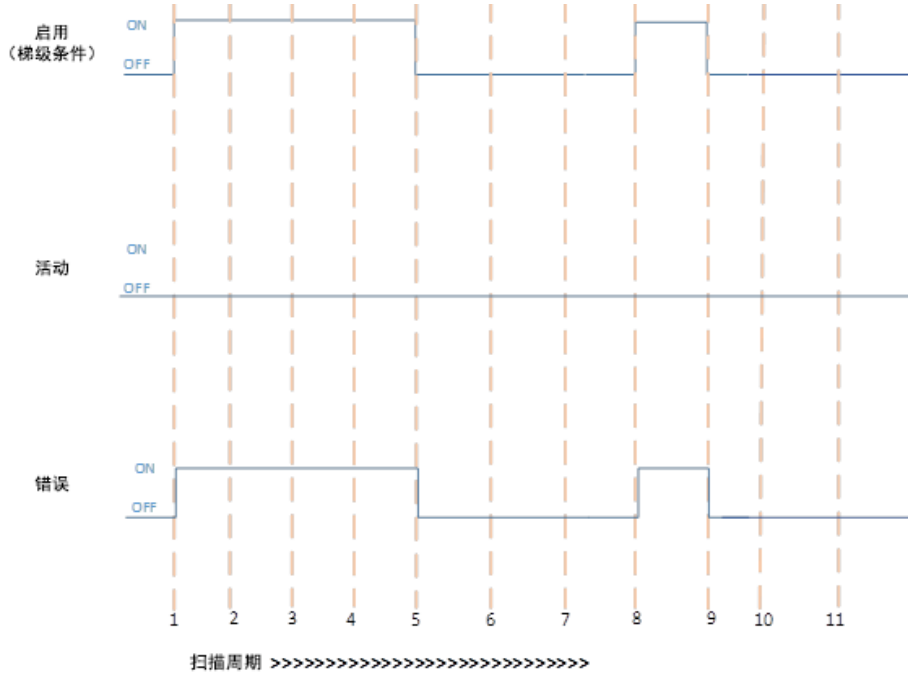
PID 执行成功



使用此表可帮助确定每个扫描周期的参数值。

扫描周期	描述
1，8	在下列情况下梯级条件变为 TRUE： • “启用”输入位为 TRUE。 • 输入参数有效。 • “活动”位为 TRUE。 • “错误”位为 FALSE。
2，3，4	梯级条件未发生更改。 • “启用”输入位为 TRUE。 • 输入参数有效。 • 更新 PID 输出参数。
5，9	在下列情况下梯级条件变为 FALSE： • “启用”位为 FALSE。 • 清除除 AbsoluteError 外的 PID 输出参数。 • AbsoluteError 基于 PV 和 SP 输入值进行计算。
6、7、10、11	梯级条件未发生更改。 • “启用”位为 FALSE。 • 清除除 AbsoluteError 外的 PID 输出参数。 • AbsoluteError 基于 PV 和 SP 输入值进行计算。

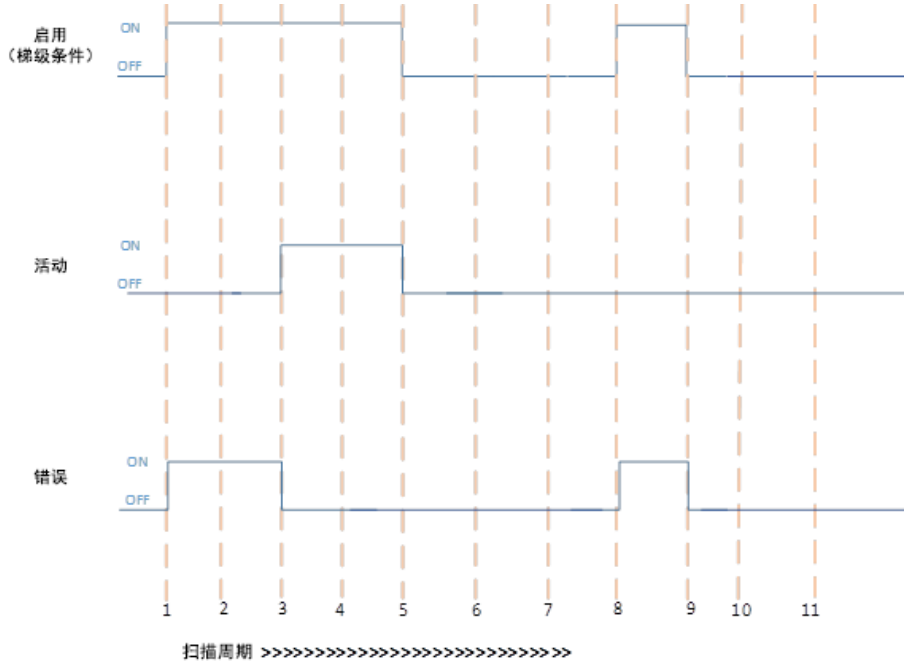
PID 执行出错



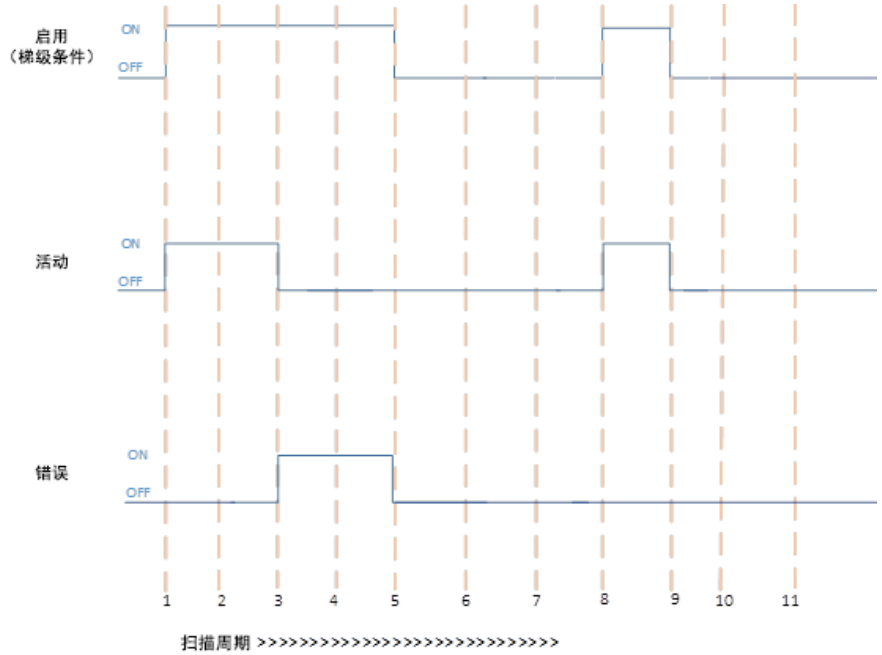
使用此表可帮助确定每个扫描周期的参数值。

扫描周期	描述
1，8	在下列情况下梯级条件变为 TRUE： • “启用”输入位为 TRUE。 • 输入参数无效。 • “活动”位为 FALSE。 • “错误”位为 TRUE。已设置 ErrorID 输出。 • CV 输出设置为 0。 • AbsoluteError 基于 PV 和 SP 输入值进行计算。
2，3，4	梯级条件未发生更改。 • “启用”输入位为 TRUE。 • 输入参数无效。 • 更新 PID 输出参数。
5，9	在下列情况下梯级条件变为 FALSE： • “启用”位为 FALSE。 • 清除除 AbsoluteError 外的 PID 输出参数。 • AbsoluteError 基于 PV 和 SP 输入值进行计算。
6、7、10、11	梯级条件未发生更改。 • “启用”位为 FALSE。 • 清除除 AbsoluteError 外的 PID 输出参数。 • AbsoluteError 基于 PV 和 SP 输入值进行计算。

PID 执行出错，然后执行成功


启用
（梯级条件）
ON
OFF
活动
ON
OFF
错误
ON
OFF
1
2
3
4
5
6
7
8
9
10
11
扫描周期 >>>>

PID 执行成功和错误



使用此表可帮助确定每个扫描周期的参数值。

扫描周期	描述
1，8	在下列情况下梯级条件变为 TRUE： • “启用”输入位为 TRUE。 • 输入参数有效。 • “活动”位为 TRUE。 • “错误”位为 FALSE。 • 更新 PID 输出参数。
2	梯级条件未发生更改。 • “启用”输入位为 TRUE。 • 输入参数有效。 • 更新 PID 输出参数。
3，4	梯级条件未发生更改。 • “启用”输入位为 TRUE。 • 输入参数无效。 • “错误”位为 TRUE。已设置 ErrorID 输出。 • CV 输出设置为 0。 • AbsoluteError 基于 PV 和 SP 输入值进行计算。
5，9	在下列情况下梯级条件变为 FALSE： • “启用”位为 FALSE。 • 清除除 AbsoluteError 外的 PID 输出参数。 • AbsoluteError 基于 PV 和 SP 输入值进行计算。
6、7、10、11	梯级条件未发生更改。 • “启用”位为 FALSE。 • 清除除 AbsoluteError 外的 PID 输出参数。 • AbsoluteError 基于 PV 和 SP 输入值进行计算。

实时时钟（RTC）指令

使用实时时钟指令配置日历和时钟。

指令	描述
RTC_READ 参考页数 358	读取实时时钟 (RTC) 模块信息。
RTC_SET 参考页数 360	将实时时钟数据设置为 RTC 模块信息。

RTC_READ（读取实时时钟）

读取实时时钟（RTC）模块信息。

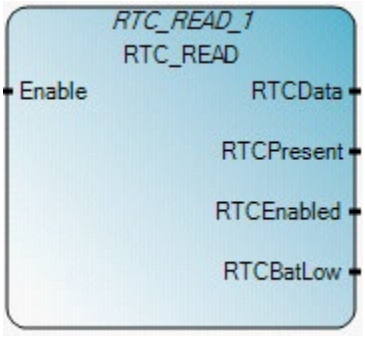
操作详细信息：

- 采用嵌入式 RTC 的 Micro810 或 Micro820 控制器：
  - RTCBatLow 始终设置为零 (0)。
  - RTCEnabled 始终设置为一 (1)。
- 如果由于失去电源而使嵌入式 RTC 丢失其电荷/内存：
  - RTCData 设置为 2000/1/1/0/0/0。
  - RTCEnabled 设置为一 (1)。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 执行 RTC 信息读取。 FALSE - 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。 使用 RTC 数据类型定义 RTCData 输出。

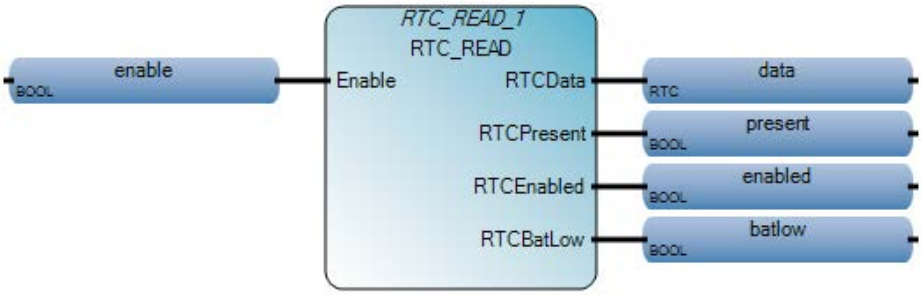
参数	参数类型	数据类型	描述
RTCPresent	输出	BOOL	TRUE - 已使用"自由运行"时钟，或已插入 RTC 硬件。 FALSE - 未使用"自由运行"时钟，或未插入 RTC 硬件。
RTCEnabled	输出	BOOL	TRUE - 已使用"自由运行"时钟，或已启用 RTC 硬件（计时）。 FALSE - 未使用"自由运行"时钟，已禁用 RTC 硬件（不计时）。
RTCBatLow	输出	BOOL	TRUE - RTC 电池电量低。 FALSE - RTC 电池电量不低。
ENO	输出	BOOL	启用"输出"。 仅适用于梯形图编程。

RTC 数据类型

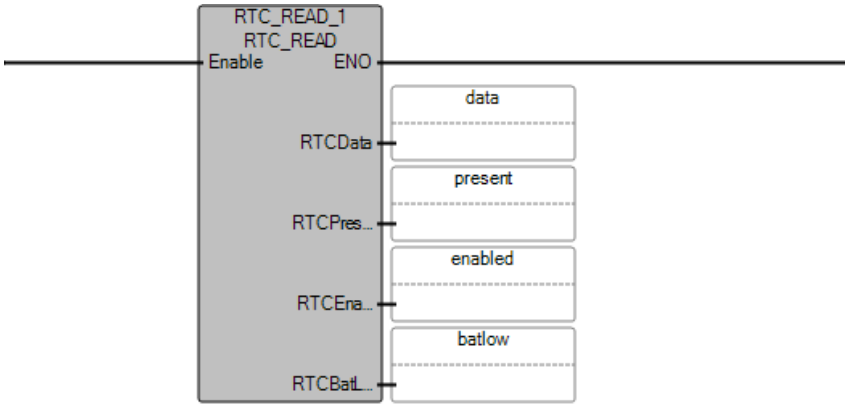
使用此表帮助确定 RTC 数据类型的参数值。

参数	数据类型	描述
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。

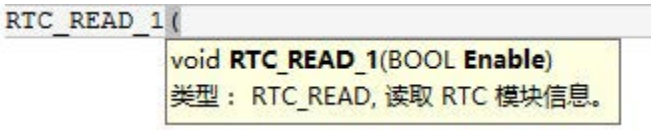
RTC_READ 功能块图示例



RTC_READ 梯形图示例



RTC_READ 结构化文本示例




```
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;
```

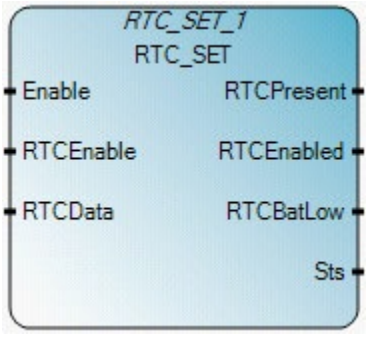
RTC_SET（设置实时时钟）

将 RTC（实时时钟）数据设置为 RTC 模块信息。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Enable	输入	BOOL	启用指令块。 TRUE - 使用输入的 RTC 信息执行 RTC_SET。在更新 RTC 时通常仅为 1 次程序扫描执行。 FALSE - 不执行 RTC_SET。设置为 FALSE 以正常操作 RTC。

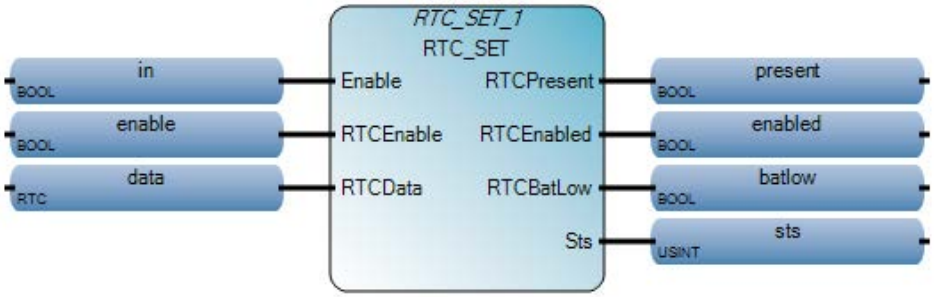
参数	参数类型	数据类型	描述
RTCEnable	输入	BOOL	TRUE - 启用 RTC 及指定的 RTC 数据。 FALSE - 禁用 RTC。
RTCData	输入	RTC	RTC 数据信息：RTC 数据类型中定义的 yy/mm/dd、hh/mm/ss、星期。 当 RTCEnable = 0 时会忽略 RTCData。
RTCPresent	输出	BOOL	TRUE - 已使用"自由运行"时钟，或已插入 RTC 硬件。 FALSE - 未使用"自由运行"时钟，或未插入 RTC 硬件。
RTCEnabled	输出	BOOL	TRUE - 已使用"自由运行"时钟，或已启用 RTC 硬件（计时）。 FALSE - 未使用"自由运行"时钟，或已禁用 RTC 硬件（不计时）。
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。 RTC_Set 状态 (Sts) 值： • 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。

RTC 数据类型

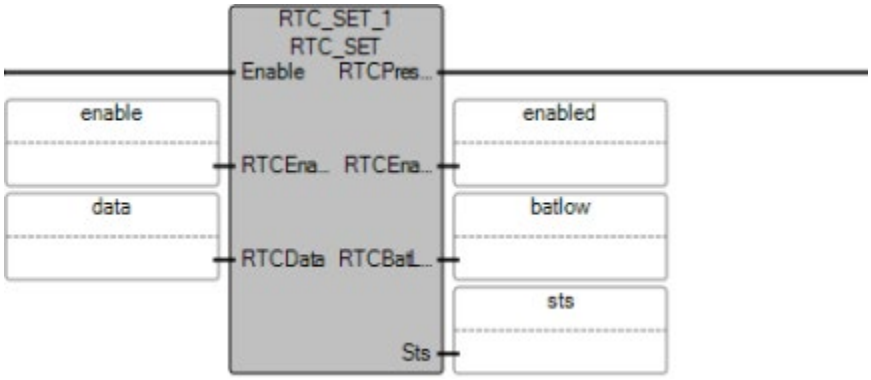
使用此表帮助确定 RTC 数据类型的参数值。

参数	数据类型	描述
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。

RTC_SET 功能块图示例



RTC_SET 梯形图示例



RTC_SET 结构化文本示例




```
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;
```

套接字指令

对不支持 Modbus TCP 和以太网/IP 的设备使用套接字协议进行以太网通信。套接字支持客户端、服务器、传输控制协议 (TCP) 和用户数据报协议 (UDP)。典型应用包括与打印机、条形码读取器和个人电脑的通信。

“运行模式更改”模式下的套接字指令行为：

- 如果在“运行模式更改”模式下在 SOCKET_OPEN、SOCKET_ACCEPT、SOCKET_READ 或 SOCKET_WRITE 指令上执行删除操作，则会删除套接字实例。
- 在“运行模式更改”模式下，如果在 BUSY 状态下操作期间对 SOCKET_READ 输入进行了任何更改，将发生错误并丢弃接收的数据包。SOCKET_READ 输入参数包括：长度、偏移、数据数组大小、数据数组变量。
- SOCKET_READ 是在“运行模式更改”模式下支持添加或修改操作的唯一套接字指令。
- 如果使用“运行模式更改”修改以太网 IP 设置，所有已创建的套接字实例会被删除，类似于 SOCKET_DELETEALL。

套接字指令的指令处理和输出更新：

- **异步**：其中所有输出与用户程序扫描（如梯形图扫描）异步更新的相应指令。异步输出不可用于沿触发检测。异步输出参数未锁定，可在相应套接字指令完成后进行更新。
- **同步**：其中所有输出与用户程序扫描同步更新的相应指令。同步输出参数已锁定，不能在相应套接字指令完成后进行修改。
- **混合**：其中一些输出与用户程序扫描同步更新的相应指令。其余输出与用户程序扫描异步更新。
- **即时指令执行**：指令先完成所需功能，然后转到下一个指令。
- **非即时指令执行**：指令采用一个以上的程序扫描来完成所需功能。指令在检测到“False > True”转换时生成输入参数的快照。

使用下表帮助确定套接字指令的使用情况。

指令	描述	TCP 客户端	TCP 服务器	UDP（采用 Open）	UDP（不采用 Open）	指令处理	指令输出更新
SOCKET_ACCEPT 参考页数 534	接受来自远程目标的 TCP 连接请求并返回用于发送和接收新创建连接上的数据的套接字实例。	NO	是	NO	NO	非即时	混合
SOCKET_CREATE 参考页数 537	创建套接字实例并返回使用后续套接字操作的实例编号。	是	是	是	是	即时	同步
SOCKET_DELETE 参考页数 541	删除创建的套接字实例。在删除之前关闭 TCP 连接。	是	是	是	是	非即时	同步

指令	描述	TCP 客户端	TCP 服务器	UDP（采用 Open）	UDP（不采用 Open）	指令处理	指令输出更新
SOCKET_DELETEALL 参考页数 543	删除创建的所有套接字实例。	是	是	是	是	非即时	同步
SOCKET_INFO 参考页数 545	返回套接字信息，如错误代码和执行状态。	是	是	是	是	即时	同步
SOCKET_OPEN 参考页数 550	TCP 连接使用指定的目标地址打开。 UDP 将目标 IP 地址和端口号关联到指定套接字。	是	NO	是	NO	非即时	同步
SOCKET_READ 参考页数 554	读取套接字上的数据。尝试接收指定的字节数，并返回收到的字节数。	是	是	是	是	非即时	混合
SOCKET_WRITE 参考页数 558	发送套接字上的数据。尝试发送请求的字节数，并返回发送的字节数。	是	是	是	是	非即时	混合

SOCKET_ACCEPT

仅传输控制协议 (TCP) 连接。接受来自远程目标的 TCP 连接请求并返回用于发送和接收新创建连接上的数据的套接字实例。

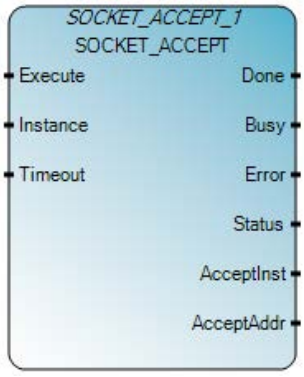
操作详细信息：

- 在执行 Socket_Accept 之前执行 SOCKET_CREATE，并指定用于接受连接的本地端口号。
- 输出与程序扫描同步更新。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro850 和 Micro870 控制器。对于 Micro800 Simulator，此指令可以添加到程序，但只是占位符，用于防止在控制器更改期间删除指令。

当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。

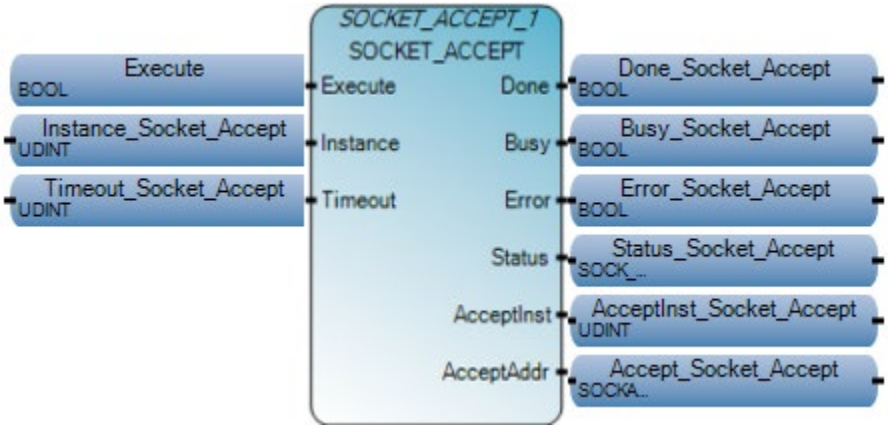


使用此表有助于确定该指令的参数值。

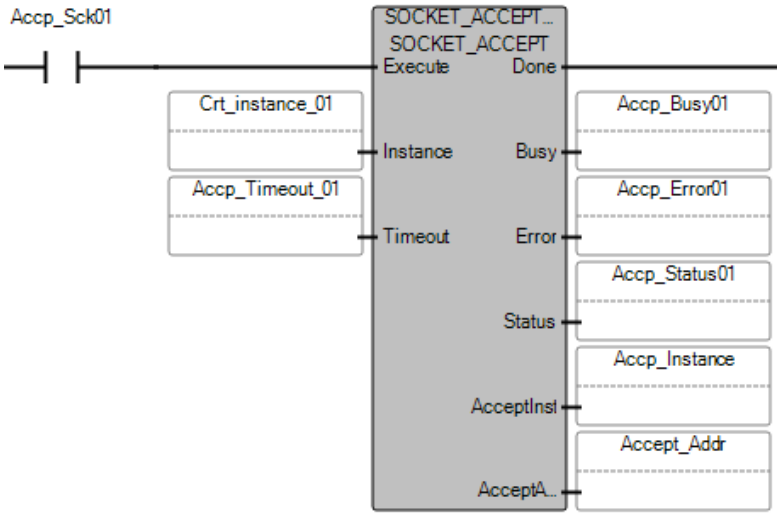
参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令块。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 未检测到上升沿，不启动指令块。

参数	参数类型	数据类型	描述
Instance	输入	UDINT	标识套接字实例。复制从 **SOCKET_CREATE** 指令返回的套接字句柄。
Timeout	输入	UDINT	**SOCKET_ACCEPT** 指令块实例的超时。如果超时值小于最小值，功能块将返回错误。 超时范围：1000 - 86400000 毫秒 将"超时"设置为 0 以使用默认值 10000（10 秒）。
Done	输出	BOOL	指示操作完成的时间。 TRUE - 操作成功完成。 FALSE - 操作正在进行或遇到错误条件。 输出与程序扫描同步更新。
Busy	输出	BOOL	TRUE - 操作未完成。 FALSE - 操作已完成。 输出与程序扫描同步更新。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。 输出与程序扫描同步更新。
Status	输出	SOCK_STATUS	使用 SOCK_STATUS 数据类型 参考页数 562 定义状态，包含 ErrorID 参考页数 562、SubErrorID 和 StatusBits 参考页数 565 信息。 输出与程序扫描同步更新。
AcceptInst	输出	UDINT	包含此套接字实例的"接受实例"。将唯一的"接受实例"编号用于此连接的后续 **SOCKET_READ** 和 **SOCKET_WRITE**。 输出与程序扫描同步更新。
AcceptAddr	输出	SOCKADDR_CFG	包含套接字的"接受地址"的数据结构。 有关详细信息，请参阅 SOCKADDR_CFG 数据类型 参考页数 561。 要指定 IP 地址 192.168.2.100 及端口 = 12000： • AcceptAddr.IPAddress[0]=192 • AcceptAddr.IPAddress[1]=168 • AcceptAddr.IPAddress[2]=2 • AcceptAddr.IPAddress[3]=100 • AcceptAddr.Port = 12000 输出与程序扫描同步更新。

SOCKET_ACCEPT 功能块图示例



SOCKET_ACCEPT 梯形图示例



SOCKET_ACCEPT 结构化文本示例

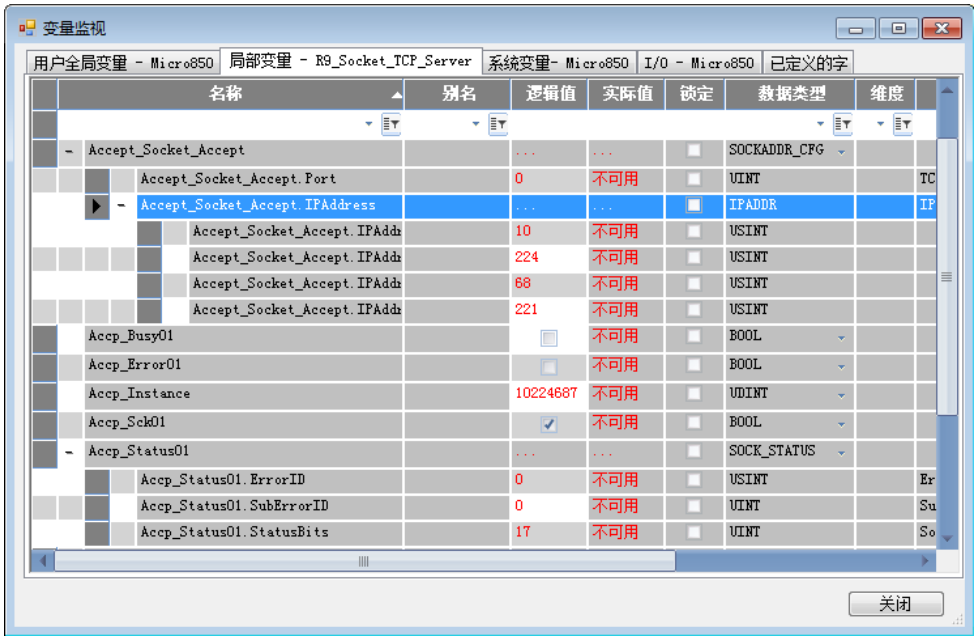
```
SOCKET_ACCEPT_1(
```

void **SOCKET_ACCEPT_1**(BOOL **Execute**, UDINT **Instance**, UDINT **Timeout**)
类型： SOCKET_ACCEPT, 接受连接

```
SOCKET_ACCEPT_1(Execute_SOCKET_ACCEPT, Instance _SOCKET_ACCEPT, Timout_SOCKET_ACCEPT);
Done_SOCKET_ACCEPT :- SOCKET_ACCEPT. Done;
Busy_SOCKET_ACCEPT :- SOCKET_ACCEPT. Busy;
Error_SOCKET_ACCEPT :- SOCKET_ACCEPT. Error;
Status_SOCKET_ACCEPT :- SOCKET_ACCEPT. Status;
AcceptInst_SOCKET_ACCEPT :- SOCKET_ACCEPT. AcceptInst;
AcceptAddr_SOCKET_ACCEPT :- SOCKET_ACCEPT. AccptAddr;
```

结果

TCP 服务器



SOCKET_CREATE

创建套接字实例并返回实例编号，以在任何后续套接字操作中用作输入 。

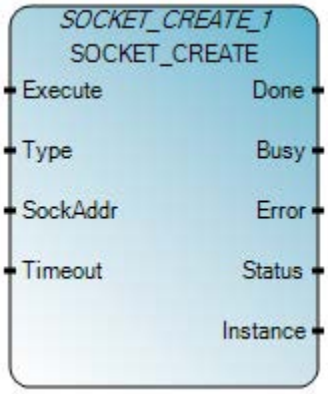
操作详细信息：

- 套接字指令 参考页数 533 支持与远程设备进行全双工通信。
- 用户数据报协议 (UDP) 连接最多可支持八个已排队的 UDP 数据报数据包。队列包含最近的数据包。
- 版本 9 或更高版本的 Micro820 和 Micro850 控制器最多可支持八个套接字实例。套接字实例支持 UDP 和 TCP 套接字：
  - 使用所有八个实例建立客户端传输控制协议 (TCP) 连接。
  - 使用所有八个实例侦听传入 TCP 连接，然后从其他设备接受八个连接。
  - 同时执行 TCP 客户端和服务器操作。
  - 同时执行 TCP 和 UDP 操作。
- 要接受到同一个端口的传入 TCP 连接，请创建新套接字实例。
- TCP 连接丢失：
  - 用户应用程序应检测 TCP 连接是否丢失并处理此事件。根据用户应用程序，考虑使控制器发生故障的选项。
    - 使控制器发生故障。
    - 尝试重新建立连接。
  - 要重新建立与其他设备的通信：
    - 删除断开的连接的套接字实例。
    - 如果连接为 TCP 客户端，则使用 SOCKET_CREATE 创建新套接字实例，并向目标设备执行 SOCKET_OPEN。
    - 如果连接为 TCP 服务器，则使用 SOCKET_CREATE 创建新套接字实例，并执行 SOCKET_ACCEPT 以等待来自远程设备的另一个连接。
  - TCP 连接的应用程序消息：
    - TCP 连接是两个应用程序之间的字节流。应用程序协议确定消息格式。
    - 消息可具有固定大小或可变大小。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro850 和 Micro870 控制器。对于 Micro800 Simulator，此指令可以添加到程序，但只是占位符，用于防止在控制器更改期间删除指令。

当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。

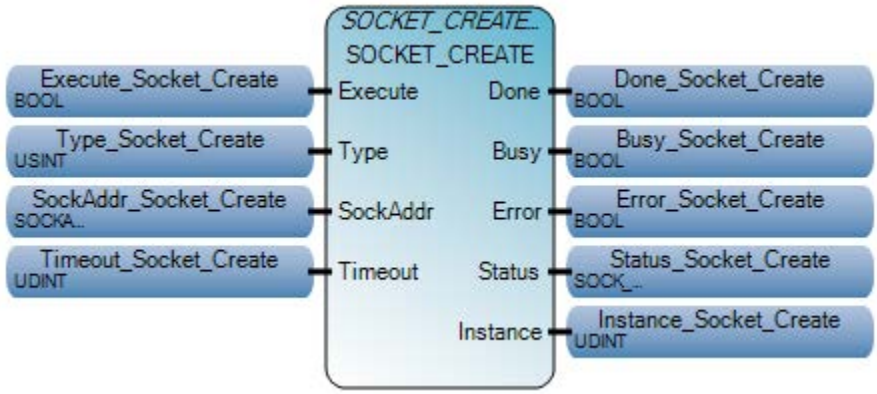


使用此表有助于确定该指令的参数值。

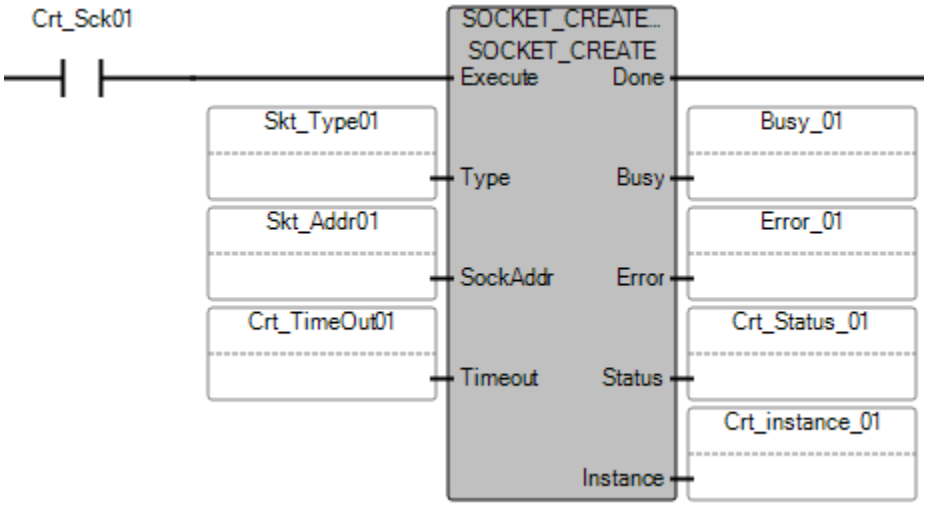
参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令块。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 未检测到上升沿。
Type	输入	USINT	指定套接字类型： • 传输控制协议 (TCP)。 • 用户数据报协议 (UDP)。
SockAddr	输入	SOCKADDR_CFG	指定套接字的地址配置。 EtherNet/IP 模块选择本地端口号，将 SockAddr 设置为 0。 指定应用程序进行侦听和接收的本地端口号，或者： • 所有数组元素均必须为零 • 对于 TCP 客户端操作，指定 0（除非您希望使用特定本地端口号）。 • 对于 TCP 服务器通信，指定用于接受传入连接请求的端口号。 • 对于 UDP，指定在特定端口上接收数据报的本地端口号。 本地端口范围：1 到 65535。 如果指定的本地端口号已被 Micro820 或 Micro850 控制器使用，将出现错误。 控制器使用以下端口号： TCP 端口： • 以太网/IP：44818 • ModbusTCP：502 • DHCP 服务器：67 • DHCP 客户端：68 UDP 端口： • 以太网/IP：2222 • DHCP 服务器：67 • DHCP 客户端：68 请参见 SOCKADDR_CFG 数据类型 参考页数 561。
Timeout	输入	UDINT	指定套接字不活动的超时。 如果套接字实例未在指定的闲置状态超时内接收到任何请求，则会删除该套接字实例。 如果已在删除套接字实例后发送请求，则会返回"套接字实例不受支持"错误。 当超时值小于最小值时，指令块会返回错误。 设置"超时"，使其不长于套接字操作之间的最长间隔。如果闲置状态超时太短，套接字实例可能会超时。超时范围：1000 - 86400000 毫秒 将"超时"设置为 0 以使用默认值 300000（5 分钟）。

参数	参数类型	数据类型	描述
Done	输出	BOOL	指示操作完成的时间。 TRUE - 操作成功完成。 FALSE - 操作正在进行或遇到错误条件。
Busy	输出	BOOL	TRUE - 操作未完成。 FALSE - 操作已完成。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。
Status	输出	SOCK_STATUS	使用 **SOCK_STATUS** 数据类型定义状态，包含 ErrorID、SubErrorID 和 StatusBits 信息。 请参见 SOCK_STATUS 数据类型 参考页数 562、套接字指令状态位 参考页数 565 和套接字错误代码 参考页数 562。
Instance	输出	UDINT	包含套接字句柄。 将实例参数用于后续套接字指令。

SOCKET_CREATE 功能块图示例



SOCKET_CREATE 梯形图示例



SOCKET_CREATE 结构化文本示例

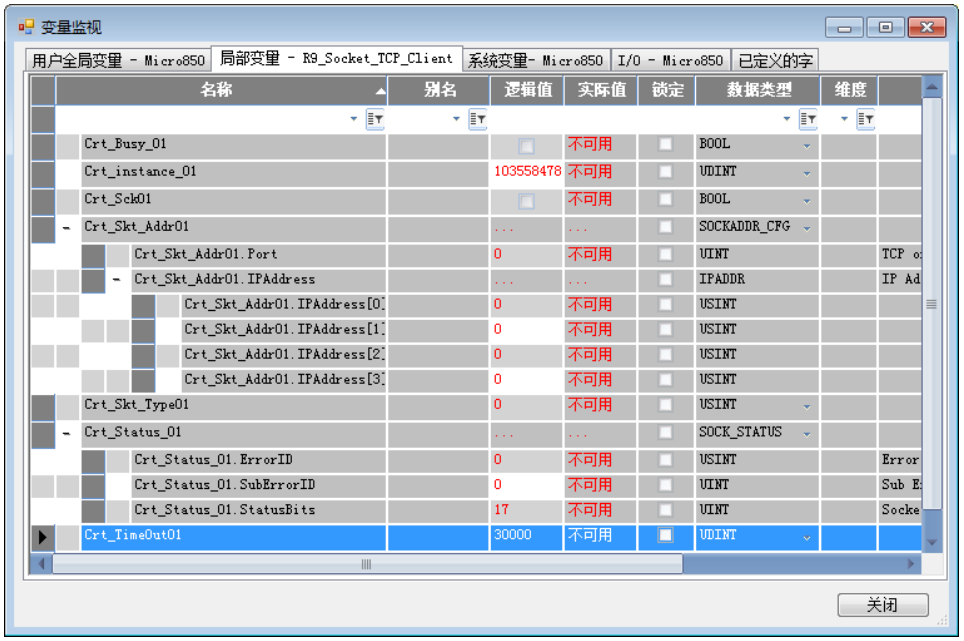
```
SOCKET_CREATE_1(
```

```
void SOCKET_CREATE_1(BOOL Execute, USINT Type, SOCKADDR_CFG SockAddr, UDINT Timeout)
类型： SOCKET_CREATE, 创建连接
```

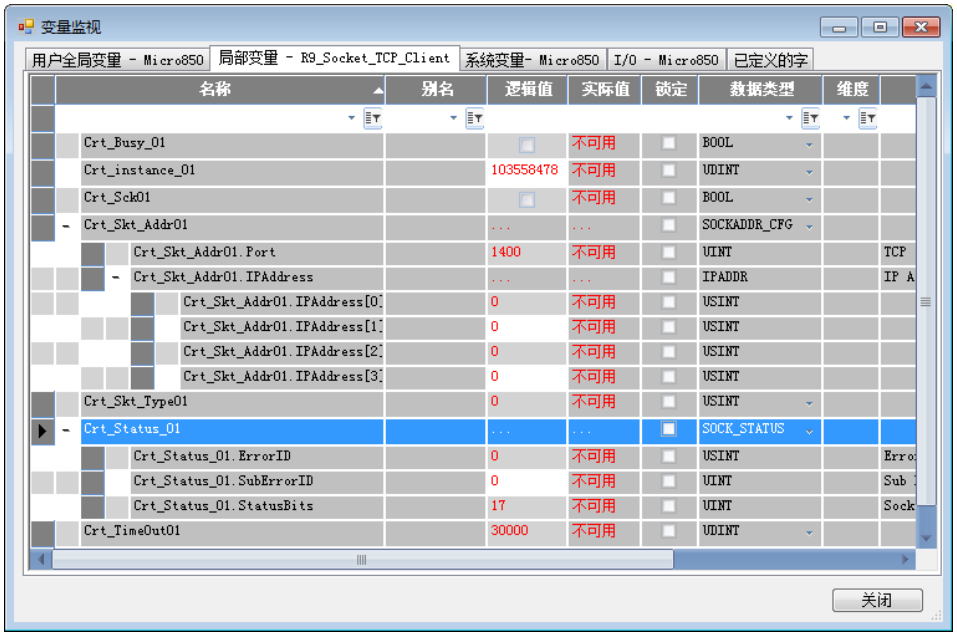
```
SOCKET_CREATE_1 (Execute_SOCKET_CREATE, Type_SOCKET_CREATE, SockAddr_SOCKET_CREATE, Timeout_SOCKET_CREATE);
Done_SOCKET_CREATE := SOCKET_CREATE.Done;
Busy_SOCKET_CREATE := SOCKET_CREATE.Busy;
Error_SOCKET_CREATE := SOCKET_CREATE.Error;
Status_SOCKET_CREATE := SOCKET_CREATE.Status;
Instance_SOCKET_CREATE := SOCKET_CREATE.Instance;
```

结果

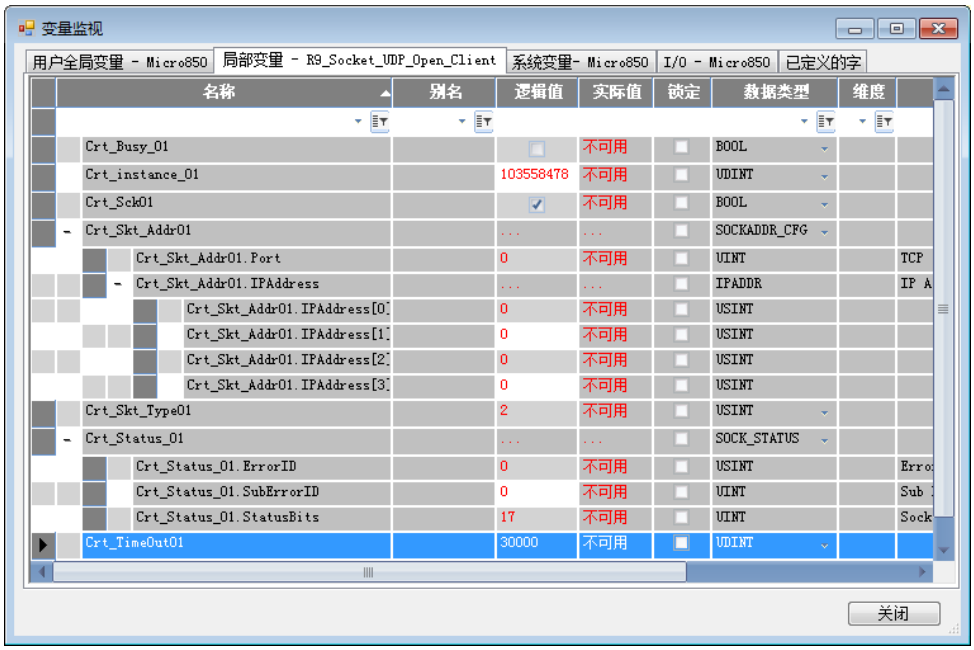
TCP 客户端示例



TCP 服务器示例



UDP 示例



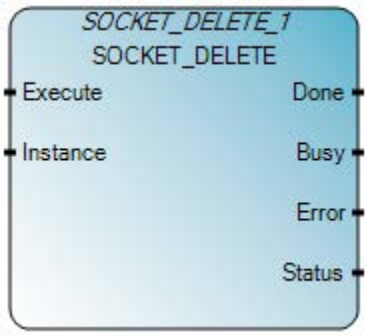
SOCKET_DELETE

删除创建的套接字实例。对于传输控制协议 (TCP) 连接，**SOCKET_DELETE** 还会在删除实例之前关闭（被动关闭）连接。输出与程序扫描同步更新。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro850 和 Micro870 控制器。对于 Micro800 Simulator，此指令可以添加到程序，但只是占位符，用于防止在控制器更改期间删除指令。

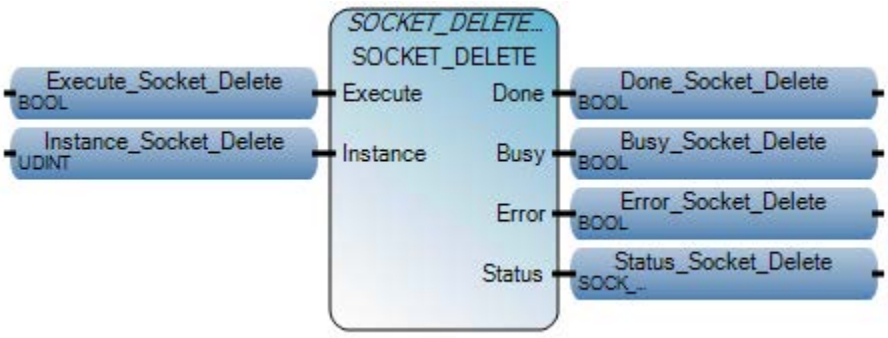
当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。



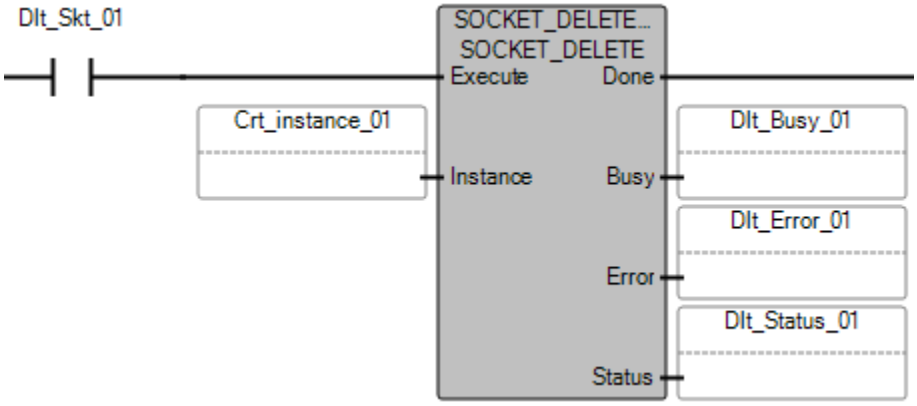
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令块。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 未检测到上升沿。
Instance	输入	UDINT	复制从"SOCKET_CREATE"或"SOCKET_ACCEPT"指令返回的套接字句柄以删除相应套接字。 • 对于 UDP 和 TCP 客户端套接字类型，复制从"SOCKET_CREATE"指令返回的套接字句柄。 • 对于 TCP 服务器套接字类型，复制从"SOCKET_ACCEPT"指令返回的套接字句柄 。
Done	输出	BOOL	指示操作完成的时间。 TRUE - 操作成功完成。 FALSE - 操作正在进行或遇到错误条件。 输出与程序扫描同步更新。
Busy	输出	BOOL	TRUE - 操作未完成。 FALSE - 操作已完成。 输出与程序扫描同步更新。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。 输出与程序扫描同步更新。
Status	输出	SOCK_STATUS	使用 SOCK_STATUS 数据类型 参考页数 562 定义状态，包含 ErrorID 参考页数 562、SubErrorID 和 StatusBits 参考页数 565 信息。 输出与程序扫描同步更新。

SOCKET_DELETE 功能块图示例



SOCKET_DELETE 梯形图示例



SOCKET_DELETE 结构化文本示例




```
SOCKET_DELETE_1 (Execute_SOCKET_DELETE, Instance_SOCKET_DELETE);
Done_SOCKET_DELETE := SOCKET_DELETE. Done;
Busy_SOCKET_DELETE := SOCKET_DELETE. Busy;
Error_SOCKET_DELETE := SOCKET_DELETE. Error;
Status_SOCKET_DELETE := SOCKET_DELETE. Status;
```

结果



SOCKET_DELETEALL

删除创建的所有套接字实例。

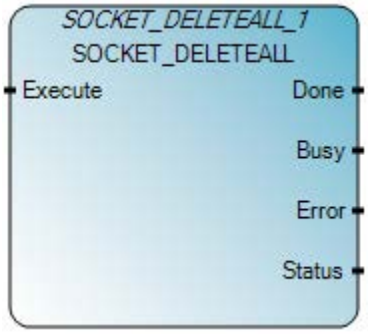
操作详细信息：

- 如果断开控制器的以太网电缆连接或更改控制器 IP 地址，可以执行 SOCKET_DELETEALL 以删除之前创建的所有套接字实例。
- 输出与程序扫描同步更新。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro850 和 Micro870 控制器。对于 Micro800 Simulator，此指令可以添加到程序，但只是占位符，用于防止在控制器更改期间删除指令。

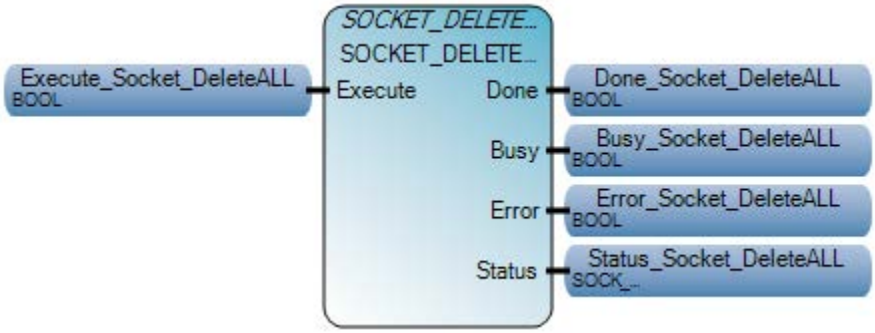
当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。



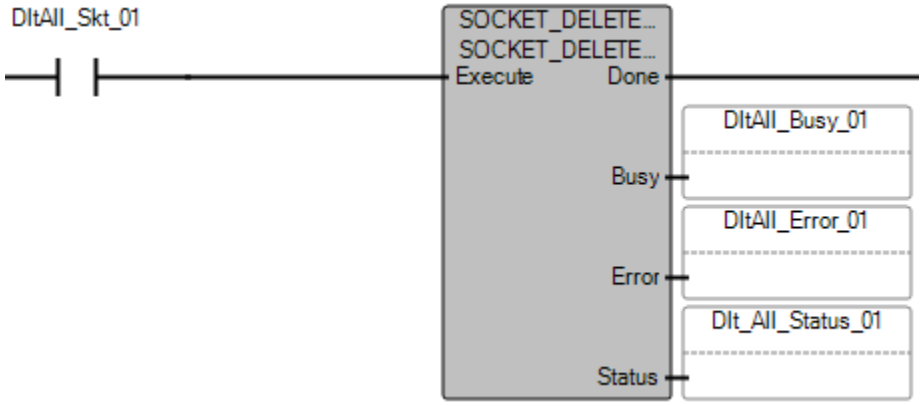
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令块。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 未检测到上升沿。
Done	输出	BOOL	指示操作完成的时间。 TRUE - 操作成功完成。 FALSE - 操作正在进行或遇到错误条件。 输出与程序扫描同步更新。
Busy	输出	BOOL	TRUE - 操作未完成。 FALSE - 操作已完成。 输出与程序扫描同步更新。
Error	输出	BOOL	当功能块执行遇到错误条件时，该字段设为 TRUE。有关详细信息，请参阅套接字错误代码 参考页数 562。 输出与程序扫描同步更新。
Status	输出	SOCK_STATUS	使用 SOCK_STATUS 数据类型 参考页数 562 定义状态，包含 ErrorID 参考页数 562、SubErrorID 和 StatusBits 参考页数 565 信息。 输出与程序扫描同步更新。

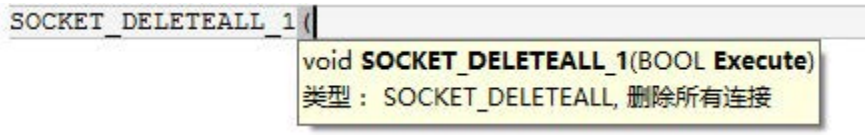
SOCKET_DELETEALL 功能块图示例



SOCKET_DELETEALL 梯形图示例

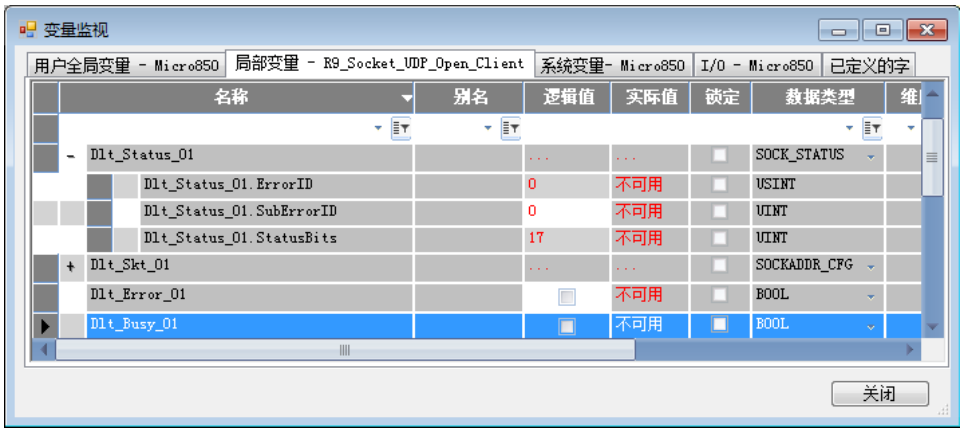


SOCKET_DELETEALL 结构化文本示例




```
SOCKET_DELETEALL (Execute_SOCKET_DELETEALL);
Done_Socket_DeleteAll := SOCKET_DELETE. Done;
Busy_Socket_DeleteAll := SOCKET_DELETE. Busy;
Error_Socket_DeleteAll := SOCKET_DELETE. Error;
Status_Socket_DeleteAll := SOCKET_DELETE. Status;
```

结果



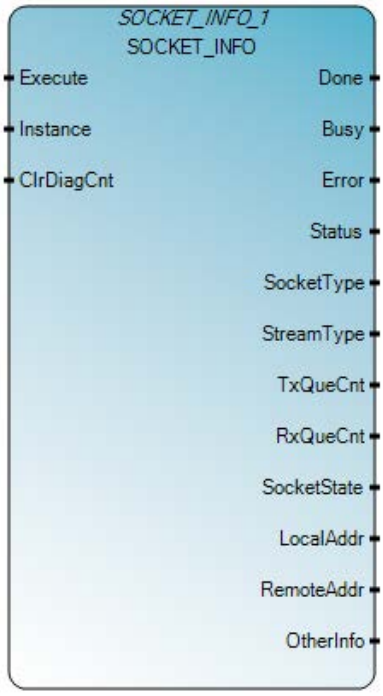
SOCKET_INFO

返回套接字实例的信息，如错误代码和执行状态。输出与程序扫描同步更新。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro850 和 Micro870 控制器。对于 Micro800 Simulator，此指令可以添加到程序，但只是占位符，用于防止在控制器更改期间删除指令。

当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。



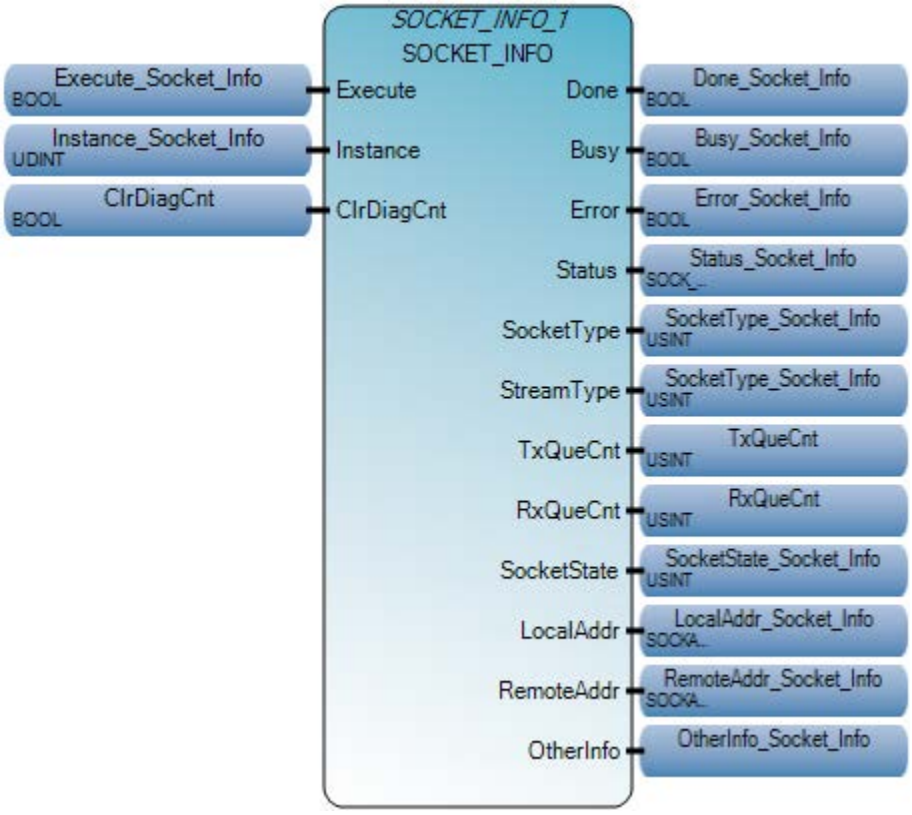
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令块。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 未检测到上升沿。
Instance	输入	UDINT	复制从"SOCKET_CREATE"或"SOCKET_ACCEPT"指令返回的套接字句柄以删除相应套接字。 • 对于 UDP 和 TCP 客户端套接字类型，复制从"SOCKET_CREATE"指令返回的套接字句柄。 • 对于 TCP 服务器套接字类型，复制从"SOCKET_ACCEPT"指令返回的套接字句柄。 当"实例"为 0 时，返回所有套接字实例的摘要。
ClrDiagCnt	输入	BOOL	TRUE - 清除传输控制协议 (TCP) 或用户数据报协议 (UDP) 诊断计数器信息。 FALSE - 不清除 TCP 或 UDP 计数器信息。 如 TCP 和 UDP，"OtherInfo"数组索引 1 至 6。 当"实例"为 0 时，清除 **OtherInfo** 数组索引 7 至 14。
Done	输出	BOOL	指示操作完成的时间。 TRUE - 操作成功完成。 FALSE - 操作正在进行或遇到错误条件。 输出与程序扫描同步更新。
Busy	输出	BOOL	TRUE - 操作未完成。 FALSE - 操作已完成。 输出与程序扫描同步更新。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。 输出与程序扫描同步更新。

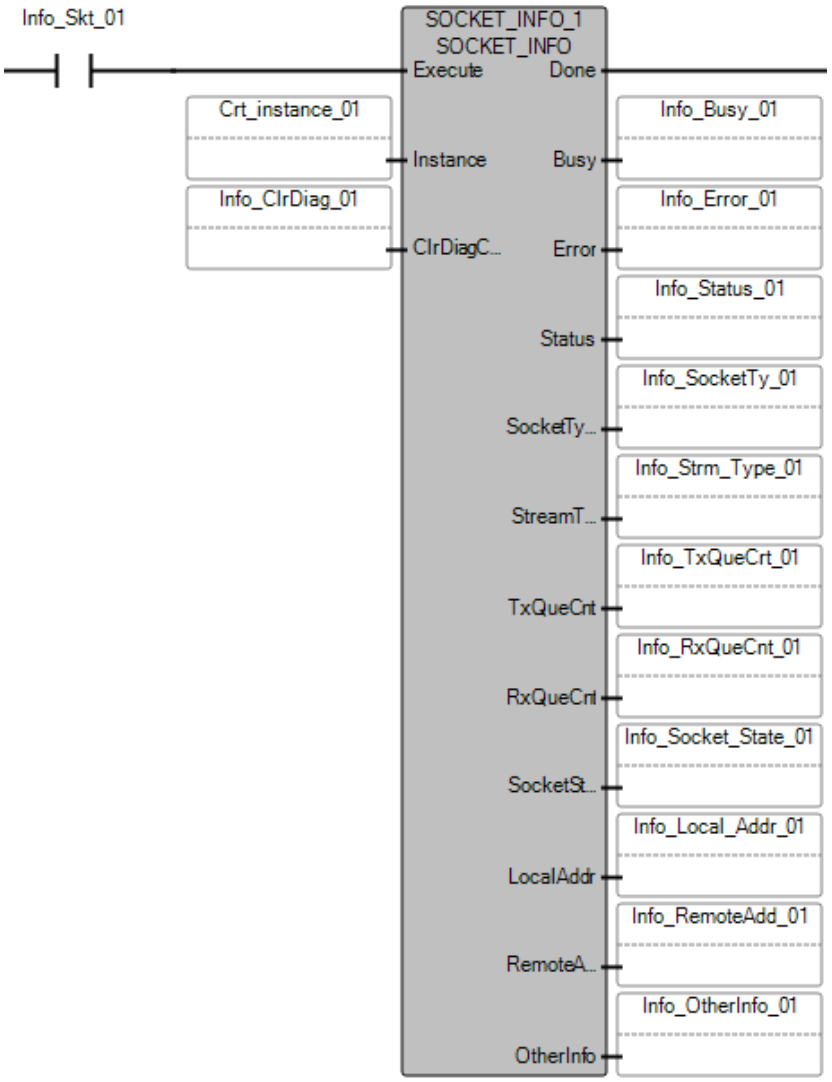
参数	参数类型	数据类型	描述
Status	输出	SOCK_STATUS	使用 SOCK_STATUS 数据类型 参考页数 562 定义状态，包含 ErrorID 参考页数 562、SubErrorID 和 StatusBits 参考页数 565 信息。 输出与程序扫描同步更新。
SocketType	输出	USINT	套接字实例类型： • 0 - 未使用 • 1 - TCP • 2 - UDP 当 Socket_Info"实例"为 0 时，"SocketType"将显示为 0。
StreamType	输出	USINT	套接字流类型： • 0 - 无 • 1 - TCP 服务器 • 2 - TCP 客户端 当 Socket_Info"实例"为 0 时，**StreamType** 将显示为 0。
TxQueCnt	输出	USINT	当前位于队列中的 Tx 消息的数量。 当 Socket_Info"实例"为 0 时，"TxQueCnt"将显示为 0。
RxQueCnt	输出	USINT	当前位于队列中的 Rx 消息的数量。 当 Socket_Info"实例"为 0 时，**RxQueCnt** 将显示为 0。
SocketState	输出	USINT	套接字指令状态信息。有关详细信息，请参阅套接字状态机。 当 Socket_Info"实例"为 0 时，"SocketState"将显示为 0。
LocalAddr	输出	SOCKETADDR_CFG	套接字的本地地址。有关详细信息，请参阅 SOCKADDR_CFG 数据类型 参考页数 561。 当 Socket_Info"实例"为 0 时，"LocalAddr"将显示为 0。
RemoteAddr	输出	SOCKETADDR_CFG	套接字的远程地址。有关详细信息，请参阅 SOCKADDR_CFG 数据类型 参考页数 561。 在下列情况下，**RemoteAddr** 将显示为 0： • Socket_Info"实例"为 0 • 没有 **SOCKET_OPEN** 的用户数据报协议 (UDP) 连接 • 具有 **SOCKET_OPEN** 并且已禁用 **RxFilter** 的 UDP。
OtherInfo		UDINT[1..15]	套接字"实例"配置为 TCP，数组索引描述为： • 1 - 已发送数据包：在套接字上发送的 TCP 数据包总数。 • 2 - 已发送字节：在套接字上发送的 TCP 字节总数。 • 3 - 已接收数据包：在套接字上接收的 TCP 数据包总数。 • 4 - 已接收字节：在套接字上接收的 TCP 字节总数。 • 5 - 重新传输数据包：TCP 数据包重新传输总数。 • 6 - 校验和错误：套接字上存在校验和错误的 TCP 数据包总数。 • 7 - TCP 状态：套接字的当前状态。 • （8 至 11）- OtherInfo 不受 TCP 支持，显示为 0。 • 12、13、14、15 - 显示为 0。 套接字实例配置为 UDP，数组索引描述为： • 1 - 已发送数据包：在套接字上发送的 UDP 数据包总数。 • 2 - 已发送字节：在套接字上发送的 UDP 字节总数。 • 3 - 已接收数据包：在套接字上接收的 UDP 数据包总数。 • 4 - 已接收字节：在套接字上接收的 UDP 字节总数。 • 5 - 已丢弃数据包：为套接字丢弃（由于超过最大队列大小界限 8）的 UDP 接收数据包总数。 • 6 - 校验和错误：套接字上存在校验和错误的 UDP 数据包总数。 • 7 至 15 - 显示为 0。 套接字实例配置为 0，数组索引描述为：

参数	参数类型	数据类型	描述
			• 1 - 可用套接字实例的计数。支持的最大套接字数。 • 2 - 已用套接字实例的计数。已成功创建的套接字数。 • 3 - 已作为 TCP 创建的套接字实例的数量。 • 4 - 已作为 TCP 客户端创建的套接字实例的数量。 • 5 - 已作为 TCP 服务器创建的套接字实例的数量。 • 6 - 已作为 UDP 创建的套接字实例的数量。 • 7 - **SOCKET_READ** 当套接字实例配置为 TCP 时的成功计数。 • 8 -"SOCKET_WRITE"当套接字实例配置为 TCP 时的成功计数。 • 9 -"SOCKET_READ"当套接字实例配置为 TCP 时的失败计数。 • 10 -"SOCKET_WRITE"当套接字实例配置为 TCP 时的失败计数。 • 11 -"SOCKET_READ"当套接字实例配置为 UDP 时的成功计数。 • 12 -"SOCKET_WRITE"当套接字实例配置为 UDP 时的成功计数。 • 13 -"SOCKET_READ"当套接字实例配置为 UDP 时的失败计数。 • 14 - **SOCKET_WRITE** 当套接字实例配置为 UDP 时的失败计数。 • 15 - 显示为 0。

SOCKET_INFO 功能块图示例



SOCKET_INFO 梯形图示例



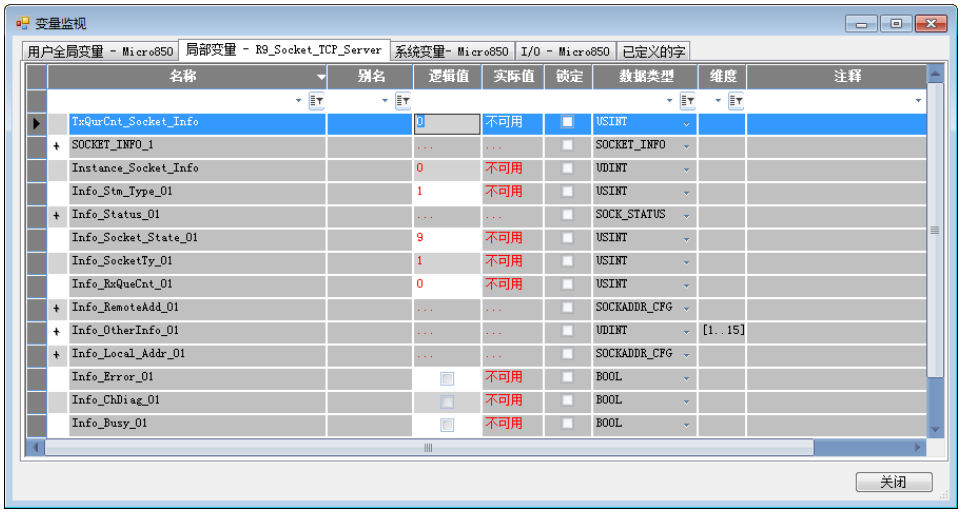
SOCKET_INFO 结构化文本示例

```
SOCKET_INFO_1(
```

void **SOCKET_INFO_1**(BOOL **Execute**, UDINT **Instance**, BOOL **ClrDiagCnt**)
类型： SOCKET_INFO, 连接实例信息

```
SOCKET_INFO_1 (Execute_SOCKET_INFO, Instance_SOCKET_INFO, ClrDiagC_SOCKET_INFO);
Done_SOCKET_INFO := SOCKET_INFO. Done;
Busy_SOCKET_INFO := SOCKET_INFO. Busy;
Error_SOCKET_INFO := SOCKET_INFO. Error;
Status_SOCKET_INFO := SOCKET_INFO. Status;
SocketType_SOCKET_INFO := SOCKET_INFO. SocketTy;
StreamType_SOCKET_INFO := SOCKET_INFO. StreamType;
TxQueCnt_SOCKET_INFO := SOCKET_INFO. TxQueCnt;
RxQueCnt_SOCKET_INFO := SOCKET_INFO. RxQueCnt;
SocketState_SOCKET_INFO := SOCKET_INFO. SocketState;
LocalAddr_SOCKET_INFO := SOCKET_INFO. LocalAddr;
RemoteAddr_SOCKET_INFO := SOCKET_INFO. RemoteAddr;
OtherInfo SOCKET INFO := SOCKET INFO. OtherInfo;
```

结果



SOCKET_OPEN

打开指定目标地址的连接，以建立传输控制协议 (TCP) 连接。对于用户数据报协议 (UDP) 连接，会将目标 IP 地址和端口号关联到指定套接字。

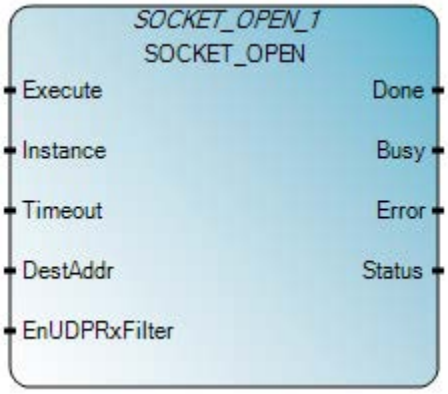
操作详细信息：

- 对于使用“SOCKET_OPEN”的用户数据报协议 (UDP) 连接，无需在每次发送数据时都使用 IP 地址和端口号。
- 对于未使用“SOCKET_OPEN”的 UDP 连接，必须在每次使用“SOCKET_WRITE”发送数据时指定目标地址。
- 当使用“SOCKET_READ”时，除了接收数据外，还将返回发送方的地址。发送方地址可用于使用“SOCKET_WRITE”发送响应。
- “SOCKET_OPEN”操作可能会在超时时间之前返回，而不创建传输控制协议 (TCP) 连接。如果目标设备正在运行但未侦听指定端口号上的连接，可能会出现这种情况，**SOCKET_OPEN** 会在超时时间之前返回并出错。
- 输出与程序扫描同步更新。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro850 和 Micro870 控制器。对于 Micro800 Simulator，此指令可以添加到程序，但只是占位符，用于防止在控制器更改期间删除指令。

当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。

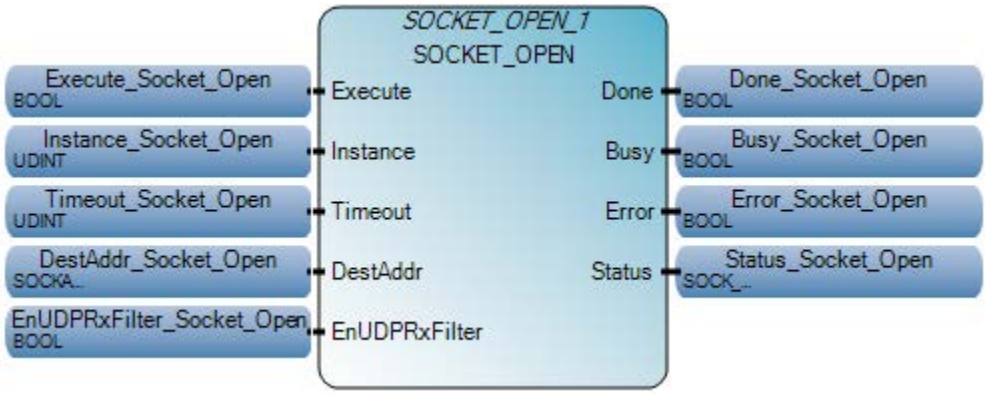


使用此表有助于确定该指令的参数值。

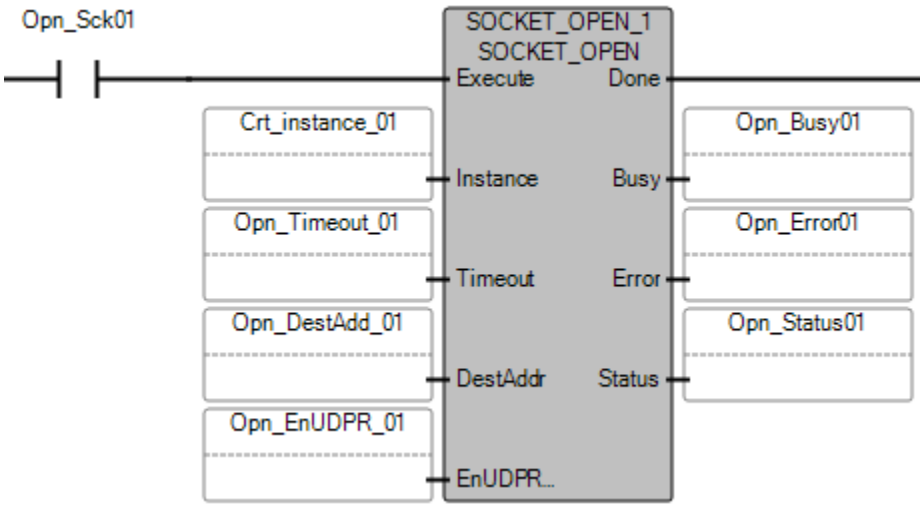
参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令块。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 未检测到上升沿。
Instance	输入	UDINT	复制从"SOCKET_CREATE"功能块返回的套接字句柄。
Timeout	输入	UDINT	SOCKET_OPEN 功能块的超时。 如果"超时"值小于最小值，功能块会返回错误。 超时范围：1000 - 1800000 毫秒 将"超时"设置为 0 以使用默认值 10000（10 秒）。
DestAddr	输入	SOCKADDR_CFG 参考页数 561	目标连接的地址。 必须在远程主机的 IP 地址和端口号之间建立连接。 以下 IP 地址不受 DestAddr 支持： • 自 IP 地址 • 回路地址 • 0.0.0.0 • 广播地址（仅支持 TCP），异常地址： • D 类多播地址 (224.x.x.x) • 本地链接地址 (169.254.x.x) IP 地址 192.168.2.100 和端口 12000 的示例： • DestAddr.IPAddress[0]=192 • DestAddr.IPAddress[1]=168 • DestAddr.IPAddress[2]=2 • DestAddr.IPAddress[3]=100 • DestAddr.Port = 12000
EnUDPRxFilter	输入	BOOL	对于 UDP 套接字，当使用 SOCKET_OPEN 和 Enable EnUDPRxFilter 时，无需在每次读取数据时都使用特定 IP 地址和端口号的数据包过滤器。 • 已创建 UDP 套接字（采用 Open）。DestAddr 为 IP 192.168.1.157 / 端口 161。EnUDPRxFilter 启用，执行 Socket_Read。 • 当控制器从 DestAddr (192.168.1.157 / 161) 接收数据时， Socket_Read 已成功完成操作。如果控制器从任何其他 IP 或端口接收数据，则 socket_Read 将忽略该数据包，并等待 DestAddr 数据包。 EnUDPRxFilter 已禁用：

参数	参数类型	数据类型	描述
			• 执行 SOCKET_READ。控制器在已配置的 UDP 端口上接收任何数据，并且 SOCKET_READ 已成功完成操作。 • 应用程序检查传入数据包是否来自于预期的设备。 • 应用程序基于 SOCKET_READ 输出参数 UDPAddr 处理过滤。
Done	输出	BOOL	指示操作完成的时间。 TRUE - 操作成功完成。 FALSE - 操作正在进行或遇到错误条件。 输出与程序扫描同步更新。
Busy	输出	BOOL	TRUE - 操作未完成。 FALSE - 操作已完成。 输出与程序扫描同步更新。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。 输出与程序扫描同步更新。
Status	输出	SOCK_STATUS	使用 SOCK_STATUS 数据类型 参考页数 562 定义状态，包含 ErrorID 参考页数 562、SubErrorID 和 StatusBits 参考页数 565 信息。 输出与程序扫描同步更新。

SOCKET_OPEN 功能块图示例



SOCKET_OPEN 梯形图示例



SOCKET_OPEN 结构化文本示例

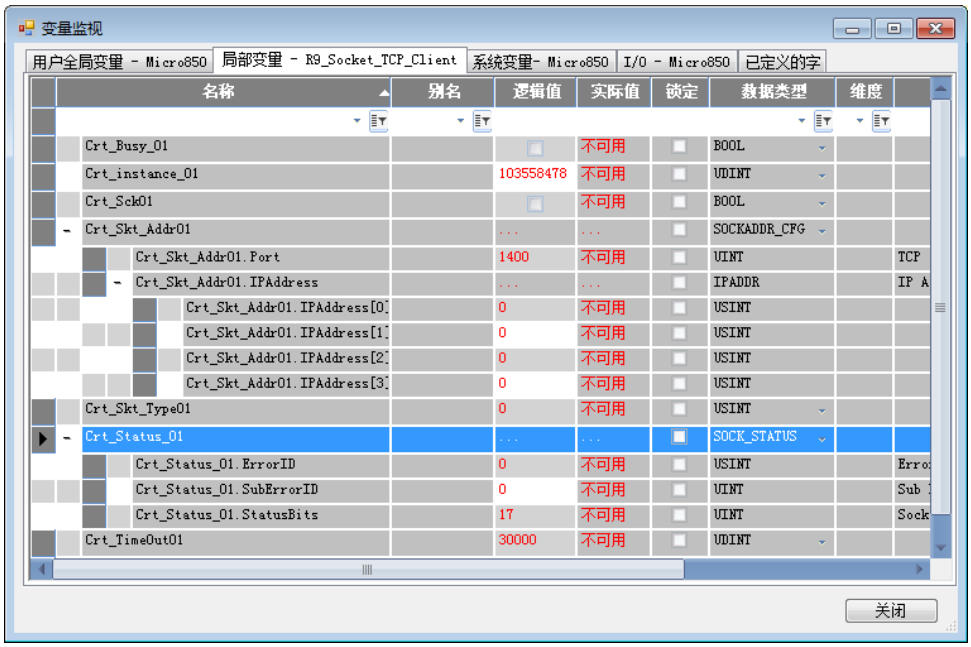
```
SOCKET_OPEN_1(
```

void **SOCKET_OPEN_1**(BOOL **Execute**, UDINT **Instance**, UDINT **Timeout**, SOCKADDR_CFG **DestAddr**, BOOL **EnUDPRxFilter**)
类型： SOCKET_OPEN, 打开连接

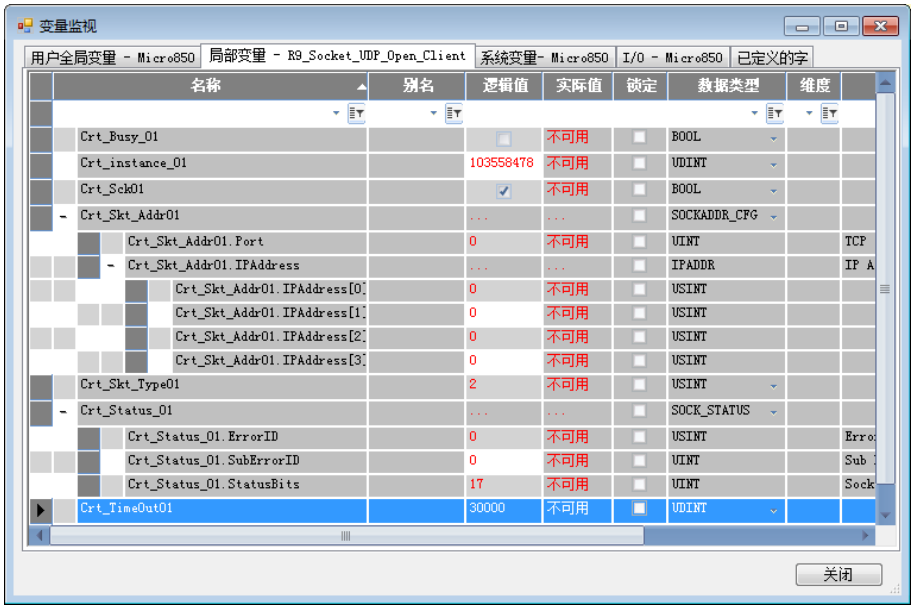
```
SOCKET_OPEN_1 (Execute_SOCKET_OPEN. Instance_SOCKET_OPEN, Timeout_SOCKET_OPEN, DestAddr_SOCKET_OPEN, EnVDPR_SOCKET_OPEN)
Done_SOCKET_OPEN := _SOCKET_OPEN. Done:
Busy_SOCKET_OPEN := _SOCKET_OPEN. Busy:
Error_SOCKET_OPEN := _SOCKET_OPEN. Error:
Status_SOCKET_OPEN := _SOCKET_OPEN. Status:
```

结果

TCP 示例



UDP 示例



SOCKET_READ

读取套接字上的数据，并返回指定的字节数。对于传输控制协议 (TCP)，返回任何已接收到的数据，最多可返回所请求的字节数。对于用户数据报协议 (UDP)，在数据报可用时完成。

操作详细信息：

以下 **SOCKET_READ** 行为可能会影响现有通信（包括非套接字通信）。

- 如果“SOCKET_READ”操作未与远程设备同步执行，则控制器将保留远程设备接收数据包，直到发生以下操作之一：
  - “SOCKET_READ”已执行。
  - 套接字超时时间到期。
  - 从远程设备接收到 RST。
  - “SOCKET_DELETE”或“SOCKET_DELETEALL”已执行。
  - 用户执行**运行模式更改**，此操作会删除所有已创建的套接字实例。
  - 控制器从运行模式更改为编程模式，这将删除所有已创建的套接字实例。
  - 控制器从运行模式更改为编程模式，这将清除套接字诊断计数器信息和各个套接字计数器信息。
- 如果在执行 SOCKET_READ 操作期间 (BUSY = True) Length 或 Offset 参数值发生更改，将发生错误并丢弃接收数据包。
- “SOCKET_READ”指令返回的字节数可能少于所请求的字节数。RxLength 包含已接收的数据字节数。写入程序以检查 RxLength，然后发出其他读取请求以接收完整应用程序消息。
- 在“运行模式更改”模式下，如果在 BUSY 状态下操作期间更改 **SOCKET_READ** 输入，将发生错误并丢弃接收的数据包。
- 输出与程序扫描同步更新。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro850 和 Micro870 控制器。对于 Micro800 Simulator，此指令可以添加到程序，但只是占位符，用于防止在控制器更改期间删除指令。

当指令应用于模拟的控制器（2080-LC50-48QWB-SIM）时，输出始终被复位。

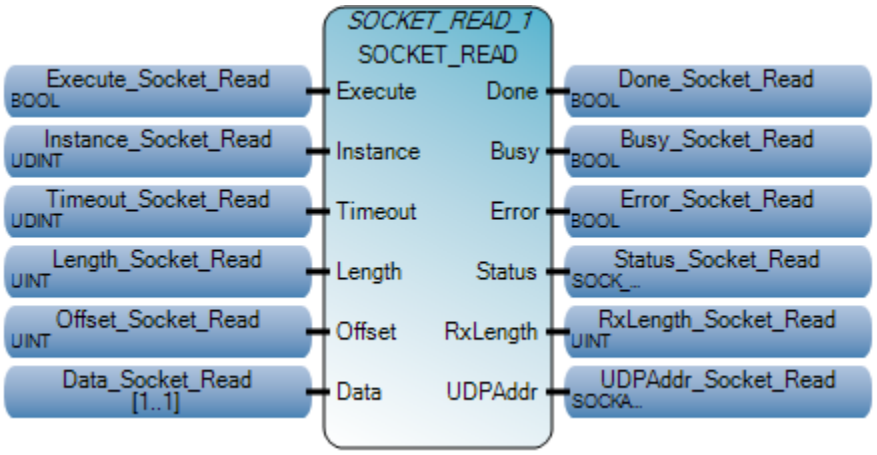


使用此表有助于确定该指令的参数值。

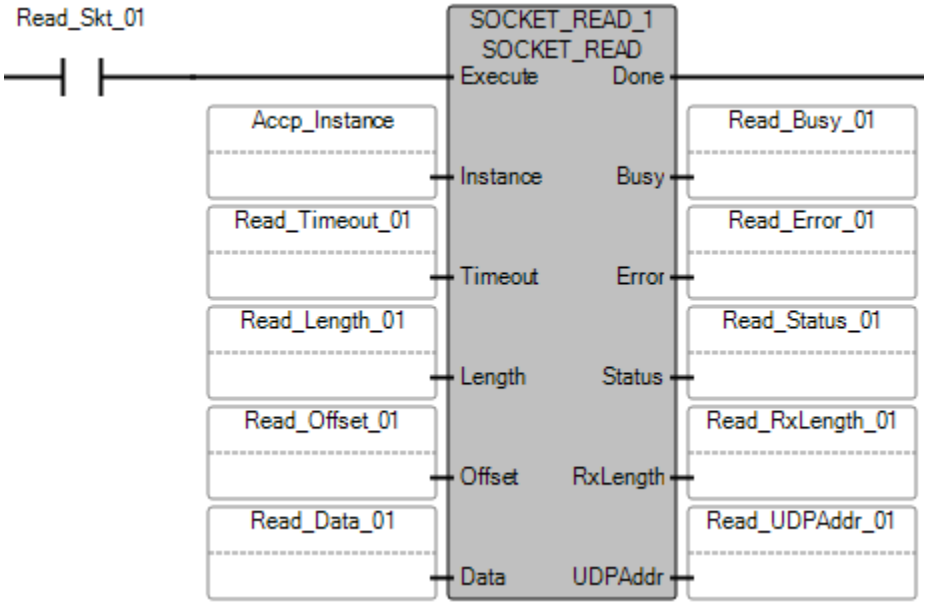
参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令。 TRUE - 检测到上升沿，启动指令，前提是上一个操作已完成。 FALSE - 不启动指令。
Instance	输入	UDINT	复制从"SOCKET_CREATE"或"SOCKET_ACCEPT"指令返回的套接字句柄。 • 对于 UDP 和 TCP 客户端套接字类型，复制从"SOCKET_CREATE"指令返回的套接字句柄。 • 对于 TCP 服务器套接字类型，复制从"SOCKET_ACCEPT"指令返回的套接字句柄。
Timeout	输入	UDINT	"SOCKET_READ"的超时。如果超时值小于最小值，指令块会返回错误。 超时范围：1000 - 86400000 毫秒 将"超时"设置为 0 以使用默认值 10000（10 秒）。
Length	输入	UINT	定义要读取的字节数。 在"RxLength"中检查实际读取的字节数。"SOCKET_READ"返回的字节数可能少于所请求的字节数。 最多支持 256 个字节。
Offset	输入	UNIT	到"Data"数组的偏移。从此位置开始读取已读取的数据。
Data	输出	USINT[1..1]	用来存储从"SOCKET_READ"读取的数据的数组。 • 数据数组大小必须 >=（"偏移"+"长度"）。 • 数据数组可以大于套接字读取"长度"。 输出与程序扫描同步更新。
Done	输出	BOOL	指示操作完成的时间。 TRUE - 操作成功完成。 FALSE - 操作正在进行或遇到错误条件。 输出与程序扫描同步更新。
Busy	输出	BOOL	TRUE - 操作未完成。 FALSE - 操作已完成。 输出与程序扫描同步更新。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。 输出与程序扫描同步更新。
Status	输出	SOCK_STATUS	使用 SOCK_STATUS 数据类型 参考页数 562 定义状态，包含 ErrorID 参考页数 562、SubErrorID 和 StatusBits 参考页数 565 信息。 输出与程序扫描同步更新。

参数	参数类型	数据类型	描述
RxLength	输出	UINT	包含接收到的数据字节数。
UDPAddr	输出	SOCKADDR_CFG 参考页数 561	发送用户数据报协议 (UDP) 数据的设备的地址。 定义 UDPAddr 192.168.2.100 和端口 12000 的示例： UDPAddr.IPAddress[0]=192 UDPAddr.IPAddress[1]=168 UDPAddr.IPAddress[2]=2 UDPAddr.IPAddress[3]=100 UDPAddr.Port = 12000 对于传输控制协议 (TCP)，此结构将不使用，并且全部包含零。TCP 连接传送远程地址信息。

SOCKET_READ 功能块图示例



SOCKET_READ 梯形图示例



SOCKET_READ 结构化文本示例

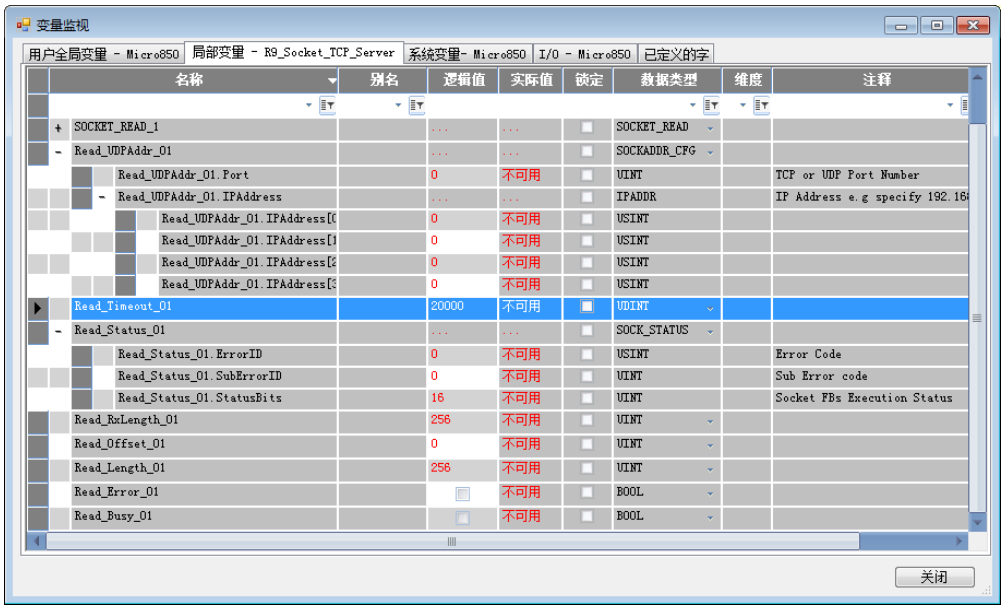
```
SOCKET_READ_1(
```

```
void SOCKET_READ_1(BOOL Execute, UDINT Instance, UDINT Timeout, UINT Length, UINT Offset, USINT[1..1] Data)
类型： SOCKET_READ, 读取连接
```

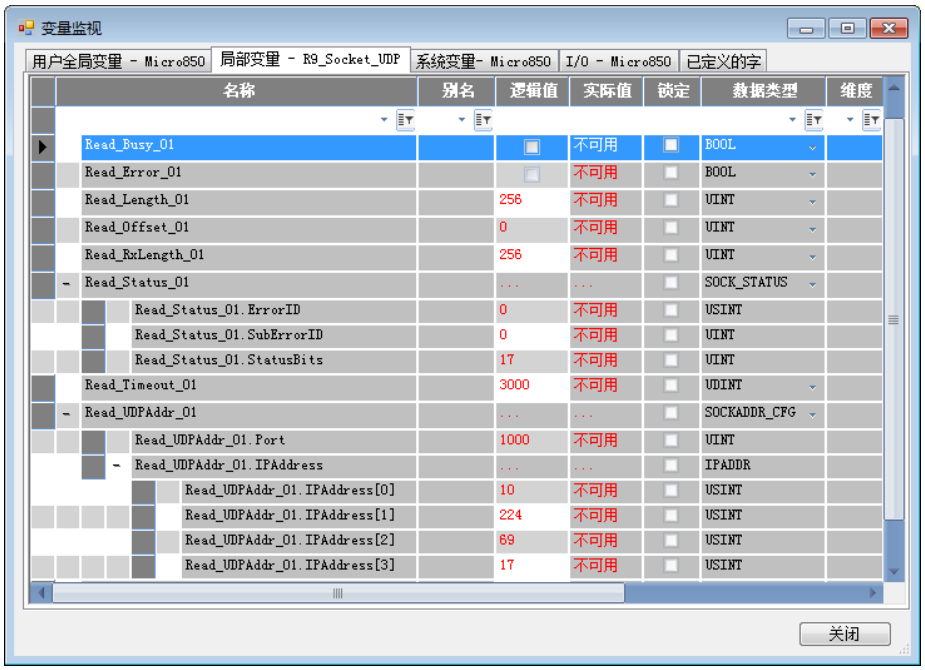
```
SOCKET_READ_1 (Execute_SOCKET_READ, Instance_SOCKET_READ, Timeout_SOCKET_READ, Length_SOCKET_READ, Offset_SOCKET_READ, Data_SOCKET_READ)
Done_SOCKET_READ := SOCKET_READ.Done;
Busy_SOCKET_READ := SOCKET_READ.Busy;
Error_SOCKET_READ := SOCKET_READ.Error;
Status_SOCKET_READ := SOCKET_READ.Status;
RxLength_SOCKET_READ := SOCKET_READ.RxLength;
UDPAddr_SOCKET_READ := SOCKET_READ.UDPAddr;
```

结果

TCP 示例：



UDP 示例：



SOCKET_WRITE

发送套接字上的数据。

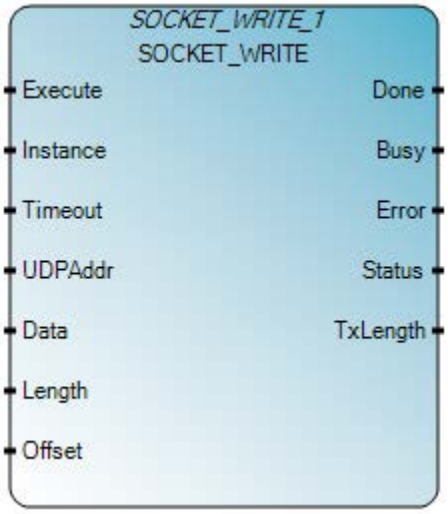
操作详细信息：

- **SOCKET_WRITE** 尝试发送 Length 参数中指定的请求数据字节数 。发送操作完成时，**SOCKET_WRITE** 返回已写入 TxLength 参数的数据字节数。
- 输出与程序扫描异步更新。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro820、Micro850 和 Micro870 控制器。对于 Micro800 Simulator，此指令可以添加到程序，但只是占位符，用于防止在控制器更改期间删除指令。

当指令应用于模拟的控制器 (2080-LC50-48QWB-SIM) 时，输出始终被复位。

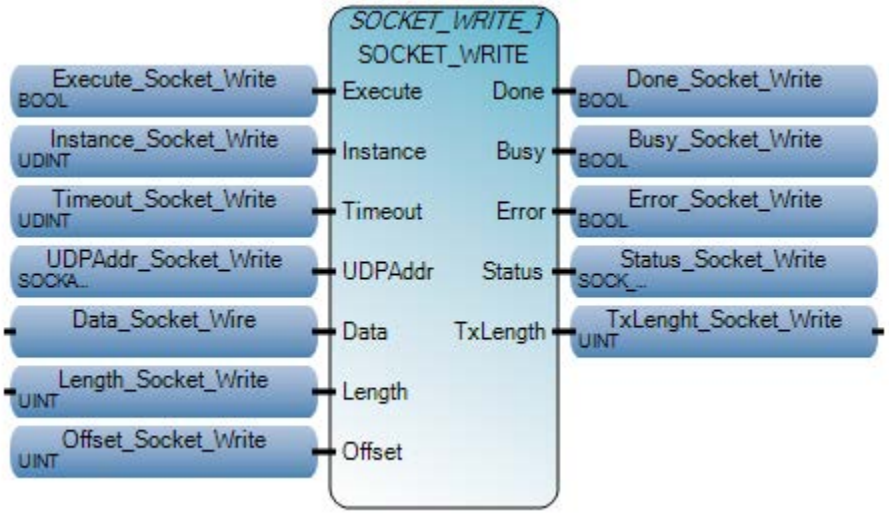


使用此表有助于确定该指令的参数值。

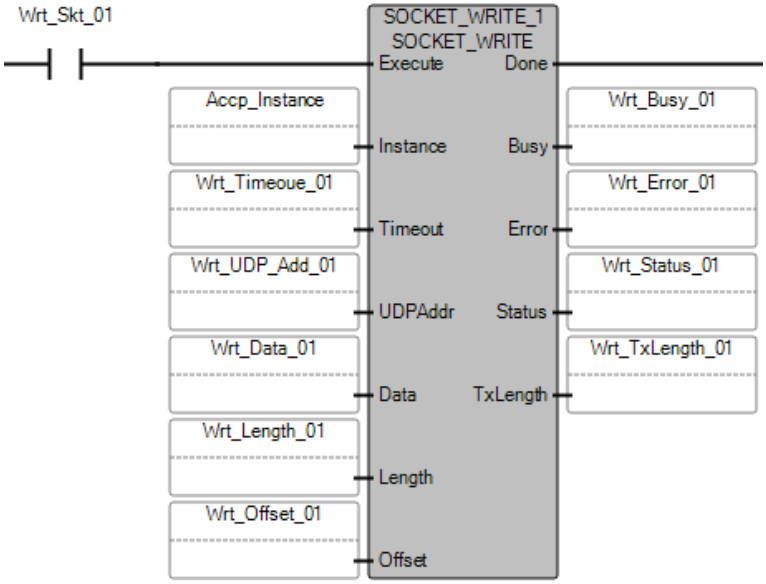
参数	参数类型	数据类型	描述
Execute	输入	BOOL	启用指令块。 TRUE - 检测到上升沿，启动指令块，前提是上一个操作已完成。 FALSE - 未检测到上升沿。
Instance	输入	UDINT	复制从“SOCKET_CREATE”或“SOCKET_ACCEPT”指令返回的套接字句柄。 • 对于 UDP 和 TCP 客户端套接字类型，复制从“SOCKET_CREATE”指令返回的套接字句柄。 • 对于 TCP - 服务器套接字类型，从 **SOCKET_ACCEPT** 指令返回的套接字句柄进行复制。
Timeout	输入	UDINT	**SOCKET_WRITE** 实例的超时。如果超时值小于最小值，指令将返回错误。 超时范围：1000 到 1800000 毫秒 将“超时”设置为 0 以使用默认值 10000（10 秒）。
UDPAddr	输入	SOCKADDR_CFG 参考页数 561	自创建套接字以来未执行 **SOCKET_OPEN** 指令的情况下要将数据写入到的 UDP 目标地址。对于 TCP，或当尚未对 UDP 执行 **SOCKET_OPEN** 指令时，此结构将不使用，并且其中应全部包含零。TCP 连接和用于 UDP 的 **SOCKET_OPEN** 指令将传输所有远程地址信息。 UDPAddr 192.168.2.100 和端口 12000 的示例： UDPAddr.IPAddress[0]=192 UDPAddr.IPAddress[1]=168

参数	参数类型	数据类型	描述
			UDPAddr.IPAddress[2]=2 UDPAddr.IPAddress[3]=100 UDPAddr.Port = 12000 使用 **SOCKADDR_CFG** 数据类型定义 UDPAddr。
Data	输入	USINT[1..1]	用于使用 **SOCKET_WRITE** 指令存储被写入相应套接字实例的数据的数组。 • 数据数组大小必须 >=（“偏移”+“长度”）。 • 数据数组可大于“SOCKET_WRITE”长度。
Length	输入	UINT	要写入的数据字节数。 最大值为 256 个字节。
Offset	输入	UNIT	到 Data 数组的偏移。从 SOCKET_WRITE 写入的数据从此位置开始。
Done	输出	BOOL	指示操作完成的时间。 TRUE - 操作成功完成。 FALSE - 操作正在进行或遇到错误条件。 输出与程序扫描同步更新。
Busy	输出	BOOL	TRUE - 操作未完成。 FALSE - 操作已完成。 输出与程序扫描同步更新。
Error	输出	BOOL	指示发生了错误。 TRUE - 检测到错误。 FALSE - 没有错误。 输出与程序扫描同步更新。
Status	输出	SOCK_STATUS	使用 SOCK_STATUS 数据类型 参考页数 562 定义状态，包含 ErrorID 参考页数 562、SubErrorID 和 StatusBits 参考页数 565 信息。 输出与程序扫描同步更新。
TxLength	输出	UNIT	写入的数据字节数。

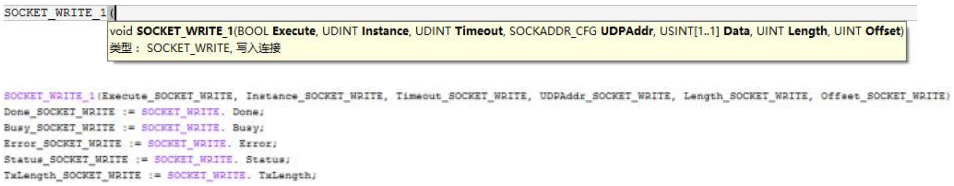
SOCKET_WRITE 功能块图示例



SOCKET_WRITE 梯形图示例

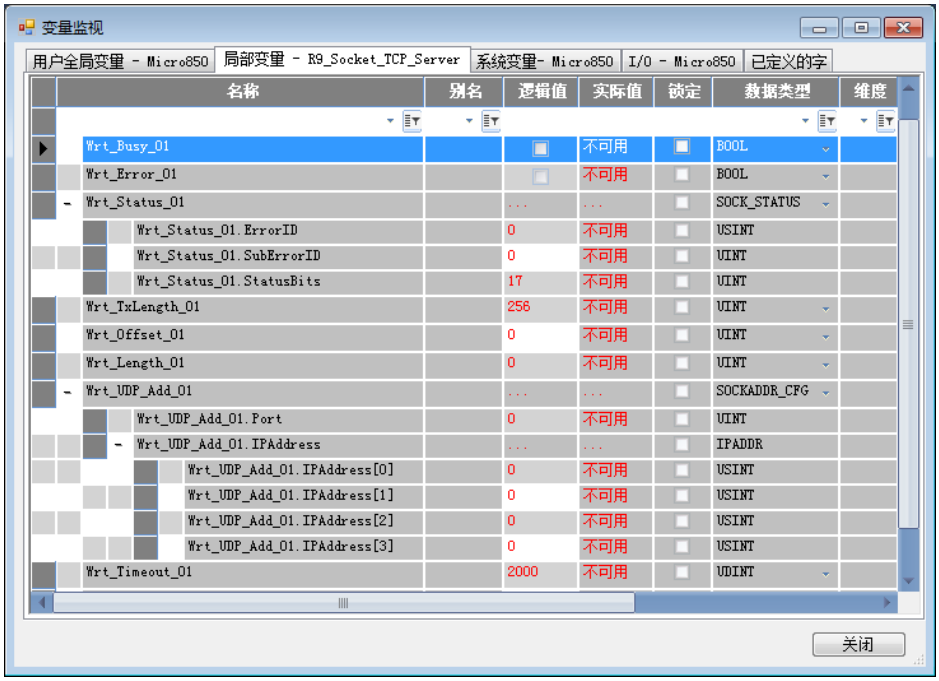


SOCKET_WRITE 结构化文本示例

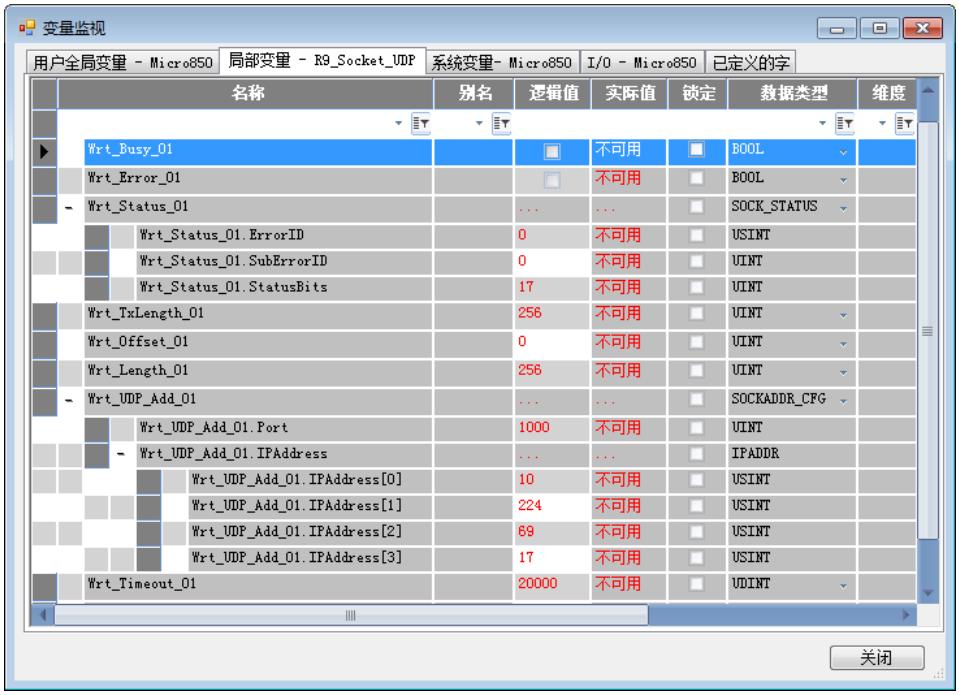


结果

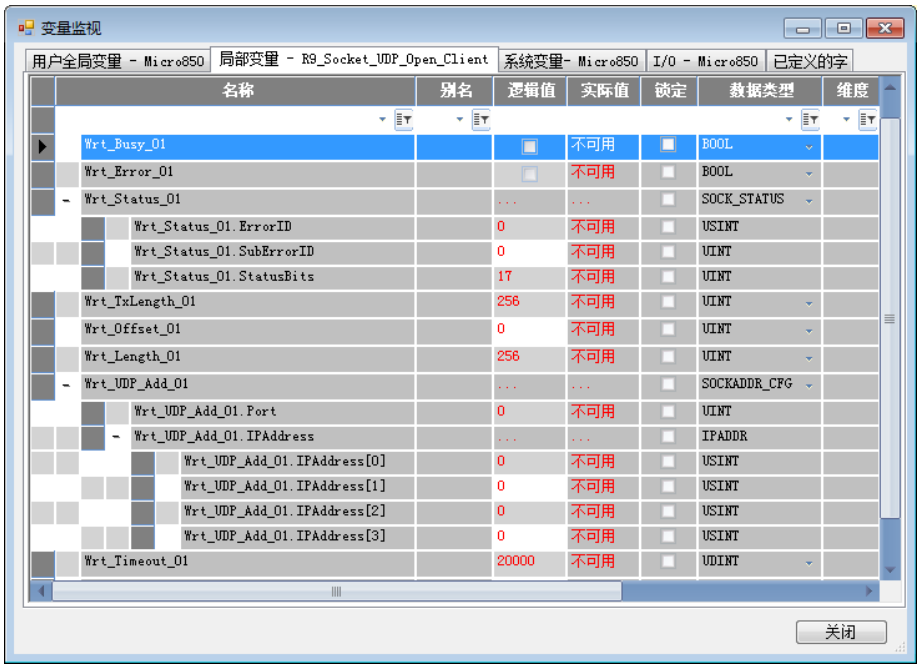
TCP 示例：



UDP（不采用 Open）示例：



UDP（采用 Open）示例：



SOCKADDR_CFG 数据类型

下表描述了 **SOCKADDR_CFG** 数据类型。

参数	数据类型	描述
Port	UINT	指定应用程序在其上进行侦听和接收的本地端口号。

参数	数据类型	描述
IP Address[4]	USINT	指定 IP 地址。 IP 地址 192.168.2.100 的示例： • IPAddress[0]=192 • IPAddress[1]=168 • IPAddress[2]=2 • IPAddress[3]=100

SOCK_STATUS 数据类型

下表描述了 **SOCK_STATUS** 数据类型。

参数	数据类型	描述
ErrorID	USINT	错误代码 参考页数 562 值。
SubErrorID	UINT	子错误代码 参考页数 562 值。
StatusBits	UINT	套接字指令的执行状态位 参考页数 565。

套接字错误代码

下表描述了套接字指令的状态错误代码：

ErrorID 代码	SubErrorID 代码	错误描述	更正措施
0		套接字指令已成功完成操作。	
1		套接字指令处于挂起状态。	
2		套接字实例不可用。	确认未删除套接字实例，或已超出超时值。
3		SOCKET_DELETEALL 操作正在进行中。	等待挂起的 SOCKET_DELETEALL 操作完成。
4	1	参数非法，通道无效。	无操作。保留供将来使用。
4	2	参数非法，套接字 IP 地址无效。 • 发生错误。SOCKET_CREATE 包含具有任何非零值的 SockAddr。 • 使用无效的目标 IP 地址执行 SOCKET_OPEN 或 SOCKET_WRITE。无效目标 IP 地址包括： • 自 IP 地址 • 0.0.0.0 • 回路地址 (127.x.x.x) • D 类多播地址 (224.x.x.x) • 本地链接地址 (169.254.x.x) • 广播地址（仅适用于 TCP 套接字实例）	对有效 IP 地址的更改。
4	3	参数非法，套接字端口地址无效。 在 SOCKET_CREATE、SOCKET_OPEN 或 SOCKET_WRITE 中指定以下端口时发生错误： • TCP 端口 • 44818 - 以太网/IP • 502 -ModbusTCP • 67 - DHCP 服务器 • 68 - DHCP 客户端 • 0 - 端口无效	更改端口地址。

ErrorID 代码	SubErrorID 代码	错误描述	更正措施
		• UDP 端口 • 2222 - 以太网/IP • 67 - DHCP 服务器 • 68 - DHCP 客户端 • 0 - 端口无效	
4	4	参数非法，套接字类型无效。 有效的套接字类型包括： • TCP - 1 • UDP - 2	更改套接字类型。
4	5	参数非法，套接字超时值无效。 有效超时值包括： • SOCKET_CREATE、SOCKET_ACCEPT 和 SOCKET_READ： • 0（默认值）或介于 1000 和 86400000ms 之间的任何值。 • SOCKET_OPEN 和 SOCKET_WRITE： • 0（默认值）或介于 1000 和 86400000ms 之间的任何值。	将超时值更改为有效值。
4	6	参数非法，套接字实例无效。 • TCP 服务器套接字类型： • 对 SOCKET_WRITE、SOCKET_READ、SOCKET_DELETE 和 SOCKET_INFO 使用在 SOCKET_ACCEPT 中返回的套接字实例。 • UDP 套接字和 TCP 客户端套接字类型： • 对 SOCKET_OPEN、SOCKET_WRITE、SOCKET_READ、SOCKET_DELETE 和 SOCKET_INFO 使用在 SOCKET_CREATE 中返回的套接字实例。	在成功执行后使用在 SOCKET_CREATE 和 SOCKET_ACCEPT 指令中返回的实例编号。
4	7	参数非法，数组长度无效。	增大用于包含套接字读取和写入数据的数组的大小。
4	8	参数非法，数组维度无效。	使用单维数组包含套接字读取和写入数据。
5		要取消操作的套接字请求。 下列情况下发生错误： • 以太网链接已禁用，或以太网电缆已断开 • 当 SOCKET_OPEN、SOCKET_ACCEPT、SOCKET_READ 或 SOCKET_WRITE 操作正在进行时执行 SOCKET_DELETE操作。 • 当 SOCKET_OPEN、SOCKET_ACCEPT、SOCKET_READ 或 SOCKET_WRITE 操作正在进行时执行"运行模式更改"操作。 • 当 SOCKET_OPEN、SOCKET_ACCEPT、SOCKET_READ 或 SOCKET_WRITE 操作正在进行时检测到 IP 地址冲突 。	基于套接字类型重启套接字操作。请参阅相应状态机以重启套接字操作。
6	1	套接字序列非法，"套接字打开"操作正在进行或已连接。 下列情况下发生错误： • SOCKET_OPEN 正在进行（BUSY 状态），并且用户对相同套接字实例执行另一个 SOCKET_OPEN 实例。 • SOCKET_OPEN 正在进行（BUSY 状态），并且用户对相同套接字实例执行 SOCKET_ACCEPT。	对相应套接字实例使用单个 SOCKET_OPEN 执行。不对同一个套接字实例执行 SOCKET_ACCEPT 操作。
6	2	套接字序列非法，套接字接受操作正在进行或已连接。 下列情况下发生错误：	对相应套接字实例使用单个 SOCKET_ACCEPT 执行。不对同一个套接字实例执行 SOCKET_OPEN 操作。

ErrorID 代码	SubErrorID 代码	错误描述	更正措施
		• SOCKET_ACCEPT 正在进行（BUSY 状态），并且用户对相同套接字实例执行另一个 SOCKET_ACCEPT 实例。 • SOCKET_ACCEPT 正在进行（BUSY 状态），并且用户对相同套接字实例执行 SOCKET_OPEN。	
6	3	套接字序列非法，套接字已配置为 TCP 客户端。	
6	4	套接字序列非法，套接字已配置为 TCP 服务器。	
6	5	套接字序列非法，套接字已连接。	确保套接字未与远程设备连接，然后再次执行 SOCKET_OPEN 或 SOCKET_ACCEPT。
6	6	套接字序列非法，套接字已配置为 UDP。	确保 UDP 套接字实例未与 SOCKET_ACCEPT 一起使用。
6	7	套接字序列非法，套接字未连接。	确认套接字已与目标连接。
7		套接字实例超时。	为 SOCKET_ACCEPT、SOCKET_OPEN、SOCKET_READ 和 SOCKET_WRITE 相应配置超时值。
8		套接字模块未初始化。	确保以太网链接已启用或控制器的以太网电缆已连接至网络，或解决控制器 IP 地址冲突。
9	1	套接字指令致命错误，缺少套接字实例。	
9	2	套接字指令致命错误，套接字实例无效。	
9	3	套接字指令致命错误，锁定套接字实例无效。	
9	4	套接字指令致命错误，套接字类型无效。	
9	5	套接字指令致命错误，缺少取消句柄。	
10	1	套接字背景处理错误，地址已被使用。	
10	2	套接字背景处理错误，UDP 接收到大数据包，且该数据包大于套接字读取长度。	确保 SOCKET_READ 长度等于或大于所接收的数据包大小。建议使用最大读取大小。SOCKET_READ 的最大大小为 256。
10	3	套接字背景处理错误，TCP 接收到大数据包。所接收的数据包大于套接字读取长度。	确保 SOCKET_READ 长度等于或大于所接收的数据包大小。由于客户端可发送的接收窗口的最大大小为 256，建议使用 256。
10	4	套接字背景处理错误，已接收到 RST 或从远程设备断开连接。	重启控制器 TCP 客户端或服务器，如状态图中所示。验证已发送 RST 的目标设备，并确保目标已重启并具有正确状态。
10	5	套接字背景处理错误，UDP 数据包从不同的设备接收到丢弃数据包。保留供将来使用。	
10	6	套接字背景处理错误，队列已满。 下列情况下发生错误： • SOCKET_READ 或 SOCKET_WRITE 在同一扫描中对同一个套接字实例执行四个指令。 • 当四个 SOCKET_READ 指令对同一个套接字实例处于 BUSY 状态时执行 SOCKET_READ。	对相应的套接字实例，等待 SOCKET_READ 和 SOCKET_WRITE 队列的可用性。
10	7	套接字背景处理错误，SOCKET_READ 参数更改（长度、偏移、数据数组大小和数据数组变量）。	当 SOCKET_READ 状态为 BUSY 时，不修改输入参数。
11	1	套接字背景处理致命错误，缺少数据指针。	
11	2	套接字背景处理致命错误，缺少会话指针。	
11	3	套接字背景处理致命错误，TCP 或 UDP 套接字指针无效。	
11	4	套接字背景处理致命错误，套接字实例类型无效。	
11	5	套接字背景处理致命错误，缺少套接字实例。	

ErrorID 代码	SubErrorID 代码	错误描述	更正措施
11	6	套接字背景处理致命错误，套接字实例无效。	
11	7	套接字背景处理致命错误，套接字状态无效。	
11	8	套接字背景处理致命错误，套接字类型无效。	
11	9	套接字背景处理致命错误，TCP 删除故障。	
11	10	套接字背景处理致命错误，UDP 删除故障。	
128	1	没有可用于断开连接消息的数据包。	
128	2	空间不足以前置 TCP 报头。	
128	3	数据包附加指针无效。	
128	7	套接字指针无效。	
128	10	服务类型、片段或存在时间选项无效。	
128	17	此服务的调用无效。	
128	18	数据包无效。	
128	20	此组件未启用。	
128	21	此组件已启用。	
128	22	没有为指定端口启用侦听。	
128	26	所请求的挂起已被中止。	
128	33	服务器 IP 地址无效。	
128	34	此套接字已绑定到另一个端口。	
128	35	端口已绑定到不同的套接字。	
128	36	套接字未绑定。	
128	38	套接字在挂起以等待接收数据包时未绑定。	
128	39	未创建套接字。	
128	51	无其他可用的侦听请求结构。	
128	52	已存在此端口的活动侦听请求。	
128	53	套接字未处于关闭状态。	
128	54	所提供的服务器套接字未处于侦听状态。	
128	55	未指定等待时间，正在进行连接尝试。	
128	56	连接失败。	
128	57	请求大于接收方的公布窗口大小（单位为字节）。	
128	64	另一个线程被挂起。仅允许一个线程。	
128	65	断开连接操作未能在超时时间内完成。	
128	66	套接字已绑定。	
128	69	无可用端口。	
128	70	无效端口。	
128	71	已存在对应于此端口的有效套接字指针，或指定端口没有活动的侦听请求。	
128	72	与 NX_SUCCESS 相同，除非已在此调用期间处理了排队的连接请求。	
128	73	TCP 传输队列超出错误。	

套接字指令状态位

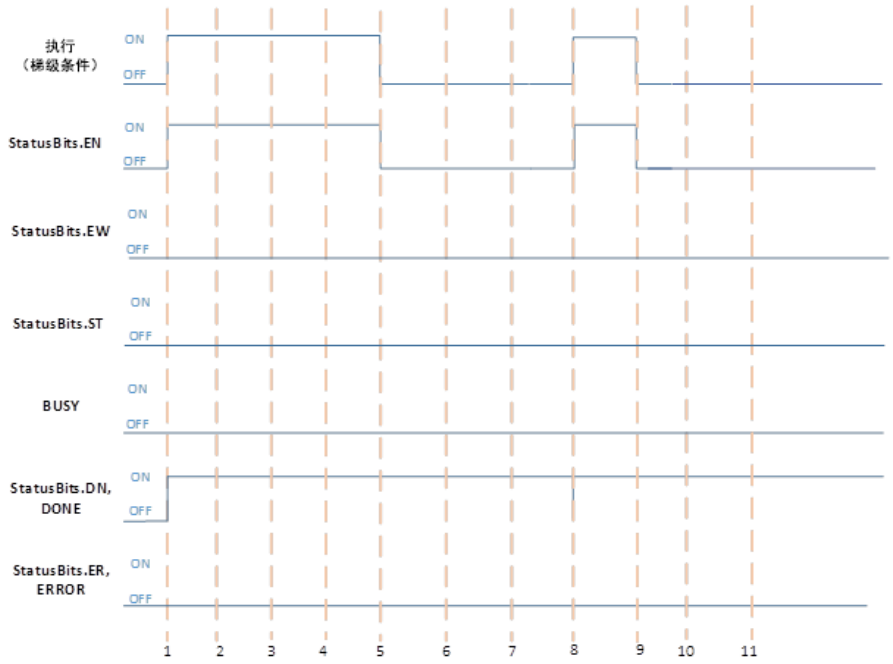
下表描述了套接字指令 参考页数 533 的执行状态位。

位数	名称	描述
0	EN - 启用位	当指令由于“False 至 True”转换而启用但未完成或出错时，EN 位被置位。

位数	名称	描述
1	EW - 启用等待队列位	当控制器检测到套接字指令请求已进入队列时，EW 位被置位。 当 ST 位被置位时，控制器将复位 EW 位。
2	ST - 启动位	当正在执行已排队的指令时，ST 位被置位。 当 DN 或 ER 被置位时，ST 位被复位。
3	ER - 错误位	表示在执行指令期间发生错误。当 run-condition-in 下次从 False 变为 True 时，ER 位被复位。
4	DN - 完成位	当套接字指令成功完成时，DN 位被置位。当 run-condition-in 下次从 False 变为 True ，DN 位被复位。

套接字指令时间图

当进程为立即进程时套接字指令 参考页数 533 执行成功



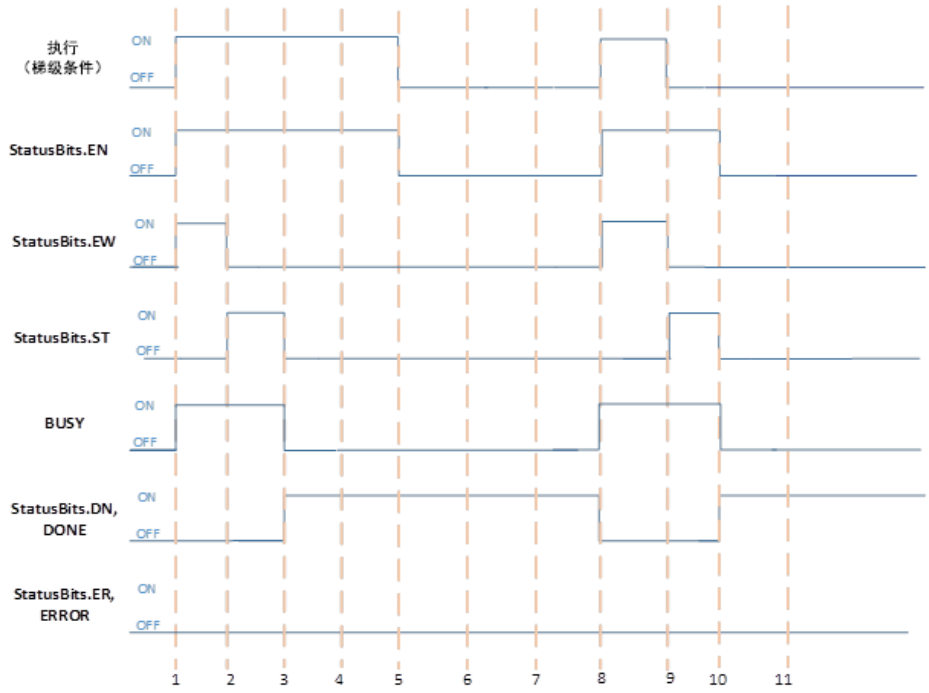
条件 A：梯级条件在指令执行期间为 TRUE

梯级条件	描述
1	在下列情况下梯级条件变为 TRUE： • 已启用套接字指令执行。 • EN 位被置位，所有其他位被清除。 • 套接字指令成功完成执行。 • DN 位和 DONE 的输出被置位。
2，3，4	梯级条件未发生更改。
5	当 EN 位被清除时梯级条件变为 FALSE。
6,7	梯级条件未发生更改。

条件 B：在指令执行期间梯级条件为 FALSE

梯级条件	描述
8	在下列情况下梯级条件变为 TRUE： • 已启用套接字指令执行。 • 套接字指令成功完成执行。 • EN 位被置位，所有其他位被清除。 • DN 位和 DONE 的输出被置位。
9	在下列情况下梯级条件变为 FALSE： • EN 位已清除。
10,11	梯级条件未发生更改。

当进程为非立即进程时套接字指令执行成功



条件 A：梯级条件在指令执行期间为 TRUE

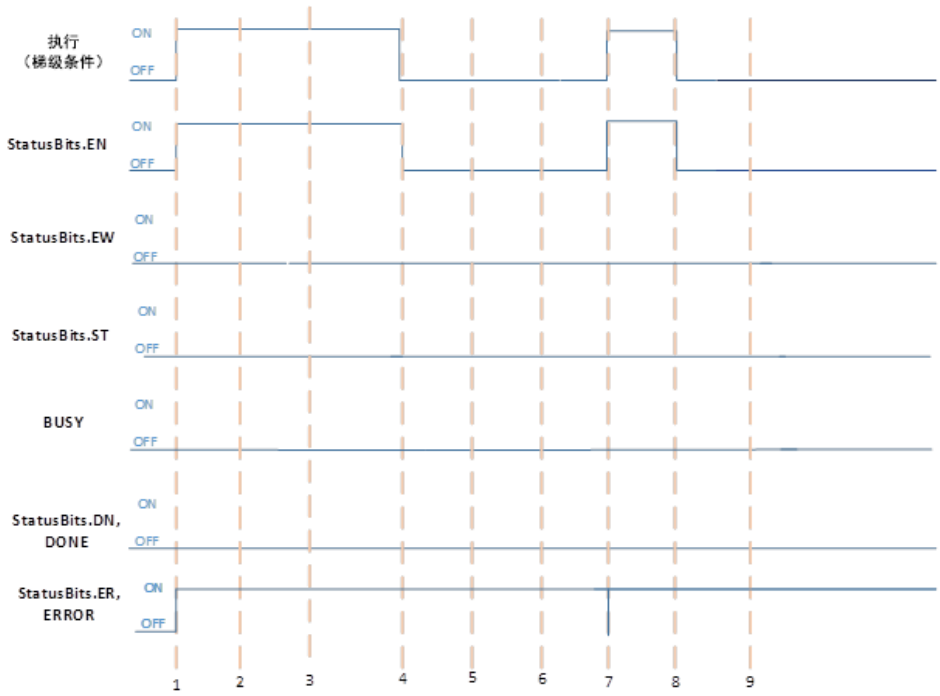
梯级条件	描述
1	在下列情况下梯级条件变为 TRUE： • 已启用套接字指令执行。 • EN 位被置位，所有其他位被清除。 • 套接字指令执行背景处理。 • EW 位和 BUSY 的输出已被置位。 • 用于进行背景处理的锁定套接字指令输入参数。
2	在下列情况下套接字指令开始执行： • EW 位被清除。 • ST 位被置位。

梯级条件	描述
3	在下列情况下套接字指令成功完成执行： • ST 位和 BUSY 的输出被清除。 • DN 位和 DONE 的输出被置位。
4	梯级条件未发生更改。
5	在下列情况下梯级条件变为 FALSE： • EN 位已清除。
6,7	梯级条件未发生更改。

条件 B：在指令执行期间梯级条件变为 FALSE

梯级条件	描述
8	在下列情况下梯级条件变为 TRUE： • 已启用套接字指令执行。 • EN 位被置位，所有其他位被清除。 • 套接字指令发送以进行背景处理。 • EW 位和 BUSY 的输出已被置位。 • 用于进行背景处理的锁定套接字指令输入参数。
9	在下列情况下套接字指令开始执行： • EW 位被清除。 • ST 位被置位。 • 梯级条件变为 FALSE。
10	在下列情况下套接字指令成功完成执行： • ST 位和 BUSY 的输出被清除。 • DN 位和 DONE 的输出被置位。 • 梯级条件为 FALSE。 • EN 位已清除。

当 EN 为 TRUE 并且 EW 和 ST 为 FALSE 时，套接字指令失败



条件 A：梯级条件在指令执行期间为 TRUE

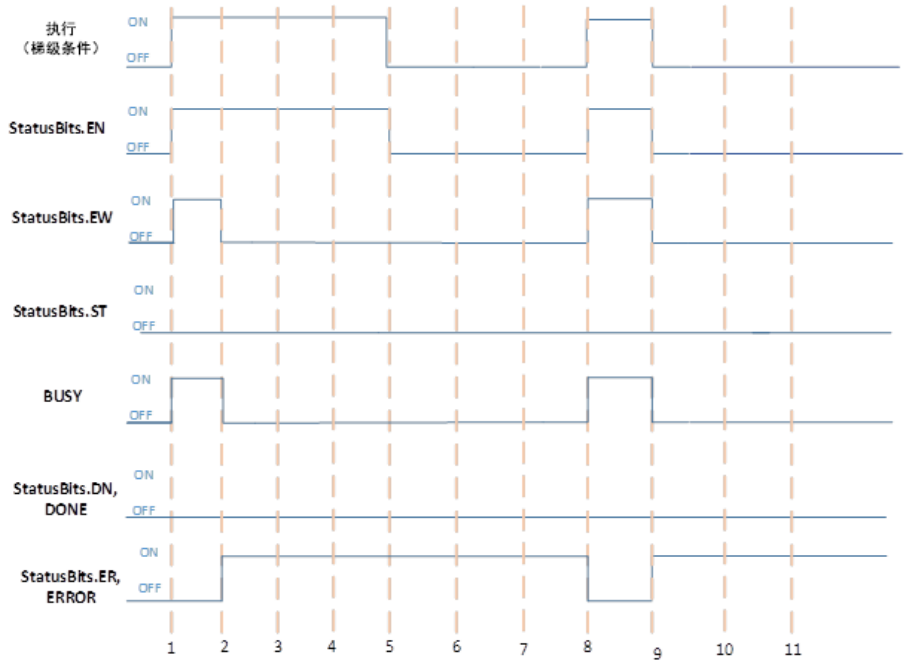
梯级条件	描述
1	在下列情况下梯级条件变为 TRUE： • 已启用套接字指令执行。 • EN 位被置位，所有其他位被清除。 • 套接字指令完成执行但出错。 • ER 位和 error 的输出被置位。
2,3	梯级条件未发生更改。
4	在下列情况下梯级条件变为 FALSE： • EN 位已清除。
5,6	梯级条件未发生更改。

条件 B：在指令执行期间梯级条件变为 FALSE

梯级条件	描述
7	在下列情况下梯级条件变为 TRUE： • 已启用套接字指令执行。 • EN 位被置位，所有其他位被清除。 • 套接字指令完成执行但出错。 • ER 位和 error 的输出被置位。

梯级条件	描述
8	在下列情况下梯级条件变为 FALSE： • EN 位已清除。
9	梯级条件未发生更改。

当 EW 为 TRUE 并且指令进程为非立即进程时，套接字指令失败



条件 A：梯级条件在指令执行期间为 TRUE

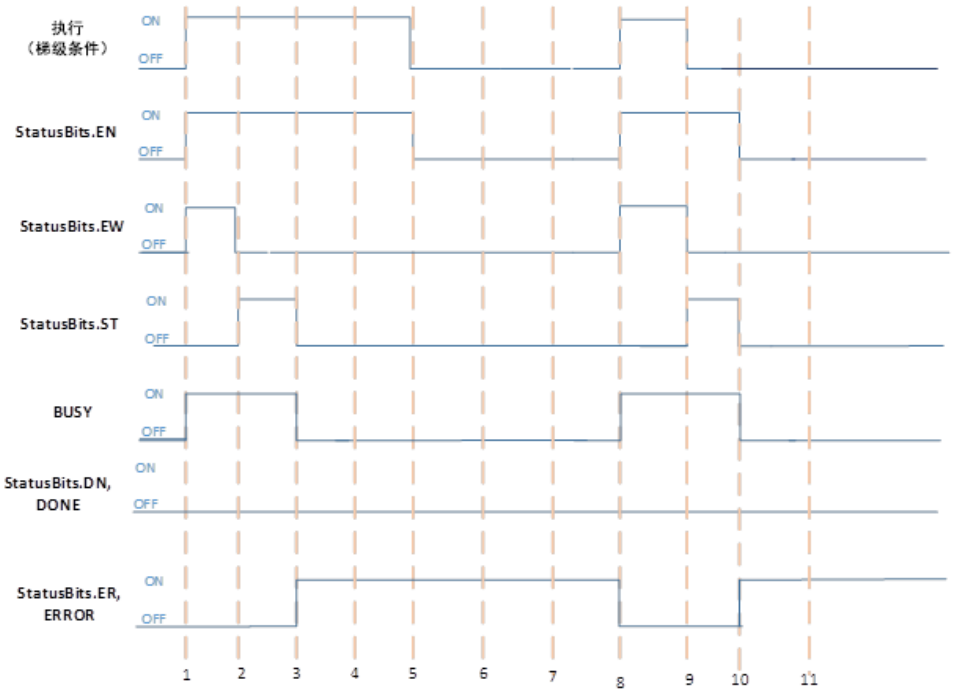
梯级条件	描述
1	在下列情况下梯级条件变为 TRUE： • 已启用套接字指令执行。 • EN 位被置位，所有其他位被清除。 • 套接字指令发送以进行背景处理。 • EW 位和 BUSY 的输出已被置位。 • 用于进行背景处理的锁定套接字指令输入参数。
2	在下列情况下套接字指令完成执行但出错： • EW 位和 BUSY 的输出被清除。 • ER 位和 error 的输出被置位。
3,4	梯级条件未发生更改。
5	在下列情况下梯级条件变为 FALSE： • EN 位已清除。

梯级条件	描述
6,7	梯级条件未发生更改。

条件 B：在指令执行期间梯级条件变为 FALSE

梯级条件	描述
8	在下列情况下梯级条件变为 TRUE： • 已启用套接字指令执行。 • EN 位被置位，所有其他位被清除。 • 套接字指令发送以进行背景处理。 • EW 位和 BUSY 的输出已被置位。 • 用于进行背景处理的锁定套接字指令输入参数。
9	在下列情况下套接字指令完成执行但出错： • EW 位和 BUSY 的输出被清除。 • ER 位和 error 的输出被置位。 • 梯级条件变为 FALSE。 • EN 位已清除。
10,11	梯级条件未发生更改。

当 ST 为 TRUE 并且指令进程为非立即进程时，套接字指令失败



条件 A：梯级条件在指令执行期间为 TRUE

梯级条件	描述
1	在下列情况下梯级条件变为 TRUE： • 已启用套接字指令执行。 • EN 位被置位，所有其他位被清除。 • 套接字指令发送以进行背景处理。 • EW 位的输出和 BUSY 输出已被置位。 • 用于进行背景处理的锁定套接字指令输入参数。
2	在下列情况下套接字指令开始执行： • EW 位被清除 • ST 位被置位。
3	在下列情况下套接字指令完成执行但出错： • ST 位的输出和 BUSY 输出被清除。 • ER 位和 error 的输出被置位。
4	梯级条件未发生更改。
5	在下列情况下梯级条件变为 FALSE： • EN 位已清除。
6,7	梯级条件未发生更改。

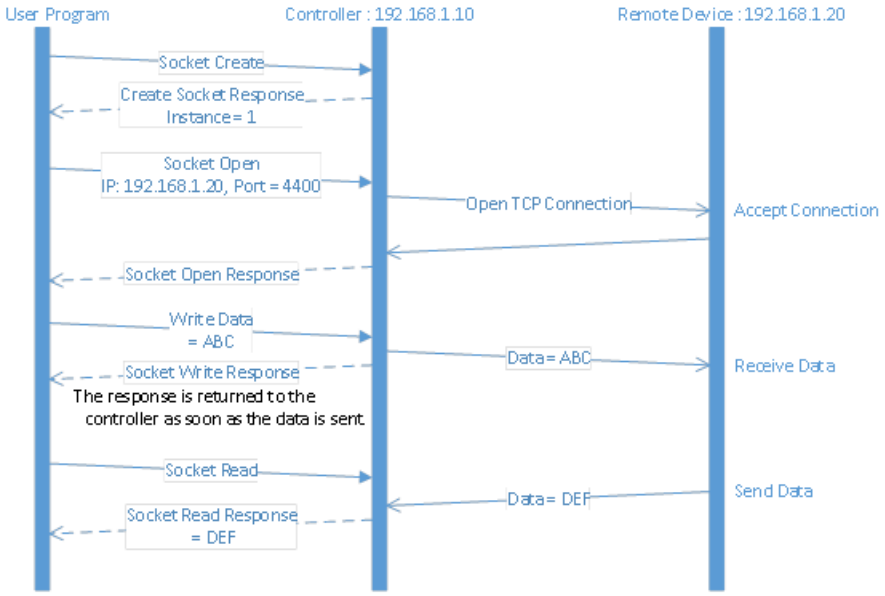
条件 B：在指令执行期间梯级条件变为 FALSE

梯级条件	描述
8	在下列情况下梯级条件变为 TRUE： • 已启用套接字指令执行。 • EN 位被置位，所有其他位被清除。 • 套接字指令发送以进行背景处理。 • EW 位和 BUSY 的输出已被置位。 • 用于进行背景处理的锁定套接字指令输入参数。
9	在下列情况下套接字指令开始执行： • EW 位被清除，并且 ST 位被置位。
10	在下列情况下套接字指令完成执行但出错： • ST 位和 BUSY 的输出被清除。 • ER 位和 error 的输出被置位。 • 梯级条件变为 FALSE。 • EN 位已清除。
11	梯级条件未发生更改。

套接字指令处理图

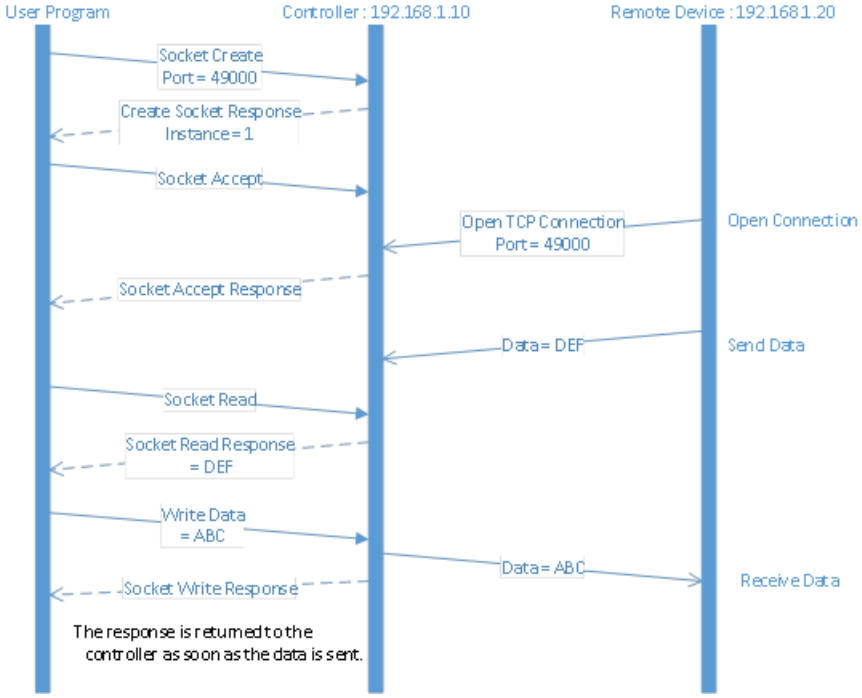
对应于 TCP 客户端的处理

下图显示了套接字界面处理的典型序列(使用控制器作为 TCP 客户端)。



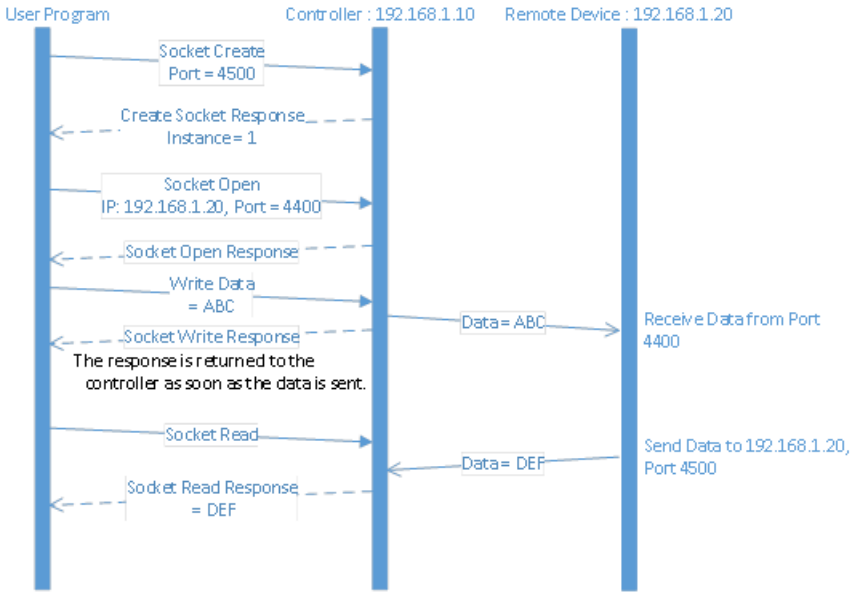
对应于 TCP 服务器的处理

下图显示了套接字界面处理的典型序列(使用控制器作为 TCP 服务器)。



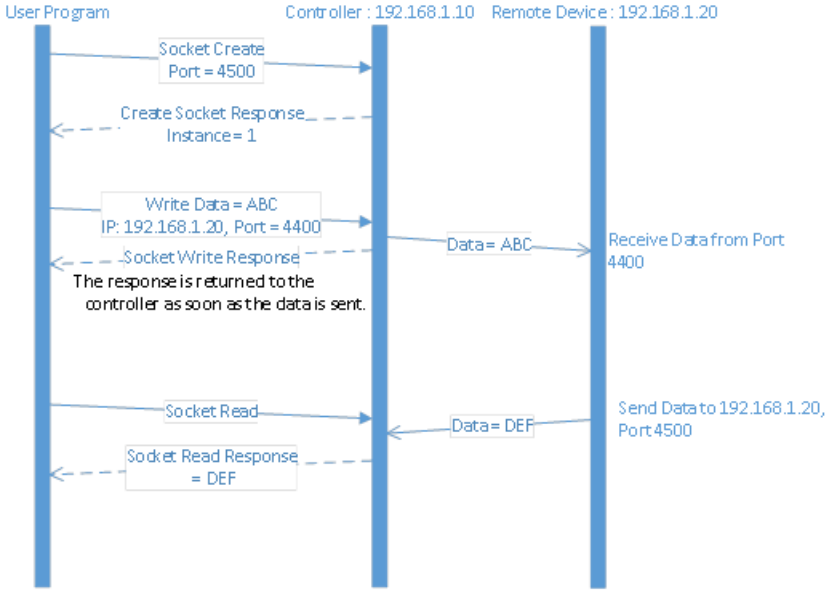
对应于采用 Open Connection 的 UDP 的处理

下图显示了 UDP 通信的套接字界面处理的典型序列（使用 Open Connection 服务指定目标地址时）。



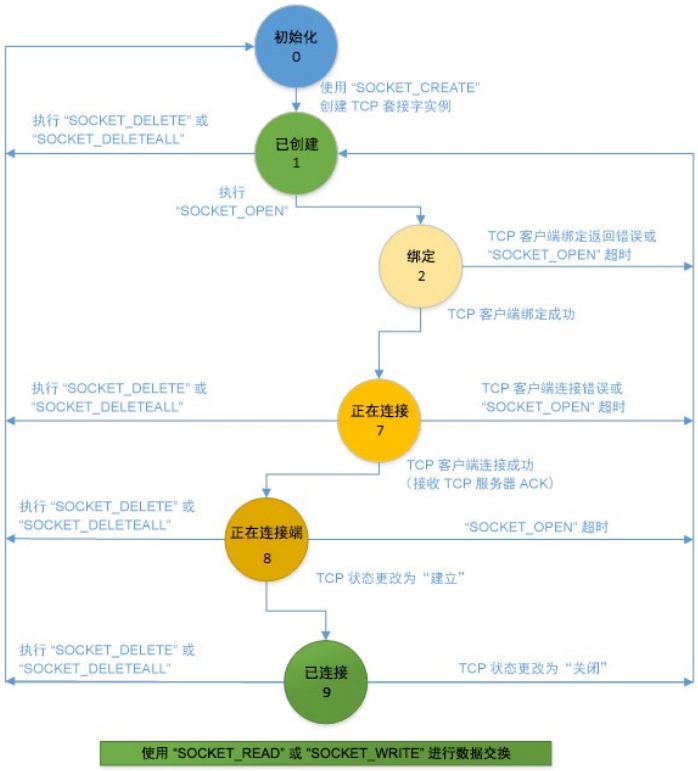
对应于未采用 Open Connection 的 UDP 的处理

下图显示了 UDP 通信的套接字界面处理的典型序列（未使用 Open Connection 服务指定目标地址时）。在此情况下，控制器为每个数据报指定目标，并接收发送方的地址及其接收的每个数据报。

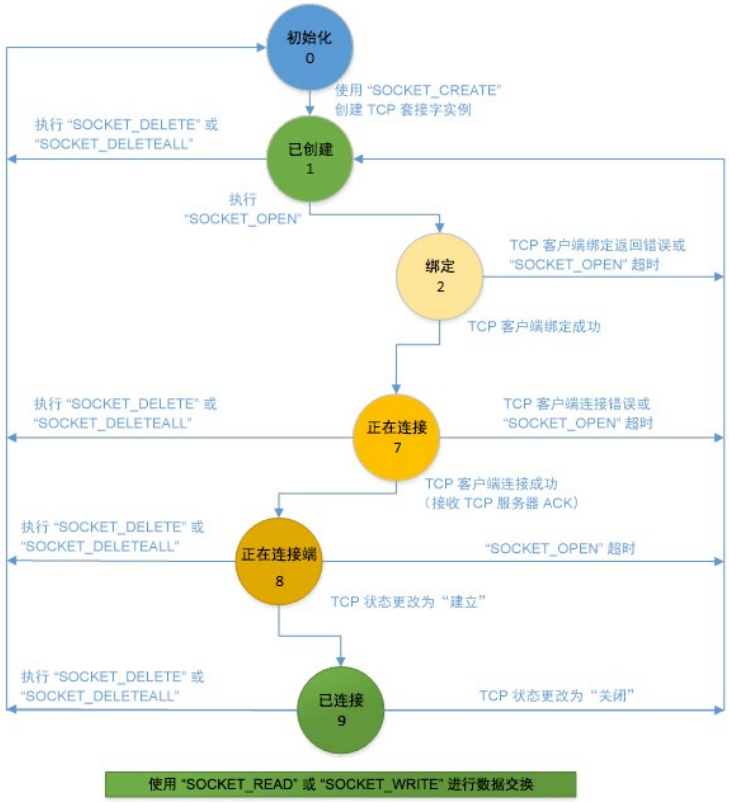


TCP 的状态机图

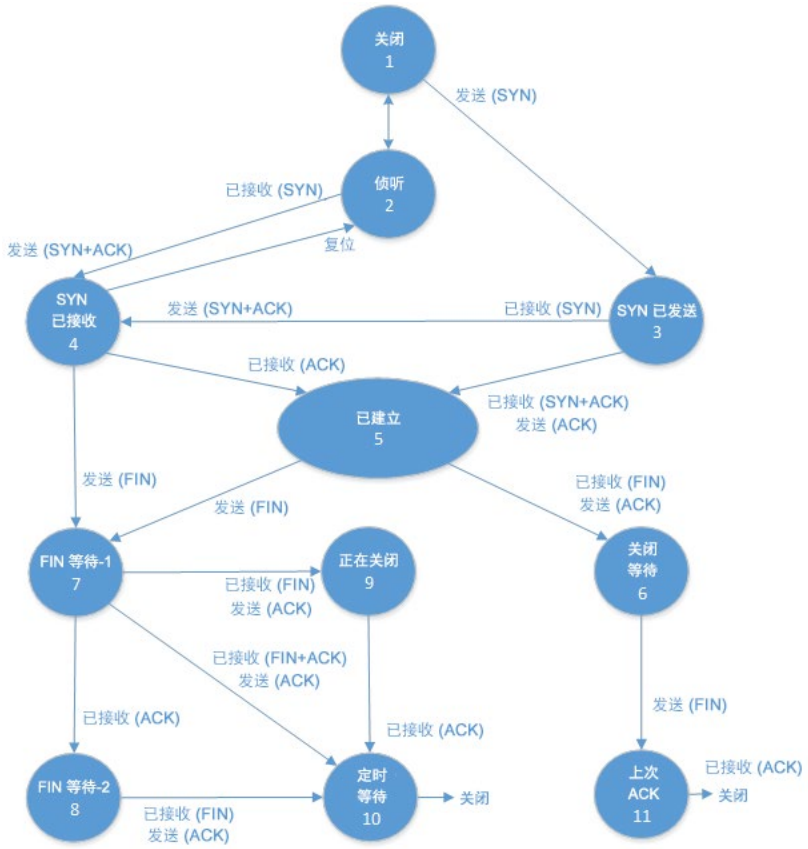
使用套接字指令 参考页数 533 的 TCP 客户端



使用套接字指令的 TCP 服务器

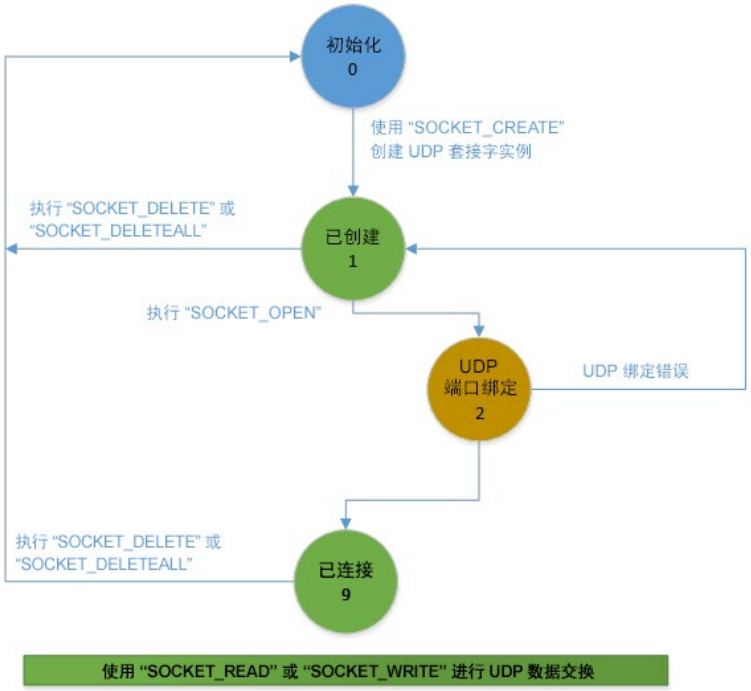


TCP 状态图

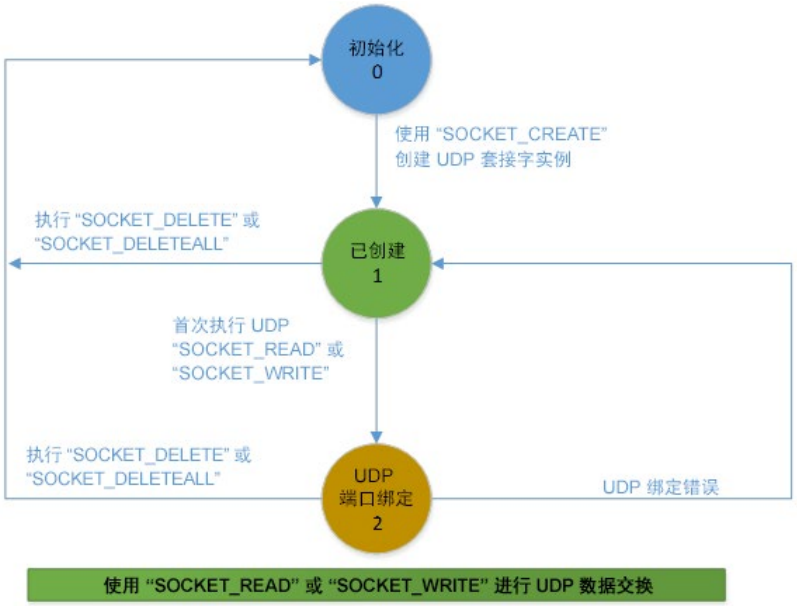


UDP 的状态机图

使用 SOCKET_OPEN 指令的 UDP 数据报



不使用 SOCKET_OPEN 指令的 UDP 数据报



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。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。

指令	描述
ASCII 参考页数 579	返回字符串形式的字符 ASCII 代码。字符 -> ASCII 代码。
CHAR 参考页数 580	返回 ASCII 代码的一个字符字符串。ASCII 代码 -> 字符。
DELETE 参考页数 582	从字符串中删除字符。
FIND 参考页数 584	在字符串中定位并提供子字符串的位置。
INSERT 参考页数 586	在字符串中用户指定的位置插入子字符串。
LEFT 参考页数 588	从字符串左侧提取字符。
MID 参考页数 589	从字符串中间提取字符。
MLEN 参考页数 591	计算字符串的长度。
REPLACE 参考页数 595	将字符串的一部分替换为新的字符集。
RIGHT 参考页数 593	从字符串右侧提取字符。

ASCII

以字符串形式返回字符的 ASCII 代码。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



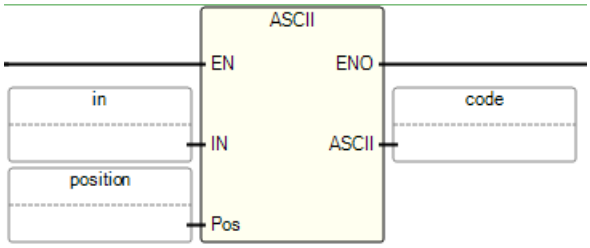
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 显示字符的 ASCII 代码。 FALSE - 不执行显示操作。 适用于梯形图编程。
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 ASCII 代码（位于集合 [0..255] 内）生成 0。
ENO	输出	BOOL	启用“输出”。 适用于梯形图编程。

ASCII 功能块图示例



ASCII 梯形图示例



ASCII 结构化文本示例

```
1  position := 1;
2  code := ASCII(in, position);
```

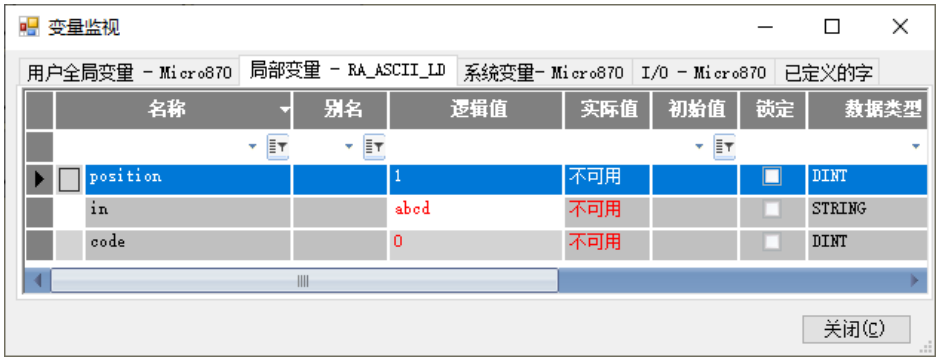
```
ASCII(
    DINT ASCII(STRING IN, DINT Pos)
    字符 -> ASCII 代码
```

(* 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



CHAR（字符串字符的 ASCII 代码）

返回 ASCII 代码的一个字符字符串。ASCII 代码 -> 字符。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



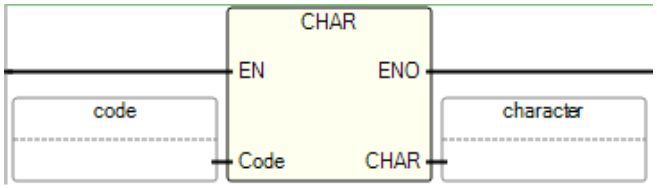
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 提供由单个字符组成的字符串。 FALSE - 不执行任何操作。 适用于梯形图编程。
Code	输入	DINT	ASCII 代码，位于集合 [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"。 适用于梯形图编程。

CHAR 功能块图示例



CHAR 梯形图示例



CHAR 结构化文本示例

```
1  code := 97;
2  character := CHAR(code);
```



(* 与之等效的 ST: *)

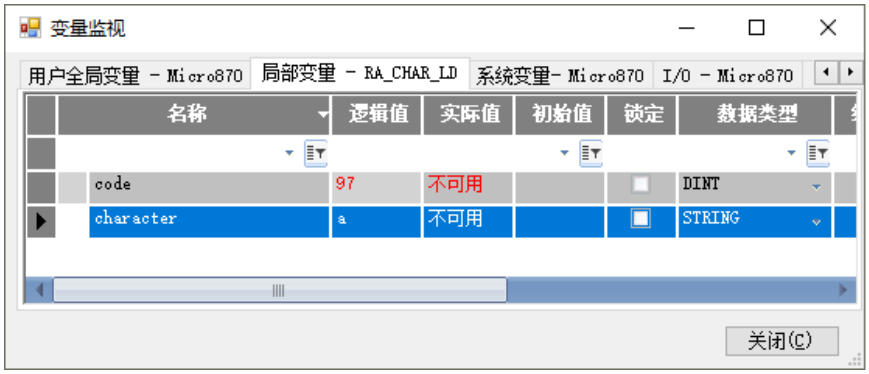
Display := CHAR (value + 48);

(* 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



DELETE（删除子字符串）

从字符串中删除字符。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



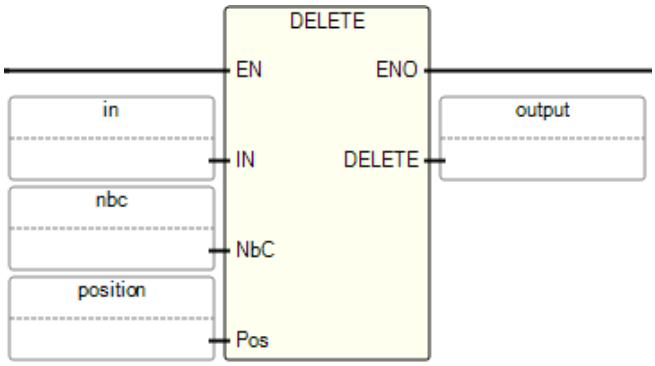
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 删除字符串的指定部分。 FALSE - 不执行任何操作。 适用于梯形图编程。
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	输出为： • 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”。 适用于梯形图编程。

DELETE 功能块图示例



DELETE 梯形图示例



DELETE 结构化文本示例




```
1  nbc := 3;
2  position := 2;
3  output := DELETE(in, nbc, position);
```

(* 与之等效的 ST: *)

complete_string := INSERT ('参考页数 ', 'EFGH', 5); (* complete_string 为"参考页数 EFGH" *)

sub_string := DELETE (complete_string, 4, 3); (* sub_string 为"ABGH " *)

结果



FIND（查找子字符串）

在字符串中定位并提供子字符串的位置。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



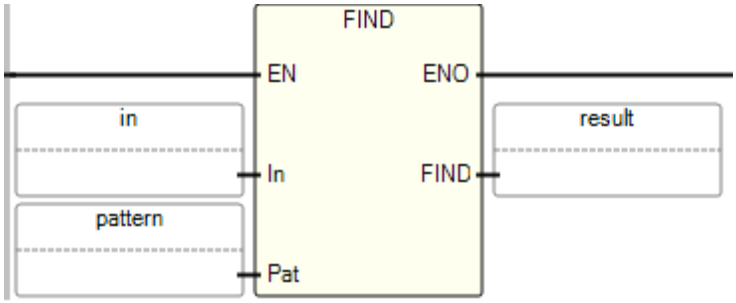
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 查找在字符串中的位置。 FALSE - 不执行查找操作。 适用于梯形图编程。
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	输出为： • 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） 此指令区分大小写。
ENO	输出	BOOL	启用"输出"。 适用于梯形图编程。

FIND 功能块图示例



FIND 梯形图示例



FIND 结构化文本示例




```
1  result := FIND(in, pattern);
```

(* 与之等效的 ST： *)

complete_string := '参考页数' + 'EFGH'; (* complete_string 为“参考页数 EFGH” *)

found := FIND (complete_string, 'CDEF'); (* found 为 3 *)

结果



INSERT（插入字符串）

在字符串中用户指定的位置插入子字符串。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

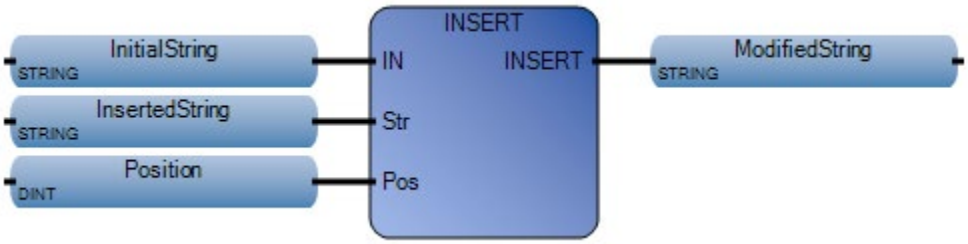
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



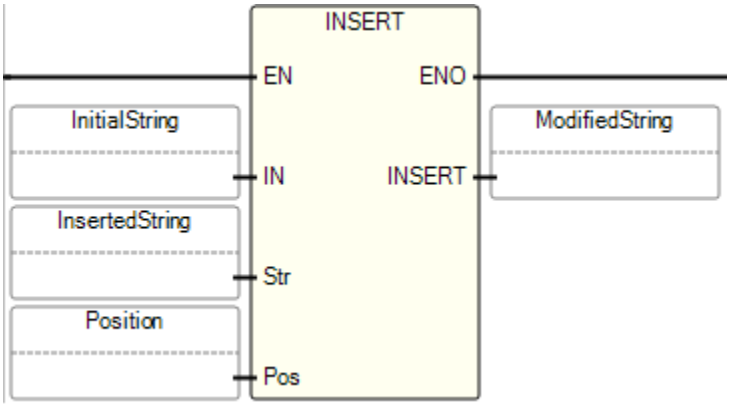
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 在字符串中插入子字符串。 FALSE - 不执行任何操作。 适用于梯形图编程。
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）。
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”。 适用于梯形图编程。

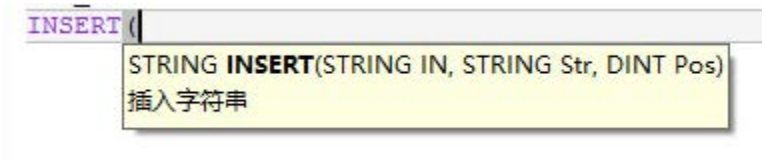
INSERT 功能块图示例



INSERT 梯形图示例



INSERT 结构化文本示例




```
1  Position := 3;
2  ModifiedString := INSERT(InitialString, InsertedString, Position);
```

(* 与之等效的 ST: *)

MyName := INSERT ('Mr JONES', 'Frank ', 4);

(* MyName 为“Mr Frank JONES” *)

结果



LEFT（提取字符串的左侧）

从字符串左侧提取字符。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

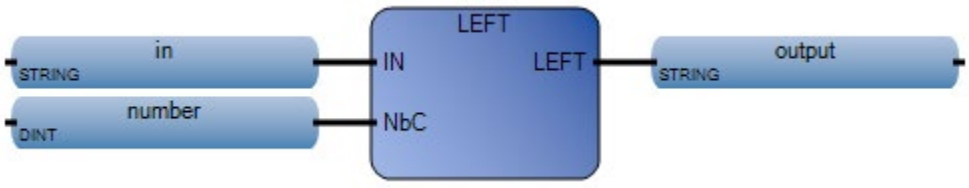
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



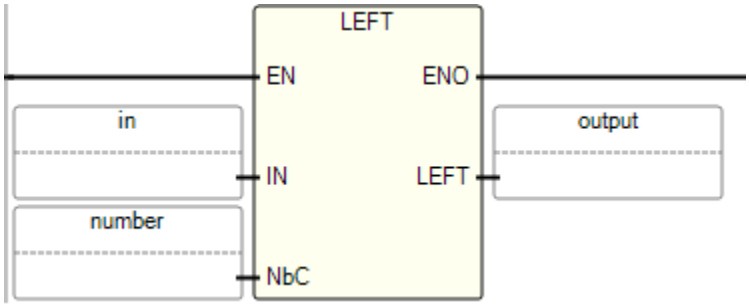
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 从字符串左侧计算字符数。 FALSE - 不执行任何操作。 适用于梯形图编程。
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”。 适用于梯形图编程。

LEFT 功能块图示例



LEFT 梯形图示例



LEFT 结构化文本示例




```
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



MID（提取字符串的中间）

从字符串中间提取字符。使用提供的字符位置和数量计算所需的字符串部分。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



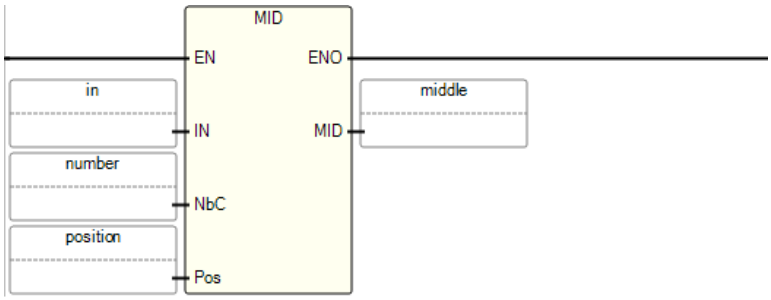
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 生成字符串的一部分。 FALSE - 不执行任何生成操作。 适用于梯形图编程。
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"。 适用于梯形图编程。

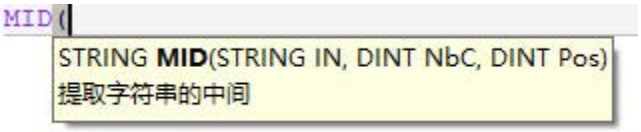
MID 功能块图示例



MID 梯形图示例



MID 结构化文本示例




```
1  number := 3;
2  position := 2;
3  middle := MID(in, number, position);
```

(* 与之等效的 ST: *)

sub_string := MID ('参考页数 efgh', 2, 4);

(* sub_string 为“de” *)

结果



MLEN（字符串长度）

计算字符串的长度。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

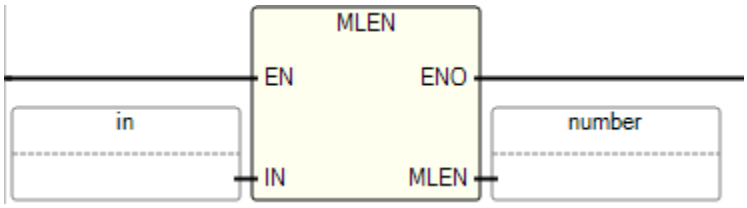
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 计算字符串的长度。 FALSE - 不执行任何操作。 适用于梯形图编程。
IN	输入	STRING	任何字符串。

参数	参数类型	数据类型	描述
MLEN	输出	DINT	IN 字符串中的字符数。
ENO	输出	BOOL	启用"输出"。 适用于梯形图编程。

MLEN 功能块图示例



MLEN 梯形图示例



MLEN 结构化文本示例




```
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

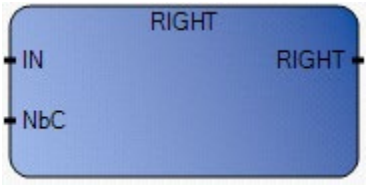


RIGHT（提取字符串的右侧）

从字符串右侧提取字符。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



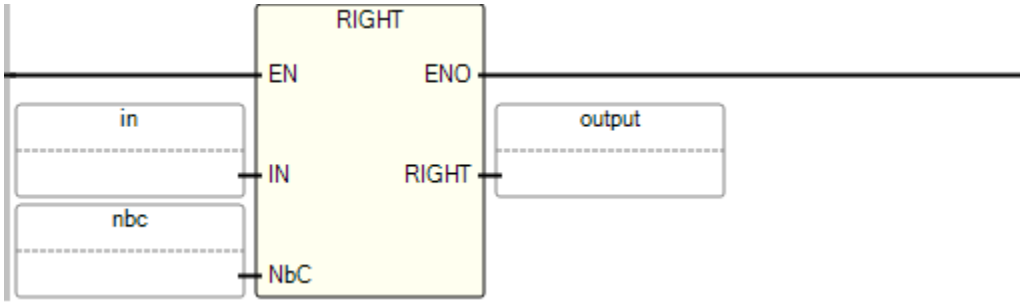
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 从字符串右侧提取指定数量的字符。 FALSE - 不执行任何操作。 适用于梯形图编程。
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”。 适用于梯形图编程。

RIGHT 功能块图示例



RIGHT 梯形图示例



RIGHT 结构化文本示例




```
1  nbc := 2;
2  output := RIGHT(in, nbc);
```

(* 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

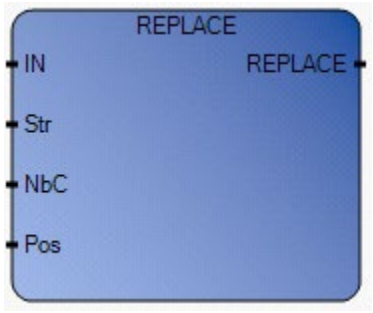


REPLACE（替换子字符串）

将字符串的一部分替换为新的字符集。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

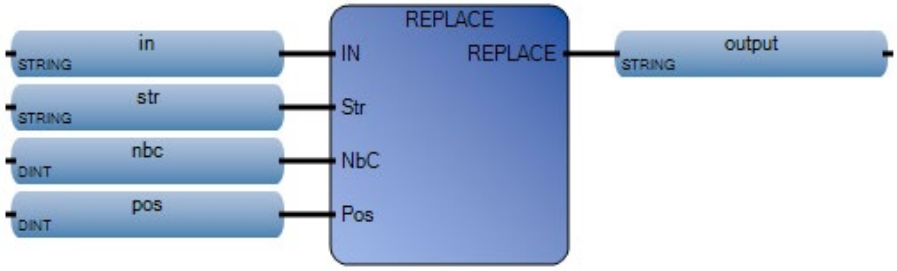
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



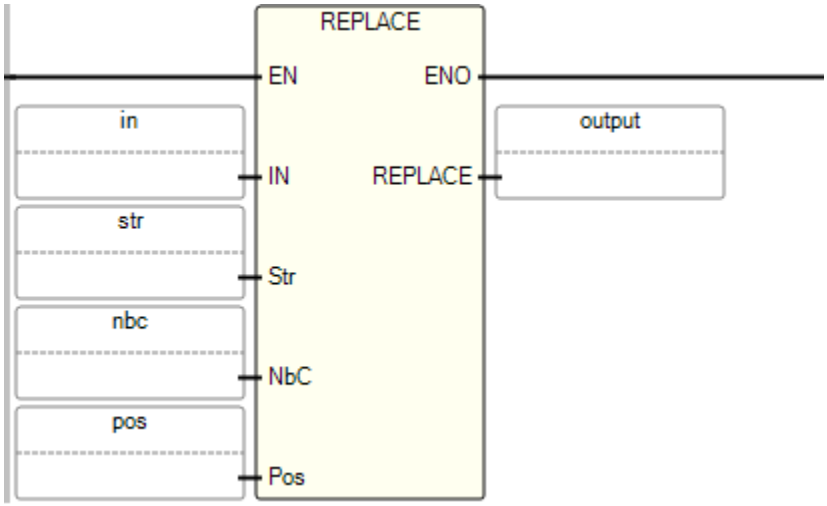
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	函数启用。 TRUE - 将字符串的一部分替换为新字符。 FALSE - 不执行任何操作。 适用于梯形图编程。
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）。
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”。 适用于梯形图编程。

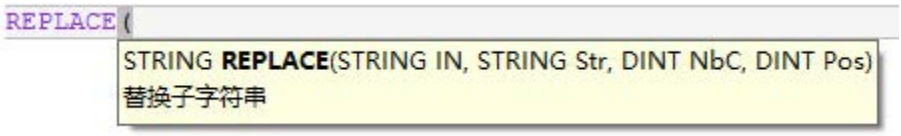
REPLACE 功能块图示例



REPLACE 梯形图示例



REPLACE 结构化文本示例




```
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


变量监视
用户全局变量 - Micro850
局部变量 - RA_REPLACE_LD
系统变量- Micro850
I/O - Micro850
名称
逻辑值
实际值
初始值
锁定
数据类型
维度
in
abcdef
不可用
STRING
nbc
4
不可用
DINT
output
aghif
不可用
STRING
pos
2
不可用
DINT
str
ghi
不可用
STRING
关闭(C)

计时器指令

计时器指令用于根据时间控制操作。

指令	描述
TOF 参考页数 599	**关**闭延时计时。将内部计时器增加至指定值。
TON 参考页数 601	**打**开延时计时。将内部计时器增加至指定值。
TONOFF 参考页数 603	**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 605	**脉**冲计时。在上升沿时，将内部计时器增加至指定值。
RTO 参考页数 607	**保**持时间。当输入处于活动状态时增加内部计时器，但当输入变为不活动状态时不复位内部计时器。
DOY 参考页数 609	**如**果实时时钟的值位于"年时间"设置范围内，则开启输出。
TDF 参考页数 611	**计**算 TimeA 和 TimeB 之间的时间差。
TOW 参考页数 613	**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 参考页数 599 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（计时器，关闭延时）

将内部计时器增加至指定值。

操作详细信息：

- 如果将 EN 参数用于本指令块，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。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

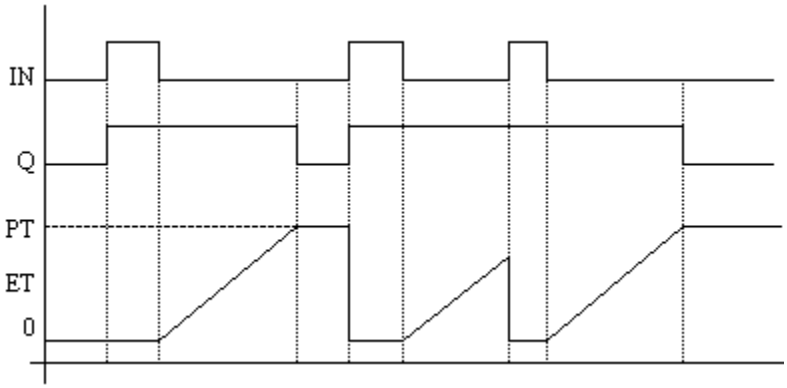
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



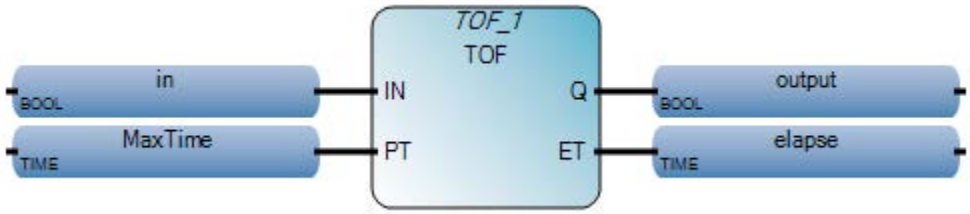
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	输入控制。 TRUE - 检测到下降沿，内部计时器开始递增。 FALSE - 检测到上升沿，停止并复位内部计时器。
PT	输入	TIME	最大编程时间。 请参见 Time 数据类型。
Q	输出	BOOL	TRUE - 总时间未过。 FALSE - 总时间已过。
ET	输出	TIME	当前已过去的时间。值的可能范围从 0ms 到 1193h2m47s294ms。

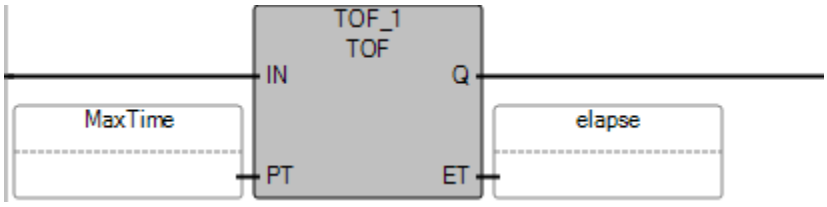
TOF 时间图



TOF 功能块图示例



TOF 梯形图示例



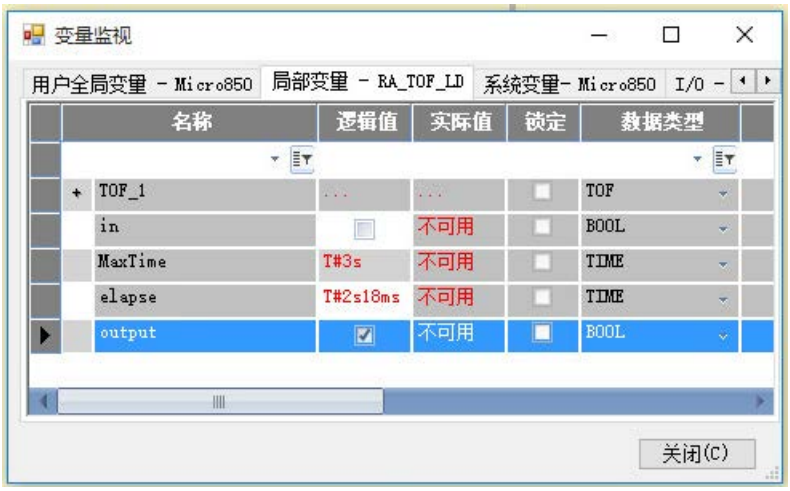
TOF 结构化文本示例

```
TOF_1(
```

void **TOF_1**(BOOL **IN**, TIME **PT**)
类型： TOF, 关闭延迟计时

```
1  MaxTime := T#3s;
2  TOF_1(in, MaxTime);
3  output := TOF_1.Q;
4  elapse := TOF_1.ET;
```

结果



TON（计时器，打开延时）

将内部计时器增加至指定值。

操作详细信息：

- 不要使用跳转跳过梯形图 (LD) 中的 TON 指令块。如果使用跳转，则 TON 计时器将在经过时间过去后继续。
- 例如：梯级 1 包含一个跳转；梯级 2 包含一个 TON 指令块，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；延时时间为 30 秒，而非延时时间中定义的 10 秒。
- 如果将 EN 参数用于 TON，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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

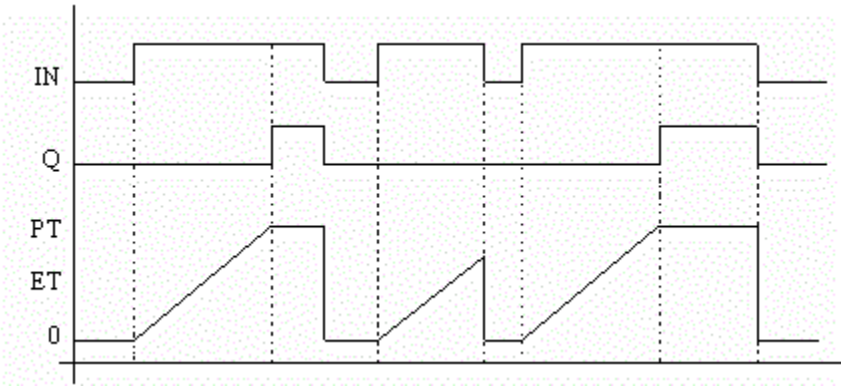
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



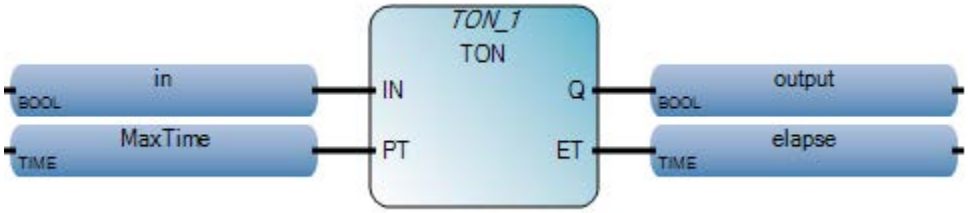
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	输入控制。 TRUE - 如果是上升沿，内部计时器开始递增。 FALSE- 如果是下降沿，停止并复位内部计时器。
PT	输入	TIME	使用时间数据类型定义最大编程时间。
Q	输出	BOOL	TRUE - 编程时间已过。 FALSE - 编程时间未过。
ET	输出	TIME	当前已过去的时间。值的可能范围从 0ms 到 1193h2m47s294ms。

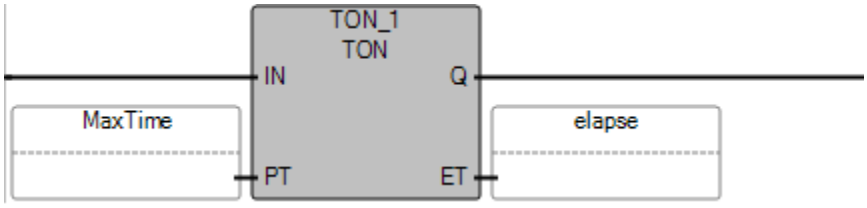
TON 时间图



TON 功能块图示例



TON 梯形图示例

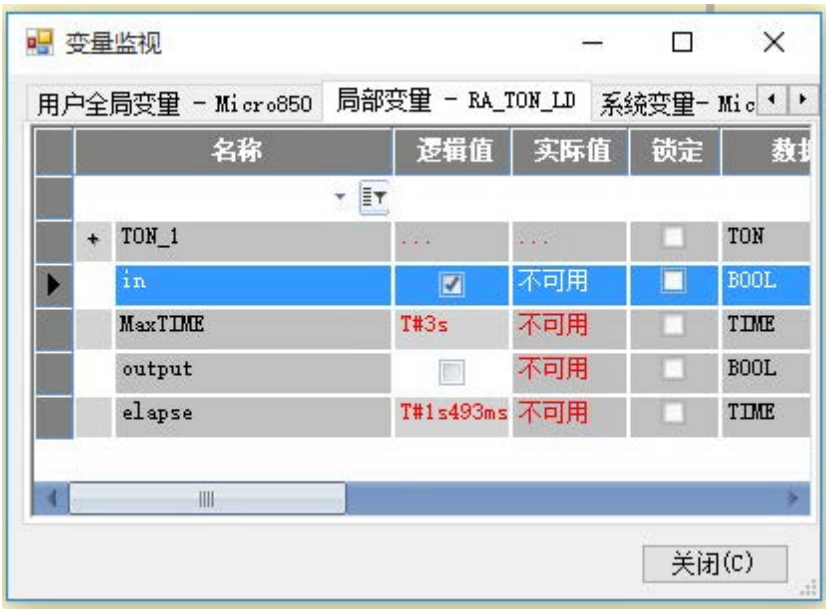


TON 结构化文本示例




```
1  MaxTime := T#3s;
2  TON_1(in, MaxTime);
3  output := TON_1.Q;
4  elapse := TON_1.ET;
```

结果



TONOFF（时间延迟，打开，关闭）

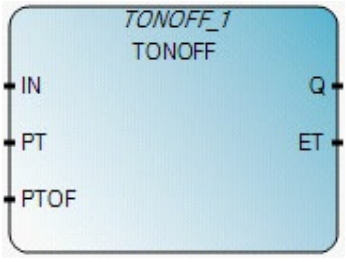
延迟打开 TRUE 梯级上的输出，然后延迟关闭 FALSE 梯级上的输出。

操作详细信息：

- 如果将 EN 参数用于 TONOFF，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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

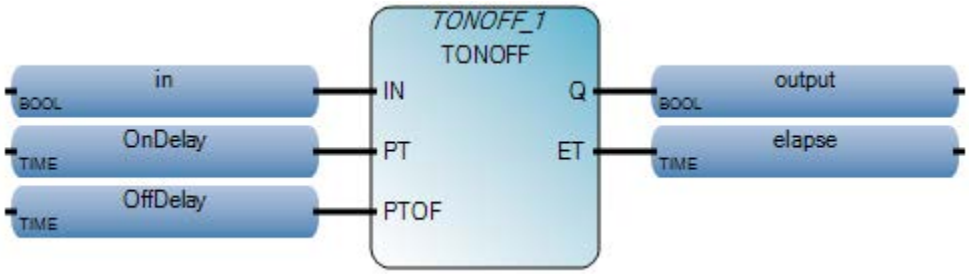
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



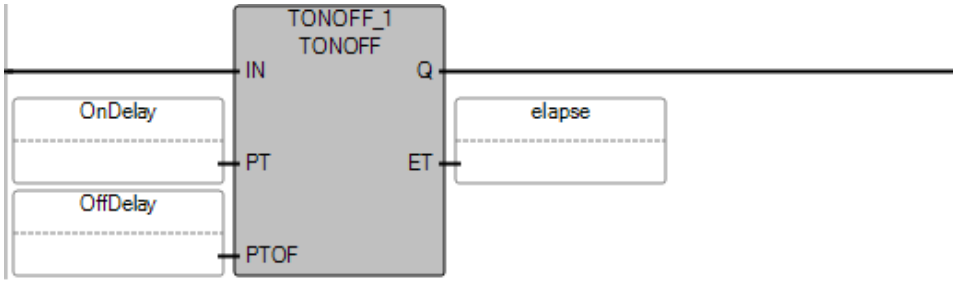
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	输入控制。 TRUE - 检测到上升沿（IN 从 0 变为 1）： • 启动打开延时计时器 (PT)。 • 如果编程关断延时 (PTOF) 未过，重启打开延时 (PT) 计时器。 FALSE - 检测到下降沿（IN 从 1 变为 0）： • 如果编程打开延时时间 (PT) 未过，停止 PT 计时器并复位 ET。 • 如果编程打开延时时间 (PT) 已过，启动关闭延时计时器 (PTOF)。
PT	输入	TIME	使用时间数据类型定义打开延时时间设置。
PTOF	输入	TIME	使用时间数据类型定义关闭延时时间设置。
Q	输出	BOOL	TRUE - 编程打开延时时间已过，而编程关闭延时时间未过。
ET	输出	TIME	当前已过去的时间。值的可能范围从 0ms 到 1193h2m47s294ms。 如果编程的打开延时时间已经过，且关闭延时计时器未启动，则已经过时间 (ET) 仍为打开延时 (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。

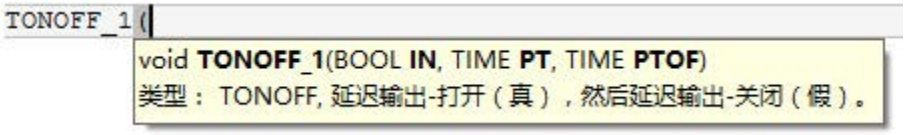
TONOFF 功能块图示例



TONOFF 梯形图示例

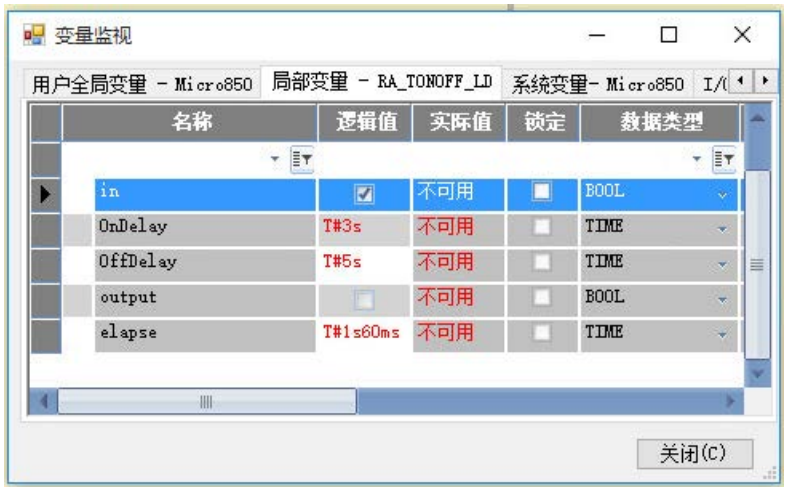


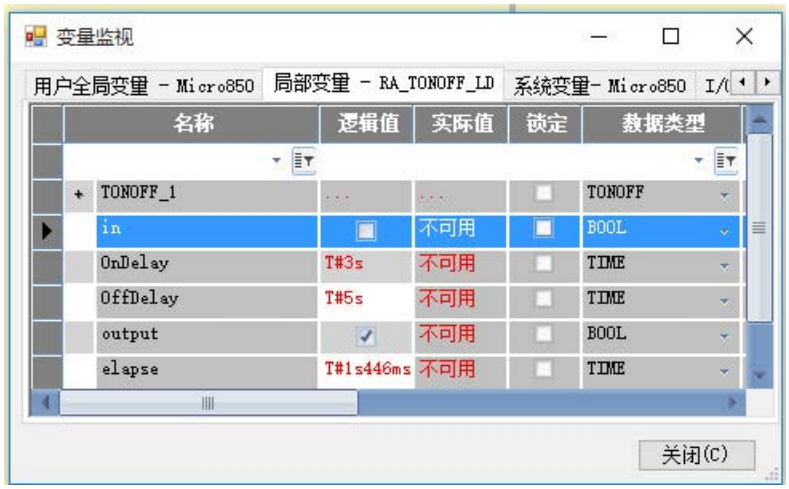
TONOFF 结构化文本示例



```
1  OnDelay := T#3s;
2  OffDelay := T#5s;
3  TONOFF_1(in, OnDelay, OffDelay);
4  output := TONOFF_1.Q;
5  elapse := TONOFF_1.ET;
```

结果





TP（脉冲计时）

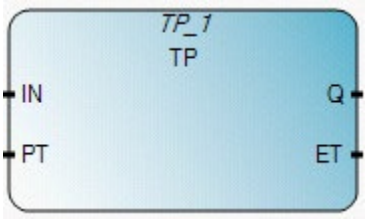
在上升沿时，将内部计时器增加至指定值。如果计时器的时间已过，将复位内部时间。

操作详细信息：

- 如果将 EN 参数用于 TP，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。

支持的语言："功能块图"、"梯形图"和"结构化文本"。

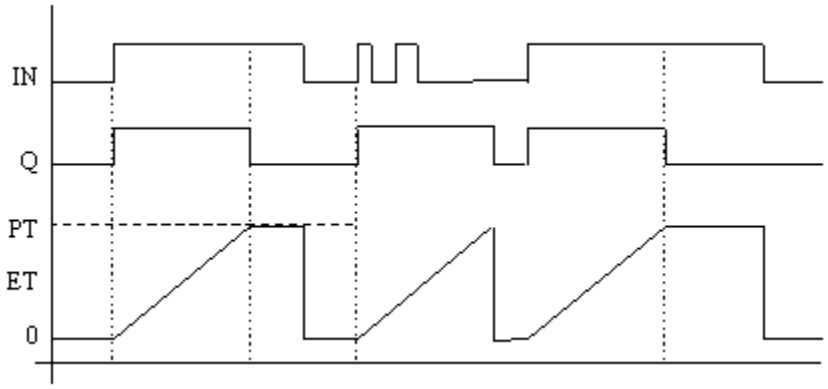
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



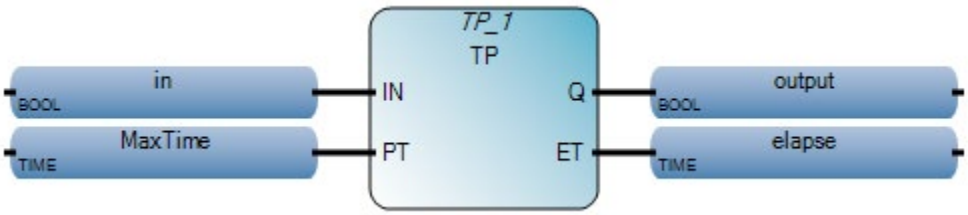
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	TRUE - 如果是上升沿，内部计时器开始递增（如果尚未递增）。 FALSE - 如果计时器的时间已过，将复位内部计时器。 计数期间对 IN 的任何更改都不生效。
PT	输入	TIME	使用时间数据类型定义最大编程时间。
Q	输出	BOOL	TRUE - 计时器正在计时。 FALSE - 计时器没有计时。
ET	输出	TIME	当前已过去的时间。 值的可能范围从 0ms 到 1193h2m47s294ms。

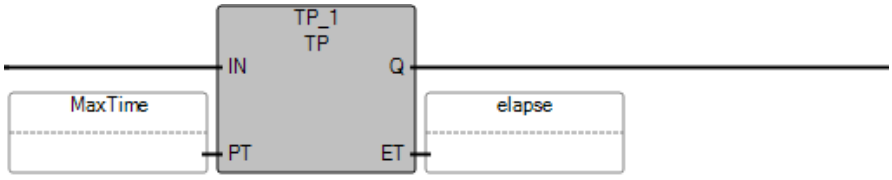
TP 时间图



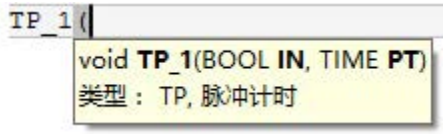
TP 功能块图示例



TP 梯形图示例

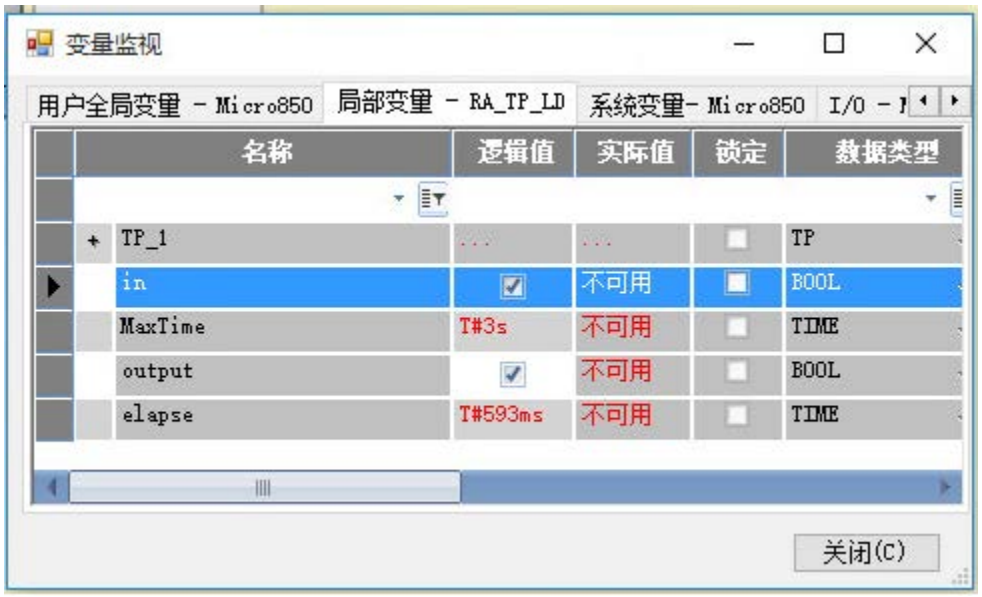


TP 结构化文本示例




```
1  MaxTime := T#3s;
2  TP_1(in, MaxTime);
3  output := TP_1.Q;
4  elapse := TP_1.ET;
```

结果



RTO（保持计时器，打开延时）

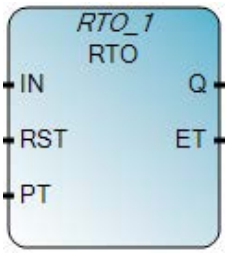
当输入处于活动状态时增加内部计时器，但当输入变为不活动状态时不复位内部计时器。

操作详细信息：

- Micro810 或 Micro820 控制器，RTO 内部计时器不会默认在断电重启过程中保持不变。要保持内部计时器不变，请将保留的配置参数设置为真。
- Micro830 或 Micro850 控制器，RTO 内部计时器在断电重启过程中保持不变。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

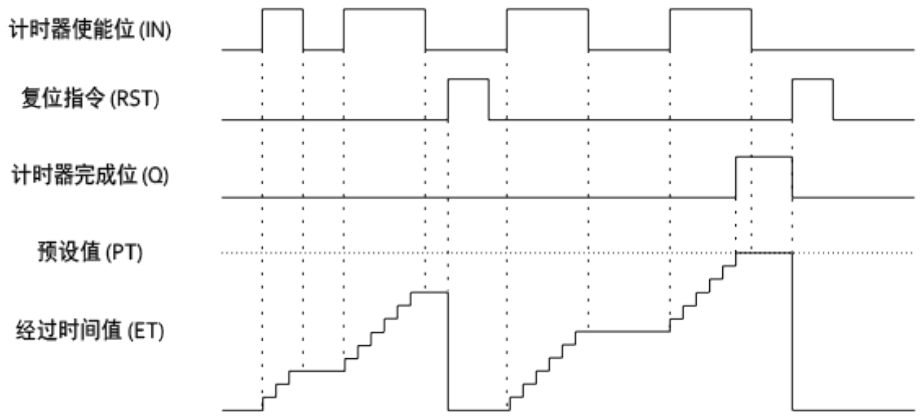
该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



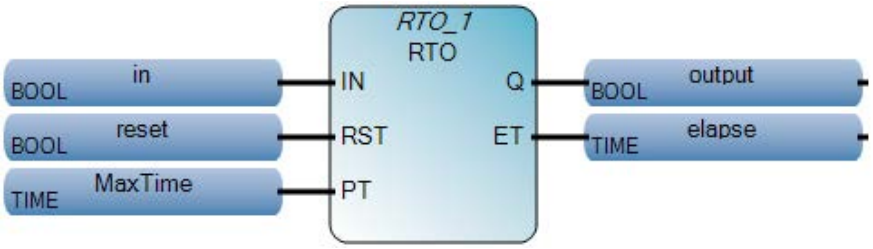
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
IN	输入	BOOL	输入控制。 TRUE - 上升沿，开始增加内部计时器。 FALSE - 下降沿，停止且不复位内部计时器。
RST	输入	BOOL	TRUE - 上升沿，复位内部计时器。 FALSE - 不复位内部计时器。
PT	输入	TIME	编程的最大打开延时时间。使用时间数据类型定义 PT。
Q	输出	BOOL	TRUE - 编程的打开延时时间已经过。 FALSE - 编程的打开延时时间未经过。
ET	输出	TIME	当前已过去的时间。 值的范围从 0ms 到 1193h2m47s294ms。 使用时间数据类型定义 ET。

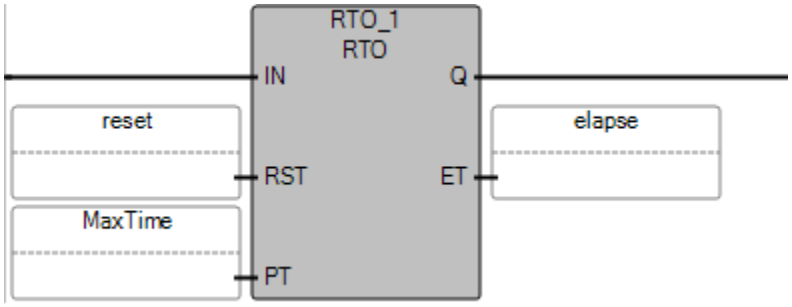
RTO 时间图示例



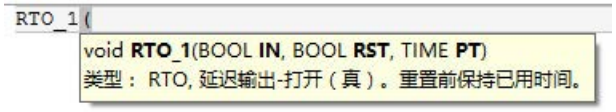
RTO 功能块图示例



RTO 梯形图示例



RTO 结构化文本示例




```
1  MaxTime := T#3s;
2  RTO_1(in, reset, MaxTime);
3  output := RTO_1.Q;
4  elapse := RTO_1.ET;
```

DOY（检查实时时钟的年份）

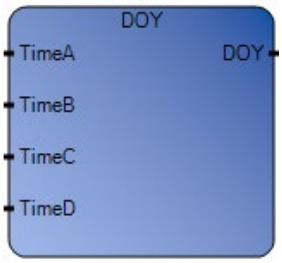
如果实时时钟 (RTC) 的值位于“年时间”设置范围内，则开启输出。

操作详细信息：

- 如果 RTC 不存在，则输出始终为关闭。
- 在“DOYDATA 数据类型”表中指定的有效范围内配置 Time 输入参数。如果将 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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

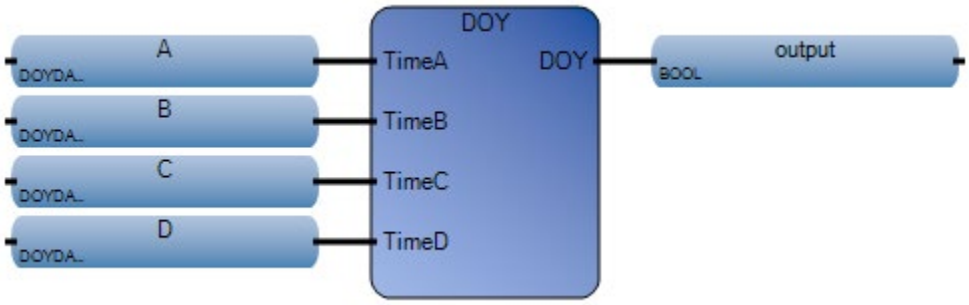
参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行操作。 FALSE - 不执行操作。
TimeA	输入	DOYDATA	通道 A 的年时间设置。 DOYDATA 数据类型用于配置 TimeA。
TimeB	输入	DOYDATA	通道 B 的年时间设置。 DOYDATA 数据类型用于配置 TimeB。
TimeC	输入	DOYDATA	通道 C 的年时间设置。 DOYDATA 数据类型用于配置 TimeC。
TimeD	输入	DOYDATA	通道 D 的年时间设置。 DOYDATA 数据类型用于配置 TimeD。
DOY	输出	BOOL	如果为 TRUE，则实时时钟的值在四个通道任意之一的"年时间"设置范围内。

DOYDATA 数据类型

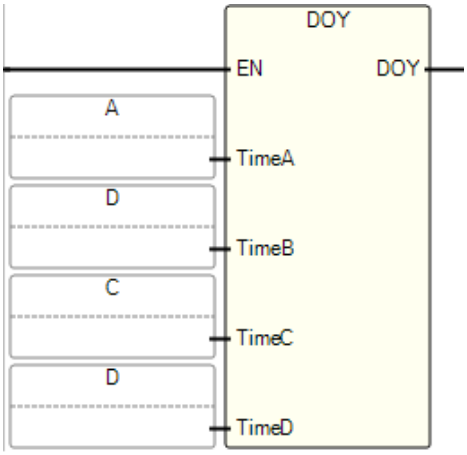
下表描述了 DOYDATA 数据类型。

参数	数据类型	描述
Enable	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）。

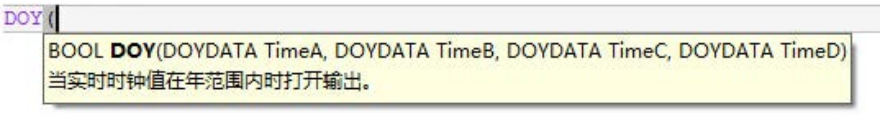
DOY 功能块图示例



DOY 梯形图示例



DOY 结构化文本示例




```
1  output := DOY(A, B, C, D);
```

(* 与之等效的 ST：*)

TESTOUTPUT := DOY(TIMEA1, TIMEB1, TIMEC1, TIMED1) ;

TDF（时间差）

计算 TimeA 和 TimeB 之间的时间差。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



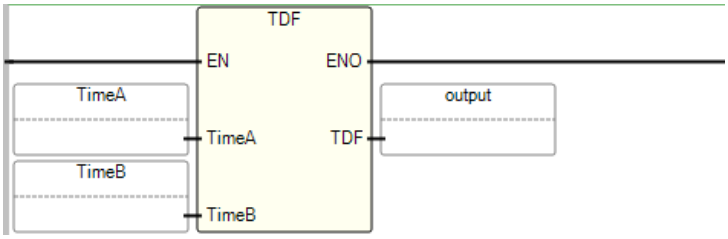
使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 TRUE - 执行当前计算。 FALSE - 不执行任何计算。 适用于梯形图编程。
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"。 适用于梯形图编程。
TDF	输出	TIME	两个时间输入的时间差。 TDF 为名称或 PIN ID

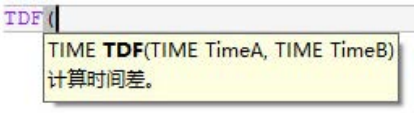
TDF 功能块图示例



TDF 梯形图示例



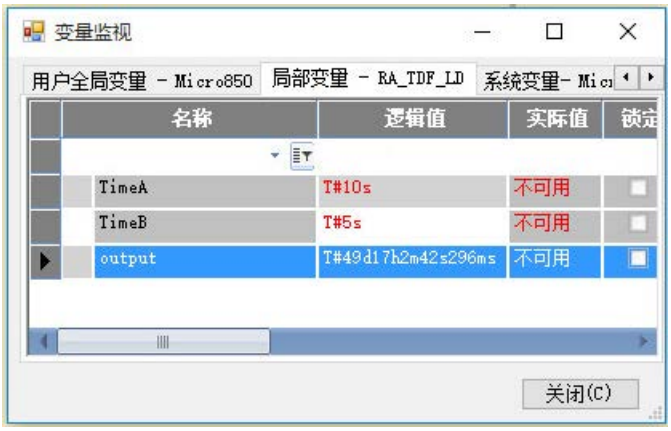
TDF 结构化文本示例




```
1  TimeA := T#10s;
2  TimeB := T#5s;
3  output := TDF(TimeA, TimeB);
```

```
(* 与之等效的 ST： *)
TESTTIMEDIFF := TDF(TESTTIME1, TESTTIME2) ;
```

结果



TOW（实时时钟复选周）

如果实时时钟 (RTC) 的值位于“周时间”设置范围内，则开启输出。

操作详细信息：

- 如果 RTC 不存在，则输出始终为关闭。
- 使用在“TOWDATA 数据类型”中指定的有效范围配置 Time 输入参数。如果将 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。

支持的语言：“功能块图”、“梯形图”和“结构化文本”。

该指令适用于 Micro810、Micro820、Micro830、Micro850、Micro870 控制器和 Micro800 Simulator。



使用此表有助于确定该指令的参数值。

参数	参数类型	数据类型	描述
EN	输入	BOOL	启用指令。 当 EN = TRUE 时，执行该操作。 当 EN = FALSE 时，不执行该操作。
TimeA	输入	TOWDATA	通道 A 的“日时间”设置。 使用 TOWDATA 数据类型定义 TimeA。
TimeB	输入	TOWDATA	通道 B 的“日时间”设置。 使用 TOWDATA 数据类型定义 TimeB。

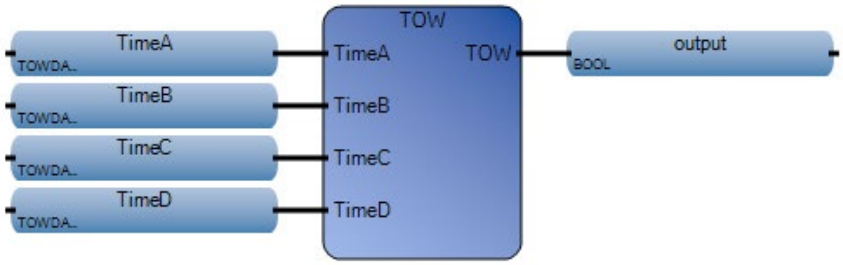
参数	参数类型	数据类型	描述
TimeC	输入	TOWDATA	通道 C 的"日时间"设置。 使用 TOWDATA 数据类型定义 TimeC。
TimeD	输入	TOWDATA	通道 D 的"日时间"设置。 使用 TOWDATA 数据类型定义 TimeD。
TOW	输出	BOOL	如果为 TRUE，则实时时钟的值在四个通道任意之一的"日时间"设置范围内。

TOWDATA 数据类型

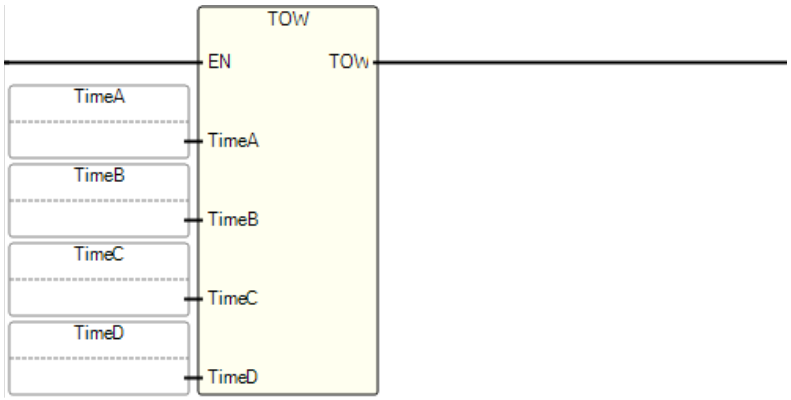
下表描述了 TOWDATA 数据类型：

参数	数据类型	描述
Enable	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）。

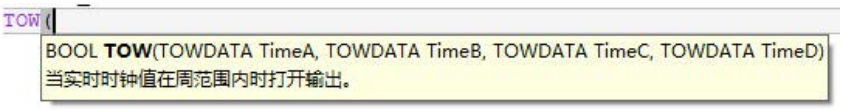
TOW 功能块图示例



TOW 梯形图示例



TOW 结构化文本示例




```
1  output := TOW(TimeA, TimeB, TimeC, TimeD);
```

(* 与之等效的 ST：*)

TESTOUTPUT := TOW(TIMEA, TIMEB, TIMEC, TIMED) ;

结果

名称	逻辑值	实际值	锁定	数据类型
TimeA	...	...		TOWDATA
TimeA.Enable	✓	不可用		BOOL
TimeA.DailyWeekly	✓	不可用		BOOL
TimeA.DayOn	1	不可用		USINT
TimeA.HourOn	10	不可用		USINT
TimeA.MinOn	20	不可用		USINT
TimeA.DayOff	2	不可用		USINT
TimeA.HourOff	15	不可用		USINT
TimeA.MinOff	30	不可用		USINT
TimeB	...	...		TOWDATA
TimeB.Enable	✓	不可用		BOOL
TimeB.DailyWeekly	✓	不可用		BOOL
TimeB.DayOn	2	不可用		USINT
TimeB.HourOn	15	不可用		USINT
TimeB.MinOn	20	不可用		USINT
TimeB.DayOff	3	不可用		USINT
TimeB.HourOff	10	不可用		USINT
TimeB.MinOff	30	不可用		USINT
TimeC	...	...		TOWDATA
TimeC.Enable	✓	不可用		BOOL
TimeC.DailyWeekly	✓	不可用		BOOL
TimeC.DayOn	3	不可用		USINT
TimeC.HourOn	20	不可用		USINT
TimeC.MinOn	10	不可用		USINT
TimeC.DayOff	4	不可用		USINT
TimeC.HourOff	25	不可用		USINT
TimeC.MinOff	55	不可用		USINT
TimeD	...	...		TOWDATA
TimeD.Enable	✓	不可用		BOOL
TimeD.DailyWeekly	✓	不可用		BOOL
TimeD.DayOn	5	不可用		USINT
TimeD.HourOn	10	不可用		USINT
TimeD.MinOn	15	不可用		USINT
TimeD.DayOff	6	不可用		USINT
TimeD.HourOff	35	不可用		USINT
TimeD.MinOff	40	不可用		USINT
output	✓	不可用		BOOL

索引

- 运算符 103

*

* 运算符 91

/

/ 运算符 82

+

+ 运算符 70

<

< 运算符 266
<= 运算符 267
<> 运算符 268

=

= 运算符 259

>

> 运算符 263
>= 运算符 264

1

1Gain 运算符 89

A

ABL 功能块 111
ABS 函数 65
ACB 功能块 117
ACL 功能块 113
ACOS 函数 67
ACOS_LREAL 函数 69
AFI 585
AHL 功能块 115
AND_MASK 函数 133
ANY_TO_BOOL 运算符 279
ANY_TO_BYTE 运算符 280
ANY_TO_DATE 运算符 281
ANY_TO_DINT 运算符 283
ANY_TO_DWORD 运算符 284
ANY_TO_INT 运算符 285
ANY_TO_LINT 运算符 287
ANY_TO_LREAL 运算符 288
ANY_TO_LWORD 运算符 289
ANY_TO_REAL 运算符 290
ANY_TO_SINT 运算符 291
ANY_TO_STRING 运算符 292
ANY_TO_TIME 运算符 294
ANY_TO_UDINT 运算符 295
ANY_TO_UINT 运算符 296
ANY_TO_ULINT 运算符 297
ANY_TO_USINT 运算符 298
ANY_TO_WORD 运算符 300
ARD 功能块 119
ARL 功能块 122
ASCII 函数 695
ASIN 函数 73
ASIN_LREAL 函数 74
ATAN 函数 76
ATAN_LREAL 函数 77
AVERAGE 功能块 303
AWA 功能块 125
AWT 功能块 127

C

CHAR 函数 697
COM_IO_WDOG 177
COP 功能块 304
COS 函数 79
COS_LREAL 函数 80
CTD 功能块 271
CTU 功能块 273
CTUD 功能块 275

D

DERIVATE 功能块 531
DOY 函数 728

E

EXPT 函数 84

F

F_TRIG 功能块 157
FBD 中的重置线圈 47
FBD（功能块图）
  反向接触 51
  反向线圈 45
  梯级，插入 35
  直接接触 50
  直接线圈 44

线圈，使用情况和可用类型 42
脉冲上升沿（正值）触点 51
触点，使用情况和可用类型 48
设置线圈 46
跳转到标签，插入 53
返回，插入 53
重设线圈 47
FFL 533
FFU 544
FIND 函数 701

H

HSC 功能块 316
HSC_SET_STS 功能块 333
HYSTER 功能块 549

I

IIM 功能块 386
INSERT 函数 703
INTEGRAL 功能块 551
IOM 功能块 388
IPIDCONTROLLER 功能块 589

K

KEY_READ 功能块 390

L

LCD 函数 374
LD 图的平行分支 38
LD 语言
键盘快捷键 58
LD（梯形图）
分支 38
反向接触 51
反向线圈 45
梯级，插入 35
直接接触 50
直接线圈 44
线圈，使用情况和可用类型 42
脉冲上升沿（正值）线圈 46
脉冲上升沿（正值）触点 51
脉冲下降沿（负值）线圈 45
脉冲下降沿（负值）触点 52
触点，使用情况和可用类型 48
设置线圈 46
跨线 53
返回语句 53
重设线圈 47
LEFT 函数 705
LIM_ALRM 功能块 61
LIMIT 函数 581
LOG 函数 86

M

MAX 函数 312
MC_AbortTrigger 功能块 456
MC_Halt 功能块 458
MC_Home 功能块 462
MC_MoveAbsolute 功能块 467
MC_MoveRelative 功能块 471
MC_MoveVelocity 功能块 476
MC_Power 功能块 480
MC_Read 488
MC_ReadActualPosition 484
MC_ReadAxisError 功能块 492
MC_ReadBoolParameter 功能块 497
MC_ReadParameter 功能块 500
MC_ReadStatus 功能块 503
MC_Reset 功能块 510
MC_SetPosition 功能块 512
MC_Stop 功能块 515
MC_TouchProbe 功能块 518
MC_WriteBoolParameter 功能块 522
MC_WriteParameter 功能块 525
MID 函数 707
MIN 函数 310
MLEN 函数 709
MMINFO 功能块 395
MOD 函数 88
MODULE_INFO 399
MOV 运算符 89
MSG_CIPGENERIC 功能块 180
MSG_CIPSYMBOLIC 功能块 188
MSG_MODBUS 功能块 192
MSG_MODBUS2 功能块 199
MUX4B 函数 173
MUX8B 函数 169

N

Neg 运算符 93
NOT_MASK 函数 142

O

OR_MASK 函数 143

P

PID 616
PLUGIN_INFO 功能块 411
PLUGIN_READ 功能块 414
PLUGIN_RESET 功能块 416
PLUGIN_WRITE 功能块 418
POW 函数 94

R

R_TRIG 功能块 159
RAND 函数 96
REPLACE 函数 713
RHC 函数 381
RIGHT 函数 711
ROL 函数 145
ROR 函数 147
RPC 函数 383
RS 功能块 160
RTC_READ 功能块 422
RTC_SET 功能块 425
RTO 功能块 726

S

SCALER 功能块 566
SHL 函数 149
SHR 函数 151
SIN 函数 98
SIN_LREAL 99
SOCKET_ACCEPT 636
SOCKET_CREATE 639
SOCKET_DELETE 645
SOCKET_DELETEALL 648
SOCKET_INFO 650
SOCKET_OPEN 657
SOCKET_READ 661
SOCKET_WRITE 666
SQRT 函数 101
SR 功能块 164
STACKINT 功能块 569
STIS 函数 433
SUS 功能块 585
SYS_INFO 功能块 427

T

TAN 函数 104
TAN_LREAL 函数 106
TDF 函数 730
TND 函数 579
TOF 功能块 717
TON 功能块 719
TONOFF 功能块 721
TOW 函数 732
TP 功能块 724
TRIMPOT_READ 功能块 430
TRUNC 函数 108
TTABLE 函数 166

U

UIC 函数 435
UID 函数 437
UIE 函数 439
UIF 函数 441

X

XOR_MASK 函数 153

不

不等于运算符 268

与

与运算符 163

中

中断
STIS 函数 433
UIC 函数 435
UID 函数 437
UIE 函数 439
UIF 函数 441

二

二进制运算
AND_MASK 函数 133
NOT_MASK 函数 142
OR_MASK 函数 143
ROL 函数 145
ROR 函数 147
SHL 函数 149
SHR 函数 151
XOR_MASK 函数 153

以

以下项的指令块说明
ABL 111
ACB 117
ACL 113
AHL 115
ARD 119
ARL 122
AVERAGE 303
AWA 125
AWT 127
COP 304
CTD 271
CTU 273
CTUD 275
DERIVATE 531
F_TRIG 157
HSC_SET_STS 333
HYSTER 549
IIM 386

INTEGRAL 551
IOM 388
IPIDCONTROLLER 589
KEY_READ 390
LIM_ALRM 61
MC_AbortTrigger 456
MC_Halt 458
MC_Home 462
MC_MoveAbsolute 467
MC_MoveRelative 471
MC_MoveVelocity 476
MC_Power 480
MC_ReadAxisError 492
MC_ReadBoolParameter 497
MC_ReadParameter 500
MC_ReadStatus 503
MC_Reset 510
MC_SetPosition 512
MC_Stop 515
MC_TouchProbe 518
MC_WriteBoolParameter 522
MC_WriteParameter 525
MM_INFO 395
MSG_CIPGENERIC 180
MSG_CIPSYMBOLIC 188
MSG_MODBUS 192
MSG_MODBUS2 199
PLUGIN_INFO 411
PLUGIN_READ 414
PLUGIN_RESET 416
PLUGIN_WRITE 418
R_TRIG 159
RST 160
RTC_READ 422
RTC_SET 425
RTO 726
SCALER 566
SRT 164
STACKINT 569
SUS 585
SYS_INFO 427
TOF 717
TON 719
TONOFF 721
TP 724
TRIMPOT_READ 430
高速计数器 316

函

函数

ABS 65
ACOS 67
ACOS_LREAL 69
AND_MASK 133
ASCII 695
ASIN 73
ASIN_LREAL 74
ATAN 76
ATAN_LREAL 77
CHAR 697
COS 79
COS_LREAL 80
DELETE 698
DOY 728
EXPT 84
FIND 701
INSERT 703
LCD 374
LEFT 705
LIMIT 581
LOG 86
MAX 312
MID 707
MIN 310
MLEN 709
MOD 88
MUX4B 173
MUX8B 169
NOT_MASK 142
OR_MASK 143
POW 94
RAND 96
REPLACE 713
RHC 381
RIGHT 711
ROL 145
ROR 147
RPC 383
SHL 149
SHR 151
SIN 98
SIN_LREAL 99
SQRT 101
STIS 433
TAN 104
TAN_LREAL 106
TDF 730
TND 579
TOW 732
TRUNC 108
TTABLE 166
UIC 435
UID 437
UIE 439
UIF 441
XOR_MASK 153

分

分支

插入 38
用于 LD 38

功

功能块

ABL 111
ACB 117
ACL 113
AHL 115
ARD 119
ARL 122
AVERAGE 303
AWA 125
AWT 127
COP 304
CTD 271
CTU 273
CTUD 275
DERIVATE 531
F_TRIG 157
FFL 533
FFU 544
HSC_SET_STS 333
HYSTER 549
IIM 386
INTEGRAL 551
IOM 388
IPIDCONTROLLER 589
KEY_READ 390
LIM_ALRM 61
MC_AbortTrigger 456
MC_Halt 458
MC_Home 462
MC_MoveAbsolute 467
MC_MoveRelative 471
MC_MoveVelocity 476
MC_Power 480
MC_ReadAxisError 492
MC_ReadBoolParameter 497
MC_ReadParameter 500
MC_ReadStatus 503
MC_Reset 510
MC_SetPosition 512
MC_Stop 515
MC_TouchProbe 518
MC_WriteBoolParameter 522
MC_WriteParameter 525
MMINFO 395
MODULE_INFO 399
MSG_CIPGENERIC 180
MSG_CIPSYMBOLIC 188
MSG_MODBUS 192
MSG_MODBUS2 199
PID 616
PLUGIN_INFO 411
PLUGIN_READ 414
PLUGIN_RESET 416
PLUGIN_WRITE 418
R_TRIG 159
RS 160
RTC_READ 422
RTC_SET 425
RTO 726
SCALER 566
SOCKET_ACCEPT 636
SOCKET_CREATE 639
SOCKET_DELETE 645
SOCKET_DELETEALL 648
SOCKET_INFO 650
SOCKET_OPEN 657
SOCKET_READ 661
SOCKET_WRITE 666
SR 164
STACKINT 569
SUS 585
SYS_INFO 427
TOF 717
TON 719
TONOFF 721
TP 724
TRIMPOT_READ 430
命名和参数 20
调用 20
高速计数器 316

参

参数

用于功能块 20

反

反向

FBD 中的线圈 45
FBD 中的触点 51
接触 51
线圈 45

命

命名规范

功能块 20

大

大于或等于运算符 264
大于运算符 263

字

字符串操作

ASCII 函数 695
CHAR 函数 697
DELETE 函数 698
FIND 函数 701
INSERT 函数 703

LEFT 函数 705
MID 函数 707
MLEN 函数 709
REPLACE 函数 713
RIGHT 函数 711

小

小于或等于运算符 267
小于运算符 266

布

布尔
MUX4B 函数 173
MUX8B 函数 169
TTABLE 函数 166
布尔运算
F_TRIG 功能块 157
R_TRIG 功能块 159
RS 功能块 160
SR 功能块 164
与运算符 163
异或运算符 163
或运算符 162
非运算符 164

异

异或运算符 163

或

或运算符 162

指

指令 31
指令块说明 180, 188, 199, 303, 304

接

接触
FBD 中的使用情况和可用类型 48
FBD 中的反向 51
FBD 中的正向 50
FBD 中的脉冲上升沿（正值） 51
使用情况和可用类型 48
反向 51
并联，插入 48
插入 48
更改类型 48
直接 50
脉冲上升沿（正值） 51
脉冲下降沿（正值） 52

插

插入
梯级（FBD 单元） 35
梯级标签 35
跳转到标签（FBD 单元） 53
返回（FBD 单元） 53

数

数据操作
AVERAGE 功能块 303
COP 功能块 304
MAX 函数 312
MIN 函数 310
数据转换
ANY_TO_BOOL 运算符 279
ANY_TO_BYTE 运算符 280
ANY_TO_DATE 运算符 281
ANY_TO_DINT 运算符 283
ANY_TO_DWORD 运算符 284
ANY_TO_INT 运算符 285
ANY_TO_LINT 运算符 287
ANY_TO_LREAL 运算符 288
ANY_TO_LWORD 运算符 289
ANY_TO_REAL 运算符 290
ANY_TO_SINT 运算符 291
ANY_TO_STRING 运算符 292
ANY_TO_TIME 运算符 294
ANY_TO_UDINT 运算符 295
ANY_TO_UINT 运算符 296
ANY_TO_ULINT 运算符 297
ANY_TO_USINT 运算符 298
ANY_TO_WORD 运算符 300

时

时间
DOY 函数 728
RTO 功能块 726
TDF 函数 730
TOF 功能块 717
TON 功能块 719
TONOFF 功能块 721
TOW 函数 732
TP 功能块 724

标

标签
梯级，插入 35

梯

梯形图 (LD)

指令 31
语言 31

梯级
功能块图，插入 35
插入 35
标签，插入 35

模

模具
可用于 FBD 的线圈类型 42
可用于 LD 图的线圈类型 42
可用于 LD 图的触点元素 48
梯级，插入到 LD 容器中 35

比

比较运算
不等于运算符 268
大于或等于运算符 264
大于运算符 263
小于或等于运算符 267
小于运算符 266
等于运算符 259

流

流程控制
DERIVATE 功能块 531
HYSTER 功能块 549
INTEGRAL 功能块 551
IPIDCONTROLLER 功能块 589
LFL（LIFO 加载） 557
LFU（LIFO 卸载） 560
LIMIT 函数 581
STACKINT 功能块 569

直

直接
FBD 中的线圈 44
FBD 中的触点 50
接触 50
线圈 44

相

相乘运算符 91
相减运算符 103
相加运算符 70
相除运算符 82

程

程序控制
SUS 功能块 585
TND 函数 579

等

等于运算符 259

算

算术
ABS 函数 65
ACOS 函数 67
ACOS_LREAL 函数 69
ASIN 函数 73
ASIN_LREAL 函数 74
ATAN 函数 76
ATAN_LREAL 函数 77
COS 函数 79
COS_LREAL 函数 80
EXPT 函数 84
LOG 函数 86
MOD 函数 88
MOV 运算符 89
Neg 运算符 93
POW 函数 94
RAND 函数 96
SIN 函数 98
SIN_LREAL 99
SQRT 函数 101
TAN 函数 104
TAN_LREAL 函数 106
TRUNC 函数 108
相乘运算符 91
相减运算符 103
相加运算符 70
相除运算符 82

算术运算
1Gain 运算符 89

线

线圈
FBD 中的反向类型 45
FBD 中的正向类型 44
FBD 中的置位 46
FBD 中的重置 47
FBD 的使用情况和可用类型 42
使用情况和可用类型 42
反向类型 45
在 FBD 中插入 42
在 FBD 更改类型 42
并联，在 FBD 中插入 42
并联，插入 42
插入 42

更改类型 42
直接类型 44
脉冲上升沿（正值） 46
脉冲下降沿（正值） 45
设置 46
重设 47

脉

脉冲上升沿（正值）
FBD 中的触点 51
接触 51
线圈 46
脉冲下降沿（负值）
接触 52
线圈 45

警

警报
LIM_ALRM 功能块 61

计

计数器
CTD 功能块 271
CTU 功能块 273
CTUD 功能块 275

语

语言
梯形图 (LD) 31

调

调用
功能块 20

跨

跨线
功能块图，插入 53
梯形图，插入 53
梯级，插入 35
用于 LD 图 53

输

输入/输出
HSC 功能块 316
HSC_SET_STS 功能块 333
IIM 功能块 386
IOM 功能块 388
KEY_READ 功能块 390
LCD 函数 374
MMINFO 功能块 395
PLUGIN_INFO 功能块 411
PLUGIN_READ 功能块 414
PLUGIN_RESET 功能块 416
PLUGIN_WRITE 功能块 418
RHC 函数 381
RPC 函数 383
RTC_READ 功能块 422
RTC_SET 功能块 425
SYS_INFO 功能块 427
TRIMPOT_READ 功能块 430

运

运动
MC_AbortTrigger 功能块 456
MC_Halt 功能块 458
MC_Home 功能块 462
MC_MoveAbsolute 功能块 467
MC_MoveRelative 功能块 471
MC_MoveVelocity 功能块 476
MC_Power 功能块 480
MC_ReadAxisError 功能块 492
MC_ReadBoolParameter 功能块 497
MC_ReadParameter 功能块 500
MC_ReadStatus 功能块 503
MC_Reset 功能块 510
MC_SetPosition 功能块 512
MC_Stop 功能块 515
MC_TouchProbe 功能块 518
MC_WriteBoolParameter 功能块 522
MC_WriteParameter 功能块 525
运动控制功能块 445
运算符
1Gain 89
AND 163
ANY_TO_BOOL 279
ANY_TO_BYTE 280
ANY_TO_DATE 281
ANY_TO_DINT 283
ANY_TO_DWORD 284
ANY_TO_INT 285
ANY_TO_LINT 287
ANY_TO_LREAL 288
ANY_TO_LWORD 289
ANY_TO_REAL 290
ANY_TO_SINT 291
ANY_TO_STRING 292
ANY_TO_TIME 294
ANY_TO_UDINT 295
ANY_TO_UINT 296
ANY_TO_ULINT 297
ANY_TO_USINT 298
ANY_TO_WORD 300
MOV 89

NOT 164
OR 162
XOR 163
不等于 268
取反 93
大于 263
大于或等于 264
小于 266
小于或等于 267
相乘 91
相减 103
相加 70
相除 82
等于 259
SCALER 566

返

返回符号
插入，FBD 单元 53
返回语句
插入 53
用于 LD 图 53

通

通信
ABL 功能块 111
ACB 功能块 117
ACL 功能块 113
AHL 功能块 115
ARD 功能块 119
ARL 功能块 122
AWA 功能块 125
AWT 功能块 127
MSG_CIPGENERIC 功能块 180
MSG_CIPSYMBOLIC 功能块 188
MSG_MODBUS 功能块 192
MSG_MODBUS2 功能块 199

重

重设
线圈 47

键

键盘快捷键
LD 语言 58

非

非运算符 164

高

高级控制块

Rockwell Automation 支持

可以使用以下资源访问支持信息。

技术支持中心	在操作说明视频、常见问题解答、聊天、用户论坛和产品通知更新中寻找帮助。	rok.auto/support
知识库	访问知识库文章。	rok.auto/knowledgebase
当地技术支持电话号码	查找您所在国家/地区的电话号码。	rok.auto/phonesupport
文献库	查找安装说明、手册、小册子和技术数据出版物。	rok.auto/literature
产品兼容性和下载中心（Product Compatibility and Download Center, PCDC）	获取用于确定产品如何交互的帮助，查看特色和功能，并查找相关的固件。	rok.auto/pcdc

文档反馈

您的意见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，请完成此表：rok.auto/docfeedback。

报废电子电气设备指令（Waste Electrical and Electronic Equipment, WEEE）



在使用寿命结束时，该设备应与任何未分类的城市垃圾分开收集。

Rockwell Automation 的网站 rok.auto/pec 中提供最新的产品环境信息。

Allen-Bradley, expanding human possibility, Logix, Rockwell Automation, and Rockwell Software are trademarks of Rockwell Automation, Inc.

EtherNet/IP is a trademark of ODVA, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Rockwell Otomayson Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenkÖy, İstanbul, Tel: +90 (216) 5698400 EEE YÖnetmeliğine Uygundur

rockwellautomation.com — expanding human possibility®

美洲地区：罗克韦尔自动化，南二大街1201号，密尔沃基市，WI 53204-2496 美国，电话：(1) 414.382.2000
欧洲/中东/非洲：罗克韦尔自动化，NV, Pegasus Park, De Kleetlaan 12a, 1831布鲁塞尔，比利时，电话：(32) 2 663 0600
亚太地区：罗克韦尔自动化SEA Pte Ltd., 2 Corporation Road, #04-05, Main Lobby, Corporation Place, 新加坡 618494，电话：(65) 6510-6608
中国总部：上海市徐汇区虹梅路1801号宏业大厦，邮编：200233，电话：(86 21) 6128 8888
客户服务电话：400 620 6620（中国地区） +852 2887 4666（香港地区）

Rockwell Automation 出版号 2080-RM001M-ZH-E - 2025 年 5 月
取代出版号 2080-RM001L-ZH-E - 2023 年 11 月

Copyright © 2025 Rockwell Automation Technologies, Inc. 保留所有权利。美国印刷。